Al-Azhar Bulletin of Science: Section B, Vol. 33, No. 1, (June) 2022, pp. 57-63
http://doi.10.21608/absb.2022.110327.1159

Al-Azhar Bulletin of Science: Section B

A NEW ALGORITHM FOR MINIMIZING MAKESPAN WITHIN CLOUD COMPUTING

Afaf A. Abdelhafiz *

Department of Mathematics, Division of Computer Science, Faculty of Science, Al-Azhar University (Branch of Girls), Cairo, Egypt.

* Corresponding Author: Afaf2azhar@azhar.edu.eg

Received: 07 Jan 2022; Revised: 25 Feb 2022; Accepted: 27 Feb 2022; Published: 01 Jun 2022

ABSTRACT

The environment of cloud computing has become widely used in a variety of applications and fields in recent years. Task and resource scheduling, on the other hand, is an area where there is still room for development. Task scheduling methods that allow the mapping of incoming tasks to resources are required to meet good performance data mapping in a heterogeneous computing system. Makespan is reduced and resource usage is maximized when resources and tasks are efficiently mapped. A novel scheduling approach is proposed in this work, which improves the makespan. There are two phases to the recommended method. The Tuples algorithm is used in the first phase that schedules tasks on resources. The second phase rearranges some tasks in order to improve the overall timeframe. The outcomes of the simulation show that the new approach for heterogeneous systems outperforms MASA, max-min, e-MASA, Tuples and Enhanced max-min algorithms in terms of makespan and time complexity.

Keywords: Scheduling algorithm; Max-min; Tuples; Makespan; Complexity.

1. Introduction

Cloud computing technology demonstrates an essential step forward in the development of a variety of applications. It uses many resources, including storage, processors, networks, memory, and applications that are provisioned as services and knowledge technology (IT) resources. Cloud computing shares resources between users of the cloud to provide various aspects of computing. These resources are often accessed at any time and from anywhere. Resource allocation (RA) is a vital aspect in cloud computing. If it's handled accurately, an efficient environment is obtained. Otherwise, services are delayed and starved. Task scheduling is considered to be an NP-complete problem which has developed as one of the attentions in cloud computing [1]. Cloud computing provides an excessive number of services supplied on-demand basis. A main issue in cloud computing is to reduce the overall time as possible which is an interesting task. Kinds of scheduling are centralized and distributed. Centralized scheduling aims to rise the general system performance. On the other hand, distributed scheduling goals to rise the user act. Cloud computing necessitates the use of a distributed and centralized scheduler, making cloud computing scheduling more difficult [2]. Cloud computing datacenters are made up of numerous physical machines that can be heterogeneous or homogeneous. The requests are allocated to physical machines after receiving them from users. Virtualization in cloud computing is actually a very momentous tactic. Virtual machines (VMs) are really created dynamically by allocating one or more

physical machines [3]. The most frequently objective in the cloud distributed systems is to decrease the execution time which is attained by a suitable allocation of tasks on the suitable resources [4].

The main contributions of this paper is the proposal of an approach for enhancing the makespan.

The following shows how this work is planned. The review of literature is mentioned in Section 2. The proposed MMCC (Minimizing Makespan within Cloud Computing) algorithm is discussed in Section 3. Section 4 contains the simulation results. Section 5 brings the paper to a close and outlines the future research.

2. Literature review

Different task scheduling algorithms for cloud computing have been proposed by many researchers. Researchers in [5] introduced a SWOT analysis of the cloud computing that can be used virtually in every industry to improve the service delivery and improvement. The work in [6] empirically compares and offers an insight into the performance of some renown state-of-the-art task scheduling heuristics concerning the throughput, average resource utilization ratio and makespan. Those approaches include task-aware, resource-aware, and some hybrid approaches. Authors of [7] proposed an algorithm for heterogeneous cloud environment that outperforms some existing algorithms by improving load balancing by 43.49% and 72.59%, respectively in average, enhancing resource utilization by 2.28% and 5.61%, respectively in average, and reducing makespan by 7.55%, and 3.75% respectively in average. Researchers in [8] proposed an Effective Load Balancing Algorithm with Deadline constraint (ELBAD). It allocates nearest deadline tasks each time to the highest speed Virtual Machines (VMs) then it balances workload among VMs. They compared their proposed algorithm with other existing algorithms and experimental results showed superiority of ELBAD over others with respect to minimizing makespan and maximizing resource utilization. The cost-efficient algorithm in [9] chooses the most suitable approach in a cloud environment to workflow implementing. In [10], Li et. al proposed a scheduling algorithm for large graphs that put cost together with schedule length into consideration. On the other hand, failed resources are not considered in their algorithm. Kim et al. [11] offered an optimization based on biogeography for scheduling tasks. Its performance is superior to last optimization algorithms like PSO (particle swarm optimization), GA (genetic algorithm) and SA (simulated annealing) used for problems of huge size. The jobs are scheduled here to clouds instead of VMs. Recently, Li et al. [12] proposed cloud list scheduling (CLS) and cloud min-min scheduling (CMMS). CLS is known as a single-phase scheduling whereas CMMS is a two-phase scheduling. CLS and CMMS allocate the tasks to the clouds without regarding the creation of VMs. The min-min algorithm schedules independent tasks on resources and initiates by selecting the smallest finishing time for all task in the set of unscheduled tasks U. Then, the least value of finishing time from the above ones is selected and the task having this finishing time is scheduled on the corresponding resource giving this minimum value. This task is then removed from U. After that, updating ready times for above resource is done. This algorithm has time complexity O (n^2m) [13]. The Max-Min algorithm in [14] begins by determining the smallest completion time for all tasks as in algorithm min-min. The following step is to select the maximum completion time from these minimums and schedule the task giving this maximum on the analogous resource. Update U by deleting this task. Ready times are also updated for the chosen resource. All tasks are mapped to their resources using these steps until the set U becomes empty. It has time complexity O (n^2m). The procedure of algorithm Enhanced max-min Task Scheduling is to select the task having (average execution time or nearest greater than average) and maps

it to the lowest-speed resource producing least completion time for this task [15]. Then, update the set U of unscheduled tasks by deleting this task. Also, all the ready times are updated for this resource. For scheduling the remaining unscheduled tasks, the max-min original algorithm is executed. The time complexity for this algorithm is also O (n^2m). An improvement for the makespan is done by the MASA algorithm [16], where in some cases, the highest average of execution times on resources may be a high value. For this case, algorithm MASA (Minimum Average Scheduling Algorithm) starts with picking out ⌊m/7⌋ tasks (dividing the number of resources by 7 and take its floor) having (least average execution times or nearest greater than least average execution times). Schedule these tasks for the corresponding resources. The old max-min procedure is used for scheduling the residual unscheduled tasks. An algorithm e-MASA (Enhancing the Minimum Average Scheduling Algorithm) in [17] enhances the makespan produced by the MASA algorithm. Select the task that has finishing time which is equal (or closest) to the arithmetic mean of the smallest finishing times of the outstanding tasks instead of selecting the task with maximum completion time. The algorithm ACTA (Average Completion Time Algorithm) in [18] computes the least completion time for every task in the set U of unscheduled tasks. The task which has its finishing time equals (or closest) to the arithmetic mean of the least finishing times of the remaining unscheduled tasks in U is picked. Picked task is given to the resultant corresponding resource. The procedure is recurring till all tasks in U are scheduled. The HASA (Half the Average Scheduling Algorithm) algorithm in [19] started by picking a task with smaller completion time than that chosen in ACTA. HASA first computes the finishing time for all tasks on all resources. Every stage, pick the task having finishing equal (or closest) to 1/2 (arithmetic mean) of the least finishing times of unscheduled tasks residual in the set U. The picked task is given to the resultant resource and erased from the set of unscheduled tasks U. Updating ready times of the chosen resource is done. This procedure is recurring until the set U of unscheduled tasks becomes empty. Most of the previously discussed algorithms have the aim of improving the makespan with a complexity of O (n^2m). However, Tuples algorithm in [20] has a different aim which is improving the running time. The Tuples idea is to allocate m task to their suitable resources into tuples (m task at a time). It picks for every resource, a task with least completion time. Picked task is erased after that from unscheduled tasks with modifying the finishing times for the analogous resource. These steps are recurring until assigning all unscheduled tasks completely. This algorithm has O (n^2) time complexity.

3. Proposed MMCC (Minimum Makespan algorithm within Cloud Computing)

Usually, the users send their service requests with the support of cloud manager. Then, the cloud manager collects the service requests from the customers where it retains the total number of worked VMs in the cloud. Also, the cloud manager works on steps of the scheduling policy found in the cloud. Heterogeneity of VMs is well-known where they possesses unlike specifications and computing abilities. The proposed algorithm MMCC consists of two phases. The first phase applies the Tuples algorithm in [15], where m service requests (tasks) having least completion times are selected and allocated to the m corresponding resources. The procedure is repeated until all unscheduled tasks in U are allocated. In the second phase, a task is chosen which have the greatest completion time and a search is done for this task to find another resource on which it has a minimum execution time such that its new completion time is less than the old makespan. Then, this selected task is rescheduled on the new resource. This procedure in the second phase is repeated m/20 times (is discussed later), where m is the number of resources. Let Eij, ti, Rj, Cij and rj denote processing time, task i, resource j,

finishing time of ti on Rj and finishing (ready) time of resource Rj respectively.

3.1 Algorithm MMCC

1) Enter the processing time Eij for every service request (task) on all resources
2) For every service request ti in U
3) For every resources Rj
4) Cij←Eij+rj
5) While there are unscheduled tasks in U
6) For every resource,
7) Determine the service request (task) with least finishing time and give it to its resource
8) Eliminate this service request from U
9) Update the ready times related to this resource
10) End For
11) End While
12) For I=1 to m/20// Rescheduling phase
13) Find service request ti having greatest completion time
14) Find the resource that possess a minimum execution time for ti such that its new finishing time not greater than or equal to the old makespan.
15) Reschedule this task to its consistent resource
16) End For

3.2. Complexity Analysis

Suppose that n>m. It is known that the Tuples (lines from 1-11) has complexity O (n^2). With respect to the rescheduling phase, line 12 repeats O(m) times. O (n) steps are needed to find a task having greatest completion time in line 13. It needs O (m) steps in line 14 to find the resource giving minimum execution time. Assigning this task to its new resource needs constant time. The rescheduling phase accordingly needs O (m(n + m))=O(mn) steps as n is greater than m. Henceforth, the complexity of MMCC is O (n^2).

4. Simulation Results

Simulation experiments were run in C++ language. The proposed MMCC algorithm was compared with max-min, Tuples, Enhanced max-min, MASA and e-MASA algorithms. The model for each of these simulations consists of m resources where it ranges from 200 to 1000 resource and n tasks where n ranges between 2000 and 10000. Processing times for these tasks are generated randomly using the predefined C++ function; rand (). To clarify the benefits of MMCC compared to others; Figure 1 schemes the makespan versus the number of tasks where the number of resources is fixed at m=100. The observation is that the MMCC algorithm has enhanced makespan than other different algorithms.

Another study is done to show and explain that 1/20 is the best fraction used in the MMCC algorithm. Figure 3 schemes makespan resulting from MMCC for different values of fractions starting from 1/10 till 1/100 with fixing n=3000 tasks and m=500 resources.

This summary table compares time complexity and makespan for some algorithms with MMCC.

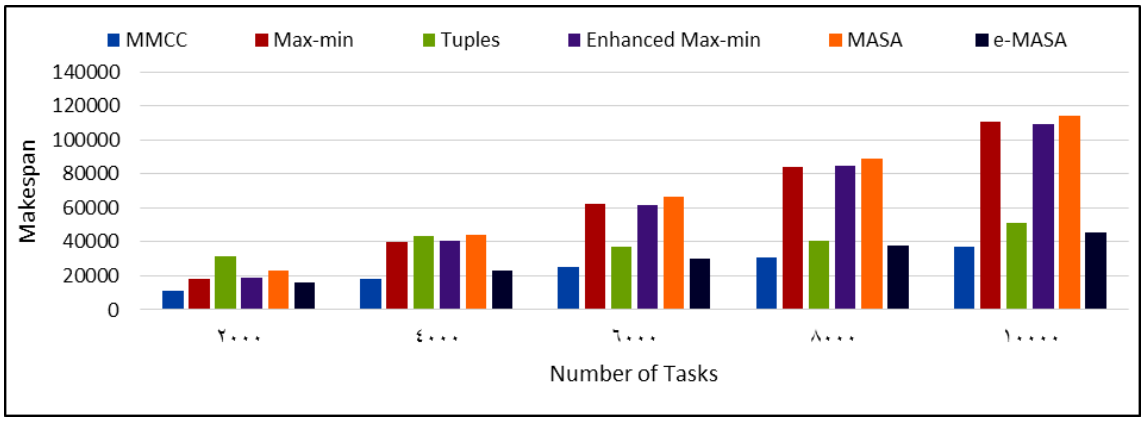

Fig. (1). Makespan sketched with the number of tasks at fixed m

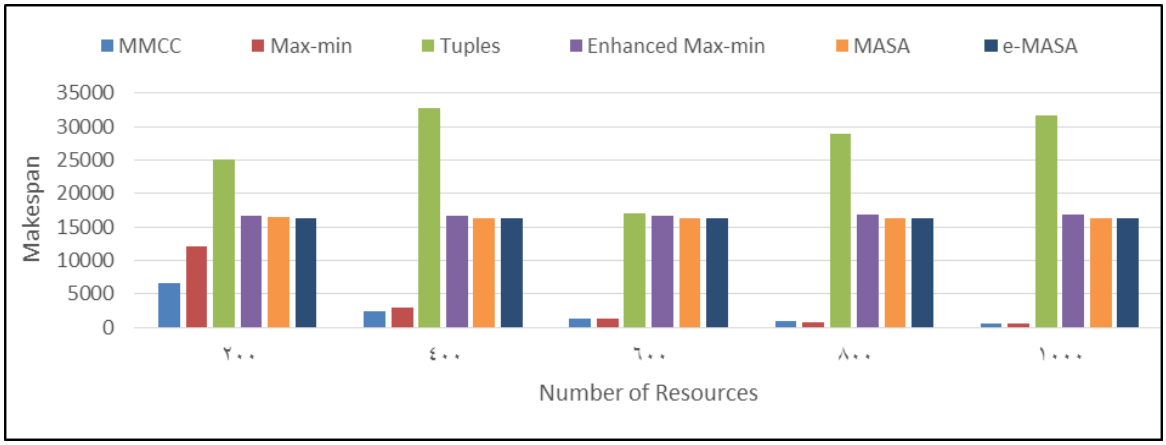

Fig. (2). Makespan against the number m of resources at fixed n

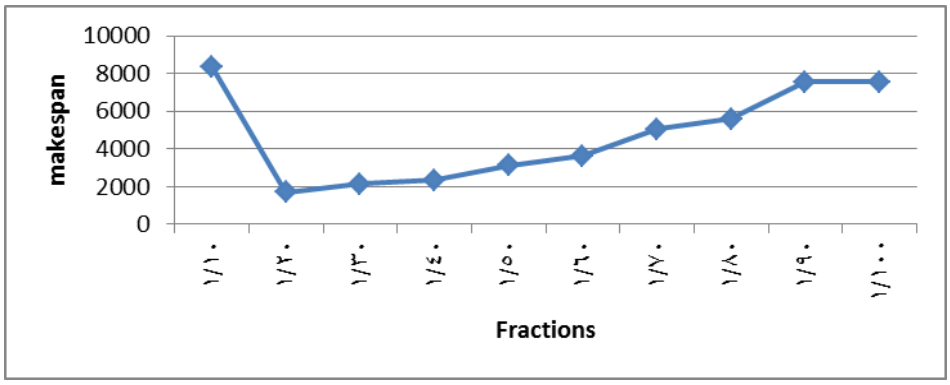

Fig. (3). Makespan versus fractions with fixing n=3000 tasks and m=500 resources

Table 1: Comparing time complexity and makespan for some algorithms with MMCC

Algorithm	**Time complexity**	**Makespan**
Max-min	O (n^2m)	MMCC is better
Improved max-min	O (n^2m)	MMCC is better
Enhanced Max-min Task Scheduling	O (n^2m)	MMCC is better
MASA	O (n^2m)	MMCC is better
e-MASA	O (n^2m)	MMCC is better
Tuples	O (n^2)	MMCC is better
MMCC	O (n^2)	

5. CONCLUSION

In this work, a novel algorithm, Minimum Makespan algorithm for Cloud Computing (MMCC), was offered. It schedules tasks using the Tuples algorithm. Then it reschedules those tasks with greatest finishing time to enhance the resulting makespan. The simulation results showed improved performance in makespan and time complexity when compared to the other scheduling algorithms namely as max-min, Tuples, Enhanced max-min, MASA and e-MASA. The simulation was based on static scheduling, which assumes that task and resource information is known in advance. Dynamic scheduling will be the focus of future research, with the goal of improving time complexity and makespan.

REFERENCES

[1] Sa'ad S, Muhammed A, Abdullahi M, Abdullah A, Ayob FH. An enhanced discrete symbiotic organism search algorithm for optimal task scheduling in the cloud. Algorithms. 2021;14(7):1–24.

[2] Sasikaladevi N. Minimum makespan task scheduling algorithm in cloud computing. Int J Grid Distrib Comput. 2016;9(11):61–70.

[3] Morariu O, Morariu C, Borangiu T. A genetic algorithm for workload scheduling in cloud based e-Learning. Proc 2nd Int Work Cloud Comput Platforms, CloudCP 2012 - Co-located with EuroSys 2012. 2012;1–6.

[4] Sana MU, Li Z. Efficiency aware scheduling techniques in cloud computing: A descriptive literature review. PeerJ Comput Sci. 2021;7:1–37.

[5] Gundu SR, Panem CA, Thimmapuram A. Real-Time Cloud-Based Load Balance Algorithms and an Analysis. SN Comput Sci [Internet]. 2020;1(4):1–9. Available from: https://doi.org/10.1007/s42979-020-00199-8

[6] Ibrahim M, Nabi S, Baz A, Naveed N, Alhakami H. Towards a task and resource aware task scheduling in Cloud Computing: An experimental comparative evaluation. Int J Networked Distrib Comput. 2020;8(3):131–138.

[7] Mahmoud H, Thabet M, Khafagy MH, Omara FA. An efficient load balancing technique for task scheduling in heterogeneous cloud environment. Cluster Comput [Internet]. 2021;24(4):3405–3419. Available from: https://doi.org/10.1007/s10586-021-03334-z

[8] Arif KI. An Effective Load Balancing Algorithm Based on Deadline Constraint under Cloud Computing. IOP Conf Ser Mater Sci Eng. 2020;928(3).

[9] Van Den Bossche R, Vanmechelen K, Broeckhove J. Cost-optimal scheduling in hybrid IaaS clouds for deadline constrained workloads. Proc - 2010 IEEE 3rd Int Conf Cloud Comput CLOUD 2010. 2010;(May 2014):228–235.

[10] Li J, Su S, Cheng X, Huang Q, Zhang Z. Cost-conscious scheduling for large graph processing in the cloud. Proc- 2011 IEEE Int Conf HPCC 2011 - 2011 IEEE Int Work FTDCS 2011 -Workshops 2011 Int Conf UIC 2011- Work 2011 Int Conf ATC 2011. 2011;808–813.

[11] Kim SS, Byeon JH, Yu H, Liu H. Biogeography-based optimization for optimal job scheduling in cloud computing. Appl Math Comput. 2014;247:266–280.

[16] Li J, Qiu M, Ming Z, Quan G, Qin X, Gu Z. Online optimization for scheduling preemptable tasks on IaaS cloud systems. J Parallel Distrib Comput. 2012;72(5):666–677.

[15] Amudha T, Dhivyaprabha TT. QoS priority based scheduling algorithm and proposed framework for task scheduling in a grid environment. Int Conf Recent Trends Inf Technol ICRTIT 2011. 2011;650–655.

[14] Kang L, Ting X. Application of adaptive load balancing algorithm based on minimum traffic in cloud computing architecture. 2015 Int Conf Logist Informatics Serv Sci LISS 2015. 2015;

[15] Bhoi U, Ramanuj P. Enhanced Max-min Task Scheduling Algorithm in Cloud Computing. Int J Appl or Innov … [Internet]. 2013;2(4):259–264. Available from: http://ijaiem.org/Volume2Issue4/IJAIEM-2013-04-30-130.pdf

[16] Eldahshan K, El-kader AA, Ghazy N. Minimum Average Scheduling Algorithm , MASA , Performance Boosting Approach. 2016;(1):23–29.

[17] Elkader AA. Enhancing the Minimum Average Scheduling Algorithm (MASA) based on Makespan Minimizing Set of all Tasks. 2017;(1):9–13.

[18] Abd A. ACTA: Average of Completion Times Algorithm. Int J Comput Appl. 2017;172(8):18–22.

[19] Elhafiz AAEA. HASA: Half The Average Scheduling Algorithm. Circ Comput Sci. 2017;2(9):35–39.

[20] Abdelkader Abdelhafiz A. Tuples: A New Scheduling Algorithm. J Comput. 2018; 13 (11): 1309–1315.

خوارزم جديد لتقليل الماكسبان في الحوسبة الذكية

عفاف ع. عبد الحفيظ

قسم الرياضيات والحاسب الالى - كلية العلوم (فرع البنات) - جامعة الازهر – القاهرة - مصر

الملخص

تم استخدام الحوسبة السحابية في السنوات الأخيرة على نطاق واسع في عدد منتشر من التطبيقات والمجالات .ومع ذلك ، تظل جدولة المهام والموارد جزءًا يحتاج إلى التحسين والتعزيز .فيما يتعلق بنظام الحوسبة غير المتجانسة ، فإن خوارزميات جدولة المهام التي تسمح بنخصيص المهام الواردة إلى الموارد ، مطلوبة لتلبية متطلبات تعيين البيانات عالية الأداء .يؤدي التخصيص الفعال بين الموارد والمهام إلى تقليل الامتداد وزيادة استخدام الموارد .خلال هذه الورقة البحثية ، يتم تقديم خوارزمية جدولة جديدة تقدم تحسينًا للوقت الكلى لاداء المهام. تتكون الخوارزمية المقترحة من مرحلتين .تطبق المرحلة الأولى خوارزمية Tuples إحدى خوارزميات تخصيص المهام للموارد .في المرحلة الثانية ، تتم إعادة الجدولة لبعض المهام لتحسين الوقت الناتج.