Task Group Scheduling in Distributed Systems

Georgios L. Stavrinides
Department of Informatics
Aristotle University of Thessaloniki
54124 Thessaloniki, Greece
gstavrin@csd.auth.gr

Helen D. Karatza
Department of Informatics
Aristotle University of Thessaloniki
54124 Thessaloniki, Greece
karatza@csd.auth.gr

***Abstract*—One of the main challenges in distributed systems is efficient scheduling of parallel applications. This paper addresses issues of scheduling parallel jobs that are bags-of-tasks on a cluster of distributed processors. It considers task group scheduling algorithms in two different cases of task routing. The goal is to examine the performance of the scheduling strategies in each of the task routing cases. Simulation is employed to evaluate the performance of the algorithms in different cases of system load. The simulation results reveal that the impact of the scheduling policies on the system performance depends on the employed routing policy, as well as the system load.**

Keywords—bags-of-tasks; scheduling; distributed systems; simulation; performance evaluation

I. INTRODUCTION

In distributed systems, it is important to efficiently execute workloads consisting of parallel jobs that have different degrees of parallelism [1-4]. It is important to partition the parallel programs into tasks, and assign and schedule the parallel tasks to the processors. One solution is to schedule groups of tasks belonging to the same job on their assigned processors. Scheduling techniques must improve the performance of the system, without compromising individual job performance by causing large delays to some jobs.

In distributed architectures, it is desirable for the workload to be distributed evenly to the processors, so that all of the computational resources are efficiently utilized. Therefore, efficient task routing techniques are of great importance.

Jobs in distributed systems often consist of parallel, independent component tasks that can be processed in any order. Such jobs are commonly referred to as *bag-of-tasks (BoT)* jobs [5-7]. A BoT job is considered finished only when the entire set of its component tasks are complete.

BoT scheduling in distributed platforms has been extensively studied in the literature for many years now, as a wide spectrum of applications in science, engineering, astronomy and business analytics are characterized as BoTs. In [8], the authors propose and evaluate a BoT assignment policy in multiple clouds. In [9], an optimization technique is proposed in order to minimize cost and maximize resource utilization in clouds, by placing bag-of-tasks in idle schedule gaps. Bags-of-tasks in parallel systems are studied in [10], while an energy-aware technique for the scheduling of BoT jobs in distributed platforms is proposed and studied in [11].

In the vast majority of research on distributed systems scheduling, it is assumed that the scheduling overhead is negligible. However, the impact of scheduling overhead can be significant and may lead to serious performance degradation. While the *First-Come-First-Served (FCFS)* policy incurs no overhead since it is the simplest scheduling technique, it often results in poor performance.

On the other hand, it has been shown that the *Shortest-Cluster-First (SCF)* policy can yield better results [12], [13]. This method schedules clusters (i.e. groups) of tasks belonging to the same job, based on their cumulative service demand or on the number of tasks in a cluster. However, this scheduling strategy has three drawbacks: (a) it is difficult to implement because it requires a priori knowledge of the service demand of each task, which is not always possible to estimate, (b) it incurs significant overhead, due to the fact that it rearranges the processor queues whenever a new task group is added, and (c) it may lead a task group to starvation, in case its cumulative service demand or its number of tasks is large.

In this paper, we investigate the performance of bags-of-tasks scheduling policies. We consider approximate service times for tasks and therefore for groups of tasks. Along with the *Approximate-Shortest-Cumulative-Time-First (ASCTF)* policy, the *Approximate-Periodic-SCTF (APSCTF)* scheduling technique is also considered. With the latter strategy, the queues of the processors are rearranged only at the end of predefined intervals, where the time interval between two consecutive rearrangements is commonly referred to as a *period* or *epoch* [14-16]. At the end of each period, the priorities of all task groups in the processor queues are recalculated by the scheduler, utilizing the ASCTF strategy. Our goal is to examine the system performance in terms of job response time and the scheduling overhead in terms of the number of queue rearrangements.

Furthermore, two cases of task assignment to processor queues are investigated: (a) one probabilistic task routing policy and (b) a routing policy that employs the 2 random choices technique proposed by Mitzenmacher in [17]. The latter policy is employed, because it is demonstrated in [17] that "in a simple dynamic load balancing model allowing customers to choose between the shortest of two servers yields an exponential improvement over distributing customers uniformly at random".

The task group scheduling strategies, in combination with the routing policies, are examined via simulation, for different period sizes and under different cases of system load. None of

978-1-5386-4599-4/18/$31.00 ©2018 IEEE

Authorized licensed use limited to: China University of Petroleum. Downloaded on January 19,2021 at 16:07:20 UTC from IEEE Xplore. Restrictions apply.

our previous works [14-16] examine workloads where the number of tasks per job are powers of 2, as in this study. Furthermore, in this paper along with a static probabilistic routing policy, we also examine a dynamic routing policy that selects the shortest of two randomly selected processor queues. The latter policy is not employed in our previous research. To the best of our knowledge, the investigation of task group scheduling in a distributed environment, within the framework described in this work, has not been investigated in previous studies.

The remainder of the paper is organized as follows: Section II presents the system and workload models. Section III describes the task routing and task group scheduling policies, whereas Section IV introduces the performance metrics. Section V describes the simulation setup and presents and analyzes the simulation results. Section VI concludes the paper, providing future research directions.

II. Problem Formulation and Methodology

A. System and Workload Models

A queuing network model of $P = 16$ homogeneous distributed processors is considered, where each processor serves its own queue (Fig. 1).

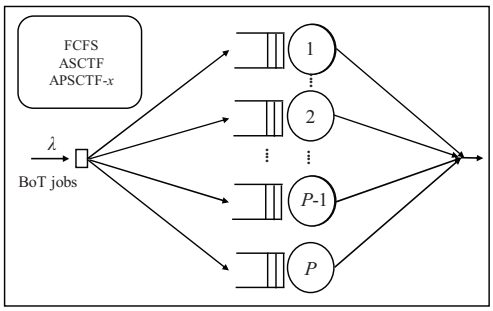

Fig. 1. The queuing network model of the system under study.

The workload consists of BoT jobs with $n \geq 2$ tasks that can be run independently from each other.

The synthetic workload utilized in our case study is defined by the distribution of the following random variables:

- Job inter-arrival time.
- Number of tasks per BoT (job size).
- Service time of BoT tasks.

The distribution of the above workload characteristics is defined below:

1) Job Inter-arrival Time Distribution: Jobs arrive dynamically at the system with inter-arrival times that are exponential with mean $1/\lambda$.

2) Job Size Distribution: The number of tasks per BoT job is given by 2^i, where i is an integer uniformly distributed in the range $[1, \log P]$. Thus, the number of tasks per BoT is a power of 2 in the range $[2^1, 2^{\log P}]$ (uniform-log model).

3) Distribution of Service Demands: The service time of BoT tasks is exponentially distributed with mean $1/\mu$, where μ is the mean service rate of the processors. Table I includes all of the notations used in this paper.

III. Routing and Scheduling

A. Routing

We employ the following two routing strategies:

1) Probabilistic Routing: The tasks of BoT jobs are assigned randomly to processor queues with equal probability. This policy does not involve any overhead and therefore it can be employed in large-scale distributed systems. Sibling tasks of a BoT job that are assigned to the same processor, form a group of tasks and have to be run one after the other. Tasks of a BoT may form multiple groups of tasks on multiple processors.

2) 2 Random Choices Routing: As it is shown in the literature, the shortest queue policy performs well because it takes into account the state of all of the processor queues. However, this policy involves considerable overhead and therefore it is difficult to implement in large-scale distributed systems. For this reason, we chose to employ the 2 Random Choices technique where each task is assigned to the shortest of two randomly selected processor queues [17].

B. Scheduling

We employ the following three scheduling policies:

1) First-Come-First-Served (FCFS): According to this policy, task groups in processor queues are served in the order of their arrival. This method is the simplest to implement as it does not require any information about the state of the processor queues. Furthermore, it is the fairest method of all.

2) Approximate-Shortest-Cumulative-Time-First (ASCTF): According to this strategy, it is assumed that a priori knowledge about the cumulative service time of a group of tasks is available. However, this information is usually not available and only an estimate can be obtained. In this paper, the estimated task service time is assumed to be uniformly distributed within $\pm E\%$ of the exact value, where E is the estimation error. In processor queues, the groups of tasks are ordered in decreasing order of their total approximate service time. This scheduling policy is vulnerable in the extreme case where the approximate cumulative service time of a group of tasks is very large. Since processor queues are rearranged each time a new task group is inserted, it is possible for some jobs to experience unbounded delays.

3) Approximate-Periodic-SCTF-x (APSCTF-x): According to this strategy, processor queues are not rearranged whenerver a new task group is inserted, but only at the end of periods with size x. At the end of each period, the scheduler recalculates the priorities of all task groups placed in processor queues, utilizing the ASCTF criterion.

Authorized licensed use limited to: China University of Petroleum. Downloaded on January 19,2021 at 16:07:20 UTC from IEEE Xplore. Restrictions apply.

IV. Performance Metrics

The response time of a job is defined as the time interval between the dispatching of the job's tasks to processor queues and the completion of the last task of the last group of the job. Therefore, the response time of a BoT includes the synchronization delay of sibling groups of tasks. The performance metrics, as well as the other parameters used in this paper, are shown in Table I.

TABLE I. System parameters and performance metrics

P	Number of processors
η	Average number of tasks per BoT
μ	Mean service rate of BoT tasks
$1/\mu$	Mean service time of BoT tasks
E	Estimation error in service time
λ	Mean arrival rate of jobs
$1/\lambda$	Mean inter-arrival time of jobs
U	Average processor utilization
x	Period length
RT	Average response time of jobs
NQR	Number of queue rearrangements
D_{RT}	Relative (%) decrease in RT when ASCTF or APSCTF-x policy is employed instead of the FCFS policy
D_{NQR}	Relative (%) decrease in NQR when the APSCTF-x policy is employed instead of the ASCTF policy

The RT parameter in Table I, represents a measure of the overall system performance. The D_{NQR} parameter provides an indication of the scheduling overhead.

V. Simulation Setup & Results Analysis

A. Experimental Setup

The performance evaluation of the system under study was conducted via discrete-event simulation, utilizing the independent replication method. For each set of input parameters, we run 30 replications of the simulation program, with different seeds of random numbers in each run. Each replication was terminated when 64000 BoT jobs had been completed. We found by experimentation that this simulation run length was long enough to minimize the effect of warm-up time in the simulation results. For every mean value, a 95% confidence interval was evaluated. The half-widths of all of the confidence intervals were less than 5% of their respective mean values. The utilized simulation input parameters are shown in Table II.

TABLE II. Input parameters

E	10%
$1/\lambda$	0.60, 0.55, 0.50
$1/\mu$	1
x	10, 20, 30

On average, each BoT job consists of η parallel tasks. The average cumulative service time per job is equal to η, where:

$$\eta = (1/\log P) * \sum_{i=1}^{\log P} 2^i = 7.5 \quad (1)$$

Consequently, at each time instance, an average of $P/\eta = 2.13$ jobs can be served, when all of the processors are busy. Therefore, a job arrival rate should be chosen so that $\lambda < 2.13$. This is required in order for the system to be stable. Thus, the mean inter-arrival time of the jobs was chosen so that:

$$1/\lambda > 0.468 \quad (2)$$

In our simulation experiments, the smallest mean inter-arrival time was chosen to be $1/\lambda = 0.50$, in order to avoid very large response times. In the case of the APSCTF-x policy, the smallest period size was chosen to be $x=10$, since the average cumulative service demand of task groups is less than 10. Furthermore, this minimum value was chosen, since it gives a smaller number of queue rearrangements than in the case of ASCTF. Larger period sizes result in even fewer queue rearrangements.

B. Simulation Results Analysis

The performance evaluation of the three task group scheduling policies, FCFS, ASCTF and APSCTF-x, in each of the two routing cases, Probabilistic and 2 Random Choices routing, is analyzed below.

1) Probabilistic Routing Case: Table III shows the mean processor utilization U versus $1/\lambda$, which is the same in all three cases of scheduling policies.

TABLE III. U, Probabilistic routing

$1/\lambda$	0.60	0.55	0.50
U	0.784	0.855	0.938

Fig. 2 shows that the worst performance is presented by the FCFS policy. RT increases with increasing load. At low load ($1/\lambda = 0.6$) and for all period sizes, APSCTF-x yields larger RT than ASCTF. In this case, RT increases with increasing period size. However, at medium load ($1/\lambda = 0.55$) and period size 10, ASCTF performs slightly worse than APSCTF-x. In the high load case ($1/\lambda = 0.50$), ASCTF performs worse than APSCTF-x for all period sizes. In this case, the superiority of APSCTF-x over ASCTF decreases with increasing period size. This may be explained by the fact that with the ASCTF method, task groups that have a small cumulative service time are not delayed by large task groups that precede them in the processor queues. However, this does not necessarily mean that the respective BoT will have a smaller response time. Some sibling task groups of the particular BoT may have to wait longer in other processor queues, because of the ASCTF criterion. This may result in increased synchronization delay and therefore in increased response time of the particular BoT. However, APSCTF-x alleviates this problem by periodically giving the chance to large task groups to be scheduled ahead of smaller groups. Consequently, APSCTF-x is fairer than ASCTF.

Authorized licensed use limited to: China University of Petroleum. Downloaded on January 19,2021 at 16:07:20 UTC from IEEE Xplore. Restrictions apply.

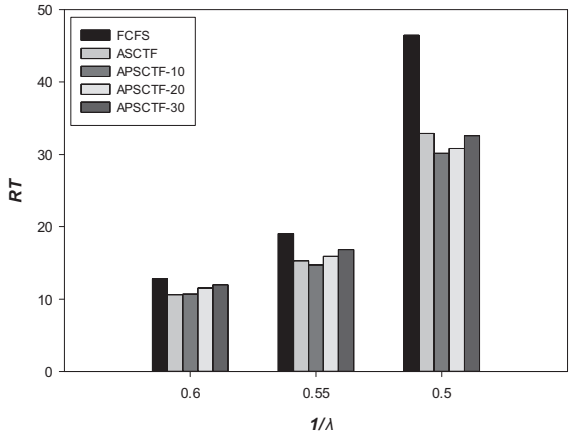

Fig. 2. *RT* versus 1/λ, Probabilistic routing case.

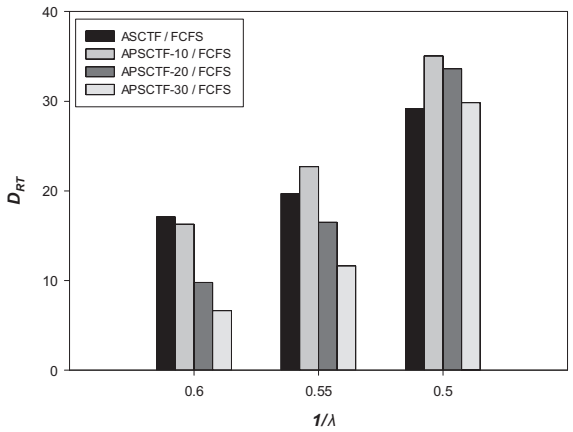

Fig. 3. D_{RT} versus 1/λ, Probabilistic routing case.

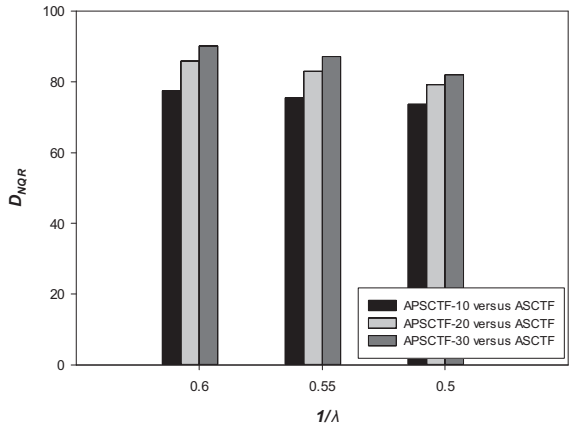

Fig. 4. D_{NQR} versus 1/λ, Probabilistic routing case.

As shown in Fig. 3, the superiority of each policy over FCFS increases as the system load increases. This is because the advantages of each policy over FCFS are better exploited when the system is under heavy load.

For all three cases of system load, the relative decrease in *NQR* due to the utilization of the APSCTF-*x* policy is significant (Fig. 4). For each value of 1/λ, D_{NQR} increases with the increase of the period size. This is because as period size increases, the number of queue rearrangements decreases. The difference in D_{NQR} is larger between smaller period sizes. In general, D_{NQR} decreases as the system load increases.

2) 2 Random Choices Routing Case: Regarding processor utilization, *U* is the same with that in the Probabilistic case. As it is shown in Fig. 2 and Fig. 5, *RT* is significantly smaller in the 2 Random Choices case. Therefore, it is apparent that load distribution affects performance by a larger degree than scheduling of individual queues. As shown in Fig. 5 and Fig. 6, in all three load cases the worst performance is presented with the ASCTF policy. This is due to the fact that this routing strategy makes a better distribution of load in the queues than probabilistic routing. However, the excessive reordering of task groups in the ASCTF policy case deteriorates performance due to the synchronization delay. This deterioration increases as the system load increases. APSCTF-*x* performs slightly better than FCFS, but the largest difference presented is only about 3%. Similarly to the probabilistic case, *RT* increases with increasing load. Also for all period sizes, the superiority of APSCTF-*x* policy over FCFS increases with increasing load.

Fig. 7 shows that for all mean inter-arrival times, D_{NQR} is a little larger in the 2 Random Choices case than when the Probabilistic routing policy is used. It is noted that all of the other observations for D_{NQR} which hold in the Probabilistic routing case, also hold in the 2 Random Choices case.

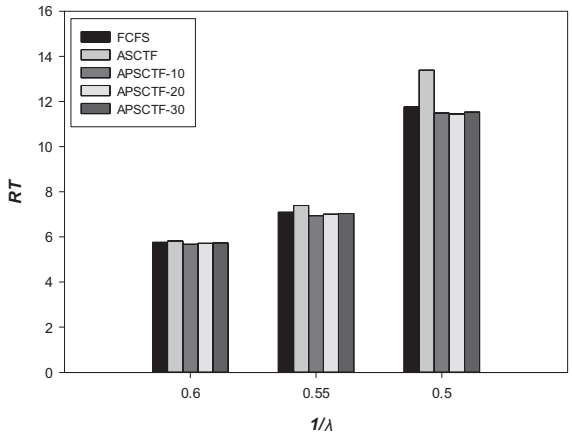

Fig. 5. *RT* versus 1/λ, 2 Random Choices routing case.

VI. Conclusions and Future Research

In this paper, we investigated via simulation the routing and scheduling of BoT jobs in a distributed system. The simulation results show that in all cases, the APSCTF-*x* policy outperforms the ASCTF strategy, in terms of overhead. When Probabilistic routing is utilized, in some cases ASCTF performs better than APSCTF-*x*, in terms of overall system performance. However, its superiority is not significant, considering the scheduling overhead. On the other hand,

Authorized licensed use limited to: China University of Petroleum. Downloaded on January 19,2021 at 16:07:20 UTC from IEEE Xplore. Restrictions apply.

performance is remarkably better when 2 Random Choices routing is employed. In this case, ASCTF is the worst method, while periodic scheduling is only up to about 3% better than FCFS. Therefore, when this routing policy can be employed, FCFS is the preferable scheduling policy, otherwise, periodic scheduling should be chosen in the Probabilistic routing case. Our future research is directed towards examining BoT scheduling performance using different distributions for inter-arrival and service times.

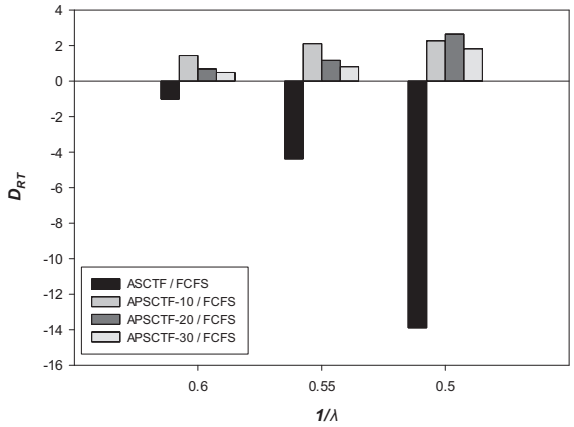

Fig. 6. D_{RT} versus $1/\lambda$, 2 Random Choices routing case.

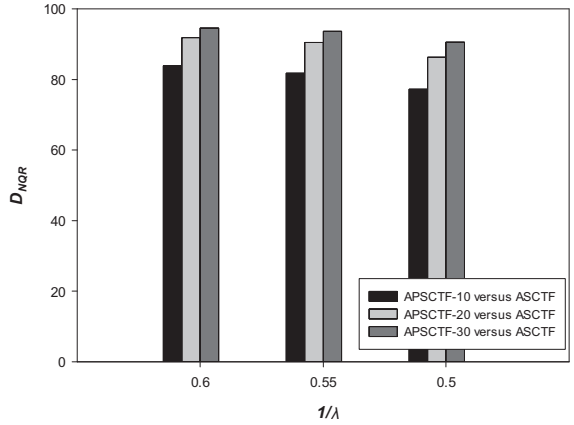

Fig. 7. D_{NQR} versus $1/\lambda$, 2 Random Choices routing case.

REFERENCES

[1] G. L. Stavrinides and H. D. Karatza, "Scheduling multiple task graphs with end-to-end deadlines in distributed real-time systems utilizing imprecise computations," *Journal of Systems and Software*, vol. 83, no. 6, pp. 1004-1014, Jun. 2010.

[2] G. L. Stavrinides and H. D. Karatza, "Scheduling real-time DAGs in heterogeneous clusters by combining imprecise computations and bin packing techniques for the exploitation of schedule holes," *Future Generation Computer Systems*, vol. 28, no. 7, pp. 977-988, Jul. 2012.

[3] G. L. Stavrinides and H. D. Karatza, "The impact of resource heterogeneity on the timeliness of hard real-time complex jobs," in *Proceedings of the 7th International Conference on PErvasive Technologies Related to Assistive Environments (PETRA'14)*, *Workshop on Distributed Sensor Systems for Assistive Environments (Di-Sensa)*, Rhodes, Greece, May 2014, pp. 65:1-65:8.

[4] G. L. Stavrinides and H. D. Karatza, "Scheduling real-time parallel applications in SaaS clouds in the presence of transient software failures," in *Proceedings of the 2016 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'16)*, Montreal, Canada, Jul. 2016, pp. 1-8.

[5] I. A. Moschakis and H. D. Karatza, "Multi-criteria scheduling of bag-of-tasks applications on heterogeneous interlinked clouds with simulated annealing," *Journal of Systems and Software*, vol. 101, pp. 1-14, Mar. 2015.

[6] 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.

[7] Z. Papazachos and H. D. Karatza, "Scheduling bags of tasks and gangs in a distributed system," in *Proceedings of the 2015 International Conference on Computer, Information and Telecommunication Systems (CITS'15)*, Gijón, Spain, Jul. 2015, pp. 1-5.

[8] 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.

[9] 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.

[10] M. Tran and L. Wolters, "Towards a profound analysis of bags-of-tasks in parallel systems and their performance impact," in *Proceedings of the 20th International ACM Symposium on High-Performance Parallel and Distributed Computing (HPDC'11)*, San Jose, CA, Jun. 2011, pp. 111-122.

[11] 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.

[12] H. D. Karatza, "A simulation model of task cluster scheduling in distributed systems," In *Proceedings of the 7th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'99)*, Cape Town, South Africa, Dec. 1999, pp. 163-168.

[13] K. Gkoutioudi and H. D. Karatza, "Task cluster scheduling in a grid system," *Simulation Modelling Practice and Theory*, vol. 18, no. 9, pp.1242-1252, Oct. 2010.

[14] H. D. Karatza**,** "Epoch task cluster scheduling in a distributed system," in *Proceedings of the 2002 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'02)*, San Diego, CA, Jul. 2002, pp. 259-265.

[15] H. D. Karatza, "Periodic task cluster scheduling in distributed systems," in *Computer System Performance Modeling in Perspective*, World Scientific, Imperial College Press, 2006, pp. 257-276.

[16] G. L. Stavrinides and H. D. Karatza, "Periodic scheduling of mixed workload in distributed systems," in *Proceedings of the 23rd ICE/IEEE International Technology Management Conference*, Madeira Island, Portugal, Jun. 2017, pp. 27-29.

[17] M. Mitzenmacher, "The power of two choices in randomized load balancing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, No. 10, Otober 2001.

Authorized licensed use limited to: China University of Petroleum. Downloaded on January 19,2021 at 16:07:20 UTC from IEEE Xplore. Restrictions apply.