How Replicated Data Management in the Cloud can benefit from a Data Grid Protocol — the Re:GRIDiT Approach*

Laura Cristiana Voicu
Database and Information Systems Group
University of Basel, Switzerland
laura.voicu@unibas.ch

Heiko Schuldt
Database and Information Systems Group
University of Basel, Switzerland
heiko.schuldt@unibas.ch

ABSTRACT

Cloud computing has recently received considerable attention both in industry and academia. Due to the great success of the first generation of Cloud-based services, providers have to deal with larger and larger volumes of data. Quality of service agreements with customers require data to be replicated across data centers in order to guarantee a high degree of availability. In this context, Cloud Data Management has to address several challenges, especially when replicated data are concurrently updated at different sites or when the system workload and the resources requested by clients change dynamically. Mostly independent from recent developments in Cloud Data Management, Data Grids have undergone a transition from pure file management with read-only access to more powerful systems. In our recent work, we have developed the Re:GRIDiT protocol for managing data in the Grid which provides concurrent access to replicated data at different sites without any global component and supports the dynamic deployment of replicas. Since it is independent from the underlying Grid middleware, it can be seamlessly transferred to other environments like the Cloud. In this paper, we compare Data Management in the Grid and the Cloud, briefly introduce the Re:GRIDiT protocol and show its applicability for Cloud Data Management.

Categories and Subject Descriptors

H.2.4 [**Systems**]: Distributed databases; H.3.4 [**Systems and Software**]: Distributed systems

General Terms

Algorithms

Keywords

Cloud Data Management, Data Grid, Replication.

*Partly supported by the Hasler Foundation (project `COSA`) and the 7^{th} Framework Programme of the EU (`D4Science`).

CloudDB'09, November 2, 2009, Hong Kong, China.

1. INTRODUCTION

Cloud computing has recently received considerable attention both in industry and academia [3]. Basically, Cloud-based computing allows customers to rent hardware and/or software resources, thus being freed from significant investments in building up and maintaining computing centers in-house by outsourcing their complete ICT infrastructure. Resources are made available according to Quality-of-service (QoS) guarantees which are negotiated between provider and customer. Depending on the type of resources which are made available to customers and the services which have been negotiated, there is a distinction between Infrastructure as a Service (IaaS) [2], Platform as a Service (PaaS), and Software as a Service (SaaS). Providers of Cloud-based services usually maintain different distributed data centers. This allows to dynamically adapt the resources provided for a particular customer based on their current needs (within the QoS agreement that has been negotiated).

A core challenge in the context of Cloud computing is the management of very large volumes of data. This is completely independent of the type of resource which is shared in the Cloud – databases are either directly visible and accessible to customers as part of the infrastructure/platform, or are hidden behind service interfaces. In terms of data management, QoS guarantees mainly encompass a high degree of availability. For providers of Cloud services, this means that data need to be partitioned and replicated across different data centers. Although traditional high throughput OLTP applications are most likely not to become the predominant applications hosted in a Cloud environment [1], replicated data management nevertheless needs to take into account updates which are performed on replicated data (either directly or via service calls). Replicated data management in the context of concurrent updates to different replicas can be addressed either by using well-established protocols such as Strict Two-Phase Locking (2PL) in combination with Two-Phase Commit (2PC) [10], or by relaxing ACID properties to increase the overall performance and throughput of the system. The latter is applied in several of today's Cloud environments (e.g., PNUTS [7]).

During the last few years, mainly motivated by the need of applications in eScience where vast amounts of data are generated by specialized instruments and need to be collaboratively accessed, processed and analyzed by a large number of scientists around the world, Grid computing has become increasingly popular [9]. The Grid started as a vision to share potentially unlimited computing power over the Internet to solve complex problems in a distributed way – a

	Cloud Data Management	Data Grids
Distribution	Few data centers	Many Grid nodes (presumably much larger than the number of data centers in the Cloud)
Environment	Homogeneous resources in data centers	Heterogeneous Grid nodes
Operations for Data Access	Usually SQL-based access to (object-)relational databases	Distinction between mutable and immutable data, materialized in different sets of access services
Replication	Needed as a consequence of QoS guarantees (availability); must be transparent to customers	Needed in order to guarantee a high degree of availability; must be transparent to Grid users and developers
Replication Granularity	Fine-grained replication (individual tuples of database tables), due to multi-tenancy	Course-grained replication sufficient (files, relations or partitions of relations)
Updates	No traditional OLTP workloads, but (concurrent) updates to replicated data need to be supported	First generations focused on read-only data; novel eScience applications also demand updates to replicated data
Global Control	Most solutions have some global component which might lead to a single point of failure or performance bottleneck	Most Data Grids consider global replica catalogs; yet novel approaches like Re:GRIDiT avoid any global component in the system
Global Correctness	Relaxation of ACID properties	Most Data Grids do not support concurrent updates; Re:GRIDiT provides provable correct (serializable) executions without global component
Dynamic Changes	Horizontal scaling and support for unpredictable workloads necessitate support for the dynamic generation of replicas	Most current approaches consider only static replicas; however, support for dynamic deployment is required by novel eScience applications to move data close to its users

Table 1: Cloud Data Management vs. Data Grids: a Comparison

first generation of Grids, so-called computational Grids, focused on CPU cycles as resources to be shared. Recent advances in Grid computing aim at virtualizing different types of resources (data, instruments, computing nodes, tools) and making them transparently available. Motivated by the success of computational Grids, a second generation of Grids, namely Data Grids, has emerged as a solution for distributed data management in data-intensive applications. The size of data required by these applications may be upto petabytes. In the earth observation community, for instance, data are acquired from satellites, sensors and other data acquisition instruments, archived along with metadata, catalogued and validated. According to [12], by the year 2010 the earth observation data archives around the world will grow to around 9.000 Tbytes and by the year 2014 to around 14.000 Tbytes. In many applications, Data Grids not only maintain raw data produced by instruments, but need to take into account also reports derived out of these raw data and image interpretations that are periodically generated and potentially concurrently updated by scientists at several sites [5]. Similar to Cloud-based systems, a high degree of availability of data can only be achieved by means of replication across different nodes in the Grid. In the presence of concurrent updates to replicas, consistency needs to be guaranteed.

In terms of replication management, most Data Grids suffer from several shortcomings as they merely deal with files as the replication granularity, do not allow replicated data to be updated, and/or require the manual placement of files [8, 13]. Recently, we have developed the Re:GRIDiT system that provides advanced data and replication management in the Grid [16, 17]. Re:GRIDiT follows a truly distributed approach to replication management in the Grid by bringing together replication management, originally developed for database clusters [15, 4], and distributed transaction management that does not rely on a global coordinator [11]. In particular, it has been designed to be independent from any underlying Grid middleware. Thus, Re:GRIDiT can also be seamlessly deployed in other environments like the Cloud.

In this paper, we compare requirements and state of the art in the Data Grid and in Cloud Data Management (Section 2). Then, we briefly introduce the Re:GRIDiT protocol, show its applicability for Cloud Data Management, and provide performance results of the evaluation of the protocol at Cloud-scale (Section 3). Section 4 concludes.

2. DISTRIBUTED DATA MANAGEMENT: CLOUD VS. GRID

Data Grids and Cloud Data Management share similar objectives. However, the development of the Grid and of the Cloud have only been loosely coupled for several reasons. First, they both focused on specific user communities: scientific communities (eScience) in case of the Grid vs. the outsourcing of ICT services for commercial customers in case of the Cloud. Second, both environments have different origins: the main driver for the Grid has been the High Energy Physics community (other eScience communities have adopted the Grid rather recently), while the proliferation of the Cloud has been dominated by large providers of IT services that already had the necessary computing resources (data centers) in place and were heading towards a more optimal utilization of their capacities. Third, the initial requirements which have been addressed were different. In the Data Grid, first solutions have focused on the controlled sharing of files within Virtual Organizations (VOs). Data management at a granularity finer than files, replication management and updates have only very recently been put on the list of requirements due to novel eScience applications (e.g., earth observation, healthcare, etc.). For Cloud-based environments, analytical data management has been identified as the predominant application [1]. However, in the presence of QoS constraints that need to be met by Cloud service providers, data need to be replicated across data centers. Although the percentage of updates will be rather low compared to traditional OLTP settings, Cloud Data Management nevertheless needs to provide correct and consistent data management in the presence of conflicting updates. Therefore, despite the initial separation between Data Grids and Cloud Data Management, the requirements both environments need to address more and more converge.

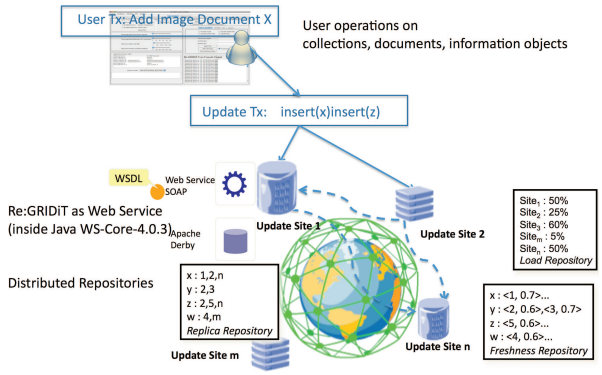

Figure 1: Re:GRIDiT Implementation

According to [6], the Cloud needs to meet the following requirements:

Multi-tenancy: *A Cloud service must support multiple, organizationally distant customers.* Multi-tenancy is also an important requirement in the Data Grid. However, support for VOs and thus for different users/customers within the same distributed infrastructure is already integral part of most Grid middleware systems.

Elasticity & Resource Sharing: *Tenants should be able to negotiate and receive resources on-demand. Spare Cloud resources should be transparently applied when a tenant's negotiated QoS is insufficient [14].* Similarly, the resources made available to a VO in a Grid should be dynamically adapted to its current needs. Therefore, QoS-based resource negotiation and allocation has recently become an important topic also in the Grid community.

Horizontal Scaling & Security: *It should be possible to add Cloud capacity in small increments, transparent to the tenants. A Cloud service should be secure in that tenants are not made vulnerable because of loopholes in the Cloud.* Similar requirements can also be found in the Grid.

Metering: *A Cloud service must support accounting that reasonably ascribes operational and capital expenditures to each of the tenants of the service.* As Data Grids have their origin in scientific communities and operate on resources which are contributed by the member institutions of VOs on a voluntary basis, this aspect has not yet been in the main focus of Grid environments.

Availability: *A Cloud service should be highly available.* This requirement case also be found in the Grid. In addition, as services need the (local) presence of the data they access, availability should be extended also to the underlying data sources by means of replication.

Operability: *A Cloud service should be easy to operate.* In general, this requirement also holds for the Grid. However, due to the heterogeneous environment in which Grids can be deployed, some current Grid middleware solutions are rather limited in that regard. But with the proliferation of Service Grids, this limitation more and more diminishes.

Table 1 summarizes the relationship between Cloud Data Management and Data Grids. The table shows that differences between both fields still exist. However, Re:GRIDiT which follows a novel approach to data management in the Grid by making use of and extending protocols that have originally been devised for database clusters, can be considered a major contribution to the convergence of both areas.

3. RE:GRIDIT

Re:GRIDiT has been developed to address the needs of novel data-intensive eScience applications in the Grid [16]. In short, Re:GRIDiT can be characterized as follows:

Replication management: Grid sites are classified as update sites (where data can be updated or read) or read-only sites (where only read access is allowed). Eager replication is applied among update sites. In addition, replication mechanisms between update and read-only sites are adopted that take into account different levels of freshness (cf. [15]).

Distributed concurrency control: user operations can span several sites, i.e., require support for distributed transaction management (when data which is read or updated in a single transaction is distributed across several sites).

No global coordinator: Re:GRIDiT enforces global correctness and consistency by means of globally serializable schedules in a completely distributed way without relying on a central coordinator with complete global knowledge [17].

Mutable and immutable data objects: Immutable data objects are created only once and kept until deleted. Mutable data objects can be subject to updates. The latter will be the predominant type of data objects in Cloud applications.

Dynamic replica deployment and management: Data objects are dynamically distributed across several replica sites to raise throughput by moving frequently used/heavy accessed data objects to relatively inactive replica sites (where they do not compete against each other for resources) so that requests can be handled faster [18].

In order to show the potential of Re:GRIDiT for Cloud Data Management, we have evaluated the system in realistic Cloud settings. Since Cloud-based environments typically contain less but more powerful resources than a Grid (e.g., several data centers of a Cloud service provider rather than a large machines with free capacities within an eScience community), we have run experiments with up to 12 sites. All sites have been equipped with a Dual Intel® CPU 3.20 GHz processor, 5 GB RAM and running Ubuntu Linux 8.0.4 as operating system. All hosts are equipped with a local Derby database and Java WS-Core. The setup is schematically presented in Figure 1.

3.1 Synchronization of Updates

We conducted several experiments with varying conflict rates (1%, 5%, 10%, 20%) and have compared Re:GRIDiT to 2PC/S2PL. Figure 2 shows the throughput of the protocol when the transaction length is kept constant at 5 operations per transaction and for a fixed number of replica sites. In this setup, the measurement have consisted of runs of 500 transactions. Even for a high conflict rate of 20% Re:GRIDiT proves to perform better, although the difference in throughput is smaller than for a conflict rate of 1%. Hence, Re:GRIDiT is highly appropriate for managing replicated data in typical Cloud applications (i.e., conflicts exist but are rather infrequent compared to traditional OLTP workloads). More detailed evaluation results of Re:GRIDiT at Grid-scale with up to 48 update sites can be found in [17].

3.2 Static vs. Dynamic Replica Deployment

We have compared Re:GRIDiT with a static approach to replica management. In the dynamic approach, the initial setup considers 12 sites while the static protocol has been configured with 12 and 6 sites. With the given workload, the dynamic replication protocol has stabilized the number

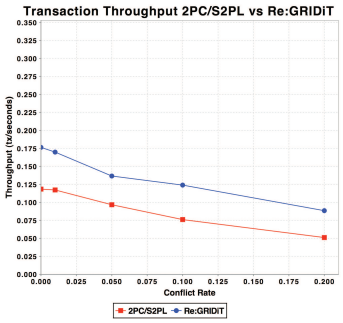

Figure 2: Replica Updates: Throughput Re:GRIDiT vs. 2PC

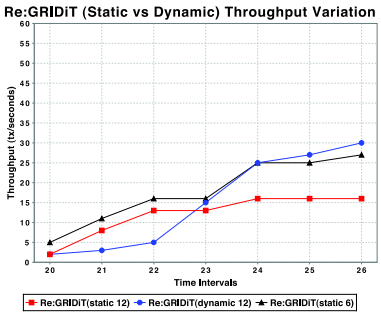

Figure 3: Replica Deployment: Throughput Static vs. Dynamic

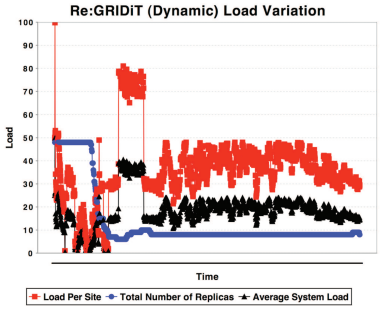

Figure 4: Dyn. Replica Deployment: Load Variation in Time

of update sites at a minimum of 6. As it can be seen from Figure 3, the throughput in the dynamic setting is higher than in the static setting. For 12 updates sites, the static Re:GRIDiT requires more time to synchronize the update to a higher (and constant) number of update sites. It can also be observed that initially the throughput in the dynamic setting is smaller than in the case of the static case with 6 update sites, due to the extra load imposed by the release of the unnecessary update sites. With a lower number of sites, the throughput of the dynamic setting is increasing and finally exceeds the one of the static setting with 6 update sites due to the dynamic selection of the least loaded sites to host replicas.

Finally, we have evaluated the load variation in the presence of additional read transactions at the sites. For this, we have recorded the mean local load variations, the mean system loads and the evolution of the number of replicas in time (dynamic setting). Figure 4 shows that for an initial number of 48 update sites, the average number of replicas converges to 6 while the impact on the sites' load is moderate as soon as the number of replicas has stabilized. More details on the comparison of dynamic replica deployment and undeployment with a static approach can be found in [18].

4. CONCLUSIONS AND OUTLOOK

Although having started as specialized solutions for different communities and with different sets of requirements, Cloud Data Management and Data Grids are more and more converging. In this paper, we have analyzed the commonalities between both areas and the differences that still exist. We have shown that Re:GRIDiT, a protocol that has originally been devised for replicated data management in the Grid, is very well suitable also for Cloud Data Management. In our future work, we plan to integrate support for more fine-grained replication into Re:GRIDiT for supporting multi-tenant applications in the Cloud.

5. REFERENCES

[1] Daniel Abadi. Data Management in the Cloud: Limitations and Opportunities. In *IEEE DE Bulletin*, 32(1):3–12, 2009.

[2] A. Aboulnaga, K. Salem, et al. Deploying Database Appliances in the Cloud. In *IEEE DE Bulletin*, 32(1):3–12, 2009.

[3] R. Agrawal et al. The Claremont Report on Database Research. In *Comm. ACM 2009*, pages 56-65. Vol. 52.

[4] F. Akal, C. Türker, H.-J. Schek, Y. Breitbart, T. Grabs, and L. Veen. Fine-Grained Replication and Scheduling with Freshness and Correctness Guarantees. In *VLDB*, pages 565–576, 2005.

[5] L. Candela, F. Akal, et al. DILIGENT: integrating digital library and Grid technologies for a new Earth observation research infrastructure. *Int. J. Digit. Libr.*, 7(1):59–80, 2007.

[6] B. Cooper, E. Baldeschwieler et al. Building a Cloud for Yahoo!. In *IEEE DE Bulletin*, 32(1):3–12, 2009.

[7] B. Cooper, R. Ramakrishnan et al. PNUTS: Yahoo!'s Hosted Data Serving Platform. In *PVLDB*, 1(2):1277–1288, 2008.

[8] EDG: The European DataGrid Project. http://eu-datagrid.web.cern.ch/eu-datagrid/.

[9] I. Foster and C. Kesselman *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2nd Edition, 2003.

[10] J. Gray and A. Reuter. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann 1993.

[11] K. Haller, H. Schuldt, and C. Türker. Decentralized coordination of transactional processes in peer-to-peer environments. In *CIKM*, pages 28–35, Bremen, Germany, 2005.

[12] R. Harris and N. Olby. Archives for Earth observation data. *Space Policy*, 16(13):223–227, 2007.

[13] M. Manohar, et all. A Replica Location Grid Service Implementation. In *GGF Data Area Workshop*, 2004.

[14] N. Paton et al. Optimizing Utility in Cloud Computing through Autonomic Workload Execution. In *IEEE DE Bulletin*, 32(1):3–12, 2009.

[15] U. Röhm, K. Böhm, H.-J. Schek, and H. Schuldt. FAS: a freshness-sensitive coordination middleware for a cluster of OLAP components. In *VLDB '02*, pages 754–765. VLDB Endowment, 2002.

[16] L. C. Voicu, H. Schuldt, Y. Breitbart and H. -J. Schek. Replicated Data Management in the Grid: The Re:GRIDiT Approach. In *Proc. ACM DaGreS'09*, May 2009. Italy.

[17] L. C. Voicu, H. Schuldt, F. Akal, Y. Breitbart and H. -J. Schek. Re:GRIDiT – Coordinating Distributed Update Transactions on Replicated Data in the Grid. In *Proc. Grid'09*, October 2009. Canada.

[18] L. C. Voicu and H. Schuldt Load-aware Dynamic Replication Management in a Data Grid. In *Proc. CoopIS'09*, November 2009. Portugal.