D4.2 Software Adaptation and Testing Reports: Delta Report

Project acronym: *MAPPER*

Project full title: Multiscale Applications on European e-Infrastructures.

Grant agreement no.: 261507

Due-Date:	Month 24
Delivery:	Month 24
Lead Partner:	UCL
Dissemination Level:	Public
Status:	Living, second release
Approved:	Q Board, Project Steering Group
Version:	0.7

Summary of changes in D4.2

Overall:

The first version of D4.2 contained information on the testing and adaptation of the QCG Middleware, GridSpace, MUSCLE and the AHE. This updated version now also contains information on MPWide, which has now also been adopted in the project.

Software adaptation report

The main updates here are the reports on the features in the upcoming MUSCLE 2.0 release, which now supports MPI, includes a C++ interface and has various usability, deployability and performance improvements. MUSCLE 2.0 will be released in the fall of 2012. This section also now reports on the MPWide communication library for wide area message passing. MPWide has been adapted to link to the subcodes in the cerebrovascular blood flow application and it is being integrated with MUSCLE. We also added some extra details on the Application Hosting Environment 3.0, which has now been officially released.

Software testing report

This section now features detailed information on performance metrics and stress tests results on QosCosGrid Broker. Among the new metrics we introduced are submission overhead and throughput, and reservation overhead and throughput. The tests have been performed using several sites in the PL-Grid segment of the EGI infrastructure. In these tests we find that the time from the user submitting the job to the user receiving a job identifier by the broker is on average roughly 0.24 seconds, and that the time from submitting a reservation to receiving the reservation ID number is on average roughly 0.13 seconds. The actual processing of the submission and reservation take slightly longer, namely on average ~0.5 seconds for submission and ~1.5 seconds for resource reservation. However, this overhead is largely hidden from the user.

We also added some information regarding the code analysis procedures in GridSpace, and added a first testing report for MPWide. The MPWide codebase is integrated with HemeLB and in the progress of being integrated with MUSCLE, and will be part of the test routines performed within those projects. We also show some first performance results of MPWide using regular internet connections in MAPPER, and compare it to several existing methods of communication. Here we find that MPWide performs consistently better than the other tools we have investigated. Together with the minimal codebase and lack of dependencies, this led us to the decision to incorporate it in MUSCLE 2.0. We also report briefly on some basic tests of new improvements in MUSCLE. These improvements resulted in a 20-fold improvement in the communication throughput and an 8-fold improvement in the communication latency when run as a single instance.