ВИРТУАЛЬНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СЕТИ ДЛЯ СЛУЧАЯ АППАРАТНОЙ ВИРТУАЛИЗАЦИИ

VIRTUAL COMPUTING NETWORKS FOR HARDWARE VIRTUALIZATION

A. Cubahiro
M. Kamande

Summary. This article concerns questions related to networking among virtual machines that utilize hardware virtualization. The case of QEMU + KVM usage in OS Linux is analyzed in details. Methods for providing such virtual machines with networking are discussed. Two virtual interfaces for this case for OS Linux are discussed: «TUN/TAP» interface and «macvtap» interface. Different aspects of these two virtual interfaces are also discussed. Such interfaces could be used not only by hypervisor, but also by programs that provides some special networking services (e.g. VPN). But in this article these interfaces are discussed in depth only for the case of hypervisor. Thereafter test nodes characteristics are given with the description of the test cases that were used to test performance of these interfaces.

Keywords: Networking, virtual machines, virtual networks, Linux, QEMU, KVM.

Чубахиро Амисси
Аспирант, Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина)
amcubahiro@gmail.com
Каманде Магдалине Вамбуи
Аспирант, Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина)
magdalynde@gmail.com

Аннотация. В данной статье рассматриваются вопросы сетевого взаимодействия в рамках виртуальных машин, для работы которых используется аппаратная виртуализация. В деталях проанализирован случай виртуализации в рамках ОС Linux с использованием QEMU+KVM. Рассмотрены способы организации сетевого взаимодействия в рамках виртуальных машин. Рассмотрена работа двух виртуальных интерфейсов для данного случая: интерфейса «TUN/TAP» и «macvtap». В конце приведены и проанализированы результаты тестирования виртуальных вычислительных сетей.

Ключевые слова: Сетевое взаимодействие, виртуальные машины, виртуальные сети, Linux, QEMU, KVM.

Введение

Использование аппаратной виртуализации нашло широкое применение для самых разнообразных задач в последние годы. В некоторых случаях использование данной технологии стало уже едва ли не стандартном де-факто. Безопасность и изоляция ресурсов, снижение совокупной стоимости владения, гибкость конфигурирования и выделения ресурсов — вот далеко не полный список преимуществ, предоставляемых этой технологией. Вопросы виртуализации рассмотрены, например, в [1]. Вопросы обеспечения сетью виртуальных машин, использующих данную технологию особенно важны, ведь такие виртуальные машины зачастую используются именно для обеспечения сетевых сервисов. К примеру, такая виртуальная машина может быть использована для запуска веб-сайта. Хостинг-провайдер может использовать несколько узлов, на которых будут запущено множество сайтов. Для каждого сайта может быть использован отдельный веб-сервер. Веб-серверы могут быть запущены в разных ОС. Использование аппаратной виртуализации отменяет привязку узла к одной запущенной в данный момент операционной системе. На узле запускается гипервизор, который может быть установлен как «на железо», так и в рамках хостовой ОС. С помощью гипервизора запускается множество виртуальных машин, на которых могут быть запущены различные ОС. Поскольку обеспечение сетевых сервисов — одна из типичных задач в данном случае, рассмотрение вопросов производительности виртуальных вычислительных сетей для такого типа виртуализации является важной и актуальной задачей. К примеру, в [2] и [3] рассматриваются вопросы производительности для различных технологий виртуализации.

На данный момент существует множество решений для аппаратной виртуализации. Есть как платные, так и бесплатные решения. Есть решения как с закрытым исходным кодом, так и с открытым исходным кодом. В рамках данной статьи рассматривается вариант применения QEMU [4] с использованием KVM в рамках ОС Linux. Вопросы виртуальных вычислительных сетей рассматриваются именно для этого случая.

Организации сетевого взаимодействия для случая аппаратной виртуализации

В общем случае для работы виртуальных машин с внешней сетью гипервизору необходимо выполнять разделение общего ресурса — физического сетевого адаптера — между множеством запущенных виртуальных машин.

В рамках ОС Linux для решения этой задачи используются виртуальные сетевые интерфейсы, наиболее популярным из которых, пожалуй, является интерфейс «TUN/TAP». Другой возможный вариант — интерфейс «macvtap». В рамках данной статьи будут рассмотрены именно эти два варианта.

И в том, и в другом случае в рамках системы создается виртуальный сетевой интерфейс (представленный экземпляром структуры «net_device»). Программа-гипервизор работает с подобным интерфейсом. Ей необходимо отправлять кадры виртуальной машины и передавать кадры из внешней сети (или от других виртуальных машин на узле) в обратном направлении. Виртуальный сетевое устройство должно, с одной стороны, предоставлять интерфейс для работы с ним программы-гипервизора, а с другой стороны, должно использовать интерфейс для отправки и получения кадров. Рассматриваемые виртуальные устройства предоставляют интерфейс в виде виртуального файла для программы-гипервизора. Для отправки и получения кадров используются интерфейсы, предоставляемые системой. Далее будут рассмотрены особенности работы упомянутых виртуальных сетевых устройств.

Интерфейс «TUN/TAP»

Устройство подобного типа используется довольно часто для обеспечения сетью виртуальных машин или для реализации специальных сервисов (например, VPN). При использовании данного интерфейса может быть использовано два типа устройств — устройства «TUN» и устройства «TAP». Главное отличие состоит в том, что устройства «TUN» работают с третьим уровнем модели OSI, а устройства «TAP» — со вторым. Поэтому для обеспечения виртуальной машины сетью, как правило, используются устройства «TAP»: гостевой ОС требуется отправить подготовленные кадры. Программа-гипервизор предоставляет виртуальной машине виртуальный сетевой адаптер, который с точки зрения виртуальной машины функционирует как обычный физический сетевой адаптер. Для физического сетевого адаптера гостевая ОС подготавливает кадр, заполняя заголовок Ethernet. Таким образом, от программы-гипервизора требуется отправить готовые кадры. Именно поэтому такие программы, как правило, используют устройства «TAP».

И для работы устройств «TUN», и для работы устройств «TAP» используется один и тот же драйвер. Отличие заключается во внутренних особенностях работы для одного и для другого случая. А выбор типа устройства («TUN» или «TAP») выполняется с помощью флага для системного вызова «ioctl». Для того, чтобы создать устройство «TUN» или устройство «TAP», программа должна выполнить следующее:

- открыть виртуальный файл устройства для чтения и записи: для работы с таким файлом, как правило, требуются специальные привилегии (определяемые «capabilities»); обычно это файл «/dev/net/tun», хотя может быть и другой файл; обычно он уже создан системой; в случае отсутствия подобного файла, его можно создать, например, из консоли при помощи «mknod»;
- выполнить системный вызов «ioctl» с нужными параметрами;
- когда указанные действия выполнены, программа сможет записывать и читать данные с использованием полученного файлового дескриптора.

В случае интерфейса «TAP» программа должна записывать полностью сформированные кадры с заголовком «Ethernet». Поскольку программы-гипервизоры, как правило, используют именно устройства «TAP», речь в дальнейшем пойдет именно о них.

Данные, записанные программой, будут интерпретированы как кадр: будут сформированы структуры данных, необходимые для обработки кадра, полученного системой по сети. Эти данные «появятся» в системе, как полученный кадр на сетевом виртуальном интерфейсе «TAP». Таким образом, программа, работающая с виртуальным файлом, связанным с интерфейсом «TAP», должна заполнить заголовки всех уровней модели OSI («Open System Interconnection») со второго уровня по седьмой. Это как раз подходит для случая использования виртуальных машин: виртуальная машина заполняет указанные заголовки и передает кадр для отправки виртуальному сетевому адаптеру, за работу которого отвечает программа-гипервизор. Далее программа-гипервизор записывает такой кадр в виртуальный файл (с использованием полученного дескриптора), связанный с интерфейсом «TAP», и кадр «появляется» в системе, как принятый на этом виртуальном интерфейсе.

Однако остается вопрос — как наладить передачу данных между виртуальными машинами или между виртуальной машиной и узлом во внешней сети. Для этих целей в ОС Linux, как правило, используется виртуальный мост — устройство типа «bridge». Это устройство работает как настоящий сетевой мост. Оно работает на втором уровне модели OSI и оперирует кадрами. В виртуальный мост могут быть добавлены сетевые интерфейсы, присутствующие в системе (при добавлении экземпляр структуры «net_device», соответствующий интерфейсу будет связан с виртуальным мостом благодаря использованию внутренних полей структуры данных). При передаче кадра мосту от одного из включенных в него интерфейсов, мост определяет порт назначения (устройства включаются в виртуальные порты моста) и отправляет кадр нужному устройству (или на все пор-

ты, кроме того, на котором появился кадр, в том случае, когда мост не может определить порт назначения по MAC-адресу назначения кадра). Кадр в итоге будет передан нужному интерфейсу — будет вызвана функция интерфейса для отправки кадра. Таким интерфейсом может быть физический сетевой адаптер — в этом случае кадр будет передан во внешнюю сеть.

Теперь необходимо рассмотреть, как виртуальная машина, использующая интерфейс «TAP», получает кадр (из внешней сети или от другой виртуальной машины). Поскольку интерфейс «TAP» является сетевым интерфейсом (пусть и виртуальным), с ним связан ряд функций, которые должны быть определены для сетевого интерфейса (физического или виртуального), в том числе и функция для отправки кадра. Интерфейс «TAP», как правило, включается в виртуальный мост. При получении кадра для виртуальной машины, такой мост в итоге должен вызвать функцию для отправки кадра интерфейса «TAP» — такие функции реализованы в драйверах сетевых устройств. В случае интерфейса «TAP» эта функция в итоге ставит кадр в очередь для приема кадров. Такая очередь связана посредством внутренних структур данных с дескриптором открытого виртуального файла устройства «TAP». После добавления кадра в очередь, если процесс ожидает появления данных на этом дескрипторе (например, выполняя системный вызов «read»), он будет «разбужен». Однако процесс может и не ждать данных. Так или иначе, когда процесс попытается прочитать данные по этому дескриптору, ему будет передан полученный кадр. Так, программа сможет прочитать кадр. Программа-гипервизор, получив кадр (из внешней сети или от другой виртуальной машины) передаст этот файл виртуальной машине. Таким образом виртуальная машина сможет получить кадр. Поскольку гостевая ОС воспринимает виртуальный сетевой интерфейс как обычный физический сетевой интерфейс, она ожидает получения именно кадров на таком интерфейс. И интерфейс «TAP» обеспечивает получение кадров. Так, с помощью виртуального моста виртуальные машины могут обмениваться кадрами посредством устройства «TAP» (используемого программой-гипервизором) как с узлами во внешней сети, так и с другими виртуальными машинами. Для использования интерфейса «TAP» с QEMU можно указать, например, следующие опции «-net nic -net tap, name=tap0». Схематично работа интерфейса «TAP» вместе с виртуальным мостом изображена на рис. 1.

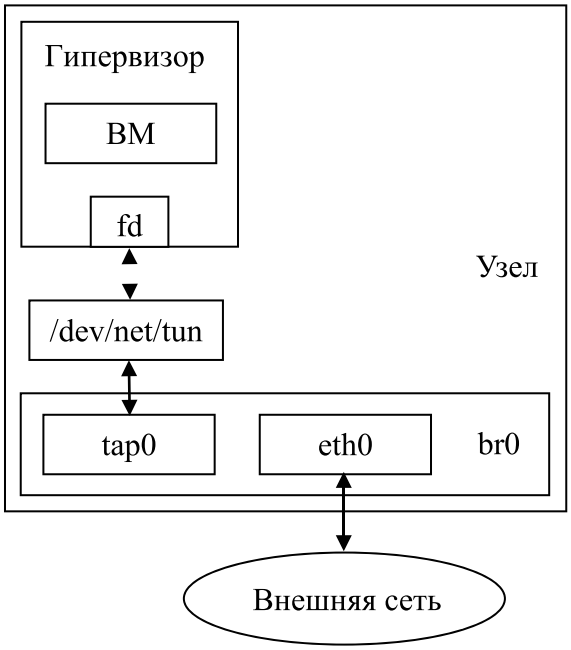

Рис. 1

На рис. 1 «tap0» является виртуальным интерфейсом «TAP», он связан с программой-гипервизором через виртуальный файл «/dev/net/tun»: программа-гипервизор открыла этот файл для чтения и записи, получив файловый дескриптор «fd», и выполнила системный вызов «ioctl» с нужными параметрами для этого файлового дескриптора, после чего в системе появился виртуальный интерфейс «tap0». После этого программа-гипервизор может отправлять кадры виртуальной машины и получать кадры для нее с использованием полученного дескриптора. При этом с дескриптором в системе ассоциируются необходимые функции — так, при записи будет вызвана необходимая функция драйвера «TUN/TAP», как и при чтении. Можно также отметить, что с одним виртуальным файлом, связанным с устройством «TAP», смогут работать множество программ (например, программ-гипервизоров). Это возможно благодаря тому, что каждая программа независимо открывает этот файл для чтения и записи — для каждого такого случая создаются отдельные экземпляры необходимых структур данных. И для каждого такого случая после выполнения необходимых действий (например, системного вызова «ioctl») будет создано отдельное виртуальное сетевое устройство «TAP». При закрытии файла, виртуальное устройство должно быть удалено. Однако есть возможность создания «постоянных» устройств «TAP» с помощью флага «TUNSETPERSIST» (задаваемого через системный вызов «ioctl») — в этом случае виртуальное устройство останется в системе даже после завершения работы программы.

Интерфейс «macvtap»

Данный интерфейс работает аналогично интерфейсу «TAP». Он даже обеспечивает определенную совместимость с интерфейсом «TAP» для программ, использующих интерфейс «TAP» (например, ряд общих флагов для системного вызова «ioctl»). Однако интерфейс «macvtap»

работает по принципу интерфейса «macvlan» и не требует моста для обеспечения коммуникации между виртуальными машинами, использующими такой интерфейс с одним и тем же «нижележащим» устройством, а также узлами во внешней сети.

Разница между интерфейсом «macvtap» и «macvlan» состоит в том, что «macvtap», помимо предоставления виртуального сетевого устройства, предоставляет виртуальный файл для работы программы с этим файлом (например, программы-гипервизора). А «macvlan» предоставляет лишь виртуальное сетевое устройство. Таким образом, устройство «macvlan» может быть использовано лишь в рамках системы, либо контейнера. Имеется ввиду, что устройство «macvlan» может быть использовано для отправки и получения данных, а не кадров — оно может быть использовано как любое другое сетевое устройство в системе (например, физический сетевой интерфейс). Конечно, для получения и отправки кадров могут быть использованы «сырые сокеты» (домена «AF_PACKET»), однако, такое решение имеет ряд ограничений: все «сырые сокеты» получают все кадры; как правило, требуются привилегии пользователя «root» для работы с такими сокетами. А в случае использования устройства «macvtap» программа работает с виртуальным файлом. Конечно, и в этом случае потребуются определенные привилегии (определяемые «capabilities»), однако главное отличие в способе работы. В случае «macvlan» отправка и получение кадров возможна, по сути, только с «сырыми сокетами» (с указанными особенностями), а в случае «macvtap» программа записывает и получает кадры через виртуальный файл.

Работа с виртуальным файлом, связанным с устройством «macvtap», аналогична случаю работы с виртуальным файлом устройства «TAP», описанному выше. В случае использования «macvtap» в системе также появляется виртуальное сетевое устройство. С точки зрения программы-гипервизора работа с использованием «macvtap» мало чем по сути отличается от варианта работы с «TAP». Кадры также отправляются и передаются с помощью виртуального файла. Главное отличие состоит в том, что происходит с этими кадрами после отправки программой-гипервизором (после записи), и в том, как они поступают в обратном направлении. В случае «macvtap» для обеспечения коммуникации с другими виртуальными машинами или внешней сетью не требуется виртуальный мост. Драйвер устройства «macvtap» использует ряд функций драйвера «macvlan». У них схожий принцип работы. Устройства «macvtap», как и устройства «macvlan» поддерживают несколько режимов работы. Для варианта запуска нескольких виртуальных машин на узле можно использовать режим «bridge», который и будет рассмотрен далее. Устройство «macvtap», как и «macvlan», ассоциируется с «нижележащим» устройством, которым может быть, например, физический сетевой интерфейс. В случае режима работы «bridge» все виртуальные устройства «macvtap», связанные с одним и тем же «нижележащим» устройством, могут обмениваться данными между собой: если адресат — виртуальная машина на узле, кадр в итоге будет передан ей, не покидая узла. Если же «нижележащее» устройство — физический сетевой адаптер, и адресат находится во внешней сети, кадр будет передан во внешнюю сеть через этот физический сетевой интерфейс.

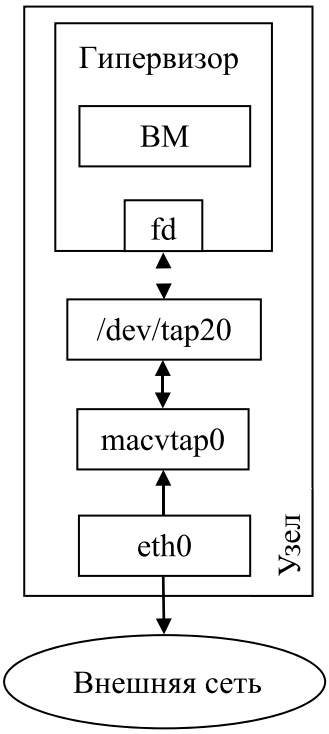

Рис. 2

С одним и тем же «нижележащим» устройством могут быть одновременно связаны не только устройства «macvtap», но и устройства «macvlan». Если устройства «macvlan» используются контейнерами, то с помощью таких виртуальных устройств «macvlan» и «macvtap» может быть налажена коммуникация между виртуальными машинами и контейнерами. Разумеется, такого же результата можно добиться и при использовании устройств «TAP» вместе с виртуальным мостом, а со стороны контейнеров — устройств «veth». Однако при использовании устройств «macvtap» и «macvlan» отдельный виртуальный мост не требуется. Схематично работа устройств «macvtap» и «macvlan» изображена на рис. 2.

На рис. 2 «macvtap0» является виртуальным интерфейсом «macvtap», он связан с программой-гипервизором через виртуальный файл «/dev/tap20»: имя вир-

Таблица 1

	Iperf, Мбит/сек	LINPACK, ГФЛОПС	GROMACS, сек	OpenFOAM тест 1, сек	OpenFOAM тест 2, сек
TAP	93.1	9.189	246.998	194	161
macvtap	93.1	9.169	246.545	192	162

туального сетевого интерфейса и виртуального файла в данном случае не связаны одним и тем же номером. В данном случая при создании нового виртуального сетевого интерфейса создается новый файл. У программы-гипервизора этот файл открыт для чтения и записи — получен файловый дескриптор «fd». Программа-гипервизор выполняет системный вызов «ioctl» для этого файлового дескриптора с необходимыми параметрами, после этого она может отправлять кадры виртуальной машины и получать кадры для нее с использованием этого дескриптора. При записи, как и при чтении будет вызвана необходимая функция драйвера «macvtap».

Следует отметить некоторые особенности работы с интерфейсом «macvtap». Во-первых, виртуальный файл в разделе «/dev», связанный с интерфейсом «macvtap», появляется только после создания виртуального сетевого устройства «macvtap» (например, с помощью «ip link»). В случае же устройства «TAP» виртуальный файл создается до появления виртуального сетевого устройства.

Во-вторых, MAC-адрес сетевого интерфейса в рамках виртуальной машины (через который гостевая ОС передает и принимает файлы) должен совпадать с MAC-адресом виртуального сетевого устройства «macvtap», с которым будет работать программа-гипервизор (в случае устройства «TAP» такого ограничения нет). Это связано со внутренними особенностями работы интерфейса «macvtap». В случае «macvtap» (как и в случае «macvlan») при получении кадра на «нижележащем» устройстве производится поиск виртуального сетевого устройства «macvtap» или «macvlan», чей MAC-адрес равен MAC-адресу назначения в заголовке Ethernet кадра. При адресации кадра виртуальной машине в заголовке кадра будет указан MAC-адрес сетевого интерфейса виртуальной машины, а не устройства «macvtap». Если эти MAC-адреса различаются, то при получении кадра, устройство «macvtap» с указанным в кадре MAC-адресом назначения найдено не будет, и кадр не дойдет до виртуальной машины. Если же MAC-адреса равны, то устройство «macvtap» будет найдено, и кадр в итоге буден доставлен виртуальной машине.

Следует также отметить, что у QEMU пока что нет стандартных опций для работы с интерфейсом «macvtap». Поэтому для использования интерфейса «macvtap» необходимо сначала открыть виртуальный файл, связанный с устройством «macvtap», для записи и чтения и затем указать файловый дескриптор (номер) через опцию «fd», например, для файлового дескриптора «3», виртуального файла «/dev/tap20» и MAC-адреса «52:54:00:12:34:80» строка запуска QEMU выглядела бы примерно следующим образом: «qemu-system-x86_64 <другие_опции> -net nic, macaddr=52:54:00:12:34:80 -net tap, fd=3 3<> /dev/tap20».

Тестирование

Для сравнения двух указанных виртуальных интерфейсов был выполнен ряд тестов. Для оценки пропускной способности сети использовался стандартный тест «iperf». Затем был выполнен тест «LINPACK». Наконец, были запущены тестовые задачи приложений «GROMACS» [5] и «OpenFOAM» [6] (для двух разных солверов).

Характеристики сети: «Fast Ethernet», скорость 100 Мбит/сек. Тесты запускались на виртуальных машинах под управлением QEMU+KVM. Хостовой ОС являлась «Fedora 26», гостевой ОС также являлась «Fedora 26» Параметры сетевого стека соответствовали значениям по умолчанию для данной ОС (описание параметров стека TCP/IP для ОС Linux приводится в [7]). Использовались узлы архитектуры «x86_64» (8-ядерный процессор) с 4 ГБ оперативной памяти. Виртуальным машинам выделялось по 8 ядер (все ядра на узле) и по 3 ГБ оперативной памяти. Тесты запускались на 2 виртуальных машинах, выполняющихся на разных узлах.

Для теста «iperf» использовались настройки по умолчанию. Тестовая задача «GROMACS» запускалась на 2 узлах, по 8 потоков на каждом узле (таким образом, должны были быть задействованы все 8 ядер процессора на каждом узле). Тестовые задачи «OpenFOAM» запускались на 2 узлах, по 8 процессов на каждом узле (аналогично должны были быть задействованы все ядра на каждом узле).

Запуск тестов в рамках хостовой ОС не выполнялся, так как аппаратная виртуализация привносит определенные накладные расходы, но оценка этих накладных расходов выходит за рамки данной статьи. В данной статье основное внимание уделяется именно сравнению виртуальных сетевых интерфейсов для случая аппарат-

ной виртуализации и возможных накладных расходов, связанных с тем или иным вариантом организации сетевого взаимодействия. Поэтому выполнялись лишь тесты на виртуальных машинах. В табл. 1 приводятся результаты выполнения тестов.

Как видно, разницы в производительности (как сети, так и производительности расчетов на тестах) практически нет. Отличия по скорости, полученной производительности и времени выполнения минимальны и могут быть объяснены влиянием случайных факторов.

Однако для окончательных выводов по поводу производительности того или иного варианта требуется больше различных тестов с разными шаблонами передачи данных, так как на производительность сетевого взаимодействия влияет большое число факторов. Но на выполненных тестах разницы в производительности практически не наблюдалось.

Выводы

Устройства «TAP» и «macvtap» позволяют обеспечить виртуальные машины, использующие аппаратную виртуализацию, в ОС Linux сетевым взаимодействием. Благодаря этим устройствам виртуальные машины могут обмениваться данными как друг с другом, так и с узлами, находящимися во внешней сети (при наличии такого подключения). Устройство «TAP» используется уже давно и достаточно широко, при этом не только программами-гипервизорами. Устройство «macvtap» предоставляет аналогичную функциональность и не требует наличия виртуального моста. Разницы в производительности на выполненных тестах в рамках данной статьи практически не наблюдалось (хотя аппаратная виртуализация сама по себе привносит определенные накладные расходы). Для устройства «TAP» у QEMU есть стандартная опция, однако «macvtap» также можно использовать с QEMU.

ЛИТЕРАТУРА

1. Tanenbaum, A. S. Modern Operating Systems [Текст] / A. S. Tanenbaum, H. Bos — 4th Edition.— Pearson Education, Inc.— 2014.— 1136 p.
2. Felter W. et al. An Updated Performance Comparison of Virtual Machines and Linux Containers [Текст] / W. Felter, A. Ferreira, R. Rajamony, J. Rubio // Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium On.— IEEE, 2015.— p. 171–172.
3. Xavier M. G. et al. Performance Evaluation of Container-Based Virtualization for High Performance Computing Environments [Текст] / M. G. Xavier, M. V. Neves, F. D. Rossi, T. C. Ferreto, T. Lange, C. A. F. De Rose // Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on.— IEEE, 2013.— p. 233–240.
4. Официальный сайт проекта QEMU [Электронный ресурс] // https://www.qemu.org/
5. Hess B. et al. GROMACS4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation [Текст] / B. Hess, C. Kutzner, D. van der Spoel, E. Lindahl // Journal of chemical theory and computation 4.3, 2008.— p. 435–447.
6. Официальный сайт проекта OpenFOAM [Электронный ресурс] // https://www.openfoam.com/
7. Параметры стека TCP/IP в ОС Linux [Электронный ресурс] // https://www.kernel.org/doc/Documentation/networking/ip-sysctl.txt

© Чубахиро Амисси (amcubahiro@gmail.com), Каманде Магдалине Вамбуи (magdalynde@gmail.com).

Журнал «Современная наука: актуальные проблемы теории и практики»

Санкт-Петербургский государственный электротехнический университет «ЛЭТИ»