DOI: 10.15439/2020F208
ISSN 2300-5963 ACSIS, Vol. 21

Simulator of a Supercomputer Job Management System as a Scientific Service

Gennadiy Savin, Boris Shabanov, Dmitriy Lyakhovets, Anton Baranov, Pavel Telegin
Joint Supercomputer Center of the Russian Academy of Sciences
Leninsky prospect, 32a, Moscow, 119334, Russian Federation
Email: [savin, shabanov]@jscc.ru, anetto@inbox.ru, antbar@mail.ru, ptelegin@jscc.ru

Abstract*—Job management system (JMS) is an important part of any supercomputer. JMS creates a schedule for launching jobs of different users. Actual job management systems are complex software systems with a number of settings. These settings have a significant impact on various JMS metrics, such as supercomputer resources utilization, mean waiting time of a job in queue, and others. Various JMS simulators are widely used to study the influence of JMS settings or modifications, new scheduling algorithms, jobs input stream parameters or available computing resources for JMS efficiency metrics. The article presents the comparative analysis results of the actual JMS simulators (Alea, ScSF, Batsim, AccaSim, Slurm simulator) and their application areas. The authors consider new ways to use the JMS simulator as a scientific service for researchers. With such a service, the researchers are able to study various hypotheses about JMS efficiency, algorithms or parameters. This gives the folowing: (1) research is performed on the service side around the clock, (2) the simulator accuracy or adequacy is provided by the service, (3) the research results reproducibility is ensured, and the simulator-as-a-service becomes a single entry point for the researchers.

I. Introduction

JOB MANAGEMENT system (JMS) is an essential software for multi-user high performance computing [1]. JMS handles a queue of user jobs, determines job launch order, allocates computing nodes for launched jobs, controls job termination and checks that nodes are freed after job termination. A number of metrics, such as resources utilization, mean waiting time of a job in queue, and others, measure JMS quality.

Modern JMS have been evolving for decades, and now JMS are complex software systems with a lot of adjustable parameters. The example of the parameters is a scheduling algorithm and its options, users and groups priority, job size limitation. The parameters optimization could be challenging because parameters influence on JMS quality can be far from obvious.

A job launch order in most of the JMS is based on job time limits specified by users. JMS terminates the job when it reaches job limit. Most of JMSs provide worst-case job launch time to a user (if no nodes are broken) when every job in queue ends at its limit. The research [2] shows that most of the jobs end far before their limits.

The work was carried out within the RFBR grant (project No. 18-29-03236), and the framework the JSCC RAS government assignment

Every job launch time forecasting is a challenge. The forecast could provide to a user more precise launch time estimation. Such forecast is especially important for geographically distributed supercomputer systems. Note the integration of geographically distributed supercomputer resources is a steady high-performance computing trend [3]. The main goal of this integration is creation of a digital platform to meet the scientific, educational and industrial needs for high-performance computing. The digital platform can include several JMSs. Each JMS could have its own input job stream, so the digital platform integrates multiple input job streams. In general, any job from any input stream can be assigned to any JMS of the platform. As a result, the job management complexity increases significantly, so as the prediction complexity of the job launch time and location. The launch location is a supercomputer in the distributed digital platform, where the job will be executed. To schedule jobs in a distributed system efficiently, it is necessary to predict the release time of the required computing resources in each of the JMS accurately [4]. Job launch time forecast allows to determine the JMS where a job could be executed faster. The forecast could be used to schedule a global job queue for the distributed supercomputer system [5]. Job from the global queue could be executed on less busy JMS which reduces resource imbalance. This can be achieved by modelling of the management system in order to predict the launch time and location for each job. We assume to model up to hundreds of thousands jobs per year executed on tens of thousands nodes from thousands of different users.

Of particular interest are different aspects of JMS modelling to solve two tasks mentioned above. The first task is JMS parameters optimization. The second task is job launch time forecasting. Most popular JMS model implementation is a simulator, thus we would use the words «model» and «simulator» as synonyms.

II. Related Work

Nowadays a number of JMS is available, for instance, SLURM [6], PBS [7], LSF [8]. SUPPZ [9] is an example of domestic JMS, which has been used in Joint Supercomputer Center of the Russian Academy of Sciences (JSCC RAS) for more than 20 years. Of particular note is JMS for relatively small clusters named OAR [10]. In the paper [11] authors investigate some techniques to provide malleable behavior on

MPI applications and the impact of this support upon the OAR resource manager.

Many JMS models can be found in the literature. There are both models of existing JMS (like SLURM, SUPPZ, OAR) and models of general JMS. Let us consider recent papers for JMS modelling. Existing JMS modelling tools can be divided into 3 classes: modelling languages, software platforms and simulators.

Modelling languages fully support the modelling process — the model time control and the object interaction in the system. This allows the researcher to focus on the description of the JMS model essential properties and characteristics. In this case, the researcher must independently reproduce the entire JMS operation logic — build a supercomputer model with the given characteristics and the jobs processing order, create a job scheduler with a given scheduling algorithm, describe the input job stream, develop program modules for conducting the experiment and collecting the necessary results. Specialized languages such as AnyLogic [12], ExtendSIM [12], GPSS World [13], Simulink [12] can be used to build a JMS model.

AnyLogic is a general-purpose modelling language developing since 2000. Model development is performed in a graphical interface, the Java programming language is supported to finalize the components. ExtendSIM has been in development since 1987. ExtendSim has a user-friendly interface and does not require special knowledge and programming skills. It is enough to draw a block diagram of the modeled process and enter the initial data using the necessary block settings. GPSS World is one of the earliest modelling languages created in 1961. A GPSS program is a sequence of statements displaying events. Simulink is a modelling language developing since 1984 that provides tight integration with MATLAB.

Software platforms for JMS modelling allow reducing the time to implement the model due to the parts of the modeled systems and components for displaying various data (for example, statistical) implemented in the platform. The software platform provides typical entities, such as «computing module», «job», «job scheduler» with a wide range of different characteristics. The developer builds a model from ready-made large blocks and configures them for solving the problem. Software platforms such as SimGrid [14], GridSim [15] are widespread. SimGrid is a software platform for developing distributed application simulators, developing since 1999. GridSim developing since 2002 is widely used by various researchers to model grid systems and JMS.

JMS simulators provide the researher with a ready model that needs to be configured. Model configuration may require code development. Examples of JMS simulators are MONARC [16], Alea [17], OptorSim [18], WorkflowSim [19]. MONARC has been developed since 2000 and is designed to analyze large-sized systems. A key aspect of this simulator is wide opportunities for monitoring system components [20]. Alea is based on GridSim and has been in development since 2007. The Alea main purpose is the scheduling algorithms study, and a number of scheduling algorithms is already implemented in the simulator. WorkflowSim has been in development since 2012. The WorkflowSim main purpose is the job stream processing optimization [20]. OptorSim has been under development since 2003. In OptorSim, it is possible to configure the network topology between computing nodes with their throughput and the job data volume.

We are going to take a detailed look at Batsim [21], created for OAR modelling. In the paper authors criticize current situation when other modelling researches are hard to reproduce. Authors mention that often models are not used after results publication. One of the problems in JMS modelling is complexity of experiment reproduction due to model unavailability, lack of input or output data (in case when average metrics are published, but not all of the outputs), no access to the experimental environment [21].The solution presented by the authors is their own model Batsim which can be used by other researchers. Beside that, the authors suggest to publish full experiment plan, including all parameters, inputs and outputs. Batsim validation consists of Gantt chart of workload, plot of the difference between the real and the simulated execution time (so as for submission and turnaround time) of all jobs in workload. The plot comparison is done visually. Authors note the difference between the real system and the model.

Paper [22] states that available modelling tools do not cover the full cycle of modelling, generation, simulation, and analysis. The authors present their own model — scheduler simulation framework (ScSF), model of SLURM. There is no way to use the fixed job input for ScSF because input generator is a mandatory part of ScSF. Paper notes the complexity of long simulations since their few weeks experiment sometimes failed due to power cuts, hypervisor or VM failures and system updates. There is no model validation found in the paper.

Paper [23] represents a new version of SLURM simulator. For model validation there are plots of job start time difference between simulated and real SLURM runs. The authors calculated average and standart deviation of all job start time differences. Plots for real system and its model are compared visually. There are data for natural modelling on a small cluster (10 computing nodes) in the paper.

There is a new Alea version presented in paper [24]. Alea is the model of general JMS. The paper provides technical details of the model and data about simulation speed. There is no model validation found in the paper.

Paper [25] presents an AccaSim model of general JMS, which is compared to Batsim and Alea. Comparison with ScSF is not done due to high system requirements and complex configuration. Also ScSF could not use fixed job input, only generation of a new one was possible during the experiment. There is no model validation found in the paper, but there is validation of input generator.

The above analysis of papers about modelling revealed two scientific problems for further development.

The first problem is model validation, more precisely lack of common ways to measure model adequacy and accuracy. Other researchers skip validation, or compare some plots visually or calculate average metrics. There are no analytic measures for

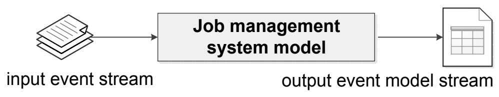

Fig. 1. JMS model research

model accuracy. The authors are investigating this problem, first results are presented at [26].

III. JMS Model as a Scientific Service

Modern way of JMS research could be explained as follows. A researcher creates a JMS model [26], for example, a JMS simulator. The researcher experiments with some input event stream and model with fixed parameters and saves output model stream (see Fig. 1). Output model stream could be saved in some kind of a plain text file or in a database. The database is not a bottleneck in this case. Authors use PostgreSQL. MySQL is used in SLURM simulator.

Then the researcher modifies input event stream (e.g., changing job density) or JMS model (e.g., changing scheduling algorithm or its options), repeats the experiment and gets new output model stream. By analyzing output stream before and after the changes the researcher can determine if modifications were good or bad (for example, old algorithm provided 95% average utilization and the new one provided 96%). Reability of the results is ensured by series of experiments, and results are presented as average of some metric and its standard deviation. Improvement is significant if it is bigger than the deviation.

The above way of research faces the following problems:

1) Researchers have to develop their own JMS model or to master the existing ones — its installing, configuring, ways to form an input event stream, getting and analyzing the result [21].
2) Researchers have to validate JMS model on their own.
3) It is hard to reproduce experiments of other researches. Used models could not be found publicly, or research could not provide all of the input or output streams (for example, only average metrics could be published).

All of the problems can be solved if a JMS model operates alongside with the real JMS. Organization providing the JMS for users could also provide a JMS model as a scientific service for researchers. In that case JMS model development, its installation, configuring and validation is done once by the organization, and researchers use the model over and over again.

JMS model as a scientific service extends the field of JMS research. Often the research relies on some old data (for a period in the past) or a generated input event stream (generated to statistically the same as real stream for some period). Simulator-as-a-service allows to experiment with most recent streams, close to the real-time. Such a simulator could be used to forecast job launch times in real-time. Besides, such a service could constantly calculate model adequacy.

Building the simulator-as-a-service is a challenging job. Let us consider various options of service organization.

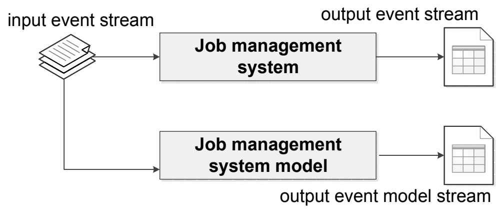

Fig. 2. Using JMS model in real-time

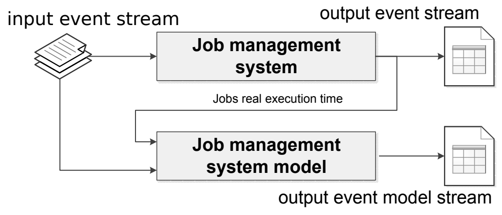

Fig. 3. Using JMS model in real-time with feedback

For real-time research we should duplicate input event stream to both real JMS and its model (see Fig. 2). In this case a technical problem reveals — there is no real job execution time in input event stream, which is often less than the limit specified by user.

Thus, JMS model does not have an important value in input stream — real execution time. Let us consider ways to get it. The first way is to add a feedback from JMS to JMS model (see Fig. 3). After the job ends, JMS notifies the model about real execution time.

The second way is to forecast real execution time. In this case for every submitted task real execution time is forecasted basing on statistics (see Fig. 4).

Result of combination of the two ways is JMS model with feedback, and forecast (see Fig. 5). For every submitted job, real execution time is forecasted basing on statistics. When the job ends, real JMS notifies the model and replaces the forecast with the actual execution time which allows to constantly correct forecasting and improve its precision.

Simulator-as-a-service reduces the complexity for researchers. They can concentrate on scientific aspects of experiments. Additional advantage is reproducibility improvements.

Using a JMS model in real-time enables the continuous comparison of real and model output streams which allows to modify a JMS model to improve its accuracy. Using a JMS model in real-time with forecasting and feedback allows to

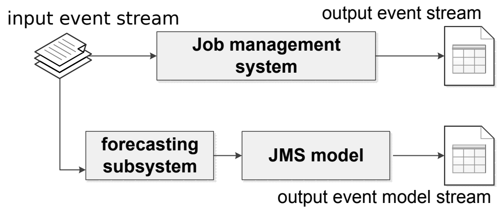

Fig. 4. Using JMS model in real-time with forecasting

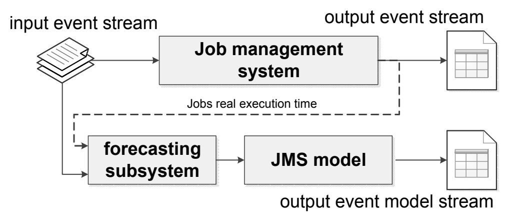

Fig. 5. Using JMS model in real-time with forecasting and feedback

improve forecasting subsystem which later could be used for scheduling.

IV. Conclusion

The authors' JMS models analysis revealed the complexity of other researchers' models reproduction. Lack of used software or its installation or configuration complexity, incomplete input or output event streams, lack of common ways to validate a model — all these makes every researcher to repeat a huge amount of work in order to search, configure, execute and validate a model.

We suggest a new approach for experimental study of JMS based on once developed and then publicly served JMS model used by researchers. The approach does not restrict researchers in using their own models or creating new simulators as a service. Simulator-as-a-service could attract more researchers making JMS model easy to use for any kind of research. The main purpose of the simulator-as-a-service is to research the influence of new scheduling algorithms or scheduling parameters on JMS quality metrics.

We outlined different ways to build a JMS model simulator-as-a-service: to work in real-time, in real-time with feedback, in real-time with forecasting, in real-time with forecasting and feedback.

References

[1] A. Reuther et al. "Scalable system scheduling for HPC and big data," *J. of Parallel and Distributed Computing,* vol. 111, 2018, pp. 76–92. https://dx.doi.org/10.1016/j.jpdc.2017.06.009

[2] A. V. Baranov, E. A. Kiselev, D. S. Lyakhovets, "The quasi scheduler for utilization of multiprocessing computing system's idle resources under control of the Management System of the Parallel Jobs," *Bul. of the South Ural State University. Series Comp. Math. and Software Engineering,* issue 3(4), 2014, pp. 75–84 (in Russian). https://dx.doi.org/10.14529/cmse140405

[3] B. Shabanov, A. Ovsiannikov, A. Baranov, S. Leshchev, B. Dolgov, and D. Derbyshev, "The distributed network of the supercomputer centers for collaborative research," *in Program systems: Theory and applications,* 8:4(35), 2017, pp. 245–262 (In Russian). https://dx.doi.org/10.25209/2079-3316-2017-8-4-245-262

[4] N. N. Kuzyurin, D. A. Grushin, and S. A. Fomin, "Two-dimensional packing problems and optimization in distributed computing systems," *in Proc.of the Institute for System Programming of the RAS,* vol. 26, no 1, 2014, pp. 483–502 (in Russian).

[5] A. I. Tikhomirov, "The English Auction Method for Scheduling Jobs in a Distributed Network of Supercomputer Centers," *Lobachevskii J. of Math.,* vol. 40, issue 5, 2019, pp. 606–613. https://dx.doi.org/10.1134/s1995080219050214

[6] A. B. Yoo, M. A. Jette, and M. Grondona, "SLURM: Simple Linux Utility for Resource Management," *Lecture Notes in Comp. Science,* vol. 2862, 2003, pp. 44–60. https://dx.doi.org/10.1007/10968987_3

[7] R. L. Henderson, "Job scheduling under the Portable Batch System," *Lecture Notes in Comp. Science,* vol. 949, 1995, pp. 279–294. https://dx.doi.org/10.1007/3-540-60153-8_34

[8] IBM Spectrum LSF overview, https://www.ibm.com/support/knowledgecenter/en/SSWRJV_10.1.0/lsf_foundations/chap_lsf_overview_foundations.html

[9] G. I. Savin, B. M. Shabanov, P. N. Telegin, and A. V. Baranov, "Joint Supercomputer Center of the Russian Academy of Sciences: Present and Future," *Lobachevskii J. of Mathematics,* vol. 40, issue 11, 2019, pp. 1853–1862. https://dx.doi.org/10.1134/S1995080219110271

[10] N. Capit et al., "A batch scheduler with high level components," *in IEEE Int. Symp. on Cluster Comp. and the Grid,* Cardiff, Wales, UK, vol. 2, 2005, pp. 776–783. https://dx.doi.org/10.1109/CCGRID.2005.1558641

[11] M. C. Cera et al., "Supporting Malleability in Parallel Architectures with Dynamic CPUSETs Mapping and Dynamic MPI," *in Distributed Computing and Networking,* 2010, pp. 242–257. https://dx.doi.org/10.1007/978-3-642-11322-2_26

[12] I. M. Yakimov, M. V. Trusfus, V. V. Mokshin, and A. P. Kirpichnikov, "AnyLogic, ExtendSim and Simulink Overview Comparison of Structural and Simulation modelling Systems," *in Proc. 3rd Russian-Pacific Conf. on Computer Technology and Applications (RPC), Vladivostok,* 2018, pp. 1–5. https://dx.doi.org/10.1109/RPC.2018.8482152

[13] S. W. Cox, "GPSS World: A brief preview," *in 1991 Winter Simulation Conference Proceedings, Phoenix, AZ, USA,* 1991, pp. 59–61. https://dx.doi.org/10.1109/WSC.1991.185591

[14] A. Legrand, M. Quinson, H. Casanova, and K. Fujiwara, "The SIMGRID Project Simulation and Deployment of Distributed Applications," *in 15th IEEE Int. Conf. on High Performance Distributed Computing, Paris,* 2006, pp. 385–386. https://dx.doi.org/10.1109/HPDC.2006.1652196

[15] S. R. Chelladurai, "Gridsim: a flexible simulator for grid integration study," 2017. https://dx.doi.org/10.24124/2017/1375

[16] I. C. Legrand and H. B. Newman, "The MONARC toolset for simulating large network-distributed processing systems," *in Winter Simulation Conf. Proc. (Cat. No.00CH37165), Orlando, FL, USA,* vol.2, 2000, pp. 1794–1801. https://dx.doi.org/10.1109/WSC.2000.899171

[17] D. Klusacek, H. Rudova, "Alea 2: job scheduling simulator," *in SIMUTOOLS ICST,* 2010. https://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8722

[18] W. H. Bell, D. G. Cameron, F. P. Millar, L. Capozza, K. Stockinger, and F. Zini, "Optorsim: A Grid Simulator for Studying Dynamic Data Replication Strategies," *The Int. J. of High Performance Computing Applications,* 17(4), 2003, pp. 403–416. https://dx.doi.org/10.1177/10943420030174005

[19] W. Chen and E. Deelman, "WorkflowSim: A toolkit for simulating scientific workflows in distributed environments," *in 2012 IEEE 8th Int. Conf. on E-Science, Chicago, IL,* 2012, pp. 1–8. https://dx.doi.org/10.1109/eScience.2012.6404430

[20] J. Taheri, A. Zomaya, S. Khan, "Grid Simulation Tools for Job Scheduling and Data File Replication," *in Scalable Computing and Communications: Theory and Practice,* New Jersey: Wiley, 2013, pp. 777–797.

[21] P. F. Dutot, M. Mercier, M. Poquet, O. Richard, "Batsim: a realistic language-independent resources and jobs management systems simulator," *in Job Scheduling Strategies for Parallel Processing,* 2015, pp. 178–197. https://dx.doi.org/10.1007/978-3-319-61756-5_10

[22] G. P. Rodrigo, E. Elmroth, P. Ostberg, L. Ramakrishnan, "ScSF: A Scheduling Simulation Framework," *Lecture Notes in Comp. Science,* vol. 10773, 2017. https://dx.doi.org/10.1007/978-3-319-77398-8_9

[23] N. A. Simakov et al., "A Slurm Simulator: Implementation and Parametric Analysis," *Lecture Notes in Comp. Science,* vol. 10724, 2017. https://dx.doi.org/10.1007/978-3-319-72971-8_10

[24] D. Klusacek, M. Soysal, F. Suter, "Alea — Complex Job Scheduling Simulator," *Lecture Notes in Comp. Science,* vol. 12044, 2019. https://dx.doi.org/10.1007/978-3-030-43222-5_19

[25] C. Galleguillos, Z. Kiziltan, A. Netti et al., "AccaSim: a customizable workload management simulator for job dispatching research in HPC systems," *in Cluster Comput.* vol. 23, 2020, pp. 107–122. https://dx.doi.org/10.1007/s10586-019-02905-5

[26] A. Baranov, P. Telegin, B. Shabanov, D. Lyakhovets, "Measure of Adequacy for the Supercomputer Job Management System Model," *in Proc. of the 2019 Fed. Conf. on Computer Science and Information Systems, ACSIS,* vol. 18, 2019, pp. 423–426. https://dx.doi.org/10.15439/2019F186