Parallel Applications

Kai Shen

Parallelism and Dependencies

- Computation-intensive
- Contain tasks that can run in parallel sometimes
- Dependencies limit available parallelism/concurrency
  - Control dependencies
  - Data dependencies
- Amdahl's law:

$$\text{Normalized runtime} = 1 - \text{fraction\_enhanced} + \frac{\text{fraction\_enhanced}}{\text{speedup\_enhanced}}$$

Load Balance

- Inefficient if many processors are idle while one processor has lots of work to do and slowdown the whole application
- Best utilizations of parallel processors
  - Require load balancing (parallel processors are typically symmetric

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
  - http://upload.wikimedia.org/wikipedia/commons/8/83/Ray_trace_diagram.svg
- Parallelism/concurrency
- Dependencies
- Load balance

Data Mining

- Data mining and machine learning techniques are used widely for big data analytics
- Example: K-means clustering of samples
  - Each iteration of K-means: given K centers, each sample is grouped into the nearest center
- Parallelism/concurrency
- Dependencies
- Load balance

Google PageRank

- PageRank …
  - Originated at Google
  - "works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites."
- Count of in-links is important; the quality of the sources of in-links are also important
  - http://upload.wikimedia.org/wikipedia/commons/6/69/PageRank-hi-res.png

Google PageRank

- PageRank scores each page
  - the score is the converged probability for a random web surfer to arrive at the page
- Scores transfer through links (random click by the surfer):
  - If pages B, C are the pages that link to A, then PR(A) = PR(B)/L(B) + PR(C)/L(C)
- Damping factor (for stability):
  - The surfer has a tendency to stay at where he/she is
  - PR(A) = (1-d)/N + d(PR(B)/L(B) + PR(C)/L(C))
- A linear system of equations (N variables, N linear equations)
- Parallelism/concurrency
- Dependencies
- Load balance

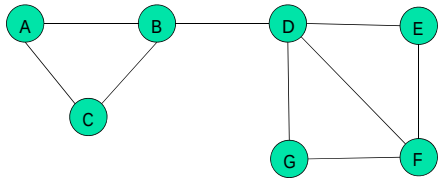
PageRank (Iterative Solver)

- PR(A) = (1-d)/N + d(PR(B)/L(B) + PR(C)/L(C))
- Iterative solver:
  - Start from an initial PR vector (1/N for each page)
  - Compute a new PR vector by the above linear transformation
  - Keep repeating the linear transformation until the PR converges (very small change after further linear transformation)
    - Even with the large, complex web graph, often converging after dozens of iterations
- Each iteration is a matrix-vector multiplication, so the whole computation is a series of matrix-vector multiplication
- Parallelism/concurrency
- Dependencies
- Load balance

Social Network Graph Analysis

- Social networks (Facebook, Google+, …) are large
  - Containing information on many users and their interactions
- Analyze friendship and relationships
  - Assess locality of relationship ⇨ assess the evolution or maturity of the social network over time
  - Communities (strongly connected subgroups) ⇨ maybe common interests so we can cluster users and sell targeted ads!

Social Network Clustering



- Use graph semantics: identify and cut edges that are least likely to be inside a cluster
- Betweenness of edge e
  - Number of pairs of nodes (X,Y) such that edge e lies on the shortest path between X and Y
  - How does it tell the likelihood of an edge to be inside a cluster?
- Remove edges on order of their betweenness until we settle

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, MapReduce/Hadoop
- Each programming model desires certain hardware and system software.

Maintain Dependencies

- Maintain dependencies
  - In distributed memory parallel computing: messages
  - In shared-memory parallel computing: synchronization
- Examples of synchronization primitives?
  - Implementation is sometimes complex (may require hardware assistance)
- Incur overhead ⇨ offset benefits of parallel computing

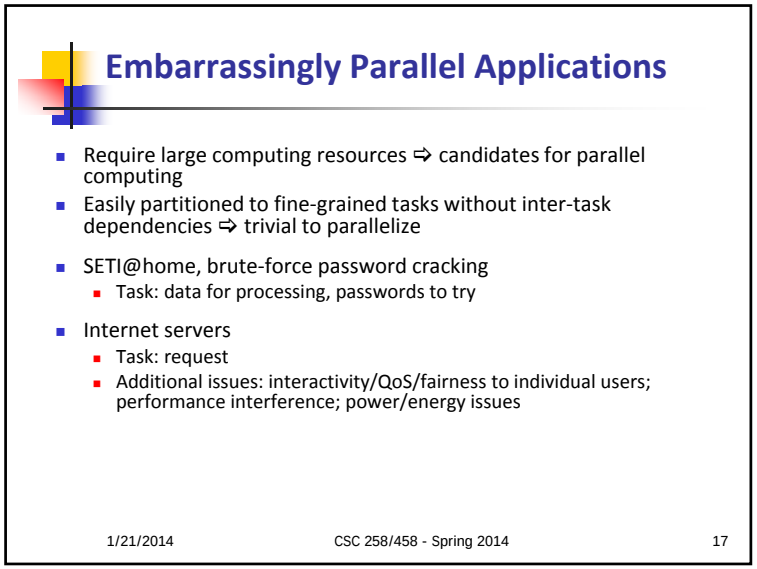
Data Locality

- Structure/configure/schedule a parallel program so that the data needed for computation is near the processor performing the computation
  - Minimize communications for message passing parallel computing
  - Improve caching performance for shared-memory parallel computing
  - Reduce behind-the-scene data movements in a MapReduce/Hadoop system

Parallel Programming Steps

Converting a sequential application to a parallel one

- Decomposition into tasks
  - Goal: formalize the dependencies; expose concurrency
  - Approach: fine-grained vs. coarse-grained decomposition
- Assign tasks to processors
  - Goal: balance load; maximize data locality
  - Approach: static vs. dynamic task assignment
- Orchestration
  - Goal: name and access shared data; synchronization
  - Approach: high vs. low synchronization frequency


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; performance interference; power/energy issues
1/21/2014
CSC 258/458 - Spring 2014
17