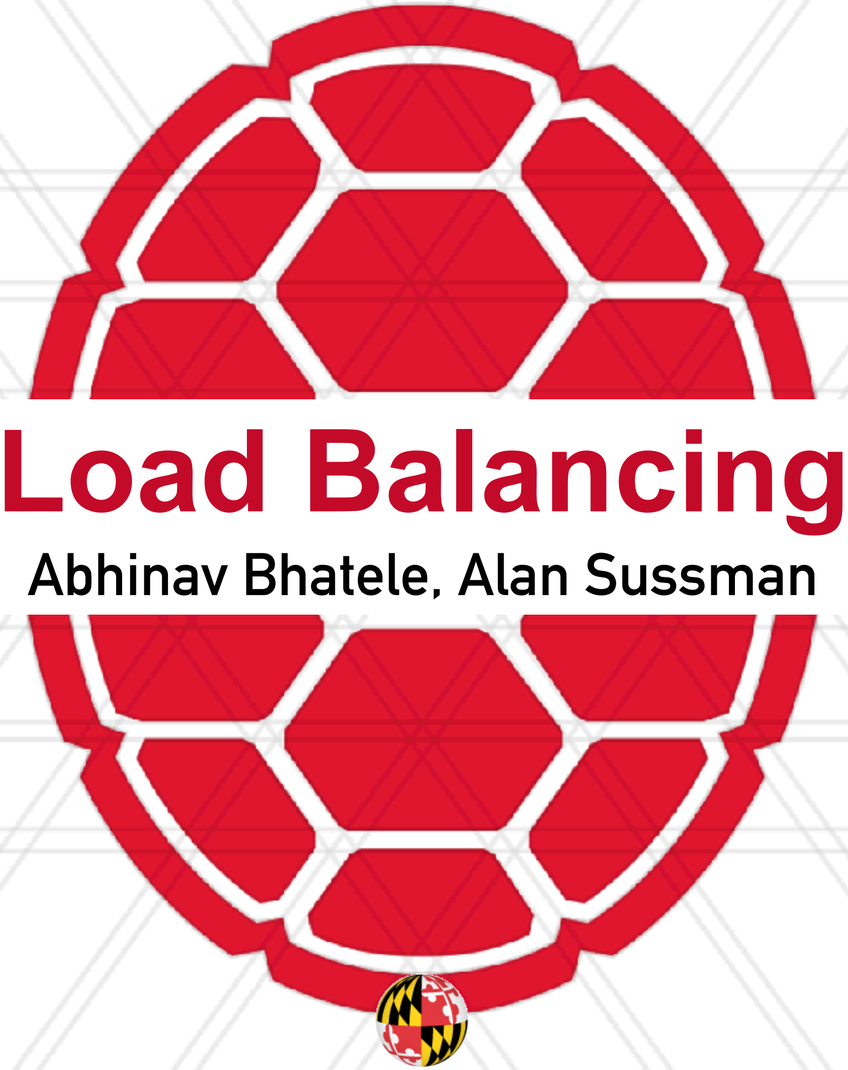
Introduction to Parallel Computing (CMSC416 / CMSC616)



Load Balancing

Abhinav Bhatele, Alan Sussman

UNIVERSITY OF MARYLAND

Announcements

- We are using MOSS to detect plagiarism in programming assignments

Performance issues

- Sequential performance issues
- Load imbalance
- Communication performance issues / parallel overhead
- Algorithmic overhead / replicated work
- Speculative loss
- Critical paths
- Insufficient parallelism
- Bottlenecks

Load imbalance

Load imbalance

- Definition: unequal amounts of "work" assigned to different processes/threads
  - Work could be computation or communication or both

Load imbalance

- Definition: unequal amounts of "work" assigned to different processes/threads
  - Work could be computation or communication or both
- Why is load imbalance bad?
  - Overloaded processes can slow down all processes

Load imbalance

- Definition: unequal amounts of "work" assigned to different processes/threads
  - Work could be computation or communication or both
- Why is load imbalance bad?
  - Overloaded processes can slow down all processes

$$\text{Load imbalance} = \frac{max\_load}{mean\_load}$$

Load balancing

- The process of balancing load across threads, processes etc.
- Goal: to bring the maximum load close to average as much as possible
- Steps for balancing load include:
  - Determine if load balancing is needed
  - Determine when and how often to load balance
  - Determine what information to gather/use for load balancing
  - Choose/design a load balancing algorithm

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

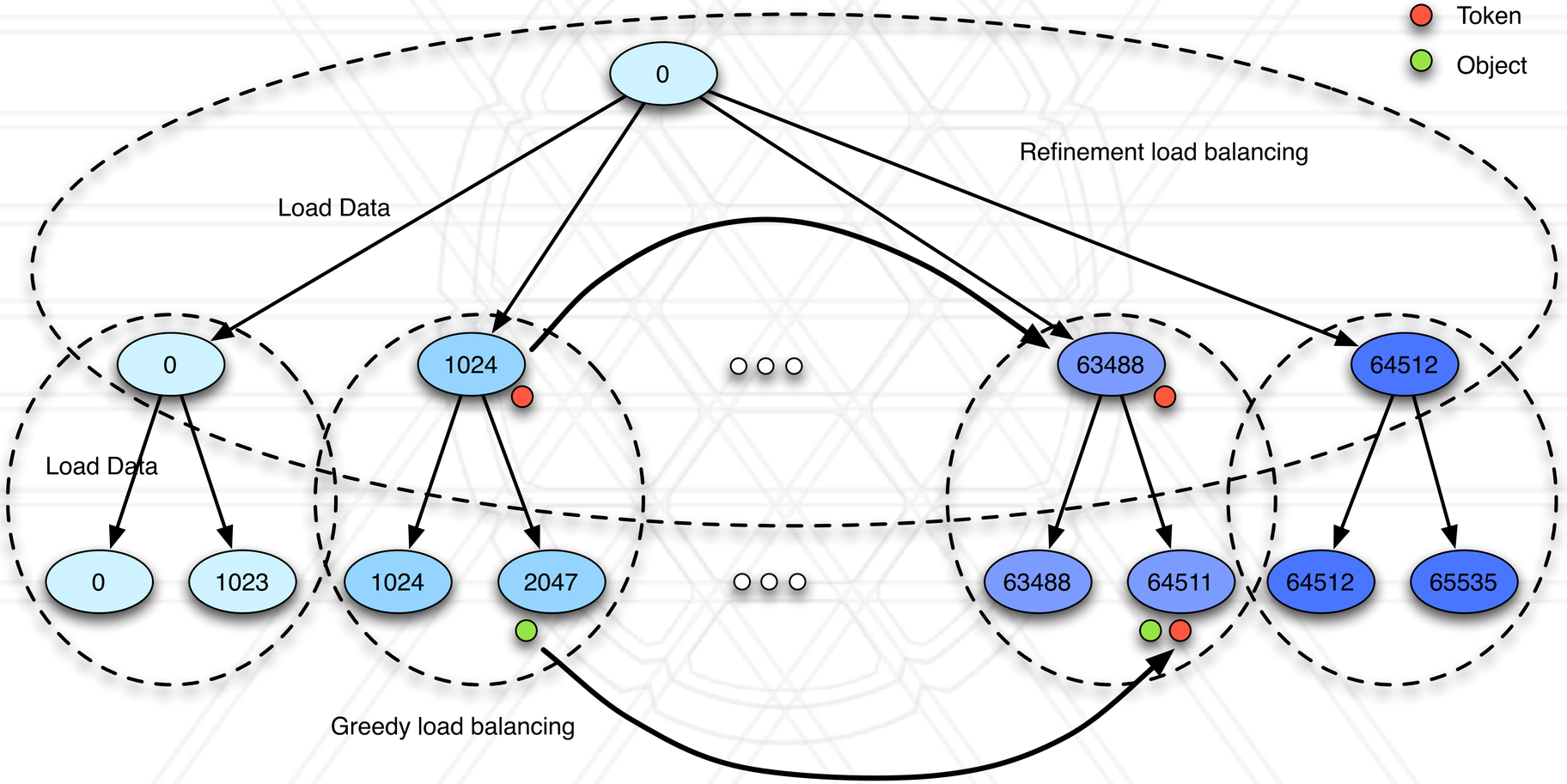
When/how often to load balance?

- Initial work distribution or partitioning or static load balancing
  - At program startup
  - Or sometimes in a separate run to determine new load distribution
- Dynamic load balancing: does load distribution evolve over time?
  - During program execution
  - How often? It depends …

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

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 data migration
- Secondary goals:
  - Balance (possibly reduce) communication load (volume)
  - Keep the time for doing load balancing to a minimum

Examples of static load balancing

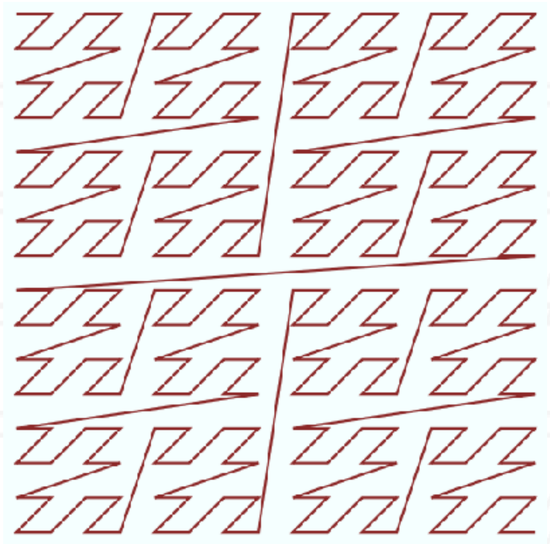
- Decomposition of *n*-D Stencil
- Using orthogonal recursive bisection (ORB), space-filling curves, etc.

http://datagenetics.com/blog/march22013/

https://en.wikipedia.org/wiki/Z-order_curve

Examples of static load balancing

- Decomposition of *n*-D Stencil
- Using orthogonal recursive bisection (ORB), space-filling curves, etc.



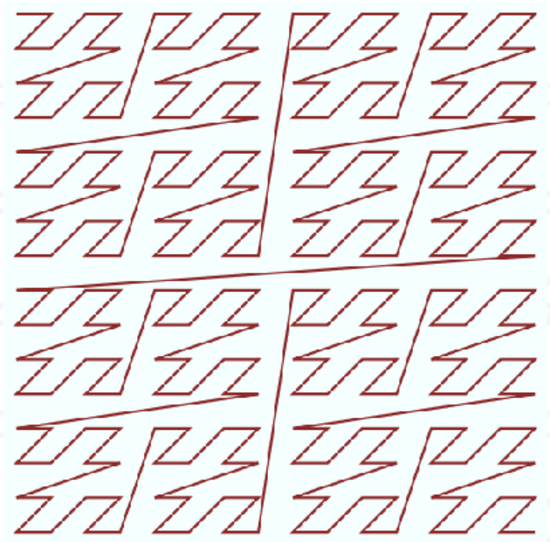
http://datagenetics.com/blog/march22013/

https://en.wikipedia.org/wiki/Z-order_curve

Examples of static load balancing

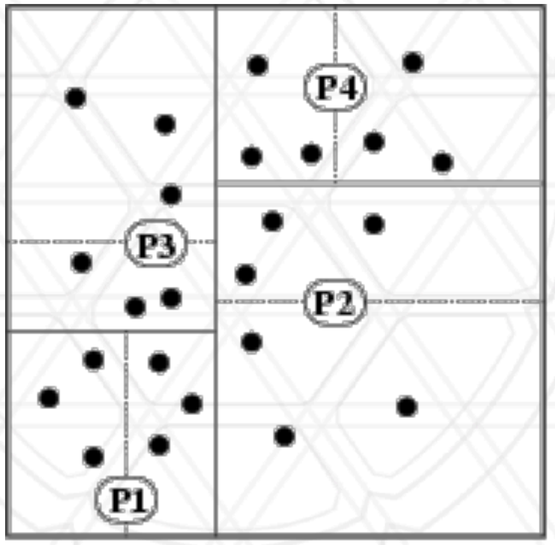
- Decomposition of *n*-D Stencil
- Using orthogonal recursive bisection (ORB), space-filling curves, etc.

Space-filling curves

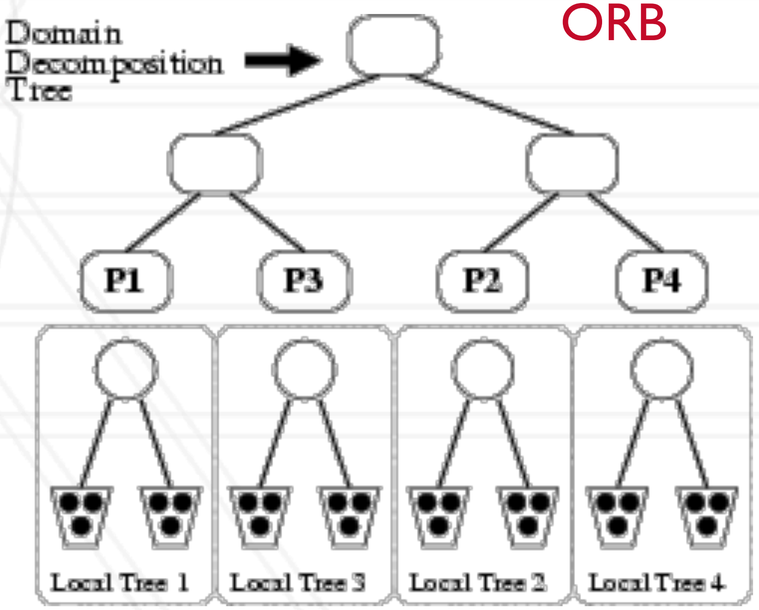


http://datagenetics.com/blog/march22013/

https://en.wikipedia.org/wiki/Z-order_curve



ORB



http://charm.cs.uiuc.edu/workshops/charmWorkshop2011/slides/CharmWorkshop2011_apps_ChaNGa.pdf

Simple greedy strategy

- Sort all the processes by their load
- Take some load (work) from the heaviest loaded process and assign that work to the most lightly loaded process

Work stealing

- Decentralized strategy where processes steal work from nearby processes when they have nothing to do
- Each process has a queue of work items
  - Looks at the other processes' queues when there are no items remaining
- Implemented in Cilk, among other languages

Other considerations

- Communication-aware load balancing
  - Try to move (units of) work to processes that this work communicates with frequently
- Network topology-aware load balancing
  - Take into account how the nodes are connected to one another to minimize some metrics (number of hops, average link load etc.)

UNIVERSITY OF
MARYLAND