Санкт-Петербургский государственный политехнический университет
Факультет технической кибернетики
Кафедра информационных и управляющих систем

Работа допущена к защите
Зав. кафедрой
_____ И.Г. Черноруцкий
"__"_____ 2012 г.

ВЫПУСКНАЯ РАБОТА БАКАЛАВРА

Тема: ***Автоматизация процесса решения ресурсоемких вычислительных задач с использованием облачных технологий и сервис-ориентированной архитектуры***

Направление: 230100.62 - Информатика и вычислительная техника

Выполнил студент гр. 4084/12 _____А.А. Лукашин

Руководитель, проф. _____В.П. Котляров

Санкт-Петербург
2012

Реферат

Abstract

Оглавление

Список иллюстраций

Список таблиц

Используемые определения и сокращения

SaaS	Программное обеспечение как сервис
	Software as a Service
IaaS	Инфраструктура как сервис
	Infrastructure as a Service
PaaS	Платформа как сервис
	Platform as a Service
CC	Контроллер облака
	Cloud Controller
SOAP	Протокол передачи объектов
	Simple Object Access Protocol
CAD	Computer Aided Design
CAE	Computer Aided Engineering
САПР	Система автоматического проектрирования
ПО	Программное обеспеспечение
ОС	Операционная система
PDA	Личный цифровой секретарь
	Personal Digital Assistant
SDK	Набор инструментариев для разработки
	Software development kit
UML	Унифицированный язык моделирования
	Unified Modeling Language

Введение

Актуальность темы

Информационно-телекоммуникационные технологии становятся важной составляющей инфраструктуры, используемой для инновационного развития научно-технической и социально-образовательной деятельности. Ключевой составляющей этой инфраструктуры являются информационно-вычислительные системы, развитие которых требует решения как фундаментальных, так и технологических проблем. Для их решения особое значение имеют исследования, связанные с созданием систем высокопроизводительных вычислений, ориентированных на решение широкого класса научно-технических задач, требующих реализации функций гетерогенности, масштабируемости и реконфигурируемости. При реализации существующих подходов к использованию информационно-телекоммуникационных технологий для создания систем распределенных вычислений пользователи сталкиваются с проблемой выбора методов конфигурации аппаратно-программных ресурсов для повышения эффективности решения выбранной прикладной задачи. Сложность создания вычислительных систем с требуемыми характеристиками операционной системы, аппаратных компонент и топологии сетевых каналов для организации локального и информационного обмена существенно зависит от особенностей решаемых прикладных задач, что делает актуальной разработку технологий автоматической реконфигурации используемых вычислительных ресурсов.

Предпосылки к созданию вычислительного сервиса

Политехнический университет, как научная мультитехническая организация, решает задачи из широкого круга проблемных областей, поэтому нуждается в обеспечении вычислительными ресурсами самых разных типов. Можно выделить различные требования к аппаратной конфигурации вычислительных ресурсов (количество вычислительных ядер, размер памяти и жесткого диска), к операционным системам (Windows, Linux, FreeBSD), к прикладному программному обеспечению (Ansys, Adams, ProEngineer, Matlab). При реализации существующих подходов к использованию информационно-телекоммуникационных технологий для создания систем распределенных вычислений, пользователи сталкиваются с проблемой выбора методов конфигурации аппаратно-программных ресурсов для повышения эффективности решения выбранной прикладной задачи. Сложность создания вычислительных систем с требуемыми характеристиками операционной системы, аппаратных компонент и топологии сетевых каналов для организации локального и информационного обмена существенно зависит от особенностей решаемых прикладных задач, что делает актуальных разработку технологий автоматической реконфигурации используемых вычислительных ресурсов. В этих условиях использование методов виртуализации и решений на основе технологии «облачных вычислений» (Cloud Computing) позволяет существенно расширить возможности управления ресурсами информационно-телекоммуникационной среды и гетерогенными компьютерными компонентами, что особенно важно при организации высокопроизводительных многопоточных вычислений, требующих дорогостоящего оборудования и сложного программного обеспечения, использование которых должно отвечать требования политикам

информационной безопасности.

Постановка цели и определение задач

Основной целью дипломного проекта является создание сервиса, предоставляющего возможность решения ресурсоемких вычислительных задач и допускающего предварительную конфигурацию вычислительной среды, необходимой для решения вычислительной задачи

В качестве задач, решаемых для достижения поставленной цели, были выделены:

- Анализ существующих методов выделения ресурсов и выполнения вычислительных задач
- Анализ возмжных архитектурных решений в данной области
- Разработка архитектуры, удовлетворяющей поставленным требованиям и здачам, на основе анализа предметной области
- Детализация отдельных модулей архитектуры
- Реализация сервиса-решателя(Solver)
- Реализация взаимодействия с облачной инфрастуктурой

Краткое содержание работы

Работа содержит 4 главы.

В главе 1 приведен анализ предметной области, описаны основные подходы к решению задачи организации описанной вычислительной среды.

В главе 2 описана основная концепция и архитектура системы.

В главе 3 приведено описание реализации системы согласно архитектуре, описанной в главе 2, дано полное описание программного

продукта.

В главе 4 производится анализ полученных результатов, представлены выводы о проделанной работе.

1 Анализ предметной области

Для решения поставленой задачи необходимо провести анализ существующих подходов и готовых решений. Для начала необходимо рассмотреть типологию задач, на которые ориентирован данный проект. Затем оценить существующие подходы к организации процесса их решения. Так как потребность в решении ресурсоемких прикладных задач достаточно высока, существует большое количество готовых программных продуктов, предоставляющих такую функциональность. Анализ предметной области нацелен на то, чтобы оценить преймущества и недостатки различных подходов, изучить существующие архитектурные решения, экономические факторы и выработать свою концепцию решения поставленной задачи.

1.1 Типы решаемых задач. CAD/CAE системы

1.1.1 CAD системы

CAD (*Computer Aided Design*) - система автоматизированного проектирования (**САПР**) — программный пакет, предназначенный для создания чертежей, конструкторской и/или технологической документации и/или 3D моделей. Современные системы автоматизированного проектирования обычно используются совместно с системами автоматизации инженерных расчётов и анализа **CAE** (*Computer-aided engineering*). Данные из CAD-систем передаются в **CAM** (*Computer-aided manufacturing*) — система автоматизированной разработки программ обработки деталей для станков с ЧПУ или ГАПС (Гибких автоматизированных производственных систем).

Обычно охватывает создание геометрических моделей изделия (твердотельных, трехмерных, составных), а также генерацию чертежей изделия и их сопровождение. Следует отметить, что русский термин «САПР» по отношению к промышленным системам имеет более широкое толкование, чем «CAD» — он включает в себя CAD, CAM и CAE.

1.1.2 CAE системы

CAE (*Computer-aided engineering*) — общее название для программ или программных пакетов, предназначенных для инженерных расчётов, анализа и симуляции физических процессов. Расчётная часть пакетов чаще всего основана на численных методах решения дифференциальных уравнений, таких как: метод конечных элементов, метод конечных объёмов, метод конечных разностей и др. Позволяют при помощи расчетных методов оценить, как поведет себя компьютерная модель изделия в реальных условиях эксплуатации. Помогают убедиться в работоспособности изделия, без привлечения больших затрат времени и средств.

Современные системы автоматизации инженерных расчётов (CAE) применяются совместно с CAD-системами (зачастую интегрируются в них, в этом случае получаются гибридные CAD/CAE-системы).

1.1.3 Обобщение

Рассмотренные классы систем объединяются в единое понятие САПР (Система Автоматического проектирования). Существует множество программных пакетов САПР, позволяющих решать поставленные задачи. Например:

- Pro/Engineer - универсальная САПР для промышленных

компаний

- MathCAD - интегрированная система решения математических, инженерно-технических и научных задач
- ANSYS - универсальная система КЭ анализа с встроенным пре-/постпроцессором;
- OpenFOAM - свободно-распространяемая универсальная система КО пространственного моделирования механики сплошных сред;
- CAElinux - дистрибутив операционной системы Линукс, включающий в себя ряд свободных CAE-программ, в том числе OpenFOAM и SALOME.
- ADAMS - система моделирования и расчёта многотельной динамики;
- ФРУНД - комплекс моделирования динамики систем твёрдых и упругих тел;

В рамках дипломного проекта рассматривается концепция сервиса, предоставляющего возможность запускать вычислительные процессы в одном из существующих пакетов САПР. При этом предоставляются лишь сведения о желаемой конфигурации вычислительной виртуальной машины и сама инженерная задача, построенная для этого пакета проектирования. Необходимым требованием является универсальность, то есть возможность адаптировать систему к различным CAE системам, не меняя ее архитектуры.

Проблемы связанные с использованием САПР пакетов основаны в основном на экономической состовляющей. Можно выделить основные затраты:

- Покупка лицензионного программного обеспечения
- Организация вычислительного комплекса, обладающего требуемыми мощностями
- Содержание персонала, обслуживающего этот комплекс
- Обучение инженеров-специалистов работе с аппаратными и программными средствами

Нередко это приводит к невозможности организации подобных вычислений в среде малого и среденего бизнеса, так как затраты могут значительно превышать прибыль.

Рассмотрим существующие подходы к организации программно-аппаратного комплекса.

1.2 Кластерные вычисления и грид-технология

В настоящее время инженерные задачи и программные пакеты для их решения, описанные в предыдущем разделе, требуют достаточно больших вычислительных ресурсов. В случае недостаточных ресурсов решение задачи может занимать очень длительное время (месяцы) или являться вовсе невозможным (нехватка памяти).

1.2.1 Вычисления на суперкомпьютере

Суперкомьютер - вычислительная машина, значительно превосходящая по своим техническим параметрам большинство существующих компьютеров. Как правило, современные суперкомпьютеры представляют собой большое число высокопроизводительных серверных компьютеров, соединённых друг с другом локальной высокоскоростной магистралью для достижения максимальной производительности в рамках подхода распараллеливания вычислительной задачи. Работа осуществялется через терминалы, имеющие доступ к вычислительному комплексу.

1.2.2 Кластерные вычисления

Кластерные вычисления — это особая технология высокопроизводительных вычислений, зародившаяся вместе с развитием коммуникационных средств и ставшая альтернативной использованию сверхмощных компьютеров. Когда стали доступны каналы связи с высокой пропускной способностью, у компьютеров появилась возможность выполнять задачу совместно. Так появилась концепция виртуального суперкомпьютера, где масштабная задача выполняется совместно в единой сети кластером обычных компьютеров. Вычислительные узлы этой сети ведут скоординированную работу, используют ресурсы друг друга и потенциально доступны из любой точки системы. Компьютеры могут быть удалены друг от друга и использовать разные типы коммуникаций, однако для конечного программного продукта и пользователя они играют роль единой вычислительной машины.

Распределённые вычисления представляют собой способ решения трудоёмких вычислительных задач с использованием двух и более компьютеров, объединённых в сеть. Они являются частным случаем параллельных вычислений, т.е. одновременного решения различных частей одной вычислительной задачи несколькими процессорами одного или нескольких компьютеров.Параллельными вычислительными системами называют физические, компьютерные, а также программные системы, реализующие тем или иным способом синхронную обработку данных на многих вычислительных узлах.

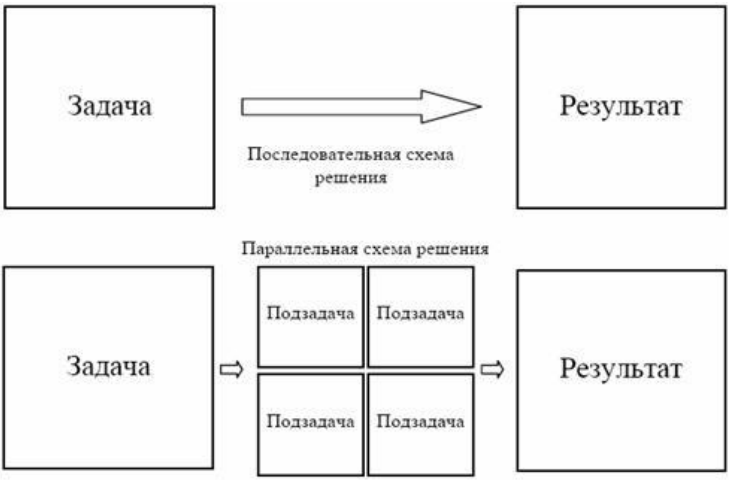

Рисунок 1. Последовательные и параллельные вычисления

1.2.3 Грид-технология

Грид-вычисления (англ. *grid* — решётка, сеть) — это форма распределённых вычислений, в которой «виртуальный суперкомпьютер» представлен в виде кластеров соединённых с помощью сети, слабосвязанных, гетерогенных компьютеров, работающих вместе для выполнения огромного количества заданий (операций, работ). Эта технология применяется для решения научных, математических задач, требующих значительных вычислительных ресурсов. С точки зрения сетевой организации грид представляет собой согласованную, открытую и стандартизованную среду, которая обеспечивает гибкое, безопасное, скоординированное разделение вычислительных ресурсов и ресурсов хранения информации, которые являются частью этой среды, в рамках одной виртуальной организации.

В настоящее время выделяют три основных типа грид-систем:

1. Добровольные гриды — гриды на основе использования добровольно предоставляемого свободного ресурса персональных компьютеров;
2. Научные гриды — хорошо распараллеливаемые приложения программируются специальным образом (например, с использованием Globus Toolkit);
3. Гриды на основе выделения вычислительных ресурсов по требованию (коммерческий грид, англ. *enterprise grid*) — обычные коммерческие приложения работают на виртуальном компьютере, который, в свою очередь, состоит из нескольких физических компьютеров, объединённых с помощью грид-технологий.

1.2.4 Разработанные стандарты организации грид-систем

Разработанный стандарт архитектуры грид получил название *OGSA* (Open Grid Services Architecture – Открытая Архитектура Грид Сервисов). Он основывается на понятии грид-сервиса. *Грид-сервисом* называется сервис, поддерживающий предоставление полной информации о текущем состоянии (потенциально временного) экземпляра сервиса, а
также поддерживающий возможность надежного и безопасного исполнения, управления временем жизни, рассылки уведомлений об изменении состояния экземпляра сервиса, управления политикой доступа к ресурсам, управления сертификатами доступа и виртуализации. Грид-сервис
поддерживает следующие стандартные интерфейсы.

- *Поиск*. Грид приложениям необходимы механизмы для поиска доступных сервисов и определения их характеристик.
- *Динамическое создание сервисов*. Возможность динамического создания и управления службами – это один из базовых принципов OGSA, требующий наличия сервисов создания новых сервисов.

- *Управление временем жизни*. Распределенная система должна обеспечивать возможность уничтожения экземпляра грид-сервиса.
- *Уведомление*. Для обеспечения работы грид приложения наборы грид сервисов должны иметь возможность асинхронно уведомлять другдруга о изменениях в их состоянии.

Первая реализация модели OGSA, разработанная в 2003 г., называлась *OGSI* (Open Grid Service Infrastructure). В связи с тем, что существовавшие тогда стандарты веб-сервисов (к которым относились WSDL,SOAP) не могли обеспечить всех требований, предъявляемых разработчиками к функциональным возможностям грид-сервисов, при создании OGSI потребовалось модифицировать и расширить соответствующие стандарты. Это привело к тому, что совместное использование веб-сервисов и грид-сервисов в одной среде стало невозможным, из-за несо-вместимости базовых стандартов.

Дальнейшие совместные усилия сообщества грид и организаций по разработке стандартов веб-сервисов привело к определению стандартов, соответствующих требованиям грид. В предложенном стандарте *WSRF* (*Web Service Resource Framework*) специфицированы универсальные механизмы для определения, просмотра и управления состоянием удаленного ресурса, что является критически-важным с точки зрения грид. На сегодняшний день реализация модели OGSA посредством стандарта WSRF (и сопутствующих стандартов, таких как WS-Notification и WS-Addressing) является наиболее распространенной в среде грид. В настоящее время, существуют две системы, обеспечивающие инфраструктуру разработки грид-систем в соответствии со стандартами OGSA, реализованными посредством WSRF: Globus Toolkit и UNICORE.

1.2.5 Обзор существующих проектов

Проект Globus начал разрабатываться в конце 1990-х годов для создания и поддержки сервисно ориентированной инфраструктуры для грид - вычислений.

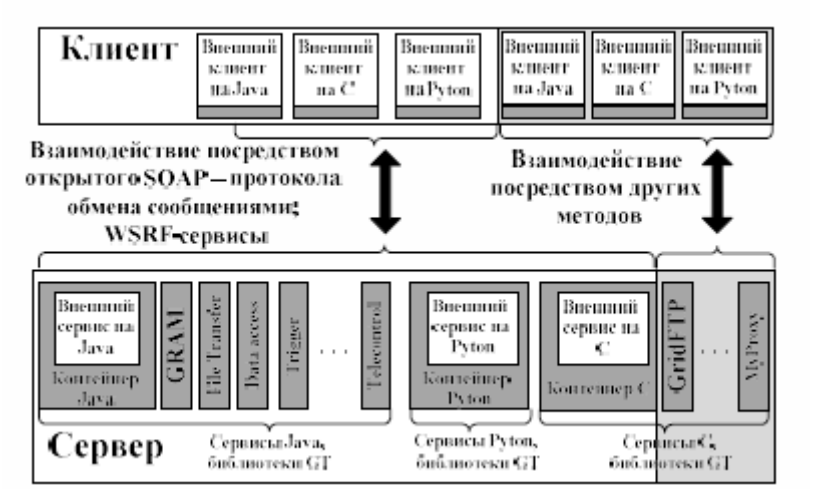

Рисунок 2. Общая схема взаимодействия компонентов Globus Toolkit 4.0.

В составе системы Globus Toolkit 4.0 , первой системы грид-вычислений, обеспечившей полноценную поддержку стандарта WSRF, можно выделить следующие функциональные группы (см. рис. 2):

- базовые сервисы (GRAM – управление ресурсами; File Transfer, GridFTP - передача файлов; Trigger, Index – поиск и каталог ресурсов и др.);
- контейнеры для пользовательских сервисов, поддерживающие аутентификацию, управление состоянием, поиск и т.п. обеспечивающие поддержку стандартов WSRF, WS-Security, WS-Notification;

- библиотеки, для обеспечения взаимодействия сторонних приложений с GTK 4.0 и/или пользовательскими сервисами.

Проект UNICORE (Uniform Interface to Computing Resources – единый интерфейс к вычислительным ресурсам) зародился в 1997 году, и к настоящему моменту представляет собой комплексное решение, ориентированное на обеспечение прозрачного безопасного доступа к ресурсам грид.

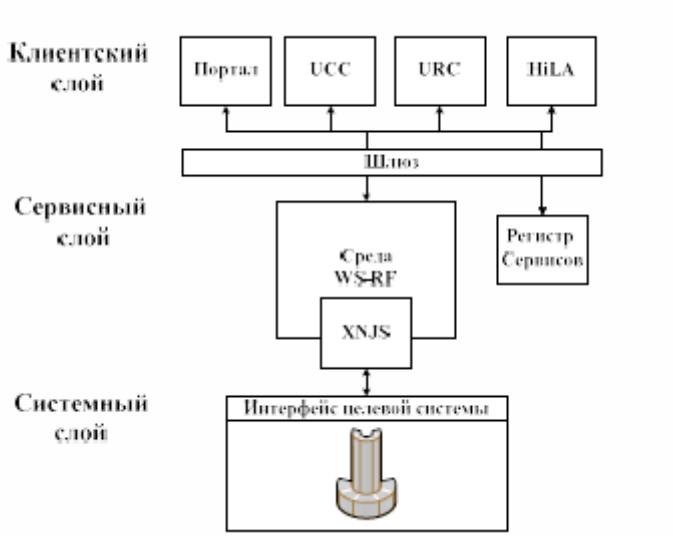

Рисунок 3. Архитектура UNICORE 6

Архитектура UNICORE 6 формируется из клиентского, сервисного и системного слоев (см. рис. 3). Верхним слоем в архитектуре является клиентский слой. В нем располагаются различные клиенты, обеспечивающие взаимодействие пользователей с грид средой:

- UCC (Unicore Command Line Client – клиент командной строки для UNICORE): клиент, обеспечивающй интерфейс командной строки для постановки задач и получения результатов;

- URC (Unicore Rich Client – многофункциональный клиент UNICORE): клиент, основанный на базе интерфейса среды Eclipce, предоставляет в графическом виде полный набор всех функциональных возможностей системы UNICORE;
- HiLA (High Level API for Grid Applications – высокоуровневый программный интерфейс для приложений грид): обеспечивает разработку клиентов к системе UNICORE;
- Порталы: доступ пользователей к грид-ресурсам через интернет, посредством интеграции UNICORE и систем интернет-порталов.

Промежуточный сервисный слой содержит все сервисы и компоненты системы UNICORE, основанные на стандартах WSRF и SOAP. Шлюз –это компонент, обеспечивающий доступ к узлу UNICORE посредством аутентификации всех входящих сообщений. Компонент XNJS обеспечивает управление задачами и исполнение ядра UNICORE 6. Регистр сервисов обеспечивает регистрацию и поиск ресурсов, доступных в грид-среде. Также, на уровне сервисного слоя обеспечивается поддержка безопасных соединений, авторизации и аутентификации пользователей.

В основании архитектуры UNICORE лежит системный слой. Интерфейс целевой системы (TSI – Target System Interface) обеспечивает взаимодействие между UNICORE и отдельным ресурсом грид-сети. Он обеспечивает трансляцию команд, поступающих из грид-среды локальной системе.

Основным достоинством использования системы UNICORE 6 для разработки распределенных вычислительных систем можно считать наличие богатого арсенала различных клиентов, обеспечивающих взаимодействие пользователя с ресурсами вычислительной сети, а также развитых средств обеспечения безопасности при разработке грид-приложений.

1.3 Концепция облачных вычислений

1.3.1 Введение

Альтернативой грид-вычислениям можно рассматривать концепцию облачных технологий, на базе которых можно организовать систему, позволяющую динамически конфигурировать вычислительные ресурсы, обладала бы единым интерфейсом доступа и требовала бы от пользователя минимальных специальных знаний.

1.3.2 Гипервизор

Гипервизор (или Монитор виртуальных машин) —программа или аппаратная схема, обеспечивающая или позволяющая одновременное, параллельное выполнение нескольких операционных систем на одном и том же хост-компьютере. Гипервизор также обеспечивает изоляцию операционных систем друг от друга, защиту и безопасность, разделение ресурсов между различными запущенными ОС и управление ресурсами.

Гипервизор также может предоставлять работающим под его управлением на одном хост-компьютере ОС средства связи и взаимодействия между собой так, как если бы эти ОС выполнялись на разных физических компьютерах.

Обзор наиболее часто используемых гипервизоров:

- **OpenVZ.** OpenVz - платформа виртуализации, встраиваевая в ядро ОС Linux. OpenVZ позволяе запускать на физическом сервере несколько изолированных сущностей ОС, называемых контейнерами. OpenVz поддерживает только операционные системы семейства Linux, такие как Centos, Fedora, Gentoo, and Debian. Одним

из недостатков является невозможность вносить изменения в ядро ОС. Все виртуальные сервера имеют ту же версию ядра, что и хост, на котором они запущены. Однако, так как OpenVz не является чистым гипервизором, он может очень эффективно работать поверх гипервизоров Xen, KVM, VMware.

- **KVM, Xen, VMware**. Следующие три платформы можно рассмотреть в одном разделе, так как они работают практически идентично. Различия в их работе не заметны для работы виртуальных серверов и конечных пользователей. Все три платформы поддерживают истинную виртуализацию ресурсов, не разжедяемых между ядром хоста и другими виртуальными серверами. Любые операционные системы могут быть запущены на этих гипервизорах.

1.3.3 Виртуализация

Виртуализация – это дополнительный уровень изоляции вычислительных процессов и реусурсов по сравнению с тем, что предоставляют сегодня операционные системы.

Таблица 1. Типы вирутальных технологий

Виртуализация серверных ОС	Виртуализация настольных ОС	Виртуализация приложений	Виртуализация представлений
Ключевая идея			
Консолидация нагрузок для более эффективного использования	Использование дополнительных изолированных операционных	Отделение приложений от настольных ОС, использование	Разделение процессов исполнения приложения и визуализации

серверных ресурсов	сред на стандартном ПК	приложений по запросу	пользовательского интерфейса, централизованная обработка и хранение данных, использование тонкого клиента
Эффект применения			
Снижение операционных расходов (оборудование, площадь, электричество) Увеличение доступности и полезного времени Простота аварийного восстановления Уменьшение перерывов в обслуживании	Поддержка унаследованных приложений, несовместимых с новыми ОС Поддержка приложений, не отвечающих корпоративным требованиям Сокращение конфликтов приложений с ОС Ускорение процесса	Снижение конфликтов приложений между собой Сокращение затрат на проведение регрессивного тестирования приложений на совместимость Централизация управления процессом обновления приложений	Сокращение конфликтов приложений с ОС Упрощение обеспечения конфиденциальности данных и соответствия нормативным требованиями Снижение затрат на администрирование настольных систем Возможность использования унаследованного клиентского

Упрощение решения задач масштабирования и балансировки нагрузки	замены ОС		оборудования

1.3.4 Облачные вычисления

Облачные вычисления – это модель позволяющая удобно, по требованию получать через сеть доступ к общему пулу конфигурируемых вычислительных ресурсов (например сети, сервера, хранилища и сервисы), которые могут быть быстро инициализированы с минимальными затратами на управление и взаиодействие с поставщиком услуг. Эта модель облачной среды способствует доступности и основана на пяти основных характеристиках, трех сервис-моделях и четырех моделях размещения.

1.3.5 Основные характеристики

- **Сервис по требованию.** Пользователь может в односторонным порядке запросить вычислительные ресурсы, такие как серверное время и сети хранения данных по необходимости. При этом выделение ресурсов должно происходить автоматически, не требуя человека с каждым сервисом провайдера.
- **Широкий сетевой доступ.** Ресурсы должны быть доступны по сети через стандартные механизмы взаимодействия, которые позволяют использовать гетерогенные клиенты («толстые» или

«тонкие»). Например мобильные телефоны, ноутбуки и персональные цифровые помощники (|PDA)

- **Объединение ресурсов.** Вычислительные ресурсы объединяются для обслуживания множества пользователей. Используется многопользовательская модель с различными физическими и виртуальными ресурсами, динамически нанзначаемыми и передаваемыми в соответсвии с запросами потребителей. При этом пользователь не может контролировать точное местоположение предоставляемых ресурсов, но может конфигурировать расположение ресурсов на более высоком уровне абстракции (например указать страну, округ, датацентр). Ресурсами могут быть хранилища данных, ресурсы процессора, память, пропускная способность сети и виртуальные машины.
- **Эластичность.** Ресурсы могут быстро и упруго, в некоторых случаях автоматически, маштабироваться и освобождаться. Для потребителя ресурсы оказываются неограниченными и могут быть приобретены в любое время.
- **Контроль метрик.** Облачные системы автоматически контролируют и оптимизируют используемые ресурсы, за счет использования возможности измерения метрик на некотором уровне абстракции, соответсвующем типу сервиса (например хранилище данных, ресурсы процессора, пропускная способность, активные пользователи). Использование ресурсов может контролироваться, управляться и публиковаться прозрачно как для поставщика, так и для потребителя услуг.

1.3.6 Модели предоставляемых сервисов

- **IaaS** – инфраструктура как сервис (Infrastructure as a Service). Потребителю предоставляется возможность использовать ресурсы процессора, хранилища данных, сетевое окружение и другие основные вычислительные ресурсы. При этом потребитель имеет возможность размещать и запускать произвольное программное обеспечение, которое включает в себя операционные системы и приложения. Потребитель не имеет контроля над базовой инфраструктурой облака, а может регулировать только работу операционных систем и размещенных приложений и ограниченно управлять или выбирать сетевые компоненты (например хосты, системы защиты).

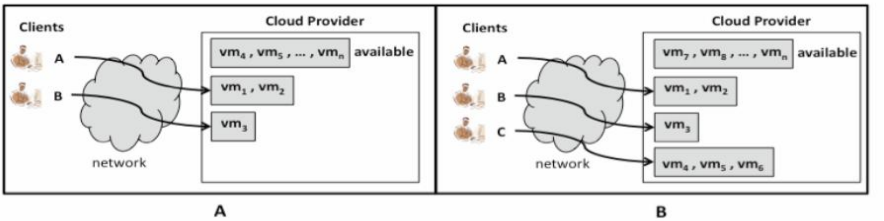

Рисунок 4. Модель IaaS

- **PaaS** – платформа как срвис (Platform as a Service). Потребителю предоставляется возможность размещения в облачной инфраструктуре созданные или приобретенных приложений, разработанных с использованием языков программирования и инструментариев, поддерживаемых провайдером облака. Потребитель не имеет возможность управлять базововой инфраструктурой облака, включающей в себя сетевое

окружение, сервера, операционные системы, хранилища данных или любыми индивидуальными приложениями. Возможное исключение – ограниченная возможность уставливать специализированные конфигурационные настройки.

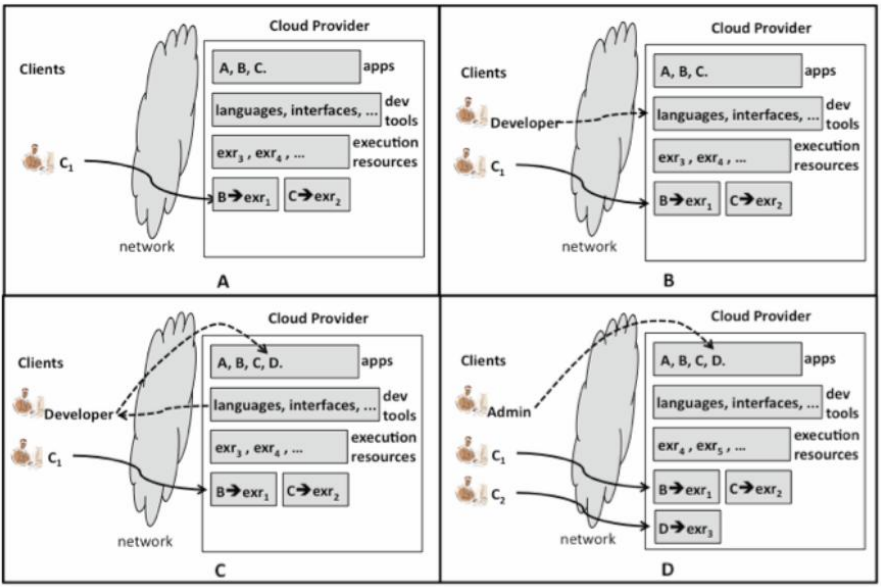

Рисунок 5. Модель PaaS

- **SaaS** – программное обеспечение как сервис (Software as a Service). Потребителю предоставляется возможность использовать готовые приложения провайдера, размещенные в инфраструктуре облака. Данные приложения могут быть задействованы на различных устройсвах пользователей с использование «тонокго» клиента, такого как Веб бразуер. Потребитель не имеет возможность управлять инфраструктурой облака, включая сетевое окружение, сервера, операционные

системы, хранилища данных и внутренние настройки приложения.

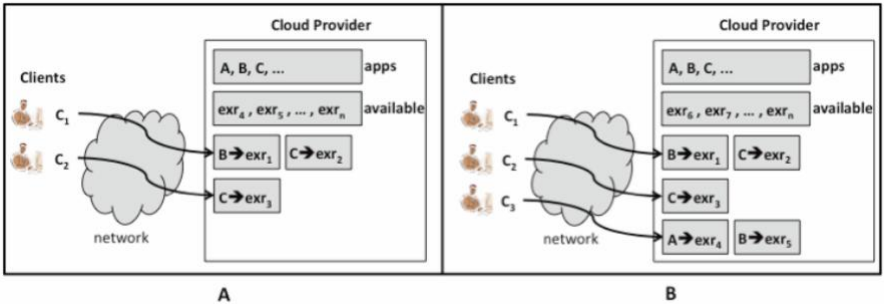

Рисунок 6. Модель SaaS

1.3.7 Модели размещения

- **Частное облако** (Private cloud). Инфраструктура облака эксплуатируется исключительно в рамках организации. Облако может управляться самой организацией или третьей стороной, с размещением в помещения организации или стороннем центре.

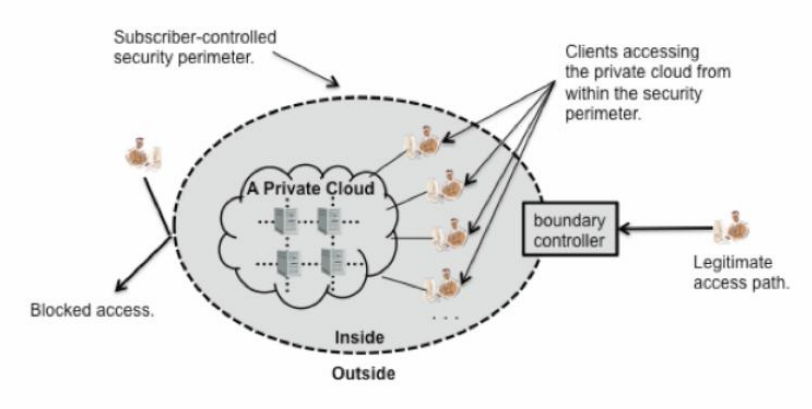

Рисунок 7. Модель частного облака

- **Совместное облако.** Облачные ресурсы разделены между

организациями и поддерживаются определенным сообществом, имеющим общие правила (например требования политики безопасности). Облако может управляться самой организацией или третьей стороной, с размещением в помещения организации или стороннем центре.

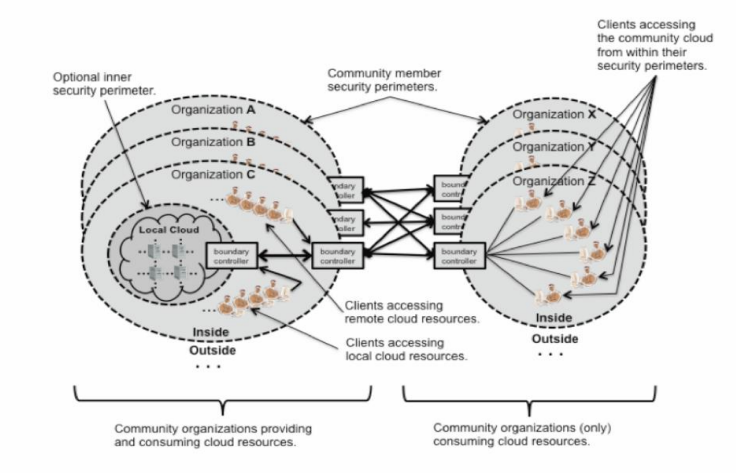

Рисунок 8. Модель совместного облака

- **Общественное облако.** Облачная среда доступна широкой общественности или большим индустриальным объединениям. Управляется организатором продаваемых сервисов.

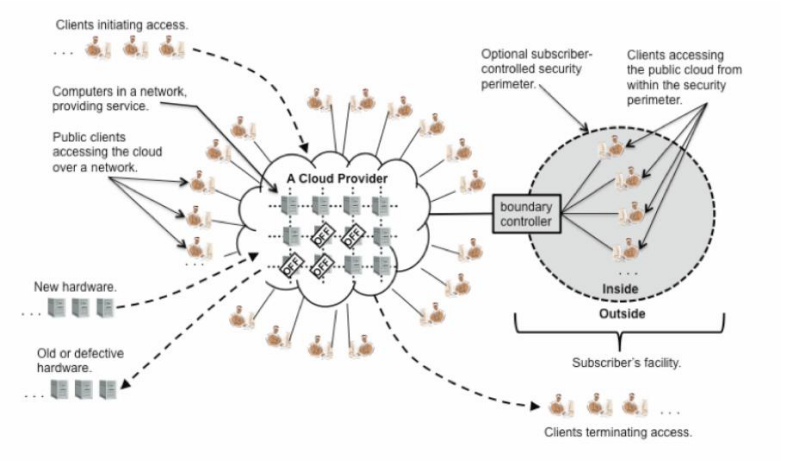

Рисунок 9. Модель общественного облака

- **Гибридное облако.** Инфраструктура представляет собой объединение двух или более облаков (частных, совместных или общественных), которы сохраняют уникальность, но связаны между собой стандартными и технологиями, которые обеспечивают переносимость данных и приложений (например детерминированность для балансировки нагрузки между облаками.)

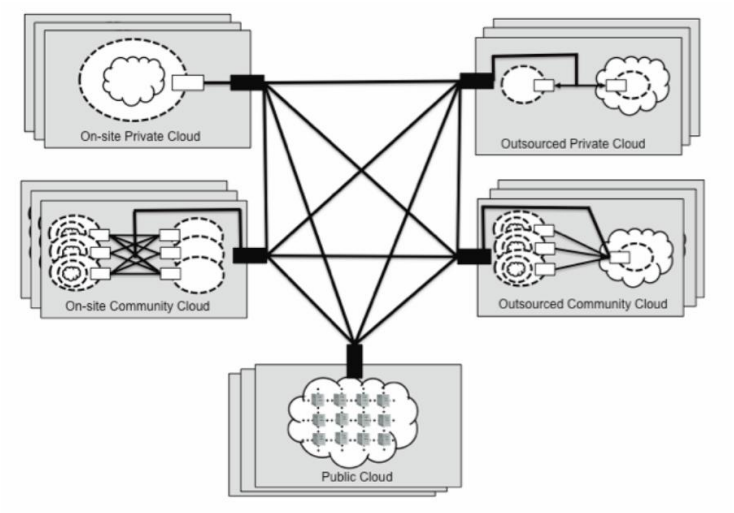

Рисунок 10. Модель гибридного облака

1.3.8 Обзор существующих облачных платформ

В настоящее время облачные технологии являются активно развивающейся областью, количество провайдеров, предоставляющих такие улуги все увеличивается, растет и спрос. Вот наиболее известные проекты, предоставляющие доступ к облачным вычислениям.

- **Amazon EC2** - Amazon Elastic Compute Cloud, веб-сервис, который предоставляет вычислительные мощности в облаке. Поддерживает ОС семейств Linux и Windows. Основан на Eucaliptus.

  С помощью EC2 можно:

  - создать Amazon Machine Image (AMI), который будет содержать ваши приложения, библиотеки, данные и

связанные с ними конфигурационные параметры. Или использовать заранее настроенные шаблоны образов для работы;

- загрузить AMI в Amazon S3(сервис-хранилище от Amazon). Amazon EC2 предоставляет инструменты, для хранения AMI. Amazon S3 обеспечивает безопасное, надёжное и быстрое хранилище для хранения образов;
- использовать Amazon EC2 Веб-сервис для настройки безопасности и сетевого доступа;
- выбирать тип(ы) операционной системы, какой вам необходим, запустить, завершить, или контролировать несколько AMI по мере необходимости, используя API Веб-сервиса, или различных инструментов управления, которые предусмотрены;
- определить необходимость работать в нескольких местах, использовать статический IP или другие варианты;
- платить только за ресурсы, которые вы собираетесь потреблять, такие как время или передача данных.

- **Rackspace** – по функциональности аналогичен Amazon EC2. И так же имеет сервисы хранилища и хостинга приложений. Поддерживает ОС семейств Linux и Windows. Основан на гипервизоре Xen и Citrix Xen Server.
- **Microsoft Azure** - является «облачной» платформой для приложений, позволяющей хранить данные и выполнять приложения в датацентрах Microsoft. Windows Azure предоставляет «облачную» операционную систему, на основе которой работают все сервисы Azure и разработанные

приложения. Внутренняя архитектура данного решения является закрытой. Нет возможности запускать виртуальные машины с ОС, не принадлежащими семейству Windows.

- **DezineForce** – облачный ресурс, реализующий концепцию предоставления сервисов для решения инженерных задач. По доступным описаниям, реализует ту же функциональность, что описана в поставленных задачах диплома. Однако внутренняя архитектура данного ресурса является закрытой, а официальный сайт провайдера недоступен. Поэтому возможности оценить данный сервис в настоящее время нет.

1.3.9 Обобщение

Облачные технологии основаны на средствах виртуализации, основная концепция – предоставление веб-сервисов разного уровня абстракции. Использование облачных вычислений позволяет повысить уровнеь изоляции вычислений по сравнению с операционными системами. Использование этого подхода экономически целесообразно, особенно для небольших компаний и научных подразделений.

1.4 Сравнительный анализ рассмотренных подходов

Предлагается рассмотреть сравнительный анализ подходов, а не конкретных систем, их реализующих. Это целесообразно, учитывая тот факт, что большинство приведенных готовых систем, отвечающим каждому из подходов, предоставляют практически идентичные возможности. Сравнительный анализ удобно представить в виде таблицы:

Обозанчения колонок:

HPC – высокопроизводительные вычисления на суперкомпьютерах

Grid - вычисления с использованием грид-технологий

Cloud – с применением концепции облачных вычислений

Таблица 2. Сравнительный анализ подходов к организации вычислений

критерий	HPC	Grid	Cloud
Центральный сервер управления	Отсутствует	Центральный сервер работает как монитор трафика и устройство распределения задач.	Контроллер Облака (Cloud Controller) следит за состоянием системы и является связующим звеном всех остальных компонентов.
Распределенность	Отсутствует.	Высокая. Грид может объединять узлы со всего мира	Низкая. Может содержаться внутри архитектуры облака или на уровне коммуникации нескольких облаков.
Качество линий связи (пропускная способность)	Высокое. FastEthernet, PCI, InfiniBand	Низкое. Разнородное.	Среднее. Чаще всего Ethernet
Точка доступа	Терминал	Сетевой доступ	Сетевой доступ
Интерфейс доступа	Специализированное проприетарное программное обеспечение.	Специализированное программное обеспечение, часто бесплатное.	Веб-интерфейс (доступ через браузер)
Распределение	Административно	На уровне	Предоставляетс

ресурсов между пользователями	е распределение времени выполнения задачи на сервере.	конфигурации центрального узла и установленной политики распределения ресурсов.	я столько ресурсов, сколько потребляется. Ограничение определяется договором между пользователем и поставщиком.
Стоимость серверного оборудования	Высокая. Больших затрат требуют как отдельные компоненты, так и ввов в эксплуатацию	Низкая. Отдельными узлами могут являться обычные настольные вычислительные машины	Средняя. Невысокая стоимость отдельных компонент и ввода в эксплуатацию.
Специальные знания для использования	Требуются. Необходимо владеть инструментами управления.	Требуются. Необходима настройка и включение узла в грид, навыки по использованию специального программного обеспечения	Не требуются. Веб интерфейс готовых сервисов не представляет сложности для пользователя.
Прогнозировани е времени выполнения задачи	Возможно	Невозможно	Возможно
Адаптируемость среды, возможность внесения изменений	Низкая.	Только на уровне центрального сервера	Высокая.
Возможность управлять вычислительны ми ресурсами на уровне	На уровне узлов системы	Отсутсвует, распределение ресурсов автоматическое	Возможность конфигурирова ть ресурсы самостоятельно с учетом

пользователя			характера задачи

Остальные характеристики, такие как значения производительности, стоимость и др. зависят уже от каждой конкретной реализации одного из изложенных подходов и не влияют на выбор концепции для выполнения цели дипломного проекта.

1.5 Обобщение проведенного анализа

В рамках проведенного анализа были рассмотрены различные подходы к использованию вычислительных ресурсов для решения прикладных инженерных задач. В настоящее время необходимость выполнения вычислений, требующих больших вычислительных мощностей все возрастает. Подобные задачи возникают как в научной сфере и образовании, так и в производстве. Однако создание необходимой вычислительной платформы требует немалых средств, а так же затрат на обучение персонала. Учитывая эти тенденции все более актуальной становится концепция предоставления сервисов, способных выполнять вычислительные задачи с использованием динамически конфигурируемых ресурсов.

Основывая на проведенном анализе сущеуствующих подходов и исходя из характера задач дипломного прокта, для реализации поставленной цели было выбрано решение, использующее концепцию облачных вычислений, модель SaaS и построенное на базе сервис-ориентированной архитектуры. Такая система будет обладать всеми необходимымми свойствами (возможно управления ресурсами, динамическое использование ресурсов , простота интерфейса) при небольших по сравнению с другими затратах. Возможность

динамического распределения ресурсов позволяет одновременно использовать облачную среду в нескольких целях, то есть возможно получение научно-исследовательского доступа наряду с выполняемыми инженерными расчетами. Тогда как получение доступа к вычислительному комплексу связано с определенными административными трудностями.

2 Основная концепция и архитектура системы

Для реализации поставленных задач необходимо изучить платформу облачных вычислений, которая будет использоваться в проекте. Необходимо разработать модуль, управляющий конфигурацией ресурсов и позволяющий загрузить задание на выполнение, а так же модуль, выполняемый на виртуальный машине, контролирующий выполнение вычислений и обработку результата.

2.1 Архитектура облачной среды

На базе вычислительных ресурсов СПбГПУ был создан стенд вычислительной среды уровня IaaS, который построен на основе системы Eucalyptus и гипервизора Xen. Важной особенностей разработанной вычислительной среды является возможность запуска гетерогенных систем, в том числе под управлением ОС Linux, Windows и FreeBSD. На базе этих операционных систем разработаны образы виртуальных машин с различными наборами программного обеспечения, включающими в себя инженерные вычислительные пакеты компьютерного инжиниринга Ansys, Adams, ProEngineer. Распределенная вычислительная среда для решения научно-технических задач представляет разнородное множество вычислительных ресурсов в виде виртуальных машин и имеют в аспекте информационной безопасности ключевые отличия, такие как наличие виртуальных машин пользователей разных групп в рамках одного гипервизора и невозможность контроля виртуализированного трафика классическими средствами разграничения доступа.

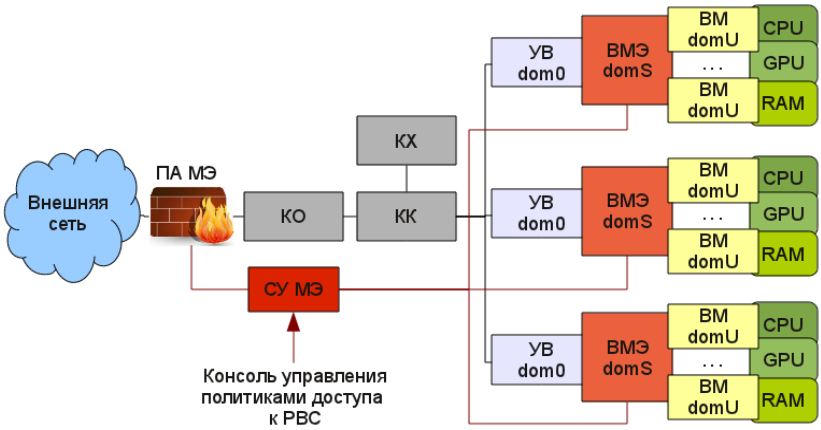

Рисунок 11. Архитектура защищенной вычислительной платформы

Контроллер узла вычислительной среды является мощным многоядерным узлом с установленным гипервизором, на котором функционирует сервисная консоль или домен уровня 0 (dom0 в терминах гипервизора XEN или сервисная консоль в терминах других гипервизоров) и виртуальные вычислительные машины (домен уровня U, domU). Для обеспечения информационной безопасности и разграничения доступа (РД) между виртуальными машинами, функционирующими в рамках одного гипервизора необходимо осуществлять контроль внутреннего («виртуального») трафика и внешнего (поступающего с других гипервизоров и из внешних сетей). Решить задачу разграничения доступа можно путем интеграции в гипервизор виртуального межсетевого экрана, функционирующего в рамках гипервизора, но отдельно от пользовательских виртуальных машин. Домен виртуального межсетевого экрана можно определить как «домен безопасности» (security domain, domS). Важным аспектом при осуществлении контроля сетевого трафика является скрытое функционирование средства фильтрации, межсетевой экран не должен

изменять топологию сетевой подсистемы гипервизора. Достичь этого можно путем использования технологии «Стелс» – осуществления невидимого для других сетевых компонентов контроля пакетного трафика.

В связи с тем, что вычислительная среда используется широким кругом пользователей, то необходимо осуществлять разграничение доступа согласно заданным политикам безопасности, хранящихся в службах каталогов (LDAP, Active Directory).

На рис 11. представлена обобщенная архитектура распределенной вычислительной среды с внедренными средствами разграничения доступа. Используются следующие сокращения: ПА МЭ – программно-аппаратный межсетевой экран, ВМЭ – виртуальный межсетевой экран, СУ МЭ – система управления межсетевыми экранами, ВМ – виртуальная машина, КО – контроллер облака, КК – контроллер кластера, КХ – контроллер хранилища. СУ МЭ решает задачи синхронизации и согласования политик безопасности между компонентами средств защиты информации (ПА МЭ, ВМЭ). При изменении политик доступа, новые правила реплицируются на все компоненты защиты РВС. Предложенный подход позволяет защитить распределенную вычислительную платформу, как от внешних, так и от внутренних угроз. Внедрение прозрачного домена безопасности domS изолирует гипервизор от виртуальных вычислительных машин, что исключает возможность атаки гипервизора из внутренней сети.

Данный вычислительный стэнд используется как для прикладный так и для исследовательских целей.

Доступен по адресу: http://195.208.117.181/www/index.html

SDK для использования: http://community.citrix.com/cdn/xs/sdks/

2.2 Архитектура приложения в целом

На базе представленной вычислительной среды, в рамках дипломной работы бакалавра, разрабатывается проект, реализующий модель предоставления вычислительных ресурсов - программное обеспечение как сервис (SaaS). Выбор сервис ориентированной архитектуры обосновывается универсальностью системы и унифицированным интерфейсом для различных типов решаемых задач.

В структурный состав модели (рис. 2) входят:

1. Универсальный интерфейс, осуществляющий взаимодействие с инженерными пакетами (Solver), включающий реализации для различных инженерных пакетов. Solver представлен в образе виртуальной машины, запускаемой в облаке.
2. Веб сервис, обеспечивающий взаимодействие между интерфейсом Solver и клиентским приложением посредством протокола SOAP.
3. Веб приложение по управлению вычислительными ресурсами и запуском задач в виртуальной машине, являющееся порталом доступа к вычислительной среде.

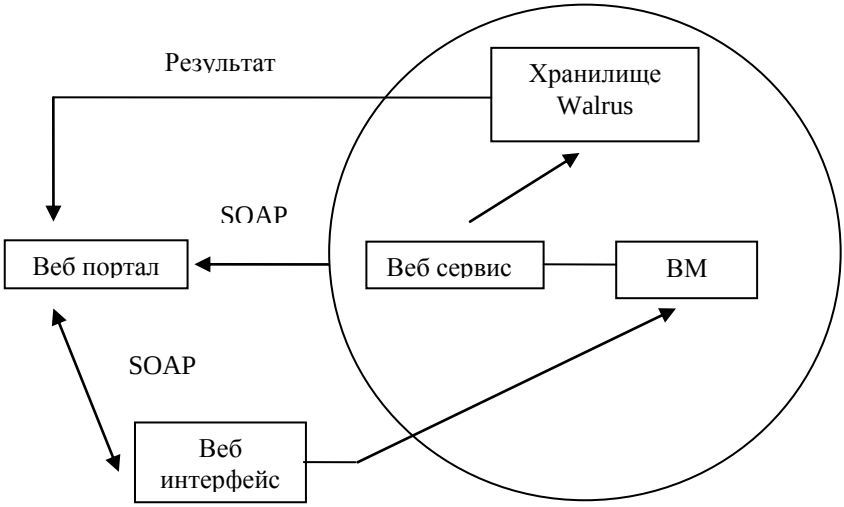

Рисунок 12. Архитектура защищенной вычислительной платформы

Использование распределенной инфраструктуры позволяет пользователю задать конфигурацию и тип запускаемой виртуальной машины. Задание оптимальной конфигурации позволяет оптимизировать время выполнения вычислений. Реализация пользовательского интерфейса как веб приложения обеспечивает простой единообразный доступ к вычислительному сервису вне зависимости от аппаратной и программной составляющих компьютера пользователя – достаточно лишь наличия браузера и доступа к данному сетевому ресурсу.

2.3 Требования к представленной архитектуре

Необходимо рассмотреть основные требования к реализации выбранной архитектуры.

2.3.1 Требования к модулю Solver

Данный модуль предсталяет собой универсальный интерфейс для запуска необходимого САПР пакета и начала выполнения вычислений.

Требования

- Независимость от платформы. Разные пакеты САПР могут работать на разных ОС.
- Универсальность интерфейса. Данный модуль должен легко адаптироваться к любой необходимой САПР системе

2.3.2 Требования к веб-сервису

Данный модуль реализует взамодействие между веб приложением и модулем, рассмотренным выше.

Требоавния:

- Использование разных методов взаимодействия. Данное требование обеспечивает возможность обращения к данному сервису различными способами. В данном проекте предлагается реализовать SOAP И REST интерфейсы.
- Должен поддерживать многопотоковую обработку запросов. Для обеспечения одновременной работы нескольких пользователей.
- Должен поддерживтаь возможность автоматического развертывания.

2.3.3 Требования к веб приложению

Веб приложение представляет собой пользовательский интерфейс и функциональность, позволяющую запустить виртуальную машину и обратиться к веб-сервису из пункта 2.3.2

Требования

- Должен поддерживать многопотоковую обработку запросов. Для обеспечения одновременной работы нескольких пользователей.
- Должен быть реализован как минимум один из методов, позволяющих обрататься к веб-сервису
- Должен содержать систему аутентификации пользователей

2.3.4 Требования к межмодульной коммуникации

Описанные выше модули взаимодействуют согласно слеующей трехзвенной архитектуре:

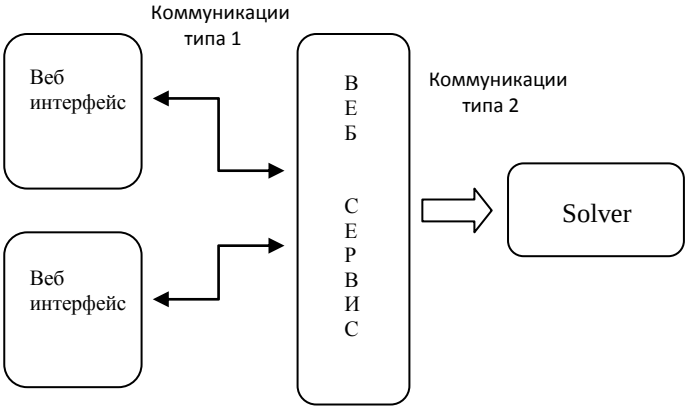

Рисунок 13. Межмодульные коммуникации

- Коммуникации 1го типа

  Реализуются на базе протокола HTTP. Представляют собой SOAP или HTTP запросы. Основным требованием является ассинхронность взаимодействия. Выполнение вычислительной задачи может занимать длительное время, поэтому после

получение веб-сервисом запросом на выполнение вычисление, соединение должно заканчиваться. После окончания вычислительного процесса уже веб-сервис инициирует соединение, оповещая веб-приложение о завершении работы.

- Коммуникации 2го типа

  Являются обычными функциональными вызовами на уровне языка программирования. Модуль Solver встраивается в программную модель веб-сервиса. Основное требование – универсальность вызовов, отсутствие завимости запроса от типа решаемой задачи.

3 Особенности реализации

3.1 Стэк технологий

В качестве платформы разработки был выбран язык программирования Java версии 1.6 и выше. Используемая версия - JDK_1.6.u32. В качестве базы данных Postgresql 9.1. Разработка осуществляется с применением фреймворка Spring Framework. В частности:

- Spring MVC (Model View Controller) для обеспечения взаимодействия веб интерфейса и серверной части, а так же для поддержки REST интерфейса
- Spring WS (Web Services) для создания веб-сервиса и обеспечения взаимодействия с ним и с веб интерфейсом облачной инфраструктуры.
- Spring DB (Data Base) для обеспечения взаимодействия приложения с базой данных
- Spring Security для обесечения информационной безопасности системы (создание сессии, механизмы аутентификации пользователей)

Наряду с этими технологиями в проекте были применены следующе средства:

- Ant – основанный на Java скриптовый язык для развертывания приложений, выполнения скриптов баз данных, сборке архивов итд.
- Apache Log4J – система логгирования, ориентированная на Java.
- Apache tomcat = веб сервер для развертывания веб приложений

3.2 Общая концепция реализации

Рассмотрим алгоритмическую структуру предложенного решения представленную ввиде графа на уровне методов и межмодульного взаимодействия. Для удобства представления разобьем весь процесс на основные этапы:

I. Этап 1. Взаимодействие с пользователем.

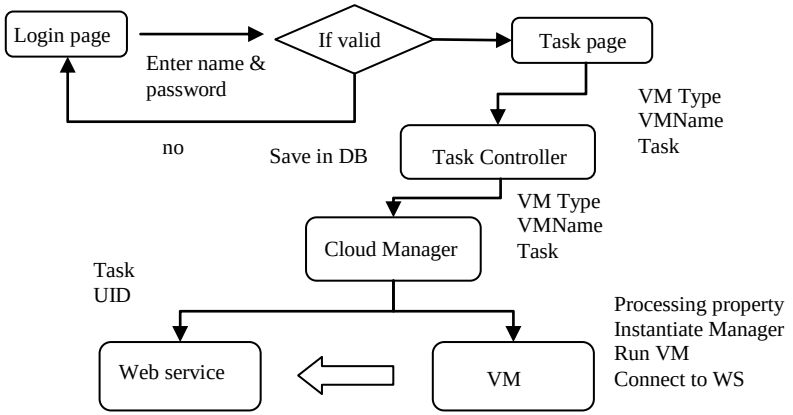

Рисунок 14. Взаимодействие с пользователем

II. Этап 2. Обработка полученных данных. Инициализация вычислений

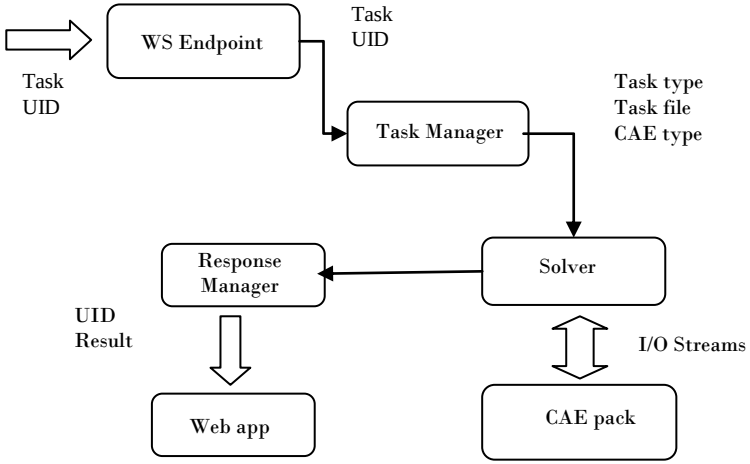

Рисунок 15. Инициализация вычислений

III. Этап 3. Оповещение об окончании выполнения задачи.

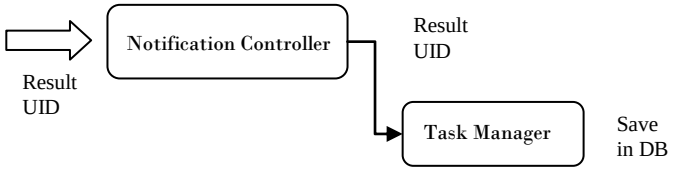

Рисунок 16. Алгоритм оповещения

3.3 Веб – сервис

XML Schema -язык описания структуры XML-документа. Спецификация XML Schema, является рекомендацией W3C. В данном проекте используется для описания SOAP сообщений. В качестве примера можно привести

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
       elementFormDefault="qualified"
targetNamespace="http://www.icsengineerservice.ru/solveTask"
       xmlns:hr="http://www.icsengineerservice.ru/solveTask>
 <xs:element name="solveTaskRequest ">
   <xs:complexType>
     <xs:sequence>
       <xs:element name="taskName" type="xs:string" />
       <xs:element name="uid" type="xs:int" />
       <xs:element name="taskType" type="xs:string" />
       <xs:element name="encodedTask" type="xs:base64Binary" />
     </xs:sequence>
   </xs:complexType>
 </xs:element>
 <xs:element name="solveTaskResponse">
   <xs:complexType>
     <xs:sequence>
       <xs:element name="uid" type="xs:string" />
       <xs:element name="respone" type="xs:string" />
     </xs:sequence>
    </xs:complexType>
 </xs:element>
</xs:schema>
```

Даннаой xsd схеме соответсвует класс, описывающий те же поля, который затем средствми Spring WS инициализирует SOAP пакет, который затем отправляет веб-сервису.

3.4 Base application

В данном разделе будут более подробно рассмотрены отдельные классы модуля основного веб-приложения. И приведени примеры кода.

3.4.1 AbstractCloudManager

Данный класс описывает базовую функциональность запуска и остановки виртуальной машины на уровне интерфейса. Так как данный класс является абстрактны, невозможно создать сущность такого типа. Классом, реализующим данную функциональность через наследование, является XCPCloudManager.

```
/**
 * This abstract class describes all manipulations with cloud
service
 * To use it you should extend this class and make
functionality.
 * @author Lukashin {@link= "an.lukashin@gmail.com"}
 *
 */
public abstract class AbstractCloudManager {
  private Properties props;
  private static final Logger logger =
Logger.getLogger(AbstractCloudManager.class);

  public String runVM(String properyName,String vmName){
    try {
        initProperties(properyName+".properties");
    } catch (FileNotFoundException e) {
        logger.error("No such properties. Name =
        "+properyName);
        return "Error";
    } catch (IOException e) {
        logger.error("I/O exception occured PropertyName =
        "+properyName);
        return "Error";
     }
     return runVirtualMachine(props,vmName);
  }

/**
* Get connection properties from property file. Than request to
* run VM
* @param props
* @return result string with VM IP
*/
  public abstract String runVirtualMachine(Properties props,
String VMName);

/**
* Than request to stop VM with this name
* @param props
* @return result
*/
  public abstract String shutDownVirtualMachine(String vmName);

  private void initProperties(String propertyName) throws
FileNotFoundException, IOException{
    props=new Properties();
    props.load(new FileInputStream(propertyName));
  }
}
```

3.4.2 Controllers

Класс типа Controller реализует функциональность, необходимую для обработки http запросов с веб интерфейса или другого сервера. В качестве примера можно рассмотреть класс StartVMController

```
/**
 * this controller process request like
startVM.html?_vmName=&_vmType=
 * and the VM parameters for future
 * @author Lukashin
 *
 */
public class StartVMController implements Controller{
  private static final Logger logger =
Logger.getLogger(StartVMController.class);
  private static final String PARAMETER_VM_NAME = "_vmName";
  private static final String PARAMETER_VM_TYPE = "_vmType";
  private static final String VIEW_NAME_SUCCESS = "success";
  private static final String VIEW_NAME_FAIL = "error";
/*
 * Processing HTTP request.
 * @see
org.springframework.web.servlet.mvc.Controller#handleRequest(ja
vax.servlet.http.HttpServletRequest,
javax.servlet.http.HttpServletResponse)
*/
@Override
  public ModelAndView handleRequest(HttpServletRequest request,
HttpServletResponse response) throws Exception {
    logger.info("StartVmController started");
    ModelAndView mav= new ModelAndView();
    String vmName=request.getParameter(PARAMETER_VM_NAME);
    String vmType=request.getParameter(PARAMETER_VM_TYPE);
    if (Util.isEmpty(vmName)||Util.isEmpty(vmType)) {
      logger.error("Empty properties recieved from request");
      mav.setViewName(VIEW_NAME_FAIL);
     return mav;
    }
    AbstractCloudManager cloudManager=new XCPCloudManager();
    String result=cloudManager.runVM(vmType, vmName);
   logger.info(result);
    //TODO process result
    mav.setViewName(VIEW_NAME_SUCCESS);
  return mav;
  }

}
```

3.5 Frontend

В настоящее время проект не имеет общирного пользовательского интерфейса. Для его формирования используются JSP (Java Server Page). В качестве примера может быть представлен код страницы login.jsp.

```
<%@page pageEncoding="UTF-8" %>
<%@page session="true" contentType="text/html; charset=UTF-8"
%>
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core_rt" %>
<%@ taglib uri="http://www.springframework.org/tags"
prefix="spring" %>
<html>
  <head>
    <title>Login</title>
  </head>
  <body>
    <h1>Login</h1>
    <c:if test="${not empty param.login_error}">
      <font color="red">
        Your login attempt was not successful, try
again.<br/><br/>
        Reason: <c:out
value="${SPRING_SECURITY_LAST_EXCEPTION.message}"/>.
      </font>
    </c:if>
    <form name="f" action="<c:url
value='j_spring_security_check'/>" method="POST">
      <table>
        <tr><td>User:</td><td><input type='text'
name='j_username' value='<c:if test="${not empty
param.login_error}"><c:out
value="${SPRING_SECURITY_LAST_USERNAME}"/></c:if>'/></td></tr>
        <tr><td>Password:</td><td><input type='password'
name='j_password'></td></tr>
        <tr><td><input type="checkbox"
name="_spring_security_remember_me"></td><td>Don't ask for my
password for two weeks</td></tr>
        <tr><td colspan='2'><input name="submit"
type="submit"></td></tr>
        <tr><td colspan='2'><input name="reset"
type="reset"></td></tr>
      </table>
    </form>
  </body>
</html>
```

3.6 Организация процесса разработки

Для оптимизации процесса разработки программного продукта, для увеличения безопасности (отказ оборудования, потеря данных) были применены следующие средства:

- РСКВ (распределенная система контроля версий) Git. Дающая возможность сохранения различных версий документа, их сравнения, отката изменений и.т.д.
- GitHub – итернет ресурс, предоставляющий бесплатную возможность удаленного размещения git репозиториев.
- Eclipse – свободная интегрированная среда разработки модульных кроссплатформенных приложений. Развивается и поддерживается Eclipse Foundation.

4 Анализ полученных результатов

4.1 Обзор проделанной работы

В рамках данной дипломной работы был проведен анализ вопросов, связанных с концепцией вынесения вычислительных процессов в распределенную среду, предоставляющую возможность решения прикладной задачи в качестве сервиса. По результатам проведенного анализа была разработа архитектура такого сервиса на базе облачной инфраструктуры. Данный подход представляет собой более высокий уровень изоляции пользователя от аппаратных ресурсов. Более того, используемая архитектура позволяет абстрагироваться от среды исполнения и оперировать системой на уровне задания конфигурации вычислительного ресурсы (ВМ) и внесение задания на вычисления. Подобная система отвечает поставленной цели дипломной работы и выполняет сформулированные задачи. Выбранный стэк технологии делает систему гибкой к изменениям и унифицированной с точки хрения использования. Применение фреймворков разработки программного обеспечения обеспечивает высокое качество реализации и соответствие концепция разработки Enterpice приложений.

4.2 Перспективы применения

Предложенная реализация модели SaaS по организации сервиса наукоемких вычислений, позволяет обеспечить потребности научных коллективов Политехнического университета в вычислительных ресурсах без дополнительных затрат на программное обеспечение и аппаратные средства. При этом система полностью удовлетворяет требованиям информационной безопасности и обеспечивает разграничение доступа, являясь при этом универсальным и масштабируемым вычислительным средством.

4.3 Возможное развитие

На данном этапе развития рассмотренной концепции система может использоваться в качестве демонстрационной и для решения узкого круга задач. Учитывая универсальность предложенного подхода, возможно расширение функциональности системы (например конфигурирование серды для работы инстурментов автоматического тестирования и выполнение тестовых наборов), увеличения числа поддерживаемых пакетов САПР. Необходима доработка пользовательского интерфейса приложения.

- Разработка единого дизайна
- Создание личных кабинетов пользователей
- Расширение возможностей существующего интерфейса

В качестве облачной лпатформы использовался научно-исследовательский стенд, созданный на базе кафедры телематики факультета при ЦНИИ РТК. В дальнейшем, при увеличении ангрузки целесообразно перенести данный сервис на общеуниверситетскую вычислительную платформу.

Список используемых материалов

1. Лукашин А.А., Лукашин А.А., Тютин Б.В., Котляров В.П. Архитектура сервиса для решения ресурсоемких задач в распределенной вычислительной среде // НТВ СПБГПУ Информатика. Телекоммуникации. Управление. № 4 (128) 2011 г. – Санкт-Петербург.: Изд-во СПБГПУ

2. Лукашин А.А., Тютин Б.В. Система для реализации распределенных вычислений на основе облачной архитектуры//Технологии Microsoft в теории и практике программирования. - Санкт-Петербург: Изд-ва Политехнического университета, 2011 - С. 101-102.
3. В. С. Заборовский, А. А. Лукашин, С. В. Купреенко, В. А. Мулюха Архитектура системы разграничения доступа к ресурсам гетерогенной вычислительной среды на основе контроля виртуальных соединений. // Вестник УГАТУ. Управление, вычислительная техника и информатика, Т.15 № 5 (45) 2011 г. – Уфа.: Изд-во Уфимского авиационного техничекого университета, 2011.
4. Радченко Г.И. Сервисно-ориентированный подход к использованию систем инженерного проектирования и анализа в распределенных вычислительных средах // Диссертация на соискание ученой степени кандидата физико-математических наук Южно-Уральский Государственный Университет 2009
5. САПР портал: http://www.techgidravlika.ru/view_post.php?id=45
6. Интернет ресурс http://www.ibm.com/developerworks/ru/library/l-grid
7. Xen Cloud Platform. Документация и SDK http://www.xen.org/products/cloudxen.html