D5.2 MAPPER vertical integration plan

Project acronym: *MAPPER*

Project full title: Multiscale Applications on European e-Infrastructures

Grant agreement no.: 261507

Due-Date:	M12
Delivery:	M12
Lead Partner:	PSNC
Dissemination Level:	PU
Status:	Final
Approved:	QAB
Version:	1.5

DOCUMENT INFO

Date and version number	Author	Details
06-09-2011 v0.1	Bartosz Bosak	Initial version
07-09-2011 v0.2	Stefan Zasada	Added section about AHE
07-09-2011 v0.3	Katarzyna Rycerz	Added section about GridSpace
09-09-2011 v0.4	Joris Borgdorff	Added section about MUSCLE
14-09-2011 v0.5	Stefan Zasada	Extended part about AHE
14-09-2011 v1.0	Bartosz Bosak	Corections to the general structure
15-09-2011 v1.1	Katarzyna Rycerz	Modifications related to WP8 components
15-09-2011 v1.2	Bartosz Bosak	Minor corrections
23-09-2011 v1.3	Bartosz Bosak	Addressed Katarzyna Rycerz's comments - minor corrections
29-09-2011 v1.4	Bartosz Bosak	Addressed Ilya Saverchenko's comments
30-09-2011 v1.5	Bartosz Bosak	Language correction

TABLE OF CONTENTS

List of Tables

List of Figures

1 Executive summary

This deliverable is a living document presenting plans and the current status of vertical software and e-Infrastructure services integration in support of large scale multiscale computing defined by the MAPPER user communities. The aim of this document is to ensure a consistent integration between components laying on different levels in the MAPPER architecture to satisfy application scenarios' needs defined in WP4 and to follow e-Infrastructure polices and best practices required for productive-quality software deployment and support presented in WP6. This document focuses on particular relations between multiscale applications provided by the MAPPER user communities, the dedicated MAPPER software for efficient and flexible distributed execution of such applications, and finally the existing or new e-Infrastructure-level tools and services. The vertical integration plans were driven by two defined general scenarios targeted on two classes of multiscale applications, namely loosely-coupled and tightly-coupled. As each group of applications requires advanced and in some places also different capabilities, the MAPPER middleware services and tools must offer a wide range of new features, especially dedicated for multiscale computations.
Various aspects of vertical integration of MAPPER components are discussed in subsequent sections of the deliverable, starting from the general requirements for MAPPER and its general architecture, through the complete analysis of the required functions for the two defined types of applications, the implementation details of scenarios, ending with the productive software tests and deployment of scenarios on e-Infrastructures, in particular on DEISA/PRACE and EGEE/EGI sites.
The current version of the document describes several services and tools classified in Description of Work - Anex I [1] as ”fast-track” and ”deep-track” components, namely: MUSCLE, GridSpace, QosCosGrid, AHE. In order to provide the most useful information, during the whole project lifetime the deliverable will periodically be updated and adjusted to available at this moment project's results.

2 Contributors

- PSNC: Bartosz Bosak, Krzysztof Kurowski, Mariusz Mamonski, Tomasz Piontek
- UvA: Joris Borgdorff
- Cyfronet: Katarzyna Rycerz, Eryk Ciepiela
- UCL: Stefan Zasada, Derek Groen
- LMU: Ilya Saverchenko

3 List of abbreviations

Item	Description
AHE	Application Hosting Environment
DEISA	Distributed European Infrastructure for Supercomputing Applications
DRMAA	Distributed Resource Management API
EGEE	Enabling Grids for E-sciencE
EGI	European Grid Initiative/Infrastructure
GRAM	Grid Resource Allocation and Management
HPC	High-Performance Computing
JSDL	Job Submission Description Language
MML	Multiscale Modeling Language
MUSCLE	Multiscale Coupling Library and Environment
NGS	National Grid Services
PBS	Portable Batch System
PL-Grid	Polish National Grid Initiative
QCG	QosCosGrid
UNICORE	Uniform Interface to Computing Resources
xMML	XML representation of MML

Table 1: Abbreviations

4 High-level vertical integration plan

A general view on vertical integration plan of MAPPER is presented in Figure 1. Although the plan was established on the basis of the expert knowledge and experiences of the partners, it may be slightly altered during the project. The components highlighted in green are being developed and extended to fulfill certain needs of multiscale computations in MAPPER, whilst the violet components are already productively exploited and will not be changed in any way within the project.

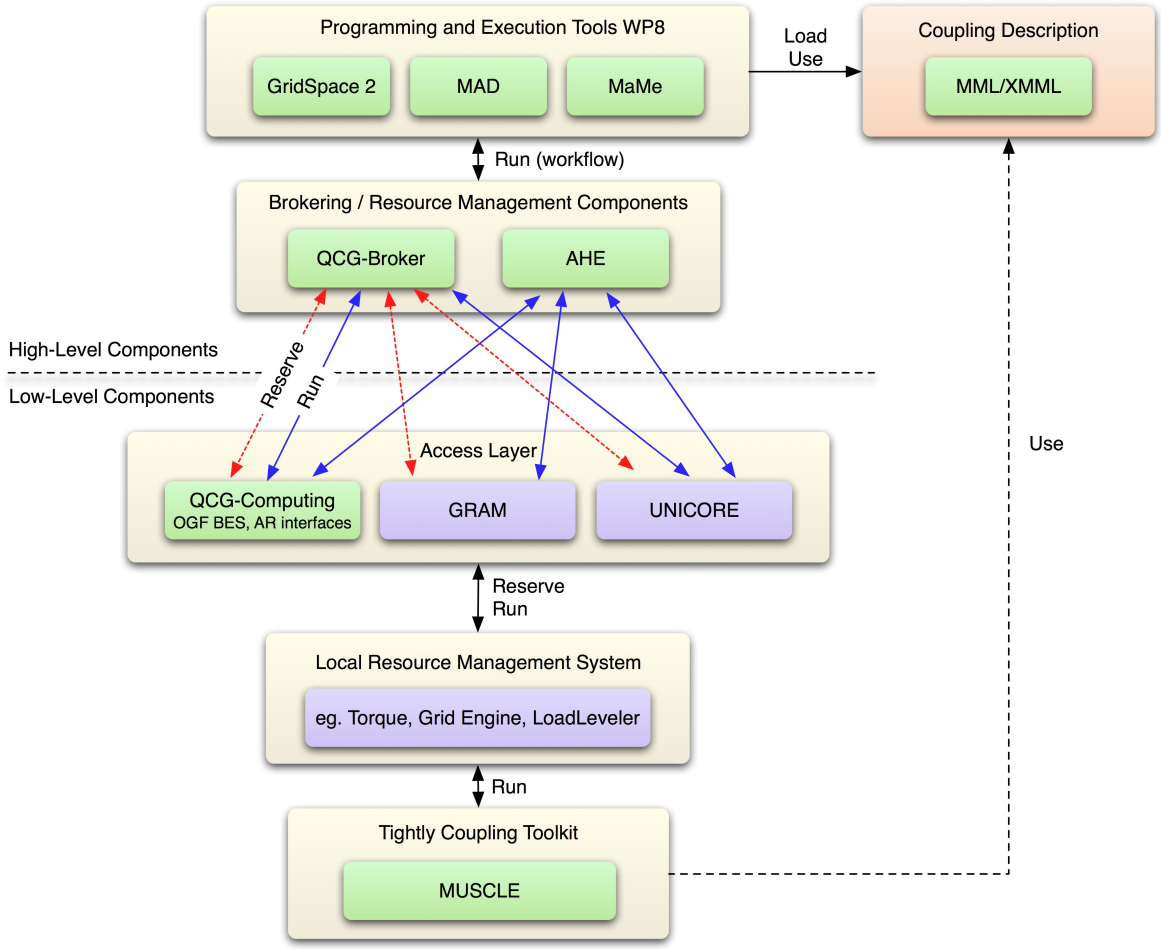


Figure 1: General vertical integration plan

Two levels of components separated in the figure by the dashed line represent a natural distinction between low-level components that must be installed locally at the resource providers side and high-level components, specific for GRID or Cloud environments. The first of the top-placed components in the proposed architecture is GridSpace 2 platform belonging to the group of programming and execution tools described in details in D8.2. Within the MAPPER project GridSpace 2 plays a role of user interface and workflow execution engine. This second feature is particularly important in scope of MAPPER as it is responsible for the processing of loosely-coupled computations. Multiscale Application Desinger (MAD) and Mapper Memory (MaMe) extend the high-level functionality by providing multiscale application composition tool and semantical storage for MAPPER data, re-

spectively. The Brokering and Resource Management components create an interoperability layer between GridSpace 2 and low-level components. QCG-Broker offers functions for co-allocating resources which are highly desirable for multi-cluster application runs. The access layer located at the top of the low-level part of the architecture consists of the three components: QCG-Computing BES/AR, Grid Resource Allocation and Management (GRAM) and UNICORE, offering remote interfaces to underlaying Local Resource Management Systems (LRMS) such as Torque [18], PBS Pro [19], Sun Grid Engine [16] or LoadLeveler [23].
The elementary (single-scale) applications are run by selected LRMS and may use various techniques for parallel execution, e.g. OpenMPI, OpenMP, CUDA. In turn, the possibility of running tightly-coupled multiscale scenarios, composed of many single-scale models is provided by the MUSCLE platform. Owing to the support for advance reservation and co-allocation functions, co-operating single-scale models may be run simultaneously on several sites.
It was decided to use XML representation for the description of coupling for both MAPPER scenarios. A new XML document type, called xMML is based on Multiscale Modeling Language (MML) described in details in [6] as well as deliverables D4.1 and D8.1. Nevertheless, in the current phase of the project, the xMML description with respect to loosely-coupled scenarios is modeled as GridSpace scripts, whilst in tightly-coupled applications it is replaced by the CxA files being the default input format for the MUSCLE framework.
It is noteworthy that in the proposed architecture both defined MAPPER scenarios, i.e. loosely-coupled and tightly-coupled, may be combined in real use-cases. In other words, depending on the xMML definition, the particular multiscale application may be executed as a workflow in GridSpace and simultaneously use MUSCLE routines on the lowest level. The detailed information how it is realized is included in D8.2.

5 Loosly coupling platform - GridSpace 2

The loosely-coupled type of applications, in which an entire application consists of several steps in foreseeable order, is well matched the GridSpace operation model. GridSpace 2 is based on the notion of exploratory programming where each experiment consists of a number of snippets creating a workfow. Each snippet may be written in a different programming language and executed independently. In this way, time-consuming experiments do not have to be started from scratch each time a modification is made during the development. The web portal of the Experiment Workbench assists users in the iterative development of simulation of workflows with the use of popular scripting languages. The Experiment Execution Environment, which is a layer underneath the Experiment Workbench, evaluates snippets and executes them on remote sites when required. Moreover, GS provides a rich set of interpreters to develop glue code with, what is important for a user having to perform various analysis of the intermediate results between subsequent steps of a multi-model simulation. The Perl interpreter is supporter, as is the bash shell to run the actual computations, but other interpreters like Ruby, Python or AWK (depends on the availability of the packages on the target machine) might also be provided if needed. All of them may be used in combination in a single computational experiment (like a workflow of tools in a pipeline). GS supports running jobs through PBS using PBS gems (applications), which, when compiled on the

target machine, allow submitting computational runs directly from script languages like Ruby, Perl or Python. Alternatively, whole interpreter process can be executed wrapped as a PBS job.
The additional requirements for GS useful for the loosely-coupled applications are the multi-machine login capabilities and the multi-site file transfer. The former will allow being simultaneously present on two different machines (required to perform the loosely-coupled scenario consisting of a full 3-step nano computation for the polymer study) in a single Experiment Workbench window. The other, when the multi-machine login is present, will allow to transfer files between these machines in a simple drag-and-drop manner. This features are going to be present in the first prototype of WP8 tools (see details in D8.2).

6 Tightly coupling platform - MUSCLE

The MUSCLE coupling library and environment [7, 12] was selected to execute tightly-coupled multiscale models on distributed resources in MAPPER.
MUSCLE offers support for scales and it has two base elements: the kernel, which contains the submodel code, and the conduit, which transfers data. When communication within a submodel occurs, it is regulated by the model developer. Conduit filters may filter data that is transferred through conduits, and a developer may implement them or reuse a pre-defined one. Operations on data that need multiple inputs or multiple outputs but do not qualify as a submodel are called mappers, but are implemented as a kernel.
Within kernels, parallel code can be used, for instance by using Java threads or MPI technique.
A MUSCLE model can be executed by starting an instance of MUSCLE on each machine that should be used and starting submodel instances on those MUSCLE instances. If a single machine setup is needed, just one MUSCLE instance needs to be started, otherwise they can communicate using TCP/IP. Machines that have no TCP/IP connections to others that run MUSCLE can only be accessed if there is an interactive node with a TCP/IP connection that controls the executable on that machine.
When a MUSCLE model is executed, each of the submodels may be started separately. A tightly-coupled scenario is enabled by having loops within the submodels that wait for messages from other submodels at certain points. When no more messages can arrive, for instance when the submodel that should send the message quits, the waiting submodel can also quit.

7 Middleware e-Infrastructure services

7.1 QosCosGrid

QosCosGrid (QCG) [2] is a set of middleware services and tools dedicated for dealing with computationally intensive large-scale, complex and parallel simulations that are often impossible to run within a single computing cluster. Basing on QosCosGrid, resources of different administrative domains may be virtually welded via Internet into one powerful computing resource. Applications running on the basis of QosCosGrid, in particular GridSpace platform, may use transparently a variety of common technologies, such as OpenMPI [3] or ProActive[4], typically used for parallel

execution only on a single machine. Within the MAPPER project, the QosCosGrid stack has been integrated with the MUSCLE coupling library enabling flexible execution of multiscale applications. Since the common multiscale simulations require simultaneous execution, the QosCosGrid capabilites for advance-reservations and co-allocation of various types of resources which are unavailable in gLite or UNICORE, provide a good opportunity to run demanding multiscale scenarios on many clusters.

7.1.1 QCG-Broker

QCG-Broker [2] is a meta-scheduling system, which in principle allows developers to build and deploy resource management systems for large-scale distributed, multi-cluster computing infrastructures. The main goal of QCG-Broker was to manage the whole process of remote job submission and advance reservation to various batch queueing systems and subsequently to underlying clusters and computational resources. QCG-Broker deals with various meta-scheduling challenges efficiently, e.g., co-allocation, load-balancing among clusters, remote job control, file staging support or job migration.
The QCG-Broker service has been designed as an independent component which can take advantage of various low-level core and grid services and existing technologies, such as QCG-Computing or QCG-Notification, as well as various grid middleware services such as gLite, Globus or UNICORE.
The XML-based job definition language supported by QCG-Broker is Job Profile. The Job Profie schema makes it relatively easy to specify the requirements of large-scale parallel applications together with the complex parallel communication topologies. Consequently, application developers and end users are able to run their experiments in parallel over multiple clusters as well to perform various benchmark-based experiments as alternative topologies are taken into account during meta-scheduling processes in QCG-Broker.

7.1.2 QCG-Computing

Another key service in the MAPPER integration plan is QCG-Computing assigned to the low-level components. The QCG-Computing service (previously SMOA-Computing) [2] is an open implementation of SOAP Web Service for multi-user access and policy-based job control routines by various Distributed Resource Management systems. It uses Distributed Resource Management Application API (DRMAA) [5] to communicate with the underlying DRM systems. The flexible QCG-Computing design and implementation support different plugins and modules for external communication, authentication, authorization as well as interactions with accounting infrastructures and other external services. QCG-Computing service is compatible with the OGF HPC Basic Profile [13] specification, which serves as a profile over the Job Submission Description Language (JSDL) [14] and OGSA Basic Execution Service [15] Open Grid Forum standards. It also offers remote interface for Advance Reservations management, and a support for basic file transfer mechanisms. Additionally, in order to provide co-allocation support, QCG-Computing may be integrated with QCG-Broker - such a solution is exploited in MAPPER scenarios.
So far, the service has been successfully tested with many Distributed Resources Management

systems, i.e.: Sun (Oracle) Grid Engine (SGE) [16], Platform LSF [17], Torque/PBSPro [18], PBS Pro [19], Condor [20], Apple XGrid [21] and Simple Linux Utility for Resource Management (SLURM) [22]. The Advance Reservations capabilities were exposed for SGE, LSF and Maui [24] (a scheduler that is typically used in conjunction with Torque) systems.

7.2 AHE

The Application Hosting Environment, AHE [8], developed at University College London, provides simple desktop and command line interfaces, to run applications on resources provided by national and international grids, in addition to local departmental and institutional clusters. It does not display the details of the underlying middleware in use by the grid to a user. In addition, a mobile interface for Windows Mobile based PDAs is available, and an iPhone interface is in development. The AHE is able to run applications on both UNICORE and Globus grids, meaning that a user can use a single AHE installation to access resources for example from the UK NGI and PRACE. Development of an EGI connector for AHE is currently conducted.
The AHE is designed to allow scientists to run unmodified, legacy applications on grid resources quickly and easily, manage the transfer of files to and from the grid resource and monitor the status of the application. The philosophy of the AHE is based on the fact that very often a group of researchers will want to access the same application, but not all of them will possess the skill or inclination to install the application on remote grid resources. In the AHE, an expert user installs the application and configures the AHE server, so that all participating users can share the same application. This community model draws a parallel with the modus operandi of numerous scientific application communities.
The AHE client is easily installed on an end users machine, requiring only that they have a Java installation and an X.509 certificate for the grid, which they want to access. The client package contains both GUI and command line clients which interoperate, allowing jobs launched with the GUI client to be manipulated with the command line tools and vice versa, and application workflows to be easily constructed.
Within the MAPPER AHE is connected to the GridSpace workflow management system, in order to allow AHE hosted applications to be orchestrated as workflow components using GridSpace. AHE is used to hide such aspects of the complexity of running applications on the grid as where application binary is located, what environment variables need to be set to run the application, and even the back end interface used to communicate with grid resource. This simplifies the process of creating and running a workflow. The workflow designer does not need to be aware of where the application resides or how to launch it, and they simply issue a command to AHE to run the tool. GridSpace contains an extension that makes use of the AHE's client API in order to launch and monitor applications.

8 Application scenarios

In order to provide scientists with the most useful environment for developing and running multi-scale simulations, the MAPPER consortium selected two real application use-cases directly cor-

responding to the two basic coupling types (tightly- and loosely-coupled). In the context of this section, we wish to describe proposed integration schemes for both use-cases, which take into account similarities and differences in the architecture of scenarios, as well as communication patterns between selected components. The detailed description of the applications is out of scope of this document and is a part of deliverable D4.1.

8.1 Loosely-coupled application scenario

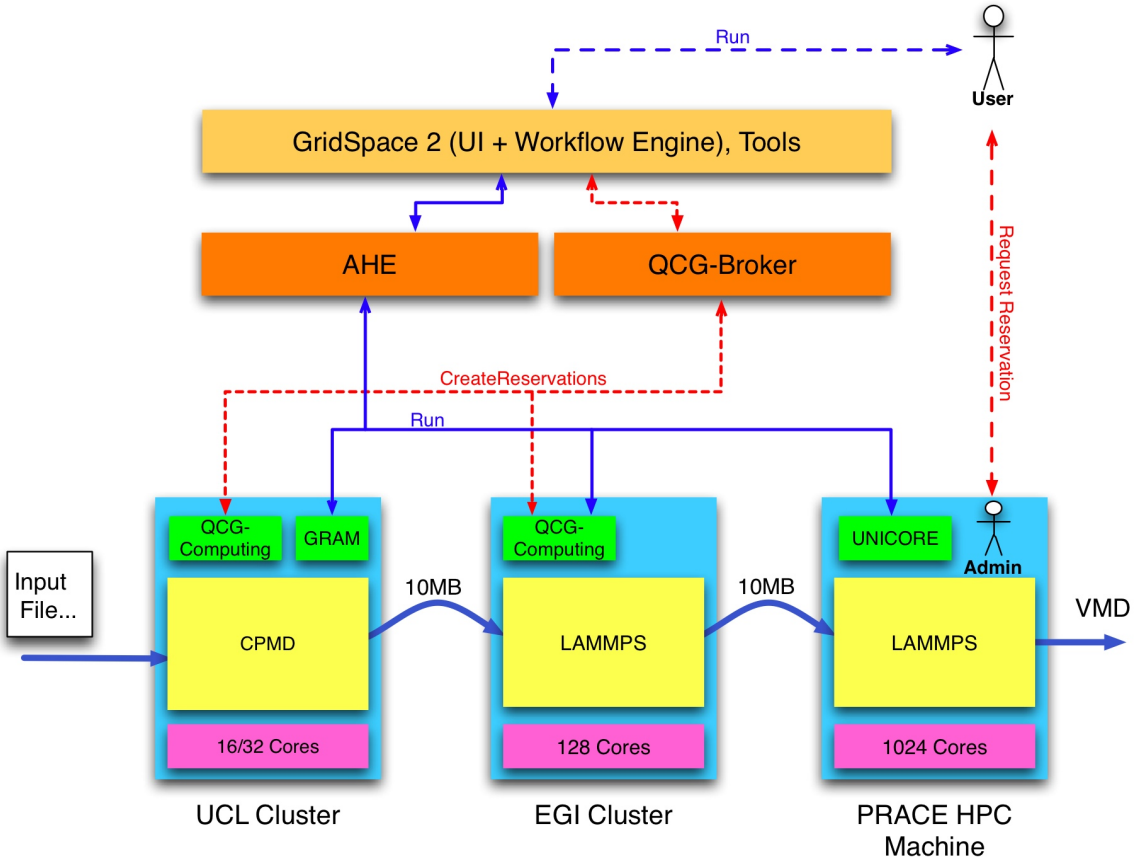


Figure 2: Loosely-coupled application scenario

The application selected to build the loosely coupling scenario, described in details in D4.1, is taken from the field of materials science. The loosely-coupled scheme consists of three levels of simulation. The lowest level simulates the electronic degrees of freedom, using the Car-Parrinello Molecular Dynamics (CPMD) code [10]. The atomic charges calculated at this step are passed to the initial models simulated using LAMMPS classical molecular dynamics code [11]. Finally, Coarse-Grained Molecular Dynamics (CGMD) simulations are performed using data received from previous step, again with the use of LAMMPS code.

The three components of the loosely-coupled application scenario are deployed on resources appropriate to their requirements. As it is presented on Figure 2, first, the CPMD application is executed on small cluster taking as an input a file located on a filesystem. The data processed at this step is transformed by GridSpace scripts into a form of input to the LAMMPS MD code

and transferred to an EGI resource running LAMMPS. Once the first LAMMPS model has been completed, its output is similarly processed into a form that can be used by the second LAMMPS model, and the data transferred to a PRACE HPC resource. At the end, the result of computations may be visualized by a VMD software [9]. The size of the transfered data between the core models is not large, approximately reaching 10MB.

As it was mentioned, the sequential execution of the different scale models is managed by the GridSpace workflow execution engine. The Application Hosting Environment is an interoperability layer offering an access to various types of computing resources. In turn, QCG-Broker is responsible for co-reserving resources to ensure a proper order of the execution of the submodels. This task is critical since a reservation of PRACE resource for processing of the last submodel must be created significant advance — as the first one; thus the reservations periods of the other two submodels must follow parameters of the PRACE reservation. It can also be noticed that the reservation of a PRACE machine is currently established outside the MAPPER components stack. It is believed that a further interconnected development of MAPPER and PRACE will allow removing this limitation.

The detailed sequence diagram showing interactions in the planned implementation of the loosely-coupled scenario for the first MAPPER review is presented in the Appendix A. The core part of the computation consists of the three steps corresponding to the main application phases and invokes one after the other. The order of creating advance reservations is important: in general, the period of reservation of HPC resource, which is created at the beginning, determines the start times and end times of the other two reservations.

8.2 Tightly-coupled application scenario

The tightly-coupled MAPPER scenario uses a three-dimensional In-stent Restenosis model (ISR3D) as the primary multiscale application. ISR3D allows 3D simulation of a stent deployment in the coronary artery and subsequent processes. The objective of the model is to help understand restenosis and to indicate improvements in the stent design. The simplified ISR3D model considered in this scenario consists of the three submodels: blood flow (BF), drug diffusion (DD) and smooth muscle cell proliferation (SMC) (For details see D4.1.). After the initialization routine, which is done in a loosely-coupled fashion, the model enters a tightly-coupled loop, where SMC, BF and DD are simultaneously computed. At some points of the compound calculations, the submodels exchange and synchronize data. In a typical run it may be about 1GB of information.

In the opposite to the loosely-coupled scenario, where the coupling is realized at the highest level by GridSpace which executes a workflow, in this scenario the routines responsible for submodels synchronization come from the MUSCLE library and are integrated at the programing level with the codes of the particular submodels' kernels (Figure 3). All interactions with the Access Layer services, including operations needed to reserve and co-allocate resources, are performed by the QCG-Broker service.

The way of the execution of tightly-coupled application differs significantly from the previous use-case. This difference is clearly shown in the Appendix B presenting a sequence diagram of the tightly-coupled scenario proposed for the first review of the project. One can note that the main part of the computations - marked by the dashed line - consists of the three MUSCLE-based models

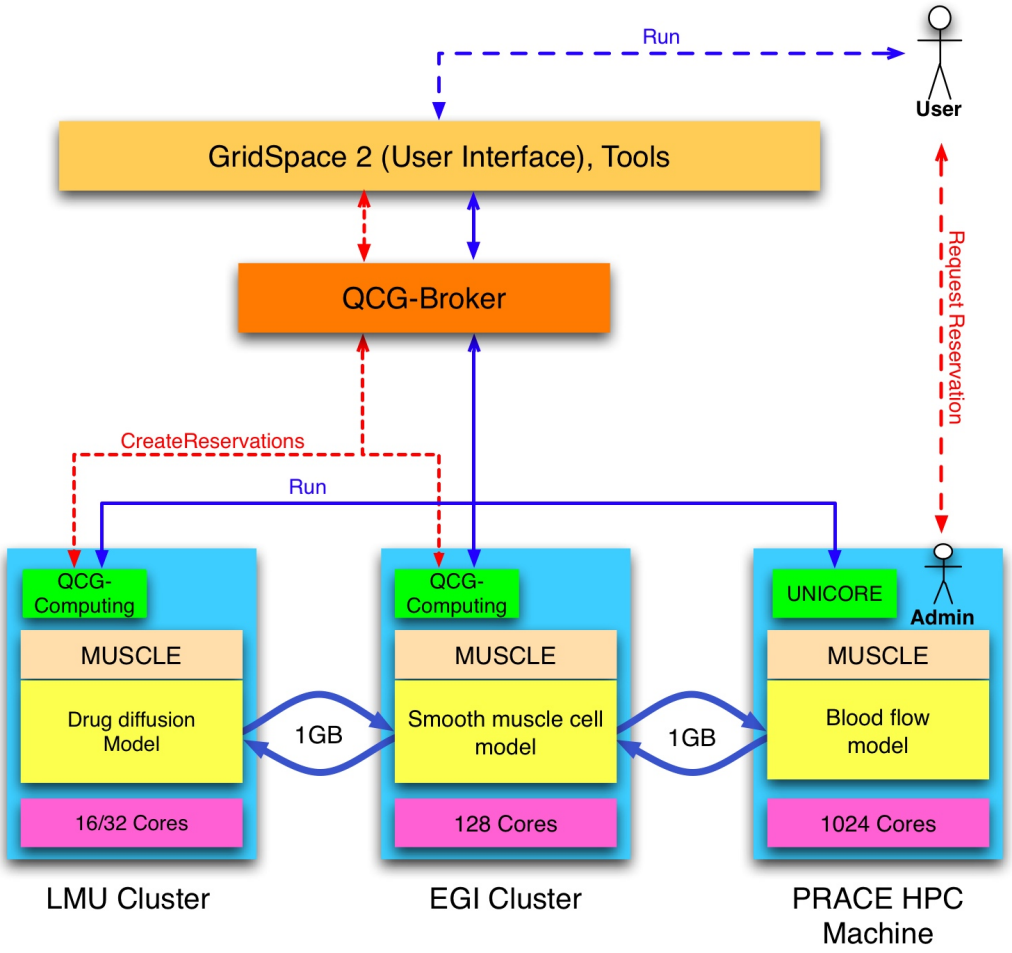


Figure 3: Tightly-coupled application scenario

running in parallel and exchanging information among themselves. In contrast to the loosely-coupled scenario, the coupling is realized on the application level, not in the GridSpace workflow engine, thus the vertical interaction steps between the components are limited significantly.

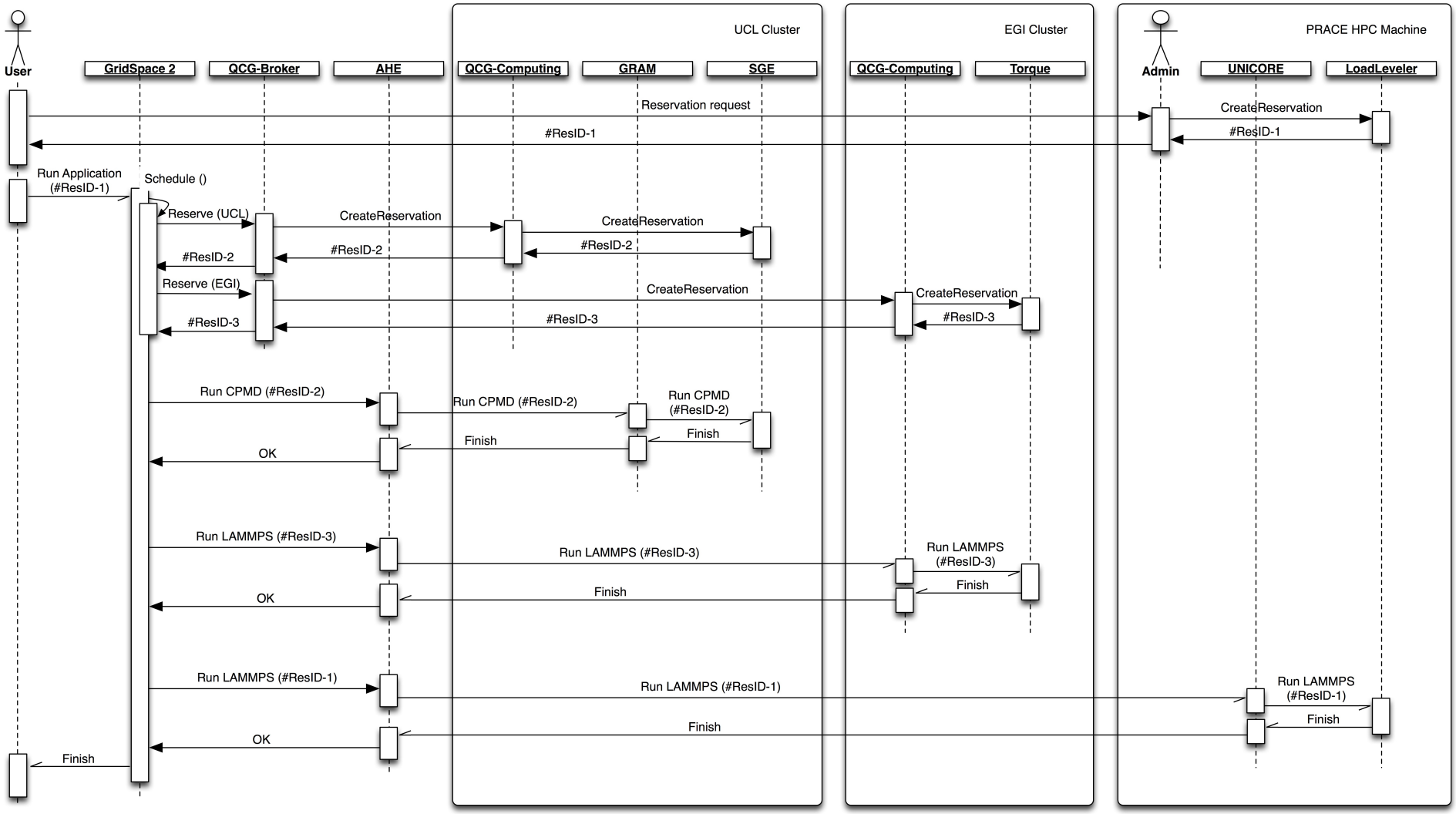
9 Conclusions

The MAPPER vertical integration plan presented in this document is based on the two complementary scenarios. The first scenario describes a concept of loosely-coupled multiscale computations. The key role in this model is performed by the GridSpace 2 platform, which executes a workflow of consecutive, essentially independent tasks. The second scenario is defined as a tightly-coupled as the multiscale coupling logic is moved to the lowest level of the MAPPER architecture and embedded using MUSCLE in the application. Nevertheless, since the components used in the two scenarios are mostly the same and suitable for both, the tightly-coupled and loosely-coupled models, the presented use-cases may be easily integrated in the future.

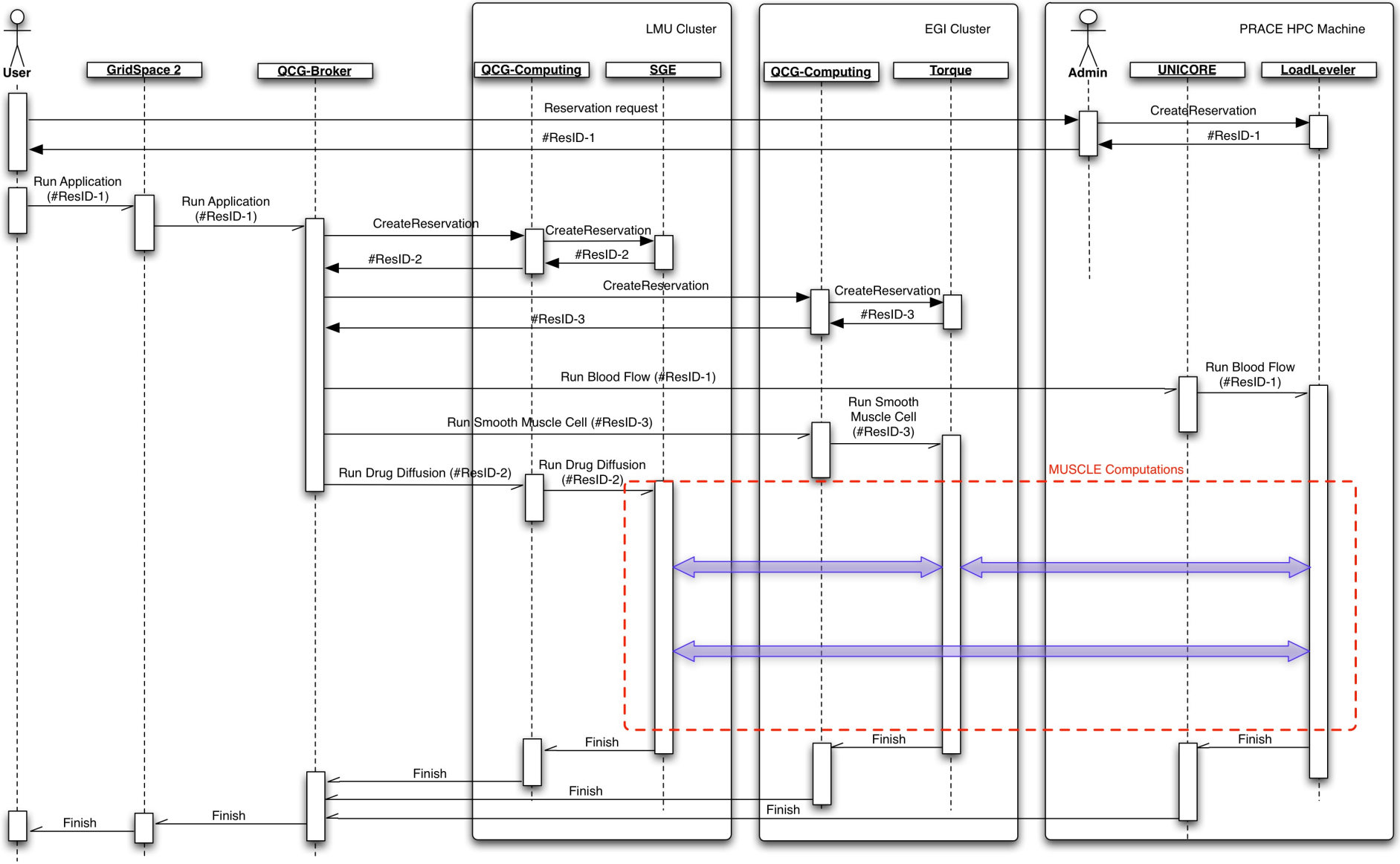
Both described scenarios will be presented during the first MAPPER review. A detailed plan for the two demonstrations is presented in a form of sequence diagrams attached in appendixes A and

B. It should be noted that depending on the integration status between the MAPPER components and the infrastructure services, the scenarios may be presented in a partially modified form. Nevertheless, any of the possible modifications will significantly not influence on the general vertical integration plan. All changes to the demonstration plans will be included in the future releases of this deliverable.

A Loosely Coupled Scenario Sequence Diagram



B Tightly Coupled Scenario Sequence Diagram



References

[1] Multiscale Applications on European e-Infrastructures (Mapper) Project – Annex I, ”Description of Work”

[2] Krzysztof Kurowski, Tomasz Piontek, Piotr Kopta, Mariusz Mamonski, Bartosz Bosak. Parallel Large Scale Simulations in the PL-Grid Environment. Computational Methods in Science and Technology, 47-56 (2010)

[3] Edgar Gabriel, Graham E. Fagg, George Bosilca, Thara Angskun, Jack J. Dongarra, Jeffrey M. Squyres, Vishal Sahay, Prabhanjan Kambadur, Brian Barrett and Andrew Lumsdaine, et al. Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation. Recent Advances in Parallel Virtual Machine and Message Passing Interface. pp. 353-377 (2004).

[4] Caromel, D., Delbe, C., di Costanzo, A. and Leyton, M.: ProActive: an Integrated Platform for Programming and Running Applications on Grids and P2P systems. Computational Methods in Science and Technology, vol. 12, no. 1, pp. 69-77 (2006).

[5] Peter Troeger, Hrabri Rajic, Andreas Haas. Standardization of an API for Distributed Resource Management Systems. Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid (CCGrid07), Rio De Janeiro, Brazil, May 2007.

[6] Jean-Luc Falcone, Bastien Chopard, Alfons Hoekstra. MML: towards a Multiscale Modeling Language, Procedia Computer Science (2010), Volume: 1, Issue: 1, Publisher: Elsevier, Pages: 819-826.

[7] Jan Hegewald, Manfred Krafczyk, Jonas Tlke, Alfons Hoekstra and Bastien Chopard. An Agent-Based Coupling Platform for Complex Automata. Lecture Notes in Computer Science, 2008, Volume 5102/2008, 227-233.

[8] P. V. Coveney, R. S. Saksena, S. J. Zasada, M. McKeown and S. Pickles, ”The Application Hosting Environment: Lightweight Middleware for Grid-Based Computational Science”, Comp. Phys. Comm., 176, 406-418 (2007).

[9] Humphrey, W., Dalke, A. and Schulten, K., “VMD - Visual Molecular Dynamics” J. Molec. Graphics 1996, 14.1, 33-38.

[10] The CPMD Consortium page, `http://cpmd.org/`.

[11] LAMMPS Molecular Dynamics Simulator, `http://lammps.sandia.gov/`.

[12] Jan, Hegewald. The Multiscale Coupling Library and Environment (MUSCLE), `http://muscle.berlios.de/` (2010).

[13] GFD 114-HPC Basic Profile, Version 1.0, `http://www.ogf.org/documents/GFD.114.pdf`.

[14] GFD 56 - Job Submission Description Language (JSDL) Specification, Version 1.0 - http://www.gridforum.org/documents/GFD.56.pdf.

[15] GFD 108 - OGSA Basic Execution Service Version 1.0, http://www.ogf.org/documents/GFD.108.pdf.

[16] Oracle Grid Engine, http://www.oracle.com/us/products/tools/oracle-grid-engine-075549.html.

[17] Platform Load Sharing Facility, http://www.platform.com/workload-management/high-performance-computing.

[18] Torque Resource Manager, http://www.clusterresources.com/pages/products/torque-resource-manager.php.

[19] PBS Professional, http://www.pbsworks.com/Product.aspx?id=1.

[20] Condor High-Throughput Computing System, http://www.cs.wisc.edu/condor/.

[21] Apple Xgrid, www.apple.com/server/macosx/technology/xgrid.html.

[22] Simple Linux Utility for Resource Management (SLURM), https://computing.llnl.gov/linux/slurm/.

[23] IBM Tivoli Workload Scheduler LoadLeveler, http://www-03.ibm.com/systems/software/loadleveler/.

[24] Maui Scheduler, http://www.clusterresources.com/products/maui/.

[25] Java Agent DEvelopment Framework, http://jade.tilab.com/ (2011).