Лекции по дисциплине
«Высокопроизводительные технологии информационных инфраструктур»
ГУТ, ИСИТ/ИУС- 2016

знать:

методы анализа высокопроизводительных технологий;
модели обработки информации с применением высокопроизводительных технологий;
состав, структуру, принципы реализации и функционирования высокопроизводительных технологий, используемых при создании информационных инфраструктур

Содержание дисциплины

1. **Представление высокопроизводительных вычислительных машин и систем (ВВМиС)** в классе ресурсов информационных инфраструктур.

   Технологии организации ВВМиС. Технологии доступа к ВВМиС. Технологии обработки информации в ВВМиС. Технологии развёртывания, администрирования и управления ресурсами ВВМиС. Технологии информационной безопасности ВВМиС.

2. **Технологии интеграции** ВВМиС в информационные инфраструктуры.

   Сетевые технологии интеграции. Безопасные и масштабируемые технологии кластеризации. Технологии виртуализации.

3. **Отображение** информационных и коммуникационных технологий на архитектуру высокопроизводительных вычислительных машин и систем.

   Отображение информационных интеллектуальных технологий. Отображение мультимедийных технологий. Отображение технологий дизайна. Отображение CAD-, CAE-технологий. Отображение технологий управления взаимоотношениями с клиентами. Отображение технологий мониторинга и ситуационного управления. Отображение технологий извлечения знаний. Отображение технологий генерации знаний. Отображение технологий управления знаниями. Базовые компоненты реализации технологий отображений.

4. **Реализации информационных инфраструктур** с высокопроизводительными вычислительными машинами и системами.

Сферы профессиональной деятельности в среде информационных инфраструктур с ВВМиС. Функциональная спецификация сред информационных инфраструктур с ВВМиС. Обзор и примеры информационных инфраструктур с ВВМиС. Технологические процессы разработки средств отображения информационных и коммуникационных технологий на архитектуру высокопроизводительных вычислительных машин и систем.

Введение

В России активно идут процессы развития национальной информационной инфраструктуры, в первую очередь, в научно-технической сфере. Обеспечить постоянное пополнение информационных ресурсов, создать на базе современных технологий национальную информационную систему в целях эффективного информационного обеспечения науки, техники, образования, укрепить государственный и предпринимательский секторы в информационной сфере, обеспечить для граждан страны условия свободного доступа к информационным ресурсам – вот основные цели государственной информационной политики как неотъемлемой составляющей государственной научно-технической политики России.

В настоящее время есть все основания утверждать, что в новое столетие инженерный мир входит с практически сформировавшейся традицией обязательного модельного обеспечения всех сколь-нибудь сложных технических разработок и научных исследований. Причем, на современном этапе речь должна идти уже не просто о тех или иных отдельных компьютерных моделях, а о различных вариантах интегрированных моделирующих сред.

Существует много фундаментальных и прикладных задач, для эффективного решения которых требуется привлечение технологий высокопроизводительных вычислений — все эти задачи объединены в группу «Grand Challenges»:

- предсказания погоды и климата;
- исследование материалов;
- построение полупроводниковых приборов;
- разработка фармацевтических препаратов;
- генетика;

- астрономия;
- транспортные задачи;
- гидро- и газодинамика;
- управляемый термоядерный синтез;
- эффективность систем сгорания топлива;
- геоинформационные системы;
- разведка недр;
- наука о мировом океане;
- распознавание и синтез речи;
- распознавание изображений.

Эффективный анализ данных требует наличия высокопроизводительных средств и методов для их обработки. Задачи **информационной инфраструктуры** науки и техники не могут быть реализованы без опоры на перспективные информационные технологии. В этой области создаются принципиально новые для России современные средства вычислительной техники с базовым программным обеспечением на новых архитектурных и физических принципах, разрабатывается элементная база и технологии производства сверхскоростных интегральных и оптоэлектронных схем для супер-ЭВМ новых поколений.

В связи с этим были разработаны способы повышения производительности вычислений, как на программном, так и на аппаратном уровне. Например, такие как **директивы многопоточного программирования (множественные потоки выполнения), использование более производительных процессоров, использование графических адаптеров для обеспечения повышенной производительности расчетов сложных математических задач. А также использование распределенных кластерных систем.**

В России благодаря Федеральной целевой программе «Интеграция» в ближайшее время ожидается существенное увеличение числа высокопроизводительных кластеров в российских образовательных и научно-исследовательских организациях, т.к. инициаторы программы справедливо считают, что использование высокопроизводительных вычислительных ресурсов жизненно необходимо для получения качественно новых результатов и поддержания выполняемых фундаментальных научных исследований на мировом уровне.

В целях оптимального развития высокопроизводительных вычислительных ресурсов отечественных суперкомпьютерных центров, оказания действенной

государственной поддержки в этой сфере Миннауки России, Российской академией науки, Российским фондом фундаментальных исследований, Минобразования России подготовлена межведомственная программа по высокопроизводительным вычислениям и их применению, реализация которой начнется в ближайшее время.

Глава 1. Функционально-структурные схемы ИС

1.1. Информационная система. Информационная сеть

Назовем ***реальной системой*** совокупность электронных устройств, соответствующего программного обеспечения, периферийного оборудования, терминалов, персонала операторов, физических процессов и средств передачи информации, являющуюся полностью автономной и осуществляющую обработку и передачу информации. В рассматриваемой совокупности обязательным компонентом является электронное устройство - комплекс взаимосвязанных электронных машин, одна электронная машина либо беспроцессорное логическое устройство. Остальные компоненты входят в состав реальной системы по мере необходимости. В дальнейшем будет рассматриваться не сама реальная система, а ее модель, именуемая ***системой***. Эта модель отображает те характеристики реальной системы, которые необходимы для выполнения ею в сети своих основных функций. Любая система создается для выполнения ею прикладных процессов. Последние подразделяются на прикладные процессы пользователей и специальные прикладные процессы (рисунок 1.1). *Пользователями* здесь и далее называются лица, для которых системы выполняют обработку информации. ***Прикладные процессы, пользователей*** **делятся на две группы. К первой** из них относятся ***программные*** *прикладные процессы***.** Каждый такой процесс определяется одной прикладной программой либо группой взаимосвязанных прикладных программ. ***Специальные прикладные процессы,*** являются вспомогательными. Они необходимы лишь для оказания помощи в выполнении прикладных процессов пользователей. Специальными являются процессы управления системой, диагностики работы системы, обеспечения безопасности информации и т. д.

В зависимости от выполняемых задач разделим системы на три класса: абонентские, административные и ассоциативные. Системы, основной задачей которых является выполнение прикладных процессов для нужд пользователей, назовем *абонентскими* (рисунок 1.1). Эти системы являются главными. Административные и ассоциативные системы предназначены для обеспечения взаимодействия абонентских систем.

Административные системы управляют процессами этого взаимодействия, а *ассоциативные системы* совместно с каналами передачи данных обеспечивают прокладку трактов, соединяющих прикладные процессы абонентских систем.

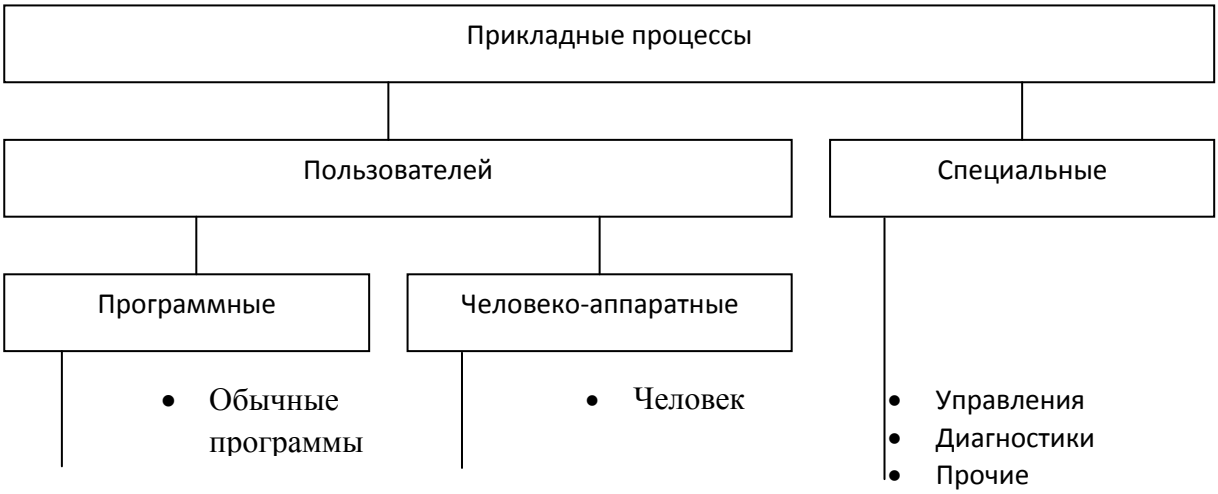


Рис. 1.1. Классификация прикладных процессов

Как следует из рис. 1.2, *абонентская система* состоит из двух основных частей: *прикладных процессов*, для выполнения которых эта система создается, и *области взаимодействия.* Назначением последней является обеспечение связи прикладных процессов друг с другом, передачи информации во внешнюю для сис*темы* физическую среду, прием системой информации из этой среды через *коммутационное ядро или транспортную систему области взаимодействия*.

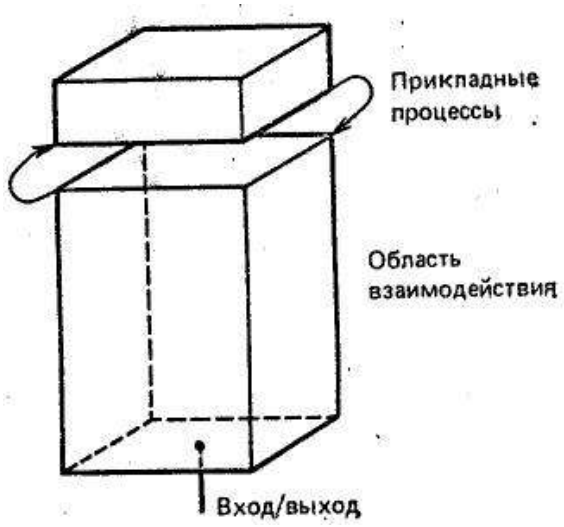


Рис.1.2. Абонентская система

Взаимодействие осуществляется через вход-выход системы. Ассоциация абонентских систем образует *информационную сеть. Информационная сеть* включает в себя как подсеть взаимодействия, так и все прикладные процессы.

Информационная сеть является базой обработки всевозможной информации, связанной с жизнью современного общества.

1.2. Компоненты информационных сетей

Разнообразие и сложность информационных сетей приводят к необходимости разработки различных типов компонентов, их образующих. Это связано, прежде всего, с необходимостью создания не только отдельных территориальных *v* локальных сетей, но и их ассоциаций. Такие *ассоциации информационных сетей* удовлетворяют самым жестким требованиям пользователей, касающимся разнообразия информационных ресурсов, надежности и скорости передачи данных.

Ассоциация состоит (рис. 1.3) из группы информационных сетей, связанных друг с другом одной либо несколькими ассоциативными системами. Каждая из этих систем соединяет два (рис. 1.4) либо более комплексов физических средств соединения.

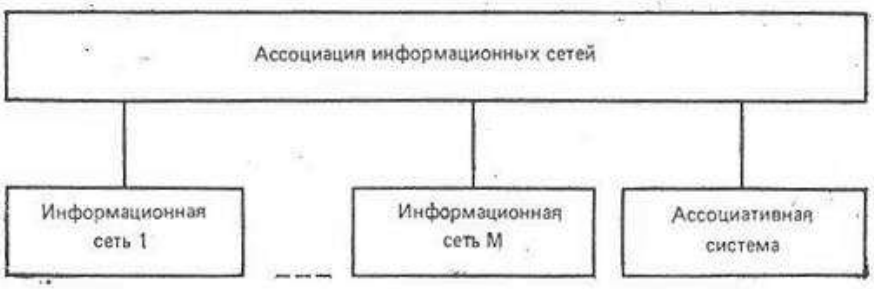


Рис. 1.3. Компоненты ассоциации информационных сетей

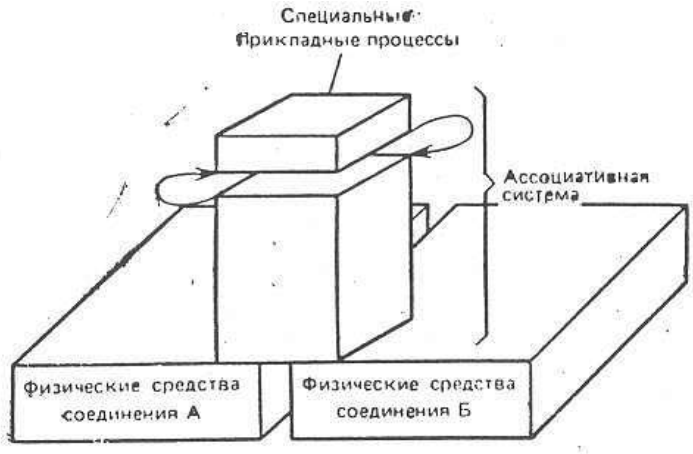


Рис. 1.4. Ассоциативная система

Специальные прикладные процессы, выполняемые ассоциативными системами, обеспечивают те виды преобразования информации, которые нужны при передаче данных из одной сети в другую.

По площади, «покрываемой» каждой из информационных сетей, их можно разделить на два класса: территориальные и локальные. *Территориальными* называют сети, охватывающие регионы, страны и континенты. *Локальными* именуют сети, включающие системы, расположенные на небольшом пространстве, ориентировочно в круге диаметром 50 км.

1.2. Базовая эталонная модель

Теоретическую основу современных информационных сетей определяет *Базовая эталонная модель Международной организации стандартов* (ISO). Она описана стандартом ISO 7498 [5], Рассматриваемая модель является основой методологии организации взаимодействия прикладных процессов, расположенных в различных абонентских системах сети. Поэтому Базовая эталонная модель ISO является концептуальной и функциональной основой разработки стандартов, которые позволяют объединять в сети системы, создаваемые на машинах различных типов и изготовленные разными производителями.

Систему, удовлетворяющую требованиям стандартов ISO, именуют *открытой.* Соответственно информационную сеть, в которой установлены открытые системы, также будем называть открытой. Далее рассматриваются только открытые информационные сети.

Важнейшей проблемой, решаемой на основе Базовой эталонной модели, является *взаимодействие открытых систем.* Под последним понимается абстрактное описание совместного функционирования прикладных процессов, расположенных в различных удаленных друг от друга открытых системах сети.

Сложность области взаимодействия открытых систем привело к тому, что в соответствии со Стандартом 7489 она разделена на семь расположенных друг над другом уровней. Это рассечение проходит через все абонентские системы, так, как показано на рис. 1.5.

Таким образом, каждый из семи *уровней* является слоем иерархического логического описания области взаимодействия открытых систем. Любой уровень состоит из активных элементов, именуемых *объектами.*

Рис.1.5.- Область взаимодействия открытых систем

Объекты одного и того же уровня связываются друг с другом *соединениями.* Последние создаются расположенными ниже уровнями области взаимодействия и коммуникационной подсетью. Пример соединения объектов уровня *i* показан на рис. 1.6. Это соединение проходит через все уровни обеих систем, расположенные под уровнем *i,* и коммуникационную сеть.

Каждый из семи уровней обеспечивает сервис для уровня, расположенного над ним. Благодаря этому верхний, прикладной уровень предоставляет прикладным процессам весь сервис, обеспечиваемый семью уровнями, т. е. всей областью взаимодействия. Нижний уровень опирается на *физические средства, соединения,* являющиеся фундаментом коммуникационной подсети.

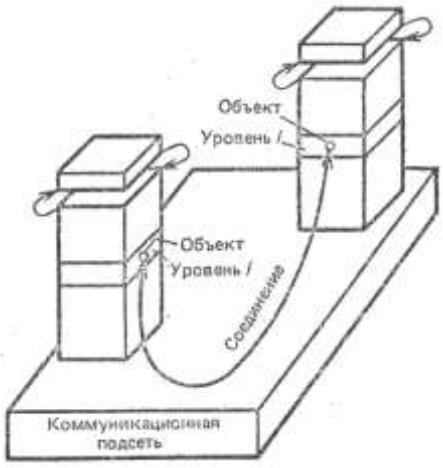


Рис.1.6.- Схема взаимодействия объектов

Основой базовой эталонной модели являются ее четыре базовых элемента:

- открытые системы;
- объекты уровня;

- соединения, связывающие объекты и позволяющие им обмениваться информацией;
- физические средства соединения.

Базовая эталонная модель определяет архитектуру сетевых международных стандартов.

1.3. Область взаимодействия открытых систем

Международная организация стандартов ISO определяет структуру и функции *области взаимодействия открытых систем*. На прикладные процессы и коммуникационную подсеть стандарты ISO не распространяются. Однако следует иметь в виду, что прикладной уровень находится рядом с прикладными процессами. Поэтому он как бы «видит» эти процессы и должен обеспечивать для них необходимый сервис. Аналогично физический уровень может взаимодействовать лишь с определенным видом коммуникационной подсети. Поэтому он также «видит» эту подсеть и предъявляет к ней определенные требования.

В базовой эталонной модели область взаимодействия систем разбивается на уровни. В соответствии с этим *открытая система* рассматривается как совокупность упорядоченного набора расположенных друг над другом логических *подсистем.* Подсистема одного и того же уровня образуют *подуровень* информационной сети. Смежные подсистемы взаимосвязаны друг с другом через их общий интерфейс.

Сущность уровневой организации функций сводится к тому, что каждый уровень расширяет сервис, обеспечиваемый нижерасположенными уровнями. Благодаря этому наивысший уровень обеспечивает прикладные процессы полным набором всех видов сервиса, необходимых этим процессам для взаимодействия друг с другом. Прикладной процесс благодаря существованию функций семи уровней получает разнообразные виды сервиса. Прикладные процессы используют *примитивы*—блоки данных, обеспечивающие их взаимодействие (например, примитивы «Подключить к системе Е», «Передать эти данные прикладному процессу М»).

Наряду с этим, уровневая организация обеспечивает относительную независимость подсистем. Это позволяет совершенствовать либо заменять подсистемы одного уровня, не затрагивая функционирование остальных уровней. Более того, уровневая структура дает возможность одному и тому же уровню работать с несколькими (на выбор) различными нижерасположенными уровнями. Иерархия функций уровней, образующих область взаимодействия открытых систем, показана в табл. 1.2.

Документы ISO для каждого уровня определяют:

- назначение уровня;
- сервис, предоставляемый данным уровнем расположенному над ним уровню;
- функции, выполняемые уровнем, и сервис, получаемый от расположенного под ним уровня.

Прикладной уровень является главным в иерархии, так как он непосредственно связан с прикладными процессами и обслуживает взаимодействие этих процессов. Остальные уровни помогают прикладному в выполнении этой задачи.

Представительный уровень обеспечивает прикладному уровню понимание смысла передаваемых по сети данных. Поэтому он описывает эти данные, в принятых стандартных формах и представляет информацию прикладным процессам. На этом уровне также обеспечивается в нужном случае преобразование формата синтаксиса данных для передачи их по сети.

Таблица 1.2. Функции, выполняемые уровнями

Уровень	Функции
7 (прикладной)	Интерфейс с прикладными процессами
6 (представительный)	Согласование формы представления информации (изображение, распечатка, строка символов...); формирование данных (двоичных, ASCII, международного алфавита № 5, EBCDIC, графических цифровых)
5 (сеансовый)	Поддержка диалога прикладных процессов; обеспечение соединения и разъединения этих процессов; осуществление обмена данными между прикладными процессами
4 (транспортный)	Обеспечение сквозного обмена информацией между системами
3 (сетевой)	Выбор вида сервиса сетевой маршрутизации; сегментирование и объединение блоков данных; обнаружение ошибок и сообщение о них
2 (канальный)	Управление каналом передачи данных; передача данных по каналу; обнаружение ошибок в канале и устранение их
1 (физический)	Обеспечение физического интерфейса с каналом передачи данных

Сеансовый уровень управляет передачей информации между прикладными процессами. Он позволяет обращаться к любому процессу по его имени независимо от того, в какой системе этот процесс находится. Благодаря этому прикладной процесс может перемещаться по сети. Например, при выходе из строя одной системы процесс может быть перемещен в другую, а пользователь об этом н не узнает.

Транспортный уровень определяет адресацию физических устройств (систем, их частей) в сети. Однако его главной задачей является обеспечение эффективных, удобных н надежных форм передачи информации между любой парой систем, в которых расположены взаимодействующие прикладные процессы. Этот уровень гарантирует доставку блоков информации адресатам, и управляет этой доставкой.

Сетевой уровень обеспечивает прокладку виртуальных каналов между взаимодействующими системами через (сквозь) коммутационную подсеть. *Виртуальный канал*—это такое функционирование компонентов сети, которое создает взаимодействующим объектам иллюзию прокладки между ними (только между ними) нужного тракта. На этом уровне также обеспечивается управление потоками передаваемых блоков данных, называемых *пакетами.*

Канальный уровень определяет процедуры передачи данных по *каналу* – совокупности аппаратных средств и физической среды непосредственно соединяющей системы. По каналу передаются блоки информации, именуемые *кадрами.* При больших размерах передаваемых блоков данных канальный уровень делит их на кадры и передает кадры в виде последовательностей. По получения кадров уровень формирует из них (восстанавливает) переданные блоки данных.

Физический уровень определяет интерфейсы системы с каналом коммуникационной подсети. Эти интерфейсы включают механические (муфты, соединители) и электрические (напряжение, ток, методы модуляции сигналов и т. д.) характеристики соединения. Уровень также описывает процедуры передачи сигналов в канал и получения их из канала.

В соответствии с рассмотренными функциями область взаимодействия открытых систем может быть разделена на три группы уровней. Первую из них, называемую обслуживающей образуют прикладной (7), представительный (6) и сеансовый (5) уровни. Задачей этой группы уровней является обслуживание прикладных процессов, в том числе обеспечение надежного и эффективного их взаимодействия. Важной особенностью обслуживающей группы является то, что уровни 5, 6, 7 «не знают» о существовании физических средств соединения, их видах, а поэтому и не зависят от того, как последние выполнены.

Обслуживающая часть области взаимодействия создает среду взаимодействия прикладных процессов. Поэтому она должна обеспечить удобную, надежную и разнообразную по формам связь этих процессов.

Вторую группу, образуемую сетевым (3), канальным (2) и физическим (1) уровнями, называют физической. Такое название связано с тем, что структура и функции этой группы определяются видом используемой коммуникационной подсети. Физическая группа уровней определяет доступ к коммуникационной подсети и обеспечивает передачу информации по этой подсети. Поэтому рассматриваемая группа может взаимодействовать только с определенными видами коммуникационной подсети. Однако физическая группа «не знает», для взаимодействия каких прикладных процессов предназначена, поэтому она должна обеспечить совместную работу любых прикладных процессов.

Между обслуживающей и физической располагается третья группа уровней области взаимодействия, включающая только один уровень – транспортный (4). Она освобождает обслуживающую часть от решения вопросов, связанных с передачей информации, и обеспечивает независимое от вида коммуникационной подсети взаимодействия любых пар абонентских систем. В соответствии со сказанным транспортная часть не зависит как от типов прикладных процессов, так и от видов коммуникационной подсети.

Архитектура Базовой эталонной модели предусматривает два способа передачи информации: с установлением и без установления соединения. *Взаимодействие с установлением соединения* предполагает, что перед передачей данных объекты двух взаимодействующих друг с другом систем выполняют процедуры, связанные с установкой этого соединения. *Взаимодействие без установления соединения* основано на том, что объекты систем знают все необходимое друг о друге заранее и направляют блоки информации, не предупреждая партнера. Второй способ проще первого. Однако адресат может быть в нерабочем состоянии либо не имеет возможности принять блоки в данный момент. В этих случаях передаваемые блоки сеть выбрасывает. Это естественно требует установления процедур получения подтверждения о том, что направленная информация получена адресатом.

Каждое соединение определяет функциональное взаимодействие двух либо больше объектов, расположенных на одном и том же уровне. Соединение между объектами уровня прокладывается .(рис. 2.2) через все уровни, расположенные под рассматриваемым уровнем, и коммуникационную подсеть.

При обеспечении взаимодействия с установлением соединения последнее создается проведением процедур установления, поддержания и расторжения соединения. Поэтому соединение объектов существует только во время их

взаимодействия, после чего аннулируется. Во время установления соединения о его создании должны договориться два объекта (инициатор и адресат), а также нижний уровень—поставщик необходимого сервиса. По установленному соединению передаются последовательности блоков данных до тех пор, пока один из объектов не заявит о расторжении соединения. Во время передачи блоков данных осуществляется управление их потоком, чтобы скорости работы соответствовали возможностям партнеров.

Если используется взаимодействие без установления соединения, то передача информации осуществляется двумя способами. Первый из них заключается в наличии двусторонней договоренности. Она состоит в знании адресов объектов, характера обмена данными, видов используемого сервиса и т. д. Имея эти сведения, объект без согласия партнера осуществляет передачу ему блоков данных. Второй способ используется в том случае, когда нет указанной выше двусторонней договоренности. В этом случае объект, которому нужно передать данные, обращается к партнеру с запросом о передаче ему нужных для работы сведений.

Каждый уровень взаимодействия систем определяется группой стандартизующих его документов, которые всегда включают две спецификации: протокол и сервис, обеспечиваемый этим протоколом для вышерасположенного уровня. *Протокол*— это свод правил и форматов, определяющих взаимодействие объектов, которые расположены на одном уровне. Протокол любого уровня обеспечивает *сервис* для расположенного над ним уровня. Этот сервис предоставляется благодаря передаче через сервисные точки доступа уровней специальных блоков данных, именуемых сервисными *примитивами.*

В соответствии с Базовой эталонной моделью взаимодействующие друг с другом системы обеспечивают совместную работу объектов, расположенных на одном уровне (рис. 1.7). Для выполнения этой работы объекты согласно протоколу передают друг другу протокольные блоки данных. Осуществляя работу, объекты уровни используют, сервис, предоставляемый уровнем, находящимся под ними.

Протокол определяет:

- процедуры передачи управляющей информации и данных между взаимодействующими объектами;
- механизм выбора указанных процедур из списка возможных;
- структуру и способ кодирования протокольных блоков данных.

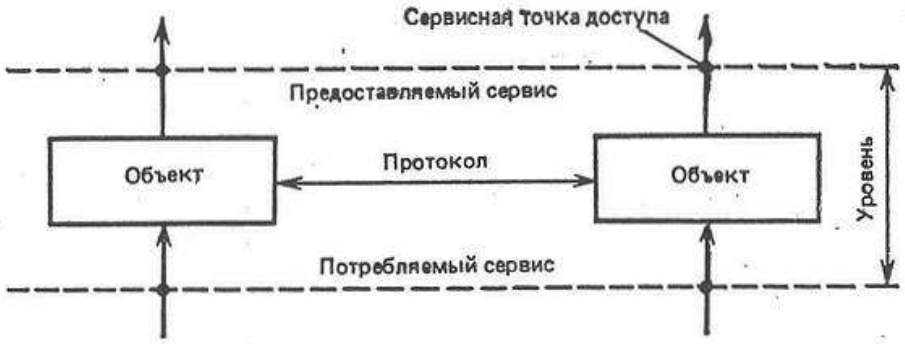


Рис. 1.7.- Модель взаимодействия объектов

Благодаря этому осуществляются взаимодействие объектов, использование сервиса нижерасположенного уровня, предоставление сервиса верхнему уровню. В результате системы, включенные в информационную сеть, управляют своими действиями и обеспечивают взаимодействие прикладных процессов.

Следует иметь в виду, что функции, выполняемые уровнем, может определять как один, так и несколько независимых друг от друга протоколов. При этом любой протокол «не знает», какие протоколы расположены на том же либо на соседних уровнях. Каждому протоколу «известен» лишь сервис, получаемый им с соседнего снизу уровня.

Процедуры, описываемые каждым протоколом, основаны на операциях теории конечных автоматов. При этом каждый протокол должен обладать свойствами полноты, непротиворечивости живучести и не приводить к тупиковым ситуациям. Рассмотрим эти требования.

Прежде всего, протокол должен обладать необходимой полнотой. Это значит, что следует устранить все неопределенные ситуации, в которых протокол «не знает», что ему нужно делать. Требование непротиворечивости связано с исключением ситуаций, когда протокол требует действий объекта, которые противоречат одно другому.

Тупиковые ситуации возникают тогда, когда несколько объектов приостанавливают свои действия, одновременно требуя предоставление одного и того же ресурса. В результате функционирование протокола прекращается и дальнейшая работа становится невозможной. Протокол должен также обладать высокой живучестью, обеспечивающей достижения необходимых целей даже при появлении помех в работе. Естественно, что протокол должен поддерживать объявленные виды сервиса, выполнять их за допустимое время и достигать конечного результата. При этом протокол при возникновении ошибок должен возвращаться в нормальное состояние за конечное время.

Для обеспечения гарантии того, что любой протокол выполняет указанные требования, его подвергают испытанию, именуемому верификацией. *Верификация* протокола является процессом его анализа, подтверждающим (либо опровергающим) то, что объекты рассматриваемого уровня, функционируя, обеспечивают выполнение поставленных задач и предоставляют вышестоящему уровню требуемые виды сервиса.

Таким образом, система, работающая в информационной сети, должна выполнять все функции области взаимодействия открытых систем. Но кроме этого, естественно, она должна обеспечивать работу прикладных процессов. Введение функций области взаимодействия в любую операционную систему электронной машины (либо машин), образующих систему, требует значительных усилий. Это связано с тем, что последняя должна управлять как прикладными процессами, так уровневыми функциями области взаимодействия.

Вопросы к главе 1.

1. Задачи информационной инфраструктуры науки и техники на современном этапе
2. Определение информационной системы и информационной сети. Классификация прикладных процессов. Абонентская система. Компоненты информационных сетей
3. Базовая эталонная модель Международной организации стандартов (ISO). Область взаимодействия открытых систем.
4. Назначение уровня. Сервис и функции, предоставляемый уровнем.
5. Модель взаимодействия объектов. Определение протокола взаимодействия объектов

Гава 2. Эволюция архитектуры информационных систем. Архитектура «клиент-сервер»

Функции стандартного приложения можно разделить на три группы, имеющие различную природу. Первая группа - это функции ввода и отображения данных. Вторая группа объединяет чисто прикладные функции, характерные для данной предметной области. Наконец, к третьей группе относятся фундаментальные функции хранения и управления данными.

В соответствии с этим в любом приложении можно выделить следующие логические компоненты:

1. компонент представления (presentation), реализующий функции первой группы;
2. прикладной компонент (business application), поддерживающий функции второй группы;
3. компонент доступа к информационным ресурсам (resource access) или менеджер ресурсов (resource manager), поддерживающий функции третьей группы.

Различия в реализации приложений в рамках технологий "хост/терминал" и "клиент/сервер" определяются тем, какие механизмы используются для реализации функций всех трех групп, и тем, как логические компоненты распределяются между компьютерами в сети.

2.1. Архитектура "хост/терминал"

В архитектуре "хост/терминал" (рис. 2.1) функции всех трех групп совмещены в одном коде, который выполняется на компьютере-сервере (хосте). Компьютер-клиент в данной архитектуре отсутствует в принципе, а ввод и отображение данных производятся через терминал или компьютер в режиме эмуляции терминала. Приложения обычно разрабатываются на языке четвертого поколения (4GL).

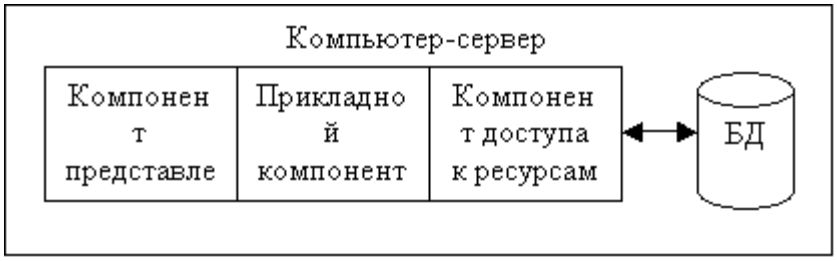


Рис. 2.1. Архитектура "хост/терминал".

Преимуществами данной архитектуры являются:

1. Простота разработки приложений.

2. Удобство администрирования и обновления ПО, т.к. все части прикладной системы размещаются на одном компьютере.
3. Низкий трафик, создаваемый в сети, т.к. по сети пересылаются только данные, вводимые пользователем, и данные, отображаемые на экране. Благодаря этому возможна работа по низкоскоростным линиям.
4. Низкая стоимость оборудования рабочих мест. На рабочих местах можно использовать терминалы или дешевые компьютеры с невысокими характеристиками в режиме эмуляции терминала.

К недостаткам можно отнести:

1. Высокие требования ко времени отклика в сети. Несмотря на небольшой объем данных, пересылаемых по сети, время отклика является критичным, т.к. каждый символ, введенный пользователем на терминале, должен быть передан на сервер, обработан приложением и возвращен обратно для вывода на экран терминала.
2. Высокие требования к характеристикам компьютера-сервера, т.к. все пользователи разделяют его ресурсы.
3. Невозможность распределения нагрузки между несколькими компьютерами.
4. Невозможность использования графического интерфейса.

2.2. Архитектура "клиент/сервер"

В архитектуре "клиент/сервер" функции приложения распределены между двумя (или более) компьютерами. В соответствии с тем, каким образом это сделано, выделяются три модели архитектуры "клиент/сервер":

1. Модель доступа к удаленным данным (Remote Data Access - RDA);
2. Модель сервера базы данных (DataBase Server - DBS);
3. Модель сервера приложений (Application Server - AS).

RDA-модель

В RDA-модели (рис. 2.2) коды компонента представления и прикладного компонента совмещены и выполнятся на компьютере-клиенте. Последний поддерживает как функции ввода и отображения данных, так и прикладные функции ("толстый" клиент).

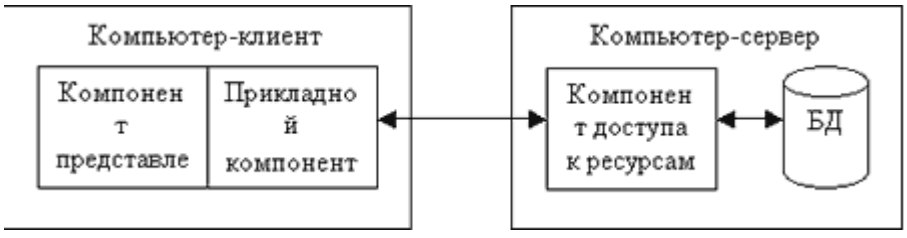


Рис. 2.2.- RDA-модель.

Доступ к информационным ресурсам обеспечивается, как правило, операторами специального языка (например, SQL) или вызовами функций специальной библиотеки (если имеется соответствующий API). Запросы к информационным ресурсам направляются по сети удаленному компьютеру-серверу базы данных. Последний обрабатывает и выполняет запросы и возвращает клиенту блоки данных. Говоря об архитектуре "клиент/сервер", в большинстве случаев имеют в виду именно эту модель.

Основным преимуществом RDA-модели является широкий выбор средств быстрой разработки приложений (RAD) различных фирм. Существует множество инструментальных средств, обеспечивающих быстрое создание приложений, работающих с SQL-ориентированными СУБД. Большинство из них поддерживают графический интерфейс пользователя в MS Windows, стандарт интерфейса ODBC, содержат средства автоматической генерации кода. Подавляющее большинство этих средств разработки на языках четвертого поколения (включая и средства автоматизации программирования) как раз и создают коды, в которых смешаны прикладные функции и функции представления.

В то же время RDA-модель имеет ряд ограничений.

1. Очень большая загрузка сети. Приложение является нераспределенным, и вся его логика локализована на компьютере-клиенте, поэтому взаимодействие его с сервером посредством SQL-запросов приводит к передаче по сети данных большого объема, возможно, избыточных. Как только число клиентов возрастает, сеть становится узким местом, ограничивая быстродействие всей информационной системы.
2. Сложность ведения больших проектов. Очевидно, что если различные по своей природе функции (функции представления и чисто прикладные функции) смешаны в одной и той же программе, написанной на языке 4GL, то при необходимости изменения прикладных функций приходится переписывать всю программу целиком. При коллективной работе над проектом, как правило, каждому разработчику поручается реализация отдельных прикладных функций, что делает невозможным контроль за их взаимной непротиворечивостью. Каждому из разработчиков приходится программировать интерфейс с пользователем, что ставит под вопрос единый стиль интерфейса и его целостность.
3. Сложность обновления программного обеспечения, т.к. его замену необходимо производить одновременно на всех компьютерах-клиентах.
4. Низкий уровень безопасности, т.к. реализация разграничения доступа по функциям возможна только на стороне клиента, а на стороне сервера

разграничение выполняется только по таблицам базы данных, что снижает защищенность.

DBS-модель

В DBS-модели (рис. 2.3) процесс, выполняемый на компьютере-клиенте, ограничивается функциями представления ("тонкий" клиент), а прикладные функции реализованы в хранимых процедурах (stored procedure), которые также называют компилируемыми резидентными процедурами, или процедурами базы данных. Они хранятся непосредственно в базе данных и выполняются на компьютере-сервере базы данных, где функционирует и компонент, управляющий доступом к данным, то есть ядро СУБД.

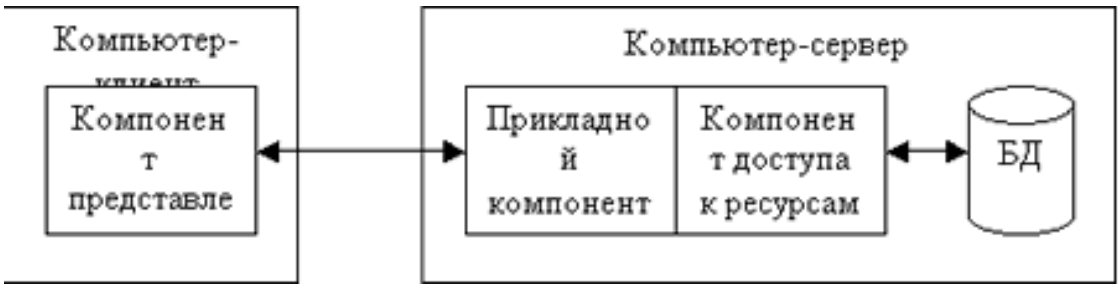


Рис. 2.3.- DBS-модель.

DBS-модель реализована в некоторых реляционных СУБД (Ingres, Sybase, Oracle). Ее основу составляет механизм хранимых процедур - средство программирования ядра СУБД. Процедуры хранятся в словаре базы данных, разделяются между несколькими клиентами и выполняются на том же компьютере, где функционирует ядро СУБД. Язык, на котором разрабатываются хранимые процедуры, представляет собой процедурное расширение языка запросов SQL.

Преимущества DBS-модели очевидны:

1. Возможность централизованного администрирования бизнес-функций, размещенных на сервере.
2. Снижение трафика в сети.
3. Возможность разделения процедуры между несколькими приложениями, и экономия ресурсов компьютера за счет использования единожды созданного плана выполнения процедуры.

Однако есть и недостатки:

1. Средства, используемые для написания хранимых процедур, строго говоря, не являются языками программирования в полном смысле слова. Это разнообразные процедурные расширения SQL, не выдерживающие сравнения по изобразительным средствам и функциональным возможностям с языками третьего поколения (C или Pascal) и тем более

четвертого поколения. Они встроены в конкретные СУБД, и, естественно, рамки их использования ограничены. Следовательно, система, в которой прикладной компонент реализован при помощи хранимых процедур, не является мобильной относительно СУБД. В большинстве СУБД отсутствуют возможности отладки и тестирования хранимых процедур, что превращает последние в весьма опасный механизм. Во многих реализациях процедуры являются интерпретируемыми, что делает их выполнение более медленным.

2. Не обеспечивается требуемой эффективности использования вычислительных ресурсов. Объективные ограничения в ядре СУБД не позволяют пока организовать в его рамках эффективный баланс загрузки, миграцию процедур на другие компьютеры-серверы БД и реализовать другие полезные функции. Попытки разработчиков СУБД предусмотреть в своих системах эти возможности (распределенные хранимые процедуры, запросы с приоритетами и т. д.) пока не позволяют добиться желаемого эффекта.
3. Децентрализация приложений (один из ключевых факторов современных информационных технологий) требует существенного разнообразия вариантов взаимодействия клиента и сервера. При реализации прикладной системы могут понадобиться такие механизмы взаимодействия, как хранимые очереди, асинхронные вызовы и т. д., которые в DBS-модели не поддерживаются.

На практике часто используются смешанные модели, когда поддержка целостности базы данных и некоторые простейшие прикладные функции поддерживаются хранимыми процедурами (DBS-модель), а более сложные функции реализуются непосредственно в прикладной программе, которая выполняется на компьютере-клиенте (RDA-модель).

AS-модель

В AS-модели (рис. 2.4) процесс, выполняющийся на компьютере-клиенте, отвечает, как обычно, за ввод и отображение данных (то есть реализует функции первой группы). Прикладные функции выполняются группой процессов (серверов приложений), функционирующих на удаленном компьютере (или нескольких компьютерах). Доступ к информационным ресурсам, необходимым для решения прикладных задач, обеспечивается таким же способом, что и в RDA-модели. Серверы приложений выполняются, как правило, на том же компьютере, где функционирует менеджер ресурсов, однако могут выполняться и на других компьютерах.

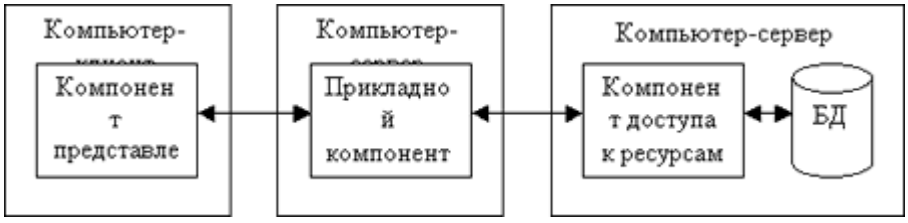


Рис.2. 4.- AS-модель

Основным элементом принятой в AS-модели трехзвенной схемы является сервер приложения. В его рамках реализовано несколько прикладных функций, каждая из которых оформлена как служба (service) и предоставляет некоторые услуги всем программам, которые желают и могут ими воспользоваться. Серверов приложений может быть несколько, и каждый их них предоставляет определенный набор услуг. Любая программа, которая пользуется ими, рассматривается как клиент приложения (Application Client - AC). Детали реализации прикладных функций в сервере приложений полностью скрыты от клиента приложения. AC обращается с запросом к конкретной службе, но не к AS, то есть серверы приложений обезличены и служат лишь своего рода "рамкой" для оформления служб, что позволяет эффективно управлять балансом загрузки. Запросы, поступающие от AC, выстраиваются в очередь к AS-процессу, который извлекает и передает их для обработки службе в соответствии с приоритетами.

AC трактуется более широко, чем компонент представления. Он может поддерживать интерфейс с конечным пользователем (тогда он является компонентом представления), может обеспечивать поступление данных от некоторых устройств (например, датчиков), может, наконец, сам по себе быть AS. Последнее позволяет реализовать прикладную систему, содержащую AS нескольких уровней. Архитектура такой системы может выглядеть как ядро, окруженное концентрическими кольцами. Ядро состоит из серверов приложений, в которых реализованы базовые прикладные функции. Кольца символизируют наборы AS, являющихся клиентами по отношению к серверам нижнего уровня. Число уровней серверов в AS-модели, вообще говоря, не ограничено.

AS-модель в наибольшей степени отражает сильные стороны технологии "клиент/сервер":

1. Четкое разграничение логических компонентов приложения.
2. Возможность баланса загрузки между несколькими серверами.
3. Значительное снижение трафика между клиентом и сервером приложений, дающее возможность работы по медленным линиям связи.

4. Высокий уровень защиты данных, т.к. они являются "спрятанными" за сервисами приложения, в которые можно встроить проверку полномочий клиента.
5. Возможность использования в качестве клиентской части приложения стандартного броузера.
6. Упрощение процесса обновления ПО.

Фундаментальное различие между моделями архитектуры "клиент/сервер" заключается в следующем. RDA- и DBS-модели опираются на двухзвенную схему разделения функций. В RDA-модели прикладные функции приданы программе-клиенту, в DBS-модели ответственность за их выполнение берет на себя ядро СУБД. В первом случае прикладной компонент сливается с компонентом представления, во втором - интегрируется в компонент доступа к информационным ресурсам. Напротив, в AS-модели реализована классическая трехзвенная схема разделения функций, где прикладной компонент выделен как важнейший элемент приложения, для его определения используются универсальные механизмы многозадачной операционной системы, и стандартизованы интерфейсы с двумя другими компонентами. Собственно, из этой особенности AS-модели и вытекают ее преимущества.

2.3. Архитектура «клиент-сервер», основанная на Web-технологии

Многие недостатки, присущие компьютерным сетям с классической архитектурой «клиент-сервер», отсутствуют в вычислительных системах новой архитектуры, которые сконцентрировали и объединили в себе лучшие качества централизованных систем и классических систем «клиент-сервер». Новая архитектура компьютерных сетей была названа интранет-архитектура. Ее часто называют также Web-архитектурой, или архитектурой «клиент-сервер», основанной на Web-технологии. Эта архитектура явилась итогом многолетних исследований и разработок в области приложения глобальных сетевых технологий Интернет к локальным сетям. Появление в 1993 г. архитектуры интранет относят к началу третьего этапа эволюции вычислительных систем.

Основной особенностью архитектуры интранет является возвращение к серверам ряда функций, которые были вынесены за пределы центральной ЭВМ на втором этапе эволюции вычислительных систем. Базисом новой архитектуры является Web-технология, пришедшая из Интернета.

В соответствии с Web-технологией на сервере размещаются так называемые Web-документы, которые визуализируются и интерпретируются программой навигации, функционирующей на рабочей станции (рисунок 2.5). Программу навигации называют еще Web-навигатором, или Web-браузером.

Рисунок 2.5. - Архитектура «клиент-сервер», основанная на Web-технологии

Логически Web-документ представляет собой гипермедийный документ, объединяющий ссылками различные Web-страницы, каждая из которых может содержать ссылки и на другие объекты. Физически Web-документ представляет собой текстовый файл специального формата, содержащий ссылки на другие объекты и Web-документы, расположенные в любом узле сети. Web-документ реально включает только одну Web-страницу, но логически может объединять любое количество таких страниц, принадлежащих различным Web-документам.

Web-страница, являясь информационным аналогом страницы бумажного носителя, может включать как текст, так и рисунки. Но, в отличие от бумажной страницы, Web-страница может быть связана с компьютерными программами и содержать ссылки на другие объекты. Программа, связанная с Web-страницей, начинает автоматически выполняться при переходе по соответствующей ссылке или открытии Web-страницы. Любые ссылки, включенные в Web-страницу, выделяются другим цветом и/или подчеркиванием. Для перехода по ссылке достаточно щелкнуть по ней мышью.

Получаемая таким образом система гиперссылок основана на том, что некоторые выделенные участки одного документа, которыми могут быть части текста и рисунки, выступают в качестве ссылок на другие логически связанные с ними объекты. При этом объекты, на которые делаются ссылки, могут

находиться на любом компьютере сети. В Web-страницу могут быть включены ссылки на следующие объекты:

- другую часть Web-документа;
- другой Web-документ или документ другого формата (например, документ Word или Excel), который может размещаться на любом компьютере сети;
- мультимедийный объект - рисунок, звук, видео;
- программу, которая при переходе на нее по ссылке будет выполняться на сервере;
- программу, которая при переходе на нее по ссылке будет передана с сервера на рабочую станцию для интерпретации или запуска на выполнение навигатором;
- любой другой сервис - электронную почту, копирование файлов с другого компьютера сети, поиск информации и т.д.

Из раскрытого понятия Web-документа становится ясно, что программа навигации, выполняемая на рабочей станции, может не только визуализировать Web-страницы и выполнять переходы к другим объектам, но и активизировать программы на сервере, а также интерпретировать и запускать на выполнение программы, относящиеся к Web-документу, для исполнения на рабочей станции.

Передачу с сервера на рабочую станцию документов и других объектов по запросам, поступающим от навигатора, обеспечивает функционирующая на сервере программа, называемая Web-сервером. Когда Web-навигатору необходимо получить документы или другие объекты от Web-сервера, он отправляет серверу соответствующий запрос. При достаточных правах доступа между сервером и навигатором устанавливается логическое соединение. Далее сервер обрабатывает запрос, передает Web-навигатору результаты обработки, например требуемый Web-документ, и разрывает установленное соединение.

Web-сервер выступает в качестве информационного концентратора, который доставляет информацию из разных источников, а потом однородным образом предоставляет ее пользователю. Навигатор, снабженный универсальным и естественным интерфейсом с человеком, позволяет последнему легко просматривать информацию вне зависимости от ее формата.

Таким образом, в рамках Web-документа может быть выполнена интеграция данных и программных объектов различных типов, расположенных в совершенно разных узлах компьютерной сети. Это позволяет рассредоточивать информацию в соответствии с естественным порядком ее создания и

потребления, а также осуществлять единообразный доступ. Приставка Web здесь, а также в названии самой технологии (англ. web - паутина), как раз и отражает тот факт, что работа пользователя осуществляется на основе перехода по ссылкам, которые как нити паутины связывают разнотипные объекты, распределенные по узлам компьютерной сети.

Web-документы, помимо связывания распределенных и разнотипных данных, позволяют рассматривать информацию с нужной степенью детализации, что существенно упрощает анализ больших объемов информации. Можно сосредоточить внимание на главном, а затем изучить выбранный материал в подробностях. Можно эффективно реализовать многомодельный подход представления материала, создавая различные «взгляды» на требуемую предметную область, отражающие точки зрения той или иной группы сотрудников организации.

Компьютер-клиент, на котором должна выполняться программа навигации, может быть полностью стандартизован. В такой компьютер помимо процессора, основной памяти и монитора достаточно включить небольшой участок внешней памяти, необходимый для хранения и работы программы навигации, а также устройство сопряжения с линией связи. Кроме того, программу навигации можно реализовать аппаратно в специализированном процессоре.

Исходя из изложенного выше можно выделить следующие отличительные черты интранет-архитектуры:

- помимо своих программ могут выполняться программы с других компьютеров сети. на сервере порождается конечная информация, предназначенная для представления пользователю программой навигации, а не полуфабрикат, как в системах с классической архитектурой «клиент-сервер»;
- все информационные ресурсы, а также прикладная система сконцентрированы на сервере;
- для обмена данными между клиентами и сервером используются протоколы открытого стандарта TCP/IP, применяемые в Интернете;
- облегчено централизованное управление не только сервером, но и компьютерами-клиентами, так как они стандартизованы с точки зрения программного обеспечения (на каждой рабочей станции достаточно наличия стандартной программы навигации);
- на рабочих станциях

Предполагается, что перечисленные особенности, за исключением последней, способствуют решению проблемы информационно-компьютерной безопасности.

Концентрация на сервере информации и прикладной системы существенно упрощает построение и администрирование системы безопасности. Использование для обмена данными между компьютерами сети протоколов открытого стандарта TCP/IP приводит к унификации всех способов взаимодействия между рабочими станциями и сервером. Не нужно решать задачу обеспечения безопасного информационного взаимодействия для множества приложений каждого компьютера. Решение по безопасности взаимодействия для одного компьютера и будет стандартным для всех. Кроме того, по отношению к протоколам открытого стандарта намного интенсивнее и шире публичное обсуждение вопросов информационной безопасности и богаче выбор защитных средств. Облегченное централизованное управление сервером и компьютерами-клиентами снижает вероятность допущения непреднамеренных ошибок пользователями, операторами и администраторами. Такие ошибки являются одной из основных угроз безопасности информации и приводят к прямому ущербу. К ним относятся неправильно введенные данные или ошибки в программе, вызвавшие остановку либо разрушение системы. Эти ошибки также создают слабые места, которыми могут воспользоваться злоумышленники. В интранет-архитектуре распределенная обработка информации предполагает возможность выполнения на рабочих станциях программ, полученных с сервера. Такой вид распределенной обработки позволил сконцентрировать всю прикладную систему на сервере. Однако возможность выполнения на рабочих станциях программ с сервера порождает новые угрозы безопасности информации, например появляется угроза подмены передаваемой с сервера программы. Соответственно возможность миграции программ предъявляет дополнительные требования по поддержанию безопасности сетевого взаимодействия.

Вопросы к главе 2.

1. Эволюция архитектуры информационных систем. Архитектура «клиент-сервер»
2. Архитектура «клиент-сервер», основанная на Web-технологии

Глава 3. Технологии и системы организации высокопроизводительных вычислений в классе ресурсов информационных инфраструктур ***[***Ивашко Е. Е. /Современные технологии высокоскоростных вычислений/. Петрозаводск, 2016***]***

Высокопроизводительные вычисления — это одна из наиболее актуальных и «горячих» тем, как в информационных технологиях, так и во многих прикладных областях. Причиной этого является та практическая польза, которую нельзя получить никакими другими технологиями, кроме как применением высокопроизводительных вычислений. Под термином «высокопроизводительные вычисления» (high-performance computing — HPC) обычно подразумевают не только выполнение большого объема расчетов, но и обработку больших объемов данных за сравнительно небольшой промежуток времени.

Существует много фундаментальных и прикладных задач, для эффективного решения которых требуется привлечение технологий высокопроизводительных вычислений — все эти задачи объединены в группу «Grand Challenges»:

- предсказания погоды и климата;
- исследование материалов;
- построение полупроводниковых приборов;
- разработка фармацевтических препаратов;
- генетика;
- астрономия;
- транспортные задачи;
- гидро- и газодинамика;
- управляемый термоядерный синтез;
- эффективность систем сгорания топлива;
- геоинформационные системы;
- разведка недр;
- наука о мировом океане;
- распознавание и синтез речи;
- распознавание изображений.

Как правило, о высокопроизводительных вычислениях можно говорить тогда, когда к ***программно-аппаратной системе предъявляется одно или несколько из следующих требований***:

- высокое быстродействие;
- наличие большого объема оперативной памяти;

- необходимость передавать большие объемы данных;
- необходимость хранить и обрабатывать большие объемы данных.

Проиллюстрируем эти требования на примерах.

Итак, есть задачи, которые можно решить только с помощью высокопроизводительных вычислений. Какие же существуют инструменты?

Вот совсем простая классификация компьютеров (см. рисунок 3.1.). Самые маленькие вычислительные способности у мобильных устройств, зато число их огромно. Следом за ними (по росту производительности и уменьшению количества) идут персональные компьютеры, затем серверы. Наконец, вершина вычислительных устройств — суперкомпьютеры.

Рис. 3.1. - Классификация компьютеров

Такая классификация интуитивно понятна, но все же нужно определить по какому параметру мы выстраиваем компьютеры. Конечно самый простой параметр — это производительность. Сейчас мы будем говорить о пиковой — т.е. теоретически максимально возможной или верхней границе — производительности вычислительного устройства (компьютера). Так уж повелось, что производительность вычисляют в числе операций с плавающей запятой, которые может выполнить компьютер за одну секунду (**Flo**ating **Op**erations per **S**econd — FLOPS, флопс).

Пиковая производительность современного персонального компьютера составляет примерно до 100 Гфлопс. В промежутке от 100 до 500 Гфлопс имеют производительность специализированные сервера. Производительность

самых мощных суперкомпьютеров мира начинается с отметки в 100 Тфлопс.

Однако не нужно пытаться использовать эти цифры для классификации вычислительных устройств. Почему?

Оксфордский толковый словарь по вычислительной технике, изданный в 1986 году, сообщает, что *«**суперкомпьютер** — это очень мощная ЭВМ с производительностью свыше 10 MFLOPS (миллионов операций с плавающей запятой в секунду)»*. Сегодня этот результат перекрывают уже не только персональные компьютеры и мобильные телефоны, но и программируемые калькуляторы. В начале 90-х годов границу суперкомпьютера проводили около отметки в 300 MFLOPS.

В 1996 году специалисты двух ведущих "суперкомпьютерных" стран — США и Японии договорились о подъеме планки суперкомпьютера до 5 GFLOPS.

Сейчас суперкомпьютер, занимающий последнее место в списке 500 самых мощных суперкомпьютеров мира имеет производительность 250 Тфлопс[3].

Следовательно, любое определение суперкомпьютера, основанное на указании минимальной производительности, устаревает очень быстро!

Иллюстрация этого тезиса — закон Мура для суперкомпьютеров (см. рис. 3.2.).

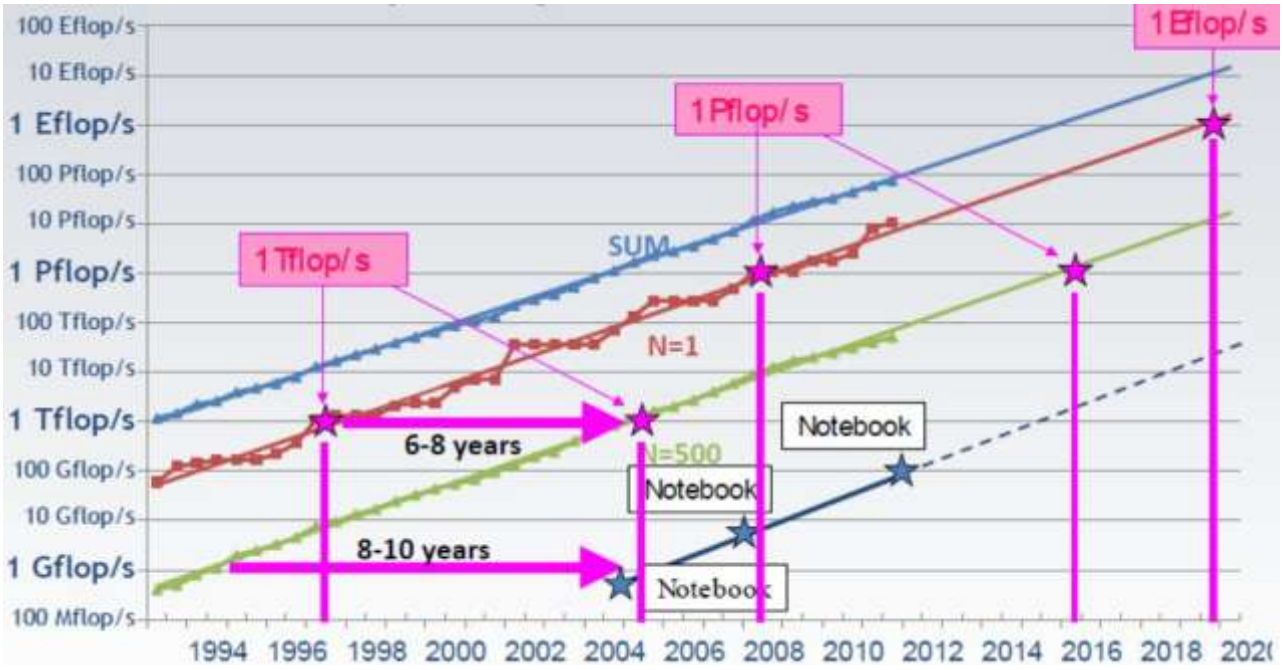


Рис. 3.2. - Закон Мура для суперкомпьютеров

Кроме того, пиковая производительность компьютера очень сильно отличается от его реальной производительности! Реальная производительность, которую компьютер может показать при решении практической задачи, зависит

от многих параметров:

- специфики конкретной решаемой задачи;
- алгоритма решения задачи;
- быстродействия оперативной памяти;
- быстродействия подсистемы ввода-вывода;
- и др.

---

3 По данным TOP500 на ноябрь 2015 г. www.top500.org

3.1. Что такое суперкомпьютер. Проблемы создания

Наиболее простой путь сделать очень мощный компьютер — это все больше и больше наращивать его производительность. Так и создавались суперкомпьютеры.

Разработка суперкомпьютера — это создание все более и более мощного компьютера. Чтобы понять как это делается, необходимо представлять себе общую схему компьютера и те узлы, где теряется или существуют ограничения на рост производительности (см. рисунок 3.4).

Рис. 3.4. - Схема вычислительного устройства

Основные компоненты компьютера следующие:

- *устройство управления* — управляет ходом вычислительного процесса, обеспечивая декодирование инструкций и автоматическое выполнение команд программы;

• *арифметико-логическое устройство* — блок вычислителя, который служит для выполнения арифметических и логических преобразований над данными под с помощью специального устройства управления;

• *оперативная память* хранит исходные данные и команды, которые необходимо выполнить над этими данными;

• *устройства ввода-вывода* по мере необходимости обеспечивают загрузку дополнительных данных и команд, не вмещающихся в оперативную память.

При разработке суперкомпьютера каждая из этих компонент может стать «узким местом», ограничивающим производительность. Не меньшее значение имеют и связи между компонентами, обеспечивающие передачу данных и управляющих команд.

Необходимость ликвидации «узких мест» привела разработчиков к созданию такого микропроцессора, общая схема которого представлена на рисунок 3.5.

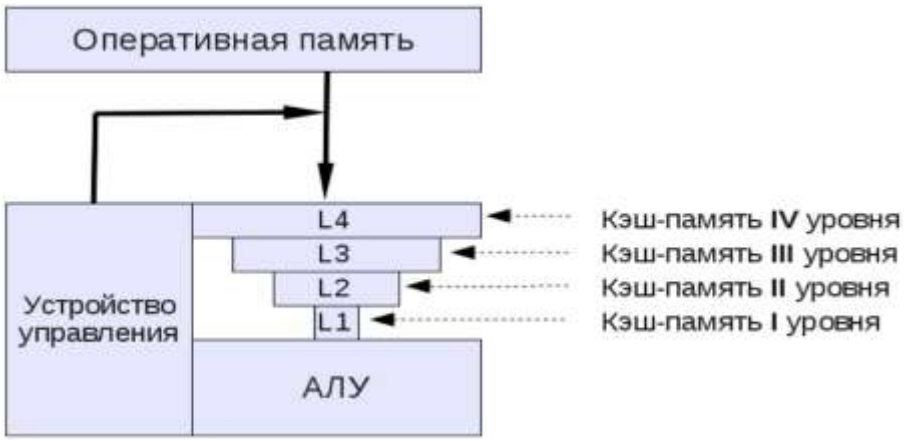


Рис. 3.5. - Общая схема современного компьютера

Были предприняты следующие шаги.

Во-первых, линии связи между компонентами необходимо было сделать как можно более короткими: устройство управления было совмещено с вычислителем (арифметико-логическим устройством), а получившийся блок был дополнен быстрой памятью (кэш- память). Каждый уровень кэш-памяти размещается ближе к потребителю (арифметико-логическим устройством), а значит, предоставляет более быстрый доступ к операндам, над которыми осуществляются операции. Однако быстрой памяти нельзя сделать много — поэтому существует несколько уровней кэш-памяти — каждый из них медленнее предыдущего, но вмещает больше данных. Это позволяет снизить задержки,

связанные с обращением к оперативной, а тем более внешней памяти.

Во-вторых, когда связи между компонентами и устройства ввода-вывода не мешают наращивать производительность, повышается скорость работы вычислительного блока (устройство управления + арифметико-логическое устройство).

Как уже говорилось выше, мощность компьютера определяется, прежде всего, пиковой производительностью — максимальным числом операций над числами с плавающей запятой, которое компьютер может произвести в секунду. (*Заметим, что пиковая производительность — это характеристика вычислительного блока, т.к. она не учитывает ограничения производительности, накладываемые, например, оперативной памятью.*)

Пиковая производительность вычисляется по следующей формуле:

*FLOPSpeak = Hz * C, где*

Hz — тактовая частота процессора;

C — максимальное число операций над числами с плавающей запятой, которое процессор может выполнить за один такт.

Следовательно, чтобы повысить производительность вычислительного блока, необходимо повысить тактовую частоту процессора и/или увеличить максимальное число операций, выполняемых за один такт. Однако и на том, и на другом пути существуют свои ограничения.

На рисунке 3.6 показан график тактовой частоты производящихся процессоров в зависимости от технологического процесса. Уменьшение технологического процесса позволяло повышать тактовую частоту процессоров. Однако начиная с техпроцесса примерно 65 нанометров, тактовая частота используемых в устройствах процессоров стала падать. Почему? Проблема заключается в тепловыделении процессора. Удвоение тактовой частоты приводит к учетверению тепловыделения, а значит, при высокой тактовой частоте возникает серьезная проблема отвода избыточного тепла от процессора. 130 Вт — это максимальная мощность, которая может отводиться от полупроводникового прибора размерами теплопроводящих поверхностей и конструкции микропроцессора. Приближение к 150 Вт грозит местными перегревами на кристалле, снижением его помехоустойчивости, чувствительности к внешним охлаждающим устройствам и соответственно, общей надежности. Тогда для повышения производительности вычислителя необходимо повысить число операций, выполняемых процессором за один такт. Этот путь называется *векторизацией*.

Действительно, векторизация позволяет существенно повысить пиковую

производительность компьютера. Однако пиковая производительность, как говорилось выше, отличается от реальной производительности, которую может показать компьютер на практически важных задачах.

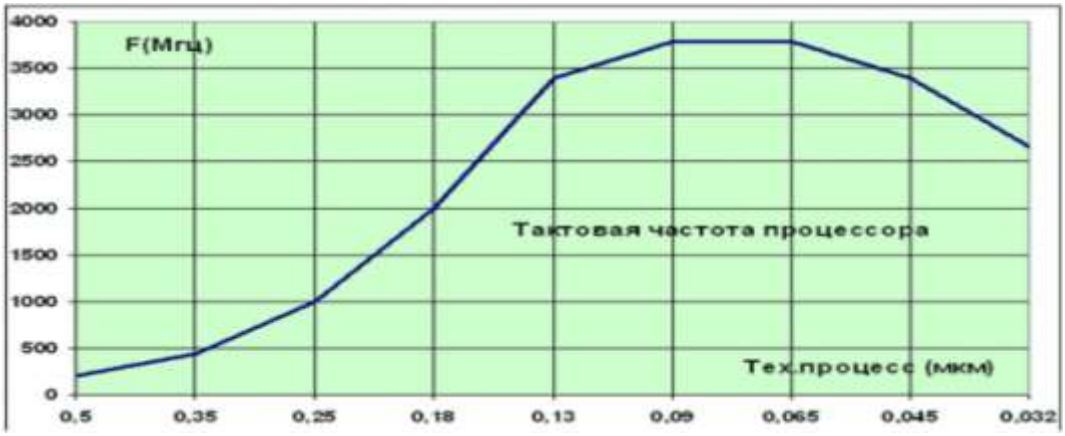


Рис. 3.6. - Тактовая частота процессора в зависимости от техпроцесса

Широкое использование векторных процессоров имеет следующие проблемы:

- *стоимость*: векторные процессоры существенно более дорогие и более сложные;
- *низкая эффективность*: эффективное программирование векторно-параллельных систем значительно сложнее, чем скалярных, т.к.
  - не все алгоритмы могут быть векторизованы;
  - низкая загрузка процессорных элементов: высокая производительность достигается только на векторных операциях, совпадающих по размеру с размером вектора процессора, в то время как на скалярных операциях, а также при обработке векторов и матриц небольшой размерности, значительная часть процессорных элементов может простаивать.

В целом, векторные системы характеризуются высокой производительностью только при полной загрузке вычислительных устройств, которая достигается при решении только достаточно узкого класса задач.

3.2. Суперкомпьютеры и особенности их архитектуры

Когда-то было дано полу-шуточное определение суперкомпьютера: ***суперкомпьютер*** — это устройство, сводящее проблему вычислений к проблеме ввода/вывода. Смысл этого определения заключается в том, что суперкомпьютер должен выполнять вычисления настолько быстро, что время расчетов будет мало по сравнению со временем, необходимым для того, чтобы ввести в программу исходные данные и вывести результаты. Более формальное определение звучит так:

суперкомпьютер *— это специализированная вычислительная машина, значительно превосходящая по своим техническим параметрам и скорости вычислений большинство существующих в мире компьютеров.*

К суперкомпьютерам относятся мощные многопроцессорные вычислительные машины с быстродействием сотни миллионов — десятки миллиардов операций в секунду. Создать такие высокопроизводительные компьютеры на одном микропроцессоре (МП) не представляется возможным ввиду ограничения, обусловленного конечным значением скорости распространения электромагнитных волн (300 000 км/с), т.к. время распространения сигнала на расстояние несколько миллиметров (линейный размер стороны МП) при быстродействии 100 млрд операций/с становится соизмеримым со временем выполнения одной операции. Поэтому суперкомпьютеры создаются в виде высокопараллельных многопроцессорных **вычислительных систем** (МПВС).

Термин вычислительная система появился в начале - середине 60-х гг. при появлении ЭВМ III поколения. Это время знаменовалось переходом на новую элементную базу - интегральные схемы. Следствием этого явилось появление новых технических решений: разделение процессов обработки информации и ее ввода-вывода, множественный доступ и коллективное использование вычислительных ресурсов в пространстве и во времени. Появились сложные режимы работы ЭВМ - многопользовательская и многопрограммная обработка.

Под ***вычислительной системой (ВС)*** понимают совокупность взаимосвязанных и взаимодействующих процессоров или ЭВМ, периферийного оборудования и программного обеспечения, предназначенную для сбора, хранения, обработки и распределения информации. ***Отличительной особенностью ВС по отношению к ЭВМ*** является ***наличие в них нескольких вычислителей, реализующих параллельную обработку***. Создание ВС преследует следующие основные цели: повышение производительности системы за счет ускорения процессов обработки данных, повышение надежности и достоверности вычислений, предоставление пользователям дополнительных сервисных услуг и т.д. На рис. 3.7 представлена принципиальная схема классификации вычислительных систем.

Высокопараллельные МПВС имеют несколько разновидностей.

1. **Магистральные** (конвейерные) МПВС, у которых процессор одновременно выполняет разные операции над последовательным потоком обрабатываемых данных. По принятой классификации такие МПВС

относятся к системам с многократным потоком команд и однократным потоком данных (МКОД или **MISD** — Multiple Instruction Single Data).

2. **Векторные** МПВС, у которых все процессоры одновременно выполняют одну команду над различными данными — однократный поток команд с многократным потоком данных (ОКМД или **SIMD** — Single Instruction Multiple Data).
3. **Матричные** МПВС, у которых микропроцессор одновременно выполняет разные операции над последовательными потоками обрабатываемых данных — многократный поток команд с многократным потоком данных (МКМД или **MIMD** — Multiple Instruction Multiple Data).

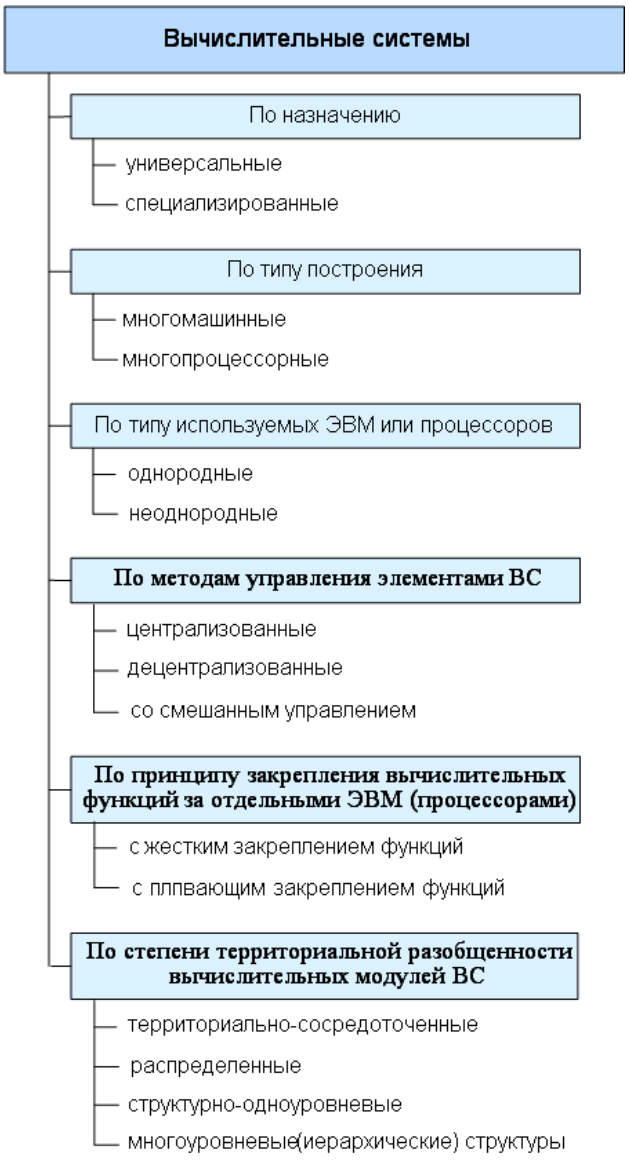
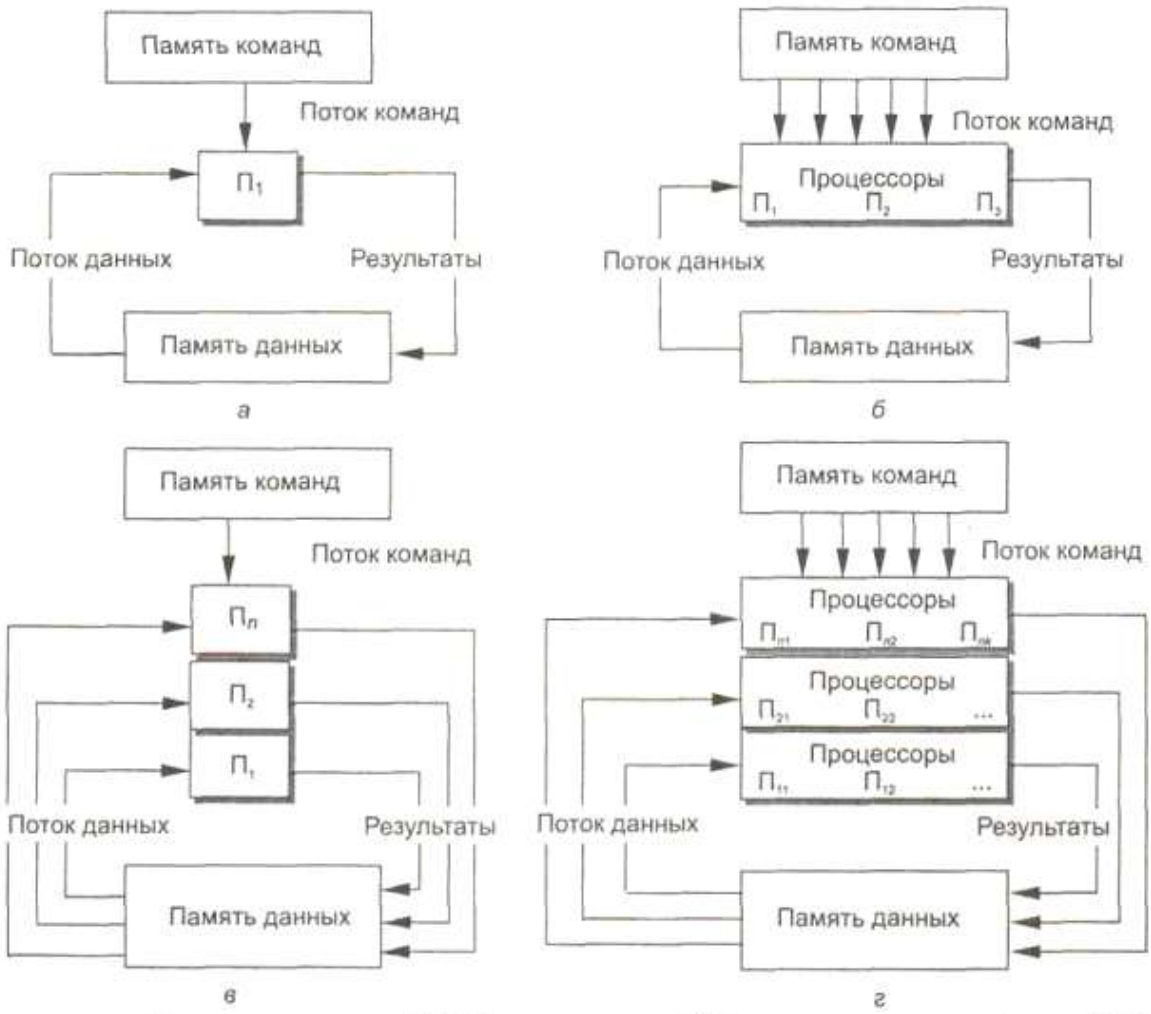


Рис.3.7 - Схема классификации вычислительных систем

В суперкомпьютере используются все три варианта архитектуры МПВС (рисунок 3.8):

- структура MIMD в классическом ее варианте;
- параллельно-конвейерная модификация, иначе MMISD, то есть многопроцессорная (Multiple) MISD-архитектура;
- *параллельно-векторная* модификация, иначе MSIMD, то есть многопроцессорная SIMD-архитектура.



Условные структуры МПВС: *а* — структура SISD — однопроцессорной суперЭВМ; *б* — структура MISD — конвейерной (магистральной) суперЭВМ; *в* — структура SIMD — векторной суперЭВМ; *г* — структура MIMD — матричной суперЭВМ

Рис. 3.8. - Условные структуры однопроцессорной (SISD) и названных многопроцессорных ВС

Оценка производительности

Напомним, что пиковая производительность определяется как максимальное число операций над числами с плавающей запятой, которое

компьютер может произвести в секунду (FLoating point OPerations per Second — FLOPS).

Для многопроцессорных и многоядерных систем, к которым относятся вычислительные кластеры, пиковая производительность считается как

$$FLOPSpeak = P*N*Hz*C,$$

где P — число процессоров; N — число вычислительных ядер на процессор; Hz — тактовая частота процессора; C — максимальное число операций над числами с плавающей запятой, которое процессор может выполнить за один такт.

Как уже говорилось ранее, пиковая производительность не совпадает с реальной.

$$FLOPSpeak \neq FLOPSreal$$

Реальная производительность, которую может показать вычислительная система при решении задач, зависит от:

- конкретной решаемой задачи;
- алгоритма решения задачи;
- быстродействия оперативной памяти;
- быстродействия подсистемы ввода-вывода;
- и целого ряда других факторов.

При этом, сам по себе FLOPS — плохая мера производительности:

- неоднозначным является само определение. «Операцией с плавающей запятой» может быть как сложение/вычитание, так и более сложные умножение/деление;
- существенную роль в вычислениях играет разрядность операндов, которая не оговаривается;
- величина FLOPS подвержена влиянию факторов, напрямую не связанных с производительностью вычислительного модуля: пропускная способность каналов связи с окружением процессора, производительность основной памяти и синхронность работы кэш-памяти разных уровней.

Как следствие, результаты, полученные на одном и том же компьютере при помощи разных программ, могут существенным образом отличаться, более того, с каждым новым испытанием разные результаты можно получить при

использовании одного алгоритма.

Чтобы устранить хотя бы часть недостатков такой оценки, как «пиковая производительность», и сделать сравнение суперкомпьютеров более объективным, был разработан тест LINPACK. Этот тест измеряет производительность, которую показывает компьютер при выполнении операций над числами с плавающей запятой. Тест измеряет насколько быстро компьютер решает случайную систему линейных уравнений вида Ax = b размерности N*N. Решение ищется методом Гаусса и требует $\frac{2}{3}N^3+2N^2$ операций над числами с плавающей запятой.

Для высокопроизводительных систем программный пакет High-Performance LINPACK Benchmark используется для ранжирования в списке TOP500 самых быстрых в мире компьютеров. Тест запускается несколько раз, чтобы найти размерность Nmax, при которой достигается наилучшая производительность (см. рисунок 3.8).

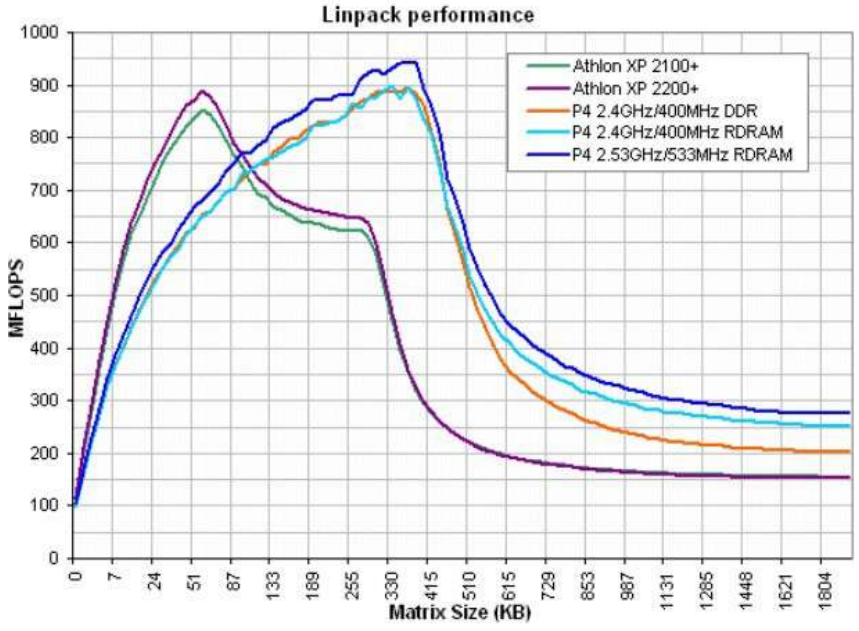


Рис. 3.8 - Пример зависимости демонстрируемой производительности в зависимости от размера системы линейных уравнений

При небольших размерностях, демонстрируемая компьютером производительность растет с увеличением N, однако вскоре выходит на свой максимум, после чего производительность снижается. Это может быть связано с ограничениями кэш-памяти, оперативной памяти (данные приходится подгружать с более медленного носителя), а также размерами векторного вычислителя.

Несмотря на недостатки, результаты производительности по LINPACK также измеряют во FLOPS. Причины популярности метрики FLOPS на основе LINPACK следующие:

- FLOPS — это абсолютная величина, с которой удобно сравнивать;
- многие задачи инженерной и научной практики в конечном итоге сводятся к решению систем линейных алгебраических уравнений;
- большинство современных суперкомпьютеров построены по классической архитектуре с использованием стандартных процессоров, что позволяет использовать единую метрику производительности.

Вопросы к главе 3.

1.Определение высокопроизводительных вычислений и фундаментальные и прикладные задачи

2.Инструменты высокопроизводительных вычислений. Классификация компьютеров

3.Что такое суперкомпьютер. Проблемы создания

4.Общая схема современного компьютера. Суперкомпьютеры и особенности их архитектуры

5.Высокопараллельные многопроцессорные вычислительные системы (МПВС). Классификация вычислительных систем. Оценка производительности

Глава 4. Технологии организации высокопроизводительных вычислений

Выделяют три основных ***технологии организации высокопроизводительных вычислений***:

1. ***вычислительный кластер*** — это группа вычислителей, объединенных высокоскоростными каналами связи и представляющая с точки зрения пользователя единый аппаратный ресурс;
2. ***грид-система*** — это группа географически распределенных вычислителей, объединенных низкоскоростными каналами связи и представляющая с точки зрения пользователя единый аппаратный ресурс;
3. ***«облачные вычисления» (cloud computing)*** — концепция обеспечения повсеместного и удобного сетевого доступа по требованию к общему пулу настраиваемых вычислительных ресурсов (например, сетям передачи данных, ресурсам процессоров и оперативной памяти, устройствам хранения данных, приложениям и сервисам — как вместе, так и по отдельности), которые могут быть оперативно предоставлены и освобождены по требованию с минимальными эксплуатационными затратами.

4.1. Технология организации высокопроизводительных вычислений на основе вычислительных кластеров

4.4.1. Что такое кластер

Развитие современных суперкомпьютерных технологий связано, прежде всего, с вычислительными кластерами. ***Кластер*** — группа вычислительных машин, объединенных высокоскоростными каналами связи, которые функционируют как один узел обработки информации (Определение DEC) и представляют с точки зрения пользователя единый аппаратный ресурс. Понятие "единый ресурс" означает наличие программного обеспечения, дающего возможность пользователям, администраторам и прикладным программам считать, что имеется только одна сущность, с которой они работают, - кластер. Например, система пакетной обработки кластера позволяет послать задание на обработку кластеру, а не какому-нибудь отдельному компьютеру. Более сложным примером являются системы баз данных. Практически у всех производителей систем баз данных имеются версии, работающие в параллельном режиме на нескольких машинах кластера. В результате приложения, использующие базу данных, не должны заботиться о том, где

выполняется их работа. СУБД отвечает за синхронизацию параллельно выполняемых действий и поддержание целостности базы данных. ***Компьютеры, образующие кластер,*** — так называемые узлы кластера — всегда относительно независимы, что допускает остановку или выключение любого из них для проведения профилактических работ или установки дополнительного оборудования без нарушения работоспособности всего кластера. В качестве вычислительных узлов в кластере обычно используются однопроцессорные персональные компьютеры, двух- или четырехпроцессорные SMP- серверы. Каждый узел работает под управлением своей копии операционной системы, в качестве которой чаще всего используются стандартные операционные системы: Linux, NT, Solaris и т.п. Состав и мощность узлов может меняться даже в рамках одного кластера, давая возможность создавать неоднородные системы. Выбор конкретной коммуникационной среды определяется многими факторами: особенностями класса решаемых задач, необходимостью последующего расширения кластера и т.п. Возможно включение в конфигурацию специализированных компьютеров, например, файл-сервера, и, как правило, предоставлена возможность удаленного доступа на кластер через Internet. Из определения архитектуры кластерных систем следует, что она включает в себя очень широкий спектр систем.

Как правило, ***кластер*** представляет собой группу однородных вычислительных узлов, соединенных между собой высокоскоростным каналом связи.

Кластер — это модульная многопроцессорная система, созданная на базе стандартных вычислительных узлов, соединенных высокоскоростной коммуникационной средой. Сейчас слова «кластер» и «суперкомпьютер» в значительной степени синонимы, но прежде чем об этом стало можно с уверенностью говорить, аппаратные средства прошли длительный цикл эволюции. В течение первых 30 лет с момента появления компьютеров, вплоть до середины 1980-х гг., под «суперкомпьютерными» технологиями понимали исключительно производство специализированных особо мощных процессоров. Однако появление однокристального микропроцессора практически стерло разницу между «массовыми» и «особо мощными» процессорами, и с этого момента единственным способом создания суперкомпьютера стал путь объединения процессоров для параллельного решения одной задачи. Привлекательной чертой кластерных технологий является то, что они позволяют для достижения необходимой производительности объединять в единые вычислительные системы

компьютеры самого разного типа, начиная от персональных компьютеров и заканчивая мощными суперкомпьютерами.

Разработчики архитектур кластерных систем преследовали различные цели при их создании. Первой была фирма Digital Equipment с кластерами VAX/VMS. Целью создания этой машины было повышение надежности работы системы, обеспечение высокой готовности и отказоустойчивости. В настоящее время существует множество аналогичных по архитектуре систем от других производителей. Другой целью создания кластерных систем является создание дешевых высокопроизводительных параллельных вычислительных систем. Один из первых проектов, давший имя целому классу параллельных систем – кластер Beowulf – возник в центре NASA Goddard Space Flight Center для поддержки необходимыми вычислительными ресурсами проекта Earth and Space Sciences. Проект Beowulf начался летом 1994 года, и вскоре был собран 16-процессорный кластер на процессорах Intel 486DX4/100 МГц. На каждом узле было установлено по 16 Мбайт оперативной памяти и по 3 сетевых Ethernet-адаптера. Эта система оказалась очень удачной по отношению цена/производительность, поэтому такую архитектуру стали развивать и широко использовать в других научных организациях и институтах. Для каждого класса кластеров характерны свои особенности архитекуры и применяемые аппаратные средства. В среднем отечественные суперкомпьютеры пока еще сильно уступают западным по производительности: машины, используемые для научных исследований, в 15 раз, вычислительные ресурсы финансовых компаний — в 10 раз, промышленные суперкомпьютеры — в 9 раз. Вывод : Кластер — это сложный программно-аппаратный комплекс, состоящий из вычислительных узлов на базе стандартных процессоров, соединенных высокоскоростной системной сетью, а также, как правило, вспомогательной и сервисной сетями. Различают четыре типа кластерных систем: вычислительные кластеры, кластеры баз данных, отказоустойчивые кластеры и кластеры для распределения загрузки. Сфера применения кластерных систем сейчас нисколько не уже, чем суперкомпьютеров с другой архитектурой: они не менее успешно справляются с задачей моделирования самых разных процессов и явлений. Суперкомпьютерное моделирование может во много раз удешевить и ускорить вывод на рынок новых продуктов, а также улучшить их качество.

Источник: http://5fan.ru/wievjob.php?id=77123

Высокопроизводительные вычисления на основе кластеров стали появляться сравнительно недавно. До конца 80-х практически все суперкомпьютеры представляли собой большой массив соединенных между собой процессоров. Подобные разработки чаще всего были уникальными и имели огромную стоимость не только приобретения, но и поддержки. Поэтому в 90-х годах все более широкое распространение стали получать кластерные системы, которые в качестве основы используют недорогие однотипные вычислительные узлы [3].
К основным преимуществам кластеров можно отнести относительно низкую стоимость вычислительных узлов и легкую расширяемость. В качестве основного недостатка выступает сложность отладки параллельных программ для систем с разделенной памятью.
Как уже было сказано выше, кластеры представляют собой системы с разделенной памятью, поэтому типичное параллельное приложение представляет собой совокупность нескольких процессов, исполняемых на разных вычислительных узлах и взаимодействующих по сети. В принципе, разработчик может полностью взять на себя программирование распределенного приложения и самостоятельно реализовать общение по сети на основе сокетов, например. Однако в настоящее время существует довольно большое число технологий, упрощающих создание параллельных приложений для кластеров (например, MPI). Эти технологии существуют уже достаточно продолжительное время, за которое они доказали свою состоятельность и легли в основу огромного числа параллельных приложений [1].
Кластеры стали фактическим стандартом в области высокопроизводительных вычислений, и можно с большой долей уверенности сказать, что этот подход будет актуален всегда: сколь бы совершенен не был один компьютер, кластер из узлов такого типа справится с любой задачей гораздо быстрее.

4.4.2. Два подхода к созданию кластеров

Кластерные системы как одна из разновидностей суперкомпьютеров получили свое становление и развитие в 1989 году с выходом однокристального микропроцессора Intel i860 [1]. С этого времени развитие суперкомпьютерной отрасли идет по двум основным направлениям:

- **Магистральное направление** предполагает использование микросхем массового выпуска для создания специализированных микропроцессорных систем. К этим системам относятся SMP-системы (Symmetric Multi Processing) – суперкомпьютеры с симметрично адресуемой общей памятью, MPP-системы (Massively Parallel Processing) – массивно-параллельная архитектура, а также NUMA-системы и некоторые другие.

- **Кластерное направление** логически во многом похоже на MPP-системы, разница заключается лишь в том, что для MPP-систем использовались специализированные и дорогие вычислители, а в кластеры по локальной сети объединялись обычные массово производимые персональные ЭВМ.

Кластерное направление, хоть и развивалось параллельно с магистральным, но до середины 90-х годов не могло составить ему конкуренцию по производительности. Во второй половине 90-х годов произошел прорыв в области аппаратуры массового выпуска, связанный с появлением в персональных компьютерах шины PCI [2] и переходом на дешевую и быструю коммутируемую сеть Fast Ethernet. Таким образом, кластеры рабочих станций по производительности сравнялись с существующими на тот момент MPP-системами, но имели намного меньшую цену реализации. Вследствие этого MPP-системы из специализированных вычислителей практически исчезли и перестали развиваться, отдав эту нишу кластерным системам [3].

*Удобство построения **кластерных ВС** заключается в том, что можно гибко регулировать необходимую производительность системы, подключая к кластеру с помощью специальных аппаратных и программных интерфейсов обычные серийные серверы до тех пор, пока не будет получен суперкомпьютер требуемой мощности. Кластеризация позволяет манипулировать группой серверов как одной системой, упрощая управление и повышая надежность*.

Важной особенностью кластеров является обеспечение доступа любого сервера к любому блоку как оперативной, так и дисковой памяти. Эта проблема успешно решается, например, объединением систем SMP-архитектуры на базе автономных серверов для организации общего поля оперативной памяти и использованием дисковых систем RAID для памяти внешней (SMP — Shared Memory multiprocessing, технология мультипроцессирования с разделением памяти).

Для создания кластеров обычно используются либо простые однопроцессорные персональные компьютеры, либо двух- или четырех-процессорные SMP-серверы. При этом не накладывается никаких ограничений на состав и архитектуру узлов. Каждый из узлов может функционировать под управлением своей собственной операционной системы. Чаще всего используются стандартные ОС: Linux, FreeBSD, Solaris, Unix, Windows NT. В тех случаях, когда узлы кластера неоднородны, то говорят о ***гетерогенных*** кластерах.

При создании кластеров можно выделить два подхода.

1. **Первый подход** применяется при создании небольших кластерных систем. В кластер объединяются полнофункциональные компьютеры, которые продолжают работать и как самостоятельные единицы, например, компьютеры учебного класса или рабочие станции лаборатории.
2. **Второй подход** применяется в тех случаях, когда целенаправленно создается мощный вычислительный ресурс. Тогда системные блоки компьютеров компактно размещаются в специальных стойках, а для управления системой и для запуска задач выделяется один или несколько полнофункциональных компьютеров, называемых хост-компьютерами. В этом случае нет необходимости снабжать компьютеры вычислительных узлов графическими картами, мониторами, дисковыми накопителями и другим периферийным оборудованием, что значительно удешевляет стоимость системы.

Кластерные системы могут использовать самые разные платформы и, как правило, классифицируются не по набору комплектующих, а по областям применения. Выделяют четыре типа кластерных систем: вычислительные кластеры, кластеры баз данных, отказоустойчивые кластеры и кластеры для распределения загрузки. Самая многочисленная группа — вычислительные кластеры. Она может быть разбита на подгруппы; правда, классификации внутри этой группы подлежат уже не собственно вычислительные машины, а готовые программно-аппаратные кластерные решения. Такие системы «под ключ» имеют предустановленное прикладное ПО, необходимое заказчику для решения его задач. Решения, оптимизированные для разных приложений, различаются подбором компонентов, обеспечивающим наиболее производительную работу именно этих приложений при наилучшем соотношении цена/качество. Кластеры баз данных появились недавно. Эти системы работают с параллельными версиями баз данных и используются в крупных организациях для работы CRM-и ERP-систем, а также транзакционных баз данных. Сегодня эти системы — серьезный конкурент традиционным серверам с общей памятью благодаря лучшему соотношению цена/производительность, масштабируемости и отказоустойчивости. Отказоустойчивые кластеры строят для того, чтобы наилучшим образом обеспечить надежность работы критически важных приложений. Работа приложения дублируется на разных узлах, и в случае ошибки на одном из них приложение продолжает работать или автоматически

перезапускается на другом. Такие кластеры не бывают большими, и пользователи часто строят их сами. Кластерные технологии также используются для распределения большого потока запросов по многим серверам. Такие решения часто применяются для поддержки Web-узлов с динамическим содержимым, постоянно обращающихся к базам данных, например, поисковых систем. В зависимости от размеров сервиса кластеры распределения загрузки могут иметь достаточно большое количество узлов. Работа кластерных систем обеспечивается четырьмя видами специализированных приложений, как то: операционные системы (как правило, Linux), средства коммуникации, средства разработки параллельных приложений и ПО для администрирования кластеров.

В литературе встречаются различные классификации кластеров:

1. По применимости в областях экономики [5] (промышленные, инженерные, для нефте-газодобывающей промышленности и т.д.) – такая классификация не отражает архитектурных особенностей систем и несет только прикладной характер.
2. По способу обеспечения отказоустойчивости [6] – классификация охватывает слишком узкую и специализированную область и определяет наличие резервирования оборудования и линий связи.
3. Достаточно полная классификация кластерных систем была дана Карпенко А.П. [7], но и она может быть дополнена некоторыми другими критериями.

Одной из важнейших компонент кластера является его коммутационная система, разновидностей которых в настоящее время множество: PCI Express, InfiniBand, Fibre Channel, семейство Ethernet, Myrinet, SCI и другие. Ни в одной из классификаций нет четких критериев по разделению кластеров по типу коммутационной системы.

4.4.3. Место кластеров в классификации вычислительных систем

Для определения положения **кластеров** внутри общей классификации архитектур вычислительных систем можно использовать таксономию **Флинна** [8, 9] как одну из самых первых (предложена в 1966 году) и широко известную и применяемую на начальных этапах классификации вычислительных систем. Флинн М. выделяет четыре класса архитектур, которые базируются на понятии потока и способов работы с ним: ОКОД (англ. «SISD» – одиночный поток команд одиночный поток данных), ОКМД (англ. «SIMD» - одиночных поток команд множественный поток данных), МКОД (англ. «MISD» - множественный поток команд одиночный поток данных), МКМД (англ. «MIMD» - множественный поток команд множественный поток данных). Кластерные системы относятся к классу MIMD-систем, поскольку этот класс предполагает, что в вычислительной системе есть несколько устройств обработки команд, объединенных в единый

комплекс и работающих каждое со своим потоком команд и данных (см. рис. 4.1) [10], что полностью описывает распределенную архитектуру кластера.

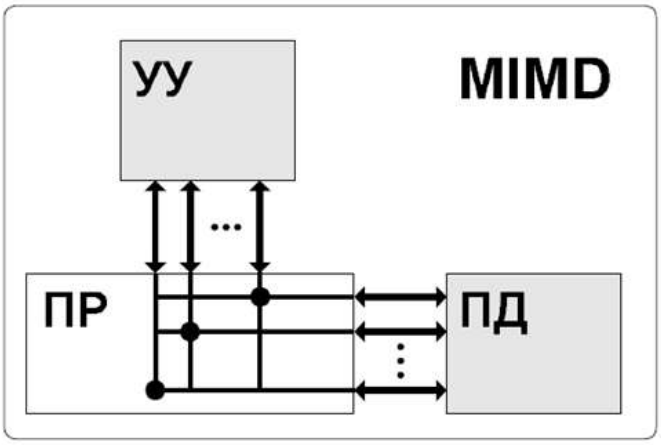


Рис. 4.1. Класс MIMD по классификации М. Флинна, где УУ – Устройство Управления, ПР – Процессорный элемент, ПД – память данных

Класс MIMD-систем в дальнейшем был конкретизирован **Р. Хокни**, который предложил свою классификацию (см. рис. 4.2) [11, 12].

Рис. 4.2 - Расширенная классификация MIMD-систем по Р. Хокни

Поскольку кластерные системы по определению представляют собой сетевую структуру, а считается, что все сетевые структуры имеют распределенную память, то дальнейшая классификация кластерных систем проводится в соответствии с топологией сети. Если архитектура кластера спланирована с использованием нескольких сетей с различной топологией, то она классифицируется как гибридная сетевая MIMD-система.

Таким образом, положение кластеров среди всего многообразия вычислительных систем изображено на рисунке 4.3.

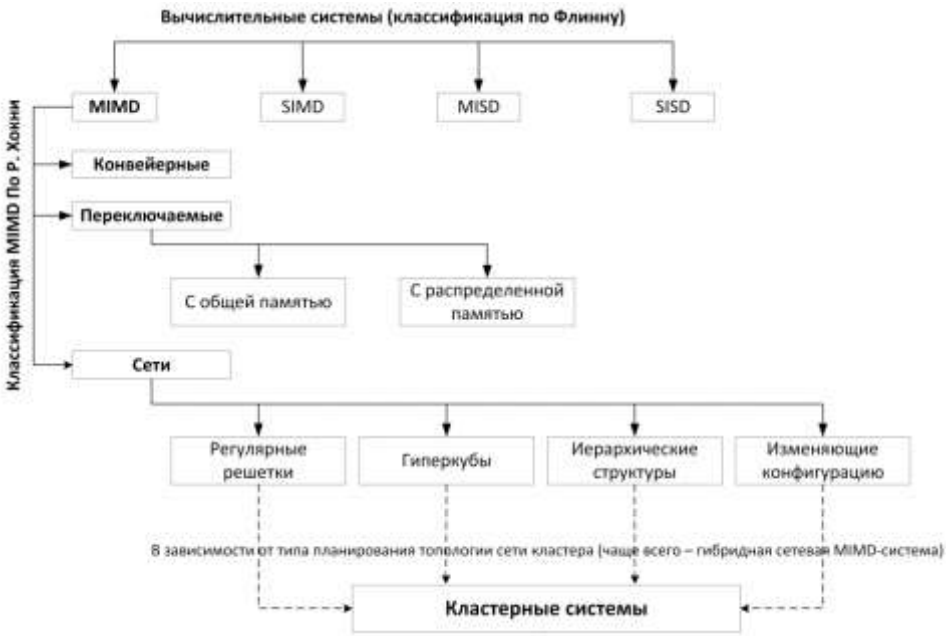


Рис. 4.3 - Положение кластеров в общей классификации вычислительных систем

4.4.5. Основные признаки классификации кластерных систем

В литературе выделяются следующие признаки (критерии) кластерных систем [13, 5, 6, 7]: по типу вычислительных устройств (ВУ), по типу используемых ОЗУ, по типу операционных систем (ОС), по стандартности комплектующих, по функциональной направленности, GRID-кластеры, по отраслям применения, по способу обеспечения отказоустойчивости.

Классификация по типу ВУ. По типу ВУ кластерные системы разделяются на гомогенные или гетерогенные, полнофункциональные или не полнофункциональные [13, 14].

Как и всякие MIMD-системы, вычислительные кластеры разделяются на однородные кластерные системы и гетерогенные кластерные системы.

Гомогенный кластер – кластер, в котором все ВУ имеют одинаковую конфигурацию, следовательно, производительность вычислителей в ВУ одинаковые.

Гетерогенный кластер – кластер, в котором ВУ имеют разную конфигурацию и, соответственно, производительность.

Обычно при проектировании кластерных систем используется подход с применением однородных ВУ. Однако достаточно часто при увеличении вычислительной мощности и, как следствие, аппаратной сложности кластера, используются процессоры, отличающиеся не только по производительности, но и

по архитектуре, от узловых процессоров кластера, заложенных при его разработке. Таким образом, постепенно однородный вычислительный кластер может стать неоднородным. Неоднородность создает следующие проблемы [7]:

- возрастает сложность в распределении задач между процессорами из-за различий в производительности;

- различие в архитектуре процессоров требует подготовки разных выполняемых файлов для разных узлов;

- из-за различий в представлении данных может потребоваться преобразование их форматов при передаче между узлами.

Узлы вычислительного кластера могут представлять собой **полнофункциональные** компьютеры, которые могут работать как самостоятельные единицы. Производительность такого кластера невысока по сравнению с не полнофункциональными [7].

При проектировании высокопроизводительных кластеров используют мощные однотипные вычислительные узлы, функционально не приспособленные для работы вне кластера **(не полнофункциональные)**. В составе кластера нет необходимости снабжать такие узлы дискретными графическими адаптерами, мониторами, дисковыми накопителями и другим периферийным оборудованием. Периферийное оборудование может быть установлено только на одном или немногих управляющих компьютерах. Использование не полнофункциональных узлов может выгодно сказаться на итоговой стоимости развертывания кластера [7].

Классификация по типу используемых ОЗУ: с распределенной общей памятью (Distributed shared memory, DSM) и с распределенной памятью (Distributed memory, DM) [13].

Если в вычислительном кластере аппаратно или аппаратно-программно реализована распределенная общая память DSM, позволяющая выполняющимся на разных узлах программам взаимодействовать через общие переменные, то такой кластер называется **DSM-кластером**.

Кластер DSM представляет собой виртуальное адресное пространство, разделяемое всеми узлами (процессорами) распределенной системы. Программы получают доступ к данным в DSM примерно так же, как они работают с данными в виртуальной памяти традиционных ЭВМ. В системах с DSM данные перемещаются между локальными памятями разных компьютеров аналогично тому, как они перемещаются между оперативной и внешней памятью одного компьютера. Объем суммарной физической памяти всех узлов может быть очень

большим. Вся память становится доступна приложению без издержек, связанных в традиционных системах с дисковыми обменами. Это достоинство становится все весомее в связи с тем, что скорости процессоров растут быстрее скоростей памяти и в то же время появляются очень быстрые коммуникации. DSM-системы могут наращиваться практически беспредельно в отличие от систем с разделяемой памятью, т.е. являются масштабируемыми.

Если распределенная общая память отсутствует и программы взаимодействуют посредством передачи сообщений, то кластер называется **DM-кластером**. В варианте с DM-памятью каждому из процессоров выделяется собственная память. Процессоры объединяются в сеть и могут при необходимости обмениваться данными, хранящимися в их памяти, передавая друг другу так называемые сообщения. Такой вид вычислительных систем еще называют слабо связанными (loosely coupled systems) [14].

Классификация по типу используемой ОС: Unix-кластеры (включая Linux) и Windows-кластеры [13].

Если обратиться к списку Top500, то подавляющее большинство суперкомпьютеров (более 95%) работают под управлением операционных систем семейства Linux [15]. Выбор операционной системы с открытым исходным кодом позволяет разработчикам тонко ее настроить: оптимизировать ядро под задачи и процессы, на решение которых направлен кластер, интегрировать с прикладным программным обеспечением на базовом программном уровне. Именно поэтому, Unix- и Linux-подобные операционные системы завоевали большую популярность. Кроме этого, часть специализированного программного обеспечения для вычислительных кластеров работает только на UNIX-платформах. Windows-кластеры практически полностью вытеснены из списка Top500, они используются в вычислительных системах более низкого уровня.

Классификация по стандартности комплектующих: стандартные или эксклюзивные [7].

Вычислительные кластеры из **стандартных комплектующих** имеют низкие цены и простое обслуживание. Широкое распространение кластерные технологии получили как средство создания именно относительно дешевых систем суперкомпьютерного класса из составных частей массового производства.

Кластеры из **эксклюзивных** (нестандартных) комплектующих, позволяют получить очень высокую производительность, но являются более дорогими. Примером таких кластеров могут служить систолические структуры (**систолы**).

Систолическая архитектура вычислительного устройства представляет собой SIMD-систему, т.е. конвейер, на каждой ступени которого происходит обработка

данных и промежуточный результат без записи в память передается на следующую ступень конвейера для обработки. Систолы являются эффективным решением для выполнения узкоспециализированных задач, и, как правило, проектируются индивидуально под каждую задачу [16].

Классификация по функциональной направленности: высокоскоростные (High Performance, HP-кластер) и высокой готовности (High Availability, HA-кластер) [7].

Высокоскоростные кластеры. Производительность узлов и скорость связи между ними определяют быстродействие High Performance кластерной системы. Причем влияние скоростных параметров как узлов, так и связей между ними на общую производительность системы зависит от характера выполняемых кластером задач. Требуется соблюдать баланс между скоростью межпроцессорного обмена и их вычислительной мощностью. Чем меньше взаимодействуют части параллельной задачи между собою, тем большие требования предъявляются к производительности вычислительных узлов, и, соответственно, чем больше взаимодействие, тем выше должна быть скорость межпроцессорного обмена. Также характер выполняемых задач накладывает определенные требования и на их параллельное программирование [6].

Кластеры высокой готовности (высокой надежности). Кластеры High Availability предоставляют высокую отказоустойчивость при достаточно низкой стоимости. Вычислительная система будет обладать высокими показателями готовности, когда она будет удовлетворять следующим условиям:

- вычислительные узлы и другие компоненты такой системы были максимально надежными;

- система должна быть отказоустойчивой, как правило это реализуется за счет аппаратной избыточности;

- система должна поддерживать технологию Hot Swap, то есть возможность «горячей» замены.

Соблюдение вышеперечисленных условий позволит операционной системе кластера при отказе одного из вычислительных узлов автоматически перераспределить задачи между другими исправными узлами вычислительного кластера.

Выделяется отдельный класс вычислительных кластеров - **вычислительные сети GRID.** GRID-вычислители (от англ. *grid* - решетка) представляют собой совокупность разного вида вычислителей: целых кластеров,

отдельных ЭВМ (как многопроцессорных, так и однопроцессорных), которые могут иметь любую территориальную удаленность [7].
К основным свойствам GRID можно отнести [17]:
- координированное использования общих ресурсов;
- отсутствие централизованного управления этими ресурсами;
- использование стандартных, открытых, универсальных протоколов и интерфейсов;
- высокое качество обслуживания (низкая латентность, надежность, аппаратная избыточность и т.д.).

4.4.6. Систематизация известных подходов к классификации КС

В приведенных выше признаках классификации кластерных систем отсутствуют признаки, которые позволяли бы разделять кластеры по коммутационным системам. Косвенно разделение по коммутационным структурам упоминается в литературе [7] при определении *высокопроизводительного кластера*. Но в этом случае производительность кластера рассматривается с комплексной точки зрения, включающей скорость взаимодействия между подзадачами, производительность вычислительных узлов и специфику параллельного программирования конкретной выполняемой задачи.

На основании вышеизложенного предлагается расширить существующие классификации кластеров дополнительным признаком – «по типу коммутационных систем»

Таким образом, с точки зрения классификации по типу используемых коммутационных системможно выделить кластер со специализированной (высокопроизводительной) КС или с КС общего назначения.

Производительность кластера, как и всякой системы с распределенной памятью, сильно зависит от производительности коммуникационной среды. Влияние производительности коммуникационной среды на общую производительность кластерной системы зависит от характера выполняемой задачи. Если задача требует частого обмена данными между подзадачами, которые решаются на разных узлах вычислительного кластера, то быстродействию коммуникационной среды следует уделить максимум внимания и использовать специализированные (высокопроизводительные) протоколы обмена данными, такие как InfiniBand, PCI Express, Fibre Channel и другие.

Соответственно, чем меньше взаимодействуют части задачи между собой, тем меньше требований предъявляется к быстродействию коммуникационной среды. В этом случае для межпроцессорного обмена могут быть использованы протоколы общего назначения, такие как Fast Ethernet, Gigabit Ethernet или другие.

Если при построении вычислительных кластеров используют достаточно дешевые коммуникационные среды, то такие кластеры обеспечивают производительность на один-два порядка более низкую, чем производительность коммуникационных сред суперкомпьютеров [7].

В большинстве современных кластеров используются несколько типов коммутационных сетей, несущих разную функциональную нагрузку. Для низколатентного обмена сообщениями или быстрой доставки большого массива данных применяются высокопроизводительные протоколы, в то время как для управления и мониторинга кластера целесообразно использовать более дешевые протоколы, основанные на технологии Ethernet.

В результате исследования известных классификаций кластеров и обобщения всех приведенных критериев, с выделением особо значимых и с учетом введения нового, в работе предложена общая схема, отражающая как положение кластеров среди всех вычислительных систем вообще, так и подробную классификацию внутри самих кластерных систем (рисунке 4.4).

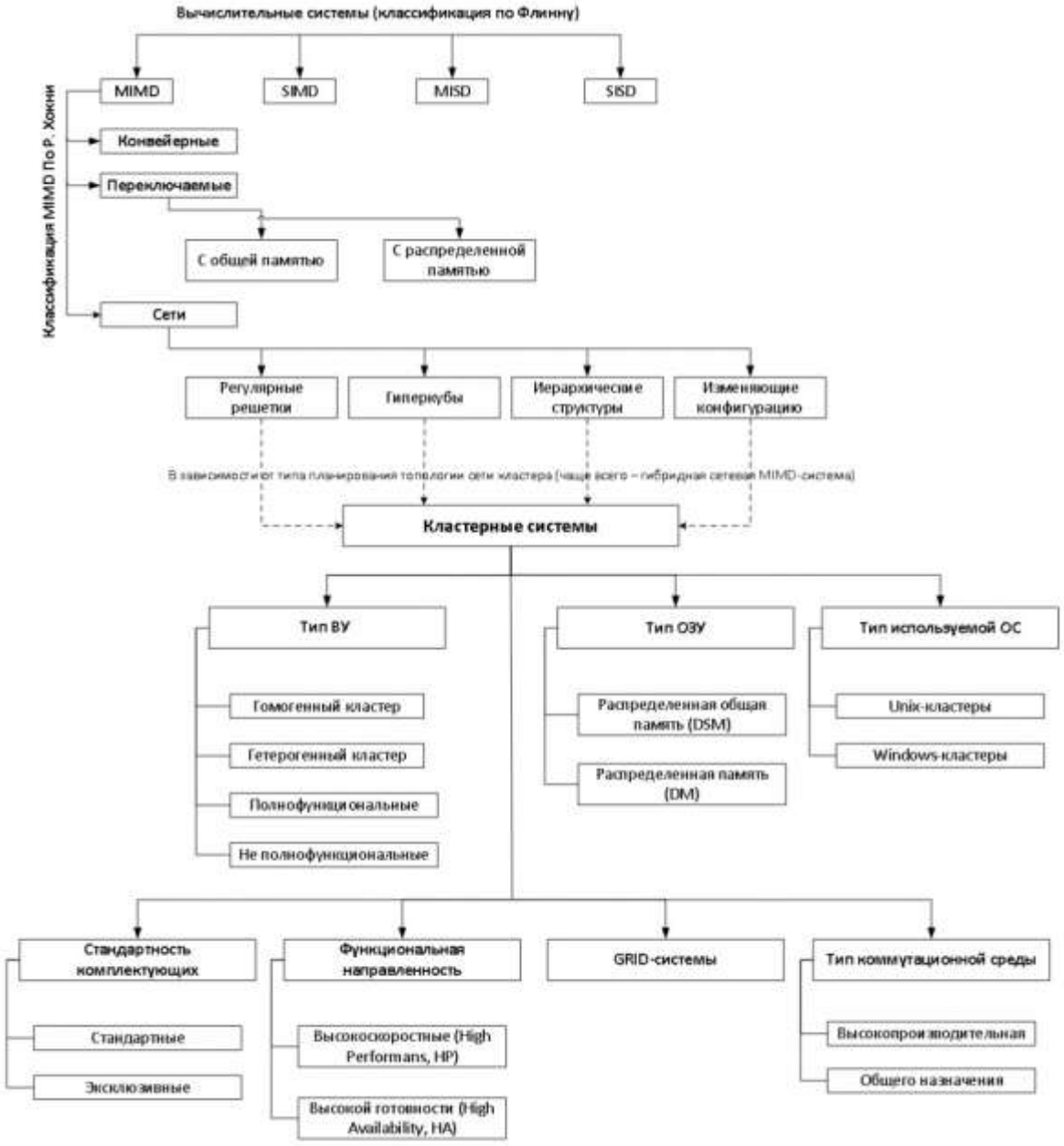


Рис. 4.4 - Обобщенная многокритериальная классификация кластерных систем

Согласно концепции определению кластерной системы, структура типичного кластера может быть представлена на рисунке 4.5.

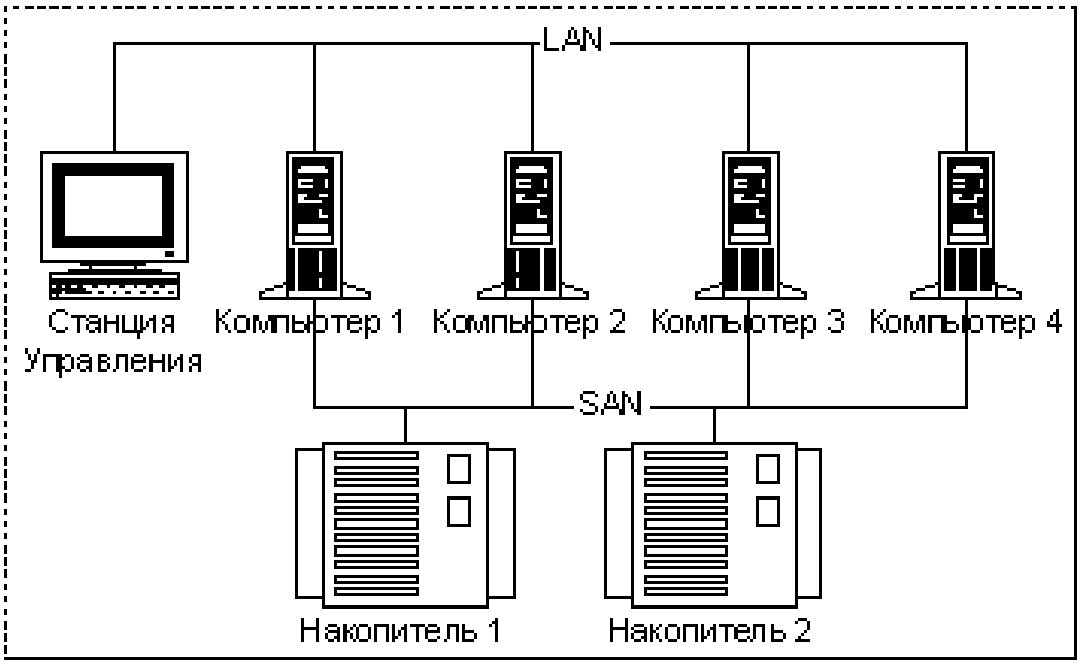


Рис. 4.5 – Типовая кластерная система

LAN — Local Area Network, локальная сеть

SAN — Storage Area Network, сеть хранения данных

Ускорение вычислений на кластере происходит благодаря ***параллельному выполнению на разных вычислителях независимых участков программы.*** В процессе выполнения программы по мере необходимости результаты вычислений объединяются, передаются на другие вычислители и используются в дальнейших вычислениях.

В общем плане под параллельными вычислениями понимаются процессы обработки данных, в которых одновременно могут выполняться нескольких машинных операций [22 http://www.intuit.ru [Электронный ресурс]. Режим доступа свободный]. Достижение параллелизма возможно только при выполнимости следующих требований к архитектурным принципам построения вычислительной системы:

- независимость функционирования отдельных устройств ЭВМ - данное требование относится в равной степени ко всем основным компонентам вычислительной системы - к устройствам ввода-вывода, к обрабатывающим процессорам и к устройствам памяти;
- избыточность элементов вычислительной системы - организация избыточности может осуществляться в следующих основных формах:
- использование специализированных устройств таких, например, как отдельных процессоров для целочисленной и вещественной арифметики, устройств многоуровневой памяти (регистры, кэш);

- дублирование устройств ЭВМ путем использования, например, нескольких однотипных обрабатывающих процессоров или нескольких устройств оперативной памяти.

Дополнительной формой обеспечения параллелизма может служить конвейерная реализация обрабатывающих устройств, при которой выполнение операций в устройствах представляется в виде исполнения последовательности составляющих операцию подкоманд; как результат, при вычислениях на таких устройствах могут находиться на разных стадиях обработки одновременно несколько различных элементов данных [2 http://www.intuit.ru [Электронный ресурс]. Режим доступа свободный].

При рассмотрении проблемы организации параллельных вычислений следует различать следующие возможные режимы выполнения независимых частей программы:

- многозадачный режим (режим разделения времени), при котором для выполнения процессов используется единственный процессор; данный режим является псевдопараллельным, когда активным (исполняемым) может быть один единственный процесс, а все остальные процессы находятся в состоянии ожидания своей очереди на использование процессора. Использование режима разделения времени может повысить эффективность организации вычислений (например, если один из процессов не может выполняться из-за ожидания вводимых данных). Процессор может быть задействован для готового к исполнению процесса, кроме того, в данном режиме проявляются многие эффекты параллельных вычислений (необходимость взаимоисключения и синхронизации процессов и др.) и, как результат, этот режим может быть использован при начальной подготовке параллельных программ;
- параллельное выполнение, когда в один и тот же момент времени может выполняться несколько команд обработки данных; данный режим вычислений может быть обеспечен не только при наличии нескольких процессоров, но реализуем и при помощи конвейерных и векторных обрабатывающих устройств;
- распределенные вычисления; данный термин обычно используют для указания параллельной обработки данных, при которой используется несколько обрабатывающих устройств, достаточно удаленных друг от друга и в которых передача данных по линиям связи приводит к существенным временным задержкам. В результате эффективная обработка данных при таком способе организации вычислений возможна только для параллельных алгоритмов с низкой интенсивностью потоков

межпроцессорных передач данных; перечисленные условия является характерными, например, при организации вычислений в многомашинных вычислительных комплексах, образуемых объединением нескольких отдельных ЭВМ с помощью каналов связи локальных или глобальных информационных сетей;

4.4.7. Принципы построения и использования параллельных вычислительных систем

Выше было показано, что высокопроизводительные вычисления нашли широкое применение во многих областях науки – математике, физике, механике, метеорологии, экономике, медицине. Возможность объединять множество компьютеров, находящихся в территориально распределенных точках, в единый вычислительный ресурс многократно усиливает мощность и быстродействие объединенного вычислительного комплекса. Однако этого мало, во многих случаях эффективность зачастую зависит от алгоритма вычислительного процесса: если начать вычисления, запрограммированные по обычному порядку последовательных вычислений, то вычисления не будут ускорены, так как незагруженные ресурсы будут просто простаивать. Следовательно, в данном случае нужен особый алгоритм, который позволит задействовать все вычислительные ресурсы и в несколько раз повысить их эффективность.

Одно из решений данной проблемы связано с реализацией концепции ***параллелизма на уровне потоков*** (Thread Level Parallelism – TLP). Если запущенное на исполнение задание не в состоянии загрузить работой все функциональные устройства, то можно разрешить процессору выполнять более чем одну задачу, создавая тем самым второй поток (тред), чтобы он загрузил простаивающие устройства. Здесь есть аналогия с ***многозадачной операционной системой*** (ОС): чтобы процессор не простаивал, когда задача оказывается в состоянии ожидания (например, завершения ввода-вывода), ОС переключается на выполнение другой задачи. Наиболее эффективной на сегодня стала архитектура с одновременным выполнением потоков (Simultaneous Multi-Threading – SMT). В такой ситуации на каждом новом такте на выполнение в какое-либо исполнительное устройство может направляться команда любого потока – в зависимости от вычислительного алгоритма [5].

В общем случае распределенные вычисления (Distributed Computing) – это способ решения трудоемких вычислительных задач с большим объемом однотипных вычислений или связанных с колоссальными количествами обрабатываемых данных. Такие вычисления проводятся с использованием многоядерных или многопроцессорных систем на одном компьютере или на двух и более обычных компьютерах, объединенных в сеть.

Распределенные вычисления являются частным случаем параллельных вычислений, поэтому необходимо, чтобы решаемая задача была сегментирована, а конкретно, разделена на подзадачи, которые могут вычисляться параллельно. При этом для распределенных вычислений приходится также учитывать возможное различие в вычислительных ресурсах, которые будут доступны для расчета различных подзадач. Однако не всякую задачу можно разделить, ускорив ее решение с помощью распределенных вычислений.

4.4.7.1. Принципы формирования параллелизма

Параллельные вычисления – это такие процессы обработки данных, когда несколько однотипных или различных операций могут выполняться одновременно на одном вычислительном устройстве или на нескольких компьютерах, объединенных в локальную сеть. Реализовать параллельные вычисления можно только при выполнении определенных **принципов построения архитектуры вычислительной среды** [6]:

• независимость функционирования отдельных компьютерных устройств (устройства ввода-вывода, память, процессоры, звуковые и видео-адаптеры и пр.);

• наличие избыточности элементов вычислительной системы (за счет организации специализированных устройств, например формирование многоуровневой памяти или дублирование функций на базе однотипных стандартных устройств).

При организации параллельных вычислений возможно несколько режимов исполнения независимых частей программы.

Многозадачный режим (режим с разделением времени). В данном случае для выполнения нескольких процессов используется один процессор. Такой режим можно назвать квазипараллельным, так как при полной загрузке процессора одной задачей все остальные дожидаются своей очереди. Разделение времени выполнения процессов на различных устройствах может повысить эффективность. Например, если один вычислительный процесс не работает из-за ожидания ввода или вывода данных, то можно запустить другой процесс на другом устройстве. Многозадачный режим можно использовать при начальной подготовке параллельных вычислений.

Параллельное выполнение. Это такой режим, когда одновременно исполняется несколько операций обработки данных. Подобный режим может быть обеспечен не только на нескольких процессорах, но и на одном процессоре при помощи эффективных конвейерных и векторных устройств.

Распределенные вычисления. Этот термин используется, когда говорят о параллельной обработке данных, для которой применяется несколько обрабатывающих устройств, территориально удаленных друг от друга настолько, что передача данных по линиям связи приводит к необходимости учитывать временные задержки сигналов в линиях. Эффективная работа с данными в этом случае возможна только для параллельных алгоритмов с невысокой интенсивностью межпроцессорного (межкомпьютерного) обмена данными и результатами вычислений. Эти условия характерны, в частности, для формирования многомашинных комплексов в региональных сетях или посредством Интернета.

4.4.7.2. Разработка параллельных методов вычислений

Разработка алгоритмов параллельных вычислений является сложной научно-технической задачей, так как помимо чисто математических вопросов построения алгоритмов и исследования их сходимости следует обязательно учитывать архитектуру вычислительного устройства, для которого разрабатывается тот или иной алгоритм. Естественно, что есть некоторые базовые требования к разработке алгоритмов.

Определение эффективных способов организации параллельных вычислений в общем случае сводится к приведенным ниже обязательным моментам (рисунок 4.6):

• требуется выполнить анализ имеющихся вычислительных схем и разделить их на части (подзадачи), которые могут быть в значительной степени реализованы независимо друг от друга;

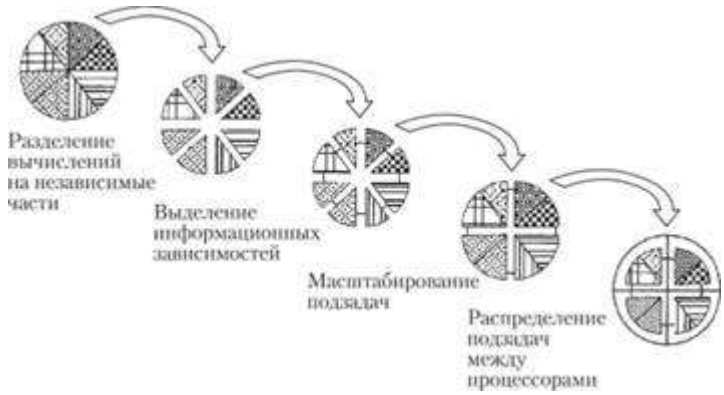


Рис. 4.6 - Общая схема разработки параллельного алгоритма

• надо выделить для сформированного набора подзадач информационные взаимодействия, которые должны быть задействованы в ходе вычислительных действий;

• следует определить необходимую для решения полной задачи вычислительную систему и произвести распределение набора подзадач между процессорами системы.

Очевидно, что распределять подзадачи нужно так, чтобы объем вычислений для каждого загружаемого процессора был примерно одинаковым (балансировка нагрузки). Кроме этого, распределение подзадач должно быть таким, чтобы количество коммуникационных взаимодействий было минимальным. Нужно оценить эффективность разрабатываемого алгоритма. Для этой цели следует определить значения таких, например, показателей качества спроектированных параллельных вычислений, как ускорение вычислений, степень сбалансированности загрузки процессоров, масштабируемость наборов подзадач и т.д. После этого общую схему составленного алгоритма подвергают детальной программной проработке, чтобы "разместить" программы решения подзадач по процессорам в соответствии с вы – бранной схемой распараллеливания. Далее программы запускают для выполнения, при этом выполнение программы для каждого массива подзадач называется **процессом**. Для реализации коммуникационных взаимодействий процессы должны иметь ***каналы передачи сообщений*** и ***средства обмена данными.***

Как правило, каждый процессор выделяется для решения определенной подзадачи, однако если подзадач больше, чем процессоров, то на один процессор следует запускать несколько подзадач, соблюдая по возможности сбалансированность общей загрузки. Такая ситуация возникает чаще всего на начальном этапе распараллеливания, когда работоспособность модельного алгоритма проверяют на одном процессоре в режиме разделения времени.

Распределение подзадач между процессорами является завершающим этапом разработки параллельного метода вычислений. Управление распределением нагрузки (пользовательская балансировка) возможна только для вычислительных систем с распределенной памятью; для многопроцессорных систем с общей памятью распределение нагрузки обычно производится автоматически.

Рассмотренный выше метод построения алгоритма распараллеливания ориентирован в значительной степени на вычислительные системы с распределенной памятью, когда необходимые информационные взаимодействия реализуются по выделенным каналам связи между процессорами. Однако этот метод с не меньшим успехом можно применять и для разработки параллельных алгоритмов для систем с общей памятью, в этом случае информационные взаимодействия осуществляются операциями доступа к общим ресурсам.

4.4.8. Основные достоинства и проблемы кластерных суперкомпьютерных систем

Основные **достоинства** кластерных суперкомпьютерных систем:

- высокая суммарная производительность;
- высокая надежность работы системы;
- наилучшее соотношение производительность/стоимость;
- возможность динамического перераспределения нагрузок между серверами;
- легкая масштабируемость, то есть наращивание вычислительной мощности путем подключения дополнительных серверов;
- удобство управления и контроля работы системы.

Несмотря на высокие показатели производительности, вычислительные кластеры имеют достаточно много **недостатков:**

●*задержки* разработки и принятия общих *стандартов;*

●*большая доля нестандартных и закрытых разработок* различных фирм, затрудняющих их совместное использование;

●*высокая стоимость*: вычислительный кластер — это очень дорогой комплекс оборудования, стоимость которого (для суперкомпьютеров из TOP500) измеряется миллионами, а иногда и сотнями миллионов долларов. Для примера: стоимость оборудования суперкомпьютера Jaguar (США, #1 в списке TOP500 2010 г.) составила 200 млн. долларов.

●*большое энергопотребление*: вычислительный кластер — это также очень энергоемкое оборудование. Например, энергопотребление суперкомпьютера Tianhe-2 (Китай, #1 в списке TOP5004) составляет 17.8 Мватт.

●*необходимость охлаждения*: вся потребляемая энергия должна быть отведена от оборудования, чтобы не произошел перегрев. Как правило, системы охлаждения потребляют до половины всей электроэнергии, а также занимают порядка 70% общей площади вычислительного центра.

---

4 По состоянию на ноябрь 2015 г.

●*плохая масштабируемость*: из-за проблем с энергопотреблением и охлаждением существенной проблемой является масштабируемость. Увеличение вычислительной мощности кластера требует соответствующего увеличения мощности энергетического и охлаждающего оборудования.

●*высокая вероятность отказа компонент*: вычислительные кластеры состоят из сотен тысяч и даже миллионов электронных компонент, таких как процессоры, планки оперативной памяти, жесткие диски, блоки питания, вентиляторы и т.п. Это приводит к высокой вероятности возникновения отказа компонент даже при очень высокой надежности отдельных элементов

системы.

●*сложность обслуживания:* вычислительные кластеры состоят из десятков и сотен тысяч вычислительных узлов, километров проводов и сотен тысяч-миллионов компонент. Система такого масштаба очень сложна в обслуживании.

●*плохая управляемость*: одно из следствий большого масштаба системы — плохая *управляемость* (трудности управления одновременным доступом к файлам; сложности с управлением конфигурацией, настройкой, развертыванием, оповещениями серверов о сбоях и т.п.). Эта проблема осложняется тем, что услугами вычислительного кластера, как правило, пользуются тысячи пользователей, имеющих различные требования к программному обеспечению системы.

●*проблемы алгоритмизации и программирования*: огромное количество вычислительных узлов, сложная архитектура, а зачастую и сильная гетерогенность, приводит к тому, что разработка параллельной программы, задействующей большое количество вычислительных ресурсов, становится очень сложной задачей.

●*Закон Амдала*: даже разработав очень эффективный параллельный алгоритм решения задачи, программист сталкивается с ограничением ускорения работы программы на высокопараллельной системе. Виной тому Закон Амдала, который говорит об ограничении роста производительности компьютера с увеличением количества вычислителей. Согласно этому закону, ускорение программы за счет распараллеливания ее инструкций на множестве вычислителей ограничено временем, необходимым для выполнения ее последовательных инструкций.

4. Библиография

1. Лацис, А.О. Как построить и использовать суперкомпьютер / А.О. Лацис- М.:Бестселлер, 2003.-240 с
2. С. Яценко. Шина PCI-опыт разработки встраиваемых устройств / Chip News, № 3В4, 1998, с. 17-22.
3. R. Buyya High Performance Cluster Computing / R. Buyya – Prentice Hall PTR. NJ, USA, 1999, 128 p.
4. Васильков А. Новый рейтинг ТОП500 суперкомпьютеров [Электронный ресурс] http://www.computerra.ru/110512/sc-2014-top500_nov/
5. База знаний электронного журнала PC Mag [Электронный ресурс] http://ru.pcmag.com/upgrade-42005/13443/issue/klasternye-sistemy?p=3
6. Савяк В. Эффективные кластерные решения [Электронный ресурс] http://www.ixbt.com/cpu/clustering.shtml

7. Карпенко А.П. Многопроцессорные системы (MIMD-системы). Вычислительные кластеры. МГТУ им. Н.Э.Баумана [Электронный ресурс] http://bigor.bmstu.ru/?cnt/?doc=Parallel/ch010104.mod/?cou=Parallel/base.cou
8. Flynn M. Very high-speed computing system // M. Flynn, Proc. IEEE. 1966. N 54. P.1901-1909.
9. Flynn M. Some Computer Organisations and Their Effectiveness // M. Flynn, IEEE Trans. Computers. 1972. V.21. N 9. P.948-960
10. Воеводин В.В. Параллельные вычисления. / Воеводин В.В., Воеводин Вл. В – Спб.: БХВ-Петербург, 2002. — 608 с.
11. Hockney R. Parallel Computers: Architecture and Performance // Proc. of Int. Conf. Parallel Computing'85. 1986. P.33-69.
12. Hockney R. Classification and Evaluation of Parallel Computer Systems // R. Hockney, Lecture Notes in Computer Science. 1987. N 295. P.13-25.
13. Во Минь Тунг. Исследование кластерных вычислительных систем и разработка моделей назначения фрагментов параллельных программ: 05.13.15-Вычислительные машины, комплексы и компьютерные сети : диссертация кандидата технических наук / Во Минь Тунг, Моск. энерг. ин-т (МЭИ ТУ) . – М., 2010 . – 218 с.
14. Орлов С. А. Организация ЭВМ и систем / Орлов С.А. – Издательский дом "Питер", 2011 – 686с.
15. Официальный сайт проекта «TOP500» [Электронный ресурс] http://www.top500.org/lists/2015/11/
16. База знаний компании Alterozoom [Электронный ресурс] https://alterozoom.com/documents/8289.html
17. Ian Foster “WHAT IS THE GRID? A THREE POINT CHECKLIST” / Ian Foster – Department of Computer Science, University of Chicago, Chicago [Электронный ресурс] http://www.mcs.anl.gov/~itf/Articles/WhatIsTheGrid.pd
18. Бородин А.В. Технико-экономическое обоснование варианта резервирования сетевой компоненты отказоустойчивой масштабируемой вычислительной системы специального назначения // Кибернетика и программирование. - 2015. - 6. - С. 55 - 70. DOI: 10.7256/2306-4196.2015.6.17523. URL: http://www.e-notabene.ru/kp/article_17523.html
5. [http://www.supercomputers.ru [Электронный ресурс]. Режим доступа свободный].

Вопросы к главе 4.

1. Технологии организации высокопроизводительных вычислений
2. Технология организации высокопроизводительных вычислений на основе вычислительных кластеров. Что такое кластер. Типовая кластерная система

3. Два подхода к созданию кластеров. Основные признаки классификации кластерных систем
4. Принципы построения и использования параллельных вычислительных систем. Принципы формирования параллелизма
5. Основные достоинства и проблемы кластерных суперкомпьютерных систем

Глава 5. МЕТОДЫ ПОСТРОЕНИЯ ВЫЧИСЛИТЕЛЬНЫХ КЛАСТЕРОВ

5.1. Методы построения и типы кластеров

Кластерные системы могут иметь различную организационную структуру. По физической реализации кластеры бывают:

- **Кластеры специальной разработки** с быстродействием 10^{11}–10^{12} плавающих операций в секунду (терафлопсовый диапазон). К таким кластерам относится СКИФ (**С**упер**к**омпьютерная **И**нициатива **Ф**еникс), созданный по программе Союза Беларусь–Россия. Появление этого кластера и сам процесс разработки подготовили основу для развития параллельных вычислений в республике, вследствие чего возникла проблема широкой подготовки программистов для решения параллельных задач.
- **Кластеры, которые строятся на базе уже имеющихся локальных сетей** из персональных компьютеров. Для создания кластеров в этом случае требуется только дополнительное программное обеспечение (ПО), поэтому их можно организовать в вузах и небольших организациях. Такие кластеры имеют относительно небольшое быстродействие, но их удобно использовать для обучения основам параллельного программирования и подготовки параллельных программ, которые затем могут выполняться на больших кластерах [4].

По способу представления вычислительных узлов можно выделить следующие группы:

- Кластер расположен в виртуальной среде. Все узлы вычислительного кластера являются виртуальными машинами. Как правило, используются

гипервизоры на основе базовой ОС(Microsoft Virtual PC, VMware Workstation, QEMU, Parallels, VirtualBox) или же гибридные гипервизоры(Microsoft Virtual Server, Sun Logical Domains, Xen, Citrix XenServer, Microsoft Hyper-V). В более редких случаях могут использоваться автономные гипервизоры, не использующие базовую операционную системы. Такие гипервизоры используют собственные драйвера и планировщик, поэтому не зависят от базовой ОС. Так как автономный гипервизор работает непосредственно на оборудовании, то он более производителен. В качестве оборудования, на котором функционируют виртуальные машины, может выступать один или несколько высокопроизводительных сервером или персональный компьютер.

- Кластер на основе физических серверов. При такой реализации, в узлах кластера расположены только серверы или персональные компьютеры. В связи с этим, производительность кластера ограничена только производительностью или количеством его узлов.

По однородности вычислительных узлов кластеры могут быть:

- Однородными. Все узлы кластера являются однотипными. Центральный процессор, оперативна память, дисковая подсистема, сетевые адаптеры - все компоненты узла являются однотипными. При такой организации узлы являются взаимозаменяемыми. Замена производится либо на аналогичный компонент вычислительного узла, либо производится замена узла целиком. Такая организация вычислительного кластера позволяет свести к минимуму задержки, связанные с неравномерной загрузкой суперкомпьютера. Как правило не возникает ситуаций при которых одни вычислительные узлы простаивают в связи с тем, что на других еще производится обработка данных.
- Смешанного типа. При такой организации в качестве узла кластера может выступать любой вычислитель, который поддерживает программное обеспечение и имеет общий доступ к разделяемым ресурсам. В связи с этим, могут наблюдаться неоднородность загрузки вычислительного кластера. Более производительные узлы будут заканчивать работу раньше, чем менее производительные. При работе на таком кластере требуется уделять большее внимание при написании программ.

5.2. Вычислительная сеть и коммутация вычислительного кластера.

Коммутационная сеть кластера беспрецедентно важна. Именно она определяет среду, в которой передаются данные между процессорами. В современных кластерах используются каналы передачи данных более 100 Гбит/с. На скорость передачи данных большее влияние оказывает сетевые провода и коммутационное оборудование. Наиболее распространенной сетью, во всем мире, является сеть InfiniBand.

Рис.5.1.- CX4 порты InfiniBand (коммутатор Voltaire ISR-6000)

InfiniBand это архитектура коммутации соединений типа "точка-точка". Каждая линия связи представляет собой четырехпроводное двунаправленное соединение с базовой пропускной способностью 2,5 Гбит/с в каждом направлении и гибким выбором физических линий передачи. В архитектуре InfiniBand определен многоуровневый протокол (физический, канальный, сетевой и транспортный уровни) для реализации аппаратными средствами, а также программный уровень для поддержки управляющих функций и скоростного обмена данными (с малыми задержками) между устройствами [5]. Перечислим основные характеристики технологии InfiniBand:

- Возможность масштабирования пропускной способности линий связи до 300 Гбит/с в дуплексном режиме;
- поддержка различных физических линий: печатных проводников (на объединительной панели), медных или оптоволоконных кабелей;
- связь на базе коммутации пакетов с сохранением целостности данных и управлением потоком;
- качество обслуживания;
- аппаратный гибкий транспортный механизм;
- оптимизированный программный интерфейс и удаленный прямой доступ в память (RDMA);

- инфраструктура управления, поддерживающая функции отказоустойчивости, аварийного переключения и "горячей" замены.

В спецификации InfiniBand определен чрезвычайно гибкий и масштабируемый физический уровень, обеспечивающий возможности дальнейшего наращивания пропускной способности и добавления новых типов поддерживаемых физических линий. Его характеристики:

- двунаправленный обмен сигналами со скоростями 2,5 Гбит/с (1X), 10 Гбит/с (4X) и 30 Гбит/с (12X). Пропускная способность указана для 10-разрядных данных. При схеме кодирования 8B/10B к 8 разрядам данных добавляются дополнительные два разряда, необходимые для синхронизации. Таким образом, реальная пропускная способность линий InfiniBand составляет 2, 8 и 24 Гбит/с соответственно;
- малое число требуемых физических проводников: всего 4, 16 или 48 проводников для линий 1X, 4X и 12X;
- встроенные тактовые импульсы со схемой кодирования 8B/10B без передачи сигналов по вспомогательному каналу;
- объединение каналов связи на физическом уровне;
- поддержка различных типов физических проводников: с медной жилой (для экономии) и с оптоволокном (на большие расстояния);
- несложный физический уровень с малым потреблением энергии.

Применение схемы кодирования 8B/10B и встраивание тактовых импульсов в поток данных снижает требования, предъявляемые к прокладке проводников по объединительным панелям по сравнению с другими технологиями ввода/вывода с раздельными шинами данных и линиями передачи синхроимпульсов, когда расфазировка сигналов данных и синхроимпульсов может нарушить целостность сигнала и привести к ошибкам передачи [5].

Имеется возможность объединения нескольких линий InfiniBand на физическом уровне в один канал на основе технологии "расщепления последовательности байтов" (byte stripping). Для канала 4X это означает одновременную посылку четырех байтов по каждой отдельной линии InfiniBand (аналогично, двенадцати байтов по каждой линии в составе канала 12X InfiniBand). Этот уникальный по своим возможностям механизм объединения на физическом уровне превосходит по своим характеристикам аналогичный механизм, реализованный в стандарте Ethernet 802.3ad. В данном случае объединенные линии всегда используются более эффективно.

Этого нельзя сказать про Ethernet, где параллельные линии могут использоваться лишь для одновременной пересылки пакетов (и только целиком) от разных источников. Тем самым предотвращается "вклинивание" маленьких пакетов в передачу больших с последующими нарушением порядка пакетов, повторными передачами и т.д. Таким образом, на практике объединение Ethernet-линий не приводит к ожидаемому повышению пропускной способности, поскольку параллельные линии используются в полную силу лишь 15 - 20% времени.

Рис. 5.2. - Кабель InfinyBand

Другое достоинство архитектуры InfiniBand заключается в применении простой дифференциальной схемы обмена сигналами на физическом уровне. Эта схема отличается более высокой устойчивостью к помехам, целостностью сигналов и меньшей вероятностью ошибок передачи при гораздо меньшем потреблении энергии, чем более сложные формирователи физического уровня как, например, в Gigabit Ethernet. Поскольку длина InfiniBand-линий ограничена 17 метрами при использовании кабелей с медным проводом, неудивительно, что формирователи InfiniBand потребляют на порядок меньше мощности, чем формирователи Gigabit Ethernet, рассчитанные на линии связи длиной более 100 метров [5].

Столь низкий уровень потребления энергии формирователями InfiniBand обеспечивает более высокую степень объединения, чем может предложить Ethernet-коммутатор. Несколько производителей уже предлагают однокристальные интегральные InfiniBand-коммутаторы с 32-мя интегрированными приемопередатчиками с пропускной способностью 2,5 Гбит/с каждый. Таким образом, интегрированные InfiniBand-коммутаторы обеспечивают значительную экономию средств, пространства и энергии по сравнению с многокристальными устройствами для Gigabit Ethernet.

В основе построения системной сети (SAN) на базе InfiniBand лежат четыре типа основных системных компонентов, связанных друг с другом линиями InfiniBand:

- канальный адаптер главного узла (HCA - Host Channel Adapter), который обеспечивают соединение центрального процессора главного узла со структурой InfiniBand и содержит аппаратные средства поддержки высокоэффективного защищенного обмена с системной памятью;
- канальный адаптер целевого узла (TCA - Target Channel Adapter), который обеспечивают соединение структуры InfiniBand с другими устройствами ввода/вывода типа Ethernet, Fibre Channel или запоминающими устройствами;
- коммутатор, который обеспечивают пересылку пакетов в конечные точки по указанному адресу и с указанным качеством обслуживания;
- маршрутизатор, который обеспечивают передачу пакетов между подсетями; Маршрутизаторы InfiniBand делят сеть InfiniBand на подсети и не являются источниками и приемниками никаких других пакетов, кроме управляющих;

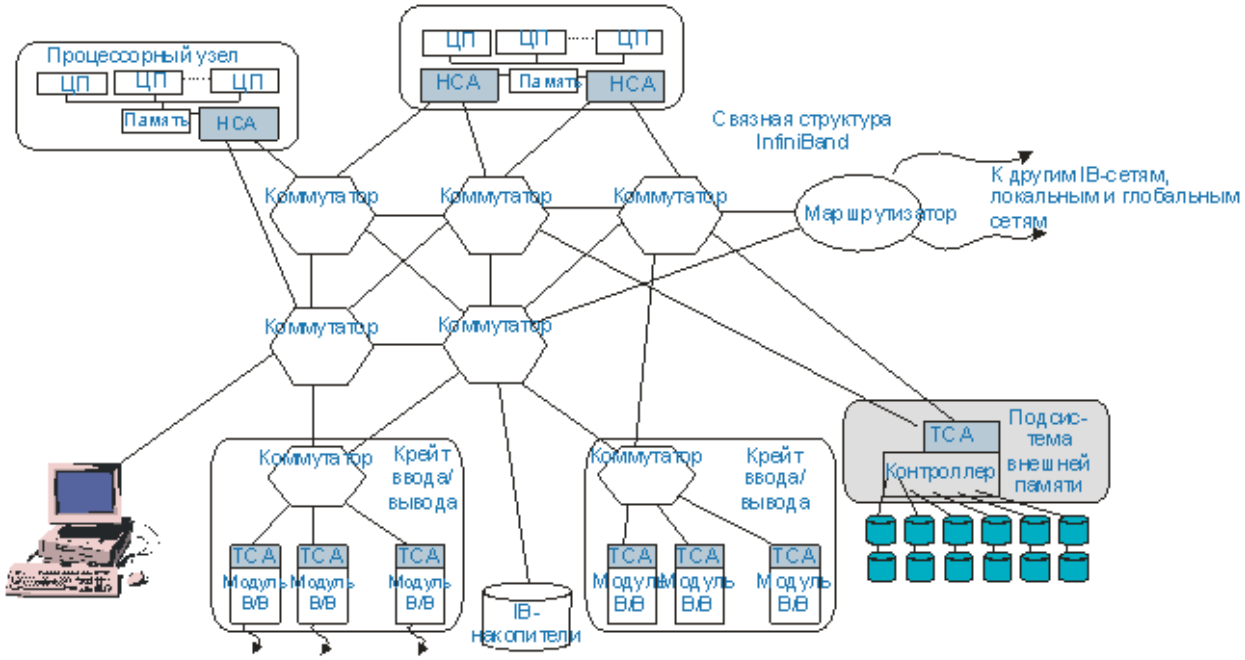


Рис. 5.3.- Основные компоненты SAN-сети на основе InfiniBand

В каждой подсети работает своя программа-менеджер подсети (SM Subnet Manager), который выполняет задачи обнаружения связной структуры, инициализации устройств и организации соединений. SM-менеджер может

исполняться в хост-системе, коммутаторе и маршрутизаторе. В состав всех компонентов входит агент управления подсетью (SMA Subnet Management Agent), исполняющийся как клиент, выполняющий запросы SM-менеджера.
В InfiniBand определен весьма гибкий набор линий связи и механизмов транспортного уровня, обеспечивающий точную настройку характеристик SAN-сети на базе InfiniBand в зависимости от прикладных требований, в число которых входят:

- пакеты переменного размера;
- максимальный размер единицы передачи: 256, 512, 1К, 2К и 4К байт;
- заголовки локальной трассы уровня 2 (LRH Local Route Header) для направления пакетов в нужный порт канального адаптера;
- дополнительный заголовок уровня 3 для глобальной маршрутизации (GRH Global route header);
- поддержка групповой передачи;
- вариантные и инвариантные контрольные суммы (VCRC и ICRC) для обеспечения целостности данных.

Выбор максимального размера единицы передачи определяет такие характеристики системы, как неустойчивость синхронизации пакетов, величина накладных расходов на инкапсуляцию и длительность задержки, используемые при разработке систем с несколькими протоколами. Возможность опускать сведения о глобальном маршруте при пересылке в пункт назначения локальной подсети позволяет снижать издержки локального обмена данными. Код VCRC рассчитывается заново каждый раз при прохождении очередного звена канала связи, а код ICRC при получении пакета пунктом назначения, что гарантирует целостность передачи по звену и всему каналу связи.
В InfiniBand определены управление потоком на основе разрешений (для предотвращения блокировок головного пакета (head of line blocking) и потерь пакетов), управление потоком на канальном уровне и сквозное управление потоком. По своим возможностям управление на канальном уровне на основе разрешений превосходит широко распространенный протокол XON/XOFF, устраняя ограничения на максимальную дальность связи и обеспечивая лучшее использование линии связи. Приемный конец линии связи посылает передающему устройству разрешения с указанием объема данных, которые могут быть надежно получены. Данные не передаются до тех пор, пока приемник не пошлет разрешение, свидетельствующее о наличии свободного пространства в приемном буфере. Механизм передачи разрешений между устройствами встроен в протоколы соединений и линий связи для гарантии надежности управления

потоком. Управление потоком на канальном уровне осуществляется для каждого виртуального канала отдельно, чем предотвращается распространение конфликтов передачи, свойственное другим технологиям типа Ethernet.

В современных кластерах сеть передачи данных обычно отделяется от управляющей сети. Вполне логичным было возникновение идеи ***построения управления вычислительной системой без использования сети Ethernet.***

Функции сети в кластерной вычислительной системе

Если мы хотим оставить только одну сеть, то необходимо обеспечить реализацию всей ***функциональности управления*** работой современной высокопроизводительной кластерной вычислительной системы. Упрощенно группами таких функций являются:

1. **Обмен данными по интерфейсу MPI**. Message Passing Interface (интерфейс передачи сообщений) – программный интерфейс (API) для передачи информации, позволяющий осуществлять обмен сообщениями между компьютерами, выполняющими параллельную задачу. Он является наиболее распространенным стандартом интер- фейса обмена данными в параллельном программировании. ***Программное обеспечение современных кластеров чаще всего выполняет обмен данными именно через MPI.***

2. **Пересылка IP пакетов.** Сетевые сервисы, необходимые для работы кластера, используют этот протокол для передачи информации по сети. Например, корневая файловая система узлов обычно монтируется с помощью Network file system (NFS), которая работает в соответствии со стеком протоколов TCP/IP, а следовательно, поддержка TCP/IP является необходимым условием функционирования кластерного комплекса.

3. **Удаленная загрузка узлов**. Учитывая большое количество вычислительных узлов в высокопроизводительных кластерах, устанавливать на каждый из них от- дельную операционную систему представляется достаточно сложным, а в случае применения бездисковых узлов и невозможным процессом. Чаще всего в кластерных комплексах

реализуется удаленная загрузка одного экземпляра операционной системы. Такой подход значительно упрощает администрирование вычислительной системы.

4. **Удаленное управление аппаратными средствами кластера, не зависящее от операционной системы.** Как известно, кластерные вычислительные системы чаще всего находятся на значительном расстоянии от пользователей и даже адми- нистраторов, поэтому любые простые операции, требующие непосредственного доступа к оборудованию, например, перезагрузка «зависшего» узла, без наличия специальных средств, превращается в проблему. Для упрощения обслуживания таких систем используют специальные средства, позволяющие многие подобные операции выполнять удаленно.

Полноценная работа современного высокопроизводительного кластера требует реализации как минимум упомянутых 4 групп функций и, если ставится цель отказаться при этом от использования *Ethernet*, то оборудованием всего одной сети. Среди существующих сейчас высокоскоростных сетей наиболее интересной, в контексте данной темы, является *InfiniBand* [3]. Причины такого интереса:

1. **Высокая пропускная способность**. Для передачи данных в *InfiniBand* приме- няются 4-проводные двунаправленные соединения. Базовая пропускная способность составляет 2,5 Гбит/с в каждом направлении при использовании Single Data Rate (SDR), поддерживается также работа с Double Data Rate (DDR) – 5 Гбит/с и Quad Data Rate (QDR) – 10 Гбит/с. Сетевые платы и коммутаторы имеют порты 4x, скорость при этом составляет соответственно 10, 20 или 40 Гбит/с.

2. **Многоплановая функциональность.** *InfiniBand* поддерживает множество протоколов, среди которых:

– *Remote Direct Memory Access (RDMA)* [3] – группа протоколов удалённого прямого доступа к памяти, при котором передача данных из памяти одного

компьютера в память другого компьютера происходит без участия операционной системы и использования ресурсов центрального процессора;

– *SCSI RDMA Protocol (SRP)* [3] – протокол обмена данными между *SCSI* уст- ройствами с использованием *RDMA*;

– *IP over InfiniBand (IPoIB)* [3] – группа протоколов, описывающих передачу IP-паке- тов через *InfiniBand*;

– *Socket Direct Protocol (SDP)* [3] – протокол установления виртуальных соединений и обмена данными между сокетами через *InfiniBand*.

Кроме поддержки многих протоколов имеются также многие программные средства, позволяющие расширить возможности применения данной сети. *Boot over InfiniBand (BoIB)* [3] является одним из них и позволяет осуществлять удаленную загрузку операционной системы в сетях *InfiniBand*, что до недавнего времени было возможным только при наличии *Ethernet*.

3. **Интенсивность развития**. *InfiniBand* является довольно востребованной и активно развивающейся технологией. Постоянно совершенствуются ее как аппаратные, так и программные средства. Функциональность постоянно расширяется, что позво- ляет поддерживать многие самые новые и передовые технологии.

Реализация необходимых условий для ункционирования кластера при использовании сети InfiniBand

Обеспечение обмена данными между приложениями в соответствии со стандартом *MPI*. В современных высокопроизводительных кластерных вычислительных системах обмен данными между приложениями, выполняющимися на разных вычислительных узлах, посредством протокола *MPI* уже давно не является функцией *Ethernet*. Как раз для этого и используются высокоскоростные сети, одной из которых и является *InfiniBand*. Следовательно, имеется большая практика успешного ее приме- нения для решения вышеупомянутой задачи. Потому никаких проблем с

реализацией данной функции без *Ethernet* возникнуть не должно.

Передача *IP* пакетов. Правила передачи *IP* пакетов через сеть *IB* описывает группа протоколов *IP over InfiniBand (IPoIB)* [3], в соответствии с которыми все сетевые приложения, использующие протоколы TCP/IP в сети *Ethernet*, могут без изменений использоваться в сети *InfiniBand*. Отдельно стоят приложения, работающие с физичес- кими адресами сетевых плат и использующие *Ethernet* пакеты напрямую (примером таких приложений являются *DHCP*-сервер и *DHCP*-клиент), в этих случаях не представляется возможным динамически получать настройки сети с *DHCP*-сервера и создаются некоторые неудобства при сетевой загрузке операционной системы.

Сетевая загрузка операционной системы. Не так давно в рамках проекта *Etherboot/gPXE* [4] была разработана технология, позволившая осуществлять удаленную загрузку операционной системы *Linux* в сетях *InfiniBand*. Она базируется на тех же принципах, что и загрузка через *Ethernet*, но из-за определенных различий в аппаратных средствах имеет свои особенности. Для понимания причины их возник- новения рассмотрим сначала классическую *PXE*-загрузку операционной системы *Linux*.

Для загрузки применяются протоколы *IP*, *UDP*, *DHCP* и *TFTP*, осуществляется она загрузчиком *pxelinux*, который можно создать на базе пакета *syslinux* [4]. Кроме того, *BIOS* сетевой платы должен иметь специальный *PXE-код*. *PXE* загрузка опера- ционной системы *Linux* имеет следующие этапы:

1. Посылка запросов *DHCP* серверу *PXE-кодом* сетевой платы и получение начальных настроек сети, а также адреса *TFTP* сервера и пути к загрузчику *pxelinux*.
2. Загрузка с *TFTP* сервера образа *pxelinux* и передача ему управления.
3. Получение с *TFTP* сервера образов ядра *Linux* и временной корневой файловой системы *initrd*, загрузка ядра и монтирование файловой система *initrd*, с последую- щей передачей управления скрипту инициализации базовой операционной системы.
4. Получение сетевых настроек от *DHCP* сервера и настройка сети.
5. Монтирование по *NFS* основной корневой файловой системы и запуск скрипта инициализации основной операционной системы.

Из этого перечисления следует, что для поддержки сетевой загрузки на обору- довании *InfiniBand* необходимо выполнение следующих условий:

- Плата InfiniBand должна поддерживать технологию PXE.
- TFTP сервер должен работать по протоколу IPoIB.

- NFS также должен работать в соответствии с протоколом IPoIB.
- Должно быть обеспечено динамическое получение настроек IPoIB сетевого интер- фейса.
- В ядро Linux должны быть включены драйвера сетевой платы IB, а также под- держка протокола IPoIB.
- Скрипт инициализации базовой операционной системы, входящий в состав initrd, должен быть адаптирован к новому сетевому окружению.

С настройкой *TFTP* и *NFS* никаких проблем не возникает, так как они без всяких изменений работают через *InfiniBand*, используя протокол *IPoIB*. В новых версиях ядер *Linux* присутствуют драйверы для плат *InfiniBand* с поддержкой про- токола *IPoIB*, т.е. необходимо собрать ядро *Linux* с этими драйверами. С остальными пунктами все не так просто. Стандартный *BIOS* плат *InfiniBand* не содержит в себе *PXE-кода*. Однако есть возможность это исправить. Для этого необходим образ ориги- нального *BIOS*, средства для его модификации и перезаписи, а также сам *PXE-код*, который входит в состав программного пакета *Boot over IB* [3]. Все эти компоненты можно получить на сайте производителя сетевого оборудования *InfiniBand*.

Отдельно нужно остановиться на динамическом получении настроек сетевого интерфейса *IPoIB* посредством протокола *DHCP*. Ни стандартный сервер *DHCP*, ни стандартный клиент не поддерживают работу с *InfiniBand*. С помощью патча, вхо- дящего в состав пакета *Boot over IB*, можно получить базовую поддержку *InfiniBand* сервером *DHCP*, однако полной совместимости пока нет. К тому же, изменения не касаются клиентской части, а значит, в дальнейшем для полной совместимости необ- ходима доработка как сервера, так и клиента. Связано это с тем, что размер физического адреса плат *InfiniBand* не соответствует протоколу *DHCP*, вследствие чего его нельзя использовать для идентификации *DHCP* сессии. В документации пакета *Boot over IB* предлагается в качестве решения этой проблемы использовать специальный иденти- фикатор, который должен быть записан в конфигурационном файле *DHCP*-клиента. Подобное решение проблемы вполне применимо при загрузке отдельных серверов, но не для загрузки вычислительных узлов кластера, так как все конфигурационные файлы у них являются общими.

Полная совместимость *InfiniBand* и *DHCP* до сих пор является только частично решенной задачей, следовательно, приходится находить локальные решения, как, например, динамическое формирование идентификатора,

зависящее от физического адреса сетевой платы, или вообще получать нужные настройки иным путем.

Удаленное управление аппаратными средствами кластера. Наиболее распрост- раненным средством для решения этой задачи является, безусловно, *IPMI* [5]. Все существующие реализации *IPMI* требуют наличия *Ethernet*. Подобных по функциональ- ности средств, способных работать в сети *InfiniBand* не создано и не планируется. Однако возможен альтернативный вариант решения данной задачи – использование сервисной сети *ServNET* [6].

Основная функциональность сервисной сети ServNET:

- селективный сброс узла;
- селективное и «плавное» включение/выключение электропитания узла (предупреждает износ оборудования и позволяет избежать сильного скачка напряжения при включе- нии системы);
- доступ к сериальной консоли узла, поддерживающий: изменение параметров *BIOS* узла; выбор (*LILO*) загружаемой ОС; параметры загрузки ядра *Linux*; любые команды в консольном режиме; мониторинг критических сообщений ОС; «посмертное» чтение (из энергонезависимой памяти платы *ServNET*) нескольких последних сообщений ОС.

Эта сервисная сеть обладает меньшей, чем в *IPMI*, но вполне достаточной функциональностью, она проще в установке, настройке и использовании, кроме того, в силу своей простоты является достаточно стабильной в работе и отказоустойчивой. При всем этом ее стоимость на порядок ниже стоимости *IPMI*. Учитывая все вышеперечисленное, *ServNET* вполне может быть использована для решения задачи удаленного управления аппаратными средствами кластера.

Преимущества и недостатки кластерной вычислительной системы без сети Ethernet

При проектировании высокопроизводительных кластерных вычислительных систем необходимо учитывать, что с ростом производительности системы значительно увеличивается количество необходимого вспомогательного оборудования, в том числе и коммутационного. В действительно больших системах это становится допол- нительной проблемой, так как может серьезно повлиять как на стоимость системы, так и на ее отказоустойчивость. Поэтому перспектива уменьшить количество сетевого оборудования и коммутационных

кабелей почти в два раза выглядит весьма неплохо. Итак, а в чем же именно мы выигрываем в данной ситуации:

- Увеличивается отказоустойчивость.
- Снижается стоимость.
- Упрощается инсталляция и обслуживание.

К недостаткам можно, пожалуй, отнести только сложность внедрения. Этот не- достаток можно объяснить «юным возрастом» технологии, вследствие чего некоторые решения не полностью реализованы. Наиболее сложной из таких проблем является недостаточная совместимость *InfiniBand* и *DHCP,* которая доставляет немало труднос- тей. Но, тем не менее, даже это не является критичным, к тому же данная проблема может быть решена в процессе дальнейшего развития технологии.

Выводы

Приведенная концепция построения суперкомпьютеров кластерной архитектуры без использования сети *Ethernet* имеет ряд существенных преимуществ по сравнению с классическими. Учитывая то, что при этом недостатки минимальны, перспективы ее развития выглядят весьма неплохими. Безусловно, имеются некоторые трудности с первоначальным внедрением технологии, но они вполне преодолимы. Практическим подтверждением этого является тот факт, что в 2008 году в Институте кибернетики НАН Украины был модернизирован один из кластеров суперкомпьютерного комп- лекса *СКИТ* [7], с применением вышеописанных подходов, что позволило полностью отключить его от сети *Ethernet*. В дальнейшем планируется развитие и внедрение данной технологии в кластерных решениях Института кибернетики НАН Украины.

Литература

1. Режим доступа: http://parallel.ru/tech/tech_dev/mpi.html
2. Режим доступа: http://parallel.ru/computers/interconnects.html
3. Режим доступа: http://www.mellanox.com
4. Режим доступа: http://www.etherboot.org
5. Режим доступа: http://www.intel.com/design/servers/ipmi/ipmi.htm
6. Режим доступа: http://www.t-platforms.ru
7. Режим доступа: https://icybcluster.org.ua/

Вопросы к главе 5.

1. Методы построения вычислительных кластеров.
2. Вычислительная сеть и коммутация вычислительного кластера. Функции сети в кластерной вычислительной системе

3. Реализация необходимых условий для ункционирования кластера при использовании сети InfiniBand. Преимущества и недостатки кластерной вычислительной системы без сети EtherNet

Глава 6. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ КЛАСТЕРОВ

От используемого программного обеспечения в вычислительном кластере зависит производительность, надежность и специфика решаемых задач. Именно программное обеспечение определяет круг задач, которые могут решаться на кластере. В качестве необходимого программного обеспечения можно выделить:

- Операционная система (или гипервизор, при отказе от базовой ОС). Определяет, какое программное обеспечение, и в каких версиях будет использовано при работе на кластере.
- Система управления задачами кластера. Представляет собой программный продукт, в котором производится управление всеми узлами кластера и поступающими задачами.
- Система мониторинга кластера. Включает в себя механизмы для отслеживания следующих параметров: производительность узлов, сетевая доступность, нагрузка на подсистему хранения данных. Резервные каналы доступа к узлам кластера.

6.1. Операционные системы

Операционная система — комплекс управляющих и обрабатывающих программ, которые, с одной стороны, выступают как интерфейс между устройствами вычислительной системы и прикладными программами, а с другой стороны — предназначены для управления устройствами, управления вычислительными процессами, эффективного распределения вычислительных ресурсов между вычислительными процессами и организации надёжных вычислений. Это определение применимо к большинству современных операционных систем общего назначения.

В качестве операционных систем для кластеров, как правило, используются серверные версии UNIX подобных операционных систем(SUSE Server, Red Hut, Ubuntu Server). До недавнего времени эти системы были фактически стандартом для вычислительных кластеров. После 2006 года в связи с выходом Microsoft HPC Cluster Server появилась возможность создавать вычислительные кластеры на основе операционных систем Microsoft.

При выборе операционной системы следует основываться, прежде всего, на рекомендациях разработчиков программного обеспечения. Однако, если есть выбор, то при прочих равных условиях следует отдать предпочтение Linux.

Под Linux доступно огромное количество серверного программного обеспечения, компиляторов, библиотек, средств отладки и пр. Большое количество программного обеспечения имеется в свободном доступе, для многих программ есть исходные коды и обширная документация. Плюсом Linux является "прозрачность" для пользователя и системного администратора, что позволяет быстрее и проще разрешать все возникающие проблемы.

Основой кластера является не операционная система, а коммуникационная среда (PVM, MPI), обеспечивающая возможность частям параллельной программы, выполняющимся на разных компьютерах, эффективно взаимодействовать между собой.

Рассмотренные ранее средства для построения кластера (PVM, MPI) имеют реализации как для операционных систем семейства UNIX (Linux, FreeBSD и т.п.), так и для систем фирмы Майкрософт. Кластер может быть создан и под ОС Windows, причем трудозатраты на установку коммуникационной среды будут такими же, как и в варианте с UNIX, то есть небольшими. Основная трудность будет заключаться в том, чтобы научиться писать параллельные программы.

Однако, следует заметить, что подавляющее большинство высокопроизводительных кластеров в мире работает все же в среде UNIX. Такой выбор обусловлен небольшими трудозатратами по настройке и установке кластера, более низкими требованиями ОС, по сравнению с ОС Windows, и большим количеством поддерживаемого программного обеспечения. Поскольку библиотеки для параллельных вычислений MPICH/MPI являются кроссплатформенными, то выбор операционной системы (Windows vs Linux) не важен. Однако следует учесть тот факт, что Linux является заметно менее ресурсоемкой системой. Например, при использовании PelicanHPC GNU Linux система занимает в оперативной памяти не более 40Мб! Вся остальная память доступна параллельной программе. Это очень важный фактор в том случае, когда кластер используется с целью моделирования процессов на как можно более подробной сетке.

6.1.1. Операционные системы для кластерных систем: Windows Compute Cluster Server 2003

Microsoft Windows Compute Cluster Server (CCS) 2003 является интегрированной платформой для проведения высокопроизводительных вычислений. С использованием этой платформы, производить вычисления на кластерах, имеющих от нескольких до сотен, и даже, тысяч узлов.

Конфигурирование, мониторинг, управление и обеспечение безопасного доступа для кластеров является очень трудной задачей, на решение которой требуется обычно большое количество времени и ресурсов. Одна из целей Windows CCS 2003 - максимально упростить управление вычислительными кластерами и сократить общую стоимость владения ими (TCO - total cost of ownership), одновременно сделав их доступными для более широкого круга пользователей. В частности, для Windows CCS 2003 процессы установки и администрирования максимально автоматизированы. Это достигается через интеграцию Windows CCS 2003 с технологиями Active Directory и Microsoft Operations Manger (MOM). Для удобства использования, в состав Windows CCS 2003 входит планировщик заданий (job scheduler), работа с которым обеспечивается через графический интерфейс пользователя и через командную строку.

Windows CCS 2003 поддерживает исполнение параллельных приложений, базирующихся на стандарте Message Passing Interface (MPI). Кроме того, для того, чтобы можно было использовать среду разработки Visual Studio 2005 для создания параллельных приложений, в нее включены поддержка стандарта OpenMP, а также возможность отладки параллельных программ, использующих MPI.

Версия MPI, используемая в Windows CCS 20033 - MS MPI, является вариантом одной из реализаций MPI, разработанной в Аргонской национальной лаборатории, и названной MPICH2.

MS MPI построен на основе WinSock API (Windows Sockets networking API), и передача данных через сеть может осуществляться либо через обычный протокол TCP/IP, либо через драйвер WinSock Direct Provider, обеспечивающий непосредственную (игнорируя слой TCP/IP) передачу данных через сетевую аппаратуру. Тем самым, MS MPI обеспечивает поддержку сетей Ethernet либо Gigabit Ethernet через TCP/IP, и сетей с малым временем задержки и широкой полосой пропускания, таких как Infiniband или Myrinet через драйверы WinSock Direct.

MS MPI обеспечивает поддержку языков программирования C, Fortran 77 и Fortran 90. Среда разработки Microsoft Visual Studio 2005 включает в себя отладчик параллельных программ, написанных с применением MS MPI.

Разработчики могут запускать свои MPI-приложения на множестве вычислительных узлов непосредственно из Visual Studio, причем Visual Studio автоматически связывается с процессами на узлах, что позволяет разработчикам, по индивидуальному выбору, останавливать программу на любом из узлов и просматривать значения программных переменных.

Windows CCS 2003 поддерживает пять различных сетевых топологий головного и вычислительных узлов кластера, где эти топологии отличаются друг от друга своей производительностью (скоростью передачи данных) и возможностями доступа в данную сеть. Каждая из топологий включает в себя, по меньшей мере, один из следующих типов сетей:

- общая (public) сеть,
- частная (private) сеть и
- MPI-сеть,

или, возможно, некоторую их комбинацию.

Общая сеть - это, обычно, существующая на предприятии или в организации Ethernet-сеть, которая объединяет, в частности, вычислительные узлы кластера. Если для кластера не предусмотрена выделенная частная сеть, то весь внутрикластерный управляющий трафик будет проходить через общую сеть.

Частная сеть - это выделенная специально для кластера сеть, через которую осуществляется внутрикластерная передача сообщений. По этой сети проходит трафик, связанный с управлением кластером и развертыванием (deployment) вычислительных узлов, а также MPI-трафик (если не существует специальной MPI-сети).

MPI-сеть - это выделенная, высокоскоростная сеть, предназначенная для передачи сообщений параллельных приложений, работающих на вычислительных узлах кластера. Если в кластере такой сети не существует, то MPI-трафик передается через частную сеть. Если и частной сети не существует, то MPI-трафик будет проходить через общую сеть.

Использование отдельных сетей для передачи внутрикластерного (служебного) и MPI-трафика между вычислительными узлами и головным узлом значительно повышает общую производительность кластера и разгружает от этого трафика общую сеть. Тем не менее, доступ в общую сеть с вычислительных узлов, по-прежнему, возможен с помощью сервиса NAT (Network Address Translation), запущенного на головном узле.

Windows CCS 2003 поддерживает кластеры, состоящие из одного головного узла и одного или нескольких вычислительных узлов, как показано на Рис 1.10.

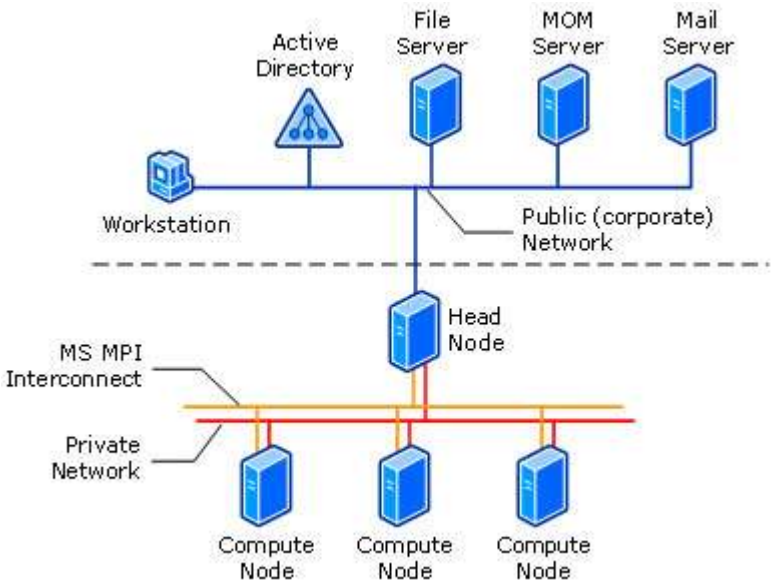


Рис. 6.1 - Типичная структура сети для работы Windows CCS 2003

Головной узел управляет и является посредником для доступа ко всем ресурсам кластера и представляет собой единую точку для управления, развертывания ресурсов и исполнения заданий на вычислительном кластере. Windows CCS 2003 может использовать существующую инфраструктуру Active Directory для обеспечения безопасности, управления учетными записями и общим управлением всеми операциями, а также может быть интегрирован с такими средствами управления, как MOM (Microsoft Operations Manager) 2005 и SMS (Systems Management Server) 2003.

Развертывание Microsoft CCS 2003 включает в себя:

1. установку базовой операционной системы на головном узле,
2. присоединение головного узла к существующему Active Directory-домену,
3. установку пакета Compute Cluster Pack.

Если планируется использовать сервис удаленной установки (RIS - Remote Installation Service), то RIS будет установлен и сконфигурирован на последнем шаге процедуры развертывания.

После окончания установки пакета Compute Cluster Pack на головном узле, на его экране отображается ToDo-список, который показывает какие шаги необходимо выполнить для завершения конфигурирования вычислительного кластера. Эти шаги включают в себя:

1. определение топологии сети,
2. конфигурирование RIS с помощью соответствующего мастера,
3. добавление вычислительных узлов в кластер,

4. конфигурирование информации о пользователях и администраторах кластера

На головном узле кластера устанавливаются следующие сервисы:

1. Management Service - управление кластером, обнаружение узлов, управление конфигурациями
2. Scheduler Service - постановка заданий в очередь, планирование их запуска, распределение ресурсов, исполнение заданий
3. SDM Store Service - операции чтения/записи в хранилище данных System Definition Model (SDM), управление целостностью
4. MPI Service - реализация MPI на основе MPICH2
5. Node Manager Service - устанавливается на головном узле, когда он сконфигурирован также и в качестве вычислительного узла; взаимодействует с Scheduler Service и отвечает за исполнение заданий
6. SQL Service - Microsoft SQL Server 2000 Desktop Engine (MSDE)

На вычислительных узлах кластера устанавливаются следующие сервисы:

1. Management Service - взаимодействует с одноименным сервисом на головном узле; отвечает за управление кластером, обнаружение узлов и управление конфигурациями со стороны вычислительного узла
2. MPI Service - реализация MPI на основе MPICH2
3. Node Manger Service - взаимодействует с Scheduler Service на головном узле; отвечает за исполнении заданий

В настоящее время основной операционной системой, используемой при проведении учебных занятий в вузах, является операционная система Windows. При всех достоинствах системы ей присущи некоторые недостатки, существенно затрудняющие ее использование. К таким недостаткам можно отнести:

1. Малую защищенность системы от неквалифицированных действий пользователей.
2. Подверженность системы различного рода "взломам" при сетевом использовании и подверженность вирусам.
3. Неустойчивость работы системы, проявляющаяся в зависаниях и потере информации.
4. Большая стоимость лицензий на использование систем.
5. Закрытость операционной системы, затрудняющая написание учебных программ в ее среде и обучение.
6. Большие требования к возможностям компьютера (память, быстродействие).

7. Частая смена версий ОС (примерно каждые два года).

На данный момент наиболее востребованной ОС для кластеров является Linux в различных вариациях. Но и в линейках серверных ОС компании Microsoft появляются дистрибутивы, ориентированные на высокопроизводительные расчёты.

6.2. Менеджер ресурсов кластера

Менеджер ресурсов является самой важной частью любого вычислительного кластера. Именно он находится в его основе и определяет то, насколько эффективно он будет использоваться. Именно менеджер ресурсов управляет всей инфраструктурой кластера как то: очередь пользовательских задач, используемые вычислительные узлы, используемые процессоры на узлах, планировщик задач.

6.2.1. Наиболее известным менеджером ресурсов для кластеров под управлением ОС Linux является Torque.

Первый шаг к использованию вычислительных ресурсов узлов в глобальной среде грид – это их локальная интеграция в многопроцессорный комплекс, который используется в режиме пакетной обработки заданий. Для этой цели разработаны программные продукты, называемые Менеджерами Ресурсов (Resource Manager), кластерными системами управления (Cluster Management System) или Системами управления пакетной обработкой (СПО).

СПО широко применяются во многих вычислительных центрах коллективной обработки. Известно более двадцати СПО, из которых наиболее популярны свободно распространяемые Torque, PBS, SGE и Condor, а также коммерческие LoadLeveler и LSF.

Использование СПО в кластере позволяет перейти от работы с индивидуальными компьютерами, распределенными в локальной сети, к работе с единым многопроцессорным вычислительным комплексом в режиме пакетной обработки заданий. Пользователь может помещать задания в общую для комплекса очередь, используя единый интерфейс для запуска, модификации, снятия и получения информация о заданиях. СПО автоматически распределяет задания по узлам с учетом их загрузки, выполняет и доставляет результаты пользователю. Все СПО имеют богатый настраиваемый набор средств для управления процессом обработки заданий.

Все СПО имеют примерно одинаковые наборы функций, но отличаются по отдельным характеристикам:

- **PBS** - Portable Batch System. Обеспечивает управление заданиями в пакетном режиме в среде компьютеров с ОС Unix. Сегодня предлагается OpenPBS – свободно распространяемая версия и PBSPro – расширенная коммерческая версия. Sun Grid Engine (SGE). Семейство из нескольких различных вариантов СПО.
- Sun Grid Engine - свободно распространяемая версия, предназначенная для управления ресурсами одного проекта или подразделения. Основана на полной централизации обслуживания ресурсов и пользователей. Sun Grid Engine реализуется открытым сообществом разработчиков и спонсируется Sun Microsystems.
- Sun Grid Engine, Enterprise Edition (SGEEE) - коммерческая версия, предназначенная для управления ресурсами предприятий (campus grid) и способная обслуживать несколько независимых проектов и групп пользователей. SGEEE включает модуль для определения политики разделения ресурсов между независимо работающими пользователями. Для каждого пользователя определяется квота от общего количества ресурсов, в соответствии с которой происходит их распределение между запущенными заданиями.
- Condor. Свободно распространяемый продукт, позволяющий использовать в кластере машин с “хозяином”, который отдает свою машину на время, пока она простаивает. В связи с этим Condor имеет нескольких важных механизмов поддержки контрольных точек, рестарта и миграции заданий. В Condor развит весьма мощный язык описания ресурсов, позволяющий формально описать как требования к типам и объемам ресурсов со стороны задания, так и ограничения на доступ к ресурсам со стороны владельцев.
- LoadLeveler - коммерческий программный продукт компании IBM, предназначенный для пакетной обработки последовательных и параллельных (многопроцессорных) заданий на кластерах из вычислительных серверов. Система обеспечивает средства для подготовки, запуска и слежения за заданиями в режиме пакетной обработки в гетерогенной сети компьютеров.

- Текущая версия LoadLeveler Version 3.1 дополнена рядом новых характеристик, таких как новые планировщики; механизм поддержки контрольных точек; интеграция с AIX WorkLoad Manager; включает примеры сценариев.
- LSF (Load Sharing Facility) - коммерческая кластерная система компании Platform Computing Corporation для управления пакетной обработкой. Представляет собой полномасштабную систему, обеспечивающую все существенные требования к СПО: поддержку очередей заданий, сбор информации о наличии и занятости ресурсов в кластере, нахождение компьютеров с подходящими ресурсами для выполнения заданий, поддержку режима контрольных точек (checkpointing), миграцию заданий и др.
- MAUI - внешний планировщик, который может использоваться взамен штатных планировщиков для нескольких СПО: PBS, SGE, Loadleveler, LSF, Wiki. MAUI - открытый продукт, который отличается большим набором режимов (политик) планирования и наличием механизма предварительного резервирования. MAUI представляет большой интерес в связи с тем, что это единственная из свободно распространяемых СПО, способная обеспечивать автоматический запуск многопроцессорных заданий, избегая при этом неоправданного простоя ресурсов. Реализация основана на мощном алгоритме планирования Backfill.

6.2.2. Torque

(Terascale Open-source Resource and QUEue Manager) – Новая **версия PBS**, разработанная компанией Cluster Resources, Inc. на основе OpenPBS. Система обладает рядом дополнительных усовершенствований:

1. Улучшена масштабируемость (работа в среде до 2500 узлов).
2. Повышена устойчивость к сбоям (внесены дополнительные проверки).
3. Усовершенствован интерфейс Планировщика с целью его обеспечения дополнительной и более точной информацией.
4. Усовершенствована система записей в log файлах.

Система управления заданиями Torque предназначена для управления запуском задач на многопроцессорных вычислительных установках (в том числе кластерных). Она позволяет автоматически распределять вычислительные

ресурсы между задачами, управлять порядком их запуска, временем работы, получать информацию о состоянии очередей. При невозможности запуска задач немедленно, они ставятся в очередь и ожидают, пока не освободятся нужные ресурсы[6].

Torque главным образом используется на многопроцессорных вычислительных установках. Объединение ресурсов в вычислительных установках обычно уменьшает необходимость в постоянном управлении ресурсами для пользователей. Настроенная однажды правильно вычислительная установка абстрагируется от многих деталей, связанных с запуском и управлением заданиями. Пользователю обычно надо установить в параметрах лишь минимальные требования к задаче, и ему нет необходимости знать даже имена вычислительных узлов, на которых задача выполняется.

Система Torque состоит из нескольких демонов, выполняющих различные функции по управлению потоком заданий. Вычислительная установка обязана иметь главный узел (консоль кластера), на котором запущен демон pbs_server. Это основной демон - менеджер, собирающий информацию о структуре кластера и запущенных заданиях. В зависимости от необходимости или параметров системы главный узел может быть предназначен только для этого или же также исполнять роль других компонент системы. Например, он может быть так же вычислительным узлом кластера.

Роль вычислительных узлов - выполнять поставленные задачи. На каждом из них работает демон pbs_mom для того, чтобы начинать, прекращать и управлять поставленными в очередь задачами. Это единственный демон, который должен быть запущен на вычислительном узле кластера.

И наконец, демон pbs_sched. Этот демон занимается собственно планированием запуска и остановки задач. Он должен быть запущен на главном компьютере кластера.

6.2.3. Microsoft HPC Cluster Server

Программное обеспечение Microsoft HPC Pack представляет собой комплект **дополнительного программного обеспечения для Microsoft Windows HPC Server и** предназначен для управления задачами в рамках высокоскоростной вычислительной сети. Microsoft HPC Pack 2008 R2 помогает повышать эффективность работы пользователей и облегчающее администрирование кластеров.

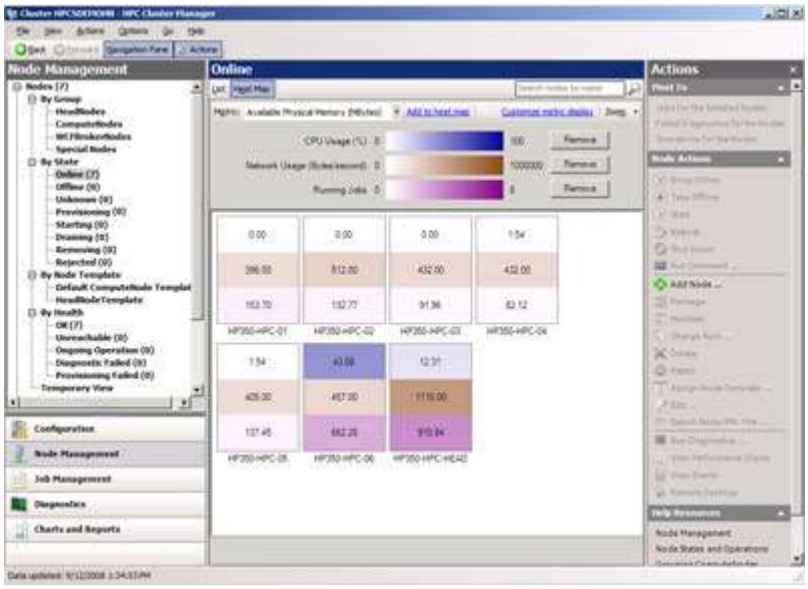


Рис.6.2 - Интерфейс менеджера Windows HPC Server 2008

Систему Windows HPC Server 2008 R2 можно эффективно использовать на кластерах с большим количеством узлов. В решении реализованы новая высокоскоростная технология NetworkDirect RDMA, эффективные и масштабируемые средства управления кластером, сервисно-ориентированная архитектура (SOA) инструмента планирования заданий и полная совместимость с инфраструктурой за счет соответствия спецификации High Performance Computing Basic Profile (HPCBP).

HPC Server 2008 R2 позволяет быстро и просто настраивать и запускать кластер. Текущий CTP включает ряд усовершенствований в инфраструктуре распределенного управления, включая возможность создавать карты нагрузок и использовать сторонние продукты для создания отчетов. Улучшенный планировщик задач информирует о выполнении задачи и позволяет осуществлять подключение/отключение дополнительных узлов к кластеру. Основные возможности менеджера:

- Быстрое развертывание и настройка кластера, инструментов мониторинга и планирования заданий.
- Использование специальных инструментов установки для сетей, удаленных систем, системы управления узлами и защиты кластера.
- Улучшенная интеграция с IT-инфраструктурой.
- Оптимизация существующей установки Active Directory для упрощения настроек аутентификации и защиты и сервисов Remote Installation Services (RIS) для удаленной инсталляции, Microsoft Systems Management Server (SMS) для управления обновлениями узлов, Microsoft Operations Manager

(MOM) для управления системами и заданиями, Microsoft Management Console (MMC) для встроенных системных инструментов.

Windows Compute Cluster Server (CCS), впервые выпущенный в июне 2006 года разработан для высокотехнологичных приложений, которые требуют кластерных вычислений. Издание разработано для развертывания на множестве компьютеров, которые собираются в кластер для достижения мощностей суперкомпьютера. Каждый кластер на Windows Compute Cluster Server состоит из одной или нескольких управляющих машин, распределяющих задания и нескольких подчиненных машин, выполняющих основную работу. Computer Cluster Server использует the Microsoft Messaging Passing Interface v2 (MS-MPI для связи между исполняющими машинами в сети-кластере. Он связывает части кластера вместе мощным коммутационным механизмом. API состоит из более чем 160 функций. MS MPI был разработан как совместимым с open source интерфейсом MPI2, который широко используется в высокопроизводительных вычислениях. За некоторыми исключениями по соображениям безопасности MS MPI покрывает функциональность MPI2 за исключением возможностей динамического порождения процессов.

Менеджер заданий в составе Windows CCS 2003 (Compute Cluster Job Manager) является приложением с графическим интерфейсом, которое обеспечивает доступ к планировщику заданий (Job Scheduler) для их создания, отправки на кластер и мониторинга исполнения. Менеджер заданий может быть установлен и исполняться на машине, не принадлежащей вычислительному кластеру, что дает возможность использовать кластер удаленно.

Графический интерфейс менеджера заданий состоит из

1. заголовка,
2. меню,
3. верхней панели отображения и
4. нижней панели отображения

Заголовок отображает имя машины, являющейся головным узлом кластера. По умолчанию, такой машиной является локальная машина (т.е., машина, где запущен менеджер заданий), имя которой отображается как localhost.

Меню состоит из пунктов File, View, Show и Help. С помощью File и View, можно подсоединиться к кластеру, создать задание, отослать его на исполнение и др. С помощью пункта Show можно просматривать очередь заданий на исполнение с

возможностью фильтрации этой очереди в соответствии с различными критериями.

В верхней панели отображается очередь заданий в табличной форме. Каждая строка таблицы представляет отдельное задание, а в столбцах отображаются статус задания, его свойства, а также статистика, относящаяся к этому заданию.

В нижней панели отображается список задач выбранного задания. Аналогично верхней панели, каждая строка представляет отдельную задачу, а в столбцах отображаются ее статус, свойства и статистическая информация, относящаяся к этому заданию.

Шаблоны заданий и задач

Менеджер заданий обеспечивает специальные шаблоны, с помощью которых могут быть составлены задания и задачи, сохранены на диске в виде XML-файлов для дальнейшего использования, а также отправлены на исполнение.

Интерфейс командной строки

Любая функция менеджера заданий имеет эквивалент в виде соответствующей командной строки. Все множество командных строк делится на 2 группы:

- операции пользователя: создание заданий, отправка их на кластер и управление ими;
- операции администратора: управление кластером как таковым.

Наличие интерфейса командной строки позволяет разрабатывать сложные сценарии (скрипты) для выполнения специальных работ на кластере и управления им.

К операциям пользователя относятся команды **job** и **task** с соответствующими параметрами.

1.
   - job new [job_terms] - создать задание
   - job add jobID [task_terms] - добавить задачи к заданию
   - job submit /id:jobid - отправить задание, созданное с помощью job new
   - job submit [job_terms] [task_terms] - отправить задание
   - job cancel jobID - снять задание
   - job modify [options] - модифицировать задание
   - job requeue [jobID] - заново поставить задание в очередь
   - job list - выдать список заданий, исполняющихся на кластере
   - job listtasks - выдать список задач задания
   - job view jobID - выдать сведения о задании

2.

- task view taskID - выдать сведения о задаче
- task cancel taskID - снять задачу
- task requeue taskID - заново поставить задачу в очередь

К операциям администратора относится команда **cluscfg**:

- cluscfg view - выдать сведения о кластере
- cluscfg listparams/setparams - просмотреть/установить конфигурационные параметры
- cluscfg listenvs/setenv - просмотреть/установить параметры окружения
- cluscfg delcreds/setcreds - удалить / установить полномочия пользователя

6.3. Интерфейс MPI, как основная среда межпроцессорной передачи данных

Взаимодействие между узлами вычислительного кластера происходит на уровне обмена сообщениями. Для этого используется интерфейс MPI. **Message Passing Interface** (MPI, интерфейс передачи сообщений) — программный интерфейс (API) для передачи информации, который позволяет обмениваться сообщениями между процессами, выполняющими одну задачу. Основные разработчики Уильямом Гроуппом, Эвином.

MPI является наиболее распространённым стандартом интерфейса обмена данными в параллельном программировании, существуют его реализации для большого числа компьютерных платформ. Используется при разработке программ для кластеров и суперкомпьютеров. Основным средством коммуникации между процессами в MPI является передача сообщений друг другу. Стандартизацией MPI занимается MPI Forum. В стандарте MPI описан интерфейс передачи сообщений, который должен поддерживаться как на платформе, так и в приложениях пользователя. В настоящее время существует большое количество бесплатных и коммерческих реализаций MPI. Существуют реализации для языков Фортран 77/90, Си и Си++.

В первую очередь MPI ориентирован на системы с распределенной памятью, то есть когда затраты на передачу данных велики, в то время как OpenMP ориентирован на системы с общей памятью (многоядерные с общим кэшем). Обе технологии могут использоваться совместно, дабы оптимально использовать в кластере многоядерные системы.

Интерфейс MPI стал фактически стандартом в сфере взаимодействия между узлами кластера. Для различных платформ существуют различные варианты реализации интерфейса MPI:

- MPI.NET — реализация MPI для Windows.
- MPICH — самая распространённая бесплатная реализация, работает на UNIX-системах и Windows NT.
- MPI/PRO for Windows NT — коммерческая реализация для Windows NT.
- Intel MPI — коммерческая реализация для Windows / Linux.
- Microsoft MPI входит в состав Compute Cluster Pack SDK. Основан на MPICH2, но включает дополнительные средства управления заданиями. Поддерживается спецификация MPI-2.
- HP-MPI — коммерческая реализация от HP.
- SGI MPT — платная библиотека MPI от SGI.
- Mvapich — бесплатная реализация MPI для Infiniband.
- Open MPI — бесплатная реализация MPI, наследник LAM/MPI.
- Oracle HPC ClusterTools — бесплатная реализация для Solaris SPARC/x86 и Linux на основе Open MPI.
- MPJ — MPI for Java.

Если учитывать то, что операционные системы семейства Linux являются наиболее распространенными на вычислительных кластерах, то наиболее распространёнными и развиваемыми являются реализации MPI именно для этих платформ. Однако не стоит забывать, что и компания Microsoft стремительно развивает свои технологии в области высокопроизводительных вычислений. ***Рассмотрим наиболее популярные из реализаций MPI.***

6.3.1. MPICH

Message Passing Interface Chameleon — это одна из самых первых разработанных библиотек MPI. На её базе было создано большое количество других библиотек как бесплатных, так и коммерческих. В настоящее время существует две ветви исходных кодов: MPICH1 и MPICH2. Разработка ветви MPICH1 заморожена. Ветвь MPICH2 активно разрабатывается в Аргонской лаборатории, с участием IBM, Cray, SiCortex, Microsoft, Intel, NetEffect, Qlogic, Myricom, Ohio state university, UBC.

Рассмотрим схему работы программы с использованием директив MPICH. Менеджер процессов smpd, который представляет собой системную службу (сервисное приложение). Менеджер процессов ведёт список вычислительных

узлов системы, и запускает на этих узлах MPI-программы, предоставляя им необходимую информацию для работы и обмена сообщениями. MPICH состоит из следующих компонентов:

- Заголовочные файлы (.h) и библиотеки стадии компиляции (.lib), необходимые для разработки MPI-программ.
- Библиотеки времени выполнения, необходимые для работы MPI-программ.
- Дополнительные утилиты, необходимые для настройки MPICH и запуска MPI-программ.

Менеджер процессов является основным компонентом, который должен быть установлен и настроен на всех компьютерах сети (библиотеки времени выполнения можно, в крайнем случае, копировать вместе с MPI-программой). Остальные файлы требуются для разработки MPI-программ и настройки некоторого «головного» компьютера, с которого будет производиться их запуск.
Менеджер работает в фоновом режиме и ждёт запросов к нему из сети со стороны «головного» менеджера процессов (по умолчанию используется сетевой порт 8676). Чтобы как-то обезопасить себя от хакеров и вирусов, менеджер требует пароль при обращении к нему. Когда один менеджер процессов обращается к другому менеджеру процессов, он передаёт ему свой пароль. Отсюда следует, что нужно указывать один и тот же пароль при установке MPICH на компьютеры сети.

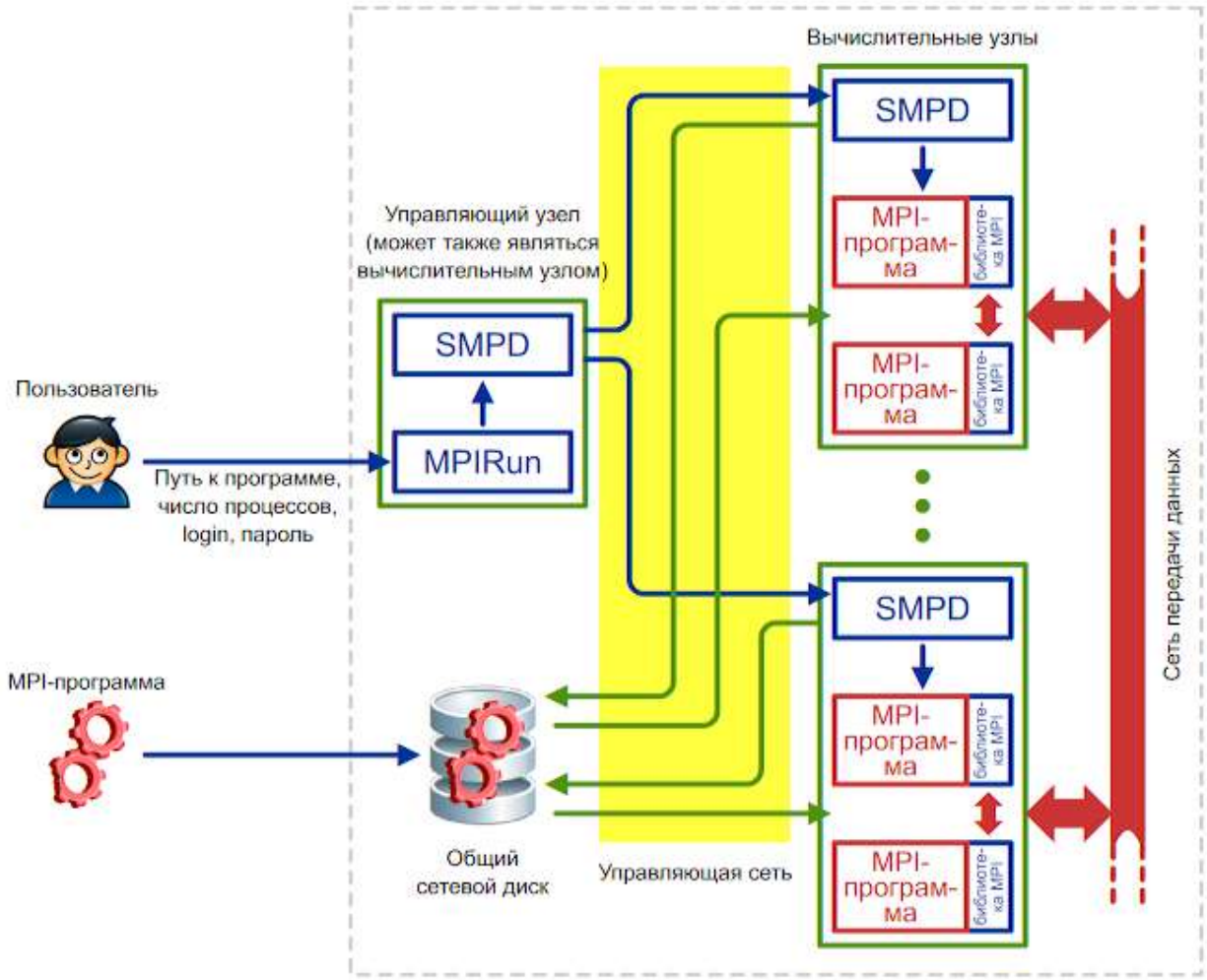


Рис. 6.2. - Схема работы MPICH на кластере

В современных кластерах сеть передачи данных обычно отделяется от управляющей сети.

Запуск MPI-программы производится следующим образом:

1. Пользователь с помощью программы Mpirun (или Mpiexec, при использовании MPICH2 под Windows) указывает имя исполняемого файла MPI-программы и требуемое число процессов. Кроме того, можно указать имя пользователя и пароль: процессы MPI-программы будут запускаться от имени этого пользователя.
2. Mpirun передаёт сведения о запуске локальному менеджеру процессов, у которого имеется список доступных вычислительных узлов.
3. Менеджер процессов обращается к вычислительным узлам по списку, передавая запущенным на них менеджерам процессов указания по запуску MPI-программы.

4. Менеджеры процессов запускают на вычислительных узлах несколько копий MPI-программы (возможно, по несколько копий на каждом узле), передавая программам необходимую информацию для связи друг с другом.

Очень важным моментом здесь является то, что перед запуском MPI-программа не копируется автоматически на вычислительные узлы кластера. Вместо этого менеджер процессов передаёт узлам путь к исполняемому файлу программы точно в том виде, в котором пользователь указал этот путь программе Mpirun. Это означает, что если вы, например, запускаете программу C:\programm.exe, то все менеджеры процессов на вычислительных узлах будут пытаться запустить файл C:\ programm.exe. Если хотя бы на одном из узлов такого файла не окажется, произойдёт ошибка запуска MPI-программы.

Чтобы каждый раз не копировать вручную программу и все необходимые для её работы файлы на вычислительные узлы кластера, обычно используют общий сетевой ресурс. В этом случае пользователь копирует программу и дополнительные файлы на сетевой ресурс, видимый всеми узлами кластера, и указывает путь к файлу программы на этом ресурсе. Дополнительным удобством такого подхода является то, что при наличии возможности записи на общий сетевой ресурс запущенные копии программы могут записывать туда результаты своей работы.

Работа MPI-программы происходит следующим образом:

Программа запускается и инициализирует библиотеку времени выполнения MPICH путём вызова функции MPI_Init.

Библиотека получает от менеджера процессов информацию о количестве и местоположении других процессов программы, и устанавливает с ними связь.

После этого запущенные копии программы могут обмениваться друг с другом информацией посредством библиотеки MPICH. С точки зрения операционной системы библиотека является частью программы (работает в том же процессе), поэтому можно считать, что запущенные копии MPI-программы обмениваются данными напрямую друг с другом, как любые другие приложения, передающие данные по сети.

Консольный ввод-вывод всех процессов MPI-программы перенаправляется на консоль, на которой запущена Mpirun. Насколько я понимаю, перенаправлением ввода-вывода занимаются менеджеры процессов, так как именно они запустили копии MPI-программы, и поэтому могут получить доступ к потокам ввода-вывода программ.

Перед завершением все процессы вызывают функцию MPI_Finalize, которая корректно завершает передачу и приём всех сообщений, и отключает MPICH.

Все описанные выше принципы действуют, даже если вы запускаете MPI-программу на одном компьютере.

6.3.2. MPI.NET

Message Passing Interface for .NET – ***это высокопроизводительное и простое в реализации применение MPI***. MPI.NET обеспечивает поддержку для всех языков .NET (в частности, C#) и включает в себя значительные расширения, такие как автоматическая сериализация объектов, что обеспечивает написание программ для кластеров. Особенностью разработки, используя MPI.NET является отсутствие необходимости изучать новые языки программирования т.к. поддерживаются все языки платформы .NET а также поддержка MPI.NET со стороны производителя операционной системы. Пакет MPI.NET может быть установлен на операционные системы Windows Compute Cluster Server, Windows HPC Server 2008, Windows XP, Windows Vista, Windows 7. Для тестирования и выполнения программ требуется установить пакет программ Windows HPC Server и MPI.NET SDK. Отладка приложений производится с помощью набора утилит, которые позволяют эмулировать кластер. В частности компиляция программ осуществляется с помощью утилиты mpiexec. Эта утилита входит в пакет MPI.NET SDK. В качестве аргументов утилита принимает число эмулируемых процессов, имена хостов, рабочую директорию и имя файла.

6.4. Система мониторинга кластера

Большее внимание необходимо уделять системам мониторинга оборудования вычислительного кластера. Система мониторинга – это контролирующее звено. При обнаружении нового оборудования, неисправностей в работе сети или любых других изменениях она должна уведомить администратора. К основным задачам системы мониторинга кластера можно отнести:

- Уровень загрузки процессоров вычислительного кластера.
- Уровень загрузки память на узлах.
- Контроль трафика по сети Ethernet.
- Контроль трафика по сети InfiniBand.
- Уровень загрузки жестких дисков на вычислительных узлах.
- Уровень загрузки кластера каждым из пользователей.
- Доступность узлов вычислительного кластера.
- Отчеты по температурным и другим показателям.

Наиболее известным и часто используемой системой мониторинга кластера для

Linux является Gunglia. При мониторинге кластера под ОС Windows часть задач берет на себя система управления кластером и сама операционная система.
Необходимо иметь резервные каналы доступа к вычислительным узлам кластера. Для Linux кластеров основным протоколом доступа является ssh. В качестве резервных каналов связи можно использовать протокол VNC или программное обеспечение ServNet.

6.5. СОВРЕМЕННЫЕ НАПРАВЛЕНИЕ РАЗВИТИЯ ВЫЧИСЛИТЕЛЬНЫХ КЛАСТЕРОВ

Вычислительные кластеры претерпевают постоянные изменения в своей архитектуре, в специфики работы в программном обеспечении. Уже сегодня можно основными направлениями развития рынка HPC является повышение производительности и уменьшение затрат на эксплуатацию вычислительного кластера. Эти показатели могут быть достигнуты при использовании:

- Высокоскоростной среды передачи данных. Как правило, в современных суперкомпьютерах используется сеть InfiniBand высокоскоростными коммутирующими устройствами. Это позволяет уменьшить латентность при проведении расчётов.
- Многосокетных узлов вычислительного кластера. Передача данных между процессорами на одном узле вычислительного кластера происходит с меньшей задержкой чем передача того же объема данных между узлами.
- Использование надежной, рассчитанной на обработку большего числа операций ввода-вывода, подсистемы хранения данных.
- Бездисковых серверов. За счет этого можно увеличить полезную площадь материнской платы и использовать дополнительные вычислительные мощности.
- Гибридных кластеров на GPU. Использование графических адаптеров для высоко сложных расчётах позволило увеличить производительность системы на один – два порядка. В сочетании с меньшей стоимостью графических карты и низким энергопотреблением, графические карты становятся необходимыми для внедрения на оборудовании HPC. Основным производителем является NVidia с линейкой продуктов Tesla.

Список использованной литературы

1. http://www.supercomputers.ru [Электронный ресурс]. Режим доступа свободный.
2. http://www.intuit.ru [Электронный ресурс]. Режим доступа свободный.
3. Корняков. К.В. “Построение и использование кластерных вычислительных систем”. - Нижний Новгород, 2007, 98 с.
4. “Руководство по работе на вычислительном кластере”. - Минск, БГУ 2004г. – 172с.
5. http://www.infinibandta.org/ [Электронный ресурс]. Режим доступа свободный.
6. http://www.adaptivecomputing.com/products/open-source/torque [Электронный ресурс]. Режим доступа свободный.

Вопросы к главе 6.

1. Программное обеспечение вычислительных кластеров. Операционные системы. Менеджер ресурсов кластера
2. Интерфейс MPI, как основная среда межпроцессорной передачи данных
3. Система мониторинга кластера

Глава 7. Грид-системы

Вторая концепция высокопроизводительных вычислений, рассматриваемая в рамках данного курса, - это грид-системы. **От суперкомпьютеров – к распределенным вычислительным комплексам.** Несмотря на появление относительно недорогих кластерных архитектур и стандартизацию соответствующих интерфейсов параллельного программирования для них, суперкомпьютерные системы на практике доступны далеко не каждому пользователю. Развитие концепции Грид первого поколения лишь отчасти способствовало улучшению ситуации. Это связано с тем, что Грид первого поколения был ориентирован, в первую очередь, на обеспечение пользовательским приложениям доступа к распределенным вычислительным ресурсам. Вопросы подготовки и настройки программного обеспечения для исполнения в Грид остаются за пользователем; при этом эффективная работа параллельной программы при запуске на целевой системе в Грид также не гарантируется. Кардинальное решение данной проблемы стало возможным лишь с появлением концепции Грид второго поколения [4]. Грид второго поколения ориентирован на консолидацию не столько распределенных вычислительных ресурсов, сколько сервисов – прикладных программ, установленных на вычислительных системах в рамках распределенной среды, и поддерживающих единый интерфейс взаимодействия [5]. В силу того, что пользователь получает удаленный доступ к сервису как к программно-аппаратному решению, это заведомо снимает вопросы совместимости внутренних интерфейсов и эффективности параллельного исполнения.

1. Вычислительные кластеры

Вычислительные кластеры зачастую являются наиболее эффективным инструментом высокопроизводительных вычислений. Однако, как былосказано выше, они имеют целый ряд недостатков. Поэтому разрабатывались и кардинально другие подходы к организации высокопроизводительных

вычислений.

7.1. Идея грид — 1: проект Beowulf

Как отмечалось ранее, суперкомпьютеры стоят очень дорого (в частности, их создание и обслуживание), поэтому они доступны лишь ведущим научным учреждениям и крупным компаниям. При этом, необходимость в высокопроизводительных вычислениях есть во многих организациях. Следовательно, актуальным является достижение высокой производительности по низкой цене.

Одно из решений этой проблемы — это создание «суперкомпьютера» из нескольких обычных компьютеров — имеющих доступные и дешевые компоненты — соединенных стандартной сетью (Ethernet).

Какие проблемы вычислительных кластеров это решает?

1. высокая стоимость создания (оборудование + инфраструктура);
2. плохая масштабируемость.

Во главу угла ставится доступность: дешевизна в создании и сопровождении.

Одним из первых известных решений в рамках этой концепции является проект Beowulf, который стартовал летом 1994 г. в научно-космическом центре NASA. Первой установкой стал 16-процессорный кластер со следующими характеристиками: процессор 486DX4/100MHz, 16MB памяти и 3 сетевых адаптера на каждом узле, 3 параллельных Ethernet-кабеля по 10Mbit. Этот кластер, получивший имя "Beowulf", создавался как вычислительный ресурс проекта Earth and Space Sciences Project (ESS).

Изначально термин "Beowulf" возник как собственное имя Linux-кластера в GSFC NASA. Затем он стал применяться ко всем аналогичным кластерным системам (Beowulf- кластер).

Avalon — другой пример Beowulf-кластера. В 1998 г. в Лос-аламосской национальной лаборатории США был запущен «суперкомпьютер Avalon» — Linux-кластер на базе процессоров DEC Alpha. Avalon изначально состоял из 68 процессоров, затем был расширен до 140. На каждом узле было установлено 256MB оперативной памяти, жесткий диск на 3.2GB, сетевой адаптер. Узлы соединены между собой с помощью четырех 36-портовых коммутаторов Fast Ethernet и расположенного "в центре" 12-портового коммутатора Gigabit Ethernet.

Стоимость Avalon — $313 тыс., производительность по LINPACK (47.7 GFLOPS) позволила ему занять 114 место в 12-й редакции списка Top500 (рядом со 152-процессорной системой IBM SP2). 70-процессорная конфигурация Avalon показала такую же производительность, как 64-процессорная система SGI Origin2000/195MHz стоимостью $1,8 млн.

На конференции SuperComputing'98 создатели Avalon представили доклад "Avalon: An Alpha/Linux Cluster Achieves 10 Gflops for $150k" и заслужили премию по показателю цена/производительность ("1998 Gordon Bell

Price/Performance Prize").

Avalon активно использовался в астрофизических, молекулярных и других научных вычислениях.

Ранее подобные системы назывались «кластерами», однако на сегодня такое именование уже является неверным. Все дело в каналах связи, обеспечивающих взаимодействие вычислителей. В конце 90-х годов XX века интерконнект вычислительных кластеров был недостаточно развит, и его можно было по техническим характеристикам сравнивать с Ethernet-сетью, доступной на широком рынке. Однако с тех пор разрыв в характеристиках очень сильно увеличился: распространенные реализации Ethernet уже не могут конкурировать со специализированным интерконнектом, а оснащение Beowulf- кластера специализированными решениями существенно увеличивает стоимость системы.

Системы типа Beowulf были призваны решить проблему высокой стоимости суперкомпьютеров. Однако сейчас Beowulf-кластеры, как правило, не используются. Это связано с тем, что их недостатки значительно перевешивают преимущество низкой стоимости:

1. *плохая производительность*: основная проблема — плохой интерконнект, имеющий малую пропускную способность и высокие задержки;
2. *недостижимость показателей современных кластеров* на основных классах решаемых задач — следствие ограничений интерконнекта.

Кроме того, Beowulf-кластерам присущи многие из тех проблем, которые имеют вычислительные кластера:

1. *большое энергопотребление*: еще более острая проблема, т.к. вычислительные узлы Beowulf-системы не спроектированы для работы в условиях высокой плотности размещения, им требуется больше энергии на выполнение вычислительных задач;
2. *необходимость охлаждения*: также еще более острая проблема, так как энергопотребление Beowulf выше энергопотребления вычислительных кластеров той же производительности, а высокая плотность неспециализированных вычислительных узлов мешает эффективному охлаждению;
3. *сложность обслуживания* как следствие большого числа вычислительных узлов и электронных компонент;
4. *плохая управляемость* как следствие сложности обслуживания;
5. *ограничение параллелизации* вследствие закона Амдала;
6. *проблемы алгоритмизации и программирования*.

При этом, основной проблемой становится надежность электронных компонент: используемые в вычислительных кластерах компоненты, как правило, имеют значительно более высокую надежность, чем доступные на

широком рынке.

7.2. Идея грид — 2: кластерные Грид

Другая идея грид исходит из следующей проблемы: масштабирование ресурсов кластера чересчур дорого, причем, закон Амдала снижает эффект от масштабирования.

Задача: необходимо обеспечить суперкомпьютерными ресурсами большое количество пользователей и сбалансировать нагрузку на существующие кластеры.

Решение: объединяем географически удаленные вычислительные кластеры в единый ресурс — с единой точкой входа и общей системой управления задачами.

Какие проблемы это решает?

7.3. большое энергопотребление (вычислители + охлаждение);

7.4. необходимость охлаждения (экология и инженерные проблемы);

7.5. плохая масштабируемость.

Ярким примером кластерной грид является проект EGEE.

Проект выполняется консорциумом из 70 институтов в 27 странах, объединенных в

региональные гриды. Научно-исследовательское сообщество Европы получило в распоряжение общую инфраструктуру высокопроизводительных вычислений, услуги которой — круглосуточный доступ к крупнейшим вычислительным ресурсам, не зависящий от месторасположения потребителей.

С российской стороны в EGEE принимают участие 8 институтов Москвы, Московской области и Санкт-Петербурга: НИИЯФ МГУ, ИТЭФ, ИФВЭ, ИМПБ РАН, ОИЯИ, ПИЯФ РАН, РНЦ КИ, а также Институт прикладной математики им. М.В.Келдыша РАН (ИПМ РАН).

Цель проекта EGEE — обработка данных с Большого адронного коллайдера (Large Hadron Collider – LHC), который построен в CERN (Женева). Эксперименты на коллайдере LHC порождают потоки данных с экстремальными характеристиками: со скоростью 100 Мбайт/сек и общим объемом несколько петабайт в год. Обработка таких потоков невыполнима традиционными способами, поэтому возникла идея делать это на Грид в распределенном режиме.

Задача EGEE не ограничена ядерной физикой и состоит в том, чтобы реализовать потенциал Грид для многих других научных и технологических областей. На очереди — биоинформатика и здравоохранение.

К моменту окончания проекта (2010 г.) EGEE представлял собой общемировую инфраструктуру с 200,000 CPU, расположенных в более чем 300 центрах по всему миру.

7.3. Идея грид — 3: volunteer computing

Наконец, третья концепция грид родилась, исходя из следующих

соображений. Суперкомпьютеры в силу высокой цены доступны лишь ведущим научным учреждениям и крупным компаниям, но необходимость в высокопроизводительных вычислениях есть у многих. В организации большое количество персональных компьютеров, вычислительные ресурсы которых не используются в полной мере.

Задача: создать грид типа Beowulf, при этом нельзя перемещать компьютеры физически, и необходимо, чтобы пользователи могли использовать персональные компьютеры в своей работе.

Решение: необходимо создать грид-сеть из персональных компьютеров, подключенных к Интернет. Управлением Грид, распределением заданий и использованием вычислительных ресурсов занимается специальная программная система.

Какие проблемы это решает?

1. высокая стоимость создания (оборудование + инфраструктура): все вычислительное оборудование принадлежит его владельцам, соответственно, на них же ложится необходимость обслуживания этого оборудования;
2. большое энергопотребление (вычислители + охлаждение);
3. необходимость охлаждения (экология и инженерные проблемы);
4. плохая масштабируемость;
5. высокая вероятность отказа компонент;
6. сложность обслуживания.

Хорошим примером подобной системы является проект Folding@HOME, направленный на решение задачи предсказания пространственной структуры белков (см. Лекцию 1). В рамках этого проекта все вычислительные задания и результаты вычислений передаются по сети Интернет. К 9 апреля 2009 г. пиковая производительность проекта превысила 5 PFLOPS — в проекте участвовали около 400,000 активных вычислителей, расположенных по всему миру. Список поддерживаемых вычислителей включает в себя персональные компьютеры, видеокарты (GPU), Sony PlayStation и кластеры.

Такая концепция вычислительной системы получила название Desktop Grid.

Desktop Grid – это грид-система, объединяющая неспециализированные вычислители (персональные компьютеры, ноутбуки, смартфоны) и использующая их временно свободные вычислительные ресурсы (idle computing time) для выполнения вычислительноемких расчетов.

Интерес к такой технологии связан с огромными потенциальными вычислительными ресурсами, скрытыми в персональных компьютерах.

В мире более 1 миллиарда персональных компьютеров. При этом средняя загрузка процессора составляет всего 10-15%. Назначение Desktop Grid — использовать с пользой свободные вычислительные ресурсы.

Потенциальная пиковая производительность Desktop Grid составляет 120

Exaflops = 2200 компьютеров Tianhe-2 (I место в TOP500 суперкомпьютеров, июнь 2014 г.). При этом, в соответствии с прогнозами суперкомпьютер достигнет пиковой производительности в 1 Exaflops в 2018-2020 гг.

7.4. Что такое грид

Грид-вычисления — это форма распределенных вычислений, в которой «виртуальный суперкомпьютер» представлен в виде кластера соединенных с помощью сети слабосвязанных компьютеров, работающих вместе для выполнения огромного количества заданий.

Грид — это группа географически распределенных вычислителей, объединенных низкоскоростными каналами связи и представляющая с точки зрения пользователя единый аппаратный ресурс.

Некоторые ученые полагают также, что грид — это дальнейшая эволюция Интернета, открывающая доступ к мощностям десятков тысяч компьютеров по всему миру. Если задачу можно разбить на части, то решать ее можно на любом компьютере, неважно — по соседству или в другой стране физически находится ресурс: пользователь может находиться в одном месте, базы данных в другом, задание выполняться — в третьем. Важно иметь каналы связи между отдельными элементами сети.

Грид — это система, которая[5]:

- координирует использование ресурсов при отсутствии централизованного управления
  данными ресурсами.
- использует стандартные, открытые, универсальные протоколы и интерфейсы.
- нетривиальным образом обеспечивает высококачественное обслуживание.

7.5. Сервисы грид

Open Grid Forum (OGF) — это сообщество работников науки, бизнеса и государства, созданное для продвижения грид-технологий за счет разработки открытых стандартов взаимодействия управляющего ПО грид. OGF опубликовал более 100 документов — спецификаций и описания опыта использования грид-систем. Спецификации OGF стандартизируют вопросы взаимодействия: назначение заданий на различных платформах, доступ к распределенным базам данных, файлам и XML-репозиториям, управление развертыванием приложений в распределенной среде и предоставление унифицированного API для грид-приложений.

Членами OGF являются более 40 организаций, включая такие крупные компании, как IBM, Microsoft и Oracle.

---

5 Ян Фостер, «Что такое грид?», 2002 г.

Рис. 7.1. Стандарты грид-систем

Классические грид-системы, в соответствии со стандартом, должны обладать широким набором предоставляемых сервисов. Наиболее важные из них перечислены ниже.

*Infrastructure services
- Именование ресурсов
- Взаимодействие ресурсов
- Отражение характеристик ресурсов

*Execution Management services
- Описание заданий
- Управление запуском заданий
- Управление выполнением заданий
- Составление расписаний

*Data services
- Хранение данных
- Доступ к данным
- Обновление данных
- Перемещение данных
- Механизмы запросов к данным

- Преобразование данных

*Resource Management services
- Мониторинг физических и логических ресурсов
- Резервирование ресурсов

* Security services
- Аутентификация
- Авторизация
- Аудит
- Защита личной информации

* Self-management services
- Механизмы самоконфигурации
- Механизмы самоисправления
- Механизмы самооптимизации

* Information services
- Обнаружение служб и ресурсов
- Обмен сообщениями
- Логгирование
- Мониторинг

В целом, грид-системы на текущий момент развиваются по трем основным направлениям:

1. создание универсальных сред: формируются глобальные полигоны, объединяющие в рамках высокоскоростных сетей значительные распределенные ресурсы. Реальные подобные системы крайне тяжелы в установке, администрировании и сопровождении; организация расчетов требует привилегированных полномочий, а многие компьютерные платформы не поддерживаются. Примером работ в данном направлении является инфраструктура EGEE.
2. Универсальность среды заменяет четкая ориентация на конкретные задачи — создание специализированных сред для решения заранее определенного набора многократно решаемых «тяжелых» вычислительных задач (специализированные вычислительные порталы). Это направление намного проще реализовать на практике, чем первое, однако в каждом случае среда жестко ориентирована на решение только одной конкретной задачи, не предоставляя средств решения других задач. Примеры систем: SETI@HOME, GIMPS, Folding@home и др.
3. Разработка инструментария быстрого содания распределенной

вычислительной среды, объединяющей все доступные вычислительные ресурсы. Инструментарий универсальный, позволяет решать широкий класс задач, прост в освоении и дает возможность быстро развернуть расчет на всех основных типах компьютерных систем с использованием полномочий обычных пользователей. Эффективность, легкость в установке, освоении и использовании, возможность задействовать все основные типы компьютеров, работа через Интернет, поддержка вычислительных сред со сверхтерафлопсным уровнем производительности — это основные преимущества таких систем. Минусом подхода является то, что не удается обеспечить большого числа сервисов грид.

7.6. Приложения для грид

Из-за слабых каналов связи грид-системы имеют существенные ограничения по классам решаемых задач. Так, требования к задаче, решаемой в грид, следующие:

– возможность разбиения на множество независимых подзаданий

– много вычислений, мало (или нет совсем) обменов данными

«Идеальные» Грид-приложения:

- переборные задачи;
- проведение моделирования при больших количествах различных наборов исходных данных;
- моделирование методом Монте-Карло

Метод Монте-Карло — общее название группы численных методов, основанных на получении большого числа реализаций стохастического (случайного) процесса, который формируется таким образом, чтобы его вероятностные характеристики совпадали с аналогичными величинами решаемой задачи. Используется для решения задач в различных областях физики, химии, математики, экономики, оптимизации, теории управления и др.

7.7. Промежуточное ПО управления грид

Промежуточное программное обеспечение грид обеспечивает пользователю «прозрачность» использования распределенных ресурсов.

Основные функции:

- предоставление интерфейса взаимодействия;
- планировщик (расписание заданий);
- управление подзаданиями;
- мониторинг состояния грид;
- обеспечение доступа к данным;
- обеспечение безопасности;
- управление ресурсами.

Пакет Globus

Разработку ведет Globus Alliance, первые версии относятся к 1996 г. Globus считается стандартом де-факто промежуточного ПО Грид.

Крупные проекты, использующие Globus Toolkit: (European DataGrid; высокоскоростная межуниверситетская сеть GriPhyN; Particle Physics Data Grid; National Virtual Observatory (США).

ARC

The Advanced Resource Connector — промежуточное ПО грид, разрабатываемое скандинавской группой NorduGrid.

При реализации использованы службы и библиотеки пакета Globus Toolkit, а также дополнительно разработанные модули Grid Manager и User Interface.

Достоинства: ARC является «неагрессивной», надежной, отказоустойчивой программной системой, считается легким и гибким в установке, настройке, управлении, расширении, программировании клиентов и служб. Поддерживается переносимость между множеством операционных систем и систем локального управления ресурсами.

Крупные проекты, использующие ARC: Шведский национальный грид (Swegrid); Датский центр научных вычислений (DCSC); Североевропейский центр DataGrid (NDGF);

Condor

Разработка Condor Research Project из University of Wisconsin-Madison (США).

Condor предоставляет механизм управления заданиями, планировщик, схему приоритетов, мониторинг и управление ресурсами.

Первая Грид на базе Condor была создана в UW-Madison Department of Computer Sciences 20 лет назад, где управляет более 1000 рабочих станций.

Согласно статистике, за один обычный день Condor получал ресурсы более 650 CPU-дней.

Condor может взаимодействовать с грид на базе Globus.

gLite

Промежуточное ПО грид, разрабатываемое в рамках проекта EGEE (Enabling Grids for E-science in Europe). gLite разрабатывается в первую очередь для платформ Scientific Linux и Debian. Версии для других платформ появляются значительно позже.

В основу gLite легли некоторые службы Globus Toolkit и Condor, а также новые компоненты более высокого уровня. В частности, реализованная подсистема пользовательского интерфейса gLite считается одним из наиболее полнофункциональных грид-решений, хорошо совместимым с другими грид-службами. Система является модульной, считается сложной в установке, настройке и использовании.

UNICORE

Uniform Interface to Computing Resources – промежуточное ПО грид, разрабатываемое на базе Суперкомпьютерного центра г. Юлих (Германия).

UNICORE реализован на Java, в состав входят несколько реализаций клиентов, предоставляющих разные функции, и библиотека для разработки клиентских приложений.

В отличие от большинства конкурентов, при разработке UNICORE не использовались инструменты пакета Globus Toolkit. Реализации базовых грид-служб и интерфейсов и модели безопасности различаются, поэтому компоненты UNICORE и Globus Toolkit являются плохо совместимыми.

Достоинства: считается легким и гибким в установке, настройке, управлении, расширении. Поддерживается переносимость между множеством ОС.

Крупные проекты, использующие UNICORE: DEISA, DGI, VIOLA, Chemomentum, OMII-Europe, A-WARE, PHOSPHORUS, D-Mon.

BOINC

The Berkeley Open Infrastructure for Network Computing (BOINC) – открытая (Open Source) программная платформа для организации Desktop Grid (в частности, volunteer computing). Изначально разработана в Университете Беркли для проекта SETI@home, сейчас
– стандарт де-факто для Desktop Grid.

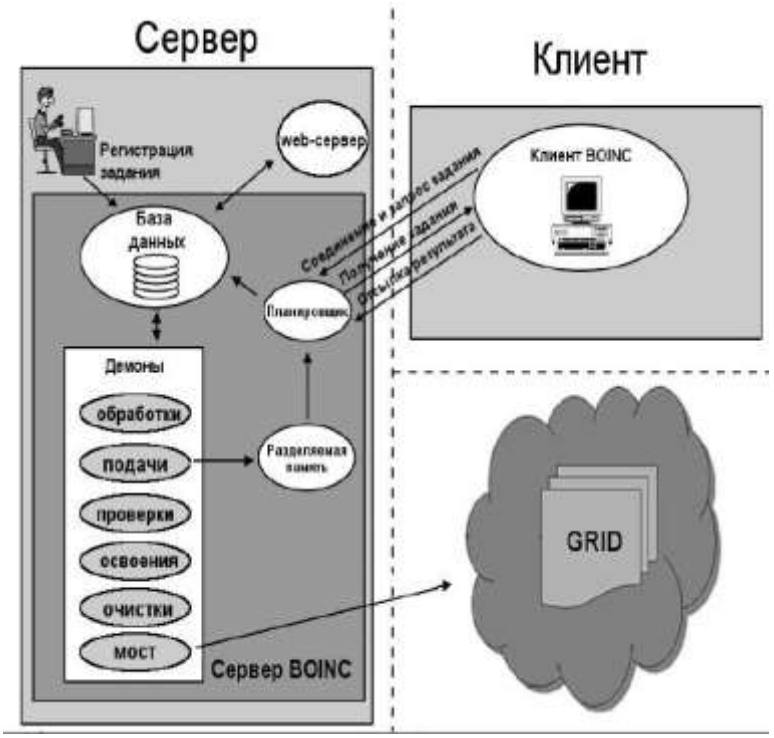


Рис. 7.2 - Архитектура BOINC

7.8. Проблемы и ограничения грид

Далее перечислены наиболее существенные проблемы грид-систем.

1. Высокая вероятность отказа узлов

- вероятность выхода из строя отдельного узла выше, чем в кластере;
- узлы могут быть недоступны в результате сбоя в сети;
- узлы могут быть временно выключены или выведены из состава Грид;

2. Эффективно решаться могут только задачи с очень хорошей степенью параллелизации.

При эксплуатации Грид существенные накладные расходы идут на формирование подзаданий, их назначение, обмени информацией с вычислительным узлом. В случае, когда для решения задачи необходимо согласование результатов нескольких вычислений, большие затраты времени будут уходить на коммуникации.

3. Сильная неоднородность узлов грид: приложения и ПО управления должны работать с разными

- физическими архитектурами;
- аппаратным обеспечением;
- операционными системами;
- программным окружением.

4. Сложность управления грид:

- большое количество вычислительных узлов;
- различные физические, аппаратные и программные архитектуры;
- большое количество пользователей;
- большое количество приложений;
- огромное количество заданий;
- разнообразные сервисы грид.

5. Возможность получения неправильных результатов: всегда есть вероятность того, что вычислительный узел вернет неправильный результат
- из-за программного/аппаратного сбоя;
- по злому умыслу.

Термин «грид» был введен в обращение Яном Фостером в начале 1998 го- да публикацией книги «Грид. Новая инфраструктура вычислений» [27]:

> ***Грид*** *– это система, которая координирует распределенные ресурсы посредством стандартных, открытых, универсальных протоколов и интерфейсов для обеспечения нетривиального качества обслуживания (QoS – Quality of Service).*

Хотя в последнее десятилетие базовая идея грид не претерпела существенных изменений, всеобъемлющего определения грид не существует до сих пор [62].

7.9. Архитектура Грид

Основной идеей, заложенной в концепции грид-вычислений, является централизованное удаленное предоставление ресурсов, необходимых для решения различного рода вычислительных задач. В каком-то смысле, концепция грид-вычислений идея рифмуется с концепцией электросети (англ. Power Grid): нам не важно, откуда к нам в розетку приходит электричество. Независимо от этого мы можем подключить к электросети утюг, компьютер или стиральную маши- ну. Также и в идеологии грид: мы можем запустить любую задачу с любого компьютера или мобильного устройств на вычисление, ресурсы же для этого вычисления должны быть автоматически предоставлены на удаленных высокопроизводительных серверах, независимо от типа нашей задачи.

С более практической точки зрения, основная задача, лежащая в основе концепции грид, это согласованное распределение ресурсов и решение задач в условиях динамических, многопрофильных *виртуальных организаций*. Распределение ресурсов, в котором заинтересованы разработчики грид, это не обмен

файлами, а прямой доступ к компьютерам, программному обеспечению, данным и другим ресурсам, которые требуются для совместного решения задач и стратегий управления ресурсами, возникающих в промышленности, науке и технике. *Виртуальной организацией (ВО)* называют ряд отдельных людей или учреждений, объединенных едиными правилами коллективного доступа к распределенным вычислительным ресурсам [31]. Для организации работы в рамках таких ВО возникает необходимость в следующем:

1. в гибких механизмах разделения ресурсов, начиная от клиент-серверных заканчивая одноранговыми;
2. в развитой системе контроля используемых ресурсов, включая контроль над мелкомодульными и другими методами доступа и использование локальных и глобальных подходов;
3. в распределенном доступе к различным ресурсам, начиная от программ, файлов и данных заканчивая компьютерами, сенсорами и сетями;
4. в различных моделях использования ресурсов (от однопользовательских до многопользовательских, от высокопроизводительных до мало затратных) и, следовательно, включающих регулирование качества предоставляемого обслуживания, планирование, перераспределение и ведение учета ресурсов.

Анализ альтернативных технологий построения распределенных вычислительных систем, проведенный в [31], показал, что их применение не позволяет в полной мере достичь исполнения всех требований, указанных выше. В соответствии с этим была предложена альтернативная архитектура грид. Исследования и разработки в сообществе грид привели к разработке протоколов, сервисов и инструментария, направленного именно на те проблемы, которые возникают при попытке создания масштабируемых ВО. Эти технологии включают в себя:

1. решения по безопасности, поддерживающие управление сертификацией и политиками безопасности, когда вычисления производятся несколькими организациями;
2. протоколы управления ресурсами и сервисами, поддерживающие безопасный удаленный доступ к вычислительным ресурсам и ресурсам данных, а также перераспределение различных ресурсов;

3. протоколы запроса информации и сервисы, обеспечивающие настройку и мониторинг состояния ресурсов, организаций и сервисов;
4. сервисы обработки данных, обеспечивающие поиск и передачу наборов данных между системами хранения данных и приложениями.

Выделяют следующие уровни архитектуры грид:

1. *Базовый уровень (Fabric)* – содержит различные ресурсы, такие как компьютеры, устройства хранения, сети, сенсоры и др.
2. *Связывающий уровень (Connectivity)* – определяет коммуникационные протоколы и протоколы аутентификации.
3. *Ресурсный уровень (Resource)* – реализует протоколы взаимодействия с ресурсами РВС и их управления.
4. *Коллективный уровень (Collective)* – управление каталогами ресурсов, диагностика, мониторинг;
5. *Прикладной уровень (Applications)* – инструментарий для работы с грид и пользовательские приложения.

На *базовом уровне* определяются службы, обеспечивающие непосредственный доступ к ресурсам, использование которых распределено посредством протоколов Грид.

1. Вычислительные ресурсы предоставляют пользователю Грид-системы (точнее говоря, задаче пользователя) процессорные мощности. Вычислительными ресурсами могут быть как кластеры, так и отдельные рабочие станции. При всем разнообразии архитектур любая вычислительная система может рассматриваться как потенциальный вычислительный ресурс Грид-системы.
2. Ресурсы памяти представляют собой пространство для хранения данных. Для доступа к ресурсам памяти также используется программное обеспечение промежуточного уровня, реализующее унифицированный интерфейс управления и передачи данных.
3. Информационные ресурсы и каталоги являются особым видом ресурсов памяти. Они служат для хранения и предоставления метаданных и информации о других ресурсах Грид-системы.
4. Сетевой ресурс является связующим звеном между распределенными ре-

сурсами Грид-системы. Основной характеристикой сетевого ресурса является скорость передачи данных.

Связывающий уровень определяет коммуникационные протоколы и протоколы аутентификации, обеспечивая передачу данных между ресурсами базового уровня. Связывающий уровень грид основан на стеке протоколов TCP/IP:

1. Интернет (IP, ICMP);
2. Транспортные протоколы (TCP, UDP);
3. Прикладные протоколы (DNS, OSRF…).

Ресурсный уровень реализует протоколы, обеспечивающие выполнение следующих функций:

- согласование политик безопасности использования ресурса;
- процедура инициации ресурса;
- мониторинг состояния ресурса;
- контроль над ресурсом;
- учет использования ресурса.

Отдельно выделяются 2 типа протоколов ресурсного уровня:

1. *Информационные протоколы* – используются для получения информации о структуре и состоянии ресурса.
2. *Протоколы управления* – используются для согласования доступа к разделяемым ресурсам, определяя требований и допустимых действий по отношению к ресурсу (например, поддержка резервирования, возможность создания процессов, доступ к данным).

Коллективный уровень отвечает за глобальную интеграцию различных наборов ресурсов и может включать в себя службы каталогов; службы совместного выделения, планирования и распределения ресурсов; службы мониторинга и диагностики ресурсов; службы репликации данных.

На *прикладном уровне* располагаются пользовательские приложения, исполняемые в среде ВО. Они могут использовать ресурсы, находящиеся на любых нижних слоях архитектуры Грид.

7.10. Стандарты Грид

Ключевым моментом в разработке грид приложений является *стандарти-*

зация, позволяющая организовать поиск, использование, размещение и мониторинг различных компонентов, составляющих единую виртуальную систему, даже если они предоставляются различными поставщиками услуг или управляются различными организациями [28]. К началу 2001 года в различных проектах были представлены различные методы реализации грид-вычислений. Но все они сходились в одном: для гибкого, прозрачного и надежного предоставления доступа к вычислительным ресурсам была предложена сервисно-ориентированная модель.

В 2001 году в качестве базы для создания стандарта архитектуры грид приложений была выбрана технология веб-сервисов. Данный выбор был обусловлен двумя основными достоинствами данной технологии. Во-первых, язык описания интерфейсов веб-сервисов WSDL (Web Service Definition Language) поддерживает стандартные механизмы для определения интерфейсов отдельно от их реализации, что в совокупности со специальными механизмами связывания (транспортным протоколом и форматом кодирования данных) обеспечивает возможность динамического поиска и компоновки сервисов в гетерогенных средах. Во-вторых, широко распространенная адаптация механизмов веб-сервисов означает, что инфраструктура, построенная на базе веб-сервисов, может использовать различные утилиты и другие существующие сервисы, такие как различные процессоры WSDL, системы планирования потоков задач и среды для размещения веб-сервисов [30].

Разработанный стандарт архитектуры грид получил название *OGSA* (Open Grid Services Architecture – Открытая архитектура грид-сервисов) [18]. 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 [30]. Грид-сервис поддерживает следующие стандартные интерфейсы.

1. *Поиск*. Грид приложениям необходимы механизмы для поиска доступных сервисов и определения их характеристик.

2. *Динамическое создание сервисов*. Возможность динамического создания и управления сервисами – это один из базовых принципов OGSA, требующий наличия сервисов создания новых сервисов.
3. *Управление временем жизни*. Распределенная система должна обеспечивать возможность уничтожения экземпляра грид-сервиса.
4. *Уведомление*. Для обеспечения работы грид приложения наборы грид сервисов должны иметь возможность асинхронно уведомлять друг друга 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 то- гда стандарты веб-сервисов (к которым относились WSDL, SOAP, UDDI) не могли обеспечить всех требований, предъявляемых разработчиками к функциональным возможностям грид-сервисов, при создании OGSI потребовалось модифицировать и расширить соответствующие стандарты [21]. Это привело к тому, что совместное использование веб-сервисов и грид-сервисов в одной среде стало невозможным, из-за несовместимости базовых стандартов [4].

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*) [13, 19, 22, 66] специфицированы универсальные механизмы для определения, просмотра и управления состоянием удаленного ресурса, что является критически-важным с точки зрения грид [25]. 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 [20] и UNICORE [48].

7.11. Система Globus

Globus – это проект по разработке и предоставлению инфраструктуры для грид-вычислений [20]. Становление данного проекта приходится на 1997 год, а его разработка продолжается и сегодня. Когда как первоначально Globus был

развитием проекта I-WAY, в процессе развития, основной акцент был перенесен с поддержки высокопроизводительных вычислений в сторону сервисов поддержки виртуальных организаций.

Цель его создания – предоставление возможности приложениям работать с распределенными разнородными вычислительными ресурсами как с единой виртуальной машиной. Основная направленность данного проекта – вычислительные грид-системы. Под вычислительной грид-системой подразумевается инфраструктура аппаратных и программных ресурсов, реализующая надежный и полномасштабный доступ к высокопроизводительным вычислительным системам, независимо от географического расположения пользователей или ресурсов.

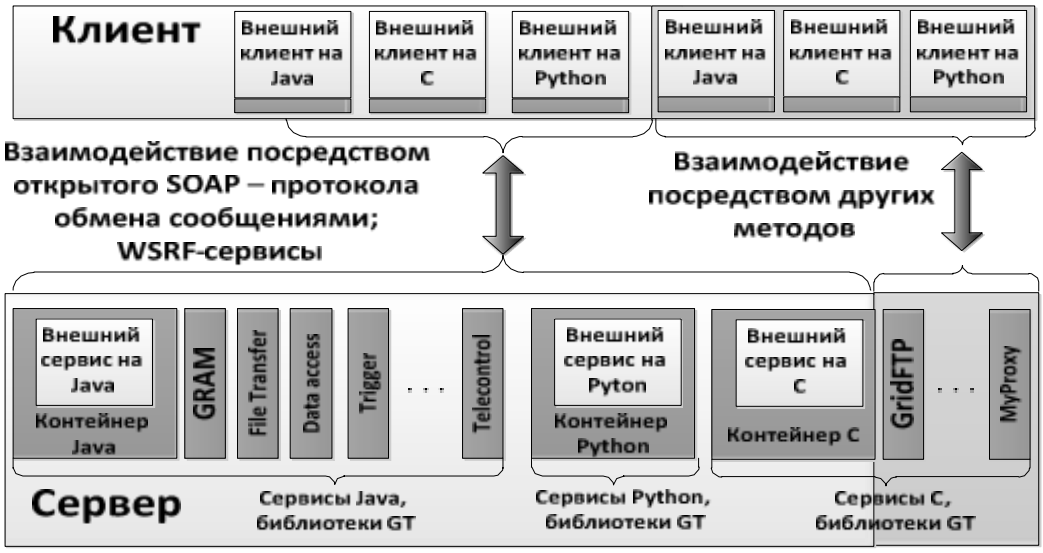


Рис.73 - Общая сх ма взаимоде ствия компонентов Globus Toolk 4.0

Базовым элементом системы Globus выступает Globus Toolkit (инструментарий Globus), описывающий базовые сервисы и возможности, необходимые для создания вычислительных грид-систем [20]. Система Globus предоставляет высокоуровневым приложениям доступ к сервисам, каждыйt из которых приложение или разработчик может использовать для достижения собственных целей. Такой метод работы может быть реализован только при высокой степени изолированности отдельных сервисов и четко определенном программном ин-

терфейсе каждого предоставляемого сервиса.

Рассмотрим базовые сервисы, предоставляемые системой Globus на сегодняшний день (см. рис. 58).

1. Протокол GRAM (“Globus Toolkit Resource Allocation Manager” – Менеджер Распределения Ресурсов Globus Toolkit) используется для распределения вычислительных ресурсов и для контроля вычислений, с использованием данных ресурсов.
2. Расширенная версия протокола передачи файлов GridFTP используется для организации доступа к данным, включая вопросы безопасности и параллелизма высокоскоростной передачи данных.
3. Контейнеры для пользовательских сервисов, поддерживающие аутентификацию, управление состоянием, поиск и т.п. обеспечивающие поддержку стандартов WSRF, WS-Security, WS-Notification.
4. Сервисы аутентификации и безопасности соединений GSI (“Grid Security Infrastructure” – Инфраструктура Безопасности Грид).
5. Распределенный доступ к информации о структуре и состоянии системы распределенных вычислений.
6. Удаленный доступ к данным посредством последовательных и параллельных интерфейсов.
7. Создание, кэширование и поиск исполняемых ресурсов.
8. Библиотеки, для обеспечения взаимодействия сторонних приложений с GTK 4.0 и/или пользовательскими сервисами.

7.12. Система UNICORE

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 безопасно5го доступа к ресурсам грид [65].

Архитектура UNICORE 6 [64] формируется из клиентского, сервисного и системного слоев (см. рис. 59). Верхним слоем в архитектуре является клиентский слой. В нем располагаются различные клиенты, обеспечивающие взаимо-

действие пользователей с грид средой:

- UCC (Unicore Command Line Client – клиент командной строки для UNICORE): клиент, обеспечивающий 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 [48]. 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.

Клиентский слой

Сервисный слой

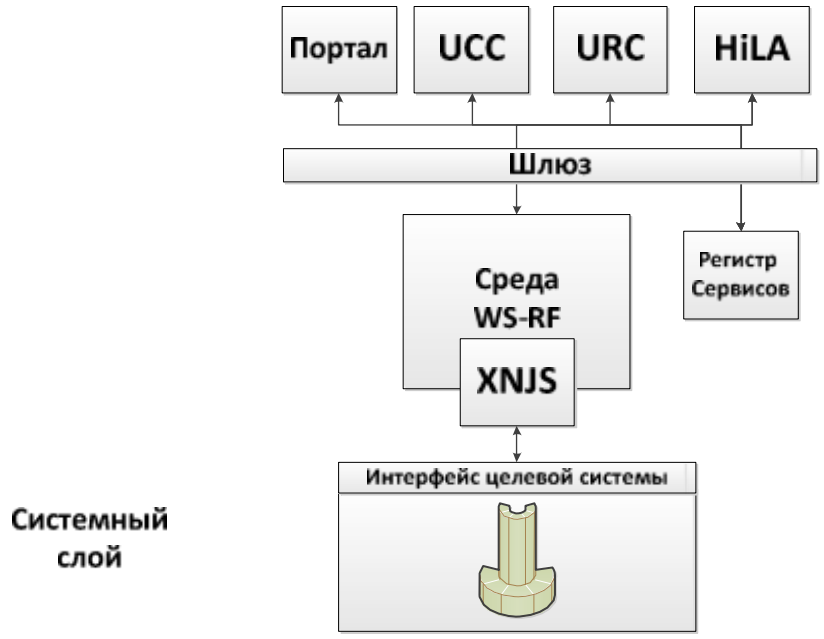


Рис. 7.4- Архитектура UNICORE 6

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.

7.13. Метакомпьютинг и технологии Grid

GRID-технологии являются одной из перспективных реализаций в области кластерных исследований. Концепция Grid ("решетка" – построение вычислительной системы наподобие модели кристалла – с компьютерами в узлах решетки) зародилась в контексте проблемы построения сверхмощных вычислительных установок. В 80-х гг. XX в. для этого уже применялись суперкомпьютерные технологии, полученный опыт оказался не только позитивным. Пришло понимание, что при высокой цене жесткие архитектуры суперкомпьютеров обладают ограниченной масштабируемостью. В то же время для решения ряда задач (в частности, для прогнозирования природных явлений, обработки данных о высокоэнергетических ядерных реакциях, для исследования

эволюции Вселенной и т.п.) требовались ***гибкие и наращиваемые*** вычислительные мощности принципиально нового уровня.

Идея объединения вычислительных и коммуникационных технологий в гибкие системы дала в 90-х гг. XX в. толчок ***метакомпьютингу*** как способу соединения отдельных вычислительных центров. В настоящее время метакомпьютинг определяется как "использование мощных вычислительных ресурсов, прозрачно доступных посредством коммуникационной среды". В дополнение к прозрачности существенны такие характеристики, как "бесшовность", "масштабируемость", "синхронизируемость" и "глобальность". Таким образом, предлагалось, "скрыв" коммуникации, использовать подключенные к сети компьютеры и целые вычислительные центры как единый вычислительный ресурс. Основной акцент делался на то, что потенциальный пользователь может получить практически неограниченные мощности для выполнения суперсложных вычислений и хранения данных [2].

В качестве процессорных ресурсов в современных Grid- системах применяются не только мощные специализированные вычислительные устройства, но также рабочие станции и обычные персональные компьютеры. Теперь под термином "вычислительные ресурсы" понимают все, что участвует в совместной компьютерной обработке данных. К ресурсам следует отнести программные комплексы, коммуникации, системы хранения, хранилища данных и информационные системы.

Например, Grid – это не просто технологии для высокопроизводительных вычислений. Современное реальное содержание Grid – это инфраструктура, обеспечивающая поддержку любой глобально распределенной обработки для многих типов приложений: электронного бизнеса, распределенного производства, исследования данных, системы обработки высокой пропускной способности (High Throughput Computing – HTC) и, конечно, распределенного суперкомпьютинга. В отличие от метакомпьютинга, для многих приложений, в том числе и с большим объемом вычислений, не требуются высокопроизводительные коммуникации; в

этом случае в качестве коммуникационной среды может выступать привычный Интернет [agora.guru.ru/abrau2008/pdf/050.pdf].

Ниже кратко рассмотрена архитектура программного обеспечения Grid. Базовым программным обеспечением Grid и де-факто международным стандартом является сегодня Globus [osp.ru/os/2003/01/182393/|. Его взяли за основу в ведущих проектах Grid (IPG, NCSA, Gryphyn, DataGrid). Бóльшая часть новых исследований и разработок в области Grid ориентируется именно на Globus [2].

Основная задача, решаемая в Grid, – обеспечение доступа к ресурсам, а поскольку ресурсы распределенные, то функционирование обеспечивается специальной формой организации программного обеспечения – службами. В отличие от модели "клиент-сервер" тот или иной набор служб устанавливается здесь на каждом ресурсе. Множество служб должно удовлетворять двум структурным условиям:

• Каждый тип служб должен иметь стандартный протокол доступа, в соответствии с которым реализуется прикладной интерфейс клиентов. В рамках стандартных протоколов допустимы различные способы реализации служб.

• Множества служб на разных ресурсах должны быть согласованными. Это предполагает известную унификацию наборов служб на основе тождественности их семантики, а также наличие общих правил, регламентов и организационных соглашений, на которые опирается конфигурирование служб.

На рис. 5.3 показан алгоритм отправки задания и загрузки свободного вычислительного устройства с учетом политики доступа и поиска свободного ресурса, реализуемый соответствующей службой.

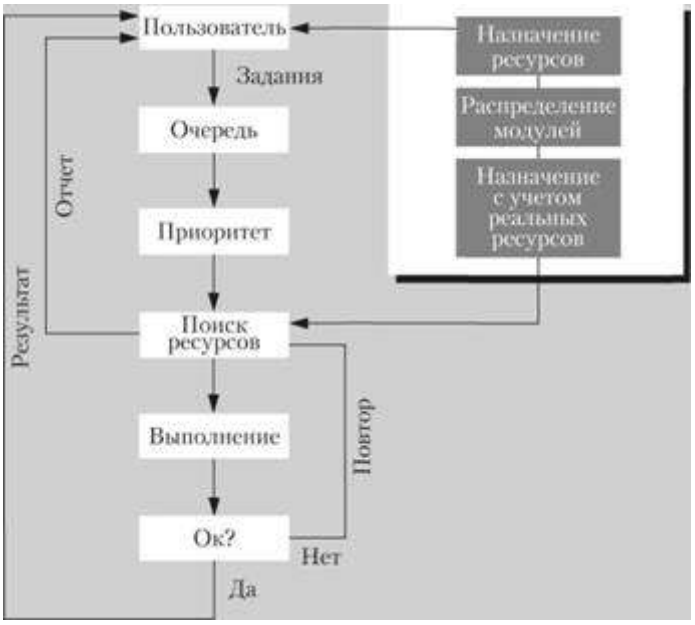


***Рис. 7.5* - Алгоритм прохождения задания в Grid-системе**

Успех проекта Globus обеспечили следующие ключевые службы и протоколы:

• протокол доступа к управлению ресурсами (Grid Resource Allocation and Management – GRAM) и служба Gatekeeper, которые обеспечивают безопасное создание удаленных процессов и управление ими;

• служба метакаталогов (Grid Information Service – GIS), отвечающая за распределенный сбор данных и информационное обслуживание;

• службы инфраструктуры безопасности (Grid Security Infrastructure – GSI), поддерживающие однократную регистрацию, делегирование полномочий и отображение прав доступа на разные локальные системы.

Работы в области Grid направлены сегодня на существенное расширение состава служб. Предложенное на данный момент решение состоит в том, чтобы "за основу" взять стандарты Web-служб, разработанные консорциумом W3C (SOAP, WSDL, WS-Inspection и т.д.). В результате архитектура Grid наследует замечательные свойства Web-служб, одновременно наследуя широкий спектр инструментов разработки и исполнительных сред на выбор (например, J2EE, Microsoft.Net, IBM WebSphere, Sun ONE). Базовой архитектурой Grid становится Open Grid Services Architecture, реализованная в Globus 3.0.

7.14. Перспективы развития технологий Грид

Сегодня распределенные инфраструктуры на базе Грид пользуются заслуженной популярностью и могут рассматриваться как инфраструктурная основа для организации предметно-ориентированных сред высокопроизводительных вычислений, рассмотренных в разделах 4 и 5. Однако на существующем этапе развития Грид инфраструктурные проекты, ориентированные на консолидацию географически разнесенных ресурсов в единую среду, уже ушли в прошлое. В настоящее время наиболее актуальной является задача продвижения технологий Грид в предметные области, обеспечивая, с одной стороны, эффективное использование ресурсов распределенной среды, а с другой – поддержку пользователей- неспециалистов в процессе разработки и запуска вычислительных приложений.

Современный Грид порождает специфические требования к инструментальным средствам, ориентированным на работу с распределенными вычислительными ресурсами. К примеру, в мире существует ряд проектов, ориентированных на упрощение работы пользователя с Грид путем создания визуальных инструментов конструирования распределенных приложений [26, 27] или даже полнофункциональных сред разработки параллельных приложений [28, 29]. Другим направлением развития инструментальных средств, оперирующих распределенными вычислительными ресурсами, является построение средств анализа распределенных инфраструктур и оптимизации распределенных приложений. Примерами таких средств могут служить:

- набор утилит ASKALON [30], предназначенный для анализа распределенных приложений в Грид-средах,
- средство для анализа производительности параллельных программ SCALEA [31],
- средство массовой автоматической генерации тестов для Грид-систем ZENTUTIO [32],
- средство для анализа производительности в параллельных и распределенных приложениях Aksum [33],
- инструмент для оценки производительности в Грид-системах приложений,

построенных в форме потока заданий (workflow, WF) [34] и пр.

Использование вычислительных ресурсов Грид для решения задач моделирования сложных систем сопряжено с необходимостью формального описания сценариев моделирования как приложений, использующих сервисы, доступные в Грид. Традиционно, такие приложения представляются в форме WF [35]. При этом должен формироваться язык взаимодействия между пользователем и системой управления вычислительной инфраструктурой. Можно выделить две базовых нотации таких языков: графическая и текстовая. В первом из этих подходов предполагается конструирование композитного приложения в форме цепочки заданий с помощью визуальных инструментов (в частности с использованием концепции *drag-and-drop*). Примерами таких систем могут служить Kepler [36], Taverna [37], Triana [38]. Во втором подходе основным инструментом описания композитных приложений является высокоуровневый (зачастую предметно-ориентированный – domain-specific) язык описания цепочек заданий. Примером системы, использующей такой подход, может служить Geodise [39]. В тоже время, независимо от подхода, исполняя роль интегрирующего инструмента, такие системы зачастую поддерживают широкий спектр открытых протоколов, стандартов и языков представления информации, что обеспечивает возможность использования сторонних сервисов, предоставляющих требуемую функциональность или доступ к необходимым данным. Как следствие, в зависимости от задач разрабатываемой системы могут использоваться различные средства уровня промежуточного программного обеспечения, обеспечивающего решение вспомогательных задач (в том числе задач интеграции). Например, в рамках проекта Virtual Laboratory for e-Science [40] для обеспечения унифицированного программного доступа к ресурсам Грид было использовано средство Grid Operation Invoker [41], а для организации доступа к данным – хранилище RegaDB [42]. Следует отметить, что при построении сложных программных решений на базе Грид возникает необходимость решения ряда вспомогательных задач. Важными задачами являются обеспечение контроля источников информации (например, эту задачу решает PROToS [43]), предоставление отладочной информации разработчику цепочки исполнения композитного приложения (для решения

может использоваться средство EclipseRCP [44]), организация миграции заданий в рамках Грид-среды (например, попытка решения этой задачи на базе стандартных механизмов ОС Linux предпринята в рамках проекта XtreemOS [45]).

В Российской Федерации исследования и разработки в области Грид во многом ориентированы на использование зарубежного опыта создания распределенных систем доступа к данным мегаустановок (БАК); в частности к ним относится российский сегмент EGI (ранее EGEE). По-видимому, на настоящий момент наиболее продвинутой является Грид-среда Национальной нанотехнологической сети (ГридННС), поддерживаемая РНЦ Курчатовский институт и НИИ ядерной физики МГУ, в рамках которой удалось отчасти отойти от традиционных для зарубежных аналогов решений на основе gLite и Globus в пользу более удобных реализаций на основе технологии REST. Грид-среды, созданные в рамках программы «СКИФ- Грид», а также разработанные РНЦ ВНИИЯФ (г. Саров) для доступа к собственным суперкомпьютерам, не являются в полной мере Грид-средами, поскольку не отвечают классической функциональности [46]. Отдельные проработки в области развития средств разработки приложений над Грид, выполнены в 2007-2010 г. в Южно-Уральском государственном университете и Национальном исследовательском университете информационных технологий, механики и оптики (что соответствует описанному выше мировому опыту). Таким образом, современный уровень развития Грид второго поколения позволяет не только обеспечить решение проблем высокопроизводительных вычислений (решение больших задач, моделирование сложных систем, консолидация ресурсов и сервисов), но и дать возможность построения распределенных сценариев моделирования сложных систем в форме WF, включающих в себя различные прикладные Грид-сервисы (что позволяет разрешить ряд аспектов раздела 4). Эти Грид-сервисы могут размещаться физически на разных узлах, принадлежать разным авторам и реализовывать различные технологии. Как следствие, с точки зрения применения Грид принципиальной проблемой является эффективное планирование исполнения композитного приложения, составленного из таких сервисов, в целом удовлетворяющего пользовательским ограничениям для

достижения минимального времени решения задачи.

7.15. Перспективы развития проблемно-ориентированных платформ распределенных вычислений

Проблемно-ориентированные среды распределенных вычислений предназначены для эффективного решения прикладных задач компьютерного моделирования и обработки данных с использованием высокопроизводительных вычислительных ресурсов. В настоящее время, как в мире, так и в Российской Федерации не существует унифицированного подхода к решению данной задачи. В частности, в настоящее время известны различные образцы отечественных платформ распределенных вычислений, находящиеся на различных уровнях развития. Например, платформа для вычислительной и квантовой химии ИПХФ РАН [47] предоставляет сервис для запуска пакета GAMESS на собственных распределенных вычислительных ресурсах. Поддерживается запуск нескольких экземпляров приложения для решения общей задачи с распараллеливанием по данным. Кроме того, поддерживается возможность доставки и развёртывания пакета GAMESS на целевой кластер. Система СИВС [48] является профессиональной платформой для поддержки профессионального сообщества пользователей с расширяемым web-интерфейсом. Интерфейс динамически строится на основе онтологического описания. Предполагается возможность использования этого интерфейса для встраивания в него высокопроизводительных приложений. Web-интегрированный исследовательский центр конструирования лекарств ИМБХ РАМН [49] представляет собой веб-портал с возможностью запуска приложений собственной разработки по конструированию лекарств и прогнозированию их свойств на основе базы данных лекарств и вычислительных мощностей ИМБХ РАМН. Портал построен на основе платформы HubZero, что определяет его основные возможности и ограничения. К преимуществам данного решения можно отнести наличие готового web-интерфейса для сообщества пользователей, а также поддержку онтологий, а к недостаткам – отсутствие поддержки WF. Библиотека GridMD [50] используется в задачах молекулярной динамики для формирования и запуска WF в распределенных средах Грид. Формирование WF

осуществляется непосредственно в выполняемой программе на языке C++. Никаких дополнительных средств библиотека не предоставляет. Проект «Пирамида» [51] представляет собой платформу для назначения подзадач распараллеленной по данным задачи на распределенные вычислительные ресурсы. Планирование в данной системе осуществляется иерархически - каждый нижестоящий обработчик заданий при отсутствии заданий в собственной очереди забирает новую порцию заданий у соответствующего ему вышестоящего обработчика. Непосредственный запуск заданий на вычислительном ресурсе осуществляют обработчики нижнего уровня. Такая иерархическая система позволяет уменьшить количество обращений к основной очереди задач и тем самым разгрузить центральный обработчик заданий, что позволяет избежать

«бутылочного горлышка» в планировщике при большом количестве коротких заданий. Система GMDH Shell [52] представляет интегрированную проблемно-ориентированную среду с графическим интерфейсом пользователя для интеллектуального анализа данных (data mining) с возможностью запуска расчета на удаленном кластере. Выбор кластера осуществляется пользователем вручную. Данный программный продукт не предоставляется в качестве сервиса и пользователь должен конфигурировать и запускать его на собственных вычислительных ресурсах. Проблемно- ориентированная среда для задач механики сплошной среды GIMM [53] предоставляет пользователю графический интерфейс с возможностью запуска на удалённом кластере. Выбор кластера осуществляет сам пользователь. Кроме того, пользователю доступна возможность загружать и запускать свои решатели. Проект MathCloud [54, 55] является проблемно- ориентированной средой для математических задач с веб-интерфейсом. Данная система предоставляет услуги по графической компоновке WF из элементов списка математических пакетов и запуску сформированного WF на предоставляемых распределённых вычислительных ресурсах. Поддерживается возможность объединения части WF в отдельный блок с его дальнейшим сворачиванием/разворачиванием. Система имитационного моделирования Triad.Net [56] предполагает задание пользователем трёхуровневой модели (объекты, сообщения, обработчики сообщений) с последующим запуском процесса моделирования на кластере. Процесс обмена

сообщениями между моделируемыми объектами является, по сути, процессом динамического формирования WF. Элементами такого WF являются ассоциированные с объектами подпрограммы-обработчики сообщений, а связями – сами сообщения. Планировщик системы осуществляет балансировку нагрузки между узлами кластера на основе мультиагентного подхода, используя знания об особенностях работы объектов имитационной модели. Платформа CAEBeans [57, 58] предоставляет возможность создания собственной проблемно-ориентированной среды с возможностью запуска WF на распределённых вычислительных ресурсах через web-интерфейс. Однако сам WF проектируется системным программистом на этапе разворачивания среды и остаётся фиксированным. Соответственно, пользователь не может ни выбрать другой WF, ни изменить существующий. Платформа CLAVIRE [59] разрабатывается ЗАО «АйТи. Информационные технологии» для поддержки различных вычислительных сервисов и композитных приложений в среде облачных вычислений.

Таблица А7.1 Сравнительные характеристики отечественных платформ распределенных вычислений

Проект	Предметная область	Интегрированная среда	SaaS	Поддержка WF	Распределенные вычислительны	Знания	Сообщество пользо-

ИПХФ РАН	Вычислительная и квантовая	веб-форма	+	по данным	+	–	–
СИВС	Платформа	–	+	–	–	+	+
w-ИЦКЛ	Конструирование	веб-форма	+	–	+	+	+
GridMD	Молекулярная	–	–	+	+	–	–
«Пирамида»	Платформа	–	–	по данным	+	–	–
GMDH Shell	Data mining	+	–	–	вручную	–	–
GIMM	Задачи механики сплошной	+	+	–	вручную	–	–
MathCloud	Математика	+	+	+	+	+	+
Triad.Net	Имитационное моделирова	–	–	неявная	–	+	–
CAEBeans	Платформа	веб-форма	+	+	+	–	–
CLAVIRE	Платформа	+	+	+	+	+	+

Сводные данные по вышеуказанным проектам по ряду качественных технико-экономических характеристик приведены в таблице А7.1. Указание значения «платформа» в графе «предметная область» означает, что данный проект не предназначен для решения каких-либо определённых прикладных задач, а является основой для создания высокопроизводительного приложения в произвольной предметной области. Как видно из приведенной таблицы А7.1, ни один

из существующих отечественных продуктов не поддерживает в полной мере всех современных тенденций развития распределенных вычислений, перечисленных в заглавной строке таблицы. Главным образом следует отметить слабое распространение композитных приложений (большинство проектов либо не поддерживают WF вовсе, либо поддерживают лишь частные случаи распараллеливания по данным), а также практически полное отсутствие проектов, использующих интеллектуальные технологии.

При этом можно выделить только две платформы, в полной мере отвечающие всем требованиям п. 2. Это среда MathCloud, разрабатываемая в Центре Грид-технологий института системного анализа РАН, и платформа CLAVIRE, разрабатываемая ЗАО «АйТи. Информационные технологии» в кооперации с Национальным исследовательским университетом информационных технологий, механики и оптики.

Вопросы к главе 7.

1. Грид (Grid)-системы. Что такое грид. Метакомпьютинг и технологии Grid Алгоритм прохождения задания в Grid-системе
2. Сервисы грид. Приложения для Грид
3. Промежуточное ПО управления Грид
4. Проблемы и ограничения Грид
5. Архитектура Грид
6. Стандарты Грид
7. Перспективы развития технологий Грид

Глава 8. Облачные вычисления

«Облачные вычисления» — это относительно новая концепция, стремительно ворвавшаяся в сферу информационных технологий. Несмотря на то, что задача «облачных вычислений» заключается в предоставлении относительно нересурсоемких сервисов отдельным пользователям, в основе этой технологии лежат высокопроизводительные вычисления. **От «коробочного» программного обеспечения – к облачным сервисам и композитным приложениям.** Характерной особенностью традиционных программных комплексов компьютерного моделирования являются высокие трудозатраты на их установку и настройку. Это связано с тем, что большинство отчуждаемых программных решений такого рода сформировалось на основе исследовательского инструментария с невысокими эргономическими характеристиками. Эта проблема полностью устраняется, когда пользователь ориентируется на распределенные композитные приложения, создаваемые на основе отдельных предметно- ориентированных сервисов, которые устанавливаются непосредственно своими разработчиками на выделенных целевых системах и могут даже не существовать в форме «коробочных» решений. Как следствие, для распространения таких приложений становится эффективной бизнес-модель SaaS (Software as a Service), в рамках которой провайдер предоставляет пользователю в аренду через Интернет приложение, функционирующее на его же технологической площадке. При этом в качестве таких приложений могут рассматриваться как отдельные сервисы, так и объединяющее их *композитное* приложение (AaaS, Application as a Service). Использование моделей SaaS и AaaS изменяет требования и к самим вычислительным

сервисам. Так, для многих задач использование суперкомпьютерных систем традиционной архитектуры далеко не всегда является оправданным в силу весьма низкой параллельной эффективности. Если пренебречь переносимостью программы (тем самым принципиально отказываясь от традиционного «коробочного» способа ее распространения), в ряде случаев можно эффективно реализовать ее сервисную версию для исполнения на вычислительных системах реконфигурируемой архитектуры (FPGA) [11]. Несмотря на то, что стоимость разработки в данном случае будет выше в разы, этот недостаток окупается за счет стоимости последующего владения комплексом в целом.

8.1. Что такое «облачные вычисления»

Облачная обработка данных — это парадигма, в рамках которой информация постоянно хранится на серверах в интернете и лишь временно кэшируется на клиентской стороне, например, на персональных компьютерах, игровых приставках, ноутбуках, смартфонах и т. д. Определение Национального Института Стандартов и Технологий США[6], официально принятое правительством США: «облачные» вычисления (англ. cloud computing) – это модель обеспечения повсеместного и удобного сетевого доступа по требованию к общему набору настраиваемых вычислительных ресурсов, которые могут быть оперативно предоставлены и освобождены с минимальными эксплуатационными затратами и/или обращению к провайдеру.

Выделяют пять основных характеристик облачных систем:

1. динамическое выделение ресурсов для удовлетворения потребностей пользователей в вычислительных ресурсах и услугах;
2. эмуляция «бесконечного» хранилища виртуальных ресурсов, доступных по первому требованию;
3. эластичность, а именно возможность динамически осуществлять горизонтальное масштабирование, как в сторону увеличения мощности систем, так и в сторону её снижения;
4. широкий спектр способов доставки вычислительных ресурсов и услуг пользователю;
5. измеряемое качество оказываемых услуг. Поддержание стабильного качества оказываемой услуги осуществляется за счет автоматического перераспределения внутренних ресурсов.

Признаки «облака»:

- сервис приспособлен для удаленного использования (через Интернет);
- одним сервисом пользуется несколько клиентов одновременно;
- оплата взимается либо в виде ежемесячной абонентской платы, либо на основе объема операций;
- техническая поддержка приложения включена в оплату;
- модернизация и обновление происходит плавно и прозрачно для клиентов.

8.2. Виды сервисов

Облачная обработка данных включает в себя следующие концепции:

- «Программное обеспечение как услуга»;
- «Платформа как услуга»;
- «Инфраструктура как услуга»;
- «Безопасность как услуга»;

---

6 National Institute of Standards and Technology www.nist.gov

- «Рабочее место как услуга»;
- «Всё как услуга».

Рассмотрим их подробнее.

8.2.1. Программное обеспечение как услуга

Software as a service (SaaS) («программное обеспечение как услуга») — бизнес-модель продажи программного обеспечения, при которой поставщик разрабатывает веб-приложение и самостоятельно управляет им, предоставляя пользователям доступ к программному обеспечению через Интернет.

Пользователи платят не за владение программным обеспечением как таковым, а за использование его через веб-интерфейс. В отличие от обычной схемы лицензирования ПО, заказчик несет сравнительно небольшие периодические затраты, и ему не требуется инвестировать значительные средства в приобретение ПО и аппаратной платформы для его развертывания, а затем поддерживать его работоспособность. Схема периодической оплаты предполагает, что если необходимость в программном обеспечении временно отсутствует, то заказчик может приостановить его использование и заморозить выплаты разработчику.

Для разработчиков проприетарного ПО модель SaaS позволяет бороться с нелицензионным использованием программного обеспечения,

поскольку само ПО не попадает к конечным заказчикам.

8.2.2. Платформа как услуга

Platform as a Service (PaaS): в отличие от SaaS, предназначенного больше для конечного пользователя, PaaS — это вариант для разработчиков. В облаке функционирует некоторый набор программ, основных сервисов и библиотек, на основе которых предлагается разрабатывать свои приложения.

Самый яркий пример – предоставление платформы LAMP (Linux-Apache-MySQL-PHP) для разработки web-приложений.

Также под PaaS понимают и отдельные части сложных систем, вроде системы базы данных или коммуникаций.

8.2.3. Инфраструктура как услуга

Infrastructure as a Service (IaaS) — это предоставление компьютерной инфраструктуры (как правило в форме виртуализации) как услуги на основе концепции облачных вычислений.

IaaS состоит из трех основных компонентов:

6. аппаратные средства (серверы, системы хранения данных, клиентские системы, сетевое оборудование);
7. операционные системы и системное ПО (средства виртуализации, автоматизации, основные средства управления ресурсами);
8. связующее ПО (например, для управления системами).

8.2.4. Безопасность как услуга

Security as a service — предоставление услуг по обеспечению информационной безопасности на основе «облачных» технологий.

Эта концепция может включать в себя, например, следующие функции:

- защита от вирусов и шпионящего ПО;
- защита от сложных угроз (обнаружение вторжений);
- безопасность браузера;
- управление трафиком;
- URL-фильтрация;
- контроль приложений Web 2.0;
- контроль полосы пропускания (в каких объемах полоса расходуется на приложения, используемые для работы, а в каких — для развлечения);
- защита от утечки данных;
- централизация политик безопасности и лог-файлов;
- централизованная политика для веб-трафика;
- консолидированная отчетность в режиме реального времени.

8.2.5. Рабочее место как услуга

Desktop as a Service (DaaS) — модель использования ПО, являющаяся развитием SaaS.

Клиенты получают полностью готовое к работе («под ключ») стандартизованное виртуальное рабочее место, которое каждый пользователь имеет возможность дополнительно настроить под свои задачи. То есть пользователь получает доступ не к отдельной программе, а к необходимому для полноценной работы программному комплексу.

Физически доступ может быть организован через локальную сеть или Интернет. В качестве терминала может использоваться ПК или ноутбук, нетбук и даже смартфон. Устройство доступа используется в качестве тонкого клиента и требования к нему минимальны.

Основные достоинства DaaS:

- возможность быстро организовать офис с минимальными первоначальными затратами;
- возможность дать доступ к полноценному рабочему месту для разъездных сотрудников (командировки, торговые представители);
- дополнительная защита основного массива корпоративной информации;
- стандартизация рабочих мест;
- контроль над потоками данных пользователей и централизованное обслуживание.

8.2.6. Всё как услуга

Декларируемая цель — пользователь может получить все, что ему необходимо, используя только «облачные» технологии: все данные (включая персональные) будут храниться на удаленных серверах, общение будет происходить исключительно по технологиям SaaS, работа — по технологиям DaaS, безопасность будет обеспечиваться технологиями Security as a Service.

Все, что нужно пользователю — устройство для входа в сеть Интернет (телефон, ноутбук, смартфон и т.п.) и средства идентификации/аутентификации.

8.3. Частные и публичные облачные системы

Выделяют четыре модели развертывания «облачных» сервисов по отношению к кругу пользователей:

1. частное облако (англ. private cloud) – инфраструктура, предназначенная для использования одной организацией, включающей несколько потребителей (например, подразделений одной организации), возможно также используется клиентами и подрядчиками данной организации.

2. Публичное облако (англ. public cloud) – инфраструктура, предназначенная для свободного использования широкой публикой. Публичное облако может находиться в собственности, управлении и эксплуатации коммерческих, научных и правительственных организаций (или какой-либо их комбинации).
3. Гибридное облако (англ. hybrid cloud) – комбинация из двух или более различных

облачных инфраструктур (частных, публичных или общественных), остающихся уникальными объектами, но связанных между собой стандартизованными или частными технологиями переносимости данных и приложений (например, кратковременное использование ресурсов публичных облаков для балансировки нагрузки между облаками).

4. Общественное облако (англ. community cloud) – вид инфраструктуры, предназначенный для использования конкретным сообществом потребителей из организаций, имеющих общие задачи (например, миссии, требований безопасности, политики, и соответствия различным требованиям).

8.3.1. Зачем нужны «облака»

Рассмотрим теперь чем концепция облачных вычислений выгодна

- разработчикам программного обеспечения;
- компаниям, предлагающим «облачные» услуги (провайдерам сервисов).

Положительные факторы SaaS для заказчиков:

- отсутствие необходимости установки ПО на рабочих местах пользователей — доступ к ПО осуществляется через обычный браузер;
- радикальное сокращение затрат на развертывание системы в организации;
- сокращение расходов на аренду помещения, организацию дата-центра, оплату труда сотрудников и т. д.;
- сокращение затрат на техническую поддержку и обновление развернутых систем (вплоть до их полного отсутствия);
- быстрота внедрения, обусловленная отсутствием затрат времени на развертывание системы;
- понятный интерфейс — большинство сотрудников уже привыкли к использованию

веб-сервисов;
- ясность и предсказуемость платежей, защита инвестиций;
- мультиплатформенность;
- возможность получить более высокий уровень обслуживания ПО.

Положительные факторы для разработчиков:
- рост популярности веб-сервисов для конечных пользователей — единая платформа;
- в долгосрочном периоде доходы от SaaS могут оказаться выше прибыли, полученной от продажи лицензий и оказания технической поддержки;
- в отличие от классической модели, заказчик SaaS привязывается к разработчику — он не может отказаться от услуг разработчика и продолжать использовать систему. Таким образом обеспечивается защита инвестиций разработчика в процесс продаж;
- отсутствие проблем с нелицензионным распространением ПО;
- легкое проникновение на глобальные рынки;
- развитие веб-технологий, большие функциональные возможности веб-приложений и простота их реализации;
- быстрые процессы внедрения и сравнительнонизкие затраты ресурсов на обслуживание конкретного клиента.

8.3.2. Недостатки облаков

Как и любая другая концепция, облачные системы имеет ряд присущих ей недостатков.

К ним относятся следующие.

Проблемы переносимости: отдав данные постороннему сервису, как вы можете гарантировать, что сможете вернуть их обратно в удобном виде, когда «облачный» сервис станет слишком дорогим или неудобным?

Использование «облачных» технологий означает привязку заказчика к единственному разработчику, который размещает программное обеспечение на своей площадке, осуществляет его администрирование и поддержку. Это может привести заказчиков к проблемам, связанным с возможной нестабильностью разработчика или каким-либо нарушениям договоров.

Проблемы управляемости: имея штат разработчиков для собственной программы, вы можете управлять тем, какие функции будет иметь программа, в каком направлении она будет развиваться. Пользуясь

свободным ПО, вы можете влиять на развитие программы либо активным участием, либо финансируя добавление необходимой вам функциональности. Пользуясь проприетарными программами, вы можете подобрать для себя ту, что наиболее подходит функционально. Пользуясь «облачными» сервисами, вы не можете напрямую влиять ни на что...

Зависимость от коммуникаций: при потере связи с «облачным» сервисом пользователь практически теряет возможности по работе с собственными данными. Это может привести как к кратковременным неудобствам, так и простоям в работе. При этом, следует опасаться обрывов связи и на стороне клиента, и на стороне «облака», и на пути между ними.

Проблема создания неконтролируемых данных: информация, оставленная пользователем, хранится годами, либо без его ведома, либо он не в состоянии изменить какую-то ее часть. Пример — сервисы Google, где пользователь не может удалить неиспользуемые им сервисы и даже удалить отдельные группы данных, созданные в некоторых из них (FeedBurner, Google Friend Connect...). В качестве альтернативы «очистке» своего профиля предлагается создать новый. Однако не стоит забывать о том, что имя пользователя уже занято предыдущей учётной записью, а новые — наподобие John22441 — устраивают не всех. Поскольку облачные платформы проприетарны, пока нет надежды на то, что пользователю предоставят все полномочия для контроля своих же собственных данных в «облаках».

Высокая цена сбоя в системе: некоторые аналитики прогнозируют появление масштабных проблем с облачными вычислениями: из-за значительного притока пользователей сервисов, использующих облачные вычисления, растет цена ошибок и утечек информации с подобных ресурсов, следовательно, произойдут крупные «катастрофы типа выхода из строя, или катастрофы, связанные с безопасностью».

Так, например, в 2009 году сервис для хранения закладок Magnolia потерял все свои данные.

Пользователи смартфонов Sidekick сети T-Mobile потеряли доступ ко всем своим данным, хранящимся на удалённых серверах. Причины происшествия не были раскрыты, а восстановление информации до последнего момента считалось невозможным. Основной идеей Sidekick является использование «облачных» технологий обработки и хранения пользовательских данных: вся информация, которую использовали владельцы смартфонов (телефонные контакты, сообщения, календари, фотографии, заметки, todo-листы и т.д.) хранилась не на самих

устройствах, а на серверах разработчика Sidekick — компании Danger. Причем, концепцией Sidekick не предусматривалось даже теоретической возможности создания локального бэкапа на собственном компьютере. Видимо, Danger безоговорочно верила в отказоустойчивость своего сервиса нового поколения.

Потеря контроля: «Использовать веб-приложения для своих вычислительных процессов не следует, например, потому, что вы теряете над ними контроль. И это не лучше, чем использовать любую проприетарную программу. Проводите свои вычисления на собственном компьютере, используя программы, уважающие вашу свободу. Если вы используете любую проприетарную программу или чужой веб-сервер, вы становитесь беззащитными. Вы становитесь игрушкой в руках того, кто разработал это ПО.»

— "Cloud computing is a trap, warns GNU founder Richard Stallman", интервью газете The Guardian

Проблемы безопасности: «облачный» сервис, обслуживающий тысячи/сотни тысяч пользователей, заинтересует многих злоумышленников. При этом проблема осложняется большим количеством разнообразных сервисов — все они потенциально имеют уязвимости в безопасности. Проблемы с безопасностью практически исключают использование
«облачных» технологий для критически важных систем, в которых обрабатывается строго конфиденциальная информация.

Неизвестный хакер 31 августа 2014 г. взломал сервис iCloud, принадлежащий компании Apple, и получил доступ к личным архивам (фото и видео), хранившихся на мобильных телефонах «звёзд». Изображения звезд, позирующих в обнаженном виде, были опубликованы на крупнейшем в мире сервисе картинок 4chan.org, главными особенностями которого являются практические полное отсутствие цензуры и анонимность. Всего злоумышленники получили доступ к учетным записям более сотни знаменитостей, в числе которых актрисы Дженнифер Лоуренс, Кейт Аптон, Кирстен Данст, певица Ариана Гранде. Утечка интимных фото звезд стала крупнейшим в истории «облачным» скандалом.

8.4. Сравнение Грид и Облачных вычислений

Описание концепции грид-вычислений в предыдущей главе и концепции облачных вычислений, которая была представлена в этой, показывают, что между ними много общего. Это повлекло за собой

множество дискуссий, как в коммерческой, так и в научной среде. Основной вопрос этих дискуссий заклю- чался в следующем: чем облачные и грид вычисления отличаются друг от дру- га? Действительно ли термин «облачные вычисления» – это просто новая мар- кетинговая уловка, под соусом которой публике предоставляются услуги грид- сред?

Некоторые исследователи считают, что основным отличием облачных вы- числений от грид является виртуализация: «Облачные вычисления, в отличие от грид, применяют виртуализацию для максимизации вычислительной мощно-
сти. Виртуализация, посредством отделения логического уровня от физическо- го, решает множество проблем, с которыми сталкиваются грид-решения».

В то время как грид-системы обеспечивают высокую загрузку вычисли- тельных ресурсов посредством распределения одной сложной задачи на не- сколько вычислительных узлов, облачные вычисления идут по пути исполне- ния нескольких задач на одном сервере в виде виртуальных машин. Кроме того, есть особенности в основных вариантах использования грид и облачных вы- числений. Тогда как грид, в основном, используется для решения задач за опре- деленный (ограниченный) промежуток времени, облачные вычисления в ос- новном ориентированы на предоставление «долгоживущих» сервисов.

Мнения сходятся в одном – облачные вычисления выросли из концепции грид. Фостер определяет взаимодействие грид и облачных вычислений следу- ющим образом:

«Мы считаем, что облачные вычисления не просто пересекаются с концеп- цией грид. На самом деле облака выросли из грид-вычислений и основываются на концепции инфраструктуры грид. Эволюция подхода заключается в том, что вместо предоставления «сырых» вычислительных ресурсов и ресурсов хране- ния обеспечивается предоставление более абстрактных ресурсов в виде серви- сов».

Таким образом, можно считать что грид и облачные вычисления дополня- ют друг друга. Интерфейсы и протоколы грид могут обеспечить взаимодей- ствие между облачными ресурсами или же обеспечить объединение облачных платформ. Также, более высокий уровень абстракции, предоставляемый облач- ными платформами, может помочь пользователям грид-систем в организации прозрачного и удобного предоставления ресурсов грид-платформ и привлечь новые группы пользователей к использованию таких ресурсов.

С точки зрения пользователя, разница между облачными вычислениями и грид вычислениями будет состоять в следующем:

- *Облачные платформы фокусируются на подходе «всё как сервис»*. Грид вычисления фокусируются на промежуточном программном обеспечении, которое предоставляется в виде открытых исходных кодов или же в виде готовых пакетов. При этом коммунальные вычисления являются всего лишь одной из форм предоставления грид. По сравнению с этим, облачные вычисления фокусируются исключительно на платном предоставлении информационных ресурсов конечному пользователю. При этом промежу- точное программное обеспечение, которое позволило бы обеспечить раз- работку собственного облака, пока не очень распространено.
- *Грид и облачные вычисления фокусируются на различные типы вычисле- ний*. Изначально, грид вычисления были ориентированы на решение науч- ных задач посредством суперкомпьютерных систем. В настоящее время грид применяется для научно-исследовательских задач, решение которых требует объединение нескольких суперкомпьютерных платформ. С другой стороны, облачные вычисления ориентированы не на решение отдельных задач, а на перманентное предоставление определенных сервисов конеч- ным пользователям. Они обеспечивают динамическое распределение фи- зических ресурсов для удовлетворения переменной загрузки таких

серви- сов.

- *Различное взаимоотношение с поставщиками ресурсов*. Грид вычисления основываются на понятии виртуальных организаций, включающих в себя несколько различных отдельных организаций с четкими правилами взаи- модействия между ними и четкими политиками предоставления програм- мно-аппаратных ресурсов. Концепция облачных вычислений обеспечивает возможность любой компании использовать облачные сервисы для реше- ния собственных задач, оплачивая только те ресурсы, которые необходи- мы.
- *Различные области применения*. Грид-платформы предоставляют базу для развертывания вычислительной инфраструктуры. Облачные вычисления предоставляют интегрированный подход на всех уровнях предоставления информационных ресурсов: IaaS, PaaS, SaaS.
- *Расширение количества пользовательских интерфейсов*. Грид вычисления ориентированы на представление различных вычислительных ресурсов в гетерогенных вычислительных средах для решения конкретных задач. Таким образом, интерфейсы грид ориентированы на взаимодействие вычис- лительных инфраструктур на физическом уровне посредством API, кото- рым может воспользоваться только профессиональный программист. Об- лачные вычисления разрабатываются таким образом, чтобы предоставлять интерфейсы конечным пользователям через веб-доступ или посредством API. На каждом слое (IaaS, PaaS, SaaS) предоставляется свой собственный интерфейс. Повышение уровня абстракции позволяет обеспечить приме- нение облачных вычислений как на уровне отдельных пользователей, так и на уровне корпоративных клиентов.

  В общем и целом, грид вычисления обеспечивают объединение гетероген- ных вычислительных ресурсов в единую вычислительную среду. Это то, с чего начинаются и на чем основываются облачные

вычисления. Облачные вычисления обеспечивают более высокий уровень абстракции, предостав- ляя вычислительные ресурсы конечным пользователям (будь то частные клиенты или организации) в виде сервисов.

8.5. Перспективы развития технологий облачных вычислений

Парадигма облачных вычислений (cloud computing) ориентирована на модели, методы и технологии для предоставления пользователю удобного доступа к массиву конфигурируемых компьютерных и информационных ресурсов, которые могут быть быстро зарезервированы и высвобождены с минимальными действиями со стороны их провайдера [60]. Она обобщает и систематизирует ранее известные бизнес-модели IaaS (инфраструктура как сервис), PaaS (платформа как сервис) и SaaS (программное обеспечение как сервис), рассматривая их как вложенные механизмы, реализуемые в облаке [61]. При этом в качестве предоставляемых ресурсов могут выступать вычислительные системы, хранилища данных, системные приложения, средства разработки, прикладные программы и композитные приложения на их основе. Для развитых сред облачных вычислений характерна интеграция между собой различных бизнес-моделей, что связано, в первую очередь, с ориентацией на конкретные потребности пользователей в различных предметных областях. Для их классификации и анализа используется модель становления облачных вычислений CCMM (Cloud Computing Maturity Model) [62]. Она декларирует пять этапов развития облачных вычислений:

I. консолидация и модернизация доступных организации ресурсов;
II. виртуализация доступных ресурсов в рамках облачной парадигмы;
III. автоматизация процессов использования виртуальных ресурсов с обесп
IV. обеспечение поддержки автоматизированных сервисов (проведен

иеаудита, проверка отказоустойчивости, обеспечение качества);

V. полнофункциональная реализация облачной инфраструктуры на основе объединения сервисов, находящихся в различных облачных средах.

В настоящее время развитие облачных инфраструктур (и реализующих их технологий) в полной мере обеспечивает прохождение *первых трех* этапов. Уже четвертый этап, связанный с проведением аудита, измерением производительности и полным контролем над выполнением сервисов, к сожалению, находится в зачаточном состоянии, как в силу специфики измерения характеристик сервисов в распределенной среде, так и из-за отсутствия достаточно продвинутых глобальных проектов облачных вычислений, требующих полномасштабного решения данных задач. Исходя из модели CCMM, имеет смысл разделить технологии облачных вычислений на два класса (по аналогии с технологиями Грид, см. раздел 5): облачные вычисления первого поколения (C-1) и облачные вычисления второго поколения (C-2). Технологии первого поколения соответствуют этапам I-III CCMM и покрывают все три бизнес-модели (IaaS, PaaS, SaaS). Рынок таких решений уже достаточно насыщен, хотя продолжает эволюционировать весьма быстро [63]. Технологии C-2, напротив, во многом остаются предметом исследовательских проектов. Это связано с тем, что требования этапов IV-V CCMM реализуемы только при совокупном использовании возможностей современных информационных технологий и априорных знаний предметных областей, на которые ориентируется разработка. Иными словами, облачные вычисления второго поколения основываются на проблемно-ориентированных технологиях решения комплексных задач в распределенной вычислительной среде.

В настоящее время развитию технологий C-2 препятствует ряд технологических аспектов, например:

- Разнообразие и неоднородность предметно-ориентированных сервисов в неструктурированном облаке, что делает необходимым

развитие интеллектуальных технологий аннотирования, поиска, применения сервисов (формализация знаний).

- Декларативная запись композитных приложений, которая не допускает явной алгоритмической интерпретации. Для решения этих задачи требуются технологии эффективного управления их исполнением в распределенной среде.
- Использование коммуникационных сетей общего назначения со случайными вариациями загрузки сказывается на специфике учета, распределения и балансировки нагрузки в облачной среде. Как следствие, целесообразно планирование исполнения, квотирование и тарификация на вероятностной основе.
- Высокая инерционность среды облачных вычислений делает необходимым, в частности, развитие специальных технологий сопряжения с системами, работающими в режиме реального времени (например, для интерактивной визуализации в ситуационных центрах).

Совокупное преодоление указанных факторов становится возможным на основе симбиотического подхода в рамках концепции iPSE (Intelligent Problem Solving Environment) [64], которая ориентирована на развитие интеллектуальных технологий поддержки жизненного цикла проблемно- ориентированных сред распределенных вычислений. В настоящее время решения на основе данной концепции развиваются в РНЦ Курчатовский институт (среда NanoCloud), ЗАО «ИБС-Экспертиза», ЗАО «АйТи. Информационные технологии» и в Национальном исследовательском университете информационных технологий, механики и оптики.

В целом развитие технологий облачных вычислений, как первого, так и второго поколения, выполняется силами ряда организаций, ориентированных на различные технологические аспекты и приложения (что характерно для облачной парадигмы). В частности, к ним относятся

Институт системного анализа РАН, Институт системного программирования РАН, Вычислительный центр РАН, Томский государственный университет, Южно- Уральский государственный университет, и др.

Вопросы к главе 8.

6. Облачные вычисления. Что такое «облачные вычисления». Виды сервисов
7. Зачем нужны «облака». Частные и публичные облачные системы.
8. Недостатки облаков
9. Сравнение Грид и Облачных вычислений
10. Перспективы развития технологий облачных вычислений