

И в блок импортов строчку:

```
from sqlalchemy import orm
```

Это позволит нам легко получать для пользователя все его новости. Обратите внимание: значение параметра `back_populates` должно указывать не на таблицу, а на атрибут класса `orm.relationship`.

Чтобы подключить модель `News` к нашей базе данных, в файл `_all_models.py` надо добавить строку:

```
from . import news
```

Удалим базу данных и запустим наше приложение заново, чтобы все таблицы пересоздались. Вот какие две таблицы у нас получились в базе данных:

```
user = db_sess.query(User).filter(User.id == 1).first()
news = News(title="Личная запись", content="Эта запись личная",
            is_private=True)
user.news.append(news)
db_sess.commit()
```

Так же легко мы можем обойти все новости конкретного пользователя:

Содержание

1 Введение	4
2 Постановка задачи	5
3 Обзор существующих решений	6
3.1 Цель и критерии обзора	6
3.2 Обзор существующих протоколов	6
3.2.1 SMB	7
3.2.2 NBD	8
3.2.3 NFSv4	9
3.3 Существующие форматы дисков для хранения образов ВМ	10
3.4 Существующие тесты производительности	12
4 Исследование и построение решения задачи	13
4.1 Формальная постановка задачи	13
5 Описание практической части	14
5.1 Программа исследования	14
5.2 Методика исследования	14
5.2.1 Схема стенда для исследования	14
5.2.2 Алгоритм тестирования	15
6 Результаты экспериментов	17
7 Заключение	24
Список литературы	25

1 Введение

Практическая деятельность обработки информации на персональном компьютере обычно требует предоставления возможности одновременной работы с внутренними, распределенными информационными ресурсами и доступом в сеть Интернет.

Необходимость решения этой задачи обуславливает использование в ряде случаев соответствующих организационно-технических методов защиты информации, в основе которых лежит идея изоляции доменов. Программное средство общего назначения с встроенными средствами защиты «Абонентский облачный терминал» (АОТ)[1] решает задачу путем совмещения на одной аппаратной единице, персональном компьютере, произвольного количества разнородных, изолированных, не зависимых друг от друга рабочих мест. Каждое место работы функционирует под контролем операционной системы, в своей программной и сетевой среде, что позволяет полностью изолировать его от сети.

Рабочая инфраструктура централизованной [облачной] вычислительной структуры (ЦВС)[1] предприятия может содержать репозитарий контейнеров с виртуальными машинами (ВМ), реализующих рабочие места сотрудников и прошедших принятую на предприятии процедуру внедрения в постоянную эксплуатацию. Репозитарий эталонных ВМ и обновлений содержит эталонные ВМ, серверы с обновлениями ПО. Задачи, решаемые этим компонентом ЦВС: формирование проверенных, согласованных, безопасных эталонных рабочих мест в виде контейнеров с типовыми ВМ. Образы ВМ копируются на локальные диски АОТ. Однако одним из существенных недостатков такого решения является то, что для размещения на персональном компьютере образа виртуальной машины с установленной на ней полноценной ОС, например, MS Windows 10, необходимо скопировать и передать по сети около 40 Gb, что выливается в длительную процедуру обновления и приводит к «коллапсу» локальной сети. Решение поставленной задачи должно снизить время загрузки образа ВМ и нагрузку на сеть.

2 Постановка задачи

Цель работы

Исследовать время запуска ВМ в режиме расширенного фильтра записи на АОТ в зависимости от способа хранения и протокола доступа к данным виртуального диска.

План работы

1. Провести анализ существующих протоколов для доступа к данным удаленного сетевого хранилища.
2. Разработка программы и методики экспериментального исследования.
3. Провести экспериментальное исследование выбранных протоколов и сделать вывод об их эффективности.
4. Предложить способ реализации репозитария в зависимости от характеристик сети и эксплуатируемых рабочих мест

3 Обзор существующих решений

3.1 Цель и критерии обзора

Существует множество различных протоколов доступа к виртуальному диску. Целью обзора является анализ их применимости для решения задачи запуска образа ВМ на АОТ в режиме оверлей.

3.2 Обзор существующих протоколов

Критерии:

1. Масштабируемость — Поддерживает ли протокол горизонтальное масштабирование.
2. Оптимизационные механизмы — Наличие у протокола механизмов, снижающих нагрузку на сеть, то есть возможность уменьшить количество сообщений между клиентом и сервером.
3. Чувствительность — Снижается ли производительность протокола при изменении задержки отправки пакетов в сети.
4. Устойчивость — Реакция протокола на разрыв соединения между хранилищем ВМ и АОТ.

3.2.1 SMB

- **Описание протокола и область применения**

SMB (Server Message Block)[2] — сетевой протокол обеспечивающий общий доступ к файлам и устройствам в компьютерной сети, будь то внешний жесткий диск или принтер. Основанный на клиент-серверной технологии, которая предоставляет приложениям простой способ чтения и записи файлов, является довольно популярным инструментом для доступа к удаленным данным.

Для соединения клиента и сервера протокол использует протоколы TCP/IP[2], NetBEUI[2] или IPX/SPX[2]. После успешного соединения с сервером клиент и сервер могут обмениваться специальными командами (SMB командами).[2]

В протоколе реализованы базовые алгоритмы шифрования, а также разные уровни прав доступа для работы с файлами. Существует два уровня защиты: user-level и share-level. Аутентификация на user-level предполагает у клиента наличие логина и пароля для доступа к некоторому "уровню" ресурсов. Share-level, в свою очередь, требует от клиента пароль от каждого конкретного ресурса при попытке взаимодействия с ним.[2]

- **Используемые метрики** - Пропускная способность сети.

- **Масштабируемость**

Протокол был разработан с упором на масштабируемость, он может работать с большим количеством клиентов и высоким уровнем сетевого трафика. Однако ограничения проявляются при одновременном доступе множества клиентов к одному файлу. В таком случае SMB может потребоваться блокировка файла для предотвращение конфликта. Это может привести к увеличению задержки и снижению пропускной способности.

- **Оптимизационные механизмы**

В протоколе реализовано множество оптимизационных механизмов, позволяющих уменьшить количество пакетов в сети и снизить общую нагрузку на нее. Например, оппортунистическая блокировка, позволяющая кэшировать некоторое количество пакетов и объединять их в один. Также лучшей производительности удает-

ся достичь благодаря переноса операций передачи данных на сетевую карту, тем самым разгружая процессор.[2]

- **Устойчивость**

При разрыве соединения между АОТ и хранилищем прокотокол поддерживает переподключение и позволит клиенту восстановить доступ к открытому файлу.[2]

- **Чувствительность**

Довольно сильно реагирует на изменение пропускной способности сети.

3.2.2 NBD

- **Описание протокола и область применения**

NBD (Network Block Device)[3]- это сетевой протокол, который позволяет клиенту получать доступ к удаленным блочным устройствам по сети. Обычно используется в средах виртуализации, где ВМ может получить доступ к образам, как если бы они были локальными. Работает по модели клиент-сервер, где клиент и сервер взаимодействуют с помощью TCP/IP.[3]

Сетевое блочное устройство представлено с помощью трех компонентов: серверной части, клиентской части и сети между ними. На клиентской машине, на которой находится узел устройства, драйвер ядра управляет устройством. Всякий раз, когда программа пытается получить доступ к устройству, драйвер ядра пересыпает запрос на серверную машину на данные в котором физически находятся. На сервере запросы от клиента обрабатываются программой пользовательского пространства.[3]

Работу протокола составляют 2 части: квитирование и передача данных. Квтирование происходит при соединении клиента с сервером. На этой стадии они устанавливают правила общения друг с другом.

- **Используемые метрики** - Пропускная способность сети.

- **Масштабируемость**

Протокол хорошо масштабируем. Причем масштабирование возможно горизонтальным путем, то есть с помощью добавления дополнительных серверов для распределения нагрузки. Также можно использовать балансировщик нагрузки для равномерного распределения клиентов между серверами. Также реализация протокола позволяет кэшировать данные, к которым часто обращаются, на стороне сервера.

- **Оптимизационные механизмы**

Нагрузка на сеть, вызываемая использованием протокола, во многом зависит от установленных настроек безопасности. У протокола есть проблемы с устойчивостью, поэтому при большом количестве соединений, разрывы могут значительно нагружать сеть. Также некоторая нагрузка создается из-за отсутствия в протоколе оптимизирующих механизмов, например, пропуск участков файлов с нулями.[3]

- **Устойчивость**

Протокол не имеет встроенных механизмов отказоустойчивости. Общение между клиентом и сервером может закончиться только по желанию клиента, то есть сервер не может самостоятельно оборвать соединение. Если разрыв соединения произошел во время передачи данных, то при повторном подключении передача начнется с самого начала. В случае, если отключается nbd устройство, то требуется полная перезагрузка сервера.

- **Чувствительность**

Благодаря кэширующим механизмам не сильно реагирует на изменение пропускной способности сети.

3.2.3 NFSv4

- **Описание протокола и область применения**

NFSv4 (Network File System version 4)[4] — протокол, созданный для работы с распределенной файловой системой. Позволяет клиенту получать доступ к удаленной

файловой системе (экспортированным каталогам) по сети. Основан на технологии клиент-сервер. Использует TCP/IP.

- **Используемые метрики** - Пропускная способность.

- **Масштабируемость**

Протокол сконструирован с упором на быструю передачу данных и хорошую масштабируемость при параллельной работе нескольких клиентов с одним файлом. Когда несколько клиентов читают данные пропускная способность делится между ними неравномерно.

Протокол поддерживает горизонтальное расширение, то есть увеличение количества серверов с данными для равномерного распределения нагрузки.

Также он содержит в себе расширение pNFS[5], написанное специально для параллельной обработки множества клиентов.

- **Оптимизационные механизмы**

Нагрузка, которую протокол создает на сеть, зависит от его настроек, а также от количества клиентов.

- **Устойчивость**

Возможно восстановление сессии клиента после перезагрузки сервера, при условии, что она была активна перед ней.

- **Чувствительность**

Довольно сильно реагирует на изменение пропускной способности сети.

3.3 Существующие форматы дисков для хранения образов ВМ

Критерии:

1. Размер - размер, который занимает образ
2. Быстродействие - скорость чтения при хранении диска в этом формате

- **Raw**

Широко поддерживаемый формат виртуального диска, который хранит данные диска без дополнительных метаданных или сжатия. Необработанные диски хранят данные в виде обычного файла, что делает их простыми в использовании и совместимыми с широким спектром платформ виртуализации.[6]

- **Размер**

Формат плохо поддерживает сжатие данных, из-за чего файлы занимают много дискового пространства.

- **Быстродействие**

Формат позволяет читать данные достаточно быстро.

- **Qcow2**

Qcow2 (QEMU Copy On Write version 2) - это формат виртуального диска, используемый платформой виртуализации QEMU. Он разработан для обеспечения эффективного использования хранилища и производительности виртуальных машин.

- **Размер**

За счет сжатия данных и кэширования формат позволяет эффективно использовать дисковое пространство.[7]

- **Быстродействие**

Кэширование и сжатие данных сильно замедляет чтение данных диска.[7]

По итогам обзора был выбран формат Raw, так как он более эффективный и позволяет быстро читать данные без дополнительной обработки.[6] Также он лучше совместим с разными системами.

3.4 Существующие тесты производительности

- iozone

Тест Iozone применяется для генерации и измерения различных файловых операций, таких как чтение и запись в различных режимах.

- bonnie++

Bonnie++ тест применяется для создания, чтения и удаления множества небольших файлов, и изменения их метаданных. В результате своей работы он даёт пользователю производительность его файловой системы в виде измерения характеристик чтения-записи.

- diskspd

Diskspd - приложение Microsoft[8]. Позволяет провести тест производительности файловой системы при помощи генерации различных файловых операций, таких как чтение и запись в различных режимах.

4 Исследование и построение решения задачи

4.1 Формальная постановка задачи

Дано:

- w - пропускная способность сети, Мб/сек
- τ_0 - задержка отправления пакетов в сети, мс
- Ncl – количество клиентов
- Mos – тип ОС (Windows, Linux)
- Cs – критерий старта
- P – протокол доступа к данным виртуального диска, $P \in \text{SMB, NBD, NFS}$

Оборудование:

- SSD – диск SanDisk SDSSDHP128G, скорость чтения 530 МБ/сек
- CPU – процессор Intel Core i5-4570 CPU 3.20GHz, L3: 6МБ кэша
- NIC – сетевая карта Intel Ethernet Connection I217-V, 256 КБ кэша

Требуется:

- Найти зависимость времени запуска $T=f_t(w, \tau_0, Ncl, Mos, Cs, P; \text{HDD, CPU, NIC})$
- Найти максимальную скорость чтения данных виртуального диска $Rx=f_{Rx}(w, \tau_0, Ncl, Mos, Cs, P; \text{HDD, CPU, NIC})$

5 Описание практической части

5.1 Программа исследования

- Замерить время запуска ВМ, критерием которого является загрузка сетевого интерфейса и графической оболочки, при использовании протоколов:
 - SMB3
 - NFSv4
 - NBD
- Замерить скорость чтения данных виртуального диска при условии доступа к нему через протоколы:
 - SMB3
 - NFSv4
 - NBD

5.2 Методика исследования

5.2.1 Схема стенда для исследования

Стенд состоит из сервера и АРМ сотрудника, которые соединены через роутер с помощью Gigabit Ethernet.

Сервер сконфигурирован на базе ОС Debian 8.11 Jessie, поверх которой установлен специальное ПО для администрирования АРМ сотрудника. Также на серверы установлены:

- SSD диск SanDisk SDSSDHP128G, скорость чтения 530 МБ/сек.
- Процессор Intel Core i5-4570 CPU 3.20GHz, L3: 6МБ кэша.
- Сетевая карта Intel Ethernet Connection I217-V, 256 КБ кэша.

На диск предварительно установлен образ ВМ Windows 7 sp1 x32, размером 15ГБ, в которой предустановлено приложение Diskspd и скрипты для его запуска. Также на

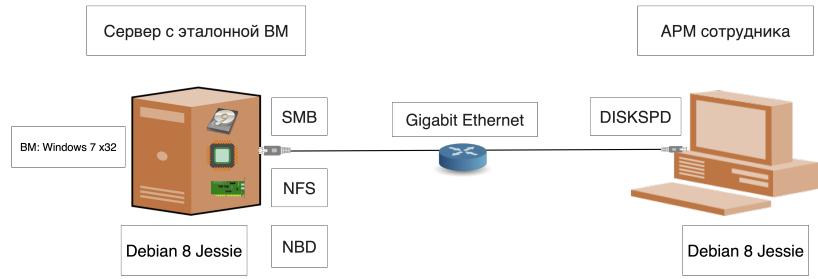


Рис. 1: Схема стенда для исследования

сервере настроены протоколы и поставлены соответствующие настройки доступа к файлу с ВМ.

АРМ сотрудника сконфигурирован на базе ОС Debian 8.11 Jessie, поверх которой установлен специальное ПО для запуска ВМ. Также на нем настроена клиентская часть выбранных протоколов.

5.2.2 Алгоритм тестирования

- На сервере и клиенте настраивается выбранный протокол:
 - SMB3
 - NFSv4
 - NBD
- На сервере устанавливается утилита Linux Traffic Control;
- Клиент подключается к серверу и производит автоматизированный запуск ВМ;

- После запуска ВМ, с помощью планировщика задач, запускаются тесты производительности;
- Тесты проходят в режимах:
 - последовательного чтения данных блоками по 1МБ;
 - случайного чтения данных по случайно выбираемым смещениям блоками по 4КБ;
 - чтения данных по случайно выбираемым смещениям блоками по 4КБ с одновременным выполнением 32 операций чтения;

6 Результаты экспериментов

По результатам исследования для каждого типа тестов были составлены таблицы, позволяющие понять какой протокол следует использовать при разных параметрах сети. Основными показателями, которые учитывались при анализе результатов тестирования было время загрузки ВМ и скорость чтения данных с диска при разных значениях задержки и скорости в сети. При анализе использовались усредненные значения.

- Задержка - задержка в сети (мс)
- Сеть - скорость в сети
- Протокол - тестируемые протокол
- Время загрузки - время загрузки ВМ
- Скорость чтения - скорость чтения данных с диска

Задержка(мс)	Сеть(Мб/с)	Протокол	Время Загрузки(с)	Скорость чтения(МБ/с)
0	1000	nbd	12.73	21.33
0	200	smb	19.44	2.22
0	150	smb	22.68	2.35
5	1000	nbd	25.2	20.83
5	200	nbd	28.3	21.15
5	150	nbd	33.19	22.34
0	100	smb	34.2	2.35
10	1000	nfs	40.09	23.64
5	100	nbd	40.63	25.47
10	200	nbd	40.66	21.39
10	150	nbd	43.71	24.77
15	1000	nfs	48.31	11.64
10	100	nbd	49.04	24.33
15	200	nbd	53.97	21.94
15	150	nbd	55.35	24.79
15	100	nbd	58.31	25.11

Таблица 1: Тестирование в режиме случайного чтения

Задержка(мс)	Сеть(Мб/с)	Протокол	Время Загрузки(с)	Скорость чтения(МБ/с)
0	1000	nbd	12.73	110.38
0	200	smb	19.44	21.88
0	150	smb	23.48	16.62
5	1000	nbd	25.89	80.88
5	200	nbd	27.4	21.88
5	150	nbd	32.19	16.98
0	100	smb	37.84	11.22
10	1000	nfs	40.59	80.5
5	100	nbd	41.6	11.25
10	200	nbd	42.55	21.0
10	150	nbd	43.71	16.38
15	1000	nfs	48.31	70.5
10	100	nbd	49.34	10.88
15	200	nbd	52.62	18.98
15	150	nbd	56.44	15.34
15	100	nbd	59.61	11.25

Таблица 2: Тестирование в режиме последовательного чтения

Задержка(мс)	Сеть(Мб/с)	Протокол	Время Загрузки(с)	Скорость чтения(МБ/с)
0	1000	nbd	13.73	31.0
0	200	smb	19.44	22.4
0	150	smb	23.48	16.03
5	1000	nbd	25.2	36.34
5	200	nbd	28.3	34.92
5	150	nbd	33.19	34.69
0	100	smb	34.2	10.83
10	1000	nfs	40.09	22.72
5	100	nbd	40.63	31.31
10	200	nbd	40.66	34.77
10	150	nbd	43.71	33.85
15	1000	nfs	48.31	36.06
10	100	nbd	49.04	30.73
15	200	nbd	53.97	34.17
15	150	nbd	55.35	30.86
15	100	nbd	58.31	28.58

Таблица 3: Тестирование в режиме случайного чтения с одновременным выполнением 32 операций чтения



Рис. 2: График зависимости скорости загрузки ВМ от скорости и задержки в сети



Рис. 3: График зависимости скорости чтения от скорости и задержки в сети при тестировании в режиме случайного чтения

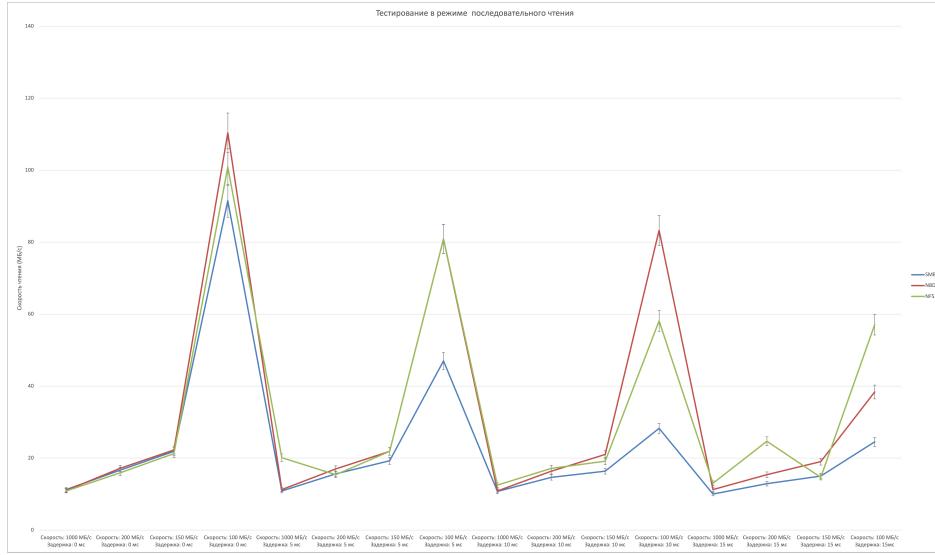


Рис. 4: График зависимости скорости чтения от скорости и задержки в сети при тестировании в режиме последовательного чтения

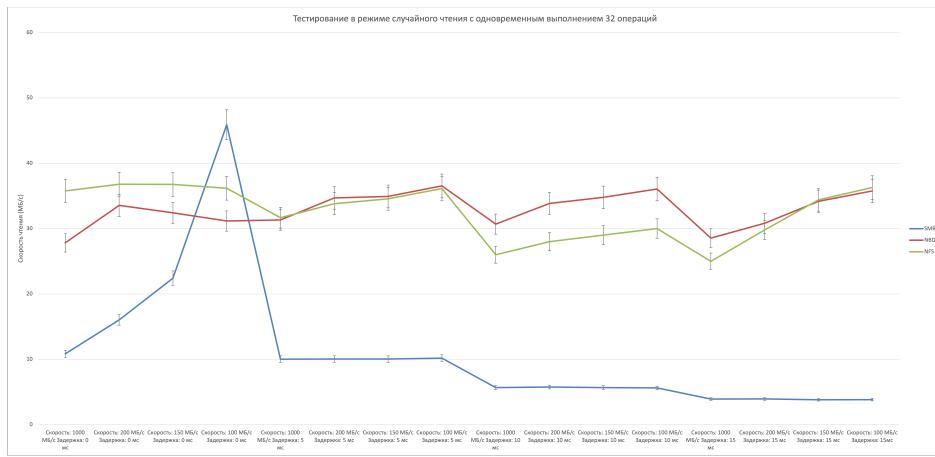


Рис. 5: График зависимости скорости чтения от скорости и задержки в сети при тестировании в режиме случайного чтения с одновременным выполнением 32 операций чтения

По результатам тестирования протокол NBD в среднем показал наилучшие результаты. Время загрузки ВМ при его использовании в большинстве случаев оказывается наименьшим. Это связано с тем, что в NBD реализованы эффективные механизмы кэширования данных, которые позволяют уменьшить количество запросов и увеличить скорость доступа к данным.

Протокол SMB показал хорошие результаты во время измерения времени загрузки ВМ при условии маленькой задержки. В других случаях время загрузки куда больше, чем у NFS и NBD, так как протокол очень чувствителен к изменению задержки в сети. Также его производительность в тестах скорости чтения оказалась очень низкой.

NFS в среднем показал результаты лучше, чем протокол SMB. Время загрузки ВМ при низкой задержке гораздо больше, чем у NBD и NFS. Однако при ее увеличении оно становится меньше, чем при использовании SMB. В тестах скорости чтения он показал результаты, очень близкие к протоколу NBD, а в некоторых условиях смог превзойти его.

7 Заключение

В ходе курсовой работы было проведено исследование времени загрузки ВМ и производительности виртуального диска в режиме расширенного фильтра записи в зависимости от протокола, используемого для доступа к данным виртуального диска, а также скорости и задержки в сети.

Исследование показало, что оптимальным выбором для работы с ВМ в таком режиме является протокол NBD. Благодаря оптимизационным механизмам, реализованным в нем, он эффективен при разных значениях скорости и задержки в сети. Однако в случае низкой задержки в сети лучше выбирать протокол SMB, так как он позволит сократить время загрузки ВМ.