**FutureGrid Platform FGPlatform: Rationale and Possible Directions**

*Geoffrey Fox June 8 2010*

# Integrating FutureGrid and Commercial Clouds

The computing landscape is rapidly changing and this offers remarkable opportunities for scientific discovery. Important changes include multicore, cloud computing, smartphone and tablet interfaces as well as the data deluge driving new requirements for Cyberinfrastructure and new opportunities for scientific discovery. FutureGrid is an NSF TeraGrid facility providing a distributed testbed for developing research applications and middleware as well as supporting new approaches to education. FutureGrid supports many leading edge Grid and Cloud Technologies including MPI, MapReduce, Nimbus, gLite and Globus. Partners are responsible for making particular technologies are available and users request hardware and software resources for their experimental work. Some experiments only need the 6 major resources currently on FutureGrid (4 IBM iDataPlex clusters, 1 Dell Cluster and a Cray XT5). Within FutureGrid, the user has full control over resources allowing reproducible experiments with hardware configured on demand between different operating systems and with either "bare metal" or virtual machine as base. This enables a suite of experiments that compare the performance of different approaches in a robust fashion. Further most of FutureGrid lies on a private network that can be isolated allowing security sensitive experiments while a programmable network fault generator gives even greater richness to FutureGrid use. However some FutureGrid experiments can involve outside resources such as Condor flocks, GPU clusters or commercial cloud resources and the unique features of external resources make use of them essential in some cases. Some of these external resources may be dedicated to FutureGrid but unlike core resources treated as though external, as the operating environment is not under the control of FutureGrid. Alternatively an experiment "centered" outside FutureGrid (say in the Azure Cloud) may wish to access FutureGrid to run a component -- high performance cluster or GPU enhanced cluster -- that is not practical to run on Azure. Our experiments are managed by a framework built around the Pegasus system from USC (acting as a manager and not a workflow engine) and the INCA monitoring environment from SDSC. Note that although we offer Eucalyptus and Nimbus on FutureGrid with similar core functionality to an Amazon cloud, one might still wish to experiment on Amazon with its rich set of functionalities such as queuing, notification and multiple storage offerings.

FutureGrid can support experiments involving Azure and Amazon (and in fact other external systems but that's not focus here) in several ways given in Table 1. Note we largely ignore the Google Application Engine as currently is targeted at Web applications while Azure and Amazon offer a general cloud environment.

 Table 1: Support of Commercial Clouds in FutureGrid

|  |
| --- |
| 1. We support experiments that link Commercial Clouds and FutureGrid experiments with one or more workflow environments and portal technology installed to link components across these platforms
 |
| 1. We support environments on FutureGrid that are similar to Commercial Clouds and natural for performance and functionality comparisons. These can both be used to prepare for using Commercial Clouds and as the most likely starting point for porting to them (item c below). One example would be support of MapReduce-like environments on FutureGrid including Hadoop on Linux and Dryad on Windows HPCS which are already part of FutureGrid portfolio of supported software. Of course offering an advanced platform on FutureGrid could be good just because the environment is more attractive than conventional scientific computing environments.
 |
| 1. We develop expertise and support porting to Commercial Clouds from other Windows or Linux environments
 |
| 1. We support comparisons between and integration of multiple commercial Cloud environments -- especially Amazon and Azure in the immediate future
 |
| 1. We develop tutorials and expertise to help users move to Commercial Clouds from other environments.
 |

# Commercial Cloud Capabilities

Commercial Clouds offer cost effective utility computing with the elasticity to scale up and down in power. However as well as this key distinguishing feature, they are adding a growing number of additional capabilities commonly termed "Platform as a Service". For Azure, current Platform features include Azure Table, Queues, Blob, Database SQL, Web and Worker roles. Amazon is often viewed as "just" Infrastructure as a Service but it continues to add Platform features including SimpleDB (similar to Azure Table), Queues, Notification, Monitoring, Content Delivery Network, Relational Database, MapReduce (Hadoop) . Google does not currently offer a broad-based cloud service but the Google Application Engine offers a powerful Web application development environment. We define a FutureGrid high performance platform FGPlatform given in Table 2 that includes those capabilities of Cloud platforms that appear particularly interesting for large scale scientific computing plus those needed to run applications that link commercial clouds to outside resources -- in particular FutureGrid itself. FGPlatform allows us to support table 1 for Azure and Amazon.

Table 2: Features of FGPlatform supporting Integration of FutureGrid and Commercial Clouds

|  |
| --- |
| **Authentication and Authorization:** Provide single sign in to both FutureGrid and Commercial Clouds linked by workflow |
| **Workflow:** Support workflows that link job components between FutureGrid and Commercial Clouds. Trident from Microsoft Research is initial candidate |
| **Data Transport:** Transport data between job components on FutureGrid and Commercial Clouds respecting custom storage patterns |
| **Program Library:** Store Images and other Program material (basic FutureGrid facility) |
| **Blob:** Basic storage concept similar to Azure Blob or Amazon S3 |
| **DPFS Data Parallel File System:** Support of file systems like Google (MapReduce), HDFS (Hadoop) or Cosmos (dryad) with compute-data affinity optimized for data processing |
| **Table:** Support of Table Data structures modeled on Apache Hbase or Amazon SimpleDB/Azure Table |
| **SQL:** Relational Database |
| **Queues:** Publish Subscribe based queuing system |
| **Worker Role:** This concept is implicitly used in both Amazon and TeraGrid but was first introduced as a high level construct by Azure |
| **MapReduce:** Support MapReduce Programming model including Hadoop on Linux, Dryad on Windows HPCS and Twister on Windows and Linux |
| **Software as a Service:** This concept is shared between Clouds and Grids and can be supported without special attention |
| **Web Role:** This is used in Azure to describe important link to user and can be supported in FutureGrid with a Portal framework |

Note there are some features like notification and monitoring that could be straightforwardly supported but did not seem as important as those in Table 2.

# Components of the FutureGrid Platform

## Authentication and Authorization

We will provide a single sign on between FutureGrid and Commercial Clouds linked by workflows with the following discussion emphasizing Azure. The current security architecture of FutureGrid is explicitly designed to allow integration with other efforts to integrate with other ongoing NSF OCI based project such as TeraGrid and in future XD. One of the features of FutureGrid will be to provide integration of multiple Identity Providers (one of which will be InCommon) as part of the authentication. Authorization can be based on an LDAP directory that integrates a project registry and allows the definition of group attributes and roles to provide a powerful secure architecture. This approach seamlessly integrates with the Azure framework while using claims based security allowing the integration of external identity providers (in our case InCommon in conjunction with our LDAP server) Thus, developers are able to leverage from both efforts while allowing authorized users of FutureGrid to develop secure applications that can delegate tasks to services found on FutureGrid or the Azure Platform. In this architecture one can either use a "FutureGrid" LiveID (Azure) account or that of individual users.

## Workflow

We need to support workflows that link job components between FutureGrid and Commercial Clouds. One possibility (especially for Azure) is to use Trident from Microsoft Research [1] which is built on top of Windows Workflow Foundation. If Trident runs on Azure, then it will in conventional fashion, use workflow services that are proxies and launch those components that need to run on FutureGrid. Alternatively one could run Trident on FutureGrid and use proxies to launch components on Amazon or Azure.

## Data Transport

The cost (in time and money) of transport of data in (and to a lesser extent) out of commercial clouds is often discussed as a difficulty in using clouds. If commercial clouds become an important component of the National Cyberinfrastructure we can expect that high bandwidth links will be made available between clouds and TeraGrid (and hence FutureGrid). The special structure of cloud data with blocks (in Azure Blobs) and Tables could allow high performance parallel algorithms but initially simple HTTP mechanisms will be used to transport data [2-4] between job components on FutureGrid and Commercial Clouds.

## Program Library

We can extend FutureGrid's virtual machine image library to manage images used in commercial clouds.

## Blobs and Drives

The basic storage concept in clouds is Blobs for Azure and S3 for Amazon. These can be organized (approximately as in directories) by Containers for Azure. Further as well as service interface for Blobs and S3, one can attach "directly" to compute instances as Azure Drives and the Elastic Block Store for Amazon. This concept is similar to shared file systems such as Lustre used in TeraGrid and offered on FutureGrid. The cloud storage is intrinsically fault tolerant while that on FutureGrid needs backup storage (HPSS at Indiana University). However the architecture ideas are similar between clouds and FutureGrid and initially we should just add support for the Simple Cloud File Storage API [5].

## DPFS Data Parallel File System

This covers the support of file systems like Google File System(MapReduce), HDFS (Hadoop) or Cosmos (Dryad) with compute-data affinity optimized for data processing. It could be possible to link DPFS to basic Blob and Drive based architecture but simpler is regard DPFS as application centric storage model with compute-data affinity and Blobs and Drives as the repository centric view. In general data transport will be needed to link these two data views. It seems important to consider this carefully for FutureGrid as DPFS file systems are precisely designed for efficient execution of data-intensive applications. However the importance of DPFS for linkage with Amazon and Azure is not clear as these clouds do not currently offer fine grain support for compute-data affinity. We note here Azure Affinity Groups as one interesting capability [6]. We expect that initially Blobs, Drives, Tables and Queues will be the areas that FutureGrid will most usefully provide a platform similar to Azure (and Amazon).

## Table and NOSQL Non Relational Databases

There has been substantial important developments in simplified database structures -- termed NOSQL [7-8] -- typically emphasizing distribution and scalability. These are present in the three major clouds: Bigtable [9] in Google; SimpleDB [10] in Amazon and Azure Table [11] for Azure. Tables are clearly important in science as illustrated by the VOTable standard in Astronomy [12] and the popularity of Excel. However there does not appear to be substantial experience in using tables outside clouds. There are of course many important uses of non relational databases -- especially in use of triple stores for metadata storage and access. Recently there is interest in building scalable RDF triple stores based on MapReduce and Tables or the Hadoop File System [13-14] with early success reported on very large stores. The current cloud Tables fall into two groups: Azure Table and Amazon SimpleDB are quite similar [15] and support lightweight storage for "document stores" while Bigtable aims to manage large mammoth distributed data sets without size limitations. All these tables are schema free (each record can have different properties) although Bigtable has a Schema for column (property) families. It seems likely that tables will grow in importance for scientific computing and FutureGrid could support this using two Apache projects Hbase [16] for Bigtable and CouchDB [17] for a document store. Another possibility is the open source SimpleDB implementation M/DB [18]. The new Simple Cloud API's [5] for File Storage, Document Storage Services and Simple Queues could help providing a common environment between FutureGrid and commercial clouds.

## SQL and Relational Databases

Both Amazon and Azure clouds offer relational databases and it is straightforward for FutureGrid to offer a similar capability unless there are issues of huge scale where in fact approaches based on Tables and/or MapReduce might be more appropriate[14]. As one early user we are developing on FutureGrid a new private cloud computing model for OMOP Observational Medical Outcomes Partnership for patient related medical data which uses Oracle and SAS where FutureGrid is adding Hadoop for scaling to many different analysis methods.

Note that databases can be used to illustrate two approaches to deploying capabilities . Traditionally one would add database software to that found on computer disk. This software is executed providing your database instance. However on Azure and Amazon, the database is installed on a separate virtual machine independent from your job (worker roles in Azure). This implements "SQL as a Service". It may have some performance issues from messaging interface but the "aaS" deployment clearly simplifies one's system. For N platform features, one only needs N services whereas number of possible images with alternative approach is a prohibitive 2N.

## Queues

Both Amazon and Azure offer similar scalable robust queuing services that are used to communicate between the components of an application. The messages are short (< 8KB) and have a REST Service interface with "deliver at least once semantics". They are controlled by time-outs for posting length and time allowed for a client to process. We can build a similar approach (on the small and less challenging} FutureGrid environment basing it on Publish Subscribe systems ActiveMQ [19] or NaradaBrokering [20-21] with which we have substantial experience.

## Worker and Web Roles

The concepts of roles introduced by Azure is an interesting concept providing non trivial functionality that FutureGrid could offer while preserving the better affinity support that is possible on FutureGrid as it is not fully virtualized. Worker roles are the basic schedulable process and are automatically launched. Note that explicit scheduling is unnecessary in clouds either for individual worker roles or for "gang-scheduling" supported transparently in MapReduce. Queues are a critical concept here as they provide a natural way to manage the task assignment in a fault tolerant distributed fashion.

Web roles provide an interesting approach to portals and here we note that the Google Application Engine is largely aimed at web applications. Science Gateways are very successful in TeraGrid but still require non trivial development. Perhaps the support of Web Roles in FutureGrid could both ease the transition to Azure and make it easier to develop Gateways.

## MapReduce

There has been substantial interest in "data parallel" languages largely aimed at loosely coupled computations which execute over different data samples. The language and runtime generate and provide efficient execution of "many task" problems that are well known as successful Grid applications. However MapReduce summarized in table 3, has several advantages over traditional implementations of many task problems as it supports dynamic execution, strong fault tolerance and an easy to use high level interface. The major commercial MapReduce implementations are Hadoop [22] and Dryad [23-26] with execution possible with or without virtual machines. Hadoop is currently offered by Amazon and we expect Dryad to be available on Azure. On FutureGrid we already intend to support Hadoop, Dryad and other MapReduce approaches including Twister [27] supporting iterative computations seen in many datamining and linear algebra applications. Note that our approach has some similarities with Cloudera[28] which offers a variety of Hadoop distributions including Amazon and Linux.

Table 3: Comparison of MapReduce type systems relevant to FutureGrid

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Google MapReduce [29]** | **Apache Hadoop [22]** | **Microsoft Dryad [25]** | **Twister [27]** | **Azure Twister** [30] |
| **Programming Model** | MapReduce | MapReduce | DAG execution, Extensible to MapReduce and other patterns | Iterative MapReduce | MapReduce-- will extend to Iterative MapReduce |
| **Data Handling** | GFS (Google File System) | HDFS (Hadoop Distributed File System) | Shared Directories & local disks  | Local disks and data management tools | Azure Blob Storage  |
| **Scheduling** | Data Locality | Data Locality; Rack aware, Dynamic task scheduling through global queue | Data locality;Networktopology basedrun time graphoptimizations; Static task partitions | Data Locality; Static task partitions | Dynamic task scheduling through global queue |
| **Failure Handling** | Re-execution of failed tasks; Duplicate execution of slow tasks | Re-execution of failed tasks; Duplicate execution of slow tasks | Re-execution of failed tasks; Duplicate execution of slow tasks | Re-execution of Iterations | Re-execution of failed tasks; Duplicate execution of slow tasks |
| **High Level Language Support** | Sawzall [31] | Pig Latin [32-33] | DryadLINQ [26] | Pregel [34] has related features | N/A |
| **Environment** | Linux Cluster.  | Linux Clusters, Amazon Elastic Map Reduce on EC2 | Windows HPCS cluster | Linux ClusterEC2 | Window Azure Compute, Windows Azure Local Development Fabric |
| **Intermediate data transfer** | File  | File, Http | File, TCP pipes, shared-memory FIFOs | Publish/Subscribe messaging | Files, TCP |

## Software as a Service

Services are used in a similar fashion in commercial clouds and most modern distributed systems. We expect users to package their programs wherever possible and so no special support is needed to enable Software as a Service. In section 3.8, we already discussed the advantages of "Systems Software as a Service".

**References**

1. Microsoft. *Project Trident: A Scientific Workflow Workbench*. 2010 [accessed 2010 June 3]; Available from: http://research.microsoft.com/en-us/collaboration/tools/trident.aspx.

2. Wei Lu, Jared Jackson, and Roger Barga, *AzureBlast: A Case Study of Developing Science Applications on the Cloud*, in *ScienceCloud: 1st Workshop on Scientific Cloud Computing co-located with HPDC 2010 (High Performance Distributed Computing)*. 2010, ACM: Chicago, IL.

3. Distributed Systems Laboratory (DSL) at University of Chicago Wiki. *Performance Comparison:Remote Usage, NFS, S3-fuse, EBS*. 2010 [accessed 2010 June 8]; Available from: http://dsl-wiki.cs.uchicago.edu/index.php/Performance\_Comparison:Remote\_Usage,\_NFS,\_S3-fuse,\_EBS.

4. Darren Jensen. *Blog entry on Compare Amazon S3 to EBS data read performance*. 2009 December 30 [accessed 2010 June 8]; Available from: http://jensendarren.wordpress.com/2009/12/30/compare-amazon-s3-to-ebs-data-read-performance/.

5. Zend PHP Company. *The Simple Cloud API for Storage, Queues and Table*. 2010 [accessed 2010 June 1]; Available from: http://www.simplecloud.org/home.

6. Microsoft. *Windows Azure Geo-location Live*. 2009 April 30 [accessed 2010 June 5]; Available from: http://blogs.msdn.com/b/windowsazure/archive/2009/04/30/windows-azure-geo-location-live.aspx.

7. NOSQL Movement. *Wikipedia list of resources*. 2010 [accessed 2010 June 5]; Available from: http://en.wikipedia.org/wiki/NoSQL.

8. NOSQL Link Archive. *LIST OF NOSQL DATABASES*. 2010 [accessed 2010 June 5]; Available from: http://nosql-database.org/.

9. Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and R.E. Gruber, *Bigtable: A Distributed Storage System for Structured Data*, in *OSDI'06: Seventh Symposium on Operating System Design and Implementation*. 2006, USENIX: Seattle, WA.

10. Amazon. *Welcome to Amazon SimpleDB*. 2010 [accessed 2010 June 5]; Available from: http://docs.amazonwebservices.com/AmazonSimpleDB/latest/DeveloperGuide/index.html.

11. Jai Haridas, Niranjan Nilakantan, and B. Calder. *WINDOWS AZURE TABLE* 2009 May [accessed 2010 June 5]; Available from: http://go.microsoft.com/fwlink/?LinkId=153401.

12. International Virtual Observatory Alliance. *VOTable Format Definition Version 1.1*. 2004 [accessed 2010 June 5]; Available from: http://www.ivoa.net/Documents/VOTable/20040811/.

13. Apache Incubator. *Heart (Highly Extensible & Accumulative RDF Table) planet-scale RDF data store and a distributed processing engine based on Hadoop & Hbase.* . 2010 [accessed 2010 June 1]; Available from: http://wiki.apache.org/incubator/HeartProposal.

14. Raytheon BBN. *SHARD (Scalable, High-Performance, Robust and Distributed) Triple Store based on Hadoop*. 2010 March [accessed 2010 June 5]; Available from: http://www.cloudera.com/blog/2010/03/how-raytheon-researchers-are-using-hadoop-to-build-a-scalable-distributed-triple-store/.

15. Matthew King. *Amazon SimpleDB and CouchDB Compared*. 2007 December 14 [accessed 2010 June 5]; Available from: http://www.automatthew.com/2007/12/amazon-simpledb-and-couchdb-compared.html.

16. Apache. *Hbase implementation of Bigtable on Hadoop File System*. 2010 [accessed 2010 June 5]; Available from: http://hbase.apache.org/.

17. Apache. *The CouchDB document-oriented database Project*. 2010 [accessed 2010 June 5]; Available from: http://couchdb.apache.org/index.html.

18. M/Gateway Developments Ltd. *M/DB Open Source "plug-compatible" alternative to Amazon's SimpleDB database*. 2009 [accessed 2010 June 5]; Available from: http://gradvs1.mgateway.com/main/index.html?path=mdb.

19. (2009) *ActiveMQ*.http://activemq.apache.org/.

20. Pallickara, S. and G. Fox. *NaradaBrokering: a distributed middleware framework and architecture for enabling durable peer-to-peer grids*. in *ACM/IFIP/USENIX 2003 International Conference on Middleware*. 2003. Rio de Janeiro, Brazil: Springer-Verlag New York, Inc.

21. NaradaBrokering. *Scalable Publish Subscribe System*. 2010 [accessed 2010 May]; Available from: http://www.naradabrokering.org/.

22. (2009) *Apache Hadoop*.http://hadoop.apache.org/.

23. Ekanayake, J., A.S. Balkir, T. Gunarathne, G. Fox, C. Poulain, N. Araujo, and R. Barga. *DryadLINQ for Scientific Analyses*. in *Fifth IEEE International Conference on eScience: 2009*. 2009. Oxford: IEEE.

24. Ekanayake, J., T. Gunarathne, J. Qiu, G. Fox, S. Beason, J.Y. Choi, Y. Ruan, S.H. Bae, and H. Li, *Applicability of DryadLINQ to Scientific Applications*. 2009, Community Grids Laboratory, Indiana University.

25. Isard, M., M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. *Dryad: Distributed data-parallel programs from sequential building blocks*. in *ACM SIGOPS Operating Systems Review*. 2007: ACM Press.

26. Yu, Y., M. Isard, D. Fetterly, M. Budiu, U. Erlingsson, P.K. Gunda, and C. J. *DryadLINQ: A System for General-Purpose Distributed Data-Parallel Computing Using a High-Level Language*. in *Symposium on Operating System Design and Implementation (OSDI)*. 2008.

27. J.Ekanayake, H.Li, B.Zhang, T.Gunarathne, S.Bae, J.Qiu, and G.Fox., *Twister: A Runtime for iterative MapReduce*, in *Proceedings of the First International Workshop on MapReduce and its Applications of ACM HPDC 2010 conference June 20-25, 2010*. 2010, ACM: Chicago, Illinois.

28. Cloudera. *CDH: A free, stable Hadoop distribution offering RPM, Debian, AWS and automatic configuration options.* 2010 [accessed 2010 June 5]; Available from: http://www.cloudera.com/hadoop/.

29. Dean, J. and S. Ghemawat, *MapReduce: simplified data processing on large clusters.* Commun. ACM, 2008. **51**(1): p. 107-113.

30. Thilina Gunarathne, *MapReduce Implementation on Azure*, Personal Communication to, G. Fox. 2010.

31. Pike, R., S. Dorward, R. Griesemer, and S. Quinlan, *Interpreting the data: Parallel analysis with sawzall.* Scientific Programming Journal Special Issue on Grids and Worldwide Computing Programming Models and Infrastructure vol. 13, no. 4, 2005: p. 227–298.

32. Pig! *Platform for analyzing large data sets*. 2010; Available from: http://hadoop.apache.org/pig/.

33. Christopher Olston, Benjamin Reed, Utkarsh Srivastava, Ravi Kumar, and Andrew Tomkins, *Pig latin: a not-so-foreign language for data processing*, in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 2008, ACM: Vancouver, Canada. p. 1099-1110

34. Grzegorz Malewicz, Matthew H. Austern, Aart J.C. Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski, *Pregel: a system for large-scale graph processing*, in *Proceedings of the twenty-first annual symposium on Parallelism in algorithms and architectures*. 2009, ACM: Calgary, Canada. p. 48-48.