GraphPool: A High Performance Data Management for 3D Simulations

Patrick Lange, Rene Weller, Gabriel Zachmann
University of Bremen, Germany
cgvr.cs.uni-bremen.de

ACM SIGSIM PADS
15-18 May 2016, Banff, AB, Canada
Data: Central Part in Simulations

- Generation, management and distribution of the global simulation state
- Managing the communication of many software components
Challenges in Data Engineering for Simulations

1. Performance (≥ realtime)
   - Simulation implementation vs. data storage

2. Scalability to massively parallel access
   - Parallelization of simulation workflow
   - Concurrency control

3. Adaptability to new data formats
   - Enrichment of simulation models
Relational Databases for Simulations

- Major data management used in modern architectures for 3D simulation applications
  - Strives for data consistency and transactional safety
  - Sacrifices performance and adaptability

- Schema and data synchronization for distributed 3D simulations
  [Hoppen'14,Rossmann'12]

- Store visualization data with collaboration [Julier'10,Walczak'12] or not [Schmalstieg'07]

- Static data schema [Haist'05] vs flexible data schema [Schmalstieg'07]
Relational Database Technology

- Motivation: Well-researched, easy-to-use, deliver out-of-the-box functionality

- Quick integration & implementation

- Relational database technology (aggregate queries, caching, consistency, …)

- Scalability and performance of massively parallel access due to serialization of queries

- Adaptability to new simulation data

- Performance bottleneck when transforming object-oriented data into table format of relational databases

Not the right tool for the job
Our Approach

- Replace relational database technology in complex simulation frameworks
  - No data transformation needed
  - No lock-based synchronization of transactions

- Our approach introduces
  - Graph-based data structure
  - Wait-free concurrency control
  - Key-based queries
  - Emulation of relational access queries
Our Approach

- Replace relational database technology in complex simulation frameworks
  - No data transformation needed
  - No lock-based synchronization of transactions

- Our approach introduces
  - Graph-based data structure
  - Wait-free concurrency control
  - Key-based queries
  - Emulation of relational access queries

System Components

- GraphPool
- Wait-Free API
- Key-based queries
- Object data

Motivation  Related Work  Our Approach  Results  Conclusion
Recap - Wait-free Hash Maps: Concept

- Assignment of unique identifiers to each data packet which is exchanged between software components
- Every data packet is stored inside a hash map which resembles the complete system state
- Relies on memory cloning and atomic operations
Recap - Wait-free Hash Maps: Features

- Guarantees access to the shared data structure in a finite number of steps (e.g. as traditional thread or OpenMP implementation)
- Does not need any traditional locking mechanism
- Delivers high performance even for massive concurrent access
Nested Hash Maps

- Emulating relational access queries requires
  - Unique identification of data
  - Linking structures between data
- Hash map representation advantages
  - Fast insert, deletion and lookup operations: $O(1)$

![Diagram of Nested Hash Maps]

**Component**

**HASH**

**Hash map bucket**

**Data**

**Motivation**

**Related Work**

**Our Approach**

**Results**

**Conclusion**
Nested Hash Maps

- One nested hash map emulates one table
- $n \cdot m$ table is represented by $m$ object keys and $n$ member keys
  - Every key acts as a SQL primary key
- Easy extension of stored data

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>University</th>
<th>Degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>Smith</td>
<td>Stanford</td>
<td>Prof.</td>
</tr>
<tr>
<td>42</td>
<td>Jones</td>
<td>Yale</td>
<td>Ph.D.</td>
</tr>
<tr>
<td>227</td>
<td>Walker</td>
<td>Cambridge</td>
<td>Ph.D</td>
</tr>
</tbody>
</table>
Property Graph Model

- Arrange nested hash maps in graph in order to enable relational queries via graph traversal
- Annotate and organize data with additional information (e.g. meta data)
## Property Graph Model: Example

### Relational table representation

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>University</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>Smith</td>
<td>Stanford</td>
</tr>
<tr>
<td>42</td>
<td>Jones</td>
<td>Yale</td>
</tr>
</tbody>
</table>

### Reference

<table>
<thead>
<tr>
<th>Reference</th>
<th>Paper</th>
<th>Contact Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>WK3</td>
<td>The 101 Simulation</td>
<td>23</td>
</tr>
</tbody>
</table>

### LID

<table>
<thead>
<tr>
<th>LID</th>
<th>ID</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>23</td>
<td>WK3</td>
</tr>
<tr>
<td>2</td>
<td>42</td>
<td>WK3</td>
</tr>
</tbody>
</table>
Query Examples

Relational table representation

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>University</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>Smith</td>
<td>Stanford</td>
</tr>
<tr>
<td>42</td>
<td>Jones</td>
<td>Yale</td>
</tr>
</tbody>
</table>

Reference | Paper | Contact Author
----------|-------|----------------|
WK3       | The 101 Simulation | 23       |

LID | ID | Reference
---|----|------------|
1   | 23 | WK3        |
2   | 42 | WK3        |

Our representation

```
Persson
   `- Uni
       `- Person
          `<- 101

Author
    `- Contact
```
Query Examples

Relational table representation

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>University</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>Smith</td>
<td>Stanford</td>
</tr>
<tr>
<td>42</td>
<td>Jones</td>
<td>Yale</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reference</th>
<th>Paper</th>
<th>Contact Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>WK3</td>
<td>The 101 Simulation</td>
<td>23</td>
</tr>
</tbody>
</table>

LID | ID  | Reference |
---|-----|-----------|
1  | 23  | WK3       |
2  | 42  | WK3       |
Evaluation

- Performance comparison of GraphPool, (on-disk/in-memory) relational databases and lock-based GraphPool
  - insert, select and aggregate queries
- Single and massively parallel access scenarios
- Verification of query results

- Test configuration:
  - C++ with -O3 optimization
  - Each test averages 10,000 read/write operations with varying data types (vectors, matrices, pointcloud data, strings, numerals)
Results: Single Access

- GraphPool
- Lock-based GraphPool
- In-memory relational databases
- Relational databases

Access time [ms]:
- 0,000
- 2,000
- 4,000

GraphPool
- Aggregate Statement
- Insert Statement
- Select Statement
Results: Single Access

- **GraphPool**: Access time range
- **Lock-based GraphPool**: Access time range
- **In-memory relational databases**: Access time range
- **Relational databases**: Access time range

Access time [ms]: 0.000 to 0.100
Results: Multi Access

- GraphPool
- Lock-based GraphPool
- In-memory relational databases
- Relational databases

Access time [ms]

- Aggregate Statement
- Insert Statement
- Select Statement
Results: Multi Access

![Bar chart showing access time for different types of databases: GraphPool, Lock-based GraphPool, In-memory relational databases, and Relational databases. The x-axis represents access time in milliseconds, ranging from 0.000 to 0.600. The y-axis lists the database types. The access time for GraphPool is the shortest, followed by Lock-based GraphPool, then In-memory relational databases, and finally Relational databases, which have the longest access time.](chart.png)
Our Contribution

- Novel data management for sophisticated (massively parallel) (3D) simulation applications
  - Allows non-locking read and write operations
  - No deadlock, no starvation of operations
  - Highly responsive, low-latency access for any number of simulation components
  - Emulates relational database access queries
- Outperforms traditional approaches by a minimum of factor 10

Performance ✓  Scalability ✓  Adaptability ✓
Thank you for your attention

Questions?

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

This research is based upon the project KaNaRiA, supported by German Aerospace Center (DLR) with funds of German Federal Ministry of Economics and Technology (BMWi) grant 50NA1318