Model of QoS Management in a Distributed Data Sharing and Archiving System

Darin Nikolow\textsuperscript{a,}\textsuperscript{*}, Renata Słota\textsuperscript{a,}\textsuperscript{b}, Stanisław Polak\textsuperscript{a}, Danilo Mitera\textsuperscript{b}, Marek Pogoda\textsuperscript{b}, Paweł Winiarczyk\textsuperscript{b}, Jacek Kitowski\textsuperscript{a,}\textsuperscript{b}

\textsuperscript{a}AGH University of Science and Technology, Faculty of Computer Science, Electronics and Telecommunications, Department of Computer Science, Kraków, Poland
\textsuperscript{b}AGH University of Science and Technology, ACC Cyfronet AGH, Kraków, Poland

Abstract

The scientific applications of the fourth paradigm of science deal with large amounts of data stored in various storage devices or systems. Distributed storage systems are often chosen for storing these data. Some of the requirements posed to those storage systems may concern Quality of Service (QoS) aspects formally expressed in a Service Level Agreement. The QoS management in distributed storage systems is a challenging task given the possible storage device heterogeneity, the dynamically changing data access patterns, the client’s concurrency and storage resource sharing. The problem becomes even more complicated when distributed computing environments with virtualized and shared resources like Clouds are considered. In this paper we present our research concerning the methods of storage performance management with respect of QoS in distributed environments which has been done within the National Data Storage 2 (NDS2) project. A new model of storage QoS management is proposed and its implementation within NDS2 is described.

Keywords: Distributed storage systems, Service level agreement, Quality of service,

1. Introduction

The scientific applications of the fourth paradigm [1] of science deal with large amounts of data, which can be stored in various storage devices or systems, taking into account the possibility of sharing, publishing and archiving. Some of the requirements posed to those storage systems may concern Quality of Service (QoS) aspects [2]. Examples of such requirements are: specifying the minimal data transfer rate, the maximal data access latency, the minimal level of data protection and availability. These requirements are formally expressed in a Service Level Agreement (SLA) being part of the contract agreed between the client and the service provider.

It happens increasingly often that legal requirements enforce companies to store certain type of data for a given time period. Examples come from fields like telecommunication and medicine. On the other hand more often scientific research centers use distributed archiving systems to deal with huge data volumes which allows them to do their research on a larger scale, e.g., earth science, biology, astronomy. Typical clients of archiving systems are museums, libraries, state administration. It should also be mentioned that users more often require certain level of QoS concerning the security of data and long-term data availability.

\textsuperscript{*}Corresponding author. Tel.: +48-12-328-33-36.
E-mail address: darin@agh.edu.pl.
Some of the modern storage systems are distributed storage systems, where the load is distributed among some number of Storage Nodes (SNs), organized in a common filesystem namespace, e.g., Lustre, GlusterFS, dCache, iRODS [3, 4, 5, 6]. The QoS management in such storage systems is a challenging task given the possible storage device heterogeneity, the dynamically changing data access patterns, the client’s concurrency and storage resource sharing. The problem becomes even more complicated when distributed computing environments with virtualized and shared resources like Clouds [7] are considered [8].

In this paper we present our research concerning the methods of storage performance management with respect of QoS in distributed environments which has been done within the National Data Storage 2 (NDS2) project [9]. The NDS2 project concerns the building of data storage and archiving system spread over the main academic computer centers in Poland. The stress of NDS2 is put on the high level of data availability, data protection and data security.

The rest of the paper is organized as follows. Section 2 presents background and related works. Section 3 describes the NDS2 project. The model of the storage QoS management implemented in the NDS2 distributed storage system is presented in Section 4. Implementation details of QMS are described in Section 5. Selected experimental results are shown in Section 6. Section 7 concludes the paper.

2. Background and related works

The problem of storage management with QoS support is addressed in many research studies in the recent two decades. The problem has been recognized with the advent of digital multimedia, and especially videos, which impose QoS performance requirements to the storage servers and network’s infrastructure. Distributed storage systems have constantly been gaining attention as the Grid and Cloud computational environments were adopted in various projects. Providing storage related SLA and QoS support for such infrastructures is still considered challenging. Selected studies related to distributed storage systems and storage management for QoS support are presented below.

There are many projects focused on providing software for building and management of distributed storage resources. Some of them are rather designed for local range cluster installations like Lustre [3] and GlusterFS [4], while other are designed with global range in mind where the storage resources are geographically separated. Examples of the latter systems are dCache [5] and iRODS [6]. iRODS is different from others since it provides a rule engine and a set of micro services allowing for creation of user-based storage workflows.

In [10] the idea of distributed network storage service with QoS guarantees is presented. The authors propose a relevant architecture and describe its key components and the related research problems.

Performance management is an essential issue raised in studies which are focused on storage QoS and SLA subjects. Storage resource reservation is one method which allows for strict performance guarantees provided a full control over the entire system is applicable. When distributed storage systems are under consideration two elements of the system are most important: storage hardware, for example disk drives, and network infrastructure. A drawback of resource reservation is that over-provisioning often takes place which leads to inefficient use of resources. One of the studies concerning this subject is [11] in which a prototype of a distributed storage system allowing for user side I/O performance reservation is described.

The problem of automatic storage system configuration, which is supposed to meet the required performance, is addressed in [12]. The described tool, namely Hippodrome, automatically chooses the appropriate storage devices and determine their configuration based on workload analyzes and performance prediction. Similar approach is proposed in [13].

Another method for storage QoS and performance management is based on statistical approach. In this approach the system constantly monitors and predicts performance based on modeling and statistics, which allows for better use of storage resources but introduces some uncertainty about the delivered performance compared to the previous method. Black box modeling for optimizing data access in virtualized data centers is presented in [14]. In [15] data allocation policies for storage QoS are proposed which allowed the authors to build a filesystem with storage QoS support based on user hints.

Our approach is based on the later method combined with a novel method for SLA policy management for differentiating user’s storage access based on their belonging to a given SLA group.
3. National Data Storage 2

NDS [16] is a Polish initiative aimed at building a geographically distributed storage system for backup and archiving. The system has been deployed nationwide with the participation of main computer centers and universities in Poland, namely, Academic Computing Center CYFRONET AGH in Cracow, Academic Computing Center in Gdansk, Częstochowa University of Technology, Marie Curie-Sklodowska University in Lublin, Poznań Supercomputing and Networking Center, Technical University of Białystok, Technical University of Łódź, Wrocław Supercomputing and Networking Center. The NDS software is used in the PLATON [17] platform which provides various computational related services for the scientific community in Poland. NDS is also being integrated with another PL-Grid infrastructure [18] as a backend storage for large scale scientific computations.

NDS2 [9] is the successor project of NDS aimed to extend the functionality of NDS with special stress put on security of data. The NDS2 project will, at its final version, implement the following functionalities:

- secure sharing of data between the NDS2 users,
- secure publishing of data for outside users via a so called sandbox mechanism,
- versioning of data and versions management,
- designing and building of hardware assisted ciphering appliance for secure data exchanging between the end points and NDS2,
- automatic and safe data storage with end-to-end encryption and data integrity control,
- hierarchical and distributed users management,
- support for QoS and SLA,
- advanced accounting.

The layered structure of NDS2 is shown in Fig.1. The access layer contains components responsible for interfacing the system with users. Users may use various methods and protocols to access their data, namely, via SFTP, SSHFS, NFS or WebDAV. The management layer contains components responsible for management of various aspects related to storage systems, for example, storage resource management, namespace and file management, users management. The most essential elements in this layer are shown in the figure. The Data Daemon (DD) is responsible for providing data streams to the upper layer using a FUSE-based [19] solution. Files in NDS2 are replicated, so the DD can choose a replica for the given read transfer or SN for the given write transfer. The Meta Catalog (MC) manages the meta-data about the stored files. The Replication Daemon (RD) controls the process of creating, migrating and removing replicas. The User Management System (UMS) manages
users related assets like permissions, groups, etc. The Life Daemon (LD) is a monitoring system based on Nagios software [20]. At the lowest level SNs are located for providing storage resources. SN is a server, which provides access to an HSM system or to a disk array storage.

4. Model of QoS Management for NDS2

The QoS Management System (QMS) (see Fig.1) is a core component which provides support for QoS and SLA. QMS conception, model of QoS focused on storage performance, implementation and preliminary test results are presented in the next sections.

The following QoS functionalities are addressed in NDS2 system:
• users can choose a minimal data transfer rate, which is satisfactory to them,
• users can choose a data protection level,
• users can choose a data availability level.

The NDS2 system meets the user’s requirements by selecting the appropriate storage resources, applying replication policy and choosing optimal values for various NDS2 configuration parameters based on continuously monitoring of relevant QoS metrics. More details about the conception and the possible methods of storage QoS management may be found in [21], which presents our previous research on the subject.

The QoS management model of NDS2 is shown in Fig.2. When a data access request comes the system reads the current values of QoS metrics and based on them and on the SLA policies an SN selection is done. After that, the data is transferred and later, at the appropriate moment, a check for SLA violations is run. If a violation is detected an attempt to find better policy is made. In order to implement this model the following elements of QMS are designed:
• QoS Monitor, which monitors QoS metrics,
• SLA Monitor, which monitors the current SLA metrics for users and reports SLA violations,
• QMD (Quality Management Daemon), which is responsible for taking actions aimed at keeping the SLA fulfilled,
• Presentation Layer, which is used to present SLA reports and set various SLA related input parameters.

QoS Monitor is based on passive monitoring\(^1\) of the QoS related low level parameters, which is done by the LD. Passive monitoring introduces little CPU and network overhead and does not use storage resources. The

---
\(^1\)In the context of this paper passive monitoring means that the monitoring does not directly use some kind of active benchmarking; it uses the information available from logs and statistical counters provided by the operating system itself or by other modules.
parameters are obtained from the SNs and used for calculating the QoS metrics (see Table 1), which are kept in a database. The SLA Monitor scans the logs file to calculate the current values of SLA parameters (see Table 2) and checks if the SLA is not violated. If a violation is discovered the monitor sends an alert to the QMD, which takes actions to resolving problems with SLA.

An users of NDS2 is assigned to an SLA group which specifies acceptable values of SLA parameters. These values, called further SLA limits, can be maximal or minimal values depending on the nature of parameter. For example, for the ReadTransferRate SLA parameter it would be the minimal acceptable value, while for the Read-LatencyTime it would be maximal acceptable value. In order to have some kind of control over the selection of SN, depending on the given SLA group, storage management policies for SLA, called further SLA policies, are introduced. Generally, the SLA policies define different preferences for selecting SN with respect to SLA. In our case SLA policy is defined as a set of coefficients which are used to calculate the preference of an SN for the given transfer and the given SLA group (see formula 1 and formula 2).

\[
SNReadPreference = k_1 \times SNMaxReadTransferRate \times filesize - \\
( k_2 \times SNCurrentReadTransferRate \times filesize - \\
( k_3 \times \frac{filesize}{SNTapeReadRate} + k_4 \times SNTapeLatency) \times \\
k_5 \times SNisFileCached - \\
k_6 \times SNLoad - \\
k_7 \times SNIOps 
\]

(1)

\[
SNWritePreference = k_1 \times SNMaxWriteTransferRate \times filesize - \\
( k_2 \times SNCurrentWriteTransferRate \times filesize - \\
k_5 \times SNisFileCached - \\
k_6 \times SNLoad - \\
k_7 \times SNIOps 
\]

(2)

One method of storage QoS management is via changing the coefficients of SLA policies. The SN preference is calculated by DD before selecting the SN for the given data transfer. The coefficients \((k_1, \ldots, k_7)\) can be changed manually by the administrator or automatically by the QMD. The SNs with higher values of \(SNReadPreference\) or \(SNWritePreference\) are preferred. The values of the coefficients can be positive or negative numbers including zero. In this way it is possible to invert the influence of a given QoS metric. For example, setting a negative value for \(k_1\) causes that SNs with lower maximal read transfer rate are preferred. The SLA limits for the given class of users are set via the web interface provided by the presentation layer and they are managed by the UMS. The QoS metrics used by DD to calculate the preference of SNs (see Eqs. 1,2) are listed in Table 1.

<table>
<thead>
<tr>
<th>QoS metric name</th>
<th>Unit</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNCurrentReadTransferRate</td>
<td>MB/s</td>
<td>Current load of the SN for read transfers. SNs with lower values are preferred.</td>
</tr>
<tr>
<td>SNMaxReadTransferRate</td>
<td>MB/s</td>
<td>Maximal SN throughput. SNs with higher values are preferred.</td>
</tr>
<tr>
<td>SNTapeReadRate</td>
<td>MB/s</td>
<td>Read transfer rate for tape drive. SNs with higher values are preferred.</td>
</tr>
<tr>
<td>SNTapeLatency</td>
<td>s</td>
<td>The average latency of tape drives. The latency includes the time necessary to load a tape and position the head. SNs with lower values are preferred.</td>
</tr>
<tr>
<td>SNisFileCached</td>
<td>n.a.</td>
<td>This parameter tells if the given file resides in cache. 0 means that the file is in cache or that there is no HSM system available on this SN. 1 means that the file resides on tape.</td>
</tr>
<tr>
<td>SNLoad</td>
<td>%</td>
<td>SN load. Shows how much of the potential storage bandwidth is currently used. SNs with lower values are preferred.</td>
</tr>
<tr>
<td>SNIOps</td>
<td>IO/s</td>
<td>Number of I/O operations per second. SNs with lower values are preferred.</td>
</tr>
</tbody>
</table>

Other methods of storage QoS management are based on replication and pre-staging. The transfer rate can be improved by creating more replicas of popular files. Pre-staging can be used to selectively stage files of users, for which SLA violations have been encountered or for which such violations are highly probable [21].

4.1. SLA parameters

The SLA parameters defined for our approach [21] are divided into three classes:
• performance parameters,
• data protection parameters, which determine probability of loss of the data or unauthorized access,
• availability parameters, which determine reliability of the NDS2 system.

The SLA parameters proposed for NDS2 system are listed in Table 2. In this paper we focus on issues with performance related SLA parameters.

Table 2. SLA parameters.

<table>
<thead>
<tr>
<th>SLA parameter name</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>performance parameters</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReadTransferRate</td>
<td>MB/s</td>
<td>Read transfer rate</td>
</tr>
<tr>
<td>WriteTransferRate</td>
<td>MB/s</td>
<td>Write transfer rate</td>
</tr>
<tr>
<td>ReadLatencyTime</td>
<td>s</td>
<td>Latency for reads</td>
</tr>
<tr>
<td>WriteLatencyTime</td>
<td>s</td>
<td>Latency for writes</td>
</tr>
<tr>
<td><strong>data protection parameters</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DataProtectionLevel</td>
<td>n/a</td>
<td>Level of data protection from data loss</td>
</tr>
<tr>
<td>DataSecurityLevel</td>
<td>n/a</td>
<td>Level of data protection from unauthorized access</td>
</tr>
<tr>
<td><strong>availability parameters</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DataAvailabilityLevel</td>
<td>n/a</td>
<td>Reliability of the data access</td>
</tr>
<tr>
<td>ServiceAvailabilityLevel</td>
<td>n/a</td>
<td>Reliability of the NDS2 services</td>
</tr>
</tbody>
</table>

5. Implementation of QMS

The whole QMS is implemented in the Python programming language using Django framework [22] and is based on the Model-View-Controller (MVC) design pattern. Currently, three modules of QMS are implemented: QoS Monitor, SLA Monitor and Presentation Layer. Each module consists of: a model class mapped by Object Relational Model (ORM) to relational database, a view class for showing data and a controller class which takes and processes data.

5.1. Architecture

The QMS architecture is presented in Fig.3. The NDS2 modules with which QMS communicates are also shown. All data are stored in one relational database (QMS database) accessed by Django ORM. The exchange of information between the modules is done via the QMS database or via the modules interfaces. Nagios [20] is used as a general monitoring system of NDS2. Nagios uses plugins installed on the monitored nodes to obtain the necessary parameters, which are placed in the returned status string of the plugins. The LD reads the raw plugins status and stores it in the database.

The proposed architecture of QMS is designed to integrate well into the NDS2 system with minimal modifications of the existing modules. The architecture implements the storage QoS management model (see Sec.2) allowing to exploit the novel idea of using SLA policies for heuristic selection of SNs.

5.2. Monitoring

There are two main monitoring modules. The first one is the QoS Monitor which monitors storage nodes by reading low level performance related parameters from the operating system usually via the /proc filesystem and processes them into QoS metrics which are next saved to its own table in the QMS database. The second one is the SLA Monitor which monitors for SLA violations. The SLA Monitor checks the performance delivered to users by periodically scanning the logs and comparing them with the SLA limits according to the SLA group assignment. If a violation is detected it alerts the admin and eventually re-calculate (using some sort of adaptive optimization based on the current performance profile) the SLA policy coefficients in order to improve the delivered performance for that user.

SN monitoring is done by the Nagios compatible plugins. LD reads the plugins output and writes them to their database. Next QoS monitoring reads those values, process them for its own use and write the resulting values...
to the QMS database. QoS monitoring module is responsible for processing the raw output from the plugins and obtaining of usable number values.

Since each open, read, write and close file operation is recorded along with its corresponding duration time in the file "history.csv" (see Fig.3), the SLA monitor processes this file to check if the SLA for the given user is not violated. The limit values for SLA parameters are determined by the SLA group which is assigned to each user.

6. Preliminary test results

The preliminary test results showing the accuracy of SLA monitoring in terms of experienced performance by the client application are presented in this section.

6.1. Test environment

The test environment consists of one Access Node (AN), on which the management and access layers are installed, and two machines with external storage volumes attached to them: SN1-kmd and SN2-kmd.

SN1-kmd is a Xen virtual machine located at a BL460c G6 blade host with Intel Xeon E5540 2.53GHz processors. The machine has a quad core processor and 2GB RAM. The data is physically stored on HP EVA 8000 Disk Array on which 2 LUNs with capacity of 1 TB each is made available exclusively for this machine and integrated additionally into a single volume by the IBM’s GPFS software.

SN2-kmd is a VMware virtual machine located at an HP BL460c G7 Server Blade with two Intel Xeon E5649 2.53GHz processors. The virtual machine is configured with two vCPUs (2 core) and 2 GB RAM. The data is stored on the server’s local disks, which are also shared by other virtual machines.
6.2. SLA alerts

The aforementioned SLA Monitor should display alert if the value of the relevant write or read transfer rate is less than the limit associated with the given user. Three SLA Groups were defined. The limits for these SLA groups were set to 20 MB/s (Limit1), 40 MB/s (Limit2) or 60 MB/s (Limit3), respectively. The test of the SLA Monitor were conducted using the sftp command. At the beginning 10 files have been created, each with a size of 2 GB. Next, each of them was sent to the AN using the following sftp commands:

1. `date +%s%n
2. put fileName
3. date +%s%n

The commands in line 1 and 3 display the current time expressed in the number of nanoseconds since 1970-01-01 00:00:00 UTC referred further as $T_{before}$ and $T_{after}$ respectively. The client side write transfer rate in MB/s is calculated using Eq.3 and the results are logged.

$$\text{transfer rate} = \frac{2048}{(T_{after} - T_{before})/10^9}$$  (3)

In the next step, the performance data from the SLA Monitor for these transfers was extracted. The accuracy of SLA monitor is shown in Fig. 4. We can see that the SLA Monitor monitors the user’s transfers well with little almost unnoticeable errors. In all cases the SLA Monitor correctly signaled violation of the SLA limit of 60 MB/s.

![Graph showing accuracy of SLA Monitor for write transfers.](image)

In a subsequent experiment these 10 files were transferred back to the local host using the following sftp commands:

1. `date +%s%n
2. get fileName
3. date +%s%n

Again, the read transfer rate was calculated, according to the formula 3, the SLA monitor activity was observed and the data recorded. The results are presented in Fig. 5. In this case the SLA Monitor was also correctly detecting violations and the accuracy was similar to the one for write transfers.

The accuracy of our method is acceptable proving that by monitoring only on the server side it is possible to properly detect SLA violations for data transfers.
7. Conclusion and future work

In this paper the storage QoS management methods designed for the NDS2 distributed storage system has been presented. As part of the research a new model of storage QoS management has been elaborated and implemented within NDS2. The proposed approach for storage QoS management is based on a statistical approach combined with a novel method for SLA policy management using heuristics for differentiating user’s storage access based on their belonging to a given SLA group.

The implemented QMS allows for monitoring of various QoS metrics and SLA parameters which have been defined as part of the project. The conducted tests showed that the chosen method of monitoring the user level SLA performance related parameters, which is run internally on the NDS2 servers, produces quite accurate data for clients connected via Internet. In this way there is no need to install additional monitoring software on the client side.

Our future work will concentrate on implementing and testing methods for automatic setting of SLA policies to keep the SLA fulfilled.

Acknowledgments

This work is supported by Polish MNiSW grant nr N R02 0025 10. AGH-UST grant no. 11.11.230.015 is also acknowledged. Thanks go to Maciej Twardy and Karol Krawętek for their support in using the storage hardware of ACC Cyfronet AGH for the presented tests.

References