Rethinking Consistency Management in Real-time Collaborative Editing Systems

Jon A Preston

Committee
Dr. Sushil K Prasad
Dr. Xiaolin Hu
Dr. Melody Moore Jackson
Dr. Rajshekhar Sunderraman
Contents

- Introduction
- Motivation
- Problem Statement and Goals
- Contributions
- Architecture
- Relaxed Consistency
- Integrating OT
- Peer-to-Peer Document Management
- Hierarchical Reduction
- Prototypes
- Conclusions
Contents

- Introduction
- Motivation
- Problem Statement and Goals
- Contributions
- Architecture
- Relaxed Consistency
- Integrating OT
- Peer-to-Peer Document Management
- Hierarchical Reduction
- Prototypes
- Conclusions
Introduction

- Computer Supported Collaborative Work (CSCW)
  - Enhancing communication and collaboration
- Real-time Collaborative Editing Systems (RTCES)
  - Share a common document among many users
  - Provide a high level of editability (locally)
  - Maintain concurrency control (consistency)
  - CCI and OT
Sun proposed the most widely adopted standard for consistency maintenance in RTCES

- **Convergence**: when the same set of operations have been executed at all local copies, then the local copies will all have the same content/state.
- **Causality-preservation**: for operations $O_1$ and $O_2$, if $O_1 \rightarrow O_2$ then $O_1$ precedes (is executed before) $O_2$ at all local copies.
- **Intention-preservation**: executing an operation $O$ does not change the effects of executing operations $O_1\ldots O_n$ where $O_1\ldots O_n$ are independent of $O$. Further, the effects of executing $O$ at any local copy is the same as the intention of $O$ (i.e. the intention is the same across all copies).
Operational Transformation (OT)

- Attempts to achieve CCI
- Causality preservation via state vectors
  - Buffers used to delay enacting non-causally ready operations
- Convergence
  - OT integration (distribution)
  - OT transformation

\[
\text{TP}_1 \quad op_1 \circ T(op_2, op_1) \equiv op_2 \circ T(op_1, op_2)
\]

\[
\text{TP}_2 \quad T(op_3, op_1 \circ T(op_2, op_1)) = T(op_3, op_2 \circ T(op_1, op_2))
\]
Contents

- Introduction
- Motivation
- Problem Statement and Goals
- Contributions
- Architecture
- Relaxed Consistency
- Integrating OT
- Peer-to-Peer Document Management
- Hierarchical Reduction
- Prototypes
- Conclusions
Motivation

- Productivity tools increasingly focus on supporting collaboration among multiple users
  - Groove, Google Docs, Sharepoint, Wikis, etc.
- Software engineering
  - Collaboration among developers
- Document Editing
  - Co-located or distributed
- CAD Systems
  - Non-linear, spatial and group-based objects
Contents

- Introduction
- Motivation
- Problem Statement and Goals
- Contributions
- Architecture
- Relaxed Consistency
- Integrating OT
- Peer-to-Peer Document Management
- Hierarchical Reduction
- Prototypes
- Conclusions
Problem Statement

**Centralized consistency**
- Single point of failure
- Bottleneck at server
- Locking reduces concurrent access

**Replica based consistency**
- High level of communication cost (broadcast all operations to all users)
- High level of storage cost (retain all operations in history buffer)

![Diagram](image)

Document state managed centrally

Replicas of the document at each client
Goals

Develop an open systems RTCES architecture
- Extend legacy server/repository technologies
- Support clients’ preferred editing technologies

Utilize dynamic locking
- Support concurrency management
- Reduce communication and computation costs

Apply a P2P approach
- Increase reliability
- Avoid performance bottlenecks at a single server

Leverage document’s structure
- Reduce the size of the history buffers needed to manage operations
- Better enable intention preservation

Develop prototype implementations
- Validate our theoretical approaches
Contents

- Introduction
- Motivation
- Problem Statement and Goals
- Contributions
- Architecture
- Relaxed Consistency
- Integrating OT
- Peer-to-Peer Document Management
- Hierarchical Reduction
- Prototypes
- Conclusions
Contributions

- An open systems architecture
- Theoretical algorithms and data structures to support dynamic locking
- Integration of OT best practices and improved CCI
- Prototype client and server technologies
Contents

- Introduction
- Motivation
- Problem Statement and Goals
- Contributions
- **Architecture**
- Relaxed Consistency
- Integrating OT
- Peer-to-Peer Document Management
- Hierarchical Reduction
- Prototypes
- Conclusions
**Architecture**

- **Feature-rich client editing systems**
  - OpenOffice
  - Microsoft Office

- **Integrated development environments (IDEs)**
  - Borland's JBuilder
  - Microsoft Visual Studio
  - Sun's NetBeans

- **CMS and repositories**
  - RCS
  - VSS
  - CVS
  - SharePoint

---

*Connect these client and server technologies via Web services*
Components

Client Listeners and Lock Managers connect client and server technologies (via Web services)
Layering Additional Functionality

RTCES Web Service API

Lock Proxy

Existing Configuration Management System API
Existing Web API
Operating System Calls

VSS RCS CVS Etc. Wiki Frontpage SharePoint Etc. File System
Translating Client Events

- Avoid mapping events in each client technology to every other client technology \((n^2)\)
- Use meta language and map to/from it
Lock Proxy Simulated

- Modeled client and server components
- Utilized DEVS Java for discrete event simulation
- Access to different sections possible
- But not true concurrency
Measuring Communication Costs

- P2P RTCES using DirectX
- Simulated 100mbps and 33.6kbps with 2% packet loss
- Reasonable times for common events
Contents

- Introduction
- Motivation
- Problem Statement and Goals
- Contributions
- Architecture
- Relaxed Consistency
- Integrating OT
- Peer-to-Peer Document Management
- Hierarchical Reduction
- Prototypes
- Conclusions
Relaxed Consistency

- OT is costly because it communicates all operations to all users.
- Caching operations and dynamically managing locking reduces communication costs.
- Allow the document state to be “correct enough”

\[ D_{total} = \bigcup_{i=1}^{n} (D_i + \Delta D_i) \]
Document Tree

- Manage the document as sections
- Map the document to a tree
- Use this tree structure to localize locks and cache changes
Each node maintains coloring
- White – no owner
- Black – owned by a user
- Grey – contention among users in sub-trees

**USER ENTER (REQUEST LOCK)**
- Traverse from root down examining color
- Demote if necessary

**USER EXIT (RELEASE LOCK)**
- Traverse from root down examining color
- Promote if possible
User Enters (Lock Requested)
User Exits (Release Lock)
Efficiency and Analysis

- These algorithms run in $O(h)$ where $h$ is the height of the document tree
- Validated via simulation
  - Communication cost dramatically reduced by as much as 95%
  - But lock requests can fail (we exclusively lock at leaves) as much as 75% of the time
- Need a way to avoid lock failure
  - Combine locking and OT (apply OT at leaves)
Contents

- Introduction
- Motivation
- Problem Statement and Goals
- Contributions
- Architecture
- Relaxed Consistency
- Integrating OT
- Peer-to-Peer Document Management
- Hierarchical Reduction
- Prototypes
- Conclusions
Integrating OT

- Use document tree and locking
  - Continue to localize changes
  - Promotion and demotion when possible
- Apply OT at leaves
  - All lock requests succeed
  - Increase communication complexity
- Decrease communication costs while still allowing complete concurrent access
Message Types

- **Document Check-out (CO)** – the client checks out and becomes a reader
- **Document Check-in (CI)** – the client exits the document and releases it
- **Lock Request (LK)** – the client wants to write to a section of the document
- **Unlock (ULK)** – the client has left the section and is no longer a writer in it
- **Promotion (P)** – informs the user that his ownership has been increased
- **Demotion (D)** – informs the user that his ownership has been decreased

- **OT Added (OTA)** – signals a user within a section that another user has been added to the section and future changes must be sent to this new user
- **OT Deleted (OTD)** – signals a user within a section that a user has left the section and no longer needs to have changes sent to him
- **OT Join (OTJ)** – tells the user requesting a lock that he has been granted write access to a section that is already using OT; this message contains a list of the existing users within the section so that the new user can send future changes to these users
- **OT Modify (OTM)** – this message tells a client that the section has been modified and a local OT must be performed based upon the operation being communicated
Results

Since all messages are broadcast to all users other than the originating user in a pure OT system, we define the number of messages generated in a pure OT system as

\[ M_{\text{PureOT}} = (n-1)W \]

where \( n \) is the number of users and \( W \) is the number of write requests (the number of times users modified the document).

The relative message overhead, \( Mo \), of our dynamic lock OT system is defined as

\[ Mo = \frac{LK + ULK + P + D + OTA + OTD + OTJ + OTM}{M_{\text{PureOT}}} \]

Thus a relative message overhead of 1 reflects the dynamic lock with OT system incurs the same number of communication cost as a pure OT system. \( Mo \) above 1 reflects our system incurs more communication that a pure OT system. \( Mo \) below 1 reflects our system incurs less communication than a pure OT system. Thus a lower value is a reduction in communication costs.
Results: Dynamic Message Overhead

Dynamic OT Message Rate = % messages dealing with OT relative to all messages
Results: Communication Efficiency

- Odd simulations = 3 users
- Even simulations = 9 users
Results: 3 Users

Client Edit Behavior and Communication Efficiency Relative to Pure OT - 3 Users
Results: 9 Users

Client Edit Behavior and Communication Efficiency
Relative to Pure OT - 9 Users
Results: 18 Users

Client Edit Behavior and Communication Efficiency Relative to Pure OT - 18 users

Efficiency Relative to Pure OT (Dynamic Messages / Pure OT Messages)

Document Structure

- Random
- Hybrid
- Clustered
Generalized OT

- Even though we simulated using OT at the leaf level, we can apply OT at any level.
- To do this, the OT primitives `Insert` & `Delete` are now defined as:
  - `Insert[O, V, P]`: insert object O within node V at position P.
  - `Delete[O, V, P]`: delete object O within node V at position P.
- V has been added, and O is now semantically significant.

5/18/2007
Contents

- Introduction
- Motivation
- Problem Statement and Goals
- Contributions
- Architecture
- Relaxed Consistency
- Integrating OT
- Peer-to-Peer Document Management
- Hierarchical Reduction
- Prototypes
- Conclusions
P2P Document Management

- Previous work was client-server based
  - Dynamic lock management
  - Cache changes
  - Apply OT locally for a subset of users in same section

- Can we adopt a P2P architecture?
  - Avoid central point of failure at server
  - Avoid bottleneck at server
  - Distribute document management among all peers
Client-Server Approach

- LockRequest (with demotion)
- LockRelease (with promotion)
P2P Lock Management
Fairness in Management

- Each node in the document tree must be managed by a peer
- First peer in the RTCES will obtain root, and thus be heavily burdened thereafter
- Need to balance workload of managing nodes (especially the topmost)
  - When lock request is made, adopt a most-recent approach
Simulation

- Model peer with same edit pattern as client-server simulation
  - Reader and writer state
  - Transitions induce lock request (R→W) and release (W→R)
- Document configurations
  - Vary mean branching factor (number of leaves)
- Measure messages sent and computations performed at each peer
Modeling the Edit Behavior

- **Enter CES (100)**
  - **Reading**
    - No Action (60)
    - Move to Another Section (20)
    - Make Modification (20)
  - **Not In CES**
    - No Action (20)
  - **Writing**
    - Modify (60)

- Move to Another Section (20)
Simulation OOD

- Peer
- Node
- Operation
- Data Structure
- enumerations
- Simulation
Load Balancing via P2P

- 1-32 Peers
- 14, 28, and 56 leaves
- Each column represents one simulation run
- Total work remains same
- Variance decreases
Communication Costs

- Communication costs remain lower than Pure OT
- This does not include OTJ, OTA, and OTD, so red plot would be higher
Contents

- Introduction
- Motivation
- Problem Statement and Goals
- Contributions
- Architecture
- Relaxed Consistency
- Integrating OT
- Peer-to-Peer Document Management
  - Hierarchical Reduction
- Prototypes
- Conclusions
Hierarchical Reduction

- Managing document hierarchically is beneficial to localize OT costs
- But as changes are made
  - History buffers of leaf nodes grow
  - Performance of the OT algorithm degrades
- Minimize OT computation costs by reducing (compressing) history buffers
- Push changes from leaves up to root and merge them
  - Create global snapshots of document state
Reduction

- Sort all operations
- Delete pairs that cancel each other
- Combine sequences of operations

Reduced HB can now be sent to Parent
Hierarchical Merging

- Upon promotion/demotion, reduce and merge up the document tree
- Affords an opportunity to check for intention violations (heretofore not possible)
Node Model

- Maintain history buffer with reduction engine
- Receive operations from children and peers, and pass them up to parent
Cost Savings of Reduction

- By keeping history buffers small, OT computation costs are reduced
- Simulated users’ behavior as before (reader/writer state)
- Increased number of users up to 88 (to get sufficient collaboration density)
- 3 simulation types
  - No reduction
  - Reduction upon promote/demote
  - Reduction on promote/demote and when user enters/leaves an OT set
Reduction Simulation Results

Decreasing OT Computation Cost via Reduction

- No reduction
- Reduce on promote/demote
- Reduce when user enters/leaves OT set and on promote/demote

Collaboration Density

Computation Cost of Performing OT
Contents

- Introduction
- Motivation
- Problem Statement and Goals
- Contributions
- Architecture
- Relaxed Consistency
- Integrating OT
- Peer-to-Peer Document Management
- Hierarchical Reduction
- Prototypes
- Conclusions
Prototypes

- Used simulation-driven design process
  - Model client and server
  - Model client and connect to real server
  - Use real client and server

- DEVS Java used as simulation system

- Open system approach: C#/.NET for Web service
Connecting the DEVS Simulation to the C# Web Service

- **OutConnection object**
  - Translates simulation messages from clients to actual calls to Web service
  - Receives responses
  - Translates back to simulation messages to return to client

- **Combination of Java and C# technologies**
Web Service Running

- Connects to CVSNT and implements document management (lock proxy)
Visualizing Document Tree
Client Editor

- Connects to Web Service
- Demonstrates working architecture
- Some limitations
  - OTJ, OTM, OTA, OTD, P, and D messages not implemented
Contents

- Introduction
- Motivation
- Problem Statement and Goals
- Contributions
- Architecture
- Relaxed Consistency
- Integrating OT
- Peer-to-Peer Document Management
- Hierarchical Reduction
- Prototypes
- Conclusions
Conclusion

- A systematic view of RTCES
- Open system architecture
- View document as a hierarchy
- Localize OT
- Reduce communication and computation costs
- Load balance document management via P2P
- Identify points for intention preservation
- Prototypes of client and server
# Achieved Goals

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Combine existing client and server technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hierarchical document management</td>
<td>Reduce communication costs</td>
</tr>
<tr>
<td>Incorporation of OT</td>
<td>Ensure all users have complete concurrent access</td>
</tr>
<tr>
<td>P2P balancing</td>
<td>Load balance document management among peers</td>
</tr>
</tbody>
</table>
| Hierarchical reduction and intention preservation | Decrease computation cost
| Prototypes            | Demonstrate viability of our approaches         |
Publications


Future Work

- Work with Srikanth Tirupathi on the client editor
- Further examine intention preservation theory
- Examine space savings of reduction
References


References (2)


References (3)


29. Eßmann, B; Funke, H; Providing Peer-to-Peer Features to Existing Client-Server CSCW Systems. In: Chen, Chi-Sheng; Filipe, Joaquim; Seruca, Isabel; Cordeiro, José editor. Proceedings of the 7th International Conference On Enterprise Information Systems (ICEIS 2005), volume 4, S. 271-274, Miami, FL, USA, 24 - 28 May 2005 INSTICC
References (4)


References (6)

References (7)


References (8)

References (9)

79. Microsoft SharePoint Services
References (10)


2. O'Reilly, C., A Weakly Constrained Approach to Software Change Coordination. ICSE 2004: 66-68


References (11)

References (12)


References (13)


Roth, J. and Unger C. Developing synchronous collaborative applications with TeamComponents. 4th International Conference on Cooperative Systems. 2000.

References (14)


References (15)


References (16)


References (17)


References (19)


