





Role-Dependent Resource Utilization Analysis for Large HPC Centers

Dmitry Nikitenko^(✉) , Pavel Shvets , Vadim Voevodin ,
and Sergey Zhumatiy 

Research Computing Center, Lomonosov Moscow State University, Moscow, Russia
{dan,shpavel,vadim,voevodin}@parallel.ru

Abstract. The resource utilization analysis of HPC systems can be performed in different ways. The method of analysis is selected depending primarily on the original focus of research. It can be a particular application and/or a series of application run analyses, a selected partition or a whole supercomputer system utilization study, a research on peculiarities of workgroup collaboration, and so on. The larger an HPC center is, the more diverse are the scenarios and user roles that arise. In this paper, we share the results of our research on possible roles and scenarios, as well as typical methods of resource utilization analysis for each role and scenario. The results obtained in this research have served as the basis for the development of appropriate modules in the Octoshell management system, which is used by all users of the largest HPC center in Russia, at Lomonosov Moscow State University.

Keywords: HPC center management
Application efficiency analysis · User roles · Analysis scenarios
Supercomputer

1 Introduction

1.1 The Variety of Resource Utilization Analysis Levels

Nowadays, the issue of computing resource utilization efficiency is a very hot topic. There are many points of view and research subjects that fit into this issue, such as workload efficiency, power consumption, and others. The most interesting thing is that all these aspects effect efficiency, and one has to take into consideration many of them at a time to gain a realistic overall picture. Moreover, there are many different levels of efficiency analysis, especially when

The results were obtained at the Research Computing Center of Lomonosov Moscow State University. The work was partially funded by the Russian Foundation for Basic Research (grant № 17-07-00719), and with financial support from the Russian Science Foundation (grant № 17-71-20114) in the part of the program implementation described in Sect. 4. The research was carried out on equipment of the shared research facilities of HPC resources at Lomonosov Moscow State University.

one considers HPC and distributed computing. For instance, we can define four levels of computing resources.

First, at the top of the pyramid, a supercomputer center administration is interested in global figures and resource utilization rates with almost no need to go through all the messy details of thousands of applications. At the same time, it is natural at this level of observation to have an interest in comparing workload with resource utilization rates of supercomputers available at an HPC center.

Second, a close level is one at which one studies resource utilization and workload for a specific system. No system holder is interested in wasting costly resources for the whole system and for each system partition.

Third, at this point levels stop being mapped to any specific part of the HPC system. This is the level of research projects. Every research project can have a number of participants that run jobs on some or all HPC center computers. Of course, both system holder and project member are interested in details of the project resource utilization, at least to fit into the granted amount of resources for the project.

The last but not the least is the level of application run. Every job is interesting because it has an effect both on the whole HPC center resource utilization profile and on its own job efficiency, as it can significantly bring closer or delay the obtention of the result.

The set of these levels or layers can be extended to a more complicated hierarchy, but even at this point we can see obviously different scopes of interest with personalized accents on some specific system utilization parameters. Moreover, every level requires its own access permissions, so we see different roles of users at each level of abstraction.

As soon as we speak about resource utilization, one of the most common techniques of getting all required information is system monitoring. There is a diversity of various monitoring systems that are focused on specific targets. In this paper, we keep to the tools that have been developed or adopted and widely used in our practice at Lomonosov Moscow State University HPC center.

1.2 The Paper Structure

The “Background” section describes the current state at MSU regarding the paper topic, which served as the basis for the research. The next section, “The Proposed Approach Principles”, provides a description of selected key roles and scenarios of resource utilization study. The “Implementation” section provides technical solution details. The “Evaluation” section describes our experience in using the methods developed during our research. The “Conclusions” section lists further steps of research and development. The “References” section ends the paper.

2 Background

Understanding resource utilization profiles regarding both machines and research projects has always been an element of primary importance at every HPC center [1]. The larger the HPC center is, the more important that element is. This has always been a hot topic for the Lomonosov State University HPC center as the largest of this kind in Russia [2].

There is an impressive number of various approaches to performance and efficiency analysis for HPC applications [3–7]. Nonetheless, the peculiarities of running a large academic supercomputing center drove MSU to develop a set of mutually reinforcing and complimentary tools and methodologies. Every part of this toolkit has originally been developed as an open-source tool. Figure 1 gives a short overview of the tools hierarchy.

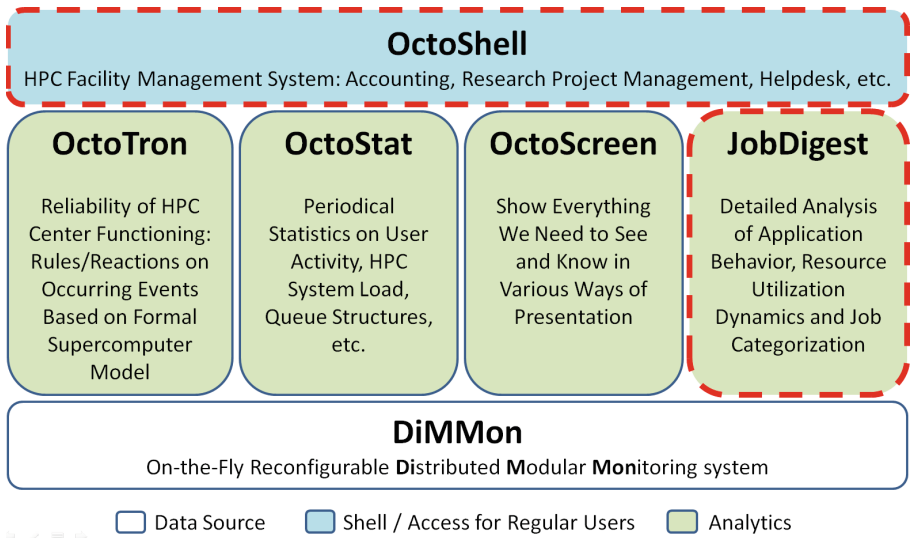


Fig. 1. JobDigest and OctoShell system as a part of the MSU HPC center toolkit (Color figure online)

The work described in this paper extends the interaction of JobDigest [8,9], a detailed application analysis tool, with Octoshell [10,11], a general management system. These two blocks are shown enclosed in dashed red boxes in Fig. 1.

The JobDigest¹ approach provides details on resource utilization for every application. This can be done in various ways [12,13], but generally the JobDigest reports can also be superfluous and some extra lightweight forms, the mini

¹ JobDigest® is a Russian registered trademark. An application for the creation of the JobDigest approach was filed and the corresponding patent was granted.

JobDigest reports, may be required. JobDigest was originally developed as a precise tool that can be used both by experienced users and by beginners, by users and by administrators. Nevertheless, the important issue of private and business data isolation from third party or unauthorized users has not been thoroughly studied yet.

As a result, this analysis tool, though perfect at its main objective, really needs to implement access privilege techniques for sharing collected data for jobs, systems and components between authorized users only, and the development of mini JobDigest is required for quick job reviews.

At this point, the authors are quite happy with having Octoshell, a modular management system, at their disposal. This system was originally developed to serve as a connecting link between managed objects of totally different kinds: accounts, users, projects, quotas, and so on. Notably, a set of user roles are present in the system basics.

These facts are a good basis for the development of a special OctoShell module that would take advantage of existing roles and authorization mechanisms of the OctoShell system and grant access to projects logics. Such a module would be aimed at encapsulating JobDigest reports in all forms, securing access to sensitive user, project or application data, as well as providing a user-friendly interface to the available resource utilization submodules for every user of the center according to the access level regarding the specified level of observation.

3 The Proposed Approach Principles

One of the first, and most important questions that should be answered in the very beginning of development is who is going to use the proposed services and what typical scenarios of usage can every type of user go through.

Actually, the main contribution of the paper is the way we combine these two things. From one side, there is a set of typical user role definitions. On the opposite side, there are typical scenarios and usage cases that are often encountered in resource utilization studies based on system monitoring and resource management data.

3.1 Levels of Analysis

As it has already been mentioned in the introduction, we can go down from the level of overall system observation to the level of detailed job analysis.

Here we emphasize the following levels of abstraction and observation:

1. Overall job run states.
2. Integral job characteristics.
3. Detailed job information.
4. Heuristics and ML-based reports.
5. HW/SW failure influence.
6. Other custom levels.

Overall Job Run States. Overall job run statistics is the top level of abstraction, which represents the actual resource utilization by the whole system or its part for a specified period of time. In our opinion, it is reasonable to limit the bottom of this level to the system partition level. The log files of most resource managers allow for grouping finished and running jobs into categories by job state and, what is more, for summarizing utilized CPU or core hours. System monitoring integration allows calculating sums for any other resource amounts utilized and/or granted. So, in a very similar way, one can observe the distribution of jobs according to their states and utilized resources for every partition, system or the whole HPC center.

It is quite useful to have an option to quickly jump to more detailed information for specified partition jobs, or even for jobs with a certain state, say, "TIMEOUT". In other words, an option to go deeper into details, going down to the next level of observation.

Integral Job Characteristics. At the integral job characteristics level, one starts seeing the details of jobs. At his step, job details are presented as basic information from the resource manager, supplied with average rates of dynamic job characteristics, such as CPU_user, load average, etc., and tags for every job. Integral job characteristics can be highlighted according to some rules or thresholds. At this step, the user can see all the available jobs with easy-to-understand general job information: was it resourceful in terms of memory, CPU or GPU usage, did it finish normally, and so on.

This step is obviously expected to have a possibility to proceed to the details of a chosen job. The detailed job information is the next to the bottom level of abstraction.

Detailed Job Information. Detailed job information can be provided in various ways. The most natural for us is the JobDigest approach. The JobDigest reports provide basic job information received from the resource manager and dynamical job characteristics from the monitoring system, in the form of heat maps, diagrams or raw data for export and further analysis. It also provides tags for every job, i.e. some automatically (and/or manually) assigned categories based on thresholds or more complicated rules.

The problem is that such a report sometimes is redundant, that is why we have introduced the lightweight version containing no diagrams but showing all important information: the so-called mini Job Digest. It is specially designed to be provided to every user of the HPC center, and for every job. If required, it can also contain unique links to the full JobDigest report version.

Heuristics and ML-Based Information. Of course, thresholds are still of a significant value to identify many categories of jobs, but recently a number of methodologies have evolved that provide efficient methods for class revealing, similarity study, and so on [14, 15]. There is an interesting direction of research at

the MSU HPC center devoted to such techniques. It allows revealing anomalies both in job profiles and in job queues [16].

Even though it does not have a user interface yet, the results obtained are already quite promising and we expect a special module to appear and become available soon.

HW/SW Failure Influence. It is quite natural that one of the root causes of drop-downs in the efficiency of an HPC system and its applications are failures of system hardware, such as interconnect interface, or software, such as problems with schedulers. Sometimes, such problems are found and fixed almost immediately. Nevertheless, the influence of such factors can be on occasions critical for the result or accuracy.

That is why we keep in mind a tool that would allow matching jobs with known problems all over the system, based on resilience system logs. In our case, the OctoTron system [17].

Other Custom Levels. As we realistically look at the problem, we understand that we should support extending this set of levels with new ones as soon as it is wanted and developed.

3.2 User Roles

Regular User. Actually, regular user is the most important role, just because all these systems are originally designed to perform actions that allow achieving real-life research goals by a scientist or an engineer. And that explains the scope of interest of most users. Some regular users do not care much about efficiency of applications, but if a user has some limitations like disk quota or limited CPU time, that becomes critical as it can prevent from obtaining the results in time or at all. Users who run their codes or packages regularly usually feel more interested in the efficiency and execution time of the routines. Moreover, most users still understand that efficient resource utilization is beneficial for everybody: both for application owners and for system holders.

Anyway, the variety of users determines the scope covered by the analysis.

The important thing is to provide means to collaborate in job efficiency and study overall stats regarding workgroup activity.

Another important task is to secure job-related data from being accessed by any other regular user outside the workgroup. One can configure the system in a way to limit job-detail access rights either to the set of jobs run by the owner or to the set of jobs run by the workgroup which the user is a member of.

Project Manager. Project manager is almost a regular user with one key difference: responsibility for the workgroup actions, being the official representative of the workgroup. So, in any case, it is quite natural for such role to have access to all personal jobs and to job stats of the workgroup members. The main difference from the user is a more concentrated focus on overall statistics, as the

project manager is more concerned about keeping the project to the granted amount of resources.

Administrator. Going to the other side, system administrators are originally targeted at running HPC systems and helping users to overcome difficulties while using these machines. That implies covering all possible levels of observation in all possible combinations.

System holders or HPC center managers have almost the same rights that administrators have, but like project managers are more focused on overall stats on system usage, and certain workgroup or account activity.

Expert. There can be supervisors with some reduced scopes of analysis. For example, a role that can be used for real-time open demonstrations of what is going in the center right now, but only for some special events regarding a selected workgroup or partition.

As for the MSU HPC center, we actively use the Expert role for annual project expertise. This allows experts to see the job history and details only of those sets of accounts that belong to the project that has been assigned to the expert for review.

3.3 Jumps Between Levels of Analysis

The described levels of analysis, as noted, are interrelated. In order to develop a more convenient and effective tool, we consider the following requirements for quick links between levels.

- Jump from overall job run states to a list of jobs with more detailed, integral characteristics for a selected set of logins (i.e. projects), for a certain state, for a certain queue, for a certain system, or for combinations thereof.
- Jump from the list of jobs to a sublist of jobs (specification of the list of jobs by tags, dates, etc.).
- Jump from the job list to detailed job info, mini JobDigest for a selected job by its ID.
- Jump from a mini JobDigest to a full-format external JobDigest for a selected job by its ID.

3.4 Functional Description of the Interface

We consider the following basic functional features for each one of the proposed levels of the prototype.

Overall Job Run States. Purpose: granted resource-utilization rate assessment by user applications and an estimation of conformity of resources utilization to allocated limits.

Content: average and total amounts of CPUh, GPUh, disk usage for multiple logins grouped by whole systems, partitions, job states.

Filtration: by system, by partition, by job states, by time interval, by project, by login.

Features: job data access segregation: user (own logins), project manager (own logins and managed projects' logins), expert (logins of additional projects assigned to an expert), administrator (all logins, all projects).

Additional features: comparison with allocated quotas for a project; comparison with the same period preceding a displayed interval, quick jump to a job list corresponding to the selected group (for example, all completed jobs or all successfully completed jobs in a specified section).

Integral Job Characteristics (Job List). Purpose: qualitative assessment of resource utilization by jobs, search for abnormal launches, comparison of application runs.

Content: a list of jobs with characteristics and color markup.

Filtration: by system, by partition, by job states, by time interval, by project, by login, by values range for each characteristic, by tags.

Features: job data access segregation: user (own logins), project manager (own logins and managed projects' logins), administrator (all logins, all projects).

Detailed Job Information. Purpose: qualitative assessment of resource utilization by a job.

Content: reduced version of JobDigest: integrated characteristics and data from the resource manager.

Features: job data access segregation: user (own logins), project manager (own logins and managed projects' logins), administrator (all logins, all projects).

Additional functions: unique link to the full JobDigest report.

4 Implementation

Let us now describe the technical implementation. All the tools are implemented as a module of the OctoShell system, which allows using the built-in roles separation mechanism, while users get access to a generally familiar interface, so as to expect a more successful and frequent use of the development by the users.

All necessary data is stored locally in the system and is obtained from a third-party tool operating in 24/7 mode, which builds a full-format JobDigest, allocates categories, and so on. The Octoshell job service retrieves all data from an external supercomputer job data storage and processing service.

Data access is performed using ORM technique, and Ruby on Rails web app development framework is used. All general data are stored in a database table

with a structure as shown in Table 1. Fields `id`, `login`, `start_time`, `end_time` are used for indexing. It allows to speed up the most common requests for user job querying in a selected time interval.

Integral job characteristics are stored using three tables in the database.

The first table contains three fields: `id`, `name`, `type`. The `name` field holds the name of the characteristic, and the `type` field its type (numeric or text). This table is used to identify what kind of characteristics are available and what kind of data they present.

Table 1. The structure of the general job information storage table

Attribute	Description
<code>id</code>	Entry ID
<code>job_id</code>	Job ID
<code>login</code>	System user name
<code>partition</code>	Supercomputer partition
<code>account</code>	Accounting user name
<code>submit_time</code>	Submit time of the job
<code>start_time</code>	Start time of the job
<code>end_time</code>	End time of the job
<code>timelimit</code>	Time limit of the job
<code>job_name</code>	Name of the job
<code>state</code>	State of the job
<code>priority</code>	Priority of the job
<code>req_cpus</code>	Number of requested cores
<code>alloc_cpus</code>	Number of allocated cores
<code>nodelist</code>	List of allocated nodes

The other two tables have the same structure, as shown in Table 2.

Those two tables are used for storing actual integral characteristics data. The only difference between them is the type of their value field.

The `id` and `task_id` fields are used for indexing. To obtain the integral characteristics for a task, one should query the characteristics metainfo from the first table and query actual data from the corresponding characteristic table.

The service allows displaying the short version of the JobDigest with optional access to the full JobDigest as an external service.

The structure of the short JobDigest is stored using a table similar to the one described previously (`id`, `name`, `type`). That table stores the description of the monitoring sensors used in JobDigest. Values are stored in a table with a structure as show in Table 3.

Table 2. The structure of the job integral characteristics storage table

Attribute	Description
id	Entry ID
name	Name of the characteristic
task_id	Job entry ID (see Table 1)
value	Value

Table 3. The structure of the job dynamic characteristics storage table

Attribute	Description
name	Name of the characteristic
task_id	Job entry ID (see Table 1)
time	Time
value	Value

The `id` and `task_id` fields are used for indexing. The access to the full Job-Digest is granted with a unique URL which is stored as usual job characteristics of text type.

The service allows using tags assigned to a job. Tags are stored in a table with a structure as show in Table 4.

Table 4. The structure of the job tags storage table

Attribute	Description
id	Entry ID
name	Tag name
task_id	Job entry ID (see Table 1)

All fields are used for indexing.

Updates are performed using external POST requests.

The first request is used to update general JSON information. If a job is not present, then a new entry is added.

The second type of request inserts data about the integral characteristics into the database, and the data is transmitted in the body of a POST request in JSON format.

The third type of request inserts the tag data into the database, and the data is sent in the body of a POST request in JSON format.

The fourth type of request adds to the database data about a series of changes in the value of the sensor during job operation. The data is sent in the request body in CSV format.

The overall system workflow is shown in Fig. 2.

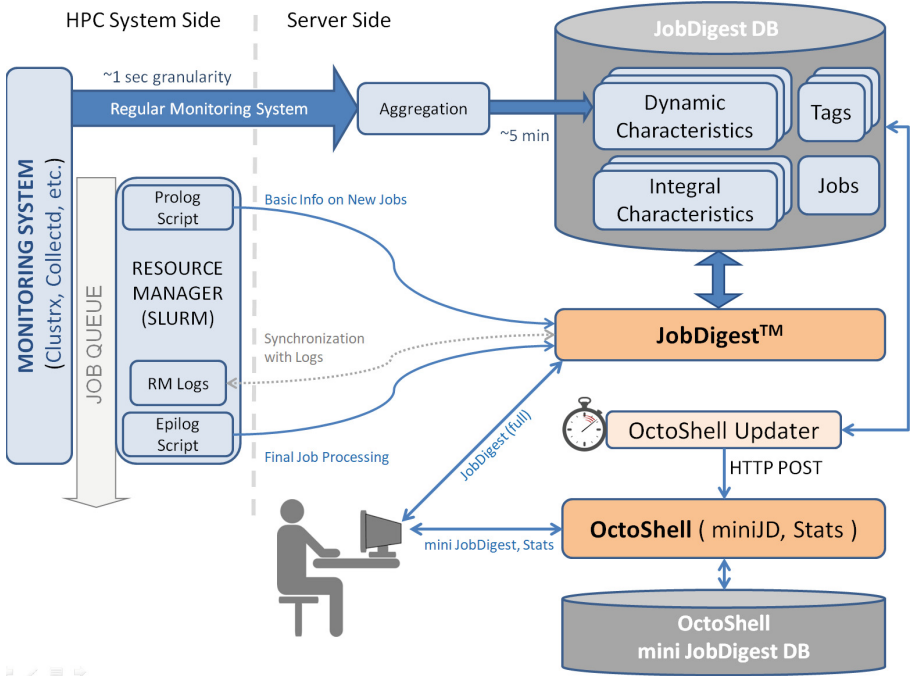


Fig. 2. General OctoShell mini JobDigest DB workflow

5 Evaluation

The implemented prototype is available for users of the MSU HPC center. At present, we are collecting feedback that should aid us in further approach elaborations. We hereby thank one of the workgroups at the MSU HPC center for depicting the interface. All presented data correspond to real research [18, 19].

Figure 3 illustrates the interface for the level of overall job run states. We can see that the “Lomonosv-2” system was used by the project only during the 2017Q2. It is quite nice to see that users did really use the test partition for testing. The majority of resources have been spent for successfully completed jobs in the compute partition during the period.

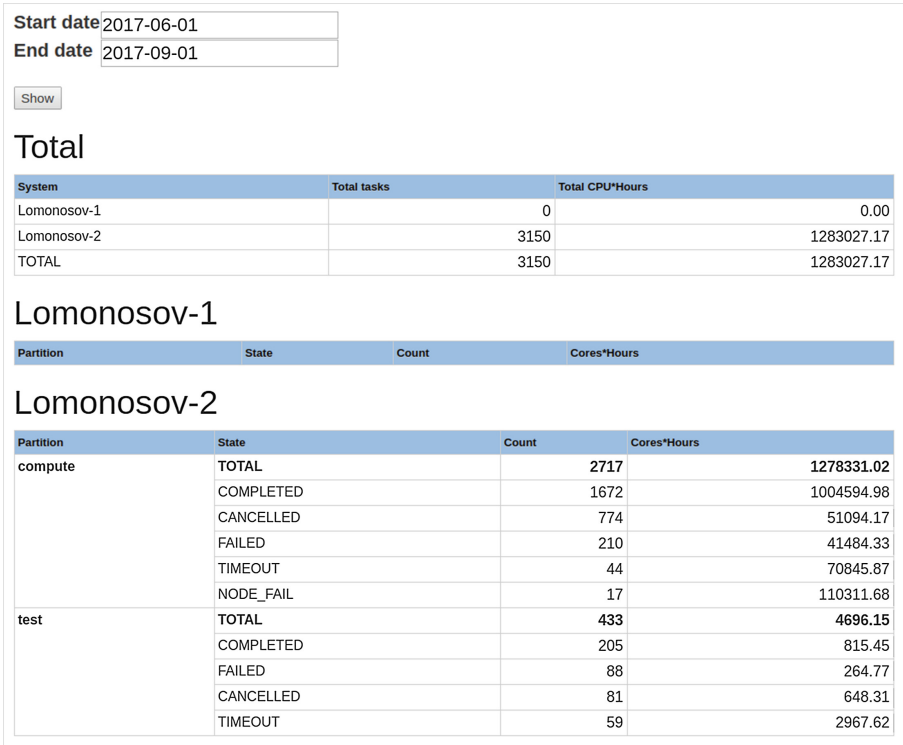


Fig. 3. States of selected project jobs for a certain period of time with total resources utilization example

Figure 4 provides the details for the compute partition run jobs. One can see job IDs, allocated amount of resources, and actually spent CPU time. This list can be easily enriched with general integral job characteristics, such as average CPU_user, Load Average, network usage, etc.

Figure 5 shows a prototype of the mini JobDigest tool. We can see all general data on the job, including command line and node list. Note that we can also see the average resource utilization rates highlighted with colors based on thresholds. The job tag corresponding to the job category of jobs with poor cache data stats has been imported also from the full size JobDigest report.

This type of short but informative report seems to be sufficient for most regular users for an initial job analysis. Nevertheless, the set of characteristics in such a brief report is subject to investigation and will be updated based on users' feedback.

Job table query interface

From: 20.10.2017 To: 27.10.2017 Cluster: Lomonosov-1

Owned logins: vurdizm

Involved logins: emel_251996

States: All, Completed, Failed, Cancelled

Partitions: All, compute, gpu, regular4

Show

Results (limited to 100)

cluster	job_id	login	partition	start_time	end_time	state	num_cores	duration(h)	cores*hours	cpu load	gpu load	loadavg	ib receive	ib sent
lomonosov-2	460043	vurdizm	compute	10/21/17 03:06:13	10/21/17 06:38:50	COMPLETED	14	3.5	49.6	50.5	0.0	15.0	117.2	117.2
lomonosov-2	460042	vurdizm	compute	10/21/17 02:58:09	10/21/17 06:31:08	COMPLETED	14	3.5	49.7	49.6	0.0	15.0	0.0	0.0
lomonosov-2	459417	vurdizm	compute	10/20/17 05:31:41	10/20/17 23:25:03	CANCELLED	140	17.9	2504.5	47.0	0.0	15.0	0.0	0.0
lomonosov-2	459415	vurdizm	compute	10/20/17 04:00:09	10/20/17 23:24:54	COMPLETED	56	19.4	1087.1	79.4	0.0	26.1	3.8	0.0

Fig. 4. List of selected project jobs for a certain period of time in the specified section with example of details

Job info for 459415

/mnt/scratch/users/vurdizm/students/shupanov/science/cub/X=0.0/System 0/without obr/t=0.001/polymer.sh

General info

cluster	lomonosov-2
Job id	459415
Login	vurdizm
State	COMPLETED
Partition	compute
Num cores	56
Submit time	10/20/17 00:42:26
Start time	10/20/17 04:00:09
End time	10/20/17 23:24:54

Job performance

Metric	Value	Ranking
Average CPU load (%)	79.35	average
Average Loadavg	26.14	good
Average GPU load (%)	0.00	low
Average IB receive data (MB/s, total MPI + FS)	3.85	low
Average IB send data (MB/s)	0.04	low

Fig. 5. Example of mini JobDigest report (Color figure online)

6 Conclusions

In the near future, our plans have a strong focus on usability for regular users. At the same time, there are at least two levels of analysis to be added to the prototype. The first is a machine-learning-based module for anomaly detection, and the second is a role-sensitive situational screen based on earlier research, known as OctoScreen or TentaView [20].

We would also like to encourage all interested HPC users to contact the authors if additional implementation and functional details are required.

References

1. Voevodin, V., Voevodin, V.: Efficiency of exascale supercomputer centers and supercomputing education. In: Gitler, I., Klapp, J. (eds.) ISUM 2015. CCIS, vol. 595, pp. 14–23. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-32243-8_2
2. Voevodin, V., et al.: Practice of “Lomonosov” supercomputer. *Open Syst. J.* **7**, 36–39 (2012)
3. Gunter, D., Tierney, B., Jackson, K., Lee, J., Stoufer, M.: Dynamic monitoring of high-performance distributed applications. In: Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing, pp. 163–170 (2002). <https://doi.org/10.1109/hpdc.2002.1029915>
4. Mellor-Crummey, J., Fowler, R.J., Marin, G., Tallent, N.: HPCVIEW: a tool for top-down analysis of node performance. *J. Supercomput.* **23**(1), 81–104 (2002). <https://doi.org/10.1023/A:1015789220266>
5. Jagode, H., Dongarra, J., Alam, S., Vetter, J., Spear, W., Malony, A.D.: A holistic approach for performance measurement and analysis for petascale applications. In: Allen, G., Nabrzyski, J., Seidel, E., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2009. LNCS, vol. 5545, pp. 686–695. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01973-9_77
6. Adhianto, L., et al.: HPCTOOLKIT: tools for performance analysis of optimized parallel programs. *Concurr. Comput.: Pract. Exper. J.* **22**(6), 685–701 (2009). <https://doi.org/10.1002/cpe.1553>
7. Kluge, M., Hackenberg, D., Nagel, W.E.: Collecting distributed performance data with dataheap: generating and exploiting a holistic system view. *Procedia Comput. Sci. J.* **9**, 1969–1978 (2012). <https://doi.org/10.1016/j.procs.2012.04.215>
8. Nikitenko, D., et al.: JobDigest - detailed system monitoring-based supercomputer application behavior analysis. In: Voevodin, V., Sobolev, S. (eds.) RuSCDays 2017. CCIS, vol. 793, pp. 516–529. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-71255-0_42
9. JobDigest components. https://github.com/srcc-msu/job_statistics
10. Nikitenko, D., Voevodin, V., Zhumatiy, S.: Resolving frontier problems of mastering large-scale supercomputer complexes. In: ACM International Conference on Computing Frontiers (CF 2016), pp. 349–352. ACM, New York (2016). <https://doi.org/10.1145/2903150.2903481>
11. Nikitenko, D., Voevodin, V., Zhumatiy, S.: Octoshell: large supercomputer complex administration system. In: Russian Supercomputing Days International Conference, Moscow, Russia, CEUR Workshop Proceedings, vol. 1482, pp. 69–83 (2015)

12. Nikitenko, D., Stefanov, K., Zhumatiy, S., Voevodin, V., Teplov, A., Shvets, P.: System monitoring-based holistic resource utilization analysis for every user of a large HPC center. In: Carretero, J., et al. (eds.) ICA3PP 2016. LNCS, vol. 10049, pp. 305–318. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49956-7_24
13. Nikitenko, D.A., et al.: Supercomputer application integral characteristics analysis for the whole queued job collection of large-scale HPC systems. In: 10th Annual International Scientific Conference on Parallel Computing Technologies, PCT 2016, Arkhangelsk, Russian Federation, CEUR Workshop Proceedings, vol. 1576, pp. 20–30 (2016)
14. Movchan, A., Zymbler, M.: Time series subsequence similarity search under dynamic time warping distance on the Intel many-core accelerators. In: Amato, G., Connor, R., Falchi, F., Gennaro, C. (eds.) SISAP 2015. LNCS, vol. 9371, pp. 295–306. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25087-8_28
15. Rechkalov, T., Zymbler, M.: Accelerating medoids-based clustering with the Intel many integrated core architecture. In: Proceedings of the 9th International Conference on Application of Information and Communication Technologies (AICT 2015), 14–16 October 2015, Rostov-on-Don, Russia, pp. 413–417. IEEE (2015). <https://doi.org/10.1109/ICAICT.2015.7338591>
16. Voevodin, V., Voevodin, V., Shaikhislamov, D., Nikitenko, D.: Data mining method for anomaly detection in the supercomputer task flow. In: Numerical Computations: Theory and Algorithms, The 2nd International Conference and Summer School, Pizzo calabro, Italy, 20–24 June 2016, AIP Conference Proceedings, vol. 1776, pp. 090015-1–090015-4 (2016). <https://doi.org/10.1063/1.4965379>
17. Antonov, A., et al.: An approach for ensuring reliable functioning of a supercomputer based on a formal model. In: Wyrzykowski, R., Deelman, E., Dongarra, J., Karczewski, K., Kitowski, J., Wiatr, K. (eds.) PPAM 2015, Part I. LNCS, vol. 9573, pp. 12–22. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-32149-3_2
18. Rudyak, V., Krakhalev, M., Sutormin, V.: Electrically induced structure transition in nematic liquid crystal droplets with conical boundary conditions. *Phys. Rev. E* **96**, 052701-1–052701-5 (2017). <https://doi.org/10.1103/PhysRevE.96.052701>
19. Guseva, D., Rudyak, V., Komarov, P., et al.: Crosslinking mechanisms, structure and glass transition in phthalonitrile resins: insight from computer multiscale simulations and experiments. *J. Polym. Sci. Part B: Polym. Phys.* (2017). <https://doi.org/10.1002/polb.24548>
20. Nikitenko, D., Zhumatiy, S., Shvets, P.: Making large-scale systems observable – another inescapable step towards exascale. *Supercomput. Front. Innov. J.* **3**(2), 72–79 (2016). <https://doi.org/10.14529/jsfi160205>