Flexible Wide Area Consistency Management

Sai Susarla
The Problem

- Wide area services are not alike
  - Different consistency & availability requirements

- But many common issues & mechanisms
  - Concurrency control, update propagation, conflict detection & resolution, replication management

- Ideal: Reuse mechanisms from existing systems

- Problem: Each system designs monolithic solution
  - Suited to its own particular needs
  - Hard to isolate individual mechanisms and reuse differently

- Managing consistency of wide–area replicated data requires significant development effort
Thesis

"It is both viable and beneficial to provide a consistency management system that can be customized to satisfy the data coherence needs of a variety of applications in the context of a wide-area data storage and caching infrastructure"

- Benefits of such a system
  - Offloads much consistency management complexity from applications.
  - Efficient due to customization
Wide Area Consistency Management

- Many common policy issues & design choices
  - When is access allowed to a replica? (optimistic vs. pessimistic)
  - Where are updates issued? (single/multiple masters)
  - How are updates propagated? (push vs. pull)
  - What triggers update transfers? (timer, staleness, eager, lazy)

- Many apps hardwire a combination of above choices
- Hard to isolate & reuse in other combinations
Exploit Commonality

- Optimistic
  - last-writer-wins, append-only
  - Single-master
    - Pull updates
    - Push updates
  - Multi-master
    - Pull updates
    - Push updates

- System files, static web content
- Scoreboard, Active dir. schema
- config files, multimedia streaming
- Bayou, calendar, Coda,
sensor logging, mobile file access
- Active dir. Data, Chat,
  whiteboard

Approach:
- Implement common mechanisms in app-independent manner
- Provide hooks to let applications customize their behavior
Contributions

- Identify policies, mechanisms that together satisfy a variety of data coherence needs
- Implement them in a wide area framework, retaining
  - Reusability: by application-independence of mechanisms
  - Efficiency: by letting applications customize system behavior
- Evaluate its effectiveness and scalability for a variety of applications
Evaluation Platform

- Khazana: configurable caching infrastructure
  - File-like data abstraction, page-grained consistency
  - Performs caching, concurrency control
  - Asynchronous update notifications

- Application controls per-file behavior
  - Consistency policy (last-writer-wins, append-only, or strict)
  - Control over per-replica quality
  - Optimistic vs. pessimistic access
  - Push vs. pull updates, single vs. multiple masters

- Per-file dynamic cache hierarchy
Replicated Objects in Khazana

Object O1: multi-master, pull-updates optimistic consistency
Object O2: single-master, push-updates strict consistency

P: Primary replica
Sn: Khazana server

Parent-child relationship between object replicas
Client accesses replica
Replicated Objects in Khazana

Object O1: multi-master, pull-updates optimistic consistency
Object O2: single-master, push-updates strict consistency

S1

S2

S3

S4

S5

P: Primary replica
Sn: Khazana server
Broken link

O2 copy inaccessible
Link down

Client accesses replica

Parent–child relationship between object replicas
Replicated Objects in Khazana

Object O1: multi-master, pull-updates optimistic consistency
Object O2: single-master, push-updates strict consistency

S1
S2
S3
S4
S5

P: Primary replica
Sn: Khazana server
→ Broken link

Parent–child relationship between object replicas
→ Client accesses replica
Experimental Setup

- Prototype Khazana & following applications
  - DataStations: distributed file store that supports multiple file sharing patterns
  - Chat room: concurrent appends to a transcript file
  - Scoreboard: broadcasts game state to registered listeners
  - Directory Service: Hash table implemented in a shared file

- Utah Network Testbed for evaluation
A Khazana–based Chat Room

Chat transcript: multi–master, push updates, optimistic append–only consistency

Update propagation path

Sample Chat client code

callback(handle, newdata) {
    display(newdata);
}

main() {
    handle = kh_open(kid, "a+");
    kh_snoop(handle, callback);
    while (! done) {
        read(&newdata);
        display(newdata);
        kh_write(handle, newdata);
    }
    kh_close(handle);
}
Evaluation

Evaluate

- Scalability, resilience to wide-area conditions (DataStations)
- Performance
  - Microbenchmarks (latency, bandwidth utilization)
  - Per-application macrobenchmarks

Demonstrate

- Application control over system mechanisms
- Scalable real-time data exchange (Chat room)
- Control over per-replica data quality (Scoreboard)

Compare performance

- Chat room with IRC
- Directory service with MS Active directory
Related Work

- Support for multiple consistency policies
  - Munin, WebFS, Fluid Replication
  - TACT
- Wide area issues in consistency management
  - Coda, Ficus, Bayou
- Applications with specific consistency semantics
  - Active Directory, Thor
- Peer-to-peer data sharing systems
  - Napster, PAST, Farsite, Freenet
Open Issues

- Optimal replica placement
- Security
- Efficient replica topologies reflecting Internet routing
Timeline

Sep
- Khazana on WAN, last-writer, push updates, DataStations
- Defend proposal

Oct
- Other policies, push, multi-master (req. file versioning)
- Experiments 1 & 2

Nov
- Op. timeouts, node failures
- Experiment 3

Dec
- Async updates, update triggers (timer, staleness ...)
- Scoreboard & chat
- Experiments 5 and 6

Jan
- Directory service
- Experiment 4

Feb
- Start writing thesis

April/May
- Finish thesis and defend.
Conclusion

• Managing consistency of replicated data is hard
  – Many mechanisms available, but hard to reuse efficiently
  – Current approach: reinvent the wheel

• Proposed a solution: generic consistency framework
  – Offloads this complexity from applications
  – Reusable, customizable, scalable

• Contributions
  – Identifying policies, mechanisms that serve a variety of coherence needs
  – Designing a wide area framework to leverage them efficiently
  – Evaluating its effectiveness & scalability for a variety of useful applications