УДК 004.75

А.А. САЛЬНИКОВ[*], В.В. ВИШНЕВСКИЙ[**], А.Ф. БОРЕЦКИЙ[*]

«ПЛАТФОРМА КАК СЕРВИС» В ГРИД ДЛЯ ИНТЕРАКТИВНОГО АНАЛИЗА МЕДИЦИНСКИХ ДАННЫХ

[*]Киевский национальный университет им. Тараса Шевченко, Киев, Украина
[**]Институт проблем математических машин и систем НАН Украины, Киев, Украина

Анотація. *Запропонована архітектура рішення, яка дозволяє використовувати грід-інфраструктуру для запуску інтерактивних застосувань з метою аналізу медичних діагностичних даних.*
Ключові слова: *грід, хмарні обчислення, телемедицина, електрокардіосигнал, інтерактивна обробка даних.*

Аннотация. *Предложена архитектура решения, позволяющего использовать грид-инфраструктуру для запуска интерактивных приложений с целью анализа медицинских диагностических данных.*
Ключевые слова: *грид, облачные вычисления, телемедицина, электрокардиосигнал, интерактивная обработка данных.*

Abstract. *Solution architecture that allows using grid-infrastructure for launching interactive applications in order to analyse medical diagnostic data has been proposed.*
Keywords: *grid, cloud computing, telemedicine, electrocardiosignal, interactive data processing.*

1. Введение

Как известно, грид-инфраструктура в Украине первоначально создавалась для решения задач физики высоких энергий при обработке данных экспериментов на большом адронном коллайдере. Однако ко времени старта национальной программы «Разработка и внедрение грид-технологий в Украине», которая проводилась в 2010–2013 годах, области использования грид-технологий и в Европейских странах, и в Украине были значительно расширены, в том числе и на задачи медицинской информатики [1, 2]. Поэтому ряд проектов грид-программы был посвящен прикладным медицинским проектам с использованием грид-технологий. В рамках этих проектов наш авторский коллектив инициировал и создал виртуальную организацию «Медгрид» (ВО «Медгрид»), а также непосредственно принимал участие в одноименном проекте «Медгрид», посвященном разработке действующих грид-приложений для массового накопления и обработки цифровых электрокардиограмм.

Таким образом, ВО «Медгрид» использует грид-инфраструктуру Украины как надежный и высокодоступный инструмент для хранения деперсонализированных медицинских диагностических данных, в частности, электрокардиограмм в цифровом формате SCP-ECG.

В ходе выполнения проекта «Медгрид» был создан ряд грид- и веб-сервисов для надежного хранения цифровых электрокардиограмм [3]. Были также разработаны алгоритмы и программы для автоматической обработки и миннесотского кодирования электрокардиограмм для пилотного популяционного обследования в Полтавском регионе [4]. При этом акцент в контактах с практическим здравоохранением делался на телемедицинские центры, которые развернуты во многих региональных кардиологических диспансерах и занимаются обработкой именно цифровых электрокардиограмм.

ISSN 1028-9763. Математичні машини і системи, 2015, № 1

Однако после многочисленных выступлений на конференциях по телемедицинской тематике и демонстрации медикам уже действующих информационных технологий проекта "Медгрид" стало понятно, что для нужд практического здравоохранения решения только задачи надежного хранения диагностических данных и организации к ним оперативного доступа в совокупности с автоматическими алгоритмами массовой обработки сигнала в эпидемиологических целях не достаточно. Логика реформ в организации медицинской помощи населению настоятельно требует организации телемедицинских сервисов для организации дистанционных консультаций врачами-кардиологами кабинетов функциональной диагностики. А для этого нужны технологии, которые предполагают интерактивную обработку одиночных сигналов.

Данная статья посвящена решению именно этой актуальной, на наш взгляд, задачи – использованию вычислительных мощностей грид-инфраструктуры для запуска интерактивных приложений в виде сервиса для пользователя, с одной стороны, и грид-задач для грид-инфраструктуры, с другой. При этом мы изначально учитывали то обстоятельство, что большинство промышленных интерактивных приложений для анализа медицинских данных разработано под ОС Windows.

2. Парадигма PaaS облачных вычислений

Модель облачных вычислений предусматривает предоставление сетевого доступа к ресурсам из доступного множества аппаратных ресурсов по требованию.

В соответствии со стандартом NIST (Национальный Институт стандартов и технологий США) парадигма платформы как услуги (Platform as a Service, PaaS) – это модель предоставления облачных вычислений, при которой потребитель получает доступ к использованию информационно-технологических платформ: операционных систем, систем управления базами данных, связующему программному обеспечению, средствам разработки и тестирования, размещённым у облачного провайдера. В этой модели вся информационно-технологическая инфраструктура, включая вычислительные сети, серверы, системы хранения, целиком управляется провайдером. Провайдером также определяется набор доступных для потребителей видов платформ и набор управляемых параметров платформ, а потребителю предоставляется возможность использовать платформы, создавать их виртуальные экземпляры, устанавливать, разрабатывать, тестировать, эксплуатировать на них прикладное программное обеспечение, при этом динамически изменяя количество потребляемых вычислительных ресурсов [5].

Практические реализации такой концепции в качестве фундамента для построения платформ используют инструментарий виртуальных машин (рис. 1).

Работу виртуальных машин обеспечивает инфраструктура виртуализации, которая включает в себя аппаратную платформу для работы виртуальных машин, гипервизор и средства централизованного управления как аппаратными компонентами, так и гипервизорами на узлах виртуализации.

Аппаратные ресурсы включают высокопродуктивные хранилища данных, узлы для запуска виртуальных машин, как правило, с аппаратной поддержкой виртуализации, сети передачи данных между хранилищами данных.

В качестве гипервизора на сегодняшний день используются различные реализации от различных разработчиков. Наиболее популярными являются решения: VMware, Citrix XEN, KVM, Microsoft Hyper-V.

При построении инфраструктуры виртуализации для PaaS могут использоваться готовые программные решения (например, OpenStack, OpenNebula) или облака (например, Amazon EC2, Microsoft Azure) [6]. Но все эти решения предусматривают либо капиталовложения в независимую инфраструктуру аппаратных ресурсов, либо аренду мощностей у облачного провайдера.

Рис. 1. Использование виртуальных машин для построения независимых платформ

Для реализации грид-программы в Украине существенное количество финансовых ресурсов было выделено на развитие аппаратных мощностей грид-кластеров. Для развития облачных вычислений создание аналогичного множества аппаратных ресурсов, при наличии огромных мощностей в грид-инфраструктуре, экономически не целесообразно.

Таким образом, возникла идея использования существующих аппаратных ресурсов грид-инфраструктуры для построения облачных сервисов без выведения их из эксплуатации грид-пользователями и службами.

3. Грид-инфраструктура как аппаратная платформа для PaaS

Современные, фактически уже классические программно-аппаратные средства управления облачными платформами (такие как OpenStack, OpenNebula и т.д.) предусматривают наличие полного доступа к централизованному множеству гомогенных серверных платформ, находящихся под управлением единого программного обеспечения под единым административным контролем [7]. Именно на этом множестве ресурсов по требованию запускаются и контролируются виртуальные машины с различными параметрами, доступ к которым получают конечные пользователи.

В грид-среде работа с аппаратными ресурсами выглядит принципиально иначе: ресурсы гетерогенные, географически распределенные, находятся под разным административным контролем под управлением разного программного обеспечения.

Тем не менее грид-инфраструктура Украины, не говоря уже о мировом масштабе грид-ресурсов, представляет собой огромные мощности, использование которых для запуска виртуальных машин в рамках облачной модели PaaS выглядит крайне привлекательно. Грид-инфраструктура уже существует, что минимизирует капиталовложения в аппаратную составляющую облака. Грид-службы замыкают на себя как механизмы централизованного доступа к множеству аппаратных ресурсов (в виде управления грид-задачами), так и механизмы аутентификации (Grid Security Infrastructure, GSI [8]) и учета использованных ресурсов (системы аккаунтинга SGAS, APEL [9, 10]).

Но в отличие от локального множества серверов под единым управлением доступ к аппаратным ресурсам через грид предусматривает прохождение многих программных уровней, начиная от самих грид-служб, заканчивая локальными механизмами контроля доступа операционной системы. Такой доступ более чем ограниченный, и возможность запуска виртуальной машины как грид-задачи требует решения ряда проблем на стороне грид-инфраструктуры, чтобы сделать такой подход возможным.

4. Вычислительный элемент грид-инфраструктруры как интерфейс доступа к аппаратным ресурсам

Точкой входа для доступа к ресурсам каждого из вычислительных кластеров грид-инфраструктуры служит вычислительный элемент (англ. Computing Element, CE). Рассмотрим типовую структуру вычислительного кластера типа Beowulf [11] как вычислительного грид-ресурса (рис. 2).

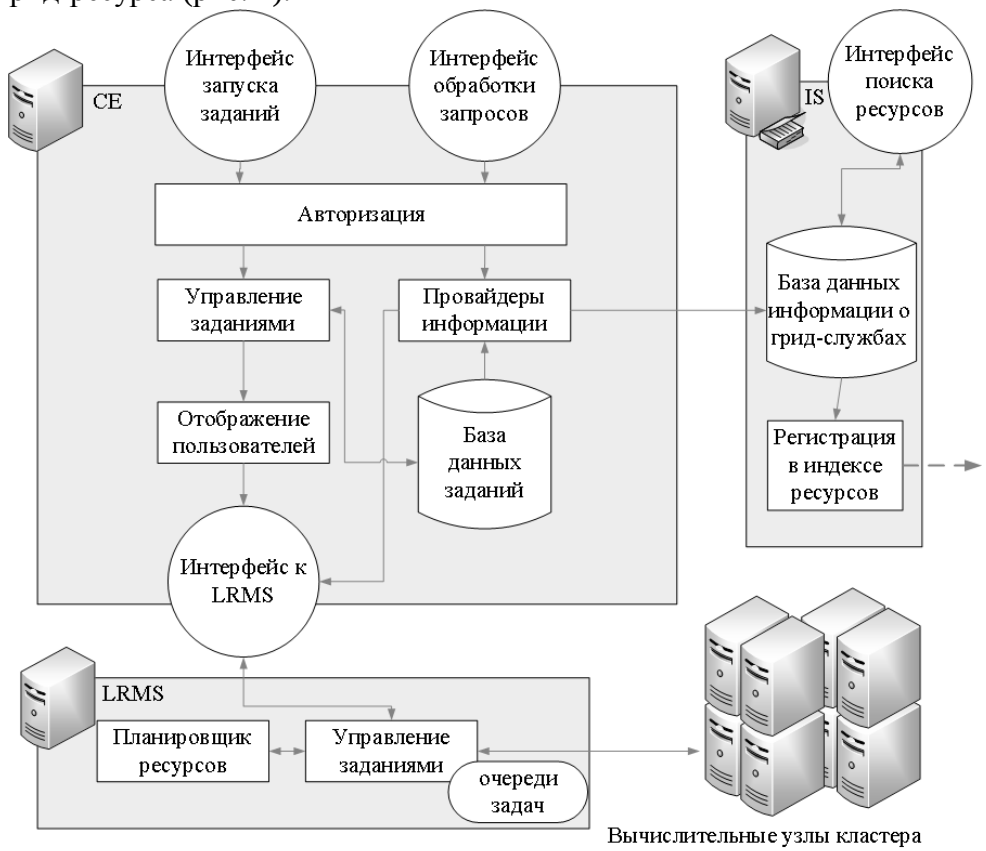

Рис. 2. Схема взаимодействия программных компонентов типового вычислительного элемента грид

Технологическая цепочка постановки грид-задачи следующая:

1. Грид-клиент получает информацию о доступных вычислительных элементах, используя индекс информационных систем.

2. Используя интерфейс поиска ресурсов информационной системы вычислительного элемента (Information System, IS), грид-клиент получает информацию о доступных аппаратных и программных ресурсах вычислительного элемента и сопоставляет их с критериями, необходимыми для запуска задачи.

3. Если вычислительный элемент удовлетворяет требованиям задачи, грид-клиент отправляет описание задачи, используя интерфейс запуска заданий вычислительного элемента.

4. После успешной авторизации задача отправляется в обработку. На этом этапе происходит подготовка к запуску задачи: создание необходимых директорий, скачивание исходных файлов и т.д.

5. Вычислительный элемент выполняет операцию отображения грид-пользователя на локальный регулярный аккаунт операционной системы и от имени локального аккаунта взаимодействует с локальной системой управления ресурсами кластера (Local Resource Management System, LRMS).

6. Задание, отправленное в LRMS, попадает в очередь и будет запущено на одном из вычислительных узлов кластера, который удовлетворяет требованиям задачи.

7. Состояние задания постоянно опрашивается и публикуется в информационной системе.

Как видно из проведённого анализа, цепочка программных средств от грид-задачи до выполнения на аппаратных ресурсах включает в себя фактически две службы, работающие в пакетном (batch) режиме: сам вычислительный элемент грид и локальную систему управления задачами.

На этапе выполнения на вычислительном узле задача работает с привилегиями локального аккаунта, на который был отображен грид-пользователь, что ограничивает возможности по работе с аппаратными ресурсами.

Таким образом, для запуска виртуальных машин на аппаратных ресурсах кластера через грид в общем случае необходимо передать информацию и исходные данные (в том числе образ виртуальной машины) через две системы пакетного режима и обеспечить выделение аппаратных ресурсов для запуска виртуальной машины, что потребует полномочий, выходящих за права доступа регулярного локального аккаунта.

5. Существующие решения, использующие грид как платформу для запуска виртуальных машин

Для решения задачи установки и настройки прикладного программного обеспечения на большом количестве гетерогенных ресурсов в швейцарской высшей федеральной технической школе Цюриха был разработан проект AppPot. В рамках этого проекта был реализован запуск расчётной задачи внутри виртуальной машины, которая, в свою очередь, запускается как грид-задача [12].

В качестве системы виртуализации используется пользовательский режим ядра Linux (User-Mode Linux, UML). UML – вариант ядра Linux, который позволяет запустить несколько виртуализованных Linux-систем в качестве обычных приложений в основной Linux-системе. Каждая гостевая система запущена как процесс в «пространстве исполнения пользователя», что позволяет запускать несколько виртуальных ОС без перенастройки основной Linux-системы [13]. Данная система виртуализации имеет небольшие накладные расходы, и расчётные процессы работают так же эффективно, как бы они работали непосредственно в ОС рабочего узла. Для управления жизненным циклом виртуальной машины используется shell-сценарий apppot-start.

Для конечных пользователей подготовлен базовый образ ОС, в который пользователи могут установить всё нужное прикладное программное обеспечение с нужными настройками. Такой подход позволяет не изменять ОС рабочих узлов и уменьшить затраты администраторов грид-узлов на установку, настройку и обновление разного прикладного программного обеспечения.

Несмотря на ряд весомых преимуществ, использование UML имеет и ряд ограничений:

- ограничение выделенной памяти для виртуальной машины;
- в качестве ОС виртуальной машины можно использовать только GNU/Linux;
- отсутствие интерактивного доступа к запущенной виртуальной машине.

6. Ресурсы для запуска виртуальной машины как платформы для интерактивного доступа

Рассмотрим множество программно-аппаратных ресурсов, которые необходимы для запуска виртуальной машины на конечном сервере.

Гипервизор. Управлением ресурсами виртуальных машин и трансляцией их запросов к реальной аппаратной составляющей сервера занимается гипервизор. Выбор гипервизора диктует как аппаратная (наличие поддержки функций гипервизора на аппаратном уровне в процессоре), так и программная (операционная система сервера, операционные системы виртуальных машин) составляющая.

В случае использования вычислительных узлов грид-кластера как платформы для запуска виртуальных машин мы имеем дело с операционными системами GNU/Linux, на которых в общем случае уже предустановлено программное обеспечение, используемое для работы с обычными задачами в виде процессов операционной системы. В европейской грид-инфраструктуре EGI официально поддерживаются дистрибутивы семейства RedHat Enterprise Linux 5 и 6.

Так как мы планируем предоставлять различные программные платформы как сервис, и учитывая тот факт, что большинство медицинского программного обеспечения для визуализации и анализа данных разрабатывается под ОС Windows, вариант использования контейнерной виртуализации с ядром Linux нам не подходит.

Для того чтобы оставить базовую инсталляцию рабочего узла максимально неизменной, целесообразно остановиться на гипервизоре второго рода (на основе базовой ОС). Для GNU/Linux таким гипервизором является KVM (Kernel-based Virtual Machine).

Сам по себе KVM не осуществляет эмуляции аппаратного обеспечения, а только обеспечивает выполнение кода виртуальной машины на реальных процессорах в изолированном адресном пространстве. Для эмуляции оборудования используется специальное приложение пользовательского режима, которое формирует начальное адресное пространство виртуальной машины и обслуживает ее ввод-вывод через специальный интерфейс /dev/kvm. В качестве эмулятора аппаратной части для гостевой ОС используется модифицированная версия программного эмулятора Qemu.

Средства для работы с Qemu в ОС GNU/Linux включают в себя как вызов процесса напрямую, так и через библиотеки и утилиты libvirt. Служба libvirtd может работать с привилегиями суперпользователя и таким образом автоматизировать работу с сетью, диском и пр. Существенным недостатком работы через libvirtd для концепции PaaS в грид является то, что виртуальная машина будет запущена вне системы управления ресурсами. А это приведет как к неправильному учету ресурсов самой системой управления, так и к отсутствию методов аккаунтинга потраченного процессорного времени конечным пользователем.

Поэтому в рамках концепции PaaS более приоритетным является вызов процесса Qemu из контекста грид-задачи. Но такой подход в свою очередь ограничивает нас в операциях, доступных непривилегированному пользователю.

Диск. Учитывая распределенный характер грид-среды, говорить о централизованном блочном хранилище, доступном для каждого грид-кластера, не приходится. Даже в условиях быстрых каналов связи (в Украинском Национальном Грид опорная сеть грид-инфраструктуры сегодня обеспечивает каналы 10Гбит/с [14]), вносимые задержки сети не позволят обеспечить должную работу централизованного SAN для образов виртуальных машин. Вместе с проблемами гетерогенности и учетом пакетного режима работы единственным вариантом остается использование файлов в качестве образа виртуальной машины.

Объем образа виртуальной машины в случае использования ОС Linux составит порядка 2–4 ГБ, для ОС Windows, в зависимости от версии, 4–10 ГБ. Даже при каналах в 1Гбит/с скорость первого скачивания образа будет исчисляться минутами, что приемлемо для времени ожидания запуска задачи в сравнении с существующими облачными платформами. В случае кеширования исходных файлов вычислительным элементом последующие запуски будут просто выполнять копирование по локальной сети.

Сеть. Сетевая инфраструктура каждого вычислительного кластера может существенно отличаться. Для взаимодействия узлов используется как разное оборудование, так и разная адресация.

Механизмы подключения виртуальной машины в сеть передачи данных кластера также могут отличаться. Среди возможных вариантов стоит исключить NAT на самом узле, так как это противоречит идее интерактивного доступа к разворачиваемым платформам. Среди методов подключения к физической сети стоит выделить два варианта: с использованием виртуального коммутатора (bridged networking) и при помощи технологии VEPA (Virtual Ethernet Port Aggregator) (рис. 3).

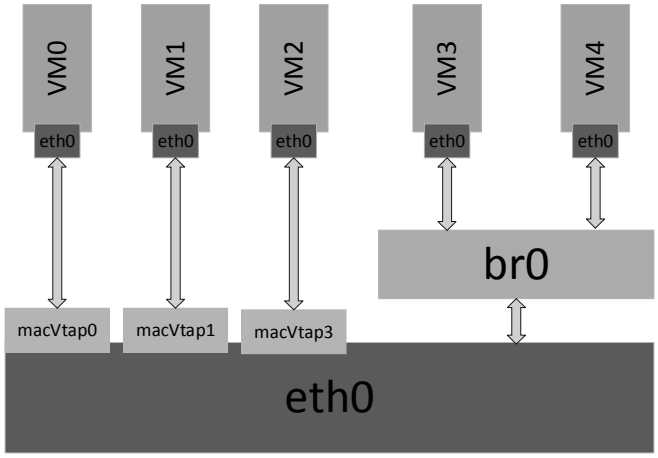

Рис. 3. Методы подключения виртуальной машины к сетевому сегменту

Использование виртуального коммутатора является классическим подходом для подключения виртуальных машин, но такой подход требует комплексной конфигурации рабочего узла специально под нужды виртуализации и как следствие является неоптимальным при соблюдении принципа минимального вмешательства в существующую систему кластера.

Альтернативой виртуальному коммутатору служит специальный драйвер – macvtap, который, по аналогии с macvlan, позволяет упростить схему коммутируемого подключения виртуальной машины. Используемая технология получила название VEPA: ethernet-кадры разных виртуальных машин могут отправляться на физический коммутатор с разными MAC-адресами источника, используя единый физический интерфейс сервера. Коммутация происходит на уровне физического коммутатора, убирая необходимость в виртуальном. Как следствие, обмен данными между физическим сервером и виртуальной машиной напрямую становится невозможным, но для разворачивания независимых платформ для PaaS в этом нет необходимости.

Таким образом, перед запуском виртуальной машины необходимо создать либо интерфейс виртуального коммутатора, либо виртуальный подынтерфейс физического адаптера в зависимости от метода организации доступа в сеть. Оба варианта требуют прав суперпользователя для таких манипуляций с сетевой конфигурацией сервера. В случае запуска грид-задачи полномочий регулярного аккаунта недостаточно и необходимо разработать дополнительные средства конфигурации сети, которые позволят создавать новые виртуальные интерфейсы таким пользователям.

Так как адресация сети для разных инсталляций разная, то универсальной схемы присвоения статических адресов виртуальным машинам не существует. При использовании грид-инфраструктуры как базы для PaaS целесообразно работать исключительно с динамически выделяемыми адресами, используя протокол динамической настройки узла (Dynamic Host Configuration Protocol – DHCP). Но в таком случае необходимы методы определения выданного в аренду виртуальной машине IP-адреса, что потребует взаимодействия с DHCP-сервером из контекста задачи.

Интерактивный доступ. В большинстве случаев сеть вычислительных узлов кластера – это приватный изолированный сегмент, доступ к которому извне отсутствует. Для обеспечения интерактивного доступа к платформе как сервису необходим посредник, который, с одной стороны, имеет доступ к сегменту сети вычислительных узлов, с другой стороны, доступен из сети Интернет.

Существуют несколько методов интерактивного доступа к платформе: доступ к физической консоли виртуальной машины и использование протоколов удаленного доступа непосредственно к гостевой ОС. В первом случае QEMU предоставляет возможность использования протоколов VNC и SPICE для доступа к физической консоли. VNC достаточно требователен к каналу передачи данных, особенно для интерактивной работы в графических приложениях без задержек и лишен возможности обмена файлами с платформой. SPICE – относительно новый протокол и требует установки специализированного программного обеспечения как на клиенте, так и на самой платформе, что накладывает ограничение на сам образ виртуальной машины.

Более гибким и прозрачным вариантом доступа является использование сервисов удаленного доступа самих гостевых ОС: для Windows протокол RDP, для Unix/Linux протокол SSH. Оба протокола предусматривают работу через достаточно медленные каналы связи и предоставляют механизмы обмена файлами с платформой. Также стоит отметить, что программы-клиенты для подключения по протоколам RDP и SSH уже установлены в большинстве ОС и доступны даже для мобильных платформ Android и iOS.

7. Архитектура и методы запуска виртуальных машин на грид-ресурсах

Принимая во внимание результаты проведенного анализа грид-технологий, парадигмы PaaS и существующих технологий создания и управления виртуальными машинами, была предложена следующая архитектура системы для работы с аппаратно-ускоренными виртуальными машинами как грид-заданиями (рис. 4).

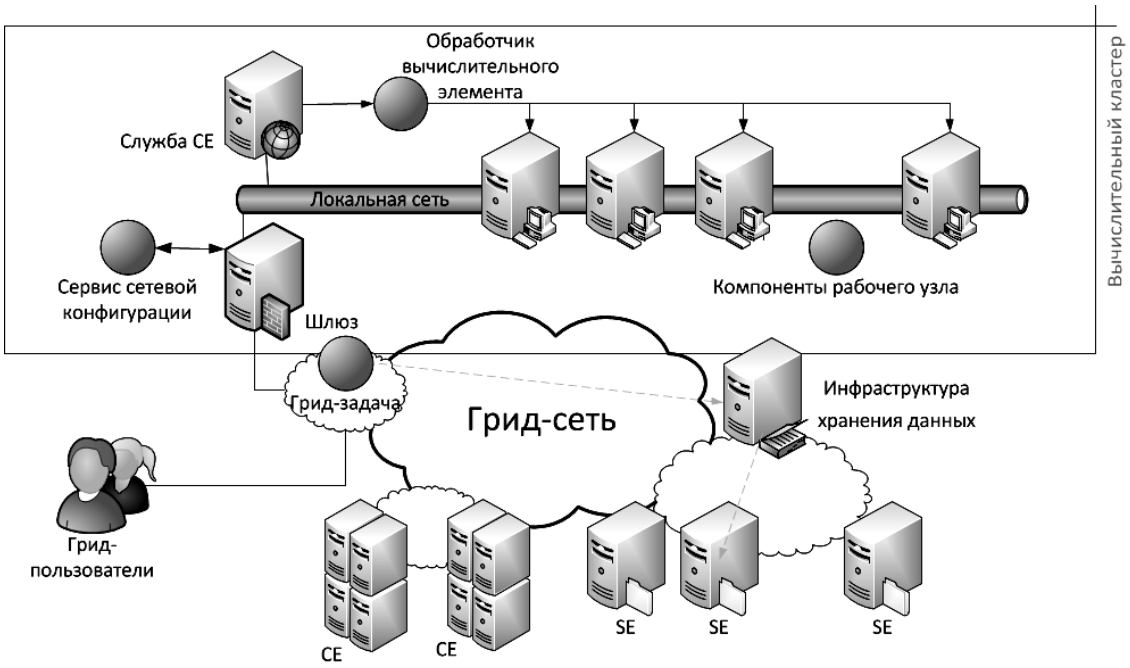

Рис. 4. Организационная структура компонентов для запуска виртуальной машины как грид-задачи

На рис. 4 обозначены программные компоненты, которые необходимо добавить в инфраструктуру вычислительного элемента, чтобы обеспечить рабочий цикл виртуальной машины, а именно:

• Сервис сетевой конфигурации (расположен на сетевом шлюзе), задачами которого является выделение (сдача в аренду) MAC-адресов сетевых адаптеров виртуальных машин, обеспечение привязки MAC-адресов к внутренним IP-адресам виртуальных машин,

выполнение функции перенаправления портов с внешнего IP-адреса на IP-адрес виртуальной машины для обеспечения интерактивного доступа к платформе.

• Обработчик вычислительного элемента (расположен на грид-шлюзе), задачами которого является подготовка информации для создания виртуальной машины, основываясь на данных грид-задачи. В рамках процесса передачи файлов грид-задачи скачивается образ виртуальной машины, который может быть расположен как в сети Интернет (используя общедоступные ресурсы), так и в распределенной высокодоступной инфраструктуре хранения данных грид.

• Компоненты рабочего узла (расположены на узлах вычислительного кластера), задачами которых является обеспечение жизненного цикла виртуальной машины, включая создание виртуальных аппаратных компонентов, обеспечение передачи данных виртуальной машине, запуск виртуальной машины с дальнейшим мониторингом состояния и завершением работы.

• Грид-задача (отправляется грид-пользователем на грид-шлюз), описание которой содержит необходимую информацию для управления платформой: файл образа виртуальной машины, данные, которые необходимо передать, параметры оборудования (процессор, память, диск), параметры интерактивного доступа и т.д.

В рамках представленной архитектуры предложен метод запуска виртуальной машины как грид-задачи. Согласно предложенному методу, технологическая цепочка функционирования системы следующая:

1. Грид-пользователь формирует файл описания задачи, в котором указывает параметры виртуальной машины, используя стандартный синтаксис семантического описания ресурсов и предварительно согласованные переменные для идентификации специфических параметров платформы (интерактивный доступ, описание виртуальных аппаратных компонентов и т.д.)

2. Грид-пользователь отправляет грид-задачу на выполнение в грид-инфраструктуру.

3. Брокер ресурсов грид определяет подходящий по требованиям к ресурсам вычислительный элемент и передает описание грид-задания грид-шлюзу (служба вычислительного элемента Computing Element – CE).

4. Служба CE выполняет передачу файлов задачи, включая файл образа виртуальной машины и данные для дальнейшей обработки, которые должны быть доступны внутри виртуальной платформы.

5. Служба CE, согласно описанию задачи, вызывает «обработчик вычислительного элемента», который, основываясь на информации из описания грид-задания, устанавливает ряд переменных, доступных из контекста задачи, на рабочих узлах кластера. Такие переменные определяют параметры жизненного цикла виртуальной машины.

6. Служба CE отправляет сформированное задание в LRMS.

7. LRMS выделяет вычислительный узел, удовлетворяющий требованиям задачи, и запускает на нем сценарий выполнения задачи на исполнение.

8. Сценарий выполнения задачи обеспечивает жизненный цикл работы виртуальной машины, используя «Компоненты рабочего узла», а именно:

• формирует виртуальный диск с данными, необходимыми внутри платформы;

• обращается к «Сервису сетевой конфигурации» и получает на время выполнения задачи MAC-адрес для виртуального сетевого адаптера;

• создает виртуальный сетевой адаптер, используя полученный MAC-адрес, обеспечивая подключение к локальной сети кластера;

• запускает виртуальную машину, используя параметры, полученные от «обработчика вычислительного элемента»;

• проверяет работоспособность и доступность виртуальной машины по локальной сети, используя «сервис сетевой конфигурации»;

• отправляет запрос на перенаправление порта «Сервису сетевой конфигурации»;

• сообщает грид-пользователю параметры интерактивного доступа к платформе;

• до окончания выделенного времени проверяет доступность виртуальной машины.

9. По окончании времени выполнения задачи происходит завершение работы виртуальной машины, удаление виртуальных сетевых интерфейсов и т.д.

8. Rainbow: реализация «платформы как сервис» в грид

Реализация компонентов инфраструктуры для создания «платформы как сервис» в грид для задач интерактивного анализа медицинских данных ВО «Медгрид», согласно предложенному методу, выполнена с учетом внедрения в Украинский национальный грид (УНГ), функционирующий преимущественно под управлением программного обеспечения Nordugrid ARC.

Комплекс программных средств получил название Rainbow («ARC in the Cloud»). В качестве обработчика вычислительного элемента, а также сценария управляющего циклом работы виртуальной машины, было принято решение использовать механизм RunTimeEnvironment (RTE) программного обеспечения Nordugrid ARC [15]. Сценарии RTE выполняются на стороне вычислительного элемента ARC (до отправки задачи в LRMS), что дает возможность реализовать все функции “Обработчика вычислительного элемента”.

В общем случае узлы вычислительного кластера гетерогенны и могут иметь различные методы подключения к сети. Поэтому в качестве “Компонентов рабочего узла” реализовано множество сценариев, работающих независимо. В зависимости от конфигурации узла сценарий RTE вызывает те или иные компоненты для обеспечения работы функций.

Реализация «Сервиса сетевой конфигурации» использует готовые элементы сетевой инфраструктуры ОС Linux: для работы с выдачей IP-адресов и перенаправлением портов служба взаимодействует со службами dhcpd и iptables. Служба запускается из-под xinetd, таким образом используя механизмы авторизации доступа упомянутой службы.

Интерактивный доступ к платформам на базе Windows осуществляется при помощи протокола удаленного подключения к рабочему столу RDP. Для обеспечения связи с виртуальной машиной, подключенной к внутреннему сегменту сети кластера, с сетью Интернет “Сервис сетевой конфигурации” динамически создает правила в iptables и сообщает TCP-порт для подключения пользователю.

Rainbow предоставляет конечному пользователю несколько сценариев обмена файлами с виртуальной машиной. Для передачи исходных файлов из грид-среды к виртуальной машине, помимо системного диска, подключается еще одно устройство. Если доступ к данным нужен только в режиме чтения, метод ISO позволяет подключить виртуальный CD-ROM с файлами. Для доступа в режиме записи метод DISK подключает виртуальный жесткий диск заданного размера и выбранной файловой системой (поддерживаются FAT32, ext3 и NTFS). Передача файлов, полученных в результате работы, назад в грид-среду производится аналогичным образом – файлы извлекаются из виртуального диска и передаются stage-out механизму программного обеспечения грид.

В описании грид-задачи, которое необходимо создать для использования Rainbow, вся дополнительная конфигурация (например, выбор метода передачи данных) вынесена в переменную среду, в то время как параметры виртуальной машины полностью приведены в соответствие с классическими требованиями к аппаратным ресурсам грид-задачи (рис. 5).

```
&
(jobName="Rainbow Cloud/VM Test")
(runTimeEnvironment="CLOUD/VM")
(executable="/bin/true")
(rsl_substitution=("CF_LOCATION" "lfc://lfc.grid.org.ua/grid/medgrid/.cloud")
    ("SYSTEM_IMAGE" "vm-medgrid.img")
)
(inputFiles=($(SYSTEM_IMAGE) $(CF_LOCATION)/$(SYSTEM_IMAGE) "cache=copy" )
    ("80_105_19900101_120000_2.scp". "lfc://lfc.grid.org.ua/grid/medgrid/kardio-
store/80/80_105_19900101_120000_2.scp")
       ("80_105_19970529_120000.scp" "lfc://lfc.grid.org.ua/grid/medgrid/kardio-
store/80/80_105_19970529_120000.scp")
)
(wallTime="720")
(memory="2048")
(count="2")
(disk="4096")
(cache="yes")
(environment=("CLOUD_VM_SYSIMAGE" $(SYSTEM_IMAGE))
    ("CLOUD_VM_DESCRIPTION" $(SYSTEM_IMAGE_DESCRIPTION))
    ("CLOUD_DATA_METHOD" "DISK")
    ("NOTIFY_EMAIL" "user@grid.org.ua")
)
```

Рис. 5. Пример описания грид-задачи для создания платформы средствами Rainbow

Работа всех компонентов архитектуры была протестирована для дистрибутивов операционных систем семейства Red Hat Enterprise Linux версий 5 и 6, которые официально поддерживаются в Европейской грид-инфраструктуре EGI. Для обеспечения простоты инсталляции разработанные программные модули были собраны в RPM-пакеты, которые помещены в отдельный репозиторий [16]. Вся конфигурация программных модулей вынесена в единый текстовый файл.

9. Выводы

В результате проведенного анализа парадигмы облачных вычислений «платформа как сервис (PaaS)» и архитектуры вычислительного грид-узла сформулированы ключевые требования к компонентам системы для разворачивания интерактивных платформ на аппаратных ресурсах грид-среды.

Предложена архитектура системы для запуска аппаратно-ускоренных виртуальных машин в грид, которая путем внедрения новых управляющих программных модулей – обработчика вычислительного элемента, сервиса сетевой конфигурации и компонентов рабочего узла, позволяет реализовать парадигму PaaS с использованием грид-ресурсов.

Предложен метод запуска виртуальной машины как грид-задачи, который путем координации работы распределенных программных компонентов архитектуры позволяет обеспечить жизненный цикл виртуальной машины, базируясь исключительно на требованиях описания грид-задачи.

Для реализации концепции создан комплекс программных средств Rainbow («ARC in the Cloud»), который включает в себя как необходимые программные компоненты, так и реализацию логики их взаимодействия в едином сценарии.

На базе предложенной архитектуры реализована виртуальная машина с предустановленным промышленным приложением по обработке электрокардиограмм «Cardiophon-

4», которая успешно прошла апробацию при проведении телемедицинских консультаций в ДП «Центр телемедицины МОЗ Украины» в 2014 году.[1]

Описание этого эксперимента выходит за рамки данной статьи и будет описано отдельно.

СПИСОК ЛИТЕРАТУРЫ

1. Авраменко В. Особливості застосування грід-технологій в медицині / В. Авраменко, А. Загородній, Є. Мартинов // Вісник НАН України. – 2008. – № 10. – С. 5 – 15.
2. Ходжибаев А.М. Новейшие информационные ГРИД-технологии в электронной медицине / А.М. Ходжибаев, Ф.Т. Адылова // Украинский журнал телемедицины и медицинской телематики. – 2005. – Т. 3, № 1. – С. 23 – 24.
3. Вишневский В.В. Грид-система для массового накопления и обработки цифровых электрокардиограмм / В.В. Вишневский // Український журнал телемедицини та медичної телематики. – 2013. – Т. 11, № 1. – С. 202 – 208.
4. Медицинская ГРИД-система на базе электрокардиограмм: новый инструмент для клинической кардиологии и популяционных исследований / В.В. Вишневский, И.А. Чайковский, Г.Д. Киржнер [и др.] // Кардиологи: от науки к практике. – 2012. – № 2. – С. 108 – 116.
5. Mell P. The NIST definition of cloud computing [Електронний ресурс] / P. Mell, T. Grance. – 2011. – Режим доступу: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.
6. Höfer C.N. Cloud computing services: taxonomy and comparison / C.N. Höfer, G. Karagiannis // Journal of Internet Services and Applications. – 2011. – Vol. 2, N 2. – P. 81 – 94.
7. Yadav S. Comparative Study on Open Source Software for Cloud Computing Platform: Eucalyptus, Openstack and Opennebula / S. Yadav // International Journal Of Engineering And Science. – 2013. – Vol. 3, N 10. – P. 51 – 54.
8. Welch V. Grid Security Infrastructure Message Specification [Електронний ресурс] / V. Welch // GFDI. 078. – 2006. – May 30. – Режим доступу: http://www.ogf.org/documents/GFD.78.pdf.
9. Scalable Grid-wide capacity allocation with the SweGrid Accounting System (SGAS) / P. Gardfjäll, E. Elmroth, L. Johnsson [et al.] // Concurrency and Computation: Practice and Experience. – 2008. – Vol. 20, N 18. – P. 2089 – 2122.
10. APEL: An implementation of Grid accounting using R-GMA [Електронний ресурс] / R. Byrom, R. Cordenonsi, L. Cornwall [et al.] // UK e-Science All Hands Conference. – 2005. – Режим доступу: http://vo.gridpp.ac.uk/abstracts/allhands2005/ahm05_rgma.pdf.
11. Beowulf cluster computing with Linux / T.L. Sterling ed. – MIT press, 2002. – P. 15 – 30, 61 – 95.
12. Murri R. AppPot: bridging the Grid and Cloud worlds [Електронний ресурс] / R. Murri, S. Maffioletti // EGI Community Forum, PoS (EGICF12-EMITC2). – 2012. – Vol. 4. – Режим доступу: http://pos.sissa.it/archive/conferences/162/004/EGICF12-EMITC2_004.pdf.
13. Murri R. Batch-oriented software appliances [Електронний ресурс] / R. Murri, S. Maffioletti // arXiv preprint arXiv:1203.1466. – 2012. – Режим доступу: http://arxiv.org/pdf/1203.1466.pdf.
14. Integration of the grid infrastructure of Ukraine in EGI / V.V. Pelykh, A.A. Salnikov, S.Ya. Svistunov [et al.] // The 5th International Conference “Distributed Computing and Grid-echnologies in Science and Education” (GRID ’12). – Dubna, Russia, 2012. – July 16 – 21. – 149 p.
15. Advanced Resource Connector middleware for lightweight computational Grids / M. Ellert [et al.] // Future Generation Computer Systems. – 2007. – Vol. 23. – P. 219 – 240.
16. Grid Blackjack repository [Електронний ресурс]. – Режим доступу: http://blackjack.grid.org.ua.

Стаття надійшла до редакції 01.12.2014

[1] Промышленный АРМ кардиолога «Cardiophon-4» использован нами при генерации виртуальной машины с любезного разрешения компании-разработчика «Ютас» (г. Киев).