МИНИСТЕРСТВО НА ОБРАЗОВАНИЕТО И НАУКАТА
УНИВЕРСИТЕТ „ПРОФ. Д-Р АСЕН ЗЛАТАРОВ“ – ГР. БУРГАС
ФАКУЛТЕТ ПО ТЕХНИЧЕСКИ НАУКИ
Катедра „Компютърни системи и технологии”

АСЕН ПЕТКОВ ИЛИЕВ

Изследване комуникациите в специализирани системи

АВТОРЕФЕРАТ

На дисертационен труд за присъждане на образователна и научна степен „ДОКТОР“ по научна специалност :
„Компютърни системи и технологии“. Професионално направление 5.3 „Комуникационна и компютърна техника“

Научни ръководители:
проф. д-р инж. Огнян Наков Наков
доц. д-р инж. Станислав Денчев Симеонов

ОКТОМВРИ, 2019 г.
ГР. БУРГАС

Дисертационният труд е обсъден на разширен катерден съвет при катедра „Компютърни системи и технологии”, Университет „Проф. д-р Асен Златаров“ – гр. Бургас, на заседание, състояло се на 15.11.2019г. и е насочен за разкриване на процедура за защита пред жури, определено със заповед……..на Ректора на Университет „Проф. д-р Асен Златаров“ – гр. Бургас.

Дисертационния труд съдържа 172 страници , 2 таблици и 56 фигури. В Библиографията са включени 92 заглавия.

Защитата на дисертационния труд ще се състои на ……от…….часа в зала …,ОК, Университет „Проф. д-р Асен Златаров“ – гр. Бургас

Материалите за защитата са на разположение на интересуващите се в деловодството на Университет „Проф. д-р Асен Златаров“.

Автор: Асен Петков Илиев

Заглавие: Изследване комуникациите в специализирани системи

СЪДЪРЖАНИЕ

ИЗПОЛЗВАНИ СЪКРАЩЕНИЯ

CPU - *Central Processing Unit / Централен Процесор*

IA32/X_86 - *Intel 32-bit Architecture/ 32-битова архитектура на Intel*

IA64 - *Intel 64-bit Architecture/ 64-битова архитектура на Intel*

GCC GNU - *Compiler Collection/ GNU Компилатор*

ICC - *Intel C/C++ Compiler*

RDTSCP - *Read Time-Stamp Counter and Processor ID IA-Асемблерна инструкция*

RTDSC - *Read Time-Stamp Counter and Processor ID IA-Асемблерна инструкция*

GNU - *Общо име на проект за създаване на софтуери с отворен код – със специфичен Лиценз*

HZ - *Честота на системния таймер в Hertz*

VMM - *Virtual Mashine Manager/Мениджър на виртуални машини*

VM -*Virtual Mashine/ виртуална машина*

XEN -*Вид VMM съдържащ планировчици на задачи реално време*

RT/Real Time - *Гарантирано време при обработка на задачите*

EDF - *Earliest Deadline First/ Алгоритъм използван при планиране в -реално време*

BogoMIPS - *Приблизително измерване на производителността на CPU от Linux-ядрото*

LPT - *Largest Processing Time/ Най-дълго процесорно време*

ACPI - *Advanced Configuration and Power Interface*

HPET - *High Precision Event Timer*

TSC - *Time Stamp Counter*

APIC - *Advanced Programmable Interrupt Controler*

kernel/кернел - *Ядро на операционна система*

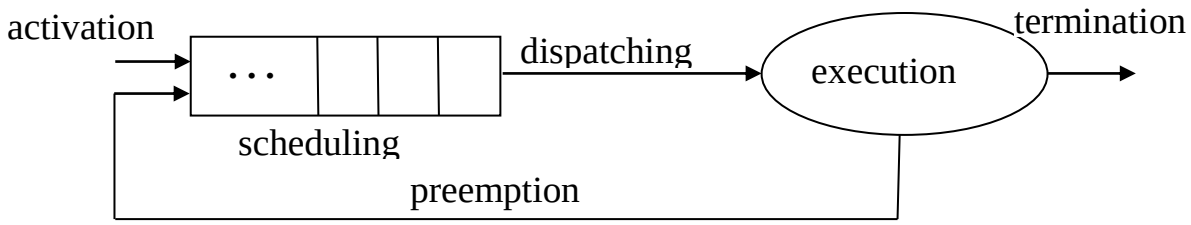
ВЪВЕДЕНИЕ И МОТИВАЦИЯ

През последните години в литературата бяха предложени множество алгоритми и методологии за повишаване възможността за предвиждане на системите от тип реално време. За представяне на резултатите е необходимо да се дефинират базови концепции, които ще бъдат използвани в следващите разглеждания. Ще се започне с базовата единица, срещана в повечето операционни системи – процес. Процесът може да се дефинира като изчисление и/или обработка, изпълнявана от CPU по сериен начин. В текста понятието процес ще се използва като синоним на задача и/или нишка. Много автори предпочитат да дефинират понятието процес като по-комплексно, състоящо се от множество конкурентни задачи или нишки, взаимно споделящи памет и други ресурси. Без да се влиза в подробност за дефиницията на понятието процес, чисто формално, процесът представлява активност, дефинирана на база аргументи, реализираща обработка и представляваща съставен елемент от операционната система.

Ако единичен процесор има за цел да обработва съвкупност от конкурентни задачи, които са задачи, застъпващи се във времето, обработката им от CPU трябва да се осъществи в съответствие с предварително определени критерии, свързани с политика на планирането (scheduling). Съвкупността от правила, които по всяко време определят реда на обработката на задачите, се нарича алгоритъм за планиране (scheduling algorithm). Специфичната задача по отпускане (разпределение) на процесора към задача, определена от алгоритъма за планиране, се определя като разпределяне (диспечеризация).

Задача, която потенциално би могла да е в обработка от CPU, може да бъде в обработка, ако е селектирана от алгоритъма за планиране, или да е в изчакване на CPU. Задача, която би могла да бъде обработвана от процесора, в зависимост от неговата актуална наличност, се нарича актуална задача. Задача, изчакваща освобождаване на процесор за обработката, се нарича

задача в готовност. Всички задачи, изчакващи обработката си от процесора се съхраняват в опашка на готовите задачи. Операционни системи, обработващи различни типове задачи, могат да притежават множество опашки на готовите задачи.



Фигура 1.1 Опашка от готови задачи, изчаквайки обработка

В много операционни системи, в които е реализирано динамично активиране на задачи, задачите за обработка могат да бъдат оценявани на няколко места. По този начин задачи с по-висока степен на важност могат да бъдат обработени непосредствено от процесора, без да се налага да изчакват. В такъв случай обработката на дадена задача може да се прекъсва и да се помества в опашката на готовите за обработка, докато процесорът се предоставя за обработка на готовата по-високо приоритетна задача, текущо променила състоянието си в готовност. Операцията по суспендиране при обработката на дадена задача от процесора, при наличност на друга задача с по висок приоритет носи наименованието присвояване (preemption). Фигура 1.1 илюстрира схематично така дефинираната концепция. В динамичните системи от тип реално време присвояването е важно по следните причини [39]:

- Задача, извършваща обработка на изключение може да измести обработвана в момента задача, защото реакцията на изключението трябва да бъде в рамките на определено време;
- Ако задачата притежава различно ниво на критичност, дефиниращо важността ѝ, присвояването на процесора позволява обработка на критичната задача възможно по-рано;

- Планирането с присвояване обикновено позволява по-висока ефективност, а в случая дава възможност обработката на задачи от тип реално време с по висока utilization (оползотворяване).

От друга страна, присвояването разрушава и нарушава локалността на програмите и вкарва допълнителна обработка (overhead) в реалното време, увеличава времето за обработка на задачата. Като следствие лимитираното присвояване в планировчиците на системите от тип реално време могат да допринесат за подобряване на планирането.

1.1. Контекст

Тъй като понятието реално време е подложено на множество интерпретации, нека първо да се даде дефиниция:.

Дефиницията за Реално Време (RT) е съгласно DIN 443000 и гласи:

„Режим на работа в реално време е режимът на работа на компютърна система, при който програмите за сбор и обработка на спорадично възникващи данни са в постоянна готовност за изпълнение, така че резултатите от обработките да са на разположение в рамките на предварително зададен интервал от време.“

Следователно системата трябва да е в състояние да реагира винаги и при всяко възникнало събитие в рамките на предварително зададен и дефиниран от естеството на обекта за автоматизация времеви интервал, независимо от:

- информационния обем;
- интензивността и динамиката на промените в обекта и
- всякакви други фактори.

Времевият интервал – „време за реакция“ на системата, се определя от момента на възникването на събитието до момента на установяването му от

системата. С други думи, системата трябва да може да приема и отработва информацията своевременно, т.е. докато тя е налична и актуална.

От дефиницията за реално време е видно, че претенциите към времето за реакция от страна на различните типове обекти за автоматизация към системата, са различни. Т.е. варират от 1 милисекунда за Енергетиката, до няколко минути за други типове обекти за автоматизация.

Пример за функции, които се изпълняват при спазването на „твърди критерии“, са тези за сбор и обработка на технологична информация от обекта. Тук информацията винаги има импулсен характер, т.е. тя се задържа за известно време, след което се възвръща в изходното си положение. Ако не се приложат „твърдите критерии“, това ще доведе до загуба на информация, т.е. до отказ на системата.

Съществуват и други критерии за реално време, при които времето за реакция, респективно за отговор, се определя на базата на статистически показатели, т.е. усреднено, то се движи в определени рамки. Такива критерии се прилагат например при функциите за визуализация, протоколиране, архивиране и пр. Тук времето за извеждане на картината спокойно може да се движи между 1 и няколко секунди и това не се отразява върху работоспособността на системата. Същото се отнася и до функциите за архивиране и протоколиране, тъй като при системите данните от обекта пристигат с времето си на възникване.

Основното противоречие при изграждането на системите за автоматизация е това между естеството на обекта за автоматизация и изискването той да се контролира и управлява в режим на реално време.

Проблемът тук се състои в това, че обектът за автоматизация, респ. неговите подобекти/съоръжения, се характеризират с голям обем процесна информация, която постъпва в системата спорадично и се „разпространява“ из нея със силно вариращa интензивност. Положението се усложнява и от недетерминираното поведение на комуникационната среда и комплексната функционалност на системата.

От друга страна, системата и в най-лошия случай (worst case) трябва да реагира адекватно и детерминирано по критериите за RT.

1.2. Научен проблем и хипотеза

Отчитането на времето е много важно за ядрото. Много от функциите в ядрото са в зависимост от времето, за разлика от тези, зависими (задвижване) на база събития. Част от тези функции са периодични като тези за балансиране на кръговите опашки на планировчика или за опресняване на екрана. Това се осъществява на база фиксирано разпределение, като напр. 100 пъти в секунда. Ядрото планира и други функции като закъснения при В/И – операции към/от диск, считано от определено относително време в бъдещето. Ядрото на ОС може да планира напр. 500 милисекунди работа от настоящия момент. Освен това, ядрото на ОС трябва да пази системното време, регистриращо от кога работи машината, управлява и сверява текущите дата и време.

Налична е разлика между относителното и абсолютното време. Планиране на събитие за 5 секунди не изисква концепция за абсолютно време, а за относително (напр. 5 секунди от този момент). Обратно, управлението на текущите ден и време изисква ядрото не само да отчита времето, но и да може абсолютно да го измерва. Тези две концепции са критични за управлението на времето.

Трябва да се прави разлика между събития, които се случват периодично, и такива, които ядрото планира за фиксирано време в бъдещето. Събитие, което се случва периодично, всеки 10 милисекунди се задейства посредством системен таймер. Системният таймер представлява апаратен елемент, генериращ прекъсване с фиксирана честота. Манипулаторът на прекъсването, наречен още таймер на прекъсването, актуализира системното време и извършва периодична работа. Системният таймер и неговия таймер на прекъсване заемат основно място в ядрото на операционната система.

Виртуализацията в съвременните компютърни системи създава условия за по-ефективно използване на ресурсите. При виртуализация отчитане на времето с определена точност е проблем, решаването на който ще даде възможност за правилното моделиране, измерване и оценка на вътрешнопроцесните комуникации. Виртуализацията в режим реално време е нерешен проблем. Макар и да съществуват класически методи за отчитане на събития в операционните системи (сигнализация, прекъсвания, изключения, капани и др.), те по дефиниция не удовлетворяват условията за реално време, особено в режим на виртуализация. Това налага изграждането на нова концепция, свързана с измерване на времето и гарантиране на неговата точност. Моделирането на динамични процеси, без дефиниция за точността на времето създава проблеми при симулацията, а по късно и при реализацията на комплексни системи.

1.3. Цел и задачи

В контекста на посоченото до тук може да се направи заключение, че наличието на високо-производителни процесорни системи налага нов подход за тяхното приложение. Един новаторски подход е виртуализацията. Виртуализацията в конвенционалните системи отдавна е доказала своята работоспособност по отношение на оптималното използване на компютърните ресурси. В системите от тип реално време, виртуализацията допринася много за оптимално натоварване и по този начин за достигане на приемливи стойности на съотношението:

$$C = |R[i]|/P$$

където абсолютната стойност на вектора R[i] представлява броя налични ресурси, а P това е цената на наличната компютърна система.

Наличието на виртуализация в системите от тип реално време налага нов подход при организацията на комуникациите между отделните приложения. Това касае основно нова концепция за измерване на времето и

неговото отчитане. Целта на изследванията, обобщени в рамките на настоящия дисертационен труд, е: На база формално представяне на времето и формализация работата на процесите от тип реално време да се даде нова постановка за измерване на реалното време в компютърната система, предразполагащо коректни планиране и комуникации. Така поставена целта, предполага решаването на следните основни задачи:

- Формализация елементите на операционните системи;
- Формално описание на системите от тип реално време, дефинирайки основни компоненти в тях;
- Описание на елементите при обработка на периодични и апериодични задачи от тип реално време;
- Изследване пълното натоварване на системи от тип реално време и определяне на допустим критерий за работоспособност;
- Практическо изследване на системи, предоставящи виртуализация и паравиртуализация, концентрирайки се върху примери от тип „Системи с отворен код“, в частност XEN;
- Разработка и частична реализация на работно обкръжение (Framework), гарантиращо точност при отчитане на времето;
- Сравнителен анализ със съществуващи системи.

2. ФОРМАЛНО ОПИСАНИЕ НА ОПЕРАЦИОННИ СИСТЕМИ, СИСТЕМИ РЕАЛНО ВРЕМЕ И ОТЧИТАНЕ НА ВРЕМЕТО

2.1. Формално описание на елементите в операционни системи

Развитието на компютърната техника поставя все по-високи изисквания към програмното осигуряване, с цел оптимално приложение. Приложното програмно осигуряване представлява сложен комплекс приложения, осигуряващи поддръжка на различни области. Изискванията към програмното осигуряване са различни с оглед на сферата на тяхното приложение. Много често тясното място на компютърните системи се оказват реализацията на компоненти от системното програмно осигуряване – операционна система, драйвери и др., и взаимодействието между приложените и системните компоненти. Оптимизацията на системните компоненти е фактор, който до голяма степен се явява решаващ за реализация на специализирани приложения. За да се реализира оптимизация на системни компоненти, се съблюдават принципи, които в някои отношения са различни от тези в приложното програмиране. В настоящия материал се представя методика за формално описание на компоненти от операционните системи. Такъв подход би дал възможност за реализация на оптимизация и моделиране на определени ситуации в дадена компютърна система.

Абстрахирайки се от конкретната компютърна архитектура и спецификата на апаратната реализация на съответната компютърна система, съставните елементи на дадена операционна система могат да се представят по следния начин [1,2,3]:

- ресурси – физически и логически;
- активности;
- интерфейси;
- протоколи.

Ресурсите в дадена операционна система са физически и логически. По презумпция на това място се пропускат и виртуални, имайки предвид, че те представляват част от логическите ресурси. Физическите ресурси са всички тези ресурси, които притежават материална репрезентация в компютърната система – физическа памет, процесор, диск, периферия и др. Логическите ресурси са въпрос на логическа дефиниция и са свързан и по-скоро с представяне на компоненти в системата – страница във виртуална памет, сегмент с определена големина, програмен текст и др. От тази гледна точка може да се каже, че програмата от гледна точка на операционната система не представлява нищо повече от определен ресурс с „право на обработка". Правото на обработка се дава експлицитно от определения потребител или администратор на системата.

2.1.1. Приложение на формалното описание

2.1.1.1. Модел на състоянията на процес

Поради факта, че в дадена компютърна система се обработват значително по-голям брой процеси от броя на процесорите е необходимо да се въведат глобални състояния на процесите. В определен момент могат да се обработват точно толкова процеси, колкото е броят на процесорите. Диаграмите на състоянията в различните операционни системи се отличават, но тези различия не са драстични спрямо общия модел, гарантиращ минимум три състояния [3]. На базата на въведената формализация, този модел може да се опише по следния начин:

- активно: Процесът притежава всички заявени от него ресурси, включително процесор. Формално това условие се описва така:

$$\left|\vec{R}_a\right| = \left|\vec{R}_r\right|$$

- готовност: Процесът притежава всички заявени от него ресурси, но без процесор. Формалното описание изглежда така:

$$\left|\vec{R}_a\right| = \left|\vec{R}_r\right| - 1$$

- блокировка: Процесът чака заделянето на заявен(и) ресурс(и). Това състояние формално изглежда така:

$$\left|\vec{R}_a\right| < \left|\vec{R}_r\right| - 1$$

Формалните дефиниции опростяват определено описанието на диаграмата на състоянието. Векторите имат следните значения:

- $\left|\overrightarrow{R_r}\right|$ – представлява заявените от определен процес ресурси. Абсолютната стойност на вектора показва броя заявени ресурси. При обработката на процес е възможно да се заявяват множество ресурси от един и същ тип. От значение за моделирането е броя на заявените ресурси;
- $\left|\overrightarrow{R_a}\right|$ – представлява предоставените на определен процес ресурси. Абсолютната стойност на вектора показва броя предоставени ресурси. При обработката на процес е възможно да се предоставят множество ресурси от един и същ тип. От значение за моделирането е броят на предоставените ресурси. Не се предоставя незаявен ресурс.

2.1.1.2. Формализация на слоен модел на операционна система

На това място, използвайки формалните дефиниции, ще бъде показан интуитивен модел на операционна система, състояща се от m на брой слоя. Множеството S на всички слоеве Si се дефинира:

$$S = \left\{S^i \mid 0 \le i \le m-1\right\}$$

Моделът се ограничава от най-ниския слой S^0, представляващ апаратната платформа и S^{m-1}, последния, най-висок в йерархията, който се определя от създаващия модела (при OSI, m=7) [5,6,7].

Два съседни слоя S^j и S^{j+1} са свързани с две различни релации:

1. По-високият слой S^{j+1} използва услугите на по-ниския S^j:

$$S^{j+1}\nabla_d S^i$$

2. По-ниският S^j слой управлява по-високия S^{j+1}:

$$S^{j+1}\nabla_s S^i$$

използва се в смисъл „извиква”, „използва”

Последователността релации при обслужване $S^{m-1}\nabla_d S^{m-2}, S^{m-2}\nabla_d S^{m-3}, ...S^1\nabla_d S^0$ конституират йерархията на обслужването. Всеки слой S^j притежава ресурси. Множеството ресурси на всеки слой се представя така:

$$R^i = \left\{R_j^i \mid 0 \le j \le n-1\right\}$$

Всеки слой може да обхваща различен брой ресурси (от там параметъра n). Всеки ресурс се консолидира само с определен слой:

$R^i \cap R^j = 0$ за $\left(0 \le i, j \le m-1 \wedge i \ne j\right)$

От множеството на всички налични ресурси в даден слой S^j могат да се определят подмножества, които в определен аспект могат да се изследват. От тях може да се формира съвкупност за управление:

$$C_j^i \subseteq R^i$$

Тази контролна единица, представя определена активност, но тя принадлежи на същия слой, от който са ресурсите. Поради факта, че всяка активност е с ограничено във времето действие (ограничен живот) може да се дефинира като временна съвкупност ресурси (temporary aggregation). Пример: На база програмен код, памет и идентификатор се създава процес. Възможният (теоретично, разбира се) брой активности ще се характеризира посредством всички подмножества на R^i, т.е. мощността на R^i:

$$\left\{C_j^i\right\} = \prod\left(R^i\right)$$

Очевидно е, че не всяко свободно обхващане на ресурси в даден слой може да формира смислена активност. Действителното формиране на

активност на база съвкупност от ресурси от даден слой S^i се поема от функция g, която е локализирана непосредствено в по ниския слой:

$$g^{i-1}:\{R^i\}\Rightarrow C^i \qquad (i\geq 1)$$

След прилагане на така дефинираната функция пасивната съвкупност ресурси се превръща в активност. Пример: системното извикване fork() (всъщност, fork() е corefunction, елемент от основната библиотека, която се занимава с функционалността на системното извикване sys_fork(), но разглеждането на системни извиквания в детайли е извън рамките на настоящия материал [3]).

Не винаги от агрегацията на ресурси от даден слой i е предпоставка за възникване на активност. По точно казано, на базата на агрегация на ресурси, вместо активност, резултатът може да е създаване на нов ресурс. Пример: Въз основа на известно количество блокове физическа памет може да се създаде логическа памет. Това „създаване" на ресурс на по-високо ниво би могло да се дефинира с друга функция f по следния начин:

$$f^i:\{R^i\}\Rightarrow R^{i+1}$$

Функцията f^i създава ресурс на слой i+1 на база на ресурс от слой i. Локализацията на такава функция е в двата слоя.

2.2. Системи реално време

2.2.1. Дефиниция и основни характеристики

Системите от тип реално време се характеризират с реализация на компютърни активности, свързани със стриктни време-ограничения, които трябва да бъдат спазвани и/или постигани за да се достигне до желаните свойства. Типично при обработката на задачата това е крайния срок за частична или пълна обработка (deadline). Това време ограничение представлява времето (в частност време-момента), преди което пълната или частична обработка на задачата трябва бъде завършена. Ако крайният срок за обработка бъде специфициран в съответствие с времето на създаване (активиране) на задачата, той се нарича относителен краен срок (relative deadline). Ако крайният срок се специфицира относно време-момент, приет за нула, тогава се нарича абсолютен краен срок (absolute deadline). В зависимост от последиците при неспазване на това ограничение, по време на обработката на задачата, крайният срок, т.н. реално време, може да се категоризира в три категории:

- Строго (Hard Realtime) – като такъв, крайният срок се дефинира, ако нарушаването му предизвиква катастрофални последици за контролираната система;
- Устойчиво (Firm Riealtime) – като такъв, крайният срок се дефинира, ако нарушаването му не предизвиква щети на системата, но изхода няма стойност;
- Леко (Soft Realtime) – като такъв, крайният срок се дефинира, ако нарушаването му води само до понижаване на производителността на системата.

2.2.2. Формулиране проблема при планиране в реално време

За да се дефинира проблема на планирането, е необходимо да се въведат три нови понятия:

- Множество от n задачи $\Gamma = \{\tau_1, \tau_2, \dots, \tau_n\}$;
- Множество от m процесора $P = \{P_1, P_2, \dots, P_m\}$;
- Множество от s вида ресурси $R = \{R_1, R_2, \dots, R_s\}$.

Възможно е да се специфицира релация по последователност на обработката между задачите на базата на граф за последователността. Време-ограничения могат да се асоциират към всяка от задачите. В този контекст планирането представлява предоставяне на процесор от множеството P и ресурси от множеството R на задачите от множеството Γ по такъв начин, че обработката на задачите напълно да се реализира, спазвайки ограниченията, дефинирани към тях. Дефиниран проблема, в неговата обща форма е доказано вече [11], че е NP – complete, което го прави компютърно труден за обработка.

3. КОНЦЕПЦИЯ И РЕАЛИЗАЦИЯ НА ТАЙМЕРНА СИСТЕМА ПРИ X_86

3.1. Възприемане на времето от ядрото

Разбира се концепцията за времето в компютъра е малко неясна. В действителност ядрото трябва да се сработи с хардуера на системата, за да се разбира и управлява времето. Хардуерът осигурява системен таймер, който се използва от ядрото, за да се прецени протичането на времето. Този системен таймер работи на база електронен източник на време, като например цифров часовник или честотата на процесора. Таймерът за система се задейства с предварително програмирана честота, наричана скорост на появяване на отметка (tick rate). Когато системният таймер изтече, се генерира прекъсване, което дава възможност на ядрото да го обработва чрез специален манипулатор на прекъсване.

3.2. Механизми за сравнителна оценка на времето за изпълнението (обработката) на код при INTEL IA32 и IA64 архитектури

Целта, която се поставя тук, е да се даде поддръжка на разработчиците на програмно осигуряване за прецизни измервания на часовниковите таймерни цикли, изискващи специфичен код на C в среда на Linux, базирана на обща архитектура на Intel – процесори [91]. Методите могат да са много полезни в контекста на измерване на производителността на CPU, при оптимизацията на код и също при фини настройки на операционната система. Във всички тези случаи разработчиците на програмно осигуряване са заинтересувани да познават точно колко таймерни цикли са изминали при изпълнението (обработката) на кода.

3.3. Измерване на времето при виртуализация

Широкото приложение на вградените системи е предпоставка за създаване на голямо разнообразие от сценарии на приложения. Виртуализацията навлиза все по-широко в различни аспекти на приложенията. Като причини за виртуализацията може да се посочат:

- Наличие на голямо количество апаратни ресурси, които не винаги се използват ефективно;
- Капсулиране на отделните виртуални машини, създаващи условия за достигане на по-висока степен на сигурност;
- Пълноценно и ефективно планиране на ресурсите в рамките на компютърната система;
- Едновременно използване на различни платформи и ефективен достъп до специализирани контролери.

До скоро виртуализацията се реализираше преди всичко на системи сървъри. В последно време тя придобива все по-голямо значение и при вградените системи. Реализацията на виртуализирани системи, в които да се запазват условията на реално време, е едно предизвикателство. В това отношение компютърните системи, изградени на база персонални компютри, са една област, която все още е не достатъчно разработена.

Виртуализацията може да се дефинира като методология за разпределяне на ресурсите на персоналния компютър между множество обкръжения, на база различни платформи. Така виртуализацията създава условия за обработка на задачи от тип реално време и едновременно с това обработка на задачи от операционни системи с общо предназначение. На база на апаратните абстракции се създават условия за лесна миграция на програмни системи от една операционна система към друга. Реализацията на виртуални системи на база хипервайзор дава възможност за скалиране и

ефективно разпределение на множество ядра в рамките на персоналния компютър. Виртуализацията създава илюзията за едновременна обработка на задачи, които представляват активности в различни операционни системи.

Паравиртуализацията е един вариант за реализация. Предвиден е междинен програмен слой, който в много случаи носи наименованието монитор на виртуалната система. Хипервайзорът управлява конкурентното използване на ресурси в рамките на виртуалните процесори.

Целта тук е да се създаде тестова установка, която да дава възможност за измерване отклоненията във времето при обработката на хетерогенни задачи от тип реално време, в режим на виртуализация. Правилното отчитане на времето е много актуална задача по отношение на обработката в реално време, в режим на виртуализация. Съвременните реализации са на база изравняване на времето между хостовите и гостовите системи с помощта на Network Time Protocol (NTP). Добавеното време в този случай е трудно предсказуемо, поради особеностите на реализацията на многозадачния режим и контекстното превключване между задачите. Като пример ще се вземе планирането на периодични задачи на база възможно по-прост алгоритъм с фиксирано разпределение на приоритетите и взаимодействието им с апериодични такива. Като апериодична задача се приема полинг сървър, от гледна точка на простотата на неговата имплементация.

3.4. Реализация на ниско-латентен таймер.

Идеята за оптимално измерване на времето не е нова. На това място се предлага нова методика за отчитане на времето при виртуализация на операционни системи от тип реално време, която е реализирана на база на XEN. Изграденият на тази база ниско-латентен таймер гарантира единно време на всички приложения във виртуалните машини. Ефективното използване на този ресурс дава възможност за реализация на по-ефективна обработка на приложенията, за сметка на малкото време за достъп в резултат от намаляването на overhead-а време, необходим за достъп до таймера.

Настоящето решение предоставя възможност за независими апаратни реализации на таймери с необходимата резолюция и тяхното приложение при планирането, разпределението и управлението на приложения от тип реално време. Измерването на време на база тази реализация ще позволи точна синхронизация в системи за измерване, контрол и управление в режим реално време. Цялата разработка се предоставяме под формата на „пач“ (patch), който е приложим не само за тук показаната експериментална постановка, но и за всяка друга система. Стремежът е да се предложи методика за реализация на програмно измерване на времето независимо от наличния хардуер.

3.4.1. Програмна структура на ниско-латентен таймер

За да се гарантират измерванията, се използва опцията на процесора constant_tsc, която гарантира, че което и от ядрата да използва стойността на TSC, то тя ще е една и съща. Забраняват се c_states на процесора, за да може ядрата да работят с максимална производителност.

Измерването на латентността на таймера в брой цикли на процесора се реализира в съответствие с препоръките за измерване на Intel [91].

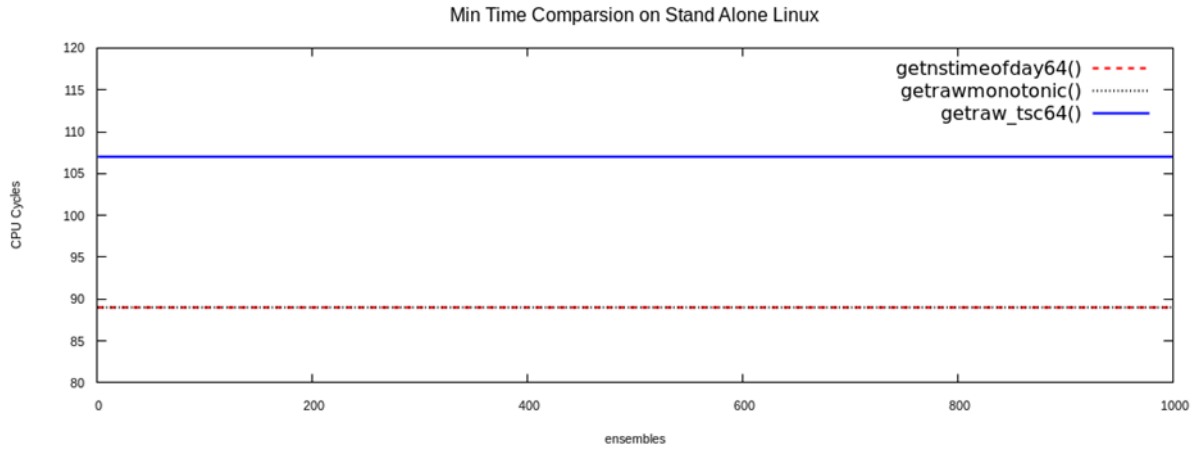
Фактическата програмна реализация е концентрирана във функцията getraw_tsc64(). Тя представлява ниско-латентен таймер, разработен специално за изследване на системи реално време. Предвидена е оптимизация с цел бързина, без възможност да бъде прекъсван. Обработката на функцията се осъществява за възможно най-малкия брой работни цикли на процесора. Самият таймер е реализиран като разширение на функционалността на "POSIX TIMERS" в инфраструктурата на linux операционните системи. Неговото приложение е както в ядрото на операционната система Linux, така и от потребителското пространство. Втората част на функционалността се осъществява посредством функцията clock_gettime(CLOCK_MONOTONIC_TSC,...) от стандартите "POSIX.1-2001,

POSIX.1-2008, SUSv2.". За доказване на работоспособността на програмната разработка е необходимо сравнение между:

- getnstimeofday64 - тази функция, представляваща таймер, е налична в ядрото (текущо се използва ядро Linux-4.14.65). Тя чете и доставя системно време с точност наносекунди, но чете времето от структурата на текущия таймер, който се обновява на всеки тик на диспечера на задачите. Това дава много бързи и стабилни времена но точността им е 1/HZ на ядрото на операционната система и може да се модифицира от NTP или на ръка, което го изключва от възможността да дава реална оценка при изследване в режим реално време.
- getrawmonotonic64 - тази функция, представляваща таймер, налична в ядрото (текущо се използва ядро Linux-4.14.65). Тя чете и доставя системно време с точност наносекунди и предоставя времето чрез директен достъп до хардуерния таймер, но има много голям overhead.
- getraw_tsc64 - разработения в рамките на настоящата работа таймер.

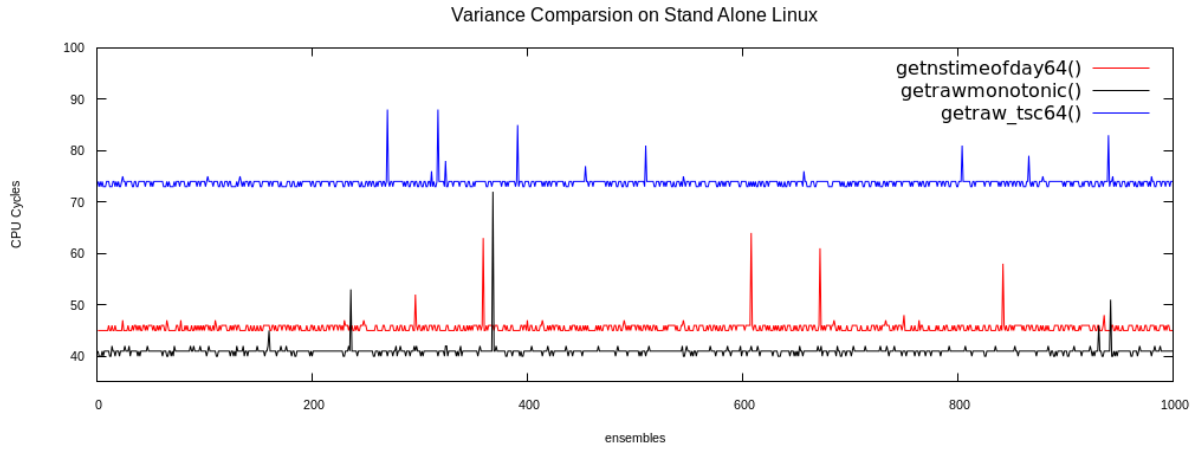
3.4.2. Сравнителен анализ на наличните таймери и новосъздадения

За измерване и сравнителен анализ се използва методиката на фирма Intel [91], модифициран вариант на която е показан в 5.6.6.6. Предвидено е място за изследваната функция. Методиката е обобщена в програмна структура (framework). Тази програмна структура е подходяща за изследване на функции, без значение от предназначението им. Фактът, че програмната структура работи във високо приоритетен режим на ядрото, способства за оптимално измерване. Пълният изходен код на модула е даден в Приложение 1. Резултатите са онагледени на фигури 5.5, 5.6 и 5.7.



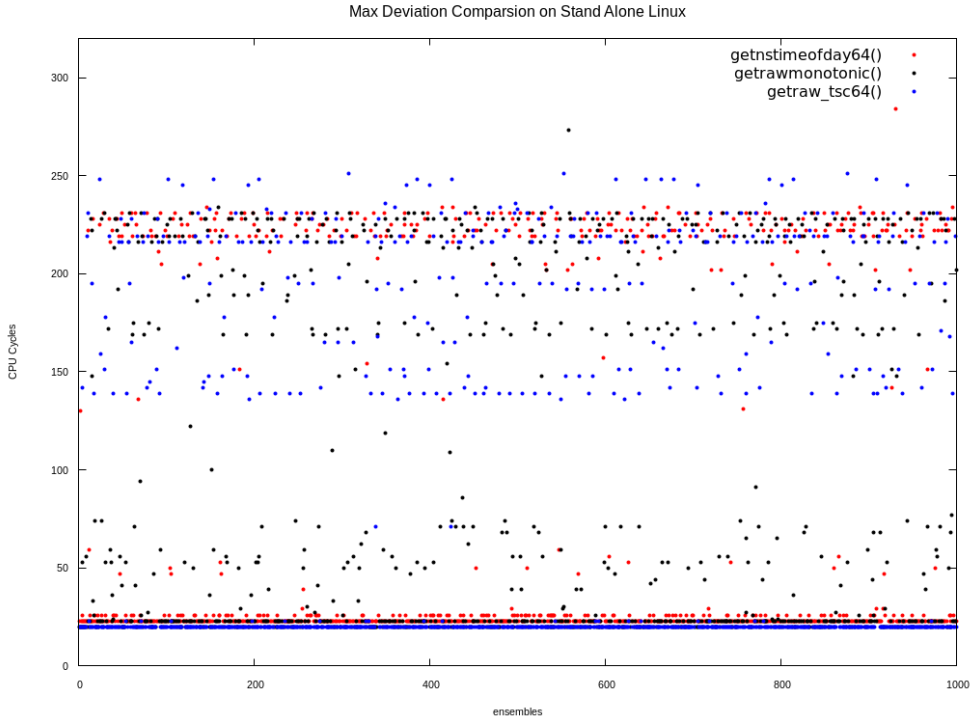
Фигура 5.5. Сравнителен анализ на латентността на вградените getnstimeofday64(), getrawmontonic() и новосъздадения getraw_tsc64() таймери

На фигура 5.5 е показана сравнителна характеристика в режим без виртуализация. Изследването на реализирания таймер е с по-висока латентност. Стойностите не се различават фрапиращо, което показва работоспособността на създадения таймер. Като причина за по-високата латентност могат да се изтъкнат особеностите на програмната реализация на новосъздадения таймер. Поради предназначението на новосъздадения таймер да обслужва време-измервания в реално време, в същия се реализира преизчисляване на стойността на всяко извикване.



Фигура 5.6. Сравнителен анализ на средно квадратично отклонение на таймерите.

На фигура 5.6 е представено средното квадратично отклонение на таймерите. И тук изследването на новосъздадения таймер (показан със синьо) показва по високо средно квадратично отклонение. Като причина може да се изтъкне същата, както при коментара на фигура 5.5.



Фигура 5.7. Девиация на стойностите на таймерите.

На фигура 5.7 са показани измерените стойности и изчислена девиацията. Пресмятането на този параметър е критерий за стабилност на таймера. На фигурата ясно се вижда, че стабилността на новосъздадения

таймер (максимална стойност на девиация 1215 работни цикли на процесора) е значително по-добра от наличните в операционната система таймери. Това е предпоставка за неговото използване в операционни системи от тип реално време.

3.5. Изследване на системи реално време, на база реализирана таймерна система

Време – прекъсванията са много важни за управлението на операционната система. Налични са голямо количество функции в ядрото за отчитането на времето. Много задачи, обработващи се периодично, включват:

- Актуализация на системното време;
- Актуализация на текущото време (the time of day);
- При симетрични мултипроцесорни системи – осигуряване баланс на опашките на планировчика;
- Проверка дали текущият процес е изчерпал време – кванта си за обработка и ако да – препланиране;
- Стартиране на всеки динамичен таймер, който е изтекъл;
- Актуализация използването на ресурсите и статистика на процесорните времена.

Част от тези дейности се извършват след всяко прекъсване на таймерното прекъсване, т.е. дейността се извършва с честотата на тиковете. Други функции се изпълняват периодично, но на всеки n – на брой прекъсвания на таймера. Тези функции се изпълняват на кратен брой тикове.

3.5.1. Изследване на трафик в комуникационни мрежи

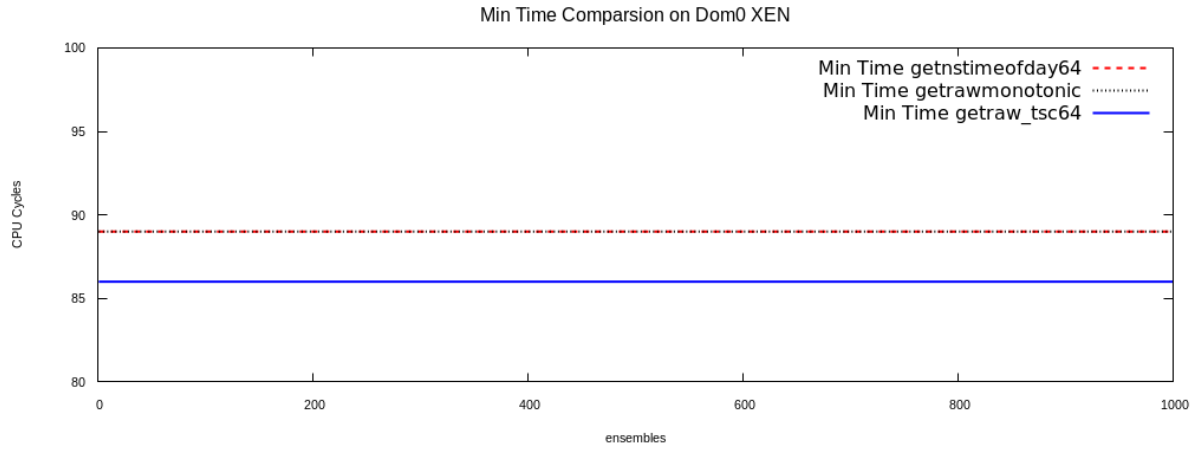
Компютърната мрежа може да се дефинира и като система за комуникация на данни, която свързва компютърните системи в различни точки. Мрежата може да бъде съставена от всяка комбинация от LAN или WAN мрежи. Мрежовият трафик може да бъде определен по много начини. Опростено може да се определи като пропускателна способност на виртуални канали за данни, присъстващи във всяка мрежа. Във всяка компютърна мрежа са налични много и разнообразни комуникационни устройства, които се опитват да получат ресурси и в същото време заявки за пренос. Наличен е голям обем на информация в мрежата под формата на данни за заявки, отговор и контрол. В съвременните компютърни мрежи, при наличната пакетна (предимно) комутация, информацията е представена под формата на огромен брой пакети. Неконтролирано, това количество данни в мрежата, води до забавяне на работата и ненужни латенции в комуникацията.

Мрежовият трафик принуждава организации със среден размер и големи такива да осъзнаят необходимостта за контрол на неговото поведение. Така може да се гарантира, че техните стратегически приложения винаги ще получават необходимите ресурси за обработка оптимално. Контролът на мрежовия трафик изисква ограничаване на пропускателната способност на определени приложения, гарантиране на минимална честотна лента на други и маркиране на трафика с цел приоритизация.

Тук ще бъде показано накратко приложение и изследване отнасянето на ниско-латентната таймерна система за точен контрол на мрежовия трафик.

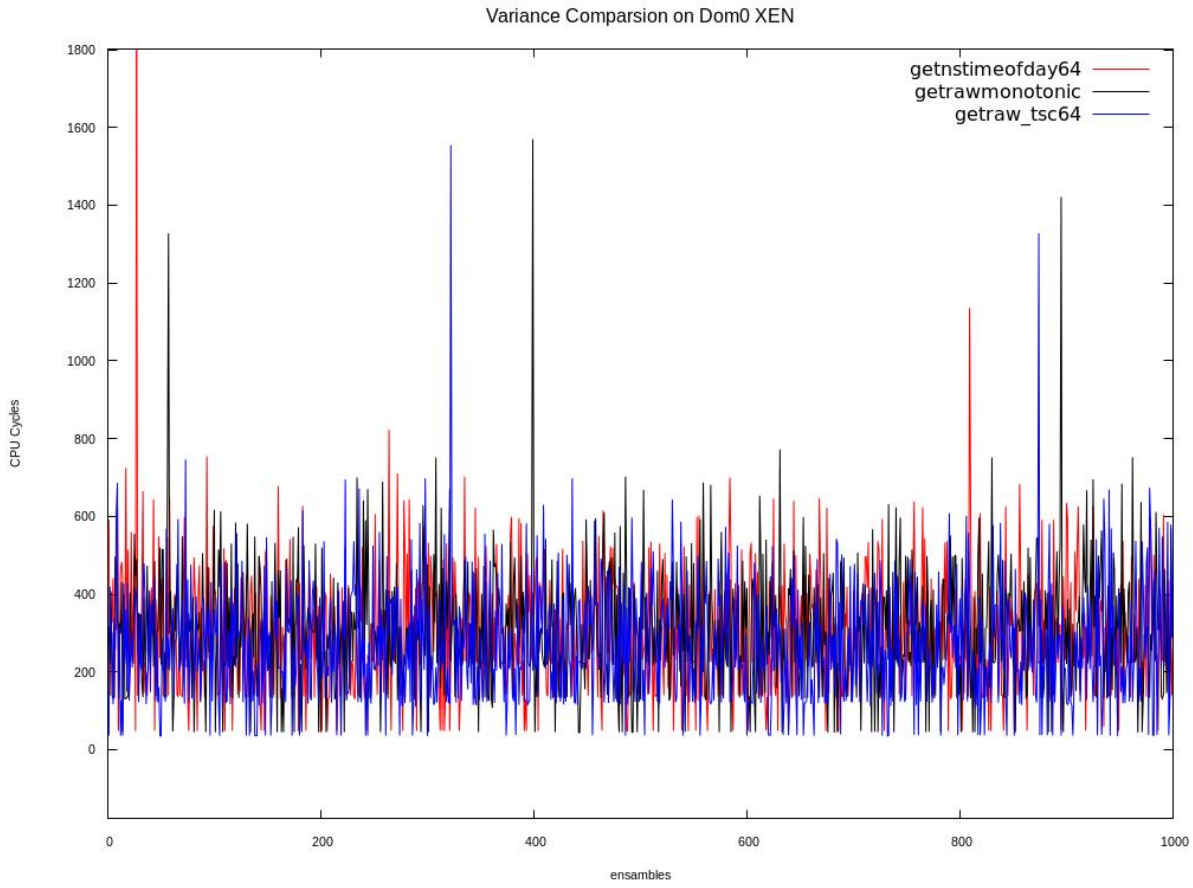
3.5.2. Изследване на таймера при приложения от тип реално време в режим на виртуализация

Спецификата на научните задачи, поставени в настоящата работа, изисква новосъздаденият таймер да е максимално ниско-латентен и по този начин, обработката му да заема минимално количество работни цикли на процесора, да гарантира висока точност и да е достъпен от всички налични виртуализации. По този начин се осигурява единно време при обработката на приложенията в отделните виртуални машини. На фигури 5.11 са показани изследванията и е направена сравнителна характеристика между наличните таймери и новосъздадения, в режим на виртуализация.



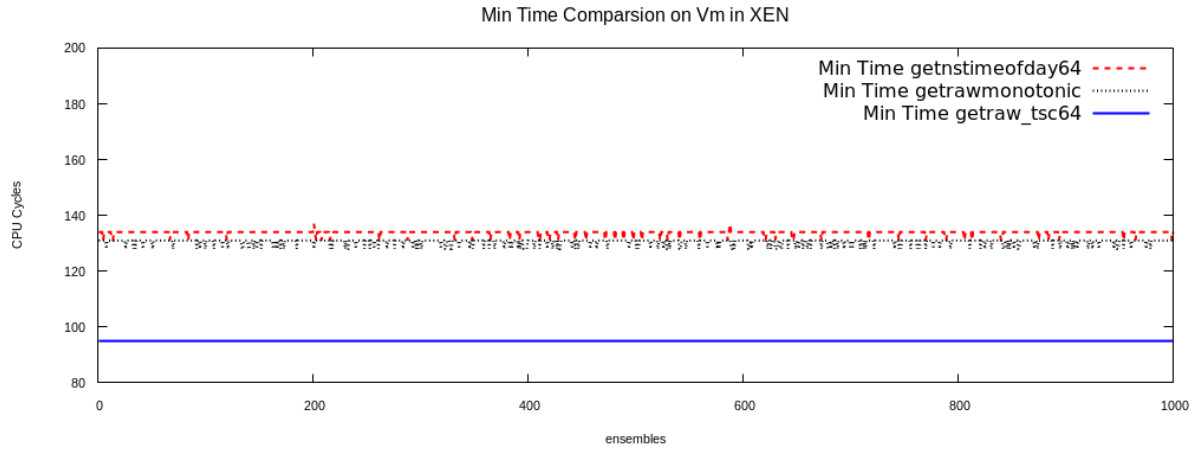
Фигура 5.11. Сравнителна характеристика на латентността на таймерите в dom0.

Фигура 5.11 показва измерената латентност при достъп до стойностите на наличните в операционната система таймери и новосъздадения таймер. Изследването показва предимството на новосъздадения таймер по отношение на латентността. Новосъзданият таймер е оптимиран за виртуализация в реално време, като при неговата програмна реализация се избягват междинни извиквания (hyper calls) и четенето на стойностите е директно от процесора.



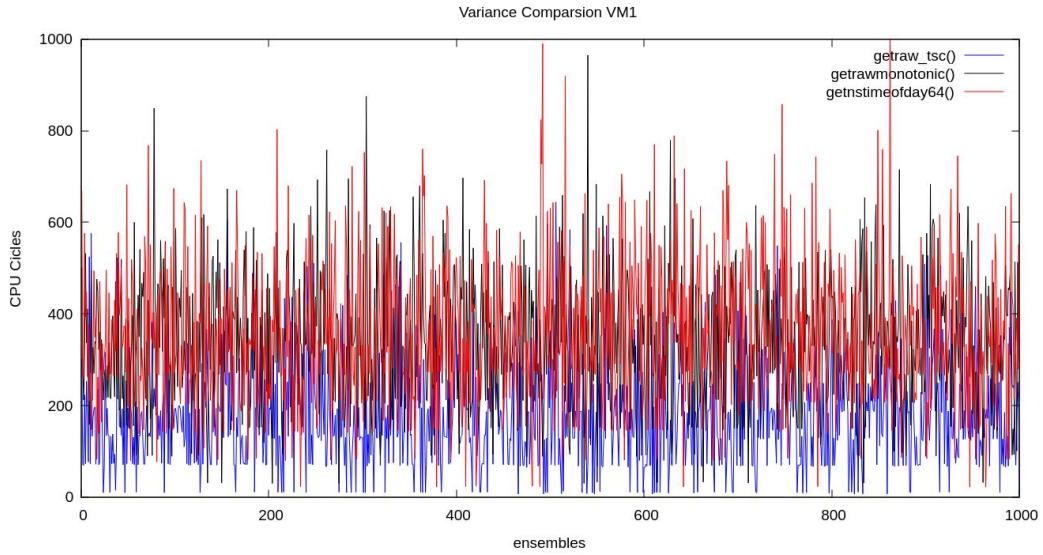
Фигура 5.12. Средно квадратично отклонение на стойностите на таймерите в dom0.

На фигура 5.12 е онагледено изследването на таймера при виртуализация на системи реално време в dom0. Новосъздаденият таймер показва значително по-висока стабилност. Средното му квадратично отклонение е 267 работни цикъла на процесора (за сравнение при наличните таймери тази стойност е над 300). Това предполага препоръчителното използване на таймера в dom0.

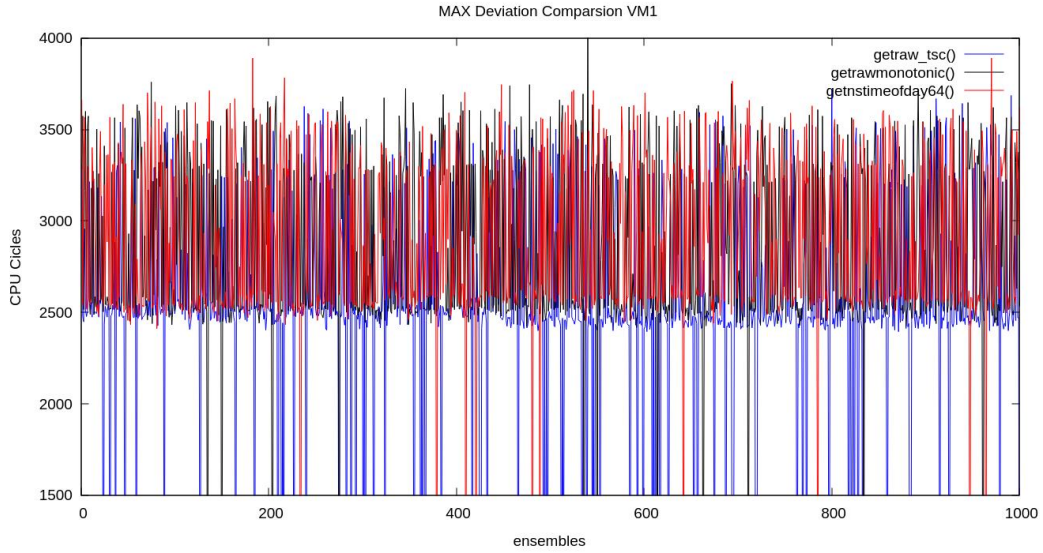


Фигура 5.13. Сравнителна характеристика на латентността на таймерите във виртуална машина vm1.

Изследването на латентността на новосъздадения таймер, както и сравнителната характеристика със съществуващите, е показана на фигура 5.13. От фигурата се виждат действителните предимства на новосъздадения таймер и работна среда за измерване на времето в режим на виртуализация на приложения в реално време. Латентността на новосъздадения таймер е значително по-малка в сравнение с наличните работни среди. Постигнато е абсолютно време за обработка на таймера, съизмеримо с това в режим без виртуализация. Това е истинското предимство на създадената система за измерване на времето, гарантираща „отнемането“ на минимална производителност (процесорно време) за обработка на служебни задачи. Това предоставя максимална процесорна производителност (процесорно време) за действителната обработка на задачата.



Фигура 5.14. Сравнителен анализ на средното квадратично отклонение в режим на виртуализация vm1.



Фигура 5.15. Сравнителен анализ на девиацията в режим на виртуализация vm1

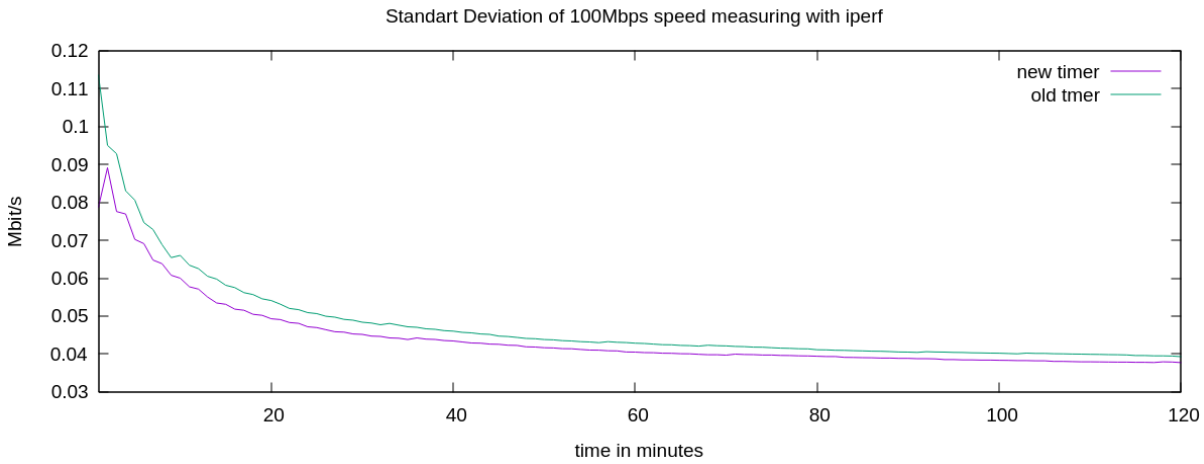
На фигури 5.14 и 5.15 са показани изследванията на девиацията и на квадратичното отклонение на таймерните системи в режим на виртуализация. Направено е сравнение между наличната таймерна система и новосъздадената. Изследванията показват значително по-добри стойности относно новосъздадената система, което я прави подходяща за използване при виртуализация на системи от тип реално време.

3.5.3. Изследване на таймера и заеманите ресурси при измерване на мрежов трафик

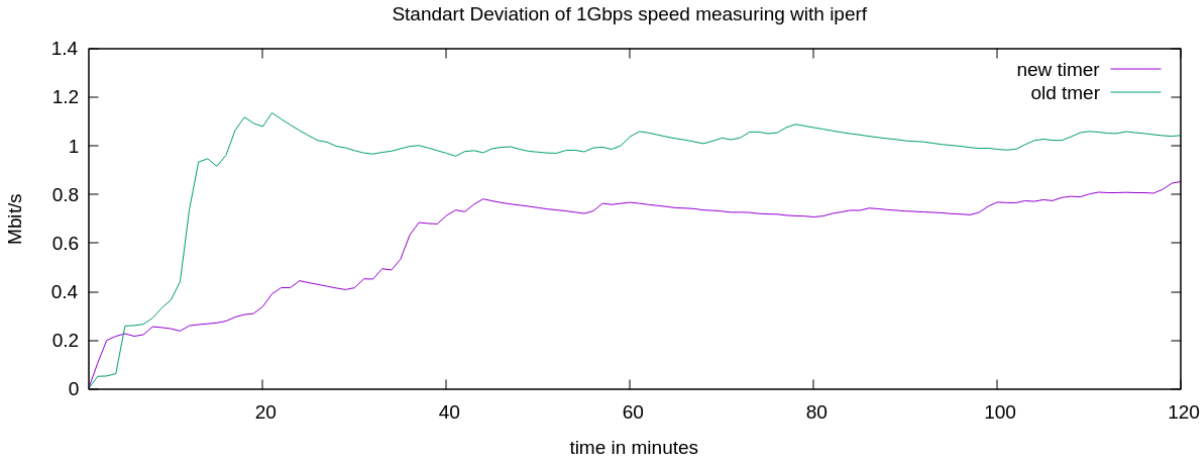
Изследването на мрежов трафик е фундаментална задача. Верните измервания са предпоставка за действителна оптимизация и правилно конфигуриране на компютърни комуникационни системи в условията на конвергенция на услугите. За измерването е направена следната опитна постановка:

- Хардуер: дънна платка ASUS H97-PRO, PCI bridge: Intel Corporation Xeon E3-1200 v3/4th Gen Core Processor, PCI Express x16 Controller, CPU Intel 4x Core i7-4790K 4.00GHz 8M Cache, RAM 32GB;
- Операционна Система: Gentoo/GNU Linux, 64 битова, версия 17.0 (stable), Linux-ядро 4.11.65, включени опции на процесора : constant_tsc и забранени c_states
- VMM Xen 4.11.1 , Две пара-виртуални машини, работещи всяка на отделен процесор с еднакви операционни системи. Към всяка една от тях е присъединена по една мрежова карта ;
- За измерване на комуникацията между виртуалните машини се използва програмата “iperf” версия 2.0.9 , като таймерът ѝ е подменен с новоразработения. Промените в изходния код на iperf са дадени под формата на patch в ПРИЛОЖЕНИЕ 3;

Предвидено е директно измерване от мрежовите карти, позволяващо да се прави реална преценка въз основа на максимално натоварване, без да се има предвид латентността на действителните мрежови интерфейси. На фигура 5.16 и 5.17 са показани снети характеристиките на трафик в така дефинираната постановка. На фигурата е онагледена сравнителна характеристика между наличните конвенционални таймери и новосъздадения, в режим на виртуализация.



Фигура 5.16. Девиация при измерване на скоростта на комуникацията между vm1 и vm2 с помощта на iperf при скорост 100Mbps



Фигура 5.17. Девиация при измерване на скоростта на комуникацията между vm1 и vm2 с помощта на iperf при скорост 1Gbps

На фигура 5.11 са показани заеманите ресурси при използване на конвенционалните таймери и на новосъздадения. Ясно се виждат предимствата на новосъздадения таймер по отношение на малкото заемане на ресурси. Този факт позволява приложение на новосъздадената таймерна система в приложения от тип реално време. Графиката ясно показва предимствата на концепцията, застъпена в рамките на настоящата разработка.

4. НАУЧНИ ПРИНОСИ

В т. 1.2 от настоящата работа беше дефинирана научната хипотеза за същността на отчитане на времето при управление в режим реално време.

Налична е разлика между относителното и абсолютното време. Планиране на събитие за 5 секунди не изисква концепция за абсолютно време, а за относително (напр. 5 секунди от този момент). Обратно, управлението на текущите ден и време изисква ядрото не само да отчита времето, но и да може абсолютно да го измерва. Тези две концепции са критични за управлението на времето. На база формално представяне на времето и формализация работата на процесите от тип реално време беше изградена нова постановка за измерване на реалното време в компютърната система, предразполагащо коректни планиране и комуникации. Конкретните приноси на настоящата работа могат да се дефинират по следния начин:

4.1. Научно-приложни

Научно-приложните приноси на дисертацтионния труд могат да бъдат формулирани по следния начин:

- Формализация елементите на операционните системи – създаден е слоен модел на операционните системи. Формализацията е на база структуриран модел на операционната система, състоящ се от ресурси, активности, интерфейси и протоколи. Формализирано са описани и ресурсите и активности като елементи на операционната система. Този подход дава възможност за симулация състоянията в дадена операционна система. Подходът използван в модела дава възможности за въвеждане на динамични характеристики на отделните компоненти. Резултатите са публикувани в [3] и [4] ;
- Формално описание на системите от тип реално време, дефинирайки основни компоненти в тях. Направени са изследвания, свързани с

оптимизация на системите реално време. Показан е модел на определени типове разпределители за операционни системи от тип реално време. Изчислени са и експериментално са доказани границите за приложимост на разпределители от тези видове. Резултатите са публикувани в [1], [2] и [6];

- Формално описание на елементите при обработка на периодични и апериодични задачи от тип реално време. Доразвита е формализацията на обработка на периодични и непериодични процеси в режим реално време. Изследвани са динамични разпределители за операционни системи от тип реално време, позволявайки реализацията на динамично приоритизиране. Въпреки предимствата на този подход все пак не се препоръчва динамично приоритизиране на процеси в режим реално време, с оглед сигурност. Резултатите са публикуван и в [7];
- Изследване пълното натоварване на системи от тип реално време и определяне на допустим критерий за работоспособност. Изследвана е и степента на оптимално натоварване на системи от тип реално време. Резултатите са публикувани в [7].

4.2. Приложни приноси

- Практическо изследване на системи, предоставящи виртуализация и паравиртуализация, концентрирайки се върху примери от тип „Системи с отворен код“, в частност XEN. Изследвани са условията за измерване на време с минимална латентност. Показана е концепция за измерване на време на системно ниво. Резултатите тепърва ще бъдат публикувани;
- Представена е реализация на работно обкръжение (Framework), гарантиращо точност при отчитане на времето – използвайки резултатите от измерванията на дисперсията и квадратичното отклонение, беше разработена концепция за изграждане на работна среда на база ниско-латентен таймер. Разработеният таймер е с висока

производителност и много ефективно използване на ресурсите в операционната система. При разработката на работното обкръжение са спазени всички условия за обръщение към функциите, в съответствие с концепцията на ядрото на Linux. Това дава възможност за приложение на работната среда при управлението на време-критични процеси. Тепърва предстои подготовка на публикации по темата, които ще допринесат за една успешна постдокторантура;

- Сравнителен анализ със съществуващи системи – в гл. 5 са дадени сравнителни характеристики на наличните конвенционални и новосъздадената таймерна система, показващи предимствата. По същество този принос може да се разгледа като научно приложен, но подходът позволява да се допълни теорията на реалното време. Тези задачи по своята фундаментална същност надхвърлят обема на настоящата работа;
- Изследвани са възможностите за реално време на процесори от фамилия INTEL. Установени са дисперсиите при измерване на времена с помощта на тези процесори. Предложена е методика за измерване на време за обработка на програмни единици;
- Програмно е предефиниран и е създадена програмна структура на единен таймер за виртуализатори, използващи се за виртуализация на системи от тип реално време. Създадена е единна таймерна система, достижима от потребителски режим. Изследванията показват предимствата на ново – създадената таймерна система, гарантираща синхронизацията, при комуникация между компонентите в операционната система. При програмната реализация са спазени всички дефиниции на POSIX, което позволява лесното интегриране на таймерни системи на всички налични виртуализатори;

- Изследвани са предимствата на новосъздадената таймерна система. Практически изследвания на системи с различни конфигурации доказват работоспособността и по – високата ефективност при работата на таймерната система.

5. НАСОКИ ЗА БЪДЕЩИ ИЗСЛЕДВАНИЯ

Формализацията на операционните системи, описана в предходните глави, създава условия за моделиране на специализирани системи със специално предназначение. Такова моделиране може да бъде основа за създаване на симулатор за изпитване на реални системи по отношение на тяхната работоспособност в условията на максимално натоварване. Разработката може да осигури проверката както на системи от спорадичен тип, така и на системи, обработващи периодични процеси.

Един неразработен проблем, който в момента е с научно и научно приложно значение е виртуализацията на системи от тип реално време. Виртуализацията дава възможност за по-ефективно използване на ресурсите в системата. Ефективното разпределение може да се оптимизира на две нива:

- базово ниво (при използване на виртуализатора XEN, това е Dom0) - на това ниво в момента се практикува използването на разпределения без динамично преизчисляване на приоритетите на виртуални машини. Факт е, че преизчисляването на приоритетите е източник на допълнително натоварване на процесора, отнемайки ресурса за "служебна работа", а не за действителна обработка на задачите. Този факт усложнява допълнително оптимизацията. При операционните системи е постигната сложност на разпределителя от тип O(0). Интерпретацията на това може да бъде следната: времето за превключване на процесите в разпределителя не зависи от броя на процесите. Този известен факт при ядрата на Linux (версии след 2.6) би могъл да се оспори, не само поради различни измервания известни в литературата , но и със замервания правени в рамките на настоящата работа. Тези измервания нямат отношение към системите от тип реално време и затова не са включени в настоящата работа. Наличието на ниско-латентен таймер, разработен тук, и съвместимостта на работната среда с концепцията на операционната система Linux е предпоставка за

оптимизация в това направление и по отношение на системите от тип реално време;

- ниво виртуална машина (Domn, в съответствие с терминологията на XEN) - направления за следващи изследвания е не само оптимизация по отношение на преизчисляване приоритета на процесите, но и по отношение на динамично преразпределение на основния апаратен ресурс - процесора. Възможно е да се мисли за динамично преразпределяне на броя на ядрата, обработващи определената виртуална машина. Такъв динамизъм би създал условия за ефективно използване на процесора.

Изследванията в рамките на настоящата работа показаха и феномен, неизвестен досега. Не са известни факти в литературата, които да обясняват драстичното намаляване на времето за обработка на таймера в режим на виртуализация. На този етап няма да се дава такава и тук, но вече се мисли за опитна постановка, която да предостави информация и разумно обяснение по въпроса. Освен че изясняването на този факт ще даде възможност не само за публикации във водещи списания, неговата формализация би могла да бъде основа за разработка на хабилитационен труд.

СПИСЪК НА ПУБЛИКАЦИИТЕ ПО ТЕМАТА НА ДИСЕРТАЦИОННИЯ ТРУД

1 Simeonov S., Simeonova N., Iliev A., Concepts for Creating Operating Systems with Special Purpose, International Scientific Conference UNITECH'12, Gabrovo,16-17 November 2012, Proceedings, p. 377-381, ISSN 1313-230X

2 Simeonov S., Simeonova N., Iliev A., Deterministic Model of CPU Scheduling in Real Time Operating System, International Scientific Conference UNITECH'13, Proceedings, Vol. II, p. 192-197, ISSN 1313-230X

3 А. Илиев, Формално Описание На Компоненти В Операционни Системи, Годишник на секция "Информатика" Съюз на учените в България Том 6, 2013, 95–102

4 Iliev A, Formal Description of Components in Operating Systems,IJITEE (ISSN: 2278 - 3075, Volume-3, Issue-4, September 2013) 96-98

5 Simeonov S., Simeonova N., Iliev A., Hierarchical Scheduling for Real-Time Systems Based on Multicore Configuration for PCs, ANNUAL of Assen Zlatarov University, Burgas, Bulgaria, 2015, v. XLIV (1)

6 Simeonov S., Simeonova N, A. Iliev., Compositional planning of real time systems in processors X_86, International Scientific UNITECH'15, v. II, pp.189-194, ISSN 1313-230X Conference

7 Todor Kostadinov, Asen Iliev, Stanislav Simeonov, AUTONOMOUS LIGHT DIAGNOSTICS SYSTEM FOR THE PURPOSES OF SEA BOUOY MONITORING, ANNUAL of Assen Zlatarov University, Burgas, Bulgaria, 2017, v. XLVI

8 Simeonov S., Simeonova N, Iliev A, MONITORING OF TECHNICAL NAVIGATION TOOLS, International Scientific Conference UNITECH'2017, Gabrovo, 17-18.11

9 Stanislav Simeonov, Asen Iliev , MONITORING OF ECOLOGIC AND NAVIGATION PARAMETERS , International Journal Bioautomation , Year 2019, In printing, SJR (2018): 0.267