Cluster-based Internet Services: Load Balancing and Request Distribution

Kai Shen
Dept. of Computer Science, University of Rochester

Cluster Load Balancing

- Load index
  - number of active service requests
  - CPU/memory/IO load
- "Broadcast" policy
  - let each node announce its workload to everybody else once a second
  - every node monitors all such announcements and maintains a local node/workload mapping table
  - load balancing is merely a local table lookup
  - can even be piggybacked with the service announcement messages

Problems with the "Broadcast" Policy

- load index staleness
  - 1 second broadcast intervals are too long for fine-grain (small) services
- flocking effect
  - all nodes have the same view between broadcast intervals
  - all nodes will choose the same node as the lightest loaded
- how to measure?
  - performance metric
  - input variables
    - broadcast frequency, service granularity, average system load

Performance of Broadcast Policy (16-node simulation)

- Too much dependent on frequent broadcasts for fine-grain services at high load.
- Reasons: load index staleness, flocking effect.
Random Polling Policy

- For each service request, a polling agent on the client node
  - randomly polls a certain number (poll size) of service nodes for load information;
  - picks the node responding with the lightest load.
- Random polling with a small poll size.
  - Requires no centralized components;
  - Per-request overhead is limited by the poll size;
  - Small load information delay due to just-in-time polling.

Any question here?

Is a Small Poll Size Enough? (simulation)

Cluster Load Balancing

- Any other ideas?

Locality-aware Request Distribution: the Idea

- Based on a paper by Pai et al at ASPLOS VIII.
- Basic idea: locality-based distribution improves the chance of directly serving out of memory; reduces disk I/O.
- Question: how to balance load and locality?
- Strategy:
  - each node maintains an affinity table based on past requests
  - each new request is directed based on load
  - each old request is directed to the node that it went to before; unless that node is too heavily loaded
- Any problems with this?
Locality-aware Request Distribution: the Practice

- Locality-aware request distribution in action:
  - for applications with little processing, lots of I/O; like the Web
  - all requests pass by a single front-end node

- An alternative:
  - using partitioning or other design techniques to make sure the working set always fit into each node's memory
  - then distributing requests strictly based on load