Bringing Handhelds to the Grid Resourcefully: A Surrogate Middleware Approach

Maria Riaz, Saad Liaquat Kiani, Anjum Shehzad, Sungyoung Lee

Computer Engineering Dept., Kyung Hee University
Yongin-Si, Gyeonggi-Do, 449-701, Republic of Korea
{maria, saad, anjum, sylee}@oslab.khu.ac.kr

Abstract. This paper presents the design of a middleware approach that aims at assisting handheld devices in accessing Grid services by wrapping the computational and resource intensive tasks in a surrogate and shifting them to a capable machine for execution. The performance of the surrogate approach is evaluated with the help of a test scenario. The reduction in computational intensity at the handheld device, achieved through task delegation, is examined and the optimization of communication mechanisms, that reduce the load on a resource constrained handheld device, is presented[1].

1 Introduction

With ever decreasing costs and increasing functionality in small sized chips, mobile handheld devices e.g., Personal Digital Assistants (PDA) and smart phones are becoming mainstream now. While mobile elements will improve in absolute ability, they will always be resource-deprived relative to their static counterparts (desktops/workstations). In [1], 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.

Owing to monotonically increasing mobility of users and greater adoption of handheld devices, job submission to Grid through handheld devices comes up as a viable option for maximizing usability of devices and efficiency of services. Constraints that hinder handheld devices from such interactions include limited network bandwidth, CPU power, memory (small network buffers) and intermittent connectivity. Keeping such 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 [2] Surrogate Architecture [3] which provides a network framework in which a device can deploy a client or a service on a device other than itself.

[1] This work is partially supported by Ministry of Commerce, Industry and Energy, Republic of Korea.

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 [4], [5], 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.

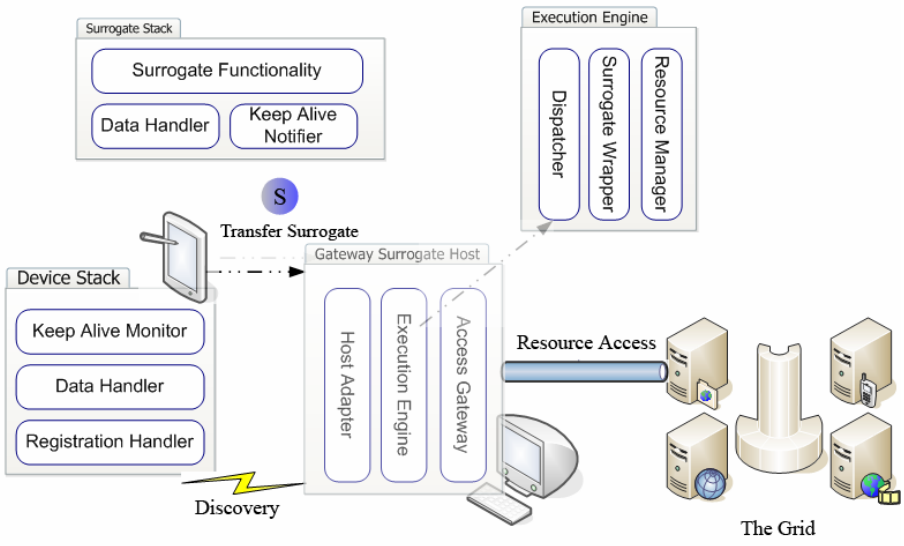

Fig. 1. Interaction between a mobile handheld device with Grid services through middleware deployed at a 'in-between' gateway

The '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" [6], also needs to exist. A handheld device that can communicate on IP (wirelessly or wired) can be programmed to shift its 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 3 – Tier Architecture

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. Consider the example of a physicist who 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 stored in data-stores from which graphs are to be generated will be in the range of several gigabytes or even terabytes. The processing of such data for the purpose of plotting graphs is not a job to be handled by a handheld device. Moreover, the handheld device may have low network bandwidth, further diminishing the prospects of a successful remote analysis by a user. By utilizing the Jini Surrogate Architecture based middleware support, one can 'pack' the functionality of access mechanisms for data-stores and graph plotting routines in a *surrogate* and transfer the *surrogate* to a *host* machine. The *host* machine will provide the *surrogate* with the 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. 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 from its surrogate. The middleware framework consists of three distinct tiers namely Device, Surrogate and Gateway. These are discussed one by one in the subsequent paragraphs.

2.1 Tier 1 – Gateway

Gateway Surrogate Host is the middleware component that aids the Device to overcome resource and computational limitations 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 a device and the host so that both can agree on the transfer of the *surrogate* after authenticating the device and its related surrogate. Once 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 resources include memory and disk space, processor, JVM (for 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 attributes relevant to its properties and capabilities 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, Throughput
- Environment e.g. Java VM availability and version, SOAP/WSDL [7], [8] XML parser etc
- Grid services available through this Surrogate Host

Advertising these attributes allows clients to identify appropriate hosts based on their location, network proximity and other desired features. 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.

2.2 Tier 2 – 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. The functionality of the generic surrogate, as shown in surrogate stack in Figure 1, is incorporated at the top layer along with the specific logic of the extended Surrogate. Moreover, the surrogate has complementary modules for communicating with the middleware stack at the Device. Surrogate can be hosted in the file system of the Device or it can be stored at a URL accessible store e.g. a web server or FTP server. Some clients may be void of any Surrogates. These sorts of clients/devices are still able to use other deployed surrogates if they can provide valid credentials as their rightful owner or users.

2.3 Tier 3 – Device

At the Device, a lightweight middleware stack is provided to facilitate the 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 and selects the *Host* and registers 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 for the application executing at the device. *Surrogate* to be transferred can be stored at the *Device* or at a URL accessible store e.g. a web server or FTP server.

3 Optimizations

There is a critical requirement of clients/devices being able to discover available *Gateway Surrogate Hosts*. A good discovery mechanism is required to avoid single points of failure in the system. 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 finding 'nearby' hosts by the devices (as multicast is geographically contained within a limited 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 query for 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 side and *surrogate* are minimal. Moreover, efficiency also depends on the proximity of *surrogate* to the actual service being accessed. Since a mobile user may be in motion with respect to the *Gateway Surrogate Host* as well as the Grid resources it wants to access, support is needed in the architecture to optimize both proximity based parameters. Each *Gateway Surrogate Host* will keep track of its access quality towards 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 improves runtime performance relative to a scenario where such optimizations are left to good luck.

Table 1. Attributes published by a surrogate host

Name	Description
Host Identification	ID, Location, Network address and Discovery/Listening port for incoming Device/Client requests
Host Resources	Currently Hosted Surrogates, Available/Allocated Resources e.g. CPU, Memory, Storage, Throughput
Host Environment	JVM availability, version; SOAP/WSDL etc
Network Resources	Grid services available through this Host, Proximity to service and client side (in terms of network access)

Table 1 lists the attributes computed and advertised by each host allowing clients to select hosts based on location, proximity and other desired features. The following pseudo-code describes a selection approach for 1) the device to choose a host and 2) the surrogate to select resources:

```
1)  Discover available Surrogate Hosts
        Listen for Multicast Announcements from Hosts
        Query Web Service W for available Hosts
    Select Optimal Gateway Surrogate Host
        For all discovered Hosts
            Retrieve attributes
        Choose best host through function 'f'
    Transfer Surrogate

2)  Retrieve Resource List from Gateway Surrogate Host
        For all known Resources
            Retrieve Resource attributes
    Choose optimal resource
```

In order to elaborate the given algorithms, let D be a set of *Devices* willing to transfer surrogates and let G be a set of available *Gateway Surrogate Hosts*:

$$D = \{d_1, d_2, d_3, ..., d_n\} \tag{1}$$

$$G = \{g_1, g_2, g_3, ..., g_n\} \tag{2}$$

Let R be the resources known to a particular Gateway Surrogate Host g_i that might be of interest to arriving surrogates:

$$R = \{r_1, r_2, r_3, \ldots, r_n\} \tag{3}$$

where R_{gi} will a subset of resources R known to host g_i.

The set A_{gi}, of attributes associated with a Gateway Surrogate Host g_i is listed as follows:

$$A_{g_i} = \{T_{g_i,d_j}, M_{g_i}, C_{g_i}, N_{g_i}\} \tag{4}$$

where $T_{gi,dj}$ represents the network throughput available between the device d_j and a host g_i, M_{gi} represents the available memory resources and C_{gi} represents the average idle CPU availability. Basing on the type of surrogate, a subset of these parameters is chosen to decide the most suitable host for the surrogate of the device.

A device with a CPU intensive surrogate task can choose a *Gateway Surrogate Host*, as follows:

$$g_{sel} = \max_{g_i \in G}\{f(C_{g_i}, N_{g_i})\} \tag{5}$$

where g_{sel} is the Gateway Surrogate Host selected as a function of processing power and number of surrogates hosted to avoid contention for CPU.

Similarly, a number of attributes can be retrieved from job schedulers and resource managers in generic grid infrastructures such as approximate wait time (AWT), network throughput, CPU availability, wait queue length; [9] describes a 'resource utilization status' (RUS) being maintained by a grid computing facilities that indicates resource availability. Attributes associated with each resource r_i include:

$$Ar_i = \{T_{r_k}, C_{r_k}, RUS_{r_k}, \ldots\} \tag{6}$$

where T_{rk} is the network throughput [10] available between the resource and the Gateway Surrogate Host and C_{rk} is the CPU availability at the resource host.

The surrogate can select the resource to access basing on these attributes.

$$r_{sel} = g\{A_{r_i}\} \tag{7}$$

where r_{sel} is the Resource selected as a function over attributes of available resources.

The attributes of a host and resource along with corresponding selection functions, as shown in (5) and (7), help in optimizing access to the resources.

4 Implementation Overview

The authors have provided a bare-bones implementation of the proposed architecture so that before this design is tested for actual Grid service interaction, its viability can be validated in a general scenario. The scenario of choice should involve considerable CPU, memory and network utilization. Simple Network Management Protocol [11] is a widely accepted and utilized way of monitoring network entities. 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 on the handheld device in the forms of dynamic line, bar and pie charts/graphs. Performance of the device and the impact of the running system will be measured and the benefits and shortcomings of the approach will be highlighted.

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

4.1 Analysis

Measurements were taken to analyze the performance of the *Device* during the course of execution. The client application on the PDA consumes fewer than 6 MB of memory at maximum. This also includes the foot print of the J9 JVM and Java AWT classes. Delay during the transmission of results from the su*rrogate* and their display in the form of graphs on the Device were found to be negligible (quite less than 1 second) owing to 100 % signal strength of the wireless connection and CPU availability to client application on the PDA 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.

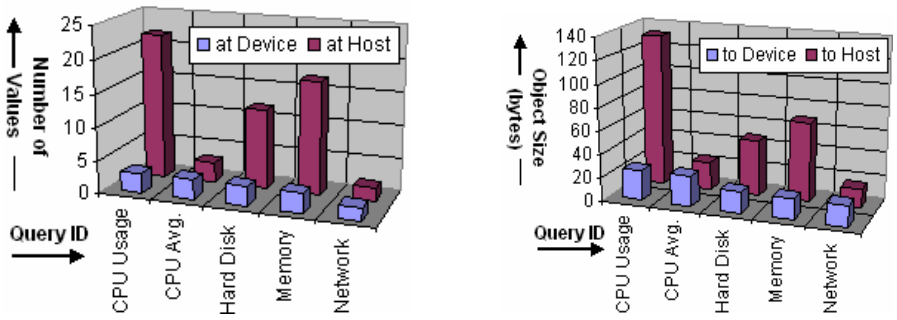

Fig. 2. Left: Comparison between number of values at the *Host* and values sent to the *Device*; **Right**: Comparison between size of intermediate results at the *Host* and size of results at *Device*

It can be observed that the number of parameters is reduced by 75% (4 times reduction) when transferring results to the *Device*. Similarly, more than 64% of the data has been filtered out in intermediate calculations and trimming at the *surrogate*. This performance markup is in addition to the communication and delay reduction achieved by careful selection of host machine and resource access mechanisms during surrogate 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 in enabling handheld held devices to interact with Grid services is in its early stages. Signal [12] 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 effects of a mobile/wireless environment. Also, the proxy schedules the jobs submitted to it by other mobile devices, but in our case as the middleware has far more resources at its disposal, so there is no particular need for scheduling. In [13] mobile agent paradigm is used to develop a middleware to allow mobile users' access to the Grid. It focus's on providing this access transparently and keeping the mobile host connected to the service. GridBlocks [14] builds a Grid application framework with standardized interfaces facilitating the creation of end user services. They state that SOAP usage on mobile devices may be 2-3 times slower than a proprietary communication protocol, but the advantages of using SOAP (such as overcoming device heterogeneity) maybe far more profitable than the effects of this 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.

Optimization of the overhead caused by an additional layer between the source service and the destination device, location based dynamic scalability, and multi-protocol discovery services, are the main focus of the research. 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 working of our framework incorporating 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. Satyanarayanan, M.: Fundamental Challenges in Mobile Computing. In: Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing, Philadelphia (1996)
2. Sun Microsystems, Inc.: Jini™ Architecture specification. http://www.sun.com/jini/specs/
3. Sun Microsystems, Inc.: JiniTM Technology Surrogate Architecture Specification. http://surrogate.jini.org/sa.pdf (2003)
4. S. Vazhkudai, S., Tuecke, S., Foster, I.,:Replica Selectionin the Globus Data Grid. Proceedings of the first IEEE/ACM International Conference on Cluster Computing and the Grid (CCGRID 2001), IEEE Computer Society Press,(2001) 106-113,
5. 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
6. Sun Microsystems, Inc.: Jini™ Technology IP Interconnect Specification. http://ipsurrogate.jini.org (2001)
7. Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., Weerawarana, S.: Unraveling the Web Services Web – An Introduction to SOAP,WSDL, and UDDI. In: IEEE Internet Computing, vol. 6, no. 2,(2002) 86–93

8. Box D., et al. Simple Object Access Protocol 1.1. Technical report, W3C.,http://www.w3.org/TR/2000/NOTESOAP-20000508/ (2000)
9. Shan, H., Oliker,L, Biswas, R.: Job Superscheduler Architecture and Performance in Computational Grid Environments. In: Super Computing Conference 2003 (SC2003), Phoenix, Arizona (2003) 15-21
10. Wolski, R.: Dynamically Forecasting Network Performance Using the Network Weather Service. In: Journal of Cluster Computing, (1998)
11. Stallings W.: SNMP, SNMPv2, SNMPv3, and RMON1 and RMON2. 3rd Edition Addison-Wesley, California (1999) 71-82
12. Hwang, P. Aravamudham Middleware Services for P2P Computing in Wireless Grid Networks. IEEE Internet Computing vol. 8, no. 4, July/August 2004, pp. 40-46
13. Bruneo, M. Scarpa, A. Zaia, A. Puliafito, Communication Paradigms for Mobile Grid Users. Proceedings 10th IEEE International Symposium in High-Performance Distributed Computing, (2001)
14. Gridblocks project (CERN) http://gridblocks.sourceforge.net/docs.htm