Multi-scale Analysis of Large Distributed Computing Systems

Lucas M. Schnorr, Arnaud Legrand, Jean-Marc Vincent
INRIA Mescal, CNRS LIG
Grenoble, France

LSAP'2011 Workshop
San Jose, CA

June 8th, 2011

Introduction

Distributed systems to monitoring tools

- Large scale distributed systems
  - Heterogeneous & Shared
  - Examples: grids, clouds, volunteer systems
  - **Composed of several thousands components**
  - Context: observation and analysis
- Monitoring tools
  - Provide high-level statistics
  - Basic resource usage traces through time
  - Examples: Ganglia, NWS, Monalisa
  - **Scalable techniques**, but often lack the details to
    - understand unexpected behavior
    - correlate application behavior to resource utilization

Introduction

Tracing techniques to large-scale distributed systems?

- Tracing techniques (from parallel computing)
  - Precise and fine grain application-level events
  - Complex behavior patterns can be detected
  - Interactive Analysis with visualization tools
  - Examples: TAU, Vampir, Jumpshot, MPE
- Some challenges remain
  - visualization scalability
  - intrusion control
  - large trace files in large-scale scenarios

Question

- Can we use tracing techniques to collect precise and fine grain in large-scale distributed systems?

Approach

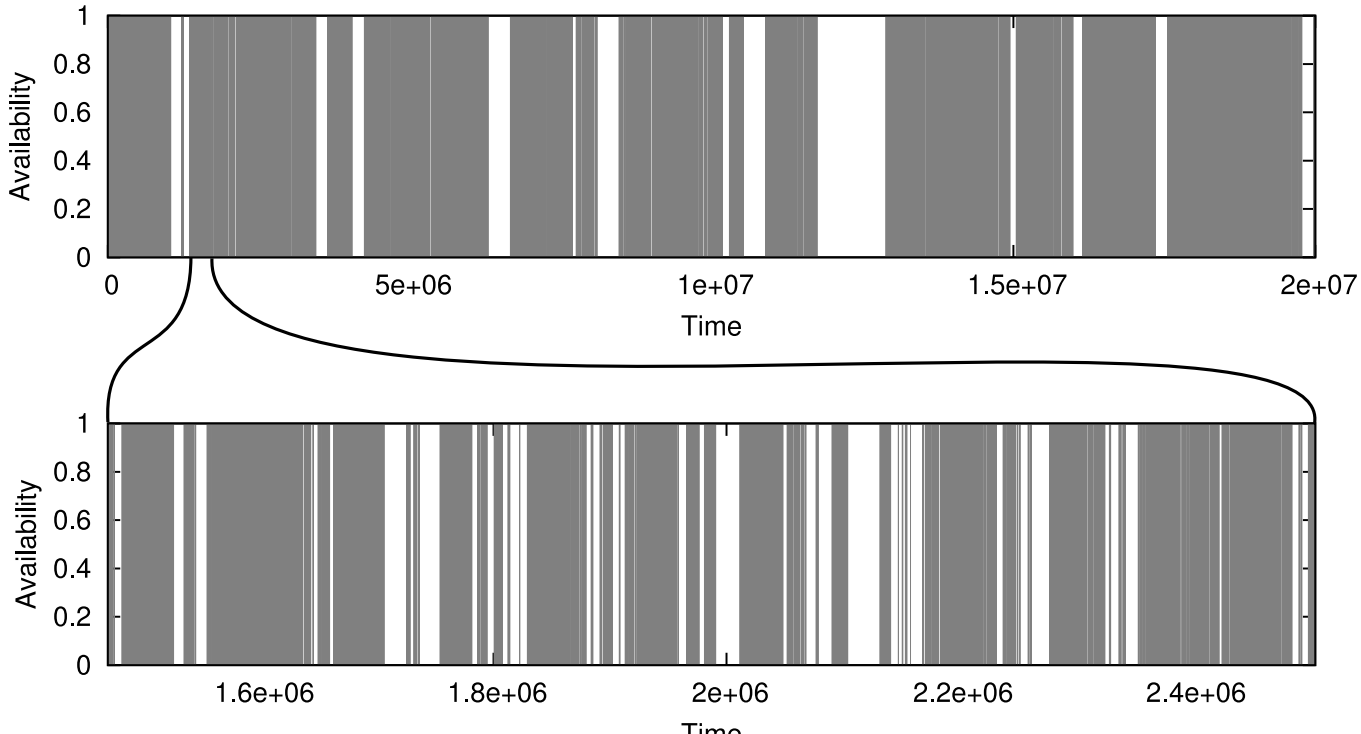
Multi-scale analysis of detailed traces

- Aggregation technique applied to
  - **time**: long periods of observation
  - **space**: a considerable amount of observed entities
- Analysis through data visualization
- Identify and understand non-trivial behavior
- Applied to BOINC client resource analysis

Outline

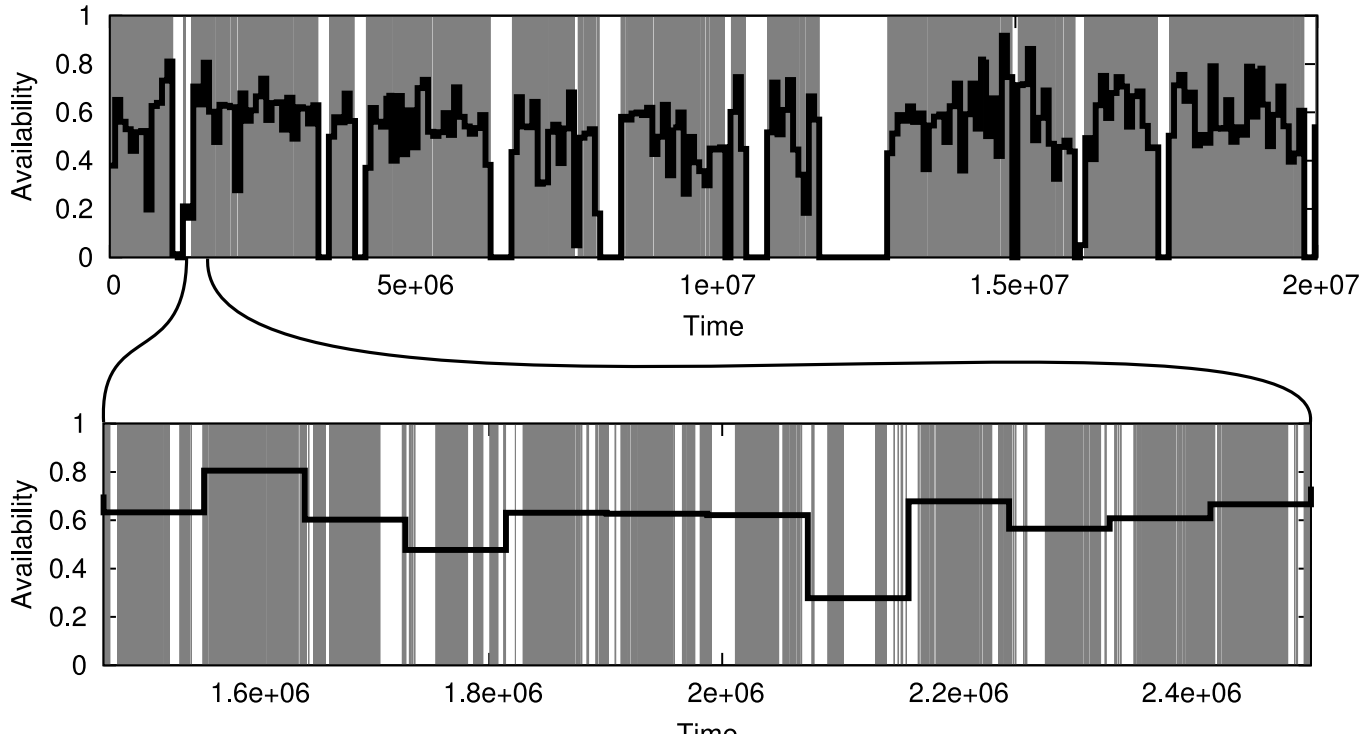
Analysis approach

- Detailed observation over long periods of time
  - Results in large data sets to be analyzed
  - Difficult to extract useful information
- Example: BOINC availability traces
  - One volunteer during an eigth-month period
  - Graphical zoom in an arbitrary twelve-day period



Analysis approach: aggregation

- Using temporal aggregation
  - Integrate using a time frame defined by the analyst
  - Analyze integrated data instead of raw traces
- Example: BOINC availability traces
  - one-hour integration
  - easier to understand patterns



Analysis approach: space dimension

- Large number of resources
  - Spatial integration of their behavior
  - Neighborhood definition

Framework: Distributed Application Scenario

- Scheduling of bag-of-tasks in volunteer computing
- BOINC architecture used as example
  - Several projects, each with a number of tasks
  - Volunteers decide to which project work for

1. Fair sharing
   - Volunteers define how much to work for each project
   - Check if a fair share is locally and globally achieved
2. Response time
   - BOINC not designed to give good response time
   - How it behaves if projects have bursts of small tasks

Framework: BOINC Simulator (with SimGrid)

- Raw traces are needed to validate our approach
- Real world large-scale platforms
  - Hard to measure, time consuming
  - Most of traces are already reduced in time
  - Sometimes also in space
- **Simulation using the BOINC Simulator**
  - Already validated against the real BOINC client
  - Supports the most important features of the real one
    - deadline scheduling, long term debt
    - fair sharing, exponential back-off
  - Considers real availability traces from the FTA
- SimGrid toolkit
  - Fast simulation of several thousands hosts
  - Able to trace categorized resource utilization

Framework: Triva Visualization Tool

- Implements the time & space data aggregation
- Analyst interaction
  - Time frame to be considered
  - Select/Filter a set of resources
  - Animation of variables through time
- Two visualization techniques
  - Treemap view: addresses scalability
  - Topology-based view: network correlation

Results and Analysis

Overview

- Two scenarios
  1. Fair sharing
  2. Response time
- For each scenario
  - **Setting** the scenario, initial configuration
  - **Expected** behavior based on previous knowledge
  - **Observed** results and visualization analysis

Fair Sharing – Setting

- Server side
  - Two BOINC projects servers with continuous jobs

Project	Task Size	Deadline
Continuous-0	30	300
Continuous-1	1	15

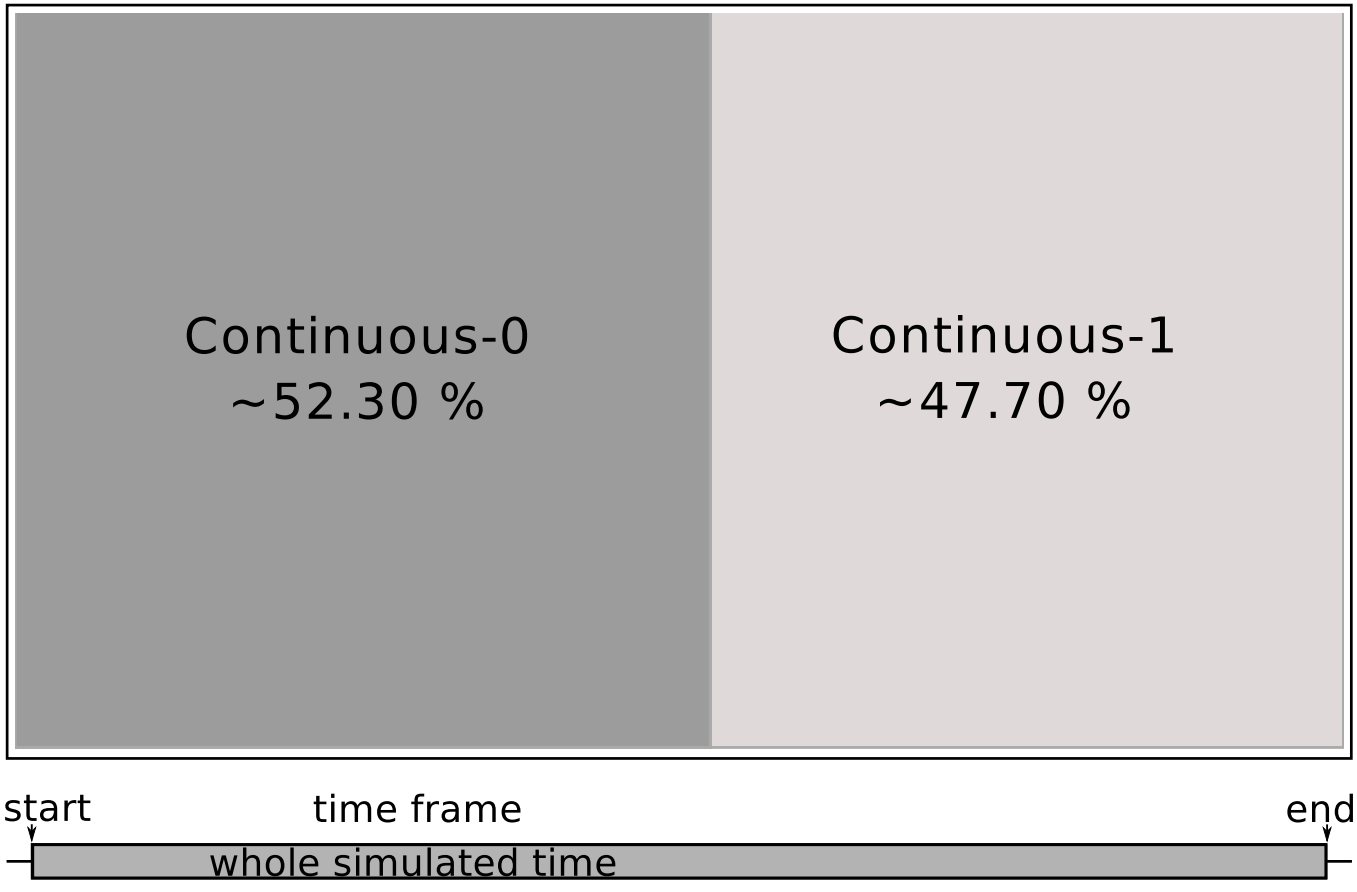
- Volunteer side
  - Number of volunteers: 65
  - Evenly share its resource
- Simulation
  - Simulated time of ten weeks
  - Volunteer hosts power and availability from FTA

Fair Sharing – Expected

- Long term period
- Equal global and local share
  - 50% for Continuous-0
  - 50% for Continuous-1

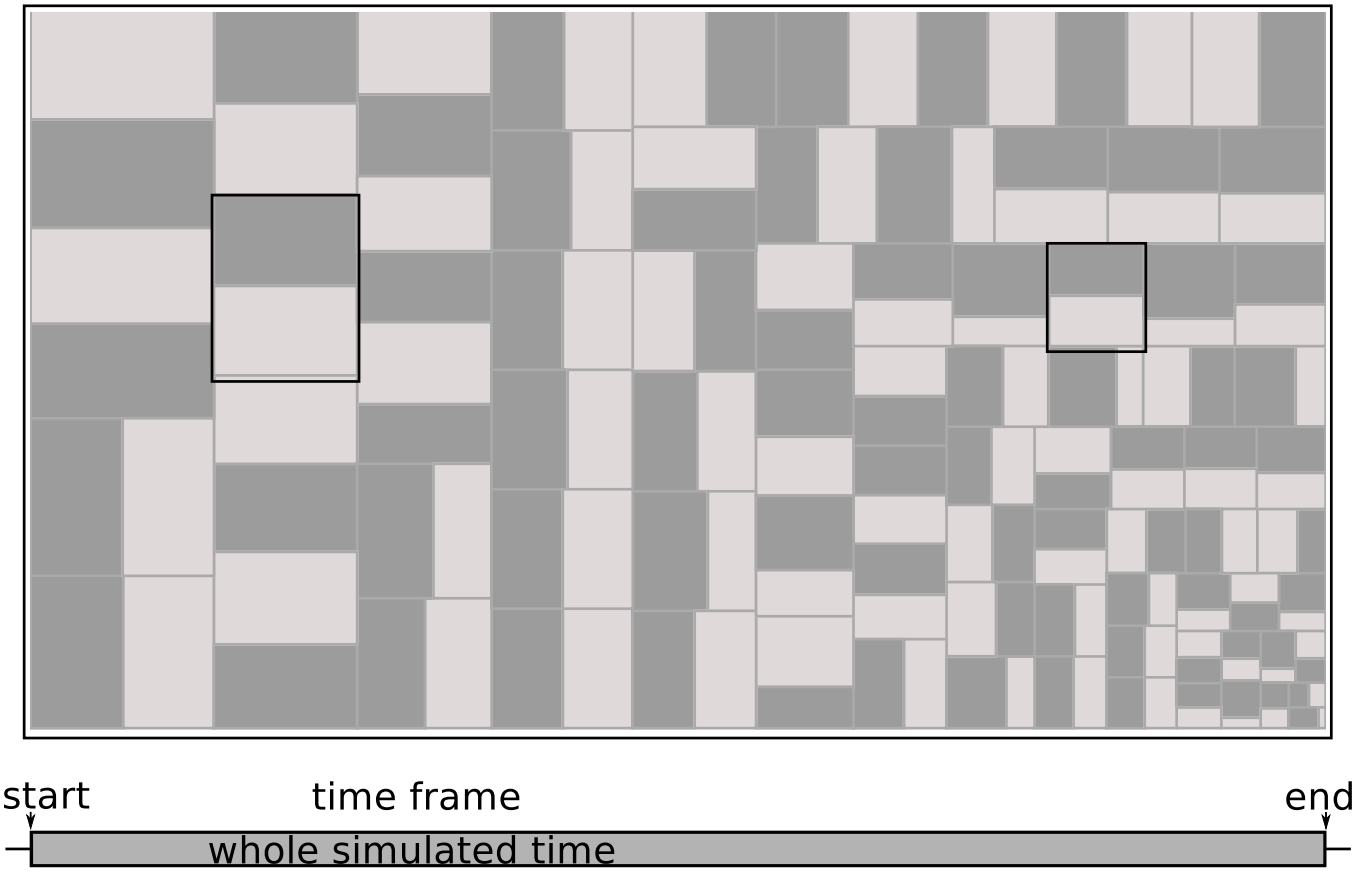
Fair Sharing – Observed 1/3

- Aggregation: whole time – all clients
- Reasonable fair



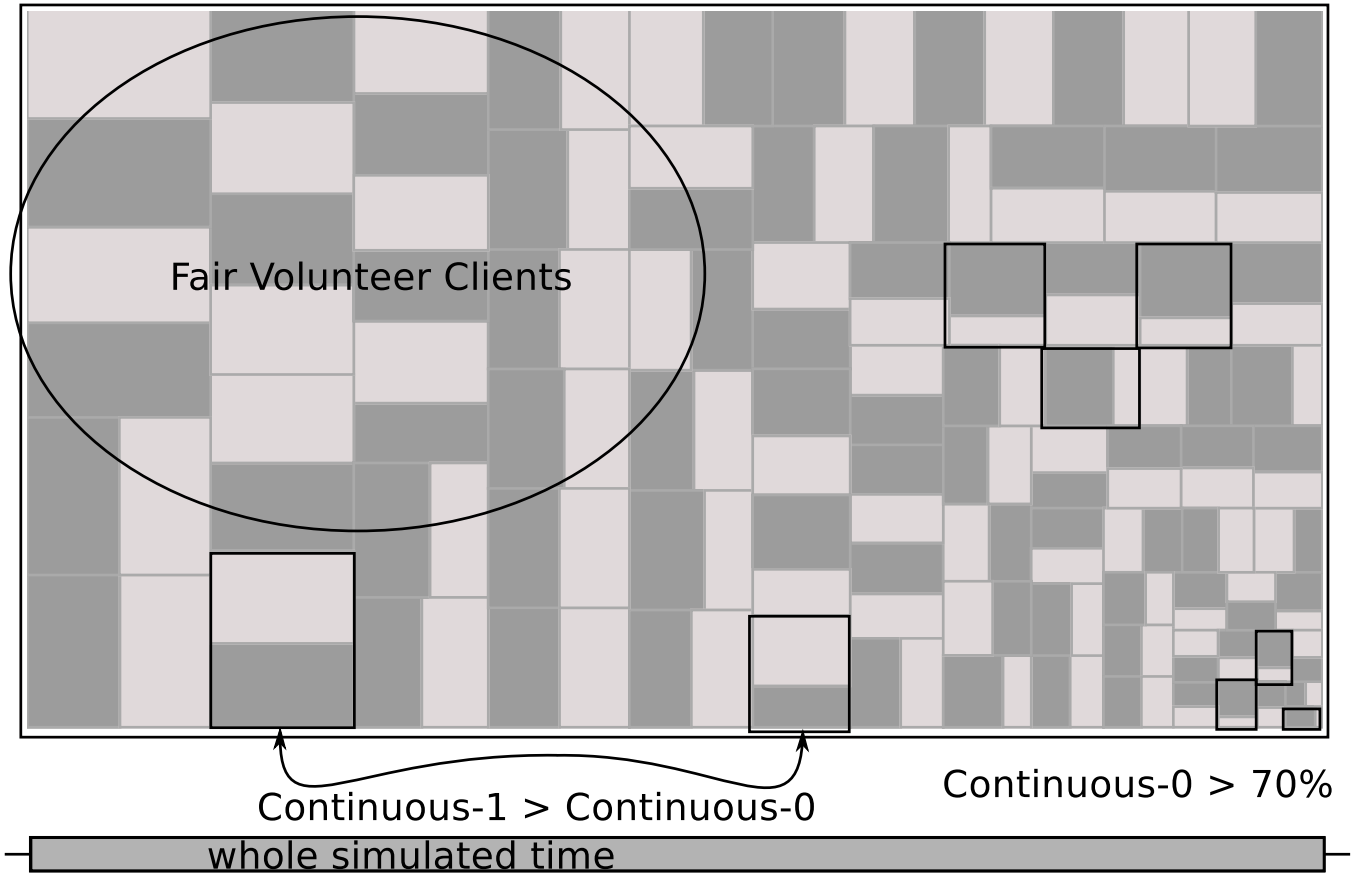
Fair Sharing – Observed 2/3

- Aggregation: whole time – per-client view
- Size occupied by a volunteer indicates its donation



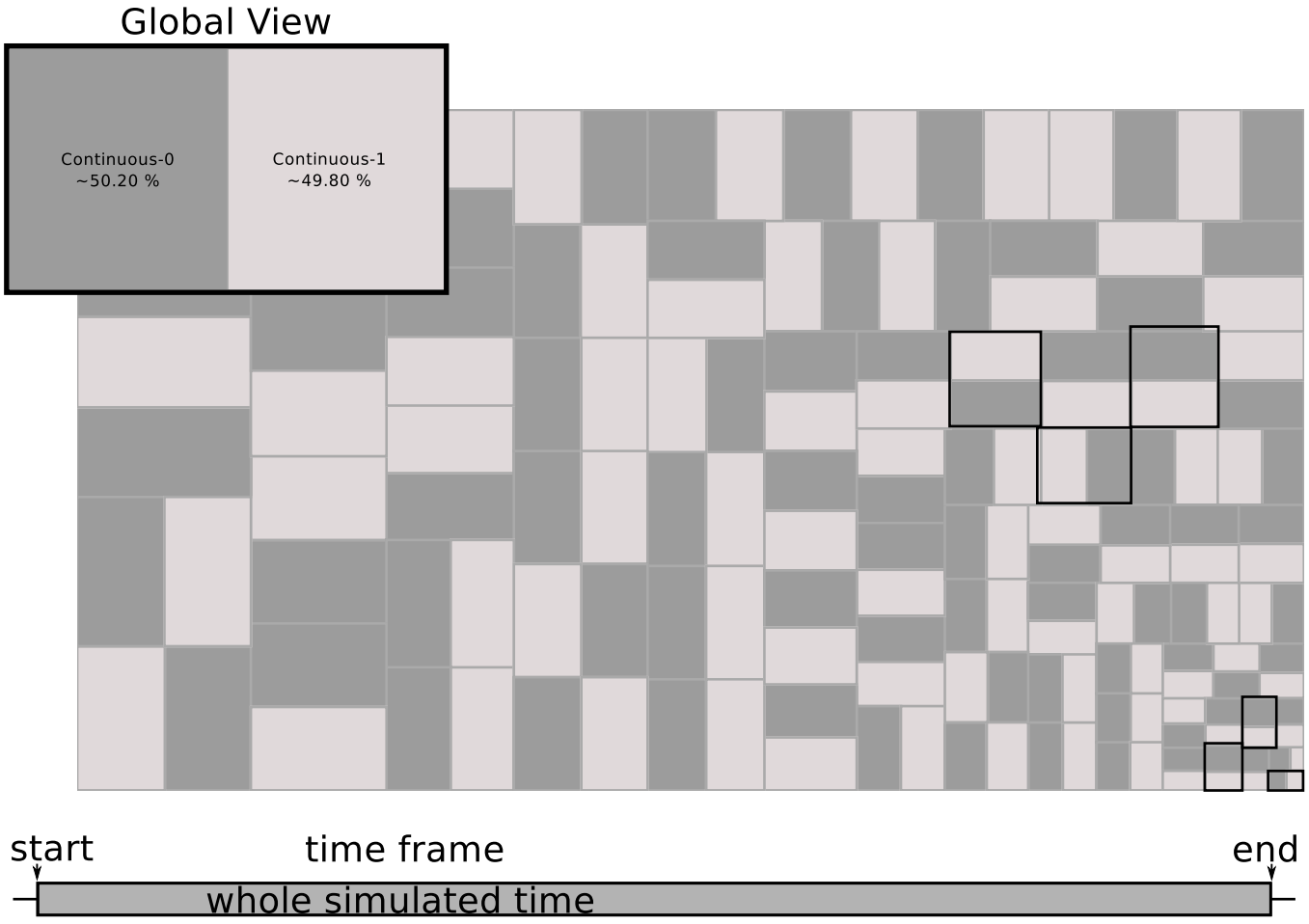
Fair Sharing – Observed 3/3

- Anomalies detected on fair sharing mechanism
- Correlation with small volunteer contribution



Fair Sharing – Final

- Early development of BOINC simulator
  - Counting of time worked for each project
  - Bug fixed



Response Time

Overview

- BOINC is for CPU-bound throughput computing
- Jobs in BOINC have a deadline attribute
- Volunteer clients try to respect the deadline
  - Earliest deadline first mode
- Projects interested in response time
  - Play with the job deadline parameter
  - May gain short-term priority over other projects

Response Time – Setting

- Server side
  - Two BOINC projects servers / different job policies

Project	Task Size	Deadline	Period	Replicas
Continuous	30	300	always	1
Burst	1	6	10 days	5

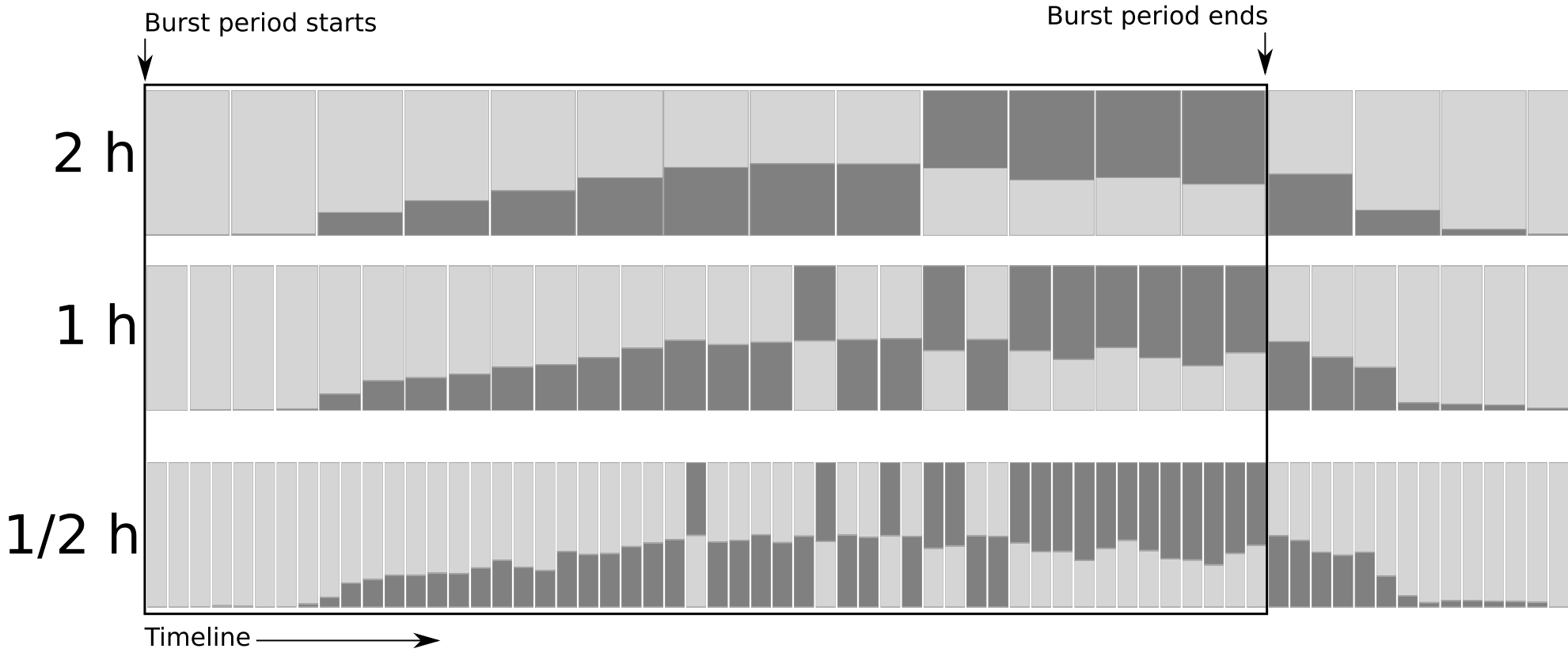
- Volunteer side
  - Number of volunteers: 65
  - Evenly share its resource, respecting deadlines
- Simulation
  - Simulated time of ten weeks
  - Volunteer hosts power and availability from FTA

Response Time – Expected

- BOINC has a pull architecture
  - Volunteers ask jobs to the project servers
  - Projects wait for volunteers to distribute jobs
- Slow-start effect for jobs that belong to burst project
- Higher priority to burst jobs
- Observe the shape of the effect in the visualization

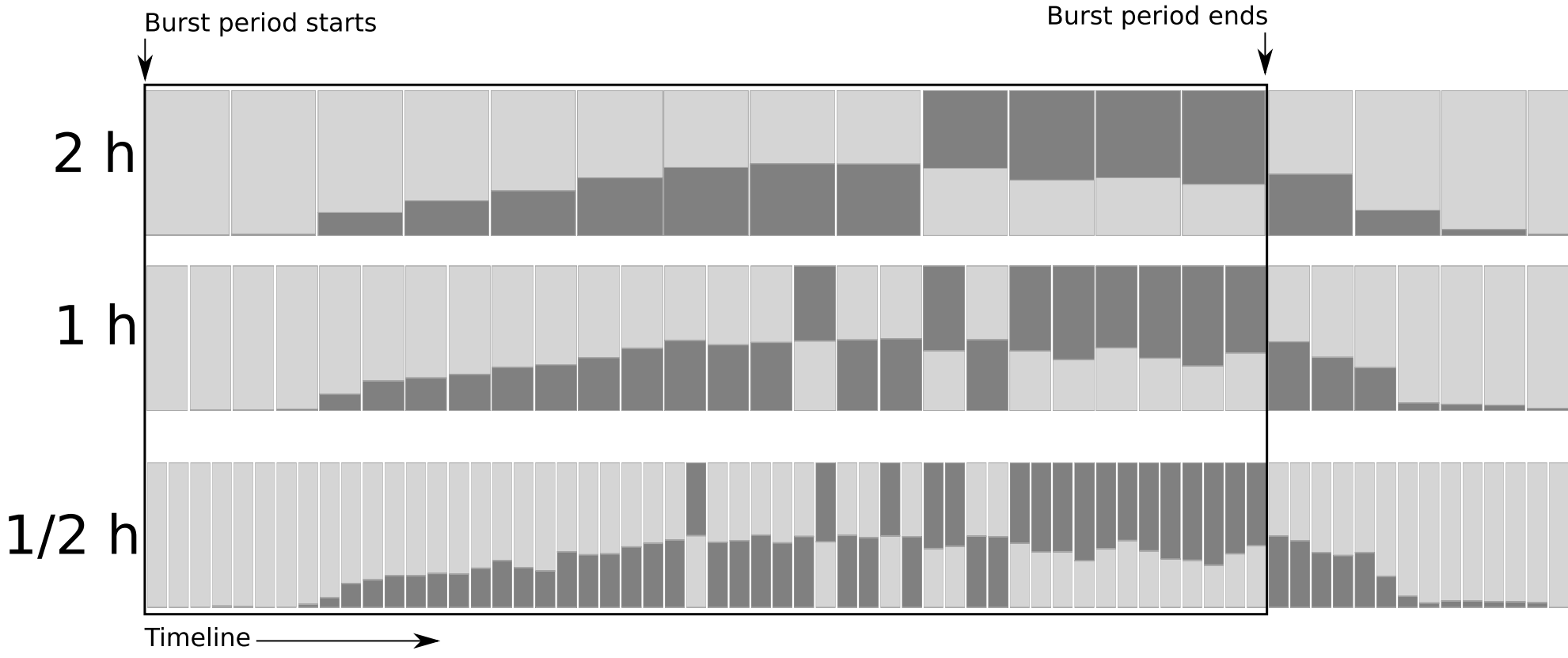
Response Time – Observed 1/2

- Aggregated work for all volunteers
  - Continuous (light gray)
  - Burst (dark gray)



Response Time – Observed 2/2

- Anomalies
  - Wasted computations (replica/deadline parameters)
  - Low priority of the Burst project (dark gray)

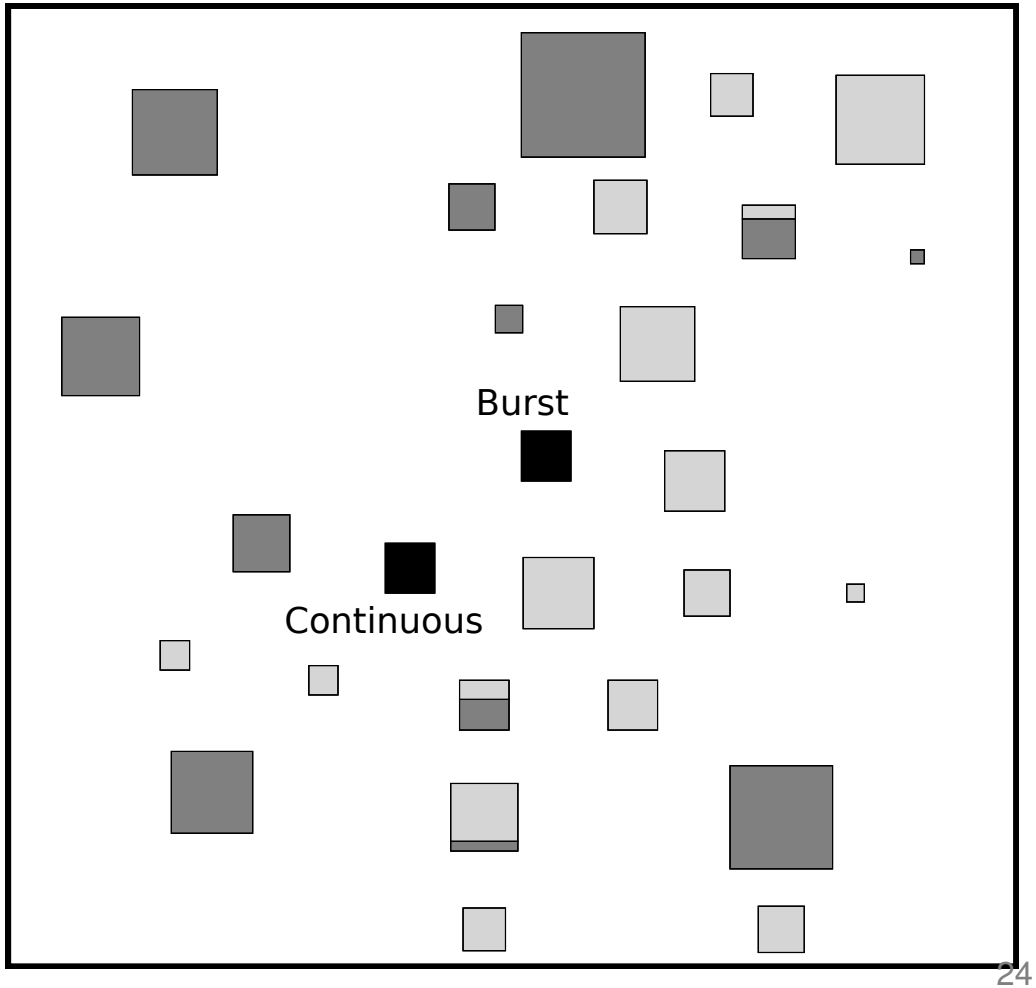


Response Time – Further Investigation 1/2

Graph-based view

- Each host is represented by a square
- Size represents CPU-power of volunteers

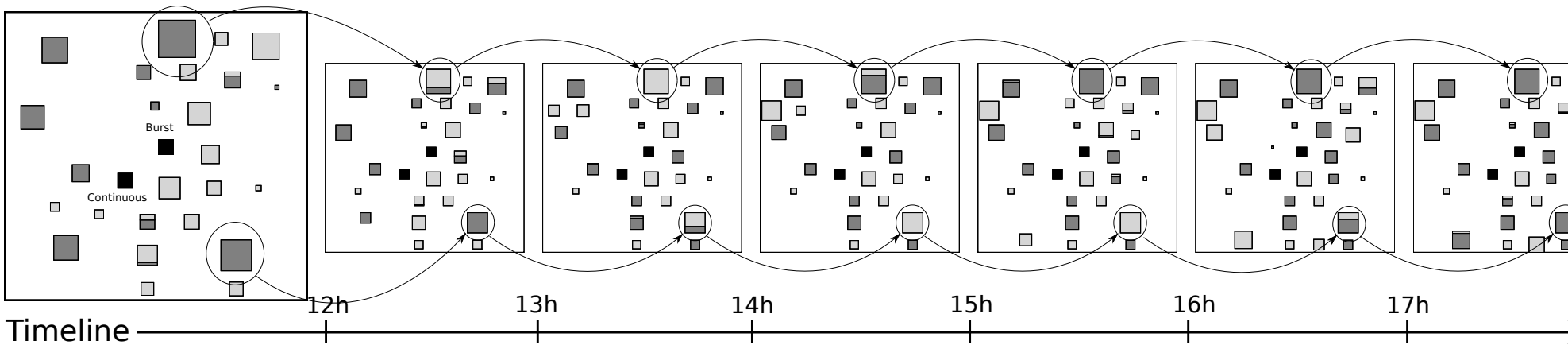
- The two servers
  - white if inactive
  - black if active
- Volunteers
  - burst: dark gray
  - continuous: light gray
- One-hour aggregated



Response Time – Further Investigation 2/2

Graph-based view

- Behavior of each volunteer in time
- Each screenshot – one-hour temporal aggregation



- Cyclic behavior in all volunteers
- Reason: short time fairness of BOINC

Final Remarks

- Monitoring systems are common
  - Scalable techniques
  - Difficult to perform a profound analysis
- This paper
  - Detailed traces on large-scale distributed systems
  - Elaborated visualization techniques for analysis
- Main contributions
  - Identification of anomalies and unexpected behavior
    - Fair sharing issue - BOINC simulator fixed
    - Identification of wasted computations
    - Suprinsingly low priority of response time projects
  - Study of response time
    - Shape of slow-start
    - BOINC's short time fairness is negative

Thank you

- Simgrid Framework – http://simgrid.gforge.inria.fr
- Triva Visualization Tool – http://triva.gforge.inria.fr

- Questions?