Implementation of Caching Algorithm in Scady Grid Framework

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

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

[3]Snehal Rindani
MCA Department, SKPIMCS Gandhinagar, Gujarat 382023, India
Snehal.rindani@gmail.com

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

***Abstract* – Demand for High Performance Applications has already grown in present scenario. Grid Computing and Cloud computing are the requirement of today and tomorrow. Many UNIX based and command based Grids are already in use. But people generally prefer window based machines. Considering this situation, we created a Grid framework called SCADY (Scalable & Dynamic) which is based on Alchemi.Net Grid Framework. From the literature review and findings it has been observed that a better fault tolerance mechanism is needed in these grid frameworks to ensure execution efficiency.**

This paper proposes a caching algorithm for better fault tolerance in Scady Grid Framework. The proposed algorithm uses the cache write-through method in which at the same time the data is written into the cache block and the corresponding main memory location. The data cached can be retrieved very fast on demand, while it is ensured that nothing will get lost if a crash, power failure, or other system disruption occurs as the same data is in main memory. Write through method is the most preferred method for storing the data for the applications where data is critical and data loss cannot be tolerated.

The algorithm is implemented and tested for efficiency. It is found that it gives comparatively better result.

Keywords: **SCADY, Caching, Write-Through, Manager, Executor**

I. INTRODUCTION

In present day scenario, computations are no longer simple. With the advancements of computing resources, industries are moving towards high performance computations. Scady [2][3] is a grid framework designed for serving high performance computations. It provides a window based GUI for execution. Scady is based on Alchemi.Net [7][8] grid framework. Alchemi.Net grid framework is gaining popularity in high performance computations as it is window based. Alchemi .Net is an open source software framework that allows using the computing power of networked machines very flexibly and developing applications to run on grid. After analyzing it, it is observed that there are some limitations to it such as fault tolerance is not addressed and there is no check pointing mechanism. It has been also observed that the present Scady framework also does not contain any check pointing mechanism.

Scady grid framework uses the same Manager-Executor model as used by Alchemi.Net framework. But there is no owner node in Scady. There can be more than one Manager Node in Scady. These Manager nodes can perform task by distributing it to the executor nodes.

Dynamic resource allocation [1] is used 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. Even if an executor node or manager node fails, the task is allocated dynamically to the next available node. To save time and effort, the results are written in cache and when the task is allocated to the new node, result from cache is read and the execution starts from the point where it stopped.

In this paper, authors have proposed the caching algorithm for Scady Grid Framework. The organization of the Paper is as follows – Section 2 presents Literature review; Section 3 describes the problem formulation; Section 4 describes the caching algorithm; Implementation, Results and result analysis are discussed in Section 5; Section 6 describes the conclusion, Section 7 describes the limitation and Last Section contains the references.

II. LITERATURE REVIEW

Achemi.Net framework [7] [8] is an open source software framework that helps in using the computing power of the nodes present in the Grid network. It gives support in developing applications to be executed on the Grid. It is the first .Net based GUI grid but is still open source. It has been observed after analyzing the Alchemi.Net framework that it is still not strong in providing fault tolerance. Fault tolerance is the property of a system that helps it to continue operating consistently with its specifications even in the event of failure of some of its parts. From a user's point of view, a distributed application should continue to operate despite failures.

Alchemi uses heartbeat mechanism [9] to reschedule or reallocate the task in case an executor node fails. In this mechanism, executor nodes send heartbeat messages to the manager nodes at specified intervals so that manager node can get the status of executor nodes. In case the manager node

does not receive any heartbeat message from an executor node, the executor node is considered as dead. The reason can be hardware failure, rebooting of operating system or stopping of the process. Manager stores the status of the task in the SQL Server database by using the thread-id & application-id. It reschedules the task to other nodes by using this information. But there is no check pointing mechanism used. So when the task is rescheduled, it starts from its beginning. This results in comparatively high task completion time. There should be a short rollback distance in case of task failure.

In [10], authors have identified that although the tasks are reallocated to nodes in case of node failure but presently there is no mechanism available with which a task can be executed from the point where it stopped in Alchemi.Net Grid framework.

In [11], authors have identified that there is a need of distributed memory caching for fail safe computation in grid environment. They have given the file based mechanism but writing and reading from file takes much time.

III. PROBLEM FORMULATION

From the literature review, it has been observed that there is no mechanism for an efficient fault tolerance situation in Alchemi.Net grid framework and Scady grid framework. So, there is a need of some mechanism that can remove this limitation and helps in improving the execution efficiency of applications in grid environment.

Authors have proposed and implemented a caching algorithm with check pointing mechanism to cater the above requirement.

IV. CACHING ALGORITHM

Working of Scady grid framework is given in [1] [2] and [3]. Briefly summarizing it – Managers and executors are installed in the grid network. We can install more than one Manager in the grid setup. When any task is to be done, Manager delegates the task to the Executors. The result is processed by the executor and sent to the Manager.

After the connection is done, application is selected. Presently four applications have been taken. These applications can be increased with the use of database. 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, thread objects are created for the task and the thread objects are delegated to the nodes. This task is processed by the executors and the result is sent to the manager. 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.

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. If a node fails, manager node gets the notification and it reallocates the task to another node based on the CPU usage of nodes. To save time and effort, the results are written in cache in the executor node. The cache write-through method is used in which at the same time the data is written into the cache block and the corresponding main memory location. The data cached can be retrieved very fast on demand, while it is ensured that nothing will get lost if a crash, power failure, or other system disruption occurs as the same data is in main memory. Write through method is the most preferred method for storing the data for the applications where data is critical and data loss cannot be tolerated.

Write through method minimizes the risk of data loss. But every write operation must be done twice – one in cache block and one in memory. This redundancy results in longer time. The active application program must wait until each block of data has been written into both the main memory and the cache before starting the next operation. When the task is allocated to the new node, result from cache is read and the execution starts from the point where it stopped. Level-2 Cache Architecture is used which is shown in Figure – I.

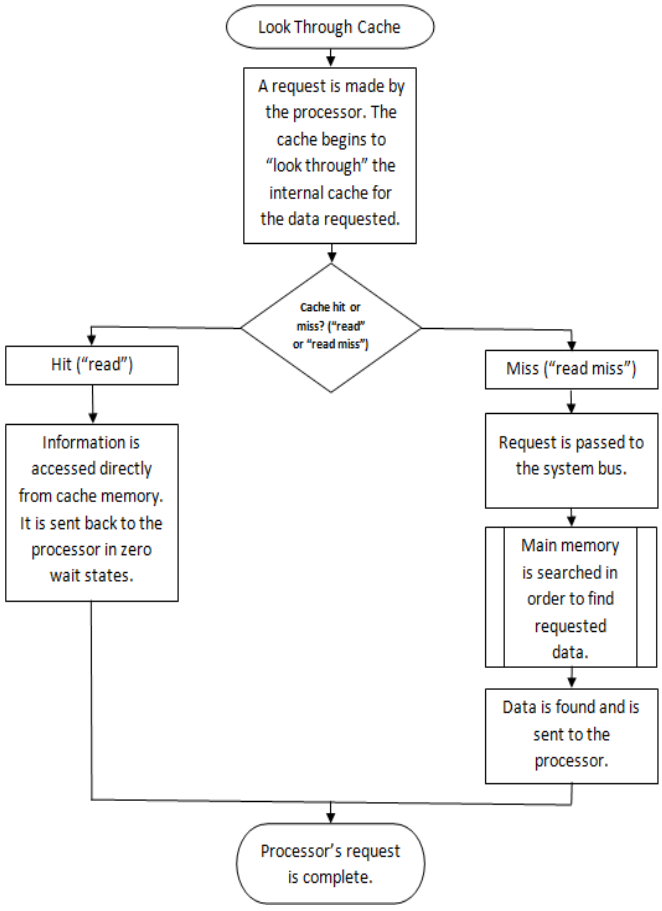

FIGURE-I: WRITE-THROUGH CACHE
Source: http://oharanetworks.com/spohara/level-2-cache-controller

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. For this we performed two experiments which are described in the next section.

Caching Algorithm in Scady Grid Framework

Input: Application (program) with number of nodes

Steps:

1. Begin
2. Check the existence of executor nodes in the network.
3. Retrieve the list of nodes with their CPU usage. Call it nodelist. Sort the nodes based on their CPU usage (Less CPU usage first)
4. Allocate the tasks to the nodes dividing into threads. A thread-id is associated with each thread.
5. At the executor node – intermediate results are written into cache using write-through method. This method stores the result in cache block as well as the main memory.
6. Contents of cache are transferred to manager nodes at specific intervals. Here concept of cache write-back method is used.
7. In case of node failure, the last transferred value of cache is taken and the thread with same id is allocated to another node of the network with less CPU usage.
8. The node starts the execution with the cache result.
9. When the execution gets completed, the results are given to the manager node.
10. End

V. IMPLEMENTATION, RESULT & RESULT ANALYSIS

To implement Caching algorithm, authors took the problem 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 as under –

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

The test was done with 3 nodes (Executors) initially, then with 10, 20 and then 50 nodes. The working is shown in Figure-II.

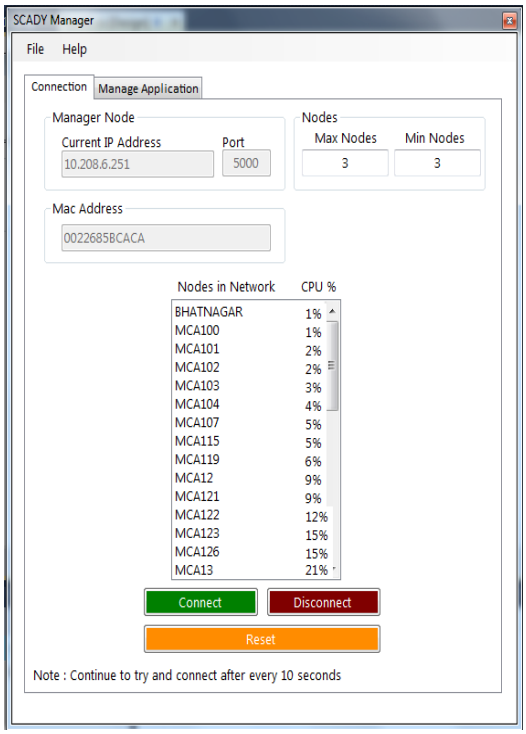

FIG-II: SCADY MANAGER (CONNECTION)

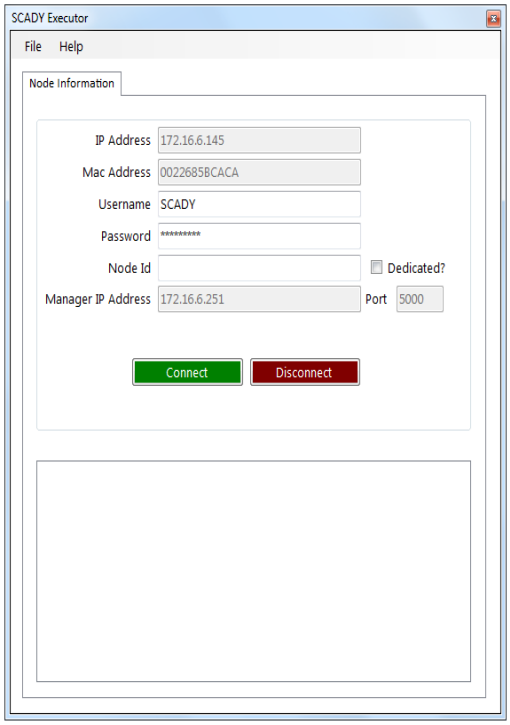

FIG-III: SCADY EXECUTOR

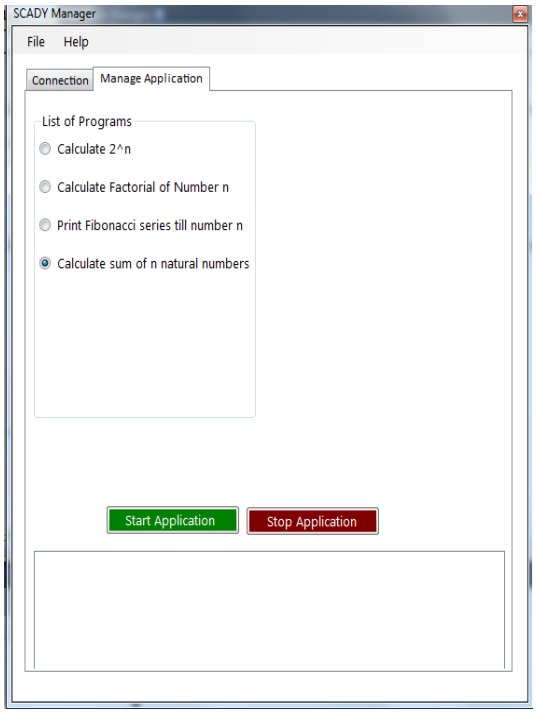

FIG-IV: SCADY MANAGER (APPLICATION MANAGER)

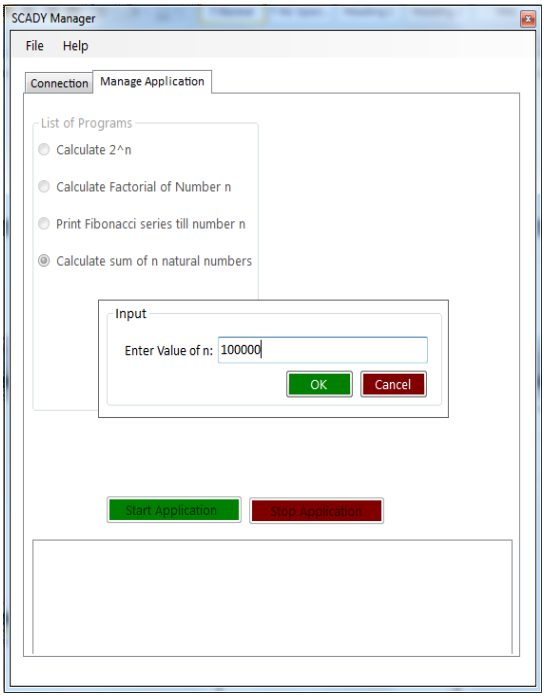

FIG-V: SCADY MANAGER (FOR SUMMATION PROGRAM)

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

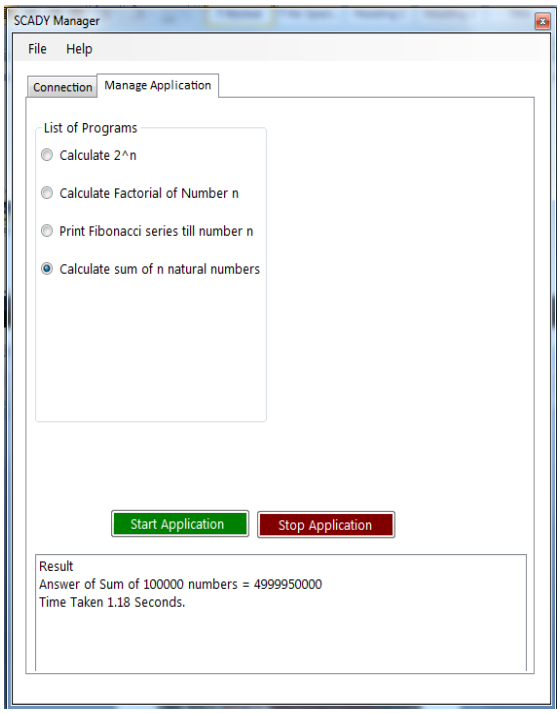

FIG-VI: SCADY MANAGER (FOR SUMMATION PROGRAM)

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

Then deliberately a node that was executing the task was shutdown and the application was tested. The result and time taken for reallocation of task to another node and its execution is noted. The results are given in Table-II.

TABLE-II: COMPARISON OF EXECUTION TIMES WITH & WITHOUT REALLOCATION

Number of Executors	Time (in Secs) - Scady Toolkit	
	Without Reallocation	With Reallocation
3	1.18	3.76
10	1.12	3.34
20	1.09	2.87
50	0.82	2.04

RESULT ANALYSIS

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 and the reallocation of task to another node in case of node failure are shown in Figure-VII.

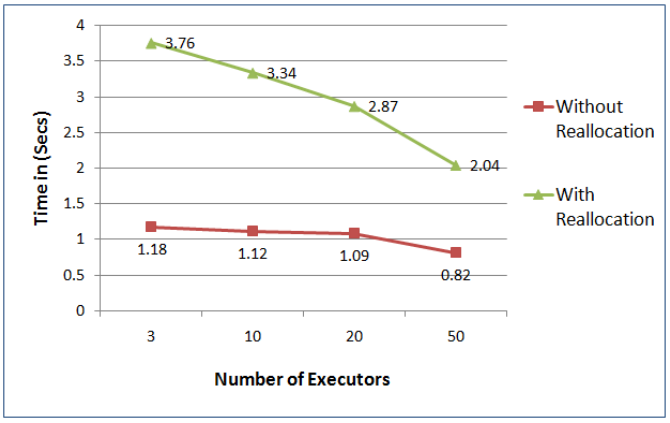

FIG. VII *COMPARISON OF EXECUTION TIMES*

It is noted that 100% efficiency is achieved in reallocation of the task to another node in case of node failure. Only exception is if there are no nodes available in the network for reallocation, the tasks are not reallocated. The time taken for the execution with reallocation is slightly more than the normal execution but can be compromised as the objective is fulfilled.

VI. CONCLUSION

In this paper, we studied Alchemi.Net Grid Framework and Scady Grid Framework for Grid Computing. We found that even though these frameworks provide good performance, there was a need of some mechanism that can provide better fault tolerance. To remove this limitation, we proposed a caching algorithm. At the executor node – intermediate results are written into cache using write-through method. This method stores the result in cache block as well as the main memory. Contents of cache are transferred to manager nodes at specific intervals. In case of node failure, the last transferred value of cache is taken and the thread with same id is allocated to another node of the network.

We implemented the caching algorithm by taking the problem of calculation of sum of n natural numbers. We found that the time taken for the execution with reallocation is slightly more than the normal execution but can be compromised as the task got completed and objective is fulfilled.

In future, some check pointing mechanisms can be used to minimize the time of reallocation of task and its execution.

VII. LIMITATION

There is only one limitation of this approach. Sometimes it may take longer time reading from main memory.

REFERENCES

[1] Rakesh Bhatnagar, Dr. Jayesh Patel, Nirav Vasoya, "Dynamic Resource Allocation in SCADY Grid Toolkit", IEEE International Conference on Computing, Communication and Automation (ICCCA-2015), 15-16 May, 2015.

[2] 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, IEEE Xplore, DOI:10.1109/PERVASIVE.2015.7087085 Publication Year: 2015

[3] 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: 2015

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

[5] 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

[6] 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.

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

[8] 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

[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] Altaf Hussain, Abu Awal Md. Shoeb, Md. Abu Naser Bikas and Mohammad Khalad Hasan, "File Based GRID Thread Implementation in the .NET based Alchemi Framework", IEEE Xplore, Multitopic Conference INMIC, 2008, DOI:10.1109/INMIC.2008.4777784 Publication Year: 2009

[11] Sabir Ismail, Abu Fazal Md Shumon, Md Ruhul Amin, "Distributed Memory Caching for the Fail Safe Computation to Improve the Grid Performance", Proceedings of 13th International Conference on Computer and Information Technology (ICCIT 2010), 23-25 December, 2010, Dhaka, Bangladesh

[12] 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

[13] "Alchemi, .NET Grid Computing Framework", Accessed April 13, 2015 from http://www.cloudbus.org/~alchemi/

[14] Caching level-2, Accessed April 13, 2015 from http://oharanetworks.com/spohara/level-2-cache-controller/

[15] Aditya Patel, Rakesh Bhatnagar, Vipul Prajapati, Dr. J.G. Pandya, "Agent Based Architecture for Scalable Grid Resource Management and Scheduling", International Journal of Computer Applications in Engineering, Technology and Sciences (IJ-CA-ETS), Vol. 2, Issue 1, (ISSN 0974-3596), Oct-Mar, 2009, Paper Number – 72, Page Number: 395 – 401

[16] Rakesh Bhatnagar, Digvijay Rathod, Vipul Prajapati, "Grid Framework – An Optimum Foundation for Service Oriented Architecture", International Journal of Computer Applications in Engineering, Technology and Sciences (IJ-CA-ETS), Vol. 2, Issue 1, (ISSN 0974-3596), Oct-Mar, 2009, Paper Number – 53, Page Number: 299 – 303.