**ШУМЕНСКИ УНИВЕРСИТЕТ**
**„ЕПИСКОП КОНСТАНТИН ПРЕСЛАВСКИ“**
**ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА**
**КАТЕДРА „КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ“**

**ДЕЛЯН ХРИСТОВ СЪРМОВ**

# ИЗСЛЕДВАНЕ НА ИНТЕГРАЦИОННИТЕ ПРОЦЕСИ В СЪВРЕМЕННИТЕ ОПЕРАЦИОННИ СИСТЕМИ

## АВТОРЕФЕРАТ

на дисертация за присъждане на образователна и научна степен „доктор“ в област на висше образование 4. Природни науки, математика и информатика, професионално направление 4.6 Информатика и компютърни науки

**Научен ръководител:**
**доц. д-р Константин Стойчев Цветков**

**Шумен, 2013**

**ШУМЕНСКИ УНИВЕРСИТЕТ**
**„ЕПИСКОП КОНСТАНТИН ПРЕСЛАВСКИ“**
**ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА**
**КАТЕДРА „КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ“**

**ДЕЛЯН ХРИСТОВ СЪРМОВ**

# ИЗСЛЕДВАНЕ НА ИНТЕГРАЦИОННИТЕ ПРОЦЕСИ В СЪВРЕМЕННИТЕ ОПЕРАЦИОННИ СИСТЕМИ

## АВТОРЕФЕРАТ

на дисертация за присъждане на образователна и научна степен „доктор“ в област на висше образование 4. Природни науки, математика и информатика, професионално направление 4.6 Информатика и компютърни науки

**Научен ръководител:**
доц. д-р Константин Стойчев Цветков

**Научно жури:**
проф. дтр Атанас Иванов Начев
проф. д-р Маргарита Кръстева Тодорова
проф. д-р Маргарита Стефанова Теодосиева
доц. д-р Георги Стоянов Тодоров
доц. д-р Константин Стойчев Цветков

**Шумен, 2013**

Дисертационният труд е обсъден и насочен за защита на заседание на катедрения съвет на катедра "Компютърни системи и технологии" към ФМИ при Шуменски университет „Епископ Константин Преславски", проведено на 05.02.2013 г. (Протокол №7).

Изследванията, представени в дисертационния труд, са проведени в катедра "Компютърни системи и технологии" при Шуменски университет.

| Данни за дисертационния труд: | |
|---|---|
| Брой страници: | 151 |
| Брой фигури: | 34 |
| Брой таблици: | 18 |
| Брой литературни източници: | 72 |
| Брой публикации по темата на дисертационния труд: | 7 |

Защитата на дисертационния труд ще се състои на 22.05.2013 г. от 13 часа в зала 219 на Шуменски университет на открито заседание на Научното жури.

Материалите по защитата са на разположение на интересуващите се в библиотеката на ШУ„Епископ Константин Преславски" и на интернет страницата на шуменски университет.

Автор: Делян Христов Сърмов
Заглавие: Изследване на интеграционните процеси в съвременните операционни системи

# Съдържание

Използвани съкращения....4
I. УВОД....5
Актуалност на проблема....5
Обект на дисертационния труд....6
Предмет на дисертационния труд....6
Цел на дисертационния труд....6
Задачи....6
Структура и състав на дисертационния труд....7
Благодарности....7
II. СЪДЪРЖАНИЕ НА ДИСЕРТАЦИОННИЯ ТРУД....7
ГЛАВА 1. Проблеми при осигуряване на съвместимост между операционни системи, цели и задачи на дисертационния труд. .7
Изводи и получени резултати....8
ГЛАВА 2. Основни средства за реализация на високопроизводителни системи и технологии за виртуализация. ....8
Изводи и получени резултати....12
ГЛАВА 3. Структурно моделиране на паралелна високопроизводителна изчислителна система с използване на изолиран изчислителен ресурс....12
Изводи и получени резултати....25
ГЛАВА 4. Изследване на производителността....25
Изводи от проведените експерименти....35
III. ЗАКЛЮЧЕНИЕ....36
Основни приноси в дисертационното изследване....36
Публикации по темата на дисертационния труд....37
Цитирана литература в автореферата....38

**Използвани съкращения**

| | | |
|---|---|---|
| ВМ | - | Виртуална машина |
| ВПС | - | Високопроизводителна система |
| ОС | - | Операционна система |
| ПУ | - | Периферни устройства |
| ЦП | - | Централен процесор |
| CFS | - | Completely Fair Scheduler. Мениджър на процеси |
| Guest OS | | ОС работеща във виртуална машина |
| Host OS | | ОС под управлението, на която работи ВМ |
| VT-x | - | Технология на Intel, използвана във ВМ |
| AMD-v | - | Технология на AMD, използвана във ВМ |
| RVI | | Rapid Virtualization Indexing. Технология за хардуерно подпомогната виртуализация от второ поколение на AMD. |
| IOMMU | | Input/output memory management unit. Технология за хардуерно подпомогната виртуализация от второ поколение на Intel. |
| passthrough | | Обобщено наименование на IOMMU и RVI. |

# I. УВОД

В тази дисертация са изследвани интеграционните процеси в съвременните операционни системи от гледна точка на използване на свободните компютърни ресурси във високопроизводителна система. Най-напред са анализирани са причините за наличието на проблема и съществуващите технологии за неговото решаване. Анализът дава основание да се определи помощен инструментариум за решаване на проблема, който включва „виртуални машини“ и „клъстери“. На тази технологична основа е разработен модел на паралелна високопроизводителна система, използваща свободните хардуерни ресурси и осигуряваща интеграция със съществуващите операционни системи с помощта на виртуални машини. За постигане на преносимост, икономическа ефективност и оптимизиране на процеса на внедряване моделът включва използване на преносим, системен носител. Разработена е конкретна реализация на предложения модел, която на следващ етап е използвана за изследване на производителността и функционалността на системата. Получените резултати предоставят информация за приложимостта на моделираната система в проблемната област.

## Актуалност на проблема

Актуалността на проблема се обуславя от съществуващото разнообразие на операционни системи (ОС). Компаниите, разработващи софтуер, създават продуктите си с изисквания към ОС, а често са налице и изисквания към версията на ОС. Потребителите имат нужда да изпълняват софтуер, но поради софтуерните изисквания често се налага да използват и различни ОС. Причините за несъвместимостта между ОС са разнообразни, но основният и практически нерешим проблем е различната архитектура на ядрата на ОС.

Съществуват различни методи за интеграция между по принцип несъвместими ОС, като най-използваните от тях са емулаторите и виртуалните машини. В тази пазарна ниша са разработени множество софтуерни продукти, като Vmware,

VirtualBox, Parallels, Xen, Wine и Cygwin. Може да се обобщи, че тези приложения удовлетворяват потребителските нужди в случаите, когато е допустим компромис с производителността на системата. Използването на високопроизводителен софтуер обаче изисква различен подход, при който се използват ОС оптимизирани за изпълняваната задача.

**Обект на дисертационния труд**

Обектът на дисертационния труд е интегриране на операционни системи.

**Предмет на дисертационния труд**

Предмет на дисертационния труд са методите и инструментите за интеграция на операционни системи, осигуряващи приоритет на изпълнението на високопроизводителния софтуер.

**Цел на дисертационния труд**

Основната цел е изследването е решаване на проблема интегриране на операционни системи при необходимост от използване на високопроизводителна система.

## Задачи

За постигане на така поставената цел, обект на решаване са следните основни задачи:

1. Разработване на методи за изграждане на високопроизводителни изчислителни системи с прилагане на гъвкави и ценово-достъпни решения.
2. Разработване на методи за оптимално използване на свободни ресурси на компютърни системи, обединени в единна изчислителна среда.
3. Разработване на методи за осигуряване на преносимост и гъвкавост на процеса на разгръщане на изчислителната система.
4. Да се разработят принципи и алгоритми за създаване на

интерфейс, обединяващ управлението на високопроизводителна система и виртуална машина.

### Структура и състав на дисертационния труд

Дисертационният труд е структуриран в увод, четири глави, заключение и приложения.

### Благодарности

Особена благодарност дължа на доц. д-р К. Цветков за постановката на задачата, препоръките и подкрепата, оказвани ми през целия период на работа върху дисертационното изследване.

Бих искал да благодаря на доц. д-р С. Станев, д-р Н. Николов и всички участници в научно-изследователския екип, създал паралелната компютърна система „Радиан-М“, с което беше поставено началото на практически изследвания в областта на паралелните системи в ШУ. Тази разработка доведе до идеята за създаването на експерименталната ВПС, представена в текущата дисертация.

## II. СЪДЪРЖАНИЕ НА ДИСЕРТАЦИОННИЯ ТРУД

### ГЛАВА 1. Проблеми при осигуряване на съвместимост между операционни системи, цели и задачи на дисертационния труд

В първа глава е изследвана същността на проблема „несъвместимост между операционните системи“ и причини за неговото пораждане. Разгледана е структурата на ОС и слоевете, в които се наблюдава анализирания проблем.

В параграф 1.3 се анализират средствата, осигуряващи среда за изпълнение на софтуер, предназначен за различни операционни системи.

### Изводи и получени резултати

1. Доказано е, че основната причина за несъвместимост на операционните системи се съдържа в различната архитектура на ядрата им.

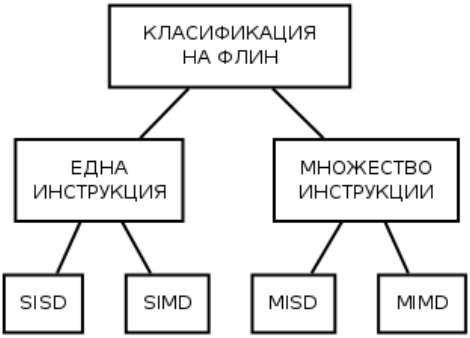
2. Доказано е, че не съществуват решения на проблема загуба на бързодействие при използване на съществуващите средства за интегриране на операционни системи.

3. Показано е, че използването на емулатори, графични терминали и външни ядра са неподходящ инструментариум за решаване на проблема, свързан с осигуряване на съвместимост с повече от една операционна система.

4. Доказано е, че осигуряването на необходимото за високопроизводителните системи бързодействие, при интегриране на различни операционни системи, налага необходимостта от разработване на специални за целта методи и средства.

## ГЛАВА 2. Основни средства за реализация на високопроизводителни системи и технологии за виртуализация.

В параграф 2.1 и 2.2 от първа глава е направен анализ на съществуващите високопроизводителни системи (ВПС). Използвана е класификацията на Флин, която е представена на Фиг. 2.1 [1].




Фиг. 2.1. Класификация на Флин

Според класификацията на Флин съществуват четири типа изчислителни системи:

• SISD (Single Instruction, Single Data) – системи, в които съществува един поток от инструкции и един поток от данни.

• SIMD (Single Instruction, Multiple Data) – един поток инструкции и множество потоци с данни.

• MISD (Multiple Instruction, Single Data) – множество инструкции и един поток от данни.

• MIMD (Multiple Instruction, Multiple Data) – множество потоци команди и данни.

Днес ВПС на практика най-често са паралелни системи от тип MIMD. Те включват изчислителни модули, процесори или компютри, свързани в единна изчислителна система, която има многократно по-висока производителност от широко разпространените персонални компютри. Системите от тип MIMD могат да се разграничат според организацията на паметта. Различават се два вида - с обща и разпределена памет.

Използването на обща памет осигурява еднороден достъп до паметта, но възниква проблема със синхронизацията на кеш паметта на процесорите. Наличието на общи данни по време на паралелни изчисления води до необходимостта да се синхронизират едновременно изпълняваните потоци команди. Когато се променят общите данни, трябва да се гарантира, че в

никой друг поток няма да се обработват тези данни, докато не се получат актуализираните стойности. Необходимостта от синхронизация възпрепятства създаването на системи с голям брой процесори.

При системите с разпределена памет всеки процесор има собствена памет. В този случай трябва да се отчита, че времето за достъп на процесора до отдалечена памет е много по-голямо от времето за достъп до собствената памет.

Многопроцесорните системи, в които всеки процесор има достъп само до своята собствена памет, се наричат no-remote memory access (NORMA) системи. В тях за предаване на данни, които се намират в паметта на други процесори е необходимо да се изпълнят операции по предаване на съобщения. Този подход се използва за създаването на двата вида разпространени в момента ВПС – многопроцесорни компютри и клъстери.

Клъстерът е група компютри, обединени в локална изчислителна мрежа, които могат да работят като единен изчислителен ресурс. Компютър, включен в клъстер, обикновено се нарича възел и има собствен процесор, памет, операционна система, входно-изходна подсистема и е в състояние да изпълни високопроизводителни изчисления заедно с други възли. Като възли в клъстера най-често се използват работни станции с Linux и друг софтуер с отворен код. Клъстерите се характеризират с балансирано натоварване, висока производителност, висока надеждност, относително ниска цена, наличие на възможности за надграждане и подмяна на хардуерни компоненти [2].

Всички изброени предимства на клъстерите водят до извода, при моделиране на ВПС да се използва именно такъв тип система.

В параграф 2.3 от първа глава са анализирани технологиите за виртуализация, като е обърнато внимание на хардуерните технологии, използвани от виртуалните машини (ВМ). Съществуват две разновидности на софтуера за виртуализация.

Първият е виртуални машини, работещи под управление на операционна система. Използват се приложения, създаващи виртуални машини, които обезпечават работата на друга ОС (Guest OS). Базовата ОС (Host OS) работи в привилегирован

режим. При втория вид виртуализация ОС работят едновременно под управление на програма мениджър на виртуални машини (МВМ). Тази програма осигурява безконфликтно разпределение на ресурсите на компютъра между ОС.

Хардуерната поддръжка се осигурява посредством инструкции, интегрирани в централния процесор (ЦП). В процесорите на компанията Intel технологията се нарича VT-x. В процесорите на AMD се използва наименованието AMD-v, а в процеса на разработка е използвано кодовото име Secure Virtual Machin (SVM) [3]. Списък на процесорите на AMD и Intel с апаратна поддръжка на ВМ е представен в Приложение 1.

Технологиите за хардуерно подпомагане на виртуализацията осигуряват директен достъп на Guest OS до ЦП. Благодарение на вградената транслация на адреси се ускоряват и операциите с оперативната памет. Резултатът от използването на VT-x и AMD-v е значително повишение на бързодействието на ОС, работеща във ВМ [3].

В процесорите Core i7 на Intel и AMD Opteron трето поколение е въведена усъвършенствана поддръжка на хардуерна виртуализация. Добавена е възможност за директен достъп на Guest OS до периферните устройства (ПУ). Използват се различни имена, описващи технологията, между които: input/output memory management unit (IOMMU), Rapid Virtualization Indexing (RVI), passthrough [4]. Въпреки високия потенциал на тази технология, ние взехме решение да не я използваме в експерименталната реализация на ВПС. Причините са слабото разпространение на ЦП, които поддържат passthrough, и нерешените проблеми със съвместимостта при ПУ.

В параграф 2.4 се анализират опитите за интеграция на виртуални машини и клъстери. Приведени са два конкретни примера за интеграция на клъстер с виртуални машини. При тези разработки възлите на клъстера са реализирани във ВМ под управление на МВМ Xen. Използват се специализирани InfiniBand мрежови устройства, до които на Guest OS се осигурява директен достъп с помощта на допълнителен софтуер (Xen-IB и Xenoprof) [5,6]. Анализирани са известните предимства и недостатъци на

съществуващите решения.

### Изводи и получени резултати

1. Доказано е, че клъстерите са предпочитано архитектурно решение за изграждане на оптимални в отношение производителност-цена високопроизводителни системи.

2. Показано е, че търсенето на оптимални решения е свързано с начина на реализация на възлите във виртуални машини;

3. Показано е, че използването на ВПС с възли, реализирани във ВМ, е свързано с компромиси по отношение на производителността и функционалността на ВПС.

## ГЛАВА 3. Структурно моделиране на паралелна високопроизводителна изчислителна система с използване на изолиран изчислителен ресурс

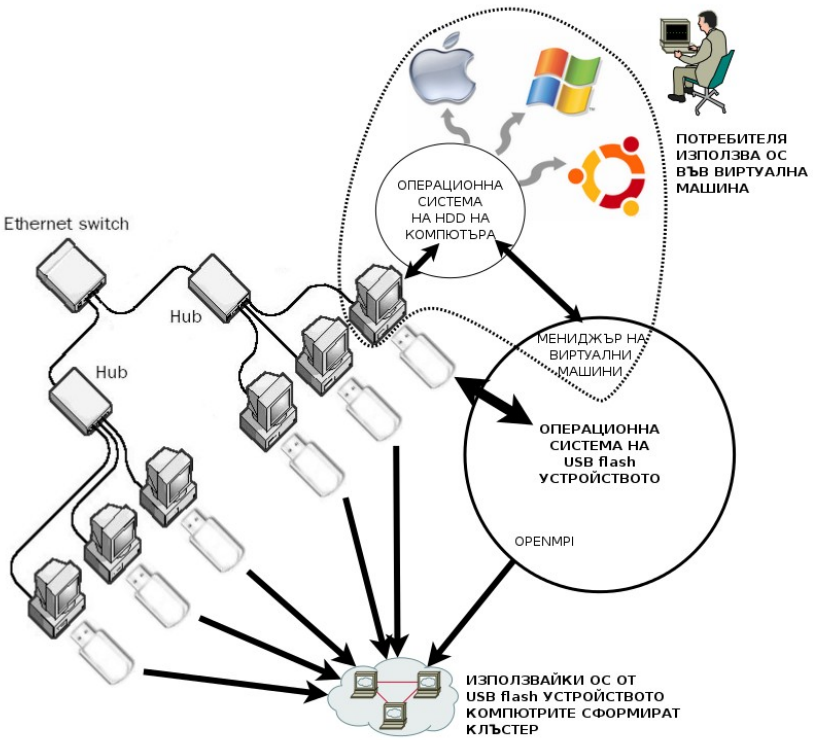
Съвременните персонални компютри работят под управлението на ОС. Всяка потребителска ОС е подложена на рискове по отношение на сигурността и стабилността, в зависимост от използваните приложения, мрежови услуги и опита на потребителя, работещ на нея. При създаване на клъстер за изчисления трябва да се отчетат тези фактори и да се минимизира риска от грешки в изчисленията, предизвикани от събития в потребителската ОС. Нашият модел включва използване на изолация на ниво операционна система, в която да функционира софтуерът, реализиращ възел от клъстер. Благодарение на бързото развитие на ВМ вече съществуват инструменти, с които това може да бъде постигнато. Тъй като бързодействието на всеки от възлите е фактор, който оказва влияние на бързодействието на целия клъстер, не е препоръчително клъстерният софтуер да работи във ВМ. Възлите от клъстера трябва да бъдат реализирани в Host OS, докато потребителската ОС остава достъпна посредством ВМ.

Изборът на Host OS (параграф 3.1) е предопределен поради факта, че GNU/Linux е на практика единствената безплатна ОС, подходяща за използване във ВПС, която при това се използва в

над 93% от суперкомпютрите в класацията TOP500. Конкретната дистрибуция, която е използвана, е Arch Linux, но всичко реализирано на нея е възможно да бъде изпълнено на всяка друга дистрибуция за обща употреба.

В параграф 3.2 са анализирани широкоразпространените устройства и интерфейсите, към които се свързват, подходящи използване като системен носител. В резултат, за реализацията на експерименталната ВПС е използвано USB flash устройство. Причините, които доведоха до този избор са:

- Възможност за зареждане на ОС от USB flash устройство;
- Ниско енергопотребление;
- Малък размер и тегло;
- Висока устойчивост на вибрации, магнитни полета и резки температурни амплитуди;
- Прилагат стандарта USB flash memory devices, поради което не са необходими драйвери;
- Широко разпространение;
- Ниска цена.




фиг. 3.1: Принцип на работа на експерименталната ВПС

На фиг. 3.1 е показан принципът, който се използва при изграждането на експерименталната ВПС. Използването на външно USB флаш устройство дава възможност ОС, която се включва в клъстер, да предостави стандартното дисково устройство за монополно използване на Guest OS. Този подход решава редица традиционни за виртуалните машини проблеми, свързани с управлението на достъпа до отделните дялове на дисковото устройство.

В параграф 3.3 е изложен метод за инсталация, илюстриран с конкретен пример на базата на дистрибуцията Arch Linux. Акцентира върху специфичните конфигурации на ОС, наложени

от употребата на  на системно USB flash memory устройство. Командите при инсталационния процес са специфични за Arch Linux, докато настройките, необходими за работа от USB flash memory, са приложими за всяка Linux дистрибуция. Инсталационните инструкции са актуални към 08.2012 г.

По време на инсталацията на базовата система е необходимо да се обърне специално внимание на особеностите, произтичащи от използването на системно USB flash устройство.

Ядрото на операционната система е конфигурирано по подразбиране да зарежда от твърдия диск, свързан към ATA интерфейс. За да се добави възможност за зареждане от други интерфейси, е необходимо да се създаде нова базова файлова система на ядрото в паметта, където да се включи поддръжка на използваните видове интерфейси. Изграждането ѝ се извършва с инструмента mkinitcpio.

USB flash устройствата имат ограничен брой цикли на запис. В зависимост от конкретния производител и качеството на устройството този брой може да варира в интервала 10000-100000 цикъла на запис. При традиционната употреба на USB flash устройствата максималният праг е трудно достижим, но при употреба като системен носител е необходимо да се предприемат мерки за ограничаване на циклите на запис. Една от мерките за постигане на тази цел е да не се създава SWAP дял и в последствие да не се използва SWAP файл. SWAP дяловете се използват интензивно, и в зависимост от употребата на ОС, е възможно максималният праг да бъде достигнат в рамките на няколко часа. Препоръчителна е и корекция във файла /etc/fstab, където трябва да се добави параметърът relatime към реда, който съдържа описанието на системния дял.

Има три начина, по които могат да се идентифицират файловите системи. Това са kernel name descriptor, label и UUID. Kernel name descriptor е името на блоковия файл в папката /dev представящ устройства или дялове от устройства. Например /dev/sda1, /dev/sda2, /dev/sdb1, /dev/sdb2. Label е името на дяла, което се присвоява най-често при създаването му. Тъй като USB flash устройството е преносимо и ще бъде включвано на различни

компютри, първите два метода на идентификация не осигуряват еднозначно идентифициране на системния дял. Например на някои от компютрите е възможно да съществува дял със същия Label или да има различен брой твърди дискове, при което kernel name descriptor ще използва различно означение за USB flash устройството. Поради тази причина трябва да се използва UUID идентификатор. Съгласно инсталационната процедура, описана в параграф 2.3, е необходимо да се използват UUID идентификатори в конфигурационните файлове /boot/grub/grub.cfg и /etc/fstab.

В параграф 3.4 е обърнато внимание на софтуерните пакети, които са добавени след базовата инсталация. Почти всички от добавените пакети са необходими за функционирането на клъстера или софтуера за виртуализация. В случаите, когато е възможен избор между различни програми, са избирани такива, които използват минимални хардуерни ресурси. Следва списък на добавени пакети, групирани по категории:

• Звукова подсистема - alsa-utils alsa-oss pulseaudio pulseaudio-alsa paprefs

• Графичен сървър, който е необходим за функционирането на графична среда и софтуера за виртуализация – xorg-server, xorg-xinit, xorg-server-utils, xorg-twm, xterm, xf86-input-keyboard, xf86-input-mouse, xf86-input-synaptics, xf86-input-evdev, xf86-video-vesa

• Пакети за графична среда - lxde, gamin, obconf

• Приложни програми за осигуряване на минимална функционалност на системата - links, midori, epdfview, leafpad, xarchiver, unrar, unace, unzip, man-pages

• Компилатори и средства за разработка - gcc, gcc-fortran, gcc-objc, gcc-docs, codeblocks, fakeroot, dialog, xdialog

• Драйвери за различни модели видеокарти, необходими за софтуера за виртуализация - xf86-video-ati, lib32-ati-dri, catalyst-dkms, lib32-catalyst-utils, xf86-video-intel, lib32-intel-dri, xf86-video-nouveau, nouveau-dri, lib32-nouveau-dri, nvidia, nvidia-utils, lib32-nvidia-utils

• Софтуер за виртуализация – qemu-kvm-spice, libvirt, virtinst, virt-manager, spice, spice-gtk, spice-protocol, virt-viewer, libcacard, celt0.5.1, cegui, brltty

• Софтуер, необходим за функционирането на клъстера – openmpi, openssh, keychain, nfs-utils

Както се вижда от представения списък пакети, в качеството на MBM се използва KVM. Това е MBM, реализиран на ниско ниво в Linux ядрото и притежава поддръжка на технологиите VT-x и AMD-v. Предимствата на такъв подход са повишено бързодействие и стабилност на виртуалната машина. В експерименталната реализация се използва и програмата за виртуализация Qemu, която в съвременните си версии притежава интеграция с KVM.

На по-високо ниво функционира libvirt. Това е приложно-програмен интерфейс и набор от инструменти за управление на виртуализацията. Libvirt разполага с широк спектър от инструменти за конфигурация, мониторинг и разгръщане на виртуални машини [7]. Някои от използваните инструменти на libvirt в реализацията на клъстерната система са:

• virsh – интерактивна обвивка за изпълнение на операции с регистрирани домейни, мрежи и дискови носители.

• virt-clone – инструмент за клониране на виртуални машини и образи на дискови носители.

• virt-install – инструмент за първоначална инсталация на нова виртуална машина.

• virt-viewer – инструмент за достъп до графична виртуална конзола, свързана с виртуална машина. Базиран е на GTK-VNC и се използва сигурна връзка до отдалечена конзола през VNC протокола. Може да се използва за връзка с виртуална машина, разположена както на локалния компютър, така и на отдалечена система.

Съвременните тенденции в развитието на софтуерните продукти за виртуализация показват, че инструменти като KVM и libvirt се превръщат в практически стандарт в технологиите за виртуализация [7]. Компаниите, разработващи софтуер за

виртуализация, все още поддържат собствените си МВМ, но по етапно мигрират системите си към KVM. Libvirt се използва от програми за виртуализация като KVM/QEMU, Xen, VirtualBox, VMWare ESX, VMWare GSX, VMWare Player, VMWare Workstation, Microsoft Hyper-V. Друго негово приложение са контейнерните системи като OpenVZ, LXC и User Mode Linux.

Друг модул от системата за виртуализация е spice. Spice е софтуерно решение с отворен код, което предоставя възможност за комуникация с виртуализирани устройства. Състои се от 3 основни компонента.

• Spice protocol – дефинира набор от съобщения за достъп, комуникация и контрол между отдалечени периферни устройства и Spice сървъра. Използва отделен комуникационен канал за всяко периферно устройство. Съществува един основен „main“ комуникационен канал и канали за връзка с мрежовия интерфейс, звуков контролер, входни устройства – клавиатура, мишка и дисплей.

• Spice server - комуникира с отдалечения клиент посредством Spice протокола и предоставя достъп на МВМ.

• Spice client – това е виртуалното устройство, което се използва в гост операционната система. Всеки канал свързва сървъра с driver на устройство в ОС, инсталирана като гост. За видео картата се използва Qxl driver, който увеличава производителността на графичната подсистема.

От точка 3.4.4 до точка 3.4.7 са предложени инструменти, осигуряващи функционирането на клъстера.

Използваните в експерименталната ВПС библиотеки за предаване на съобщения между възлите са OpenMPI. Това са библиотеки, които комбинират технологии от други проекти като FT-MPI, LA-MPI, LAM/MPI и PACX-MPI. Актуалните в момента версии на OpenMPI са базирани на MPI-2, но се поддържа обратна съвместимост и с MPI-1 стандарта. Кодът може да бъде разделен на три големи модула [8]:

• OMPI – базов код, реализиращ MPI протокола.

• Open Run-Time Environment (ORTE) – обвивка за

изпълнение на програми, в които се използват библиотеките на OpenMPI.

• Open Portable Access Layer (OPAL) – най-ниския слой в MPI. В него са реализирани функции за управление на процесите, изпълнявани на конкретния възел от клъстера. Осигурява преносимост на основния MPI код между различните операционни системи при управление на хардуерните ресурси, като управление на мрежови интерфейси, споделена памет между процесите, процесорно време и други.

Съществена част от всяка реализация на MPI е предаването на съобщения между възлите в клъстера. В експерименталната ВПС се използва мрежовият протокол Secure shell (SSH) за осъществяване на комуникация между възлите. Оторизацията при използване на SSH протокола може да се осъществи с парола или с криптографски публичен-ключ. Вторият подход дава възможност оторизацията да се избърши без предаване на паролата през мрежата и е препоръчително да се използва за клъстера.

При всяко осъществяване на SSH връзка се изисква въвеждане на парола. За улеснение на администратора, който оперира на сървъра, е необходимо централизирано управление на публичните ключове. Такива програми са познати под името SSH агенти и позволяват връзка към множество SSH сървъри, без да е необходимо въвеждане на парола за всеки един от тях.

Примери за SSH агенти са програмите ssh-agent и GnuPG Agent. За реализацията на клъстера ще бъде използвана програмата Keychain. Тя представлява надстройка на ssh-agent и е реализирана като скриптове на обвивката. Предимство на Keychain спрямо аналогични програми е възможността за управление на публичните ключове с минимална намеса от страна на потребителя. Използва се един процес, който работи по време на множество потребителски сесии. Това означава, че е необходимо само едно въвеждане на паролата за всяко стартиране на компютъра.

За функционирането на OpenMPI е необходимо всеки от

възлите в клъстера да има достъп до програмата, която трябва да се изпълни. В експерименталната ВПС е използвана споделена директория на сървъра, на която всички възли поддържат копия. За целта е използвана мрежова файлова система NFS.

В параграф 3.5 са описани алгоритмите, с помощта на които се осигурява управлението на комплекса използван софтуер. Разработени са командни BASH файлове, подпомагащи дейността по конфигурация и поддръжка на системата. Интерфейсът е изграден с помощта на dialog и xdialog. Dialog е програма, с помощта на която се изобразява текстов потребителски интерфейс. Xdialog има същото предназначение, но изобразяването е в графичен интерфейс. За да се осигури съвместимост между текстов и графичен интерфейс са използвани само команди, които функционират еднакво в dialog и xdialog. Двете програми имат общ размер под 2 MB и ниски системни изисквания. В Приложение 2 е представена функционална схема на скриптовете, с помощта на които е реализиран контрола на MPI-клъстера и виртуалните машини. В приложение 3 е приведен кодът на файловете. Скриптовете са разделени в няколко файла, разположени в домашната папка на потребителски профил:

• header.sh – заглавен файл, използван от останалите скриптове. В него се определя дали интерфейсът да използва dialog или Xdialog. Инициализира параметър на обвивката $tempfile, свързан с временен файл в ОП, който се използва в останалите скриптове.

• menu.sh – основно меню, което препраща управлението към функция в един от останалите четири командни файла.

• network.sh - скрипт, който позволява конфигурацията на мрежовия интерфейс.

• prep_node.sh - основната цел на този скрипт е подготовката на компютърната система за включване като възел в клъстера. Изпраща се информация до сървъра за IP адреса на компютъра и броя на процесорните ядра. Монтира се мрежовата папка /parallel.

• prep_server.sh – Този скрипт се изпълнява само на компютъра, избран за сървър. Стартира услугата nfs-server, която

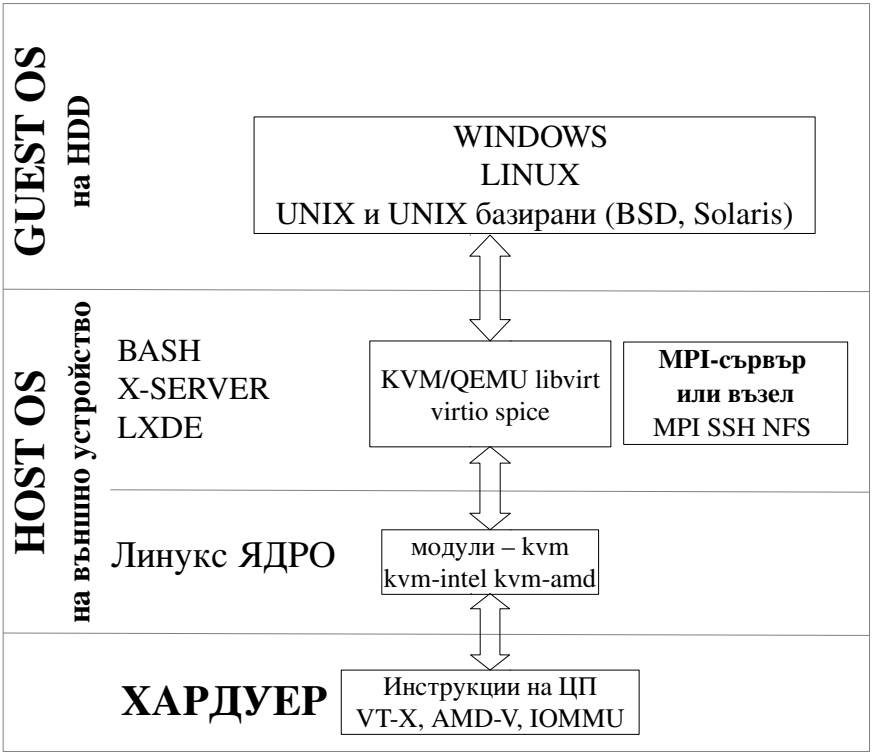
е необходима на останалите възли, за да монтират папката /parallel. Другата важна задача е да се генерира конфигурационен файл .mhost, който се използва от библиотеката openmpi за комуникация с останалите възли в клъстера. За целта се обединява информацията от всички файлове, изпратени от останалите възли.

• vm.sh - В този файл е реализирано меню с най-необходимите функции за управление на ВМ. Операциите, които се поддържат са: „извеждане на списък с дефинираните ВМ в системата“, „промяна на ВМ по-подразбиране“, „извеждане на информация дали ЦП поддържа инструкциите VT-x или AMD-v“, „стартиране“, „изпращане на сигнал за рестартиране“, „изпращане на сигнал за изключване“, „изключване“, „рестартиране“, „свързване“, „прекъсване на връзката“, „модифициране на BIOS на ВМ“, „възстановяване на стандартния BIOS на ВМ“.

• vm – команден файл, към който се обръща „vm.sh“. Позволява да се изпълняват операции с ВМ с команда, без да се използва интерфейса от „menu.sh“.

• inst_seaslic.sh, ro_seaslic.sh – Изпълняват съответно операциите „модифициране на BIOS на ВМ“ и „възстановяване на стандартния BIOS на ВМ“, които се стартират от „vm.sh“.

В параграф 3.6 са описани процедури, които се изпълняват по време на работа с експерименталната ВПС. Експлоатацията на експерименталната ВПС, предполага познаване на организацията на системата. На фиг. 3.2 е представена структурата на софтуера и разпределението му в слоевете на експерименталната ВПС.




фиг. 3.2. Структура на софтуера в един компютър от експерименталната ВПС

Експерименталната ВПС е инсталирана на USB flash устройство с големина 4GB. Всяко такова устройство може да управлява една компютърна система, която се включва в клъстера като възел. За формиране на клъстер от N на брой възела са необходими N на брой USB flash устройства, на които е инсталирана експерименталната ВПС. Най-лесният начин едно USB flash устройство да бъде инсталирано с експерименталната ВПС, е посредством копиране на цялата софтуерна система.

Възможно е да се работи както в текстов, така и в графичен режим. USB flash устройствата могат да бъдат включени към произволен USB порт в компютърната система. Необходимо е да

се укаже BIOS да зареди ОС от USB устройството. В ОС е създаден потребителски акаунт с име „vm“, който се използва за влизане в системата.

Необходимите функции са достъпни с изпълнение на „menu.sh“ в графичен или текстов режим. Ако е необходимо да се промени мрежовата конфигурация, изпълнява се менюто „Network“. Един от компютрите се избира за сървър и на него се изпълнява менюто „Server“, докато на останалите компютри се изпълнява менюто „Node“. За работа с ОС, инсталирана на твърдия диск, се използва командата „Virtual mashine“ от „menu.sh“.

В параграф 2.7 са анализирани проблемите, възникващи при миграция на ОС във ВМ. Програмите за виртуализация разполагат с относително развит инструментариум за създаване и управление на ВМ. Във ВМ могат лесно да бъдат инсталирани и конфигурирани различни ОС. Когато е необходимо обаче във ВМ да се стартира вече инсталирана на компютъра ОС, възникват редица проблеми, които затрудняват миграцията на ОС. Проблемните области при миграцията могат да се групират в три категории:

• Несъвместимост на използваното ядро с виртуалния хардуер. Всички ОС разполагат с ядро, което може да се изпълнява на широк спектър компютърни системи. Това ядро, като правило, се използва по време на инсталационния процес. След завършване на инсталацията в ОС се използва друго, оптимизирано за конкретната компютърна система ядро. Това ядро в някои случаи е възможно да не функционира правилно във ВМ;

• Несъвместимост на използвания Hardware Abstraction Layer (HAL) с виртуалния хардуер. HAL е софтуерно реализиран абстрактен слой, условно намиращ се между хардуера и софтуера. HAL има за цел да скрива разликите в хардуера, за да може софтуерът да се изпълнява без промяна на различни системи. В някои ОС обаче самият HAL е платформено зависим;

• Несъвместимост в драйвери на периферни устройства.

Драйверите са софтуер, който е способен да комуникира с конкретни хардуерни компоненти. Драйверите имат платформено зависима част реализирана на ниско ниво. Поради тази причина използването на други периферни устройства във ВМ води до грешки в ОС. Най-често проблемните драйвери са за видеокартата и звуковата карта.

В зависимост от характера на възникващите проблеми при миграцията, ОС са разделени на три групи, като за всяка група е предложен метод за изпълнение на миграционния процес:

• UNIX съвместими ОС, при които не е необходима предварителна подготовка за включване във ВМ. Най-характерният проблем при тези ОС е невъзможност за зареждане на графичната среда поради отсъствие или неподходяща конфигурация на драйвер за видеокарта. Методът включва инсталиране на един от драйверите „xf86-video-vesa“ или „xf86-video-qxl“. В случай, че „xorg-server“ използва конфигурационен файл „/etc/X11/xorg.conf“, е необходимо създаване на такъв за използвания драйвер.

• Операционни системи на Microsoft - Windows XP и Windows Server 2000/2003. При тези ОС комплексно се съчетават трите вида проблеми при миграцията, но с помощта на вградения в ОС инструмент „Hardware Profiles“ е възможно отстраняването им.

• Операционни системи на Microsoft - Windows Vista, Windows 7 и Windows Server 2008. Проблемите, възникващи при миграцията на Windows Vista/7/2008, също са от трите типа. Решението им обаче е затруднено поради факта, че в тези ОС е премахнат инструментът „Hardware Profiles“. Поради тази причина се използва алтернативен инструмент System Preparation tool (Sysprep). При анализа на изпълнението на „Sysprep“ са открити грешки, документирани в официалния сайт на Microsoft. Разработени са два командни файла „prep_sysprep.bat“ и „post_sysprep.bat“ за успешно завършване на миграцията със „Sysprep“.

Във файл spice-guest-tools-0.1.exe на системното USB flash

устройство е записан комплект от драйвери за ПУ във ВМ. Тези драйвери са необходими за всички версии на ОС Windows, за да функционират правилно във ВМ.

### Изводи и получени резултати

1. Предложен е метод за създаване на паралелни ВПС чрез използване на системен софтуер, инсталиран върху преносими външни устройства.

2. Предложена е експериментална високопроизводителна система, която доказва възможността за приложение на разработените принципи за създаване на възел от Beowulf клъстер, като софтуерът е инсталиран на външно USB flash устройство.

3. Предложен е алгоритъм за конфигурация на мрежовия интерфейс.

4. Разработен е алгоритъм за подготовка на компютъра за включване в клъстера като мрежови възел.

5. Предложен е алгоритъм за подготовка на компютъра за включване в клъстъра като сървър.

6. Предложен е алгоритъм за управление на виртуални машини.

7. Дефинирани са съпътстващи проблеми при миграция на съществуващите операционни системи във виртуална машина.

8. Предложени са методи за изпълнение на миграционния процес на операционните системи.

## ГЛАВА 4. Изследване на производителността

За провеждане на експериментите, описани в текущата глава, са използвани 16 еднакви компютърни системи със следните характеристики.

Таблица 4.1. Параметри на тестовите конфигурации

| | |
|---|---|
| Централен процесор | DualCore Intel Pentium E5700, 3000 MHz (15 x 200) |
| Дънна платка | Gigabyte GA-G41MT-S2PT v1.1 (2 PCI, 1 PCI-E x1, 1 PCI-E x16, 2 DDR3 DIMM, Audio, Video, Gigabit LAN) |
| Чипсет на дънната платка | Intel Eaglelake G41 |
| Оперативна памет | 4 GB DDR3-1333 DDR3 SDRAM |
| BIOS | Award Modular (08/03/11) |
| Комуникационни портове | COM1, LPT1 |
| Видеокарта | Intel GMA X4500 |
| Звуков адаптер | Realtek ALC889 @ Intel 82801GB ICH7 - High Definition Audio Controller |
| IDE контролер | Intel(R) N10/ICH7 Family Serial ATA Storage Controller - 27C0 |
| Дисково устройство | Hitachi HDS721050CLA362 ATA Device (500 GB, 7200 RPM, SATA-II) |
| USB2 контролер | Intel 82801GB ICH7 - Enhanced USB2 Controller |
| USB1 контролер | 4 броя - Intel 82801GB ICH7 - USB Universal Host Controller |
| Мрежова карта | Realtek RTL8139/810x Family Fast Ethernet NIC |

Компютрите са свързани в локална мрежа посредством мрежово устройство:

TRENDnet TE100-S24 10/100Mbps NWay switch

За системен носител са използвани 16 броя USB flash 2.0 устройства с размер 4GB, на които е инсталирана ОС и приложен софтуер по описания в глава 2 начин.

На твърдия диск (HDD) на компютърните системи са инсталирани две операционни системи Windows 7 32bit и Arch

Linux 64bit. При експериментите посочените ОС са използвани в качеството на Guest OS, работещи във виртуална машина.

Използваният софтуерен пакет за тестване на производителността е Beowulf Perforamnce Suite (BPS), от който са използвани програмите:

• bonnie++ – софтуер за измерване на производителността на дисковата подсистема;

• stream – софтуер за отчитане на производителността на паметта;

• netperf – програма за тестване на скоростта при мрежов трансфер;

• NAS Parallel Benchmarks - набор от програми, предназначени да подпомогнат оценката на високопроизводителни паралелни системи.

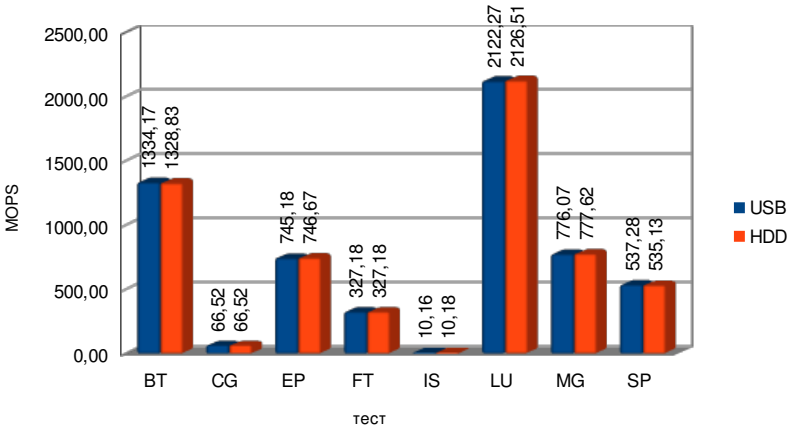
В параграф 4.2 е изследван ефектът върху производителността на клъстера при използване на USB flash drive като системно устройство. За целта са проведени няколко теста при използване на системно USB flash устройство и системен HDD. Най-напред е измерена производителността на дисковата подсистема при операции с файлове на един от компютрите с помощта на програмата bonnie++. Измерена е и производителността на системата при дискови операции със съпътстваща обработка – разархивиране.

Получените резултати ясно показват, че високата производителност на твърдия диск се отразява с приблизително 4 пъти по-високи резултати при всички операции с изключение на категориите „последователно четене на символи“ и „търсене“. Резултатите от теста за разархивиране, който е изпълнен с голям брой малки по обем файлове, показва приблизително равна производителност.

Налага се общият извод, че използването на системно USB flash устройство води до значително забавяне на дисковата подсистема при операции с големи файлове. Такива операции обаче не са типични за клъстер, предназначен за изчисления. При операциите „последователно четене на символи“, „търсене“, както

и при наличие на съпътстваща обработка от ЦП (разархивиране), производителността на USB flash устройство е съизмерима с производителността на HDD. Именно това са файловите операции, които преобладават в клъстер, предназначен за изчисления.

След като е известна разликата в производителността на дисковата подсистема, се проверява на практика разликата при реален изчислителен процес в клъстера с използване на NAS Parallel Benchmarks. Изброените тестове са проведени с ОС Arch Linux 64bit инсталирана на USB flash устройство, форматирано на файлова система EXT4. Тестовете са повторени под управление същата ОС и файлова система, инсталирана на твърдия диск.




фиг. 4.1.Сравнителни данни за изпълнение на NAS Parallel Benchmarks на 32 ядра

В Приложение 4 от дисертацията е налична пълната таблична информация от проведения експеримент. Освен MOPS са изнесени данните и за MOPS/CPU (милиони операции за секунда изпълнени на едно ядро) и общо време за завършване на теста.

Получените резултати (фиг. 4.1) показват, че при практическа експлоатация на клъстерната система не се наблюдава съществено изменение на производителността при използване на външно USB

flash устройство. Причините за този резултат могат да се търсят в характера на изпълняваните операции. На първо място самите паралелни програми не са големи по размер. Тяхната обща големина е под 500 KB и времето за зареждането им е пренебрежимо малко, на фона на времето за изпълнение на тестовете. Освен това при тестовете проведени с bonnie++ вече беше установено, че при четене на малки файлове или части от тях USB flash устройството постига скорост, сравнима със скоростта на твърд диск.

В параграф 4.3 е изследван ефектът върху производителността на клъстера при включване на виртуална машина. Разработената експериментална ВПС позволява заедно с изчислителния процес, да се използва ОС и приложния софтуер, инсталиран на твърдия диск на компютърните системи. В следващия експеримент е използвана виртуална машина, конфигурирана по следния начин:

Таблица 4.2. Конфигурация на виртуалната машина

| | |
|---|---|
| Централен процесор | Използва се реалният процесор, от който за виртуалната машина е заделено първото ядро. |
| Оперативна памет | 2 GB |
| BIOS | SeaBIOS версия 1.7 |
| Видеокарта | VGA compatible controller: Red Hat, Inc. |
| Звуков адаптер | Intel 82801AA AC97 |
| IDE контролер | IDE controller |
| Дисково устройство | Целият твърд диск е предоставен на виртуалната машина за монополно използване |
| Мрежова карта | Virtio network controller |

Guest OS, работеща във виртуална машина, се натоварва до степен да използва максималния заделен за нея ресурс. За целта се стартират няколко приложения, работещи едновременно:

• поточно видео – за натоварване на мрежовата връзка и видео подсистемата;

• glxgears – натоварване на видео подсистемата;

• копиране на папка с 150 000 файла – за натоварване на дисковата подсистема;

• Super Pi – програма, която изчислява π с точност до 32 милиона знака след десетичния разделител. Програмата се използва за натоварване на ЦП във ВМ до 100%.

При тези условия са изпълнени тестовете за измерване на производителността на клъстера с работеща ВМ. За сравнение са дадени резултатите без включена ВМ.

При проведения експеримент с bonnie++ производителността на дисковата подсистема намалява средно с 45%. Цялостното натоварване на ЦП и входно-изходната система във виртуалната машина оказва влияние върху резултатите в Host OS.

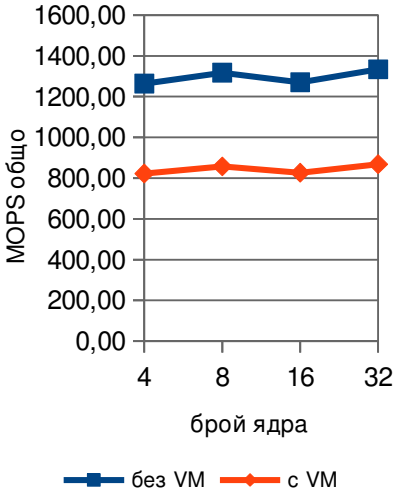
За времето на изпълнение на netperf във ВМ се наблюдава спад в скоростта на мрежовия трафик. За осъществяване на връзка между Host OS и Guest OS се използва мостов интерфейс, в който заявките на Host OS имат приоритет. Поради тази причина скоростта на мрежовата връзка в Host OS не се влияе съществено от работата на виртуалната машина.

Централните процесори са много по-бързи от контролера на паметта в компютърни системи. Програмите често са ограничени в бързодействието си от пропускателната способност на паметта на системата, а не от изчислителна производителност на процесора. За измерване на скоростта на оперативната памет се използва програмата „stream". Програмата е специално проектирана да работи с масиви от данни, по-големи от наличния кеш на която и да е система, така че резултатите да са показателни за изпълнението на приложения, обработващи вектори.

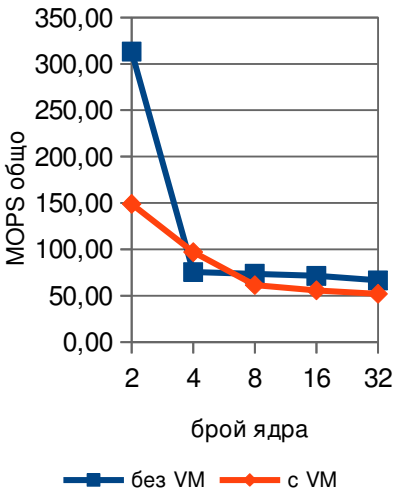
Табличните данни от изпълнението на „stream" са приведени в таблици 4.10 и 4.11, в точка 4.3.4 от дисертацията. Тестът е показателен за определяне на слабостите в системата. Централният процесор запазва висока производителност след стартирането на ВМ, докато достъпът до паметта показва сравнително ниски резултати. Също както при достъпа до

файловата система в точка 4.3.2, с увеличаване на тежестта на входно-изходните операции в тестовете се наблюдава влошаване на производителността.

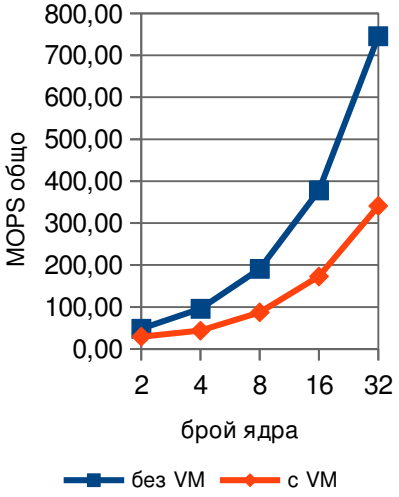
В точка 4.3.5 са представени резултатите от теста NAS Parallel Benchmarks, изпълнен с проблем от клас A последователно на 2, 4, 8, 16, 32 ядра. Пълните таблични резултати са налични в приложение 4 от дисертацията. От фиг. 4.2 до фиг. 4.9 са представени графично резултатите от изпълнението на всяка една от тестовите програми, включена в клас A на NAS Parallel Benchmarks. Тестовете са изпълнени с и без включена ВМ.



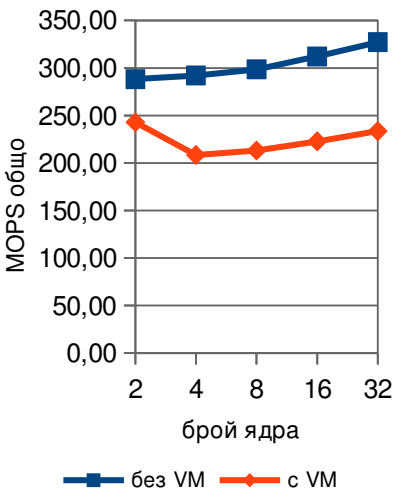

фиг. 4.2. NAS parallel benchmark - BT



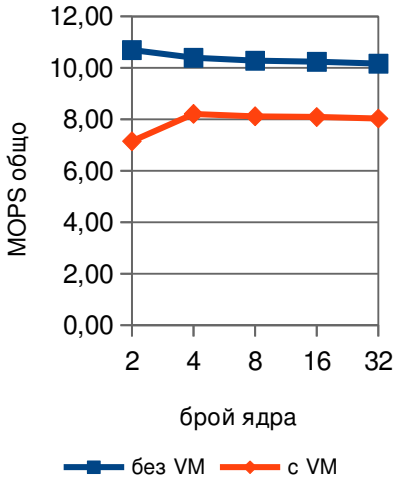

фиг. 4.3. NAS parallel benchmark - CG



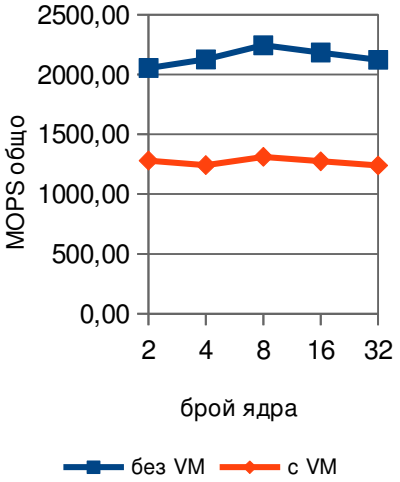

фиг. 4.4. NAS parallel benchmark - EP



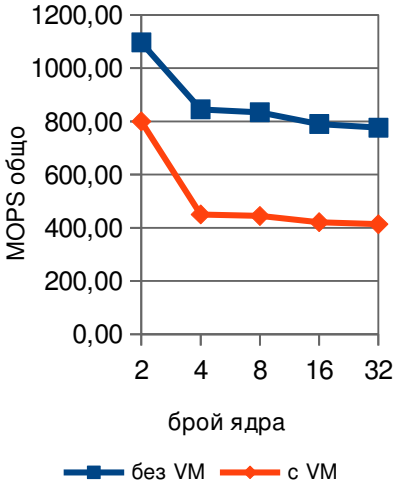

фиг. 4.5. NAS parallel benchmark - FT



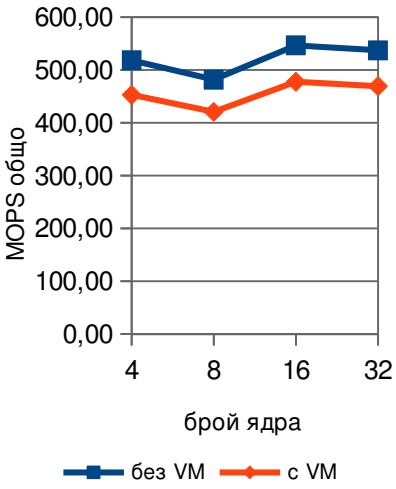

фиг. 4.6. NAS parallel benchmark - IS




фиг. 4.7. NAS parallel benchmark - LU




фиг. 4.8. NAS parallel benchmark - MG




фиг. 4.9. NAS parallel benchmark - SP

Натоварването на централния процесор достига 56% без стартирането на изчисления в клъстера. В резултатите, обаче, при нито една от изпълнените програми не се наблюдава понижение на производителността с 56%. Само при тестовете EP, MG се наблюдава рязко понижение на производителността при включване на виртуална машина. Тези тестове са специално разработени да извлекат максимума от възможностите на паралелна ВПС, но дори и при тях производителността е над очакваната. В останалите тестове, като например CG, IS и SP, производителността намалява в значително по-ниски граници.

В точка 4.3.5 е направен анализ, целящ да обясни получените резултати. Той се базира на алгоритмите, използвани в програмите от NAS Parallel Benchmarks, от гледна точка на декомпозицията на задачите, трансфера на данни между възлите в клъстера и използваните блокиращи функции. В точка 4.3.6 е разгледано влиянието на мениджъра на процеси Completely Fair Scheduler (CFS) върху оползотворяването на ЦП при изпълнение на

паралелни програми.

В параграф 4.4 е изследвана производителността на Guest OS. Експерименталните резултати показват производителност при изпълнение на операции във ВМ спрямо физическият компютър, както следва:

- 29%-36% - запис на информация на твърдия диск;
- 42%-86% - четене на данни от твърдия диск. Високите стойности са постигнати при четене на блокове;
- 88% - разархивиране.

Може да се обобщи, че при работа с ОС във ВМ се наблюдава спад в производителността спрямо изпълнение на физическата компютърна система. Въпреки това системата запазва работоспособност и остава подходяща за извършване на редица традиционни дейности:

- Работа с офис пакет – текстообработка, електронни таблици, презентации;
- Бизнес приложения - счетоводен, складов софтуер, юридически софтуер;
- Работа със среди за програмиране;
- Използване на интернет;

С помощта на VT-x и AMD-V във ВМ се постига производителност на ЦП, сравнима с изпълнението на физическия компютър, но общото бързодействие на компютърна система зависи и от ПУ. Във ВМ са използвани паравиртуални ПУ. В резултат виртуалният компютър е напълно различен от физическия. Някои от компонентите му не само имат по-ниска производителност, но се характеризират и с намалена функционалност. Например звуковият адаптер в тестовата система е 8-канален, но звуковият адаптер във ВМ е само 4-канален.

Налице е и по-комплексен проблем с видеокартата. Използваната във виртуалната машина видеокарта е със значително по-ниски характеристики от реалната. Занижена е цялостно производителността на видеоподсистемата, а част от функциите за триизмерна обработка не са налични.

Всички тези функционални ограничения налагат нуждата да се дефинират „проблемните“ области, в които използването на ВМ е неподходящо решение:

• При необходимост да се работи със специализирани мултимедийни продукти, изискващи висока производителност от видеокартата и звуковата карта;

• При необходимост да се работи с приложения, използващи активно поддръжка на 3D функции от видеокартата.

При управлението на периферни устройства е възможно да се използват технологиите RVI и IOMMU, за да се осигури на ВМ директен достъп (passthrough) и до ПУ. Това е оптимален подход по отношение на производителността в Guest OS, но все още слабото разпространение на ЦП, поддържащи RVI и IOMMU, не позволява използването на тези технологии. Друг недостатък е, че при използването на passthrough е необходима ръчна конфигурация на всеки компютър, за всяко устройство, което използва технологията. Експерименталната ВПС е преносима и затова е препоръчително да може да функционира на разнообразни компютърни системи, с минимална намеса от страна на потребителя. Поради тези причини виртуалната машина, по-подразбиране, не използва passthrough. Въпреки това в системата са налични всички необходими инструменти за използването на технологията.

## Изводи от проведените експерименти

1. Резултатите от тестовете на дисковата подсистема, с използване на системно USB flash устройство, показаха намаление на производителността при операциите „последователен запис на блокове“, „последователен запис на символи“ и „последователно четене на блокове“, които не са типични за клъстери на високопроизводителни изчисления. Не е констатирано намаляване на производителността на експерименталната високопроизводителна система при изпълнение на паралелни изчислителни процеси.

2. При проведения експеримент с Nas paralell benchmarks в

параграф 4.3 след включване на виртуална машина натоварването на централния процесор достига 56%, докато оставащото процесорно време за клъстера е 44%. В зависимост от тестовата програма резултатите показват производителност в интервала 47-87%, спрямо резултатите без използване на виртуална машина. Наблюдава се повишено оползотворяване на централния процесор, изразяващо се в резултати надхвърлящи разполагаемия ресурс.

3. Установени са групите софтуер, които могат да се използват във виртуална машина.

4. Определени са проблемни групи софтуерни продукти, които не са подходящи за изпълнение във виртуална машина.

5. Предложени са мерки за увеличаване на производителността на Guest OS, когато е необходимо да се използват „проблемни“ софтуерни продукти.

# III. ЗАКЛЮЧЕНИЕ

## Основни приноси в дисертационното изследване

1. Изследвани са проблемите, свързани с интегрирането на различни операционни системи при създаване на високопроизводителни изчислителни системи и е определено влиянието на ядрата на операционните системи върху осигуряване на съвместимостта.

2. Изследвани са и са дефинирани факторите, които влияят бързодействието при интегриране на различни операционни системи, за нуждите на разработената система.

3. Дефинирани са предпочитаните архитектурни решения за изграждане на високо ефективни изчислителни системи, използващи услугите на интегрирани ОС.

4. Разработен е метод за създаване на паралелни високопроизводителни изчислителни системи, включваща и набор от алгоритми за решаване на този проблем.

5. Разработен е метод за оценка на решенията за интегриране на операционни системи и високопроизводителни изчислителни системи. Показано е, че предложените решения позволяват да се

постигне подобряване на икономическата ефективност без загуба на производителност, с доказана повишена степен на оползотворяване на процесорите на системата.

## Публикации по темата на дисертационния труд

1. К. Цветков, Д. Сърмов, Анализ на основните средства за осигуряване на среда, която позволява изпълнение на софтуер, предназначен за различни операционни системи, Военно-научен форум, България, Велико Търново, НВУ „Васил Левски", 2006.

2. К. Цветков, Д. Сърмов, Причини за пораждане на несъвместимост между операционните системи, Военно-научен форум, България, Велико Търново, НВУ „Васил Левски", 2006.

3. Konstantin Tscvetkov, Delian H. Sarmov, Software solutions for insurance of an environment, which allows an execution of software designed for different Operating Systems, Romania, Bucharest, MTA review, Volume XVII, No. 1, Pages 79-86, Jun. 2007.

4. К. Цветков, Д. Сърмов, Софтуерни средства за осигуряване на среда позволяваща изпълнение на софтуер, предназначен за различни операционни системи, Сборник научни трудове, Шумен, 2007.

5. Константин Цветков, Делян Сърмов, Модел на виртуална машина, управляваща операционни системи, Телеком 2007, Варна, 2007.

6. Делян Сърмов, Примерна реализация на преносима високопроизводителна паралелна изчислителна система, MATTEX, 2012 – под печат.

7. Delian H. Sarmov, Konstantin Tscvetkov, A case for high performance computing with using isolated computational resource CY-ICER, Procedia-Social and Behavioral Sciences, Ataturk Teacher Training Academy, Lefkosa, North Cyprus, February 2013 – под печат.

## Цитирана литература в автореферата

1. Баденко В. Л., Высокопроизводительные вычисления, Санкт-Петербург Издательство Политехнического университета, 2010.
2. Михайлов Б.М., Р.Ф. Халабия, Классификация и организация вычислительных систем, 2010.
3. AMD, AMD64 Virtualization Codenamed “Pacifica” Technology: Secure Virtual Machine Architecture Reference Manual, 2005.
4. Bhatia N., Performance Evaluation of AMD RVI Hardware Assist, Vmware, 2008-2009
5. Liu J., W. Huang, B. Abali, D. K. Panda, High Performance VMM-Bypass I/O in Virtual Machines, Proceedings of 2006 USENIX Annual Technical Conference, June 2006.
6. Liu J., W. Huang, B. Abali, D. K. Panda, High Performance VMM-Bypass I/O in Virtual Machines, Proceedings of 2006 USENIX Annual Technical Conference, June 2006.
7. Warnke R., Ritzau Th., Qemu-kvm & libvirt, Amazon Media EU S.à r.l., 2010.
8. Squyres J. M., The Architecture of Open Source Applications, 2005.

**Konstantin Preslavsky University of Shumen**

**Faculty of Mathematics and Computer Science**

**Author: Delyan Sarmov**

# *Research of integration processes in modern operating systems*

## Summary

There are many problems whose solutions set high demands on system performance. Traditionally cost-effective systems associated with the implementation of cluster systems. Cluster calculations allow to use existing computer systems to build a high performance system (HPS). There are many software products that have been developed in which as node of cluster an operating system (OS) is used installed in virtual machines. In these solutions flexibility, security and isolation are achieved, but at the cost of some loss of performance.

This thesis presents a model of the system running on an external device that provides collaboration two operating systems. One is intended for inclusion as a node in HPC, while the second is used to implement the existing hard disk Operating System.

Based on the proposed model experimental system is implemented using a virtual machine. HPC nodes are implemented in Host OS, while the available computer software remains accessible via the Guest OS. In this way a HPS is realized using free hardware resource of computer systems. Current implementation is based on the Kernel Virtual Machine (KVM) and QEMU environment used under management of GNU/Linux OS. However, many of ideas are readily applicable to other VM environments.

The applicability of the developed experimental system was verified by a series of experiments.