Лекция № 13

Распределенные вычислительные системы

Юрий Иванович Молородов
yumo@ict.sbras.ru
Декабрь 2012г.

Содержание курса

1. Основы распределенных вычислительных систем
2. Системы переноса кода
3. Агентные технологии
4. Технологии одноранговых вычислений
5. Объектные и компонентные распределенные системы
6. Сервис-ориентированные технологии и веб-сервисы
7. Грид-технологии

Распределенная вычислительная система

«***Распределенной вычислительной системой*** называется такая система, в которой отказ компьютера, о существовании которого вы даже не подозревали, может сделать ваш собственный
компьютер непригодным к использованию»

Лесли Лампорт,
Microsoft Corporation

«***Распределенная вычислительная система (РВС)***
– это набор соединенных каналами связи независимых компьютеров, которые с точки зрения пользователя некоторого программного обеспечения выглядят единым целым»

Э. Таненбаум

Распределенная вычислительная система

Здесь указано на два существенных момента:

- ✓автономность узлов распределенной ВС и
- ✓представление системы пользователю, как единой структуры.

Такой подход к определению РВС имеет свои недостатки. Например, все используемое в такой системе программное обеспечение могло бы работать и на одном единственном компьютере, однако с точки зрения приведенного выше определения такая система уже перестанет быть распределенной.

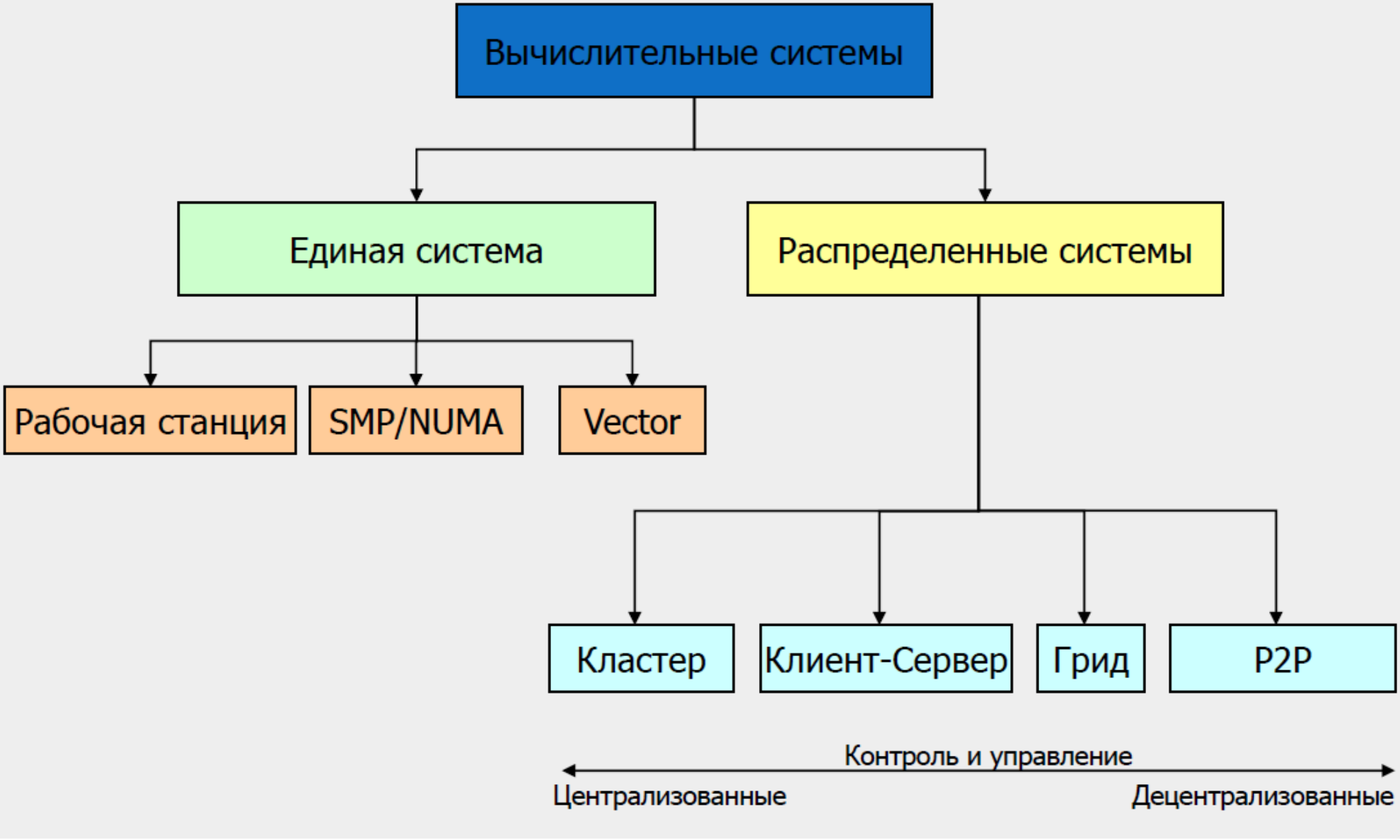
Распределенная вычислительная система

Сама формулировка «*выглядят единым целым*» также является весьма расплывчатой. Порой трудно четко выделить ситуации, в которых «набор соединенных каналами связи независимых компьютеров» работает, как сеть рабочих станций, или как распределенная ВС. С точки зрения аппаратного обеспечения, топологии сети и т.д. разницы между этими двумя ситуациями нет никакой.

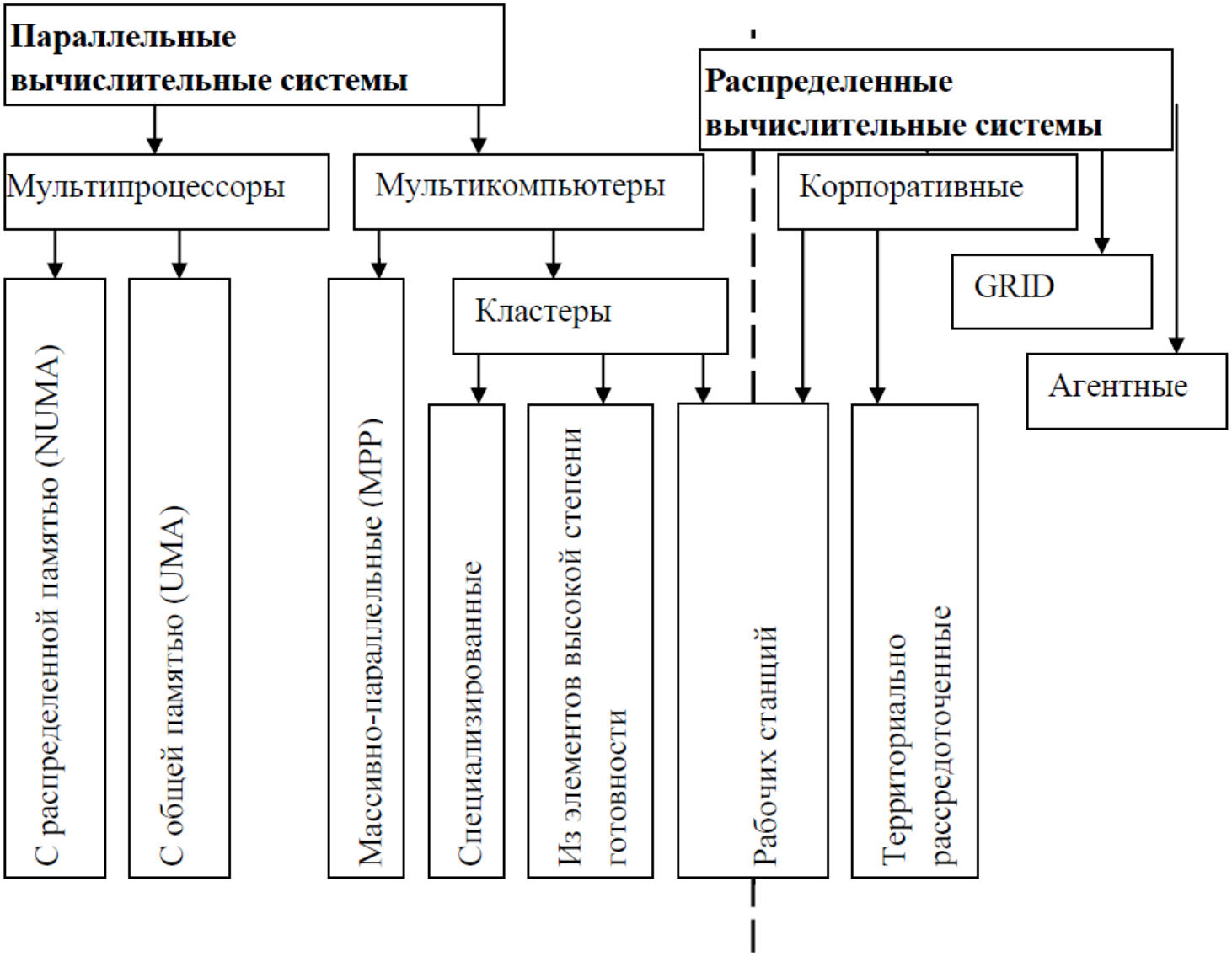
Значит, можно сделать вывод о тем, что распределенная ВС создается программным обеспечением.

Поэтому понятие распределенной системы, должно основываться на анализе образующего такую систему программного обеспечения.

Виды вычислительных систем



Распределенные объектные технологии



Распределенные объектные технологии

Термины «распределенная вычислительная система» и «параллельная вычислительная система» достаточно близки между собой, определим соотношение между этими типами вычислительных систем. Они базируются на используемых способах организации оперативной памяти.

Такой поход позволяет различать два важных типа параллельных систем:

– мультипроцессоры, или системы с общей разделяемой памятью;

– мультикомпьютеры, или системы с распределенной памятью.

Распределенные объектные технологии

Для более детальной классификации мультипроцессоров учитывается способ построения общей памяти.
Возможны следующие варианты:
– использование единой (централизованной) общей памяти.
– использование физически распределенной памяти.
Мультикомпьютеры (системы с распределенной памятью) уже не обеспечивают общий доступ ко всей имеющейся в системах памяти. Такой подход используется при построении двух важных типов многопроцессорных вычислительных систем:
– массивно-параллельные системы (Massively Parallel Processor, MPP);
– кластеры (clusters).

Распределенные объектные технологии

MPP имеют более скоростные, как правило, специализированные, каналы связи между модулями и широкие возможности по масштабированию.

Кластер образуется из модулей, объединенных системой связи или разделяемыми устройствами внешней памяти, например, дисковыми массивами. В настоящее время для образования кластеров используются либо специализированные фирменные средства (например, MEMORY CHANNEL (DEC)), либо универсальные локальные и глобальные сети такие, как Ethernet, другие сети, например, с протоколами TCP/IP, либо дисковые массивы. Размер кластера варьируется от нескольких модулей до нескольких десятков модулей.

Возможна организация кластеров на базе уже существующих сетей рабочих станций.

Т.е. рабочие станции пользователей могут использоваться в качестве узлов кластера ночью и в нерабочие дни.

Системы такого типа называют *COW* (*Cluster of Workstations*) – кластеры рабочих станций.

Терминология РВС

В сфере распределенных вычислительных систем используется набор терминов для обозначения элементов сети: узел, агент, сервер, сервис и т.п.

- *Ресурсом* называется это любая программная или аппаратная сущность, представленная или используемая в распределенной сети. Например: компьютер; устройство хранения; файл; коммуникационный канал; сервис и т.п.
- Узел – это общий термин, обозначающий любое устройство в распределенной вычислительной системе.

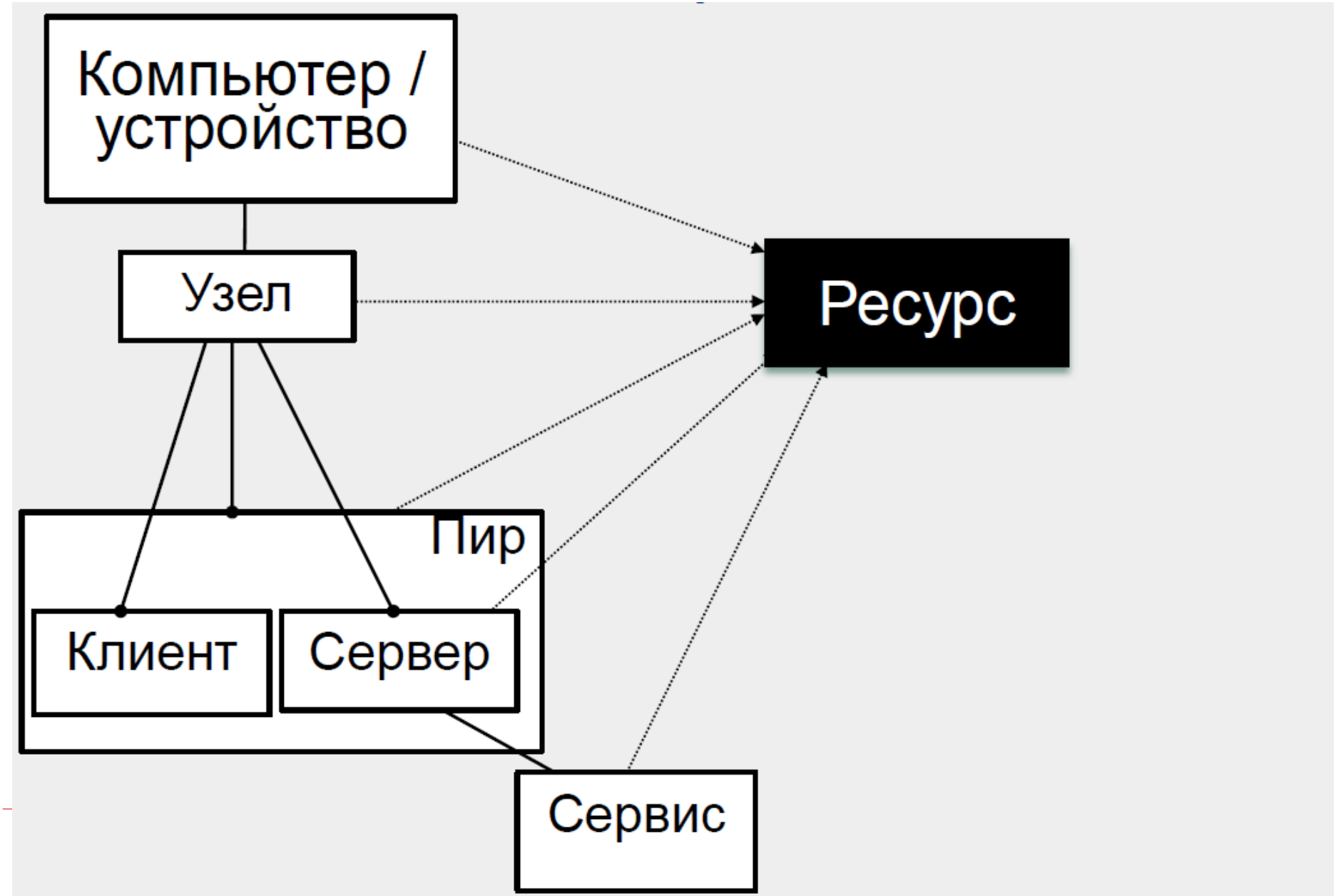
Узел, предоставляющий одну (или несколько) функциональных возможностей часто называют *сервисом*.

Терминология РВС

- *Сервер* – это поставщик информации в РВС (например, веб-сервер).
- *Клиент* – это потребитель информации в РВС (например, веб-браузер).
- Пир – это узел, совмещающий в себе как клиентскую, так и серверную часть (т.е. и поставщик и потребитель информации одновременно).
- Сервис – это сетевая сущность, предоставляющая определенные функциональные возможности (например, веб-сервер может предоставлять сервис передачи файлов по протоколу HTTP). В рамках одного узла могут предоставляться несколько различных сервисов.

Ниже видны взаимоотношения между этими терминами.

Основные термины РВС

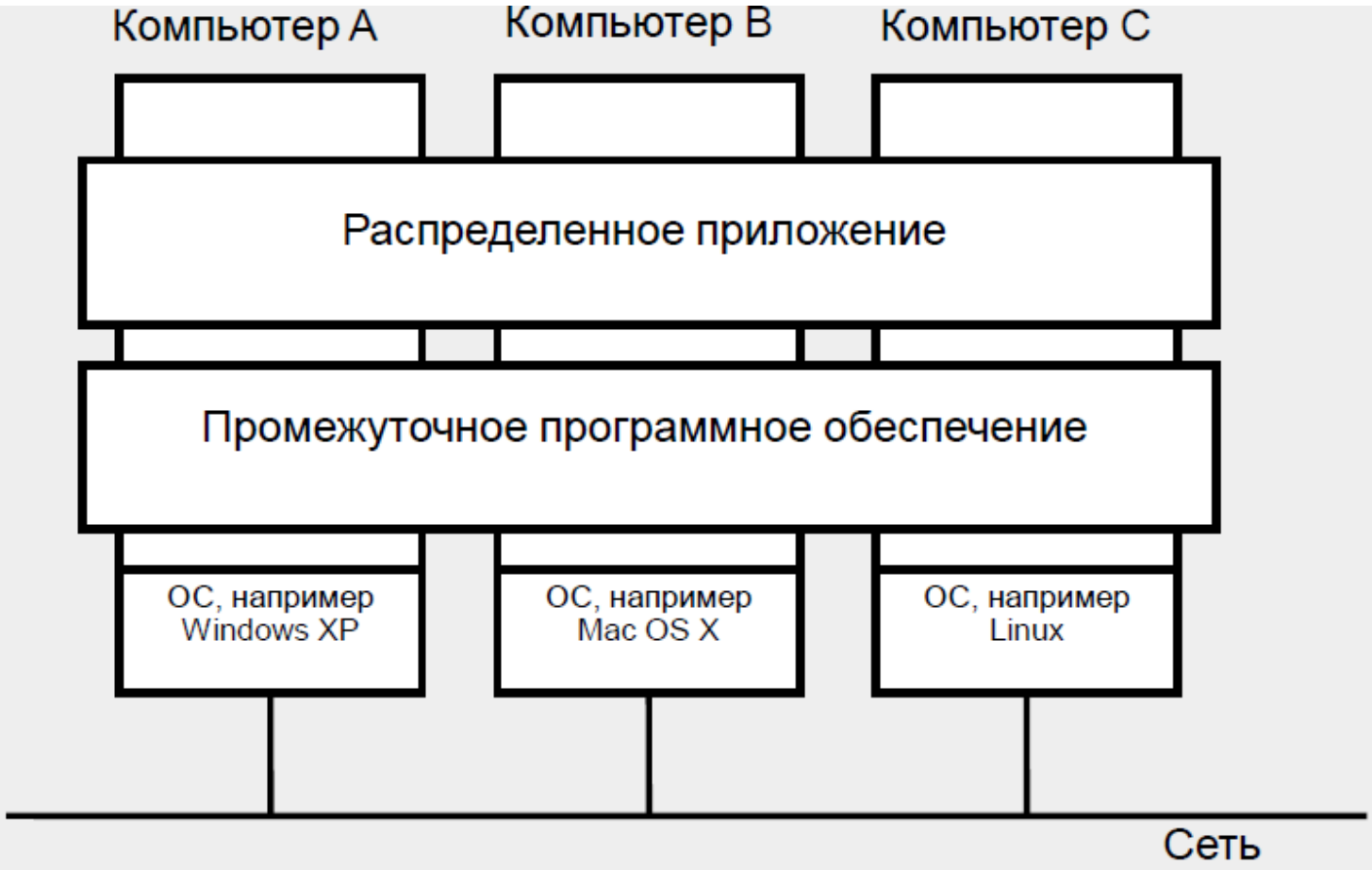


Терминология РВС

Видно, что каждый компьютер или устройство представляет собой *сущность* в распределенной вычислительной системе в виде *узла*. При этом на каждом узле может располагаться несколько *клиентов*, *серверов*, *служб* или *пиров*. Важно заметить, что любой узел, сервер, пир или сервис *(но не клиент!)* являются *ресурсами* распределенной вычислительной системы.

Сервис можно определить как некую замену вызова функции на локальном компьютере. Существует множество технологий, обеспечивающих создание и сопровождение сервисов в распределенных вычислительных системах: технология XML Веб-сервисов, веб-серверы, сервисы REST и пр.

Слои ПО в РВС



- Гетерогенная среда – обеспечение взаимодействия разных ОС
- Сокрытие гетерогенности от пользователя и приложений
- Обеспечение простоты расширения и масштабируемости
- Взаимодействие на основе обмена сообщениями

Слои ПО в РВС

Для обеспечения работы гетерогенного оборудования РВС в виде единого целого, стек ПО обычно разбивают на два слоя. На *верхнем слое* располагаются приложения, отвечающие за решение определенных прикладных задач средствами РВС. Их функциональные возможности базируются на *нижнем слое* – *промежуточном программном обеспечении (ППО).*
ППО взаимодействует с системным ПО и сетевым уровнем, для обеспечения прозрачности работы приложений в РВС.

Типы прозрачности в РВС

Выделяют следующие типы прозрачности в РВС:

- *Доступ к ресурсам* – разница в представлении данных и в способах доступа к ресурсам РВС должна быть скрыта от пользователей;
- *Местоположение ресурсов* – место физического расположения требуемого ресурса должно быть несущественно для пользователя;
- *Перенос ресурсов* – в процессе работы распределенной ВС те или иные ресурсы (информационные – базы данных, физические – сетевые принтеры и т.д.) могут менять свое местоположение; это не должно оказывать влияния на работу пользователя и его приложений;

Типы прозрачности в РВС

Выделяют следующие типы прозрачности в РВС:

- *Репликация* – сокрытие от пользователя того, что в реальности существует более одной копии используемых ресурсов;
- *Параллельный доступ* – возможность совместного (одновременного) использования одного и того же ресурса различными пользователями независимо друг от друга. При этом факт совместного использования ресурса должен оставаться скрытым от пользователя;
- *Отказы* – отказ (отключение) каких-либо ресурсов распределенной ВС не должен оказывать влияния на работу пользователя и его приложения.

Прозрачность РВС

- Прозрачный доступ к ресурсам
- Прозрачное местоположение ресурсов
- Прозрачная репликация
- Возможность параллельного доступа
- Прозрачность отказов

Классификация РВС

Можно выделить следующие признаки классификации РВС по шкале «Централизованный – Децентрализованный»:

- Методы обнаружения ресурсов;
- Доступность ресурсов;
- Методы взаимодействия ресурсов.

В рамках РВС *обнаружение ресурсов* играет важнейшую роль. Служба, отвечающая за обнаружение ресурсов сети, называется *службой обнаружения* (например, DNS, Jini Lookup, UDDI и др.). Существует ряд механизмов обнаружения ресурсов, которые сильно зависят от типа приложения и ППО.

Классификация РВС

Примером *централизованного метода обнаружения* ресурсов может служить служба DNS. Данная служба работает по принципам, чрезвычайно похожим на принцип работы телефонной книги. Вы даете ей имя сайта (например, www.susu.ac.ru) и DNS возвращает его IP-адрес (например, 85.143.41.59). Таким образом, сервер DNS представляет собой большую базу данных о тех ресурсах, которые существуют в настоящий момент в РВС. Существует ограниченное количество серверов, которые предоставляют службу DNS. Обычно пользователь указывает ограниченное количество таких серверов для работы (1 или 2). И если данные сервера отключаются, то процесс обнаружения ресурсов останавливается.

Классификация РВС

При использовании *децентрализованного метода обнаружения* ресурсов (например, в сети Gnutella) запрос на поиск отправляется всем узлам, известным отправителю. Эти узлы производят поиск ресурса у себя, и транслируют запрос далее. Таким образом, отсутствуют выделенные узлы для обнаружения и централизованное хранилище информации о ресурсах, доступных в сети.

Другим важным фактором является *доступность ресурсов* РВС. Примером *централизованной доступности ресурсов* в РВС может являться технология Веб-служб. Существует только один сервер с выделенным IP-адресом который предоставляет определенную Веб-службу или сайт. Если данный узел выйдет из строя, или будет отключен от сети, данная служба станет недоступна.

Классификация P2P

Естественно, можно применить методы репликации для расширения доступности определенного сайта или службы, но доступность определенного IP-адреса останется прежней. Существуют системы, предоставляющие *децентрализованные подходы к доступности ресурсов* посредством множественного дублирования сервисов, которые могут обеспечить функциональность, необходимую пользователю. Наиболее яркими примерами децентрализованной доступности ресурсов могут служить одноранговые вычислительные системы (BitTorrent, Gnutella, Napster), где каждый узел играет роль как клиента, так и сервера, который может предоставлять ресурсы и сервисы, аналогичные остальным устройствам данной сети (поиск, передача данных и др.)

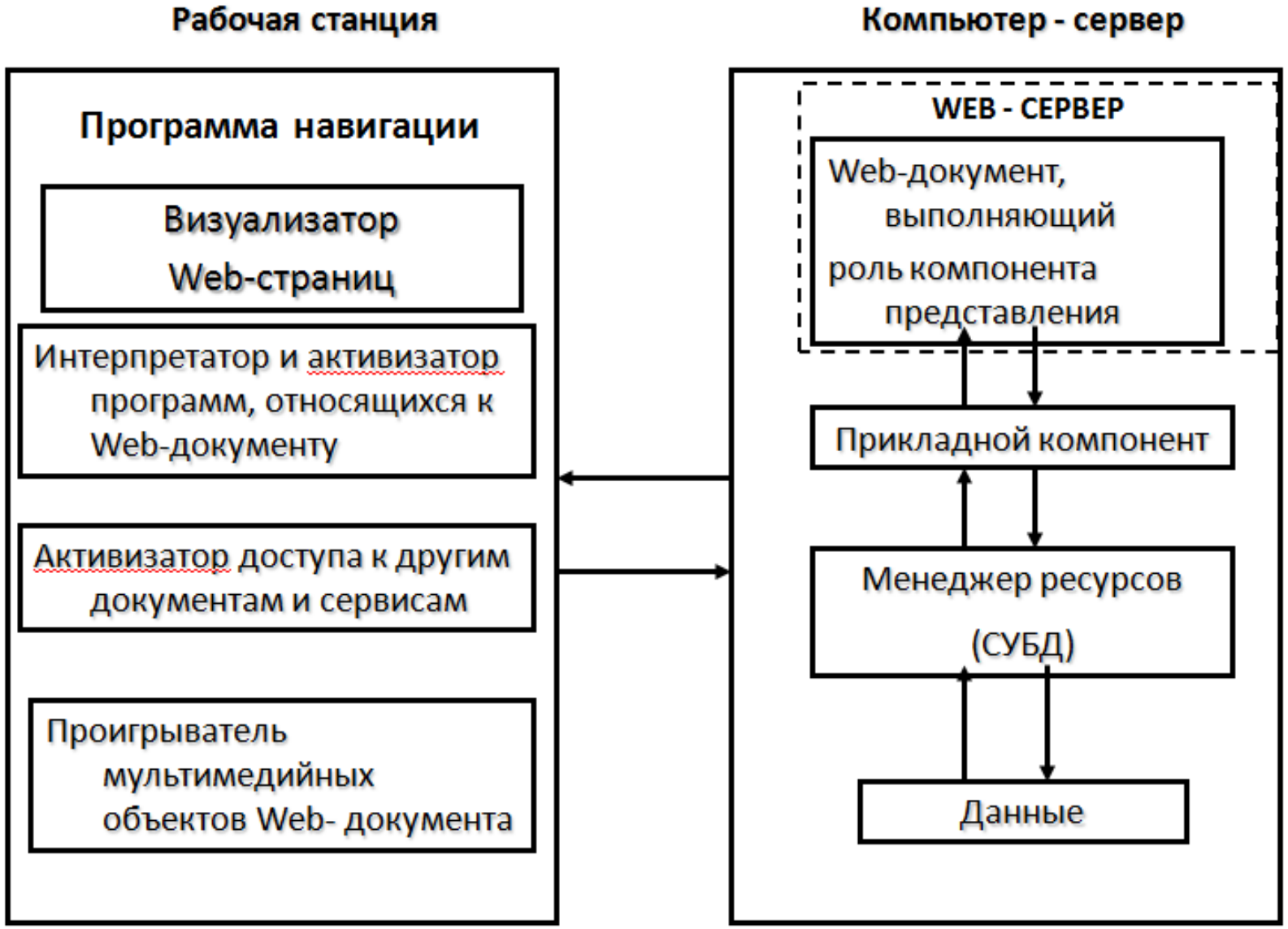
Классификация РВС

Еще одним критерием классификации РВС могут служить *методы взаимодействия узлов.*

Централизованный подход к взаимодействию узлов основан на том, что взаимодействие между узлами всегда происходит через специальный центральный сервер. Таким образом, один узел не может обратиться к другому непосредственно.

Децентрализованный подход к взаимодействию реализуется в одноранговых вычислительных системах. Такой подход основывается на прямом взаимодействии между узлами РВС, т.к. каждый узел играет как роль клиента, так и роль сервера.

Архитектура «клиент - сервер» на основе Web – технологии



Организация связи в РВС. Протокол

Взаимодействие базируется на протоколах.

Протокол - это набор правил и соглашений, описывающий процедуру взаимодействия между компонентами системы.

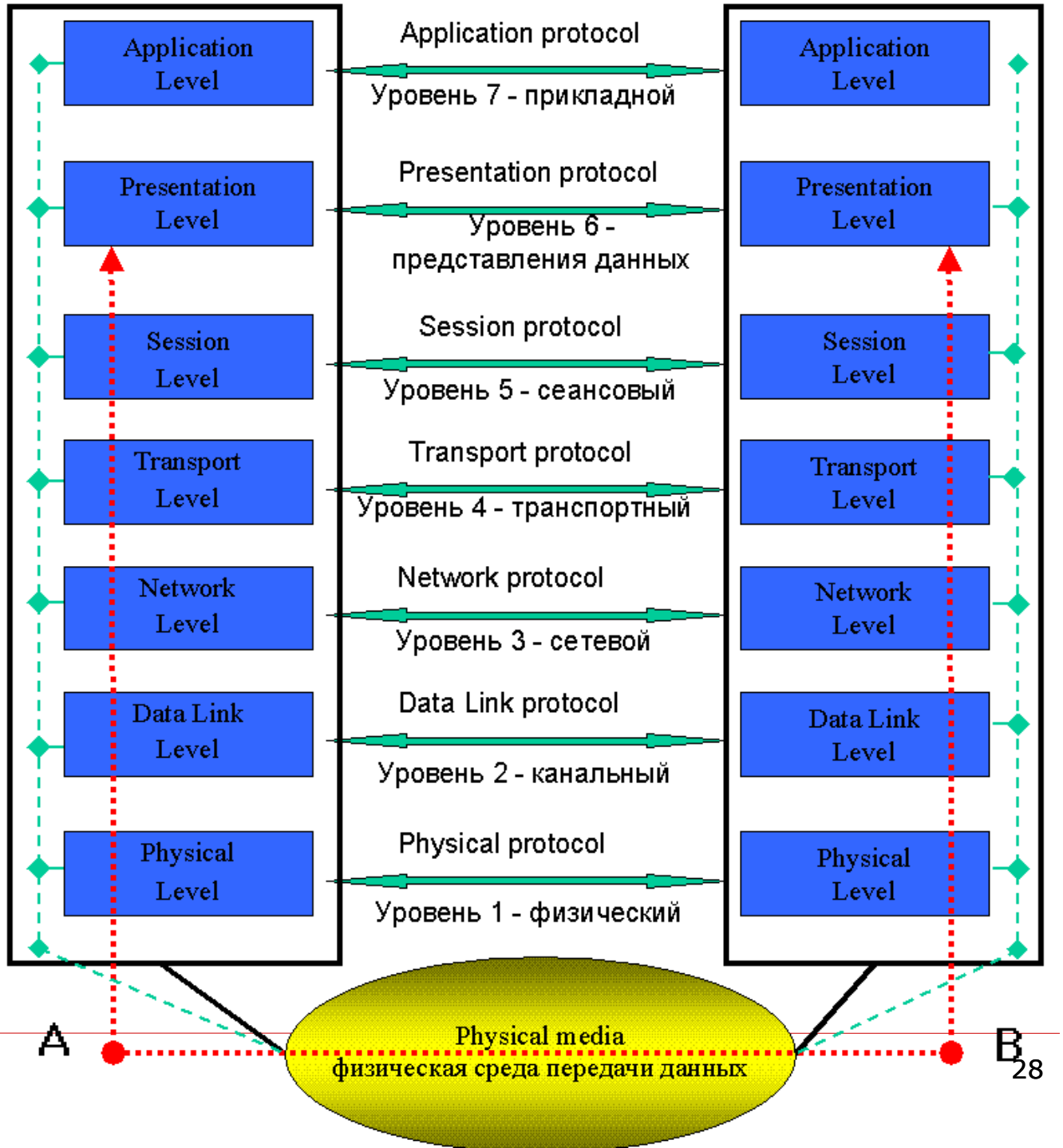
Работу сети обеспечивает множество различных *протоколов*:

протоколы управления физической связью, протоколы установления связи по сети,

протоколы доступа к различным *ресурсам* и т.д.

В сети Интернет принята семиуровневая структура организации сетевого взаимодействия, с целью упорядочить множество протоколов и отношений.

Модель OSI


Application Level
Application protocol
Уровень 7 - прикладной
Presentation Level
Presentation protocol
Уровень 6 - представления данных
Session Level
Session protocol
Уровень 5 - сеансовый
Transport Level
Transport protocol
Уровень 4 - транспортный
Network Level
Network protocol
Уровень 3 - сетевой
Data Link Level
Data Link protocol
Уровень 2 - канальный
Physical Level
Physical protocol
Уровень 1 - физический
A
Physical media
физическая среда передачи данных
B

Стек протоколов OSI

Организация обмена сообщениями.

- **Прямая передача сообщений**
  - возможна только если принимающая сторона готова к приему сообщения в этот момент времени
- **Использование менеджера сообщений**
  - компонента высылает сообщение в очередь менеджера, из которой, в дальнейшем, принимающая сторона извлекает полученное сообщение

Прямая передача сообщений: сокеты.

- Используется на основе интерфейса сокетов
- Т.е. используется непосредственно транспортный уровень в виде Middleware.

- **Сокет –** абстрактный объект, представляющий конечную точку соединения.
- **Сокет TCP/IP –** комбинация IP-адреса и номера порта, например 10.10.10.10:80.
- Интерфейс сокетов впервые появился в BSD Unix.

Гетерогенная Grid-система на базе BOINC

Одной из наиболее популярных форм организации вычислительных ресурсов для проведения большого числа расчетов является *Grid*.

Grid-вычисления - это форма организации распределенных вычислений. В ней участвуют территориально распределенные разнородные компьютеры, объединенные с помощью сети передачи данных и работающие совместно для решения задач, требующих значительных вычислительных ресурсов.

С точки зрения сетевой организации *grid* представляет собой согласованную, открытую и стандартизованную среду, которая обеспечивает гибкое, безопасное, скоординированное разделение вычислительных ресурсов и ресурсов хранения данных.

Гетерогенная Grid-система на базе BOINC

Наиболее эффективно использование *grid* при выполнении следующих задач:

- анализ и обработка независимых наборов данных;
- решение задач, обладающих хорошей степенью параллелизации по данным.

Существует большое количество систем промежуточного программного обеспечения (СППО), предназначенных для организации, сопровождения и управления *grid* .

По назначению СППО можно условно разделить на две группы.

Гетерогенная Grid-система на базе BOINC

Первая группа содержит системы, предназначенные для объединения относительно небольшого числа высокопроизводительных вычислителей (кластеров). К таким системам относятся, например, Condor , Globus Toolkit, Unicore и др.

Вторая группа СППО содержит системы, цель которых заключается в объединении в *grid* большого числа (до сотен тысяч) вычислителей, каждый из которых обладает относительно невысокой производительностью.

Гетерогенная Grid-система на базе BOINC

Такие *grid* -сети также называют системами распределенных вычислений (*distributed computing*) и системами добровольных вычислений (volunteer computing). Их специфика заключается, в высокой вероятности недоступности отдельных вычислительных узлов.

Наиболее популярной СППО этой группы является платформа *BOINC* (Berkeley Open Infrastructure for Network Computing)

Другой пример □ разработка НИВЦ МГУ им. М.В. Ломоносова система *X-Com*.

Гетерогенная Grid-система на базе BOINC

Платформа организации распределенных вычислений *BOINC* - активно развивающаяся СППО с открытым исходным кодом. Она стала основой большого числа независимых проектов добровольных вычислений (наиболее популярные из них - Climateprediction.net , SETI@home и Einstein@home). Платформа *BOINC* отличается простотой в установке, настройке и администрировании. Она обладает хорошими возможностями по масштабируемости, простоте подключения новых вычислительных узлов, использованию дополнительного ПО, интеграции с другими *grid* –системами.

Гетерогенная Grid-система на базе BOINC

Платформа BOINC имеет архитектуру "*клиент-сервер*. *Клиентская часть* может работать на компьютерах с различными аппаратными и программными характеристиками. Ключевым объектом системы является проект - автономная сущность, в рамках которой производятся распределенные вычисления.

BOINC-сервер поддерживает одновременную работу большого числа независимых проектов. Каждый вычислительный узел может одновременно производить вычисления для нескольких BOINC- проектов.

Проект однозначно идентифицируется своим URL-адресом (*URL - Universal Resource Locator*).

(URI - Uniform Resource Identifier)

BOINC предоставляет возможность гибкой настройки клиентской части, регулируя максимальный размер загружаемых файлов, время выполнения рабочих заданий, загрузку CPU или GPU, выделяемый объем оперативной памяти и дискового пространства.

Архитектура системы на базе BOINC

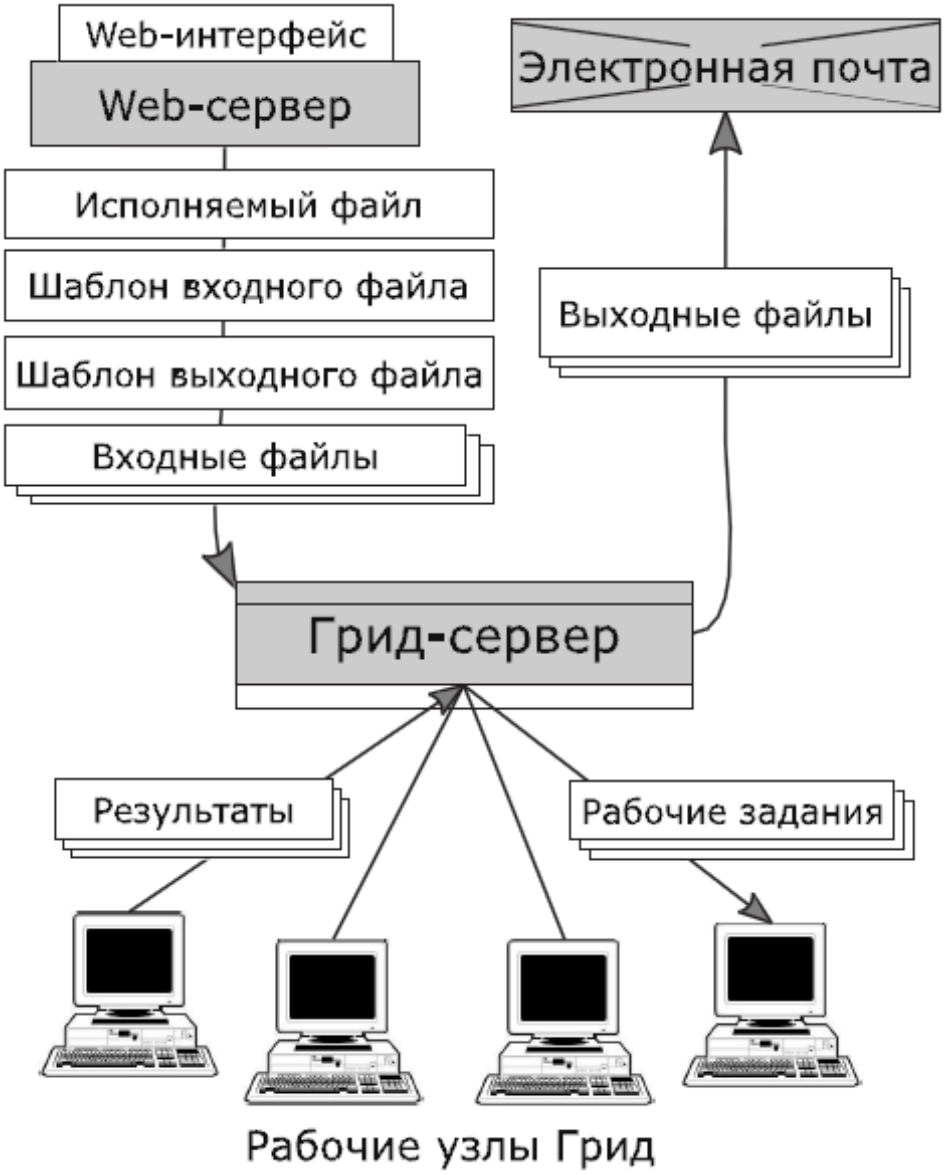
Рабочий процесс в *grid* -системе, основанной на платформе BOINC, организован следующим образом.
Вычислительные узлы, имеющие свободные ресурсы, обращаются к серверу для получения новых рабочих заданий.
Сервер BOINC рассылает клиентским приложениям экземпляры рабочих заданий, клиенты выполняют вычисления и отсылают обратно результаты.
После получения результатов сервер проверяет и обрабатывает их, например занося в базу данных или автоматически создавая на их основе новые рабочие задания.

Архитектура системы на базе BOINC

Платформа BOINC позволяет выполнять в *grid* -системе специализированные приложения, в которых, в частности, может быть реализован механизм сохранения промежуточных результатов вычислений, графическое отображение процесса выполнения приложения, автоматическое распараллеливание вычислительного процесса в зависимости от количества доступных на клиенте вычислительных ядер и т.п.

Но унаследованные (неадаптированные для работы в *grid*) приложения с помощью специальных *приложений-"оберток"* также могут быть перенесены на платформу BOINC без необходимости изменения их исходного кода и перекомпиляции. Приложения-"обертки" берут на себя взаимодействие с ядром программы-клиента, запуская исходное приложение как свой дочерний процесс.

Архитектура системы на базе BOINC



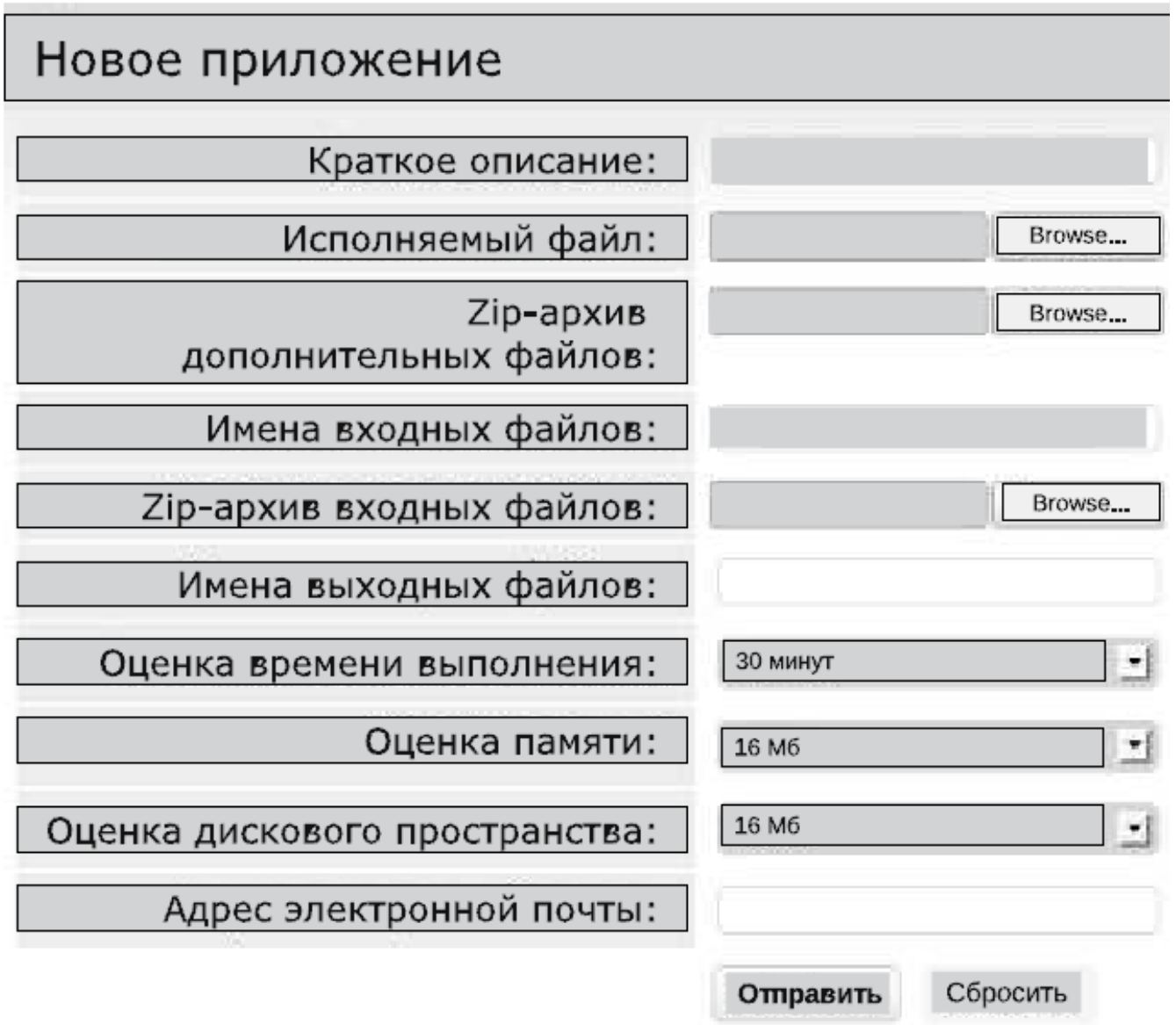
Web –интерфейс доступа к grid-сегменту

Для обеспечения удобного доступа пользователей к вычислительным ресурсам *grid* был разработан специальный web-интерфейс.

HTML-форма содержит:

- поля для загрузки на web-сервер исполняемого файла приложения, дополнительных файлов приложения (при необходимости) и zip-архива входных файлов;
- поля для ввода имен входных и выходных файлов, которые требует исполняемый файл приложения;
- поля для ввода оценки ресурсов, требуемых для выполнения заданий;
- поле для ввода адреса электронной почты, на который требуется отправить результат.

Web –интерфейс доступа к grid-сегменту



Web –интерфейс доступа к grid-сегменту

Обязательными для заполнения являются поле для загрузки исполняемого файла приложения и поле для ввода адреса электронной почты.
Пользователь загружает исполняемый и дополнительные файлы приложения, указывает имена входных и выходных файлов и загружает zip-архив входных файлов.
Каждый из них послужит основой для создания отдельного рабочего задания.
Данные, введенные пользователем, обрабатываются на стороне Web-сервера и передаются BOINC-серверу.

Web –интерфейс доступа к grid-сегменту

Программа обработки введенных данных на стороне Web-сервера создает новое BOINC-приложение и шаблоны входных и выходных файлов.

В данном случае каждое приложение включает в себя программу-"обертку" в дополнение к исполняемому файлу приложения, загруженному пользователем.

Рабочие задания BOINC генерируются на основе загруженных входных файлов и созданных шаблонов.

Web –интерфейс доступа к grid-сегменту

Рабочие узлы *grid* периодически отсылают на *grid* -сервер запросы новых заданий. Программа-планировщик, входящая в состав BOINC, отвечает на запросы, распределяя неразосланные рабочие задания между узлами, которые запросили задания и которые соответствуют определенным критериям.

Например имеют достаточный объем оперативной памяти и свободного дискового пространства для выполнения заданий.

Ход выполнения рабочих заданий отображается на Web-сайте. Пользователю доступна информация об общем количестве рабочих заданий на сервере и их статусе (например, "в процессе выполнения" или "завершено").

Web –интерфейс доступа к grid-сегменту

По мере того как завершенные результаты поступают с рабочих узлов *grid* на сервер, программа-валидатор, входящая в состав BOINC, проверяет выходные файлы и отмечает результат как правильный или ошибочный. Правильные результаты обрабатываются программой-ассимилятором, которая упаковывает их в zip-архив и отсылает пользователю по электронной почте после того, как все результаты рабочих заданий успешно обработаны.

Применение grid-сегмента в квантово-химических расчетах

Для этих расчетов использовался наиболее популярный программный пакет Firefly, частично основанный на исходном коде системы GAMESS (US). Firefly предназначен для выполнения расчетов ab initio и DFT (основе теории функционала плотности) на Intel-совместимых процессорах x86, AMD64 и EM64T.

Вычислительные методы ab initio, или неэмпирические, предназначены для расчета с максимально возможной точностью физических и химических свойств заданного химического соединения (как многоатомной системы) на основе представлений и методов квантовой механики.

Применение grid-сегмента в квантово-химических расчетах

Такие расчеты позволяют оценить вклад энергии, требующейся для перестройки электронных оболочек атомов, изменения геометрической структуры комплекса и сближения катионов между собой, в общую энергию образования частицы, состоящей из комплекса и катионов.
Исходные данные для проведения вычислительных экспериментов представляют собой наборы параметров, регулирующих работу программы, в том числе координаты атомов, метод расчета и базис волновых функций. Выходные файлы должны содержать информацию о полной энергии рассчитываемой системы, ее энергетических уровнях, параметрах молекулярных орбиталей и т.п.

Применение grid-сегмента в квантово-химических расчетах

Такие расчеты позволяют оценить вклад энергии, требующейся для перестройки электронных оболочек атомов, изменения геометрической структуры комплекса и сближения катионов между собой, в общую энергию образования частицы, состоящей из комплекса и катионов.
Исходные данные для проведения вычислительных экспериментов представляют собой наборы параметров, регулирующих работу программы, в том числе координаты атомов, метод расчета и базис волновых функций. Выходные файлы должны содержать информацию о полной энергии рассчитываемой системы, ее энергетических уровнях, параметрах молекулярных орбиталей и т.п.

Применение grid-сегмента в квантово-химических расчетах

При проведении вычислительных экспериментов в состав *grid*-сегмента входили 14 вычислительных узлов с различными аппаратными и программными характеристиками, и разными настройками, связанными с организацией вычислений.

Два узла обслуживали проекты BOINC в монопольном режиме, а на десяти вычислительных узлах кластера расчеты в рамках *grid* регулярно приостанавливались для выполнения расчетов пользовательских задач, запускаемых на кластере с помощью системы управления заданиями.

Суммарная пиковая производительность *grid* составила 1.04 TFLOPS.

Система X-Com. http://x-com.parallel.ru/

Система метакомпьютинга *X-Com* предназначена для быстрого развертывания и проведения распределенных расчетов. Система представляет собой инструментарий для адаптации и поддержки выполнения программ в распределенных неоднородных средах. Система поддерживает среды с десятками тысяч вычислительных узлов (процессоров), обеспечивает корректную работу в условиях высокой динамичности состава среды, не требует административного доступа к ресурсам. Система X-Com написана на языке Perl. Это обеспечивает ее работоспособность на подавляющем большинстве современных программно-аппаратных платформ.

Система X-Com. http://x-com.parallel.ru/

Система метакомпьютинга *X-Com* предназначена для быстрого развертывания и проведения распределенных расчетов. Система представляет собой инструментарий для адаптации и поддержки выполнения программ в распределенных неоднородных средах. Система поддерживает среды с десятками тысяч вычислительных узлов (процессоров), обеспечивает корректную работу в условиях высокой динамичности состава среды, не требует административного доступа к ресурсам. Система X-Com написана на языке Perl. Это обеспечивает ее работоспособность на подавляющем большинстве современных программно-аппаратных платформ.

Система X-Com.

Ее архитектура основана на технологии клиент-сервер. Поэтому прикладная задача должна быть логически разделена на две части: *серверную* и *клиентскую*.

Серверная часть отвечает за разбиение задачи на множество независимых порций и объединение результатов. Для реализации серверной части задачи в системе *X-Com* имеется два прикладных интерфейса *API* – (Application Programming Interface) - интерфейс создания приложений.

В *простейшем случае*, если необходимо выполнить одну и ту же программу на множестве различных входных файлов, используется *API Files*, и тогда серверная часть задачи описывается в параметрах настройки сервера *X-Com* (указываются пути к входным и выходным каталогам и список файлов для обработки).

В менее тривиальных случаях применяется *API Perl*. Этот интерфейс предполагает написание модуля на языке Perl, реализующего заданный набор функций:

- ✓инициализацию серверной части задачи,
- ✓генерацию номера первой и последней порции,
- ✓генерацию тела порции по нее номеру,
- ✓обработку результатов готовой порции,
- ✓условия завершения работы серверной части задачи,
- ✓а также действия, выполняемые перед завершением.

Система X-Com.

Аналогичный подход применяется и для реализации *клиентской* части задачи.

В элементарном случае в параметрах настройки сервера *X-Com* указывается формат команды с именами входных и выходных файлов, которая будет запущена клиентом X-Com на вычислительном узле.

В более сложном случае клиентская часть задачи может быть описана двумя функциями на языке Perl:

- ✓инициализация задачи на узле и
- ✓обработка каждой порции данных.

После реализации клиентской и серверной части прикладной задачи формируется непосредственно вычислительная среда.

Система X-Com.

Этот процесс состоит из двух частей:

- ✓запуск клиентов X-Com на вычислительных узлах и
- ✓настройка и запуск сервера X-Com.

Клиент X-Com может быть запущен на вычислительном узле постоянно (в монопольном режиме) или запускаться в те моменты времени, когда узел не занят другими процессами (работа по занятости).

На высокопроизводительных вычислительных комплексах используются интерфейсы к штатным системам очередей.

В текущей версии X-Com поддерживается взаимодействие с *Cleo, Torque, LoadLeveler, Slurm*, а также *Unicore*.

Система X-Com.

Сервер X-Com запускается на специальной выделенной машине вручную либо с помощью подсистемы управления заданиями. Такой вариант предоставляет пользователям более удобный способ работы с распределенной средой, позволяя им оперировать привычными понятиями очереди заданий. При этом обеспечивается и последовательное выполнение задач на всех доступных ресурсах, и параллельное выполнение одновременно нескольких заданий.
Возможен запуск заданий с учетом их требований к ресурсам, на которые они будут распределены.
Существенная особенность системы *X-Com* – возможность построения иерархических распределенных сред с произвольным количеством уровней.

Система X-Com.

Данная функциональность реализована с помощью промежуточных серверов X-Com.
Промежуточный сервер получает задания и необходимые данные от вышестоящего сервера X-Com (для этого сервера он представляется клиентом X-Com) и распределяет их внутри пула своих клиентов (для них он представляется единственным сервером X-Com).
Введение промежуточных серверов позволяет снизить нагрузку на центральный сервер распределенной среды, оптимизировать потоки данных и подключить к расчетам вычислительные ресурсы, находящиеся внутри закрытых сетей.

Система X-Com.

С помощью системы *X-Com* был решен целый ряд вычислительно емких задач из различных научных областей.
В каждом расчете отрабатывалась та или иная функциональность системы.
Система X-Com может применяться как основа для построения сервисов по распределению заданий на доступные вычислительные ресурсы. Совместно с компаниями Тесис и Сигма Технология в 2009-2010 гг. был реализован программный комплекс для решения оптимизационных гидродинамических задач. Программный комплекс объединял оптимизатор *IOSO*, решатель *FlowVision* и систему *X-Com*. Последняя отвечала за взаимодействие между оптимизатором, установленном на рабочем месте пользователя комплекса и генерирующим пакеты заданий для решателя, и системой очередей, через которую осуществлялся запуск пакета *FlowVision* на суперкомпьютерах МГУ.

Система X-Com.

На суперкомпьютере *СКИФ МГУ "Чебышев"* с помощью системы *X-Com* был реализован сервис выполнения пакетов однопроцессорных задач на суперкомпьютерах. Основная задача сервиса – дополнение системы управления прохождением задач - Cleo функциональностью по группировке нескольких однопроцессорных задач на один узел суперкомпьютера. Это повышало эффективность использования ресурсов вычислительной системы. Система X-Com применялась для исследования свойств прикладных задач на процессорном полигоне НИВЦ МГУ.

Система X-Com.

В рамках этой работы проводилось сравнение времени выполнения приложения, откомпилированного с использованием различных компиляторов, запускаемого по одному или более процессов на узел, на имеющихся в полигоне программно-аппаратных платформах.
Таким образом оценивалась эффективность работы приложения в зависимости от набора факторов (архитектура узла, компилятор и его опции, наличие или отсутствие параллельно работающих процессов).

Лекция окончена!

Благодарю за внимание!