Dynamic Resource Allocation in SCADY Grid Toolkit

Rakesh Bhatnagar[1]
MCA Department, SKPIMCS Gandhinagar, Gujarat 382023, India
rakesh_bhatnagar@rediffmail.com

Dr. Jayesh Patel[2]
MCA Department, AMPICS Kherva, Gujarat 382711, India
jayeshpatel_mca@yahoo.com

Nirav Vasoya[3]
Software Engineer, Medusind Solution, Ahmedabad, Gujarat 380015, India
bhulku.niravvasoya@gmail.com

***Abstract* – With the advancement in high performance applications and building of low cost computing resources, resource allocation to tasks in Grid Computing has became one of the most critical job. If resource allocation is done properly, it can lead to high performance throughput.**

This paper analyzes the Windows based Grid toolkit – SCADY (Scalable and Dynamic) and proposes the dynamic resource allocation mechanism for tasks based on the CPU usage and load so that high performance applications can be catered. Idle nodes present in the network are identified and used for computation.

Keywords: **SCADY, CPU Usage, Resource Allocation, Manager, Executor**

I. INTRODUCTION

Scady[1][2] is a Scalable and Dynamic GUI based Grid toolkit developed to serve high performance applications. The objective to create the toolkit is to have a solution to the performance related problems occurring in the present Grid networks.

Scady[1][2] is built for the reason that the demand for the use of distributed resources and computing cycles are increasing day by day. Moreover, the power present in the networks (LANs) is not being utilized properly. Scady is based on Alchemi[10][11] Grid Toolkit which is becoming very popular for catering high performance applications in a grid environment. Alchemi[10][11] is windows based, works on .Net platform and supports all the features that are provided by the platform.

In this paper, it is observed that the resource allocation in the present toolkit works manually. Nodes in a network are selected manually and the tasks are delegated to them in the same fashion. To make the resource allocation dynamic, we have proposed the Resource Allocation algorithm for Scady toolkit that is based on the CPU usage and load using which high performance applications can be catered. According to the CPU usage, nodes are selected dynamically and tasks are delegated to them. This helps in achieving higher performance throughput.
The Paper is organized as follows – Section 2 is literature review, Section 3 describes Scady and its working. Section 4 highlights the problem statement, Section 5 describes the Resource Allocation algorithm that removes the limitations, Section 6 describes the experiment done, Section 7 describes the conclusion and Last Section contains the references.

II. LITERATURE REVIEW

We studied various Grid systems and their functionality. In [4] & [5] authors have identified that there are performance monitoring tools like Ganglia and Autopilot, which are simple and robust It is very easy to integrate Ganglia into other information services. But these tools has several drawbacks and they do not support features such as scalability, substantial input/output activities etc..

In [6], authors have observed that there is lack of appropriate support to handle the failure of key components in most of the existing grid solutions. They support re-configurability to a limited extent.

In [7], it has been identified that presently available middleware does not satisfy the re-configurability requirements of heterogeneous environments. They proposed a new middleware to address the problem. But the issues of memory and flexibility are still to be worked upon.

In [8], it has been identified that as per the requirements, new applications are being designed. Number of failures and fluctuations of load are critical issues. Authors have proposed a middleware infrastructure based on MPI implementation.

In [9], authors have described the security mechanism of Alchemi Toolkit. A QoS based grid resource brokering algorithm is proposed in [12] for high performance parallel applications. Authors have proposed a behavioral modeling of RESTful Web Services using UML in [13].

Authors have identified the need of re-configuration in [14], [17] and [23], architecture is proposed in [15], [16] & [17] and a dispatching system is discussed in [19]. An adaptive middleware for distributed computing is discussed in [20] and [21]. The scalability issue is discussed in [24] & [25].

III. SCADY TOOLKIT

Scady (Scalable & Dynamic) [1][2] is a windows based toolkit developed for Grid systems. It is developed with the objective to provide the features such as multithreading, inter-thread communication, task reallocation in case of node failure etc. Working of Scady is listed out in [1][2].

Brief of Working –
Scady has two components – Manager and an Executor. Both can be installed on all the nodes in a network. The Manager is responsible to delegate the task and retrieve the result. Manager contains the information of the node like its IP address, Port number, and MAC address. All the connected nodes present in the network are displayed. Nodes are selected through checkbox manually. Connection is done to these nodes. And then the application is selected through the manage application tab. Tasks is delegated to these nodes. Executor executes the task and the processed result is sent to the Manager. This working is shown in Figure-I and Figure-II.

Connection is monitored every 10 seconds to check whether the node is available or not. If the node is not available, another node from the list is selected and the tasks are delegated to that node. This is done manually. If due to some reason, a node is not able to process the tasks even when it is available, user can disconnect it explicitly and reallocates the task to another node with the same process as described above.

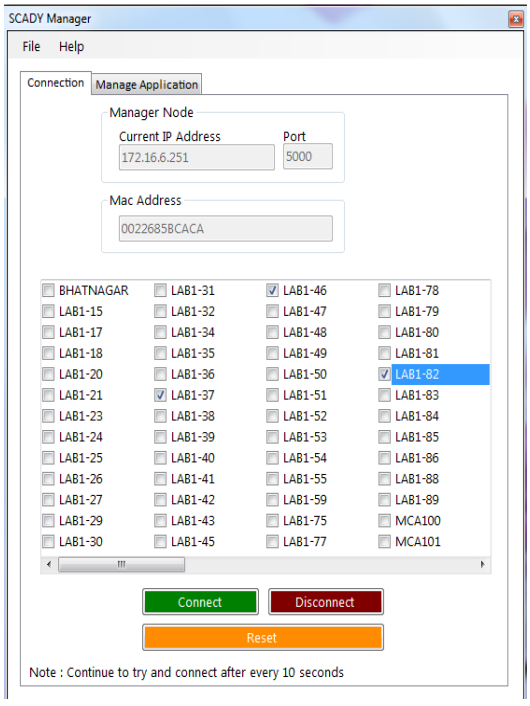

FIG. 1. SCADY MANAGER (CONNECTION)

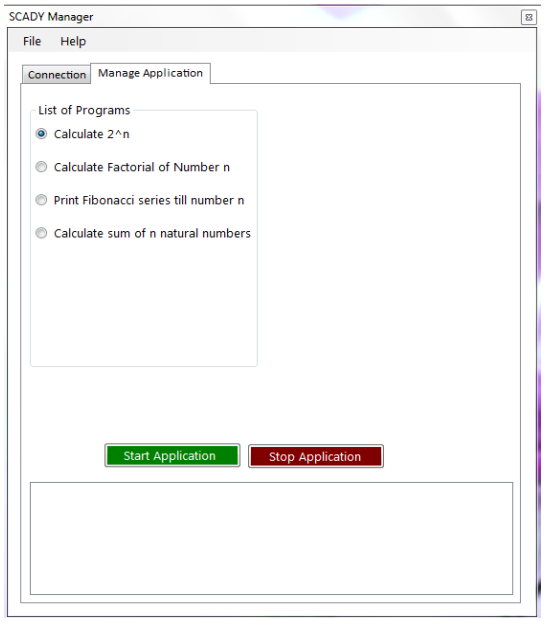

FIG. 2. *SCADY MANAGER (APPLICATION MANAGER)*

IV. PROBLEM STATEMENT

The selection of the nodes is manual. So the tasks may get allocated to nodes that are already loaded. There is no means to know which nodes are less loaded and hence this can affect the performance of the toolkit. It has been also observed that when the nodes are increased considerably, the time taken to complete the task increases instead of decreasing. Also there is no provision for the user to provide the information of number of nodes required to complete the task. User should get the provision of providing the number of minimum and/or maximum nodes for the task.

Hence, there is a need to change the design of the toolkit and to have more functionality in it. There is also a need of an efficient resource allocation algorithm that can help in getting high performance.

V. RESOURCE ALLOCATION ALGORITHM

A slight modification is proposed in the design of Scady. Instead of manual selection of node, automatic selection of node is implemented in Scady. This selection is done dynamically. If the user wants, the number of minimum nodes and maximum nodes can be provided as per the requirement. Minimum nodes to perform a task are kept as three. Presently, the maximum numbers of nodes are kept equal to minimum number of nodes. This can be changed by the user as per the need.

Nodes in the network are found and shown to the user. These nodes are then sorted based on their CPU usage. Nodes with less CPU usage are displayed first. If the user wants, he/she can select the nodes dynamically and then connect to the selected nodes otherwise the first three nodes will be selected. Then the task to be performed is selected from the manage application tab and is delegated to the nodes.

Resource Allocation Algorithm

Input: Application (program) with number of nodes

Steps:

1. Begin
2. Retrieve the list of nodes with their CPU usage. Call it nodelist.
3. If
    - there are no nodes available, show message and exit
4. Else
    - sort the nodes based on their CPU usage (Less CPU usage first)
    - Get the number of minimum and maximum nodes required
    - If
        - user has not changed the number of minimum and maximum nodes, select the first three nodes from the nodelist
    - else
        - get the number of minimum and maximum number of nodes
    - Select the nodes from the nodelist equal to the number of minimum number of nodes.
    - Allocate the tasks to the selected nodes dividing into parallel processes.
5. End

This working is shown in Figure – III.

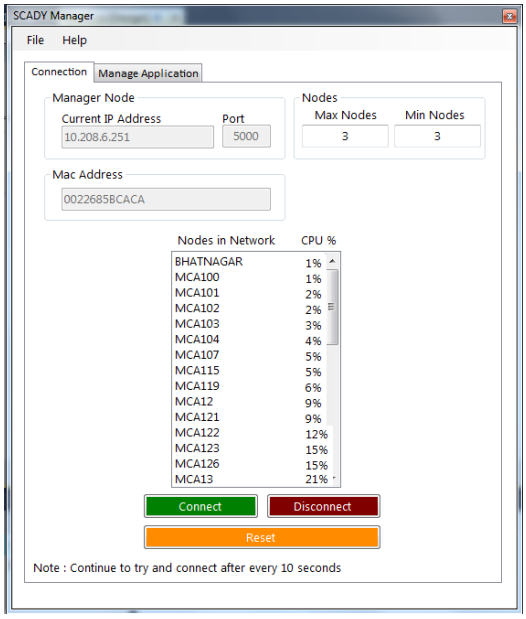

FIG. 3. SCADY MANAGER (CONNECTION)

Executor is running on each node. Executor computes the tasks and sends back the result to the Manager. If Executor is not running on the nodes, Manager is notified for the same.
If due to any reason an Executor goes down in between the task, Manager dynamically reallocates it to another executor and the task is completed. For this the Manager has to dynamically monitor the executors. This is implemented by checking the connection every 10 seconds through the session variable. If a node fails, the task is reallocated to another node by manager based on the CPU usage of nodes. CPU usage is refreshed every 10 seconds.

With this model, node failure and unfinished task problem is solved. Also as the number of managers and executors are more, the time taken to complete the task is less as compared to Alchemi .Net toolkit.

VI. EXPERIMENT

Experiment is done to calculate sum of n natural numbers. The network taken for the experiment is intranet (campus network) in which there are 4 sub-networks, each sub-network having a minimum of 50 client machines. Total computers in the network are around 300. Hardware and Software configuration used are –

- nodes with 3 GB RAM and 4 GB RAM,
- 2.66 GHz,
- Core-to-Dual Processors, Core-to-Quad Processors and i3 Processors,
- 24 ports Network Switch,
- Visual Studio 2012, .Net Framework 4.5 and SQL Server Express Edition.

The test was done with 3 nodes (Executors), then with 10 and then 50 nodes. The working is shown in Figure-IV.

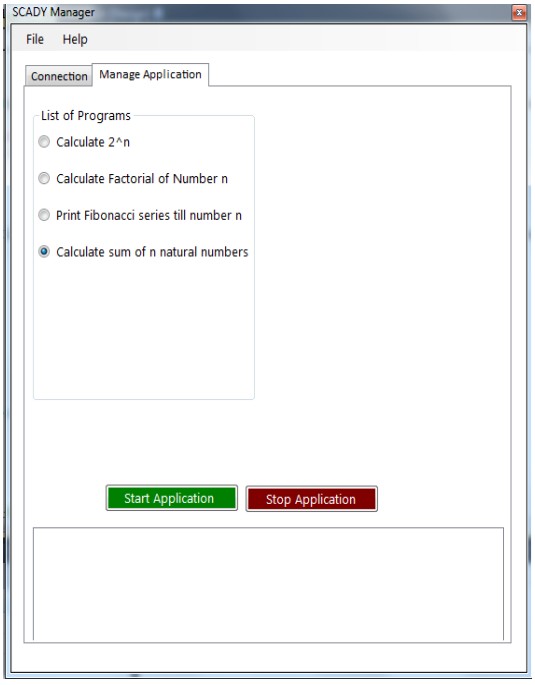

FIG. 4. SCADY MANAGER (APPLICATION MANAGER)

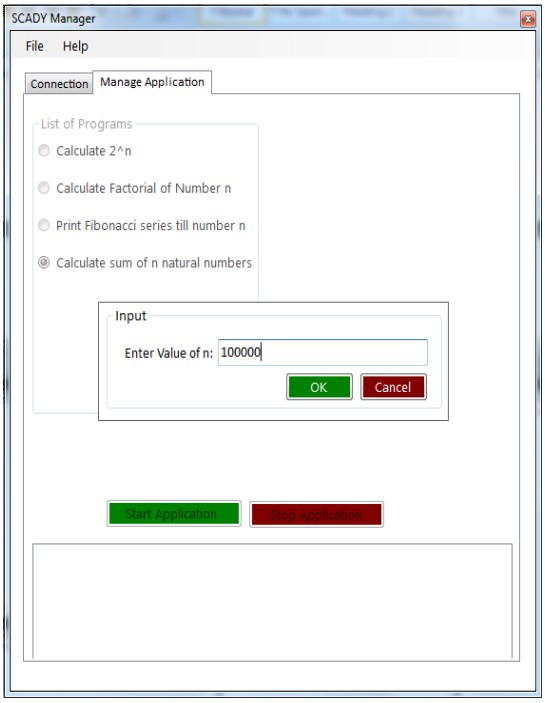

FIG. 5. SCADY MANAGER (FOR SUMMATION PROGRAM)

Task is then delegated to the executors as parallel processes and executors return the result after processing.

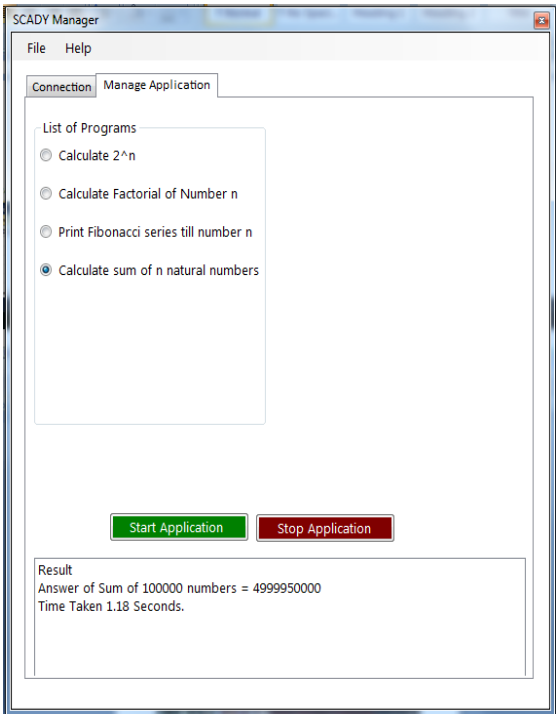

FIG. 6. SCADY MANAGER (FOR SUMMATION PROGRAM)

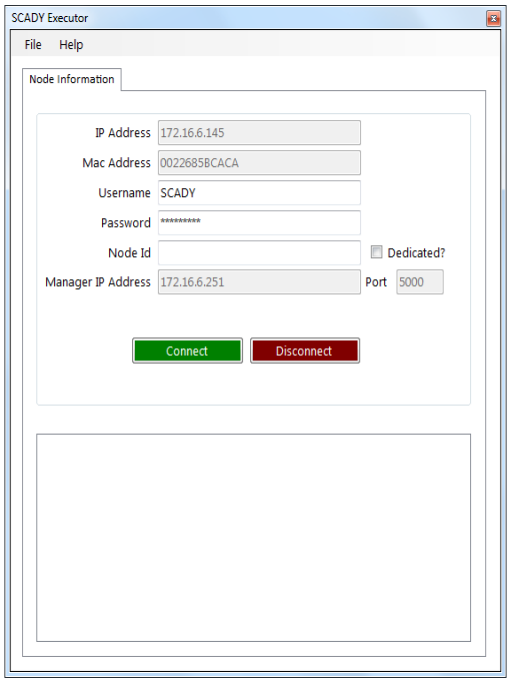

FIG. 7. SCADY EXECUTOR

The result and time taken for calculation is printed in the result window. This experiment is repeated for 10, 20 and 50 nodes also. The results are given in Table-I.

TABLE-I: EXECUTION TIME OF SUM OF N NUMBERS PROGRAM

Number of Executors	Time (in Secs)	
	Alchemi Toolkit	Scady Toolkit
3	1.21	1.18
10	1.19	1.12
20	1.14	1.09
50	0.91	0.82

RESULTS

The results of the experiment of calculating sum of n natural numbers (100000 in our experiment) by using 3 nodes, 10 nodes, 20 nodes and 50 nodes are shown in Figure-VIII.

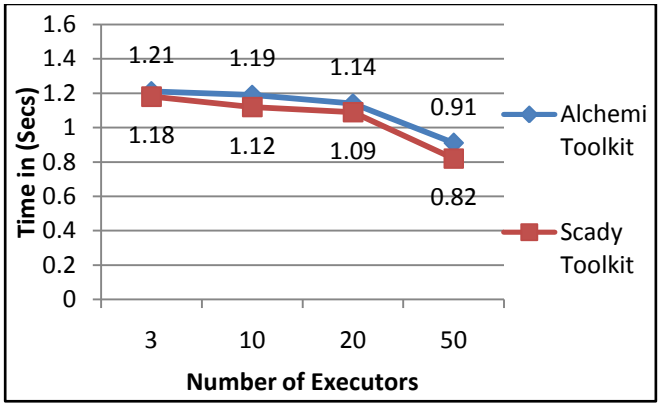

FIG. 8. SCADY & ALCHEMI COMPARISON

It can be noted that the time taken to calculate sum of n natural numbers (100000 in our experiment) decreases when the number of nodes increases.

VII. Conclusion

In this paper, we studied Scady toolkit for Grid Computing. We found that even though, performance of Scady is better in grid computations, there was a need of Resource Allocation algorithm that can help in achieving high performance. To remove this limitation, we made changes in the design of Scady and proposed a Resource Allocation algorithm based on CPU usage. Tasks are allocated according to the CPU usage. This usage for each node is checked every 10 seconds and if a node is found to be failed, the task is allocated to the next available node with less CPU load. We conducted experiment using this Resource Allocation algorithm and found that tasks are now scheduled effectively.

In future, high performance applications like Monte Carlo Simulation, Histogram calculation etc. will be used in Scady and tested.

REFERENCES

[1]. Rakesh Bhatnagar, Dr. Jayesh Patel, "Scady: A Scalable & Dynamic Toolkit for Enhanced Performance in Grid Computing", IEEE International Conference on Pervasive Computing ICPC - 2015, Jan 8-10, 2015, Pune.

[2]. Rakesh Bhatnagar, Dr. Jayesh Patel, "API Specification for Small Grid Middleware - Scady", IEEE Xplore, India Conference (INDICON), 2014 Annual IEEE, DOI:10.1109/INDICON.2014.7030545 Publication Year: 2014, Page(s): 1 - 5.

[3]. Rakesh Bhatnagar, Dr. Jayesh Patel, "An Empirical study of Secur- ity issues in Grid Middleware", International Journal of Emerging Technology & Advanced Engineering , ISSN 2250–2459**,** Volume 4 Issue 1, Jan 2014, Pages 470-474

[4]. Rakesh Bhatnagar, Dr. Jayesh Patel, "Performance Analysis of a Grid Monitoring System - Ganglia", International Journal of Emerging Technology & Advanced Engineering, ISSN 2250–2459**,** Volume 3 Issue 8, Aug 2013, Pages 362-365

[5]. Rakesh Bhatnagar, Dr. Jayesh Patel, "Performance Analysis of a Grid Monitoring System - Autopilot", Journal of Sci-Tech Research (KSV – JSTR) - Volume 4, Issue 2, July - Dec 2013, ISSN 0974 – 9780, Pages 15 – 20.

[6]. M. Venkateswara Reddy, A. Vijay Srinivas, Tarun Gopinath, and D. Janakiram, "Vishwa: A reconfigurable P2P middleware for Grid Computations", http://dos.iitm.ac.in/LabPapers/icppVishwa.pdf

[7]. Hirunavukkarasu Sivaharan, Gordon Blair, and Geoff Coulson, "GREEN: A Configurable and Re-configurable Publish-Subscribe Middleware for Pervasive Computing", On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE, Lecture Notes in Computer Science Volume 3760, Springer 2005, pp 732-749

[8]. Kaoutar El Maghraouia, Travis Desella, Boleslaw K. Szymanskia, James D. Terescob, and Carlos A. Varela, "Towards a Middleware Framework for Dynamically Reconfigurable Scientific Computing", Elsevier, Advances in Parallel Computing, Volume 14, 2005, Pages 275–301 and http://www.cs.rpi.edu/~szymansk/papers/hpc.04.pdf

[9]. Manisha Bhardwaj, Sarbjeet Singh & Makhan Singh, "Implementation of Single Sign-on and Delegation mechanism in Alchemi.Net based Grid Computing Framework", International Journal of Information Technology and Knowledge Management January-June 2011, Volume 4, No. 1, pp. 289-292

[10]. Rajkumar Buyya, Akshay Luther, Rajiv Ranjan, and Srikumar Venugopal, "Alchemi: A .NET-based Enterprise Grid Computing System", Internet Computing, 2005 www.cloudbus.org/papers/alchemi_icomp05.pdf

[11]. Rajkumar Buyya, Akshay Luther, Rajiv Ranjan, and Srikumar Venugopal, "Alchemi: A .NET-based Grid Computing Framework and its integration into Global Grids", www.cloudbus.org/papers/alchemi.pdf

[12]. Aditya Patel, Pratik Thanawala, Dr. J G Pandya, "Grid Resource Brokering for High Performance Parallel Applications", ", International Journal of Emerging Technology & Advanced Engineering, ISSN 2250–2459**,** Volume 3 Issue 4, April 2013, Pages 181-187

[13]. D. M Rathod, S. M Parikh and B.V Buddhadev,"Structural and behavioral modeling of RESTful web service interface using UML", Proceedings of 2013, IEEE International Conference on Intelligent Systems and Signal Processing (ISSP), 2013, pp. 28-33 DOI: 10.1109/ISSP.2013.6526869, Print ISBN: 978-1-4799-0316-0

[14]. Paul Grace, http://www.academia.edu/6142596/A_component based_middleware_framework_for_configurable_and_reconfigurable_Grid_computing

[15]. Yuhui Deng, Frank Wang, Na Helian, Sining Wu, Chenhan Liao, "Dynamic and scalable storage management architecture for Grid Oriented Storage devices", Elsevier, Parallel Computing, Volume 34, Issue 1, January 2008, Pages 17–31

[16]. T. Purusothaman, S. Annadurai, H. Vijay Ganesh, C.T. Chockalingam and B. Uthra Kumar, "Dynamically Scalable, Heterogeneous and Generic Architecture for a Grid of Workstations", Journal of Grid Computing (2004) 2: 239–24, Springer 2005

[17]. Kaoutar El Maghraouia, Boleslaw K. Szymanski, and Carlos A. Varela, "An Architecture for Reconfigurable Iterative MPI Applications in Dynamic Environments", Parallel Processing and Applied Mathematics, Lecture Notes in Computer Science Volume 3911, 2006, pp 258-271 http://link.springer.com/chapter/10.1007/11752578_32

[18]. R. Nagaraja, Dr. G. T. Raju, "COMMPC - Component Based Middleware for Pervasive Computing", IJCSNS International Journal of Computer Science and Network Security, VOL.11 No.9, September 2011

[19]. Gianpaolo Cugola, Gian Pietro Picco, "REDS: a reconfigurable dispatching system", SEM '06 Proceedings of the 6th international workshop on Software engineering and middleware, Pages 9 – 16

[20]. Kaoutar El Maghraoui, Travis J. Desell, Boleslaw K. Szymanski, and Carlos A. Varela, "The Internet Operating System: Middleware for Adaptive Distributed Computing", International Journal of High Performance Computing Applications, Vol. 20, No. 4, Winter 2006, pp. 467-480

[21]. Carlos A. Varela, Paolo Ciancarini, Kenjiro Taura, "Worldwide computing: Adaptive middleware and programming technology for dynamic Grid environments", Journal Scientific Programming - Dynamic Grids and Worldwide Computing, Volume 13 Issue 4, October 2005, Pages 255-263

[22]. Coulson, G., Blair, G.S., Clark, M., Parlavantzas, N., "The Design of a Highly Configurable and Reconfigurable Middleware Platform", ACM Distributed Computing Journal, Vol 15, No 2, pp 109-126, April 2002.

[23]. Kon, F., Costa, F., Blair, G.S., Campbell, R., "The Case for Reflective Middleware: Building Middleware that is Flexible, Reconfigurable, and yet simple to Use", CACM, Vol. 45, No. 6, pp 33-38, 2002.

[24]. Geoffrey Fox, Shrideep Pallickara, and Xi Rao. A scaleable event infrastructure for peer to peer grids. In JGI '02: Proceedings of the 2002 joint ACM-ISCOPE conference on Java Grande, pages 66.75, New York, NY, USA, 2002. ACM Press.

[25]. A. Carzaniga, D.S. Rosenblum, and A.L. Wolf "Achieving Expressiveness and Scalability in an Internet-Scale Event Notification Service", 19th ACM Symposium on Principles of Distributed Computing (PODC2000), Portland OR. July, 2000