**СУППЗ ПИРАМИДА**

**Слайд 2.** Во многих случаях последовательные задачи можно с удовлетворительной эффективностью в автоматическом режиме привести к виду параллельных задач. При этом не потребуется помощь специалиста по параллельному программированию.

На кафедре разработан программный комплекс «Пирамида». Он представляет собой систему, позволяющую организовать параллельное выполнение последовательной программы с распараллеливанием по данным посредством запуска множества ее экземпляров на иерархически организованной вычислительной системе.

На центральном сервере функционирует центральный менеджер. Он запускает менеджеры кластеров на серверах кластеров. Серверы кластеров запускают менеджеры вычислительных модулей на вычислительных модулях.

Если помимо ВМ в состав кластера входит другой кластер, то менеджер кластера верхнего уровня запускает на сервере управления подчиненного кластера соответствующий менеджер кластера.

Менеджер ВМ запускает один или несколько экземпляров ОПП и контролирует их выполнение.

На каждом вычислительном модуле выполняется одна и та же копия программы с различными значениями входных параметров.

«Пирамида» повышает отказоустойчивость вычислений. Выход из строя одного или нескольких вычислительных модулей, а также одного или нескольких кластеров, не останавливает расчеты, а только снижает их скорость.

**Слайд 3.** Коллективный доступ пользователей к вычислительной установке с возможностью использования на ней ПК «Пирамида» позволяет организовать система управления прохождением параллельных задач (СУППЗ). СУППЗ – обеспечивает выделение и освобождение ресурсов вычислительной установки, а также ведение очередей на запуск задач.

На управляющей ЭВМ с установленной СУППЗ функционируют сервер запросов, сервер очередей и сервер запуска.

Сервер запросов выполняет обработку пользовательских команд. В качестве команды может выступать запрос на выполнение задания, запрос на остановку выполнения задания, запрос на удаление задания из очереди задач, запрос состояния задания, запрос состояния очереди задач и другие.

Сервер очередей осуществляет планирование заданий и обеспечивает постановку их на выполнение, для этого сервер очередей вызывает сервер запуска.

Сервер запуска осуществляет запуск и останов заданий.

**Слайд 4.** Задача совмещения ПК Пирамида и СУППЗ решена ранее, в ходе дипломной работы выпускника 732 кафедры Кондратьева.

Для каждого задания СУППЗ динамически выделяет вычислительные модули. В случае заданий для Пирамиды автоматически конфигурируется иерархическая вычислительная система.

**Слайд 5.** Актуальной задачей является разработка прозрачного веб-интерфейса к ПК «Пирамида». Интерфейс должен быть построен таким образом, чтобы избавить пользователя от необходимости изучения работы ПК «Пирамида» и СУППЗ при постановке заданий на обработку.

Кроме этого, возможности ПК «Пирамида» по организации вычислений с распараллеливанием по данным делают целесообразным построение на его базе специализированного облачного сервиса вида SaaS.

Разработка подобного облачного сервиса и является целью настоящей дипломной работы.

Предоставление ПК «Пирамида» в виде облачного сервиса с веб-интерфейсом позволит снизить порог вхождения пользователей В настоящее время, пользователи ПК «Пирамида» вынуждены вручную выполнять подготовительные действия по составлению паспорта заданий и конфигурации системы. При выполнении задания на нескольких комплексах «Пирамида» подготовительные действия фактически дублируются для каждого комплекса.

В распоряжении организации предоставляющей ПК «Пирамида» как сервис может быть несколько вычислительных установок. При этом вычислительные установки могут быть распределены территориально. Облачный сервис в этом случает должен выполнять роль дополнительного уровня абстракции, позволяющего объединить вычислительные установки и организовать к ним единый интерфейс управления

**ТРЕБОВАНИЯ**

**Слайд 8.** Разобьем общую задачу на составляющие подзадачи, определяющие требования в соответствии с классификацией, приведенной на рисунке.

Общей задачей облачного сервиса является организация многопользовательского доступа к ПК «Пирамида» посредством совместно работающих комплексов СУППЗ. Выполнение этой задачи облачным сервисом является функциональным требованием бизнес уровня.

**Слайд 9.** Для роли «пользователь» облачный сервис должен предоставлять следующие интерфейсы:

* регистрации и авторизации пользователя;
* формирования заданий пользователя;
* отслеживания состояния и управления задачами пользователя.

Для роли «администратор» должен быть предоставлен интерфейс управления учетными записями пользователей.

**Слайд 10.**

* Доступ к сервису осуществляется посредством web-интерфейса и возможен только для авторизованных пользователей;
* пользователи должны быть изолированы друг от друга – работа пользователя не влияет на работу других пользователей.

**Слайд 11.**

* Сервис должен отслеживать состояние соединений с СУППЗ и производить запуск заданий по доступным соединениям;
* Взаимодействие сервиса с вычислительными установками должно осуществляться через ssh-соединение.

**АНАЛИЗ ГОТОВЫХ РЕШЕНИЙ**

**Слайд 12.** В соответствии с эталонной архитектурой облачных вычислений, предложенной NIST, сервис вида SaaS может быть построен на основе более гибких PaaS и IaaS решений. Изучение открытых решений облачных сервисов привело к выводу, что все ведущие SaaS решения построены именно таким образом.

Рассмотренные решения построения облачных сервисов, направлены на формирование сервисов общего назначения. В готовых решениях не было найдено модулей, позволяющих организовать взаимодействие облачного сервиса с установленной на вычислительных установках СУППЗ по протоколу SSH. Такой модуль, а в результате набор модулей, необходимо разрабатывать самостоятельно. Другие модули рассмотренных готовых решений избыточны в рамках настоящей дипломной работы.

В работе не использовались готовые решения для построения облачных сервисов и их модули.

**ИНСТРУМЕНТЫ**

**Слайд 13.** Разработка облачного сервиса подразумевает работу над тремя составляющими:

* клиентская часть приложения;
* серверная часть приложения;
* интерфейс взаимодействия клиентской и серверной частей приложения.

При выборе инструментальных средств разработки облачного сервиса я руководствовался следующими критериями:

* новизна и перспективность технологий;
* практика использования в известных, крупных проектах;
* полнота документации и объем сообщества.

**Серверная часть облачного сервиса**

Взаимодействие между клиентской и серверной частями выполняется по протоколу HTTP. Для такого взаимодействия в серверной части облачного сервиса должен функционировать веб-сервер. Выбран веб-сервер Apache.

Логика серверной части сервиса разработана на языке программирования Ruby с использованием шаблона разработки веб-приложений Ruby on Rail. Ruby библиотека Passenger организует взаимодействие RoR-приложения c веб-сервером Apache.

Логику серверной части облачного сервиса можно разбить на 3 наиболее крупных модуля:

* модуль системы аутентификации;

основан на Ruby библиотеке Devise. Библиотека позволяет разрабатывать собственные системы аутентификации, используя современные алгоритмы авторизации и взаимодействия с пользователем.

* модуль взаимодействия с вычислительными установками по протоколу SSH основан на Ruby библиотеке Net-ssh-shell.

В качестве СУБД выбрана MySQL. Работа с реляционной БД ведется через объектное представление ActiveRecord.

**Клиентская часть облачного сервиса**

Разработка клиентской части веб-приложения предполагает использование стандартного стека веб-технологий: HTML5, CSS3, JavaScript.

Мной выбран шаблон проектирования Angularjs от корпорации Google.

Модуль Angular-device средствами одноименной JavaScript библиотеки организует клиентские функции механизма аутентификации Device.

Клиентское приложение имеет локальное браузерное хранилище localStorage (Web Storage по спецификации HTML5) объемом 10 МБ для браузера Chrome и 5 МБ для других браузеров.

**СТРУКТУРА**

**Слайд 14.** Взаимодействие клиентской и серверной частей сервиса строится в соответствии с архитектурой RESTful. Данные, которыми обмениваются клиент и сервер организованы в единый формат JSON.

*Сервисы REST API* клиентской части сервиса преобразуют запросы контроллеров в формат JSON и производят обмен данными с сервером по протоколу HTTP.

*Контроллеры REST API* серверной части сервиса в ответ на запрос пользователя производят необходимые действия с БД или СУППЗ и отправляют ответ клиенту.

Перед запуском задания из доступных пользователю SSH-соединений выбирается следующее за соединением с токеном. Если соединение работоспособно, токен переходит к выбранному соединению и задание запускается.