Lecture 19: Load Balance
Abhinav Bhave, Department of Computer Science
Announcements

- Quiz 2: Nov 12
- Quiz 3: Dec 3
Performance issues

- Algorithmic overhead
- Speculative loss
- Critical paths
- Bottlenecks
- Sequential performance issues
- Load imbalance
- Communication performance
Load imbalance
Load imbalance

• Definition: unequal amounts of “work” assigned to different processes

  • Work could be computation or communication or both
Load imbalance

- Definition: unequal amounts of “work” assigned to different processes
  - Work could be computation or communication or both

- Why is load imbalance bad?
  - Overloaded processes can slow down everyone
Load imbalance

- Definition: unequal amounts of “work” assigned to different processes
  - Work could be computation or communication or both

- Why is load imbalance bad?
  - Overloaded processes can slow down everyone

\[
\text{Load imbalance} = \frac{\text{max\_load}}{\text{avg\_load}}
\]
Load balancing

• The process of balancing load across threads, processes etc.
• Goal: to bring the maximum load close to average as much as possible
• Determine if load balancing is needed
• Determine when to load balance
• Determine what information to gather/use for load balancing
Is load balancing needed?

- Need the distribution of load (“work”) across processes
- Collect empirical information using performance tools
- Developer knowledge
- Analytical models of load distribution
When to load balance?

- Initial work distribution or static load balancing
  - At program startup
  - Or sometimes in a separate run to determine load distribution
- Dynamic load balancing: does load distribution evolve over time?
  - During program execution
Information gathering for load balancing

- Centralized load balancing
  - Gather all load information at one process — global view of data

- Distributed load balancing
  - Every process only knows the load of a constant number of “neighbors”

- Hybrid or hierarchical load balancing
What information is used for load balancing

- Computational load
- Possibly, communication load (number/sizes of messages)
- Communication graph
Load balancing algorithms

- **Input:** Amount of work \((n_i)\) assigned to each process \(p_i\)
- **Output:** New assignments of work units to different processes
- **Goals:**
  - Bring maximum load close to average
  - Minimize the amount of load migration
- **Secondary goals:**
  - Balance (possibly reduce) communication load
  - Keep the time for doing load balancing short
Examples of static load balancing

- Decomposition of 2D Stencil
- Using orthogonal recursive bisection (ORB)

http://datagenetics.com/blog/march22013/
https://en.wikipedia.org/wiki/Z-order_curve
Examples of static load balancing

- Decomposition of 2D Stencil
- Using orthogonal recursive bisection (ORB)

http://datagenetics.com/blog/march22013/
https://en.wikipedia.org/wiki/Z-order_curve
Examples of static load balancing

- Decomposition of 2D Stencil
- Using orthogonal recursive bisection (ORB)

[Websites and references provided for further reading on space-filling curves and ORB.]
Simple greedy strategy
Other considerations

- Communication-aware load balancing
- Network topology-aware load balancing
Hierarchical load balancing

The hierarchical load balancing scheme involves the following steps:

1. When load balancing decisions are made, lightweight tokens that carry only the objects' workload data are created and sent to the destination group leaders of the sub-domains.
2. These tokens represent the movement of objects from an overloaded domain to an underloaded domain.
3. When the tokens representing the incoming objects arrive at the destination group leader, their load data are integrated into the existing load database on that processor.
4. After this phase, the load database of all the group leaders at the lower level domains is updated, reflecting the load balancing decisions made – new load database entries are created for the incoming objects, and load database entries corresponding to the outgoing objects are removed from the database.
5. This new database can then be used to make load balancing decisions at that level.
6. At the intermediate levels of the tree, load balancing decisions are made in the form of which object migrates to which sub-domain.
7. This process repeats until load balancing reaches the lowest level, where final load balancing decisions are made on migrating objects and their final destination processors.

At this point, tokens representing a migration of an object may have traveled across several load balancing domains, therefore its original processor needs to know which final destination processor the token has traveled to. In order to match original processors with their tokens, a global collective operation is performed on the tree. By sending tokens instead of actual object data in the intermediate load balancing phases of the hierarchical tree, this load balancing scheme ensures that objects are only migrated once after all the final migration decisions are made.