Перспективы виртуализации высокопроизводительных систем архитектуры x64

А. О. Кудрявцев, alexk@ispras.ru, В. К. Кошелев, vedun@ispras.ru
А. И. Аветисян, arut@ispras.ru

Аннотация. В данной работе изучаются перспективы применения технологий виртуализации в области высокопроизводительных вычислений на платформе x64. Рассматриваются основные причины падения производительности при запуске параллельных программ в виртуальной среде. Подробно рассматриваются системы виртуализации KVM/QEMU и Palacios, в качестве тестовых пакетов используются HPC Challenge и NAS Parallel Benchmarks. Тестирование выполняется на современном вычислительном кластере, построенном на базе высокоскоростной сети Infiniband. Результаты проведенного исследования в целом показывают целесообразность применения виртуализации для большого класса высокопроизводительных приложений. Доводка рассматриваемых систем виртуализации позволила снизить накладные расходы с 10-60% до 1-5% на большинстве тестов пакетов HPC Challenge и NAS Parallel Benchmarks. Основными “узкими местами” систем виртуализации являются уменьшенная производительность системы памяти (критично только для узкого класса задач), расходы при виртуализации устройств, а также повышенный уровень “шума”, источником которого становятся основная ОС и гипервизор. Шум может оказывать негативное влияние на производительность и масштабируемость “мелкозернистых” приложений (приложений с частыми коммуникациями небольшого объема). При увеличении числа узлов в системе, влияние шума существенно усиливается.

Ключевые слова: высокопроизводительные системы; виртуализация; мониторы виртуальных машин; высокопроизводительные вычисления; параллельные вычисления

1. Введение

Возможности технологий виртуализации платформы x86_64 в последние годы растут высокими темпами. Технологии аппаратной и контейнерной виртуализации уже позволяют обеспечить производительность отдельных классов приложений в виртуальных средах, практически не отличимую от производительности на реальном оборудовании. В результате исследователи по всему миру начали изучать возможности и ограничения виртуализации высокопроизводительных вычислений (HPC, High Performance Computing).

Достоинства применения виртуализации в области высокопроизводительных вычислений широко обсуждаются [1], [2]. Среди них устойчивость к сбоям, совместимость и гибкость при работе с виртуальными машинами (ВМ). Также весьма интересна идея применения концепции облачных вычислений для создания высокопроизводительных облачных систем, предоставляющих масштабируемость, эффективность использования ресурсов и удобство доступа. Последние исследования показывают, что виртуализация HPC имеет смысл по крайней мере для отдельных классов приложений [1]. Несмотря на это, в настоящее время существует серьезная нехватка экспериментальных данных. Необходимо исследовать поведение различных приложений при запуске на различном оборудовании с использованием ряда систем виртуализации, для того, чтобы достоверно оценить ограничения и возможности существующих технологий. Современные многопроцессорные, многоядерные системы предъявляют новые требования к системам виртуализации, включая корректную эмуляцию архитектуры NUMA (Non-Uniform Memory Access, архитектура с неравномерным доступом к памяти) в гостевой системе.

Одной из целей данной работы было достичь максимально возможной производительности задач, запущенных в виртуальной среде, по сравнению с запуском на реальном оборудовании. Основной целью работы является оценка накладных расходов виртуализации в целом, а также выявление причин и минимизация этих расходов. Исследования проводятся с использованием системы виртуализации KVM (Kernel-based Virtual Machine) [3] и гипервизора Palacios [4], который изначально разрабатывался специально для высокопроизводительных систем. Необходимо отметить, что в ходе данной работы использовалась специально модифицированная версия гипервизора Palacios, поскольку оригинальная версия недоработана и не может быть запущена на используемом тестовом оборудовании. Сделанные изменения будут кратко описаны в тексте статьи.

Для достижения наилучшей производительности (относительно случая запуска на реальном оборудовании), виртуальным машинам выделяется максимально возможное количество ресурсов, включая все процессорные ядра. Помимо этого, виртуальным машинам предоставляется реальное коммуникационное устройство, с использованием технологий виртуализации ввода-вывода Intel VT-d в случае KVM и паравиртуализации в случае Palacios. Также в ВМ эмулируется архитектура NUMA в соответствии с реальной конфигурацией узлов кластера.

В качестве тестовых пакетов используются пакеты HPC Challenge [5] и NAS Parallel Benchmarks [6]. Данные бенчмарки достаточно широко распространены в области высокопроизводительных вычислений, а пакет HPCC используется для построения рейтинга суперкомпьютеров Top500. Тестирование производительности выполнялось на современном кластере, с использованием до 8 узлов, построенных на базе процессоров Intel Xeon (в

сумме до 96 процессорных ядер). Узлы кластера связаны высокоскоростной сетью Infiniband 40Гбит/сек. Стоит отметить, что многие исследования в области виртуализации HPC, представленные ранее, проводились на системах из одного узла, что не позволяло полноценно оценить накладные расходы, вызываемые виртуализацией.

В ходе выполнения данной работы достигнуты следующие результаты:

- Описаны базовые, но эффективные методы настройки KVM/QEMU для запуска HPC-приложений в виртуальной среде. Показана необходимость корректной эмуляции архитектуры NUMA в виртуальной среде в соответствии с реальной топологией системы.
- Проведена оценка накладных расходов виртуализации для современного HPC кластера, построенного на базе высокоскоростной сети Infiniband. Узлы кластера – двухпроцессорные 12-ядерные серверы архитектуры NUMA, используется до 8 узлов.
- Выполнено сравнение гипервизора Palacios, встроенного в основную ОС Kitten, и системы виртуализации KVM/QEMU, с основной ОС Linux. Полученные данные показывают, что KVM/QEMU дает более стабильные и предсказуемые результаты, в то время как Palacios имеет меньшие накладные расходы на “мелкозернистых” тестах, особенно при использовании большого числа вычислительных ядер.
- Проведено исследование полученных тестовых данных и накладных расходов, вызванных системой виртуализации. В частности, исследовано влияние гранулярности коммуникаций HPC-приложения и частоты прерываний на величину накладных расходов.

Текст данной работы организован следующим образом. В следующем разделе приводится обзор работ. В разделе 3 описаны исследуемые системы виртуализации и некоторые базовые методы по улучшению производительности ВМ. В разделе 4 описывается методика оценки производительности, приводятся результаты тестов и их анализ. Раздел 5 содержит заключение.

2. Обзор работ

Развитие технологий виртуализации открывает широкие перспективы в различных областях исследований. В частности, аппаратная виртуализация достаточно часто рассматривается в контексте информационной безопасности [7], а системы динамической бинарной трансляции нашли применение в области отладки. На базе симулятора QEMU, в ИСП РАН была реализована система детерминированного воспроизведения процесса выполнения программ в ВМ [8], позволяющая отлаживать трудновоспроизводимые ошибки как в приложениях, так и в коде ОС с использованием методов реверсивной отладки. В последнее время исследователи все чаще рассматривают перспективы применения виртуализации в области высокопроизводительных вычислений.

Подробный анализ существующих работ в области виртуализации HPC проведен в работе [1]. Авторы отмечают, что результаты экспериментов, доступные из статей, зачастую противоречат друг другу, и ставят задачу объективного сравнения существующих систем виртуализации, включая KVM, Xen [9], VirtualBox [10]. Для сравнения используются такие критерии как доступная функциональность, удобство использования и производительность. Авторы использовали тестовые пакеты HPCC и SPEC OpenMP; для теста HPL пакета HPCC накладные расходы виртуализации составили около 30% на 8-ядерной системе. Падение производительности в остальных тестах составило до 50% по сравнению с запуском на реальном оборудовании. Как показывают результаты наших исследований, такие значительные накладные расходы можно существенно сократить. Авторы статьи [1] отмечают, что производительность системы Xen нестабильна – различные запуски одного и того же приложения в одинаковых условиях дают результаты с существенным разбросом по производительности – что сильно снижает масштабируемость параллельных приложений. Наши предварительные эксперименты также показали, что гипервизор Xen не приспособлен для высокопроизводительных задач, по крайней мере, без специальной настройки. Необходимо также отметить, что авторы статьи выполняли все тесты на одной машине, что не позволило оценить падение производительности виртуальной среды при масштабировании числа узлов. Масштабируемость некоторых HPC приложений может также ухудшаться из-за дополнительного шума, вызываемого системой виртуализации. Шумом в контексте выполнения конкретного приложения называют влияние некоторых действий системы, не относящихся к приложению, на его работу. Среди таких действий – работа программ-демонов, обработчиков прерываний, других программ в системе.

Авторы работы [1] приходят к следующему выводу. Виртуализация применима, по крайней мере, к некоторым HPC-приложениям, а наиболее приспособленной системой виртуализации для таких задач является KVM/QEMU. С нашей точки зрения, система KVM становится одной из наиболее развитых систем виртуализации с открытым исходным кодом. По сравнению с Xen, KVM намного проще для установки и понимания, поскольку является частью исходного кода ядра Linux, а гипервизор Xen представлен в виде патчей. В целом, гипервизор KVM кажется более перспективным, в связи с чем в наших исследованиях было решено использовать именно его. При этом, нельзя утверждать, что производительность системы Xen хуже, чем KVM. Проблема в том, что каждая система виртуализации вместе с основной ОС должны быть тщательно настроены для запуска высокопроизводительных задач, однако для Xen такой настройки не проводилось. Также одной из причин исследования именно системы KVM/QEMU является наличие опыта оптимизации симулятора QEMU в ИСП РАН [11].

В работе [12] проводится сравнительный анализ производительности ввода-вывода систем виртуализации KVM, Xen, OpenVZ [13] и облачного сервиса фирмы Amazon “EC2 Cluster Compute Node”. Авторы отмечают, что вопросы виртуализации процессора в настоящее время тщательно изучены, и различные подходы к реализации гипервизоров позволяют достичь производительности, близкой к производительности реального процессора при запуске приложения, требовательного к вычислительной мощности. С другой стороны, накладные расходы виртуализации ввода-вывода изучены недостаточно подробно, и в работе авторы оценивают производительность дискового и сетевого ввода-вывода. Впервые оценивается эффективность работы адаптера Infiniband при его пробросе внутрь ВМ с использованием технологии Intel VT-d. Результаты пакета NAS Parallel Benchmarks полученные при запуске на системе из 4-х узлов на базе процессоров Intel Xeon E5520, до 32 процессорных ядер в сумме, показывают ухудшение производительности в худшем случае до 30 раз для KVM и до 50% для Xen, по сравнению с запуском на реальном оборудовании. Возможно, авторы использовали устаревшую версию KVM/QEMU, или была допущена ошибка в методике тестирования, поскольку результаты, полученные в нашей работе, намного более оптимистичны.

Производительность системы OpenVZ не тестировалась в рассматриваемой работе, поскольку в данной системе отсутствует поддержка технологии Infiniband в контейнерах, однако авторы утверждают, что OpenVZ дает производительность ввода-вывода, идентичную случаю без виртуализации. В целом, это утверждение правдоподобно, поскольку виртуализация уровня ОС (контейнерная виртуализация) требует намного меньше дополнительных ресурсов, чем полная виртуализация. С другой стороны, технологии аппаратной виртуализации продолжают развиваться, и с нашей точки зрения в ближайшее время производительность подсистемы ввода-вывода также улучшится.

Как утверждают авторы работы [4], основной причиной падения производительности устройства ввода-вывода, проброшенного внутрь ВМ, являются накладные расходы на обработку прерываний устройства. При каждом прерывании процессор должен передать управление от виртуальной машины гипервизору. Далее, гипервизор передает управление основной ОС, которая, в свою очередь, определяет, что прерывание принадлежит гипервизору. После этого, гипервизор производит “вброс” прерывания в гостевую систему. После окончания обработки прерывания гостевой системой, сигнал окончания обработки, подаваемый контроллеру прерываний, также обрабатывается гипервизором. Эта цепочка вызовов может замедлить доставку прерываний и серьезно увеличить задержку коммуникаций. В работе [14] данное предположение подтверждается. Авторы предложили систему ELI (ExitLess Interrupts), которая позволяет обрабатывать часть прерываний в гостевой системе без необходимости выходов в гипервизор и основную ОС. Результаты тестов показали, что система ELI может уменьшить накладные

расходы KVM/QEMU с 40% до 1-2% при работе ВМ на одном вычислительном ядре и использовании Netperf, Apache и Memcached в качестве тестовых приложений. В данном случае ухудшение производительности было связано с большим числом прерываний, генерируемых проброшенной внутрь ВМ сетевой картой Ethernet 10Гбит/сек. С использованием ELI, число выходов из ВМ в гипервизор сократилось примерно в 100 раз.

В работе [15] показано, что при запуске тестов NAS Parallel Benchmarks в виртуальных машинах, выполняющихся на системах архитектуры NUMA, производительность снижается в среднем на 50% по сравнению с запуском на реальном оборудовании. Авторы отмечают, что поддержка NUMA в существующих системах виртуализации не полностью реализована, вследствие чего не удается достигнуть локальности данных в гостевой системе. Более того, авторы считают, что даже реализация полной поддержки NUMA в ВМ не сильно поможет при работе нескольких ВМ на одном сервере. Для решения проблемы предлагается использовать паравиртуализацию путем добавления нового бит-транспортного протокола “bypass” в реализацию библиотеки OpenMPI. Этот протокол позволяет осуществлять передачу данных между ВМ с использованием общей памяти, доступной всем участвующим в обмене ВМ. При запуске по одной ВМ на процессорном сокете, авторы добились снижения накладных расходов до 5-7%. В нашей работе тесты выполнялись на 2-х процессорных 12-ядерных серверах архитектуры NUMA, однако, накладные расходы в целом меньше 50%. В случае виртуализации HPC-систем, гипервизор может эмулировать топологию NUMA в соответствии с реальной топологией сервера, что позволит избежать описанных в статье проблем, поскольку гостевая ОС будет осуществлять управление памятью с учетом архитектуры NUMA.

Исследования по виртуализации суперкомпьютеров выполнены в работе [4]. Использовался специально разработанный гипервизор Palacios, встроенный в основную ОС Kitten [16], предназначенную для работы на узлах суперкомпьютера. Гипервизор Palacios разрабатывается в рамках проекта V3VEE [17]. Авторы получили результаты для ряда HPC-приложений и тестовых пакетов, накладные расходы виртуализации сохраняются на уровне не более 5% с увеличением числа узлов до 4096. Однако необходимо отметить, что данные результаты были получены при запуске по одной 1-ядерной ВМ на каждом 4-х ядерном узле суперкомпьютера. Как показывает наше исследование, современные многопроцессорные системы требуют соответствующей поддержки со стороны гипервизора, включая корректную эмуляцию архитектуры NUMA. Авторы рассматриваемой статьи также исследовали различные методы страничной адресации, включая теневую страничную адресацию (shadow paging) и вложенную страничную адресацию (nested paging). Вложенная адресация дает лучшие результаты для гостевых систем с частым переключением таблиц страниц, например для ОС Linux. В

нашей работе мы также использовали вложенную адресацию, поскольку в качестве гостевой ОС используется Linux.

Авторы гипервизора Palacios также отмечают проблему влияния шума ОС на работу параллельных приложений. Влияние шума широко изучается [18], [19]. В целом, влияние шума сильно зависит от ряда факторов, в том числе от соотношения объема вычислений и коммуникаций приложения, гранулярности коммуникаций (размер и частота посылок), характеристик сетки процессов. Для ряда приложений, даже минимальный уровень шума может привести к значительному ухудшению производительности и масштабируемости. Таким образом, при виртуализации высокопроизводительной системы, особенно с большим числом узлов, необходимо учитывать дополнительный шум от гипервизора и основной ОС. С этой точки зрения, ОС Kitten имеет преимущество перед Linux, как основная ОС, так как данная ОС специально разработана для запуска HPC-приложений на системах с большим числом узлов.

3. Рассматриваемые системы виртуализации

Наиболее распространенные системы виртуализации в целом обладают одинаковыми возможностями. Такие системы используют доступные на сервере технологии аппаратной виртуализации. Среди систем с открытым исходным кодом, наиболее популярны KVM и Xen. Как уже отмечалось ранее, в данной работе исследуется гипервизор KVM, поскольку, на наш взгляд, KVM более перспективен и весьма удобен в использовании. Для установки и исследования Xen требуется больше времени, так как существует несколько различных версий данного гипервизора. Также рассматривается гипервизор Palacios, который специально разработан для исследования архитектуры компьютеров и использования в области высокопроизводительных вычислений.

3.1. Система виртуализации KVM/QEMU

Гипервизор KVM состоит из нескольких модулей ядра Linux и управляется посредством файла устройства, доступ к которому осуществляется с использованием системного вызова ioctl. KVM реализует виртуальный процессор и систему памяти, остальные компоненты виртуальной машины реализованы эмулятором QEMU. QEMU управляет ресурсами и предоставляет виртуальные устройства. С помощью QEMU можно запускать ВМ с использованием технологий аппаратной виртуализации (для этого используется KVM) либо бинарной трансляции (генератор кода встроен в QEMU).

KVM/QEMU вместе с основной ОС Linux позволяют назначать реальные устройства отдельным ВМ (пробрасывать устройства внутрь ВМ). Для данной возможности необходимо наличие специальной аппаратуры – устройства управления памятью ввода-вывода (IOMMU, IO Memory Management Unit). В

случае платформы производства Intel технология называется Intel VT-d, у фирмы AMD – AMD IOMMU. Основной задачей IOMMU является отображение адресного пространства пробрасываемого устройства в адресное пространство физической памяти гостевой системы. При отсутствии такого устройства, гостевая система должна быть осведомлена о наличии гипервизора и положении гостевой физической памяти в реальной физической памяти, что позволит выполнять запросы на прямой доступ к памяти (DMA, Direct Memory Access) корректно.

Для возможности использования высокоскоростной сети в виртуальной машине, коммуникационный адаптер Infiniband был проброшен внутрь ВМ с использованием Intel VT-d. Необходимо отметить, что назначенные ВМ устройства могут иметь более низкую производительность, чем если бы они работали в основной ОС. Как было описано выше, это может быть вызвано накладными расходами по обработке прерываний. Более подробно данная проблема рассматривается в разделе 4.

3.2. Использование больших страниц

Система виртуальной памяти архитектуры x86_64 поддерживает несколько размеров страниц при страничной адресации, среди них 4Кб, 2Мб и 1Гб страницы. Ядро Linux по умолчанию использует 4Кб страницы, страницы большего размера могут быть задействованы с использованием системы HugeTLB. С точки зрения виртуализации памяти, большие страницы могут быть использованы для уменьшения числа доступов к памяти, необходимого для трансляции гостевого физического адреса в реальный физический адрес, а также для уменьшения числа промахов TLB (Translation Lookaside Buffer, буфер трансляции страниц).

Ядро Linux предоставляет два способа использования HugeTLB: Transparent Hugepages и HugeTLBfs. Механизм Transparent Hugepages позволяет ядру выделять память большими страницами автоматически, если запрошенный объем памяти кратен размеру большой страницы, при этом приложение не требуется модифицировать. В отличие от Transparent Hugepages, механизм HugeTLBfs использует заранее зарезервированные группы больших страниц. HugeTLBfs представляет собой специальную файловую систему, для хранения файлов в которой используются большие страницы. Программа может отобразить такие файлы в память (используя системный вызов mmap) и для этого отображения ядро Linux будет использовать большие страницы.

Используемая нами версия QEMU (1.0) выделяет память для гостевой системы посредством вызова posix_memalign с 2Мб выравниванием, поэтому механизм Transparent Hugepages должен работать по умолчанию.

3.3. Предоставление ВМ реальной топологии NUMA

Как показывают результаты тестов, запуск виртуальной SMP системы на реальной системе архитектуры NUMA ведет к существенным накладным

расходам. Каждый узел в топологии NUMA имеет свой набор процессорных ядер и участков физической памяти, и доступ процессора одного узла к памяти другого узла требует существенно больше тактов, чем доступ к локальной памяти узла. Эмулятор QEMU имеет поддержку архитектуры NUMA в ВМ, однако эмулируемая топология никак не связана с реальной топологией системы. Для решения этой проблемы, были внесены изменения в исходный код QEMU. Прежде всего, была осуществлена "привязка" виртуальных процессоров (ядер) к отдельным физическим ядрам процессора, что позволило предотвратить перемещение нитей QEMU между ядрами процессоров. Далее, был использован системный вызов mbind для выделения отрезков памяти гостевой системы на соответствующих узлах реального сервера. Было также выявлено, что вызов mbind препятствует работе механизма Transparent Hugepages, поэтому память выделяется посредством HugeTLBfs с использованием ключа командной строки QEMU -mempath. Механизм HugeTLBfs имеет некоторые проблемы с безопасностью и масштабируемостью [20], однако в данном случае этим можно пренебречь.

3.4. Гипервизор Palacios

Гипервизор Palacios разрабатывался с целью эффективной виртуализации высокопроизводительных систем. Версия 1.0 вышла в 2008 году, последняя версия 1.3 вышла в декабре 2011 года. Отличительной особенностью Palacios является его встраиваемость, реализованная путем выделения набора унифицированных интерфейсов с основной ОС. Код Palacios может быть встроен в различные системы путем реализации этих интерфейсов. Изначально гипервизор Palacios был встроен в специализированную ОС Kitten, в последней версии 1.3 также реализован модуль для встраивания в Linux. Другой отличительной особенностью Palacios является его конфигурируемость, то есть возможность на этапе компиляции включить или исключить различные компоненты гипервизора. Эта особенность позволяет легко создавать версии гипервизора, специально "подогнанные" под конкретные нужды. На наш взгляд, система Palacios является весьма перспективной. Основная ОС Kitten дает важное преимущество – эта ОС генерирует намного меньше шума по сравнению с Linux, и это преимущество особенно интересно с точки зрения проводимых исследований. Часть кода ОС Kitten заимствована из ядра Linux, однако критические с точки зрения производительности подсистемы разработаны с учетом особенностей высокопроизводительных систем.

Palacios поддерживает вложенную страничную адресацию и несколько методов теневой адресации. Для экспериментов использовалась вложенная адресация, так как данный метод лучше подходит для гостевой ОС Linux [4].

3.5. Метод проброса устройств

Авторы Palacios отмечают необходимость проброса реального коммуникационного устройства внутрь ВМ для достижения сопоставимого с

реальным оборудованием уровня производительности. Для проброса устройств используется специальный паравиртуальный интерфейс (все взаимодействие между гипервизором и гостевой системой в рамках Palacios называется "Симбиотическая виртуализация"). Память для гостевой системы выделяется единым непрерывным отрезком, после чего гостевая система запрашивает у гипервизора смещение памяти ВМ в реальной физической памяти. С использованием этого смещения, гостевая ОС может правильно формировать DMA-запросы для проброшенных в ВМ устройств путем прибавления смещения к целевому адресу. Изменения гостевой ОС, необходимые для поддержки данного механизма, минимальны. Был написан патч для ядра Linux версии 2.6.18, который позже был портирован на более новые версии ядра.

Поскольку ОС Kitten практически не содержит драйверов устройств, помимо адаптера Infiniband внутрь ВМ были также проброшены SATA-контроллер и сетевая карта Ethernet.

3.6. Возникшие трудности с гипервизором Palacios

Во время экспериментов с гипервизором Palacios был обнаружен ряд трудностей, вызванных "незрелостью" системы. Трудности включали в себя неполную поддержку проброса устройств (элемент Capabilities конфигурационного пространства PCI был недоступен в ВМ, из-за чего были недоступны прерывания типов MSI и MSI-X), неполная реализация поддержки технологии Intel VT-x, максимальный объем памяти ВМ примерно 3.5Гб, отсутствие поддержки архитектуры NUMA в ВМ и ряд ошибок в исходном коде. Была создана локальная ветка репозитория Palacios, в которой была реализована недостающая функциональность. Часть созданного кода была передана разработчикам Palacios в виде патчей, некоторые из которых доступны на официальном сайте проекта V3VEE. Часть патчей включены в основную ветку разработки Palacios.

Одна серьезная проблема совсем не была решена –проблема с реализацией системных таймеров в ВМ. Время в ВМ гипервизора Palacios идет неточно, за сутки достигается сдвиг часов примерно на 30-40 минут. Для того чтобы контролировать этот эффект во время запуска тестов, использовалась синхронизация по протоколу NTP и проверялось значение смещения времени.

4. Оценка и анализ производительности

За счет использования двух популярных тестовых пакетов HPC Challenge и NAS Parallel Benchmarks, оценивается достаточно широкий класс HPC-приложений. В следующих подразделах описываются экспериментальная среда, методология тестирования и результаты. В конце раздела анализируются причины накладных расходов в различных случаях.

4.1. Экспериментальная среда

В качестве стенда использовался кластер фирмы HP. Кластер состоит из 8 блейдов HP ProLiant BL2x220c G7, каждый блейд комбинирует в себе по два вычислительных узла. Для тестов использовалось до 8 узлов одновременно. Каждый узел содержит по 2 процессора Intel Xeon X5670 с частотой 2.93 Ггц (по 6 ядер на процессор), а также 24Гб ОЗУ. Технология Hyperthreading деактивирована на всех узлах. Узлы взаимодействуют посредством сервисной сети Ethernet 1Гбит/сек. и вычислительной сети Infiniband 40Гбит/сек.

Для тестов на реальном оборудовании и в качестве гостевой ОС использовалась система Linux CentOS 6.0 с ядром версии 2.6.32-71.29.1.el6.x86_64. Ядро ОС модифицировано для поддержки проброса устройств Palacios. Версия библиотеки MPI – OpenMPI 1.4.3, компилятора GCC – 4.4.4. ПО Infiniband – Mellanox OFED 1.5.3-1.0.0-rhel6-x86_64.

Основная ОС для KVM/QEMU – Linux CentOS 6.2, версия ядра 2.6.32-220.2.1.el6.x86_64. Версия QEMU – qemu-kvm-1.0 с модификациями, описанными ранее. Версия Palacios – 1.2 с изменениями, описанными выше. Версия ОС Kitten – 1.2.0 с изменениями, необходимыми для корректного проброса устройств в ВМ.

Гостевая система сконфигурирована с 16 Гб ОЗУ, 12-ю процессорными ядрами, архитектуры SMP или NUMA. Контроллер Infiniband проброшен в ВМ, как описано ранее. Для Palacios также пробрасывается сетевая карта и SATA-контроллер.

4.1.1.Бенчмарки

Для оценки производительности используются тестовые пакеты HPC Challenge и NAS Parallel Benchmarks. Каждый тестовый пакет выполнялся на 2, 4 и 8 узлах.

Версия NPB – 3.3.1 MPI. Из данного пакета используются тесты IS, EP, CG, MG, FT, BT и LU, класс задачи C. Для тестов IS, CG, FT, MG запускается по 8 процессов на узел (число процессов должно быть степенью двойки). Для EP и LU – по 12 процессов на узел, для BT – по 8, 9 и 8 процессов на узел соответственно (суммарное число процессов должно быть квадратом). Также тесты NPB выполнялись на одном узле, по 9 процессов для BT, по 12 для EP и LU, по 8 для остальных тестов. Результаты тестов NPB приведены к унифицированному виду и выражаются в секундах.

В тесте IS (Integer Sorting) выполняется цифровая сортировка, большой объем коммуникаций и случайных доступов к памяти. EP (Embarrassingly Parallel) – генератор значений Гауссовых случайных величин. CG – метод сопряженных градиентов нахождения локального минимума, с нерегулярным характером доступов к памяти и коммуникаций. В тесте MG аппроксимируется решение дискретного уравнения Пуассона, тест характеризуется интенсивной работой с памятью и коммуникациями. FT – дискретное преобразование Фурье,

использует коммуникации типа “all-to-all”. Тесты BT и LU – решатели систем дифференциальных уравнений в частных производных с использованием различных методов.

Используемая версия пакета HPCC – 1.4.1, библиотеки ATLAS – 3.8.4. Тесты пакета HPCC выполняются с запуском по 12 процессов на узел. Оцениваются результаты тестов STREAM, RandomAccess, HPL. Тест PTRANS выполняется слишком быстро (0.1-0.5 секунды) и принадлежит к тому же классу задач, что и STREAM. Также из рассмотрения исключены тесты DGEMM, Bandwidth/Latency, FFT из-за ограничений по объему материала. Исследуемые тесты пакета HPCC имеют следующие характеристики:

- STREAM. Оценка устойчивой пропускной способности памяти для векторной операции (копирование, умножение на константу, сложение, комбинация), в данной работе исследовались результаты комбинированной операции (в Гб/сек.). Оцениваются результаты для версий Single (выполняется ровно один процесс) и EP (Embarrassingly Parallel) – все доступные ядра выполняют одну и ту же задачу независимо. Запуск версий Single и EP осуществляется только на одном узле.
- RandomAccess. Оценка скорости обновлений памяти по случайным адресам в миллиардах обновлений в секунду (GUP/s). Известно, что данный тест работает в виртуальной среде достаточно плохо из-за высокой нагрузки на буфер трансляции (TLB). Используются версии Single, EP и MPI.
- HPL (High Performance Linpack). Решение системы линейных уравнений, результат – скорость решения в GFlops. Параметры теста: размер матрицы – 30000, размер блока 150, вычислительная сетка в виде квадрата, когда это возможно.

Известно, что время в виртуальных машинах идет неточно. Гипервизор и основная ОС являются дополнительными источниками шума и вызывают неточности в таймерах гостевой ОС. В результате, время выполнения одного теста при многократных запусках в ВМ имеет больший разброс по сравнению с запусками на реальном оборудовании. Для того чтобы результаты проводимых экспериментов были более точными, большая часть запусков NPB выполнялась по 50 раз, HPCC – по 20 раз. В качестве результата серии запусков, вычисляется оценочное математическое ожидание (среднее арифметическое) и доверительный интервал с вероятностью 97%, вычисленный с использованием распределения Стьюдента. Доверительный интервал позволяет оценивать достоверность полученных данных. Хотя последовательные запуски тестов не могут считаться независимыми экспериментами, доверительный интервал может помочь оценить влияние различных источников ошибки в измерении времени и удостовериться, что выполнено достаточное число запусков.

4.2. Результаты тестов

Первоначальные тесты проводились с конфигурацией QEMU “по умолчанию”, по 12 виртуальных ядер на ВМ. Первичные результаты для пакета NPB представлены на рис. 1. “KVM” обозначает конфигурацию “по умолчанию”, тонкие черные столбцы – доверительный интервал для соответствующих средних значений. Данные изображены относительно случая Native (запуск на реальном оборудовании), которому соответствует среднее значение 0 в каждой группе столбцов. Подпись под каждой группой столбцов состоит из названия теста и числа процессов, на котором выполнялся тест. Как показывает рисунок, в конфигурации “KVM” накладные расходы изменяются от 2-4% на тесте EP до 32% на тесте MG, 64 процесса.

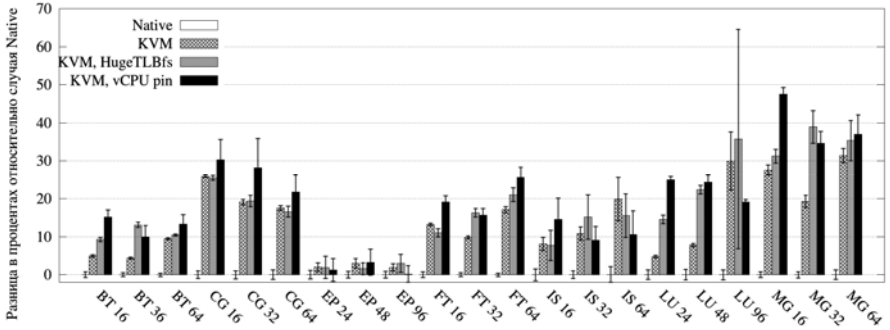

Рис. 1. Первичные результаты KVM/QEMU на тестах пакета NPB.

Далее было предположено, что миграция нитей QEMU между ядрами процессоров может негативно сказываться на производительности. Чтобы предотвратить миграцию нитей, была выполнена привязка этих нитей к соответствующим ядрам процессоров, по одному виртуальному ядру на реальное. Также было проверено, что механизм HugeTLBfs дает такие же накладные расходы, как и Transparent Hugepages. Результаты для описанных модификаций представлены также на рис. 1, столбцы “KVM, vCPU pin”, “KVM, HugeTLBfs”. Данные две группы тестов были выполнены только по 10 раз из-за нехватки времени. Как показывает рисунок, после изменений накладные расходы остались такими же и даже увеличились в ряде случаев. Случай привязки нитей дает более нестабильные результаты для тестов BT, CG, FT. Это может быть вызвано нелокальными доступами к памяти. Также интересен случай LU, 96 процессов, где виден весьма нестабильный результат. Предположительно, влияние оказал неизвестный неучтенный фактор.

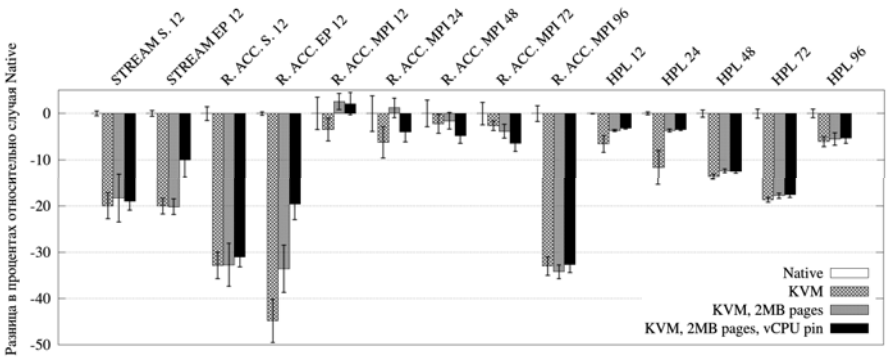

Рис. 2. Первичные результаты KVM/QEMU на тестах пакета HPCC.

Первичные результаты тестов пакета HPCC представлены на рис. 2. В данном случае, оценивались конфигурации с использованием HugeTLBfs и с привязкой нитей вместе с HugeTLBfs. Сокращение “R. ACC.” означает тест RandomAccess, “S.” – Single-версия теста. Разница в процентах в данном случае отрицательна, поскольку измеряемые величины пропорциональны производительности. Для тестов Stream версий Single, EP накладные расходы около 20% кроме случая привязки нитей, где для версии EP только 10% расходов. Тест RandomAccess показывает заметное падение производительности для версий Single и EP – от 20 до 45%, опять же, случай привязки нитей дает лучший результат для версии EP, 20% расходов. Результаты RandomAccess MPI нестабильны и расходы менее 7% при запуске до 72 процессов, однако для 96 процессов ситуация кардинально меняется – накладные расходы устойчивы на уровне 34%. Возможно, на 96 процессах узким местом становятся коммуникации, однако подробно данный вопрос не исследовался.

Наконец, было выдвинуто предположение о том, что основной причиной падения производительности является несоответствие виртуальной SMP архитектуры реальной архитектуре NUMA. По умолчанию, QEMU эмулирует SMP систему, что ведет к неучтенным гостевой ОС накладным расходам при нелокальном доступе. Следующим шагом было предоставление гостевой системе архитектуры NUMA так, чтобы топология соответствовала реальной структуре системы. Такая возможность была реализована на основе поддержки эмуляции NUMA в эмуляторе QEMU, как это описано в предыдущем разделе.

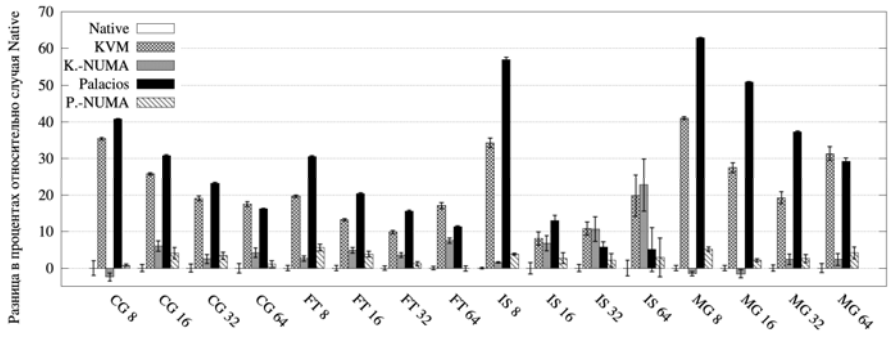

Рис. 3. Результаты KVM/QEMU и Palacios для тестов CG, FT, MG, IS пакета NPB.

Для системы Kitten + Palacios тестировались две конфигурации: BM архитектуры SMP и NUMA. В обоих случаях использовалась вложенная страничная адресация с 2Мб страницами. Результаты сравнения NUMA и SMP на тестах NPB для систем KVM/QEMU и Palacios представлены на рис. 3, 4, результаты тестов HPCC – на рис. 5.

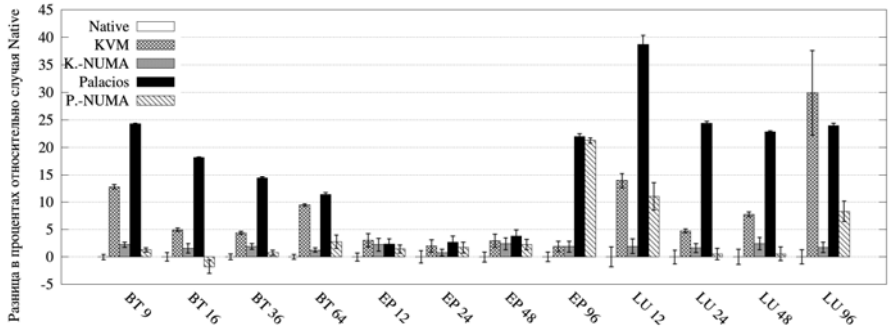

Рис. 4. Результаты KVM/QEMU и Palacios для тестов BT, EP, LU пакета NPB.

Первый достаточно очевидный вывод, который можно сделать из полученных данных – влияние корректной эмуляции NUMA на производительность весьма значительно. При запуске BM архитектуры SMP, Palacios и KVM показывают падение производительности более чем 60% и 40% соответственно на тесте MG 8, при эмуляции NUMA данные расходы практически исчезают. То же самое верно для большинства других тестов. Интересен также тот факт, что для многих результатов виртуальной архитектуры SMP, накладные расходы уменьшаются при увеличении числа процессов, причем разница для одного

теста может быть значительной – 30% и более, см. тесты IS и MG. В то же время, для результатов эмуляции архитектуры NUMA на многих тестах накладные расходы колеблются незначительно.

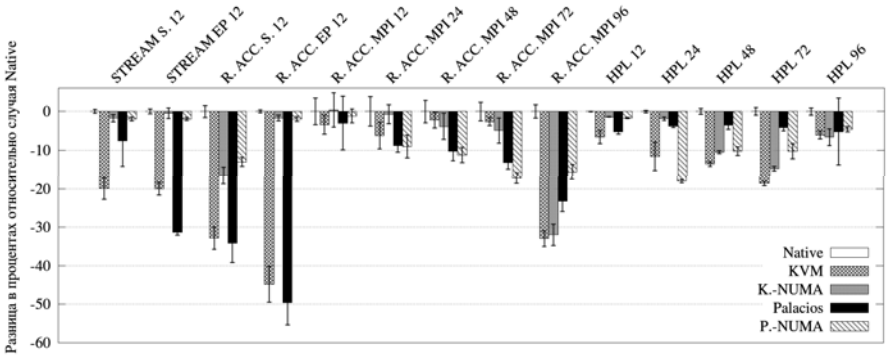

Рис. 5. Результаты KVM/QEMU и Palacios для тестов пакета HPCC.

Для тестов HPCC, эмуляция NUMA улучшает производительность на тестах STREAM, RandomAccess версий Single и EP. Для теста RandomAccess EP, эмуляция NUMA позволяет практически полностью устранить накладные расходы 45-50% случая без NUMA. Для теста RandomAccess MPI, ситуация не столь однозначна. KVM показывает лучшие результаты при запуске на 24, 48, 72 процессах, Palacios – при запуске на 96 процессах.

Можно сделать следующий вывод: реализация Palacios имеет некоторые недостатки, что показывает сравнение с результатами KVM без поддержки NUMA для практически всех тестов пакета NPB и теста RandomAccess для 24, 48, 72 процессов. Наибольшая разница между результатами двух гипервизоров около 24% на тесте LU, 12 процессов. Также Palacios дает “аномальные” результаты для теста EP на 96 процессах, учитывая тот факт, что данный тест оценивает производительность процессора. Результаты теста HPL также вызывают вопросы – на 24, 48 и 72 процессах Palacios без эмуляции NUMA показывает наилучший результат, однако с эмуляцией накладные расходы резко увеличиваются. Основная гипотеза, объясняющая разницу в производительности между KVM и Palacios в случае эмуляции архитектуры SMP, заключается в следующем. Ядро Linux имеет преимущество, автоматически управляя памятью и процессами в случае использования конфигурации QEMU “по умолчанию”, в то время как для гипервизора Palacios отображение виртуальных ядер в реальные изначально фиксировано. Как принудительное назначение ядер нитям, так и использование механизма HugeTLBfs ведут к ухудшению производительности, поскольку механизмы управления памятью и процессами ОС Linux теряют часть своих функций.

В нескольких случаях результаты в виртуальной среде оказываются лучше результатов на реальном оборудовании. Например, тесты CG 8, MG 8, MG 16, где KVM с эмуляцией NUMA дает лучшие результаты, чем в случае Native. Аналогичная ситуация с гипервизором Palacios на тесте BT 16. Вероятно, причиной такого поведения является наличие неких неучтенных устойчивых факторов, повлиявших на результаты данных тестов, однако требуется более тщательное исследование этого феномена.

Наконец, результаты тестов CG, FT, IS на 64 процессах и RandomAccess MPI на 96 процессах показывают преимущество связки Kitten + Palacios над Linux + KVM/QEMU в случае эмуляции архитектуры NUMA. Преимущество варьируется от 3.5 до 21%. Вероятно, такое поведение связано с уменьшенным шумом ОС Kitten по сравнению с Linux, особенно для теста IS. Также исследовалась гранулярность коммуникаций в тестах NPB путем оценки частоты прерываний, см. следующий подраздел.

Если сравнить результаты с архитектурой NUMA систем KVM/QEMU и Palacios и использовать порог в 3%, KVM будет лучше в 11 случаях, Palacios в 9 случаях, результаты одинаковы в пределах порога в 22 случаях. При использовании порога в 5%, KVM лучше в 8 случаях, Palacios – только в 4. В целом, KVM предоставляет более стабильные и предсказуемые результаты, в то время как Palacios выигрывает на “мелкозернистых” тестах типа IS (особенно при большом числе процессов). В то же время, Palacios имеет аномалии в производительности на некоторых тестах.

4.2.1 Влияние частоты прерываний

Согласно статье [14], виртуализация прерываний может привести к существенному падению производительности реального устройства, назначенного ВМ. Для проверки данной теории, были проведены замеры частоты прерываний коммуникационного устройства Infiniband во время выполнения тестов NPB. Результаты представлены на рис. 6. На данном рисунке сгруппированы два набора данных: красным цветом отмечено падение производительности в процентах, зеленым – среднее число прерываний в секунду (IPS, Interrupts per second) для устройства Infiniband.

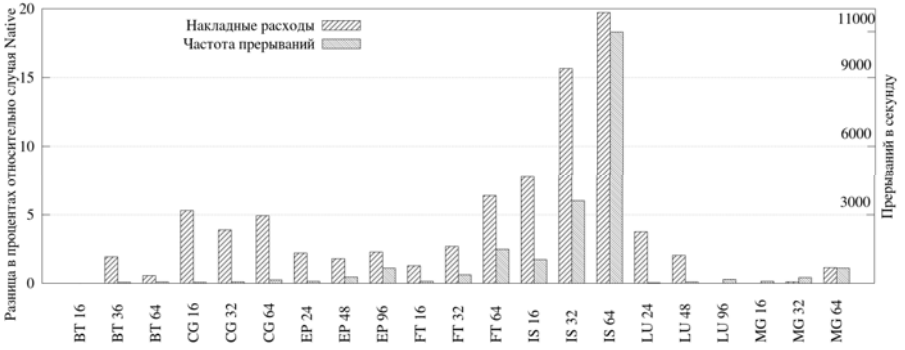

Рис. 6. Сравнение накладных расходов на тестах пакета NPB и частоты прерываний для случая эмуляции архитектуры NUMA, система виртуализации KVM/QEMU.

Максимальная частота прерываний равна около 11000 IPS, что означает по крайней мере 11000 выходов в гипервизор в секунду, что должно привести к заметному снижению производительности. Хотя в оригинальной работе [14] частота прерываний была выше (40000-60000 прерываний в секунду), вероятно, в случае теста IS, по крайней мере часть расходов связана с виртуализацией прерываний. Чтобы доказать это предположение, необходимо провести тесты с аналогом системы ExitLess Interrupts.

Высокая частота прерываний для теста IS на 32, 64 процессах также отражает гранулярность коммуникаций в данном тесте. В данном случае, коммуникации имеют “мелкозернистый” характер, что объясняет большое число прерываний. На рис. 3 видно, что накладные расходы гипервизора Palacios меньше на 20% чем расходы KVM, что может быть связано с пониженным уровнем шума основной ОС Kitten.

5. Заключение

Основным вкладом данной работы является демонстрация необходимости корректной эмуляции архитектуры NUMA в соответствии с реальной конфигурацией при запуске HPC-приложений в виртуальных машинах на серверах архитектуры NUMA. Исследования проведены с использованием систем виртуализации KVM/QEMU и Palacios. KVM/QEMU широко используется в индустрии, в то время как Palacios – гипервизор, специально разрабатываемый для виртуализации HPC. В исходный код QEMU и Palacios были внесены необходимые изменения для предоставления гостевой системе реальной топологии NUMA. С использованием эмуляции NUMA, накладные расходы виртуализации были снижены с 10-60% до 1-5% на многих тестах из пакетов HPCC и NAS Parallel Benchmarks. Единственные тесты, на которых

накладные расходы выше – RandomAccess и HPL из пакета HPCC, с потерями производительности до 30% и 15% соответственно.

Все тесты выполнялись на современном высокопроизводительном кластере, объединенном сетью Infiniband. Для вычислений использовалось до 96 процессорных ядер. Полученные результаты позволили оценить накладные расходы, вызванные виртуализацией как процессора, так и коммуникационной среды. Также было проведено исследование связи частоты прерываний коммуникационного устройства и накладных расходов виртуализации; для адаптера Infiniband, измерена максимальная частота до 11000 прерываний в секунду на тесте IS пакета NPB при запуске на 64 процессах, накладные расходы при этом достигали 20% для системы виртуализации KVM/QEMU. Нельзя утверждать, что данные расходы вызваны только высокой частотой прерываний, однако, такая частота может свидетельствовать о “мелкозернистом” характере коммуникаций теста IS.

Сравнивая системы виртуализации KVM/QEMU и Palacios, можно сделать вывод, что в среднем результаты тестов с эмуляцией NUMA одинаковы для обеих систем, причем KVM дает более стабильные и предсказуемые результаты, в то время как Palacios показывает лучшие результаты на “мелкозернистых” тестах при большом числе задействованных процессов. В то же время, Palacios имеет аномалии в производительности на некоторых тестах. Шум основной ОС имеет большое значение для масштабируемости “мелкозернистых” тестов, и в этом аспекте основная ОС Kitten имеет преимущество по сравнению с ОС Linux, что дает лучшую проихводительность и масштабируемость для тестов, выполняющихся в ВМ гипервизора Palacios. С нашей точки зрения, объем шума, генерируемого основной ОС и системой виртуализации, является критически важным параметром для успешной виртуализации многопроцессорных, многоядерных высокопроизводительных систем с достаточно большим числом вычислительных узлов.

В целом, полученные результаты свидетельствуют о целесообразности применения виртуализации для большого класса высокопроизводительных приложений. Основными “узкими местами” систем виртуализации являются уменьшенная производительность системы памяти (критично только для узкого класса задач, в том числе RandomAccess), расходы при виртуализации устройств, а также повышенный уровень “шума”, источником которого становятся основная ОС и гипервизор.

Список литературы

[1] A. J. Younge, R. Henschel, J. Brown, G. von Laszewski, J. Qiu, и G. C. Fox. Analysis of Virtualization Technologies for High Performance Computing Environments. In The 4th International Conference on Cloud Computing (IEEE CLOUD 2011), июль 2011.

[2] A. Gavrilovska, S. Kumar, H. Raj, K. Schwan, V. Gupta, R. Nathuji, R. Niranjan, A. Ranadive, и P. Saraiya. Abstract High-Performance Hypervisor Architectures:

Virtualization in HPC Systems. In 1st Workshop on System-level Virtualization for High Performance Computing (HPCVirt), in conjunction with EuroSys 2007, 2007.

[3] A. Kivity, Y. Kamay, D. Laor, U. Lublin, и A. Liguori. KVM: the Linux virtual machine monitor. In OLS ’07: The 2007 Ottawa Linux Symposium, с. 225–230, июль 2007.

[4] J. R. Lange, K. Pedretti, P. Dinda, P. G. Bridges, C. Bae, P. Soltero, и A. Merritt. Minimal-overhead virtualization of a large scale supercomputer. In Proceedings of the 7th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, VEE ’11, с. 169–180, 2011.

[5] P. R. Luszczek, D. H. Bailey, J. J. Dongarra, J. Kepner, R. F. Lucas, R. Rabenseifner, и D. Takahashi. The HPC Challenge (HPCC) benchmark suite. In Proceedings of the 2006 ACM/IEEE conference on Supercomputing, SC ’06, 2006.

[6] D. Bailey, T. Harris, W. Saphir, R. van der Wijngaart, A. Woo, и M. Yarrow. The NAS parallel benchmarks 2.0. Technical Report NAS-95-020, NASA Ames Research Center. Декабрь 1995.

[7] Д. В. Силаков. Использование аппаратной виртуализации в контексте информационной безопасности. Труды Института системного программирования РАН, том 20, с. 25-36, 2011.

[8] П. Довгалюк. Детерминированное воспроизведение процесса выполнения программ в виртуальной машине. Труды Института системного программирования РАН, том 21, с. 123-132, 2011.

[9] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, и A. Warfield. Xen and the art of virtualization. SIGOPS Oper. Syst. Rev., 37, с. 164–177, октябрь 2003.

[10] J. Watson. VirtualBox: bits and bytes masquerading as machines. Linux Journal, февраль 2008.

[11] К. Батузов, А. Меркулов. Оптимизация динамической двоичной трансляции. Труды Института системного программирования РАН, том 20, с. 37-50, 2011.

[12] N. Regola и J.-C. Ducom. Recommendations for virtualization technologies in high performance computing. In Proceedings of the 2010 IEEE Second International Conference on Cloud Computing Technology and Science, CLOUDCOM ’10, с. 409–416, 2010.

[13] OpenVZ: container-based virtualization for Linux, http://openvz.org/.

[14] A. Gordon, N. Amit, N. Har’El, M. Ben-Yehuda, A. Landau, A. Schuster, и D. Tsafrir. ELI: Bare-Metal Performance for I/O Virtualization. In Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2012), 2012.

[15] K. Z. Ibrahim, S. Hofmeyr, и C. Iancu. Characterizing the performance of parallel applications on multi-socket virtual machines. In Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGRID ’11, с. 1–12, 2011.

[16] J. Lange, K. Pedretti, T. Hudson, P. Dinda, Z. Cui, L. Xia, P. Bridges, A. Gocke, S. Jaconette, M. Levenhagen, и R. Brightwell. Palacios and Kitten: New high performance operating systems for scalable virtualized and native supercomputing. In 2010 IEEE International Symposium on Parallel Distributed Processing (IPDPS), с. 1 –12, апрель 2010.

[17] V3VEE: An Open Source Virtual Machine Monitor Framework For Modern Architectures, http://v3vee.org/.

[18] K. Ferreira, P. Bridges, и R. Brightwell. Characterizing application sensitivity to OS interference using kernel-level noise injection. In International Conference for High

Performance Computing, Networking, Storage and Analysis, 2008, с. 1–12, ноябрь 2008.
[19] F. Petrini, D. Kerbyson, и S. Pakin. The Case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q. In 2003 ACM/IEEE Conference on Supercomputing, с. 55, ноябрь 2003.
[20] A. Arcangeli. Transparent Hugepage Support, http://www.linux-kvm.org/wiki/images/9/9e/2010-forum-thp.pdf. KVM Forum 2010.