A Comparison of Parallelization Methods for Data Flow Networks

Stephan Rehfeld and Marc Erich Latoschik

Beuth Hochschule für Technik Berlin and Universität Bayreuth

March 21, 2010
Outline

1. Overview
2. Parallelization
3. Traverser
4. Test environments
5. Results
Overview

Systems
- FlowVR
- DLoVe and instantreality
- SUED
- other systems

Multi-Threading developments
- nodes have own process or thread
- subgraphs
- distributed VR applications
- major system modules: rendering, physics and reading of sensors
Features and Architecture

Features

- data flow based
- scene graph is part of the data flow network
- head-tracking
- stereoscopy
- picking
- physics connection
- script connection
Architecture

Data flow network

- implemented in own library
- just an interface for traversers
- traversers are implemented separately
Traverser

- following separation on design level
- high flexibility
- changed at run time and used adaptively
- evaluation and propagation traverser ↔ Semantic traverser
- easy modification of order of processing
- target to tune performance
**Actions**

- traversal
- evaluation
Actions

- traversal
- evaluation

Observation

- evaluation normally takes more time
- complex graphs result in longer (pure) traversals, but they also consist of more nodes for evaluation.
Actions

- traversal
- evaluation
**Actions**

- traversal
- evaluation

**Observation**

- evaluation normally takes more time
- complex graphs result in longer (pure) traversals, but they also consist of more nodes for evaluation.
Parallelization

Techniques
- horizontal or
- vertical
Parallelization

Techniques

- horizontal or
- vertical
Parallelization

**Techniques**

- horizontal *or*
- vertical
Approach

Evaluate independent nodes
- easier to implement
- gives first results

Parameters
- parallelization technology
- amount of threads
- life cycle threads
- affinity of a thread to a core
Simple Traverser (ST)

Attributes
- most simplest traverser algorithm
- multiple evaluations
- can not process graphs with loops
Simple Traverser (ST)

Attributes

- most simplest traverser algorithm
- multiple evaluations
- can not process graphs with loops
Simple Traverser (ST)

Attributes

- most simplest traverser algorithm
- multiple evaluations
- can not process graphs with loops
Simple Traverser (ST)

Attributes

- most simplest traverser algorithm
- multiple evaluations
- can not process graphs with loops
Simple Traverser (ST)

Attributes

- most simplest traverser algorithm
- multiple evaluations
- can not process graphs with loops
Simple Traverser (ST)

Attributes

- most simplest traverser algorithm
- multiple evaluations
- can not process graphs with loops
Simple Traverser (ST)

Attributes

- most simplest traverser algorithm
- multiple evaluations
- can not process graphs with loops
Simple Traverser (ST)

**Attributes**
- most simplest traverser algorithm
- multiple evaluations
- can not process graphs with loops
Simple Traverser (ST)

Attributes

- most simplest traverser algorithm
- multiple evaluations
- can not process graphs with loops
Presorting Traverser (PST)

Attributes

- ordered by dependency of data
- no multiple evaluations
- cannot process graphs with loops
Presorting Traverser (PST)

**Attributes**

- ordered by dependency of data
- no multiple evaluations
- can not process graphs with loops
Presorting Traverser (PST)

Attributes

- ordered by dependency of data
- no multiple evaluations
- cannot process graphs with loops
Presorting Traverser (PST)

Attributes
- ordered by dependency of data
- no multiple evaluations
- can not process graphs with loops
Presorting Traverser (PST)

Attributes

- ordered by dependency of data
- no multiple evaluations
- can not process graphs with loops
Presorting Traverser (PST)

Attributes

- ordered by dependency of data
- no multiple evaluations
- can not process graphs with loops
Presorting Traverser (PST)

Attributes

- ordered by dependency of data
- no multiple evaluations
- cannot process graphs with loops
Presorting Traverser (PST)

Attributes

- ordered by dependency of data
- no multiple evaluations
- cannot process graphs with loops
Presorting Traverser (PST)

Attributes

- ordered by dependency of data
- no multiple evaluations
- can not process graphs with loops
Presorting Traverser (PST)

Attributes
- ordered by dependency of data
- no multiple evaluations
- can not process graphs with loops
Presorting Traverser (PST)

Attributes

- ordered by dependency of data
- no multiple evaluations
- cannot process graphs with loops
Loop breaking Traverser (LBT)

Loop breaking

- searching loops recursively
- set stop marks to break the loops
- using the stop marks for building the lists
Multi Thread Traverser without thread reusing (MT1)

Attributes

- sorting is already implemented
- only few changes necessary
- pthread
Multi Thread Traverser without thread reusing (MT1)

Attributes
- sorting is already implemented
- only few changes necessary
- pthread
Multi Thread Traverser without thread reusing (MT1)

Attributes

- sorting is already implemented
- only few changes necessary
- pthread
Multi Thread Traverser without thread reusing (MT1)

Attributes
- sorting is already implemented
- only few changes necessary
- pthread
Multi Thread Traverser with thread reuse (MT2)

Thread-Pools

- create threads on init (and let them sleep)
- dispatch lists of containers to threads
- wait until the threads sleep again
Multi Thread Traverser using OpenMP (MT3)

**OpenMP**

- annotate loop over current container
- dispatching and synchronization by OpenMP
- less changes than all other implementations
Test environments

- engine and data flow network with clock node, interpolator node and scene graph nodes
Test environments

- engine and data flow network with clock node, interpolator node and scene graph nodes
- graph with containers evaluating heavy calculation for searching prime number (good reproducibility)
First test environment

<table>
<thead>
<tr>
<th>interpolators</th>
<th>ST</th>
<th>PST</th>
<th>LBT</th>
<th>MT1</th>
<th>MT2</th>
<th>MT3</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>123</td>
<td>191</td>
<td>291</td>
<td>345</td>
<td>412</td>
<td>354</td>
</tr>
<tr>
<td>1,000</td>
<td>1,245</td>
<td>1,941</td>
<td>2,947</td>
<td>2,809</td>
<td>3,364</td>
<td>2,970</td>
</tr>
<tr>
<td>10,000</td>
<td>24,121</td>
<td>35,618</td>
<td>52,372</td>
<td>47,242</td>
<td>52,745</td>
<td>52,026</td>
</tr>
</tbody>
</table>

**Table:** Arithmetic mean of 10,000 measurements of the first test program denoted in $\mu$s.
## First test environment

<table>
<thead>
<tr>
<th>interpolators</th>
<th>ST</th>
<th>PST</th>
<th>LBT</th>
<th>MT1</th>
<th>MT2</th>
<th>MT3</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>123</td>
<td>191</td>
<td>291</td>
<td>345</td>
<td>412</td>
<td>354</td>
</tr>
<tr>
<td>1,000</td>
<td>1,245</td>
<td>1,941</td>
<td>2,947</td>
<td>2,809</td>
<td>3,364</td>
<td>2,970</td>
</tr>
<tr>
<td>10,000</td>
<td>24,121</td>
<td>35,618</td>
<td>52,372</td>
<td>47,242</td>
<td>52,745</td>
<td>52,026</td>
</tr>
</tbody>
</table>

**Table:** Arithmetic mean of 10,000 measurements of the first test program denoted in $\mu$s.
Second test environment
Second test environment

![Graph showing the performance of different parallelization methods across varying limits.](image)
Second test environment

![Graph showing the comparison of parallelization methods MT1, MT2, and MT3. The x-axis represents the limit, ranging from 0 to 8000, and the y-axis represents the ratio, ranging from 0 to 2. The graph compares three methods: MT1 - 2 Threads implicit, MT2 - 2 Threads implicit, and MT3 - 2 Threads implicit. The x-axis limits are indicated by arrows at the appropriate points on the graph.]
Second test environment
Second test environment

A Comparison of Parallelization Methods 19 (31) Stephan Rehfeld and Marc Erich Latoschik
Second test environment

![Graph showing the comparison of parallelization methods. The graph plots the ratio against the limit for different test environments: MT1, MT2, and MT3, each with 4 Threads implicit. The graph indicates the performance of each method across varying limits.]
Second test environment
Second test environment

![Graph showing the comparison of parallelization methods for different test environments. The graph plots the ratio against the limit for MT1, MT2, and MT3, each representing 8 Threads implicit. The x-axis represents the limit, and the y-axis represents the ratio. The graph shows how the ratio changes with increasing limits for each test environment.]
Second test environment

A Comparison of Parallelization Methods

Stephan Rehfeld and Marc Erich Latoschik
Second test environment

![Graph showing the performance of different parallelization methods.](image)

- MT2 - 2 Threads implicit
- MT2 - 4 Threads implicit
- MT2 - 8 Threads implicit

The graph compares the ratio of performance with varying limits for different thread counts, illustrating the efficiency gains achieved with increased parallelization.
Second test environment

![Graph showing comparison of parallelization methods](image)

- MT2 - 2 Threads implicit
- MT2 - 4 Threads implicit
- MT2 - 8 Threads implicit

A Comparison of Parallelization Methods 21 (31) Stephan Rehfeld and Marc Erich Latoschik
Second test environment

The diagram shows the performance of different parallelization methods across varying limits. The x-axis represents the limit, while the y-axis shows the ratio. The methods compared are:
- MT2 - 2 Threads implicit
- MT2 - 4 Threads implicit
- MT2 - 8 Threads implicit

The graph indicates that as the limit increases, the ratio decreases, showing improved performance. Arrows point to specific limit values where the performance is significantly improved.
Second test environment

A Comparison of Parallelization Methods
Second test environment

![Graph showing the comparison of parallelization methods with OMP. The graph plots ratio against limit for MT1, MT2, and MT3, each represented by a different line color. MT1 is in red, MT2 in green, and MT3 in blue. The lines show the performance of parallelization methods with OMP, indicating how they scale with increasing limit.]
Second test environment

A Comparison of Parallelization Methods
Second test environment

![Graph showing results for different parallelization methods]

- MT1 - 4 Threads explicit with OMP
- MT2 - 4 Threads explicit with OMP
- MT3 - 4 Threads explicit with OMP

The graph compares the ratios for different limits, with MT1 having the highest ratio and MT3 having the lowest.
Second test environment

![Graph showing the comparison of parallelization methods with OMP for MT1, MT2, and MT3. The x-axis represents the limit, and the y-axis represents the ratio. The graph shows the performance improvement with different limits for each method.](image)
Second test environment

![Graph showing comparison of parallelization methods](image)
Second test environment

- MT1 - 8 Threads explicit with OMP
- MT2 - 8 Threads explicit with OMP
- MT3 - 8 Threads explicit with OMP

A Comparison of Parallelization Methods 24 (31) Stephan Rehfeld and Marc Erich Latoschik
Second test environment

![Graph showing the comparison of parallelization methods.](image)

- MT1 - 8 Threads explicit with OMP
- MT2 - 8 Threads explicit with OMP
- MT3 - 8 Threads explicit with OMP

A Comparison of Parallelization Methods 24 (31) Stephan Rehfeld and Marc Erich Latoschik
Second test environment

A Comparison of Parallelization Methods 25 (31) Stephan Rehfeld and Marc Erich Latoschik
Second test environment

MT1 - 2 Threads explicit
MT2 - 2 Threads explicit
Second test environment

A Comparison of Parallelization Methods 25 (31) Stephan Rehfeld and Marc Erich Latoschik
Second test environment
Second test environment

![Graph showing comparison between MT1 and MT2 parallelization methods. The graph plots ratio against limit, with MT1 and MT2 represented by red and green lines, respectively. The Y-axis is labeled 'ratio' ranging from 0.5 to 2, and the X-axis is labeled 'limit' ranging from 0 to 8000. The graph indicates a decrease in ratio as the limit increases, with MT1 consistently below MT2.]
Second test environment

![Graph comparing parallelization methods]

MT1 - 8 Threads explicit
MT2 - 8 Threads explicit
Second test environment

![Graph showing the performance of different parallelization methods](image-url)
Second test environment

A Comparison of Parallelization Methods

Stephan Rehfeld and Marc Erich Latoschik
Second test environment

A Comparison of Parallelization Methods

Stephan Rehfeld and Marc Erich Latoschik
Second test environment

![Graph showing performance comparison between different parallelization methods.]

- MT2 - 2 Threads implicit
- MT3 - 2 Threads explicit with OMP

A Comparison of Parallelization Methods
Stephan Rehfeld and Marc Erich Latoschik
Conclusion

- not appropriate for many light-weight containers
- good for few heavy-weight containers
- reuse threads and set affinity
- break even for OpenMP at 92 $\mu$s other at 140 $\mu$s
- number of threads $=$ number of cores
- prevent cache-mismatches
- performance can nearly be doubled
Conclusion

Thank you for your attention