УДК 004.457+004.031.2+004.382.2
DOI: 10.15827/2311-6749.18.4.8

МЕТОДЫ И СРЕДСТВА СОВМЕЩЕНИЯ ПОТОКОВ ЗАДАНИЙ ОТ ОБЛАЧНЫХ ПЛАТФОРМ И МЕНЕДЖЕРОВ УПРАВЛЕНИЯ РЕСУРСАМИ СУПЕРКОМПЬЮТЕРА

О.С. Аладышев, к.т.н., aladysjev@jscc.ru;
А.В. Баранов, к.т.н., доцент, antbar@mail.ru; А.П. Овсянников, apo@jscc.ru
(Межведомственный суперкомпьютерный центр Российской академии наук – филиал ФГУ «Федеральный научный центр Научно-исследовательский институт системных исследований Российской академии наук», г. Москва, 119334, Россия);
Г.А. Балаян, balayan_cpp@mail.ru; В.С. Синицин, v.s.sinitsin@yandex.ru
(Московский физико-технический институт (государственный университет), г. Москва, 117303, Россия)

В статье рассмотрен подход к реализации облачного сервиса высокопроизводительных вычислений, обеспечивающий совмещение двух потоков суперкомпьютерных заданий: поступающего от облачной платформы и поступающего от менеджера управления ресурсами суперкомпьютера.

Авторами предложен метод совмещения потоков заданий, заключающийся в представлении менеджера управления ресурсами в виде гипервизора. Метод был реализован в Межведомственном суперкомпьютерном центре РАН для облачной платформы OpenStack, в качестве менеджера управления ресурсами суперкомпьютера выступила отечественная система управления прохождением параллельных заданий. Представление системы в виде гипервизора было осуществлено путем разработки драйвера для библиотеки libvirt, используемой платформой OpenStack для распределения виртуальных машин по доступным вычислительным ресурсам.

Рассмотренный в статье подход обеспечивает интеграцию существующей системы управления суперкомпьютером в стандартный стек ПО облачных вычислений и позволяет избежать больших накладных расходов на виртуализацию, а также сохранить традиционный порядок работы пользователей суперкомпьютерных центров при переходе к облачным вычислениям.

Ключевые слова: *высокопроизводительные вычисления, облачные вычисления, облачная платформа, OpenStack, libvirt, collectd, СУППЗ.*

Число суперкомпьютерных пользователей с момента появления суперЭВМ непрерывно растет. Сегодня в распоряжении у пользователя может быть даже несколько вычислительных установок с различными архитектурами. Несмотря на различия в архитектуре, в большинстве суперкомпьютерных центров принят примерно одинаковый порядок работы. Зарегистрированный пользователь получает учетную запись и пароль, которые позволяют осуществлять удаленный доступ к суперкомпьютерным ресурсам и сервисам, как правило, по протоколу ssh [1]. Современный суперкомпьютер кластерного типа представляет собой вычислительную установку, состоящую из многопроцессорных вычислительных модулей, объединенных высокоскоростной сетью обмена данными типа Infiniband или Intel Omni-Path в единое решающее поле. Традиционно доступ пользователей к ресурсам суперкомпьютера осуществляется через специально выделенные модули – front-end узлы. На этих модулях пользователь формирует так называемое вычислительное задание – информационный объект, включающий параллельную программу, исходные данные и свои требования к выделяемым ресурсам (сколько вычислительных модулей или процессорных ядер и на какое время требуется программе). Распределением ресурсов для вычислительных заданий занимается специальный менеджер управления ресурсами (workload manager), часто *называемый системой управления заданиями* (СУЗ), который работает на управляющем или front-end модуле суперкомпьютера. Известными и широко распространенными подобными менеджерами являются PBS [2], SLURM [3], Moab [4], а также отечественная *система управлением прохождением параллельных заданий* (СУППЗ) [5]. Менеджеры обрабатывают входящий поток заданий, ведут очередь, выделяют ресурсы для новых заданий и освобождают от старых, ведут статистику использования ресурсов.

После подготовки задания и постановки в очередь СУЗ пользователю необходимо дождаться его старта и завершения, после чего забрать результаты. При работе на нескольких суперкомпьютерных установках пользователь вынужден повторять однотипные действия для каждой из них, что далеко не всегда удобно и эффективно. При этом очень часто пользователи сами не разрабатывают новые программы, а используют для расчетов готовые прикладные пакеты. Установка и настройка пакета для нескольких суперкомпьютерных установок – трудоемкий процесс, которого пользователь всегда желает избежать.

Решить проблему с неоднородностью суперкомпьютеров можно через введение дополнительного уровня абстракции – облачного сервиса, который обеспечит единый интерфейс управления. Но при этом необходимо учитывать и традиционный порядок работы пользователей через СУЗ. В *Межведомственном суперкомпьютерном центре Российской академии наук* (МСЦ РАН) был рассмотрен один из вариантов решения этой задачи [6], представлявший собой веб-приложение, основанное на специально подобранном программном стеке. Созданный макет обеспечивал взаимодействие веб-приложения с менеджерами нескольких суперкомпьютеров, совмещая потоки заданий от облачного сервиса вида SaaS и от традиционного СУЗ. Решение [6] получилось узкоспециализированным, возникли трудности с его модификацией, разработанный сервис был несовместим с известными стандартными решениями, что фактически делало невозможным его развитие.

В существующих решениях в области организации облачных вычислений, как и в суперкомпьютерных расчетах, подразумевается определенный типовой порядок действий пользователя. Пользователь при помощи клиентских инструментов инициирует запрос на создание/использование необходимых ему *виртуальных машин* (ВМ). С помощью специального компонента – контроллера облака – облачная платформа подбирает наиболее подходящие вычислительные ресурсы, на которых начинается выполнение ВМ. При подборе вычислительных ресурсов и запуске на них ВМ контроллер облака непосредственно взаимодействует с гипервизорами физических вычислительных модулей (серверов), анализируя текущую загруженность решающего поля и балансируя вычислительную нагрузку. Другими словами, решающее поле, которое в суперкомпьютерах управляется СУЗ, должно быть передано под полное управление облачной платформе.

Таким образом, прямое использование СУЗ для обработки потока облачных заданий, как и прямое использование облачных платформ для выполнения традиционных суперкомпьютерных вычислений, невозможно. Исследованию и разработке методов и средств объединения двух потоков заданий, облачного и суперкомпьютерного, и посвящена настоящая работа. Ее целью является поиск перспективного решения, максимально совместимого с общепринятыми стандартами в облачных вычислениях и позволяющего распределять задания из облачного потока на суперкомпьютерные ресурсы, не ломая типовой порядок их использования.

Современное состояние исследований в области высокопроизводительных облачных вычислений

Концепция облачных вычислений подразумевает, что конечному пользователю по требованию и на определенное время будет предоставлен удобный сетевой доступ к некоторому набору настраиваемых вычислительных ресурсов: подмножеству узлов решающего поля суперкомпьютера, сетям передачи данных, устройствам хранения данных, программным приложениям и пакетам. В терминологии облачных вычислений предоставляемый пользователю доступ к ресурсам называется услугой или сервисом, а сам процесс предоставления сервиса – обслуживанием. Среди основных моделей обслуживания выделяют ПО как услугу (SaaS), платформу как услугу (PaaS) и инфраструктуру как услугу (IaaS).

К обязательным характеристикам облачных вычислений обычно относят следующие:

− самообслуживание по требованию (услуга предоставляется по запросу пользователя, который самостоятельно определяет ее объем и продолжительность);

− универсальный доступ по сети (услуга доступна пользователям по сети вне зависимости от используемого оконечного устройства (терминала, клиента));

− динамическое перераспределение вычислительных мощностей в условиях постоянного изменения нагрузок за счет объединения ресурсов в единый пул;

− эластичность (объем предоставляемых услуг по требованию пользователя может быть в любой момент времени изменен как в большую, так и в меньшую сторону);

− оплата пользователем только фактически потребленных ресурсов.

С точки зрения пользователя, облака упрощают доступ к необходимым сервисам, делая прозрачным путь от запроса к вычислительным ресурсам. При этом пользователь может существенно сэкономить как на организации вычислений, так и на том, что оплатит лишь фактически потребленные ресурсы. Выгода владельца ресурсов состоит в кратном увеличении прибыли за счет более эффективной утилизации ресурсов и консолидации затрат на их управление. Для эффективной организации облачных вычислений на базе имеющихся ресурсов создается специальная инфраструктура с помощью так называемых облачных платформ и систем виртуализации, как показано на рисунке 1.

Нетрудно заметить, что любая СУЗ суперкомпьютера в рамках своей *суперкомпьютерной вычислительной установки* (СВУ) в некоторой степени абстракции обладает всеми перечисленными характеристиками облачных вычислений:

− самообслуживание по требованию и эластичность (зарегистрированный пользователь для непосредственного производства расчетов формирует и направляет в СУЗ вычислительное задание, самостоятельно задавая необходимые число узлов решающего поля и время для выполнения задания);

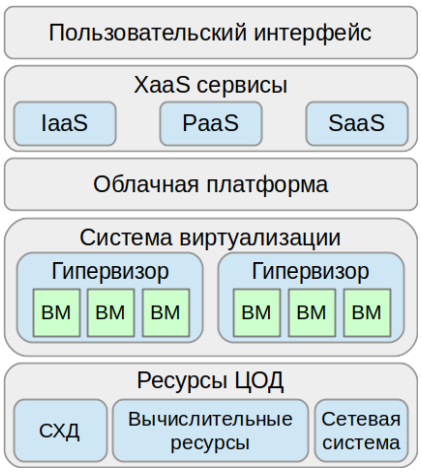

Рис. 1. Программный стек при стандартной организации облачных вычислений

− универсальный доступ по сети (зарегистрированный пользователь имеет круглосуточный удаленный доступ к суперкомпьютеру и может использовать для доступа произвольное оконечное устройство (терминал));

− менеджер управления ресурсами суперкомпьютера динамически выделяет узлы решающего поля для каждого задания, обеспечивая полноту и равномерную загрузку решающего поля;

− биллинговая подсистема СУЗ ведет точный учет фактического потребления пользователями суперкомпьютерных ресурсов.

Легко заметить по перечисленным далее признакам и принципиальные отличия высокопроизводительных вычислений от облачных, которые не позволяют осуществлять их простую интеграцию:

− уникальность вычислительных ресурсов суперкомпьютеров не всегда поддерживается универсальными гипервизорами;

− обеспечивая минимальное время выполнения своего задания и максимальную эффективность использования вычислительных устройств, пользователь суперкомпьютера предпочитает иметь непосредственный, а не опосредованный системой виртуализации доступ к суперкомпьютерным ресурсам;

− для широкого класса суперкомпьютерных задач использование виртуализации не оправдано, так как она требует существенных накладных расходов на вычисления [7, 8].

Так, в работе [9] авторы рассматривают облачную инфраструктуру, созданную на базе вычислительных ресурсов морально устаревшего суперкомпьютера, который был заменен системой нового поколения. На базе «старых» ресурсов была организована масштабируемая вычислительная инфраструктура для консолидации компьютерного оборудования, используемого в целях разработки, отладки и развертывания ПО, а также дидактической поддержки образовательных курсов. Тем самым было продемонстрировано, что облачные вычисления менее требовательны к вычислительным ресурсам, а высокопроизводительные – напротив, постоянно требуют усовершенствования производительности.

Тем не менее, попытки интеграции облачных и высокопроизводительных вычислений осуществляются постоянно на протяжении многих лет как за рубежом, так и в России. Среди исследований и разработок в этой области можно выделить три направления.

Первое направление связано с применением в суперкомпьютерах технологий виртуализации как основы организации облачных вычислений. Исследования по этому направлению проводятся в основном с целью снижения накладных расходов на виртуализацию. Например, в работе [10] рассматривается методология точного измерения накладных расходов на старт ВМ виртуального кластера для распространенных облачных платформ. Успехи в этом направлении связаны с технологиями контейнерной виртуализации, привносящей наименьшие накладные расходы в вычислительный процесс [11, 12]. В [13] не только содержится достаточно полный обзор исследований в области контейнерной виртуализации, но и предлагаются решения по представлению вычислительных заданий в виде контейнеров в системе управления заданиями SLURM.

По второму направлению ведутся разработки облачных сервисов для высокопроизводительных вычислений на основе построения собственных облачных платформ [14], в том числе и с использованием программных комплексов организации добровольных вычислений [15].

Третье направление связано с исследованиями в области построения и применения облачных платформ для высокопроизводительных вычислений. В настоящее время существуют несколько программных продуктов для построения инфраструктуры частного, публичного или гибридного облака, среди которых в научных исследованиях выделяется свободно распространяемая платформа OpenStack [16], предназначенная для построения облаков с моделью предоставления облачных вычислений IaaS. Например, работа [17] посвящена построению на базе платформы OpenStack прототипа системы с виртуальными кластерами, каждый из которых имеет свою виртуализованную сеть Infiniband. В работе [18] описывается концепция динамического разделения вычислительного поля суперкомпьютерного кластера на стандартно управляемую часть и виртуализированный на базе OpenStack раздел и интеграцию данного механизма с менеджером управления ресурсами LSF. Наиболее близкой к теме настоящей статьи является работа [19], в которой предлагается в облаке, управляемом платформой OpenStack, выделять по запросу пользователя виртуальный кластер не из ВМ, а из части физических узлов, то есть некий гибридный подход к построению облака для высокопроизводительных вычислений. В работах [8, 20] описаны ос-

новные проблемы, возникающие при переносе высокопроизводительных вычислений в облако, и рассмотрен подход к организации высокопроизводительного облачного сервиса с использованием виртуализации. Отмечается основное препятствие – высокие накладные расходы, которые даже при достаточно глубоких исследованиях авторов [20] составили не менее 10 %. Схожий опыт демонстрирует работа [21], где попытка внедрения облачной платформы OpenStack на вычислительных узлах суперкомпьютера Cray привела к накладным расходам в 20 %. Кроме этого, для эффективного управления высокопроизводительными ресурсами авторам работы [20] потребовалось создание собственного планировщика [22], учитывающего топологию коммуникационной среды. Как показывает опыт [6], собственные разработки при всей их эффективности на узкоспециализированном участке исследований часто оказываются несовместимыми с общепринятыми стандартными подходами.

В отличие от рассмотренных работ авторы настоящей статьи видят свою задачу в выработке такого решения, которое удовлетворяло бы следующим требованиям:

- не ломало исторически сложившийся порядок обслуживания пользователей суперкомпьютеров;
- ограничивало применение гипервизорной виртуализации с целью недопущения больших накладных расходов и обеспечения непосредственного доступа пользователей к уникальным возможностям суперкомпьютерного оборудования;
- максимально использовало стандартные механизмы облачных вычислений (рис. 1).

Основным методом, позволяющим удовлетворить перечисленные требования, авторам видится представление менеджера управления ресурсами суперкомпьютера в виде гипервизора, что позволит встроить его в стандартный стек облачных решений.

Представление менеджера управления ресурсами в качестве гипервизора

Авторы рассмотрели ряд облачных платформ, свободно распространяемых в исходных текстах. Среди таких платформ следует выделить OpenStack [16], OpenNebula [23–25], Eucalyptus [25, 26], CloudStack [27], Nimbus [24, 26]. Сравнительный анализ некоторых облачных платформ приведен в статьях [24, 25].

В настоящей работе внимание направлено на платформу OpenStack, которая, по мнению авторов, наиболее вероятно станет стандартом в области построения частных облаков. Об этом свидетельствует, среди прочего, широкое использование OpenStack крупными компьютерными корпорациями, в частности IBM [28].

Со многими облачными платформами OpenStack объединяет специализированная библиотека libvirt [29]. Библиотека управления виртуализацией libvirt представляет собой дополнительный уровень абстракции, позволяющий работать с ВМ под управлением различных гипервизоров, и поддерживает большое количество гипервизоров, таких как KVM, Xen, VMware ESX, VMware Workstation. Библиотека входит в состав множества программных продуктов, а также является одним из основных средств по управлению гипервизорами в облачной платформе OpenStack [20, 21] (рис. 2).

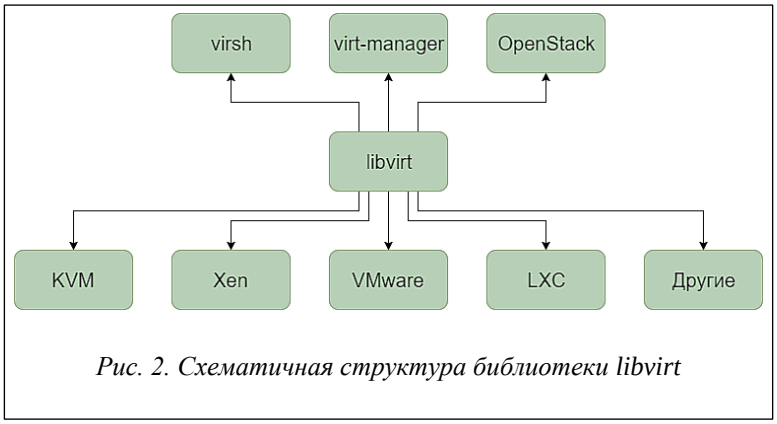

Рис. 2. Схематичная структура библиотеки libvirt

Для запуска ВМ библиотека libvirt использует описание ее ресурсов в виде XML-файла, в котором указываются название ВМ, размер оперативной памяти, путь к файлу образа ВМ в системе хранения данных, количество процессорных ядер и т.д. Библиотека libvirt входит в состав компонента nova-compute облачной платформы OpenStack и используется последней для непосредственного запуска ВМ с помощью различных гипервизоров. При запуске ВМ обращение к библиотеке происходит следующим образом.

1. Опрос гипервизоров через библиотеку libvirt на вычислительных узлах на наличие свободных ресурсов для запуска ВМ.
2. После выбора узла выборка образа ВМ из хранилища образов.
3. Копирование образа ВМ на выбранный узел.
4. Генерация XML-описания ВМ для запуска с помощью библиотеки libvirt.
5. Запуск ВМ и контроль ее состояния с помощью библиотеки libvirt.

Иерархическая схема стека ПО платформы OpenStack, работающей через библиотеку libvirt, представлена на рисунке 3.

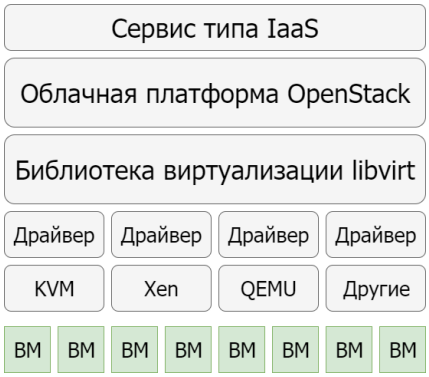

Рис. 3. Стек ПО облачного сервиса типа IaaS на базе платформы OpenStack и библиотеки libvirt

В состав библиотеки libvirt входит отдельный драйвер для каждого гипервизора. При этом сохраняется возможность реализации и встраивания в библиотеку собственного драйвера для своего гипервизора. Основная идея настоящей работы заключается в представлении менеджера управления ресурсами суперкомпьютера, СУЗ, в виде некоторого специализированного гипервизора путем разработки соответствующего драйвера для библиотеки libvirt, что должно обеспечить прозрачную интеграцию СУЗ суперкомпьютера в стек ПО облачной платформы.

В качестве менеджера управления ресурсами выступила СУППЗ, для сопряжения которой с облачной платформой OpenStack был создан собственный драйвер для библиотеки libvirt. Драйвер преобразует команды управления ВМ облачной платформы в команды управления заданиями суперкомпьютера. Для этого были введены следующие абстракции.

1. СУППЗ является гипервизором в абстракциях облачного сервиса.
2. Задание является ВМ в абстракциях облачного сервиса. Как и у ВМ, у задания может быть несколько состояний: «в очереди», «выполняется», «не выполняется». Были введены аналогичные абстракции: «ВМ работает» – «задание выполняется», «ВМ не работает» – «задание не выполняется», «ВМ на паузе» – «задание в очереди».
3. Суперкомпьютерные ресурсы, требующиеся для выполнения задания, являются ресурсами ВМ. Были введены соответствующие абстракции: «число процессоров ВМ» – «число процессоров для задания», «размер оперативной памяти ВМ» – «максимальное время выполнения задания».

В качестве методов API библиотеки libvirt авторами были реализованы следующие функции:

- соединение драйвера с библиотекой;
- просмотр текущего списка суперкомпьютерных заданий;
- обновление списка заданий;
- запуск задания;
- завершение задания.

В процессе работы выяснилось, что отсутствует обратная совместимость версий библиотеки libvirt, в связи с чем разработанный для более ранней версии драйвер не может быть непосредственно подключен к библиотеке более поздней версии. Для преодоления несовместимости авторами была разработана специальная методика создания драйвера библиотеки libvirt, заключающаяся в следующем.

1. Для разработки стороннего драйвера для библиотеки libvirt необходимо ознакомиться с изменениями API libvirt, приведенными в документации новой версии.
2. Для облегчения разработки драйвера необходимо изменять уже существующий драйвер. В качестве основы рекомендуется использовать два драйвера – VMware и Mock (тестовый драйвер-заглушка). В драйвере VMware реализован минимальный функционал для управления ВМ, в Mock-драйвере, напротив, собран весь возможный функционал.
3. Для реализации драйвера необходимо переопределить методы и атрибуты, входящие в состав абстрактной структуры virHypervisorDriver. Общий принцип работы всех методов заключается в вызове управляющих команд СУППЗ с помощью функции virRun(), перенаправлении стандартного вывода и применении синтаксического анализа результатов вызова управляющих команд СУППЗ, например, команды просмотра состояния очереди mqinfo [5].
4. Сторонние средства отладки для создания драйверов библиотекой libvirt не предусмотрены, поэтому для отладки необходимо использовать отладочную печать.
5. Для проверки работоспособности драйвера рекомендуется использовать оболочку virsh (рис. 2), исходные коды которой находятся в библиотеке libvirt. Для корректного взаимодействия оболочки с новым драйвером необходимо устанавливать ее вместе с библиотекой.
6. Внедрение стороннего драйвера в уже установленную библиотеку libvirt невозможно. После получения исходных текстов libvirt требуемой версии для стороннего драйвера необходимо создать отдельный каталог, а также произвести соответствующие записи в файлах configure и Makefile таким образом, чтобы вместе с установкой библиотеки происходили компиляция и сборка стороннего драйвера. Во время конфигурации и сборки библиотеки libvirt в нее будет добавлен сторонний драйвер.

Для возможности сопряжения платформы OpenStack и СУППЗ, помимо внедрения собственного libvirt-драйвера, была произведена модификация исходного кода платформы, предназначенного для работы

с libvirt. Модификация потребовалась для снятия ограничения по использованию сторонних драйверов, так как платформа OpenStack изначально предполагает использование только гипервизоров KVM, QEMU, Parallels, LXC. Изменения были внесены в файлы, реализующие генерацию XML-описания и подключение к гипервизорам KVM, LXC, QEMU и Parallels, в эти файлы была добавлена возможность подключения платформы OpenStack к СУППЗ.

Облачный сервис для высокопроизводительных вычислений на базе платформы OpenStack и СУППЗ

В основе сервиса лежат СВУ, каждая из которых управляется СУППЗ. Облачная платформа осуществляет управление СУППЗ с помощью библиотеки libvirt через драйвер СУППЗ. На базе платформы строится интерфейс сервиса типа IaaS, соответствующий стек ПО представлен на рисунке 4.

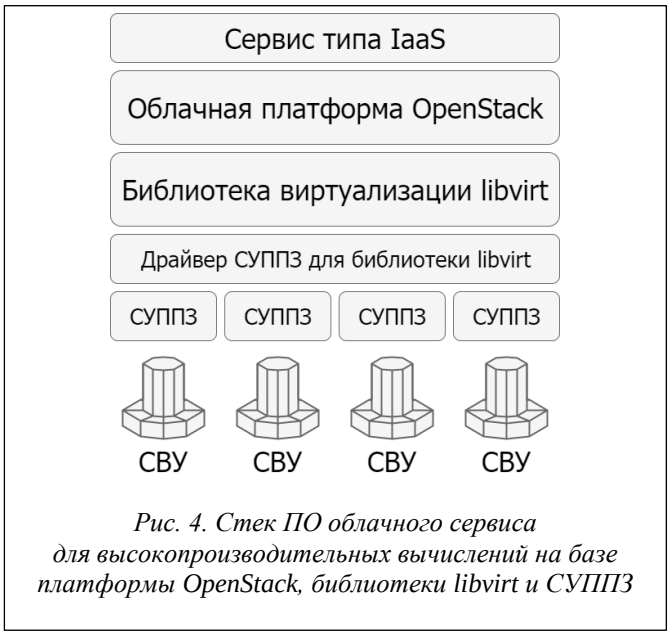

Рис. 4. Стек ПО облачного сервиса для высокопроизводительных вычислений на базе платформы OpenStack, библиотеки libvirt и СУППЗ

Структура разработанного авторами облачного сервиса представлена на рисунке 5. Облачный сервис состоит из двух частей: управляющей и вычислительной. В управляющей части размещаются основные компоненты облачной платформы OpenStack:

− контроллер облачной платформы OpenStack, осуществляющий обработку поступающих платформе команд и подготовку данных для запуска суперкомпьютерных заданий, а также отвечающий за балансировку нагрузки между вычислительными узлами;

− служба хранения образов платформы OpenStack, предназначенная для хранения файлов, необходимых для запуска суперкомпьютерного задания;

− служба сбора статистики облачной платформы OpenStack, предназначенная для сбора информации о состоянии вычислительных узлов (суперкомпьютеров), входящих в состав платформы;

− интерфейс доступа пользователей к платформе, осуществляющий трансляцию команды пользователей в программные вызовы облачной платформы OpenStack.

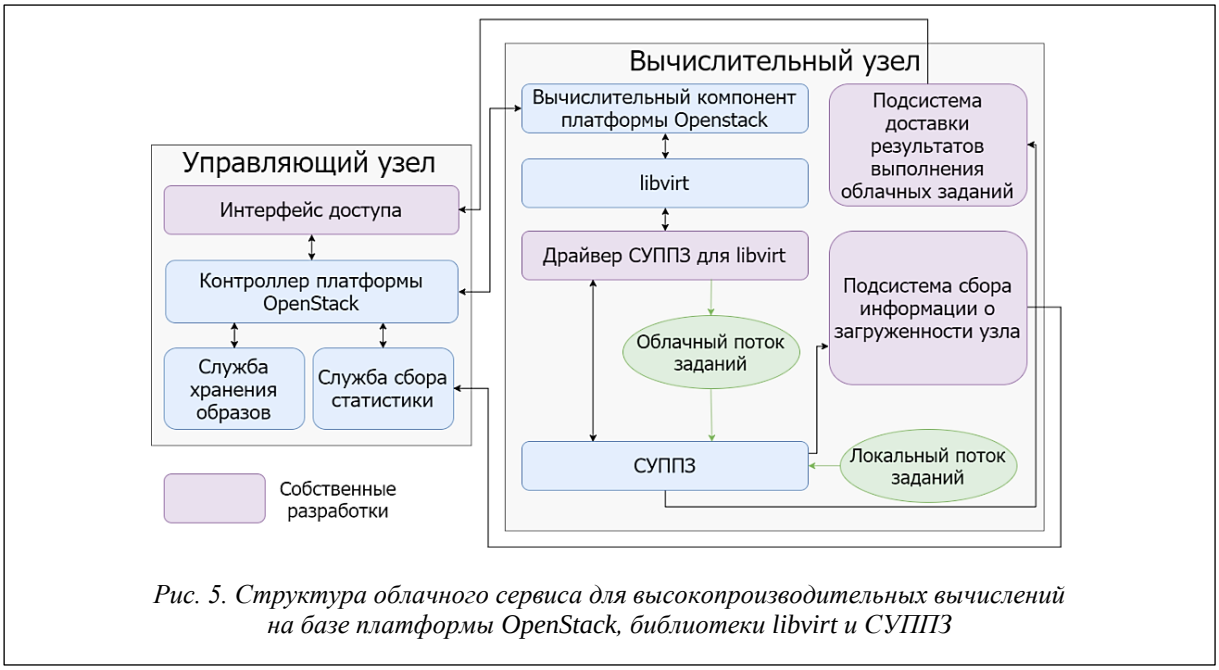

Рис. 5. Структура облачного сервиса для высокопроизводительных вычислений на базе платформы OpenStack, библиотеки libvirt и СУППЗ

Вычислительная часть может состоять из нескольких вычислительных узлов, каждый из которых представляет отдельный суперкомпьютер и имеет следующий состав:

− вычислительный компонент облачной платформы OpenStack, транслирующий команды облачной платформы OpenStack в вызовы библиотеки виртуализации libvirt;

− библиотека виртуализации libvirt, транслирующая вызовы библиотеки libvirt в вызовы драйвера гипервизора СУППЗ;

− СУППЗ, представленная в виде гипервизора и получающая от соответствующего драйвера команды управления суперкомпьютерными заданиями;

− подсистема доставки результатов облачных заданий на управляющий узел;

− подсистема сбора информации о загруженности вычислительного узла.

После организации взаимодействия платформы и СУППЗ был выявлен ряд существенных недостатков, связанных с недостаточно полным функционалом облачной платформы OpenStack. В частности, следует отметить два из них.

1. Невозможность получения результатов выполнения суперкомпьютерных заданий. В составе облачной платформы OpenStack отсутствуют механизмы непосредственной передачи файлов между вычислительными узлами, в качестве которых выступают суперкомпьютерные установки, и управляющим узлом.

2. Некорректный сбор информации о состоянии очереди суперкомпьютера. Базовые средства ведения статистики собирают информацию о потреблении ресурсов только пользователями, получившими доступ к СУППЗ через облачный сервис. Но в СУППЗ также может поступать поток заданий от локальных пользователей. В этом случае контроллер облачной платформы будет некорректно осуществлять выбор вычислительного узла для запуска заданий пользователей облачного сервиса, так как у него нет информации о загруженности вычислительной установки локальными заданиями.

Указанные недостатки потребовали разработки собственных программных решений, реализующих недостающий функционал. Основными требованиями к разрабатываемым решениям были определены следующие:

− решения не должны нарушать типовую архитектуру облачного сервиса, представленную на рисунке 3;

− реализация должна осуществляться на базе распространенного ПО, взаимодействие которого с облачной платформой должно осуществляться с помощью драйверов или плагинов.

Разработанные решения основываются на применении известной службы collectd [30], предназначенной для сбора, хранения и передачи информации о состоянии компьютерных систем и различного ПО. Инструмент collectd с успехом применяется [31, 32] для мониторинга облачных сред и инфраструктур. Сбор данных в collectd реализован с помощью плагинов, представляющих собой командные файлы-сценарии (скрипты). Служба collectd через указанный промежуток времени вызывает сценарии и разбирает информацию, помещаемую сценариями в стандартный поток вывода. В службе collectd имеется поддержка множества способов выдачи собранной информации (электронная почта, протокол SNMP, запись в CSV-файл и другие способы). Разработчиками облачной платформы OpenStack был реализован компонент, позволяющий помещать собранную collectd информацию в службу сбора статистики облачной платформы.

Для возможности сбора информации о СУППЗ был реализован собственный плагин, который периодически опрашивает состояние СУППЗ. Разработанный компонент позволяет сообщать информацию о загруженности суперкомпьютера заданиями локальных пользователей, что способствует корректной балансировке нагрузки контроллером облачной платформы OpenStack между разными суперкомпьютерами. Функциональная схема работы данного компонента представлена на рисунке 6.

В отличие от ВМ задания пользователей предполагают наличие результатов, которые после завершения задания должны быть доставлены на управляющий узел облачной платформы. Существующие компоненты платформы OpenStack не предоставляют такую возможность, поэтому авторами был создан программный модуль для СУППЗ, реализующий соответствующий функционал. Разработанный модуль основан на использовании командного сценария-эпилога, автоматически вызываемого СУППЗ после завершения задания.

Макет представленного на рисунке 5 облачного сервиса был реализован на суперкомпьютерных ресурсах МСЦ РАН. В состав макета вошли один управляющий узел OpenStack и два вычислительных узла, каждый из которых представлял отдельный сегмент суперкомпьютера МВС-100К под управлением СУППЗ. В платформе OpenStack сегменты суперкомпьютера были видны как вычислительные узлы с большим числом процессорных ядер, представлявших ядра суперкомпьютера, и со значительным объемом оперативной памяти, представлявшим емкость очереди заданий СУППЗ. Запросы к OpenStack через драйвер СУППЗ библиотеки libvirt трансформировались в команды постановки в очередь суперкомпьютерных заданий. Платформа OpenStack адекватно отображала загрузку вычислительных узлов с учетом поступающих напрямую в СУППЗ локальных заданий, автоматически производя балансировку нагрузки между двумя суперкомпьютерными сегментами.

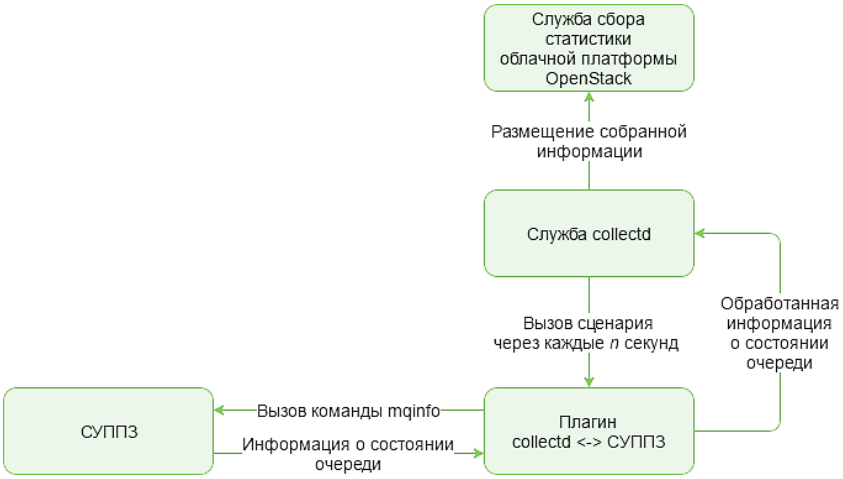

Рис. 6. Функциональная схема работы подсистемы сбора статистической информации

Заключение

Существующие исследования и разработки в области организации облачных сервисов для высокопроизводительных вычислений сталкиваются с двумя основными сложностями – высокими накладными расходами на гипервизорную виртуализацию и необходимостью передачи вычислительных ресурсов под управление облачной платформы. Последнее при переходе к облачным вычислениям порождает невозможность сохранения преемственности порядка работы пользователей с менеджерами управления ресурсами суперкомпьютеров. Для преодоления указанных трудностей необходимо организовать совмещение двух потоков заданий – поступающего из облачной платформы и локального потока заданий, поступающих непосредственно в суперкомпьютер через менеджер управления ресурсами. При этом предлагаемое решение должно быть максимально совместимо со стандартным программным стеком организации облачных вычислений. Задача авторов настоящей статьи состояла в поиске и разработке подобного решения.

Результатом исследований и разработок стал метод совмещения двух потоков заданий, заключающийся в представлении менеджера управления ресурсами суперкомпьютера в виде гипервизора. Основным средством реализации метода явилось создание собственного драйвера менеджера для библиотеки управления ВМ libvirt, на которую опирается большинство существующих облачных платформ. Создание драйвера позволило осуществить прозрачное включение менеджера управления ресурсами суперкомпьютера в стек ПО облачного сервиса.

Предложенный метод был реализован авторами в виде макета облачного сервиса для высокопроизводительных вычислений, развернутого на ресурсах установленного в МСЦ РАН суперкомпьютера МВС-100К. Макет основан на облачной платформе OpenStack, библиотеке libvirt с драйвером СУППЗ, используемой в МСЦ РАН в качестве менеджера управления ресурсами.

Статья подготовлена при поддержке грантов РФФИ № 18-07-01325 № 16-07-01098\18.

Литература

1. Barrett D., Silverman R., Byrnes R. SSH, the secure shell: the definitive guide. O'Reilly Media, 2009, 672 p.

2. Henderson R.L. Job scheduling under the Portable Batch System. In: Feitelson D.G., Rudolph L. (eds). JSSPP 1995, vol. 949. Springer, Berlin, Heidelberg, pp. 279–294. DOI: 10.1007/3-540-60153-8_34.

3. Yoo A.B., Jette M.A., Grondona M. (2003) SLURM: Simple Linux Utility for Resource Management. In: Feitelson D., Rudolph L., Schwiegelshohn U. (eds) Job Scheduling Strategies for Parallel Processing. JSSPP 2003. Lecture Notes in Computer Science, vol. 2862. Springer, Berlin, Heidelberg, pp. 44–60. DOI: 10.1007/10968987_3.

4. Moab HPC suite enterprise edition. URL: http://www.adaptivecomputing.com/products/hpc-products/moab-hpc-suite-enterprise-edition (дата обращения: 12.07.2018).

5. Система управления прохождением параллельных заданий (СУППЗ). Руководство программиста (пользователя). URL: http://www.jscc.ru/wp-content/uploads/2017/06/SUPPZ-user-guide-2016.pdf (дата обращения: 23.08.2018).

6. Баранов А.В., Зонов А.А. Вариант организации облачного сервиса для высокопроизводительных вычислений // Программные системы: теория и приложения. 2016. № 7. Т 3. С. 3–23. DOI: 10.25209/2079-3316-2016-7-3-3-23.

7. Аладышев О.С., Баранов А.В., Ионин Р.П., Киселев Е.А., Орлов В.А. Сравнительный анализ вариантов развертывания программных платформ для высокопроизводительных вычислений // Вестн. Уфимского гос. авиац. технич. ун-та. 2014. Т. 18. № 3. С. 295–300.

8. Кудрявцев А.О., Кошелев В.К., Избышев А.О., Аветисян А.И. Высокопроизводительные вычисления как облачный сервис: ключевые проблемы // Параллельные вычислительные технологии-2013 (ПаВТ'2013): сб. тр. Междунар. науч. конф. 2013. С. 432–438.

9. Ермаков Д.Г., Усталов Д.А. Экспериментальная среда облачных вычислений в Институте математики и механики УрО РАН // Программные продукты и системы. 2012. № 4. С. 110–115.

10. Jones M., et al. Scalability of VM provisioning systems. Proc. HPEC, 2016 IEEE. DOI: 10.1109/HPEC.2016.7761629.

11. Adufu T., Choi J., and Kim Y. Is container-based technology a winner for high performance scientific applications? Proc.17th APNOMS, 2015, pp. 507–510. DOI: 10.1109/APNOMS.2015.7275379.

12. Баранов А.В., Николаев Д.С. Использование контейнерной виртуализации в организации высокопроизводительных вычислений // Программные системы: теория и приложения. 2016. Т. 7. № 1. С. 117–134. DOI: 10.25209/2079-3316-2016-7-1-117-134.

13. Корнеев В.В., Николаев Д.С. Использование механизмов контейнерной виртуализации в высокопроизводительных вычислительных комплексах с системой планирования заданий SLURM // Программная инженерия. 2017. № 4. С. 147–160. DOI: 10.17587/prin.8.147-160.

14. Baranov A., Balashov N., Kutovsky N, and Semenov R. Cloud infrastucture of JINR. Computer Research and Modeling, 2015, no. 3, pp. 463–467.

15. Sukhoroslov O.V., Rubtsov A.O., Volkov S.Yu. Development of distributed computing applications and services with Everest cloud platform. Computer Research and Modeling, 2015, no. 3, pp. 593–599.

16. Маркелов А. Облачная операционная система OpenStack. Ч. 1. Введение // Системный администратор. 2015. № 4. С. 16–19.

17. Mauch V. Deployment of Virtual InfiniBand Clusters with Multi-tenancy for Cloud Computing. Proc. 6th Intern. Conf. CLOUD COMPUTING, 2015, pp. 66–69.

18. Ciaschini V., Dal Pra S., and dell'Agnello L. Dynamic partitioning as a way to exploit new computing paradigms: the cloud use case. Proc. 21st Intern. Conf. CHEP, 2015. DOI: 10.1088/1742-6596/664/2/022014.

19. Rad P., et al. Benchmarking bare metal cloud servers for HPC applications. Proc. CCEM, IEEE, 2015. DOI: 10.1109/CCEM.2015.13.

20. Кудрявцев А.О., Кошелев В.К., Избышев А.О., Дудина И.А., Курмангалеев Ш.Ф., Аветисян А.И., Иванников В.П., Велихов В.Е., Рябинкин Е.А. Разработка и реализация облачной системы для решения высокопроизводительных задач // Тр. ИСП РАН. 2013. Т. 24. С. 13–34.

21. Younge A.J., Pedretti K., Grant R.E., Gaines B.L. and Brightwell R. Enabling diverse software stacks on supercomputers using high performance virtual clusters. Proc. IEEE Intern. Conf. CLUSTER, Honolulu, HI, 2017, pp. 310–321. DOI: 10.1109/CLUSTER.2017.92.

22. Дудина И.А., Кудрявцев А.О., Гайсарян С.С. Разработка и реализация облачного планировщика, учитывающего топологию коммуникационной среды при высокопроизводительных вычислениях // Тр. ИСП РАН. 2013. Т. 24. С. 35–48.

23. Яремчук С. Проект OpenNebula. Решение для организации IaaS // Системный администратор. 2012. № 9. С. 14–20.

24. Богданов А.В., Е Мьинт Найнг. Сравнение нескольких платформ облачных вычислений // Вестн. СПб ун-та: Прикладная математика. Информатика. Процессы управления. 2013. № 2. С. 102–110.

25. Sempolinski P. and Thain D. A Comparison and Critique of Eucalyptus, OpenNebula and Nimbus. 2010 Proc. IEEE 2nd Intern. Conf. CLOUD COMPUTING, Indianapolis, IN, 2010, pp. 417–426. DOI: 10.1109/CloudCom.2010.42.

26. Nurmi D. et al. The Eucalyptus Open-Source Cloud-Computing System. 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, Shanghai, 2009, pp. 124–131. DOI: 10.1109/CCGRID.2009.93.

27. Руденко А. Платформа CloudStack 4. Архитектура, особенности и недостатки // Системный администратор. 2013. № 4. С. 10–15.

28. IBM ставит OpenStack в центр своей облачной стратегии // Открытые системы. СУБД. 2013. № 2. С. 4–11h.

29. libvirt: The virtualization API. URL: https://libvirt.org/ (дата обращения: 26.11.2018).

30. collectd – The system statistics collection daemon. URL: https://collectd.org (дата обращения: 26.11.2018).

31. Gorbil G., Garcia Perez D., Huedo Cuesta E. Principles of Pervasive Cloud Monitoring. In: Czachórski T., Gelenbe E., Lent R. (eds). Information Sciences and Systems 2014. Springer, Cham, pp. 117–124. DOI: 10.1007/978-3-319-09465-6_13.

32. Bellavista P., Giannelli C. and Mattetti M. A practical approach to easily monitoring and managing IaaS environments. 2013 IEEE Symposium on Computers and Communications (ISCC), Split, 2013, pp. 000016–000021. DOI: 10.1109/ISCC.2013.6754916.