Design and Implementation of a Wireless Grid based on WSRF.NET

Abbas Riazi, *Student Member, IEEE,* Hadi Shahriar Shahhoseini

***Abstract* — The use of grid computing has been successfully applied in many areas of computationally intensive applications including scientific research, bioinformatics, weather prediction and other topics. Rapidly growing wireless devices such as cellular phones, PDA, notebooks, tablet PCs, handheld devices force us to prepare new applications based on globally prepared standards such as OGSI/OGSA or WSRF for wireless grids. In this paper, we implement client/server applications based on WSRF.NET. The server application manages clients and gives them all required data and algorithm and schedule tasks based on client resources such as CPU power or memory/storage capacity.**

***Keywords* — Wireless Grid, WLAN Grid.**

I. Introduction

GRID computing consists of heterogeneous computing platforms sharing their resources for a common goal. Grid computing originally employed for processing of massive amounts of data such as weather prediction, bioinformatics, cosmos signal computations and other topics. In grid networks, resources are distributed geographically. Any nodes in grid has its own resources and the heterogeneous nature of grid, required a system for managing nodes and scheduling jobs.

In traditional grids, each node connected to other nodes using wired network. Today by growing amount of wireless devices, connections made by WLAN or other wireless technologies such as WiMAX, GPRS, UMTS and so on. Wireless grid has many similarities to the traditional grids; however they must follow common protocols and standards that communication between wired and wireless nodes can be done without any problem.

Wireless grid has many similarities with wired grids but there are some challenges we faced. Wireless nodes enter to/leave from network frequently that extend their distributed nature and have restrictions in processing power, memory, storage capacity, battery life and bandwidth [1]-[3]. The nodes that created wired grid are generally stationary and connected through cable/fiber based networks, devices in wireless grid, are generally viewed as their simple movements and mobilities.

Wireless grids could be used for various applications and purposes that we can group them into three main categories [2], [4], [5]:

a) Computational grids (clients focuses on computations. They get data and the functions that must be applied on them and get back results)
b) Sensor grids that must gather information/data from the environment using their sensors. They have little processing power and only do simple processing on raw data. Server has this duty to filter unusable data and process informative data.
c) Data grids that its task is data sharing between nodes.

This paper introduces a software implementation of latest standard of grid (WSRF) using Microsoft .NET Framework and SQL Server 2008 as database server for storing state of web services, job definitions, job progress, client management and other required stuffs. The software focuses on the first type of grid. In other hand clients process incoming data and get back results. So the software does not suitable for other types.

Later sections describe the features of the software that distinguished it from previous works. In brief we can say:

a) Sending job progress (how much process successfully done) from clients to server asynchronously.
b) Job scheduling based on node capabilities (processing power, storage, bandwidth, battery life)
c) Nodes can be enter/leave the network as much as they want. This will not disturb progress of entire work.

The result shows that by growing the number of devices that contribute to process, time of progress will be decrease.

II. RELATED WORKS

A number of research, projects and papers are dealing with wireless grids. In [2] the authors use middleware approach to provide peer-to-peer operation but the approach does not use widely used OGSA/OGSI standard. In [6], [7] Humphrey and his colleagues proposed an implementation of OGSI based grid using .NET Framework. As we expected, the architecture used in OGSI.NET uses web services for communication between clients and server. Humphrey also has another implementation that integrates mobile devices into Legion grid computing system [10].

Gonzalez-Castano employed mobile devices into Condor as client front-end for job submission and job querying to traditional grid networks [9].

Abbas Riazi is MSc. student of secure telecommunication in Iran University of Science and Technology, Narmak, Tehran, Iran (phone: +98 912 111 6377; e-mail: a.riazi@ieee.org).

H. S. Shahhoseini is assistant professor in Electrical Engineering department of Iran University of Science and Technology, Narmak, Tehran, Iran; (phone: +98 21 77451500, e-mail: hshsh@iust.ac.ir).

In [8] authors used proxy-based architecture to allow the utilization of various mobile devices in the form of a single virtual cluster. In their approach, instead of building the underlying platform with an exhaustive set of features, they created a lightweight platform with minimal features. Like the previous one, their approach didn't follow OGSI [11] or WSRF rules.

III. DESIGN GOALS

Before describing our work in depth, we outline goals and requirements. First of all, we wished to provide a new platform based on latest open standard. In this way, we choose WSRF [12]. WSRF as new generation of grid standard has features that are absent in previous releases of OGSI. The most are:

a) Notification. Notification is a way for server to find how progress of assigned task goes on. By notification service, client reports status of work to server periodically.
b) Service attributes
c) Security issues

Second the desired architecture should implement a platform for better incorporating of wireless devices as well as supporting traditional non wireless devices and networks (such as servers and workstations).

Third, the architecture should operate on most hardware devices not only smart phones or PDAs. (We focused on devices that support Windows CE or Windows XP Embedded as well as other Windows based machines that run .NET Framework).

Fourth, the platform must address the restrictions of wireless devices. In other hand it must be cover issues that related only to wireless/mobile devices such as network connectivity, availability, absence/presence of device in network, issues related to energy (battery life), signal quality and bandwidth, processing power, storage (RAM/Disk), network latency and so on.

After identifying design goals, we translated goals to requirements. First communication between devices must be implemented using open and widely used protocols. The preferred mechanism is Open Grid Services Infrastructure (OGSI) and its recent developments WSRF [8], [9]. We implement our software on top of .NET framework and Web Service Enhancement (WSE 3.0) released by Microsoft. Combining .NET Framework classes and WSE, cuts down developing time and increase productivity.

IV. ARCHITECTURE

In respect with Globus toolkit, the main goal of designing WSRF was implementing service state. WSRF is based on OGSI with a little difference that there is no need to change anything when using web service tools and is very close with today web service standards [13].

Our software implementation uses WSRF.NET that is based on .NET Framework and ASP.NET. So from server perspective we need IIS as web server (any open source or close source software application that can compile ASP.NET assemblies, can be used as web server). Figure 1 shows flow graph of our implementation.

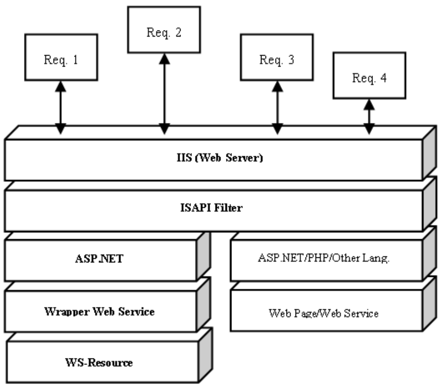


Fig. 1. Flow graph of implementation

As seen in figure 1, requests forwarded to ISAPI filter by IIS and if the request is calling a web service, it redirects to desired web service and if this web service is type of WSRF, all required data will be saved or retrieved from WS-Resource.

WS-Resources defined in WSRF as stateful resources. Stateful means state of resources saved between web service calls [13]. Saving state of resources will be done using SQL Server as database server.

From developers point of view stateful resources could be implemented using simple attributes that embed in code. The attribute is [Resource] that must be writing before defining resource variable.

V. IMPLEMENTED SOFTWARE

In this paper, we want to propose a WSRF.NET based software for running on wireless devices. Based on this assumption, the software must comply with following:

a) Discover wireless nodes and based on their capabilities, assigns tasks.
b) Coordinates unlimited number of clients and asynchronously get progress of completion of job
c) Supporting enter/leave of wireless devices

So software must enable hybrid wired/wireless devices to share their computational resources in order to solve a resource intensive task. In this way, task broke into sub tasks and using a job scheduling mechanism, forwarded to registered devices. Devices start computing algorithm on data sent by server. Any progress of completion of sub tasks, get backs to server by WS-Notification.

The software package has three parts:

a) Grid server that has duty of scheduling jobs, storing state of web services, storing results get backed from clients and managing clients. Server is implemented based on WSRF.NET and using ASP.NET (C#.NET). Data is stored in SQL Server 2008. The server has no GUI and contains only web services that used by clients.

b) Grid client that presents itself to server gets required data and information from server and do computing tasks. After completing task, gets back results to server. We implemented the client in two versions, one with GUI and one without GUI (an application that runs in background, like a service in Windows). The GUI version gives us better view of work progress and control.

The client could be run simultaneously on various devices. Tasks are defined by server and assigned to clients. Managing the clients will be done by server.

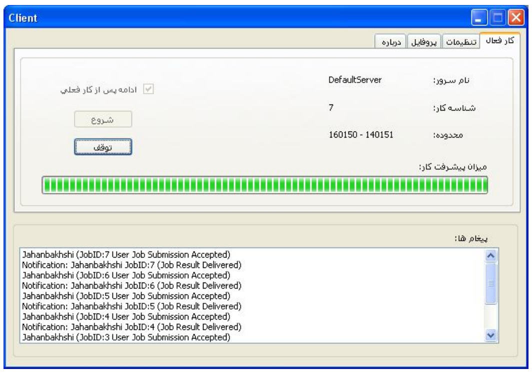
c) Server monitor that represent progress of whole the task. So we can see how clients work, how many clients participate, how much of work is done, how much time is spent for every task and so on. Server monitor is used to present graphical statistics of sub tasks and state of clients. It extracts all stored parameters from database.

VI. PROBLEM STATEMENT

For having better view of how software solves the problem, we define computing prime numbers between one and one hundred million. For this reason, the range divided into sub ranges and every sub range sends to free client. After computation, results sent back by client to server. Results saved in database for further usage.

Before doing any task, client must register itself to server. Registration will be done by sending valid username/password to server. Administrator of system must define all username/password of clients in database. After successful authentication, server lists client as free and if any sub task is ready for submission, it schedules it for client.

Figure 2 shows a client during computation. In the first tab, server name, task ID, number sub range and progress of task can be seen.



نام سرور:	DefaultServer
شناسه کار:	7
محدوده:	160150 - 140151

میزان پیشرفت کار:

پیغام ها:

```
Jahanbakhshi (JobID:7 User Job Submission Accepted)
Notification: Jahanbakhshi JobID:7 (Job Result Delivered)
Jahanbakhshi (JobID:6 User Job Submission Accepted)
Notification: Jahanbakhshi JobID:6 (Job Result Delivered)
Jahanbakhshi (JobID:5 User Job Submission Accepted)
Notification: Jahanbakhshi JobID:5 (Job Result Delivered)
Jahanbakhshi (JobID:4 User Job Submission Accepted)
Notification: Jahanbakhshi JobID:4 (Job Result Delivered)
Jahanbakhshi (JobID:3 User Job Submission Accepted)
```

Fig. 2. Client in middle of computing

Second tab is used for setting username and password as well as IP address and port of client for bidirectional communication (notification).

As described in section V, server has no GUI. For this reason, we implemented GUI based software that shows graphical statistics of whole task, clients, time of every sub task and so on. In addition of showing online clients (those who login successfully to server), software shows scheduled tasks, their ID and clients that run the tasks.

First tab of server is used for listing online clients and showing notifications/messages of clients, including scheduled task, client request and client delivery of task.

Second tab is consisting of parameters of task. For example the range of numbers that prime numbers must be computed, step of computation and pointer of current task. The third tab shows three graphical statistics (Fig 3):

- The time that is spent for every sub task in millisecond. Axis x is task ID and axis y is time.
- Second chart (left down) shows percent of participation of every client while working on whole task.
- Third chart (right down) shows number of tasks that completed successfully by any client. Axis x is number of tasks.

Remember that because of difference of capabilities (processing power, bandwidth, battery life, storage, …) in clients, the time required for completing task is differ from client to client. So in the first chart, time of every task is differing from other. Weaker devices spent more time to complete the task and stronger devices spent less.

One of advantages of the software is clients can enter or leave in grid while server and other clients are computing. So as soon as entering a client to grid and authenticated by server, new task can be assigned to it.

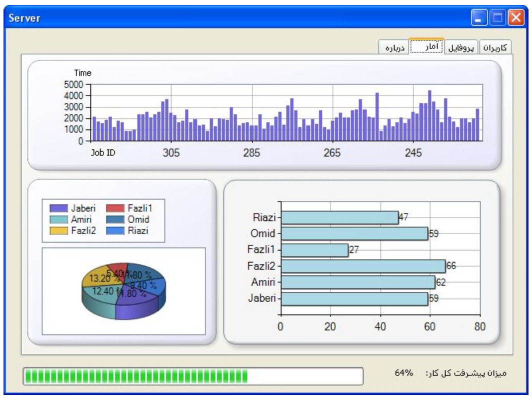


Fig. 3. Statistics tab (every task duration – top – percent of participation of every client – left down – and number tasks completed by every client – right down)

VII. TASK SCHEDULING PARAMETERS

As server decides to assign tasks to clients, it can be done intelligently to increase performance of computation and productivity. In order to do this, server can assign a ranking number to each client and pick the best client when a new task is ready to assign. The server also can reject any client that its rank is below of predefined threshold. So poor clients will be rejected or forwarded to end of queue.

Table 1. Effective parameters for determining rank of clients

Parameter	Direct/ Inverse	Normal Value	w_1	w_2	w_3
Processing Power	D	1 GHz	10	3	3
Storage (RAM)	D	1 GB	10	10	10
Storage (HDD)	D	100GB	4	7	10
Bandwidth	D	1 mb/s	5	5	10
Signal Strength	D	%100	5	10	10
Battery Life Time	D	2 Hour	10	10	10
Enter/Leave frequency into grid	I	1	10	3	10
Availability	D	%100	10	3	10
Connectivity	D	%100	10	3	10
Latency (Ping)	I	300 ms	3	3	10

Table 1 shows effective parameters for computing rank. There are two types of parameters: parameters that have direct relation to increase performance and parameters that have inverse relation. The weight of a parameter for computing rank of client depends on type of grid. So we defined w_1, w_2 and w_3 as weight of every parameter for computational grids, sensor grids and data utility grids. Weights are numbers between 1 (least important) and 10 (most important). Values of weights are our suggestion and may be changed in different applications. Also normal values for parameter can be set by grid administrator; however rank computation follows above rule. For determining rank of clients, server must normalized parameter values of any client. Formula (1-1) is used for normalizing:

$$Normalized\_Parameter = \frac{Parameter\_Value}{Normal\_Parameter} \qquad (1)$$

Then:

$$Rank = \frac{\sum (NP_i) W_i}{\sum W_i} \qquad (2)$$

From (2) rank of client can be determined. Here is an example. Rank of a client with following parameters for a computational grid is as follow:

Processing Power = 3.0 GHz	NP=3
RAM = 4 GB	NP=4
HDD = 100 GB	NP=1
Bandwidth = 56 kb/s	NP=0.05
Signal Strength = %75	NP=0.75
Battery Life Time = 75 min	NP=0.625
Enter/Leave frequency = 15	NP=15 (I)
Availability = %50	NP=0.5
Connectivity = %50	NP=0.5
Latency (ping) = 600 msec	NP=2 (I)

$$\sum (NP_i)(W_i) = 96.42$$

$$\sum W_i = 77$$

$$R_1 = \frac{96.42}{77} = 1.25$$

VIII. PERFORMANCE EVALUATION

For evaluating the performance, we test performance on various machines. For this reason we choose range of 20,000 and 50,000 numbers for every step. Figure 4 shows results. As expected, by increasing number of participants, time of computing will be decreased.

IX. CONCLUSION AND FUTURE WORKS

In this paper we presented an implementation of WSRF based server and client to distribute process on various machines. The software consists of several parts that some them are used for computing and others used for monitoring and getting real time statistics of participants in grid. For having a better view of capabilities of clients that candidate to participate, we suggested a formula to determine rank of client. Based on rank of client and type of the grid, server decides to employ or reject the client. The parameters for ranking are listed in table 1. The importance of parameters is determined respect to the application that runs in grid. The software has this ability to employs as many as clients however time of computing whole task has direct relation with some parameters including processing power, storage, bandwidth, battery life time, connectivity, availability and signal strength and inverse relation with frequency of enter/leave to/from grid and latency.

The software is based on WSRF and we implemented interesting features like WS-Notification, WS-Resource Properties, WS-ResourceLifeTime and WS-ServiceGroup. We will implement WS-BaseFaults in next release. Our future works are around below items:

- Using open standards for multicore processing such as OpenMP on Mobile devices
- Implementing proxy based approach for devices that use other wireless technologies such as GPRS, GSM data link, Ad hoc networks and so on.

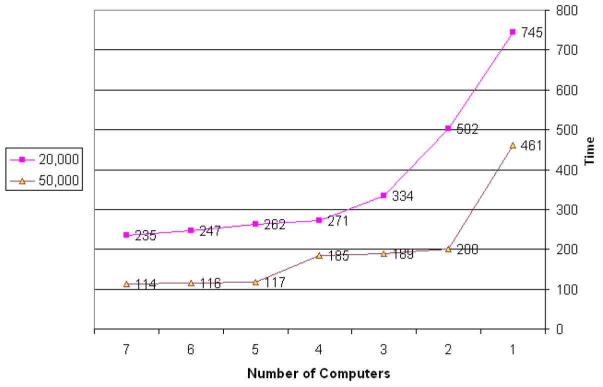


Fig. 4. Performance evaluation for two different steps

REFERENCES

[1] D. Chu, and M. Humphrey. "Mobile OGSI.NET: Grid Computing on Mobile Devices," Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04), Nov. 2004.

[2] A. Agarwal, D. Norman and A. Gupta. "Wireless Grids: Approaches, Architectures and Technical Challenges," MIT Sloan School of Management, Jan. 2004.

[3] T. Phan, L. Huang and C. Dulan. "Integrating Wireless Devices into the Computational Grid," Proceedings of the 8th Annual Intl. Conference on Mobile Computing & Networking, 2002.

[4] L. McKnight, J. Howison, S. Brander, "Distributed Resource Sharing by Mobile, Nomadic and Fixed Devices," Proceedings of IEEE Internet Computing, July-August 2004.

[5] L. McKingh, "Wireless Grid Draft," June 2003. Available: http://www.wirelessgrids.net/docs/draft-ggflwmcknight-wgissues-o.pdf

[6] G. Wasson, N. Beekwilder, M. Morgan, M. Humphrey, "OGSI.NET: OGSI-Compliance on the .NET Framework," Proceedings of the IEEE International Symposium on Cluster Computing and the Grid. April 19-22, 2004, Chicago, Illinois.

[7] M. Humphrey, D. C. Chu, "Mobile OGSI.NET: Grid Computing on Mobile Devices," Proceedings of the 5th IEEE/ACM Intl. Workshop on Grid Computing, November 8, 2004, Pittsburg, PA.

[8] Stavros Isaiadis, Vladimir Getov, "A Lightweight Platform for Integration of Mobile Devices into Pervasive Grids," HPCC 2005, LNCS 3726, pp. 1058-1063, 2005, Springer

[9] F. Gonzalez-Castano, J. Vales-Alonso, and M. Livny. "Condor Grid Computing from Mobile Handheld Devices," Mobile Computing and Communications Review. Vol. 6, No. 2, April 2002.

[10] B. Clarke, M. Humphrey. "Beyond the Device as Porta: Meeting the Requirements of Wireless and Mobile Devices in the Legion Grid Computing System," 2nd Intl. Workshop on Parallel & Distributed Computing Issues in Wireless Networks & Mobile Computing, April 2002.

[11] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maquire, T. Sandholm., D. Snelling, and P. Vanderbilt. "Open Grid Services Infrastructue (OGSI) Version 1.0," Global Grid Forum Drafts. http://www.ggf.org/ogsi-wg. April 2003.

[12] Czajkowski, K., Ferguson, D., Foster, I., Fery, J., Graham, S., Sedukhin, I., Snelling, D., Tuecke, S., Vanbenepe, W., 2004, The WS-Resource Framework. http://www-106.ibm.com/developerworks/library/ws-resource/WS-WSRF.pdf

[13] M. Humphrey, G. Wasson, "Architectural Foundation of WSRF.NET," International Journal of Web Services Research, pp. 83-97, April-June 2005.