Lecture 17: Performance Issues
Abhinav Bhavele, Department of Computer Science
Announcements

• Assignment 3 is due on Nov 9

• Interim report for the group project is due on Nov 16
  • Provide more details about the project: serial algorithm, parallel algorithm, languages being used
  • Deliverables and metrics for success
  • Contributions of individual group members
Performance metrics

- Time to solution
- Time per step (iteration)
- Science progress (figure of merit per unit time)
- Floating point operations per second (flop/s)

When comparing multiple data points:
  - Speedup, efficiency
What is the best performance we can get?

- Peak flop/s
- Peak memory bandwidth
- Peak network bandwidth
- Why do we not achieve peak performance?
What is happening in a program

- Integer operations
- Floating point operations
- Conditional instructions (branches)
- Loads/stores
- Data movement across the network (messages + I/O)
Performance issues

- Algorithmic overhead
  - More computation when running in parallel (e.g. prefix sum)

- Speculative loss
  - Perform extra computation speculatively but not use all of it for the result

- Critical paths
  - Dependencies between computations spread across processes / threads

- Bottlenecks
  - Serial bottlenecks: one process doing some computation and holding everyone up
Performance issues

- Sequential performance issues
  - Inefficient memory access: data movement in the memory hierarchy
  - Inefficient floating point operations

- Load imbalance
  - Some processes doing more work than most

- Communication performance
  - Spending increasing proportion of time on communication
Communication performance

- Overhead and grain size (Lots of tiny messages or a few very large messages)
- No overlap between communication and computation
- Increasing amounts of communication as we run on more processes
- Frequent global synchronization
Critical paths

- A long chain of dependencies across processes
- We want to identify and avoid having long critical paths
- Solutions:
  - Eliminate completely if possible
  - Shorten the critical path
  - Reduce time spent in a path by removing work on the critical path
Bottlenecks

• Detect bottlenecks
  • One process busy while all others wait

• Examples:
  • Reduce to one process and then broadcast
  • One process responsible for input/output
  • One process responsible for assigning work to others

• Solutions:
  • Parallelize as much as possible, use hierarchical schemes
Sequential performance issues

• Identify issues using performance tools

• Solutions:
  • Minimize data movement
  • Data reuse
  • Optimize floating point calculations
Abhinav Bhatele
5218 Brendan Iribe Center (IRB) / College Park, MD 20742
phone: 301.405.4507 / e-mail: bhatele@cs.umd.edu