Cluster consistency for multipeer collaborative applications

Anders Gidenstam, Boris Koldehofe, Marina Papatriantafilou and Philippas Tsigas
Chalmers University of Technology
Distributed Computing and Systems group,
Department of Computer Science and Engineering
Outline

- Introduction
  - Collaborative Environments
  - Group communication

- Causal Cluster Consistency
  - Achieving optimistic causal order
  - Managing senders

- Future Work
Collaborative Environments

- Possible applications with physically distributed “users”:
  - Conferencing, CVEs
  - Simulation, Training, Entertainment
  - Administration of distributed (e.g. telecom, transport) systems
  - Ad-hoc networks

- Trade-off
  - Overhead v.s. Consistency

- Users join the world, create/read/modify/delete objects, and leave with a response.

-(self-)modify mobile objects
Defining the problem

- Multicast for a large group
  - Event delivery in causal order
  - Scalability important

- Opportunities
  - Delivery with high probability is enough
  - Limited per-user domain of interest
    - Nobody is interested in everything at once
  - Events have lifetimes/deadlines
  - Often more observers than updaters
Scalable group communication with ordering guarantees

- Clusters - Disjoint subsets of objects
  - Interested processes join
  - Gossip-based communication
  - Readers – everyone
  - Updaters
    - Only a limited number at a time
    - Core of the cluster
Causally ordered delivery

- Vector timestamps
  - For each event in cluster
  - \#simultaneous updaters limited => limited number of vector entries in timestamps
  - Can detect missing dependencies
    - Recovery may be attempted
      - Ask the source
      - Ask k peers
  - Deliver in causal order
    - Skip events not recovered in time
Implementation: A Layered approach

- Implemented in C++
- Causal layer
  - Causal delivery
  - Recovery
- Dissemination layer
  - Gossip protocol
  - Reader membership
- Point-2-point communication layer
  - TCP
    - Concurrent connections
  - UDP

Application

Ordered delivery:
Cluster Consistency

Dissemination:
PrCast

Network transport service

Send/receive

Ordered, predictably reliable
disseminate/receive

disseminate/receive

recover
Managing the Core

- At most $n$ members/coordinators at any time
  - One unique vector entry each
  - Coordinators join and leave
  - Coordinators might fail
    - Stop failures
    - Communication failures
Cluster Management Algorithm

- Inspired by DHT
  - Clock entry Ids form a cycle
  - Each process manage the entries immediately before it.
- Contact any coordinator to join
  - Notify successor if given an entry
  - Notify all about the new coord.
- Failure detection
  - Heartbeats
    - Send to $2k + 1$ closest successors
    - Receive from $2k + 1$ closest predecessors
    - If $< k + 1$ received, stop
Experiments: Scalability

Throughput, under low communication failures and event loss

- 5 Updater Gossip/TCP
- 25 Updater Gossip/TCP
- Full Updater Gossip/TCP

Messages per second vs. Processes
Experiments: Scalability

Latency, under low communication failures and event loss

- 5 Updater Gossip/TCP
- 25 Updater Gossip/TCP
- Full Updater Gossip/TCP

Delay in ms vs Processes
Experiments: Reliability

Event loss

Percentage of known events lost

Probability to create a new event

- Causal layer with R4 recovery
- Causal layer with R1 recovery
- Causal layer without recovery
- No Causal layer
Discussion

Summary
- Optimistic causal multicast
  - Based on gossip dissemination
  - Analysis of buffering for event recovery
- Decentralized cluster management algorithm
  - Fault-tolerant

Towards lightweight solutions
- Reliable multicast -> gossip dissemination
- Causal order -> optimistic causal order
Future Work

- Causal Cluster Consistency
  - Application case study
    - E.g. distributed monitoring
  - Mobile and/or self-modifying objects
  - Self-stabilizing fault-tolerant group communication
  - Plausible clocks for ordering
    - Alternative to the cluster vector clock
    - No strict need to limit #updaters
    - Event recovery not (easily?) possible
Questions?

- **Contact Information:**
  - **Address:**
    Anders Gidenstam  
    Computing Science  
    Chalmers University of Technology
  - **Email:**
    andersg @ cs.chalmers.se
  - **Web:**
    http://www.cs.chalmers.se/~dcs  
    http://www.cs.chalmers.se/~andersg

- **Technical reports**
  - TR 2005-09 “Causal Cluster Consistency”
  - TR 2005-10 “Dynamic and fault-tolerant cluster management”