Computer Science • 13 (1) 2012 http://dx.doi.org/10.7494/csci.2012.13.1.63

Renata Słota
Dariusz Król
Kornel Skałkowski
Michał Orzechowski
Darin Nikolow
Bartosz Kryza
Michał Wrzeszcz
Jacek Kitowski

A TOOLKIT FOR STORAGE QOS PROVISIONING FOR DATA-INTENSIVE APPLICATIONS

Abstract *This paper describes a programming toolkit developed in the PL-Grid project, named QStorMan, which supports storage QoS provisioning for data-intensive applications in distributed environments. QStorMan exploits knowledge-oriented methods for matching storage resources to non-functional requirements, which are defined for a data-intensive application. In order to support various usage scenarios, QStorMan provides two interfaces, such as programming libraries or a web portal. The interfaces allow to define the requirements either directly in an application source code or by using an intuitive graphical interface. The first way provides finer granularity, e.g., each portion of data processed by an application can define a different set of requirements. The second method is aimed at legacy applications support, which source code can not be modified. The toolkit has been evaluated using synthetic benchmarks and the production infrastructure of PL-Grid, in particular its storage infrastructure, which utilizes the Lustre file system.*

Keywords data-intensive application, storage management, QoS, Grid

1. Introduction

For many years scientists around the world have been conducting simulations and experiments contributing to large-scale studies in high-energy physics, biomedicine and other disciplines. In order to make their applications more efficient and accurate, those people are looking for technologies of handling data in a more systematic and controlled manner because their application requirements concerning computational and storage infrastructure are continuously growing. Furthermore, with the emergence of globally distributed computing resources and increase in data output from scientific experiments, scientists perceive the necessity of handling data in conjunction with computational tasks. Rapid increase of software complexity entails the expansion of both computational load and produced data volume. Since most of the existing middlewares and tools in distributed computing environments like Grids or Clouds address the problem of efficient provisioning of computing resources, I/O operations become a bottleneck of many applications running on. The issues associated with efficient access to storage resources emerge especially in applications which perform computations concurrently with file operations. For such applications efficient data access constitutes a crucial factor, highly affecting their execution time.

In this paper we present a toolkit named QStorMan (an acronym that stands for Quality-based Storage Management), a set of tools which can support data-intensive applications. The toolkit allows users to take advantage of storage QoS provisioning, i.e. facilitating appropriate quality of access to storage resources, for Grid applications that heavily utilize storage resources based on defined non-functional requirements. It constitutes a part of a larger framework, called Framework for Intelligent Virtual Organization (FiVO) [1] which supports the definition of a Virtual Organization (VO) starting from VO contract negotation, automatic deployment and management of the VO according to the defined contract. The QStorMan toolkit was developed within the PL-Grid project [2], which is a Polish National Grid initiative aimed at supporting Polish science by providing dedicated hardware infrastructure and sophisticated software for various science disciplines. Although, the data management approach proposed by QStorMan is orthogonal to the actual storage infrastructure, its current implementation utilizes the Lustre file system [12] to manage the data, which can work in any distributed system, e.g. clusters, Grids or Clouds.

The rest of the paper is organized as follows. In Section 2, the most important features of data-driven computing are described. Section 3 discusses several existing solutions related to optimization of data access time and QoS assurance. In Section 4, we briefly describe the QStorMan toolkit, its components, and usage examples for supporting data-intensive applications. Next, in Section 5 an experimental evaluation of the toolkit is presented. Finally, Section 6 concludes the paper and discusses future work.

2. Data-driven computing

Today, many newly developed applications produce and analyze data in the amounts of terabytes and more. Examples of such applications can be Google Apps, i.e., search engine, YouTube and Gmail, which everyday process over 20 PB of data [3] or German Climate Computing Center (DKRZ) which has over 60 PB of climate data for analysis [4]. Other, more science-related examples can be find in [5], [6] and [7] Such a kind of applications is often referred to as *data-intensive* applications. The most characteristic feature of this sort of applications is higher workload of storage resources than utilization of computing power. In other words, if the requirements of a resource bandwidth for data transfer highly outweigh an application's computational requirements, then the application can be classified as a data-intensive one. Applications which are characterized by such a feature usually execute a repeated sequence of file operations, like reading, processing, and dumping computations results. Obviously, some applications perform mostly write or mostly read operations, and in that case only high write or read transfer rate is required.

A special type of data-intensive applications constitute the *out-of-core* applications [8, 9] which operate on very large data structures like trees or multi-dimensional arrays. Data structures processed by those applications are so large, that it is impossible to process them wholly using only a system memory. In order to overcome these limitations, only a part of the needed data is loaded into the memory, while the rest remains on storage devices. Since most storage devices are much slower than system memory, the described approach requires more time to load, transfer or store the data in comparison with the execution of computational parts of an application's workflow.

3. Related Work

The related research studies discussed in this section are mainly focused on two aspects, which are important for supporting data-intensive applications:

1. data access time optimization for distributed applications,
2. Quality of Service (QoS) support for data storage systems.

Regarding the first aspect, many researchers investigate performance of data access in MPI applications. In [10] a method for increasing the efficiency of MPI-IO data access for applications using Gfarm File System is proposed. The author proposes an optimization technique for the improvement of a single file parallel write performance by replacing Gfarm's N-1 access pattern with the N-N in order to increase performance of parallel writes to a single file. Instead of writing a part of a single file, each process creates a different file in the same directory. In that case, the Gfarm file system preferentially creates each file on the local storage of the node, and each process writes data to the locally stored file. In order to access to the split files transparently, as if they were a single large file, the proposed method stores information about the mapping of the single file into a file called the location information file. Using this

information, it can be determined which part of the single file is accessed by each process and it allows us to translate the file location between the original large single file and split files.

In [11] the performance and scalability problems of MPI-IO with the Lustre file system [12] are addressed. The authors proposed a software library, called Y-lib, which optimizes data access time by redistributing data between multiple storage resources. Nonetheless, data distribution in the studies mentioned above is done without monitoring the current state of the storage resources, thus those approach can be applied only to static environments, e.g. a single computing cluster, rather than dynamic environments such as Grids.

The second aspect, i.e. QoS support for data storage systems, as being critical for many storage systems has been given a lot of attention in the last years. In [13] the author proposes a system capable of providing support for Quality of Service in Grid computing applications, allowing developers to specify end-to-end Grid QoS parameters. The approach relies on reserving computer resources and later usage the reservations for QoS provisioning. The reservations guarantee that the system resource manager will provide the specified Quality of Service level for applications. Although the system is very popular among the Grid community – it is based on the Globus Toolkit – it does not employ some important Grid technologies (e.g. dynamic resources monitoring), nor does it supply functionalities such as an aggregation of similar user's SLA into a one common SLA defined on the Virtual Organization level.

Summarizing, the solutions discussed in this section are specific for a selected storage system or some version of grid middleware software, thus their area of applications is narrowed to either a particular type of storage resources or a specific middleware software. In contrast, the approach presented in this paper is a general one, since it takes into account various heterogeneous storage devices and systems. Moreover, it uses semantic technologies for defining QoS metrics making it flexible and adaptable to different types of storage resources.

4. QStorMan Toolkit Overview

The QStorMan toolkit consists of a number of loosely coupled components as it is presented in Fig. 1. Each of the components is responsible for a different part of the toolkit functionality. As the input, QStorMan accepts a set of non-functional requirements for storage resources. The requirements can be defined on the following levels: VOs, users, and applications. From the grid applications perspective QStorMan responds to the following types of requests:

- finding a worker node for a task submission, with access to storage resources which meet basic requirements of a data-intensive application, e.g. storage capacity,
- selecting a storage resource accessible from a certain worker node where data produced by an application should be placed during its execution, based on non-functional requirements and current workload on the resources.

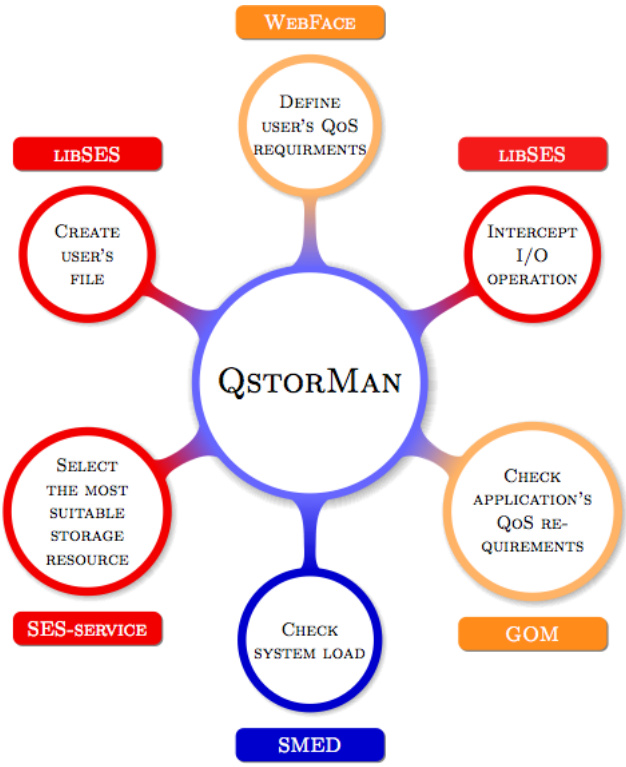

Figure 1. QStorMan components and their responsibilities

In either case, the decision is made based on storage environment semantic descriptions and dynamic information provided by a monitoring system, dedicated to storage resources. As we do not utilize resource allocation or reservation methods, QStorMan works in the best-effort mode, i.e. there is no guarantees that the requested QoS level will be actually met. In the next subsection we briefly sketch the QStorMan toolkit components along with their functionality. A more thorough description of the QStorMan toolkit architecture can be found in [14] and [15]. The use of semantic techniques within QStorMan toolkit can be found in [16].

4.1. System components

The toolkit consists of five independent subsystems each serving its unique function:

- The knowledge base (**GOM**) that stores descriptions of a storage environment along with non-functional requirements defined for users, applications or VOs.
- The Storage Element Selection service (**SES-service**) which finds the most suitable storage resource according to defined requirements and current environment workload. As a result, the service returns a list of storage resource addresses along

with a distance between the storage resource and the given requirements using a defined formula. The smaller the distance is the better the storage element suits the requirements. The set of currently supported set of non-functional requirements includes: current/average read/write transfer rates and free capacity.

- Two programming libraries: **libSES C++ API**, which provide functions for declaring non-functional requirements explicitly in an application's source code to manage a file creation process in a distributed storage environment and **libSES C library**, which automatically intercepts I/O operations addressed to a file system and distribute data to the selected storage resources according to specified non-functional requirements.
- The monitoring system (**SMED**) which continuously monitors storage resources and provides information about current or average values of different storage QoS parameters.
- The portal (**WebFace**) where a user can define non-functional requirements for his/her applications.

4.2. Usage Examples

As one cannot expect that all types of data intensive applications operate in a similar way, QStorMan supports three different usage scenarios, giving the user the flexibility of choice: using libSES C library either with default or defined requirements, or using libSES C++ API.

```
LD_PRELOAD=/opt/libs/libses-wrapper.so ./dataIntensiveApplication
```

Figure 2. Starting a data-intensive application with the libSES C library enabled

The first two options are dedicated for supporting legacy applications, whose source code can not be modified. In this case, non-functional requirements can be declared for the whole application with the QStorMan WebFace, or the user can utilize default requirements declared by his/her home Virtual Organization. In either case, the applications has to be launched with the libSES C library linked as depicted in Fig. 2. During the application's runtime, each system call for file creation is intercepted by the libSES C library to decide to which storage resources the file should be stripped. The main drawback of this method is the usage of the same set of requirements to each created file.

The third option is suitable for new applications, whose non-functional requirements regarding storage may vary during runtime. The requirements are defined in an application's source code as depicted in Fig. 3. In this case, each subsequent file can have a different *StoragePolicy* object assigned with other set of requirements.

```
LustreManager manager;
StoragePolicy policy;

policy.setAverageReadTransferRate(50);
policy.setCapacity(100);

int descriptor = manager.createFile("fileName.dat", &policy);
```

Figure 3. Declaring non-functional requirements for a file with the libSES C++ library

5. Performance Evaluation

To evaluate the effectiveness of the proposed approach, a number of tests were performed, which used a synthetic workload representing multiple users executing data-intensive applications simultaneously using the production infrastructure of the PL-Grid project. As the infrastructure utilizes the Lustre file system [12] to access storage resources, as a default choice for storing data for computation, the current implementation of QStorMan operates on the Lustre file system. During tests, the Lustre installation included 24 FATA disks, divided into 8 pools with equal sizes, with over 170 TB of total capacity. Some of the benchmarked users were using the QStorMan system via the SES programming libraries, while the others were running the same application without QStorMan support. The set of defined non-functional requirements included a current read transfer rate set to 500 MB/s and a current write transfer rate set to 300 MB/s, due to the involvement of both read and write operations. Based on the current workload of storage resources, QStorMan decided where new files were created to provision the closest to the requested QoS level. Hence, in the described test case QStorMan can be considered as a load balancer on top of storage resources. However, it is even more efficient, when different users have different requirements regarding storage.

Performed tests were aimed at assessing the speedup gain by the users executing their applications with QStorMan support in comparison to the users which do not use QStorMan. It goes without saying that all the users executed the same data-intensive application scenario that involved three steps: read the data from the Lustre file system, perform some computation, and write the data to the Lustre file system. These steps were repeated sequentially for each file size. In each case the same number of users were benchmarked, which executed their applications with and without QStorMan support. The users which did not use the QStorMan selected storage resources in a random fashion, which is a default algorithm when using the Lustre file system.

Cases of 6, 8, and 10 simulated users were investigated with files sizes varying from 1GB to 50GB with 1GB step, hence each user performed 50 iteration of the reading-computing-writing loop. Overall test results are presented in Table 1. Depending on the values of those parameters, the average observed reduction in an

application's execution time was in the range of 14% to 46% in favor of users supported by the QStorMan toolkit. Fig. 4 shows an example evaluation with 6 users, three of them used QStorMan. To compare overall runtime of applications supported and not supported by QStorMan, we divided them into pairs. Furthermore, it should also be noted, that the impact of the QStorMan services on the existing grid environment was minimal, i.e. the average SES-service response time was between 0.20 and 0.25 second, while the SMED subsystem has been configured to generate less than 5% overhead to the Lustre file system. QStorMan components were run on an independent server, thus there was no additinal overhead generated. It allows us to conclude that QStorMan can significantly reduce the execution time of data-intensive grid applications.

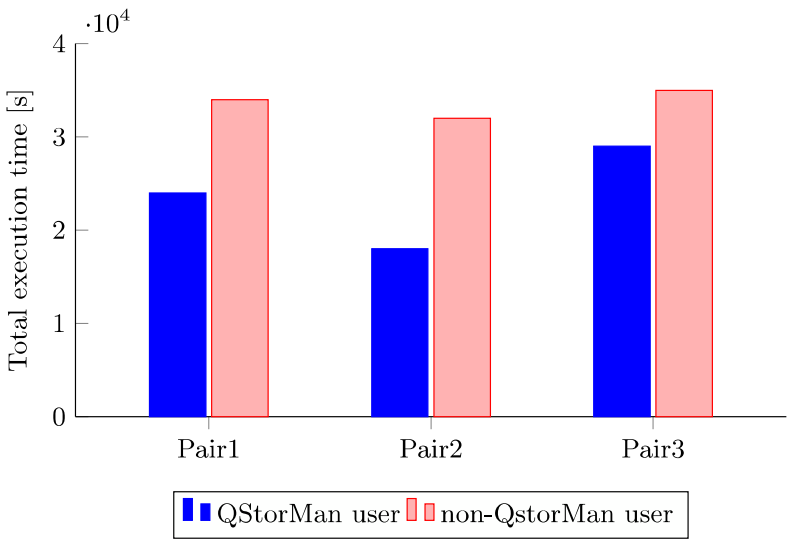

Figure 4. Execution time of application with and without QStorMan support

Table 1
Experimental evaluation of the QStorMan toolkit with a synthetic workload

Number of users	Avg time of a QStorMan supported user	Avg time of standard users	Performance gain [%]
6	3 h 8 min	5 h 54 min	46
8	2 h 21 min	2 h 44 min	14
10	2 h 32 min	3 h 31 min	27

6. Conclusions and Future Work

In this paper, we have described a toolkit for data management based on non-functional requirements of data-intensive applications. The results show that the execution time of data-intensive applications can be decreased by using monitoring data of storage QoS parameters to distribute application data between available storage resources according to specified non-functional requirements. Future work will include further tests of the proposed system using real data-intensive applications in the framework of the PL-Grid project. Furthermore, we plan to extend the area of non-functional requirements definition and, by exploiting the semantic technologies, we intend to give users the opportunity to express their requirements at a higher, more domain-specific level of abstraction. Moreover, we intend to incorporate algorithms introduced in [17] to address read-only data and to enhance the monitoring subsystem by access time estimation mechanisms presented in [18].

Acknowledgements

This research is supported partly by the European Regional Development Fund program no. POIG.02.03.00-00-007/08-00 as part of the PL-Grid Project. We thank Patryk Lason (CYFRONET), Marek Magrys (CYFRONET) and Lukasz Flis (CYFRONET) for the help in preparing the test environment based on the PL-Grid infrastructure.

References

[1] Kryza B., Dutka L., Slota R., Kitowski J.: *Dynamic VO Establishment in Distributed Heterogeneous Business Environment.* [in:] G. Allen, J. Nabrzyski, E. Seidel, G.D. van Albada, J. Dongarra, and P. M. A. Sloot (Eds.), LNCS, vol 5545(2009) pp. 709–718.

[2] *The PL-Grid project website.* [on-line: `www.plgrid.pl`, as of December 16, 2011]

[3] Google Processing 20 Petabytes a Day, [on-line: `www.datacenterknowledge.com/archives/2008/01/09/google-processing-20-petabytes-a-day/`, as of December 16, 2011].

[4] German Climate Computing Center (DKRZ) website, [on-line: `http://www.dkrz.de/daten-en`, as of December 16, 2011].

[5] Deelman E., Chervenak A.: *Data Management Challenges of Data-Intensive Scientific Workflows.* Proc. of the 8th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2008, pp. 687–692.

[6] Ekanayake J., Li H., Zhang B., Gunarathne T., Bae S., Qiu J., Fox G.: *Twister: a runtime for iterative MapReduce.* Proc. of the International Symposium on High Performance Distributed Computing, 2010, pp. 810–818.

[7] Raicu I., Foster I., Zhao Y., Little P., Moretti C., Chaudhary A., Thain D.: *The quest for scalable support of data-intensive workloads in distributed systems.* Proc. of the International Symposium on High Performance Distributed Computing, 2009, pp. 207–216.

[8] Paszyński M., Pardo D., Torres-Verdín C., Demkowicz L., Calo A.: *A parallel direct solver for the self-adaptive hp Finite Element Method*. Journal of Parallel and Distributed Computing, vol. 70(3) (2010) pp. 270–281.

[9] Cettei M., Ligon W., Ross R.: *Support for parallel out of core applications on Beowulf workstations*. Proc. of Aerospace Conference, vol. 4(1998), pp. 355–365.

[10] Kimura H., Tatebe O.: *MPI-IO/Gfarm: An Optimized Implementation of MPI-IO for the Gfarm File System*. Cluster, Cloud and Grid Computing (CCGrid), May 2011, doi: 10.1109/CCGrid.2011.82, pp. 610–611.

[11] Dickens P., Logan J.: *Y-Lib: A User Level Library to Increase the Performance of MPI-IO in a Lustre File System Environment*. Proc. of the 18th ACM international symposium on High performance distributed computing, 2009, Garching, Germany, ACM, pp. 31–38.

[12] *The Lustre file system website*: [on-line: `wiki.lustre.org`, as of June 20, 2011].

[13] Roy A. J.: *End-to-end quality of service for high-end applications*. PhD thesis. Adviser – Ian Foster, 2001.

[14] Krol D., Kryza B., Skalkowski K., Nikolow D., Slota R., Kitowski J.: *QoS Provisioning for Data-Oriented Applications in PL-GRID*. [in:] M. Bubak, M. Turala, K. Wiatr (Eds.), Proc. of Cracow Grid Workshop - CGW'10, October 2010, ACC Cyfronet AGH, 2011, Krakow, pp. 142–150.

[15] Slota R., Krol D., Skalkowski K., Kryza B., Nikolow D., Kitowski J.: *FiVO /QStorMan: toolkit for supporting data-oriented applications in PL-Grid*. Proc. of KU KDM 2011, March, 2011, Zakopane, ACC Cyfronet AGH, ISBN 978-83-61-433-04-0, pp. 68.

[16] Slota R., Krol D., Funika W., Kryza B., Nikolow D., Kitowski J.: *FiVO/QStorMan Semantic Toolkit for Supporting Data-Intensive Applications in Distributed Environments*. Computing and Informatics, 2011, in press.

[17] Slota R., Skital L., Nikolow D., Kitowski J.: *Algorithms for Automatic Data Replication in Grid Environment*. 6th international conference, PPAM 2005. Poznań, Poland, September 11–14, 2005, eds. Roman Wyrzykowski *et al.*, Berlin, Heidelberg, Springer-Verlag, 2006, LNCS, vol. 3911, pp. 707–714.

[18] Nikolow D., Slota R., Dziewierz M., Kitowski J.: *Access time estimation for tertiary storage systems*, Euro-Par 2002, Paderborn, Germany, August 27–30, 2002, Proc., eds. Burkhard Monien, Rainer Feldmann, Berlin, Springer, 2002, LNCS, vol. 2400, pp. 873–880.

Affiliations

Renata Słota
AGH University of Science and Technology, Faculty of Electrical Engineering, Automatics, Computer Science and Electronics, Department of Computer Science, Krakow, Poland, AGH University of Science and Technology, ACC Cyfronet AGH, Krakow, Poland, `rena@agh.edu.pl`

Dariusz Król
AGH University of Science and Technology, Faculty of Electrical Engineering, Automatics, Computer Science and Electronics, Department of Computer Science, Krakow, Poland, dkrol@agh.edu.pl

Kornel Skałkowski
AGH University of Science and Technology, Faculty of Electrical Engineering, Automatics, Computer Science and Electronics, Department of Computer Science, Krakow, Poland, AGH University of Science and Technology, ACC Cyfronet AGH, Krakow, Poland

Michał Orzechowski
AGH University of Science and Technology, Faculty of Electrical Engineering, Automatics, Computer Science and Electronics, Department of Computer Science, Krakow, Poland

Darin Nikolow
AGH University of Science and Technology, Faculty of Electrical Engineering, Automatics, Computer Science and Electronics, Department of Computer Science, Krakow, Poland, AGH University of Science and Technology, ACC Cyfronet AGH, Krakow, Poland

Bartosz Kryza
AGH University of Science and Technology, Faculty of Electrical Engineering, Automatics, Computer Science and Electronics, Department of Computer Science, Krakow, Poland

Michał Wrzeszcz
AGH University of Science and Technology, Faculty of Electrical Engineering, Automatics, Computer Science and Electronics, Department of Computer Science, Krakow, Poland

Jacek Kitowski
AGH University of Science and Technology, Faculty of Electrical Engineering, Automatics, Computer Science and Electronics, Department of Computer Science, Krakow, Poland, AGH University of Science and Technology, ACC Cyfronet AGH, Krakow, Poland

Received: 28.12.2011
Revised: 29.01.2012
Accepted: 30.01.2012