Using OGSA-DQP to Support Scientific Applications for the Grid

M. Nedim Alpdemir[1], Arijit Mukherjee[2], Anastasios Gounaris[1], Norman W. Paton[1], Alvaro A.A. Fernandes[1], Rizos Sakellariou[1], Paul Watson[2], and Peter Li[2]

[1] Department of Computer Science,
University of Manchester,
Oxford Road, Manchester M13 9PL, UK
[2] School of Computing Science,
University of Newcastle upon Tyne,
Newcastle upon Tyne NE1 7RU, UK

Abstract. The data management problems in grid computing are often challenging in many aspects such as data volumes, heterogeneity, structural complexity and semantic content. Thus, e-Scientists and scientific application developers stand to benefit from tools and environments that either hide, or help to manage, the inherent complexity involved in accessing and making concerted use of the diverse resources. This paper describes OGSA-DQP, a high level data integration tool for service-based grids, and illustrates how it can be used to support grid users, via an example scientific study in bioinformatics. The paper also discusses various options for employing OGSA-DQP to handle data integration tasks as service orchestrations involving both data and analysis services.

1 Introduction

Both commercial and scientific applications increasingly require access to distributed resources. Grid technologies have been introduced to facilitate efficient sharing of resources in a heterogeneous distributed environment. However, from its inception, grid computing has provided mechanisms for data access that lie at a much lower level than those provided by commercial database technology [7]. This is despite the fact that the data management problems in grid computing are not likely to be less complex, rather the contrary, insofar as in all relevant aspects (viz., data volumes, structural complexity and semantic content) data in the grid is likely to be at least as complex as that found in current commercial environments. Furthermore, in those applications for which grid solutions seem particularly appropriate (e.g., scientific ones), data is often more fragmented and more in need of computationally-demanding analyses than in classical Web applications (e.g., e-commerce ones). Thus, high-level data access and integration services are needed if applications that have large amounts of data with complex structure and complex semantics are to benefit from the grid. This paper briefly describes OGSA-DQP [1], a high level data integration tool for

P. Herrero, M.S. Pérez, and V. Robles (Eds.): SAG 2004, LNCS 3458, pp. 13–24, 2005.

service-based grids, and aims to illustrate how it can be used to support grid users in accessing distributed resources in a bioinformatics context. The paper also discusses how OGSA-DQP can be exploited to provide a relatively low-cost implementation of complex scientific applications for the grid.

The rest of the paper is structured as follows: Section 2 briefly introduces the service-oriented approaches to resource utilisation on the grid; Section 3 describes the architecture and usage of OGSA-DQP as a high-level data access and integration tool for service-based grids; Section 4 illustrates how OGSA-DQP can be exploited to support e-scientists in conducting their studies, through an example bioinformatics application; Section 5 briefly discusses various options for employing OGSA-DQP in more complex grid applications and finally Section 6 presents a number of conclusions.

2 Service-Oriented Architectures for Resource Utilisation

Service-based approaches [4] (such as Web Services and the Open Grid Services Architecture) have gained considerable attention recently for supporting distributed application development in e-business and e-science. The service-based approach seems to many a good solution to the problem of modelling a virtual organisation as a distributed system, and is perceived to offer a convenient paradigm for resource sharing through resource virtualisation. Web Services, in particular in conjunction with the resource access and management facilities of grid computing, show considerable promise as an infrastructure over which distributed applications in e-business and e-science can be developed. As such, it is argued that uniformly treating the diversity of resources and applications as services significantly simplifies their use and management [5].

One particular impact of service-oriented approaches on application development, is the introduction of new techniques that permit various models for aggregating distributed software modules as loosely-coupled compositions of coarse-grained services to construct more complex applications [6]. Workflow languages such as Business Process Execution Language (BPEL) appear to be central to service aggregation approaches. However, It is worth noting that although it is likely that workflow languages will have a prominent role, service-based Distributed Query Processing (DQP) also offers service orchestration capabilities, accomplishing system-supported optimisation of declarative requests with implicit parallelism, a combination that should yield significant programmer productivity and performance benefits for large-scale, data intensive applications. OGSA-DQP is one approach to provide such capabilities.

3 OGSA-DQP: A Grid Service Framework for Data Integration and Analysis

3.1 Overview

OGSA-DQP [1] is essentially a high-throughput distributed data-flow engine that relies on a service-oriented abstraction of grid resources and assumes that data

sources are accessible through service-based interfaces. OGSA-DQP relies on infrastructure support from other grid Middleware at two distinct levels: it uses the reference implementation of Open Grid Services Architecture (OGSA) [3] viz., Globus Toolkit 3 (GT3) [8], which implements a service-based architecture over virtualised resources referred to as Grid Services (GSs), thus enabling dynamic allocation of resources necessary for efficient evaluation of a distributed query; it also builds upon OGSA-DAI [2] which implements Grid Data Services (GDSs) that insulate users from certain aspects of data source heterogeneity, ensuring that metadata and data held in a particular data source are accessed via a standard, well-defined and uniform interface. By building on those layers, OGSA-DQP delivers a framework that

- supports declarative queries over many *Grid Database Services* (GDSs) by creating a union of the database schemas of the participating data sources.
- supports calls to external web services through insertion of the web service operation invocations into a query, thereby combining data access with data analysis;
- adapts techniques from parallel databases to provide implicit parallelism for complex data-intensive requests; and
- automates complex, onerous, expert configuration and resource utilisation decisions on behalf of users via its query optimisation module.

OGSA-DQP provides two services to fulfil its functions: The *Grid Distributed Query Service (GDQS)* and the *Grid Query Evaluation Service (GQES)*. The GDQS provides the primary interaction interfaces for the user, collects the necessary metadata, and acts as a coordinator between the underlying query compiler/optimiser engine and the GQES instances. The GQES, on the other hand, is used to evaluate (i.e. execute) a query sub-plan assigned to it by the GDQS. The number of GQES instances and their location on the grid is specified by the GDQS, based on the decisions made by a query optimiser and represented as an execution schedule for query partitions (i.e. sub-plans). GQES instances are created and scheduled dynamically, to evaluate the partitions of a query constructed by the optimiser of the GDQS.

Figure 1 illustrates the high-level architecture of OGSA-DQP, where the client application queries multiple data resources via a global schema that presents a union of the schemas of the participating data sources. Notice that OGSA-DQP utilizes the computational resources and data resources available to it, via services provided by a core grid middleware (i.e. OGSA and OGSA-DAI). As such, by virtue of this core middleware support the query execution engine is constructed dynamically (i.e. at run time) by instantiating GQESs for each section (or partition) of the distributed query plan, as stipulated by the query optimizer encapsulated within the GDQS.

3.2 Using OGSA-DQP for Querying Distributed Data Sources

This section describes, briefly, how OGSA-DQP can be used in practice as a high-level tool for retrieving and combining data from multiple data sources, as well as feeding retrieved data into analysis services if desired.

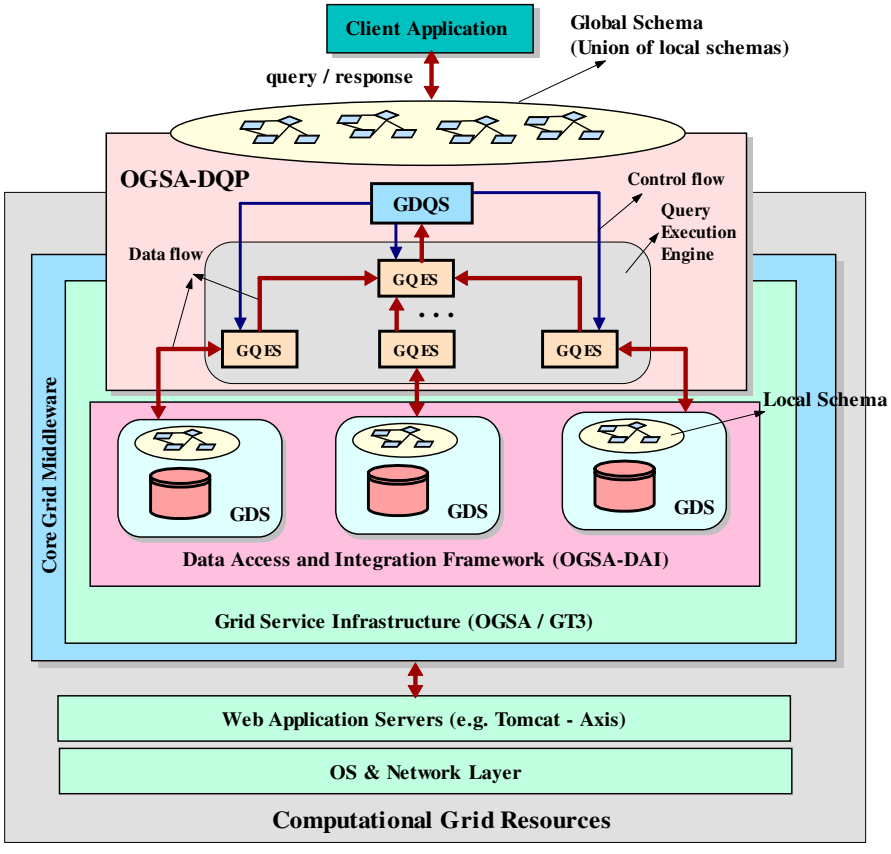

Fig. 1. A High-level Architecture of OGSA-DQP

Starting a Query Session. Preparing OGSA-DQP for query submission involves identifying the data sources and the analysis services to be used in a query session. This process may start with a search and discovery phase at the end of which the user finds the set of resources s/he is interested in. OGSA-DQP does not currently offer direct support for this initial discovery process, largely because this is conceived to be an application level functionality. Instead, it is assumed that the user has already identified the required resources. Thus, the user submits an XML document containing the list of the data sources and analysis services as illustrated by the following XML fragment:

```
<GDQDataSourceList>
    <importedDataSource>
        <GDSFactoryHandle>
            http://host:port/ogsa/services/ogsadai/ProteinDBGDSFactory
        </GDSFactoryHandle>
    </importedDataSource>
    <importedDataSource>
        <GDSFactoryHandle>
            http://host:port/ogsa/services/ogsadai/GenesDBGDSFactory
```

```
        </GDSFactoryHandle>
    </importedDataSource>
    <importedDataSource>
        <GDSFactoryHandle>
            http://host:port/ogsa/services/ogsadai/MicroarrayDBGDSFactory
        </GDSFactoryHandle>
    </importedDataSource>
    <importedService
        name="EntropyAnalyser"
        wsdlURL="http://host:port/services/EntropyAnalyserService?wsdl"/>
</GDQDataSourceList>
```

Note that the data sources are indicated by the Grid Data Service Factory (GDSF) handles of the services that wrap those data sources. The analysis services are indicated by URLs that point to the WSDL documents describing those services.

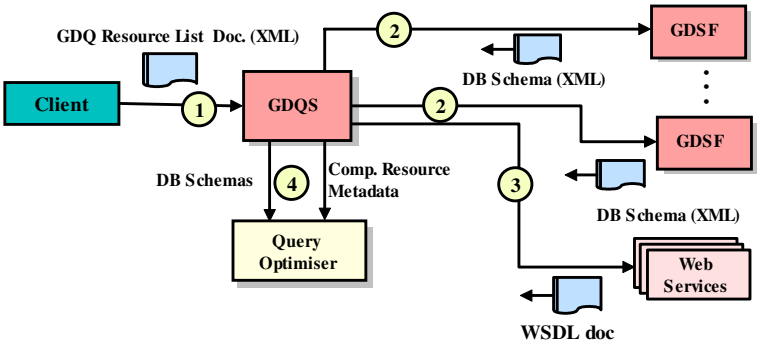

Fig. 2. Metadata Retrieval During OGSA-DQP Set-up Phase

As illustrated in Figure 2, on receipt of the XML document containing the resource list (interaction 1), the GDQS obtains metadata about each resource in the list (interactions 2 and 3) to aid the query optimiser in generating an efficient execution plan. This metadata includes database schemas (both the logical structure of the data and some physical characteristics such as index information, cardinality, row sizes, etc.) that are obtained from the data sources, and WSDL documents that are obtained from the web services.

Submitting Query Requests. After the GDQS is set-up with a resource set and a query session is initiated, the user can submit multiple query requests until the GDQS instance is destroyed, which effectively terminates the session.

As illustrated in Figure 3 (a), for each query request a GDQS instance compiles, optimises, partitions and schedules the query to generate a distributed query plan optimised for specific requirements of the submitted query. Each partition in the distributed query plan is assigned to one or more execution nodes. The GDQS, then, commands the creation of GQESs as stipulated by the partitioning and scheduling decided on by the compiler (Figure 3 (b), interaction 1), and co-ordinates the GQESs into executing the plan. Each execution node

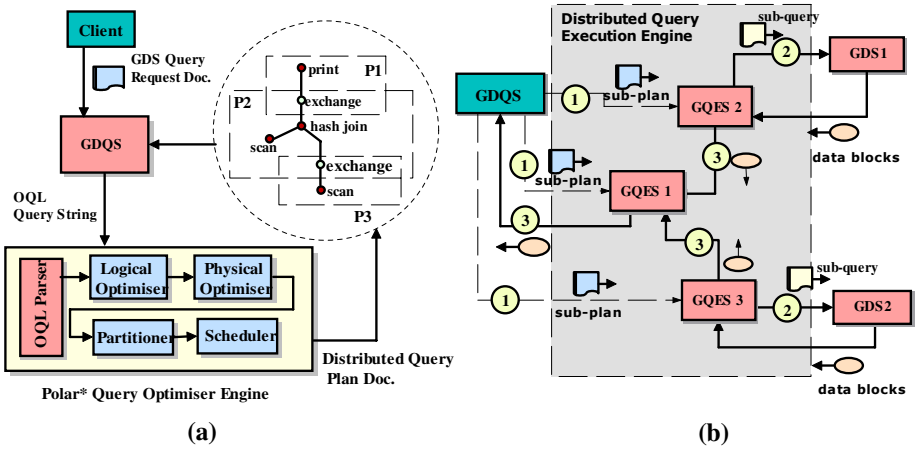

Fig. 3. Query Optimisation and Execution Process

corresponds to a GQES instance, each of which initiates its evaluation upon receiving its plan partition. The whole process effectively constructs a tree-like data flow system with the GDQS instance at the root, GDS instances at the leaf and a collection of GQESs in the middle (Figure 3 (b), interactions 2-3).

4 How Grid Users Can Benefit from OGSA-DQP

This section illustrates OGSA-DQP in use as part of the solution to a bioinformatics problem. First a brief explanation of the problem is provided in Section 4.1, followed by a description of how OGSA-DQP is put into use to aid in providing a solution, in Section 4.2.

4.1 An Example Bioinformatics Application: The Graves' Disease Scenario

The myGrid project (www.mygrid.org.uk) has developed an application that uses a number of middleware services to build in-silico tools for a study that seeks to identify genes and SNPs associated with a genetic autoimmune condition known as *Graves disease (GD)*. The condition is an disease of the thyroid in which the immune system of an individual attacks the cells of the thyroid gland resulting in hyperthyroidism (thyroid overactivity).

Researchers studying human genetic disease such as this, ultimately wish to establish which genes are affected in the diseased state, the changes in those genes between individuals and the underlying molecular mechanisms that lead to the autoimmune response. The hypothesis is that single nucleotide polymorphisms (SNPs) are instrumental in the disease mechanism. Thus, there are three objectives:

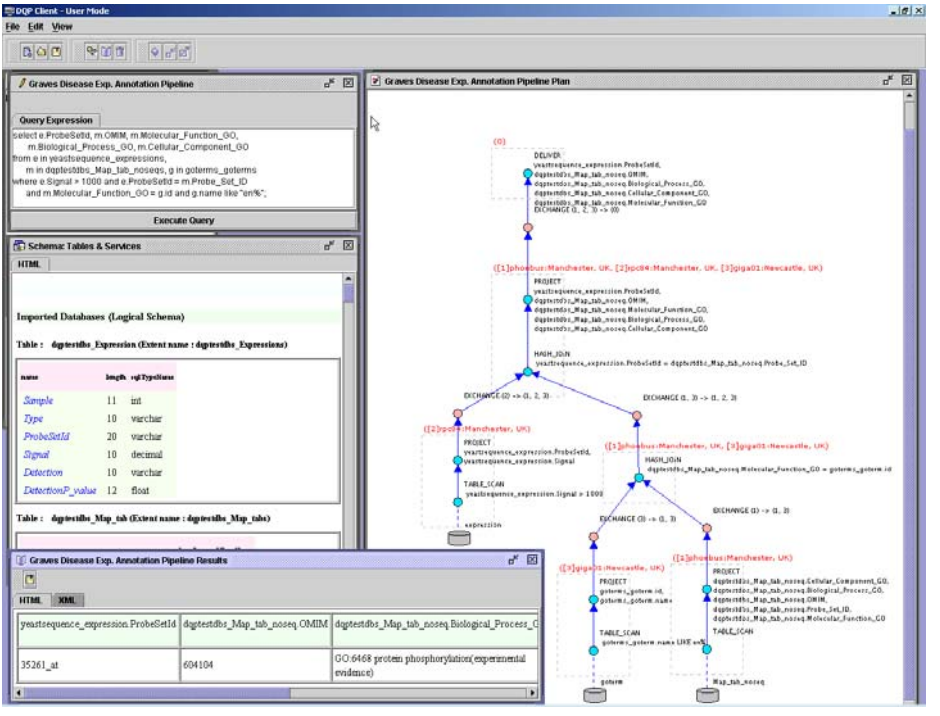

Fig. 4. OGSA-DQP GUI Client Screen Shot showing the Query Plan and the Results

1. Investigate what genes and loci are involved in GD.
2. Examine which single nucleotide polymorphisms (SNPs) located in genes are involved in GD.
3. Develop genotyping experiments to test the above hypotheses.

One of the several in-silico experiments designed for the whole study is an annotation pipeline that aims to help the user establish which genes in the candidate gene pool may be involved in the diseased state [9]. In other words, the main purpose is to retrieve information associated with candidate genes that were differentially expressed in GD. The user can assimilate the information provided and make a decision as to which gene or genes they wish to examine in more detail, and ultimately take back to laboratory studies. To achieve that, however, it is necessary to return links to annotation data from a range of genomic databases and the literature, for each gene in the dataset. A *distributed query* over grid enabled databases can achieve the required data integration at a relatively low cost compared to other approaches that require a separate, isolated interaction with each of the databases and do the integration as a custom post-processing step (e.g. application-specific scripting solutions, or workflow-based solutions that either include application-specific logic in each of the data-access services or a separate service in the workflow to perform the integration).

4.2 The Distributed Queries for the Annotation Pipeline

The key functionality required in the annotation pipeline is the ability to map from the Affymetrix probe set identifiers referencing a candidate gene to sequence or database identifiers in biological databases. For the nucleotide sequence annotation pipeline, the mappings from an Affymetrix probe set ID to EMBL accession number, and OMIM, GO and Medline identifiers are required. Some of those mappings can be generated by using existing annotation tools and stored in a custom database. In the example application presented in this paper, the mapping between probe set IDs and OMIM, GO and Medline IDs are stored in a single database. Thus, in total three distributed databases need to be accessed to retrieve and join the required information. Those are:

1. The AffyMapper database (named as *Map_tabs* in the query below) which is a custom database created by obtaining mappings using Affymetrix's NetAffx gene annotation tool.
2. An arbitrary microarray database (named as *expressions* in the query below) containing gene expression data from the Affymetrix microarray analyses.
3. The Gene Ontology (GO) database (named as *goterms* in the query below).

The following is an example query that integrates data from the three databases listed above:

```
QUERY1:
select
    e.ProbeSetId, m.OMIM, m.Molecular_Function_GO,
    m.Biological_Process_GO, m.Cellular_Component_GO
from
    e in expressions, m in Map_tabs, g in goterms
where
    e.Signal>1000 and e.ProbeSetId = m.Probe_Set_ID and
    m.Molecular_Function_GO = g.id and g.name like <input  name>
```

The query selects those Affymetrix probe set IDs which have an expression signal over 1000 and which have also been annotated with a given GO ID. In other words, the query answers questions such as *"which genes are expressed in my samples and have a molecular function activity x?"*.

Figure 4 illustrates the OGSA-DQP GUI client that was used to execute the example query. The GUI client supports an administrator mode where a system administrator can create and save configurations with a resource set and pre-defined queries, and a user mode where a more novice user can load a configuration, execute pre-defined queries or add more queries if necessary. The screen shot illustrates the user mode and includes three small windows on the left representing – from top to bottom in order– the query text, the tabular representation of the union of the database schemas of the participating data sources, and the query results in tabular form. The figure also shows a graphical representation of the query execution plan on the right hand side of the window. The plan is dynamically generated for each executed query and is annotated to indicate the partitioning and scheduling of the distributed plan (denoted by dashed rectangular boundaries), algorithms employed to evaluate each sub-plan (denoted by circular nodes with textual annotations such as HASH_JOIN, TABLE_SCAN,

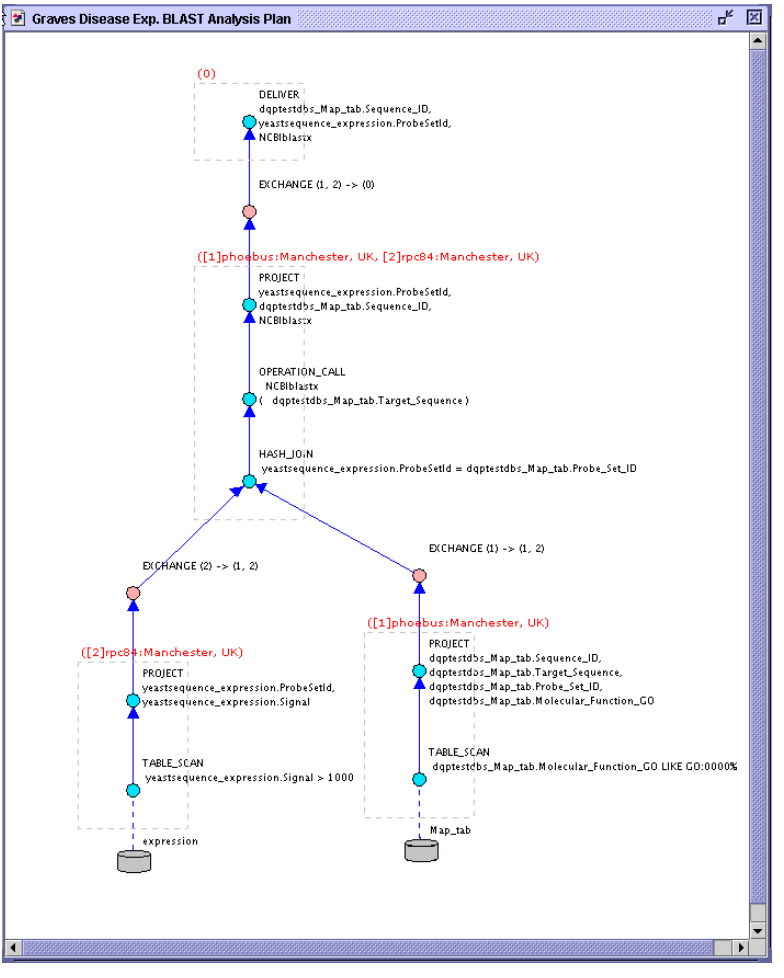

Fig. 5. The Query Plan with the BLAST Call

etc.) and the location of the servers that were used to execute a particular subplan (denoted by textual labels above the dashed rectangular boundaries).

OGSA-DQP can also be used to implement another step in the GD study which involves applying the Basic Local Alignment Search Tool (BLAST) analysis service. The BLAST call here can be used for identifying PDB records that might provide information on the structure of the protein that is encoded by the nucleotide sequence of the candidate gene. An example query is given below:

```
QUERY2:
select
    e.ProbeSetId, m.Sequence_ID, NCBIblast(s.sequence)
from
    e in expressions, m in Map_tabs, s in sequences
where
    e.Signal > 1000 and e.ProbeSetId = m.Probe_Set_ID and
    m.Molecular_Function_GO like <input GO:Id>
```

This query includes a call to a Web Service that wraps the BLAST analysis program hosted by the European Bioinformatics Institute (EBI) (see http://www .ebi.ac.uk/blastall/index.html) and it demonstrates a powerful feature of OGSA-DQP; that of combining invocations to analysis programs with data retrieval in a single query statement. Notice that the gene sequences retrieved from the *sequences* database (which may potentially constitute a set), are fed into the NCBIblast function call. Figure 5 shows the distributed query plan generated for QUERY2. Note that the sub-plan that contains an invocation to BLAST web service (i.e. OPERATION_CALL operator) is parallelised across two nodes (i.e. servers) on the grid (indicated by the textual label above the rectangular box in the middle with two machine names separated by a comma), since BLAST is a relatively high-cost operation.

Note that during execution of the queries, the databases are accessed via grid services (i.e. GDSs) and the intermediate data processing computations are also carried out by grid services (i.e. GQESs) all of which are linked with dynamically forged data-flow and control-flow paths, which effectively constructs a service orchestration framework. Note also that with the second query, the service orchestration is extended beyond data services and internal query evaluator services (i.e. GQESs) to external (i.e. third party) analysis services, making OGSA-DQP a declarative (i.e. query-driven) alternative to procedural (e.g. workflow-based) service-orchestration systems.

5 Alternative Methods for Using OGSA-DQP in Grid Applications

Although Section 4 described how OGSA-DQP can be used via a stand-alone GUI client, this is not the only way OGSA-DQP delivers its functionality. As OGSA-DQP is itself a Grid Service, exposing a programming interface in conformance to interaction patterns specified by the OGSA-DAI project, it can be integrated into higher-level applications in at least two other ways:

1. An application can discover the GDQS from public service registries, and interact with it via its service interface to submit the list of resources required for the distributed queries, and subsequently to pass the query requests themselves. The results received as a response to the query requests can then be transmitted to other processing modules in the application or presented to the user via application-specific user interfaces.
2. The GDQS can be invoked in an intermediate step in a more complex workflow involving calls to other services. This is a particularly interesting case as it leads to simplified workflows due to the replacement of many interlinked activities (i.e. a sub-workflow) with a call to OGSA-DQP as a single task, and could potentially result in performance gains, since OGSA-DQP optimizes and parallelises its query plans.

6 Conclusions and Future Work

This paper has described a distributed query processing service, namely OGSA-DQP, for service-based grids, and demonstrated how it can support the development of scientific grid applications via an example application from the bioinformatics domain. In summary, developers and users of scientific applications for the grid can stand to benefit from OGSA-DQP from several angles:

1. The users of the grid can benefit from OGSA-DQP as a generic data integration and analysis tool. A typical use-case involves deploying OGSA-DQP to a virtual organisation application server and configuring it with a set of frequently used data and analysis resources; a procedure most likely to be carried out by a system administrator. The configured OGSA-DQP can then be used to pose queries against the resource set. Section 4.2 described how the user mode of the OGSA-DQP GUI client allows one to query a set of resources.
2. The developers of scientific applications for the grid can delegate data integration tasks to OGSA-DQP, to implement a distinct functional part of a sophisticated application. It is worth noting that since a GDQS is fully compliant with data delivery and transformation patterns specified by OGSA-DAI, the application can command the GDQS to channel the results to another Grid Data Service, to send the results to a remote file system via FTP, to compress and save the results to the local file system, or to deliver the results asynchronously in blocks via streaming. See the OGSA-DQP user guide at www.ogsadai.org.uk/dqp for details.
3. As programming practice evolves from traditional coding models to service composition or service coreography, the developers of scientific application workflows can employ the GDQS to undertake the orchestration of a sub-set of services required for the overall solution. Our future work plans include integrating OGSA-DQP into various workflow execution environments, and carrying out more quantitative research on comparing performance characteristics of the two.

Acknowledgements. The work reported in this paper has been supported by the UK e-Science Programme.

References

1. M. Alpdemir, A. Mukherjee, N. W. Paton, P. Watson, A. A. Fernandes, A. Gounaris, and J. Smith. Service-based distributed querying on the grid. In M. E. Orlowska, S. W. M. P. Papazoglou, and J. Yang, editors, *the Proceedings of the First International Conference on Service Oriented Computing*, pages 467–482. Springer–Verlag, 15–18 December 2003.
2. A. Anjomshoaa et al. The design and implementation of grid database services in OGSA-DAI. In S. J. Cox, editor, *Proceedings of UK e-Science All Hands Meeting Nottingham*. EPSRC, 2–4 September 2003.

3. I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Technical report, OGSI-WG, Global Grid Forum, 2002. Draft 2.9, June 22, 2002.
4. K. Gottschalk, S. Graham, H. Kreger, and J. Snell. Introduction to Web Services Architecture. *IBM Sys. Journal*, 41(2):170–177, 2002.
5. S. Graupner, V. Kotov, A. Andrzejak, and H. Trinks. Service-centric globally distributed computing. *IEEE Internet Computing*, 7(4):36 – 43, July/August 2003.
6. R. Khalaf and F. Leymann. On web services aggregation. In B. Benatallah and M. C. Shan, editors, *Proceedings of VLDB Technologies for e-Services Workshop*, LNCS 2819, pages 1 – 13. Springer–Verlag, 2003.
7. R. W. Moore, C. Baru, R. Marciano, A. Rajasekar, and M. Wan. Data-Intensive Computing. In I. Foster and C. Kesselman, editors, *The Grid: Blueprint for a New Computing Infrastrcuture*, chapter 5, pages 105–129. Morgan Kaufmann, 1999.
8. T. Sandholm and J. Gawor. Globus Toolkit 3 Core A Grid Service Container Framework. Technical report, 2003. www-unix.globus.org/toolkit/3.0/ogsa/docs/.
9. R. Stevens et al. Performing in silico experiments on the grid: a users perspective. In S. J. Cox, editor, *Proceedings of UK e-Science All Hands Meeting Nottingham*, pages 43 – 50. EPSRC, 2–4 September 2003.