Grid Access Middleware for Handheld Devices

Saad Liaquat Kiani[1], Maria Riaz[1], Sungyoung Lee[1],
Taewoong Jeon[2], and Hagbae Kim[3]

[1] Computer Engineering Department, Kyung Hee University,
Giheung, Yongin, Gyeonggi 449-701, Korea
{saad, maria, sylee}@oslab.khu.ac.kr
[2] Department of Computer & Information Science, Korea University, Korea
jeon@selab.korea.ac.kr
[3] School of Electrical & Electronic Engineering, Yonsei University, Korea
hbkim@yonsei.ac.kr

Abstract. Grid technology attempts to support flexible, secure, coordinated information sharing among dynamic collections of individuals, institutions, and resources. The use of Grid services requires a resourceful workstation, specialized software installed locally and expert intervention. Mobile handheld devices in general do not posses enough computational and communication assets to meet the criteria for utilizing the Grid infrastructure services. We present the design of a middleware approach that aids handheld devices in this regard by wrapping the computational and resource intensive tasks in a surrogate and shifting them to a capable machine for execution[1]. Reduction in computational burden at the handheld device is analyzed in a test scenario.

1 Introduction

Grid [1] computing harnesses the abundant spare, and sometimes dedicated computational resources in a globally distributed computing environment and puts them to effective and optimal use. Grid setup is clearly a value addition to any organization, commercial or research. One of the main motivations behind the Grid infrastructure is to provide "a hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to high-end computational capabilities" [2]. Grid infrastructure has been put to use in areas like high energy physics [3], bio-medicine [4], aerospace and earth sciences, health-care [5] etc and is continuing to evolve and expand. Similar technology adoption trends are seen at the smaller scale. With ever decreasing costs and increasing functionality of small sized chips, mobile handheld devices e.g., Personal Digital Assistants (PDA) and smart phones, are becoming mainstream now. For a mobile user, a PDA takes place of his home/office PC while he is on the move; he can not only use the internet and check emails through wireless connectivity

[1] This work is supported in part by the Korea Ministry of Information and Communications' ITRC program in joint collaboration with ICU Korea.

P.M.A. Sloot et al. (Eds.): EGC 2005, LNCS 3470, pp. 1002–1011, 2005.

but can also write documents, play games, find street maps, make reservations at hotels and restaurants and perform similar utility tasks. A broad spectrum of internet services has become available for a mobile user. Grid and mobile computing however remain two disjoint phenomenons as yet, keeping users of both technologies from utilizing some propitious mutual benefits. While mobile elements will improve in absolute ability, they will always be resource-deprived relative to their static counterparts (desktops/workstations). In [6], the author argues that for a given cost and level of technology, considerations of weight, power, size and ergonomics will exact a penalty in computational resources such as processor speed, memory size, and disk capacity. These devices do not have enough resources in effect to utilize the Grid services comprehensively. The potential benefits of facilitating mobile devices in interacting with Grid services in the numerous fields are:

- Health care: A physician submitting digital charts to mammography Grid services [7] for analysis
- Emergency medical services: Submitting vital characteristics, medical history of a trauma patient to Grid services for identification
- Research: A physicist who needs to see graph plots of data produced as a result of high energy collisions between atoms and sub-particles on his PDA. The amount of information in data-stores, from which graphs are to be generated, will be in the range of several gigabytes or even tera bytes
- Weather: Forecasting and analyzing local weather conditions, storm formations while on the move
- Geology: Geologists sampling rocks and terrain and using handheld devices to submit data to Grid services for analysis

All these domains represent scenarios where a user wants to execute a resource intensive task at a location where computation resources are not available at hand. With ever increasing mobility of users and greater adoption of handheld devices, job submission to the Grid through handheld devices presents a viable solution for maximizing efficiency. Constraints that hinder handheld devices from such interactions include limited network bandwidth, CPU power, memory (small network buffers) and intermittent connectivity. Keeping the limitations in mind, we aim to define a middleware approach that will allow handheld devices, e.g. PDA units, to interact with Grid services while inducing minimal burden on the device itself. We demonstrate a solution based on Jini Network Technology's [8] Surrogate Architecture [9] which provides a network framework in which a device can deploy a client or a service on a device other than itself. Since we are stepping in a new realm of Grid access through handheld devices, many design and performance challenges need to be considered and countered. In the domain of Grid infrastructure, where services and data resources are replicated across geographical boundaries [10, 11], communication costs can be minimized by careful selection of intermediate network. The communication mechanisms involved in job submission, execution and resource access are optimized at three levels: 1) Selection of the host to which the device will submit the job/task for execution, 2) Resource access by the surrogate during execution and 3) filtering

and optimization of intermediate results that are to be transferred to the device from the remote machine. One possible approach for facilitating handheld device interaction with the Grid is to narrow down the criteria for Grid access and make it less resource hungry; but doing so will also take away several benefits. How can a resource constrained device be configured and supplemented with software based techniques to make it Grid-interaction capable? A handheld device wishing to host a service and unable to do so can be allowed to delegate this task to a relatively powerful machine (desktop, server). Conversely, if the interaction with remote Grid services proves too much for limited local resources of a handheld device, it can deploy the actual client functionality at an intermediate machine and receive the results in a form that is in keeping with its hardware resources. This second scenario has a greater probability of being used in real world applications and is the focus of our research. The 'service' or 'client' process, transferred from the device, is called a 'surrogate' (The term 'surrogate' is used to describe an entity that performs some action on behalf of another entity). The middleware component at intermediate machine, which provides the execution environment and access to extensive resources for the handheld device's surrogate, is called the 'Gateway Surrogate Host' or simply 'Host'. An interconnect mechanism, defined as "logical and physical connection between the surrogate host and a device" [12], also needs to exist. A handheld device that can communicate over IP (wireless or wired) can be programmed to shift its task processing to a host capable machine. An overview of our middleware approach is presented in Sect. 2. Section 3 deals with the communication mechanisms and the proposed optimizations in the middleware. Prototype implementation and test results are presented in Sect. 4. We conclude our discussion in Sect. 5 and also list relevant related work.

2 The Grid Access Middleware Architecture

A handheld mobile device having wired/wireless connectivity can utilize the functionality of its more capable computing peers for resource demanding tasks such as Grid service access, with the device itself only managing less intensive tasks like displaying the tailored results. The main concept driving our approach is to shift the 1) access to generic Grid services and 2) intensive task processing, from a resource constrained handheld device to a resource rich system (i.e. the Surrogate Host). This is to be achieved by wrapping the access and processing mechanisms in a 'surrogate' module and transferring to the host. Consider the example of a physicist provided in Sect. 1, where he needs to see graph plots, on his PDA, of data produced as a result of high energy collisions between atomic particles. The amount of information in data-stores from which graphs are to be generated will be in the range of several gigabytes or even tera bytes. Analysis of such data for the plotting of graphs is not a job for the handheld device. Moreover, the handheld device may have reduced network bandwidth, further diminishing the prospects of a successful remote analysis by the user. By utilizing the Jini Surrogate Architecture based middleware support, one can 'pack' the

functionality for data-stores' access mechanisms and graph plotting routines in a surrogate and transfer this surrogate to a host machine. The host machine will provide the surrogate with necessary resource rich execution environment and network connectivity. The surrogate is able to communicate back to the device (PDA) through available interconnects e.g. IP, USB, Bluetooth etc. In this way, the aforementioned tasks of service access and intensive processing can be shifted from the handheld device to a more appropriate host machine, with the device only managing less intensive tasks of displaying the tailored results returned by its surrogate. Figure 1 shows the middleware framework which consists of three distinct stacks deployed at the Gateway Surrogate Host, the Device and the surrogate. These are discussed one by one in the subsequent paragraphs.

2.1 Gateway Surrogate Host

Major technical hurdles make it impossible for Devices to exploit the benefits made available by the computational and data Grids, including the ability to execute applications whose computational requirements exceed local resources and reduction in job turn around time through workload balancing across multiple computational facilities. Gateway Surrogate Host is the middleware component that aids the Device to overcome these hurdles by accepting tasks, packed as surrogates, for execution. The middleware provided at these hosts consists of three main sub-modules. Host Adapter sub-module offers an interface to client devices for accessing the Gateway Surrogate Host. It enables the initial communication between the device and the host so that both can agree on the transfer of surrogate after authenticating the device and its corresponding surrogate. Once

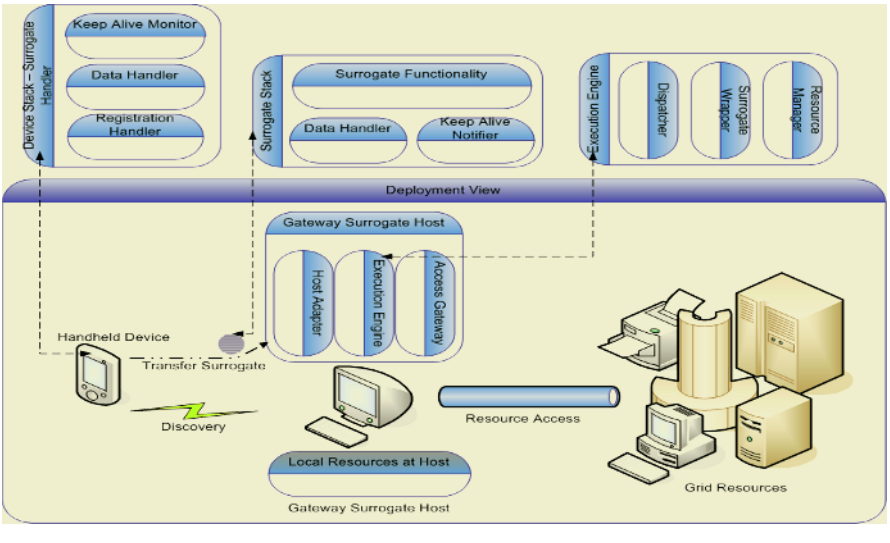

Fig. 1. Grid Access Middleware Stacks at the Device, Surrogate and Host

the surrogate is available at the host, it is delivered to the Execution Engine sub-module. It consists of a Surrogate Wrapper that exposes the functionality of the surrogate that is required to facilitate surrogate's execution at the host. Dispatcher allocates a separate thread for the execution of the surrogate from a thread pool and then activates the surrogate. Resources required for surrogates' execution are resolved and handled by the Resource Manager module. These include memory and disk space, JVM (form Java based surrogates, as is the case with our implementation), network resources etc. The Access Gateway sub-module provides interface to the external resources e.g. discovery of available Grid services and resources. A Gateway Surrogate Host announces relevant attributes including, but not restricted to:

- ID, Location, Currently hosted surrogates etc
- Network address and Discovery/Listening port for incoming Device/Client requests
- Available/Allocated Resources e.g. CPU, Memory, Storage, Network throughput
- Environment e.g. Java VM availability and version, SOAP/WSDL [13, 14], XML parser etc
- Grid services available through this Surrogate Host
- Proximity to service and client side

Advertising these attributes allows clients to locate appropriate hosts based on their location, network proximity and other desired features. This is further elaborated in section 3.1. Administrator of a host can restrict the number of surrogates that are allowed to execute, restrict memory, bandwidth allocation etc on per surrogate basis. Security policies can be configured based on public/private key pairs and digital certificates. The Gateway Surrogate Host is an extension of the basic Surrogate Host with added functionality for Grid access through the Access Gateway. It overcomes the major technical hurdles that keep the Devices from exploiting the benefits made available by the computational and data Grids [13, 14] by providing an interface to the Devices on one hand and to the Grid services on the other.

2.2 Device Stack

At the Device, a lightweight middleware stack is provided for facilitating coordination with its exported surrogate. The stack consists of a Surrogate Handler module which has three sub modules for providing services complementary to the middleware at the Gateway Surrogate Host. Registration Handler discovers, selects and registers with the Host, and transfers the surrogate. Once the surrogate is transferred, Keep Alive Monitor keeps track of the status of the surrogate. Data Handler retrieves the results from the surrogate-side corresponding module, and makes them available to the application executing at the device. Surrogate to be transferred can be stored at the Device itself or at a URL accessible store e.g. a web server or an FTP server.

2.3 Generic Surrogate

A generic surrogate for Grid service access contains the following features:

- Client authentication based on public/private key pairs
- Generic functionality to communicate and interact using WSDL/SOAP for web service based Grid services
- Persistency safe i.e. to be put to persistent storage if its functionality is periodic
- Migration - To be able to stop and save current execution, mark restore points and migrate to a different Surrogate Host

Functionality of the generic surrogate is incorporated at the top layer of the surrogate stack as shown in Figure 1, along with the specific logic of the extended Surrogate. Moreover, the surrogate has complementary modules for communicating with the middleware stack at the Device. The downloadable Surrogate can be located in the file system of the Device or at a URL accessible store e.g. a web server or FTP server. Some clients may be void of any Surrogates. These clients/devices are still able to use other deployed surrogates if they can provide valid credentials as rightful owner or users.

3 Discovery and QoS

There is a critical requirement for the clients/devices to be able to discover available Gateway Surrogate Hosts. Absence of a discovery mechanism has the potential to pose as a single point of failure. For reasons of efficiency and fault tolerance, multiple discovery techniques are provided in the architecture. The foremost method of discovery is multicast announcements from Gateway Surrogate Hosts. This automatically provides for locating 'nearby' hosts to the devices (as multicast is often geographically limited to a network boundary by most administrators). HTTP based discovery is provided as a supplement. All available Gateway Surrogate Hosts register with a web service hosted on a known location. Client devices/applications can inquire about a particular host by submitting appropriate parameters to this service over HTTP.

The surrogate paradigm will function most efficiently when the network delays between the device/client and surrogate are minimal. Moreover, efficiency also depends on the proximity of surrogate to the service being accessed. Since the user may be mobile with respect to the Gateway Surrogate Host and Grid resources, support is needed in the architecture to optimize the proximity based parameters. Each Gateway Surrogate Host will keep track of its access quality towards known/available Grid service hosts/networks. On the other hand, before deploying a surrogate, client side application can determine its network connectivity and temporal efficiency with a specific host. This procedure poses a certain one time per start-up burden, but offers better QoS relative to a scenario where such optimizations are left to good luck.

4 Implementation Overview

The authors have provided a bare-bones implementation of the proposed architecture. Before this design is tested for actual Grid service interaction, it is necessary to validate its viability in a general scenario which involves considerable CPU, memory and network utilization. Simple Network Management Protocol [15] is a widely accepted and utilized way of monitoring network entities and we have chosen to verify our approach by monitoring a remote server for 14 system statistics periodically, through a handheld device. Handheld device has network connectivity through a wireless LAN interface. A desktop machine is configured to act as a Gateway Surrogate Host. A Surrogate has been coded for the handheld device with the functionality of monitoring the remote server through SNMP queries and adjusting the results to be sent back to the Device. The results of these queries are to be displayed at the handheld device in the form of dynamic line, bar and pie charts/graphs. Performance of the device and the impact of our executing system will be measured and the benefits and shortcomings of the approach will be highlighted.

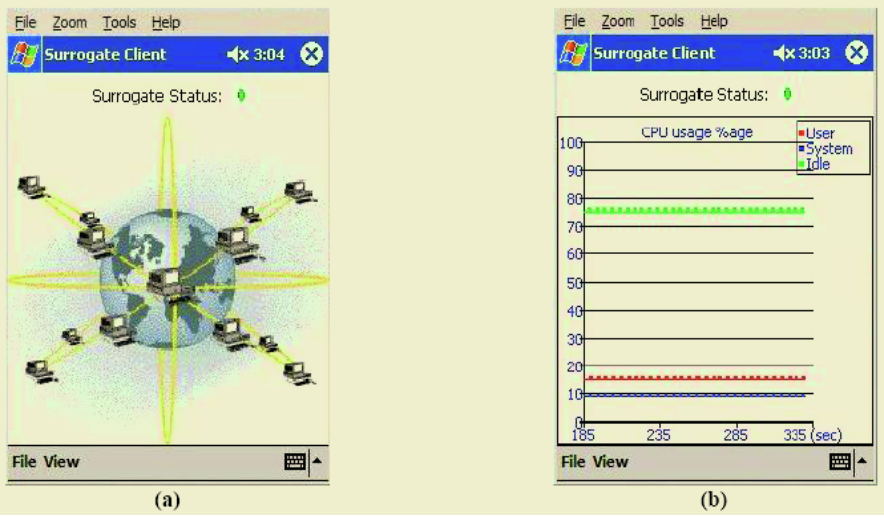

Fig. 2. (a) Main window of the client application executing on the device (b) Remote host's CPU usage statistics by category (user, system, idle) over a period of time, as seen on the device

The Gateway Surrogate Host module has been implemented by modifying and extending the Surrogate Host provided with the reference implementation of Jini Surrogate Architecture specification. The extensions include addition of useful attributes to be announced, additional discovery mechanism and addition of an SNMP agent. IBM's J9 VM for java is used to implement the surrogate

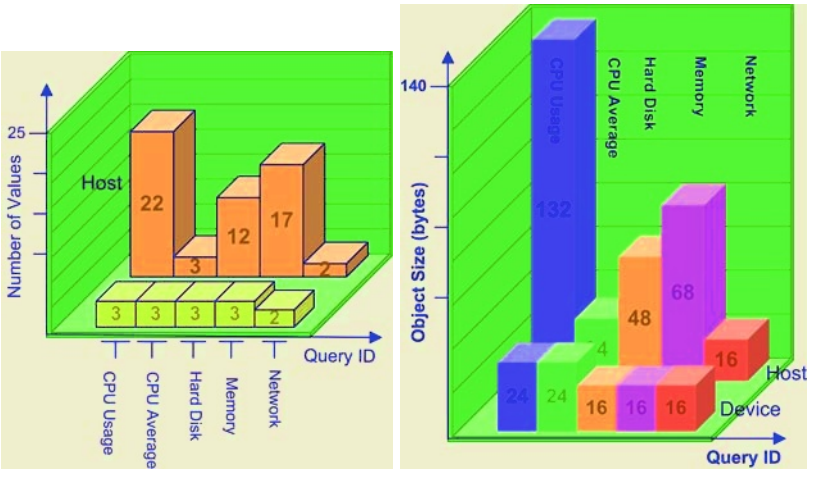

Fig. 3. (Left) Comparison between number of values at the Host and values sent to the Device; (Right) Comparison between size of intermediate results at the Host and size of results at Device

for the handheld device and contains classes which implement the task that the Device wishes to execute. Moreover, it contains the 'device-to-surrogate' interconnect implementation which, in the case of this scenario, is based on IP Interconnect Specification.

4.1 Performance Measurements

Measurements taken to analyze the performance of the Device during the course of execution are presented in Table 1.

Table 1. Result parameter count and size comparison at Gateway Surrogate Host and Device

Query Type	Number of values received at Host	Intermediate result size at Host (bytes)	Number of values sent to Device	Result size sent to Device (bytes)
CPU Usage	22	132+	3	24+2
CPU Avg. Load	3	24+	3	24+2
HDD Utilization	12	48+	3	16+2
RAM Utilization	17	68+	3	16+2
Network I/O	2	16	2	16+2
Total	56	288+	14	106

The size of result object depends on the type of values stored in the fields. The 14 statistical values are received in 5 'Result' objects and amount to, on

average, 62 bytes of results per 5 seconds with additional 44 bytes after every minute. An interesting comparison is made by considering the number of result parameters and their size as retrieved by the surrogate (executing at the Gateway Surrogate Host) with the corresponding values at the Device. A significant amount of information can be condensed by applying intermediate calculations and filtration of values at the surrogate module.

It can be observed that the number of parameters is reduced by 75 percent (4 times reduction) when transferring results to the Device. Similarly, more than 64 percent of the data has been filtered out in intermediate calculations and trimming at the surrogate. This performance markup is in addition to the communication reduction achieved by careful selection of host machine and resources access mechanisms throughout the surrogate's lifetime, as explained earlier. The burden on PDA has been reduced to a few hundred bytes of data and graph formation.

5 Conclusion and Related Work

Research and development for facilitating handheld held devices to interact with Grid services is in early stages. Signal [16] proposes a mobile proxy-based architecture that can execute jobs submitted to mobile devices, in-effect making a grid of mobile devices, but this approach may affect the fault tolerance of the system as the mobile device hosting the proxy also has to deal with the adverse conditions of a mobile/wireless environment. Moreover, the proxy has to schedule the jobs submitted to it by other mobile devices. In our case, the middleware has far more resources at its disposal, so the scheduling can be more flexible and concurrent. GridBlocks [17] builds a Grid application framework with standardized interfaces facilitating the creation of end user services. They argue that SOAP usage at mobile devices may be 2-3 times slower than a proprietary communication protocol, but the advantages of using SOAP (such as overcoming device heterogeneity) may be far more profitable than its limitation.

A solution based on Jini Surrogate Architecture to access Grid services is demonstrated in this paper. In the proposed approach, a resource constraint device wishing to access a resource-demanding service is allowed to delegate this task to a relatively powerful machine (desktop, server). Specifically, CPU intensive, network oriented tasks can be efficiently delegated to such systems when network connectivity is available. In case of intermittent connectivity, applications and services requiring on-demand or periodic network access can benefit from this approach. The implementation has been tested for a moderately intensive task. We intend to extend and implement the architecture to interact with existing Grid services and analyze the performance of our framework. These include HTTP discovery, client authentication, and surrogate migration support. A notable constraints suffered by our approach include the requirement of Java virtual machine at the device. Furthermore, at present we have not addressed the notions of client/surrogate authentication and authorization and are the focus of our future work.

References

1. Foster, I.: What is the Grid? A Three Point Checklist. In: GRIDToday (2002)
2. Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. In: Morgan Kaufmann Publishers, San Fransisco (1999)
3. Bunn J., Newman H.: Data-intensive Grids for High-Energy Physics. In: Berman, F., Fox, G.E., Hey, A.J.C. (eds.): Grid Computing: Making the Global Infrastructure a Reality, Wiley (2002). 859-906
4. Hastings, S., Kurc, T., Langella, S., Catalyurek, U., Pan, T., Saltz, J.: Image Processing for the Grid: A Toolkit for Building Grid-enabled Image Processing Applications. In: 3rd International Symposium on Cluster Computing and the Grid, May 12 - 15, 2003, Tokyo, Japan.
5. Breton, V.: Health Grid. In: International Symposium on Grid Computing 2003, Academia Sinica, Taipei, Taiwan (2003)
6. Satyanarayanan, M.: Fundamental Challenges in Mobile Computing. In: Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing, Philadelphia (1996)
7. Amendolia, S.R., Brady, M., McClatchey, R., Mulet-Parada, M., Odeh, M., Solomonides, T.: MammoGrid: Large-Scale Distributed Mammogram Analysis. In: Proceedings of the XV111th Medical Informatics Europe conference (MIE'2003). St Malo, France May 2003. Volume 95 of Studies in Health Technology and Informatics, pp 194-199 IOS Press, Amsterdam.
8. Sun Microsystems, Inc.: JiniTM Architecture specification. http://www.sun.com/jini/specs/
9. Sun Microsystems, Inc.: JiniTM Technology Surrogate Architecture Specification. http://surrogate.jini.org/sa.pdf (2003)
10. S. Vazhkudai, S., Tuecke, S., Foster, I.,:Replica Selection in the Globus Data Grid. In: Proceedings of the first IEEE/ACM International Conference on Cluster Computing and the Grid (CCGRID 2001), IEEE Computer Society Press,(2001) 106-113,
11. Lee, B., Weissman, J.B.: Dynamic Replica Management in the Service Grid. In: High Performance Distributed Computing 2001 (HPDC-10"01), San Francisco, California (2001) p. 0433
12. Sun Microsystems, Inc.: JiniTM Technology IP Interconnect Specification. http://ipsurrogate.jini.org (2001)
13. Lee, S., Gerla, M.: Dynamic Load-Aware Routing in Ad hoc Networks. In: Proceedings of The Third IEEE Symposium on Application-Specific Systems and Software Engineering Technology (ASSET 2000), Richardson Texas (2000)
14. Godfrey, B., et al.: Load Balancing in Dynamic Structured P2P Systems. In: IEEE INFOCOM 2004, Addis Ababa, Ethiopia (2004)
15. Stallings W.: SNMP, SNMPv2, SNMPv3, and RMON1 and RMON2. 3rd Edition Addison-Wesley, California (1999) 71-82
16. Hwang, P. Aravamudham Middleware Services for P2P Computing in Wireless Grid Networks. In: IEEE Internet Computing vol. 8, no. 4, July/August 2004, pp. 40-46
17. Gridblocks project (CERN) http://gridblocks.sourceforge.net/docs.htm