УДК 004.031.4

***А. В. Данеев*[1], *В.С. Тютиков*[1]**

[1] *Иркутский государственный университет путей сообщения, г. Иркутск, Российская Федерация*

# ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ УПРАВЛЕНИЯ БЛОКИРОВКАМИ СЕТЕВОГО ТРАФИКА ВИРТУАЛЬНЫХ МАШИН КЛИЕНТОВ ХОСТИНГА

***Аннотация.*** В статье приводится возможное решение проблемы точечного управления ограничениями сетевого трафика для клиентов хостинг-провайдеров, использующих виртуальные выделенные сервера на узлах виртуализации под управлением гипервизора KVM, или с использованием технологии виртуализации OpenVZ, работающих на операционной системе Linux. Предлагается решение в виде реализации асинхронной информационной системы, централизованно хранящей информацию о текущем местоположении всех виртуальных машин (что актуально при наличии большого числа узлов виртуализации, когда в целях распределения нагрузки сервера могут мигрировать с одного узла на другой), закреплённых за ними сетевых адресах, а также позволяющей в режиме реального времени через веб-интерфейс управлять черными списками блокировки отдельных виртуальных машин или конкретных ip-адресов, с возможностью отложенной блокировки и отправки уведомлений клиентам о применении санкций за нарушение условий договора о предоставлении услуг. Описывается и обосновывается выбор архитектуры системы, а также системы управления базами данных с подробным описанием самой схемы хранения данных.

***Ключевые слова:*** Виртуализация, Linux, KVM, OpenVZ, виртуальные машины, контейнерная виртуализация, ограничение доступа, блокировка сетевого трафика, узел виртуализации, база данных, асинхронное программирование.

***A.V. Daneev*[1], *V.S. Tyutikov*[1]**

[1] *Irkutsk State Transport University, Irkutsk, Russian Federation*

# DESIGNING AN INFORMATION SYSTEM TO MANAGE NETWORK TRAFFIC BLOCKING OF VIRTUAL MACHINES OF HOSTING CLIENTS

***Abstract.*** The article offers a possible solution to the problem of spot controlling network traffic restrictions for web-hosting clients running virtual dedicated servers on virtualization nodes under KVM hypervisor or using OpenVZ virtualization technology running on Linux operating system. The proposed solution is an asynchronous information system that centrally stores information about the current location of all virtual machines (which is important if there are a large number of virtualization nodes, when in order to distribute the load servers can migrate from one host to another), assigned to them network addresses, as well as allowing a real-time web interface to manage blacklists to block individual virtual machines or specific ip-addresses, with the possibility of delayed blocking and unblocking. Describes and justifies the choice of system architecture and database management system with a detailed description of the data storage scheme itself.

***Keywords:*** Virtualization, Linux, KVM, OpenVZ, virtual machines, container virtualization, access restriction, blocking network traffic, virtualization node, database, asynchronous programming.

## Введение

С появлением альтернативы физическим выделенным серверам, целиком и полностью находящимся под управлением одного клиента в виде виртуальных выделенных серверов, как более простого в создании и обслуживании и, как следствие, более дешевого варианта, когда на одном физическом сервере может находиться довольно большое количество виртуальных машин, каждая из которых имеет собственный сетевой адрес (или несколько адресов), перед хостинг-провайдерами встаёт вопрос: каким образом точечно ограничивать входящий или исходящий сетевой трафик для конкретного адреса, либо виртуального сервера?

Классический вариант предоставления услуг хостинг провайдерами - выделенные серверы с фиксированным набором ресурсов, обусловленным техническими характеристиками подобранного оборудования. Само оборудование стоит недешево, что влечет за собой и весьма высокую стоимость аренды для клиентов, чтобы обеспечить должный уровень окупа-

емости всего набора комплектующих. Однако не всем клиентам необходимы полные возможности сервера 100% времени и не все готовы переплачивать за неиспользуемые ресурсы.

Ситуация в корне изменилась с появлением и активным распространением технологий виртуализации. На одном узле виртуализации может находиться сразу несколько виртуальных серверов, полностью автономных и независимых друг от друга. Они будут иметь свой заявленный объём оперативной памяти и дискового пространства, но количество процессорного времени (в относительных величинах) и операций ввода-вывода в единицу времени может изменяться от общей нагрузки на узел виртуализации, создаваемой другими клиентами [1]. Таким образом, ряд характеристик перестал иметь фиксированное значение, а сместился в сторону некоторой гарантированной величины, ниже которой обозначенные характеристики не опустятся, но могут колебаться на интервале сверх этой величины. Это позволяет значительно снизить время простоя оборудования, в среднем повысив срок окупаемости комплектующих, и, как следствие, снизить стоимость аренды для конечного пользователя, не говоря уже о том, что создать новую виртуальную машину на узле гораздо быстрее, чем собрать выделенный сервер.

Новые технологии влекут и новые технические задачи. Если способ ограничить сетевой трафик к выделенному серверу только один - это можно сделать извне, настройками на сетевом оборудовании, то, как ограничить доступ к конкретной виртуальной машине по сети через определенный порт? К примеру, система мониторинга зафиксировала резкое увеличение объёма почтового трафика, поддерживающее высокие значения в течение долгого времени? Или соответствующая жалоба на спам пришла извне от надзорных органов?

Далее в статье предлагается реализация информационной системы централизованного управления блокировками виртуальных машин клиентов на узлах виртуализации хостинг-провайдера, позволяющей стандартными средствами, по умолчанию встроенными во все ведущие дистрибутивы Linux, реализовать описанную функциональность.

**Описание системы.**

Для полноценного функционирования предлагаемая программная система должна состоять из модуля синхронизации данных обо всех кластерах, узлах виртуализации, входящих в состав этих кластеров, а также информации обо всех виртуальных машинах и их сетевых адресах, актуальных в любой момент времени. Конечно, не удастся обойтись без базы данных, в которой как раз и будет содержаться актуальная информация [2]. Также для полноценной работы системе необходим веб-сервер, способный в любой момент времени предоставить тот или иной список ip-адресов, представленных к блокировке, как в форме веб-страницы с удобочитаемым интерфейсом для контроля ответственным оператором, так и в виде списка адресов, который смогут запрашивать узлы виртуализации и своевременно обновлять наборы правил в своих межсетевых фильтрах iptables (утилита командной строки для управления сетевым экраном, по умолчанию предустанавливаемая в большинство дистрибутивов Linux [3]). Здесь же отметим, что речь идет именно о системах виртуализации под управлением систем на ядре Linux (в первую очередь наиболее популярные и активно развивающиеся в последнее время KVM и OpenVZ), так как для них ещё нет каких-то устоявшихся, проверенных временем способов решения описанных задач. И, наконец, последний, но не менее важный компонент — модуль отправки уведомлений пользователям о том, что они были внесены в тот или иной список, автоматически предоставляющий всю необходимую информацию, от пунктов договора, которые были нарушены до шагов исправления нарушений, а также возврата к состоянию сетевой доступности. Приведём схему программной системы на рисунке 1.

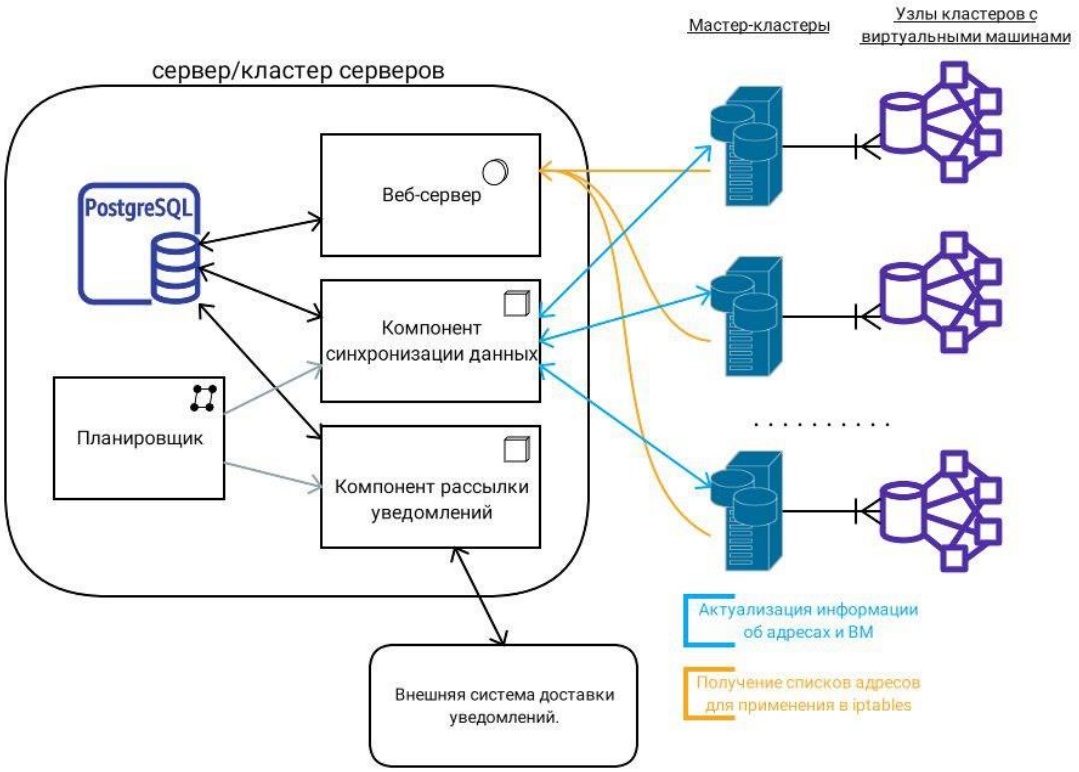


**Рис. 1. Общая схема модуля управления блокировками пользователей**

**Моделирование базы данных**

Опишем, какие данные нам необходимо будет хранить в базе данных. Основные логические компоненты, это, конечно же, виртуальные машины и ip-адреса, к ним привязанные. Между этими двумя сущностями логическая связь один ко многим, так как к одной виртуальной машине могут быть привязаны несколько ip-адресов. В то же время у компании в подавляющем большинстве случаев уже есть база клиентов, так называемая биллинговая информационная система (от англ. bill – счёт), через которую происходит всё взаимодействие клиентов с технической поддержкой и сотрудниками других отделов, управление списанием денежных средств за использование услуг (здесь - работу виртуальных машин), а также полный учёт используемых клиентом услуг. Потому нет никакого смысла дублировать какую-либо дополнительную информацию о клиентах в базе проектируемой системы, оставим лишь поля для хранения идентификаторов клиентов в базе внешней биллинг-системы, чтобы при желании можно было получить все необходимые данные уже через предоставляемый биллинг-системой интерфейс [4].

Сами виртуальные машины - не абстрактные объекты, существующие в вакууме, они должны располагаться на «хостовых» машинах, осуществляющих виртуализацию, так называемых узлах виртуализации (далее для краткости будем именовать их «нодами», от англ. node – узел), а ноды, в свою очередь, работают под управлением так называемых мастер-кластеров, агрегирующих в реальном времени информацию о работе узлов, и позволяющих переносить виртуальные машины с одного узла на другой для равномерного распределения нагрузки на систему. Некоторые системы позволяют также проводить живую миграцию, что многократно упрощает управление кластерами виртуализации. На мастер-кластеры обычно устанавливаются специализированные панели управления, так для KVM и OpenVZ виртуализации довольно часто применяется программный комплекс VMmanager [5] или некоторые его аналоги. Для большей наглядности на рисунке 2 отображены основные логические сущности [6] системы на диаграмме в нотации Питера-Чена.

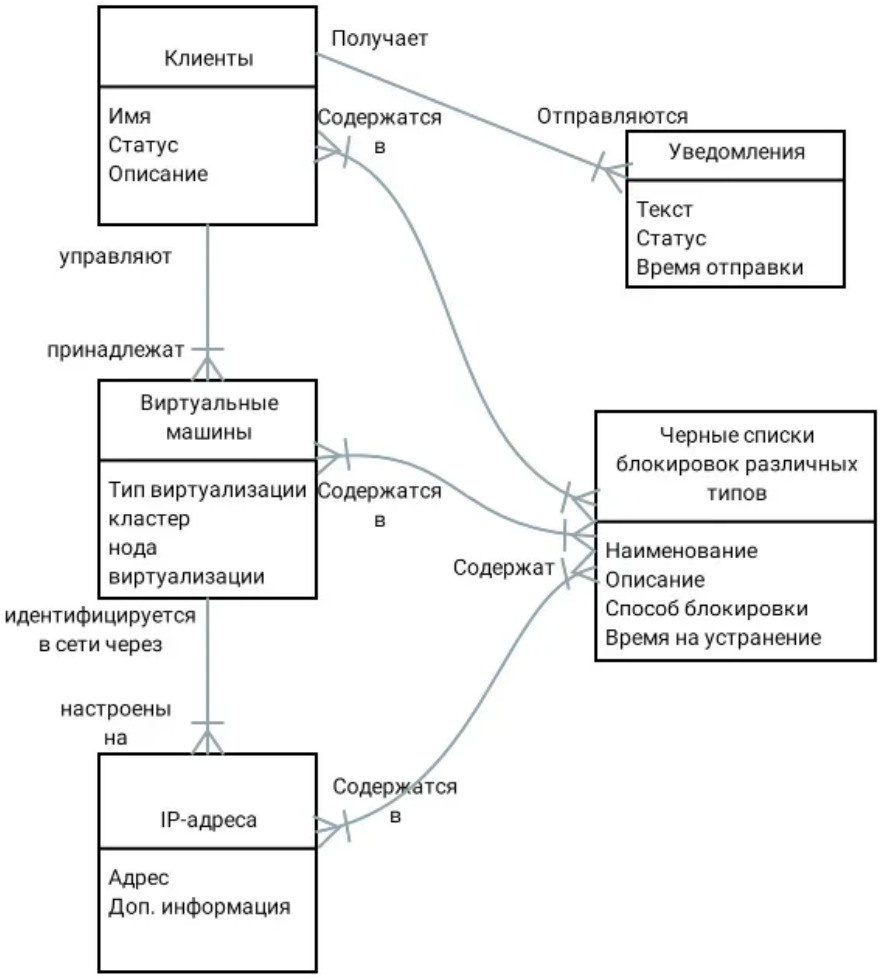


**Рис. 2. ER-диаграмма проектируемой системы в нотации Питера-Чена**

Непосредственно в базе данных, в целях её нормализации, безусловно, стоит разнести мастер-кластеры и узлы виртуализации по отдельным таблицам, отделив их от виртуальных машин.

Теперь отметим, что блокировки тоже могут иметь различный характер. Два основных вида блокировок составляют полная блокировка всего интернет-трафика и избирательная блокировка только почтового, т.н. smtp-трафика. Помимо перечисленных категорий можно выделить ещё целый ряд методик блокировки, и для хранения всех этих типов в нашей базе также необходимо выделить отдельную таблицу. Озаглавим её list, так как внутри будет храниться не что иное, как наименования чёрных списков.

Следующий немаловажный момент – объект, к которому необходимо применить штрафные санкции. В различных ситуациях может потребоваться как ограничить трафик как для конкретных ip-адресов, так и для виртуальной машины (или нескольких машин) в целом. Так как на интерфейсах одной виртуальной машины может быть настроено несколько адресов – необходимо логически разделять эти сущности, что на практике выльется в две отдельные таблицы в базе данных (vm и ip соответственно). Также сформируем отдельные таблицы для связи основных сущностей со списками блокировки, озаглавив их соответствующим образом. Для наглядности приведём логическую схему базы данных на рисунке 3.

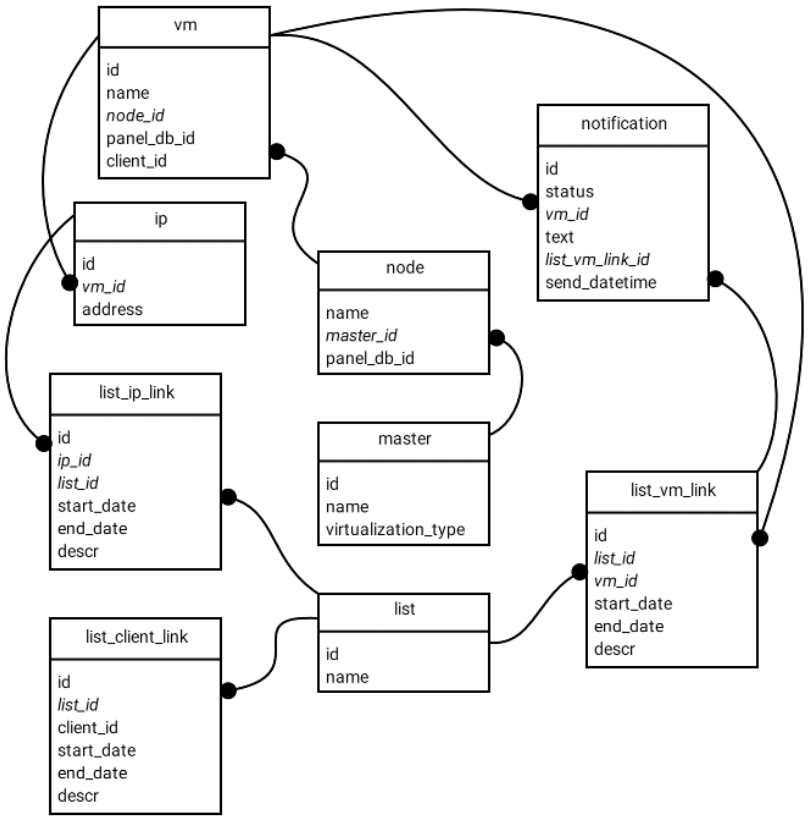


**Рис. 3. Модель базы данных системы управления блокировками пользователей**

**Выбор системы управления базой данных**

Так мы предполагаем, что наша информационная система будет работать на выделенном сервере, либо кластере серверов и с очень высокой долей вероятности сервер будет управляться unix-подобной операционной системой, было решено присмотреться к соответствующим системам управления базами данных, которые используются в связке с linux в подавляющем большинстве случаев. Также не стоит забывать, что в первую очередь проектируемая информационная система призвана решать задачи бизнеса, потому в первую очередь обратим внимание не на дорогостоящих гигантов вроде Oracle с его промышленной СУБД, а на свободно распространяемые бесплатные продукты, среди которых: MongoDB, PostgreSQL, MySQL, SQLite, Percona, MariaDB, и Firebird. На данном этапе можно с уверенностью сказать, что в проектируемой системе предполагается довольно интенсивное использование базы данных, в связи с чем у нас появляется довольно серьезное требование к способности обрабатывать множество параллельных запросов, и делать это достаточно быстро, не требуя при этом запредельного количества ресурсов. Сами по себе запросы не будут слишком сложными, затрагивающими больше количество таблиц, максимум, который действительно стоит принимать во внимание на операциях выборки – соединение того или иного вида до четырех таблиц и несколько дополнительных условий выборки, по которым будут отбираться, например, заблокированные в данный момент клиенты, их виртуальные машины или адреса, находящиеся в пользовании соответственно обозначенным временным интервалам действия санкций.

Довольно важен в рассматриваемом случае высокий уровень защищенности информации, но, стоит признать, что большее значение (особенно при такой высокой нагрузке и исполнении множества параллельных запросов одновременно) имеет целостность информации. Из приведённых выше систем под описанные критерии хорошо подходят две СУБД: MySQL и PostgreSQL. Учитывая особенности нашей предметной области можно сказать, что нам нужна производительная, безопасная СУБД, которая при этом строго следует стандартам языка SQL, а также в полной мере соответствует требованиям ACID (Atomicity, Consistency, Isolation и Durability, или в переводе атомарности, согласованности, изолированности и

прочности). В большей степени описанным параметрам удовлетворяет PostgreSQL, которая весьма эффективно управляет параллельным доступом посредством многоверсионности (MVCC), предотвращает повреждение данных и сохраняет их целостность на транзакционном уровне [7], а также предоставляет встроенный способ работы с нестандартными типами данных для различных структур (ip-адресов и JSON-подобных текстовых конструкций), что в нашем случае явно не будет лишним и позволит сэкономить немного ресурсов на преобразовании подобных данных в обыкновенные строки и обратно.

И вот теперь, подробно описав внутреннюю структуру базы данных в таблицах, мы можем воспроизвести в выбранной системе управления нашу базу данных по её логической модели с учётом связей и типов данных атрибутов. Отобразим физическую модель на рисунке 4.

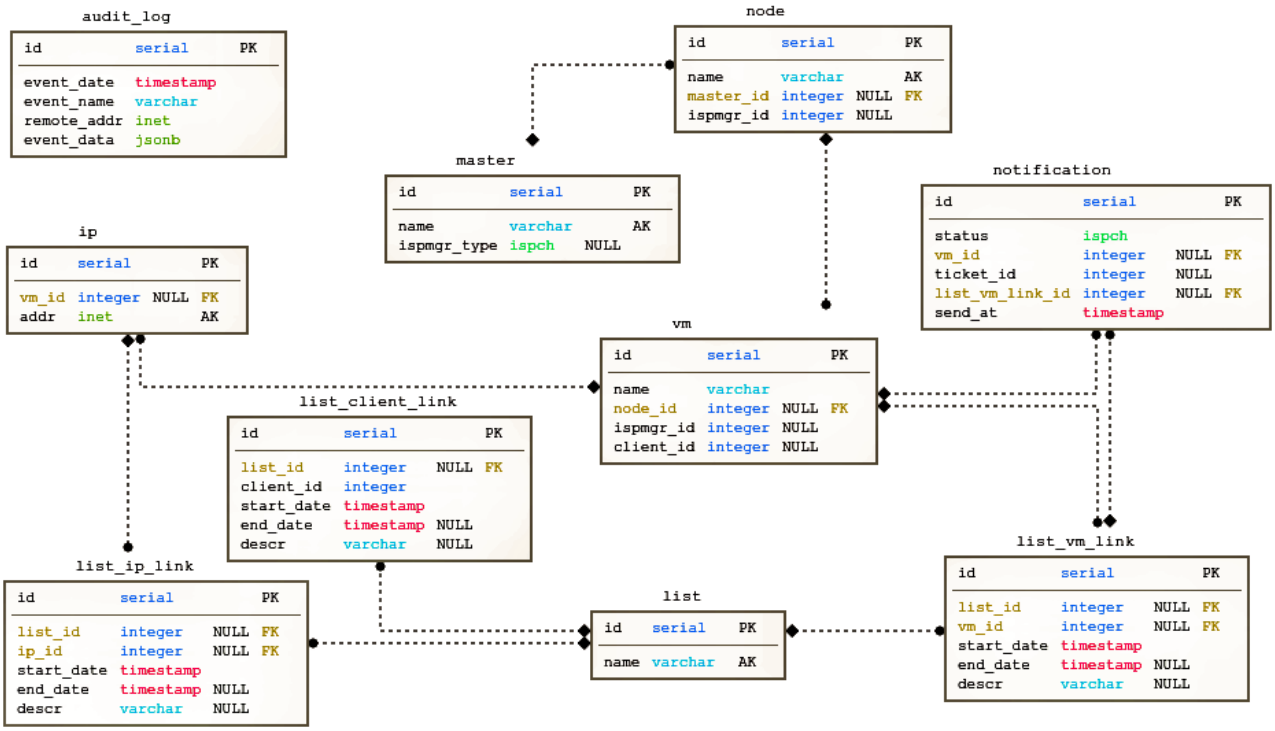


**Рис. 4. Физическая модель БД системы управления блокировками пользователей PostgreSQL**

### Выбор модели функционирования системы

Еще одно решение, которое необходимо принять на этапе проектирования — на основе какой модели осуществления операций будет действовать система. Асинхронная модель призвана в ряде случаев заменить собой классическую синхронную модель, значительно увеличив общую производительность и скорость выполнения функций, плотно завязанных на операциях ввода-вывода. В синхронной программе все действия выполняются строго последовательно, в том порядке, в котором они записаны в исходном коде программы, этот вариант является самым простым.

В асинхронной программе всё несколько сложнее. Если рассматривать однопоточный вариант (а заводить несколько потоков по обозначенным выше причинам нам нет никакого смысла), то в нём всё также происходит выполнение лишь одной задачи в каждый отдельно-взятый момент времени, но есть возможность особо затратные за счёт ожидания чего-либо по времени задачи приостанавливать до момента, пока это ожидание не закончится. Если точнее, мы имеем возможность запустить эту задачу в неблокирующем режиме системного вызова, при котором поток программы сможет продолжать обработку. Ожидать задачи могут, к примеру, ответа удалённого сервера, ответа базы данных, завершения работы запущенного подпроцесса, либо же, как в нашем случае, просто работы таймера, позволяющего программе не терять ритм, а выполнять определенные действия строго с заданной периодичностью. То есть в те самые моменты простоя операций, контекст выполнения программы бу-

дет передаваться другим операциям, ожидающим в этот момент своей очереди [8]. Так, в приложении к исследуемой задаче, во время работы таймера свободно могут выполняться другие задачи: обращения к базе данных (в ожидании ответов от которой, к слову, тоже можно нагрузить процессор работой, если найдутся невыполненные задачи), либо же инициализация рассылки уведомлений.

Продемонстрируем процесс работы синхронной и асинхронной программ, выполняющих HTTP-запрос к удалённому серверу на рисунке 5.

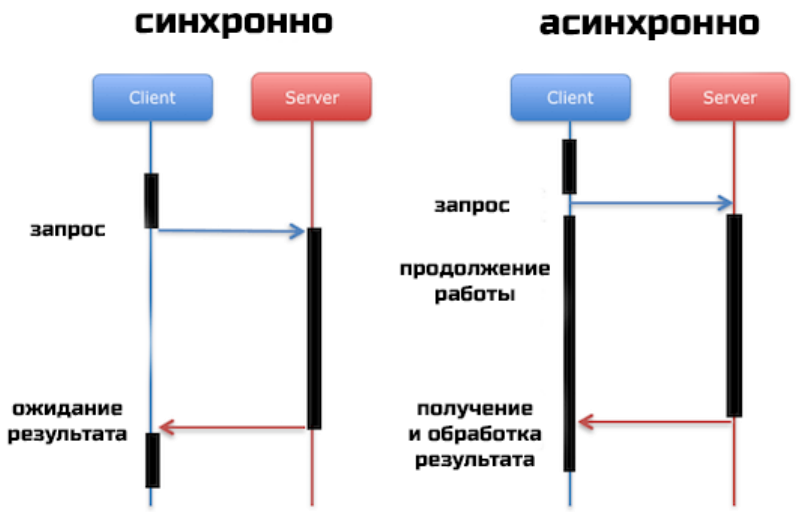


**Рис. 5. Сравнение принципов работы синхронной и асинхронной программ**

Так как модуль синхронизации данных в итоге должен будет связываться с большим количеством мастер-кластеров – асинхронный однопоточный программный алгоритм придётся как никогда кстати. Запросы на получение информации об узлах виртуализации, виртуальных машинах и ip-адресах будут инициироваться внутри так называемых корутин, и сразу же передавать управление следующей корутине [8]. Фокус потока исполнения переключится обратно на эту функцию лишь в тот момент, когда данные от мастер-кластера будут получены и подготовлены к обработке. Это сэкономит нам довольно много времени и в целом уменьшит длительность одной итерации синхронизации, за счёт использования промежутков простоя, в синхронной вариации вызванного операциями ввода-вывода. Следуя принципу универсальности остановим выбор языка программирования для нашей информационной системы на Python, в стандартный набор библиотек которого с версии 3.5 входить библиотека асинхронных операций asyncio. Интерпретатор python доступен для установки из репозиториев всех основных unix систем. В качестве основы асинхронного веб-сервера возьмём библиотеку aiohttp, поддерживаемую официальными разработчиками asyncio, чтобы обеспечить максимально стабильную и быструю работу нашей системы.

**Заключение**

Описанная система способна помочь хостинг-провайдерам, предоставляющим услуги виртуальных выделенных серверов под управлением гипервизора KVM, или с использованием технологии виртуализации OpenVZ решить проблему управления ограничениями сетевого трафика для определенных сетевых адресов или виртуальных машин, независимо от узла виртуализации, на котором находится сервер. При этом управление черными списками производится централизованно через веб-интерфейс, а сама система за счёт асинхронной модели функционирования сможет максимально эффективно использовать ресурсы, предоставленные под её работу, сведя время ответа к минимуму. Установка сопутствующих скриптов автоматизации обновления iptables после небольшой адаптации под инфраструктуру провайдера на узлах виртуализации также не вызовет особых проблем, за счёт распространенности используемых инструментов.

## БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Дэйкинч В., Веттатху Э., Прасад М. Виртуализация KVM, Полное руководство, 2е изд // Современные технологии. Системный анализ. Моделирование. - Packt Publishing Ltd. 2020. 157 с. - ISBN 978-1-83882-871-4

2. Георгий Исаев: Проектирование информационных систем. Учебное пособие - 1-е издание - ред. Григораш М. Л. Омега-Л, 2015 г. - 424 с. - ISBN: 978-5-370-02508-2

3. Майкл Раш Сетевые экраны Linux, обнаружение и пресечение угроз с iptables, psad и fwsnort - No Starch Press, 2007 г. – 336 с. - ISBN: 978-1-593-27141-1

4. «Биллинговые системы: основные понятия» // ixbt.com [Электронный ресурс] URL: https://www.ixbt.com/mobile/review/billing.shtml (17.04.2021)

5. «VMmanager — платформа управления виртуализацией» // ispsystem.ru [Электронный ресурс] URL: https://www.ispsystem.ru/software/vmmanager (19.04.2021)

6. Буч Г., Рамбо Д., Джекобсон А. Язык UML. Руководство пользователя/ Г. Буч, Д. Рамбо, А. Джекобсон - 2-е изд. - М., СПб. ДМК Пресс, Питер, 2004. - 432 с. - ISBN 5-94074-260-2.

7. Кэмпбелл Л., Мейджорс Ч., Базы данных. Инжиниринг надежности - СПб. ДМК Пресс, Питер, 2004. - 304 с. - ISBN 978-5-4461-1310-1.

8. Хаттинг К. – Использование Asyncio в Python: Понимание особенностей асинхронного программирования Python 1-е издание - O'Reilly Media, 2020. - 166 с. – ISBN 978-1492075332

## REFERENCES

1. Dakinch V., Vettathu E., Prasad M. KVM Virtualization, Complete Guide, 2nd ed. Systems Analysis. Modeling. - Packt Publishing Ltd. 2020. 157 c. - ISBN 978-1-83882-871-4

2. Georgy Isaev: Designing Information Systems. Tutorial - 1st edition - ed. by Grigorash M. L. Omega-L, 2015. - 424 c. - ISBN: 978-5-370-02508-2

3. Michael Rush Linux firewalls, detecting and stopping threats with iptables, psad and fwsnort - No Starch Press, 2007 - 336 p. - ISBN: 978-1-593-27141-1

4. "Billing systems: basic concepts" // ixbt.com [Internet resource] URL: https://www.ixbt.com/mobile/review/billing.shtml (17.04.2021)

5. "VMmanager - virtualization management platform" // ispsystem.ru [Internet resource] URL: https://www.ispsystem.com/software/vmmanager (19.04.2021)

6. Butch G., Rambo D., Jacobson A. UML Language. User's guide / G. Buch, D. Rambo, A. Jacobson - 2nd ed. - M., SPb. DMK Press, Peter, 2004. - 432 c. - ISBN 5-94074-260-2.

7. Campbell L., Majors C., Databases. Reliability engineering - SPb. DMK Press, Peter, 2004. - 304 c. - ISBN 978-5-4461-1310-1.

8. Hattingh K. - Using Asyncio in Python: An Introduction to Asynchronous Programming in Python 1st Edition - O'Reilly Media, 2020. - 166 c. - ISBN 978-1492075332

### Информация об авторах

*Алексей Васильевич Данеев* – д. т. н., профессор, профессор кафедры «Информационные системы и защита информации», Иркутский государственный университет путей сообщения, г. Иркутск, e-mail: daneev@mail.ru

*Вадим Сергеевич Тютиков* – студент, Иркутский государственный университет путей сообщения, г. Иркутск, e-mail: tyutikoff2010@yandex.ru

### Authors

*Alexey Vasilievich Daneev* – Doctor of Technical Science, Professor, the Subdepartment Information systems and information security, Irkutsk State Transport University, Irkutsk, e-mail: daneev@mail.ru

*Vadim Sergeevich Tyutikov* - student, Irkutsk State Transport University, Irkutsk, e-mail: tyutikoff2010@yandex.ru

**Для цитирования**

Данеев А.В., Тютиков В.С. Проектирование информационной системы управления блокировками сетевого трафика виртуальных машин клиентов хостинга // Информационные технологии и математическое моделирование в управлении сложными системами: электрон. науч. журн. 2021. – №4(12). – С. 11-19 – DOI: 10.26731/2658-3704.2021.4(12).11-19 – Режим доступа: http://ismm-irgups.ru/toma/412-2021, свободный. – Загл. с экрана. – Яз. рус., англ. (дата обращения: 27.01.2022)

**For citations**

Daneev A.V., Tyutikov V.S. Designing an information system to manage network traffic blocking of virtual machines of hosting clients // Informacionnye tehnologii i matematicheskoe modelirovanie v upravlenii slozhnymi sistemami: ehlektronnyj nauchnyj zhurnal [Information technology and mathematical modeling in the management of complex systems: electronic scientific journal], 2021. No. 4(12). P. 11-19. DOI: 10.26731/2658-3704.2021.4(12).11-19 [Accessed 27/01/22]