Roadmap

- Introduction
- Static data replication
- Dynamic data replication
- Experimental (simulation) results
- Summary
Data Replication

- The problem: given a data item and its popularity, determine *how many* replicas to put
- For writable data, *where* to put
- Destination: node(s) in a distributed environment
- Replicas are identical copies of the original data
Quality-Aware Replication

- Replicas are of different “quality”
- Destination: point(s) in a metric *quality space*
- Costs of transformation among different qualities are very high
- Applications
  - Multimedia
  - Materialized view
  - Biological structure
- Good news: read-only
- Bad news: too much storage needed
Delivery of Multimedia Data

- Quality (QoS) critical
  - Temporal/spatial resolution
  - Color
  - Format
- Varieties of user quality requirements
  - Determined by user preference and resource availability
  - Large number of quality combinations
- Adaptation techniques to satisfy quality needs
  - Dynamic adaptation: online transcoding
  - Static adaptation: retrieve precoded replica from disk
Dynamic adaptation

- Transcoding is very expensive in terms of CPU cost
- Online transcoding is not feasible in most cases
- Situation may improve in the future
- Layered coding
  - Not standardized yet.
  - Less popular than people expected
Static adaptation

• Little CPU cost
• Choice of many commercial service providers
• What about storage cost?
  • On the order of total number of quality points
  • Ignored in previous research assuming
    • Very few quality profiles
    • Storage is dirt cheap
• Excessively high for service providers

\[ O(n^d) ! \]

Table 2: Total relative storage in a 3D space.

<table>
<thead>
<tr>
<th>n</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage</td>
<td>20.23</td>
<td>117.7</td>
<td>354.8</td>
<td>755.9</td>
<td>1496.5</td>
</tr>
</tbody>
</table>
The fixed-storage replica selection (FSRS) Problem

- An optimization: get the highest utility given the popularity ($f_k$), storage cost ($s_k$) of all quality points under total storage $S$.
  - $u(j,k)$: the utility when a request on quality $j$ is served by replica of quality $k$
- Utility is given as a function of distance in quality space.
  - Requests served by the closest replica

\[
\text{maximize} \quad \sum_{j \in J} \sum_{k \in J} f_j u(j, k) Y_{jk} \\
\text{subject to} \quad \sum_{k \in J} X_k s_k \leq S, \\
\sum_{k \in J} Y_{jk} = 1, \\
Y_{jk} \leq X_k, \\
Y_{jk} \in \{0, 1\}, \\
X_k \in \{0, 1\}
\]
Roadmap

- Introduction
- Static data replication
- Dynamic data replication
- Experimental (simulation) results
- Summary
The FSRS Algorithms (I)

• Problem is NP–hard: a variation of the $k$–means problem
• We propose a heuristic algorithm named \textit{Greedy}
  - Aggresively selects replicas based on the ratio of marginal utility gain ($\Delta u$) to cost ($s_k$)

\begin{verbatim}
selected replica set $P := \emptyset$
available storage $s' := S$
while $s' > 0$
    add the quality point that yields the largest $\Delta u/s_k$ value to $P$
    decrease $s'$ by $s_k$
\end{verbatim}

\textbf{return} $P$

• Time complexity: $O(m^2 I)$ where $I$ is the # of replicas selected and $m$ the total # of possible replicas
An illustration: *Greedy*
The FSRS Algorithms (II)

- *Greedy* could pick some bad replicas, especially the earlier selections
- Remedy: remove those bad choices and re-select
- The *Iterative Greedy* algorithm:

  \[
  P \leftarrow \text{a solution given by } Greedy
  
  \text{while there exists solution } P' \text{ s.t. } U(P') > U(P)
  
  \quad \text{do } P \leftarrow P'
  
  \text{return } P
  \]

- Time complexity: same as *Greedy* with a larger coefficient
An illustration: *Iterative Greedy*
Handling multiple media objects

- There are $V (V > 1)$ media objects in the database, each with its own quality space and FSRS solution.
- However, the storage constraint $S$ is global.
- Both Greedy and Iterative Greedy can be easily extended to solve FSRS for multiple media objects.
- The trick: view the $V$ physical media objects as replicas of a virtual object.
- Model the difference in the content of the $V$ objects as values in a new quality dimension.
- Time complexity: $O(IV^2m^2)$, can be reduced to $O(IVm^2)$ with some tweaks.
Roadmap

- Introduction
- Static data replication
- Dynamic data replication
- Experimental (simulation) results
- Summary
Dynamic replication

• Popularity $f$ of replicas could change over time
• We only consider the situation where popularity of all replicas of a media object changes together
  - Reasonable assumption in many systems
  - Problem becomes competition for storage among media objects
  - Study of the more general case is underway
• Desirable dynamic replication algorithms:
  - Find solutions as optimal as those by static FSRS algorithms
  - Fast enough to make online decisions
• Naïve solution: run $Greedy$ every time a change of $f$ occurs
Replication Roadmap (RR)

- Consider the order replicas are selected by Greedy – follow a predefined path (RR) for each media object
- RRs are all convex
- Exchanges of storage may happen between two media objects, triggered by the increase/decrease of $f$
  - The one that becomes more popular takes storage from the least popular one
  - The one that becomes less popular gives up storage to the most popular one
  - It is efficient to make exchanges at the frontiers of the RRs, no need to look inside
Replication Roadmap (continued)

- Storage exchanges, example:

![Graph showing utility rate and storage occupation for media A and B.]

Media A should take storage from media B as the slope of its current segment in RR is greater than that of B’s.
Dynamic FSRS algorithm

- Based on the RR idea
- Proved performance: results given are as optimal as those chosen by Greedy
- Preprocess phase:
  - Build the RR
- Online phase:
  - Performing exchanges till total utility converges
  - Time complexity: $O(I \log V)$ where $I$: # of storage exchanges occurs and $V$ is the # of media objects

```java
storage ← available storage
k ← 0, j ← V - 1
while k ≤ 0
  do $r_0 ← flist[k]$
  do $victims ← ∅$
  while storage < size of replica $r_0$
    do $r_1 ← blist[j]$
      if $r_0$ and $r_1$ belong to the same video
        continue
      if utility density of $r_1 >$ utility density of $r_0$
        $k ← k + 1$
        rollback $blist$ to its status on line 6
        break
      else append $r_1$ to $victims$
        update and sort $blist$
    if storage ≥ size of replica $r_0$
      EXCHANGE $(r_0, victims)$
      update and sort both $flist$ and $blist$
```
Roadmap

- Introduction
- Static data replication
- Dynamic data replication
- Experimental (simulation) results
- Summary
Effectiveness of algorithms

- For comparison:
  - The optimal solution (by CPLEX)
  - Random selections
  - Local popularity-based
Efficiency of algorithms

- CPLEX < Iterative Greedy < Greedy < Random < Local
- Results on a P4 2.4 GHz CPU:
Dynamic replication

- Randomly generated changes of $f$
- Compare with Greedy
- Results with (almost) the same optimality as Greedy
- Reason: small number of storage exchanges
Summary

- Storage cost in static adaptation prohibits replication of all qualities
- Need to optimize toward the highest utility given storage constraints
- Two heuristics are proposed for static replication that gives near-optimal choices
- Fast online algorithm for one dynamic replication problem
- Unsolved puzzles:
  - General case of dynamic replication
  - Is there a bound for the performance of \textit{Greedy}?
    - Conjecture: \textit{Greedy} is $4/3$–competitive!
Acknowledgements

- Jingfeng Yan (Purdue)
- Sunil Prabhakar (Purdue)
And
- Leming Qu (Boise State)
- Steve Luke
- Shan Lei (Google)
- Hong Wan (Purdue)
Storage for replication

- Empirical formula to calculate storage after transcoding to a lower quality in one dimension:
  \[ F = F_0 (1 - R_0^{\frac{1}{\beta}}) \]

- Sum of all replicas when there are \( n \) qualities

\[
\sum_{i=0}^{n} F_0 (1 - R_i^{\frac{1}{\beta}}) = F_0 \left( n - \sum_{i=0}^{n} \left( \frac{i}{n} \right)^{\frac{1}{\beta}} \right)
\]

\[
\approx F_0 \left( n - \int_{0}^{1} \left( \frac{x}{n} \right)^{\frac{1}{\beta}} \right)
\]

\[
= F_0 \left( n - \frac{n^{\beta}}{\beta + 1} \right)
\]

\[
= F_0 \frac{n}{\beta + 1} = F_0 O(n).
\]

- Three dimensions: \( F = \alpha F_0 (1 - R_A^{\frac{1}{\beta}})(1 - R_B^{\frac{1}{\beta}})(1 - R_C^{\frac{1}{\beta}}) \), total storage is thus \( O(n^3) \)
- For \( d \) dimensions, \( O(n^d) \)
More experimental results

Selection of replicas by *Greedy*, 21x21 2-D quality space with larger number representing lower quality (i.e., point (20,20) is of the lowest quality), $V = 30$

![Minimum penalty](image1)
![Manhattan distance](image2)

Same input