Energy-Efficient Task Scheduling in Cloud Environment

Shamita Phutane[1], Ankith Poojari[2], Saurabh Nyati[3], Bhavna Arora[4]

[1,2,3]BE student, Computer Department, Atharva College, Mumbai University
[4] Professor, Computer Department, Atharva College, Mumbai University

---***---

Abstract - *Cloud data centers consume large amounts of cooling power. Previous research was primarily based on the provision of mission plans, optimizing both computational power and overall performance gains. However, with the development of cloud facts, user needs are changing and cannot be met by traditional scheduling algorithms. One of the necessities is to maintain the value of cooling the facts in the middle as much as possible. According to literature reviews, the value of cooling power is half a million dollars. Executing value in a cloud environment requires a good mission planning approach. Temperature affects not only reliability, but also the overall performance, performance, and cost of embedded systems. This section describes the heat-aware task assignments and set of rules for cloud data centers. Mission plans are built in ways that not only reduce computational costs but also reduce cooling. Rigorous simulations were performed and compared to state-of-the-art graphics algorithms. Experimental results show that the rules of thermal task planning are superior to other strategies. This project aims to provide cloud data centers with a heat-aware task scheduling approach and improve data center performance by adopting task scheduling techniques. The basic motivation is to provide a valuable and powerful green algorithm through the fact center.*

Key Words: **Energy-performance, Green cloud, Multi resource, Task scheduling, energy-aware systems.**

1. INTRODUCTION

Loud computing provides users with convenient, on-demand network access to shared computing resources, making it a promising standard for service providers and users. In most cases, these resources are heterogeneous, inherently allocated, and can be acquired and rejected immediately with nominal supervision or support from the service provider. The increasing cost of use in data centers and the associated environmental hazards have increased the need for intensity-aware computing. Power consumption has emerged as an important issue in cloud computing, but some work has been done to manage power consumption efficiently. It becomes more important while other issues are raised in cloud computing. The cloud computing environment manages a large number of clusters that hold running services in the data center. Services are provided in the form of Platform as a Service (PaaS), Software as a Service (SaAS), and Infrastructure as a Service (IAAS). To take into account the amount of power consumed by overloaded and underutilized computer systems, you need a good algorithm for task planning that can use resources efficiently. The reason is that most servers remain idle or resources are being used inefficiently. It needs a serious concern for society cloud data center lot. To improve the efficiency of energy consumption, we propose a framework for deep reinforcement learning. This document uses an energy-efficient depth forced learning-based job scheduling framework. Reinforcement learning is part of machine learning often used to solve automation problems. The main feature of reinforcement learning is that no pre-configured data is required for training. The agent itself needs to investigate the training data. Forced learning is used to maximize output. In our case, this is energy optimization. For agents specific steps are required to maximize the required functionality defined, this leads to optimization of energy. This is the reason reinforcement learning makes it an ideal learning system for energy-efficient work planning on a cloud platform.

2. REVIEW OF LITERATURE

In a cloud computing environment, user services always require heterogeneous resources (CPU, I / O, memory, etc.). but also reduce energy consumption and the time it takes to complete a user's order.Therefore, scheduling and load balancing techniques are very important for an efficient cloud setup with limited resources. Cloud computing task scheduling has been tackled by many researchers in the past.

Virtualization has become a powerful technology in data centers and cloud computing [3]. This technology allows you to share physical servers or hosts between different virtual machines through a virtual machine monitor (VMM) or hypervisor to increase resource utilization. One of the main benefits of virtualization is Dynamic VM Integration (DVMC), which is used in data centers to reduce energy consumption. Power efficiency is achieved by consolidating VMs into a minimal number of hosts and hibernating idle hosts. DVMC uses the live migration feature provided by virtualization technology to transfer running VMs from one host to another. VM migration is useful if your host commits significantly under or over. Therefore, resource management policies become more flexible through migration operations. This paper [4] proposes a hybrid assignment scheduling set of rules named FMPSO that is primarily based totally on the Fuzzy device and the Modified Particle Swarm Optimization method to decorate load balancing and cloud throughput. FMPSO approach at the start considers 4 changed pace updating strategies and roulette wheel choice method to decorate the worldwide seek capability. Then, it makes use of crossover and mutation operators to conquer a few drawbacks of PSO consisting of nearby optima. Finally, this schema applies a fuzzy inference device for health calculations. The enter parameters for the proposed fuzzy device are the period of tasks, the velocity of CPU, length of RAM, and overall execution time. By including those fuzzy systems, the FMPSO approach achieves the purpose of minimizing the execution time and aiding usage. The FMPSO set of rules, the usage of the CloudSim toolkit, and simulation effects reveal that the proposed approach has a higher overall performance in phrases of makespan, development ratio, imbalance degree, efficiency, and overall execution time evaluating different approaches.

The purpose of these paintings is to distribute Big information for allotted processing withinside the Cloud in a manner to lower the overall strength intake of the Cloud, without scarifying the utility`s overall performance. Therefore, they develop a strength-conscious assignment scheduler (EATS) whose purpose is to lessen the electricity intake of the applied resources and the processing time of a utility. Divisible load packages are a category of packages that may be divided into impartial duties and assigned to distributed sources and not using synchronization and inter-duties communication. The divisible load takes place in lots of scientificand engineering packages to investigate and method Big information, inclusive of looking for a pattern, compression, generic search packages, multimedia and video processing, picture processing, and information mining. The distribution of processing is supposed to boost the overall performance of the allotted utility as compared to its sequential execution. The advanced scheduler to boost the overall performance of divisible load packages in a Cloud Computing environment. The scheduler follows a linear-version method and no longer recollects the energy consumption of the cloud. In this work, the expansion of non-linear programming is primarily based scheduling version which aims at optimizing the overall performance of the utility and the strength intake of the underlying sources. This indicates that idle, and underutilized sources devour a massive quantity of strength as compared to that of complete use. The version takes into attention this commentary to optimize strength intake in the course of distributed computing.

3. PROPOSED SYSTEM

Many cloud architectures are available in the market, but most of them are focusing only on cloud architectural performance. However, in further development of the cloud data center, the needs of the user can not be diversified and can not be achieved by traditional planning algorithms. The Cloud Data Center is equipped with hostile/cold floor tiles on the high floor. Computer room conditininer (crac) is air conditioning for data centers and cold air. Air flows through the front end, absorbs heat as it flows through these frames, and exits in front of the rear end of the frame. Warm air leaves the rack and is returned to the air-conditioned air intake above the warm aisle. Each rack is equipped with many cases, and each case is equipped with many computing resources such as servers and network equipment. A large amount of heat is released, polluting the environment and leading to global warming.

The main purpose of this post is to develop a cloud data center heat-aware task scheduling strategy and apply task scheduling techniques to improve data center performance. According to [3], the task scheduling algorithm finds suitable resources for the task sent to a cloud data center according to the corresponding characteristics. It is speculated that the proposed method can minimize proper scheduling of data center tasks by increasing thermal recirculation, helping the data center operations in energy-efficient modes and achieve savings. Will be done.

Our project uses a greedy approach to develop temperature-based energy-efficient task scheduling algorithms. The main purpose of this post is to develop a cloud data center heat-aware task scheduling strategy and adopt task scheduling techniques to improve data center performance. According to [3], the task planning algorithm finds suitable resources for the task sent to the cloud data center according to the appropriate characteristics. In the proposed method, the increased thermal recirculation can minimize the proper planning of tasks in the data center, and it is speculated that it will help the data center to operate in an energy-efficient model to save. The proposed model is based on DRL where the temperature is considered along with the other parameters. The temperate parameter will help to plan the execution of tasks by the thermal efficiency of the server.

1. With the advancement of cloud data centers, the needs of users are diversifying, which is achieved by the JARS algorithm.
2. To make cloud data centers green and reduce carbon emissions, energy consumption must also be minimized.
3. Greening, the cloud architecture can also minimize the waste generated from the cloud data center
4. Task scheduling is designed to not only reduce computational costs.

3.1. Model Description

Orders submitted by users are independent of each other. There are different servers in the cloud, and each server has only a VM. Also, each VM can only run one job at a time. These VMs are used for online work and planning. VM was assigned to a position, with no prior knowledge of the profession

Many public cloud providers, such as Amazon Web Services in the cloud and Microsoft Azure, have several alternative options for VMs. If you are considering an AWS EC2 instance. It consists of a faster CPU with more processing power and a slower midrange CPU with different processing power. For example, in the case of, small jobs run smoothly with the following jobs, providing the CPU VM with a more powerful CPU to run in more intensive jobs. These instances with different prices on public cloud platforms also differ because these are measured on a pay-as-you-go basis.

The job scheduling process is somehow professionally tracked and jobs are scheduled for specific VMs available in the resource pool on which this energy-efficient model is based. Multiple user input jobs in real-time. Let us explain this further. The number of job queues in the resource pool, for temporary positions. A specific runtime for a job. VMs cannot consider different jobs because other VMs only run one job. If you interrupt the process, you will not be able to do the work in progress and you will not be able to purchase it in advance.. In our model, we will be having S servers and each server will be having exactly one VM on it,

$VMj = \{ VM\_idj , VM\_Comj , Ej \}$ (1)

User jobs are defined by job arriving time, computational power, and user request time i.e QoS.

$Jobi = \{ Job\_idi , arrival\_Timei , Job\_sizei , QoSi \}$ (2)

When the job queue is empty and a job arrives, will be executed immediately even if a job occurs there is a job queue and there is a job with an arrival time before it all jobs prior to the last job will only run once it is planned soon:

$Ti = Ti\ wait + Ti\_exe$ (3)

The job when completes its execution on a VM and returns to the end-user and the time taken by the job to get completed can be evaluated as

$Ti\_leave = arrival\_Timei + Ti$ (4)

4. Implementation

Looking back on previous research on this technical topic arena One can realize some beliefs the study group holds about reinforcement learning. The current strengthening discipline is that it takes a long time to learn something together. Some people think that reinforcement learning techniques can only be used in practice. The method is either pre-training or transfer in the simulator. Knowledge learned between different domains when compared to other machine learning methods, this philosophy limits the technology used. Some people hesitate to answer real questions and make choices for further research in the field. This may be the reason this area has been very successful in video games such as strategy games like Atari or Go with known rules predefined. It contains all the pre-processing required for other proposed algorithms. First, create a network topology that defines the number of data centers, hosts, and VMs. Then configure all the VMs and the value of each VM according to its characteristics. Third, determine the path of your network topology. Lastly calculate the total number of resources (CPU, memory, memory) in each path of the network total each resource. The proposed task describes the planner as an agent.

4.1. Task Scheduling algorithm

The following algorithm is used in our model to predict the best-case scenario considering the energy consumption using both deep Q learning techniques and offline decision techniques such as experience replay and target networks to increase the efficiency

Algorithm 1: Job And Resources Scheduling Algorithm (JARS)

```
for episode =1, do
    Reset cloud server environment to initial state
    with probability e randomly choose
    an action a j ;
    Schedule job j according to action a j, receive
    reward r j, and observe state transition at next
    decision time t j + 1 with a new state s j + 1
    Store transition j
    if j > t and j ≡ 0 then
        if j ≡ 0 then
            Reset
        end if
        randomly select samples state from states
        for each transition in the state do
            target k = rewards
        end for
        end if
end for
```

Algorithm 2: Temperature Seeker Algorithm

```
    for episode in 1, do
        if check enough resources available on the
        server for a substitute task then
            calculatetemperature(Counter, ti) ;
        end if
        if (temperature less than Mtemp) then
            Allocate tasks and resources;
            Mtemp=temperature;
        end if
end for
```

4.3. DRL-DQN Approach

At each time step t, the agent monitors the current state of the environment and takes action. From this interaction, the environment tells the agent how well the agent is doing. The agent's goal is to maximize rewards over time. This issue schedules tasks based on CPU and memory allocation. The selection is made from a stack of tasks. The schedule only considers the CPU and memory, does not fragment the resources, and assumes that the resources consumed by the task are known in advance

State: The state is defined as all tasks in the queue. In this case, each stack has a total of s tasks. You can choose from a total of s states.

Actions: For each action, the agent can select a task or nothing. If the agent does not select anything, the selection phase ends and all selected tasks are processed. . You cannot reselect the selected task, so each time you make a selection, the number of possible actions is reduced until the end of the selection phase.

Reward: The environment works with agent actions. The environment returns rewards and new conditions. The new state is the entire task queued in the previous selection minus the selected task. If the choice is wrong, the environment will punish the agent by setting the reward

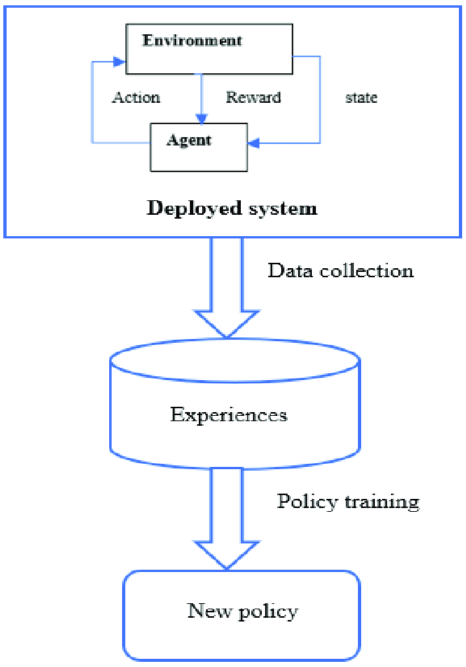

Chart -1: DRL working

Negative value. The total number of resources is important for the correct selection of rewards

Two different Neural networks i.e target network and evaluation network are used in our model. The Target Network is considered a temporarily frozen network. The parameters of the target network are copied from the evaluation network after every step. From there, a search scheme (usually Epsilon Leady) is used to stochastically choose between the best Q-factor action and random action. At a higher level, deep learning works as follows: It uses the current policy to collect samples and store them in the replay buffer, randomly sample the experience from the replay buffer (so-called experience replay), and use the sample experience to update and repeat the Q network. Successive experiences are strongly (temporarily) correlated. Statistical learning and optimization tasks require an independent distribution of data. In other words, you don't want the data you're feeding to be correlated. A random sample of experience decomposes this temporal correlation of behavior and distributes/averages it into many previous states. This avoids large fluctuations and biases (problems that can arise from correlation data) in the Model. To update the Q network, you need to minimize the root mean square error between the Q target value (according to the Bellman equation) and the actual Q output. To reduce errors, this means that the results of the current policy will become more and more similar to the true Q value. Therefore, use the loss function defined above to perform the gradient step of the loss function.

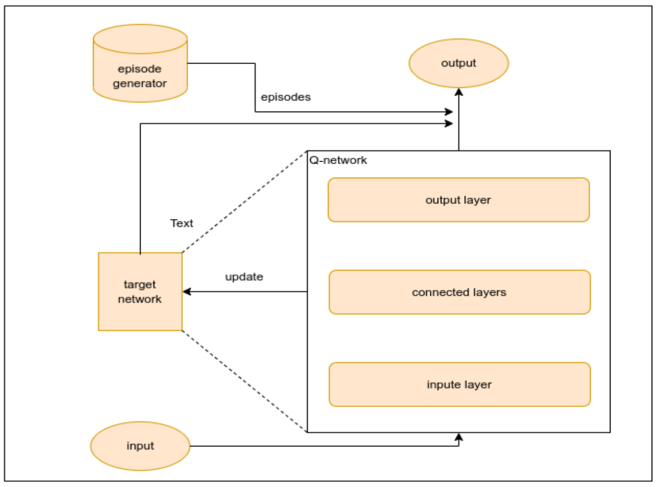

Chart -2: System Architecture

5. Result and Conclusion

This figure shows the success rate in terms of energy efficiency. We have defined various test series. One is a small (1000) workstation, a medium (5000) workstation, and a large (10000) workstation. All jobs of all sizes will be online at different intervals. Each generated job does not depend on any other job. In addition, each job has different parameter values that are efficiently scheduled with energy consumption as well as parameters such as QoS. The comparison is based on the energy factor. Compared to other baselines such as round-robin, random, and earlies algorithms, JARS outperforms them with 80% consistent accuracy for all three workloads, Small, Medium, Large. Other baselines achieved less than 55% accuracy on any workload. The round-robin policy schedules jobs instantly without making a decision. Job allocation is very fast, so if you have a small number of VMs, the response time will be very fast. Round-robin indicates 51-currency when the workload is light and 48-currency when the workload is increased. Medium workload accuracy, 45 ° accuracy for heavy workloads. The Random Policy, as the name implies, randomly assigns jobs to different servers, makes no decisions, and performs random actions, similar to Round Robin, so response times are very fast. The average power efficiency of the random policy was 47%, and the workload adopted was the lightest of the various policies used in the model. The original policy selected the first scheduled job on the server with the highest power consumption and queued other jobs, averaging across the workload, exceeding the round-robin and random policies. You can see that the policy is more efficient than random and round-robin, but still very inaccurate. Deep Reinforcement Learning, on the other hand, proves an average accuracy of over 80% for different workloads and runs the best across the four algorithms used. JARS response times are longer than the other three, but scheduling jobs on a server-by-server basis requires careful decision-making. JARS is superior to the other three algorithms in that it uses an intelligent job scheduling mechanism to minimize power consumption and optimize power consumption. Reinforcement learning requires a long way of improvement and development by mitigating problems, which ultimately leads to great research value on this subject. It will be a new field of study. Reinforcement learning will be an implementable technology that will bring great perspectives to dynamic planning by strengthening the direction of many studies where the work of data centers and users will change dynamically shortly. In a changing environment, cloud computing resources need to be optimally operated. Therefore, the generic JARS-based algorithm is suitable for cloud computing because it uses resources effectively and reduces power and margins. Experimental results show that the proposed method using the largest application and a scheduling algorithm for random scheduling is excellent for further research. The optimized model needs to add other necessary goals such as cost, bandwidth, and loss balance. We also need to focus more on robust algorithms.

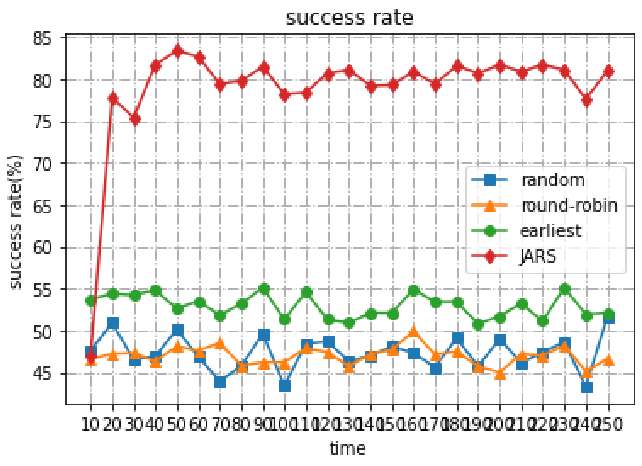

Chart -3: Final Output

REFERENCES

[1]Farahnakian, Fahimeh & Pahikkala, Tapio & Liljeberg, Pasi & Plosila, Juha & Tenhunen, Hannu. (2015). Utilization Prediction Aware VM Consolidation Approach for Green Cloud Computing. 10.1109/CLOUD.2015.58.

[2]Mansouri, Name & Mohammad Hasani Zade, Behnam & Javidi, M.M.. (2019). Hybrid Task Scheduling Strategy for Cloud Computing by Modified Particle Swarm Optimization and Fuzzy Theory. Computers & Industrial Engineering. 130. 10.1016/j.cie.2019.03.006.

[3]Ismail, Leila & Fardoun, A.A.. (2016). EATS: Energy-Aware Tasks Scheduling in Cloud Computing Systems. Procedia Computer Science. 83. 870-877. 10.1016/j.procs.2016.04.178.

[4]Tang, Qinghui & Sandeep, Kornepati & Varsamopoulos, Georgios. (2007). Thermal-Aware Task Scheduling for Datacenters through Minimizing Heat Recirculation. Proceedings - IEEE International Conference on Cluster Computing, ICCC. 129-138. 129-138. 10.1109/CLUSTR.2007.4629225.

[5]Donald, James & Martonosi, Margaret. (2006). Techniques for Multicore Thermal Management: Classification and New Exploration. ACM Sigarch Computer Architecture News. 34. 78-88. 10.1109/ISCA.2006.39.

[6]Chen, Y., Das, A., Qin, W., Sivasubramaniam, A., Wang, Q., & Gautam, N. (2005). Managing server energy and operational costs in hosting centers. ACM SIGMETRICS Performance Evaluation Review, 33(1), 303–314. https://doi.org/10.1145/1071690.1064253

[7]Springer, R., Lowenthal, D.K., Rountree, B., & Freeh, V.W. (2006). Minimizing execution time in MPI programs on an energy-constrained, power-scalable cluster. PPoPP '06.

[8]Bash, Cullen & Forman, George. (2007). Cool Job Allocation: Measuring the Power Savings of Placing Jobs at Cooling-Efficient Locations in the Data Center.. 363-368.

[9]Reinforcement Learning Techniques for Optimal Power Control in Grid-Connected Microgrids: A Comprehensive Review - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/A-data-flow-diagram-for-a-batch-reinforcement-learning-algorithm-A-memory-location-is_fig3_346110033 [accessed 30 Mar, 2022]

[10]T. Heath, B. Diniz, E. V. Carrera, W. Meira Jr, and R. Bianchini, "Energy conservation in heterogeneous server clusters," in Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming.ACM,2005,pp.186–195.

[11]P. Ranganathan, P. Leech, D. Irwin, and J. Chase, "Ensemble-level power management for dense blade servers," in ACM SIGARCH Computer Architecture News, vol. 34, no. 2. IEEE Computer Society, 2006, pp.66–77.

[12]Shu et al.: A novel energy-efficient resource allocation algorithm based on immune clonal optimization for green cloud computing. EURASIP Journal on Wireless Communications and Networking 2014 2014:64

[13]P.Bohrer,D.Cohn,E.Elnozahy,T.Keller,M.Kistler,C.Lefurgy, R. Rajamony, F. Rawson, and E. Hensbergen, "Energy conservation for servers," in Workshop on Power Management for Real-Time and Embedded Systems, 2001.