Course Review

Mark Greenstreet

CpSc 418 – April 8, 2016

- Algorithms
- Architectures
- Performance
- Paradigms

Parallel Algorithms

- data-parallel maps
- tree-structured algorithms
- sorting

Data-Parallel maps

- simulation
  - for multiple (initial) conditions: logistic map
  - Monte-Carlo simulation: percolation
  - Many applications:
    - ★ Scientific computing: molecular dynamics, climate modeling, quantum mechanics, social sciences, . . .
    - ★ Financial (risk analysis), games (generate move-trees)
- matrix multiplication
  - The dot-products for each result element can be computed independently
  - Or, compute rows, columns, or blocks independently.
- map-reduce
  - Really data-parallel with a “sort” (or scatter, if you prefer) step in the middle.
  - Useful for analysing large data-sets.

Reduce and Scan

- Tree-structure for communication
- Can be used with any associative operator.
- Obvious examples include: sum, product, max, and similar – both for the final total (reduce) and the prefix versions (scan).
- Many other problems can be formulated as reduce or scan if they can be performed using an associative operator.

Sorting

- Sorting Networks
  - Describe a sorting-network as a composition of compare-and-swap modules.
  - Describe a sorting-network as a decision tree.
  - Given a small network of compare-and-swap modules, draw the decision tree.
  - Given a small decision tree, draw the network of compare-and-swap modules.
  - Given a small sorting network, fill in what it computes for a specified input.
- The 0-1 principle
  - Be able to state it.
  - Explain how it is used in deriving/justifying the bitonic sort algorithm.
  - Explain why it applies to sorting networks, but not arbitrary programs.
- Bitonic sort
  - It's merge-sort, with bitonic merge
  - What is a bitonic sequence?
  - The key ideas behind bitonic merge

Bitonic Questions

- How the odd and even subsequences have nearly matched numbers of ones.
- Why this makes the final step of merge “easy”.
- The recursive structure of the merge.
- $O(N \log^2 N)$ comparisons
- $O(\log^2 N)$ time
- be able to justify both (in simple terms).

Parallel Architectures

Parallelism in CPU Architectures

- Pipelined execution
  - The goal: one instruction per cycle
  - Pipelining: an assembly line for instruction execution
  - Data hazards:
    - ★ bypassing eliminates many hazards,
    - ★ for the others, the pipeline stalls.
  - Control hazards:
    - ★ Move branch execution to a early pipeline stage
    - ★ Expose the hazard: delay slots.
    - ★ Or stall – common on multithreaded architectures.
- Supescalar
  - The goal: many instructions per cycle
  - The solution: dynamically compute dependencies, resolve instructions when dependencies resolved.
  - Key techniques: register renaming, branch prediction and speculation.
  - Scalability: poor – hardware size grows as Parallelism^2.

Message Message Passing Architectures

- Many machines communicating through a network.
- Issues of network topology:
  - Network diameter: determines worst-case latency.
  - Cross-section bandwidth: performance for communication intensive algorithms.
  - Local bandwidth: often, algorithms are faster if they exploit communication between neigbouring machines.
  - Scalability: if the topology isn't naturally 3-dimensional, the machine becomes "all wire" as the number of processors grow.
- In practice:
  - Clusters for commercial applications usually use commodity networks (e.g. ethernet).
  - Scientific super computers are message-passing machines that use high-performance networks (e.g. infiniband).
  - In either case, the networking hardware can be as expensive as the computers themselves.

Shared memory Architectures

- Dividing memory into banks, and using a switching network between the processors and the memory
  - This is the approach of 1970's shared memroy machines.
  - It's also used for the on-chip shared memory of GPUs.
- Use caches and a coherence protocol
  - The approach used for multi-core CPUs
  - MESI: be able to describe the protocol, explain what happens for reads and writes.
  - Weak-consistency: know that real CPUs don't quite provide sequential consistency
    - ★ Use a thread library
    - ★ In the rare case that's note enough, you'll need to learn about the subtleties of real-world, cache-coherence protocols (a "known unknown").

Data parallel architectures

- GPU and SIMD execution
- The basic structure of a “streaming multiprocessor”
- Predicated execution and branch divergence.
- Exposed memory hierarchy

Parallel Performance

Speed-Up

- Definition
- Amdahl's law
- Superlinear speed-up
- Why speed-up can be hard to measure in practice
  - Difficulties in defining the "best" sequential implementation
  - Doesn't exist, no-time to implement, couldn't run on a realistic problem size, ...

Overhead

- Types of overhead
  - Be able to describe and give examples

Model

- Communcation costs matter.
- The CTA λ model:
  - Communicating N "words" costs $\lambda + N$.
  - Why do both terms matter in the model?
- Implicit communication such as synchronization and shared-memory accesses also incurs these costs.

Parallel Paradigms

The Message Passing Paradigm

The Data Parallel Paradigm

The Shared Memory Paradigm