On Communication Optimization in Grid Access Middleware for Handheld Devices

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

Computer Engineering Department, Kyung Hee Univerity
1, Seocheon, Giheung, Yongin, Gyeonggi 449-701 KOREA
{maria, saad, anjum, sylee}@oslab.khu.ac.kr

Abstract. This paper describes the optimization techniques for interaction of constrained handheld devices with Grid services by employing the concepts of Jini Surrogate Architecture. Handheld devices in general do not posses enough computational and communicational assets to meet the criteria for utilizing the Grid infrastructure services. We present the design of a middleware approach[1] 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. We analyze the reduction in computational intensity at the handheld device, achieved through task delegation, and present the optimization of communication mechanisms that reduce the load on a resource constrained handheld device.

1 Introduction

With ever decreasing costs and increasing functionality in small sized chips, handheld devices e.g., Personal Digital Assistants (PDA) and smart phones are becoming mainstream now. A broad spectrum of internet services is already becoming available for a mobile user, Grid [1] computing and mobile computing however remain two disjoint phenomena as yet, keeping users of both from 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 [2], 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.

Consider a physicist who needs to see graph plots of data produced as a result of high energy collisions between atoms 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 terabytes. Constraints that hinder handheld devices from such interactions include limited network bandwidth, CPU power, memory

[1] This work is supported by grant No. R01-000-00357-0 from Korea Science and Engineering Foundation (KOSEF).

(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 with inducing minimal burden on the device itself. We demonstrate a solution based on Jini Network Technology's [3] Surrogate Architecture [4] which provides a network framework in which a device can deploy a client or a service on a device other than itself.

Our proposed middleware builds upon the concepts of Jini Surrogate Architecture and ports the functionality of accessing Grid network and services to a handheld device. A handheld device (hereafter referred to as 'Device'), wishing to be a service host or to interact with services hosted elsewhere and unable to do so itself, is allowed to deploy the actual functionality at an intermediate powerful machine and receive the messages/results in a form that is in keeping with its hardware resources. The 'service' or 'client' process, transferred from the device, is called a 'surrogate'. The middleware at intermediate machine, which provides the execution environment and access to extensive resources for the handheld device's surrogate, is called the 'SurrogateHost' or simply 'Host'. An interconnect, defined as "logical and physical connection between the surrogate host and a device" [5], also needs to exist.

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 [6, 7], 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 of and optimization of intermediate results that are to be transferred to the device from the remote machine.

An overview of our middleware approach is presented in section 2. Section 3 deals with the communication mechanisms and the proposed optimizations in the middleware. Prototype implementation and test results are presented in section 4. We conclude our discussion in section 5.

2 The Grid Access Middleware Architecture

The architecture is derived from the fact that any 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 only managing less intensive tasks of displaying the tailored results returned.

Figure 1 shows the proposed middleware framework which consists of three distinct stacks deployed at the GatewaySurrogateHost, the Device and the surrogate. These are discussed one by one in the subsequent paragraphs.

The GatewaySurrogateHost is an extension of the basic SurrogateHost with added functionality for Grid access through the AccessGateway. It overcomes

the major technical hurdles that keep the Devices from exploiting the benefits made available by the computational and data Grids [8, 9] by providing an interface to the Devices on one hand and to the Grid services at the other. The middleware at these hosts consists of three main sub-modules. HostAdapter sub-module enables the initial communication between the device and the host for agreeing on the transfer of surrogate. Once the surrogate is available at the host, it is delivered to the ExecutionEngine sub-module. SurrogateWrapper exposes the functionality of a surrogate required to facilitate its execution at the host, Dispatcher allocates a separate thread for execution, and Resource Manager resolves the resources (e.g., memory, disk space etc) required by surrogate during it's lifetime. The AccessGateway sub-module provides interface to the external resources e.g. discovery of available Grid services and resources. Administrators can restrict the number of surrogates, memory, bandwidth allocation etc on per surrogate basis at the host. Security policies can be configured based on public/private key pairs and digital certificates.

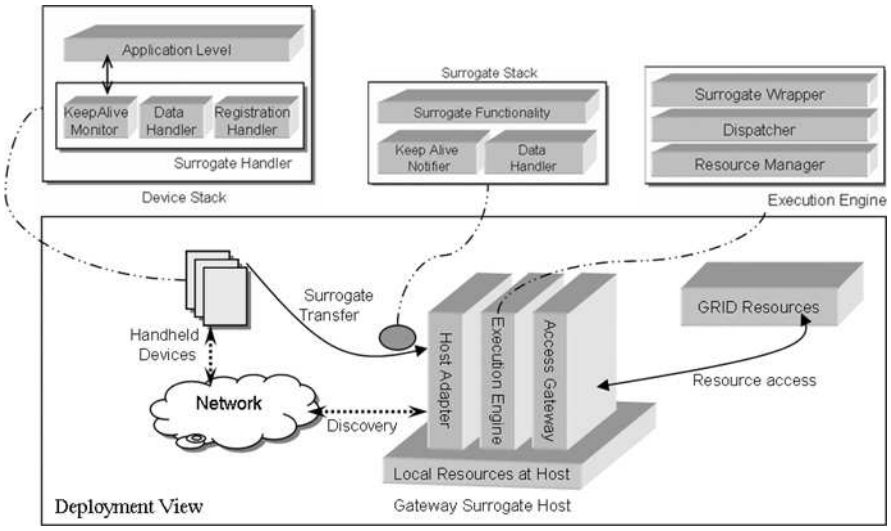

Fig. 1. Proposed Grid Access Middleware Stacks at the Device, Surrogate and Host

At the Device, a lightweight middleware stack is provided for facilitating the registering and communication with its exported surrogate. The stack consists of a SurrogateHandler module which has three sub modules for providing services complementary to the middleware at the GatewaySurrogateHost. RegistrationHandler discovers, selects and registers with the host, and transfers the surrogate. Once the surrogate is transferred, KeepAliveMonitor keeps track of the status of the surrogate. DataHandler retrieves the results from the surrogate-side corresponding module, and makes them available for the application executing at the device which simply displays the results.

A generic surrogate for Grid service access contains interconnect-specific communication mechanism, persistency safe behavior, and migration ability. This functionality is incorporated at the top layer of the surrogate stack. The surrogate has complementary modules for communicating with the middleware stack at the Device. KeepAliveNotifier module sends notifications about the surrogate's status at regular intervals. DataHandler module performs intermediate computations, filtering and trimming of results from surrogate to a format prespecified by the application running 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.

This distributed middleware approach helps the GatewaySurrogateHost to announce its presence, enables Devices to discover available hosts and request to transfer their surrogates, and allows surrogates to communicate status and result information back to their respective devices.

3 Communication Mechanisms and Optimizations

There is a critical requirement of clients/devices being able to discover and communicate with available GatewaySurrogateHosts in an optimal manner. 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 hosts which allows discovery of hosts within the network boundary. Provision for unicast discovery is also incorporated in order to discover a geographically remote host that is closer to a resource of interest. HTTP based discovery is provided as a supplement where all available hosts register with a web service [10] hosted on a known location.

3.1 Attributes Based Dynamic Host and Resource Selection

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 service being accessed. Support is provided in the architecture (refer to Ttable 1) to optimize both proximity based parameters. Each GatewaySurrogateHost will keep track of its access quality towards known/available Grid service hosts/networks.

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 [11, 12] 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 of the host computed and advertised allowing clients to select hosts based on location, proximity and other desired features. This poses a certain one time per start-up burden, but improves the overall runtime performance. Following pseudo-code describes a selection approach for 1) host selection at device and 2) resource selection by surrogate:

```
1)  Discover available Surrogate Hosts
        Listen for Multicast Announcements from Hosts
        Query Web Service W for available Hosts
    Select Optimal GatewaySurrogateHost
        For all discovered Hosts
            Retrieve attributes
        Choose best host through function 'f'
    Transfer Surrogate
2)  Retrieve Resource List from GatewaySurrogateHost
        For all known Resources
            Retrieve Resource attributes
    Choose optimal resource
```

In order to elaborate the selection approach, let D be a set of Devices willing to transfer surrogates and let G be a set of available GatewaySurrogateHosts:

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

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

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

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

where R_{g_i} will a subset of resources R known to host g_i. Set A_{g_i}, of attributes associated with a GatewaySurrogateHost gi is as follows:

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

where T_{g_i,d_j} represents the network throughput available between the device dj and a host gi, M_{g_i} represents the available memory resources and C_{g_i} represent the average idle CPU availability. Basing on the type of the 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 Host, as follows:

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

where g_{sel} is the selected GatewaySurrogateHost 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 e.g. approximate wait time (AWT), network throughput, CPU availability, wait queue length; [13] describes a 'resource

utilization status' (RUS) being maintained by a grid computing facilities that indicates resource availability. Associated with each resource r_i:

$$A_{r_i} = \{T_{r_k}, C_{r_k}, RUS_{r_k}, ...\} \tag{6}$$

where T_{r_k} is the network throughput [14] available between the resource and the GatewaySurrogateHost and C_{r_k} is the CPU availability at the resource host. The surrogate can select the resource to access as:

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

where r_{sel} is selected by applying 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) helps in achieving the first two of the three optimizations we aim to achieve. They not only help in initial selection of host and resource but also help to decide migration pattern of a surrogate as discussed in the following section.

3.2 Surrogate Migration

The Generic Surrogate is equipped with the ability to migrate. This provision is given for those Surrogates which have to access geographically distributed services. It serves the interest of efficiency that services be accessed from locations where delays can be minimized. In addition to this, a GatewaySurrogateHost can become over-burdened and can request the surrogates to pack and leave, instead of terminating them. Surrogate migration can be achieved by employing the DIAMOnDS [15] system which provides a framework for code migration by using the marshalling support in the Java programming language. The decision of the surrogate to migrate is based on one or more of the conditions described in Table 2.

Table.2. Migration conditions overview

Condition	Cause	Description
A	The resource access quality from current execution host is not optimal	(a) Network bottleneck (b) Resource to be accessed is far away (in terms of network costs)
B	The device to host communication has become slow	(a) The device has moved away (b) Network conditions have deteriorated
C	Current execution host is over burdened	(a)Too many surrogates (b) Excessive CPU utilization (c) Low on memory etc

Condition A. Assuming the surrogate in question is executing on gateway surrogate host g_i it can retrieve the parameter T_{r_k} from g_i which gives the network thorough put between the host and the resource. Other parameters e.g.

RUS_k and C_{r_k} give the information required by the surrogate to deduce if access parameters are below a minimal acceptable value M_A. If resource access quality is below M_A, the surrogate can migrate (8). Condition A can be represented with the following expression:

$$f\left(\{T_{r_k}, C_{r_k}, RUS_{r_k}, ...\}\right) < M_A \tag{8}$$

where f is a application specific function that computes current access quality basing on available resource attributes. Applications will define different weights to the attributes over which f is computed depending upon the importance of each attribute to the task being performed by the surrogate.

The requirement of migration may not only be due to poor access quality but also because the surrogate has finished accessing resource X and now wants to access resource Y which can be better accessed from some other host. In such a case:

$$\begin{aligned} &for\, G\,[1-n] \\ &\quad choose\, max\left(f\left(\{T_{r_k}, C_{r_k}, RUS_{r_k}, ...\}\right) > M_A\right) \\ &migrate\left(g_i\right); \end{aligned} \tag{9}$$

Condition B. A mobile handheld device may move away from the surrogate during the course of surrogate's execution. In case dedicated connectivity is required for task execution, the surrogate, upon explicit notification from the device, can be migrated to a host that is closer to the device's new location. The attribute set A_{g_i} gives the attributes associated with the device and the host on which its surrogate is executing. The surrogate may also migrate if it finds that the communication quality with the device over a specific interconnect is below the minimal acceptable value M_B.

$$Migrate(g_i); \tag{10}$$

$$\begin{aligned} &if(T_{g_i,d_i} < M_B) \\ &\quad for\, G\,[1-n] \\ &\quad\quad choose\, max(T_{g_i,d_i} > M_B); \\ &\quad migrate\left(g_i\right); \end{aligned} \tag{11}$$

Condition C. The surrogate can query the current execution host and its load history as shown in (4). The host can either ask the surrogate to migrate or the surrogate can compute that execution environment attributes have fallen below the minimal acceptable level M_C. If the surrogate has to take the decision itself, the condition it will monitor is:

$$if(T_{g_i,d_i}, M_{g_i}, C_{g_i}, N_{g_i}) < M_C \tag{12}$$

where f is a case specific function over host attributes used by the surrogate to determine host's execution environment quality.

4 Implementation Overview

Before this design is tested for actual Grid service interaction, it is necessary to validate its viability in a general scenario. The scenario of choice should involve considerable CPU, memory and network utilization. Simple Network Management Protocol [16] 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, after every few seconds. Handheld device has network connectivity through a wireless LAN interface. A desktop machine is configured to act as a GatewaySurrogateHost. 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.

A GatewaySurrogateHost module has been implemented by modifying and extending the Surrogate Host provided with the reference implementation of Jini Surrogate Architecture specification. 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. Moreover, it contains the 'device-to-surrogate' interconnect implementation which, in the case of this scenario, is based on IP Interconnect Specification.

4.1 Measurements and Result Optimization

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 in transmission of results from the surrogate 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. Table 3 sums up the measurements.

Table.3. Result parameter count and size comparison at GatewaySurrogateHost 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 GatewaySurrogateHost) 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.

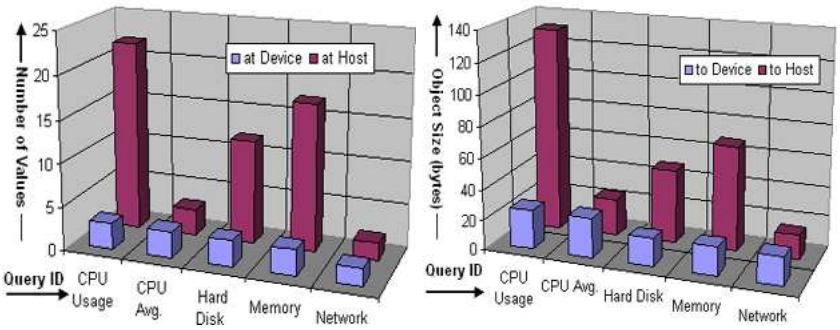

Fig. 2. Left: Comparison between number of values at the Host and values sent to the Device; Right: Comparison between size of intermediate results at the Host and size of results at Device.

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 reduction achieved by careful selection of host machine and resources access mechanisms during surrogate lifetime, as explained earlier. The burden on PDA has really been reduced to a few hundred bytes of data and graph formation.

5 Conclusion

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 efficiently be 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. Target Grid services have been identified and work is under progress to deploy and use it in a real world scenario. Some issues that have been ignored in the test bed implementation will be integrated into the architecture. These include HTTP discovery, client authentication, and surrogate verification and migration support.

References

1. Foster, I.: What is the Grid? A Three Point Checklist. In: GRIDToday (2002)
2. Satyanarayanan, M.: Fundamental Challenges in Mobile Computing. In: Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing, Philadelphia (1996)
3. Sun Microsystems, Inc.: Jini^{TM}Architecture specification. http://www.sun.com/jini/specs/
4. Sun Microsystems, Inc.: Jini^{TM}Technology Surrogate Architecture Specification. http://surrogate.jini.org/sa.pdf (2003)
5. Sun Microsystems, Inc.: Jini^{TM}Technology IP Interconnect Specification. http://ipsurrogate.jini.org (2001)
6. 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
7. 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
8. Lee, S., Gerla, M.: Dynamic Load-Aware Routing in Ad hoc Networks. Proceedings of The Third IEEE Symposium on Application-Specific Systems and Software Engineering Technology (ASSET 2000), Richardson Texas (2000)
9. Godfrey, B., et al.: Load Balancing in Dynamic Structured P2P Systems. In: IEEE INFOCOM 2004, Addis Ababa, Ethiopia (2004)
10. 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
11. Box D., et al. Simple Object Access Protocol 1.1. Technical report, W3C.,http://www.w3.org/TR/2000/NOTESOAP-20000508/ (2000)
12. Chiu, K., Govindaraju, M., Bramley, R.: Investigating the Limits of SOAP Performance for Scientific Computing. In: The Eleventh IEEE International Symposium on High Performance Distributed Computing (HPDC-11), Edinburgh International Conference Center, Edinburgh, Scotland (2002)
13. 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
14. Wolski, R.: Dynamically Forecasting Network Performance Using the Network Weather Service. In: Journal of Cluster Computing, (1998)
15. Shafi, M.A., et al.: DIAMOnDS - Distributed Agents for Mobile and Dynamic Services. In: Computing in High Energy and Nuclear Physics (CHEP2003), La Jolla, California (2003)
16. Stallings W.: SNMP, SNMPv2, SNMPv3, and RMON1 and RMON2. 3rd Edition Addison-Wesley, California (1999) 71-82