23.11.2013

(переписывал у другого человека, ибо я пропустил эту лекцию)

Производительность ЭВМ

Единицы измерения:

1. Тактовая частота – герц – узкая область применимости метрики, т.к. важно знание архитектуры процессора + пропускная способность шины (Penium x64 – 528 гбит/c)
2. MIPS – Million instructions Per Second. Пакет для проверки теста Dhrynstone – смесь 100 команд. Кажется ещё были тесты VAX, IBM предложили RS/6000 – 1MIPS = 1,6 VAX
3. SPEC (Standard Performance Evaluation Corporation) – проверка при помощи, эталон – этот параметр – необходимая единица измерения любого процессора. В 92 предложили 2 пакета – работа с целыми числами и с плавающей запятой

Это не нормированная единица, всё сравнивается с некоторой эталонной машиной.

При измерениях берётся среднее геометрическое целочисленных и вещественных замеров.

1. MFLOPS – миллион операций с плавающей запятой в секунду.

Этот рейтинг широко используется для оценки производительности как средних, так и пиковых

ITANIUM2 – это epic – 1,3 Ггц, 1322 – spec.int, 2199 – spec.float

Pentium4 – 1,3 1620 1494

Alpha (тут нестандартный if) 1,13 877 1482

Rpeak – пиковая производительность, т.е. нету ни простоев, ни промахов, но многое зависит от архитектуры

Rmax – максимум на проверочном тесте на LINPAC

Реальная производительность – исполнение проверочных программ (они стандартные Benchmark (тут главное значение – эталон, по которому можно сравнивать))

Основные программы и типовые процессоры:

1. LINPAC – решение системы уравнений методом гаусса, точно известно число операций с плавающей запятой. Обычно размер примера – 100 или 10. Существуют версии для параллельных машин.
2. Более совершенный тест – AIX Technology – система тестовых нагрузок. Есть стандартные и заказные пакеты – моделирование средних, типовых задач.
3. Универсальной системы измерения средств вычислительной техники не существует.

Какие проблемы бывают:

1. Что лучше поставить векторную матрицу, или NVIDEA – т.е., как учитывать производительность графических процессоров.
2. Если придумали тест, то как доказать, что он лучше LINPACa

Топ 500 (упорядочен по LINPAC по Tmax)

1. Вышел 41 рейтинг. Лидер – Китай Tiahe-2 (Млечный путь 2) 3120000 ядер, максимум 33.867 Tflop/s. Вторые – Американцы. IPM. Япония.
2. В некоторых развёрнутых машинах входит 2-й параметр – Nhalf – длинна входного вектора, на котором достигается предельная производительность.

Top50 (самые мощные – отечественные машины, с каждой версией мощность удваивается)

История советской ЭВМ

1. MBC-100 – единственная, когда были задачи, но не хватало мощности
2. MBC-1000 – в вычислительном центре академии наук из элементов, купленных на западе и модифицированных. Потом эти штуки начали называть кластерами. Сейчас кластерами управляет «T-система»

Построить эту штуку – просто. Тяжело её загрузить

Оптимизация. Обычно у компилятора 4 стадии

1. Транслятор. (неприятности можно получить даже на элементарных оптимизациях)

Исключение общих подвыражений

Избыточность используемых регистров

Свёртка констант

Упрощение алгебраических выражений (x-x = 0 и прочее)

Команды сдвига, замена умножения на 2 сдвигом.

Каждый вид оптимизации имеет флаг – который можно заблокировать.

Пути конструкции получаются в процессе оптимизации.

X=A+B+C+D REG=B+C

Y=B+C+E => X=A+D+REG

 Y=E+REG

Если данные в младших разрядах критичны, то нарушение правильности арифметики, из-за смены порядка вычисления

Опция 0-1 на группах линейных участков

1. Оптимизация обращения к кешу

for (int i = 0; i < 100; i++)

for (int j = 1; j < 10; j++) -- тут лучше поменять их местами для кеша

С двумерным массивом работать лучше построчно, а не по столбцам

Interchange – сначала всё исполняем, потом всё вытесняет и не будет перезагружать.

Есть понятие блокировка цикла, тогда нам нужно сначала вычислить, только потом перестроить

Задача – определить фактор блокировки (обычно это делается компилятром автоматически, но мы должны уметь сами) зависит от того, сколько загружается данных за такт и какой у нас кеш

Это главная оптимизация, которую нужно было использовать, об этом нужно было говорить в управлении памятью

Ещё преобразование:

Unrolling – разворачивание цикла (например, если у нас сумматор с уровнем 2, то цикл по суммированию полезно будет развернуть в 2 раза)

Компилятор обычно имеет константу – во сколько раз можно расширить код. И в хорошей СП до этой константы можно добраться и изменить. Такие вещи обычно протоколируются.

Во всех оптимизациях очень важно проверить допустимость и корректность

1. Ещё преобразование:

Unrolling – разворачивание цикла (например, если у нас сумматор с уровнем 2, то цикл по суммированию полезно будет развернуть в 2 раза)

Компилятор обычно имеет константу – во сколько раз можно расширить код. И в хорошей СП до этой константы можно добраться и изменить. Такие вещи обычно протоколируются.

1. Во всех оптимизациях очень важно проверить допустимость и корректность
2. Распараллеливание