An Intelligent Dynamic Replica Selection Model within Grid Systems

Nour Mostafa, Ismaeel Al Ridhawi, Ahmed Hamza
College of Engineering and Technology
American University of the Middle East (AUM)
Egaila, Kuwait
{Nour.Moustafa; Ismaeel.Al-Ridhawi; Ahmed.Hamza}@aum.edu.kw

Abstract— **Grid systems have emerged as a means of sharing computational resources and information. Providing services for accessing, sharing and modifying large databases is a crucial task for grid management systems. This paper proposes an artificial neural network (ANN) prediction mechanism that provides an enhancement to data replication solutions within grid systems. Current replication services often exhibit an increase in response time, reflecting the problems associated with the ever increasing size of databases. The proposed replica selection prediction model will locate files for incoming jobs using users' historical executions. Experimental results demonstrate the significant gains achieved by the proposed solution in terms of high accuracy and low overheads.**

Keywords— artificial neural networks; data-intensive grids; distributed databases; replication strategies; cloud computing.

I. INTRODUCTION

Data Grid systems are growing in complexity. They are handling ever-greater numbers of users, resources and job requests. The data files produced by and used in Data Grid systems are also increasing in size and number. Delays in job execution and longer job execution times are a consequence of systems having to search for, transfer and process large volumes of data in response to individual user requests.

While the volume of data that needs to be accessed and intelligently managed on a Data Grid today is of the order of terabytes, it is expected to reach petabytes in the near future. Ensuring efficient access to such huge and widely distributed data is a serious challenge to network and Grid designers.

Replication is one widely accepted solution in distributed environments. By storing data at more than one site, if a data site fails, a system can operate using replicated data, thus increasing availability and fault tolerance. At the same time, as the data are stored at multiple sites, it is often possible to find the required data close to the site where the request originated, thus increasing the performance of the system, reducing bandwidth consumption, and improving scalability of the overall system [1].

In order to tackle the problem of scalability, we propose a neural network based intelligent replication service for grid systems that will optimize the response time in meeting the user's data requirements. The proposed model presents a predictive component to predict the location of data required by users. This predictive element will determine a file's location either in cache, local or remote resources by using a neural network to analyze user's past history.

II. BACKGROUND & RELATED WORK

Replication is regarded as one of the major optimization techniques for providing fast data access [2]. Replication services are designed to facilitate and support data grid applications. A replication service aims to select the most appropriate data replica from those available in order to minimize application access time. Replica catalogues (RCs) maintain indexes that represent the mapping between logical file names (LFNs) and physical file names (PFNs). When a system requests an LFN that does not exist in the local regional replica catalogue (regionalRC1 in Figure 1), the local regional RC forwards the request to the root replica catalogue (TopregionalRC in Figure 1). If the file is registered, the root replica catalogue gives the local regional RC the address of the remote regional RC (regionalRC2 in Figure 1) at which the resource is held. The local regional RC then requests the PFN from the remote regional RC and the PFN is forwarded to the system that originated the request.

The following example sets the process in context. A scientific experiment site generates a large volume of data which are stored in a data centre. The data centre notifies the local regional RC of a list of available data sets in the centre. The local regional RC in turn notifies the root RC of the new data sets that have been generated and stored in the data centre. A user submits a job that causes his/her system to request a copy of a data set. The local regional RC is checked first and if the data set is stored at the local centre, the PFN will be sent to the requesting system. Otherwise the local regional RC will send a request to the root RC and the process will proceed as described in the preceding paragraph.

Locating files for job requests is one of the most important issues to be considered by Grid infrastructure developers. The majority of Data Grid systems locate files by using a replica catalogue, either hierarchical (as just described) or centralized [3]. The work of Sayal et al. [4] was one of the first replica approaches. Sayal's aim was to select the closest replica to the user according to selected metrics – e.g. the distance between the user and the replica, and the latency associated with the

978-4799-8422-0/15/$31.00

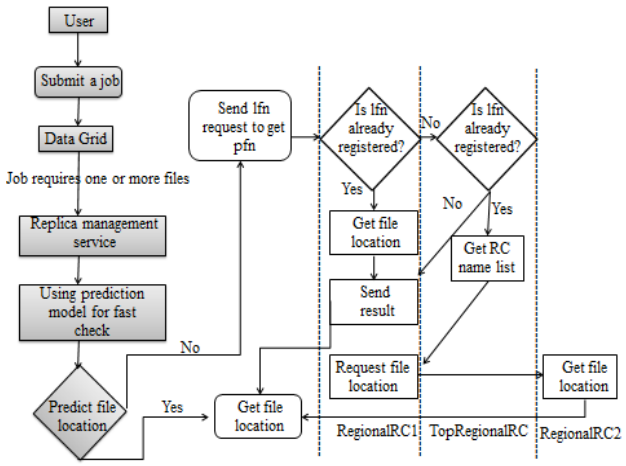

Fig. 1. Interaction between the prediction model and GridSim.

World Wide Web and the HyperText Transfer Protocol (HTTP). The work provided a distributed algorithm for a replication-based approach to web services: the choice of location for storing a replica is based on statistics from previous executions; a server accepts a resource only if the expected waiting time for its clients is reduced. Sayal's work was aimed at web applications and not grid-intensive applications.

A hierarchical model was introduced by H. Stockinger [5] and the EU Data Grid project [6]. In this model the elements of the RC are connected as a web tree. The root of the tree stores some information about files (e.g. filenames) and a list of leaf RCs, while the leaf RCs store mappings between filenames and the resources on which the corresponding physical files reside. Information is switched between the root and the leaves if one RC fails to process the query. On the other hand, a centralized model, of which Napster [7] is an example, uses only one RC that handles all queries sent by resources and users. However, as users, resources and queries grow in number, the RC is likely to prove a major bottleneck in the system [8].

These approaches use a simple matchmaking approach based on filename. Replica catalogues work tolerably well for small systems. However, in more complex configurations, job turnaround time increases each time a requested file proves not to be registered in a particular RC and an alternative RC has to be sought. Current databases already store petabytes of information and there is a continuing trend towards still larger databases and increasingly complex retrieval paths. Intelligent management of very large scale data transfer over wide area networks is therefore a particularly pressing problem. Retrieval strategies will increasingly have to cope with requests for multiple large files stored in dispersed locations. Centralized algorithms are likely to be ineffective: a scheduling algorithm that focuses only on maximizing job utilization, and disregards costs associated with fetching remote data, is certain to be inefficient [9].

Different distributed replica management strategies have also been proposed in the literature [10], [11], [12], [13], and [14]. Unfortunately, research to date has described the general idea of replication and the implementation of replication services within small systems such as mobile and disconnected computers or file systems and Internet services. Now researchers are turning to the problem of scalability in the face of increasing database size and larger numbers of data-intensive applications.

In this paper, the proposed solution for locating files uses a traditional approach for searching, i.e. searching local, top, or remote replica catalogues. The proposed approach uses an artificial neural network (ANN) that predicts file locations after training on characteristics taken from Gridsim [15]. Artificial neural networks are computer programs that are trained to recognize input patterns and associate them with particular outputs – in this case the task is to associate different combinations of job attributes with file locations [16]. ANNs are suitable for training over hundreds or even thousands of passes through large data sets [17], [18]. Data Grid deals with such very large data sets. ANNs have an advantage over a potential rival technology, expert systems (ES), in that ANNs are domain free, which means chemistry, physics, computer and genetic domains can work in the same Grid. ESs, by contrast, are domain-specific. Beltran [19] used an ES approach in P2P and overlay networks systems. Finally ANNs are adaptive and need little domain-specific programming – in marked contrast to ESs [17], [18].

III. A Predictive ANN Mechanism

The proposed model introduced in this paper provides an efficient solution to access local and remote files. The model assumes the following:

- There will be multiple users, spread over different remote sites.
- Users will submit a number of tasks (jobs).
- Tasks (jobs) can require one or more files.
- Files can be located in local or remote resources.

Where appropriate, existing techniques for improved file access – including file replication – are incorporated into the proposed system. Crucially the proposed system uses the new ANN prediction tool, reducing the overhead of job turnaround time by predicting file locations for incoming job queries.

The prediction tool uses a neural network trained on users' past history to anticipate a file's location either in cache, local resources or remote resources. Figure 1 shows the interaction between a typical current replication model and the new predictive model.

Artificial neural networks generally have at least three layers. In such models these layers are referred to as input, middle (or hidden), and output [20]. Data enter the system at the input layer. The input data are passed to intermediate processing units in one or more hidden layers. The hidden layers process the inputs and pass the new signal on to the output layer. The network learns through repeated adjustment of the connection weights between the artificial neurons. Adjustment proceeds according to a back-propagation learning algorithm: if the actual output deviates sufficiently from the desired output, back-propagation causes error information to be passed back through the network; this error information is

used to adjust the connection weights, and the process is repeated over multiple passes through the training data until actual and desired output is sufficiently close for training to stop.

It has been observed that many users of parallel computers do the same work and use the same data repeatedly [21]. The proposed model can therefore exploit habitual job parameters from execution logs (user id, filename, file location, resource id, etc.) to predict the file locations required for new jobs. After a task is completed, the parameter sets which are used to find file locations are stored in an RC database. These parameter sets are used as training vectors for an artificial neural network, which will then be used to predict file locations for future jobs. A large relevant training set is one factor that increases accuracy of prediction [22].

The neural network tool that has been used in this project is justNN [23], a prototyping and development system for the training and testing of multilayer perceptions. A major advantage of using justNN is that it allows input characteristics to be defined, and output produced, in the form of numeric and textual data. This fits with our model structure.

One way of reducing the expected delays in accessing remote files is to predict file (i.e. data) requirements and to pre-load (or pre-access) this data. The proposed prediction model supports this strategy by predicting file location, thereby reducing the overheads associated with potentially complicated searches in local, top and remote replica catalogues. Importantly, the proposed prediction system supports a number of Data Grid environments. Our results (Section IV) show a good performance in small, medium and large systems, promising a flexible, expandable and sustainable solution. In current Grid configurations, especially at times of high system usage, the sequence of accesses within a hierarchy of replica catalogues becomes a very significant obstacle to efficient job processing. Using the proposed prediction model, the replica management service is able to determine the location of a file in one step and inform the requesting job immediately.

In the proposed technique, our ANN uses the 4 key parameters previously identified (User ID, Resource ID, Required File, and File Size) to predict file locations for new jobs. If a job completes successfully on the basis of a file location prediction, the job parameters are stored in a 'history database' separate from the RC. This history database is used to support our prediction model (PM). Each time a job enters the system, the database is searched, and, if it contains the file location for a particular job configuration (some instantiation of the 4 key attributes), the result is sent back. Otherwise the ANN makes a new prediction, which if successful, is also added to the prediction model for future use. If a prediction is incorrect, or if there is no match in the prediction model for an incoming job, a conventional RC search is initiated.

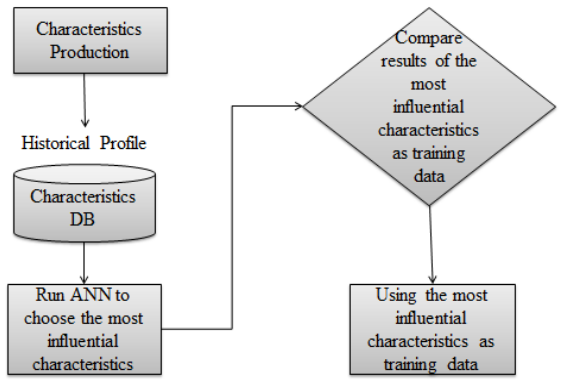

Fig. 2. The acquisition workflow of the required training data.

IV. Experimental Evaluation

Identification of appropriate features for inclusion in a training set or corpus plays a vital part in the development of an artificial neural network. In our case, the candidate features for the training set included details of users, jobs, resources and files. They represented a subset of GridSim characteristics that, to a human observer, are the most likely predictors of file locations. The data were available from GridSim's historical profiler, which stores user, resource, and file characteristics for every job run on the system. Having identified candidate features, we then ran a series of experiments to help us identify a minimum feature set that would allow accurate prediction while reducing the complexity of the prototype ANN's processing task.

Figure 2 illustrates the initial stages of modelling the ANN training data. A set of 500 feature vectors was used for the ANN training process. At the start of the process to identify the minimum feature set, each feature vector comprised a total of 14 elements as shown below, these being divided into 7 job characteristics and 7 file characteristics:

1. Job characteristics
   - 1.1. User ID: the user or owner ID of this job.
   - 1.2. Job ID: the job ID - GridLetID.
   - 1.3. Resource ID: the latest resource ID that processes this Gridlet.
   - 1.4. Actual CPU Time: the total execution time of this job in a given Grid Resource ID.
   - 1.5. ExecStartTime: The latest executions start time.
   - 1.6. Required File: the list files that this job needs for execution.
   - 1.7. ExecFinishTime: the finish time of this job in a Grid Resource.
2. File characteristics:
   - 2.1. Name: the file name.
   - 2.2. Owner Name: the owner name of this file.
   - 2.3. Attribute Size: the size of this object (in byte). This object size is not the actual file size. Moreover, this size is used for transferring this object over a network.
   - 2.4. Size: the file size (in MBytes).
   - 2.5. Resource ID: the resource ID that stores this file.
   - 2.6. Creation Time: the file creation time (in milliseconds).
   - 2.7. Transaction Time: the last transaction time of this file (in seconds).

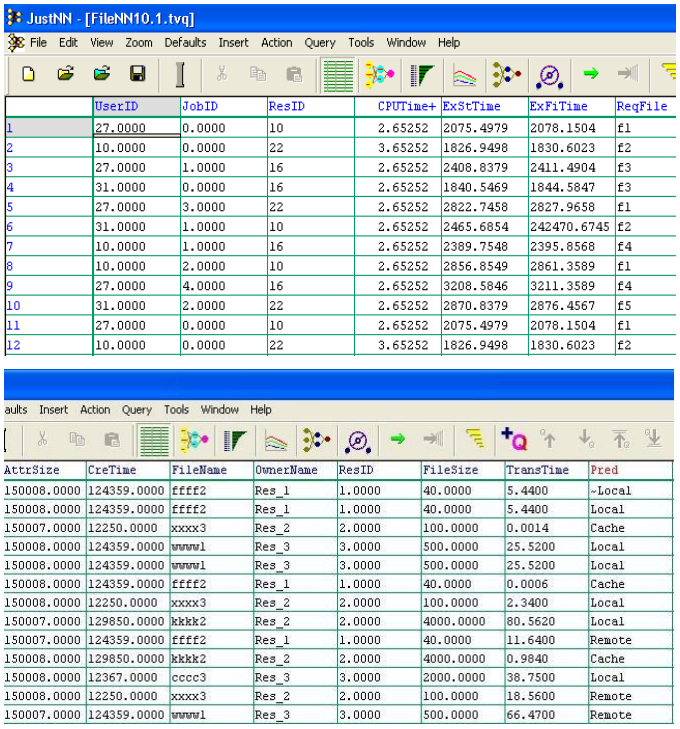

Fig 3. Segment of 1st training run.

To evaluate the prediction model and determine the minimum feature set, the prototype ANN was reconfigured and run with progressively fewer input characteristics. Its output predictions were then compared on each run to the actual resource locations in simulated replica catalogues. It was determined that the input layer can be reduced, without loss of predictive accuracy, to four critical input characteristics: user id, resource id, required file, and file size (Table I).

TABLE I. MAXIMUM AND AVERAGE ERROR RATES IN TRAINING RUNS.

Training Run	Input Features	Training Vectors	Maximum Error Rate
1	14	100	0.056
2	8	100	0.059
3	7	100	0.056
4	6	100	0.053
5	5	100	0.043
6	4	100	0.035

The generation of the training data (inputs and actual outputs – the latter being the file locations, the 'desired output' that the ANN must attempt to predict) was achieved through use of GridSim.

Figure 3, showing output from the ANN development tool during the first training run, has been split vertically to accommodate the 14 columns that represent the 14 input features to the ANN. These columns are followed by a final column that represents the ANN's predicted file location (remote, local or cache) for each input vector. In each case, predicted and actual file location matched.

Figure 4 shows the changing error results in the course of the first training run. In an artificial neural network, 'error', or 'error signal', is the difference between the actual output and the desired output for a particular ANN configuration and a particular set of feature vectors. Figure 4 represents the maximum error rate at a given point in the training process, the minimum error rate and the average error rate. At the end of the first training run, with 14 input features and 5 cycles through a set of 100 training vectors, the ANN's average error rate was already below (and therefore bettered) the target error rate, though the maximum error rate was still a little high. In 5 subsequent training runs we used progressively fewer input features from each of the training vectors but the same number of training cycles. With only 4 input features the lowest maximum error rate was achieved at the end of the training process and the average error rate was still within target. Four input features – user id, resource id, required file, and file size – were therefore used for the predictor model that we tested subsequently. Details of all 6 training runs are shown in Table I.

To test our new predictive approach, we generated a prediction model using actual query data from GridSim and the resultant file location predictions from our ANN. A number of jobs were executed, first using the RC model then the prediction model. The time taken to get file locations in both models was calculated. The following values were used in our calculations. Basic job turnaround time (JTT_S) represents the processing overhead for a single job, excluding times associated with resource search, and is a measure of overall system performance. Replica catalogue time (RCT) and prediction model time (PMT) respectively represent time taken to return a file location by each of the strategies under consideration. TRC in this instance represents time taken to access and exit the top replica catalogue. JTT is the total overhead for the job comprising processing and access times.

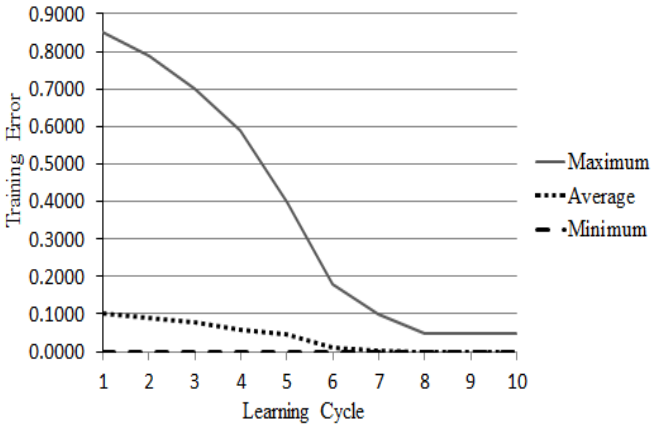

Fig 4. Learning progress in 1st training run.

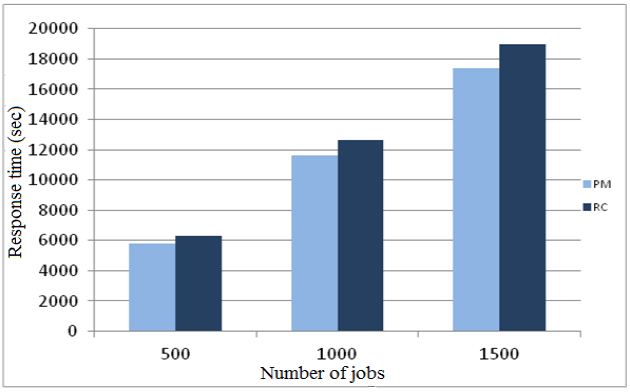

Fig 5. Number of jobs versus response time in RC and PM.

The following equations are used to calculate JTT. Using the replica catalogue model:

$$JTT_{RC} = JTT_S + TRC \quad (1)$$

Using the prediction model:

$$JTT_{PM} = JTT_S + PMT \quad (2)$$

The following is an example of the formulae in action and uses actual experimental data.

JTT_S (basic job turnaround time) = 11.31s

TRC (top replica catalogue time) = 1.34s

PMT (prediction model time) = 0.3s

$$JTT_{RC} = 11.31 + 1.34 = 12.65 \quad (3)$$

$$JTT_{PM} = 11.31 + 0.3 = 11.61 \quad (4)$$

The full simulation results, as shown in Figure 5 and Table II, show that for all scenarios JTT_{PM}, which uses the prediction model, is less than JTT_{RC}, which uses replica catalogue model.

The overall time to execute 500 jobs using the RC model was 6,325 seconds, while the overall time for the same number of jobs using the prediction model was 5,805 seconds.

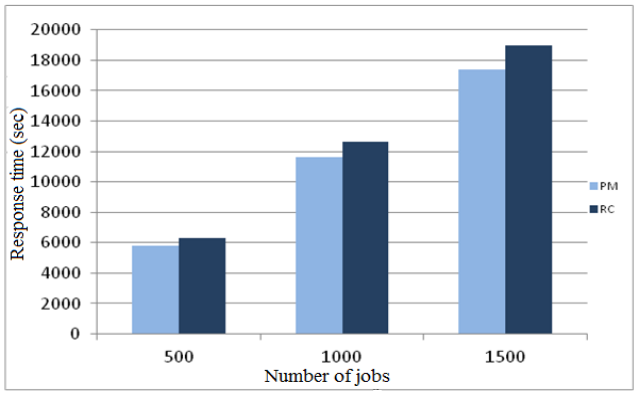

Fig 6. Number of jobs versus response time in RC and PM.

TABLE II. SIMULATION RESULTS: JTT USING RC AND PM.

No. of jobs	JTT using RC	JTT using PM	Difference
500	6325s	5805s	520s
100	12650s	11610s	1040s
1500	18975s	17415s	1560s
Total	37950s	34830s	3120s
Average	12650s	11610s	9%

The difference between the RC model and the proposed model is 520 seconds. The proposed model therefore reduces the overall time taken for job processing by 9%.

It can be seen from Figure 6 that the PM approach outperforms RC, showing significant time savings across the complete load range of 500 to 2000 jobs.

V. CONCLUSION & FUTURE WORK

In summary, this paper describes a successful prediction model for file location. Currently the process of locating files uses a traditional catalogue search strategy, examining local, top, and remote replica catalogues. The proposed approach uses a neural network tool which is responsible for accurately predicting locations of files based on characteristics taken from GridSim. Prediction overhead is always required to be at minimum because it adds to total turnaround time. The magnitude of the prediction overhead depends on the type of application being run. For smaller applications, the effect of prediction delay is more obvious than those affecting applications with larger execution times. In both cases a smaller prediction overhead is always desirable. The results presented in this paper demonstrate that the overhead from our proposed prediction strategy is consistently lower than that associated with the established replica catalogue search strategy. We have demonstrated that that the time taken to get a file location from an RC is higher than the time taken to get the same file location using the new prediction model. The RC model suffers from the problem that it does not store the user's history to inform its decisions on future execution: the RC model treats each job as a new job. The prediction model on the other hand uses past history to inform the system about incoming queries, and the database is updated constantly with new predictions generated by the ANN. The resulting prediction model is simple, has low overhead, and provides very high accuracy.

We plan to extend our model to include other state-of-the-art search and prediction mechanisms [24], [25] used in closely related disciplines to enhance and provide a much higher accuracy and lower overhead.

REFERENCES

[1] G. Sushant and R. Buyya, "Data Replication Strategies in Wide Area Distributed Systems", Enterprise Service Computing: From Concept to Deployment, Robin G. Qiu (ed), pp. 211-241, ISBN 1-599044181-2, Idea Group Inc.,Hershey, PA, USA, 2006.

[2] H. Bell, G. Cameron, L. Capozza, A. Millar, K. Stockinger, and F.Zini, "OptorSim - A Grid Simulator for Studying Dynamic Data

Replication Strategies. International Journal of High Performance Computing Applications, 17(4), 2003.

[3] A. Sulistio, U. Cibej , B. Robic, and R. Buyya, "A Toolkit for Modelling and Simulation of Data Grids with Integration of Data Storage, Replication and Analysis", Elsevier Science, 17 January 2006.

[4] M. Sayal, P. Scheuermann and R. Vingralek, "Content replication in web++". In: Proceedings of the 2nd IEEE International Symposium on Network Computing and Applications (NCA¨03), April 16-18, IEEE Computer Society Washington, DC, USA, Doi: 10.1109/NCA.2003.1201130.

[5] H. Stockinger. Database Replication in World wide Distributed Data Grids. PhD thesis, Fakult¨at f¨urWirtschaftswissenschaften und Informatik, UniversitätWien, 2001.

[6] The European DataGrid project homepage. http://eu-datagrid.web.cern.ch/eu-datagrid, 2005.

[7] "Napster, http://www.napster.com/".

[8] S. Venugopal, R. Buyya, and K. Ramamohanarao, 2006. A taxonomy of Data Grids for distributed data sharing, management, and processing. ACM Comput. Surv. 38, 1 Jun. 2006.

[9] R. Kavitha, and I. Foster, "Decouplin Computation and Data Scheduling in Distributed Data-Intensive Applications". HPDC 2002: 352-358.

[10] A. S. Tanenbaum and M. v. Steen, "Distributed Systems, Principles and Paradigms" first edition, Prentice Hall, 2002.

[11] D.H. Ratner, "Roam: A Scalable Replication System for Mobile and Disconnected Computing" PhD Thesis University of California, Los Angeles, CA January 1998.

[12] R. Guy, P. Reiher, D. Ratner, M. Gunter, W. Mak, and G. Popek"Rumor: Mobile Data Access Through Optimistic Peer-to-Peer Replication," Workshop on Mobile Data Access, November 1998.

[13] R. Guy, J. Heidmenn, W. Mak, T. Page, G. Popek, and D. Rothmeier,"Implementation of the Ficus Replicated File system," Proceedings of the summer Usenix Conference, 1990.

[14] Y. Saito and H. Levy, "Optimistic Replication for Internet Data Services", In Proc. of the 14th Intl Conf. on Distributed Computing, p. 297-314, October 2000.

[15] R. Buyya, and M. Murshed," GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing", John Wiley & Sons Ltd, 2002.

[16] S. Yashpal, S. Alok, "NEURAL NETWORKS IN DATA MINING", Journal of Theoretical and Applied Information Technology, 2005 - 2009 JATIT.

[17] C. Kriegger "Nueral Network in Data Mining", 1996.

[18] S. Duggal, R. Chhabra , "Learning Systems and Their Applications: Future of Strategic Expert System". Issues in Information Systems, Vol. III, 2002.

[19] A. Beltran, "Efficient Access to Distributed Information using Structured Peer-to-Peer Systems," PhD Thesis, Queen's University Belfast, 2008.

[20] C. Krieger, Neural Networks in Data Mining, technician report, 1996.

[21] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, S. Tuecke, "The Data Grid: Towards Architecture for the Distributed Management and Analysis of Large Scientific Datasets". Journal of Network and ComputerApplications, 23: pp. 187–200, 2001.

[22] I. Rao and E.-N. Huh, "A probabilistic and adaptive scheduling algorithm using system-generated predictions for inter-grid resource sharing", Journal of Supercomput. (45), 185–204 (2008).

[23] "JustNN Neural Netowork software, http://www.justnn.com/".

[24] Y. Al Ridhawi, I. Al Ridhawi, A. Karmouch, A. Nayak, "A context-aware and location prediction framework for dynamic environments", in Proc. 7th IEEE International conference on Wireless and Mobile Computing, Networking and Communications (WiMob), October 10-12, 2011.

[25] I. Al Ridhawi, N. Samaan, A. Karmouch, "Simulator-Assisted Joint Service-Level-Agreement and Vertical-Handover Adaptation for Profit Maximization", in Proc. 12th IEEE 12th International Symposium on Applications and the Internet (SAINT), July 16-20, 2012.