9. Vipul's Razor Homepage [Электронный ресурс]. – 2008. – Режим доступа: http://razor.sourceforge.net/ - Дата доступа: 17.09.2008.
10. Recurrent Pattern Detection Technology (RPD™) // CommTouch [Электронный ресурс]. – 2008. – Режим доступа: http://www.commtouch.com/site/Products/technology.asp - Дата доступа: 18.09.2008.
11. В.А. Головко. Нейронные сети: обучение, организация и применение. Кн. 10: Учеб. пособие для вузов / Общая ред. А. И. Галушкина. - М.: ИПРЖР, 2000. –С.114-129.
12. Е. Касперский. Компьютерное зловредство. – СПб.: Питер, 2007. - 208 с.
13. Kohonen T. Sell-organised formation of topologically correct feature maps// Biological Cybernetics. - 1982. - N43.-P.59-69.

Материал поступил в редакцию 23.09.08

GOLOVKO V.A., BEZOBRAZOV S.V., MELENCHUK V.V. Neural network approach for spam detection

The neural network approach for spam detection is described. The most used up-to-date methods for spam detection is examined. The weaknesses of described method for spam detection are showed. Spam detection system based on artificial neural networks applying is designed. Research results are submitted.

УДК 004.75:004.451

Отвагин А.В., Пынькин Д.А.

СРЕДСТВА СОЗДАНИЯ ВИРТУАЛЬНОГО ОПЕРАЦИОННОГО ОКРУЖЕНИЯ ДЛЯ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Введение. Современный этап развития вычислительной техники и скорость обновления вычислительных ресурсов приводит к тому, что многие организации, предоставляющие коллективный доступ к высокопроизводительным вычислениям, тратят значительные средства на поддержку и эксплуатацию вычислительных и коммуникационных систем. В этих условиях абсолютно необходимо внедрять современные технологии, обеспечивающие снижение расходов на поддержание компьютерной инфраструктуры. Вместе с тем вычислительные центры, организованные на базе различных аппаратных и программных платформ, могут существенно снизить производительность вычислений из-за отсутствия унифицированной вычислительной среды, что приводит к заметному снижению экономического эффекта использования таких центров. По этой причине многие вычислительные центры не стремятся своевременно модернизировать свое оборудование и вводить в эксплуатацию новые мощности.

Одной из перспективных технологий освоения и внедрения новых вычислительных ресурсов является виртуализация. Применение виртуализации позволяет добиться снижения стоимости вычислительных ресурсов, повышения производительности и адаптации имеющихся ресурсов к постоянно изменяющимся запросам пользователей. В настоящее время уже созданы базовые технологии виртуализации, и она становится все более эффективной, гибкой и надежной [1].

Технология виртуализации предназначена для запуска на одной аппаратно-операционной платформе нескольких операционных систем и/или приложений для них в независимых разделах; в этом случае физическая вычислительная система рассматривается как множество виртуальных. Пользователь при этом эксплуатирует свои приложения в собственном независимом разделе под управлением определенной версии операционной системы (ОС), а новые программные средства будут работать в другом разделе под управлением, возможно, другой версии или совершенно отличной по архитектуре ОС. Особенности эксплуатации новых приложений или системного программного обеспечения (ПО) не оказывают влияния на функциональность прежних версий, а необходимость поддержки устаревшего лицензионного ПО или оборудования не ограничивает возможности внедрения новых программных средств. Виртуализация позволяет инкрементальную модернизацию вычислительных средств, не ограничивая общую производительность системы.

Одним из преимуществ виртуализации является также возможность использования ее в качестве средства встраивания существующих вычислительных ресурсов в глобальные информационно-коммуникационные структуры, так называемые ГРИД-системы [2]. В этом случае виртуализация призвана сгладить отличия конкретной технологии организации высокопроизводительных ресурсов и гарантировать единое прозрачное представление ресурса для пользователя.

1. Выбор технологии виртуализации операционного окружения. Виртуализация является основным способом повысить производительность и эффективность использования вычислительных систем без кардинальной замены аппаратного обеспечения. Технология виртуализации основана на применении виртуальных машин (ВМ) – комплекса программных средств, обеспечивающих функционирование ПО, разработанного под конкретную программно-аппаратную платформу, в окружении, создаваемом с помощью ВМ [3, 4]. Основные цели виртуализации и преимущества, предоставляемые ее использованием, - это [5]:

1. Изоляция и абстрагирование аппаратного обеспечения. Предоставляя абстракцию аппаратуры, ВМ одновременно изолируют хостовую (базовую) ОС от реального аппаратного обеспечения, позволяя проводить обновление вычислительных мощностей без потери работоспособности ОС.
2. Использование ПО, ориентированного на уникальные архитектуры и ОС. В настоящее время вывод системного ПО из эксплуатации (например, из-за обновления аппаратной базы), приводит к немедленному прекращению использования программ, разработанных под данную платформу. Технология виртуализации позволяет эмулировать аппаратуру, для которой предназначено устаревшее системное ПО, тем самым обеспечивая его корректное функционирование на новых вычислительных ресурсах.
3. Тестирование нового системного ПО. Установка или адаптация нового системного ПО на вычислительные ресурсы, находящиеся в эксплуатации, прежде всего, требует периода тестирования, позволяющего судить о пригодности выбранных для обновления программных средств. Тестирование может выполняться на ВМ, специально выделенной для испытания нового ПО, тем самым, предохраняя общие вычислительные ресурсы от прекращения их корректного функционирования. Наличие нескольких тестовых ВМ позволяет разделить работу между тестировщиками и ускорить процессы миграции на новое ПО.
4. Экономия энергоресурсов. Вместо выделения отдельных разделов вычислительного центра, предназначенных для решения задач конкретной аппаратно-операционной платформы, виртуализация обеспечивает прозрачную эксплуатацию вычислитель-

***Отвагин А.В.**, старший научный сотрудник Объединенного института проблем информатики НАН Беларуси.*
Беларусь, ОИПИ НАН Беларуси, 220012, г. Минск, ул. Сурганова, 6.
***Пынькин Д.А.**, ассистент Белорусского государственного университета информатики и радиоэлектроники.*
Беларусь, БГУиР, 220013 г. Минск, ул. П.Бровки 6.

ных средств как единой системы, гарантирующей решение широкого спектра пользовательских задач без простоя ресурсов и ожидания освобождения специфических ресурсов.

Целью разработки технологии виртуализации, описанной далее, является практическая задача интеграции вычислительных ресурсов Республиканского суперкомпьютерного центра коллективного пользования (РСКЦ КП), расположенного в Объединенном институте проблем информатики Национальной академии наук Беларуси, в международные ГРИД-структуры, в частности в организацию BalticGrid [6]. Необходимым требованием для интеграции ресурсов является установка на узлах вычислительных средств специального варианта ОС, распространяемого среди членов BalticGrid. В настоящее время официальной версией ОС для консорциума является Linux, а именно дистрибутив Scientific Linux. В то же время, функционирование и управление вычислительными ресурсами РСКЦ КП осуществляется с использованием ОС Linux дистрибутива Fedora Core. Хотя отличия дистрибутивов не касаются основных принципов функционирования ОС, тем не менее они играют существенную роль при интеграции ресурсов в территориально-распределенные вычислительные структуры ГРИД.

Очевидным и наиболее простым решением в данном случае является замена действующего дистрибутива на требуемую версию. Однако такая замена неизбежно повлечет за собой остановку работы РСКЦ КП, которая может продлиться неопределенное время, поскольку нет гарантии, что аппаратная конфигурация вычислительных средств РСКЦ полностью совместима с предлагаемой консорциумом BalticGrid версией ОС. Кроме того, любая остановка означает существенные финансовые потери РСКЦ, поскольку на имеющихся вычислительных средствах решают свои вычислительные задачи не только внутренние, но и внешние пользователи. Смена дистрибутива также требует переподготовки персонала, работающего с вычислительными мощностями РСКЦ. Кроме того, уже имеющееся ПО, разработанное под уже используемый дистрибутив Fedora Core, потребует дополнительного тестирования и, возможно, адаптации в среде дистрибутива Scientific Linux.

Исходя из указанных причин, единственным приемлемым решением проблемы является виртуализация. Ее цель состоит в прозрачной подмене существующего операционного окружения на базе Fedora Core виртуальным окружением Scientific Linux для требуемых задач. В этом случае каждый узел будет представлять собой два виртуальных узла, разделяющие физические ресурсы и обеспечивающие видимость двух абсолютно изолированных версий ОС. Каждый пользователь может работать в своей версии ОС, не догадываясь о том, какая из них реально управляет аппаратурой вычислительного узла. Прозрачные абстракции системных средств, созданные для виртуального окружения, полностью поддерживают функционирование виртуальной ОС без каких-либо ограничений.

Для решения задачи прозрачного запуска приложений гостевой ОС в хостовой наиболее оптимальным является использование эмуляции прикладного интерфейса гостевой ОС, поскольку этот подход не требует дополнительных ресурсов. При этом достигается независимость от окружения хостовой ОС. Кроме того, несомненным плюсом такой техники является то, что все приложения, запущенные в выделенном окружении, будут использовать аппаратные ресурсы узла совместно со всеми другими программами и, соответственно, могут управляться диспетчером задач хостовой ОС.

Управление заданиями вычислительного кластера является одним из ключевых моментов обеспечения его производительности и эффективности. Внедрение средств виртуализации должно обеспечивать пользователям единые механизмы планирования заданий, чтобы предоставить им услуги вне зависимости от того, к какой категории они относятся и какой ОС пользуются. Удаленные пользователи ГРИД не должны ощущать никаких отличий по производительности по сравнению с локальными пользователями РСКЦ.

Одной из задач, которые должна решать предлагаемая технология, является актуализация ПО вычислительного узла. Актуализация сама по себе является ответственным моментом, поскольку ее качество существенно определяет качество предоставляемых пользователю услуг. При использовании виртуальных операционных окружений необходимо не только своевременно обновлять их ПО, но и гарантировать совместимость с хостовой ОС.

Таким образом, разработанная технология виртуализации решает следующие задачи:

1. Установка виртуального операционного окружения ОС Scientific Linux в ОС Fedora Core.
2. Настройка ПО вычислительного узла и среды выполнения.
3. Формирование и поддержка информационной среды приложения во время его работы.
4. Контроль выполнения, протоколирование процесса работы приложения.
5. Взаимодействие с локальными средствами планирования нагрузки вычислительного узла.
6. Контроль репозитория ОС Scientific Linux и ОС Fedora Core в РСКЦ.
7. Актуализация программного обеспечения вычислительных узлов в режиме автоматического обновления и по требованию системного администратора РСКЦ.

Для решения задачи создания виртуального операционного окружения была выбрана технология эмуляции прикладного интерфейса на базе средств chroot ОС Linux.

2. Реализация технологии виртуализации операционного окружения на базе chroot. Для запуска отдельно взятого приложения в виртуальном операционном окружении необходимо подготовить пакет программного обеспечения, предназначенный для установки в хостовую ОС, с необходимыми библиотеками и настройками, взятыми из гостевой ОС. Однако такой подход хорошо работает только для частных случаев и, как правило, применяется производителями ПО для обеспечения совместимости. При этом список необходимых библиотек заранее известен и жестко фиксирован, поэтому данный подход не является гибким и ограничивает возможности решения задачи для различных пользовательских приложений.

Для реализации технологии виртуализации наиболее оптимальным является создание выделенного окружения и стандартной для ОС Linux технологии «chroot», с помощью которой создается замкнутая среда, изолирующая запущенное приложение на уровне файловой системы.

Данная технология в основном используется для повышения безопасности приложений, предоставляющих внешний сервис в сетевой среде. Например, в дистрибутивах ALT Linux технология chroot используется многими сервисами, работающими с сетью, например, система управления базами данных Postgres, сервер имен bind, что обеспечивает безопасное функционирование хост системы даже в случае успешной атаки на аккаунт postmaster. Технология, частично похожая на chroot, используется сервисом vsftpd - основным ftp-сервером для Red Hat и Fedora Core. Процессы, запущенные в chroot-окружении, не изолированы от других процессов и ресурсов в системе. Процесс, запущенный в окружении chroot, может взаимодействовать с процессами, которые выполняются вне его chroot-среды. Это считается недостатком с точки зрения безопасности, однако очень подходит для обеспечения виртуализации ресурсов, где проблемы безопасности решаются на другом уровне и другими средствами.

Общая архитектура средств виртуализации операционного окружения представлена на рис. 1.

Основная операционная система полностью обслуживает аппаратное обеспечение вычислительного узла и используется для выполнения пользовательских расчетных задач. Разворачиваемое на ее основе виртуальное операционное окружение осуществляет эмуляцию прикладного интерфейса ОС и обеспечивает прозрачное взаимодействие гостевых ОС с аппаратурой. Виртуальное окружение также реализует прозрачный доступ пользователя к его файлам и каталогам согласно модели принадлежности ресурсов ОС Linux. Гостевая ОС формируется с помощью технологии создания пакетных окружений с использованием программного обеспечения SPT3 [7].

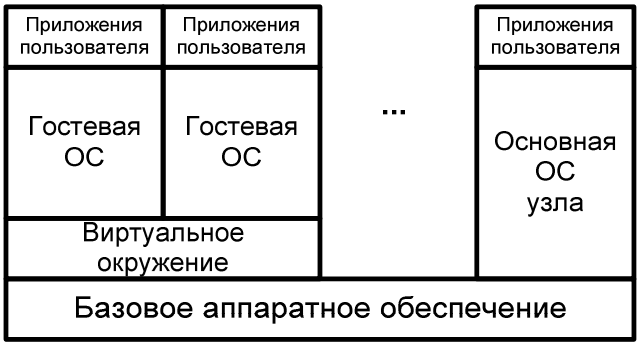

Рис. 1. *Архитектура средств виртуализации*

SPT3 представляет собой инструмент для создания решений на базе определенного репозитория и выполнен в виде раздельных скриптов, выполняющих строго определенные задачи, в соответствии с идеологией Unix и Linux. Механизм работы программного пакета SPT3 состоит в следующем. Вначале создается произвольный каталог, в котором будет выполняться процесс сборки окружения (рабочий каталог). Перед сборкой в нем должен быть помещен каталог profile с профилем, который либо создается с нуля, либо копируется из прилагаемых к пакету SPT3 примеров. Профиль может конфигурироваться для различных задач, т.е. profile содержит описания правил сборки виртуального окружения.

Профиль включает в себя файл recipe в каталоге профиля, который является инструкцией по сборке, выполнение которой приводит к созданию нужного окружения, а также все вспомогательные файлы, требующиеся для сборки. Инструкции – это перечисления некоего количество минимальных этапов сборки и указания их опций. Сборка выполняется запуском утилиты spt, которая выполняет по очереди все шаги, вызывая каждый из них с нужными опциями. Перед вызовом данной утилиты должна быть подготовлена среда с готовым репозиторием, определены и настроены опции для каждого скрипта в файле рецепта. Репозиторий должен содержать все необходимые пакеты заданной архитектуры, используемые как для работы самой системы SPT3, так и для размещения на конечном продукте.

Для развертывания созданных виртуальных операционных окружений создан программный комплекс из двух инсталляционных пакетов для ОС Fedora Core, устанавливаемых на вычислительный узел РСКЦ КП и формирующих виртуальное операционное окружение ОС Scientific Linux. Сформированное окружение используется для прозрачного запуска приложений. Кроме того, входящие в состав программного комплекса средства осуществляют автоматическое или принудительное обновление операционного окружения с использованием удаленного репозитория программных пакетов. Существует возможность создания инсталляционных пакетов на основе базового операционного окружения для запуска специализированных приложений, требующих использования дополнительных библиотек или определенных версий ПО (рис. 2).

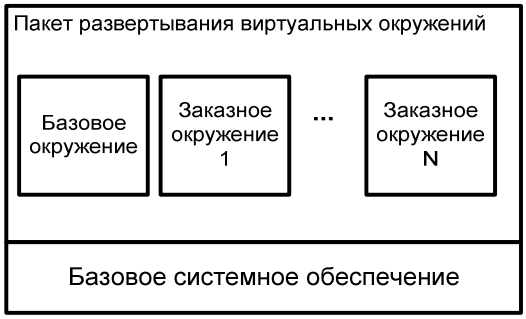

Рис. 2. *Состав ПО интеграции ресурсов*

Пакет развертывания виртуальных операционных окружений создает их с использованием дополнительных пакетов, представляющих собой архивы соответствующих окружений. Наряду с базовым виртуальным окружением могут быть созданы и инсталлированы заказные окружения под нужды конкретных приложений Грид.

Одна из основных проблем, которые необходимо решить при запуске задач в выделенном окружении — это синхронизация настроек между хостовой и гостевой ОС. Установленное виртуальное операционное окружение включает в себя часть пользовательских ресурсов, разделяемую между хостовой и гостевой системами. Эта часть в основном представляет статичные пользовательские документы, например, исполняемые файлы приложений. Для решения указанной проблемы создан инициализационный скрипт, в задачу которого входит экспорт директорий, необходимых для корректной работы гостевого окружения (например, /proc, /sys), либо директорий, использующихся для обеспечения взаимодействия между хостовой и гостевой ОС (например /home). Кроме того в момент инициализации синхронизируются некоторые файлы конфигурации, такие, как списки узлов кластера, методы доступа к спискам узлов кластера, списки локальных пользователей и т.д. Поскольку дистрибутивы Fedora Core и Scientific Linux относятся к семейству RedHat-подобных, то синхронизация между этими системами может происходить с помощью простого копирования файлов. В случае других дистрибутивов, особенно устаревших, может потребоваться преобразование некоторых файлов.

3. Интеграция виртуального операционного окружения со средствами управления нагрузкой. Средства управления нагрузкой являются основным способом обеспечения эффективности использования кластерных систем. Они входят в состав обязательного программного обеспечения всех современных вычислительных центров, поэтому при разработке технологии и средств виртуализации следует учитывать этот факт. Системы пакетного планирования, представляющие собой основное средство управления нагрузкой, реализуются в виде распределенной архитектуры клиент-сервер, обеспечивающей запуск приложений, сбор и анализ статистики выполнения, управление очередями задач, слежение за состоянием вычислительной системы.

Особенность применения систем управления заданиями в ГРИД-средах является необходимость прозрачного взаимодействия различных элементов системы, которые могут функционировать на совершенно отличающихся архитектурах. При этом следует обеспечить не только работу планировщика в среде ГРИД, но и сохранить его функциональность для пользователей вычислительного центра, не входящих в ГРИД. Система пакетного планирования должна различать категории пользователей, чтобы запускать их задания в соответствующем окружении, но при этом обеспечивать одинаковый уровень услуг и создавать иллюзию изолированной вычислительной системы, полностью ориентированной на нужды конкретного пользователя.

Общая схема системы пакетного планирования (на примере пакета Torque [8]) приведена на рис. 3.

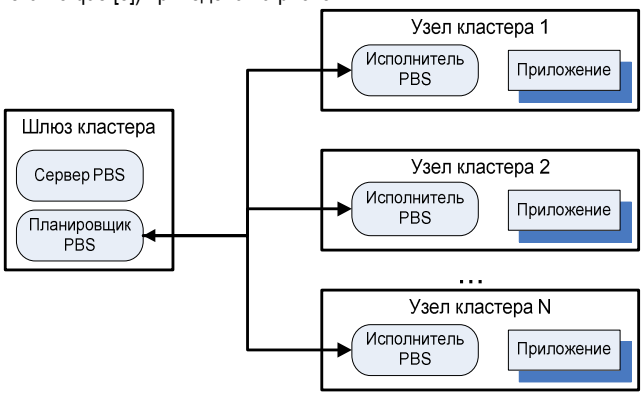

Рис. 3. *Архитектура системы пакетного планирования*

На каждом узле исполнитель системы пакетного планирования (процесс pbs_mom) осуществляет мониторинг узла и ожидает указаний от сервера по запуску приложений. Если сервер системы, рас-

положенный на узле-шлюзе кластера, получает от пользователя команду на выполнение задания, он помещает задание в определенную очередь. Планировщик (процесс pbs_scheduler) выбирает задания из очередей согласно дисциплине планирования и отправляет их на исполнение соответствующим узлам. Исполнитель системы пакетного планирования запускает процессы пользовательского приложения и управляет их исполнением, возвращая информацию серверу для учета ресурсов.

При использовании системы виртуализации возникает ряд трудностей с осуществлением управления по указанной схеме. Прежде всего, виртуальные окружения обычно вызываются только по требованию, т.е. пользователь одновременно может использовать узел лишь в одном окружении. При этом каждый узел вычислительного узла должен иметь возможность запустить виртуальное окружение, т.е. нельзя выделить в кластере какое-то фиксированное подмножество узлов для работы пользователей ГРИД. Анализ вычислительной нагрузки показывает, что доля задач пользователей ГРИД пока что ниже, чем доля задач локальных пользователей РСКЦ. Поэтому настройка задания для работы в виртуальном окружении, запуск задачи в нем и последующее свертывание должно производиться динамически.

Для динамической настройки виртуального окружения в состав пакета Torque, а именно в реализацию исполнителя системы пакетного планирования pbs_mom были внесены изменения. Архитектура системы приобрела следующий вид (рис. 4).

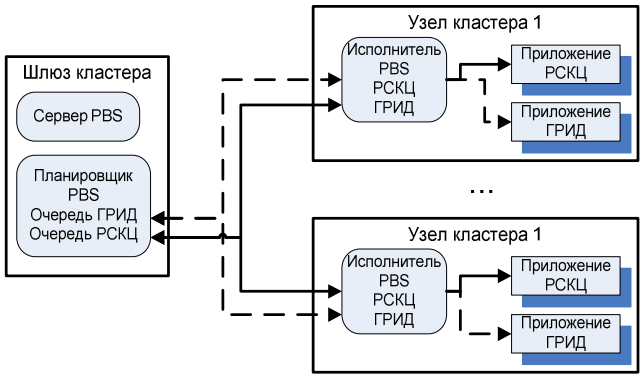

Рис. 4. *Модификация архитектуры системы пакетного планирования*

В системе планирования заданий была выделена специальная очередь для задач от пользователей ГРИД. Задания, поступающие в эту очередь, отсылаются исполнителям на узлах. Исполнитель определяет очередь, из которой поступило задание, и на основе своей конфигурации принимает решение о запуске нужного окружения. Если задание относится к локальным заданиям РСКЦ, то оно запускается в основной ОС по обычной процедуре. Если задание относится к заданиям ГРИД, то оно запускается через скрипт-обертку, который вначале устанавливает виртуальное окружение, а затем запускает задачу внутри него. После завершения задания виртуальное окружение сворачивается, возвращая результаты работы задания в пользовательское пространство файловой системы.

Преимущество такого построения системы состоит в том, что все части системы пакетного планирования работают под управлением основной ОС и имеют одну и ту же версию, что гарантирует их совместимость по протоколам. Адаптация существующей системы планирования заданий заключается лишь в замене исполняемого файла pbs_mom на новую версию.

Настройка системы производится с помощью файлов prologue, epilogue и конфигурационного файла, пример которого показан ниже.

Данный файл позволяет настроить запуск требуемого операционного окружения для определенного пользователя, группы пользователей или очереди задач. Гибкость конфигурации позволяет одновременно использовать любое множество операционных окружений для решения различных задач.

```
# Файл конфигурации для виртуальных окружений.

# Формат:
# CLASS            объект       путь
# где:
# CLASS принимает значения: QUEUE USER GROUP
# объект - имя пользователя, группы, очереди
# путь - путь к виртуальному окружению

# приоритет классификации задания по принадлежности:
# 1 - QUEUE
# 2 - USER
# 3 - GROUP

# примеры:
# USER skif   /var/skif/run/sl308-devel
# GROUP       skif   /var/skif/run/sl308
# QUEUE       def    /var/skif/run/sl308

USER          skif   /var/skif/run/sl308-
devel
GROUP         skif   /var/skif/run/sl308
QUEUE         def    /var/skif/run/sl308
```

Рис. 5. *Конфигурационный файл для запуска виртуальных окружений*

Еще одна задача, которую необходимо решить — единообразие доступа пользователя к операционному окружению. В общем случае пользователь может иметь доступ как к гостевому, так и к хостовому операционному окружению, поэтому необходим специальный скрипт, который позволяет зайти в гостевое операционное окружение и использовать ПО в этом окружении. Необходимость такого промежуточного скрипта вызвана тем, что нужно использовать утилиту «chroot», которая повышает привилегии пользователя для для изменения операционного окружения. Таким образом, промежуточный скрипт жестко задает параметры для вызова утилиты «chroot», а также понижает привилегии пользователя в гостевом операционном окружении.

В процессе тестирования решения выяснилось, что для некоторых задач, например с использованием технологии MPI, необходимо предоставить пользователю доступ на другие узлы кластера с использованием гостевого операционного окружения. На момент разработки такой удаленный доступ, с одновременной сменой операционного окружения, можно было обеспечить только с помощью модуля системы PAM (Pluggable Authentication Modules for Linux) pam_chroot и соответствующей его настройки, однако в последней версии сервера удаленного доступа «sshd» появилась возможность указать в настройках, какое операционное окружение необходимо использовать пользователю, либо группе пользователей. Несмотря на то, что использование системы PAM является более универсальным — в частности можно использовать смену операционного окружения при попытке доступа в систему с использованием любого метода доступа, для пользовательских задач, запускаемых в кластере, достаточно использовать смену операционного окружения пользователя только для сервера удаленного доступа.

Заключение. Разработанная технология виртуализации операционного окружения является очень простой и дешевой с точки зрения пользователя. Простота сборки заказных окружений позволяет создавать их для любого достаточно сложного приложения и практически для любой гостевой ОС. Созданные виртуальные окружения очень легко обновлять и поддерживать. При запуске задач через систему пакетного планирования задержка на организацию виртуального окружения незначительна и не отражается на эксплуатационных характеристиках и производительности вычислительной задачи.

В перспективе планируется дальнейшая доработка предложенной технологии виртуализации, в частности разработка базовых

виртуальных окружений для различных ОС Linux, а также автоматизированное создание заказных окружений для различных пользовательских приложений.

СПИСОК ЦИТИРОВАННЫХ ИСТОЧНИКОВ

1. А.Колесов. Наступает пора виртуализации / Колесов А. // Byte/Россия. - 2006. - № 5. - С. 17-19.
2. Foster I., Kesselman C. The Grid: Blueprint for a New Computing Infrastructure. – Morgan Kaufmann Publishers, 1998.
3. Гультяев А.. Виртуальные машины. Несколько компьютеров в одном. – Спб.: Питер, 2006. – 224 с.
4. Д. Циммер. Управление виртуальными машинами / Циммер Д. // LAN. - 2006. - Т. 12, № 5. - С. 56-59.
5. М.Михеев. VMware как архитектор виртуализации / Михеев М. // Byte/Россия. - 2006. - № 5. - С. 20-25.
6. BalticGrid-II Project Website [Электронный ресурс] / BalticGrid Consortium, 2008. – Режим доступа: http://www.balticgrid.org. – Дата доступа: 05.06.2008.
7. Якушин М. Описание работы с spt3. - [Электронный ресурс]. - Режим доступа: http://freesource.info/wiki/ALTLinux/Sisyphus/devel/spt3 – Дата доступа: 11.10.2007.
8. TORQUE Resource Manager. - [Электронный ресурс]. - Режим доступа: http://www.clusterresources.com/pages/products/torque-resource-manager.php – Дата доступа: 11.10.2007.

Материал поступил в редакцию 20.09.08

OTWAGIN A.V., PYNKIN D.A. The tools for virtual operational environment creation executed on a high-performance computing systems

The architecture and implementation of toolset for virtual operational environment creation is considered. The virtual environments are used for resource integration from high-performance computing systems to GRID architectures. A proposed toolset allows easy integration with standard batch planning system. The developed tools are included to system software of State Supercomputer Multi-access Center (SSMC) at United Institute of Informatics Problems (National Academy of Sciences of Belarus, Minsk).

УДК 004.8.03220

Кабыш А.С., Головко В.А.

КОЛЛЕКТИВНОЕ ПОВЕДЕНИЕ АГЕНТОВ НА ОСНОВЕ ПОДКРЕПЛЯЮЩЕГО ОБУЧЕНИЯ

Введение. Что такое агент. Термин «агент» используется множеством людей для обозначения множества различных понятий. Очень часто используются следующие варианты термина агент: Мобильный агент, Обучающийся Агент, Автономный Агент (т.е. робот), Планирующий агент, Модельный Агент, Распределенный агент, Программный Агент. Все перечисленные реализации агентов имеют различный состав, структуру и назначение, но в то же время являются агентами, т.е. объедены общими принципами создания и функционирования.

Согласно классическому определению, агент определяется как сущность, способная вести себя автономно, воспринимать среду, выполнять план действий и распознавать другие сущности в протоколах, в которых они участвуют. Среда и множество взаимодействующих агентов в ней образуют многоагентную систему (МАС). Несмотря на достаточно большое количество вопросов при проектировании МАС, для решения конкретных прикладных задач существуют общие подходы, рекомендации и стандарты для разработки. В первую очередь это стандарты на архитектуру многоагентной системы «Foundation of Intelligent Physical Agent» и «Open Agent Architecture», языки коммуникации агентов KQML и FIPA-ACL, языки описания онтологий и баз знаний.

Классическая схема коммуникации между агентами строится по принципу «запрос-ответ». Если агенту не хватает информации для принятия решения, он отправляет запрос «кто может мне помочь?» во внешнюю среду. Если ответов больше одного, чаще всего они фильтруются, либо выбирается ответ от наиболее авторитетного источника. Смысл коммуникации в том, что согласно принципам агентного моделирования, один агент никогда не должен обладать всей полнотой информации о системе. И именно за счет коммуникации осуществляется обмен требуемой информацией между агентами, а, следовательно, и достигается решение поставленной задачи.

Внутренняя онтология, или внутренняя база знаний агента представляет собой накопленные знания агентов. Способ принятия решений и механизм обучения тесно связаны и зависят от разработчика и решаемой задачи. В некоторых случаях предпочтительнее использовать нейронные сети и связанные с ними способы обучения. Очень часто многоагентные системы рассматривают как модель в рамках теории игр. В этом случае, в процесс функционирования МАС вплетены алгоритмы из теории игр – принятие решений, описание стратегий, формирование коалиций, планирование и др., а для изучения МАС используются методы данной теории.

При разработке многоагентной системы под конкретную задачу разработчику необходимо решить следующие вопросы:

1. Внутренний механизм принятия решений агента. Как агент воспринимает информацию о внешнем мире? Каким образом выбирает нужные действия? Как он обучается на своем, либо на чужом опыте?
2. Вопросы взаимодействия агентов между собой для решения своих локальных задач. Каким образом агенты передают знания друг другу? Каким образом они могут совместно решать задачи?
3. Внешняя среда должна представлять решаемую задачу. И главные вопросы здесь следующие: Каким образом описать задачу? Какие данные сообщаются агенту для решения задачи? Какой способ решения задачи в рамках многоагентного подхода?

1. Теория подкрепляющего обучения. Методология обучения с подкреплением [3] занимает промежуточное положение между обучением с учителем и без учителя. При использовании данной парадигмы в качестве обучающего сигнала выступает только “награда” и явно не указывается, какое выходное значение было правильным, а какое нет. Таким образом, в данном случае обучающаяся система получает намного меньше информации, чем при обучении с учителем и немного больше, чем при обучении без учителя. Обучение с подкреплением является сложным процессом, т.к. он основан на методе проб и ошибок, а сигнал награды обычно имеет невысокую информативность. Основной сферой применения данной парадигмы являются задачи, в которых необходимо найти оптимальную последовательность действий.

Рассматривается агент, взаимодействующий с внешней средой (рис. 1). Время предполагается дискретным: $t = 1,2,\ldots,\infty$ В текущей ситуации $s(t)$ агент выполняет действие $a(t)$. На следующем шаге $t = t+1$ он получит подкрепление $r(t) = r(s(t),a(t))$ за совершенное им на предыдущем шаге действие и перейдет в состо-

***Кабыш А.С.**, магистрант кафедры интеллектуальных информационных технологий Брестского государственного технического университета.*

Беларусь, БрГТУ, 224017, г. Брест, ул. Московская, 267.