Scalable Concurrency Control for Massively Collaborative Virtual Environments

Patrick Lange, Rene Weller, Gabriel Zachmann

University of Bremen, Germany

cgvr.cs.uni-bremen.de

MMVE 2015, 20 March 2014, Portland
Concurrency Control for CVE

- Process of managing simultaneous execution of user transactions on shared virtual objects
Concurrent Control for CVE

- Process of managing simultaneous execution of user transactions on shared virtual objects

(Mesh data from Stanford 3D Models
https://graphics.stanford.edu/data/3Dscanrep/)
Concurrency Control for CVE

- Process of managing simultaneous execution of user transactions on shared virtual objects
Concurrent Control for CVE

- Process of managing simultaneous execution of user transactions on shared virtual objects
Concurrent Control for CVE

- Process of managing simultaneous execution of user transactions on shared virtual objects
Concurrent Control for CVE

- Process of managing simultaneous execution of user transactions on shared virtual objects

Responsiveness

Scalability <-> Consistency

Motivation  Related Work  Our Approach  Results  Conclusion
Concurrent Control for CVE

- Process of managing simultaneous execution of user transactions on shared virtual objects
  - Can lead to frustrated user experience or even user completely losing interest in the application [Roberts‘04][Bouckerche‘05]

Motivation  Related Work  Our Approach  Results  Conclusion
CCM for CVEs so far

Concurrent Control Mechanisms
CCM for CVEs so far

Concurrency Control Mechanisms

- Locking
  - Standard
    - User A: LAT Access LRT
    - User B: LAT Access LRT
    - User C: LAT Access
  - Filtered
CCM for CVEs so far

- **VSculpt**: A distributed virtual environment for collaborative design [Li‘03]
- **Architectures for shared haptic virtual environments** [Buttolo‘97]
CCM for CVEs so far

- ATLAS – A scalable network framework for distributed virtual environments [Lee‘07],
- Scalable prediction based concurrency control for distributed virtual environments [Yang‘00]
**CCM for CVEs so far**

- Performance evaluation of compromised synchronization control mechanism for distributed virtual environment
  [Wongwirat‘06]
CCM for CVEs so far

Concurrency Control Mechanisms

Locking

- Standard
  - User A: LAT, Access, LRT
  - User B: LAT, Access, LRT
  - User C: LAT, Access

- Filtered

Non-Locking

- Lock-Free
  - User A: Access
  - User B: Access
  - User C: Access, Rollback

- Wait-Free
  - User A: Access
  - User B: Access
  - User C: Access, Rollback

Motivation

Related Work

Our Approach

Results

Conclusion
Our Contribution

- Novel approach to concurrency control for massively collaborative virtual environments
  - Not affected by network delays
  - No problems from previous approaches like deadlocks or starvation
- High performance access
  - Almost constant runtime with very low synchronisation overhead
  - Multiple wait-free read and write operations

Responsiveness

Scalability

Consistency
Basic Idea

- Assignment of unique key-value pair to each data packet which is exchanged between users and virtual objects
- Key-value pool holds complete shared world state
- De-coupling and parallelization of read, write and data deletion processes
Wait-Free Read

Producer

Consumer

Related Work

Our Approach

Results

Conclusion
Wait-Free Read

Producer

Consumer

Key-Value Pool

Bunny

Producer Reference

Consumer Reference

Motivation

Related Work

Our Approach

Results

Conclusion
Wait-Free Read

Producer

Consumer

Key-Value Pool

Bunny

Producer Reference

Consumer Reference

Producer

Consumer
Wait-Free Read

Key-Value Pool

Producer

Consumer

Producer Reference

Consumer Reference

P1

C1

Our Approach
Wait-Free Read

Key-Value Pool

Bunny

Producer Reference

Consumer Reference

Producer

P1

P2

Consumer

C1
Wait-Free Read

Key-Value Pool

Producer

Consumer

P1

C1

Our Approach
Wait-Free Read

Key-Value Pool

Producer

Consumer

P1

P2

C1

Our Approach
Wait-Free Read

- Producer
- Parallel Deletion
- Consumer

Motivation  Related Work  Our Approach  Results  Conclusion
Wait-Free Read

Motivation
Related Work
Our Approach
Results
Conclusion
Wait-Free Read

Key-Value Pool

- Producer
- Consumer

P1
P2

C1
C2

Producer Reference
Consumer Reference

Bunny

Producer Reference

Key-Value Pool

Motivation
Related Work
Our Approach
Results
Conclusion
Wait-Free Read

Key-Value Pool

Bunny

Producer Reference

Consumer Reference

Local Marker

Producer

P1

C1

P2

C2

Related Work

Our Approach

Results

Conclusion
Wait-Free Write

Producer

P1

Bunny

Producer Reference

Consumer Reference

Key-Value Pool

Producer

P2

Motivation

Related Work

Our Approach

Results

Conclusion
Wait-Free Write

Related Work

Our Approach

Results

Conclusion
Wait-Free Write

Motivation

Our Approach

Results

Conclusion
Wait-Free Write

Motivation

Related Work

Our Approach

Results

Conclusion

Producer

P1

P3

Consumer

Bunny

Producer Reference

Key-Value Pool

Consumer Reference

P2

P4
Wait-Free Write

Producer
P1
P3

Parallel Merge
P3
P4

Merge Strategy
Producer Reference

Consumer Reference

Bunny

Producer Reference

Key-Value Pool

Producer

P2
P4

Related Work

Our Approach

Results

Conclusion
Wait-Free Write

Motivation

Related Work

Our Approach

Results

Conclusion
Wait-Free Write

Motivation

Related Work

Our Approach

Results

Conclusion

Producer

Parallel Merge

Consumer

P1

P2

P3

P4

Key-Value Pool
Merging Example with Two Producers

Motivation

Related Work

Our Approach

Results

Conclusion
Merging Example with Two Producers

P1

P2
Merging Example with Two Producers

Motivation
Related Work
Our Approach
Results
Conclusion
Merging Example with Two Producers

\[ V = (10, 12, 10) \]

\[ P1(V) = (10, 15, 10) \]

\[ P2(V) = (12, 17, 10) \]
Merging Example with Two Producers

\[ V = (10, 12, 10) \]
\[ P1(V) = (10, 15, 10) \]
\[ P2(V) = (12, 17, 10) \]

\[ \text{Merge}(V) = (11, 16, 10) \]
Results

- Performance comparison with four competitors
  1. Hash map with standard locking mechanisms from the boost library
     - Read and write operations are locking
  2. Wait-free hash map based on previous work [Lange‘14]
     - Wait-free read and single wait-free write operations
  3. Optimistic hash map based on [Wongwirat‘07]
     - No locking for read operations, rollback of transaction if transaction fail occurs
  4. Filtered hash map based on [Li‘03]
     - Restriction on lock cast
Read (25%) & Write (75%) Operations

- **Our Approach**
- **Lock-Based Approach**
- **Wait-Free Approach**
- **Optimistic Approach**
- **Filtered Approach**

Access time in ms

Number of users accessing the key-value pool
Read (50%) & Write (50%) Operations

Our Approach
Lock-Based Approach
Wait-Free Approach
Optimistic Approach
Filtered Approach

Access time in ms
Number of users accessing the key-value pool
Conclusions

1. Scalable CCM for massively collaborative virtual environments
   - No deadlock, no starvation of user actions
   - Supports arbitrary non-blocking user interactions

2. Our novel CCM outperforms traditional approaches
   - Faster than a factor of 8-35
   - Less than 74% memory usage than our previous approach

3. Our novel CCM allows easy customization for many CVE applications
   - Data merge function can be defined for arbitrary purpose
   - Merge can also represent traditional approaches
Future Work

- Distributed implementation and testing
  1. Key-value pool as central host
Future Work

- Distributed implementation and testing
  1. Key-value pool as central host
  2. Distributing key-value pools

![Diagram showing Key-Value Pools](image-url)
Thank you for your attention

Questions?

Patrick Lange, Rene Weller, Gabriel Zachmann
{lange,weller,zach}@cs.uni-bremen.de

Funding by DLR, contract 50NA1318