Комбинированное использование высокопроизводительных ресурсов и грид-инфраструктур в рамках облачной платформы Everest*

О.В. Сухорослов

Институт проблем передачи информации Российской академии наук

Everest — облачная платформа, поддерживающая публикацию, выполнение и композицию вычислительных приложений в распределенной среде. Одной из отличительных черт платформы является возможность запуска приложений на произвольных комбинациях внешних вычислительных ресурсов, подключенных пользователями. В работе описывается интеграция платформы Everest с одиночными вычислительными ресурсами и грид-инфраструктурой EGI, а также рассматриваются проблемы, связанные с комбинированным использованием данных ресурсов для выполнения размещенных в Everest приложений.

1. Введение

В настоящее время остро стоит проблема объединения вычислительного потенциала доступных исследователям ресурсов в гетерогенные распределенные вычислительные среды (ГРВС) для решения сложных научных задач. Решение данной проблемы позволило бы многократно повысить производительность исследований и эффективность использования отдельных ресурсов. Существующие методы и технологии не в полной мере решают указанную проблему, поскольку в большинстве своём ориентированы только на определенные типы ресурсов и приложений.

Грид-технологии (Globus Toolkit, gLite, Unicore) позволяют организовать разделяемый доступ к множеству вычислительных ресурсов с возможностью автоматического распределения заданий между ресурсами. Однако данные технологии ориентированы только на определенный класс вычислительных ресурсов (кластеры и суперкомпьютеры), при этом далеко не все ресурсы доступны через грид-системы. Грид-технологии сложны в установке, использовании и сопровождении, что затрудняет их применение в рамках небольших научных проектов или для создания персональных ГРВС.

В свою очередь, технологии добровольных вычислений (BOINC, XtremWeb, OurGrid) ориентированы на интеграцию простаивающих ресурсов персональных компьютеров, что затрудняет их использование с другими типами ресурсов. Например, использование технологии BOINC на суперкомпьютере требует разработки дополнительных программных средств, таких как CluBORun. Исключением является система HTCondor, разработчики которой одними из первых предложили механизм интеграции с ресурсами суперкомпьютеров и грид-инфраструктур путем запуска заданий-агентов.

Другим подходом к построению ГРВС является применение метапланировщиков и других программных средств (HTCondor-G, Nimrod-G, GridWay, Ganga, DIRAC, PanDA), реализующих распределение и запуск заданий на заданном пуле вычислительных ресурсов. Данные средства позволяют сформировать ГРВС из произвольного набора ресурсов различных типов. При этом, как правило, не требуется установка дополнительного программного обеспечения (далее – ПО) на ресурсах, что значительно облегчает создание ГРВС. Вместе с тем, данные программные средства также обладают рядом недостатков. Остается необходимость установки и настройки данного ПО, как правило, на постоянно доступном сервере с внешним IP-адресом. Ряд средств не реализуют многопользовательский режим, то есть ориентированы только на персональное использование. Другие средства поддерживают работу с ГРВС нескольких пользователей, но только в рамках общего набора ресурсов, заданного администратором.

* Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 14-07-00309.

Платформа Everest [1-3] реализует новый подход к решению проблем публикации, выполнения и композиции вычислительных приложений в распределенной среде. В отличие от существующих решений, Everest является облачной платформой, реализующей модель облачных вычислений Platform as a Service (PaaS). Это означает, что вся функциональность платформы доступна через удаленные интерфейсы: пользовательский веб-интерфейс и программный интерфейс (REST API). Один экземпляр платформы может обслуживать много пользователей, позволяя им размещать свои приложения, запускать их и делиться ими с другими пользователями без необходимости установки дополнительного ПО на компьютеры пользователей. Размещенные в Everest приложения автоматически становятся доступными через веб-интерфейс и REST API. Последний позволяет автоматизировать работу с приложениями, осуществлять их композицию в рамках расчетных схем (workflow) и работать с приложениями из внешних систем.

Отличительной чертой Everest является поддержка запуска приложений на произвольных комбинациях внешних вычислительных ресурсов. Пользователи платформы могут подключать к ней имеющиеся в их распоряжении вычислительные ресурсы и создавать персональные ГРВС. В отличие от существующих средств организации вычислений в ГРВС, все действия по конфигурации ресурсов и запуску вычислительных заданий осуществляются через веб-браузер. При этом платформа поддерживает одновременное функционирование множества ГРВС, созданных пользователями. В работе описывается интеграция платформы Everest с одиночными вычислительными ресурсами и грид-инфраструктурой EGI, а также рассматриваются проблемы, связанные с комбинированным использованием данных ресурсов для выполнения размещенных в Everest приложений.

2. Архитектура платформы Everest

Рассмотрим кратко общую архитектуру платформы Everest (Рис. 1). Серверная часть платформы состоит из трех уровней: программный интерфейс (REST API), приложения и вычислительное ядро. Клиентская часть платформы включает пользовательский веб-интерфейс (Web UI) и клиентские библиотеки.

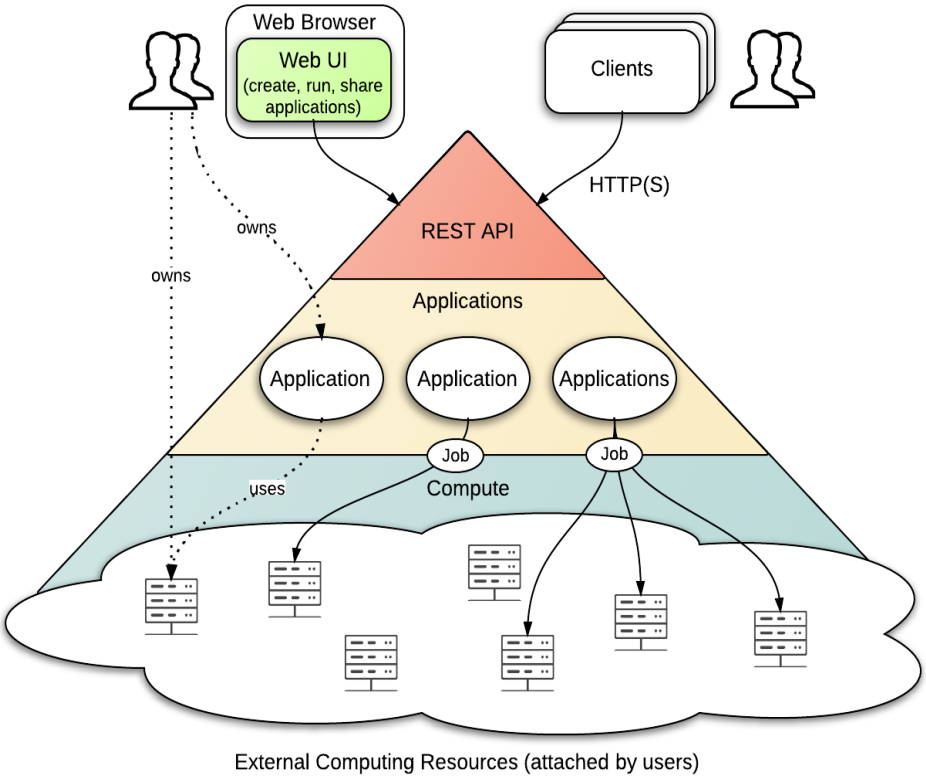

Рис. 1. Архитектура платформы Everest

REST API — программный интерфейс платформы, реализованный в виде веб-сервиса на основе стиля REST. Данный интерфейс включает операции для работы со всеми основными объектами платформы, такими как приложения, задания, ресурсы. Это единая точка входа для всех клиентов платформы, включая веб-интерфейс и клиентские библиотеки.

Уровень приложений соответствует среде размещения приложений, созданных пользователями платформы. Приложения является главными объектами платформы, представляющими собой вычислительные блоки с четко описанными интерфейсами. Каждое размещенное в Everest приложение автоматически публикуется как веб-сервис через REST API. Владелец приложения может настраивать список пользователей, которым доступен запуск приложения.

Платформа Everest не предоставляет собственных ресурсов для выполнения приложений и не связана жестко с какой-либо одной инфраструктурой. Вместо этого она позволяет пользователям подключать к себе внешние ресурсы и запускать приложения на любых сочетаниях данных ресурсов.

Вычислительное ядро управляет выполнением приложений на удаленных ресурсах. При вызове приложения через интерфейс платформы формируется задание (job), состоящее из одной или нескольких вычислительных задач (tasks). Ядро реализует все действия связанные с передачей данных, запуском и мониторингом задач на удаленных ресурсах. Также ядро осуществляет мониторинг состояния самих ресурсов и использует данную информацию при планировании задач.

Веб-интерфейс (Web UI) представляет собой графический интерфейс для работы пользователей с платформой. Данный интерфейс реализован в виде приложения на языке JavaScript, которое может выполняться в веб-браузере без необходимости установки дополнительного ПО. Веб-интерфейс взаимодействует с платформой через REST API, то есть использует тот же интерфейс, что и остальные клиенты платформы.

Клиентские библиотеки предназначены для упрощения программного доступа к платформе через REST API. Они позволяют пользователями легко писать программы, которые вызывают приложения и комбинируют их в произвольные сценарии. В настоящий момент реализована одна такая библиотека для языка Python.

3. Интеграция с одиночными вычислительными ресурсами

Традиционно для интеграции программных систем с удаленными вычислительными ресурсами применяются два подхода.

Первый основан на использовании общих протоколов удаленного доступа, поддерживаемых ресурсами, таких как SSH. Данный подход часто используется для интеграции с высокопроизводительными ресурсами. Основным его преимуществом является легкость подключения ресурса, поскольку для этого требуется только передать системе реквизиты доступа к ресурсу (например, SSH-ключ). Однако, в случае использования облачной платформы, не контролируемой непосредственно пользователем, передача реквизитов доступа системе может быть нежелательна из соображений безопасности. Более того, многие суперкомпьютерные центры запрещают их пользователям передачу своих реквизитов третьим лицам. Наконец, данный подход не применим в случае отсутствия как такового удаленного доступа к ресурсу, что имеет место для ресурсов, находящихся за межсетевым экраном, а также персональных компьютеров.

Второй подход основан на запуске на стороне ресурса специальной программы-агента, играющей роль посредника между ресурсом и системой. Данный подход распространен сред систем, ориентированных изначально на использование ресурсов персональных компьютеров, таких как Condor и BOINC. Преимуществами данного подхода являются поддержка интеграции с любыми ресурсами, вне зависимости от наличия доступа к ним извне, а также возможность реализации на уровне агента гибкого механизма безопасности. Данный подход также позволяет оптимизировать сетевой трафик и лучше масштабируется на большое число ресурсов. Основным недостатком второго подхода является необходимость установки и запуска специального ПО на стороне ресурса, что снижает удобство подключения ресурса.

На основе проведенного анализа в качестве основного метода интеграции платформы Everest с внешними вычислительными ресурсами был выбран второй подход, поскольку он об-

ладает большей универсальностью и позволяет обеспечить защиту ресурса. При этом были сформулированы следующие основные требования к агенту:

- простота установки и использования (минимальные системные требования, запуск без прав суперпользователя),
- поддержка доступа к ресурсу, находящемуся за межсетевым экраном, с минимальными требованиями к открытым «наружу» портам,
- защита ресурса от несанкционированного использования и запуска вредоносного ПО,
- открытый исходный код и открытый протокол взаимодействия с клиентами.

Исходя из данных требований, была выполнена разработка вычислительного агента для платформы Everest (Рис. 2). Агент реализован на языке Python, что позволило обеспечить переносимость агента на основные современные ОС. Кроме того, среда выполнения Python 2 присутствует сейчас практически на всех вычислительных ресурсах. Агент имеет минимум зависимостей и может быть быстро развернут на ресурсе пользователем без специальной подготовки и прав суперпользователя. С целью повышения уровня доверия пользователей, исходный код агента открыто доступен [4].

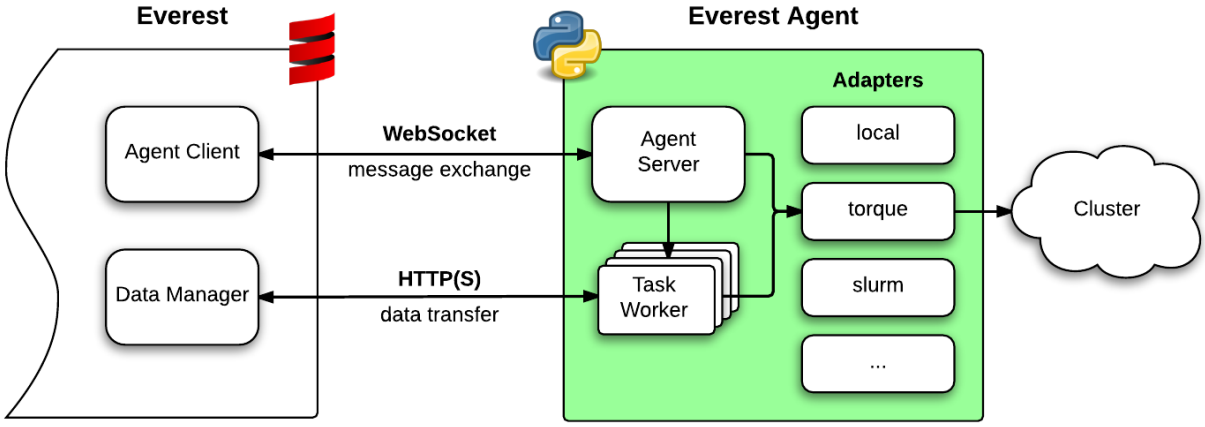

Рис. 2. Архитектура агента Everest

Взаимодействие между агентом и платформой реализовано на базе протоколов WebSocket и HTTP. Протокол WebSocket, позволяющий создать двусторонний канал связи между клиентом и веб-сервером, используется для передачи агенту команд (например, запуск нового задания) и приема от него различных уведомлений (например, изменение состояния задания). Протокол HTTP используется для передачи данных вычислительных задач между агентом и платформой. Данный подход обеспечивает связь агента с платформой с минимальными требованиями к открытым «наружу» портам и разрешенным протоколам на стороне ресурса. Подобные ограничения часто имеют место в суперкомпьютерных центрах и корпоративных сетях.

Для того чтобы подключить ресурс к платформе, пользователь должен зарегистрировать его через веб-интерфейс и получить секретный код ресурса. После этого необходимо развернуть агента на ресурсе, указав в конфигурации агента полученный секретный код. После успешного подключения агента к платформе он начинает передавать в Everest информацию о состоянии ресурса, которая отображается в веб-интерфейсе.

Поддержка взаимодействия агента с различными типами ресурсов реализована через механизм подключаемых адаптеров. Адаптер транслирует поступающие к нему со стороны агента вызовы (запрос состояния ресурса, запуск задачи, запрос состояния задачи, отмена задачи) в вызовы команд, специфических для данного типа ресурса. В настоящий момент реализованы и используются адаптеры для запуска задач на локальной машине в отдельных процессах или внутри одноразовых Linux-контейнеров, а также на вычислительных кластерах под управлением менеджеров ресурсов TORQUE, SLURM и Sun Grid Engine. В случае кластера агент выполняется на запускающей машине.

Для защиты ресурса от несанкционированного использования и атак реализованы следующие механизмы. При установлении соединения агент и платформа производят взаимную аутен-

тификацию. Пользователь может ограничить запускаемые агентом приложения, указав в конфигурации список разрешенных команд в виде регулярных выражений. Кроме того, реализация агента устойчива к атакам типа shell injection. Однако это не обеспечивает полной защиты в случае наличия уязвимых мест в самих приложениях или при запуске непроверенных программ. Одним из реализованных подходов к решению этой проблемы является запуск вычислительных задач в одноразовых Linux-контейнерах на основе технологии Docker.

В 2015 году функциональность агента была существенно развита. В частности, была реализована возможность взаимодействия с произвольными клиентами через открытый протокол (ранее агент мог подключаться и взаимодействовать только с Everest), а также реализовано управление агентом через графический интерфейс. Новая версия агента является самостоятельным программным решением, которое может использоваться для запуска вычислений на удаленных ресурсах в рамках произвольных систем.

4. Интеграция с грид-инфраструктурой EGI

European Grid Infrastructure (EGI) - крупнейшая грид-инфраструктура, объединяющая ресурсы сотен вычислительных центров по всему миру и ориентированная на поддержку исследований в различных областях науки. Внушительный объем агрегированных вычислительных ресурсов (около полумиллиона процессорных ядер в случае EGI) делает важным поддержку работы с подобными инфраструктурами в рамках платформы Everest.

Интеграция с EGI заметно отличается от интеграции с одиночным ресурсом. Во-первых, для запуска заданий в EGI необходимо иметь доступ к машине с пользовательским интерфейсом (т. н. User Interface). Однако у многих пользователей отсутствует доступ к такой машине. Таким образом, вариант с размещением агента Everest на запускающей машине грида, по аналогии с вычислительным кластером, отпадает. Во-вторых, для грида характерны высокие задержки при запуске заданий, часто имеющие непредсказуемый характер. В частности, некоторые задания могут «зависать» надолго в очередях отдельных ресурсов. Поэтому широкое распространение получила стратегия так называемых «пилотных заданий» (pilot jobs), когда в грид запускается универсальное задание-агент, связывающееся с центральным сервером и выполняющее назначаемые ему сервером задачи.

Исходя из описанных особенностей, для интеграции с EGI был реализован подход, заключающийся в запуске по требованию в грид разработанных ранее агентов (Рис. 3). Запуск агентов реализован через пользовательский интерфейс (EMI User Interface), развернутый на стороне платформы. Для всех пользователей используется один экземпляр UI, взаимодействие с которым осуществляется по протоколу SSH.

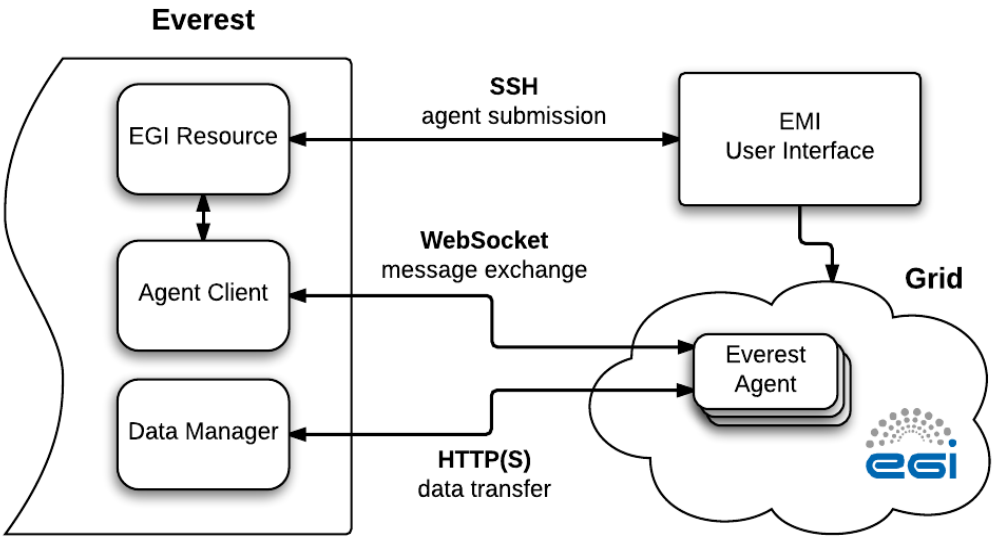

Рис. 3. Схема интеграции Everest с грид-инфраструктурой EGI

Everest контролирует число выполняющихся в грид агентов, запуская новых агентов по мере необходимости, но не больше установленного пользователем лимита. Будучи запущенным на ресурсе грида агент связывается с Everest и приступает к выполнению поступающих задач. Взаимодействие агента и платформы здесь ничем не отличается от случая одиночного ресурса. Однако агентов теперь запускает не пользователь, и с одним ресурсом в терминах Everest теперь может быть связано сразу несколько агентов. Кроме того, в случае отсутствия новых задач, агент автоматически завершает свое выполнение, чтобы освободить занимаемый ресурс.

Поскольку доступ к ресурсам в EGI регламентируется на основе участия пользователей в виртуальных организациях (ВО), то подключение ресурсов EGI к Everest также реализовано на уровне отдельных ВО. Пользователь Everest может подключить к платформе в качестве ресурса определенную ВО, передав платформе действующий прокси-сертификат. По истечении срока действия прокси-сертификата ресурс становится недоступным до тех пор, пока пользователь не обновит сертификат. В будущем планируется реализовать поддержку генерации и автоматического обновления прокси-сертификатов.

5. Заключение

В настоящее время платформа Everest позволяет относительно легко подключать к ней внешние вычислительные ресурсы различного типа, что открывает возможности для комбинированного использования нескольких ресурсов и создания ГРВС. Например, при проведении расчетов можно сочетать использование локальных вычислительных ресурсов организации пользователя с ресурсами грид-инфраструктуры.

Данная функциональность уже предусмотрена в платформе - при конфигурации или запуске приложения пользователь может указать сразу несколько ресурсов. Для простых приложений, состоящих из одной задачи, из нескольких альтернатив необходимо выбрать один ресурс. Для сложносоставных приложений, таких как многовариантный расчет, необходимо распределить задачи приложения между имеющимися в распоряжении пользователя ресурсами. В обоих случаях необходимо учитывать характеристики и текущую загрузку ресурсов.

Предварительные вычислительные эксперименты с использованием ГРВС из трех серверов и трех вычислительных кластеров, подключенных к Everest, показали, что платформа способна полностью загрузить используемые ресурсы. Тем не менее, эффективное решение данной задачи, оптимизирующее время выполнения приложений, требует доработки планировщика Everest. Также в будущем планируется провести аналогичные эксперименты с использованием ресурсов грид-инфраструктуры.

Другой проблемой, связанной с использованием нескольких ресурсов, является обеспечение переносимости приложений. Многие вычислительные приложения изначально создавались и использовались в рамках определенных ресурсов. При попытке запуска приложения на новом ресурсе могут возникать проблемы, связанные с отсутствием требуемых зависимостей и системных библиотек. Для решения данной проблемы предлагается использовать существующие подходы, основанные на виртуализации приложений.

Литература

1. Sukhoroslov O., Afanasiev A. Everest: A Cloud Platform for Computational Web Services. In Proceedings of the 4th International Conference on Cloud Computing and Services Science (CLOSER 2014). SCITEPRESS - Science and and Technology Publications, 2014, pp. 411-416.
2. О. В. Сухоросло в. Интеграция вычислительных приложений и распределенных ресурсов на базе облачной программной платформы // Программные системы: теория и приложения: электрон. научн. журн. 2014. Т. 5, No 4(22), с. 171–182.
3. Everest. http://everest.distcomp.org/
4. Everest Agent. https://gitlab.com/everest/agent/