БЪЛГАРСКА АКАДЕМИЯ НА НАУКИТЕ

ИНСТИТУТ ПО ИНФОРМАЦИОННИ И КОМУНИКАЦИОННИ ТЕХНОЛОГИИ

Секция „Йерархични системи”

СТАНИСЛАВ ДИМИТРОВ ДИМИТРОВ

ПРИЛОЖЕНИЕ НА ЗАКОНИТЕ НА ТЕОРИЯТА НА УПРАВЛЕНИЕТО В ПРОГРАМНИ СИСТЕМИ

ДИСЕРТАЦИЯ

за присъждане на образователна и научна степен „доктор”

по специалност 02.21.10 „Приложение на принципите и методите на кибернетиката в различни области на науката”

по научно направление 5.2 „Електротехника, електроника и автоматика”

Научен ръководител:
Проф. д.т.н. Тодор Стоилов

СОФИЯ
2016

СЪДЪРЖАНИЕ

УВОД

Широкото използване на различни компютърни конфигурации за достъп до разнообразни уеб сайтове, предлагащи уеб услуги, генерира изключително натоварен мрежови трафик в устройствата, свързващи клиентите и доставчиците на услуги. Това на свой ред натоварва компонентите, изграждащи машината, на която се поддържат интернет-страниците. Сървърът е система, в която протичат много задачи едновременно и неговото обслужване представлява сложен процес на управление.

Разбираемо, подобно интензивно използване на компютърните и комуникационни системи причинява липса на гарантирано време за обслужване на поставените им задачи и прави статичните системи все по-неприложими. Това изисква да се проектират и прилагат техники, които позволяват динамика и гъвкавост по отношение на промени в натоварването и използването на ресурсите.

В този контекст подходи, основани на теория на автоматичното управление за моделиране, анализиране и проектиране на вградени компютри и комуникационни системи, се превръщат в обещаваща платформа за контролиране на променливостта на големи и сложни системи, работещи в реално време. Системите с обратна връзка стават чест обект на изследване с оглед на проблеми като адаптиране на изчисления в реално време, гарантиране на производителността при възникнали смущения и ефективно планиране на ресурсите. В тази връзка, модели от теорията на автоматичното управление се прилагат за описание на поведението, за контролиране на времето за реакция, както и за повлияване оптималността, устойчивостта и адаптацията на управлението на реални системи. Всичко това задава ***актуалността и значимостта*** на дисертационния труд, който ще се съсредоточи именно върху приложимостта на принципи от теория на автоматичното управление по отношение на програмни системи. Последното представлява ***обект*** на изследване в този текст. Съответно ***предмет*** на дисертационния труд е управлението на компютърни системи като сложни системи, следвайки постулатите на теория на автоматичното управление, с цел подобряване на тяхната производителност.

В рамките на тази конкретна теоретична перспектива, особен интерес за настоящото изследване ще представлява връзката между зададените задачи от потребителите и използваемостта на хардуерните компоненти.

Като цяло ***целта*** на изследването в дисертационния труд е да се разработят и приложат модели за управление от теория на управлението при функционирането на сложни програмни системи.

За постигането ѝ се дефинират и решават следните ***задачи***:

- ✓ Да се разработи функционален модел на компютърната система при различни входни въздействия;
- ✓ Да се реализира управление в изследваната компютърна система и да се синтезира управление чрез настройка на TCP протокола, уеб сървъра Apache и операционната система;
- ✓ Да се приложат методи за идентификация на системи и да се определят динамичните характеристики на компютърната система при управление на програмата за уеб сървър;
- ✓ Да се изследва ефективността на приложените настройки за управлението системата чрез показателите на качеството.
- ✓ Да се синтезират регулатори за управление на изследваната система и да се оценят качествата на затворена система за управление;

Работната ***хипотеза*** на дисертационното изследване е, че третирането на компютърни системи като обект на управление в съответствие с принципите на теория на автоматичното управление, ще доведе до по-добра производителност и съответно до по-високо качество на обслужване на потребителите.

В хода на проверката на тази хипотеза, са проведени експерименти с конкретно софтуерно приложение за обслужване на потребители – платформата на списание „Автоматика и информатика“. Чрез симулации, касаещи промени в конфигурационния файл и настройки, са изследвани показателите за производителност – време за отговор, латентност, бързодействие по отношение на потреблението на различни видове файлове (текст, аудио, видео и база данни).

Така, в изпълнение на горепоставените цел и задачи, са използвани основни положения от теория на автоматичното управление (ТАУ), реализирани в следната ***методология***:

- Генериране на входящи въздействия чрез симулиране на потребители на информационна система с цел получаване на изходящи сигнали за работата на системата;

- Прилагане на методи за идентификация с цел събиране на данни за работата на програмната система;
- Използване на статистически методи за анализ на измерените изходни данни;
- Прилагане на управление с обратна връзка на програмна система с цел проверка на възможностите за подобряване обслужването на потребители на софтуерното приложение;
- Прилагане метода на контролните карти при измерване качеството на показателите на производителността на програмната система;
- Прилагане на нелинеен модел за описание поведението на системата при подобрение на обслужването на потребителите на софтуерното приложение.

В хода на ***експерименталната работа*** са използвани:

- Генератори на входни въздействия Jmeter и Apache Bench;
- Инструмент SAR за наблюдение на операционната система Linux;
- MATLAB – за статистически анализ на данни и извеждане на динамични модели, за реализиране на симулация на нелинейни модели и за настройка на регулаторите на компютърната система;
- Linux програмата PS и език за програмиране AWK за реализиране на управлението за обратна връзка.

Информационната система е реализирана чрез софтуерни продукти от тип “open source”, като например програмен език PHP, MySQL.

Очакваният резултат от проведените експерименти в рамките на дисертационното изследване е да се потвърди първоначалното предположение за приложимостта на теория на автоматичното управление по отношение на програмни системи и да се наблюдава подобрение в показателите за производителност (време за отговор и натоварване на процесора) на конкретно изследвано софтуерно приложение за обслужване на потребители на списание „Автоматика и информатика“.

Изложението на дисертационното изследване се разгръща в следните четири глави:

Първа глава представя основни формализми от теория на автоматичното управление, които са използвани за основа на експерименталната работа, и предлага обзор на програмни системи, при които е приложено управление с обратна връзка. Тази част от текста описва софтуерните обекти, използвани за извършване на проведените изследвания. В нея са разгледани начините за моделиране на компютърни системи и са предложени примери за измерване на производителността на уеб сървъри.

Във ***втора глава*** на дисертационния труд са представени начини за анализ на компютърни системи, които да подпомогнат изграждането на модели. Отбелязани са някои фактори, които влияят на производителността на системата. Тази част на текста представя и структурата на компютърна система, използвана в хода на експерименталната работа. Описани са методологиите за изследване производителността на компютърните системи и са разгледани възможностите за наблюдение на софтуерните и хардуерни компоненти на последните. В главата се прави анализ на програмите за генериране на входни въздействия и се изгражда сценарий на тяхното прилагане. В резултат на това се изработва функционален модел при използването на софтуерното приложение, който да послужи за целите на експерименталната част на дисертационното изследване.

В ***трета глава*** на дисертационния труд са представени хардуерните и софтуерните характеристики на компютърната система. Текстът разглежда софтуерното приложение, което се използва за изследване на производителността на програмната система. Приложено е управление чрез настройка на комуникационния протокол TCP и на програмната система – уеб сървъра Apache. Описана е програмата за автоматично настройване на конфигурационния файл на Apache. В тази глава се прави опит са се демонстрира ефективността на направените подобрения чрез показателите за качество на компютърната система (време за отговор и бързодействие).

В последната ***четвърта*** глава на дисертацията е представено прилагането на методите за идентификация за конкретна компютърна система. По този начин се цели да се определят динамичните характеристики на системата чрез показатели на производителността ѝ. Представените характеристики се апроксимират и се изграждат модели на системата. Тяхната ефективност и приложимост се тества чрез метода на контролните карти. Реализираните софтуерни подобрения в изследваната система са моделирани чрез нелинейни модели, поради тяхната гъвкавост относно сложни явления и процеси. Получените модели са симулирани в симулационна среда на MATLAB, за да се изследва поведението на системата при включване на пропорционално-интегрален (ПИ) регулатор за управление на натоварването на процесора.

Ограниченията на дисертационното изследване произтичат от това, че:

– използваната в изследването компютърна конфигурация е с не много високи характеристики на компонентите;

– теория на автоматичното управление има собствени ограничения, касаещи реалните възможности за управление на сложни системи;

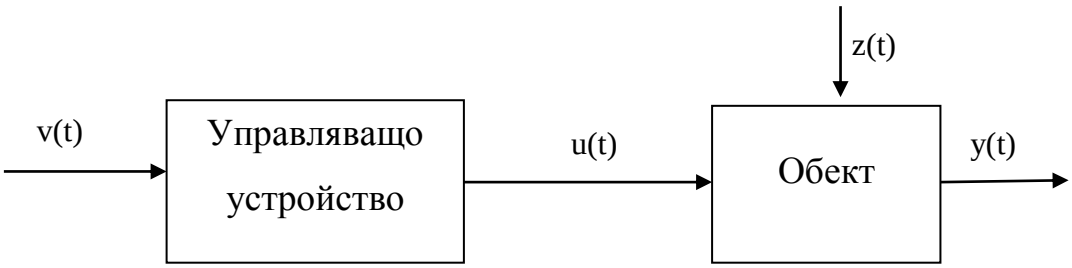
– програмната система (уеб сървърът Apache) и MySQL базата данни са инсталирани на една машина, което на свой ред влияе на експериментално получените резултати.

Въпреки ограниченията, в хода на дисертационното изследване бе установено, че прилагането на теория на автоматичното управление за подобряване производителността на компютърни системи е обещаващ подход в търсенето на решения за удовлетворяване нуждите на потребителите от по-висока скорост на обслужване. В тази връзка, бъдещи области на научен интерес, които биха доразвили търсенията на настоящия труд са: управление на облачни системи; управление на клъстърни системи; управление на програмни системи за адаптация; идентификация на данни и т.н.

I. ПРОГРАМНИ СИСТЕМИ И ТЕОРИЯ НА АВТОМАТИЧНОТО УПРАВЛЕНИЕ

I.1. Системи за автоматично управление

Теорията на автоматичното управление изследва принципите за изграждане на автоматични системи и закономерностите, касаещи протичащите в тях процеси, осъществява структурен анализ и синтез на системи. Системите за автоматично управление се състоят от управляващо устройство и обект за управление (фиг.I.1). Обектът за управление се характеризира с различни величини и променливи, които взаимодействат помежду си. Чрез тяхната свързаност се образува система за управление. Величините, които въздействат върху правилното функциониране на системата, се наричат входове на системата u(t), а тези, които възпрепятстват нормалното ѝ функциониране, се наричат смущения z(t). Изразяването на тези отношения между величините с помощта на математически изрази се нарича математически модел на системата.



Фиг.I.1. Обща схема на система за автоматично управление

В зависимост от информацията за системата относно данните за входните, изходните величини и смущенията, могат да се изградят различни начини за управление на обекта от управляващото устройство и да се приложат методи за автоматично управление като:

- **Управление по задание**

Управлението се осъществява само по зададена стойност u=f(v). Изходната величина и смущаващото въздействие не се контролират, а основната структура на системата няма допълнителни връзки. Недостатък на този метод за управление е, че не осигурява точност на системата, а обектът трябва да е с постоянни характеристики и незначителни смущения [ИщевК-07].

- **Управление по отклонение**

Управлението по отклонение е познато още като управление с обратна връзка. Изходният сигнал на системата се подава на управляващото устройство, което сравнява зададената стойност (v) с данните за изходната величина (y). Разликата между тях се нарича грешка на системата ε, ε = v – y. Последната се използва за формиране на управляващия сигнал (u), а той се подава на управлявания обект. Методът е по-универсален от метода на компенсацията, понеже се отчитат отклоненията на изходната величина и се отстраняват смущенията върху обекта. При управление с „обратна връзка" се извършва непрекъснат обмен на информация между входа и изхода, и обратно [ИщевК-07].

- **Управление по смущение**

Управлението по смущение се нарича още принцип на компенсацията. При него се контролират смущението на системата и текущата стойност на входната величина. Управляващото устройство подава сигнал към обекта u = f(v,z). По този начин се намаляват или отстраняват нежелани промени в изходната величина. За прилагане на този начин за управление е необходимо смущението да може да се измери и да се познава достатъчно точно неговото влияние върху изходната величина. Недостатъкът му е, че не се отчита изходната величина и/или други смущения върху системата. Управлението се осъществява само по изградените компенсиращи канали и поради тази причина методът рядко се използва самостоятелно [ИщевК-07].

I.1.1. Видове системи за управление

Изборът на начин за управление зависи от вида на системите, които се изследват. Те се класифицират въз основа на типа на модела, характера на сигналите или с оглед на други признаци и свойства на системата:

- **в зависимост от броя входове и изходи на системата**: системите за управление биват едномерни, когато имат един вход и един изход, или многомерни, когато са с повече входове и/или изходи.
- **в зависимост от изменението на системата във времето**: системата е стационарна, когато реагира по един и същи начин на произволно входно въздействие, независимо от момента на прилагане на въздействието. Ако реакцията на системата се променя с течение на времето, системата е нестационарна.

- **в зависимост от връзката между входните и изходните данни**: системите са

линейни и нелинейни. При линейните системи е в сила принципът на суперпозицията. Те притежават свойствата адитивност и хомогенност. Адитивността означава, че при две входни въздействия u_1 и u_2, реакцията на системата е сума от двете реакции на системата y_1 и y_1. Ако при въздействие на системата (u), умножено по константа, реакцията на системата (y) е също умножена по константа, то системата е хомогенна. При нелинейните системи не е в сила принципът на суперпозицията, т.е. изходът на системата не е пропорционален на входа.

- **в зависимост от смущенията на системата:** когато смущенията на системата са

контролируеми, тя е детерминирана. Ако смущенията са случайни, системата е стохастична.

- **в зависимост от сигнала от управляващото устройство**: системата е

непрекъсната, когато сигналът, който постъпва и се предава от управляващото устройство е непрекъснат. Когато сигналът се подава през определен интервал от време (интервал на дискретизация, Δt), той е дискретен, а системата се нарича дискретна.

- **в зависимост от разпределението на данните в пространството**; когато

поведението на системата се описва от краен брой променливи на състоянието, които са функция на един аргумент (време), системата е със *съсредоточени параметри*. Ако броят на променливите на състоянието е безкраен и/или те са функция на повече от един аргумент (време и пространствени координати), системата е с *разпределени параметри*.

Системите могат да се класифицират и според тяхното предназначение:

- *Система за стабилизация* – когато входната величина е зададена постоянна, задачата на системата е да поддържа постоянна изходната величина [ИщевК-07].
- *Програмно управление* – системата има допълнителен елемент, който задава входния сигнал на системата и променя параметрите на системата по определен закон във времето [ИщевК-07].
- *Следяща система* – осигурява изменение на изходната величина без предварително да е известна входната величина или последната е случаен процес [ИщевК-07].

В настоящото дисертационно изследване се използват следяща система, която измерва показателите на качеството на компютърната система и система с програмно управление, в която се осъществява управление на параметрите на системата чрез bash

скрипт в зависимост от показателите на входния сигнал. За да се реализират тези системи за управление е необходимо да се идентифицират и анализират показателите на системата.

I.1.2. Анализ и идентификация на системите за автоматично управление

При анализиране на една система за автоматично управление проследяването на нейното качество е от първостепенно значение. Оценява се качеството на преходните процеси относно времето, необходимо за достигане на установен режим, формата на преходния режим (монотонен или затихващи колебания) и се определят параметрите на колебанията на процеса.

Методите за синтез на системи за автоматично управление се използват, за да реализират промени в нея, така че изискванията да бъдат удовлетворени. За да се извършат анализът и синтезът, трябва да се разполага с подходящи математически описания – модели или характеристики на системата/обекта. Изискванията към моделите са да бъдат точни и да не са сложни. Видът на моделите може да бъде аналитичен, графо-аналитичен, графичен. Аналитичните модели се представят чрез уравнения, предавателни функции и структурни схеми. Графо-аналитичните модели са с времеви и честотни характеристики. Графичните модели използват графи за описание на системата.

Получаването на математическо описание на системите (процеси, обекти) може да се реализира въз основа на експерименти и по този начин да се идентифицират характеристиките на поведението на изследваните системи.

Експерименталните изследвания за идентифициране на обекта представляват процес, при който се измерват приложените върху системата входни въздействия. Теорията на автоматичното управление позволява да се използват различни стандартни входни въздействия като:

- едично входно въздействие, което определя преходната функция;
- функция на Дирак, която определя импулсно-преходната функция;
- хармонично въздействие, което определя амплитудно-честотната, фазо-честотната и амплитудно-фазовата характеристики.

Реакцията на системата при изменение на входното въздействие определя изходните характеристики на системата. Поради характера на избрания обект на управление тук – програмна система (уеб сървър Apache), характеризираща се с едновременно протичане на много процеси, по-нататъшните изследвания се отнасят именно към този тип идентификация на обекта. В дисертационното изследване входните

въздействия се осъществяват чрез единични входни въздействия. Изходните сигнали всъщност са носители на информация, която се обработва. Прилагането на методите за идентификация цели събиране на данни за работата на програмната система. Това позволява да се анализират получените данни и да се определят зависимости между измерваните изходни данни. По този начин се постига по-точна информация за работата на програмната система и се осигурява възможност за реализиране на допълнително управление, което ще подпомогне и облекчи нейното функциониране.

I.2. Програмни системи като обекти за управление

Обект за управление в настоящата дисертация е програмна система. Към нея трябва да се приложи подходящо управление, за да се удовлетворят най-добре изискванията на потребителите, за които е предназначена системата. От разгледаните по-горе принципи на управление ще бъде приложен принципът за управление с обратна връзка към избрания обект.

При него целта е да се използват измерените изходи на системата като време за отговор, бързодействие и използваемост за постигане на специфични задачи. Това се осъществява чрез настройка на параметри на входовете на системата, които влияят на размера на буфера, на политиките на планиране и на едновременното използване на една и съща входна област на системата. Управлението с обратна връзка се реализира в различни софтуерни приложения за управление на бизнес функционалности, за управление на мрежови устройства, за управление на натоварването на клъстери от сървъри, за управлението на поточни медии и др.[DGHPT-02]. По-долу са представени някои известни приложения на теория на автоматичното управление за информационни системи, които имат пряко отношение, макар и в различна степен, към характера на изследванията в дисертацията и типа на обекта за управление.

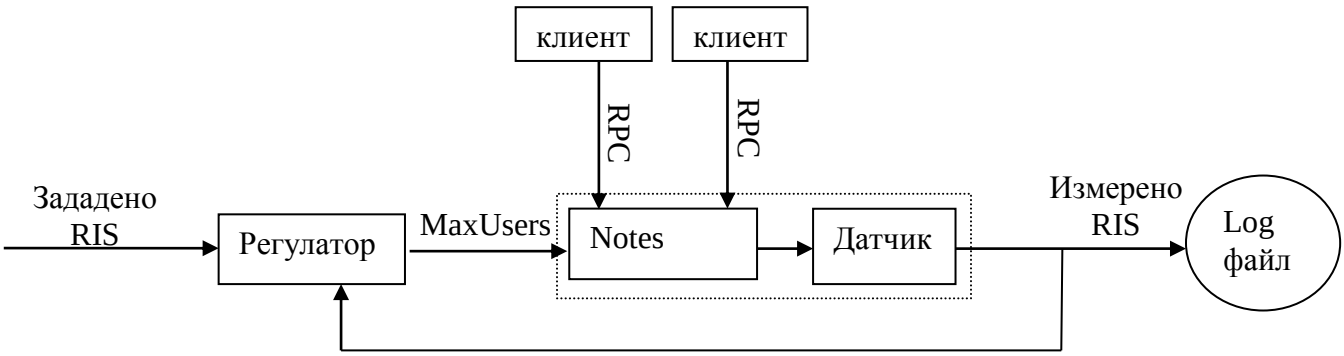
- **IBM Lotus Domino Server**

IBM Notes (бивша Lotus Notes) и IBM Domino (бивша Lotus Domino) са съответно клиент и сървър на софтуерната платформа на IBM. Сървърният софтуер се нарича IBM Domino, а клиентският е IBM Notes. Сървърният софтуер работи с операционни системи Windows, Unix, Linux, AIX и позволява да се поддържат десетки хиляди потребители на един сървър.

IBM Notes предоставя обединени бизнес функционалности като email, календари, списъци, управление на контакти, teamrooms (екипни стаи за разговори), дискусионни

форуми, обмен на файлове, микро-блокове, мигновени съобщения, блогове, потребителски директории. Предлага се също така достъп и интеграция с други IBM Domino приложения и база данни.

В [HeDPT-04] е разгледан mail сървър на приложението като обект за управление (фиг.I.2). Комуникацията между клиента и сървъра се осъществява чрез RPCs (remote procedure calls), които се обозначават с RIS стойност, много близка до броя на активните клиенти. Когато сървърът се използва от голям брой клиенти, това може да доведе до прекомерно използване на процесора и паметта на сървъра. Затова е важно управлението на броя едновременно обслужвани клиенти (RIS).



Фиг.I.2. Управление с обратна връзка на mail server [HeDPT-04]

Показателят, който се управлява, е максималният брой клиенти MaxUsers, т.е. входният сигнал. Зададената стойност на системата е броят едновременно обслужвани клиенти (RIS). Обектът, който се управлява, е комбинация от IBM Lotus Domino Server и датчик, използващ данните от log файла. Грешката на системата произтича от разликата между двете стойности: зададената и измерената. Изходните сигнали на системата се записват в log файл.

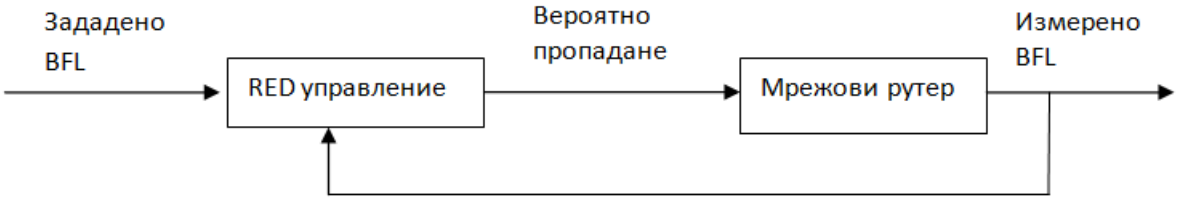
Това изследване демонстрира приложението на теория на автоматичното управление за управлението на софтуерна система, на която входният сигнал е броят на потребителите. Последното влияе на използването на процесора и паметта на системата. Подобен начин на изследване се използва в дисертацията с цел проследяване поведението на системата при различен брой потребители на софтуерното приложение. Недостатък на горепосочения софтуер е, че е с платен лиценз, а в дисертационното изследване се използва само свободен софтуер.

- **Random Early Detection of Router Overloads**

"Random Early Detection of Router Overloads" е технология за произволно отстраняване на пакети от рутер с цел защита от претоварване. Transmission control protocol (TCP) е основен протокол за комуникация между различни мрежи. Рутерите са устройства, които ги свързват. Проблемът, който се разглежда в [HoMTG-01a], [HoMTG-01b], е регулирането на мрежовия поток с цел да се избегнат мрежовите задръствания. Размерът на буфера на рутерите, които приемат комуникационните пакет, е ограничен и отхвърлянето на пакети води до необходимостта от повторното им изпращане.

Регулирането на буфера се извършва чрез технологията RED (Random Early Detection), която отстранява произволни пакети при проверка нивото на запълване на буфера (buffer full level). Използването на RED е с цел намаляване на мрежовите задръствания и увеличаване на бързодействието на системата. Предизвикателството при използването на RED е да се настрои параметъра BFL (нивото на запълване на буфера) .

В статията [HoMTG-01a] е предложено управление с обратна връзка, при което зададената стойност е желаното BFL, а управляващият сигнал е вероятността за премахване на пакети. Изходният сигнал е измереното ниво на запълване на буфера. Така, чрез коригиране на параметрите на RED протокола, се постига управление на потока на данни и се намаляват колебанията в работата на рутера (фиг.I.3).



Фиг.I.3. Управление с обратна връзка в мрежови рутер [HoMTG-01a]

Поради факта, че задачите на дисертацията са ограничени само до реализиране на управление на ниво програмна система, управлението на мрежовите устройства не представлява интерес на този етап, но показва, че се прилага управление чрез законите на автоматичното управление за TCP комуникационния прокол. В рамките на настоящото дисертационно изследване управлението на TCP комуникационния протокол се реализира чрез промяна на конфигурационните му параметри.

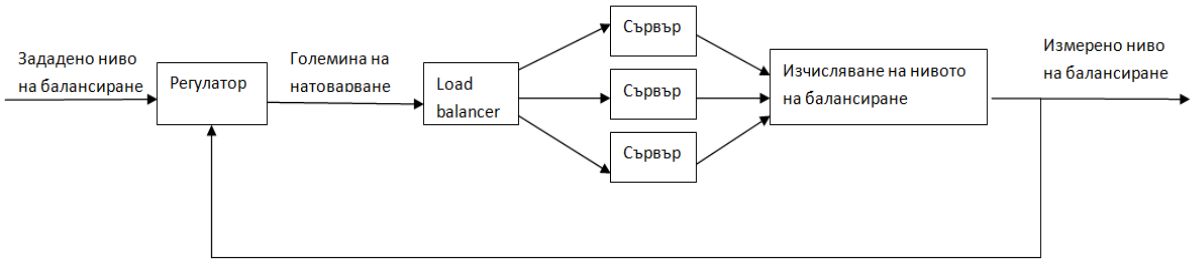
- **Балансирано натоварване (Load balancing)**

Тази техника служи за балансиране на натоварването на високопроизводителни компютърни системи. Системата за балансирано натоварване се използва при управление на клъстери, ферма от сървъри, ключове (switches) от високо ниво, при хранилища на данни и уеб сървъри.

Целта на балансираното натоварване е да се оптимизира използването на ресурса, да се максимизира бързодействието, да се минимизира времето за отговор и да се избегне пренатоварване на някой от ресурсите [GibbR-15], [LawDi-10].

Технологията може да се използва като начин за управление на система с обратна връзка. Целта на системата е да се постигне определена използваемост (utilization) или друг измерим показател в балансираната компютърна система, като към нея се добави допълнителен блок за изчисление на нивото на балансиране. То се определя от най-много и най-малко използваните сървъри. Контролерът определя големината на натоварването от разликата на изчисленото ниво на балансиране и зададената стойност. Управляващият сигнал на системата е големината на натоварването, което трябва да бъде разпределено на сървърите. Изходният сигнал за системата е нивото на балансирано натоварване (фиг.I.4.).

Използваният хардуер за провеждане на експериментални изследвания в дисертационния труд не позволява да се реализира този начин на управление, поради ограничения му ресурс, но би бил от полза при внедряването и разширяване на функционалните възможности на разработеното софтуерно приложение.



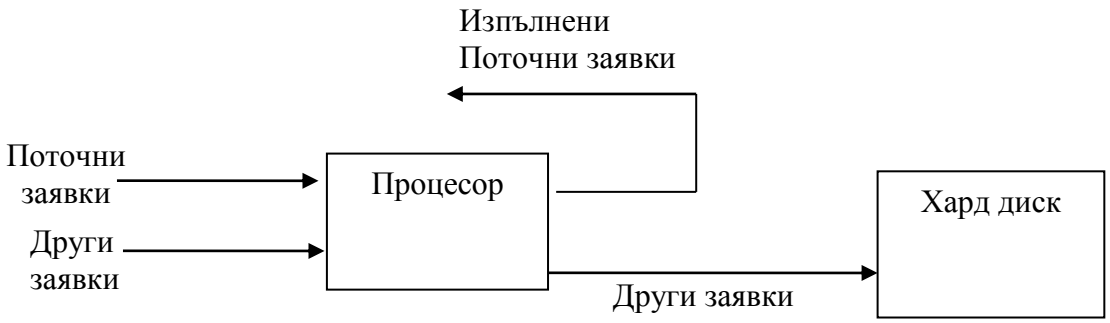
Фиг.I.4. Управление на load balancer с обратна връзка

- **Поточна медия (Streaming Media)**

В развитието на интернет технологията се появява и поточната медия (streaming media) като средство за доставяне на телевизия, радио или музика на живо. Тези приложения изискват непрекъсната доставка на информация с определена големина и качество. Предизвикателство при управлението на такива системи е постигането на максимално бързодействие, което да гарантира големината на доставките.

Целта е предоставяната услуга да поддържа кратко време за отговор, за да се избегне „засичане” на видеото или „прекъсване” на звука. Поддържането на поточна медия може да се осъществи чрез регулиране на времето за отговор, а последното – чрез управление на други непоточни задачи.

За постигане на споменатите цели за обслужване на медийния поток се изискват подходящи процесор и хард диск, които да приемат видео поток от 30 фрейма/секунда (фиг.I.5). По този начин се осигурява пълно обслужване на поточната работа, гарантиране на нивото на обслужване и максимизиране на бързодействието на ресурсите. Поточната медия може да бъде реализирана с продукти от тип свободен софтуер, за да се предостави онлайн звук и видео от провеждани научи конференции, обсъждания и семинари. Но трябва да се отбележи, че изграждането на поточна медия също изисква голям хардуерен ресурс.



Фиг.I.5. Управление на поточни заявки

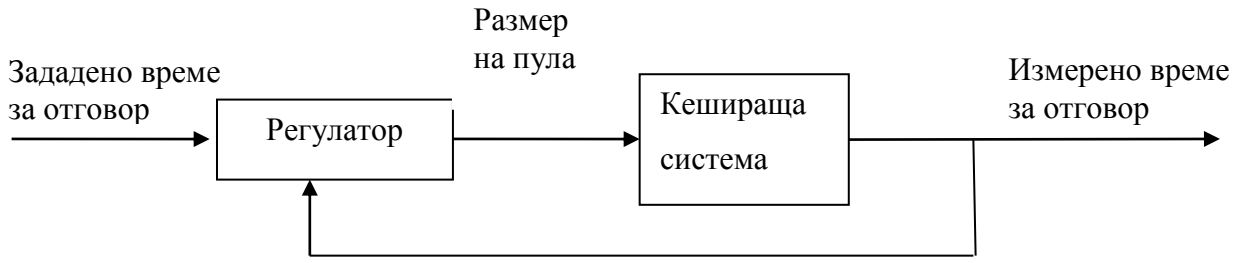
- **Кеширане (Caching Differentiated Service)**

Кеширането широко се използва за подобряване на производителността на компютърните системи. С поставянето на уеб страници в основната памет се намалява времето за извличане на страницата чрез елиминиране на достъпа до дисковете. Поради физическите ограничения на основната памет, броят на съхраняваните страници в нея е също е ограничен. Терминът hit се използва за справка на страницата (четене или писане), който е решен чрез поставяне в памет (кеширане) на страницата. Терминът miss обозначава достъп до паметта, която се изисква за четене от дисковата памет, т.е. некеширана страница. Система, която запазва в паметта тези страници, които имат най-голяма вероятност да получат следващия достъп, се счита за добра.

Производителността на система за кеширане се измерва от гледна точка на средното време за достъп. Тъй като времето за достъп до кеширани страница обикновено е

много по-малко от това за достъп до страница, разположена на хард диска, производителността се определя до голяма степен от hit вероятността. Реализирането на кеширане в настоящото дисертационно изследване е постигнато чрез активиране на кеширащия модул Memcached на уеб сървъра Apache, като по този начин се цели подобряване производителността на системата.

В [HeDPT-04] е разгледана една вариация на кеширането. Представено е обслужване на уеб съдържание за клиенти в няколко категории, разделени по заплащане за услуга. Предоставянето на различни услуги изисква системата за кеширане, която да може да достави желаните хитови вероятности за всеки клас клиенти. Този вид система работи по следния начин: заявките се извършват към конкретни кеш пулове (блокове на паметта) с различни размери, а заявките, които не могат да бъдат изпълнени чрез паметта за данни, изискват достъп от хард диска, което отнема значително повече време (≈1000 пъти). Това представлява проблем, който задава нужда от регулиране. Автоматичното управление използва система с обратна връзка, при която зададените стойности са желаното време за отговор за всеки клас на обслужване, а управляваният входен сигнал е размерът на пула (фиг.I.6). Също така е добре да се отбележи, че системата е с множество входни сигнали и всеки от тях се контролира поотделно. От това следва, че има и множество изходни сигнали, измерени за всеки клас услуга. Тъй като реализирането на това решение е сложна задача, а и изследваната система има само два вида потребители, неговото прилагане не е уместно. Освен това реализираните експерименти не изследват производителността на системата, когато тя се използва от потребители, които имат ограничени права за достъп до разработеното софтуерно приложение.



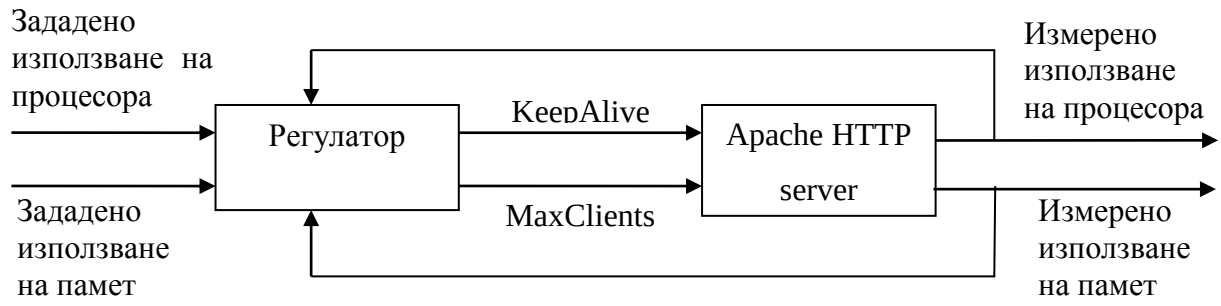
Фиг.I.6. Управление на кешираща система с обратна връзка

- **Уеб сървър Apache**

Уеб сървърът Apache е най-използваният софтуер, пример за клиент/сървър технология, който намира приложение за разпределение на информация, електронна търговия и пр. В класическия случай клиентите генерират натоварвания, които пристигат

по протокола http и създават изисквания, наречени request (заявка) за дадена услуга като например разглеждане на страници, сваляне на файлове и пр., а уеб сървърът отговаря с подходящата информация[BalkS-09].

В [DGHPT-02] авторите прилагат управление с обратна връзка за регулиране на използването на процесора и използването на паметта на компютърна система. В разгледаната система за управление зададените величини са две: използване на процесор и използване на памет, а въз основа на измерените стойности за използване на процесора и паметта се настройват параметрите на Apache: MaxClients и KeepAlive. Авторите са забелязали, че двата входни параметъра влияят на използването на процесора.



Фиг.I.7. Управление на web server Apache с обратна връзка [DGHPT-02]

Представените резултати в [DGHPT-02] имат следното ограничение: изменението в използването на процесора и паметта зависят и от други параметри на конфигурационния файл. Изследването им дава възможност за по-задълбочен анализ с оглед задачите на настоящия дисертационен труд.

От софтуерните приложения като обекти за управление, разгледани в т. I.2, интерес за дисертацията представлява само управлението на уеб сървър Apache, тъй като той се използва от членове на секция „Йерархични системи“ на ИИКТ-БАН за изследователски и приложни цели. Освен това реализираното софтуерно приложение е създадено с продукти с отворен код, работещи изключително ефективно с уеб сървър Apache и операционна система Linux. Трета предпоставка за изследване на уеб сървъра Apache е, че използваната хардуерна конфигурация е с ограничен ресурс, което не позволява да се изследват по-сложни софтуерни приложения.

Теорията на автоматичното управление се прилага и за изследване на компютърните системи като обект на управление. Те, със съответните програмни системи, непрекъснато се усложняват с оглед удовлетворяване на нарастващите изисквания към тях

от страна на потребителите. Възниква необходимост от прилагане на специализирани методи за управление на тези системи, тъй като заложените класически методи за управление в компютърните системи не дават задоволителни резултати. В този смисъл, прилагането на методите и принципите на управление от теорията на автоматичното регулиране към горепосочения клас обекти представлява належаща задача, за чието решаване са насочени усилията в тази работа.

Сложността на софтуера се поражда от изискванията към него, а именно: да бъде бърз, с висока производителност и кратко време за реакция; да е вграден – изпълним с минимални ресурси и взаимодействащ с външната среда; да е отказоустойчив – работещ в критична среда; да е разпределен – работещ на няколко свързани компютъра; да е интелигентен – с независимо поведение; да е отворен – със свободен достъп; да е хетерогенен – изпълним на различни платформи. Софтуерът, който се използва от вградените контролери за управление на такива системи също трябва да отговаря на тези изисквания, а самите контролери – да са лесни за работа и да постигат висока производителност на системите. Машинните контролери, контролерите на процеси, серво-контролерите, PLC и робот-контролерите са изключително използвани в индустрията, вградени в големи разпределени системи, работещи в реално време, и поддържащи стандартни интерфейси за оперативна съвместимост.

Традиционните системи за управление са статични, но технологичният напредък и изискванията на пазара ги промениха в динамични системи, при които гъвкавостта е ключов атрибут. Интелигентността на локалните компоненти се повишава и големите разпределени контролери са разработени като обединение от взаимодействащи си интелигентни агенти.

Комуникативността на системите за управление се увеличава чрез интернет и мобилните устройства. Стандартизацията и използването на отворен код са текущи изисквания към системите за управление. Новите езици като Java, C# и Corba са лесни за употреба, преносими са и е възможно използването им за разпределени хетерогенни системи за управление. Системите за управление са важна част на вградените системи, работещи в реално време, за които най-важният критерий е предвидимостта. Софтуерната реализация на контролер е част от компютъризирана обратна връзка, която налага времеви ограничения на софтуера и платформите.

При традиционния подход системите, работещи в реално време, се проектират така, че да са максимално времево детерминирани. При софтуерните приложения времевата

неопределеност се разглежда като променливост или смущения. Това дава възможност на динамичните системи да управляват приложения и изпълнения на платформи, които предоставят достъп до споделен ресурс като процесорно време и комуникационен канал. По този начин управлението на производителността може да се разглежда като параметър за качеството на услугата.

Използването на автоматично управление цели увеличаване гъвкавостта на системата, като запазва нейната надеждност и ефективност. Техниките за управление могат да послужат за компенсиране на недостатъци и несъвършенства в платформите за изпълнение [SRAKE-03].

Теорията на автоматичното управление се прилага в различни области на компютърните системи, които са част от сложни системи за управление като:

- **Приложение в компютърни мрежи**

В предходни години при прилагането на теорията на автоматичното управление в компютърни системи значителен интерес е представлявало изследването на управлението на комуникационния поток. Един от пионерите, Keshav [Kesha-91] изследва производителността на потока пакети на ниво мрежови ключове (switches) при различни допускания относно инфраструктурата и протоколите на мрежата. Той оценява показателите на модела на системата чрез оценка на Калман и размита логика. Други [LSWPS-01] прилагат теорията на автоматичното управление към кратните изменения на времевите граници за обслужване на TCP потока.

Протоколът Transmission Control Protocol (TCP) се използва за комуникация между мрежите, а предаването на голямо количество данни чрез него може да предизвика задръстване на мрежовия поток. В тази връзка в [LSWPS-01] се изследва задръстването на TCP като нелинейна система с обратна връзка, която динамично донастройва скоростта на предаване спрямо състоянията на задръстване на мрежата. В статията се използва хибридно пространство на състоянието, за да се изследва поведението на споделената честотна лента (bandwidth) при еднакво количество предавани данни през няколко потока едновременно. В изследването са поставени ограничения относно скоростта на предаване на данните и нивото на запълване на буфера, за да се докаже, че системата е устойчива и клони към устойчив граничен цикъл при синхронно отхвърляне на всички потоци.

Резултатите потвърждават, че при теоретичните friendly TCP потоци[1] прогнозата за количеството предадени данни не гарантира споделяне на честотната лента според конкурентни потоци и че са нужни допълнителни показатели за равноправно споделяне на bandwidth. Целта е да се конструира последователност на TCP трафика спрямо средни времеви данни, но да не бъдат по-малки от кратките TCP времеви данни. Идеята е да се измери вероятността на загубите, а RTT (Round-trip time) измерва времето необходимо за изпращане на сигнал и получаване на потвърждение за получаването му, за да се състави модел на TCP бързодействието. Очакваното бързодействие от модела се използва за директно управление на големината на потока.

TCP протоколът е един от най-използваните в мрежовата комуникация. Прилагането на теория на автоматичното управление при неговото изследване позволява да се определят зависимостите при функционирането му. В дисертационния труд са анализирани неговите конфигурационни параметри и са приложени промени, за да се повишат неговите бързодействие и отказоустойчивост.

За управление на задръстването във високоскоростни мрежи в [Masco-99] се използва предиктор на Смит. Математическият анализ показва, че приложеният закон за управление гарантира устойчивост на мрежовите опашки и пълното използване на мрежовите свързвания в общата мрежова топология в преходно и установено състояние. Този алгоритъм е приложим и за TCP/IP, за да се управлява window формата или буфера за получаване на пакети. В настоящето дисертационно изследване се прилага управление на тези две опции на TCP комуникационния протокол.

Теорията на автоматичното управление се използва за анализиране действията в TCP протокола за система, която е Active Queue Management (AQM) и за обяснение на компромисите за избор на параметрите на RED алгоритъма [HoMTG-01a]. Системата за автоматично управление е с обратна връзка, която е необходима, за да се поддържа относителна дължина на опашката. Недостатъкът на този метод е, че при по-бързи системи се получават по-големи закъснения и не се използва ефективно наличния хардуерен ресурс, както и че възникват смущения, които влияят на производителността на системата и на други протоколи. Друго ограничение на RED алгоритъма е директното

[1] TCP-frendly rate control е механизъм за управление на задръстванията в unicast поток, интернет операции и конкурентен TCP трафик. Unicast е предаване на съобщения между един подател и един получател в дадена мрежа.

влияние на дължината на опашката върху загубите в системата с обратна връзка. Процесът се управлява чрез позиционен регулатор (реле), което води до нежелани колебания при управлението на дължината на опашката. Същите автори изследват аналогична система, но заменят позиционния регулатор с пропорционален и пропорционално-интегрален регулатор [HoMTG-01b]. Те установяват, че и двата регулатора превъзхождат RED алгоритъма, като осъществяват по-бърз отговор и достигане до установено състояние. При тяхното използване се постига по-малка латентност, по-висока мрежова производителност и по-добро използване на опашката. Тези изводи ще бъдат използвани в настоящата работа, за да се приложат допълнителни настройки към опциите на комуникационния протокол TCP.

- **Приложение на теория на автоматичното управление в междинно софтуерно ниво (Middleware)**

Middleware са софтуерни системи, които улесняват разработването на стабилни приложения от междинно ниво, например сървърни приложения (Apache http сървър), системи за управление на база данни (IBM's Universal Database Server) и e-mail сървъри (IBM Lotus Domino Server).

Всички тези системи са изправени пред едни и същи проблеми – те трябва да осигуряват необходимото ниво на обслужване на клиентите, да регулират използването на ресурсите и да оптимизират софтуерните конфигурации.

Изследването на първия проблем за Apache е описано в [AbdeB-99]. Уеб сървърът Apache се разглежда като архитектура, която предоставя е-услуга с увеличаващи се изисквания за качеството, надеждност и сигурност в силно динамична среда. За да се постигне желаното качество на услугите за клиента, се въвежда управление на качеството QoS на уеб сървъра, което да разграничава производителността според класа на клиента, изолираността на независими услуги, планирането на капацитета за предоставяне на качество на услугите с определена големина на заявките и bandwidth delivered (количество доставени данни за определено време). Подходът предоставя управление на ресурса на уеб сървъра чрез адаптация на уеб съдържанието. Също така той подобрява сървърната производителност чрез избирателно адаптиране на съдържанието според условията на натоварване и изискванията за качество на услугата. Реализирането на управлението се извършва чрез API приложение, написано на C и работещо на ОС Linux. Законът за управление на натоварването на процесора е PI. Целта на управлението е да се постигне оптимално качество на използването на всяка виртуална машина, като се отчита, че

използването на процеса, който се управлява, е функция на Ri (големината на заявките) и Wi (количество данни), и се задава приоритет на сайтовете, поддържани на всеки виртуален сървър. Като недостатък на изследването може да се посочи, че то се фокусира само върху заявки към статични уеб страници, а последните намират все по-малко приложение.

Управление с обратна връзка, приложено за уеб сървър Apache, е докладвано в [LLASS-06]. Сървърът се регулира относно желаното закъснение за обслужване на потребителските заявки, като се извършва адаптиране на ресурсите на сървъра спрямо неговото натоварване. Управлението е по два критерия – относително закъснение и абсолютно закъснение. Относителното закъснение се дефинира за ненатоварен сървър, а абсолютното – според класа на клиента. Принос на статията е разработването на програмен код, който да разпределя ресурса на сървъра за постигане на желаните закъснения, а предизвикателството за създаване на кода са предварително неизвестните натоварвания.

В [DGHPT-02] се изследва използването на Multi-Input-Multi-Output система за управление, приложена върху обект за изследване Apache. Авторите провеждат експерименти и установяват, че съществува взаимосвързаност между входните въздействия на обекта KeepAlive (KA) и MaxClent (MC) и между измерените изходи CPU процесор и MEM памет. На базата на експерименти е извършена идентификация на обекта и е съставен модел на обекта. Приложената MIMO система предоставя значителни ползи за управлението по отношение на скорост, на сходимост и на чувствителност към случайни колебания при изпълнение на желани политики.

Приложението на това управление е ценно, понеже използваните ресурси – памет и процесор са ограничени. Целта е да се състави съответното разпределение на капацитета, така че приложенията (напр. файл сървър, база данни сървър) да избягват задръстванията и пропаданията в резултат на пренатоварване.

Методи за онлайн оптимизация на параметъра MaxClient на уеб сървъра Apache, който контролира броя на едновременно работещите workers, се разглеждат в [LSDFP-03]. В изследванията се изгражда функция на MaxClient в зависимост от времето за отговор (Response Time) и се вижда, че тя има покачване, което е предпоставка да се търси оптимална стойност за MaxClient. Трябва да се има в предвид, че MaxClient е компромис между закъсненията в TCP приемаща заявка и хардуерния ресурс на системата. Използват се два начина за намиране на оптималната стойност на MaxClient – чрез аналитични

техники и чрез измервания. При първия начин са включени метод на Нютон и размита логика, а при втория се измерват ограниченията на използвания ресурс и минимизиране на времето за отговор. И трите подхода намаляват времето за отговор 10 пъти, но всеки от тях има недостатъци: методът на Нютон не води до надеждни резултати за силно променящи се данни като времето за отговор; размитото управление е стабилно, но има бавна сходимост; евристичния подход дава добри резултати за изследваната система, но не може да се обобщи, понеже се изискват познания за ограниченията на хардуерния ресурс на системата и измервания на тяхната използваност. Ограничение на изследването е, че се оптимизира само един показател MaxClient, а е пропуснат друг – като KeepAliveTimeout. Изследва се малка и слаба компютърна конфигурация, а не например голям база данни сървър, сървър за приложение или разпределена система. В [DiaHP-02] се използват данни от предходната статия и се изгражда контролер с размита логика, за да се минимизира времето за отговор.

Широко използван подход за постигане на желаното ниво на обслужване на email сървъри е добавянето на контролер, който манипулира параметрите на системата. В [PGHTJ-02] се предлага методология за проектиране на такива контролери, основаваща се на класическата теория на автоматичното управление и състояща се от два етапа – идентификация на системата и проектиране на контролера. В първия етап се създават математически модели на email сървъра, от типа ARMA, които съвпадат с измерванията на контролирания сървър. Моделите са приложими към по-голям клас системи, например Lotus Notes Groupware Server, за който са получени много добри резултати. Във втория етап на проектиране на контролера, анализът на моделите води до създаване на контролер, който ще постигне нивото на обслужване. В статията е докладван анализ на затворена система, чрез прилагане на интегрален закон за управление върху Lotus Notes, като целта е да се поддържа желана дължина на опашката. Използвайки ходограф на корените (root-locus) от теорията на автоматичното управление, авторите предсказват възникването или липсата на колебания в отговора на системата, които да произхождат от контролера. Такива колебания са нежелани, тъй като те увеличават променливостта на системата и по този начин става невъзможно постигането на нивото на обслужване. Създаденият контролер е приложен за реална lotus note система и е наблюдавано значително съответствие между поведението на реалната система и предсказанията от анализа [PGHTJ-02].

В [DiaHP-01] се предлага контролер, който цели реализиране на високи печалби. Авторите разработват автоматично изпълняване на услуги на ниво обслужване чрез информационни технологии с обратна връзка за планиране на процесора и балансирано натоварване, и за постигане на максимална печалба от предоставяните услуги. Разработва се работна рамка, в която печалбите се определят от приходите от предлаганите услуги, а отстъпките за клиентите се правят от недостъпни услуги и дълго време за отговор. Описана е методология за проектиране на такъв контролер, която се прилага за lotus mail server. Използвани са прости линейни модели за изследване на динамиката на mail server при различни работни натоварвания. Изследването показва, че при използване на по-бързи контролери се получават по-големи печалби, понеже рядко се нарушава договореното време за отговор. Авторите изтъкват също, че зададената стойност на контролерите има изключително влияние върху печалбата.

Други изследвания също дискутират нови начини за повишаване на печалбата от предлагани електронни услуги. В [MeAFM-00] се разглежда управлението на система за електронен магазин въз основа на неговия приход и печалба. Това е нов метод, понеже качеството на услугите на електронния магазин се определя чрез управление на ресурсите – процесор, дискове, капацитет на мрежата и чрез следене на показатели на производителността като време за отговор, бързодействие и надеждност. В експеримента се извършва оптимизиране на бизнес показатели, вместо на показатели за производителността. Проследяват се потребителските сесии и се изгражда модел на поведението на потребителя, на базата на който се определя приоритетна политика за управлението на ресурса. Приоритетите се променят въз основа на състоянието на потребителя и парите в неговата потребителска кошница. Представен е симулационен модел, който показва, че бизнес ориентирания подход би осигурил по-висок приход за секунда при пикови натоварвания.

В статия [RoCAS-02] се описва проектирането и изпълнението на ControlWare, архитектура от междинно ниво за управление на QoS на базата на теория на управлението, и се мотивира от нуждите на производителността за предоставяне на сигурни интернет услуги. Авторите представят ефикасността на разработената архитектурата за постигането на QoS в реалистични условия при управление на натоварване на уеб сървър и прокси сървър.

За уеб сървъри, които обслужват статично съдържание, обработването на данните по мрежа може значително да повлияе върху производителността на процесора. Поради

тази причина авторите в статия [PatLS-04] чрез аналитичен модел изчисляват максималното възможно подобрение на производителността, което се получава, когато се използва разтоварващ протокол (offload protocol), който да намали натоварването от статични страници с до 50%. Този протокол е ефективен при мрежови устройства с голяма честотна лента, поради факта, че в използваната компютърна система има мрежова карта, която е с пропусквателна способност 100Mbps. Настройки за този протокол не се прилагат в настоящето дисертационно изследване.

- **Приложение на теория на автоматичното управление в операционни системи**

Съществува значителен интерес към прилагане на теорията на автоматичното управление при операционни системи. Тази теория широко се използва при управление на виртуалните машини на IBM за постигане на различни нива на обслужване, като това се основава на детайлно познаване на управляваните входове и измерените изходи на операционната система. В статията [AmEEP-97] се докладва управление на работното натоварване, функция на OS/390, което позволява инсталации за определяне на бизнес цели в клъстерна среда (Parallel Sysplex). Операционната система осигурява ресурс за постигане на специфични бизнес цели. Разработените алгоритми опростяват управлението на производителността, настройват динамично компютърните ресурси и балансират работа между паралелни системи. Авторите изследват алгоритми, чрез които настройват разпределението на системния ресурс, за да постигнат адаптация към променящите се условия.

- **Приложение на теория на автоматичното управление в самонастройващи се системи**

Високите разходи за работа на големи компютърни системи обосновават интереса към опити за намаляване на човешкия фактор и създаване на самоуправляващи се системи. Основно предизвикателство в реализирането на такива е разбирането как автоматизираните действия влияят върху поведението на системата и по-точно върху нейната стабилност. Теорията на автоматичното управление предлага богат набор от методологии за създаване на автоматизирани само-диагностициращи се и адаптивни системи с характеристики като устойчивост, кратко време за достигане на установен режим, точно регулиране.

В [DHPGK-06] се представя модел за изграждане на самоуправляваща се система, описват се елементите на системата чрез архитектурата на IBM autonomic computing и се намира съответствие между структурата на системата и теорията на автоматичното

управление. Отбелязват се трудностите за прилагането на теорията на автоматичното управление в компютърните архитектури, като се разработват надеждни модели на ресурсите на системата, управление на сензорните закъснения, определя се времето за изпълнение на процес от изпълнителните механизми и програми за изследване на системата (benchmark). Други автори използват софтуера за разработване на агенти Jade за проектиране и изпълнение на среда за автономно управление на съществуващи системи [BPHDT-06]. Средата се състои от две работни рамки. Едната се администрира от съществуващите системи, а другата е независим мениджър, който регулира настройките на управляваните елементи по специфични критерии. За да се демонстрира ефективността на подхода, тези рамки се прилагат за оптимизация на J2EE приложения в условия на големи натоварвания, получавани от интернет приложенията. В литературата се срещат и източници, описващи приложението на теория на автоматичното управление и в други компоненти на компютърни система като например управление на мултимедийна среда, управление на електрическото захранване на компютърните системи и за управление на работната температура на процесорите [CPSCW-95], [CBBLM-02], [StanS-03], [SkaTM-02].

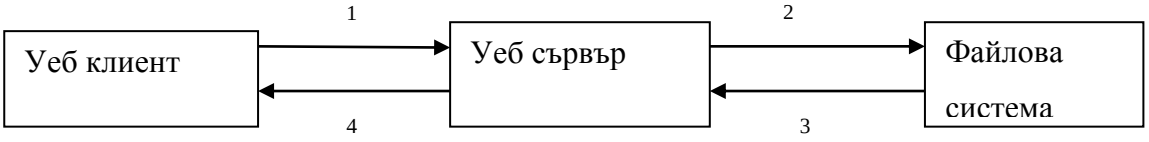
За да се приложи теорията на автоматичното управление при компютърни системи, трябва да се изследват и анализират възможностите на съществуващите софтуерни приложения и хардуерни конфигурации. В следващата част са описани софтуерните системи – уеб сървъра Apache и операционната система Linux, които са в основата на проведеното дисертационно изследване.

I.3. Характеристики на софтуерните обектите за управление

I.3.1. Уеб сървъри

- **Общ принцип на работа на уеб сървърите**

Процесът на работа в клиент/сървър технологията се разделя на четири етапа (фиг.I.8), в които съответно клиент (уеб браузър) и сървър (уеб сървър) си взаимодействат [СпаКШ-09].



Фиг.I.8. Процесът на заявка-отговор между уеб клиент и уеб сървърите

В рамките на тези етапи:

1. Клиентът чрез уеб браузера осъществява свързване с уеб сървъра и изпраща заявка към него.

2. Сървърът получава заявката, открива файла или програмата във файловата система, която се изисква в заявката.

3. Сървърът извлича файла от файловата система.

4. Сървърът изпраща файла на уеб браузъра и информацията стига до клиента.

- **Видове уеб сървъри**

В зависимост от начина на обслужване на потребителските заявки, могат да се отграничат следните видове уеб сървъри:

- *Еднонишков уеб сървър*: когато пристигат http заявките, уеб сървърът започва да извлича ресурса, който се изисква чрез тях от клиента. Докато уеб сървърът извлича ресурса, клиентът може да изпраща още заявки. Тези заявки трябва или да се игнорират, или да се обработят едновременно. Уеб сървърът, който игнорира или нарежда заявките в опашки, докато е зает, се нарича еднонишков (single-threaded). Този вид сървър е подходящ за малък до умерен трафик.

Уеб сървърите могат да управляват едновременно заявки по два начина – когато стартират нов процес или когато се стартира нова нишка в процеса. С оглед на това те биват:

- *Многопроцесорен (multiprocessed) уеб сървър* – когато уеб сървърът стартира нов процес за всяка заявка.Такъв е и Apache за Linux.

- *Многонишков (multithreaded) уеб сървър* – когато уеб сървърът стартира нова нишка за всяка заявка. Такъв например е IIS. При платформата Windows липсва (forking) разклонение на клиентски сървъри (процеси) за разлика от Unix платформата, което прави последната по-ефективна. Unix е софтуер за хардуерна архитектура, базирана на уеб сървър.

I.3.2. Уеб сървър Apache

Първоначално Apache е замислен като алтернатива на Netscape web server. Той представлява отворен код с функционалности, работещи на различни платформи. Изграден е и е поддържан от сайта Apache.org. Той е съвместим с много уеб приложения и е основен компонент от стека LAMPP (XAMPP)[TruvL-08].

Първата версия на уеб сървъра Apache е създадена от Robert McCool през 1996 г. Съществуват две версии за произхода на името „Apache”. Първата го свързва с името на индианското племе Apache, а втората – с кода му, който представлява сбор от хетерогенни части – a patches.

Конкурентите на Apache са разделени в две групи: свободни уеб сървъри и лицензирани уеб сървъри. Представители на първата група са Boa, Red Hat Content Accelerator, thttpd, Mathopd, а на втората група – Microsoft IIS, IBM, Zeus и IPlanet. Основни конкуренти на Apache са IIS на Microsoft, Sun Java System web server и много други.

Apache е изключително популярен уеб сървър, тъй като разполага с полезни възможности и предоставя добра скорост, преносимост, стабилност и сигурност, което се потвърждава от много изследвания и статии [WebTS-16].

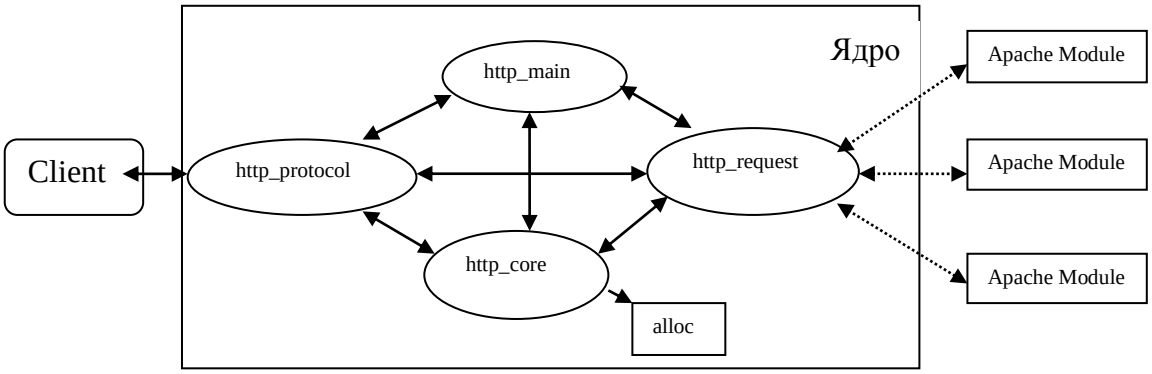
Apache поддържа статични и динамични уеб страници. Страници, на които не се променя съдържанието се наричат статични. Те се създават чрез таговете HTML и се използват от ранните години на интернет до днес. По-късно се създават динамични страници, които притежават визуална привлекателност и по-голяма интерактивност. Тяхното съдържание се променя, докато клиентът общува със сървъра, въз основа на входни данни, зададени от потребителите на страницата.

- **Архитектура на Apache**

Уеб сървърът Apache е изграден от Apache-ядро и Apache-модули. Модулите предоставят начин за добавяне на нови функционалности без те да се изпълняват в ядрото по подразбиране. По този начин Apache не е код, който изпълнява всички функционалности. Предимство на Apache е лесното изпълнение и добавяне на различни възможности чрез модули.

Ядро на Apache

Ядрото на Apache изпълнява основните функционалности на уеб сървъра чрез използване на определен брой полезни функции и осъществява взаимодействие с всички компоненти около него. Ядрото се разделя на малки компоненти, които извършват основни действия за уеб сървъра. Ядрото на Apache съдържа следните компоненти: http_protocol.c; http_main.c; http_request.c; http_core.c (фиг.I.9).



Фиг.I.9. Компоненти и взаимовръзки в структурата на Apache

Компонентите на ядрото са поредица от класове, които изпълняват конкретни задачи. Те се различават от модулите, които се добавят за изпълнението на различни изисквания към уеб сървъра и по този начин персонализират действията му. Ядрото предоставя основни функционалности на Apache, а модулите разширяват функционалностите.

- Компонент „http_protocol.c" – управлява всички процедури, които комуникират директно с клиента чрез http протокола и сокет свързванията. Чрез него клиентът се свързва със сървъра и се прехвърлят всички данни.
- Компонент „http_main.c" – този компонент е отговорен за стартирането на сървъра и поддържането на основните задължения на сървъра като изчакване и приемане на свързването. Също така е отговорен за timeouts (времената за прекъсване).
- Компонент „http_request" – управлява потока на обработване на заявките, предава управлението на модулите в определен ред и отговаря за отстраняването на грешките.
- Компонент „http_core.c" – изпълнява основни функционалности и самообслужва документи.
- Alloc.c – отговаря за разпределението и наблюдението на ресурсите на пуловете.
- http_config.c – осигурява функции за други дейности като четене на конфигурационни файлове, управление на информацията събрана от тях, поддържане на виртуални хостове, предоставяне на списък с модули, които ще се използват в различни фази на обслужване на заявките.

Компонентите в ядрото добавят функционалности и позволяват на уеб сървъра да бъде по-сигурен и устойчив.

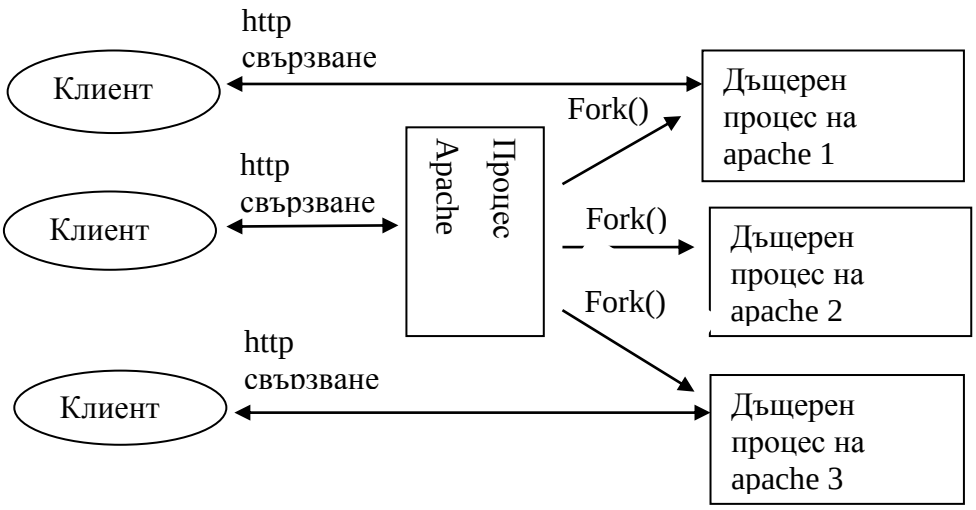
Модули на Apache

Всички модули на Apache са свързани към ядрото. Те нямат директна връзка помежду си, а комуникацията им се осъществява чрез ядрото. По този начин се постига стабилност, надеждност и възможност за откриване на грешки.

Използването на модулите позволява на Apache да постигне устойчивост на програмата, по-бързо действие, а динамичното зареждане на модулите позволява изключително скоростно първоначално зареждане на програмата.

Модулите на Apache са разширени функционалности на ядрото. Те се управляват чрез handler. Handler-ът е действие, което трябва да се изпълни в дадена фаза на обслужване на заявката. Например handler-ът, който изисква файл, трябва да отвори файла, да го прочете и да го изпрати на ядрото за препращане към клиента. Handler-ът се определя от модулите, като те се специфицират за една или няколко фази на изпълнението на заявката. Той изпраща обратно обработената информация от модула на Apache към ядрото.

На фиг. I.9 е показана схема на Apache, при която се обслужват едновременно три клиента на уеб сървъра. Обслужването се извършва съответно от: процес Apache; дъщерен процес на Apache 1 и дъщерен процес на Apache 3. Дъщерен процес на Apache 2 е в изчакване на следващия клиент. Разклонението на процеса Apache се извършва чрез системно извикване fork().



Фиг.I.10. Обслужване на няколко клиента [TruvL-08]

Apache поддържа два вида процеси: родителски и дъщерни. Родителският процес извиква няколко дъщерни процеса. Когато заявките са изпратени към Apache сървъра,

родителският процес получава заявките, след което той препраща заявката към един от дъщерните процеси. Дъщерните процеси обработват заявките и отговарят за тях.

Apache използва определен брой дъщерни процеси, които се разклоняват при самото стартиране на програмата. По този начин се обслужват независимо пристигащите заявки.

Apache сървърът контролира динамично броя на дъщерните процеси, в зависимост от моментното натоварване на хардуерната конфигурация.

Представената по-горе архитектура на работа на Apache се използва във версия 1 и е класическа архитектура, работеща много добре в среда на Linux, но е неефективна в среда на Windows, понеже разклоняването на процеси е скъпа операция. Затова във версия 2 на Apache специалистите са разработили платформа, която е по-ефективна чрез pluggable (сменяем) модул Multi-Processing Module и може да се оптимизира за различни операционни среди. По този начин е възможно да се избере най-подходящия MPM модул за съответната операционна система [KewN-05].

- **Архитектури, използвани от Apache 2.0**

Във версия Apache 2.0 се поддържат следните видове многозадачни архитектури:

Многозадачни архитектури на Apache и MPMs Мулти-платформени модули

В зависимост от платформата, на която работи сървърът, се избира видът на многозадачната архитектура – многонишкова или многопроцесорна. Това позволява да се направи правилния избор за MPM и за коректното ѝ конфигуриране. Ако е необходимо да се допусне компромис относно производителността, Apache е проектиран така, че да позволи намаляване на латентността и увеличаване на бързодействието при обслужване на повече заявки. Така се гарантира надеждна обработка на заявките в разумни срокове.

Друг избор, който трябва да се направи, е относно начина на комуникиране между задачите. Обикновено това се осъществява чрез споделяне на памет, сигнали, събития, mutex, семафори, именувани канали или сокети.

Всички MPM използват стратегия пул от задачи (task pool). Те са изградени от свободни workers, оставени в спряно състояние, докато пристигат заявките. Workers, на свой ред, обработят заявките веднага след като последните пристигнат. Също така MPM разполагат със средство за принудително спиране на всички свободни процеси и за събуждане, когато възникнат заявките. За тази цел операционните системи използват

механизми като условна променлива или семафор, за да изпълняват mutex, а задачите се прекратяват принудително чрез процедура за блокирането на mutex.

Сървърните сокети са ограничен ресурс, така че може да има само един слушател за сокет или един слушател на всички сокети. Има надеждни слушатели на задачи, които се използват за обработка на опашката и предават изискваните данни към worker задачите. Възможно е всички сървърни задачи да играят и двете роли – един празен worker става слушател, получава заявката, преобразува се в worker и обработва заявката.

Изпълняването на множество процеси от Apache е напълно променено във версия 2.0 чрез въвеждане на multi processing modules (MPM) (множество модули за обработване). Модулите представляват напълно капсулирани многопроцесорни архитектури, които подлежат на избор. Всеки един модул е стандартна модулна структура, която включва командна таблица. Изборът на модула, който ще работи на сървъра, се осъществява при компилирането на сорс кода и не може да се извършва динамично. Само един модел на многозадачната архитектура MPM може да бъде включен на един сървър в даден момент. Версия 2.0 на Apache поддържа четири групи MPM:

- Preforking and netware са модули, които имат функционалности подобни на prefork архитектурите от Apache 1.3;
- WinNT е подобна Win32 версия на 1.3, с добавена iopc концепция;
- Worker – използва се за нишки и процеси, и има по-добра производителност от prefork;
- Leader and PerChild са две експериментални MPM, които са алтернативи на worker и prefork при Linux [ApaMP-04].

Разликата между различните многозадачни архитектури се състои в средствата за създаване и организиране на дъщерните процеси и за създаването на дъщерни процеси по време на рестартиране.

Многопроцесорни архитектури

Основната задача на многопроцесорната архитектура е да осигури бърз отговор от сървъра, когато се използва ефективно операционната система. Всяка архитектура трябва да се приема като компромис между стабилността и производителността.

Многопроцесорните архитектури са полезни при обслужване на клиентски заявки, изискващи повече време или при обслужване на сесии, когато трябва да се запази комуникацията за дъщерния процес между клиента и сървъра.

Предварително разклонени многопроцесорни архитектури

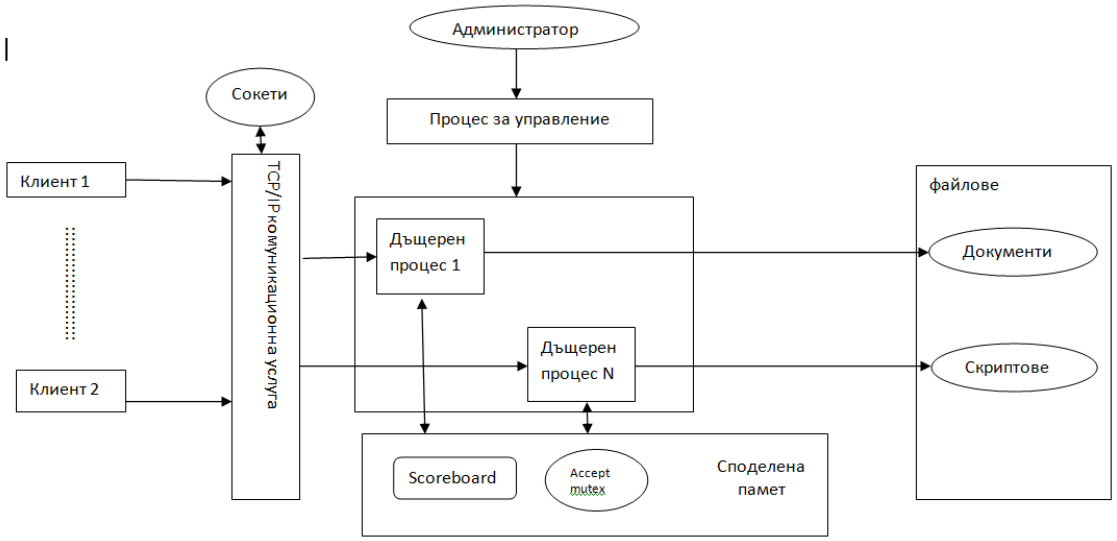
Модел лидер-последователи (leader-followers pattern)

Този модел на предварително разклонената (Preforking) архитектура се основава на пул от задачи, където всички задачи (процеси/нишки) изпълняват три различни роли: wait for requests (listener) – изчакване за заявки; process a request (worker) – обработва заявки; queue in and wait to become the listener (idle worker) – нарежда на опашката заявките и изчаква да стане лидер. Предимства на този тип архитектура са, че пристигащите заявки се обработват веднага от listener за задачата. Не трябва да се създава нова задача и винаги има определен брой свободни worker за обработка на задачите. Не е необходимо информация относно заявката да се подава на друга задача.

Модел Prefork

Prefork архитектурата е първата многозадачна архитектура на Apache и се използва по подразбиране в Apache 2.0. Тази архитектура е подобна на архитектурата от версия 1. Prefork архитектурата на Apache е традиционен подход, при който всеки дъщерен процес се обработва от самосебе си. Prefork архитектурата е стабилна и същевременно с намалена производителност и затова тя ще бъде обект на изследвания по-нататък в дисертацията.

Нововъведението в Apache 2.0, за разлика от Apache 1.3 е, че процесът за управление използва „pipe of death" вместо сигнали за изключването на дъщерните процеси при gracefull (елегантен) рестарт. Той извършва рестартиране на Apache, но оставя „живи" активните дъщерни процеси.



Фиг.I.11. Prefork архитектура

Prefork архитектурата (фиг.I.11) разполага с един процес за управление и много дъщерни процеси, които са създадени преди да пристигнат заявките. Процесът за

управление създава дъщерните процесите чрез системното извикване fork() и от това произлиза наименованието на архитектурата, наречена prefork. Пристигащите заявки се обработват директно от дъщерните процеси и не изчакват процеса за управление.

Процесът за управление е създал дъщерни процеси, които са регистрирани в TCP/IP комуникационната услуга за приемане на следващата заявка. Така първият дъщерен процес приема свързването, изпълнява заявката, изпраща отговор и започва да изчаква за следваща заявка. Процесът за управление самонастройва броя на свободните дъщерни сървъри в определени граници. Дъщерните процеси се създават чрез системното извикване fork(). Те имат отделни области от паметта и не им се позволява да четат и пишат върху памет на други процеси. Това е добър начин дъщерният процес да се конфигурира единствено от процеса за управление, който е различен от всички дъщерни сървъри. Конфигурацията се съхранява в областта на споделената памет, което позволява да се чете от всеки дъщерен процес. Дъщерните сървърни процеси се клонират от процеса за управление и поради това имат същата конфигурирана информация, която никога не се променя.

- Комуникация между процесите

Apache се контролира чрез сигнали. Сигналите са вид софтуерни прекъсвания. Те могат да възникнат по всяко време на изпълнение на програмата. Процесорът спира нормалното изпълнение на програмата и обработва сигнал на handler процедура. Handler по подразбиране обикновено се използва за прекъсване на настоящата програма. След като се изпълни настоящият handler, процесорът се връща към нормално изпълнение, освен ако програмата не е била прекратена от сигнал.

- Управление на процеса за управление от администратора

Администраторът може да изпраща сигнали директно чрез командата kill в командния прозорец или написан от него код. Процесът за управление реагира на три сигнала: Sigtern за изключване, Sighup за рестартиране и Sigusr1 за елегантно рестартиране, които са регистрирани в процедурата set_signal.

Поведението на сигналните handler на процеса за управление се определя от една малка мрежа на Петри. Apache активира и деактивира сигналните handler в определени точки на инициализация и на рестарт.

- Управление на дъщерните процеси от основния процес

Администраторът управлява процеса за управление (master server) единствено чрез сигнали, а процесът за управление използва или сигнали, или pipe (именувани канали), за

да управлява броя на дъщерните процеси и pipe of death. При shutdown (изключване) или restart (рестарт), процесът за управление изпраща sighup сигнал до групата на процеса. Операционната система ги „разпределя” до всички дъщерни процеси, създадени от процеса за управление. Ако не се прекратят всички дъщерни процеси, трябва да се използва по-силно средство.

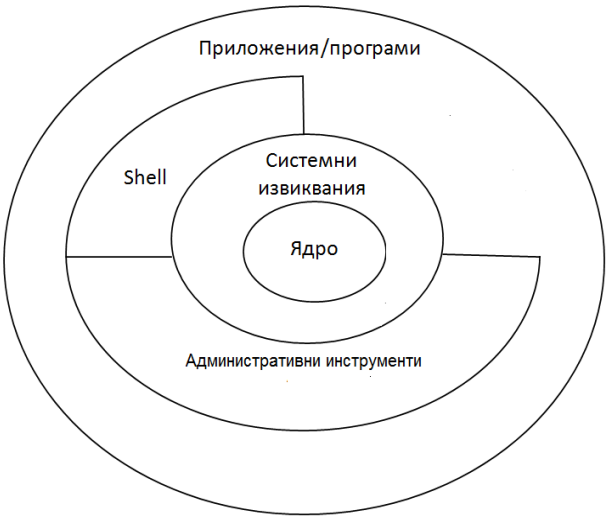
- Приемане на свързванията - Accept mutex

Prefork архитектурата използва accept mutex за разпределяне на входящите свързвания между множеството дъщерни сървъри. Accept mutex (взаимно изключване) е средство за контролиране на достъпа до TCP/IP услугата. Използва се, за да гарантира, че в даден момент само един процес изчаква за TCP/IP свързваща заявка. В архитектура с множество дъщерни процеси, които изчакват да се обслужат от множество портове, комбинацията от системните извиквания select () и accept () не е приемлива, за да се получи взаимно изключване между workers.

I.3.3. Операционна система Linux

За да се извърши качествен анализ на производителността на компютърната система, е необходимо да се познават добре операционната система и нейното ядро. Това ще позволи да се изградят подходящи хипотези и изследвания на системата относно работещите процеси, планирането на нишките, ограниченията на паметта, обработката на входно/изходните процеси.

Linux е многопотребителска операционна система с времеделение. Операционната система представлява софтуер, чиято основна цел е да свърже потребителските програми с хардуера, който да изпълни поставените задачи. Linux (GNU/Linux) е общо наименование за всички операционни системи с ядро Linux и различни системни извиквания и библиотеки, свързани към него. Структурата на Linux е показана на фиг. 12.



Фиг.I.12. Структура на Linux

I.3.3.1. Ядро на операционната система

Ядрото е програма, която управлява планирането на процесора, паметта, мрежови протоколи, мрежови карти и дискове. Тя предоставя достъп до устройствата и услугите на ядрото (kernel services) чрез системни извиквания.

Изпълнението на ядрото се осъществява при поискване от програма, работеща в потребителското пространство чрез системно извикване или когато устройство изпрати прекъсване.

Натоварвания, които генерират чести входно/изходни операции като при уеб сървърите, се изпълняват в контекста на ядрото. Натоварвания, които генерират интензивни изчисления, по възможност се изпълняват самостоятелно, за да не възникват прекъсвания на работата на процесора. Смята се, че ядрото не може да влияе на производителността на такива процеси.

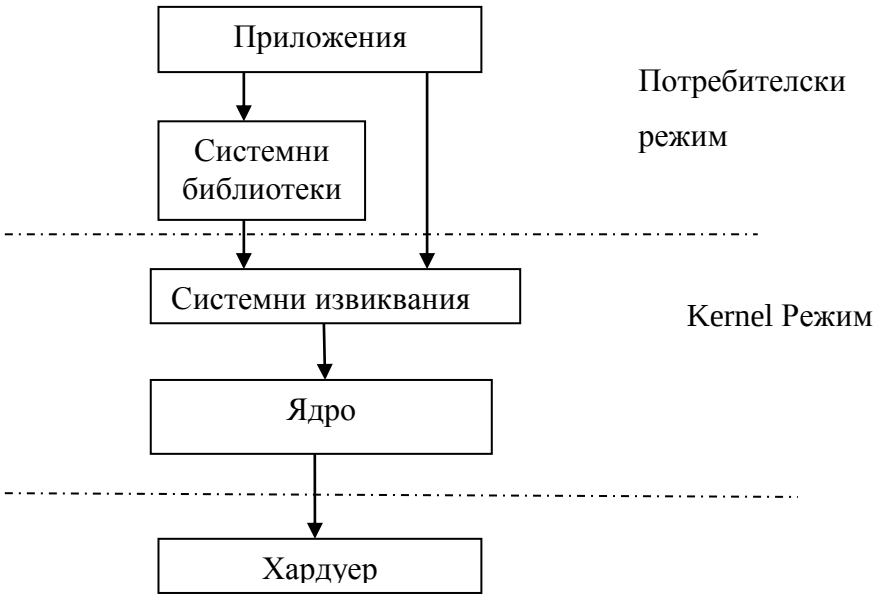
Ядрото избира процесора, който ще изпълни нишката, въз основа на характеристики като по-подходящо хардуерно кеширане и по-добра уседналост на процеса в паметта. Това от своя страна подобрява производителността на системата.

I.3.3.2. Режими на операционната система

Процесите, работещи под Linux се изпълняват в два режима – потребителски и режим на ядрото. Връзките между режимите на работа и хардуера са показани на фиг. 13.

Режим на ядро (kernel mode)

Ядрото е единствената програма, която работи в специален режим на процесора, наречен kernel mode, позволяващ пълен достъп до устройствата и изпълнение на привилегировани инструкции.



Фиг.I.13. Режими на работа на Linux

Потребителски режим (user mode)

Потребителските програми работят в потребителски режим, където те изискват привилегировани операции от ядрото чрез системни извиквания, такива като входно/изходните. За да удовлетвори системните извиквания, операционната система ще превключи от user към kernel режим на работа и тогава ще се изпълнят операциите от по-високо приоритетно ниво.

Всеки режим има собствен софтуерен изпълним контекст, включително стек (stack) и регистри. Изпълнението на привилегировани инструкции в user режим причинява изключения, които се управляват от kernel режима. Превключването между режимите отнема процесорно време и добавя пренатоварване за всички входно-изходни операции.

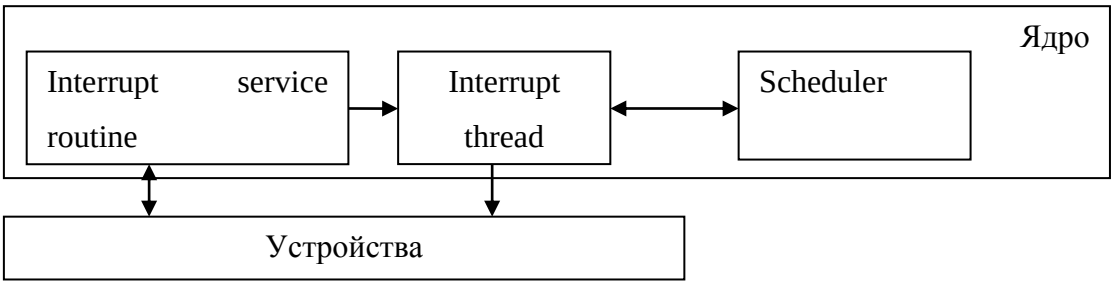
Разликата между user-stack и kernel-stack е използването на различно място в паметта, т.е. различна стойност за stackpinter регистъра и различни права за достъп до паметта. Стекът съдържа данни за изпълнението на предшестващата нишка с оглед на функции и регистри. Стековете се използват от процесорите за обработка на функции в софтуера. Инспекцията на стека е безценен инструмент за отстраняване на грешки и за

анализ на производителността. Стекът показва пътя на системното извикване до текущото състояние и често разкрива защо определена операция се изпълнява.

В рамките на потребителкия режим се извикват и системните библиотеки и приложения. Системните библиотеки се използват за предоставяне на по-богат и по-лесен интерфейс за програмиране от системните извиквания. Приложенията са програми, работещи в потребителското ниво като уеб сървъри, база данни, shell и административни инструменти.

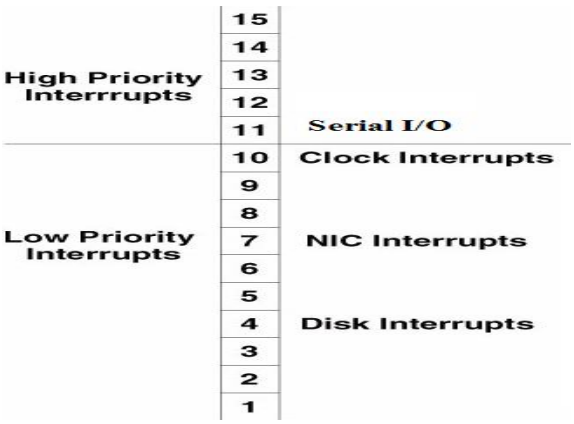
Прекъсвания

Ядрото е отговорно както за системните извиквания, така и за обслужването на заявки от устройствата, наричани interrupts (прекъсвания), понеже последните прекъсват текущото изпълнение. Interrupt service routine е софтуер за обработване на прекъсванията от устройствата. Неговата задача е да обработи прекъсванията възможно най-бързо и да намали действието на прекъсващите активни нишки (Фиг.I.14).



Фиг.I.14. Схема на софтуера за обработване на прекъсванията

Приоритетно ниво на прекъсванията (Interrupt priority level) изпълнява приоритета на текущия активен прекъсващ сигнал в interrupt service routine. Принципът му на действие е следният: по време на доставянето на прекъсващ сигнал, прекъсващото приоритетно ниво (IPL) чете от процесора и сравнява; ако нивото на подадения сигнал е по-високо от текущото, то го изпълнява; в противен случай го нарежда на опашката, за да се изпълни по-късно. В нивата на прекъсване дейностите на ядрото (kernel service) са с висок приоритет от 1 до 10 и се изпълняват като прекъсващи нишки, следвани от Serial O/I. Те също имат висок приоритет, поради малкия хардуерен буфер. Последното изисква бърза обработка, за да се избегне overflow (фиг.I.15).



Фиг.I.15. Приоритетно ниво на прекъсванията

I.3.3.3. Процес в Linux

Процесът е изпълнение на програма на потребителско ниво. Състои се от памет на адресните пространства, файлови дескриптори, стекове на нишките и регистри. Процесите представляват множество от задачи за ядрото. От друга страна, ядрото поддържа изпълнението на хиляди процеси в системата. Всеки процес се идентифицира чрез индивидуално и уникално ID на процеса – PID.

Процесът съдържа една или повече нишки, които работят в адресното пространство на процеса и споделят едни и същи файлови дескриптори. Нишката е изпълним контекст, състоящ се от стек, регистър и програмен брояч. Множеството нишки позволяват един процес да се изпълни паралелно от множество процесори.

Scheduler

Unix базираните операционни системи са времеразделящи системи. Този вид организиране на работата на процесите позволява на множество процеси да работят по едно и също време, чрез разделяне на времето за изпълнение между тях. Scheduler е програма, която извършва планирането на използваните процесори и индивидуалните ядра на процесорите за обработката на процесите. Основната му задача е да се раздели процесорно време между активните процеси и нишки, и да изпълни задачите с по-висок приоритет по-рано.

Scheduler проследява всички заявки в състояние ready-to-run (готови за изпълнение) и традиционните per-priority опашки наречени run queues.

Процесорните приоритети могат да се модифицират динамично от Scheduler, за да се подобри производителността на определени работни натоварвания. Работните натоварвания могат да бъдат категоризирани като:

- Cpu-bound: приложения, които изпълняват тежки изчисления за математически и научен анализ, и изискват дълго време за работа. Те биват ограничени от ресурса на процесора.
- I/o-bound: приложения, които изпълняват малки изчисления, като уеб сървъри, файлови сървъри и взаимодействащи си шелове, където малката латентност на отговорите е желателна. Когато натоварването се увеличи, те са ограничени от входа/изхода на дисковото устройство и мрежовите ресурси.

Scheduler може да се съобрази с тези изисквания за приоритетите, но трябва да се изчисли съотношението между текущото време за изпълнение и изтеклото време, и да се намали приоритета на процесите с високо съотношение. Този механизъм дава предпочитания на по-късите стартирани процеси, т.е. на входно/изходните процеси и на интерактивни процеси.

Модерните ядра поддържат множество планиращи класове, които прилагат различни алгоритми за управление на приоритетите и изпълнение на нишките. Те могат да включват реално времеви планиращи класове, които да използват приоритет по-висок от всички некритични задачи и нишки на ядрото.

I.3.3.4. Системни извиквания

Системните извиквания изискват от ядрото да изпълнява привилегировани системни действия и представляват интерфейс на ядрото на операционната система. Броят на системните извиквания може да достигне до стотици, но за да се запази простотата на ядрото се използва възможно най-малък брой. По-сложни интерфейси могат да бъдат изградени в адресното пространство, когато се използват системни библиотеки, което улеснява използването и поддържането на системните извиквания.

Системните извиквания са като цяло прост и последователен интерфейс с възможност за настройване на специална променлива за идентифициране на грешка и на нейния вид. Най-използваните системни извиквания са представи в таблица I.1:

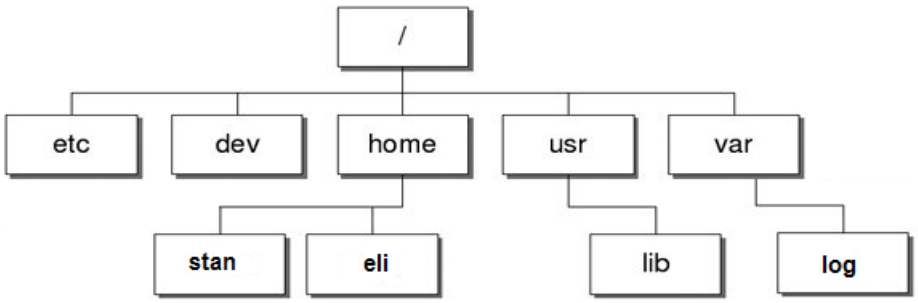
Таблица.I.1. Най-често използвани системни извиквания

Системно извикване	Read()	Write()	Open()	Close()	Accept()	Fork()
Описание	Чете байтове	Записва байтове	Отваря файл	Затваря файл	Приема свързване от мрежата	Създава нов процес

I.3.3.5. Памет и Файлова система

Файловите системи са организация на данни във вид на файлове и директории. Те имат файл-базиран интерфейс за достъп до тях, който обикновено се основава на posix стандарта. Ядрата поддържат множество видове файлови системи. Най-важната задача на операционната система е осигуряване на файлова система.

Операционната система предоставя глобални наименувани файлови пространства, организирани в топология дърво, която започва от ниво root (/). Добавянето на други файлови системи към дървото се осъществява чрез опцията mount и поставяне на собствено дърво в директория mount. В най-горното ниво на дървото са директориите: etc – конфигурация на системни файлове; usr – достъп до системни програми и библиотеки, достъпни на ниво потребител; dev – съхранява файловете на устройствата; var – съхранява различни файлове като log, tmp; и home – потребителска директория (Фиг.I.16).

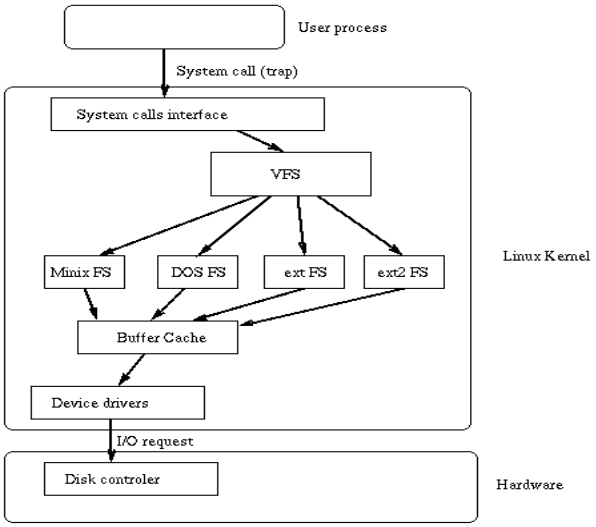


Фиг.I.16. Йерархия на файлова система

Повечето видове файлови системи използват дискове за съхранение на тяхното съдържание, но има динамични файлове, създадени от ядрото, като /proc и /dev.

- Виртуална файлова система

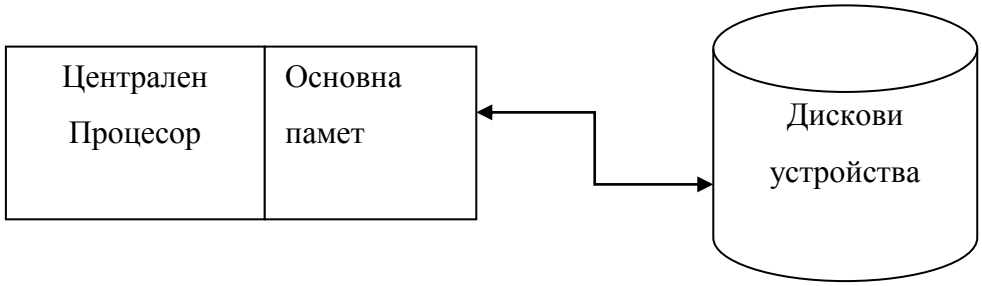
Виртуална файлова система (Фиг.I.17) е интерфейс на ядрото към абстрактни видове файлови системи. Такива са ufs (unix file system) и nfs. Интерфейсът на виртуалната файлова система осъществява по-лесно добавянето на нови видове файлови системи към ядрото. Той също така поддържа предоставянето на глобални именувани файлови пространства така, че потребителските програми и приложения да имат достъп до различни видове файлови системи.



Фиг.I.17. Виртуална файлова система

- Виртуална памет

Наличната оперативна памет в компютърните системи е с ограничен капацитет и това представлява проблем при изпълнение на много големи програми, работещи с голямо количество данни. За решаването на този проблем се извършва добавяне на външна памет към основната. По този начин се създава абстракция, наречена виртуална памет, която представя основната памет като безкрайна за процесите и ядрото. Тя позволява ядрото и процесът да се изпълняват в собствени адресни пространства без притеснения за капацитета, за поддържането на многозадачност и за презаписването на основната памет. Това означава, че чрез виртуалната памет се дава възможност на операционната система да свърже основната памет и хард диска (фиг.I.18). Създаването на виртуална памет се осъществява чрез допълнителни настройка на операционната система.



Фиг.I.18. Комуникация между основната памет и дисковите устройства

- Кеширане

При обработка на входно-изходните данни от диска се добавя голяма латентност. Много от входно-изходните стекове се опитват да избегнат това, като осъществяват четене на кеширани данни и записването им в буфери. Кеширането може да се осъществи чрез различни програми като кеширане от страна на уеб сървъра, например в Apache cache, кеширане в сървър за кеширане – memcache, кеширане в база данните – Mysql кеширане, segvn буфер кеш на операционната система и др.

I.3.3.6. Комуникация с други устройства

Ядрото на операционна система Linux разполага със стек с вградени мрежови протоколи, които позволяват на компютърната системата да комуникира по мрежата и да бъде част от разпределени системи. Така например стекът TCP/IP използва протоколите TCP и IP. Приложенията, работещи в потребителското ниво, достигат до мрежата чрез крайни точки, наречени sockets.

Поради необходимостта ядрото да комуникира с голям брой физически устройства се изисква създаването на софтуер за устройствата, наречен **drivers (драйвери)**. Това е софтуер на ядрото за управление на устройствата и вх/изх данни. Драйверите се разработват от производителите на устройствата. Те могат да имат plugged драйвери, т.е. да се зареждат и премахват с поставянето и отстраняването на устройствата, например usb flash. Драйверите изпълняват поредица от операции, които могат да се разделят в две категории – blocks и characters. Блоковете (blocks) съхраняват данните и изпълняват операции с определен размер. Достъпът до блокове е на произволен принцип. Такъв тип драйвери имат твърдите дискове и cd-rom-ите. Character съхранява и предава данни с произволен размер. Той също така генерира прекъсване след всеки байт или изпълнява вътрешно буфериране. Такива драйвери имат мишки, модеми, клавиатури [SilGP-98].

I.3.4. Анализ на модели на софтуерни системи

Изследването на дадена компютърна система може да се използва за различни цели, но предимно за анализ на мащаба, т.е. изменението на производителността при промяна в натоварването или изменението на използването на ресурса. Например при хардуера се оценява производителността на процесора при изменение на ядрата на процесора, а при софтуера – изменение на процесите или нишките води до промяна на производителността. Мащабируемостта измерва възможността на системата, мрежата или процесът да се справят с нарастващия обем работа, или казано по друг начин, възможността да бъде

разширена системата, за да се справи с нарасналото потребление. Производителността (Performance) е мярка за количеството работа, извършено от компютърната система. Ефективността на една компютърна система в зависимост от контекста, в който се използва, се определя от време за отговор, бързодействие, натоварване на хардуерните ресурси, достъпност до компютърната система или определено приложение, капацитет на използваната мрежата и др. [ArnoA-94].

I.3.4.1. Методи за анализ на софтуерни системи

Изследването на софтуерната система може да се осъществи по няколко начина: чрез оценка на производителността, измерване на параметрите на системата, експериментални изследвания, симулации и аналитични модели.

Производителността на системата най-ефективно се определя чрез използване на два или повече от гореспоменатите подходи [BoGMT-05], [Balbo-07].

Оценяване на производителността

За да се намерят и установят проблеми на производителността се извършва оценка на системата, известна като Performance Evaluation. Оценката съдържа показатели като бързодействие (throughput), използване на ресурс, време за отговор. Чрез тях се откриват ограниченията на системата, както и проблеми, свързани с производителността при проектиране на компютърна система.

Измервания

Измерванията са директна оценка на реална или предварителна (тестова) система при различни работни натоварвания. Оценката на производителността по този начин предоставя най-точни резултати. Измерванията се използват за различни цели като например получаване на показатели на производителността за оценка на софтуера, идентифициране на ограниченията на системата, оценка на параметрите на модел на производителността и оценка на модела [SmitW-02].

Измерванията на процесите и създаване на техни „профили” са полезни, понеже предоставят статистически данни по отношение на изпълнението на софтуера, показват графики на системните извиквания, отразяват връзката между извиканите функции на операционната система и извикваните функции от приложението.

Експериментални тестове

Оценяването на производителността чрез тестове за натоварване дава задоволителни и приемливи резултати. При този метод се използва натоварващ генератор, който създава виртуални потребители с цел имитиране поведението на реални

потребители на системата. Когато се прилагат тестове за натоварване, могат да се наблюдават и оценяват системата и/или приложението. Данните се записват, анализират и се прави извод за производителността [SmitW-02].

Симулация

Симулацията и аналитичното моделиране са подходи за моделиране на производителността. Симулацията е софтуерно представяне на системата и нейните взаимодействия. Събират се статистически данни от изпълнението на софтуерния модел и се предоставят показатели за производителността [KounB-03]. Тези модели се отличават с висока точност, но е необходимо време за разработването и симулирането им. Те са важни и значими за големи системи, при които е нужно много точно предсказване [BoGMT-05], [KounB-03]. Управлението на тези модели е предизвикателство, тъй като при промяна в проектирането на система, моделът става неточен.

Аналитичен модел

Аналитичният модел се представя чрез математически уравнения и тяхното решение. Той показва по-ниска точност в сравнение с другите методи за оценка. Неговите предимства се крият в по-бързото създаване и решаване на математическите задачи, които са лесни за манипулиране и са полезни във всички етапи на разработване на софтуера. Чрез тях изключително бързо може да се оцени производителността на системата. Едни от най-известните методи за аналитично моделиране са вериги на Марков, мрежи от опашки и мрежи на Петри [PasRO-08].

I.3.4.2. Моделиране на производителността на уеб сървъри

Моделирането на производителността е основна област на изследване при уеб сървърите. Точният модел представя прогноза на показатели на производителността и за планиране на необходимия капацитет на ресурсите за изпълнение при различни настройки. Примери за такива изследвания са описани в [HuDoS-98], където на базата на емпирични изследвания се анализират ограниченията и относителната производителност на уеб сървъри при ATM мрежи, като целта е да се постигне оптимална производителност. В статия [MenaA-98] се анализират уеб приложенията и работните характеристики, които въздействат върху производителността на уеб сървъра и се разработват модели от опашки за хардуера на сървъра, за да се определят компромиси между производителността и необходимия хардуер при големи натоварвания. Друга важна особеност е управлението на пренатовареността (overload) на уеб сървъри, които поддържат популярни сайтове, като целта е да се получи приемливо време за отговор [CaANM-01]. Изследвания показват, че

когато се използват стратегии за управление, т.е. начина на постъпване на заявките от заявки (request) в сесии (session), се наблюдават определени подобрения в режим на пренатовареност [CherP-99], [Widel-02].

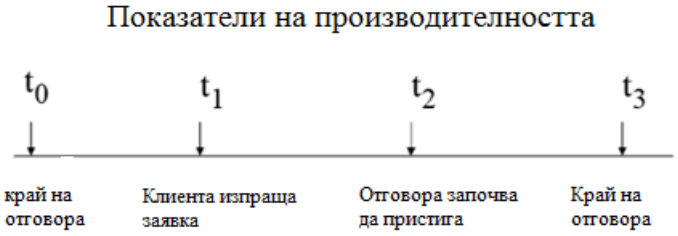
Представянето на работните характеристики на натоварванията на уеб сървър е направено в статии като [CrovB-97] и [ArliW-97]. В тях се показва, че уеб трафикът е интензивен и произволен в различни времеви скали (timescale), но е себеподобен (self-similar). Разпределенията на размера на файловете, времената на предаване (time transmission) и време за мислене (thinking time) са във вид на дълги опашки (heavy tail).

В статията [Mahba-97] изграждат модел на мрежовия трафик, предоставян от http чрез използване на уеб приложения. Моделът е съставен от поредица вероятностни разпределения (probability distributions), определени чрез анализ на реални http комуникации. От проследяването на пакетите се изгражда модел на комуникация на по-високо ниво. Чрез генериране на уеб трафик неговите характеристики показват, че http заявките са бимодално разпределени, а отговорът на http заявките има разпределение тип „дълга опашка“. На базата на наблюдавания http трафик и на статистически изследвания се определя съхранението на файлове и техните препратки.

В [LiuNJ-01] се изследва стохастичен модел на http трафика на сесийно ниво, когато се извършва статистически анализ на данните на сървърите. Изследва се поведението на потребители чрез хиперлинковете, за да се установи кога са посетили уеб страницата, колко заявки са генерирали, поредицата от кликвания в сесията, времето между кликванията, предаването на данните и използването на процесора.

За да се опише поведението на уеб сървъра се използват някои показатели на производителността на най-ниско ниво. По този начин се изчисляват времена на работа на системата:

Показателите на производителността на най-примитивно ниво са илюстрирани на фиг. I.19.



Фиг.I.19. Показатели на производителността

t_1-t_0 –време за размисъл (thinking time), т.е. време за изчакване на следващата заявка;

t_2-t_1 – време за реакция от страна на сървъра;

t_3-t_1 – време на отговор – това е времето от изпращане на заявката до връщане на нейния отговор.

На базата на тези измервания се изчисляват показатели като: average response time (средно време за отговор); средно бързодействие, т.е. заявки/секунда; средна дължина на опашката (броя на заявките, които очакват да бъдат обслужени); вероятност от блокиране, т.е. вероятността заявката да бъде отхвърлена. Тези показатели са удобен начин за анализиране на системите и се използват в теорията на опашките.

За изготвяне на по-задълбочени изследвания на уеб сървърите и компютърните системи се използват QN мрежа от опашки.

Един от първите анализи на производителността на уеб системи е направен в статия [Sloth-96], където се разглеждат четири състояния на отворен QN модел: клиент, уеб сървър, мрежа и статичен файл. Целта на статията е да се представи модел на връзката между уеб сървъра и мрежата. Освен това моделът е разширен за оценка на ефектите от множество уеб сървъри.

В [Menas-03] е представен модел на софтуерна и хардуерна архитектура на един уеб сървър, като се използват Маркови вериги и мрежи с опашки. Статията описва влиянието на софтуерните конфликти върху производителността на уеб сървъра и показва поведението на модела на производителността при динамично конфигуриране на системата. Ефектите на конфигурацията на уеб сървъра се основават на избора на обработващ модел и поведението на размера на пула (pool-size).

В [KounB-03] се описва затворен модел на опашки при изследване на натоварването на SPECjAppServer2002. Моделът е изграден от клиент, сървърно приложение във вид на клъстер и production line stations. Определя се общото време за обслужване на заявките от ресурса, като входните данни за модела са получени чрез законите на QN. Моделите се оценяват при различни натоварвания за брой потребители едновременно: ниски – 130, средни – 250 и високи – 350. Моделите се оценяват чрез измервания, а резултатите показват висока точност на предсказване на производителността.

В [UPSSA-07] е представен затворен QN модел на многослоести интернет приложения, които се състоят от кеширане, ограничение за едновременно обработване на задачи и множество сесии, основаващи се на класове на заявките. Моделът е оценен чрез

две приложения като Rubis и RuBBoS, които се изпълняват на Linux клъстери. Моделът е бил използван за обезпечаване на ресурса с непрекъснато променящи се изисквания.

В [LiuHS-05] се описва затворен QN модел на трислойно уеб приложение, състоящо се от Apache уеб сървър, Tomcat приложение и mysql база данни сървър. В модела са зададени ограничения на едновременни заявки, както и максимален брой едновременно изпълнявани заявки/процеси за Apache и Mysql. Използват се опашки в много възли. Решението на модела се извършва чрез анализ на средните стойности. Изследването на натоварването се извършва чрез TPC-W емулатор и резултатите се използват за оценка на модела.

В статиите по-горе се докладва моделиране на уеб системи и техните измервания. Но описанието на големи софтуерни и хардуерни системи, софтуерните конфликти и тяхното влияние върху производителността е трудно осъществимо с използване на мрежа от опашки, поради което тя се заменя с разширени мрежи от опашки (LQN). Чрез последните може да се моделира сървър, както е демонстрирано в [TiwaM-06 и [UfimM-06]], но и да се представят приоритетни взаимодействия и паралелни изпълнения като and-fork и or-fork, и да се изгради и оцени модел [Frank-99].

В [TriMW-10] се използва UML за представяне на модел на мобилна система за плащане, където UML моделът се преобразува в LQN модел на производителността. LQN анализът показва, че ако се използва еднопроцесорен сървър, това е ограничение на ресурса. Извършва се оценка за използване на процесор с две ядра.

Моделирани са и други системи като CGI уеб сървър и joomla система, състояща се от Apache-php-mysql чрез използване на LQN. В [DiFJR-98] моделът на CGI сървъра включва обслужване на динамични и статични страници, но не се изследва база данни. Времето за отговор от оценката на модела е много близко до времето за отговор за период от два месеца. Данните са събрани чрез програма, разработена от авторите. Те служат като входни данни за LQN модела и са използвани за валидиране му. В [PasRO-08] се извършват натоварващи тестове при различен брой потребители от 1 до 40, за да се предскаже производителността на системата. Резултатите показват добро предсказване на производителността чрез модела. Системата се състои от два уеб сървъра – система за балансиране на натоварванията и сървър за база данните. За моделиране на производителността и за оценяване на показателите ѝ се използват и други формализми като Stochastic Process Algebra и Stochastic Petri Nets.

I.4. Изводи

Целта на глава първа бе да направи обзор на постигнатото в областта на прилагане на теория на автоматичното управление при различни програмни системи и да очертае насоките за експерименталната част на настоящето дисертационно изследване. В тази връзка бяха разгледани основни принципи за изграждане и методи за анализ и синтез на системи за автоматично управление. Описани бяха и различни софтуерни приложения, при които се използва управление с обратна връзка, каквато е и уеб сървъра Apache. Бе установено, че теория на автоматичното управление е приложима за различни сложни системи за управлние като компютърни мрежи, операционни и самонастройващи се системи. Тази част от текста се концентрира върху особеностите на уеб сървъра Apache, в качеството му на програмна система, както и върху тези на операционната система Linux.

В глава първа бе установено, че използването на системи от опашки за моделиране на производителността на компютърни системи осигурява отличен начин за изследване на различни проблеми в последните, разглеждайки ги като мрежа от опашки и сървъри. Те ефективно моделират поведението на компютърните системи в установено състояние. Но трябва да се има предвид, че при опити за моделиране на динамиката на компютърните системи, тяхното използване създава редица затруднения. Това означава, че за описание на динамиката на изследваната в този дисертационен труд система ще се наложи прилагането на линейни диференциални уравнения. Те са характерно средство за описание на системите в рамките на теория на автоматичното управление и са изключително полезни при изграждането на модели чрез статистически данни или входно-изходни данни.

Този подход за проучване на системите се разглежда и в теорията за идентификация на системи. В рамките на дисертационния труд разбирането на техните възможности и ограничения се търси чрез генериране на входни въздействия към системата и със софтуерни инструменти за измерване, наблюдение и диагностика на компютърната система. Това е необходима стъпка по посока на една от основните задачи на дисертацията – прилагане принципите на теория на автоматичното управление с цел получаване на адекватна информация за работата на програмната система.

II. МОДЕЛИРАНЕ НА КОМПЮТЪРНА СИСТЕМА ПРИ РАЗЛИЧНИ ВХОДНИ ВЪЗДЕЙСТВИЯ

II.1. Моделиране на компютърната система

За да се изготви модел на системата, трябва да се извърши анализ на хардуера и софтуера на конфигурацията, която се използва. За целта се прилагат различни входни въздействия и измервания чрез програми, работещи в операционната система. По този начин се определят техните предимства и недостатъци, анализира се производителността на цялата система. Анализът на производителността е изключително важен за всички потребители – от най-обикновения потребител, през поддържащия персонал на ниско ниво и разработчиците на софтуер до администратори на операционни системи, база данни и уеб сървъри. Чрез него те могат да разберат повече подробности за производителността на операционната система и приложенията, работещи на нея.

Има две основни цели, поради които се извършва анализ на производителността: първата е подобряване на съотношението цена/производителност, където и най-малките подобрения водят до намаляване на разходите за информационни технологии. Втората е намаляване на латентността, защото високата латентност при големи натоварвания забавя заявките към приложенията, а това разочарова потребителите на услугите.

Други дейности, свързани с производителността на системите, касаят извършване на експерименти с различни входни въздействия, за да се оценят компютърните системи и софтуера, да се планира капацитета, да се отстранят ограниченията на системата, да се анализира и коригира мащабируемостта.

Наблюдаването на производителността на операционната система дава възможност за предпазване от различни неудобства като пропадане на уеб сайтове, бавна работа на машината, бавно обслужване на заявките към база данните и бавна работа на хард диска. Освен това подпомага откриването на източниците на проблеми – например пълното използване на наличния ресурс, изчакване в наситени опашки, генериране на грешки от пропадане на пакети, некоректни отговори, мъртви схватки (deadlocks) и изтичане на времето за заявка.

Анализът на производителността се извършва в две посоки – анализиране на работното натоварване и анализиране на ресурсите. Първата посока се използва основно от разработчиците на софтуер, понеже те се интересуват от възможностите на техния продукт. Втората се използва от системните администратори, които са отговорни за ресурсите на системата.

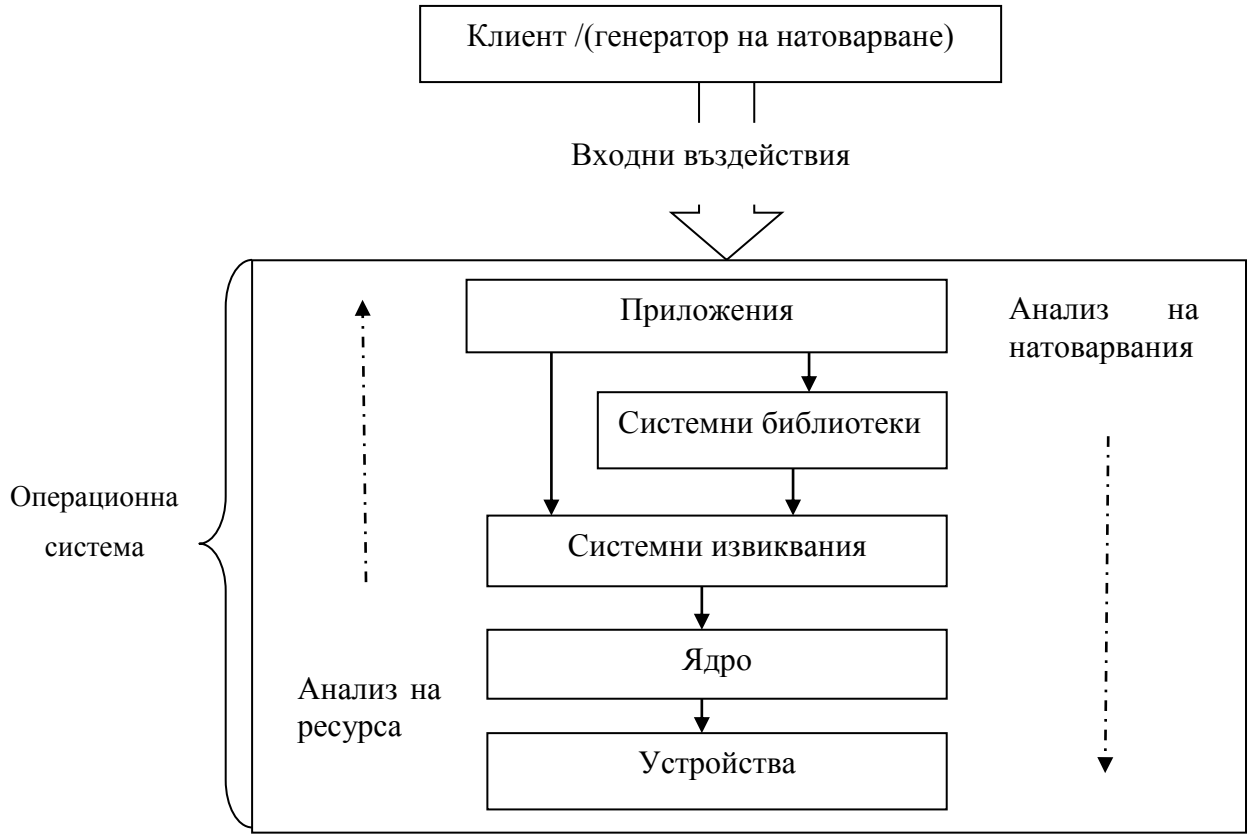
II.1.1. Анализ на хардуерния ресурс при различни входни въздействия

Анализът на ресурса е свързан с хардуера на компютърната система като процесори, памет, диск, мрежови интерфейси, шини и взаимовръзки. Дейностите, които се включват са: *изследване на проблеми на производителността,* т.е. преценява се дали даден вид ресурс е надежден, и *планиране на капацитета,* т.е. извличане на полезна информация относно необходимия размер на новата система и ограниченията на съществуващата. Анализът на ресурса се осъществява с цел идентифициране на лимита му. Това се прави поради факта, че някои ресурси, като процесорите, се отчитат с показател за използваност и са лесни за достъп. За други ресурси, като например мрежовата карта, на базата на налични показатели за изпратени и получени битове за секунда (mbps) (throughput) при известен максимален капацитет на мрежата (bandwidth), може лесно да се изчисли използваността (utilization)..

Достъпът до данните за изполваност се осъществява лесно с някои от инструментите на операционната система. Така например инструментът за наблюдение iostat извиква показателя %b в проценти заетост.

Други подходящи показатели за анализиране на ресурса са iops, throughput, saturation. Чрез тях се измерва колко операции извършва ресурсът, неговата наситеност и натоварване. До тези показатели може да се достигне чрез инструменти като vmstat, iostat, mpstat.

На фиг. II.1 е показано действието на анализа на ресурса и анализа на работното натоварване. На нея ясно се вижда, че при първия вид анализ действията започват от устройствата и стигат по приложения, а при втория анализ – действията са в обратна посока [Gregg-13].



Фиг. II.1. Анализ на ресурса и анализ на работните натоварвания

II.1.2. Анализ на софтуерните приложения при различни входни въздействия

Анализ на натоварването (Workload analysis) изследва производителността на приложенията като се прилагат работни натоварвания и се наблюдава как отговаря приложението. Този начин най-често се използва от разработчици на софтуер и от поддържащ персонал, който е отговорен за софтуерните приложения и конфигурациите им. Задачите, които са поставени при този анализ са: прилагане на работно натоварване на приложението чрез заявки (requests); измерване на неговата латентност (latency), т.е. времето за отговор на приложението; наблюдение за възникване на грешки (фиг. II.2).

Фиг. II.2. Анализ на работните натоварвания

Изследването на работни натоварвания чрез заявки обикновено включва проверка и обобщаване на техните атрибути, т.е. обработка на характеристиките на работните натоварвания. Например атрибутите за база данните са хост на клиента, име на база данните, таблица и стринга на заявката. Чрез тази информация се идентифицира ненужната или небалансирана работа, а това представлява метод за намаляване или елиминиране на възложената работа на база данните.

Латентността (времето за отговор) е най-важният показател за измерване на производителността на приложението. При база данните Mysql говорим за латентност на заявка, а при Apache server – за латентност на HTTP заявка. В този контекст латентността се използва за обозначаване на времето за отговор.

Задачи на работното натоварване са: търсене на латентност извън определен праг, идентифициране източника на тази латентност, анализ на приложени корекции върху латентността, както и проследяване на статуса на приложението, защото при грешки се причинява повторно изпращане на заявки и се генерира латентност.

II.2. Смущения в компютърните системи

Смущенията в компютърните системи оказват влияние върху резултатите на производителността на програмната система. Те може да са породени от много фактори като планиране на системни действия, други потребители или други натоварвания. В някои случаи произходът на смущенията е неясен и това изисква внимателно изследване на компютърната система за тяхното определяне. Така например, определянето на смущенията е трудно при сложни системи като облачните технологии, където хостващата система не наблюдава вътрешните действия на потребителите. Това важи и за системи със свързани мрежови компоненти за обслужването на входящи работни натоварвания – системи за балансиране на натоварванията, сървъри на база данни, сървъри на приложения и системи за съхраняване на данни.

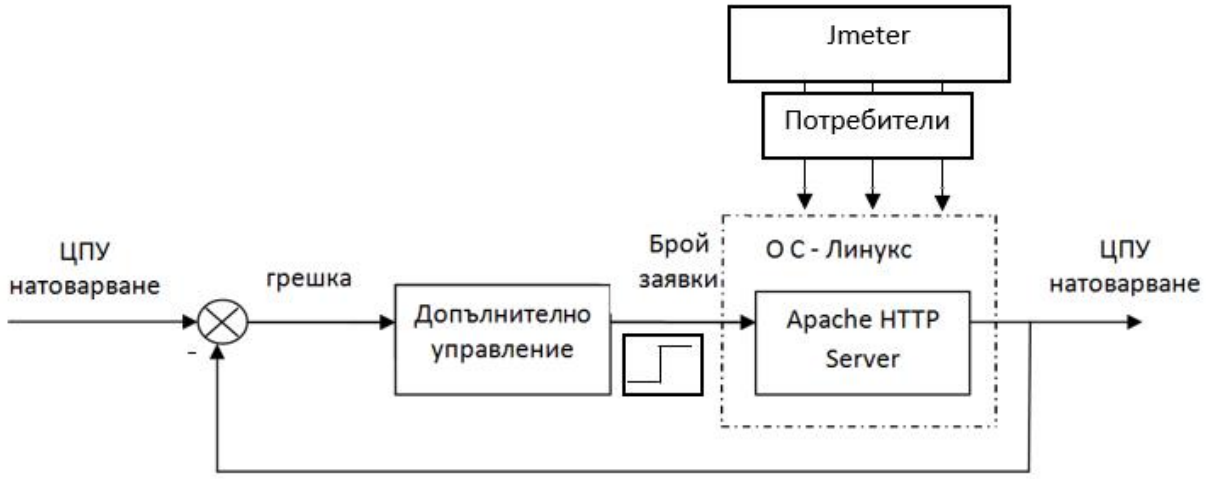
Фиг. II.3. Смущения в компютърни системи

Откриването на смущенията може да се осъществи чрез изграждане на блок-схема на работата на сложна система или чрез изграждане на модел на средата, който да включва мрежа от системни опашки за аналитично изследване.

II.3. Структурна схема на изследваната компютърна система

По своята същност компютърните системи представляват сложни системи за управление, в които протичат много процеси едновременно. Определени процеси могат да се разглеждат като входни въздействия за системата, други – като смущения за системата, а трети – като изходни данни от нейната работа. Това позволява компютърната система да се представи като система за автоматично управление (САУ). Според теория на информацията, САУ е информационна система, която преобразува съответната информация за постигането на определени цели като регулиране, следене и програмно управление [СапуГ-07]. За изграждането на система за автоматично управление е необходимо да се проведат експериментални изследвания и да се създаде математически модел на обекта. Експерименти за компютърната система могат да се реализират посредством анализа на натоварването и анализа на ресурсите, за да се определи моделът на изследвания обект – уеб сървъра Apache. Използваната компютърна система за провеждане на експериментални изследвания в дисертационния труд работи под операционна система Linux и инсталиран на нея уеб сървър Apache. Те съвместно осигуряват обслужването на потребители на софтуерното приложение за достъп до научно списание. При измерване на натоварването на процесите на Apache се реализира управление чрез програма, написана на bash скрипт и езика за програмиране AWK. Тя се изпълнява в потребителския режим на операционната система. Реализираната програма за управление следи нивото на натоварване на процесора от процесите, които създава уеб сървърът Apache. Промяната на броя потребители се явява измеримо смущение за натоварването на процесора. В нормален режим на работа то е много малко – 3-5%. При постъпване на заявки от до 20 потребителя на софтуерното приложение, той се натоварва с до 30 %. За ефективното им обслужване се въвежда първият вариант на настройки – highSetting, а когато натоварването достигне 60% се въвежда вторият вариант на настройките на Apache – bestSettings. По този начин допълнителното управление действа като двупозиционен регулатор.

Реализираната програма за управление преконфигурура настройките на Apache. Структурна схема на изследваната система е представена на фиг. II.4.



Фиг.II.4. Структурна схема на изследваната система

II.4. Методологии за анализиране на производителността

При сложна система или система в пренатоварен режим възникват няколко предизвикателства. Първото е посоката, в която ще се извърши анализа, а следващите са начина на събиране на данните и тяхното анализиране. Проблемите, които са възникнали при работа на системата може да са породени от различни източници като софтуер, хардуер или пътя до данните.

За да се осъществи анализ, се изискват последователни стъпки за провеждането на анализа като локализиране и анализиране на проблемите с производителността, списъци за проверка на липсващи данни, методи за количествено потвърждение на заключенията, идентифициране на важни проблеми.

Различните методологии за анализиране на производителността могат да се разделят в няколко категории [Gregg-13]:

- **Липса на конкретни действия**

Пример за такъв метод е **streetlight**, в който се анализира производителността чрез инструменти за наблюдение, които са произволни, познават се бегло или са намерени в интернет. По този начин може да се открие проблема, но има по-голяма вероятност да се пропуснат други немаловажни проблеми. Допълнително, използваната програма може да е бавна в откриването на проблема и приложените настройки да не са ефективни. Подобен подход е **random change –** прилагане на произволни промени в настройките, докато

изчезне проблема. За да се определи дали производителността се е подобрила се изискват показатели за приложението като време на работа (running time), операции за секунда, бързодействие и др. Това отнема дълго време за изследване и води до неясна ефективност на настройките. Друг метод от същата група е **Blame-someone-elseanti-method.** При него не се изследва производителността, а се избира един проблем и се поставя хипотеза. Насочва се към отговорните за това лица, те решават проблема и след това се избира друг проблем. **Tools** методът на свой ред се състои в изготвяне на списък с инструменти за анализиране на производителността и в избор на показателите от даден инструмент за интерпретиране. Може да се постигнат добри резултати при добро познаване на инструментите, но може да има пропуски с оглед много важни показатели, тъй като се изисква време за обработка и филтриране на данните.

- **Задаване на въпроси**

В тази група чрез задаване на въпроси се поставят хипотези, провеждат се експерименти и се анализират резултатите.

Декларирането на проблема е рутинно задължение на отговорния за това персонал след поставяне на въпроси. Това е първият подход, използван за проследяване на нови проблеми. Друг подход е научният метод, който е разширение на декларирането на проблема, като неизвестното се изследва още с предположение, експеримент и анализ.

Деклариране на проблем се извършва чрез задаване на въпроса, свързан с производителността. Изготвя се предположение за причината за ниската производителност, след което се извършва проверка, която може да бъде наблюдаваща или експериментална. Проверките се правят на базата на хипотези. Накрая се анализират събраните данни.

- **Диагностичен цикъл**

Методът е подобен на научния – след като се постави хипотеза, се провеждат изследвания и събиране на данни. На базата на тези данни може да се изготви нова хипотеза и да се филтрира ненужната информация. Така чрез малки проверки се събират данни и може лесно и бързо да се анализира системата. И двата метода са добра комбинация между практика и теория.

- **Методи за изследване на ограниченията**

При Ad-hoc метод чрез контролните списъци натоварването се увеличава стъпаловидно. Използва се от специалист, за да се провери и настрои системата за кратко време. Предимство на списъците е, че се изготвят на базата на последното изследване и са

подходящи за фокусиране и документиране на проблеми и отчитане на определени поправки.

- **USE метод**

USE методът е разработен от Грегг [Gregg-13] и е подходящ да се използва в началото на изследванията за производителността на системата, за да се идентифицират нейните ограничения. Принципът на работа на метода е в проследяване на използването, насищането и грешките на всички компоненти. За разлика от Tools метода, итерациите се включват над системния ресурс (чрез език за програмиране), вместо с инструменти. Това е изключително полезно по отношение на съставяне на списък с инструменти, които да анализират производителността. По този начин анализът се насочва директно към ограничен брой ключови показатели и се дава възможност да се провери по-бързо системния ресурс. При неоткрити проблеми се използват други методи.

USE методът е най-ефективен за системи, които страдат от намалена производителност, поради голямото им използване или насищане.

- **Анализ в определена насока**

- Workload analysis

Това е лесен метод за идентифициране на проблеми, дължащи се на натоварването [Agile-13]. Той се фокусира върху въвежданите данни в системата, а не върху резултатите на производителността. Представлява ефективен метод за системи, които нямат архитектурни или конфигурационни проблеми, а такива, дължащи се на по-голямо натоварване от това, което може да понесе системата. Методът е полезен при изграждане на следващи симулации, тъй като събира данни за изменението и разпределението на данните.

- Drill-down анализ

Методът започва изследването на определен проблем на високо ниво. На базата на резултатите във високото ниво се определят резултати, които представляват интерес за изследването и се извършва задълбочено проучване за тях. Процесът на откриване на главния проблем може да изисква задълбочено навлизане надолу по софтуерните слоеве до хардуера.

Този начин за анализ е изготвен от Solaris performance and tools и има три състояния [McDoR-07]: *мониторинг* – използва се за непрекъсното записване на статистика на високо ниво като идентифицира или алармира за проблем; *идентификация*– при заподозряване на проблем се стеснява изследването до определен ресурс или области

на интерес, идентифицират се възможни ограничения; *анализ* – разглеждат се конкретни области от системата, за да се опише качествено проблема.

- Анализ на латентността

Този анализ изследва времето за завършване на една операция, след което въпросното време се разпределя по компонентите, през които е преминала операцията. Латентностите се сортират по големина и така се открива проблемът. Методът е подобен на drill-down анализа, понеже отново се преминава през слоевете на софтуерния стек, за да се открие проблемът.

- Method R

Методът е предназначен за анализ на производителността и е разработен за Oracle база данни [MillH-03]. Фокусира се върху първоначалния източник на латентността и се основава на проследяване на oracle събитията. Той се описва като метод за подобряване на времето за отговор, който дава максимална икономическа стойност на бизнес решения. Може да бъде използван и за други приложения.

- **Проследяване на събития (Event tracing)**

Системите работят чрез обработка на дискретни събития. Видовете дискретни събития са: процесорни инструкции, входно/изходни задачи към твърдия диск, мрежови пакети, системни извиквания, извиквания на системни библиотеки, транзакции на приложения, заявки към база данни и др.

Анализите на производителност обикновено изследват обобщенията на тези събития като операции за секунда, битове за секунда, средна латентност. Това е недостатък, тъй като се губят важни подробности за работата на системата. Подходящо е събитията да се инспектират индивидуално. Например, при наличие на мрежови проблеми се извършва инспектиране пакет по пакет чрез инструменти като TCP dump и wireshark.

- **Статистически данни за работата на машината**

Когато се записват данни на показатели за производителността, последното е полезно за изследването на системата, например за целите на сравняването на моментна стойност с предходна такава. Това позволява да се идентифицират промените в натоварването, в използването на даден ресурс или при възникването на определен проблем. Събирането на тези данни се осъществява чрез различни инструменти за наблюдение или чрез log файлове. Статичните настройки на производителността изследват системата, когато не е натоварена и се интересуват от конфигурационните настройки на последната. Поради факта, че управлението в използваната компютърна

система се реализира чрез конфигурационни настройки и тя не е в натоварен режим този метод се прилага при анализа на резултатите на проведените изследвания в дисертационния труд.

II.5. Критерии за изграждането и функционирането на програмни системи

При изграждането на компютърна система, която се използва за поддържането на програмна система, могат да се извършват компромиси на различни нива. На етап проектиране на програмната система трябва да се вземе оптимално решение относно Време за отговор/Производителност/Цена. В някои от случаите производителността се поставя на втори план и това може да доведе до несъвместимост между езика за програмиране, архитектурата, операционната система и инструментите за анализ. За постигане желаната производителност на програмната система е необходимо да се вземе решение и за основните хардуерни компоненти – памет и процесор. В случаите, в които паметта се използва като кеш, се намалява използването на процесора. Възможен е и друг подход в системите с множество процесори – те да се използват за компресиране на данните, за да се намали използването на паметта.

Решението за настройка на софтуерно ниво на параметрите лежи в избора на размера на записа на файловата система (File system record size) или на размера на блока (blocksize). Ако размерът на блока е малък, т.е. близък до този за вход/изход на приложението, то ще изпълнява по-добре произволни работни натоварвания и ще направи използването на кеша на файловата система по-ефективно. Големият размер на записа ще подобри работата на системата при streaming (поточно) работно натоварване и при backup-а на файловата система.

Решението за настройка на ниво мрежова карта е Network buffer size. Малкият размер на буфера ще намали натоварването на паметта при свързването. Големият размер ще подобри мрежовото бързодействие.

Трябва да се отбележи, че има случаи като например при фондовите борси и high-frequency traders (първична форма на алгоритмична търговия в областта на финансите), където не може да се прави никакъв компромис, защото се изисква да се постигне висока производителност и ниска латентност на системата. За работата на тези финансови системи се полагат огромни усилия и финансови средства. Така например е планирано

поставянето на трансатлантически кабел между борсите в Ню Йорк и Лондон на стойност $300 000 000, за да се намали латентността до 6 ms [DeliM-11].

За да се разберат възможностите на операционната система, на хардуерните компоненти, на използвания софтуер и на допълнителни настройки, се използват инструменти за мониторинг.

II.6. Инструменти за мониторинг

Операционната система Linux позволява да се използват много инструменти за наблюдаване на софтуерните и хардуерните компоненти. Това дава възможност да се следи всичко в системата до най-малки подробности. Но инструментите имат пропуски и са необходими допълнителни усилия като извършване на индиректни измервания и използването на статистики от страна на системните администратори за тълкуване и извеждане на изводи от резултатите. Следващ етап в разработването на инструменти за наблюдение са инструментите за динамично проследяване на системата, което позволява да се наблюдават и визуализират директно всякакви дейности на операционната система и да се прилагат различни комбинации при наблюдаването им.

Има инструменти и ресурси, които проследяват уеб сървъри и събират данни за производителността им на много нива. Данните от измерванията, които обикновено се предоставят от инструментите за мониторинг, могат да бъдат групирани в две основни категории: такива на системно ниво и такива на сървърно ниво.

Инструментите за мониторинг на системно ниво предоставят статистически данни за цялата система (компютърна система) относно използването на ресурсите, като CPU, памет и хард диск. За тази цел се използват системни инструменти на операционната система като SAR за Linux или NT Performance Monitor за Windows.

Статистическите данни на ниво сървър могат да бъдат получени от log файлове. Уеб сървърите могат да бъдат конфигурирани така, че да записват информация за всички заявки, които се обработват от сървъра. Въпреки че са полезни за характеризирането на работните натоварвания, те не предоставят информация за времето, което е необходимо за обработка на заявките, нито такава за производителността и необходимите хардуерни ресурси за обработката им.

Разбиране поведението на уеб-сървърите е много важно, но трябва да се има предвид, че няма свободно достъпни инструменти за измерване на производителността, както и на времето за обработването на HTTP заявките. Поради тази причина в

дисертационния труд се реализират експерименти, които да предоставят необходимата информация за работата на уеб сървъра.

Инструментите за наблюдаване на производителността се основат на даден брояч (counters) или проследяване (tracing). Те могат да касаят цялата система или определен процес. Това разделение не е коректно, защото някои инструменти като dtrace и top могат да се използват по няколко начина, а други се използват за изготвяне на профил на компонент чрез поредица от измервания за определен процес или за цялата система. Най-често използваните програми в операционната система Linux за изследване на производителността са представени в таблица II.1.

Таблица II.1. Обобщена таблица за използваните програми

	За процес	Цялата система
Брояч	Ps	Vmstat
	Top	Mpstat
	Pmap	Iostat
		Sar
Проследяване	Strace	TCPdump
	Truss	Snoop
	Gdb	Dtrace
	Mdb	Stap

- Брояч (counter)

Ядрото поддържа различни статистики, наречени броячи, които броят събития. Те се изпълняват като целочислени при възникване на събитие (брой на получените пакети, изпълнени системни извиквания и пр.). Включени са по подразбиране и се поддържат в ядрото. Броячите могат да се използват в рамките на един процес или на цялата система. Недостатък е, че се изискват допълнителни усилия за четене от потребителите.

- Проследяване (Tracing)

Проследяване – събира данни от събития за последващо анализиране. Този начин на събиране на данни не е включен по подразбиране в операционната система, тъй като причинява натоварване на процесора и изисква дисково пространство за съхраняване на данните. Трябва да се отбележи, че използването на този вид измервания може да въздейства негативно на изследвания обект, а това е необходимо да се вземе под внимание при анализиране на времето за действие. В операционните системи по подразбиране се включва системен log (регистър), който е вид нискочестотно проследяване, което записва данни за събития, грешки и опасности.

- Мониторинг

Мониторингът е най-често използваният инструмент за наблюдение на операционната система и нейните действия. Sar е продукт, който записва състоянието на системните броячи и може да се използва с Linux административното приложение cron, което през определен интервал от време или чрез командния ред извършва дадено действие. Sar чете предишните статистически данни и записва текущи измервания. Освен това дава и допълнителни опции относно броячите и интервалите. Като цяло sar предоставя много статистически данни, които не винаги са изчерпателни, а понякога са дори и подвеждащи. Поради това са разработени алтернативи на sar – system data record и Collectl.

Обобщение на различни интерфейси и работни рамки, които представят статистики и данни от различни наблюдения, са представени в Таблица II.2 [Gregg-13].

Таблица II.2. Обобщена таблица на интерфейси и работни рамки за статистика

Вид	Особеност
Броячи за процеси	/proc
Броячи за цялата система	/proc, /sys
Драйвери на устройствата и debugинфо	/sys
Проследяване на процес	Ptrace, dtrace
Броячи на произволността на процесора	Perf_event
Проследяване на мрежата	Libpcap
Измерване на латентността на нишките	Delay account
Проследяване на цялата система	Tracepoints, kprobes, ftrace

Основните източници на статистически данни за производителността на системата се съхраняват в директориите /proc, /sys.

/proc е интерфейс на файлова система за статистика на ядрото. Той съдържа много директории, които са кръстени на идентификатора на процеса (PID), за когото съхранява информация. В тези директории са записани определен брой файлове с информация и статистика за всеки процес, свързана с данните от ядрото.

/proc се създава динамично от ядрото и не се записва на хард диска. Тази информация се използва само за четене и предоставя статистика от инструменти за наблюдение. Има и изключения на файлове, които се записват с цел контролиране на процеса и на поведението на ядрото. Списъкът на файлове се определя от версията на ядрото и config опциите. Файловете на представящи статистиката на Linux процеси в /proc може да се разделят на файлове за определен процес и файлове за цялата система.

/sys – Linux предоставя sysfs файлова система монтирана в /sys. Тя е въведена е във версия 2.6 на ядрото и дава статистически данни за него във вид на директории. Различава се от /proc по това, че има подобрения, като например добавяне на нови статистики в различни поддиректории. Sysfs първоначално е проектирана за предоставяне статистика за драйвърите на устройствата, но в последствие е разширена за всякакъв вид статистика.

II.7. Програми за генериране на входни въздействия

За изследването на компютърните системи и събирането на данни за ефективността на тяхната работа се използват програми за генериране на входни въздействия. Чрез тях се планират задачите за изпълнение от приложението, прилагат се различни сценарии и конфигурации с цел събиране на показатели за неговата производителност и надеждност.

Програмите, които се използват от софтуерни дизайнери, администратори и обикновени потребители, изискват познания за системата, която се изследва. Те предоставят функционалности, необходими за конфигуриране и генериране на различни входни въздействия, а също така показват ясно чрез задълбочено наблюдение какво е поведението на услугите като използват няколко показателя за сравняване и оценка на тяхното качество.

Тези показатели се явяват като компоненти на SLA (Service Level Agreement), които са договори между клиенти и доставчици, където QoS (Quality of service) са конкретни стойности за наличност, капацитет за бързодействие, латентност и други показатели. Корпорациите SPEC (Standard Performance Evaluation Corporation) и TPPC (Transaction Processing Performance Council) са се фокусирали върху създаване на спецификации и проектиране на стандарти за генериране на входни въздействия. Въз основа на това са разработени инструменти във връзка с няколко цели: за наблюдаване на системата и посочване на недостатъците при емулиране на стотици клиенти, за бързо и ефективно реализиране на входни въздействия, за разпределение на заявките.

II.7.1. Видове модели, използвани при генерирането на входни въздействия

Генерирането на входни въздействия, според терминологията, която се използва от специалистите, изследващи производителността на компютърни системи, се осъществява в рамките на тестови модели. Те са няколко типа [TesPL-13]:

Stress Test – определя ограниченията на системата чрез използване на входни въздействия с постоянна големина на заявките или чрез стъпаловидно увеличаване на заявките, докато сървърът се срине (“crashed”).

Real load – оценяване на натоварването на системата чрез експерименти с цел изследване на показателите за производителността при нормален режим на работа на системата, за да се гарантира, че критериите за качество QoS са изпълнени.

Други методи, които се използват за генериране на входни въздействия, са: стъпаловидно увеличаващо се входно въздействие; използване на математически разпределения; или изменения в зависимост от данните от обратната връзка за производителността на сървъра.

II.7.2. Анализ на програмите за генериране на входни въздействия

Изследвания относно производителността на програмни системи като уеб сървъра Apache се реализират чрез генериране на входни въздействия към статични ресурси като JavaScript и HTML, и към динамични ресурси като jsp, ajax, php.

Httperf е инструмент, работещ с Unix базирани операционни системи за измерване на производителността на уеб сървъри [HePDC-09]. Въвеждането на входните въздействия се осъществява чрез команда в терминала. Тяхното генериране се осъществява във вид на заявки и може да стане по няколко начина: http заявки с определена големина на единичното стъпаловидно въздействие; сесии, състоящи се от поредица http заявки във вид на единични импулсни въздейсттвия; и сесии с определена големина на единичното стъпаловидно въздействие. Отчетите на httperf за производителността съдържат информация относно свързванията, заявките и отговорите на изследваната система. Представят се статистически данни за използването на процесора и мрежовата карта.

Предимство на httperf е бързата настройка на експериментите и ниското натоварване на системата, богатият избор на изходни показатели. Недостатъци на програмата са ограниченият брой едновременни свързвания (до 1020) и платформената зависимост.

Apachebench е удобен скрипт за натоварващ (stress) тест на уеб сървър чрез увеличаване броя на заявките. Той е подобрение на инструмента httperf чрез нови параметри. Резултатите от изследването се записват в текстови файл и могат да бъдат визуализирани с допълнителен софтуер.

WebSTONE е разпределена многопроцесорна програма за генериране на входни въздействия. Той позволява да се създадат stress tests (натоварващи тестове) чрез входни въздействия с постоянна големи на заявките. Генерираните входни въздействия се прилагат чрез заявки към файлове и страници. SPECweb работи по същия начин. И двете програми се основават на LADDIS индустриални стандарти за изследване на файлови системи. За разлика от WeBSTONE, в SPECweb натоварването се генерира от четири класа, а големината на всеки клас се определя от анализ на log файловете.

Друг начин за генериране на заявки се осъществява чрез **Hbench**, където големината на входните въздействия се получава чрез анализиране на съществуващите log файлове на сървъра, за да се определят страниците и профилите на потребителите и времената между потребителските заявки. На свой ред инструментът SURGE имитира потоци от http заявки от съществуващи заявки към уеб сървъри. На базата на генераторите, описани по-горе, INRIA (Institute for Research in Computer Science and Automation) създава инструмент за генериране на заявки WAGON, който е изграден от генератор за уеб трафик, система, която изпраща и анализира заявките, и инструмент за наблюдение. Той може да генерира входни въздействия за различни видове заявки от различни машини с различни закъснения и големина. В сравнение с предните инструменти може да генерира трафик по-близък до реалния, а трафик моделът може по-лесно да се параметризира.

Съществуват и инструменти от следващо поколение за генериране на трафик. Пример за такъв инструмент е **HP's LoadRunner**, който използва Controller за прихващане на реалния поток от заявки за генериране на входно въздействие [Guru-15].

Инструментът извършва генериране на входни въздействия, чрез които анализира поведението и производителността на системата. Инфраструктурата на програмата позволява използването на функционалности като емулиране на хиляди потребители и наблюдаване на производителността с оглед идентифициране и изолиране на проблеми. Използва се за изследване на различни системи като уеб сървъри, сървъри за поточна медия, база данни сървъри и др. Допълнителен модул към програмата е AstraLoadTest, който се използва за проверка на мащабируемостта на уеб сървърите. AstraLoadTest разполага с модули за създаване и изпълнение на входни въздействия с различен вид и големина, и за визуализиране на резултатите.

Предимства на програмата са възможността за създаване на виртуални потребители от прихванат трафик, наблюдение на всички машини, както и съвместимостта ѝ с няколко

операционни системи. Недостатъци на програмата са: входните въздействия трябва да бъдат предварително възпроизвеждани; липса на модули за разширяване на функционалностите ѝ; експерименти, осъществявани на разпределени машини, могат да се реализират само с платена версия на софтуера. Понеже в настоящия дисертационен труд се насърчава използването на свободен софтуер за генерирането на входни въздействия към системата, тази програма не е подходяща за използване.

TPCW java implementation

TPCW е инструмент за изпращане на тестови заявки, основаващи се на работни натоварвания на системи, предоставящи електронни услуги. Той използва емулиращи браузъри за изпращане на множество заявки за една и съща сесия, заявки към различни URL за търсене или покупки и възпроизвеждане на натоварвания на реални потребители. TPCW използва 14 servels, които позволяват разнообразни взаимоотношения със сървъра. Действията, които са включени в програмата са търсене, покупки, наблюдаване на продуктите в потребителската кошница и логване. Всички данни за изпълнението на заявките се съхраняват и възпроизвеждат от база данни. Последователността на изпращане на заявките е произволна, но със свойства на Маркови вериги – посещението на следващата страница зависи от текущата.

Предимства на програмата са, че използва спецификации установени от TPC и разполага с множество възможности за взаимодействие с услуги за електронен бизнес чрез матрици на потока и Маркови вериги. Също така предоставя възможност за лесно реализиране на разпределени тестове. Недостатъци на програмата са трудното ѝ настройване и лошо написания програмен код. Затова тя няма да се използва в за целите на настоящето дисертационно изследване.

Apache Jmeter

Apache Jmeter е „отворен код” софтуер, 100% Java приложение, разработено от Apache Jakarta и предназначено за генериране на входни въздействия за натоварване на система. Чрез него се изследва функционалното поведение и производителността. Той предоставя различни графични анализи за последната. Първоначалното му предназначение е за изследване на уеб приложения, но в последствие е разширен и с други функции за анализиране. Последната му версия предоставя възможност за проучване на производителността и функционалните възможности на различни видове сървъри, както и на Java приложения, протоколи и статични ресурси.

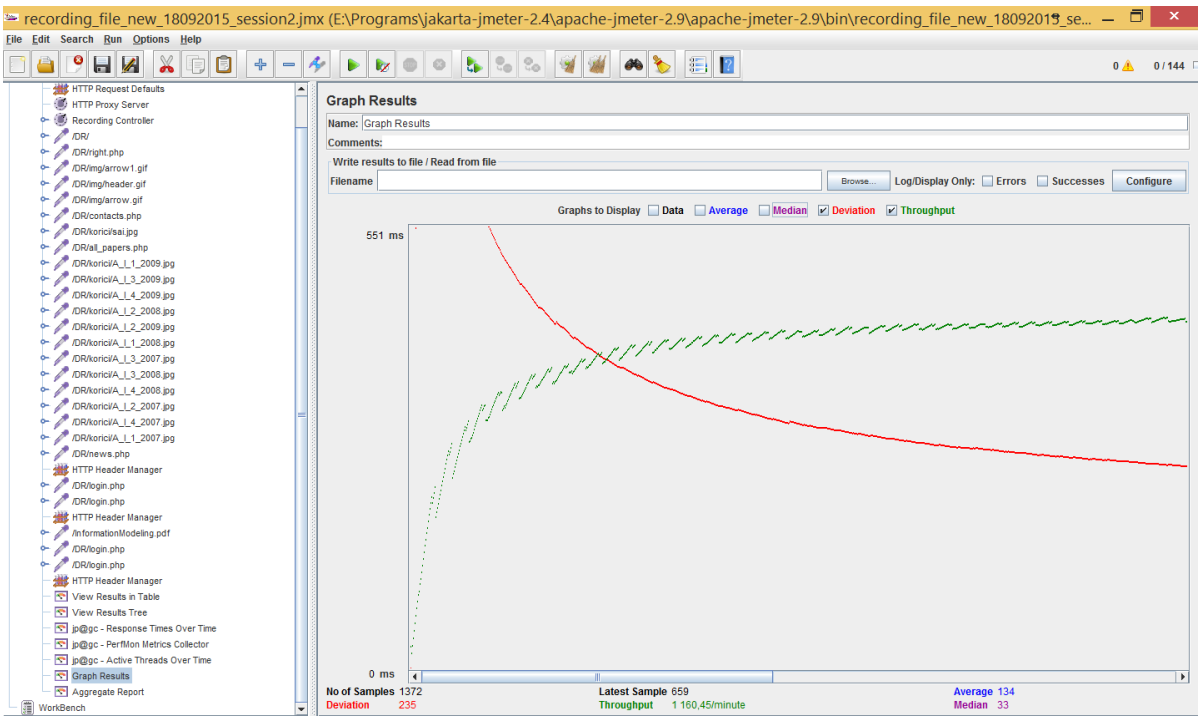
Предимства на Jmeter са, че позволява осъществяването на разпределени експерименти (реализирани на различни машини), както и че се характеризира с платформена независимост. Той предоставя възможност за използване на допълнителни компоненти в случаи на провеждане на множество изследвания едновременно. Трябва обаче да се има предвид, че използването на допълнителни компоненти е сложна задача, която изисква време и често води до неубедителни резултати.

Показателите на тестовете на JMETER са следните:

- *Sample jmeter* е пореден номер на опита, т.е. на заявката;
- *Sampletime* е еквивалентно на load time, elapsed time и response time. Това е интервалът от време, в който заявката е била изпратена и е получен отговор от клиента;
- *Latency* е интервал от време, в което заявката е била получена от сървъра и генерираният отговор е започнал да се изпраща;
- *Throughput* е големина на бързодействието. Измерва заявките, изпълнени за секунда/час/минута, които сървърът може да управлява.

Друг важен параметър е *Deviation (отклонение).* Ако бързодействието е високо и отклонението е ниско, това би означавало, че производителността на сървъра е добра.

Приложимостта на гореспоменатите показатели е демонстрирана и в рамките на настоящото дисертационно изследване чрез експерименти, в които се генерират входни въздействия във вид на http заявки към уеб сървъра Apache. На фиг. II.5 са показани резултати от извършените тестове с програмата Jmeter по отношение на изследваната програмна система. На нея се вижда, че сървърът, който се използва за експерименталните изследвания може да управлява 1160,45 заявки за минута. Резултатите, представени на фигурата, са получени при извършване на експеримент, при който в интервал от десет секунди софтуерното приложение е посетено от двама потребители. Става ясно, че компютърната система може да обслужи потребителски заявки за по-малко време от това, което е необходимо за тяхното пристигане. От гледна точка на качеството на процеса, графиката на изходните данни от системата показва, че процесът е апериодичен и установен.



Фиг.II.5. Показатели на производителността Throughput и Deviation

Програмите за генериране на входни въздействия се използват, за да се решат определени задачи при изследването на потребителски приложиния. Първоначалната задача е да се събере информация за ограниченията на приложението, за да може да се осъществи качествен експеримент. Предназначението на изследването на производителността е да подмогне: идентифициране на ограниченията на компютърната система; настройките на програмната система; и оценяването на статистиките за бъдещи експерименти. Резултатите от изследването за производителността могат да допринесат за оценяване на хардуерната конфигурация, с която работи приложението [MFBBR-07]. Данните, получени при изследване производителността на компютърната система, са приложими и за изграждане на модел на действията, извършвани от нея.

Изследването на производителността се състои от следните етапи:

1. **Определяне на средата за провеждане на експеримент,** т.е. идентифициране на физическата среда, приложенията и ресурсите, налични за проучване. Физическата среда включва хардуер, софтуер и мрежово конфигуриране. По-доброто идентифициране на системата подпомага извършването на по-добро проектиране и планиране на изследването. В някои случаи тази дейност трябва да се преразглежда периодично през целия жизнен цикъл на проекта.

2. **Определяне на приемливи критерии на производителността** – определяне на времето за отговор, throughput (бързодействие), използване на ресурсите и ограниченията. Като цяло времето за отговор представлява интерес за потребителя, бързодействието предизвиква интерес от страна на бизнеса, а използването на ресурсите е интересно за системата. Освен това съществуват критерии за успешна работа на системата, които не могат да бъдат уловени от тези цели и ограничения. Например изследването на производителност може да се използва за оценка на комбинацията от конфигурационни настройки за най-желаните характеристики на производителността.

3. **Проектиране и планиране на изследването** – изработват се ключови сценарии; идентифицират се различията между потребителите и се определя начина, по който да бъдат симулирани; определят се входните данни за реализирането на изследването; и се установяват показателите, които трябва да бъдат наблюдавани. След това тази информация се обединява в един или повече модели за използването на системата, които съответно се изпълняват и анализират.

4. **Конфигуриране на средата за проучване –** подготвя се средата за проучване, както и инструментите и ресурсите, необходими за всички стратегии като характеристики (features) и достъпни компоненти.

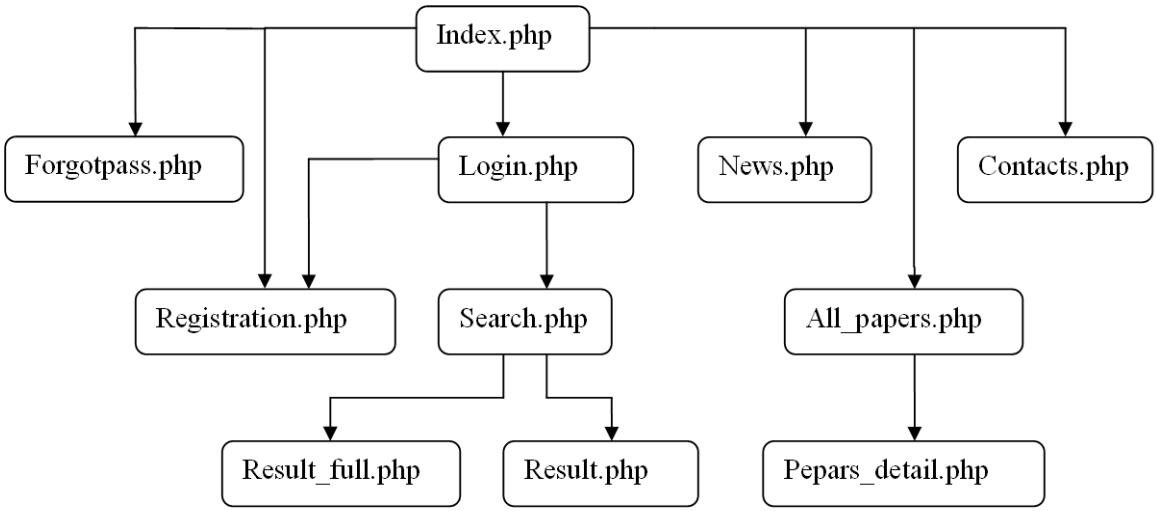
5. **Имплементиране на теста –** разработване на експеримент за производителността, според проектирането на изследването.

6. **Изпълнение на експеримента –** включва следните действия: изпълнение, наблюдение, валидиране на данните, събиране на резултатите. Изпълнение на експеримента позволява оценяване на различни опити с цел анализ, когато се наблюдават данните и използваната среда за провеждане на експеримента.

7. **Анализиране на резултатите, отчети и извършване на експериментите отново.** Обединяване и споделяне на резултатите от експериментите. Анализиране на данните индивидуално или чрез зависимости между функционалностите. Приоритизиране на определени експерименти и изпълнението им отново. Когато данните за всички показатели са в приемливи граници и нито едно от ограниченията не е нарушено, следва обединяване на данните, прекратяване на експериментите и приемане настройките на получената конфигурация.

II.7.3. Описание на използвания сценарий на входните въздействия към изследваното софтуерно приложение

В настоящия параграф се представя описание на сценария, по който се изследва софтуерното приложение, използвано за експерименталните цели на дисертацията. Неговото предназначение е да осигури на клиентите си достъп до желаните от тях услуги. На фиг.II.6 е показана функционалната схема на използваното софтуерно приложение.



Фиг.II.6 Функционална схема на софтуерното приложение

Клиентите зареждат сайта hs22.iccs.bas.bg и се насочват към страница index.php, след което потребители на софтуерното приложение разглеждат последователно страниците за новини (news.php), контакти (contats.php) и всички публикувани списания, които се разпространяват по електронен начин чрез посещаване на страницата за списанията (all_papers.php). Извършеният експеримент е с клиент, който вече е регистриран в системата, така че директно се използва страницата login.php, където се въвеждат потребителско име и парола. Проверката установява, че името и паролата съществуват в базата данни и това позволява на потребителя да извърши търсене в нея на целия текст на статията. Той търси две статии по два критерия. Първият е „име на автор“, а вторият е „година на публикуване“. След това потребителят прочита статиите в pdf.

Всички тези действия, които се извършват от софтуерното приложение, се използват, за да се симулират заявки от клиентите на системата в определен момент. По този начин се изследват възможностите на система за обслужване на потребителите на уеб базираното приложение за електронно разпространение на научно списание.

Осъществяване на проучване с различен брой потребители цели да се изгради модел на параметрите, измерващи производителността на системата.

II.7.4. Функционално моделиране на компютърната система

За да се създаде функционален модел за използваната компютърна система е необходимо да се извършат изследвания, при които се натоварва уеб сървъра, като се симулират множество потребители, които потребяват неговите услуги едновременно. По този начин се моделират функциите, изпълнявани от системата като действия, операции или процеси и се определят реакциите на системата като бързодействие, устойчивост, надеждност и мащабируемост. Моделът гарантира, че софтуерното приложение ще работи успешно при пускането му в експлоатация за нуждите на реални клиенти. Едновременното използване на даден сайт от потребители е необходимо, за да се определят работните натоварвания в реализираните експерименти. В действителност има много фактори, които влияят на натоварването на сървърите. Те въздействат и на броя едновременни потребители. В много изследвания последните се използват като входни данни.

В реалните компютърни системи равномерното разпределение на броя едновременни потребители е рядко явление, което налага изследвания за наблюдаването, разбирането и моделирането им в изследваната система.

Jmeter позволява да се използват линейно и равномерно разпределение на потребители при симулации и да се измерва производителността след тяхното стартиране. Този начин на стартиране на потребителите в изследваната система представя реалното натоварване с малка вероятност.

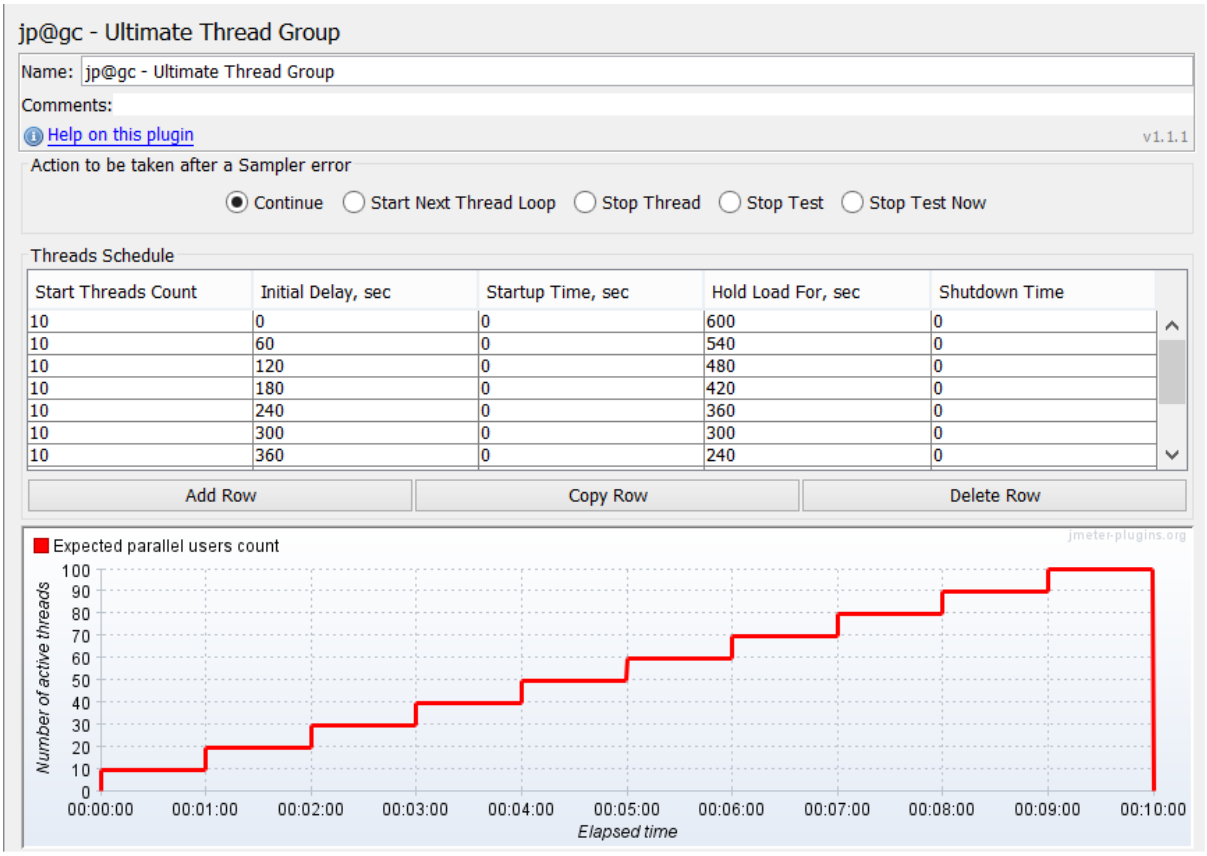
За реализиране на функционален модел се използва модулът Ultimate Thread Group на Jmeter, който позволява да се планират безкраен брой записи. Той осигурява повече възможности като стартиране на различен брой потребители едновременно, времеви интервал за стартирането им (ramp-up) и продължителност на потреблението. По този начин се създават по-реалистични експерименти за използваните работни натоварвания.

Този подход за изчисляване на броя потребители, използващи определен уеб сайт едновременно, може да доведе до някои проблеми при анализиране на данните. Във връзка с това е нужно да се приемат следните допускания:

- Посетителите са равномерно разпределени в наблюдавания период; функцията на разпределение има форма на Поасонов процес.

- Броят потребители може да бъде погрешно оценен, особено ако се използва IP адрес за определяне на даден потребител, понеже един IP адрес може да се използва от повече потребители.

Модулът позволява да се приложат условия при натоварването за изследване на сървъра. В експеримента се симулира използване на сървъра от 110 потребителя, които започват да изпращат заявки в различни времеви интервали. На фиг.II.7 е показано, че в първите 60 секунди услугите, предлагани от него, се използват от 10 потребителя, в следващите 60 секунди се добавят нови 10 и така с всяка минута се добавят по 10 потребителя.



Фиг.II.7. Настройване на потребителите чрез Ultimate Thread Group

Модулът разполага с няколко атрибута, които се използват в изследването:

Start thread count организира потребителите да изпращат заявки към сървъра един по един. В първия ред на фиг.II.7 са въведени 10 потребители, което означава, че сървърът ще бъде натоварен с първите 10 потребители.

Initial Delay на свой ред настройва закъснението, след което първият потребител ще започне да изпраща заявки след стартиране на програмата Jmeter.

Start-up time е времето за започване изпращането на заявки, което се разделя между всички потребители.

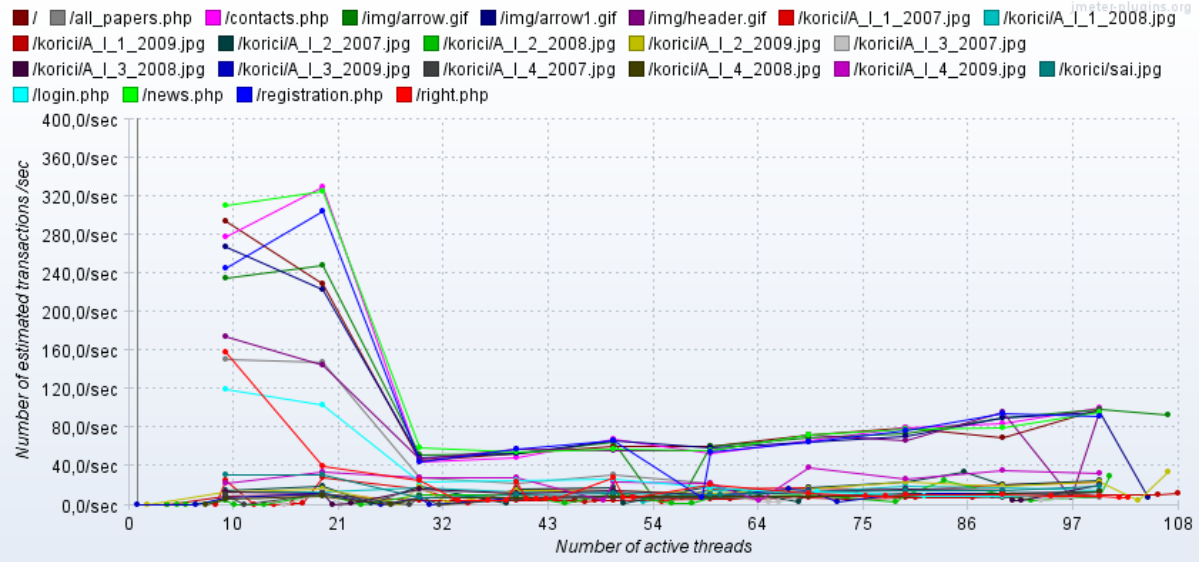
Hold load for определя времевия интервал, през който потребителите ще използват ресурсите на сървъра.

Shut down time определя времевия интервал, през който потребителите ще спрат да натоварват сървъра.

В рамките на настоящото дисертационно изследване се измерват основни показатели за производителността на системата, за да се моделират нейните възможности. Това става с помощта на listener елементите на програмата Jmeter, които съхраняват резултатите от http заявките във файлове и представят визуален модел на данните [ApaSF-16].

Първият показател, който се използва за моделиране на компютърната система, е Throughput. Той измерва броя на извършение операциите по време на изследването и отразява капацитета на един сайт или едно софтуерно приложение. Целта на неговото използване е да се определи броят на заявки за час/минута/секунда, с които може да се справи приложението. Показателят throughput се изследва с елемента за получаване на данни Transaction Throughput vs Thread.

Фигура II.8 показва статистически данни за максималния брой възможни транзакции (обслужени заявки от сървъра) на базата на броя на потребителите, имащи достъп до приложението. На нея се демонстрира изменението на транзакциите, които се изпълняват от сървъра в завимост от броя потребители. Линиите на фигурата представят всяка страница на информационна система, която се изследва. Раздалечеността между линиите се получава, защото една част от страниците съдържат статични файлове снимки, а друга част са файлове, които се генерират динамично чрез езика php. Фигурата демонстрира ограничение на софтуерното приложение – малък брой обслужени транзакции за статични файлове, както и необходимостта от промяна големината на снимките.

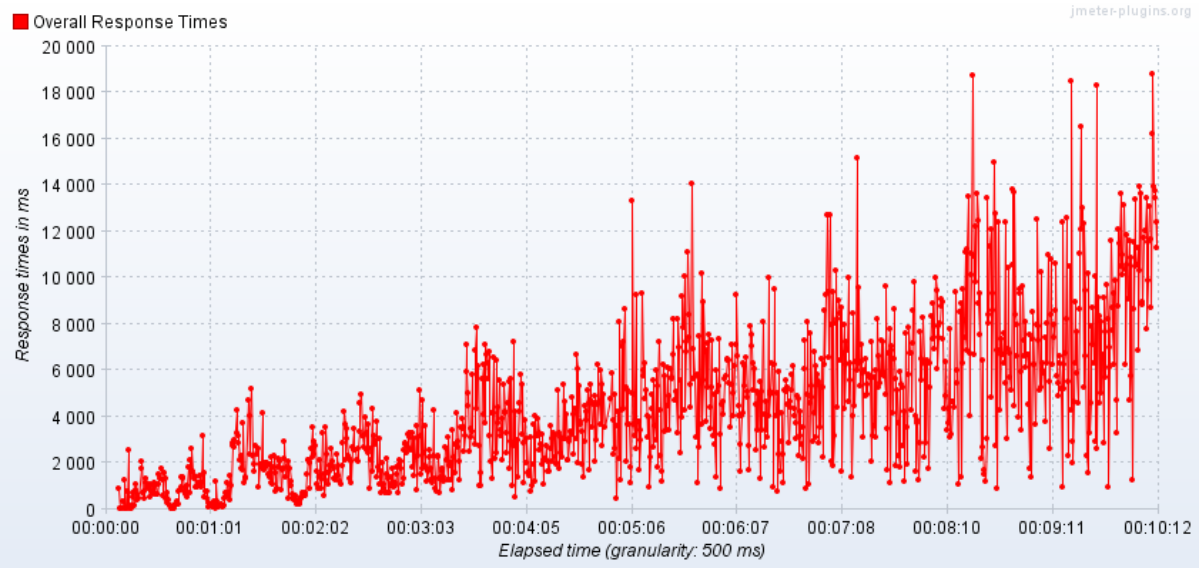


Фиг.II.8. Зависимост между броя транзакции и броя потребители

Има и други показатели като време за отговор, латентност и др., които също трябва да бъдат взети под внимание при изследване производителността на приложенията. Следващият показател, който се моделира чрез програмата Jmeter, е времето за отговор. Данните се получават чрез елемента Response times over time, който показва средното време за обработка на заявките от сървъра в милисекунди.

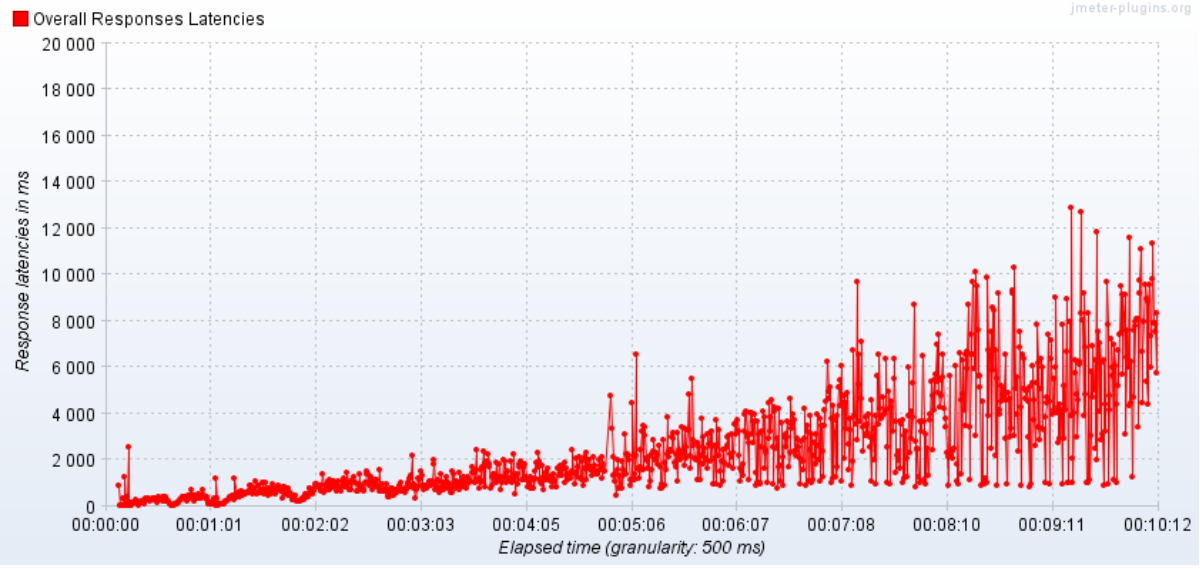
Фиг II.9 показва измерването на времето за отговор като функция на времето. Чрез нея се определя изменението във времето за отговор при промяна броя на потребителите. От фигурата може да се направи заключение, че времето за отговор се изменя в зависимост от изпратените заявки от потребителите. Това ни дава информация за съществуващите ограничения в работата на системата.

Третият показател, който се моделира чрез Jmeter, е латентността в отгорвора на потребителските заявки. Данните за него се получават от елемента на Latencies over time.



Фиг.II.9 Промяна във времето за отговор на системата в зависимост от броя потребители

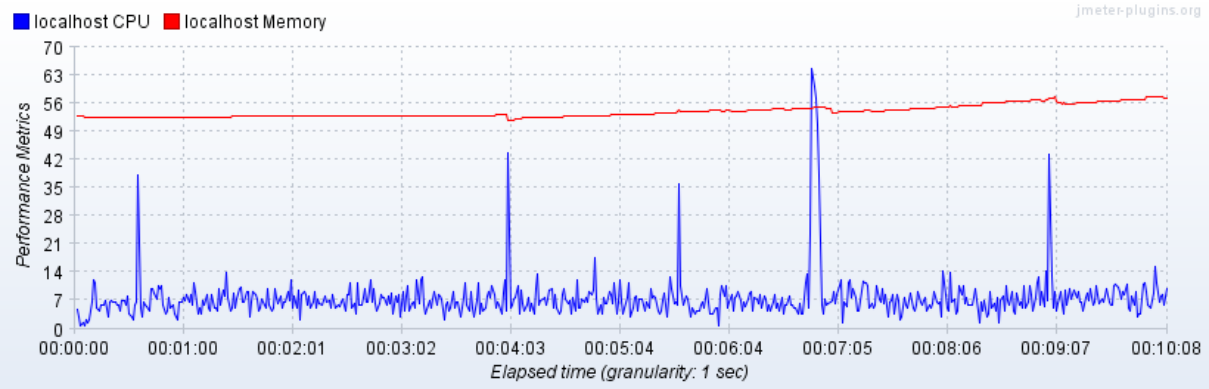
Фиг.II.10 показва латентността в отговор, изпратен от сървъра по време на извършване на изследване на последния чрез натоварване. Латентност изразява времето между получаването на заявката от сървъра и започването на изпращане на отговор от негова страна. С представените на фиг.II.10 данни се анализира времето, след което ще започнат да се обработват заявките в зависимост от броя потребители. Тя показва, че латентността се увеличава при увеличаване на потребителските заявки.



Фиг.II.10 Промяна в латентност на системата в зависимост от броя потребители

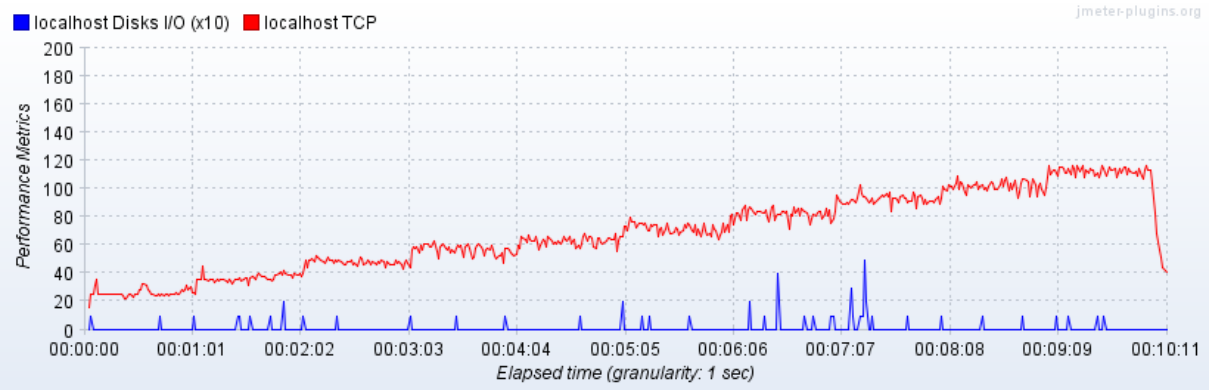
За да се обяснят промените в латентността и времето за отговор се измерват и хардуерните компоненти на системата – процесор, памет, хард диск и др. За тази цел се използва елемента PerfMonCollector, който позволява да се измерят приложените входни въздействия към сървъра при извършване на изследвания за производителността на системата. Целта на неговото използване е да се съберат статистически данни за процесора, паметта, swap и др. Той е много подходящ инструмент, който се поддържа от различни операционни системи.

На фиг.II.11 е представено изменението в натоварването на паметта и на процесора като функция на времето. От нея може да се заключи, че паметта се променя в зависимост от потребителите на софтуерното приложение, а процесорът се натоварва константно през цялото време на изследването. Резултатите, представени на фиг.II.11, не показват пренатоварване на хардуерните ресурси. Това налага изследване на други показатели на компютърната система.



Фиг.II.11. Натоварване на процесора и паметта на системата

На фиг.II.12 е показано натоварването на TCP комуникационния протокол и твърдия диск на системата. Тяхните измервания се извършват с цел да се проследи източника на увеличението на времето за отговор и латентността.



Фиг. II.12 Натоварване на TCP комуникационния протокол и твърдия диск на системата

От представените резултати на фиг.II.12 може да се направи заключение, че TCP комуникационният протокол е в пренатоварено състояние в последния времеви интервал на изследването на системата, което обяснява и увеличаването на времето за отговор и латентността на системата. Освен това фигурата показва, че неговото натоварване е пропорционално на броя потребители на системата. Синята линия на фигурата демонстрира, че натоварването на твърдия диск на системата е малко, което означава, че увеличаването на времето за отговор не е породено от него. Пиковете в нейното поведение се получават, когато процесите не могат да се обслужат от паметта и се предават към хард диска [GheGF-13].

II.8. Изводи

Във втора глава на дисертационния труд бяха представени начини за анализ на компютърни системи, които да подпомогнат изграждането на тяхните модели. Бяха отбелязани някои фактори, които влияят на производителността на сложни компютърни системи. Представена бе и структурата на използваната в хода експерименталната част на настоящето дисертационното изследване система. Текстът също така предложи описание на принципа на действие на създадената програма за управление на изследваната програмна система. Разгледани бяха и методологиите за измерване производителността на компютърните системи и възможностите за наблюдение на софтуерните и хардуерни компоненти на последните. В главата се прави анализ на програмите за генериране на входни въздействия и се изгражда сценарий за тяхното прилагане към обекта на изследване в настоящия дисертационен труд – уеб сървъра Apache.

Втора глава на този текст демонстрира, че опциите на програмата Jmeter позволяват да се изгради модел на функционалните възможности на компютърната система чрез изследване разработеното приложение и хардуера, който осигурява неговото поддържане. По този начин става възможно да се анализира бързо и лесно голямо количество данни от измервания на генерираното натоварване към уеб сървъра и да се изгради математически модел за промените в показателите за производителността на компютърната система. Функционалният модел на системата дава възможност да се изследва изменението в показателите за производителност по отношение на нейните устойчивост и надеждност.

III. РЕАЛИЗИРАНЕ НА УПРАВЛЕНИЕ В ИЗСЛЕДВАНАТА КОМПЮТЪРНА СИСТЕМА

Провеждането на експериментите в дисертационния труд има две основни цели. Първата е свързана със създаденото софтуерно приложение за достъп до статиите на научно списание. Тя е да се обоснове и анализира предложения от автора по-модерен, по-лесен и по-функционален начин за достъп и търсене в списанията, разпространявани от Съюза по автоматика и информатика.

Втората цел е да докаже, че промяната на параметрите на системата, в това число параметрите на конфигурационния файл на Apache и на комуникационния протокол TCP, която поддържа софтуерното приложение за електронно разпространение на списанията, ще подпомогне едновременния достъп на повече читатели до системата. Постигнатите подобрения в компютърната система могат да се проследят чрез показателите за качество на производителността на програмната система: време за отговор и бързодействие.

III.1. Характеристики на компютърната система

Обект за управление в настоящите изследвания е компютърна система със следните технически и програмни характеристики:

- ✓ Процесор Intel Core 2 T5300 1,73 GHz
- ✓ Памет 2 GB
- ✓ Твърд диск ATA FUJITSU MHW2160B 160 GB
- ✓ Операционна система Linux Centos 6.5
- ✓ Файлова система EXT 4
- ✓ Мрежова карта RTL8101E/RTL8102E 100Mbit/s
- ✓ Програмен език PHP 5.5.24
- ✓ Уеб сървър Apache prefork 2.4.12
- ✓ База данни MySQL 5.6.24
- ✓ Генератор на натоварване ApacheJmeter 2.9

Компютърната конфигурация използва многопроцесорна архитектура (SMP), която е изградена от симетрична многопроцесорна хардуерна и софтуерна система с два еднакви процесора, свързани към една споделена основна памет. Те имат пълен достъп до всички входно/изходни устройства и се управляват от една операционна система. Така всички процесори се натоварват еднакво. SMP архитектурите се използват в сървърни системи за обслужване на приложения, които имат множество процеси, изпълнявани паралелно, като

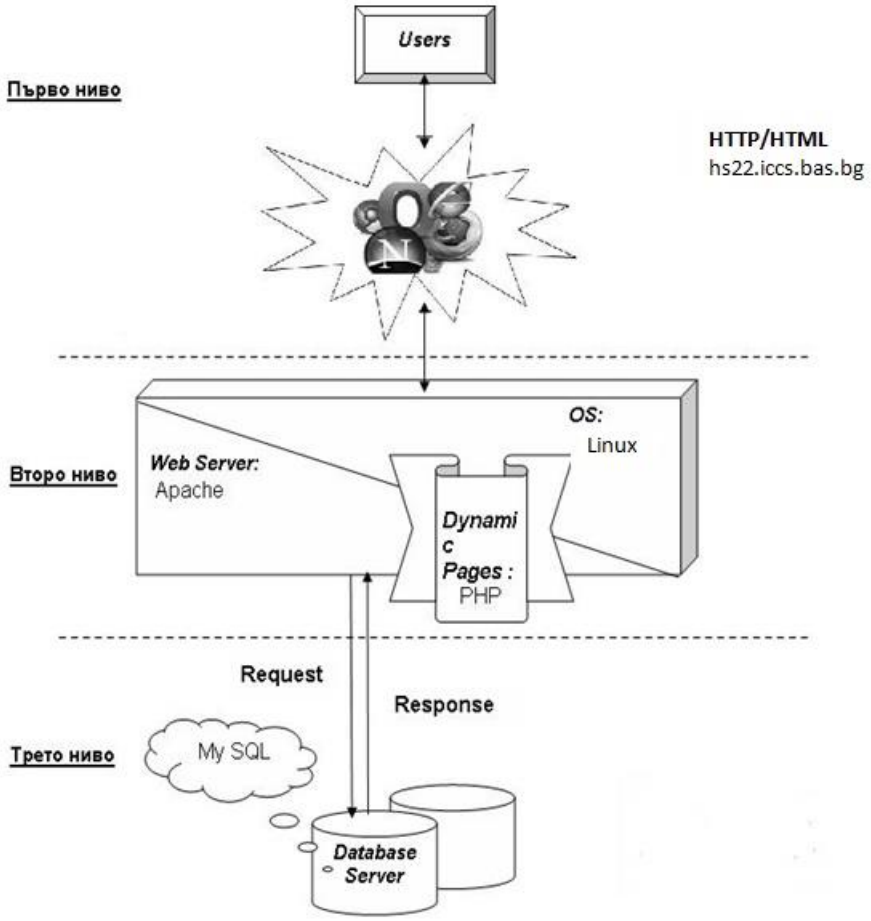
така се обслужват различни процеси на различни процесори. Уеб сървърът Apache е от този вид приложения. Многопроцесорната архитектура SMP позволява да се реализира динамично балансиране на натоварването в компютрите системи. По този начин се обслужват по-бързо повече потребители.

Всички използвани програми са от тип свободен софтуер (open source) и са инсталирани на сървър с посочените по-горе хардуерни характеристики. Той е разположен в локалната мрежа на секция „Йерархични системи" при ИИКТ-БАН, а чрез използването на софтуерните продукти е разработено уеб базирано приложение за разпространение на научно списание.

Софтуерно приложение за достъп до статиите на научно списание

За целите на експерименталната работа в настоящото дисертационно изследване е използвано интернет-базирано софтуерно приложение, което позволява селективно търсене, обработване и извличане на информация за статиите, публикувани в българско научно списание. Чрез него се предлага информационна услуга в Интернет за електронен абонамент на научно списание, която да позволява регистриране, следене статуса на потребителя, селективно търсене, обработване и извличане на информация за статиите, публикувани в списание „Автоматика и информатика". Това софтуерно приложение използва трислоен клиент/сървър модел за динамичното генериране и предоставяне на информация [TricE-11]. То е реализирано чрез няколко софтуерни модули, разработени посредством PHP (HypertextPreprocessor). Чрез тези модули се реализира SQL заявка (request) и SQL отговор (response), необходими за функционирането му. След получаването на SQL заявката се претърсва базата данни MySql, за да бъде извлечена желаната от потребителя информация. Данните от базата се обработват и структурират в HTML формат с цел отговаряне на SQL заявката като SQL отговор (response). Това са продукти от така наречения свободен код (open source). Те са системно независими, поради което е възможно използването им при Windows, Unix и Linux базирани машини.

Фигура III.1 илюстрира архитектурата на приложението. То има три логически обособени йерархични нива [TricE-11].



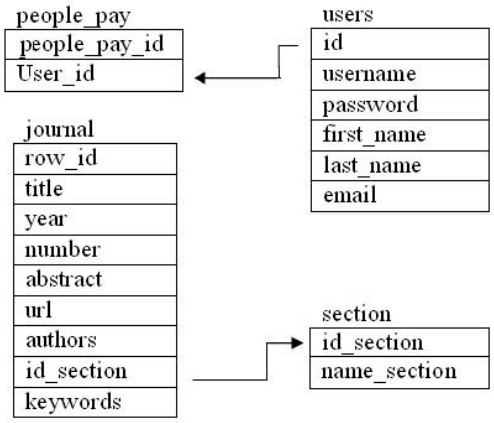
Фиг.III.1. Архитектура на приложението

Първото функционално ниво е клиентът, който (на фигурата е обозначен като Users) използва обикновен браузер (например Internet Explorer или Google Chrome, Mozilla), който се свързва със сървърска част на системата. Комуникацията между двете нива се осъществява чрез протокола http.

Второто ниво е изградено под Linux операционна система и се реализира чрез уеб сървъра Apache, който обслужва софтуерното приложение. Осъществява се динамичното генериране и предоставяне на информация. При отправено потребителско запитване сървърът стартира PHP обработваща програма, която генерира HTML страница в реално време.

Третото ниво е това на базата данни. То е реализирано чрез MySQL database. MySQL сървърът е много бърз, мощен и лесен за употреба. Той е разработен да управлява и поддържа големи бази данни много по-бързо от съществуващите за това решения. В последно време се използва много успешно в среди с високи изисквания относно производителността. В конкретната база данни има четири отделни таблици. В тях се

съдържа подробна информация съответно за статиите, регистрираните потребители, потребителите с платен абонамент и направлението, в което са публикациите. На фиг. III.2. са показани връзките между таблиците на база данните.



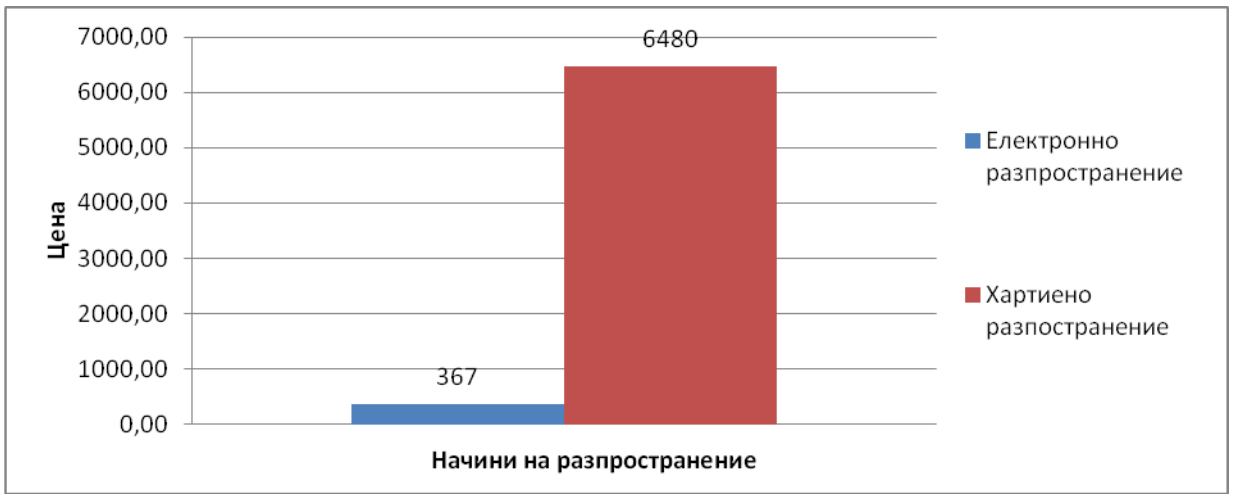
Фиг.III.2. Обща схема на връзките между таблиците

В изпълнение на първата поставена цел на настоящата глава, бе изследвана икономическата полза от използването на създаденото приложение. Сравнение на разходи за електронно и хартиено разпространение на списание „Автоматика и Информатика" е направено в таблица III.1, а графичното представяне на резултатите е показано на фиг.III.3.

Разходите за електронно разпространение на списанието са изчислени въз основа на цената на електроенергията и консумацията на електроенергия от сървъра за една година. В разходите за хартиено разпространение са включени тираж на списанието за година (4 книжки годишно х 500 броя=2000 бр.) и цена на една книжка.

Таблица III.1 Сравнение на електронно и хартиено разпространение на списание „Автоматика и Информатика"

Електронно разпространение		**Хартиено разпространение**	
Тарифа на ел. енергия	0,12702 лв./kWh	Тираж	500 бр.
Моментна консумация на ток	1,5 A	Тиражи за година	4 пъти
Консумация на електроенергия	0,33kW	Цена за печат на 1 тираж (500 бр)	1620 лв.
Консумация на електроенергия за година	2890,8 kW/h	Цена за напечатването на едно списание	3,24 лв.
Разходи за година	367 лв.	Разход за година	6480 лв.



Фиг. III.3. Сравнение на начините на разпространение на списание „Автоматика и информатика”

Резултатите показват, че разпространението на списанието по електронен път е 17 пъти по-евтино от начина, използван в момента. Поради икономическата ефективност на приложението, неговото усъвършенстване и пускане в експлоатация са в процес на договаряне. Възможностите на разработеното приложението и неговите функционалности са представени на конференция по автоматика и информатика през 2013г. [Димит-13а].

За да работи ефективно създаденото софтуерно приложение е необходимо да се изследва и производителността на уеб сървъра, който го поддържа. Поради тази причина тя е разгледана по-обстойно в следващата точка на дисертационния труд.

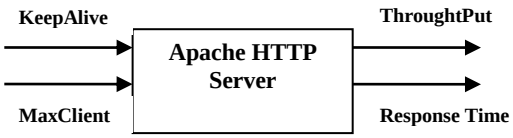
III.2. Изследване на производителността на програмната система

Изследването на производителността се извършва с цел анализиране на натоварванията, които може да обслужи уеб сървъра или един уеб сайт, установяване на критични проблеми, които възпрепятстват оптималното потребление на уеб базирано приложение и определяне на използвания хардуерния ресурс. То позволява да се измери количеството работа, извършено от компютърната система. Ефективността на една компютърна система, в зависимост от контекста, в който се използва, се определя от време за отговор, бързодействие, натоварване на хардуерните ресурси, достъпност до компютърната система или определено приложение, капацитет на използваната мрежа и др. [ArnoA-94].

По този начин се разбират възможностите на разработения софтуер и машината, която се използва за неговото поддържане.

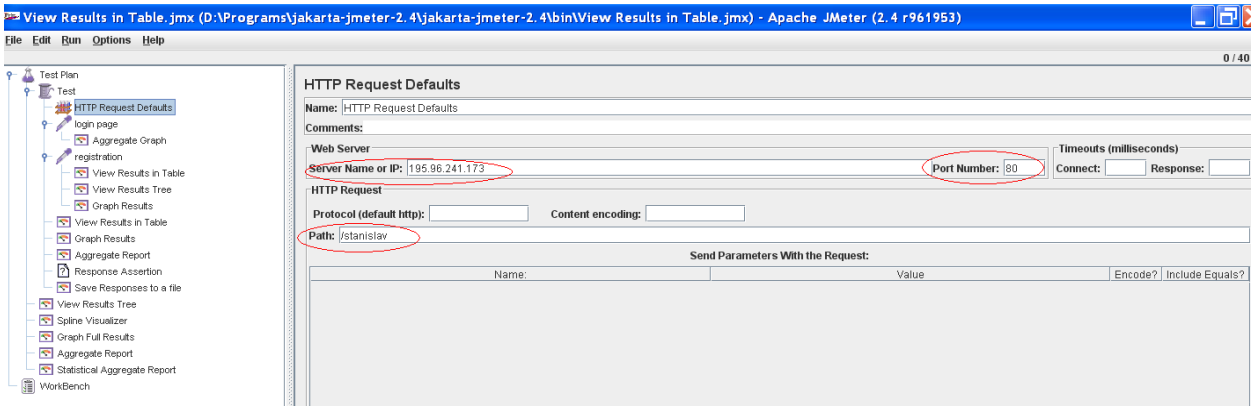
III.2.1. Изследване на програмната система като система за автоматично регулиране и генериране на входни въздействия с Jmeter

За целите на това дисертационно изследване се извършва симулиране чрез http заявки от Jmeter и се генерират входни въздействия от множество потребители едновременно към уеб сървъра Apache, който поддържа софтуерното приложение. По този начин се изследва производителността на уеб сървъра Apache при нормален и пренатоварен режим, за да се гарантира предоставянето на информация и услуги за потребителите на софтуерното приложение. Така се анализират настройките на уеб сървъра Apache, които са отговорни за обслужването на потребителски заявки. Той е представен като система за автоматично управление, за която са изследвани показателите на производителността – бързодействие и времето за отговор на програмната система при зададени максимален брой свързвания KeepAlive, които сървърът разрешава за определен брой потребители MaxClients в даден момент от време (Фиг. III.4). По този начин се изследва програмната система като стабилизираща при зададени постоянни стойности на конфигурационните параметри на Apache: KeepAlive и MaxClients.



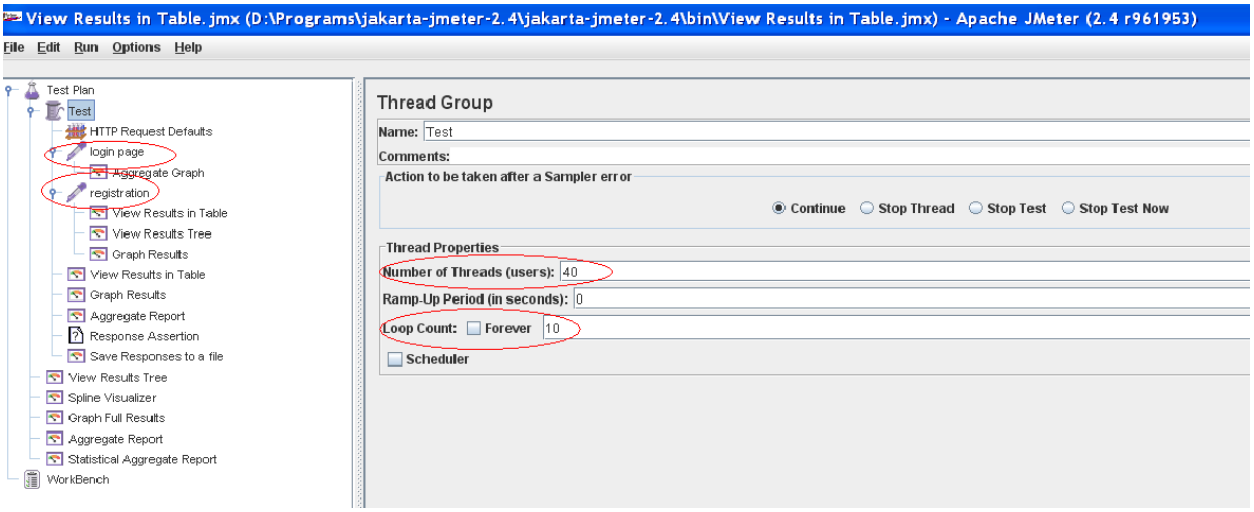
Фиг. III.4. Apache HTTP Server, представен като система за автоматично регулиране

В хода на настоящето изследване са зададени HTTP заявки (HTTPRequest) по метода Post към хоста www4.iccs.bas.bg, който се поддържа на Apache 1.3.23 Server. Комуникацията се извършва чрез http заявки през port 80, което е показано на фиг.III.5.



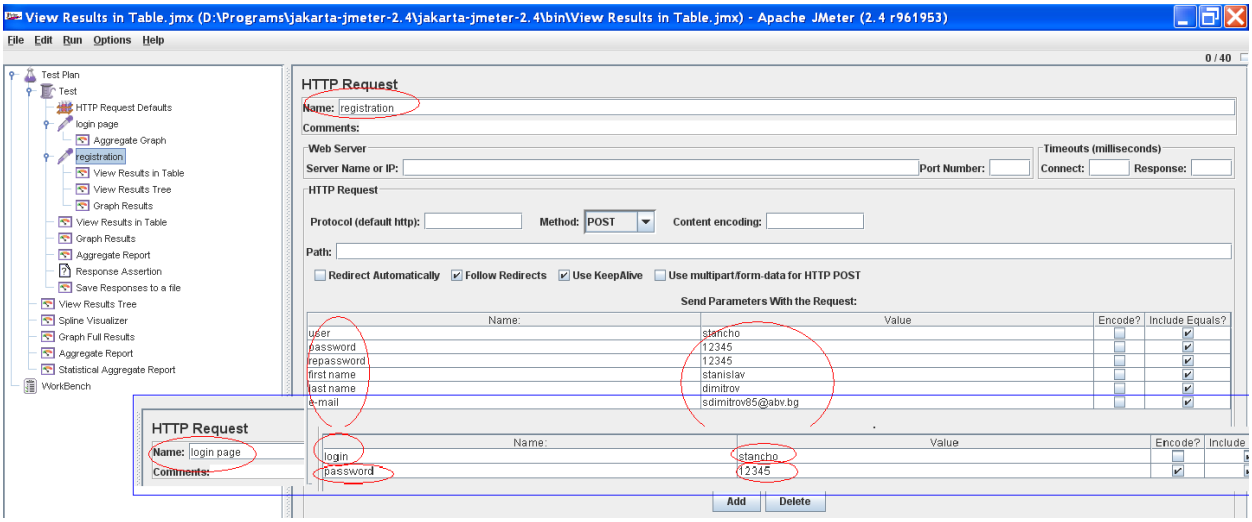
Фиг.III.5. http комуникация със сървъра

Проследява се поведението на сървъра при свързване с 40 потребителя, които изпращат по 10 заявки. Експериментът е проведен при условие, че потребителите се регистрират (registration) и влизат в софтуерното приложение (login page), както е показано на фиг.III.6.



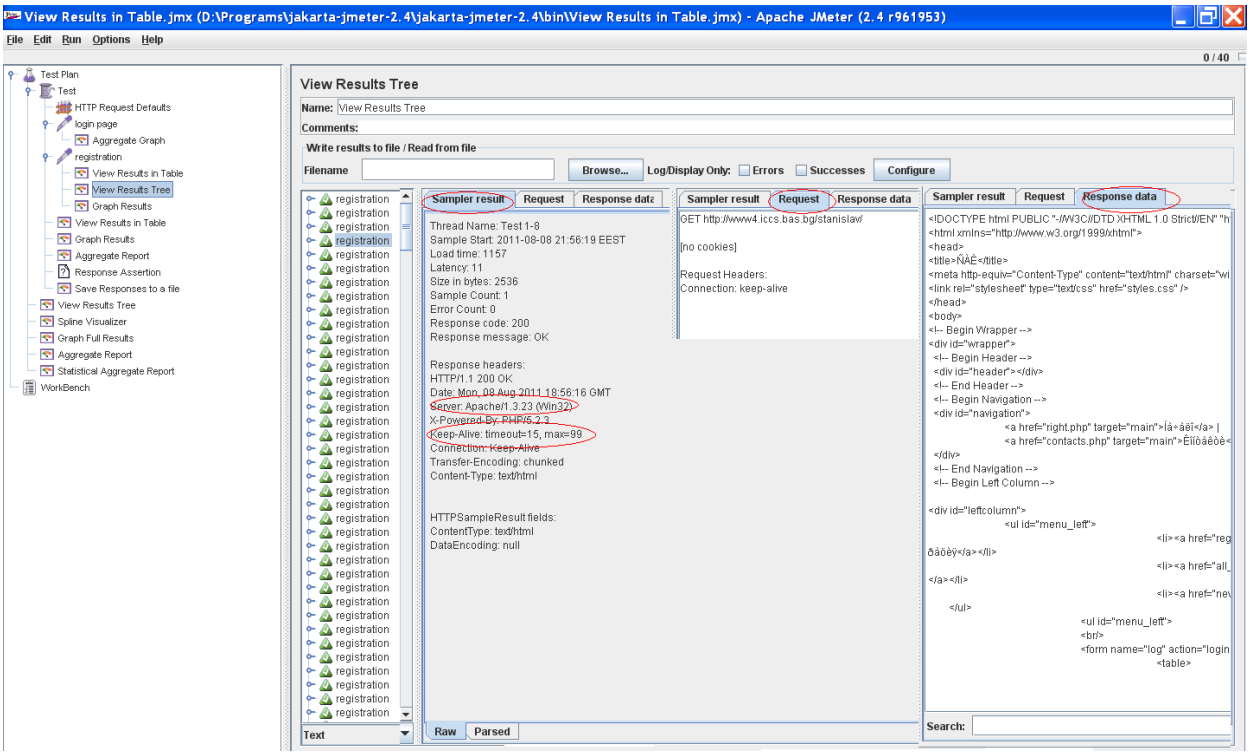
Фиг.III.6. Thread group – Конфигуриране на сценария

На фиг. III.7 са показани елементите на: login page, в която се задава заявка за потребителско име и парола; и registration, в която са зададени име, парола, потвърждение на паролата, собствено име, фамилно име и електронна поща.



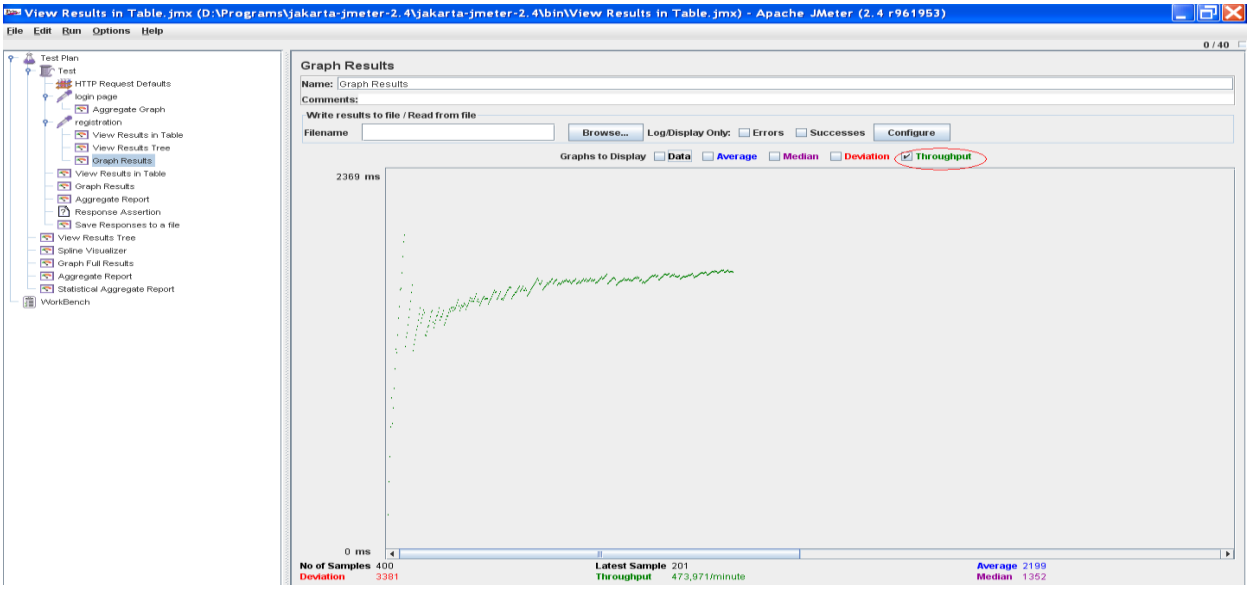
Фиг. III.7. Изпратени данни чрез заявката

Резултатите са показани на фиг.III.8. В таблицата „Simple result" се дава най-важната информация за заявката, използваната версия на уеб сървъра, максималният брой на заявките и времето за отговор на Apache HTTP Server. „Request" показва към кого е изпратена заявката, „Request data" – данните, които съдържа заявката, а „Filename" – името на файла, в който се записва информация за уеб сървъра. След описването на тези данни може да се направи съответният анализ.



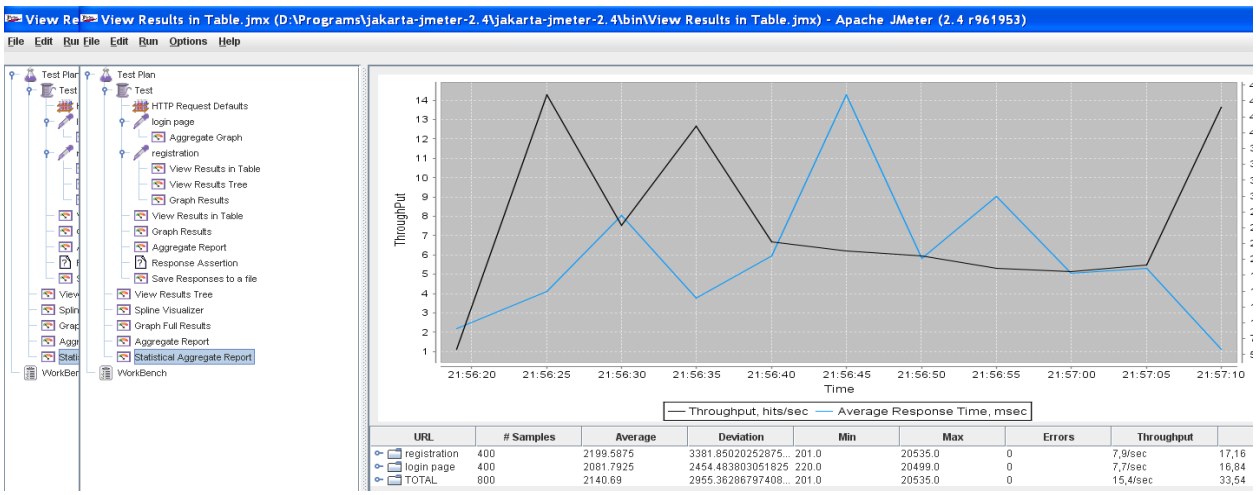
Фиг. III.8. Преглед на резултатите

Фигура III.9 показва графични резултати за производителността на системата. От преходния процес на показателя на производителността Throughput се вижда, че системата е устойчива.



Фиг.III.9 Графични резултати

Фиг. III.10. на свой ред ни информира за най-важните показатели на компютърната система при регистрация и влизане в софтуерното приложение. Тя предоставя информация за изходните величини: Време за отговор (Response Time) и бързодействие на сървъра (Throughput). Данните са с близки стойности, а това означава, че системата за автоматично управление е стабилна и надеждна. Колебанията в получените изходни резултати се получават от подаваните заявки към база данните MySQL. От друга страна, изходните данни показват, че при задаване на постоянни стойности за параметри KeepAlive и MaxClients на уеб сървъра Apache, той се саморегулира много добре. Резултатите от тези експерименти са описани по-подробно в [Димит-11].



Фиг.III.10. Обобщена статистика

III.2.2. Реализиране на входни въздействия към програмната система чрез видео файлове

За целите на настоящето дисертационно изследване софтуерното приложение за разпространение на научни статии е разширено чрез добавяне на модул за разпространение на видео файлове от конференции, обсъждания и семинари. Видео файловете се характеризират с голям обем и това води до по-голямо натоварване на ресурсите на хардуерната конфигурация. Затова в статия на автора [DimiT-13] се изследва подобряването на работа на програмната система чрез включване на допълнителен модул mod_deflate, тъй като той компресира съдържанието на файловете и намалява техния размер преди да се изпратят от сървъра към потребителя. В нея се използва генератор за статични страници Apachebench. Той е прост генератор, който позволява генериране и изпращане на заявки с различна последователност и големина. Получените резултати от изследванията с Apachebench дават информация за:

- вида на изпратените заявки до уеб сървъра;
- статистиката на изпълнените заявки;
- информация за уеб сървъра като вид, версия, и хост директория;
- размер на файла, който е изследван, броя на едновременно изпратените заявки и времето необходимо за тяхното обработване.

В експериментите на настоящия дисертационен труд се изследва поведението на хардуерните компоненти – процесора и паметта, при натоварване на сървъра чрез различни по големина и начин на изпращане на заявки към видео файл. Отчита се включването на параметъра на mod_deflate на Apache и неговото влияние върху използването на хардуерните компоненти.

Използвани са две опции на инструмента ApacheBench за генериране на различна последователност от заявки: с (-n) се генерира изпращане на заявки, а с (-c) се генерира едновременно изпращане на заявки. По този начин се симулират различен брой изпратени заявки от потребители за гледане на видео клип. В експериментите се изпращат общо 250 заявки от Apachebench, които са разделени на групи с различна големина. Тези действия, погледнати от перспективата на теория на автоматичното управление, представляват реални входни въздействия с различна амплитуда. Целта на проведените в рамките на дисертационното изследване експерименти е да се сравнят конфигурационните настройки на уеб сървъра Apache по подразбиране с предложените от автора подобрени настройки,

чрез които се променят стойностите на директивите на конфигурационния файл на Apache.

Примерът „ab -n 250 -c 2 http://hs22.iccs.bas.bg/venko.mp4" показва изпращане на общо 250 заявки, групирани по две, които се обработват едновременно от сървъра.

Получените резултати от проведените експерименти при различни настройки на уеб сървъра (по подразбиране и подобрени) са представени на фиг III.11. Лявата колона на фигурата показва, че прилагането на подобрени настройки увеличава производителността на сървъра, което е отчетено чрез показателите: продължителност на теста (Time taken for test) и време, необходимо за обслужване на заявка (Time taken for request).

```
[root@hs12 ~]# ab -n 250 -c 250
http://hs12.iccs.bas.bg:80/venko.mp4
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd,
http://www.zeustech.net/
Licensed to The Apache Software Foundation,
http://www.apache.org/
Benchmarking hs12.iccs.bas.bg (be patient)
Completed 100 requests
Completed 200 requests
Finished 250 requests
Server Software:        Apache/2.2.15
Server Hostname:        hs12.iccs.bas.bg
Server Port:            80
Document Path:          /venko.mp4
Document Length:        77695623 bytes
Concurrency Level:      250
Time taken for tests:   36.547 seconds
Complete requests:      250
Failed requests:        0
Write errors:           0
Total transferred:      19423971750bytes
HTML transferred:       19423905750 bytes
Requests per second:    6.84 [#/sec] (mean)
Time per request:       36546.521 [ms] (mean)
Time per request:       146.186 [ms] (mean, across all
                        concurrent requests)
Transfer rate:          519029.50[Kbytes/sec]received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       12  15  1.5     16      17
Processing:   896 28446 11862.2  34455   36521
Waiting:        8 6677 3099.0   6761    12673
Total:        914 28461 11861.4  34471   36534
Percentage of the requests served within a certain time (ms)
  50%  34471
  66%  35601
  75%  36147
  80%  36150
  90%  36389
  95%  36533
  98%  36534
  99%  36534
 100%  36534 (longest request)
```

```
[root@hs12 ~]# ab -n 250 -c 250
http://hs12.iccs.bas.bg:80/venko.mp4
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd,
http://www.zeustech.net/
Licensed to The Apache Software Foundation,
http://www.apache.org/
Benchmarking hs12.iccs.bas.bg (be patient)
Completed 100 requests
Completed 200 requests
Finished 250 requests
Server Software:        Apache/2.2.15
Server Hostname:        hs12.iccs.bas.bg
Server Port:            80
Document Path:          /venko.mp4
Document Length:        77695623 bytes
Concurrency Level:      250
Time taken for tests:   38.199 seconds
Complete requests:      250
Failed requests:        0
Write errors:           0
Total transferred:      19423971750bytes
HTML transferred:       19423905750bytes
Requests per second:    6.54[#/sec](mean)
Time per request:       38198.849[ms](mean)
Time per request:       152.795 [ms] (mean, across
                        all concurrent requests)
Transfer rate:          496578.38
                        [Kbytes/sec]received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       13  16  1.6     16      18
Processing:   974 30516 11947.9  36139   38172
Waiting:        8 6374 3019.0   6304    12674
Total:        991 30532 11946.9  36156   38186
Percentage of the requests served within a certain time (ms)
  50%  36156
  66%  37262
  75%  37794
  80%  37798
  90%  38029
  95%  38185
  98%  38185
  99%  38185
 100%  38186 (longest request)
```

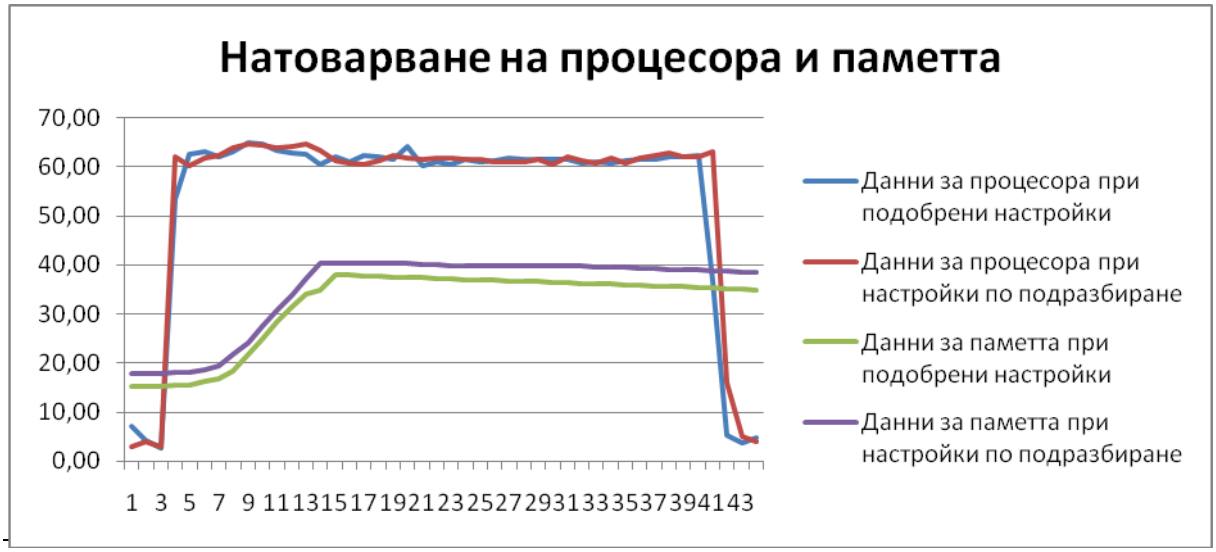
Фиг. III.11. Резултати от експеримента – сравнение на настройките по подразбиране и предложените подобрени настройки

Трябва да се отбележи, че предложените настройки на конфигурационния файл са получени при провеждане на експерименти, в които се наблюдава производителността на сървъра по отношение на заявки за видео файл. Увеличаването на производителността на Apache при обслужване на видео файл подобрява производителността на конкретната хардуерна конфигурация. Това обаче не е универсално решение, което ще подобри

производителността на други компютърни конфигурации или обслужването на друг вид файлове.

По време на изпращане на заявките чрез Apachebench, с програмата SAR е проследено поведението на хардуерните компоненти (процесор и памет), отнасящо се до промените в конфигурационния файл на Apache. SAR е команда на операционната система Linux, която брои и показва дейностите, изпълнявани от операционната система.

По този начин се изследва влиянието на подобрените конфигурационни настройки към основните хардуерни компоненти. На фиг. III.12 са показани промените в натоварванията на процесора и на паметта при различни настройки на конфигурационния файл на Apache. От тях може да се заключи, че натоварването на процесора и паметта е намаляло, съответно на процесора с 2,37%, а на паметта – с 8,93%.

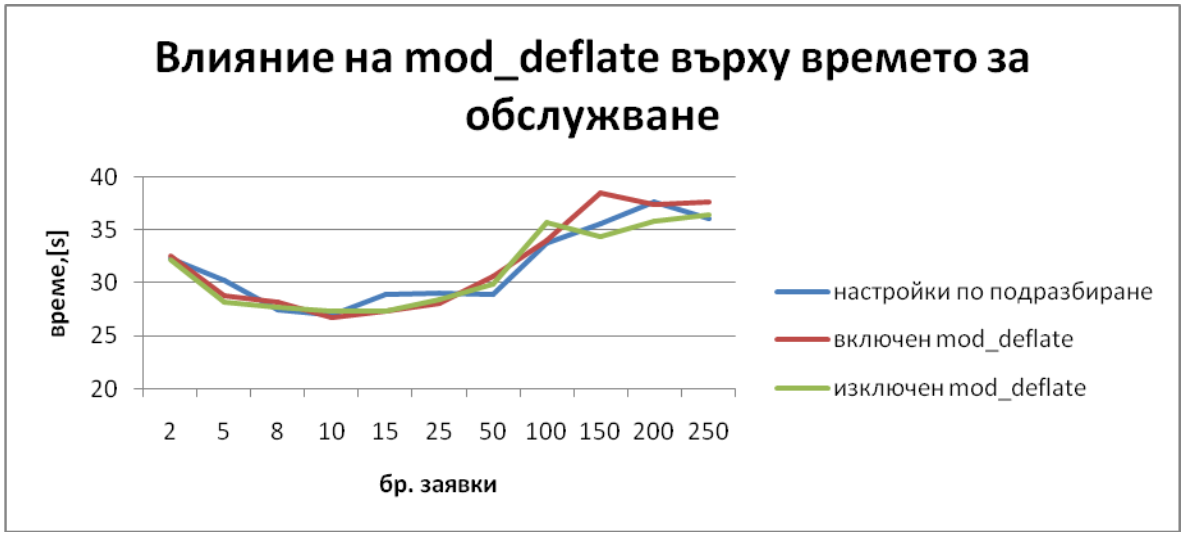


Фиг. III.12. Натоварване на процесора и паметта при различни настройки на конфигурационния файл

В рамките на експериментите за целите на настоящия дисертационен труд е направен анализ и на влиянието на модула mod_deflate върху времето за обслужване на заявки от сървъра. При него са изследвани три варианта за настройка на уеб сървъра Apache. Първият е настройки по подразбиране на Apache, вторият е с оптимизирани настройки на prefork и изключен модул mod_deflate, а третият е с подобрени настройки на prefork и включен модул mod_deflate.

От резултатите, показани на фиг.III.13, може да се направи изводът, че включването на модула mod_deflate не влияе на времето за обслужване на заявките. Неговото самостоятелно включване не е ефективно решение за подобряване на

производителността на системата за обработване на заявки към видео файлове. Това означава, че трябва да се търси друго решение. Резултатите от тези експерименти са описани по-подробно в статия на автора [DimiT-13].

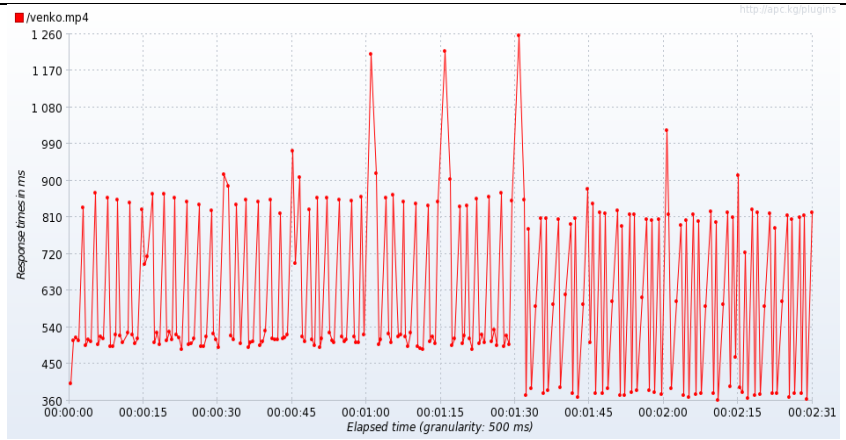
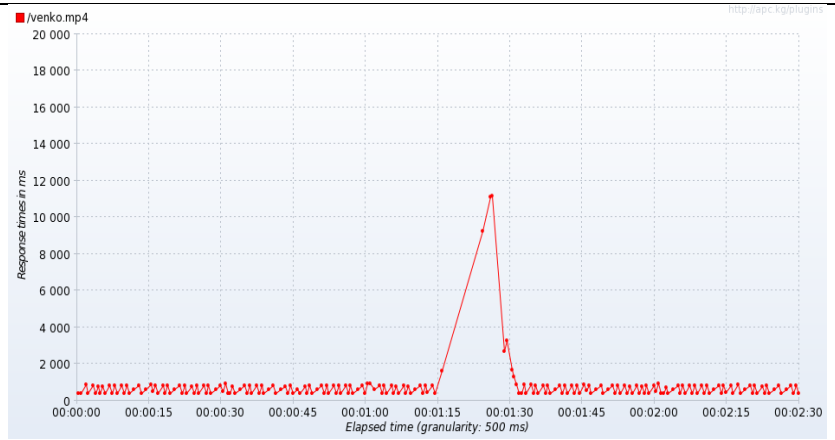


Фиг.III.13. Влияние на mod_deflate върху времето за обслужване на заявките

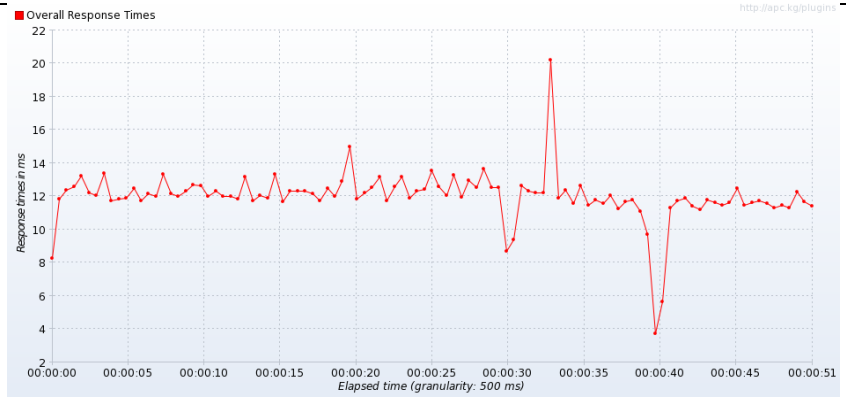
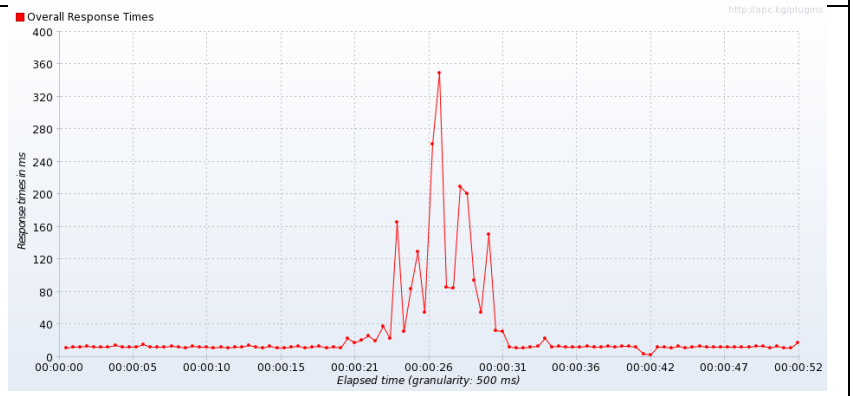
III.2.3. Реализиране на входни въздействия към програмната система чрез различни видове файлове

Обектът на управление в настоящия дисертационен труд – Apache сървър, е изследван и по отношение на други видове входни въздействия – музикални, видео, текстови файлове и база данни. Чрез експерименти е измерена натовареността на хардуерните компоненти – процесор и памет.

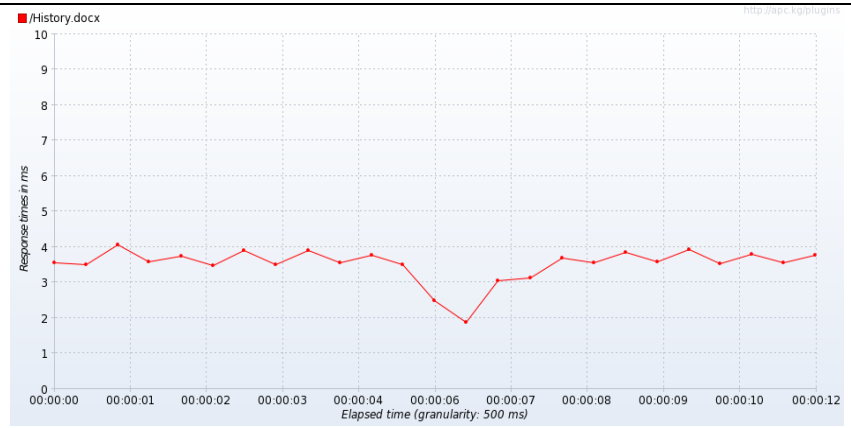
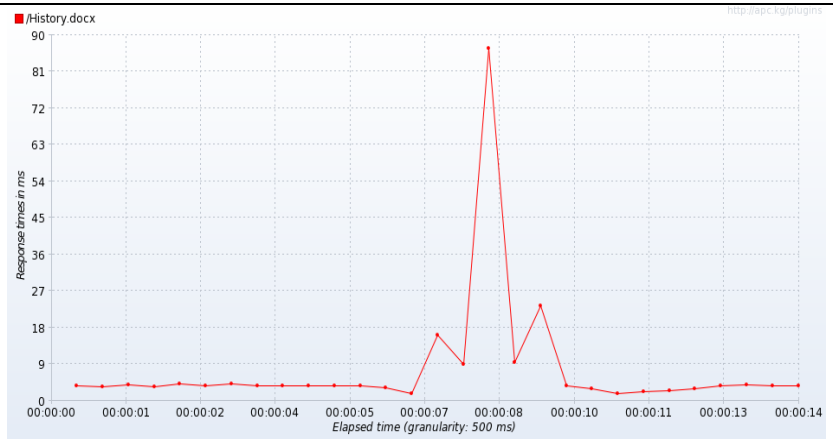
В тях се използва следваща версия на програмата Jmeter за генериране на входни заявки, която предоставя повече възможности. Това позволява входните заявки да бъдат представени като типови сигнали от теория на автоматичното управление, да се генерират с заявки по-реалистичен вид и да се проследи поведението на най-важните хардуерни компоненти – процесора и паметта. Заявките са изпращани с постоянна големина и чрез импулс в определен момент от време, като се наблюдава поведението на показателите за производителност – време за отговор и брой на заявките в приемащата
опашка на уеб сървъра.

	Постоянна големина заявките	Импулс
Видео файл		

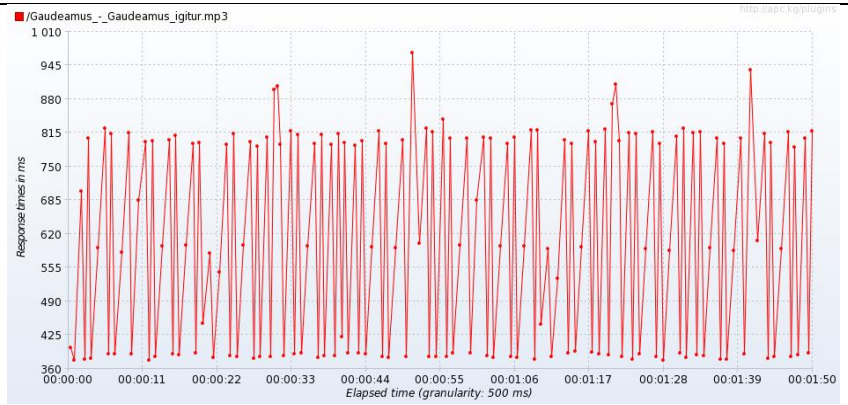
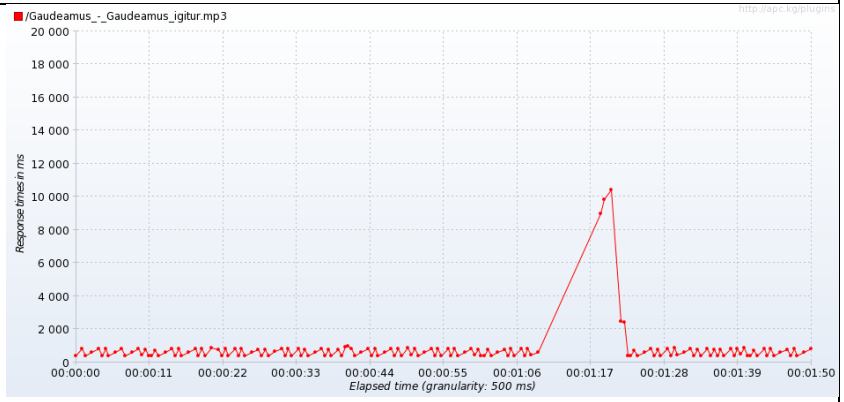
Фиг.III.14-а. Измененията на времето за отговор при входен сигнал – видео файл

	Постоянна големина заявките	Импулс
База данни файл		

Фиг.III.14-б. Измененията на времето за отговор при входен сигнал – файл база данни

	Постоянна големина заявките	Импулс
Текстови файл		

Фиг.III.14-в. Измененията на времето за отговор при входен сигнал – текстови файл

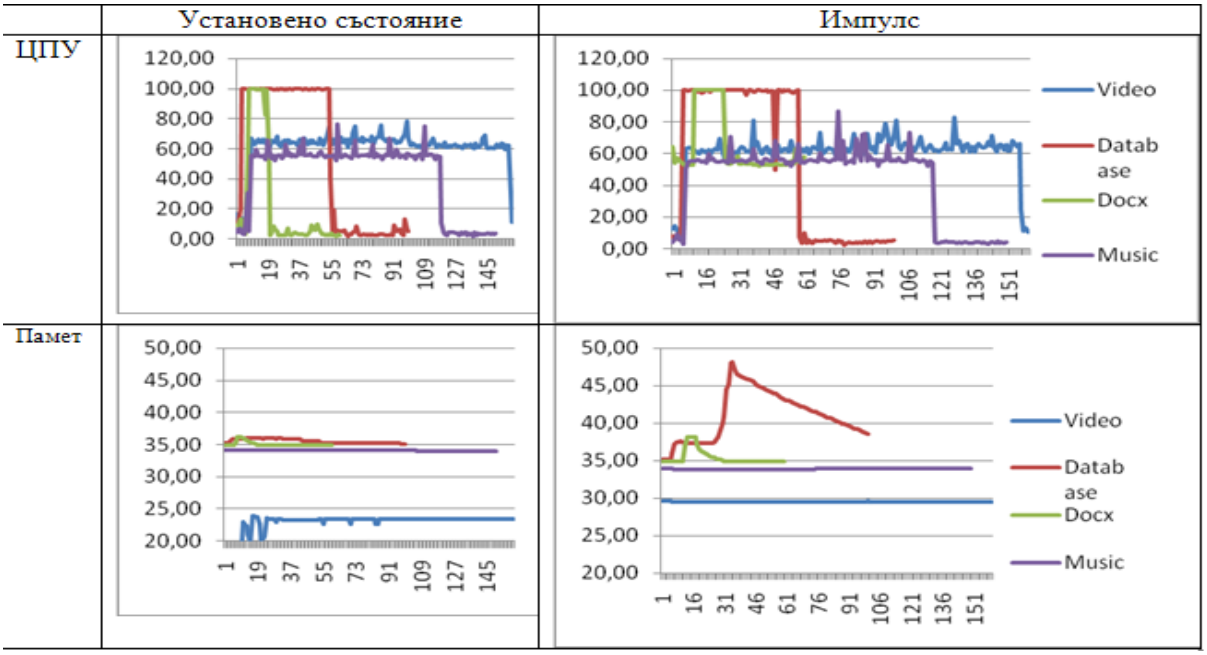
	Постоянна големина заявките	Импулс
Mp3 файл		

Фиг.III.14-г. Измененията на времето за отговор при входен сигнал – mp3 файл

От фиг.14(а-г) се вижда, че когато заявките се изпращат с еднаква големина към програмната система, времето за отговор е постоянно, а при изпращане на импулс към него времето за отговор се увеличава изключително много. Това е предпоставка да се търси решение за намаляване времето за отговор от системата. Изпращането по този начин на заявки към системата, според терминологията, използвана в теорията на автоматично управление, се представя като реално входно въздействие и като единична импулсна функция.

При изпращането на заявките към изследваната система се измерва и натоварването на хардуерните компоненти. Констатира се, че при изпращане на импулс се наблюдават колебания в системата, в използването на хардуерните компоненти процесор и памет (което е показано на фиг.III.15.) Те са представени с по-големи и по-чести флуктуации в използването на процесора и паметта при всички видове файлове.

По-подробно резултатите от тези експерименти са описани в статия на автора [Димит-13].



Фиг. III.15. Реакция на изходните сигнали

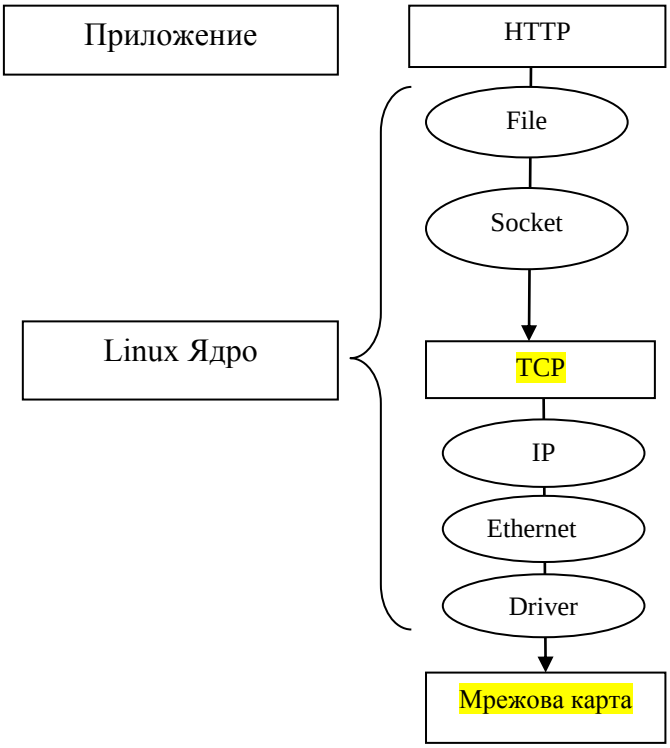
На базата на тези смущения се достига до извода, че е необходима промяна в конфигурационните параметри на уеб сървъра Apache и на TCP комуникационния протокол. Така ще се облекчи работата на програмната и компютърната системи и ще намалят смущенията в тях.

III.3. Управление на настройките на комуникационния протокол TCP на изследваната компютърна система

Създаването на софтуерно приложение за достъп до статиите на научно списание цели да се предложи по-модерен, по-лесен и по-функционален начин за достъп и търсене в списанията, разпространявани от Съюза по автоматика и информатика. Промяната на параметрите на системата, в това число параметрите на конфигурационния файл на Apache и на комуникационния протокол TCP, ще подпомогне едновременния достъп на повече читатели до софтуерното приложение. Това бе потвърдено и от експерименталните резултати, представени в подглава III.2. Въвеждането на допълнителни настройки на различни нива в компютърната система увеличава производителността на програмната система. Например, подходящите настройки на софтуерното приложение и на база данните могат да подобрят производителността на компютърната системата до 20 пъти [Gregg-13]. Подобряването на настройките на устройство за съхраняване на данни може да подобри производителността с 20%. Постигането на такива високи нива на подобрение на производителността на ниво софтуерно приложение е възможно, тъй като при разработването му усилията на екипите са насочени към бързо внедряване, като се отделя малко време за измерване на производителността и оптимизирането ѝ. Това представлява недостатък и е предпоставка за търсене на начини за подобряване на използваните уеб приложение, уеб сървър и операционна система. В търсене на такива начини и с оглед целите на настоящия труд бе направена и справка във форуми за потребители на операционната система Linux и уеб сървъра Apache относно недостатъци и пропуски при разработването им.

III.3.1. Архитектура на използваните системи

Компютърната система може да се представи като последователност от подсистеми, работещи в синхрон. Тя включва подсистемата на TCP протокола, подсистемата на http протокола и подсистемата на i/o. Но параметрите на тези подсистеми имат нужда от настройка за специфичната хардуерна конфигурация, на която работят [MDVHR-01].

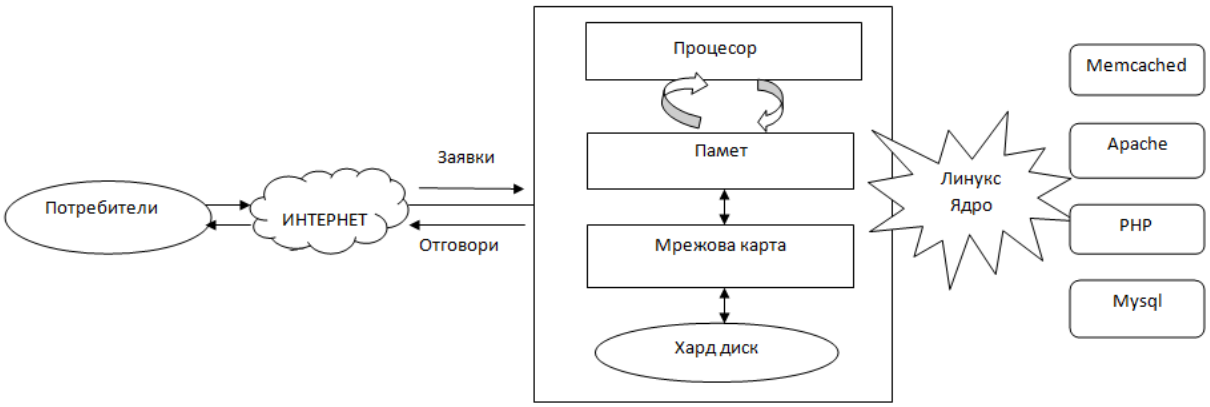


Фиг.III.16 Последователни подсистеми

В дисертационния труд е извършена допълнителна настройка на параметрите в Linux ядрото и по-специално в мрежовите настройки и в TCP комуникационния протокол (в жълт цвят на фиг.III.16).

Както е показано на фиг.III.17, комуникацията между крайните потребители преминава през три основни зони – първата зона е самата програмна система – уеб сървъра Apache, втората зона е компютърната система, която включва операционната система и TCP настройките, а третата зона е пътят по мрежата. За отстраняване на проблемите във всяка от тези зони се използват различни подходи.

Настоящото дисертационно изследване се концентрира върху управление, което може да се извърши в програмната система, т.е. в уеб сървъра, както и във втората зона (компютърната система), където се анализират и подобряват настройките на TCP протокола, и се инсталират програми за подобряване производителността на операционната система.



Фиг.III.17. Комуникацията между потребителите и сървъра

III.3.2. Управление в системата чрез настройки на TCP протокола

Производителността на компютърните системи е ограничена от мрежи, които ги свързват. В комуницакията между различните устройства TCP протоколът изпълнява основна роля. Той е ограничен от два прозореца: прозорец на задръстванията (congestion window) и прозорец на получаване (receive window). За да се използват ефективно компютърните мрежи се прилагат два вида управление на техните процеси: управление на задръстванията, което цели да не се надвишава капацитета на мрежата, и управление на потока, при което има стремеж да не се надвишава капацитета на приемащия прозорец.

За постигане на по-качествено управление се изисква да се направи допълнителна настройка на параметрите на TCP протокола. Параметрите на TCP протокола са разделят в няколко категории, според тяхното предназначение [WigeN-13],[OracC-00],[SoftP-15]:

- За подобряване на производителността

Производителността на TCP протокола се подобрява чрез регулиране размера на опашките, изключване на селективното потвърждение, повишаване на прозореца на задръстванията и др.

Операционната система разполага с настройки за регулиране размера на опашката между мрежовата подсистема на ядрото и драйвера на мрежовата карта. Размерът на опашката трябва да се настрои така, че да няма загуби на данни, поради препълване на локалния буфер. Ако се използват неподходящи настройки на TCP протокола, той ще попадне в състояние на управление на задръстването, което ще ограничи големината на изпратените пакети. Освен това запълнените опашки ще причинят загуби на udp пакетите. За да се извърши настройка на получаващия буфер, се използват два параметъра – txqueuelen (отнася се до дължината на предаващата опашка) и netdev_max_backlog

(определя размера на приемащата опашка).

Като се имат предвид обработваните данни на сървъра, техния трафик, и информация от форуми за поддръжка на Linux системи, е преценено, че не е необходимо да се променят въведените настройки, които са по 1000 пакета.

Изключването на селективното потвърждение (SACK) е оптимизация на TCP протокола при работа в нормални условия с цел подобряване производителността на системата и намаляване на използването на процесора. При това се намалява възможността за Ddos атаки. SACK се изключва с командата /sbin/sysctl -wnet.ipv4.TCP_sack=0.

Настройката на първоначалния прозорец на задръстванията (initcwnd) на сървъра значително подобрява неговата производителност и TCP действията. Постига се по-бързо сваляне на файлове и по-бързо предоставяне на уеб страници. Промяната на настройката се осъществява чрез командата echo > ip route change_default via 192.168.0.1 dev eh0 proto static initcwnd 10.

Производителността на системата се повишава при премахване на бавното стартиране на TCP прозореца. По подразбиране системата започва новите TCP свързвания и TCP свързвания, които са били в свободно състояние повече от 1 сек. с малък размер на TCP window. Когато сървърът използва постоянни свързвания, ще попадне в такава ситуация, което налага и въвеждането на следната настройка TCP_slow_start_after_idle=0, за да се реши проблемът.

Опцията TCP_window_scaling позволява на мащабируемия прозорец (scalabity window) да си промени размерите динамично и така максималния TCP прозорец да е по-голям от 65535 бита като достигне до 1 073 725 440 бита. Командата е /proc/sys/net/ipv4.TCP_window_scaling=1

Опцията TCP_timestamp добавя 12 байта към хедъра на TCP свързването. Чрез командата echo> /proc/sys/net/ipv4/TCP_timestamps=0 опцията се изключва.

- За обслужване на повече клиенти

За сървъри, които обслужват голям брой сесии едновременно, има няколко опции, които трябва да бъдат включени.

Променливата ip_local_port_range определя диапазона на използваните портове от udp и TCP протоколите, като се оптимизира производителността при голям брой клиенти. Настройката се осъществява чрез командата echo 1024 65000 > /proc/sys/net/ipv4/ip_local_port_range. Така се позволява повече локални портове да са на

разположение [STILS-01].

За да могат да се управляват повече свързвания по време на използване на TCP протокола, се променят настройките за времето за затваряне на свързването (close connection) и времето за убиване на остарял процес. Това се извършва чрез опциите TCP_fin_timeout и TCP_keepalive_time [Moura-00].

Променливата TCP_fin_timeout определя времето, което трябва да изтече преди TCP/IP да затвори свързването и повторно да използва нейния ресурс. Чрез намаляване на тази стойност, TCP/IP по-бързо се освобождава от затворени връзки и се предоставят повече ресурси за нови свързвания. По този начин повторното отваряне на връзката между клиента и сървъра изисква по-малко ресурс, отколкото при създаването на нова връзка. Така се подобрява производителността на системата и се намаляват връзките в състояние time_wait.

Настройката на keepalive включва три променливи, които се съхраняват в директорията /proc/sys/net/ipv4/. Променливата keepalive_time задава интервала между последния изпратен пакет с данни и първия keepalive опит за потвърждаване, и потвърждава, че свободното свързване (idle connection) все още е в активно състояние. Чрез промяната на този параметър се задава по-кратък времеви интервал. Променливата TCP_keepalive_intvl определя интервала между последователни keepalive опити, независимо дали свързването осъществява обмен на данни. Променливата TCP_keepalive_probe определя броя опити, при които не е получен ack отговор и връзката се маркира като унищожена, изпраща се съобщение за timeout изтекло време и се уведомява приложния слой [Speed-16].

Опцията TCP_max_syn_backlog определя максималния брой свързващи заявки, които не са получили потвърждение от клиента. Стойността по подразбиране е 1024, но ако се очаква сървърът да попада в пренатоварено състояние и машината има повече от 128 MB памет, се препоръчва тази стойност да се увеличи [Garre-09].

- за подпомагане работата на web server

Когато на използваната машина е инсталиран уеб сървър, който обслужва много кратки TCP свързвания, е подходящо да се настроят следните опции.

С параметъра TCP_tw_recycle се включва бързо изчистване на time_wait сокетите. Особеността при тази опция е, че може да създаде проблеми, ако се използва със системи за балансиране на натоварването или със системи, работещи с резервирани доставчици. В

нашия случай не използваме такива системи, така че е уместно включването на тази опция.

TCP_reuse позволява повторно използване на time_wait състоянията за нови свързвания, когато това е сигурно от гледна точка на TCP протокола. Тя представлява по-безопасна алтернатива на TCP_tw_recycle. TCP_reuse е много полезна за приложения като уеб сървър и системи за балансиране на натоварването, при които има много на брой отворени свързвания, оставени в състояние time_wait. Тази опция за повторно използване на сокета може да намали натоварването на сървъра.

TCP_synack_retries определя броя synack пакети, изпратени преди ядрото да откаже свързването. За да отвори друга страница, ядрото трябва да изпрати синхронизиращо потвърждение. Това е втората стъпка от тристранното ръкостискане.

С опцията /Net/core/someconn се настройва ограничението на сокета listen()backlog. Броят заявки, приети от сокета по подразбиране, е 128. Стойността трябва да бъде увеличена значително, за да може системата на обработва голям брой заявки едновременно [HTCAG-10].

- Други настройки

Опциите w(r)mem_max, TCP_w(r)mem и тези за споделената памет не се променят за изследванията за дисертацията. В някои литературни източници се препоръчва тяхната промяна, но в нашия случай те надвишават многократно препоръчителните стойности.

Настройките на cat /proc/sys/net/ipv4/TCP_mem не е желателно да се променят по две причини: стойността се задава в страници (pages), а не в байтове и това лесно може да доведе до объркване; самонастройката в Linux се определя много добре на базата на размера на паметта [Garre-09].

III.3.3. Управление на изследваната система чрез настройки на програмната система

Apache уеб сървърът позволява лесно изграждане на среда за поддържане на уеб сайтове с минимални конфигурации и усилия от страна на администратора. Но трябва да се има предвид, че при използването на настройките по подразбиране може да се изисква използване на голям ресурс от компютърната конфигурация, особено що се отнася до памет. Това поставя нелеката задача за оптимизиране на максимална производителност на системата, тъй като липсват лесни и бързи правила за нейното решение. За да се постигне максимална производителност се изискват много експерименти с различни опции и условия, които включват вида на заявките, изпълнявани от сървъра, оптимизиране на база

данните, кеширане на php скрипта. В настоящия дисертационен труд са разгледани и подобрени всички тези условия на използваната система [Avoya-11].

Оптимизиране на модули, използвани от Apache

Приложени са няколко вида настройки на уеб сървър Apache за постигане на максимална производителност, както следва:

- Изключване на ненужни модули

Първото подобрение, което се изисква да се извърши, е премахването на ненужните модули за статично конфигуриране, като по този начин се намалява използването на памет. По-долу е показан конфигурационният код, чрез който това се прави:

```
# LoadModule authn_dbm_module modules/mod_authn_dbm.so
# LoadModule authn_anon_module modules/mod_authn_anon.so
# LoadModule authz_dbm_module modules/mod_authz_dbm.so
# LoadModule authz_owner_module modules/mod_authz_owner.so
# LoadModule authnz_ldap_module modules/mod_authnz_ldap.so
# LoadModule ext_filter_module modules/mod_ext_filter.so
# LoadModule substitute_module modules/mod_substitute.so
# LoadModule ldap_module modules/mod_ldap.so
```

- Използване на модули с приоритет

В този дисертационен труд се препоръчва да се използва модулът mod_disk_cache вместо модула mod_mem_cache, тъй като вторият модул има за задача да споделя своята кеширана информация с различни процеси на Apache, което води до високо използване на памет и ниска производителност на системата. По този начин вероятността една и съща страница да се обслужи два пъти в един процес е много малка.

Използва се плоска йерархия за конфигуриране на модула mod_disk_cache, за да се гарантира използването на CacheDirLength=2 и CacheDirLevels=1 и да се постигне бързо изчистване на кеширащата директория.

- Символни връзки

При включване на модула FollowSymLinks и изключване на модула SymLinksIfOwnerMatch се позволява осъществяване на символни връзки. В противен случай Apache създава допълнителни системни извиквания за всяко име на файл, за да гарантира, че свързването не е символна връзка. Когато FollowSymLinks опцията е настроена, сървърът следва символните връзки за зададената директория.

Начин на задаване:

<Directory />

Options FollowSymLinks Indexes

AllowOverride None

</Directory>

- Промяна на последователността в log файловете (Piped logging)

По подразбиране Apache записва данните директно в log файла. Ако се използва piped logging, последователността на записване в log файла се променя, като по този начин се намалява недостигът на памет и се постига по-голяма гъвкавост на записване на данните.

Друг начин на подобрение на съхранението на файловете за регистри е чрез запис на друг диск.

Начин на задаване:

CustomLog "|/opt/lampp/bin/rotatelogs /opt/lampp/logs/access_log.%a 86400" common

ErrorLog "|/opt/lampp/bin/rotatelogs /opt/lampp/logs/error_log.%a 86400"

- Определяне на индекс файла

Препоръчва се да се избягва употребата на звездичка (*) в индекс директорията, а да се използват определени файлове като index.html и index.php

- Включване и изключване на модули

Включването на модула keepalive позволява да се осъществява постоянно свързване между клиента и сървъра. Начинът на задаване е KeepAlive On.

Изключването на HostnameLookups спира dns търсенето и премахва времето за изчакване (латентност) за dns. По-добрият вариант за търсене на хостваните сайтове е да се използват IPадреси. Начинът на задаване е HostnameLookups Off.

Изключването на модул .htaccess, заедно с командата allowoverride none, премахва проверката на Apache в .htaaccess за всяка заявка.

<Directory />

AllowOverride none

</Directory>

ExtendedStatus извършва допълнителни системни извиквания са всяка заявка с цел подобряване статистика на процесите в Apache. Изключването му води до улесняване работата на Apache. Начин на задаване: #ExtendedStatus On

- Стартиране на модули за компресиране

При стартиране на модулите mod_gzip или mod_deflate, които компресират съдържанието на файловете преди да се изпратят на клиента, се намалява размерът на предаваните данни. Когато данните пристигнат при клиента, той ги разархивира.

Настройка на опциите Expires, Etag и cache-control-headers се използва в кеша, за да се определи давността на файла. В противен случай няма да има ползва от кеша. Друга възможност представлява доставянето на кеш в отделно дисково пространство, за да е по-бърз достъпът, без да се забавят другите процеси.

- Memcached

Използването на кеша за често обработвани данни и сесии предоставя чудесна възможност за ускоряване на времето за изобразяване, за осъществяване на по-бърз достъп до данните и за намаляване на комуникацията с база данните. Допълнителният модул за кеширане Memcached е конфигуриран да работи с използваната в дисертационните изследвания система, както е показано на фиг.III.4.

memcache

memcache support	enabled
Version	3.0.8
Revision	$Revision: 329835 $

Directive	Local Value	Master Value
memcache.allow_failover	1	1
memcache.chunk_size	32768	32768
memcache.compress_threshold	20000	20000
memcache.default_port	11211	11211
memcache.hash_function	crc32	crc32
memcache.hash_strategy	consistent	consistent
memcache.lock_timeout	15	15
memcache.max_failover_attempts	20	20
memcache.protocol	ascii	ascii
memcache.redundancy	1	1
memcache.session_redundancy	2	2

Фиг.III.18. Memcached

Управление чрез настройка на директиви

Настройването на директивите на уеб сървър Apache се реализира чрез промяна на текстовия файл httpd.conf, който съхранява множество опции, като се добавят и премахват коментари или се изтриват опции. Редактирането на конфигурационния файл на Apache е най-използваният метод за конфигуриране на сървъра и е изключително предпочитан от администраторите, работещи с операционна система Linux. Директивите имат точно

определен синтаксис, който наподобява програмен език. Те представляват инструкции към сървъра Apache, които му помагат да работи по определен начин и да открива необходимите си ресурси. Най-общо директивите могат да се разделят на основни директиви и на конфигурационни директиви. Основните директиви са най-важните директиви. Те са компилирани в изпълнимия файл на Apache и са налични по подразбиране. Използват се за нормалното функциониране на уеб сървъра Apache. Допълнителните директиви се добавят от администратора чрез допълнителни модули.

За да се улесни конфигурирането на конфигурационните файлове на Apache, те се разделят на три основни раздела: раздел за обкръжението, раздел за конфигурирането на основния сървър и раздел за виртуалните хостове. Директивите от раздела на обкръжението влияят върху цялостната работа на уеб сървъра. Те определят управлението на процесите на Apache. Директивите от раздела за конфигуриране на основния сървър определят параметрите, използвани от него. Директивите от раздела за виртуалните хостове се отнасят за конфигурирането на виртуалните хостове – това са всички сървъри, настроени при Apache извън главния сървър. Използването им позволява да се хостват по повече от един уеб сайт с различни домейни на един уеб сървър.

Поради увеличените функционалности на Apache във версия 2.4, директивите са разделени чрез нови конфигурационни файлове. Директивите за подобряване работа на уеб сървъра Apache, използвани в дисертацията, се съхраняват в допълнителни модули http_default.conf, http_mpm.conf, http_vhost.conf и в основния конфигурационен файл http.conf. За да се подобри производителността на уеб сървъра Apache е необходимо да се изследват и променят неговите основни директиви. По този начин се цели да се подобри производителността му. В настоящия дисертационен труд е приложено е управление чрез следните директиви:

- **Директива Timeout** – задава времето, което Apache трябва да изчака, преди да изпрати съобщение за изтекло време за отговор на заявката. По подразбиране Timeout е много дълго време – 5 минути за един процес. С това се цели да се осигури възможност да се управляват разнообразни ситуации при пристигане на клиенти. Редица източници посочват като разумна стойност на времето за отговор 45 секунди или по-малко. Чрез намаляване на този параметър се въздейства на възможността за Ddos атаки. При използваните сценарии (посочени в таблица 2), настройките са 40 и 45 секунди [Freit-05].
- **Директива KeepAlive –** позволява на клиентите да установят множество връзки в определен момент от време, което установява процеса на изтегляне на данни.

- **Директива MaxKeepAliveRequests –** задава максималния брой заявки, разрешени за всяка връзка. Използва се за определянето на максималния брой на заявките, които могат да бъдат изпратени до сървъра за всяка отделна връзка при активиране на опцията за постоянни връзки (по подразбиране MaxKeepAliveRequests е 100).

Поради факта, че изследваният в дисертацията уеб сървър Apache обслужва уеб страници с малко количество данни, включващи по няколко снимки, php скриптовете и css код, то зададената стойност е 14. По този начин се позволява да се намали натоварването на паметта.

- **Директива KeepAliveTimeout**

Задава времето, след което Apache ще прекрати постоянната връзка, ако клиентът не е изпратил повече заявки.

Особеност при настройката на keepalivetimeout е изборът на подходяща стойност. Зададената стойност не трябва да е много голяма, тъй като дъщерните процеси се блокират в изчакване на заявки от клиента и не могат да обслужват нови клиенти. Задаването на по-малка стойност е по-добро решение. Прието е стойността на keepalivetimeout да бъде от 1.5 до 2 пъти по-голяма от времето за извличане на страницата (page load speed). В изследването е измерено време 600 ms за извличане на страница чрез инструментите за инспектиране на страници на Chrome, т.е. keepalivetimeout може да се настрои за 1-2 секунди. В същото време се препоръчва тази стойност да е между 2-5 секунди. Ето защо избраното време на keepalivetimeout е 2 секунди [Ram-05].

- **Директива StartServers**

Задава броя на дъщерните процеси, които се създават при стартирането на сървъра. Тъй като Apache е интелигентен сървър, той преценява натоварването си преди да определи броя на стартираните дъщерни процеси в определен момент.

- **Директиви MinSpareServers и MaxSpareServers**

Задават минималният и максималният брой процеси, които са свободни и изчакват заявки в даден момент от време. Директивата MinSpareServers определя автоматичното създаване на минимален брой дъщерни процеси, а директивата MaxSpareServers определя максималния брой извикани дъщерни процеси. Ако извиканите процеси са повече, тогава родителският процес отстранява излишните процеси.

В случаи на натоварени сайтове StartServers, MinSpareServers и MaxSpareServers е по-добре да не се корегират, тъй като Apache има опция да ги саморегулира много добре.

Понякога се изисква време за започване обслужването на пристигналите заявки от сървъра. Това забавяне зависи от операционната система, от броя на предварително инсталираните модули и от процесите, натоварващи машината. Поради тази причина е добре настройките на StartServers и MinSpareServers да имат високи стойности. Така, ако се получат големи натоварвания, новите дъщерни процеси ще са готови незабавно да обслужват заявките. Когато стойностите на MaxSpareServer са близки до MaxClients, последният е настроен да използва пълните възможности на ресурсите. Така системата е в състояние да отговори на заявките с максимална скорост, ако броят на заявките не е по-голям от Maxclients. Чрез промяна на тези параметри, в условия на високи натоварвания, сървърът ще отдели повече ресурс за управлението на Apache.

Когато при наблюдение на сървъра се установи, че броят на активните процеси е малък, поддържането на ниски нива на MinSpareServers е добро решение. Така дъщерните процеси ще бъдат прекратени преди да се използват.

Ако сървърът изпълнява други процеси освен уеб обслужване, малките стойности на MinSpareServers ще предоставят паметта за другите процеси, а не за дъщерните процеси. Ако натоварването на сървъра е високо, при генериране на голям брой клиентски заявки, от сървъра се изисква бърз отговор за всички клиенти. При задаване на големи стойности на MinSpareServers новите дъщерни процеси ще бъдат създадени предварително и ще са в готовност да изпълнят голям брой заявки изпратени към сървъра.

- **Директива MaxRequestsPerchild**

Тази директива задава максималния брой заявки, които един дъщерен процес обслужва преди Apache да го прекъсне [Liqui-16].

Когато Apache уеб сървър се използва за обслужване на програмен код, в който няма разпиляване на памет, задаването на висока стойност (1000-2000 броя заявки) за параметъра MaxRequestsPerChild е приемлива. Стойността 0, зададена по подразбиране, означава безкраен брой процеси. При грешно написан код това може да доведе до затруднения в работата на Apache и излишно използване на памет. Задаването на малка стойност също не е приемливо, понеже създаването на нови процеси причинява пренатоварване.

- **Директива MaxClients**

Задава максималния брой клиенти, които могат едновременно да се свържат в даден момент от време. Например, MaxClients 150 означава, че Apache не може да обслужва повече от 150 заявки в даден момент [Апу-04]. За да се определят стойностите

на основните директиви в конфигурационните файлове на Apache е необходимо да се наблюдава и анализира използваната памет от системата [Ram-05].

Анализ за използваната памет от системата

Определянето на паметта, използвана от системата при големи натоварвания, не е лесна задача, особено ако Apache работи с Mysql. Обикновено Apache се използва съвместно с Mysql. Когато се увеличава използването на паметта от Apache, се увеличава и паметта на Mysql. Чрез скрипта mysqltuner.pl се установява максималното количество памет, която се използва от Mysql [Timme-16]. В нашия случай за използваната конфигурация максималното количество памет за Mysql е 449.2 MB. Препоръчва се да се предвиди определено количество памет и за други приложения.

Определянето на стойността за maxclients се съгласува с максималната памет, използвана от Mysql, и с паметта, използвана от всички приложения.

Програмният код ps_mem.py определя паметта, използвана от един процес на Apache, както и от всички други приложения.

```
root@hs22 bin]# ./ps_mem.py
Private  +   Shared  =  RAM used       Program

 4.0 KiB +  13.5 KiB =  17.5 KiB       portreserve
 4.0 KiB +  26.0 KiB =  30.0 KiB       abrtd
 4.0 KiB +  28.0 KiB =  32.0 KiB       hald-runner
 4.0 KiB +  32.5 KiB =  36.5 KiB       hald-addon-rfkill-killswitch
 4.0 KiB +  33.5 KiB =  37.5 KiB       hald-addon-input
 4.0 KiB +  46.5 KiB =  50.5 KiB       wpa_supplicant
 8.0 KiB +  51.0 KiB =  59.0 KiB       dbus-launch (2)
32.0 KiB +  28.5 KiB =  60.5 KiB       atd
 22.3 MiB +  12.4 MiB =  34.8 MiB       /opt/lampp/bin/ (12)
 36.2 MiB +   2.0 MiB =  38.2 MiB       TeamViewer.exe
 22.3 MiB +  20.8 MiB =  43.1 MiB       monitorix (2)
 95.5 MiB +   1.6 MiB =  97.1 MiB       nautilus
130.0 MiB + 365.5 KiB = 130.4 MiB       mysqld (2)
155.9 MiB +   2.4 MiB = 158.3 MiB       TeamViewer_Desktop
224.8 MiB +   1.3 MiB = 226.1 MiB       firefox
---------------------------------
                           1.0 GiB
=================================
```

Фиг. III.19. Действие на програмен код ps_mem.py

От фигура III.19 се вижда, че паметта, използвана от един дъщерен процес на Apache, е 12.4 MB; че използваната памет от Mysql е 130,4 MB; а паметта, използвана от всички процеси е 1GB. С получените данни се изчислява стойността на параметъра maxclients:

$$maxclients = \frac{2048-(449+1024)}{12,4} = 46,37 \qquad (1)$$

Ето защо въведената стойност за maxclients при преконфигурираното на конфигурационните файлове на Apache в настоящия дисертационен труд е 50 клиента.

Друг начин за изчисляване на параметъра maxclients е представен в [WampD-15]. Той позволява чрез команди на езика awk да се установи средното използване на памет от един процес на Apache. То е Avg=119776/30=3.899MB, а общото количество памет, използвано от всички процеси на Apache, е 116,967 MB. Командата е ps -aylC /opt/lamp/bin --sort:rss | awk '{sum+=$8; ++n} END {print "Tot="sum"("n")";print "Avg="sum"/"n"="sum/n/1024"MB"}'.

По този начин се използва по-малко хардуер ресурс за установяване на показателите на системата, тъй като не е необходимо да се стартират и компилират допълнителни програми.

Чрез програмата free, която показва стойностите на свободната и използваната памет от системата, в рамките на дисертационните експерименти бе установено, че използваната памет от всички програми е 1545 MB, наличната физическа памет е 1875 MB, т.е. свободната памет в системата е 330 MB, а от известната големина на паметта за един процес на Apache се определя параметърът maxclients на конфигурационния файл на Apache. Въз основа на получените данни по опростена формула се изчислява стойността за maxclients.

$$maxclients = \frac{\text{свободна памет}}{\text{дъщерен процес памет}} = \frac{(1875-1545)}{3{,}899} = 84{,}637 \qquad (2)$$

В рамките на експериментите за целите на настоящия дисертационен труд двата начина за изчисляване на MaxClients се извършват при два режима на работни натоварвания на системата. В първия режим на работа всички процеси, работещи в операционната система, използват 1,5 GB памет, а дъщерните процеси на Apache използват 3,899 MB. Във втория режим на работни натоварвания на системата всички процеси, работещи в операционната система използват 1 GB памет, а дъщерните процеси на Apache използват 12,4 MB. Трябва да се отбележи, че данните са получени при генериране на един и същ брой клиенти към системата.

На базата на направените изчисления за използване на паметта на хардуерната конфигурация се предлагат два варианта за промяна на настройките на модула Prefork на Apache.

При първия се намаляват стойностите на параметрите му. Целта е да се разгледат възможностите за реакция на системата и да се реализира управление, при което при по-малък брой потребители на уеб приложението, ресурсите на системата да се използват от другите програми, работещи с операционната система. Във втория се увеличават стойностите на параметрите на prefork модула. По този начин се изисква повече хардуерен ресурс за уеб сървъра Apache и се стартират допълнителни дъщерни процеси при по-голям брой потребители на системата. Това се осъществява чрез промяна на параметрите, както е показано в таблица III.2. В нея също така са отбелязани параметрите на директивите, приложени към системата при двата варианта, наименувани съответно bestSettings и highSettings. Това са основни директиви за настройка на уеб сървъра Apache, които се използват за определяне на процесите, изпълнявани от него.

Таблица III.2. Параметри на директивите на Apache

Prefork	bestSettings	highSettings
StartServers	3	10
MinSpareServers	3	10
MaxSpareServers	10	15
MaxRequestWorkes	50	84
MaxConnectionPerChild	1000	1000
Timeout	40	45
MaxKeepAliveRequests	14	14
KeepAliveTimeout	2	2

В резултат от направените промени в настройки на директивите за конфигуриране на уеб сървъра Apache се установи, че се променя и натоварването на основните компоненти на системата – процесора и паметта.

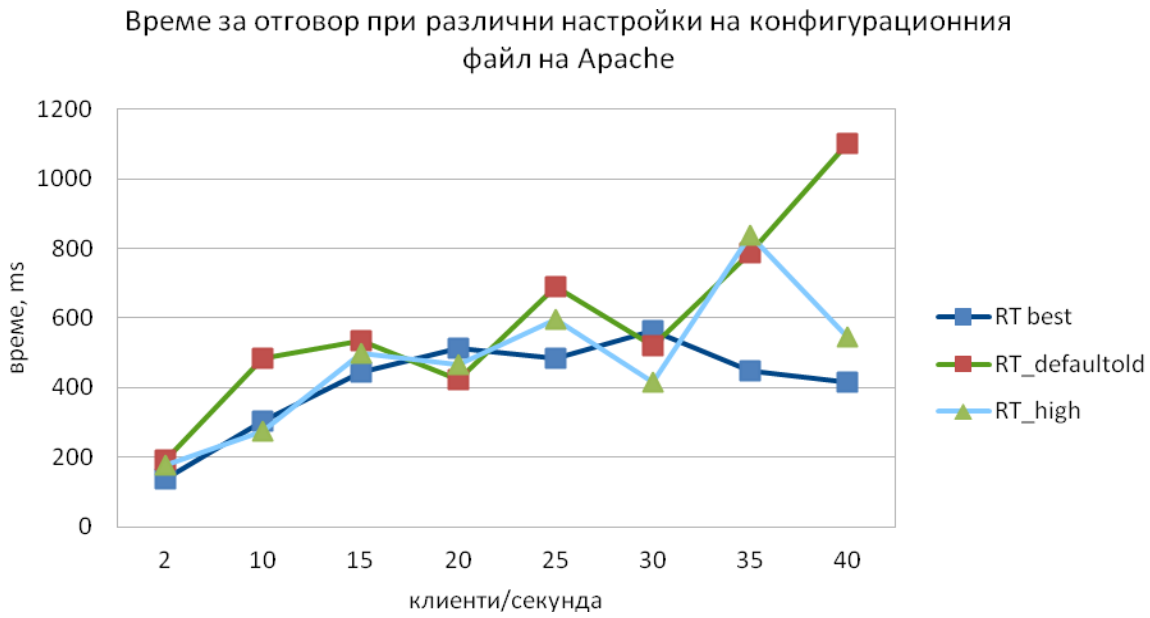
В таблица III.3 са показани промени, които се наблюдават в използването на процесора и паметта за различни настройки на конфигурационния файл (посочени в таблица III.2). Използването на процесора при настройки highSetting е представено в колона “cpu_high”, а при bestSettings – в колона “cpu_best”. Използването на паметта при настройки highSetting е показано в колона “mem_high”, а при bestSettings – в колона “mem_best”. От резултатите става ясно, че когато постъпват по-малко от 20 клиента, настройките highSetting са по-ефективни за обслужването на клиентите, понеже се

използва по-голям ресурс на процесора и по-малък ресурс на паметта. За по-голям брой клиенти използването на конфигурационните настройки bestSettings е по-ефективно, тъй като се използва по-малко натоварване на паметта и по-голямо на процесора. Чрез прилагането на два варианта на настройки за конфигурационния файл се постига по-бързо обслужване на клиенти и по-ефективно използване на наличните хардуерни ресурси.

Таблица III.3. Промени (в %) на използваните памет и процесор

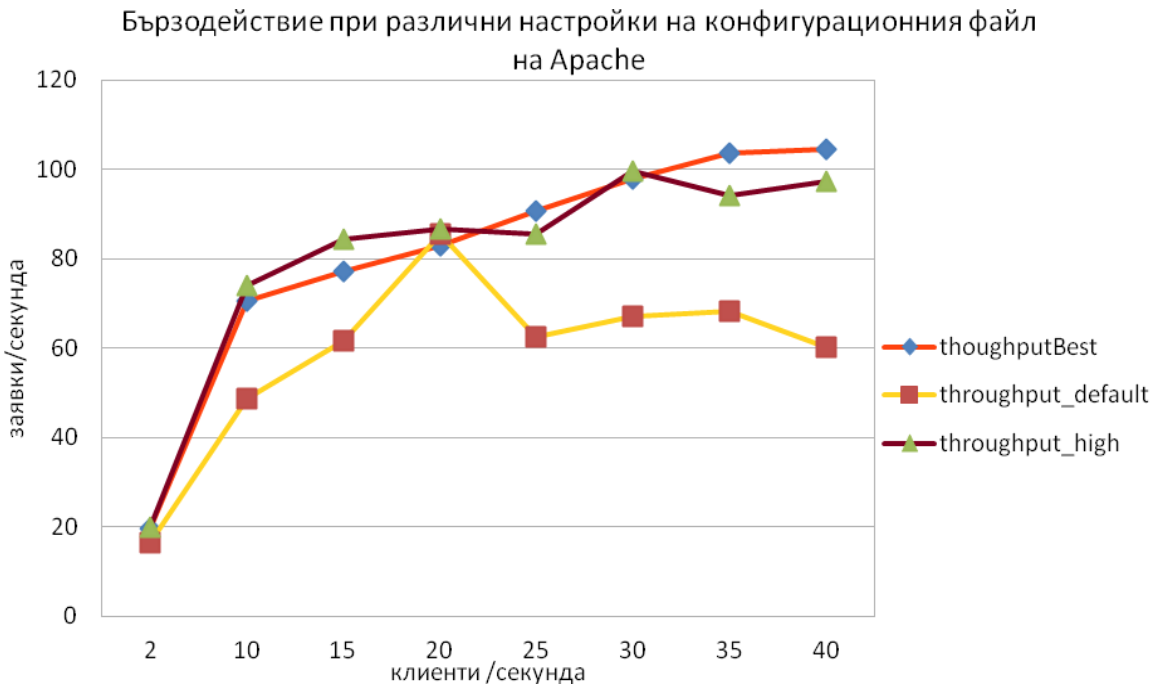
Клиенти	cpu_best, %	mem_best, %	cpu_high, %	mem_high, %
2	34,123	95,129	43,4142	88,845
10	39,974	94,84	43,324	89,271
15	37,989	91,729	36,188	93,521
20	39,934	94,284	42,788	93,209
25	40,072	92,164	37,689	84,995
30	40,319	91,923	40,313	85,19
35	40,559	90,401	44,864	84,963
40	46,41	82,648	42,476	89,137

Ефективността на направените промени в конфигурационния файл на Apache се бе изследвана и чрез показателите за производителност на системата – бързодействие и време за отговор. Наблюденията в тях са представени с оглед на три случая: с настройки по подразбиране (default); с преконфигурирани настройки на конфигурационния файл на Apache bestSettings; с преконфигурирани настройки на конфигурационния файл на Apache highSettings. Резултатите от направените промени са предствени графично на фиг. III.20 и фиг. III.21.



Фиг. III.20. Изменение на времето за отговор при различни настройки на конфигурационния файл

От фиг. III.20 се вижда, че за настройки по подразбиране времето за отговор (RT_defaultold) е по-голямо, отколкото за двата случая с преконфигурирани настройки. Резултатите, показани на фиг. III.20 демонстрират, че когато сървърът работи с настройки по подразбиране, времето за обслужване на потребителските заявки е по-голямо. Те също така показват, че предложените два варианта за преконфигурирани настройки на параметрите на директивите на Apache водят до по-бързо обслужване на потребителските задачи.



Фиг.III.21. Бързодействие при различни настройки на конфигурационния файл.

От фиг.III.21 може да се направи изводът, че бързодействието с настройките по подразбиране е по-малко в сравнение с двата случая на преконфигурираните настройки. От друга страна, бързодействието с настройки highSettings е по-добро, когато се обслужват по-малко от 20 клиента едновременно.

Създаване на програма за автоматично преконфигуриране на конфигурационните файлове на Apache

Създаването на управление за автоматично преконфигуриране на Apache цели по-бързо обработване на заявките в зависимост от броя на потребителите.

От представените резултати на фигури III.2 и III.3 става ясно, че програмата за автоматично преконфигуриране на Apache трябва да използва настройките, наречени highSettings, когато приложението използва до 20 потребителя, а настройките bestSettings– когато потребителите са повече от 20. В програмния код за преконфигуриране на Apache се използват Linux програмата PS и езикът AWK.

PS е една от основните програми на Linux за преглеждане на процесите, протичащи в системата. Тя предоставя кратък преглед на текущите процеси, както и подробна информация относно потребителско ID, използване на процесора, използване на памет, име на командата и др.

Параметри на ps aux:

User – собственик на процеса;

PID – идентификационен номер на процеса;

% cpu – съотношение между процесорното време (cputime) и времето за изпълнение на процеса;

%mem – съотношение между rss (resident set size) и физическата памет на машината;

VSZ – виртуална памет, използвана от целия процес (в KB);

RSS – част от RAM паметта (в KB), която се използва от процеса в хода на изпълнението му.

Преконфигурирането на уеб сървъра Apache се реализира чрез измерване на параметъра %cpu. В програмата за управление той измерва колко процесорно време се използва от процеса. По този начин се преконфигурира конфигурационния файл на Apache. Този параметър се проверява в цикъла while, след което се прави избор кои конфигурационни файлове да се използват на базата на неговата стойност.

Управляващата програма е от типа:

```
#!/bin/bash
#!/bin/sed
while :
do
for i in $( ps aux | grep opt/lampp/bin/httpd |awk '{s+=$3} END {print s}' );
do
if [ $i \> 3.9 ]
then
cp httpd-mpm_2.conf httpd-mpm.conf
cp httpd-default_2.conf httpd-default.conf
sudo /opt/lampp/lampp reloadApache
echo "Load new settings!"
elif [ $i \< 1.00 ]
then
cp httpd-mpm_1.conf httpd-mpm.conf
cp httpd-default_1.conf httpd-default.conf
sudo /opt/lampp/lampp reloadApache
```

```
echo "Load settings by default!"
fi
done
sleep 20
done
```

Представеният програмен код осъществява автоматично преконфирурирane на уеб сървъра Apache на потребителско ниво, в зависимост от броя потребители на уеб приложението. По този начин се прилага по-ефективно обслужване на потребители на разработеното приложение. Зададените стойности на конфигурационните файлове са представени в таблица III.2.

III.3.4. Управление на параметрите на файловата система

В точки III.3.2 и III.3.3 са описани параметрите на комуникационния протокол TCP и на уеб сървъра Apache, за които е направена пренастройка с цел повишаване производителността на изследваната в този дисертационен труд система. От литературния обзор става ясно, че производителността на изследваната система може да се подобри и чрез промяна на параметрите на файловата система Swapiness и file-max. Те са от съществено значение за нейното функциониране.

- Swapiness

Този параметър определя при какво натоварване на паметта процесите да започнат да използват от swap паметта.

Факт е, че swap паметта е по-бавна от ram паметта, което означава, че ако процесите се изпълняват от swap паметта, времето за отговор на системата и на приложенията нараства. Ако процесите се изпълняват често, то те се насочват към swap паметта. Ето защо, за да се подобри производителността на системата, в рамките на експерименталната работа на настоящото дисертационно изследване, бе променено нивото на натоварване, след което да започне да се използва swap паметта. Това се осъществява чрез командата: echo 10 > /proc/sys/vm/swappiness[Askub-12].

- file-max

Този параметър определя максималния брой едновременно отворени файлове.

В Linux всеки отворен мрежови сокет изисква файлов дискриптор. При увеличаването на стойността на file-max се гарантира, че файлови дискриптори няма да

влияят на способността да се управляват едновременно много заявки. По този начин се намалява времето, в което сокетите ще останат в състояние time_wait. В интернет ръководствата за настройка на операционната система се препоръчва file-max параметърът да бъде по-голям от 32000. В настоящото изследване приложената стойност на file-max е 188154 (Fs.file-max= 188154).

III.3.5. Подобряване на производителността чрез други механизми

Подобряване на производителността може да се постигне чрез обновяване на хардуерната конфигурация – паметта и хард диска. Въпреки че това не е специална модификация за Apache, производителността на сървъра се подобрява. Предоставянето на повече памет за Apache означава, че броят на заявките, изпълнявани едновременно, се увеличава, а бързодействието на системата се подобрява. Обновяването на хард диска подпомага по-бързата поддръжка на входно/изходните операции, обслужването на база данните и дисковите кеш-базирани транзакции. В нашия случай това подобрение не е реализирано и се използва една и съща конфигурационна система за цялото изследване.

Подобряването от страна на клиентския браузър има положителни въздействия както за клиента, така и за сървъра. Клиентите са удовлетворени, осъществява се по-голяма комуникация със сървъра и се увеличава SEO рейтинга. Ползите за сървъра са по-голям капацитет на хардуера, по-лесна мащабируемост и спестени разходи.

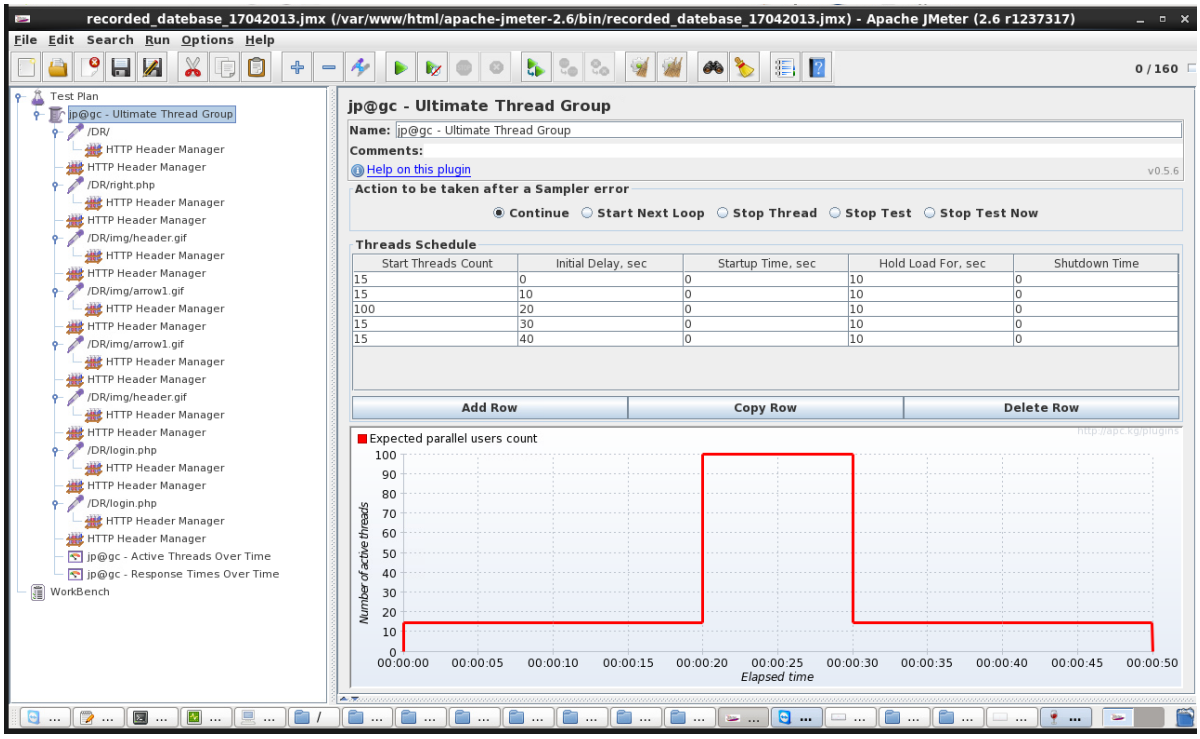
Кеширането от страна на браузера на статични страници намалява времето за натоварване на страниците и намалява броя на заявките, които сървърът ще трябва да обслужи. С това се постига използване на по-малко ресурс за заявките, намаляване големината на извършената работа, намаляване на количеството предавани данни и намаляване на времето за отговор.

В много от случаите при инсталация на Apache, той работи добре, но с течение на времето възниква по-голяма латентост, поради увеличаване на заявките към уеб сървъра. В такива случаи се препоръчва да се ползва професионална помощ от хостинг компанията.

III.3.6. Сравнения на ефективността на реализираните преконфигурационни настройки

За да се представят по-добре и по-ясно резултатите от направените подобрения за системата, се изследва работа с база данни чрез въвеждане на нов сценарий на изпращане

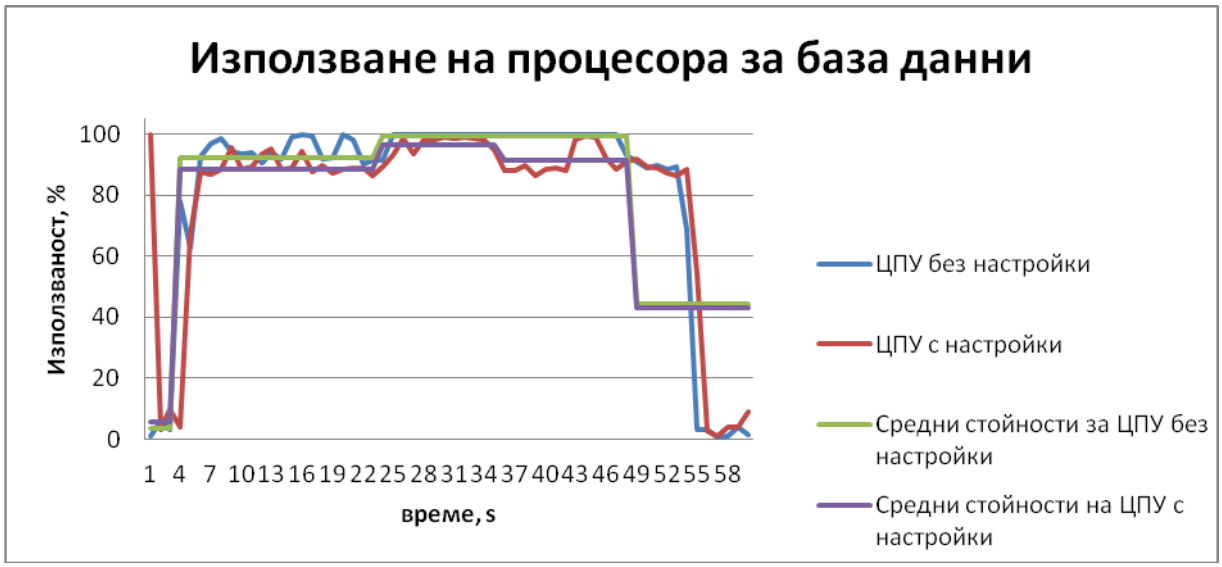
на заявките. Входният сигнал е във вид на импулс, генериран от програмата за симулиране на клиенти, както е видно от фиг. III.22.



Фиг. III.22. Входен сигнал

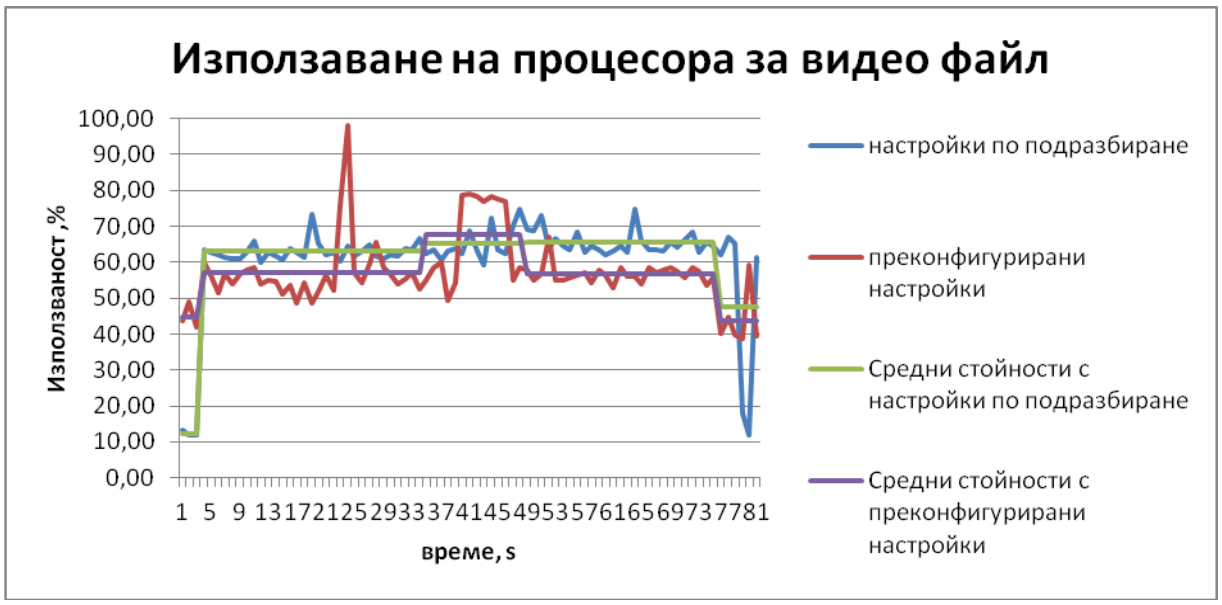
В началото на графиката е показано измереното използване на процесора при нулеви начални условия. Осреднените стойности за използване на процесора, когато настройките на Apache са по подразбиране, са представени със зелена линия. Използваността в този случай е 3,53 %. В случая с преконфигурирани настройки на системата, използваността на процесора е 5,68 %. По този начин се определя началната точка на работа на процесора, при която започва изпращането на заявки. В този момент чрез генератора на входен сигнал се симулира едновременно изпращане на заявки от три клиента към системата. В този период на изпращане на заявки средното използване на процесора е 92,44% при настройки по подразбиране и 88,54% – при преконфигурирани настройки. Изпращането на заявки с такава големина трае 20 секунди, след което се стартира симулиране на изпращане на заявки от пет клиента едновременно. В този период средното използване на процесора при настройки по подразбиране е 99,28%, а с подобрени настройки е 96,39%. Изследван е и трети интервал, в който отново се изпращат заявки от три клиента едновременно. При него използването на процесора с настройките

по подразбиране е 99,50% и 91,24% – с подобрените настройки. Използването на процесора в трите изследвани интервала намалява съответно с 3,90%, 2,89% и 8,26%.



Фиг.III.23. Сравнение на използването на процесора при работа с база данни

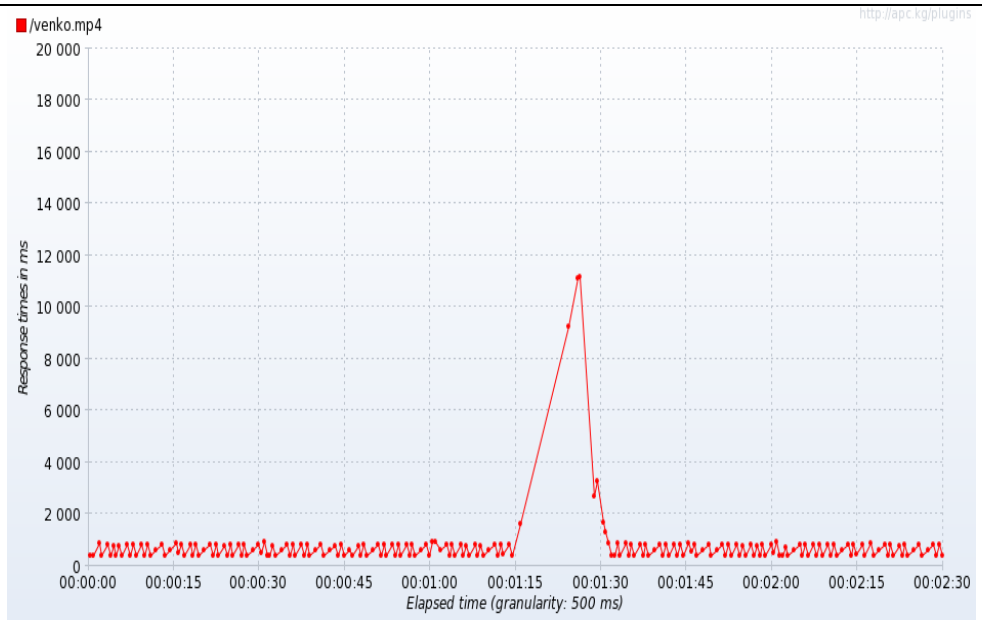
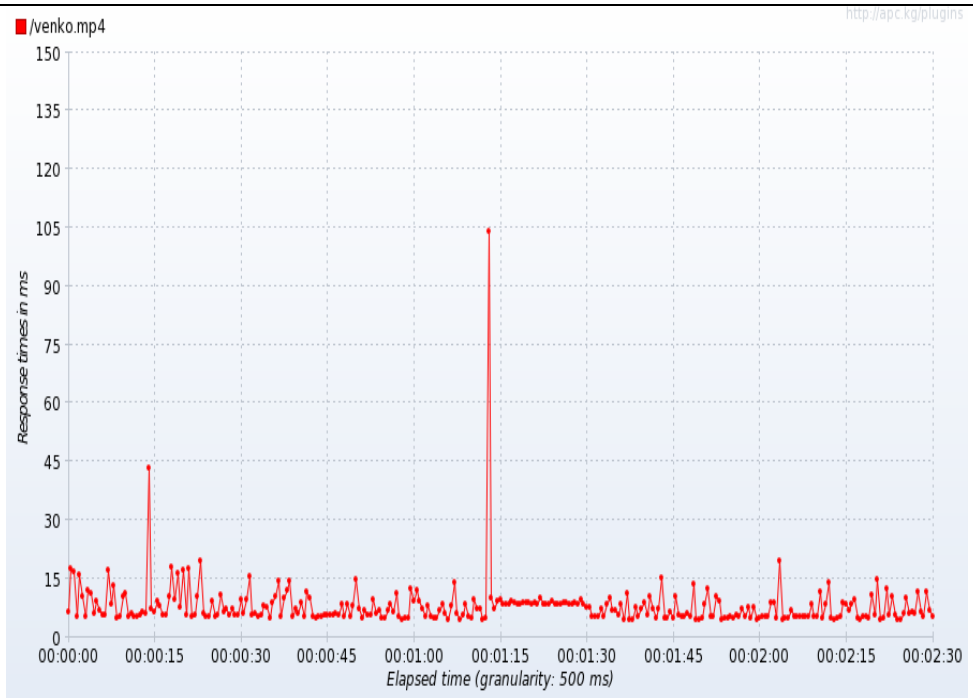
Направените настройки на конфигурационния файл са приложени и за работа с видео файл. Средното използване на процесора при начални условия, когато изследването се извършва с настройки по подразбиране, е 12,35%, а при преконфигурирани настройки е 44,82%. Изпращането на заявки се осъществява по следния сценарий: в начало се изпращат заявки от един клиент, след което се увеличава генерирането на заявки от 10 клиента едновременнно, и отново изпращане на заявки от един клиент. При такъв сценарий на изпращане на заявките става ясно, че средното използване на процесора в първия интервал на изпращане на заявки е намаляло с 38,20%, във втория – с 29,91,%, а при следващото изпращане на заявка от един клиент е намаляло с 41,27%.



Фиг. III.24. Сравнение на използването на процесора при работа с видео файл

Подобренията от приложените настройки на конфигурационния файл на Apache и TCP протокола се потвърждават и от графиките за времето за отговор. То намалява с около 10 пъти при нововъведената конфигурация, което представлява значителен принос на настоящото дисертационно изследване.

В заключение, от двете представени сравнения на резултатите – за база данни и за видео файл става ясно, че преконфигурирането на TCP протокола и на конфигурационния файл на Apache водят до подобрение в използването на процесора и до намалява времето за отговор на системата. По този начин се доказва, че предложените в дисертацията промени в настройките на уеб сървъра Apache и комуникационния протокол TCP са ефективно решение за създаденото уеб приложение, поддържащо база данни с научни статии и видео файлове от семинари, конференции и обсъждания.

	Времето за отговор
конфигурация по подразбиране	
подобрена конфигурация	

Фиг. III.25. Време за отговор при изследване на видео файл

III.4. Изводи

В тази трета глава на дисертационния труд бяха представени хардуерните и софтуерните характеристики на тестваната в експерименталната работа компютърна система. Текстът описа софтуерното приложение, което се използва за изследване на производителността на програмната система. Последната бе представена като система за автоматично управление и бяха изследвани нейната надежност и способността ѝ да се

саморегулира и стабилизира. При оценяване на производителността бе изследвана възможността на програмната система да обслужи различни видове файлове, изисквани от потребителите на софтуерното приложение.

Осъществено бе управление чрез настройка на комуникационния протокол TCP и на програмната система – уеб сървъра Apache. По този начин се подпомогна работоспособността на използваната компютърна система, което стана видно в измерените показатели за производителност. Описана бе и програмата за автоматично настройване на конфигурационния файл на Apache, която се използва за прилагане на подходящи настройки за уеб сървъра в зависимост от броя потребители. Това позволява хардуерните компоненти да се експлоатират по-ефективно.

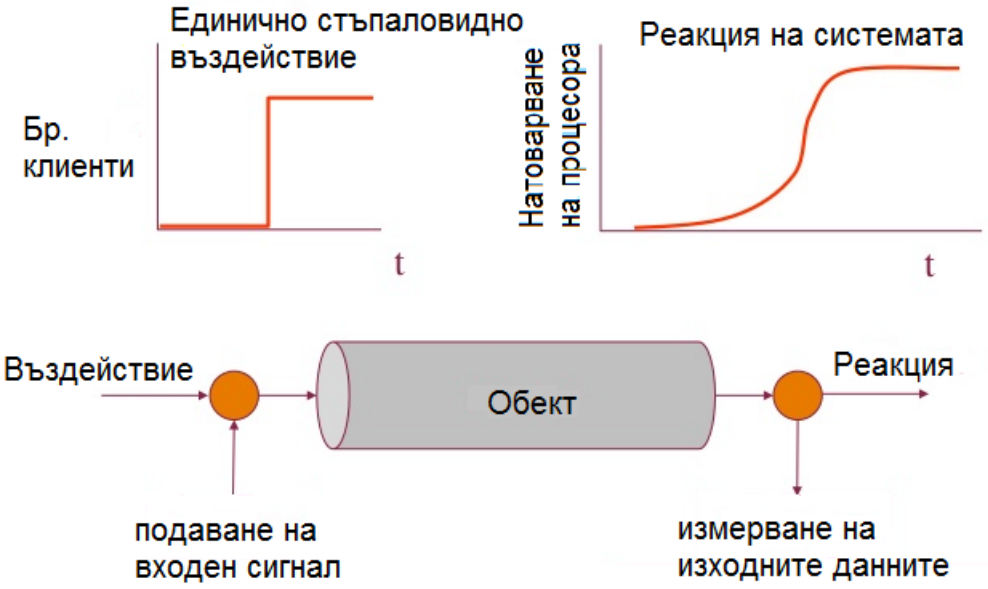
Така глава трета на дисертационния труд демонстрира приложимостта на теория на автоматичното управление с цел повишаване ефективността на програмни системи. Експерименталните резултати показват, че предложените промени в TCP комуникационния протокол и в конфигурационния файл на Apache, водят до подобрение в показателите за качество на производителността – натоварване на процесора, натоварване на паметта и време за отговор.

IV. ПОВИШАВАНЕ НА ПРОИЗВОДИТЕЛНОСТТА НА КОМПЮТЪРНА СИСТЕМА ЧРЕЗ ПРИЛАГАНЕ НА МЕТОДИ ОТ ТЕОРИЯ НА УПРАВЛЕНИЕТО

IV.1. Идентификация на системи

Идентификация на системата е научна област, изследваща изграждането на математически модели на динамични системи чрез наблюдения на входно-изходни данни. Тя обединява действията на реалните приложения, математическата формализация на теория на управлението и моделните абстракции. Идентификация на системата е обширна област, която прилага различни методи за оценяване на системи в зависимост от характера на моделите: линейни, нелинейни, хибрид, непараметричен т.н. В същото време тя се характеризира с малък брой водещи принципи, които ръководят аргументираното описание на правилни решения между сложност на модел, съдържание на информация в данните и ефективно валидиране на данните.

Прилагането на методите за идентификация цели събиране на данни за работата на компютърната система. Анализът на получените данни дава възможност да се определят зависимости между измерваните изходни данни. По този начин се постига по-точна информация за работата на компютърната система и се осигурява възможност за реализиране на допълнително управление, което да подобри нейната работа. За да се постигне това, компютърната система се представя като система за автоматично управление, която има управляващо устройство и обект за управление. При изследване на обекта за управление, т.е. програмната система, основните задачи, които се решават могат да се обобщят като идентифициране и моделиране на техните характеристики, и реализиране на управление. Един от начините за идентифициране на обекта за управление е чрез подаване на единично стъпаловидно входно въздействие към него при нулеви начални условия. Реакцията на обекта е неговата преходна характеристика (функция). На фиг.IV.1 е показан графично процесът на идентифициране на обект при подаване на единично входно въздействие.



Фиг.IV.1 Процес на идентифициране

Получаването на преходната характеристика се осъществява чрез провеждане на експеримент. Целта е събиране на недостъпна и неявна информация за обекта, и въз основа на статистически методи да се извърши моделирането му. Всеки експеримент се състои от опити и наблюдения. Опитът е събиране на данни за m входни фактора и n изходни параметъра. Всеки опит се изгражда от едно или няколко наблюдения. При снемане на преходни характеристики експериментално, т.е. при реализиране на наблюдение, данните много често са зашумени от влиянието на неконтролируеми въздействия. Това налага осъществяването на експериментално снемане на данните няколко пъти (4-6). За получаване на преходните характеристики се изисква входното въздействие да се подава през дискретни интервали от време [Цочев-96], [Вучко-90].

Трябва да се отбележи, че провеждането на експеримент за изследване на система за автоматично управление е сложно действие, което е свързано с редица проблеми като например възможностите на техническите средства за измервания (в нашия случай програмите за измервания), параметрите, разходи, труд, време и др. При някои изследвания провеждането на голям брой експерименти е икономически неефективно и затова се използват други техники за осредняване на данните [БожаВ-73].

Моделиране на изследваната система

При моделиране на система за автоматично управление чрез експериментални изследвания за получаване на техните преходни характеристики се използват таблици за записване на получените данни. Целта на тяхното прилагане е нагледност на резултатите и улесняване на последващата обработка на данните.

В таблица IV.1 са показани първите 15 измерени данни за натоварването на процесора (Cpu%$_{1-4}$), изчислените преходни характеристики на натоварването на процесора (Y_{1-4}) и осредената преходната характеристика при подаване на единично въздействие от 10 потребителя.

Получаването на експериментални преходни характеристики се осъществява чрез определянето на нулеви начални условия y(0)=0, т.е. момента, в който се подава единично стъпаловидно въздействие. Изискването, което трябва да се спази, е обектът да е в установено състояние. В случая на експериментите за целите на настоящия дисертационен труд началният момент е различен от нула, защото непрекъснато протичат процеси, които генерират, макар и минимално, натоварване на процесора. За да може началният момент да започне от 0 се използва следната зависимост [Гарип-07]:

$$y_{ij}^{0} = y_{ij} - y_{j0} \ , \qquad \text{i=0, 1,…,(N-1) j=1,2..n.} \tag{3}$$

Така преходната функция се премества в началото на координатната система и започва от 0. Резултатите от прилагането на тези действия са показани в колони $Y_{1-4.}$ на таблица IV.1. В последната колона се изчислява средната стойност от четирите експериментално проведени опита.

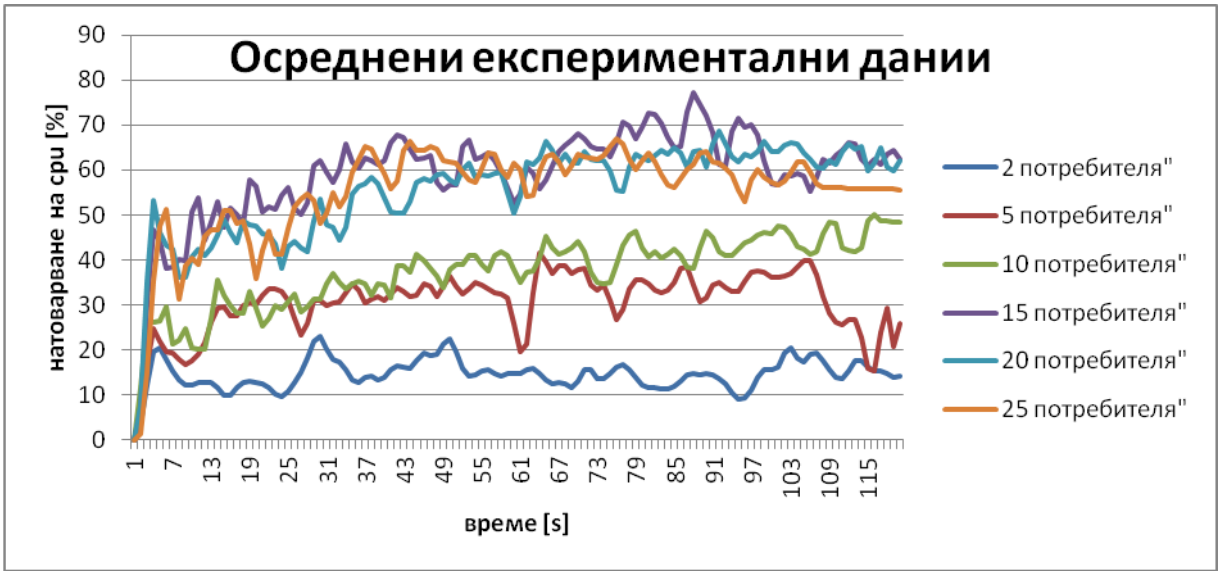
Този начин на обработване на експерименталните данни се използва за получаване на осреднени преходни характеристики за натоварването на процесора и за бързодействието (получените данни от системата) при обслужването на различен брой клиенти (2, 5, 10,15, 20, 25).

Табл.IV.1 Получаване на осреднена експериментална преходна характеристика

	Измерени данни				При нулеви начални условия				
T,[s]	Cpu%$_1$	Cpu%$_2$	Cpu%$_3$	Cpu%$_4$	Y_1	Y_2	Y_3	Y_4	$\bar{Y}$
1	19,6420	56,6896	62,4040	67,9263	0,0000	0,0000	0,0000	0,0000	0,0000
2	21,8174	59,9757	61,0201	67,6415	2,1754	5,0404	1,6464	1,2349	2,5243
3	27,2572	56,1521	61,0108	67,7781	7,6152	14,6565	5,9758	6,8977	8,7863
4	33,4232	48,7413	62,8116	67,7385	13,7812	24,8439	8,1972	12,5882	14,8526
5	38,8170	43,8372	63,5706	66,5161	19,1750	32,9478	9,7298	15,5006	19,3383
6	43,8703	42,8724	61,2725	62,6580	24,2283	36,2339	13,6123	19,3758	23,3625
7	49,0946	43,6702	57,4372	57,4163	29,4526	32,4102	21,0318	27,9603	27,7137
8	54,6025	44,2807	55,6166	55,6947	34,9605	24,9995	29,9253	37,4323	31,8294
9	59,9135	44,6167	58,2741	60,0223	40,2715	20,0954	35,9234	42,2437	34,6335
10	62,7727	45,7113	64,4806	66,5499	43,1307	19,1306	36,9471	42,7508	35,4898
11	61,8238	47,0324	70,0408	69,6327	42,1818	19,9284	35,5633	42,4660	35,0348
12	59,2710	47,4592	71,4321	68,3844	39,6290	20,5389	35,5539	42,6026	34,5811
13	56,8640	48,4573	69,3436	66,4653	37,2220	20,8748	37,3547	42,5630	34,5036
14	53,4603	52,3111	67,1831	66,7838	33,8184	21,9694	38,1138	41,3406	33,8105
15	48,6432	58,0175	66,7838	68,1800	29,0012	23,2906	35,8156	37,4825	31,3975

Изчислените осреднени преходни характеристика са графично представени на фиг. IV.2 и фиг.IV.4. На фиг.IV.2 по оста x са представени данните от измервванията във времето с момент на дискретизация 1 секунда, а по оста y са представени данните от натоварването на процесора в [%]. На фиг.IV.4. по оста x са представени данните от измерванията във времето с момент на дискретизация 1 секунда, а по оста y – данните за бързодействието на системата в [bytes].

От показаните на фиг.IV.2 осреднени преходни характеристики се вижда, че те се разделят на две групи – когато изследваната системата обслужва по-малко или повече от 15 потребителя. За да се изведат обобщени преходни характеристики е необходимо получените данни да имат близки стойности. Въз основа на това изискване се изгражда модел, който ще определи зависимостта между стойностите по оста *x* и тези по оста *y*. Полученият модел за натоварването на процесора на изследваната в този дисертационен труд система се използва за прилагане на допълнителни настройки, чрез които да се постигне по-добро обслужване на потребителите на системата. Натоварването на система в голяма степен зависи от броя потребители на системата. Затова разделянето на модела на две нива е подходящо решение. От друга страна, честата промяната на настройките на системата изисква допълнителни изчисления и хардуерен ресурс, а по този начин се ограничават изчисленията и се намалява използвания ресурс за настройка.



Фиг. IV.2 Осреднени експериментални данни за натоварването на процесора

За описание на получените експериментални данни се използват математически модели. Те представляват приблизително описание на дадено явление чрез математически уравнения, които се решават чрез числова апроксимация. Целта на апроксимацията е да се опрости изследването на различни числови характеристики и качествени свойства на обектите до модели, чиито характеристики и свойства са вече познати или по-удобни за работа. Моделът се използва, за да се определи алгебричният полином, който най-добре описва редица известни данни, нанесени по осите x и y.

Поради факта, че експериментално получените данни са с изразена нелинейност се прилага полиномна зависимост. Тя позволява по-бързи изчисления, по-лесно апроксимиране на нелинейната зависимост и предоставя по-голяма достоверност от диференчните уравнения на предавателната функция.

Посоката на развитието на данните се нарича тренд или тенденция. За данните, представени на фиг. IV.2, тенденцията се описва най-добре с алгебричен полином от трета степен. Полиномният регресионен модел представя връзката между независимата променлива x и зависимата променлива y, моделирана като полином от n-ти ред и определя нелинейна зависимост между тях. Уравнението на полином от n-ти ред е:

$$y = a_0 + a_1 x + a_2 x^2 + a_3 x^3 \ldots\ldots a_m x^m \, , m < n \qquad (4)$$

За определяне коефициентите на полинома от n-ти ред е необходимо да се изгради система от линейни уравнения.

Тези уравнения могат да се представят в матрична форма:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ : \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^m \\ 1 & x_2 & x_2^2 & \dots & x_2^m \\ 1 & x_3 & x_3^2 & \dots & x_3^m \\ : & : & : & \dots & : \\ 1 & x_n & x_n^2 & \dots & x_n^m \end{bmatrix} + \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ : \\ a_m \end{bmatrix} \quad (5)$$

където *y* е векторът на измерените данни по оста y.

Изгражда се матрица с данните по оста *x*, където в нейните колони се записват съответните стойности на степенните редове на данните по оста *x*. В първата колона се записват стойностите за нулевия ред, а в следващите се записват данните за по-високите степенни редове. *m* определя реда на полинома, който търсим. ε е векторът на остатъците (разликите) между избраните и измерените данни. a е векторът на полиномните коефициенти на уравнението.

За получаване на коефициентите на регресорите, векторите и матриците се преобразуват в следния вид:

$$\vec{a} = (X^T X)^{-1} X^T \vec{y} \quad (6)$$

За намиране на вектора за полиномните регресорни коефициенти се използва МНМК (методът на най-малките квадрати). Той е най-често използваният метод за получаване на алгебричния полином на модел. Идеята на този метод се състои в намирането на коефициентите на полиномните регресори на модела така, че сумата от квадратите от отклоненията на всички точки до най-подходящата крива на модела да е минимална.

Надеждността на модела се определя от коефициент на детерминираност R^2. Колкото по-близка е получената стойност до 1, толкова по-добър е моделът, т.е. в по-голяма степен изчислените стойности за него съответстват на експериментално измерените данни.

Коефициентът на детерминация се изчислява чрез следната формула:

$$r_{xy}^2 = \left[\frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}} \right]^2 \quad (7)$$

Получените осреднени преходни характеристики в хода на експерименталните изследвания са представени на фиг.IV.2. Те са подходящи за определяне на модел с параметрите на системата за управление на две нива – когато потребителите са по-малко или повече от 15. Изборът на нивата се основава на близостта на представените криви. Изчисляват се модели за повече и по-малко от 15 потребителя.

Редът на полинома, който апроксимира данните, се определя от броя на флуктуациите в данните или от минимумите и максимумите на кривата. Моделите за натоварването на процесора се описват с алгебричен полином от трета степен. Той описва модела с висок коефициент детерминация и е по-лесно реализуем от моделите от по-висока степен. Резултатите от изчисленията с експерименталните данни са представени графично на фиг.IV.3.

На фигурата е представена и осъществената апроксимация на експерименталните данни с полином от трета степен за по-малък брой потребители:

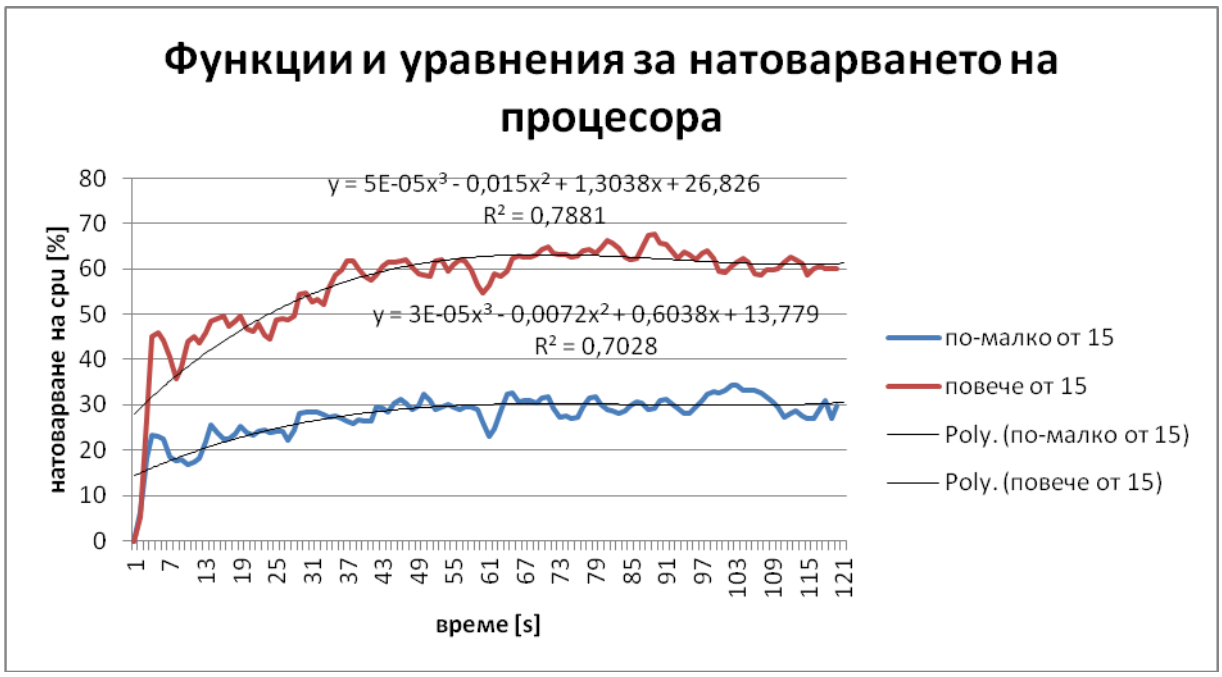
$$y=0,0003x^3-0,0072x^2+0,6038x+13,779 \quad (8)$$

Коефициентът на детерминация на получения модел е $R^2=0,7028$. Това означава, че 70,28% от данните за натоварването на процесора се изменят във времето според установения полином от трети ред при по-малко от 15 потребителя на разработеното приложение.

Апроксимация на експерименталните данни с полином от трета степен за повече от 15 потребителя се описва с:

$$y=0,00005x^3-0,015x^2+0,6038x+26,826 \quad (9)$$

Коефициентът на детерминация е $R^2=0.7881$. Това означава, че 78,81% от данните за натоварването на процесора се изменят във времето според установения полином от трети ред при повече от 15 потребителя на разработеното приложение.

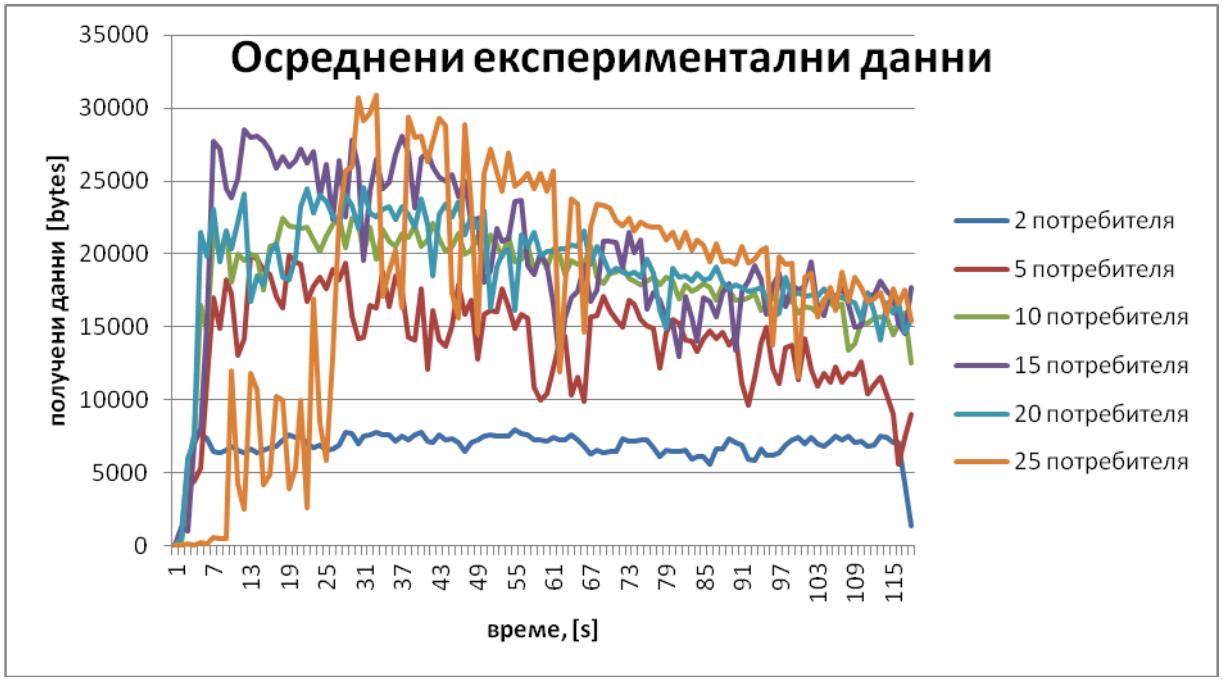


Фиг.IV.3. Функции и уравнения на натоварването на процесора

Системата бе изследвана допълнително чрез анализиране на данните от втори параметър, за да се установи зависимостта на количеството получени и изпратени дани от системата, и броят потребители на системата. Целта е да се изгради модел на неговото поведение в зависимост от броя потребители и да се получи по-ясна представа за бързодействието ѝ. Измерванията за параметъра се получават чрез модула на Bytes throughput over time на програмата Jmeter. Той показва количеството данни, получени и изпратени от Jmeter при генериране на входни въздействия към системата. Модулът измерва данни за получаване и изпращане, тъй като те са с преходни характеристики от един и същи вид. Затова анализът в настоящия дисертационен труд е ограничен само до получените данни, т.е. до бързодействието на обработването им. На фиг. IV.4 са показани осреднените експериментални данни за бързодействието на системата при подаване на единично входно въздействие чрез различен брой потребители на изследваната уеб страница.

Осреднените преходните характеристики за този параметър се наблюдават с по-голяма колебливост и това явление най-ясно проличава при експериментите с 25 потребителя. Представените експериментални преходни характеристики за бързодействието на системата имат по-близки стойности, когато потребителите са повече от 15 и се отдалечават от стойностите за по-малък брой потребители. При определянето на обобщена преходна характеристика е необходимо получените експериментални данни да

имат близки стойности. Въз основа на резултатите за получените осреднени експериментални данни от програмата Jmeter и на горепосоченото условие, отново се изгражда модел на две нива.



Фиг. IV.4. Осреднени експериментални данни за бързодействието на системата

Моделите на бързодействието на системата се описват с алгебричен полином от четвърти ред, поради по-големия броя на флуктуациите в данните. Полиномите на моделите за бързодействието на системата са показани на фиг. IV.5. Полиномът, който описва модела за получените данни за повече от 15 потребителя е:

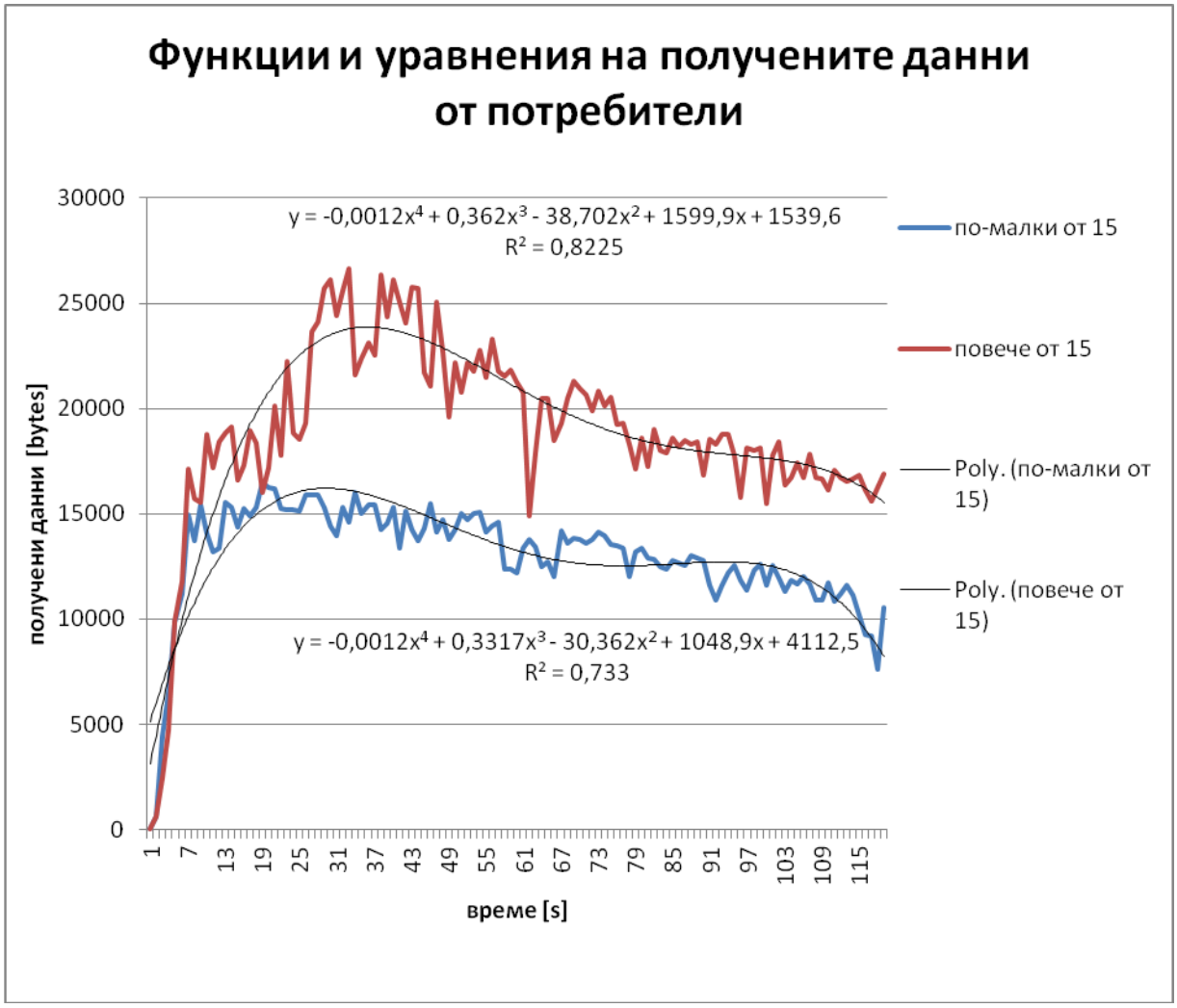
$$y = -0{,}0012x^4 + 0{,}362x^3 - 38{,}702x^2 + 1599{,}9x + 1539{,}6 \quad (10)$$

Коефициентът на детерминация е $R^2 = 0{,}8225$, което означава, че 82,25% от данните за бързодействието на системата се изменят във времето според установения полином от четвърти ред при повече от 15 потребителя на разработеното приложение.

Полиномът, който описва модела за получените за по-малко от 15 потребителя е:

$$y = -0{,}0012x^4 + 0{,}3317x^3 - 30{,}362x^2 + 1048{,}9x + 4112{,}5 \quad (11)$$

Коефициентът на детерминация е $R^2 = 0,733$, което означава, че 73,3% от данните за бързодействието на системата се изменят във времето според установения полином от четвърти ред за по-малко от 15 потребителя на разработеното приложение.



Фиг.IV.5. Функции и уравнения на бързодействието на системата

След провеждането на подобен тип експерименти е необходима допълнителна обработка на данните с оглед по-ясното представяне на значимите данни и елиминиране на ненужните [Гарип-07].

За тази цел се използват методи за допълнително изглаждане на преходната характеристика. При снемането на експериментални преходни характеристики често се срещат интензивни случаи на смущения или икономически неизгодно провеждане на голям брой експерименти. В такива случаи се използват методи за допълнително изглаждане на преходната характеристика. Най-често се използва Метод на пълзящото осредняване, представен по-долу [Гарип-07].

Методът представлява линеен дискретен филтър от l-ти ред, който филтрира влошаването на характеристиката. Изглаждането на всяка точка h*(k) от преходната характеристика се получава от средно аритметичната стойност на l съседни стойности. Прието е l=2, когато измерените точки N<=30, а при N>30 l=N/10. При l=2 се осредняват три стойности.

$$h_{i+\frac{l}{2}} = \frac{1}{l+1}\sum_{j=0}^{l} h_{i+j} \qquad (12)$$

Недостатък на метода е, че липсват l/2 точки в началото и в края на изгладената крива. Поради неговата простота и лесната му употреба, той е използван в настоящия дисертационен труд за допълнително изглаждане на преходните функции.

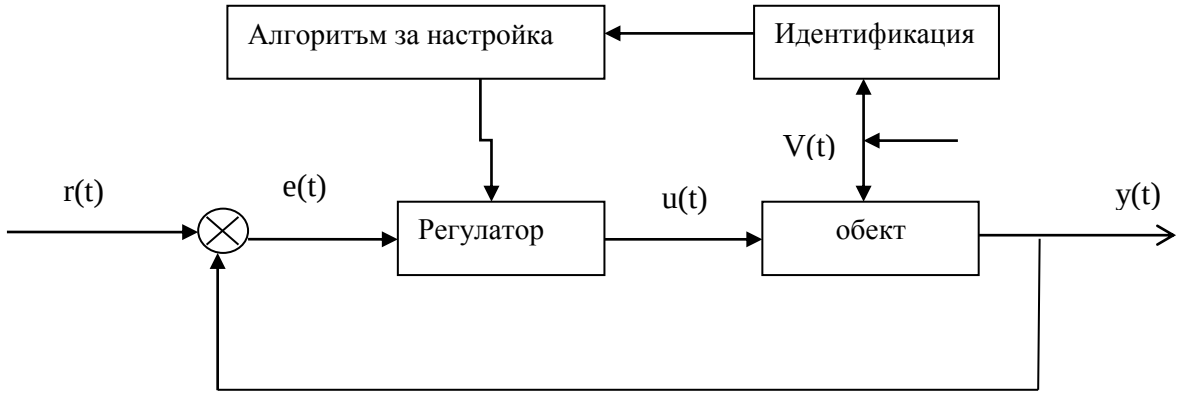
В литературата са описани и други методи за допълнителна обработка на резултатите, получени по експериментален начин като метод на четвъртите разлики, метод на експоненциалното изглаждане и интерполационни методи [Гарип-07].

В настоящия дисертационен труд е изграден модел на обект, представляващ компютърна система. Измерват се показателите на натоварване на процесора и бързодействие при подаване на единично входно въздействие във вид на брой потребители на уеб сайт. Установено е от резултатите, представени на фиг. IV.2 и фиг.IV.4, че моделите трябва да се изградят на две нива – когато броят на потребителите е по-голям или по-малък от 15. Реакцията на изследвания обект се променя при увеличаване броя на потребителите на софтуерната система. За описанието на обекти, които си изменят преходните характеристики при промяна на входните въздействия, се използват системи с твърда адаптация.

IV.2. Системи с твърда адаптация

При системите с твърда адаптация изменението на параметрите на регулатора се осъществява въз основа на модела на обекта. Чрез измерване на работните натоварвания на променливите на обекта на процеса или на въздействащите смущения се установява промяната на модела. При системите с твърда адаптация настройката на параметрите на регулатора се осъществява по метода на компенсацията, при промяна на входния сигнал в компенсатора (който е изграден от блоковете за идентификация и алгоритъма за настройка). В системата за адаптация се съдържа още и блок за идентификация, който е предварително определен чрез offline идентификация. Моделът на обекта зависи от външна променлива v(t) и от средствата за нейното измерване. Полученият модел се

използва от алгоритъм за настройка, който е заложен за оптимална модификация на параметрите на регулатора. В зависимост от стойностите на входния сигнал v(t), адаптивният компенсатор може да се реализира като табличен автомат. В него предварително са поставени стойности на параметрите на регулатора [Велев-94].



фиг. IV.6. Блок схема на система с твърда адаптация [Велев-94]

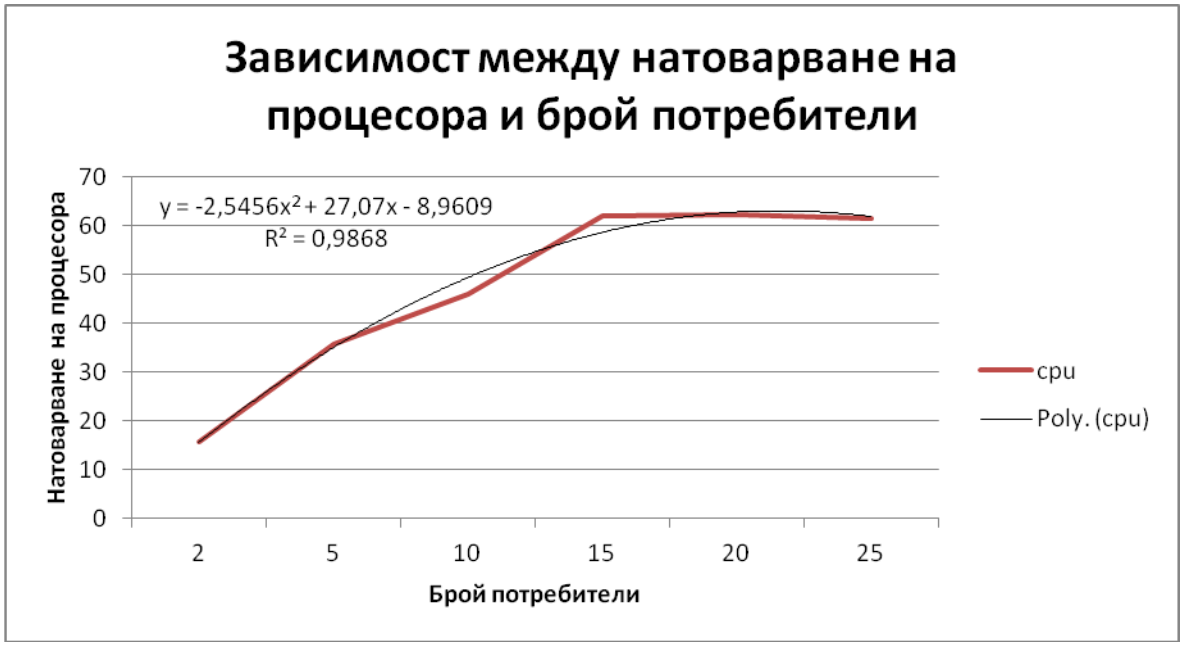
В технологичните системи е прието променливата v(t) да определя условията на функциониране на обекта. Това може да бъде заданието на системата, натоварването на обекта, съставът на преработваната суровина и др. При тяхното изменение се променят коефициентът на обекта, времеконстантите на обекта и чистото закъснение.

Предимство на изграждане на системи с твърда адаптация, използващи табличен автомат, е че се постигат прости и груби решения. Реализирането на чести измервания на функционалната променлива позволява да се постигне по-бърза пренастройка на параметрите на регулатора, намаляване на времето за адаптация и повишаване на качеството на процесите. Изграждането на такъв вид системи има също и недостатъци, като например необходимостта от задълбочено предварително изследване на обекта с цел изграждане на модел. Друг недостатък е невъзможността за измерване на всички функционални величини, които могат да окажат влияние на обекта. Затова е прието системите с твърда адаптация да се използват със системи за самонастройка [Велев-94].

Системите с твърда адаптация могат много добре да се приложат и за компютърни системи, тъй като при изменение на броя потребители се променят и характеристиките на изследвания обект. В нашия случай това е уеб сървъра Apache. Идентификацията се осъществява чрез идентифициране на натоварването на процесора, а настройките на регулатора се реализират чрез промяна на конфигурационния файл на Apache. Използваната операционна система Linux и уеб сървъра Apache имат много добре

направени настройки за саморегулиране по подразбиране. По този начин може много бързо системата да реагира на променението натоварване на процесора и на бързодействието на системата.

За да се приложи система с твърда адаптация при управление на изследваната в дисертационния труд компютърна система се изчисляват средни аритметични стойности за натоварването на процесора при определен брой потребители. Чрез регресионен анализ се получава квардатична функция за натоварването на процесора в зависимост от броя на потребителите. Тя позволява да се адаптират подходящи настройки на уеб сървъра. Получената функция е представена графично на фиг.IV.7. Тя има коефициент на детерминация R^2=0,9868. Последният показва, че получената функция е с високо ниво на достоверност.



фиг.IV.7. Регресионната зависимост между броя потребители и натоварването на процесора

IV.3. Качество на процеса

За използването на системи с твърда адаптация е необходимо да се изчислят показателите за качество на измерваните параметри с цел определяне на състоянията, при които предложеното управление е приложимо.

Качеството по своята същност е съвкупност от свойства и белези на даден продукт, процес или услуга, които определят съответствието им с предварително зададени изисквания или задоволяват определени потребности. За решаване на проблеми на

качеството се използват мощни средства като маркетинга и управлението на производствени процеси. Тяхната ефективност нараства изключително много, когато се комбинират със статистически методи за управление и контрол. Измененията на характеристиките на даден продукт или процес се влияят от много фактори. Някои фактори влияят по-силно от останалите. Основна задача на управлението на качеството е да се отстранят най-силно влияещите негативни фактори. Обработката на факторите в процеса на управление на качеството най-добре се решава чрез моделиране на процеса с методи и инструменти на математическата статистика. Статистическите методи са сложни и изискват добра математическа подготовка, но Съюзът на учение и инженерите в Япония е утвърдил седем прости и нагледни метода за анализ на процесите, които се използват масово. Това са:

- *Контролните листове* са лесни форми за събиране на данни с цел уточняване честотата на различни ефекти.
- *Стратификация* е начин за откриване на някаква особеност. Целта е да се сортират данните съгласно определен критерий или променлива и да се визуализират резултатите. Използва се за класифициране на масив от данни в различни групи с общ характер. Особеност на метода е изборът на критерий за сортиране.
- *Контролни карти* – целта на техното използване е да проследят и изобразят промените на даден показател на качеството при задаване на горна и долна граница на наблюдавания. Въвеждането на граници е с цел откриване на аномалиите на процеса.
- *Диаграма на разсейване* – изобразява отношението между причината и следствието. При диаграмите на разсейване измерванията се показват чрез точки, като мярката за качество се нанася на едната ос, а параметрите, влияещи на качеството – на другата. Използват се за откриване на непознати зависимости или за потвърждение на предполагаеми зависимости.
- *Хистограмите* са средство за изучаване на разсейването на показателите за качеството при разделяне на получените данни в интервали, които се наричат толерансни граници.
- *Парето диаграми* – използват се, за да се визуализира приносът на различните производствени дефекти към общия брой дефекти, като се разработват логаритмични модели. Анализът на Парето се базира на факта, че само малко на брой от многото възможни фактори имат основен принос за наблюдаваното въздействие (например 20% от всички видове дефекти водят до 80% от разходите по дефектите).

- *Диаграми на причините и следствията* – представят зависимостите между дадено действие и неговите възможни причини. Понеже възможните причини може да са много, те се декомпозират на няколко нива.

От изброените по-горе методи в дисертацията се използва методът на контролните карти, понеже е предназначен за наблюдаване на текущите стойности на качествените показатели на даден продукт. Той позволява бързо вземане на решение с цел коригиране и избягване на отклонения от желаните стойности. При контролните карти се използва правилото на трите сигми, т.е. стойности на показателя на качеството x имат нормален закон за разпределение и попадат в интервала $m_x \pm 3\sigma$ с вероятност 0,9973, където m_x е математическото очакване на показателя, а σ стандартното му отклонение.

Тук приемаме, че показателите за нашата изследвана система са натоварването на процесора и постъпилите данни от потребителите, т.е. бързодействието на обслужване. Техните данни се изчисляват в интервал ±σ, поради близостта на получените данни и възможността те да бъдат обработвани с по-малка вероятност. Изчислени са стандартни отклонения за двата модела при повече и при по-малко от 15 потребителя. Тези стойности се използват за определяне на граници, в които да се приложат изчислените модели. В таблици IV.2 и IV.3 са представени обработените данни, които се получават от системата при различен брой потребители.

Таблица IV.2 Гранични и средни данни за бързодействието

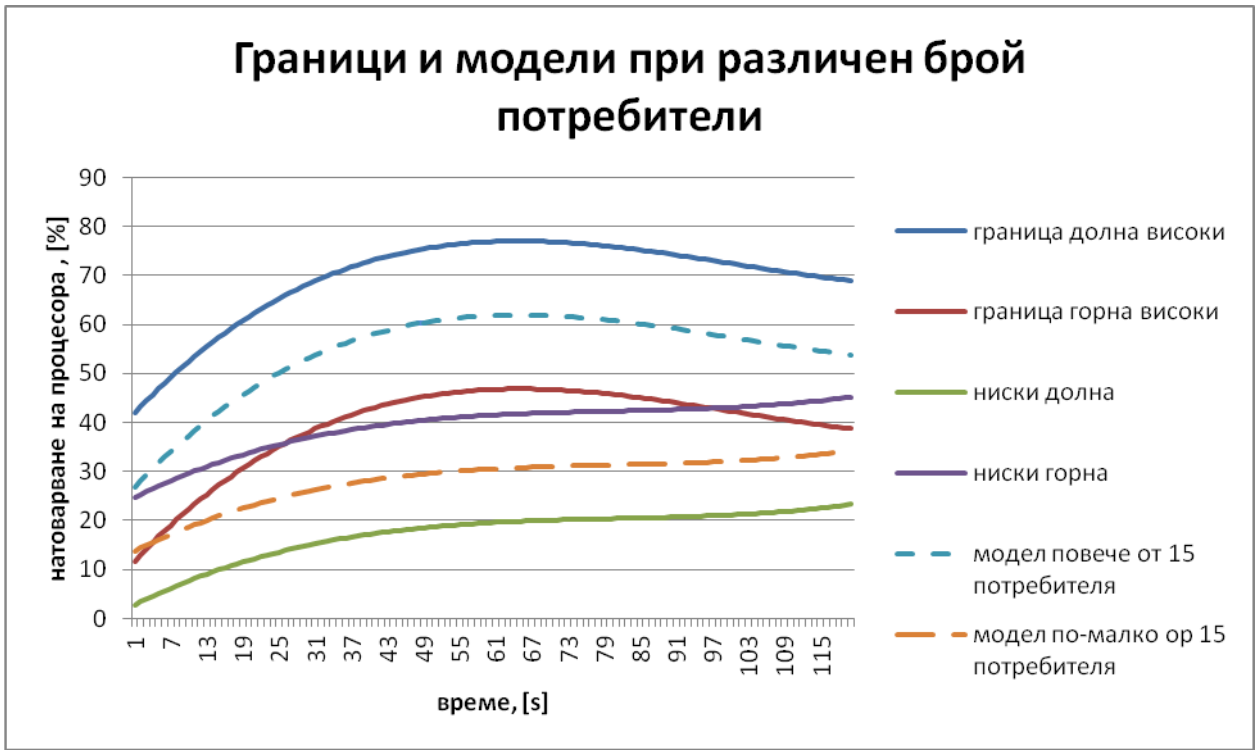
	Количество получени данни			
Брой потребители	Средна стойност	Стандартно отклонение	Долна граница	Горна граница
2	6826,944	1139,203	5687,74	7966,147
5	13680,06	3998,196	9681,866	17678,26
10	17966,29	3869,998	14096,29	21836,29
15	19553,77	5692,959	13860,82	25246,73
20	19017,4	3924,679	22942,08	15092,72
25	17817,67	8012,867	25830,54	9804,804
Средно за по-малко от 15 потребителя	12824,43	3002,466	9821,965	15826,9
Средно за повече от 15 потребителя	18796,28	5876,835	20877,81	16714,75

Таблица IV.3 Гранични и средни данни на натоварването на процесора

Натоварване на процесора				
Брой потребители	Средна стойност	Стандартно отклонение	Долна граница	Горна граница
2	15,80886	7,14315	8,665707	22,95201
5	35,72459	12,70855	23,01603	48,43314
10	45,96397	13,04046	32,92351	59,00443
15	61,89054	17,43184	44,45869	79,32238
20	62,13022	14,58613	47,54409	76,71635
25	61,53487	13,23754	48,29733	74,77241
Средно за по-малко от 15 потребителя	32,49914	10,96405	21,53508	43,46319
Средно за повече от 15 потребителя	61,85187	15,08517	46,7667	76,93705

Функциите, изчислени за моделите за натоварването на процесора и за бързодействието на система, съвместно с изчислените стандартни отклонения на експерименталните данни, позволяват да се определят границите на действията на подобренията на конфигурационните настройки на изследваната тук система.

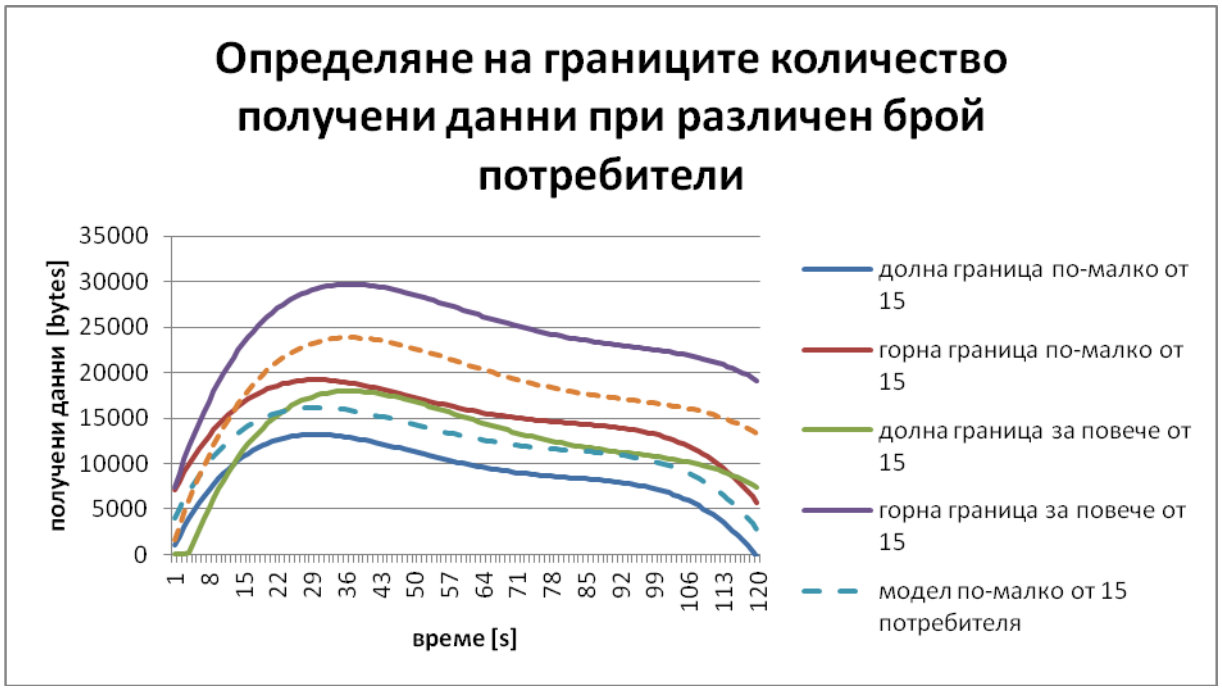
На фиг.IV.8 са показани графично функциите на моделите и границите, в които те могат да работят за показателя на натоварване на процесора.



фиг.IV.8. Функции на моделите и границите на показателя на натоварване на процесора.

От фиг.IV.8 се вижда, че долната граница на модела за повече от 15 потребителя в два момента пресича горната граница на модела за по-малко от 15 потребителя. Освен това се наблюдава един малък интервал от пространството, който не се покрива от нито един от двата модела. За да се отстранят тези недостатъци на моделите, може да се приложат допълнителни изчисления, като например намиране на средната стойност за двете граници.

На фиг.IV.9 са показани графично функциите на моделите и границите, в които те могат да работят за показателя на бързодействие на системата.



фиг.IV.9. Функциите на моделите и границите на бързодействие на системата.

От графиката се вижда, че определената долна граница за повече от 15 потребителя е по-малка от горната граница за по-малък брой потребители. Затова е необходимо или да се намали областта, в която да се проследява количеството постъпили данни от потребителите на системата, или да се използва само горната граница за по-малко от 15 потребители. Трето възможно решение е да се изчисли средната стойност за двете граници.

IV.4. Нелинеен анализ на приложеното управление на системата

За да се анализира реализираното софтуерно управление чрез прилагане принципите за автоматично управление се използва инструмента на MATLAB – System identification tools. Той позволява чрез Matlab функции и симулационни блокове да се изградят математически модели. Той е инструмент, който има широко приложение за идентифицирането на времеви и честотни входно-изходни данни, непрекъснати и дискретни предавателни функции и др. Освен това инструментът за идентификация позволява да се представи динамиката на нелинейни системи чрез изчисляване на Хамерщайн-Винер модели или нелинейни ARX модели с вълновидни мрежи, дървовидни части и сигмоидни мрежови нелинейности. Използването на нелинейни модели за идентификация се прилага, когато линейните модели не представят коректно динамиката на системите. Нелинейните предоставят по-голяма гъвкавост за установяване на сложни явления.

При нелинейните системи не е в сила принципа на суперпозицията, т.е. изходът на системата не е пропорционален на входа. Нелинейните проблеми представляват интерес за много науки, поради факта, че повечето от заобикалящите ни системи са нелинейни по своята природа. Компютърните системи не представляват изключение. Функционалният ефект на управлявания входен сигнал върху измерения изходен сигнал може да бъде източник на нелинейност при компютърните системи. Така например размерът на буфера има нелинейно въздействие върху използването на хардуерните компоненти, заявките в системата и времето за отговор. Пример за нелинейно въздействие при уеб сървъра Apache е управлението на параметъра MaxClients върху времето за отговор. Друг източник на нелинейност в компютърните системи е ограничението на стойностите на показателите на производителността. Големината на заявките може да се представи като линейна функция на използваните компоненти на компютърната система само в случаи, при които големината на пристигащите заявки е равна на обслужваните от системата заявки. Но в момент на увеличаване на броя на постъпващите заявки използването на компонентите не се увеличава, понеже те са ограничени до 100%. Други ограничения на компютърната системата са породени от хардуерната конфигурация. Например твърдият диск има определен капацитет, а входно-изходната големина на данните не може да го надвишава. Също така дължината на опашката не може да надвишава капацитета на буфера, който съхранява заявките.

Поради факта, че изключително голям брой процеси в компютърните системи са нелинейни, използването на нелинейна ARX структура за тяхното моделиране е изключително лесно, защото се определят единствено редовете на полиномите A, B и времезакъснението nk за системата. В нелинейната ARX структура използването на нелинейния оценител с вълновидна мрежа често предоставя задоволителни резултати и се използва като модел за бърза оценка. Редът и закъснението на нелинейните ARX модели са целочислени и положителни стойности, въвеждани чрез параметрите na, nb, nk, където na определя броя изходни стойности за предвиждане на текущата стойност на модела, nb – броя на входните стойности, а nk – закъснението от стойности.

Използването на нелинейните ARX модели цели да се представят модели на подобренията, които са реализирани чрез софтуерните приложения: TCP комуникационния протокол, конфигурационния файл на уеб сървъра Apache и програмата за автоматичното им преконфигуриране, всички работещи с операционната система Linux.

В създадените в хода на дисертационното изследване модели за входни стойности са използвани данните от натоварването на процесора от програмната система, преди да се извършат промените в гореспоменатите приложения, а изходните – след извършените промени. Данните, използвани за изграждане на моделите, са получени при различен брой потребители на разработеното софтуерно приложение за електронно разпространение на научно списание, когато се използва видео файл или база данни.

IV.4.1. Нелинейни модели на потребители на база данните на софтуерното приложение

Нелинейните ARX модели трябва да се разглеждат като разширение на линейните ARX модели на системи с един вход и един изход (SISOARX). При тях входно-изходната зависимост на системата може да се представи със следното уравнение:

$$y(t)+a_1y(t-1)+a_2y(t-2)+...+a_{na}y(t-na)=b_1u(t)+b_2u(t-1)+...+b_{nb}u(t-nb+1)+e(t), \quad (13)$$

където, за да се опрости изразът, броят тактове на времезакъснението е nk=0

Това уравнение представя текущата изходна стойност като претеглена сума на предходните изходни стойности, текущата входна стойност и предходните входни стойности. Уравнението може да се представи и като произведение

$$y_p(t)=[-a_1,-a_2,...,-a_{na},b_1,b_2,..,b_{nb}]*[y(t-1),y(t-2),...,y(t-na),u(t),u(t-1),...,u(t-nb-1)]^T \quad (14)$$

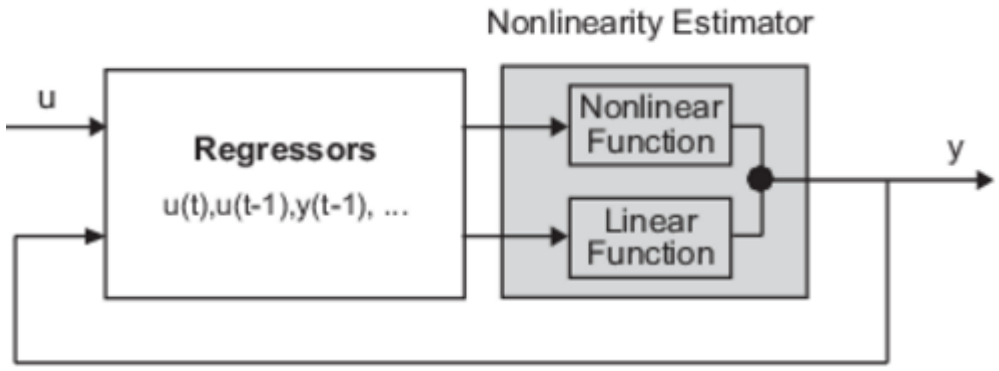
където $y(t-1),y(t-2),...,y(t-na),u(t),u(t-1),...,u(t-nb-1)$ са стойностите на входните и изходни закъснения, които се наричат регресори. Линейния ARX модел предсказва текущата изходна стойност $y_p(t)$ като претеглена сума на нейните регресори. Вместо претеглена сума, която представя линейна функция на зависимостта между данните, може да се използва нелинеен ARX модел, който използва по-гъвкава нелинейна функция:

$$y_p(t)=f(y(t-1),y(t-2),y(t-3),...,u(t),u(t-1),u(t-2),..) \quad (15)$$

където f е нелинейна функция.

Входните данни за f са регресорите на модела. В структурата на нелинейните модели се избира нелинейна функция за нелинейния оценител. При нелинейните ARX модели (NARX модели) регресорите могат да бъдат входните и изходните закъснения или по-сложни техни зависимости.

Структурата на NARX моделите се представя чрез регресори и нелинеен оценител, който има линейна и нелинейна функция.



Фиг. IV.9 Структурата на нелинейните ARX модели

Нелинейните ARX модели изчисляват изходна *y* в две състояния:

- в първото състояние се изчисляват регресорите чрез текущата и предходната входна стойности и предходната изходна стойност. Регресорите са входни и изходни закъснения. По подразбиране всички регресори са входни данни за линейния и нелинейния функционален блок на оценителя на нелинейността.
- във второто състояние нелинейният оценител преобразува регресорите до изходен модел, използвайки комбинация от нелинейни и линейни функции.

Нелинейните оценители могат да са дървовидни мрежи, вълнови мрежи и многослойни невронни мрежи.

Нелинейният оценител използва линейния и нелинейния функционален блок паралелно, за да изчисли f нелинейна функция:

$$F(x)=L^T(x-r)+d+g(Q(x-r)), \quad (16)$$

където x е вектор на регресорите,

$L^T(x)+d$ е изходът от линейния функционален блок и е линейна функция, когато $d\neq 0$,

d е скаларно отместване,

g(Q(x−r) е изходът на нелинейния функционален блок,

r е средната стойност на регресорите x,

Q е проектна матрица, която представя изчисленията.

Точната форма на f(x) зависи от избора на нелинеен оценител. Оценителят на нелинеен ARX модел изчислява стойностите на параметрите на модела L, r, d и Q.

Нелинейните оценители представляват нелинейна функция като сума от последователни нелинейни елементи. Оценяващият алгоритъм във вълновидната мрежа определя броя на използваните елементите автоматично. Той използва следната формула за определяне на функцията g(x):

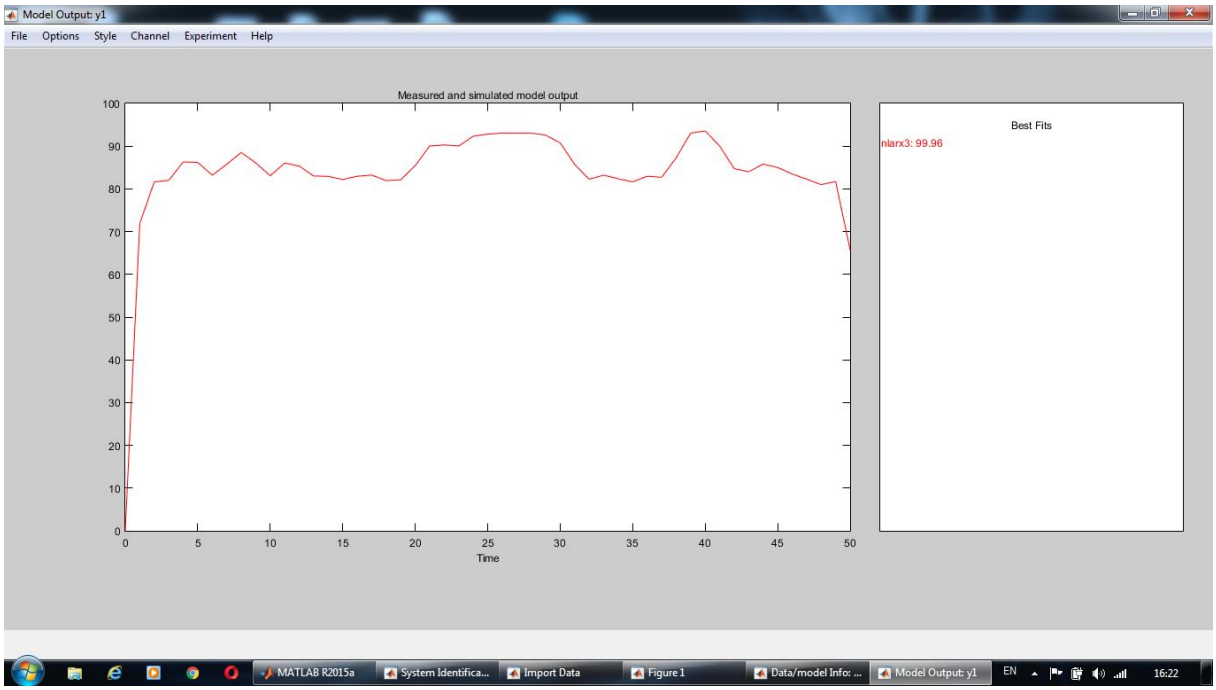
$$g(x) = \sum_{k=1}^{n} \alpha_k k(\beta_k(x - y_k)) \quad (17)$$

където κ(s) е вълновидна функция.

Определянето на регресорите на модела се осъществява чрез задаване на степенния ред на полинома и закъснението на модела, които съответно създават поредицата от стандартни регресори на модела.

За целите на настоящия дисертационен труд първо е изследвано поведението на използваната програмна система с нелинеен модел за три и пет потребители на база данни на разработеното софтуерно приложение за натоварване на процесора при обслужването им. На фиг.IV.10 са представени експерименталните данни и изградения нелинеен ARX модел с вълновидна мрежа за оценяване на нелинейността, след като са приложени настройките към изследваната система. Моделът е изграден чрез инструмента за

идентификация Identification tool на MATLAB. Достоверността на модела е 99,96%. Избраните степенни редовете на полиномите са na=3, nb=3, nk=1.

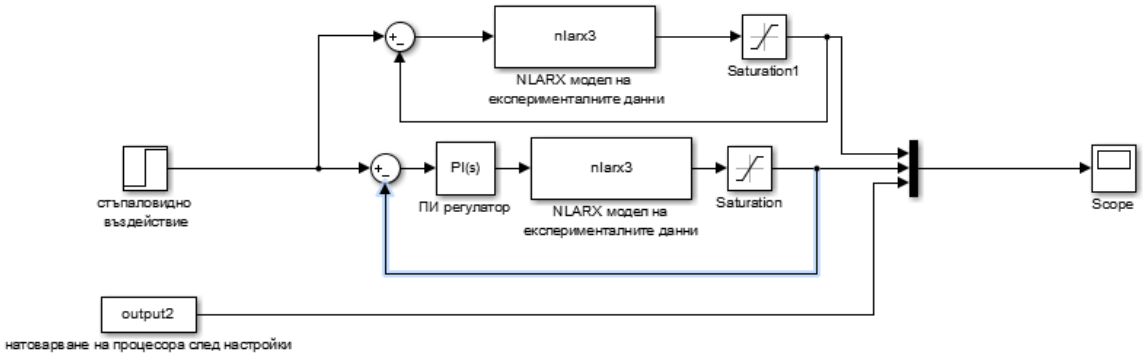


Фиг.IV.10. Нелинеен ARX модел на изследваната система при три и пет потребителя на база данните

Полученият модел е изследван в симулационна среда на MATLAB. На фиг.IV.11 е показана симулационната схема, чрез която се проучват резултатите, получени от приложеното софтуерно управление. Чрез нея се изследва приложимостта на линейни регулатори в използваната програмна система. Регулатор е обобщено понятие за сензор, изпълнителен механизъм и регулиращо устройство. Регулаторите се класифицират според алгоритъма или функционалната зависимост, с която се изработва регулиращото въздействие. В зависимост от вида на алгоритъма, регулаторите се разделят на П–пропорционални, И–интегрални, Д–диференциални и на такива, отразяващи различни комбинации от тези алгоритми. Те се прилагат, когато се изисква постигане на експлоатационни качества на процесите или при решаването на сложни задачи за управление. Задачата, която решава регулаторът, е да се поддържа максимално близко управляваната променлива (натоварването на процесора) до зададената променлива (стъпаловидното въздействие). Това се постига чрез сравняване на двете стойности и се подава на изхода на регулатора манипулираната стойност. В зависимост от желаните изходни стойности, които се изисква да се постигнат след действието на регулатора, се използват различни начин за управление.

За да се симулира управление в изследваната система се използва ПИ регулатор. Пропорционално-интегралният регулатор позволява бърза реакция и компенсиране на смущенията в установен режим.

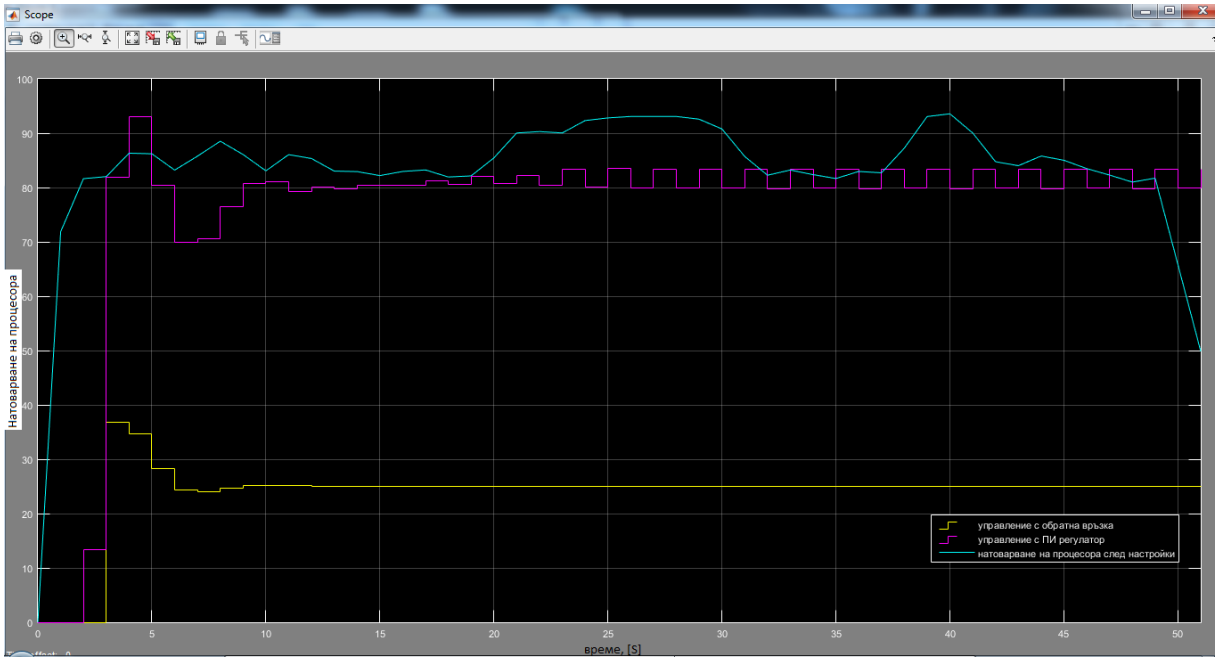
На симулационната схема (фиг.IV.11) (output2) е изходният сигнал от експериментално получените данни от натоварването на процесора след извършените настройки в използваните програми за управление на системата. Те се извличат от текстови файл. Симулирано е управление в затворена система за автоматично управление с пропорционално-интегрален регулатор и е представено негово влияние за програмна система, която е моделирана с нелинеен ARX. Настройката на ПИ регулатора е процес, при който се определят стойностите на П и И компонентите на регулатора за постигане на желаното качество на процеса с цел да се отговори на поставените изисквания. ПИД регулаторите се настройват ръчно или въз основа на определени правила за настройка. Ръчните методи са интерактивни и изискват много време за постигане на желания резултат. Определените правила също имат ограничения – не се поддържат от неустойчиви системи, от системи от по-висок ред или от системи без времезакъснение. Затова се използва PID Tuner – за да се постигне оптимално проектиране на регулатора с оглед на изискванията, когато правилата за настройка и ръчно настройване не могат да се справят добре. Инструментът PID Tuner подпомага интерактивното настройване на регулатора за постигане на желаните настройки. Той позволява лесно да се определят стойностите на компонентите на регулатора за постигане на желаното качество на изходния сигнал.



Фиг.IV.11. Симулационен модел на изследваната система

На фиг.IV.12 са показани получените резултати от симулационния модел. Зелената линия отразява данните, получени след прилагане на подобрените настройки на

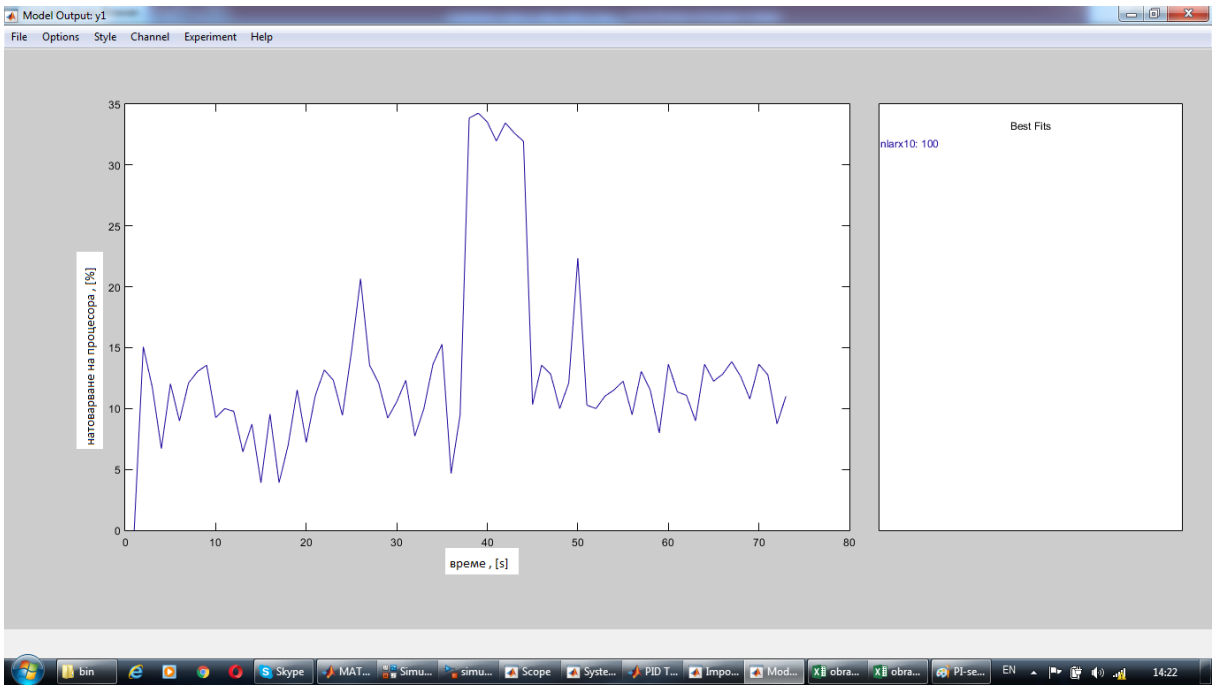
системата, а жълтата линия представя управление само с обратна връзка без регулатор. Лилавата линия показва прилагането на ПИ регулатор в система с обратна връзка с пропорционална съставка П=1,2201 и интегрална съставка И=0.53072 Стойностите на ПИ регулатора са получени чрез инструмента за настройване на регулатори PID tuner. Чрез него експериментално се постига по-бърза реализация на системата и допълнително изглаждане на получения изходен сигнал.



Фиг.IV.12 Получените резултати от симулационния модел при три и пет потребителя на база данните

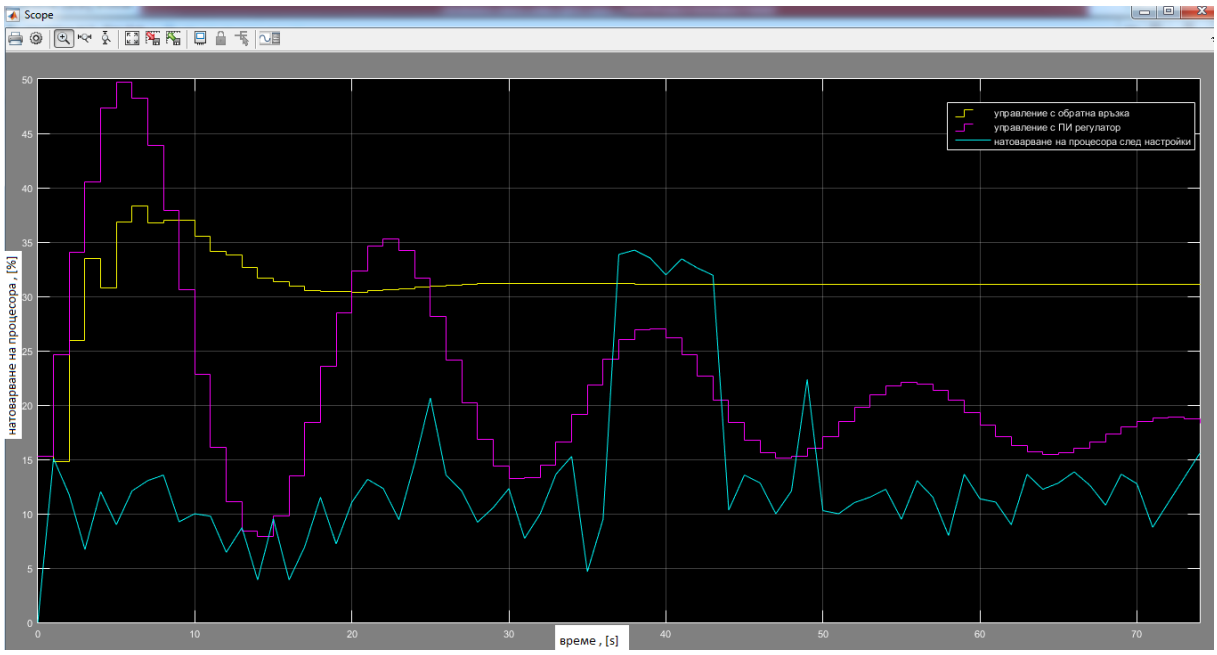
IV.4.2. Нелинейни модели на потребители на видео файл на софтуерното приложение

В хода на дисертационния труд бяха изследвани реализираните подобрения в системата чрез нелинеен ARX модел и при използване на видео файл от потребители на софтуерното приложение. Моделът е изграден чрез инструмента за идентификация Identification tool на MATLAB. Той описва натоварването на процесора от програмната система с достоверност 100%. Избраните степенни редове на полиномите са na=6, nb=4, nk=1. Измерените изходни данни и симулирания нелинеен модел, получен чрез MATLAB, са показани на фиг. IV.15.



фиг.IV.15. Нелинеен arx модел при един и десет потребителя на видео файл

Полученият модел е изследван чрез симулационната схема показана по-горе (на фиг.IV.11). Чрез нея се симулира прилагането на регулатор за натоварването на процесора при обслужване на видео файлове. Резултатите от реализирана симулация са представени на фиг.IV.16. На нея зелената линия показва данните, получени след прилагане на настройките на системата, а жълтата линия представя управление само с обратна връзка без регулатор. Лилавата линия показва прилагането на ПИ регулатор в система с обратна връзка с пропорционална съставка П=-0,04992 и интегрална съставка И=-0.048602. Стойностите на ПИ регулатора са получени чрез инструмента за настройване на регулатори PID tuner. Чрез него експериментално се постига по-бърза настройка на системата и допълнително изглаждане на получения изходен сигнал.



Фиг.IV.16. Получени резултати от симулационнната схема за видео файл при един и десет потребителя

От направените симулационни изследвания може да се изведе заключението, че използването на ПИ регулатор за управление на показателя за производителност - натоварването на процесор, който се моделира с нелинеен ARX модел, ще намали натоварването на процесора при увеличаване на броя потребители на софтуерното приложение.

IV.5.Изводи

В последната четвърта глава на дисертацията бе представено прилагането на методи за идентификация за конкретна компютърна система. По този начин се целеше да се определят динамичните ѝ характеристики чрез показатели на производителността. Получените стойности на характеристиките бяха апроксимирани за изграждането на полиномни модели на системата. Тяхната ефективност се оценява чрез метода на контролните карти, за да се определи приложимостта им. Изчислени бяха и средни стойности за натоварването на процесора в зависимост от броя потребители на софтуерното приложение и бе определена квадратична функция, която позволява да се адаптират подходящи настройки.

Реализираните софтуерни подобрения в изследваната система бяха моделирани чрез нелинейни модели, поради тяхната гъвкавост относно сложни явления и процеси.

Получените модели бяха симулирани в симулационна среда на MATLAB, за да се изследва поведението на системата при включване на пропорционално-интегрален (ПИ) регулатор за управление на натоварването на процесора. Така се демонстрира, че неговото използване ще подобри работа на изследваната система при обслужването на по-голямо брой потребители на софтуерното приложение.

Четвърта глава на дисертационния труд също така демонстрира приложимостта на нелинейни ARX модели, за да представи реализираното софтуерно подобрение и да се разгледа възможността за допълнително управление чрез ПИ регулатори. От експерименталната работа може да се заключи, че при потребление на видео файлове и база данни е подходящо да се реализира ПИ управление за изследваната система.

ЗАКЛЮЧЕНИЕ

Съвременното управление на компютърни и комуникационни системи е изправено пред редица предизвикателства, произтичащи от натоварения мрежови трафик в устройствата, свързващи клиентите с доставчиците на услуги. Често срещани проблеми са забавено време за отговор, намален капацитет за обработване на заявки, неоптимално планиране на задачите и неефективно използване на ресурсите, поради натоварване на хардуерните компоненти – процесор, памет, хард диск. В този контекст статичните системи за управление се оказват все по-неприложими, което налага търсенето на решения, позволяващи гъвкавост и динамика при използването на ресурсите в зависимост от натоварването на компонентите. Така подходи, произтичащи от теория на автоматичното управление, се оказват много полезни и намират все по-голямо приложение при моделиране, анализиране и проектиране на сложни компютърни и комуникационни системи.

Целта на този дисертационен труд бе да демонстрира приложимостта на теория на автоматичното управление по отношение на програмни системи. В хода на изложението бе потвърдена първоначалната хипотеза на този труд за ефикасността на един такъв подход с оглед подобряване на производителността и качеството на обслужване на клиентите. За целта бяха изпълнени следните задачи:

- разработване на функционален модел на компютърната система при различни входни въздействия;
- реализиране на управление в изследваната компютърна система и синтезиране на управление чрез настройка на TCP протокола, уеб сървъра Apache и файловата система.
- прилагане на методи за идентификация на системи и определяне динамичните характеристики на компютърната система при управление на програмата за уеб сървър.
- изследване ефективността на показателите на качество на управлението на системата.
- синтезиране на регулатори за управление на изследваната система и оценяване качествата на отворена и затворена система за управление.

За осъществяването на тези задачи бе проведена работа с конкретен експериментален обект – софтуерно приложение за обслужване на потребители на различни видове файлове в платформата на списание „Автоматика и информатика“.

Установено бе, че регулирането ѝ като сложна система в съответствие с принципите на теория на автоматичното управление води до по-добри резултати с оглед на нейната производителност, изразена в промяна на натоварването на процесора и на времето за отговор.

Тъй като интересът на дисерацията е прилагане на ТАУ в софтуерни приложения, текстът се спря на някои техни свойства и характеристики, като описа компонентите и процесите, протичащи в операционната система Linux и уеб сървъра Apache.

В хода на текста бяха разгледани основни положения от теорията на автоматичното управление, принципи за изграждане на автоматичните системи, както и закономерности, характерни за протичащите в тях процеси. Бе установено, че приложението на управление с обратна връзка е полезно за адаптиране на изчисленията, гарантиране на производителността и планиране на ресурсите. Също така в резултат на литературния преглед бе заключено, че разглеждането на компютърните системи като мрежа от опашки и сървъри е добър начин за изследване на проблемите в тях чрез използване на системи от опашки за моделиране на производителността. Те са удобен инструмент за моделиране поведението на системи в установено състояние, но създават редица затруднения при опити за моделиране на динамиката на компютърните системи.

Това налага прилагането на един характерен за теория на автоматичното управление метод за описание на системите – линейни диференциални уравнения. Те са изключително полезни при изграждането на модели чрез статистически данни или входно-изходни данни. Като подход за проучване на системите те се разглеждат и в теорията за идентификация на системи. В рамките на дисертационния труд разбирането на техните възможности и ограничения бе търсено чрез генериране на входни въздействия към системата и със софтуерни инструменти за измерване, наблюдение и диагностика на компютърната система. Това бе необходима стъпка по посока на една от основните задачи на настоящето изследване – прилагане принципите на теория на автоматичното управление с цел получаване на адекватна информация за работата на програмната система.

В преследване на тази цел, текстът първо се концентрира върху начини за анализ на компютърни системи, които да подпомогнат изграждането на модели и се спря на фактори, които влияят на производителността на системата. Много важна стъпка в тази насока бе да се представи структурата на компютърната система, използвана в хода на експерименталната работа. Тя бе разгледана като сложна система за управление, в която

протичат много процеси едновременно – определени процеси могат да се разглеждат като входни въздействия за системата, други – като смущения за системата, а трети – като изходни данни от нейната работа. Така тя бе представена като система за автоматично управление (САУ), т.е. информационна система, която преобразува съответната информация за постигането на определени цели като регулиране, следене и програмно управление.

За да се определи моделът на изследвания обект (в нашия случай – уеб сървър Apache) се реализират анализ на натоварването и анализ на ресурсите. В експерименталните изследвания, касаещи този дисертационен труд, бе използвана операционна система Linux и инсталиран на нея уеб сървър Apache. Те съвместно осигуряваха обслужването на потребители на изследваното уеб приложение за достъп до научно списание. Управлението на системата бе реализирано чрез преконфигуриране на настройките на Apache и чрез съпътстващо измерване на натоварването на процесите на него.

Във връзка с това в текста бяха описани методологиите за изследване производителността на компютърните системи и за наблюдение на софтуерните и хардуерни компоненти на последните. Бе направен анализ на програмите за генериране на входни въздействия и бе изграден сценарий на тяхното прилагане. В резултат бе изработен функционален модел на използването на софтуерно проложение, който да послужи за целите на експерименталната част на дисертационното изследване.

В хода на експерименталната работа бе установено, че опциите на програмата Jmeter позволяват да се изгради модел на функционалните възможности на системата чрез изследване разработеното приложение и хардуера, който осигурява неговото поддържане. Стана ясно, че така е възможно да се анализира бързо и лесно голямо количество данни от измервания на генерираното натоварване към уеб сървъра и да се изгради математически модел за промените в показателите за производителността на системата по отношение на нейните устойчивост и надеждност. С негова помощ бяха измерени показателите време за отговор, латентност и бързодействие и бе установено, че имат линейна зависимост с броя потребители до достигане в пренатоварено състояние.

Реализирането на управление в изследваната компютърна система имаше две основни цели. Първата бе да се демонстрира, че създаденото софтуерно приложение за достъп до статиите на научно списание предоставя по-модерен, по-лесен и по-функционален начин за достъп и търсене в материалите, разпространявани от Съюза по

автоматика и информатика. В резултат на експериментите бе установено, че разпространението на списанието по електронен път е икономически много по-ефективно (17 пъти по-евтино от начина, използван в момента). Втората цел бе да се докаже, че промяната на параметрите на компютърната система, в това число параметрите на конфигурационния файл на Apache и на комуникационния протокол TCP, ще подпомогне едновременния достъп на повече читатели до системата. Резултатите от експерименталната работа по отношение показателите за качество на производителността компютърната система (натоварване на процесора, натоварване на паметта, време за отговор) демонстрираха ефективността на направените подобрения.

Дисертационният труд също така демонстрира приложимостта на нелинейни ARX модели, за да представи реализираното софтуерно подобрение и да разгледа възможността за допълнително управление чрез ПИ регулатори. От експерименталната работа бе установено, че в случай на потребление на база данни и видео файлове е подходящо да се реализира ПИ управление на използваната система

Приноси на дисертационния труд

1. Разработен е функционален модел чрез прилагане на формалния апарат на теорията на автоматичното управление за компютърна система.

2. Реализирано е управление в изследваната компютърна система като се извършва синтез на параметрите на TCP комуникационния протокол, уеб сървъра Apache и операционната система.

3. Експериментално са определени динамичните характеристики на компютърната система при управление на уеб сървъра Apache и са изведени техните математически зависимостите, описващи нейната работа.

4. На базата на стандартни показатели за качество на компютърната система е изследвана ефективността на приложените настройки за системата.

5. Предложен е подход за настройка на регулатори с пропорционално-интегрално действие – с цел постигане на по-добро качество на управление в затворения контур.

Публикации по темата на дисертационния труд

1	[DimiT-13]	Dimitrov S., T.Stoilov. Loading test of the Apache HTTP Server by video file and usage measurements of the hardware components. Proceedings of the International Conference Computer Systems and Technologies, CompSysTech'13, 28-29 June 2013, Rouse, ACM Conference Proceeding Series, pp.59-66.
2	[Димит-13]	Димитров, Ст., Тест за натоварване на Apache HTTP Server чрез различни видове файлове и измерване използването на хардуерните компоненти, Първа национална тематична школа и борса за научни идеи в областта на информационните и комуникационните технологии, 27-28 юни,2013, Русе, ISSN 1314-9024, стр. 212-217
3	[Димит-13а]	Димитров, Ст., Информационна система за електронен абонамент на научно списание, Международна конференция „Автоматика и информатика" 2013 3-7 Октомври, София, ISSN 1313-1850 2013 стр. I-185 - I-188
4	[Димит-11]	Димитров, Ст., Разглеждане на Apache HTTP Server като система за автоматично регулиране и извършване на тестове с Jmeter , VII Национална студентска научно-техническа конференция 23-27.09.2011г., гр.Созопол, ISSN 1314-0442, 2011,стр. 263-269.

БИБЛИОГРАФИЯ

1	[AbdeB-99]	Abdelzaher T.F., Bhatti N., Web server QoS management by Adaptive Content Delivery Quality of Service, IWQoS '99, Seventh International Workshop on, pp.216 – 225, 31 May -04 Jun. 1999, ISBN:0-7803-5671-3
2	[Agile-13]	AgileLoad, Performance Test Workload Modeling, 12 Jun 2013, http://www.agileload.com/agileload/blog/2013/06/12/performance-test-workload-modeling]
3	[AmEEP-97]	Aman J., C. K. Eilert, D. Emmes, P. Yocom, and D. Dillenberger. Adaptive algorithms for managing a distributed data processing workload. IBM Systems Journal, 36(2), 1997
4	[ApaMP-04]	Apache Modeling Portal, Multitasking server architectures, 29 October 2004,(http://www.fmc-modeling.org/category/projects/Apache/amp/4_3Multitasking_server.html)
5	[ApaSF-16]	Apache Software Foundation, Apache Jmeter – User's Manual, Building a web testplan, 2016, http://jmeter.Apache.org/usermanual/build-web-test-plan
6	[ArliW-97]	Arlitt M. F., Williamson C.L, "Internet Web servers: workload characterization and performance implications", IEEE/ACM Trans. Networking, Vol. 5, No. 5, Oct 1997
7	[ArnoA-94]	Arnold O. A.,Computer Performance Analysis with Mathematica, Academic Press, 1994. *$1.1 Introduction, pg 1.*
8	[ASCRY-03]	Abdelzaher T.F., J.A. Stankovic, Lu Chenyang ; Zhang Ronghua, Lu Ying, Feedback performance control in software services, Control Systems, IEEE Vol.23, Issue 3, pp 16-30, June 2003, ISSN1066-033X
9	[Askub-12]	Askubuntu.com, How do I congifure swapiness?, http://askubuntu.com/questions/103915/how-do-i-configure-swappiness
10	[Avoya-11]	Hovhannes Avoyan, (2011), 25 Apache Performance Tuning Tips, http://www.monitis.com/blog/2011/07/05/25-Apache-performance-tuning-tips/
11	[Balbo-07]	G. Balbo, "Introduction to generalized stochastic petri nets," in Proceedings of the 7th international conference on Formal methods for performance evaluation (SFM'07), M. Bernardo and J. Hillston, Eds. Springer Berlin / Heidelberg, 2007, pp. 83-131.
12	[BalkS-09]	Balkhi S, WPbeginner, What is Apache, 2009 .http://www.wpbeginner.com/glossary/Apache/
13	[BoGMT-05]	G. Bolch, S. Greiner, H. de Meer and K. S. Trivedi, Queueing networks and Markov chains: Modeling and Performance Evaluation with Computer Science Applications, Wiley-Interscience, 2005.
14	[BPHDT-06]	Bouchenak S, De Palma N., Hagimont D., Krakowiak S., Taton Ch.,Autonomic management of internet services: experience with self-

		optimization, Autonomic Computing, 2006. ICAC '06, IEEE, pp.309 – 310, 13-16 June 2006, ISBN 1-4244-0175-5
15	[CaANM-01]	Cao J., Andersson M., Nyberg C., Kihl, M. Web server performance modeling using an M/G/1/K*PS queue, Telecommunications, 2003. ICT 2003. 10th International Conference on Vol.2 23 Feb.-1 March 2003, pp. 1501-1506
16	[CBBLM-02]	E. Chung, L. Benini, A. Bogliolo, Y. Lu, and G. De Micheli. Dynamic power management for nonstationary service requests. IEEE Transactions on Computers, 51(11):1345–1361, 2002.
17	[CherP-99]	Cherkasova L., Phaal P., Session-based admission control: a mechanism for improving performance of commercial Web sites, Quality of Service, 1999, pp.226 – 235 Print ISBN 0-7803-5671-3
18	[CPSCW-95]	Cen Sh., Pu C., Staehli R., Cowan C., Walpole J., Demonstrating the effect of software feedback on a distributed real-time MPEG video audio player. In Proceedings of the Third ACM International Multimedia Conference, November 1995.
19	[CrovB-97]	Crovella M.E., and Bestavros A, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes", IEEE/ACM Trans. Networking, Vol. 5, No. 6, Dec 1997
20	[DeliM-11]	M. Delinger, Fiber-optic transatlantic cable could save milliseconds, millions by speeding data to stock traders, 25.04.2011,http://www.popsci.com/technology/article/2011-04/new-transatlantic-cable-will-speed-information-exchange-price
21	[DGHPT-02]	Diao Y., Gandhi N., Hellerstein J.L., Parekh S., Tilbury D., Using MIMO feedback control to enforce policies for interrelated metrics with application to the Apache Web server. IEEE/IFIP Network Operations and Management, April 2002
22	[DHPGK-06]	Diao Y., Hellerstein J. L., Parekh S., Griffith R., Kaiser G.E., Phung D., A control theory foundation for self-managing computing systems, IEEE Journal on Selected Areas in Communications, vol. 23, Issue 12, Sep 2006, pp 2213-2222, IEEE Press Piscataway, NJ, USA
23	[DiaHP-01]	Diao Y., Hellerstein J. L., Parekh S. S., A Business-Oriented Approach to the Design of Feedback Loops for Performance Management. DSOM 2001 pp.11-22
24	[DiaHP-02]	Diao Y., Hellerstein J.L., Parekh S., Optimizing Quality of Service Using Fuzzy Control, 13th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, DSOM 2002 Montreal, Canada, October 21–23, 2002 Proceedings, pp. 42-53, Online ISBN 978-3-540-36110-7
25	[DiFJR-98]	Dilley J., Friechich R., Jin T. , Rolia J., Web server performance measurement and modeling techniques, Performance Evaluation – Special issue on tools for performance evaluation vol. 33, issue 1, June 1998. pp. 5-26

26	[Frank-99]	G. Franks, Performance Analysis of Distributed Server Systems. PhD Thesis, Report OCIEE-00-01, Carleton University, Ottawa, Ontario, Canada, December 1999.
27	[Freit-05]	P. Freitag, 20 ways to Secure your Apache Configuration (6.12.2005), (http://www.petefreitag.com/item/505.cfm)
28	[Garre-09]	Russ Garrett, (2009), Linux Kernal Tuning, http://russ.garrett.co.uk/2009/01/01/linux-kernel-tuning/
29	[GheGF-13]	Gheorghe G. F., Web Application Performance Testing using Apache JMeter2 March 2013, http://grosan.co.uk/web-application-performance-testing-using-Apache-jmeter-part-2-monitor-server-resources/
30	[GibbR-15]	Gibb R, Visual glossary, What is Load Balancing?, 6 february 2015 https://www.maxcdn.com/one/visual-glossary/load-balancing/
31	[Gregg-13]	Gregg B., Systems Performance: Enterprise and the Cloud 1st Edition,Prentice Hall, 2013, ISBN-10: 0133390098
32	[Guru-15]	Guru99, Introduction to HP LoadRunner and its Architecture, 2015 ,[http://www.guru99.com/introduction-to-hp-loadrunner-and-its-architecture.html]
33	[HeDPT-04]	Hellerstein J., Y. Diao, S. Parekh, D. Tilbury. Feedback Control of Computing Systems, 2004, John Wiley & Sons, Inc., New Jersey, ISBN-13: 978-0471266372, ISBN-10: 047126637X
34	[HePDC-09]	Hewlett-Packard Development Company, L.P., Welcome to the httperf homepage, 2009, http://www.labs.hpe.com/research/linux/httperf/
35	[HoMTG-01a]	Hollot C. V., Misra V., Towsley D., Gong W.-B, A Control Theoretic Analysis of RED, INFOCOM 2001, Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE (Vol.3), 22-26 Apr 2001, ISBN:0-7803-7016-3 , pp 1510-1519
36	[HoMTG-01b]	Hollot, C.V., Misra V., Towsley, D., Gong, W.-B., INFOCOM 2001, On Designing Improved Controller for AMQ Routers Supporting TCP flow Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE (Vol.3), 22-26 Apr 2001, ISBN:0-7803-7016-3 , pp 1726-1734
37	[HTCAG-10]	High Throughput Computing Administration Guide, (2010) https://computing.llnl.gov/linux/slurm/high_throughput.html
38	[HuDoS-98]	Hu J.C, Mungee S, Douglas C. Schmidt D.C, Principles for Developing and Measuring High-performance Web Servers over ATM, 1998, INFOCOM ’98
39	[Kesha-91]	Keshav St., A control-theoretic approach to flow control, ACM SIGCOMM Computer Communication Review Volume 21 Issue 4, Sept. 1991, pp3-15, NY USA
40	[KewN-05]	Kew N., The Apache Platform and Architecture, 2005, http://Apachecon.com/2007/notes/t02-a.pdf
41	[KounB-03]	S. Kounev, and A. Buchmann, “Performance modeling and evaluation of large-scale J2EE applications,” In Proceedings of the Computer

		Measurement Group's International Conference (CMG'03), 2003, pp. 273-283. Налична на https://www.dvs.tu-darmstadt.de/publications/pdf/03-cmg-SPECjAS02_QN.pdf
42	[LawDi-10]	The Law Dictionary, What is LOAD BALANCING? Black's Law Dictionary Free 2nd Ed., http://thelawdictionary.org/load-balancing/
43	[Liqui-16]	Liquid Web, (2016), Apache optimization, http://www.liquidweb.com/kb/Apache-optimization/
44	[LiuHS-05]	X. Liu , J. Heo , L. Sha, "Modeling 3-Tiered Web Applications," in Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '05) , September 27-29, 2005, pp. 307-31
45	[LiuNJ-01]	Liu Zh, Niclausse N., Jalpa-Villanueva C., Traffic model and performance evaluation of Web servers, Journal Performance Evaluation Vol. 46, issue 2-3, October 2001, pp.77-100
46	[LLASS-06]	Lu Ch., Lu Y., Abdelzaher T. F., Stankovic J. A., Son S. H., Feedback control architecture and design methodology for service delay guarantees in web server, 1014-1027, ieee transactions on parallel and distributed systems, vol. 17, no. 9, September 2006
47	[LSDFP-03]	Liu X., Sha L. , Diao Y., Froehlich St., Hellerstein J. L., Parekh S., Online Response Time Optimization of Apache Web Server, 11th International Workshop Berkeley, CA, USA, June 2–4, 2003 Proceedings, IX, pp 461-478, Print ISBN 978-3-540-40281-7
48	[LSWPS-01]	Li K., Shor M. H., Walpole J., Pu C., Steere D.C., Modeling the effect of short-term Rate Variations on TCP-Friendly Congestion Control Behavior, http://www.cc.gatech.edu/projects/infosphere/papers/acc2001b.pdf
49	[Mahba-97]	Mah B.A., An empirical model of HTTP network traffic, INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution., Proceedings IEEE (Volume:2), pp. 592-600, Print ISBN: 0-8186-7780-5
50	[Masco-99]	Mascolo S. Congestion control in high-speed communication networks using the Smith principle, Automatica, Vol. 35, Issue 12, December 1999, pp. 1921–1935
51	[McDoR-07]	McDougall R., Solaris Performance and Tools: DTrace and MDB Techniques for Solaris 10 and OpenSolaris,2007, Sun Microsystems, Inc. http://flylib.com/books/en/2.831.1.12/1/
52	[MDVHR-01]	R. D. Van der Mei, R. Hariharan and P. Reeser," Web Server Performance Modeling, "Telecommunication Systems, Vol. 16, No. 3-4, 2001, pp. 361-378. doi:10.1023/A:1016667027983
53	[MeAFM-00]	Menascéa D.A., Almeidab V.A.F., Fonsecab R., Mendesb M.A., Business-oriented resource management policies for e-commerce servers, Performance Evaluation Volume 42, Issues 2–3, 29 September

		2000, Pages 223–239
54	[MenaA-98]	Menasce D.A., Almeida V.A.F Capacity planning for Web performance: metrics, models, and methods,Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1998, ISBN:0-13-693822-1
55	[Menas-03]	D. A. Menasce, "Web Server Software Architectures," IEEE Internet Computing, vol. 7, no. 6, pp. 78-81, November 2003.
56	[MFBBR-07]	J.D. Meier, C. Farre, Pr. Bansode, S. Barber, D. Rea, Performance Testing Guidance for Web Applications, Microsoft Press Redmond, WA, USA ©2007. ISBN: 9780735625709
57	[MillH-03]	Millsap C., J. Holt, Optimizing Oracle Performance: A Practitioner's Guide to Optimizing Response Time, ISBN-10: 059600527X,2003
58	[Moura-00]	G. Mourani, (2000), Securing and Optimizing Linux RedHat Edition -A Hands on Guide, chapter 6 Linux General Optimzation, http://tldp.org/LDP/solrhe/Securing-Optimizing-Linux-RH-Edition-v1.3/chap6sec75.html
59	[OracC-00]	Oracle Company, Oracle9i Application Server Oracle HTTP Server powered by Apache Performance Guide, 2000, https://docs.oracle.com/cd/A95427_01/httpperf/listener.htm
60	[PasRO-08]	A. Pastsyak, Y. Rebrova, and V. Okulevich, "Performance Prediction of Client-Server Systems By High-Level Abstraction Models," In 4th Software Engineering Conference (Russia) 2008 (SEC(R) 2008) (Oct 23-24 2008), 2008
61	[PatLS-04]	Patwardhan J.P., Lebeck, A.R. , Sorin, D.J., Communication breakdown: analyzing CPU usage in commercial Web workloads, Performance Analysis of Systems and Software, 2004 IEEE International Symposium on – ISPASS, pp. 12-19, ISBN 0-7803-8385-0
62	[PGHTJ-02]	Parekh S., Gandhi N., Hellerstein J.L., Tilbury D., Jayram TS., Bigus J.,Using Control Theory to Achieve Service Level Objectives in Performance Management, Real Time Systems Journal, Vol.23, No. 1-2, 2002.
63	[Ram-05]	VishnuRam, Configuring Apache for maximum performance, https://www.howtoforge.com/configuring_Apache_for_maximum_performance, 2005
64	[RoCAS-02]	Ronghua Zh, Chenyang Lu , Abdelzaher, T.F. , Stankovic, J.A., ControlWare: a middleware architecture for feedback control of software performance, Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on, 2002, pp.301 – 310, Print ISBN 0-7695-1585-1
65	[SilGP-98]	Silberschatz A., P. B. Galvin, Operating System Concepts, Addison Wesley, ISBN 0-201-59113-8, 1998, Fifth Edition
66	[SkaTM-02]	K. Skadron, T. Abdelzaher, and M. Stan. Control-theoretic techniques and thermal rc modeling for accurate and localized dynamic thermal mangemen. In International Symposium on High Performance Computer

		Architecture, Cambridge, MA, February 2002
67	[Sloth-96]	L. P. Slothouber, "A model of web server performance," In Proceedings of the 5th International World Wide Web Conference, 1996.
68	[SmitW-02]	C.U. Smith and L.G. Williams, Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software, Addison-Wesley, 2002.
69	[SoftP-15]	SoftPanorama, Linux TCP Performance Tuning, 20 october 2015, http://www.softpanorama.org/Commercial_linuxes/Performance_tuning/TCP_performance_tuning.shtml
70	[Speed-16]	SpeedGuide, (2016), Linux Broadband Tweaks, http://www.speedguide.net/articles/linux-broadband-tweaks-121
71	[SRAKE-03]	Sanz R., Årzen K., Trend in software and control ,12-15, in IEEE Control System Magazine, June 2003 Arancón J., Polo L., Berrueta D., Lesaffre L., Abajo N., Ontology-based Knowledge Management in the Steel Industry, 243-272, , in The Semantic Web Real-World Applications from Industry, Volume 6, 2008
72	[StanS-03]	M. R. Stan and K. Skadron. Power-aware computing. IEEE Computer, December:35–38, 2003
73	[STILS-01]	System Tuning Info for Linux Server, 19 Novembre 2001 [http://people.redhat.com/alikins/system_tuning.html]
74	[TesPL-13]	Testing Performance Ltd, What is Performance Testing?, 2013, http://www.testingperformance.org/definitions/what-is-performance-testing
75	[Timme-16]	F. Timme, Tuning MySQL Performance with MySQLTuner, 2016, https://www.howtoforge.com/tuning-mysql-performance-with-mysqltuner
76	[TiwaM-06]	N. Tiwari, and P. Mynampati, “Experiences of using LQN and QPN tools for performance modeling of a J2EE Application,” In Proc. of Computer Measurement Group (CMG) Conference, 2006, pp. 537–548.
77	[TricE-11]	Trichkova, E., “PHPbasedsite-templatefordistancelearninginmedicaldisciplines”, ProceedingsofE-Learning'11 conference - E-LearningandtheKnowledgeSociety, 25-26 August 2011, Bucharest, Romania, pp. 87-92, ISBN 978-606-505-459-2
78	[TriMW-10]	M. Tribastone, P. Mayer and M. Wirsing, "Performance prediction of service-oriented systems with layered queueing networks," in Leveraging Applications of Formal Methods, Verification, and Validation, T. Margaria and B. Steffen, Eds. Springer Berlin / Heidelberg, 2010, pp. 51-65
79	[TruvL-08]	Trivino L., Apache Server Architecture, 2008, http://www.researchgate.net/file.PostFileLoader.html?id=54abb76dd11b8b811e8b45b0&assetKey=AS%3A273664036016128%401442257998065
80	[UfimM-06]	A. Ufimtsev, and L. Murphy, “Performance modeling of a JavaEE

		component application using layered queuing networks: revised approach and a case study," In Proceedings of the 2006 Conference on Specification and Verification of Component-Based Systems, 2006, pp. 11-18.
81	[UPSSA-07]	B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi, "Analytic modeling of multitier Internet applications," ACM Trans. Web, vol. 1, no. 1, pp. 1-35, May 2007
82	[WampD-15]	WampDeveloperPro, (2015), Apache Performance Tunning, https://www.devside.net/articles/Apache-performance-tuning
83	[WebTS-16]	Web Technologies Servey, W3Techs, Usage statistics and market share of Apache for websites, 2016 http://w3techs.com/technologies/details/ws-Apache/all/all
84	[Widel-02]	N.Widell, Performance of distributed performance systems, http://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=532482&fileOId=625317
85	[WigeN-13]	Wiger N, Linux Network Tuning for 2013, 6 Apr 2013, http://www.nateware.com/linux-network-tuning-for-2013.html
86	[XuOWM-05]	J. Xu, A. Oufimtsev, M. Woodside, and L. Murphy, "Performance modeling and prediction of enterprise JavaBeans with layered queuing network templates," In Proceedings of the 2005 conference on Specification and verification of component-based systems (SAVCBS '05). Article 5, 2005.
87	[Апу-04]	Апу, А, Администриране и Защита на Apache Server, Primer press 2004
88	[БожаВ-73]	Божанов, Е., И. Вучков, Статистически методи за моделиране и оптимизиране на многофакторни обекти, 1973
89	[Велев-94]	Велев, К., Адаптивни системи, 1994
90	[Вучко-90]	Вучков, Ив., Експериментални изследвания и идентификация, Техника, София, 1990
91	[Гарип-07]	Гарипов, Е., Идентификация на системи: I част (Идентификацияп чрез непрекъснати модели) (трето допълнено и преработено издание).ТУ - София, 2007, 133 стр
92	[ИщевК-07]	Ищев, К., Теория на автоматичното управление, ТУ-София, 2007
93	[СапуГ-07]	(Георги Сапунджиев, Информационни системи в индустрията ISBN 954-438-481-2)
94	[СпаКШ-09]	Гр. Спасов, Н. Каканаков, М. Шопов, Ръководство за лобораторни упражнения по компютърни мрежи, София, 2009, ISBN 978-954-438-790-7
95	[Цочев-96]	Цочев, В., Ръководство за лобароторни упражнения по ИДЕНТИФИКАЦИЯ, 1996