Multiprocessor

COMP4611

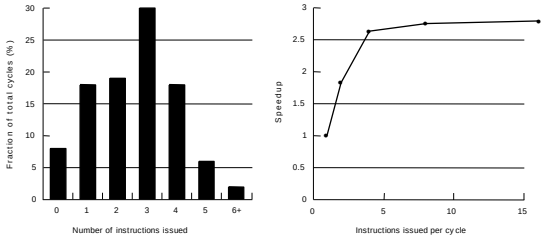
Tutorial 11

Nov. 26 – 30, 2012

Outline

- Why Multiprocessor?
- Performance Potential
- New Problems
  - Cache Coherence Problem & Solution
- Challenges of Parallel Processing

The Bottleneck of Single Processor

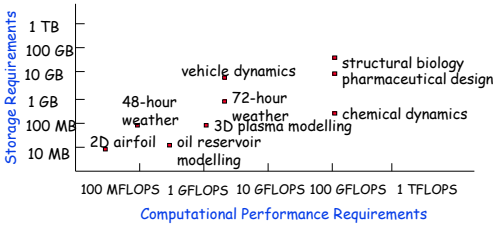


Even various ways of increasing a single processor performance have been introduced, the performance of a single processor still has its bottleneck.

The figures above demonstrate the limitation of ILP.

We Need High Performance

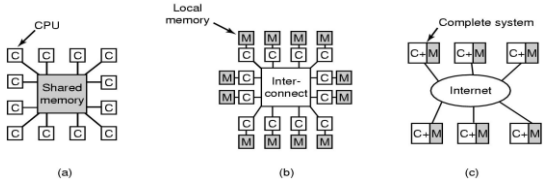
- **Less time** for a time-consuming operation
- **More operations** in a period of time



Multiprocessing

Concurrent execution of tasks (programs) using multiple computing, memory and interconnection resources

- Provide alternative to faster processors
- Use multiple processors to solve a problem



Outline

- Why Multiprocessor?
- Performance Potential
- New Problems
  - Cache Coherence Problem & Solution
- Challenges in Using Multiprocessors

Performance Potential Using Multiprocessor

- Amdahl's Law
  - Using multiple processors to solve the same problems as the single processor
  - Pessimistic, limited speedup factor
- Gustafson's View
  - Using multiple processors to solve larger or more complex problems
  - Parallel portion increases as the problem size increases
  - Speedup factor can be increased by deploying more processors

7

Amdahl's Law

- A parallel program has sequential parts and parallel parts, the proportion for both of them are β and $(1-\beta)$
  - The total execution time for a single processer is
    - $T_1 = \beta T_1 + (1-\beta)T_1$
  - The total execution time for *p* processors would be
    - $T_p = \beta T_1 + (1-\beta)T_1 / p$
- Speedup(p) = $1 / (\beta + (1-\beta)/p) \leq 1 / \beta$

8

Amdahl's Law

- *S* is the serial part
- *P* is the part that can be parallelized in 6 ways

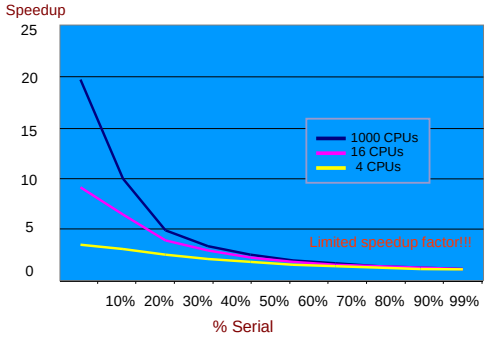
```
Serial:       SSPPPPPP
6 processors: SSP
                P
                P
                P
                P
                P
```

- Speedup = 8/3 = 2.67

- Speedup(p) = $1 / (\beta + (1- \beta)/p)$
- As $p \rightarrow \infty$, Speedup(p) $\rightarrow 1/ \beta$

9

Amdahl's Law



10

Gustafson's View

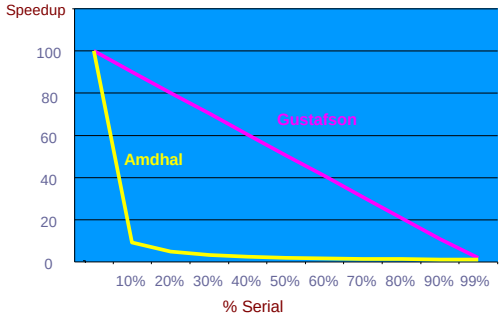
Parallel portion increases as the problem size increases

```
Old Serial:   SSPPPPPP
Hypothetical Serial: SSPPPPPP PPPPPP PPPPPP PPPPPP
PPPPPP PPPPPP
6 processors: SSPPPPPP
                PPPPPP
                PPPPPP
                PPPPPP
                PPPPPP
                PPPPPP
```

- Speedup(6) = (8+5*6)/8 = 4.75
- Speedup(p) = p(1- β) + β; **Speedup(p)** $_{p \rightarrow \infty} \rightarrow \infty$

11

Amdahl vs. Gustafson



12

Example 1. To achieve a speedup of 80 with 100 processors, the fraction of the original computation can be sequential

- Amdahl's law
  - 1 / (β + (1- β)/100) = 80
  - β = **0.25%**
- Gustafson's view
  - 100(1- β) + β = 80
  - β =20%

To achieve high speedup, only a very small fraction of a program can be sequential

In practice, programs usually have much more sequential fraction

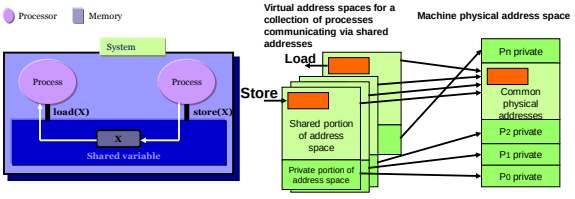
13

Outline

- Why Multiprocessor?
- Performance Potential
- New Problems
  - Cache Coherence Problem & Solution
- Challenges in Using Multiprocessors

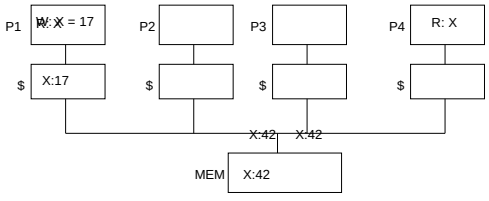
14

Programming with Shared Memory



15

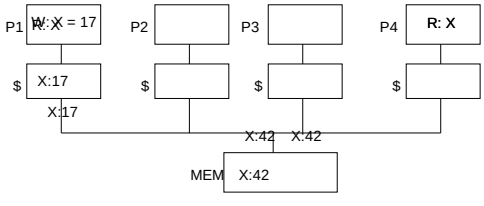
Cache Coherence Problem



Processor P4 does not see the value written by processor P1

16

Write Through Does Not Help



Processor P4 sees 42 in cache and does not get the value (17) from memory.

17

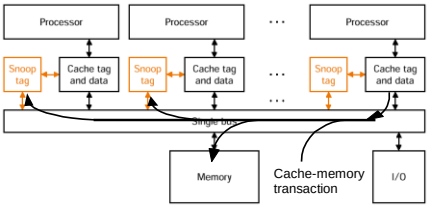
Some Solutions to Cache Coherence Problem

- Shared Cache
  - Cache placement identical to single cache
  - Only one copy of any cached block which results in limited bandwidth
- Snoopy Cache-Coherence Protocols for Distributed Cache

18

Snooping Cache Coherency

Bus snoop

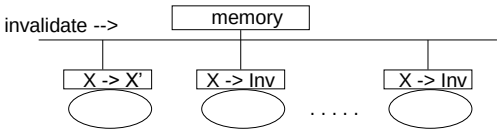


- Cache Controller "snoops" all transactions on the shared bus and takes action to ensure coherence (invalidate, update, or supply value)
- A transaction is a relevant transaction if it involves a cache block currently contained in this cache

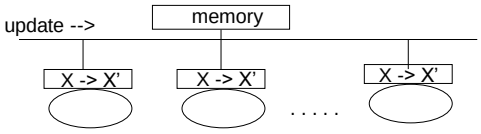
19

Hardware Cache Coherence Demonstration

- **write-invalidate**



- **write-update** (also called distributed write)



Outline

- Why Multiprocessor?
- Performance Potential
- New Problems
  - Cache Coherence Problem & Solution
- Challenges in Using Multiprocessors

21

Challenges in Using Multiprocessors

- **Limited parallelism available in programs**
  - Example 1
- **Relatively high cost of communications**
  - Example 2

22

Example 2

- We have a 32-processor multiprocessor
  - Clock rate is 3.3 GHz
  - Base CPI is 0.5 (no remote request)
  - Shared memory - nonuniform memory access
  - 200 ns to handle a remote memory reference
  - A processor is stalled when it invokes a remote memory access
- How much slower if 0.2% of the instructions involve remote requests?

23

- **Effective CPI**
  = Base CPI + Remote req. rate x Remote req. cost
  = 0.5 + 0.2% x Remote req. cost
- **Remote req. cost**
  = Remote access cost / cycle time
  = 200 ns / (1/3.3) ns = 660 cycles
- **CPI** = 0.5 + 0.2% x 660 = 1.8
- **CPI / Base CPI** = 1.8 / 0.5 = **3.6**

0.2% remote requests make the program 3.6 times slower

24

Questions?