CS 525M – Mobile and Ubiquitous Computing Seminar

Ioanna Symeou
Broadcast Disks: Data Management for Asymmetric Communication Environments

Swarup Acharya, Brown University
Rafael Alonso MITL
Michael Franklin, University of Maryland
Stanley Zdonik, Brown University

• Asymmetric Communication Environments: downstream capacity differs from upstream capacity (ex. Wireless networks)

• Two reasons for asymmetry:
  – Bandwidth limitations
  – Patterns of information flow

• Improved performance in asymmetric communication environments
Broadcast Disks: Abstract & Introduction

- Pull based systems Vs push based systems:
  - Push based for asymmetric environments so channel becomes a disk
  - Items broadcasted more often appear to be on faster spinning disks closer to the clients.

- Broadcast disks: Broadcast technique providing “illusion” of having data on multiple disks running in different speeds.

- Broadcast algorithms: Based on client population and data access probabilities
- New cache management policies: How each client manages its own cache
Broadcast Disks: Broadcast Algorithms

- Simple scenario: Flat broadcast
  - Server broadcasts data from all requests
  - Wait time: half broadcast period
- Broadcast items with different frequency
  - Bandwidth allocation problem
  - Match needs of different clients
- Additional advantages

<table>
<thead>
<tr>
<th>Access Probability</th>
<th>Expected Delay (in broadcast units)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Flat</td>
</tr>
<tr>
<td></td>
<td>(a)</td>
</tr>
<tr>
<td>A 0.333</td>
<td>1.50</td>
</tr>
<tr>
<td>B 0.333</td>
<td>1.50</td>
</tr>
<tr>
<td>C 0.333</td>
<td>1.50</td>
</tr>
<tr>
<td>A 0.50</td>
<td>1.50</td>
</tr>
<tr>
<td>B 0.25</td>
<td>1.50</td>
</tr>
<tr>
<td>C 0.25</td>
<td>1.50</td>
</tr>
<tr>
<td>A 0.75</td>
<td>1.50</td>
</tr>
<tr>
<td>B 0.125</td>
<td>1.50</td>
</tr>
<tr>
<td>C 0.125</td>
<td>1.50</td>
</tr>
<tr>
<td>A 0.90</td>
<td>1.50</td>
</tr>
<tr>
<td>B 0.05</td>
<td>1.50</td>
</tr>
<tr>
<td>C 0.05</td>
<td>1.50</td>
</tr>
<tr>
<td>A 1.0</td>
<td>1.50</td>
</tr>
</tbody>
</table>
Broadcast Disks: Broadcast Algorithms

- Broadcast program generator
  - Order pages to lists from hottest to coldest
  - Partition lists to multiple ranges (disks) based on access probabilities
  - Choose relative frequency of broadcast for each disk
  - Split disk into smaller units (chunks)
    - $\text{Max\_chunks} = \text{LCM of frequencies}$
    - $\text{Num\_chunks} = \text{Max\_chunks} / \text{rel\_frequency}$
  - Send chunks:
    - for $i = 0$ to $\text{max\_chunks} - 1$
      - for $j = 1$ to $\text{num\_disks}$
        - broadcast chunk $j, i \mod(\text{num\_chunks}(j))$
Broadcast Disks: Broadcast Algorithm

- Three parameters influence broadcast:
  - Number of disks determines number of frequencies
  - Number of pages per disk and relative frequencies determine size of broadcast
- Disadvantage: Some slots may be unused
  - Send extra info
  - Small fraction of slots
  - Adjust relative frequencies to reduce unused slots
Broadcast Disks: Cache management

- Traditionally: Clients cache *their* hottest data
- BUT in push based systems broadcast might not be optimal for a client
  - Inaccurate/time sensitive information from client about its access distribution
  - Higher priority to other clients/large client population
- Cache data for which local access probability is much greater than their broadcast frequency
- Cost-based page replacement: When replacing a page on a cache miss calculate cost of obtaining the page
  - Replace page with lowest P/X ratio
  - Requires though knowledge of access probabilities and comparison of values for all pages
Broadcast Disks: Environment model

**Client parameters**

- **CacheSize**: Client cache size (in pages)
- **ThinkTime**: Time between client page accesses (in broadcast units)
- **AccessRange**: # of pages in range accessed by client
- **θ**: Zipf distribution parameter
- **RegionSize**: # of pages per region for Zipf distribution

**Server parameters**

- **ServerDBSize**: Number of distinct pages to be broadcast
- **NumDisks**: Number of disks
- **DiskSize**: Size of disk \( \pi \) (in pages)
- **Δ**: Broadcast shape parameter
- **Offset**: Offset from default client access
- **Noise**: % workload deviation

\[
\text{rel}_{-}\text{frequency}(i) / \text{rel}_{-}\text{frequency}(N) = (N - i)^2 + 1
\]
Broadcast Disks: Experiments and results

• Parameters settings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ThinkTime</td>
<td>2.0</td>
</tr>
<tr>
<td>ServerDBSize</td>
<td>5000</td>
</tr>
<tr>
<td>AccessRange</td>
<td>1000</td>
</tr>
<tr>
<td>CacheSize</td>
<td>50(5%), 250(25%), 500(50%)</td>
</tr>
<tr>
<td>Δ</td>
<td>1, 2, ..., 7</td>
</tr>
<tr>
<td>θ</td>
<td>0.95</td>
</tr>
<tr>
<td>Offset</td>
<td>0, CacheSize</td>
</tr>
<tr>
<td>Noise</td>
<td>0%, 15%, 30%, 45%, 60%, 75%</td>
</tr>
<tr>
<td>RegionSize</td>
<td>50</td>
</tr>
</tbody>
</table>

• Experiment 1: No caching, 0%Noise
Broadcast Disks: Experiments and results

- Experiment 2: No caching and noise
- Two disk configuration
  <2500,2500>
- Three disk configuration
  <300,1200,3500>
• Experiment 3: Caching and noise
  Three disk configuration
  \(<300,1200,3500>\>
  P replacement policy
  cache size = 500

• Experiment 4: Caching and noise
  Three disk configuration
  \(<300,1200,3500>\>
  P/X replacement policy
  cache size = 500
Broadcast Disks: Experiments and results

• Experiment 5: Caching and noise
  Three disk configuration <300,1200,3500>
  cache size = 500
  L/X replacement policy

• L/X replacement policy
  – One list for each disk
  – Always enter new page to a list according to its disk
  – Replace page with lower lix value
    • \( \text{lix} = \frac{p_i}{\text{rel\_frequency}} \)
    • \( p_i = \frac{?}{(\text{CurrentTime} - t_i)} + (1 - ?)p_i \)
Broadcast Disks: Conclusions

- Applicable technique for asymmetric environments
- Two and three level disks can have better performance than flat broadcast
- Need for new cache replacement policy (cost based caching)
- What about:
  - Write and read case?
  - Data change in each cycle?
  - Intelligent clients?
  - Pre-fetching?
  - User initiated broadcast?