# Storage QoS Aspects in Distributed Virtualized Environments

Darin Nikolow*, Renata Słota*, Jacek Kitowski*†
*Department of Computer Science, AGH-UST,
al. Mickiewicza 30, 30-059, Kraków, Poland
†Academic Computer Center CYFRONET-AGH,
ul. Nawojki 11, 30-950 Kraków, Poland
email: {darin,rena,kito}@agh.edu.pl
phone: (+48 12) 617 3964, fax: (+48 12) 338 054

***Abstract*—Storage performance of a single virtual machine in a cloud computing environment may be affected by other machines using the same physical storage. At the same time, user requirements concerning quality of service continue to increase, which brings new challenges for virtualized environments. In this paper we present the results of our research concerning data storage QoS. We discuss the quality aspects of storage services and propose a method for ensuring storage QoS by using monitoring and performance prediction based on heuristics. We also propose several heuristic policies, which can be implemented in a data transfer scheduler. Finally, we discuss test results obtained in a virtualized environment.**

***Keywords-Quality of Service; Service Level Agreement; Storage Cloud; virtualization.***

## I. INTRODUCTION

Virtualization technologies are becoming extremely popular. One of the reasons behind this evolution is the increased flexibility in creating new computing environments for research, development or production. Another reason is that virtualization supports green computing by allowing for more efficient use of physical resources – this, in turn, reduces the need for electrical energy.

Since virtual machines share physical resources, the performance of a single machine may be affected by other machines running on the same host. It should also be noted that user requirements concerning QoS (Quality of Service) continue to increase, bringing new challenges for virtualized environments.

In the case of scientific applications (such as simulations, experiments in high energy physics or out-of-core computations) where the application needs to access huge amounts of data, as well as in the scope of backup and archiving services (for example storage clouds), proper management of data (or files), supporting QoS with efficient storage performance utilization, remains essential. Taking into account the dynamic nature of virtualized environments and the diversity/heterogeneity of storage systems (such as Hierarchical Storage Management – HSM and disk arrays), monitoring is required to cope with the problem of delivering data with appropriate QoS. Relevant policies for controlling and distributing data transfer requests are necessary for achieving QoS.

This paper presents the results of our research concerning storage QoS delivery. We discuss quality aspects of data storage services and propose a method of achieving storage QoS by using monitoring and performance prediction based on heuristics. We also present several heuristic policies, which can be implemented in a data transfer scheduler and discuss test results obtained in a virtualized environment.

The presented research forms part of a more general project called OntoStor [1], which deals with data access optimization. As a result of the project, an object-based mass storage model for HSM systems and disk arrays, called CMSSM (Common Mass Storage System Model) [2] has been proposed. A CIM (Common Information Model) version of CMSSM is also available. Implementation of monitoring sensors basing on the proposed model allows us to create data access estimators required to ensure storage QoS via proper management of storage resources.

This paper is organized as follows. The following section presents the state of the art. Subsequently, methods for achieving storage QoS are described. Section 4 describes test results, while the final section summarizes our research contributions.

## II. STATE OF THE ART

As virtualization technologies are becoming more mature, a Cloud computing paradigm has evolved [3]. In terms of data storage, we can distinguish two types of clouds: the Storage Cloud (providing services for file-based or block-based access to data) and the Data Cloud (providing services for database-oriented access).

Examples of storage clouds include the Hadoop distributed file system [4], Amazon's S3 storage cloud [5], Google file system [6], Sector [7], and Chelonia [8].

Some studies on storage clouds involve high-performance data access, which is correlated with our research. Storage clouds, as well as data clouds, can be used as a data access layer for cloud computing. In [7], an implementation of the Sector data cloud is presented. Sector is a high-performance data cloud, which can operate on geographically-distributed data shared between data centers.

 ISBN: 978-1-61208-106-9 

A study concerning data access prediction has been presented in [9]. The authors define five elements of the I/O path essential for end-to-end storage performance prediction, from device I/O scheduling through the cache and network layers, to client-side buffering. The aim of this research is to reduce the complexity of prediction. For example, in [10] and [11], the SAN (Storage Area Network) is addressed. The former paper [10] shows how performance management can be approached in IBM TotalStorage products and how it has evolved. In the latter one [11], SAN configuration middleware is introduced. The middleware aids the management of heterogeneous SAN deployments to meet customer SLA Service Level Agreement).

There are some papers concerning storage QoS. In [12], an I/O scheduling algorithm for managing the performance of an underlying storage device is presented. Scheduling is performed in such a way as to enable multiple users (or applications) to obtain the requested transfer-rate QoS.

In [13], a two-level framework for I/O scheduling, using monitoring and queuing theories, is presented. The proposed scheduler maintains QoS in terms of latency and transfer rates. Monitoring of underlying storage devices is used to provide feedback to the I/O scheduling algorithm.

In [14], a method for optimizing the workload distribution in storage area networks in order to meet SLA is presented. The method is static in the sense that the result of optimization is a concrete SAN configuration, not subject to frequent changes.

In [15], a method for storage performance insulation of services sharing common storage servers is proposed. The system automatically configures prefetching, write-back and cache partitioning to achieve storage performance for a given service, which is independent from the I/O activity of other services sharing the same storage.

In [16], the Chameleon framework capable of providing predictable performance for multiple clients sharing a common storage infrastructure is introduced. It bases on a black-box approach for capturing the behavior of the underlying storage system.

The presented studies show that the problem of achieving QoS in distributed storage environments is not a trivial task. Given the changing state of the environment, relevant monitoring is required. In our previous studies we have proposed a Common Mass Storage System Model for HSM systems [17], disk arrays and local disks. The aim of this paper is to present a unified monitoring layer, which can be used by estimation services for prediction of data access to mass storage systems [18][19]. The proposed model has been used in the NDS (National Data Storage) [18] and Platon [20] projects as well as in the PL-Grid project [21].

## III. Problem Solution Description

We assume that the data needed by applications running in the Cloud is stored in a distributed storage system where it can be replicated for performance or availability reasons. The nodes of the storage system can also be located in the Cloud as VMs (Virtual Machines), sharing common resources. Applications may have various requirements concerning QoS. Some applications, such as HEP tools, which process data in the context of ongoing experiment require write QoS in order to prevent buffer overflows and data loss. In contrast, read QoS is needed by applications, which deliver on-demand multimedia content and by interactive applications dealing with large data sets, for example medical visualizations. In the case of write QoS, a decision on which storage node to use has to be made, while in the case of read QoS, an existing replica needs to be selected. Our vision of QoS-aware distributed storage environment is presented below.

### A. QoS-aware distributed storage environment - a vision

The concept of a QoS-aware distributed storage environment is shown in Figure 1. The mass storage system layer may include divergent storage systems (HSM systems, disk arrays, local hard disks). They can be either complete systems with appropriate management software (HSM, disk arrays) or single storage devices such as local disks. The sensor layer provides a unified interface for monitoring of storage performance parameters, facilitating the development of storage-independent upper layer services. The estimation layer contains estimation services, which can be used to predict the performance of a storage node at a given moment. The estimation services use monitoring data provided by the sensor layer and can apply various algorithms (more or less sophisticated) to predict the performance of a given storage node. The data management layer contains services needed for efficient access to data. Examples of such services include the meta-catalog, the replica manager and the QoS scheduler.

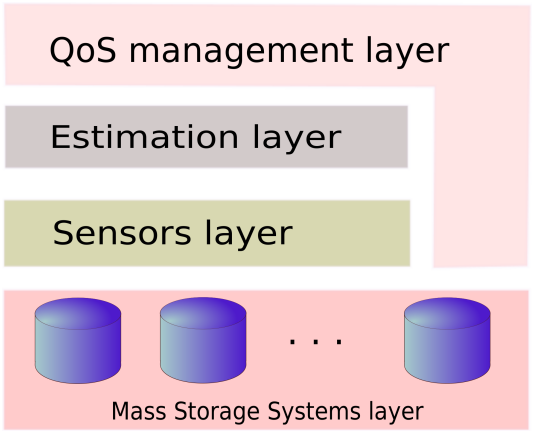



Figure 1. QoS-aware distributed Storage Environment - a vision

The problem that we address in this paper concerns efficient storage resource allocation. In order to effectively

 ISBN: 978-1-61208-106-9 

use the available storage resources in terms of capacity and bandwidth, the storage system has to make decisions about scheduling and allocating storage nodes with QoS in mind. Storage-related QoS requirements include transfer rate and latency time. Several types of transfer rate can be specified in this context: current, average, moving average, etc. The QoS requirements used in our tests involve average transfer rates for a given transfer. For each data transfer request, a storage node is selected from a prepared list. The list only contains storage nodes suitable for the current request. In the case of a write request, it can include all nodes with the required level of availability or storage cost. In the case of read requests, it is limited to nodes containing replicas of the requested file. If the storage system is unable to fulfill the QoS for a given request, the request is postponed.

The proposed method of resource allocation with QoS uses bandwidth reservation along with storage performance monitoring. Such monitoring is necessary since the storage resources can be shared among VMs (for example, high-capacity disk arrays in a storage area network) and their performance can not be predicted based on reservations only. In addition, if an application does not fully exhaust its requested (or reserved) bandwidth, a wastage of storage performance may occur. Using proper monitoring we can detect such wastage and, conditionally, allocate bandwidth to other ongoing transfers. The proposed heuristic-based policies are presented below. They can be implemented in a QoS scheduler and shared by the management and estimation layers (see Figure 1).

### B. Resource allocation policies with storage QoS delivery

The proposed monitoring-based heuristic policies designated SM-R1, SM-R2, SM-R3 are described below. Their usefulness has been tested (see next section) and compared with policies, which do not employ monitoring, i.e., RR (Round Robin) and StaticQoS.

The following monitoring-based heuristic policies have been considered:

- SM-R1 - scheduling with monitoring using the following rule to check if a node can meet the QoS requirements of a given transfer:

$$max - res\_ban \times k_1 > rb, \tag{1}$$

  where $max$ is the peak transfer rate for the requested operation (read or write) concerning the given disk storage resource, $res\_ban$ is the amount of reserved bandwidth, $k_1$ is the overhead coefficient and $rb$ is the requested bandwidth,
- SM-R2 - for this policy the rule is:

$$max - ma_{10} \times k_2 > rb, \tag{2}$$

  where $ma_{10}$ is the moving average covering the last 10 transfer rate measurements for the given storage resource while $k_2$ is the overhead coefficient,
- SM-R3 - for this policy the checking rule is:

$$\frac{ma_{10}}{np+1} > rb \times k_3, \tag{3}$$

  where $np$ is the number of scheduled active data transfers and $k_3$ is the overhead coefficient.

The monitoring-independent resource allocation policies are as follows:

- RR - round robin. Here, the next available storage node is selected for the current transfer.
- StaticQoS - the number of allowed concurrent transfers for the given node is limited and static. The limit $\lambda$ is calculated using the following formula:

$$\lambda = \frac{BSN_{max}}{rb \times k_4}, \tag{4}$$

  where $BSN_{max}$ is the peak bandwidth for a storage node and $k_4$ is the concurrent transfer overhead coefficient.

## IV. Storage QoS test results

This section presents our test environment and the results of tests involving the previously-described policies. Tests have been conducted under idle and loaded conditions.

### A. Test environment

Our testing environment consists of three storage nodes, each of which is a virtual machine with dedicated storage. All VMs are hosted by the same physical machine, thus sharing its hardware resources. The physical machine is part of the Cloud environment. This test environment allows us to study the manageability of storage QoS given the sharing of resources due to virtualization and the heterogeneity of storage systems. We should notice that hosting too many storage node VMs on a single physical machine may cause the I/O bus of the physical host to become a bottleneck, thereby wasting the performance capabilities of the attached storage systems.

The testing environment is presented in Figure 2. The first node is a client of the Lustre cluster filesystem, the second has a disk array volume mounted and the last one uses a local logical volume. The measured bandwidth of the attached storage is as follows: 20 MB/s for the Lustre FS, 300 MB/s for the disk array and 60 MB/s for the local disk. This gives us the peak total throughput of 380 MB/s. The low bandwidth of Lustre is due to the network link of OST (Object Storage Target). On every node a monitoring daemon has been started in order to gather storage performance statistics (peak, current and moving-average transfer rates).

The testing procedure was as follows:

1) select the most appropriate storage node for a transfer according to the tested policy (if there is no node capable of satisfying the QoS requirement then wait until there is one). The QoS requirement for all transfers has been set at 10 MB/s,

 ISBN: 978-1-61208-106-9 

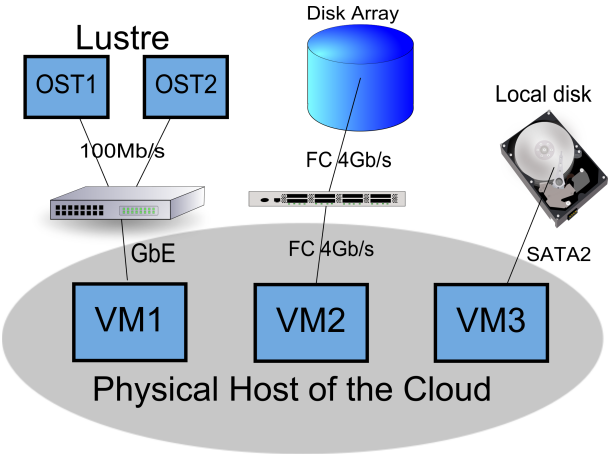



Figure 2. Test environment for storage QoS.

2) fork a write data transfer process for the selected node,
3) sleep 1 second (a configurable parameter),
4) repeat previous steps until the defined number of iterations is reached. For the presented results the number of iterations is set at 100. The number of concurrent transfers depends on the tested policy, e.g., the RR policy can potentially launch many more transfers then other policies since no condition is checked.

### B. Results for idle system

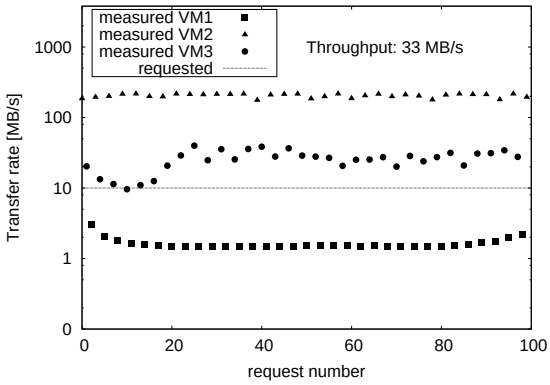
The test results are presented in Figure 3 and Figure 4.

We can see that in the case of the RR policy (see Figure 3(a)) most transfers did not receive the required bandwidth. In this case, we are dealing with very inefficient use of storage resources. Since all three SNs have divergent storage devices attached, we see three separate lines, which depict the performance of each device. The throughput is very low and limited by the device with the worst performance. In addition, sending more concurrent transfers further degrades the overall performance.

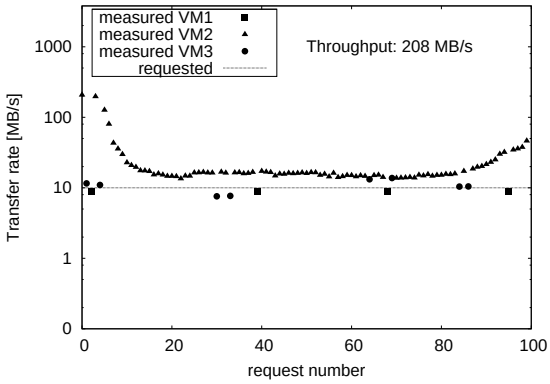
In the case of a static QoS policy (see Figure 3(b)) we can see that only several transfers are below the required bandwidth (though their performance remains quite close to expectations). Total throughput is much better than in the RR case. This simple policy may already be deemed satisfactory, but it might be possible to achieve even higher throughput.

In the case of the SM-R1 policy (see Figure 4(a)), in which monitoring is used along with resource reservation, we have high total throughput but more requests fall below the line compared to static QoS. This is a consequence of the fact that with concurrent transfers, the throughput of storage devices is usually lower than given a single transfer. The number of concurrent transfers for a given storage device is not monitored and so its influence is not predicted.

The SM-R2 policy (see Figure 4(b)) yields very low performance because the rule remains true even when many




(a) Round robin




(b) Static QoS

Figure 3. Test result for policies with no monitoring

concurrent transfers are ongoing. The average throughput $ma_{10}$ (see Subsection III-B) is highly degraded and cannot approach $max$. This degradation is caused by the fact that data transfer processes need to compete for CPU time instead of waiting for I/O access.

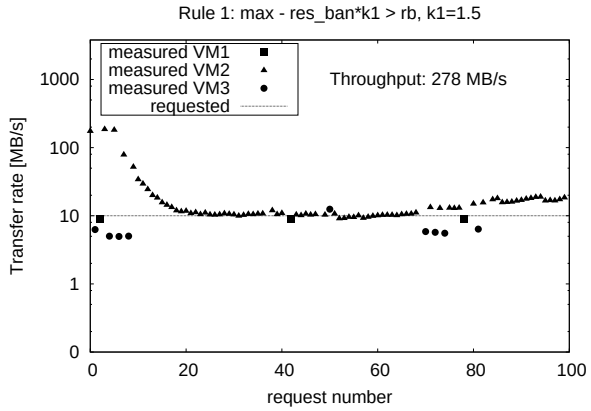
In the final case, SM-R3 (see Figure 4(c)), we have obtained the best results, but it should be mentioned that the rule proves false if there are no ongoing transfers - $ma_{10} = 0$. That is why an additional condition was introduced: if no transfers are present then start one.

### C. Results for loaded conditions

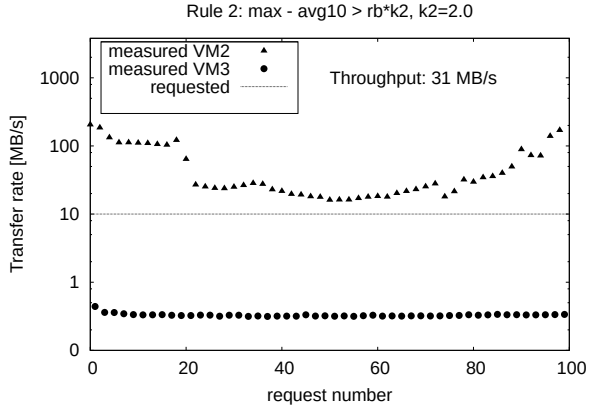
The same tests have been run in an environment where the disk was subjected to additional, external load.

The result for static QoS and SM-R3 are presented in Figure 5. These policies have been selected since they give the best results for idle storage. As can be seen, the extra load has a heavy impact on the static-policy QoS while in the case of SM-R3 it does not significantly increase the number of transfers, which fall bellow the 10MB/s line (the requested bandwidth). It should be noticed that even if the transfer rate is below 10MB/s it is still quite close to the required value. The total throughput for the SM-R3 policy

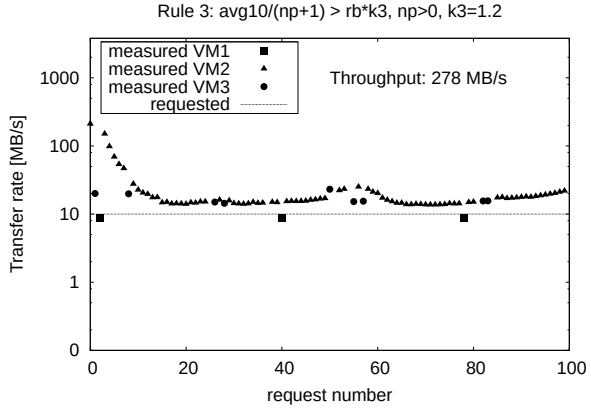
 ISBN: 978-1-61208-106-9 




(a) SM-R1

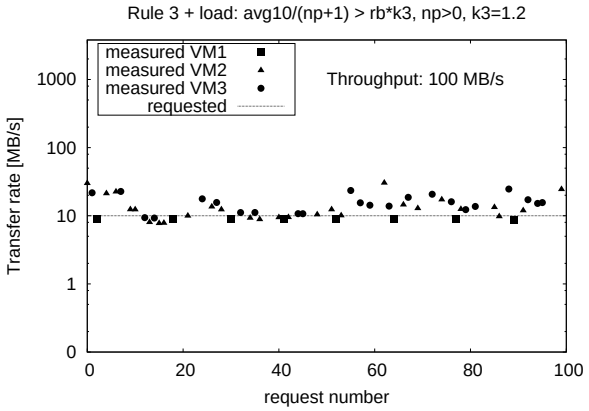



(b) SM-R2

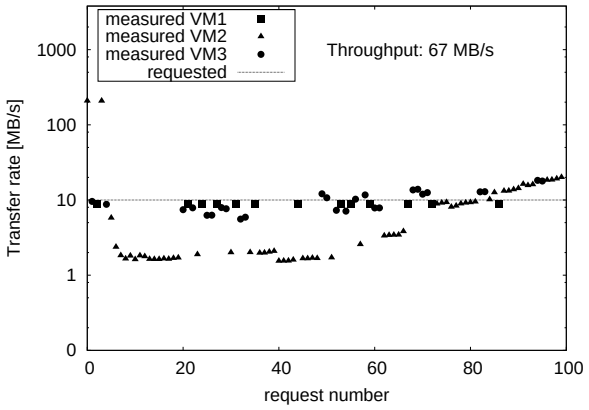



(c) SM-R3

Figure 4. Test result for policies using monitoring




(a) SM-R3 with external load




(b) Static QoS with external load

Figure 5. Test result for selected policies with external storage load

is also much better.

## V. Conclusion and Future Work

This paper presented the results of our research concerning storage QoS delivery. A method of achieving storage QoS by using monitoring and performance prediction based on heuristics is proposed and several heuristic policies are defined. Test results for a virtualized environment show that scheduling data transfers with the aid of monitoring policies can yield better results than without monitoring. Policies with static resource reservation do not perform well in a virtualized environment with shared resources, similar to a storage cloud. In such cases, storage performance monitoring and prediction is a must, especially when managing storage resources with QoS (or SLA). The problem is not trivial given the heterogeneity of storage systems and the lack of relevant performance parameters in modern information models like CIM and GLUE (Grid Laboratory for a Uniform Environment). While methods of performance monitoring and estimation, as well as their impact on storage systems have been discussed in our previous work, this paper presents a vision of how they can be applied for the purpose of delivering storage QoS.

**Acknowledgments**

This research is supported by Polish MNiSW grant no. N N516 405535. The AGH-UST grant no. 11.11.120.865 is also acknowledged.

 ISBN: 978-1-61208-106-9 

## REFERENCES


[1] OntoStor project. [Online]. Available: http://www.icsr.agh.edu.pl/ontostor/17.08.2010

[2] "Common Mass Storage System Model - project Ontostor internal report," 2009. [Online]. Available: http://www.icsr.agh.edu.pl/trac/ontostor/browser/CMSSM/raport.odt17.08.2010

[3] K. Kant, "Data center evolution," *Comput. Netw.*, vol. 53, no. 17, pp. 2939–2965, 2009.

[4] Welcome to Apache Hadoop! [Online]. Available: http://hadoop.apache.org/17.08.2010

[5] Amazon Simple Storage Service (Amazon S3). [Online]. Available: http://aws.amazon.com/s3/17.08.2010

[6] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 29–43, 2003.

[7] Y. Gu and R. L. Grossman, "Sector and Sphere: the design and implementation of a high-performance data cloud," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 367, no. 1897, pp. 2429–2445, 2009. [Online]. Available: http://rsta.royalsocietypublishing.org/content/367/1897/2429.abstract

[8] J. K. Nilsen, S. Toor, Z. Nagy, B. Mohn, and A. L. Read, "Performance and Stability of the Chelonia Storage Cloud," *CoRR*, vol. abs/1002.0712, 2010.

[9] D. O. Bigelow, S. Iyer, T. Kaldewey, R. C. Pineiro, A. Povzner, S. A. Brandt, R. A. Golding, T. M. Wong, and C. Maltzahn, "End-to-end performance management for scalable distributed storage," in *PDSW*, G. A. Gibson, Ed. ACM Press, 2007, pp. 30–34.

[10] S. Gopisetty, S. Agarwala, E. Butler, D. Jadav, S. Jaquet, M. Korupolu, R. Routray, P. Sarkar, A. Singh, M. Sivan-Zimet, C.-H. Tan, S. Uttamchandani, D. Merbach, S. Padbidri, A. Dieberger, E. M. Haber, E. Kandogan, C. A. Kieliszewski, D. Agrawal, M. Devarakonda, K.-W. Lee, K. Magoutis, D. C. Verma, and N. G. Vogl, "Evolution of storage management: transforming raw data into information," *IBM J. Res. Dev.*, vol. 52, no. 4, pp. 341–352, 2008.

[11] D. M. Eyers, R. Routray, R. Zhang, D. Willcocks, and P. Pietzuch, "Towards a middleware for configuring large-scale storage infrastructures," in *MGC '09: Proceedings of the 7th International Workshop on Middleware for Grids, Clouds and e-Science*. New York, NY, USA: ACM, 2009, pp. 1–6.

[12] T. M. Wong, R. A. Golding, C. Lin, and R. A. Becker-szendy, "Zygaria: storage performance as a managed resource," in *In IEEE Real Time and Embedded Technology and Applications Symposium (RTAS 06*, 2006, pp. 125–134.

[13] J. Zhang, A. Sivasubramaniam, Q. Wang, A. Riska, and E. Riedel, "Storage performance virtualization via throughput and latency control," *Trans. Storage*, vol. 2, no. 3, pp. 283–308, 2006.

[14] E. Gencay, C. Sinz, and W. Kuchlin, "Towards SLA-based optimal workload distribution in SANs," in *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*, 7-11 2008, pp. 755 –758.

[15] M. Wachs, M. Abd-el-malek, E. Thereska, and G. R. Ganger, "Argon: Performance insulation for shared storage servers," in *In Proceedings of the 5th USENIX Conference on File and Storage Technologies. USENIX Association*. USENIX Association, 2007, pp. 61–76.

[16] S. Uttamchandani, L. Yin, G. A. Alvarez, J. Palmer, and G. Agha, "CHAMELEON: a self-evolving, fully-adaptive resource arbitrator for storage systems," in *ATEC '05: Proceedings of the annual conference on USENIX Annual Technical Conference*. Berkeley, CA, USA: USENIX Association, 2005, pp. 75–88.

[17] D. Nikolow, L. R. Slota, and J. Kitowski, "Grid Services for HSM Systems Monitoring," in *in: R. Wyrzykowski, J. Dongarra, K. Karczewski, and J. Wasniewski (Eds.), Proceedings of 7-th International Conference, PPAM 2007*. Springer, 2008, pp. 321–330.

[18] R. Słota, D. Nikolow, S. Polak, M. Kuta, M. Kapanowski, K. Skałkowski, M. Pogoda, and J. Kitowski, "Prediction and Load Balancing System for Distributed Storage," *Scalable Computing: Practice and Experience*, vol. 11, no. 2, pp. 121–130. [Online]. Available: http://www.scpe.org

[19] D. Nikolow, R. Slota, J. Marmuszewski, M. Pogoda, and J. Kitowski, "Disk Array Performance Estimation," in *Proceedings of Cracow Grid Workshop - CGW'09*. ACC-Cyfronet AGH, 2010, pp. 287–294.

[20] PLATON - Service Platform for e-Science. [Online]. Available: http://www.platon.pionier.net.pl/online/?lang=en17.08.2010

[21] PL-Grid Polish Grid Initiative. [Online]. Available: http://www.plgrid.pl/en17.08.2010


 ISBN: 978-1-61208-106-9 