Semester 1, 2005

Mobile and Wireless Computing
COSC2303/COSC2304

Kwong Yuen Lai

Mobile Data Management
SEER - The Coda file system

- Dealing with client disconnections

- Header

- Managing data consistency through broadcast invalidation

- Broadcast index

- Broadcast disk

- Push vs. Pull-based systems

- Applications of push-based systems

- Data broadcasting

- Limitations of mobile environments

---

Semester 1, 2005

Mobile and Wireless Computing
- Preserve battery life and reduce access costs
- Wireless interface switched off voluntarily
  Voluntary disconnection •

- Efficient energy management needed
  - Mobile devices powered by batteries to maximise portability
    Limited battery life •

- Poor accessibility
  - Resulting in long access delays, intermittent service, dropped and
    coverage

- Intermittent connectivity
  - Due to propagation losses, signal interference and limited network

Limitations of Mobile Environments
Efficient data access methods needed!

- Lack of physical space inside the device
- Small mobile devices generally much less powerful than desktop computers
- Limited computation power and storage
- Next generation applications will demand even more bandwidth
- Limited wireless spectrum
- Low bandwidth
  
  Semester 1, 2005
Wireless media suitable for broadcasting

- Transmission from server-side more economical than from client-side
- Limited battery life
- Base stations are powerful transmitters while mobile clients have
- One to many relationship between servers and clients
- Downlink channel capacity higher than uplink channel, due to:
- Takes advantage of the asymmetric property of wireless networks

Clients

Server

Data Broadcasting
Applications

- Consistency Information (Cache invalidation)
- Product catalogue/Price
- Advertisements
- Train/Pilot, flight timetables
- Local tourist information
- Stock prices
- Traffic information
- News push
- Television and Radio broadcast
<table>
<thead>
<tr>
<th>High scalability</th>
<th>Poor scalability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low energy requirement on clients</td>
<td>High energy requirement on clients</td>
</tr>
<tr>
<td>Share similar interests</td>
<td>System load is light</td>
</tr>
<tr>
<td>Good performance when clients</td>
<td>Good performance when</td>
</tr>
<tr>
<td>Information broadcast periodically</td>
<td>Information sent when requested</td>
</tr>
<tr>
<td>Broadcast/Multicast</td>
<td>Request-Reply</td>
</tr>
<tr>
<td>Server proactive</td>
<td>Server reactive</td>
</tr>
<tr>
<td><strong>Push</strong></td>
<td><strong>Pull</strong></td>
</tr>
</tbody>
</table>

**On-demand access**

- Traditional client-request/server-response is referred to as **Pull-based or Push vs. Pull**.
Access to broadcast disk is sequential (i.e., not random)

- Only frequently accessed data items (hot spots) are broadcast
- Disk in the air
- Communication channel acting as a form of storage
- Server periodically broadcasts data items

Broadcast Disk
reduce access latency

- Adjust the number of disks and the rotation speed of the disks to

- Supports variation in access patterns

  Multi-disk broadcast:  •

  - Only useful if access pattern is uniformly distributed

  - Simple to construct

Flat broadcast:  •

  - Frequently than less popular ones

Multi-disk broadcast - Popular data items broadcast more

Flat broadcast - All data items broadcast at the same frequency

  • Two types broadcast schedules:

  A broadcast schedule/program is needed.

  • How to organise data for broadcast?
In our example, \( ICM = 2 \times 3 = 6 \).

4. Calculate the least common multiple (LCM) of the relative frequencies.

3. Choose the relative broadcast frequency of each disk. For example,

\[
f_{\text{free}}(2) = 2.
\]

2. Partition the data items into \( n \) groups so each group contains data items with similar access probabilities. Each of these groups are referred to as a disk.

1. Order data items based on access probabilities.

Assume we want to create a \( n \) disk program:

[Cortejada et al., 2005] Creating a multi-disk broadcast program
where $C_j \mod m(\ell)$ is the $j$-th chunk of disk $\ell$. 

\begin{align*}
&\{ \\
&\{ \\
&\text{Broadcast} \quad C_j \mod m(\ell) \\
&\text{for } j \text{ between } 1 \text{ to } n \\
&(1 - IWD) \text{ for } i \text{ between } 0 \text{ to } n \}
\end{align*}

6. Create the broadcast program:

\[
7 = \frac{7}{9} = (2) \mod m(1) \\
3 = \frac{3}{9} = (3) \mod m, \text{ and } \frac{3}{9} = (1) \mod m
\]

In our example, $m(2) = 9$. Number of chunks for disk $\ell$.

5. Split each disk into $m(\ell)$ chunks where $m(\ell) = IWD + 1$.
\[
8 = \frac{1}{8} = (g)_{\text{free}}/\text{GTM} = (3)_{\text{in}}
\]
\[
7 = \frac{2}{8} = (g)_{\text{free}}/\text{GTM} = (2)_{\text{in}}
\]
\[
1 = \frac{8}{8} = (1)_{\text{free}}/\text{GTM} = (1)_{\text{in}}
\]

Therefore:

The lowest common multiple (LCM) of these frequencies is 8.

That is, frequency 1 is twice as frequent as disk 2 and eight times as frequent as disk 3.

Assume disk 1 is to be broadcast twice as frequently as disk 2 and eight times as disk 3.

\{\text{Disk 3} = \{3, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22\}\}

\{\text{Disk 2} = \{2, 3, 4, 5, 6, 7, 8, 9\}\}

\{\text{Disk 1} = \{1\}\}

Assume we have 25 data items to be broadcast over 3 disks.

\underline{Broadcast disk example}
Broadcast disk example
How does a client know when a data item will be broadcast?

- Achieve higher cache hit ratio

Cache replacement can be performed based on the broadcast schedule to

\[
L_8 = L \times 8 \times \frac{r}{Z} = (3)T -
\]

\[
L_7 = L \times 7 \times \frac{r}{Z} = (2)T -
\]

\[
L_5 = L \times 1 \times \frac{r}{Z} = (1)T -
\]

In our example:

where \( L \) is the time it takes to broadcast one data item

\[
L \times (\text{?}) \times \frac{z}{(\text{minor cycle})} = (? T)
\]

Fixed average access delay, \(? T\):
Data Index

Client in doze mode

"#" 

$"$

% 

\&

• Reduce tune-in time, therefore saves energy

• Retuned to as selective tuning

• Clients can switch into doze mode until the data they need is broadcast

broadcast schedule

Server broadcasts an index periodically to inform clients about the

Diagram:

Time

Data

Client in doze mode

Index

Broadcast Index

Semester 1, 2005

Mobile and Wireless Computing
Time-to-Live - Weak consistency can also lead to false invalidation

• Each client ← high overhead

Server validation - Server must maintain record of what is cached by

to bandwidth limitations

Polling - Good for few clients, not scalable in wireless environments due

• Polling - Time-to-Live

• Time-to-Live - Server validation

• Polling - Traditional techniques include:

How to maintain cache consistency?

• Clients

Caching reduce energy consumption and lowers access delay for mobile

Application of Broadcasting - Maintaining Cache Consistency
(d) Broadcast Invalidation

Server

Client 1

Client 2

Client 3

Items 2 and 4 updated

Items 2 and 4 updated

(a) Polling

Server

Client 1

Client 2

Client 3

Items 2 and 4 updated

Items 2 and 4 updated

(b) Server Validation

Server

Client 1

Client 2

Client 3

Items 2 and 4 updated

Items 2 and 4 updated

(c) Time-to-Live

Server

Client 1

Client 2

Client 3

Items 2 and 4 updated

Items 2 and 4 updated

(d) Broadcast Invalidation

Server

Client 1

Client 2

Client 3

Items 2 and 4 updated

Items 2 and 4 updated

(e) Polling

Server

Client 1

Client 2

Client 3

Items 2 and 4 updated

Items 2 and 4 updated

Still valid

Still valid

Still valid

Still valid

Still valid

Still valid

Still valid

Still valid

Still valid

Still valid

Still valid

Still valid

Still valid

Out-dated data objects are removed from the cache.

When client receives report, it compares its cache against the list.

Updated

Each report contains a list indicating which data items have been

Server periodically broadcasts invalidation reports

Most existing work are: Stateless, Synchronous, Invalidations

Update invalidation vs. propagation

Synchronous vs. Asynchronous broadcasts

Stateful vs. Stateless server

Choices

Use broadcast channel to communicate consistency information

Broadcast-based Cache Invalidation
Report covers the last \( T \) time units to support short client disconnections.

\[ \text{IR}^{r} = \{ A, 6 \}, \{ B, 8 \}, \{ C, 12 \} \]

\( \{ A, 6 \}, \{ B, 8 \}, \{ C, 12 \} \)

Updated data item C
Updated data item B
Updated data item A

last updated such that \( T \)

where \( ID \) is an identifier of data item \( \ell \) and \( \ell \) is the time at which item \( \ell \) was

\( \{ [\ell, ID] \} \equiv \text{IR}^{r} \)

The invalidation report at time \( T \) is:

\text{The Broadcast Timestamp (TS) method}
Reduced delay in answering queries •

Smaller in size than IR and broadcast more frequently •

UIRs contain details about updates since the last IR was broadcast •

Invalidation Reports (UIRs) every T/m time units

Apart from broadcasting IRs every T time units, also broadcast UIRs

The Update Invalidation Report (UIR) method
reduce the number and delaying the occurrence of cache misses

Goal of heuristics: Provide support for disconnected operations by operation altogether

A critical cache miss may prevent a client from continuing its

Cache misses during disconnection are expensive

consumption + access costs

Voluntary: Communication interface switcheth off to reduce energy

Involuntary: Due to poor network connectivity

Two types of disconnections:

Mobile clients often disconnected from the network

Client Disconnections

Semester I, 2005
Previous access pattern

- Automated: Performs automatically through analysis of user's
  - Manual: User makes hoarding selections manually

There are two types of hoarding algorithms:

- Capacity
- Problem

Which data items to hoard given limited client storage

and generally only concerned with the short term.
Similar to prefetching, however prefetching aims to reduce access delay

store them locally in the client's cache

Predicts the data items a client may need during disconnection and

Hoarding focuses on voluntary disconnections
• User has less control over what is hoarded

• Poor performance if user behaves very differently to the past

Disadvantages of automated hoarding:

• More likely to pick up critical data items that the user may not be aware of can be missed (e.g.,

  sexism files) –

  Tedious –

  Time consuming –

Disadvantages of manual hoarding:

• User has control over what is hoarded

• User's knowledge ensures only useful data items are hoarded

Advantages of automated hoarding:

• Can be performed in the background – less time consuming –

User friendly –
The board database allows users to explicitly indicate files that are of interest.

- Client's personal board database
  - Recent reference history using LRU
    - Hoarding is performed based on
      - Allows clients to continue to read and write to data in cache during disconnections
      - Location-transparent shared UNIX file system
      - Developed at Carnegie Mellon University

The Coda File System [Kistler et al.]
Correlator module - Calculate semantic distances between files.

Observer module - Records open and close of files.

Assumes users' access behaviour exhibits semantic locality.

I. Observe user behaviour

II. Fully automated boarding system

3. Cluster files into projects based on their semantic distances

4. Select projects for boarding

SEER [Kuehnle and Popk]
Geometric mean of $d_{i..}\cdot p_{r..}$ for $A$ and $B$:

The semantic distance between two files (or $B$) is then calculated as the

Stores the last $x$ distances between references (say $d_{i..}$) for each pair of files

For example, consider the sequence: $A^0, B^0, C^0, A^0, C^0, D^0$
are considered overlapped. Add each file into the opposite cluster.

3. If two files share less than \( k \) neighbors but at least \( k \) neighbors, they


are in the same cluster.

2. If two files share at least \( k \) neighbors, put them in the


same cluster.

1. Find the nearest neighbor of each file (based on semantic distance).

Algorithm:

Clustering of files through an agglomerative (bottom-up) clustering.

\[
\text{Distance} = \frac{4.128917}{4 \cdot 2 \times 5 \times 10^3} = 0.00003
\]

The semantic distance between files A and B is equal to:

\[
\{4, 5, 7, 10, 3\}
\]

The last five distances between references of files A and B are:

Assume \( x = 5 \) and,
<table>
<thead>
<tr>
<th>From: A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>To:</td>
<td>f</td>
<td>y</td>
<td>u</td>
<td>y</td>
<td>f</td>
<td>u</td>
</tr>
</tbody>
</table>

**Example:**
- 4: (A', B', C', D'), E, F, G
- 3: (A', B'), C', D', E, F, G
- 2: (A', B'), C', D', E, F, G
- 1: A', B', C', D', E, F, G
Adaptation to changing environment

Client mobility

Change in access patterns

Human-Computer Interface

Other issues:

Efficient use of communication channel, client cache space and client battery

Handing supports disconnected operations by anticipatory clients’ needs

Client disconnections can occur due to voluntary/involuntary

Maintaining cache consistency through broadcast invalidation

Trade-offs between push and pull methods

Characteristics of wireless environment put new requirements on data access

Summary
References


