Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования «Севастопольский государственный университет»

Институт информационных технологий и управления в технических системах
(полное название института)

Кафедра информационных технологий и компьютерных систем
(полное название кафедры)

Пояснительная записка

к выпускной работе бакалавра
(выпускной квалификационной работе, дипломному проекту (работе))

на тему: Разработка виртуальной инфраструктуры для реализации облачных услуг

Выполнил: студент 4 курса, группы ВТб-41д

направления подготовки (специальности) 09.03.01 – информатика и вычислительная техника
(шифр и название направления подготовки (специальности))

направленность/профиль/специализация 09.03.01.01 «ЭВМ, системы и сети»

Умеров Амет Ремзиевич
(фамилия, имя, отчество студента)

Руководитель Мащенко Е.Н., доцент
(фамилия, инициалы, ученая степень, звание, должность)

Дата допуска к защите «____»_____ 2015 г.

Зав. кафедрой _____ Брюховецкий А.А.
(подпись) (инициалы, фамилия)

2015 г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования «Севастопольский государственный университет»

Институт информационных технологий и управления в технических системах
Кафедра информационных технологий и компьютерных систем
Направление подготовки (специальность) 09.03.01 «Информатика и вычислительная техника»
(код и название)
Направленность/профиль/специализация 09.03.01.01 «ЭВМ, системы и сети»

УТВЕРЖДАЮ

Заведующий кафедрой ИтиКС
А.А. Брюховецкий

“26” марта 2015 года

З А Д А Н И Е

на выпускную квалификационную работу бакалавра
(указать форму в соответствии с ФГОС, при наличии)
студенту Умерову Амету Ремзиевичу

1. Тема работы (проекта) Разработка виртуальной инфраструктуры для реализации облачных услуг
руководитель работы (проекта) Мащенко Елена Николаевна, канд.техн.наук, доцент
(фамилия, имя, отчество, степень, звание, должность)
Утверждены приказом высшего учебного заведения от “01” апреля 2015 года № 129-П

2. Срок подачи студентом работы (проекта) 08.06.2015 г.

3. Входные данные к работе (проекту) параметры клиентских контейнеров, параметры администрирования виртуальной инфраструктуры, параметры системы мониторинга и резервного копирования, перечень оказываемых облачных услуг; критерии качества обслуживания клиентов. Технологии виртуализации: OpenVZ, KVM. Критерии эффективности виртуальной инфраструктуры по производительности и надежности определены в соглашении об уровне обслуживания облачного провайдера.

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые нужно разработать) Введение. 1. Постановка задачи. 2. Обзор современных методов и технологий серверной виртуализации. 3. Системный анализ виртуальной инфраструктуры 4. Описание виртуальной инфраструктуры. 5. Руководство администратора. 6. Руководство пользователя 7. Результаты тестирования. 8. Безопасность жизнедеятельности. Заключение. Библиографический список. Приложения.

5. Перечень графического материала (с точным указанием обязательных чертежей)
1.Постановка задачи (1 плакат)
2. Структурная схема виртуальной инфраструктуры (1 лист)
3. Схемы алгоритмов функционирования виртуальной инфраструктуры (2 листа)
4. Результаты тестирования виртуальной инфраструктуры (1 плакат)

6. Консультанты разделов работы (проекта)

Раздел	Фамилия, инициалы и должность консультанта	Подпись, дата	
		задание выдал	задание принял
Безопасность жизнедеятельности	ст.пр. Григорьев Е.Ф.		
Нормоконтроль	Доц. Корепанова Н.Л.		

7. Дата выдачи задания 26.03.2015 г.

КАЛЕНДАРНЫЙ ПЛАН

№ п/п	Название этапов работы (проекта)	Срок выполнения этапов работы (проекта)	Примечание
1	Анализ постановки задачи, системный анализ	30.03.2015 – 26.04.2015	
2	Разработка алгоритмов функционирования системы и основных подсистем	27.04.2015 – 10.05.2015	
3	Разработка виртуальной инфраструктуры	11.05.2015 – 21.05.2015	
4	Испытание виртуальной инфраструктуры	22.05.2015 – 31.05.2015	
5	Оформление пояснительной записки и чертежей	01.06.2015 – 07.06.2015	
6	Представление работы на кафедру	08.06.2015	
7	Защита работы в ГЭК	17.06.2015	

Студент _____ Умеров А.Р.
(подпись) (фамилия и инициалы)

Руководитель работы (проекта) _____ Мащенко Е.Н.
(подпись) (фамилия и инициалы)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Облачные услуги — это способ предоставления, потребления и управления технологией. Данный тип услуг выводит гибкость и эффективность на новый уровень, путем эволюции способов управления, таких как непрерывность, безопасность, резервирование и самообслуживание, которые соединяют физическую и виртуальную среду. В данной сфере возрастает потребность в качественно продуманной архитектуре, позволяющей надежно и правильно организовать облачную инфраструктуру.

Для эффективной работы облачной инфраструктуры требуется эффективная структура и организация. Небольшая команда из специалистов и бизнес-пользователей может создать обоснованный план и организовать свою работу в инфраструктуре. Данная выделенная группа может намного эффективнее построить и управлять нестандартной облачной инфраструктурой, чем если компании будут просто продолжать добавлять дополнительные сервера и сервисы для поддержки центра обработки данных (ЦОД).

IaaS (Infrastructure as a Service) — это предоставление пользователю компьютерной и сетевой инфраструктуры и их обслуживание как услуги в форме виртуализации, то есть виртуальной инфраструктуры. Другими словами, на базе физической инфраструктуры дата-центров (ДЦ) провайдер создает виртуальную инфраструктуру, которую предоставляет пользователям как сервис. Стоит отметить, что IaaS не предполагает передачи в аренду программного обеспечения, а всего лишь предоставляет доступ к вычислительным мощностям.

Технология виртуализации ресурсов позволяет физическое оборудование (сервера, хранилища данных, сети передачи данных) разделить между пользователями на несколько частей, которые используются ими для выполнения текущих задач. К примеру, на одном физическом сервере можно запустить сотни виртуальных серверов, а пользователю для решения задач выделить время доступа к ним.

1 ПОСТАНОВКА ЗАДАЧИ

Конечная цель проектирования — разработка виртуальной инфраструктуры для реализации облачных услуг.

Виртуальная инфраструктура должна обладать следующими характеристиками:

- использование, по возможности, продуктов, распространяющихся под свободной лицензией (GNU GPL) для организации инфраструктуры;
- устранение единой точки отказа при проектировании инфраструктуры;
- защита от распределенных атак на отказ (DDoS);
- использование инфраструктуры в бизнесе для предоставления облачных услуг клиентам.

Аппаратное обеспечение должно базироваться в дата-центре ориентированным на требования стандарта Tier III, состоящего из информационной, телекоммуникационной и инженерной инфраструктуры, с возможностью аренды выделенных серверов и аппаратной защиты от DDoS-атак, а также поддержкой аппаратного RAID. Рекомендуемые характеристики выделенных серверов виртуализации:

- 2xIntel® Xeon E5430 @ 2.66GHz;
- минимальный объем ОЗУ 32 Гб;
- минимум 960 Гб места на жестком диске (SSD);
- поддержка аппаратного RAID;
- операционная система не ниже Debian 7 GNU/Linux или CentOS 6.5;
- интернет-канал с пропускной способностью не менее 100 Мб/с;
- возможность добавления дополнительных IP-адресов к серверам;
- возможность удаленного доступа к серверам посредством IPMI (Intelligent Platform Management Interface).

Следует предусмотреть возможность развертывания дополнительных виртуальных серверов, для организации DNS-серверов, системы мониторинга и платежной системы. Также необходимо предусмотреть наличие

системы хранения данных (СХД) для резервных копий, а также расширение дискового пространства на выделенных серверах.

Для разработки виртуальной инфраструктуры необходимо реализовать следующие этапы:

- выбор технологий виртуализации;
- выбор физических серверов на основе услуги IaaS в ДЦ;
- приобретение лицензий на программное обеспечение;
- приобретение подсетей IP-адресов;
- создание и подключение инфраструктуры мониторинга, платежной системы (биллинга), резервного копирования;
- реализация аппаратной защиты от DDoS-атак;
- выбор перечня PaaS-услуг;
- внедрение перечня предоставляемых PaaS-услуг;
- создание руководства администратора по виртуализации;
- создание руководства для клиентов по часто задаваемым вопросам;
- внедрение инфраструктуры на рынке PaaS-услуг.

2 ОБЗОР СОВРЕМЕННЫХ МЕТОДОВ И ТЕХНОЛОГИЙ СЕРВЕРНОЙ ВИРТУАЛИЗАЦИИ

Понятие облачных вычислений на сегодняшний день уже достаточно хорошо известно и в информационных технологиях (ИТ) и бизнесе. Облачные вычисления являются новой сервисной моделью предоставления вычислительных услуг. Еженедельно появляются новые статьи, презентации, книги об облачных вычислениях, что говорит о заинтересованности сообщества в данных технологиях.

За время существования информационных технологий создавались и изменялись подходы к построению информационных систем. Первой моделью информационной системы была монолитная архитектура. В данной модели на одном компьютере работали и приложения и база данных (БД), а пользователи сидели у «тонких» терминалов которые отображали информацию с компьютера. У архитектуры было большое количество недостатков, поэтому впоследствии ее сменила более перспективная клиент-серверная архитектура. В этом случае на компьютере располагался выделенный сервер баз данных, а пользователи с «толстых клиентов» в разгружали сервер БД. Затем появилась еще более современная многоуровневая архитектура, у которой логика приложений вынесена на отдельный компьютер, который называется сервер приложений, а пользователи работали на «тонких» клиентах через веб-браузеры. В современном информационном мире большинство приложений выполнено именно в многоуровневой архитектуре. Она подразумевает развертывание всей ИТ-инфраструктуры на территории заказчика [1].

Облачные вычисления являются следующим шагом в эволюции архитектуры построения информационных систем. Благодаря преимуществам данного подхода вполне очевидно, что многие информационные системы в ближайшее время переносятся или будут перенесены в облако. Процесс уже идет полным ходом и его игнорирование или недооценка может привести

к поражению в конкурентной борьбе на рынке. В данном случае имеется ввиду не только отставание ИТ, или неоправданные затраты на него, но и отставание в развитии бизнеса компании, которая зависит от гибкости информационной инфраструктуры и скорости вывода новых сервисов и продуктов на рынок.

В феврале 2011 года, ИТ-директор американского правительства Вивек Кундра опубликовал стратегию стратегию переноса части информационных систем в облако. Стратегия была опубликована в документе под названием «Federal Cloud Computing Strategy», который четко описывает порядок и сроки переноса. Целью работ является уменьшение сложности и повышение управляемости ИТ, увеличение нагрузки оборудования до 70-80%, а также уменьшение количества центров обработки данных.

Дата-центр, иначе именуемый центром обработки данных является специализированное здание, в котором размещается серверное и сетевое оборудование, которое подключает абонентов к сети Интернет. Основное требование предъявляемое к центрам обработки данных — отказоустойчивость. При этом подразумевается отключение дата-центра как на время планово-предупредительных работ и профилактики оборудования, так и внеплановых аварийных ситуаций.

Классификация Tier описывает надежность функционирования ЦОД и является необходимой для компаний, как желающих построить свой ЦОД, так и для арендующих чужие вычислительные мощности. В зависимости от критичности бизнеса, в зависимости от потерь, которые понесет компания в случае остановки бизнес-процессов выбирается тот или иной уровень надежности. В свою очередь, высокий уровень надежности требует высоких материальных и эксплуатационных затрат, поэтому и стоимость вычислительных мощностей зависит от уровня надежности ЦОД [2].

На сегодняшний день существует четыре уровня надежности ЦОД названные Tier I, Tier II, Tier III и Tier IV, которые были введены организацией Uptime Institute (Институт бесперебойных процессов, США):

– Tier I: время простоя 28.8 часов в год, коэффициент отказоустойчивости 99.671%;
– Tier II: 22.0 часа в год, 99.749%;
– Tier III: 1.6 часа в год, 99.982%;
– Tier IV: 0.4 часа в год, 99.995%.

ЦОД уровня Tier I (базовый уровень) подвержен нарушениями работы как от плановых, так и от внеплановых действий. Применение фальшпола, источников бесперебойного питания (ИБП), дизель-генераторных установок (ДГУ) не обязательно. Если ИБП и ДГУ используются, то выбираются более простые модели, без резерва, с множеством точек отказа. Возможны самопроизвольные отказы оборудования. К простою ЦОД также приведут ошибки в действиях обслуживающего персонала. В таких ЦОД отсутствует защита от случайных и намеренных событий, обусловленных действиями человека.

В ЦОД уровня Tier II (с резервированными компонентами) время простоя возможно в связи с плановыми и внеплановыми работами, а также аварийными ситуациями, однако оно сокращено благодаря внедрению одной резервной единицы оборудования в каждой системе. Таким образом, системы кондиционирования, ИБП и ДГУ имеют одну резервную единицу, тем не менее, профилактические работы требуют отключения ЦОД. В центрах обработки данных с резервированными компонентами требуется наличие минимальных защитных мер от влияния человека.

Третий уровень надежности (уровень с возможность параллельного проведения ремонтных работ) требует осуществления любой плановой деятельности без остановки ЦОД. Под плановыми работами подразумевается профилактическое и программируемое техническое обслуживание, ремонт и замена компонентов, добавления или удаление компонентов, а также их тестирование. В таком случае необходимо иметь резерв, благодаря которому можно пустить всю нагрузку по другому пути, во время работ на первом. Для реализации Tier III необходима схема резервирования блоков схем кондиционирования, ИБП, ДГУ N+1, также требуется наличие двух комплектов

трубопроводов для системы кондиционирования, построенной на основе чиллера (холодильной машины). Строительные требования обязывают сохранять работоспособность ЦОД при большинстве случаев намеренных и случайных вмешательств человека. Также следует предусмотреть резервные входы, дублирующие подъездные пути, контроль доступа, отсутствие окон, защиту от электромагнитного изучения.

Четвертый уровень надежности ЦОД (отказоустойчивый) характеризуется безостановочной работой при проведении плановых мероприятий и способен выдержать один серьезный отказ без последствий для критически важной нагрузки. Необходим дублированный подвод питания, резервирование системы кондиционирования и ИБП по схеме 2(N+1). Для дизель-генераторных установок необходима отдельная площадка с зоной хранения топлива. Tier IV требует защиту от всех потенциальных проблем в связи с человеческим фактором. Регламентированы избыточные средства защиты от намеренных или случайных действий человека. Учтено влияние сейсмоявлений, потопов, пожаров, ураганов, штормов, терроризма.

Дата-центры по виду использования подразделяют на корпоративные и коммерческие (аутсорсинговые). Корпоративные ДЦ предназначены для обслуживания конкретной компании, коммерческие, в свою очередь, предоставляют услуги всем желающим.

Некоторые ДЦ предлагают клиентам дополнительные услуги по использованию оборудования, по автоматическому уходу от различных видов атак. Команда специалистов круглосуточно производит мониторинг серверов. Для обеспечения сохранности данных используются системы резервного копирования. Для предотвращения кражи данных, в дата-центрах используются различные системы ограничения физического доступа, системы видеонаблюдения.

Дата-центры предоставляют несколько основных типов услуг, среди которых:

- виртуальный хостинг (shared hosting);
- виртуальный сервер (virtual private/dedicated server);

– выделенный сервер (dedicated server);
– размещение сервера (colocation);
– выделенная зона (dedicated area).

Виртуальный хостинг используется для размещения большого количества сайтов на одном веб-сервере. В основном для построения веб-сервра используется типичный стек технологий LAMP, где в качестве операционной системы выступает GNU/Linux, http-сервер Apache (зачастую в связке с nginx), сервер баз данных MySQL, интерпретируемые скриптовые языки PHP, Perl, Python. Существует решение на базе ОС Windows Server, где в качестве http-сервера используется IIS, в качестве СУБД выступает MS SQL, а также существует поддержка платформы ASP.NET. Разделение ресурсов на виртуальном хостинге основывается на ограничении дискового пространства, сетевого трафика, количества используемых доменов, почтовых ящиков, баз данных, FTP-аккаунтов, ограничение на использование процессорного времени, памяти для PHP-скриптов и так далее.

Виртуальный выделенный сервер эмулирует работу отдельного физического сервера. На одной физической машине может быть запущено несколько виртуальных серверов, при этом каждый виртуальный сервер имеет свои процессы, ресурсы и отдельное администрирование. Для реализации виртуальных машин используются технологии виртуализации, как системы с открытым исходным кодом, так и коммерческие.

В случае выделенного сервера, клиенту целиком предоставляется отдельная физическая машина. Владелец сервера имеет возможность смены конфигурации оборудования, установки любой операционной системы. Такой тип хостинга подходит для высоконагруженных проектов.

Размещение сервера отличается от услуги предоставления выделенного сервера тем, что ДЦ размещает у себя сервер, который заранее подготовил клиент. Дата-центр подключает его в общую инфраструктуру ЦОДа, обеспечивает бесперебойное электропитание, охлаждение, доступ к сетевому каналу, удаленный доступ к серверу, охрану, мониторинг и другие услуги.

Выделенная зона предоставляется в основном для специальных клиентов, имеющих строгие нормы безопасности. В этом случае дата-центр предоставляет выделенную зону, обеспеченную электроснабжением, холодоснабжением и системами безопасности, а клиент сам создает свой дата-центр внутри этого пространства.

Также можно выделить такую услугу, как аренда телекоммуникационных стоек, которая является частным случаем размещения сервера, с отличием в том, что арендаторами в основном являются юридические лица.

При построении облачной инфраструктуры важную роль играет виртуализация.

Виртуализация — абстракция вычислительных ресурсов и предоставление пользователю системы, которая инкапсулирует (скрывает в себе) собственную реализацию. Проще говоря, пользователь работает с удобным для себя представлением объекта, и для него не имеет значения, как объект устроен в действительности. Термин «виртуализация» появился в шестидесятых годах XX века, а в девяностых — стали очевидны перспективы подхода: с ростом аппаратных мощностей, как персональных компьютеров, так и серверных решений, вскоре представится возможность использовать несколько виртуальных машин на одной физической платформе.

Понятие виртуализации можно условно разделить на две категории [3]:

- виртуализация платформ, продуктом этого вида виртуализации являются виртуальные машины — некие программные абстракции, запускаемые на платформе реально аппаратно-программных систем;
- виртуализация ресурсов преследует целью комбинирование или упрощение представления аппаратных ресурсов для пользователя и получение неких пользовательских абстракций оборудования, пространств имен, сетей.

Когда производится виртуализация, существует множество способов ее осуществления. Фактически есть несколько путей, с помощью которых достигаются одинаковые результаты через разные уровни абстракции [4]:

- эмуляция аппаратных средств;

– полная виртуализация;
– паравиртуализация;
– виртуализация уровня ОС.

Эмуляция аппаратных средств является одним из самых сложных методов виртуализации. В то же время, главной проблемой при эмуляции аппаратных средств является низкая скорость работы, в связи с тем, что каждая команда моделируется на основных аппаратных средствах. В эмуляции оборудования используется механизм динамической трансляции, то есть каждая из инструкций эмулируемой платформы заменяется на заранее подготовленный фрагмент инструкций физического процессора [5]. Архитектура процесса эмуляции представлена на рисунке 2.1.

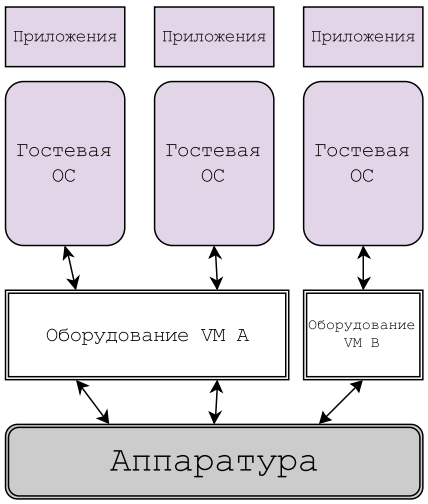

Рисунок 2.1 – Эмуляция аппаратных средств

Примерами виртуализации посредством эмуляции являются программные платформы QEMU и Bochs.

Система QEMU поддерживает два режима эмуляции: пользовательский и системный. Пользовательский режим эмуляции позволяет процессу, созданному на одном процессоре, работать на другом (выполняется динамический перевод инструкций для принимающего процессора и конвертация системных вызовов Linux). Системный режим позволяет эмулировать систему целиком, включая процессор и разнообразную периферию. Достоинством QEMU является его быстрый и компактный динамический транслятор. Ди-

намический транслятор позволяет во время исполнения переводить инструкции целевого (гостевого) процессора в инструкции центрального процессора хоста для обеспечения эмуляции. QEMU обеспечивает динамическую трансляцию преобразованием целевой инструкции в микрооперации. Эти микрооперации представляют собой элементы C-кода, которые компилируются в объекты. Затем запускается основной транслятор, который отображает целевые инструкции на микрооперации для динамической трансляции. Такой подход не только эффективен, но и обеспечивает переносимость.

Использование QEMU в качестве эмулятора персонального компьютера обеспечивает поддержку разнообразных периферийных устройств. Сюда входят стандартные периферийные устройства — эмулятор аппаратного видеоадаптера (VGA), мыши и клавиатуры PS/2, интерфейс IDE для жестких дисков, интерфейс CD-ROM и эмуляция дисковода. Кроме того, QEMU имеет возможность эмуляции сетевых адаптеров NE2000 (PCI), последовательных портов, многочисленных звуковых плат и контроллера PCI Universal Host Controller Interface (UHCI), Universal Serial Bus (USB) (с виртуальным USB концентратором). Также поддерживается до 255 процессоров с поддержкой симметричной многопроцессорности (SMP).

Полная виртуализация использует гипервизор, который осуществляет связь между гостевой ОС и аппаратными средствами физического сервера. В связи с тем, что вся работа с гостевой операционной системой проходит через гипервизор, скорость работы данного типа виртуализации ниже чем в случае прямого взаимодействия с аппаратурой. Основным преимуществом является то, что в ОС не вносятся никакие изменения, единственное ограничение — операционная система должна поддерживать основные аппаратные средства. Архитектура полной виртуализации представлена на рисунке 2.2.

Полная виртуализация возможна исключительно при условии правильной комбинации оборудования и программного обеспечения. Например, она была невозможной ни в серии IBM System/360, за исключением IBM System/360-67, ни в ранних IBM System/370, пока IBM не добавила оборудование виртуальной памяти в своих System/370 в 1972 г. Аналогичная

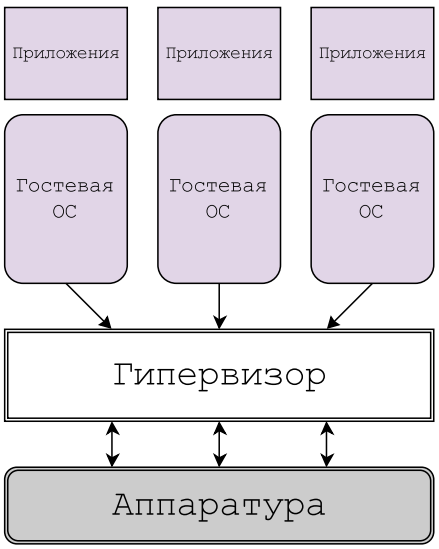

Рисунок 2.2 – Архитектура полной виртуализации

ситуация и с платформой x86: полная виртуализация была возможна не в полной мере, до добавления технологий AMD-V и Intel VT.

KVM (Kernel-based Virtual Machine) — программное решение, обеспечивающее виртуализацию в среде Linux, которая поддерживает аппаратную виртуализацию на базе Intel VT (Virtualization Technology) либо AMD SVM (Secure Virtual Machine) [6]. KVM не выполняет никакой самоэмуляции, вместо этого, программа, работающая в пользовательском пространстве, применяет интерфейс /dev/kvm для настройки адресного пространства гостевого виртуального сервера, берет его смоделированные ресурсы ввода/вывода и отображает его образ на образ хоста.

В архитектуре KVM, виртуальная машина выполняется как обычный Linux-процесс, запланированный стандартным планировщиком Linux. На самом деле виртуальный процессор представляется как обычный Linux-процесс, это позволяет KVM пользоваться всеми возможностями ядра Linux. Эмуляцией устройств управляет модифицированная версия QEMU, которая обеспечивает эмуляцию BIOS, шины PCI, шины USB, а также стандартный набор устройств, таких как дисковые контроллеры IDE и SCSI, сетевые карты и другие.

Hyper-V является технологией виртуализации на основе гипервизора для 64х-разрядной версии операционной системы Windows Server 2008.

Платформа позволяет несколько изолированным операционным системам разделять одну аппаратную платформу. Hyper-V оперирует понятием «раздел». Стек виртуализации работает в родительском разделе и имеет прямой доступ к аппаратным устройствам. Корневой раздел создает дочерние разделы, в которых располагаются гостевые ОС. Корневой раздел создает дочерние разделы с помощью интерфейса программирования приложений гипервызовов (API).

Разделы не имеют прямого доступа к физическому процессору и они не могут обрабатывать прерывания. Вместо этого, они имеют доступ к виртуальному процессору и работают в виртуальной памяти. Гипервизор обрабатывает прерывания к процессора и перенаправляет их к соответствующему разделу. Hyper-V позволяет ускорить трансляцию адресов между различными гостевыми виртуальными адресными пространствами с помощью IOMMU (Input Output Memory Management Unit), который работает независимо от аппаратного управления памятью используемой процессором.

Дочерние разделы также не имеют прямого доступа к аппаратным ресурсам, ресурсы в них представлены в качестве виртуальных устройств (VDevs). Запросы к виртуальным устройствам перенаправляются либо через VMBus, либо через гипервизор. Hyper-V требует процессор, которые поддерживает аппаратную виртуализацию (Intel VT или AMD-V).

Паравиртуализация имеет некоторые сходства с полной виртуализацией. Этот метод использует гипервизор для разделения доступа к основным аппаратным средствам, но объединяет код, касающийся виртуализации, в непосредственно операционную систему, поэтому недостатком метода является то, что гостевая ОС должна быть изменена для гипервизора. Но паравиртуализация существенно быстрее полной виртуализации, скорость работы виртуальной машины приближена к скорости реальной, это осуществляется за счет отсутствия эмуляции аппаратуры и учета существования гипервизора при выполнении системных вызовов в коде ядра. Вместо привилегированных операций совершаются гипервызовы обращения ядра

гостевой ОС к гипервизору с просьбой о выполнении операции. Архитектура паравиртуализации представлена на рисунке 2.3.

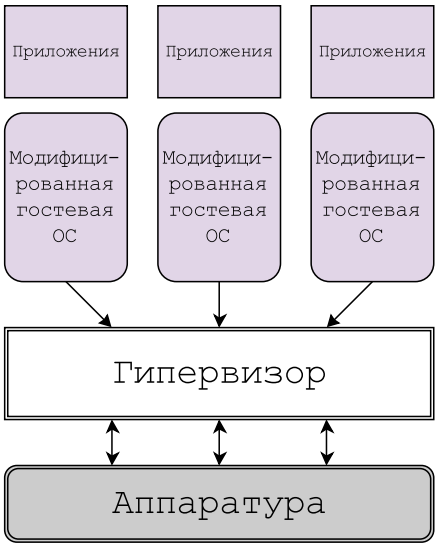

Рисунок 2.3 – Архитектура паравиртуализации

Для организации паравиртуализации используется программный продукт Xen.

Xen — это монитор виртуальных машин (VMM) или гипервизор с поддержкой паравиртуализации для процессоров архитектуры x86, распространяющийся с открытым исходным кодом. Xen может организовать совместное безопасное исполнение нескольких виртуальных машин на одной физической системе, с производительностью близкой к той, которая была бы непосредственно на физическом оборудовании. Он перекладывает большинство задач по поддержке аппаратуры на гостевую ОС, работающую в управляющей виртуальной машине, также известной как домен 0 (dom0) [8]. Сам Xen содержит только код, который необходим для обнаружения и запуска остальных процессоров системы, настройки обработки прерываний и нумерации шины PCI. Драйверы устройств работают не в Xen, а внутри привилегированной гостевой операционной системы. Такой подход обеспечивает совместимость с большинством устройств, поддерживаемых Linux. Сборка XenLinux по умолчанию содержит поддержку большого количества серверного сетевого и дискового оборудования, однако при необходимости

можно добавить поддержку других устройств, скомпилировав ядро Linux обычным способом.

В паравиртуальном режиме (PV) оборудование не эмулируется, и гостевая операционная система должна быть специальным образом модифицирована, для работы в таком окружении. Начиная с версии 3.0, ядро Linux поддерживает запуск в паравиртуальном режиме без перекомпиляции со сторонними патчами. Преимущество режима паравиртуализации состоит в том, что он не требует поддержки аппаратной виртуализации со стороны процессора, а также не тратит вычислительные ресурсы для эмуляции оборудования на шине PCI. Режим аппаратной виртуализации (HVM), который появился в Xen, начиная с версии 3.0 гипервизора требует поддержки со стороны оборудования. В этом режиме для эмуляции виртуальных устройств используется QEMU, который весьма медлителен несмотря на паравиртуальные драйверами. Однако со временем поддержка аппаратной виртуализации в оборудовании получила настолько широкое распространение, что используется даже в современных процессорах лэптопов.

Виртуализация уровня операционной системы отличается от других. Она использует технику, при которой сервера виртуализируются непосредственно над ОС. Недостатком метода является то, что поддерживается одна единственная операционная система на физическом сервере, которая изолирует контейнеры друг от друга. Преимуществом виртуализации уровня ОС является «родная» производительность. Виртуализация уровня ОС — метод виртуализации, при котором ядро операционной системы поддерживает несколько изолированных экземпляров пространства пользователя, вместо одного. Эти экземпляры с точки зрения пользователя полностью идентичны реальному серверу. Для систем на базе UNIX, эта технология может рассматриваться как улучшенная реализация механизма chroot. Ядро обеспечивает полную изолированность контейнеров, поэтому программы из разных контейнеров не могут воздействовать друг на друга. Архитектура виртуализации уровня ОС представлена на рисунке 2.4.

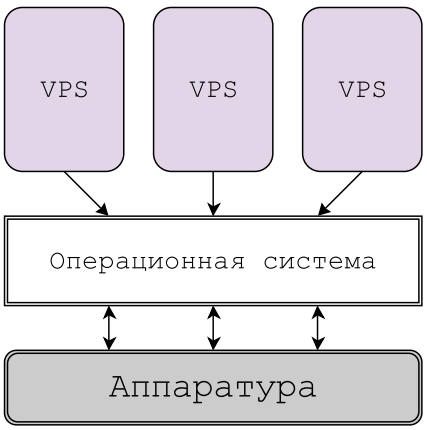

Рисунок 2.4 – Архитектура виртуализации уровня ОС

Для реализации виртуализации уровня операционной системы часто используется продукт OpenVZ.

OpenVZ разрабатывается как патч к исходным текстам ядра Linux. В модифицированном ядре добавлен массив дополнительных сущностей — виртуальных окружений (VE) или контейнеров (CT), а для всех имеющихся объектов (процессы, сокеты и прочие) введены дополнительные поля — номер контейнера, к которому этот объект относится, и номер объекта внутри контейнера. Каждое виртуальное окружение имеет собственный набор квот на потребление системных ресурсов и отдельный каталог для использования в качестве корневой файловой системы. Дополнительные модули ядра — vzdev, vzmon и прочие, отвечают за работу ограничений, мониторинг, эмуляцию сети в контейнере, сохранение и восстановление текущего состояния запущенных контейнеров. К преимуществам OpenVZ, по сравнению с более универсальными инструментами виртуализации, такими как Xen и KVM, относят прозрачный доступ из внешней системы к процессам, файлам и прочим ресурсам в контейнере.

OpenVZ разрабатывается фирмой Parallels как часть более крупного коммерческого продукта под названием Parallels Virtuozzo Containers (PVC) [9]. В число преимуществ Virtuozzo, по сравнению с OpenVZ, входят:

– файловая система VZFS;
– управление через графическую консоль и веб-интерфейс;

- программный интерфейс на базе XML для создания собственных инструментов управления и контроля;
- средства миграции с физическом системы в контейнер и обратно;
- средства контроля за полосой и суммарным потреблением трафика;
- интеграция с Plesk, коммерческой панелью управления хостингом;
- круглосуточная техническая поддержка.

VZFS позволяет совмещать файловые системы контейнеров, при этом базовый образ используется всеми контейнерами, а изменения в нем для каждого контейнера сохраняются раздельно. Преимущества такого подхода:

- место, занимаемое программами на диске, становится фиксированным и не зависит от количества контейнеров, в которые эти программы установлены;
- уменьшается расход оперативной памяти, так как код нескольких экземпляров программы или библиотеки, запущенной из одного и того же исполняемого файла, размещается в памяти в единственном экземпляре;
- обновление программного обеспечения в группе контейнеров выполняется одной командой.

LXC (Linux Containers) — система виртуализации на уровне операционной системы. Данная система сходна с OpenVZ и Linux-VServer для Linux, а также FreeBSD jail и Solaris Containers. LXC основана на технологии cgroups, входящей в ядро Linux, начиная с версии 2.6.29. Ее нельзя рассматривать как законченный продукт, фактически это набор из нескольких совершенно самостоятельных функций ядра Linux и пользовательских утилит, которые позволяют удобно создавать и управлять изолированными контейнерами [10]. Практически вся функциональность LXC представления известными механизмами ядра cgroups и namespaces:

cgroups (Control Groups) — позволяет ограничить аппаратные ресурсы некоторого набора процессов. Под аппаратными ресурсами подразумеваются: процессорное время, память, дисковая и сетевая подсистемы. Набор или группа процессов могут быть определены различными критериями.

Например, это может быть целая иерархия процессов, получающая все лимиты родительского процесса. Кроме этого, возможен подсчет расходуемых группой ресурсов, заморозка (freezing) групп, создание контрольных точке (checkpointing) и их перезагрузка. Для управления этим полезным механизмом существует специальная библиотека libcgroups, в состав которой входят такие утилиты, как cgcreate, cgexec и некоторые другие.

namespaces — пространства имен. Это механизм ядра, который позволяет изолировать процессы друг от друга. Изоляция может быть выполнена в шести контекстах (пространствах имен):

- mount — предоставляет процессам собственную иерархию файловой системы и изолирует ее от других таких же иерархий, по аналогии с chroot;
- PID — изолирует идентификаторы процессов (PID) одного пространства имен от процессов с такими же идентификаторами другого пространства;
- network — предоставляет отдельным процессам логически изолированный от других стек сетевых протоколов, сетевой интерфейс, IP-адрес, таблицу маршрутизации, ARP и прочие реквизиты;
- IPC — обеспечивает разделяемую память и взаимодействие между процессами;
- UTS — изоляция идентификаторов узла, такого как имя хоста (hostname) и домена (domainname);
- user — позволяет иметь один и тот же набор пользователей и групп в рамках разных пространств имен, в каждом контейнере могут быть свой root и любые другие пользователи и группы.

Одно из главных преимуществ LXC — это присутствие его базовых блоков (cgroups и namespaces) во всех современных ядрах Linux. Это означает, что нет необходимости что-то компилировать или использовать стороннее ядро, как в случае с OpenVZ. Единственное, что необходимо установить, это пакет утилит управления (vzctl, Docker, libvirt, systemd).

К системам управления можно отнести Docker.

Docker — это программное обеспечение для автоматизации развертывания и управления приложениями в среде виртуализации на уровне операционной системы, например LXC. Docker позволяет «упаковать» приложение и все его окружение и зависимости в контейнер, который легко переносится на любую Linux-систему с поддержкой cgroups в ядре, а также предоставляет среду по управлению контейнерами.

Идея Docker состоит в том, что производитель сам собирает программу со всеми необходимыми зависимостями и поставляет ее в виде контейнера, работающего на платформе Docker, все что останется, это скачать контейнер и запустить его, а впоследствии обновлять.

Для экономии дискового пространства используется файловая система (ФС) Aufs с поддержкой каскадно-объединенного монтирования. Контейнеры используют образ базовой ОС, а изменения записываются в отдельную область. Также поддерживается размещение контейнеров в файловой системе Btrfs, в котором включен режим копирования при записи. В состав Docker входит демон, являющийся сервером контейнеров, клиентские средства, позволяющие из интерфейса командной строки управлять образами и контейнерами, а также программный интерфейс, позволяющий в стиле REST программно управлять контейнерами. Демон обеспечивает полную изоляцию запускаемых на узле контейнеров на уровне ФС (каждому контейнеру доступна собственная файловая система), на уровне процессов (процессы не имеют доступа к чужой файловой системе контейнера, а ресурсы разделены средствами LXC), на уровне сети (каждому контейнеру привязано сетевое пространства имен и соответствующие виртуальные сетевые интерфейсы).

Набор средств клиента позволяет запускать, приостанавливать и возобновлять процессы в новых контейнерах, останавливать и запускать контейнеры. Серия команд позволяет осуществлять мониторинг запущенных процессов (по аналогии с ps, top). Новые образы создаются из специального сценарного файла (dockerfile), в котором возможно записать все изменения, сделанные в контейнере в новый образ. Все команды могут работать как с docker-демоном локальной системы, так и с сервером docker, доступным

по сети. В интерфейсе командной строки встроены возможности по взаимодействию с публичным репозиторием, именуемым Docker Hub, в котором размещены образы контейнеров, предварительно собранные пользователями, образы можно скачивать в локальную систему, также возможна отправка локально собранных образов в Docker Hub.

Виртуальная инфраструктура является частным облаком, размещаемом на оборудовании провайдера. Инфраструктура представляет собой динамическое распределение ресурсов в соответствии с нормами предприятия. Виртуальная машина использует ресурсы одного компьютера, а виртуальная инфраструктура — всей информационной среды, формируя из компьютеров, а также из подключенных к ним сетей и хранилищ единый пул ресурсов (кластер).

Виртуальная инфраструктура включает в себя следующие компоненты [11]:

- гипервизоры для одного узла для полной виртуализации каждого компьютера;
- пакет услуг инфраструктуры распределенных систем на основе виртуализации (например, управление ресурсами) для оптимального распределения доступных ресурсов между виртуальными машинами;
- решения для автоматизации, обеспечивающие особые возможности оптимизации того или иного ИТ-процесса (например инициализации или восстановления в критических ситуациях).

Благодаря отделению программной среды от исходной аппаратной инфраструктуры, виртуализация позволяет объединить ряд серверов, инфраструктур хранения данных и сетей в единый пул ресурсов, который динамически, безопасно и надежно распределяется между приложениями по мере необходимости. С помощью этого решения организации могут создать вычислительную инфраструктуру с максимальной гибкостью, эффективностью, доступностью и автоматизацией, состоящую из недорогих серверов и соответствующих отраслевому стандарту.

Предпочтение виртуальной инфраструктуре отдают по причинам:

- экономии на обслуживающем персонале, при условии сохранения полной отказоустойчивости системы;
- отсутствии необходимости выделять бюджет на обновление оборудования;
- возможности объединения в общую виртуальную среду целые офисы, которые географически находятся в разных местах мира;
- возможности быстрого масштабирования проекта;
- быстрого доступа к данным.

Под каждый тарифный план создается изолированное частное облако с фиксированным количеством выделенных ему ресурсов. Выделенное частное облако становится гибкой виртуальной оболочкой и функционирующие внутри нее виртуальные машины могут объединяться в виртуальные сети. Изменение их вычислительной мощности происходит в зависимости от решаемых задач, а смена или преобразование тарифных планов производится в режиме реального времени без перебоев в работе.

Управление ресурсами осуществляется через удобный для пользователя веб-интерфейс, что позволяет делать все необходимые операции с виртуальными машинами, такие как:

- создание арендуемых виртуальных серверов самостоятельно;
- изменение конфигурации за несколько минут, часть операций даже без остановки и перезагрузки сервера (для некоторых операционных систем);
- включение, выключение, установка, переустановка ОС и приложений самостоятельно и удаленно;
- осуществление резервного копирование и сохранение состояний (снапшотов) работающих виртуальных машин.

3 СИСТЕМНЫЙ АНАЛИЗ ВИРТУАЛЬНОЙ ИНФРАСТРУКТУРЫ

На данный момент при анализе и синтезе сложных программных и аппаратных систем все чаще используется системный подход. Важным моментом для системного подхода является определение структуры системы — совокупности связей между элементами системы, отражающих их взаимодействие.

Принципы системного анализа — это положения общего характера, являющиеся обобщением опыта работы человека со сложными системами. Пренебрежение принципами при проектировании любой нетривиальной технической системы, непременно приводит к потерям того или иного характера, от увеличения затрат в процессе проектирования до снижения качества и эффективности конечного продукта.

Системный анализ выполнен в соответствии с [12].

3.1 Принцип конечной цели

Принцип конечной цели — основополагающий принцип системного анализа. Конечная цель имеет абсолютный приоритет, в системе все должно быть подчинено достижению конечной цели. Принцип имеет несколько правил:

- для проведения системного анализа необходимо в первую очередь, сформулировать цель функционирования системы, так как расплывчатые, не полностью определенные цели влекут за собой неверные выводы;
- анализ системы следует вести на базе первоочередного уяснения основной цели исследуемой системы, что позволит определить ее существенные свойства, показатели качества и критерии оценки;
- при синтезе систем любая попытка изменения или совершенствования должна быть в первую очередь рассмотрена с позиции его полезности в достижении конечной цели;

– цель функционирования искусственной системы задается, как правило, системой, в которой исследуемая система является составной частью.

При использовании данного принципа, разрабатываемая виртуальная инфраструктура будет рассматриваться в виде «черного ящика», функционирование которого описывается формулой (3.1):

$$Y=F(X, Z) \quad (3.1)$$

где Y — выходной вектор системы, который функционально зависит от входного вектора X и вектора внутреннего состояния системы Z (рисунке 3.1). Входными данными (вектор X) будут являться запросы пользователей на обработку и обслуживание. Внутреннее состояние системы (вектор Z) представляет собой создание услуг по требованию пользователя. Выходными данными (вектор Y) будут являться предоставление доступа к услугам пользователя.

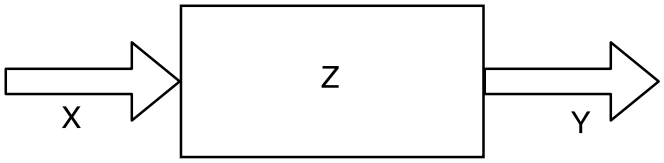

Рисунок 3.1 – Проектируемая система в виде черного ящика

В соответствии с данным принципом должна быть четко сформулирована конечная цель — назначение проектируемой системы и сформирован список функций, которые должна выполнять система.

Цель проектирования — разработка виртуальной инфраструктуры для реализации облачных услуг. Список функций проектируемой системы:

– Ф1 — прием и регистрация обращений пользователей;
– Ф2 — идентификация и обработка инцидентов и запросов на обслуживание;
– Ф3 — создание, смена, обновление и удаление услуг по требованию;
– Ф4 — предоставление доступа к услугам;
– Ф5 — мониторинг состояния инфраструктуры.

3.2 Принцип единства

Принцип единства — это совместное рассмотрение системы как целого и как совокупности частей. Принцип ориентирован на декомпозицию с сохранением целостных представлений о системе. Система должна рассматриваться и как целое, и как совокупность элементов. Расчленение системы необходимо производить, сохраняя целостное представление о системе. Принцип подразумевает выделение подсистем, композиция которых в совокупности со связями позволяет выполнять все функции проектируемой системы, определяет ее структуру и может быть рассмотрена как единая, целостная система

На основании функций проектируемой системы, представленных выше, в ней можно выделить следующие подсистемы:

а) подсистема взаимодействия с пользователем;

б) подсистема управления услугами;

в) подсистема управления инфраструктурой;

г) подсистема защиты и обеспечения целостности данных.

На рисунке 3.2 представлена схема взаимодействия между подсистемами.

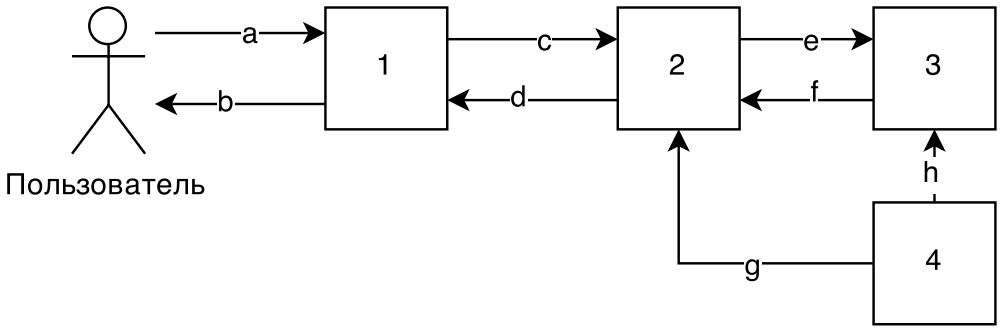

Рисунок 3.2 – Взаимодействие между подсистемами и их связь с окружающей средой

Обозначения, приведенные на рисунке 3.2 требуют пояснения:

a — информация, предоставляемая пользователем, передается на сервер;

b — выходная информация (результат выполнения);

c — проверка корректности переданных данных;

d — в случае неправильно введенных данных, возвращается управление к подсистеме взаимодействия с пользователем;

e — создание услуги;

f — возврат информации о созданной услуге;

g — обеспечение целостности информации об услуге;

h — обеспечение целостности данных пользователя.

3.3 Принцип связности

Любая часть системы должна рассматриваться со всеми своими связями с окружающими ее объектами, как внешними по отношению ко всей системе, так и внутренними — другими элементами системы. Если некоторая подсистема имеет связи только с внешней средой, то есть смысл реализовать ее в виде отдельной системы. Подсистема, не связанная ни с внешней средой, ни с другой подсистемой, является избыточной и должна быть удалена из системы.

Подсистема взаимодействия с пользователем представлена на рисунке 3.3.

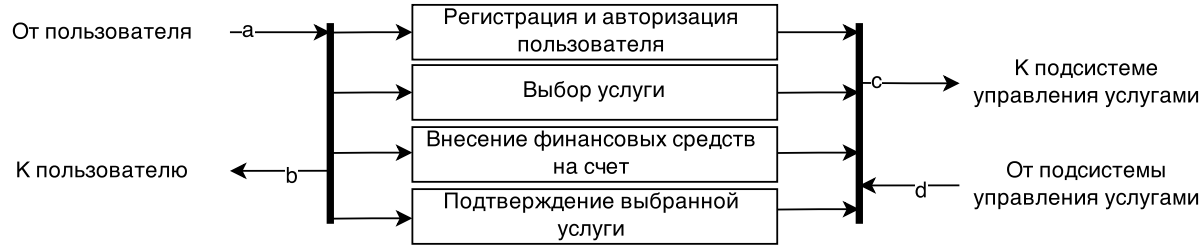

Рисунок 3.3 – Внутренние и внешние связи подсистемы взаимодействия с пользователем

Подсистема управления услугами представлена на рисунке 3.4.

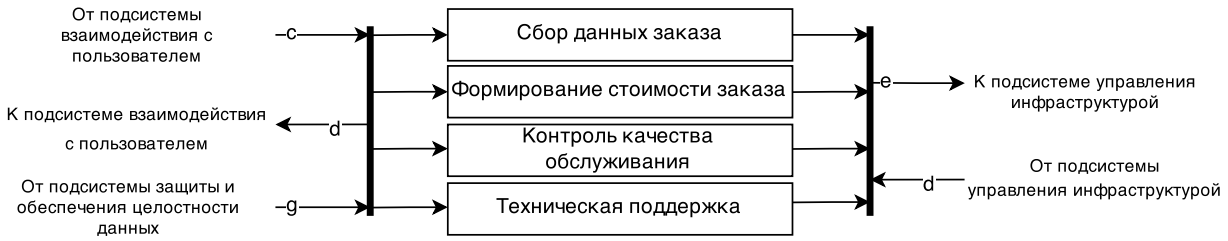

Рисунок 3.4 – Внутренние и внешние связи подсистемы управления услугами

Подсистема управления инфраструктурой представлена на рисунке 3.5.

Рисунок 3.5 – Внутренние и внешние связи подсистемы управления инфраструктурой

Подсистема защиты и обеспечения целостности данных представлена на рисунке 3.6.

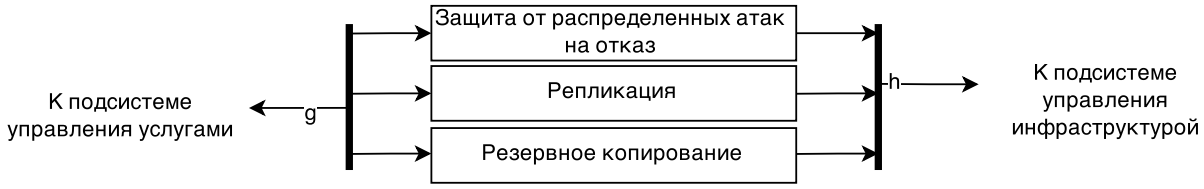

Рисунок 3.6 – Внутренние и внешние связи подсистемы защиты и обеспечения целостности данных

3.4 Принцип модульности

В соответствии с этим принципом, выделение модулей системы, если таковые имеются, продуктивно и оправдано. Под модулями здесь понимаются относительно автономные и достаточно простые блоки, выполняющие ограниченный набор функций. Модуль в отличие от подсистем, которые имеют нерегулярную структуру и, как правило, несут определенную функцию,

частично отражающую функцию системы, имеет регулярную структуру и характерные для него внутренние и внешние связи. Принцип указывает на возможность вместо части системы исследовать совокупность ее входных и выходных воздействий. Выделению модулей соответствует декопмозиция сложной задачи на множество более простых подзадач.

Принцип модульности для разрабатываемой системы поясняется с помощью рисунка 3.7, описывающего разбиение на модули системы взаимодействия с пользователем.

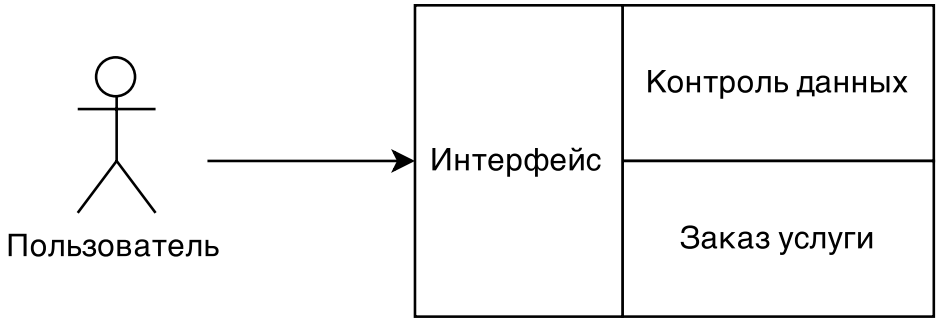

Рисунок 3.7 – Принцип модульности на примере подсистемы взаимодействия с пользователем

Излишняя детализация не требуется, поэтому остальные системы на модули принято решение не разбивать.

3.5 Принцип функциональности

Принцип определяет первичность функции по отношению к структуре, так же, как цель первична для функции. Другими словами, цель определяет функции системы, а функции определяют ее структуру — совокупность элементов с их связями. Структура же, в свою очередь, определяет параметры системы. В случае придания системе новых функций полезно пересматривать ее структуру, а не пытаться втиснуть новую функцию в старую схему. Поскольку выполняемые функции составляют процессы, то целесообразно рассматривать отдельно процессы, функции, структуры.

Функции подсистем приведены в пункте 3.1.

Матрица инциденций функций системы и функций назначения подсистем приведена в таблице 3.1.

Таблица 3.1 – Матрица инциденций

Функции	Подсистемы				Виртуальная инфраструктура
	1	2	3	4	
Ф1	+				+
Ф2	+				+
Ф3		+			+
Ф4		+			+
Ф5			+	+	+

В матрице инциденций знаком «+» обозначены функции, которые реализуются для каждой из подсистем.

Детализация функций подсистемы на примере подсистемы взаимодействия с пользователем:

а) регистрация и авторизация пользователя в платежной системе;

б) обеспечение перечня услуг;

в) обеспечение возможных способов оплаты услуг;

г) возможность уточнения или изменения услуги.

Входными данными для подсистемы является информация о пользователе, а выходными — подтвержденный заказ услуг.

3.6 Принцип иерархии

Полезно введение иерархии частей и их ранжирование, что упрощает разработку системы и устанавливает порядок рассмотрения частей. Разработка иерархий классов является нетривиальной задачей. Грамотно спроектированные иерархии классов позволяют создавать высокоэффективные системы, плохо спроектированная иерархия приводит к созданию сложных и запутанных систем.

Выполнение принципа иерархичности для разрабатываемой системы на примере подсистемы защиты и обеспечения целостности данных проиллюстрировано на рисунке 3.8.

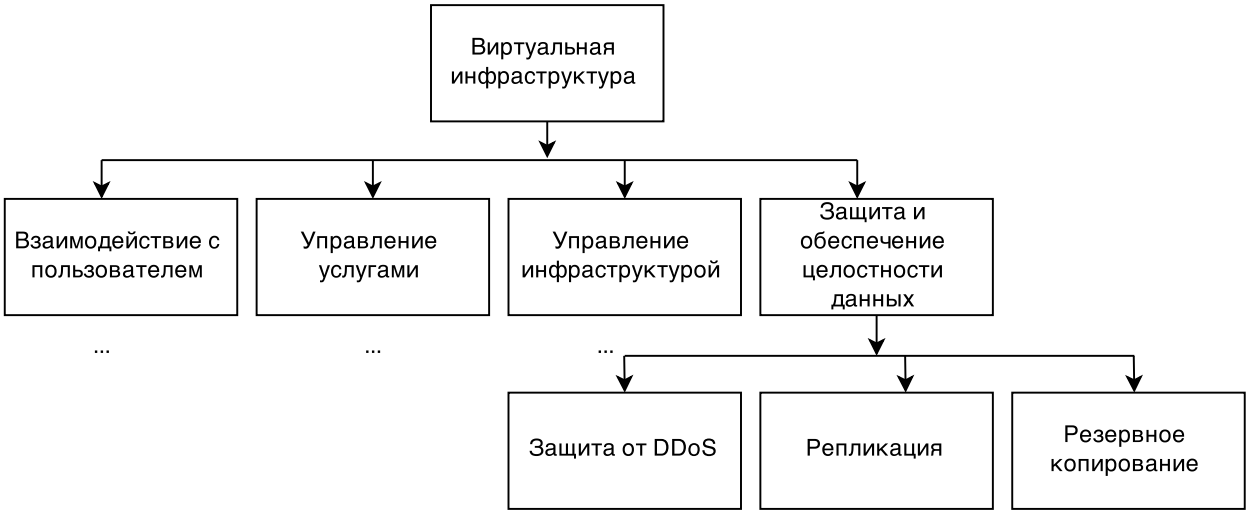

Рисунок 3.8 – Принцип иерархии на примере подсистемы защиты и обеспечения целостности данных

3.7 Принцип сочетания централизации и децентрализации

Степень централизации должна быть минимальной, обеспечивающей выполнение поставленной цели. Соотношение централизации и децентрализации определяется уровнями, на которых вырабатываются и принимаются управленческие решения.

Недостатком децентрализованного управления является увеличение времени адаптации системы, которое существенно влияет на функционирование системы в быстро меняющихся средах. То, что в централизованных системах можно сделать за короткий промежуток времени, в децентрализованной системе будет осуществляться весьма медленно. Недостатком централизованного управления является сложность управления из-за огромного потока информации, подлежащей переработке в старшей системе управления. В медленно меняющейся обстановке децентрализованная часть системы успешно справляется с адаптацией поведения системы к среде и с достижением глобальной цели системы за счет оперативного управления, а при резких изменениях среды осуществляется централизованное управление по переводу системы в новое состояние.

Например, можно выполнить декомпозицию подсистемы взаимодействия с пользователем таким образом:

а) подсистема работы с пользователями;

б) подсистема работы пользователей с услугами;

в) подсистема учета финансовых средств.

Такое разбиение позволит реализовать полученные подмножества в виде отдельных модулей.

3.8 Принцип развития

Учет изменяемости системы, ее способности к развитию, адаптации, расширению, замене частей, накапливанию информации. В основу синтезируемой системы требуется закладывать возможность развития, наращивания усовершенствования. Обычно расширение функций предусматривается за счет обеспечения возможности включения новых модулей, совместимых с уже имеющимися. С другой стороны, при анализе принцип развития ориентирует на необходимость учета предыстории развития системы и тенденций, имеющихся в настоящее время, для вскрытия закономерностей ее функционирования.

Одним из способов учета этого принципа разработчиками является рассмотрение системы относительно ее жизненного цикла. Условными фазами жизненного цикла являются: проектирование, изготовление, ввод в эксплуатацию, эксплуатация, наращивание возможностей (модернизация), вывод из эксплуатации (замена), уничтожение.

Отдельные авторы этот принцип называют принципом изменения (историчности) или открытости. Для того чтобы система функционировала, она должна изменяться, взаимодействовать со средой.

Проектируемая система может быть развита следующим образом:

- добавлением модуля конвертации валюты для оплаты услуги;
- расширением перечня услуг;
- внедрением дополнительных технологий виртуализации;
- увеличением штата технической поддержки;
- введением расширенного мониторинга;
- многоуровневой защитой от атак на отказ;

– расширением памяти на системах хранения данных;
– многоуровневой репликацией и резервным копированием.

3.9 Принцип учета случайностей

Можно иметь дело с системой, в которой структура, функционирование или внешние воздействия не полностью определены.

Сложные системы не всегда подчиняются вероятностным законам. В таких системах можно оценивать «наихудшие» ситуации и проводить рассмотрение для них. Этот способ обычно называют методом гарантируемого результата, он применим, когда неопределенность не описывается аппаратом теории вероятностей.

При наличии информации о вероятностных характеристиках случайностей можно определять вероятностные характеристики выходов в системе.

События и действия, некорректные с точки зрения правил функционирования системы:

– попытка заказа услуги без наличия финансовых средств на счету;
– неподтвержденные заказы услуг;
– сбой в работе аппаратуры, в частности устройств хранения данных;
– сбой сети в в пределах дата-центра или на магистрали;
– отсутствие своевременной работы технической поддержки;
– ошибки и уязвимости в используемых программных платформах.

Кроме того, необходимо вести контроль успешности и целостности проведения операций с компонентами системы, данными пользователей и корректно обрабатывать исключения, возникновение которых возможно в процессе работы системы.

Перечисленные выше принципы обладают высокой степенью общности. Такая интерпретация может привести к обоснованному выводу о незначимости какого-либо принципа. Однако знание и учет принципов позволяет лучше увидеть существенные стороны решаемой проблемы, учесть весь комплекс взаимосвязей, обеспечить системную интеграцию.

4 ОПИСАНИЕ ВИРТУАЛЬНОЙ ИНФРАСТРУКТУРЫ

4.1 Назначение виртуальной инфраструктуры

5 РУКОВОДСТВО АДМИНИСТРАТОРА

5.1 Расположение серверов в сетевой инфраструктуре

Во всей инфраструктуре на физических серверах первого дата-центра располагаются:

- shared-хостинг (10.0.0.100);
- виртуализация OpenVZ (10.0.0.200);
- виртуализация KVM (10.0.0.300);
- сервер резервного копирования (10.0.0.400);
- физические сервера клиентов (10.0.0.500 — 10.0.0.700).

В свою очередь некоторые важные сервисы инфраструктуры располагаются на виртуальных машинах, такие как:

- сервер биллинга (10.0.0.201), OpenVZ;
- сервер управления IP-адесами (10.0.0.301), KVM;
- сервер DNS 1 (10.0.0.302), KVM.

Сервер мониторинга и сервер DNS 2 располагаются в виртуальных машинах KVM в другом дата-центре. Сервер shared-хостинга выполняет роль главного (master) DNS-сервера.

5.2 Сервер shared-хостинга

Сервер shared-хостинга является самым уязвимым местом всей инфраструктуры из-за большого количества и плотности клиентов на сервере. Распределение ресурсов происходит за счет встроенных в ядро Linux механизмов.

На сервере установлен и настроен веб-сервер в составе следующего ПО:

- роль http-сервера выполняет связка между Apache HTTP Server, кэширующего и проксирующего http-сервера nginx и обработчика динамических запросов php-fpm;
- memcached кэширует динамические запросы в оперативной памяти;
- роли серверов баз данных выполняют MySQL и PostgreSQL.

nginx принимает запросы к серверу с порта 80 и решает каким образом обрабатывать запрос, если это запрос на получение статического файла, то nginx сам его обрабатывает и в случае необходимости кэширует. Если же запрашивается динамическое содержимое, то nginx передает запрос на бэкенд, где его, в зависимости от настройки виртуального хоста сайта обрабатывает либо php-fpm, либо встроенный модуль Apache mod_php. nginx не может кэшировать динамические запросы, поэтому для этого имеется memcached, который работает в связке с Apache и позволяет кэшировать динамические запросы. Для правильной работы Apache с memcached необходима установка модуля PHP php5-memcached:

```
# apt—get install php5—memcached
```

В качестве связки для работы почты выступает SMTP-сервер Exim Internet Mailer, запросы по протоколам POP3 и IMAP принимает Dovecot. В качестве антиспам-связки выступает Postgrey, Spamassasin, ClamAV и OpenDKIM.

ProFTPd необходим для работы пользователей по протоколу FTP.

Сервер управления временем NTP обращается к следующим серверам за уточнением времени на сервере:

```
# cat /etc/ntp.conf | grep ^server | awk '{print $2}'
0.debian.pool.ntp.org
1.debian.pool.ntp.org
2.debian.pool.ntp.org
3.debian.pool.ntp.org
```

Конфигурация SSH требует отдельного пояснения. Доступ к SSH по стандартному порту закрыт, это позволяет избавиться от большинства злоумышленников, которые пытаются скомпрометировать пароль доступа к серверу. Также запрещен вход от пользователя root, для этого создан отдельный пользователь. Также можно ограничить доступ к серверу по SSH, оставив только доверенные адреса или подсети адресов.

```
Adding user for SSH:
# useradd sshuser
```

```
# passwd sshuser
SSH config:
# cat /etc/ssh/sshd_config
#change SSH port 222:
Port 222
#close root login:
PermitRootLogin yes
#allow IP—addresses:
Match host 10.0.0.111, 10.0.0.222
  #root for allowed IP's:
  PermitRootLogin yes
Restart SSH:
# service ssh restart
```

Разрешен вход на сервер по ключам:

```
Key Generating:
user@10.0.0.111~$ ssh—keygen
user@10.0.0.111~$ ssh—keygen —p
Copy private key to server:
user@10.0.0.111~$ ssh—copy—id —p 222 root@10.0.0.100
Reconnect to server:
user@10.0.0.111~$ ssh —p 222 root@10.0.0.100
```

Резервные копии с сервера делаются средствами панели ISPmanager в ночное время, применяется инкрементальный метод резервного копирования, когда сначала делается полная копия, а затем инкрементальные копии только измененных данных.

На сервере функционирует скрипт блокировки адресов, которые слишком часто подбирают пароли доступа к панели администратора наиболее популярных CMS, таких как WordPress и Joomla. Скрипт находится в открытом доступе и имеет простое использование:

```
# cat /etc/nginx/blockips.sh
#!/bin/bash
# admin@amet13.name
# v0.2 30.06.2014
command=$(cat /var/log/nginx/access.log | \
```

```
grep ”administrator/index.php \|wp—login.php ” | \
cut —f1 —d ” ” | sort | \
uniq —c | sort —n | \
awk ’{ if ($1 > 2999) print $2}’)
# Add new ip’s to database
for word in $command; do
  # Is that ip already in database?
  grep $word /etc/nginx/blockips.conf > /dev/null
  if (( $? )); then
    sed —i ”s/allow all;/deny $word;\nallow all;/” \
    /etc/nginx/blockips.conf
  fi
done
exit 0
```

Основные конфигурации настроек веб-сервера хранятся в:

```
/etc/nginx/nginx.conf
/etc/apache2/apache2.conf
/etc/php5/{apache2,cgi,cli,fpm}/php.ini
/etc/mysql/my.cnf
/etc/postgresql/9.1/main/postgresql.conf
```

Для сканирования уязвимостей на сайтах клиентов используется утилита maldet, которая находит подозрительные файлы в системе и составляет отчет по найденным файлам. Принцип пользования утилитой:

```
Instalaltion:
# cd /tmp/
# wget http://www.rfxn.com/downloads/maldetect—current.tar.gz
# tar xfz maldetect—current.tar.gz
# cd maldetect—*
# ./install.sh
Run scanning (—b —— background mode):
# maldet —b ——scan—all /var/www/
Show reports list:
# maldet ——report list
Report view:
# maldet ——report 021715—1414.3266
```

```
Send infected files to quarantine:
# maldet —q 021715—1414.3266
```

На сервере shared-хостинга в сайтах пользователей большое количество уязвимостей, которые часто используются для рассылки спама с сервера. Для обнаружения большого числа рассылок с сервера написан плагин для системы мониторинга Nagios, который контролирует число писем в почтовой очереди, а также проверяет наиболее популярные организации, которые собирают данные о спам-серверах.

Для просмотра списка очереди используется команда exim -bp или mailq. Таким образом можно просмотреть количество писем для разных доменов и их количество:

```
# exim —bp | exiqsumm —c —s | head —10
Count Volume Oldest Newest Domain
————— —————— —————— —————— ——————
12151 1378KB  14h    0m spamsite.ru > host.ru
   93  264KB   14h    5m spamsite.ru > gmail.com
   16  154KB   13h   11h site.ru > kk.com
    8   77KB   13h   11h site.ru > ww.com
...
```

В данном случае очевидна рассылка писем с сайта spamsite.ru. Посмотреть ID письма а также содержимое заголовков и тела письма можно командами:

```
# exiqgrep —b —f 'spamsite.ru' | head —1
1YyRgs—0000zD—Qy From: <fobos@spamsite.ru> To: maria@host.ru
# exim —Mvh 1YyRgs—0000zD—Qy
# exim —Mvb 1YyRgs—0000zD—Qy
```

Если сайт действительно заражен, то следует отключить его, оповестить пользователя о закрытии сайта и необходимости избавиться от вредоносного кода и удалить очередь сообщений из этих писем:

```
# exiqgrep —b —f 'spamsite.ru' | awk '{print $1}' | xargs exim —
   Mrm
```

В случае неработоспособности панели ISPmanager 5 возможна ее перезагрузка:

```
# pgrep –l core
4437 core
7129 core
21194 core
# killall core
```

Обновление компонентов системы:

```
# apt–get update
# apt–get upgrade
```

5.3 Сервер виртуализации OpenVZ

Список существующих в системе контейнеров можно узнать командой vzlist:

```
# vzlist –a
     CTID    NPROC STATUS IP_ADDR  HOSTNAME
      145      134 running 10.0.0.245 site1.ru
      146       41 running 10.0.0.246 site2.ru
      205        – stopped 10.0.0.205 site3.ru
...
```

В заголовках показывается показывается ID контейнера, число процессов в контейнере, текущий статус, IP-адрес и имя контейнера.

Список всех доступных шаблонов операционной системы и список шаблонов установленных локально:

```
# vztmpl–dl ––list–all
...
# vztmpl–dl ––list–local
centos–7–x86_64
debian–7.0–x86_64
ubuntu–14.04–x86_64
```

Каждый контейнер имеет свой конфигурационный файл, который хранится в каталоге /etc/sysconfig/vz-scripts/. Именуются эти файлы по CTID

контейнера. Например, для контейнера с CTID=101, файл будет называться 101.conf.

При создании контейнера можно использовать типовую конфигурацию для VPS. Типовые файлы конфигураций находятся в том же каталоге:

```
# ls –1 /etc/sysconfig/vz–scripts/ | grep sample
ve–basic.conf–sample
ve–ispbasic.conf–sample
ve–light.conf–sample
ve–vswap–1024m.conf–sample
ve–vswap–1g.conf–sample
ve–vswap–256m.conf–sample
ve–vswap–2g.conf–sample
ve–vswap–4g.conf–sample
ve–vswap–512m.conf–sample
```

В этих конфигурационных файлах описаны контрольные параметры ресурсов, выделенное дисковое пространство, оперативная память и прочие ресурсы. На базе уже существующего фала конфигурации создан типовой файл, подходящий для большинства контейнеров именуемый ve-custom.conf-sample. Таким образом, при использовании этого конфигурационного файла, будет создаваться контейнер, которому будет доступен 1 Гб выделенного дискового пространства, 128 Мб оперативной памяти и 128 Мб swap. В дальнейшем, при создании контейнеров следует использовать данный конфигурационный файл.

Создание контейнера:

```
# vzctl create 101 ––ostemplate debian–7.0–x86_64 ––config
   custom
```

где 101 — CTID контейнера, --ostemplate — шаблон ОС, --config — используемый шаблон конфигурационного файла.

Первым запуском контейнера необходимо установить его IP адрес, hostname, указать DNS сервер и задать пароль суперпользователя.

Для настройки VPS используется команда vzctl set. Для того, чтобы контейнер запускался при старте хост-компьютера (например после перезагрузки), необходимо использовать команду:

```
# vzctl set 101 --onboot yes --save
CT configuration saved to /etc/vz/conf/101.conf
```

При использовании ключа --save, сохраняются параметры контейнера в соответствующий конфигурационный файл. Аналогично можно задать hostname:

```
# vzctl set 101 --hostname site.com --save
CT configuration saved to /etc/vz/conf/101.conf
```

Установка IP адреса:

```
# vzctl set 101 --ipadd 10.0.0.210 --save
CT configuration saved to /etc/vz/conf/101.conf
```

Адрес DNS сервера (в большинстве случаев адрес DNS совпадает с адресом хост-компьютера, поэтому можно вместо адреса указать параметр inherit):

```
# vzctl set 101 --nameserver inherit --save
CT configuration saved to /etc/vz/conf/101.conf
```

Установка пароля суперпользователя:

```
# vzctl set 101 --userpasswd root:p@ssw0rd
Starting container...
...
Unmounting file system at /vz/root/101
Unmounting device /dev/ploop37965
Container is unmounted
```

Пароль будет установлен в VPS, в файл `/etc/shadow` и не будет сохранен в конфигурационный файл контейнера. Если же пароль будет утерян или забыт, то можно будет просто задать новый.

После настроек нового контейнера, его можно запустить:

```
# vzctl start 101
Starting container...
```

```
Opening delta /vz/private/101/root.hdd/root.hdd
Adding delta dev=/dev/ploop37965 img=/vz/private/101/root.hdd/
   root.hdd (rw)
Mounting /dev/ploop37965p1 at /vz/root/101 fstype=ext4 data='
   balloon_ino=12,'
Container is mounted
Adding IP address(es): 10.0.0.210
Setting CPU units: 1000
Container start in progress...
```

Проверка сетевых интерфейсов внутри гостевой ОС:

```
# vzctl exec 101 ifconfig | grep "lo\|venet" -A 1
lo      Link encap:Local Loopback
        inet addr:127.0.0.1 Mask:255.0.0.0
--
venet0  Link encap:UNSPEC HWaddr
   00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
        inet addr:127.0.0.2 P-t-P:127.0.0.2 Bcast:0.0.0.0 Mask
           :255.255.255.255
--
venet0:0 Link encap:UNSPEC HWaddr
   00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
        inet addr:10.0.0.210 P-t-P:192.168.0.101 Bcast
           :192.168.0.101 Mask:255.255.255.255
```

Должны присутствовать сетевые интерфейсы:

- lo (127.0.0.1);
- venet0 (127.0.0.2);
- venet0:0 (10.0.0.210).

Если сеть в порядке, то можно соединиться к контейнеру по SSH с хост-компьютера:

```
# ssh root@10.0.0.210
root@10.0.0.210's password: p@ssw0rd
```

Вход в контейнер напрямую с хост-компьютера осуществляется командой vzctl enter:

```
# vzctl enter 101
```

```
entered into CT 101
root@site:/#
```

Выход из контейнера:

```
root@site:/# exit
logout
exited from CT 101
```

Для остановки контейнера используется команда vzctl stop. Для полной остановки контейнера, системе требуется немного времени. Иногда нужно выключить VPS как можно быстрее, например, если контейнер был подвержен взлому. Для того чтобы срочно выключить VPS, нужно использовать ключ --fast:

```
# vzctl stop 101 ––fast
Killing container ...
Container was stopped
Unmounting file system at /vz/root/101
Unmounting device /dev/ploop37965
Container is unmounted
```

Для перезапуска контейнера можно использовать команду vzctl restart.

Для того чтобы удалить контейнер, его нужно сначала остановить:

```
# vzctl stop 101
Stopping container ...
Container was stopped
Unmounting file system at /vz/root/101
Unmounting device /dev/ploop37965
Container is unmounted
```

Для удаления используется команда:

```
# vzctl destroy 101
CTID 101 deleted unmounted down
```

Команда выполняет удаление частной области сервера и переименовывает файл конфигурации, дописывая к нему .destroyed.

Иногда бывает нужно выполнить команду на нескольких VPS. Для этого можно использовать команду:

```
# for i in ‘vzlist —o veid —H‘; do \
> echo ”VPS $i”; vzctl exec $i command; done
```

Например, можно узнать, сколько времени работают все запущенные контейнеры:

```
# for i in ‘vzlist —o veid —H‘; do \
> echo ”VPS $i”; vzctl exec $i uptime; done
VPS 101
 05:45:01 up 2 min, 0 users, load average: 0.01, 0.02, 0.03
VPS 102
 05:46:01 up 1 min, 0 users, load average: 0.04, 0.05, 0.06
```

Если на хост-ноде наблюдается высокое значение Load Average, то в первую очередь следует посмотреть данные значения непосредственно у контейнеров:

```
# vzlist —o veid,laverage
     CTID    LAVERAGE
      145 0.00/0.01/0.05
      146 0.00/0.00/0.00
      148 0.60/0.51/0.34
...
```

В случае обнаружения нагружающего контейнера, стоит войти в него и устранить причину нагрузки.

Непосредственно с хост-ноды можно узнать, какому контейнеру принадлежит процесс:

```
# vzpid 1048108
Pid   CTID Name
1048108 145 php—cgi
```

Свободное место в контейнере можно узнать командой df:

```
# df —h
Filesystem      Size Used Avail Use% Mounted on
/dev/mapper/vg_id3611—lv_root
                 50G 5,6G 42G 12% /
tmpfs            16G   0  16G  0% /dev/shm
/dev/cciss/c0d0p1 485M 276M 184M 61% /boot
```

```
/dev/mapper/vg_id3611—lv_vz
                489G 403G 62G 87% /vz

# df —i
Filesystem       Inodes IUsed IFree IUse% Mounted on
/dev/mapper/vg_id3611—lv_root
                3276800 55229 3221571 2% /
tmpfs           4101290  1 4101289 1% /dev/shm
/dev/cciss/c0d0p1 128016 96  127920  1% /boot
/dev/mapper/vg_id3611—lv_vz
               32546816 248 32546568 1% /vz
```

Подключение модуля TUN для контейнера (необходимо для работы OpenVPN. Прежде чем запускать контейнер нужно убедиться, что модуль TUN загружен на хост-ноде:

```
# lsmod | grep tun
```

В случае, если он не загружен, загрузить его можно следующей командой:

```
# modprobe tun
# lsmod | grep tun
tun   1957  0
```

Разрешаем использовать устройство TUN контейнеру:

```
# vzctl set 101 ——devnodes net/tun:rw ——save
CT configuration saved to /etc/vz/conf/101.conf
# vzctl set 101 ——devices c:10:200:rw ——save
CT configuration saved to /etc/vz/conf/101.conf
# vzctl set 101 ——capability net_admin:on ——save
CT configuration saved to /etc/vz/conf/101.conf
```

Запускаем контейнер:

```
# vzctl start 101
Starting container...
...
Container start in progress...
```

Создаем в контейнере собственное устройство TUN:

```
# vzctl exec 101 mkdir –p /dev/net
# vzctl exec 101 mknod /dev/net/tun c 10 200
# vzctl exec 101 chmod 600 /dev/net/tun
```

Для пользователей в контейнерах доступен файерволл IPTables, для его работы в файле /etc/vz/vz.conf должна присутствовать строка:

```
IPTABLES=”ipt_owner ipt_REDIRECT ipt_recent ip_tables
   iptable_filter iptable_mangle ipt_limit ipt_multiport ipt_tos
   ipt_TOS ipt_REJECT ipt_TCPMSS ipt_tcpmss ipt_ttl ipt_LOG
   ipt_length ip_conntrack ip_conntrack_ftp ipt_state iptable_nat
    ip_nat_ftp”
```

ля того, чтобы для контейнеров был доступен FUSE, его необходимо включить на хост-ноде:

```
# modprobe fuse
```

Проверить, что модуль успешно подключен:

```
# lsmod | grep fuse
fuse 92980 54
```

Также необходимо добавить автозагрузку модуля при перезапуске хост-ноды:

```
# vim /etc/rc.local
#!/bin/sh
...
modprobe fuse
```

Проброс FUSE для контейнера 101:

```
# vzctl stop 101
# vzctl set 101 ––devnodes fuse:rw ––save
# vzctl set 101 ––devices c:10:229:rw ––save
# vzctl start 101
```

В контейнере проверяем, пробросилось ли устройство:

```
[root@site /]# ls /dev/fuse
/dev/fuse
```

Резервное копирование контейнеров осуществляется утилитами vzdump и vzrestore:

```
# vzdump ––suspend ––compress ––dumpdir /root/ ––exclude–path
   /tmp/ 101
# vzrestore /vz/dump/vzdump–openvz–101–2015_01_01–12_12_12.tar
   103
```

Для работы OpenVZ с файлами используется ploop, так как simfs является устаревшим методом. Диски с файловыми системами контейнеров хранятся в /vz/private и имеют имя root.hdd.

Смена размера диска для контейнера:

```
# vzctl set 103 ––diskspace 2G ––save
dumpe2fs 1.41.12 (17–May–2010)
Changing balloon size old_size=1182793728 new_size
=217055232
Successfully truncated balloon from 1182793728 to 217055232
bytes
UB limits were set successfully
CT configuration saved to /etc/vz/conf/103.conf
```

5.4 Сервер виртуализации KVM

6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

7 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

8 БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ

8.1 Краткая характеристика помещения

Согласно санитарным нормам, ширина стола должна быть не менее 60 см, глубина не менее 80 см, также необходимо обеспечить расстояние между боковыми поверхностями мониторов не менее 1.2 м.

На рисунке 8.1 изображена планировка и размещение оборудования на рабочем месте.

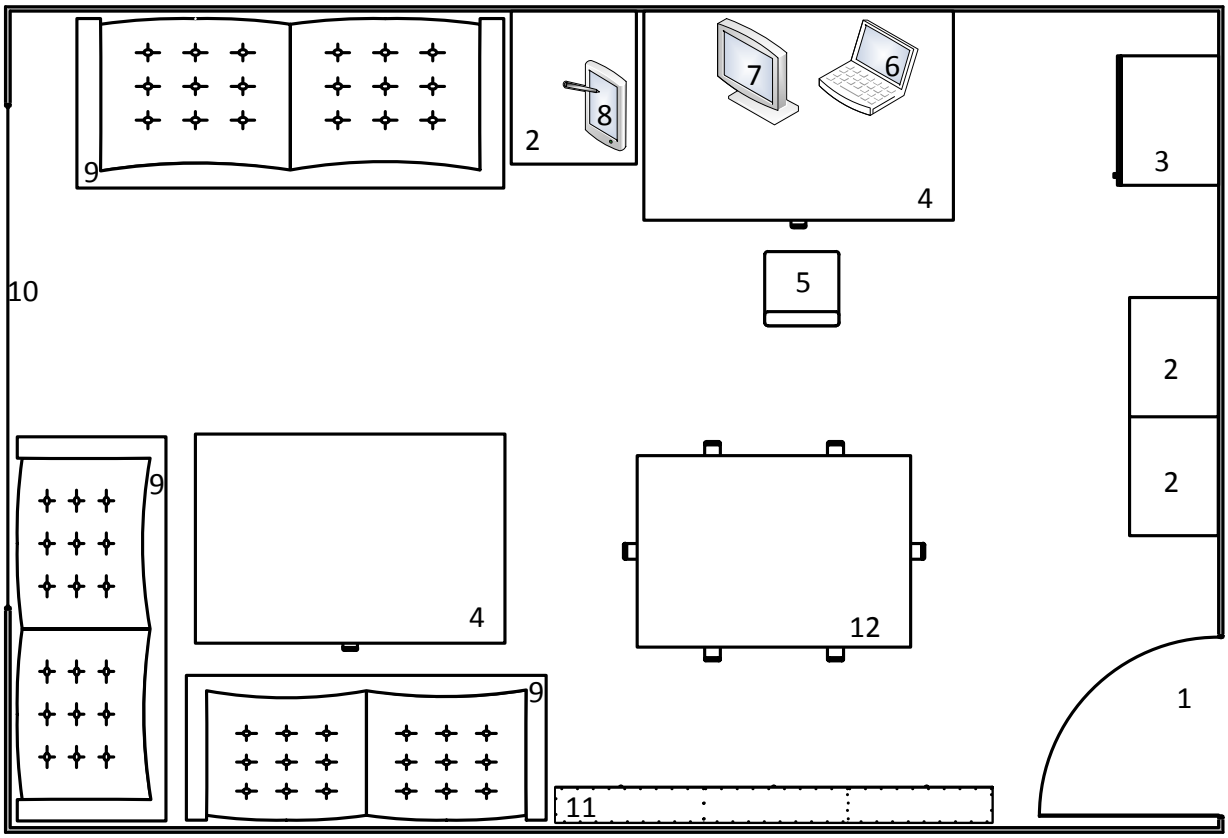

Рисунок 8.1 – Планировка и размещение оборудования на рабочем месте

1 — дверь; 2 — тумбочка; 3 — холодильник; 4 — стол; 5 — кресло; 6 — ноутбук; 7 — монитор; 8 — планшет; 9 — кровать; 10 — окно; 11 — шкаф; 12 — обеденный стол

8.2 Микроклимат

Метеорологические условия (микроклимат) характеризуются следующими параметрами:

- температурой воздуха;
- относительной влажностью;
- скоростью движения воздуха на рабочем месте;
- барометрическим давлением.

В комнате, где находится персональный компьютер, должен поддерживаться определенный температурный режим для нормальной эксплуатации ЭВМ и условий труда администратора.

Параметры воздушной среды в кабинете должны соответствовать требованиям ГОСТ 12.1.005-88 «Воздух рабочей зоны». При этом необходимо учитывать, что работа администратора относится к разряду легких работ (разряд 1а — затраты энергии до 150 ккал/час), а кабинет — к производственным помещениям.

В помещении имеются источники избыточного тепла:

- тепловыделение от ноутбука;
- тепло, выделяемое персоналом.

Тепловыделение от светильников отсутствует, так как используется люминисцентное освещение. Для поддержания оптимального уровня температуры используется как естественная вентиляция (через двери и окна путем проветривания), так и искусственная (при помощи вентилятора), так как в текущих условиях проживания невозможно установить кондиционер. Для обогрева помещения в зимнее время используется водяное отопление.

8.3 Шум и вибрация

Источниками шума в помещении являются:

- непосредственно персональный компьютер (вентиляторы охлаждения процессора и видеокарты);
- разговорная речь;
- шум вне рабочей зоны.

Постоянный шум оказывает отрицательное воздействие на человека, как биологически, так и психологически, что отражается на качестве работы и общей производительности труда сотрудников. Снижается производительность труда и повышается количество допущенных ошибок, некоторые из которых могут быть критическими.

В помещениях для администраторов по ГОСТ 12.1.003-83 «Шум. Общие требования безопасности» допустимые уровни звукового давления приведены в таблице 8.2. Допустимый уровень звука — 50 дБА, при работающем оборудовании в кабинете ожидаемый уровень звука — 40-48 дБА.

Таблица 8.2 – Допустимые уровни звукового давления

Тип помещения	Допустимые уровни звукового давления
Служебные помещения	50 дБА
Помещения для отдыха	38 дБА

Сравнив допустимый и ожидаемый уровни звука в помещении, видно, что нет необходимости проводить мероприятия по борьбе с шумом.

8.4 Освещение

В зависимости от необходимости, производственное освещение в кабинете может быть как естественным, создаваемым непосредственно солнцем и диффузным (рассеянным) светом, так и искусственным, осуществляемым электрическими лампами. Естественное освещение характеризуется тем, что создаваемое освещение изменяется в очень широких пределах, в зависимости от времени года, дня и метеорологических факторов. При выборе норм естественного освещения учитывается разряд выполняемых работ, система освещения, коэффициент солнечности, коэффициент светового климата. По СНиП П-4-79 для 3 разряда зрительной работы (контраст большой, фон светлый) освещенность при боковом освещении от окон равна 2.0%.

Для искусственного освещения учитывается разряд выполняемых зрительных работ, подразряд работы, вид ламп, система освещения. В данном

случае подразряд работы будет «В» при котором нормируемое значение для комбинированного освещения 2500 лк и 750 лк для общего освещения.

В кабинете имеются светильники с люминесцентными лампами расположены параллельно стене. Коэффициент естественного освещения E_H должен быть не ниже 1.5%.

Мероприятия, за счет которых выполняются требования норм СНиП П-4-79:

- проверка, не реже одного раза в год, соответствия освещенности на рабочей поверхности нормам искусственного освещения;
- очистка светильников не реже одного раза в три месяца;
- протирка окон (стекол) не реже двух раз в год.

Освещение рабочих мест в помещении для работы администратора должно планироваться так, чтобы свет не падал прямо в глаза, отсутствовали мерцающие тени, блики, «мигание» люминесцентных ламп, яркость должна быть распределена равномерно.

8.5 Электробезопасность

В кабинет электроэнергия поступает для питания персональных компьютеров и электрического освещения. Питание осуществляется от трехфазной сети переменного тока напряжением 380/220 В (+10..-15%) частотой 50 Гц (+1 Гц).

Поскольку помещение сухое (относительная влажность не более 75%), температура не превышает 30 °С, то, согласно ПУЭ («Правилам устройства электроустановок»), оно не относится к категории помещений повышенной опасности. Однако возможна потенциальная опасность поражения людей электрическим током. Источниками и причинами опасности являются:

- открытые токопроводящие части оборудования, кабельной проводки;
- неисправность электрооборудования, электрических розеток;
- короткое замыкание в результате повреждения изоляции.

Для предотвращения поражения электрическим током потребителей электроэнергии в кабинете необходимо предусмотреть следующие технические мероприятия:

- все токопроводящие части оборудования и кабельной проводки должны быть защищены ограждающими кожухами;
- все металлические конструкции, которые могут оказаться под напряжением в результате короткого замыкания, должны быть заземлены, защищены и выполнено защитное отключение.

В качестве заземляющих проводников должны быть использованы элементы металлических конструкций, металлическое обрамление кабельных каналов. Здание должно быть оборудовано комплексом мер, предотвращающих попадание энергии молнии в электрическую сеть, а также поражение людей, для чего на здание устанавливаются громоотводы.

Кроме технических, необходимо проведение организационных мероприятий:

- к работе с электроустановками допускаются только лица, прошедшие инструктаж и проверку знаний правил техники безопасности в соответствии с ГОСТ 12.0.004-79, ПТЭ и ПТБ;
- периодически осуществляется контроль сопротивления электрической изоляции токоведущих частей (в соответствии с требованиями ПУЭ, оно не должно быть ниже 0.5 мм по отношению к корпусу ЭВМ).

8.6 Пожаробезопасность

Пожар в помещении может возникнуть при взаимодействии горючих веществ, окислителя (условия пожара) и источников воспламенения (причина пожара). Горючие вещества в кабинете: деревянные столы, двери, полы (паркет), покрытия стен, изоляция соединительных кабелей, жидкости для протирки узлов компьютера и другие.

Возможные источники и причины возникновения пожара:

- эксплуатация неиспользованного оборудования;
- неправильное применение электронагревательных приборов;

– короткое замыкание;
– неисправность проводки;
– нарушение правил пожарной безопасности.

Для отвода тепла от персонального компьютера необходимы работающие вентиляторы, помещение проветривается, поэтому кислород, как окислитель процессов горения, имеется в достаточном количестве. Исходя из этого, помещение кабинета, согласно нормам СНиП-П-90-81, по степени пожаробезопасности следует отнести к категории Д (помещения, в которых в обращении находятся негорючие вещества и материалы в холодном состоянии).

В качестве средств тушения пожара применяются углекислотные огнетушители, используемые для тушения электроустановок, находящихся под напряжением.

8.7 Эргономика и техническая эстетика

Эффективность работы администратора (программиста) во многом зависит от организации рабочих мест. Рабочее положение администратора — сидячее. Стул по возможности должен быть регулируемым по высоте, поскольку клавиатура и дисплей компьютеров должны находиться в зоне наилучшего обзора. Для сохранения работоспособности имеет большое значение выбор основной рабочей позы.

Техническая эстетика позволяет снижать нервное утомление и вредные воздействия на функции организма в процессе труда. Огромное значение в эстетическом оформлении производства имеет цвет. Окраска, форма, внешний вид производственного помещения и оборудования улучшают условия освещения, а также психологическое состояние человека. Стены имеют светло-зеленый цвет, не вызывающий раздражения, потолок — белый цвет, что обеспечивает максимальное отражение света.

Рассматриваемое помещение соответствует требованиям ГОСТ 12.2.032-78.

8.8 Режим труда и отдыха

Работа администратора относится к категории работ связанных с опасными и вредными условиями труда. В процессе труда на администратора оказывают действие следующие опасные и вредные производственные факторы. Физические:

- повышенный уровень статического электричества;
- повышенные уровни запыленности воздуха рабочей зоны;
- повышенный уровень шума;
- повышенный или пониженный уровень освещенности;
- неравномерность распределения яркости в поле зрения;
- повышенная яркость светового изображения;
- повышенное значение напряжения в электрической цепи, замыкание которой может произойти через тело человека.

Химические:

- напряжение зрения;
- напряжение внимания;
- интеллектуальные нагрузки;
- монотонность труда;
- большой объем информации, обрабатываемой в единицу времени;
- нерациональная организация рабочего места.

Биологическим фактором является повышенное содержание в воздухе рабочей зоны микроорганизмов.

Рациональный режим труда и отдыха — это правильное чередование работы и перерывов в ней в течение смены, суток, недели, года, устанавливаемое с целью обеспечения высокой производительности труда и сохранения здоровья работающих. Основным перерывом является перерыв на обед. Обеденный перерыв при 8-часовой рабочей смене устанавливается продолжительностью не менее 30 мин через 4 часа после начала работы.

Режим отдыха складывается из нескольких компонентов:

- времени на гигиенические процедуры и личные надобности (2-3) от сменного времени независимо от вида труда;
- времени регламентированных перерывов на отдых (входят в состав рабочего времени), определяемого по показателю условий труда или по интегральному показателю снижения работоспособности;
- времени микропауз, а также времени обеденного перерыва (нерабочего времени), остающегося от приема пищи.

Режим труда и отдыха должен быть построен в соответствии с особенностями трудовой деятельности пользователей персонального компьютера и характером функциональных изменений со стороны различных систем организма работников.

ЗАКЛЮЧЕНИЕ

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Каталог программных продуктов семейства Oracle [Электронный ресурс]. – Электрон. текстовые данные (11126315 bytes) – Режим доступа: http://oracle.ocs.ru/files/catalog_Oracle_Database_12C.pdf

2. Tier: уровни надежности ЦОД и что из этого следует [Электронный ресурс]. – Электрон. текстовые данные (38916 bytes) – Режим доступа: http://www.aboutdc.ru/page/390.php

3. Виртуальный Linux. Обзор методов виртуализации, архитектур и реализаций [Электронный ресурс]. – Электрон. текстовые данные (109051 bytes) – Режим доступа: http://www.ibm.com/developerworks/ru/library/l-linuxvirt/index.html

4. Руководство по созданию и управлению контейнерами на базе OpenVZ [Электронный ресурс]. – Электрон. текстовые данные (55237 bytes) – Режим доступа: https://github.com/Amet13/openvz-tutorial/blob/master/main.pdf

5. Эмуляция систем с помощью QEMU [Электронный ресурс]. – Электрон. текстовые данные (84008 bytes) – Режим доступа: http://www.ibm.com/developerworks/ru/library/l-qemu/

6. Гипервизоры, виртуализация и облако: Анализ гипервизора KVM [Электронный ресурс]. – Электрон. текстовые данные (79196 bytes) – Режим доступа: http://www.ibm.com/developerworks/ru/library/cl-hypervisorcompare-kvm/

7. Hyper-V drivers in mainline kernel - what’s next [Электронный ресурс]. – Электрон. текстовые данные (146405 bytes) – Режим доступа: https://social.technet.microsoft.com/Forums/en-US/7007beff-db11-4261-a9f2-0f4461a9cc92/hyperv-drivers-in-mainline-kernel-whats-next

8. Руководство пользователя Xen v3.0 [Электронный ресурс]. – Электрон. текстовые данные (272945 bytes) – Режим доступа: http://xgu.ru/xen/manual/

9. Контейнеризация на Linux в деталях – LXC и OpenVZ Часть 2 [Электронный ресурс]. – Электрон. текстовые данные (186899 bytes) – Режим доступа: http://habrahabr.ru/company/FastVPS/blog/209084/

10. Виртуализация на уровне ОС: теория и практика LXC [Электронный ресурс]. – Электрон. текстовые данные (118259 bytes) – Режим доступа: http://creativeyp.com/568-virtualizaciya-na-urovne-os-teoriya-i-praktika-lxc.html

11. Основы виртуализации. Инфраструктура [Электронный ресурс]. – Электрон. текстовые данные (32477 bytes) – Режим доступа: http://www.bw-it.ru/virtualization_virtual_infrastructure.php

12. Методические указания «Процедура системного анализа при проектировании программных систем» для студентов-дипломников дневной и заочной формы обучения специальности 7.091501 / Сост.: Сергеев Г.Г., Скатков А.В., Мащенко Е.Н. – Севастополь: Изд-во СевНТУ, 2005. – 32с.

ПРИЛОЖЕНИЕ А — ИСХОДНЫЙ КОД ПРОГРАММЫ DDOS-DEFLATE

```
ddos-deflate
============
Shell script blocking DDoS attacks. Fork of [(D)DoS Deflate](http
   ://deflate.medialayer.com/).

It works on Debian 7 and CentOS 7 (please tell me if you've
   tested script on other distros).

Installation
------------
'''bash
su -
cd /tmp
wget https://raw.githubusercontent.com/Amet13/ddos-deflate/
   master/install.sh
chmod +x install.sh
./install.sh
'''
Add your contact e-mail:
'''bash
vim /usr/local/ddos/ddos.conf
EMAIL_TO="mail@example.com"
'''
Add your ignore ip's to ignorelist:
'''bash
vim /usr/local/ddos/ignore.ip.list
127.0.0.1
1.1.1.1
2.2.2.2
'''
Add cronjob:
'''bash
crontab -e
# run script every minute
* * * * * /usr/local/ddos/ddos.sh >/dev/null 2>&1
```

```
Check:
```bash
/usr/local/ddos/ddos.sh
   724 127.0.0.1
   214 2.2.2.2
    59 3.3.3.3
...
```

Testing

Run ab from another computer:
```bash
user@192.168.0.100 ~ $ ab -n 200000 -c 100 http://server-ip/
```
Check new IPTables rules on server:
```bash
iptables -L INPUT
Chain INPUT (policy ACCEPT)
target  prot opt source        destination
DROP    all -- 192.168.0.100  anywhere
```

Alternative usage

If you want block only HTTP attackers, you can replace:
```bash
netstat -ntu | awk '{print $5}' | cut -d: -f1 | sort | uniq -c |
    sort -nr > $BAD_IP_LIST
```
to
```bash
netstat -ntu | grep ":80" | awk '{print $5}' | cut -d: -f1 |
   sort | uniq -c | sort -nr > $BAD_IP_LIST
```
in `/usr/local/ddos/ddos.sh`
```

Updating
————————
‘‘‘bash
cd /usr/local/ddos/
wget https://raw.githubusercontent.com/Amet13/ddos—deflate/
   master/ddos.sh —O ddos.sh
‘‘‘

Uninstallation
——————————————
‘‘‘bash
su —
cd /tmp
wget https://raw.githubusercontent.com/Amet13/ddos—deflate/
   master/uninstall.sh
chmod +x uninstall.sh
./uninstall.sh
crontab —e
#*/1 * * * * /usr/local/ddos/ddos.sh >/dev/null 2>&1
‘‘‘

Original author
———————————————
zaf@vsnl.com

Paths of the script and other files
PROGDIR=”/usr/local/ddos”
PROG=”/usr/local/ddos/ddos.sh”
IGNORE_IP_LIST=”/usr/local/ddos/ignore.ip.list”
APF=”/etc/apf/apf”
IPT=”/sbin/iptables”

How many connections define a bad IP? Indicate that below.
NO_OF_CONNECTIONS=150

APF_BAN=1 (Make sure your APF version is atleast 0.96)
APF_BAN=0 (Uses iptables for banning ips instead of APF)

```
APF_BAN=0

##### KILL=0 (Bad IPs are'nt banned, good for interactive
   execution of script)
##### KILL=1 (Recommended setting)
KILL=1

##### An email is sent to the following address when an IP is
   banned.
##### Blank would suppress sending of mails
EMAIL_TO="root"

##### Number of seconds the banned ip should remain in blacklist.
BAN_PERIOD=600

#!/bin/bash
###################################################################
# DDoS-Deflate version 0.6 Author: Zaf <zaf@vsnl.com>
# It's fork of DDoS-Deflate by Amet13 <admin@amet13.name>
# https://github.com/Amet13/ddos-deflate
###################################################################
# This program is distributed under the "Artistic License"
# Agreement
#
# The LICENSE file is located in the same directory as
# this program. Please read the LICENSE file before you
# make copies or distribute this program
###################################################################
load_conf()
{
  CONF="/usr/local/ddos/ddos.conf"
  if [ -f "$CONF" ] && [ "$CONF" != "" ]; then
    source $CONF
  else
    head
    echo "\$CONF not found."
    exit 1
  fi
```

```
}

head()
{
  echo "DDoS-Deflate version 0.6"
  echo "Copyright (C) 2005, Zaf <zaf@vsnl.com>"
  echo "It's fork of DDoS-Deflate by Amet13 <admin@amet13.name>"
  echo
}

showhelp()
{
  head
  echo 'Usage: ddos.sh [OPTIONS] [N]'
  echo 'N : number of tcp/udp connections (default 150)'
  echo 'OPTIONS:'
  echo '-h | --help: Show this help screen'
  echo '-k | --kill: Block the offending ip making more than N
     connections'
}

unbanip()
{
  UNBAN_SCRIPT=`mktemp /tmp/unban.XXXXXXXX`
  TMP_FILE=`mktemp /tmp/unban.XXXXXXXX`
  UNBAN_IP_LIST=`mktemp /tmp/unban.XXXXXXXX`
  echo '#!/bin/sh' > $UNBAN_SCRIPT
  echo "sleep $BAN_PERIOD" >> $UNBAN_SCRIPT
  if [ $APF_BAN -eq 1 ]; then
    while read line; do
      echo "$APF -u $line" >> $UNBAN_SCRIPT
      echo $line >> $UNBAN_IP_LIST
    done < $BANNED_IP_LIST
  else
    while read line; do
      echo "$IPT -D INPUT -s $line -j DROP" >> $UNBAN_SCRIPT
      echo $line >> $UNBAN_IP_LIST
```

```
   done < $BANNED_IP_LIST
  fi
  echo "grep -v --file=$UNBAN_IP_LIST $IGNORE_IP_LIST >
     $TMP_FILE" >> $UNBAN_SCRIPT
  echo "mv $TMP_FILE $IGNORE_IP_LIST" >> $UNBAN_SCRIPT
  echo "rm -f $UNBAN_SCRIPT" >> $UNBAN_SCRIPT
  echo "rm -f $UNBAN_IP_LIST" >> $UNBAN_SCRIPT
  echo "rm -f $TMP_FILE" >> $UNBAN_SCRIPT
  . $UNBAN_SCRIPT &
}

load_conf
while [ $1 ]; do
  case $1 in
    '-h' | '--help' | '?' )
      showhelp
      exit
      ;;
    '--kill' | '-k' )
      KILL=1
      ;;
     *[0-9]* )
      NO_OF_CONNECTIONS=$1
      ;;
    * )
      showhelp
      exit
      ;;
  esac
  shift
done

TMP_PREFIX='/tmp/ddos'
TMP_FILE="mktemp $TMP_PREFIX.XXXXXXXX"
BANNED_IP_MAIL=`$TMP_FILE`
BANNED_IP_LIST=`$TMP_FILE`
```

```
echo ”Banned the following ip addresses on ‘date‘” >
   $BANNED_IP_MAIL
echo ”From ‘hostname —f‘ (‘hostname ——ip—address‘)” >>
   $BANNED_IP_MAIL
echo >> $BANNED_IP_MAIL
BAD_IP_LIST=‘$TMP_FILE‘
netstat —ntu | awk ’{print $5}’ | cut —d: —f1 | sort | uniq —c |
    sort —nr > $BAD_IP_LIST
cat $BAD_IP_LIST
if [ $KILL —eq 1 ]; then
  IP_BAN_NOW=0
  while read line; do
    CURR_LINE_CONN=$(echo $line | cut —d” ” —f1)
    CURR_LINE_IP=$(echo $line | cut —d” ” —f2)
    if [ $CURR_LINE_CONN —lt $NO_OF_CONNECTIONS ]; then
      break
    fi
    IGNORE_BAN=‘grep —c $CURR_LINE_IP $IGNORE_IP_LIST‘
    if [ $IGNORE_BAN —ge 1 ]; then
      continue
    fi
    IP_BAN_NOW=1
    echo ”$CURR_LINE_IP with $CURR_LINE_CONN connections” >>
       $BANNED_IP_MAIL
    echo $CURR_LINE_IP >> $BANNED_IP_LIST
    echo $CURR_LINE_IP >> $IGNORE_IP_LIST
    if [ $APF_BAN —eq 1 ]; then
      $APF —d $CURR_LINE_IP
    else
      $IPT —I INPUT —s $CURR_LINE_IP —j DROP
    fi
  done < $BAD_IP_LIST
  if [ $IP_BAN_NOW —eq 1 ]; then
    dt=‘date‘
    if [ $EMAIL_TO != ”” ]; then
      cat $BANNED_IP_MAIL | mail —s ”IP addresses banned on $dt”
         $EMAIL_TO
```

```
   fi
   unbanip
  fi
fi
rm -f $TMP_PREFIX.*

127.0.0.1

#!/bin/sh
if [ -d '/usr/local/ddos' ]; then
  echo; echo; echo "Please uninstall the previous version first"
  exit 0
else
  mkdir /usr/local/ddos
fi
clear
echo; echo 'Installing DOS-Deflate 0.6'; echo
echo; echo -n 'Downloading source files...' ; echo
wget -q -O /usr/local/ddos/ddos.conf https://raw.
   githubusercontent.com/Amet13/ddos-deflate/master/ddos.conf
echo -n '30% '
wget -q -O /usr/local/ddos/LICENSE https://raw.githubusercontent
   .com/Amet13/ddos-deflate/master/LICENSE
echo -n '50% '
wget -q -O /usr/local/ddos/ignore.ip.list https://raw.
   githubusercontent.com/Amet13/ddos-deflate/master/ignore.ip.
   list
echo -n '100%'
wget -q -O /usr/local/ddos/ddos.sh https://raw.githubusercontent
   .com/Amet13/ddos-deflate/master/ddos.sh
chmod 0755 /usr/local/ddos/ddos.sh
echo '... done'
echo; echo 'Installation has completed'
echo 'Config file is at /usr/local/ddos/ddos.conf'
echo 'Please send in your comments and/or suggestions to zaf@vsnl
   .com'
echo 'About new changes: admin@amet13.name'
echo 'Do not forget create a cron job!'
```

```
cat /usr/local/ddos/LICENSE | less

#!/bin/sh
echo; echo "Uninstalling DOS-Deflate"
echo; echo; echo -n "Deleting script files..."
if [ -e '/usr/local/sbin/ddos' ]; then
  rm -f /usr/local/sbin/ddos
  echo -n "..."
fi
if [ -d '/usr/local/ddos' ]; then
  rm -rf /usr/local/ddos
  echo -n "..."
fi
echo "done"
echo; echo "Uninstall Complete"; echo
```