International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Effective Scheduling Algorithm for Resource Allocation in MANET with Cloud Computing

S. Saranya [a], K. S. Rajeswari [b*]

[a, b] *Assistant Professor, Department of Computer Applications, Idhaya College for Women, Sarugani.*

A B S T R A C T

Workflows are often used to model large-scale scientific problems in fields such as bioinformatics, astronomy, and physics. Such scientific workflows are increasingly data and computationally demanding and thus require high-performance computing environments to run in a reasonable amount of time. These workflows are typically modeled as a set of tasks interrelated by data or computational dependencies. The orchestration of these tasks in distributed resources has been extensively studied over the years, with an emphasis on environments such as grids and clusters. However, with the emergence of new paradigms such as cloud computing, new approaches need to be developed to address the specific challenges and opportunities of these technologies. Cloud computing, the latest distributed computing paradigm, offers enormous opportunities to solve large-scale scientific problems. Furthermore, existing jobs do not meet user quality of service (QoS) requirements or incorporate some fundamental principles of cloud computing, such as elasticity and heterogeneity of computing resources. This paper presents a resource provisioning and scheduling strategy for scientific workflows in Infrastructure as a Service (IaaS) clouds. We propose an algorithm based on a metaheuristic optimization technique, Particle Swarm Optimization (PSO), whose goal is to minimize the overall cost of running a workflow and satisfy deadline constraints. Our heuristics are evaluated using CloudSim and several well-known scientific workflows at different scales. The results show that our method outperforms the current state-of-the-art algorithms.

Keywords: Sheduling, Resource Allocation, MANET, Cloud Computing , Performance

1. Introduction

Cloud computing is to deliver computing as a service rather than a product, thereby providing shared resources, software and information to users through the network. Cloud computing providers deliver applications over the Internet, accessible from a web browser, while business software and data are stored on servers in remote locations.

Cloud computing is an emerging trend in distributed computing that facilitates software application platforms and hardware infrastructure as a service. Cloud service providers deliver these services under customized service level agreements (SLAs) that define the quality of service (QoS) parameters required by users. Cloud computing reduces investment in various resources, such as hardware and software resources, which can be rented and released. Reduce initial investment, maintenance costs and operating costs. Cloud services are hosted on the service provider's own infrastructure or a third-party cloud infrastructure provider. Three main types of services are offered; Platform as a Service (PaaS), Infrastructure as a Service (IaaS), and Software as a Service (SaaS). Cloud users who use this service, as long as they need it, can pay according to their needs through a pay-per-use model. The cloud provides the ability to adjust resource capacity in response to changing application needs, often referred to as autoscaling. However, giving users more control also requires the development of new methods of task scheduling and resource provisioning. Resource management decisions required in cloud scenarios must not only consider performance-related metrics such as workflow spam or resource utilization, but also must consider budgetary constraints, since resources from commercial clouds often have monetary costs associated with them. Information on resource management challenges when running the scientific workflow suite in the cloud. We address an important new problem of maximizing the number of complete workflows in a set subject to budget and deadline constraints.

In addition to many applications, cloud computing environments can also be used for the execution of workflows. Running a workflow involves scheduling the workflow. Workflow scheduling involves mapping workflow tasks to available resources in such a way that certain predefined criteria are met. Workflow scheduling is a well-known NP-complete problem [4], and it is a key problem in workflow management systems. Moving workflows to cloud computing enables us to take advantage of the cloud to execute workflows. Programming can also be multi-objective. Multi-objective programming is more difficult to solve. Different researchers have proposed many heuristics and metaheuristics for workflow programming.

Currently, workflow scheduling algorithms for cloud systems mainly focus on two parameters, viz. cost and time. Since the cloud adopts the pay-as-you-go model, all services are charged. The cost mainly depends on the QoS (Quality of Service) provided. Service providers charge more for higher

* Corresponding author. Tel.: +0-000-000-0000 ; fax: +0-000-000-0000.
E-mail address: author@institute.xxx

QoS and less for lower QoS. From a cloud user perspective, early and reliable job execution is another important factor, but it incurs additional costs. Users may require their workflow tasks to be completed quickly and reliably at a manageable cost, among other QoS requirements. These types of requirements make cloud workflow scheduling even more important and complex

2. Literature Review

Brintha, N. C., JT Winowlin Jappes, and M. S. Vignesh. In this paper, we consider deadlines as the primary constraint and propose a deadline-based constrained workflow scheduling algorithm that executes work within a manageable cost while satisfying the deadline constraints defined by deadlines flow. The algorithm uses a scoring concept that represents the capabilities of hardware resources. Use this fractional value when allocating resources to various tasks in the workflow application. The algorithm allocates those resources to workflow applications that are reliable, reduce operating costs, and complete workflow applications within a user-specified time frame. [1]

Chandran, Ramesh, S. Rakesh Kumar, and N. Gayathri. developed the concept of "dynamic dataflow", which uses alternating tasks as an additional control over dataflow cost and QoS. Furthermore, we formalize an optimization problem to represent the deployment and configuration of runtime resources, allowing us to balance QoS, value, and application resource cost. We propose two centralized and shared greedy heuristics based on variable-size packing algorithms and compare them with a genetic algorithm (GA)-based heuristic that gives near-optimal solutions to Program. Large-scale simulation studies using linear road benchmarks and AWS public cloud VM performance tracking show that while GA-based heuristics provide better scheduling quality, greedy heuristics are more practical and you can use cloud elasticity intelligently to mitigate the impact . Variability, including input data rate and throughput of cloud resources to meet QoS for fast data applications. [two]

Haoxiang, Wang, and S. Smys. Cloud Workflow System is a platform service that facilitates the automation of distributed applications based on new cloud infrastructures. So far, many scheduling strategies have been proposed aiming at maximizing the amount of work done while satisfying QoS constraints such as deadlines and budgets. However, many of them are not optimal for incorporating some fundamental principles of cloud computing, such as elasticity and heterogeneity of computing resources. Therefore, our work focuses on studying various problems related to workflow scheduling. We study various existing workflow scheduling algorithms in cloud computing and list their various parameters as well as the tools and algorithms used. Through a large number of references, it is found that there are many kinds of scheduling algorithms, and these algorithms are different in scheduling factors and parameters. It is also analyzed that workflow scheduling is a difficult NP problem, and it is impossible to generate optimal solutions in polynomial time, and the algorithm focuses on generating approximate or near optimal solutions. A lot of research has been done on workflow scheduling, but there are still some areas that need further attention, for example, the elastic and heterogeneous nature of cloud is not considered to meet the QoS requirements. [3]

Sengan, Sudhakar, V. Priya, and Pankaj Dadheech. Generalized Assignment Problem (GAP), a 0–1 integer programming problem (IP) of assigning a set of n items to a set of m knapsacks, where each item must be assigned to exactly one knapsack and there are constraints about the items Allocated resource availability has recently been further generalized to include the case where a pair of adjacent knapsacks can share items. This problem is known as the generalized assignment of special ordered sets of type 2 (GAPS2). For fairly large values of m and n, the NP-hard GAPS2 combinatorial problem becomes intractable for standard IP software, so heuristic algorithms need to be developed to solve such problems.

Chakraborty, Chinmay, and Joel JCP Rodrigues vary the running time as a probabilistic random variable and solve the heterogeneous assignment problem with probability (HAP). The solution to the HAP problem assigns an appropriate FU type to each task such that the total cost is minimized while satisfying time constraints and guaranteeing confidence probabilities. A probabilistic approach to high-level synthesis of embedded system-specific architectures in real-time using heterogeneous functional units with probabilistic execution times. For the probabilistic heterogeneous assignment problem (HAP), path assignment and tree assignment algorithms are proposed, which give the optimal solution when the input graph is a simple path and tree, respectively. Two other algorithms, an optimal algorithm and a near-optimal heuristic algorithm, are proposed to solve general problems. [5].

3. Problem Description

In recent years, ad-hoc parallel data processing has become one of the best applications for infrastructure-as-a-service (IaaS) clouds. Major cloud computing companies have begun to integrate parallel data processing frameworks into their product portfolios, making it easier for customers to access these services and implement their programs. However, currently used processing frameworks are designed for homogeneous, static cluster configurations and do not take into account the special nature of the cloud. As a result, the allocated computing resources may not be sufficient to process most of the submitted work, unnecessarily increasing processing time and cost.

To reduce the impact of public cloud resource performance changes on workflow timelines, we propose a new algorithm called EIPR, which considers the behavior of cloud resources during scheduling and applies task replication to increase compliance. Chance. Application deadline.

3.1 EIPR Algorithm

The existing algorithm performs following steps

- Combined Provisioning and Scheduling

- Data-Transfer Aware Provisioning Adjust
- Task Replication

The first step in the combined provisioning and scheduling EIPR algorithm involves determining the number and type of virtual machines used to execute the workflow, as well as the start and end times for each virtual machine (configuration) and determining the sequence and task location. on the allocated resources (scheduling). Provisioning and scheduling issues are closely related, because VM availability affects scheduling, and scheduling affects virtual VM completion time. Therefore, more efficient scheduling and provisioning can be achieved if these two problems are addressed as one rather than separately.

3.2. Combined Provisioning and Scheduling

The first step of the EIPR algorithm consists of determining the number and type of virtual machines that will be used to execute the workflow, as well as the start and end time and order and sequence of each virtual machine (configuration). The task's position among those assigned resources. (planning). Provisioning and scheduling issues are closely related, because VM availability affects scheduling, and scheduling affects virtual VM completion time. Therefore, more efficient scheduling and provisioning can be achieved if these two problems are addressed as one rather than separately.

3.3. Data-Transfer Aware Provisioning Adjust

In the second step of the EIPR algorithm, the supply decisions made in step 1 are adjusted to account for the above factors. For each non-input task scheduled to be the first task of the virtual machine, and each non-output task scheduled to be the last task of the virtual machine, the algorithm starts by setting the time to the first day of the Start machine before. The first task and/or set the end time of the machine later than the end time of the last task, depending on where the extra time is needed. Finally, the start of the first allocation slot for each virtual machine is predicted by the estimated time to deploy and start the virtual machine, which is taken into account during the scheduling process.

3.4. Demerits of Existing Work

- Policy issues remain regarding how to adaptively allocate to meet the resource needs of a virtual machine while minimizing the amount of PM used.
- The resource requirements of a virtual machine are heterogeneous, which can be challenging because the virtual machine runs a different set of applications and changes as the workload grows and shrinks. The two main disadvantages are avoiding overload and green computing.
- Expensive
- Complex
- Increases data base organization
- The processing framework is then responsible for distributing the program among the available nodes and executing each instance of the program on the appropriate data block. In particular, Nephele is the first data processing framework capable of dynamically allocating/deallocating different cloud computing resources according to your schedule and job execution.

4. Proposed Work

4.1. Proposed work

The proposed priority algorithm helps cloud administrators to decide the priority among users and allocate resources efficiently according to the priority. This resource allocation technique is more efficient than network and utility computing because in these systems there is no prioritization between user requests and the cloud administrator decides randomly and prioritizes the user sending the request. First come first served basis. But with the advent of cloud computing, by using this implemented priority algorithm, cloud administrators can easily make decisions based on the different parameters discussed above to decide the priority between different user requests, so that administrators can Allocate available resources efficiently and profitably. and user satisfaction.

After receiving the user's request, the cloud checks the allocation of resources to the user according to the user's request. User demand batches will be created based on the task type, the number of processors the user requires, and the user execution time.

If the resource is not available, the user must wait for the resource to become available. The user's waiting request will be compared with all waiting resources and assigned the corresponding priority. Performance values are calculated based on processor and RAM usage. If two requests with equal requirements have the same priority, resources will be allocated according to FCFS (First Come First Served).

4.2. Advantages of proposed system

- The advantages of proposed systems are as follows:
- Dynamic resource allocation
- Parallelism is implemented
- Designed to run data analysis jobs on a large amount of data
- Many Task Computing (MTC) has been developed
- Less expensive
- More Effective
- More Faster
- In this paper, the design and implementation of an automated resource management system that strikes a good balance between these two goals is presented. Two goals are avoiding overload and green computing.
- Overload avoidance: The capacity of a PM should be sufficient to satisfy the resource needs of all VMs running on it. Otherwise, the PM is overloaded and can lead to degraded performance of its VMs.
- Green computing: The number of PMs used should be kept to a minimum as long as the needs of all virtual machines can be met. Idle PMs can be turned off to save power.

5. Methodology

5.1. Network Simulator

Network Simulator is the name of a family of discrete-event network simulators, notably NS-1, NS-2, and NS-3. They are all discrete event network simulators, mainly used for research and teaching. NS-3 is free software, publicly available under the GNU GPLv2 license for research, development and use. The goal of the ns-2 project is to create an open simulation environment for network research that will be the first choice of the research community. It must be consistent with the simulation needs of modern network research. You should encourage community contributions, peer reviews, and software validation.

Because the process of creating a network simulator containing a sufficient number of validated, tested, and actively maintained high-quality models is labor-intensive, the ns-2 project distributes this effort among a large community of users and developers.

Currently, ns-2 contains over 300,000 lines of source code, and there may be a considerable amount of contributed code that is not directly integrated into the main release (there are many forks of ns-2, both maintained and unmaintained). It runs on GNU/Linux, FreeBSD, Solaris, Mac OS X, and Cygwin-compatible versions of Windows. It is licensed under version 2 of the GNU General Public License. Although NS-3 is under active development, it is incompressible for the work done on NS-2. So for this project, NS-2 is very supportive.

5.2. CYGWIN

Cygwin is a Unix-like environment and command-line interface for Microsoft Windows. Cygwin provides native integration of Windows-based applications, data, and other system resources with applications, software tools, and data from Unix-like environments. Thus, it is possible to launch Windows applications from within the Cygwin environment, and to use Cygwin tools and applications within the Windows operating context.

It consists of two parts: a dynamic-link library (DLL) as an API compatibility layer that provides most of the POSIX API functionality, and a number of tools and software applications that provide a POSIX.Unix-like look and feel. Cygwin was originally developed by Cygnus Solutions, which was later acquired by Red Hat. It is free and open source software, released under version 3 of the GNU General Public License. Currently maintained by Red Hat employees, NetApp, and many other volunteers.

It consists of a library that implements the POSIX system call API as Win32 system calls, a GNU development toolchain (including GCC and GDB) to support software development, and a number of Unix system equivalents. Developers have ported many programs and packages from Unix, GNU, BSD, and Linux to Cygwin, including the X Window System, K Desktop Environment 3, GNOME, [3] Apache, and TeX. Cygwin allows inetd, syslogd, sshd, Apache, and other daemons to be installed as standard Windows services, allowing Microsoft Windows systems to emulate Unix and Linux servers.

6. Scheduling Algorithm

In the proposed priority-based scheduling algorithm, we modify the scheduling heuristic to execute the highest-priority task with early reservations by preempting completed best-effort tasks. This algorithm shows the Priority Based Scheduling Algorithm (PBSA) pseudocode.

Introducing scheduling helps in achieving service level agreements with quick response from service providers. In our proposed method, QoS metrics such as response time are achieved by first executing high-priority jobs (deadline-based jobs) with the help of the task scheduler from the remaining generated at work.

Scheduling and proposed a particle swarm optimization (PSO) algorithm based on the small position value rule. To improve efficiency, optimization of task scheduling is necessary. In cloud computing, where resources are distributed around the world, data is usually larger, and bandwidth is usually narrower, these issues are even more important. In this paper, the authors introduce an optimization method for task scheduling in cloud computing, and formulate a task scheduling model to minimize the cost of the problem, and solve it by the PSO algorithm. The experimental results show that the PSO algorithm obtains the optimal solution and converges faster than the other two algorithms in large-scale tasks. Moreover, the execution time is shorter than the other two, it is clear that PSO is more suitable for cloud computing.

6.1. Priority based Job Scheduling Algorithm in Cloud Computing

This work proposed a new algorithm for scheduling jobs in cloud computing using mathematical statistics. This algorithm is based on the priority property, which is why it is called a priority-based algorithm. It is based on the multi-criteria decision-making model. In 1980, Thomas Saaty was the first to develop a model for pairwise comparisons based on multiple criteria and multiple attributes, which he called the Analytic Hierarchy Process (AHP). AHP is entirely based on a consistent comparison matrix, so when using AHP, the comparison matrix is calculated based on the reachability of attributes and criteria. In this algorithm, each job requests resources with a predetermined priority. Then, according to the availability of resources, a comparison matrix is calculated for each job. The author also calculated a resource comparison matrix, which is helpful for later career selection. The authors then compute a priority vector (weight vector) for each comparison matrix, and finally compute a normal matrix for all jobs, called Δ. Similarly, calculate the normal matrix of all resources and mark this matrix as γ. The next step in the algorithm is to compute the priority vector (PVS) of S, where S is a set of jobs. PVS is calculated by multiplying the delta matrix by the gamma matrix. In the final step, the algorithm selects the job with the highest compute priority based on the correct resource allocation for that job. Now that the job list is updated, the scheduling process continues until all jobs are assigned to the correct resources. Experimental results show that the algorithm has reasonable complexity. But there are some issues such as complexity, consistency and completion time.

ALGORITHM: Priority Based Scheduling Algorithm (PBSA)

1. **Input:** UserServiceRequest
2. //call Algorithm 2 to form the list of task based on priorities
3. get globalAvailableVMList and gloableUsedVMList and also available ResourceList from each cloud schedular
4. // find the appropriate VMList fromeach cloud scheduler
5. **if** AP(R,AR) != ϕ **then**
6. // call the algorithm 1 load balancer
7. deployableVm=load-balancer(AP(R,AR))
8. Deploy service on deployableVM
9. deploy=true\
10. **Else if** R has advance reservation and best-effort task is running on any cloud **then**
11. // Call algorithm 3 CMMS for executing R with advance reservation
12. Deployed=true
13. **Else if** globalResourceAbleToHostExtraVM **then**
14. Start newVMInstance
15. Add VMToAvailbaleVMList
16. Deploy service on newVM
17. Deployed=true
18. **Else**
19. queue serviceReuest until
20. queueTime > waitingTime
21. Deployed=false
22. **End if**
23. If deployed then
24. return successful
25. terminate
26. **Else**
27. return failure
28. Terminate

6.2. Cloud min-min scheduling (CMMS)

Min-min scheduling is a popular greedy algorithm. There is no attention to the dependencies between tasks in the original min-min algorithm. Therefore, in the dynamic min-min algorithm used, the authors maintain task dependencies by updating the set of mappable tasks at each preparation step. Tasks whose parents are both assigned are placed in the set of assignable tasks. Algorithm 3 shows the quasi-code of the CMMS algorithm.

The cloud scheduler records the deployment plan for all resources using the slot. Once an AR task is assigned to a cloud, the cloud scheduler will first check the availability of reservations on that cloud. Therefore, best effort tasks can be replaced by AR tasks, which is the only case where most of the resources are allocated to other AR tasks. Later, there are not enough resources left for this AR task in the requested time period. If the AR task is not disabled, meaning sufficient resources are available for the task, a random set of required virtual machines is selected.

Due to resource reasons, the estimated completion time of the task may not coincide with the actual completion time Argument in personal cloud. later according to The latest information authors online propose an adaptive scheduling process.

During future online adaptations, the distribution of remaining static resources will be periodically re-evaluated at a predefined rate of occurrence. At each reevaluation, the dispatcher will recalculate the estimated completion time of its tasks. Note that so-called cloud programmers only rethink Tasks at work yield to this cloud, not errands assigned to it.

Algorithm 3 Cloud min-min scheduling (CMMS)

Require: A set of tasks, m different clouds ETM matrix

Ensure: A schedule generated by CMMS

1. For a mappable task set P
2. **While** there are tasks not assigned **do**
3. Update mappable task set P
4. **For** I = task vi ∈ P **do**
5. Send task check requests of vi to all other cloud schedulers
6. Receive the earliest resource available time response and And list of task with their priorities form all other cloud scheduler
7. Find the cloud Cmin(vi) giving the earliest finish time of vi, assuming no other task preempts vi
8. **End for**
9. Find the task-cloud pair (vk, Cmin(vk)) with earliest finish time in the pairs generated in forloop
10. Assign task v k to cloud Dmin(vk)
11. Remove v k form P
12. Update the mappbale task set P
13. **End while**

Algorithm: To compute and assign the priority for each request based on the threshold value and allocate the service to each request‘s.

Step 1: [Read the clients request data i.e, time, importance, price, node and requested server name] Insert all values into the linked list

Step 2: [For each request and its tasks find the **time** priority value based on the predefined conditions] Assign priority value to each task for the client‘s request.

t_p[i] = priority value

Step 3: [For each request and its tasks find the **node** priority value based on the predefined conditions]

Assign priority value to each task for the client‘s request.

n_p[i] = priority value;

Step 4: [For each client‘s input data check whether it is within the threshold value or not]

if (input value is within the threshold limit and total node <= available node)

[Add respective computed time and node priority value and other parameters like importance and price]

Sum[k] = t_p[i] + n_p[i] + importance + price

Print ―Ready to execute available node = available node – total node

else if (input value is within the threshold limit)

sum[k] = t_p[i] + n_p[i] + importance + price

print ―within the limit but it is in queue

else

print ―Exeed the condition

Step 5: [Sort the sum[k] values]

Step 6: Client‘s request is ready to execute from least values of sum[k]

Stop

To run a particular model, huge computing resources are required, such as memory in servers, storage disks, processors, software, etc. Also, some jobs need to run in parallel while others need to run sequentially. In that case, the type of work is also a very important parameter.

In a cloud environment, the type of user, i.e. whether the user is inside the cloud (in the case of a private cloud) or outside the cloud (in the case of a public cloud), is another parameter. It is important to consider the account when submitting your work. Therefore, the developed prioritization algorithm analyzes in detail how it will effectively help the cloud administrator decide or calculate the priority among user requests.

After the resource allocation algorithm has successfully executed, the job requested by the user should be submitted. The main difficulty in allocating resources in cloud systems is making correct decisions for job scheduling, job execution, job status management, etc. In addition to the traditional best fit and binning algorithms, this paper develops an algorithm to do the job. Allocation in a cloud environment will be determined by the cloud administrator.

Prioritization based on client and server requirements and user requests. In the present algorithm, to decide on resource allocation in a better and unbiased way, a technique based on thresholds of all parameters (client and server side) is considered. For example, the number of processors requested cannot exceed 20, etc. (server), the maximum execution time of the job is 200 hours (user).

7. Results and Discussion

7.1. Setup

We evaluate our priority-based performance by simulating a scheduling algorithm. One-on-one task force simulations are done in 10 games. In each run of the simulation, a set of 70 different simulated service applications (i.e. jobs), each including service requests for up to 18 subtasks. We believe in cloud simulation. 70 will randomly request cloud services for short periods of time. Of those 70 request services, 15 applications were in AR mode, while the rest were working best for different SLA goals. The parameters in Table 1 were randomly set in the simulation according to their maximum and minimum values. Since we only focus on scheduling algorithms, we simulate locally without deploying to any existing cloud systems or using VM interface APIs.

Table 1- Parameters and its range

Parameter	Minimum	Maximum
No. of virtual machine in cloud	23	120
No. of CPU in a VM	1	7
Disk Space	8000	100000
Memory	16	2048
Speed	100	1000

7.2. Result

Figure 1 shows the average execution job loss. We discovered the PBSA algorithm. Minimum average execution time. The resource parameter AR job when the job occurs is preferably overridden. Target's part-time live jobs are expected to draw to a close, as resource contention at least eases. Therefore, the adaptation procedure will not significantly affect the implementation date.

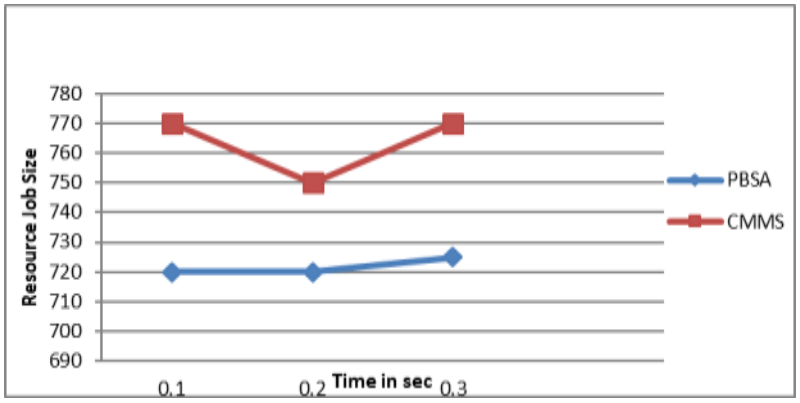
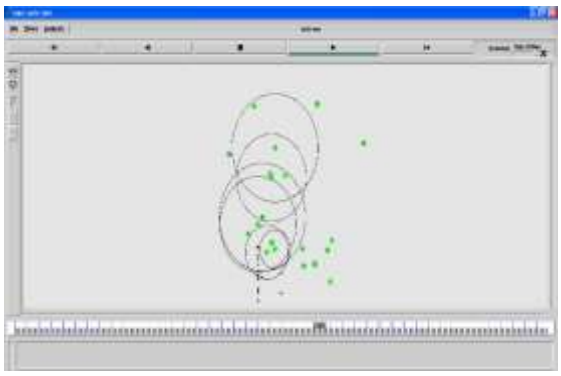


Fig 1. Average job execution time in loose situation



In the difficult case shown in Figure 2, the performance of PBSA CMMS is better. In stressful situations, competition for resources is greatest when the work is actually done, often arriving later than expected. Because AR prioritization works better, the process of adapting and updating information is more effective in difficult situations.

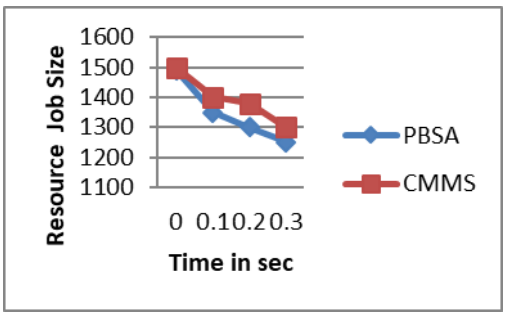
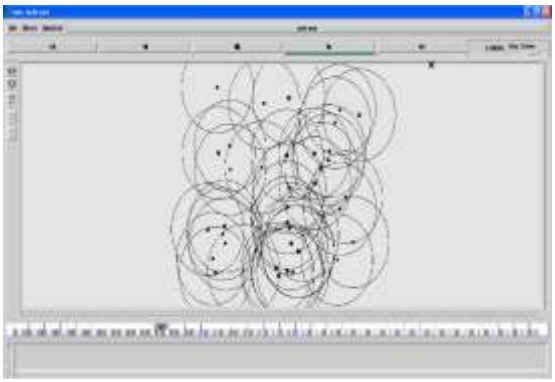


Fig 2. Average job execution time in tight situation

This requires a dynamic estimate of the energy consumed by the method during execution. This was inspired by a recent PowerTutor model that considered the power consumption of the CPU, LCD screen, GPS, WiFi, 3G, and audio interface in the HTC Dream and HTC Magic phones.

Power Tutor shows that the estimated power varies widely between different types of phones and provides a detailed model for the HTC Dream phone used in our experiments. We modified the original PowerTutor model to accommodate the fact that some components, such as GPS and audio, must run locally and cannot be migrated to the cloud.



By measuring the power consumption of the phone under different cross products of extreme power states, the PowerTutor model further shows that the maximum error is 6.27% if each component is measured individually. This shows that the sum of the specific power estimates of the individual components is sufficient to estimate the overall power consumption of the system.

Using this approach, we devised a method with only minor deviations from the results obtained with PowerTutor. We implemented this energy estimation model in ThinkAir Energy Profiler and used it to dynamically estimate the energy consumption of each operating method.

8. Conclusion

Cloud computing resources refer to the management of software (such as database servers, load balancers, etc.) and hardware resources (such as CPU storage, networks, etc.) during selection, deployment, and management to ensure the safe performance of applications. These technologies can improve response time, performance, energy saving, quality of service, SLA. The ultimate goal of resource allocation is to maximize the cloud's benefits from the user's perspective for the cloud and service providers to reduce costs.

There are many strategic resource allocation challenges today. A mechanism must be used to overcome the challenges faced by existing technologies. The architecture should be proposed if it is suitable for high-performance computing and data-intensive applications. Also in actual workload. The mechanism should recommend efficient use of cloud computing resources to allow violations of QoS and SLAs while minimizing dynamic cloud allocations. Additionally, these mechanisms should also be used to provision SaaS and IaaS users.

References

Brintha, N. C., JT Winowlin Jappes, and M. S. Vignesh. "Managing & Detecting Fishy Nodes in MANET using Cloud Concepts." 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT). IEEE, 2020.

Chandran, Ramesh, S. Rakesh Kumar, and N. Gayathri. "Genetic algorithm-based tabu search for optimal energy-aware allocation of data center resources." Soft Computing 24.21 (2020): 16705-16718.

Haoxiang, Wang, and S. Smys. "MC-SVM based work flow preparation in cloud with named entity identification." Journal of Soft Computing Paradigm (JSCP) 2.02 (2020): 130-139.

Sengan, Sudhakar, V. Priya, and Pankaj Dadheech. "Energy and Green IT Resource Management Analysis and Formation in Geographically Distributed Environmental Cloud Data Centre." Energy 29.6 (2020): 4144-4155.

Chakraborty, Chinmay, and Joel JCP Rodrigues. "A comprehensive review on device-to-device communication paradigm: trends, challenges and applications." Wireless Personal Communications 114.1 (2020): 185-207.

Gupta, Punit, and Pradeep Kumar Gupta. "Introduction to Multilayered Cloud Computing." Trust & Fault in Multi Layered Cloud Computing Architecture. Springer, Cham, 2020. 1-13.

Sheikh, Muhammad Sameer, Jun Liang, and Wensong Wang. "Security and privacy in vehicular ad hoc network and vehicle cloud computing: a survey." Wireless Communications and Mobile Computing 2020 (2020).

Devi, Karuppiah, D. Paulraj, and Balasubramanian Muthusenthil. "Deep Learning Based Security Model for Cloud based Task Scheduling." KSII Transactions on Internet and Information Systems (TIIS) 14.9 (2020): 3663-3679.

Venuthurumilli, Pradeep, and Sridhar Mandapati. "Hybrid Invasive Weed Optimization with Tabu Search Algorithm for an Energy and Deadline Aware Scheduling in Cloud Computing." International Journal of Advanced Computer Science and Applications 11.12 (2020).

Aggarwal, Sheetal, and Vandana Dabass. "Design of Energy Efficient Scheduling Algorithm for Cloud Computing Environment." (2022).

Alotaibi, Majid. "Hybrid metaheuristic technique for optimal container resource allocation in cloud." Computer Communications (2022).

Maray, Mohammed, and Junaid Shuja. "Computation offloading in mobile cloud computing and mobile edge computing: survey, taxonomy, and open issues." Mobile Information Systems 2022 (2022).

Sharma, Neetu, and Puneet Garg. "Ant colony based optimization model for QoS-based task scheduling in cloud computing environment." Measurement: Sensors 24 (2022): 100531.

Vemireddy, Satish, and Rashmi Ranjan Rout. "Fuzzy reinforcement learning for energy efficient task offloading in vehicular fog computing." Computer Networks 199 (2021): 108463.

Mallikarjuna, Basetty. "The effective tasks management of workflows inspired by NIM-game strategy in smart grid environment." International Journal of Power and Energy Conversion 13.1 (2022): 24-47.

Kadhim, Abrar Saad, and Mehdi Ebady Manaa. "Hybrid load-balancing algorithm for distributed fog computing in internet of things environment." Bulletin of Electrical Engineering and Informatics 11.6 (2022): 3462-3470.

Sharma, Neetu, and Puneet Garg. "Ant colony based optimization model for QoS-based task scheduling in cloud computing environment." Measurement: Sensors 24 (2022): 100531.

Bhuvaneswari, M., B. Paramasivan, and B. Shunmugapriya. "Enhanced Resource Allocation in Vehicular Adhoc Networks using Colonel Blotto Game Model." (2022).

Jinarajadasa, G. M., and S. R. Liyanage. "Evolutionary Algorithms for Enhancing Mobile Ad Hoc Network Security." Security Issues in Fog Computing from 5G to 6G. Springer, Cham, 2022. 15-30.

Guo, Chao, Moshe Zukerman, and Tianjiao Wang. "Radar: Reliable Resource Scheduling for Composable/Disaggregated Data Centers." IEEE Transactions on Industrial Informatics (2022).