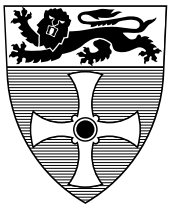
UNIVERSITY OF
NEWCASTLE



University of Newcastle upon Tyne

COMPUTING SCIENCE

Virtual Machines in DynaSOAr: Creating an on-demand ad-hoc Virtual Grid

A. Mukherjee and P. Watson.

TECHNICAL REPORT SERIES

No. CS-TR-1002 February, 2007

TECHNICAL REPORT SERIES

No. CS-TR-1002 February, 2007

Virtual Machines in DynaSOAr: Creating an on-demand ad-hoc Virtual Grid

Arijit Mukherjee, Paul Watson.

Abstract

DynaSOAr is an infrastructure for dynamically deploying web services over a Grid or a set of networked resources. The DynaSOAr view of grid computing focussed entirely on the concept of services, rather than the more traditional jobs. Services are deployed on demand to meet the changing performance requirements. DynaSOAr includes the support to deploy services in pre-built Virtual Machines on demand thereby creating an ad-hoc Virtual Grid. This paper describes the DynaSOAr architecture with respect to the on-demand deployment of Virtual Machines and shows that in case of services involving a large amount of data transfer, there are advantages in creating an ad-hoc grid of services close to the data through the dynamic deployment of Virtual Machines.

Printed and published by the University of Newcastle upon Tyne,
Computing Science, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, England.

Bibliographical details

MUKHERJEE, A., WATSON, P..

Virtual Machines in DynaSOAr: Creating an on-demand ad-hoc Virtual Grid
[By] A. Mukherjee, P. Watson.

Newcastle upon Tyne: University of Newcastle upon Tyne: Computing Science, 2007.

(University of Newcastle upon Tyne, Computing Science, Technical Report Series, No. CS-TR-1002)

Added entries

UNIVERSITY OF NEWCASTLE UPON TYNE
Computing Science. Technical Report Series. CS-TR-1002

Abstract

DynaSOAr is an infrastructure for dynamically deploying web services over a Grid or a set of networked resources. The DynaSOAr view of grid computing focussed entirely on the concept of services, rather than the more traditional jobs. Services are deployed on demand to meet the changing performance requirements. DynaSOAr includes the support to deploy services in pre-built Virtual Machines on demand thereby creating an ad-hoc Virtual Grid. This paper describes the DynaSOAr architecture with respect to the on-demand deployment of Virtual Machines and shows that in case of services involving a large amount of data transfer, there are advantages in creating an ad-hoc grid of services close to the data through the dynamic deployment of Virtual Machines.

About the author

Arijit joined the Computing Science department as a Research Associate in the MyGrid project in May 2002. He did his Masters in Computer Science and Engineering from Jadavpur University, India in 1997, and since then worked in the software industry. He joined Tata Consultancy Services (the largest software consultancy in South East Asia) in 1997, and worked in the Networks Group in various projects for the telecommunication companies like Bell Atlantic, Nokia, Siemens. His major projects were related to Residential Broadband (Bell Atlantic), Intelligent Network (Nokia), Corporate GSM (Siemens/Opuswave). In 2000, he joined Verizon Communications Inc. (formerly Bell Atlantic) in the US as a software engineer and worked there on Verizon's Data Network Management System till May 2002.Currently, Arijit works in the MyGrid project as well as the Distributed Query Processing of OGSA-DAI. He is also doing his PhD under Prof. Paul Watson.

Paul Watson is Professor of Computer Science and Director of the North East Regional e-Science Centre. He graduated in 1983 with a BSc (I) in Computer Engineering from Manchester University, followed by a PhD in 1986. In the 80s, as a Lecturer at Manchester University, he was a designer of the Alvey Flagship and Esprit EDS systems. From 1990-5 he worked for ICL as a system designer of the Goldrush MegaServer parallel database server, which was released as a product in 1994. In August 1995 he moved to Newcastle University, where he has been an investigator on research projects worth over £13M. His research interests are in scalable information management. This includes parallel database servers, data-intensive e-science and grid computing. In total, he has over thirty refereed publications, and three patents. Professor Watson is a Chartered Engineer, a Fellow of the British Computer Society, and a member of the UK Computing Research Committee.

Suggested keywords

DYNAMIC DEPLOYMENT,
WEB SERVICE,
GRID,
DISTRIBUTED QUERY PROCESSING,
CIRTUAL MACHINES

Virtual Machines in DynaSOAr: Creating an on-demand ad-hoc Virtual Grid

Arijit Mukherjee and Paul Watson

School of Computing Science, Newcastle University,
Claremont Tower, Claremont Road, Newcastle Upon Tyne, United Kingdom
{Arijit.Mukherjee,Paul.Watson}@ncl.ac.uk
http://www.cs.ncl.ac.uk

Abstract. *DynaSOAr is an infrastructure for dynamically deploying web services over a Grid or a set of networked resources. The DynaSOAr view of grid computing focussed entirely on the concept of services, rather than the more traditional jobs. Services are deployed on demand to meet the changing performance requirements. DynaSOAr includes the support to deploy services in pre-built Virtual Machines on demand thereby creating an ad-hoc Virtual Grid. This paper describes the DynaSOAr architecture with respect to the on-demand deployment of Virtual Machines and shows that in case of services involving a large amount of data transfer, there are advantages in creating an ad-hoc grid of services close to the data through the dynamic deployment of Virtual Machines.*

Key words: *Dynamic deployment, Web Service, Grid, distributed query processing, Virtual Machines.*

1 Introduction

Most traditional Grid computing infrastructures, such as Globus[1], Condor[2] have as their core, a distributed job scheduling framework capable of dynamically routing jobs - a combination of executable code and data - to remote compute resources for execution. The consumer creates and submits the job to a job-scheduling system where it is routed for execution on the most suitable resource available at the time.

In recent times, there has been a shift towards building Grid-infrastructures and distributed applications using standardized web services technologies. An application is built as a set of interacting services with defined interfaces, described using the standard description languages WSDL[3], and communicating between themselves by exchanging SOAP[4] messages. However, the concept of a job is still important - when the underlying computing environment available to a service cannot meet the performance requirements, the conventional approach is to

create a job and submit to a job scheduling system for execution. The consequence being that the developers have to deal with two different computational entities - services and jobs. Similarly, the Grid infrastructure has to support both entities.

We have been investigating an alternative approach, proposed in [7] - a dynamic service deployment framework, known as DynaSOAr. DynaSOAr offers a service-only approach to building Grid applications and so is free of the job abstraction. To achieve this, DynaSOAr maintains a repository of deployable services, and when no existing service deployments can meet the computational requirements, a service is deployed dynamically on an available host in order to exploit the computational resources offered by it. In [8] we have described how DynaSOAr can be integrated with distributed Web service frameworks, such as OGSA-DQP[5] in order to benefit from the on-demand deployment of services on local clusters rather than using remote hosts for frequent, long running requests involving large amounts of data transfer. In this paper, we investigate the use of virtualization technologies such as VMWare[10], to provide a basis for an ad-hoc virtual grid through the on-demand deployment of virtual machines to meet the changing demands of services. This has the advantage that a complex software environment - service code, libraries, operating system, local data - can be encapsulated and deployed in a Virtual machine.

The rest of the paper is structured as follows: Section 2 provides a brief introduction to DynaSOAr architecture. Section 3 describes the usage of Virtualization techniques in DynaSOAr, followed by the experimental setup, results and analysis in Section 4. Related works are discussed in Section 5, current and future directions in Section 6, with conclusions in Section 7.

2 Basic DynaSOAr Architecture

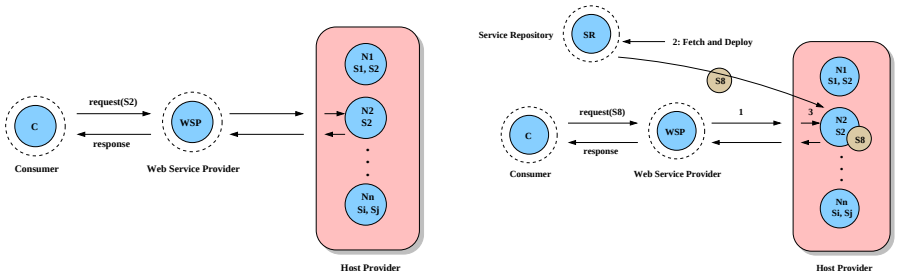
DynaSOAr provides a generic infrastructure for deploying web services as and when required, on available nodes[7]. This dynamic deployment is achieved by processing the incoming consumer request in two stages using two different components, namely a *DynaSOAr Web Service Provider* and a *Host Provider*, with a defined interface between them.

- The *DynaSOAr Web Service Provider* exposes an endpoint to which clients can send requests in a normal way though a SOAP message. It selects an appropriate *Host Provider* and forwards the message to it along with any associated Quality of Service (QoS) parameters and a new element in the message header identifying a *software repository* from where the service code can be accessed for deployment if necessary.
- The *Host Provider* is responsible for controlling the computational resources, such as a cluster or a grid, on which services can be deployed, and requests

processed. It accepts the SOAP messages forwarded by the *Service Provider* on behalf of the hosted services, and sends back any response generated after processing the request.

Depending on whether or not the requested service is already deployed on the node, there can be two different interaction patterns at the *Host Provider* -

1. If the service is already deployed on the computational node where the request is to be processed, then the *Host Provider* routes the SOAP message to the service on that node, and sends the response back, as shown in Figure 1(a).
2. When the consumer makes a request for a service, which is not already deployed on any of the available nodes, such as the request for service S8 sent by the consumer in Figure 1(b), a decision is made about the target node on which the service is to be deployed and the message is forwarded to that node. The node downloads the service code from the *software repository*, deploys the service dynamically, and processes the request. This will also occur if the service is already deployed, but existing deployments are unable to meet the performance requirements. In this case another instance of the service will be deployed and incoming requests shared between it and the existing deployments.



(a) Routing request to existing deployment

(b) Request service not already deployed

Fig. 1. Routing requests in DynaSOAr

It either scenarios, the consumer is not aware of the resources behind the *Web Service Provider* or the fact that the service has been dynamically deployed. They interact with the *Web Service Provider* by sending SOAP messages in the standard way.

Two additional components, namely a *Registry Service*, and a *Service Repository*, support dynamic deployment. The *DynaSOAr Registry Service* is provided by

GRIMOIRES[9], which is a UDDI-based registry, with added support for storing metadata as RDF triplets. The *Service Repository* manages the *upload* or *download* of the service code. The Host Providers communicate with this service while downloading the service code in order to deploy a service and are capable of deploying Web Services, Virtual Machines and SQL Stored Procedures with a uniform approach.

3 Using Virtualization

In recent times, virtualization technologies have become a popular choice in the Grid world. Virtualization as defined by VMWare is: "an abstraction layer that decouples the physical hardware from the operating system to deliver greater IT resource utilization and flexibility[10]." Using virtualization techniques, it is possible to run several virtual machines (VM) simultaneously, with different operating systems, on the same physical host by isolating each one from the physical environment. The advantages provided by VMs regarding partitioning, isolation and encapsulation make them useful in the Grid infrastructure[15]. DynaSOAr utilizes this concept and not only provides an extensive *service-oriented* framework which allows on-demand deployment of services, stored procedures over SQL databases and VMs, but also performs a scheduling process to determine the best possible candidate node to deploy the service, VM or stored procedure based on the heuristical algorithm devised in the associated GridSHED[12] project.

As opposed to [15], the DynaSOAr method of deployment is transparent to the consumer, and is uniform for all the supported components. Instances of VMs are pre-built, and services are deployed on them on an Apache Tomcat (or any other) Web service container. Special environments, such as third-party libraries, databases, required by the service are included in the VM. Similar to a normal Web Service, the VM is uploaded to the DynaSOAr Service Repository, and for each Web Service offered by it, an entry is created in the DynaSOAr registry. When a request is received from a consumer for a service that is either not yet deployed, or for which the existing deployments are overloaded or unavailable, the VM image is downloaded to an appropriate host and started. The request is then forwarded to the hosted service. Starting up a VM also ensures that all the other hosted services in it are available, and any requests for these services can be forwarded to the service without the need for redeployment. It should be noted however, that downloading and starting up a virtual machine is costly, and DynaSOAr ensures that this is done only when necessary to reduce the overhead.

Each available node in DynaSOAr hosts a HostProvider service, exposing the underlying resource. This service is responsible for downloading and deploying the service or VM image from the repository when a dynamic deployment is called for. Otherwise, it forwards the request to the already deployed service and

sends the response back to the consumer via the DynaSOAr Service Provider. When deploying a VM, the service invokes a VirtualMachineInstaller component, which uses the VMWare Server commands to start up the virtual machine, fetch the IP address of the newly started VM, and the consumer request is then forwarded to the corresponding service on the VM.

The current prototype is built with the VMWare Server. The service containers and databases are configured to start as system processes during the VM boot, thus nullifying the requirement of any user input. VMWare Server also provides an extensive set of command line APIs and Perl scripts which can be used to communicate with the server to control to VM. We have used both "host-only" and "bridged" form of networking in the virtual machines. The "host-only" configuration where only the host nodes are able to communicate with the guest VMs, is particularly suitable for organizations who do not want to expose the details of their computational resources. In some cases, however, the "bridged" configuration is necessary, in which case we have used IP addresses from a pre-reserved pool.

4 Experiment

As our experimental application, we selected OGSA-DQP[5] which is a publicly available service-oriented distributed query processing system providing distributed query functionality over databases spread across the Grid. OGSA-DQP is composed of two major services - (i) Grid Distributed Query Service (GDQS) and (ii) Query Evaluation Service (QES). The GDQS is implemented as an extension to the standard OGSA-DAI[6] service, and is deployed as an OGSA-DAI data service with an exposed data service resource. The DQP data service resource thus exposed, supports declarative querying over a set of OGSA-DAI data services, each wrapping a database on some computational node. It also supports the invocation of analysis services over the query results. The databases and analysis services used for distributed queries can be spread over a large geographical area. Based on the schema and WSDL imported from the data and the analysis services and the resources available to it, a query compiler/optimizer component, Polar*[19], generates sub-plans which are partitions of a parallel query plan. These sub-plans are distributed to the participating evaluation services each of which is responsible for evaluating the sub-plan assigned to it and conveying the result back either to the root partition or other evaluation services. Finally, the result is collected at the node evaluating the root partition and sent to the GDQS and hence to the consumer. Figure 2 shows the interactions between the OGSA-DQP components for a typical query.

Our argument in selecting OGSA-DQP as our experimental application is to investigate if using DynaSOAr to dynamically co-locate the data access service and the analysis service with the data, can increase performance, especially when

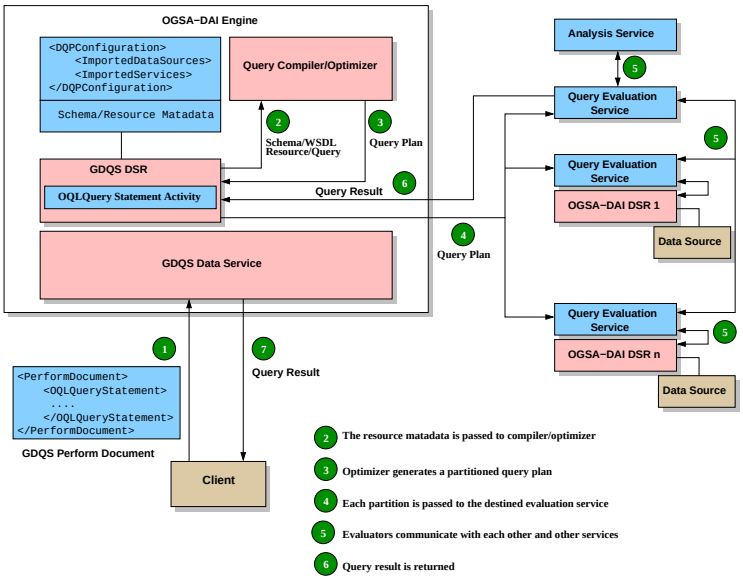


Fig. 2. Interactions in the OGSA-DQP system

frequent, long-running queries are executed. The usage scenarios which could benefit from this use of the DynaSOAr include:

- A database can be deployed in a virtual machine to enable OGSA-DQP to serve queries involving the new database.
- In case of a fault in any of the data access, evaluation and analysis resources used by DQP, the virtual machine deployment of another copy of the failed resource can keep the system working using the techniques of [13].

4.1 Setup

The experiments to analyze the impact of dynamically deployed virtual machines on distributed query processing activities primarily concentrated on the deployment of a pre-built virtual machine with services required for OGSA-DQP. A dynamic DQP framework was setup on a set of Linux machines within the Newcastle University GIGA cluster - each of them being a four-processor Intel® Xeon^{TM} CPU 2.80GHz system, with 2GB memory. The GDQS was deployed on another Linux machine - a four-processor Intel® Xeon^{TM} CPU 3.06GHz with 1GB memory. A virtual machine with a Fedora Core 4 guest O/S and 256MB memory was built such that it mirrored one of the DQP resources. A copy of the analysis service was deployed on a Linux (one-processor Intel® Xeon^{TM}

CPU 2.40GHz system with 1GB memory) system at the Edinburgh Parallel Computing Centre (EPCC). To compare the results with a real-world situation, an exactly similar DQP framework with databases, data access, evaluation and analysis services was set up on Planetlab[11], with compute resources located at geographically remote locations.

One of the databases used for the test queries was loaded with several tables, each with 100,000 records, and fixed record sizes of 128 bytes, 256 bytes, 512 bytes, 1 Kbytes, 2 Kbytes, 4 Kbytes, 8 Kbytes and 10 Kbytes. The experiments were designed to fetch data out of each table with varying cardinalities and perform the analysis on each tuple using the analysis service. Results were collected in order to compare the performance of the the various setups, such as (i) a planetlab setup with remote analysis service, (ii) a planetlab setup where the evaluation and analysis services are co-located on a planetlab node, (iii) the local setup with only a remote analysis service, (iv) the complete local setup and (iv) the local setup with one dynamically deployed virtual machine.

4.2 Results And Analysis

In preliminary experiments, the DynaSOAr framework was used to deploy the pre-built virtual machine. The deployment cost includes the time required to transfer the virtual machine image from the repository to the target host and the time taken for the the virtual machine to boot. Figure 3 shows the time taken to deploy and start up the virtual machine.

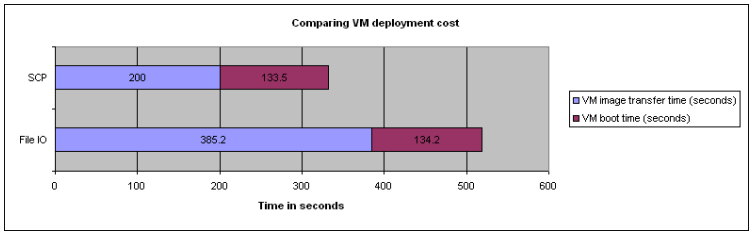


Fig. 3. Virtual Machine deployment cost

The average time required for a virtual machine deployment on a computational node was approximately 520 seconds using File I/O and 334 seconds using SCP. This is a one-time cost and is incurred only during the DQP initialization phase. A major part of the deployment cost of the VM is in transporting the VM image, which is around 4GB in size, over the network. Once the virtual machine is deployed locally, the performance of the queries executed by this GDQS while using this VM is almost equivalent to that running natively on the same machine,

as is evident from the other experiments. Further, the deployment of the virtual machine makes available all the services that are part of this VM are available and can be used if required.

A set of eight queries were executed on a test database, retrieving results with different cardinalities from the database. Each query was used to retrieve datasets of different sizes. Each query also invoked the analysis service for each retrieved tuple. An example query used in the tests is as follows:

```
%print select p.id, calculateEntropy(p.sequence) from p in
proteinsequence_random_sequence_128s where p.id < 3999;
```

This query from a bio-informatics application, retrieves 4,000 tuples from the database and invokes the analysis service (*calculateEntropy* in this case) on the sequence attribute of each tuple. The results of these experiments are shown in Figure 4(a) and Figure 4(b).

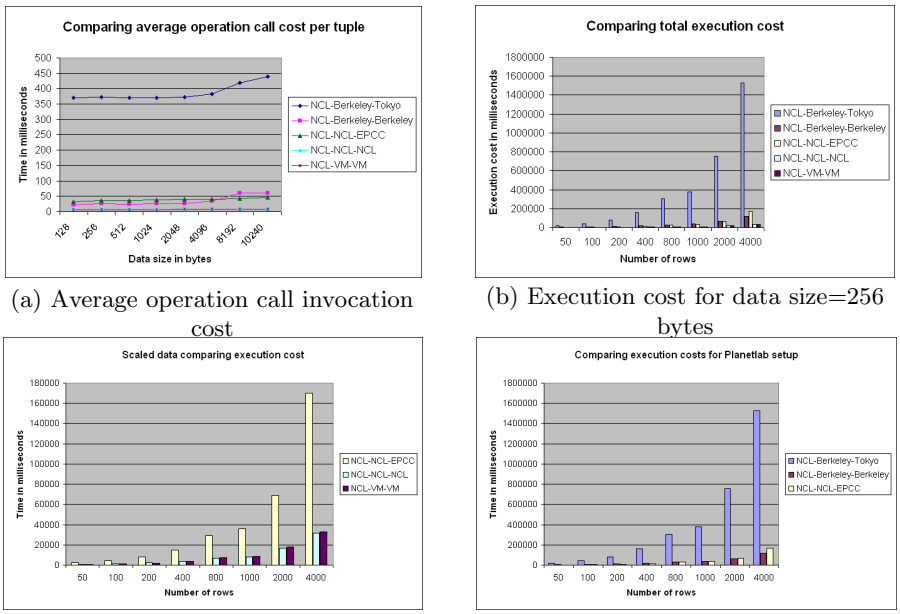
Figure 4(a) compares the average analysis service invocation cost per tuple (in milliseconds) in all the setups for different sized tuples, from 128 bytes to 10 Kbytes. Figure 4(b) compares the total query execution cost (in milliseconds) in all the setups for a typical data size of 256 bytes. For other data sizes, the results show a similar trend. Figure 4(c) and (d) are scaled versions of figure 4(b)[1].

These results clearly show that for the queries using analysis services over the data, the performance when the analysis service is co-located with the data access and evaluation services is much better than when the analysis service is remote. The difference in the performance is quite noticeable considering the fact that a very high-speed Internet backbone exists between the server at Edinburgh Parallel Computing Centre and the Linux cluster at Newcastle University. The performance differences are even more prominent when the analysis service resides much further apart geographically, because a higher communication cost is incurred in this case. Further, the performance of the system with a dynamically deployed VM is almost similar to the performance of a local setup, which shows the benefit of transparent dynamic service deployment so that they are local to each other.

5 Related Work

In the DynaSOAr architecture, although the Host Provider is exposed as a high level service, it can still exploit the outcome of other projects producing similar

[1] The charts use a key such as X-Y-Z, where X is the location of the GDQS, Y is the location of the data sources and evaluation services and Z is the location of the analysis service



(a) Average operation call invocation cost

(b) Execution cost for data size=256 bytes

(c) Scaled graph comparing the PlanetLab setup with EPCC setup

(d) Scaled graph comparing the local and VM setup with EPCC data

Fig. 4. Performance Analysis of DynaSOAr-enabled OGSA-DQP

components. Dynamic deployment of services raises issues such as deciding on whether to deploy a service on a new node or to use an existing, but possibly overloaded, deployment. The GridSHED[12] project for job scheduling has been investigating this area, and the results are being utilized to design a better scheduling model for DynaSOAr.

There is some existing work on creation of ad-hoc virtual grid or workspaces using virtual machines. In [15] the authors speak about creating Virtual Workspace over a cluster of virtual machines. In Virtuoso[16] and VIOLIN[17], work has been done with regards to internetworking issues arising from a VM setting. VMPlants[18] propose a way of automated configuration and creation of flexible VMs. These differ from our work in that we propose an approach of creating a uniform service-oriented layer which abstracts the underlying resources and allows the creation of an on-demand ad-hoc virtual grid transparent to the consumer by enabling on-demand deployment of different types of components such as web services, virtual machines, stored procedures, .NET services together in one framework.

6 Current And Future Directions

Work is under way to enable the DynaSOAr framework use the findings from the GridSHED project to create a better scheduler which would make decisions about new deployment of services/virtual machines based on node performance data from the available nodes. These findings can be utilized in the context of the query compiler in OGSA-DQP which performs some static scheduling based on a simple cost model, but is ignorant about the inherent dynamism and changing node performance in a Grid system. The effects of this has led to related work on *adaptive distributed query processing*[14].

Some work has been done in this area of *fault-tolerant distributed query processing*[13]. The concepts of the dynamic DQP system are also relevant to fault-tolerant query processing systems where a failure of an evaluation or a data node can be handled through the deployment of the same service on another node or a virtual machine as a replacement of the failed node, and by replaying certain sections of the query evaluation to regain the state where the processing stopped due to the failure.

Virtual Machines are one of the most important directions for the DynaSOAr framework, and further investigation is being undertaken regarding the deployment of databases in virtual machines addressing issues such as keeping dynamically deployed copies in sync in the presence of updates.

7 Conclusion

In this paper, we present work on enabling an on-demand ad-hoc virtual grid environment using the DynaSOAr framework. We selected distributed query processing as our experimental application as we believe that DQP can benefit from the dynamic deployment mechanisms of DynaSOAr by deploying evaluation and analysis services closer to the data. And virtualization technologies allow a complex software environment required for the services to be encapsulated in a VM pre-built for dynamic deployment. This claim is supported by the experimental results which show that the cost of deploying a VM is outweighed by the increased performance especially when the queries involve a large amount of data transfer over the network.

Acknowledgments. We wish to extend our gratitude to our collaborators in the DynaSOAr, CRISP, GridSHED, OGSA-DAI, OGSA-DQP, CoreGrid and DAIT projects at Newcastle University, Manchester University and EPCC. We particularly wish to thank Jim Smith for his helpful discussions. We would also like to thank the UK Engineering and Physical Sciences Research Council, and the DTI for funding the GridSHED and DAIT projects.

References

1. Globus Toolkit, http://www.globus.org/toolkit
2. Tannenbaum, T., Wright, D., Miller, K., and Livny, M.:Condor - A Distributed Job Scheduler. In:Beowulf Cluster Computing with Linux, T. Sterling, Ed.: The MIT Press, 2002.
3. W3C, Web Service Description Language, http://www.w3.org/2002/ws/desc
4. Simple Object Access Protocol (SOAP), http://www.w3.org/TR/2000/NOTE-SOAP-20000508
5. OGSA-DQP, http://www.ogsadai.org.uk/about/ogsa-dqp/
6. OGSA-DAI, http://www.ogsadai.org.uk/
7. Watson, P., Fowler, C., Kubicek, C., Mukherjee, A. et. al.: Dynamically Deploying Web Services on a Grid using Dynasoar. In: Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing, ISORC 2006, IEEE Computer Society 2006
8. Mukherjee, A., Watson, P.: Adding Dynamism To OGSA-DQP: Incorporating The DynaSOAr Framework In Distributed Query Processing. In: CoreGrid Workshop on Grid Middleware, EuroPar 2006
9. GRIMOIRES, http://twiki.grimoires.org/bin/view/Grimoires/
10. VMWare, http://www.vmware.com/
11. PlanetLab, http://www.planet-lab.org
12. Palmer, J., Mitrani, I.: Optimal Server Allocation in Reconfigurable Clusters with Multiple Job Types. In: Computational Science and its Applications (ICCSA 2004), Assisi, Italy, 2004.
13. Smith, J., Watson, P.: Fault-Tolerance in Distributed Query Processing. In: 9th International Database Engineering And Application Symposium. IDEAS 2005, http://ideas.concordia.ca/ideas2005/, IEEE, 329–338
14. Gounaris, A., Paton, N.W., Sakellariou, R., Fernandes, A.A.A. et. al.: Adapting to Changing Resource Performance in Grid Query Processing. In: VLDB Workshop on Data Management in Grids, DMG 2005, http://liris.cnrs.fr/~jpierson/DMG_VLDB05/
15. Keahey, K., I. Foster, T. Freeman, X. Zhang, D. Galron.: Virtual Workspaces in the Grid. In: Europar 2005, Lisbon, Portugal, September, 2005
16. Lin, Bin and Dinda, Peter A.: VSched: Mixing Batch And Interactive Virtual Machines Using Periodic Real-time Scheduling. In: Proceedings of the 2005 ACM/IEEE conference on Supercomputing, Washington, DC, USA, 2005
17. Jiang, X., and Xu, D.:Violin: Virtual internetworking on overlay infrastructure. In:Tech. Rep. CSD TR 03-027, Department of Computer Sciences, Purdue University, July 2003.
18. Krsul, I., and Ganguly, A., and Zhang, J., and Fortes, Jose A. B., and Figueiredo, Renato J.: VMPlants: Providing and Managing Virtual Machine Execution Environments for Grid Computing. In: Proceedings of the 2004 ACM/IEEE conference on Supercomputing, Washington, DC, USA, 2004
19. Smith, J., Gounaris, A., Watson, P. et. al.: Distributed Query Processing on the Grid. In: Grid Computing 2002. Lecture Notes in Computer Science, Vol. 2536. Springer-Verlag, 279–290