Разработка и реализация облачной системы для решения высокопроизводительных задач[1]

А.О. Кудрявцев, В.К. Кошелев, А.О. Избышев, И.А. Дудина, Ш.Ф. Курмангалеев, А.И. Аветисян, В.П. Иванников, В.Е. Велихов, Е.А. Рябинкин

Аннотация. В данной работе описаны основные проблемы, возникающие при переносе высокопроизводительных вычислений в облако. Рассматривается подход к организации высокопроизводительного облачного сервиса с использованием виртуализации. Описана архитектура разработанной системы «виртуальный суперкомпьютер», основанной на облачной платформе OpenStack и системе виртуализации KVM/QEMU. Компоненты системы ВСК доработаны для учета специфики высокопроизводительных вычислений, в частности, выполнена доводка и настройка системы виртуализации, что позволило достичь уровня накладных расходов не более 10% при использовании по крайней мере 1024 процессорных ядер.

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

1 Введение

При решении различных вычислительных задач, как правило, возникают колебания в объеме задействованных ресурсов, которые связаны с множеством факторов, начиная от характера решаемых задач и заканчивая временем года. В результате возникают ситуации, когда вычислительные ресурсы простаивают, либо ощущается нехватка ресурсов для решения задач пользователей, и, как следствие, возникают убытки от простоя оборудования или срыва сроков.

[1] Работа выполнена при финансовой поддержке Минобрнауки России по государственному контракту от 15.06.2012 г. № 07.524.11.4018 в рамках ФЦП «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2007-2013 годы»

С развитием сети Интернет появилась возможность удаленной работы с вычислительными ресурсами. В результате развития этой идеи была предложена концепция облачных вычислений, подразумевающая удаленную работу с ресурсом, причем задействованный объем ресурса может варьироваться в зависимости от потребностей. Такой ресурс предоставляется облачными провайдерами в многопользовательском режиме, оплата происходит по факту использованного объема ресурсов.

В последние годы популярна концепция облачного сервиса уровня инфраструктуры [1], позволяющая пользователю получать доступ к серверам, сетям передачи данных, хранилищам. Такой сервис позволяет организациям полностью отказаться от использования собственных вычислительных ресурсов. Экономия достигается за счет отсутствия необходимости покупки, установки, обслуживания собственного оборудования.

Ключевой особенностью облачных систем уровня инфраструктуры является использование виртуализации для реализации облачных серверов. Виртуализация предоставляет изолированное окружение, или «виртуальную машину» (ВМ), в которой возможно выполнение всего стека ПО сервера, включая операционную систему. Виртуальные машины могут быть консолидированы – запущены на одном сервере, что позволяет максимально задействовать ресурсы конкретных серверов. В связи с этим, такие облачные системы позволяют достигать существенной экономии средств за счет обеспечения высокой загрузки серверов. Виртуализация на платформе x86 изначально была реализована программным путем, в настоящее время используются аппаратные расширения процессора, позволяющие существенно повысить производительность виртуальных машин.

Облачные системы всегда были интересны научному сообществу как источник «бесконечных» ресурсов. Важным преимуществом переноса научных задач в облако является переносимость всего стека ПО между используемыми вычислительными ресурсами, а также между исследователями. Однако существующие ограничения производительности облачных систем долгое время не позволяли эффективно решать высокопроизводительные задачи.

Кроме того, существующие облачные платформы требуют доработки для обеспечения удобства пользователей – должна быть разработана экосистема, позволяющая ученому решать свои задачи без необходимости большого числа подготовительных шагов. Например, при необходимости запуска задачи на MPI-кластере, пользователь не должен самостоятельно настраивать данный кластер с использованием выделенных в облаке ресурсов.

Отдельной проблемой является обеспечение эффективного хранилища для данных (общей файловой системы). Такое хранилище должно обеспечивать высокую производительность в облаке, в том числе поддерживать параллельный ввод-вывод. Кроме того, должны

поддерживаться удобные средства вывода данных из системы. Проблема заключается в том, что при росте объема данных (и переходе к концепции BigData) становится все сложнее удовлетворять данным требованиям. При работе с BigData требуется также поддержка специализированных моделей обработки таких данных (в частности, MapReduce).

При переносе в облако также возникают проблемы, связанные с безопасностью, поскольку возможности пользователя по конфигурированию всего стека ПО существенно возрастают по сравнению с традиционными вычислительными кластерами. Неопытный пользователь может настроить облачный сервер таким образом, что он станет уязвимым для вредоносного ПО.

В настоящее время, в связи с развитием технологий виртуализации и повышением производительности облачных ресурсов, возникают новые возможности применения концепции облачных вычислений. Целью данной работы является разработка облачной системы, предназначенной для организации высокопроизводительных вычислений. В работе приводится обзор архитектуры разработанной и реализованной облачной системы «виртуальный суперкомпьютер» (далее – ВСК), позволяющей эффективно (с минимальными накладными расходами) решать высокопроизводительные задачи. Основным вкладом данной работы является:

1) Описание основных проблем, возникающих при переносе высокопроизводительных вычислений в облако, и методов их решения. В том числе, выявление основных источников накладных расходов при виртуализации и разработка методов их снижения.
2) Разработка и реализация облачной системы «виртуальный суперкомпьютер», предназначенной для решения высокопроизводительных задач в облаке. В системе ВСК реализован сервис уровня инфраструктуры, на базе которого реализован сервис по предоставлению доступа к MPI-кластеру, а также сервис OpenFOAM, демонстрирующий возможность работы с пакетом OpenFOAM как с облачным приложением. В системе основной упор делается на обеспечение производительности виртуальной среды на уровне не менее 90% от производительности реального кластера.
3) Описание перспективных подходов к организации научных и промышленных вычислений в облачной среде, перспективных разработок в данной области в мире.

В разделе 2 приводятся обзор литературы. Особенности архитектуры разработанной системы ВСК рассматриваются в разделе 3.

2 Обзор литературы

Концепция облачных вычислений подразумевает удаленный доступ к вычислительным ресурсам с возможностью получения ровно того объема ресурсов, который требуется. Согласно определению Национального института стандартов и технологий США (NIST) [2], облачные вычисления – это модель организации удаленного доступа по запросу к разделяемому набору конфигурируемых вычислительных ресурсов (например, сетей, серверов, хранилищ, приложений и сервисов), которые могут быть быстро выделены и освобождены с минимальными расходами на управление или взаимодействие с провайдером услуг.

Согласно NIST, выделяют несколько уровней сервиса, предоставляемого облачной системой. Сервис самого «низкого» уровня – предоставление инфраструктуры (IaaS, infrastructure as a service), позволяет получать доступ к вычислительным серверам, сетям передачи данных, блочным хранилищам. Как правило, сервис такого уровня реализуется с использованием технологий виртуализации, однако возможны и решения на базе аппаратного управления вычислительными ресурсами [3, 4] (которые также можно считать подходом наподобие виртуализации). Наиболее известными облачными платформами уровня инфраструктуры с открытым исходным кодом являются системы Eucalyptus [5], Nimbus [6], OpenNebula [7], CloudStack [8], OpenStack [9]. Среди систем с закрытым исходным кодом известны Amazon EC2 [10], Windows Azure [11], Google Compute Engine [12] и другие.

Сервис более высокого уровня – предоставление доступа к платформе (PaaS, platform as a service) и приложению (SaaS, software as a service). Сервис уровня платформы подразумевает доступ к вычислительной платформе, например, к окружению для разработки и выполнения программ, к СУБД, веб-серверу. Сервис уровня приложений позволяет работать с конкретным приложением удаленно. Зачастую сервисы более высокого уровня реализуются на основе сервиса IaaS, предоставляющего удобные средства для управления набором необходимых вычислительных ресурсов.

С точки зрения высокопроизводительных задач, можно выделить некоторые наиболее актуальные требования к облачной системе для эффективного решения таких задач:

1) Обеспечение высокой производительности вычислительных ресурсов. Данная задача должна быть решена на уровне сервиса инфраструктуры. В рамках данной работы рассматриваются методы эффективного выполнения параллельных программ в виртуальных машинах.
2) Предоставление доступа к коммуникационной среде, достаточной по своим характеристикам для решения конкретной задачи. Данная задача также относится к сервису

инфраструктуры. В рамках системы ВСК предоставляется доступ к сети Infiniband.

3) Организация удобной для пользователя экосистемы, позволяющей решать требуемые задачи без необходимости выполнения лишних шагов по конфигурированию и настройке, которые могут быть автоматизированы. Данное требование, по сути, определяет специализированный сервис уровня платформы. В данной работе приводится пример организации такого сервиса по доступу к приложению OpenFOAM.
4) Ключевым компонентом облачной экосистемы для решения научных задач также является высокопроизводительное хранилище. Такое хранилище должно быть доступно посредством различных интерфейсов, по крайней мере, оно должно быть доступно в облачной среде (в виде общей ФС) и должна быть возможность удобного подключения хранилища к конкретному облачному сервису. Также необходимо предусмотреть возможность работы с хранилищем на ПК пользователя. В настоящее время подобной системы организации облачного хранилища не существует.

В следующем подразделе приводится классификация HPC-приложений, позволяющая выделить классы приложений, которые могут быть эффективно перенесены в облако.

2.1 *Классификация HPC приложений и переносимость в облако*

Можно выделить несколько классов HPC-приложений в соответствии с их требованиями к вычислительной среде. В работе [13] существующие HPC-приложения классифицируются по характеру используемых ресурсов: сильносвязанные вычисления большого масштаба, вычисления среднего масштаба и высокомощные вычисления (high throughput computing). Для каждого из классов можно выделить основные плюсы от переноса в облако.

Сильносвязанные широкомасштабные вычисления подразумевают запуск MPI-задач на суперкомпьютерах с использованием большого числа процессоров (порядка тысяч и более ядер). Такие задачи требуют значительного времени для завершения и, как правило, запускаются посредством системы очередей задач. Суперкомпьютерные центры предоставляют для таких задач архивное хранилище и параллельные ФС. Такие задачи, как правило, показывают существенное снижение производительности в облаке.

Сильносвязанные вычисления среднего масштаба подразумевают запуск на ресурсах объемом 10-1000 вычислительных ядер. Для запуска таких задач чаще используются небольшие кластеры, обслуживаемые самими

научными группами. Данный класс задач хорошо подходит для переноса в облако.

Высокомощными называют задачи, которые подразумевают выполнение асинхронных независимых вычислений. Такие задачи часто связаны с пред- и пост-обработкой данных, и во многих случаях производятся на ПК исследователя. Однако в последние годы объем научных данных неуклонно растет (в частности, происходит переход к концепции BigData), и стадии пред- и пост-обработки также выполняются на высокопроизводительных вычислителях. Высокомощные задачи очень хорошо подходят для переноса в облако, однако требуют большой пропускной способности и значительного объема хранилища.

При разработке системы ВСК основной упор делался на приложения, относящиеся к классу сильносвязанных вычислений среднего масштаба, поскольку именно такие приложения требуют достаточно высокой производительности системы виртуализации и при этом могут эффективно выполняться в виртуальной среде. Высокомощные задачи решаются и в существующих облачных системах.

В следующем подразделе приводится обзор перспективных подходов к переносу HPC в облако, разрабатываемых в мире.

2.2 *Перспективные подходы к переносу HPC в облако*

В настоящее время исследователи рассматривают несколько основных задач, связанных с переносом HPC в облако.

Как правило, исследование облачных систем требует значительного объема вычислительных ресурсов. Исследовательскими организациями развернуты тестовые платформы, позволяющие оценить существующие и перспективные технологии. В США разрабатывается проект FutureGrid [14], который представляет собой программно-аппаратную платформу, объединяющую несколько удаленных вычислительных кластеров и предназначенную для проведения экспериментов в области облачных и грид вычислений. Общий объем задействованных ресурсов составляет около 5600 вычислительных ядер. Платформа поддерживает запуск как в виртуализированной среде, так и на реальном оборудовании. Особый упор делается на возможность сравнить различные технологии с точки зрения как производительности, так и удобства использования, а также на воспроизводимость экспериментов. В качестве облачных платформ доступны системы Nimbus и Eucalyptus.

Особенностью платформы FutureGrid является динамическое выделение ресурсов в зависимости от текущих запросов пользователей. В частности, возможно динамическое переконфигурирование серверов для решения HPC-задач в кластере либо для предоставления облачных ресурсов (виртуальных машин), причем поддерживается создание различных специфичных окружений для решения задач в конкретной области. Для переконфигурирования используются системы управления

аппаратными ресурсами xCAT [3] и Moab [4]. Пользователям платформы доступны команды для создания конкретных окружений, например, Hadoop-кластера. Фактически, пользователю предоставляется облачный сервис уровня платформы, реализованный с использованием виртуализации на уровне оборудования.

Особое внимание в проекте FutureGrid уделяется воспроизводимости экспериментов. Каждый эксперимент определяется как совокупность образа и последовательности определенных шагов. Образ в системе может быть использован как для запуска в облаке, так и для запуска на реальном сервере. Образ специально подготавливается для интеграции в систему FutureGrid. В качестве средства конфигурирования ПО образа применяется система BCFG2 [15], клиент которой интегрируется в каждый образ. Это позволяет пользователю при генерации образа указать требуемый набор ПО.

Другим известным проектом по анализу применимости облачных систем для решения HPC-задач стал проект Magellan [13]. В рамках данного проекта, выполненного в США по заказу Министерства энергетики (Department of Energy), проведено исследование облачных систем Eucalyptus, OpenStack, а также открытой платформы Hadoop. В качестве тестового стенда для оценки облачных технологий использовались HPC ресурсы общим объемом порядка 10000 ядер.

В настоящее время также рассматриваются вопросы использования существующих облачных систем уровня инфраструктуры для расширения вычислительного поля задачи по запросу. В данном случае речь идет о специализированных сервисах уровня платформы, позволяющих использовать преимущества облачных вычислений для повышения качества сервиса.

В работе [16] авторы описывают облачную систему, позволяющую объединять вычислительные ресурсы уровня инфраструктуры в кластеры различного масштаба, при этом поддерживается гибкость (эластичность) системы, то есть возможность динамически изменять объем задействованных в кластере серверов (ВМ). Система основана на облачном планировщике EPU (Elastic Processing Unit) [17] и для масштабирования производительности может использовать ресурсы проекта FutureGrid, а также сервиса Amazon EC2.

Для осуществления обратной связи выполнена интеграция с планировщиком задач Torque [18]. В планировщик встроен сенсор, позволяющий отслеживать текущие запросы ресурсов приложениями (в частности, отслеживается размер очереди задач). Для обработки данных сенсора реализован механизм принятия решений, использующий заданные администратором политики. По результатам работы механизма возможен запуск или остановка ВМ с использованием удаленных облачных ресурсов.

Для ускорения базовой конфигурации ВМ авторы используют фреймворк Chef [19], позволяющий автоматизировать действия по

системному администрированию ВМ, вследствие чего возможно использование практически любого базового образа ВМ на основе Linux, доступного в конкретном облаке.

Отдельным компонентом системы является брокер контекста, позволяющий интегрировать вновь запущенные ВМ в вычислительную среду, а также отсоединить удаляемые ВМ. Брокер, в частности, обеспечивает обмен IP-адресами и SSH-ключами в кластере. Брокер при этом выполняется непрерывно, что позволяет динамически изменять размер вычислительного кластера.

В целом, в качестве особенностей системы можно выделить независимость от конкретного поставщика облачного сервиса, а также независимость от конкретного параллельного приложения.

В работе [20] этой же группы исследователей рассматривается проблема ненадежности удаленных облачных ресурсов и каналов доступа к ним. Авторы описывают архитектуру сервиса, ключевой особенностью которого является поддержка высокой доступности (high availability) ресурсов и самого сервиса. Сервис использует облачные ресурсы различных провайдеров, что позволяет снизить риски при сбое одного из провайдеров. При этом концепция сервиса подразумевает, что выполняемая задача является слабосвязанной и возобновляемой при сбое конкретной ВМ. Таким образом, накладываются существенные ограничения на модель программирования, подразумевается наличие упорядоченных и структурированных подзадач, которые могут независимо выполняться в ВМ.

Отдельным направлением исследований является адаптация существующих приложений к облачной среде либо разработка специализированных моделей программирования для облачных вычислений.

В работе [21] авторы рассматривают особенности переноса научного приложения в облачную систему, построенную на основе ресурсов проектов FutureGrid, Magellan, а также облака Amazon EC2. Рассматриваются приложения, относящиеся к классу слабосвязанных параллельных приложений. Такие приложения могут быть сравнительно эффективно выполнены в облачной среде за счет невысоких требований к производительности и пропускной способности вычислительной сети.

Авторы рассматривают научную задачу в концепции процесса (workflow), описывающего последовательности вызываемых программ и их входных и выходных данных в виде направленного графа. Для выполнения процесса необходимо выделить ресурсы, отобразить процесс (вызываемые программы) на выделенные ресурсы, выполнить задачу, и освободить ресурсы.

Отображение процесса на вычислительные ресурсы может выполняться с использованием различных подходов. Авторы применяют для этого фреймворк Pegasus [22]. Фреймворк позволяет преобразовать абстрактное описание процесса в набор описаний задач, пригодный для

выполнения. Далее задача решается с использованием нескольких независимых облачных систем. Такая концепция также известна под названием «Sky computing» [23].

С нашей точки зрения, в настоящее время наиболее актуален перенос HPC-приложений в облако без необходимости модификации самого приложения. В частности, объем существующих MPI-приложений настолько велик, что зачастую не представляется возможным каким-либо образом переписать приложение в новой модели программирования. При этом сервис уровня инфраструктуры является плохо применимым для научных вычислений, так как требует от пользователя умений в области системного администрирования ОС и кластера в целом. Таким образом, целесообразно развивать сервисы более высокого уровня, которые бы позволяли пользователю быстро получать доступ к необходимой вычислительной среде, будь то MPI кластер или конкретный прикладной пакет. В следующем разделе описывается архитектура системы ВСК, в которой решаются некоторые из проблем, возникающих при переносе HPC в облако.

3 Особенности архитектуры облачной системы ВСК

Система ВСК состоит из нескольких относительно независимых компонентов. Облачная платформа разработана на основе системы OpenStack.

Основными компонентами системы ВСК являются (см. рис. 1):

1) Система виртуализации. В рамках системы ВСК применяется полная виртуализация платформы с использованием аппаратных возможностей процессора. Более подробное описание системы виртуализации приведено в подразделе 3.1.
2) Система управления вычислительными ресурсами (контроллер облака) – OpenStack Compute (Nova). Отвечает за предоставление виртуальных машин посредством управления монитором ВМ, выделение и освобождение различных ресурсов (виртуальных сетей, хранилищ), реализацию API (включая Amazon EC2 API и собственный интерфейс OpenStack API). Более подробное описание контроллера облака приведено в подразделе 3.2.
3) Хранилище образов ВМ OpenStack Image Service (Glance). Предназначено для организации хранения данных образов ВМ в различных форматах и поддержания каталога информации о доступных образах ВМ. Данные могут храниться с использованием различных систем хранения.

4) Хранилище данных. Предоставляет объектное и блочное хранилище. Функциональность аналогична сервисам Amazon S3 (Simple Storage Service) и EBS (Elastic Block Storage).

   Объектное хранилище позволяет пользователю хранить данные произвольной природы с использованием удаленного веб-сервиса. В хранилище обеспечивается автоматическая репликация данных. Объектное хранилище может быть использовано для хранения данных образов.

   Блочное хранилище (Cinder) позволяет выделать блочные устройства, которые являются устойчивым хранилищем информации и могут быть подключены к ВМ. В то же время, основные диски ВМ не являются устойчивыми, после удаления ВМ основной диск также удаляется. Блочное устройство может быть одновременно подключено не более чем к одной ВМ.

   Как объектное, так и блочное хранилища реализованы с использованием распределенной ФС CEPH [24]. CEPH предоставляет ряд интерфейсов для доступа к данным, среди которых S3-совместимое API, API уровня блочного хранилища, библиотека для программного доступа. Таким образом, CEPH позволяет реализовать все необходимые сервисы хранения данных на основе единого распределенного хранилища, поддерживающего репликацию.

5) Система авторизации и аутентификации OpenStack Identity (Keystone). Система используется остальными компонентами для авторизации и аутентификации различных агентов (пользователей, сервисов).

6) Пользовательский веб-интерфейс, основанный на системе OpenStack Dashboard (Horizon). Интерфейс переработан с учетом специфики решаемой задачи, выполнен перевод на русский язык.

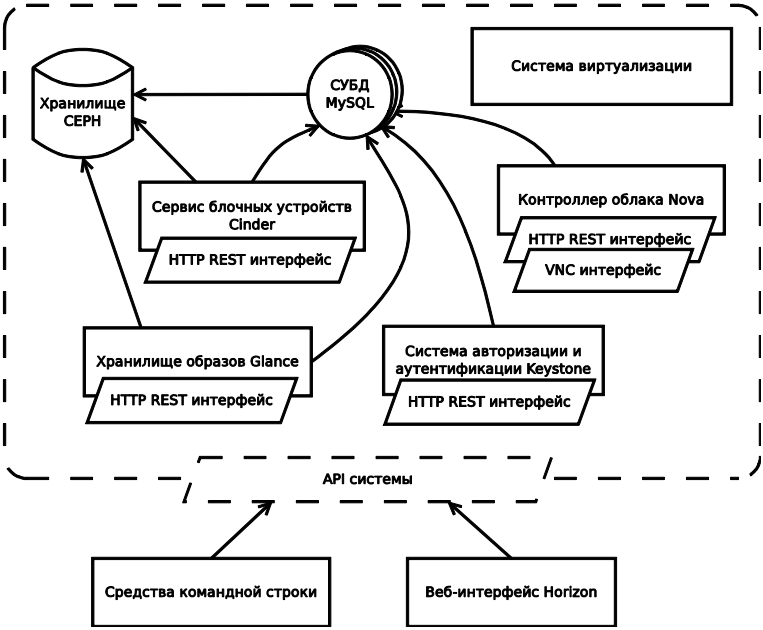

Рис. 1. Компоненты системы ВСК.

Компоненты Nova, Keystone, Glance, Cinder используют для работы БД SQL. В системе ВСК используется СУБД MySQL. Каждый из перечисленных компонентов работает с БД независимо. При этом для хранения БД также используется реплицируемое хранилище CEPH, что повышает надежность БД.

Компоненты Nova, Keystone, Glance, Cinder реализуют HTTP REST интерфейсы, с использованием которых осуществляется взаимодействие между ними. Данные интерфейсы также используются системами взаимодействия с пользователем (веб-интерфейс Horizon, средства командной строки).

Недостатком системы ВСК является отсутствие интегрированной в облако концепции высокопроизводительной общей ФС. Как известно, существующие облачные системы не имеют средств для предоставления и управления такими ФС. Более того, на основе выделяемых пользователю ресурсов практически невозможно организовать общую ФС достаточной производительности. В рамках системы ВСК данная проблема обходится путем организации независимой от системы общей ФС, используемой специализированными образами ВМ. С точки зрения концепции облачных вычислений, такое решение не является удобным, масштабируемым, безопасным, поскольку в рамках облачной системы все используемые ресурсы должны находиться под управлением самой системы. Таким образом, в настоящее время существует потребность в

специализированных сервисах, позволяющих предоставлять ВМ доступ к высокопроизводительным ФС в рамках облачной платформы.

3.1 *Система 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 (Kernel-based Virtual Machine) [25] и гипервизоры, основанные на Xen [26]. В данной работе используется гипервизор KVM, который, на наш взгляд, более перспективен и удобен в использовании.

Гипервизор KVM реализует виртуальный процессор и систему памяти, остальные компоненты виртуальной машины реализованы эмулятором QEMU [27]. 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). Для взаимодействия с гипервизором используется библиотека libvirt.

3.1.1 Обеспечение производительности ВМ

Основные полученные результаты в области производительности системы виртуализации на платформе x86 описаны в работах [28-31].

Можно выделить следующие этапы и методы улучшения производительности, разработанные в ходе выполнения проекта:

1) Выделение всех доступных ресурсов ВМ. В частности, выделение всех процессорных ядер, большей части памяти ВМ.
2) Обеспечение корректного отображения ресурсов сервера в ВМ. Необходимо правильно отобразить топологию системы (SMP или NUMA) в ВМ для того, чтобы гостевая ОС могла оптимально управлять процессами и памятью.
3) Настройка основной ОС, гостевой ОС для улучшения производительности приложений. Для выявления накладных расходов на виртуализацию необходимо добиться эффективной работы конкретных приложений.
4) Оптимизация производительности ввода-вывода. Требуется передать коммуникационное устройство в ВМ, что ведет к дополнительным накладным расходам.

Для тестирования применялись пакеты NAS Parallel Benchmarks [32], HPC Challenge [33], SPEC MPI2007 [34]. В целом, достигнут заданный уровень накладных расходов – не более 10% (на большинстве тестов в пределах 6%) при использовании до 1024 процессорных ядер. При этом в ходе работ были выявлены характеристики приложения, влияющие на производительность в ВМ. Среди таких характеристик:

1) Характер коммуникаций. Определяется частотой, объемом, типом коммуникаций. При виртуализации накладные расходы при «мелкозернистых» коммуникациях существенно повышаются. Синтетические тесты, содержащие частые вызовы групповых операций MPI, могут приводить к накладным расходам от десятков до сотен процентов, в зависимости от объема посылок и частоты коммуникаций. Это связано с возрастающим влиянием шума основной ОС на производительность. Проблема влияния шума ОС на производительность параллельного приложения рассматривается в статье [35].
2) Особенности работы с вводом-выводом данных. В рамках данной работы исследовались приложения, не предъявляющие высоких требований к вводу-выводу данных, в том числе к общей ФС. В качестве общей ФС использовалась система NFS, предоставляемая отдельным сервером. Данная ФС подключалась к вычислительным серверам посредством сети Ethernet, в случае виртуальных машин ряд компонентов такой сети (сетевой адаптер ВМ, сетевой мост) реализованы программным путем. Это приводит к дополнительным накладным расходам, которые, однако, не оказали существенного влияния на исследуемые приложения. При рассмотрении приложений, осуществляющих большие объемы взаимодействия с хранилищем, в том числе параллельный ввод-вывод, необходимо использовать файловую систему, реализованную на основе сети высокой пропускной способности (например, Fibre Channel, Infiniband). В таком случае могут возникнуть проблемы, связанные с эффективным предоставлением коммуникационного адаптера виртуальной машине.

3.2 *Контроллер облака*

Система Nova построена в концепции «неразделяемых ресурсов» (shared-nothing architecture) с использованием модели передачи сообщений. За счет этого, все основные компоненты системы Nova могут выполняться на отдельных серверах. Взаимодействие происходит с использованием удаленного вызова процедур (RPC, remote procedure call), реализованного на базе протокола AMQP [36]. Данные системы хранятся в БД SQL, используемой всеми компонентами Nova. БД обновляется при выполнении каждой операции.

Основными компонентами (сервисами в смысле режима выполнения) контроллера облака Nova являются (см. *Рис.* ***2***):

1) Сервис nova-api. Обрабатывает API-запросы от пользователей. Поддерживается несколько API, включая Amazon EC2,

OpenStack API. Инициирует большую часть операций в облаке, а также выполняет проверку квот. Реализация имеет модульную структуру, что позволяет предоставлять только необходимые API.

2) Сервис nova-compute. Данный сервис получает запросы на управление ВМ из очереди сообщений и взаимодействует с конкретным гипервизором на вычислительном узле. Для взаимодействия используется API, специфичное для гипервизора. Поддерживаются все наиболее распространенные гипервизоры (XenServer/XCP, KVM/QEMU, LXC, VMWare ESXi, Hyper-V), а также виртуализация на уровне аппаратуры.
3) Сервис nova-schedule. Планировщик ВМ и хранилища. Определяет, на каком сервере должна быть запущена ВМ и где должно быть выделено блочное хранилище в соответствии с заданными политиками. При разработке системы ВСК для учета специфики высокопроизводительных вычислений был разработан специальный планировщик, учитывающий топологию вычислительной сети. Более подробно см. [37].
4) Сервис nova-network. Процесс, отвечающий за организацию виртуальных сетей для групп ВМ. nova-network поддерживает несколько базовых моделей организации сетей; в рамках проекта OpenStack также разрабатывается компонент Quantum, предоставляющий сервис по управлению виртуальными сетями различной сложности.

   В рамках системы ВСК каждая группа ВМ объединена приватной сетью, к такой сети нельзя получить доступ извне. При этом также поддерживается концепция публичных IP-адресов. Публичный адрес может быть выделен по запросу пользователя из пула доступных адресов и назначен конкретной ВМ.
5) Поддержка удаленного доступа к ВМ по протоколу VNC. В рамках системы ВСК обеспечивается сервисами nova-consoleauth и nova-novncproxy. Сервис nova-novncproxy обеспечивает проксирование между компонентом noVNC, обеспечивающим отображение дисплея в веб-интерфейсе, и VNC-сервером на конкретном сервере ВМ. Сервис nova-consoleauth предназначен для авторизации доступа к дисплею ВМ.
6) Сервис nova-conductor. Сервис отвечает за взаимодействие с БД. Основная задача сервиса – ограничить взаимодействие других компонентов (прежде всего nova-compute) с БД. Это необходимо для повышения безопасности (так как nova-compute является наиболее уязвимым компонентом) и для упрощения обновления системы.

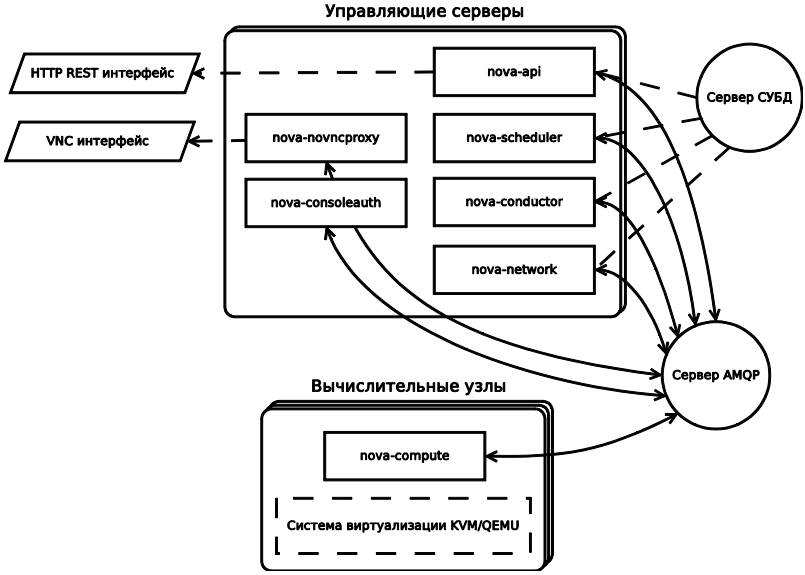

Рис. 2. Архитектура облачного контроллера Nova.

На рис. 2 описаны связи отдельных сервисов облачного контроллера Nova. Как видно, на вычислительных узлах достаточно запуска только сервисов nova-compute, что способствует снижению шума при виртуализации. За счет использования протокола AMQP отдельные сервисы могут быть запущены на произвольных серверах, связанных сетью. Кроме того, число процессов конкретного сервиса может быть любым, что позволяет при необходимости масштабировать производительность отдельных сервисов.

3.2.1 Особенности контроллера, связанные с HPC

В рамках сервиса уровня инфраструктуры основной единицей конфигурации ПО облачного сервера является образ ВМ (image), содержащий в себе данные диска, пригодные для запуска сервера. Образ может содержать предустановленную ОС и ПО. Образ используется для инициализации данных диска запускаемого сервера.

При создании облачного сервера недостаточно наличия только образа, который является только шаблоном и требует дополнительной конфигурации. Облачные системы поддерживают средства дополнительной настройки сервера, некоторые исследователи также используют понятие контекстуализации [38]. Среди необходимых настроек – публичные ключи для осуществления удаленного доступа к серверу. Для передачи настроек в ВМ может использоваться встраивание файлов в диск сервера до запуска (при этом облачная

система должна учитывать особенности конкретной ОС, установленной в образ, и используемой ФС). Кроме того, может быть реализована поддержка специального сервера метаданных либо дополнительного конфигурационного диска сервера (при этом внутри образа должен быть предустановлен агент облачной системы, который может взаимодействовать с такими источниками информации).

В системе ВСК сервис по предоставлению MPI-кластера реализован с использованием специального образа ВМ. После запуска группы ВМ из такого образа, достаточно выбрать какую-либо ВМ в качестве управляющего узла и запускать задачи командой mpirun. Единственное, что необходимо сделать пользователю – подготовить файл с IP-адресами серверов и передать его как параметр команде mpirun.

Концепция групп ВМ была специально разработана для учета специфики высокопроизводительных вычислений. Каждая группа ВМ при запуске планируется как единое целое, что позволяет планировщику располагать ВМ из одной группы оптимальным способом. Для задания характеристик оборудования облачного сервера используется концепция типа ВМ (flavor), позволяющего описать аппаратные требования сервера, например, частоту и число ядер процессора, объем ОЗУ, основного диска. Каждой группе ВМ ставится в соответствие конкретный тип ВМ, что обеспечивает запуск каждой ВМ группы с использованием одного и того же объема ресурсов и обеспечивает однородность виртуального кластера (см.рис. 3).

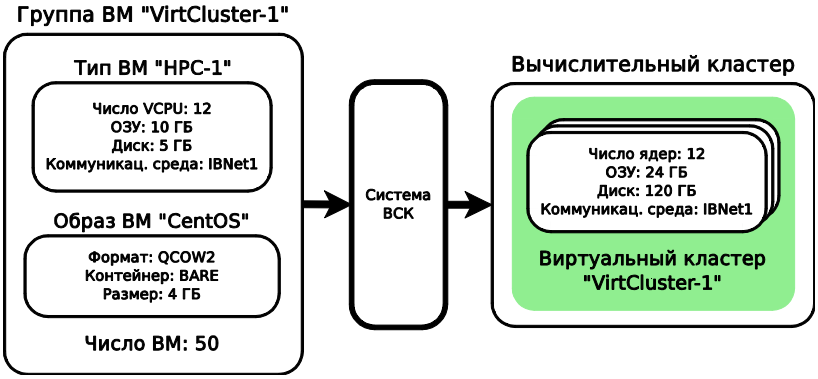

Рис. 3. Концепция Групп ВМ, запуск виртуального кластера.

Концепция типа ВМ была дополнена для обеспечения возможности создания виртуальных машин высокой производительности. Добавлена возможность задавать посредством типа ВМ конфигурацию топологии ВМ (например, параметры NUMA), задавать тип страниц для выделения памяти ВМ.

Также была реализована поддержка проброса устройств сервера в ВМ на основе интерфейса библиотеки libvirt. Проброс устройств необходим для

эффективной работы коммуникационной среды в группе ВМ. На каждом сервере необходимо указать пробрасываемые устройства в конфигурации облачного контроллера Nova. Каждое устройство имеет метку, указывающую его принадлежность к конкретной группе устройств (например, вычислительной сети). Для запроса устройства в ВМ необходимо создать новый тип ВМ, где указывается требуемый для запуска набор меток устройств.

Для конфигурации сетей в рамках системы ВСК используются предустановленные настройки. Виртуальные машины каждой группы объединены в частную сеть, не имеющую доступа в сеть Интернет. Для доступа к сети Интернет облачная система позволяет отдельно назначить публичный IP-адрес конкретной ВМ.

3.3 Особенности архитектуры сервиса OpenFOAM

Разработка и реализация сервиса OpenFOAM была выполнена в целях демонстрации возможностей концепции облачных вычислений в применении к высокопроизводительным задачам. Сервис позволяет решать набор задач в области механики сплошной среды.

Сервис OpenFOAM предоставляет пользователю дополнительную вкладку в веб-интерфейсе, на которой доступны операции по управлению задачами. Каждая задача определяется конкретными входными данными (OpenFOAM case), заданными пользователем. Также параметрами задачи являются метод разбиения сетки, используемый решатель, параметры решателя, число используемых процессов для решения. Число процессов определяет объем создаваемого при запуске виртуального кластера.

После задания входных данных, все стадии выполнения задачи происходят автоматически. Во время работы можно отслеживать текущее состояние задачи в веб-интерфейсе. После окончания выполнения, пользователь может загрузить результаты и журналы работы на свой ПК.

При реализации сервиса были использованы API, предоставляемые системой. При этом все данные о задачах хранятся в объектном хранилище, доступном посредством Swift API. При запуске задачи на управляющем сервере создается процесс, обеспечивающий выполнение жизненного цикла задачи.

Заключение

В данной работе описаны основные проблемы, возникающие при переносе высокопроизводительных вычислений в облако. Ключевой проблемой является падение производительности при использовании виртуализации. В ходе работы были выявлены основные источники накладных расходов и разработаны методы повышения

производительности ВМ. Также существенной проблемой является предоставление общей ФС, подходящей для высокопроизводительного (в том числе параллельного) ввода-вывода в рамках облачной системы.

Рассматривается подход к организации высокопроизводительного облачного сервиса с использованием виртуализации. Описана архитектура разработанной и реализованной облачной системы «виртуальный суперкомпьютер», основанной на платформе OpenStack и системе виртуализации KVM/QEMU. Компоненты системы ВСК доработаны для учета специфики высокопроизводительных вычислений, в частности, выполнена доводка и настройка системы виртуализации, что позволило достичь уровня накладных расходов не более 10% (не более 6% для большинства тестов) при использовании, по крайней мере, 1024 процессорных ядер. В рамках платформы OpenStack выполнена разработка и реализация необходимых концепций, в частности, добавлена возможность передачи устройства сервера в ВМ, а также возможность объединения виртуальных машин в группы для совместного планирования. Разработан специализированный планировщик ВМ, учитывающий топологию «толстое дерево» сети Infiniband.

В рамках разработанной системы ВСК предоставляется доступ к сервису уровня инфраструктуры, на базе которого реализован сервис OpenFOAM, демонстрирующий возможность работы с пакетом OpenFOAM как с облачным приложением.

В перспективе при разработке сервисов, ориентированных на применение в области HPC, необходимо, прежде всего, обеспечивать удобство решения задач для пользователя путем создания сервисов уровня платформы и приложений. Производительность может достигаться с использованием виртуализации платформы и существующих наработок, либо с использованием виртуализации на уровне оборудования. Кроме того, необходимо обеспечить пользователю единую среду для хранения данных, доступную как в облачной инфраструктуре, так и на компьютере пользователя. При этом такая среда должна обладать достаточной производительностью для решения задач пользователя. В целом, все предоставляемые сервисы должны быть объединены в единую экосистему, удобную для пользователя.

Список литературы

[1] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. Above the Clouds: A Berkeley View of Cloud Computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009.

[2] Peter Mell and Tim Grance. The NIST Definition of Cloud Computing. Technical report, July 2009.

[3] xCAT – Extreme Cloud Administration Toolkit. http://xcat.sourceforge.net/, 1999. [Online; accessed 14-May-2013].

[4] Moab Suite. http://www.adaptivecomputing.com/products/. [Online; accessed 14-May-2013].

[5] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The eucalyptus open-source cloud-computing system. In *Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on*, pages 124–131, 2009.

[6] Katarzyna Keahey, Ian Foster, Tim Freeman, and Xuehai Zhang. Virtual workspaces: Achieving quality of service and quality of life in the grid. *Scientific Programming*, 13(4):265–275, 2005.

[7] J Fontán, T Vázquez, L Gonzalez, Ruben S Montero, and IM Llorente. Opennebula: The open source virtual machine manager for cluster computing. In *Open Source Grid and Cluster Software Conference*, 2008.

[8] Apache CloudStack: Open Source Cloud Computing. http://-cloudstack.apache.org/, 2010. [Online; accessed 14-May-2013].

[9] OpenStack Open Source Cloud Computing Software. http://-www.openstack.org/, 2010. [Online; accessed 14-May-2013].

[10] Amazon Elastic Compute Cloud (Amazon EC2). http://aws.amazon.com/ec2/, 2006. [Online; accessed 14-May-2013].

[11] Windows Azure: Microsoft's Cloud Platform. http://-www.windowsazure.com/en-us/, 2010. [Online; accessed 14-May-2013].

[12] Google Compute Engine. https://cloud.google.com/products/compute-engine, 2012. [Online; accessed 14-May-2013].

[13] Lavanya Ramakrishnan, Piotr T. Zbiegel, Scott Campbell, Rick Bradshaw, Richard Shane Canon, Susan Coghlan, Iwona Sakrejda, Narayan Desai, Tina Declerck, and Anping Liu. Magellan: experiences from a science cloud. In *Proceedings of the 2nd international workshop on Scientific cloud computing*, ScienceCloud '11, pages 49–58, New York, NY, USA, 2011. ACM.

[14] G. von Laszewski, G.C. Fox, Fugang Wang, A.J. Younge, A. Kulshrestha, G.G. Pike, W. Smith, J. Vöckler, R.J. Figueiredo, J. Fortes, and K. Keahey. Design of the futuregrid experiment management framework. In *Gateway Computing Environments Workshop (GCE), 2010*, pages 1–10, 2010.

[15] Bcfg2 – A Configuration Management System. http://trac.mcs.anl.gov/-projects/bcfg2/, 2004. [Online; accessed 14-May-2013].

[16] Paul Marshall, Henry Tufo, Kate Keahey, David LaBissoniere, and H.M. Woitaszek. Architecting a large-scale elastic environment – recontextualization and adaptive cloud services for scientific computing. In *Proceedings of the 7th International Conference on Software Paradigm Trends (ICSOFT)*, Rome, Italy, 2012.

[17] CEI Elastic Processing Unit (EPU) Services and Agents. https://github.com/-ooici/epu, 2011. [Online; accessed 14-May-2013].
[18] TORQUE Resource Manager. http://www.adaptivecomputing.com/products/-open-source/torque/, 2003. [Online; accessed 14-May-2013].
[19] Chef Configuration Management. http://www.adaptivecomputing.com/-products/, 2009. [Online; accessed 14-May-2013].
[20] Kate Keahey, Patrick Armstrong, John Bresnahan, David LaBissoniere, and Pierre Riteau. Infrastructure outsourcing in multi-cloud environment. In *Proceedings of the 2012 workshop on Cloud services, federation, and the 8th open cirrus summit*, FederatedClouds '12, pages 33–38, New York, NY, USA, 2012. ACM.
[21] Jens-Sönke Vöckler, Gideon Juve, Ewa Deelman, Mats Rynge, and Bruce Berriman. Experiences using cloud computing for a scientific workflow application. In *Proceedings of the 2nd international workshop on Scientific cloud computing*, ScienceCloud '11, pages 15–24, New York, NY, USA, 2011. ACM.
[22] Ewa Deelman, Gurmeet Singh, Mei-Hui Su, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Karan Vahi, G Bruce Berriman, John Good, Anastasia Laity, Joseph C. Jacob, and Daniel S. Katz. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 13(3):219–237, 2005.
[23] K. Keahey, M. Tsugawa, A. Matsunaga, and J. A B Fortes. Sky computing. *Internet Computing, IEEE*, 13(5):43–51, 2009.
[24] Sage A Weil, Scott A Brandt, Ethan L Miller, Darrell DE Long, and Carlos Maltzahn. Ceph: A scalable, high-performance distributed file system. In *Proceedings of the 7th symposium on Operating systems design and implementation*, pages 307–320. USENIX Association, 2006.
[25] Avi Kivity, Yaniv Kamay, Dor Laor, Uri Lublin, and Anthony Liguori. kvm: the linux virtual machine monitor. In *Proceedings of the Linux Symposium*, volume 1, pages 225–230, 2007.
[26] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. *ACM SIGOPS Operating Systems Review*, 37(5):164–177, 2003.
[27] Fabrice Bellard. Qemu, a fast and portable dynamic translator. USENIX, 2005.
[28] А. О. Кудрявцев, В. К. Кошелев, А. И. Аветисян. Перспективы виртуализации высокопроизводительных систем архитектуры x64. *Труды Института системного программирования РАН, том 22*, с. 189–209, 2012.
[29] Alexander Kudryavtsev, Vladimir Koshelev, and Arutyun Avetisyan. Modern HPC cluster virtualization using KVM and palacios. In *High Performance Computing (HiPC), 2012 19th International Conference on*, pages 1–9, 2012.
[30] Alexander Kudryavtsev, Vladimir Koshelev, Boris Pavlovic, and Arutyun Avetisyan. Virtualizing HPC applications using modern hypervisors. In *Proceedings of the 2012 workshop on Cloud services, federation, and the 8th open cirrus summit*, FederatedClouds '12, pages 7–12, New York, NY, USA, 2012. ACM.
[31] А.О. Кудрявцев, В.К. Кошелев, А.О. Избышев, А.И. Аветисян. Высокопроизводительные вычисления как облачный сервис: ключевые

проблемы. *Параллельные вычислительные технологии (ПаВТ'2013): труды международной научной конференции*, с. 432–438, 2013.
[32] Rob F Van der Wijngaart and Parkson Wong. Nas parallel benchmarks version 2.4. Technical report, NAS technical report, NAS-02-007, 2002.
[33] Piotr R Luszczek, David H Bailey, Jack J Dongarra, Jeremy Kepner, Robert F Lucas, Rolf Rabenseifner, and Daisuke Takahashi. The hpc challenge (hpcc) benchmark suite. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 213. Citeseer, 2006.
[34] Matthias S Müller, Matthijs van Waveren, Ron Lieberman, Brian Whitney, Hideki Saito, Kalyan Kumaran, John Baron, William C Brantley, Chris Parrott, Tom Elken, et al. Spec mpi2007—an application benchmark suite for parallel systems using mpi. *Concurrency and Computation: Practice and Experience*, 22(2):191–205, 2010.
[35] Kurt B. Ferreira, Patrick Bridges, and Ron Brightwell. Characterizing application sensitivity to OS interference using kernel-level noise injection. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, SC '08, pages 19:1–19:12, Piscataway, NJ, USA, 2008. IEEE Press.
[36] S. Vinoski. Advanced Message Queuing Protocol. *Internet Computing, IEEE*, 10(6):87–89, 2006.
[37] И.А. Дудина, А.О. Кудрявцев. Разработка и реализация облачного планировщика, учитывающего топологию коммуникационной среды при высокопроизводительных вычислениях. *Труды Института системного программирования РАН, том 24*, принято к публикации, 2013.
[38] Katarzyna Keahey and Tim Freeman. Contextualization: Providing One-Click Virtual Clusters. In *Proceedings of the 2008 Fourth IEEE International Conference on eScience*, ESCIENCE '08, pages 301–308, Washington, DC, USA, 2008. IEEE Computer Society.

HPC cloud system design and implementation

Kudryavtsev A.O.: alexk@ispras.ru, ISP RAS, Moscow, Russia
Koshelev V.K.: vedun@ispras.ru, ISP RAS, Moscow, Russia
Izbyshev A.O.: izbyshev@ispras.ru, ISP RAS, Moscow, Russia
Dudina I.A.: eupharina@ispras.ru, ISP RAS, Moscow, Russia
Kurmangaleev Sh.F.: kursh@ispras.ru, ISP RAS, Moscow, Russia
Avetisyan A.I.: arut@ispras.ru, ISP RAS, Moscow, Russia
Ivannikov V.P.: ivan@ispras.ru, ISP RAS, Moscow, Russia
Velikhov V.E.: velikhovve@kiae.ru, NRC KI, Moscow, Russia
Ryabinkin E.A.: rea@grid.kiae.ru, NRC KI, Moscow, Russia

Abstract. There is pronounced interest to cloud computing in the scientific community. However, current cloud computing offerings are rarely suitable for high-performance computing, in large part due to an overhead level of underlying virtualization components. The purpose of this paper is to propose a design and implementation of a cloud system that possesses a small enough overhead level to allow it to be practically used for a wide range of scientific workloads. First, we describe requirements for the desired system and classify workloads to identify those that are practical to transfer to the cloud. Then, we review related work. Finally, we describe our cloud system, "Virtual Supercomputer", which is based on the OpenStack cloud infrastructure and KVM/QEMU hypervisor. Most components of the original infrastructure were modified to satisfy the requirements. In particular, we tuned KVM/QEMU and the host operating system, introduced the concept of virtual machine groups and implemented a topology-aware scheduler to reduce communication overhead between network nodes belonging to the same virtual machine group. Also, we implemented a proof-of-concept web service on top of our system that allows to use OpenFOAM toolbox in software-as-a-service manner. The main result of our work is that "Virtual Supercomputer" achieved the overhead level of less than 10% on industry standard benchmarks when using up to 1024 processor cores. We deem this overhead level as acceptable for practical use.

Keywords: cloud computing; virtualization; hypervisors; high-performance computing; parallel computing

References

[1]. Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. Above the Clouds: A Berkeley View of Cloud Computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009.

[2]. Peter Mell and Tim Grance. The NIST Definition of Cloud Computing. Technical report, July 2009.

[3]. xCAT – Extreme Cloud Administration Toolkit. http://xcat.sourceforge.net/, 1999. [Online; accessed 14-May-2013].

[4]. Moab Suite. http://www.adaptivecomputing.com/products/. [Online; accessed 14-May-2013].
[5]. D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The eucalyptus open-source cloud-computing system. In Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on, pages 124–131, 2009.
[6]. Katarzyna Keahey, Ian Foster, Tim Freeman, and Xuehai Zhang. Virtual workspaces: Achieving quality of service and quality of life in the grid. Scientific Programming, 13(4):265–275, 2005.
[7]. J Fontán, T Vázquez, L Gonzalez, Ruben S Montero, and IM Llorente. Opennebula: The open source virtual machine manager for cluster computing. In Open Source Grid and Cluster Software Conference, 2008.
[8]. Apache CloudStack: Open Source Cloud Computing. http://cloudstack.apache.org/, 2010. [Online; accessed 14-May-2013].
[9]. OpenStack Open Source Cloud Computing Software. http://www.openstack.org/, 2010. [Online; accessed 14-May-2013].
[10]. Amazon Elastic Compute Cloud (Amazon EC2). http://aws.amazon.com/ec2/, 2006. [Online; accessed 14-May-2013].
[11]. Windows Azure: Microsoft's Cloud Platform. http://www.windowsazure.com/en-us/, 2010. [Online; accessed 14-May-2013].
[12]. Google Compute Engine. https://cloud.google.com/products/compute-engine, 2012. [Online; accessed 14-May-2013].
[13]. Lavanya Ramakrishnan, Piotr T. Zbiegel, Scott Campbell, Rick Bradshaw, Richard Shane Canon, Susan Coghlan, Iwona Sakrejda, Narayan Desai, Tina Declerck, and Anping Liu. Magellan: experiences from a science cloud. In Proceedings of the 2nd international workshop on Scientific cloud computing, ScienceCloud '11, pages 49–58, New York, NY, USA, 2011. ACM.
[14]. G. von Laszewski, G.C. Fox, Fugang Wang, A.J. Younge, A. Kulshrestha, G.G. Pike, W. Smith, J. Vöckler, R.J. Figueiredo, J. Fortes, and K. Keahey. Design of the futuregrid experiment management framework. In Gateway Computing Environments Workshop (GCE), 2010, pages 1–10, 2010.
[15]. Bcfg2 – A Configuration Management System. http://trac.mcs.anl.gov/projects/bcfg2/, 2004. [Online; accessed 14-May-2013].
[16]. Paul Marshall, Henry Tufo, Kate Keahey, David LaBissoniere, and H.M. Woitaszek. Architecting a large-scale elastic environment – recontextualization and adaptive cloud services for scientific computing. In Proceedings of the 7th International Conference on Software Paradigm Trends (ICSOFT), Rome, Italy, 2012
[17]. CEI Elastic Processing Unit (EPU) Services and Agents. https://github.com/ooici/epu, 2011.[Online; accessed 14-May-2013].
[18]. TORQUE Resource Manager. http://www.adaptivecomputing.com/products/open-source/torque/, 2003. [Online; accessed 14-May-2013].
[19]. Chef Configuration Management. http://www.adaptivecomputing.com/products/, 2009. [Online; accessed 14-May-2013].
[20]. Kate Keahey, Patrick Armstrong, John Bresnahan, David LaBissoniere, and Pierre Riteau. Infrastructure outsourcing in multi-cloud environment. In Proceedings of the 2012 workshop on Cloud services, federation, and the 8th open cirrus summit, FederatedClouds '12, pages 33–38, New York, NY, USA, 2012. ACM.

[21]. Jens-Sönke Vöckler, Gideon Juve, Ewa Deelman, Mats Rynge, and Bruce Berriman. Experiences using cloud computing for a scientific workflow application. In Proceedings of the 2nd international workshop on Scientific cloud computing, ScienceCloud '11, pages 15–24, New York, NY, USA, 2011. ACM.
[22]. Ewa Deelman, Gurmeet Singh, Mei-Hui Su, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Karan Vahi, G Bruce Berriman, John Good, Anastasia Laity, Joseph C. Jacob, and Daniel S. Katz. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. Scientific Programming, 13(3):219–237, 2005.
[23]. K. Keahey, M. Tsugawa, A. Matsunaga, and J. A B Fortes. Sky computing. Internet Computing, IEEE, 13(5):43–51, 2009.
[24]. Sage A Weil, Scott A Brandt, Ethan L Miller, Darrell DE Long, and Carlos Maltzahn. Ceph: A scalable, high-performance distributed file system. In Proceedings of the 7th symposium on Operating systems design and implementation, pages 307–320. USENIX Association, 2006.
[25]. Avi Kivity, Yaniv Kamay, Dor Laor, Uri Lublin, and Anthony Liguori. kvm: the linux virtual machine monitor. In Proceedings of the Linux Symposium, volume 1, pages 225–230, 2007.
[26]. Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. ACM SIGOPS Operating Systems Review, 37(5):164–177, 2003.
[27]. Fabrice Bellard. Qemu, a fast and portable dynamic translator. USENIX, 2005.
[28]. Kudryavtsev A.O., Koshelev V.K., Avetisyan A.I. Perspektivy virtualizatsii vysokoproizvoditel'nykh sistem arkhitektury x64 [The Prospects for Virtualization of High Performance x64 Systems]. Trudy ISP RAN [The Proceedings of ISP RAS], 2012, vol. 22, pp. 189–209 (in Russian).
[29]. Alexander Kudryavtsev, Vladimir Koshelev, and Arutyun Avetisyan. Modern HPC cluster virtualization using KVM and palacios. In High Performance Computing (HiPC), 2012 19th International Conference on, pages 1–9, 2012.
[30]. Alexander Kudryavtsev, Vladimir Koshelev, Boris Pavlovic, and Arutyun Avetisyan. Virtualizing HPC applications using modern hypervisors. In Proceedings of the 2012 workshop on Cloud services, federation, and the 8th open cirrus summit, FederatedClouds '12, pages 7–12, New York, NY, USA, 2012. ACM.
[31]. Kudryavtsev A.O., Koshelev V.K., Izbyshev A.O., Avetisyan A.I. Vysokoproizvoditel'nye vychisleniya kak oblachnyj servis: klyuchevye problemy [High Performance Computing as a Cloud Service: Key Issues]. Trudy mezhdunarodnoj nauchnoj konferentsii PaVT'2013 [The Proceedings of International Conference PCT'2013], 2013, pp. 432–438 (in Russian).
[32]. Rob F Van der Wijngaart and Parkson Wong. Nas parallel benchmarks version 2.4. Technical report, NAS technical report, NAS-02-007, 2002.
[33]. Piotr R Luszczek, David H Bailey, Jack J Dongarra, Jeremy Kepner, Robert F Lucas, Rolf Rabenseifner, and Daisuke Takahashi. The hpc challenge (hpcc) benchmark suite. In Proceedings of the 2006 ACM/IEEE conference on Supercomputing, page 213. Citeseer, 2006.
[34]. Matthias S Müller, Matthijs van Waveren, Ron Lieberman, Brian Whitney, Hideki Saito, Kalyan Kumaran, John Baron, William C Brantley, Chris Parrott, Tom Elken, et al. Spec mpi2007—an application benchmark suite for parallel systems using mpi. Concurrency and Computation: Practice and Experience, 22(2):191–205, 2010.

[35]. Kurt B. Ferreira, Patrick Bridges, and Ron Brightwell. Characterizing application sensitivity to OS interference using kernel-level noise injection. In Proceedings of the 2008 ACM/IEEE conference on Supercomputing, SC '08, pages 19:1–19:12, Piscataway, NJ, USA, 2008. IEEE Press.
[36]. S. Vinoski. Advanced Message Queuing Protocol. Internet Computing, IEEE, 10(6):87–89, 2006.
[37]. Dudina I.A., Kudryavtsev A.O., Gaissaryan S.S. Razrabotka i realizatsiya oblachnogo planirovshhika, uchityvayushhego topologiyu kommunikatsionnoj sredy pri vysokoproizvoditel'nykh vychisleniyakh [Topology-aware cloud scheduling for HPC]. Trudy ISP RAN [The Proceedings of ISP RAS], 2013, vol. 24, pp. 189–209 (in Russian).
[38]. Katarzyna Keahey and Tim Freeman. Contextualization: Providing One-Click Virtual Clusters. In Proceedings of the 2008 Fourth IEEE International Conference on eScience, ESCIENCE '08, pages 301–308, Washington, DC, USA, 2008. IEEE Computer Society.