ОСОБЕННОСТИ ПЛАНИРОВАНИЯ ПРОЦЕССА ВЫПОЛНЕНИЯ КОМПОЗИТНЫХ ПРИЛОЖЕНИЙ НА ВИРТУАЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ РЕСУРСАХ

А.А. Пащенко, С.В. Марьин, С.В. Ковальчук

СПбГУ ИТМО

E-mail: sergey.v.kovalchuk@gmail.com

На современном этапе развития вычислительной техники все большую популярность обретают распределенные вычислительные среды на базе концепции облачных вычислений [1]. Эта концепция ориентирована на организацию работы вычислительных систем на уровне абстрактных ресурсов (программных, аппаратных, информационных), организованных в рамках сервисной среды с унификацией интерфейсов доступа. При этом важным вопросом является организация оптимального планирования вычислительного процесса, усложняющаяся за счет необходимости конфигурации абстрактных вычислительных ресурсов с целью формирования оптимального плана выполнения. В работе предлагается подход к планированию вычислительного процесса, основанный на концепции iPSE [2], обеспечивающей высокоуровневую организацию вычислительного процесса с использованием формализованных экспертных знаний. В рамках данной концепции используется абстрактное описание процесса решения вычислительных задач в форме цепочки заданий (workflow – WF), требующей отображения на существующие вычислительные ресурсы.

Использование концепции облачных вычислений традиционно сопряжено с применением технологии виртуальных машин, обеспечивающей возможность легкой настройки вычислительной инфраструктуры в соответствии с задачами пользователя. В рамках предлагаемого решения используются виртуальные вычислительные ресурсы для более гибкой автоматической конфигурации инфраструктуры в соответствии с разработанным планом выполнения композитного приложения. Такой подход обеспечивает широкие возможности по оптимизации вычислительного процесса вплоть до настройки характеристик вычислительной техники. С другой стороны, использование такого подхода обеспечивает возможность быстрого формирования в автоматическом режиме сервисной среды из прикладных сервисов, состав и структура которых оптимизирована с учетом особенностей решаемой задачи.

Следует отметить, что реализация планирования вычислительного процесса традиционно относится к NP-сложным задачам, что приводит к необходимости использования эвристических и метаэвристических подходов для оптимизации ее решения [3]. В рамках данной работы используется сравнительный анализ известных эвристик планирования (MinMin, MaxMin, Sufferage и пр.), реализуемый на основе параметрических моделей производительности, описывающих отдельные элементы WF (вызовы прикладных сервисов). Это позволяет произвести оптимизацию параметров, как для вызовов отдельных вычислительных сервисов, так и для настройки вычислительных ресурсов.

Для сравнения на рис. 1 приведены результаты экспериментального исследования предложенного подхода с использованием средства имитационного моделирования процесса планирования для композитного приложения, включающего в себя параллельный запуск вычислительного пакета для решения задачи ансамблевого прогнози-

рования уровня Балтийского моря. В ходе выполнения композитного приложения планировщиком проводился априорный сравнительный анализ применяемых эвристик планирования с учетом стохастического характера изменчивости параметров вычислительной среды: в данном эксперименте оценка времени работы отдельного пакета считалась распределенной по нормальному закону с математическим ожиданием, равным полученной модельной оценки, и СКО равным 10% этой оценки.

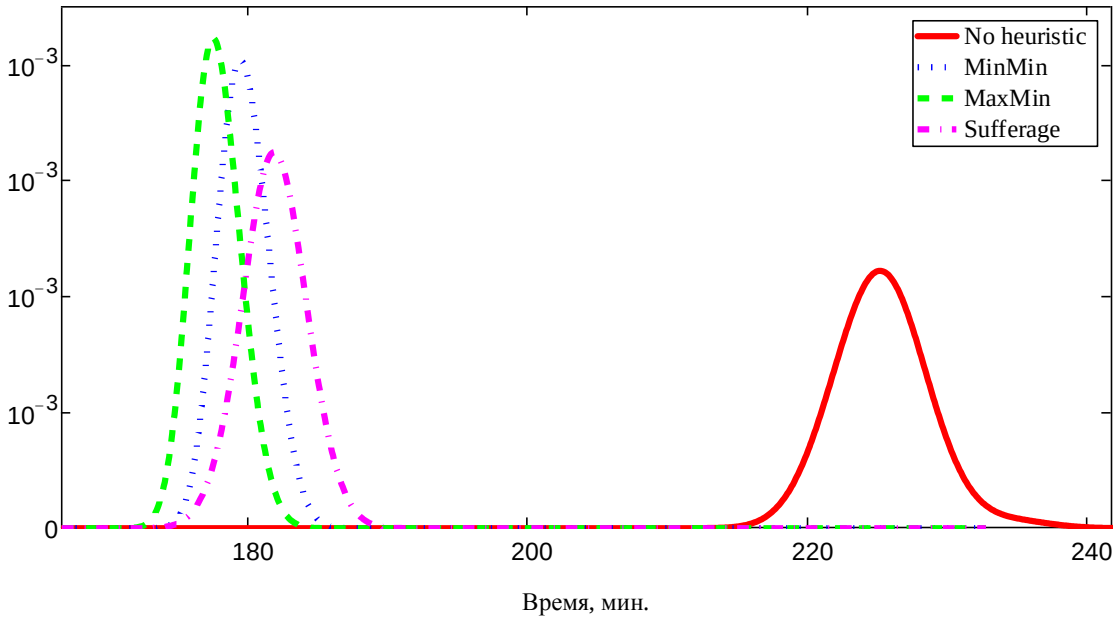

Рис. 1. Ядерные оценки плотности распределения времени работы композитного приложения при использовании различных эвристик планирования

Как можно заметить, применение эвристик планирования позволяет заметно уменьшить время работы композитного приложения. С другой стороны, при реализации композитного приложения с однородным набором блоков, готовых для исполнения, различие между эвристиками становятся незначительными. Однако целесообразность применения эвристик должна возрастать в случае применения описанного подхода в многопользовательской среде с одновременным планированием множества разнородных заданий, сформированных пользователями системы.

Любопытным моментом в контексте выполнения композитных приложений представляется возможность запуска задач WF на виртуальных вычислительных узлах. Это обусловлено ростом популярности технологий виртуализации в последние годы и постоянным расширением возможностей этих технологий вкупе с ростом производительности компьютеров в целом. Существует целый ряд различных современных средств системной виртуализации – распространяемых как свободно, так и за плату, предназначенных для решений разного масштаба, а также инструменты программирования и управления для этих систем. Все это позволяет решать на виртуальных системах самый широкий спектр задач с относительно небольшими накладными расходами на виртуализацию и значительно большей гибкостью администрирования, нежели при использовании реальных вычислительных узлов. В рамках работы подсистема работы с виртуальными машинами реализована в виде модуля системы управления выполнением композитных приложений в облачной среде, и может быть использована как напрямую в виде динамической библиотеки, так и через тривиальный интерфейс веб-сервиса. Эта подсистема устроена таким образом, чтобы обеспечить возможность унифицированной работы с виртуальными машинами, работающими под управлением различных систем виртуализации – для этого введен унифицированный интерфейс драйвера, на основе

которого можно реализовывать модули, инкапсулирующие особенности работы с конкретной системой, такой как VirtualBox, VMWare, или XenServer/XCP. Таким же образом можно реализовать и драйвер для работы с существующей конфигурацией такой системы управления виртуальной инфраструктурой, как, скажем, OpenStack - в первом случае можно будет работать с API сторонних систем с меньшими накладными расходами и меньшими усилиями на настройку конфигураций всех серверов, а плюсами второго случая являются дополнительный уровень абстракции от деталей работы систем виртуализации и б***о***льшая гибкость в построении различных конфигураций. Интерфейс драйвера весьма абстрактный и поддерживает следующие элементарные функции работы с виртуальными машинами:

- подключение к хосту, на котором запущены службы виртуализации;
- получение списка виртуальных машин хоста и их текущих состояний;
- базовые операции - запуск, перезапуск, останов, приостановка и возобновление работы виртуальных машин.

В настоящий момент реализован драйвер для работы с виртуальными машинами VirtualBox. Он представляет собой .NET-библиотеку, написанную на Microsoft C++/CLI, и фактически является интерфейсом к Java-библиотеке для работы с VirtualBox Web Service API, входящей в состав VirtualBox SDK, скрывая особенности работы с этим API и взаимодействия с Java-библиотекой из среды .NET за простым внешним интерфейсом, поддерживающим основные операции драйвера по работе с виртуальными машинами. Для работы этого драйвера необходимо, чтобы на целевом хосте, где размещаются собственно виртуальные машины, была запущена и доступна по сети (адрес, порт и т.п.) служба VirtualBox Web Service. Последняя, согласно документации к VirtualBox SDK, представляет собой SOAP-веб-сервис, позволяющий программно управлять виртуальными машинами на доступных на сетевом уровне хостах.

При оценке времени выполнения задач на виртуальных вычислительных узлах учитываются накладные расходы на запуск виртуальной машины, если она остановлена на момент запуска на ней задачи. В данный момент для учета этих расходов необходима предварительная оценка путем проведения нескольких экспериментов с конкретной виртуальной машиной на конкретной физической системе – величина этих расходов зависит от типа системы виртуализации, типа гостевой операционной системы, вычислительной мощности физической системы и других факторов. После получения этих оценок администратор добавляет в конфигурацию программного комплекса, относящуюся к имеющимся вычислительным ресурсам, элемент, соответствующий узлу, фактически являющемуся виртуальной машиной, а также прописывает ее параметры производительности и параметры соединения с хостом, где она размещена (в случае с VirtualBox, например, это IP-адрес, порт, имя пользователя и пароль). Эти данные имеют одинаковый формат и структуру у физических и виртуальных вычислительных узлов – кроме, понятно, данных о подключении к хосту, на котором размешена виртуальная машина: в случае физических узлов эта информация бессмысленна.

В результате при планировании выполнения задач планировщик рассматривает при подборе конфигурации запуска как физические, так и виртуальные вычислительные узлы, не делая на этом уровне никаких различий между ними. После завершения планирования незапущенные виртуальные машины, которые, согласно плану, предполагается использовать для выполнения запланированных задач, запускаются или возобновляют работу из приостановленного состояния. Концептуальная схема работы подсистемы виртуализации изображена на рис. 2.

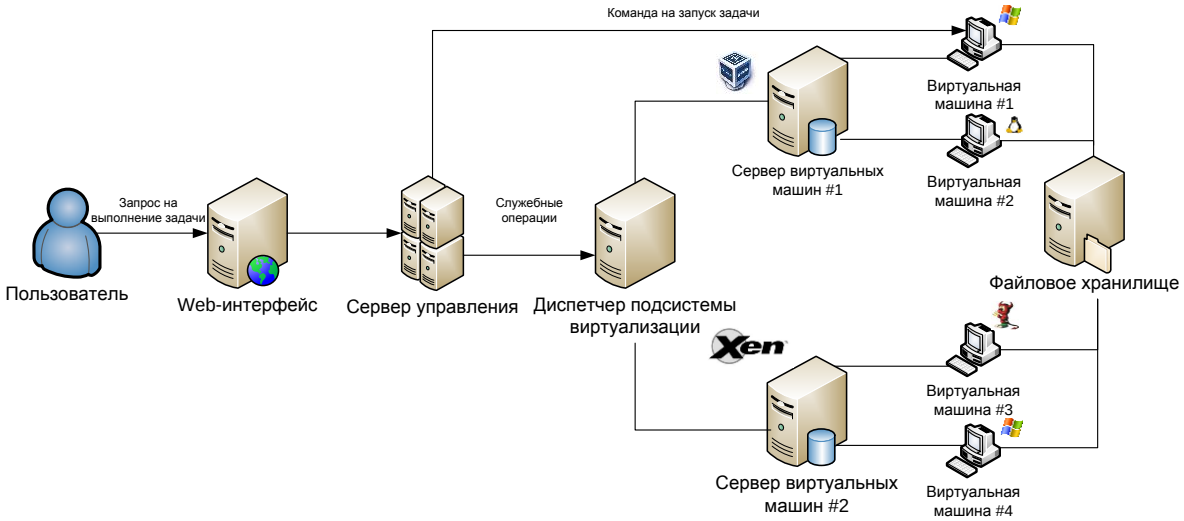

Рис. 2. Концептуальная схема работы подсистемы виртуализации

Процедура подготовки виртуальной машины для использования в качестве вычислительного узла внутри облака ресурсов не отличается от аналогичной процедуры для физических узлов: необходимо установить и настроить операционную систему и прикладные пакеты для расчетов, настроить должным образом сетевое соединение и обеспечить работу на машине службы удаленного запуска задач, которая также входит в состав предлагаемого решения, а также ее доступность по сети с узла, управляющего выполнением (обеспечить доступность необходимых портов и т.п.). В данный момент служба удаленного запуска проверена на функционирование в среде Microsoft Windows XP Professional SP2 x86 и Microsoft Windows 7 Professional SP1 x64: задачи запускаются, выполняются, а результаты поступают в центральное файловое хранилище системы управления вычислениями. Служба удаленного запуска также запускается в Linux-среде при условии наличия на машине установленной среды выполнения Mono Framework, являющейся свободно распространяемым аналогом среды Microsoft .NET (проверено на Open SuSE Linux 11.4 x64). Полноценный запуск задач на Linux-узлах пока, однако, невозможен - система запуска задач на целевой машине требует доработки с учетом особенностей UNIX-среды.

Литература

1. Foster I., Zhao Y., Raicu I., Lu S. Cloud Computing and Grid Computing 360-Degree Compared // eprint arXiv:0901.0131, 2008 (электронная версия: [http://arxiv.org/ftp/arxiv/papers/0901/0901.0131.pdf].
2. Интеллектуальные высокопроизводительные программные комплексы моделирования сложных систем: концепция, архитектура и примеры реализации / А.В. Бухановский, С.В. Ковальчук, С.В. Марьин // Известия высших учебных заведений. Приборостроение. 2009. №10. С. 5-24.
3. Workflow Scheduling Algorithms for Grid Computing / Yu [et al.] // Meta-heuristics for Grid Scheduling Problems. Springer, 2008. P. 173-214.