MAPPER Components

From Wiki

Contents

Introduction

Here we briefly describe several components being part of the MAPPER project's fast track set. The selections of the components to be evaluated was based on the requirements of the two use cases of the multi-scale application: Nano-materials (UCL) and Instent Re-stenosis (UvA). We plan to demonstrate the two aforementioned application, using PRACE, EGI and local resources during the first MAPPER Review (November 2011).

Moreover only the components that have to be installed on the local resources level was described in more detail.

QCG-Computing

Description

- Component name: QosCosGrid Computing
- Related components: QCG-Broker, QCG-Notification
- Developer: Poznan Supercomputing and Networking Center
- Technical contact: Mariusz Mamonski <mamonski@man.poznan.pl>
- Website: http://www.qoscosgrid.org/trac/qcg-computing
- Documentation: http://www.qoscosgrid.org/trac/qcg-computing/wiki/InstallingFromSources
- Description: The QCG-Computing is an open source service acting as a computing provider exposing on demand access to computing resources and jobs over the HPC Basic Profile compliant Web Services interface. In addition the QCG Computing offers remote interface for Advance Reservations management.
- Deployment scenario: The component should be deployed on the frontend (i.e. submission host) of the

cluster

System Requirements

- Programming language: The core of the service was written in C, some extension modules are written in Python.
- Software requirements: QCG-Core (http://www.qoscosgrid.org/trac/qcg-core) utility library (which later depends on libxml, unixODBC, OpenSSL) database backend compatible with ODBC (recommended: PostgreSQL). For operation with the QCG-Broker a gridFTP server must be installed on the cluster. It can be existing gridFTP installation (even on different machine) as soon as this installation use stating accounts.
- Hardware requirements: any commodity CPU, 512 MB of memory, after installation about 80 MB + some space for logs
- Maintenance requirements: Cleaning old logs, making backups of the accounting database, keeping the GSI infrastructure, e.g. grid-mapfile, up-to-date (this task can be automated by using CRON jobs)

Security

- Incoming ports: One incoming port (for the service).
- Outgoing ports: One outgoing port (outgoing connection to QCG-Notification).
- System privileges:
  - QCG-Computing need to be started as root, however its adheres to the privilege separation model and thus only small part of the code actually runs with user id 0.
  - Advance Reservation service need to be reservation administrator (e.g. ADMIN2 in Maui, member of the ar_users in SGE, regular user in PBS Pro)
  - It is possible to enforce additional Local AR Polices (http://www.qoscosgrid.org/trac/qcg-computing/wiki/InstallingUsingRPMS#Restrictingadvancereservationl) using QCG services. A feature that is currently missing in many of the modern batch systems.
  - The service is currently audited (for the second time) by the PL-Grid security team.

QCG-Notification

Description

- Component name: QosCosGrid Notification
- Related components: QCG-Broker, QCG-Computing
- Developer: Poznan Supercomputing and Networking Center
- Technical contact: Bartosz Bosak <bbosak@man.poznan.pl>
- Website: http://www.qoscosgrid.org/trac/qcg-notification
- Documentation: http://www.qoscosgrid.org/trac/qcg-notification/wiki/Installation
- Description: QCG-Notification is an implementation of the of OASIS WS-Notification standard (brokered scenario). Among others QCG-Notification provides functionality for managing subscriptions and filter notifications.
- Deployment scenario: QCG-Notification should be deployed on a machine accessible from the QCG-Broker service and having access to QCG-Computing and QCG-Broker services. We suggest to install QCG-Notification on the same machine as QCG-Computing (i.e. submission host).

System Requirements

- Programming language: The core of the service was written in C.
- Software requirements: QCG-Core (http://www.qoscosgrid.org/trac/qcg-core) utility library (which later depends on libxml, unixODBC, OpenSSL), database backend compatible with ODBC (recommended: PostgreSQL).
- Hardware requirements: any commodity CPU, 512 MB of memory, after installation about 40 MB +

some space for logs
- Maintenance requirements: Cleaning old logs.

Security

- Incoming ports: One incoming port (for the service - incoming connections from QCG-Computing and QCG-Broker).
- Outgoing ports: One outgoing port (outgoing connection to QCG-Broker)
- System privileges: regular user

MUSCLE

Description

- Component name: MUSCLE - Multiscale Coupling Library and Environment
- Related components: QCG-Computing (QCG-Computing executes MUSCLE kernels)
- Developer: Primarly Jan Hegewald, currently as part of the MAPPER project
- Technical contact: Mariusz Mamonski <mamonski@man.poznan.pl>, Joris Borgdorff <j.borgdorff@uva.nl>, Bartosz Bosak <bbosak@man.poznan.pl>
- Website: http://www.qoscosgrid.org/trac/muscle
- Documentation: http://www.qoscosgrid.org/trac/muscle/wiki/Installation
- Description: MUSCLE is a platform independent agent system to couple multiscale simulations. It provides the software framework to build simulations according to the complex automata theory defined in the COAST project (http://www.complex-automata.org/).
- Deployment scenario: MUSCLE should be installed on worker nodes.

System Requirements

- Programming language: MUSCLE was generally written in Java, hovewer parts of the software are also written in C++ and Ruby.
- Software requirements: jre/jdk 1.6, Ruby 1.8.x, c++ compiler, otf (http://tu-dresden.de/die_tu_dresden/zentrale_einrichtungen/zih/forschung/software_werkzeuge_zur_unterstuetzung_ (optional dependency)
- Hardware requirements: any commodity CPU, 512 MB of memory, after installation about 60 MB
- Maintenance requirements: none

Security

- Incoming ports:
  - one open port at the frontend/interactive node (the port must be accessible from the other clusters involved in cross-cluster simulation)
- Outgoing ports:
  - connectivity from worker nodes to the frontend/interactive node
- System privileges: regular user

Other Grid level components

Here we shortly describe the other components playing a major role in the overall system. These components are not directly part of the evaluation process because they do not have to be installed on local resources.

QCG-Broker

QCG-Broker is an open source meta-scheduling system which allows developers to build and deploy resource management systems for large scale distributed computing infrastructures. The main goal of QCG-Broker is to manage the whole process of remote job submission to administrative domains controlled by domain level QCG components. For more information see http://www.qoscosgrid.org/trac/qcg-broker/

QCG-Coordinator

The QCG-Coordinator service is a lightweight service that coordinates startup of cross-cluster parallel aplication. It may be installed from the sources included in the QCG-Computing package.

Contact information

- Krzysztof Kurowski <krzysztof.kurowski@man.poznan.pl>
- Ilya Saverchenko <Ilya.Saverchenko@lrz.de>
- Joris Borgdorff <J.Borgdorff@uva.nl>,
- Bartosz Bosak <bbosak@man.poznan.pl>,
- Mariusz Mamoński <mamonski@man.poznan.pl>.