Optimization of Resources Allocation in High Performance Computing under Utilization Uncertainty

Victor Toporkov [0000-0002-1484-2255], Dmitry Yemelyanov [0000-0002-9359-8245] and Maksim Grigorenko

National Research University "MPEI", ul. Krasnokazarmennaya, 14, Moscow, 111250, Russia
{ToporkovVV, YemelyanovDM, GrigorenkoMO}@mpei.ru

Abstract. In this work, we study resources co-allocation approaches for a dependable execution of parallel jobs in high performance computing systems with heterogeneous hosts. Complex computing systems often operate under conditions of the resources availability uncertainty caused by job-flow execution features, local operations, and other static and dynamic utilization events. At the same time, there is a high demand for reliable computational services ensuring an adequate quality of service level. Thus, it is necessary to maintain a trade-off between the available scheduling services (for example, guaranteed resources reservations) and the overall resources usage efficiency. The proposed solution can optimize resources allocation and reservation procedure for parallel jobs' execution considering static and dynamic features of the resources' utilization by using the resources availability as a target criterion.

Keywords: Computing, Grid, Resource, Scheduling, Uncertainty, Dynamic, Availability, Probability, Job, Allocation, Optimization

1 Introduction

Today, Grid and cloud computing systems are used universally. Due to their commercial reach and low entry threshold, they attract users with different technical skills, who solve a wide range of computational tasks (time- and volume-wise) and require different quality of service.

It usually takes certain economic costs to build and manage the necessary computing infrastructure, including the purchase and installation of equipment, the provision of power supply, and user support. Thus, when a budget for job performance is limited, it becomes important to allocate suitable resources efficiently in accordance with both technical specification and a constraint on the total cost [1-6].

The system's resources may include computational nodes, storage devices, data communication links, software, etc. Each resource has a set of characteristics, their values determine its suitability for performing a specific job. Generally, computational nodes have the widest set of characteristics. For example, a virtual machine is the main computing resource in the commonly used CloudSim simulator [2, 3], its char-

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77980-1_41

acteristics include overall performance, number of CPU cores, size of RAM and disk memory, bandwidth limit of the data link.

It is worth mentioning the dynamic utilization issue of available resources and computational nodes at time. High performance and distributed computing systems (HPDCS) are the dynamic systems, in which the following processes take place: execution of parallel jobs from multiple users, utilization with local jobs, maintenance works, a physical shutdown of nodes (both scheduled and unscheduled). To procure the reliability and dependability of such systems, an advance allocation mechanism is used [5-8]. This mechanism allows one to pre-allocate resources for a specific job and, thereby, prevents possible contention between jobs. Thus, a utilization schedule for each resource can be obtained: a list of utilization intervals (allocated time, scheduled maintenance, and outages) and downtime periods. Downtime periods can be used to perform other jobs and to allocate the resources for the execution of user jobs.
The problem of scheduling and co-allocating resources for executing parallel jobs in a distributed computing system with non-dedicated resources is stated as follows.

- The set R of the computing system resources, as a rule, is heterogeneous and includes resources r_i of several types with different sets of characteristics C_i. The values of these characteristics for the resources of the same type may also differ. Among the most important characteristics of a resource, one can single out its performance, which affects the execution time of a job, as well as the cost required to allocate the resource. Besides, at any specific time, some subsets of the resources may be unavailable for a user job. Therefore, available resources, as a rule, are represented in classical models as a set of slots - intervals of availability of each resource [5-8].
- Resources co-allocation for a parallel job execution typically requires selection (allocation) of a set of resources with types and characteristics defined by the user who is running the job. The resource request for the job execution includes the number of concurrently required resources n, the minimum suitable values of the characteristics Ch_i, the volume of task V (the number of calculations/instructions) or the ordered resource allocation time T, as well as the total execution budget C [1-8].

However, as a rule, the structure and specifics of submitted jobs in HPDCS imply some uncertainty, primarily in the execution time and load of the allocated resources. So, users can only roughly estimate the execution time of their jobs, while special expert systems for predicting the execution time of user programs or the level of resource load (based on the use of machine learning, statistics, and big data) present the results in the form of probabilities of outcomes [4, 9-14].

In this paper, we propose proactive algorithm for resources allocation and reservation in heterogeneous market-based computing environments considering static and dynamic resources availability uncertainties. The uncertainties are formalized with the availability probability functions as a natural way of statistical and machine learning predictions presentation. The novelty of the proposed solution is in general knapsack-

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77980-1_41

based resources selection procedure performing resources availability maximization according to the parallel job requirements.

The paper is organized as follows. Section 2 presents a brief overview of works, related to the jobs execution uncertainties and probabilities in parallel computing environments. Section 3 presents a formal model of the resources' utilization and a general procedure for the dynamic resources' allocation optimization. Additional details are provided for the subset selection and time scan algorithms. Section 4 provides details about the simulation experiment setup, simulation results and analysis. Section 5 summarizes the paper and describes further research topics.

2 Related Works

Existing systems of distributed computing usually perform resources allocation and distribution based on deterministic models of the resource scheduling [1-3, 5-7]. As a result, the expected efficiency and accuracy of such scheduling methods are reduced due to unforeseen resource events (failures, maintenance works), inaccurate estimates and predictions of the jobs' characteristics and execution times. Late job completion time requires rescheduling of all the subsequent jobs or shutting down the job with possible loss of results. Early release of resources also requires rescheduling to minimize the resource downtime. For example, according to an existing approach [8], a scheduler may double the user's runtime estimates to improve the efficiency of the job flow.

Many such systems rely on the reactive approach [4, 9, 10], when an actual state of the computing environment is analyzed, and the appropriate migration and rescheduling decisions are made on the fly. However, these rescheduling and migration operations incur additional time, cost, and network losses. Thus, proactive algorithms, which concentrate on the resource utilization predictions and advanced resources allocations may improve the overall resources usage efficiency.

In [4] a simple uncertainty-based scheduling approach is proposed for a workflow job execution. Concepts of deadline, budget and execution surety are defined to choose the Pareto optimal set of the schedules, satisfying the user requirements. The task execution surety parameter is provided for each available resource by their owners/administrators.

Paper [11] discusses the problem of scheduling a flow of sequential jobs with the execution time uncertainties. Different resources allocation strategies are studied to minimize the total execution time based on the runtime probabilities of the queued jobs. The jobs' execution times are modeled as self-similar heavy-tail processes. In [12] a single-machine scheduling model is proposed to minimize a total flowtime of jobs with processing times characterized by normally distributed random variables. In [13], a set of distinct availability states is defined to model resource behavior and probabilities state transitions

In [14] we studied the problem of a static resource co-allocation for a parallel job execution in a computing environment with utilization uncertainties. Similarly to [4] we used concepts of the execution time deadline, cost limit (budget) and the probabil-

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77980-1_41

ity of a successful execution as a target optimization criterion. We used a knapsack-based algorithm to maximize an aggregate availability probability of a set of simultaneously allocated resources with the corresponding time and cost constraints. However, in this static scenario, the resources' availability probabilities are modeled as simple normally distributed estimates at the given static moment of time.

Current paper extends [14] by studying the dynamic variation of a resources co-allocation problem during some scheduling interval: when the parallel job may be executed at any time inside the given interval. For this purpose, for each independent resource we use heavy-tail distribution to model utilization uncertainties caused by inaccurate estimates in other jobs' execution runtimes. Additional scheduling optimization methods are proposed and analyzed to handle the emerging time scan problem.

3 Resource Selection Algorithm

3.1 Resources Utilization Model

We consider a set R of heterogeneous computing nodes with different performance p_i and price c_i characteristics.

The probabilities (predictions) of the resource's availability and utilization for the whole scheduling interval L are provided as input data. Dynamic job execution uncertainties are modeled as a sequence of *allocation*, *occupation* (actual execution) and *release* events with the *occupation* probability $P_o \leq 1$. Global (static) resources utilization uncertainties, such as maintenance works or network failures, are modeled as a continuous *occupation* events with $P_o << 1$ during the whole considered scheduling interval.

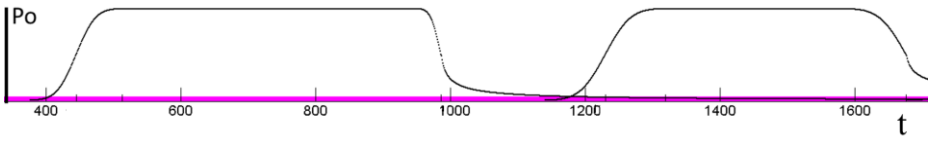

Fig. 1. Example of a resource utilization probability schedule.

Fig. 1 shows an example of a single resource occupation probability P_o schedule. With two jobs already assigned to the resource, there are two resources allocation events (with expected times of allocation at 445 and 1230 time units), two resources occupation events (starting at 513 and 1319 time units) and two resources release events (expected release times are 986 and 1676 time units respectively). Gray translucent bar at the bottom of the diagram represents a sum of global utilization events with a total resource occupation probability $P_o = 0.05$.

A detailed analysis of the main utilization characteristics of real HPDCS systems was made to design and simulate an adequate resources utilization model. As the basis for modeling the availability and utilization probability of computational nodes, the log files of the ForHLR II supercomputer from the Karlsruhe Institute of Technol-

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77980-1_41

ogy in Germany were taken for the analysis [15, 16]. The available files contain information on the execution of jobs from June 2016 to January 2018.

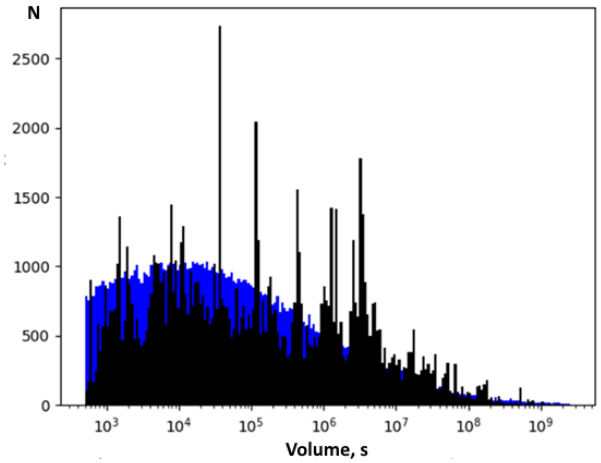

Fig. 2. Job size distributions of real (black) and simulated (blue) job-flows.

After carrying out many experiments, the normal distribution on a logarithmic scale (lognormal) was chosen as the most suitable for modeling the jobs' length and size characteristics. The main parameters of the distribution (mathematical expectation and variance) were selected experimentally to achieve an acceptable accuracy. As a result, the generated distribution by form largely replicates the original one (Fig. 2). More formal comparison gives 0.14 value by the Kolmogorov - Smirnov test.

Thus, the resources *allocation* events are modeled by random variables with a normal distribution. Resources *release* events are modeled with lognormal distribution and expose heavy tails [11, 16]. Expected allocation and release times are derived from the job's replication and execution time estimations.

3.2 Resources Allocation under Uncertainties

To execute a parallel job a set of simultaneously idle nodes (a window) should be allocated ensuring user requirements from the resource request. The resource request usually specifies number n of nodes required simultaneously, their minimum applicable performance p, job's total computational volume V and a maximum available resources allocation budget C.

These parameters constitute a formal generalization for resource requests common among distributed computing systems and simulators [2, 5, 7].

Common allocation and release times for all the window resources ensure the possibility of inter-node communications during the whole job execution. In this way, the occupation and availability probabilities should be estimated for each resource during the scheduling interval L. For the job scheduling, values $P_a^{r_i}(t; t+T)$ may be derived,

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77980-1_41

representing a probability that resource r_i will be available for the whole job execution interval T starting at time t.

When a set of n resources is required for a job execution for a period T, the total window availability P_a^{w} during the expected job execution interval can be estimated as a product of availability probabilities of each independent window nodes:

$$P_a^w(t) = \prod_i^n P_a^{r_i}(t; t+T). \quad (1)$$

If any of the window nodes will be occupied during the expected job execution interval T, the whole parallel job will be postponed or even aborted. Therefore, a common resources allocation problem is a maximization of a total resources' availability probability.

Based on the model above we consider the following job resources allocation problem in heterogeneous computing environment with non-dedicated resources and utilization uncertainties: during a scheduling interval L allocate a set of n nodes with performance $p_i \geq p$ for a time T, with common allocation and release times and a restriction C on the total allocation cost. As a target optimization criterion, we assume maximization of a whole window availability probability P_a^{w} (1).

The solution for this problem may be divided into two sub-problems.

1. Static sub-problem. Given the time t_k and values $P_a^{r_i}(t_k; t_k + T)$ of the resources' availability for the following period T, allocate a subset of n resources according to the job requirements with the maximum probability $P_a^{\mathrm{w}}(t_k)$.
2. Dynamic generalization. Perform time scan and execute the first sub-problem for each time moment $t_k \in [0; L]$. The solution is then obtained as a maximum from all the intermediate solutions: $P_a^{\mathrm{w}} = \max_{t_k} P_a^{\mathrm{w}}(t_k)$.

Thus, further in this paper we study different approaches for these two sub-problems implementation.

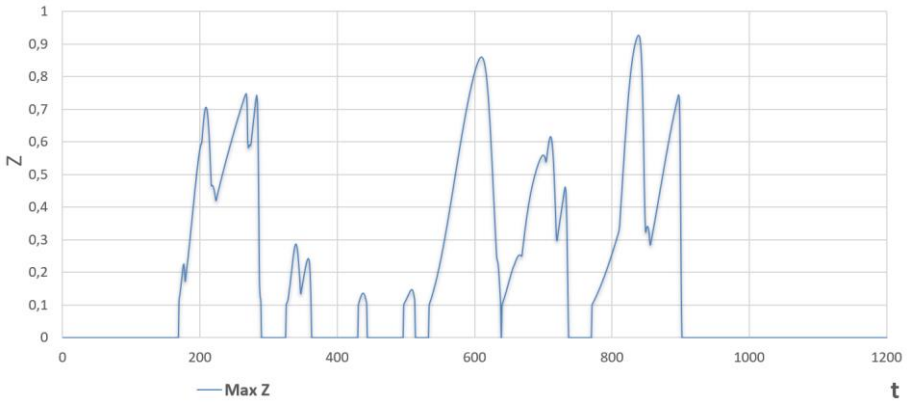

Fig. 3. An example of max $P_a^w(t)$ function for a parallel job resources allocation.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77980-1_41

As an example, Fig. 3 shows maximum values of function $Z = P_a^w(t)$ obtained for a parallel job on the interval $[0; 1200]$ with the maximum availability probability reaching 0.93 at $t^{\max} = 834$.

3.3 Near-optimal Resources Allocation

Let us discuss in more details the procedure which allocates an optimal (according to the probability criterion P_a^w) subset of n resources at some static time moment t_k.

We consider the following total resources availability criterion $P_a^w = \prod_i^n P_a^{r_i}$, where $P_a^{r_i} = P_i$ is an availability probability of a single resource r_i on the interval $[t_k; t_k + T]$.

In this way we can state the following problem of an n - size window subset allocation out of m available nodes in the system:

$$P_a^w = \prod_{j=1}^m x_j P_j, \tag{2}$$

with the following restrictions:

$$\sum_{j=1}^m x_j c_j \le C,$$

$$\sum_{j=1}^m x_j = n,$$

$$x_j \in \{0,1\}, j = 1..m,$$

where c_j is total cost required to allocate resource r_j for a time T, x_j - is a decision variable determining whether to allocate resource r_j ($x_j = 1$) or not ($x_j = 0$) for the current window.

In [14] based on a classical 0-1 *Knapsack* problem solution we proposed the following dynamic programming recurrent scheme to solve problem (2):

$$f_j(c, v) = \max\{f_{j-1}(c, v), f_{j-1}(c - c_j, v - 1) * P_j\}, \tag{3}$$

$$j = 1, .., m, c = 1, .., C, v = 1, .., n,$$

where $f_j(c, v)$ defines the maximum availability probability value for a v-size window allocated from the first j considered resources for a budget c. After the forward induction procedure (3) is finished the maximum availability value $P_{a\ max}^w = f_m(C, n)$. x_j values are then obtained by a backward induction procedure. Further in this paper we will refer to this algorithm simply as *Knapsack*.

An estimated computational complexity of the presented recurrent scheme is $O(m * n * C)$, which is n times harder compared to the original *Knapsack* problem ($O(m * C)$).

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77980-1_41

3.4 Greedy Resources Allocation Algorithm

Another approach for the static subset allocation sub-problem is to use more computationally efficient greedy algorithms. We outline four main greedy algorithms to solve the problem (2).

1. *MaxP* selects first n nodes providing maximum availability probability P_j values. This algorithm does not consider total usage cost limit and may provide infeasible solutions. Nevertheless, *MaxP* can be used to determine the best possible availability options and estimate a budget required to obtain them.
2. An opposite approach *MinC* selects first n nodes providing minimum usage cost c_j or an empty list in case it exceeds a total cost limit C. In this way, *MinC* does not perform any availability optimization, but always provides feasible solutions when it is possible. Besides, *MinC* outlines a lower bound on a budget required to obtain a feasible solution.
3. Third option is to use a weight function to regularize nodes in an appropriate manner. *MaxP/C* uses $w_j = P_j/c_j$ as a weight function and selects first n nodes providing maximum w_j values. Such an approach does not guarantee feasible solutions but performs some availability optimization by implementing a compromise solution between *MaxP* and *MaxC*.
4. Finally, we consider a joint approach *GreedyJnt* for a more efficient greedy-based resources allocation. The algorithm consists of three stages.
   a. Obtain *MaxP* solution and return it if the constraint on a total usage cost is met.
   b. Else, obtain *MaxP/C* solution and return it if the constraint on a total usage cost is met.
   c. Else, obtain *MinC* solution and return it if the constraint on a total usage cost is met.

   This combined algorithm is designed to perform the best possible greedy optimization considering restrictions on total resources allocation size and cost.

Estimated computational complexity for the greedy resources' allocation step is $O(m * \log m)$.

3.5 Time Scan Optimization

Dynamic generalization of the static resources' allocation problem requires a full-time scan performed over all the considered scheduling interval L. In general, this leads to a significant increase in the computational cost of the dynamic scheduling algorithm (especially, when a full knapsack-based optimization should be performed for all time moments $t_k \in [0; L]$).

To optimize the performance of the proposed resources allocation procedure during the time scan we consider a computational method which performs search for the maximum from a set of starting time points. Assuming, that the functions $P_a^{r_i}(t)$ for each resource are continuous in time (see Fig. 1), then their product $P_a^w(t)$ will be continuous as well. This means that certain computational algorithms are applicable for $P_a^w(t)$ function study and the extrema search. Fig. 2 shows an example of $P_a^w(t)$

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77980-1_41

function calculated by the resources allocation algorithm after scanning all time points if $[0; 1200]$ interval.

A general procedure for $\max P_a^w(t)$ search optimization during the scheduling interval L can be presented as follows.

1. A set of starting time points is allocated on the interval L. Their particular locations can be given as 1) uniform, 2) randomized, 3) a combination of options 1 and 2.
2. At each starting time point t_i^s the value of $P_a^w(t_i^s)$ is calculated by the static resources' allocation algorithm (*Knapsack* or *GreedyJnt*) based on actual resources state at t_i^s.
3. The gradient value is determined for each starting point by calculating and comparing neighbor values $P_a^w(t_i^s + 1)$ and $P_a^w(t_i^s - 1)$ with $P_a^w(t_i^s)$.
4. From each starting point t_i^s an incremental movement is performed in the direction of increasing the gradient by the sequential calculation of $P_a^w(t_i^s \pm \delta * k) = P_a^w(t_i^s, k)$, where k is a step number. The movement is stopped if the maximum is reached (when $P_a^w(t_i^s, k) < P_a^w(t_i^s, k-1)$) and, thus, can be found on the interval $[t_i^s \pm \delta * (k-1) ; t_i^s \pm \delta * k]$. Besides, the search movement stops if any other starting points t_{i+1}^s or t_{i-1}^s are reached. In this case, the search will be continued independently, starting from the corresponding points.

It should be noted that the above optimization procedure does not guarantee an exact solution: scenarios of finding local maxima or missing abrupt function changes are possible. Improving the accuracy is possible by increasing the set of starting points and by decreasing the search step length δ. On the other hand, the performance of this procedure is significantly increased compared to the full-time scan: the calculation of function $P_a^w(t)$ is performed on a limited set of time points, guaranteed to be smaller than the whole interval L.

4 Simulation Study

4.1 Simulation Environment

We performed a series of simulations to study optimization properties of the proposed dynamic resources allocation approaches. An experiment was prepared as follows using a custom distributed environment simulator [5, 6, 14]. For our purpose, it implements a heterogeneous resource domain model: nodes have different usage costs and performance levels. A space-shared resources allocation policy simulates a local queuing system (like in CloudSim [2, 3]) and, thus, each node can process only one task at any given simulation time. Additionally, each node supports a list of active global and local job utilization events.

Global static uncertainty events represent resources failure or shutdown susceptibility and keep a constant occupation probability during the whole scheduling interval L. Static utilization is generated for each resource based on a random variable P_o of occupancy probability with a normal. System-wide global-load parameter defines a standard deviation for P_o and is used to set an average global utilization for the whole

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77980-1_41

computing environment. Thus, for example, when global load = 0.05, about 68% of the resources on average have global occupancy probability $P_o^{r_j} < 0.05$. More detailed study of a static resources' allocation problem under global utilization uncertainties was provided in [14].

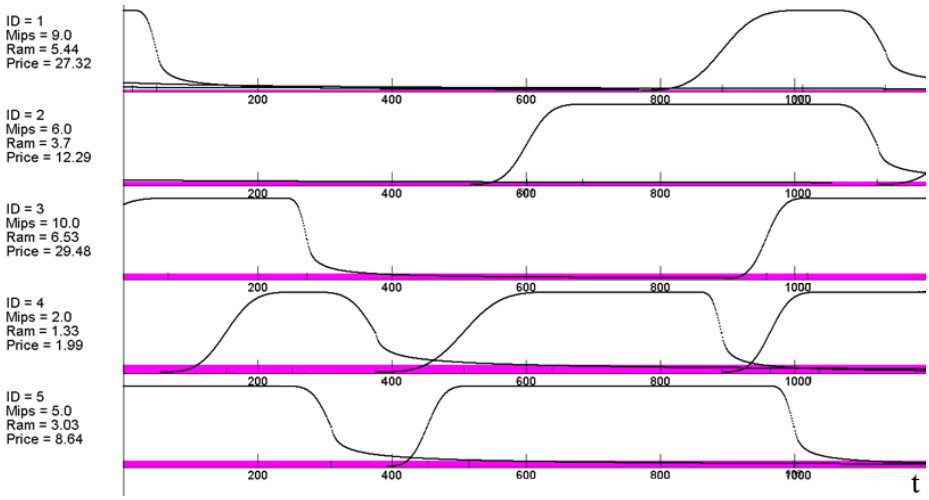

Fig. 4. An example of static and dynamic load generated for system resources.

Dynamic job-based utilization uncertainty is generated based on a preliminary job-flow scheduling simulation. For each resource, a list of single-node jobs is generated with random jobs submit times, lengths, start time and finish time uncertainty estimations. The jobs are ordered by their submit time and are scheduled in advance starting either at the submit time, or after the previous job is finished. During this scheduling, a chain of the resource *allocation*, *occupation* and *release* events is generated for each job. Corresponding expected times and standard deviations are defined by the job length and uncertainty parameters. More details regarding the simulated job-flow properties provided in Section 3.1. A total length of jobs generated for each resource is determined by a system wide job-load parameter. For example, when job-load = 0.1, a total length of locally generated jobs constitutes nearly 10% of the considered scheduling interval L.

Fig. 1 shows a single resource utilization schedule with global and dynamic utilization events generated based on the procedures described above. Fig. 4 shows an example of global and dynamic utilization uncertainties generated for a subset of the system resources in simulator [14].

4.2 Dynamic Resources Allocation

To solve the dynamic resources allocation problem for a parallel job, it is necessary to consider the available resources' schedule and utilization events which change over

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77980-1_41

time Thus, the scheduling problem requires allocation of a set of suitable resources not at some static moment t_k, but during a given time interval.

Since the computational complexity and working time of the algorithms under consideration increase in proportion to the size of the considered scheduling interval, the following parameters of the scheduling problem were chosen to minimize the simulation time. It is required to maximize the probability of simultaneous availability of 6 concurrently available nodes to perform a job with a volume of 200 computational units on a time interval [0; 800] in a computing environment that includes 64 heterogeneous computational nodes. Initial load of computational nodes with global events global load = 0.05. The dynamic load of the computing system changed during the simulation within the limits of job-load $\in [0; 1]$.

The obtained results indicating the availability of the resources selected by the Knapsack (Section 3.3) and GreedyJnt (Section 3.4) algorithms depending on the dynamic load job-load values, are presented in Fig. 5. To obtain these results, more than 10,000 independent scenarios of scheduling and resources allocation were performed by each of the considered algorithms.

It should be noted that with job-load = 0 the advantage of the Knapsack algorithm is about 9%, and the probability of simultaneous availability of the selected resources is 0.96 for Knapsack and 0.87 for GreedyJnt.

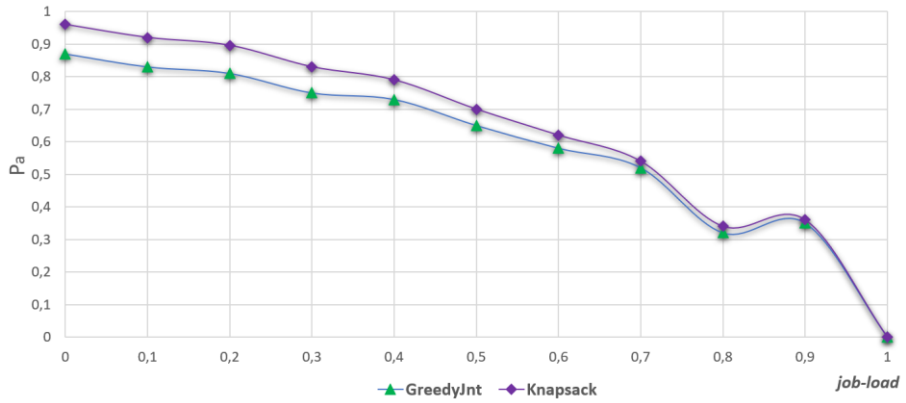

Fig. 5. Simulation results: P_a^w resources availability obtained by *Knapsack* and *GreedyJnt* algorithms depending on the resources utilization level.

With an increase in the dynamic load of the system (*job-load* > 0), the highest achievable probability P_a^w of simultaneous resource availability, as expected, sensibly monotonically decreases. The local maximum at *job-load* = 0.9 is explained by the fact that under conditions of extra high dynamic load, the number of experiments in which it was possible to find six concurrently available resources, turned out to be statistically insignificant (about 10 results). At the same time, when the value of job-load = 1 (full initial utilization of the system) was reached, a suitable set of resources ($P_a^w = 0$) was not found in any of the experiments at any time instant $t_k \in [0; 800]$.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77980-1_41

Also, *Knapsack* provided a higher availability probability P_a^w of the required set of resources at all the considered values of the dynamic load (*job-load* <1) in comparison to *GreedyJnt*. However, the relative advantage decreases from about 9% to almost 0% as the *job-load* increases. This decrease in relative efficiency is explained by a decrease in the dimensionality and variability of problem (2) with an increase of the resources load. For example, when *job-load* = 0 all 64 resources are available at every instant with a probability of at least 0.95 (due to global-load = 0.05). Then as the *job-load* increases, many resources fall out of consideration due to a high probability of being utilized by other jobs (see Fig. 4). Thus, for large *job-load* values, the static algorithms often solved the degenerated problem of selecting a set of 6 concurrently available resources from 6 resources in the system that remained unloaded.

4.3 Time Scan Optimization

The time and accuracy characteristics of the proposed time scan optimization procedure (Section 3.5) were studied based on a resource's allocation problem in computing environment with dynamically changing utilization level. Fig. 4 presents an example of a utilization schedule generated for a few computational nodes in the simulation environment [14].

To obtain reliable results, we performed more 1000 independent simulation scenarios of resources allocation for a single parallel job. The computing environment consisted of 64 heterogeneous computing nodes of varying cost and performance characteristics with dynamically changing occupation function $P_o(t)$: *job-load* = 0.5, *global-load* = 0.05. The job scheduling problem required allocation of six nodes for 200 units of time on the interval $L \in [0; 800]$. The target optimization criterion P_a^w is a simultaneous availability of the selected resources. As an additional criterion, a total algorithm working times was measured.

The time scan optimization procedure described in Section 3.5 was implemented with a different number of the starting points: {1, 5, 10, 20, 50, 100}.

Tab 1 shows the relative results in terms of working time (performance acceleration) and accuracy in comparison with the full scan approach.

Table 1. Algorithms' efficiency comparison in terms of accuracy and performance (working time acceleration) relative to the *Knapsack* full time scan implementation

Algorithm	Optimization Accuracy	Time Acceleration
Full scan (*Knapsack*)	1	1
1 starting point (*Knapsack*)	0,8	65
5 starting points (*Knapsack*)	0,93	17
10 starting points (*Knapsack*)	0,96	10,5
20 starting points (*Knapsack*)	0,97	8,3
50 starting points (*Knapsack*)	0,99	6,8
100 starting points (*Knapsack*)	0,99	3,7
Full scan (*GreedyJnt*)	0,953	143

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77980-1_41

As expected, with an increase in the number of starting points the accuracy of the approximate procedure tends to 1 (i.e., to the optimal solution obtained with a full scan search). Already with 50 starting points (on an interval of 801 points) the accuracy of the optimized solution reaches 99%, while the calculation time is accelerated by almost 7 times.

On the other hand, full scan procedure with *GreedyJnt* algorithm achieves 95% accuracy with a 143x speedup! Thus, it is advisable to apply *Knapsack* with this time scan optimization technique if it is necessary to achieve a high accuracy in the presence of the light computation time restrictions. In this case, it is possible to speed up the work time by about an order of magnitude. With tighter time constraints, additional speedup can be achieved by using a greedy counterpart. In addition, the time scan optimization is applicable to *GreedyJnt* algorithm as well: for example, running *GreedyJnt* algorithm from 50 starting points allows you to speed up the computation time by 1000 times, while the solution accuracy will decrease only to 94%.

5 Conclusion and Future Work

In this work, we presented procedure for a reliable resources' allocation in high performance computing systems with heterogeneous hosts considering utilization uncertainty. The uncertainties are formalized with probability functions as a natural way of statistical and machine learning predictions representation. The proposed solution uses an availability criterion to optimize resources allocation under static and dynamic utilization features. *Knapsack*-based and greedy algorithms were implemented and compared in a dynamic procedure performing optimized time scan over a specified scheduling interval. Both approaches were able to successfully optimize availability of the selected resources.

We considered several types of static and dynamic job-based resources utilization events with different load levels.

The simulation study addressed two main criteria: optimization efficiency and algorithms working time. *Knapsack*-based solution advantage over the greedy approach by the resources availability criterion at average reaches 5% but requires nearly 100 times more time for the calculations. Considering a relatively high computation complexity of the *Knapsack*-based solution, several optimization options were proposed to provide 99% accuracy 10 times faster or almost 94% accuracy 1000 times faster.

In our further work, we will refine the resource utilization model to simulate different types of global and local utilization events closer to real systems.

References

1. Lee, Y.C., Wang, C., Zomaya, A.Y., Zhou, B.B.: Profit-driven scheduling for cloud services with data access awareness. J. of Parallel and Distributed Computing, 72(4), 591–602 (2012)
2. Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. F., and Buyya, R.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77980-1_41

resource provisioning algorithms. J. Software: Practice and Experience, 41(1), 23-50 (2011)
3. Samimi, P., Teimouri, Y., Mukhtar M.: A combinatorial double auction resource allocation model in cloud computing. J. Information Sciences, 357(C), 201-216 (2016)
4. Sample N., Keyani P., Wiederhold G.: Scheduling under uncertainty: planning for the ubiquitous Grid. In: Arbab F., Talcott C. (Eds.): Coordination Models and Languages. COORDINATION 2002. Lecture Notes in Computer Science, 2315, Springer, Berlin, Heidelberg, 300-316 (2002)
5. Toporkov, V., Yemelyanov, D.: Optimization of resources selection for jobs scheduling in heterogeneous distributed computing environments. Lecture Notes in Computer Science, 10861, Springer Verlag, 574–583 (2018)
6. Toporkov, V., Yemelyanov, D., and Toporkova, A.: Coordinated global and private job-flow scheduling in Grid virtual organizations. Simulation Modelling Practice and Theory, 107, 102228 (2021)
7. Jackson, D., Snell, Q., Clement, M.: Core algorithms of the Maui scheduler. In: Revised Papers from the 7th International Workshop on Job Scheduling Strategies for Parallel Processing, JSSPP '01, pp. 87-102 (2001)
8. Tsafrir, D., Etsion, Y., Feitelson, D.G.: Backfilling using system-generated predictions rather than user runtime estimates. IEEE Transactions on Parallel and Distributed Systems, 18(6), 789 - 803 (2007)
9. Tchernykh, A., Schwiegelsohn, U., El-ghazali, T., Babenko, M.: Towards understanding uncertainty in cloud computing with risks of confidentiality, integrity, and availability. J. Comput. Sci. 36 100581 (2019)
10. Chaari, T., Chaabane, S., Aissani, N., and Trentesaux, D.: Scheduling under uncertainty: survey and research directions. 2014 International Conference on Advanced Logistics and Transport (ICALT), 229-234 (2014)
11. Ramírez-Velarde, R., Tchernykh, A., Barba-Jimenez, C., Hirales-Carbajal, A., Nolazco-Flores, J.: Adaptive resource allocation with job runtime uncertainty. J. of Grid Computing, 15(4), 415–434 (2017)
12. Wu, C.W., Brown, K.N. and Beck, J.C.: Scheduling with uncertain durations: modeling beta-robust scheduling with constraints. J. Computers and Operations Research, 36 (8), 2348-2356 (2009)
13. Rood, B., Lewis, M.J.: Grid Resource Availability Prediction-Based Scheduling and Task Replication. J. Grid Computing, 7, 479-500 (2009)
14. Toporkov, V., Yemelyanov, D.: Availability-based resources allocation algorithms in distributed computing. In: V. Voevodin and S. Sobolev (Eds.): RuSCDays 2020, CCIS 1331, Springer Nature Switzerland AG, 551-562 (2020)
15. https://www.cse.huji.ac.il/labs/parallel/workload/ (2021)
16. Feitelson, D.G.: Workload modeling for computer systems performance evaluation. New York: Cambridge university press, 501-540 (2015)

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77980-1_41