New Capabilities in QosCosGrid Middleware for Advanced Job Management, Advance Reservation and Co-allocation of Computing Resources – Quantum Chemistry Application Use Case

Bartosz Bosak[1], Jacek Komasa[2], Piotr Kopta[1], Krzysztof Kurowski[1], Mariusz Mamoński[1], and Tomasz Piontek[1]

[1] Poznań Supercomputing and Networking Center, Poznań, Poland
{bbosak,pkopta,krzysztof.kurowski,mamonski,piontek}@man.poznan.pl
[2] Adam Mickiewicz University, Faculty of Chemistry,
Grunwaldzka 6 Street, 60-780 Poznań, Poland
komasa@man.poznan.pl

Abstract. In this chapter we present the new capabilities of QosCosGrid (QCG) middleware for advanced job and resource management in the grid environment. By connecting many computing clusters together, QosCosGrid offers easy-to-use mapping, execution and monitoring capabilities for a variety of complex computations, such as parameter sweep, workflows, MPI or hybrid MPI-OpenMP as well as multiscale simulations. Thanks to QosCosGrid, large-scale programming models written in Fortran, C, C++ or Java can be automatically distributed over a network of computing resources with guaranteed Quality of Service – for example guaranteed startup time of a job. Consequently, applications can be run at specified periods with reduced execution time and waiting times. This enables more complex problem instances to be addressed. In order to prove the usefulness of the new functionality of QosCosGrid a detailed description of the system along with a real use case scenario from the quantum chemistry science domain will be presented in this chapter.

Keywords: parallel computing, MPI, metascheduling, advance reservation, QoS, High Performance Computing, High Throughput Computing.

1 Introduction

End users interested in the PL-Grid infrastructure have frequently voiced their demand for efficient programming and execution tools to run large parallel simulations and experiments requiring a certain Quality of Service. Highly parallel and coupled applications with significant inter-process communication that are not supported by existing grid infrastructures based on gLite [14] or UNICORE [19], represent a growing and promising class of simulations. To meet end-user

M. Bubak, T. Szepieniec, and K. Wiatr (Eds.): PL-Grid 2011, LNCS 7136, pp. 40–55, 2012.

needs, a new middleware infrastructure called QosCosGrid (or QCG for short) was designed, developed and deployed in the PL-Grid project [11]. QCG successfully integrates many new services and tools in order to deliver to PL-Grid users a new multilayered e-Infrastructure capable of dealing with computationally intensive simulations, including parameter sweep studies, workflows and, more importantly, large-scale parallel applications. QCG enables computing clusters in different administrative domains to be integrated into a single powerful virtual computing resource that can be treated as a quasi-opportunistic supercomputer, whose computational power exceeds the power offered by a single administrative domain (data center). In order to bring this supercomputer-like performance and structure (requested and expected by scientists) to bear on cross-cluster computations in an user-friendly way, various well-known application tools and services, including the OpenMPI and ProActive programming and execution environments, have been tuned to work in a multi-cluster QCG environment [1,11]. The cross-cluster scheduling enabled by QCG supports not only parallel simulations on many clusters (creating new possibilities and offering improvements for end users), but also – and more importantly from the resource owners' point of view – utilizing PL-Grid computing resources in a more efficient way, thus increasing the overall system throughput. The decomposition of large-scale tasks between many clusters decreases cluster "defragmentation" and results in better resource utilization.

The rest of the chapter is organized as follows. In Section 2 we present the key features of QCG. Section 2.1 describes QCG support for creating and running parallel applications. Section 2.2 describes how QCG enables execution of workflows. Section 2.3 presents QCG functions for Advance Reservation and Co-allocation. The following section, 3, describes the QCG architecture from a general point of view and introduces the main QCG components. A real usage scenario, based on a legacy application and exploiting the capabilities and features of QCG is discussed in Section 4. Finally, in Section 6, we present a short summary and list the ongoing and future work in the scope of QCG.

2 QCG Capabilities

2.1 QCG for Parallel Applications

To support large-scale applications in multi-cluster environments many novel mechanisms have been implemented. As presented in [9], QCG services are able to schedule and execute parallel applications consisting of groups of processes with different and often mutually contradictory resource requirements. For example, functional decomposition of the application and its implementation can result in a situation in which some processes should be run on a vector machine (for performance reasons) while others reside on a regular computing cluster. By defining groups of parallel processes it becomes possible to determine different resource requirements for each group and specify whether a given group can be split between resources or whether it should be allocated to a single resource. To avoid performance and communication bottlenecks caused by the fact that

local connections have lower latency and higher bandwidth than long-distance ones (by two to four orders of magnitude), the QCG resource manager can either schedule the application in a topology-aware manner to meet its requirements or expose the physical topology to the application which then dynamically adapts itself to that topology. Such topology-awareness implies that, while matching the resource offers with requests, the scheduler has to take into account not only the computational properties of the resources, but also their interconnections. All these scenarios, except for the one involving self-adaptation of the application to match the available topology, do not require any changes in application code and are fully implementable owing to tight integration of QCG services with adapted OpenMPI and ProActive frameworks [11]. Running large-scale simulations, both sequential and parallel, in a multi-cluster environment requires not only launching and controlling processes on the available resources, but also some means of enabling inter-process communication between parts of a parallel application. Parallel processes running on different computing clusters must be able to communicate with one another without running afoul of the security mechanisms protecting each cluster (such as firewalls blocking connections and NATs reducing the number of public IPs required by the cluster). To address this primary requirement an open-source implementation of the MPI standard – OpenMPI – as well as the Java ProActive library have been extended with several basic and advanced connectivity techniques intended to bypass firewalls and NATs, and integrated with QCG services [1].

2.2 QCG for Workflow Applications

As has already been mentioned, QCG supports not only large-scale parallel applications but also other kinds of popular computational experiments. QCG is able to deal with complex applications defined as a set of tasks with precedence relationships (workflows). The workflow model is based on direct acyclic graphs (DAG). In this approach the end user has to specify (in advance) task precedence constraints in the form of task state relationships. A very interesting and novel feature of QCG, which distinguishes it from other middleware services supporting workflows, is that – in addition to being associated with input or output files – every task can be triggered by any combination of other tasks or conditional rules. This feature is very useful in many scenarios. For instance, one can imagine that a user would like to execute an application as soon as another one starts running, e.g. for client-server communication. Another example could involve redirecting the flow of computations in the event of a failure of one of the scheduled tasks (failover mechanisms). QCG also supports popular parameter sweep experiments and supports many instances of a single application, each with a different set of arguments. For every task in the collection, the value of one or more of the task parameters may be changed in some preordained fashion, creating a parameters space. This is also a very useful feature, providing the end user with an easy way to browse the parameters space in search for a specific set of parameters that meet the defined criterion. Parameter sweep tasks can be a part of a larger experiment, e.g. a workflow, and all parent-child dependencies are

automatically converted by the system to take the whole collection of generated tasks into consideration. What distinguishes QCG from other middleware packages dealing with parameter sweep tasks, is its support for multi-dimensional parameter spaces, in which many variables can be regulated to construct the aforementioned space of parameters.

2.3 Advance Reservation and Co-allocation in QCG

The next feature distinguishing QCG from other e-Infrastructures offering access to PL-Grid resources like gLite and UNICORE is support for scenarios which call for a specific level of quality of service. As the only such infrastructure, QCG supports advance reservation of computational resources to guarantee the requested execution parameters. Advance reservation is used internally by QCG services in the case of cross-cluster execution but is also provided directly to end users. The reservation mechanism is applied in the scheduling process to co-allocate resources and then to synchronize execution of application parts in a multi-cluster environment. Cross-cluster scheduling and co-allocation of resources are tightly connected with support for groups of processes and communication topologies. When co-allocating resources and assigning tasks to specific resources, QCG may also consider user requirements regarding task execution time. Upon submitting a task to the system the user may specify resource requirements as well as the requested quality of service, including task duration and – optionally – the period when the task should be executed. QCG supports both the strict and best-effort approaches to resource reservation. In the former approach resources are reserved only if it is possible to fully meet user requirements (also known as the "all or nothing" approach), whereas in the the latter case the system reserves as much resources as possible and there is no guarantee that all requested resources (cores) will be reserved [13].

Recently, the QCG infrastructure has been extended to support new types of parallel applications based on the MPI/OpenMP hybrid programming approach. The user may request the application to execute on a given number of computational nodes with a predefined number of slots (cores) on each. In this solution, for each node participating in the computations, the user can specify the number of MPI processes that should be started.

The functional comparison of QCG middleware with gLite and UNICORE is presented in Table 1.

Table 1. Functional comparison of three grid middleware solutions: QCG, gLite and UNICORE.

Middleware	Single jobs	Workflows	MPI jobs	Cross-cluster MPI jobs	Interactive jobs	Parametric jobs
gLite	Yes	Yes	Yes	No	Yes	Yes
UNICORE	Yes	Yes	Yes	No	No	Yes
QCG	Yes	Yes	Yes	Yes	No	Yes

3 QosCosGrid Architecture

Though QCG middleware generally follows a multi-layered design approach, two main levels can be distinguished: grid domain and administrative domain. Services belonging to the grid domain provide a high-level interface to the grid or cloud environments (e.g. GridSpace [7], Nano-Science Gateway [8]) and control and schedule the execution of applications which are distributed over independent administrative domains. In turn, the administrative domain represents a single resource provider (e.g. HPC or data center) which contributes computational resources (e.g. clusters) to a particular grid or cloud infrastructure. Note that logical separation of administrative domains is intentional and corresponds to the fact that resources (and also users) may come from different institutions or resource owners. It is fully natural that each institution may need to preserve a certain level of independence and enforce its own resource allocation/sharing policies.

The general architecture of QCG is presented in Fig. 1. The critical service on the grid level is QCG-Broker; a meta-scheduling framework controlling execution of applications via services located in the administrative domains. On this level the QCG-Computing component (tightly connected with QCG-Broker) provides remote access to underlying queuing systems. Among others, QCG-Computing supports execution of jobs in several parallel execution environments, namely OpenMPI, ProActive and MUSCLE. Additionally, it exposes an interface for creation and management of advance reservations. Another relevant service in

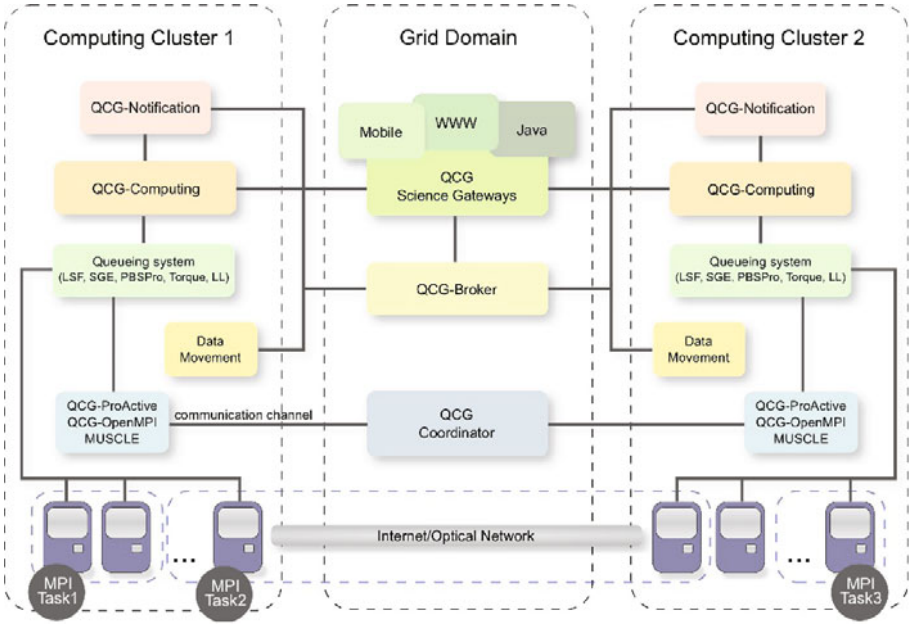

Fig. 1. General QCG middleware architecture

the administrative domain is called QCG-Notification: its task is to implement a notification mechanism. The QCG middleware structure is complemented by coordinator services controlling cross-cluster execution, as well as data movement services managing input and output data for applications.

3.1 QCG-Broker

The QCG-Broker[1] is an open-source metascheduling system which allows developers to build and deploy resource management systems for large-scale distributed computing infrastructures. Based on dynamic resource selection, mapping and advanced scheduling methodology combined with feedback control mechanisms, QCG-Broker deals efficiently with various metascheduling challenges, e.g. co-allocation, load balancing among clusters, remote job control, file staging support and job migration, as has been demonstrated in [10]. The main goal of QCG-Broker is to manage the whole process of remote job submission to administrative domains controlled by domain-level QCG components, and then to underlying clusters and computational resources. It has been designed as an independent core component for resource management processes which can take full advantage of various low-level core and grid services and existing technologies, such as QCG-Computing and QCG-Notification or GridFTP, as well as various grid middleware components, e.g. Grid Authorization Service, Data Management Service and others. All these services work together to provide a consistent, adaptive and robust grid middleware layer which fits dynamically to many different distributed computing infrastructures, enabling large scale simulations and providing the requested Quality of Service. One of the main assumptions for QCG-Broker is to perform remote jobs control and management in a way which satisfies users (Job Owners) and their applications while respecting the constraints and policies imposed by other stakeholders, i.e. resource owners and grid or Virtual Organization administrators. Simultaneously, Resource Administrators (Resource Owners) have full control over resources on which all jobs and operations are performed, through appropriate setup and installation of QCG components. Note that QCG-Broker, together with administrative-domain level QCG components, reduces the operational and integration costs for Administrators by enabling grid deployment across previously incompatible cluster and resources. The heart of QCG-Broker is its metascheduling framework, responsible for scheduling tasks in the controlled environment. QCG-Broker has been successfully integrated with the scheduling framework designed, implemented and used in the Grid Scheduling SIMulator [12], enabling grid administrators to modify scheduling policies in an easy and flexible way, using different scheduling plugins. All experiments controlled by QCG-Broker (including workflows, large-scale parallel applications with groups of processes and topology requirements, parameter sweep tasks and simple jobs) can be easily expressed in a formal way using the XML-based job definition language called Job Profile.

[1] `http://www.qoscosgrid.org/trac/qcg-broker`

3.2 QCG-Computing

The key component in the QCG administrative domain is the QCG-Computing service. Technically, QCG-Computing[2] is an open implementation of SOAP Web services for multiuser access and policy-based job control routines by various queuing and batch systems managing local computational resources. To communicate with underlying queuing systems, the service uses the Distributed Resource Management Application API (DRMAA) [20]. It has been successfully tested with many products and supports a variety of well-known queuing systems, including:

- Grid Engine,
- Platform LSF,
- Torque/Maui,
- PBS Pro,
- Condor,
- Apple XGrid,
- SLURM,
- LoadLeveler.

The QCG-Computing service is compliant with the OGF HPC Basic Profile specification [22], which serves as a profile over other Open Grid Forum standards like JSDL and OGSA Basic Execution Service. Moreover, it offers innovative remote interfaces for advance reservation management and supports basic file transfer mechanisms.

QCG-Computing has been designed to support a variety of plugins and modules for external communication as well as to handle a large number of concurrent requests from external clients and services. Consequently, it can be used and integrated with various authentication, authorization and accounting services. An example of integration with other PL-Grid services is described in Section 3.5.

3.3 QCG-Notification

QCG-Notification[3] is another service belonging to the administrative domain. Its main function in QCG is brokering asynchronous notifications concerning job state changes between the QCG-Computing and QCG-Broker services. Nevertheless, depending on demands, QCG-Notification may be variously configured and adapted to specific requirements. In general, QCG-Notification is based on the OASIS standards for Web Service notifications – WS-BaseNotification, WS-BrokeredNotification and WS-Topics [23], and provides an adjustable and efficient interface for message exchange between interested parties. Thanks to QCG-Notification, components which produce notifications may be logically separated from components interested in receiving those notifications, as presented in Fig. 2. Therefore, since some features required for communication between

[2] `http://www.qoscosgrid.org/trac/qcg-computing`
[3] `http://www.qoscosgrid.org/trac/qcg-notification`

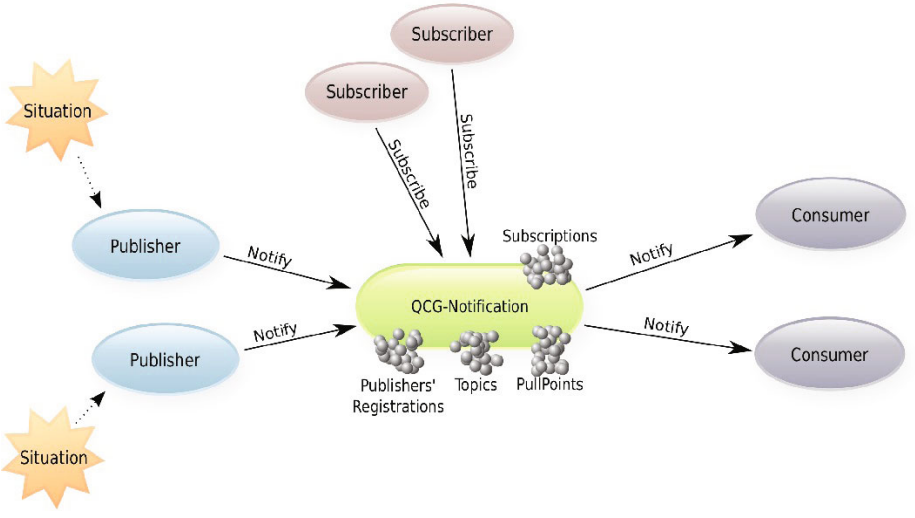

Fig. 2. Brokered notification scenario supported by QCG-Notification

specific components are delegated to an external entity, the overall performance of the system may significantly increase. Such a situation is also common in QCG. QCG-Notification implements many patterns defined in the WS-Notification specification and provides some extensions to the standard. Performed tests and comparisons with other well-known WS-Notification implementations have shown that QCG-Notification offers a rich feature set while remaining highly efficient. QCG-Notification is characterized by the following list of features:

- support for HTTP/HTTPS and XMPP transport protocols,
- subscription and publishers' registration handling,
- advanced two-level notification filtering based on hierarchical topic namespaces and notification message contents (XPath-based filters),
- pull and push styles of distributing notification messages with fault tolerance mechanisms,
- good performance owing to carefully selected data structures and internal algorithms,
- plenty of configuration and customization options, available through a bundled management interface,
- extensible architecture (pluggable modules for transport, authentication and authorization protocols).

3.4 Cross-Cluster Communication

QCG support for parallel cross-cluster execution consists of the three environments, namely QCG-OMPI [1], QCG-ProActive and MUSCLE, each targeting a different groups of application developers. The first environment, QCG-OMPI[4],

[4] `http://www.qoscosgrid.org/trac/qcg-openmpi`

is based on OpenMPI and preserves all standard properties of this library. Therefore it aims at C/C++ and FORTRAN code. QCG-ProActive, in turn, is an extended version of the ProActive Java library which may be easily used in new or existing parallel Java applications. The final environment, MUSCLE (still under development in the MAPPER [26] project), simplifies the development of multiscale computation scenarios.

Since the standard deployment methodologies used in OpenMPI or ProActive are limited to single-cluster runs, in order to support cross-cluster execution and spawning of parallel application processes on co-allocated computational resources, QCG (besides minor extensions to the OpenMPI and ProActive libraries) provides special services called coordinators. Coordinators are implemented as Web Services and should be accessible from all participating administrative domains. Depending on particular scenarios, coordinators may be configured in various ways. In general, two situations which influence deployment of QCG middleware can be distinguished:

1. All computing clusters have public IP addresses,
2. At least one computing cluster has private IP addresses.

In the former case, port range techniques can be applied to enable communication between processes executing in separate clusters. It is a simple approach founded upon the idea of using a predefined range of unprivileged ports. If, however, some clusters use private set(s) of IP addresses, a different solution is necessary. We have decided to take advantage of proxy mechanisms, where SOCKS services are deployed on frontend machines to route incoming traffic to the MPI and ProActive processes running inside clusters with private IP addresses.

3.5 Integration with PL-Grid Infrastructure

Almost every e-Infrastructure enforces its own authentication, authorization and accounting (called AAA for short) through a set of custom policies. Those polices are usually governed by a separate unit called the Operations Center. Therefore, any new middleware stack wishing to become a part of such an infrastructure must integrate itself with the existing AAA ecosystem.

The PL-Grid infrastructure offers two authentication mechanisms: password-based and X.509 certificate-based. The former is usually used while logging into the PL-Grid Portal, gLite UI machines and batch job submission hosts. The latter is required while contacting grid services. In PL-Grid every QCG-Computing instance is configured to accept RFC3820 [21] compliant proxy certificates.

The QCG-Computing services in PL-Grid are configured to use the plain grid-mapfile. This means that, much like UNICORE and contrary to gLite, QCG uses static accounts. The grid-mapfile is generated automatically based on information available in a local LDAP (Lightweight Directory Access Protocol) replica. Hence, each PL-Grid user who applies for QCG services and is cleared by local administrators, may be automatically added to this file. Moreover, system administrators are able to define their own lists of locally denied/accepted users.

In the PL-Grid project a completely new system called BAT[5], used for collecting accounting information, has been developed. The system consists of one central service (called BAT broker), which gathers resource usage records produced by clients (called BAT agents) deployed in every organizational unit. There exist two classes of BAT agents: local and grid. The former rely on information available in the batch system's (Torque and PBS Professional in PL-Grid) log files such as the job's wall-clock time, local id, etc., while the latter augment such information with high-level data – e.g. the user's certificate, distinguished name or grid job id. In PL-Grid a new BAT agent has been developed for QCG-Computing, alongside gLite and UNICORE agents. This agent periodically reads job data from the QCG-Computing accounting database and sends it over a secure channel to the BAT broker.

The QCG-Computing service also acts as a lightweight information service, providing (via a Web Service interface) information about PL-Grid users, groups and grants in a particular organizational unit. The consumer of this information is the QCG-Broker service, which later exploits it for scheduling purposes.

In addition, as every production infrastructure has to be monitored constantly, a set of Nagios[6] probes was provided for the QCG-Computing, QCG-Notification and QCG-Broker services.

The final requirement of the Operations Center was to provide binary RPM (Red Hat Package Manager) packages compatible with the Scientific Linux operating system.

4 Quantum Chemistry Application Using QCG

4.1 Motivation

This section presents the main assumptions and challenges for computationally demanding simulations for quantum chemistry, in particular an actual scientific application called NEL, representing numerical algorithms related to large-scale quantum-chemical calculations. The legacy quantum chemistry application discussed in [4], originally implemented in Fortran, has been redesigned and optimized to take full advantage of QCG. The optimization was performed in cooperation with the author of the original version, as part of the user support activity in the PL-Grid project.

The electronic structure of atoms and molecules is determined on the basis of the Schrödinger equation. However, in the case of many-electron systems this equation is not solvable and approximation schemes have to be employed. Moreover, the solving is often performed numerically with the use of advanced computations. Orbital methods, which posit that each electron moves in the average field of the other electrons, are quite ubiquitous but, in spite of their being generally successful, they are not applicable to a wide variety of quantum-chemical problems in which high precision of physical outcome is expected. Numerous

[5] `https://gforge.cyfronet.pl/projects/bat-plgrid/`
[6] `http://www.nagios.org/`

variational methods of solving the Schrödinger equation with high precision have been developed – for instance basing on the explicitly correlated Gaussian (ECG) functions, exploiting the growing computational power of state-of-the-art supercomputers and clusters. This is mostly associated with the ECG method's potential that might be used in the case of atoms with more than three electrons and many-electron, multicenter molecules [2,5].

There have also been variational attempts to solve the Schrödinger equation

$$H\,\Psi = E\,\Psi\,, \tag{1}$$

in which Ψ constitutes the wave function reflecting a particular state of a molecule or an atom and H is the clamped nuclei Hamiltonian, describing all the Coulomb interactions between electrons and nuclei as well as the electrons' kinetic energies. E stands for the electronic energy of the system. Both the energy and wave function are sought when trying to solve this equation for a given Hamiltonian. The following equation reflects the so-called trial wave function. This function will be represented as a K-term linear combination of N-electron ECG basis functions ϕ_i

$$\Psi = \sum_{i=1}^{K} c_i\,\phi_i\,, \tag{2}$$

in which

$$\phi_i = \mathcal{A}_N \left\{ \mathcal{P} \left[\exp\left(-\sum_{p=1}^{N-1} \sum_{q=p+1}^{N} A_{ipq}\,(\boldsymbol{r}_p - \boldsymbol{s}_{ip})\,(\boldsymbol{r}_q - \boldsymbol{s}_{iq}) \right) \right] \Theta \right\}. \tag{3}$$

Within this formula, spatial electronic coordinates are marked as $\boldsymbol{r}_k$, antisymmetrizer (working on space and spin coordinates) as $\mathcal{A}_N$, the spatial symmetry projector as $\mathcal{P}$, and Θ is the N-electron spin function.

Identical particles are indistinguishable from the point of view of their physical properties and the antisymmetry projector $\mathcal{A}_N$ is responsible for taking this feature into account. When electron coordinates are considered and the calculations performed on them, it returns a sum of $N!$ terms differing by electron coordinate permutations. All the elements of the Hamiltonian matrix are affected by the $N!$ explosion, which is one of the factors limiting the size of examined systems to a few electrons. The total number of nonlinear parameters, collected in $\boldsymbol{A}_i$ and $\boldsymbol{s}_i$ which are variables of the optimization process, depends on the size of atoms or molecules (assessed by the number of electrons N and nuclei) and the expansion (2). In the most advanced/complex cases, there are over 100 000 non-linear parameters which have impact on the wave function (and energy). In such cases determining the energy minimum becomes a very computationally demanding task. A solution of the Schrödinger equation presented in the matrix form of the general symmetric eigenvalue problem (GSEP) offers the optimal vector of the linear parameters c_i and the corresponding approximation ϵ for the exact electronic energy

$$\mathcal{H}\,\boldsymbol{c} = \epsilon\,\mathcal{S}\,\boldsymbol{c}\,. \tag{4}$$

The Hamiltonian $\mathcal{H}$ and the overlap $\mathcal{S}$ matrices are composed of elements defined using $4N$-dimensional integrals over all coordinates $dV_1 \ldots dV_N$ of the electrons [3,18,5]

$$\mathcal{H}_{ij} = \int \phi_i H \phi_j \, dV_1 \ldots dV_N \,, \tag{5}$$

$$\mathcal{S}_{ij} = \int \phi_i \phi_j \, dV_1 \ldots dV_N \,. \tag{6}$$

Although the proposed method is general and can theoretically be applied to any N-electron atomic or molecular system, these calculations are commonly employed for systems containing no more than four electrons in order to preserve the accuracy level [5,6,17]. An extension of the applicability of the ECG method to larger systems is the challenge we face. QCG opens up yet another possibility of reaching this goal.

The ECG wave functions within the above mentioned conditions are accurate; however, any additional electron appearing in the system would require reassessment of the algorithms. When the number of electrons in a system is increased to five, accurate calculations call for $\sim 10^3 - 10^4$ ϕ_i functions. Moreover, the energy has to be minimized by the variational parameters within a multidimensional space. Arriving at a near-exact estimate of energy E would imply evaluations of the energy ϵ conducted $10^6 - 10^7$ times. On the grounds of the aforementioned theory, computations within a hybrid parallel computing model which involve the Message Passing Interface (MPI) are carried out so as to facilitate communication of processes over the network, along with a shared memory model (OpenMP) on local nodes.

4.2 Application Requirements

The scalar part of the program is responsible for the matrix elements of Hamiltonian $\mathcal{H}$, Eq. (5), and the overlap matrix $\mathcal{S}$, Eq. (6), whereas the vector part is composed of GSEP solutions, Eq. (4), and they are both deemed the most demanding parts with respect to computations. Cholesky decomposition of the matrix $\mathcal{H} - \varepsilon_t\,\mathcal{S}$ with a trial value ε_t (close to the desired energy ϵ) lays the foundations for the aforementioned demanding parts of the algorithm. As the next step on the way to optimizing the energy ε_t, an inverse iteration procedure is implemented [16]. In this computation a triangular system of equations is solved in each iteration and, consequently, the energy converges to ϵ following a few iterations. Optimization of the very large number of non-linear variational parameters $\boldsymbol{A}_i$ and $\boldsymbol{s}_i$ is yet another vital part of the algorithm. It is recommended to apply the iterative optimization procedure from $i = 1$ to $i = K$ tuning (in the i-th step) the parameters of a single-basis function ϕ_i. Therefore, during the optimization process and within the Powell's conjugate directions method [15],

variational parameters reflecting the objective energy function undergo changes and if these changes do not improve the energy, the algorithm is stopped.

The optimization shot is another important concept which should be mentioned here. It involves a single execution of the above mentioned parts that is scalar- and vector-based, returning a single value of the energy ϵ. The number of shots required to achieve convergence approaches 100 times the number of nonlinear parameters. For instance, in the case of a four-electron wave function with 2400 terms and 14 nonlinear parameters per term (LiH molecule), these tasks have to be performed more than 10^6 times:

- scalar part – evaluation of the matrix elements $\mathcal{H}_{ij}$, $\mathcal{S}_{ij}$, with thousands of floating-point operations per each element,
- vector part – solution of the GSEP for matrices with $K = 2400$ rows and columns.

The whole application works as follows. First, the multithreaded master MPI process reads all input parameters. Subsequently, it forwards tasks to other MPI processes. Once data is received, an MPI process calculates $\mathcal{H}$ and $\mathcal{S}$ elements. Different matrix parts are computed by different processes. Having completed all the tasks, the initial energy is calculated by the master process with the use of the GSEP algorithm. The GotoBLAS2 library is employed in this part of the application.

The nonlinear optimization process for subsequent rows of the $\mathcal{H}$ and $\mathcal{S}$ can commence once the initial energy is known. Dozens of optimization shots are carried out, on average, for a single basis function. These shots are computations of the elements of a single row in the GSEP algorithm. All worker processes are responsible for calculating row elements while the master process manages the GSEP algorithm with the use of parallel threads.

A recent version of the presented application was successfully and efficiently executed in a multi-cluster environment managed by QCG services using geographically distributed resources in Poznań and Kraków. In both cases resources used in this experiment belonged to the production PL-Grid infrastructure and the main motivation to use co-allocated resources was to shorten the time which the task (requiring hundreds of computational cores) spends in the queue until it obtains the requested resources. We have observed that, especially in the case of the application implementing a master/slave paradigm with no intensive communication between processes, the performance overhead stemming from network delays incurred by the geographical distribution of participating computations is more than compensated for by the shorter queuing time. The requested number of cores (256) was co-allocated by the QCG-Broker on Reef and Zeus clusters respectively in Poznań and Kraków and parts of the application were started in a synchronous way by local QCG-Computing services. The main computing and communications steps in the hybrid parallel NEL application, with its decomposition between two clusters, are shown in Fig. 3.

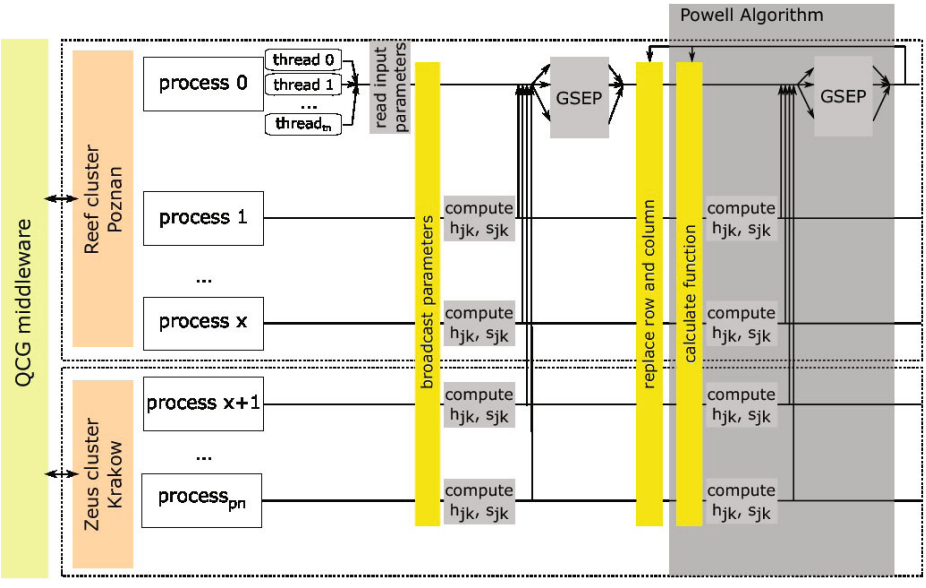

Fig. 3. Main computing and communications steps of the NEL application with its decomposition between two clusters

5 Deployment Status

The QCG middleware was successfully deployed in several production HPC environments belonging to the PL-Grid infrastructure (e.g. Poznań Supercomputing and Networking Center – PSNC, Academic Computer Centre Cyfronet AGH, TASK). Moreover, there are ongoing deployment activities at the Leibniz-Rechenzentrum – LRZ, University College London – UCL, National Institute for Research in Computer Science and Control – INRIA, and the Dortmund University.

6 Future Work

National and international grid e-Infrastructures provide resources comparable to the largest existing large-scale parallel computing environments. However, current grids typically do not address sophisticated scenarios which require specific Quality of Service guarantees to support simultaneous management of many kinds of resources, storage and networks. Existing grid middleware solutions, including the popular gLite [14] and UNICORE [19] systems, do not satisfy all demands of modern scientific simulations and computing models. One of the shortcomings of these systems is poor support for advance reservation, which makes it difficult (or indeed impossible) to run jobs on co-allocated resources. Basing on the outcome of earlier European projects (e.g. GridLab [25], BREIN

[24], QosCosGrid [28]) we propose QCG as an alternative grid middleware platform. QCG services support the latest open standards, including OGF HPC Basic Profile, JSDL, OGSA BES and WS-Notification, thereby providing a flexible, interoperable interface upon which to run, execute and monitor complex jobs as well as create advance reservations and co-allocations. To the best of our knowledge, QCG currently provides the most efficient and powerful multi-user access to job management and co-scheduling features, compared to other existing grid middleware services.

In order to meet the emerging end-user requirements, QCG will be integrated with various new services and application tools for distributed multiscale computing in the scope of the MAPPER project [26]. QCG middleware will also be evaluated by partners involved in other National Grid Infrastructures and PRACE partners [27].

References

1. Agullo, E., Coti, C., Herault, T., Langou, J., Peyronnet, S., Rezmerita, A., Cappello, F., Dongarra, J.: QCG-OMPI: MPI Applications on Grids. Future Gener. Comput. Syst. 27, 357–369 (2011)
2. Bachorz, R., Komasa, J.: Variational calculations on H2+ using exponentially correlated Gaussian wave functions. Computational Methods in Science and Technology 11(1), 5–9 (2005)
3. Boys, S.F.: The Integral Formulae for the Variational Solution of the Molecular Many-Electron Wave Equations in Terms of Gaussian Functions with Direct Electronic Correlation. Royal Society of London Proceedings Series A 258, 402–411 (1960)
4. Cencek, W., Komasa, J., Rychlewski, J.: High-performance Computing in Molecular Sciences. In: Handbook on Parallel and Distributed Processing, p. 205. Springer, Heidelberg (2000)
5. Cencek, W., Rychlewski, J.: Many-electron Explicitly Correlated Gaussian Functions. I. General Theory and Test Results 98(2), 1252–1261 (1993)
6. Cencek, W., Szalewicz, K.: Ultra-high Accuracy Calculations for Hydrogen Molecule and Helium Dimer. International Journal of Quantum Chemistry 108, 2191–2198 (2008)
7. Ciepiela, E., Nowakowski, P., Kocot, J., Harężlak, D., Gubała, T., Meizner, J., Kasztelnik, M., Bartyński, T., Malawski, M., Bubak, M.: Managing Entire Lifecycles of e-Science Applications in GridSpace2 Virtual Laboratory – from Motivation through Idea to Operable Web-Accessible Environment Built on Top of PL-Grid e-Infrastructure. In: Bubak, M., Szepieniec, T., Wiatr, K. (eds.) PL-Grid 2011. LNCS, vol. 7136, pp. 228–239. Springer, Heidelberg (2012)
8. Dziubecki, P., Grabowski, P., Krysiński, M., Kuczyński, T., Kurowski, K., Piontek, T., Szejnfeld, D.: Online Web-Based Science Gateway for Nanotechnology Research. In: Bubak, M., Szepieniec, T., Wiatr, K. (eds.) PL-Grid 2011. LNCS, vol. 7136, pp. 205–216. Springer, Heidelberg (2012)
9. Kravtsov, V., Bar, P., Carmeli, D., Schuster, A., Swain, M.: A scheduling framework for large-scale, parallel, and topology-aware applications. Journal of Parallel and Distributed Computing 70(9), 983–992 (2010)
10. Kurowski, K., Ludwiczak, B., Nabrzyski, J., Oleksiak, A., Pukacki, J.: Dynamic Grid Scheduling with Job Migration and Rescheduling in the GridLab Resource Management System. Sci. Program. 12, 263–273 (2004)

11. Kurowski, K., de Back, W., Dubitzky, W., Gulyás, L., Kampis, G., Mamonski, M., Szemes, G., Swain, M.: Complex System Simulations with QosCosGrid. In: Allen, G., Nabrzyski, J., Seidel, E., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2009, Part I. LNCS, vol. 5544, pp. 387–396. Springer, Heidelberg (2009)
12. Kurowski, K., Nabrzyski, J., Oleksiak, A., Weglarz, J.: Grid Scheduling Simulations with GSSIM. In: Proceedings of the 13th International Conference on Parallel and Distributed Systems, vol. 02, pp. 1–8. IEEE Computer Society, Washington, DC, USA (2007)
13. Kurowski, K., Oleksiak, A., Weglarz, J.: Multicriteria, Multi-user Scheduling in Grids with Advance Reservation. J. of Scheduling 13, 493–508 (2010)
14. Laure, E., Grandi, C., Fisher, S., Frohner, A., Kunszt, P., Krenek, A., Mulmo, O., Pacini, F., Prelz, F., White, J., Barroso, M., Buncic, P., Byrom, R., Cornwall, L., Craig, M., Di Meglio, A., Djaoui, A., Giacomini, F., Hahkala, J., Hemmer, F., Hicks, S., Edlund, A., Maraschini, A., Middleton, R., Sgaravatto, M., Steenbakkers, M., Walk, J., Wilson, A.: Programming the Grid with gLite. In: Computational Methods in Science and Technology (2006)
15. Powell, M.J.D.: An Efficient Method for Finding the Minimum of a Function of Several Variables Without Calculating Derivatives. The Computer Journal 7(2), 155–162 (1964)
16. Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T.: Numerical Recipes in Fortran 77: The Art of Scientific Computing, 2nd edn. Cambridge University Press (September 1992)
17. Przybytek, M., Cencek, W., Komasa, J., Łach, G., Jeziorski, B., Szalewicz, K.: Relativistic and Quantum Electrodynamics Effects in the Helium Pair Potential. Phys. Rev. Lett. 104(18), 183003 (2010)
18. Singer, K.: The Use of Gaussian (Exponential Quadratic) Wave Functions in Molecular Problems. I. General Formulae for the Evaluation of Integrals. Royal Society of London Proceedings Series A 258, 412–420 (1960)
19. Streit, A., Erwin, D., Mallmann, D., Menday, R., Rambadt, M., Riedel, M., Romberg, M., Schuller, B., Wieder, P.: UNICORE – From Project Results to Production Grids. In: Grid Computing and New Frontiers of High Performance Processing (2005)
20. Troger, P., Rajic, H., Haas, A., Domagalski, P.: Standardization of an API for Distributed Resource Management Systems. In: Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid, CCGRID 2007, pp. 619–626. IEEE Computer Society, Washington, DC, USA (2007)
21. Tuecke, S., Welch, V., Engert, D., Pearlman, L., Thompson, M.: Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile. RFC 3820 (Proposed Standard) (June 2004)
22. HPC Basic Profile Version 1.0, `http://www.ogf.org/documents/GFD.114.pdf`
23. OASIS Web Services Notification, `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsn`
24. BREIN Project, `http://www.eu-brein.com/`
25. GridLab Project, `http://www.gridlab.org`
26. MAPPER – Multiscale Applications on European e-Infrastructures, `http://www.mapper-project.eu/`
27. PRACE – The Partnership for Advanced Computing in Europe, `http://www.prace-project.eu/`
28. QosCosGrid Project, `http://biomed.science.ulster.ac.uk/archive/www.qoscosgrid.eu/`