ЗАЩИЩЕННАЯ ПЛАТФОРМА ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ ДЛЯ ЗАДАЧ КОМПЬЮТЕРНОГО ИНЖИНИРИНГА

В.С. Заборовский, А.А. Лукашин, А.С. Ильяшенко

1. Введение

Компьютерные технологии (КТ) становятся важнейшим инструментом развития науки, промышленности, транспорта и средств массовой информации. В современных условиях от надежности, производительности и масштабируемости компьютерных систем существенно зависит деятельность, как отдельных предприятий, так и эффективность функционирования экономики страны в целом. Создание вычислительных сервисов, обеспечивающих оперативное и качественное решение научных, производственных и проектно-технологических задач является приоритетным направлением развития КТ. При этом факторами, тормозящими создание таких сервисов и снижающих эффективность внедрения современных КТ в сферу наукоемкого промышленного производства, являются: отсутствие специалистов, имеющих знания и опыт в решении мультидисциплинарных инженерно-технических задач с использованием КТ, недостаточная техническая оснащенность предприятий средствами высокопроизводительных вычислений, высокая стоимость владения системным и прикладным программным обеспечением. Таким образом, комплексное решение задачи внедрения КТ в сферу промышленного производства требует: 1) модернизации существующей системы подготовки специалистов, обладающих знаниями и опытом применения современных компьютерных и программных технологий на всех стадиях планирования, проектирования, производства и испытаний новой техники; 2) разработки методов компьютерного инжиниринга для решения инженерно-технических задач в режиме удаленного защищенного доступа; 3) развития технологии виртуализации вычислительных ресурсов в рамках многоуровневой модели предоставления сервисов класса SaaS/IaaS/PaaS.

В докладе рассматривается подход к созданию вычислительной платформы, ориентированной на решение междисциплинарных задач компьютерного инжиниринга в среде облачных вычислений. Подход опирается на концепцию суперкомпьютерного центра «Политехнический», в рамках которого класс задач инженерного анализа декомпозируется в соответствии с особенностями организации вычислительных процессов на базе SIMD или MIMD архитектур, возможностями параллельной обработки, применения сопроцессоров и специальных вычислительных структур. Предложенные решения позволяют использовать преимущества микропроцессоров нового поколения, включая применение виртуальных многоядерных вычислительных узлов и реконфигурируемых решений на базе ПЛИС.

Проект создания платформы является частью плана развития научных исследований на кафедре «Телематика» (при ЦНИИ РТК), созданной на базе ФГБОУ ВПО «СПбГПУ», предусматривающего организацию обучения бакалавров и магистров по направлению 010200 «математика и компьютерные науки» подготовку аспирантов и проведение НИОКР, отвечающих требованиям международных стандартов качества инженерных разработок.

2. Программные технологии построения распределенных облачных сервисов

Одной из задач развития КТ в рамках концепции облачных сервисов является снижение сложности управления и повышение производительности решения прикладных задач за счет использования методов параллельных вычислений.

Хотя все современные императивные языки программирования поддерживают возможность организации параллельной обработки информации, однако разработка и отладка параллельных программ требует больших затрат как временных со стороны разработчика, так и денежных со стороны заказчика на оплату труда. Отладка многопоточных приложений крайне сложна и зачастую не может быть проведена в полном объеме для всех возможных наборов входных данных. Эта особенность обусловлена тем, что написание приложения, которое может выполняться на нескольких потоках, требует от программиста создание корректных механизмов синхронизации обращения к разделяемым данным (критическим секциям) в потоках выполнения. Синхронизация обращений к данным обусловлена особенностями императивного подхода, в котором функции-методы, вызываемые в процессе обработки данных, могут зависеть не только от их аргументов, но и от состояния системы в целом. Именно поэтому остается важным порядок вызова функций, так как они влияют на состояние системы и на процесс вычисления функций, вызываемых дальше в процессе работы системы. Эта особенность затрудняет параллельное выполнение. Таким образом, если вызываемая функция не зависит от текущего состояния, то ее легче отлаживать и тестировать разработчику, что ускоряет процесс разработки и исключает возможность появления трудновоспроизводимых ошибок, которые являются следствием допущенной программной ошибки в синхронизации доступа к данным из разных потоков.

Вышесказанное, является весомым аргументом в пользу выбора парадигмы функционального программирования, базовым принципом которой является вычисление значений функций от исходных данных и результатов вычислений других функций независимо от состояния системы, которое по умолчанию считается

неизменяемым (англ. immutable state), с использованием «чистых» функций (англ. pure functions). Также, вызываемые функции не могут быть зависимыми от внешних переменных, которые не участвуют в процессе вычислений в явном виде, а только влияют на поведение этого процесса. Императивное же программирование основано на понятии переменной, которое подразумевает возможность изменения значений в памяти в процессе функционирования системы, поэтому даже при подаче одинаковых входных данных функции нет гарантии, что на выходе всегда будет одинаковый результат.

Применение функционального подхода упрощает задачу распределения вычислительных процессов между узлами среды облачных вычислений, так как использование «чистых» функций позволяет добиться чрезвычайно высокого уровня параллелизма. При таком способе разработки в программе отсутствуют критические секции, и пропадает необходимость контроля порядка вызова функций из-за независимости результата от порядка их следования в программе.

В последнее время начинают набирать популярность мультипарадигменные языки, которые сочетают в себе возможность разработки кода с использованием как объектно-ориентированного, так и функционального подходов. Использование нескольких парадигм позволяет расширить операционные возможности языка и предоставить разработчикам возможность как писать в привычном стиле, так и использовать более широкий и гибкой набор операций .

Программной средой разрабатываемой платформы выбран язык Scala(SCAlable LAnguage), который позволяет разработчику применять функциональный подход, а также писать код на императивном языке, например на Java. Scala имеет реализации как для Java, так и для .NET, что обеспечивает возможности широкого применения большого количества разработанных библиотек, а также применять сам язык Scala для развития существующих приложений.

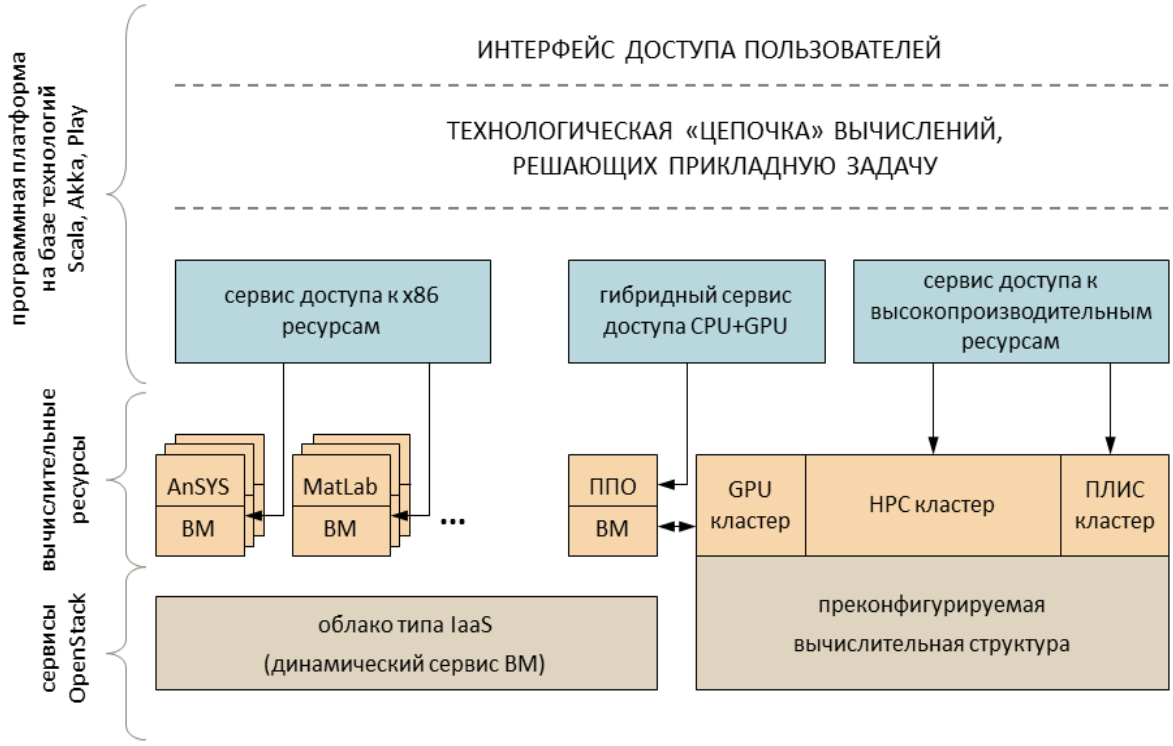

Рис. 1. Интеграция компонент платформы с помощью сервисов на базе технологий Scala

С точки зрения разработчика язык Scala имеет ряд преимуществ по сравнению с Java. Одним из них является перегрузка операторов, которая при грамотном задании имен этих операторов позволяет писать более лаконичный код, который может читаться практически как естественный текст или как математические формульные записи, что делает программный код простым для понимания. Также, язык Scala имеет богатый список стандартных неизменяемых (immutable) коллекций, которые отлично реализуют алгоритмы распараллеливания их обработки и извлечения данных. Реализованные стандартные алгоритмы обработки коллекций не заставляют разработчика «изобретать велосипед», а обращать внимание на более глобальные проблемы, которые связаны непосредственно с решаемой задачей.

Именно по этим причинам инструментарий языка Scala подходит для реализации распределенных приложений для платформы облачных вычислений и позволяет получить высокий уровень производительности при предоставлении сервисов пользователем облачной среды.

В рамках концепции инжинирингового центра было решено использовать программные технологии на базе Scala, в том числе сервис распределенных вычислений Akka и Web-фреймворк Play для интеграции

вычислительных ресурсов в единую экосистему. Сервисная платформа будет взаимодействовать со специализированным программным обеспечением, облачными сервисами и суперкомпьютерными ресурсами, а также интегрировать гетерогенные программно-аппаратные среды в единое вычислительное пространство, которое реконфигурируется для решения задач, формирующих технологическую цепочку. Такой подход позволит реализовать согласованное взаимодействие вычислительных сервисов и прикладных специалистов, обеспечив решение мультидисциплинарной вычислительной задачи, поставленной заказчиком услуги, что позволяет перейти к предоставлению высокотехнологического наукоемкого SaaS сервиса. Схема интеграции вычислительных ресурсов в единый вычислительный сервис представлена на рис. 1.

Кроме задачи построения сервисной архитектуры, рассматриваемые технологии могут быть использованы для создания научных сервисов с высоким уровнем абстракции. С помощью мультипарадигменного подхода возможно создание языков класса DSL, которые позволяют ученым формировать требования для решения задачи на языке, близком к языку проблемной области.

3. Архитектура и компоненты облачной платформы

Разрабатываемая облачная платформа для задач компьютерного инжиниринга обладает принципиальными отличиями, как от частных облачных систем, так и от публичных провайдеров. Во-первых, спектр предоставляемых сервисов гораздо шире. Как правило, облачные провайдеры, например, Amazon, предоставляют вычислительный сервис уровня IaaS, PaaS или SaaS. В случае Amazon это в основном IaaS — виртуальные машины и сетевая инфраструктура. В разрабатываемой облачной платформе для задач инжиниринга нет четко заданного набора сервисов, но при этом вычислительная среда базируется на IaaS платформе, что позволяет организовать на ее базе как PaaS, так и SaaS сервисы. При этом для разработчиков масштабируемых вычислительных систем облачная платформа предоставляет IaaS сервис, для инженеров и специалистов в проблемной области (например, прочностные расчеты или газодинамика) предоставляется PaaS сервис, а для заказчика-потребителя вычислительной услуги сервис уровня SaaS. На рис.2 приведена функциональная модель, которая иллюстрирует описанную систему.

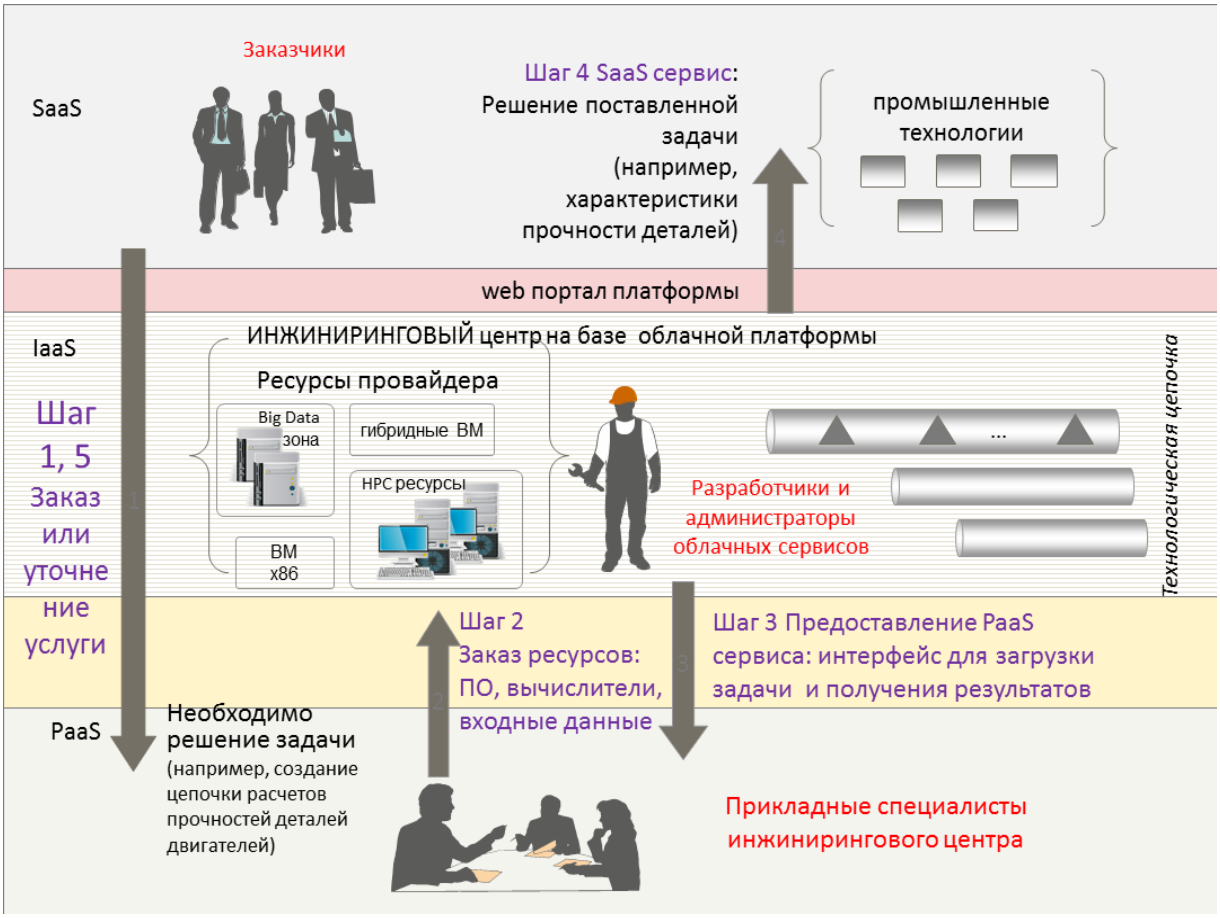

Рис. 2. Функциональная модель использования облачной платформы

К облачной системе рассматриваемого класса необходимо предъявить следующие требования:

1. Возможность масштабирования ресурсов, подключения аппаратных компонент без остановки функционирования облачной системы.
2. Наличие программных сервисов управления жизненным циклом вычислительных ресурсов.

3. Поддержка автоматической конфигурации программных и аппаратных компонент вычислительных ресурсов облака под решаемую задачу.
4. Наличие распределенного хранилища данных, доступного вычислительным ресурсам облака.
5. Обеспечение заданной политики безопасности в облаке. Разграничение прав между вычислительными ресурсами разных проектов, пользователей и групп.

В Государственном научном центре Российской Федерации – Федеральном государственном автономном научном учреждении "Центральный научно-исследовательский и опытно-конструкторский институт робототехники и технической кибернетики" совместно с кафедрой «Телематика» (при ЦНИИ РТК) ФГБОУ ВПО «СПбГПУ» осуществляется разработка защищенной среды облачных вычислений с перечисленными характеристиками. Кроме заявленных требований, разрабатываемая среда обладает поддержкой гетерогенных ускорителей на базе GPGPU, планируется поддержка реконфигурируемых ускорителей на базе ПЛИС [1]. Среда облачных вычислений разработана на базе сервисов OpenStack с открытым программным кодом, что обеспечивает возможность доработки платформы, отсюда возможна сертификация и проверка на отсутствие закладок в программном обеспечении. Пилотный облачный сегмент доступен в сети Интернет. Веб интерфейс платформы доступен по адресу http://cloudlet.stu.neva.ru.

4. Система контроля доступа к ресурсам платформы

Отдельное внимание необходимо уделить рассмотрению системы разграничения доступа в среде облачных вычислений. Особенностью данной системы является динамическая реконфигурация функционирующих в облаке вычислительных ресурсов: меняется состав виртуальных машин, программные компоненты, сессии пользователей, сетевые адреса. Поэтому, политика разграничения доступа должна быть сформулирована в виде инварианта, но при этом необходимо обеспечить реконфигурацию системы разграничения доступа в соответствии с текущим состоянием среды облачных вычислений. Подробно вопрос разграничения доступа в облачной среде рассмотрен в работе [2]. Авторами исследования предлагается рассматривать межсетевой экран как средство контроля доступа в облачной среде. Однако классические межсетевые экраны являются адресными устройствами, идентифицируемыми в сети. Чтобы обеспечить контроль доступа «каждый с каждым» при взаимодействии виртуальных машин был применен межсетевой экран, функционирующий в скрытном безадресном режиме [3], и осуществлена реконфигурация сетевой подсистемы среды облачных вычислений для перенаправления сетевого трафика на безадресный пакетный фильтр. Межсетевой экран в виде виртуальной машины был установлен в каждый сервер виртуализации облачной платформы и осуществляет контроль сетевого взаимодействия виртуальных машин. Межсетевой экран осуществляет контроль доступа в соответствии с правилами фильтрации, которые обычно задаются системным администратором. В случае фильтрации трафика в среде облачных вычислений необходимо регулярное переформирование правил фильтрации в соответствии с состоянием среды облачных вычислений. Рассмотрим эту задачу.

Для классификации сетевого трафика на соответствие политике доступа в среде облачных вычислений использована ролевая (RBAC) модель разграничения доступа, которая формально может быть представлена в виде совокупности следующих параметров:

$$M=<U,R,P,C>,$$

где U – множество идентификаторов пользователей среды облачных вычислений, которые осуществляют управление виртуальными машинами и информационными сервисами, а P – множество привилегий в форме описания разрешенных информационных сервисов, задаваемых в следующим образом:

- пользователь среды облачных вычислений u, предоставляющий информационный сервис;
- rul – множество правил, идентифицирующих информационный сервис. Правило r=<transport,port,protocol,ext>, где transport – транспортный протокол, port – порт по которому функционирует информационный сервис, protocol – прикладной протокол, ext – дополнительные признаки для полей заданного прикладного протокола.

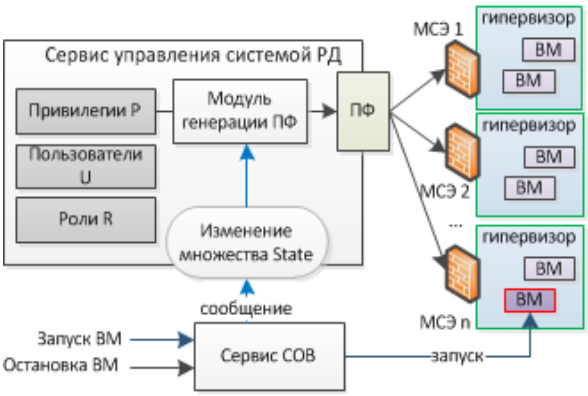

Рис. 3. Схема методики конфигурации правил фильтрации для межсетевых экранов

Например, привилегия может быть задана следующим образом: *{user:Ivan, [{transport:"TCP", port:"80", protocol:"HTTP", ext:[{method:"GET"}]}]}*. Данная привилегия задает правило доступа к веб серверу по протоколу http с использованием метода GET по 80 порту к виртуальной машине, принадлежащим пользователю Ivan.

Также в RBAC модели R - множество ролей, которые могут быть назначены пользователям. Роль задана в виде множества привилегий {p}. C – множество сеансов пользователей в среде облачных вычислений.

Состояние среды облачных вычислений зададим множеством IP адресов виртуальных машин с присвоенными метками пользователей $State=\{vm_i\}$. При изменении множества State необходимо осуществлять генерацию

правил фильтрации и конфигурацию межсетевых экранов. Для этого разработана методика динамической конфигурации правил.

Операция генерации правил фильтрации для заданной привилегии p заключается в подстановке IP адреса субъекта a_s (адреса виртуальной машины пользователя, обладающего привилегией) и объекта a_o (адреса виртуальной машины пользователя, предоставляющего информационный сервис) в каждое правило rul привилегии p. Схема методики конфигурации представлена на рис. 3.

С учетом изложенной методики, можно перейти к составу системы защиты информационного взаимодействия в среде облачных вычислений. Для осуществления функций фильтрации необходимо наличие виртуального межсетевого экрана в каждом сервере виртуализации среды облачных вычислений. Для осуществления функций генерации правил фильтрации для межсетевых экранов и хранения заданной политики доступа необходим отдельный программный сервис, размещенный во внутренней сети облака и соединенный с межсетевыми экранами сетью передачи данных для обмена управляющими сообщениями. В рассматриваемой облачной системе присутствует выделенная сеть управления компонентами облака, отделенная от виртуальных машин, поэтому является целесообразным использовать эту сеть для обмена информацией между компонентами системы разграничения доступа. Для информационного обмена и получения сообщений об изменении состояния облачной среды был использован протокол передачи сообщений AMQP, в частности его реализация в виде сервиса RabbitMQ. Использование общей шины для обмена сообщениями между компонентами облачной среды позволило реализовать оперативное получение информации подсистемой защиты и снизить затраты вычислительных ресурсов.

На рис. 4 приведена архитектура облачной среды, в которую интегрирована система разграничения доступа. Использованы следующие сокращения и обозначения: гипервизор – сервер виртуализации, СОВ – среда облачных вычислений, МСЭ – межсетевой экран, ВМ – виртуальная машина, eth – сетевой интерфейс, ПФ – правила фильтрации, РД – разграничение доступа.

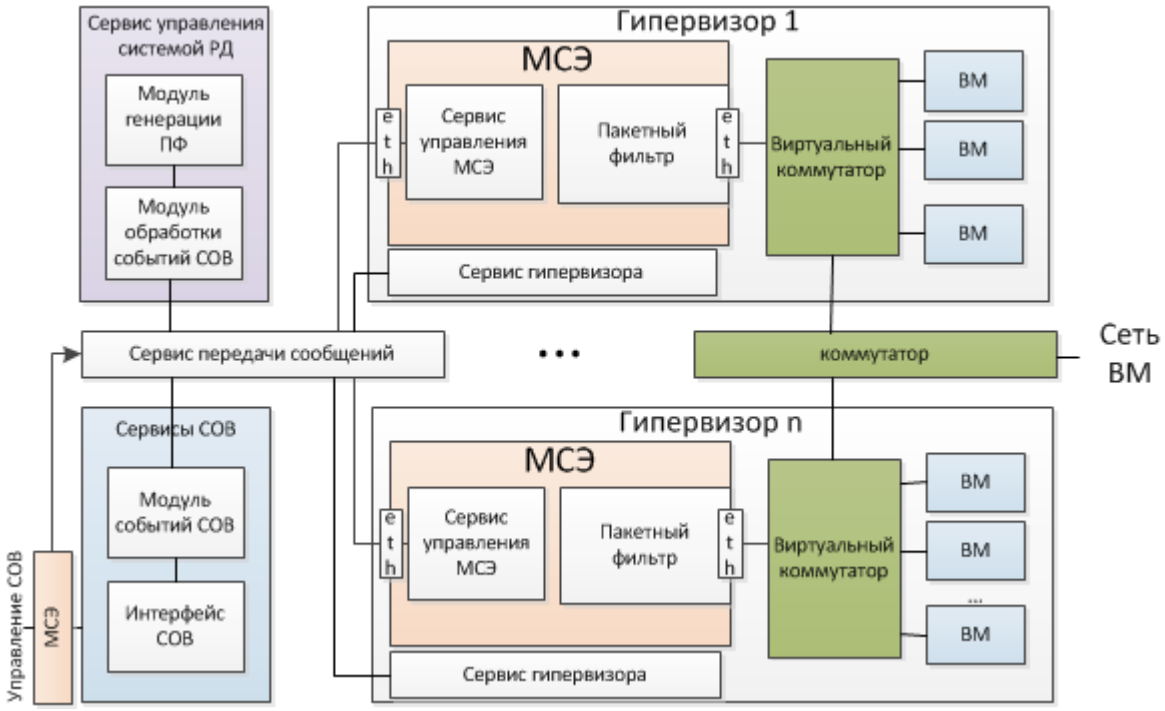

Рис. 4. Архитектура облачной среды с интегрированной системой разграничения доступа

Необходимо отметить, что реализация функций разграничения доступа требует дополнительных вычислительных ресурсов. Фильтрация сетевого взаимодействия требует дополнительных вычислительных ресурсов гипервизора. При использовании типового сервера виртуализации (12 вычислительных ядер, 64 Гб ОЗУ) затраты на фильтрацию трафика составят около 5-10% от ресурсов сервера. Кроме того, необходимы дополнительные аппаратные ресурсы на функционирование сервиса управления системой разграничения доступа. Однако за счет возможности горизонтального масштабирования производительности среды облачных вычислений, возможно решение проблемы потери вычислительных ресурсов с помощью подключения дополнительных аппаратных средств в облачную среду.

5. Заключение

Совокупность технических и организационных решений в рамках концепции инжинирингового центра позволяет создать высокопроизводительную-вычислительную систему, для управления которой используется команда высококвалифицированных специалистов из разных областей науки и техники, а технические решения интегрированы в единую сервисную среду облачных вычислений и предоставляются пользователям системы в

виде сервисов. Современные технологии программирования и построения распределенных вычислительных сервисов позволяют организовать управляемую, масштабируемую высокопроизводительную систему, способную решать широкий класс задач, которые могут быть интегрированы в единую технологическую цепочку, решающую комплексную задачу. Предлагаемый подход позволяет снизить расходы при решении сложных вычислительных задач за счет решения проблемы их длительного согласования и разработки интерфейсов взаимодействия между специалистами из разных проблемных областей. Единая масштабируемая облачная платформа позволяет реконфигурировать гетерогенные вычислительные ресурсы для решения задач разного класса, а представленная система разграничения доступа обеспечивает возможность совместного использования вычислительных ресурсов и при этом выполнение заданной политики безопасности.

Работа выполнена при поддержке Министерства образования и науки, государственный контракт № 14.A19.21.0593 от 20.08.2012.

ЛИТЕРАТУРА:

1. Лабошин Л.Ю., Лукашин А.А., Семеновский В.Б. Поддержка вычислений на графических ускорителях в виртуальной среде XEN для обработки потоков данных робототехнических систем [Текст] / Научно-технические ведомости СПбГПУ. Информатика, Телекоммуникации, Управление. №6 (162) 2012. – СПб.: Изд-во Политехн. Ун-та, С. 97-100.
2. Лукашин А.А., Заборовский В.С. Система защиты информационных сервисов в среде облачных вычислений [Текст] / Научно-методический журнал "Информатизация образования и науки". №2(18) 2013, С. 39-53.
3. В.А. Мулюха, А.Г. Новопашенный, Ю.Е. Подгурский, В.С. Заборовский Методы и средства защиты компьютерной информации. Межсетевое экранирование. Издательство СПб Государственный политехнический университет, СПб, 2010, 90 с.