Some Visualization Models applied to the Analysis of Parallel Applications

Lucas Mello Schnorr

Advisors:
Philippe O. A. Navaux & Denis Trystram & Guillaume Huard

Federal University of Rio Grande do Sul, Brazil
Grenoble Institute of Technology, France
Introduction - Context

- Distributed Systems → Grids
- Grid Interconnection and Scalability
  - Topology and Connectivity
  - Performance: bandwidth and latency
  - New resources can be added very easily

- Influence in the application execution
- Visualization – Performance Analysis
Introduction - Existing Tools/Techniques

- **Statistical Techniques**
  - ParaGraph (1990) – bar charts, utilization Count
  - Pablo (1993) – bar charts + 3D scatter plot

- **Behavioral Techniques**
  - Vampir (1996) – time-line system view

- **Structural Techniques**
  - ParaGraph (1990) – network display / hypercube
Introduction - Problem Identification

- Lack of a network-aware analysis
  - Difficult to analyze using space-time views
  - Structural techniques undeveloped
- Problems of visualization scalability
  - Visualization techniques limitations reached
  - Analysis are limited to hundreds of entities

Desirable Characteristics for Application Analysis

→ The Objectives

- Consider network properties
- Visualization scalability in the analysis
Introduction - The Thesis Approach

- Explore techniques from Information Visualization
- Context of parallel application analysis
  - Grid resources
  - Thread/Process parallel applications

Proposed Visualization Models

- Behavioral and Structural/Statistical (3D)
  - Communication Pattern
  - Network topology + Communication Pattern
  - Logical representation
- Visual Aggregation
  - Large-scale traces
  - Local and Global summaries
3D Model - Visual Conception

- Resources represented in 2D
  - Structural (e.g. a graph)
  - Statistical
- Vertical dimension is time
  - Objects’ Behavior Evolution
  - States and Links
- Interaction Techniques
  - Notion of a Camera
  - Rotation
  - Translation
  - Objects Animation
  - Replay step-by-step
3D Model - Differences from existing tools

- 3D Statistical Representation
  - Pablo → 3D Scatter Plot
  - Paradyn → 3D Terrain
  - ParaProf → Triang Mesh, 3D Bar and 3D Scatter Plot

- 3D Behavioral Representation
  - ParaProf → 2 metrics and time
  - Virtue → the time-tunnel view

Our Approach

- Presence of a timeline to show objects’ evolution
- Multiple Configurations in the visualization base
3D Model - Abstract Component Model

Input Data

- Application Traces
  - Timestamp-based events
  - Behavior registered
- Resources Description
  - Network topology: graph
  - Logical resource organization: tree
3D Model - The Trace Reader

- Deals directly with application traces and events
- Only trace-dependent part of the model
- Transform events into high-level visual objects
  - Container → Entities
  - State/Variable/Event → Evolution
  - Link → Communications
- No semantics → Visualization is generic
- Supply entity matcher needs: links and entities
- Attribute entities with location data
  - where a process is executed
  - which process a thread belongs to
- Input is also redirected to the Visualization module
3D Model - The Entity Matcher

- Responsible for the Visualization Base layout
- Three possibilities of configuration are proposed
  - Communication Pattern (deadlocks, ...)
  - Network Topology (network utilization, routes, ..)
  - Logical Organization (load balancing, ...)

```
Selected Entities by the Extractor → Entity Matcher → Comm. Pattern

Network Interconnection

Network Topology

Hierarchical Organization → Logical Organization
```

Diagram showing the relationships between selected entities, entity matcher, communication pattern, network topology, and logical organization.
3D Model - Visualization

- How the visual objects are represented in 3D
3D Model - Visualization

- How the visual objects are represented in 3D
- Rendering the visualization base
  - Application Communication Pattern
3D Model - Visualization

- How the visual objects are represented in 3D
- Rendering the visualization base
  - Application Communication Pattern
  - Network Topology + App. Communication Pattern

Flow of Visual objects from the Extractor

Network Topology + Communication Pattern generated by the Entity Matcher
3D Model - Visualization

- How the visual objects are represented in 3D
- Rendering the visualization base
  - Application Communication Pattern
  - Network Topology + App. Communication Pattern
  - Logical Organization of Resources
Outline
Aggregation Model - Overview

- Enable large-scale trace analysis
- Visually compare entities behavior
- Detect global and local characteristics

Steps of the Model

1. Hierarchical Monitoring Data
2. Time-Slice algorithm (temporal integration)
3. Aggregation model (spatial integration)
4. Treemap representation

Visualization differences from existing tools

- PlanetLab’s CoVisualize → resources
- Treemap for Workload Visualization [Stephen 2003]
- Lack of configurable time intervals, aggregated data
Hierarchical Monitoring Data

- Monitoring systems register entities behavior
- Entities can be processes and threads
- They can be organized as a hierarchy
  - Logical hierarchy
  - Geographical Location hierarchy
  - Other possibilities: libraries, components
- Grid’5000 example
Objective: annotate leaf nodes of the hierarchy

- Time-slice definition
- Summary of trace events on the interval
  - States, Variables, Links, Events, ...

Output of the Algorithm

- Hierarchy of input + computed values on leaves
Time-Slice Algorithm - Example

Blocked (seconds) Executing (seconds)

G

C1

M1

A (5)

B (2)

C (6)

D (0)

E (5)

C2

M2

M3

G

C1

M1

A (4)

B (7)

C (3)

D (9)

E (4)

C2

M2

M3
Time-Slice Algorithm - Example

(Blocked, Executing)

G

C1

M1

A

(5, 4)

C2

M2

B

(2, 7)

C

(6, 3)

M3

D

(0, 9)

E

(5, 4)
Aggregation Model

- Objective: **aggregated values** at intermediary levels

**Aggregation Functions**

- add, subtract, multiply, divide, max, min, median, ...
- Depends on
  - what type of value the leaves have
  - the desired statistical result
Aggregation Model

- **Objective:** aggregated values at intermediary levels

**Aggregation Functions**

- add, subtract, multiply, divide, max, min, median, ...
- Depends on
  - what type of value the leaves have
  - the desired statistical result

Diagram:
- Nodes: G, C1, C2, M1, M2, M3, P1, P2, P3, P4, P5
- Edges:
  - M1 to C1, value: (7, 11)
  - M2 to C2, value: (6, 12)
  - M3 to (5, 4)
  - P1 to M1, value: (5, 4)
  - P2 to M2, value: (2, 7)
  - P3 to P4, value: (6, 3)
  - P4 to M3, value: (0, 9)
  - P5 to M3, value: (5, 4)
  - Edge label: +
Aggregation Model

- **Objective**: aggregated values at intermediary levels

### Aggregation Functions

- add, subtract, multiply, divide, max, min, median, ...
- Depends on
  - what type of value the leaves have
  - the desired statistical result

```
G
  /\    /
 C1 (7, 11) + C2 (11, 16)
  /    /  
 M1 (7, 11) + M2 (6, 12) + M3 (5, 4)
  |    |    |    |
 P1 (5, 4) P2 (2, 7) P3 (6, 3) P4 (0, 9) P5 (5, 4)
```
Aggregation Model

- Objective: aggregated values at intermediary levels

Aggregation Functions
- add, subtract, multiply, divide, max, min, median, ...
- Depends on
  - what type of value the leaves have
  - the desired statistical result

Diagram:
- G: (18, 27)
- C1: (7, 11)
- C2: (11, 16)
- M1: (7, 11)
- M2: (6, 12)
- M3: (5, 4)
- P1: (5, 4)
- P2: (2, 7)
- P3: (6, 3)
- P4: (0, 9)
- P5: (5, 4)
Visualization of the Approach - Treemaps

- Technique created in 1991
- Scalable hierarchical representation
- Algorithm
  - Top-down drawing
  - For a given node, split screen space among children

Original algorithm has several evolutions

- Squarified treemap is used here
  - Keeps rectangles as close to squares as possible
Treemap to view the Aggregated Hierarchy

G (18, 27)
  ─── C1 (7, 11)
  ─── C2 (11, 16)
    ├── M1 (7, 11)
    │    ├── P1
    │    │    │    (5, 4)
    │    │    └── P2
    │    └── M2 (6, 12)
    └── M3 (5, 4)
        ├── P3
        │    │    (2, 7)
        │    └── P4
        └── P5
            │    (6, 3)
            └── P5
                (0, 9)
                (5, 4)
Treemap to view the Aggregated Hierarchy

G (18, 27)

C1 (7, 11)  C2 (11, 16)

M1 (7, 11)  M2 (6, 12)  M3 (5, 4)

P1  P2  P3  P4  P5

(5, 4) (2, 7) (6, 3) (0, 9) (5, 4)
Treemap to view the Aggregated Hierarchy

G (18, 27)
  C1 (7, 11)
    M1 (7, 11)
    P1 (5, 4)
    P2 (2, 7)
  C2 (11, 16)
    M2 (6, 12)
    P3 (6, 3)
    P4 (0, 9)
    P5 (5, 4)

Cluster

B
E

C2
Treemap to view the Aggregated Hierarchy

![Treemap Diagram]

- **G**: (18, 27)
- **C1**: (7, 11)
- **C2**: (11, 16)
  - **M1**: (7, 11)
  - **M2**: (6, 12)
  - **M3**: (5, 4)
    - **P1**: (5, 4)
    - **P2**: (2, 7)
    - **P3**: (6, 3)
    - **P4**: (0, 9)
    - **P5**: (5, 4)
Triva Prototype Implementation

- Developed in Objective-C and C++
- Combine several existing tools
  - DIMVisual library
  - Pajé Components (the Simulator)
  - Graphviz, Ogre3D, wxWidgets
- Performance evaluation of Pajé
  - Able to handle large-scale traces
  - Small response-time
  - Memory limitations
DIMVisualReader - Trace Reader

- Built-in instrumentation of KAAPI library

- MPIRastro wrapper for MPI applications
TrivaView - The 3D Approach

- Model: Extractor, Entity Matcher & Visualization
- Interaction Techniques (Ambient, CameraManager)
- Base configuration
  - Application Comm. Pattern created with GraphViz
  - Network Topology description (dot format)
  - Logical Organization (plist format)
- Placement on the Visualization Base
- Rendering the 3D Timestamped Pajé Objects
TimeSliceView - The Aggregation Model

- Only two components
  - TimeSlice Filter
  - Triva2DFrame

- Time-Slice Algorithm and Aggregation Model
- Implementation of the Squarified Treemap Algorithm
- Drawing the rectangles with the wxWidgets
Results

- Different application traces are used as input
- Results are composed of screenshots of the prototype

Objective

- Check if 3D visualizations enable a better understanding of traces with the network topology
- Check if large-scale analysis are possible with the aggregation model

- Traces Description
- 3D Visualization
- Treemap Visualization
Results - Trace Description

- Synthetic traces
  - Large-scale hierarchies (up to 100 thousand)
  - Typical Communication Patterns

- Real traces
  - KAAPI Traces
  - MPI Traces
  - Grid’5000 platform in France
  - Xiru Cluster at Porto Alegre
3D Visualization - Communication Patterns

- Differences from the space-time diagram
  → 3D enables Graph-like representations
  → with time evolution

Ring Communication Pattern

Fully-Connected

Star
3D Visualization - KAAPI Trace

- Fibonacci Application
- 26 processes, two sites, two clusters
- Lines represent steal requests
- Different number of communication between clusters
  - beginning → big tasks, less communication
  - end → smaller tasks, more communication
60 processes, two sites, three clusters
Total execution time of a KAAPI fibonacci application
Observe number of requests in time

Rennes Nancy
Paraquad (25)
Paramount (5)
Grelon (30)
Rennes
Router
Nancy
Router
Paramount (5)
Grelon (30)
Paraquad (25)
Rennes
Router
Nancy
Router
More WS Requests
Less WS Requests
3D Visualization - KAAPI Trace

- 200 processes, 200 machines, two sites, five clusters
- Annotated manually with bandwidth limitations

Initial Execution of Application with Link Properties

Interconnection becomes bottleneck, possible hints to better allocation

Too many WS Requests on low bandwidth Link
2900 processes, four sites, thirteen clusters

End of Execution
Interaction Techniques: mouse wheel, mouse over

Detailed information is available in the status bar
Run and RSteal states, 2900 processes, 310 processors
- Synthetic trace with 100 thousand processes
- Two states, four-level hierarchy
400 processes, 50 machines, one site
- 8 processes per machine
  - Overload of some machines with 2 CPUs
  - Unusual amount of time in Steal state
- Machines with 4 CPUs show normal behavior
- 188 processes, 188 machines, five sites
- Different behavior at Porto Alegre
- Probably due to the interconnection
  - Latency for Grid’5000 in France: ~10 ms
  - Latency between Porto Alegre and France: ~300 ms
- More time spent in work stealing functions
Treemap Visualization - MPI Trace

- Traces from the EP application – NAS Benchmark
- 32 processes – time spent in each MPI operation
- Init/Barrier: might indicate a linear implementation

A. With States Running, MPI_Init, MPI_Barrier and MPI_AllReduce

B. Only MPI_INIT state

C. Only MPI_BARRIER state

Maximum Aggregation

Only Process Rank 21

~4.5 s
~0.9 s
~5.7 s
~0.3 s
Conclusions

The problem identified in the Thesis
- Lack of structural visualization analysis
- Visualization scalability

Main Achievements
- Behavioral with Structural/Statistical Model (3D)
  - Analysis considering network structure
  - Experiments using Grid’5000 platform
  - Identification of behavior in KAAPI work stealing
- Time-Slice Technique & Aggregation Model
  - Validated with real-scenario with 2900 processes
  - Tested with synthetic traces up to 100K processes
  - Load-balance efficiency / global and local summaries
Perspectives and Implications

■ Perspectives
  ■ Show aggregated objects in the 3D visualization
  ■ Other types of information for the time-slice technique
  ■ Use of other aggregation functions
  ■ Aggregation model to merge communication patterns
  ■ ...

■ Implications
  ■ Better understanding of parallel applications
    → consider execution environment details
    → large-scale visual analysis
  ■ Re-thinking behavioral visualization
    → Do we need a timeline in representations?
    → Aggregated data
  ■ Use of information visualization techniques
Acknowledgements

- Thesis financed with scholarships by
  - CAPES and CNPq
  - CAPES/Cofecub – Project 4602/06-4

- Thanks to
  - Advisors: Navaux, Guillaume and Denis
  - Family and friends