ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 8, August 2013

Analysis Of Fault Tolerance Approach For Data Replication In Data Intensive Scientific Applications

Sinu Nambiar[1], Prof. Rakesh Pandit[2], Prof. Sachin Patel [3]

Student(M.Tech), PCST,Department of Information Technology,Indore,MP,India[1]

Asst. Professor, PCST,Department of Information Technology,Indore,MP,India[2]

Asst. Professor & HOD, PCST,Department of Information Technology,Indore,MP,India[3]

Abstract: Data Intensive scientific computing involves organizing, moving, visualizing, and analyzing massive amounts of data from around the world, as well as employing large-scale computation. There are systems that try to unify job and data management, but these are two different tasks to face in a Grid environment. Existing systems have tried to achieve only either job scheduling or data replication and does not provide fault tolerance. Data replication and job scheduling are two different but complementary functions in Data Grids: one to minimize the total file access cost (thus total job execution time of all sites), and the other to minimize the MakeSpan (the maximum job completion time among all sites). The two main challenges: first, how to formulate a problem that incorporates not only data replication but also job scheduling, and which addresses both total access cost and maximum access cost; and second, how to find an efficient algorithm that, if it cannot find optimal solutions of minimizing total/maximum access cost, gives near-optimal solution for both objectives in the proposed system. A novel method to achieve maximum fault tolerance in the Grid environment system by using a fault recovery pool is proposed.

Keywords: MakeSpan, Optimal MakeSpan, Nominal Distribution, Data Grid, GridSim

I INTRODUCTION

i) Introduction

Grid computing joins together many individual devices, creating a distributed system with massive computational power that far surpasses the power of a handful of supercomputers. Because the work is split into small pieces that can be processed simultaneously, research time is reduced from years to months. The technology is also more cost-effective, enabling better use of critical funds. The grid computing is a special kind of distributed computing. In distributed computing, different computers within the same network share one or more resources. In the ideal grid computing system, every resource is shared, turning a computer network into a powerful supercomputer. With the right user interface, accessing a grid computing system would look no different than accessing a local machine's resources. Every authorized computer would have access to enormous processing power and storage capacity.

Grid computing paradigm unites geographically-distributed and heterogeneous computing, storage, and network resources and provide unified, secure, and pervasive access to their combined capabilities. Therefore, Grid platforms enable sharing, exchange, discovery, selection, and aggregation of distributed heterogeneous resources such as computers, databases, visualization devices, and scientific instruments. Grid Computing can be defined as applying resources from many computers in a network to a single problem, usually one that requires a large number of processing cycles or access to large amounts of data.

Grid computing systems work on the principle of pooled resources. A grid computing system uses that same concept: share the load across multiple computers to complete tasks more efficiently and quickly. Computer's resources are:

- **Central processing unit (CPU)**: A CPU is a microprocessor that performs mathematical operations and directs data to different memory locations. Computers can have more than one CPU.
- **Memory**: In general, a computer's memory is a kind of temporary electronic storage. Memory keeps relevant data close at hand for the microprocessor. Without memory, the microprocessor would have to search and retrieve data from a more permanent storage device such as a hard disk drive.
- **Storage**: In grid computing terms, storage refers to permanent data storage devices like hard disk drives or databases.

Normally, a computer can only operate within the limitations of its own resources. There's an upper limit to

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 8, August 2013

how fast it can complete an operation or how much information it can store. Most computers are upgradeable, which means it's possible to add more power or capacity to a single computer, but that's still just an incremental increase in performance.

Grid computing systems link computer resources together in a way that lets someone use one computer to access and leverage the collected power of all the computers in the system. To the individual user, it's as if the user's computer has transformed into a supercomputer.

ii) Data Grids

A **data grid** is an architecture or set of services that gives individuals or groups of users the ability to access, modify and transfer extremely large amounts of geographically distributed data for research purposes. Data grids make this possible through a host of middleware applications and services that pull together data and resources from multiple administrative domains and then present it to users upon request. The data in a data grid can be located at a single site or multiple sites where each site can be its own administrative domain governed by a set of security restrictions as to who may access the data. Likewise, multiple replicas of the data may be distributed throughout the grid outside their original administrative domain and the security restrictions placed on the original data for who may access it must be equally applied to the replicas. Specifically developed data grid middleware is what handles the integration between users and the data they request by controlling access while making it available as efficiently as possible.

Therefore, Grids are concerned with issues such as: sharing of resources, authentication and authorization of entities, and resource management and scheduling for efficient and effective use of available resources. Naturally, Data Grids share these general concerns, but have their own unique set of characteristics and challenges listed below:

- **Massive Datasets:**

Data-intensive applications are characterized by the presence of large datasets of the size of Gigabytes (GB) and beyond. For example, the CMS experiment at the LHC is expected to produce 1 PB (1015 bytes) of RAW data and 2 PB of Event Summary Data (ESD) annually when it begins production .Resource management within Data Grids therefore extends to minimizing latencies of bulk data transfers, creating replicas through appropriate replication strategies and managing storage resources.

- **Shared Data Collections:**

Resource sharing within Data Grids also includes, among others, sharing distributed data collections. For example, participants within a scientific collaboration would want to use the same repositories as sources for data and for storing the outputs of their analyses.

- **Unified Namespace:**

The data in a Data Grid share the same logical namespace in which every data element has a unique logical filename. The logical filename is mapped to one or more physical filenames on various storage resources across a Data Grid.

- **Access Restrictions:**

Users might wish to ensure confidentiality of their data or restrict distribution to close collaborators. Authentication and authorization in Data Grids involves coarse to fine-grained access controls over shared data collections.

- **Accelerating Large- Scale Data Exploration through Data Diffusion**

An alternative approach data diffusion approach, in which resources required for data analysis are acquired dynamically, in response to demand. Resources may be acquired either “locally” or “remotely”; their location only matters in terms of associated cost tradeoffs. Both data and applications are copied (they “diffuse”) to newly acquired resources for processing. Acquired resources (computers and storage) and the data that they hold can be “cached” for some time, thus allowing more rapid responses to subsequent requests. If demand drops, resources can be released, allowing their use for other purposes. Thus, data diffuses over an increasing number of CPUs as demand increases, and then contracting as load reduces.

Data diffusion thus involves a combination of dynamic resource provisioning, data caching, and data-aware scheduling. The approach is reminiscent of cooperative caching cooperative web-caching and peer-to-peer storage systems (Other data-aware scheduling approaches tend to assume static resources However, in our approach we need to acquire dynamically not only storage resources but also computing resources. In addition, datasets may be terabytes in size and data access is for analysis (not retrieval). Further complicating the situation is our limited knowledge of workloads, which may involve many different applications. The restrictions prevailing in this model is it does not allow for keeping multiple copies of an object simultaneously in different sites.

II EXISTING SYSTEM

Replication has been an active research topic for many years in World Wide Web [33], peer-to-peer networks [3], ad hoc and sensor networking [23], [40], and mesh networks [26]. In Data Grids, enormous scientific data and complex scientific applications call for new replication algorithms, which have attracted much research recently. The most closely related work to ours is by copies of an object simultaneously in different stores. In our model, we assume each object can have multiple copies, each on a different site. Some economical model-based replica schemes are proposed. The authors in [9], [7] use an auction protocol to make the replication decision and to trigger long-term optimization by using file access patterns. You et al. [40] propose utility-based replication strategies. Lei et al. [28] and Schintke and Reinefeld [39] address the data replication for availability in the face of unreliable components, which is different from our work. Jiang and Zhang [25] propose a technique in Data Grids to measure how soon a file is to be reaccessed before being evicted compared with other files. They consider both the consumed disk space and disk cache size. Their approach

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 8, August 2013

can accurately rank the value of each file to be cached. Tang et al. [40] present a dynamic replication algorithm for multitier Data Grids. They propose two dynamic replica algorithms: Single Bottom Up and Aggregate Bottom Up. Performance results show both algorithms reduce the average response time of data access compared to a static replication strategy in a multitier Data Grid. Chang and Chang [11] propose a dynamic data replication mechanism called Latest Access Largest Weight (LALW). LALW associates a different weight to each historical data access record: a more recent data access has a larger weight. LALW gives a more precise metric to determine a popular file for replication. Park et al. [31] propose a dynamic replica replication strategy, called BHR, which benefits from "network- level locality" to reduce data access time by avoiding networking congestion in a Data Grid. Lamehamedi et al. [27] propose a lightweight Data Grid middleware, at the core of which are some dynamic data and replica location and placement techniques. Ranganathan and Foster [37] present six different replication strategies: No Replication or Caching, Best Client, Cascading Replication, Plain caching, Caching plus Cascading Replication, and Fast Spread. All of these strategies are evaluated with three user access patterns (Random Access, Temporal Locality, and Geographical plus Temporal Locality). Via the simulation, the authors find that Fast Spread performs the best under Random Access, and Cascading would work better than others under Geographical and Temporal Locality. Due to its wide popularity in the literature and its simplicity to implement, we compare our distributed replication algorithm to this work. Systemwise, there are several real testbed and system implementations utilizing data replication. One system is the Data Replication Service (DRS) designed by Chervenak et al. [15]. DRS is a higher level data management service for scientific collaborations in Grid environments. It replicates a specified set of files onto a storage system and registers the new files in the replica catalog of the site. Another system is Physics Experimental Data Export (PheDEx) system [19]. PheDEx supports both the hierarchical distribution and subscription-based transfer of data. To execute the submitted jobs, each Grid site either gets the needed input data files to its local computing resource, schedules the job at sites where the needed input data files are stored, or transfers both the data and the job to a third site that performs the computation and returns the result. In this paper, we focus on the first approach. We leave the job scheduling problem [40], which studies how to map jobs into Grid resources for execution, and it's coupling with data replication, as our future research. We are aware of very active research studying the relationship between these two [10], [36], [14], [21], [12], [40], [17], [8]. The focus of our paper, however, is on the data replication strategy with a provable performance guarantee. As demonstrated by the experiments of Chervenak et al. [15], the time to execute a scientific job is mainly the time it takes to transfer the needed input files from server sites to local sites. Similar to other work in replica management for Data Grids [28], [39], [8], we only consider the file transfer time (access time), not the job execution time in the processor or any other internal storage processing or I/O time. Since the data are read only for many Data Grid applications [36], we do not consider consistency maintenance between the master file and the replica files. For readers who are interested in the consistency maintenance in Data Grids, please refer to [18], [40], [32].

III PROPOSED SYSTEM

Data replication and job scheduling are two different but complementary functions in Data Grids: one to minimize the total file access cost (thus total job execution time of all sites), and the other to minimize the Makespan (the maximum job completion time among all sites).There are two main challenges: first, how to formulate a problem that incorporates not only data replication but also job scheduling, and which addresses both total access cost and maximum access cost; and second, how to find an efficient algorithm that, if it cannot find optimal solutions of minimizing total/maximum access cost, gives near-optimal solution for both objectives. The prime motive of data replication in intensive application is that to reduce the data file transfer time and bandwidth consumption. In order to minimize the MakeSpan we use Optimal MakeSpan algorithm and for polynomial time Nominal Distribution strategy has been used and it reduces the total data file access delay by at least half of that reduced is proposed in this paper. Both the algorithms are adaptive to the dynamic change of file access patterns in Data Grids. Since grid applications run in a very heterogeneous computing environment, fault tolerance is important in order to ensure their correct behavior. Using GridSim, a popular distributed Grid simulator, it can be demonstrated that the technique significantly outperforms an existing popular file caching technique in Data Grids.

i) System Architecture

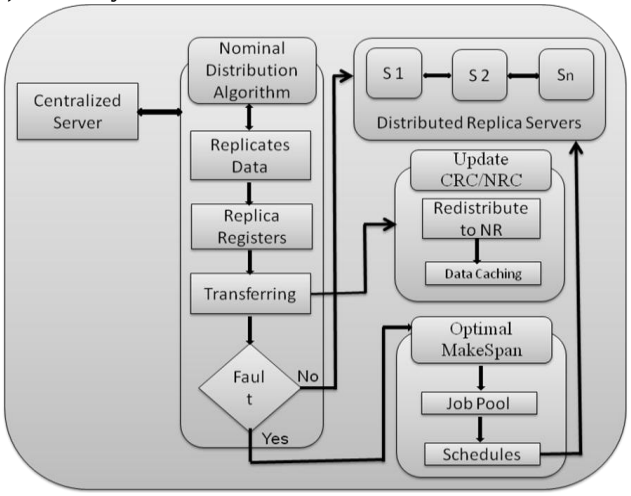

Fig:1 System Architecture

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 8, August 2013

Modules Description

- Building Sites
- Centralized Site
- Replica Sites
- Replica Distribution
- Optimal Cost Constraint
- Data Caching in Replicas
- Fault Tolerant Mechanism
- Makespan Algorithm
- CRC/NRC Updations

- **Building Site**

In this module the centralized site is created. The sensus dataset is collected and organized in the form of .txt Files. As given in our paper the subsites are created according to the files. For instance the number of files is equal to the number of subsites. The top level site created is the centralized management entity in the entire Data Grid environment, and its major role is to mange the Centralized Replica Catalogue (CRC). CRC provides location information about each data file and its replicas, and it is essentially a mapping between each data file and all the institutional sites where the data is replicated. Each site (top level site or institutional site) may contain multiple grid resources. A grid resource could be either a computing resource, which allows users to submit and execute jobs, or a storage resource, which allows users to store data files. We assume that each site has both computing and storage capacities, and that within each site, the bandwidth is high enough that the communication delay inside the site is negligible.

- **Replica Distribution**

In this we predict the Optimizing cost constraint and implement the ***Nominal Distribution Algorithm*** and then transmit as One file per site and terminates replicating files as per cost constraint.

- **Optimal Cost Constraints**

Let D be the centralized site.Assume Grid site G as the centralized Site. Let the Site G has n jobs {j_1,j_2,----------j_n } i.e Input Files M - A_1,A_2 . . ..A_n, where Aj ⊂ V is a set of Grid sites that store a replica copy of Dj,evaluate Total No of jobs submitted, Assume that the Grid site i has n_i submitted jobs. The job is denoted as t_{ik} Which means that needs a subset Fik of D as its input files. Calculate w_{ij} to denote the number of times that site i needs Dj as an input file, wij =$\sum_{k=1}^{n} c_k$ where c_k = 1 if D_j needs file F_{ik} and c_k = 0 otherwise. The transmission time of sending data file Dj along any edge (Network) is s_j/B. We use d_{ij} to denote the number of transmissions to transmit a data file from site i and j (which is equal to the number of edges between these two sites). The total data file access cost in Data Grid before replication is the total transmission time spent to get all needed data files for each site:

$$\sum_{i=1}^{n} \sum_{j=1}^{p} w_{ij} * d_i s_j * s_j / B$$

- **Data Caching in Replicas**

Objective is to minimize the total access cost in the Data Grid:Our Nominal Distribution is a greedy algorithm. First, all Grid sites have all empty storage space (except for sites that originally produce and store some files). Then, at each step, it places one data file into the storage space of one site such that the reduction of total access cost in the Data Grid is maximized at that step.

- **Fault Tolerant Mechanism**

A partial failure may happen when one component in such system fails. This failure may affect the proper operation of other components while at the same time leaving yet other components totally unaffected. An important goal in such systems design is to construct the system in a way that it can automatically recover from partial failures without seriously affecting the overall performance and continue to operate in an acceptable way while repairs are being made when applied to large scale distributed settings (e.g., the Internet). In particular, they fail in providing the desired degree of configurability, scalability, and customizability.

- **Makespan Algorithm**

The job scheduling system is responsible to select best suitable machines in a grid for user jobs. The management and scheduling system generates job schedules for each machine in the grid by taking static restrictions and dynamic parameters of jobs and machines into consideration.

- **CRC/NRC Updations**

The top level site maintains a Centralized Replica Catalogue (CRC), which is essentially a list of replica site list C_j for each data file Dj. The replica site list Cj contains the set of sites (including source site Sj) that has a copy of Dj. Nearest Replica Catalog (NRC). Each site i in the Grid maintains an NRC, and each entry in the NRC is of the form (D_j,N_j).Each site i original graph G has m memory space, the nearest site that has a replica of Dj. When a site executes a job, from its NRC, it determines the nearest replicate site for each of its input data files and goes to it directly to fetch the file (provided the input file is not in its local storage). As the initialization stage, the source sites send messages to the top level site informing it about their original data files. Thus, the centralized replica catalog initially records each data file and its source site. The top level site then broadcasts the replica catalogue to the entire Data Grid. Each Grid site initializes its NRC to the source site of each data file. Note that if i is the source site of Dj or has cached Dj, then Nj is interpreted as the second nearest replica site, i.e., the closest site (other than i itself) that has a copy of Dj. The second nearest replica site information is helpful when site i decides to remove the cached file Dj. If there is a cache miss, the request is redirected to the top level site, which sends the site replica site list for that data file.

For processing this module the data files are collected again. The neighboring sites are analyzed and the files are sent to the replicas.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 8, August 2013

On successfully sending those files to replica sites, the Localized data caching algorithm is implemented.. Since each site has limited storage capacity, a good data caching algorithm that runs distributedly on each site is needed. To do this, each site observes the data access traffic locally for a sufficiently long time. The local access traffic observed by site i includes its own local data requests, nonlocal data requests to data files cached at i, and the data request traffic that the site i is forwarding to other sites in the network.

The file request is consequently provided at the traffic less sites.

IV CONCLUSION

In this paper we focus on how to replicate data files in data intensive scientific applications, to reduce the file access time with the consideration of limited storage space of Grid sites along with fault tolerance. GridSim, a popular distributed Grid simulator, is used to demonstrate a technique that significantly outperforms an existing popular file caching technique in Data Grids

References

[1] Dharma Teja Nukarapu, Liqiang Wang and Shiyong Lu "Data Replication in Data Intensive Scientific Applications with Performance Guarantee", August 2011

[2] The Large Hadron Collider, http://public.web.cern.ch/Public/en/LHC/LHC-en.html, 2011.

[3] Worldwide Lhc Computing Grid, http://lcg.web.cern.ch/LCG/, 2011.

[4] A. Aazami, S. Ghandeharizadeh, and T. Helmi, "Near Optimal Number of Replicas for Continuous Media in Ad-Hoc Networks of Wireless Devices," Proc. Int'l Workshop Multimedia Information Systems, 2004.

[5] B. Allcock, J. Bester, J. Bresnahan, A.L. Chervenak, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, S. Tuecke, and I. Foster, "Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing," Proc. IEEE Symp. Mass Storage Systems and Technologies, 2001.

[6] I. Baev and R. Rajaraman, "Approximation Algorithms for Data Placement in Arbitrary Networks," Proc. ACM-SIAM Symp. Discrete Algorithms (SODA), 2001.

[7] I. Baev, R. Rajaraman, and C. Swamy, "Approximation Algorithms for Data Placement Problems," SIAM J. Computing, vol. 38, no. 4, pp. 1411-1429, 2008.

[8] W.H. Bell, D.G. Cameron, R. Cavajal-Schiaffino, A.P. Millar, K. Stockinger, and F. Zini, "Evaluation of an Economy-Based File Replication Strategy for a Data Grid," Proc. Int'l Workshop Agent Based Cluster Computing and Grid (CCGrid), 2003.

[9] D.G. Cameron, A.P. Millar, C. Nicholson, R. Carvajal-Schiaffino, K. Stockinger, and F. Zini, "Analysis of Scheduling and Replica Optimisation Strategies for Data Grids Using Optorsim," J. Grid Computing, vol. 2, no. 1, pp. 57-69, 2004.

[10] M. Carman, F. Zini, L. Serafini, and K. Stockinger, "Towards an Economy-Based Optimization of File Access and Replication on a Data Grid," Proc. Int'l Workshop Agent Based Cluster Computing and Grid (CCGrid), 2002.

[11] A. Chakrabarti and S. Sengupta, "Scalable and Distributed Mechanisms for Integrated Scheduling and Replication in Data Grids," Proc. 10th Int'l Conf. Distributed Computing and Networking (ICDCN), 2008.

[12] R.-S. Chang and H.-P. Chang, "A Dynamic Data Replication Strategy Using Access-Weight in Data Grids," J. Supercomputing, vol. 45, pp. 277-295, 2008.

[13] R.-S. Chang, J.-S. Chang, and S.-Y. Lin, "Job Scheduling and Data Replication on Data Grids," Future Generation Computer Systems, vol. 23, no. 7, pp. 846-860, Aug. 2007.

[14] A. Chebotko, X. Fei, C. Lin, S. Lu, and F. Fotouhi, "Storing and Querying Scientific Workflow Provenance Metadata Using an Rdbms," Proc. IEEE Int'l Conf. e-Science and Grid Computing, 2007.

[15] A. Chervenak, E. Deelman, M. Livny, M.-H. Su, R. Schuler, S. Bharathi, G. Mehta, and K. Vahi, "Data Placement for Scientific Applications in Distributed Environments," Proc. IEEE/ACM Int'l Conf. Grid Computing, 2007.

[16] A. Chervenak, R. Schuler, C. Kesselman, S. Koranda, and B. Moe, "Wide Area Data Replication for Scientific Collaboration," Proc. IEEE/ACM Int'l Workshop Grid Computing, 2005.

[17] A. Chervenak, R. Schuler, M. Ripeanu, M.A. Amer, S. Bharathi, I. Foster, and C. Kesselman, "The Globus Replica Location Service: Design and Experience," IEEE Trans. Parallel and Distributed Systems, vol. 20, no. 9, pp. 1260-1272, Sept. 2009.

[18] N.N. Dang and S.B. Lim, "Combination of Replication and Scheduling in Data Grids," Int'l J. Computer Science and Network Security, vol. 7, no. 3, pp. 304-308, Mar. 2007.

[19] D. Du¨ llmann and B. Segal, "Models for Replica Synchronisation and Consistency in a Data Grid," Proc. 10th IEEE Int'l Symp. High Performance Distributed Computing (HPDC), 2001.

[20] J. Rehn et al., "Phedex: High-Throughput Data Transfer Management System," Proc. Computing in High Energy and Nuclear Physics (CHEP), 2006.

[21] I. Foster, "The Grid: A New Infrastructure for 21st Century Science," Physics Today, vol. 55, pp. 42-47, 2002.

[22] I. Foster and K. Ranganathan, "Decoupling Computation and Data Scheduling in Distributed Data-Intensive Applications," Proc. 11th IEEE Int'l Symp. High Performance Distributed Computing (HPDC), 2002.

[23] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-Degrees Compared," Proc. Grid Computing Environments Workshop, pp. 1-10, 2008.

[24] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," Proc. ACM MobiCom, 2000.

[25] J.C. Jacob, D.S. Katz, T. Prince, G.B. Berriman, J.C. Good, A.C. Laity, E. Deelman, G. Singh, and M.-H Su, "The Montage Architecture for Grid-Enabled Science Processing of Large, Distributed Datasets," Proc. Earth Science technology Conf., 2004.

[26] S. Jiang and X. Zhang, "Efficient Distributed Disk Caching in Data Grid Management," Proc. IEEE Int'l Conf. Cluster Computing, 2003.

[27] S. Jin and L. Wang, "Content and Service Replication Strategies in Multi-Hop Wireless Mesh Networks," Proc. ACM Int'l Conf. Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM), 2005.

[28] H. Lamehamedi, B.K. Szymanski, and B. Conte, "Distributed Data Management Services for Dynamic Data Grids," unpublished.

[29] M. Lei, S.V. Vrbsky, and X. Hong, "An Online Replication Strategyto Increase Availability in Data Grids," Future Generation Computer Systems, vol. 24, pp. 85-98, 2008.

[30] C. Lin, S. Lu, X. Fei, A. Chebotko, D. Pai, Z. Lai, F. Fotouhi, and J. Hua, "A Reference Architecture for Scientific Workflow Management Systems and the View Soa Solution," IEEE Trans. Services Computing, vol. 2, no. 1, pp. 79-92, Jan.-Mar. 2009.

[31] M. Mineter, C. Jarvis, and S. Dowers, "From Stand-Alone Programs towards Grid-Aware Services and Components: A Case Study in Agricultural Modelling with Interpolated Climate Data," Environmental Modelling and Software, vol. 18, no. 4, pp. 379-391, 2003.

[32] S.M. Park, J.H. Kim, Y.B. Lo, and W.S. Yoon, "Dynamic Data Grid Replication Strategy Based on Internet Hierarchy," Proc. Second Int'l Workshop Grid and Cooperative Computing (GCC), 2003.

[33] J. Pe´rez, F. Garcı´a-Carballeira, J. Carretero, A. Caldero´ n, and J. erna´ndez, "Branch Replication Scheme: A New Model for Data Replication in Large Scale Data Grids," Future Generation Computer Systems, vol. 26, no. 1, pp. 12-20, 2010.

[34] L. Qiu, V.N. Padmanabhan, and G.M. Voelker, "On the Placement of Web Server Replicas," Proc. IEEE INFOCOM, 2001.

[35] I. Raicu, I. Foster, Y. Zhao, P. Little, C. Moretti, A. Chaudhary, and D. Thain, "The Quest for Scalable Support of Data Intensive Workloads in Distributed systems," Proc. ACM Int'l Symp. High Performance Distributed Computing (HPDC), 2009.

[36] I. Raicu, Y. Zhao, I. Foster, and A. Szalay, "Accelerating Large-Scale Data Exploration through Data Diffusion," Proc. Int'l Workshop Data-Aware Distributed Computing (DADC), 2008.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 8, August 2013

[37] A. Ramakrishnan, G. Singh, H. Zhao, E. Deelman, R. Sakellariou, K. Vahi, K. Blackburn, D. Meyers, and M. Samidi, "Scheduling Data-Intensive Workflows onto Storage-Constrained Distributed Resources," Proc. Seventh IEEE Int'l Symp. Cluster Computing and the Grid (CCGRID), 2007.
[38] K. Ranganathan and I.T. Foster, "Identifying Dynamic Replication Strategies for a High-Performance Data Grid," Proc. Second Int'l Workshop Grid Computing (GRID), 2001.
[39] A. Rodriguez, D. Sulakhe, E. Marland, N. Nefedova, M. Wilde, and N. Maltsev, "Grid Enabled Server for High-Throughput Analysis of Genomes," Proc. Workshop Case Studies on Grid Applications, 2004.
[40] F. Schintke and A. Reinefeld, "Modeling Replica Availability in Large Data Grids," J. Grid Computing, vol. 2, no. 1, pp. 219-227, 2003.
[41] H. Stockinger, A. Samar, K. Holtman, B. Allcock, I. Foster, and B. Tierney, "File and Object Replication in Data Grids," Proc. 10th IEEE Int'l Symp. High Performance Distributed Computing (HPDC)