A Survey Study and Analysis of Task Scheduling Approaches for the Major Computing Environments

Dalia Kamal A. A. Rizk[1], Hoda M. Hosny[1], El-Sayed M. El-Horbaty[2] and Abdel-Badeeh M. Salem[2]
[1]*Faculty of Computer Science, The American University in Cairo, Egypt*
[2]*Faculty of Computer & Info. Sciences, Ain Shams University, Egypt*

Keywords: Task Scheduling, Cloud Computing, Grid Computing, Fog Computing, Workflow-based, Static Tasks, Makespan.

Abstract: Nowadays, task scheduling is the central point of attraction with respect to cloud computing. Retrieve, store, or compute /analyse data on the cloud are typical types of such tasks. Due to the huge amount of data that are found on the cloud and the need for deep analysis and heavy computation; the importance of task scheduling in an appropriate way for execution increases. In this paper, we present diverse types of algorithms for task scheduling on different environments namely: cloud, grid, and fog under the two widely known types of task representations (task-based and workflow-based).

1 INTRODUCTION

The infrastructure used to grow frameworks like healthcare systems, should give high computing ability, abundant capacity volume, and solid communication with the goal that the task can be completed within the time constraint (Sahoo et al., 2018). Cloud computing is broadly utilized, joined with the advantages of diminishing expense through sharing computing and storage resources. Additional benefits of cloud computing are found in its reliability, security, and scalability features. The task scheduling choice should ensure proficient utilization of cloud infrastructure with least task execution cost conceivable while guaranteeing the application's quality of service (QoS) requirements. Different tasks going to the cloud request a specific kind of Virtual Machine (VM). Because of the idea of open settings in the cloud computing, the needs for information increase rapidly. Accordingly, this issue can be addressed by appropriate usage of tasks alongside accessible resources in order to minimize the required time of execution (Nayak and Padhi, 2019). Most algorithms developed to assign the suitable task to the right resource are task dependent where the most important feature about these scheduling algorithms is to optimize resource utilization. The makespan is the most used measuring factor for almost all scheduling algorithms. It is the total time taken to execute all tasks by the resources. Although, tasks are divided into static or dynamic ones, most studied algorithms dealt with the static tasks only. Furthermore, algorithms could be executed on different environments such as: cloud, grid, or fog computing. Another important aspect that deals with the scheduling process is that the tasks could be either task-based or workflow-based.

Section 2 covers the surveyed task-based representation approaches and associated algorithms. Section 3 covers the surveyed workflow-based representation approaches and their algorithms. Section 4 presents our comparative analysis of the surveyed approaches and section 5 concludes our analysis results.

2 TASK-BASED REPRESENTATION APPROACHES

Task scheduling is an effective method to accomplish a performance perfection of the cloud system. The cloud scheduler should coordinate the heterogeneous tasks to the best-fit VM instance which is picked based on the goal of task scheduling like energy, cost, or time minimization, etc (Sahoo et al., 2018). Task priority or urgency strategy depends on the weight of the task that has a remarkable ability for proficient

Rizk, D., Hosny, H., El-Horbaty, E. and Salem, A.
A Survey Study and Analysis of Task Scheduling Approaches for the Major Computing Environments.
DOI: 10.5220/0010711300003058
In *Proceedings of the 17th International Conference on Web Information Systems and Technologies (WEBIST 2021)*, pages 412-421
ISBN: 978-989-758-536-4; ISSN: 2184-3252

task scheduling and improved output of the system. There are many strategies to assign a priority to a task based on a selected algorithm. The following subsections summarise some of the most widely used algorithms under those strategies within the various environments (Cloud, Grid, and Fog).

2.1 Cloud Computing

Task scheduling is an important way for enhancing the performance of running applications on the cloud.

2.1.1 PCA (2015)

The authors (Al-Sammarraie et al., 2015) here are presenting a schedule algorithm for enhancing the usage of the resources along with providing acceptable cost for the presented services. They achieved their goal by giving priorities to the tasks based on their profits. They took into consideration two important factors of the cloud computing which are the service performance and cost. Their literature survey mentioned a lot of readings, but unfortunately, they just wrote the reference numbers without further explanation. The only explained algorithms in the literature survey were for the ones they are going to use later on for comparison. The proposed algorithm is called the Performance and Cost Algorithm (PCA) because it doesn't only target the minimization of the cost service for the user and the maximization of the profit for the provider, but it also targets the enhancement of the services completion time through minimization and the resources consumption through maximization. Therefore, the proposed algorithm provides cost economical services along with high performance by taking into account the completion time as well as the cost-priority. This is accomplished by optimizing the resource utilization and minimizing the maksepan.

2.1.2 TS-GA (2016)

The authors (Hamad and Omara, 2016) of this manuscript altered the way of execution of the original Genetic Algorithm (GA) to present a new one which minimizes the completion time and the cost of tasks, while maximizes the resource utilization. They introduced task scheduling as one of the resource management topics where the algorithm should consider the huge number of tasks versus the number of available resources on one side, while on the other side the point of interest issues to the Cloud user and provider. Those interest issues could be completion time and cost from the Cloud user's side, and resource utilization from the Cloud provider's side. They studied 8 different genetic algorithms that target modifications in the original GA in order to achieve better performance from the Cloud either user's side or provider's side. According to the authors, task scheduling is an optimization problem that has different parameters as minimum makespan, resources utilization, and minimum cost and it can be solved by a heuristic algorithm.

The original Genetic Algorithm (GA) consists of five steps whereas the proposed Tournament Selection Genetic Algorithm (TS-GA) consists of six steps. The TS-GA applies some modifications to the original GA and introduces a new step which is "Keep Best Solution". This final step uses the new population along with the old one. The first step of the algorithm in the GA is the "Initial Population": which are all individuals that are used in the GA to find out the ideal solution, then after some operations, a new generation is developed based on specific criteria used for the mating chromosomes. The same step is proposed by the new TS-GA, but with defining that the initial population is from a random generation of binary encoding. The second step is the "Fitness Function": which is the motivation factor in the GA as the individual survive or die based on its function value. The TS-GA used the same step by showing the method of computation to achieve a reduction in the completion time for executing all tasks on the existing resources. The third step in the original Genetic algorithm is the "Selection": which is the way to choose the best chromosomes using different strategies: roulette wheel, tournament selection, Boltsmann strategy, or rank selection. The TS-GA used the tournament selection strategy as its third step in order to overcome the impediment of the population size. "Crossover" is the fourth step and it is the creation of new individuals through the hybridization operation in the GA. In the proposed TS-GA, they altered this step to include the parents of the new individuals to the new population as new children. The final step in the original GA is the "Mutation": which is the evolution operator that diverse the gene values when a homogeneous state occurs to the chromosomes. This final step is not found in the proposed TS-GA; however, it is replaced by two other steps: "Initialize Subpopulation" and "Keep Best Solution". The "Initialize Subpopulation" is responsible to add the new generated population after the crossover to the old parent population. While the "Keep Best Solution" returns the solution chosen during the crossover process back to the old population presuming that it has been the satisfying solution for the fitness function.

The authors used the CloudSim toolkit to run their experimental tests by comparing their proposed algorithm results against the Round-Robin (RR) and the original GA. They compared their results using five parameters: completion time, cost, resource utilization, speedup, and efficiency. Their results showed a better performance for all five parameters using their proposed algorithm.

2.1.3 Pricing Models (2016)

The authors (Ibrahim et al., 2016) presented an enhancement algorithm to reduce the makespan and the price of executing independent tasks. According to the authors, they used the static scheduling as it takes into account the pre-getting of the needed data and the pipelining of the various phases of task execution. Static scheduling also forces minimum runtime overhead. The main target of their enhancement algorithm was to allocate users' tasks to VMs based on processing power and taking into consideration the price of the VMs. The authors used Amazon EC2 and Google as their pricing models.

The authors first calculated the total processing power for all existing virtual machines (VMs). Then calculated the total processing power requested by the tasks. Then the proposed algorithm designated for each VM a partial amount of the total power requested based on its power factor. Next, they calculated the ratio between the needed power by the allotted users' tasks and the total processing power of the available resources. Finally, the algorithm calculated the execution time and the price of each task per each VM.

The authors then explained their experimental environment where they used the CloudSim simulator to evaluate the proposed enhancement algorithm and the other algorithms used for comparison. As a Cloudlets benchmark, they used the formatted workload generated by the High-Performance Computing center HPC2N. They compared their results against the default First Come First Services (FCFS), the Genetic Algorithm (GA), and the Particle Swarm Optimization (PSO). The results showed the efficiency of the enhancement algorithm over other algorithms by reducing makespan and decreasing the price of the running tasks.

2.1.4 DPQ-PSO (2017)

The authors (Ben Alla et al., 2017) here proposed a task scheduling algorithm based on Dynamic Priority-Queue (DPQ). The proposed algorithm guaranteed good performance along with task priority and load balancing while improving the resource utilization. The proposed DPQ is based on the Analytic Hierarchy Process (AHP) and Particle Swarm Optimization (PSO). The authors conducted their survey over six previous researches that mainly dealt with prioritization of tasks based on different perspectives. Accordingly, they believed that an appropriate task scheduling algorithm should consider task prioritization depending on multiple criteria.

The authors' main objectives for their proposed algorithm were first to calculate the task prioritization according to the various rules using the AHP. Then communicate these tasks between dynamic Priority-Queues according to the choice distribution and the priority. Finally, the tasks should be scheduled according to the meta-heuristic algorithm PSO. The authors used the Analytic Hierarchy Process (AHP) as it is the most common method for Multi-Criteria Decision Making (MCDM). The importance of AHP lies in its sub-dividing a problem into three levels: objectives, attributes, and alternatives. Moreover, they chose Particle Swarm Optimization (PSO) as their meta-heuristic algorithm because they supposed that it is the best choice against other optimization algorithms for the sake of its easiness in implementation and its capabilities of converging to an acceptable solution.

The architecture of their system starts by receiving tasks and storing them based on their arrival time in the global queue. Then the AHP prioritize the tasks and save them in the global priority queue based on the objective, attributes, or alternative level. The authors used: task length, waiting time, burst time, and deadline as their criteria for the attributes level. Then they used the Dynamic Priority-Queues algorithm (DPQ) to classify the tasks using a quartile method into the prioritize levels: low, medium, high. These prioritized queues reach the scheduler and according to the PSO module in the DPQ algorithm, the queues are scheduled for appropriate VMs. The usage of the PSO guaranteed the execution of tasks with a minimum makespan and cost of resource usage, while maximize the resource utilization.

The authors also used the CloudSim as their evaluator platform for their proposed algorithm and compared their makespan results to those of FCFS and PSO alone. The proposed DPQ-PSO algorithm showed better results in performance and in resource utilization. This is due to the execution of the high priority tasks first with high privilege from the VMs.

2.1.5 TCA (2018)

The authors (Sahoo et al., 2018) proposed a task scheduling algorithm taking into consideration the time and cost constraints. Accordingly, they proposed two versions of their Time and Cost efficient (TCA) scheduling algorithm. The first takes the best fit into consideration while the second takes the first fit.

They presented a scheduling algorithm to minimize both the execution cost and execution time of a task. The Time and Cost efficient (TCA) scheduling algorithm takes into consideration the deadline constraint. They assumed that the cloud consists of a number of VM where each is featured by its speed and execution cost. They featured each task by its arrival time, length or size, and deadline. Accordingly, they considered the heterogeneity of both the task and the VM during mapping each task to an appropriate VM. The scheduler they proposed consisted of a task priority calculator, a real-time controller, and a resource allocator. On the arrival of a new task, the scheduler starts by giving a priority value to the task based on its deadline and the worst-case execution time (WC ET). The following step is to sort these tasks by their priority value. The real-time controller then chooses which task can comply with its time constraint or not. In the event that there is no VM that can complete the task within its time limit, the real-time controller notifies the resource allocator to add new VMs. Finally, if the task's time limit can be met, at that point the resource allocator appoints a suitable VM to the task. Therefore, the real-time controller and the resource allocator cooperate together to fulfil the task's timing prerequisite and afterwards diminish the execution cost by passing on to the best fit VM.

The presented algorithm is divided into two parts: the first computes for each task its priority value through using a calculator module. Then all tasks are sorted by their priority value in descending order. The second part of the algorithm is composed of two for loops nested inside each other; the first one is for each task in the previous sorted list, the second for loop is for each VM. Here, the start time, the execution time, and the finish time are calculated per each VM. Then for each VM, a check for whether the finish time is less than or equal the deadline time is met or not. If it is met, then a utility function is computed. This utility function is used to choose the best VM from both time and cost perspectives versus time only or cost only algorithms. Otherwise, if the finish time of any VM is not less than the deadline time of the task then a new VM is added. Finally, a best fit VM is selected for the current task and then the final combination is saved in the schedule plan (SCH). Then a new task from the sorted list is being introduced to be executed.

They examined the presented algorithm through extensive simulations and experiments. They used 3 measuring factors: guarantee ratio (GR), average cost (AC) and average execution time (AET) to show the effectiveness of TCA over some existing arrangements. The GR is the ratio of the total tasks sustaining their deadline limits against the whole total number of tasks. The AC is the average cost needed to finish a task within the known constraints. Finally, AET is the average execution time for the whole system including all the tasks. In conclusion, the offered algorithm showed best results for the 3 measuring factors against the other two compared algorithms: Heterogeneous Earliest Finish Time (HEFT) and the Cost Aware Algorithm (CAA) where the HEFT depends on the earliest finish time only, while the CAA depends on the execution time only.

2.1.6 Heuristic Approach (2018)

The authors (Gawali and Shinde, 2018) here are presenting a hybrid algorithm as they combined five different technologies to form the proposed heuristic approach. The five used technologies are: the "analytic hierarchy process (AHP)", but they added some modifications to it. The second and third technologies used are the "bandwidth aware divisible scheduling (BATS) + BAR optimization", but combined in one algorithm. The fourth technology they introduced was the "longest expected processing time preemption (LEPT)". Finally, they used the "divide-and-conquer method". Their targeted performance metrics which they measured were the turnaround time and the response time. According to the authors, the turnaround time is the duration of time per each task from its submission to its completion, yet the response time is the required time to check the request against the readiness of the resource. They used both the Cybershake and Epigenomics scientific workflows as their testbeds.

The proposed system started by using the AHP to arrange the tasks based on their length and their runtime and then give ranks to these tasks; however, the authors modified this segment by calculating new ranks for the succeeding tasks to the current tasks at the server. The following step in the proposed system is to allocate the cloud computing resources mainly the CPU, memory, and bandwidth by using the BATS algorithm. However, they also modified this section in order to choose the correct task to be executed next. This modification was done by using the bar systems algorithm (BAR) where they also used a bipartite graph to help in the detection of the next task to be executed. The third step in the proposed system is the preemption methodology which is conducted through the LEPT policy. This policy deals with checking the usage of a virtual machine when there is accumulation of tasks on this VM, then a preemption is required in order to redistribute the waiting tasks to other virtual machines. The final stage in the proposed system is the divide-and-conquer technique which takes the waiting tasks in the previous step and redistribute them among other VMs.

According to the authors, using this presented heuristic approach, they covered both aspects of task scheduling and resource allocation simultaneously. They tested their system against the existing BATS and IDEA frameworks using the Cybershake and Epigenomics as their testbeds. The carried assessments were to evaluate the turnaround time, the response time, and the utilization of CPU, memory, and bandwidth. The results of their proposed system showed justifiable results than the existing BATS and IDEA frameworks.

2.1.7 Mapping Independent Tasks (2019)

The authors (Nayak and Padhi, 2019) proposed an algorithm which is intended to manage variable length tasks by taking the upsides of the distinctive heuristic algorithms and guarantees ideal task scheduling with difference accessible resources to upgrade the quality of the medical services framework. According to the authors, they knew that the transferred data in the case of cloud computing is huge and to guarantee a better service of transferring, there should be a suitable uninterrupted resource scheduling algorithm. This procedure satisfies both the managing of task load along with the dynamic assignment of task while taking into account the non-requirement features such as: availability, flexibility, scalability, and the minimal cost. The scheduling process begins when the user submits a task to the scheduler. This task could be either inserting, processing, or accessing data. Then the scheduler plans the task to the available resource.
The authors listed with description the performance metrics for effective load balancing, but they focused on one specific metrics that they related it to their algorithm which is the “Makespan (MS)”. It tends to be characterized as the maximum time needed for the system to go through the whole data center. It is directly proportional to load balance; therefore, a less makespan means a less load balance and this is a significant quality of acceptable task scheduling algorithm.

The authors discussed some different algorithms such as the MinMin and the MaxMin who use the completion time as their main criteria. The MinMin algorithm is suitable for small/short tasks as they are usually assigned to available resources based on the minimum calculation of completion time. However, large/long tasks are left unattended due to their long completion time. On the other hand, MaxMin algorithm is the opposite of MinMin where it fulfils the large tasks at the favor of the small ones.

Their proposed algorithm is a mixture model in order to overcome the starvation problem of the MinMin and MaxMin algorithms. They execute their algorithm based on the number of small tasks versus the number of large tasks. If the number of small tasks is more than the number of large tasks, then the large tasks are assigned the resources first in order to enhance the efficiency and manage the maximum completion time. Whereas when the number of small tasks is less than the number of large tasks, then the small tasks are assigned the resources first in order to improve the computing performance and to avoid starvation.

2.2 Grid Computing

Grid systems which are one of the parallel computing structures, require task scheduling as one of their most important factors.

2.2.1 MMBLB (2015)

The authors (Daood et al., 2015) here are presenting a task scheduling algorithm which is based on load balancing. Their algorithm takes into consideration that small tasks to be executed on slower resources while the larger tasks be executed on the faster ones. They achieve the load balancing of resources utilization while reducing the total completion time. According to the authors, the presented algorithm is an extension to the Min-Min algorithm, and therefore, they called it: Min-Min Based on Load Balancing (MMBLB) algorithm. The proposed algorithm also, changed the completion time matrix to be called: Expected Sum Completed Time (ESCT).

The proposed algorithm consisted of two phases where the first phase is the Min-Min algorithm, while the second phase overcome the disadvantage of the Min-Min which is improving the load balancing. The Min-Min algorithm is firstly used to assign the small tasks for execution on slower resources. Then the second phase of the proposed algorithm is to concurrently execute the large tasks on the fastest resources. This improvement enhances the chance of simultaneous execution of tasks on resources.

A new performance matrix (ESCT) is also being introduced in this algorithm which is a modification of the Expected Complete Time (ECT) matrix that is being used by almost all scheduling algorithms. This ESCT works on both increasing the load balance and optimizing the makespan outputted from the Min-Min algorithm. It also works on reducing the usage of resources by using this ESCT instead of the ECT and the Expected Execution Time (EET). According to their outcomes which showed an optimization in the makespan and an enhancement in the productivity usage of resources. Consequently, this proved that their development is producing better results than that of using Min-Min algorithm alone.

2.3 Fog Computing

Although, Aladwani (Aladwani, 2019) presented a chapter on a survey about the different task scheduling algorithms for the cloud computing environment, the author previously proposed to use the fog computing especially for healthcare tasks. The author presumed that there will be latency time during the process of handling healthcare tasks over the cloud. The proposed algorithm contains a new method for Task Classification and Virtual Machines Categorization (TCVC) based on tasks importance. Aladwani, classified the tasks to high, medium, and low important tasks based on the patient's health status. She referenced some of the advantages of fog computing for the IoT applications to support the idea of using the fog computing along with the healthcare IoT applications. Moreover, she gave the tasks scheduling algorithms the following definition: "a set of rules and policies used to assign tasks to the suitable resources (CPU, memory, and bandwidth) to get the highest level possible of performance and resources utilization".

The literature review presented by Aladwani, covered 4 reviews only regarding the tasks scheduling algorithms in the fog computing. Aladwani, then moved to the architecture of the healthcare system where she introduced fog computing layer. She referenced what each of the three layers (Devices/Sensors Layer, Fog Computing Layer, Cloud Computing Layer) consists of and how they communicate with each other briefly.

Furthermore, in the motivation section of her research, which was mainly about giving priority to the tasks based on their importance instead of their length. She believed that this means unfairness and load unbalance to tasks that are of higher importance. The algorithm, Aladwani presented to solve this problem is to schedule the tasks after arrival into three groups according to their importance. Then inside each group, the tasks are resorted according to the MAX-MIN scheduling algorithm. Finally, each group of tasks is assigned to the appropriate VMs group to be executed. According to the proposed algorithm, the author assumed that the VMs should be of different capabilities and performance.

Finally, the author showed the simulation that was conducted on the CloudSim simulator of the proposed algorithm and its output versus the output of the MAX-MIN algorithm alone. The comparison conducted by the author, showed that the proposed TCVC scheduling algorithm is better than the MAX-MIN scheduling algorithm alone with regards to the Average Waiting Time (AWT), Average Execution Time (AET), and Average Finish Time (AFT).

Unfortunately, Aladwani's literature review didn't justify why fog computing is better than cloud computing with IoT healthcare applications. It also, didn't mention from where or what was the reference she used for the patient's health status to accordingly classify the tasks to high, medium, and low important tasks. The proposed schedule algorithm classified the tasks based on critical, important, and general tasks without giving the reference health information.

Regarding the survey about the different task scheduling algorithms for the cloud computing, Aladwani (Aladwani, 2020) chose to work on three major static algorithms. The three task scheduling algorithms under investigation were: Fist Come First Service (FCFS), Short Job First (SJF), and the MAX-MIN. According to the author, the parameters used for the measurement of their influence on different tasks were: algorithm complexity, resource availability, Total Waiting Time (TWT), Total Execution Time (TET), and Total Finish Time (TFT).

The author believed that the biggest challenges in cloud computing are task scheduling and load balancing. She further divided the scheduling algorithms into two levels: one at the host and the other at the Virtual Machine (VM), but the author focused on the VM level. Accordingly, the author started mentioning the advantages of all task scheduling algorithms, then explained how algorithms work in the cloud computing environment by being divided into three levels. Cloudlets is the first level where it is the set of tasks that needs to be executed, then mapping the different tasks to the appropriate resources in order to highly utilize the resource while maintaining a minimum makespan. The last level in the task scheduling algorithms is the set of VMs that are used for executing the Cloudlets tasks which is furtherly divided into two steps that the author referenced them.

Further on, the author started discussing the advantages, the disadvantages, and the mechanism of each of the surveyed algorithms (FCFS, SJF, MAX-

MIN). The results for each task scheduling algorithm are also given when using fifteen tasks against six VMs with the assumption of different properties for the VMs. The simulation of these tests was conducted on the CloudSim simulator. Finally, a comparison for the TWT and TFT only for the three static task scheduling algorithms was conducted.

However, throughout the chapter, the calculations given never showed any values to the parameters: algorithm complexity, resource availability, and TET which were mentioned earlier. Another concern in this chapter, is when discussing the arrangement of tasks per each VM in the FCFS algorithm. The author represented the tasks to the VMs differently in the figure that discusses the "FCFS work mechanism" from the table that discusses "waiting times of tasks in FCFS", where she swapped two tasks when waiting for the VMs. Accordingly, there were some errors in the calculations and even in the comparison table.

3 WORKFLOW-BASED REPRESENTATION APPORACHES

The following subsections summarise some of the most widely used algorithms based on the workflow representation.

3.1 ICTS

The authors (Amoon et al., 2018) proposed a workflow-based scheduling algorithm for applications to the virtual machines (VMs) in the cloud computing environment. The proposed algorithm is divided into three parts: level sorting, task prioritization, and then virtual machine selection. They named the proposed algorithm as ICTS that stands for Improved Cost Task Scheduling since their target was to improve the schedule length and to save the monetary cost simultaneously. They introduced the directed acyclic graph (DAG) as the representation used for the workflow in the cloud computing. This DAG consists of nodes and edges. They conducted a survey of eight previous researches regarding the scheduling method in cloud computing. Accordingly, they found that most of the scheduling process targets the minimization of makespan while not taking into consideration the monetary charges.

The algorithm architecture is based on cloud clients presenting their workflow jobs to be executed using the cloud interfaces, then these jobs are subdivided into more basic tasks along with their dependence. The subdivided jobs/ tasks act as nodes in the DAG, while their dependence acts as the edges. Then, the DAG is passed to the Scheduler which is responsible to assign the tasks to the appropriate virtual machines to complete the execution process. It is worth mentioning that the cloud customers pay according to either the number of used VMs and their types or the number of needed CPU cycles. The authors used here the second type which is the number of needed CPU cycles. Therefore, the authors believed that their algorithm has faced a major challenge which is solving the competition between minimizing the time cost while at the same time minimizing the monetary costs.

The algorithm is divided into three stages; the first stage is the level sorting which is conducted through traversing the DAG from top to bottom grouping tasks into levels based on their dependency, and then sorting these levels. The second stage is that for each level; the tasks are ranked through a computed equation then sorted in a new task list based on their decreasing order of ranks. The final stage is that of VM selection, which is conducted through calculating the Makespan-on-Cost Ratio (MKCR) for each task per each virtual machine. Finally, the VM with the largest MKCR gets its related task. According to the authors, this arrangement attempts to profit from the idle time slots between scheduled tasks per each VM to limit the finish time of a DAG. The authors also defined the "time complexity" to be the time needed to appoint each task to a particular VM based on a special priority.

In conclusion, the authors presented their evaluation of the proposed algorithm through the makespan and the monetary costs metrics. They normalized these metrics through calculated equations to get the schedule length ratio (SLR) and the monetary cost ratio (MCR) respectively. They used random generated number of tasks that ranged from 80 to 400 tasks per a DAG. They compared their performance results to that of the Hybrid algorithm. As a result, the ICTS proposed algorithm showed better results for both SLR and MCR over the different number of ranges of tasks than the Hybrid algorithm. Their justification was because the ICTS when calculating the rank of any task took into account both the correspondence times between the parent tasks and this task in addition to the correspondence times between the same task and its successors. Accordingly, the most complex tasks in the DAG will be sorted at the top of the ranked list in the ICTS and thus processed first. On the other hand, in the Hybrid algorithm, the calculation of the task's rank was only for the correspondence times between a task and its successors only.

3.2 NMMWS

This paper (Gupta et al., 2018) dealt with generating a schedule plan for mapping tasks of a workflow to active virtual machines on a cloud server while achieving their two goals: minimize the makespan and maximize the average cloud utilization. The proposed algorithm used the min-max normalization, then calculated a dynamic threshold for the assignment of tasks to virtual machines (VMs). The workflows they used are of course represented by a directed acyclic graph (DAG) where the nodes represent the tasks and the edges represent the data dependency. The authors stated that: "The problem of workflow scheduling in cloud computing is to map each task to a suitable VM and to schedule the tasks for their execution." Accordingly, they did an extensive survey regarding diverse aspects such as: different scheduling algorithms either for independent tasks or for workflow applications, least makespan and maximum cloud resource utilization, cost optimality, minimum energy consumption, or maximum reliability. Since the authors used the workflows scheduling then they used the communication-to-computation ratio (CCR) value in order to categorize the workflows into data-intensive and/or compute-intensive. The lower the CCR, then the workflow is a compute-intensive; the higher the CCR, then it is a data-intensive.

The authors used performance parameters for evaluating the effectiveness of the proposed algorithm against the other existing algorithms. The definitions given by the authors for these parameters; the makespan (MS) and the average cloud utilization (CU), are as follows:

Makespan (MS): is the complete processing time of a workflow application.

Average Cloud Utilization (CU): is the average utilization of all installed VMs on a single cloud server where the utilization of the VMs is the time consumption rate of any VM relating to the cloud server.

The algorithm presumably consisted of the following "actors":

1. Workflow manager: who receives a workflow, then splits it into separate tasks to be forwarded to the "Global Cloud Manager".
2. Global cloud manager: acts as the cloud scheduler which receives the tasks and starts scheduling them to the VMs.
3. VM manager: is responsible for creating and deleting VMs.

The authors used matrices to represent the Data Transfer Time (DTT) and the Estimated Computation Time (ECT). For the DTT, they used the upper triangular matrix to represent the time for data transfer between task nodes. The assumption they did regarding the DTT was that for any two VMs belonging to the same cloud server, the DTT is negligible. For the ECT, which is the execution time taken per each task on the different VMs can be calculated through using any estimation routine technique. Although the authors referenced two types of these routine techniques, but unfortunately, they didn't mention which one they used for their calculation. Other assumptions taken by the authors were that the runtime of the tasks should be known prior, a task per each VM at a particular time, and no stopping for any task once allocated to a VM.

The main algorithm calls three different sub-algorithms. The first module of the proposed workflow scheduling algorithm is the min-max normalization of the estimated computation time (ECT) for all tasks. The second sub-algorithm is responsible for the computation of the Earliest Finish time (EFT). Then the last module uses the dynamic threshold value for dividing the tasks in the ready queue into small and large batches. These batches are assigned to the VMs in the main algorithm as their final step. The authors did a profound simulation on various benchmarks: scientific or real-life workflows such as: Cybershake, Epigenomic, Inspiral, Sipht, and Montage. The results of the simulation were compared against four well-known algorithms namely: heterogeneous earliest finish time (HEFT), dynamic level scheduling (DLS), Min-Min, and Max-Min. According to the results, they reported from the simulation of the proposed algorithm NMMWS against the others, it was shown that the NMMWS is providing far better results for both the makespan and the average cloud utilization. Finally, the authors did another hypothetical test using the statistical ANOVA for validating the results which also showed remarkable results for their proposed NMMWS.

4 COMPARATIVE ANALYSIS

All the aforementioned scheduling algorithms dealt with how to arrange tasks for execution in the different computing environments. Table 1 summarises our comparative analysis of the characteristics of each approach and its points of strength.

Table 1: Comparison between the studied task scheduling algorithms under the studied approaches.

Approach	Reference	Characteristics	Points of Strength
PCA	(AL-Sammarraie et al., 2015)	The proposed algorithm works on optimizing the resource utilization as well as minimizing the maksepan.	Their results of applying the algorithm and comparing them with previous known algorithms showed better performance in makespan and resource utilization.
TS-GA	(Hamad and Omara, 2016)	The proposed algorithm is a modification to the original Genetic Algorithm (GA). The algorithm also contains a new step: “Keep Best Solution” which uses the new population solution along with the old one.	The results of their proposed algorithm showed a minimization of completion time and cost of tasks, and maximization of resource utilization at the same time.
Pricing Models	(Ibrahim et al., 2016)	It deals with the processing power of virtual machines versus the processing power required by the users’ tasks as well as makespan and cost.	They gave evidence of the success of their novice calculation of the processing power in addition to the makespan and cost reduction effectiveness.
DPQ-PSO	(Ben Alla et al., 2017)	The proposed algorithm is a task scheduling algorithm which guarantees good performance while taking into consideration task priority and load balancing which together improved the resource utilization.	Application of the Priority of tasks and appropriate scheduling while achieving a minimum makespan and cost of resource usage and maximum resource utilization.
TCA	(Sahoo et al., 2018)	Two algorithms: best fit and first fit. The features required for each task were: arrival time, length / size, deadline. The proposed algorithm consists of: task priority calculator, a real-time controller, and a resource allocator.	The algorithm considers both the execution time and the execution cost of a task. The utilization function chooses the best fit VM per each task.
Heuristic Approach	(Gawali and Shinde, 2018)	The proposed algorithm is a hybrid one which consists of five technologies namely: AHP (Analytic Hierarchy Process), BATS + Bar, LEPT (Longest Expected Processing Time), and “divide -and-conquer method”	The compared values for the turnaround time, the response time, and the utilization of CPU, memory, and bandwidth showed major progress as promised.
Mapping In-dependent Tasks	(Nayak and Padhi, 2019)	The proposed algorithm’s target is to manage tasks based on their variable length. The claim is that the algorithm guarantees ideal task scheduling with difference accessible resources.	The mixture of MinMin and MaxMin showed higher performance of the overall system based on Makespan.
MMBLB	(Daood et al., 2015)	The proposed algorithm addresses the deficiency of the Min-Min algorithm. It works on both small and large tasks simultaneously. They also introduced a new performance matrix (ESCT).	The proposed algorithm tried to overcome the weak point in the original Min-Min algorithm by giving the large tasks concurrent time as the small tasks and hence achieves lower makespan.
TCVC	(Aladwani, 2019)	The proposed algorithm uses the Fog computing instead of Cloud computing. The algorithm sorted the tasks based on their importance instead of their length.	Using the CloudSim simulator the results showed lower Average Waiting Time (AWT), Average Execution Time (AET), and Average Finish Time (AFT) from the output of the proposed algorithm compared to the MaxMin algorithm.
ICTS	(Amoon et al., 2018)	It is a workflow scheduling algorithm which is divided into: level sorting, task prioritization, and virtual machine selection.	The proposed algorithm took into account both the makespan and the monetary costs metrics and showed improved results in both measurements.
NMMWS	(Gupta et al., 2018)	It is a workflow scheduling algorithm where they first used the min-max normalization. Then a dynamic threshold is calculated for the assignment of tasks to the virtual machines.	The communication-to-computation ratio (CCR) value used to categorize the workflows. The proposed algorithm showed better results for both the makespan and average cloud utilization.

5 CONCLUSION

In this survey, we tried to present and compare the most widely applied types of static task related scheduling algorithms under the different environments: cloud, grid, and fog. We showed the different representations of tasks under the task-based and workflow-based approaches. Based on this analysis, we found that the most effective measurement factors are: optimization (resource utilization), makespan, and cost. Other measurement indicators could be load balancing, processing power, ESCT, and CCR. The algorithms which paid more attention to those measurement factors are: TS-GA, DPQ-PSO, and TCA. Of course, other algorithms consider either one or more of these factors, but not all the three factors together. Another important issue that was noticed through this analysis is that some algorithms prefer to do task prioritization based on different aspects. Some use the length of the task such as the DPQ-PSO and Heuristic Approach algorithms. Others use the importance of tasks such as TCVC and ICTS whereas ICTS uses a specific computed equation to rank the tasks. Another algorithm, the TCA uses the deadline and the WC ET as their reference for the prioritization of tasks. The aim of the above comparison is not to give any ranking to the methods, but rather to help the developers select the most suitable algorithm for their appropriate requirements and needs. Hence, based on the developer's preference of the factors' priority, the most relevant approach and associated algorithm would be selected. For example, if resource utilization is of highest preference, then the task-based approach may be selected with the PCA algorithm. Whereas if all three measurement factors are required with equal priority, then the task-based approach along with any of the three algorithms (TS-GA, DPQ-PSO, and TCA) may be applied.

REFERENCES

Aladwani, T. (2019). Scheduling IoT Healthcare Tasks in Fog Computing Based on their Importance. *16th International Learning & Technology Conference*, 560-569.

Aladwani, T. (2020). Types of Task Scheduling Algorithms in Cloud Computing Environment. *Chapter at IntechOpen*: 10.5772/intechopen.86873.

AL-Sammarraie, N., Al-Rahmawy, M., Rashad, M. (2015). A Scheduling Algorithm to Enhance the Performance and the Cost of Cloud Services. *International Journal of Intelligent Computing and Information Sciences*, 15(1), 1-14.

Amoon, M., El-Bahnasawy, N., ElKazaz, M. (2018). An efficient cost-based algorithm for scheduling workflow tasks in cloud computing system. *Neural Computing and Applications*, 31, 1353- 1363.

Ben Alla, H., Ben Alla, S., Ezzati, A., Touhafi, A. (2017). An Efficient Dynamic Priority-Queue Algorithm Based on AHP and PSO for Task Scheduling in Cloud Computing. *Proceedings of the 16th International Conference on Hybrid Intelligent Systems* (pp. 134-143).

Daood, J., Abuelenin, S., Elmougy, S. (2015). Enhanced Min-Min Task Scheduling Algorithm Based on Load Balancing in Grid Computing. *International Journal of Intelligent Computing and Information Sciences*, 15(2), 15-29.

Gawali, M., Shinde, S. (2018). Task Scheduling and Resource Allocation in Cloud Computing Using a Heuristic Approach. *Journal of Cloud Computing: Advances, Systems and Applications*, 7(4), 1-16.

Gupta, I., Kumar, M., Jana, P. (2018). Efficient Workflow Scheduling Algorithm for Cloud Computing System: A Dynamic Priority-Based Approach. *Arabia Journal for Science and Engineering*, 43, 7945-7960.

Hamad, S., Omara, F. (2016). Genetic-Based Task Scheduling Algorithm in Cloud Computing Environment. *International Journal of Advanced Computer Science and Applications*, 7(4), 550-556.

Ibrahim, E., El-Bahnasawy, N., Omara, F. (2016). Task Scheduling Algorithm in Cloud Computing Environment Based on Cloud Pricing Models. *World Symposium on Computer Applications & Research* (pp. 65-71).

Nayak, B., Padhi, S. (2019). Mapping of Independent Tasks in the Cloud Computing Environment. *International Journal of Advanced Computer Science and Applications*, 10(8), 314-318.

Sahoo, S., Sahu, S., Rath, T., Sahoo, B., Turuk, A. (2018). TCA: A Multi Constraint Real-Time Task Scheduling Algorithm for Heterogeneous Cloud Environment. *International Conference on Information Technology* (pp. 132-136).