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)

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

§7.2 The Difficulty of Creating Parallel Processing Programs

Amdahl's Law

- Sequential part can limit speedup
- Example: 100 processors, 90× speedup?
  - $T_{new} = T_{parallelizable}/100 + T_{sequential}$
  - $$\text{Speedup} = \frac{1}{(1 - F_{parallelizable}) + F_{parallelizable}/100} = 90$$
  - Solving: $F_{parallelizable} = 0.999$
- 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 = $(10 + 100) \times t_{add}$
- 10 processors
  - Time = $10 \times t_{add} + 100/10 \times t_{add} = 20 \times t_{add}$
  - Speedup = 110/20 = 5.5 (55% of potential)
- 100 processors
  - Time = $10 \times t_{add} + 100/100 \times t_{add} = 11 \times t_{add}$
  - Speedup = 110/11 = 10 (10% of potential)
- Assumes load can be balanced across processors

Scaling Example (cont)

- What if matrix size is 100 × 100?
- Single processor: Time = (10 + 10000) × t_{add}
- 10 processors
  - Time = 10 × t_{add} + 10000/10 × t_{add} = 1010 × t_{add}
  - Speedup = 10010/1010 = 9.9 (99% of potential)
- 100 processors
  - Time = 10 × t_{add} + 10000/100 × t_{add} = 110 × t_{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 × 10 matrix
    - Time = $20 \times t_{add}$
  - 100 processors, 32 × 32 matrix
    - Time = $10 \times t_{add} + 1000/100 \times t_{add} = 20 \times t_{add}$
  - Constant performance in this example