Параллельное программирование для высокопроизводительных систем

сентябрь – декабрь 2021 г.

Лектор доцент Попова Нина Николаевна

Лекция 1

9 сентября 2021 г.

План лекции

- О курсе
- Основы архитектур современных процессоров. Параллелизм инструкций.

План лекции

- О курсе

Основное содержание курса

- Модели и технологии параллельного программирования и их реализация.
- Базовые параллельные алгоритмы.
- Методы разработки параллельных программ.
- Методы и средства анализа эффективности параллельных программ.

Некоторые темы будут также изучаться в курсе «Суперкомпьютеры и параллельная обработка данных».

Модули курса

- Лекции + семинары

- Семинары:

  Выполнение практических заданий по курсу.

  Обсуждение и сдача заданий

  Для выполнения заданий потребуется удаленный доступ на вычислительные системы ВМК. Доступ будет оформляться преподавателем семинара.

Оценка по курсу

- **Оценка за семинар**

- оценки за задания (5-ти бальная система)

- своевременность сдачи заданий.

По каждому заданию будут указываться контрольные сроки. Невыполнение сроков приводит к автоматическому снижению оценки.

- качество выполненных заданий.

- **Оценка на экзамене**

Экзамен устный. В билете 2 теоретических вопроса и задача.

- Посещение лекций и семинаров

Вычислительные системы
http://hpc.cs.msu.ru/

- Доступные (домашние) вычислительные системы
- Многопроцессорные вычислительные системы факультета ВМК (http://hpc.cs.msu.ru/)

  - 2048-процессорная система IBM **Blue Gene/P (** http://hpc.cs.msu.ru/bgp)

  - 10-процессорный кластер на базе процессоров IBM Power8 **Polus (http://hpc.cs.msu.ru/polus)**

Учебные материалы

- Презентации лекций
- Основные учебники
- Интернет (статьи, материалы конференций, туториалы)

Тема

- Основы архитектур современных процессоров

Принципы архитектуры фон Неймана

Принцип двоичности.
Для представления данных и команд используется двоичная система счисления.

Принцип программного управления.
Программа состоит из набора команд, которые выполняются процессором друг за другом в определённой последовательности.

Принцип однородности памяти.
Как программы (команды), так и данные хранятся в одной и той же памяти. Над командами можно выполнять такие же действия, как и над данными.

Принципы архитектуры фон Неймана

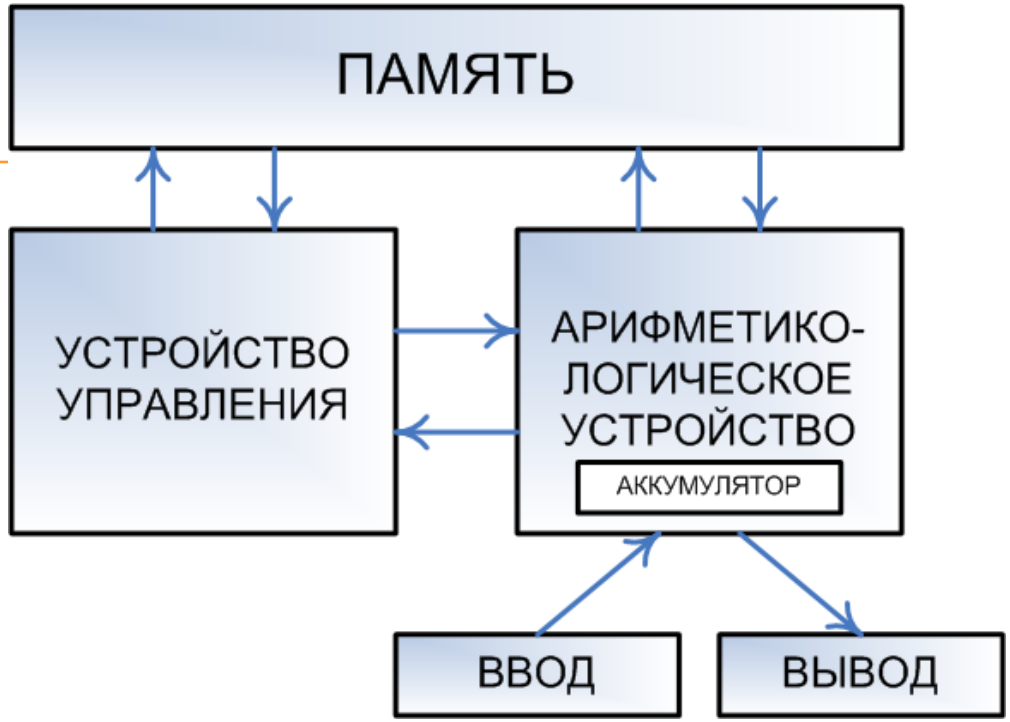
Принцип адресуемости памяти.
Структурно основная память состоит из пронумерованных ячеек; процессору в произвольный момент времени доступна любая ячейка.

Принцип последовательного программного управления.
Все команды располагаются в памяти и выполняются последовательно, одна после завершения другой.

Принцип условного перехода.
Команды из программы не всегда выполняются одна за другой. Возможно присутствие в программе команд условного перехода, которые изменяют последовательность выполнения команд в зависимости от значений данных.

Последовательный компьютер

Джон фон Нейман (1903 – 1957)
американский математик венгерского происхождения



Узкое место архитектуры Дж. фон Неймана – совместное использование шины для памяти программ и памяти данных (это ограничивает пропускную способность между процессором и памятью)

Тенденции развития современных процессоров

•Повышение тактовой частоты.

- Пример: частота CPU семейства x86 выросла с 4.77 MHz до 3 GHz

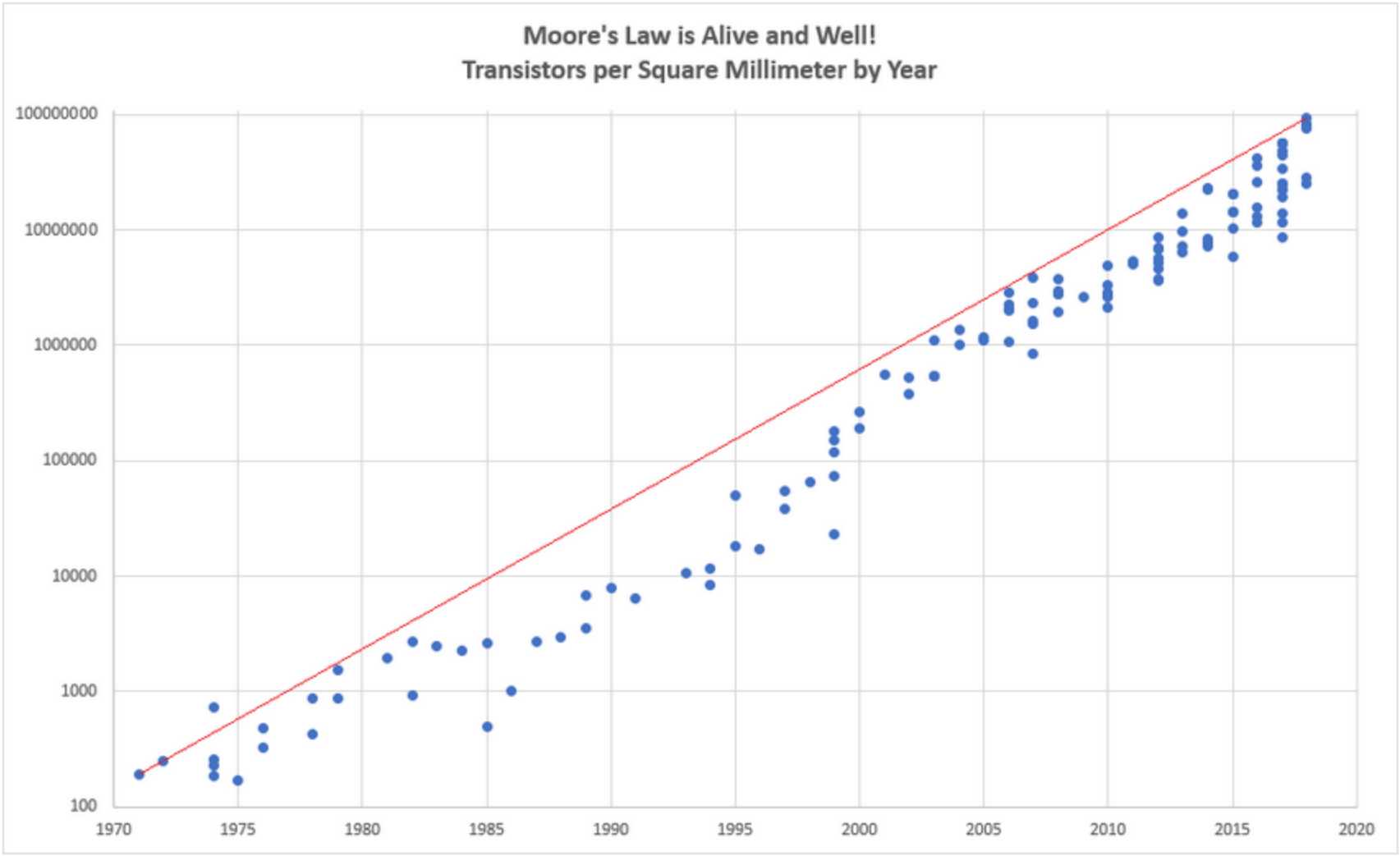
•Увеличение плотности упаковки транзисторов на плате

- Совершенствование архитектуры процессора (параллельное и конвейерное выполнение команд)

• Удвоение быстродействия каждые 1.5 – 2 года .

Закон Мура (1965 г.) –удвоение числа транзисторов

Гордон Мур, (1929 г.)
Выпускник университета в Беркли, США. Один из основателей компании Интел



Transistors per Square Millimeter by Year, 1971–2018. Logarithmic scale. Data from Wikipedia.

Закон масштабирования Деннарда

Закон масштабирования Деннарда (*Dennard scaling* также известен как закон масштабирования MOSFET) — эмпирический закон масштабирования открытый в 1974 году. Один из авторов – Роберт Деннард.

«Уменьшая размеры и повышая тактовую частоту процессора, можно повышать его производительность».

Повышение производительности процессоров

Совершенствование процесса выполнения инструкций

- Совершенствование архитектуры набора команд:
  RISC, CISC, MISC, VLIW

- Параллелизм уровня инструкций:
  конвейерная архитектура, суперскалярная обработка

- Параллелизм уровня потоков:
  многопроцессорные системы, одновременная многопоточность, многоядерные процессоры

- Параллелизм данных: векторная обработка данных (векторные процессоры/инструкции)

Параллелизм уровня инструкций (Instruction Level Parallelism –ILP)

ILP эра: 1982-2005

Конвейерные ФУ

Схема из логических элементов на процессоре, которая выполняет определенные операции называется ***функциональным устройством***

Большинство ФУ, выполняющих целочисленную арифметику и операции с плавающей точкой являются ***конвейерными***
Каждая стадия конвейерного ФУ работает одновременно (со своими операндами)
Цель: после начальной инициализации конвейера генерировать результат за один такт работы процессора

Этапы выполнения инструкции (пусть каждый этап длится 1 такт):

- **IF** – выбор инструкции
- **ID** – декодирование инструкции
- **IE** – выполнение инструкции
- **WB** – запись результата

```
i1: movl  %ebx, %eax
i2: cmpl  $0x10, %ecx
i3: movl  %edx, %ebx
i4: xorl  %r8d, %r8d
```

16 тактов

Процессор (ядро)

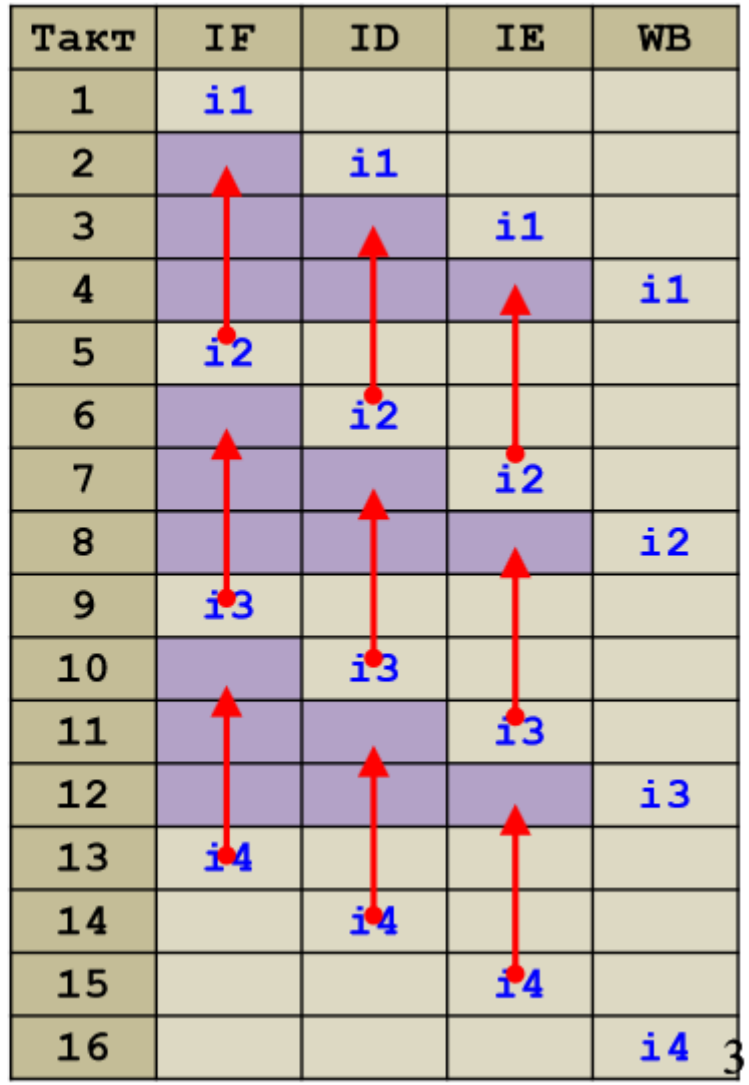
Такт	IF	ID	IE	WB
1	i1			
2		i1		
3			i1	
4				i1
5	i2			
6		i2		
7			i2	
8				i2
9	i3			
10		i3		
11			i3	
12				i3
13	i4			
14		i4		
15			i4	
16				i4

Этапы обработки инструкции можно сделать независимыми и сократить ожидания

```
i1: movl   %ebx, %eax
i2: cmpl   $0x10, %ecx
i3: movl   %edx, %ebx
i4: xorl   %r8d, %r8d
```

16 тактов

Процессор (ядро)



Достигнутое ускорение (Speedup)

$$S_4 = \frac{16}{7} = 2.3$$

```
i1: movl  %ebx, %eax
i2: cmpl  $0x10, %ecx
i3: movl  %edx, %ebx
i4: xorl  %r8d, %r8d
```

7 тактов

Процессор (ядро)

Такт	IF	ID	IE	WB
1	i1			
2	i2	i1		
3	i3	i2	i1	
4	i4	i3	i2	i1
5		i4	i3	i2
6			i4	i3
7				i4
8				
9				
10				
11				
12				
13				
14				
15				
16				

Вычислительный конвейер (Instruction pipeline)

1.IF (Instruction Fetch) –загрузка инструкции из памяти (кэша инструкций, 1 такт)
2.ID (Instruction Decode)–декодирование инструкции
3.EX (Execution)–выполнение (Reg-RegOp. –1 такт, Mem. ref. –2 такта;Div, Mul, FP Ops.–требуют > 1 такта)
4.MEM (Memory access)–доступ к памяти (чтение/запись)
5.WB (Register Write Back) –запись в регистры

Анализ ускорения (Speedup)

- n – количество этапов (stages) конвейера
- L – количество инструкций в программе
- t – время выполнения одного этапа

$$S_n = \frac{T_{serial}}{T_{pipeline}} = n$$

$$T_{serial} = Lnt$$

$$T_{pipeline} = (n-1)t + Lt$$

$$S = \frac{Lnt}{(n-1+L)t} = \frac{Ln}{n-1+L}$$

$$\lim_{L\to\infty} S_n = \frac{Ln}{n-1+L} = n$$

Такт	IF	ID	IE	WB
1	i1			
2	i2	i1		
3	i3	i2	i1	
4	i4	i3	i2	i1
5		i4	i3	i2
6			i4	i3
7				i4
8				
9				
10				
11				
12				
13				
14				
15				
16				

Вычислительный конвейер (Instruction pipeline)

ıntel 80486 –5-stage (scalar, CISC)

Intel Pentium –5-stage (2 integer execution units)

Intel Pentium Pro –14-stage pipeline

Intel Pentium 4 (Cedar Mill) –31-stage pipeline

Intel Core i7 4771 (Haswell) –14-stage pipeline

- **Конфликт конвейера (Hazard)** – это ситуация, когда выполнение следующей инструкции не может быть начато/продолжено
- Виды конфликтов:
  - Структурные конфликты (Structural Hazards)
  - Конфликты данных (Data Hazards)
  - Конфликты управления (Control Hazards)

Конфликты являются причиной замедления работы конвейера

Конфликты конвейера

•*Структурные конфликты* возникают в том случае, когда аппаратные средства процессора не могут поддерживать все возможные комбинации команд в режиме одновременного выполнения с совмещением.

•*Конфликты по данным* возникают в случаях, когда выполнение одной команды зависит от результата выполнения предыдущей команды

•*Конфликты управления* возникают при конвейеризации команд переходов и других команд, изменяющих значение счетчика команд

Устранение конфликтов данных

1. Техника перенаправления (forwarding/bypassing/ short-circuiting)

2. Внеочередное исполнение команд (Out-of-order execution)

Устранение конфликтов данных

•Перенаправление – в конвейере реализуется возможность передачи значений от инструкции к инструкции минуя регистровый файл

• Выход некоторых функциональных устройств аппаратно связывается со входом других

Внеочередное исполнение команд (ООЕ)

Более поздние (по коду) операции могут исполняться перед более ранними, если не зависят от порождаемых ими результатов.

Должно быть гарантировано условие, чтобы результаты «внеочередного» выполнения программы совпадали с результатами «правильного» последовательного выполнения.

Механизм внеочередного исполнения позволяет в значительной степени сгладить эффект от ожидания считывания данных из кэшей верхних уровней и из оперативной памяти, что может занимать десятки и сотни тактов.

Внеочередное исполнение команд (OOE)

1. Инструкция выбирается из памяти (одна или несколько)
2. Инструкция направляется (dispatch) в очередь инструкций (instruction queue, instruction buffer, reservation station)
3. Находясь в очереди инструкция ожидает пока её операнды станут доступными. После чего инструкция может покинуть очередь раньше более старых команд
4. Инструкция направляется на подходящее исполняющее устройство
5. Результаты выполнения инструкции помещаются в очередь
6. Инструкция записывает данные в регистровый файл, только после того как более старые инструкции сохранили свои результаты – retire stage

Предсказание переходов

• Модуль предсказания условных переходов (Branch Prediction Unit, BPU) – модуль процессора, определяющий будет ли выполнен переход и куда

• Предсказывает условные переходы, вызовы/возвраты из функций

• Вероятность предсказания переходов в современных процессорах может быть очень высокой

• После предсказания процессор начинает спекулятивно выполнять инструкции (speculative execution)
• Альтернативный подход (без BPU) – спекулятивно выполнять обе ветви ветвления, пока не будет вычислено управляющее выражение (условие)

Суперскалярные процессоры

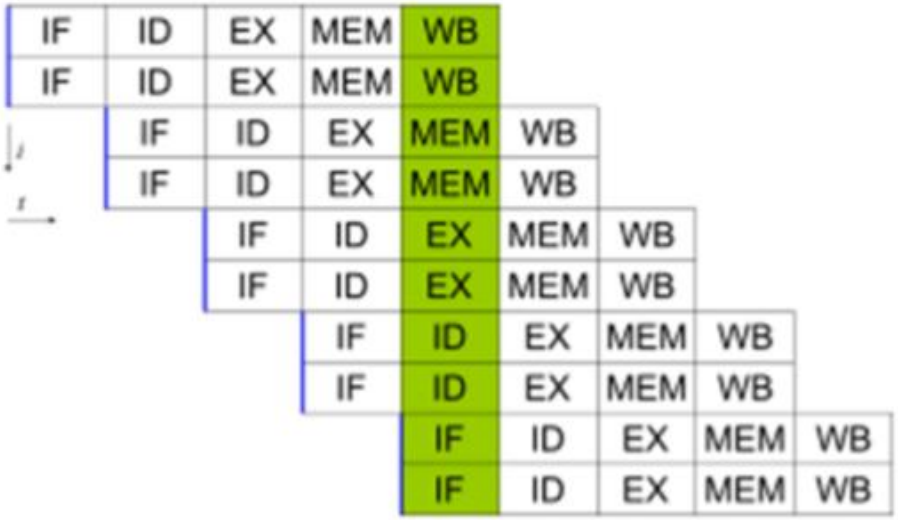
•Некоторые блоки конвейера (АЛУ, сдвиговое устройство, умножитель, …) присутствуют в нескольких экземплярах

•За один такт процессор выполняет несколько инструкций

•Процессор динамически проверяет входные инструкции на зависимость по данным –динамический планировщик (dispatcher, динамическое распараллеливание)

Суперскалярные процессоры

```
a = b + c
d = e + f
```




```
a = b + c
b = e + f
```

Суперскалярные процессоры

CDC 6600 (1965)
Intel i960CA (1988)
AMD 29000-series 29050 (1990)
Intel Pentium –первый суперскалярный x86 процессор, 2 datapaths (pipelines)