Computer Science • 19(4) 2018 https://doi.org/10.7494/csci.2018.19.4.2878

Darin Nikolow
Renata Słota
Stanisław Polak
Marek Pogoda
Jacek Kitowski

POLICY-BASED SLA STORAGE MANAGEMENT MODEL FOR DISTRIBUTED DATA STORAGE SERVICES

Abstract *There is high demand for storage related services supporting scientists in their research activities. Those services are expected to provide not only capacity but also features allowing for more flexible and cost efficient usage. Such features include easy multiplatform data access, long term data retention, support for performance and cost differentiating of SLA restricted data access. The paper presents a policy-based SLA storage management model for distributed data storage services. The model allows for automated management of distributed data aimed at QoS provisioning with no strict resource reservation. The problem of providing users with the required QoS requirements is complex, and therefore the model implements heuristic approach for solving it. The corresponding system architecture, metrics and methods for SLA focused storage management are developed and tested in a real, nationwide environment.*

Keywords distributed storage systems, quality of service, service level agreement, management model, heuristics

Citation Computer Science 19(4) 2018: 403–429

1. Introduction

To address frontiers of research modern science needs support from computational infrastructures, so many European and nationwide initiatives are dealing with distributed, grid and cloud infrastructures. The notable examples are Helix-Nebula project [15], European Grid Infrastructure (EGI) [11], European Open Science Cloud (EOSC-hub) [9] or Extreme Data Cloud (XDC) [56]. Due to high demand from scientific applications storage related services are highly requested for dealing with huge amount of data and their starvation for storage resources. Those services are expected to provide both, performance and features allowing for more flexible and cost efficient usage of such services. Easy multiplatform data access, long term data retention, support for performance and cost of data access are elements to differentiate on the basis of service level agreements.

In order to address the needs of scientific community concerning the infrastructural support a couple of national initiatives have been also started in Poland. Results from the family of the PL-Grid projects provide the computational infrastructure to run large scale simulations and calculations on high performance computing clusters [32], supported with domain oriented services, solutions and environments [10, 33, 34]. The Pionier infrastructure [31] provides high bandwidth optical networks connecting main computer centers hence used in the PL-Grid infrastructure. Since the scientific related data produced by simulations, sensors or instruments and used by scientific applications need to be stored for future research relevant storage services are needed for the users. Some of the requirements expected by the users concern storage Quality of Service (QoS) and Service Level Agreement (SLA) aspects.

One of the early projects supporting the activities mentioned above was the nationwide NDS2 project [26] resulting in developing a national geographically distributed storage solution for secure accessing, sharing and archiving of data [5]. The particular functionalities of NDS2 were assisted with automated management of storage services supporting the SLA as one of the features. The NDS2 project was focused on alleviating some problems of allocation of resources and their efficient utilization as well as on elastic management. Such topics are currently still of interest in the field of distributed, grid and cloud computing, being subject of scientific activities in the frame of so-called Software Defined Storage described in a SNIA's white paper [6].

The goal of the paper is to describe a policy-based SLA storage management model, as resulting from the NDS2 project, for automated management of data for QoS provisioning and developing a method for SLA storage management with no strict resource reservation. The corresponding system architecture, metrics and methods for the SLA focused storage management have been developed and tested in a real, nationwide environment.

The rest of the paper is organized as follows. The state of the art of distributed storage systems, storage QoS provisioning, and SLA management is given in section 2. Section 3 provides details about the proposed policy-based SLA storage management model in which the heuristic approach is applied. Details of the NDS2 project concept and the implementation of the model are outlined in Section 4. The fifth section presents the results of testing of our approach at the test deployment of NDS2. The last section concludes the paper.

2. State of the art

The design and implementation of distributed storage services with SLA/QoS support for optimized automatic resource management is a complex task in many aspects, hence the selected problems of distributed storage systems, storage QoS provisioning and SLA management are gathered and discussed below.

This section includes analysis of various types of the solutions:

- distributed storage systems,
- storage QoS provisioning,
- SLA management.

More discussion of the existing solutions can be found in [55].

2.1. Distributed storage systems

iRODS [19, 57], as a successor of Storage Resource Broker, SRB [3], is an integrated rule oriented software solution of data storage widely used by the scientific community. iRODS is a flexible solution due to the micro-services, which can be used to program the behavior of the storage according to the users needs.

Onedata [29,54], as a successor of VeilFS [40], is a software storage solution aimed at building globally distributed data storage system to integrate storage resources from different providers, which may not trust each other. The user can have his data kept in storage spaces provided by different storage providers and the data is available transparently – the user can see all his distributed storage resources as a single namespace with transparent access to data. Onedata has a plugin mechanism for event processing allowing the system to take actions according to preconfigured rules. This mechanism could be used for storage allocation taking into account QoS. The system represents a solution which removes or minimizing barriers arising between the user and its data in a distributed storage environment with many storage resource providers, taking into account security, sharing, protection and scalability aspects of data access.

Scality RING [35] is a commercial distributed object storage solution based on the peer-to-peer model. It uses patented modification of the Chord algorithm [45] originally developed at MIT for locating storage nodes and data. The system is scalable, with no centralized components. Ceph [51] is another object storage system mainly used as a storage for clouds or clusters. It provides also block and file based storage through its modules RBD and CephFS. Similarly to the other distributed storage systems, in the Ceph storage system the metadata is decoupled from data for better scalability. Data is distributed among the nodes by using the CRUSH [52] algorithm. GlusterFS [12], as one more cluster filesystem, employs a hashing algorithm for node allocation. Workflow scheduling is one of the hot topics currently, see for example [8]. Multi-tenant distributed or cloud storage problems are also well represented (e.g. [36, 37])

The presented distributed storage systems do not directly support storage QoS in general, although it could be delivered with some additional effort. For the systems focused on flexibility with built-in storage controlling mechanisms (like Onedata and iRODS) it could be done by developing appropriate plugins or microservices. For the systems designed for distributed performance and scalability (like Scality RING, Ceph and GlusterFS) it could be done by building an additional software layer on top of such distributed storage systems.

2.2. Storage QoS provisioning

The problem of delivering data for applications with QoS requirements is addressed in some storage systems. Generally, those systems can be divided into two main classes: the systems which can guarantee QoS and the systems which provide best effort support for meeting the QoS requirements (via optimizing the storage resource management). The first class is based on strict resource/bandwidth reservation combined with some sort of admission control to deny requests for which no more resources can be allocated at the given moment. The second class is based on storage resource selection and replica management. The selected studies related to storage QoS are presented below.

Chuang, *et al.* [7] propose a framework of distributed network storage service with QoS guarantees. The paper describes the main components and a key mechanism behind the presented idea and identifies essential research areas and challenges like real-time storage management, storage resource reservation and admission control. In [46] a distributed storage system solution allowing for explicit reservation of storage resource initiated a priori by the users application is proposed. If accepted, the reservation is time-based and the requested performance is guaranteed during the reservation time interval. A working prototype is described realizing the reservation performance requirements by prioritization of I/O requests coming from applications having ongoing reservation. PARDA [14] is a data transfer control solution located at the cloud storage virtualization layer to provide fair data access of the virtual machines to a shared storage system. In [16, 17] a QoS-oriented capacity provisioning mechanism is proposed. It make use of the queueing theory and selects a suitable queueing model representing the pattern of current workloads. The model is used to forecast the demand for resources and to offer such capacity which is adequate to the required resource capacity.

Lumb et. al. [25] proposed a virtualized storage solution with performance guarantees which uses a feedback-based control of intercepted IO requests for implementing QoS for a given workload. The solution is designed for use in a data center where the clients (hosts) with different performance requirements access a block-level shared set of storage hardware over a storage area network. A similar solution using the adaptive distributed storage controller is presented in [21]. In [50] a distributed file system with storage QoS provision is presented. The system addresses data accessing from the applications running in a Cloud environment. Negotiations of the storage bandwidth are done by using the bidding-based negotiation model called ECNP (Extended Contract Net Protocol). Storage@desk [18] is a system which uses the storage space available on the given institutions desktop computers to provide block storage via iSCSI. A method for automated performance control employing a market-based model for resource management and feedback controller is used to cope with the QoS issues. Slota, *et al.* [42] propose an approach for QoS-aware data access management in the grid environment. The approach is based on a storage system model and a relevant ontology for presenting the performance state of storage systems and making storage management decisions. In [41] a semantic based system for QoS provisioning of distributed data called FiVO/QStorMan is presented. For the current file access operation a storage node is selected dynamically based on the QoS monitoring with ontology enrichment.

Replication is a common technique used to improve the storage performance in distributed systems [43]. In [24] two algorithms for QoS-aware data replication for the cloud environments are proposed and evaluated. Another algorithm, which takes into account also the content importance, is presented in [2]. Shue, *et al.* [36] propose a solution for performance isolation of cloud tenants accessing a key-value store. The solution uses combination of replication, partitioning and fair queueing techniques. Uttamchandani, *et al.* [48] present approach for arbitrating storage resources among competitive clients using the same storage systems. Voulodimos, *et al.* [49] identify management models describing resources, services and requirements for cloud storage environment and propose a unified management model integrating the defined models with SLA schemas, which makes resource allocation in the context of storage QoS easier. A storage QoS model – RSMM (Resource Storage Management Model) targeted at QoS provisioning for data storage service based on HSM (Hierarchical Storage Management) systems is presented in [20]. Replication is also used in the Onedata system for managing consistency for global scalable data access [53]. The further problems, like hierarchical QoS for packet scheduling for different workloads in a hypervisor or dealing with multi-tiered storage systems can be found [4, 8].

Although some storage QoS systems exist they use reservations which can cause overprovisioning resulting in inefficient resource utilization. Therefore, there is still a need for more research on efficient usage of distributed and heterogeneous resources with respect of QoS to full utilization of the available storage performance alleviating overprovisioning problems.

2.3. SLA management

The SLA (Service Level Agreement) is a contractually bind agreement between a service provider and a client. It concerns the quality level of the contracted service.

Typically, each SLA contains Service Level Objectives (SLO), which numerically specify ranges of values of QoS metrics and conditions under which the values are met. An SLO example might sound like this: "The service response time should be below 1 s for 95% of the requests." An additional element in the SLA is the penalty for its violation.

SLA managements frameworks and languages have been developed to support the process of SLA negotiation [1, 22, 23, 38]. The Web Services Agreement Specification [1], proposed by the Open Grid Forum is a standard, which can be used for the SLA definition of Web services using XML documents. It provides a schema for specifying the general structure of the document describing an agreement as well as a protocol for SLA creation. The WSLA framework [22], developed by IBM, is intended for specifying and monitoring of Web services using XML for expressing the SLA. The proposed architecture allows for monitoring of QoS metrics and reporting violations to the client and the service provider. Support for complex QoS metrics based on the existing ones is included too. Another XML-based SLA management proposition is given in [38]. The authors propose a language which defines a SLA related vocabulary for web services. In their approach the third party monitoring is also supported. In [23] a framework for intelligent virtual organizations is described. The solution allows for controlled SLA negotiations when creating a virtual organization using shared resources provided by business partners. This semantic-based framework allows for automatic setting of security and monitoring software layers able to fulfill the business goals of the virtual organization. Examples of the SLA metrics definitions in the context of distributed environment are presented in [39].

2.4. Summary

Taking into account the presented studies we found no studies comprehensively addressing the SLA usage for distributed storage services and implementing a solution in which each layer is aware of the SLA or QoS. Our research, resulting in the presented model outlined below, thoroughly covers (1) aspects of providing the SLA support for distributed storage services allowing heterogeneity of underlying storage resources, (2) performance related dynamic storage allocation based on current and previous performance metrics and (3) takes into account the users storage QoS requirements resulting from the SLA support.

3. Policy-based SLA storage management model

3.1. High level description

In the context of this paper the SLA storage management is the management of a set of data storage resources aimed at providing QoS according to the SLA. The management relies on selection of appropriate storage resources for a current request of data access with the goal of minimizing of SLA violations. It is assumed that (a) the SLA is already agreed and (b) there are a few rarely changed SLA profiles defined by the provider from which the users select the most adequate for them. The profiles are introduced into the model for differentiating the users by selection of the Storage Nodes (SN) according to the users' needs. SLA policies define importance of the parameters used for the selection of SN (see Sec. 3.3). The Policy-based SLA management model (PoSLAM), built around a mechanism of SLA Policy adaptation (cf. Fig. 1), is introduced to describe time-dependent performance changes of the storage system selection for the goal of keeping users requirements to storage QoS as close as possible.

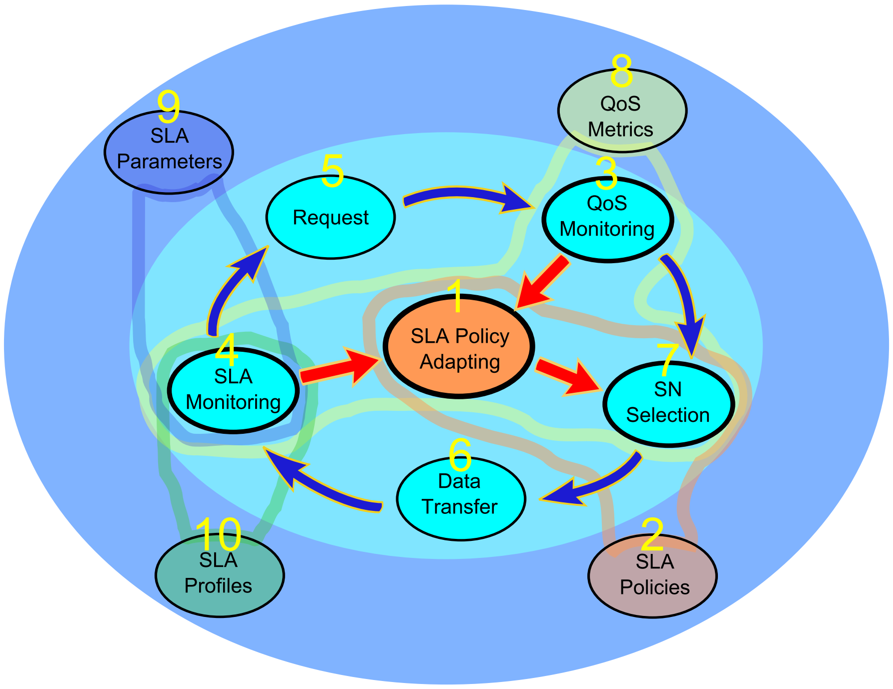

Figure 1. Policy-based SLA storage management model

The mechanism of SLA Policy Adapting (1) modifies SLA Policies (2) taking into account the information obtained by QoS Monitoring (3) and SLA Monitoring (4). When serving Request (5) a suitable storage node for Data Transfer (6) is selected by the process of SN Selection (7) according to the ranking of storage nodes. The selection is done based on the current SLA Policies (2) and the storage QoS metrics (8). SLA Monitoring (4) updates SLA Parameters (9) after Data Transfer (6) and provides information about SLA violations based on SLA Profiles (10). The outer area in Figure 1 contains data entities used as input or modified by the actions in the center. The relations between the data entities and the actions are visualized by color outlines. For example, QoS Metrics (8) are modified by QoS Monitoring (3) and used as input by SN Selection (7), SLA Policy Adapting (1) and SLA Monitoring (4). In the case of SLA violations, indicated by alerts from SLA monitoring, SLA polices are adapted, as described in the next sections, to minimize the chance of SLA violations in the near future.

Since the SLA Policy adaptation prefers less loaded nodes, the model provides natural load balancing. It is assumed that the values of storage QoS metrics and the SLA parameters reflect the actual system state. In reality there is a delay from the moment of the real change of a given system parameter to the moment when it is actually available from the monitoring database. The delay depends on the monitoring measurement interval which reasonable value is a few minutes for data archiving systems like HSM. Ranking of the storage nodes is useful. When the request rates of file access are getting higher the load is to be distributed among the storage nodes according to the ranking list to avoid node overloading. Setting very short update interval can solve the problem, but for the cost of increased monitoring overhead.

Examination of the state of the system and modification the SLA policy are performed by an adaptation algorithm accordingly. The management model however, on which the paper is focused, does not specify how exactly the algorithm works. This has to be specified in detail during implementation of the model given the requirements and characteristics of the utilized distributed storage system. Though, a simple example of adapting algorithm with test results is shown in section 4.

If the performance needed to serve the requests with respect to SLA exceeds the performance capability of the system then some requests will not be served with the desired QoS expressed in the SLA. No admission control is provided to cope with this case since we assume that the number of users and the SLA Profiles are properly planned taking into account users performance usage statistics and infrastructure capabilities.

3.2. Model of storage infrastructure

The model of the storage infrastructure together with the related environment is presented in Figure 2. The storage resources are accessed via Storage Nodes (SN). The SNs provide access methods to the storage resources (e.g., NFS, GridFTP and WebDAV) for Access Nodes (AN). The clients typically access their data by connecting to the ANs located in their data center, but using the AN from another location is allowed. According to the assumptions, the SNs and ANs are connected with a high speed network which does not get saturated by the data transfer.

Three layers – access nodes, storage nodes and physical storage devices – are distinguished (see Fig. 3). Defining $m = |U|$ as the number of users, $n = |AN|$ – the number of access nodes and $l = |SN|$ – the number of storage nodes the following hold $l \approx n$ and $m \gg n$ for U, AN, SN being the sets of users, access nodes and storage nodes respectively. The number of simultaneous transfers per AN differs substantially from m/n, since only small percentage of users transfer data simultaneously in the distributed, nationwide infrastructure.

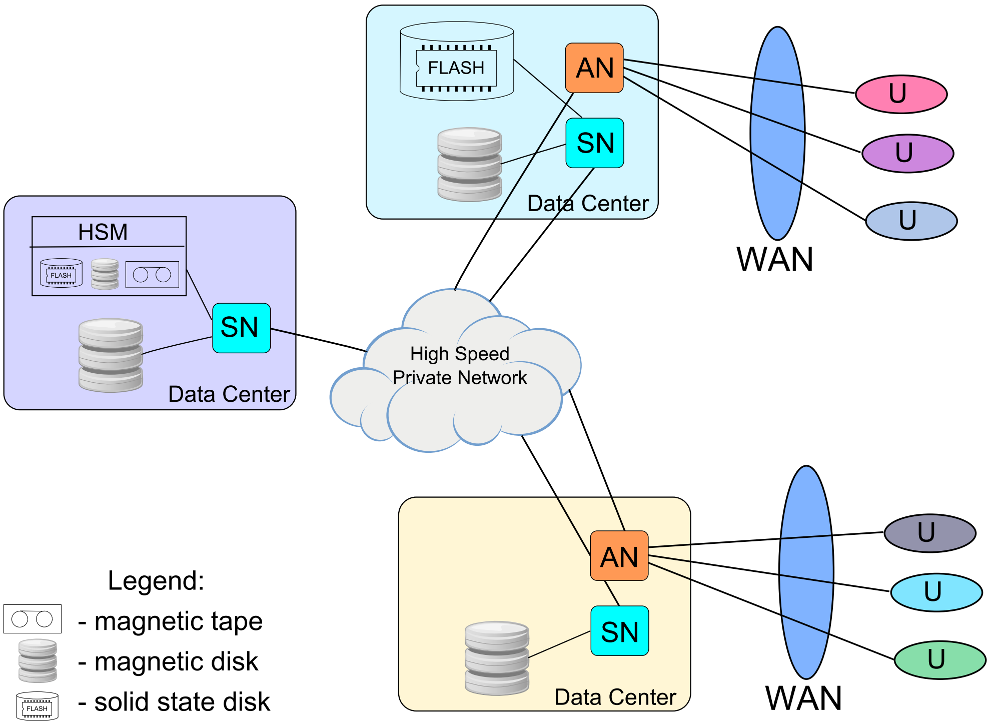

Figure 2. Model of the storage infrastructure

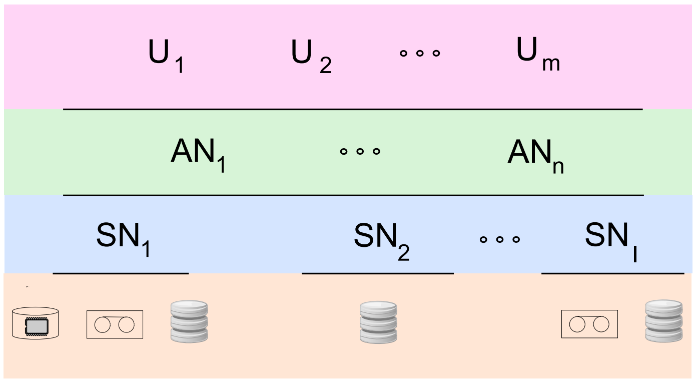

Figure 3. Environment layered model

3.3. Storage SLA parameters and profiles

The SLA parameters, representing values used in defining SLO, provide formalized way to express the user QoS requirements. They are determined based on QoS metrics, which can correspond to raw quantities directly available from the operating system or combined from other metrics. Although several types of SLA parameters can be defined [27, 28], for further considerations only the SLA parameters concerning the performance of `read` and `write` operations are taken into account.

An SLA profile is defined as a set of SLA parameter values which specify typical user QoS requirements. The SLA profiles are objects of more general meaning than SLA policies, for example there can be two SLA profiles — for faster access to data and for the slower basic access — which correspond to different SLA policies defined separately for each profile. Each SLA profile is associated with an SLA policy (see Fig. 4).

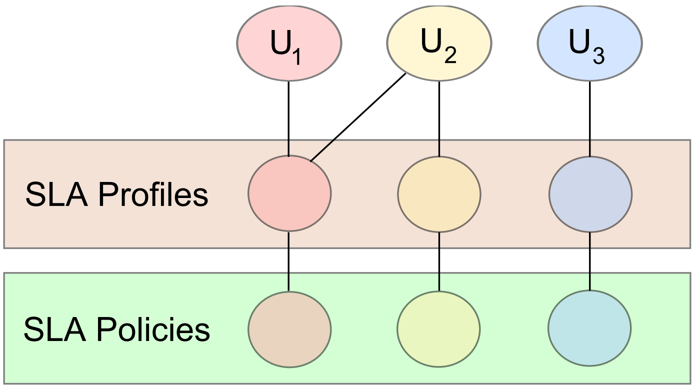

Figure 4. Users, profiles and policies

The SLA polices — selected according to the users' request — are dynamically adopted to time-dependent SN loads during the runtime (cf. section 3.1), that reflects the model assumptions on no strict resource reservations. In our model, the SLA policy, P_{SLA}, is represented by a set of parameters, κ_i,

$$P_{SLA} = (\kappa_1, \kappa_2, \ldots, \kappa_L), \qquad \kappa_{1..L} \in \mathbb{R}. \tag{1}$$

Due to the heuristic approach applied, the set of κ_i parameters is used to build heuristics for defining preferences thus to formulate the SN ranking of storage nodes to be selected, as mentioned in section 3.1. The κ_i parameters can be time-dependent to influence the ranking. Different SLA policies, so different sets of κ_i parameters can be defined. The number of elements, L, is equal to the number of metrics elements used for ranking formulation with preference indicator (see Eq. 3). The usage of sample parameters is presented by the example of preference indicator in Section 4.4.

In general, the profile does not depend on the users preference only. There can be more factors influencing the usage of SLA profiles. One factor can be the internal resource allocation policy of a given computer center, which can have categorized the users in some way and forced them to use a given SLA profile. Other factors are costs – the users may choose worse profiles since they are cheaper or profile limits according to the users rights. Though important, those factors are out of our research currently.

3.4. Storage node selection

One of the requirements for the SLA storage management system is to have a method to differentiate the users by selection the SN according to their SLA profiles. Due to the complexity of the distributed storage systems and according to the efficiency in operation a heuristic preference indicator, *SNP*, for SN selection is introduced, defined for `read` and `write` requests separately. For each user request and for each storage node, SN_I, capable for serving the given request ($SN_I \subseteq S \subseteq SN$), *SNP* is evaluated and the list of k nodes, SN_I is computed,

$$SN_I : SN_I \subseteq S, \text{ where } I = \{i_1, i_2, \dots i_k\}, \text{ s.t.} \tag{2}$$
$$SNP_I = \text{sorted}(SNP)_{\{1,\dots,k\}},$$

where `sorted` means the sorted list in descending order and S is a set of storage nodes capable of serving the given request.

The heuristic preference indicator, *SNP*, is defined for each SN_I as a function of file related metrics, SN metrics and relevant SLA policy,

$$SNP = f(M_F, M_{SN}, P_{SLA}), \tag{3}$$

where

- $M_F = (f_S, f_C)$ are file related metrics, with the file size, f_S, and location of file, f_C, respectively,
- M_{SN} is the SN QoS metrics, with elements $M_{SN} = (sn_1, sn_2, \dots, sn_L)$, where L represents the number of metrics elements,
- P_{SLA} is a SLA policy for the SLA profile used by the requesting user (cf. Eq. 1).

Depending on the f_S value different storage nodes can become suitable. For example, for the small files a low bandwidth/low latency storage node is suitable while for the large files a high bandwidth/high latency node is more useful. The f_C value provides information about the location of the file – if it is cached or not. It directly influences performance being essential for the HSM systems since for the case when the file is cached the access latency can be orders of magnitude lower than otherwise. The storage node QoS metrics, M_{SN}, directly address functional and operational parameters of the SN.

4. Application of PoSLAM in NDS2

In this section an example implementation of the PoSLAM model as the QoS Management System (QMS) module is presented. Also, more details about the implementation and integration of QMS with the real-life National Data Storage service are given with some experimental evaluation.

4.1. National Data Storage

The National Data Storage (NDS2) project has been aimed at providing data storage service mainly for scientific and educational institutions in Poland. The primary purpose of the service is archiving and backup but it also can be used for direct access to the data in the same way the data is accessed from the local disk. The service is provided by a nationwide geographically distributed storage system with nodes located in the main computational centers in Poland. The system uses the high speed network Pionier infrastructure provided as a backbone [31].

The main feature of the system is automatic and safe data storage with end-to-end encryption and data integrity control. In order to provide secure and efficient access to the data a hardware assisted ciphering appliance for data exchanging between the end points and the NDS2 system has been designed and built. The growing users requirements address usability and easiness of access to the data. In result, the NDS2 project takes into account additional requirements concerning hierarchical and distributed users management and advanced accounting based on performance, security, availability and protection profiling. In order to meet those requirements a support for QoS and SLA was needed [28]. The general architecture of NDS2 is presented in Figure 5.

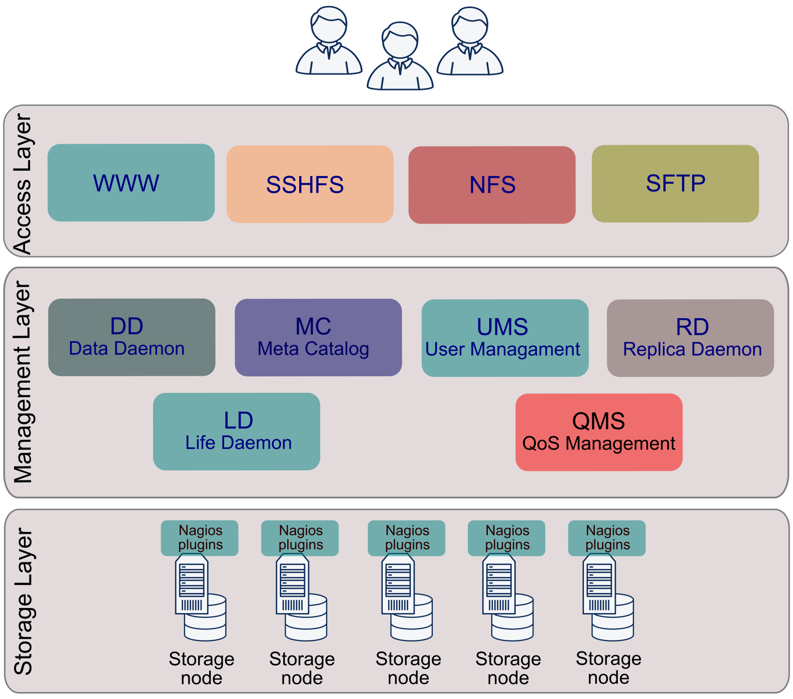

Figure 5. Layered structure of NDS2

Three layers are distinguished: Access Layer, Management Layer and Storage Layer. The Access Layer contains software components providing system storage connectivity options. NDS2 supports various data access protocols and interfaces, like WWW to manage data from a web browser, SSHFS to mount remote user directory accessible via SSH protocol as local directory, NFS and SFTP.

The Management Layer is constructed out of cooperating daemons and services. The core functionality is provided by Data Daemon (DD) and Meta Catalog (MC). DD implements block level data transfers from the storage nodes (SN) via the given access methods to the users. MC implements name space and file attribute operations organizing the files into typical directory-based hierarchy. Management of user accounts is done by the User Management System (UMS). NDS2 supports file replication via the Replica Daemon (RD) which is responsible for creating new replicas and controlling the correctness of replicas creation. Life Daemon (LD) collects information about the health of the other components using the Nagios monitoring system as a basis.

The Storage Layer consists of storage nodes which provide unified access in the form of directory mount point to the underlying storage systems. These storage systems can be of any kind – magnetic disk arrays, solid state storage, distributed filesystems or HSM systems storing efficiently large data volumes on magnetic tapes. Since the primarily purpose of the service is archiving tape-based storage has been chosen. Nagios plugins are installed on the SNs to provide LD with monitoring data.

4.2. Architecture of QMS

QMS controls the performance of the system with respect to the QoS levels specified in the SLA profiles of users (see Fig. 5). It helps DD to make decisions about replica to be used and it helps RD in choosing SN for new replicas.

The architecture of QMS makes use of the PoSLAM model presented in Section 3, while its integration with other modules of NDS2 is shown in Figure 6. It consists of four modules: QMD (QoS Management Daemon), QoS Monitor, SLA Monitor and Presentation Layer. NDS2 uses a monitoring database (DBMS in Fig. 6) to store various monitoring parameters like raw values from Nagios Plugins (NP) run on the storage nodes. QoS Monitor is responsible for processing those raw values and providing convenient QoS metrics which are stored in Database. SLA Monitor calculates SLA parameters based on the QoS metrics and the performance information obtained from the systems logs. SLA Monitor constantly verifies if the SLA parameters do not exceed the SLA limits for the given SLA profile. In the case of SLA violations, indicated by alerts from SLA monitoring, the SLA polices are adapted, as described previously (see Section 3.1) to minimize the chance of near future SLA violations.

The SLA limits are kept in the UMS module set by the administrator or other entities responsible for contract negotiation. The alerts generated by SLA Monitor can be observed by the administrator through the web interface of the Presentation Layer. QMD is responsible for managing the SLA policies based on the data from the

SLA Monitor and the provided SLA Policy Adapting algorithm. Based on the QoS metrics, the SLA policies and the provided heuristics DD decides which storage node to use for the given transfer. It also logs performance data.

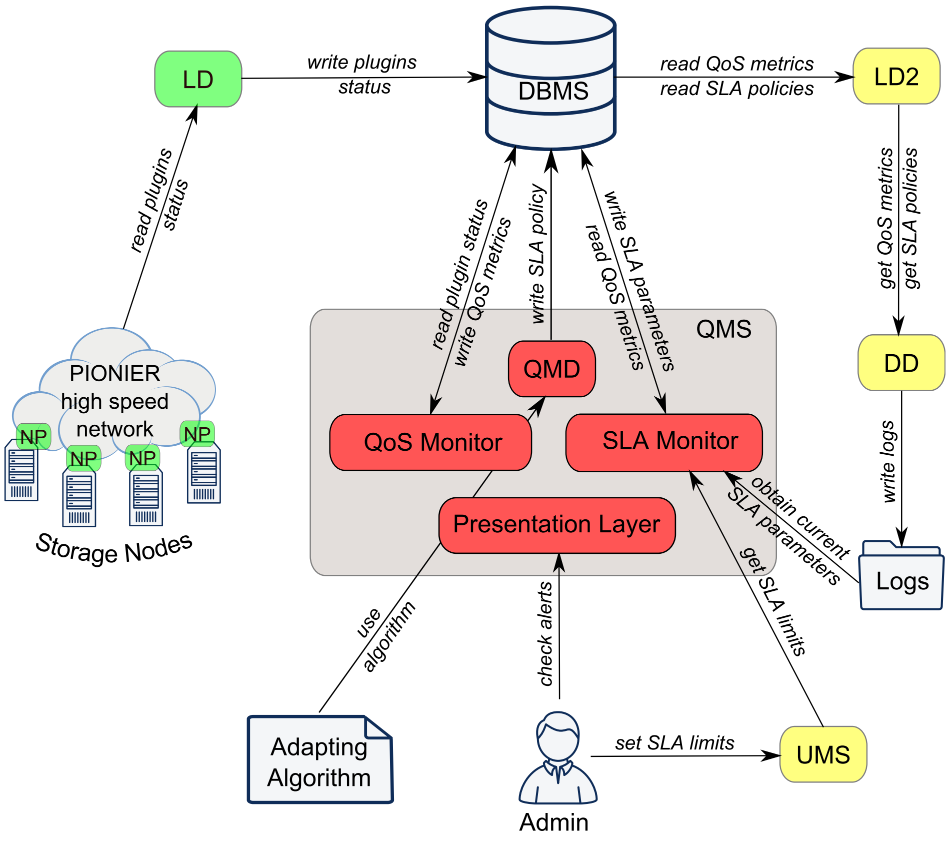

Figure 6. QMS modules and their integration with other NDS2 modules

In this way QMS realizes QoS management model depicted in Figure 1. Namely, QoS Metrics are provided by QoS Monitor and SN Selection is done by DD based on the QoS Metrics and SLA Policies. SLA Monitoring is performed by SLA Monitor and finally the SLA Policy adaptation is done by QMD.

4.3. PoSLAM model implementation in QMS

In the implementation the SLA parameters adopted in the study are shown in Table 1. They are user oriented and represent the performance as experienced by the user, so they are selected as the user metrics for both `read` and `write` operations, utilized by SLA Monitor when checking violations of the SLA as mentioned previously.

The user metrics can be measured at the users side, which gives the most accurate measurement, or at the server side, which generally provides enough accuracy but introduces some network related uncertainty.

Table 1
Storage performance SLA parameters

SLA parameter	Unit	Description
ReadTransferRate	MB/s	Average transfer rate when reading sequentially whole files.
WriteTransferRate	MB/s	Average transfer rate when writing sequentially whole files.
ReadLatencyTime	s	Latency time for read access measured as the time from issuing a request till receiving the first portion of data.
WriteLatencyTime	s	Latency time for write access measured as the time from sending a portion of data till the receiving of successful write acknowledge.

The selected performance SN QoS metrics are presented in Table 2. They are related to a given storage node and provide information about its performance capabilities. The current transfer rate metrics, $R_{SN\_cur\_read}$ and $R_{SN\_cur\_write}$, can be obtained from the operating system. The maximal transfer rate metrics $R_{SN\_max\_read}$ and $R_{SN\_max\_write}$, are selected from a sequence of current transfer rate values stored in the database.

Table 2
Performance SN QoS metrics

M_{SN} metrics	Name	Unit	Description
1	$R_{SN\_cur\_read}$	MB/s	Current transfer rate for data read
2	$R_{SN\_max\_read}$	MB/s	Maximal transfer rate for data read
3	$R_{SN\_cur\_write}$	MB/s	Current transfer rate for data write
4	$R_{SN\_max\_write}$	MB/s	Maximal transfer rate for data write
5	R_{SN_tape}	MB/s	Average transfer rate for accessing data on a tape
6	L_{SN_tape}	s	Average latency for accessing data on a tape
7	λ_{SN}	–	Storage load of SN.
8	ψ_{SN}	IO/s	Number of IO operations per second

$$\lambda_{SN} = \frac{R_{SN\_cur\_read}}{R_{SN\_max\_read}} + \frac{R_{SN\_cur\_write}}{R_{SN\_max\_write}}$$

is a complex metrics (see Table 2) representing the storage load of the node. Generally, it does not exceed 100% but for the storage systems capable of full duplex transfers, it can approach 200% when reading and writing data simultaneously with full speed is performed. ψ_{SN} measures the number of IO operation per second being especially meaningful for random block access for which the disks head positioning time and rotational latency introduce high overhead to the transfer rates. L_{SN_tape} and R_{SN_tape} are the latency and transfer rate of tape drive respectively.

4.4. Implementation of QMS

Having defined the PoSLAM model the preference indicators *SNP* for `read` and `write` operations in this implementation using P_{SLA} policy are:

$$\begin{aligned} SNP_{read} = \;& (k_1 R_{SN\_max\_read} - k_3 R_{SN\_cur\_read}) f_S - \\ & (k_5 \frac{f_S}{R_{SN_{tape}}} + k_6 L_{SN_{tape}}) \; k_7 f_C - \\ & k_8 \lambda_{SN} - k_9 \psi_{SN}, \end{aligned} \tag{4}$$

$$\begin{aligned} SNP_{write} = \;& (k_2 R_{SN\_max\_write} - k_4 R_{SN\_cur\_write}) f_S - \\ & k_8 \lambda_{SN} - k_9 \psi_{SN}. \end{aligned} \tag{5}$$

According to the P_{SLA} policy definition (cf. Eq. 1) the following mapping is implemented, which results from the heuristic form of Equations (4)–(5).

$$(\kappa_1, \kappa_2, \kappa_3, \kappa_4, \kappa_5, \kappa_6, \kappa_7, \kappa_8, \kappa_9) := (k_1, k_2, -k_3, -k_4, -k_5, -k_6, -k_7, -k_8, -k_9) \tag{6}$$

The metrics $R_{SN\_max\_read}$ and $R_{SN\_max\_write}$ inform about potential SN performance and obviously higher values are better, hence, they are included in the formula with positive sign. $R_{SN\_cur\_read}$ and $R_{SN\_cur\_write}$ are current transfer rates for reading and writing respectively. These metrics inform about the current usage of the potential performance and are included with negative sign since higher values indicate higher load. This part of the equation is additionally multiplied by the size of the file, f_S, to amplify it for larger files since the available transfer rate is essential for accessing them.

The metrics, R_{SN_tape} and L_{SN_tape} influence the SNP_{read} function only if the given file is not cached and has to be staged from the tape. This part of the equation roughly estimates the time to access data from the tape. Lower values are better so this part is subtracted. Due to the negative influence on performance of storage load, λ_{SN} and input/output operation per second, ψ_{SN}, they are taken into account with negative coefficients. We can additionally amplify the importance of a given metrics element by increasing the appropriate coefficients of the SLA Policy or we can completely ignore some metrics by setting the corresponding coefficients to 0. As an analytical example, the SNP_{read} function limits for typical metric values are shown in Table 3. In those calculations $f_C = 0$ and $k_1, \ldots, k_9 = 1$.

Table 3
SNP_{read} values

f_S [MB]	$R_{SN\_max\_read}$ [MB/s]	$R_{SN\_cur\_read}$ [MB/s]	λ_{SN}	ψ_{SN}	SNP_{read}
100	100	100	1	200	-201
100000	1000	0	0	0	10^8

As shown in the previous chapters the presented approach introduces P_{SLA} adaptation on a high level of generality. That opens room for development of sophisticated algorithms of adaptation in the stage of production implementation for a given environment. For the purpose of validation however, a sample adapting algorithm for QMS is presented below (see Algorithm 1). The algorithm is based on the assumption that two SLA performance profiles are defined – P_{SLA1} for the standard users and P_{SLA2} for the users whose demand for the transfer rate is no less than the specified limit. Another assumption is that clients using P_{SLA2} are more important, so their current request fulfillment should be most closely related to their SLA. Although keeping the SLA for the clients using P_{SLA1} is not of high priority, the system should try to use them provided if there is an unused storage processing power and this provision would not cause alerts for the clients using P_{SLA2}.

$Alerts_{SLA1}$ and $Alerts_{SLA2}$ give the number of alerts representing violations of the SLA occurred during the last monitoring time interval for clients using profiles, hence policies P_{SLA1} and P_{SLA2} respectively. We assume that single alerts or rare alerts are not essential and can be ignored, but when a certain limit ($AlertLimit_{SLA1}$ or $AlertLimit_{SLA2}$) is exceeded the system should adopt the policies by modifying k coefficients to counteract the SLA violations.

Algorithm 1: Algorithm of changing the SLA policies – an example heuristic

```
1 if $Alerts_{SLA2} > AlertLimit_{SLA2}$ then
2     $k1_{SLA1} = -1$;
3 else if $Alerts_{SLA1} > AlertLimit_{SLA1}$ then
4     if $k1_{SLA1} < 10$ then
5         $k1_{SLA1}$ += 1;
6     if $k1_{SLA1} == 0$ then
7         $k1_{SLA1} = 1$;
```

In order to keep the sample algorithm simple we used only one coefficient, namely $k1_{SLA1}$, for controlling the load distribution taking into account the R_{max_read} or R_{max_write} (depending on the type of the current operation) which just provide the maximal performance capability of a SN. $k1_{SLA1}$ takes signed integer values. Setting $k1_{SLA1}$ to a negative value means reverting of meaning and the less performing SN will have higher *SNP* value and the worst performing SN will be selected for serving a request with the P_{SLA1} profile (see line 2, in Algorithm 1). In other words the algorithm pushes the requests using the P_{SLA1} policy to the worse nodes releasing in this way the performance of the better nodes for serving requests requiring the P_{SLA2} policy.

5. Test results

The system has been tested to verify the usability, correctness and performance of the proposed approach. Details about the tests are presented in the subsections below.

5.1. Deployment environment

The system was deployed in an environment consisting of one AN and three SNs. The AN and one SN were running at the PSNC datacenter (PSNC) located in Poznań while the other two SNs were running at the ACC Cyfronet datacenter (CYF) located in Kraków. The distance between those two cities is 335 km. All nodes are virtual machines provided by a VMware based environment running on a set of HP Proliant DL385 G6 servers with AMD Opteron 2435 2,6GHz CPUs. Different storage resources with different performance characteristics are attached to the SNs via FC links. An additional node named DB is hosting the PostgreSQL DBMS, which stores monitoring data and data related to MC, as well as the MC itself. Connection of all nodes is organized with the use of Pionier national academic network. Currently Pionier allows to achieve single physical link bandwidth of 800 Gb/s using DWDM (Dense Wavelength Division Multiplexing) technology with 80 channels of 10 Gb/s. In the nearest future channels of 100 Gb/s and 400 Gb/s will be available. In the reported experiments. All nodes are connected by a 1Gbps links. More details about the environment are given in Table 4.

Table 4
Deployment environment

Node	Cores	Mem	Storage	Modules	Loc
AN	2	4 GB	–	DD, LD, QMS	PSNC
SN1	2	2 GB	2TB GPFS	NP	PSNC
SN2	2	2 GB	2TB DA	NP	CYF
SN3	2	2 GB	2TB HSM	NP	CYF
DB	1	1 GB	10GB DA	DBMS, MC	PSNC

In the "Storage" column some abbreviations are used to specify the underlying storage systems. GPFS (General Parallel Filesystem) [13] is a distributed filesystem developed by IBM Corp. DA (Disk Array) is a storage volume on an HP EVA8000 array. HSM (Hierarchical Storage Management) is a system which uses different types of storage media (disks and tapes) together. In our environment the HSM system has been built on top of GPFS and TSM (Tivoli Storage Manager) [47].

5.2. Procedure

The goal of the presented procedure/scenario is to study the system behavior in terms of SLA violations in the case when new demanding clients using the P_{SLA2} policy are started in a system which is already serving non demanding clients with the P_{SLA1} policy. The SLA limits are chosen empirically so that the system performs around its limit, which means that if more clients are started then SLA violations will begin occurring. $AlertLimit_{SLA2}$ is set to 2 and $AlertLimit_{SLA1}$ is 4. The experiments are based on issuing data transfer requests to the system using different SLA profiles and access patterns and observing the system's behavior and performance metrics. For the presented tests the following scenario has been used (see Fig. 7a).

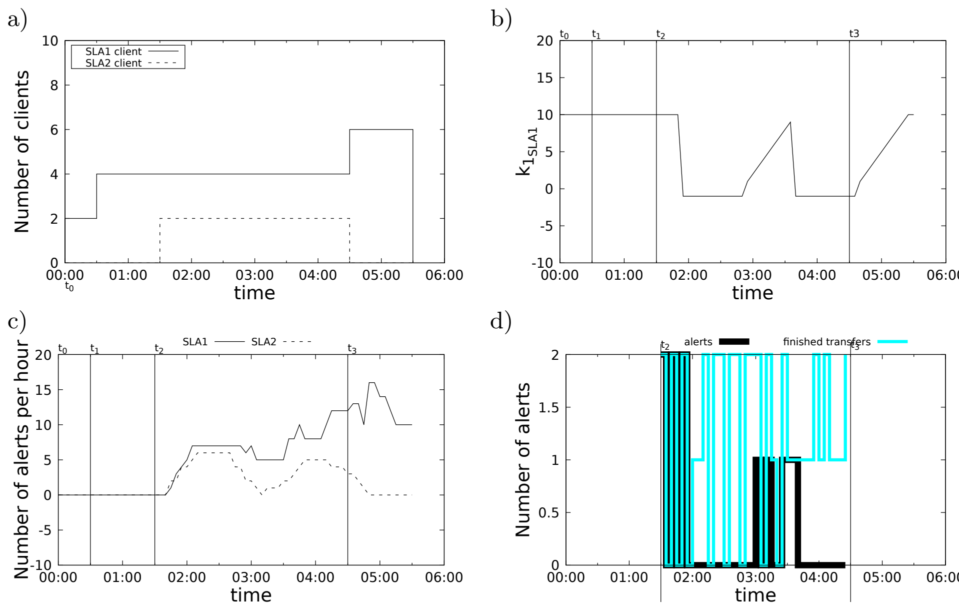

Figure 7. Test results: a) Scenario of tests; b) Policy adapting with $k_{1_{SLA1}}$; c) SLA alerts measured at server; d) SLA2 alerts measured at clients

At $t = t_0$ two clients using P_{SLA1} policy are started which are constantly writing files each of size 6 GB. For those clients the limit of the WriteTransferRate SLA parameter is set to 9 MB/s, which means that if the write transfer from the user's perspective is less than 9 MB/s an alert for that transfer is generated. At $t = t_0 + 30\text{m}$ two more clients using P_{SLA1} are started. At $t = t_0 + 1\text{h}30\text{m}$ two clients with P_{SLA2} are started. For those clients the limit of the WriteTransferRate SLA parameter is set to 11 MB/s. For the next 3 hours the mentioned clients, namely four clients using P_{SLA1} and two clients with P_{SLA2} are concurrently writing data to the system competing for storage bandwidth. After that, at $t = t_0 + 4\text{h}30\text{m}$, the clients with P_{SLA2} are terminated and two more with P_{SLA1} are started. For the next hour 6 P_{SLA1} clients are running. At $t = t_0 + 5\text{h}30\text{m}$ all of the running clients are terminated and the test ends.

‘

5.3. Discussion and limitations

The obtained results are presented in the Figures 7b–7d. Figure 7b shows the changes of $k_{1_{SLA1}}$ during the experiment (due to Algorithm 1), while Figure 7c outlines the number of alerts recorded by QMS during the last hour. The vertical red lines indicate moments when the number or type of clients are changed. We see that until $t = t_2$ system performs well – there are no alerts (see Fig. 7c), so the value of $k_{1_{SLA1}}$ remains unchanged. At $t = t_2$ the demanding clients using P_{SLA2} policy are started and after a while the alerts begin appearing. The system finds SLA violations and changes the values of $k_{1_{SLA1}}$, which decreases the number of alerts for clients with P_{SLA2}. At the same time the number of alerts for clients using P_{SLA1} remains high despite the fact that the limit for them is lower. At $t = 3$h the alerts for P_{SLA2} clients go below the limit, so the system recognizes that it has some more unused storage processing power and starts increasing the value of $k_{1_{SLA1}}$. The number of alerts for P_{SLA1} clients decreases a little but the number of alerts for P_{SLA2} clients starts increasing, so the system brings back $k_{1_{SLA1}} = -1$ and the alerts for P_{SLA2} clients decreases again.

The test shows that the system behaves properly and automatically adopts SLA policy according to our SLA storage management assumptions. Using our SLA storage management method with simple algorithm of changing the SLA policies QMS was able to differentiate clients according to their SLA profiles. The system provides performance to the demanding clients despite the fact that the system was loaded above its limit, which is manifested by the high number of alerts for the non-demanding clients using P_{SLA1} policy.

Figure 7d shows the number of finished transfer and the number of alerts for the given $\Delta t = 5$m interval (in a form of histogram) measured at the client side. We can see that changing the value of $k_{1_{SLA1}}$ (at $t = 1$h50m) has an immediate effect on the transfers as seen from the client side – there are no alerts until $t = 3$h when the value of $k_{1_{SLA1}}$ starts climbing up. This is different from the server perspective since the QMS takes into account alerts occurring one hour in the past (see Fig. 7c). It is important to notice that the tests were conducted on the system which was not fully controlled, since there were more virtual machines using the same hardware. In addition, the storage resources were allocated on storage systems shared with others. Nevertheless, the results of the experiments confirm the potential of the proposed solution.

6. Conclusion and future work

Taking into account the wide range of features implemented in the system, its geographical distribution and the heterogeneity of the underlying storage systems it should be noted that the data management supporting QoS and SLA in such a system becomes challenging. In order to meet those challenges a novel approach to QoS management for distributed storage systems without reservation of storage resources has been presented with the PoSLAM model implemented as QMS module and integrated with the NDS2 system. QMS controls the performance of the system with respect to the QoS levels specified in the users' SLA profiles by changing the given policy coefficients using the adaptation algorithm which can be tuned to the needs of the given system. The advantage of the approach is that the usage of heuristic approach simplifies the performance control and does not introduce non-negligible overhead. Using the model the automatic SLA policy management can be applied for distributed storage systems which allows to distribute efficiently the storage transfer load with respect to the SLA.

The main contribution of the paper is the design of a concept for QoS management with respect to the SLA, i.e., to develop and to implement the PoSLAM model. As mentioned, the model does not specify how exactly the adaptation algorithm works, since it has to be specified in detail during the production implementation in relation to the environment. Due to the real, worldwide environment used for verification, the only simple experiments with a sample algorithm have been possible to conduct. However, the application of the illustrative algorithm for controlling the SLA has proved that the proposed approach allows for differentiating of clients depending on their SLA profile as well as for optimizing of storage resources allocation with respect to the SLA.

Our future works will concentrate on development of new algorithms for adapting of the SLA policies and analyzing their effectiveness. Also, other SLA parameters, concerning data protection, availability, network performance and replication strategies will be taken into consideration, as well as latency distribution (represented by percentile) as a targeted metrics. The research on methods for prediction of SLA violations [30], agent-based adapting [44] and preventive storage management is also planned.

Acknowledgements

This work is supported by AGH-UST statutory grant. The authors are grateful to ACC Cyfronet AGH, Krakow/Poland and to PSNC Poznan/Poland for using the resources. Special thanks go to Gracjan Jankowski for help with the testbed configuration, to Marcin Kuta for support and to the NDS2 Consortium for fruitful cooperation.

References

[1] Andrieux A., Czajkowski K., Dan A., Keahey K., Ludwig H., Nakata T., Pruyne J., Rofrano J., Tuecke S., Xu M.: Web Services Agreement Specification (WS-Agreement), OGF proposed recommendation (GFD.107), 2007.

[2] Andronikou V., Mamouras K., Tserpes K., Kyriazis D., Varvarigou T.: Dynamic QoS-aware data replication in grid environments based on data "importance", *Future Generation Computer Systems*, vol. 28(3), pp. 544–553, 2012. `https://doi.org/10.1016/j.future.2011.02.003`.

[3] Baru C., Moore R.W., Rajasekar A., Wan M.: The SDSC storage resource broker. In: *CASCON First Decade High Impact Papers*, CASCON'10, pp. 189–200. IBM Corp., Riverton, NJ, 2010. `https://doi.org/10.1145/1925805.1925816`.

[4] Billaud J.P., Gulati A.: hClock: hierarchical QoS for packet scheduling in a hypervisor. In: Hanzálek Z., Härtig H., Castro M., Kaashoek M.F. (eds.), *Proceedings of Eighth Eurosys Conference 2013, EuroSys '13*, Prague, Czech Republic, April 14–17, 2013, pp. 309–322, 2013. `https://doi.org/10.1145/2465351.2465382`.

[5] Brzeźniak M., Jankowski G., Jankowski M., Jankowski S., Jankowski T., Meyer N., Mikołajczak R., Zawada A., Zdanowski S. (PSNC): National Data Storage 2: secure storage cloud with efficient and easy data access. In: Foster D. (ed.), *Proceedings of 29th TERENA Networking Conference (TNC)*, 2013. `http://www.terena.org/publications/tnc2013-proceedings/`.

[6] Carlson M., Yoder A., Schoeb L., Deel D., Pratt C., Lionetti Ch., Voigt D.: *Software Defined Storage*, white paper, Storage Networking Industry Association (SNIA), 2015. `http://www.snia.org/sites/default/files/SNIA_Software_Defined_Storage_%20White_Paper_v1.pdf`.

[7] Chuang J.C., Sirbu M.A.: Distributed network storage service with quality-of--service guarantees, *Journal of Network and Computer Applications*, vol. 23(3), pp. 163–185, 2000. `https://doi.org/10.1006/jnca.2000.0109`.

[8] Elnably A., Wang H., Gulati A., Varman P.J.: Efficient QoS for Multi-Tiered Storage Systems. In: Rangaswami R., (ed.) *4th USENIX Workshop on Hot Topics in Storage and File Systems, HotStorage'12, Boston, MA, USA, June 13–14, 2012*. USENIX Association, 2012. `https://www.usenix.org/conference/hotstorage12/workshop-program/presentation/elnably`.

[9] EOSC-hub – Services for the European Open Science Cloud. `http://www.eosc-hub.eu/`.

[10] *eScience on Distributed Computing Infrastructure – Achievements of PLGrid Plus Domain-Specific Services and Tools*, *Lecture Notes in Computer Science*, vol. 8500, Bubak M., Kitowski J., Wiatr K. (eds.), Springer, 2014. `https://doi.org/10.1007/978-3-319-10894-0`.

[11] European Grid Infrastructure. `http://www.egi.eu/`.

[12] GlusterFS. `http://www.gluster.org/`.

[13] GPFS – General Parallel File System. `https://www.ibm.com/support/knowledgecenter/en/SSFKCN/gpfs_welcome.html/`.

[14] Gulati A., Ahmad I., Waldspurger C.A.: PARDA: Proportional Allocation of Resources for Distributed Storage Access. In: Seltzer M., Wheeler R., (eds.) *Proccedings of the 7th Conference on File and Storage Technologies*, FAST'09, pp. 85–98. USENIX Association, Berkeley, CA, USA, 2009. `http://dl.acm.org/citation.cfm?id=1525908.1525915`.

[15] Helix-Nebula – The Science Cloud. `http://www.helix-nebula.eu/`.

[16] Hu C., Deng Y.: QoS-Oriented Capacity Provisioning in Storage Clusters by Modeling Workload Patterns. In: *2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)*, pp. 108–115, 2017. `https://doi.org/10.1109/ISPA/IUCC.2017.00025`.

[17] Hu C., Deng Y., Yang L.T.: On-Demand Capacity Provisioning in Storage Clusters Through Workload Pattern Modeling, *IEEE Access*, vol. 5, pp. 24830–24841, 2017. `https://doi.org/10.1109/ACCESS.2017.2767703`.

[18] Huang H.H., Grimshaw A.S.: Design, implementation and evaluation of a virtual storage system, *Concurrency and Computation: Practice and Experience*, vol. 23(4), pp. 311–331, 2011. `http://dblp.uni-trier.de/db/journals/concurrency/concurrency23.html#HuangG11`.

[19] iRODS – Open Source Data Management Software. `http://www.irods.org`.

[20] Kapanowski M., Słota R., Kitowski J.: Resource Storage Management Model for Ensuring Quality of Service in the Cloud Archive Systems, *Computer Science*, vol. 15(1), pp. 3–18, 2014. `http://journals.agh.edu.pl/csci/article/view/577`.

[21] Karlsson M., Karamanolis C., Zhu X.: Triage: Performance differentiation for storage systems using adaptive control, *ACM Transactions on Storage (TOS)*, vol. 1(4), pp. 457–480, 2005. `https://doi.org/10.1145/1111609.1111612`.

[22] Keller E., Ludwig H.: The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services, *Journal of Network and Systems Management*, vol. 11, pp. 57–81, 2003.

[23] Kryza B., Dutka L., Slota R., Kitowski J.: Dynamic VO Establishment in Distributed Heterogeneous Business Environments. In: Allen G., Nabrzyski J., Seidel E., van Albada G.D., Dongarra J., Sloot P.M.A. (eds.), *Computational Science – ICCS 2009*, *Lecture Notes in Computer Science*, vol. 5545, pp. 709–718. Springer, Berlin – Heidelberg, 2009. `http://dx.doi.org/10.1007/978-3-642-01973-9_79`.

[24] Lin J.W., Chen C.H., Chang J.M.: QoS-Aware Data Replication for Data-Intensive Applications in Cloud Computing Systems, *IEEE Transactions on Cloud Computing*, vol. 1(1), pp. 101–115, 2013. `http://dx.doi.org/http://doi.ieeecomputersociety.org/10.1109/TCC.2013.1`.

[25] Lumb C.R., Merchant A., Alvarez G.A.: Façade: Virtual Storage Devices with Performance Guarantees. In: *Proceedings of the 2Nd USENIX Conference on File and Storage Technologies*, FAST '03, pp. 131–144. USENIX Association, Berkeley, CA, USA, 2003. `http://dl.acm.org/citation.cfm?id=1090694.1090710`.

[26] National Data Storage project. `http://nds.psnc.pl`.

[27] Nikolow D.: Semantic-Based Storage QoS Management Methodology – Case Study for Distributed Environments, *Computing and Informatics*, vol. 31(6), pp. 1345–1366, 2012. http://www.cai.sk/ojs/index.php/cai/article/view/1311.

[28] Nikolow D., Słota R., Lakovic D., Winiarczyk P., Pogoda M., Kitowski J.: Management Methods in SLA-aware Distributed Storage Systems, *Computer Science*, vol. 13(3), pp. 35–44, 2012.

[29] Onedata project. http://onedata.org.

[30] Orzechowski M., Kapanowski M., Słota R., Kitowski J.: Storage System Control with Decision Trees for Storage QoS Provisioning. In: *Proceedings of KU KDM 2014: seventh ACC Cyfronet AGH users' conference*, pp. 91–92. ACK Cyfronet AGH, 2014.

[31] Polish Optical Internet PIONIER. http://www.pionier.net.pl/.

[32] PL-Grid project. http://projekt.plgrid.pl/en.

[33] PL-Grid Core project. http://www.plgrid.pl/en/projects/core.

[34] PL-Grid NG project. http://www.plgrid.pl/en/projects/ng.

[35] Scality RING. http://www.scality.com/ring/.

[36] Shue D., Freedman M.J., Shaikh A.: Performance Isolation and Fairness for Multi-Tenant Cloud Storage. In: Thekkath C., Vahdat A. (eds.), *10th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2012, Hollywood, CA, USA, October 8–10, 2012*, pp. 349–362. USENIX Association, 2012. https://www.usenix.org/conference/osdi12/technical-sessions/presentation/shue.

[37] Shue D., Freedman M.J., Shaikh A.: Fairness and isolation in multi-tenant storage as optimization decomposition, *Operating Systems Review*, vol. 47(1), pp. 16–21, 2013. https://doi.org/10.1145/2433140.2433145.

[38] Skene J., Lamanna D.D., Emmerich W.: Precise Service Level Agreements. In: *Proceedings of the 26th International Conference on Software Engineering*, ICSE'04, pp. 179–188. IEEE Computer Society, Washington, DC, USA, 2004. http://dl.acm.org/citation.cfm?id=998675.999422.

[39] Skitał L., Janusz M., Słota R., Kitowski J.: Service Level Agreement Metrics for Real-Time Application on the Grid. In: Wyrzykowski R., Dongarra J., Karczewski K., Wasniewski J. (eds.), *Parallel Processing and Applied Mathematics, 7th International Conference, PPAM, Gdansk, Poland, September 9–12, 2007, Revised Selected Papers*, *Lecture Notes in Computer Science*, vol. 4967, pp. 798–806. Springer, 2008.

[40] Słota R., Dutka L., Wrzeszcz M., Kryza B., Nikolow D., Król D., Kitowski J.: Storage Management Systems for Organizationally Distributed Environments PLGrid PLUS Case Study. In: Wyrzykowski R., Dongarra J., Karczewski K., Wasniewski J. (eds.), *Parallel Processing and Applied Mathematics, 10th International Conference, PPAM 2013, Warsaw, Poland, September 8–11, 2013, Revised Selected Papers, Part I*, *Lecture Notes in Computer Science*, vol. 8384, pp. 724–733, Springer, 2014.

[41] Słota R., Nikolow D., Kitowski J., Król D., Kryza B.: FiVO/QStorMan Semantic Toolkit for Supporting Data-Intensive Applications in Distributed Environments, *Computing and Informatics*, vol. 31(5), pp. 1003–1024, 2012.

[42] Słota R., Nikolow D., Skalkowski K., Kitowski J.: Management of Data Access with Quality of Service in PL-Grid Enironment, *Computing and Informatics*, vol. 31(2), pp. 463–479, 2012.

[43] Słota R., Nikolow D., Skitał L., Kitowski J.: Implementation of Replication Methods in the Grid Environment. In: Sloot P.M.A., Hoekstra A.G., Priol T., Reinefeld A., Bubak M. (eds.), *Advances in Grid Computing – EGC 2005, European Grid Conference, Amsterdam, The Netherlands, February 14–16, 2005, Revised Selected Papers*, *Lecture Notes in Computer Science*, vol. 3470, pp. 474–484, Springer, 2005. `https://doi.org/10.1007/11508380_49`.

[44] Śnieżyński B.: Agent-based Adaptation System for Service-oriented Architectures Using Supervised Learning. In: Abramson D., Lees M., Krzhizhanovskaya V.V., Dongarra J., Sloot P.M.A. (eds.), *Proceedings of the International Conference on Computational Science, ICCS 2014, Cairns, Queensland, Australia, 10–12 June, 2014*, *Procedia Computer Science*, vol. 29, pp. 1057–1067, Elsevier, 2014. `https://doi.org/10.1016/j.procs.2014.05.095`.

[45] Stoica I., Morris R., Liben-Nowell D., Karger D.R., Kaashoek M.F., Dabek F., Balakrishnan H.: Chord: A scalable peer-to-peer lookup protocol for Internet applications, *IEEE/ACM Transactions on Networking*, vol. 11(1), pp. 17–32, 2003. `https://doi.org/10.1109/TNET.2002.808407`.

[46] Tanimura Y., Hidetaka K., Kudoh T., Kojima I., Tanaka Y.: A distributed storage system allowing application users to reserve I/O performance in advance for achieving SLA. In: *Grid Computing (GRID), 2010 11th IEEE/ACM International Conference on*, pp. 193–200. 2010. `https://doi.org/10.1109/GRID.2010.5697948`.

[47] Tivoli Storage Manager for Space Management. `http://ibm-tivoli-storage-manager.software.informer.com/wiki/`.

[48] Uttamchandani S., Alvarez G., Agha G.: DecisionQoS: an adaptive, self-evolving QoS arbitration module for storage systems. In: *Policies for Distributed Systems and Networks, 2004, POLICY 2004, Proceedings, Fifth IEEE International Workshop on*, pp. 67–76, 2004. `https://doi.org/10.1109/POLICY.2004.1309151`.

[49] Voulodimos A., Gogouvitis S.V., Mavrogeorgi N., Talyansky R., Kyriazis D., Koutsoutos S., Alexandrou V., Kolodner E.K., Brand P., Varvarigou T.A.: A Unified Management Model for Data Intensive Storage Clouds. In: *NCCA*, pp. 69–72, 2011. https://doi.org/10.1109/NCCA.2011.18.

[50] Wang C.M., Yeh T.C., Tseng G.F.: Provision of Storage QoS in Distributed File Systems for Clouds. In: *2012 41st International Conference on Parallel Processing*, Pittsburgh, PA, pp. 189–198, 2012. https://doi.org/10.1109/ICPP.2012.52.

[51] Weil S.A., Brandt S.A., Miller E.L., Long D.D.E., Maltzahn C.: Ceph: A Scalable, High-Performance Distributed File System. In: *Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI'06)*, pp. 307–320, 2006.

[52] Weil S.A., Brandt S.A., Miller E.L., Maltzahn C.: CRUSH: Controlled, Scalable, Decentralized Placement of Replicated Data. In: *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, SC '06. ACM, New York, NY, USA, 2006. http://dx.doi.org/10.1145/1188455.1188582.

[53] Wrzeszcz M., Nikolow D., Lichoń T., Słota R., Dutka L., Słota R., Kitowski J.: Consistency Models for Global Scalable Data Access Services. In: Wyrzykowski R., Dongarra J.J., Deelman E., Karczewski K. (eds.), *Parallel Processing and Applied Mathematics – 12th International Conference, PPAM 2017, Lublin, Poland, September 10–13, 2017, RevisedSelected Papers, Part I*, *Lecture Notes in Computer Science*, vol. 10777, pp. 471–480, Springer, 2017. https://doi.org/10.1007/978-3-319-78024-5_41.

[54] Wrzeszcz M., Opioła L., Zemek K., Kryza B., Dutka L., Słota R., Kitowski J.: Effective and Scalable Data Access Control in Onedata Large Scale Distributed Virtual File System. In: Koumoutsakos P., Lees M., Krzhizhanovskaya V.V., Dongarra J.J., Sloot P.M.A. (eds.), *International Conference on Computational Science, ICCS 2017, 12–14 June 2017, Zurich, Switzerland*, *Procedia Computer Science*, vol. 108, pp. 445–454, Elsevier, 2017. https://doi.org/10.1016/j.procs.2017.05.054.

[55] Wrzeszcz M., Słota R., Kitowski J.: Towards Transparent Data Access with Context Awareness, *Computer Science*, vol. 19(2), pp. 201–221, 2018. https://doi.org/10.7494/csci.2018.19.2.2844.

[56] XDC – Extreme Data Cloud. http://www.extreme-datacloud.eu/.

[57] Xu H., Russell T., Coposky J., Rajasekar A., Moore R., de Torcy A., Wan M., Shroeder W., Chen S.Y.: *iRODS Primer 2: Integrated Rule-Oriented Data System*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers, 2017. https://doi.org/10.2200/S00760ED1V01Y201702ICR057.

Affiliations

Darin Nikolow
AGH University of Science and Technology, ACC Cyfronet AGH, ul. Nawojki 11, 30-950 Krakow, Poland; School of Banking and Management, al. Kijowska 14, 30-079 Krakow, Poland, darin@agh.edu.pl

Renata Słota
AGH University of Science and Technology, ACC Cyfronet AGH, ul. Nawojki 11, 30-950 Krakow, Poland; AGH University of Science and Technology, Faculty of Computer Science, Electronics and Telecommunications, Department of Computer Science, al. A. Mickiewicza 30, 30-059 Krakow, Poland, rena@agh.edu.pl, ORCID ID: https://orcid.org/0000-0001-7424-9317

Stanisław Polak
AGH University of Science and Technology, Faculty of Computer Science, Electronics and Telecommunications, Department of Computer Science, al. A. Mickiewicza 30, 30-059 Krakow, Poland, polak@agh.edu.pl, ORCID ID: https://orcid.org/0000-0003-3041-6683

Marek Pogoda
AGH University of Science and Technology, ACC Cyfronet AGH, ul. Nawojki 11, 30-950 Krakow, Poland and AGH University of Science, M.Pogoda@cyfronet.krakow.pl

Jacek Kitowski
AGH University of Science and Technology, ACC Cyfronet AGH, ul. Nawojki 11, 30-950 Krakow, Poland; AGH University of Science and Technology, Faculty of Computer Science, Electronics and Telecommunications, Department of Computer Science, al. A. Mickiewicza 30, 30-059 Krakow, Poland, kito@agh.edu.pl, ORCID ID: https://orcid.org/0000-0003-3902-8310

Received: 11.05.2018
Revised: 13.09.2018
Accepted: 14.09.2018