

Enterprise Grids: Challenges Ahead

R. Jiménez-Peris · M. Patiño-Martínez · B. Kemme

Received: 15 October 2006 / Accepted: 17 January 2007
© Springer Science + Business Media B.V. 2007

Abstract Grid technologies have matured over the last few years. This level of maturity is especially true in the field of scientific computing in which Grids have become the main infrastructure for scientific problem solving. Due to its success, the use of Grid technology rapidly finds its introduction into other fields. One of such fields is enterprise computing in which Grids are seen as a new architecture for data centers. In this paper, we describe the vision of enterprise Grids, current scientific achievements that will leverage this vision, and challenges ahead.

1 Introduction

Grids have succeeded in providing an infrastructure for deploying parallel applications in a distributed setting with a high degree of automation. The definition of Grids has been redefined along the years. Initially Grids were defined as an infrastructure to provide easy and inexpensive access to high-end computing [24]. Then, it was refined in [18] as an infrastructure to share resources for collaborative problem solving. More recently, in [19] the Grid definition evolves into an infrastructure to pool and virtualize resources and enable their use in a transparent fashion.

Grid infrastructure exhibits several interesting features. One of the main functionalities is that a Grid hides the heterogeneity of its underlying components such as hardware, operating systems or storage systems, and serves as a middleware that provides seamless connectivity of the components it manages. Another interesting characteristic is the transparent pooling of many kinds of resources such as computing power, storage, data, and services. This pooling enables applications deployed on the Grid to transparently share resources and utilize the provided capacity more effectively. A related attribute is the ability to allocate and reserve virtualized resources. Virtualization facilitates the sharing of resources: it allows the preservation of quality of service guarantees for time critical applications, and at

This work has been partially funded by the European S4ALL project, the Spanish Ministry of Education and Science (MEC) under grant TIN2004-07474-C02-01, the Madrid Regional Research Council (CAM) under grant 0505/TIC/000285, and the Spanish Ministry of Industry (MITyC) under grant FIT-340000-2006-138.

R. Jiménez-Peris (✉) · M. Patiño-Martínez
Facultad de Informática, Universidad Politécnica
de Madrid (UPM), Madrid, Spain
e-mail: rjimenez@fi.upm.es

M. Patiño-Martínez
e-mail: mpatino@fi.upm.es

B. Kemme
McGill University, School of Computer Science,
Montreal, Quebec, Canada
e-mail: kemme@cs.mcgill.ca

the same time offers true resource scavenging by taking advantage of unused resources to perform batch computations.

These features make Grid computing attractive for enterprises that want to convert it into their infrastructure for deploying enterprise applications. This interest has resulted in minting a new term, *Enterprise Grids*. Enterprise Grid computing reflects the use of Grid computing within the context of a business or enterprise rather than for scientific applications. A first generation of enterprise Grid solutions, as developed by IBM [15] or Oracle [47], start to appear. A proof for the increasing interest in Enterprise Grids is the recent creation of the Enterprise Grid Alliance (EGA) with the mission of developing a common Enterprise Grid reference model [14] and fostering the use of Grids for enterprise computing. Very recently EGA and the Global Grid Forum have joined into the Open Grid Forum (OGF) [46]. OGF aims at bringing together the goals of academic and enterprise applications.

However, Enterprise Grids have not yet found a wide adoption in industry. The problem is that in order to effectively apply Grid technology to enterprise applications, many technical and scientific hurdles still have to be overcome. Originally, the need to exploit distributed resources for compute intensive scientific applications triggered the development of Grid architectures. These applications are well-suited for automated distribution and scheduling, and Grid technology focused on providing a virtualized abstraction of computing resources to these applications. However, enterprise applications have many characteristics that make them inherently harder to deploy on a Grid infrastructure. Examples are the stateful nature of business applications, the typical underlying multi-tier architecture where request execution follows a complex path through a diverse set of components, the existence of transactional data, and the need for transactional interaction between the different application components. In this paper, we aim at identifying some of the gaps between what Grid infrastructures currently provide and what enterprise Grids need to offer. From there, we survey recent achievements in the field of distributed computing that will help to address

some of the issues, and identify the open challenges to realize the enterprise Grid vision.

The paper is structured as follows. First, in Section 2 we identify some of the main requirements in Enterprise Grids and contrapose them with current functionality provided by Grid systems in order to identify any mismatches. Then, we describe some of the recent advances in distributed systems that can be helpful to overcome the discrepancies in Section 3. We continue in Section 4 by identifying the existing open scientific challenges to realize the Enterprise Grid vision. Finally we present our conclusions in Section 5.

2 The Gap between Existing Grid Technology and the Requirements for Enterprise Grids

In this section we explore in detail several gaps between what is currently provided by Grid technology and what is needed to make Grids support a wider range of enterprise applications.

Gap 1: Support for Online Applications Traditional enterprise computing is usually composed of a mix of batch and online applications. Grids already have a good grip on batch applications and provide sophisticated automation making batch applications run efficiently on large distributed infrastructures. Abstractions such as “task bags” can very well serve as a basis for automating the programming and execution of batch enterprise applications. For such tasks, the use of scavenging idle resources within an organization allows for better resource utilization and increased throughput.

However, a large fraction of enterprise applications are inherently interactive. We refer to them as online applications because an end user is directly connected to the system. Online applications require timely execution of requests and prompt responses to the client. As such, a single bottleneck within the request execution can lead to unsatisfying performance. That is, for online applications, the main performance metrics is the average response time, often with strict constraints on its statistical distribution. This need is clearly documented in benchmarks for online systems such as TPC-C for database applications [10], ECperf [64] and SPECjAppServer [63] for

information systems, or TPC-W [11] for web based systems. Much of the current Grid technology, however, focuses on optimizing the throughput for batch-style applications, i.e., serving a large set of tasks without paying attention to the response time of individual requests.

Gap 2: Support for Transactional Data Business applications are often data-intensive. They access, process and manipulate large amounts of data, most of which reside in database repositories. Access to such data is nearly always in the context of transactions in order to guarantee consistency and durability of changes to the data. While general computational requests can be assigned to computing resources quite arbitrarily and at runtime, database requests have to be executed at sites that contain the database. Dynamic migration or replication of entire databases to currently free machines, however, cannot be achieved that easily. It also has the additional overhead of synchronization protocols that keep track of all data replicas for consistency purposes. For instance, while Oracle 10g allows a database to run across multiple nodes in a cluster, its configuration of database servers is quite complex and relatively static.

Thus, current provisioning in most Grid environments focuses on stateless applications, and sometimes on applications that access read-only persistent data, in which replication and caching can be applied quite easily.

There has been some work related to “data Grids.” The emphasis has been on data integration and management of large collections of data. The goal of data integration is to provide a homogeneous view of heterogeneous data schemas containing similar information. Data integration can be an important issue in some enterprise applications in which data from different domains exhibit different schemas and a homogeneous view and access means should be provided. Additionally, there is the trend of converting data access into services. In this context, the OGSA-DAI [45] middleware provides support for data integration and access via the Grid. It supports the exposure of various data sources (such as relational and XML data), via homogeneous web

service interfaces, which can be a very useful abstraction for business applications. In general, there has been a tremendous amount of work in the context of data integration, and OGSA-DAI has performed first steps to bring data integration to the Grid. As such, data integration might not be a challenging gap anymore. However, guaranteeing transactional properties in a heterogeneous and distributed Grid environment raises new challenges.

Gap 3: Support for Stateful and Transactional Applications Business applications do not only access transactional data residing in backend databases, they often maintain state within the application code. Often, this state reflects the interactions between an online client and the application (such as session information or a shopping cart). Such interaction patterns reduce the flexibility in which individual tasks can be assigned to Grid resources. Another problem in this context is that a failure of a site can result in losing important application state. Business processes can last for days, weeks, months, or even years. This is very common for workflows and orchestrated web services. The loss of a stateful interaction would be unacceptable in this context. Furthermore, advanced transaction models are often used [26, 67] in this context in order to relax isolation. In the advent of a failure one possibility to attain consistency is to abort all running transactions. However, for long running transactions this is not really an option since they have to progress forward despite failures, unless it is impossible to attain their goal. Availability support for such long running applications is limited with current Grid technology.

Gap 4: Support for Multi-tier Architectures Enterprise applications are typically deployed on multi-tier architectures consisting of web, application server and database tiers, such as J2EE [65]. As mentioned in Gaps 2 and 3, managing each of the individual tiers on the Grid is challenging, in particular because of the state they manage. Providing a Grid solution that supports multiple tiers is even more complicated due to the interaction patterns between the tiers. For instance, web-user requests typically trigger execution on all tiers.

Often the execution on all tiers is encapsulated in a single transaction, that is, the actions of the transaction can span data updates on all tiers. This means, gridification of multi-tier applications needs to preserve consistency across tiers considering any data and execution dependencies between the tiers. Also failures need to be handled appropriately. That is, failover should be performed in such a way that consistency is still preserved. Certainly, Grid systems provide support for multiple applications and servers. However, this support tends to deal with different servers as isolated entities. In order to extend functionality to multi-tier systems, a systematic approach is needed to understand the interaction patterns between adjacent tiers, the execution across all tiers, and the impact of both on the gridification process. Current Grid-support for multi-tier or multi-component applications is limited. We believe much more has to be done in this direction.

Gap 5: Autonomic support Another important goal for the Enterprise Grid vision is that of autonomic management [34]. Autonomic management encompasses several attributes such as self-healing, self-provisioning, self-optimizing, and self-configuring. *Self-healing* capabilities, which allow tolerating failures, provide the Enterprise Grid continuous availability. Fault tolerance is typically attained by replication. Another facet required to realize a self-healing enterprise Grid and the associated high availability is the ability to recover failed sites. Most approaches to recovery of replicas imply stopping system operation in order to obtain a quiescent state and transfer it to the recovering replica. Once the recovery is completed the system operation is resumed. However, this offline recovery goes against the initial goal of high availability. On the other hand, recovery is needed to maintain a particular level of availability. Otherwise, replicas will fail and eventually the system will become unavailable. Therefore, there is a need for performing recovery in a non-intrusive fashion maintaining the system online. Currently, Grid systems provide self-healing facilities but mainly for stateless applications. Sites can join and leave, but little is done to keep the consistency between the state of working sites and the sites joining

the Grid. Therefore, there is still a large gap between the needs of Enterprise Grids and what is provided by current Grid systems relative to stateful applications.

Another essential aspect of enterprise Grids is performance and the need for *self-optimization*. Underlying middleware and database systems have numerous parameters that enable to tune them to maximize their performance. Unfortunately, these parameters are set manually and require a highly qualified administrator. What is more, the right value for these parameters depends on the actual workload. An enterprise Grid cannot rely on human tuning and should be able to tune itself to maximize performance. Ideally, human administrators only need to set high level goals in form of utility functions to hint the system in which dimensions to maximize first. Autonomic performance tuning implies to monitor the system load and performance. When the system detects that performance decreases, tuning parameters need to be adjusted in order to stabilize the system and achieve a configuration that provides optimal performance for the the current workload and its characteristics. This self-optimizing behavior can only be built upon an analytical model of the system that enables to understand the system state by monitoring the relationship between load and performance and knowing which parameters influence this relationship. This means that the self-optimizing support requires specific work for each kind of server, and in the case of enterprise Grids, for each particular tier. Certainly, there are currently no provisions for self-optimization of multi-tier architectures in Grids.

Self-optimization at each individual site/server/tier guarantees that its performance is maximized given the received workload. However, individual self-optimization is not enough to maximize performance at the global level. The reason is that some sites might be overloaded whilst others are idle, despite each site being tuned to maximize its individual performance. Thus, processing power is being lost due to the imbalance in the load. This means that an enterprise Grid should be able to *self-configure* itself to distribute the load across sites in a balanced way. This load-balancing should be performed on a continuous basis and in a non-intrusive fashion. There has been

substantial work in providing load-balancing solutions for Grid environments. However, although load-balancing techniques and policies for different kinds of applications and servers share a common background, each server and/or application has typically very specific characteristics which lead to individual load-balancing requirements. Thus, while current Grid systems offer load-balancing mechanisms that are well suited for parallel and scientific applications they do not serve well transactional stateful applications with their many constraints in regard to data and execution dependencies. In particular, also the migration of already running jobs is inherently more complex due to the state and execution dependencies.

Another interesting challenge arises if the computing power requirements of an enterprise application change over time as this requires some form of *self-provisioning*. If more processing power is needed, more resources need to be added to be able to cope with the load. This should happen autonomously. Thus, a pool of free sites needs to be available. When load submitted increases and the system feels that the given processing power will soon be unable to handle this load, the system should self-provision itself and increase the amount of devoted resources. Conversely, if the load is reduced the resources should shrink to free unneeded resources. However, self-provisioning in enterprise Grids is very challenging due to their stateful nature. Adding a new replica means that state should be first transferred to it. This means that online recovery is needed to enable self-provisioning as it was needed for attaining the self-healing property. Grid systems have achieved significant progress in self-provisioning. This infrastructure can be generic enough to be reused in the context of Enterprise Grids, although support for dealing with stateful applications will have to be integrated. However, the recovery solutions for self-healing purposes can be reused in this context.

Finally, one has to realize that autonomic management in an Enterprise Grid cannot just rely on a component or server-based approach. Instead, autonomic management in Enterprise Grids needs to adopt a holistic approach. The Enterprise Grid Architecture (EGA) suggests having an entity in charge of management, denoted as

Grid Management Entity (GME). This GME is a recursive component. There is a GME at the global Enterprise Grid level. Then, each subsystem within the Enterprise Grid has its own GME that interacts with the higher level GME and the lower level GMEs corresponding to the contained subsystems. For instance, in a multi-tier architecture, there will be a GME for the full system, and then a GME for each gridified tier, e.g., web tier, application tier, and database tier. There will also be a GME within each instance of the server in a tier. That is, if there are n servers in the application server tier, each of them will have its own GME for local autonomic management. In this area, the gap with what is currently provided by Grids is significant, since Grids typically deal with independent servers or at least deal with them as if they were independent.

Gap 6: Support for Service Level Agreements An important class of enterprise applications provides service provisioning governed by service level agreements (SLAs). SLAs set a contract between client and provider regarding the offered service, quality of service, cost vs. service tradeoffs, tracking of the quality service offered, etc. One of the main concerns is that by sharing resources across different clients, the demand from one client can lead to violations of the SLA of another client. This problem can be addressed by resource virtualization. The idea is to split the resources of a site into multiple isolated execution environments that can be assigned to different clients. The performance isolation achieved by virtualization enables to adopt policies to maximize the fulfillment of SLAs. Grid computing has produced many results regarding resource virtualization that can be exploited to attain Enterprise Grids. However, there is still an important gap regarding how to handle SLAs and the mapping of resource virtualization to clients, especially when the resources are spread across different tiers in a multi-tier architecture.

3 Filling the Gap: Recent Advances in Distributed Systems

Although there are many open challenges and problems that need to be addressed before

Enterprise Grids become a mainstream infrastructure, recent developments in several areas have proposed promising solutions that can possibly be applied in the context of Enterprise Grids. In here, we outline some of them.

Database Replication Database replication [2, 7, 13, 25, 30, 38, 48–50, 52, 55, 59, 68] is important to support the gridification of stateful and transactional online applications and the database tier itself. Database replication has been an active research topic for many years. Especially in recent years, many new approaches have been proposed. What is important is that many of these approaches have response time as an important performance metrics which is needed to allow for gridification of online applications.

Some solutions looked at lazy replication where update transactions are executed only at one replica and the resulting updates are propagated only after commit to the other replicas. For instance, in [7], consistency is provided by assigning each object a master and only the master can accept and propagate updates; furthermore assignment of objects to masters and the propagation topology is restricted to guarantee correct executions. A different research line aimed at making the lack of consistency explicit to the user, providing different degrees of data freshness to the user [48, 49, 59]. Freshness measures the amount of updates that might have been missed by transactions and therefore quantifies an upper bound of the staleness of the data read by a transaction. Another set of lazy approaches, such as IceCube [35] and its clustered extension [40], has been devoted to reconciliation in optimistic replication. Replicas are allowed to proceed in parallel without any measure to prevent conflicts. Later, conflicts due to concurrent accesses are fixed by means of semantic corrective actions.

A different line of research looked at scalable solutions for eager data replication, where updates are propagated before commit, hence providing much better consistency guarantees. Most of these approaches look at solutions providing 1CS. Many of them, such as [50, 52], build on group communication to guarantee consistency in advent of any failure scenario. They additionally exploit asymmetric update processing [28] in

which update transactions are only fully processed at one replica (typically, different transactions are fully processed at different replicas), while other replicas only apply the updated tuples. This asymmetric processing is essential to attain scalability for update workloads [28]. Other eager approaches are based on schedulers [2] that provided the necessary message ordering and consistency guarantees. All these approaches attain some reasonable scalability up to several tens of nodes while providing full consistency.

Both lazy and eager replication can find their place in enterprise Grids since they both allow to take advantage of additional resources and distribute load among them.

An important issue that has also been looked at is how to architect database replication. Three different approaches have been identified: white box, black box, and gray box. The white box approach consists in implementing replication within the database. The advantage is that it can be implemented efficiently however it requires access to the database code and the replication logic becomes interwoven with the database logic. On the other extreme of the spectrum, one finds the black box approaches that do not modify the database [1]. However, it has the shortcoming that transactions need to be executed serially what hampers the performance of the database. Finally, gray box approaches [50] advocate for implementing some minimal functionality within the database that enables efficient and scalable replication at the middleware level. Recently, a number of reflective approaches [44, 60] have been studied to expose some database functionality to enable performant replication at the middleware level. In an enterprise Grid, where Grid technology needs to control to some degree provisioning and distribution, a middleware-based replication approach is likely to be more advantageous because it allows for a better interaction between the replica control software and the Grid environment. A step in this direction could be the concept of data farming in which different databases are replicated and there is a farm of database servers serving requests for all the applications [56].

Clustering in Multi-tier Architectures The research in clustering of multi-tier architectures will

be useful to support gridification of multi-tier architectures and the stateful and transactional applications that are typical in these architectures.

The replication of the web server tier has been studied very deeply during the last decade [8]. The research has basically set the ground work for virtualizing IP addresses in different ways, typically through a router. The result is a virtual IP address that can be served by a pool of servers with different physical IP addresses. When moving web clusters to a Grid environment, additional problems have to be solved. Currently, most routers guarantee that a request is sent to a working server, but there are no guarantees about serving the request, since the server can crash before completing request execution. What is lacking is a solution that provides end-to-end exactly-once semantics to enable a transparent fault-tolerant and highly available interaction. Furthermore, in order to integrate the self-optimizing and self-provisioning features of the Grid system into IP virtualization, it should be possible to change the set of sites serving a virtual IP address dynamically.

A significant body of research has looked into data caching for web servers covering issues such as cache consistency combined with caching of dynamic data. However, distributed and replicated data caching is still an open problem. One of the main difficulties is how to accomplish an independent gridified data caching tier (not recognized as a tier in J2EE), such that its performance and resilience can be increased whilst still providing full data coherence.

In the application server tier, most work has focused on providing availability via primary-backup replication or active replication. Early work started with CORBA (e.g. Eternal [43]) ignoring the interactions with other tiers. The integration of a replicated application server with other, non-replicated tiers, and in particular, the database tier has been first discussed for CORBA and .NET [4, 16, 69] in general settings [21]. More recently, research on J2EE application servers has yielded several promising results in the context of the European Adapt project [5]. For instance, [68] discusses integration of replication of the application server with advanced execution and transaction patterns. Some recent results show that it is possible to provide highly available

transactions in which, despite failovers, transactions are not aborted from the client perspective [54]. However, research in this area is still at an early stage in which only solutions for fault tolerance (primary-backup) have been provided. More work is needed which takes self-optimization and self-provisioning into account.

Furthermore, little work has been done when looking at gridification, or at least replication, across several tiers. The X-ability framework [20] allows to reason about correctness in a general multi-tier architecture. [32] looks at different integration approaches when several tiers are replicated. Integration can basically be achieved in two different ways. One is that each tier is replicated independently (see Fig. 1a). That is, the application server (AS) tier is replicated independently of the database tier, also known as horizontal replication. The second approach consists of taking a pair of application and database servers and replicating this pair (see Fig. 1b). This approach is known as vertical replication [54].

Autonomic Computing The sheer size and complexity of modern information systems often overwhelms even the most skilled engineers that are supposed to configure, optimize, secure, and repair them. In fact, if the current trend of complexity growth continues, the administration of the next generation systems will be beyond human capabilities. Even if the complexity does not increase, it is predicted that the number of deployed systems in the near future will be substantially higher than the number of qualified engineers to manage them [34]. In fact, these studies have resulted in launching research initiatives in most large software companies to create a new generation of systems able to self-manage themselves

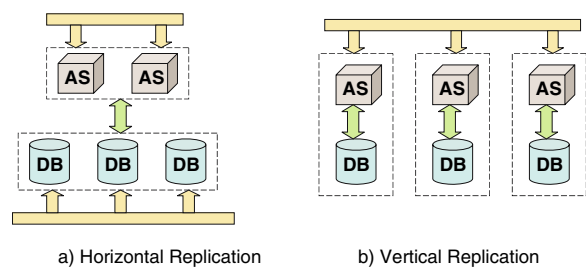


Fig. 1 Replication approaches

with no human intervention except for setting high level goals or deciding between performance and/or costs tradeoffs. Examples of these initiatives are, among others, IBM autonomic computing, HP Adaptive Enterprise, Microsoft Dynamic Systems, and Intel Proactive Computing. The vision is to attain systems able to manage themselves, that is, systems able to self-heal, self-provision, self-optimize, and self-configure themselves. This vision for self-management and self-adaptation is also part of the Enterprise Grid vision [14].

As mentioned before, self-healing requires recovering failed replicas or introducing fresh ones. This will be crucial in order to allow for dynamic resource provisioning and adjustments in the load-distribution when either the user demands or the capacity of the resources changes. In a stateful context, this requires to transmit the state of working replicas to these new replicas so they can join a Grid of servers. Since such recovery might take considerable time it is impossible to stop execution during this process since this would violate the initial goal of high availability. Recent work has looked into online recovery for self-healing databases [27, 31]. Kemme et al. [31] proposed different ways to perform online recovery within the database kernel; Jiménez-Peris et al. [27] studies how to implement online recovery at the middleware level providing toggles for a higher level self-configuration entity to adapt the resources devoted to recovery to the current load. It has also been recently researched how to perform recovery in a context in which the data centers from different collaborating organizations (e.g. branches under a common hotel chain name) are interconnected to perform data warehousing. Some redundancy is introduced in the system that enables to perform recovery with a service interface [36].

The potential scalability provided by new replication approaches can only be attained by resorting to autonomic protocols that self-optimize locally and self-configure globally to maximize throughput. In the database arena, a significant effort has been made during the last decade to develop self-tuning and self-optimization protocols, that act locally on a particular server. One of the subsystems in a database system is memory management. When the server overloads, the sys-

tem enters into thrashing without grateful degradation. Several self-management techniques have been proposed for database memory management in [66] such as access pattern aware adaptive caching or speculative, self-adapting pre-fetching policies.

Another important area for self-optimization in databases has been adaptive load control. It aims at choosing the optimal configuration for a changing system load. It is also important in the Grid context since we have to assume that the capacity of Grid resources might vary depending on what other tasks they take on. One of the most important parameters that affect contention is the multiprogramming level (MPL), i.e., the number of transactions effectively executed concurrently. For instance, if the system has twenty outstanding transactions, it can decide to execute them one by one, two by two, or all twenty concurrently. Some approaches have looked at contention occasioned by locking. [42] determines the optimal degree of conflict and adjusts the MPL to maintain this optimal degree. [23, 61] apply feedback control to adjust the MPL to maximize the throughput of a dataset. [41] proposes a hierarchical self-configuration/optimization for clustered databases where load-control is performed individually at each replica and load-balancing is performed globally across all replicas.

Another important requirement of Enterprise Grids is the self-provisioning capacity. In the area of workflows, there has been some interesting work, in which replicas of the workflow server are allocated dynamically depending on the system load [51]. In the context of database replication, feedback-based approaches and machine learning techniques have been used [9, 62] to dynamically determine the set of active database replicas for database applications with changing workload patterns.

4 Challenges Ahead

While the last section showed individual accomplishments that have a great potential to be transferred to Enterprise Grids, more work needs to be done to develop a holistic solution. In this section, we shortly outline some of the issues.

Boosting the Scalability of Data Grids Most of the existing data replication strategies assume full replication (i.e., replicating the full database at each site). However, such approach has a scalability ceiling that might be not enough for future very large data centers. Partial replication just replicates a subset of data at each site and can be a solution for the scalability ([12]). However, there are still many issues to resolve in this context. One of the interesting research problems is whether a combination of full and partial replication can yield a better scalability. Another important problem is that solutions relying on pure partial replication will likely need to resort to distributed atomic commitment that is the current bottleneck of distributed information systems. Recent protocols for high performance distributed atomic commitment such as the commit server [29] can be a solution to leverage partial replication to boost the scalability of database replication.

The use of group communication might hamper the scalability. More concretely, one option lies in exploiting the use of total order multicast. Optimistic delivery, a more aggressive version of optimistic multicast [53], has been proposed to mask latency of multicast by exploiting the spontaneous total order that happens in LANs [3, 33, 58]. [27, 33, 50] have exploited successfully optimistic delivery combined with a reordering technique to reduce the latency of data replication and minimize the number of aborts when the optimistic delivery was wrong.

Low-latency Gridification across WANs A requirement of Enterprise Grids is the interconnection of data centers across WANs. This will need support for data replication across WANs. WANs differ from LANs mainly in the latency of the communication. This higher latency becomes especially troublesome for group communication based replication approaches in which the required ordering and reliability guarantees might require several rounds of messages. However, purely lazy replication strategies might not be acceptable for some of the transactional, update-intensive data. Thus, new replication protocols are needed with higher levels of scalability and lower latencies while at the same time achieving acceptable levels of consistency.

Some initial steps are being taken towards data replication in WANs [1, 22, 39] in which the use of group communication across WANs is either minimized or just dismissed.

Reducing the Overhead of Gridification Another crucial aspect for the successful gridification of enterprise applications is the reduction of the inherent overhead of the coordination among sites. Modern system area networks provide functionality that might be exploited to reduce this overhead by delegating functions, possibly dynamically, to the underlying hardware. Some researchers have successfully exploited modern system area networks in the context of a distributed shared memory middleware [6].

Holistic Approach to Virtualization and SLAs Despite the advances in virtualization of the different kinds of resources (storage, networking, server tier etc.), new research is needed to adopt a holistic approach to virtualization, in which the computational requirements of an application across various tiers are automatically analyzed. From there, the necessary resources are devoted for deploying and running the new application and adjusted autonomically to the changes in the workload and priorities extracted from SLAs. There have been some early approaches to SLAs for commercial Grids, such as [37].

Scalable and Consistent Gridification of Multi-tier Architectures The current state of the art has addressed solutions for some of the gridification issues of individual tiers. Much more work has to be done, both in regard to gridification of individual tiers, and across the entire multi-tier architecture, taking into account any form of data and execution dependencies.

Systemic Autonomic Computing Most current approaches for autonomic computing focus on individual servers. New approaches addressing the hierarchical self-management of multi-tier systems or complex systems as a whole are needed. Recent advances such as [57], in which performance bottlenecks of complex systems deployed in multi-tier architectures are automatically identified, are a start in this direction.

Accurate Models for Self-optimization and Self-provisioning Despite the progress attained in self-* properties, the attained results do not seem to be enough for the Enterprise Grid vision. One of the aspects that need further research is the modeling of systems with higher accuracy and autonomic protocols with higher guarantees than current best effort approaches. Economic-based approaches seem to be a promising path to follow that provides mathematical models for resource sharing [17].

5 Conclusions

In this paper we have identified some of the shortfalls of current Grid technology when it comes to enterprise applications. The problems are mainly due to the interactive nature of business applications, the large amount of data that resides in database systems and requires transactional access, and the component-based and multi-tier architecture of current enterprise applications. However, we believe that many recent research advances in the area of data management and distributed systems can be applied and incorporated in a Grid infrastructure, to get a step closer to what OGF envisions as the Enterprise Grid. Replication, caching, online reconfiguration techniques, farming, and load-balancing mechanisms that have been proposed, seem promising candidates. However, we believe that adjustments to proposed technology need to be made to make them work in a Grid environment. This paper aims at fostering cross-fertilization between the Grid and the distributed systems communities to make this happen. We also think that there are some outstanding issues that still need further fundamental research.

References

1. Amir, Y., Danilov, C., Miskin-Amir, M., Stanton, J., Tutu, C.: On the performance of consistent wide-area database replication. Technical Report CNDS-2003-3, John Hopkins University (2003)
2. Amza, C., Cox, A.L., Zwaenepoel, W.: Distributed versioning: consistent replication for scaling backend databases of dynamic content web sites. In: *Int. Middleware Conf.* (2003)
3. Balakrishnan, M., Birman, K.: PLATO: Predictive latency-aware total ordering. In: *Proc. of the Int. Symp. on Reliable Distributed Systems (SRDS)* (2006)
4. Barga, R., Lomet, D., Weikum, G.: Recovery guarantees for general multi-tier applications. In: *Int. Conf. on Data Engineering (ICDE)* (2002)
5. Bartoli, A., Jiménez-Peris, R., Kemme, B., and all: Adapt: towards autonomic web services. In: *Distributed Systems Online* (2005)
6. Bilas, A., Iftode, L., Singh, J.P.: Evaluation of hardware support for shared virtual memory clusters. In: *Proc. of the 12th ACM International Conference on Supercomputing (ICS98)* (1998)
7. Breitbart, Y., Korth, H.F.: Replication and consistency: being lazy helps sometimes. In: *ACM Int. Conf. on Principles of Database Systems (PODS)* (1997)
8. Cardellini, V., Casalicchio, E., Colajanni, M., Yu, P.S.: The state of the art in locally distributed Web-server systems. *ACM Comput. Surv.* **34**(2), 263–311 (2002)
9. Chen, J., Soundararajan, G., Amza, C.: Autonomic provisioning of backend databases in dynamic content web servers. In: *Int. Conf. on Autonomic Computing (ICAC)* (2006)
10. Council, T.P.P.: TPC Benchmark C (2006)
11. Council, T.P.P.: TPC Benchmark W (2006)
12. de Sousa, A.L.P.F., Oliveira, R.C., Moura, F., Pedone, F.: Partial replication in the database state machine. In: *IEEE Int. Symposium on Network Computing and Applications* (2001)
13. Elnikety, S., Zwaenepoel, W., Pedone, F.: Database replication using generalized snapshot isolation. In: *IEEE Int. Symp. on Reliable Distributed Systems (SRDS)* (2005)
14. Enterprise Grid Alliance: EGA Reference Model (2005)
15. Ferreira, L., Easton, J., Kra, D., et al.: *Patterns: Emerging Patterns for Enterprise Grids*. IBM RedBooks (2006)
16. Felber, P., Narasimhan, P.: Reconciling replication and transactions for the end-to-end reliability of CORBA applications. In: *DOA* (2002)
17. Ferguson, D.F., Nikolau, C., Sairamesh, J., Yemini, Y.: Economic models for allocating resources in computer systems. In: Clearwater, S.H. (ed.) *Market-based Control: A Paradigm for Distributed Resource Allocation*, pp. 156–183. World Scientific Publishing Co. Inc. (1996)
18. Foster, I.T.: The anatomy of the Grid. In: *CCGRID* (2001)
19. Foster, I.T., Kesselman, C., Nick, J., Tuecke, S.: The physiology of the Grid. In: *Global Grid Forum* (2002)
20. Frølund, S., Guerraoui, R.: X-ability: a theory of replication. In: *Symp. on Principles of Distributed Computing (PODC)* (2000)
21. Frølund, S., Guerraoui, R.: e-Transactions: end-to-end reliability for three-tier architectures. *IEEE Trans. Softw. Eng.* **28**(4), 378–395 (2002)
22. Grov, J., Soares, L., Correia Jr., A., Pereira, J., Oliveira, R., Pedone, F.: A pragmatic protocol for database replication in interconnected clusters. In: *Proc. of the 12th IEEE Int. Symp. Pacific Rim Dependable Computing (PRDC)*. Riverside, CA (2006)

23. Heiss, H., Wagner, R.: Adaptive load control in transaction processing systems. In: Proc. of 17th Very Large Data Bases Conf. (VLDB) (1991)
24. Foster, I., Kesselman, C.: The Grid. MKP, Budapest (1998)
25. Irún-Briz, L., Decker, H., de Juan-Marín, R., Castro-Company, F., Armendáriz-Íñigo, J.E., Muñoz-Escóí, F.D.: MADIS: a slim middleware for database replication. In: Euro-Par, pp. 349–359 (2005)
26. Jajodia, S., Kerschberg, L. (eds.): Advanced Transaction Models and Architectures. Kluwer, Dordrecht (1997)
27. Jiménez-Peris, R., Patiño-Martínez, M., Alonso, G.: Non-intrusive, parallel recovery of replicated data. In: IEEE Symp. on Reliable Distributed Systems (SRDS) (2002)
28. Jiménez-Peris, R., Patiño-Martínez, M., Alonso, G., Kemme, B.: Are quorums an alternative for data replication. *ACM Trans. Database Syst.* **28**(3), 257–294 (2003)
29. Jiménez-Peris, R., Patiño-Martínez, M., Alonso, G., Arevalo, S.: A low-latency non-blocking atomic commitment. In: Int. Conf. on Distributed Computing (DISC) (2001)
30. Kemme, B., Alonso, G.: Postgres-R, a new way to implement database replication. In: Int. Conf. on Very Large Data Bases (VLDB) (2000)
31. Kemme, B., Bartoli, A., Babaoglu, O.: Online reconfiguration in replicated databases based on group communication. In: Int. Conf. on Dependable Systems and Networks (DSN) (2001)
32. Kemme, B., Jiménez-Peris, R., Patiño-Martínez, M., Salas, J.: Exactly once interaction in a multi-tier architecture. In: VLDB Workshop on Design, implementation, and deployment of database replication (2005)
33. Kemme, B., Pedone, F., Alonso, G., Schiper, A., Wiesmann, M.: Using optimistic atomic broadcast in transaction processing systems. *IEEE Trans. Knowl. Data Eng.* **15**(4), 1018–1032 (2003)
34. Kephart, J., Chess, D.: The vision of autonomic computing. *IEEE Comput.* **36**(1), 41–50 (2003)
35. Kermarrec, A.-M., Rowstron, A.I.T., Shapiro, M., Druschel, P.: The IceCube approach to the reconciliation of divergent replicas. In: ACM Int. Conf. on Principles of Distributed Computing (PODC) (2001)
36. Lau, E., Madden, S.: An integrated approach to recovery and high availability in an updatable, distributed data warehouse. In: Proc. of the 32nd Int. Conf. on Very Large Data Bases (VLDB) (2006)
37. Leff, A., Rayfield, J.T., Dias, D.M.: Service-level agreements and commercial Grids. *IEEE Internet Computing* **7**(4), 44–50 (2003)
38. Lin, Y., Kemme, B., Patiño-Martínez, M., Jiménez-Peris, R.: Middleware based Data replication providing snapshot isolation. In: ACM Int. Conf. on Management of Data (SIGMOD) (2005)
39. Lin, Y., Kemme, B., Patiño-Martínez, M., Jiménez-Peris, R.: Consistent data replication: is it feasible in WANs? In: Euro-Par (2005)
40. Martins, V., Pacitti, E., Valduriez, P.: A dynamic distributed algorithm for semantic reconciliation. In: 6th Workshop on Distributed Data and Structures (WDAS) (2006)
41. Milan, J., Jiménez-Peris, R., Patiño-Martínez, M., Kemme, B.: Adaptive middleware for data replication. In: Int. Middleware Conf. (Middleware) (2004)
42. Moenkeberg, A., Weikum, G.: Performance evaluation of an adaptive and robust load control method for the avoidance of data contention trashing. In: Int. Conf. on Very Large Data Bases (VLDB) (1992)
43. Moser, L.E., Melliar-Smith, P.M., Narasimhan, P., Tewksbury, L., Kalogeraki, V.: The eternal system: an architecture for enterprise applications. In: Int. on Enterprise Computing Conf. (EDOC) (1999)
44. Muñoz-Escóí, F.D., Pla-Civera, J., Ruiz-Fuertes, M.I., Irún-Briz, L., Decker, H., Armendáriz-Íñigo, J.E., de Mendivil, J.R.G.: Managing transaction conflicts in middleware-based database replication architectures. In: IEEE Int. Symp. On Reliable Distributed Systems (SRDS) (2006)
45. OGSA-DAI: <http://www.ogsadai.org.uk/>.
46. Open Grid Forum: <http://www.ogf.org/>.
47. Oracle: Grid Computing with Oracle. White Paper (2005)
48. Pacitti, E., Simon, E.: Update propagation strategies to improve freshness in lazy master replicated databases. *VLDB Journal* **8**(3,4), 305–318 (2000)
49. Pape, C.L., Gañarski, S., Valduriez, P.: Refresco: Improving query performance through freshness control in a database cluster. In: CoopIS (2004)
50. Patiño-Martínez, M., Jiménez-Peris, R., Kemme, B., Alonso, G.: Middle-R: Consistent database replication at the middleware level. *ACM Trans. Comput. Syst. (TOCS)* **23**(4), 275–423 (2005)
51. Pautasso, C., Heinis, T., Alonso, G.: Autonomic execution of web service compositions. In: Int. Conf. on Web Services (ICWS) (2005)
52. Pedone, F., Guerraoui, R., Schiper, A.: The database state machine approach. *Distributed and Parallel Databases* **14**(1), 71–98 (2003)
53. Pedone, F., Schiper, A.: Optimistic atomic broadcast. In: Kutten, S. (ed.) Proc. of 12th Distributed Computing Conference (DISC), Vol. LNCS 1499, pp. 318–332 (1998)
54. Perez, F., Vuckovic, J., Patiño-Martínez, M., Jiménez-Peris, R.: Highly available long running transactions and activities for J2EE applications. In: IEEE Int. Conf. on Distributed Computing Systems (ICDCS) (2006)
55. Plattner, C., Alonso, G.: Ganymed: Scalable replication for transactional web applications. In: Proc. of the ACM/IFIP/USENIX Int. Middleware Conf. (2004)
56. Plattner, C., Alonso, G., Ozsu, T.: DBFarm: A scalable cluster for multiple databases. In: Proc. of the ACM/IFIP/USENIX Int. Middleware Conf. (2006)
57. Robertson, P., Williams, B.: Automatic recovery from software failure. *Commun. ACM* **49**, 41–47 (2006)
58. Rodrigues, L., Mocito, J., Carvalho, N.: From spontaneous total order to uniform total order: Different degrees of optimistic delivery. In: Proc. of the ACM Symposium on Applied Computing (SAC), pp. 723–727 (2006)

59. Röhm, U., Böhm, K., Schek, H.-J., Schuldt, H.: FAS - A freshness-sensitive coordination middleware for a cluster of OLAP components. In: *Int. Conf. on Very Large Data Bases (VLDB)* (2002)
60. Salas, J., Jiménez-Peris, R., Patiño-Martínez, M., Kemme, B.: Lightweight reflection for middleware based database replication. In: *IEEE Symp. on Reliable Distributed Systems (SRDS)* (2006)
61. Schroeder, B., Harchol-Balter, M., Iyengar, A., Nahum, E.M., Wierman, A.: How to determine a good multi-programming level for external scheduling. In: *Int. Conf. on Data Engineering (ICDE)* (2006)
62. Soundararajan, G., Amza, C., Goel, A.: Database replication policies for dynamic content applications. In: *ACM SIGOPS EuroSys* (2006)
63. Standard Performance Evaluation Corporation: SPEC-jAppServer2004 version 1.03. Standard Performance Evaluation Corporation (2006)
64. Sun Microsystems: ECperf specification v1.1 final release. Sun Microsystems (2003a)
65. Sun Microsystems: Java 2 Platform Enterprise Edition v1.4. Sun Microsystems (2003b)
66. Weikum, G., Christian, A., Kraiss, A., Sinnwell, M.: Towards self-tuning memory management for data servers. *IEEE Data Eng. Bull.* **22**(2), 3–11 (1999)
67. WS-CAF: Web services composite application framework (WS-CAF). OASIS (2005)
68. Wu, S., Kemme, B.: Postgres-R(SI): Combining replica control with concurrency control based on snapshot isolation. In: *IEEE Int. Conf. on Data Engineering (ICDE)* (2005)
69. Zhao, W., Moser, L.E., Melliar-Smith, P.M.: Unification of replication and transaction processing in three-tier architectures. In: *IEEE Int. Conf. on Distributed Computing Systems (ICDCS)*, pp. 290–300 (2002)