Studying the Latency of Concurrent Access to Network Storage*

A.S. Igumnov, A.Y. Bersenev, A.A. Popel, P.A. Vasev

N.N. Krasovskii Institute of Mathematics and Mechanics UB RAS

For many years of the URAN cluster maintenance, there were short periods of abnormal increase in latency of NFS operations. The article proposes the model that describes a possible mechanism for such delays occurrence. Presented experiments confirm that the URAN cluster storage system behavior corresponds to the behavior of the proposed model. In this paper, we assess the statistical data informativeness for identifying applications that cause low latency of the storage system.

Keywords: NFS, latency, performance, parallel computing.

1. Introduction

URAN cluster uses the EMC Unity 300 system to store the user data. The system is available via NFS v3 [1] over TCP protocol. The storage system is used as a home directory for users on the head node and as a data storage on computing nodes.

We use the Collectl program to analyze the I/O load on the compute nodes of the cluster. The following indicators are stored for each cluster node with 30-second interval: NFS-Read/s, NFS-Write/s, Ethernet-out-Bytes/s. These types of indicators are typical for supercomputer monitoring systems [2]. The data was collected for several years, and it is available for the analysis. Also, we obtain an information about the correspondence between computational nodes and computing tasks from the Slurm system [3] log.

The URAN cluster storage subsystem configuration:

- the EMC Unity 300 storage system which includes two disk shelves with 9 and 15 12 Gbps NLSAS disks. The file system uses a RAID10 array of 18 disks and can be accessed via the NFS v3 protocol over TCP by all cluster nodes;
- an Ethernet switch X670V-48t with 48 ports 10GiE and a total bandwidth of 1280 Gbit/s;
- 180 cluster nodes that are connected to the storage system via 1GiE ports;
- 16 cluster nodes that are connected to the storage system via 10GiE ports.

With some kinds of workload generated by computing nodes, the response time of the network file system dramatically grows. This situation looks like an operating system lag for users of the front machine and adversely affects the interactive work of users (editing source texts, compiling programs, etc.).

The purposes of this study are:

- to assess the informativeness of the collected statistical indicators;
- to determine time periods with anomalous latency;
- to determine the type of IO load, leading to increased latency;
- to develop techniques to determine tasks that create such a load.

*Our work was performed using URAN supercomputer of IMM UB RAS

The paper proposes a simulation model of the parallel access to the storage system with internal I/O queue. The behavior of the model with the predominant write operations and alternation write and read operations is described.

The latency of the EMC Unity 300 storage system was measured on simulated access patterns. It is shown that the behavior of the storage system corresponds to the proposed model.

2. Storage System Modeling

For a theoretical assessment of I/O performance of a storage system, we developed a basic simulation model of its work. Also, we developed a program that implements this model. The program was used for dynamic simulation of a storage system operation under various access patterns.

The model is based on the following principles:

1. Both read and write operations are simulated;
2. The storage system performs one operation per N ticks;
3. New operations are queued, the storage system retrieves them one by one. The queue is limited to M entries;
4. Reads are considered completed at the time of execution and writes are considered completed at the time of enqueuing.

In addition to the storage system model, we developed a model of the program, which uses the storage system. The program in the course of its work performs the following actions:

1. Initializes for some time D0. No requests are made to the storage system during this phase;
2. Reads input data, creating R read requests to the storage system. Waits for each request completion;
3. Handles input data for some time D1. No requests are made to the storage system during this phase;
4. Writes output data, creating W write requests to the storage system. Awaits their execution.

When the storage system queue is full, the program tries to repeat the request until it is placed to the queue.

The simulation program creates several virtual processes with individual parameters [D0, R, D1, W] and a storage system with parameters [N, M]. On each time tick, the program behavior is modeled and the order of programs execution is fixed. For each process, there is a graph of how many operations were completed for this process in the tick. The requests pattern specified by the parameters [D0, R, D1, W] is repeated endlessly for each of the processes.

2.1. Results of Storage System Simulation

2.1.1. Simulation of a Single Process

Let us run one process with parameters [2000, 200, 1000,100] on the storage system with parameters [4, 5]. The result of its work is shown in Fig. 1. The horizontal axis shows the time, where every thousandth tick is marked with a dotted line. The vertical axis — the number of operations, where the maximum value corresponds to the maximum number of operations that the storage system is theoretically capable to perform.

We can clearly see the pairs of bursts, which correspond to the read and write phases.

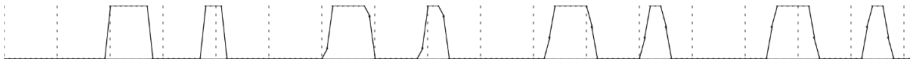

Fig. 1. The result of the simulation of the storage system and one user process

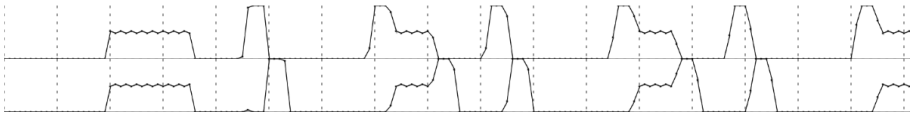

Fig. 2. The result of the simulation of the storage system and two user processes

2.1.2. Simulation of Two Processes with the Same Parameters

Now let us emulate a couple of process with the same parameters as in 2.1.1. The result is shown in Fig. 2.

It may be noticed that the read operations were distributed between processes approximately equally, but the write operations of the first task fully blocked the write operations of the second task. This is due to relatively small size of the request queue, which can be filled by requests of a single task. In addition, the task scheduler here emulates processes sequentially.

As the queue size increases to 500, the graphs for read and write operations are no longer differ (see Fig. 3).

If the queue size is 5 and random scheduler are used, that effect is not so noticeable (see Fig. 4).

2.1.3. Simulation of Parallel Access with One Intensively Writing Process

Let us emulate a NAS with a queue size of 500 and three tasks. Two tasks have parameters [1000, 200, 1000, 100] – waiting for 1000 ticks, performing 200 reads, waiting for another 1000 ticks and performing 100 writes. The third process has the parameters [0, 0, 6000, 100000], i.e. starts to continuously write after the tick 6000. Within one tick tasks are scheduled randomly.

The result of the emulation is shown in Fig. 5.

When the third process starts writing, the storage system almost stops processing requests from the other two. It seems that this result contradicts practice but we managed to reproduce it on two real storage systems. A detailed description of experiments on real storage systems is given below.

If we reduce the queue size to 5, then the third process can no longer exclusively use the most of the storage system resources (see Fig 6).

Thus, reducing the queue size can improve the situation when there are one or many writing processes.

2.1.4. Simulation of Parallel Access with Intensively Reading Processes

Now let us emulate a storage system with a queue size of 500 and three tasks with the following parameters [500, 100, 1000, 0], [600, 100, 1000, 0] and [700, 100, 1000, 0]. Such tasks are common for computing clusters. The result is shown in Fig. 7).

Now we add two constantly reading processes, i.e. processes that have parameters [0, 1000000,

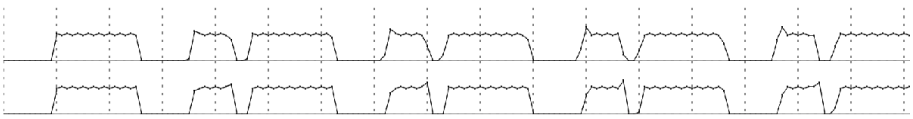

Fig. 3. The result of a storage system simulation with a queue size of 500 and two user processes

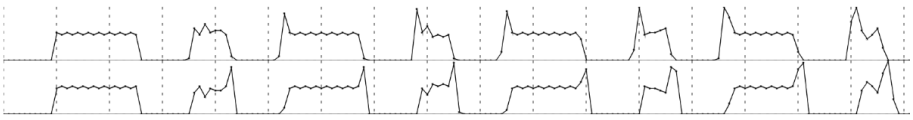

Fig. 4. The result of the storage system modeling with a queue size of 5 and two user processes under random scheduler

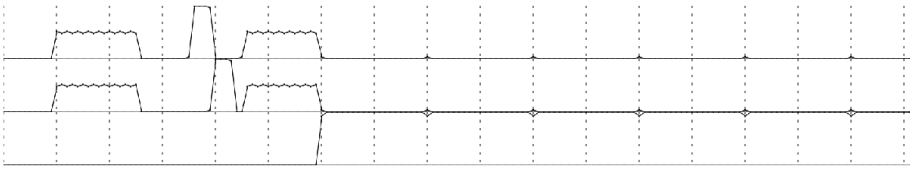

Fig. 5. The result of the storage system simulation with the actively writing process

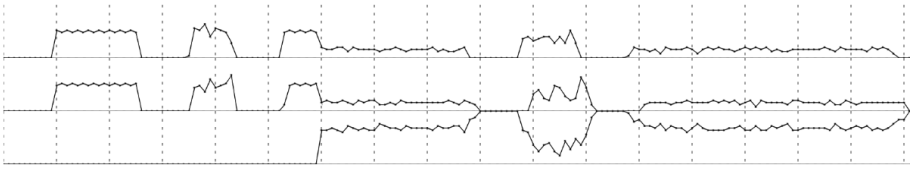

Fig. 6. The result of the storage system simulation with actively writing process when the queue has a small size

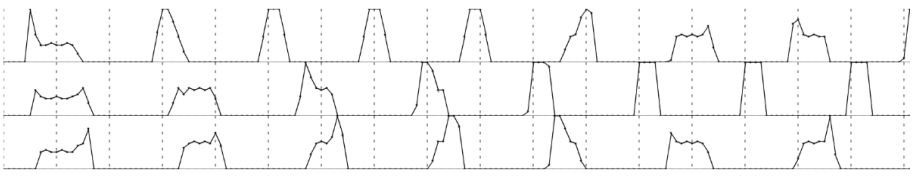

Fig. 7. The result of the storage system simulation with three periodicaly reading processes

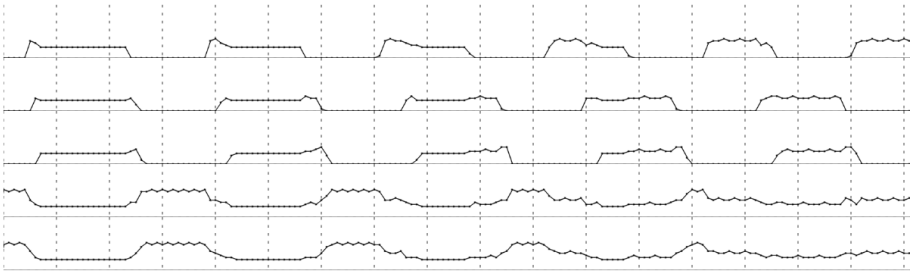

Fig. 8. The result of a storage system simulation with three periodicaly reading processes that and two continuously reading processes

0, 0]. Let us compare the result with the previous one (see. Fig 7 and Fig 8).

Two things can be seen. First: the processes started to longer wait for I/O. It seems a natural result of increasing the load of the storage system. The second thing is less obvious. In the situation of a high load, it is becomes difficult to find a "guilty"process or process group basing only on the number of operation that they perform, because they all contribute roughly equally to the storage load.

3. Stress Testing

3.1. Stress Testing Bypassing the OS Kernel

3.1.1. The Measurement Procedure

In order to reproduce the denial of service situation in the presence of an intensive writing process described in 2.1.3 on a real storage system, the Python script was written. It interacts with a storage system using NFS protocol over several TCP connections, effectively bypassing the NFS implementation in the OS kernel.

To simulate intensive write operations on the storage system, we created two files of 100 megabytes each filled with random content. A stress-testing script was making write requests with various intensity. Every write request overwrites the first eight bytes of the first file with a random data.

To measure the latency, the program reads one byte of the second file every 0.1 seconds and measure the time it took to perform this operation. Then we analyzed the dependence of the longest read operation on the intensity of write requests.

The tests were conducted on the Dell EMC Unity 300 storage system, as well as on the EMC Celerra NS-480 storage system with the RAID controller cache disabled.

3.1.2. The Stress Testing Results

Fig. 9 shows the result of testing the EMC Celerra NS-480 storage system.

Starting with a number of write request per second, there are I/O delays. A further increase in the rate of write requests leads to proportional delay growth up to tens of seconds. For the final users, such delays look like a complete file system failure. A relatively small number of requests processed are caused by a write cache that is disabled.

Results of the same test performed on the Dell EMC Unity 300 storage system show similar behavior. They are presented in Fig. 10.

A high variation of the maximum waiting time in the measurement process can be noticed. When averaging the values, there is a linear dependence of the waiting time on the requests rate.

On larger values of the requests per second, we managed to achieve full denial of service. The storage system has stopped responding, writing logs and passing a self-test. As a result, the

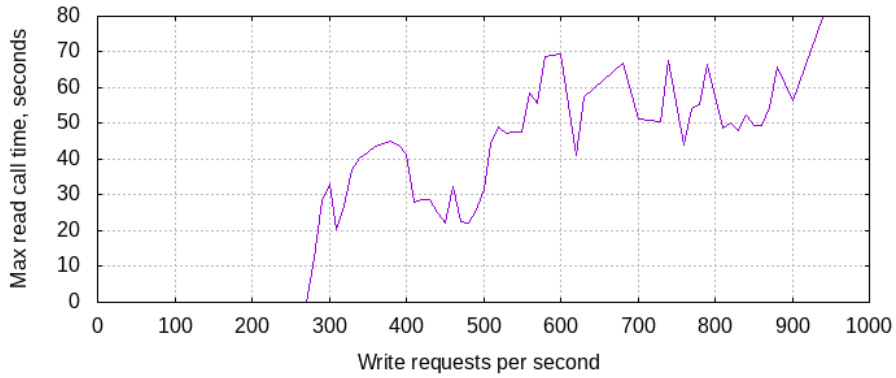

Fig. 9. Dependence of the maximum execution time of read operation on the intensity of write operations on the EMC Celerra NS-480 storage system

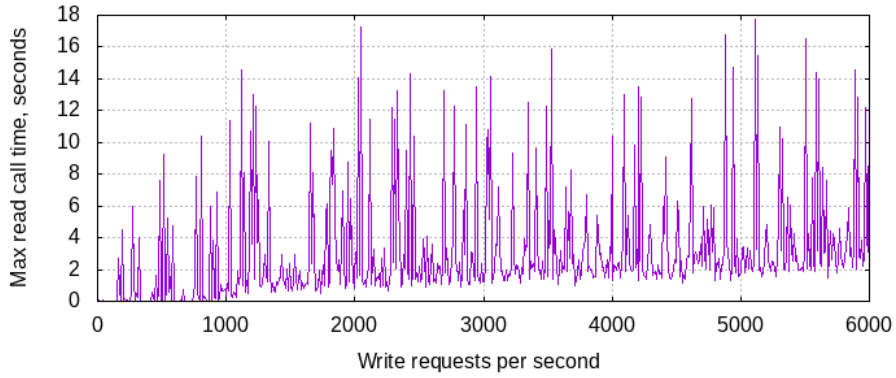

Fig. 10. Dependence of the maximum read execution time on the write operation intensity on the Dell EMC Unity 300 storage system

storage system attempted to reboot their request handling servers.

Thus, in order to cause a denial of service, one intensively writing node is sufficient.

3.1.3. Conclusions

On the tests using the kernel API for accessing the file system of the storage system described in 3.2, significant delays were achieved only with parallel access from 20 or more nodes. This can be explained by the write-caching mechanism built into the OS, which significantly reduces the number of requests to the storage system. For example, when overwriting the data on similar offsets or sequentially writing to a file, several write requests can be combined into one. In addition, if the file was opened for reading/writing using the O_RDWR flag, then each write request is accompanied by a request to read 4096 bytes of data and wait for it to complete. Another significant factor is that the client built into the Linux kernel uses only one TCP connection, which limits the number of outgoing requests to the maximum size of the receive buffer on the storage system side. It can be concluded that the increase in the number of TCP connections in this test roughly corresponds to the increase in the number of active nodes in the real-world conditions.

3.2. Stress testing through the OS kernel API

3.2.1. Measurement technique

Tests in this section simulate the situation of loading the storage system data by user processes [5]. The goal of these tests is to make such load that cause the storage system not responding to requests in adequate time. File creation time was selected as a measure of latency. On a not loaded system, the file creation operation takes about 0.01 s. Excess of this value more than an order of magnitude interpreted as a significant increase in the latency under load.

The storage system response time was measured with the following script:

```
#!/bin/bash

while true; do
  date +%d-%m-%Y.%H-%M-%S
  time touch test.file | grep real && rm test.file
  sleep 1
done
```

This command measures the empty file creation time after which the file is deleted.

To measure the count of file system requests, two data sources were used. On the cluster nodes, the Collectl program monitors the performance of the network file system. Collectl collects information with a 30-second averaging of the use of an Ethernet network and the number of NFS read and write operations. The data storage system has its own monitoring system, which collects information averaged at 10-second intervals. Comparison of Collectl logs and logs of the storage system showed the coincidence of quantitative indicators measured by two different systems with an accuracy of up to 10%.

To eliminate the influence of the local cache on the nodes, it was forcibly pushed out onto the storage system and then cleared with the command:

```
while true; do
  echo 3 > /proc/sys/vm/drop_caches
  sleep 0.1
done
```

Tests were run on 62, 96 and 144 nodes. Each of the tests includes three subtests with sizes of written blocks of 1 Kb, 64 Kb, and 1024 Kb respectively.

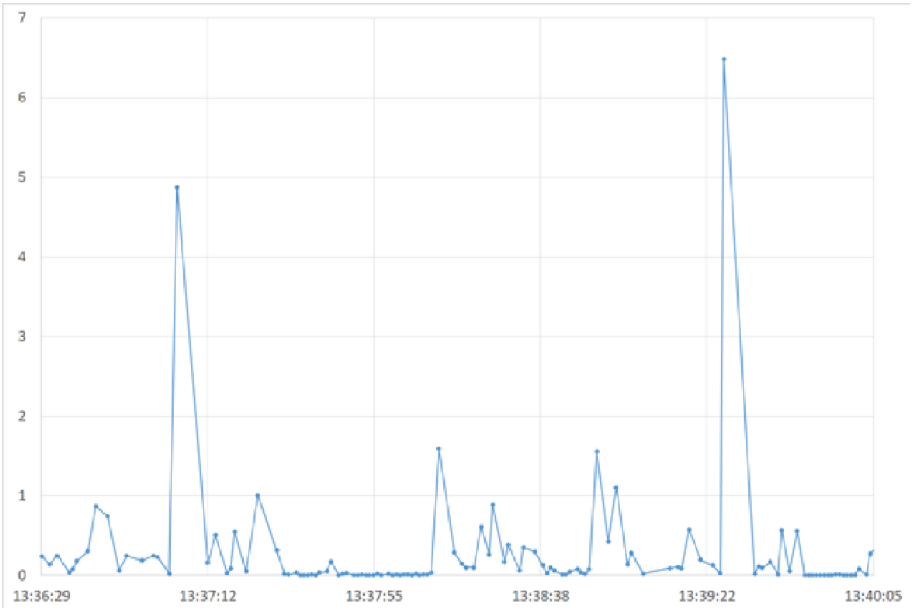

Fig. 11. Variation of measurements at short time intervals

3.2.2. Test result

At short intervals of time, measurements show a large variability of latency, seen Fig. 11. The calculation of the maximum over a large interval is a rather informative indicator of the increased latency over the averaging period.

The graphs show data on a series of experiments, the parameters of which are as follows:

- From 10:34 to 10:47 62 nodes writes blocks of 1 Kb;
- From 10:51 to 10:58 62 nodes writes blocks of 64 Kb;
- From 11:04 to 11:11 62 nodes writes blocks of 1024 Kb;
- From 11:21 to 11:33 96 nodes writes blocks of 1024 Kb;
- From 11:41 to 11:51 96 nodes writes blocks of 64 Kb;
- From 12:10 to 12:29 96 nodes writes blocks of 1 Kb;
- From 12:32 to 13:03 144 nodes writes blocks of 1 Kb;
- From 13:08 to 13:33 144 nodes writes blocks of 64 Kb;
- From 13:52 to 14:15 144 nodes writes blocks of 1024 Kb

On a graph showing the number of NFS IOPS, one can observe the characteristic values of the number of IOPS for a different number of write processes.

During the simultaneous recording of a large number of files the limiting factor is the location of hard drives. The corresponding IOPS for individual disks show that in this mode they reach the maximum value. This situation is visible on the graph (Fig. 12) as a cut at approximately 2000 operations per second and does not depend on the size of the block. On a small number of recording processes, other characteristic values are distinguished: 4000 IOPS, which is typical when recording by a small number of processes and a value of 6000 IOPS, which is close to the upper limit.

Testing has shown that increasing of the number of writing processes significantly affects latency value than the block size (Fig. 13).

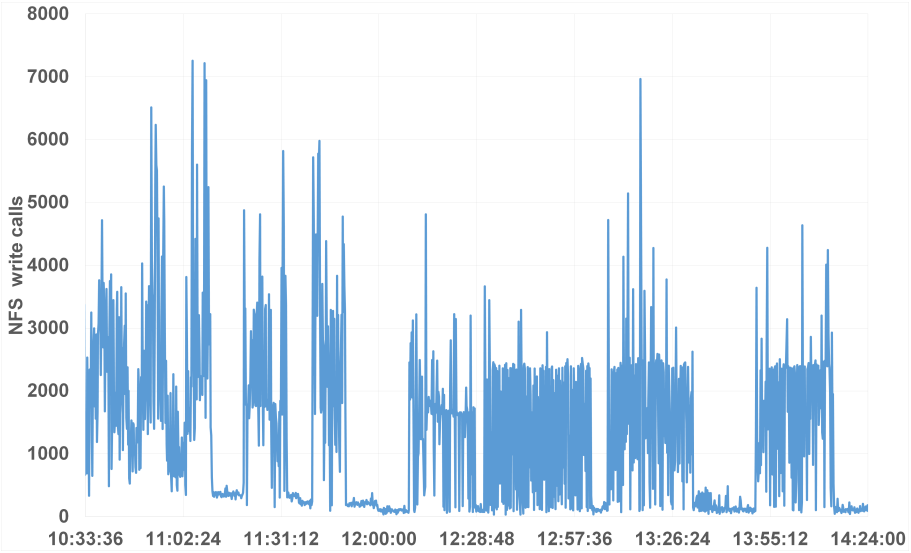

Fig. 12. Number of NFS write requests according storage system logs

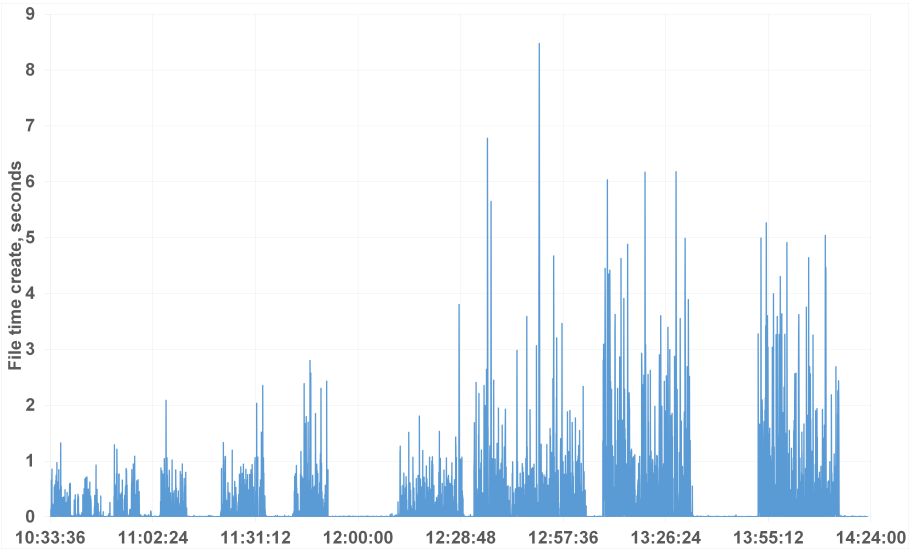

Fig. 13. File creation times

4. Analysis of Collectl logs

Due to the fact that NFS I/O activity is registered at each node independently, it is possible to compare certain types of workload with specific user tasks obtained from job management systems Slurm.

In accordance with the proposed simulation model, when there are a large number of processes with large amounts of I/O, it becomes impossible to determine which process caused the request queue to overflow. To test this hypothesis, we analyzed 30-second time intervals, at which the total rate of IO operations on all nodes is about 2000 IOPS. This value was chosen because it corresponds to the value obtained at maximum load in the tests.

The analysis of logs for the year 2018 showed that at each time interval, a number (from 2 to 50) of nodes can be distinguished which account for more than 50% of all I/O operations. Selective additional analysis showed that these nodes are associated with one or two tasks running across on this nodes. A sufficiently large number of tasks do not deal with IO during the execution time. From this, it follows that the theoretical possibility of spreading I/O for several tasks does not come true on the existing task flow. Therefore, the indicator of the count of I/O operations on the node is informative and allows to accurately identify tasks with large I/O.

To perform an additional visual analysis of collected NFS IO metrics, a specialized visual analytics system was created. It provides different views to a supercomputer state. The primary view is presented in the following figures.

The view's annotation is:

1. The time increases from the bottom to the top.
2. Each column corresponds to a user of supercomputer.
3. The width of the column at each specific time is the sum of NFS IOPS generated by all the running tasks of that user.
4. An investigator may choose to emphasise time moments when the sum of supercomputer's IOPS fits into a specific range.

Both figures represent data for the period of October 2018. For printing accuracy, only a subset of the users is presented on these figures.

The visual study with various parameters brought us to the following conclusion:

- Most of the time, the supercomputer operates at 0..1000 NFS IOPS (Fig. 14).
- In the remaining time, there are spikes of the NFS usage. Those spikes are clustered into various types, and the most widespread spike type observed is 3500..4500 IOPS (Fig. 15).

Visual analytics appeared to us to be an effective tool for both illustration and understanding. It seems to have an additional potential for new investigations in our case.

It's common to use visual analytics for supercomputer operation analysis. Arising questions may be too specific, which leads to the development of new specialized visual analytics systems. One of the examples of a new visual analytics system for supercomputing may be found at [4].

Our visual analytics system operates in a web browser, loads data in JSON format, and uses WebGL for rendering. It is developed using Viewlang [6], a modern 3D web engine for scientific visualization with built-in support for virtual reality.

5. Conclusion and Future Work

A model for storage system access latency has been developed. The reliability of the model was experimentally confirmed by two independent stress tests. The model is applicable to at least two models of storage systems - Dell EMC Unity 300 and EMC Celerra NS-480.

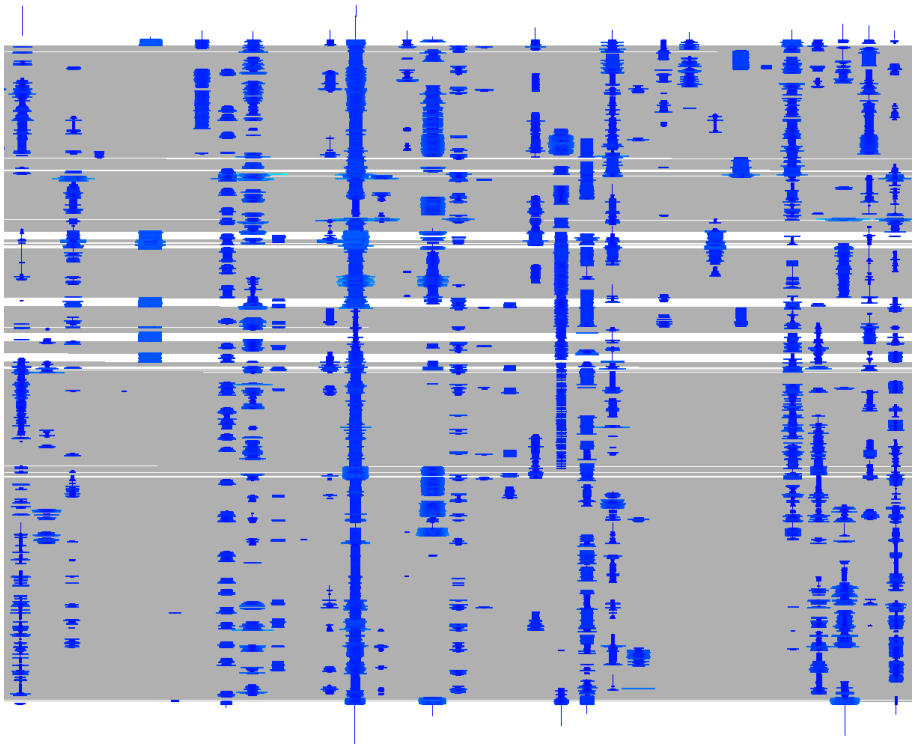

Fig. 14. Per-user NFS IOPS during one month. Grey background means time when supercomputer operated at 0..1000 IOPS

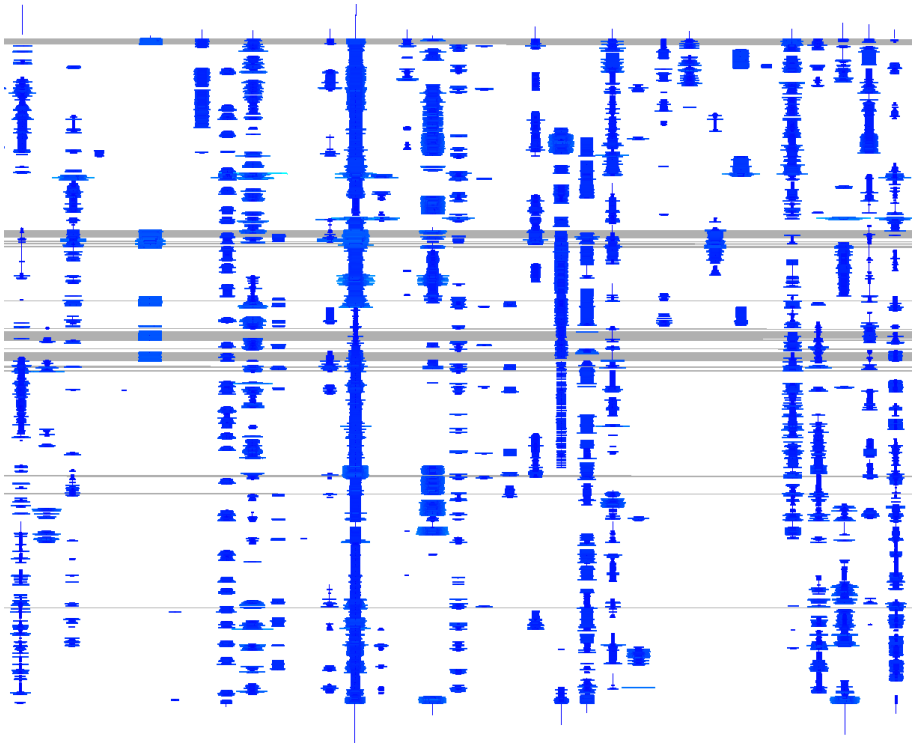

Fig. 15. Horizontal grey lines are the times when supercomputer operated at 3500..4500 IOPS

In the existing I/O history of the cluster storage system URAN, it was not possible to isolate the uniform distribution of cluster nodes I/O rates predicted by the model. Most likely, this is caused by a small percentage of tasks with large amounts of I/O.

Experiments with an artificial load of NFS have shown that a high rate of small-sized I/O operations lead to a significant increase in the latency of access to the storage system with very little traffic. Thus, when searching for tasks leading to increased input latency, IOP/s counters are more informative than network traffic counters.

At this stage of the research, it was not possible to reveal a clear correlation between the IOPS values and the latency. An increased latency was observed at IOPS values close to the limiting ones; however, high IOPS values themselves during the execution of computational tasks did not always lead to an increase in latency.

To track the latency of the storage system, we need to add a separate indicator to the statistics being collected. Such an indicator can use the measurement of the execution time of a non-cached NFS operation. Due to the large variability of the time for performing such an operation, this indicator should be polled as often as possible and saved as the maximum value over the averaging period.

References

1. Pawlowski B., Juszczak C., Staubach P., Smith C., Lebel D., Hitz D. Nfs Version-3 - Design and Implementation. Berkeley: Usenix Assoc, 1994.

2. Nikitenko D., Antonov A., Shvets P., Sobolev S., Stefanov K., Voevodin V., Voevodin V., Zhumatiy S. Jobdigest detailed system monitoring-based super- computer application behavior analysis // Communications in Computer and Information Science. 2017. Vol. 793, P. 516–529. DOI: 10.1007/978-3-319-71255-0 42.

3. Slurm workload manager. URL: https://slurm.schedmd.com/ (accessed: 27.11.2018)

4. Galkin T., Grigoryeva M., Klimentov A., Korchuganova T., Milman I., Padolski S., Pilyugin V., Popov D., Titov M. The new approach to monitor the workflow management system ProdSys2/PanDA of the ATLAS experiment at LHC by using methods and techniques of visual analytics // Scientific Visualization. 2018. Vol. 10, No. 1. P. 77–88. DOI: 10.26583/sv.10.1.06.

5. Ellard D., Seltzer M. NFS Tricks and Benchmarking Traps. Berkeley: Usenix Assoc, 2003.

6. Averbukh V.L., Bakhterev M.O., Vasev P.A., Manakov D. V, Starodubtsev I.S. Development of approaches to realization of specialized visualization systems // GraphiCon'2015 proceedings. P. 17–21.