Scalable Computing: Practice and Experience
Volume 11, Number 2, pp. 121–130. http://www.scpe.org
ISSN 1895-1767

PREDICTION AND LOAD BALANCING SYSTEM FOR DISTRIBUTED STORAGE

RENATA SŁOTA[†], DARIN NIKOLOW[†], STANISŁAW POLAK[†], MARCIN KUTA[†], MARIUSZ KAPANOWSKI[†], KORNEL SKAŁKOWSKI[†], MAREK POGODA[‡], AND JACEK KITOWSKI[†‡]

Abstract. National Data Storage is a distributed data storage system intended to provide high quality backup, archiving and data access services. These services guarantee high level of data protection as well as high performance of data storing and retrieval by using replication techniques. Monitoring and data access prediction are necessary for successful deployment of replication. Common Mass Storage System Model (CMSSM) is used to present a storage performance view of storage nodes in unified way for monitoring and prediction purposes. In this paper some conceptual and implementation details on using CMSSM for creating a Prediction and Load Balancing Subsystem for replica management are presented. Real system test results are also shown.

1. Introduction. National Data Storage (NDS) is a distributed data storage system intended to provide high quality backup, archiving and data access services [1]. These services are capable of providing high level of data protection, data availability and data access performance. In order to guarantee these capabilities replication techniques are used. Two problems arise with using this approach: selecting physical storage locations for newly created replicas and choosing the best replica for a given data transfer. If these problems get solved we can count on faster data access.

The client access to NDS is provided by Access Nodes (ANs). ANs spread over the country are located in national computer centers having direct links to the NDS Pionier backbone network [2]. The general idea is that client requests come via different ANs and the requested data is served by the most appropriate Storage Node (SN), selected separately for each request, being the one which can provide requested data fastest. In this way some natural load balancing is achieved depending on the client access pattern.

In order to provide the required high performance data access functionality a replica management system called Prediction and Load Balancing System (PLBS) was implemented. This paper presents some conceptual and implementation details on creating PLBS. Essential part of this research concerns replication and the development of replication policies, which should help achieving reasonable level of storage load balancing. These policies are based on CMSSM storage model [3] which allows to provide an unified layer for monitoring purposes. The potential scope of application of the proposed approach is wide, for example it could be used for multi-player on-line games [4] as well as for data storing from HEP experiments [5].

The rest of the paper is organized as follows: The next section presents the state of the art in the field of replication strategies and storage system modeling and monitoring. The third section describes the CMSSM storage model. Fourth section shows how CMSSM is adopted in the PLBS subsystem. The fifth section gives some overview of the replication strategies used in the system. The test results are presented in the sixth section and the last section concludes the paper.

The paper is an extended and modified version of the article [6] presented at the PPAM09 conference. The extension concerns: (i) description of CMSSM, (ii) showing how CMSSM is adopted in the PLBS subsystem, (iii) presenting new experimental results.

2. State of the Art. With the steadily growing users demand for data storage space and access quality the data management techniques become an important issue of any distributed computing environment [7]. In [8] we can find survey of data management techniques in various distributed systems. Scalability taxonomy of data management is also proposed. A part of the data management systems concerns data replication. There are many researches focused on replication strategies. In [9] five replication strategies for read only data are presented. The strategies have been tested using three different access patterns. The study assumes tiered network with a central data source. Similar network model having constant storage nodes locations in the network hierarchy is studied in [10]. Park et al. in [11] study replication with another network hierarchy with no central storage node, where the node distance is expressed as link bandwidth. Their technique might be better in the case when the Internet is used for data transfer.

The mentioned studies assume static hierarchy and do not take into account the dynamic changes of bandwidth and latency resulting from the load of distributed system. In [12] an attempt to cope with this problem has been made. For replica selection they propose a neural net based algorithm predicting the network

[†]Institute of Computer Science, AGH-UST, al. Mickiewicza 30, 30-059, Kraków, Poland (rena,darin,kito@agh.edu.pl)
[‡]Academic Computer Center CYFRONET-AGH, ul. Nawojki 11, 30-950 Kraków, Poland

transfer time of a replica. Another example of research on replica access time prediction based on previous data transfers measurements is [13] in which the Markov chains are used for prediction.

Authors of paper [14] propose 3 heuristic algorithms for selecting the location of new replica based on network latency parameters and number of client requests observed in a certain time interval from the past. Fair-Share replication presented in [15] for choosing new replica location takes into account previous access load of server as well as availability of storage device represented by their storage load. In this way better load balancing among the storage servers is achieved.

Tests of the proposed replication strategies in the studies mentioned by now are conducted by using simulations. Results of real implementation of proposed models and strategies using monitoring of existing storage environment are shown in [16] and in their previous studies. The presented in these papers replication algorithms embody, besides the data access cost imposed by the network, also the cost caused by storage devices capabilities. In the case when a distributed storage system uses high bandwidth network it turns out that the system bottleneck are the storage devices, which bandwidth can be additionally limited according to their actual access load.

The majority of replica selection algorithms assumes that many users access the same data sets. In the case of data storage service holding mainly private user data, users will rather access their own particular files (holding backups or archives). That is why, essential in this case is access load balancing increasing the overall system utilization and thus reducing the access cost. In the proposed solution essential part of the process of existing replica selection and the process of new replica location selection is focused on the evaluating of storage system performance and evaluating of server load. The evaluation is based on the adopted Common Mass Storage System Model (CMSSM) proposed in [17].

Using of general monitoring system like Nagios [18], Ganglia [19] or Gemini [20] for monitoring the performance of HSM systems is troublesome since these systems do not provide the necessary utilities and methods of data gathering, for example, interactive queries to the monitored system or monitoring of storage request queue. Taking into account the heterogeneity of storage systems in term of hardware and software it is important to have an unified access to the monitoring data. Some effort in this area is done by Distributed Management Task Force [21] by specifying the Common Information Model (CIM) [22] for managing computer systems. SMI-S [23] is a CIM based standard which defines interface for managing storage environments. Grid Laboratory Uniform Environment (GLUE) [24] is a conceptual, object-oriented, information model of Grid environments, and its main goal is to provide interoperability among elements of the Grid infrastructure. The above models do not accurately present HSM systems.

The review of related works shows that the problem of data management for performance and load balancing purposes in distributed system with underlying HSM systems as storage horses is not fully addressed which motivated us to start this research.

3. Common Mass Storage System Model. Storage systems used in modern grid-like computing environment are often heterogeneous for historic or economics reasons. Within a virtualized environment with dynamic changing parameters a proper performance monitoring is a hard task. The lack of an unified representation of performance related view of storage systems and especially of HSM storage system was the motivation for developing the Common Mass Storage System Model [3].

The model supports various storage systems with special attention taken to the HSM systems. In the model a set of performance related parameters are defined and grouped in appropriated classes according to type of storage subsystems or parts (e.g. tape drive, library, etc) they are related to.

The model can be used for internal object based representation of storage system for simulation purposes and is used by various monitoring services to give a performance point of view at the given storage system.

The class diagram for the model representing HSM systems is shown in Fig.3.1. The model is able to represent any HSM systems with a certain accuracy, which depends on the vendor's availability of methods for obtaining the performance related parameters specified in the model. We can see that the model can be quite accurate since parameters like queue of waiting requests with detailed information about each request are specified. This allows for very accurate performance prediction for a particular piece of data residing anywhere in the storage hierarchy since it is possible to monitor where the data resides - on disk cache or tertiary storage (which tape, block, etc), it is possible to monitor if there are free resources (for example tape drives) to fulfill the request.

Our vision of a general distributed storage system which makes use of CMSSM for data access prediction is presented in Fig.3.2. The system has well defined functional layers. On the bottom there is the storage layer

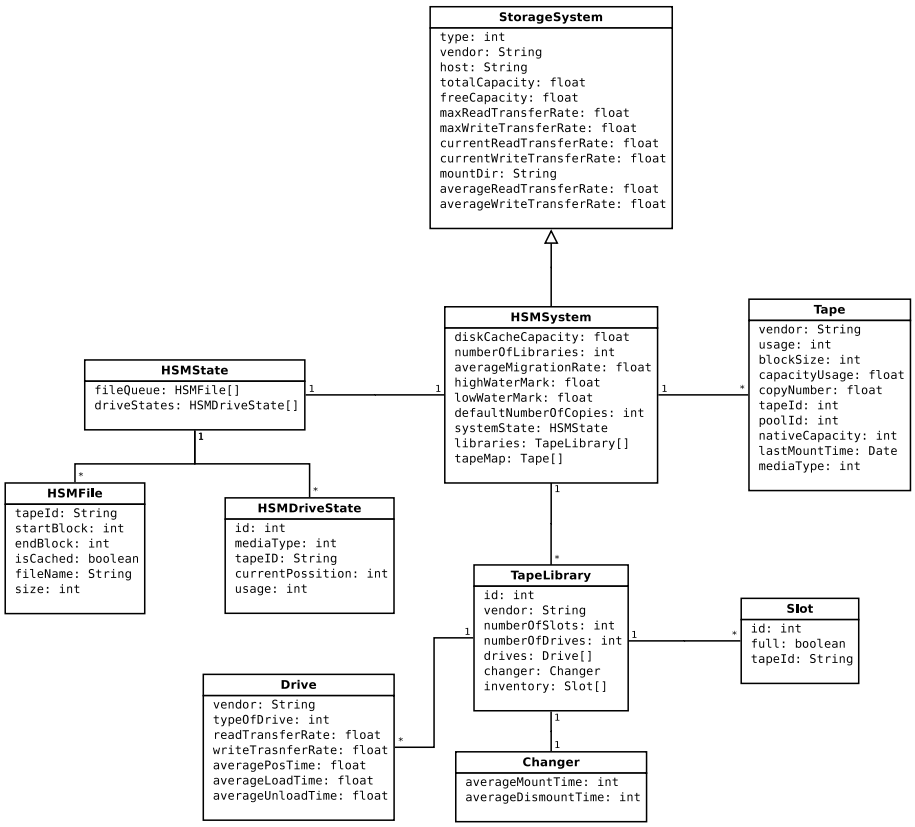

FIG. 3.1. *CMSSM's class diagram for HSM system.*

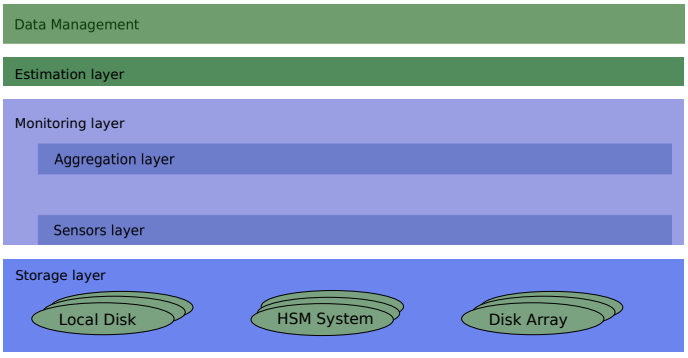

FIG. 3.2. *Layered view of distributed storage system with data access prediction using CMSSM.*

representing complete storage systems (hardware + software) like HSM systems, disk arrays, etc. Sensor layer contains pieces of software which are able to obtain one or more parameters about the given storage system. Generally, sensors are storage system dependent, but in some cases, like for instance a sensor measuring the read/write performance by doing real file I/O, one sensor can fit to more than one types of storage system. Sensors have well defined interface and can be included as plugins to a higher level monitor. Monitoring layer contains of monitors running on the monitored storage node. Monitors have well defined unified interface allowing other services to get selected parameters from them via the network. Requested parameters are sent in data format compatible with CMSSM. Estimation layer contains estimation services, which estimates the performance of storage system or time-to-complete for particular future data transfer. There can be various estimation services using different algorithms and having different estimation accuracy.

Services associated with a certain layer can use services from the lower layers and can bypass layers. For instance management layer service can use directly storage layer tools.

4. Application of CMSSM in NDS. NDS is aimed at building a national storage system providing high quality backup, archiving and data access. The system uses high bandwidth network - Pionier, as a backbone. The system consists of Access Nodes (ANs), which provide the end user interface to system, and storage nodes (SNs) with HSM systems attached. There are also a couple of management nodes. One of the goals of the project is to provide high efficiency based on replication techniques according to user profiles. In the case of data reading the best replica needs to be selected while in the case of data writing the best storage location needs to be chosen. These tasks are completed by the PLBS system briefly described below.

4.1. PLBS. Prediction and Load Balancing System (PLBS) is responsible for load balancing of data access requests among the SNs being part of the NDS. PLBS consists of three subsystems (see Fig.4.1): adopted JMX Infrastructure Monitoring System (JIMS) [25], Advanced Monitoring Database (AMDB) for keeping the values of monitored parameters and Advanced Monitoring and Prediction Daemon (AMPD).

The JIMS based monitoring system consists of Monitoring Agents (JIMS+AMT) installed on every HSM system being part of NDS, and JIMS Gateway collecting data from the agents and storing it to database. JIMS+AMT, JIMS Gateway and AMDB implement the sensors and aggregation layer (see section 3) respectively. The AMPD is responsible for proposing the best replica and location according to the chosen replication policy (see section 5) and implements the estimation layer. One of the requirements for the AMPD is that it must quickly respond, so the user requesting a storage operation does not experience system or data unavailability. Monitoring parameters are measured cyclically by background threads and are stored in the database. In this way the actual parameters (for a certain time interval) can be quickly retrieved from the database and the AMPD can return the results.

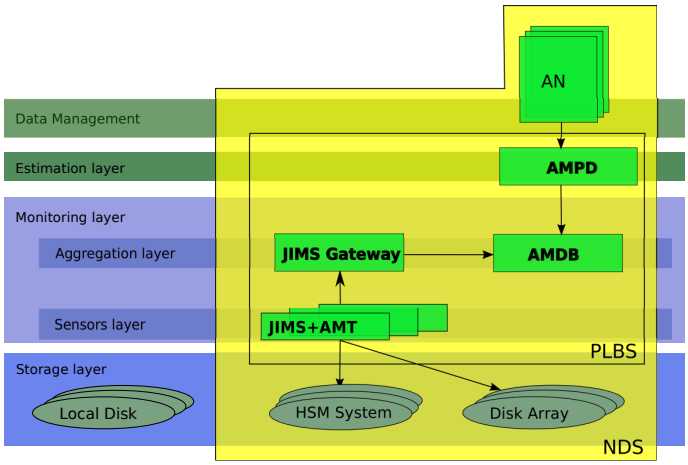

FIG. 4.1. *Application of CMSSM in NDS.*

The HSM monitoring parameters are derived from the CMSSM model described in section 3. The parameters are used to predict the system performance. The parameters are divided into two categories: static parameters changing their values rarely and dynamic parameters changing their values frequently. Part of the model used by the replication policies implemented in PLBS so far is presented below along with the database description.

4.2. The PLBS database–AMDB. The goal of the AMDB is to collect monitored parameters (from the CMSSM model) of distributed nodes in a single place. The approach to store current values of monitored parameters in a database has been chosen, because it allows to completely separate an application logic layer from a monitoring layer.

The AMDB is realized in the standard relational model and conforms to the CMSSM model. The structure of the database is derived from the structure of the monitored systems, which means that the database tables suit to essential HSM components, such as: libraries, drives, pools, tapes, disk cache, etc. The parameters, stored in the database, are divided into two groups: static parameters and dynamic parameters. A simple diagram showing relations between the PLBS database tables is shown in Fig. 4.2. The table columns specification is omitted for simplicity. The dynamic_hsm_parameters_history table stores history of changes of the dynamic

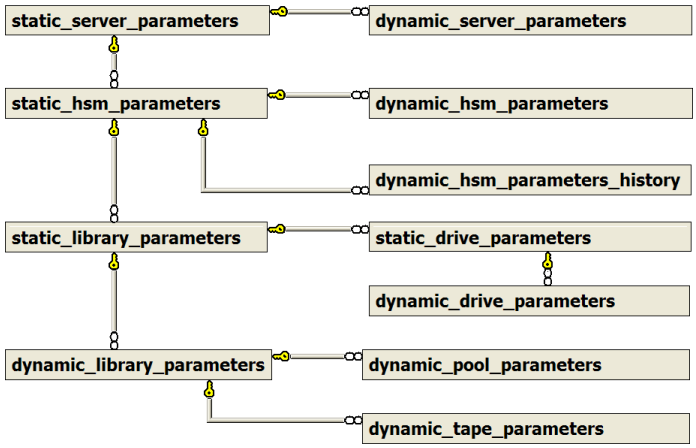

FIG. 4.2. *The AMDB diagram.*

HSM parameters. This table allows the application logic layer to make decisions based not only on the current values of parameters, but also on their statistical values.

Table 4.1 presents summary of the static parameters stored in the AMDB, which are used in the replication policies. Updates of these parameters are performed only on user's demand, for example after a HSM system reconfiguration. Some of these parameters constitutes average values (like average disk cache transfer rate), which are provided by external measurements.

TABLE 4.1
Description of static parameters used in the replication policies.

Parameter name	*Description*	*Implementation*
TotalCapacity	Estimated total capacity of a storage system installed on a single server.	The value of this parameter is estimated as a sum of disk cache capacities
TotalDCCapacity	Total capacity of a single HSM system disk cache.	The value of this parameter is received from the **df** UNIX systems command.
AverageDCReadRate	Estimated value of average disk cache read transfer rate.	The value of this parameter is measured by special benchmarks.
AverageDCWriteRate	Estimated value of average disk cache write transfer rate.	The value of this parameter is measured by special benchmarks.
NumberOfLibraries	Total number of tape libraries connected to a single server.	The value of this parameter is received from configuration files.

Table 4.2 presents summary of the dynamic parameters stored in the AMDB, which are used in the NDS replication policies. These parameters are updated periodically. The update interval is set manually in the PLBS configuration files.

5. Replication policies. The selection of SN for a given data access request is done by heuristic methods taking into account relevant monitoring parameters described in the previous section. Depending on the user profile an appropriate method (called further policy) is used. The AMPD component implements 5 replication policies: round robin—RR, reading in shortest time—R_ST, reading from the minimally loaded device—R_ML, writing replicas of big files—W_BF, writing replicas to the minimally loaded device—W_ML.

TABLE 4.2
Description of dynamic parameters, which are used in the replication policies.

Parameter name	*Description*	*Implementation*
FreeCapacity	Estimated free capacity of a storage system installed on a single server.	The value of this parameter is estimated as a sum of free tapes capacity.
FreeDCCapacity	Free space in a single HSM system disk cache.	The value of this parameter is obtained from the **df** UNIX systems command.
CurrentRate	Transfer rate value from the last measurement.	The value of this parameter is measured by periodically.
HSMLoad	Number of requests waiting or being processed by the HSM system.	The value of this parameter is received from the **dsmq** command for the Tivoli Storage Manager (TSM) systems and from the **fsejob** command for the File System Extender (FSE) systems.

The RR policy is implemented mainly for testing purposes—it does not require monitoring data and it just selects cyclically the next available SN for subsequent requests. The other four policies select the location, for which the value *Loc*, defined in equations 5.1–5.4, is maximized. The R_ST policy is defined by:

$$Loc = \alpha_1 \cdot \frac{RD}{RD_{Max}} + \alpha_2 \cdot \frac{CT}{RD} + \alpha_3 \cdot \frac{1}{1+HL}\,, \tag{5.1}$$

where RD—average disk cache read transfer rate, RD_{Max}—maximal value of average disk cache read transfer rate, taken over all locations, CT—current transfer rate, HL—hsm load, $\sum_{i\in\{1..3\}} \alpha_i = 1$, $\alpha_i > 0$. The exact meaning of these values is given in Tables 4.1 and 4.2.

Equation (5.2) expresses the R_ML policy:

$$Loc = \beta_1 \cdot \frac{ND}{ND_{Max}} + \beta_2 \cdot \frac{1}{1+HL} + \beta_3 \cdot \frac{1}{1+CL}\,, \tag{5.2}$$

where ND—number of drives, ND_{Max}—maximal value of number of drives, taken over all locations, CL—CPU load, $\sum_{i\in\{1..3\}} \beta_i = 1$, $\beta_i > 0$.

Each writing policy determines first whether enough free space is available in a HSM system. Equation 5.3 defines the W_BF policy.

$$Loc = \gamma_i \cdot \frac{FC_{DC}}{TC_{DC}} + \gamma_2 \cdot \frac{FC}{TC} + \gamma_3 \cdot \frac{WR}{WR_{Max}} + \gamma_4 \cdot \frac{1}{1+HL}\,, \tag{5.3}$$

where FC_{DC}—free disk cache capacity, TC_{DC}—total disk cache capacity, FC—free capacity, TC—total capacity, WR—average disk cache write transfer rate, WR_{Max}—maximal value of average disk cache write transfer rate, taken over all locations, $\sum_{i\in\{1..4\}} \gamma_i = 1$, $\gamma_i > 0$.

The policy W_ML is defined by equation 5.4,

$$Loc = \delta_1 \cdot \frac{FC_{DC}}{TC_{DC}} + \delta_2 \cdot \frac{WR}{WR_{Max}} + \delta_3 \cdot \frac{1}{1+HL}\,, \tag{5.4}$$

where $\sum_{i\in\{1..3\}} \delta_i = 1$, $\delta_i > 0$.

α, β, γ and δ are coefficients specifying the impact of the particular monitoring parameters being used in the above formulas. They need to be tuned for the given environment. The above policies are chosen according to the client profile making request and the type of the request. For instance, if the client has defined in the profile that it needs the data as fast as possible than the R_ML policy is chosen.

6. Test results. Four types of tests has been conducted:

- Monitoring influence tests—showing PLBS impact on the performance of the monitored HSM systems,
- Response time tests—showing how fast PLBS responds,
- Load balancing tests—showing data access requests distribution among the storage nodes in multi user and multi requests data access paradigm,
- Throughput tests—showing the total throughput of NDS for selected PLBS replication policy.

The monitoring influence tests are targeted at the JIMS+AMT module while the response time tests are targeted at AMPD module and the both concern the overhead introduced by PLBS to NDS. Load balancing and throughput tests concern the efficiency of data access which can be obtained due to deploying of PLBS to NDS and are provided for a selected policy.

The tests have been conducted using the PLATON [26] infrastructure on which the NDS described above is deployed. The testing environment consists of 5 nodes described in detail in Table 6.1 and connected via Pionier network with 1Gb links. Monitoring agents are installed on every SN while JIMS Gateway, AMPD and AMDB are installed on the kmd2 host (see Sec. 4). The results are presented in the following subsections.

Table 6.1
Test environment nodes

name	location	type	CPU	HSM	drives	HSM cache [TB]
Cyfronet	Krakow	SN	2×Xeon 3.3GHz	IBM TSM	4x LTO	2
PCSS	Poznan	SN	2×Xeon 2.8GHz	IBM TSM	3x LTO	2
WCSS	Wroclaw	SN	2×Xeon 2.8GHz	IBM TSM	2x LTO	1
TASK	Gdansk	SN	2×Xeon 2.8GHz	IBM TSM	3x LTO	0.2
kmd2	Krakow	MN	2×Xeon 2.8GHz	na	na	na

SN—Storage Node, MN—Monitoring Node

6.1. Influence tests. In order for the JIMS to retrieve monitoring data from storage nodes a monitoring agent (JIMS+AMT) (see Fig.4.1) needs to be present on these nodes. The goal of these tests is to measure the influence to performance of storage system when the JIMS+AMT is running on the same node. These tests were performed on the Cyfronet SN (see Table 6.1). This HSM system is in production and the measurements were done during periods of low activity. The main disk storage of the server resides on HP EVA8000 disk array and is attached via 2 FC 2Gb/s links. It is used as disk cache for the HSM system. Repeated patterns of simulated users activities were generated by ftp transfers from other hosts (HSM clients). The JIMS+AMT performed measurements every 10 minutes. Disk reads and writes generated by the measurements had little impact (maximum 5%) on overall execution times of data transfers to and from clients. An example test result is shown in Fig. 6.1

The most influence of JIMS+AMT activity on users data transfers occurs in short periods when the agent measures disk write performance used to calculate AverageDCWriteRate (see Table 4.1). The system utilization statistics come from `sar` program. The user data rates were taken from network traffic statistics as there was no other network traffic on the server during the tests.

6.2. Response tests. Response tests measure the time of processing prediction requests to AMPD. Table 6.2 presents test results for the implemented replication policies. Each value is taken as an average over 5000 requests. We distinguished two cases: (1) the client is on the same machine that AMPD, (2) the client is located remotely to the AMPD component.

We can see that the response times are acceptable for all policies and they do not exceed 102 ms for remote clients and 65 ms for local ones. The network overhead has great influence on the final response times - without it the processing time is shorter by about 37 ms.

6.3. Load balancing test. Load balancing test shows how the requests get distributed among the storage nodes. One monitoring node and four storage nodes have taken part in this test (see Table 6.1). The testing procedure is as follows: First, a set of 100 files has been written to the storage nodes in such way that all files are replicated to all four storage nodes. The file size is 1GiB. Next, a script requesting storage nodes performance prediction and reading data from the appropriate replica is run. The script starts new requests

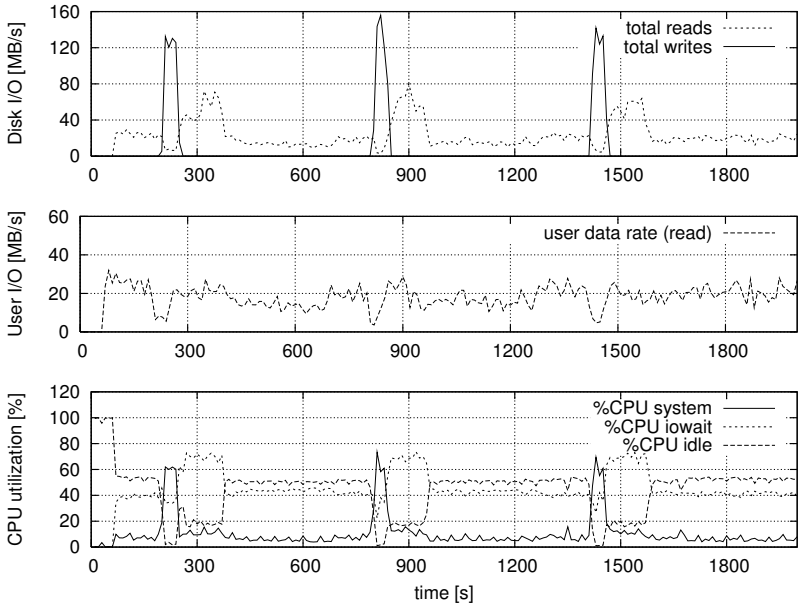

FIG. 6.1. *An example result of monitoring influence test.*

TABLE 6.2
Time of serving prediction requests

Replication policy	Response time [ms]	
	local client	remote client
Reading in shortest time	64.5	101.6
Reading from the minimally loaded device	18.6	55.1
Writing big files	14.7	51.3
Writing to the minimally loaded device	13.3	50.0
Round Robin	11.8	48.8

until 8 concurrent transfers get present. When a transfer is over another request is started. For each test run 2000 requests have been done. The number of requests has been chosen big enough to allow at least few updates of monitoring performance data (used for the prediction) to occur.

It should be noted that result of performance prediction is a sorted list of storage nodes. The nodes are sorted according to the value obtained for the given replication policy. A normalized value (between 0–100) is assigned to each node indicating its relative storage performance. In order to prevent overloading of one storage node (by always sending the request to the best node) the script selects a storage node with a certain probability which is proportional to its current storage performance (obtained from the monitoring and prediction).

Figure 6.2 provides results of prediction tests for the R_ST policy with the following coefficient values: $\alpha_1 = 0.6$, $\alpha_2 = 0.2$, $\alpha_3 = 0.1$. Each point represents the fraction of requests for which a particular host has been selected within the given range of request numbers. The range has been set to 100 requests. At a given moment additional storage load has been issued to the best storage node (PCSS) to study the adaptability of the system in changing environment (mark A). We can see that shortly after that the requests distribution gets reorganized and the TASK storage node gets serving more requests.

We can see that the requests are distributed between the nodes according to their storage processing power—the most powerful storage nodes (*PCSS* and *TASK*) have served the majority of requests.

6.4. Throughput tests. The throughput tests are intended to compare the overall storage throughput of the system for the R_ST replication policy and the RR policy. The testing procedure is the same as the one for the load balancing tests. The tests have been conducted for idle and loaded storage devices. The results are presented in Table 6.3. We can see that the R_ST policy is much better than the RR policy especially

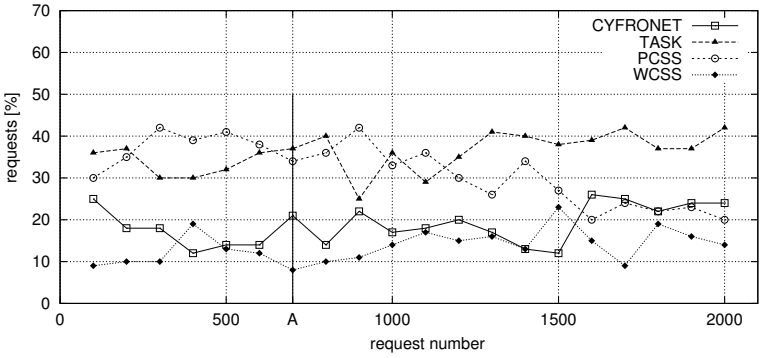

FIG. 6.2. *Replication tests for R_ST policy*

if additional load is put on the system. We can also observe that for the R_ST policy the throughput is not influenced by the additional storage load. It is because the load gets distributed proportionally to the other nodes which are still having unused storage performance.

TABLE 6.3
Storage throughput

Replication policy	Storage throughput [MiB/s]	
	idle	loaded
R_ST	325	327
RR	233	101

7. Summary and future work. In this paper the application of CMSSM in the national distributed storage system, NDS, has been described. The PLBS subsystem being a part of the NDS system and providing advanced monitoring and prediction functionalities has been presented. The system makes use of replication techniques to increase availability and performance of data access. Monitoring parameters, methods for retrieving them and replication policies have been described. The influence tests showed that the monitoring did not cause essential storage system performance degradation. The system response times are within the tens of milliseconds range which is satisfying. Load balancing test shows that requests get distributed between the nodes proportionally according to their storage processing power. Our future plans focus on using CMSSM and its ontological representation in knowledge supported distributed storage system with quality of service.

Acknowledgments. This research is partially supported by the MNiSW grant nr N N516 405535 and PLATON project POIG.02.03.00-00-028/08. AGH-UST grant nr 11.11.120.865 is also acknowledged. Thanks go to: prof. K. Zieliński for giving access to the JIMS software, M. Brzeźniak for support with PLATON infrastructure, M. Jarząb for support with JIMS, PLATON partners for sharing storage resources.

REFERENCES

[1] National Data Storage project, Polish MNiSW grant nr R02170170170172055 03, `https://kmd.pcss.pl`
[2] Pionier—Polish Optical Internet, `http://www.pionier.gov.pl`
[3] D. Nikolow, R. Slota, J. Kitowski, "Knowledge Supported Data Access in Distributed Environment", Proc. of CGW'08, October 13-15 2008, ACC-Cyfronet AGH, 2009, Krakow, pp. 320-325.
[4] J.H. Han, D.H. Lee, H. Kim, H. P. In, H.S. Chae, Y.I. Eom "A situation-aware cross-platform architecture for ubiquitous game", Computing and Informatics, vol. 28, nr 5, 2009, pp. 619-633
[5] Funika, W. - Korcyl, K. - Pieczykolan, J. - Skital, L. - Balos, K. - Slota, R. - Guzy, K. - Dutka, L. - Kitowski, J. - Zieli1717ski, K. "Adapting a HEP Application for Running on the Grid", Computing and Informatics, vol. 28, nr 3, 2009, pp. 353-367
[6] Słota, R., Nikolow, D., Kuta, M., Kapanowski, M., Skałkowski, K., Pogoda, M., Kitowski, J., "Replica Management for National Data Storage", Proceedings PPAM09, LNCS 6068, Springer, in print.
[7] Chai, E., Matsumoto, K., Uehara, M., Mori, H., "Virtual Large-scale Disk Base on PC GRID", Scalable Computing: Practice and Experience, vol. 10, nr 1, SWPS, 2009, pp.87-98.
[8] Srinivas, A., Janakiram, D., "Data Management in Distributed Systems: A Scalability Taxonomy", Scalable Computing: Practice and Experience, vol. 8, nr 1, SWPS, 2007, pp.115-130.

[9] Ranganathan K., Foster I.: "Identifying Dynamic Replication Strategies for a High-Performance Data Grid". in: Proc. Int. Workshop on Grid Computing, Denver, Nov. 2001.
[10] Lamehamedi H., Szymański B., Deelman E., "Data Replication Strategies in Grid Environments", in: IEEE Computer Science Press, Los Alamitos, CA, 2002, pp. 378-383.
[11] Park S., Kim J., Ko Y., Yoon W., "Dynamic Data Grid Replication Strategy Based on Internet Hierarchy", LNCS 3033, Springer, 2004, pp.838-846.
[12] Rahman R.M., Barker K., Alhajj R., "A Predictive Technique for Replica Selection in Grid Environment", in: 7-th IEEE Int. Symp. on Cluster Computing and the Grid, IEEE Computer Society, 2007.
[13] Li, J., "A Replica Selection Approach based on Prediction in Data Grid", in: Proc. Third Int. Conf. on Semantics, Knowledge and Grid - SKG2007, 29-31 Oct. 2007, Xi'an, Shan Xi, China, pp. 274-277.
[14] Rahman R.M., Barker K., Alhajj R., "Replica placement Strategies in Data Grid", in: J Grid Computing (2008) 6:103-123, Springer Science + Business media B.V. 2007.
[15] Rasool, Q., Li, J., Oreku, G.S., Zhang, S., Yang, D., "A load balancing replica placement strategy in Data Grid", Third IEEE Int. Conf. on Digital Information Management (ICDIM), Nov. 13-16, 2008, London, UK, Proc. IEEE 2008, pp. 751-756.
[16] Słota, R., Skitał, L., Nikolow D., Kitowski J., "Algorithms for Automatic Data Replication in Grid Environment", in: LNCS, 3911, Springer, 2006, pp. 707-714.
[17] D. Nikolow, R. Słota, and J. Kitowski, "Grid Services for HSM Systems Monitoring", LNCS 4967, Springer, 2008, pp.321-330.
[18] Nagios, `http://www.nagios.org/`
[19] Ganglia monitoring system, `http://ganglia.sourceforge.net`
[20] B. Baliś, M. Bubak and B. Labno, GEMINI: Generic Monitoring Infrastructure for Grid Resources and Applications, Proc. of Cracow'06 Grid Workshop, Oct 15-18, 2006, Cracow, Poland, ACC Cyfronet AGH, 2007, pp. 60-73.
[21] Distributed Management Task Force, `http://www.dmtf.org`
[22] Common Information Model, `http://www.dmtf.org/standards/cim/`.
[23] Storage Management Initiative Specification (SMI-S), `http://www.snia.org/tech_activities/standards/curr_standards/smi`.
[24] Grid Laboratory Uniform Environment (GLUE), `http://forge.gridforum.org/sf/projects/glue-wg`.
[25] Zieliński, K., Jarząb, M., Balos, K., Wieczorek, D., "Open Interface for Autonomic Management of Virtualized Resources in Complex Systems—Construction Methodology", in: FGCS, vol. 24, Issue 5, May 2008, pp. 390-401.
[26] PLATON—Service Platform for e-Science, `http://www.platon.pionier.net.pl/`.

Edited by: Norbert Meyer
Received: March 30, 2010
Accepted: June 18, 2010