Maintaining Coherency of Dynamic Data in Cooperating Repositories

Shetal Shah
Krithi Ramamritham
Prashant Shenoy
Dynamic Data

- rapid and unpredictable changes
  - stock prices, sensor data
- used in on-line decision making
- ideal world: every change delivered to every user.

coherency requirement \((c)\):
- E.g. Infosys stock price changes by $5

\[
\left| U(t) - S(t) \right| < c
\]
Design a dissemination system for dynamic data
-- meet users’ coherence requirements
-- minimize fidelity loss

**Metric:**

$$\text{Fidelity} = \frac{\text{length of time for which coherence req is met}}{\text{total length of observations}}$$

Dissemination systems for the web include Akamai, Dynamai
Cooperative Repository Architecture

- Source(s), repositories (and clients)
- Each repository specifies its coherency requirement
- Source pushes specific changes to selected repositories
- Repositories cooperate with
  - each other
  - the source
Dissemination Graph: Example

Data Set: p, q, r, s
degree of cooperation: 2

Source

A
p: 0.2, q: 0.3

B
r: 0.2

C
p: 0.4, r: 0.3
q: 0.3

D

Client
q: 0.4
3 Parts of the Problem

1. When should a repository disseminate updates?
   - data dissemination problem
2. What should be the logical interconnection between repositories?
   - layout problem
3. How much should a repository cooperate?
   - cooperation problem
Different users have different coherency req for the same data item.

Coherency requirement at a repository should be at least as stringent as that of the dependents.

Repositories disseminate only changes of interest.
Data dissemination -- must be done with care

\[ c^P = 0.3 \quad c^Q = 0.5 \]

Source \[\rightarrow\] Repository P \[\rightarrow\] Repository Q

1 \[\rightarrow\] 1 \[\rightarrow\] 1
1.2 \[\rightarrow\] 1 \[\rightarrow\] 1
1.4 \[\rightarrow\] 1.4 \[\rightarrow\] 1
1.5 \[\rightarrow\] 1.4 \[\rightarrow\] 1
1.7 \[\rightarrow\] 1.7 \[\rightarrow\] 1.7

should prevent missed updates!
Dissemination Algorithms

- Source Based  (Centralized)
- Repository Based  (Distributed)
For each data item, source maintains:
- unique coherency requirements of repositories
- the last update sent for that coherency

For every change,
- source finds the maximum coherency for which it must be disseminated
- tags the change with that coherency
- disseminates (changed data, tag)
Source Based Dissemination Algorithm

\[ c^P = 0.3 \quad c^Q = 0.5 \]

- Source → Repository P → Repository Q
  - 1 → 1 → 1
  - 1.2 → 1 → 1
  - 1.4 → 1.4 → 1
  - 1.5 → 1.5 → 1.5
  - 1.7 → 1.5 → 1.5
A repository $P$ sends changes of interest to the dependent $Q$ if

$$\left| x^P - x^Q \right| \geq c^Q$$
Data dissemination -- must be done with care

\[
\begin{align*}
\mathcal{P} & = 0.3 \\
\mathcal{Q} & = 0.5
\end{align*}
\]

\[
\text{Source} \xrightarrow{1} \text{Repository P} \xrightarrow{1} \text{Repository Q}
\]

\[
\begin{align*}
1 & \xrightarrow{1.2} 1 \\
1.4 & \xrightarrow{1.5} 1.4 \\
1.7 & \xrightarrow{1.7} 1.7 \\
\end{align*}
\]

should prevent missed updates!
A repository P sends changes of interest to the dependent Q if

\[ |x^P - x^Q| \geq c^Q - c^P \]
Repository Based Dissemination Algorithm

\[ c^P = 0.3 \]

\[ c^Q = 0.5 \]

Source → Repository P → Repository Q

1 → 1 → 1
1.2 → 1 → 1
1.4 → 1.4 → 1.4
1.5 (highlighted) → 1.4 → 1.4
1.7 → 1.7 → 1.7
Problem #2

1. When should a repository disseminate updates?  
   – data dissemination problem

2. What should be the logical interconnection between repositories?  
   – layout problem

3. How much should a repository cooperate?  
   - cooperation problem.
Fidelity offered by the layout network depends upon:
- Maximum end-to-end delay for disseminating updates.
- Overhead (load) of disseminating updates at each repository.

To achieve high fidelity, these delays should be minimized.
Constructing the Layout Network

Insert repositories one by one

- Check level by level starting from the source
  - Each level has a load controller.
  - The load controller tries to find data providers for the new repository (Q).
Repositories with low preference factor are considered as potential data providers.

The most preferred repository with a needed data item is made the provider of that data item.

The most preferred repository is made to provide the remaining data items.
Preference Factor

- **Resource Availability factor:**
  Can repository (P) be the provider for one more dependent?

- **Data Availability Factor:**
  \# data items that P can provide for the new repository Q.

- **Computational delay factor:**
  \# dependents P provides for.

- **Communication delay factor:**
  network delay between the 2 repositories.

\[
\text{delay} (P, Q) = \frac{\# \text{ data items P can serve for } Q}{\# \text{ data items P can serve for }}
\]
Performance Evaluation

- Single source, 100 repositories.
- Real time traces of various stocks
- 50-100 data items.
- Link delay: Computed by a heavy tailed function. Average link delay: 20-30 ms.
- Computation delay: 12.5 ms/client
- Rate of change of data-item: 1 change/sec
Performance Metrics

- Dissemination algorithms
  - Number of checks at source
  - Number of messages.

- Layout algorithm
  - Loss in fidelity
    - For different coherency requirements
    - For different degrees of cooperation
Dissemination Algorithms - number of checks by source.

Repository based algorithm requires *fewer* checks at source.
Dissemination Algorithms

- number of messages

Source based algorithm requires *less* messages
The less stringent the coherency requirement, the better the fidelity. T% of the data items have stringent coherency requirements.
Loss of fidelity vs. Degree of cooperation

too little / no cooperation
=>
loss of fidelity is high
Too much cooperation? can hurt

Under high degree of cooperation, computational delays dominate.

Under low degree of cooperation, network delays dominate.
1. When should a repository disseminate updates? 
   – data dissemination problem

2. What should be the logical interconnection between repositories? 
   – layout problem

3. How much should a repository cooperate? 
   - cooperation problem.
Controlled cooperation

Actual degree of cooperation

\[
= \frac{\text{average network delay}}{\text{average comp delay} \times \# \text{interested dependents}}
\]
Controlled cooperation is essential

Without controlled cooperation

With controlled cooperation
Cooperation is essential
-- to achieve high fidelity

But, need to control the cooperation offered
-- when delays are non-negligible

Selective Peer to Peer Dissemination of Streaming Data!