Introduction to Parallel Computing (CMSC416)

Load Balancing

Alan Sussman, Department of Computer Science

UNIVERSITY OF
MARYLAND

Announcements

- OpenMP Assignment 3 due Wed., April 12
  - questions?
- Midterm exam grades should be available by end of day tomorrow

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

- 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 all processes

$$\text{Loadimbalance} = \frac{max\_load}{mean\_load}$$

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 to follow include:
  - 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

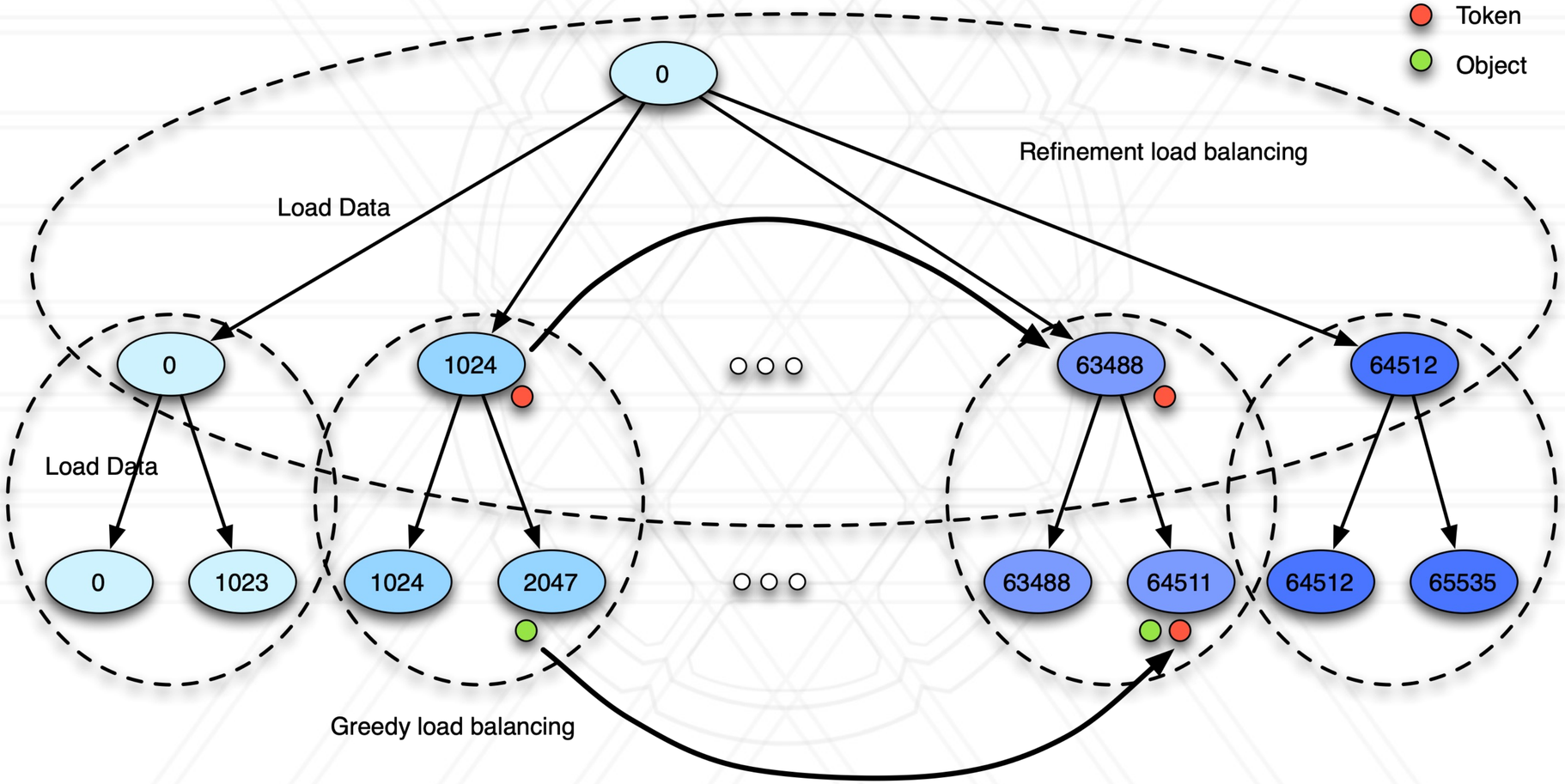
When to load balance?

- Initial work distribution or static load balancing
  - At program startup
  - Or sometimes in a separate execution 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

Hierarchical load balancing



What information is used for load balancing

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

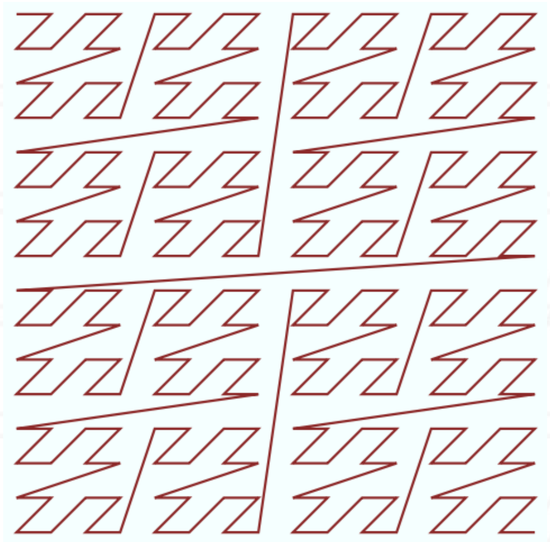
Load balancing algorithms

- 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
  - Keep the time for doing load balancing to a minimum

Examples of static load balancing

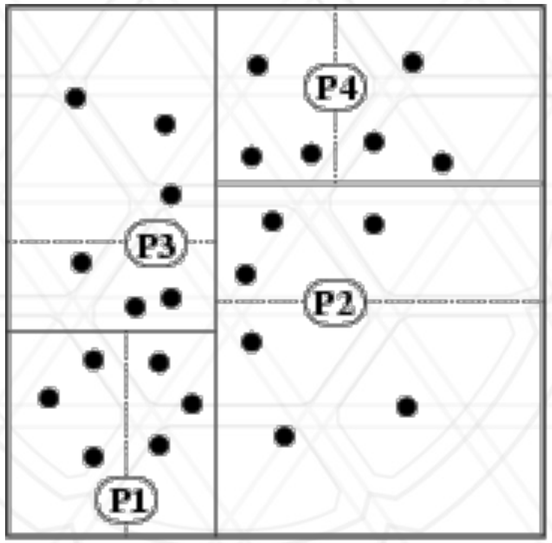
- Decomposition of 2D (or higher-D) Stencil
- Using orthogonal recursive bisection (ORB), space filling curves, etc.

Space-filling curves

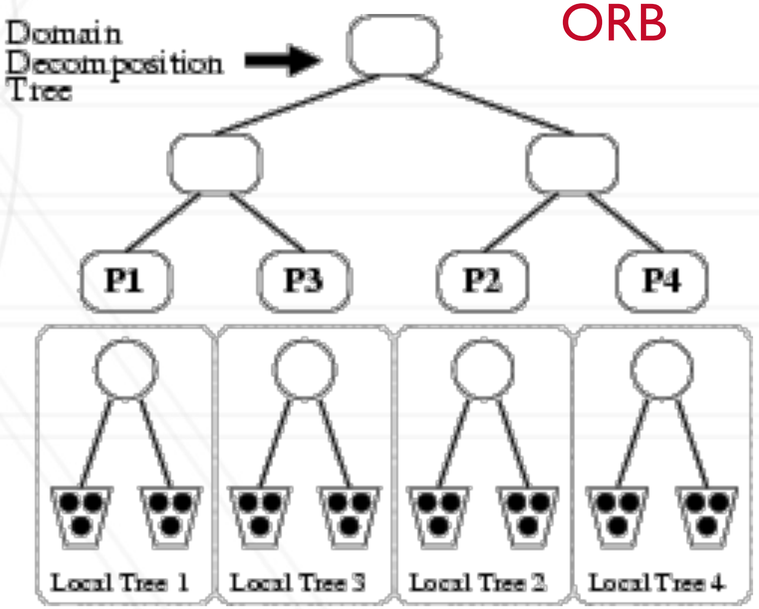


http://datagenetics.com/blog/march22013/

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 from the heaviest loaded process (the one with the most work) 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, and various systems

Other considerations

- Communication-aware load balancing
  - Try to move units of work to where other units that depend on them (so communicate) are already located, to minimize communication
- Network topology-aware load balancing
  - Take into account how nodes are connected to move work near where its results will be needed

UNIVERSITY OF
MARYLAND