UNIVERSITY OF WESTMINSTER

WestminsterResearch

http://www.wmin.ac.uk/westminsterresearch

Dependability in hybrid grid systems: a virtual clusters approach.

Stavros Isaiadis
Vladimir Getov

Harrow School of Computer Science

Dependability in Hybrid Grid Systems: A Virtual Clusters Approach[*]

Stavros Isaiadis and Vladimir Getov
Harrow School of Computer Science, University of Westminster
{S.Isaiadis, V.S.Getov}@Westminster.ac.uk

Abstract

With the rapid evolution of mobile and ubiquitous computing, small-scale devices like personal digital assistants, smart "converged" phones and laptops now dominate the market. Despite the parallel emergence of the grid as the new distributed computing infrastructure, an integrated hybrid grid system that consists of both traditional fixed nodes but mobile and limited devices as well, has only recently gained some popularity. One of the main challenges in realizing this integration is the very low levels of reliability and availability of small scale and mobile devices –a characteristic that could compromise the overall dependability and performance of the whole grid system. In this paper we try to identify the main dependability requirements for such a hybrid grid system. We then present our own approach to an efficient integration using our virtual clusters platform, and explain how we manage to meet the dependability requirements as set by our research.

1. Introduction

The grid [1, 2, 3] has evolved rapidly over the last few years, and despite still being at infancy stage there have been numerous applications and success stories mainly from the e-Science and high performance and distributed computing communities. The concept of the grid has been embraced widely and heavy research and funding is currently made available to overcome the many research challenges towards future generation grids. The goal is to make future grid systems truly ubiquitous and pervasive, and we have already seen that the grid concept is expanding rapidly to application domains that were not considered suitable initially. One such domain is mobile computing.

The idea of integrating mobile devices such as laptops, smart phones, and wearable computers into grid systems has slowly been introduced in some research efforts. Such integration can yield great benefits for the mobile community, since they can still have the advantages of mobility and ubiquity, but they can also now reap the benefits of grid computing, acquiring access to distributed resources such as very powerful hardware, specialized software, large databases and a wide range of services available within a grid system.

This integration can be beneficial for the grid as well. Mobile devices increasingly offer functionality not found in traditional grid nodes, closely associated with mobility, location and context awareness, like global positioning systems, intelligent wireless sensors, multimedia equipment, sensory data collection and more. In addition, they could cover geographical areas where traditional wired grids are not able to reach, thus extending the deployment area of the distributed system. And that is exactly where the core advantage of this integration is found and not in providing yet another pool of traditional resources for high performance computing –although coordinated access to the aggregated resources could yield significant results as our research shows.

Applications that can benefit from this integrated hybrid wired/wireless model are those that exploit the natural characteristics of mobile devices –mobility and ubiquity, such as context-based information services, data collection services, collaborative applications requiring the distribution of computational load or data across different nodes.

However, such integration presents many challenges. A very important one is the very different levels of reliability and availability between the traditional nodes that usually comprise a grid system and the mobile/limited devices to be integrated. The former, are relatively static in nature, reliable and predictable. The latter, are very unreliable due to a wider range of failures they are susceptible at, unreliable communication links and unpredictable mobility patterns. Inherent limitations like increased battery consumption sensitivity, and location sensitive connectivity make things even worse.

By realizing such integration, the resulting hybrid grid system will have two separated domains that vary significantly in terms of dynamicity, reliability and robustness. If we directly expose these unwanted characteristics to an existing grid system, which has converged around certain acceptable levels, the overall availability and reliability may drop dramatically. Hence, looking at the integration challenge from the grid perspective, we would like to diminish these limitations and manage an efficient and useful integration that will

benefit the grid community without imposing any performance penalties.

In the rest of the paper, in Section 2 we present relevant work in this area while pointing out their differences and/or limitations. In Section 3 we present the dependability requirements in the context of such hybrid systems. In Section 4 we present our architecture for realizing such hybrid environments and discuss how we are improving dependability measures of hybrid systems. Finally, Section 5 concludes the paper and provides the roadmap for future research and development.

2. Relevant Work

Ubiquitous grid computing is gaining popularity lately due to the increased share of the market that mobile and ubiquitous devices hold and there are currently many research efforts dealing with this issue.

In [5] the authors propose a proxy-based clustered approach for integrating mobile devices into the Grid. The proxy will serve as the representative of a group of mobile devices to the Grid. When a request arrives at the proxy –which is called "interlocutor", it will take care of distributing the request in the mobile devices. Introducing an abstraction/proxy layer can provide the foundations for dealing with reliability and availability. The authors do not provide any implementation, although LEECH [4] is based on this research. LEECH is a framework for virtualizing and integrating small-scale devices into the Grid. It makes use of a message passing layer on top of MPI [10] in order to provide intercommunication. It does require, however, the installation of various components, while also imposing a specific programming model and requires the modification of existing services. Our approach follows a purely component/service oriented approach, providing support for contribution of mobile resources in an efficient way without imposing any performance penalties to the existing grid infrastructure – a very important issue that many projects seem to underestimate.

In [9] the authors also propose a proxy based cluster approach and a middleware to provide peer-to-peer operations but do not address any of resource virtualization, federation of resources, or the provision of a relatively reliable environment through some sort of a failure detection and recovery mechanism.

In [6] an agents approach is adopted to tackle device mobility. The authors introduce the term "Hybrid Flexible Cluster" to define a cluster consisting of both mobile and fixed nodes with the ability to adapt to network failures, maintaining a flexible topology. However, this architecture only allows mobile devices to be the consumers of services and not the providers.

In [11] the authors resolve to a proxy based design but in order to provide access to grid resources for mobile devices and not contribution of mobile resources.

3. Dependability Requirements

Dependability is a very important property for a grid system as it guarantees a certain degree of stability, fault tolerance and availability. In the context of a hybrid grid system, and because these consist of mobile and limited devices (mainly), dependability becomes more important and much more difficult to achieve. The increased ratio of failures, either due to operational failures like low battery level, or because of mobility and roaming between different access areas, means that the overall stability of the system can be reduced significantly.

3.1. Availability

Availability is the property of a device to execute a task on demand and is normally used by scheduling systems in order to arrange resource reservation -although advanced resource reservation may be a problem in this case due to the unpredictable nature of the devices.

Scheduling systems must be "protected" from the reduced availability and unpredictability of the mobile resources, in order to keep the response and performance at the same levels even after integrating the devices to the grid. Mechanisms to hide or efficiently manage the low availability of mobile and limited devices should be deployed, and an efficient balance between performance and availability must be determined. The integration should not induce further complexity or place extra load to the system's components.

3.2. Reliability

Further to the availability of mobile devices, the importance of reliability and robustness becomes amplified in the light of a hybrid system. In a distributed system, an application will almost always make use of various resources probably spanning multiple administrative domains, the coordination of which is taken care of by the underlying middleware/platform. It is imperative that the application semantics are respected throughout the whole duration of the execution, the results obtained are correct in every aspect, and above all, that the application will eventually complete gracefully regardless of the state of the system or the failures that may occur along the way. "Gracefully" in this context means either successful completion involving returning the results, or successful termination without side effects –e.g. rolling back and keeping the "only once" semantics in case of database update transactions.

Mobile devices present an unreliable, weak link in the application execution "chain" that can easily be broken due to their aforementioned inherent limitations. In order to maintain a certain degree of reliability for the whole grid system, we must find ways to respond to or compensate for these limitations.

3.3. Transparency, Virtualization and Client Abstraction

In most brokering/scheduling systems or portals that provide access to the grid, the user is presented with available resources and he can select the ones that better suit his application. The same result can also be obtained programmatically, when the user presents his requirements and the middleware/grid system will try to automatically match these to the available resources. Requirements may include time constraints, cost boundaries, the use of specific nodes and so on. In both cases, knowledge for addressing and binding to the available resources is required either by the portal user or the middleware. With the topology map of the mobile domain changing so often, and the availability levels constantly fluctuating, the job of the middleware is not only more difficult, but less reliable as well. The same stands true for service development or service composition now, since the number of available services may be much higher now (due to the large numbers of available mobile devices) but the consistency level would be very low. To overcome these problems, we need to provide application level abstractions, in order to make binding, invocation and composition easier for the higher level users -be it developers, middleware or application components or portals.

4. Virtual Clusters

4.1. Architecture

In the light of the aforementioned challenges, our efforts for integrating mobile devices into complex distributed systems in general, the grid in particular, have focused on the concept of the virtual cluster [7, 8]. The latter allows for the efficient integration of heterogeneous, service/component oriented mobile and resource limited devices into existing distributed systems. In this section we will provide details of the virtual cluster architecture and how it helps enhance reliability and availability of the whole system, as well as provide developer/application level abstractions.

The virtual cluster consists of all mobile/limited devices that fall into the same domain (be it a logical arrangement of the devices or a physical domain such as, for example, the IP sub-domain of the wireless access point in range). We call it a "cluster" even though it is not a cluster in its traditional form, yet it borrows some characteristics from the cluster domain especially in the way it is managed by our platform.

The whole virtual cluster of devices is represented to the grid through a proxy. The grid has no explicit knowledge of the topology of the virtual cluster, the participating devices or their location. In fact, the grid doesn't even know such a cluster exists, since everything is hidden behind the proxy.

In more detail, we create the virtual cluster by aggregating/federating similar resources (where similarity is determined by the interface the respective services or components implement) into a single aggregator/virtual service that is published at the proxy (Figure 1). This will allow us to present a single, consistent and permanent interface point to functionality possibly available in multiple devices, generally unreliable and possibly limited. This abstraction layer we introduce reduces knowledge required by clients if they wish to connect to an underlying service: they now only have to invoke the aggregator service which will then take care of forwarding the request to any available nodes in the virtual cluster that match the requirements.
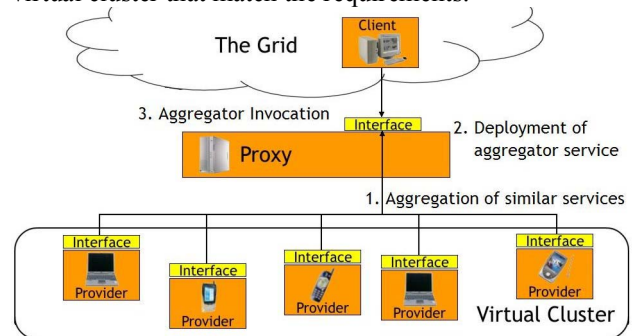

Figure 1: Virtual Clusters Architecture

The core advantage of the integrated virtual cluster is not in utilizing yet more resources –since there should already be plenty of them in the existing system. Instead, it focuses on exploiting the context and location aware applications as well as data acquisition capabilities of mobile devices. However, it is desirable in some cases to provide access to traditional "raw" resources, like CPU, memory, storage etc. One such example might be hot spots in conferences, meetings or seminars where we can exploit the resources in the large number of laptops for ad-hoc high performance computing. Because of the relatively limited resources virtualizing the whole cluster and presenting it as a single resource pool would yield important benefits. In this case it makes sense to leave some spare resources inside the cluster in order to cover up possible failures. This will allow us to continue execution even on the dawn of a number of failures, and not declare the whole virtual entity as failed. Thus, we do not publish the total aggregate of the available resource but only a fraction of it as discussed in the following section under parallel invocation.

When we built the virtual aggregator services we encapsulate more functionality than just the one provided by the underlying services. Among others, we provide support for mirrored execution (for increased reliability and utilization whenever required); parallel execution with distribution of load (for increased performance); and

collective operation on all available nodes (for data collection, statistical operations and more).

In the context of the virtual cluster, there are now two different views of dependability: we need to provide dependability mechanisms internally in the cluster, and also present the whole cluster as a reliable and stable unity. The main objective here is to provide a relatively reliable and stable environment (i.e. the virtual cluster) as seen by the clients, through the aggregator services deployed at the proxy layer. These services must present consistent interface points to the underlying resources, must be highly available at all times and reliable during method invocations –regardless of the state of the aggregated resources.

4.2. Providing a Virtually Stable and Reliable Environment

Virtual clusters have very positive effects in the field of reliability. Failures within the cluster generate notifications that only go up to the proxy layer and not further up the chain to the higher level grid components - which have absolutely no knowledge of the internal state of the cluster. Failures are dealt with internally by means of redistribution of tasks, check-pointing and migration of processes or any other mechanisms the dependability framework that is in place provides –this can be customized and any third party framework can be deployed.

Device Reliability is determined by following standard reliability formula:

$$P_n^{(t)} = 1 - e^{\frac{-t}{Mn}} \qquad ①$$

where Mn is the Mean Time Between Failures for device n, and in the case of battery operated devices it becomes a more dynamic variable –perhaps using a weight like the current battery level.

If we want to incorporate mobility and disconnection failures in this probability, we can use the straightforward and realistic assumption that the closer the device is to the access point, the better the chances it will remain in range. Distance in this case is measured in milliseconds as the end-to-end communication delay between the node and the proxy behind the access point. This is in essence a "virtual" distance as it doesn't depend on the actual distance but on the environmental conditions and obstacles between the two endpoints. In order to measure this distance we can use simple ICMP packets –i.e. ping utility. Then, let S_f denote the network's standard range, the probability that a device n stays in range can be calculated as:

$$P_n^r = \frac{|S_f - S_n|}{S_f}$$

where S_n is the "virtual" distance of the device from the proxy. S_n can further be enhanced by taking into consideration previous measurements so that we try to "predict" the next distance. Assuming we keep track of the last j distances, S_n can be calculated as follows:

$$S_n = \sum_{k=1}^{j} W_k S_n^{j-k}$$

where W_k is a weighting factor to give more importance to recent measurements. If

$$C = \sum_{k=1}^{j} k$$

then W_k can simply be:

$$W_k = \frac{k}{C}$$

System Reliability in the virtual cluster can be categorized according to the execution semantics: it can be either direct (single or part of an application service composition -the semantics are the same in both cases) or parallel invocation (for execution of a job using raw resources like CPU, memory, storage etc). There is also mirrored invocation that can be facilitated in order to provide an N degree of fault tolerance, where N is the number of mirrors selected. This can be used e.g. in collective operations or to implement voting schemes, and has the same reliability factor as the single invocation. System Reliability is represented by the probability $P^{(t)}_{vc}$ that the whole virtual cluster will remain online for a period t and is determined depending on the aforementioned execution methods:

o **Single**: if a failure occurs, the aggregator service will automatically invoke the next available host that provides the same functionality. Hence the probability that the whole system will stay online (and therefore the reliability factor) is:

$$P_{vc}^{(t)} = 1 - \prod_{k=1}^{N} P_k^{(t)}$$

and is of course proportionally increased as N gets higher –pushing the whole system towards virtually 100% dependability values when N is high enough.

o **Mirrored**: For mirrored invocation, reliability remains the same, although efficiency and performance is obviously increased especially in the case of failures. To illustrate this, we run a series of experiments where we measured the total time (computation and communication) from a client that wished to invoke a simple data transfer service in a virtual cluster of 3 laptops. The results in Figure 2 clearly demonstrate the advantages of the mirrored invocation while also showing the overall communication time gains of using a proxy.
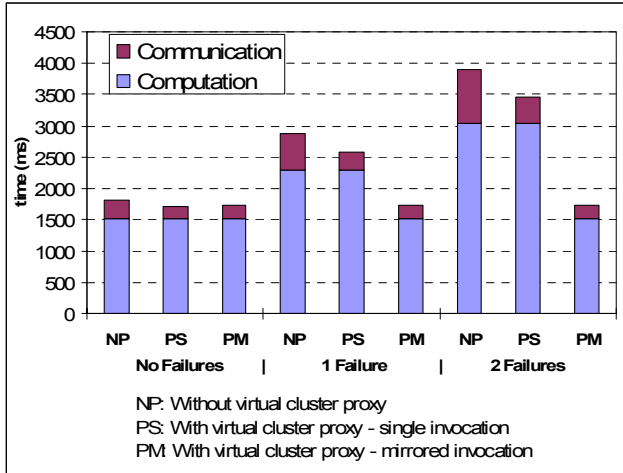
Figure 2: Total time cost for a single service invocation

o   *Parallel*: the failed task is redistributed to one of the spare resources, while the active nodes still work on the remaining distributed tasks. In order to determine a realistic and efficient percentage for the spare resources, we make use of the failure model for the devices: assuming no spare resources at all, the probability $P^{(t)}_{vc}$ for the virtual cluster to remain online within a time period t of an execution is:

$$P^{(t)}_{vc} = \prod_{k=1}^{N}(1 - P^{(t)}_{k})$$

If we allow for x out of N spare resources and assume that all devices have nearly equal failure probabilities $P^{(t)}$, then the probability that the whole system will stay online is the complement to the probability of more than x of the devices to fail:

$$P^{(t)}_{vc} = 1 - \sum_{k=x+1}^{N} C(N,k)(P^{(t)})^{k}(1-P^{(t)})^{N-k}$$

In order to determine an efficient allocation, we will use a simple yet typical example with 10 laptops gathering in a meeting room. We want to integrate the aggregated computational power to the backbone Grid system as an instant high performance facility for the whole duration of the meeting (e.g. 1 hour). Assuming a 4 hour battery life and no recharging, the probability for each laptop to fail within the session according to equation 1, is:

$$P^{(t)}_{n} \cong 0.25$$

Now we can calculate the various reliability values for different allocation of spare resources every time. In the following diagram we show the resulting values.
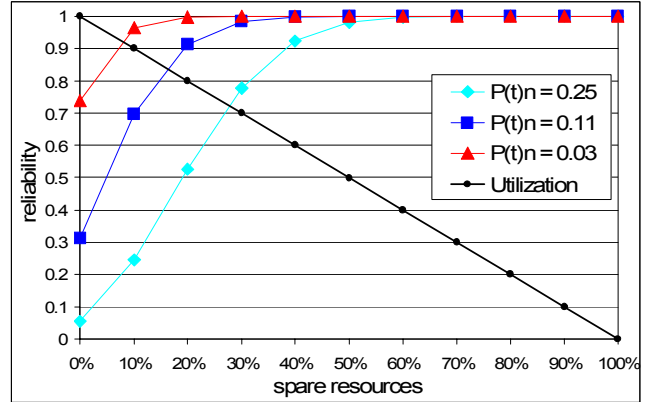

Figure 3: Reliability and Utilization for Different Independent Failure Probabilities

We can see that in our particular example, reliability climbs above 90% when 40% of the resources are used as backup resources. In more realistic scenarios, however, an application session lasts for a few seconds in simple cases, or up to a few minutes in more demanding situations –especially considering that the hybrid system is used mainly for knowledge-based, data acquisition and context aware utility computing and not for high performance computing. Hence, a more realistic failure probability would be in the area of 3% where we can see from the diagram that reliability for the whole system passes the 95% barrier only with 10% spare resources, while for 20% spares it has reached a 99.9% figure. In any case, there is no perfect allocation and it all depends on the respective usage scenario for the hybrid system, the application requirements and the administrator's demands.

4.3. Transparency and Virtualization

A clear benefit of the abstraction proxy layer we have introduced is the possibility to federate under the same aggregator service, services that conform to different frameworks or component models. For example the services in the mobile devices can be represented as Web Services, Grid Services, RMI active objects or any suitable component framework. The client will communicate directly with the relevant aggregator service which will determine at run time the necessary communication protocol needed to connect and invoke the underlying services. In this arrangement, only the proxy has full knowledge of the underlying topology and communication protocols, thus relieving the clients from this need.

Furthermore, it hides the dynamicity and location and binding details from prospective clients. This has benefits in communication costs for the client: consider the following example cases:

o   a distribution company needs to gather information about their field personnel

o   automatic collection of sensory data from multiple geographically dispersed sensors

In both cases and using the virtual clusters approach, instead of N long haul communications the client now needs only one long haul communication with the proxy – which will then deal with the N shorter haul links with the micro devices. Since the proxy is usually deployed right behind the wireless distribution network, the actual gains are in the link between the client and the proxy.
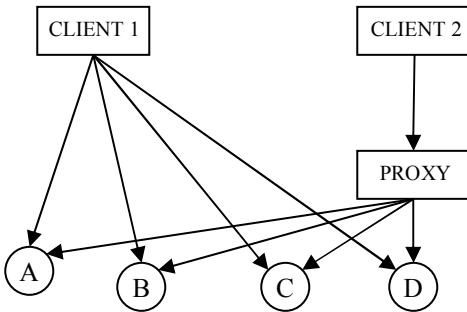

Figure 4: Direct and Proxy Client Communication

Assuming a distance of d, and an overhead of t for each message, the total communication gain for the client is:

$$\Delta C = C - C_p = (N-1)(t+d)$$

where C and C_p are the communication costs with and without proxy respectively for the same distance d. These gains get much bigger considering dense communication, a large number of messages and/or a congested, heavily loaded environment. Furthermore, if you put into the equation the need for authentication handshakes or similar certificate based security measures, the impact can be significant.

5. Conclusion and Future Work

Dependability is a very important property of complex and highly distributed systems like the grid. In the wake of a new ubiquitous era for grid computing with the integration of mobile devices, dependability gains more focus in an effort to smooth out the negative effects that the low reliability and availability of such devices would have to the overall grid system.

In the context of the virtual clusters approach, we identified the most important dependability requirements and presented our techniques for dealing with them and providing a virtually stable, reliable and highly available hybrid grid system. Our approach can lay the foundations for further work on the dependability field in hybrid grid systems, while our experiences from using the implemented platform can help us identify more challenges and further improve it.

References

1. Foster, C. Kesselman and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International Journal of Supercomputer Applications*, vol. 15. pp 200-222, 2001.

2. D. de Roure, M. Baker, N. Jennings and N. Shadbolt, "The Evolution of the Grid." , in F. Berman, G. Fox, T. Hey, Eds., *Grid Computing: Making the Global Infrastructure a Reality*, Wiley & Sons, 2003, pp. 65-100.

3. Foster, C. Kesselman, S. Tuecke and J. Nick, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration," Open Grid Service Infrastructure WG, Global Grid Forum, 2002.

4. N. Ruiz, "A Framework for Integrating Heterogeneous, Small Scale Devices into Computational Grids and Clusters," M.S. Thesis, University of California, US, 2003.

5. T. Phan, L. Huang and C. Dulan, "Challenge: Integrating Mobile Devices into the Computational Grid," in *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking*, 2002, pp 271-278.

6. L. Cheng, A. Wanchoo and I. Marsic, "Hybrid Cluster Computing with Mobile Objects," in Proceedings of the 4th International Conference on High-Performance Computing in the Asia-Pacific Region (HPC-Asia), 2000, pp. 909-914.

7. S. Isaiadis and V. Getov, "Integrating Mobile Devices into the Grid: Design Considerations and Evaluation," in *Proceedings of the 11th International Euro-Par Conference on Parallel Processing*, 2005, pp 1080-1088.

8. S. Isaiadis and V. Getov, "A Lightweight Platform for Integration of Mobile Devices into Pervasive Grids", in *Proceedings of High Performance Computing and Communications (HPCC)*, 2005.

9. J. Hwang and P. Aravamudham, "Middleware Services for Peer-to-Peer Computing in Wireless Grid Networks," *IEEE Internet Computing*, vol. 08, no. 4, pp. 40-46, July/August 2004.

10. Message Passing Interface Forum, "MPI: A Message Passing Interface Standard," 1995, www.mpi-forum.org/docs/mpi-11-html/mpi-report.html.

11. D. E. Millard, A. Woukeu, F. Tao, H. C. Davis, "Experiences with Writing Grid Clients for Mobile devices," School of Electronics and Computer Science, University of Southampton, Southampton, UK, 2005

IEEE
COMPUTER
SOCIETY