J Grid Computing manuscript No.
(will be inserted by the editor)

Modeling Allocation Utilization Strategies on Supercomputers

Alexey Poyda · Mikhail Titov · Alexei Klimentov · Jack C. Wells · Sarp Oral · Kaushik De · Danila Oleynik · Shantenu Jha

Received: date / Accepted: date

Abstract In the current era of compute-intensive applications and exa-scale processings, HPCs and supercomputers along with such technologies as grid (HTC) and cloud computing play an important role. Thus the organization of the calculation and execution processes within these instruments require a particular attention. Sharing of computing resources between its clients such as individual users and groups that represent certain projects is determined by predefined usage policies, resource quota per user/group and its dynamic workload based on usage activities. Thus, the load on a supercomputer depends on the number and parameters of computing jobs running there: the number of required nodes, required execution time (walltime), and jobs generation rate. Predefined job parameters such as its number, size, length, rate, are referred as an execution strategy.

The aim of this work is to identify execution strategies geared towards the goal of maximizing the probability of utilization of allocated resources per defined project on a supercomputer in a given time period. Resources allocation is a number of provided cores or computing nodes for a limited time ($cores \times hours$), also mentioned as an allocation time. This work also gives a possibility to estimate a potential resource utilization based on provided job parameters for a given time period and the current supercomputer workload. A simplified model for utilization of allocation time and a simulator based on Queueing Theory were designed. The model was tested on both synthetic and real log data over several months of the supercomputer work (the Titan supercomputer work was examined), and identified strategies were compared with other possible strategies. Experiments conducted using the simulator, showed that in most cases identified strategies increase the probability of utilizing allocation faster than a random choice of job processing parameters.

Keywords Supercomputers · Utilization modeling · Execution strategy · The Titan supercomputer

1 Introduction

Most supercomputers are represented as computational facilities of collective use, providing access to computing resources on a competitive basis. In these conditions an individual user or a project group, with a large quota of allocated resources at the supercomputer, confronts the real task: what strategy to choose in order to utilize this quota successfully. Resource utilization is an actual usage of a partial or full amount of allocated resources within the time that is equal or less to the time for which these resources were allocated, so $ResourceUtilization \leq ResourceAllocation$, ($cores \times hours$). The term “successfully” is understood as an user’s ability to utilize the allocated quota in the range of the requested time. First of all, this is affected

A. Poyda
National Research Centre Kurchatov Institute, Moscow, Russia
E-mail: poyda@wdcb.ru

M. Titov
Lomonosov Moscow State University, Moscow, Russia
E-mail: mikhail.titov@cern.ch

A. Klimentov · S. Jha
Brookhaven National Laboratory, Upton, NY, USA

J.C. Wells · S. Oral
Oak Ridge National Laboratory, Oak Ridge, TN, USA

K. De · D. Oleynik
University of Texas at Arlington, Arlington, TX, USA

S. Jha
Rutgers University, Piscataway, NJ, USA

by a dynamically changing supercomputer load (i.e., number of busy nodes at a certain time), by competition for computing resources with other users and by the local policy of a particular supercomputer which sets the rules for this competition.

Meanwhile, user is able to vary a number of parameters, which would be called as variable parameters, at jobs launch on the supercomputer, which can eventually strongly influence the amount of computing resources consumed during the requested time. In the first place, such parameters include size and length of the job, expressed in requested number of nodes and walltime, respectively, since the values of these parameters can affect the job waiting time in the supercomputer's queue. And this in turn can affect the total number of jobs that will be launched on a supercomputer in a specified time and, as a result, the total amount of consumed resources. A set of specific values of variable parameters defined by a specified user or a project group will be referred to as a job launch strategy for this user or group.

Of course, in some projects, not all variable parameters can change their values. For example, there are projects that can not work with more than one node. But for other projects such choice is possible. In this case, the task of choosing a strategy, which increases probability of successful utilization of a large allocated quota of supercomputer time, becomes relevant.

Finding the strategy that outperforms any arbitrary one is a complex task due to dependence of the corresponding variable parameters on a great number of dynamically varying factors that are difficult to predict because of activities of other users. The task can be simplified by looking for a static execution strategy, which means a strategy with values of variable parameters that do not change over time. Our hypothesis, which should also be tested, is that in most cases the static execution strategy will give a higher probability of successful utilization of a large allocated quota of supercomputer time than, for example, a random dynamic strategy.

There is no only one such outperforming strategy that would work for all supercomputers or will fit all users and project groups needs, since every strategy depends on a particular computing workflow and needs of one who requests this strategy.

Therefore, an effective utilization of allocated resources relies on well-defined execution strategy. Execution strategy should guarantee an achievement of the requested utilization over defined time interval, thus such strategy will maximize the probability of utilizing a particular allocated resources completely. It is also possible that the utilization of the whole allocation time is not feasible, in this case the appropriate execution strategy will facilitate to increase the utilization in compare to no strategy at all. Furthermore, such approach is applicable for the estimation of probable utilization value of potentially allocated resources according to defined job launch parameters, supercomputer load, and chosen execution strategy.

This paper provides key points of design and development of an approach to and technique of finding static strategies of jobs launch for a given project on given supercomputer resources, which increases the probability of successful utilization of a large allocated quota of supercomputer time.

2 Related works

Efforts to increase the efficiency of jobs processings (i.e., to maximize the chance of utilization of the total allocation time) at supercomputers/HPCs could be categorized into the following classes of approaches: i) *queue time predictions*, also could be referred as batch queue prediction; ii) *runtime prediction* or prediction of job execution time; iii) meta-scheduling, which is performed by the corresponding workload management system or service and is responsible for jobs placement according to its internal mechanisms.

One of the latest works related to the area of queue time prediction was presented by Murali and Vadhiyar [1], and is about an integrated adaptive framework, Qespera, used for prediction of queue waiting times on parallel systems. This framework uses algorithm based on spatial clustering for predictions using history of job submissions and executions. Thus, the proposed approach includes such processes as finding similarities between the target job and the history jobs using a weighted distance metric, clustering to characterize the feature neighborhood of the target job based on the calculated distances, calculating the predicted queue waiting time by one of the following methods: an SDM, NN method, and ridge regression. Among the works using the statistical method can be noted the research work of Nurmi et al. [2] about a forecasting system QBETS (Queue Bounds Estimation from Time Series) that generates a predicted bound on the queue waiting time for the target job using a stationary history of previous jobs which have similar quantitative characteristics. The identification of similarity is estimated based on hierarchical clustering algorithm, and queue length-based downtime detection algorithm is used to identify system failures that affect job queuing delay.

Another approaches for efficient resources utilization is to apply forecasting for jobs runtime. One of such approaches was presented by Yang et al. [3] with the proposed observation-based prediction. Authors demonstrated that short partial executions of an application are usable for the prediction generation, while using method approaching cross-platform performance translation based on relative performance between two platforms. This is applicable since most parallel codes are iterative and behave predictably manner after a minimal startup period. Guo et al. [4] propose a data-driven approach for predicting job statuses on HPC systems. The binary classification problem (having either underestimation of runtime or not) is addressed, and machine learning algorithms, such as XGBoost and Random Forests, are applied. Thus, the proposed model can be used to accurately predict whether a job can be completed before its estimated runtime expires.

Meta-scheduling is mostly inherent to Grid computing, but it is expandable into heterogeneous infrastructure and multicluster systems. It uses both metrics estimated from gathered jobs meta-information (e.g., queue waiting time, run-time) and parameters assigned by the system itself or the user (e.g., priorities, min/max processing requirements, etc.). Its goal is to minimize the average job turnaround time in a non-dedicated environment. Work of Lerida et al. [5] presents meta-schedulers for multi-clusters systems with focus on MetaLoRaS, a two-level meta-scheduler that assigns applications (PVM, MPI) according to forecasted turnaround time in each particular cluster. Proposed meta-scheduling techniques take the dynamics of the local workload into account (including job simulation in all clusters that is the base for the prediction algorithm) with further comparison of their effects on system performance. Sotiriadis et al. [6] give an overview of meta-scheduling approaches with focus on inter-cloud schedulers, requirements, topologies, scheduling algorithms, etc. Later in this paper we will introduce a workload management system (Section 4.1.2) which is considered as a meta-scheduler. Its jobs that are predetermined for the execution at a supercomputer are used in analysis to develop the corresponding strategy of their execution at the particular supercomputer.

3 Approach for static strategy selection

3.1 General description

For the proposed approach the following parameters are significant (these parameters, as mentioned earlier, define execution strategy): job size, requested job walltime, job launching scheme (e.g., number of parallel job launching streams, launching interval for consecutive jobs, etc.). Speaking of a static strategy we mean that parameters mentioned above stay constant during the total assessed period of time. In the current implementation of the approach, launching scheme is defined by user, and we assess job size and walltime that better fit for the given scheme.

Static strategy is already an improvement, but has its own limitations, which are characterized by slow reaction on the following changes:

- workload changes;
- resources availability (changes status from online to offline and backward), where time scale over availability of resources is constant.

Of course, if it would be possible to predict workload changes, then the strategy for such expected workload changes would be adapted dynamically, thus move from static to dynamic strategy. But it would require a prediction with high accuracy, which is not presented in this field.

As for resources availability changes, we believe that for large and stable supercomputers (e.g., Titan), commissioning or decommissioning significant amounts of computing resources is quite rare that can be ignored for one or two year time period.

One more restriction for static strategy implementation is large research time interval, which equals to large number of launched jobs and large amount of used allocated computing time. This is necessary to smooth local spikes of key parameters on large time interval. By our estimate, namely hundreds of thousands of core-hours and more.

The designed approach, which is applied for supercomputers (follows job processing scheme presented in Figure 1), consists of the following steps:

- Choose job launching scheme (e.g., number of parallel job launching streams, launching interval for consecutive jobs, etc.), time interval during which the specified strategy will be used, and the total utilization of computing resources that can be achieved.
- Go through all possible combinations of job size and requested walltime, determining probability of achieving the specified disposal in a given time for the chosen job launching scheme for each combination.
- Optional step, repeat previous two steps for other launching schemes.
- Choose as the outperforming static strategy three parameters (job launching scheme, requested number of nodes, requested walltime), which gives the highest probability.

It is almost impossible to check through all possible combinations of job size and requested walltime, since job size can vary from 1 core in 1 node to the maximum allowable value (e.g., it is more than 18 thousands nodes

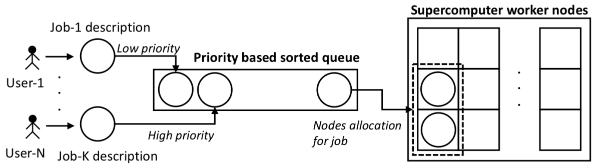


Fig. 1: General scheme of jobs processing at supercomputers

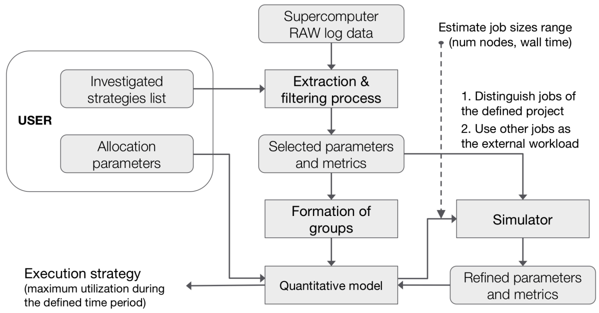


Fig. 2: Workflow of the analysis process

in the Titan supercomputer, more details are in Section 4.1.1), while walltime can vary from several minutes to several days. Therefore, we propose to group each parameter's range into small categories, where jobs from each category can be assumed having similar basic characteristics, such as, for example, utilization per a single job.

In order to determine a probability of achieving a certain utilization in a defined time interval for a specified pair $\{job\_size, requested\_walltime\}$, we have developed a quantitative model. The model assists in calculation of probability of achieving defined utilization during the defined time by jobs of certain size with corresponding walltime (waiting time for a job in the queue is estimated by other parameters). The model allows to set job's size, walltime and queue waiting time as random variables with a given expectation and variance.

To determine parameters of a random variable that specifies queue waiting time for a job of a certain size with corresponding walltime, one can use:

- recorded (historical) data;
- simulated (synthetic) data.

By evaluating recorded (historical) data, it is possible to filter out jobs with similar characteristics and estimate for them queue waiting time. The disadvantage of using only recorded (historical) data is that we cannot take into account changes in system's workload from new jobs, that we will launch on it.

To take into account such changes, as well as to verify calculations of the quantitative model, it is possible to use synthetic data from a simulator of the supercomputer load. We have developed such modeling tool (it will be described in details in Section 3.3).

Figure 2 illustrates the scheme of the designed approach. The user sets the strategy launching schemes; similar jobs are selected from the log for the given schemes; for the selected jobs, the main characteristics are defined: job size, queue waiting time, walltime; if necessary, these characteristics are refined using the simulator mentioned earlier; parameter space is divided into categories and for each category we calculate probability of achieving the specified utilization using the quantitative model; if necessary, utilization calculated for the quantitative model is verified using the simulator.

3.2 Quantitative model of utilization of allocation time

The developed model allows to calculate the probability of achieving the required utilization of allocated resources in a given time by defining specific parameters for jobs processing. Some of these parameters are set by the user (e.g., the number of requested nodes and walltime), while other parameters are determined by the workload of the supercomputer and the actions (i.e., activity) of other users (e.g., the waiting time of the job in the supercomputer's queue before it runs on computing nodes).

The base version of the model assumes that jobs, which utilization is under the estimation, are launched sequentially: every next job arrives to the supercomputer queue only after the previous job has been started to run on computing nodes. Also, the model can be adapted to other schemes of jobs launching. For example, if jobs are launched sequentially according to the scheme that "the every next job enters the queue only after the previous job left it", then the basic model can be used with "the virtual waiting time of the job", which equals to the sum of their real waiting time and their real execution time. If the launching scheme assumes several input streams, e.g., 2-3, then the basic model with one stream can be used, but the defined time for calculated utilization will be reduced by 2-3 times respectively. A formal description of the input and output data for the basic model is presented below.

Given assumptions:

- Jobs J of project Pr;
- Jobs J require N nodes, where N is a random variable with expected value μ_N and variance σ_N^2;

- Jobs J require walltime E, where E is a random variable with expected value μ_E and variance σ_E^2;
- Execution times of jobs J equal to their walltime values;
- Duration of waiting time in the queue for jobs J is described by a random variable Q with expected value μ and variance σ^2;
- Jobs J come into the supercomputer queue sequentially: next job is allocated to the queue after the previous one has left the queue to computing nodes.

Values to find: $P(U > U_0)$ - the probability that utilization U during the time interval T_0 will exceed the predefined value U_0, where T_0 is big.

The derivation process is presented in the appendices (Appendix A), and here is the final equation that describes the quantitative model:

$$P(U > U_0) = \sum_{n=100}^{\infty} \left[\int_{U_0}^{\infty} f(x, n\mu_U, n\sigma_U^2)dx \times \left(\int_{-\infty}^{T_0} f(x, n\mu, n\sigma^2)dx - \int_{-\infty}^{T_0} f(x, (n+1)\mu, (n+1)\sigma^2)dx \right) \right] \quad (1)$$

The outcome of the Equation 1 is the probability that the utilization of resources, which is achieved by a sequential set of processed jobs using capabilities of the supercomputer during the time interval T_0, is greater than the predefined value U_0. It implies that:

- Utilization of every single job is described by a random variable with the expected value μ_U and the variance σ_U^2;
- The time interval between launches of sequential jobs is described by a random variable with the expected value μ and the variance σ^2.

3.3 Simulator

This designed analysis tool is aimed to simulate the workload on a supercomputer and to produce job traces for a given workload, as well, it is used for the quantitative model validation and adjustment. There are two modes to run the simulator: i) set operational parameters, such as job generation rate and job execution rate, to produce synthetic data only; ii) use historical data for key parameters of the job, such as timestamp of job arrival to the queue and job execution time, to produce simulated data of real job processing lifecycle.

The simulator is based on Queueing Theory [7] and according to the Kendall's Notation [8] for queues, usually referred as A/B/C/D/E, it is characterized as following:

- A -*arrival process* is represented by streams that are responsible for job generation and is described either by a Poisson process or by a deterministic model;
- B - *service/server process* is represented by a set of nodes that simulate job execution process and is described either by a Poisson process or by a deterministic model as well;
- C - *number of servers* that corresponds to the number of computing nodes (in terms of the Titan supercomputer);
- D - *capacity of the queue or system overall*, which is "on", if the queue limit is set (either per stream or for the total number of jobs in the queue) and queue buffer is not used, otherwise the capacity is unlimited;
- E - *queueing discipline* is provided in two options: FIFO or Priority.

Some of the parameters are set as requirements and restrictions applied to a specific supercomputer and its policy, e.g., the total number of computing nodes that are available for computing jobs, the limit of the number of jobs in the queue per user/group, etc.

3.3.1 Simulator description

Implementation of the simulator (Queueing System Simulator) [9] was done by using Python[1], and the following key classes and generators were designed (to emulate internal supercomputer processes):

- *Job* - contains parameters to describe job's processing lifecycle, such as arrival timestamp, start execution timestamp, completion timestamp that is based on walltime / execution time along with the previous parameter, number of required nodes for its execution, stream (i.e., source name), label (i.e., project name), priority and priority group name;
- *Stream* - generates jobs with the predefined parameters as an input for the simulator;
- *Queue Manager* - emulates buffer ahead of the queue, the queue itself, and manages jobs while waiting for their execution, it lets to define the queue discipline such as FIFO and Priority, and set the limits per input job stream;
- *Schedule Manager* - emulates a backfill mode - gets information about job sizes, assigns corresponding nodes for execution, gives a schedule when each job starts to be executed;
- *QSS* - the core class that manages and tracks job's processing lifecycle.

[1] High-level programming language Python, `https://docs.python.org/2.7/` [accessed on 2019-04-15]

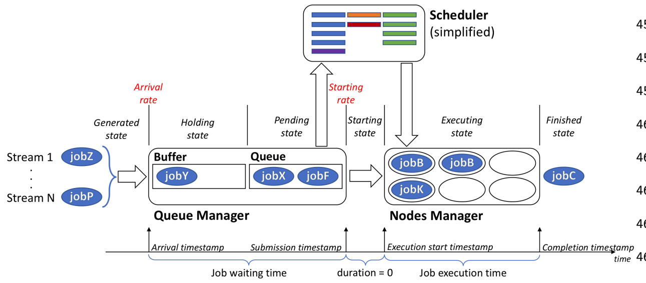


Fig. 3: Job state transitions at the simulator

3.3.2 State model

Job lifecycle (in terms of the simulator, Figure 3) includes the following states: Generated - Holding (i.e., Titan notation: blocked) [buffer] - Pending (i.e., Titan notation: eligible-to-run) [queue] - Starting - Executing - Finished. Some of the states might be skipped if certain components are turned off (e.g., if the queue buffer is not used then there is no state "holding") or if there is some initial restriction (e.g., state "starting" is neglected, since the assumption that job execution starts right after it leaves the queue).

3.4 Testing and validation

3.4.1 Model validation using utilization estimation

The validation of the designed quantitative model was conducted based on a simplified model for which utilization is computed using theoretical calculations. The following conditions have been applied:

- Starting rate of jobs is defined as a random variable with distribution denoted as SRD;
- Execution time of jobs is defined as a random variable with distribution denoted as ETD;
- Number of nodes used (requested) by examined jobs is defined as a random variable with distribution denoted as NND.

With this model it is possible to estimate the expected utilization achieved in a given long period of time. The expected utilization $U(t)$ depends on particular values of the random values in it, therefore it is mutable. But for big numbers of the time t it is possible to take advantage of the law of large numbers (LLN). In this case, it can be argued that the value of $U(t)$ will tend to the same value, regardless of particular values of the random variables within it.

The theoretically calculated utilization for large values of t can be estimated by the following equation:

$$U(t) \approx t \times E(SRD) \times E(ETD) \times E(NND) \tag{2}$$

where $E(x)$ is the expected value (mathematical expectation) of a random variable. The next step was to compare the outcome of Equation 1 (model) and Equation 2 (theoretical estimation). Thereby, the expected value was calculated approximately, while going from the distribution function of a random variable to the utilization derivation.

Parameters of the experiment:

- Starting rate is a random variable with the Poisson distribution, the event rate (i.e., rate parameter and considered as an expected value) $\lambda = 100$;
- Execution time is a random variable with the Normal distribution, mean of the distribution $\mu = 4$;
- Number of nodes used by a single job is a random variable with the Poisson distribution, $\lambda = 8$;
- Examined time interval is half a year ≈ 4320 *hours*.

Results of the experiment: the utilization calculated using Equation 1 is equal to $13,822,864$, while the utilization calculated using Equation 2 (theoretical, based on expected values of SRD, ETD, NND only) is equal to $13,824,000$. Thus, the results are very close, and the little difference is due to approximations related to conducted computations.

3.4.2 Model validation based on mathematical calculations and synthetic data from the simulator

The further analysis of the quantitative model and the simulator is a simplified validation process that demonstrates equal results of both with the same input data.

The following common parameters are chosen:

- the total number of nodes is equal to 1;
- expected value (μ) and variance (σ^2) for random parameters of job's waiting and execution times respectively are equal to 1;
- the total processing time is equal to 5000 time units (hours).

Simulation process follows additional specific parameters:

- job waiting time is defined according to the Poisson distribution;
- job execution time is defined according to the Normal distribution;
- job launching scheme: one stream and there is always one job in the queue;
- the total number of simulation runs is equal to 100.

Figure 4 shows the plot with two lines that represent the probability that a given utilization will be achieved in a given time interval (that is defined by the total processing time): the blue line corresponds to the results obtained using the simulator, while the red line corresponds to calculations with the quantitative model.

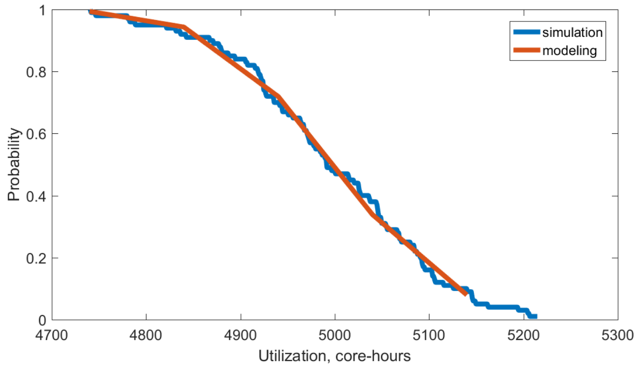


Fig. 4: Probability (axis Y) that utilization will reach the corresponding utilization value (axis X) during the time of 5000 hours

4 Experiments

4.1 Background of the study

4.1.1 The Titan supercomputer

One of the supercomputers that we consider as a use case for modeling allocation utilization is Titan that is located at the Oak Ridge Leadership Computing Facility (OLCF) in the Oak Ridge National Laboratory (USA). Titan[2] is a hybrid-architecture Cray XK7 system that contains both CPUs (16-core AMD Opteron) and GPUs (NVIDIA Kepler). It features 18,688 compute nodes, a total system memory of 710 TB, and Cray's high-performance Gemini network. Titan's theoretical peak performance exceeding 27 petaFLOPS.

The general overview of jobs processing at the Titan supercomputer is presented by the following plots (Figures 5, 6) that demonstrate metrics such as waiting and execution times, as well as requested and eventually used nodes per job (Figure 7) during the defined period of time.

Quantitative characteristics of the presented plots are the following:

- waiting time (hours) - mean=6.21, std=29.77;
- execution time (hours) - mean=0.61, std=1.51;
- number of nodes per job - mean=135.07, std=712.31.

These numbers give a general overview of jobs key characteristics while processing at the Titan supercomputer, and which are compared with the simulation results (outcome of the simulator).

[2] The Titan supercomputer, `https://www.olcf.ornl.gov/olcf-resources/compute-systems/titan/` [accessed on 2019-04-15]

Fig. 5: Waiting times for jobs in the queue at the Titan supercomputer (562,079 computing jobs from May 2017 to April 2018)

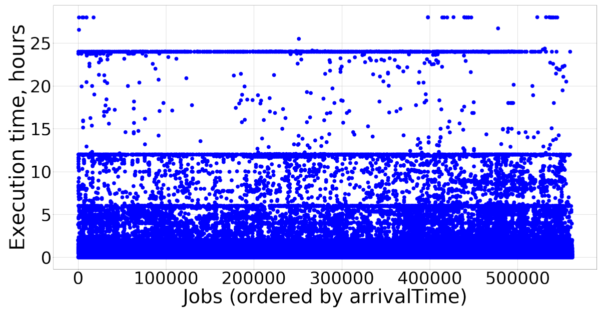


Fig. 6: Execution times for jobs processing at the Titan supercomputer (562,079 computing jobs from May 2017 to April 2018)

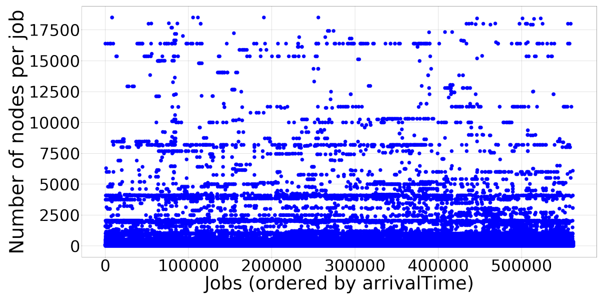


Fig. 7: Number of nodes per job at the Titan supercomputer (562,079 computing jobs from May 2017 to April 2018)

4.1.2 Production and Distributed Analysis system PanDA

The PanDA (Production and Distributed Analysis) system is a workload management system (WMS) for job scheduling on the distributed computational infrastructure [10], it federates hundreds of heterogeneous computing resources (including Grid, supercomputers, and public and private computing clouds) into a unique job submission system. It was originally developed for US physicists and adopted as the ATLAS [11] wide WMS in 2008 (in use for all computing applications of the ATLAS experiment at the Large Hadron Collider).

Key features of PanDA are the following: i) pilot-based job execution system with late binding (i.e., there is a lightweight process scheduled on computing nodes that interacts with the core to schedule computing jobs); ii) central job queue; iii) fair-share or policy driven priorities for thousands of users and hundreds of resources; iv) automated brokerage based on CPU and storage resources.

PanDA started to use Titan as one of its resources several years ago under the project of integration with it by enhancing with tools and methods relevant to work on HPC [12]. Thus, the pilot runs on Titan's data transfer nodes (DTNs) and submits corresponding payloads to the worker nodes. It uses the local job scheduling and management system (Moab) via the SAGA (Simple API for Grid Applications) interface [13] for monitoring and management of PanDA jobs running on Titan's worker nodes.

In 2017, under the ALCC[3] program it was allocated computational resources (computing hours) at the OLCF supercomputer Titan for ATLAS payload via PanDA.

4.2 Simulator with Titan log data

The following parameters correspond to the Titan supercomputer: 1) number of nodes is 18,688; 2) the limit of jobs in the queue per stream is 4; 3) the queue buffer is on - that corresponds to no dropped jobs.

Obtained Titan log data (for the period from May 2017 to April 2018) were used in simulator to test it. Certain job parameters were used for deterministic models for arrival and service processes in the simulator, thus Titan log data provided the following jobs characteristics: arrival timestamp (timestamp of queueing), number of requested nodes per job, real execution time. Also, there are restrictions and requirements applied to the queue (according to the Titan Scheduling Policy[4]): i) priority discipline in the queue - job's age in the queue increases its priority accordingly; ii) there are 5 groups of jobs (in Titan notation: bins) according to job size (requested wall time and the number of nodes), and several of that groups have initial priority; iii) every stream (in Titan notation: user) has the limit of 4 jobs in the queue, if the limit is reached then jobs stay in the queue buffer. Figure 8 shows jobs waiting times taken from Titan logs and from Simulator logs. This shows that the simulator is not able to reconstruct the exact workflow of the supercomputer, but gives a certain estimations about jobs processing.

Figure 9 shows the load of the simulated nodes (i.e., the number of busy nodes at every simulated time unit, seconds) during the simulation process of jobs described earlier. The average utilization rate of the set of service nodes is 88.2 %.

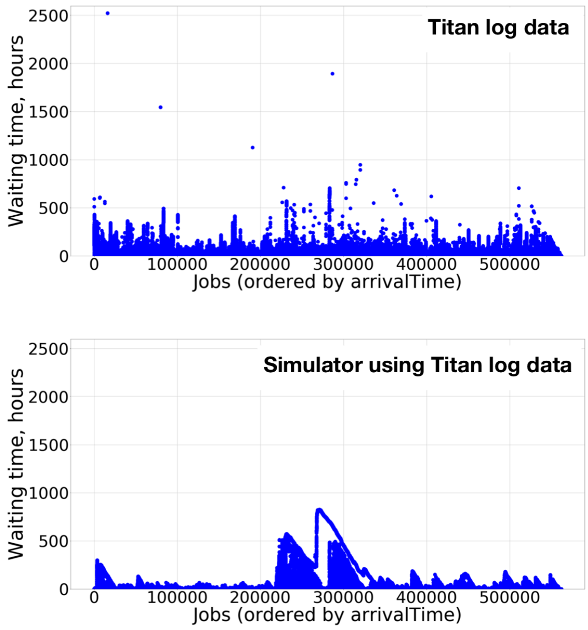


Fig. 8: Job waiting times based on Titan log data and Simulator log data with initial parameters from Titan log data (waiting time, hours - Titan: mean=6.21, std=29.77; Simulator: mean=30.51, std=97.34)

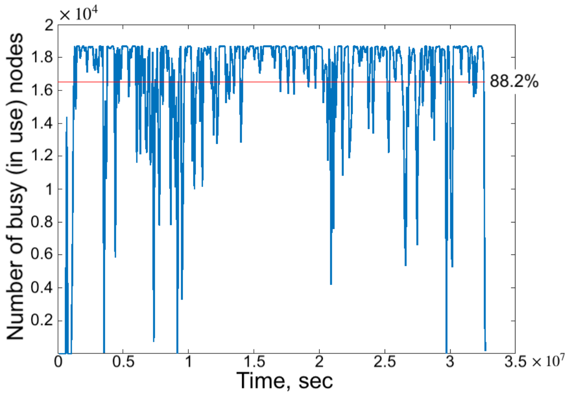


Fig. 9: The load of simulated service nodes (18,688 nodes)

[3] ALCC: the ASCR (Advanced Scientific Computing Research) Leadership Computing Challenge, `https://science.energy.gov/ascr/facilities/accessing-ascr-facilities/alcc/` [accessed on 2019-04-15]

[4] Titan Scheduling Policy, `https://www.olcf.ornl.gov/for-users/system-user-guides/titan/titan-user-guide/#titan-scheduling-policy` [accessed on 2019-04-15]

4.3 Analysis of Titan log data

Conducted experiments are based on obtained the Titan supercomputer log data that were used for the quantitative model. The following information was extracted from the log: i) job arrival timestamp (time when the job was queued); ii) execution start timestamp (i.e., startTime); iii) completion timestamp (i.e., endTime); iv) the number of requested nodes (1 node = 16 cores in Titan); v) requested walltime.

As the first approach in analysis there were no separation of jobs from different projects, jobs were grouped only based on their sizes. Further analysis was applied on jobs from just one project to extend the potential applicability of our model.

4.3.1 Analysis using log data of all projects

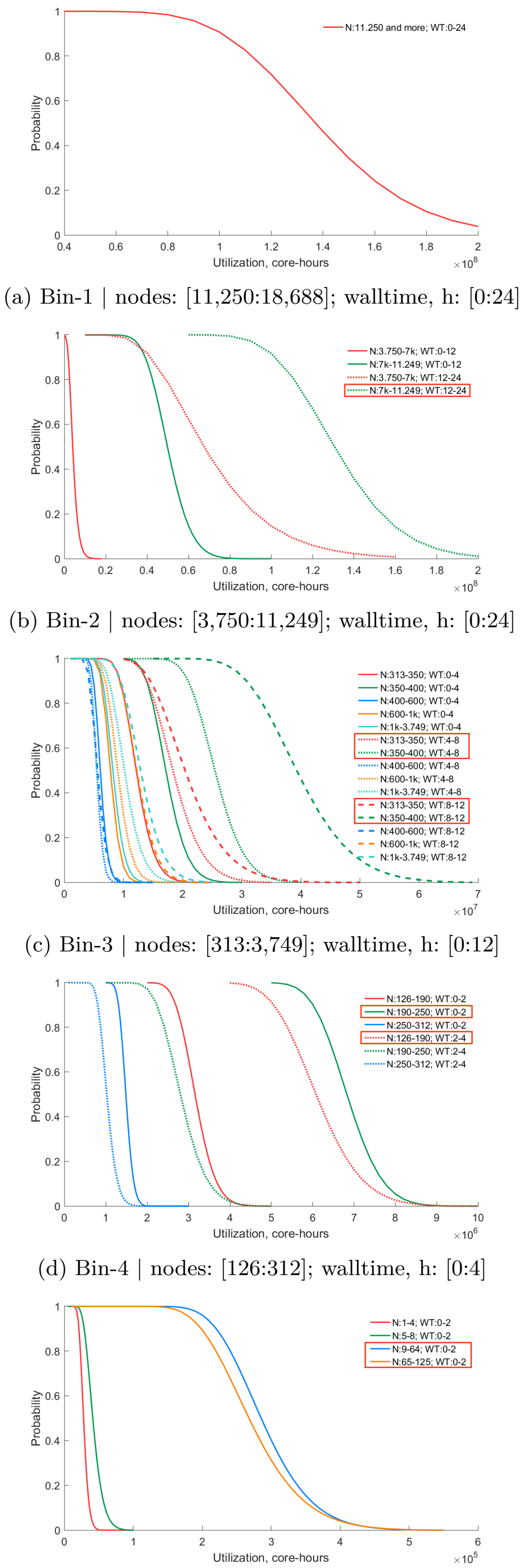
The following analysis actions are applied:

- all jobs are divided into categories according to the number of required nodes and the volume of walltime requested (every category corresponds to a particular Titan's bin, where bin is a group of jobs that are treated equally);
- for each category the following values are calculated: the expected value and variance of random variables describing waiting time in the queue and the utilization achieved by a single job;
- obtained values were used as input data in equation for the quantitative model to calculate the probability that jobs of a given category will be able to utilize provided allocation in 3 months;
- job launching scheme: one stream and there is always one job in the queue.

The outcome of the performed analysis is presented by the set of 5 plots (one per Titan's bin) in Figure 10 (outperformed groups are highlighted at the legend). Log data was collected for the period from May 2017 to April 2018.

4.3.2 Analysis using log data of one project (HEP110)

Project HEP110 is associated with ALCC program, and computing jobs running under this project are from PanDA for the ATLAS experiment at LHC, CERN (actual ATLAS data wasn't in use for the current analysis, neither any data from PanDA). The outcome of the performed analysis for the project HEP110 is presented by the set of 2 plots (using jobs "execution time" and "walltime"/"requested processing time") in Figure 11.



(a) Bin-1 | nodes: [11,250:18,688]; walltime, h: [0:24]

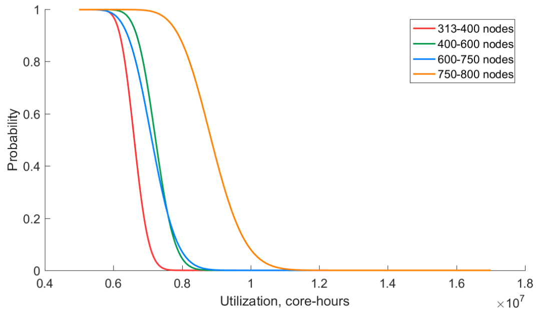
(b) Bin-2 | nodes: [3,750:11,249]; walltime, h: [0:24]

(c) Bin-3 | nodes: [313:3,749]; walltime, h: [0:12]

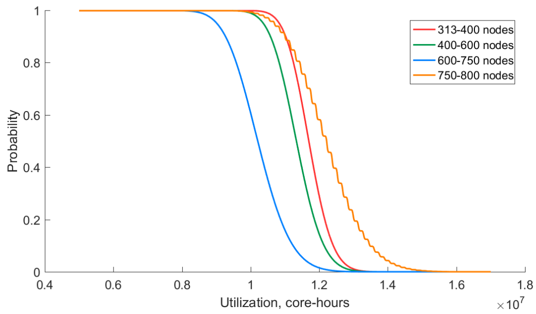
(d) Bin-4 | nodes: [126:312]; walltime, h: [0:4]

(e) Bin-5 | nodes: [1:125]; walltime, h: [0:2]

Fig. 10: Probability distributions of utilization of allocation time during 3 months per Titan's bins (based on Titan log data for 12 months)



(a) Based on provided real execution time



(b) Based on requested wall time

Fig. 11: Probability distributions of utilization of allocation time during 3 months for HEP110 project at the Titan supercomputer (based on Titan log data for 6 months)

5 Conclusion and discussion

Developed tools provide possibility to adjust jobs parameters to regulate and improve the probability of utilizing a given allocation for a given project. Further work for the proposed approach improvement includes reinforcement of applied requirements (i.e., decrease the number of applied assumptions for the developed model and simulator). The model and simulator are preliminary and require further tuning, to understand the accuracy and sensitivity (to initial conditions, training duration, workload types). This early work will be extended to consider different kinds of workflows as well as different types of workloads of heterogeneous resources.

Acknowledgements

References

1. P Murali and S Vadhiyar, Qespera: an adaptive framework for prediction of queue waiting times in supercomputer systems, Concurrency and Computation: Practice and Experience, 28, 2685–2710 (2016)
2. D Nurmi, J Brevik, R Wolski, Job Scheduling Strategies for Parallel Processing (Lecture Notes in Computer Science, 4942), 76–101. Springer, Berlin, Heidelberg (2008)
3. L T Yang, X Ma, F Mueller, Cross-Platform Performance Prediction of Parallel Applications Using Partial Execution, SC05: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing (2005)
4. J Guo, A Nomura, R Barton, H Zhang, S Matsuoka, SCFA 2018: Supercomputing Frontiers (Lecture Notes in Computer Science, 10776), 179–198. Springer, Cham (2018)
5. J L Lérida, F Solsona, F Giné, M Hanzich, J R García, P Hernández, EuroPVM/MPI 2007: Recent Advances in Parallel Virtual Machine and Message Passing Interface (Lecture Notes in Computer Science, 4757), 195–203. Springer, Berlin, Heidelberg (2007)
6. S Sotiriadis, N Bessis, N Antonopoulos, Towards Inter-cloud Schedulers: A Survey of Meta-scheduling Approaches, International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 59–66 (2011)
7. R B Cooper, Introduction to Queueing Theory, 2nd Ed. Elsevier/North-Holland (1981)
8. D G Kendall, Some Problems in the Theory of Queues, Journal of the Royal Statistical Society. Series B (Methodological), 13 (2), 151-185 (1951)
9. M Titov et al. Queueing System Simulator (QSS) [software], `https://github.com/ATLAS-Titan/allocation-modeling` [accessed on 2019-04-15]
10. T Maeno et al, PanDA for ATLAS distributed computing in the next decade, Journal of Physics: Conference Series, 898, 052002 (2017)
11. G Aad et al [ATLAS Collaboration] The ATLAS Experiment at the CERN Large Hadron Collider, JINST, 3, S08003 (2008)
12. F Barreiro Megino et al, Integration of Titan supercomputer at OLCF with ATLAS Production System, Journal of Physics: Conference Series, 898, 092002 (2017)
13. T Goodale et al, SAGA: A Simple API for Grid Applications. High-level application programming on the Grid, Computational Methods in Science and Technology, 21, 7–20 (2006)

A Derivation of the quantitative model

Given assumptions:

1. Project Pr;
2. Jobs J of project Pr;
3. Jobs J require N nodes, where N is a random variable with expected value μ_N and variance σ_N^2;
4. Jobs J require walltime E, where E is a random variable with expected value μ_E and variance σ_E^2;
5. Execution times of jobs J equal to their walltime values;
6. Jobs J come into the supercomputer queue sequentially: next job is allocated to the queue after the previous one has left the queue to computing nodes;
7. Duration of waiting time in the queue for jobs J is described by a random variable Q with expected value μ and variance σ^2.

Values to find: $P(U > U_0)$ - the probability that utilization U during the time interval T_0 will exceed the predefined value U_0, where T_0 is big.

Solution. Using the Law of total probability we can write the following equation:

$$P(U > U_0) = \sum_{n=1}^{\infty} P(U(n) > U_0)P(n) \tag{3}$$

where $U(n)$ is a random variable for utilization achieved by n sequential jobs J; $P(n)$ is a probability that during the time T_0 exactly n jobs J will be running. We assume that T_0 is big, so values of $P(n)$ for the first several n (e.g., for n from 1 to 99) are equal to zero, thus the sum would start with $n = 100$.

To calculate $P(n)$ let's consider a random variable $T(n)$ describing the time required to run n sequential jobs J. Taking into account one of the assumptions (6) we can write:

$$T(n) = \sum_{i=1}^{n} Q_i \tag{4}$$

where $Q_i = Q$ is a random variable describing duration of waiting time in the queue for the job J_i. And, using Central limit theorem we can write for big values of n:

$$\sum_{i=1}^{n} Q_i \approx N(n\mu, n\sigma^2) \tag{5}$$

where $N(\mu, \sigma^2)$ is a normal distribution with expected value μ and variance σ^2; μ is an expected value of a random variable Q describing duration of waiting time in the queue for jobs J; σ^2 is a variance of a random variable Q describing duration of waiting time in the queue for jobs J.

The probability $P(n)$ can be rewritten as following:

$$P(n_0) = P(n \geq n_0) - P(n \geq n_0 + 1) \tag{6}$$

where $P(n_0)$ is a probability that during the time T_0 precisely n_0 jobs J will be running; $P(n \geq n_0)$ is a probability that during the time T_0 not less than n_0 jobs J will be running. We can mention that the event $n \geq n_0$ (during the time T_0 not less than n_0 jobs J will be running) is equal to the event $T(n_0) \leq T_0$ (the time required to run n_0 jobs J is less than or equal to the time T_0). Considering that we have: $P(n \geq n_0) = P(T(n_0) \leq T_0)$ and $P(n \geq n_0 + 1) = P(T(n_0 + 1) \leq T_0)$. Thus, inserting these equations into Equation 6 and using Equations 4 and 5 we can have the following:

$$\begin{aligned} P(n_0) = {} & P(N(n_0\mu, n_0\sigma^2) \leq T_0) - \\ & P(N((n_0+1)\mu, (n_0+1)\sigma^2) \leq T_0) \end{aligned} \tag{7}$$

This equation can be updated by using the probability density of the normal distribution:

$$\begin{aligned} P(n_0) = {} & \int_{-\infty}^{T_0} f(x, n_0\mu, n_0\sigma^2)dx - \\ & \int_{-\infty}^{T_0} f(x, (n_0+1)\mu, (n_0+1)\sigma^2)dx \end{aligned} \tag{8}$$

where $f(x, \mu, \sigma^2)$ is a function of probability density of the normal distribution $N(\mu, \sigma^2)$.

To calculate value of $P(U(n) > U_0)$ in Equation 3 let's write a random variable $U(n)$ as a sum of random variables U_i describing utilization of the single job J_i: $U(n_0) = \sum_{i=1}^{n_0} U_i$. Assuming that all random variables U_i have the same expected values (let's denote them as μ_U) and variances (let's denote them as σ_U^2) and using as before the Central limit theorem we can write for big values of n:

$$U(n_0) = \sum_{i=1}^{n_0} U_i \approx N(n_0\mu_U, n_0\sigma_U^2) \tag{9}$$

Using the probability density of the normal distribution and Equation 9 we can write $P(U(n) > U_0)$ in a way:

$$P(U(n) > U_0) = \int_{U_0}^{\infty} f(x, n\mu_U, n\sigma_U^2)dx \tag{10}$$

To get the final equation we insert Equations 8 and 10 into the Equation 3:

$$\begin{aligned} P(U > U_0) = \sum_{n=100}^{\infty} \Bigg[& \int_{U_0}^{\infty} f(x, n\mu_U, n\sigma_U^2)dx \times \\ & \bigg(\int_{-\infty}^{T_0} f(x, n\mu, n\sigma^2)dx - \\ & \int_{-\infty}^{T_0} f(x, (n+1)\mu, (n+1)\sigma^2)dx \bigg) \Bigg] \end{aligned} \tag{11}$$

The outcome of the Equation 11 is the probability that the utilization of resources, which is achieved by a sequential set of processed jobs using capabilities of the supercomputer during the time interval T_0, is greater than the predefined value U_0. It implies that:

- Utilization of every single job is described by a random variable with the expected value μ_U and the variance σ_U^2;
- The time interval between launches of sequential jobs is described by a random variable with the expected value μ and the variance σ^2.

Generally, the distribution of a random variable that describes the size of the job in a way as the number of occupied cores (denote this random variable as N) and the distribution of a random variable that describes the time to complete for a single job (denote this random variable as E) are more commonly known compare to the distribution of a random variable that describes the utilization achieved by a single job (denote this random variable as U). Since, the utilization of a single job is a product of the amount of the used resources by the time to complete this job, then, in case of mutual independence of random variables N and E, the values μ_U and σ_U^2 can be found by the following equations:

$$\mu_U = \mu_N \mu_E \tag{12}$$

where μ_N is the expected value of the random variable N; μ_E is the expected value of the random variable E.

$$\sigma_U^2 = \sigma_N^2\sigma_E^2 + \mu_N\sigma_E^2 + \mu_E\sigma_N^2 \tag{13}$$

where σ_N^2 is the variance of the random variable N; σ_E^2 is the variance of the random variable E.