DOI: 10.14529/jsfi180201

Deep Analysis of Job State Statistics on Lomonosov-2 Supercomputer

Dmitry A. Nikitenko[1], *Vadim V. Voevodin*[1], *Sergey A. Zhumatiy*[1]

It is a common knowledge that the increasingly growing capabilities of HPC systems are always limited by a number of efficiency related issues. The reasons can be very different: hardware failures, incorrect job scheduling, peculiarities of algorithm, chosen programming technology specifics, etc. Most of these issues can be detected after precise analysis, but is a very resourceful way to study every application run. Therefore we performed less complicated analysis of the whole supercomputer job flow. In this paper we share our experience of analyzing user applications job states assigned by the SLURM resource manager that is used on the Lomonosov-2 system at Supercomputing center of Lomonosov Moscow State University. The statistics on job states was collected and it revealed that the ratio of correctly finished jobs (with the COMPLETED state) was rather low. The jobs owners were asked if the distribution of their jobs states is normal regarding their applications. This user feedback was processed, and some new ways of efficiency gain were revealed as the result.

Keywords: HPC, supercomputer, parallel computing, efficiency analysis, job state.

Introduction

Supercomputers are constantly getting more and more powerful – the current #1 supercomputer in the world, Summit, has more than 180 PFlop/s in peak [1]. Such trend is dictated by the desire to solve increasingly complex computational problems that arise in various scientific areas. But in order to achieve this goal it is necessary not only to increase the performance of computing systems, but also to ensure high efficiency of their functioning. And this is a big problem in modern systems, since the efficiency of their usage is usually low [2].

This problem is directly related to the low efficiency of individual parallel applications running on the supercomputer. One of the main reasons for applications not being effective is quite a significant ratio of incorrect application launches. There are various reasons for that: incorrect input data or wrong input parameters specified in the application, issues with the file system or interconnect switches, MPI-related issues, compute node fails, etc. All these problems decrease the useful utilization of supercomputers, so it is natural for the management of supercomputer centers to try minimizing the number of such problems.

In the Research computing center of Moscow State University (RCC MSU) we decided to conduct a study of these issues on the Lomonosov-2 supercomputer installed in our supercomputer center. One of the basic information that can be used to estimate the correctness of supercomputer job flow is the states of finished job runs. We collected the statistics on the states of all finished jobs during several months of Lomonosov-2 functioning and found out that the ratio of correctly finished jobs (with the COMPLETED state) is surprisingly low. So it was decided to ask the users about the reasons of such seemingly unproductive behavior, concerning their jobs.

This paper is devoted to the deep analysis of job state statistics on Lomonosov-2 supercomputer based on the communication with the users.

The rest of the paper is organized as follows. Section 1 describes our previous research concerning efficiency analysis of particular parallel applications and the supercomputer usage in

[1]Research Computing Center of Lomonosov Moscow State University, Moscow, Russian Federation

general, as well as related works performed by other scientists. The process of collecting data for the job state analysis is described in Section 2. Section 3 is devoted to the analysis of collected data as well as the results and useful ideas developed during this study. Last section contains conclusion and our plans for future work.

1. Background and Related Work

The efficiency of any supercomputer is affected by many factors. Our team has a long experience in optimizing the efficiency of our supercomputers. The first important step is to understand the supercomputer behavior in general and identify the key factors that cause inefficiency. Then it is possible to eliminate or at least to reduce the impact of the found factors and evaluate the results of taken actions.

Our RCC MSU team uses the approach described in [3], which enables us to collect the most detailed information on both the hardware part [4] and the computing job [5]. The collected data is analyzed both in automatic and manual modes. The system monitoring data gives information on how well the supercomputer is loaded. But a high load does not always mean high efficiency – if a large amount of resources is spent on jobs that did not produce a result, then this time was wasted.

Similar statistics can be found, for example, in [6], where various aspects of computing jobs behavior on the Blue Waters supercomputer are considered, although there is no data on the distribution of job exit states. A more interesting analysis is given in the article [7], which deals with distributions of jobs of the Kraken supercomputer, including such characteristics as the frequency of cancellation of jobs and the accuracy of setting their the time limits. Another existing solution is based on using XDMoD tool designed for managing of HPC systems [8, 9].

In these papers, a lot of analyzing results of the job behavior on the largest supercomputers are given, but no conclusions or assumptions are made on how to improve their efficiency. In our work we propose methods for increasing the efficiency of the supercomputer based on the results achieved in communication with the users of Lomonosov-2 supercomputer.

2. Collecting Information about Job States

We store all the data collected by monitoring system on running and finished jobs, as well as accounting data from SLURM manager, in our single PostgreSQL database. Therefore we only need to filter out needed information and analyze it accordingly. In this study, we analyzed all jobs finished since January 1, 2018 till May 28, 2018. Our goal was to find cases and users whose jobs mostly seems as “abnormal” according to the exit states of their jobs. For each user, a total amount of cpu-hours for his jobs was computed and compared with the amount of cpu-hours for his jobs with particular exit states. We used the following criteria in order to detect users with “abnormal” behavior:

- (amount of cpu-hours for user jobs with COMPLETED state) / (Total amount of cpu-hours for this user) < 0.5;
- *(The same for jobs with CANCELLED or FAILED state)* > 0.2;
- *(The same for jobs with TIMEOUT state)* > 0.2.

If at least one of these criteria was fulfilled, user was asked via email to explain the reasons for such big ratio of jobs with not COMPLETED state. We sent personalized statistics about jobs in emails to help users to better understand our interest. We chose thresholds for these

criteria empirically, using our experience and expectations. This process was repeated for each of 3 main partitions of Lomonosov-2 supercomputer, because different partitions have specific limits and different sets of jobs running on them. Therefore, several users were emailed more than once with questions about different partitions.

We detected 97 cases of criteria triggering, each case was a unique combination of user name and supercomputer partition. Since we are interested in analyzing overall statistics on the behavior of the whole supercomputer, there is no need to analyze small cases, so we filtered out ones with less than 20 000 cpu-hours. It resulted in 65 cases for 53 different users; an email was sent for each of these cases. It should be noted that these users cover the majority of the Lomonosov-2 usage – these users have consumed 85% of cpu-hours provided by the supercomputer during the considered time period.

Most of the users received only 1 email, but there were 8 users that received 2 emails (concerning 2 different partitions) and 2 users was asked questions about all our partitions, therefore each received 3 emails.

In found cases all 3 criteria were triggered 23 times, only 2 criteria – 26 times and 17 times only one criteria was triggered. We sent questions about small ratio of COMPLETED states (first criteria) 47 times, as well as 36 and 52 questions about big ratio of CANCELLED+FAILED and TIMEOUT states, accordingly.

3. Analysis of Job State Information

After collecting the information on the job states from the users, the next step is to analyze the obtained data. We received answers from 39 users out of 53 (74% users responded). The responded users were responsible for 56.5% of all CPU hours consumed by all users during the considered time period (between Jan 1 and May 28). It seems to be a good result since there were no obligations to provide feedback answering our questions.

We grouped all the answers from the users based on the job state (TIMEOUT, CANCELLED and FAILED states) and the reason for such job behavior. Further the statistics for each state is described.

3.1. TIMEOUT Job State

We received the most number of answers about big ratio of TIMEOUT job state – 32 users responded. The majority of these users (21 of 32) noticed that such behavior was absolutely normal for their jobs since the current time limit for job execution was not enough for them. In this case a user ran the job till the time limit and created a control point, so he could continue the calculation from that point in the next job run.

Many of users intentionally going for the time limit showed us that we needed to distinguish such behavior from unintentionally exiting with TIMEOUT job state, since the former was a normal job behavior and the latter could be an indicator of incorrect or inefficient job execution. So, after analyzing this information, we decided to use special option specified for the SLURM resource manager that would be used in Lomonosov-2 supercomputer. Specifying this option, users can explicitly mark that this new job is planned to reach TIMEOUT limit. This will help us in future to divide such cases, leading to more accurate statistics.

Next, 7 users mentioned that they had not correctly estimated the time needed for job execution; 2 of them added that they had already fixed that issue, and 4 of them said that the problem was most likely related to the issues with the used software package.

Another 4 users admitted that the TIMEOUT job state was due to inefficient software implementation bad parallelization, outdated software, packages working surprisingly inefficiently on Lomonosov-2 supercomputer. This is a good opportunity for optimization since it will help users to conduct experiments faster, and to increase the overall efficiency of the supercomputer as well.

The last 7 of 32 users said that the reason for many of their jobs to be finished with TIMEOUT job state was that they evaluated new methods and approaches. In this case it is hard to deal with these issues, since the problem is not technical but more of semantic kind.

3.2. CANCELLED Job State

We received response from 24 users about big ratio of jobs with CANCELLED job state. 10 of them said that, as in the case of TIMEOUT state, this situation was completely normal. It was dictated by the specifics of such jobs: it was impossible to predict when the calculation is completed in advance, so users had to specify bigger time limit. And when users decided that the needed computational goal was achieved, they manually canceled the job, since after that moment there would be no useful work done. Since such job behavior is normal, it would be useful to divide this scenario from other types of cancelled jobs; but as for now, there is no simple way to accurately identify it.

Other types of jobs with CANCELLED job state are the following. 13 users stated that they needed debugging job launches, but the reasons were different – it could be rather a test launch in order to check the correctness of planned experiment or an evaluation of new method or approach. The main optimization that can be done in this situation is to reduce the number of test launches in the main working partitions of the supercomputer – in our case, a special test partition should be used. Since it seems that in some cases users do not use this test partition on purpose, it is probably necessary to adjust the quotas and time limits for this partition to better suit the needs of users.

It is worth mentioning separately the interesting response of the last user. This user said that, due to the quotas on the maximum number of jobs running on the Lomonosov-2 supercomputer, he sometimes needed to cancel less priority jobs in order to be able to launch more priority jobs. This situation requires a more detailed discussion with the user – maybe this could be a signal to system administrators to adjust the quotas and policies.

3.3. FAILED Job State

The information about failed jobs is usually the most useful for us since it covers most of the situations that we can potentially fix. We received answers from 19 users which can be grouped into 5 different types.

5 users claimed that their jobs failed due to different system problems like compute node fails or issues with MPI library. This is exactly the situation where we as system administrators need to take prompt actions. Modern supercomputers are very big and complex, so such issues are going to happen in any way, therefore our goal is to eliminate such problems as quickly as

possible and take prophylactic measures trying to minimize the frequency of occurrence of such errors.

The problems on the user software side were the main reasons of failed jobs, as stated by the biggest group of the responders (7 out of 19). As in the case of inefficient implementation mentioned in subsection 3.1, it is desirable to interact with such users in order to detect the root causes of these problems and to correct and optimize their applications, which will lead to the increase of supercomputer useful utilization and help users to conduct experiments more effectively.

Next, 2 users stated that there were some system problems that led to failed jobs, but at the moment they were fixed by the supercomputer support team. In such case there is no need to take any action.

As mentioned by 3 more users, the big ratio of failed jobs in their cases was caused by the need for many test launches. As in the case of cancelled jobs, we should try to reduce the number of such launches by convincing users to use test partitions created specifically for this goal.

The last type of responses is quite unusual in our opinion. 2 users said the FAILED job state was a completely normal situation that could appear in software they were using. On the one hand, this situation should be fixed because FAILED job state should be only used in case of some errors occurred, but here it is used to label normal behavior. On the other hand, this is usually quite difficult to change the behavior of large and complex user software, especially with some ready-to-use packages involved. In any case, such information is a subject for further detailed study and discussion with the users.

Conclusions and Future Work

Many of the jobs with TIMEOUT and CANCELLED states appeared to be inefficient by the application nature and implementation peculiarities, that was proved by our users – jobs owners. So this illustrates the necessity for system support to work on improving the efficiency of supercomputer applications together with their owners.

It is revealed, that the ratio of COMPLETED state jobs which are usually considered as successful runs, is lower than can be expected. Some jobs just cannot finish with such state because of their specifics, for example, the jobs that compute until they meet timeout and are marked as TIMEOUT by state. It means that there is at least a good way to provide a SLURM option for the users that can mark the job as considered to be run until the timeout. This would allow subtracting these jobs from suspicious jobs category. Then, it would be reasonable to adjust currently used job statistics, splitting the TIMEOUT and CANCELLED categories in two groups each.

Many user applications are based on common packages and libraries. It seems a good idea to check the correlation of all revealed cases with the usage of these popular software packages and libraries. This would allow us detecting similar problems that experience different users and workgroups. This work is planned to be done using XALT tool designed for collecting job-level information about used libraries, modules and executables [10].

Another interesting way of future research related to the obtained results is to automate the process of getting user feedback. This can be developed as a special service of currently used Octoshell system [11] which is used for our supercomputer center workflow management. It could provide reports and surveys with statistics of the job states, enriched with more detailed data on

job resource utilization, revealed efficiency-related anomalies [12], used software packages and libraries, etc.

Acknowledgements

The results are obtained with the financial support of the Russian Foundation for Basic Research (grants No. 17-07-00664 and 17-07-00719). The research is carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University.

References

1. TOP500 List – June 2018 – TOP500 Supercomputer Sites. `https://www.top500.org/list/2018/06/`

2. Voevodin, V., Voevodin, V.: Efficiency of Exascale Supercomputer Centers and Supercomputing Education. In: High Performance Computer Applications: Proceedings of the 6th International Supercomputing Conference in Mexico (ISUM 2015). pp. 14–23. Springer, Cham (2016), DOI: 10.1007/978-3-319-32243-8_2

3. Nikitenko, D., Stefanov, K., Zhumatiy, S., Teplov, A., Shvets, P., Voevodin, Vad.: System monitoring-based holistic resource utilization analysis for every user of a large HPC center. Algorithms and Architectures for Parallel Processing, LNCS. 10049. 305–318. Springer (2016). DOI: 10.1007/978-3-319-49956-7_24

4. Stefanov, K., Voevodin, V., Zhumatiy, S., Voevodin, V.: Dynamically Reconfigurable Distributed Modular Monitoring System for Supercomputers (DiMMon). Procedia Computer Science 66, 625–634 (2015), DOI: 10.1016/j.procs.2015.11.071

5. Nikitenko, D., Antonov, A., Shvets, P., Sobolev, S., Stefanov, K., Voevodin, V., Voevodin, V., Zhumatiy, S.: JobDigest Detailed System Monitoring-Based Supercomputer Application Behavior Analysis. In: Supercomputing. Third Russian Supercomputing Days, RuSCDays 2017, Moscow, Russia, September 25–26, 2017, Revised Selected Papers. pp. 516–529. Springer, Cham (2017), DOI: 10.1007/978-3-319-71255-0_42

6. Jones, M.D., White, J.P., Innus, M., DeLeon, R.L., Simakov, N., Palmer, J.T., Gallo, S.M., Furlani, T.R., Showerman, M., Brunner, R., Kot, A., Bauer, G., Bode, B., Enos, J., Kramer, W.: Workload Analysis of Blue Waters (2017), `http://arxiv.org/abs/1703.00924`

7. You, H., Zhang, H.: Comprehensive workload analysis and modeling of a petascale supercomputer. In: Workshop on Job Scheduling Strategies for Parallel Processing. pp. 253–271. Springer (2012), DOI: 10.1007/978-3-642-35867-8_14

8. Furlani, T.R., Schneider, B.L., Jones, M.D., et al.: Using XDMoD to facilitate XSEDE operations, planning and analysis In:Proceedings of the Conference on Extreme Science

and Engineering Discovery Environment: Gateway to Discovery, ACM, p. 46, (2013), DOI: 10.1145/2484762.2484763

9. Palmer, J.T., Gallo, S.M., Furlani, T.R., et al.: Open XDMoD: A tool for the comprehensive management of high-performance computing resources. In: Computing in Science & Engineering, vol. 17, no. 4, pp. 52–62. IEEE (2015), DOI: 10.1109/MCSE.2015.68

10. Agrawal, K., Fahey, M.R., McLay, R., Doug, J.: User environment tracking and problem detection with XALT. In: Proceedings of the First International Workshop on HPC User Support Tools, pp. 32–40. IEEE press (2014), DOI: 10.1109/HUST.2014.6

11. Nikitenko, D., Voevodin, Vl., Zhumatiy, S.: Resolving frontier problems of mastering large-scale supercomputer complexes. In: ACM International Conference on Computing Frontiers (CF'16), pp. 349–352. ACM New York (2016), DOI: 10.1145/2903150.2903481

12. Voevodin, Vl., Voevodin, Vad., 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, June 20–24, 2016. AIP Conference Proceedings, vol. 1776, pp. 090015-1–090015-4 (2016), DOI: 10.1063/1.4965379