Replica Management

• Content placement
  – Finding the best servers for placing content

• Update Propagation
  – Propagating updated content to the relevant replica servers

Three Types of Replicas

• **Permanent replicas** are the initial set of replicas that constitute a distributed data store
  – Created by the data store owner
  – The data store can be replicated across a cluster of servers at a single location or across a number of geographically dispersed servers (called mirror sites).
  – The number of permanent replicas is small

• **Server-initiated replicas** are replicas created in order to enhance the performance of the system
  – Created at the request of the data store owner, often placed on servers maintained by others
  – Server-initiated replicas are placed close to large concentrations of clients
  – Dynamic replication algorithms are used to decide where to place content

• **Client-initiated replicas** are temporary copies created by clients to improve their access times to the data (commonly known as client caches)
  – Examples: Web browser caches, Web proxy caches
Dynamic Replication

• The decisions about where to place server-initiated replicas and when to create new ones/destroy existing ones/migrate existing ones are made dynamically by the system

• Requirements
  – A network of servers willing to host replicas
  – Collecting usage pattern of data items

Dynamic Replication: Web Hosting Service Example

• Each server keeps track of access counts per file, aggregated by considering the server closest to the requesting clients
  – If C1 and C2 share the same closest server P, all access requests for F at Q from C1 and C2 are registered at Q as a single access count cntQ(P, F)

• Replication rules
  – Number of accesses for F at Q below threshold D \( \rightarrow \) remove F from Q
  – Number of accesses for F at Q exceeds threshold R \( \rightarrow \) replicate F on another server
  – Number of accesses for F at Q between D and R \( \rightarrow \) migrate F to server P if \( \text{cnt}_Q(P, F) \) exceeds half of total requests for F at Q
Update Propagation

• What to propagate?
  – Propagate an invalidation
    • Uses little network bandwidth: only needs to specify which data is no longer valid
    • Works best when read-to-write ratio is small
  – Propagate the modified data
    • Useful when read-to-write ratio is high
  – Propagate the update operation and the parameter values needed by the operation
    • Updates can be propagated at minimal bandwidth costs
    • More processing power is required by each replica (to carry out the update operation)

Pull vs. Push

• Push-based approach (server initiated)
  – Updates are propagated to other replicas when they occur
  – Often used between permanent and server-initiated replicas, also used to push updates to client caches
  – Efficient when read-to-update ratio is high (e.g., large shared caches)
• Pull-based approach (client initiated)
  – A server or client requests another server to send it any updates it has at that moment
  – Often used by client caches
  – Efficient when read-to-update ratio is low (e.g., nonshared client caches)
Comparison between Pull and Push

<table>
<thead>
<tr>
<th>Issue</th>
<th>Push-based</th>
<th>Pull-based</th>
</tr>
</thead>
<tbody>
<tr>
<td>State at server</td>
<td>List of client replicas and caches</td>
<td>None</td>
</tr>
<tr>
<td>Messages sent</td>
<td>Update (and possibly fetch update later)</td>
<td>Poll and update</td>
</tr>
<tr>
<td>Response time at client</td>
<td>Immediate (or fetch-update time)</td>
<td>Fetch-update time</td>
</tr>
</tbody>
</table>

A comparison between push-based and pull-based approaches in the case of multiple-client, single-server systems.

A Hybrid Approach: Leases

- We can dynamically switch between push and pull using leases - a contract in which the server promises to push updates to the client until the lease expires.
  - When a client is interested in receiving updates from the server, it requests a lease
  - When a lease expires, the client must poll the server for updates, or it can renew its lease
Adaptive Leases

• The length of a lease can be dynamically adapted depending on different criteria:
  – **Age-based leases**: A data item that hasn’t changed for a long time will receive a long-lasting lease
  – **Renewal-frequency based leases**: Clients that often request to have their cached copy updated will receive longer leases than those that do so infrequently
  – **State-based leases**: the more loaded a server is, the shorter the lease length becomes
    • The server dynamically switches to a more stateless mode of operation when it becomes overloaded