DOI: 10.14529/cmse160403

SUPERCOMPUTER APPLICATION INTEGRAL CHARACTERISTICS ANALYSIS FOR THE WHOLE QUEUED JOB COLLECTION OF LARGE-SCALE HPC SYSTEMS*

© 2016 D.A. Nikitenko, V.V. Voevodin, A.M. Teplov, S.A. Zhumatiy, Vad.V. Voevodin, K.S. Stefanov, P.A. Shvets

Research Computing Center, M.V. Lomonosov Moscow State University (Leninskie Gory 1, Moscow, 119991 Russia)

E-mail: dan@parallel.ru, alex-teplov@yandex.ru, serg@parallel.ru, vadim@parallel.ru, cstef@parallel.ru, shvets.pavel.srcc@gmail.com

Received: 11.04.2016

Efficient use and high output of any supercomputer depends on a great number of factors. The problem of controlling granted resource utilization is one of those, and becomes especially noticeable in conditions of concurrent work of many user projects. It is important to provide users with detailed information on peculiarities of their executed jobs. At the same time it is important to provide project managers with detailed information on resource utilization by project members by giving access to the detailed job analysis. Unfortunately, such information is rarely available. This gap should be eliminated with our proposed approach to supercomputer application integral characteristics analysis for the whole queued job collection of large-scale HPC systems based on system monitoring data management and study, building integral job characteristics, revealing job categories and single job run peculiarities.

Keywords: supercomputer, efficiency, system monitoring, job categories, integral job characteristics, queued job collection, job queue, resource utilization control.

FOR CITATION

Nikitenko D.A., Voevodin V.V., Teplov A.M., Zhumatiy S.A., Voevodin Vad.V., Stefanov K.S., Shvets P.A. Supercomputer Application Integral Characteristics Analysis for the Whole Queued Job Collection of Large-Scale HPC Systems. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2016. vol. 5, no. 4. pp. 32–45. DOI: 10.14529/cmse160403.

Introduction

Securing efficient resource utilization of HPC systems is one of the most important and challenging tasks at present trends of rapid growth of scales and capabilities of modern supercomputers [1, 2]. There is a variety of approaches that are aimed at analysis of efficient utilization of certain HPC system components or systems as a whole. Some of them are based on system monitoring data analysis [3, 4]. This type of approaches sets especially strict requirements on monitoring system implementation and configuration [5], as well as for the means of data storage and access. At the same time these approaches possess a number of fundamental advantages.

*The paper was recommended for publication by the program committee of the International Scientific Conference "Parallel Computing Technologies – 2016".

First, the analyzed data reflects physical, real levels of HPC system components and appropriate resource utilization.

Second, filtering system monitoring data obtained from known set of components and period of time allows binding this data to certain jobs. Thus, allowing analyzing resource utilization history and trends by certain applications, users, projects, partitions, and so on.

Third, typically it is possible to configure monitoring systems obtaining data from the whole system in such a way that it induces acceptable overhead. This allows collecting data with a rougher granulation, when possible, but still sufficient for basic analysis of resource utilization by any and every job. To have more detailed information on certain job, of course, used monitoring system should likely support data acquisition rate reconfiguration on-the-fly for specified sensor sets and sources. If not (of course, it is much less efficient way), most monitoring systems can be started in a higher granularity mode to record certain job activity and restarted in a normal mode afterwards. There are other options available, for example, data aggregation implementation that is precise for first, say, 30 seconds of job execution (to study short jobs) that is later switches to rough mode (for longer jobs). Anyhow, there are a number of techniques available to study certain application behavior.

The existing methods and techniques that base on system monitoring data analysis, allow both analysis of dynamic characteristics of certain application runs and peculiarities of resource utilization within system partitions and systems as a whole [6]. With a project-oriented workflow, when a number of users run jobs as a part of one applied research, it is very useful to let administrator and system manager have a clear view of resource utilization distribution in the workgroup to have a possibility to influence permissions or workflow inside the workgroup to meet the granted resources limitation [7, 8]. Nevertheless there is still need for specialized tools and techniques to analyze available system monitoring data. In a point of fact, the one is needed as a valuable additional tool to the set of implemented approaches in every-day practice of MSU Supercomputer Center [9–12] — a tool for job queue analysis based on system monitoring that would allow revealing job categories, job grouping by some criteria, starting from belonging to user or project domain and other resource manager specific characteristics, to categories by levels and peculiarities of HPC system resource utilization or its combinations. As a basic technique for such grouping implementing tagging system seems to be an adequate option — assigning special tags to a job description as soon as each tag description criteria is met by job characteristics. Tagging principles are widely successfully used for categorizing and search purposes managing huge collections of data in Internet: news, videos, photos, notes, and so forth, that is quite close to the challenge that is being tackled.

The paper is organized as follows. Section 1 is devoted to job categories and tagging principles. Section 2 describes implementation. Section 3 provides examples and use cases. Conclusion section includes summary as well as future work overview.

1. Job categories and tagging

The combined analysis of system monitoring data and resource manager log data, as was already mentioned, allows binding raw system monitoring data to certain jobs. This provides means to analyze job dynamics as far as data granularity allows. To analyze the average rate of application resource utilization every dynamic characteristic can serve basis for the calculation of minimum, maximum, average and median values. These types of values are often named integral job characteristics.

When one takes a look at the whole scope of executed jobs for analysis of job queue structure, application run sequences, jobs comparative analysis and even searching for outstanding single job behavior it becomes obvious that it would be very useful to have tools that provide means for revealing job categories based on various criteria.

This functionality can be implemented by introducing special tags. Every tag is based on its own criteria, based on a single integral job characteristic or its combination, resource manager job-related information and any other available info from used data sources. For example tags can correspond to certain average rates of various resource utilization, job ownership, job duration, resource utilization specifics, special execution modes, detailed system monitoring data availability, and so forth.

The approach features means to make efficient grouping and filtration of whole job queue history collection by any improvised combination of specified tags. Driven by experience of application efficiency and scalability study based on system monitoring data analysis, the authors propose introducing the following job categories on the first stage of implementation. Tag naming is designed to give self-explanatory tag description, nevertheless, every tag must have a detailed full-format description available.

1.1. System monitoring data based categories

CPU utilization

- Tag name: avg_CPU_user LOW
  Category: Low CPU user utilization.
  Criteria: Average value of CPU_user doesn't exceed 20%.
- Tag name: avg_CPU_user HIGH
  Category: High CPU user utilization.
  Criteria: Average value of CPU_user exceeds 40%.
- Tag name: avg_CPU_idle TOO HIGH
  Category: CPU is idle for a considerable time.
  Criteria: Average CPU_idle value exceeds 25%.

Competition of processes for CPU cores

- Tag name: avg_LA LOW
  Category: User job is almost out of action, almost no utilization of CPU.
  Criteria: Average Load Average is below 1.
- Tag name: avg_LA SINGLE CORE
  Category: Only one process per node is active as an average.
  Criteria: Average Load Average is approximately 1.
- Tag name: avg_LA NORMAL
  Category: Optimal competition of processes.
  Criteria: Average Load Average is approximately equal to the number of cores per node.
- Tag name: avg_LA HYPERTHREADED
  Category: Normal process competition with hyperthreading is on.
  Criteria: Average Load Average value is approximately equal to the double number of CPU cores per node.

Floating point operations

- Tag name: avg_Flops HIGH
  Category: Intensive CPU floating point operations.

Criteria: Average value of floating point operations number exceeds 10% of theoretical CPU peak.

Interconnect activity

- Tag name: avg_IB_packages_num LOW
  Category: Low number of inter-node data transmissions.
  Criteria: Average package send rate does not exceed 103 packages per second.
- Tag name: avg_IB_packages_size TOO LOW
  Category: Small size of packages.
  Criteria: Average package send rate exceeds 103 packages per second while average data transmission rate is below 2 kilobytes per second.
- Tag name: avg_IB_speed HIGH
  Category: High data transmission intensity.
  Criteria: Average data transmission rate is over 0,2 Gigabytes per second and up to 1 Gigabytes per second.
- Tag name: avg_IB_speed TOO HIGH
  Category: Very high data transmission intensity.
  Criteria: Average data transmission rate is over 1 Gigabytes per second.

Memory utilization

- Tag name: avg_cache_L1/L3 TOO LOW
  Category: Very low efficiency of cache stack utilization.
  Criteria: Ratio of the number of L1 misses to the number of L3 misses is below 5.
- Tag name: avg_cache_L1/L3 LOW
  Category: Reduced efficiency of cache stack utilization.
  Criteria: Ratio of the number of L1 misses to the number of L3 misses is below 10.
- Tag name: avg_cache_L1/L3 HIGH
  Category: Good efficiency of cache stack utilization.
  Criteria: Ratio of the number of L1 misses to the number of L3 misses exceeds 10.
- Tag name: avg_mem/cache_L1 LOW
  Category: Reduced efficiency of cache L1 utilization.
  Criteria: Ratio of the number of total memory operations to the number of L1 misses does not exceed 15.
- Tag name: avg_memload HIGH
  Category: Intensive memory operations.
  Criteria: Average number of memory operations exceeds 109 operations per second.

1.2. Resource manager based categories

Job execution status

- Tag name: job_status COMPLETED
  Category: Job is successfully finished.
- Tag name: job_status FAILED
  Category: Job is finished with an error in program.
- Tag name: job_status CANCELED
  Category: Job was cancelled by user.
- Tag name: job_status TIMEOUT
  Category: Job was cancelled by exceeding time limit.

- Tag name: job_status NODE_FAIL
  Category: Job is finished with system error.

Job submission details

- Tag name: job_time_limit CUSTOM
  Category: Requested time limit is custom.
- Tag name: job_start_script CUSTOM
  Category: Job batch file is custom.
- Tag name: job_cores_requested FEW
  Category: Not all available CPU cores per node requested.
- Tag name: job_cores_requested SINGLE
  Category: Just a single CPU core per node requested.
- Tag name: job_MPI INTEL
  Category: MPI type used: Intel MPI.
- Tag name: job_MPI OpenMPI
  Category: MPI type used: OpenMPI.
- Tag name: job_nnodes SINGLE
  Category: Job used a single node.
- Tag name: job_nnodes FEW
  Category: Job used from 2 up to 8 nodes.
- Tag name: job_nnodes MANY
  Category: Job used 8 nodes and above.

System-dependent peculiarities and partition usage
Illustrated by the example of "Lomonosov" supercomputer partitions.

- Tag name: job_partition REGULAR4
  Category: Job allocated to REGULAR4 partition.
- Tag name: job_partition REGULAR6
  Category: Job allocated to REGULAR6 partition.
- Tag name: job_partition HDD4
  Category: Job allocated to HDD4 partition.
- Tag name: job_partition HDD6
  Category: Job allocated to HDD6 partition.
- Tag name: job_partition SMP
  Category: Job allocated to SMP partition.
- Tag name: job_partition GPU
  Category: Job allocated to GPU partition.
- Tag name: job_partition TEST
  Category: Job allocated to TEST partition.
- Tag name: job_partition GPUTEST
  Category: Job allocated to GPUTEST partition.
- Tag name: job_partition EXCEPT TEST
  Category: Job allocated to regular or high priority partition.
- Tag name: job_priority HIGH
  Category: Job allocated to partitions with a higher priority (queues reg4prio, gpu_p, dedicated6)

Matching partition specifics

- Tag name: job_accell GPU
  Category: User application uses accelerators. Accelerator type: GPU.
- Tag name: job_accel GPU UNUSED
  Category: Job is run on GPU partition, but never uses GPUs.
- Tag name: job_disks UNUSED
  Category: Job is run on HDD-equipped partition, but never uses local I/O.
- Tag name: job_disks TOO LOW
  Category: Job is run on HDD-equipped partition, but I/O rate is very low.

1.3. Other categories

Beyond the tags that can be assigned automatically, it is possible to introduce manually-set tags. This is useful when the criteria is cannot be automatically determined. There are now different manual setting options available. First, most typical, selecting the one from known tags lost or introducing new one with human-read and formal description. Second, is pushing some tags like "higher system monitoring rate for the job" via the command line when submitting a job.

This applies for instance to general job description characterizing type of data processing as it is usually known a priori or determined in the course of job behavior study by a specialist: job_behavior DATA MINING, job_behavior MASTER-SLAVE, job_behavior COMMUNICATION, job_behavior ITERATIVE, etc.

In the same manner typical anomalies encountered during analysis course can be specified: job_bug DEADLOCK, job_bug DATA RACE, etc.

It is very useful to specify if a widely used algorithm implementation or software package is used. Just in case, this can provide a great contribution to scalability and algorithms-studying projects, like AlgoWiki [13]: job_sw VASP, job_sw FIREFLY, job_sw GROMACS, etc.

If detailed reports on job efficiency analysis or issues is available, or specific standard report like JobDigest is available, it is useful to mark such a feature with another tag, for example: job_analized, job_analized JobDigest.

2. Implementation

We keep to the basis of building a tool that might be deployed at any supercomputer center with minimal efforts. We currently support Slurm [14], Cleo [15] resource managers and Ganglia [16], Collectd [17], Clustrx [18], DiMMon [5] (most promising) monitoring systems.

As for integral job characteristics derivation and tagging, PostgreSQL is used as data storage for coarsened system monitoring data and saved job information from resource managers. The saved job info is processed by JavaScript, jQuery with jQuery UI [19] and TagIt [20].

The tag can be assigned to a job only if it is already declared in tag description table. Such a table includes tag id, name, human-readable description, criteria (a specification of SQL request for automatic processing), comments, and a flag of availability that can be set only by administrator. Any user can suggest introducing a new tag, but it will be available only after administrator approval. Information on new tag author is saved in the comments attribute, added the user tag description suggestion and motivation.

All tags can be assigned in two ways: automatically and manually. Any tag set by mistake or error can be manually removed from a job.

Automatic mode. In this mode, the tags are automatically assigned:

- to all finished jobs according to SQL-based criteria regarding saved integral job characteristics data, information from resource manager and other available saved data;
- as a result of running a special script that processes whole saved job collection info.

In this mode a special attribute would indicate that the tag was set in package (automatic) mode of tag assignment.

Manual mode. Manual tag assignment is usually done by user, project manager or administrator in the following cases:

- as a result of certain job analysis (specifying algorithm implemented, etc.);
- as a result of specifying the tag via command line when submitting a job;
- any tag in a user-specific tag space (marking out important job runs as a part of the project, etc.).

In this mode an attribute addressing tag author is set, that also allows finding jobs, marked as a part of a certain project or by a certain user.

User-specific tag space consists of regular tags and custom user tags. Any manually assigned tags by a user are seen only in the scope of the project and system administrators. The members of other project see their own tag spaces and the general tag space is available for all of the users.

3. Use cases

Of course, real life use cases are very diverse. It this section we would like to share our experience of every-day usage of the proposed technique as a part of the developed tool approbation at Supercomputer Center of Moscow State University on a few examples just to give a general idea of it.

3.1. Revealing jobs, users and projects that practice inappropriate resource utilization

One of the problems of every-day practice of large-scale supercomputer center with a number of heterogeneous resources and considerable number of users concurring for the resources is a problem of inacceptable efficiency or inappropriate resource utilization. This is of a higher priority for specific limited resources, like compute nodes equipped with specialized accelerators, local disks, extra memory or other hardware and software that is critical for some applications and at the same time these nodes can still be used by applications that do not need that specific type of resources that the nodes possess. Such nodes usually have a high potential for resource-demanding specific applications and for the large systems like “Lomonosov” are usually managed as a separate partition with a special queuing options to allow submitting jobs to the appropriate partition. This is vital for projects that perform computations only due to the advantages of such partitions, so by queuing to the desired partition user get a guarantee that their application would have all necessary resources at disposal.

Nevertheless, when analyzing the whole job collection for such partitions it appears that there are numerous job runs that do not use any partition facilities benefits. Of course, sometimes algorithm peculiarities can use resources with totally different intensity, but further analysis usually shows that the majority of suspicious jobs never use any benefit of such partitions. The reasons can be different, but usually it is a shorter wait time in a queue.

This can be seen on GPU partitions with user job runs that never use GPUs. A slightly different situation is seen on HDD-equipped nodes with absent or extreme low disk usage rate

and finally, single-process application that don't benefit from multiple CPU cores can be seen almost on any partition regardless of hardware and software.

It is important to find the root cause of such applications behavior and as soon as the reason is found and changes by user or administrator are applied, the ratio of such jobs can be lowered that would immediately raise HPC system efficiency and overall throughput.

The most popular reasons are:

- Problems inside the application, program or algorithm. The user is sure that he needs resources, but in practice application doesn't utilize any or utilizes at extremely low rates.
- Problems of HPC system. The declared resources are not available on the nodes.
- Inappropriate job allocation. This can be both a mistake, and cheating for lower job waiting time.

Regardless of real reason, these job runs lead to a higher wait time for the jobs that really need specific resources.

The search for such jobs can be automated using integral job characteristics and some of introduced tags.

For example, to filter the jobs allocated to GPU partition with no usage of accelerator one can use tags job_partition GPU and job_accel GPU UNUSED at the same time. Next, one can cut off jobs allocated to the test partition as of no interest. The rest jobs that are assigned job_status COMPLETED tag probably do not need GPUs at all, as finished successfully with no registered GPU usage. At this point two options are available whether it is a mistake (user or system) or it was done by user intentionally, trying to reduce job wait time as wait time in specific partitions is sometimes less than in regular.

A very similar situation is seen for HDD-equipped nodes. Jobs that are tagged with job_partition HDD4, job_partition HDD6, job_disks UNUSED or job_disks TOO LOW can potentially be successfully executed at regular partitions. Note that there appears an option of very low resource utilization. This means that disk operations might be easily replaced with network file system operations with minimal additional overhead or even without it.

For those jobs that are tagged with job_nodes SINGLE and avg_LA SINGLE CORE or avg_LA LOW, it is quite reasonable to inquire what for it was submitted to the supercomputer. Such jobs use a single node and a single core (or just few processes per node) and can potentially run well on a desktop. Unfortunately such jobs are met very often.

Users who submit types of jobs mentioned above must be contacted to figure out the reasons of the revealed facts of inappropriate and inefficient resource utilization. The problems found should be resolved. If cheating is met or it is proved that the executed jobs do not really need HPC resources, quotas for corresponding user accounts and projects can be reduced to the extent of blocking.

Let us take a look at one of real-life examples. Figure 1 illustrates the filtered job list allocated to regular partitions with automatically avg_LA_SINGLE_CORE tag assigned. It is clearly seen that the jobs have a low LoadAverage close to 1 as filtered by the tag, at the same time having very low CPU_user. Note, that it is not a test partition and all jobs are run on a single node, grabbing 8 cores on regular4 and hdd4 partitions!

A close look at the longest job owner that was cancelled by timeout illustrates that the user always runs such single-node, even single-process jobs regardless of partitions (Figure 2).

Fig. 1. Filtered single-process jobs found in real job queue in various regular partitions

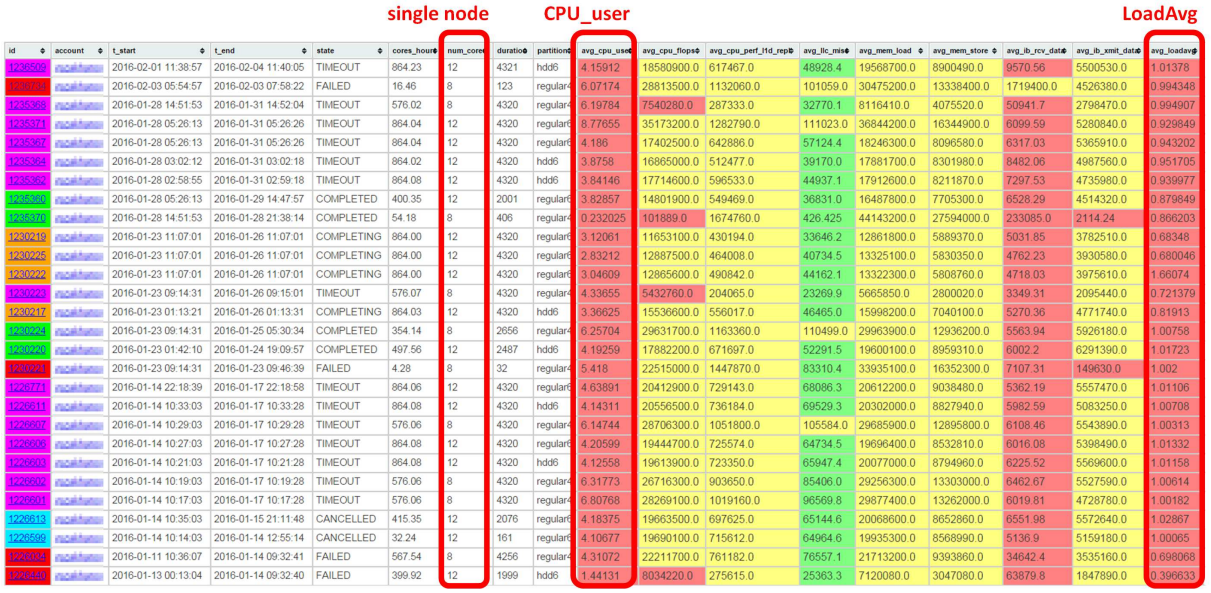

Fig. 2. Filtered single-process jobs found in real job queue in various regular partitions

3.2. Finding jobs with high Flops intensity and high efficiency

Apart from finding problem cases, there is a task of finding well-optimized jobs that utilize HPC resources with high efficiency. These jobs owners are usually experienced users with high qualification in parallel programming and fine tuning of software that secures highly efficient supercomputer load. The contact with such users is very important first and foremost to learn the techniques used and share them with novice users, contributing to FAQ/wiki sections of helpdesk and so on. The experienced users are very likely to be invited to give public lectures, make reports on seminars and join other educational activities.

As a rule, most of jobs with high floating point intensity are tagged with avg_Flops HIGH, avg_CPU_user HIGH tags and most efficient apps in terms of memory utilization are tagged

avg_cache_L1/L3 HIGH. Filtered by these criteria jobs are usually having good data locality, they are well-balanced and show high performance.

The deeper analysis of such jobs allows revealing optimal command line and compiler options for the variety of categories of standard applications and algorithm implementations. Once such a job is approved to be a well-optimized typical example of a SW package usage or algorithm implemented, a proper tag, corresponding to such a category can be set (like job_sw VASP). This provides means for the comparative analysis of similar jobs. This can also serve as a good basis for the more detailed analysis of the whole job collection and revealing inefficient applications and users that use resources inefficiently.

3.3. Finding applications with special need for large amounts of memory

Many users of supercomputer complex run applications that are resource-demanding regarding amount of available memory per process. Such applications are usually effective enough, but are often scaled down in different ways to fit available memory, for example, reducing number of MPI processes per node and so on.

Such applications are usually run on a considerable number of nodes and LoadAvg values are below the number of CPU cores per node. These jobs are usually tagged with avg_memload HIGH, and related to node and core usage tags avg_LA SINGLE CORE, job_nnodes FEW or job_nnodes MANY.

If such a job is found in the 6-cored CPU "Regular 6" partition (tagged with job_partition REGULAR6), even changing allocation to the "Regular 4" partition can be an optimization choice leading to reducing the number of idle CPU cores per node by 4 cores (2*6–2*4).

Some of such applications can also benefit from moving to hybrid MPI+OpemMP or MPI+Cilk models. If such a model cannot be applied, some of the applications can be reallocated to SMP partition with much larger amounts of memory available.

3.4. Revealing categories of issues and inefficient behavior

The accumulation of statistics and knowledge on the problems of parallel applications is one of the most important components of the HPC center job collection analysis. The ability to add tags to the analyzed jobs related to the implementation issues found is a useful feature for this purpose as well as tags corresponding to non-efficient use of computing resources, hardware problems or other features found in course of job execution characteristics analysis.

When analyzing inefficient, abnormal application behavior of a single run or of a sequence of jobs, based on the certain software package, it is often needed to contact the user, the application owner who can provide additional information on the program details: algorithm implementation used, program architecture and structure, computing model and so on up to dependencies on input data and command line options. All this information should be recorded to aid further analysis of similar applications and categories.

If any application run is being analyzed it is useful to mark and tag the used system software details. This is true first of all regarding the math libraries used, compiler and compiler options, MPI type, etc. This provides the basis for the comparative analysis of similar jobs or sequences of jobs. If differences in behavior are found, one can continue deep study on the reason origin: user application reaction, system software configuration, etc.

All widely-met issues like data race, deadlocks and so forth can be marked by special tags (job_bug DATA RACE, job_bug DEADLOCK, etc.). This can help in further analysis of other

jobs. One can compare strange program behavior to the analyzed profiles marked as having specific issues. Once a similar behavior is found, it can be a key to resolving the problems of the originally analyzed job.

Conclusion and future work

Close-future plans include implementation of the Octoshell [7, 8] module for full project-oriented workflow support and authentication, thus securing accessibility of the proposed service for any user. We expect it ready by the middle of 2016. By that time we also plan to extend supported tag set and adjust criteria for existing tags if needed.

To sum up, a user-friendly, useful and effective technique for filtration, grouping and further analysis of the whole queued job collection of large-scale HPC systems based on system monitoring and resource manager data is proposed and implemented. The developed tool is evaluated in the every-day practice of the Supercomputer Center of Lomonosov Moscow State University, providing means for effective analysis for any and every user application run. The priceless collection of information on all finished jobs is already being enriched in a 24/7 mode for several month.

The work was funded in part by the Russian Foundation for Basic Research (grants №16–07–00972A, №13–07–00786A), Russian Presidential study grant (SP-1981.2016.5), and by the Ministry of Education and Science of the Russian Federation, Agreement No. 14.607.21.0006 (unique identifier RFMEFI60714X0006).

References

1. *Top50 Supercomputers of Russia and CIS*. Available at: http://top50.supercomputers.ru/ (accessed 15.02.2016).
2. *Top500 Supercomputer Sites*. Available at: http://top500.org/ (accessed:15.02.2016).
3. Antonov A., Zhumatiy S., Nikitenko D., Stefanov K., Teplov A., Shvets P. *Analysis of Dynamic Characteristics of Job Stream on Supercomputer System Numerical Methods and Programming*. 2013. vol. 14, no. 2. pp. 104–108.
4. Safonov A., Kostenetskiy P., Borodulin K., Melekhin F. A Monitoring System for Supercomputers of SUSU. *Russian Supercomputing Days International Conference, Moscow, Russian Federation, 28-29 September, 2015, Proceedings*. CEUR Workshop Proceedings, 2015. vol. 1482. pp. 662–666.
5. Stefanov K. et al. Dynamically Reconfigurable Distributed Modular Monitoring System for Supercomputers (DiMMon). *Procedia Computer Science / Elsevier B.V.*. 2015. vol. 66. pp. 625–634. DOI: 10.1016/j.procs.2015.11.071.
6. Nikitenko D. Complex Approach to Performance Analysis of Supercomputer Systems Based on System Monitoring Data. *Numerical Methods and Programming*. 2014. vol. 15. pp. 85–97.
7. Voevodin V., Zhumatiy S., Nikitenko D. Octoshell: Large Supercomputer Complex Administration System. *Russian Supercomputing Days International Conference, Moscow, Russian Federation, 28-29 September, 2015, Proceedings*. CEUR Workshop Proceedings, 2015. vol. 1482. pp. 69–83.
8. Nikitenko D., Voevodin V., Zhumatiy S. Resolving Frontier Problems of Mastering Large-Scale Supercomputer Complexes. *Proceedings of the ACM International Conference on Computing*

Frontiers (CF'16), Como, Italy, 16-18 May, 2016. ACM New York, NY, USA, 2016. pp. 349–352. DOI: 10.1145/2903150.2903481.

9. Voevodin Vl., Antonov A., Bryzgalov P., Nikitenko D., Zhumatiy S., Sobolev S., Stefanov K., Voevodin Vad. Practice of "Lomonosov" Supercomputer. *Open Systems*. 2012. no. 7. pp. 36–39.
10. Zhumatiy S., Nikitenko D. Approach to Flexible Supercomputers Management. *International Supercomputing Conference Scientific Services & Internet: All Parallelism Edges, Novorossiysk, Russian Federation, 23-28 September, 2013, Proceedings*. MSU, 2013. pp. 296–300.
11. Voevodin Vl. Supercomputer Situational Screen. *Open Systems*. 2014. no. 3. pp. 36–39.
12. Shvets P. , Antonov A., Nikitenko D., Sobolev S., Stefanov K., Voevodin Vad., Voevodin V., Zhumatiy S. An Approach for Ensuring Reliable Functioning of a Supercomputer Based on a Formal Model. *Parallel Processing and Applied Mathematics. 11th International Conference, PPAM 2015, Krakow, Poland, September 6-9, 2015*. Springer International Publishing. vol. 9573. pp. 12–22. DOI: 10.1007/978-3-319-32149-3_2.
13. Voevodin V., Antonov A., Dongarra J. AlgoWiki: an Open Encyclopedia of Parallel Algorithmic Features. *Supercomputing Frontiers and Innovations*. 2015. vol. 2, no. 1. pp. 4–18. DOI: 10.14529/jsfi150101.
14. SLURM Workload Manager. Available at: http://slurm.schedmd.com/ (accessed: 15.02.2016).
15. Cleo Cluster Batch System. Available at: http://sourceforge.net/projects/cleo-bs/ (accessed: 15.02.2016).
16. Ganglia Monitoring System. Available at: http://ganglia.sourceforge.net/ (accessed:15.02.2016).
17. Collectd – The System Statistics Collection Daemon. Available at: https://collectd.org/ (accessed: 15.02.2016).
18. Clustrx. Available at: http://www.t-platforms.ru/products/software/clustrxproductfamily/clustrxwatch.html (accessed: 15.02.2016).
19. jQuery & jQuery UI. Available at: http://jqueryui.com/ (accessed: 15.02.2016).
20. TagIt. Available at: http://aehlke.github.io/tag-it/ (accessed 15.02.2016).

УДК 004.457, 004.382.2 DOI: 10.14529/cmse160403

ИССЛЕДОВАНИЕ ИНТЕГРАЛЬНЫХ ХАРАКТЕРИСТИК СУПЕРКОМПЬЮТЕРНЫХ ПРИЛОЖЕНИЙ ДЛЯ ВСЕГО ПОТОКА ЗАДАЧ БОЛЬШИХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

© 2016 г. Д.А. Никитенко, В.В. Воеводин, А.М. Теплов, С.А. Жуматий, Вад.В. Воеводин, К.С. Стефанов, П.А. Швец

Московский государственный университет имени М.В. Ломоносова
(119991 Москва, ул. Ленинские Горы, д. 1)
E-mail: dan@parallel.ru, alex-teplov@yandex.ru, serg@parallel.ru, vadim@parallel.ru, cstef@parallel.ru, shvets.pavel.srcc@gmail.com

Поступила в редакцию: 11.04.2016

Эффективность работы суперкомпьютерных систем зависит от множества факторов. В условиях одновременной работы множества пользователей особую роль играет контроль использования выделенных для расчетов ресурсов. Важно, чтобы в распоряжении пользователей была подробная информация о свойствах выполненных задач. В условиях групповой работы над прикладными задачами дополнительно стоит выделить необходимость контроля использования ресурсов участниками проекта руководителем работ. К сожалению, такие сведения сейчас как правило не доступны. Этот пробел призван восполнить разработанный авторами подход к получению и исследованию интегральных характеристик суперкомпьютерных приложений для всего потока задач больших суперкомпьютерных систем. В основе подхода лежит использование данных системного мониторинга, построение интегральных характеристик отдельных запусков для всего множества выполненных задач, деление их на классы, выявление особенностей запусков.

Ключевые слова: суперкомпьютер, эффективность, системный мониторинг, классы задач, интегральные характеристики задач, поток задач, контроль использования вы-числительных ресурсов.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Nikitenko D.A., Voevodin V.V., Teplov A.M., Zhumatiy S.A., Voevodin Vad.V., Stefanov K.S., Shvets P.A. Supercomputer Application Integral Characteristics Analysis for the Whole Queued Job Collection of Large-Scale HPC Systems // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2016. Т. 5, № 4. С. 32–45. DOI: 10.14529/cmse160403.

Литература

1. Top50 Supercomputers of Russia and CIS. URL: http://top50.supercomputers.ru/ (дата обращения: 15.02.2016).

2. Top500 Supercomputer sites. URL: http://top500.org/ (дата обращения: 15.02.2016).

3. Antonov A., Zhumatiy S., Nikitenko D., Stefanov K., Teplov A., Shvets P. Analysis of dynamic characteristics of job stream on supercomputer system Numerical Methods and Programming, 2013. Vol. 14, No. 2. P. 104–108.

4. Safonov A., Kostenetskiy P., Borodulin K., Melekhin F. A monitoring system for supercomputers of SUSU // Russian Supercomputing Days International Conference, Moscow, Russian Federation, 28-29 September, 2015, Proceedings. CEUR Workshop Proceedings, 2015. Vol. 1482. P. 662–666.

5. Stefanov K. et al. Dynamically Reconfigurable Distributed Modular Monitoring System for Supercomputers (DiMMon) // Procedia Computer Science / Elsevier B.V., 2015. Vol. 66. P. 625–634.

6. Nikitenko D. Complex approach to performance analysis of supercomputer systems based on system monitoring data // Numerical Methods and Programming, 2014. Vol. 15. P. 85–97.

7. Voevodin V., Zhumatiy S., Nikitenko D. Octoshell: Large Supercomputer Complex Administration System // Russian Supercomputing Days International Conference, Moscow, Russian Federation, 28-29 September, 2015, Proceedings. CEUR Workshop Proceedings, 2015. Vol. 1482. P. 69–83.

8. Nikitenko D., Voevodin V., Zhumatiy S. Resolving frontier problems of mastering large-scale supercomputer complexes // Proceedings of the ACM International Conference on Computing Frontiers (CF'16), Como, Italy, 16-18 May, 2016. ACM New York, NY, USA, 2016. P. 349–352.

9. Voevodin Vl., Antonov A., Bryzgalov P., Nikitenko D., Zhumatiy S., Sobolev S., Stefanov K., Voevodin Vad. Practice of "Lomonosov" Supercomputer // Open systems, 2012. No. 7. P. 36–39.

10. Zhumatiy S., Nikitenko D. Approach to flexible supercomputers management // International supercomputing conference Scientific Services & Internet: all parallelism edges, Novorossiysk, Russian Federation, 23-28 September, 2013, Proceedings. MSU, 2013. P. 296–300.

11. Voevodin Vl. Supercomputer situational screen // Open systems, 2014. No. 3. P. 36–39.

12. Shvets P. , Antonov A., Nikitenko D., Sobolev S., Stefanov K., Voevodin Vad., Voevodin V., Zhumatiy S. An Approach for Ensuring Reliable Functioning of a Supercomputer Based on a Formal Model // Parallel Processing and Applied Mathematics. 11th International Conference, PPAM 2015, Krakow, Poland, September 6-9, 2015. Springer International Publishing. Vol. 9573. P. 12–22.

13. Voevodin V., Antonov A., Dongarra J. AlgoWiki: an Open Encyclopedia of Parallel Algorithmic Features // Supercomputing Frontiers and Innovations, 2015. Vol. 2, No.1. P. 4–18.

14. SLURM workload manager. URL: http://slurm.schedmd.com/ (дата обращения: 15.02.2016).

15. Cleo cluster batch system. URL: http://sourceforge.net/projects/cleo-bs/ (дата обращения: 15.02.2016).

16. Ganglia Monitoring System. URL: http://ganglia.sourceforge.net/ (дата обращения: 15.02.2016).

17. Collectd – The system statistics collection daemon. URL: https://collectd.org/ (дата обращения: 15.02.2016).

18. Clustrx. URL: http://www.t-platforms.ru/products/software/clustrxproductfamily/clustrxwatch.html (дата обращения: 15.02.2016).

19. jQuery & jQuery UI. URL: http://jqueryui.com/ (дата обращения: 15.02.2016).

20. TagIt. URL: http://aehlke.github.io/tag-it/ (дата обращения: 15.02.2016).