

EVALUATION OF DYNAMIC CLUSTERING ARCHITECTURE FOR UTILISING MOBILE RESOURCES

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

ABSTRACT

In recent years we have witnessed the emergence of pervasive and mobile computing. Outdoors field operations, greater team collaboration, and flexibility demands have motivated the nurturing of pervasive solutions that provide ubiquitous access to internal networks and valuable data. However, the evolution of small-scale and mobile devices has only recently brought the idea of contributing mobile resources into the light. With a suitably lightweight service framework and a relevant middleware management platform, the efficient and meaningful exploitation of functionality not found in traditional distributed systems, can open up a whole new range of opportunities for efficient data gathering and dissemination. In this paper we briefly present the design and implementation of our middleware platform for collective and transparent management of mobile and small-scale devices based on the ‘Virtual Cluster’ approach. More importantly we describe and analyze an extended series of experiments that show the many benefits of this approach in exploiting mobile and limited devices for the purposes of mobile and pervasive computing.

KEYWORDS

Virtual Clusters, Hybrid Grids, Mobile Grids

1. Introduction

The Distributed Computing and more specifically the Grid [1] community has traditionally focused on the high performance and distributed computing, and e-Science domains [2]. However, with the rapid emergence of ubiquitous and mobile/nomadic computing, future generation distributed systems need to adapt in order to provide support for this already vast community. There have been efforts lately to provide Grid resources as a backbone infrastructure for limited and mobile devices in order for mobile users to delegate demanding applications/tasks to the backbone, thus not taking up resources or draining battery from their own device. Mobile users can submit their jobs usually through some sort of Grid portals, and they can receive back the results later.

Nevertheless, as such devices are becoming more and more powerful, and considering the numerous mobile-

aware applications/services that they can offer, in this paper we follow a different route. We discuss why and show how, mobile and possibly limited devices could contribute location aware, flexible services to a wider Grid community. Such devices may well contribute traditional resources (CPU, memory, storage etc.) as they become more and more abundant in these devices. Further, they can provide significant aggregated computational power when gathered in hotspots, where Grid systems can be formed on site with the option for “instant” high performance capabilities (for example in conferences, meetings, seminars etc). The benefits of such integration have been discussed in several recent papers [3, 4, 5].

However, we envision a different source of added value for hybrid pervasive distributed systems. Mobile devices can extend the Grid boundaries and usage of hybrid applications to places where traditional wired infrastructure cannot be deployed. For instance, in outdoors field operations, such integrated system can enable much better team cohesion and collaboration, towards better and more efficient data gathering, automated personnel scheduling and allocation, and transparent data dissemination, and aggregation of equipment such as wireless instruments, mobile facilities, satellite imaging services, and more. Hence the core feature that microdevices will provide is not performance related, but mainly information related and utility computing. As such it can add another dimension to the current monolithic and relatively static concept of today’s typical distributed systems.

For such complete integration to be smooth and efficient, a number of challenges have to be considered. These originate from two distinct domains: the inherent limitations of the mobile devices individually, and the challenges associated with the properties of the integrated system as a whole (Section 2). To the best of our knowledge, no current project tries to overcome challenges from both domains. There are efforts dealing with the small scale device limitations but not their mobility, dynamicity, or integration and vice versa. Other projects deal with accessing functionality in small scale devices, but not in a standard way and with too many restrictions (Section 3). The Virtual Clusters (VC) architecture, presented in Section 4, focuses solely on

overcoming all the challenges associated with such integrated environments, in an effort to facilitate standards based access to mobile device functionality; consider efficient communication and monitoring models to handle dynamicity and heterogeneity in the mobile domain; and introduce a complete proxy-based architecture and the accompanying middleware to enable transparent and collective management of the big number of mobile devices. The implemented fully functional prototype has reached mature versions and has been extensively tested over significant time periods. Having presented and described the VC platform in recent papers, we now focus on the results acquired over this period and their analysis (Section 5) to demonstrate the benefits of the VC approach towards a truly pervasive and feature rich integrated environment, able to support field operations, and hybrid applications.

2. Challenges and Requirements

2.1 Mobile Computing Challenges

The inherent resource limitations of small-scale devices present some very challenging issues when considering service hosting. Typical Web servers (such as Apache Tomcat/Axis), Grid middleware (such as Globus Toolkit[6, 7]), application servers all have significant requirements in terms of memory, storage and CPU capabilities, and usually have a number of software and technology dependencies as well. Thus, a major issue has been the development of a suitable service-hosting environment in order to access services and functionality in mobile devices. Some experimental or proprietary efforts do not have a generic applicability, and are unsuitable for a hybrid environment where the backbone system follows a service-based approach.

There are currently a number of efforts focusing on this field, such as the Microservices Framework [8, 9], and the Nokia/Symbian alliance to port Apache's popular httpd daemon and Web-hosting environment into the S60 Nokia mobile phone series (Raccoon [10]).

Mobility may offer many benefits to the end user but it has also introduced a number of relatively new quality attributes, user/session and terminal/service mobility, which when viewed in the context of an integrated hybrid system, are magnified. This extra dynamicity induced in the system makes management difficult, increases utilization of system components dealing with non-functional aspects (fault-tolerance, load balancing), and reduces the stability. Current approaches fail to meet the requirements of modern dynamic and diverse mobile systems because they are built on the assumption of a relatively homogeneous environment. Thus, they cannot opportunistically exploit local resources (e.g. locally available high-resolution screens, high performance facilities), nor handle the diversity of the environment.

Finally, dynamicity can also originate from a wide range of possible failures, due to intermittent connectivity, or hardware failures such as battery drainage. This high rate

of failures in the mobile domain has a direct impact on the application level failure rates and response times.

2.2 Integration Challenges

Even with the development of suitable service hosting environments, handling each device as a single peer is not efficient because of the limited usable resources. A collective management framework would enable leveraging resources from a number of devices to form higher-level services and dynamic resource pools and compensate for the limited individual contribution. Furthermore, an efficient collective/ group abstraction would ease software development and reduce communication overhead for the higher layers.

Reduced dependability presents another argument against treating mobile devices as full member of the system. As more diverse and dynamic nodes join the system, the average reliability and availability decreases and could result in severe performance degradation. From a business perspective, such integration should mean exploitation of more resources without significant administration overhead.

Heterogeneity issues also humble the mobile domain, and at the heart of them all is the very large variety of hardware platforms and operating systems, but most importantly, the lack of interoperability at the application development, and connectivity layers. At these layers many distributed computing technologies may be adopted, for reasons of either interoperability (Web Services -WS) or performance (stripped down RMI, proprietary approaches, plain sockets). However, in a mobile domain, the adoption of a single standards based approach can significantly reduce the set of nodes that can meet the requirements and participate in the system. For example, WS technologies induce a large processing and communication overhead, and even lightweight Web servers cannot be installed in some resource-constrained devices. The variety of resource availability, device models and functionality in the mobile domain, implies that it is impossible for all devices to adhere to a specific approach. In an integrated mobile system, openness and agile computing are amongst the main design goals and thus no unnecessary restrictions should be imposed. The system should be able to seamlessly integrate a wide range of different technologies and protocols.

3. Related Work

The focus of this paper is not on suitable server-side containers for limited devices, but rather on the enabling middleware that will provide the means for an efficient integration of mobile and limited services. We are not aware of research efforts focusing on that area, and driven by the challenges and requirements identified in Section 2. AKOGRIMO (Access to Knowledge through the GRId in a MOBILE world) [11, 12] aims to evaluate the mobile Grid concept through challenging applications such as eHealth, eLearning, and crisis management. It favours research on ubiquitous access to pervasive and mobile Grid services, Authorization, Authentication, and

Accounting, personalization and context awareness, and the integration of Mobile IPv6 to support mobility in dynamic Grids.

WSPeer [26] is a framework for hosting and invoking Web services in an environment agnostic manner. Current WS-RF implementations are based on a generic architecture, which utilizes a service container where all deployed services are statically deployed. Instead, WSPeer enables the use of WS technologies (SOAP, WSDL, WS-RF, WS-Notifications) in a late-bound P2P context without the need for its own runtime environment, and without requiring HTTP as the transport protocol. This P2P orientation allows the use of WS-RF services in more diverse application domains, while also enabling the integration of mobile services into a SOA system.

Both AKOGRIMO and WSPeer, despite enabling mobile services, do not discuss small-scale devices, or any suitable collective management framework. Instead, the developer (and clients) is exposed to the cardinality of the mobile domain and its undesired dynamic characteristics.

Another set of projects deal with exploiting functionality on mobile and microdevices, namely LEECH and K*Grid. LEECH [13] stands for Leveraging Every Existing Computer out tHere and focuses on the aggregation of raw computational resources in a proxy-based cluster of small-scale devices, in order to execute computational jobs in them through the use of an MPI [14] library.

K*Grid [15] aims to study and analyse current wireless mobile networks, and mobile Grid technologies, and use a testbed comprised of PDAs and wireless LANs to develop and evaluate a resource harnessing mobile Grid infrastructure. It is currently at an early stage and no deliverables (and thus evaluation) have been reported yet.

K*Grid and LEECH share the same common drawbacks: first, they only take into consideration raw computing resources, when the core competence of mobile devices clearly is on their context awareness and flexibility. Second, they do not follow a standards based approach; instead they impose certain restrictions on the programmers.

There are a number of related efforts that focus on integrated hybrid mobile Grid environments, but they have significant shortcomings in more than one area. For instance, Hybrid Flexible Clusters [3], MoGrid [16], and the P2P Wireless Grid approach presented in [17], discuss ways for enabling access to the Grid backbone through mobile devices, and not exploiting mobile device functionality. Finally, the generic field of Surrogate Computing (or Cyberforaging) includes a number of efforts [18, 19, 20, 21, 22] that focus on the offloading of applications executing in mobile devices to nearby more powerful nodes (surrogates).

From the above brief discussion, it becomes clear that, realistically and to the best of our knowledge, there are no middleware efforts focusing on managing an efficient integration of small and mobile devices and dealing with the vital issues identified in Section 2, namely service mobility, services run on microdevices, lightweight monitoring, collective management and cardinality

abstraction, enhanced collective operations, and failure masking. The VC approach presented in the rest of this paper is an effort to meet all these requirements in a flexible, extendable, and efficient manner.

4. Architecture

The challenges identified in Section 2, all point to the same direction: directly connecting each mobile device, as a full member of the hybrid Grid system is not efficient because of their unwanted properties and limitations. This has been the driving force behind the design of the VC architecture. It focuses on providing the means to overcome the resource limitations, dynamicity, and mobility, while at the same time enabling an efficient collective approach and monitoring components to care for stable availability. The main conceptual idea is to hide the big cardinality in the mobile domain by exposing the mobile devices as a single virtual entity in a proxy-based, partially centralized design, where the Grid system is merely extended to include a single extra node: the proxy. The latter acts as the gateway for the underlying group of mobile devices, which is now coined a Virtual Cluster. An extensive set of simulations presented in [23] validated this design and verified the efficiency of our approach.

The main entity in the realization of a VC is the aggregator service (or simply aggregator). An aggregator resides at the proxy, and is responsible for a group of mobile devices that provide a similar resource or service (where similarity is determined by the implemented interfaces). Aggregators allow us to present single, and consistent interface points to functionality available in multiple devices, generally unreliable, and possibly resource limited. Each aggregator is generated and deployed automatically, the first time that a particular service registers in a VC. For instance, when a provider contributes an image acquisition service that has not been aggregated before, a new aggregator will be generated to represent this new functionality. If another provider joins the VC and provides the same service, the latter will be integrated in the existing aggregator. Hence, there is one aggregator for each different service or resource type. Fig. 1, depicts the VC architecture and the most important middleware service groups: monitoring for controlling dynamicity, discovery for managing mobility, collective layer for management of large numbers of dynamic nodes, and there is also a number of supporting service groups, including indexing, invocation and more. Details of the aggregation procedure are available in [24].

Following the brief high-level description of the architecture and its most important entities, each of the requirements identified in Section 2 is now mapped to a specific architectural aspect or one or more service group components.

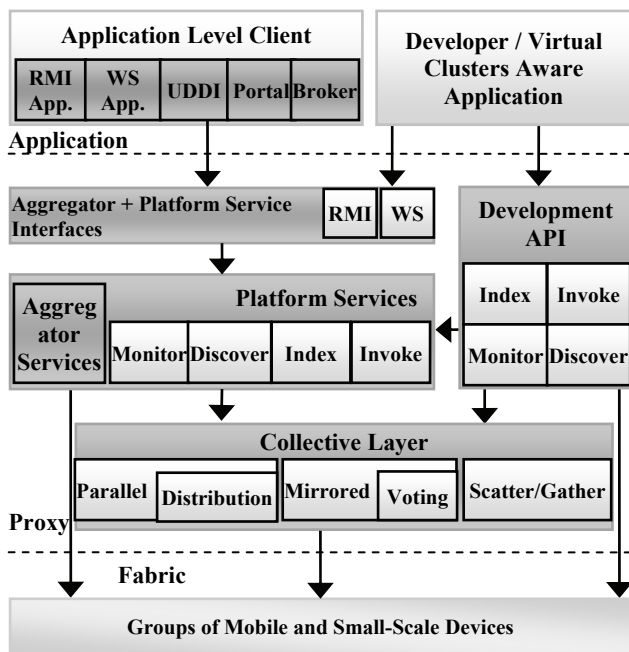


Fig. 1. High level overview of the most important components

• **Dynamicity and Dependability Issues:** The building block for handling failures in the mobile domain is the proxy layer on top of the unreliable fabric layer. This enables the encapsulation of internal failures, protecting this way the rest of the system from the inherent instability of the underlying nodes. Dealing with device failures directly, encompasses two features: failure prediction, and failure detection. Prediction is possible due to the nature of the failures in the mobile domain. These are caused mainly from mobility and battery drainage. The monitoring component monitors the battery levels with a tiny “agent” installed in the device, and tracks mobility facilitating Virtual Distance mechanism [25] to calculate the probability of a device moving out of coverage. Failure detection is accomplished by a lightweight heartbeat mechanism. The absence of the heartbeat triggers a “suspicion” period and denotes that a silent failure may have occurred.

Dealing with the consequences of device failures has a three-fold meaning: fault tolerance -for high reliability, failure masking -for stabilized availability, and failure recovery -to support both. To achieve fault tolerance, the VC takes advantage of the cardinality of the underlying services in the mobile domain, to automatically mirror the execution of a task to more than one available services. For failure masking, failures within the cluster generate notifications that only go up to the proxy layer while they are dealt with internally with mechanisms such as state migration or redistribution. Finally, for failure recovery, and since the nature of failures in the mobile domain is fundamentally different than in traditional static domains, slightly different techniques are required. First of all, a temporary disconnection of the device, due to intermittent wireless connectivity, is regarded as a failure in a traditional system. However, the device might actually come back online almost immediately. In such case, there

is enough flexibility in the VC platform, to allow the immediate reactivation of the device, without any consequences in the higher layers, and without the need to redistribute or restart the execution. If, however, the failure is permanent or non-recoverable, then there is build-in support for automatic redistribution of the task to available services, as well as support for migratable services in case of predicted failures. In [25] an extended discussion about dependability issues in such hybrid environments is presented, along with the detailed description of the VC approach to dependability.

• **Resource Constraints:** The platform allows the integration of server-side components, exposed through different technologies (such as RMI components, Web Services, Grid Services, JB through Plain Sockets) in order to provide freedom of choice of the most suitable framework for each small scale device. Further, an efficient pull-based communication model has been introduced, and communication is minimized. In addition, to compensate for the limited contribution of each single peer, the server side components are virtualized to form single virtual resource pools through the aggregators. Further exploiting the virtualization of the underlying services, the ability to attach enhanced functionality to the aggregators is now acquired. This comes in the form of two main additions: an asynchronous invocation facility, and, most importantly, a collective layer. The latter is responsible for the collective management of the underlying nodes, and provides a number of advanced interfaces, namely a mirrored interface, for higher reliability and fault tolerance; a parallel interface, for data-parallel distribution towards high performance; a voting interface, for operations like leader elections, and reaching consensus; and finally, a scatter/gather interface to support data acquisition in an MPI pull-like scheme.

• **Abstractions:** The implementation of the VC abstraction, enables a unified and consistent interface to multiple and dynamic services/resources, possibly utilizing different binding protocols. Server-side components available in the underlying fabric layer can make use of a variety of transport protocols, such as WS, RMI, plain TCP sockets, and more. At the programming level, the VC abstraction will significantly ease software development and simplify some high-level programming models, such as the master-worker and the voting models, and will provide the foundations for collective (group) operations on the VC. Furthermore, it provides transparency from the underlying binding protocols and dynamic addressing schemes, and works as a bridging layer that provides unified access to heterogeneous services. Finally yet importantly, an Application Programming Interface (API) is made available to developers that want to have total control over the VCs.

Communication with the higher layers is achieved through either of two widely adopted technologies: WS and RMI. Access to the underlying functionality is facilitated through the aggregators, which are exported as standard Web Services and RMI objects in a local (in the proxy) RMI registry. Furthermore, the provided API is

built around RMI. Most core middleware services are implemented as RMI objects and are accessible through the API. Communication within the system (i.e. between the middleware components) is facilitated using RMI or plain sockets in some cases.

Entities in the higher layers may include: Portals, that will display the whole VC as a single resource; Grid middleware, where again, the VC is a static, permanent Web service; Application Clients, that need not change although with minor modifications they can make use of the enhanced functionality offered by the aggregator services (collective interfaces, index queries, monitoring information and more); Application Developers, who can take advantage of the enhanced functionality available by the aggregators, and make use of the extended API; Integration middleware, which can easily integrate both the WS and the RMI based aggregator services into existing distributed systems.

5. Experimental Results and Evaluation

5.1 Evaluation Plan

The collection of comprehensive evaluation results entails analysing the effectiveness and performance of the VC platform from a variety of perspectives:

- Architecture design level: validation of the design is necessary to ensure the suitability of the proxy-based architecture and has been presented in [23] after a series of simulations that indeed validated the VC approach.
- Lightweight middleware: in order to be deployable in mid-to-low range devices. This stems from mainly two target goals: ability to deploy the platform to non-dedicated and potentially resource limited devices, and second, to support and enable infrastructure-less environments, where a lower-level device can act as a proxy (or group super-node).
- Invocation overhead: since the proxy handles requests on behalf of the higher level clients, there might be an overhead involved due to this delegation of responsibilities. However, this absolute overhead is balanced out by the efficient communication model, and the much-reduced index queries involved due to the full knowledge of the proxy for the topology and bindings.
- As it is all about collective management and clustering, an number of experiments were run to evaluate the performance and benefits of the collective interfaces.

5.2 Monitoring and Control Overhead

The evaluation of the resource requirements is necessary to ensure suitability for constrained and non-dedicated environments. Management and control procedures involve the aggregation of services, the registration of services as they come online, and the overhead involved in index and service availability queries. The latter is part of the invocation process and hence is studied and analysed under the performance evaluation section. However, the most demanding aspect of the platform is the monitoring service group, and more specifically the heartbeat scheme for monitoring and failure management.

It should be noted that the aggregation and registration of services are fixed cost operations. Hence, the evaluation of the control overhead was based on the only variables that affect the operational overhead of the platform: the heartbeat interval and the number of nodes in the VC.

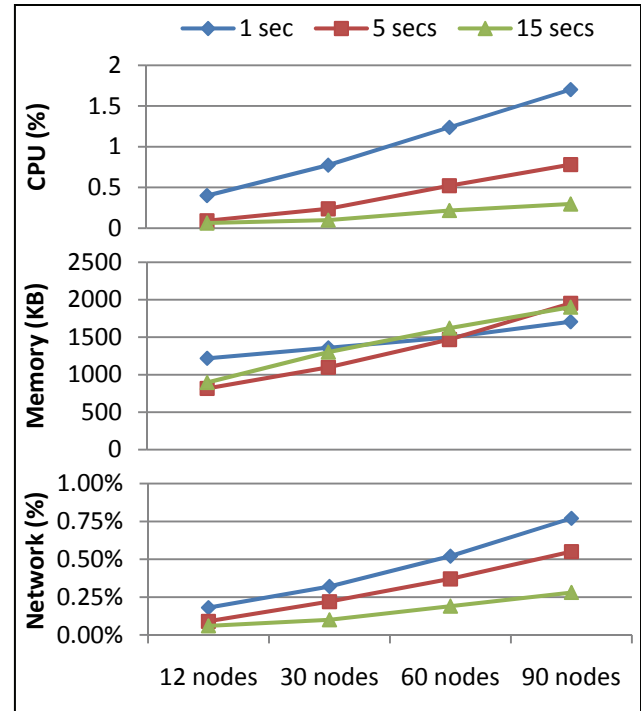


Fig. 2. CPU, Memory, Network requirements of the VC platform

The proxy middleware was installed and tested in a mid-range Laptop in an idle condition. Results (Fig. 2) showed that CPU utilization is minimal (around 1 - 2%) and does not in any way affect the normal operation of the laptop. Similarly, memory overhead remains very low as well. The total memory requirements remain under 2MB of memory even for a very high number of monitored nodes (90) and frequent intervals (every second). Finally, the Network utilization overhead is negligible, as it never goes beyond 0.8% in the 24MB /s wireless channel used in the experiments. This is attributed to the extremely small heartbeat packages, and the relatively fast 802.11g Wi-Fi protocol in use. While an extended discussion of these results is not in the scope of this performance oriented evaluation, we must point out the very small CPU, memory and network footprint of the platform, rendering it suitable even for non-dedicated proxy servers running on “small” devices –and not only typical server machines.

5.3 Performance Evaluation

The experimental environment consisted of a total of 6 nodes, 2 of each representative performance levels:

- 2x Type B laptops: 1.6GHz AMD laptops, 512MB, Windows XP, Axis/Tomcat plus RMI Registry.
- 2x Type A laptops: 2.2GHz Intel laptops, 1GB, Windows XP, Axis/Tomcat plus RMI Registry.
- 2x CDC Smart-phones: Sony Ericsson P990i, VC Lightweight Socket-Based Server-Side Container.

All of the devices are equipped with an 11MB/s Wireless Interface Card, and the supporting network was based on 24MB/s 802.11g, and a backbone of 100MB/s Ethernet. While this initial testbed consists of a relatively small number of test nodes, it demonstrates the response of the system in failures, and its benefits stemming from its aggregation and dynamic reconfiguration capabilities. Further, scalability and management using a very large number of nodes has been addressed in extensive simulations in [23]. The simulation results for up to six nodes were verified by the experimental measurements within 2-3%, adding confidence that the simulation results are equally valid for larger number of nodes that we were not able to test experimentally.

The application used exhibits the possibilities of such integrated environments, and involves aggregating content from dynamic and heterogeneous sources. It can be used, for instance, in geological studies where a mobile operation centre can gather content from wireless instruments and portable equipment, analyse it, and automatically re-allocate the field personnel in response to the latest findings (for example to focus more on a specific area or re-tune some of the instruments). For this scenario the mobile centre requests and gathers scanned 3D images of the area where each instrument is deployed. Of course the evaluation discussion that follows remains generic and not application-specific. For the purpose of studying the impact of failures in the VC platform, we injected hard failures during the service operations in order to study the reaction of the system and measure the overhead involved in responding to failures. We used a small script to inject failures during the executions at various points: near the beginning of the execution - around the 25% of total execution time, ($\pm 5\%$), near the middle of the execution - around 50% ($\pm 5\%$) and finally, as the execution was closing in termination - around 75% ($\pm 5\%$). All the results presented here are the averages from around 500 runs of each scenario, meaning that any possible interference or environmental effects that could skew some of the measurements have become negligible. The participating wireless nodes expose this functionality through different means: the laptops as both a Web service and an RMI component, while the PDAs using a Lightweight Server-side Container (LSC) we developed specifically for our experiments [27].

The first experimental scenario involved the invocation of the same image service first facilitating direct communication with the service and then through the VC proxy, in order to measure the overhead induced by the VC. As already presumed, the VC proxy introduces a slight overhead in the range of 1.5 - 2.5% compared to direct invocation (Fig. 3). The measurements from this first experiment also verified the performance superiority of our LSC container to RMI and WS. However, a number of remarks will clarify the results a little more. Both the direct and VC single invocations assumed prior knowledge of the service endpoint and the binding protocol. In a more realistic scenario, the direct invocation would involve a query to discover available

services (hard-coded addresses are unrealistic for mobile environments). In contrast, the VC proxy always has full knowledge of the underlying services, topology, addresses, and binding protocols (thanks to its monitoring and indexing components). Such benefits become clearer when studying the performance of the collective interfaces, but also in the next set of experiments, when we introduce failures in the system.

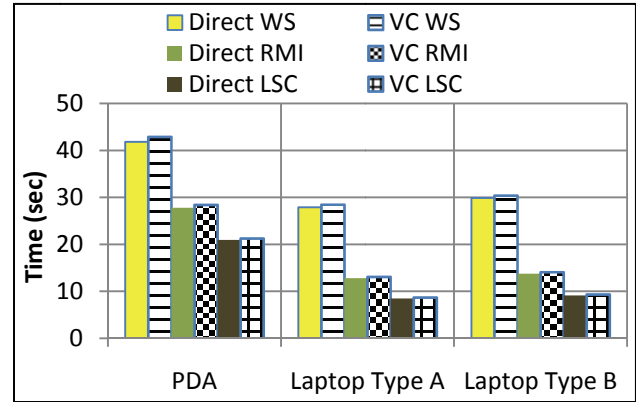


Fig. 3. Comparison of Direct and VC single invocations

When injecting a single failure in the system during execution (Fig. 4) the benefits of the VC approach become evident: as the proxy has full knowledge of the topology and capabilities of the available devices, the system is guaranteed always to select the best possible node for execution recovery. For instance, in the experiments run, the proxy would always redistribute the task to the second laptop available if the first one failed (leading to performance gains of up to 37% in the best case), whereas in a direct approach, there is always a chance to select the lower scale PDA to failover to. The latter can occur because of lack of complete knowledge or metadata for each system, lack of dynamic state information that can affect the performance (e.g. current load), or lack of availability information, which happens very often when dealing with very dynamic environments. The last column in Fig. 4 corresponds to automatic mirrored invocation (clearly superior to the other methods), which is discussed in detail later in this paper.

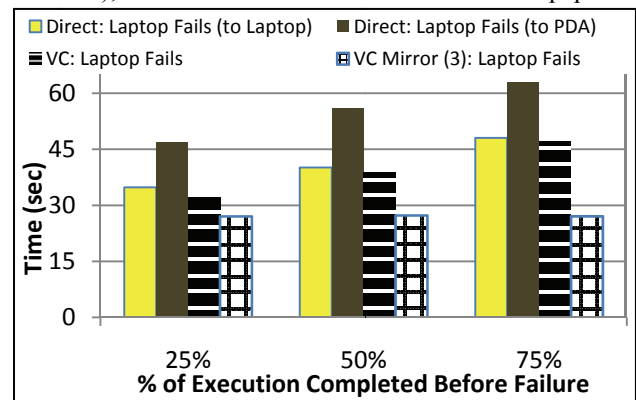


Fig. 4. Direct, VC, and VC Mirrored invocations with one failure

However, even if we assume that a relatively intelligent brokering or registration system helps select the best

possible resource in the direct approach, a possible failure would still have to propagate as high as at least the brokering system, sometimes up to the client level. In contrast, in the VC approach it only goes up one level – one hop in essence, till the proxy, which takes care of redistribution immediately due to its full a priori knowledge of the cluster condition. This contributes to a small redistribution gain in the region of 2 - 4% in the case of the proxy.

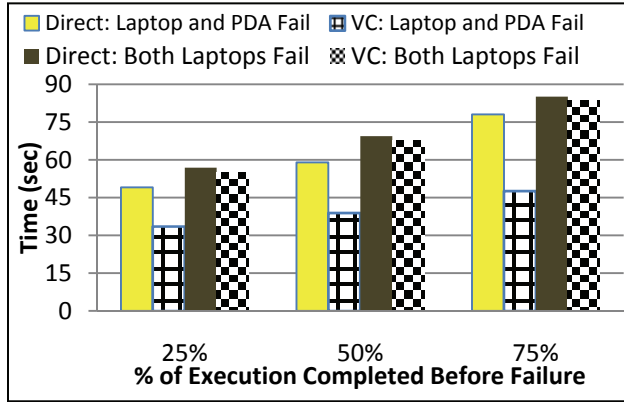


Fig. 5. Direct and VC invocations with two failures

The benefits of the VC platform are even better in the case of two failures, a laptop and PDA (Fig. 5): the PDA failure would never affect the execution in a VC-based system, as the failover always happens to the more powerful laptop instead. Hence, the performance gains in this experiment are justified by the fact that even if there are two device failures, there is actually only one task failure in the VC-based system.

When both laptops fail they do so in a sequential order, thus leaving the PDA as the sole “survivor” able to take over the execution of the task. Hence, the gains in this example (in the area of 3-5%), originate from the masking and internal handling of the failures in the VC approach, while in the direct approach, both failure notifications propagate up to higher-level components, generating more load and more resource queries in the process.

5.4 Collective Layer Performance Evaluation

An integral part of the middleware is the collective interfaces, that enable the efficient utilization of scarce resources and individually small contribution if the mobile devices. The benefits of the collective layer have been discussed and analyzed in [25]. In this section, we evaluate the benefits of mirrored execution, which adds to the overall reliability and availability of the VC, and the parallel and gather interfaces that enable the distribution of tasks to many nodes, and the effortless gathering of data from multiple nodes respectively.

Mirrored Interface

Mirrored execution guarantees to return in the shortest possible time since the first service that completes returns immediately. We run a number of experiments, where a brokering and scheduling system with no metadata information provided the resources upon which to execute the required task. We also run the same experiment

utilising our VC mirrored functionality with resource indexing. The comparison shows that the mirrored execution at least matched and usually was far better than the traditional system.

Mirrored invocation has even more significant advantages in the light of failures occurring in the fabric layer, in that it offers a certain degree of fault tolerance. As is obvious from Fig. 6, not only are the gains massive (especially compared to the non-VC, direct invocation) at times halving the total time cost, they are also stable regardless of the failure time -as the task has already been distributed to all three devices, thus not requiring redistribution and restarting when a failure is encountered.

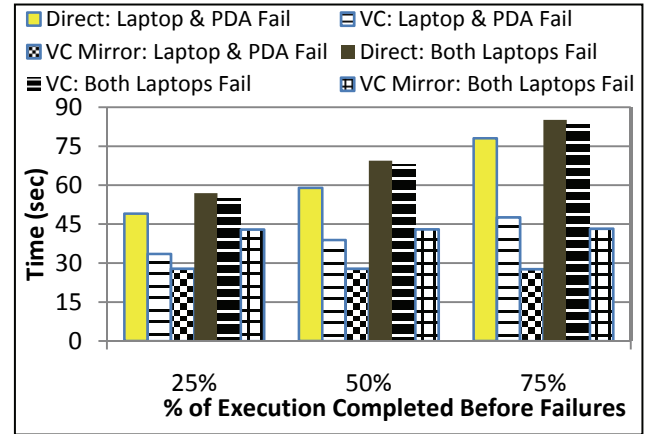


Fig. 6. Direct, VC, and VC Mirrored invocations with two failures

The trends in the case of two failures (Fig. 6) tend to amplify the importance of transparent mirrored invocation whenever there is resource availability. Total time costs when a laptop and a PDA fail, are almost halved compared to the direct approach, and there is significant improvement (from 15% up to 85%) when compared to the VC single invocation.

Finally, in the last scenario, both laptops fail sequentially, thus leaving the execution burden to the single available PDA. The total time cost in the mirrored execution is equal to the total time taken by the PDA to complete the task, since the failures do not require redistribution. Once again, there is a clear improvement when using the mirrored interface compared to both the direct and the VC single invocation.

Parallel Interface

The last category of experiments, focused on evaluating the performance of the simple data parallel interface. In order to extract some baseline results disregarding failure, the same image gathering application was executed using three different image sizes. The application was first executed on a single Type A laptop to measure single non-parallel execution. Then, three distributed versions were executed, first on a Type B laptop and a PDA, then on two Type B laptops, and finally, on two Type B laptops and a PDA.

Following the execution runs, there are a number of observations that can be made from the results as demonstrated in Fig. 7. First, the correlation of the size to

the total time required is approximately linear, and thus the rest of the experiments will use only 512KB image sizes without loss of generality and in order to simplify the charts. Second, and as expected, due to the limited nature of the participating devices, the effect of the distribution is not linear on the number of nodes. Distributing to a laptop and a PDA does not (and could not) yield an improvement of factor 2. Instead, there was a reduced, albeit significant, improvement in the region of 20-30%. In the case of distributing to two laptops, results were of course better, yielding an improvement around 35-42%. Finally, the best results were obtained when distributing to all three devices (two laptops and a PDA), when the performance gains were at the region of 36-48%.

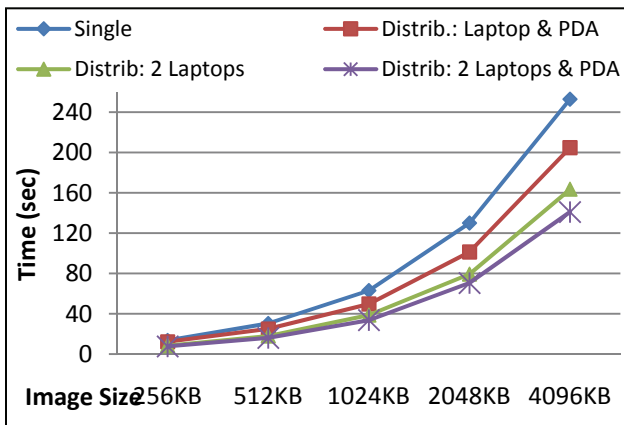


Fig. 7. Direct and data parallel invocation, two laptops and one PDA

To study the affect of failures in the parallel execution, an extended environment of four laptops and two PDAs was used to provide backup for the failovers. For the initial distribution of the application load, two Type B laptops and one PDA were used. From the diagrams in Fig. 8, it is clear that even in the worst case of two failures, parallelization can yield significant results (thus providing a virtually stable and reliable environment) of around a factor of 2.0 at best, and near a factor of 1.3 at the worst cases.

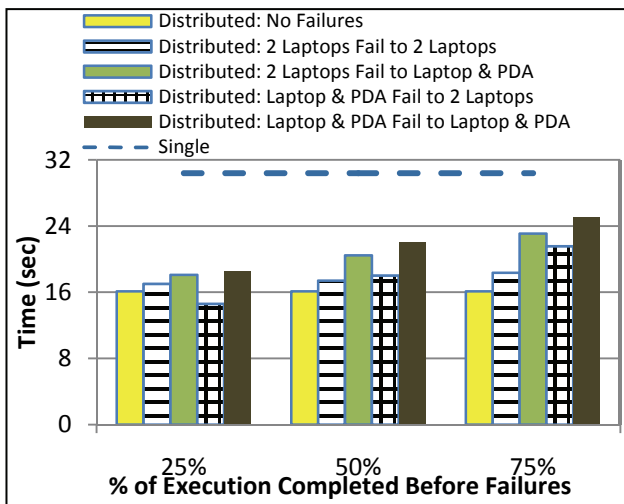


Fig. 8. Impact of two failures in parallel invocation

6. Conclusions

The results presented in this paper exhibit the suitability of the Virtual Cluster approach through an extensive set of experiments. All the results have demonstrated the significant benefits that the VC proxy-based platform can have in integrated hybrid Grid systems. These benefits include not only the very straightforward and quantifiable performance gains, but also the simplification of otherwise troublesome procedures, such as resource availability querying, resource selection, failure notifications and general management of the mobile domain. Further, they have shown that parallel execution is suitable even for very dynamic and limited environments, when the VC platform is deployed to encapsulate the dynamicity and failures. This enables the rapid high-level hybrid application development, which is further enhanced through the VC abstractions, the VC API, and the collective interfaces.

Acknowledgment

This research has been partly supported under the EU Network of Excellence CoreGRID (Contract IST-2002-004265).

References

- [1] Foster, I. and Kesselman, C., [ed.]. *The Grid II: Blueprint for a New Computing Infrastructure* (Morgan Kaufmann, 2003).
- [2] UK e-Science Program. [Online] <http://www.rcuk.ac.uk/escience/default.htm>.
- [3] Cheng, L., Wanchoo, A. and Marsic, I. Hybrid Cluster Computing with Mobile Object., *Proceedings of the 4th International Conference on High-Performance Computing in the Asia-Pacific Region (HPC-Asia)*. 2000, pp. 909-914.
- [4] McKnight, L. and Howison, J. Wireless Grids: Distributed Resource Sharing by Mobile, Nomadic, and Fixed Devices, *IEEE Internet Computing*. July/August, Vol. 8(4), 2004, pp. 24-31.
- [5] Phan, T., Huang, L. and Dulan, C. Challenge: Integrating Mobile Devices into the Computational Grid, *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking*, 2002, pp. 271-278.
- [6] Kesselman, C. and Foster, I. Globus: A Metacomputing Infrastructure Toolkit, *International Journal of Supercomputer Applications*, Vol. 11, 1997, pp. 115-128.
- [7] The Globus Toolkit. [Online] <http://www.globus.org>.
- [8] NET Compact Framework Mobile Web Server Architecture. [Online] <http://msdn2.microsoft.com/en-us/library/aa446537.aspx>.
- [9] Schall, D., Aiello, M. and Dustdar, S. Web Services on Embedded Devices, *International Journal of Web Information Systems (IJWIS)*. 2007

- [10] Raccoon Mobile Web Server. [Online] <http://research.nokia.com/research/projects/mobile-web-server/>.
- [11] The AKOGRIMO project. [Online] <http://www.akogrimo.org>.
- [12] Wesner, S., Jahnert, J. M. and Escudero, M. *Mobile Collaborative Business Grids – A short overview of the Akogrimo Project*. [Online] http://www.akogrimo.org/download/White_Papers_and_Publications/Akogrimo_WhitePaper_Overview.pdf.
- [13] Ruiz, N. *A Framework for Integrating Heterogeneous, Small Scale Devices into Computational Grids and Clusters*. MSc Thesis. University of California, US, 2003.
- [14] Message Passing Interface Forum. *MPI: A Message Passing Interface Standard*. [Online]. www.mpi-forum.org/docs/mpi-11-html/mpi-report.html. 1995
- [15] The K*Grid Mobile Grid project. [Online] <http://gridcenter.or.kr/MobileGrid/index.php>.
- [16] Lima, L.; Gomes, A. T. A.; Ziviani, A.; Endler, M.; Soares, L. F. G.; Schulze, B. Peer-to-Peer Resource Discovery in Mobile Grids, *3rd International Workshop on Middleware for Grid Computing*, 2005, pp. 1-6.
- [17] Hwang, J. and Aravamudham, P. Middleware Services for Peer-to-Peer Computing in Wireless Grid Networks, *IEEE Internet Computing*. July/August, Vol. 08(4), 2004, pp. 40-46.
- [18] Messer, A.; Greeberg, I.; Bernadat, P.; Milojevic, D. Towards a Distributed Platform for Resource-Constrained Devices, *Proceedings of the IEEE 22nd International Conference on Distributed Computing Systems (ICDCS'2002)*, 2002
- [19] Balan, R. K.; Satyanarayanan, M.; Park, S.; Okoshi, T. Tactics-Based Remote Execution for Mobile Computing, *Proceedings of the 1st USENIX International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2003, pp. 273-286.
- [20] Narayanan, D., Flinn, J. and Satyanarayanan, M. Using history to improve mobile application adaptation, *Proceedings of the 3rd IEEE Workshop on Mobile Computing Systems and Applications*, 2000
- [21] Flinn, J., Narayanan, D. and Satyanarayanan, M. Self-tuned remote execution for pervasive computing, *Hot Topics on Operating Systems (HotOS-VIII)*, 2001.
- [22] Li, Z., Wang, C. and Xu, R. Computation offloading to save energy on handheld devices: A partition scheme, *Proceedings of the International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, 2001, pp. 238-246.
- [23] Isaiadis, S. and Getov, V. Integrating Mobile Devices into the Grid: Design Considerations and Evaluation, *Proceedings of the 11th International Euro-Par Conference on Parallel Processing*, 2005, pp. 1080-1088.
- [24] Isaiadis, S. and Getov, V. A Lightweight Platform for Integration of Mobile Devices into Pervasive Grids, *Proceedings of the 1st High Performance Computing and Communications (HPCC)*, 2005
- [25] Isaiadis, S. and Getov, V. Dependability in Hybrid Grid Systems: a Virtual Clusters Approach, *Proceedings of the IEEE JVA International Symposium on Modern Computing*, 2006
- [26] Harrison, A. and Taylor, I. Dynamic Web Service Deployment Using WSPeer, *Proceedings of the 13th Annual Mardi Gras Conference - Frontiers of Grid Applications and Technologies*, 2005
- [27] Isaiadis, S. and Getov, V. Dynamic Service Aggregation in Heterogeneous Grids, *Proceedings of the CoreGRID Institute of Grid Systems, Tools, and Environments Workshop*, Heraklion, Greece, 2007