Теория и практика многопоточного программирования

Семинар 1

Неганов Алексей

Московский физико-технический институт (национальный исследовательский университет)
Кафедра теоретической и прикладной информатики

Москва 2020

Ожидаемые знания

- Основы теории ОС
- Основы архитектуры компьютера
- Языки C и C++

Горизонты

- Основные концепции параллельного программирования с разделяемой памятью
- Базовые сведения об архитектуре многопроцессорных компьютеров
- Декомпозиция задач
- Синхронизация
- Разбор распространённых проблем
- Разбор работы с потоками и процессами ОС
- Ввод-вывод в многопоточной системе
- Реализация важнейших примитивов и алгоритмов на языках C и C++
- Исследование поведения паралельных программ под нагрузкой

Дополнительно:

- Демонстрация некоторых решений на языке Go
- OpenMP
- Проблемы параллельного исполнения транзакций БД
- ...

Многозадачность

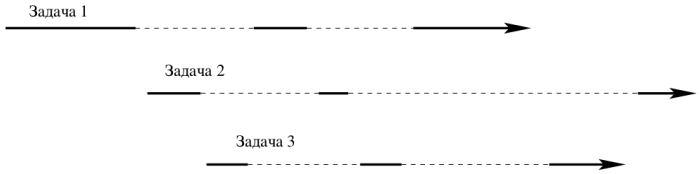

Рисунок 1 – Одновременное выполнение задач на одном процессоре

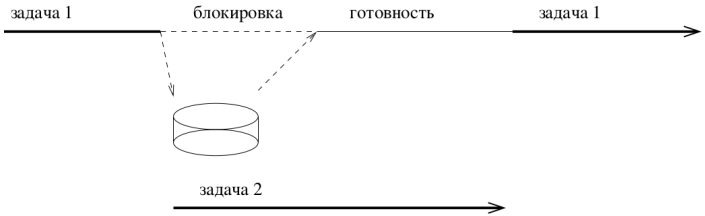

Рисунок 2 – Пакетный режим

Классификация архитектур ЭВМ

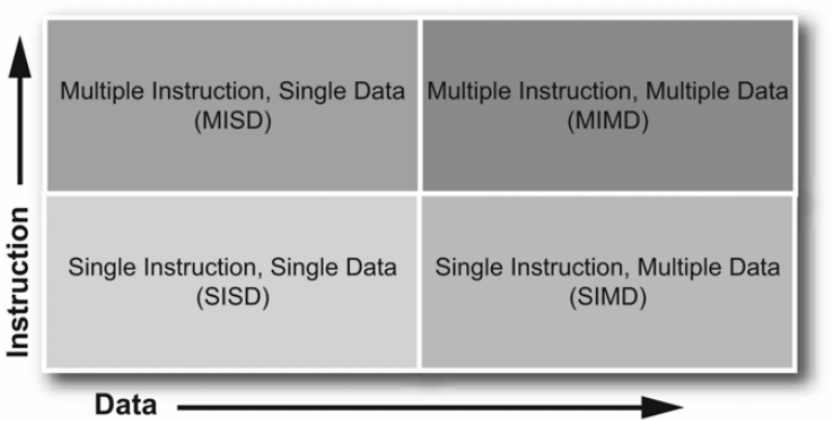

Рисунок 3 – Таксономия Флинна

Атомарность операций

Чему может оказаться равен x?

```
int x = 17;

process foo {
    x++;
}

process bar {
    x++;
}

process baz {
    x++;
}
```

Атомарность операций

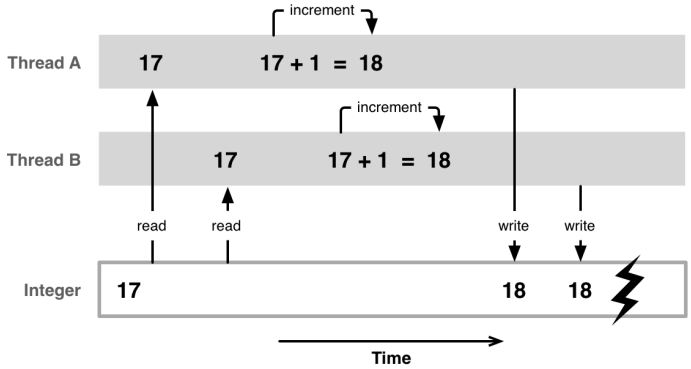

Рисунок 4 – Иллюстрация состояния гонки для оператора инкремента

Связность памяти

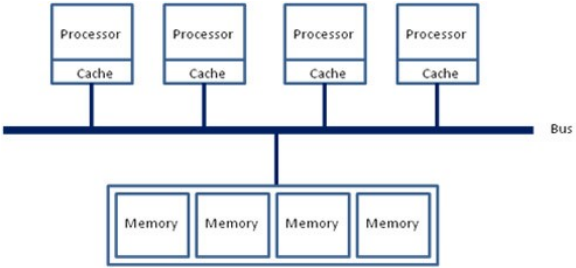

Рисунок 5 – Соединение типа SMP

Связность памяти

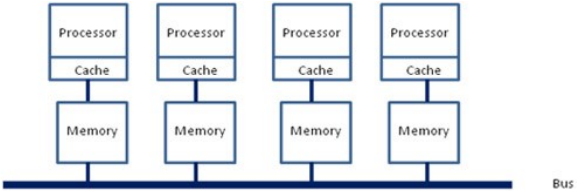

Рисунок 6 – Соединение типа NUMA

Связность памяти

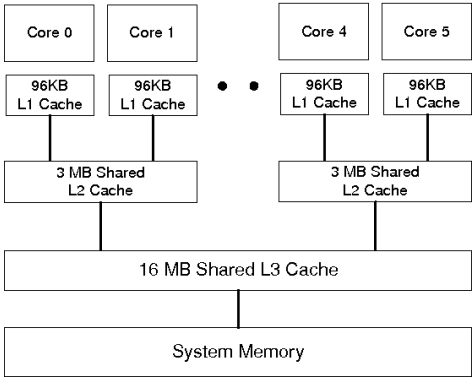

Рисунок 7 – Иерархия памяти для многоядерной SMP-системы

Коварный кэш

Может ли выполниться `assert()`?

```
int a = 0;
int b = 0;

void foo(void) {
      a = 1;
      b = 1;
}

void bar(void) {
      while(b == 0);
      assert(a == 1);
}
```

Коварный кэш

Может ли выполниться `assert()`?

```
int a = 0;
int b = 0;

void foo(void) {
      a = 1;
      smp_mb();
      b = 1;
}

void bar(void) {
      while(b == 0);
      assert(a == 1);
}
```

Коварный кэш

Чем плох этот код?

```
struct foo {
    int x;
    int y;
};

static struct foo f;

int sum(void)
{
    int s = 0;
    for (int i = 0; i < 1000000; ++i)
        s += f.x;
    return s;
}

void inc(void)
{
    for (int i = 0; i < 1000000; ++i)
        ++f.y;
}
```