Лекция 1

Современные подходы, классы и особенности

Пути достижения параллелизма:

* Независимость функционирования отдельных устройств компьютеров
* Избыточность элементов вычислительной системы (дублирование, резервирование)
* Конвейерная реализация обрабатывающих устройств

Сдерживающие факторы:

* Высокая стоимость параллельных вычислительных систем
* Необходимость обобщения последовательных программ
* Потери производительности при организации параллелизма
* Постоянное совершенствование последовательных компьютеров
* Зависимость эффективности параллелизма от учета архитектуры параллельных систем

Режимы выполнения независимых частей программы:

* Многозадачный режим (разделение времени). Используется единственные процессор, такой режим псевдопараллельный, так как активный только один процесс, все остальные находятся в очереди и ожидают доступности ресурсов. Использование режима разделения времени позволяет повысить эффективность организации вычислений за счет разделения во времени вычислительных операций и операций связанными с вводом и выводом.
* Параллельные выполнение. Предполагает, что в один момент времени может выполнятся несколько команд обработки данных, такой режим обеспечивается не только при наличии нескольких процессоров, но в случае реализации конвейерных или векторных обрабатывающих устройств.
* Распределенные вычисления. Данный термин используется для указания параллельной обработки данных для которой используется несколько обрабатывающих устройств, удаленных друг от друга, в которых передача данных по линиям связи приводит к существенным временным задержкам. Распределенные вычисления позволяют эффективно обрабатывать данные только для параллельных алгоритмов с низкой интенсивности потоков межпроцессорных передач данных. Такие задачи обычно решаются на многомашинных вычислительных комплексов, когда несколько ЭВМ связываются по локальным или глобальным информационным сетям.

Мультипроцессорные системы (многопроцессорные)

Многопроцессорные системы с общей разделяемой памятью. Такие системы принято называть симметричными многопроцессорными системами. SMP. Основной чертой является то, что каждый процессор имеет прямой и равноправный доступ к любой точке общей памяти. К памяти процессоры подключаются через общую системную шину, при этом упрощается межпроцессорное взаимодействии, упрощается программирование, так как программа работает в едином адресном пространстве.

* + Такая архитектура не пригодна для создания масштабных систем из-за большого числа конфликтов при обращении к общей памяти.
  + Во всех современных машинах иерархическая память. В связи с этим возникает проблема синхронизации информации расположенной в отдельных элементах кэш памяти и основной памяти системы. Современные системы состоят из однородных микропроцессоров с массивом общей памяти. Подключается при помощи шины или комутатора

Для построения масштабируемых систем используются кластерные системы или архитектура типа NUMA. Неоднородный доступ к памяти. В такой архитектуре память распределена физически, но логически общедоступна. С одной стороны это позволяет работать всем процессорам с единым адресным пространством, с другой стороны увеличивает масштабируемость системы, сокращает вероятность конфликтов при доступе к одной и той же ячейке. Когерентность решается аппаратно, в отличии от чистой SMP система с NUM архитектурой вводится 3-х уровневая память:

* Кэш память процессора
* Локальная оперативная память
* Удаленная оперативная память.

Время обращение к элементам памяти разного уровня отличается на порядки, что усложняет написание эффективных параллельных программ. В любой момент времени модули соединенные с помощью высокоскоростного коммутатора могут получать доступ к произвольному элементу удаленной памяти. Масштабируемость NUMa систем ограничивается размером адресного пространства, аппаратуры для поддержки когерентности, и аппаратуры для управления числа процессоров. В настоящее время максимально можно поставить 256 процессоров. Общая производительность около 200 млрд оп\сек.

У любой системы с разделяемой памятью цена растет очень быстро по сравнению с ростом производительности.

Системы с распределенной памятью. MPP

Такие системы представляют собой много процессорные системы с распределенной памяти, которые с помощью коммутационной среды объединяются в выч узлы. Каждый узел состоит из одного или нескольких процессорных элементов, собственной ОЗУ, коммутационного оборудования и подсистемы ввода вывода. Каждый узел обладает всеми компонентами для независимого функционирования. На каждом узле может функционировать либо полноценная ОС, либо ее урезанный вариант, поддерживающие базовые функции ядра.

Процессоры имеют прямой доступ только к своей локальной памяти. Доступ к памяти других узлов реализуется с помощью механизма передачи сообщений. Такая архитектура позволяет устранить конфликты при обращении к памяти и проблему когерентности кэшей. Считается, что имеется возможность наращивания числа процессоров с увеличением производительности.

Основной плюс – высокая масштабируемость.

Система с распределенной памятью подходит для выполнения параллельно независимых программ с малой интенсивностью передачи информации.

Для решения проблем возникающих в выше представленных архитектурах. Решаются векторные системы. В их состав включаются векторные конвейерные системы. Они имеют векторно-конвейерные процессоры. Несколько таких процессоров работают одновременно над общей памятью в рамках многопроцессорных конфигураций. PVP системы являются гибридными системами.

Основной недостаток – цена.

Исследования:

* Исследования в области теории параллельных алгоритмов
* Исследования в области формальных языков методов и технологии параллельного программирования и оптимизации параллельных программ
* Исследования в области теории управления параллельными вычислительными процессами

Эффективность решения конкретной задачи является особенностью реализации параллельности процесса. В целом процесс параллельного решения задачи можно определить как одновременное выполнение отдельных фрагментов задачи, распределенном в вычислительной задаче. Для организации таких вычислений решаемая задача должна обладать свойствами параллелизма.

Свойства параллелизма

* Задачи с естественным параллелизмом. Такие задачи предполагают наличие в процессе своего решения совокупность операции, которые могут выполнятся одновременно без дополнительной передачи информации. Процесс решения задачи с естественным параллелизмом разделяется на несколько процессов, которые выполняются независимости, при этом не требуются специфики вычислений.
* Задачи с параллелизмом множества объектов. Выполняется обработка различных или однотипных объектов по одной и той же программе. В отличии от задач с естественным параллелизмом встречаются в ситуации, когда отдельные участки вычислений должны выделятся по разному для объектов. Основной характеристикой является ранг т.е. количество параметров по которым должна вестись обработка. В качестве ранга задачи может выступать количество разнотипных объектов в системе.
* Задачи с параллелизмом независимых ветвей. Наиболее разработанные тип параллелизма. При решении крупных задач могут быть выделены независимые части или ветви. Условия независимости ветви задачи У от ветки Х являются:
  + Между ветвями отсутствуют функциональные связи. Выход Х не является вход Y
  + Между ветвями нет связи по памяти
  + Независимость в программном отношении
  + Взаимонезависимость по управлению.

Все задачи решаемые одновременно могут быть связаны по данным и по управлению в общей вычислительной среде. При этом задачи разделяются по степени связывания. Степень связывания определяет зернистость параллелизма. Пусть существует некоторая сложная задача, которая может быть разбита но множество простых задач, тогда задача называется несвязанной если все элементы матрицы связности являются нулевыми. Одиночные задачи соединяются в сложную несвязную задачу в виде набора простых, вычисления при решении простых задач ведутся независимо друг от друга, обмен происходит довольно редко. Объектом распралалеливания являются простые задачи. Такие вычисления являются крупно зернистыми.

Если часть элементов матрицы связности равны нулю, то сложная задача состоящая из набора простых называется слабо связанной. Такие задачи образуют обширный класс, в котором простые задачи объединяются либо функциональными связями, либо связями по данным. Принято считать, что общий объем взаимодействия по данным меньше объема вычисления, то появляющийся набор простых задач можно считать общей слабосвязной задачей. Наличие связности между задачами не позволяет эффективно решать ее на распределенных устройствах. Объектом распараллеливания является набор вызываемых процедур. Организация управления вычисления определяется программистом. Такую организацию принято называть средне-зернистым параллелизмом.

Сложная задача, состоящая из набора простых называют сильно связанное, если все элементы имеют не нулевые значения. В сильносвязных задачах число обменных взаимодействий по управлению данных сопоставимо с объемом вычислением в простых задачах. В предельном случае на каждом шаге вычислений выполняется обмен данными между простыми задачами. Распараллеливание выполняется на уровне внутренних блоков (циклов). Это мелкозернистая организация.

Методы и средства организации параллельных вычислений определяются уровнем организации параллельных вычислений. Традиционно принято выделять 4 уровня:

* Уровень заданий.
* Уровень программ.
* Уровень команд. Реализуется средствами аппаратного ядра ОС.
* Арифметический уровень. Обеспечивается параллельное или конвейерное исполнение команды. Реализуется аппаратными средствами.

Модели параллельного программирования

Определяют набор свойств, присущие разрабатываемому алгоритму для создания максимально эффективной программы. Основными свойствами являются:

* Параллелизм. Возможность выполнят несколько действий одновременно
* Масштабируемость. Возможность увеличения производительности при увеличении числа процессоров
* Локальность. Предпочтения использование локальной памяти по сравнению с удаленной
* Модульность. Декомпозиция сложных программ на простые компоненты

Модели:

* Задача\канал
* Передача сообщения
* Разделяемая общая память
* Параллелизм данных