

Министерство образования и науки Российской Федерации

Федеральное государственное образовательное учреждение высшего
образования

”Алтайский государственный технический университет им. И. И. Ползунова”

Факультет

информационных технологий

наименование подразделения

Кафедра

прикладной математики

наименование кафедры

Отчет защищен с оценкой _____
“ ” _____ 2017 г.

Руководитель от вуза

_____/ Старолетов С.М.
подпись *Ф. И. О.*

ОТЧЕТ

О научно-исследовательской работе

Разработка микросервиса под управлением операционной системы IncludeOS

общая формулировка задания

В Soft-logic
наименование организации

Студент гр. 8ПИ-51 _____ Борисов В.В. _____
индекс группы *подпись* *Ф. И. О.*

Руководитель от организации _____
подпись *Ф. И. О.*

Руководитель от университета _____ Старолетов С.М.
подпись *Ф. И. О.*

2017 г.

Задание и календарный план практики (научно-исследовательской работы)

ФГБОУ ВО «Алтайский государственный технический университет
им. И. И. Ползунова»

Кафедра прикладной математики

УТВЕРЖДАЮ

Зав. кафедрой _____ С.А. Кантор
“ ” _____ 2017 г.

ЗАДАНИЕ

По научно-исследовательской работе

студенту группы 8ПИ-51 _____ Борисова В.В.

фамилия, имя, отчество

09.04.04 «Программная инженерия»

код и наименование направления

База практики(научно-исследовательской работы) _____ Soft-logic

наименование организации

Способ проведения практики _____ стационарная

стационарная, выездная, другие

Срок практики с 09.01.17 г. по 19.03.17

Разработка микросервиса под управлением операционной системы IncludeOS

общая формулировка задания

Календарный план практики

Наименование задач (мероприятий), составляющих задание	Дата выполнения задачи (мероприятия)	Подпись руководителя практики от организации
Постановка задачи	09.01.17 г.	
Обзор предметной области	09.01.17 г. - 19.01.17 г.	
Обзор имеющихся разработок по теме, их преимуществ и недостатков	20.01.17 г. - 03.02.17 г.	
Разработка web сервиса	04.02.17 г. - 19.03.17 г.	

Срок представления работы к защите _____ 20.03.17 г.

Руководитель практики от вуза

_____ Старолетов С.М., доцент каф. ПМ

Содержание

Задание и календарный план практики (научно-исследовательской работы)...	2
Введение.....	4
1 Описание предметной области.....	6
1.1 Микросервис.....	6
1.2 Гипервизор.....	7
1.3 IncludeOS.....	8
2 Обзор конкурирующих систем.....	11
2.1 Виртуализация на основе продуктов VMware.....	11
2.2 Виртуализация на основе продуктов Xen.....	13
2.3 Виртуализация на основе продуктов Hyper-V.....	15
3 Исследование.....	17
3.1 Клиент.....	17
3.2 Сервер.....	18
3.3 Контроллер.....	20
3.4 Консоль администратора.....	22
3.5 Виртуальная машина.....	23
Постановка задачи.....	26
Заключение.....	27
Список использованных источников.....	28

Введение

Местом, для прохождения практики была выбрана компания «Soft-logic»[1]. Компания Soft-logic основана в 2008 году. Специализация компании - разработка ПО для платежных систем, банков и расчетных центров.

Миссия компании - внедрение инновационных платежных решений в различных отраслях промышленности и финансового сектора, разработка и внедрение комплексных решений: "Процессинговый центр Pay-logic", SmartKeeper, "Расчетный центр Pay-logic", Фискальный сервер, а также решений для оплаты товаров и услуг в сети Интернет: Платформа мобильной коммерции, мобильные клиенты Android и IOS.

В настоящее время на решениях Soft-logic работает 54 коммерческих проекта в России, а также странах ближнего и дальнего зарубежья.

Флагманским продуктом компании является платежная платформа «Процессинговый центр Pay-logic». Данный программный комплекс предназначен для организации приема платежей в пользу любых поставщиков услуг через платежные терминалы, персональный компьютер, pos-терминал, мобильные устройства и интернет.

Сегодня Soft-logic — это самостоятельная, динамично развивающаяся компания, специализирующаяся на разработке комплексных платежных решений.

Большинство специалистов работают в этой сфере с начала 2000 годов, с момента зарождения терминального бизнеса в РФ.

Накопленные знания и опыт в области платежных решений, штат из высококвалифицированных специалистов, а также репутация разработчика качественных программных решений позволяют говорить о Soft-logic как о надежном партнере, предоставляющем качественные решения в финансовой сфере.

Во время прохождения практики была поставлена задача запуска и

интегрирования в тестовые окружение web сервисов, под управлением одномодульной операционных системы IncludeOS [2]. Данные web сервисы должны принимать get, set и put запросы от клиентов мобильного платёжного приложения под управлением Android и Ios, производить обработку полученных данных и возвращать результат на клиенты.

IncludeOS предоставляет минимальное необходимое самодостаточное окружение, которое взаимодействует непосредственно с гипервизором и предоставляет загрузчик, ядро, минимальный набор библиотек и модулей, достаточный для выполнения кода на языке C++, написанный с использованием стандартной библиотеки классов. Окружение компонуется с предназначенным для выполнения приложением и оформляется в виде загрузочного образа виртуальной машины, образуя готовый облачный сервис.

1 Описание предметной области

1.1 Микросервис

Во многих IT компаниях, для обработки однотипной информации в больших объёмах используют микросервисы [3]. Архитектурный стиль микросервисов — это подход, при котором единое приложение строится как набор небольших сервисов, каждый из которых работает в собственном процессе и коммуницирует с остальными используя легковесные механизмы, как правило HTTP. Эти сервисы построены вокруг бизнес-потребностей и развертываются независимо с использованием полностью автоматизированной среды. Существует абсолютный минимум централизованного управления этими сервисами [4]. Сами по себе эти сервисы могут быть написаны на разных языках и использовать разные технологии хранения данных.

Сервис [5] представляет из себя программный продукт, выполняющий узконаправленные действия, такие как: конвертирование аудио файла из одного формата в другой, поиск на изображении лиц, построение панорамного изображения из нескольких, выполнение математических расчётов и т.д. Сервис запускается в реальной операционной системе (Windows или Unix подобных) на виртуальной машине. Для запуска, контролирования виртуальных машин, а также масштабирования всей системы используют гипервизор.

1.2 Гипервизор

Гипервизор [6] или монитор виртуальных машин — это, как правило комплекс программ обеспечивающий параллельное, одновременное выполнение нескольких операционных систем на одном и том же хост – компьютере.

Гипервизор в планируемой системе будет отвечать за запуск, остановку и контроль нагрузки виртуальных машин, например: если нагрузка падает и несколько виртуальных машин простаивают, то их можно отключить и наоборот, если нагрузка возрастает и данное количество виртуальных машин не справляется, то запускаются ещё n-ое количество необходимых сервисов.

Сервисы на виртуальных машинах запускаются, как правило, под управлением операционных систем семейства Windows или Unix-подобных. Windows или Unix-подобные операционные системы на виртуальных машинах занимают достаточно большое количество ресурсов реального хост – компьютера, часто превосходящие затраты на выполнение самого сервиса, т. к. необходимо поддерживать работу самой операционной системы в которой запущен сервис, а именно, поддерживать графический интерфейс [7] (для Windows), поддерживать работу неиспользуемых драйверов (видеокарты, драйвер принтера, звуковой карты и т. д.).

1.3 IncludeOS

В 2016 году на конференции CppCon [8] был представлен рабочий alpha прототип проекта одномодульной микро операционной системы IncludeOS [9]. IncludeOS предоставляет минимальное необходимое самодостаточное окружение, которое взаимодействует непосредственно с гипервизором и предоставляет загрузчик, ядро системы, минимальный набор библиотек, модулей и драйверов, достаточный для выполнения кода на языке C++, написанный с использованием стандартной библиотеки классов [10]. Окружение компонуется с предназначенным для выполнения приложением и оформляется в виде загрузочного образа виртуальной машины, образуя готовый облачный сервис. Из систем виртуализации, в которых могут работать подобные окружения, поддерживаются KVM/Linux, VirtualBox и QEMU. Схематичное представление построение загрузочного образа виртуальной машины с готовым микросервисом изображён на рисунке 1.

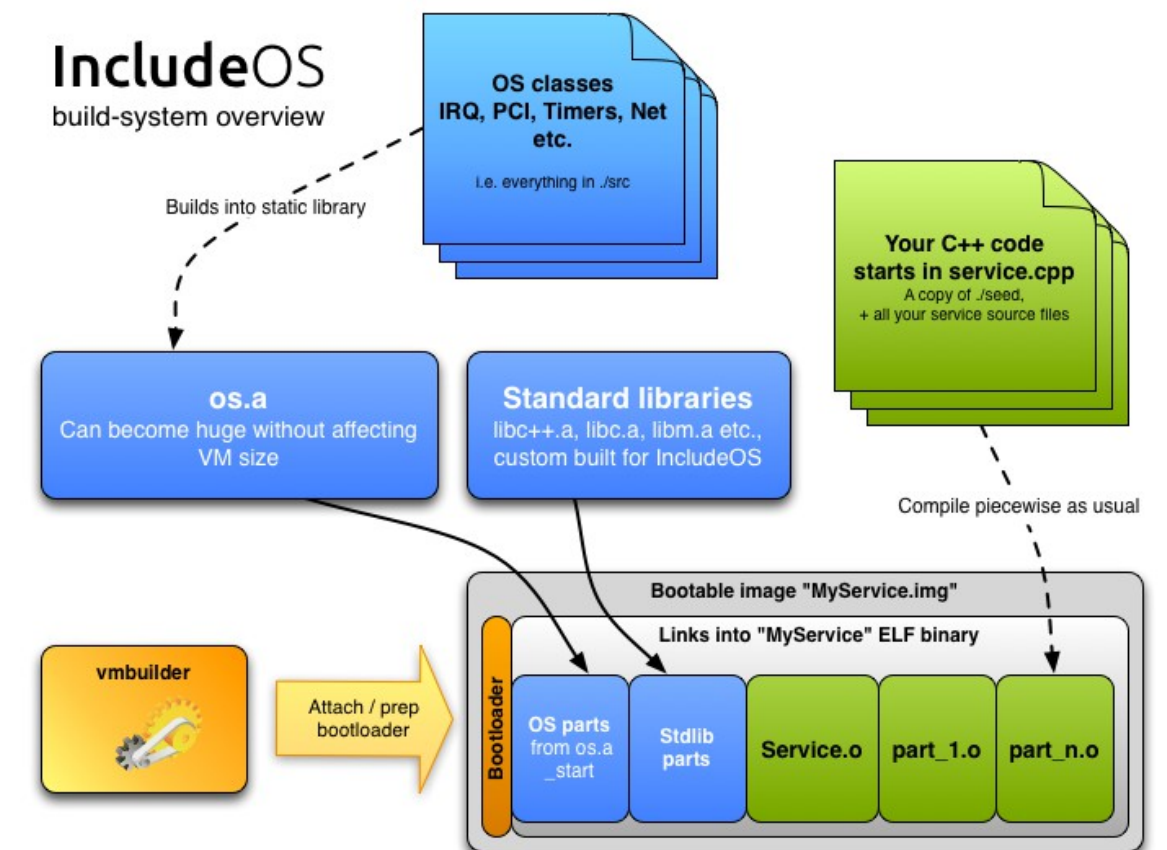


Рисунок 1.1 - Построение образа виртуальной машины

По затратам машинных ресурсов IncludeOS можно сравнить с встраиваемой операционной системой.

Встраиваемые операционные системы [XXX1] – операционные системы (ОС), предназначенные не для запуска на виртуальных машинах, а для управления специализированными устройствами (промышленные контроллеры, радиоаппаратура и т.д.) и вследствие этого способные работать в условиях ограниченных ресурсов (малые объёмы памяти, недостаток вычислительных мощностей и т.п.) и в необслуживаемом режиме. Характерными особенностями встраиваемых ОС являются модульная структура, компактность, производительность, масштабируемость и повышенная отказоустойчивость.

Основным отличием IncludeOS от встраиваемой операционной системой является то, что IncludeOS не предназначена для управления специализированными устройствами, хотя в её ядре есть модули, позволяющие взаимодействовать с USB и COM портами.

IncludeOS, в основном, разрабатывается для выполнения web сервисов. Суммарный размер библиотек и компонентов скомпилированной операционной системы составляет около 1 Мб [11]. В связи с тем что IncludeOS в большей степени не является встраиваемой операционной системой и с тем, что гипервизоры в подавляющем большинстве случаев используют виртуальные машины под управлением операционных систем семейства Windows или Unix-подобных проведём сравнение использования машинных ресурсов IncludeOS и операционной системы Ubuntu. На тестах с операционной системой Ubuntu, IncludeOS показал значительно меньшее потребление ресурсов реального хост – компьютера, а именно: занимаемое пространство на жёстком диске меньше в 300 раз; использование оперативной памяти меньше в 280 раз [12]; загрузка системы происходит менее чем за 0,5 секунды, для сравнения Ubuntu загружается в среднем за 10 секунд; нагрузка на CPU при работе экспериментального DNS - сервера, построенного на базе IncludeOS, оценивается в 5-20% по сравнению с

запуском того же исполняемого файла в Ubuntu, всё это объясняется тем, что в IncludeOS есть только необходимые элементы операционной системы для работы сервиса. Сравнительная диаграмма используемых ресурсов хост – компьютера для Ubuntu и IncludeOS изображена на рисунке 2.

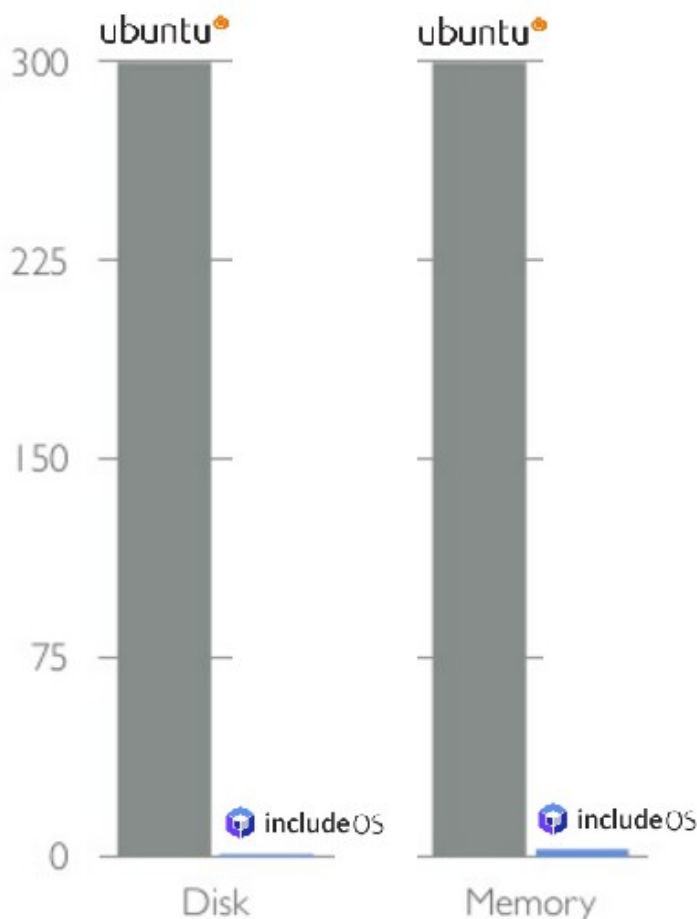


Рисунок 1.2 - Используемые ресурсы Ubuntu и IncludeOS

В связи с этим использование IncludeOS в качестве операционной системы для виртуальной машины в настоящее время в разы эффективней, чем с любой другой операционной системы, т.к. на одних и тех-же аппаратных мощностях можно запускать в разы больше виртуальных машин.

Так как IncludeOS относительно новый проект и на данный момент находится в состоянии разработки, то поддерживаются только следующие системы виртуализации: KVM/Linux, VirtualBox и QEMU, а это значит, что запустить виртуальные машины под управлением данной операционной системы, для работы сервисов на наиболее известных гипервизорах, таких как VMware ESX Server [13], XenServer [14], Citrix [15] и других не

получится. Исходя из изложенного, отказываться от преимуществ IncludeOS будет ошибкой, в связи с этим было принято решение о разработке собственного менеджера виртуальных машин с возможностью запуска, остановкой, просмотра состояния нагрузки виртуальных машин и т. д.

Планируется разработка программного комплекса, в который будет входить: клиентский компонент системы, для формирования данных для обработки сервисами (передача данных для обработки и получение результата обработки); сервер для мониторинга виртуальных машин, который будет в зависимости от загруженности виртуальных машин принимать решения о запуске или остановки сервисов, а также будет посредником для передачи данных для обработки от клиента сервисам; контроллер будет установлен на хост-машину и предназначен для запуска, остановки, сбора информации о загрузке виртуальной машины; консоль администратора, будет устанавливаться параллельно с клиентским компонентом для просмотра состояния загруженности виртуальных машин.

2 Обзор конкурирующих систем.

2.1 Виртуализация на основе продуктов VMware.

ESX и ESXi являются встроенными гипервизорами и ставятся непосредственно «на железо» [16], то есть при установке не требуют наличия на машине установленной операционной системы.

ESX Server представляет собой гипервизор типа 1 [17], который создает логические пулы системных ресурсов, позволяя множеству виртуальных машин разделять одни и те же физические ресурсы.

ESX Server - это операционная система, которая функционирует как гипервизор и работает непосредственно на оборудовании сервера. ESX Server добавляет уровень виртуализации между аппаратной частью системы и виртуальными машинами, превращая оборудование системы в пул логических вычислительных ресурсов, которые ESX Server может динамически выделять любой гостевой операционной системе. Операционные системы, работающие в виртуальных машинах, взаимодействуют с виртуальными ресурсами, как если бы это были физические ресурсы.

Ключевые компоненты архитектуры ESX Server:

1. Уровень виртуализации ESX Server: отделяет основные физические ресурсы от виртуальных машин
2. Менеджер ресурсов: создает виртуальные машины и обеспечивает их ресурсами процессора, памяти, сети и дисковой подсистемы
3. Служебная консоль: управляет установкой, настройкой, администрированием, устранением неисправностей и техническим обслуживанием ESX Server
4. Компоненты аппаратного интерфейса, в том числе драйверы устройств.

Возможности [18]:

1. ESX Server использует механизм пропорционального распределения

ресурсов процессоров, памяти и дисков, когда несколько виртуальных машин претендуют на одни и те же ресурсы

2. ESX Server может выделять емкость процессора на основе разделения времени, предотвращая возможность монополизации ресурсов процессора какой-либо виртуальной машиной
3. ESX Server выделяет память в зависимости от нагрузки виртуальной машины и заданного минимума. Например, если виртуальной машине недостаточно памяти, ESX Server может занять память у одной виртуальной машины, передать ее другой виртуальной машине, и в случае необходимости вернуть эту память первоначальной виртуальной машине
4. ESX Server управляет пропускной способностью сети с формированием сетевого трафика. Разделение ресурсов сети осуществляется с помощью выделения маркеров или в зависимости от потребления, исходя из средней или максимальной потребности в пропускной способности виртуальной машины.

Плюсы [19]:

1. С технической точки зрения самый продвинутый гипервизор
2. Бесплатен (можно скачать с сайта Vmware)
3. Поддерживает множество ОС внутри себя (Windows, Linux, BSD, Solaris, и т.д.)
4. Легко установить и настроить.

Минусы:

1. Продвинутые инструменты администрирования платные
2. Может установиться только на ограниченное количество серверов
3. Занимает большое количество аппаратных ресурсов
4. Не поддерживает IncludeOS.

2.2 Виртуализация на основе продуктов Xen.

Xen [20] представляет собой гипервизор типа 1, который создает логические пулы системных ресурсов, позволяя множеству виртуальных машин разделять одни и те же физические ресурсы.

Xen – это гипервизор, который работает непосредственно на оборудовании системы. Xen добавляет уровень виртуализации между аппаратной частью системы и виртуальными машинами, превращая оборудование системы в пул логических вычислительных ресурсов, которые Xen может динамически выделять любой гостевой операционной системе. Операционные системы, работающие в виртуальных машинах, взаимодействуют с виртуальными ресурсами, как если бы это были физические ресурсы.

Xen основан на паравиртуализации [21]; он требует внесения в гостевые операционные системы изменений, направленных на поддержку операционной среды Xen. Тем не менее, пользовательские приложения и библиотеки никакой модификации не требуют.

Внесение изменений в операционную систему необходимо для того, чтобы:

1. Xen мог заменить операционную систему в качестве наиболее привилегированной программы
2. Xen мог использовать более эффективные интерфейсы (например, виртуальные блочные устройства и виртуальные сетевые интерфейсы) для эмуляции устройств - это повышает производительность.

Xen (с помощью стека управления) поддерживает миграцию гостевых виртуальных машин по сети. Миграция паравиртуальных машин поддерживается с версии Xen 2, а HVM – с версии 3. Миграция может происходить с выключением гостевой системы, или прямо в процессе работы, так называемая «живая» миграция без потери доступности.

Необходимо, чтобы оба физические сервера Xen видели одно и то же хранилище, на котором находятся данные виртуальной машины. Общее

хранилище может быть организовано на основе различных технологий SAN или NAS, например Fibre Channel, iSCSI или DRBD.

Четвертое поколение продуктов виртуализации компании XenSource [22] на основе гипервизора Xen включает в себя три версии платформы:

XenExpress - бесплатное стартовое издание системы виртуализации, включающее в себя возможность размещения нескольких виртуальных машин в пределах одного физического сервера. Эта версия является скорее ознакомительной и хорошо подходит для домашних пользователей и небольших компаний, планирующих внедрение виртуализации.

XenServer - издание платформы для сектора SMB (Small and Medium Business), обеспечивающее решение основных задач по консолидации виртуальных серверов на нескольких физических хостах.

XenEnterprise - наиболее полная версия платформы виртуализации, включающая в себя, помимо возможностей XenServer, также и необходимые средства по агрегации ресурсов, распределению нагрузки, живой миграции и обеспечению высокой доступности.

Плюсы:

1. Поддерживает множество ОС внутри себя
2. Бесплатен
3. Поддерживает достаточно большое количество серверов.

Минусы:

1. Продвинутые инструменты администрирования платные
2. Занимает большое количество аппаратных ресурсов
3. Может установиться только на ограниченное количество серверов
4. Не поддерживает IncludeOS.

2.3 Виртуализация на основе продуктов Hyper-V.

Hyper-V существует в двух вариантах [23]:

1. Как отдельный продукт Microsoft Hyper-V Server
2. Как роль Windows Server 2012, Windows Server 2008 R2, Windows Server 2008 и x64 битная версия Windows 8 Pro.

Отдельная версия Hyper-V Server является бесплатной. Является базовым («Server Core») вариантом Windows Server 2008, то есть включает в себя полную функциональность Hyper-V; прочие роли Windows 2008 Server отключены, также лимитированы службы Windows. Бесплатная 64-битная Core-версия Hyper-V ограничена интерфейсом командной строки (CLI PowerShell), где конфигурация текущей ОС, физического аппаратного и программного оборудования выполняется при помощи команд оболочки. Новое меню интерфейса управления позволяет выполнить простую первичную конфигурацию, а некоторые свободно распространяемые скрипты расширяют данную концепцию.

Администрирование и конфигурирование виртуального сервера осуществляется при помощи ПО, установленного на ПК под управлением Windows Vista, Windows 7 или Windows 2008 Server с установленным дополнением для администрирования Hyper-V из MMC [24]. Другим вариантом администрирования/конфигурирования сервера Windows 2008 Core является использование удаленной Windows или Windows Server при перенаправлении (некоторой) консоли управления (MMC), указывающей на Core Server.

В родительском разделе, в пространстве пользователя, имеются:

1. WMI-провайдеры – управление виртуальными машинами локально и удаленно
2. Сервис управления виртуальными машинами (VMMS) – управляет виртуальными машинами
3. Рабочие процессы виртуальных машин (VMWP) – процессы, в которых

выполняются все действия виртуальных машин – обращение к виртуальным процессорам, устройствам, и т.д

В пространстве ядра родительского раздела выполняются:

1. Драйвер виртуальной инфраструктуры (VID) – осуществляет управление разделами, а так же процессорами и памятью виртуальных машин
2. Провайдер сервисов виртуализации (VSP) – предоставляет специфические функции виртуальных устройств (так называемые «синтетические устройства) посредством VMBus и при наличии интеграционных компонент на стороне гостевой ОС
3. Шина виртуальных устройств (VMBus) – осуществляет обмен информацией между виртуальными устройствами внутри дочерних разделов и родительским разделом
4. Драйверы устройств – только родительский раздел имеет прямой доступ к аппаратным устройствам, и потому именно внутри него работают все драйверы
5. Windows Kernel – собственно ядро хостовой ОС.

Плюсы:

1. Бесплатный
2. Хорошо подходит для виртуализации ОС от Microsoft
3. Большинство продуктов Microsoft поддерживают работу в виртуальной среде Hyper-V
4. Лёгкий в установке и настройке
5. Может установиться на любой сервер.

Минусы:

1. Плохо подходит для виртуализации ОС не от Microsoft (т.е. не Windows)
2. Продвинутые инструменты администрирования платные
3. Занимает большое количество аппаратных ресурсов
4. Не поддерживает IncludeOS.

3 Архитектура IncludeOS

3.1 Принцип нулевых издержек

Для любого сервиса, предназначенного для масштабного развертывания на нескольких виртуальных машинах важно, чтобы каждая такая машина использовали минимальное количество ресурсов [XXX2].

В отличие от классических операционных систем, которые включают в себя как можно больше функций, IncludeOS стремится к минимальному набору используемых функций по умолчанию в том смысле, что служба не должна быть включена по умолчанию, если она явно не используется. Это соответствует принципу нулевых издержек, например, C++: «Вы не должны платить за то, что не используете» [XXX3].

3.2 Статически линкующиеся библиотеки и GCC-toolchain

Статически линкующиеся библиотеки - это архивы объектных файлов, который подключается к программе во время линковки. Статические библиотеки делают программу более автономной: программа, скомпонованная со статической библиотекой может запускаться на любом компьютере, не требуя наличия этой библиотеки (она уже включена в бинарные файлы) [XXX4].

GCC toolchain — набор созданных в рамках проекта GNU пакетов программ, необходимых для компиляции и генерации выполняемого кода из исходных текстов. Являются стандартным средством разработки программ и ядра ОС Linux [XXX5].

Используется механизм для извлечения только тех библиотек, которые требуется от операционной системы. Данный механизм установлен по умолчанию во всех современных линкерах. Каждая часть операционной системы компилируется в объектный файл, например, `ip4.o`, `udp.o`, `pci_device.o` и т.д., которые затем объединяются в виде статической

библиотеки `os.a`. Линковщик автоматически выбирает необходимые библиотеки и только они упакуются в итоговый бинарный файл. Чтобы облегчить этот процесс был создан кастомный GCC toolchain.

IncludeOS не имеет программного загрузчика, поэтому нет классической `main`-функции с параметрами и возвращаемого значения. Вместо этого, используется класс `Service`, а пользователь должен реализовать функцию `Service::start`, которая будет вызвана после завершения инициализации операционной системы [XXX2].

3.3 Стандарт библиотек

Новые библиотеки от RedHat были выбраны в качестве стандарта реализации библиотек на языке C, прежде всего, потому что они имеют достаточно малый размер, и разработаны чтобы полагаться только на несколько системных вызовов, а также они компилируются в статически линкуемые библиотеки. Таким образом, компоновщик будет включать в итоговый бинарный файл только те части стандартной библиотеки, которые будут использоваться или операционной системой или выполняемым сервисом.

Стандартная библиотека шаблонов (STL) C++ является довольно крупной и сложной. Так как контейнеры STL используют исключения, в IncludeOS они не используются внутри ядра, вместо этого используется библиотека шаблонов от Electronic Arts, а именно — EASTL [XXX6]. Хотя данная реализация и содержит наиболее важные части STL, такие как `string`, `streams`, `vector` и `map`, этого не достаточно, так что некоторые компоненты были реализованы, а некоторые все еще находятся в стадии разработки. IncludeOS версия этой библиотеки будет доступна для исполняемых сервисов (т.е. доступна в пространстве сервиса). Будущие реализации IncludeOS будут включать в себя порт полнофункциональной реализации, такой как GCC `libstdc++` [XXX2].

3.4 Сетевой драйвер Virtio

Сетевой драйвер Virtio — это технология улучшающая производительность виртуальных машин под управлением KVM. Данные драйвера реализуют «паравиртуализацию». В этом режиме работа устройства не полностью эмулируется гипервизором. Драйвер устройства в виртуальной машине знает о том, что он работает не с настоящим устройством, и взаимодействует с гипервизором, что обеспечивает большую производительность [XXX7].

IncludeOS имеет только один драйвер устройства, а именно драйвер VirtioNet. Virtio 1.0 является стандартом Oasis [XXX8], но ни один из гипервизоров, используемых во время разработки IncludeOS, не поддерживал ни одной из новых функций. Поэтому драйвер в настоящее время реализует только устаревшую функциональность Virtio, но разработка была сделана с будущей поддержкой Virtio 1.0. Текущая реализация использует шину PCI и не включила MSI-х (Message signaled interrupts) [XXX2].

3.5 Асинхронный ввод-вывод и IRQ

В настоящее время, все обработчики запросов на прерывание (IRQ) в IncludeOS только обновляют счетчик, и откладывают дальнейшее обращение к основной работе приложения, когда есть время. Это избавляет от необходимости переключения контекста, также устраняя связанные с параллелизмом проблемы, такие как гонка данных. Процессор занят, поскольку все операции ввода-вывода асинхронны, поэтому блокирование не происходит. Это поощряет модель программирования на основе обратного вызова (Callback) [XXX2].

Callback в программировании — передача исполняемого кода в качестве одного из параметров другого кода. Обратный вызов позволяет в функции исполнять код, который задаётся в аргументах при её вызове [XXX9].

4 Постановка задачи

Целью данной работы является разработка программного комплекса, в который будет входить: клиентский компонент системы для формирования запросов с данными для обработки; сервер для распределения нагрузки между виртуальными машинами; контроллер для управление виртуальных машин на хост-компьютере; консоль администратора для просмотра информации о количестве и загруженности виртуальных машин; дополнения сервиса виртуальной машины.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Проектирование механизмов взаимодействия компонентов системы.
 - 1.1. Проектирование протокола запроса на исполнение задачи.
 - 1.2. Проектирование протокола запроса на выдачу информации о состоянии виртуальной машины.
 - 1.3. Проектирование протокола управления состоянием виртуальной машины.
 - 1.4. Проектирование протокола запуска и остановки экземпляров виртуальных машин на хост-компьютере.
2. Разработка серверного компонента.
 - 2.1. Разработка подсистемы перенаправления запросов на исполнения задачи.
 - 2.2. Разработка подсистемы сбора данных о состояниях виртуальных машин.
 - 2.3. Разработка подсистемы принятия решений о горизонтальном масштабировании.
 - 2.4. Разработка подсистемы отображения информации о текущей загрузки системы.
3. Разработка компонента виртуальных машин.

- 3.1. Разработка веб-сервиса, исполняемого в виртуальной машине.
- 3.2. Разработка дополнения веб-сервиса, исполняемого в виртуальной машине, обеспечивающее предоставление информации о загрузке системы.
- 4. Разработка контроллера виртуальной машины.
 - 4.1. Разработка внутреннего обработчика контроллера, для обработки вывода виртуальной машины.
 - 4.2. Разработка механизма перенаправления вывода консоли виртуальной машины во внутренний обработчик контроллера.
 - 4.3. Разработка механизма запуска, остановки и получения состояния о загрузке виртуальных машин.
- 5. Разработка клиентского компонента.
- 6. Разработка консоли администратора.

5 Подход к решению задачи

5.1 Клиент

Для взаимодействия клиента с системой реализован «клиентский компонент» системы. С помощью данного компонента пользователь сможет предоставлять исходные данные для обработки сервисами, а также получать и просматривать обработанные данные. Данный компонент будет написан на языке программирования Java.

В связи с вышесказанным возникает ряд вопросов:

1. Как передавать данные для обработки и получать ответ?
2. Куда передавать данные для обработки?

Решением первого вопроса обмена данными между клиентским приложением и сервером был принят формат обмена данными json, а именно, json запросы на сервер и json ответы сервера.

Json [25] — простой текстовый формат обмена данными, основанный на JavaScript. Json базируется на двух основных концепциях:

1. Коллекция пар ключ/значение, практически во всех языках программирования реализованы данные типы данных
2. Упорядоченный список значений.

Основываясь на данных концепциях, такой формат данных независим от языка программирования.

Решением вопроса о передачи данных для обработки стало разработка серверного компонента системы, о котором будет написано ниже.

5.2 Сервер

Сервер будет использоваться для следующих возможностей:

1. Для принятия решений о запуске и остановки виртуальных машин на хост — компьютере
2. Для передачи исходных данных от клиентского компонента системы для обработки сервисами

3. Для получения обработанных данных от витальных машин и передачи этих данных клиентскому компоненту
4. Для сбора информации о состоянии загруженности виртуальных машин.

Сервер будет представлять из себя Java сервлет, принимающий запросы и возвращающий ответа для клиентского компонента, консоли администратора, контроллера и виртуальных машин.

В связи с вышесказанным возникает ряд вопросов:

1. Где будет запускаться сервер?
2. В каком формате получать и передавать данные как от клиента, консоли администратора так и от витальных машин?
3. Как следить за состоянием виртуальных машин?

Решением первого вопроса о запуске сервера было принять решение, что сервлет будет разворачиваться на GlassFish, так как GlassFish легко устанавливается и вместе с ним идёт web консоль администратора для настройки.

GlassFish [26] — сервер приложений, распространяемый под лицензией CDDL, с открытым исходным кодом, разработанный Sun Microsystems. В настоящее время спонсируется корпорацией Oracle. Актуальная версия платформы называется Oracle GlassFish Server.

В основу GlassFish легли части кода Java System Application Server [27] компании Sun и ORM TopLink. В качестве сервлет-контейнера в нём используется модифицированный Apache Tomcat, дополненный компонентом Grizzly, использующим технологию Java NIO.

Решением второго вопроса обмена данными между клиентским приложением и витальными машинами, как уже было описано в клиентском приложении, был принят формат обмена данными json.

Решением третьего вопроса о мониторинге состояний виртуальных машин, было принято решение о хранении всей информации, о состояний виртуальных машин в базе данных. В качестве базы данных была выбрана

база данных PostgreSQL [28].

PostgreSQL — свободно распространяемая объектно — реляционная система управления базами данных (СУБД).

Существует в реализациях для множества UNIX-подобных платформ, включая AIX, различные BSD-системы, PostgreSQL базируется на языке SQL и поддерживает многие из возможностей стандарта SQL:2011

Сильными сторонами PostgreSQL считаются:

1. Высокопроизводительные и надёжные механизмы транзакций и репликации
2. Расширяемая система встроенных языков программирования
3. Наследование
4. Легкая расширяемость.

Основные возможности:

1. Функции являются блоками кода, исполняемыми на сервере, а не на клиенте БД
2. Триггеры определяются как функции, инициируемые операциями вставки, удаления или изменения данных
3. Механизм правил представляет собой механизм создания пользовательских обработчиков не только операций вставки, удаления или изменения данных, но и операции выборки
4. Имеется поддержка индексов следующих типов: В-дерево, хэш, R-дерево, GiST, GIN
5. Поддерживается одновременная модификация БД несколькими пользователями с помощью механизма Multiversion Concurrency Control
6. Поддерживается большой набор встроенных типов данных

5.3 Контроллер

Запуск контроллера виртуальных машин планируется на Unix подобной операционной системе, так как сборку образа IncludeOS с необходимым сервисом можно проводить через командную оболочку, а именно запуск всех

скриптов по сборке и запуску виртуальных машин. Контроллер будет представлять из себя программное обеспечение написанное на языке программирования java, установленное на хост – компьютере. Контроллер будет взаимодействовать с виртуальными машинами следующим образом:

1. При инициализации контроллера на хост машине автоматически запускается первая виртуальная машина
2. Через определённые промежутки времени виртуальная машина будет передавать контроллеру информацию о загрузке процессора и используемой оперативной памяти
3. Контроллер через перенаправление вывода консоли виртуальной машины считывает данные
4. При получении новых данных, контроллер обрабатывает их и совершает одно из трёх действий:
 - 4.1. Запустить ещё n образов виртуальных машин с новыми данными
 - 4.2. Остановить n образов виртуальных машин
 - 4.3. Отправить информацию об используемых ресурсах виртуальной машины
 - 4.4. Ничего не делать.

Для выполнения поставленной задачи нужно решить вопрос:

1. Каким образом взаимодействовать с виртуальными машинами?

Решением данного вопроса, взаимодействия с виртуальными машинами, было принято использование скриптов, написанных для выполнения в BASH [29], для компилирования образа виртуальной машины с необходимыми настройками dhcp (ip адреса, маски подсети и т. д.), запуска созданной виртуальной машины QEMU и перенаправления вывода запущенной виртуальной машины. Например, ниже представлен скрипт компилирования и запуска виртуальной машины под управлением IncludeOS с тестовым сервисом.

```
pushd examples/demo_service
mkdir -p build
pushd build
cmake ..
make
echo -e "Build complete \n"
echo -e "Starting VM with Qemu. "
echo -e "You should now get a boot message from the virtual
machine:"
../run.sh build/IncludeOS_example.img
echo -e "\nTest complete.\n"
popd
trap - EXIT
```

Текст скрипта компилирования и запуска тестового сервиса test.sh

Bash — усовершенствованная и модернизированная вариация командной оболочки Bourne shell. Одна из наиболее популярных современных разновидностей командной оболочки UNIX.

Bash — это командный процессор, работающий, как правило, в интерактивном режиме в текстовом окне. Bash также может читать команды из файла, который называется скриптом (или сценарием). Как и все Unix-оболочки, он поддерживает автодополнение имён файлов и директорий, подстановку вывода результата команд, переменные, контроль за порядком выполнения, операторы ветвления и цикла. Ключевые слова, синтаксис и другие основные особенности языка были заимствованы из sh.

5.4 Консоль администратора

Консоль администратора разрабатывается для мониторинга состояния виртуальных машин и отображение графиков их состояний. Данный компонент будет написан на языке программирования Java.

В связи с вышесказанным возникает ряд вопросов:

1. Какой формат запросов использовать для запроса состояний виртуальных машин?

2. Как отображать графики состояний виртуальных машин?

Решением первого вопроса обмена данными между консолью администратора и сервером, как уже было описано в клиентском приложении, был принят формат обмена данными json.

Решением второго вопроса о построении графиков состояний виртуальных машин стал JfreeChart [30] – библиотека с открытым исходным кодом на Java, распространяемое по лицензии LGPL и используемое для создания широкого спектра графиков. Используя JFreeChart, мы можем создать все основные типы 2D и 3D графики, таких как круговой диаграммы, гистограммы, линейный график, XY графика и 3D графики.

Причины выбора JfreeChart:

1. Бесплатный проект с открытым исходным кодом
2. Поставляется с хорошо документированных API
3. Поддерживает широкий спектр типов диаграмм, таких как круговая диаграмма, линия диаграммы, гистограммы, диаграммы Area и 3D графики
4. JFreeChart легко расширяемый.

5.5 Виртуальная машина

Сервис каждой виртуальной машины будет дополнен модулем, который выполняет следующие действия:

1. При старте считывает из параметров запуска ip адрес, на котором будет расположен сервис
2. В определённый промежуток времени выводит в консоль контроллера информацию о себе, а именно:
 - 2.1. Использование вычислительной мощности виртуального процессора, занимаемое виртуальной машиной
 - 2.2. Занимаемый объём оперативной памяти.

Для выполнения поставленных задач нужно решить следующие вопросы:

1. Как конфигурировать сетевую подсистему виртуальных машин для доступности сервиса?
2. Какое использовать программное обеспечение для запуска виртуальных машин?

Решение первого вопроса о конфигурировании сетевой подсистемы стало использованием, поставляемых вместе с IncludeOS библиотек, а именно `net4` [31]. Ниже приведён пример настройки сетевой подсистемы виртуальной машины.

```
#include <net/inet4>
void Service::start(const std::string&)
{
    auto& inet = net::Inet4::ifconfig(10.0);
    net::Inet4::ifconfig(
        { 10,0,0,42 },      // IP
        { 255,255,255,0 },  // Netmask
        { 10,0,0,1 },       // Gateway
        { 10,0,0,1 });      // DNS
}
```

Пример настройки сетевой подсистемы виртуальной машины.

Решением третьего вопроса о программном обеспечении для запуска виртуальных машин было принято использование QEMU.

QEMU [32] — свободно распространяемый программный продукт по лицензии GNU GPL 2 с открытым исходным кодом для эмуляции аппаратного обеспечения различных платформ.

Включает в себя эмуляцию процессоров Intel x86 и устройств ввода-вывода. Может эмулировать 80386, 80486, Pentium, Pentium Pro, AMD64 и другие x86-совместимые процессоры; PowerPC, ARM, MIPS, SPARC, SPARC64, m68k — лишь частично.

По умолчанию входит практически во все дистрибутивы Unix подобных операционных систем.

В настоящее время идёт разработка поддержки технологий аппаратной виртуализации (Intel VT и AMD SVM) на x86-совместимых процессорах Intel и AMD в QEMU. Первоначально разработка велась в рамках проекта Linux

KVM (Kernel-based Virtual Machine), в котором, помимо собственно KVM (поддержки технологий аппаратной виртуализации x86-совместимых процессоров на уровне ядра Linux), разрабатывались патчи для QEMU, позволяющие QEMU использовать функциональность KVM. Однако недавно разработчики QEMU в содружестве с разработчиками KVM приняли решение в ближайшем будущем интегрировать поддержку KVM в основную ветку QEMU (mainline).

На рисунке 3 изображён тестовый микросервис запущенный в операционной системе IncludeOS на QEMU.

```

Terminal
http://10.0.0.42/
10.0.0.42
IncludeOS
Hello all!!!!!!
This is improvised http
© 2016, IncludeOS AS

user@user-VirtualBox: ~/Documents/IncludeOS
IP: 10.0.0.42
Netmask: 255.255.255.0
Gateway: 10.0.0.1
DNS Server: 10.0.0.1
*** TEST SERVICE STARTED ***
=====
IncludeOS v0.10.0-rc.2-2-gc1e9172
+--> Running [ IncludeOS minimal example ]
<Service> TCP STATUS:
LISTENERS:
Port Queued
80 0
CONNECTIONS:
Proto Recv Send In Out Local Remote State
[ DHCPv4 ] Negotiation timed out
<Service> @on_connect: Connection 10.0.0.1:57898 successfully established.
<Service> @on_read: 317 bytes received.
<Service> Request:
GET / HTTP/1.1
Host: 10.0.0.42
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:51.0) Gecko/20100101 Firefox/51.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
<Service> Responding with 200 OK.
<Service> @on_write: 632 bytes written.
<Service> @on_connect: Connection 10.0.0.1:57906 successfully established.
<Service> @on_read: 298 bytes received.
<Service> Unable to parse request.
<Service> TCP STATUS:
LISTENERS:
Port Queued
80 0
CONNECTIONS:
Proto Recv Send In Out Local Remote State
tcp4 10.0.0.42:80 10.0.0.1:57906 ESTABLISHED

```

Рисунок 4.1 — Запущенный тестовый микросервис

Заключение

После анализа литературы по заданной теме и изучения документации по проекту IncludeOS был написан тестовый web сервис на языке программирования C++, под управлением одномодульной операционной системы IncludeOS. Данный web сервис принимает get, set и put запросы от клиентов мобильного платёжного приложения под управлением Android и Ios, производит обработку полученных данных и возвращает результат на клиенты.

Разработка программного обеспечения, в который будет входить: клиентский компонент системы, сервер для мониторинга виртуальных машин, контроллер и консоль администратора началась, но в поставленный сжатый срок разработка комплекса таких масштабов невозможна, в связи с этим было принято решение о выделении данной задачи как тему магистерской диссертации.

Список использованных источников

1. Главная страница сайта ООО «Soft-logic» [Электронный ресурс]. - Режим доступа: <https://soft-logic.ru/>
 2. Главная страница сайта IncludeOS [Электронный ресурс]. - Режим доступа: <http://www.includeos.org/>
 3. Микросервисы (Microservices) [Электронный ресурс]. - Режим доступа: <https://habrahabr.ru/post/249183/>
 4. Преимущества и недостатки микросервисной архитектуры [Электронный ресурс]. - Режим доступа: <http://eax.me/micro-service-architecture/>
 5. Микросервисы на практике - SmartMe University [Электронный ресурс]. - Режим доступа: <http://smartme.university/course/microservices-in-practice/>
 6. Национальная библиотека им. Н.Э.Баумана Bauman National Library [Электронный ресурс]. - Режим доступа: <http://ru.bmstu.wiki/Гипервизор>
 7. Графический интерфейс пользователя [Электронный ресурс]. - Режим доступа: <http://www.microchip.com.ru/Support/GUI.html>
 8. CppCon 2016 [Электронный ресурс]. - Режим доступа: <https://cppcon2016.sched.com/>
 9. CppCon 2016: #Include <os>: from bootloader to REST API with the new C++ [Электронный ресурс]. - Режим доступа: <https://cppcon2016.sched.com/event/7nLe/include-ltosgt-from-bootloader-to-rest-api-with-the-new-c>
 10. В рамках проекта IncludeOS развивается ядро для обособленного запуска C++ приложений [Электронный ресурс]. - Режим доступа: <https://www.opennet.ru/opennews/art.shtml?num=43444>
- [XXX1] Встраиваемые операционные системы [Электронный ресурс]. - Режим доступа: <http://www.itechno.ru/index.php&>**

**option=com_k2&view=itemlist&task=category&id=153:Встраиваемые
операционные системы**

11. IncludeOS [Hatred's Log Place] — Programming [Электронный ресурс]. - Режим доступа: <https://htrd.su/wiki/zhurnal/2016/09/22/includeos>
12. #Include os [Электронный ресурс]. - Режим доступа: <https://www.slideshare.net/IncludeOS/include-ltos-from-bootloader-to-rest-api-with-the-new-c>
13. VMware ESX [Электронный ресурс]. - Режим доступа: <http://www.vmware.com/ru/products/esxi-and-esx.html>
14. XenServer | Open Source Server Virtualization [Электронный ресурс]. - Режим доступа: <https://xenserver.org/>
15. Citrix [Электронный ресурс]. - Режим доступа: <https://www.citrix.ru/>
16. Виртуализация VMware [Электронный ресурс]. - Режим доступа: <http://www.vmware.com/ru.html>
17. Гипервизоры, виртуализация и облако: О гипервизорах, виртуализации систем и о том, как это работает в облачной среде [Электронный ресурс]. - Режим доступа: <http://www.ibm.com/developerworks/ru/library/cl-hypervisorcompare/>
18. Базовая информация о VMware vSphere [Электронный ресурс]. - Режим доступа: <https://habrahabr.ru/post/226231/>
19. Анализ гипервизора VMware, его преимущества и недостатки [Электронный ресурс]. - Режим доступа: <https://xakep.ru/2011/11/07/Vmware>
20. Xen [Электронный ресурс]. - Режим доступа: <http://xgu.ru/wiki/Xen>
21. Национальная библиотека им. Н. Э. Баумана Bauman National Library / Паравиртуализация [Электронный ресурс]. - Режим доступа: <http://ru.bmstu.wiki/Паравиртуализация>
22. XenServer - Server Virtualization and Consolidation – Citrix [Электронный ресурс]. - Режим доступа: <https://www.citrix.com/products/xenserver/>
23. Архитектура Hyper-V [Электронный ресурс]. - Режим доступа:

- <https://habrahabr.ru/post/98580/>
24. Review Hyper-V [Электронный ресурс]. - Режим доступа: [https://msdn.microsoft.com/library/hh831531\(v=ws.11\).aspx](https://msdn.microsoft.com/library/hh831531(v=ws.11).aspx)
- [XXX2] IncludeOS: A minimal, resource efficient unikernel for cloud services [Электронный ресурс]. - Режим доступа: <https://folk.uio.no/paalee/publications/2015-cloudcom.pdf>**
- [XXX3] B. Stroustrup, The C++ Programming Language (4th. edition). Addison-Wesley, 2013. – Pp 66-67.**
- [XXX4] Статические библиотеки [Электронный ресурс]. - Режим доступа: <https://www.opennet.ru/docs/RUS/zlp/003.html>**
- [XXX5] GCC toolchain [Электронный ресурс]. - Режим доступа: https://ru.wikipedia.org/wiki/GNU_toolchain**
- [XXX6] P. Pedriana, “EASTL - Electronic Arts Standard Template library,” Open Standards, Tech. Rep., Apr. 2007. [Электронный ресурс]. - Режим доступа: <http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2007/n2271.html>**
- [XXX7] KVM virtio drivers [Электронный ресурс]. - Режим доступа: <http://itman.in/kvm-virtio-drivers/>**
- [XXX8] R. Russell, M. S. Tsirkin, C. Huck, and P. Moll, “Virtual I/O Device (VIRTIO) Version 1.0,” OASIS Standard, OASIS Committee Specification 02, January 2015. [Электронный ресурс]. - Режим доступа: <http://docs.oasis-open.org/virtio/virtio/v1.0/virtio-v1.0.html>**
- [XXX9] Callback (программирование) [Электронный ресурс]. - Режим доступа: <https://ru.wikipedia.org/wiki/Callback>**
25. Json [Электронный ресурс]. - Режим доступа: <http://www.json.org/json-ru.html>
26. GlassFish Server Documentation [Электронный ресурс]. - Режим доступа: <https://glassfish.java.net/documentation.html>
27. GlassFish Server [Электронный ресурс]. - Режим доступа: <http://www.oracle.com/us/products/middleware/cloud-app->

- foundation/glassfish-server/overview/index.html
28. PostgreSQL: The world's most advanced open source database [Электронный ресурс]. - Режим доступа: <https://www.postgresql.org/>
 29. Bash (Unix shell) [Электронный ресурс]. - Режим доступа: [https://en.wikipedia.org/wiki/Bash_\(Unix_shell\)](https://en.wikipedia.org/wiki/Bash_(Unix_shell))
 30. JFreeChart – Jfree.org [Электронный ресурс]. - Режим доступа: <http://www.jfree.org/jfreechart/>
 31. inet4.cpp [Электронный ресурс]. - Режим доступа: <https://github.com/hioa-cs/IncludeOS/blob/master/src/net/inet4.cpp>
 32. QEMU [Электронный ресурс]. - Режим доступа: <http://www.qemu-project.org/>