Раздел 8
Классификация архитектур ВС

В данном разделе изучаются классы вычислительных систем (SISD, MISD, SIMD, MIMD); производится классификация архитектур параллельных систем (векторные процессоры, матричные процессоры, архитектура VLIW, системы с общей памятью и распределённой памятью); исследуются эволюция и тенденции развития микропроцессорных архитектур.

Слайд 3

Известно более десятка различных классификаций вычислительных систем (подробности на сайте www.parallel.ru или в книге Воеводин В.В., Воеводин Вл.В., “Параллельные вычисления” – СПб: БХВ-Петербург, 2004). Однако ключевой является классификация Флинна, на базе которой возникли новые классификации, детализирующие её исходные классы.

Слайд 4

Флинн в 1966 году предложил рассматривать и потоки команд, и потоки данных либо как одиночные, либо как множественные. Так появились четыре класса вычислительных систем: SISD, MISD, SIMD, MIMD. Просто и адекватно.

Классические фон-неймановские машины попадают в тривиальный класс SISD, в котором одиночный поток инструкций обрабатывает одиночный поток данных. Классические языки высокого уровня (такие, как C++) также ориентированы на программирование в классе SISD. В настоящее время выпуск SISD процессоров почти прекращён из-за их низкой производительности, которая обусловлена низким уровнем параллелизма вычислений (используется только мелкозернистый параллелизм).

Класс MISD, в котором множественный поток инструкций обрабатывает одиночный поток данных, пуст (пока бессмысленен).

Все современные топовые процессоры, как общего, так и специального назначения, попадают в класс MIMD. Они одновременно исполняют сразу несколько независимых потоков инструкций, аппаратно обеспечивая крупнозернистый параллелизм. Изъян классификации Флинна заключается в неразличимости современных направлений процессоростроения, что привело к возникновению множества новых классификаций.

Класс SIMD держится особо. Чистых его представителей совсем немного. Класс ориентирован на выполнение программ, для которых характерна обработка больших регулярных массивов чисел. Здесь одиночный поток инструкций обрабатывает множественный поток данных. Именно представители класса SIMD впервые достигнули производительности порядка GFLOPS.

Наиболее популярная идея класса SIMD – векторное процессирование. Векторный процессор поддерживает обработку не только скалярных, но и векторных операндов. Эффективное декодирование инструкций и удобные данные сказываются на производительности крайне позитивно. Поэтому векторную обработку внедряют и в процессоры классов SISD и MIMD. Например, SIMD расширение IA32 – технологии MMX и SSE. Этому вопросу будут посвящены отдельные лекции.

Слайд 5

Представители класса SIMD: векторные процессоры, матричные процессоры и процессоры с архитектурой VLIW.

У матричных процессоров нет аналогии с векторными. Матричный процессор – это массив процессоров с единым потоком команд. Большой исходный массив данных разделяют на части, подлежащие идентичной обработке. Каждый процессор массива обрабатывает соответствующую часть данных, выполняя единый поток инструкций.

Перспективным представителем класса SIMD является архитектура VLIW (очень длинное командное слово). Одна инструкция в такой системе команд представляет собой кортеж из нескольких инструкций, которые независимы по данным между собой. VLIW процессору не нужно проверять инструкции кортежа на выявление структурных зависимостей, зависимостей по данным или по управлению. Теперь эти функции возложены на компилятор. Процессор сразу может переходить к фазе исполнения. Такие блоки, как динамический планировщик, станции резервации и т.д., здесь упразднены. Высвободившиеся ресурсы (транзисторы) перераспределяются для повышения производительности системы. Таким образом, процессор и компилятор обеспечивают хороший уровень параллелизма команд. Этому вопросу будут посвящены отдельные лекции.

Представителей класса MIMD можно разделить на системы с общей или распределённой памятью.

Системы с общей памятью строятся посредством увеличения количества процессоров в машине. Такая мультипроцессорная система симметрична по процессорам. Т.е. задание, вытесненное на одном из предыдущих квантов времени, может быть возобновлено на любом из процессоров. С точки зрения ОС каждый процессор – обычный ресурс, который перераспределяется между заданиями (потоками). Безусловно, ОС стремится к минимизации частоты смены процессора для исполняемых программ. Это позволяет наиболее эффективно использовать ресурсы каждого из процессоров (кешей, буфера BTB, буфера TLB и т.д.). Такой режим симметричного мультипроцессирования (SMP) поддерживается во всех современных ОС.

Количественные изменения по закону Мура к настоящему времени привели к возможности размещения на одном чипе сразу нескольких процессоров (CMP). Такой многоядерный процессор с точки зрения ОС выглядит как несколько одноядерных и допускает симметричное мультипроцессирование. Иногда различные ядра имеют общий кеш второго или третьего уровней.

Мультикомпьютерные системы – системы с распределённой памятью. Такие системы представляют собой массив мощных серверов, объединённых в единый вычислительный ресурс при помощи высокопроизводительной коммуникационной сети (MPP). Кластерные системы – упрощенный вариант MPP. Например, несколько персональных компьютеров в сети Ethernet плюс некоторый механизм распределения вычислительной нагрузки – это уже простейший кластер!

Слайд 6

Итак, в настоящее время мощные центры вычислений базируются на крупнозернистом параллелизме и строятся либо как SMP, либо как MPP. Какой подход лучше?

Процессора в SMP системе представляют собой монолитный ресурс. Программа может быть легко переброшена с перегруженного процессора на разгруженный процессор. Многопоточной программе динамически выделяется то или иное количество процессоров в зависимости от загруженности системы в целом. Всплески вычислительной нагрузки легко перераспределяются планировщиком ОС. Более того, даже зависание одной или нескольких программ не приводит ОС в режим голодания! Но такие позитивные черты обусловлены общей памятью, к которой предъявляются очень высокие требования по производительности. Именно память ограничивает масштабируемость SMP.

В MPP частая передача вычислительной нагрузки между серверами противопоказана, так как приводит к медленным процедурам внешнего ввода/вывода. Фактически, это не единый

мощный ресурс. Это объединение ресурсов для решения той или иной задачи, требующее программирования специального вида. Требование специальной организации вычислений вынуждает либо существенно модифицировать имеющееся программное обеспечение, либо разрабатывать его вновь, что, безусловно, затрудняет переход от SMP к MPP. Ключевым преимуществом MPP является отличная масштабируемость, которая ограничивается только предельными параметрами коммуникационной сети.

Слайд 7

Архитектура SMP. Система состоит из нескольких однородных процессоров и массива общей памяти. Все процессоры имеют доступ к любой точке памяти с одинаковой скоростью. Аппаратно поддерживается когерентность кешей. Процессоры подключены к памяти либо с помощью общей шины, либо с помощью crossbar-коммутатора. Коммутатор, базируясь на принципе локальности программ, с целью распараллеливания банков памяти коммутирует процессора с банками памяти. Фактически, в каждый текущий момент времени каждый процессор имеет доступ только к некоторому фрагменту общей памяти.

Масштабируемость SMP. Наличие общей памяти сильно упрощает взаимодействие процессоров между собой, однако накладывает существенные ограничения на их число. В реальных системах установлено не более 32 ядер (сами ядра, как правило, многопоточные).

Операционная система для SMP. Вся система работает под управлением единой ОС, которая автоматически распределяет процессы по процессорам.

Модель программирования в SMP. Достаточно организовать классические потоки POSIX.

Слайд 8

Архитектура MPP. Система состоит из однородных вычислительных узлов, включающих один или несколько центральных процессоров, локальную память (прямой доступ к памяти других узлов невозможен), коммуникационный процессор. К системе могут быть добавлены специальные узлы ввода-вывода и управляющие узлы. Узлы связаны через некоторую коммуникационную среду.

Масштабируемость MPP. Общее число процессоров в реальных системах достигает нескольких тысяч!

Операционная система для MPP. Полноценная ОС работает только на управляющей машине. На каждом узле работает сильно урезанный вариант ОС, обеспечивающий только работу расположенной в нём ветви параллельного приложения.

Модель программирования в MPP. Программирование производится в рамках модели передачи сообщений (MPI, PVM, BSPlib).

Слайд 9

Кластерные системы. Набор рабочих станций общего назначения используется в качестве дешёвого варианта массивно-параллельного компьютера. Для связи узлов используется одна из стандартных сетевых технологий на базе шинной архитектуры или коммутатора. На каждом узле работает стандартная для рабочих станций ОС вместе со специальными средствами поддержки параллельного программирования и распределения нагрузки. Программирование ведётся в рамках модели передачи сообщений (MPI). Дешевизна подобных систем обуславливает большие накладные расходы на взаимодействие параллельных процессов между собой, что сильно сужает класс решаемых задач.

Слайд 10

Рассмотрим основные вехи в эволюции микропроцессорных архитектур. На ранних стадиях эволюции смена поколений ассоциировалась с революционными технологическими прорывами: каждое из первых четырёх поколений имело чётко выраженные отличительные технологические признаки. Последующие нечёткие деления на поколения основаны на изменениях в архитектуре вычислительных систем, а не на изменениях элементной базы.

Итак, вплоть до 1945 года была “механическая” эра эволюции вычислительной техники (нулевое поколение). Узловым моментом эволюции является концепция вычислительной машины с хранимой в памяти программой, сформулированная в 1945 году Джоном фон Нейманом. Относительно неё историю развития вычислительной техники можно представить в виде трёх этапов: донеймановкого периода, эры вычислительных машин с фон-неймановской архитектурой и постнеймановской эпохи (эпохи параллельных и распределённых вычислений). Напомним, что сущность фон-неймановской концепции состоит из четырёх принципов: двоичного кодирования, программного управления, однородности памяти (с позиций современного программирования не приветствуется) и адресности.

Первое поколение вычислительной техники (1937–1953) использовало электронные компоненты, которые могли переключаться в тысячу раз быстрее электромеханических аналогов. Программирование осуществлялось в машинном коде, а в пятидесятых годах вместо числовой записи появилась символьная нотация (ассемблер). В примитивном дизайне процессора был только аккумулятор.

Второе поколение (1954–1962) ознаменовалось переходом от электронных ламп к полупроводниковым диодам и транзисторам со временем переключения порядка 0.3 мс. Появление регистрового файла позволило организовать машины либо с регистровой архитектурой, либо со стековой архитектурой. Тогда стековая архитектура превалировала над регистровой, благодаря очень компактному коду. Безоперандные команды брали аргументы из стека и клали результат в стек. Для кодирования такой команды достаточно нескольких бит. В условиях ограниченности объёма доступной памяти это было очень веским преимуществом. В дальнейшем появление полупроводниковых запоминающих устройств привело к резкому увеличению объёма оперативной памяти и, соответственно, к безжизненности стековой архитектуры.

Третье поколение вычислительной техники (1963–1972) ознаменовалось переходом от дискретных полупроводниковых элементов к интегральным микросхемам, что обусловило скачок производительности. Зарождаются принципы конвейеризации и суперскалярности (принципы мелкозернистого параллелизма). Появляются многозадачные ОС. Архитектура системы команд (ISA) сильно развивается, предоставляя программистам (эра ассемблера) более развитые команды. Такую ISA впоследствии назовут CISC.

Четвёртое поколение (1972–1984) осуществило переход на СБИС (тысячи транзисторов на одном кристалле). Появление языков высокого уровня привело к резкому снижению востребованности CISC инструкций, так как компиляторы использовали только небольшое подмножество из них. Оказалось целесообразным перейти к архитектуре с сокращённым набором команд (RISC) и использовать высвобожденные ресурсы (транзисторы) для улучшения количественных характеристик CPU.

В девяностых годах доминируют суперскалярные процессоры. Принципы мелкозернистого параллелизма эксплуатируются максимально. Процессор содержит сложную логику динамического планирования в неупорядоченной модели обработки. Увеличение числа конвейеров и станций резервации приводит к экспоненциальному росту взаимосвязей между ними, что ограничивает дальнейшее распараллеливание на уровне команд. Назрела необходимость использования более высоких уровней параллелизма, началась эпоха параллельных и распределённых вычислений (постнеймановская эпоха).

Заметим, что история развития дизайна процессоров (включая эволюцию ISA) связана с агрегированием всё более высоких уровней параллелизма, что в настоящее время привело к мультипроцессорным и мультикомпьютерным системам.

Система команд первых трёх поколений вычислительной техники стремилась удовлетворить растущие потребности развивающихся технологий программирования, включая в состав ISA новые, более развитые команды. Однако угнаться за высоким темпом развития технологий программирования процессоростроители не могли. Образовался семантический разрыв. Так самая сложная в мире CISC система команд x86 не содержит никакой поддержки, например, объектно-ориентированного программирования. Отказ от языковой гонки (переход на RISC) и постоянное совершенствование языков программирования (Java и т.д.) закрепили семантический разрыв окончательно. Тем не менее, стоит отметить, что в последнее время всё активнее осуществляется аппаратная поддержка процессорами общего назначения машинного кода Java (иначе программная интерпретация кода Java имеет низкую производительность) либо в полном объёме, либо частично. Сейчас язык Java и код Java возродили интерес к стековой архитектуре (ROSC). Интегрированным виртуальным машинам будет посвящена отдельная лекция.

Все современные процессора базируются на RISC. Но некоторые могут возразить: например, машинный код Pentium 4 типа CISC, а машинный код сопроцессора Pentium 4 типа ROSC. Тем не менее, Pentium 4 имеет RISC ядро (начиная с i486). Поддержка CISC и ROSC инструкций выполнена с использованием техники Code Morphing для обеспечения преемственности x86 программ. Фактически, CISC и ROSC интерпретируются процессором при помощи декодирования в RISC инструкции (в микрокод). Технике Code Morphing будет посвящена отдельная лекция.

Слайд 11

Итак, назревшая необходимость использования более высоких уровней параллелизма определила перспективные пути развития GPP: VLIW, SMT и CMP. Переход от RISC к VLIW (как в процессорах общего, так и специального назначения) позволил преодолеть ограниченность ILP. Тенденции многопоточного процессирования (SMT) и многоядерного процессирования (CMP) привели к аппаратной поддержке крупнозернистого параллелизма. Этим вопросам будут посвящены отдельные лекции.
