Locality-Aware Request Distribution in Cluster-based Network Servers

Vivek S. Pai‡, Mohit Aron†, Gaurav Banga†,
Michael Svendsen†, Peter Druschel†,
Willy Zwaenepoel†, Erich Nahum¶

‡ Department of Electrical and Computer Engineering, Rice University

† Department of Computer Science, Rice University

¶ IBM T.J. Watson Research Center

Oct 6, 1998
Motivation

Growing demands on network servers
  • growth of Internet
  • bandwidth increases
  • site consolidation

PCs & LANs are attractive
  • cheap hardware
  • parallelizable problem

Our goal: to improve throughput
Our focus: front-end request distribution policy
Weighted Round Robin

Locality-Aware Request Distribution...
Insight

Use request content to improve locality example: divide working set

---

Locality-Aware Request Distribution...
Contributions

- Using content to improve locality
- Locality-Aware Request Distribution achieving both load balance and locality
- Simulation studies and comparisons
- Prototype implementation & evaluation
Purely Locality-Based

Goal: “equally” divide working set
LARD Approach

Goal: divide working set to balance work
Note: no backend cache modeling
Request Assignment

Purely Locality-Based

- assign a backend node for each unique URL
- record assigned backend for each URL
- in all cases, send requests for that URL to that backend

LARD

- assign initial backend for each unique URL
- record last assigned backend for each URL
- if balanced, use last assignment
- if unbalanced, pick new assignment
Defining Load

Load = \# of active connections at a node

- good summary metric
- robust
- available in front-end
- simple
- quickly adapts
Back-end Load Levels

- Unacceptable
- Acceptable
- Delay Difference
- Target Delay Difference
- Fully utilized
- Underutilized

Locality-Aware Request Distribution...
Balancing Decisions

Always Reassign (unbalanced)

Potentially Reassign (balanced?)

Never Reassign (balanced)

(unbalanced)

Load

Unacceptable Delay Difference

Acceptable Delay Difference

Target Delay Difference

Underutilized
LARD Strategy

find last backend node assignment for URL

if (balanced) /* common case */
    send request to that backend node
else
    reassign to least-loaded node

Define unbalanced as:
(load > \( T_{limit} \)) OR
(load > \( T_{high} \) AND \( \exists \) node with load < \( T_{low} \))
LARD Strategy (cont.)

- Unacceptable
- Acceptable Delay Difference
- Target Delay Difference
- Underutilized

Load
Simulation Model

CPU model: 300 MHz Pentium II

Network: 100 Mbps

Disk model: Western Digital Enterprise

Caching: Greedy-Dual-Size (Cao & Irani)

Filesystem: Berkeley FFS

Input: Rice trace
and www.ibm.com trace

Strategies: WRR, LB-locality, LARD
Throughput

Locality-Aware Request Distribution
Miss Rate

- WRR
- LARD
- LB-Localiry

% reqs missed

# nodes in cluster

Locality-Aware Request Distribution...
Load Balance

% time both CPU, disk have idle time

# nodes in cluster

LB–Locality
LARD
WRR
Faster CPUs: LARD & WRR

Throughput (reqs/sec) vs. # nodes in cluster

Locality-Aware Request Distribution...
Extra Disks: LARD & WRR

- 4 disks ea.
- 3 disks ea.
- 2 disks ea.
- 1 disk each

Throughput (reqs/sec) vs. # nodes in cluster.

Locality-Aware Request Distribution.
Experimental Environment

300 MHz Pentium II

128 MB memory

Switched 100 Mbps Fast Ethernet

FreeBSD 2.2.5

Apache 1.2.4 webserver

Rice trace
Cluster Results

Throughput (reqs/sec) vs. # nodes in cluster.

- LARD
- WRR
In The Paper

Refinement of LARD
• includes explicit replication

Front-end information
• connection handoff design & performance
Ongoing Work

dynamic content

optimizations for persistent connections

parallel/distributed front-ends
Conclusions

LARD - Locality Aware Request Distribution
• miss rate near pure locality-based
• load balance near weighted round-robin
• no extra communications

Result: better throughput