Comparative Analysis of Simulated Cloud Task Scheduling

Adrián Hamada
University of Žilina
Žilina, Slovakia
hamada@stud.uniza.sk

Jarmila Škrinárová
Matej Bel University
Banská Bystrica, Slovakia
jarmila.skrinarova@umb.sk

***Abstract*—Cloud systems are autonomous systems, that are managed on the base of complex set of criteria. In clouds systems, where large number of tasks runs on numerous machines, optimized task scheduling causes significant reducing of computing time. The process of scheduling tasks in a cloud computing system consists of two steps. The first step is to allocate computing tasks to virtual machines and the second step is to create virtual machines on physical machines. In two steps scheduling takes place, so we talk about combinations of scheduling models. We introduce a complex analysis of the behavior of a simulated cloud system to find appropriate combinations of scheduling models for scaled and dimensional sets of computational tasks. The tasks have different requirements on the system, such as those that are focused on the number of computational cores. We use the minimum completion time of the last task as the main quality criterion. Based on experiments, 9000 simulations, and subsequent comparative analyses, we find that the scheduling of tasks to virtual machines has a significant impact on makespan. We have shown that the optimized task scheduling (with minimum makespan values), reduces the 6 original classes of combined models to 2 classes. These are on time sharing and space sharing models for scheduling tasks on virtual machines.**

Keywords—cloud computing, task scheduling, analysis

I. INTRODUCTION

Cloud systems are autonomous systems, that are managed on the base of complex set of criteria. In clouds systems, where large number of tasks runs on numerous machines, optimized task scheduling causes significant reducing of computing time. A quality cloud system must respect the task requirements for architecture, number of compute cores, operating system, compilers, and other application software. Task requirements are also related to quality of service, which can be expressed in terms of different time conditions. These are, for example, requirements for the maximum slowdown of a task, or the maximum completion time of a task. It is essential that the system provides the tasks with sufficient computational resource capacity, particularly the computational time of the processors. In a complex system, the requirements of many tasks meet and need to be optimally scheduled. Efficient scheduling of tasks will result in maximum throughput, minimum waiting time for individual tasks, minimum completion time for the last task and meet task requirements. Ensuring the quality criteria, ultimately minimizes the power consumption of the computing system. Clouds are virtualized by virtual machines or containers [11].

The process of scheduling tasks in a computing system consists of:

- allocation computational tasks to virtual machines.
- virtual machine (VM) creation on physical machines (PMs).

Understanding the behavior of a computing system is a prerequisite for achieving high system performance. Therefore, in this paper, we have studied in detail the behavior of the computing system, in a simulated CloudSim environment [1].

The aim is to reach a comprehensive understanding of the simulator's behavior when using various scheduling models and under different loads. This will enable us to modify the simulator's source code with additional functionalities for Edge and Fog computing in the future.

The primary contribution of this work is the comparative analysis required to implement modern heterogeneous cloud systems (Edge, Fog, microclouds) in the simulator and evaluate the quality criteria necessary for managing the system to achieve enhanced performance [11].

The CloudSim simulator allows the creation of scalable simulations [2]. The aim of this paper is a comparative analysis of combined scheduling models in the CloudSim cloud simulator. In this paper, we ask how the processing time of tasks on VMs will be affected if tasks are multitasked (preemptively) switched or run continuously on VMs from start to completion i.e., without interruption. We interpret the results of the analysis to decide which combined scheduling models are appropriate to use to make the processing of the simulation tasks as efficient as possible. The specific objectives of the paper are:

- To perform a thorough analysis and comparison of the combined models of task scheduling on virtual machines in CloudSim simulator.
- To create experiments based on which we compare the behavior of the system when using different scheduling models using scalable datasets of tasks accomplished by specifications of requirements.
- To perform comparative analysis and investigate effect of using different combined scheduling models with varying loads represented by number tasks on simulation runtime when quality criterion is the shortest time of completion of all tasks in a dataset.

We created a simulated cloud system and 30 scaled task load sets that represented different system workloads. The number of tasks in these sets varies from 500 to 15 000 tasks. The tasks had requirements that specified the number of computational cores. For creating task datasets, we inspired by computing tasks, which we have used in the High-Performance Computing Center of the Matej Bel University in Banska Bystrica, Slovakia.

CloudSim supports 6 different combinations of scheduling models, which can (as we will show) be classified into 2 basic groups into time sharing (denoted by Ts) and space sharing (denoted by Ss), depending on how computational tasks are allocated to virtual machines.

We designed and performed experiments to decide which combined task scheduling model to choose for individual types of computing system workloads. We did 9000 simulations, in total. We analyzed the results of each experiment and then compared the behavior of the system using each scheduling combination. Then, we analyzed the behavior of the system when minimal time of task completion was reached. The minimal time represents the best schedule.

We will show that the combined models Ss consume significantly more overhead time to switch tasks to VMs compared to Ts. If the computing system has enough computational capacity to solve the tasks while the task computation quality requirements are met, then the time sharing combined model runs preferentially in the system. If the set of tasks to be processed has requirements on the computing system that cannot be met, the space sharing combined task scheduling model must be used.

We show analysis of the behavior of the computing system with Ss based task scheduling in more details. We show that if the task requests load the system to such an extent that it is not feasible to schedule the tasks with the Ts model, then it is most appropriate to use the combined SsTs model. We show that the optimized scheduling algorithms will enable classification of the system behavior.

This paper is structured as follows: Section II describes related works that deal with task scheduling and VMs. Section III describes the scheduling of tasks and VMs along with the combined task scheduling models. Section IV characterizes computational resources and computational tasks. Section V contains experiments with different types of datasets. Section VI contains an analysis of the behavior of combined scheduling models using minimum simulation times.

II. Related Works

The objective of the present paper [3] was to compare the response times and CPU utilization of VM's for scheduling tasks in CloudSim simulator with different scheduling algorithms. These algorithms were used along with combined time shared and space shared scheduling models. The research authors measured the least response time using shortest job first (SJF) algorithm for task scheduling combined with VM scheduling using round robin (RR) algorithm. They observed the lowest VM's CPU utilization using SJF for task scheduling combined with virtual machine scheduling using the length-wise allocation algorithm (LwA) [3].

The research [4] focused on scheduling tasks in CloudSim cloud simulator through a hybrid algorithm combining the Grey Wolf Optimization Algorithm (GWO) and the Genetic Algorithm (GA). The objective of the research was to minimize the response time, energy consumption and resource utilization cost of the cloud system. Scheduling tasks using hybrid algorithm combining the Grey Wolf Optimization Algorithm was compared with scheduling algorithms GA, GWO and Particle Swarm Optimization (PSO). A dataset of both real and synthetic data was used in the comparison in which the authors confirmed the advantages of the new hybrid algorithm [4].

Metaheuristics are among the most popular methods currently used for optimal task scheduling in heterogeneous cloud systems, but also for optimizing real-world scheduling problems [5]. The present research [5] focused on the analysis and comparison of six biologically inspired metaheuristics. The authors of this paper first performed a detailed analysis of these optimization algorithms. Then, they performed a comparison of the response times and computational resource utilization costs of all six metaheuristic algorithms. From the research result, they found that Crow Search Algorithm is the best method for optimizing the response time parameters and resource utilization cost in cloud system [5].

In the given research [6], the authors load balanced the cloud system using a hybrid algorithm combining the Genetic Algorithm (GA) and the PSO. CloudSim simulator was used in the research. The authors aimed to minimize the total simulation time, total cost of running VMs, average waiting time and system response time. The authors concluded that the hybrid algorithm combining the Genetic Algorithm (GA) and the PSO is the best algorithm to optimize all the compared parameters. Hence, the genetic algorithm should be combined with other metaheuristic methods to improve the efficiency of the optimization algorithm [6].

The research [7] focused on scheduling tasks in CloudSim simulator to save power using Greedy strategy. The authors developed a cloud task clustering method based on three-way decision (TWD-CTC) and compared it with different scheduling algorithms. This method divided the pre-scheduling tasks into three groups. The first group were the tasks with high CPU power requirements. The second group were tasks with high RAM capacity requirements. The third group, called Mix, included tasks that did not make it into the first two groups. This ensured that tasks were allocated to VM's according to the demands on computing resources. In their research, the authors found that using the TWD-CTC scheduling method, the cloud system had the lowest power consumption. Interestingly, they measured that the TWD-CTC scheduling method had three times lower power consumption compared to the Min-Max-Min and Min-Max algorithms [7].

The authors of the given research [8] focused on optimizing the task scheduling in the cloud using Hybrid Cuckoo Search Algorithm (CSA). The comparison was performed using the CloudSim simulator and three real datasets NASA, HPC2N, and SDSC. In that paper, the authors presented the hybrid Nesterov Accelerated Gradient-based Cuckoo Search Algorithm (NAGCSA) and compared it with GA, PSO, and Cuckoo Search Algorithm (CSA). The result of the research was that the hybrid NAGCSA algorithm was able to reduce the task processing time by a maximum of 19.20% and the VM running cost by a maximum of 35% [8].

The authors in [4], [5], [6], [7], [8] discuss the design and implementation of optimized scheduling algorithms in the CloudSim environment. However, unlike our approach, these papers do not delve into combined scheduling models. In the work by Sahkhar et al. [3], four combined task scheduling models are utilized. The authors employed 10 VMs and two datasets with 100 and 500 tasks.

All tasks required a single computational core, and the task lengths ranged from 10 000 to 12 000 MI. Our work provides a comprehensive analysis of a system's behavior based on 100 VMs. We loaded the system with 30 different scaled task datasets with varying sizes from 500 to 15 000 tasks. The tasks required varying numbers of computational cores (1-4), and the task lengths ranged from 10 000 to 50 000 MI.

III. Scheduling of Tasks and VMs

A quality cloud system must respect the requirements of the tasks. It is essential that the system provides the tasks with sufficient computing resource capacity, especially CPU computing time. The requirements of many tasks collide in the system and therefore scheduling needs to be addressed. Since in this paper we focus on scheduling tasks in a system and for purpose of scheduling a dimensional set of computational tasks, it is necessary to specify a set of computational tasks and a set of computational resources, we define the different components involved in this problem in this section of the paper.

Let $J = \{J_1, J_2, \ldots, J_n\}$ is the set of tasks, where the task J_i can be described as part of the total work performed by the system in solving a computational problem. Let $M = \{M_1, M_2, \ldots, M_m\}$ is the set of machines, where a machine or computational unit represents a set of accumulated resources (processors, memories, and storage) [9].

We denote by J_{ij} the j - here job processed on the i-th machine of the system. We denote the completion time of the computation associated with solving job J_{ij} by C_{ij}. The completion time of the last job on the i-th machine is denoted by C_{maxi}, where $C_{maxi} = max\ \{C_{i1}, \ldots, C_{in}\}$. The completion time of the last task on all machines is denoted by C_{max}, where $C_{max} = max\ \{C_{max1}, \ldots, C_{maxm}\}$ [9]. This parameter forms the basic criterion for optimizing the runtime of jobs in the system, on which other system requirements depend. Such requirements, which need to be satisfied to guarantee Quality of Service (QoS) compliance in cloud systems, can be the basic task quality requirements [10]:

- Maximum task slowdown SD_{max} is introduce in relation (1), where S_j is the execution start time of task j, r_j is the time when task j is available for processing, and D_j is the completion time of task j that can be estimated by the client [11].

$$SD(j) = \frac{(S_j - r_j + D_j)}{D_j} \tag{1}$$

- Reserving computing resources in advance - in static job scheduling, the client can agree with the cloud provider to provide the service for a specific time, this will allow the cloud provider to prepare the required resources in advance before starting the required service [12].
- Last possible time T_j^S to execute task j you can see in relation (2).

$$T_j S = r_j + D_j(SD_{max} - 1) \tag{2}$$

Where r_j is the time when job j will be available for processing, D_j is the completion time of job j that is estimated by the client, and SD_{max} is the maximum slowdown of job j. [11].

A. Scheduling Definition

Task scheduling in a computing system is the assignment of processing tasks to resources for a specific time interval so that no two tasks are executed concurrently on the same resource and the capacity of the computing resource is not exceeded. The schedule specifies, for each time instant, the set of tasks to be executed at that instant and the set of resources on which they are to be executed [12], [13], [14].

The process of scheduling tasks in a computing system consists of:

- virtual machine (VM) creation on physical machines (PMs) - determine the method that determines whether VMs will run on physical machines continuously or intermittently,
- allocating compute tasks to virtual machines - determining the method that determines whether tasks are processed on virtual machines as a single unit or in parts. Two basic task scheduling models, referred to as the time shared (Ts) and space shared (Ss) models [15], are implemented in the CloudSim simulator. Both basic scheduling models can be used for mapping virtual machines to physical machines and for allocating computation tasks to virtual machines.

(1) A model for scheduling virtual machine runs on physical machines:

The time shared scheduling model from the perspective of creating virtual machines on physical machines is based on the principle of running virtual machines on physical machines continuously without interruptions. The time shared model works in the same way as non-preemptive process scheduling. When using the space shared scheduling model, the running of virtual machines can be intermittent. For example, at a certain point in time, VM1 runs on a certain physical machine, then VM2, then VM1 again, and so on. The time shared scheduling model, for allocating VMs to physical machines, can be extended with oversubscription (To) - allowing the creation of VMs with more CPU capacity than is available on the physical machines.

(2) A model for scheduling computation tasks on virtual machines

The time-shared scheduling model in terms of allocating computation tasks to virtual machines is implemented in such a way that the processing of tasks is not divided into parts, but the tasks are processed as a single unit. In the case of using the space shared scheduling model, the tasks are divided into multiple parts. Subsequently, the given tasks are also processed in parts (multitasking) [1].

B. Combined Models of Task Scheduling

There are 6 possible combinations of the above scheduling models. We always list the combinations in order, first for scheduling tasks on VMs and then for scheduling virtual machines on physical machines. Therefore, we denote the specified combinations by:

- TsTs - time shared task scheduling model on VMs and time shared VM scheduling model on PMs.

- TsSs - time shared task scheduling model on VMs and space shared VM scheduling model on PMs.
- SsTs - space shared task scheduling model on VMs and time shared VM scheduling model on PMs.
- TsTo - time shared task scheduling model on VMs and time shared oversubscription scheduling model on PMs.
- SsTo - space shared task scheduling model on VMs and time shared oversubscription scheduling model on PMs.
- SsSs - space shared task scheduling model on VMs and space shared VM scheduling model on PMs.

We do not show the combined TsTo, SsTo scheduling models below, as they are the same TsTs and SsTs with oversubscription reasoning.

(1) Combined scheduling model time shared for tasks and VMs

The TsTs scheduling model, applied to tasks and virtual machines, is shown in Fig. 1. This combined scheduling model is suitable when we have sufficient computational capacity. Therefore, both tasks and multiple virtual machines can run concurrently. Neither tasks nor virtual machines are switched. It is possible to have multiple identical virtual machines running on different physical machines.

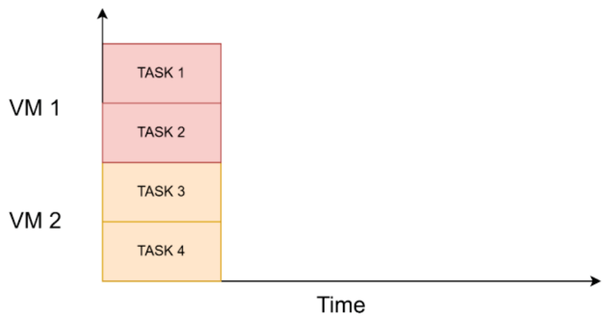

Fig. 1. Combined model time shared – time shared (TsTs)

(2) Combined scheduling model time shared for tasks and space shared for VMs

Fig. 2 shows the scheduling model of TsSs. The time shared scheduling model means that tasks run continuously without interruptions. Individual tasks can be processed simultaneously. The space shared model is used to allocate virtual machines, so virtual machines cannot run concurrently. Each virtual machine can only process its tasks after the previous machine has finished processing all tasks.

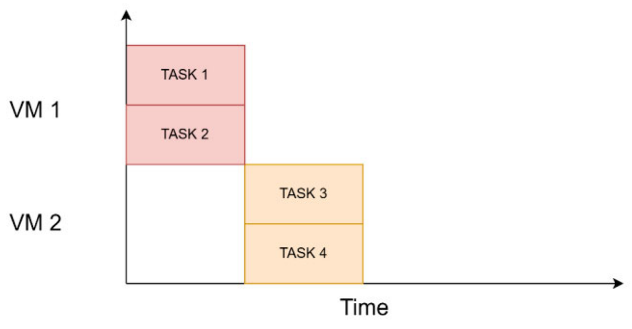

Fig. 2. Combined model time shared – space shared (TsSs)

(3) Combined scheduling model space shared for tasks and time shared for VMs

Fig. 3 shows the scheduling model of SsTs. Space shared task scheduling means that task processing is divided into multiple parts. Hence, the tasks on a VM are processed sequentially in parts and not all at the same time. VMs are allocated in a time shared manner, which makes VMs work concurrently.

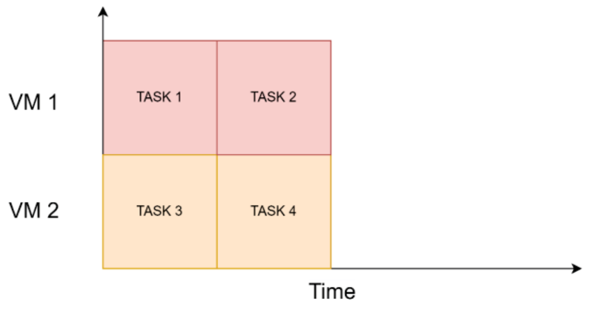

Fig.3. Combined model space shared – time shared (SsTs)

(4) Combined space shared for tasks and VMs scheduling model

In Fig. 4, we see the SsSs type scheduling model. Space shared task scheduling means that tasks are divided into multiple parts that are processed sequentially. Space shared scheduling model is also used for virtual machines, so VMs cannot run concurrently but run sequentially. For example, VM1 must process all its tasks first and then the other VMs.

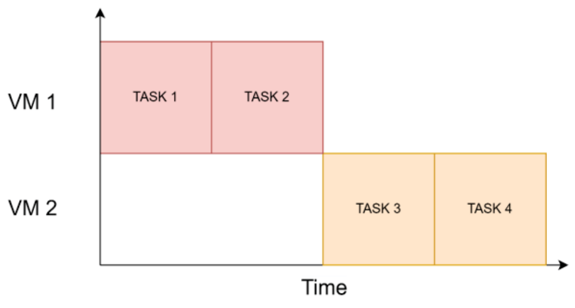

Fig. 4. Combined model space shared – space shared (SsSs)

(5) Main differences between the combined scheduling models

Based on our knowledge of the behavior of the different task scheduling models, we can conclude that the combined models starting at Ss consume significantly more overhead time to switch tasks to VMs compared to Ts. If the computing system has enough computational capacity to solve the tasks while the task computation quality requirements are met, then of course the time sharing combined model (starting at Ts) runs preferentially in the system. If the set of tasks to be processed has requirements on the computing system that cannot be met, then the space sharing combined task scheduling model (starting with Ss) must be used. The behavior of the computing system with Ss-based scheduling needs to be analyzed. Therefore, we have designed and performed experiments to decide which combined task scheduling model to choose for which type of computing system workload.

The workload is composed of datasets with complex tasks with specific requirements. The datasets are scaled, i.e., they contain increasing numbers of computational tasks.

IV. Cloud system modelling

We have created a set of computational resources and a set of computational tasks in CloudSim simulator to meet the research objectives, which are:

- Perform a thorough analysis and comparison of the combined scheduling models used in the CloudSim simulator.
- Set up experiments to compare the behavior of the system using different scheduling models. •
- Investigate how the simulation runtime in CloudSim is affected when using different combined scheduling models. We are also interested in how the simulation run changes when we use different task datasets.

A. Modelling of computational resources

Datacenters in the CloudSim simulator represent geographically separated locations. Each datacenter contains a certain number of physical machines (Hosts). Each Host contains a certain number of physical computing elements (PEs). PEs represent the computational cores of the physical machines and are defined by their performance in MIPS. Hosts have defined RAM, storage, and bandwidth.

In the simulator, we first created 2 cloud computing datacenters, see Table I.

TABLE I. Parameters of cloud datacenters in CloudSim

Host type	**Datacenter_0**	**Datacenter_1**
Num of hosts	5	10
Num of cores	50	60
RAM	70 440 MB	70 440 MB
MIPS	200 000	200 000
Bandwidth	200 GB/s	200 GB/s
Storage	1000 GB	1000 GB

We designed the datacenters to have enough power to create 100 virtual machines with multiple processors. All physical machines that are located in datacenters run on x86 system architecture with Linux operating system. Cloud computing usually runs on virtual machines. Therefore, based on the specified computing infrastructure, we created 4 types of virtual machines using a simulated Xen hypervisor. A total of 100 virtual machines with 25 machines of each type.

TABLE II Parameters of 4 types of virtual machines in CloudSim

VM type	**VM_0**	**VM_1**	**VM_2**	**VM_3**
Num of cores	7	6	5	4
RAM	512 MB	1024 MB	2048 MB	4096 MB
MIPS	1000	1200	1400	1600
Bandwidth	1000	1200	1400	1600

Our goal is to model a heterogeneous cloud system, so we designed four types of virtual machines. We specified the parameters of the virtual machines so that the cloud system contains virtual machines with different performance, see Table II.

The designed virtual machines have a minimum of 4 computational cores to be able to process each of the four types of tasks, see Table III.

B. Modelling computational tasks

We designed and created sets of computational tasks. The tasks are referred to as cloudlets in the CloudSim simulator. We then created three types of datasets that contain different types of tasks.

For each dataset type, we designed 10 different sets of computational tasks (30 scaled datasets in total). Each dataset contains a different number of tasks. The number of tasks in each dataset increases from 500 to 15000 tasks as follows: 500, 750, 1000, 2000, 3000, 4000, 5000, 7500, 10000 and 15000 tasks.

A comparison of the sum of task lengths across all 30 scaled datasets is shown in Table V.

Each dataset contains 4 types of computational tasks (see Table III.). The different types of tasks are specified:

- size (task length in millions of instructions, MI),
- the size of the program and input data files in MB,
- the size of the output result file in MB,
- the number of computational cores that the task requires for its processing.

Parallel tasks require more than 1 compute core. Task_0 requires 4 compute cores, Task_1 requires 3 compute cores, and Task_3 requires 2 compute cores. Task type Task_2 is not parallel, it requires one compute core.

TABLE III Parameters of 4 types of computational tasks

Task type	**Task_0**	**Task_1**	**Task_2**	**Task_3**
Num of cores	4	3	1	2
Length of tasks	10000 - 20000 MI	21000 - 30000 MI	31000 - 40000 MI	41000 - 50000 MI
File size	100 - 200 MB	210 - 300 MB	310 - 400 MB	410 - 500 MB
Output size	100 - 200 MB	210 - 300 MB	310 - 400 MB	410 - 500 MB

An example of a task specification in the simulator is for example as follows:

Task type 0, ID 10, Task Length 13353, Task FileSize 196, Task OutputSize = 184, Num. of CPU = 4.

Using the proposed method, we created 3 datasets of 10 task sets each, and the distribution of task types in each dataset is shown in Table IV. For example, the dataset__type_3 contains 70% of the tasks of type Task_0, 10% of the tasks of type Task_1, 10% of the tasks of type Task_2, and 10% of the tasks of type Task_3.

TABLE IV Datasets with the distribution of 4 types of tasks

	Task_0	**Task_1**	**Task_2**	**Task_3**
Dataset_type_1	25 %	25 %	25 %	25 %
Dataset_type_2	10 %	10 %	70 %	10 %
Dataset_type_3	70 %	10 %	10 %	10 %

TABLE V Dataset sizes of the tasks used in the experiments

Num. of tasks	Tasks lengths (MI)		
	Dataset_type_1	**Dataset_type_2**	**Dataset_type_3**
500	32 413 223	23 448 341	30 479 450
750	48 651 018	35 309 391	45 949 821
1000	65 152 498	47 109 564	61 338 456
2000	130 604 285	94 611 721	123 464 296
3000	196 121 716	141 931 078	185 461 201
4000	261 859 196	189 630 435	247 767 496
5000	327 361 919	236 849 196	309 728 238
7500	491 487 245	355 869 593	464 480 243
10000	655 207 027	474 469 725	620 213 700
15000	983 899 657	712 448 595	931 206 322

V. EXPERIMENTS - BEHAVIOR MODELS OF CLOUD TASK SCHEDULING ON THE BASE OF VIRTUAL MACHINES

This part of the paper consists of three experiments, labeled as 1 to 3. In each experiment, we used 10 datasets of tasks of the same type (see Table IV). The dataset sizes are listed in Table V. The different datasets represent scaled system workloads with tasks of different computational requirements. We modeled a computing environment that contains 2 datacenters, specified in Table I and Table II, in the CloudSim simulator.

The objective of the experiments is Comparative Analysis of Combined Models Cloud Task Scheduling on the basis of virtual machines. Therefore, we sequentially made runs of the simulations by using 6 different combined scheduling models for each run: TsTs, TsSs, TsTo, SsTs, SsTo, SsSs. The behavior of a differently loaded system with different combined scheduling models is the subject of our investigation. We are concerned with the analysis of the system using combined models that start with Ts versus Ss for scheduling tasks to VMs. In all simulations, the First Come First Served (FCFS) scheduling algorithm is used for allocating tasks to VMs, thus guaranteeing scheduling consistency between experiments.

For each experiment, we ran each simulation with 10 datasets 50 times, each time with a different randomly shuffled task dataset. This caused the tasks to always be scheduled on different VMs. We ran the simulations for 10 different scaled datasets by 50 iterations with different VMs. Given that we repeated all simulations for 6 types of combined scheduling models, we ran a total of 3000 simulations.

A. Experiment 1 with dataset_type_1

In Experiment 1, we used dataset_type_1, which contains 25% of each type of considered tasks, whose parameters are depicted in Table III.

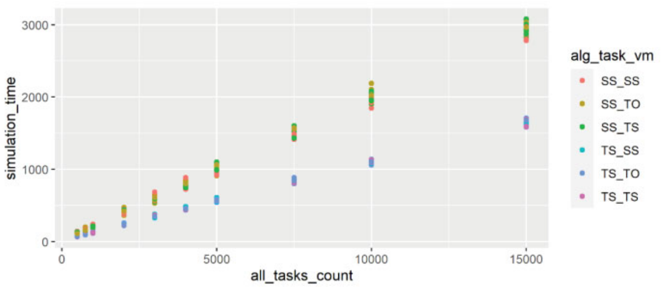

Fig. 5. Comparison of task processing time for datataset_type_1

In this experiment, we compare the simulation times involved in scheduling tasks on virtual machines. This is the group of combined models that start at Ts compared with the group of combined models Ss.

In Fig 5 we can see that when using the task scheduling model on the Ts virtual machines, the simulation times are significantly lower than when using the Ss scheduling model as we expected. The larger the number of tasks the more appropriate it is to use task scheduling on Ts virtual machines. To model the behavior of this system with a workload of 15,000 tasks, we measured simulation times ranging from 2748.53 sec. for the Ss combined models. to 3102.45 sec. and for Ts combined model's times in the interval from 1584.74 sec. to 1692.82 sec.

B. Experiment 2 with dataset_type_2

In Experiment 2, we used dataset_type_2, which contains 70% of the tasks that require 1 computational core for their processing. The goal of this experiment is to investigate the processing time values of the tasks and compare them with experiment 1.

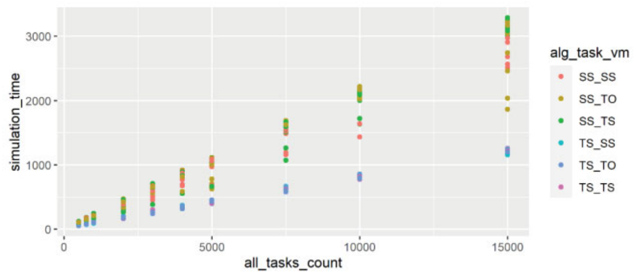

Fig. 6. Comparison of taks processing time for dataset_type_2

Fig 6 shows a comparison of simulation times for six different scheduling models. We can see that the task scheduling model on VMs of type Ss causes the simulation times to be significantly higher than when using the Ts scheduling model. It can be seen that when using the tasks from dataset_type_2 and the task scheduling model on VMs of type Ss, the measured values of the times are significantly more dispersed than for the tasks in dataset_type_1. The minimum values of the simulation times for Ss are almost close to the values of the times for Ts. See Table VI.

TABLE VI COMPARISON OF SIMULATION TIMES FOR SS TASK SCHEDULING DATASET TYPE 2, WITH THE NUMBERS OF TASKS 7500, 10000, 15000

Num of tasks	**Min. simulation time (s)**	**Max. simulation time (s)**	**Difference max. – min. (s)**	**Time increase in %**
7500	1100.45	1750.84	650.39	58.8
10000	1400.55	2300.25	899.7	64.2
15000	1800.10	3288.12	1488.02	82.6

In the analysis, we focused on examining the variance of the values. There was a significant difference between the simulation times for the number of tasks of 7,500, 10,000 and 15,000. For the number of 7,500 tasks, the difference in values between the minimum and maximum simulation time was 650.39 s (58.8%), for 10,000 tasks it was 899.7 s (64.2%) and for 15,000 tasks it was 1488.02 s (82.6%). Hence, if a combined scheduling model starting at Ss (Ss scheduling tasks on VMs) is to be used, then the completion time of the last task should be minimized using an optimization algorithm.

C. Experiment 3 with dataset_type_3

In Experiment 3, we used dataset_type_3, which contains 70% of the tasks that require 4 computational cores for their processing. The parameters of all task types are depicted in Table III.

The purpose of this experiment is to investigate the values of task processing times (simulation times) and compare them with Experiment 1.1 and Experiment 1.2. The comparison of simulation times for six different scheduling models is shown in Fig 7. Again, we are interested in the behavior of the system using the Ss model.

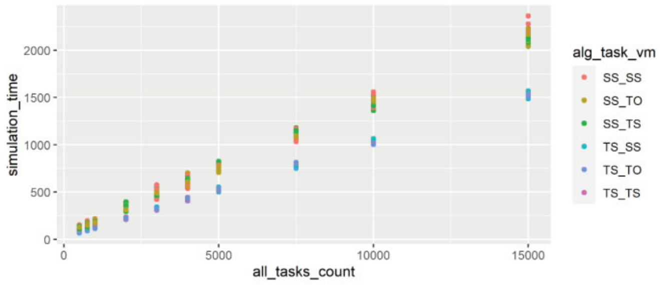

Fig. 7. Comparison of task processing time for dataset_type_3

In Fig. 7. we can see that using dataset_type_3 and a scheduling model that starts at Ss, for a dataset with a task count of 15000, we observed simulation times ranging from 2048.98 sec. to 2453.84 sec. The maximum values of simulation times in Experiments 1.1 and 1.2 were 3102.45 sec. and 3288.12 sec. In experiment 1.3, the maximum simulation time is 2453.84 sec. and it is the lowest time among all the experiments. Also, the variance of the measured time values is not as large as that of experiment 1.2. For the combined scheduling model starting at Ts, we measured simulation times ranging from 1480.74 sec to 1625.75 sec. These values are slightly higher than those of experiment 1.2. We can conclude that for a system workload with tasks, 70% of which require 4 compute cores, it is appropriate to use the Ss model of scheduling the tasks on VMs.

VI. Behavioral Analysis of Combined Models of Cloud Task Scheduling

In this section, we performed the analysis of the combined scheduling models using all the prepared datasets. We used the results of previous experiments for the analysis. We focused on comparing the minimum simulation times because the minimum simulation times represent the optimized task schedules. The goal of the given analysis was to investigate how the simulation runtime in CloudSim is affected when using different combined scheduling models, and how the simulation runtime changes if we use different task datasets.

A. Analysis of combined Ts and Ss task scheduling models

We analyze the minimum simulation times for the combined TsTs and SsSs scheduling models and investigate the system behavior using all 30 datasets. In total, there are 60 values. See Fig 8.

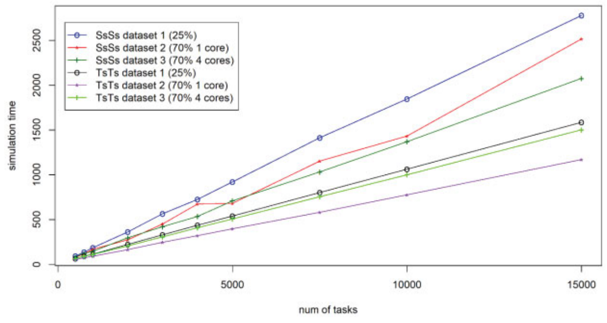

Fig. 8. Comparison of simulation times for different numbers of tasks from three different datasets

In Fig. 8, we can see that using each of the 30 datasets, the minimum simulation time is always higher for the SsSs combination than for the TsTs combination. For the combined SsSs model, it is most appropriate to use a workload where 70% of the tasks have requirements for 4 computational cores for their processing defined in Type 3 datasets. The simulation time is higher when using Type 2 datasets and highest when using Type 1 datasets.

TABLE VII Comparison of min task processing times of all three datasets using the combined TsTs and SsSs models

	Dataset_type_1		Dataset_type_2		Dataset_type_3	
Number of tasks	500	15000	500	15000	500	15000
Min time SsSs (s)	93.83	2778.91	79.62	2516.66	84.39	2075.98
Min time TsTs (s)	66.20	1585.51	56.04	1171.17	67.15	1502.20
Enhancement	30 %	43 %	30 %	54 %	21 %	28 %

In Table VII, we can know that using the combined SsSs model for dataset 1 with the number of tasks 500, we measured a minimum time of 93.83 sec. For the same dataset, using the combined TsTs algorithm, we measured a minimum time of 66.20 sec, which is an improvement of less than 30%. When using the combined SsSs model for dataset 1 with 15000 tasks, we measured a minimum time of 2778.91 sec. For the same dataset, using the combined TsTs algorithm, we measured a minimum time of 1585.51 sec, an improvement of almost 43%. Using the combined SsSs model for dataset 2 with a task count of 500, we measured a minimum time of 79.62 sec. For the same dataset, using the combined TsTs algorithm, we measured a minimum time of 56.04 sec, an improvement of less than 30%. However, when using the combined SsSs model for dataset 2 with a task count of 15000, we measured a minimum time of 2516.66 sec. For the same dataset, using the combined TsTs algorithm, we measured a minimum time of 1171.17 sec, an improvement of almost 54%.

Using the combined SsSs model for dataset 3 with a task count of 500, we measured a minimum time of 84.39 sec. For the same dataset, using the combined TsTs algorithm, we measured a minimum time of 67.15 sec, an improvement of less than 21%. However, when using the combined SsSs model for dataset 3 with a task count of 15000, we measured a minimum time of 2075.98 sec. For the same dataset, using the combined TsTs algorithm, we measured a minimum time of 1502.20 sec, an improvement of almost 28%.

Based on the experiments, we can conclude that the use of the combined TsTs model is preferable to the SsSs model (it has a shorter completion time for the last task) for all types of workloads. The reduction in the last task completion time depends on the task composition of the dataset and also the number of tasks. In our case, we observed the smallest reduction in minimum simulation time for dataset 3, namely 21% for 500 tasks and 28% for 15000 tasks. We measured a larger reduction in minimum simulation time for dataset 1, namely 30% for 500 tasks and 43% for 15000 tasks. The largest improvement in minimum simulation time was observed for dataset 2, namely 30% for 500 tasks and 54% for 15000 tasks. This suggests that the combined scheduling model of TsTs is best suited to dataset 2, which contains 70% of the tasks that require only 1 core for their processing.

B. Analysis of combined scheduling models using tasks in type 1 datasets

We analyze the minimum simulation times using 10 different scaled type-1 datasets and all 6 combined task scheduling models. We analyze 60 minimum simulation times.

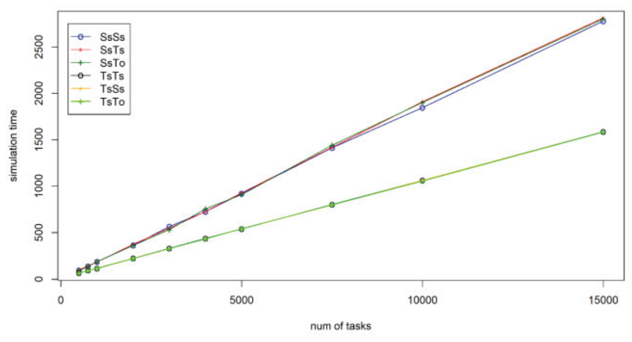

Fig. 9. Comparison of simulation times for different numbers of tasks from dataset_type_1

We can see that two triads of tenses have arisen which are almost identical and overlap. The first triplet consists of the minimum simulation times that have been generated using the three combined Ss-based scheduling models. The second triplet is formed by the minimum simulation times that were generated using the three combined Ts based scheduling models. In we go that the minimum simulation times for the combined scheduling models, where the combined scheduling model Ts is more appropriate for all the workloads represented by our datasets. Interestingly, it does not matter the detail of the combined model, just choose a model starting with Ts or Ss as appropriate.

TABLE VIII COMPARISON OF MIN TASK PROCESSING TIMES FOR DATASET_TYPE_1 USING THE COMBINED TSSS AND SSTS MODELS

Number of tasks	**500**	**2000**	**5000**	**7500**	**10000**	**15000**
Min time SsTs (s)	99.32	374.53	928.98	1418.56	1909.09	2813.60
Min time TsSs (s)	66.02	224.47	543.77	798.90	1063.31	1583.35
Enhancement	34 %	40 %	42 %	44 %	44 %	44 %

In Table VIII The values of the minimum run times of the tasks that when using the combined models SsTs and TsSs for dataset 1 are shown.

Using the combined SsTs model for dataset 1 with a task count of 10000, we measured a minimum time of 1909.09 sec. For the same dataset, using the combined TsSs algorithm, we measured a minimum time of 1063.31 sec, an improvement of less than 44%. When using the combined SsTs model for dataset 1 with 15000 tasks, we measured a minimum time of 2813.60 sec. For the same dataset, using the combined TsSs algorithm, we measured a minimum time of 1583.35 sec, an improvement of almost 44%.

C. Analysis of combined scheduling models using tasks in type 2 datasets

We analyze the minimum simulation times using 10 different scaled type-2 datasets (70% of the tasks that require 1 computational core for their processing) and all 6 combined task scheduling models. We analyze 60 minimum simulation times. Fig 10. shows a comparison of the minimum simulation times for the six different combined scheduling models.

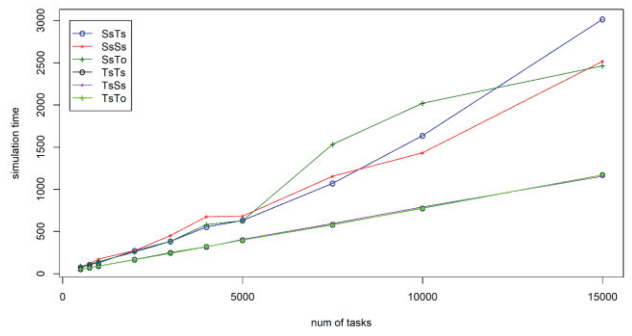

Fig. 10. Comparison of simulation times for different numbers of tasks from dataset_type_2

All the simulation run times of the tasks that we schedule based on Ts are very similar and overlap. However, the simulation run times of tasks where tasks are allocated using Ss only overlap when the number of tasks range from 500 to 5000. Where the number of tasks to be processed was greater than or equal to 7500, so are the values of the lowest simulation times for different combinations, see Fig. 10.

TABLE IX COMPARISON OF MIN TASK PROCESSING TIMES FOR DATASET_TYPE_2 USING THE COMBINED TSSS AND SSTS MODELS

Number of tasks	**500**	**2000**	**5000**	**7500**	**10000**	**15000**
Min time SsTs (s)	75.44	271.67	630.60	1071.33	1635.63	3016.41
Min time TsSs (s)	54.32	167.97	407.71	596.33	790.67	1155.37
Enhancement	28 %	38 %	35 %	44 %	52 %	62 %

In Table IX. you can know that using the combined SsTs model for dataset 2 with a number of tasks of 500, we measured a minimum time of 75.44 sec. For the same dataset, when using the combined TsSs algorithm, we measured a minimum time of 54.32 sec, an improvement of less than 28%. Using the combined SsTs model for dataset 2 with a task count of 2000, we measured a minimum time of 271.67 sec. For the same dataset, using the combined TsSs algorithm, we measured a minimum time of 167.97 sec, an improvement of almost 38%.

Using the combined SsTs model for dataset 2 with a task count of 5000, we measured a minimum time of 630.60 sec. For the same dataset, using the combined TsSs algorithm, we measured a minimum time of 407.71 sec, an improvement of less than 35%. Using the combined SsTs model for dataset 2 with a task count of 7500, we measured a minimum time of 1071.33 sec. For the same dataset, using the combined TsSs algorithm, we measured a minimum time of 596.33 sec, an improvement of almost 44%.

Using the combined SsTs model for dataset 2 with a task count of 10000, we measured a minimum time of 1635.63 sec. For the same dataset, using the combined TsSs algorithm, we measured a minimum time of 790.67 sec, an improvement of less than 52%. When using the combined SsTs model for dataset 2 with a task count of 15000, we measured a minimum time of 3016.41 sec. For the same dataset, using the combined TsSs algorithm, we measured a minimum time of 1155.37 sec, an improvement of almost 62%.

D. Analysis of combined scheduling models using tasks in type 3 datasets

We analyze the minimum simulation times using 10 different scaled datasets of type 3 and all 6 combined task scheduling models. We analyze 60 minimum simulation times. Fig 11. shows a comparison of the minimum simulation times for the six different combined scheduling models.

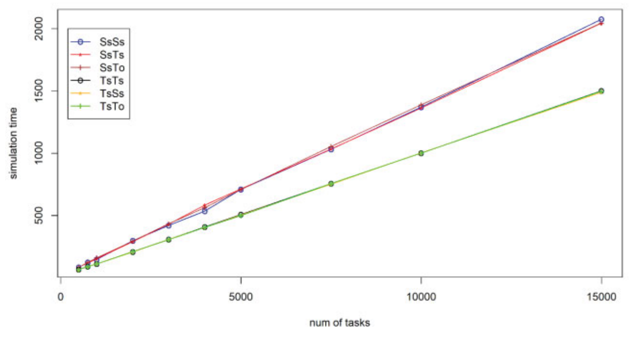

Fig. 11. Comparison of simulation times for different numbers of tasks from dataset_type_3

In Fig. 11 we can see that we again only get two resulting trajectories of simulation times, such that three of the six combined scheduling models overlap each time. The first waveform of the task run times is when using the Ss based combined models. The second is the waveform of task run times when using the Ts-based combined models.

Thus, as we expected we see that the simulation times for the combined scheduling models where the scheduling model Ts is used for the tasks are always lower than the simulation times for the combined scheduling models in which the tasks are allocated through the scheduling model Ss.

We can see that the smallest difference between the simulation times when combined scheduling models are used, where tasks are allocated via a time shared or space shared scheduling model, is when dataset 3 is used, which contains 70% of the tasks that require 4 cores for their processing.

TABLE X COMPARISON OF MIN TASK PROCESSING TIMES FOR DATASET_TYPE_3 USING THE COMBINED TSSS AND SSTS MODELS

Number of tasks	**500**	**2000**	**5000**	**7500**	**10000**	**15000**
Min time SsTs (s)	87.02	292.53	713.61	1034.36	1363.19	2044.31
Min time TsSs (s)	66.71	207.99	505.92	751.77	1003.09	1488.89
Enhancement	23 %	28 %	29 %	27 %	26 %	27 %

In Table X. The values of the minimum run times of the tasks that when using the combined models SsTs and TsSs for dataset 3 are shown.

Using the combined SsTs model for dataset 3 with a task count of 10000, we measured a minimum time of 1363.19 sec. For the same dataset, using the combined TsSs algorithm, we measured a minimum time of 1003.09 sec, an improvement of less than 26%. When using the combined SsTs model for dataset 3 with a task count of 15000, we measured a minimum time of 2044.31 sec. For the same dataset, using the combined TsSs algorithm, we measured a minimum time of 1488.89 sec, an improvement of almost 27%.

E. Evaluation of the analysis of combined scheduling models

We used the results of previous experiments for the analysis. We focused on minimum simulation times using all prepared datasets. The minimum simulation times represent the optimized task schedules. The optimized task schedules showed interesting analysis results. Comparing the combinations of TsTs and SsSs used. For the optimized combination model of SsSs, it is most appropriate to use tasks in datasets of type 3 and for the optimized combination model of TsTs, it is most appropriate to use tasks in datasets of type 2. Interestingly, for the optimized models starting with Ts or Ss, they behave the same for datasets of type 1 and 3. We can conclude that as long as the sets of computational tasks have all requirements satisfied, it is necessary to use scheduling models starting at Ts.

VII. CONCLUSION

A quality cloud system must respect the requirements of the tasks. It is essential that the system provides the tasks with sufficient computing resource capacity, especially CPU computing time. The requirements of many tasks collide in the system and need to be optimally scheduled. Quality scheduling of tasks will be reflected in maximum throughput, minimum waiting time for individual tasks, minimum completion time for the last task (e.g. in batch processing) and adherence to task requirements. Ensuring the above quality criteria will ultimately result in minimizing the power consumption of the computing system.

Understanding the behavior of a computing system is a prerequisite for achieving high system performance. Therefore, in this paper, we have studied in detail the behavior of the computing system, in a simulated CloudSim environment. We created a simulated cloud system and 30 scaled task load sets that represented different workloads of the system. The number of tasks in these sets grew from 500 to 15,000 tasks. The tasks had requirements that specified the number of computational cores.

CloudSim supports 6 different combinations of scheduling models, which can be classified into 2 basic groups of time sharing and space sharing, depending on how compute tasks are allocated to virtual machines.

Based on our knowledge of the behavior of the different task scheduling models, we can conclude that the combined models starting at Ss consume significantly more overhead time to switch tasks to VMs compared to Ts. If the computing system has enough computational capacity to solve the tasks while the task computation quality requirements are met, then of course the time sharing combined model (starting at Ts) runs preferentially in the system. If the set of tasks to be processed has requirements on the computing system that cannot be met, then the space sharing combined task scheduling model (starting with Ss) must be used. The behavior of the computing system with Ss-based tasks scheduling needs to be analyzed in more details.

Therefore, we designed and performed experiments to decide which combined task scheduling model to choose for which type of computing system workload. In total, we did 9000 simulations.

We analyzed the results of each experiment and then compared the behavior of the system using each scheduling combination.

We then analyzed the behavior of the system in achieving the best task completion times. The minimum time represents the best schedule.

As we expected, the Ts combined task scheduling model on VMs has a shorter completion time than Ss for the last task for all types of workloads. If we consider a dataset with 15000 tasks, then using the Ts model will reduce the completion time of the last task by 43% when using a dataset of type 1 tasks, by 54% when using a dataset of type 2 tasks, and by 28% when using a dataset of type 3 tasks, compared to the Ss model.

If the task requests are so taxing on the system that it is not feasible to schedule tasks with the Ts model, then it is most appropriate to use the combined SsTs model and compose the task dataset with requests from 70% of the tasks with 4 cores and 10% each of the tasks with 1, 2, and 3 cores.

Based on the experiments, we can conclude that the use of the combined TsTs model is preferable to the SsSs model (it has a shorter completion time for the last task) for all types of workloads. The reduction in the last task completion time depends on the task composition of the dataset and the number of tasks. In our case, we observed the smallest reduction in minimum simulation time for dataset 3, namely 21% for 500 tasks and 28% for 15000 tasks. We measured a larger reduction in minimum simulation time for dataset 1, namely 30% for 500 tasks and 43% for 15000 tasks. The largest improvement in minimum simulation time was observed for dataset 2, namely 30% for 500 tasks and 54% for 15000 tasks. This suggests that the combined scheduling model of TsTs is best suited to dataset 2, which contains 70% of the tasks that require only 1 core for their processing. Which inspires us to extend the automatic scheduling model based on big data analysis [16], [17], [18].

We used the results of previous experiments for the analysis. We focused on minimum simulation times using all prepared datasets. The minimum simulation times represent the optimized task schedules. The optimized task schedules showed interesting analysis results. Comparing the combinations of TsTs and SsSs used. For the optimized combination model of SsSs, it is most appropriate to use tasks in datasets of type 3 and for the optimized combination model of TsTs, it is most appropriate to use tasks in datasets of type 2. Interestingly, for the optimized models starting with Ts or Ss, they behave the same for datasets of type 1 and 3. We can conclude that as long as the sets of computational tasks have all requirements satisfied, it is necessary to use scheduling models starting at Ts.

ACKNOWLEDGMENT

Computing was performed in the High-Performance Computing Center of the Matej Bel University in Banska Bystrica using the HPC infrastructure acquired in project ITMS26230120002 and 26210120002 (Slovak infrastructure for high-performance computing) supported by the Research & Development Operational Programme funded by the ERDF.

It was supported by the Erasmus+ project: Project number: 2022-1-SK01-KA220-HED-000089149, Project title: Including EVERyone in GREEN Data Analysis (EVERGREEN) funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the Slovak Academic Association for International Cooperation (SAAIC). Neither the European Union nor SAAIC can be held responsible for them.

Co-funded by the European Union

EverGreen DATA Analytics

REFERENCES

[1] Buyya, R., Ranjan, R., & Calheiros, R. (2009). Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities. IEEE Access, 10, 34996-35011. ISBN 978-1-4244-4906-4.

[2] Calheiros, R. N., Ranjan, R., Rose, C. A. F. D., & Buyya, R. (2009). CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services.

[3] Sahkhar, L., & Yadav, S. S. (2022). Efficient Cloudlet Allocation to Virtual Machine to Impact Cloud System Performance. International Journal of Information System Modeling and Design. International Journal of Information System Modeling and Design (IJISMD) 13(6). DOI: 10.4018/IJISMD.297630

[4] Behera, I., & Sobhanayak, S. (2024). Task scheduling optimization in heterogeneous cloud computing environments: A hybrid GA-GWO approach. Journal of Parallel and Distributed Computing. Volume 183. DOI: 10.1016/j.jpdc.2023.104766.

[5] Singh, H., Tyagi, S., Kumar, P., Gill, S. S., & Buyya, R. (2021). Metaheuristics for scheduling of heterogeneous tasks in cloud computing environments: Analysis, performance evaluation, and future directions. Simulation Modelling Practice and Theory. DOI: 10.1016/j.simpat.2021.102353

[6] Vijay, R., & Sree, T. R. (2023). Resource Scheduling and Load Balancing Algorithms in Cloud Computing. SN Computer Science. DOI: 10.1007/s42979-022-01609-9

[7] Liu, S., Ma, X., Jia, Y., & Liu, Y. (2022). An Energy-Saving Task Scheduling Model via Greedy Strategy under Cloud Environment. Hindawi Wireless Communications and Mobile Computing. DOI: 10.1155/2022/8769674

[8] Kumar, M., & Suman. (2022). Hybrid Cuckoo Search Algorithm for Scheduling in Cloud Computing. Computers, Materials & Continua. DOI: 10.32604/cmc.2022.021793

[9] Škrinárová, J., & Dudáš, A. (2022). Optimization of the functional decomposition of parallel and distributed computations in graph coloring with the use of high-performance computing. IEEE Access, 10, 34996-35011. ISSN 2169-3536.

[10] Hussain, A., Aleem, M., Iqbal, M. A., & Islam, M. A. (2019). Investigation of cloud scheduling algorithms for resource utilization using cloudsim. Comput Inform. DOI: 10.31577/cai_2019_3_525

[11] Škrinárová, J. (2022). Heterogeneous Cloud Systems and Criteria for Enhanced Performance. 2022 IEEE 16th International Scientific Conference on Informatics. 10.1109/Informatics57926.2022.10083443

[12] Chawla, Y., & Bhonsle, M. (2012). A study on scheduling methods in cloud computing. Int. J. Emerg. Trends Technol. Comput. Sci., 1(3), 12-17.

[13] Aladwani, T. (2019). Scheduling IoT Healthcare Tasks in Fog Computing Based on their Importance. International Learning & Technology Conference. DOI: 10.1016/j.procs.2019.12.138

[14] Ma, T., Chu, Y., Zhao, L., & Ankhbayar, O. (2014). Resource Allocation and Scheduling in Cloud Computing: Policy and Algorithm. IETE Tech. Rev., 31(1), 4-16. DOI: 10.1080/02564602.2014.890837

[15] Himthani, P., & Dubey, G. P. (2019). Performance Analysis of Space Shared Scheduling and Time Shared Scheduling in Cloud Sim. IJRDET. DOI: 10.5120/17092-7629

[16] Purkrabková, Z., et al. (2021). Traffic Accident Risk Classification Using Neural Networks. Neural Network World, 31(5). DOI:10.14311/NNW.2021.31.019

[17] Horaisova, K., et al. (2018). Discrimination between Alzheimer's disease and amyotrophic lateral sclerosis via affine invariant spherical harmonics analysis of spect images. Neural Network World, 28(1). DOI:10.14311/NNW.2018.28.002

[18] Čerešňák, R., et al. (2021). Mapping rules for schema transformation: SQL to NoSQL and back. In 2021 International Conference on Information and Digital Technologies (IDT) (pp. 52-58). Zilina, Slovakia. doi: 10.1109/IDT52577.2021.9497629.