

# 1 Первая лекция (вторая)

Преподаватель: Андрей Горобец.

Сферы применения: можно рассчитывать свойства предметов до того как тот был создан.

На лекции показывали примеры.

g++ - formp функция что это?

рассмотрели unroll.

Классификация параллелизма Флинна.

Флопсы количество произведенных инструкций деленные на время.

TPP: пиковая производительность устройства. BW: пропускная способность памяти. AI: флоп на байт вычислительная интенсивность (как сильно

нагружен процессор к памяти) TPB: теоретически достижимая производительность.

Rmax: фактически достигаемая производительность.

CPI - количество тактов на операцию.

IPC - количество инструкций выполняемых за такт.

FLOP per cycle - количество операций с плавающей точкой.

ILP - конвейерный параллелизм и суперскалярность.

SIMD расширение - параллелизм ядра процессора.

FMA операции (AVX-512), векторные инструкции. + в может производить за один такт последовательно сложение плюс умножение. 16 операций за такт.

Ключевой момент - доступ к памяти.

Устройство конвейера. Разные типы конфликтов, которые прерывают конвейер.

Как определяется область видимости переменной в си.

OoOE (out-of-order) процессор растаскивает инструкции.

vLIW - very long instruction word. Компилятор сам распределяет инструкции.

SMT - simultaneous multithreading одновременная многопоточность.

Команды из разных потоков выполняются одновременно.

Иерархия памяти.

Поколения памяти - пропускная возможность увеличивается, но задержка операций снижается.

Кэш. Выравнивание из-за кэша.

## 2 Лекция 2

FUA - fused multiplier addition.

LLC - распределенный кэш.

### 2.1 Основы стандарта OpenMP (Старый).

макрос для секций

PU - processing unit. С общей памятью.

Характеристики памяти: Пропускная способность и время доступа (латентность).

Неоднородная общая память - NUMA (в разные места разное время доступа и пропускная способность).

Презентацию добавить.

Неоднородность работы.

Определение NUMA раздела с помощью масок.

## 3 Лекция 3

Имя лектора: Михаил Владимирович Якобовский.

Цель курса - дать понимание перед тем как писать программу имеет ли смысл ее распараллеливать.

Большая часть суперкомпьютеров > 100000

Четыре области применения многопроцессорных систем

1. Сокращение времени решения высокопроизводительных задач.

2. Сокращение времени обработки больших данных.
3. Решение задач в реальном времени.
4. Создание систем высокой надежности.

Оптически матричный умножитель - хорошая производительность, но низкая точность (специфический класс систем).

На нижнем уровне данные всегда передаются синхронным методом.

В итоге все равно будет передача точка-точка, даже в групповых передачах данных.

За микросекунду процессор выполняет примерно 1000 операций (порядок  $10^3$ ).

Ускорение это  $S_p = \frac{T_1}{T_p}$

Эффективность  $E_p = \frac{S_p}{p}$

Предел масштабируемости (не универсальное определение) - минимальное число процессоров, при котором достигается максимальное ускорение.

Каскадная схема сложения 8-ми чисел в 1000 раз медленнее на многопроцессорной системе (задача очень маленькая, из-за этого идет замедление, слишком много взаимодействий между процессорами).

$T_1^*$  - Самый быстрый последовательный алгоритм. Относительно него не должно быть сверхлинейного ускорения.

## 4 Лекция 4

$\tau_c$  — время операции

$\tau_s$  время передачи сообщения

### 4.1 Метод герметрического параллелизма

Задания должны быть связаны только локально, то он эффективен.

$$T_1 = (kn) = \tau_c kn$$

$$T_p(kn) = \tau_c \frac{kn}{p} + 4k\tau_S$$

$$S_p(kn) = p \frac{1}{1+4}$$

Перед тем как выполнить алгоритм нужно потратить еще логарифм от количества узлов передач сообщений.

Статическая балансировка и динамическая балансировка - различается тем что в статической заранее известно какие задачи будут выполняться на каком процессоре.

## 4.2 Метод коллективного решения

$$T_p = \frac{N}{p}(\tau_c + \tau_S)$$

$$T_p = \min(\frac{N}{p}(\tau_c + \tau_S); \frac{\tau_c}{\tau_s})$$

## 5 Лекция 5

Дали задание ( есть в чате во вложении)

Остальное есть в презентации.

Жуков Михаил Владимирович.

## 6 Лекция 6

Закон Амадаля

$$S(p) = \frac{1}{\alpha + \frac{1-\alpha}{p}}$$

$$E(p) = \frac{1}{1 + \alpha(p - 1)}$$

Критерии декомпозиции графов

1. По номерам узлов.
2. Равномерное распределение суммарного веса ребер и узлов.
3. Выделить граничные слои и посчитать ее позже.
4. Сделать так, чтобы у каждого домена не более двух соседей.
5. Если области не связны, то некоторые алгоритмы будут плохо сходиться.

## 7 Лекция 7

Локальное уточнение - берем уже готовое разбиение на домены и уточняем ее за счет граничных доменов. Именно на этом этапе возникает фрагментация.

Инкрементный алгоритм декомпозиции графа. Критерий "хорошего" домена. Простой алгоритм. Берем внешний контур домена, если оно связное, то убираем его и смотрим следующий домен. Чем больше слоев окажется связными тем более "хорошим" получится домен.

Редуцирование доменов - если домены оказались плохими, мы можем освободить часть точек доменов и заново перераспределить их.

Геометрическая декомпозиция может не работать, если неизвестно где находятся вершины (например для решения СЛАУ, где переменные представлены в виде вершин)

Разбиение сетки на микро домены зачастую весьма выгодно (можно менять количество доменов на ходу, легко визуализировать и анализировать и тд)

Использование адаптивной сетки подходит для разных задач.

## 8 Лекция

Проблема отказоустойчивости.

## 9 лекция

Визуализация сеточных данных.

ПК на несколько порядков слабее суперкомпьютера, но получаем данные мы через ПК, а точнее через экран.

Формат

1. Скалярные
2. векторные
1. Стационарные
2. Зависящие от времени
1. Решетки
2. Треугольные и тетраэдральные сетки

Есть два основных подхода:

1. Визуализировать прямо во время расчета
2. Сохранять на диск данные и затем его расшифровывать и визуализировать.