Data Aggregation and Alternative Visualization Techniques for Parallel and Distributed Program Analysis

Lucas Mello Schnorr (CNRS)  
LIG-MESCAL, Grenoble, France

TUD-ZIH-Colloquium – Dresden, Germany  
July 26th, 2012
Introduction

- Performance analysis $\rightarrow$ visualization
  - Register behavior during program execution
  - Offline, or post-mortem, visual analysis

- Traces characteristics
  - Timestamped and typed events
  - Very detailed in time $\rightarrow$ micro to nanoseconds
  - Many observed entities (processes, threads)
Introduction

- Performance analysis → visualization
  - Register behavior during program execution
  - Offline, or post-mortem, visual analysis
- Traces characteristics
  - Timestamped and typed events
  - Very detailed in time → micro to nanoseconds
  - Many observed entities (processes, threads)

Analysis through trace visualization

- Explore human perception, intuitive
- Interactive and exploratory approach
Challenges and Motivation

- Very large applications
  → Top500 has machines with 1.5 million cores
- Low-intrusion tracing techniques
  → Buffering, hardware support, simulation traces
Challenges and Motivation

- Very large applications
  - Top500 has machines with 1.5 million cores
- Low-intrusion tracing techniques
  - Buffering, hardware support, simulation traces

Space/Time trace size explosion

- Very detailed in time, many entities in space
- Data representation without care
  - may deceive understanding
Challenges and Motivation

- Very large applications
  → Top500 has machines with 1.5 million cores
- Low-intrusion tracing techniques
  → Buffering, hardware support, simulation traces

Space/Time trace size explosion

- Very detailed in time, many entities in space
- Data representation without care
  → may deceive understanding

- Real BOINC availability trace file
  - Availability is either true or false
  - 8-month period, then 12-day zoom
  - One volunteer
- Plot with GNUPlot to a PDF (vector) file
Motivation (BOINC example)

- One volunteer client (top: 8-month, bottom: 12-day)
- Reasonable view, with a zoom for details
<table>
<thead>
<tr>
<th>Motivation – trust the rendering?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same vector file, two different views</td>
</tr>
<tr>
<td>→ Different interpretation depending on the viewer</td>
</tr>
</tbody>
</table>
Motivation – trust the rendering?

Same vector file, two different views

→ Different interpretation depending on the viewer

Evince

Acroread
Motivation – trust the rendering?

Same vector file, two different views

→ Different interpretation depending on the viewer

- Should we trust the rendering?
  - No!
  - We need to make choices before visualizing data
Motivation → data aggregation

- 24-hour time integration
Space/Time views for trace analysis

- Widespread, useful, intuitive, fast adoption
- Space (vertical axis) and Time (horizontal)
- All trace events represented, causal order

Paje
http://paje.sf.net

Vite
http://vite.gforge.inria.fr

Vampir
http://vampir.eu

However...

Also impacted by ever larger trace sizes
Limited visualization scalability
Space/Time views for trace analysis

- Widespread, useful, intuitive, fast adoption
- Space (vertical axis) and Time (horizontal)
- All trace events represented, causal order

Paje
http://paje.sf.net

Vite
http://vite.gforge.inria.fr

Vampir
http://vampir.eu

However...

- Also impacted by ever larger trace sizes
- Limited visualization scalability
Space/Time views – closer look (ViTe tool)

- Trust the OpenGL rendering, no data aggregation

Source: http://vite.gforge.inria.fr
Space/Time views – closer look (new Pajé)

- Trust the two types of rendering
  → without or with OpenGL
Trust the two types of rendering
→ without or **with** OpenGL
Space/Time views – closer look (old Pajé)

- Opaque aggregating filter (no user interaction)
  → Slashed rectangles represent time-integrated states
- Self-configure depending on temporal zoom

Source: http://paje.sourceforge.net
Space/Time views – closer look (old Pajé)

- Space dimension: one process per vertical pixel
  → at best, 1000 processes represented at the same time

- Clustering algorithms by process behavior?
  → Remove similar processes and choose a representative
Introduction – summary and approach

- Data aggregation is **key** for large-scale visualization
  → Avoid graphical aggregation rendering

- Aggregated data may be more **representative**

- **Note**: Concerns with behavior attenuation
  - Aggregation may remove important details
  - Flexible aggregation: operators & neighborhood
Introduction – summary and approach

- Data aggregation is **key** for large-scale visualization → Avoid graphical aggregation rendering

- Aggregated data may be more **representative**

- **Note:** Concerns with behavior attenuation
  - Aggregation may remove important details
  - Flexible aggregation: operators & neighborhood

**Main idea:**
Visualization techniques based upon aggregated data

- Spatial and temporal trace aggregation
- **Alternative** visualization techniques
  - Squarified Treemap View
  - Hierarchical Graph View
Outline

1. Space/Time Trace Aggregation

2. Visualization techniques
   - Squarified Treemap View
   - Hierarchical Graph View

3. Some scenarios

4. Conclusion
Temporal integration

1. Time interval defined during the analysis
2. Summary of events for each monitored entity
Space/Time Trace Aggregation

- Temporal integration
  1. Time interval defined during the analysis
  2. Summary of events for each monitored entity

Time-integrated summary for processes

Numbers are in seconds (Execution, Blocked)

A=(4,5)  B=(7,2)  C=(3,6)  D=(9,0)  E=(4,5)
Space/Time Trace Aggregation

- Spatial integration
  1. Define a neighborhood for each monitored entity
  2. Apply an aggregating operator on the neighborhood

- Neighborhood as a hierarchy
  - Resource-based
  - Application groups

- Deeper the hierarchy $\rightarrow$ higher the quality
Space/Time Trace Aggregation

- Spatial integration
  1. Define a neighborhood for each monitored entity
  2. Apply an aggregating operator on the neighborhood

- Neighborhood as a hierarchy
  - Resource-based
  - Application groups
  - Deeper the hierarchy → higher the quality

Space-integrated summary

Aggregating operator: addition (Execution, Blocked)

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>M1</td>
<td>C1</td>
<td>G</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>M2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>C2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>M3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- A=(4,5)
- B=(7,2)
- C=(3,6)
- D=(9,0)
- E=(4,5)
- M1=(11,7)
- M2=(12,6)
- M3=(4,5)
- C1=(11,7)
- C2=(16,11)
- G=(27,18)
Alternative Visualization techniques

- Based on trace aggregation data
- Keep visualization scalability under control

Techniques

- Squarified Treemap View
  - Observe outliers, differences of behavior
  - Compare behavior
- Hierarchical Graph View
  - Pin-point resource contention
  - Correlate application behavior to network topology

Design
- How time and space-aggregated data is represented
- An example
Squarified Treemap View – Basic concepts

- **Scalable** representation for hierarchies  
  → when compared to node-link diagrams  
  → better **visualization scalability** for large trees

- Complementary to the space/time view

- Hard to discern the structure of hierarchy  
  → Borders help, but occupy space. Cushion treemaps?  
  - Adopt the simple algorithm + interaction
Squarified Treemap View – Basic concepts

- **Scalable** representation for hierarchies
  - when compared to node-link diagrams
  - better visualization scalability for large trees

- Complementary to the space/time view

- Hard to discern the structure of hierarchy
  - Borders help, but occupy space. Cushion treemaps?
    - Adopt the simple algorithm + interaction

- Space-filling top-down recursive layout algorithm
  - Node value → space occupied in the screen
  - Squarified version → keeps rectangles ratio close to 1

```
T=6
M=3
N=2
O=1
A=1 B=1 C=1 D=1 E=1 F=1
```
Squarified Treemap View – Basic concepts

- **Scalable** representation for hierarchies
  → when compared to node-link diagrams
  → better visualization scalability for large trees

- Complementary to the space/time view

- Hard to discern the structure of hierarchy
  → Borders help, but occupy space. Cushion treemaps?
    - Adopt the simple algorithm + interaction

- Space-filling top-down recursive layout algorithm
  - Node value → space occupied in the screen
  - Squarified version → keeps rectangles ratio close to 1
Squarified Treemap View – Basic concepts

- **Scalable** representation for hierarchies
  - when compared to node-link diagrams
  - better *visualization scalability* for large trees
- Complementary to the space/time view
- Hard to discern the structure of hierarchy
  - Borders help, but occupy space. Cushion treemaps?
    - Adopt the simple algorithm + interaction
- Space-filling top-down recursive layout algorithm
  - Node value → space occupied in the screen
  - Squarified version → keeps rectangles ratio close to 1
Squarified Treemap View – Basic concepts

- **Scalable** representation for hierarchies
  → when compared to node-link diagrams
  → better visualization scalability for large trees

- Complementary to the space/time view

- Hard to discern the structure of hierarchy
  → Borders help, but occupy space. Cushion treemaps?
    - Adopt the simple algorithm + interaction

- Space-filling top-down recursive layout algorithm
  - Node value → space occupied in the screen
  - Squarified version → keeps rectangles ratio close to 1
Considering **temporal** aggregation only

- Sum of all time-integrated data $\rightarrow$ node screen space
- Each time-integrated variable $\rightarrow$ inner-node division

**Note:** nodes might have different sizes $\rightarrow$ depends on the time-slice and the state values

Time-slice changes $\rightarrow$ new treemap layout rendered
Consider the **spatial-temporal aggregation**

**Sum of space-integrated data → node screen space**

**Each space-integrated variable → inner-node division**
Considering spatial-temporal aggregation

- Sum of space-integrated data → node screen space
- Each space-integrated variable → inner-node division
Squarified Treemap View – Aggregated Data

- Considering **spatial-temporal aggregation**

- Sum of space-integrated data → node screen space
- Each space-integrated variable → inner-node division
Squarified Treemap View – Aggregated Data

- Considering **spatial-temporal aggregation**

- **Sum of space-integrated data** → node screen space
- Each space-integrated variable → inner-node division
Squarified Treemap View – Aggregated Data

- Considering **spatial-temporal aggregation**

  - Sum of space-integrated data → node screen space
  - Each space-integrated variable → inner-node division

  ![Diagram showing spatial-temporal aggregation](image-url)
Considering **spatial-temporal** aggregation

- Sum of space-integrated data → node screen space
- Each space-integrated variable → inner-node division

The analyst decides
- time-slice
- hierarchy depth to draw
Squarified Treemap View – an example

- 1000 processes, in one of two states (synthetic)
- Aggregation level: 0, 1, 2, 3
Squarified Treemap View – an example

- 1000 processes, in one of two states (synthetic)
- Aggregation level: 0, 1, 2, 3
Squarified Treemap View – an example

- 1000 processes, in one of two states (synthetic)
- Aggregation level: 0, 1, 2, 3
Squarified Treemap View – an example

- 1000 processes, in one of two states (synthetic)
- Aggregation level: 0, 1, 2, 3
Hierarchical Graph View

- **Scalable** representation for graphs
  - Topology, with application-level metrics
  - Identify resource **bottleneck** in space and time
- Use spatial-temporal aggregated traces
- Interactive force-directed layout (Barnes-Hut algorithm)
Hierarchical Graph View

- **Scalable** representation for graphs
  - Topology, with application-level metrics
  - Identify resource **bottleneck** in space and time
- Use spatial-temporal aggregated traces
- Interactive force-directed layout (Barnes-Hut algorithm)
- Map trace metrics to geometrical attributes
  - Size, shape, filling, colors
  - **Nodes**: monitored entities
  - **Edges**: relationship among entities

![Diagram of hierarchical graph view](image)

- Hosts → squares
- Links → diamonds
Hierarchical Graph View - Aggregated Data

- Considering **temporal** aggregation only

- Graph changes $\rightarrow$ force-directed updates positions
- Time-slice changes $\rightarrow$ new layout is rendered
Hierarchical Graph View - Aggregated Data

- Considering **spatial-temporal** aggregation
  - Nodes are organized as a hierarchy
    - Based on geo or logical location
    - Application-dependent – get from traces
  - Analyst can control the level

- Aggregated representation
  - Many shapes depending on aggregated entities
Hierarchical Graph View - Aggregated Data

- Considering *spatial-temporal* aggregation
  - Nodes are organized as a hierarchy
    → Based on geo or logical location
    → Application-dependent – get from traces
  - Analyst can control the level
- Aggregated representation
  - Many shapes depending on aggregated entities
    → First aggregation
Considering spatial-temporal aggregation
- Nodes are organized as a hierarchy
  → Based on geo or logical location
  → Application-dependent – get from traces
- Analyst can control the level

Aggregated representation
- Many shapes depending on aggregated entities

→ First aggregation  → Second aggregation
Hierarchical Graph View - Aggregated Data

- Considering **spatial-temporal** aggregation
  - Nodes are organized as a hierarchy
    - Based on geo or logical location
    - Application-dependent – get from traces
  - Analyst can control the level

- Aggregated representation
  - Many shapes depending on aggregated entities

- Analyst decides
  - time-slice
  - the cut on the hierarchy (defining a new graph)
Hierarchical Graph View - an example

- Squares are hosts, diamonds are network links
  - Colors represent different applications
  - or parts of it (task type, phase)
- Two clusters interconnected by four network links
  - Cluster backbones have larger bandwidth capacity
  - Each host connected to the backbone by a private link
Some scenarios

1. BOINC fair sharing
2. Work stealing with KAAPI
3. Large scale treemap visualization
4. NAS-DT with graph view
Scenario 1 – BOINC fair sharing

- Setting: BOINC simulation (simulate the client side)
  - Two BOINC project servers with continuous jobs
  - 65 volunteers, must be fair between the two projects
  - Ten weeks, real availability traces from FTA

- Aggregation: whole time – all volunteer clients
Scenario 1 – BOINC fair sharing

- Setting: BOINC simulation (simulate the client side)
  - Two BOINC project servers with continuous jobs
  - 65 volunteers, must be fair between the two projects
  - Ten weeks, real availability traces from FTA
- Aggregation: whole time – all volunteer clients

<table>
<thead>
<tr>
<th>Project A</th>
<th>Project B</th>
</tr>
</thead>
<tbody>
<tr>
<td>~52.30 %</td>
<td>~47.70 %</td>
</tr>
</tbody>
</table>
Scenario 1 – BOINC fair sharing

- Setting: BOINC simulation (simulate the client side)
  - Two BOINC project servers with continuous jobs
  - 65 volunteers, must be fair between the two projects
  - Ten weeks, real availability traces from FTA
- Aggregation: whole time – all volunteer clients
Scenario 1 – BOINC fair sharing

- Setting: BOINC simulation (simulate the client side)
  - Two BOINC project servers with continuous jobs
  - 65 volunteers, must be fair between the two projects
  - Ten weeks, real availability traces from FTA
- Aggregation: whole time – all volunteer clients

- Analysis: Small volunteer contribution → not fair
Scenario 2 - Work stealing with KAAPI

- KAAPI: load balancing through random work stealing
- 188 processes running on five clusters

Analysis: stealing requests depends on latency
- Porto Alegre – France: ~300 ms
- In France: ~10 ms
Scenario 2 - Work stealing with KAAPI

- KAAPI: load balancing through random work stealing
- 188 processes running on five clusters
Scenario 2 - Work stealing with KAAPI

- KAAPI: load balancing through random work stealing
- 188 processes running on five clusters

Analysis: stealing requests depends on latency
Porto Alegre – France: ~300 ms
In France: ~10 ms
Scenario 3 - Synthetic, Large Scale

- Synthetic trace with 100 thousand processes
- Two states, four-level hierarchy
- Visualization artifacts without spatial aggregation

Hierarchy: Site (10) - Cluster(10) - Machine (10) - **Processor** (100)
Scenario 3 - Synthetic, Large Scale

- Synthetic trace with 100 thousand processes
- Two states, four-level hierarchy
- Visualization artifacts without spatial aggregation

Hierarchy: Site (10) - Cluster(10) - Machine (10) - Processor (100)
Scenario 3 - Synthetic, Large Scale

- Synthetic trace with 100 thousand processes
- Two states, four-level hierarchy
- Visualization artifacts without spatial aggregation

Hierarchy: Site (10) - Cluster (10) - Machine (10) - Processor (100)
Scenario 3 - Synthetic, Large Scale

- Synthetic trace with 100 thousand processes
- Two states, four-level hierarchy
- Visualization artifacts without spatial aggregation

Hierarchy: **Site** (10) - Cluster (10) - Machine (10) - Processor (100)
Scenario 3 - Synthetic, Large Scale

- Synthetic trace with 100 thousand processes
- Two states, four-level hierarchy
- Visualization artifacts without spatial aggregation

Maximum Aggregation
Scenario 4 - NAS-DT Class A WH

- NAS DT Class A White Hole algorithm
  → Traces from SMPI (Simulated MPI, part of SimGrid)
- Network topology – resource utilization by red filling
- Only temporal aggregation

Analysis: interconnection backbone is the bottleneck
Scenario 4 - NAS-DT Class A WH

- NAS DT Class A White Hole algorithm
  → Traces from SMPI (Simulated MPI, part of SimGrid)
- Network topology – resource utilization by red filling
- Only temporal aggregation
Scenario 4 - NAS-DT Class A WH

- NAS DT Class A White Hole algorithm
  → Traces from SMPI (Simulated MPI, part of SimGrid)
- Network topology – resource utilization by red filling
- Only temporal aggregation
Scenario 4 - NAS-DT Class A WH

- NAS DT Class A White Hole algorithm
  → Traces from SMPI (Simulated MPI, part of SimGrid)
- Network topology – resource utilization by red filling
- Only temporal aggregation
Scenario 4 - NAS-DT Class A WH

- NAS DT Class A White Hole algorithm
  → Traces from SMPI (Simulated MPI, part of SimGrid)
- Network topology – resource utilization by red filling
- Only temporal aggregation

- Analysis: interconnection backbone is the bottleneck
Another deployment with a different mapping
→ by changing the order of machines in hostfile

Explore communication locality
Scenario 4 - NAS-DT Class A WH (second try)

- Another deployment with a different mapping → by changing the order of machines in hostfile
- Explore communication locality

Note: Small scale and easy scenario – but it is a start
Conclusion

- Data aggregation
  - **Key** to scale data visualization for analysis
  - No pre-defined or fixed parameters
  - Fully configurable by the analyst
    - Time and space-slice, operators
Conclusion

- **Data aggregation**
  - **Key** to scale data visualization for analysis
  - No pre-defined or fixed parameters
  - Fully configurable by the analyst
    - Time and space-slice, operators

- **Visualization techniques**
  - Based upon aggregated data
  - Complementary to existing techniques
  - Continuous evaluation of visualization scalability
    - With larger data-sets, does it remain useful?
Conclusion

- **Data aggregation**
  - **Key** to scale data visualization for analysis
  - No pre-defined or fixed parameters
  - Fully configurable by the analyst
    - Time and space-slice, operators

- **Visualization techniques**
  - Based upon aggregated data
  - Complementary to existing techniques
  - Continuous evaluation of *visualization scalability*
    - With larger data-sets, does it remain useful?

- **Aggregation → behavior attenuation**
  - Have to be able to find the right time/space level
  - Keep the analyst in control
Open-source tools

- **Paje** (Space/Time views, pie-charts), LGPL
  - [http://paje.sourceforge.net](http://paje.sourceforge.net)
  - Since 2000, GNUstep-based, written in Objective-C
  - Not only a monolithic visualization tool
    - Component-based, graph of components
    - Framework for developing other tools
  - **Paje Protocol**
  - 30K SLOC, hard to maintain, hard to install GNUstep
Open-source tools

- **Paje** (Space/Time views, pie-charts), LGPL
  http://paje.sourceforge.net
  - Since 2000, GNUstep-based, written in Objective-C
  - Not only a monolithic visualization tool
    - Component-based, graph of components
    - Framework for developing other tools
    - **Paje Protocol**
  - 30K SLOC, hard to maintain, hard to install GNUstep

- **Triva** (Treemaps, Hierarchical graph), LGPL
  http://triva.gforge.inria.fr
  - Since 2007, GNUstep and Paje-based, also in Obj-C
    - Follows the Paje protocol
  - GNUstep runtime poses scalability problems
Future work

Technical

- **Paje++** (or Paje2) – complete re-write in C++, Qt
- **Viva** – visualization tool (Treemap, Hierarchical Graph) Also as a sandbox for developing new techniques
  [https://github.com/schnorr/viva](https://github.com/schnorr/viva) (coming soon)
- For both, debian packaging
Future work

Technical

- Paje++ (or Paje2) – complete re-write in C++, Qt
- Viva – visualization tool (Treemap, Hierachical Graph)
  Also as a sandbox for developing new techniques
  [https://github.com/schnorr/viva](https://github.com/schnorr/viva) (coming soon)
- For both, debian packaging

Research

- Better aggregation algorithms – for performance
- What about other aggregation operators?
- Aggregated data → space/time view
Thank you for your attention

Some references


More information

→ http://mescal.imag.fr/membres/lucas.schnorr/

INFRA-SONGS Project (WP-7)
http://infra-songs.gforge.inria.fr/
Simulation of Next Generation Systems
WP-7: Visualization and Analysis