Parallel Applications

Kai Shen

Parallelism and Dependencies

- Computation-intensive
- Contain tasks that can run in parallel
- Dependencies limit available parallelism/concurrency
  - Amdahl’s law:
    \[
    \text{Normalized runtime} = \frac{1}{(1 - \text{frac}_{\text{enhanced}}) + \frac{\text{frac}_{\text{enhanced}}}{\text{speedup}_{\text{enhanced}}}}
    \]
- Examples of dependencies?

Load Balance

- Parallel processors are typically symmetric
- Best utilizations of parallel processors
  - Require load balancing

Simulation of Ocean Currents

- Entity in ocean floor: a unit body of water
- Variables: velocity, direction, ...
- Simulation over time
  - Velocity/direction of water body depends on the
    velocity/direction of nearby water bodies in the last time unit

  - Parallelism/concurrency
  - Dependencies
  - Load balance
Evolution of Galaxies
- Entity in space: a star
- Variables: velocity, direction, location, ...
- Simulation over time
  - Velocity/direction/location of a star depends on the mass and location of other stars in the last time unit
- N-body simulation
- Parallelism/concurrency
- Dependencies
- Load balance

Ray Tracing
- Ray tracing
- Parallelism/concurrency
- Dependencies
- Load balance

Google PageRank
- To calculate the probability of arriving at a page after clicking a link at random from another page
- Page A is linked by pages B and C, then:
  \[ PR(A) = PR(B)/L(B) + PR(C)/L(C) \]
- Add a damping factor:
  \[ PR(A) = (1-d)/N + d(PR(B)/L(B) + PR(C)/L(C)) \]
- We arrive at a large linear system of equations
- Parallelism/concurrency
- Dependencies
- Load balance

Parallel Programming Models
- Shared memory
  - Writes to a shared location are visible to all
  - Example language: pthreads
- Message passing
  - No shared memory; data is partitioned and must be shared through explicit communication
  - Example language: MPI
- Data parallel
  - Multiple processors run the same code on different data, or single-instruction-multiple-data (SIMD)
  - Example language: High Performance Fortran
- Parallel machines available to match programming models. But the matching isn’t necessary.
Data Locality
- Minimize communications for message passing parallel computing
- Improve caching performance for shared-memory parallel computing

Synchronization
- Maintain dependencies
- Examples of synchronization primitives?
- Implementation is sometimes complex
- Essential for correct computing semantics
- Incur overhead ⇒ offset benefits of parallel computing

Parallel Programming Steps
Converting a sequential application to a parallel one
- Decomposition into tasks
  - Formalize the dependencies; expose concurrency
  - Fine-grain vs. coarse-grained decomposition
- Assign tasks to processors
  - Balance load; maximize locality
  - Static vs. dynamic task assignment
- Orchestration
  - Name and access shared data; synchronization
  - Synchronization frequency vs. promptness
- 1-to-1 binding between processes to processors?

Embarrassingly Parallel Applications
- Require large computing resources ⇒ candidates for parallel computing
- Easily partitioned to fine-grained tasks without inter-task dependencies ⇒ trivial to parallelize
- SETI@home, brute-force password cracking
  - Task: data for processing, passwords to try
- Internet servers
  - Task: request
  - Additional issues: interactivity/QoS/fairness to individual users; 24x7 availability
- Large back-end data processing (as in mapreduce)