УДК 004.021

JOB DIGEST: ПОДХОД К ИССЛЕДОВАНИЮ ДИНАМИЧЕСКИХ СВОЙСТВ ЗАДАЧ НА СУПЕРКОМПЬЮТЕРНЫХ СИСТЕМАХ

А. В. Адинец[1], П. А. Брызгалов[1], Вад. В. Воеводин[1], С. А. Жуматий[1], Д. А. Никитенко[1], К. С. Стефанов[1]

Вместе с ростом масштаба вычислительных систем и решаемых на них задач растет и сложность написания эффективных программ. Причина этого кроется в том, что также возрастает и множество факторов, которые могут влиять на эффективность приложений. Свойства аппаратного и программного обеспечения суперкомпьютера, свойства самой исполняемой программы, взаимное влияние исполняемых программ друг на друга — все это необходимо учитывать для достижения высокой производительности. Это приводит к необходимости создания инструмента, который позволит разобраться, где, а главное почему происходит потеря производительности при выполнении программ и использовании суперкомпьютеров. В настоящей статье мы расскажем о разрабатываемом инструментарии и подробно остановимся на одном из используемых в нем подходов, который предназначен для исследования поведения задачи во время выполнения. Данный подход изучает динамические свойства задач, исследуемые с помощью средств мониторинга. Его цель состоит в предоставлении как администратору системы, так и пользователю базовых характеристик задачи по ее завершению для получения как качественной, так и детальной оценки каждого отдельно взятого запуска. Рассматриваемый подход, а также полученный в результате его применения отчет получили название “Job Digest”.

Ключевые слова: суперкомпьютер, производительность, исследование эффективности, мониторинг, параллельные вычисления, динамические характеристики задач, высокопроизводительные вычисления.

With the scale of supercomputing systems and applications growing fast, the difficulty of developing performance efficient applications also grows rapidly. The reason for this is an extensive number of factors that potentially influence the application performance. Hardware and software specifics of the supercomputer, peculiarities of the application, interference of jobs running simultaneously — everything needs to be taken into account when trying to achieve high performance. With supercomputers constantly evolving, all these specifics become more and more complicated. HPC clusters with the highest performance contain millions of cores. How is it essentially possible to organize efficient computing on such a scale? The memory hierarchy gets more complicated every year. How this memory can be utilized efficiently?

Application performance is not the only aspect to optimize in HPC. It is also very important to learn how to measure and enhance the efficiency of supercomputer utilization, i.e., how efficiently are CPU time and other resources of the supercomputer utilized. Our experience indicates that even problems with aspects like job scheduling or managing quotas can often lead to performance degradation.

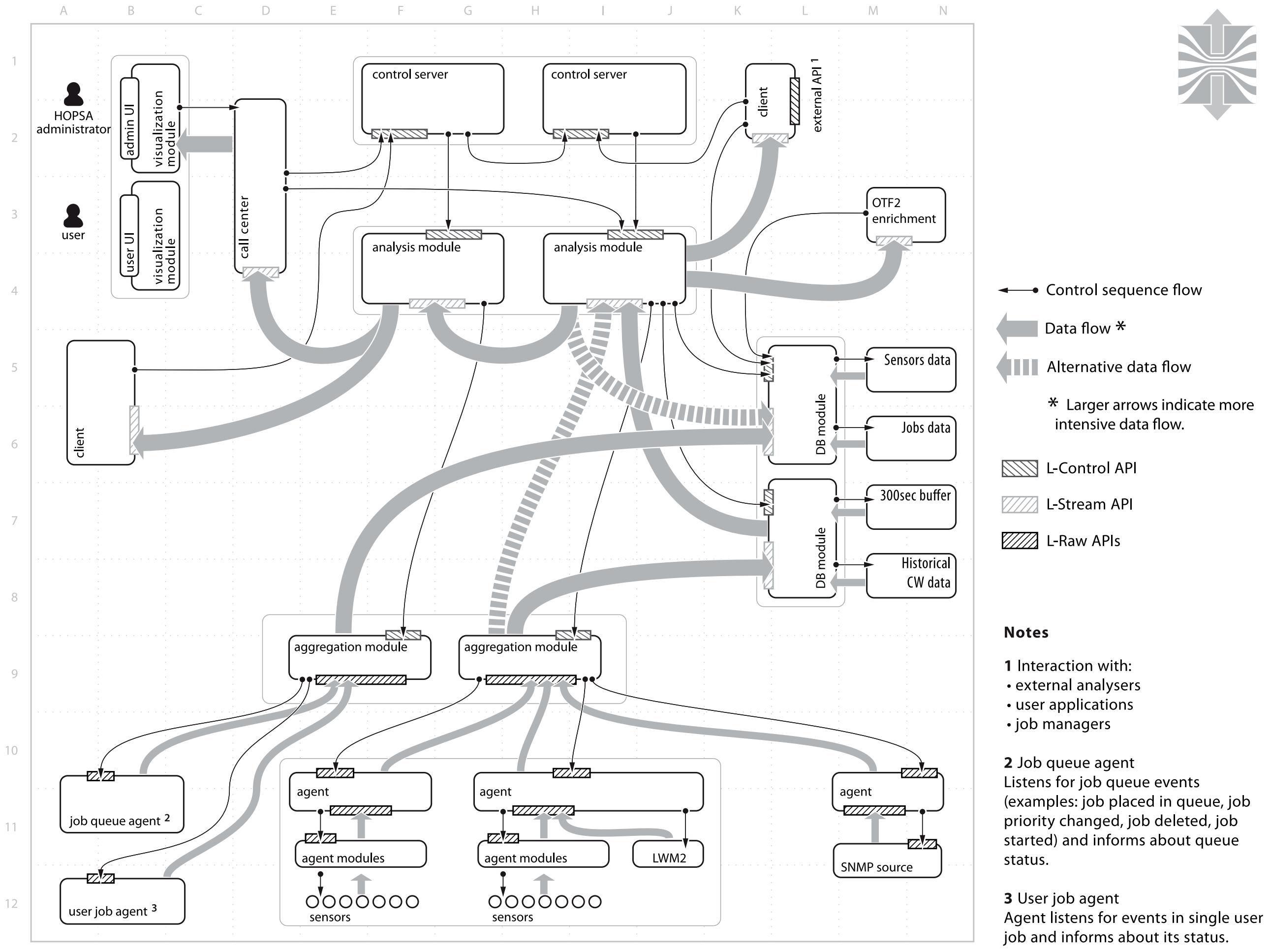
All the reasons mentioned above indicate the demand for a specific tool that would allow seeing where and, what is more important, why does the performance loss happen. It is absolutely clear that such tool should consider a variety of special data on job behavior and supercomputer status to provide high-quality and versatile estimates. The tool of this type is being developed in the Laboratory of Parallel Information Technologies of Research Computing Center of Lomonosov Moscow State University. It was named LAPTA, which stands for “Lapta is a pAckage for Performance moniToring and Analysis”. This tool is being developed in the framework of joint Russian-European HOPSA project, which stands for “HOlistic Performance System Analysis” [1, 2].

In this paper we discuss the LAPTA system and discuss in detail one of the approaches aimed at studying the application behavior during the job run. This approach studies the dynamic characteristics of jobs that are gathered by monitoring tools. Its aim is to provide system administrators and users with overall job characteristics in order to get both overall and detailed analysis of every separate job run. This approach and the generated detailed report have been named “Job Digest”.

[1] Научно-исследовательский вычислительный центр, Московский государственный университет им. М.В. Ломоносова, Ленинские горы, д. 1., стр. 4, 119992, Москва; А.В. Адинец, науч. сотр., e-mail: adinetz@gmail.com; П. А. Брызгалов, ст. науч. сотр., e-mail: pyotr777@guru.ru; Вад. В. Воеводин, науч. сотр., e-mail: vadim@parallel.ru; С. А. Жуматий, вед. науч. сотр., e-mail: serg@parallel.ru; Д. А. Никитенко, науч. сотр., e-mail: dan@parallel.ru; К. С. Стефанов, ст. науч. сотр., e-mail: cstef@parallel.ru

1. Architecture of LAPTA system. The LAPTA system is designed for the versatile analysis of supercomputer parallel program dynamic characteristics. The characteristics registered from submitting a job to its termination are taken into consideration during such an analysis. This allows obtaining full information both on a single job and on all simultaneous jobs running on the system.

The general scheme of the LAPTA system is given in Fig. 1. Data from various hardware sensors are collected by agents and are transmitted to the aggregation level on cluster nodes. Agents collect data from the Resource Manager as well. Aggregation modules save the collected data into the databases using DB modules. Every DB module supports one DB type. Later, the data can be accessed for further analysis by the corresponding analyzing modules.



Notes

1 Interaction with:
- external analysers
- user applications
- job managers

2 Job queue agent
Listens for job queue events (examples: job placed in queue, job priority changed, job deleted, job started) and informs about queue status.

3 User job agent
Agent listens for events in single user job and informs about its status.

Fig. 1. Architecture of LAPTA system

The purpose of the analyzing modules is data processing. An analysis procedure can be initiated from different components of the system. For example, an analysis request can be received from a visualization module and from the so-called "call center" (a request managing module) as a result of user web-browser actions in the interface of the system. In this case the major task of the request is usually to perform the analysis of dynamic behavior of a parallel application. The analysis procedure can be initiated by a system process, for example, launching the report generation on supercomputer daily activity by timer once a day. The data can be requested by external integration and visualization systems as well. In any case, the request generator contacts at first the control server that launches special analyzing modules if needed and controls them. Data are not transmitted through a control server since data are returned directly by analyzing modules to the request initiator.

2. Job Digest approach. General ideas and implementation basis of the Job Digest approach as a part of the LAPTA system will be described in this section.

When the job is being submitted, an agent that will look for job status changes starts its operation. Optionally, the identifier of the job can be passed to the agent if it already runs. This agent finds the mark in the Resource Manager output stream that identifies the job termination and provides basic information (node list, time range, etc.) on this job run for a further detailed Job Digest report generation.

The Resource Manager starts the job and puts all job status changes to its output. Currently, CLEO and SLURM Resource Managers are supported together with the Joint Supercomputer Center Resource Manager support being currently implemented. At the same time, the monitoring data are being continuously collected. Further, the collected data, in some cases being aggregated, are saved for a further a posteriori analysis [3]. In addition to the main set of monitoring sensors, data from other sources can be collected at the same time, e.g. traces from the application level collected by software such as LWM2. Interfaces are implemented for such integration with systems of data collection and data storage.

This approach assumes that the infrastructure for accessing stored monitoring data is provided. Such an infrastructure is developed by the Russian side in the framework of the above-mentioned joint HOPSA project. For example, Hoplang language [4, 5] is designed for forming and managing the monitoring system data requests.

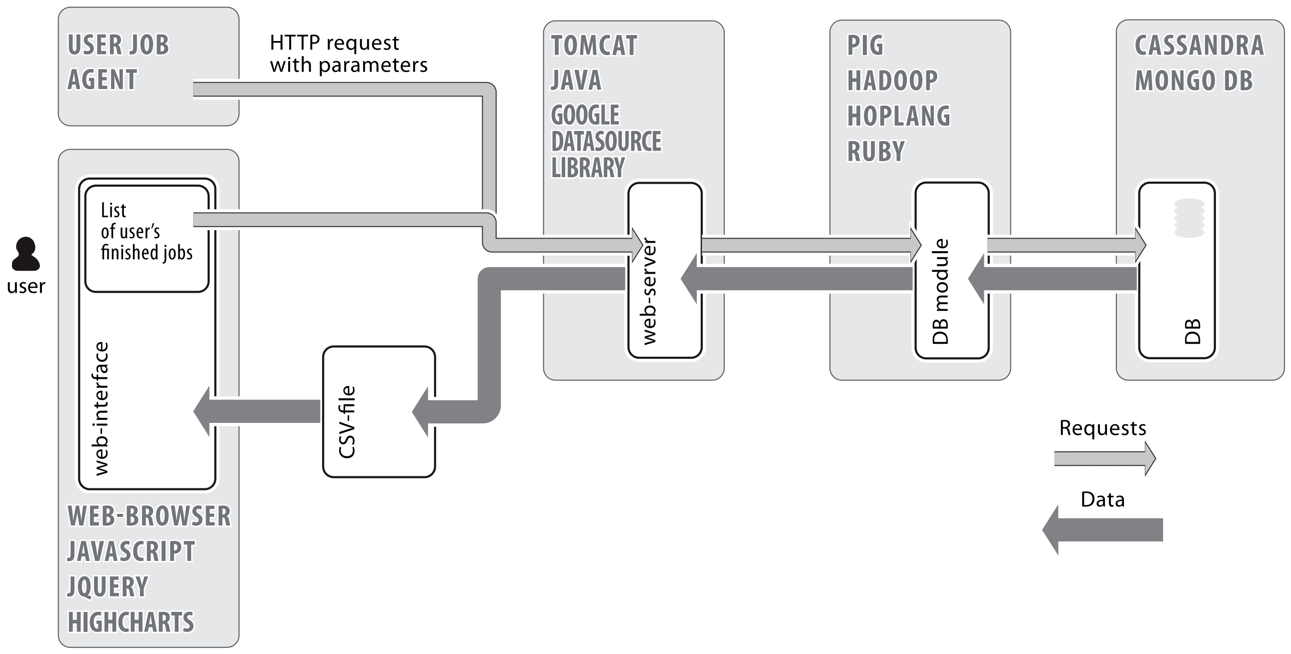


Fig. 2. Overall workflow of report generation

Figure 2 illustrates the overall workflow of report generation. Cassandra and Mongo databases are used for data storage. The modules for database access are implemented in two versions: based on Pig and Hadoop technologies and based on Hoplang technology. Standard data access modules for Pig and Hadoop are used [6, 7], while data access modules for Hoplang are implemented in Ruby. The first version [8] uses the Pig latin language for the description of data requests, whereas the second one uses Hoplang.

3. Generation of Job Digest report. The Job Digest report is currently generated when the user requests the generation of such a report manually via the web user interface. A script that generates a request to a Java servlet can be set up to run on any specific event — typically, for every (or specified) job when it finishes its operation.

The request contains information identifying the job and the report generation template to be used. If the report has already been generated (this can be seen by a Java servlet positive response), the report information is put to an HTML file that will be accessible via a web browser. The address of this report is stored in a separate file together with the brief job run information that was provided by Resource Manager, so the user could access this report later.

The web server is implemented on Java language as a set of servlets running under Apache Tomcat. The web server interacts with database module by RPC calls using AVRO protocol. The database module receives the requests from web server and returns data in CSV format.

The web server performs the additional processing of the received data using Google Datasource. This allows using Google query language for post processing the data received from the database module. This library is also used for formatting data before writing in CSV files. Data visualization is performed in a web browser using JavaScript together with jQuery and Highcharts JavaScript libraries.

A Java servlet executes a number of requests with job parameters and the resulting data are written in the CSV files. The set of the requests and the set of parameter-value pairs are located in a template that is

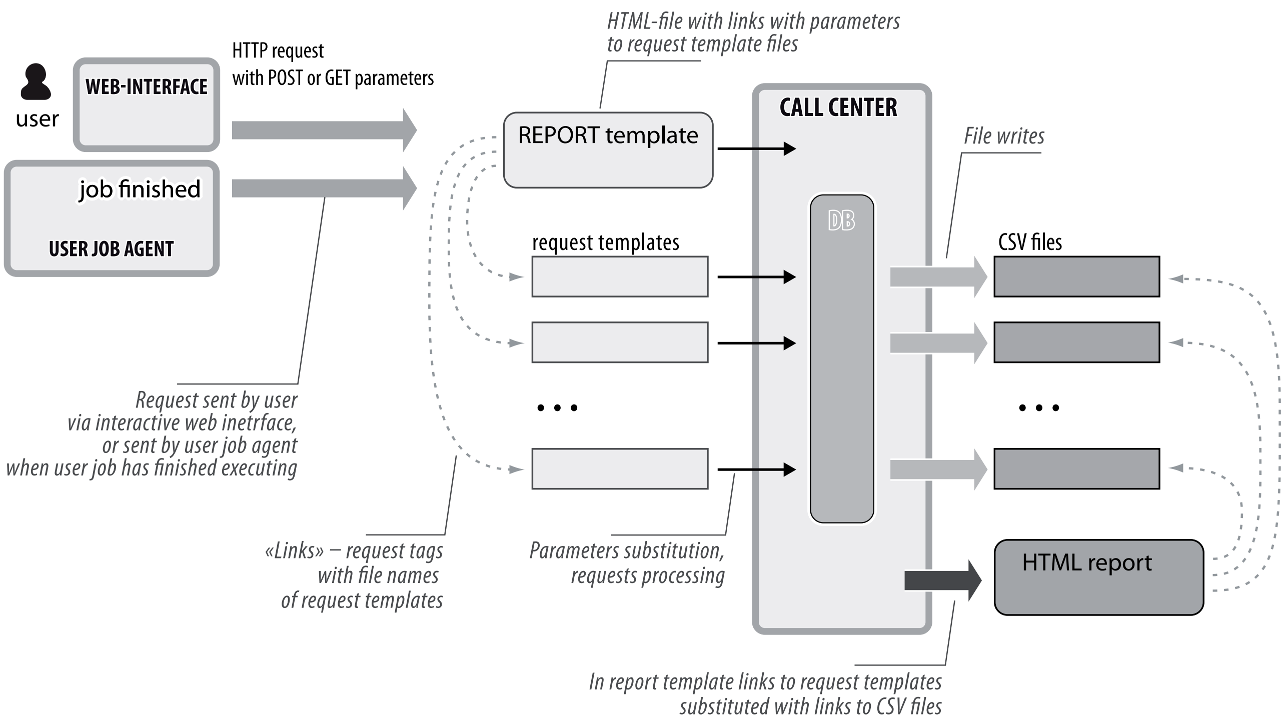
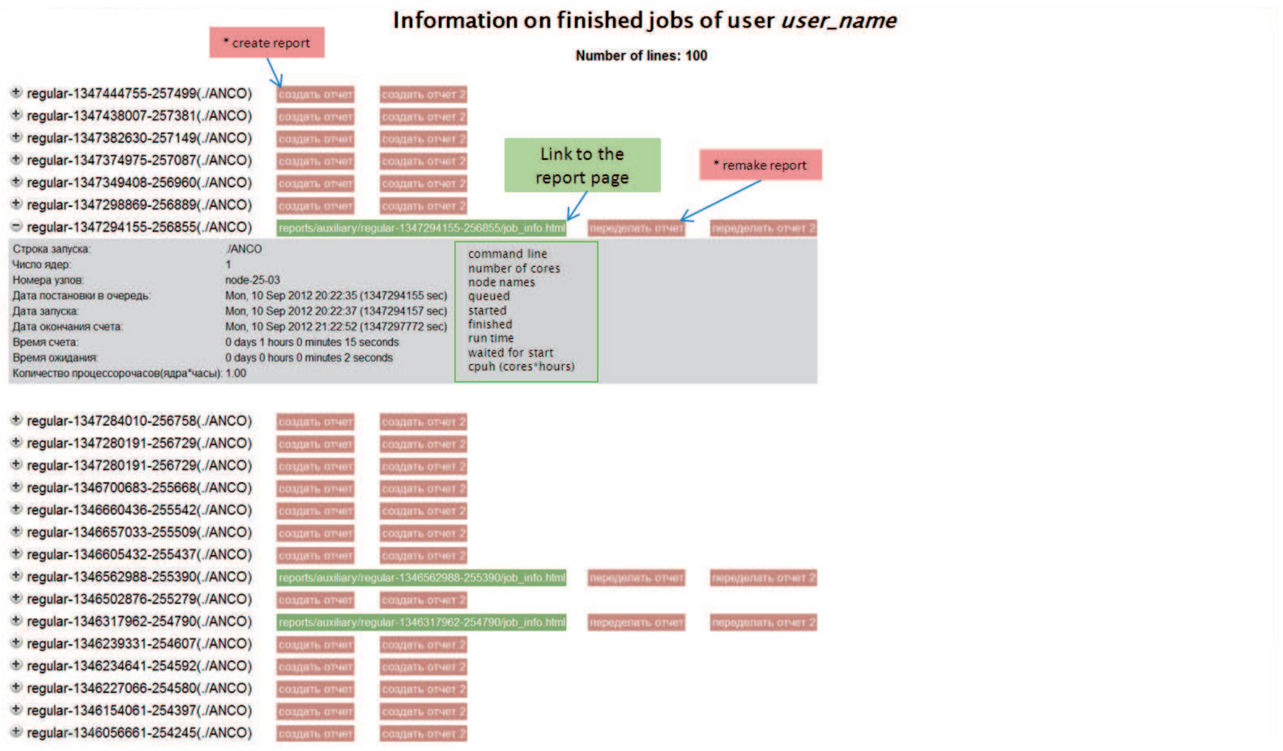

user
WEB-INTERFACE
job finished
USER JOB AGENT
HTTP request
with POST or GET parameters
HTML-file with links with parameters
to request template files
REPORT template
CALL CENTER
DB
request templates
File writes
CSV files
...
...
HTML report
Request sent by user
via interactive web inetrface,
or sent by user job agent
when user job has finished executing
«Links» – request tags
with file names
of request templates
Parameters substitution,
requests processing
In report template links to request templates
substituted with links to CSV files

Fig. 3. Operation sequence for Job Digest report generation

actually an HTML file with links to these requests. The requests are stored in text files, the so-called "request templates". The requests are executed one by one according to their order in the HTML template. As soon as requests finished its execution, the report is generated. It is created from the HTML template by replacing the links to separate requests with special links to the CSV files with request results. These links are implemented in a way to provide diagrams and tables with request results from the appropriate CSV files when the report page is opened in the web browser.

The overall operation sequence in the visualization module is presented in Fig. 3.



Information on finished jobs of user *user_name*

Number of lines: 100

+ regular-1347444755-257499(./ANCO)	создать отчет	создать отчет 2		
+ regular-1347438007-257381(./ANCO)	создать отчет	создать отчет 2		
+ regular-1347382630-257149(./ANCO)	создать отчет	создать отчет 2		
+ regular-1347374975-257087(./ANCO)	создать отчет	создать отчет 2		
+ regular-1347349408-256960(./ANCO)	создать отчет	создать отчет 2		
+ regular-1347298869-256889(./ANCO)	создать отчет	создать отчет 2		
− regular-1347294155-256855(./ANCO)	reports/auxiliary/regular-1347294155-256855/job_info.html		переделать отчет	переделать отчет 2

Строка запуска:	./ANCO	command line
Число ядер:	1	number of cores
Номера узлов:	node-25-03	node names
Дата постановки в очередь:	Mon, 10 Sep 2012 20:22:35 (1347294155 sec)	queued
Дата запуска:	Mon, 10 Sep 2012 20:22:37 (1347294157 sec)	started
Дата окончания счета:	Mon, 10 Sep 2012 21:22:52 (1347297772 sec)	finished
Время счета:	0 days 1 hours 0 minutes 15 seconds	run time
Время ожидания:	0 days 0 hours 0 minutes 2 seconds	waited for start
Количество процессорочасов(ядра*часы):	1.00	cpuh (cores*hours)

+ regular-1347284010-256758(./ANCO)	создать отчет	создать отчет 2		
+ regular-1347280191-256729(./ANCO)	создать отчет	создать отчет 2		
+ regular-1347280191-256729(./ANCO)	создать отчет	создать отчет 2		
+ regular-1346700683-255668(./ANCO)	создать отчет	создать отчет 2		
+ regular-1346660436-255542(./ANCO)	создать отчет	создать отчет 2		
+ regular-1346657033-255509(./ANCO)	создать отчет	создать отчет 2		
+ regular-1346605432-255437(./ANCO)	создать отчет	создать отчет 2		
+ regular-1346562988-255390(./ANCO)	reports/auxiliary/regular-1346562988-255390/job_info.html		переделать отчет	переделать отчет 2
+ regular-1346502876-255279(./ANCO)	создать отчет	создать отчет 2		
+ regular-1346317962-254790(./ANCO)	reports/auxiliary/regular-1346317962-254790/job_info.html		переделать отчет	переделать отчет 2
+ regular-1346239331-254607(./ANCO)	создать отчет	создать отчет 2		
+ regular-1346234641-254592(./ANCO)	создать отчет	создать отчет 2		
+ regular-1346227066-254580(./ANCO)	создать отчет	создать отчет 2		
+ regular-1346154061-254397(./ANCO)	создать отчет	создать отчет 2		
+ regular-1346056661-254245(./ANCO)	создать отчет	создать отчет 2		

Fig. 4. Web page with information on finished jobs for a specific user

All the dynamic user application run characteristics are available in full detail to system administrators together with some special monitoring tools [9]. For the non-administrator users, access is granted to the results of their own jobs only. The users are provided with a page, as demonstrated in Fig. 4, which contains a list of their finished jobs on the supercomputer. The link to generate a report on a job is given next to every job in the list if the report is not generated yet. A link to the report is given if this report was generated previously.

When the report is ready, the user can open it in a web browser. The report is based on the information extracted from system monitoring data. Any number of graphs and diagrams that illustrate various specifics on the job can be included in the report. Our examples include CPU usage (Fig. 5), Flop/s (Fig. 6), L2 cache misses (Fig. 7), Ethernet speed (Fig. 8), Infiniband speed (Fig. 9), and Load Average (Fig. 10).

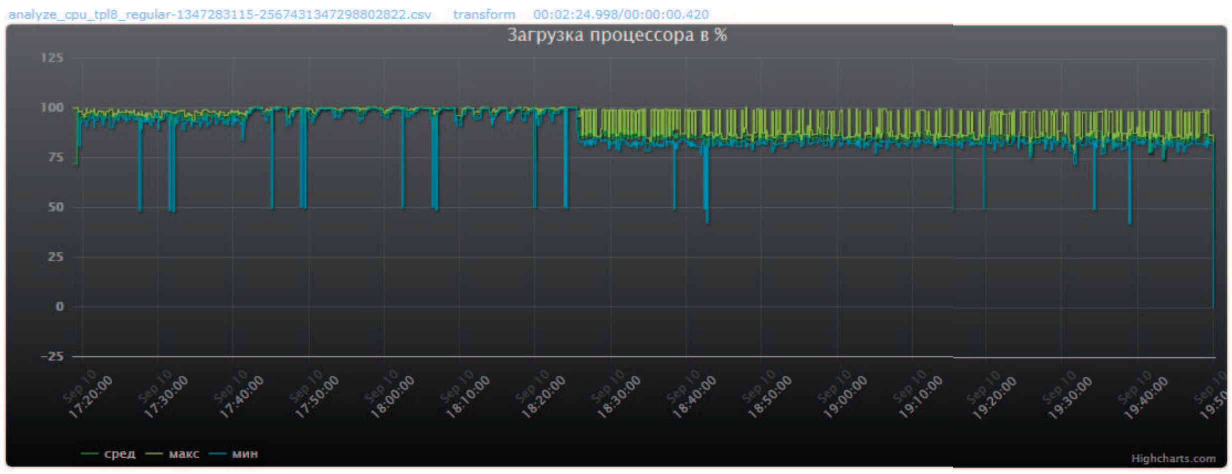


Fig. 5. CPU usage in percent

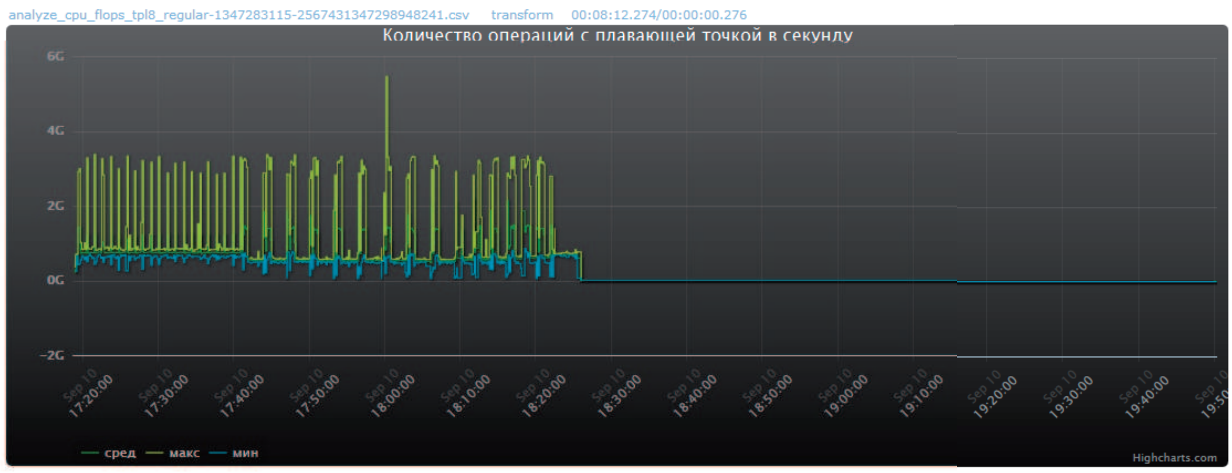


Fig. 6. Floating point operations per second (Flop/s)

These graphs allow making meaningful conclusions on program behavior. For example, while the load average illustrates that the working load of cores was normal (Fig. 10), CPU usage, flop/s and cache misses counters (Fig. 5–7) show that the CPU usage for floating point computing was almost zero from the middle of the job run. At the same time, Ethernet (Fig. 8) and Infiniband (Fig. 9) speeds prove that the first part was computational (Infiniband interconnect for node interaction) and almost all the second part of that job run was spent in output. This can show to the user that a significant job time is dedicated to output and can be an issue for an optimization of the appropriate part of the program.

4. Conclusion. A useful and effective tool for the quality analysis of every job run without the necessity of special program preparation before execution is developed and is currently being tested. This allows the users to significantly enhance the understanding of how their applications really behave. For example, it allows finding abnormal situations that are caused by specifics of a parallel program or localizing the interference with other programs. In the latter case, the administrator potentially can lower the negative effect by performing a resource management optimization or tune scheduling policy or by fixing some issues with general system software setup.

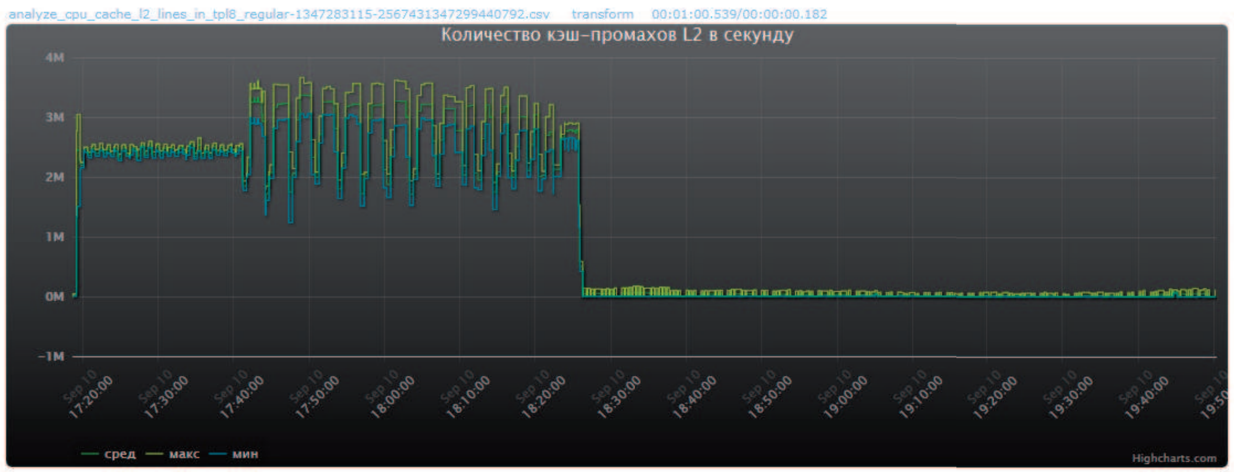


Fig. 7. Level2 cache misses per second

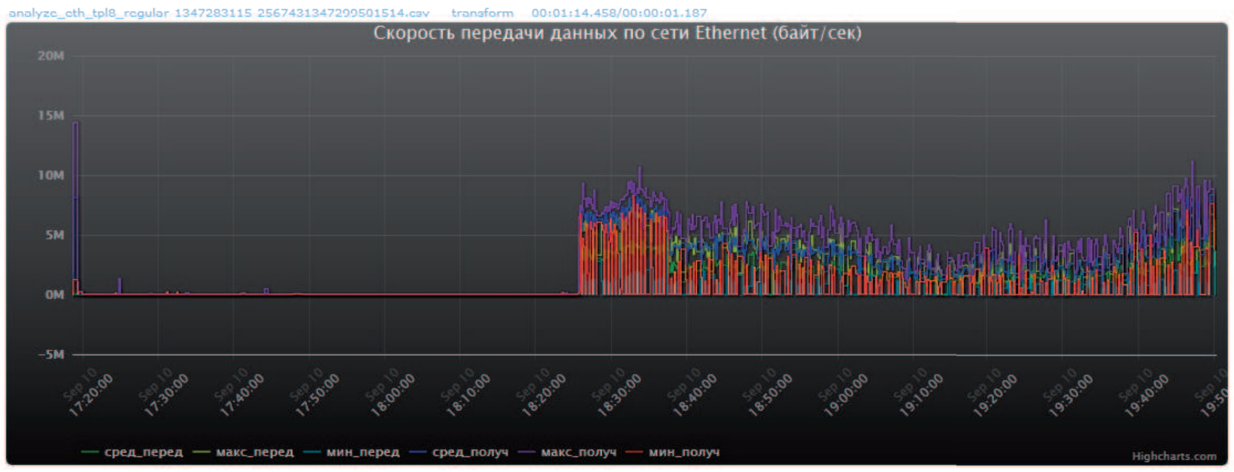


Fig. 8. Ethernet sent bytes per second

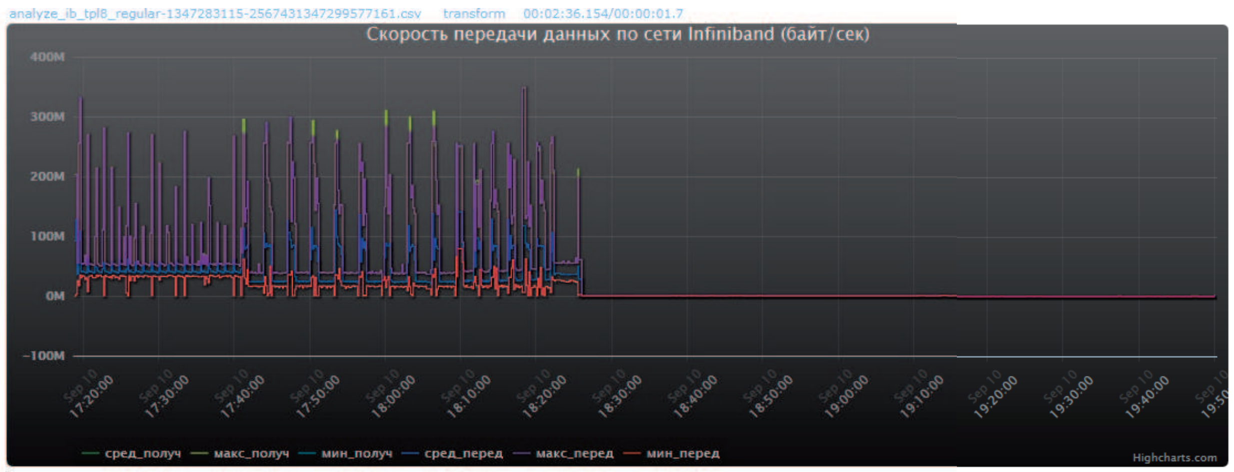


Fig. 9. Infiniband sent bytes per second

This approach will be definitely developed further not only by extending its current functionality but also by enhancing the quality of the information provided. This includes the idea of automatic search for potential bottlenecks in the a given job run that appeared with the very first ideas on this approach. Such a functionality would allow directing the attention of users to potential problems in their application even in automatic mode and, in some cases, even suggesting the tools for further investigation and ways of eleminating the difficulties if they exist. Automatic processing of system monitoring data and providing recommendations are very demanded

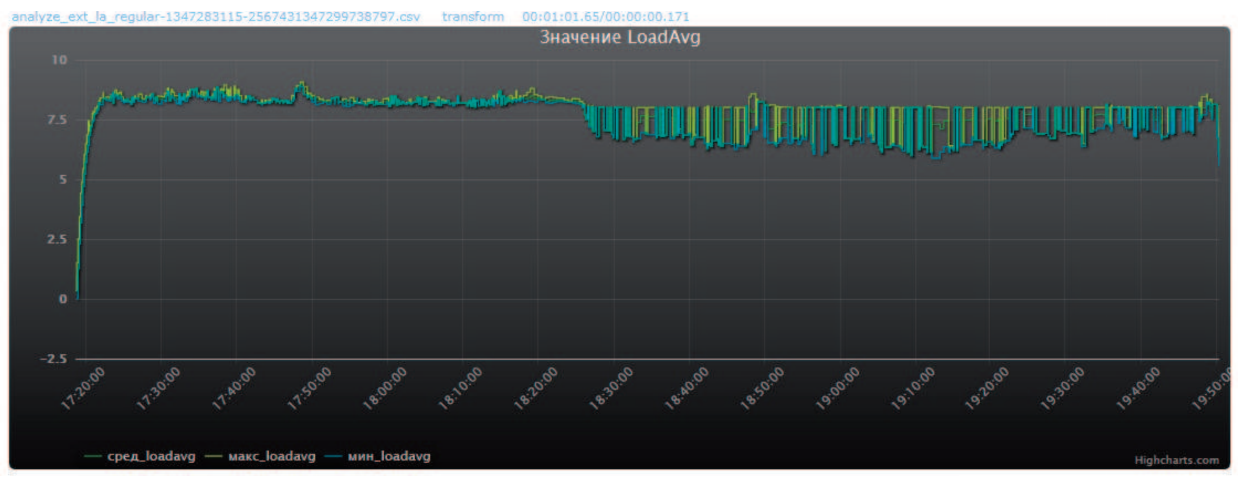


Fig. 10. Load Average

and important for supercomputer users that can be not so well acquainted with peculiarities of job execution on a particular supercomputer system, thus missing even typical symptoms of inefficiency study of their application run history. Special interfaces for integration with external applications for further investigation are available in the developed HOPSA system. For example, this can be made through the OTF2 trace file enriched with system monitoring data, providing information for analysis with external analyzers like Vampir or Scalasca.

The testing of LAPTA is currently being held using supercomputing facilities of Lomonosov Moscow State University.

5. Acknowledgements. This work was supported by the Ministry of Education and Science of the Russian Federation (contract 07.514.12.4001).

СПИСОК ЛИТЕРАТУРЫ

1. Methods and instrumental systems development for analysis of effectiveness of parallel programs and supercomputers (official site of RU-part of HOPSA project) (http://hopsa.parallel.ru).
2. HOlistic Performance System Analysis (EU HOPSA website) (http://vi-hps.org/projects/hopsa/).
3. *Никитенко Д.А., Стефанов К.С.* Исследование эффективности параллельных программ по данным мониторинга // Вычислительные методы и программирование. 2012. **13**. 97–102.
4. *Адинец А.В., Жуматий С.А., Никитенко Д.А.* Hoplang — язык обработки потоков данных мониторинга // Тр. Междунар. науч. конф. “Параллельные вычислительные технологии” (ПаВТ) 2012 (Новосибирск, март 26–30, 2012). Челябинск: ЮУрГУ, 2012. 351–359.
5. Hoplang language for data processing of cluster monitoring systems (http://github.com/zhum/hoplang).
6. Hadoop project homepage URL: http://hadoop.apache.org/.
7. Apache PIG Latin official cite URL: http://pig.apache.org/.
8. *Адинец А.В., Брызгалов П.А., Воеводин Вад.В., Жуматий С.А., Никитенко Д.А.* Об одном подходе к мониторингу, анализу и визуализации потока заданий на кластерной системе // Вычислительные методы и программирование. 2011. **12**. 90–93.
9. *Адинец А.В., Брызгалов П.А., Жуматий С.А., Никитенко Д.А.* Система визуализации параметров работы больших вычислительных систем // Тр. Междунар. науч. конф. “Параллельные вычислительные технологии” (ПаВТ) 2012 (Новосибирск, март 26–30, 2012). Челябинск: ЮУрГУ, 2012. 714.

Поступила в редакцию
27.11.2012