Performance Monitoring and Analysis System for MUSCLE-based Applications

W. Funika, M. Janczykowski, K. Jopek,
M. Dudek, and M. Grzegorczyk
Institute of Computer Science AGH,
al. Mickiewicza 30, 30-059 Krakow, Poland

Outline

- Motivation
- Research goals
- Overview of proposed solution
- System architecture
- Implementation details
- Performance issues
- Case study
- Conclusions and future work

Motivation

- The design and simulation of multi-scale systems are crucial for different branches of science,
- Easing user's interactions with the monitoring system, turning them into a kind of user-friendly collaboration with the system,
- Ontologies make possible to change with little effort the focus and granularity of performance analysis as well as to support the reasoning on performance flaws,
- Flexible semantic-based description allows to facilitate adapting the monitoring tool to a monitored system

Research goals

- Creation of a set of ontologies covering MUSCLE-bound monitored resources
- Visualisation of application behavior and resources' usage at run-time
- Making possible to investigate the dependencies between various measurements at different levels of abstraction

Research goals (cont'd)

- Need of gathering data on the MUSCLE system, at the lowest possible cost, at different granularity and with different monitoring data suppliers:
  - using Nagios to monitor resources usage,
  - using SemMon to provide the user - who is carrying out the experiment - with a complex view on experiment's progress,
- The relevant stored data is used to analyze the status of a running application:
  - facilitating the work of the programmer

Overview of proposed solution

- Monitoring tool for MUSCLE's applications
- Extending SemMon tool features by low-level monitoring data coming from Nagios;
- Further, coming to features of:
  - MUSCLE,
  - SemMon,
  - Nagios

MUSCLE

- The Multi-Scale Coupling Library and Environment;
- a platform independent agent system to couple multi-scale simulations/experiments;
- communication is based on the actor-based concurrency model;

- implementation uses Java Agent DEvelopment framework - JADE
- provides an ability to describe a multi-scale application (experiment) as a set of connected single-scale modules
- provides a software framework to build experiments according to the finite cellular automata theory

SemMon

- an agent-based, high-level monitoring tool which takes advantage of semantic description of monitored resources exploited for distributed computations;
- provides a model for the information to be collected and enables the correlating of measurements coming from multiple distributed monitoring data sources

---

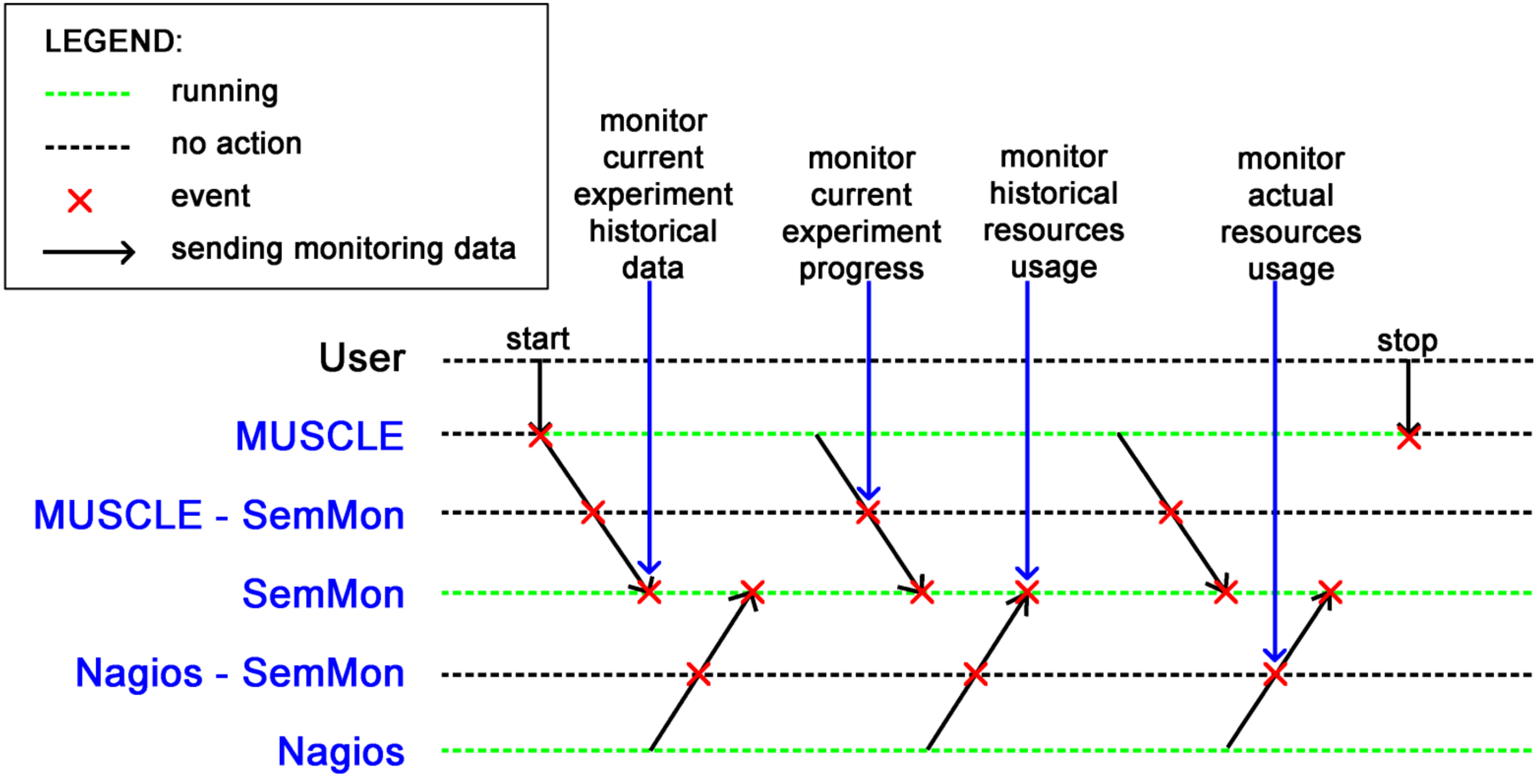
- Integration of various low-level monitoring tools via specialized adapters;
- SemMon is a distributed tool – core, GUI module, and reasoners are capable of working on different hosts owing to the communication mechanisms involved, e.g. RMI and JMX

Nagios

- a mature, full-featured, low-level monitoring system;
- extensible tool

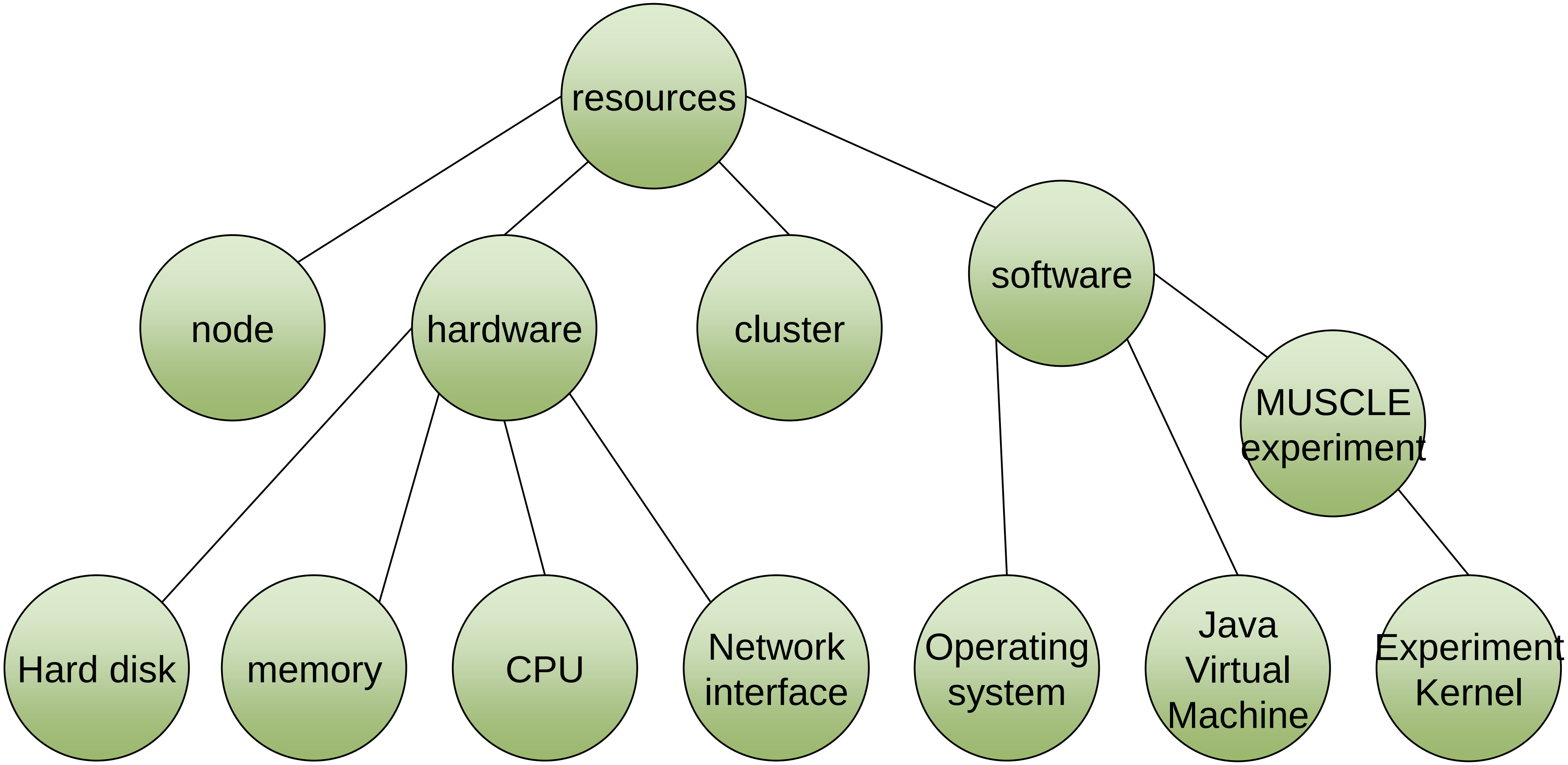
- the architecture of Nagios allows connecting to the SemMon tool via BSD sockets;
- obtains low-level system data, e.g. on resources usage

Sequence diagram of monitoring actors' interactions



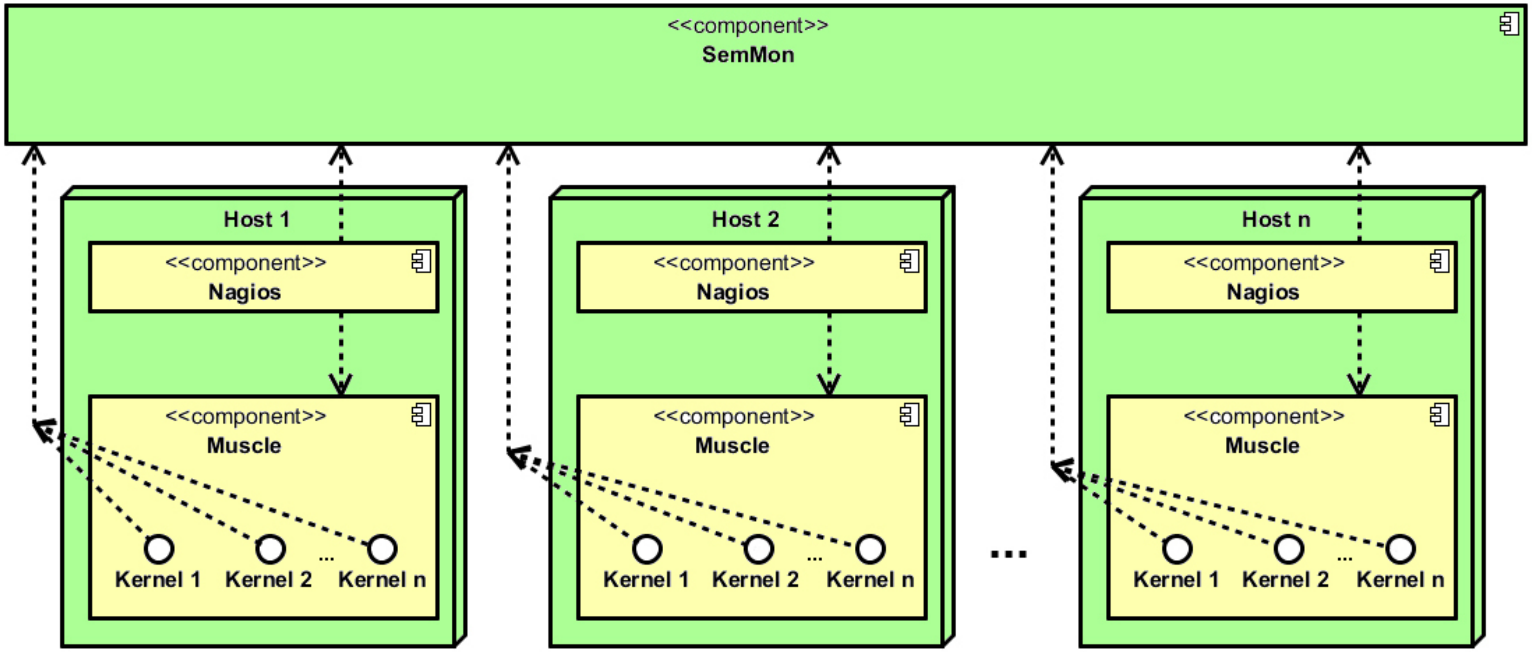
Ontology of resource classes of MUSCLE-based applications

- design solution involves specifying a semantic description of the application's elements and related ones



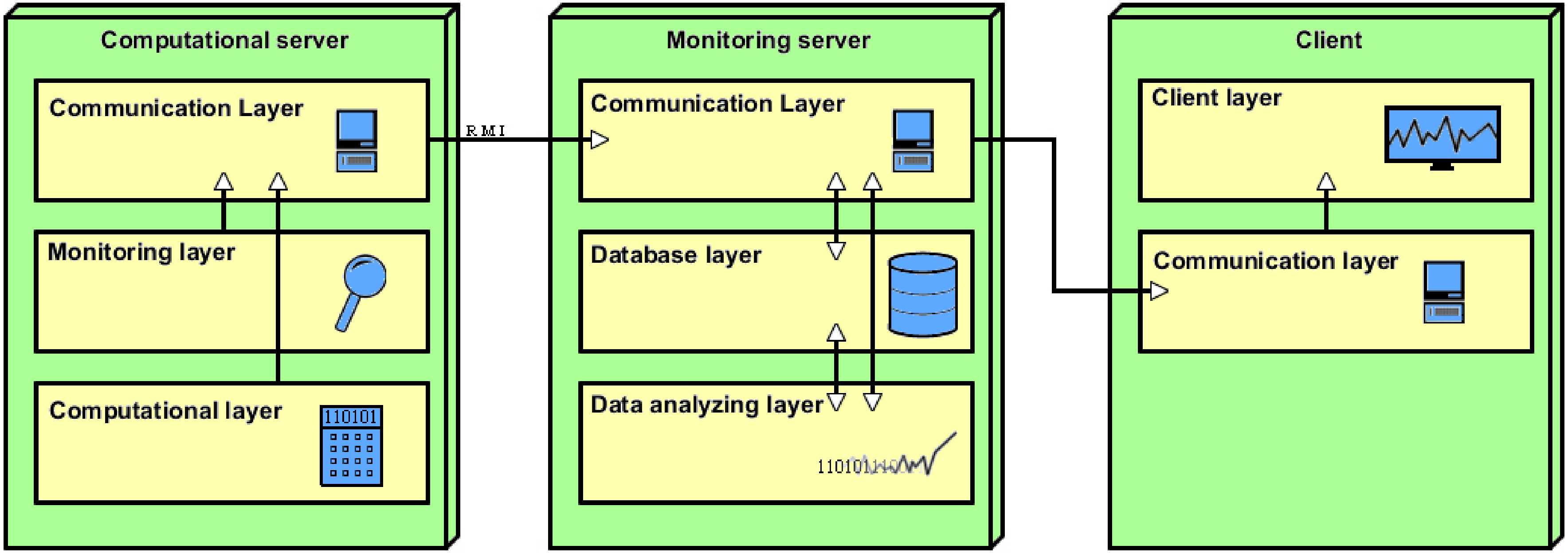
System architecture

- SemMon is a top component of the architecture:
  - obtains low-level monitoring data from Nagios;
  - traces the communication between MUSCLE kernels;
  - receives current kernel state (if kernel is computing, waiting for message or preparing a new message);
- Experiment is being computed in MUSCLE's kernels.
- *Overall architecture of MUSCLE-based application monitoring:*



System architecture in layered form

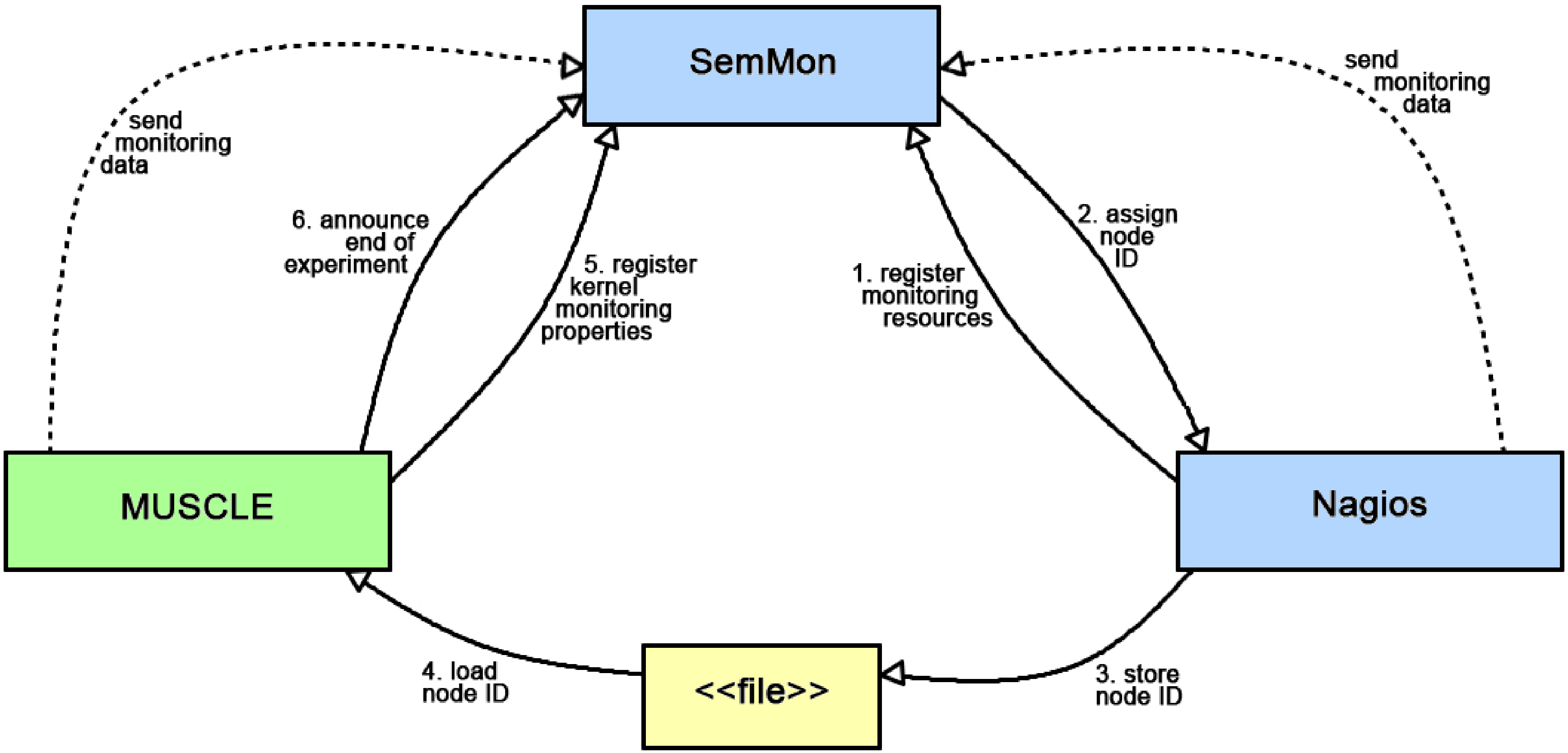
- Computational Server:
  - Computational layer – MUSCLE computing its experiments,
  - Monitoring layer – both MUSCLE's and Nagios' monitoring plugin,
- Monitoring server:
  - SemMon server devided into layers,
- Client:
  - Java GUI client app or webbrowser which communicates with monitoring server



Implementation details

- The protocol designed for monitoring purposes - mainly based on an XML set of rules;
- Communication between Nagios and SemMon is implemented using BSD sockets;
- Communication between MUSCLE and SemMon is resolved by the Remote Method Invocation mechanism;
- Strings are being sent with a standardized Java format: a stream of chars is preceded by two bytes encoding the total length of string in the big-endian order;
- Java-to-Java communication uses a conventional method invocation with arguments as strings.

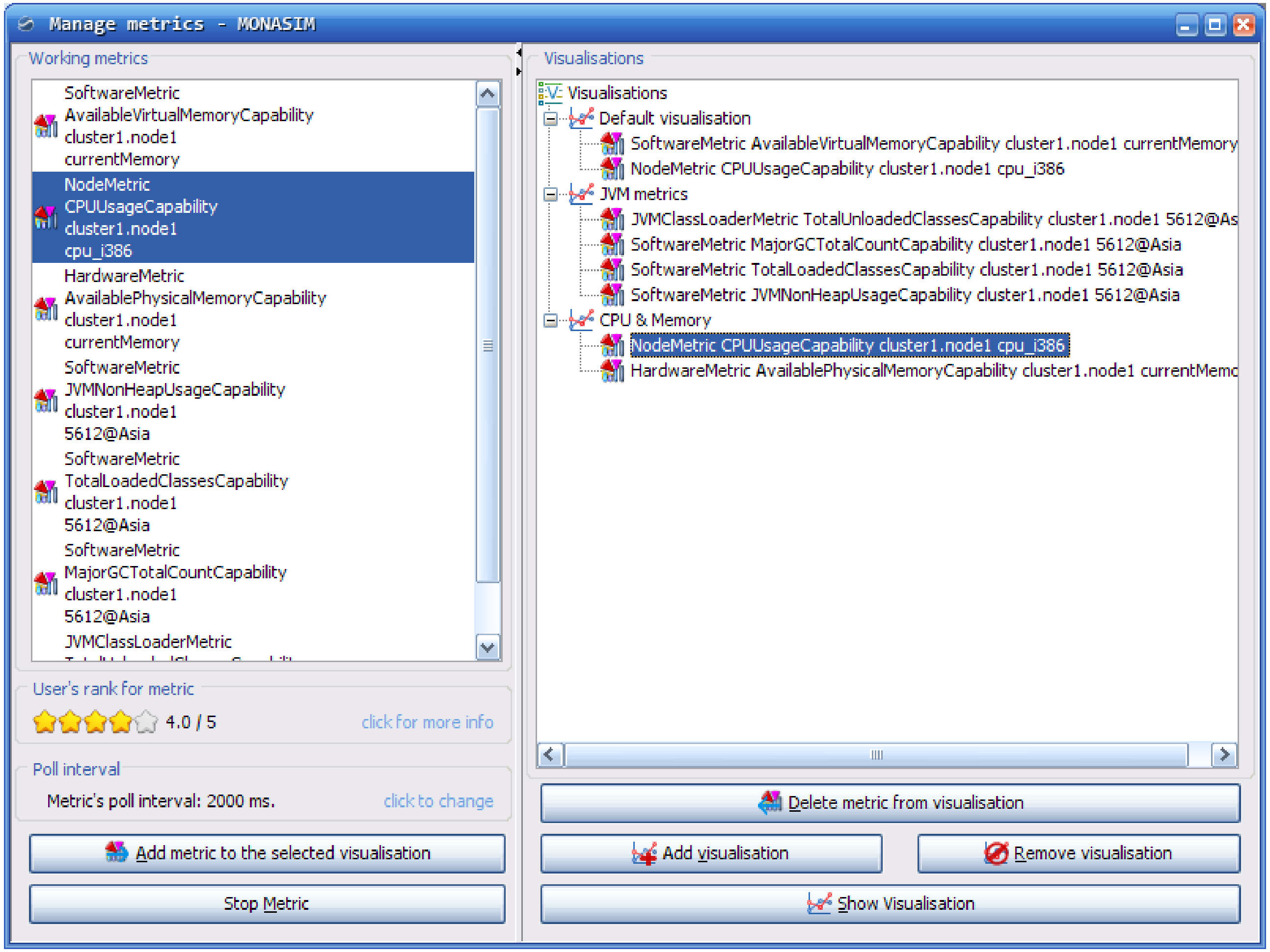
Communication model for MUSCLE-based application monitoring:



Visualization

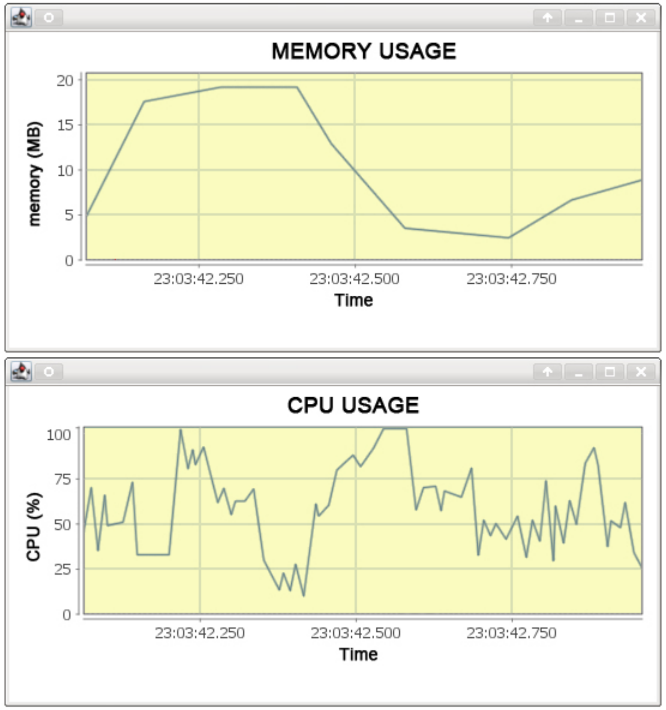
- Data visualization component allows the user to define not only a view related to a simple hardware metric, but also to create specialized metrics covering more complex characteristics like the execution of experiment;
- GUI enables choosing a needed metric and tune a relevant display to visualize performance analysis results;
- Visualization chart of the kernel’s activity in the form of an extended space-time display and its integration into the SemMon tool

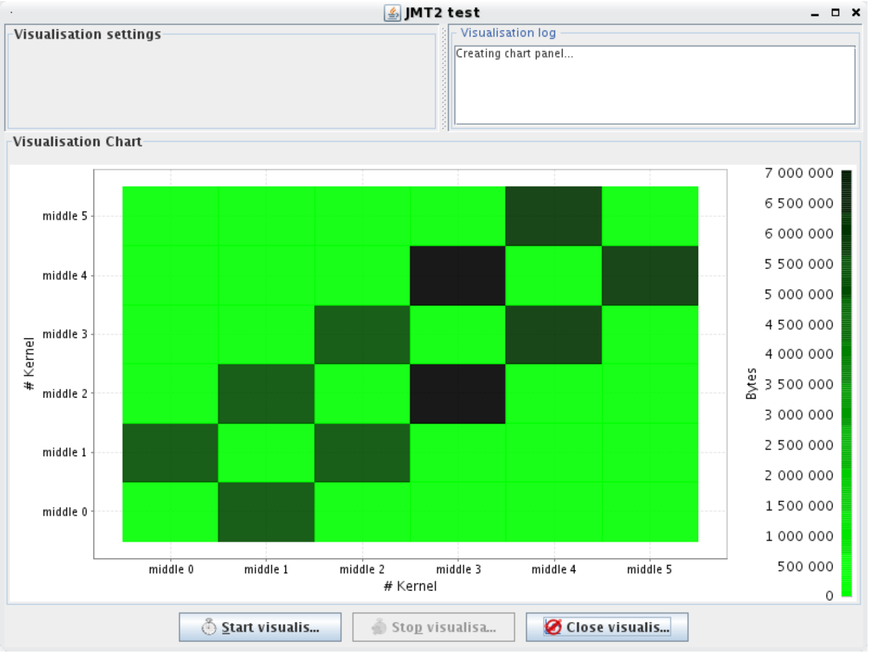
Measurement and visualization management



Visualization

(resources usage and communication matrix display)



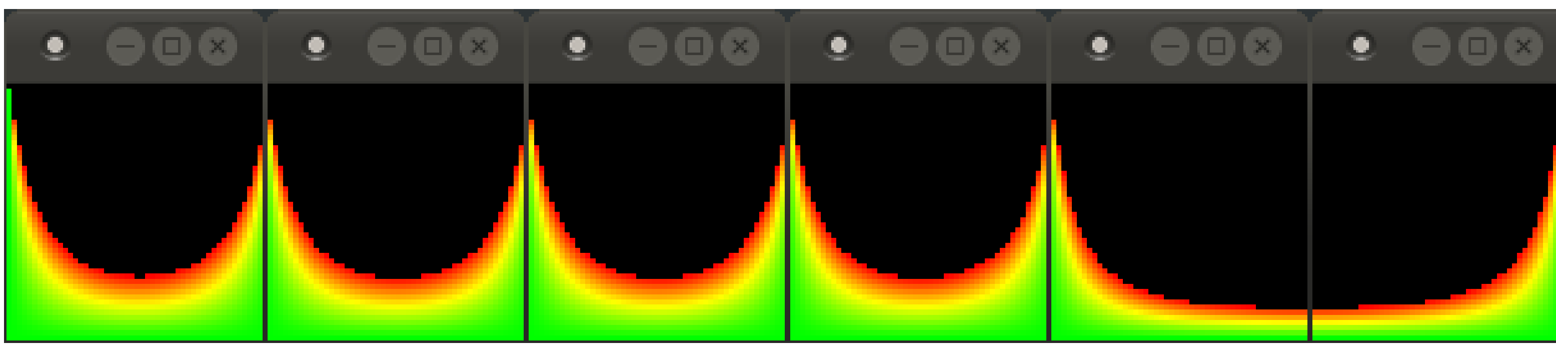


Performance issues

- Without checking the real size data, the whole execution time (proper execution plus monitoring) grew from 6.3% to 8.1%;
- With small messages monitoring costs are similar – the overhead is ca 8.3% and remains constant vs. the messages count;
- When the message size decreases, the monitoring decreases as well; so a way to contribute to a lower overhead is seek for speeding up the computation of data volume instead of serializing the objects transferred;
- Another source for cutting monitoring costs is to handle monitoring data at a lower level to avoid data transfer and to aggregate data;
- The user can decide whether they want to obtain the real size of data transferred or the MUSCLE's message size

Case study

- An experiment performing heat flow in the object;
- Six kernels which are communicating with each other (communication matrix was shown above);
- Every kernel is connected – and therefore communicating – to two other kernels, the exception are boundary kernels which are connected to only one kernel;
  - Heat flow in object. Results from multiple kernels



- Used only small messages (about 400B – in fact this is a table of 50 java double primitives) and the overhead was ca 6%;
  - :

Conclusions

- MUSCLE extension, providing information about inter-kernel communication and kernel's state;
- Specialized SemMon adapter, which gathers data from Nagios and MUSCLE. The adapter provides collected data for SemMon core;
- Dedicated visualisations for communication between kernels;
- Measured serialization's impact on experiment's execution

Future work

- New types of visualization, like extended space-time digram;
- Adaptation to other applications, built with the message passing paradigm;
- Use of some existing reasoning mechanisms searching for the reasons of performance flaws, e.g. fuzzy logic.

Acknowledgements

The research is partly supported by Polish Infrastructure for Supporting Computational Science in the European Research Space *PL-Grid*.

Inspiration and support from Dr. Kasia Rycerz is appreciated.

Thank You!