Параллельная обработка данных

Лекция №1

Введение в предмет. Проблемы параллельных вычислений. Распараллеливание — один из способов увеличения скорости обработки.

План лекции:

- Цель изучения параллельной обработки.
- Концепция параллельной обработки
- Области применения параллельных вычислительных систем

Продолжительность курса:

	Очное	Заочное
лекций	30 часов	12 часов
практического обучения	60 часов	6 часов
самостоятельного обучения	90 часов	18 часов

Темы курса

- **Введение**
- **Тема 1. Архитектура высокопроизводительных систем**
- Классификация современных вычислительных систем (по Флинну). Мультипроцессоры и мультикомпьютеры. SMP, MPP, NUMA, PVP системы. Кластеры
- **Тема 2. Параллелизм на уровне операционных систем**
- Виды операционных систем. Распределенные операционные системы. Операционные системы для однопроцессорных компьютеров. Мультипроцессорные операционные системы. Мультикомпьютерные операционные системы. Системы с распределенной разделяемой памятью. Сетевые операционные системы.

Темы курса

- **Тема 3. Промежуточное программное обеспечение**
- Позиционирование программного обеспечения промежуточного уровня. Модели промежуточного уровня. Задачи, решаемые промежуточным уровнем. Организация связи. Идентификация. Синхронизация. Транзакции. Безопасность. Надежность
- **Тема 4. Параллельные программы**
- Параллелизм на прикладном уровне. Параллелизм данных. Параллелизм задач. Моделирование выполнения параллельных программ.

Темы курса

- **Тема 5. Принципы разработки параллельных алгоритмов**
- Методика разбиения алгоритмов на параллельные части. Этапы разработки параллельных алгоритмов. Примеры распараллеливания методов сортировки. Принципы распараллеливания. Масштабирование параллельных вычислений. Оценка параллельных алгоритмов
- **Тема 6. Системы параллельного программирования**
- OpenMP. PVM. MPI.

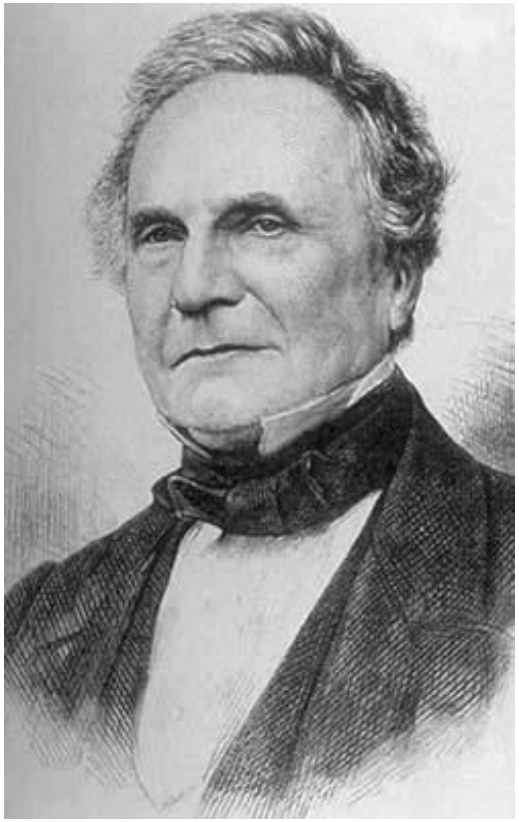
Литература

- Распределенные системы. Принципы и парадигмы/ Э. Таненбаум, М. ван Стеен. - СПб.: Питер, 2003 -877с.
- Технологии создания распределенных систем Для профессионалов/ А.А. Цимбал, М.П. Аншина, СПб.: Питер, 2003. г. -576 с.
- А.В.Богданов, В.В.Корхов, В.В.Мареев, Е.Н.Станкова Архитектуры и топологии многопроцессорных вычислительных систем. Курс лекций. Учебное пособие. – М.: ИНТУИТ.РУ «Интернет-Университет Информационных Технологий», 2004 http://www.intuit.ru/department/hardware/atmcs/
- Дж.Ортега Введение в параллельные и векторные методы решения линейных систем. – М.:Мир, 1991
- Г.Р.Эндрюс Основы многопоточного, параллельного и распределенного программирования. – М.: Издательский дом «Вильямс», 2003.
- С.Немнюгин, О.Стесик, Параллельное программирование для многопроцессорных вычислительных систем. СПб:, "БХВ-Петербург", 2002

Литература

- В.П.Гергель Теория и практика параллельных вычислений. Учебное пособие – М.: ИНТУИТ.РУ «Интернет-Университет Информационных Технологий», 2007 http://www.intuit.ru/department/calculate/paralltp/
- Воеводин В.В., Воеводин Вл.В.- Параллельные вычисления. – М.: "БХВ",2002
- К.Ю.Богачев Основы параллельного программирования. – М.: БИНОМ. Лаборатория знаний, 2003.
- Р.Миллер, Л.Боксер Последовательные и параллельные алгоритмы. – М.: БИНОМ. Лаборатория знаний, 2006
- Корнеев В.В. Параллельные вычислительные системы. – М.: “Нолидж”, 1999
- http://www.parallel.ru/
- www.top500.org
- wwwwww.www.ibmwww.ibm.www.ibm.com

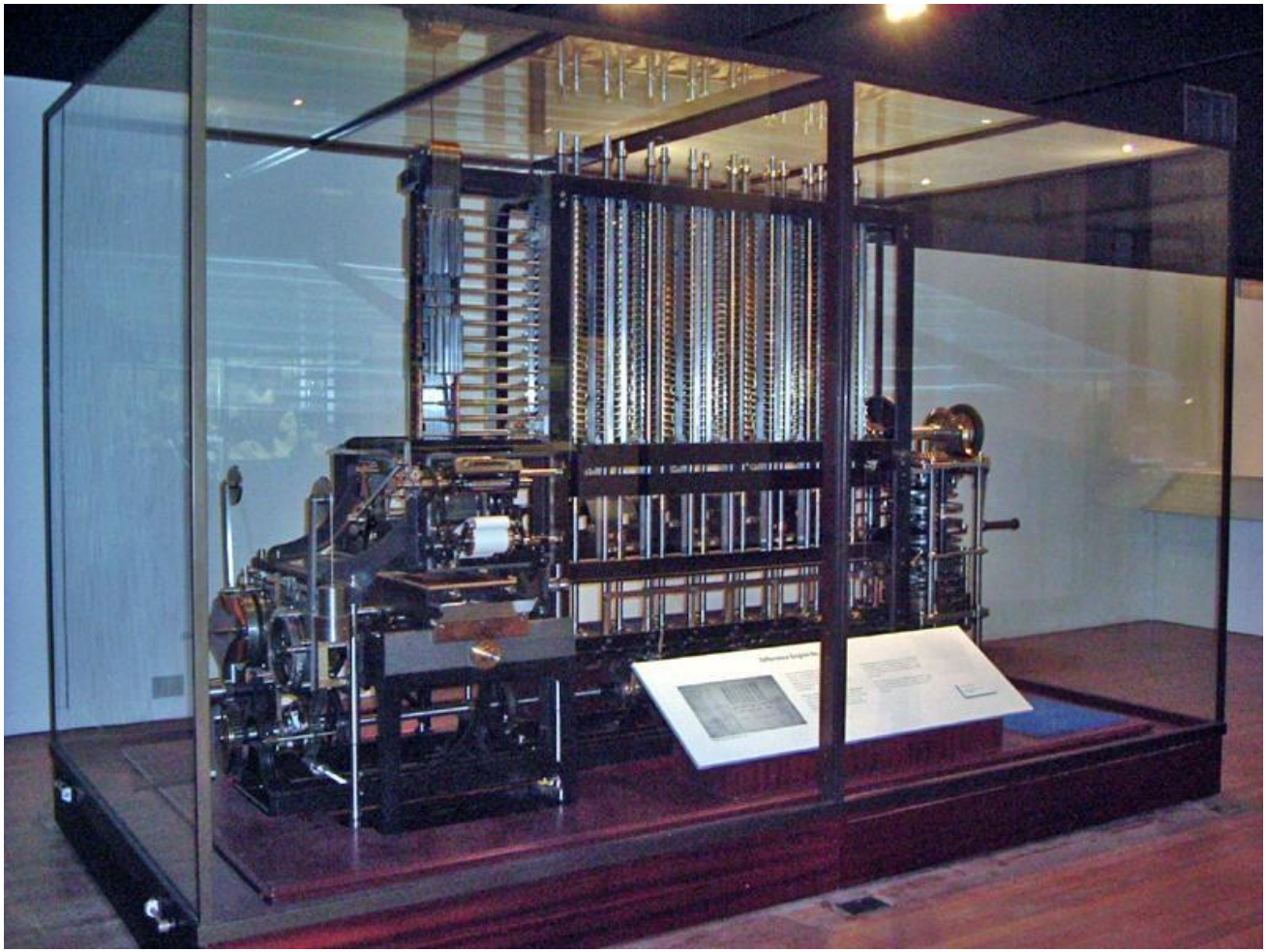
Чарльз Бэббидж (род.26 декабря 1791 года в Лондоне.: первое упоминание о параллелизме



- " В случае выполнения серии идентичных вычислений, подобных операции умножения и необходимых для формирования цифровых таблиц, машина может быть введена в действие с целью выдачи нескольких результатов одновременно, что очень существенно сократит весь объем процессов"

- В 1819 году Чарльз Бэббидж приступил к созданию малой разностной машины.
- 1822 году он закончил её строительство и выступил перед Королевским Астрономическим обществом с докладом о применении машинного механизма для вычисления астрономических и математических таблиц.
- Он продемонстрировал работу машины на примере вычисления членов последовательности.
- Работа разностной машины была основана на методе конечных разностей.
- Малая машина была полностью механической и состояла из множества шестерёнок и рычагов.
- В ней использовалась десятичная система счисления.
- Она оперировала 18-разрядными числами с точностью до восьмого знака после запятой и обеспечивала скорость вычислений 12 членов последовательности в 1 минуту. Малая разностная машина могла считать значения многочленов 7-й степени.

Чарльз Бэббидж: вычислительная машина



Определение параллелизма

- ***А.С. Головкин***

Параллельная вычислительная система - вычислительная система, у которой имеется по меньшей мере более одного устройства управления или более одного центрального обрабатывающего устройства, которые работают одновременно.

Определение параллелизма

- ***П.М. Коуги***

Параллелизм - воспроизведение в нескольких копиях некоторой аппаратной структуры, что позволяет достигнуть повышения производительности за счет одновременной работы всех элементов структуры, осуществляющих решение различных частей этой задачи.

Определение параллелизма

- ***Хокни, Джесссхоуп***

Параллелизм - способность к частичному совмещению или одновременному выполнению операций.

Развитие элементной базы и рост производительности параллельных вычислительных систем

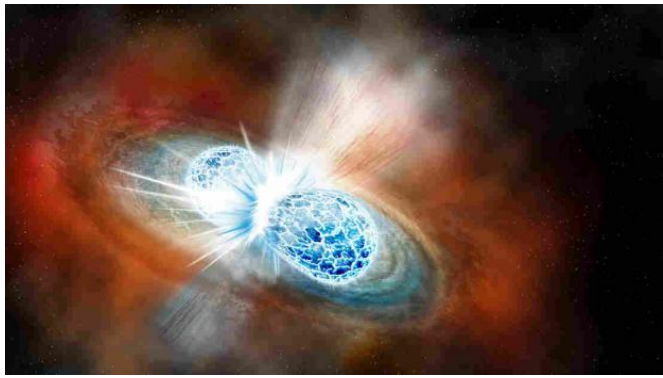
Период	Элементная база	Задержка	Быстрод-е элементной базы	Быстрод-е ЭВМ
1940-1950	Лампы	1 мкс	Рост В 1000 раз	Рост в 100000 раз
Начало 1960гг	Дискретные германиевые транзисторы	0,3 мкс		
Середина 1960 гг	Биполярные ИС малой степени интеграции	0,1 мкс=10 нс		
Середина 1970 гг	- «» -	До 1 нс		
Конец 1970	Переход к МОП	До 10 нс	Снижение	Рост

Области применения параллельных вычислительных систем

"Grand challenges" - круг фундаментальных и прикладных проблем, эффективное решение которых возможно только с использованием сверхмощных вычислительных ресурсов.

Области применения параллельных вычислительных систем

Физика



- Астрофизика (моделирование астрофизических объектов, таких как звездные недра и сверхновые)
- Физика высоких энергий (достижение детального понимания эффектов сильного ядерного взаимодействия, т.е. можно ли при помощи стандартных моделей описать явления на субядерном уровне)
- Физика ускорителей (точное моделирование поведения различных ускорителей)
- Ядерная физика (реалистическое моделирование характеристик кварк-глюонной плазмы)

Области применения параллельных вычислительных систем

Нанотехнологии

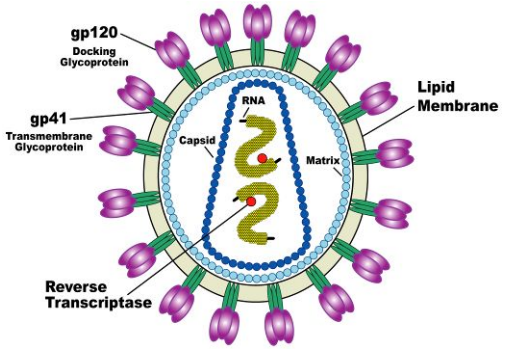
- Расчет однородных и неоднородных каталитических моделей
- Моделирование работы наномасштабных электронных устройств умеренной сложности
- Моделирование и предсказание механических и магнитных свойств простых наноструктурных материалов

Области применения параллельных вычислительных систем

Аэронавтика

- **Моделирование полета воздушно-космических судов** (например, маневры самолета, спуск-подъем шатла)
- **Моделирование подсистем жидкостных ракетных двигателей** (турбонасосы, комбинированные двигатели)
- **Моделирование авиационных систем** (моделирование с высокой точностью, системы поддержки принятия решений

Области применения параллельных вычислительных систем



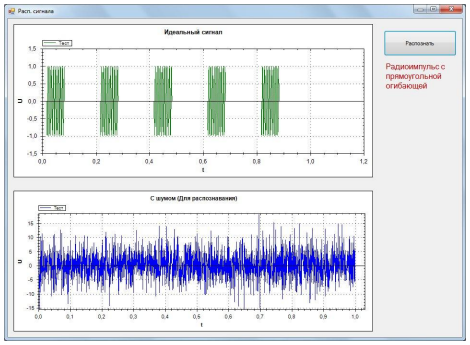
Науки о жизни

- Структурная и системная биология (Моделирование ферментного катализа, укладка протеина, транспорт ионов через клеточную мембрану)
- Пути трансдукции сигнала (Развитие расчетных моделей атомного уровня и сложных биомолекул)

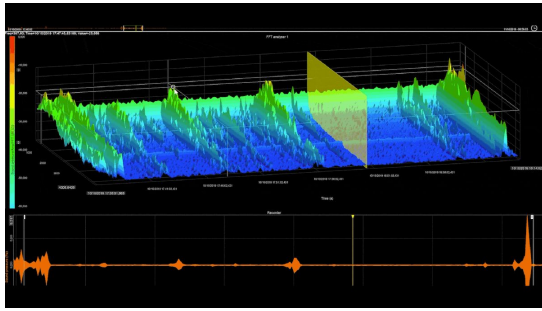
Области применения параллельных вычислительных систем

Национальная безопасность

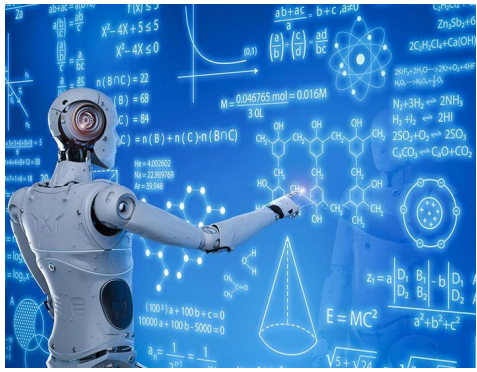
- Распознавание сигналов (радиотехническая разведка)



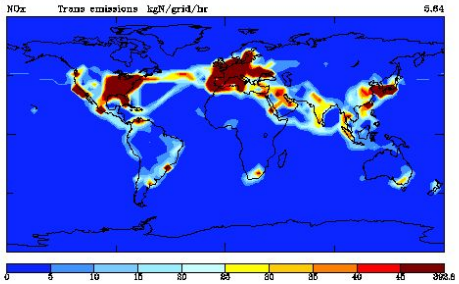
- Обработка сигналов и изображений, автоматическое распознавание целей (замена тестирования реальной цели при помощи рассеянного электромагнитного поля численным моделированием виртуальной цели)

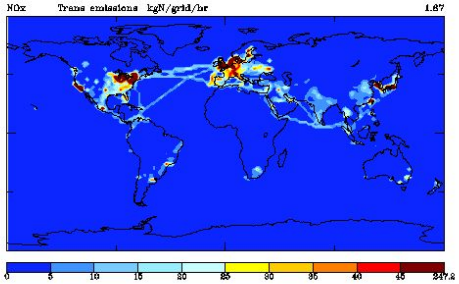


- Комплексное моделирование и тестирование систем вооружения



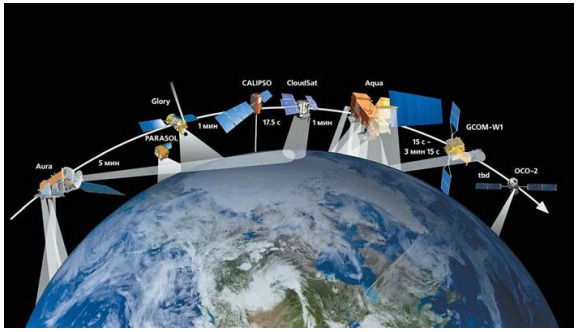
Области применения параллельных вычислительных систем





Науки о земле и атмосфере

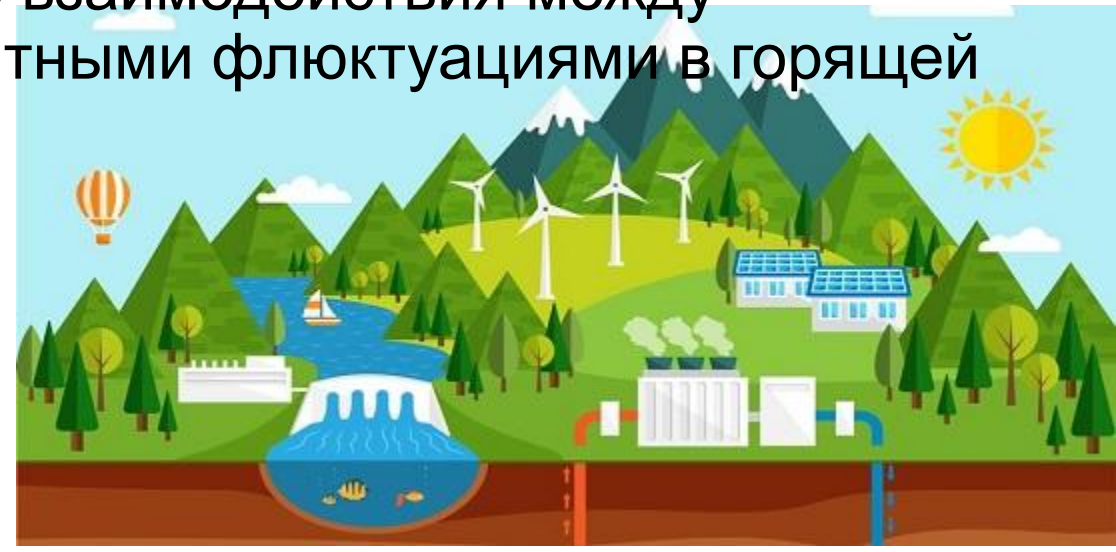
- Климатология
- Прогнозирование погоды и краткосрочного изменения климата (Динамическое предсказание частоты и интенсивности ураганов/тайфунов, тяжелых зимних штормов за 90 дней)
- Геология (Улучшение прогнозирование землетрясений)
- Наука о космосе (Реалистическое моделирование явлений на Солнце, распространение энергии, влияние на изменение магнитосферы, ионосферы и термосферы Земли)



Области применения параллельных вычислительных систем

Энергетика и окружающая среда

- Подземное загрязнение (Моделирование опасных рисков и распространения радионуклеидов и органических загрязнений под поверхностью)
- Магнитосинтез (Оптимизация баланса между саморазогревом плазмы и утечкой тепла, вызванного электромагнитной турбулентностью)
- Горение (Понимание взаимодействия между горением и турбулентными флюктуациями в горящей жидкости)



Новые области приложения параллельных алгоритмов:

Использованию компьютерных технологий и электронного документооборота для повышения эффективности бизнеса:

- построение централизованных вычислительных систем для критически важных приложений, связанных с
  - обработкой транзакций,
  - управлением базами данных
  - и обслуживанием телекоммуникаций

Новые области приложения параллельных алгоритмов:

- Системы для глобальных корпоративных вычислений:
  - централизованная система
  - работают практически все пользователи в корпорации
  - отличается повышенной производительностью
  - масштабируема
  - характеризуется минимально допустимым временем простоя

Типы параллельных систем:

- системы высокой надежности;
- системы для высокопроизводительных вычислений;
- многопоточные системы

Однако:

- границы между всеми типами до некоторой степени размыты
- часто система может иметь такие свойства или функции, которые выходят за рамки перечисленных типов
- при конфигурировании систем общего назначения, приходится выделять блоки, выполняющие все перечисленные функции

Системы высокой надежности

- Параллельные системы являются идеальной схемой для повышения надежности информационно-вычислительной системы
- Благодаря единому представлению, отдельные узлы или компоненты могут незаметно для пользователя заменять неисправные элементы, обеспечивая непрерывность и безотказную работу даже таких сложных приложений как базы данных
- Катастрофоустойчивые решения создаются на основе разнесения узлов многопроцессорной системы на сотни километров и обеспечения механизмов глобальной синхронизации данных между такими узлами

МВС для высокопроизводительных вычислений

- предназначены для параллельных расчетов
- обычно собраны из множества вычислительных узлов
- требуют постоянного согласования таких вопросов как
  - инсталляция
  - эксплуатация
  - одновременное управление большим числом вычислительных узлов
  - технические требований параллельного и высокопроизводительного доступа к одному и тому же системному файлу (или файлам)
  - межпроцессорная связь между узлами
  - координация работы в параллельном режиме

Эти проблемы проще всего решаются при обеспечении единого образа операционной системы для всего кластера. Однако реализовать подобную схему удается далеко не всегда, и обычно она применяется лишь для небольших систем

Многопоточные системы

- Многопоточные системы используются для обеспечения единого интерфейса к ряду ресурсов, которые могут со временем произвольно наращиваться (или сокращаться). Типичным примером может служить группа web-серверов.

Оценка производительности параллельных вычислительных систем

Главной отличительной особенностью многопроцессорной вычислительной системы является ее производительность:

количество операций, производимых системой за единицу времени.

Различают *пиковую* и *реальную* производительность.

- Пиковая производительность - величина, равная произведению пиковой производительности одного процессора на число таких процессоров в данной машине.
- Реальная производительность – производительность параллельной системы, достигаемая на данном приложении

Пиковая производительность

- вычисляется однозначно при условии, что все устройства вычислительной системы работают в максимально производительном режиме
- является базовой характеристикой, по которой производят сравнение высокопроизводительных вычислительных систем
- чем больше пиковая производительность, тем (теоретически) быстрее пользователь сможет решить свою задачу
- является величиной теоретической и, вообще говоря, недостижимой при запуске конкретного приложения, поскольку:
  - достигается только при отсутствии конфликтов при обращении к памяти
  - при равномерной загрузке всех устройств
  - в реальных условиях на выполнение конкретной программы влияют:
    - особенности структуры процессора,
    - системы команд,
    - состав функциональных устройств,
    - реализация ввода/вывода,
    - эффективность работы компиляторов

Способы оценки пиковой производительности компьютера

1. Опирается на число команд, выполняемых компьютером за единицу времени
   - единицей измерения, как правило, является MIPS (Million Instructions Per Second)
   - говорит о скорости выполнения компьютером своих же инструкций

   <u>недостатки:</u>
   - заранее не ясно, в какое количество инструкций отобразится конкретная программа
   - каждая программа обладает своей спецификой, и число команд от программы к программе может меняться очень сильно

Способы оценки пиковой производительности компьютера

2. Заключается в определении числа вещественных операций, выполняемых компьютером за единицу времени
   - Единицей измерения является Flops (Floating point operations per second) – число операций с плавающей точкой, производимых компьютером за одну секунду:

   Обычно используют производные единицы:
   - *MFLOPS* (мегафлопс – миллион операций с плавающей точкой в секунду)
   - *GFLOPS* (гигафлопс – миллиард (10^9) операций с плавающей точкой в секунду),
   - *TFLOPS* (терафлопс – триллион (10^{12}) операций с плавающей точкой в секунду

Преимущества:

более удобен для пользователя, поскольку, если ему известна вычислительная сложность программы, то, пользуясь этой характеристикой, пользователь может легко получить нижнюю оценку времени ее выполнения

Организация доступа к памяти

Время взаимодействия с памятью является одним из определяющих факторов, влияющих на производительность параллельных систем. Оно определяется:

- строением памяти,
- объемом памяти
- архитектурой подсистем доступа в память

Многоуровневая иерархическая память является наиболее эффективной систем доступа к памяти

- в качестве уровней используются:
  - регистры
  - регистровая память,
  - основная оперативная память,
  - кэш-память,
  - виртуальные и жесткие диски,
  - ленточные роботы

Принцип формирования иерархии памяти

- при повышении уровня памяти:
  - скорость обработки данных должна увеличиваться,
  - объем уровня памяти должен уменьшаться
- часто используемых данные хранятся в памяти верхнего уровня:
  - время доступа к ней минимально
  - память верхнего уровня обходится достаточно дорого => ее объем не может быть большим

Тесты, используемые для оценки производительности:

- ***LINPACK***
  - программа, предназначенная для решения системы линейных алгебраических уравнений с плотной матрицей и выбором главного элемента по строке.
  - используется для формирования списка Top500 – пятисот самых мощных компьютеров мира.
  - существенный недостаток: программа полностью распараллеливается, поэтому невозможно оценить эффективность работы коммуникационного компонента суперкомпьютера.

Тесты, используемые для оценки производительности:

24 Ливерморских цикла (***The Livermore Fortran Kernels, LFK***) и пакет ***NAS Parallel Benchmarks (NPB) :***

- позволяют оценить производительность компьютера действительно на реальных задачах
- отражают различные стороны реальных программ вычислительной гидродинамики

Тесты, используемые для оценки производительности:

- NAS тесты являются альтернативой LINPACK, поскольку они относительно просты и в то же время содержат значительно больше вычислений, чем, например, LINPACK или LFK
- Тестовые программы не могут дать полного представления о работе компьютера в различных режимах.
- Задача определения реальной производительности многопроцессорных вычислительных систем остается пока нерешенной.

Машина фон-Неймана

- Модель современной вычислительной системы, была сформулирова в первой половине XX века математиком Фон-Нейманом.
- Эта модель оказалась базовой при проектировании практически всех современных компьютерных систем, включая суперкомпьютеры.
- Модель состоит из четырех ключевых компонентов:

Машина фон-Неймана

- *Система памяти*, которая хранит как команды, так и данные. Известна как система с хранимой программой. Доступ к этой памяти осуществляется с помощью регистра адреса, куда подсистема памяти помещает адрес ячейки памяти и регистра данных , куда она помещает данные из ячейки с указанным адресом.
- По крайней мере один *блок обработки данных,* наиболее известный как арифметико-логическое устройство. Эти блоки чаще называют центральными процессорами (CPU). Этот блок отвечает за выполнение всех команд. Процессор также имеет небольшой объем памяти, называемый набором регистров.
- *Блок управления*, отвечающий за операции между компонентами модели. Включает в себя счетчик команд, содержащий следующую команду для загрузки, и регистр команд, в котором находится текущая команда.
- Системе необходим энергонезависимый способ хранения данных, а также выдача их пользователю и принятия входных данных. Это осуществляется подсистемой ввода-вывода (I/O)/

Принципы архитектуры фон Неймана

1. принцип программного управления выполнением программы
2. принцип хранимой в памяти программы

1+2 легли в основу понятия ***фон-Неймановской архитектуры***, широко использующей *счетчик команд*.

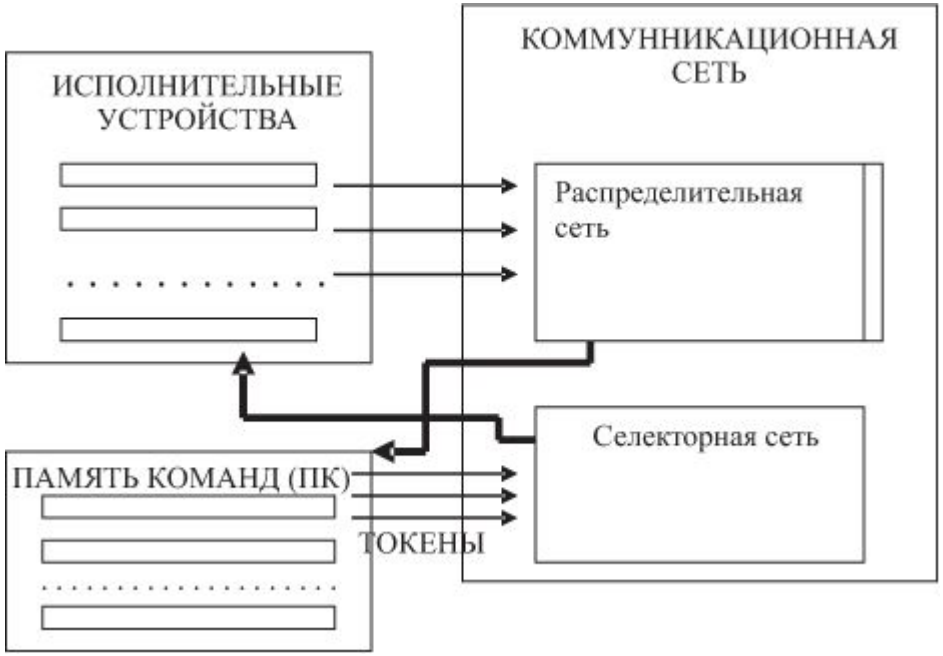
Счетчик команд отражает "узкое горло", которое ограничивает поток команд, поступающих на исполнение, их последовательным анализом

«Не-фон-Неймановская" архитектура

- допускает одновременный анализ более одной команды
- отсутствует счетчик команд
- порядок выполнения команд определяется наличием исходной информации для выполнения каждой из них
- принципиально возможно выполнение нескольких команд таким же количеством свободных процессоров

Вычислительные системы «»не-фон-Неймановской» архитектуры управляются *потоком данных (**data flow**).*

Общая схема потоковых ВС



Память команд (ПК)

- содержит программу или ее часть (сегмент)
- состоит из ячеек команд

Структура команд:
{код операции, операнд 1, ..., операнд L, адрес результата 1, ..., адрес результата M}

Результаты выполнения одних команд в качестве операндов могут поступать в текст других команд

Возможно альтернативное задание адреса результата (ИЛИ — ИЛИ) с помощью команд проверки условия

Команда не готова к выполнению, если в ее тексте отсутствует хотя бы один операнд

Ячейка, обладающая полным набором операндов, переходит в возбужденное состояние и передает в селекторную сеть информационный пакет (токен), содержащий код операции и необходимую числовую и связную информацию.

- селекторная сеть передает токен на одно из исполнительных устройств
- исполнительное устройство выполняет операцию и выдает результирующий пакет в распределительную сеть
- результирующий пакет содержит результат вычислений и адреса назначения в памяти команд (ПК)
- результат поступает в ПК согласно полученным адресам, создавая возможность активизации новых ячеек
- после выдачи токена в селекторную сеть операнды в тексте команды уничтожаются, что обеспечивает повторное выполнение команды в цикле, если это необходимо

Селекторная и распределительная сети образуют ***коммуникационную сеть*** ВС

Ожидаемая сверхвысокая производительность такой системы может быть достигнута за счет

- одновременной и независимой активизации большого числа готовых команд
- допущении о бесконфликтной передаче пакетов по сетям
- параллельной работы многих исполнительных устройств

«Не-фон-Неймановские" архитектуры не обрели технического воплощения для массового применения в "классическом", исполнении.

Но:

при совместном решении общей задачи

взаимодействие процессоров

и их синхронизация

при использовании общих данных

основаны на анализе готовности данных для обработки

т.е. реализуется принцип *data flow*.

Уровни параллельных систем

Под параллельными вычислениями понимаются процессы обработки данных, в которых одновременно могут выполняться несколько операций компьютерной системы

Выделяют следующие уровни параллельных систем:

- Аппаратный уровень – включающий в себя компьютеры и их организацию;
- Уровень ОС – объединяет ОС компьютеров;
- Промежуточный уровень – включает службы обеспечивающие работу распределенных приложений;
- Уровень параллельных приложений – сами параллельные приложения

Организация параллельных вычислений

Возможные режимы выполнения независимых частей программы:

- **многозадачный режим** (режим разделения времени), при котором для выполнения нескольких процессов используется единственный процессор
  - является псевдопараллельным, поскольку активным (исполняемым) может быть один, единственный процесс, а все остальные процессы находятся в состоянии ожидания своей очереди
  - может повысить эффективность организации вычислений (например, если один из процессов не может выполняться из-за ожидания вводимых данных, процессор может быть задействован для выполнения другого, готового к исполнению процесса
  - проявляются многие эффекты параллельных вычислений (необходимость взаимоисключения и синхронизации процессов и др.), и, => этот режим может быть использован при начальной подготовке параллельных программ

Организация параллельных вычислений

Возможные режимы выполнения независимых частей программы:

- ***параллельный режим:***
  - ***в один и тот же момент может выполняться несколько команд обработки данных.***
  - ***может быть обеспечен не только при наличии нескольких процессоров, но и при помощи конвейерных и векторных обрабатывающих устройств***

Организация параллельных вычислений

Возможные режимы выполнения независимых частей программы:

- ***распределенные вычисления :***
  - ***предполагает параллельную обработку данных***
  - ***используется несколько обрабатывающих устройств***
  - ***устройства достаточно удаленны друг от друга***
  - ***передача данных по линиям связи приводит к существенным временным задержкам***
  - ***эффективная обработка данных при таком способе организации вычислений возможна только для параллельных алгоритмов с низкой интенсивностью потоков межпроцессорных передач данных***

Уровни параллельных систем: аппаратный уровень

Достижение параллелизма возможно, если соблюдаются следующие принципы построения аппаратной вычислительной среды:

- **независимость функционирования отдельных устройств ЭВМ**
  - относится в равной степени ко всем основным компонентам вычислительной системы: к устройствам ввода-вывода, обрабатывающим процессорам и устройствам памяти;
- **избыточность элементов вычислительной системы** – организация избыточности может осуществляться в следующих основных формах:
  - **использование специализированных устройств**, таких, например, как отдельные процессоры для целочисленной и вещественной арифметики, устройства многоуровневой памяти (регистры, кэш);
  - **дублирование устройств ЭВМ** путем использования, например, нескольких однотипных обрабатывающих процессоров или нескольких устройств оперативной памяти
- **необычные архитектурные решения**, направленные на повышение производительности (работа с векторными операциями, организация быстрого обмена сообщениями между процессорами или организация глобальной памяти в многопроцессорных системах и др.)

Архитектуры параллельных систем

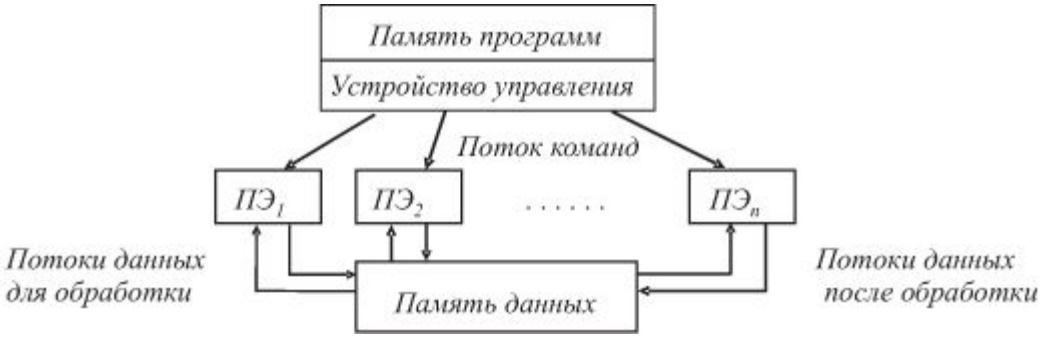
Классификация Флинна

- предложена в 1966 году ***М.Флинном*** (Flynn)
- в основу было положено понятие ***потока***, под которым понимается последовательность элементов, команд или данных, обрабатываемая процессором
- основана на рассмотрении ***числа потоков инструкций*** и ***потоков данных***
- описывает четыре архитектурных класса:
  - **SISD** = Single Instruction stream Single Data stream
    MISD = Multiple Instruction stream Single Data stream
    SIMD = Single Instruction stream Multiple Data stream
    MIMD = Multiple Instruction stream Multiple Data stream)

SISD(Single Instruction, Single Data Stream)

- предполагает последовательную обработку команд и данных
- команды поступают одна за другой (за исключением точек ветвления программы), и для них из ОЗУ или регистров так же последовательно поступают операнды
- одной команде (операции) соответствует один необходимый ей набор операндов (как правило, два для бинарных операций)
- используется в ЭВМ классической архитектуры

SIMD(векторные компьютеры)



- одной командой обрабатывается набор данных (вектор) и вырабатывается множество результатов
- векторы данных, распределены между *процессорными (обрабатывающими) элементами ПЭ или процессорами*
- используется в векторных и матричных вычислительных системах

MISD

- аналог - работа банка, когда в любого терминала можно подать команду и обработать имеющийся банк данных; у база данных одна, команд много => множественныq поток команд и одиночный поток данных
- вычислительных машин такого класса практически нет
- единственный пример - *систолический массив* процессоров:
  - процессоры находятся в узлах регулярной решетки
  - роль ребер которой играют межпроцессорные соединения
  - все процессорные элементы управляются общим тактовым генератором
  - в каждом цикле работы каждый процессорный элемент получает данные от своих соседей, выполняет свою команду и передает результат соседям
- к этой категории принято относить векторный конвейер

MIMD(локальные, глобальные сети)

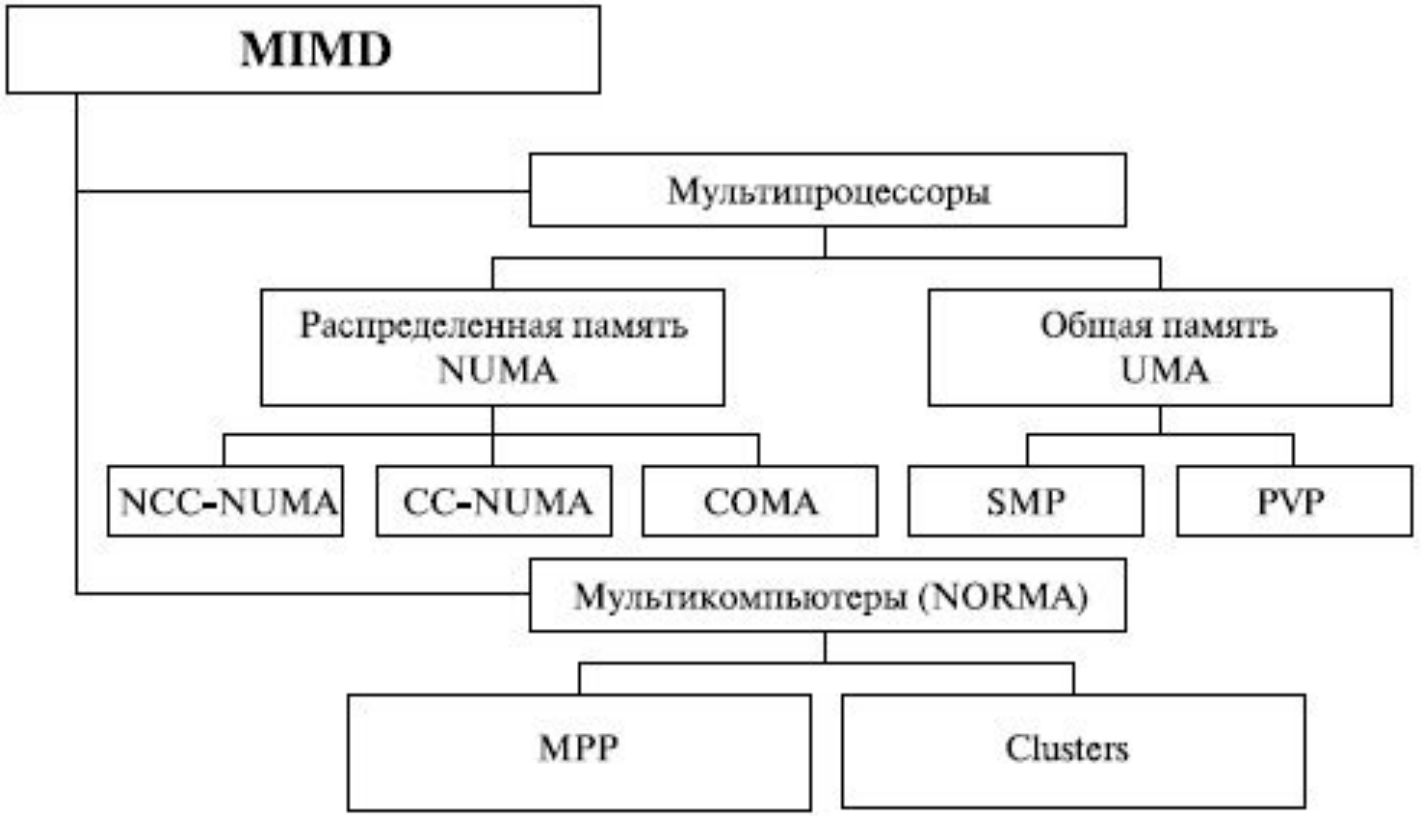
- MIMD (multiple instruction stream / multiple data stream) - множественный поток команд и множественный поток данных
- параллельно выполняют несколько потоков инструкций над различными потоками данных
- команды и данные связаны, потому что они представляют различные части одной и той же задачи
- используется при построении
  - симметричные параллельные вычислительные системы,
  - рабочие станции с несколькими процессорами,
  - кластеры рабочих станций
- большое разнообразие попадающих в данный класс систем (практически все ВС, предназначенные для параллельных вычислений) делает классификацию Флинна не полностью адекватной и требует замены

Мультипроцессоры и мультикомпьютеры

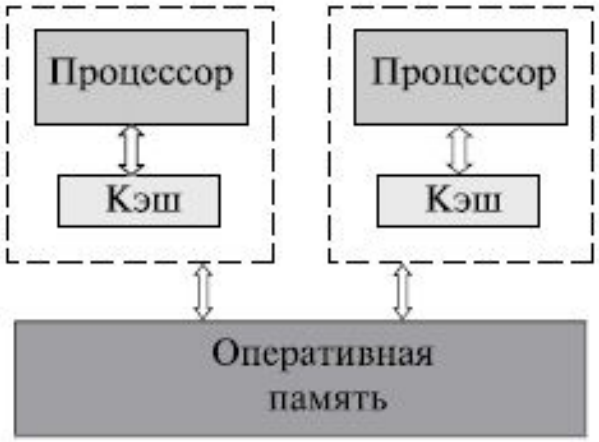
Для класса MIMD предложена новая классификация:

- основывана на способах организации оперативной памяти
- различать два важных типа многопроцессорных систем:
  - **multiprocessors** - мультипроцессоры или системы с общей разделяемой памятью
  - **multicomputers** - мультикомпьютеры или системы с распределенной памятью

Классификация многопроцессорных вычислительных систем



Мультипроцессоры



а)

- используют единую (централизованную) общую память (shared memory)
- обеспечивают однородный доступ к памяти (uniform memory access или UMA)
- служиат основой для построения векторных параллельных процессоров (parallel vector processor или PVP) и симметричных мультипроцессоров (symmetric multiprocessor или SMP)
- суперкомпьютер Cray T90 - пример PVP архитектуры
- IBM eServer, Sun StarFire, HP Superdome, SGI Origin – примеры SMP архитектуры

SMP системы

Основные преимущества SMP-систем:

- простота и универсальность для программирования:
  - архитектура SMP не накладывает ограничений на модель программирования
  - обычно используется модель параллельных ветвей, когда все процессоры работают независимо друг от друга
  - можно реализовать и модели, использующие межпроцессорный обмен.
- использование *общей памяти* увеличивает *скорость* обмена данными
- пользователь имеет доступ сразу ко всему объему памяти
- для SMP-систем существуют довольно эффективные средства автоматического распараллеливания;
- простота эксплуатации
  - SMP-системы, как правило используют систему кондиционирования, основанную на воздушном охлаждении, что облегчает их техническое обслуживание;
- относительно невысокая цена

Недостатки SMP систем

- *поддержка однозначности (когерентности) содержимого разных кэшей:*
  - при наличии общих данных копии значений одних и тех же переменных могут оказаться в кэшах разных процессоров
  - если один из процессоров выполнит изменение значения разделяемой переменной, то значения копий в кэшах других процессоров окажутся не соответствующими действительности и их использование приведет к некорректности вычислений
  - когерентность кэшей обычно реализуется на аппаратном уровне:
    - после изменения значения общей переменной все копии этой переменной в кэшах отмечаются как недействительные и последующий доступ к переменной потребует обязательного обращения к основной памяти
    - необходимость обеспечения когерентности приводит к некоторому снижению скорости вычислений и затрудняет создание систем с достаточно большим количеством процессоров

Недостатки SMP систем

- *проблемы взаимоисключения и синхронизации*
  - Наличие общих данных при параллельных вычислениях приводит к необходимости синхронизации взаимодействия одновременно выполняемых потоков команд
  - если изменение общих данных требует для своего выполнения некоторой последовательности действий, то необходимо обеспечить взаимоисключение (mutual exclusion), чтобы эти изменения в любой момент времени мог выполнять только один командный поток.

Недостатки SMP систем

- *системы с общей памятью плохо масштабируются*
  - причины:
    - шина способна обрабатывать только одну транзакцию, вследствие чего возникают проблемы разрешения конфликтов при одновременном обращении нескольких процессоров к одним и тем же областям *общей физической памяти*. Вычислительные элементы начинают друг другу мешать
    - системная шина имеет ограниченную (хоть и высокую) пропускную способность (ПС) и ограниченное число слотов

NUMA - non-uniform memory access

б)

- общий доступ к данным обеспечивается при физически распределенной памяти
- при этом, длительность доступа уже не будет одинаковой для всех элементов памяти => происходит неоднородный доступ к памяти (non-uniform memory access)

По существу архитектура NUMA является MPP (массивно-параллельной) архитектурой, где в качестве отдельных вычислительных элементов берутся SMP (симметричная многопроцессорная архитектура) узлы.

Доступ к памяти и обмен данными внутри одного SMP-узла осуществляется через локальную память узла и происходит очень быстро, а к процессорам другого SMP-узла тоже есть доступ, но более медленный и через более сложную систему адресации.

NUMA - non-uniform memory access

Системы, имеющие архитектуру NUMA, подразделяются на:

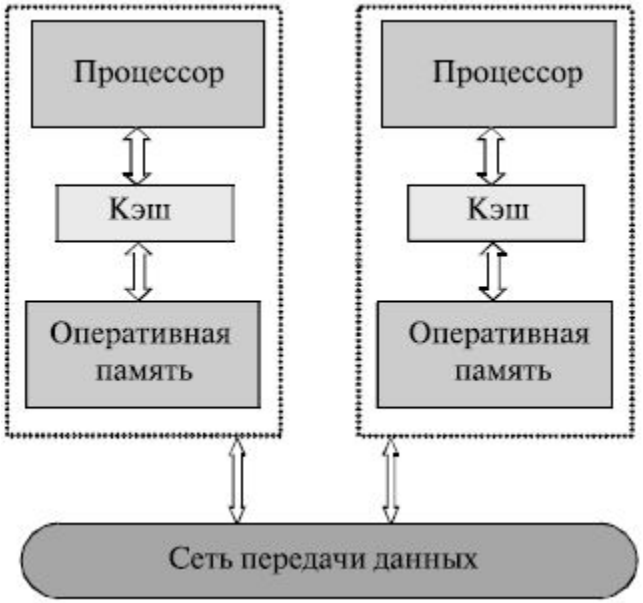
- системы, в которых для представления данных используется только локальная кэш-память имеющихся процессоров (cache-only memory architecture или COMA
- системы, в которых обеспечивается когерентность локальных кэшей разных процессоров (cache-coherent NUMA или CC-NUMA); среди таких систем: SGI Origin 2000, Sun HPC 10000, IBM/Sequent NUMA-Q 2000;
- системы, в которых обеспечивается общий доступ к локальной памяти разных процессоров без поддержки на аппаратном уровне когерентности кэша (non-cache coherent NUMA или NCC-NUMA); например, система Cray T3E

PVP (Parallel Vector Process) - параллельная архитектура с векторными процессорам

- Основным признаком PVP-систем является наличие специальных векторно-конвейерных процессоров, в которых предусмотрены команды однотипной обработки векторов независимых данных, эффективно выполняющиеся на конвейерных функциональных устройствах.
- Как правило, несколько таких процессоров (1-16) работают одновременно с общей памятью (аналогично SMP) в рамках многопроцессорных конфигураций. Несколько узлов могут быть объединены с помощью коммутатора (аналогично MPP).
- Поскольку передача данных в векторном формате осуществляется намного быстрее, чем в скалярном (максимальная скорость может составлять 64 Гбайт/с, что на 2 порядка быстрее, чем в скалярных машинах), то проблема взаимодействия между потоками данных при распараллеливании становится несущественной.
- То, что плохо распараллеливается на скалярных машинах, хорошо распараллеливается на векторных.
- Таким образом, системы PVP-архитектуры могут являться машинами общего назначения (general purpose systems).
- Однако, поскольку векторные процессоры весьма дорого стоят, эти машины не могут быть общедоступными.

- Парадигма программирования на PVP-системах предусматривает векторизацию циклов (для достижения разумной производительности одного процессора) и их распараллеливание (для одновременной загрузки нескольких процессоров одним приложением).
- На практике рекомендуется выполнять следующие процедуры:
  - производить векторизацию вручную, чтобы перевести задачу в матричную форму. При этом, в соответствии с длиной вектора, размеры матрицы должны быть кратны 128 или 256.
  - работать с векторами в виртуальном пространстве, разлагая искомую функцию в ряд и оставляя число членов ряда, кратное 128 или 256.
  - За счет большой физической памяти (доли терабайта) даже плохо векторизуемые задачи на PVP-системах решаются быстрее, чем на машинах со скалярными процессорами.

Мультикомпьютеры



- мультикомпьютеры -многопроцессорные системы с распределенной памятью (no-remote memory access или NORMA)
- не обеспечивают общего доступа ко всей имеющейся в системах памяти
- принципиальное отличие от системам с распределенной общей памятью:
  - каждый процессор системы может использовать только свою локальную память
  - для доступа к данным, располагаемым на других процессорах, необходимо явно выполнить операции передачи сообщений (message passing operations)
- типы мультикомпьютеров:
  - массивно-параллельные системы (massively parallel processor или MPP)
  - кластеров (clusters).

MPP (massive parallel processing) – ***массивно-параллельная архитектура***

- память физически разделена
- система строится из отдельных модулей, содержащих процессор, локальный банк операционной памяти (ОП), *коммуникационные процессоры* (рутеры) или *сетевые адаптеры*, иногда – жесткие диски и/или другие устройства ввода/вывода
- доступ к банку ОП из данного модуля имеют только процессоры (ЦП) из этого же модуля
- модули соединяются специальными коммуникационными каналами
- пользователь может определить логический номер процессора, к которому он подключен, и организовать обмен сообщениями с другими процессорами
- используются два варианта работы операционной системы (ОС):
  1. полноценная операционная система (ОС) работает только на управляющей машине (front-end), на каждом отдельном модуле функционирует сильно урезанный вариант ОС, обеспечивающий работу только расположенной в нем ветви параллельного приложения
  2. на каждом модуле работает полноценная UNIX-подобная ОС, устанавливаемая отдельно

MPP - преимущества

- хорошая *масштабируемость*
  - каждый процессор имеет доступ только к своей *локальной памяти* => не возникает необходимости в потактовой синхронизации процессоров

Практически все рекорды по производительности на сегодня устанавливаются на машинах именно такой архитектуры, состоящих из нескольких тысяч процессоров (ASCI Red, ASCI Blue Pacific)

MPP - недостатки

- низкая *скорость межпроцессорного обмена, поскольку* нет общей среды для хранения данных, предназначенных для обмена между процессорами
- требуется специальная техника программирования для реализации обмена сообщениями между процессорами;
- каждый процессор может использовать только ограниченный объем локального банка памяти;
- недостатков требуются значительные усилия для того, чтобы максимально использовать системные ресурсы.
- высокая цена программного обеспечения