Multiprocessors

HPC

Prof. Robert van Engelen
Overview

- The PMS model
- Shared memory multiprocessors
  - Basic shared memory systems
  - SMP, Multicore, and COMA
- Distributed memory multicomputers
  - MPP systems
  - Network topologies for message-passing multicomputers
  - Distributed shared memory
- Pipeline and vector processors
- Comparison
- Taxonomies
PMS Architecture Model

A simple PMS model

- Processor (P)
  - A device that performs operations on data
- Memory (M)
  - A device that stores data
- Switch (S)
  - A device that facilitates transfer of data between devices
- Arcs denote connectivity

Each component has different performance characteristics

An example computer system with CPU and peripherals
Shared Memory Multiprocessor

- Processors access *shared memory* via a common switch, e.g. a *bus*
  - Problem: a single bus results in a *bottleneck*
- Shared memory has a *single address space*
- Architecture sometimes referred to as a “*dance hall*”
Shared Memory: the Bus Contention Problem

- Each processor competes for access to shared memory
  - Fetching instructions
  - Loading and storing data

- Performance of a single bus $S$: *bus contention*
  - Access to memory is restricted to one processor at a time
  - This limits the speedup and scalability with respect to the number of processors
  - Assume that each instruction requires $0 < m < 1$ memory operations (the average fraction of loads or stores per instruction), $F$ instructions are performed per unit of time, and a maximum of $W$ words can be moved over the bus per unit of time, then
    \[ S_P \leq \frac{W}{mF} \]
    regardless of the number of processors $P$
  - In other words, the parallel efficiency is limited unless
    \[ P \leq \frac{W}{mF} \]
Shared Memory: Work-Memory Ratio

Work-memory ratio (FP:M ratio): ratio of the number of floating point operations to the number of distinct memory locations referenced in the innermost loop:

- Same location is counted just once in innermost loop
- Assumes effective use of registers (and cache) in innermost loop
- Assumes no reuse across outer loops (registers/cache use saturated in inner loop)

Note that $FP:M = m^{-1} - 1$ so efficient utilization of shared memory multiprocessors requires $P \leq (FP:M + 1) \times W / F$

```
for (i=0; i<1000; i++)
    x = x + i;
```

2 distinct memory locations and one float add:
$FP:M = 500$

```
for (i=0; i<N; i++)
    x = x + a[i]*b[i];
```

2N+1 distinct memory locations and 2N FP operations:
$FP:M = 1$
when N is large
Shared Memory Multiprocessor with Local Cache

- Add local cache to improve performance when \( W / F \) is small
  - With today’s systems we have \( W / F << 1 \)
- Problem: how to ensure cache coherence?
Shared Memory: Cache Coherence

- A cache coherence protocol ensures that processors obtain newly altered data when shared data is modified by another processor.
- Because caches operate on cache lines, more data than the shared object alone can be effected, which may lead to false sharing.

Thread 1 modifies shared data

Thread 0 reads modified shared data

Cache coherence protocol ensures that thread 0 obtains the newly altered data
**COMA**

- *Cache-only memory architecture* (COMA)
- Large cache per processor to replace shared memory
- A data item is either in one cache (non-shared) or in multiple caches (shared)
- Switch includes an engine that provides a single global address space and ensures *cache coherence*
Distributed Memory Multicomputer

- Massively parallel processor (MPP) systems with \( P > 1000 \)
- Communication via message passing
- Nonuniform memory access (NUMA)
- Network topologies
  - Mesh
  - Hypercube
  - Cross-bar switch
Distributed Shared Memory

- Distributed shared memory (DSM) systems use physically distributed memory modules and a global address space that gives the illusion of shared virtual memory that is usually NUMA.
- Hardware is used to automatically translate a memory address into a local address or a remote memory address (via message passing).
- Software approaches add a programming layer to simplify access to shared objects (hiding the message passing communications).
**Computation-Communication Ratio**

The *computation-communication ratio*:

\[ \frac{t_{\text{comp}}}{t_{\text{comm}}} \]

Usually assessed analytically and verified empirically

High communication overhead decreases speedup, so ratio should be as high as possible

For example: data size \( n \), number of processors \( P \), and ratio \( t_{\text{comp}} / t_{\text{comm}} = 1000n / 10n^2 \)

\[ S_P = \frac{t_s}{t_P} = \frac{1000n}{(1000n / P + 10n^2)} \]
Mesh Topology

- Network of $P$ nodes has mesh size $\sqrt{P} \times \sqrt{P}$
- Diameter $2 \times (\sqrt{P} - 1)$
- Torus network wraps the ends
  - Diameter $\sqrt{P} - 1$
Hypercube Topology

- $d$-dimensional hypercube has $P = 2^d$ nodes
- Diameter is $d = \log_2 P$
- Node addressing is simple
  - Node number of nearest neighbor node differs in one bit
  - Routing algorithm flips bits to determine possible paths, e.g.
  - From node 001 to 111 has two shortest paths
    - $001 \rightarrow 011 \rightarrow 111$
    - $001 \rightarrow 101 \rightarrow 111$
Cross-bar Switches

- Processors and memories are connected by a set of switches.
- Enables simultaneous (contention free) communication between processor $i$ and $\sigma(i)$, where $\sigma$ is an arbitrary permutation of $1 \ldots P$.

Cross-bar switch

$\sigma(1)=2$
$\sigma(2)=1$
$\sigma(3)=3$
Multistage Interconnect Network

- Each switch has an upper output (0) and a lower output (1)
- A message travels through switch based on destination address
  - Each bit in destination address is used to control a switch from start to destination
  - For example, from 001 to 100
    - First switch selects lower output (1)
    - Second switch selects upper output (0)
    - Third switch selects upper output (0)
- Contention can occur when two messages are routed through the same switch
Pipeline and Vector Processors

- **Vector processors** run operations on multiple data elements simultaneously.
- Vector processor has a maximum vector length, e.g. 512.
- *Strip mining* the loop results in an outer loop with stride 512 to enable vectorization of longer vector operations.
- *Pipelined vector architectures* dispatch multiple vector operations per clock cycle.
- *Vector chaining* allows the result of a previous vector operation to be directly fed into the next operation in the pipeline.

```
DO i = 0,9999
   z(i) = x(i) + y(i)
ENDDO

DO j = 0,18,512
   DO i = 0,511
      z(j+i) = x(j+i) + y(j+i)
   ENDDO
ENDDO
DO i = 0,271
   z(9728+i) = x(9728+i) + y(9728+i)
ENDDO

DO j = 0,18,512
   z(j:j+511) = x(j:j+511) + y(j:j+511)
ENDDO

z(9728:9999) = x(9728:9999) + y(9728:9999)
```
Comparison: Bandwidth, Latency and Capacity
Further Reading

- [PP2] pages 13-26
- [SPC] pages 71-95
- [HPC] pages 25-28