PSO-LRU Algorithm for DataGrid Replication Service

Víctor Méndez Muñoz[1] and Felix García Carballeira[2]

[1] Universidad de Zaragoza, CPS, Edificio Ada Byron, María de Luna, 1. 50018 Zaragoza, Spain
`vmendez@unizar.es`, `eureka@nodo50.org`,
[2] Universidad Carlos III de Madrid, EPS, Edificio Sabatini, Av. de la Universidad, 30, 28911 Leganés. Madrid. Spain
`fgcarbal@inf.uc3m.es`

Abstract. Data grid replication is critical for improving the performance of data intensive applications. Most of the used techniques for data replication use Replica Location Services (RLS) to resolve the logical name of files to its physical locations. An example of such service is Giggle, which can be found in the OGSA/Globus architecture. Classical algorithms also need some catalog and optimization services. For example, the EGEE DataGrid project, based in Globus open source components, implements for this purpose the Replica Optimization Service (ROS) and the Replica Metadata Catalog (RMC). In this paper we propose a new approach for improving the performance of Data grid replication. With this aim, we apply Emergent Artificial Intelligence (EAI) techniques to data replication. The paper describes a new algorithm for replica selection in grid environments based on a PSO-LRU (Particle Swarm Optimization) approach. For evaluating this technique we have implemented a grid simulator called SiCoGrid. The simulation results presented in the paper demonstrate that the new technique improve the performance compared with traditional solutions.

1 Introduction

Grid replication of remote data is critical for data intensive enterprise and scientific applications, mostly implemented over Globus middleware[1]. Virtual Organisations are usually geographical and user affinity communities around a big data producer, in the scale of Tera Bytes a day, with the aim of extract information from this read-only remote data, by running jobs on the Grid. On this context replication is used for fault tolerance as well as to provide load balancing by distributed replicas of data.

The OGSA[2] and therefore the Globus Toolkit 4.0 assumes the Giggle[3] as a framework for constructing scalable Replica Location Services(RLS) that allows the registration and discovery of replicas. Given a logical identifier of a file(LFN), the RLS must to provide the physical locations of the replicas for the file(PFN). The RLS consists of two components. Local Replica Catalogs

(LRCs) manages consistent information about logical to physical mappings on each site or node. Replica Location Indices (RLIs) hold the information about the mappings contained in one or more LRC. Strong consistency is not required on the RLIs, a soft protocol send LRC state information to connected RLIs, which then incorporate this information into their indices and delete time outs entries. The basic Giggle architecture on Figure 1 shows two layers, but the architecture is usually configured on N layers of hierarchical RLI.

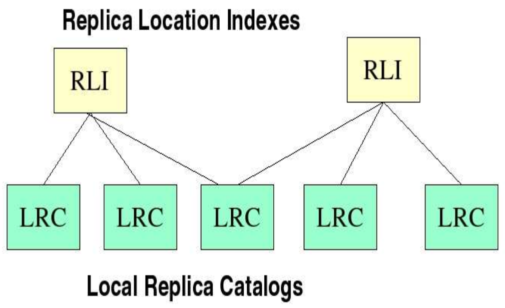

Fig. 1. Basic RLS Architecture

Many research groups have developed algorithms and architectures for replica selection and location. Ann Chervenak et al. propose the Giggle[3] as a framework, and several concrete instantiations based on a hierarchical RLS topology, that are characterized with six parameters shown on the Table 1 of the contribs section, in wich we have added some new values that will be explained.

Other important OGSA/Globus data Grid service components are: GridFTP a not Web Service(WS) component for files transfer, Reliable File Transfer (RFT) for GridFTP monitoring, Data Replication Service (DRS) is the WS component that encapsulate the non-WS RLS and RFT for GT4, OGSA-DAI it is a WS GT4 component for relational data base and XML objects replication. Furthermore, usually it is need some aditional funtionalities, thus the EGEE DataGrid has the ROS and RMC components for the data Grid service framework.

Next section of this paper describes the related work on some aspects of data Grid service:

– Replica state of the art algorithms.
– We analyze the research branches to get some theoretical conclusions.
– We describe the features of the Grid simulators used for experimental test of this algorithms.

After related work section we explain the three main contributions of our approach: a framework review for an enhanced Giggle, a better performance algorithm for replica selection based on PSO and LRU, and we present a complete grid elements shaped on our SiCoGrid (*Simulador Completo Grid*).

On the fourth section we explain the evaluation methodology, and on fifth we present experimental results of our improved approach to the data Grid. Finally we summarise some conclusions.

2 Related Work

Chervenak et al.[3] present some initial performance results for five implementation approaches based on the following Giggle configurations:

– RLS1: Single RLI for all LRCs.
– RLS2: LFN Partitioning, Redundancy, Bloom Filters.
– RLS3: Compression, Partitioning based on Collections.
– RLS4: Replica Site Partitioning, Redundancy, Bloom Filters.
– RLS5: A Hierarchical Index.

They use prototype implementations that show good scalability but does not include network simulation, the prototype is focused on disks throughput, but both disks and network could be system lack depending on study issue class.

There are some approaches that propose an economical algorithm for replica selection where the costs of a file transfers are evaluated as:

$$cost(f,i,j) = f(bandwidht_{i,j}, size_f) \tag{1}$$

Lamehamedi and Deelman approach[4] uses bouth hierarchical and flat propagation graphs spanning the overall set of replicas to overlay replicas on the data grid and minimizing inter-replica communications cost. Beginning on the hierarchical Giggle topology they introduce a flat-tree structure with redundant interconnections for its nodes; closer the node is to the root, more interconnections it has. The flat-tree was originally introduced by Leisersons[5] to improve the performance of interconnection networks in parallel computing systems. Lamehamedi et al. identifies on this approach that flat-tree on a ring topology suits best than hierarchical with multiple servers or peer replica applications. For simulation framework they use a network simulation[6], without considerer the disks throughput, so results are limited by the premise that the system lack is on the network. Anyway they obtain rough network resource consumption evaluation comparing with the pure hierarchical RLS.

Another economic approach[7][8] understand the Grid as a market where data files represent the goods. They are purchased by Computing Elements for jobs and by Storage Element in order to make an investment that will improve their revenues in the future. The files are sold by Storage Elements to Compute Elements and to other Storage Elements. Compute Elements try to minimise the file purchase cost and the Storage Elements have the goal of maximising profits.

When a replication decision is taken, the file transfer cost is the price for the good, like the function 1 show above. The Replica Optimiser may replicate or not based on whether the replication(with associated file transfer and file deletion) will result in to reduce the expected future access cost for the local Computing Elements. Replica Optimiser keeps track of the file requests it receives and uses an evaluation function: $E(f, r, n)$, defined in [9] that returns the predicted number of times a file f, will be request in the next n, based on the past r request history base line. The prediction function E is calculated for a new file request received on Replica Optimiser for file f. E is also calculated for every file in the storage node. If there is no file with less value than the value of new file request f, then no replication occurs. Otherwise least value file is selected for deletion an new replica is created for f.

The research group that propose this approach also present OptorSim [10] [11], the first Grid simulator that holds network and in some way disk costs. The first version was time driven but second version is event driven and it also has others scheduling improvements[12]. Results [7] present some specific realistic cases where the economic model shows marked performance improvements over traditional methods.

A Peer-to-Peer replica location service based on a distributed hash table[13] is fill on Giggle with Peer-to-Peer-RLI(P-RLI). P-RLI uses the Chord algorithm to self-organise P-RLI and it exploits the Chord overlay network to replicate P-RLI mappings. The Chord algorithm also route adaptively the P-RLI logical names with LRC sites. The replication of mappings provides a high level of reliability in the P-RLI, the consistency is stronger than in simple RLI nodes. The P-RLS performance is tested on a 16-node cluster scale with the network size. It is also tested with a simulation for larger network of P-RLI nodes, evaluating the distribution of mappings in the P-RLS network. The simulation for this test section is not a complete simulation of the P-RLS system, but rather, it focuses on how keys are mapped to the P-RLI nodes and how queries for mappings are resolved in the network.

Nowadays there are many approaches with similar methods and similar performances as state of the art above. Other descentralized adaptive replication mechanism[14] organise nodes into overlay network and distribute location information, but do not route requests. Each node that participates in the distribution network build, in time, a view of the whole system and can answer queries locally without forwarding request. Unfortunatelly this is not common on large scale scientific datasets, that suppose the most of the operative Grid infrastructures.

3 Contributions

3.1 Proposed Data Grid Service Framework

The enhanced Giggle shown on Table 1 avoid the restrictions for the flat approaches. Now it is not necessary to store the LFN mapping out of the local node. It is not necessary to implement any RLI layer on the architecture. Therefore the RLS is completely consistent. On the function used to partitioning the

LFN name space, we add a entry for flat architecture with no partitioning by LFN. Every LRC manage the name space locally independent. First introduced value is for G = 0 pointing out a flat RLS composed only by LRCs and no RLI layer. There are a no partitioning actions for LFN names space ($P_L = flat$), and no partitioning the RLI name space ($P_R = flat$). For the degree of redundancy in the index space we add a new case R = 0 for the LFN mapping only on the LRC. We also have include economic and flat heuristic for possible S values (the function used to determine what LRC information to send to other catalog entities and when).

Table 1. The six parameters enhanced Giggle RLS structures and values.

G		The number of RLIs
	G = 0	A flat partitioned index, only LRC on a flat layer
	G = 1	A centralised, non-redundant or partitioned index
	$G > 1$	An index that includes partitioning and/or redundancy
	$G \geq N$	A highly descentralized index
P_L		The function used to partitioning the LFN name space
	$P_L = O$	No partitioning by LFN. The RLIs must have storage to record information about all LFNs, a large number
	$P_L = hash$	Random partitioning. +load balance, -locality
	$P_L = coll$	Partitioning on collection name. -load balance, +locality
	$P_L = flat$	No partitioning by LFN. Every LRC is locally manage
P_R		Function used to partition replica site name space
	$P_R = 0$	No partitioning by site name. Indices have entries for every replica of every LFN they are responsible for.
	$P_R = IP$	Partitioning by domain name or similar.
	$P_R = flat$	There are no index for partitioning site name space.
R		The degree of redundancy in the index space
	R = 0	The LFN mapping is only on the LRC
	R = 1	No redundancy: each replica is indexed by only one RLI
	$R = G > 1$	Full index of all replicas at each RLI. Implies no partitioning, much redundancy/space overhead.
	$1 < R < G$	A highly descentralized index.
C		The function used to compress LRC information
	C = O	No compression: RLIs receives full LFN/site information
	C = bloom	RLIs receive bloom filters summaries
	C = coll	RLIs receive summaries based on collection distribution
S		Function to set what LRC information to send where
	S = full	Periodically send entire state to relevant RLIs
	S = partial	In addition, send periodic summaries of updates
	S = economic	Every economic decision send entire state to RLIs
	S = flat	Only statistical information is send for flat heuristic

At the end of the day we will have a stand alone LRC for each node, with a local location service and need an implicit global location interface, but with a distributed RLS service.

The data Grid service framework proposed, will complementary need a modified ROS, with flat heuristic features on two maners:

– Those algorithms like PSO that need to seend some statistical information, bind pear to pear conexion between ROS servers in each node.
– Other Emergent Artificial Intelligence (EAI) algorithms are stand alone ROS servers, and does not need any control information transfer.

There also is a distributed ROS service.

The typical RMC service si not necesary for our goal. We do not use a GUID, because each local catalog make mapping betwing LFNs and PFN in a oneness way for the local node. But the LRC will need two aditional entries for the metadata information and the original producer node of the file. So we use an enhaced RLS with some soft catalog funtionalities.

This new theoretical approach requires an heuristic that realizes enough performances with only statistical information about LRC, and a request routing scheme self described. This is our goal on the next subsection proposing PSO file location and selection scheme and LRU deletion mechanism as an alternative to traditional approaches. Our data Grid service framework is also valid for any new approach that may walk on flat heuristic way.

3.2 The algorithm: PSO-LRU

PSO is an Emergent Artificial Intelligence technique. EI is an Artificial Intelligence branch that uses the natural social behaviour as ant colons or PSO[15] inspired on bees swarm or birds flocks searching food. PSO has been proved as a valid approach for many different real solutions[16][17].

On Grid environments we introduce some tactic modifications, based on the strategy "follow the closer bird from the food chunk" as social PSO flavour.

– A bird flock is in a random search for food in an area.
– For each bird there is only one valid kind of food.
– The bird does not known where is the food chunk, but its known how long is from the different areas and it know how many birds are finding they food chunk on this areas, this is called food chirp. This is the social component of our approach, thus the distance to the food chunk is calculated for each bird flock, not for individual birds.
– The strategy is to follow the closer bird flock with best success food search.

Translating this analogy to the Grid, we suppose that a file location request is a bird searching food. When the bird stand on an area it is on a Grid node, when the bird fly looking for food to another node is moving through the Remote Network. The bird takes the decision from where to search based on the flock food chirp, that is the best performance external hit ratio of different nodes.

On the other hand, the food chirp will decreased across distance. If the bird is over-flying a node and find food then it will change direction to get it, if the bird arrive to destination and is no food then start again from this point. Thus the performance function for file f to node j from node i looks as following. The PSO-Grid uses a performance metric for a file replication between two nodes i, j , defined as follow in equation 2. We use b as the identifier of the node with the best performance metric asociated to i, from the evaluated j nodes. Initially b is the producer node of the replica, that will be return by LRC soft catalog metadata information described above, and in the pseudo-code below is the get producer function. We use e for the external hit ratio and c for the network cost.

$$p_{i,j} = (e_j * c_{i,j}) + ((1 - e_b) * c_{i,b}) \tag{2}$$

The external hit ration is calculated based on N lasts external success request ratio on node j. The external ration events are the information that is sent from one ROS in each node to another. Considering network access cost we propose the following:

$$cost(f, i, j) = f(latency_{i,j}, bandwidht_{i,j}, size_f) \tag{3}$$

Latency is a constant but do not mean neutral on transfers[18], the latencies are growing from one network to an other, the bandwidth on a network connection is the minimal bandwidth assigned from one network to another.

The performance function is balancing the probability of find a replica in a node j with the probability of not finding on j, where we have to reply from the node with best metric$b$, initially the producer.

The core pseudo-code is for the function getPSOBest that return the best the best performance node from node-Id to LFN referenced on file catalogs index. The get PSO metric function calculate the performance PSO metric described in the equation above 2. The 3 equation is implemented on the get network cost function.

```
NodeIdType getPSOBest(NodeIdType i, FileIdType f)
{
   bestIdNode = get_producer(f)
   bestPSOmetric = get_network_cost(i,bestIdNode)

   For each j from the Grid node set repeat
   {
      if ( i != j )
      {
         if ( get_PSO_metric(i,j} < bestPSOmetric})
         {
            bestPSOmetric =  obtenerMetricaPSO(i,j)
            bestIdNode = j
         }
      }
```

```
    }
    return(bestIdNode)
}
```

The deletion decision is taken in each node only to serve local request, using the LRU or LFU algorithm for selectint the file target. When a file deleted is on process to remote node reply, the node trigger a new PSO reply in the name of the in-reply remote node for the rest of the file transfer.

4 Evaluation Methodology

We have developed a tool that creates log files for the given input arguments: access pattern, random seed, number of Grid clients by node, number of jobs by Grid client.

The access pattern are full file, sequential block access, random, unitary random walk, gaussian random walk, same as OptorSim simulator[10][7]. The random seed is for statistical experiment repetitions. Number of Grid Clients in a node is a component of the simultaneous request on a node. The number of jobs by grid client is a temporal component of the simulation.

Each job will request many file blocks. The create logs application return for each file request, the block requested, an Active Time and a Passive Time. Those times are empirical model of Web document arrivals at access link[19]. After a job get a file block response it spend an Active Time for process the block part of the job, this time is calculated based on Computer Elements featured specifications on network configuration file. Passive Time is the time that the user hold between one job and another. For this parameter we use a Pareto distribution with k=1 and alfa = 0.9 with infinite mean and variance, that is a characteristic Web Service users distribution[20].

We have implemented SiCoGrid, developed in Parsec[21] that is a combination of C and a simulator parser for creating event driven simulators, and also use DiskSim[22] for the storage disks simulation subsystem. SiCoGrid use these log files and some parameters. Possible Grid algorithm values:

– Unconditional replication, lest frequent use(LFU) file delete.
– Unconditional replication, LRU file delete.
– Economic Model.
– PSO + LFU
– PSO + LRU

As we have seen on related work section, the best reliable Grid simulation shape should considerer disks throughput and network traffic. For this purpose we have implemented both of them. Figure 2 shows the SiCoGrid node elements communicated with a local network simulation. This Grid node configuration is based in OptorSim and Globus. Between nodes there is a remote network simulation with an infrastructure described on network configuration file.

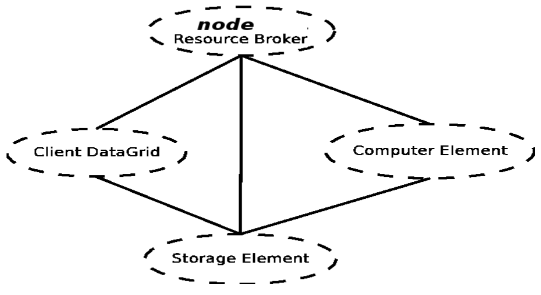

Fig. 2. SiCoGrid node scheme

4.1 Simulation Infrastructure

We have configured our SiCoGrid for a common Grid stage[23] shown on the Table 2. These is the typical CERN datagrid specification for node tier class of a Virtual Organisation. The storage capacity, file size, and network bandwidth is scaled in the magnitude of twenty, for time simulation reasons. Therefore the obtained time results will be on the same magnitude.

Table 2. Scaled Grid Stage

Tier Class	Real MB/s	Scaled Mb/s / 20	Real TB	Scaled TB / 20
1	2048	102.4	220	11
2	320	16	100	5
3	10	0.5	20	1

On the Figure 3 we can see the network infrastructure used in our experiment. The graph disposes a nomenclature where the nodes has a first number that is the tier class, and after the point another identification number. Below there is the storage size of the node in TB. The networks have assigned two numbers, the first one is the latency in ms and the other is the bandwidth in MB/s.

5 Simulation Results

On the following Figure 4 we present Grid simulation results based on the stage described above. We use Gaussian random walk, that is the best performance for the state of the art economic OptorSim approach[10]. The mean and standard deviation job response time is scale in the magnitude of 20 to usual jobs duration from hours to some days. There are shown three different Grid sizes, expressed in the number of Grid clients by node and the number of jobs submitted by a Grid

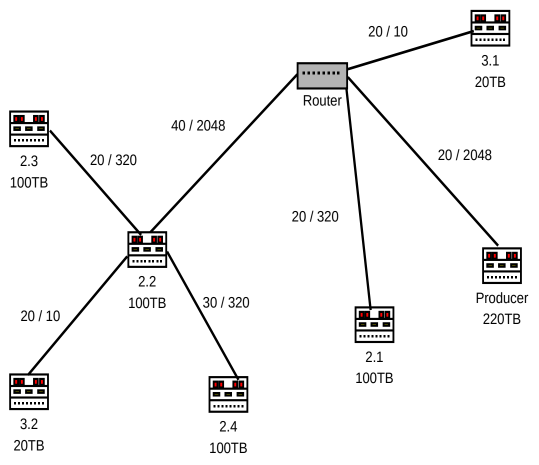

Fig. 3. Simulated Grid Stage

client. The simulations are pure Data Grid, thus all the Active Time for process data are run on the Grid client side. We present results for the best performance LRU over LFU deletion scheme. The OptorSim economic approach also uses LRU for secondary deletion decisions. The Figure shows the unconditional-LRU

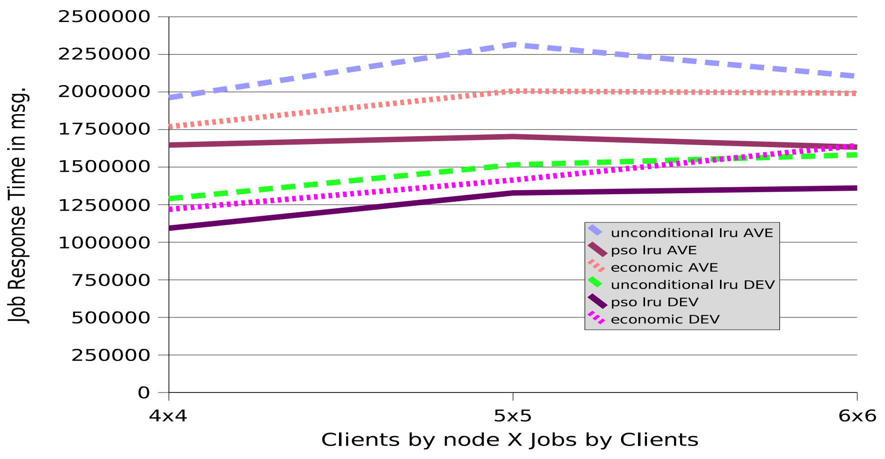

Fig. 4. Results in Simulated Grid Stage

performances with less dashed line style, the economic with a more dashed lines style, and the PSO-LRU with continuous line style. The standard deviations are the tree lines at the bottom, and the average are at the top of the chart. Our PSO-LRU approach has much better performances than the other algorithms for average and standard deviation. As it was expected the unconditional used for base compare, has the worst results. The PSO-LRU approach improve speed over unconditional in percents of 19%, 36% and 29% for simulation size serial of 4X4, 5X5 and 6X6. The PSO-LRU approach improve speed over economic approach in percents of 7%, 18% and 22% for the same simulation serial.

PSO-Grid performance is better due to its features: less control trafic, distributed optimization, localization and selection services, autonomous management of each node will fit best on user and geografical afinities, colaborative strategie against competitive strategie of the economic, that usually performs better on the long term.

6 Conclusions and Future Work

We have described two relevant contributions to the Data Grid corpus. The enhanced Giggle framework that consider flat RLS structures, opening the door to the EI and other EAI approaches for the OGSA data Grid replication architecture. Specific PSO-LRU algorithm has been proved as the better performance job response time and much better scalability features than traditional approaches, using a full network and disk subsystem simulation, SiCoGrid.

We have open research lines for the following targets: Cyclical graph grid infrastructure simulations, other emergent EAI algorithms like Ant Colony Optimization and a depth variable correlations studies.

7 Acknowledgments

This work has been supported by the Spanish Ministry of Education and Science under the TIN2004-02156 contract.

References

1. Foster, I., Kesselman, C.: Globus: A metacomputing infrastructure toolkit. IJSA **11**(2) (1997) 115–128
2. Foster, I., Kesselman, C., M.Nick, J., Tuecke, S.: The physiology of the grid an open grid services architecture for distributed system integration. Technical report, Globus Proyect Draft Overwiev Paper (2002)
3. Chervenak, A.L., Deelman, E., Foster, I., Iamnitchi, A., Kesselman, C., Hoschek, W., Kunszt, P., Ripeanu, M., Schwartzkopf, B., Stockinger, H., Stockinger, K., Tierney, B.: Giggle: A framework for constructing scalable replica location services. In: Proc. of the IEEE Supercomputing Conference (SC 2002), IEEE Computer Society Press (November 2002)
4. Lamehamedi, H., Szymanski, B., shentu, Z., Deelman, E.: Data replication strategies in grid environments. In: Proceedings Of the Fifth International Conference on Algorithms and Architectures for Parallel Processing, ICA3PP02) (2002)
5. Leiserson, C.H.: Flat-trees: Universal network for hardware-efficient supercomputing. IEEE Transactions on Computers **C-34**(10) (1985) 892–901

6. http://www mash.cs.berkeley.edu/ns: Ns network simulator (1989)
7. Bell, W.H., Cameron, D.G., Capozza, L., Millar, A.P., Stockinger, K., Zini, F.: Simulation of dynamic grid replication strategies in optorsim. In: Proc. of the ACM/IEEE Workshop on Grid Computing (Grid 2002), Springer-Verlag (November 2002)
8. Cameron, D.G., Carvajal-Schiaffino, R., Millar, A.P., Nicholson, C., Stockinger, K., Zini, F.: Evaluating scheduling and replica optimisation strategies in optorsim. In: International Workshop on Grid Computing (Grid 2003), IEEE Computer Societe Press (November 2003)
9. Capozza, L., Stockinger, K., , Zini., F.: Preliminary evaluation of revenue prediction functions for economically-effective file replication. Technical report, DataGrid-02-TED-020724, Geneva, Switzerland, July 2002 (July 2002)
10. Bell, W.H., Cameron, D.G., Capozza, L., Millar, A.P., Stockinger, K., Zini, F.: Optorsim - a grid simulator for studying dynamic data replication strategies. International Journal of High Performance Computing Applications **17**(4) (2003)
11. Cameron, D.G., Carvajal-Schiaffino, R., Millar, A.P., Nicholson, C., Stockinger, K., Zini, F.: Analysis of scheduling and replica optimisation strategies for data grids using optorsim. International Journal of Grid Computing **2**(1) (2004) 57–69
12. Cameron, D.G., Carvajal-Schiaffino, R., Millar, A.P., Nicholson, C., Stockinger, K., Zini, F.: Optorsim: A simulation tool for scheduling and replica optimisation in data grids. In: International Conference for Computing in High Energy and Nuclear Physics (CHEP 2004), Interlaken (September 2004)
13. Min Cai, Ann Chervenak, M.F.: A peer-to-peer replica location service based on a distributed hash table. In: Proceedings of the High Performance Computing, Networking and Storage Conference, SCGlobal (2004)
14. Ripeanu, M., Foster, I.: A decentralized, adaptive replica location mechanism. In: 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11). (2002)
15. Shi, Y. ;Eberhart, R.: A modified particle swarm optimizer. In: Proceedings of the IEEE International Conference on Evolutionary Computation, IEEE Press. Piscataway, NY (1998) 69–73
16. Cockshott, Hartman: Improving the fermentation medium for echinocandin b production. part ii: Particle swarm optimization. Process Biochemistry **36** (2001) 661–669
17. Yoshida, Kawata, Fukuyama: A particle swarm optimization for reactive power and voltage control considering voltage security assessment. IEEE Trans. on Power Systems **15** (2001) 1232–1239
18. Cheshire, S.: It's the latency, stupid. Technical report, Stanford University (1996)
19. Deng, S.: Empirical model of www document arrivals at access link. In: Proceedings of the 1996 IEEE International Conference on Communication, IEEE-P (1996)
20. Barford, P., Crovella, M.: Generating representative web workloads. In: Network and Server Performance Evaluation In Proceedings of the 1998 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, ACM SIGMETRICS (1998) 151–160
21. Leijen, D.: Parsec, a fast combinator parser. Technical report, Computer Science Department, University of Utrecht (2002)
22. R.Granger, G., L.Worthington, B., N.Patt, Y., eds.: The DiskSim Simulation Environment. Version 2.0 Reference Manual. University of Michigan (1999)
23. Ranganathan, K., Foster, I.: Identifying dynamic replication strategies for a high-performance data grid. Technical report, Departament of Computer Science, The University of Chicago (2000)