Chapter 7

Multicores, Multiprocessors, and Clusters

Introduction

- Goal: connecting multiple computers to get higher performance
  - Multiprocessors
  - Scalability, availability, power efficiency
- Job-level (process-level) parallelism
  - High throughput for independent jobs
- Parallel processing program
  - Single program run on multiple processors
- Multicore microprocessors
  - Chips with multiple processors (cores)
Hardware and Software

- **Hardware**
  - Serial: e.g., Pentium 4
  - Parallel: e.g., quad-core Xeon e5345

- **Software**
  - Sequential: e.g., matrix multiplication
  - Concurrent: e.g., operating system, multithreaded matrix multiplication

Sequential/concurrent software can run on serial/parallel hardware
- Challenge: making effective use of parallel hardware

Parallel Programming

- Parallel software is the problem
- Need to get significant performance improvement
  - Otherwise, just use a faster uniprocessor, since it’s easier!

- Difficulties
  - Partitioning
  - Coordination
  - Communications overhead
Amdahl’s Law

- Sequential part can limit speedup
- Example: 100 processors, 90× speedup?
  \[ T_{\text{new}} = \frac{T_{\text{parallelizable}}}{100} + T_{\text{sequential}} \]
  \[ \text{Speedup} = \frac{1}{1 - F_{\text{parallelizable}} + \frac{F_{\text{parallelizable}}}{100}} = 90 \]
  - Solving: \( F_{\text{parallelizable}} = 0.999 \)
- Need sequential part to be 0.1% of original time

Scaling Example

- Workload: sum of 10 scalars, and 10 × 10 matrix sum
  - Speed up from 10 to 100 processors
- Single processor: Time = (10 + 100) \( \times t_{\text{add}} \)
- 10 processors
  - Time = 10 \( \times t_{\text{add}} \) + 100/10 \( \times t_{\text{add}} \) = 20 \( \times t_{\text{add}} \)
  - Speedup = 110/20 = 5.5 (55% of potential)
- 100 processors
  - Time = 10 \( \times t_{\text{add}} \) + 100/100 \( \times t_{\text{add}} \) = 11 \( \times t_{\text{add}} \)
  - Speedup = 110/11 = 10 (10% of potential)
- Assumes load can be balanced across processors
Scaling Example (cont)

- What if matrix size is $100 \times 100$?
- Single processor: $Time = (10 + 10000) \times t_{\text{add}}$
- 10 processors
  - $Time = 10 \times t_{\text{add}} + 10000/10 \times t_{\text{add}} = 1010 \times t_{\text{add}}$
  - Speedup $= 10010/1010 = 9.9$ (99% of potential)
- 100 processors
  - $Time = 10 \times t_{\text{add}} + 10000/100 \times t_{\text{add}} = 110 \times t_{\text{add}}$
  - Speedup $= 10010/110 = 91$ (91% of potential)
- Assuming load balanced

Strong vs Weak Scaling

- Strong scaling: problem size fixed
  - As in example
- Weak scaling: problem size proportional to number of processors
  - 10 processors, $10 \times 10$ matrix
    - $Time = 20 \times t_{\text{add}}$
  - 100 processors, $32 \times 32$ matrix
    - $Time = 10 \times t_{\text{add}} + 1000/100 \times t_{\text{add}} = 20 \times t_{\text{add}}$
    - Constant performance in this example
Shared Memory

- SMP: shared memory multiprocessor
  - Hardware provides single physical address space for all processors
  - Synchronize shared variables using locks
  - Cache coherence issue
  - Memory access time
    - UMA (uniform) vs. NUMA (nonuniform)

Multi-cores processors:
Message Passing

- Each processor has private physical address space
- Hardware sends/receives messages between processors

Loosely Coupled Clusters

- Network of independent computers
  - Each has private memory and OS
  - Connected using I/O system
    - E.g., Ethernet/switch, Internet
- Suitable for applications with independent tasks
  - Web servers, databases, simulations, …
- High availability, scalable, affordable
- Problems
  - Administration cost (prefer virtual machines)
  - Low interconnect bandwidth
    - c.f. processor/memory bandwidth on an SMP
Grid Computing

- Separate computers interconnected by long-haul networks
  - E.g., Internet connections
  - Work units farmed out, results sent back
- Can make use of idle time on PCs
  - E.g., SETI@home, World Community Grid

- Other parallel computers
  - Supercomputers, vector computers, GPUs, cells, etc.

Concluding Remarks

- Goal: higher performance by using multiple processors

- Difficulties
  - Developing parallel software
  - Devising appropriate architectures

- Many reasons for optimism
  - Changing software and application environment
  - Chip-level multiprocessors with lower latency, higher bandwidth interconnect

- An ongoing challenge for computer architects, software/tool/compiler developers/researchers!