CS 525M – Mobile and Ubiquitous Computing Seminar

Josh Schullman &
Ted Goodwin
Broadcast Disk Implementation

- Contents
  - Introduction
  - Design
  - Results
  - Conclusion
  - Future Work
Broadcast Disk Implementation

• Introduction
  – Implemented Multicast Broadcast disk
  • Client / Server
  • Cache Policy
  • Broadcast Disk Algorithm
  • Test Script
Broadcast Disk Implementation

– Design
  • Implementation tools
    – UDP Packets
    – Java Networking
    – Eclipse IDE for development
Network Layout

Server

UDP Multicast

Clients

802.11
(not implemented)
Broadcast Disk Implementation

Design Diagram

- Client
  - Cache Policy
    - LRU
    - L
    - LIX
  - UDP Multicast
- Server
  - Broadcast Program Generation
  - Packet as byte Stream
Broadcast Disk Implementation

Design- Cache Policy

– Implemented
  • LRU - linked list accessed pages are moved to the top of the list
  • LIX - Link list for each broadcast disk smallest lix value of bottom pages ejected.
  • L - like LIX except same frequency value for each disk

– Not Implemented
  • P – highest access probability in cache
  • PIX – lowest ratio of access probability to broadcast frequency
Design- Cache Policy

- Lix Example
  - $p_i = \frac{?}{(\text{currentTime} - t_i)} + (1 - ?)p_i$
  - $\text{lix} = \frac{p_i}{\text{frequency of the page}}$

![Diagram of page replacement in LTA](image-url)
Design – Broadcast Program Generation
- Create page vector
- Apply noise values
- Calculates Chunks per disk
- Generate relative frequency per disk
- Interleave chunks while applying offset
- Generate page list
Broadcast Disk Implementation

Design – Client
- Executes as a java thread
- Listens on UDP multicast
- Uses three different caches or no cache
- Receives the following parameters:

<table>
<thead>
<tr>
<th>CacheSize</th>
<th>Client cache size (in pages)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ThinkTime</td>
<td>Time between client page accesses (in broadcast units)</td>
</tr>
<tr>
<td>AccessRange</td>
<td># of pages in range accessed by client</td>
</tr>
<tr>
<td>θ</td>
<td>Zipf distribution parameter</td>
</tr>
<tr>
<td>RegionSize</td>
<td># of pages per region for Zipf distribution</td>
</tr>
</tbody>
</table>
Broadcast Disk Implementation

Design – Server
- Executed as a thread.
- Sends UDP packets over multicast
- Receives the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServerDBSize</td>
<td>Number of distinct pages to be broadcast</td>
</tr>
<tr>
<td>NumDisks</td>
<td>Number of disks</td>
</tr>
<tr>
<td>DiskSize_i</td>
<td>Size of disk i (in pages)</td>
</tr>
<tr>
<td>Δ</td>
<td>Broadcast shape parameter</td>
</tr>
<tr>
<td>Offset</td>
<td>Offset from default client access</td>
</tr>
<tr>
<td>Noise</td>
<td>% workload deviation</td>
</tr>
</tbody>
</table>
Broadcast Disk Implementation

![Bar chart showing % of pages accessed for LRU, L, and LIX for Cache, Disk1, Disk2, and Disk3.](image)
Broadcast Disk Implementation

<500,4500>, 0 offset, 0% noise, delta(5)
Figure 13: Sensitivity to $\Delta$ - Disk D5

CacheSize = 500. Noise = 30%
Broadcast Disk Implementation

<500,4500>, 0 offset, 0% noise
Response time vs. delta
Broadcast Disk Implementation

0 delta, 0 noise, 0% offset
Response time vs cache types/sizes…. ??????

null 0.00
L 50.00
LIX
LRU
L 250.00
LIX
LRU

<500, 4500>
<900, 4100>
Broadcast Disk Implementation

• Relative Frequency
  – The speed of a disk, relative to slowest disk.
  – relFreq(i)/relFreq(N) = 1 + ?(N – i)
  – N = the disk speed of the slowest disk.

• Max Chunks
  – max_chunks = LCM (relFreqs)

• NumChunks
  – num_chunks(i): the number of chunks the disk is broken into.
  – num_chunks(i) = max_chunks/rel_freq(i)
Delta = 0
relFreq(1) = 1, relFreq(2) = 1
numChunks(1) = 1, numChunks(2) = 1

Mini cycle 1

\[
\begin{array}{cccccc}
 & 0 & \ldots & 499 & 500 & \ldots & 4999 \\
\end{array}
\]

\[
\begin{array}{cccccc}
 & 0 & \ldots & 499 & 500 & \ldots & 4999 \\
\end{array}
\]

Delta = 1
relFreq(1) = 2, relFreq(2) = 1
numChunks(1) = 1, numChunks(2) = 2

Mini cycle 1

\[
\begin{array}{cccccc}
 & 0 & \ldots & 499 & 500 & \ldots & 2749 \\
\end{array}
\]

Mini cycle 2

\[
\begin{array}{cccccc}
 & 0 & \ldots & 499 & 2750 & \ldots & 4999 \\
\end{array}
\]

Broadcast Disk Implementation
Delta = 0
relFreq(1) = 1, relFreq(2) = 1
numChunks(1) = 1, numChunks(2) = 1

Mini cycle 1

Delta = 1
relFreq(1) = 2, relFreq(2) = 1
numChunks(1) = 1, numChunks(2) = 2

Mini cycle 1
Mini cycle 2
Future Work
- Run using wireless emulator
- Implement P and PIX
- Let tests finish running
- Implement Server in c or c++
- Run on dedicated network and machines
- Run on different packet sizes,
  - current 512 bytes
- Implement PT and APT caching methods.
• Conclusion
  – Time limitations and long run times, kept us from gathering all our results
  – Still L and LIX benefited from larger caches and larger deltas.
  – At small caches overhead of L and LIX not really worthwhile.