Monitoring of the UNICORE middleware

Piotr Bała, Krzysztof Benedyczak, Mariusz Strzelecki
Faculty of Mathematics and Computer Science
Nicolaus Copernicus University &
Interdisciplinary Center for Mathematical and Computational Modeling
Warsaw University

May 19, 2010

Outline

Introduction

Reasons for grid monitoring:

- to ensure reliability or find cause of failure immediately
- to collect statistical data
- to inform users about the state of grid
- to see if production status is reached

Areas of grid monitoring:

- System-level grid monitoring
- User-level grid monitoring
- Server-side middleware monitoring

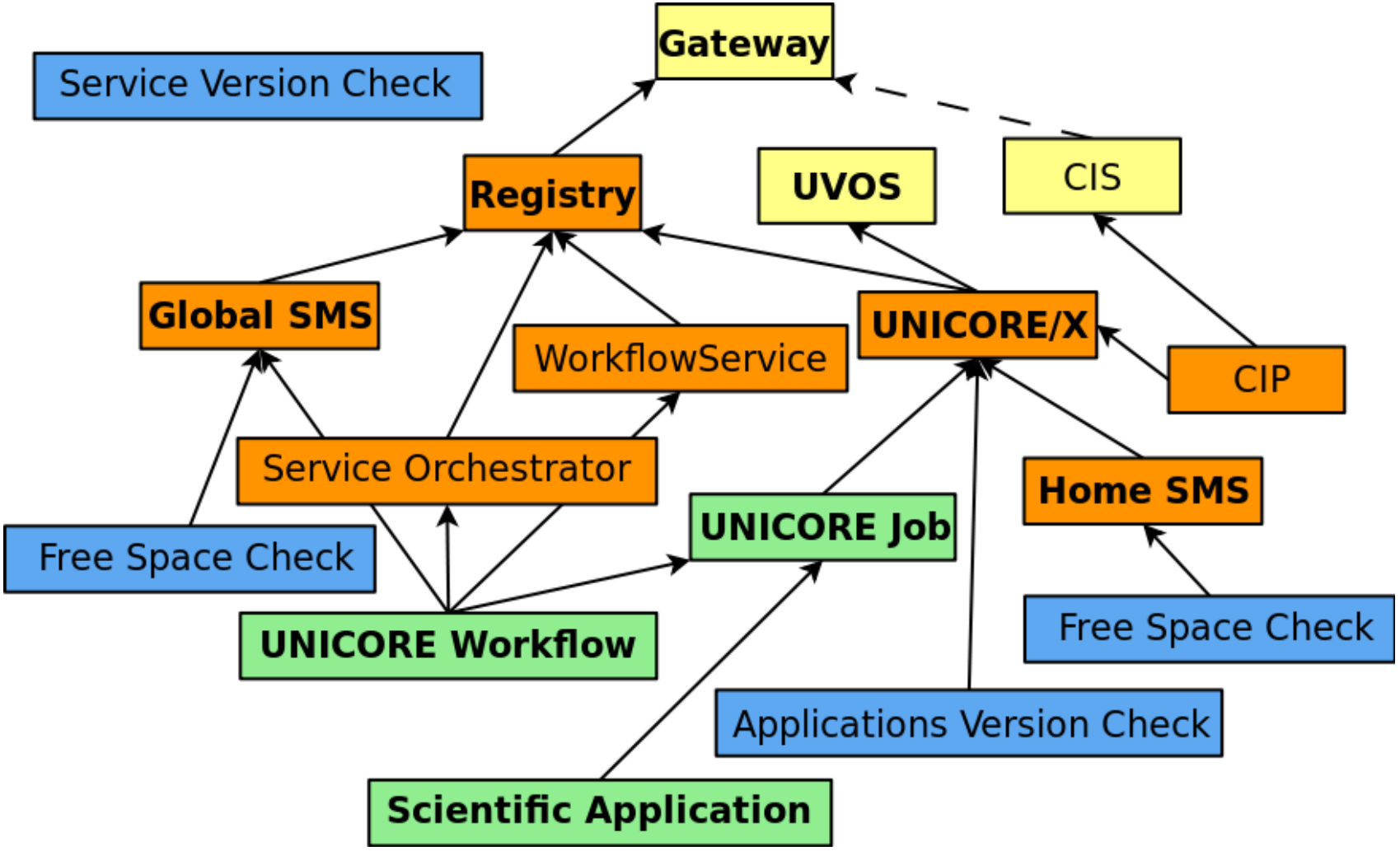
Monitoring of UNICORE so far

- Common Information Service - collects grid usage data, checks CIP (UNICORE/X) availability
- SIte MONitoring 6 - runs a workflow or job using UCC and checks potential exceptions or errors on client side

PL-Grid UNICORE monitoring system

- functionality **test for each service** or service container
- ability to quickly and **accurately indicate reason of a failure**
- tests dependency allows **overhead minimization** of UNICORE
- **test for** every installed **scientific application**
- tests for additional data that the services provide (free disk space on SMS, installed application versions, UNICORE service versions)
- **easy** one-script **installation**
- programmed using Perl and Java, distributed as **open source**
- **Nagios** as well-known, mature **test execution and notification environment**

Tests dependency graph



- negative and positive dependencies

check_sms

- generates test file of given size and then tests all commands that can be executed via SMS
  - lists SMS content: `ucc ls`
  - uploads file: `ucc put-file`
  - downloads file: `ucc get-file`
  - removes file: `ucc rm`
- checks output from every command to grab errors and at the end checks if files (generated and downloaded) are the same, additionally informs user about upload and download speed
- in standard UNICORE environment this script test both: Global SMS and SMS of every UNICORE/X instance
- additional test (in development state) checks free space on SMS every 24 hours (using `ucc wsrf getproperties [sms_address]`)

check_application

- submits previously prepared job description and waits to finish the job
- tests output files with given logical condition, e.g. `equals(#stdout,/etc/nagios/UNICORE/R/r_output.stdout) and valid_pdf(#plots.pdf)`
- files which are proper output from job are saved in installation process
- if job fails or condition is not fulfilled, test informs user about the situation
- used to check every scientific application (currently supported: Blast, Clustal, Fluent, R) and also acts as test "UNICORE Job" (just calling `echo` to test if UNICORE/X job submission works)

check_workflow

- this test takes the following steps:
  - uploads test file to Global SMS
  - submits workflow to Workflow Service asynchronously
  - checks the status of Workflow every 10 seconds and waits to finish it
  - checks status and output file
- default workflow executes `cat` commands chain over all TSFs that Service Orchestrator manages (workflow description is created by installation scripts)
- this is the integration test: if it succeeds, no other test is executed (to minimize network traffic and overhead of UNICORE)

Other minor probes

- check_gateway - connects to Gateway and analyses "monkeys page", checks if every required VSite is available
- check_registry - executes `ucc system-info` to get services from Registry, checks if every entry marked in configuration as required is available
- check_uvos - gets identity description from UVOS and compares it with data get in installation process
- check_unicorex - checks if TSF is available in Registry, optionally lists entries in Local Registry of UNICORE/X, tries to create TSS if not found

Missing probes (in development stage)

- check_cis - connects to CIS and lists CIPs that are managed
- check_cip - connects to CIS, gets the data from requested CIP and compares them with data gathered from CIP directly (via `ucc query-cip`)
- check_servorch - submits work assignment to Service Orchestrator
- check_workflow_service - uses `ucc workflow-info` to check if Workflow Service is available and responds for a query

Monitoring of additional information

- can gather information on free space on SMS
- can check versions of UNICORE services containers
- can check if scientific applications installed on target systems are not too old

Logs monitoring

- application that periodically runs on server side and checks for errors or warnings in log files from last time
- ability to ignore "known" error log entries
- if notices a problem, sends information to monitoring host via Nagios Service Check Acceptor
- Nagios sends notifications to administrators with observed log lines (log monitor stops working unless administrator confirms the problem as solved)

Installation of monitoring infrastructure

- installer must be initialized only with the following properties:
  - UCC path and configuration file
  - UVOS CLC path and configuration file
  - Registry URL
  - Site name
  - Workflow service name
  - Nagios contact group (to notify about problems)
  - UNICORE installations dependencies description
- installer generates required Nagios configuration and:
  - stores data gathered from UVOS for future tests
  - checks availability of Workflow system in given Registry and makes complex workflow description to test all TSFs
  - checks for availability of scientific applications and installs tests from templates

Nagios demonstration

https://alfred.studmat.umk.pl/nagios/

Summary

- the monitoring system is able to supervise all UNICORE atomic services, the Workflow system and the UVOS server
- all tests are placed in dependency graph to minimize UNICORE monitoring overhead
- the additional effort has been performed to monitor application functionality
- the constant monitoring allowed to identify few bugs in the UNICORE middleware
- the system is deployed in PL-Grid project, today one monitoring site regularly checks 4 complete UNICORE installations

Current version of monitoring system we develop can be obtained from: `http://unicore-life.svn.sourceforge.net/viewvc/unicore-life/monitoring/`

Future work

1. Developing tool to monitor logs on server side
2. Completing services probes list
3. Adding additional information probes
4. Making it possible to get statistical data (job execution time, free space) from probes and draw graphs

Release of whole monitoring system is planned at the end of work with points 1-3.

Thank you