Scheduling a Time-Varying Workload of Multiple Types of Jobs on Distributed Resources

Georgios L. Stavrinides
Department of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece
gstavrin@csd.auth.gr

Helen D. Karatza
Department of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece
karatza@csd.auth.gr

***Abstract*—Recent technological advances, such as cloud and fog computing, have made prevalent the utilization of distributed computational resources. There is a variety of complex workloads that need to be processed on such platforms. The type of the workload may vary over time. Two of the most important challenges in a distributed environment are efficient resource allocation and the employment of appropriate workload scheduling techniques. In this paper, we examine dynamic scheduling of complex workloads on distributed resources. The workload consists of single-task jobs, bag-of-task jobs and linear workflows. The characteristics of the investigated workload vary over time. Two scheduling techniques are studied. Simulation modeling is used to compare their performance under different workload and system load scenarios.**

Keywords—scheduling; time-varying workload; bags-of-tasks; linear workflows; distributed resources; performance evaluation

I. INTRODUCTION

The paradigm shift from traditional approaches to cloud and fog computing, has made prevalent the utilization of large-scale distributed computational resources. Such distributed platforms are commonly used to process a wide range of complex jobs. Two of the most important challenges in a distributed environment are efficient resource allocation and the employment of effective workload scheduling techniques. In its general form, the scheduling problem concerns the allocation of resources to jobs and determines the order in which the jobs will be executed on their assigned resources. There are many objectives of the scheduling problem. The most common one is the minimization of job execution time. Efficient job scheduling is a key factor in order to obtain good overall system performance and provide fairness to end-users [1-9].

There is a variety of complex workloads that need to be processed in distributed environments. Jobs should be effectively assigned and scheduled on the distributed processors, so that the load is evenly distributed among them. Complex jobs may consist of tasks that are independent from each other or have dependencies among them. *Bag-of-tasks (BoT)* jobs are complex jobs which consist of independent tasks that can run on any processor and in any order. There are no precedence constraints among the tasks. The execution of a BoT job is finished only when all of its component tasks have finished processing [10-19]. A different type of complex jobs is *Linear Workflows (LW)* [20-23]. A LW job comprises multiple tasks that need to run in sequential order, one after the other. These tasks have precedence constraints among them and they run on the same processor in order to leverage data locality.

In this paper, we examine dynamic scheduling of complex workloads on distributed resources. The workload consists of jobs that are a mixture of Single-Task Jobs (STJs), BoTs and LWs. Scheduling complex workloads in distributed systems is not easy due to multiple tasks competing for resources. The simplest scheduling strategy to implement is the *First-Come-First-Served (FCFS)*. This strategy does not require any information about the characteristics of the jobs. However, it does not always provide good overall performance. On the other hand, scheduling strategies that use information about task service demands or number of tasks per job, usually perform better than the FCFS scheduling policy [16-17]. However, such techniques typically involve considerable overhead.

In this work we consider that task service demands are known in advance. This information can be obtained either by task profiling or from historical data of previous task executions. We study a scheduling policy that takes into account the total service time of all tasks in each job. However, this policy rearranges queues only at the end of some time periods, called epochs, in an attempt to reduce the incurred scheduling overhead. The performance of this strategy is compared to the FCFS policy. We also employ a routing algorithm known as the 2 Random Choices technique [24]. Based on this approach, the shortest queue between two randomly chosen processors is selected, in contrast to other methods that consider all of the available processors during each routing decision. Consequently, in our research, each STJ, each task of a BoT job and each LW, are assigned to the shortest of two randomly selected distributed queues.

In distributed environments the workload pattern typically changes over time. For instance, during some time intervals the jobs that arrive are STJs, while in other time intervals they are BoTs or LWs. For this reason, in this paper we consider a time-varying workload, where the type of jobs that arrive at the system changes in exponentially distributed time intervals. The scope of this research is to study the performance of the employed scheduling strategies in time-varying workload conditions. The performance is expressed in terms of the mean response time of the jobs. The fairness in job execution is expressed as the maximum response time observed during the

SummerSim-SPECTS 2020, July 20-22, Madrid, Spain;

simulation experiments. Different system load cases are examined.

Previous works that studied linear workflows [20-23] did not examine LWs together with BoTs and STJs. Furthermore, they did not consider time-varying workloads. To the best of our knowledge, scheduling time-varying complex workloads consisting of single-task jobs, bag-of-task jobs and linear workflows, in the context presented in this paper, has never been studied in the research literature before.

The remainder of the paper is organized as follows: Section II describes the model of the distributed system under study and the time-varying workload. Section III presents the employed routing method and scheduling techniques. Section IV defines the performance parameters. Section V describes the experimental setup and the simulation results. Finally, Section VI provides conclusions and future research plans.

II. Problem Definition

A. System and Workload Models

Fig. 1 presents the queueing network model of the distributed resources under study. The considered distributed resources form a cluster of P = 16 distributed processors. Each processor is equipped with its own queue where STJs, tasks of BoTs and LWs wait for execution.

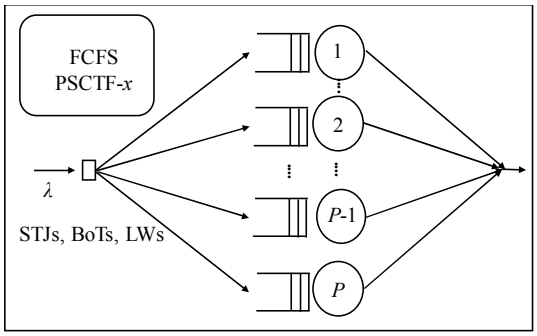
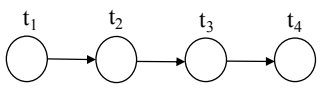


Fig. 1. The queueing model of the distributed system under study.

The workload comprises jobs of different types: single-task jobs, bag-of-tasks jobs and linear workflows. STJs are CPU intensive compared to individual tasks of BoTs and LWs. A BoT job J consists of N_J independent tasks, which can run on any processor and in any order. On the other hand, a LW job J consists of N_J sequential tasks that have precedence constraints among them. Consequently, in a LW dependencies between tasks can be represented by a linear graph. Fig. 2 depicts an example of a LW job with 4 tasks.



Linear Workflow

Fig. 2. A Linear Workflow job with four tasks.

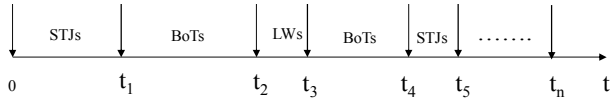


Fig. 3. Time-varying workload.

The considered workload is characterized by the following random variables:

- Job interarrival time.
- Time intervals of workload change.
- Number of tasks per BoT job.
- Number of tasks per LW.
- Service time of STJs.
- Service time of BoT tasks.
- Service time of LW tasks.

The random variables presented above have the following distributions:

1) Distribution of Job Interarrival Time: STJs, BoT jobs and LWs arrive in a single arrival stream (Fig.1). The interarrival times are exponential random variables with mean $1/\lambda$, where λ is the mean arrival rate of the jobs.

2) Distribution of Time Intervals of Workload Change: We assume that the characteristics of the workload change in time intervals that are exponentially distributed with mean D. After the end of each time interval the arriving jobs belong to one of the other two workload types with equal probability (Fig. 3). The jobs that arrive at the processors within the same time interval belong to the same job type. However, during the same time interval there may exist some jobs at the processors that arrived during a past time interval and may belong to different job types. These jobs may still wait at the processor queues or may be under service.

3) Distribution of the Number of Tasks per BoT and LW: The number of tasks in a BoT or a LW is an integer uniformly distributed in the range [2, 4]. Consequently, the mean number of tasks per BoT and LW is $\eta = 3$.

4) Distribution of Task Service Demand: The service demands of the tasks follow a hyperexponential distribution with coefficient of variation CV. The mean service time for the STJs is $1/\mu_1$, whereas the mean service time for the tasks of BoT jobs and LWs is $1/\mu_2$, where μ_1 and μ_2 are the mean service rates respectively.

If s_i is the service time of task i in a BoT J or in a LW J, then the total cumulative service time of all N_J tasks of the job J is given by the relation:

$$cst_j = \sum_{i=1}^{N_J} s_i \quad (1)$$

We consider that $1/\mu_1 = \eta \times (1/\mu_2)$. Therefore, the average service demand of a STJ is equal to the average cumulative service demand of all tasks of a BoT job or a LW. The system and workload parameters used in the paper are shown in Table I.

TABLE I. SYSTEM AND WORKLOAD PARAMETERS

P	Number of processors
$1/\lambda$	Mean interarrival time of jobs
D	Mean time between workload type changes
η	Average number of tasks per BoT job and per LW
$1/\mu_1$	Mean service demand of STJs
$1/\mu_2$	Mean service demand of BoT tasks and LW tasks
CV	Coefficient of variation of service time distribution

III. ROUTING AND SCHEDULING

A. Routing Technique

2 Random Choices Routing: It has been shown in the literature that the shortest queue policy in many cases performs well. However, it is not recommended in the case of large-scale distributed systems, as it involves significant overhead. On the other hand, a probabilistic routing policy usually distributes the workload unevenly to the distributed resources. For this reason, in this work the routing is based on the 2 Random Choices approach [24]. Each STJ, each task of a BoT job and each LW is assigned to the shortest of two randomly selected processor queues.

B. Scheduling Policies

The following two strategies are utilized for the scheduling of the STJs, BoT tasks and LWs on the distributed processors:

1) First-Come-First-Served (FCFS): With this scheduling strategy, STJs, BoT tasks and LWs are processed in the order of their assignment to the processor queues. This is the simplest scheduling algorithm as it does not require any queue rearrangement when a new STJ, BoT task or LW is inserted into a queue.

2) Periodic-Shortest-Cumulative-Time-First with period x (PSCTF-x): With this strategy, processor queues are rearranged only at the end of predefined time periods with size x. At the end of each period the scheduler recalculates the priorities of all STJs, BoT tasks and LWs, using the criterion of scheduling in each queue first the STJ, BoT task or the LW that has the shortest total service time. This scheduling technique gives priority to smaller STJs, BoT tasks and LWs only from time to time. Therefore, it does not cause excessive delays to larger STJs, BoT tasks and LWs, as they are only periodically bypassed by smaller STJs, BoT tasks and LWs.

IV. PERFORMANCE EVALUATION PARAMETERS

The metrics used for the performance evaluation of the framework under study are presented in Table II.

TABLE II. PERFORMANCE METRICS

U	Mean processor utilization
RT	Average response time of STJs, BoT jobs and LWs
D_{RT}	Relative (%) decrease in RT when the PSCTF-x policy is utilized instead of the FCFS policy
MRT	Average maximum response time of jobs
MRT-$Ratio$	The ratio of MRT in the PSCTF-x case divided by MRT in the FCFS case

RT represents the overall performance of the system. In this paper the fairness in job execution is indicated by MRT-$Ratio$.

V. EXPERIMENTAL SETUP & SIMULATION RESULTS

A. Experimental Methodology

We developed a discrete event simulator in C for the performance evaluation of the system. We chose to implement a custom simulator due to the complexity of the problem under study and in order to have full control on all of the system and workload parameters. We used the independent replications method for the simulation experiments. We conducted 30 replications of the simulation program for each set of input parameters, with different seeds of random numbers in each run. For every mean value, a 95% confidence interval was calculated. The half-widths of the confidence intervals were smaller than 5% of their respective mean values. Table III presents the parameters used as input in our simulation model.

TABLE III. INPUT PARAMETERS

$1/\lambda$	0.24, 0.22, 0.20
$1/\mu_1$	3
$1/\mu_2$	1
CV	2
x	5, 10
D	5, 10

According to the parameters we used, on average each BoT job and each LW consisted of $\eta = 3$ tasks. Therefore, the average cumulative service time per BoT job, as well as per LW was equal to η, as the mean service demand of each BoT task and each LW task was equal to 1. Furthermore, the average service time per single task job was also η, as the mean service time of STJs was three times larger than the mean service time of each task of the other two types of jobs. In case all of the processors were busy, an average of $P/\eta = 5.33$ jobs could be served at each unit of time. Consequently, λ should be such that $\lambda < 5.33$, so that the processor queues would not be overloaded. Therefore, we had to choose a value of mean interarrival time $1/\lambda$ such that:

$$1/\lambda > 0.1875 \quad (2)$$

Due to the complex nature of the workload, and in order to avoid large synchronization delays of sibling BoT tasks, and thus large response times, in the simulation experiments we chose the following three cases of mean interarrival time: $1/\lambda$ = 0.24, 0.22, 0.20. Furthermore, we chose two period sizes, as well as two sizes of time intervals for the workload type change. The periods and workload change time intervals were long enough so that several jobs arrived at the system during each period and during each time interval of the workload type change.

B. Simulation Results

Tables IV and V present the mean processor utilization U for $D = 5$ and $D = 10$, respectively. We can observe that in all cases both scheduling algorithms, FCFS and PSCTF-x, provided approximately the same mean processor utilization.

TABLE IV. MEAN PROCESSOR UTILIZATION U FOR $D = 5$

$1/\lambda$	0.24	0.22	0.20
PSCTF-5	0.783	0.855	0.937
PSCTF-10	0.783	0.855	0.937
FCFS	0.783	0.855	0.938

TABLE V. MEAN PROCESSOR UTILIZATION U FOR $D = 10$

$1/\lambda$	0.24	0.22	0.20
PSCTF-5	0.783	0.855	0.938
PSCTF-10	0.783	0.855	0.937
FCFS	0.783	0.855	0.938

Regarding the mean response time, Fig. 4, 5, 7 and 8, reveal that the PSCTF-x scheduling policy performed better than FCFS. This is due to the fact that with the FCFS policy there were cases where STJs, BoT tasks and LWs with a small total service demand had to wait behind other large STJs, BoT tasks and LWs that preceded them in the processor queues. Furthermore, the results show that PSCTF-5 performed better than PSCTF-10, due to a larger number of queue rearrangements performed in the former case. The results also show that RT increased with decreasing interarrival time due to the increasing load.

It is apparent that the relative performance of the different policies was not significantly affected by the size of the time interval D for the workload type change. For each D, D_{RT} increased with decreasing $1/\lambda$. This is due to the fact that the superiority of PSTF-x over FCFS was better exploited when there were more jobs in the system.

It is noted though that when the PSCTF-x scheduling strategy executed first the shortest STJ, BoT task or LW in a queue, other jobs could suffer from longer response times as if they were served on a FCFS basis. Especially in the case of BoTs, when a small task of a BoT was served first in a queue, this did not necessarily mean that the BoT to which the particular task belonged would have a smaller response time. This is because some sibling BoT tasks of the same BoT could experience larger delays in other processor queues due to this scheduling policy criterion. This could cause larger synchronization delays of sibling BoT tasks and therefore result in larger response times of the respective jobs. For this reason, periodic scheduling is preferable than the strict shortest cumulative time first policy, because it prevents the frequent reordering in the processor queues, which may result in the unfair scheduling of some jobs.

Fig. 6 and 9 show that in all of the examined cases, the mean maximum response time was larger in the PSCTF-x case than in the case of FCFS. This is because when queues were rearranged, some very large STJs, BoT tasks or LWs could be bypassed several times by smaller tasks. This could cause large response times in some jobs. *MRT-Ratio* was larger when the system load was high, as in that case jobs were bypassed many times by other jobs, due to the PSCTF-x criterion.

MRT-Ratio was larger in the PSCTF-5 case than in the case PSCTF-10 was employed, due to the larger number of queue rearrangements performed in the former case.

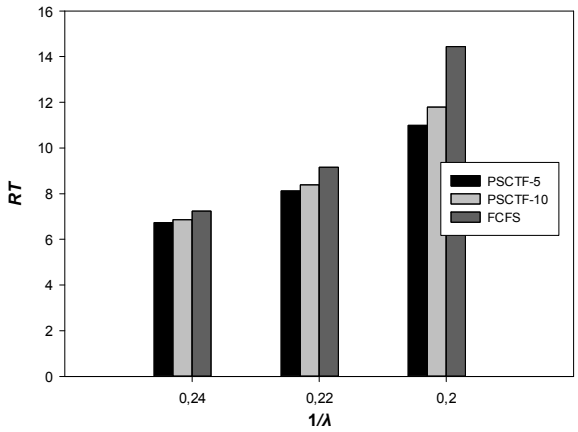


Fig. 4. RT versus $1/\lambda$, D=5.

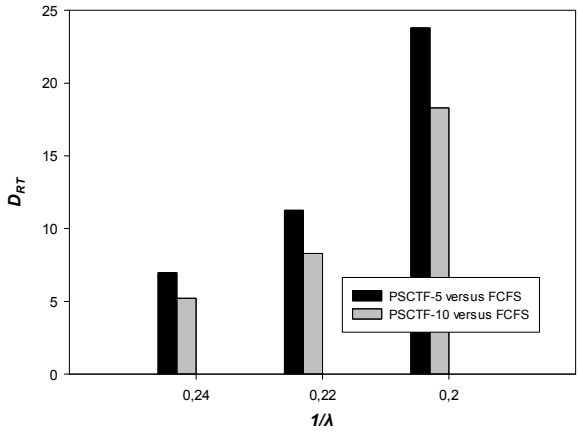


Fig. 5. D_{RT} versus $1/\lambda$, $D = 5$.

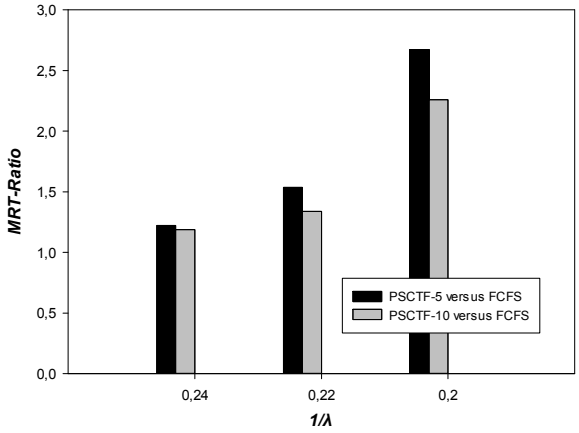


Fig. 6. *MRT-Ratio* versus $1/\lambda$, $D = 5$.

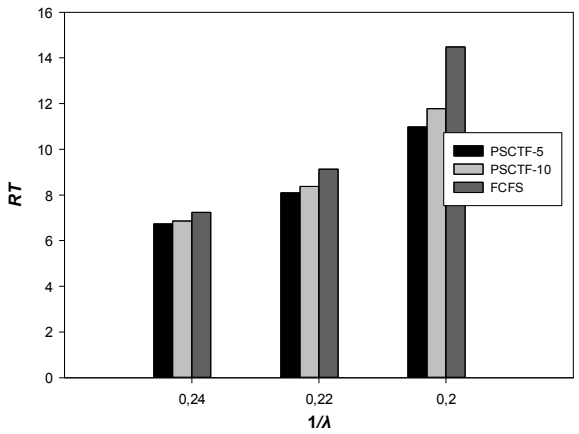


Fig. 7. *RT* versus $1/\lambda$, $D = 10$.

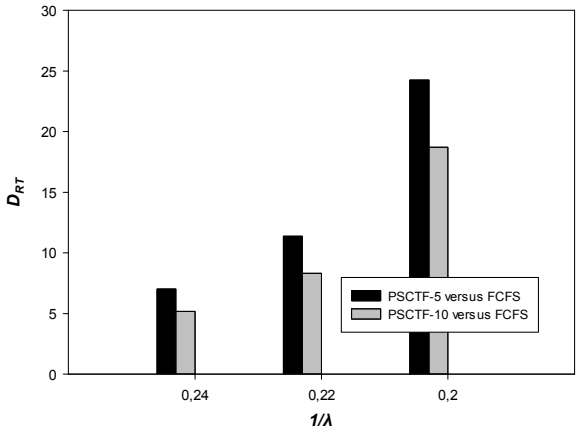


Fig. 8. D_{RT} versus $1/\lambda$, $D = 10$.

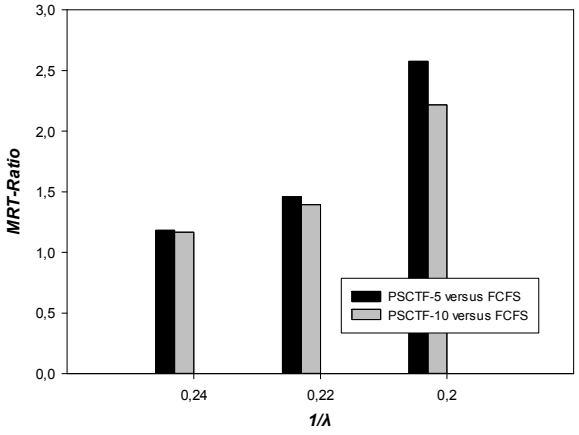


Fig. 9. *MRT-Ratio* versus $1/\lambda$, $D = 10$.

VI. Conclusions & Future Work

In this paper, we studied scheduling of complex workloads on distributed resources. The workload consisted of single-task-jobs, bag-of-tasks jobs and linear workflows. The characteristics of the workload varied over time in exponentially distributed time intervals. We employed a simulation model in order to evaluate and compare the performance of the utilized scheduling algorithms.

The simulation results revealed that the PSCTF-*x* scheduling technique performed better than the FCFS strategy in the framework under study. The superiority of the former method over the latter was more significant in the case of higher system utilization. The maximum response time observed during the simulation experiments was also larger when the system load was high. Overall, considering that the FCFS policy is simpler and fairer than PSCTF-*x*, we can conclude that for the complex workload types examined in this paper, FCFS should be utilized in low system load conditions. In all other cases the PSCTF-*x* policy should be employed.

In this work, we investigated workload comprising three categories of jobs. As future research, we plan to examine more types of complex workloads, such as gang jobs and non-linear workflows. Furthermore, we plan to investigate different distributions for the task service demands.

References

[1] G. L. Stavrinides and H. D. Karatza, "Scheduling data-intensive workloads in large-scale distributed systems: trends and challenges," in *Modeling and Simulation in HPC and Cloud Systems*, 1st ed., ser. Studies in Big Data, Feb. 2018, vol. 36, ch. 2, pp. 19-43.

[2] S. Zikos and H. D. Karatza, "Communication cost effective scheduling policies of nonclairvoyant jobs with load balancing in a grid," *Journal of Systems and Software*, vol. 82, no. 12, pp. 2103-2116, Dec. 2009.

[3] Z. C. Papazachos and H. D. Karatza, "Gang scheduling in multi-core clusters implementing migrations," *Future Generation Computer Systems*, vol. 27, no. 8, pp. 1153-1165, Oct. 2011.

[4] G. L. Stavrinides and H. D. Karatza, "Cost-effective utilization of complementary cloud resources for the scheduling of real-time workflow applications in a fog environment," in *Proceedings of the 7th International Conference on Future Internet of Things and Cloud (FiCloud'19)*, Istanbul, Turkey, Aug. 2019, pp. 1-8.

[5] G. L. Stavrinides and H. D. Karatza, "A hybrid approach to scheduling real-time IoT workflows in fog and cloud environments," *Multimedia Tools and Applications*, vol. 78, no. 17, pp. 24639-24655, Sep. 2019.

[6] G. L. Stavrinides and H. D. Karatza, "Scheduling real-time jobs in distributed systems - simulation and performance analysis," in *Proceedings of the First International Workshop on Sustainable Ultrascale Computing Systems (NESUS'14)*, Porto, Portugal, Aug. 2014, pp. 13-18.

[7] G. L. Stavrinides and H. D. Karatza, "Performance evaluation of a SaaS cloud under different levels of workload computational demand variability and tardiness bounds," *Simulation Modelling Practice and Theory*, vol. 91, pp. 1-12, Feb. 2019.

[8] G. L. Stavrinides and H. D. Karatza, "An energy-efficient, QoS-aware and cost-effective scheduling approach for real-time workflow applications in cloud computing systems utilizing DVFS and approximate computations," *Future Generation Computer Systems*, vol. 96, pp. 216-226, Jul. 2019.

[9] H. D. Karatza, "Performance of gang scheduling policies in the presence of critical sporadic jobs in distributed systems," in *Proceedings of the 2007 International Symposium on Performance Evaluation of Computer*

and Telecommunication Systems (SPECTS'07), San Diego, CA, Jul. 2007, pp. 547-554.

[10] A. M. Oprescu, T. Kielmann, and H. Leahu, "Stochastic tail-phase optimization for bag-of-tasks execution in clouds," in *Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing (UCC'12)*, Chicago, IL, Nov. 2012, pp. 204-208.

[11] G. L. Stavrinides and H. D. Karatza, "Scheduling a job mix of bag-of-tasks and bag-of-task-chains on distributed resources," in *Proceedings of the 11th International Conference on Information and Communication Systems (ICICS'20)*, Irbid, Jordan, Apr. 2020, pp. 394-399.

[12] M. H. Farahabady, Y. C. Lee, and A. Y. Zomaya, "Non-clairvoyant assignment of bag-of-tasks applications across multiple clouds," in *Proceedings of the 13th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'12)*, Beijing, China, Dec. 2012, pp. 423-428.

[13] G. L. Stavrinides and H. D. Karatza, "The impact of data locality on the performance of a SaaS cloud with real-time data-intensive applications," in *Proceedings of the 21st IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications (DS-RT'17)*, Rome, Italy, Oct. 2017, pp. 1-8.

[14] I. A. Moschakis and H. D. Karatza, "A meta-heuristic optimization approach to the scheduling of bag-of-tasks applications on heterogeneous clouds with multi-level arrivals and critical jobs," *Simulation Modelling Practice and Theory*, vol. 57, pp. 1-25, Sep. 2015.

[15] G. L. Stavrinides and H. D. Karatza, "Simulation-based performance evaluation of an energy-aware heuristic for the scheduling of HPC applications in large-scale distributed systems," in *Proceedings of the 8th ACM/SPEC International Conference on Performance Engineering (ICPE'17)*, *3rd International Workshop on Energy-aware Simulation (ENERGY-SIM'17)*, L'Aquila, Italy, Apr. 2017, pp. 49-54.

[16] G. L. Stavrinides and H. D. Karatza, "Periodic scheduling of mixed workload in distributed systems," in *Proceedings of the 23rd ICE/IEEE International Conference on Engineering, Technology and Innovation (ICE'17)*, Madeira Island, Portugal, Jun. 2017, pp. 94-99.

[17] G. L. Stavrinides and H. D. Karatza, "Task group scheduling in distributed systems," in *Proceedings of the 2018 International Conference on Computer, Information and Telecommunication Systems (CITS'18),* Colmar, France, Jul. 2018, pp. 1-5.

[18] G. L. Stavrinides and H. D. Karatza, "The impact of workload variability on the energy efficiency of large-scale heterogeneous distributed systems," *Simulation Modelling Practice and Theory*, vol. 89, pp. 135-143, Dec. 2018.

[19] G. L. Stavrinides and H. D. Karatza, "Scheduling real-time bag-of-tasks applications with approximate computations in SaaS clouds," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 1, e4208, Jan. 2020.

[20] K. Agrawal, A. Benoit and Y. Robert, "Mapping linear workflows with computation/communication overlap," in *Proceedings of the 14th IEEE International Conference on Parallel and Distributed Systems (ICPADS'08),* Melbourne, Australia, Dec. 2008, pp. 195-202.

[21] K. Agrawal, A. Benoit, L. Magnan and Y. Robert, "Scheduling algorithms for linear workflow optimization," in *Proceedings of the 2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS'10)*, Atlanta, GA, Apr. 2010, pp. 1-12.

[22] A. Benoit, J. Nicod and V. Rehn-Sonigo, "Optimizing buffer sizes for pipeline workflow scheduling with setup times," in *Proceedings of the 2014 IEEE International Parallel & Distributed Processing Symposium Workshops (IPDPS'14)*, Phoenix, AZ, May 2014, pp. 662-670.

[23] A. Benoit, A. Cavelan, Y. Robert, H. Sun, "Multi-level checkpointing and silent error detection for linear workflows," *Journal of Computational Science*, vol. 28, pp. 398-415, Sep. 2018.

[24] M. Mitzenmacher, "The power of two choices in randomized load balancing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 10, pp. 1094-1104, Oct. 2001.