Distributed Meta-data Servers: Architecture and Design

Sarah Sharafkandi
David H.C. Du

DISC
Outline

- Meta-Data Server (MDS) functions
- Why a distributed and global Architecture?
- Problem description
- Current distributed solutions
- Proposed architecture
- Challenges for the proposed architecture
Metadata server responsibilities

- Map a Globally Unique Identifier (GUID) to each object based on its local directory and user
- Given the GUID of each object, find the nearby copy
- Maintain object deletion, migration, versioning, locking
- Access permissions (at the MDS level)
- Answer queries from search applications
Why a distributed and global infrastructure?

- **Scalability:**
  - Number of *objects* in enterprises, is growing fast
  - A centralized solution simply does not scale well

- **Accessibility:**
  - accessible by clients from anywhere
  - several copies from an object: Global infrastructure

- **Policy based support:**
  - Redundant copies, backup policies, fault tolerance policies, QoS requirement, Security policy
Problem description

- Design a **scalable** and **efficient** Distributed MDS (DMDS) structure which provides lookup services in an **extremely large scale** and preserves:
  - Locality (Finding a nearby copy)
  - Object Replication, migration
  - Data consistency
  - Load balancing
  - Versioning
Current distributed solutions

- Distributed Hash Table:
  - Application: Peer to peer networks (P2P) (Chord, CAN) where the structure is very dynamic (nodes join and leave frequently)
  - Design: Both object id and node id are randomly hashed to the same id space
  - Advantage: scalable (but not at the scale of global file system)
  - Disadvantage:
    1. Don’t preserve the locality
    2. Usually load balancing is not considered. Hot spots can occur.
Current distributed solutions (cont.)

- **OceanStore**
  - **Application:** Global file system:
  - **Design:**
    - For look up uses P2P solutions: Tapestry
  - **Advantage:** Versioning and update support for objects
  - **Disadvantage:**
    - High cost of publishing the pointer for each object
    - The locality is preserved with high cost
How about Internet?

- Domain Name System:
  - Application: Internet
  - Design: A Hierarchical structure
    - Resolves names to the IP addresses
  - Advantage: Simple
  - Disadvantage: File naming depends on the location of the file
    - if the file migrates the address is not valid anymore
## DMDS Versus P2P

<table>
<thead>
<tr>
<th>P2P</th>
<th>DMDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Dynamic: Nodes join and leave the network frequently</td>
<td>1. More Rigid</td>
</tr>
<tr>
<td>2. No Data Consistency</td>
<td>2. Data Consistency</td>
</tr>
<tr>
<td>3. Locality may not be maintained</td>
<td>3. Locality is maintained</td>
</tr>
<tr>
<td>4. There is no knowledge about user rolls and groups</td>
<td>4. Knowledge about user rolls and groups is available</td>
</tr>
</tbody>
</table>
Proposed Architecture for DMDS

Topology: A two layer topology

- First layer: Hierarchical structure in low levels based on:
  - Proximity
  - User groups
- Second layer: An overlay structure
Proposed topology

- Two layer Architecture:
  - Layer 1: hierarchical structure,
  - Layer 2: Overlay structure
Two layer topology (cont.)

- **Layer 1:** Hierarchical structure based on:
  - Proximity:
    - Geographical location
  - User groups:
    - Users are members of groups based on their file requests and their role in the company
    - Each user may be in multiple groups and each group may be in different sites
    - Roles and groups may change dynamically according to the changes in user’s demand

- **Layer 2:** An overlay infrastructure
  - p2p techniques for search can be used
Advantages of two layer design:

- Scalable
- Flexible structure
- Locality awareness at layer 1
- Efficient search by user grouping
- Avoid long search paths using the overlay structure
  - The popular objects can be replicated at overlay layer
Challenges:

- What is the optimal number of levels at layer 1?
- How many overlay nodes are needed?
- How to preserve data consistency with low cost?
- How to maintain the security?
- How to provide a good balance between proximity and user groups?
- How to migrate data based on the object popularity?
Thank you!