D5.2 First report on ComPat middleware services

Due Date	18
Delivery	18
Lead Partner	PSNC
Dissemination Level	PU
Status	Final
Approved	Internal review - YES
Version	1.0

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 671564.

DOCUMENT INFO

Date and version number	Author	Comments
27.02.2017 v0.1	Bartosz Bosak	Table of contents
27.03.2017 v0.2	Bartosz Bosak	Merged input received from contributors
27.03.2017 v0.3	Bartosz Bosak	Minor corrections
28.03.2017 v0.4	Vytautas Jancauskas	Comments and Editing
31.03.2017 v0.5	Neil Morgan	Comments and Editing
04.04.2017 v0.6	Tomasz Piontek	Final editing

CONTRIBUTORS

The contributors to this deliverable are:

Contributor	Role
Tomasz Piontek	Technical Leader, WP5 Leader and author of this deliverable
Bartosz Bosak	WP5 contributor
Piotr Kopta	WP5 contributor
Maciej Tronowski	WP5 contributor
Saad Alowayyed	WP3 contributor

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF LISTINGS

LIST OF ABREVIATIONS

DRMAA	Distributed Resource Management Application API
ECOS	Energy Consumption Optimization Service
EEE	Experimental Execution Environment [Project Testbed]
ES	Extreme Scaling [Pattern]
HMC	Heterogeneous Multiscale Computing [Pattern]
HPMC	High Performance Multiscale Computing
NUMA	Non-Uniform Memory Access
PRACE	Partnership for Advanced Computing in Europe
RC	Replica Computing [Pattern]
QCG	Quality in Cloud and Grid [middleware]

1 Executive summary

This report is a deliverable summarizing the role as well as the current development and deployment status of the ComPat middleware services at month 18 of the project. It describes how the new capabilities of the middleware layer have been designed and implemented to fulfil the needs of the Multiscale Computing Patterns defined by Work Package 2. We further define how these patterns are used by multiscale applications to target the exascale resources being developed in Work Package 3. We present the most significant achievements so far and discuss solutions to the problems we encountered.

The results of the work described in this report are in conformance with the architecture of the ComPat system described in the deliverable *D5.1 – Architecture of the ComPat system*. One of the major achievements of the project is the first version of the ComPat system developed, deployed and running on EEE resources. To this end we developed and released new middleware services and updated existing ones. Additionaly this work supported the delivery of the M10 milestone and the release of Middleware Services.

Middleware services have been extended to allow ComPat users access to all of the Experimental Execution Environment (EEE) resources. This document describes all the work that was needed to provide versions of services that could be deployed on PSNC, LRZ and STFC resources and details unexpected, but substantial problems encountered at LRZ and STFC sites.

Energy consumption issues become especially important when dealing with computations in the exascale. Current hardware architectures when scaled up to deliver exacsale levels of compute require an uneconomic and impractical level of energy consumption. In order to deliver a viable solution it is necessary to optimise efficiency at all levels of the hardware and software stack. In order to provide the necessary capabilities to effectively execute High Performance Multiscale Computing (HPMC) pattern-based applications a through analysis was performed. Software was developed to optimise energy use when dealing with memory-bound multiscale applications. Implementation details are discussed later in the document explaining how this has allowed us to achieve a 30% improvement in energy efficiency.

As part of this work package a significant amount of effort has been committed to implement ComPat's HPMC patterns. To this end we have added new functionality to the QCG-Broker service, which includes the creation of a new pattern aware brokering algorithm. This is in addition to other tasks that were necessary to correctly implement resource allocation plans proposed by WP2.

2 Status of the ComPat Middleware

The main aim of Work Package 5 is the provisioning of middleware services that are capable of efficient execution of multiscale applications on top of the high-end e-Infrastructure, consisting of single or multiple resources of the Tier0/1 class. As presented in Figure 1 and described in detail in *D5.1 – Architecture of the ComPat system* [1] the ComPat project has selected the QCG system to be a primary middleware element.

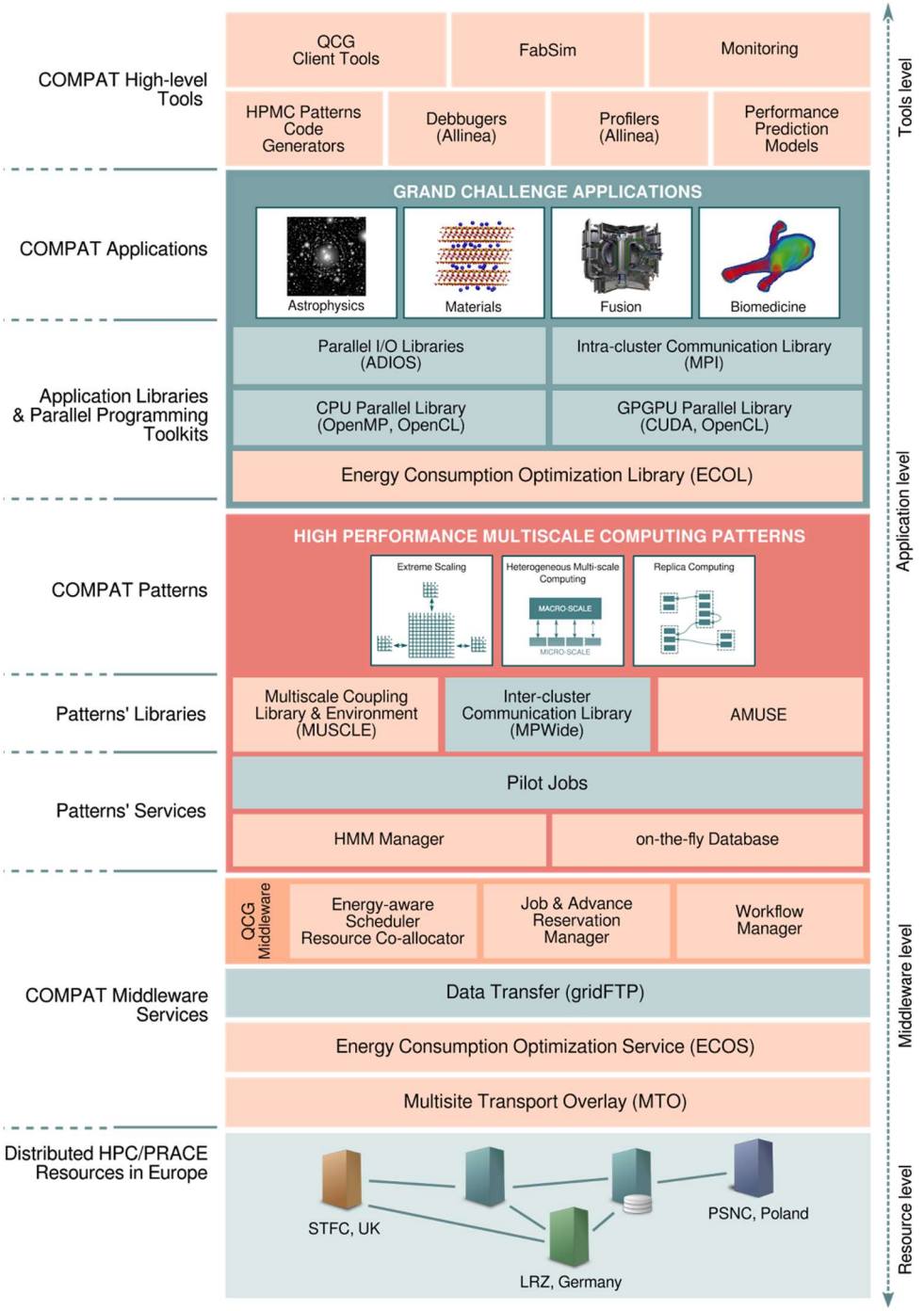

Figure 1. ComPat system architecture

In order to offer the requested functionality and bring the anticipated levels of abstraction over heterogeneous resources, the QCG middleware had to be functionally integrated with the Experimental Execution Environment (EEE, the ComPat testbed) core services, as well as cooperate with the ComPat High Performance Multiscale Computing (HPMC) patterns tools and services. WP5 is also responsible for the development of intelligent mechanisms that minimise energy-usage of pattern-based applications. This is necessary to efficiently run highly resource demanding multiscale simulations. The next subsections present current achievements of the project in all these areas.

2.1 Extensions in middleware to enable unified execution of applications on the EEE

The primary requirement of all ComPat participants, but especially the application groups, was to have instant access to the project testing environment - (EEE). It was decided to offer minimally adapted services as early as possible to provide an environment for testing. In order to enable access, cooperation between WP5 and WP6 was settled in the first months of the project. The deployment of QCG middleware was quickly accomplished on the PSNC's resources Eagle and Inula, where the new release 4.0 of the all QCG services and tools (the initial version for the created ComPat branch) was deployed. However, checking compatibility of the existing implementations with policies and restrictions at LRZ and STFC/Hartree Centre identified several unexpected problems. Before the basic version of middleware could be deployed on those sites, substantial work was required to resolve the aforementioned barriers.

Further extensions of the middleware services and tools were introduced to make the use of the middleware services themselves and the EEE resources more straightforward. In the rest of this section we describe all major middleware development and adaptation tasks associated with the provisioning and implmentation of the EEE.

2.1.1 Grid Security Infrastructure, Distributed Resource Management Application AP (GSI DRMAA)

A significant initial problem related to the deployment of middleware services was encountered on the SuperMUC machine at LRZ. It emerged that administrative restrictions present at LRZ's site resulted in the necessity to implement a new version of the DRMAA (Distributed Resource Management Application API) library. The new library used by the QCG-Computing service relies on LoadLeveler commands executed over the gsi-ssh protocol instead of the typically used queuing system API. This also had an influence on the QCG-Computing service itself, which had to be adapted to an essentially different way of interacting with queuing systems.

Development of the new version of the DRMAA library and its integration with QCG-Computing required significant effort which was not anticipated at the planning stage. The list below summarizes the steps that were needed to create the new GSI DRMAA library and integrate it with the QCG stack:

- Preparation of a skeleton for GSISSH/GSIFTP-based DRMAA library.
- Extension of the DRMAA module interface and implementation of a new mechanism to copy files using the gridFTP protocol.
- New mechanism on the DRMAA library level allowing remote commands using the GSISSH protocol.
- New mechanism to establish and store the status of the job submission based on the parsed output of the “llsubmit” command.
- New mechanism to monitor and store the status of the execution of a job based on the parsed output of the “llq” command.
- Modification and extensions to the DRMAA module configuration scheme.
- New mechanism for storing in a persistent way information about the final status of job execution together with the exit code.
- New mechanism to discover the user’s home directory on the remote system.
- New mechanism to create/remove working directory for the job on a remote system.
- New mechanism to pass from QCG-Computing to the DRMAA library a user proxy and to delegate it to the remote resource.
- Extension of the QCG-Computing configuration mechanism to deal with remote resource access.
- Modification of the QCG-Computing NAGIOS test subsystem to work with GSISSH-DRMAA and LRZ restrictions.
- Implementation of a mechanism for remote execution of the “llstatus” command together with parsing output and storing the results.
- Implementation of a mechanism for remote execution of the “llclass” command together with parsing output and storing the results.
- Implementation of a new facility for retrieving information about modules available on the systems resources.
- Compilation from sources of all the dependencies missing in SLES repositories as well as of QCG-Computing and QCG-Notification services.
- Adaptation of the QCG service management scripts.
- Final integration tests on the production infrastructure in LRZ.

2.1.2 Integration of QCG middleware with ComPat Monitoring and Downtime Services

A supplementary capability expected by ComPat users and technology providers was the capability to monitor the status of the infrastructure. For the sake of this requirement, during the first half of the project, the QCG middleware services were integrated with the ComPat Monitoring service developed by WP6. The integration was done by providing a set of dedicated, QCG specific, probes for the NAGIOS system. During the lifetime of the project the business logic and implementation of the initial versions of NAGIOS probes have been altered to allow the testing of new versions of the services, as well as to provide more accurate information about their statuses. The probe for the QCG-Computing service has been redesigned and reimplemented to allow job submission to the LoadLeveler queueing system on the SuperMUC cluster in LRZ. It was necessary to adjust the existing data transfer mechanisms to the newly developed DRMAA implementation. Additionally, the logic of the QCG-Broker probe has been modified to avoid false positive service failure detection, in cases where a job submitted to QCG hasn't finished in the assumed period of time, due to prolonged queue times.

Building of the unified ComPat execution environment required some additional implementation effort on both EEE and QCG sides. The computational resources forming the EEE belong to various European and local research infrastructures and they are managed within the scope of various initiatives and their operational procedures:

- SuperMUC (LRZ) - Partnership for Advanced Computing in Europe (PRACE)
- Eagle, Inula (PSNC) - European Grid Infrastructure (EGI)
- Neale (STFC) - Hartree Center,

The consequence of this heterogeneity is the fact that the EEE resources did not have a single information system storing and publishing information on current and planned downtimes. The knowledge about downtimes in a complex, distributed environment is required for proper allocation of tasks to resources. In the scope of WP6 a new service that integrates data about current and planned downtimes for all of the resources available for the project has been created and made available in the ComPat ecosystem. To achieve the logical integration of resources into a single research infrastructure, the functionality of the QCG-Broker service has been extended to take into consideration information about system downtimes and exclude from the list of available resources these with planned downtime that would make them unavailable for use. The integration was done by developing a new plugin for the ResourceDiscovery module of QCG-Broker that filters out resources with impending downtime that would impact on planned execution.

2.1.3 Integration with the Allinea tools

Interactive debugging and profiling of multiscale applications is enabled in ComPat by the Allinea tools being adapted by the project in WP4. In order to ensure usability of the Allinea tools the QCG middleware was configured to support interactive tasks on resources. The interactive task functionality was selected as a way of integration of the Allinea tools with the ComPat ecosystem. QCG tools and services allow a job to run in either interactive mode or to connect to an already running job through an interactive session. Both these approaches enable a system user to start and control the process of remote debugging and profiling by sending text commands from the QCG-Client side to the Allinea tools and daemons executed together with the application on the EEE. To allow seamless integration of the Allinea tools with the QCG infrastructure, the qcg-connect tool has been modified to eliminate from the output of the command any diagnostic and user intended information that could break the internal Allinea protocol used to control the remote process. The so called "quiet mode" has been implemented and added to the qcg-connect tool. The implementation of the "quiet mode" required changes and modifications in the structure of modules on the QCG-Client to allow interpretation of parameters passed to the command prior to any other action to prevent any QCG specific information being sent to the output stream of the command. The syntax of the qcg-command is shown in Listing 1.

```
[plgpiontek@qcg ~]$ qcg-connect --help
Connect to the task. Open the interactive terminal in the working directory of the
task

usage: qcg-connect [-h] [-Q] [-V]   JOBID[/TASKID]
usage: qcg-client connect_to_task [-h] [-Q] [-V]   JOBID[/TASKID]

Options:
-h,--help      display help message
-Q,--quiet     quiet mode
-V,--version   display version

Arguments:
JOBID  identifier of the job
TASKID optional identifier of the task
               The default identifier of task is 'task'
```

Listing 1. qcg-connect command usage

2.1.3.1 Network tunelling

In LRZ and STFC the network configuration disallows direct connections from the computing nodes to the machines outside the cluster's private networks. The LRZ network policy is especially restrictive - the incoming connections to the login node are accepted only for ssh/gsissh services and, what's more, any outgoing connections from the login node are prohibited. In order to provide an interactive task capability on these sites, which was a requirement for the integration of the Allinea tools with the EEE, a special, dedicated solution had to be designed. It was decided to use the well-known, reliable and secure SSH tunneling service as a base solution. However, a single SSH tunnel can be created only to the particular site and port, which limits its usage to a single service. In cases

when many connections to the various services are needed, more tunnels have to be created. Fortunately, the SSH service has SOCKS (versions 4 & 5) server implementation built-in which can be used with remote tunnels. The tunneled SOCKS server in combination with the jump hosts mechanism, which allows the creation of connections via one or more intermediate hosts, creates a very universal and flexible solution that has been used as a proof of concept for the QCG remote console capability in a network restricted environment.

Figure 2 presents the proposed solution for the LRZ site, where there is an additional jump host due to the network policy restrictions on the login node.

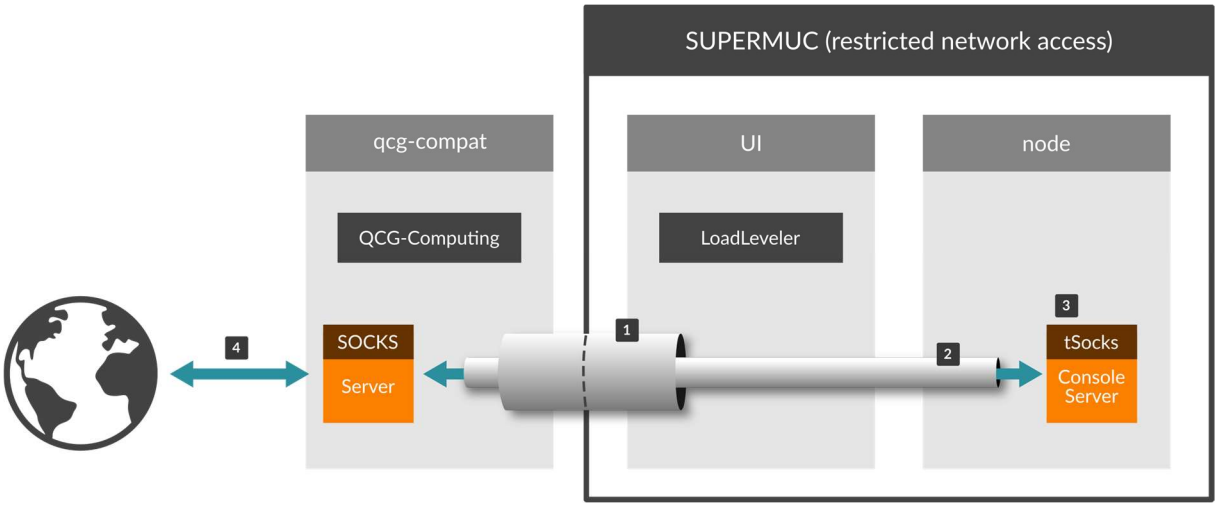

Figure 2. Network tunelling proposed for the SuperMUC restrictions

In the first step (1), the tunnel from the qcg-compat machine to the login node is created to allow SSH connections from the computing nodes to the qcg-compat site. This tunnel should be permanent as it will be shared by all QCG jobs. The second step, sets up a remote SOCKS server at qcg-compat site through a tunnel (2), and is unique for each QCG job and lasts as long as a job is executed. The tunnel is created to the qcg-compat site with one jump host, which is the login node. At this stage, on the computing node a listening SOCKS v5 socket is opened which forwards connections to the world with the SOCKS protocol. This protocol is not transparent to the applications - they need to use a special SOCKS connection. Fortunately, transparency can be achieved by use of a specialized wrapper tool, which catches all calls to socket functions and replaces them with SOCKS aware invocations. The legacy applications do not need to be modified. In the presented solution, the *tsocks* [3] wrapper tool is used (3). The described solution allows the QCG console server, executed with the aforementioned wrapper on a worker node of the cluster, to use the created SOCKS server and connect to the QCG console client application run on the user side (4).

The developed proof of concept solution confirmed that the network restrictions can be successfully overcome by tunneling and hopping over several machines. Although this mechanism is quite sophisticated, it offers good flexibility, security and performance. The final implementation was postponed until the positive acceptance of the proposed solution by the resource providers.

2.1.4 Other improvements in the middleware tools and services for multiscale applications

A large amount of work within WP5 has been devoted to the development of various improvements to the QCG stack. These improvements - although general purpose on their own - were motivated by the concrete needs of the ComPat applications and patterns. Among other, less obvious, capabilities, the QCG middleware has been extended with the several major features presented below.

2.1.4.1 Job Array Support

The capabilities of QCG services and tools have been unified and extended to support job array functionality on the EEE resources. This functionality allows a set of independent tasks to be run on the EEE inside a single job. Thus it can be easily used for example to search a parameter space for a specific problem. In context of the ComPat project, Job array support is used to address the needs of the Replica Computing pattern, in which the main concept is to run a set of independent tasks (replicas). QCG allows submission and control of multiple such tasks as a single QCG task. Most importantly - all the sub-tasks are scheduled independently and can be executed on various clusters to balance the overall load on the EEE, to increase the overall throughput of the system and decrease the total time to completion for the replica pattern. The QCG Simple Description format was extended to support two ways of defining the job array specification either by providing a range or a list of values. The example of the QCG job with two alternative definitions of the job array is presented in Listing 2.

```
#!/bin/bash
#QCG host=agave
#QCG walltime=PT10M
#QCG output=${JOB_ID}.output
#QCG error=${JOB_ID}.error
#QCG stage-out-dir=.-
>${JOB_ID}

#QCG array=1-10:2
#QCG array=1,3,5,7,9
sleep $(($DRMAA_ARRAY_TASK_ID*60))
```

Listing 2. The QCG job's description with job arrays

Additionally, the QCG-Client was extended to report the status of the submitted job array. For the job array task the qcg-info command provides general information about the total number of elements in the array and a list of statuses with a number of sub-tasks in them. Listing 3 presents the output of an example invocation of the qcg-info command.

```
qcg-info J1489153370445__9669"

J1489153370445__9669 :
Note:
UserDN: /C=PL/O=PL-Grid/O=Uzytkownik/O=PL-Grid/CN=Tomasz Piontek/CN=plgpiontek
TaskType: SINGLE
SubmissionTime: Fri Mar 10 14:42:50 CET 2017
FinishTime:
ProxyLifetime: P24DT6H11M39S
Status: RUNNING
StatusDesc:
StartTime: Fri Mar 10 14:42:51 CET 2017
Purged: false

Allocation:
HostName: agave
ProcessesCount: 1
ProcessesGroupId:
Status: RUNNING
StatusDescription:
SubmissionTime: Fri Mar 10 14:42:51 CET 2017
FinishTime:
LocalSubmissionTime: Fri Mar 10 14:42:53 CET 2017
LocalStartTime: Fri Mar 10 14:42:54 CET 2017
LocalFinishTime:
Purged: false
JobArrayStatus: TOTAL=5 EXECUTING=2 FINISHED=3
WorkingDirectory:
gsiftp://agave16.man.poznan.pl/~/J1489153370445__9669_task_1489153371056_333
```

Listing 3. The output of qcg-info command for job array

In the example presented above, the total number of elements in the job array is 5. Two subtasks are being executed and three have already finished.

2.1.4.2 Extensions to the Workflow execution

To address the specific needs of the ComPat project and to allow efficient execution of the patterns (mainly the Replica Computing Pattern) the workflow execution capabilities of the QCG-Broker service have been adapted and extended. The efficient execution of the real example of the Replica Computing Pattern - *Binding Affinity Calculator (BAC)* application required the following general improvements on the QCG-Broker side:

- The “extension” mechanism has been introduced to allow the running of dependent tasks in the working directory of the parent task to avoid necessity to transfer data between tasks. The dependencies between tasks are expressed in the QCG Job description:
  `<taskid=”amber” extension=”namd”>`
  The above definition specifies that the “amber” task should be started in the working directory of the previously executed “namd” task.
- Also for the BAC scenario the general extension mechanism was extended to support parameter sweep tasks to enable data transfer optimization for a set of tasks.

In such a case, in the name of the parent task of some task in a workflow, a parameter sweep variable may be provided. This variable is then substituted by the proper value during the execution of the workflow:

```
<task taskId="amber" extension="namd_PSit${PS_it}">
```

To ensure the proper order of the execution of tasks, the QCG-Broker service has been additionally extended to allow the expression of the workflow dependencies between parameter sweep tasks using variables:

```
<parent triggerState="FINISHED">namd_PSit${PS_it}</parent>
```

2.1.4.3 Workflow support in the QCG simple description format

To allow end-users to run simple workflows without the necessity to use the non-user friendly QCG Job Profile XML description format, the Simple Description dialect has been extended to support basic workflows. A set of new #QCG directives has been added to allow users to specify workflow dependencies using the QCG Simple format:

- `#QCG global` - optional section that allows a user to specify QCG directives as a part of the script that will be used for all tasks in the workflow. The global section can occur multiple times in the description.
- `#QCG task` - this directive opens definition of the given task. The task definition ends with the end of the file, another task directive or global directive. The directive specifies the name of the task.
- `#QCG depends` - this directive allows a user to specify that the task depends on another one.
- `#QCG extends` - this directive allows a user to specify that for the transfer optimisation purpose the task should be executed in the working directory of another task.

The example of workflow description in the simple QCG format is presented in Listing 4.

```
#QCG global
#QCG walltime=PT10M
#QCG output=output.${JOB_ID}.${TASK_ID}
#QCG error=error.${JOB_ID}.${TASK_ID}
#QCG persistent
#QCG host=agave

#QCG task=task1
echo task1
sleep 15
hostname

#QCG task=task2
#QCG depends=task1
#QCG extends=task1
echo task2
sleep 15
hostname
```

Listing 4. Description of a workflow job in the QCG-Simple format

The above description is equivalent to the XML description attached in Anex 5.1. One can notice that the Simple format is much more intuitive, shorter, more understandable and generally more user-friendly.

2.1.4.4 Quasi-dynamic information about the status of the EEE

Besides the integration with the Monitoring and Downtime services, QCG stack was extended to allow end-users to obtain quasi-dynamic information about the status of the resources forming the EEE. QCG services collect and update information about the status of the infrastructure every 5 minutes. The information presented to end-users allows, based on the current availability of resources, to plan the execution of experiments (select the resource and amount of requested resources) without the use of Pattern Performance Service. The QCG client allows information about the EEE to be requested at various levels of granularity and detail. The syntax of the qcg-resources command is presented in Listing 5.

```
qcg-resources --help
Privides information about controlled resources
usage: qcg-client resources [-a <show_hidden>] [-h] [-m] [-n <PROPERTIES>] [-q] [-R
<RESOURCES>] [-s] [-u] [-V] [-v]
Options:
-a,--applications <show_hidden>   print information about applications
-h,--help                         display help message
-m,--modules                      print information about modules
-n,--nodes <PROPERTIES>           print information about nodes
-q,--queues                       print information about queues
-R,--resources <RESOURCES>        List of comma separated names of resources
-s,--summary                      print summary information about the resource
-u,--users                        print information about users
-V,--version                      display version
-v,--vos                          print information about vos
```

Listing 5. qcg-resources command usage

Listing 6. shows two outputs of the invocation of the qcg-resources command. The former invocation displays a summary about the status of the EEE, the later details information about the nodes of the Eagle cluster.

```
[plgkopta@qcg ~]$ qcg-resources
httpg://compat-broker.man.poznan.pl:8443/qcg/services/
/C=PL/O=GRID/O=PSNC/CN=qcg-broker/qcg-broker.man.poznan.pl
UserDN = /C=PL/O=GRID/O=PSNC/CN=Piotr Kopta
ProxyLifetime = 28 Days 22 Hours 30 Minutes 59 Seconds
SUMMARY:
HOST        NODES  UP           OFF        DOWN        APPS   MODULES  USERS  JOBS   WAIT  RUN    QUEUES  RES     VOS
supermuc    9493   9489[100.0%] 1[0.0%]    3[0.0%]     2      621      0      667    179   482    10      0       0
hartree     118    116[98.3%]   0          2[1.7%]     5      54       5      0      0     0      5       0       0
eagle       1005   941[93.6%]   4[0.4%]    60[6.0%]    16     230      1239   4616   26    4588   15      0       0

[plgpiontek@qcg ~]$ qcg-resources --nodes --resources eagle
---- CLUSTER: eagle ----
Category: [intel, haswell, haswell_2600mhz, fdr, huawei, 128GB]
              Nodes: 460 (down=49[10,7%] offline=26[5,7%] up=385[83,7%])
              Cores: 12880 (avail=10780[83,7%] free=7924[73,5%])
     Cores per node: 385x28cores
    Memory per node: 385x128617MB
```

```
Free cores per node: 94x0 1x3 1x18 2x4 1x20 1x5 1x25 2x24 80x27 2x26 1x13 199x28
Category: [intel, haswell, haswell_2600mhz, fdr, huawei, 256GB]
              Nodes: 52 (down=1[1,9%] up=51[98,1%])
              Cores: 1456 (avail=1428[98,1%] free=1178[82,5%])
     Cores per node: 51x28cores
    Memory per node: 51x257641MB
Free cores per node: 6x0 1x7 1x10 43x27
Category: [intel, haswell, haswell_2600mhz, fdr, huawei, 64GB]
              Nodes: 487 (down=24[4,9%] offline=32[6,6%] up=431[88,5%])
              Cores: 13636 (avail=12068[88,5%] free=11382[94,3%])
     Cores per node: 431x28cores
    Memory per node: 431x64105MB
Free cores per node: 22x0 70x27 339x28
```

Listing 6. Example report returned by the qcg-resources command

2.2 Energy optimization – the ECOS service

One of the challenges addressed in the ComPat project relates to the energy efficiency requirement of exascale systems and need to monitor, predict and manage power consumption. In order to provide the necessary capabilities in this context, complex analysis was performed which resulted in the decision to develop a new Energy Consumption Optimization Service (ECOS). The first version of the component focuses on optimization of energy use for memory-bound applications on the level of a single scale kernel. The service utilizes information from performance counters and appropriately modifies the frequency of processors. The conducted tests have shown that the service allows a saving of up to 30% of energy. During the remainder of the project we will integrate this service with the QCG stack and extend it further to support CPU-bound applications and complex multiscale scenarios.

2.2.1 Review of the available power-saving solutions in a modern HPC hardware

The main method for energy saving in the high performance computing (but also in personal computers) is dynamic voltage and frequency scaling (DVFS). Many tools that use this method, have been proposed over the years to save energy during computations while at the same time trying to minimize impact on performance [4][5][6][7][8][9][10]. ECOS draws on experience gained while analysing those existing tools. Additionally it offers better utilization of new energy related functions, available in modern hardware [11][12][13][14][15].

2.2.2 Initial tests of hardware and software capabilities

For the initial testing phase three applications were chosen to imitate common workload types in HPC: CPU-bound, memory-bound and communication-bound processing. For each workload type a simple multi-threaded application was developed. CPU-bound processing was simulated by executing multiple iterations of small matrix multiplication code (small enough to fit in the processor cache). Memory-bound workloads were also simulated by matrix multiplication applictions, but in this case

the matrix was much larger, forcing data to be fetched from system memory into local cache. Communication-bound processing was simulated by repeatedly sending a message from one process to every other in MPI application.

The tests were performed to compare application performance and energy use with regards to different processor settings (TurboBoost on/off, different performance bias levels), different operating system's frequency scaling drivers (CPUfreq, intel_pstate) and policies (e.g. performance, powersave).
These tests demonstarted clearly a case where frequency scaling drivers weren't performing optimally. In the case of memory-bound processing there was no difference in performance between running application with highest and lowest CPU frequency, but there was considerable difference in energy usage. This case was the motivation to propose a better policy for frequency scaling which allows energy saving without compromising performance. The CPU clock speed, when running a memory-bound program, does not affect performance because most of the time the processor stalls while waiting for the data to get fetched from main memory.

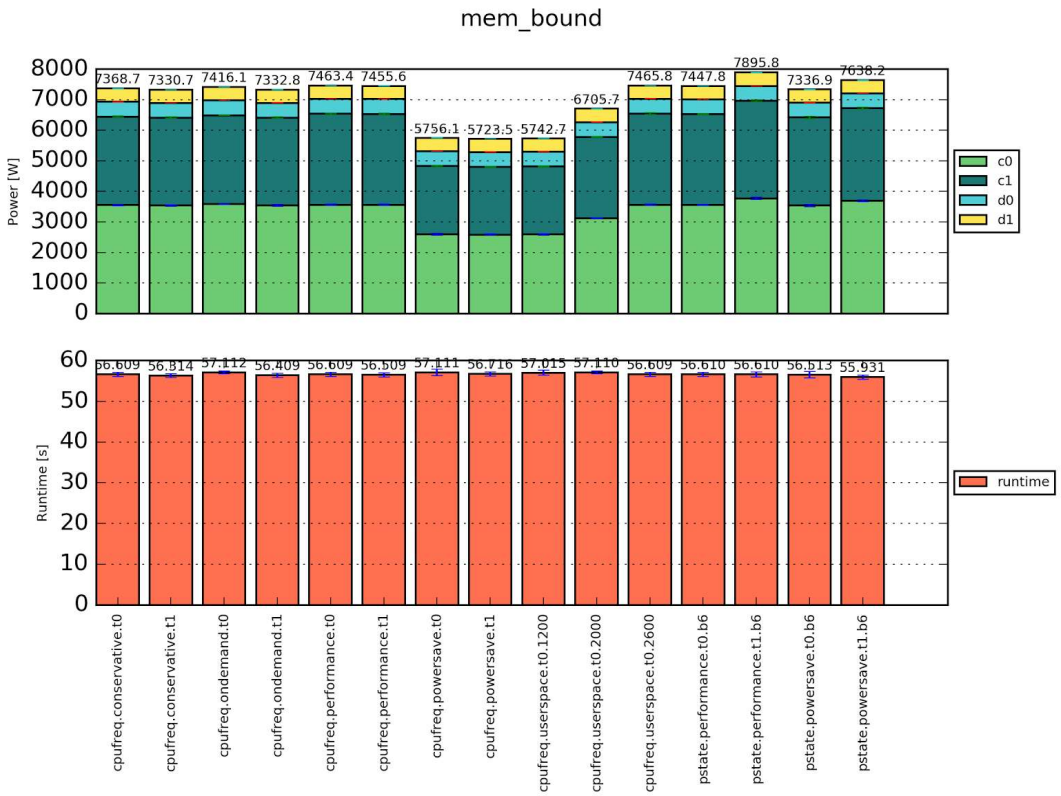

Figure 3. Results of the tests with different hw/sw settings

Tests results for memory-bound benchmarks are presented on the plot shown in Figure 3. The columns represent subsequent program runs with different parameters. Upper chart shows total power usage during the program run, and the lower chart present program total execution time. We can observe,

that when the settings lower CPU frequency, program runtime doesn't change, but there is significantly lower energy usage.

CPU memory stalls affect modern multi-core processor performance only when there are many cores referencing main memory simultaneously. This happens because memory bandwidth is limited and many cores are able to saturate it, but one core is not able to do so, even when running with maximum frequency.

The plot presented in Figure 4 shows total memory bandwidth in relation to the number of running threads. Different data series correspond to program runs with different CPU frequencies. We can observe that when application is run with a small number of threads, memory bandwidth rises almost linearly until it saturates when application is run with 6 threads. After that point, memory bandwidth is constant and doesn't change with further increase of number of threads nor a change in clock speed.

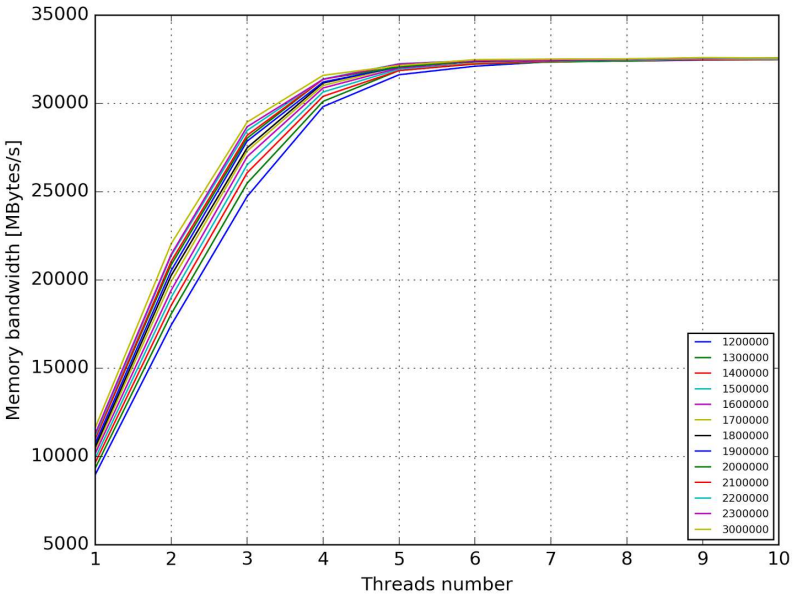

Figure 4. Total memory bandwidth usage for a different number of threads

2.2.3 Automatic workload characterization

Because the tests show only a narrow, specialized case, where there was an opportunity to save energy, there was a need to automatically detect when this case is occurring during the execution of a broad range of different applications. Therefore subsequent tests were performed to find a suitable metric for automatic workload type characterization. The most suitable metric turned out to be the CPU stall cycles ratio. It is a ratio of the number of stall memory cycles to the total number of cycles.

2.2.4 ECOS service

ECOS was created as a system service, that runs in the background and is independent of any other applications running in the system. ECOS controls clock speed separately for every hardware thread. During ECOS operation three stages can be distinguished, which are executed sequentially in an infinite loop:

1. Statistics gathering – in the first stage ECOS reads statistics from performance counters. ECOS is using counters for two events: number of stall memory cycles and total number of cycles. To read the counters ECOS is using the LIKWID library, which offers high level, platform agnostic API to access performance counters.
2. Decision – based on the latest statistics ECOS tries to decide if system is performing memory-bound computations. If it is, the CPU frequency could be limited which results in energy savings. The decision rule is applied independently for every hardware thread, and it is based on a value of the metric for an individual thread, but also an average value for a socket. An average value for a socket is taken into account because different sockets belong to different NUMA domains, and different NUMA domains have independent memory bandwidth budgets.
3. Setting new frequency – the last stage of ECOS operation is executed only if there is a decision to change current CPU frequency. ECOS could decide to lower clock speed, but naturally there is also an complementary action when clock speed is increased. The new CPU frequency is enforced using API of the CPUFreq module built-in into the Linux kernel. It allows to set a range of operational frequency for each logic processor in the system. ECOS modifies upper limit of this range, which effectively results in lowering the CPU frequency.

2.2.5 Results

The results for the ECOS test of memory-bound benchmark application are in line with original assumptions and show approximately 28% energy savings with negligible impact on performance. For the CPU-bound application there wasn't any substantial performance or energy consumption changes compared to the reference execution.

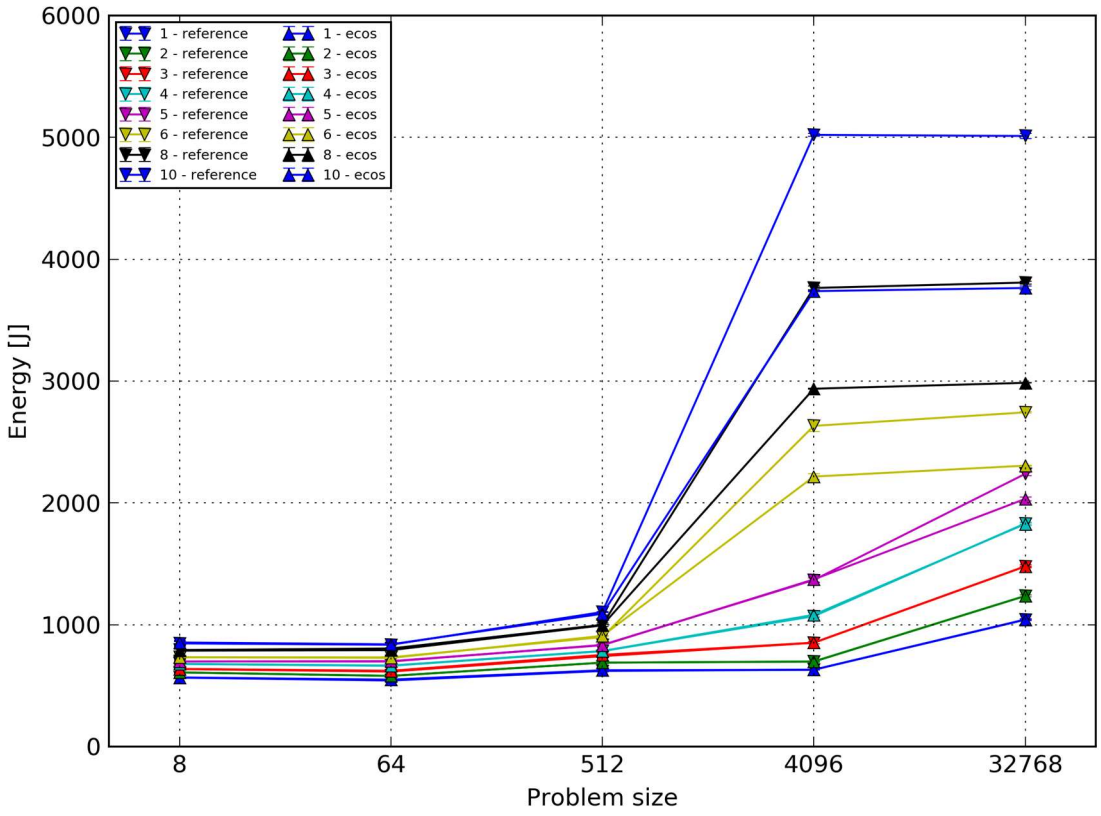

Figure 5. Comparison of energy usage with and without ECOS for different problem sizes

The plot in **Figure 5** presents total energy usage for the synthetic benchmarks. Small problem sizes (8, 64, 512) represent CPU-bound processing, and bigger problem sizes (4096 and 32768) represent memory-bound processing. The different data series on the plot represent program runs with different numbers of threads, and there are two runs for each setting of a number of threads – one with and one without ECOS running. We can observe that ECOS results in sizable energy savings in the case of the memory-bound program runs with a large number of threads.

Unfortunately tests with the *namd* application haven't shown any energy saving, but running ECOS didn't result in any differences in performance. The Namd application is strictly CPU-bound and ECOS isn't suitable for this workload type.

2.3 Support for High Performance Multiscale Computing Patterns

The three reusable HPMC patterns are fundamental for the ComPat project. Although the development of the patterns is managed by Work Package 2, WP5 has a responsibility to offer all the necessary functionality for the proposed patterns through the middleware. Through the joint efforts of WP2, WP3 and WP5 a set of features in the pattern and middleware services has been defined and utilised to

inform and steer developments in respective work packages. In this document, we focus on the results of particular developments made in WP5. In order to clearly present the role of middleware in ComPat pattern handling, the provided discussion starts with an overview of the developed pattern scenario. The detailed information about the pattern subsystem is available externally in the deliverable *D2.2 – First Report on Multiscale Computing Patterns and Algorithms* [2]

2.3.1 General idea of optimization for patterns

The idea behind the patterns is to identify a set of general multiscale executions scenarios, which cover the majority of multiscale applications execution cases. The identification of these common scenarios will allow the proposal of a the methodology and development of a set of general and specialized tools and services to enable end-users to create multi-scale applications and to execute them in an efficient way on the distributed exascale infrastructure. The benefits from utilzing patterns are many-fold, starting with the speedup of the process of creation of the application and ending with simplifying the execution of multiscale scientific simulations on distributed exascale infrastructure. The patterns give special attention to the efficiency of applications execution in terms of performance and energy consumption as well as fault tolerance.

The High Performance Multiscale Computing (HPMC) is defined as a set of “high-level call sequences that exploit (take advantage of) the functional decomposition of multiscale models in terms of single scale models”. [16]

2.3.1.1 HPMC Pattern based Development and Execution Environment

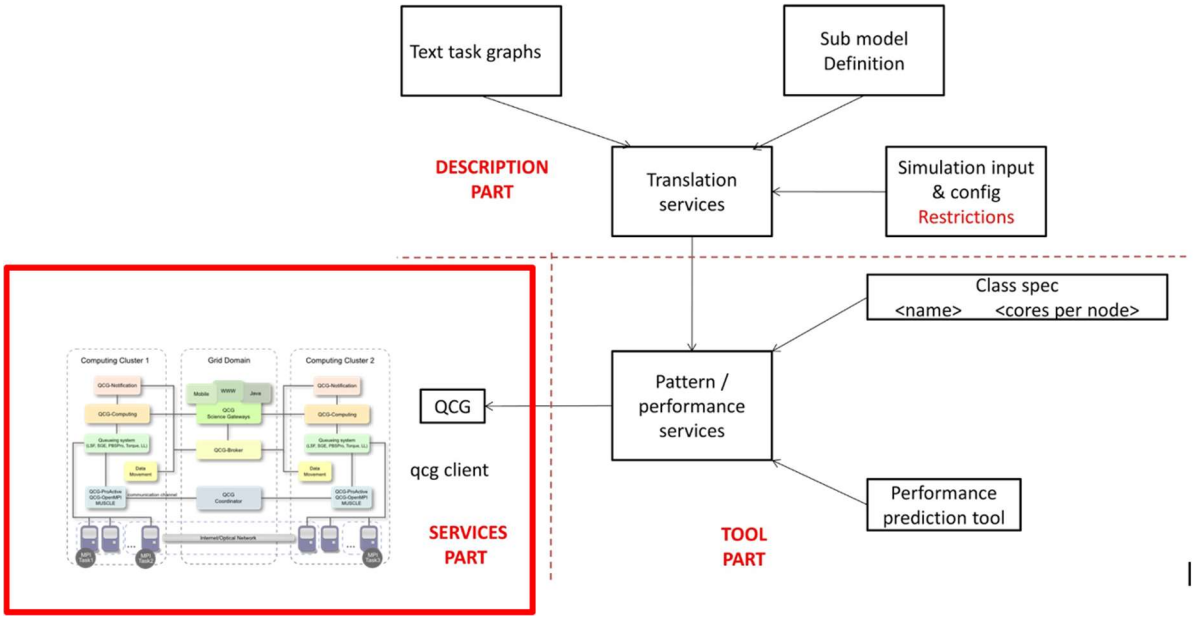

Figure 6. The Pattern based Development and Execution Environment

The HPMC Pattern Based Development and Execution Environment is the core of the ComPat system and the instantiation of the architecture of the system proposed in the deliverable D5.1 [1].

The Pattern based Development and Execution Environment, as shown in Figure 6, is composed of three main parts, namely “description”, “tool” and “services”. The description part is a set of tools responsible for building the formal description and the requirements of the multiscale application. The second part is the tool part, which is the main part that uses the concept of the Performance Matrix and applies the required algorithms according to the pattern type and the required measurement. Currently the performance matrix consists of entries presenting the wall clock time needed to calculate a submodel on a given amount of resources. At a later stage, this matrix will be re-used for energy as well. To fulfil the requirement of the scientific applications, we can do scaling interpolation as a combination of data and model. In this form we can add another dimension to the matrix to be a performance matrix per submodel, per problem size. For more information, we refer to D2.2 [2]. The output of the tool part is a QCG script, which includes a set of alternative allocation plans for the complex multiscale application. Each plan maps submodels to a different number of resources.

Finally, the third part is the QCG middleware stack which is responsible for selection of the best allocation plan for the multiscale application (mapping of computational kernels to the concrete number of physical resources of a specific type) and then the efficient and reliable execution of the application on the distributed heterogeneous infrastructure. The extension and adaptations to the QCG middleware stack to support the proposed concept of the HPMC Patterns are, next to the general extensions for multiscale applications, the main part of the WP5 work and the primary topic of this document.

2.3.2 New capabilities in QCG services for Patterns

In general, assigning of tasks to resources is a multi-criteria problem dependent on highly dynamic factors. At present, it is simply not possible unambiguously to select the best resources for a given use-case. Different user requirements, different priorities, unpredictable resources loads, diverse energy-usage constraints - these are only some of many problems faced by middleware developers. The special nature of complex multiscale computations addressed in the ComPat project makes the problem of scheduling and brokering even more difficult. Thus the crucial, but also the most time-consuming implementation task to enable execution of ComPat applications on the EEE was the creation of a new pattern-aware brokering module for QCG-Broker - the Pattern Execution Scheduler – see Figure 7.

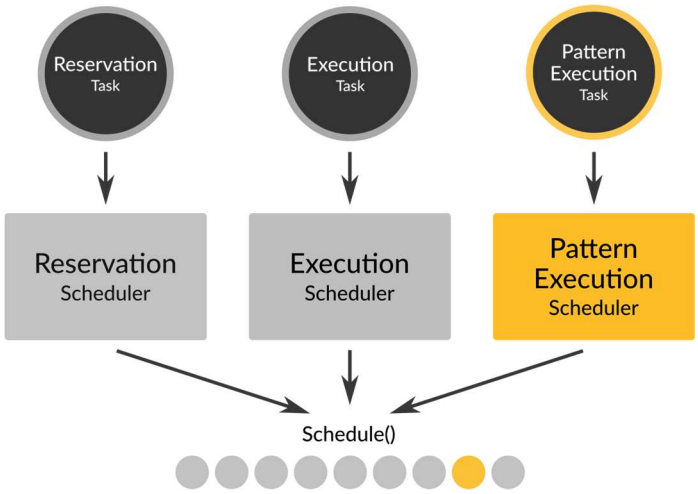

Figure 7. The new scheduler in QCG-Broker for execution of pattern-based tasks. Reservation scheduler and Execution scheduler existed before ComPat.

This new component was developed to efficiently handle a new type of task – pattern tasks with resource allocation plans created by the HPMC Pattern service. Thanks to this implementation, the brokering procedure is aware of the specific requirements of kernels constituting the multiscale, pattern-based application and may select the best resources to optimize the requested criterion (e.g. time to finish or total energy-usage). Since the new brokering module uses new input parameters to specify the requirements of the HPMC patterns, it was necessary to modify the format of the job description and also some internal schemas used to exchange information between components of QCG-Broker service. The QCG JobDescription schema has been extended by the "patternTopology" element that allows the description of the topology of the multiscale application. The whole structure of the patternTopology element is presented in Figure 8.

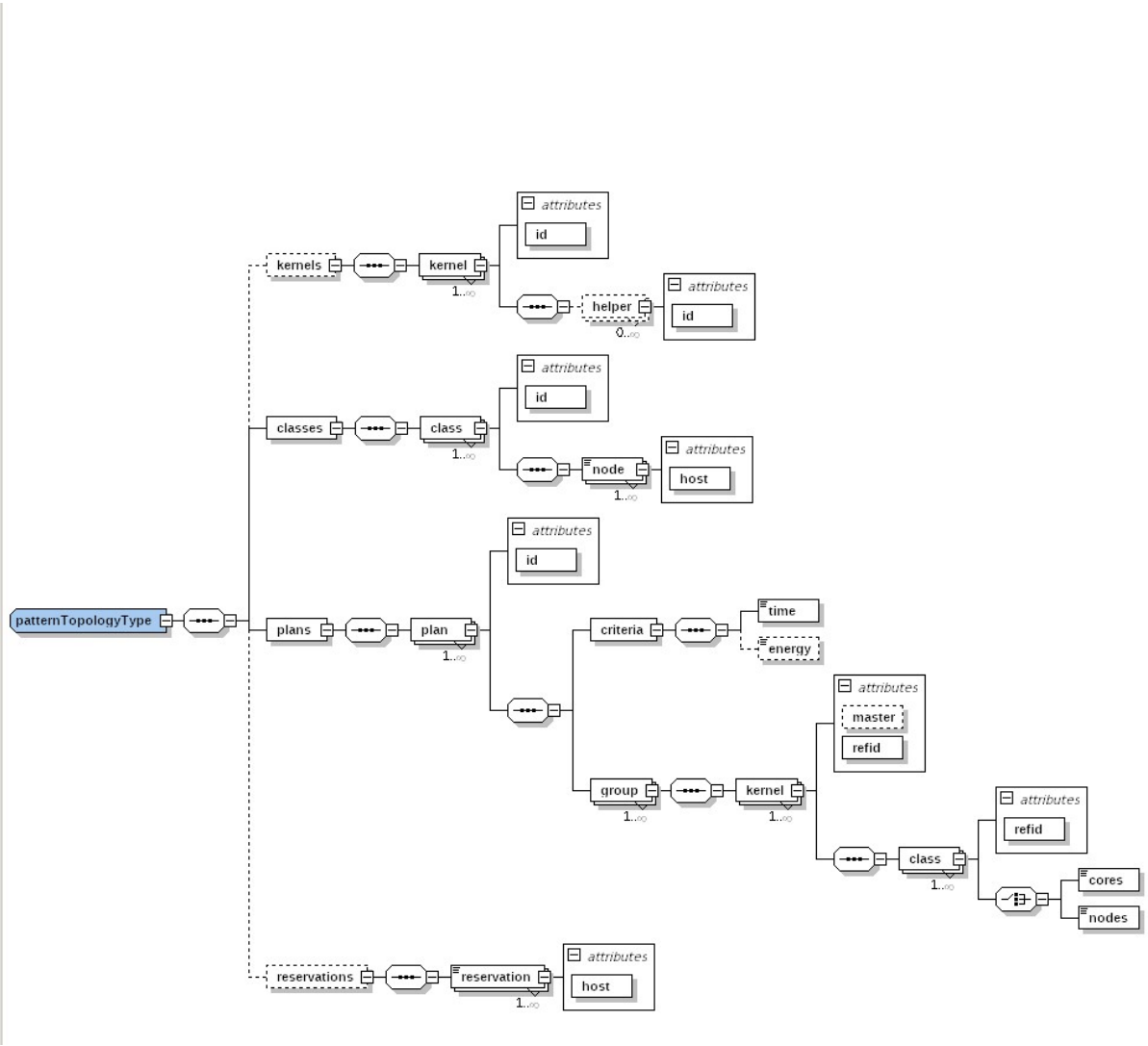

Figure 8. patternTopologyType – the new element that allows to describe pattern-based applications in QCG

The meaning of the main elements of the patternTopology is as follows:

- **<kernels>** - the "kernels" element contains a list of all kernels (<kernel> element) constituting the multiscale applications together with an optional list of auxiliary modules (mappers and filters) (<helper> element) that have to be started together with the main kernel,
- **<classes>** - the "classes" element contains a list of classes of resources. The class of resources is a set of types of nodes (<node> element) that are equal from the point of view of the performance of the specific kernel,
- **<plans>** - the "plans" element contains a list of alternative allocation plans generated by the Pattern Performance Service and corresponding to them a set of criteria that allows QCG-Broker to select the optimal plan based on the given criteria and the current status and availability of resources in the EEE. The only criterion taken into consideration at this stage of the project is execution time, but in the future the energy consumption will be used as a second criterion. The kernels can be arranged into groups to prevent the system placing them

on various disperate resources that could negatively impact on the overall performance of the multiscale application. For every single kernel it is possible to specify a list of alternative classes of resources with an amount of resources (cores or nodes) that should be allocated to that kernel in case of selection of the given class (one of the node types belonging to that class).

- **<reservations>** - a list of reservations on clusters that can be used to ensure the requested SLA or to synchronize the execution of parts of multiscale application distributed acros many resources.

Currently the broker provides basic functionality for Extreme Scaling and Replica Computing use-cases taking into account only the time to finish criterion. The example job descriptions for these patterns are attached in Anexes 5.2 and 5.3. It is planed to extend the functionality in future to support Heterogeneous Multiscale Computing pattern and to address also the energy-efficiency of calculations.

2.3.2.1 Pattern-aware scheduling algorithm for QCG-Broker

To efficiently schedule the ComPat applications, the traditional scheduling algorithms are simply not suitable. Therefore, next to two other previously existing brokering algorithms, in WP5 a new experimental methodology has been proposed and used to implement a new Pattern Execution Scheduler for QCG-Broker. The scheduling algorithm used in this new component takes into account several aspects of modern multiscale computing. First of all, it is aware of the multi-kernel nature of multiscale application and the fact that each kernel may behave differently in the context of performance and energy-usage when executed on different resources. The notable advantage of the new scheduler when comparing it to the existing earlier implementations relates to the functionality for handling alternative plans of execution. The plans constructed by the HPMC Pattern Generator on the basis of performance measurements are compared with respect to current load of computing resources. Then the best plan is selected to run.

In Figure 9 the newly developed scheduling algorithm is outlined. This algorithm was successfully embedded into QCG-Broker as Pattern Execution Scheduler and deployed on EEE resources.

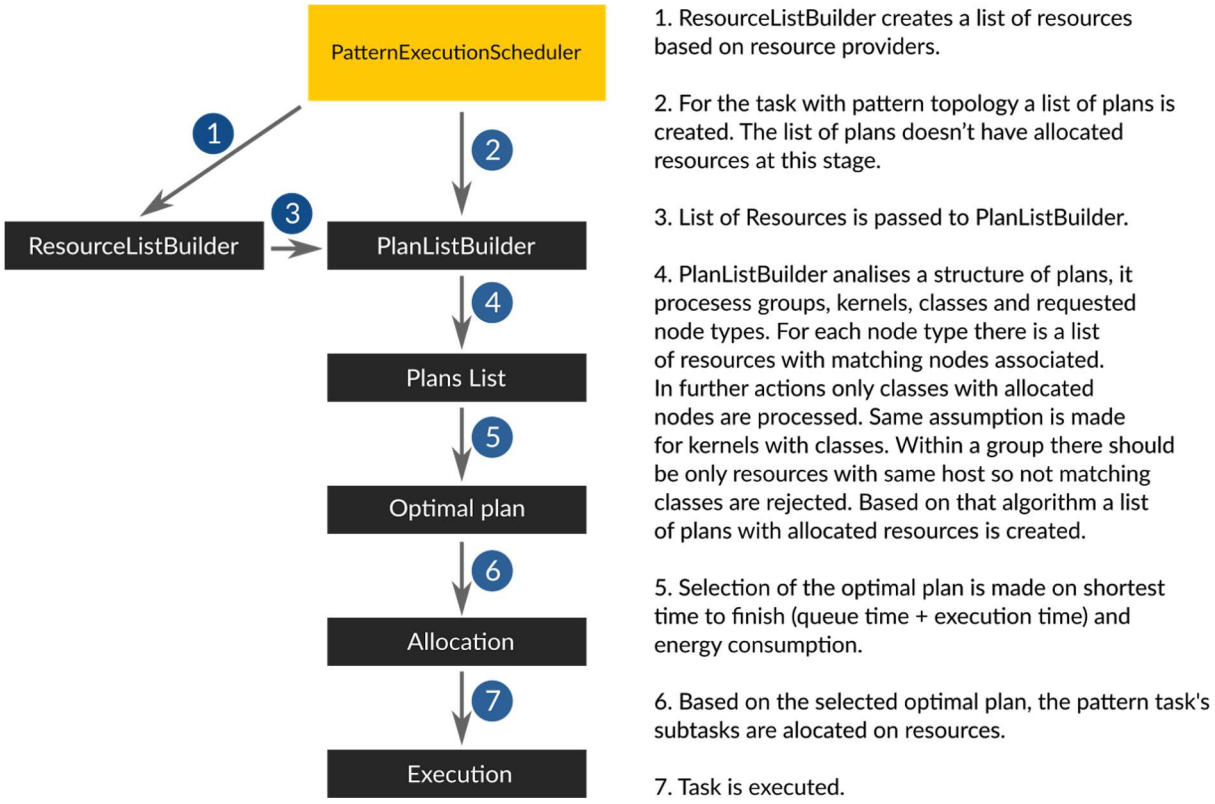

Figure 9. Scheduling algorithm used by Pattern Execution Scheduler

As it is described above, the Pattern Execution scheduler based on the plans provided by the Pattern Performance Service creates a space of all combinations of feasible resource allocations for kernels of the multiscale application. All the feasible allocations are evaluated taking into account the considered criteria. At this stage of the project the only criterion taken into consideration is time to completion. The algorithm searches for the allocation for which the total time to complete, defined as a sum of the queue time and execution time is minimal. For the prediction of the queue time, development of a new service is planned in the second part of the project. Currently the queue time is a random value proportional to the amount of requested resources.

3 Summary of main achievements and plans for phase 2 (M18-M36) of the project

The main achievements of WP5 at the M18 are as follows:

- design of the general architecture of the ComPat system (deliverable D5.1, milestone MS5);
- adaptation of QCG services to allow their deployment on EEE resources and creation of the unified environment for the project;

- adaptation and modification of QGG services and tools to support two of the three types of Pattern Scenarios: Replica Computing and Extreme Scaling;
- design and implementation of the first version of the Pattern-aware brokering algorithm for the QCG-Broker service;
- development of a mechanism for decreasing energy consumption when executing memory-bound applications;
- Release of ready-to-use middleware services and tools: currently, the most recent version of the QCG software source packages can be obtained from svn repository (https://apps.man.poznan.pl/svn/qcg-broker/tags/compat-milestone-1), for the majority of QCG components RPMs packages for SL6 and SL7 systems are also available. (milestone MS10);
- integration of QCG services with other ComPat specific tools and services to create an integrated Pattern-based Development and Execution Environment (deliverable D5.2, milestone MS10).

Plans for the second stage of the project:

- support for the Heterogeneous Multi-scale Computing Pattern;
- implementation of a service for queing time prediction;
- taking into account the aspect of energy efficiency in the brokering algorithm;
- fault tolerance issues associated with multiscale jobs within and exascale environment.

4 Conclusions

This deliverable describes all major achievements of WP5 reached until the end of M18.

The range of new functionality developed in the middleware layer and the successful integration with both the EEE and Pattern layers described in this document demonstrates significant progress, in both WP5 and the project more widely. This workpackage has managed to achieve its primary goals as defined in the project's DoW for this period. All the milestones for M18 has been fully accomplished with the deployment of the Middleware stack on the EEE.

As we have not had or expect any significant delays, we anticipate to continue working on WP5 as planned during the next period. Since deployment limitations have been mostly solved, we will focus on developing new components and adding functionality to the ComPat middleware.

5 Annexes

5.1 Workflow description expressed in the QCG XML dialect

```
<qcgJob appId="">
   <task persistent="true" taskId="task1">
      <candidateHosts>
          <hostName type="INCLUDE">agave</hostName>
      </candidateHosts>
      <execution type="single">
          <executable>
              <execFile>
                  <file name="wf.qcg">
                      <location type="URL">qcg://USE_SCRIPT_CONTENT/wf.qcg</location>
                  </file>
              </execFile>
          </executable>
          <stdout>
              <file>
                  <location
type="URL">gsiftp://qcg.man.poznan.pl:2811//home/plgrid/plgpiontek/reef/WF/output.${JOB_ID}.${TASK_ID}</location>
              </file>
          </stdout>
          <stderr>
              <file>
                  <location
type="URL">gsiftp://qcg.man.poznan.pl:2811//home/plgrid/plgpiontek/reef/WF/error.${JOB_ID}.${TASK_ID}</location>
              </file>
          </stderr>
          <stageInOut>
              <directory creationFlag="OVERWRITE" name="${JOB_ID}.${TASK_ID}" type="out">
                  <location
type="URL">gsiftp://qcg.man.poznan.pl:2811//home/plgrid/plgpiontek/reef/WF/${JOB_ID}.${TASK_ID}</location>
              </directory>
          </stageInOut>
      </execution>
      <executionTime useReservation="false">
          <executionDuration>PT10M</executionDuration>
      </executionTime>
   </task>
   <task extension="task1" persistent="true" taskId="task2">
      <candidateHosts>
          <hostName type="INCLUDE">agave</hostName>
      </candidateHosts>
      <execution type="single">
          <executable>
              <execFile>
                  <file name="wf.qcg">
                      <location type="URL">qcg://USE_SCRIPT_CONTENT/wf.qcg</location>
                  </file>
              </execFile>
          </executable>
          <stdout>
              <file>
                  <location
type="URL">gsiftp://qcg.man.poznan.pl:2811//home/plgrid/plgpiontek/reef/WF/output.${JOB_ID}.${TASK_ID}</location>
              </file>
          </stdout>
          <stderr>
              <file>
                  <location
type="URL">gsiftp://qcg.man.poznan.pl:2811//home/plgrid/plgpiontek/reef/WF/error.${JOB_ID}.${TASK_ID}</location>
              </file>
          </stderr>
          <stageInOut>
              <directory creationFlag="OVERWRITE" name="${JOB_ID}.${TASK_ID}" type="out">
                  <location
type="URL">gsiftp://qcg.man.poznan.pl:2811//home/plgrid/plgpiontek/reef/WF/${JOB_ID}.${TASK_ID}</location>
              </directory>
          </stageInOut>
      </execution>
      <executionTime useReservation="false">
          <executionDuration>PT10M</executionDuration>
      </executionTime>
      <workflow>
          <parent triggerState="FINISHED">task1</parent>
      </workflow>
   </task>
</qcgJob>
```

5.2 QCG job description for Replica Computing pattern (BAC application)

```
<?xml version="1.0"?>
<qcgJob appId="bac-compat-muc">
    <task persistent="true" taskId="namd">
        <requirements>
            <resourceRequirements>
                <computingResource>
                    <hostParameter name="queue">
                        <stringValue value="test"/>
                    </hostParameter>
                </computingResource>
            </resourceRequirements>
            <patternTopology>
                <kernels>
                    <kernel id="namd_kernel"/>
                </kernels>
                <classes>
                    <class id="c1">
                        <node host="supermuc">thin</node>
                    </class>
                </classes>
                <plans>
                    <plan id="plan1">
                        <criteria>
                            <time>P0Y0M0DT3H0M</time>
                        </criteria>
                        <group>
                            <kernel refid="namd_kernel">
                                <class refid="c1">
                                    <nodes>32</nodes>
                                </class>
                            </kernel>
                        </group>
                    </plan>
                </plans>
            </patternTopology>
        </requirements>
        <execution type="compat">
            <executable>
                <application name="bash"/>
            </executable>
            <arguments>
                <value>bac-namd.sh</value>
                <value>${PS_rep}</value>
            </arguments>
            <stageInOut>
                <file name="bac-namd.sh" type="in">
                    <location type="URL">gsiftp://qcg.man.poznan.pl/compat/bac-xml/muc-bac-namd.sh</location>
                </file>
                <directory name="input" type="in">
                    <location type="URL">gsiftp://qcg.man.poznan.pl/compat/bac-xml/input</location>
                </directory>
                <file name="qcg.debug" type="out">
                    <location type="URL">gsiftp://qcg.man.poznan.pl/compat/bac-xml/output/output-muc-${JOB_ID}-namd-
rep${PS_rep}.qcg.debug</location>
                </file>
                <file name="_stdouterr" type="out">
                    <location type="URL">gsiftp://qcg.man.poznan.pl/compat/bac-xml/output/output-muc-${JOB_ID}-namd-
rep${PS_rep}.stdouterr</location>
                </file>
            </stageInOut>
        </execution>
        <executionTime useReservation="false">
            <!-- walltime 1h for this task -->
            <executionDuration>P0Y0M0DT3H0M</executionDuration>
        </executionTime>
        <parametersSweep>
            <!-- this task will be replicated (parametrized) according to the 'rep' parameter
                 (in this case 2 times) -->
            <parameter>
                <name>rep</name>
                <value>
                    <loop>
                        <start>1</start>
                        <end>2</end>
                        <step>1</step>
                        <decimalPlaces>0</decimalPlaces>
                    </loop>
                </value>
            </parameter>
        </parametersSweep>
    </task>
    <!-- each 'amber' task will be executed in the corresponding 'namd' working directory
         - there is no need to transfer files between two stages of the workflow -->
    <task taskId="amber" extension="namd_PSit${PS_it}">
        <requirements>
            <resourceRequirements>
                <computingResource>
                    <hostParameter name="queue">
                        <stringValue value="test"/>
                    </hostParameter>
                </computingResource>
            </resourceRequirements>
            <patternTopology>
                <kernels>
```

```
                    <kernel id="amber_kernel"/>
                </kernels>
                <classes>
                    <class id="c1">
                        <node host="supermuc">thin</node>
                    </class>
                </classes>
                <plans>
                    <plan id="plan1">
                        <criteria>
                            <time>P0Y0M0DT3H0M</time>
                        </criteria>
                        <group>
                            <kernel refid="amber_kernel">
                                <class refid="c1">
                                    <cores>2</cores>
                                </class>
                            </kernel>
                        </group>
                    </plan>
                </plans>
            </patternTopology>
        </requirements>
        <execution type="compat">
            <executable>
                <application name="bash"/>
            </executable>
            <arguments>
                <value>bac-amber.sh</value>
                <value>${PS_rep}</value>
            </arguments>
            <stageInOut>
                <file name="bac-amber.sh" type="in">
                    <location type="URL">gsiftp://qcg.man.poznan.pl/compat/bac-xml/muc-bac-amber.sh</location>
                </file>
                <directory name="input/replicas/rep${PS_rep}" type="out">
                    <location type="URL">gsiftp://qcg.man.poznan.pl/compat/bac-xml/output/output-muc-${JOB_ID}-
rep${PS_rep}</location>
                </directory>
                <file name="qcg.debug" type="out">
                    <location type="URL">gsiftp://qcg.man.poznan.pl/compat/bac-xml/output/output-muc-${JOB_ID}-amber-
rep${PS_rep}.qcg.debug</location>
                </file>
                <file name="amber.log" type="out">
                    <location type="URL">gsiftp://qcg.man.poznan.pl/compat/bac-xml/output/output-muc-${JOB_ID}-amber-
rep${PS_rep}.log</location>
                </file>
                <file name="_stdouterr" type="out">
                    <location type="URL">gsiftp://qcg.man.poznan.pl/compat/bac-xml/output/output-muc-${JOB_ID}-amber-
rep${PS_rep}.stdouterr</location>
                </file>
            </stageInOut>
        </execution>
        <executionTime useReservation="false">
            <executionDuration>P0Y0M0DT3H0M</executionDuration>
        </executionTime>
        <workflow>
            <!-- the 'amber' task will start after 'namd' successfully finish;
                 each 'amber' parameter sweep task is started after coresponding 'namd'
                 task ('rep' variable) -->
            <parent triggerState="FINISHED">namd_PSit${PS_it}</parent>
        </workflow>
        <parametersSweep>
            <parameter>
                <name>it</name>
                <value>
                    <loop>
                        <start>0</start>
                        <end>1</end>
                        <step>1</step>
                        <decimalPlaces>0</decimalPlaces>
                    </loop>
                </value>
            </parameter>
            <parameter>
                <name>rep</name>
                <value>
                    <loop>
                        <start>1</start>
                        <end>2</end>
                        <step>1</step>
                        <decimalPlaces>0</decimalPlaces>
                    </loop>
                </value>
            </parameter>
        </parametersSweep>
    </task>
</qcgJob>
```

5.3 QCG job description for Extreme Scalling pattern (FUSION application)

```
<?xml version="1.0"?>
<qcgJob appId="compat-FUSION-test" project="compat" xmlns:jxb="http://java.sun.com/xml/ns/jaxb"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="E:\Documents\Work\ComPat\Londyn-
Brunel\desc\QCGJobDescriptionSchema.xsd">
  <task persistent="true" taskId="ES-task-2">
    <requirements>
      <patternTopology>
    <!-- here we define kernels as logical units for the use in the resource requirements section;
              definition might be extended with additional parameters in the future  -->
    <kernels>
      <kernel id="turb">
        <helper id="init"/>
        <helper id="transp"/>
        <helper id="equil"/>
        <helper id="f2dv"/>
        <helper id="dupEquil"/>
        <helper id="dupCorep"/>
      </kernel>
    </kernels>
    <!-- definition of the classes - as a sets of node types -->
    <classes>
      <class id="c1">
        <node host="supermuc"> thin </node>
        <node host="eagle"> haswell_128 </node>
      </class>
      <class id="c2">
        <node host="supermuc"> thin </node>
      </class>
      <class id="c3">
        <node host="eagle"> haswell_128 </node>
      </class>
    </classes>
    <plans>
      <plan id="plan1">
        <criteria>
          <time>PT14H00M</time>
          <!-- check -->
        </criteria>
        <group>
          <kernel refid="turb">
        <class refid="c1">
          <cores>1024</cores>
        </class>
          </kernel>
        </group>
      </plan>
      <plan id="plan2">
        <criteria>
          <time>PT28H00M</time>
          <!-- check -->
        </criteria>
        <group>
          <kernel refid="turb">
        <class refid="c2">
          <cores>512</cores>
        </class>
          </kernel>
        </group>
      </plan>
    </plans>
    <!-- use of reservations is optional
          <reservations>
         <reservation host="eagle">eagle-res-1</reservation>
         <reservation host="inula">inula-res-2</reservation>
         <reservation host="zeus">zeus-res-2</reservation>
         </reservations>-->
      </patternTopology>
    </requirements>
    <execution type="compat">
      <executable>
    <application name="muscle2" version="compat-1.2"/>
      </executable>
      <arguments>
    <value>test.cxa.rb</value>
      </arguments>
      <stdout>
    <directory>
      <location type="URL">gsiftp://qcg.man.poznan.pl//home/plgrid-groups/plggcompat/Fusion/FastTrack/qcg-
test/results</location>
    </directory>
      </stdout>
      <stderr>
    <directory>
      <location type="URL">gsiftp://qcg.man.poznan.pl//home/plgrid-groups/plggcompat/Fusion/FastTrack/qcg-
test/results</location>
    </directory>
      </stderr>
      <stageInOut>
    <file name="test.cxa.rb" type="in">
      <location type="URL">gsiftp://qcg.man.poznan.pl//home/plgrid-groups/plggcompat/Fusion/FastTrack/qcg-
test/test.cxa.rb</location>
    </file>
    <file name="inputs.tgz" type="in">
      <location type="URL">gsiftp://qcg.man.poznan.pl//home/plgrid-groups/plggcompat/Fusion/FastTrack/qcg-
```

```
test/inputs.tgz</location>
    </file>
    <file name="extract_inputs.sh" type="in">
      <location type="URL">gsiftp://qcg.man.poznan.pl//home/plgrid-groups/plggcompat/Fusion/FastTrack/qcg-
test/extract_inputs.sh</location>
    </file>
    <directory name="outputs" type="out">
      <location type="URL">gsiftp://qcg.man.poznan.pl//home/plgrid-groups/plggcompat/Fusion/FastTrack/qcg-
test/results</location>
    </directory>
      </stageInOut>
      <environment>
    <variable name="QCG_MODULES_LIST">compat/apps/fusion</variable>
    <variable name="QCG_PREPROCESS">extract_inputs.sh</variable>
    <!-- is it needed? -->
    <!--<variable name="QCG_POSTPROCESS">ex1.postprocess</variable> is it needed? -->
      </environment>
    </execution>
    <executionTime>
      <!-- we do not need this anymore -->
      <executionDuration>P0Y0M0DT14H00M</executionDuration>
    </executionTime>
  </task>
</qcgJob>
```

6 References

[1] Deliverable *D5.1 – Architecture of the ComPat system*: http://www.compat-project.eu/wp-content/uploads/2016/07/ComPat_D5.1.pdf

[2] Deliverable *D2.2 – First Report on Multiscale Computing Patterns and Algorithms*

[3] tsocks library: http://tsocks.sourceforge.net

[4] Hsu, Chung-Hsing, and Wu-Chun Feng. "Effective dynamic voltage scaling through CPU-boundedness detection." International Workshop on Power-Aware Computer Systems. Springer Berlin Heidelberg, 2004.

[5] Freeh, Vincent W., et al. "Exploring the energy-time tradeoff in mpi programs on a power-scalable cluster." *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*. IEEE, 2005.

[6] Ge, Rong, et al. "CPU miser: A performance-directed, run-time system for power-aware clusters." *Parallel Processing, 2007. ICPP 2007. International Conference on*. IEEE, 2007.

[7] Rountree, Barry, et al. "Adagio: making DVS practical for complex HPC applications." *Proceedings of the 23rd international conference on Supercomputing*. ACM, 2009.

[8] Ge, Rong, et al. "Powerpack: Energy profiling and analysis of high-performance systems and applications." *IEEE Transactions on Parallel and Distributed Systems* 21.5 (2010): 658-671.

[9] Valentini, Giorgio Luigi, et al. "An overview of energy efficiency techniques in cluster computing systems." *Cluster Computing* 16.1 (2013): 3-15.

[10] Marathe, Aniruddha, et al. "A run-time system for power-constrained HPC applications." *International Conference on High Performance Computing*. Springer International Publishing, 2015.

[11] Rountree, Barry, et al. "Beyond DVFS: A first look at performance under a hardware-enforced power bound." *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*. IEEE, 2012.

[12] Patki, Tapasya, et al. "Exploring hardware overprovisioning in power-constrained, high performance computing." *Proceedings of the 27th international ACM conference on International conference on supercomputing*. ACM, 2013.

[13] Hackenberg, Daniel, et al. "An energy efficiency feature survey of the intel haswell processor." *Parallel and Distributed Processing Symposium Workshop (IPDPSW), 2015 IEEE International*. IEEE, 2015.

[14] Inadomi, Yuichi, et al. "Analyzing and mitigating the impact of manufacturing variability in power-constrained supercomputing." *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2015.

[15] Acun, Bilge, Phil Miller, and Laxmikant V. Kale. "Variation among processors under turbo boost in hpc systems." *Proceedings of the 2016 International Conference on Supercomputing*. ACM, 2016.

[16] Alowayyed, Saad, et al. "Multiscale Computing in the Exascale Era." *arXiv preprint arXiv:1612.02467,* 2016.