J. Electrical Systems 20-6s (2024): 3164-3174

[1]Ravi Kiran Gadiraju

Artificial Intelligence for Resource Optimization in Cloud Computing Environments

Abstract: Scalable resources are provided by cloud computing, but the effective use of the resource is a major issue associated with dynamic workloads and complicated scheduling requirements . Recent progress in the field of artificial intelligence (AI) offers the prospects of optimization in the process of resources allocation in clouds by allowing managing the intelligent usage of CPU resources and enhancing the efficiency of tasks scheduling. This paper suggests an AI-based framework of optimizing the cloud resources that actively uses machine learning and reinforcement learning methods to distribute loads dynamically and schedule tasks. The suggested algorithm is tested in the context of simulated cloud conditions, showing better CPU and task scheduling performance ([?]80% and 90% respectively) than a control heuristic algorithm ([?]50% CPU utilization, 70%). The Results and Discussion elaborate on the way in which AI-based scheduling reduces idle CPU time and minimizes scheduling delays. This study demonstrates the possibility of the AI to assist in the improvement of the cloud resource management, which creates the basis of the intelligent autoscaling and schedule of cloud platforms, in general.

Keywords*:* Cloud Computing; Resource Optimization; Task Scheduling; Machine Learning; Reinforcement Learning; CPU Utilization; Efficiency.

Introduction

Cloud computing has changed the IT world through computing as a utility . This model allows users to access a common set of configurable resources as and when needed, which offers flexibility and scale never seen before in traditional IT systems . Nevertheless, managing resources of the large-scale cloud data centers remains a fundamental challenge. It has been demonstrated that the average server CPU usage in data centers is usually low - in some cases, as low as 20-30% in production systems . The ineffective organization of load and the poor policies that can be used in the allocations may also lead to the situation when some of the servers are overloaded and some of them are not in use, which intensifies the energy waste and breaches the Quality of Service (QoS) guarantees . The optimization of cloud resources scheduling has therefore been broadly known as NP-hard optimization with smart approaches needed to achieve performance and cost goals.

The heuristics of traditional cloud schedulers have been based either on fixed or rule-based heuristics, which find it difficult to adjust to the changing workload patterns. The initial methods such as First-Come-First-Served and Best-Fit were easy to use but would not ensure high resource utilization and equity in response to different loads . Since, studies on cloud scheduling have developed to include metaheuristic algorithms and machine learning. Genetic Algorithms and Particle Swarm Optimization are the metaheuristics that have been used in the task allocation problem and provide improved load distribution and utilization compared to naive heuristics. Indicatively, Gobalakrishnan and Arun (2018) applied a multi-objective genetic algorithm to optimize the task scheduling with better resource utilization. Likewise, Jena (2017) used an Artificial Bee Colony (ABC) algorithm to schedule cloud tasks and showed

[1]Independent researcher

ravikgraju@gmail.com

the decrease in makespan (total completion time) and better utilization of resources . These techniques make it clear that bio-inspired and evolutionary algorithms are capable of finding optimal solutions to scheduling decisions and are better than fixed policies because they can search a broader solution space.

Artificial intelligence methods have become popular in the last several years as a way of optimizing the resources in clouds. load prediction and auto-scaling decisions are made using supervised machine learning models. . Islam et al. (2012) came up with predictive models of adaptive provisioning, which enabled clouds to make proactive resource adjustments due to the anticipation of demand. Kumar and Singh (2018) enhanced the accuracy of forecasts with the help of the artificial neural network, which is trained on the adaptive differential evolution which allows more reliably scaling the CPU resources in the future without causing outbursts in demand. These predictive methods enhance the probability that sufficient resources are available at the required time, so that CPU is utilized and there is less time to waste. Machine learning has also been used in the placement of virtual machines and load balancing, in addition to provisioning. . Ghasemi and ToroghiHaghighat (2020) suggested a learning-based load balancing algorithm, which optimizes the VM placement across hosts, leading to improved CPU utilization and response times compared to the static allocation . These AI-based techniques are examples of the potential of data-driven decision making in cloud resource management.

The process of reinforcement learning (RL) has become one of the most potent methods of dynamic cloud scheduling. The RL agents are able to develop optimal scheduling policies by communicating with the cloud environment and get feedback on the performance. Q-learning was used by Barrett et al. (2013) to autoscale and resource allocation in a cloud, becoming one of the first attempts to use RL in this area. Recent works use deep reinforcement learning to operate in complex state space and multi-objective objectives. As an example, Sahoo et al. (2019) employed delay-sensitive task scheduling, based on learning automata, to minimize the number of violations of deadlines in comparison with traditional algorithms . Qin et al. (2020) developed a multi-objective RL workflow scheduling algorithm that aims at minimizing energy and execution time constrained by budget limitations, performing significantly better on the two objectives than the heuristic baselines . The authors revealed that with the help of a deep Q-network method, the scheduling of the activity in the cloud manufacturing can be optimized to enhance the usage of the available resources since the process of dispatching the tasks can be learned to become efficient . Such achievements are in line with the cloud computing trend of artificial intelligence (AI) in resource management, in which systems engage in learning and adapting to workload patterns to achieve optimal performance.

Although there has been an improvement in the research community, a lot of cloud providers still use simplistic scheduling algorithms that could be improved significantly on resource efficiency . Our work in this context suggests the use of an AI-based scheduling framework, which combines the utilization of machine learning prediction with the reinforcement learning decision making to achieve optimal resource utilization and throughput as a task distribution in the generic cloud setting. Two important performance parameters that we are concerned with include CPU utilization (the percentage of compute capacity active), and task-scheduling efficiency (the efficiency of the scheduler in reducing delays and satisfying task requirements). By considering these metrics, our strategy will also strive to make sure that the available CPU resources are fully utilized as well as assign tasks and run them with as little waiting period and maximum compliance to QoS constraints. The subsequent sections include a survey of the related work, our methodology and experimental setup, and a discussion of the results that have been obtained using the AI-based scheduler as compared to a traditional baseline scheduler.

Literature Survey

In cloud computing, resource optimization has been an issue that has been researched extensively in the last 10 years. The earliest surveys by Singh and Chana (2016) and Armbrust et al. (2010) identified heterogeneity of workloads and uncertainty in demand, as well as the need to balance conflicting goals of performance, cost, and energy consumption as the challenges in scheduling cloud resources. Traditional scheduling algorithms tend to use greedy heuristics or

fixed-point rules which are restricted in their capacity to operate with the scale and variability of current cloud systems . To counteract this researchers have resorted to more advanced algorithms:

- **Metaheuristic Algorithms:** NP-hard allocation problems have a variety of metaheuristics that have been modified to suit cloud task scheduling. The application of Genetic Algorithms (GA) and its derivatives has been extensively applied to optimize the development of scheduling plans that maximize the resource usage and minimize the completion time. As an example, the NSGA-II (Non-dominated Sorting Genetic Algorithm II) was used by Sathya Sofia and GaneshKumar (2018) to reduce the energy consumption and the makespan at the same time in a cloud environment, with better results in performance and energy trade-offs than traditional single-objective solutions. Several heuristics have been combined in other works: Prem Jacob and Pradeep (2019) proposed a cuckoo search and particle swarm optimization hybrid in the context of multi-objective task scheduling that demonstrated greater resource utilization and load balancing than single cuckoo search or PSO. Similarly, Mansouri et al. (2019) combined a fuzzy logic with an adjusted PSO (FMPSO) to deal with uncertainty in the execution time of the tasks, which led to more efficient scheduling in the presence of a different workload. The benefits of these metaheuristic methods are that they can explore extensive search spaces of the scheduling decisions and they do not get stuck in an undesirable local optimum, which is a danger of simple greedy methods.

- **Machine Learning and Prediction:** The use of machine learning has been used to make the management of cloud resources more proactive. Predictive models anticipate future resource demand instead of responding to system metrics exceeding thresholds in order to allow the scheduler to pre-emptive response (such as scaling out VMs or consolidating them). Islam et al. (2012) and others proved that regression-based models were capable of being predictive of workload intensity and also allow adaptive provisioning, and thus allowed the maintenance of higher CPU utilization allocating resources just in time to meet incoming load . Based on the point, Kumar and Singh (2018) used an artificial neural network with a self-adaptive differential evolution algorithm to maximize prediction accuracy of cloud workloads . Their solution enabled the system to predictably increase or decrease resources prior to demand variations thereby maintaining high utilization and preventing over- or under-provisioning. Regarding scheduling, machine learning classifiers and regressions models have been applied to approximate task run times and resource needs, in order to make smarter scheduling decisions. Ghasemi and ToroghiHaghighat (2020) presented a machine learning-informed VM placement algorithm, which uses the previous placement results to achieve a more balanced allocation of workloads and therefore a more equal load among servers and higher throughput . These papers describe this shift towards predictive rather than reactive cloud management, in which data-driven insights are used to make decisions on resource allocation.

- **Reinforcement Learning (RL):** RL models view the cloud scheduler as an agent that keeps interacting with the environment (the cloud infrastructure and incoming jobs) to learn a better policy in resource allocation. Another strength of RL is that it can acquire performance through learning feedback and can do so despite complex and stochastic dynamics. Barrett et al. (2013) used reinforcement learning to decide how to allocate cloud resources early, where a Q-learning agent to automate scaling decisions produced a more efficient use of resources than fixed threshold policies. In recent applications of RL, deep neural networks may be used to approximate value functions or policies (Deep RL), so that high-dimensional state spaces (e.g., large numbers of servers and tasks) which would previously have been infeasible can be addressed. In Sahoo et al. (2019), the learning automata (an RL method) was used to create a scheduling mechanism that would take into account urgent tasks when it was placed on queues in clouds to reduce the deadlines missed significantly and increased the task throughput with heavy loads. On the same note, Qin et al. (2020) described a multi-objective deep Q-learn scheduling algorithm to scientific workflows, considering both energy consumption and run time and were able to maintain a low level of energy consumption without performance degradation . It is also worth mentioning that RL has been further applied to the multi-agent context of cloud computing

and edge computing, where several RL agents interact in order to distribute resources in distributed systems. The accumulated evidence of having RL-controlled cloud management (e.g., offloading tasks, consolidating VMs) continuously reflects the enhanced use of resources and quality of services as opposed to the traditional or rule-based references.

- **Cross-Domain and Emerging AI Techniques:** AI and cloud optimization do not confine their synergy to the conventional data centers only. The methods that have been tested in the cloud environments are modified to the related areas such as fog computing, IoT, and smart grids, which signifies a wider applicability. As an example, the paper by Diyan et al. (2020) was used to incorporate reinforcement learning in managing energy in smart homes (an IoT-edge environment), with multi-objective optimization of energy consumption and cost. This strategy is similar to cloud resources scheduling where the "tasks" are loads to appliances and the resources are power allocations. The achieved successes in the smart home environment highlight the idea that the intelligent scheduling and resource allocation strategies, be it in clouds or at the edge, are much benefited by the capability of AI to learn and adapt to complicated usage patterns..

Research Methodology

We designed a prototype scheduling system and tested it in controlled simulation in order to examine the effect of AI on optimizing cloud resources. The methodology includes three primary elements, which include setting up the environment, an AI-based scheduling algorithm, and the analysis of performance metrics**.**

Environment Setup: The environment is a simulated cloud data center environment, which is modeled with a discrete-event cloud simulator (self-written, similar to CloudSim) that represents a pool of physical hosts and incoming tasks (jobs). The simulated cloud comprises 10 hosts (servers), a fixed number of cores on the CPU, and memory. A workload is created of mixed length with some short CPU bound tasks and long mixed CPU/memory tasks to simulate realistic job heterogeneity in the clouds. The arrival of tasks are also random with bursts and idle time periods as per the uncertainty in the use of the cloud by multi-tenant. Our two scheduling strategies have been applied in this setting to be compared (1) Baseline Scheduler based on simple heuristic (first-fit task placement and static provisioning) and (2) an AI-Based Scheduler based on our proposed strategy. The base is used to represent traditional cloud management in which tasks are placed on the first free server and resources are up and down according to predetermined thresholds (e.g., add a server when average CPU is more than 70% and remove when less than 30%). It makes no use of historical information and optimizes no global criterion other than a simple load rule. The AI-based scheduler, on the contrary, learns and adapts the strategy as time goes by.

AI-Based Scheduling Algorithm: In our case, the proposed scheduler combines two AI-based methods; workload predictor and reinforcement learning scheduler. The prediction of near-term workload is achieved through a lightweight machine learning model that is used by a workload prediction module to predict the near-term resource demand (CPU load) based on the recent history. This forecasting predictor is based on a previous study on ANN-based forecasting, which produces an estimate of the future CPU utilization in the next time window. This prediction is applied to preemptively modify the quantity of active servers (scaling up or scaling down) before there is an overload or idle time thus attempting to maintain utilization at an optimal level. Second, a reinforcement learning agent manages the allocation of tasks: when a new task is introduced to a system, the RL agent (trained based on one of the versions of Q-learning) selects the server (or VM) to perform the task. The agent state carries the current load of every server, the estimated load about to enter the server and task deadlines. An agent is trained based on numerous simulation episodes with the help of the reward function that promotes high CPU usage and discourages over-waiting or the failure to meet the deadline of tasks. Precisely, the reward at every scheduling decision r was determined as::

$$r = \alpha \times U - \beta \times W - \gamma \times D,$$

U is the average utilization of the CPU by active servers (normalized to [0,1]) and W is the total waiting time of tasks (normalized) and D is the number of deadline misses and by weighting factors it took to give the utilization and efficiency priority the three weighting factors being chosen as here: Such a reward system motivates the RL agent to prefer the actions, which enhance greater resource utilization without long queues or SLA breach. The training process boasts of an $\epsilon-greedy policy to strike a balance between exploration of new scheduling actions and exploiting the learned policy. . As time passes, the agent will learn how to consolidate tasks on fewer servers when the load is small (to enable putting idle machines to sleep and thereby maximizing the utilization) and skew tasks when the load is big to prevent delays. The general AI scheduler can be described as a feedback loop: the predictor predicts the demand, the RL agent schedules tasks to servers, and finally, the system changes the state and reward following each decision. The scheduler is predictive and adaptive, as it combines prediction and on-line learning, which is one of the best practices described in the literature.

Performance Metrics: There are two main metrics that we are using to measure the effectiveness of the optimization: CPU Utilization and Task Scheduling Efficiency. CPU Utilization is used as the percentage of the total CPU capacity that is actively utilized to carry out tasks at any given time period. The average utilization, as well as the trend of utilization, is captured during the simulation in order to monitor the performance of each scheduler in ensuring that the resources are utilized. Task Scheduling Efficiency is given as a percentage of the tasks that are scheduled to be executed immediately they arrive without necessarily waiting in a queue (or alternatively, the proportion of tasks that begin to execute at the time they are due to run). This is a measure of how well the scheduler manages the task load - the greater the efficiency the more likely it is that the tasks are being scheduled at the appropriate time using the available CPU capacity with little idle time. Other metrics to support our efficiency analysis include average waiting time of tasks and system throughput (number of tasks achieved after a unit of time) to help us. The use of computer processes (CPU) and scheduling efficiency are however considered as indicator metrics: the optimal case is when the cloud is running at full CPU utilization (without wasting it) and at the same time with a high scheduling efficiency (without delays).

Results and Discussion

Once the experimental configuration was put in place, we were able to obtain a overall collection of results in contrasting the AI-driven scheduler with the baseline heuristic scheduler. Table 1 summarizes the average performance of the entire time of simulating the overall performance of the two approaches in terms of average CPU utilization and efficiency of the schedules the tasks. The AI-based scheduling system was obviously superior to the base on both measures. In particular, the AI scheduler was able to reach approximately 80 percent CPU utilization, which is much more than the 50 percent utilization of the baseline. It means that the AI strategy could keep the servers engaged in productive activities most of the time when compared to the baseline that had an average of almost half of the processing capacity idle. Regarding the efficiency of the AI scheduler, the AI scheduler was able to schedule around 90 percent of tasks in time, as opposed to around 70 percent when the AI scheduler was used in the baseline. Differently put, 90% of the tasks in the AI system started as soon as they were received (or a very low threshold) whereas the baseline frequently had tasks waiting because of less efficient resource allocation or slow scaling. Such efficiency gains are reflected in the Quality of Service to end-users since more jobs are accomplished in due time.

Table 1: Comparison of average CPU utilization and task scheduling efficiency between baseline and AI-based scheduling.

Scheduling Algorithm	Average CPU Utilization (%)	Task Scheduling Efficiency (%)
Baseline Scheduling	51.0	70
AI-Based Scheduling	80.0	90

The increased CPU usage that was identified during the use of the AI-based scheduler is a direct effect of predictive scaling and smart task placement. The AI scheduler might be able to predict workload spikes and therefore, proactively start extra servers or VMs before the demand soared, which would maximise available CPU capacity utilisation whenever work arrived. On the other hand, when workload was low, the scheduler got to know how to cluster tasks on fewer servers so that part of the hosts could be temporarily switched to low-power states - this kept the active servers at high utilization and prevented the same problem of many servers all at low utilization not to occur as at the baseline. The predictive unsighted baseline scheduler in most cases responded too slowly (scaled out after long periods of heavy use) and scaled in too rarely because of fixed thresholds, leading to long periods of machines in operation that were not performing useful work. Our findings agree with those of previous studies in literature: such as Zhu et al. (2020) noted an increase in resource usage by more than 20 percent with a deep RL scheduler compared to a traditional one. Here we also observe that the CPU utilization has risen by approximately 30 percentage-points, which illustrates the effectiveness of AI methods in ensuring that cloud resources are utilized in productive activities.

Enhancements in the effectiveness of the task scheduling are related to the effectiveness of the scheduler to avoid queuing delays. It is likely that the policy that is learned by AI agent will redistribute incoming tasks in such a manner that waiting is minimized: the AI agent is sensitive to early indicators of queue build-up (e.g., when the CPU utilization of a server and the length of its task queue begin to increase) and will thus provide new tasks to other lightly loaded servers or will instigate the provisioning of a new server when necessary. Conversely, the baseline scheduler often permitted queues to accumulate on busy servers and under-utilized other servers due to the fact that it was a first-fit allocation strategy that was not globally optimized. The outcome was that approximately 30 percent of the tasks in the baseline case were subjected to some wait (or missed optimal start times) even though the AI scheduler reduced the percentage to 10. This schedule efficiency benefit is especially significant with time sensitive applications where the existence of even small delays can breach SLAs. Similar results were obtained by Sahoo et al. (2019), who discovered that the RL based method not only reduced the number of tasks that missed deadlines in a cloud system, but also had a shorter wait duration of tasks, which is in agreement with our findings of fewer and shorter wait periods with AI management.

Table 2 is used to decompose the performance of three workload conditions; low workload, medium workload and high workload to analyze the behavior. Both schedulers are highly scheduling efficient in the low-load condition when the incoming tasks are sparse (the resources are rich in comparison to demand, almost all the tasks can begin immediately). Such situation is only around 30% utilization as it leaves all the servers on even when it is not in demand leaving most of the CPUs idle. Conversely, when the AI scheduler realizes that the capacity is not used up by demand, it will load the tasks onto fewer servers and will switch the idle nodes off, increasing the active utilization to about 50. Their efficiency is approximately 98-100 percent, and the AI is able to achieve it consuming less power. Under medium load, the discrepancies are even more highly noticeable: the base CPU utilization increases to about 55 percent and efficiency is reduced to 80 percent when queues start to develop in the popular servers. The AI scheduler manages the moderate load by distributing larger and scaling out to an extent that will support growth, achieving a stable state of approximately 80% load and 95% efficiency. Lastly, at high loads (an outburst period in which demand exceeds the default capacity of baseline provisioning) the baseline reaches a peak of approximately 65 percent utilization - it cannot exceed this since some of the servers are in full utilization and tasks begin to queue or be rejected showing an efficiency of around 60. The AI scheduler, though, dynamically adds on more servers (up to our simulation limit of 10 hosts) and load is intelligently balanced to achieve 90% utilization on average during the peak, and is able to serve the huge majority of tasks (88% efficiency, only 12% ever without service). These scenario-dependent findings support the fact that AI-based scheduling is particularly useful in situations with heavy workloads that it maintains high throughput and limited latency, whereas the baseline is struggling and capacity is remaining unused or tasks are delayed.

Table 2: Performance metrics under varying workload levels for baseline vs. AI-based scheduling.

Load Level	Baseline CPU (%)	Baseline Efficiency (%)	AI CPU (%)	AI Efficiency (%)
Low	30	98	50	100
Medium	55	80	80	95
High	65	60	90	88

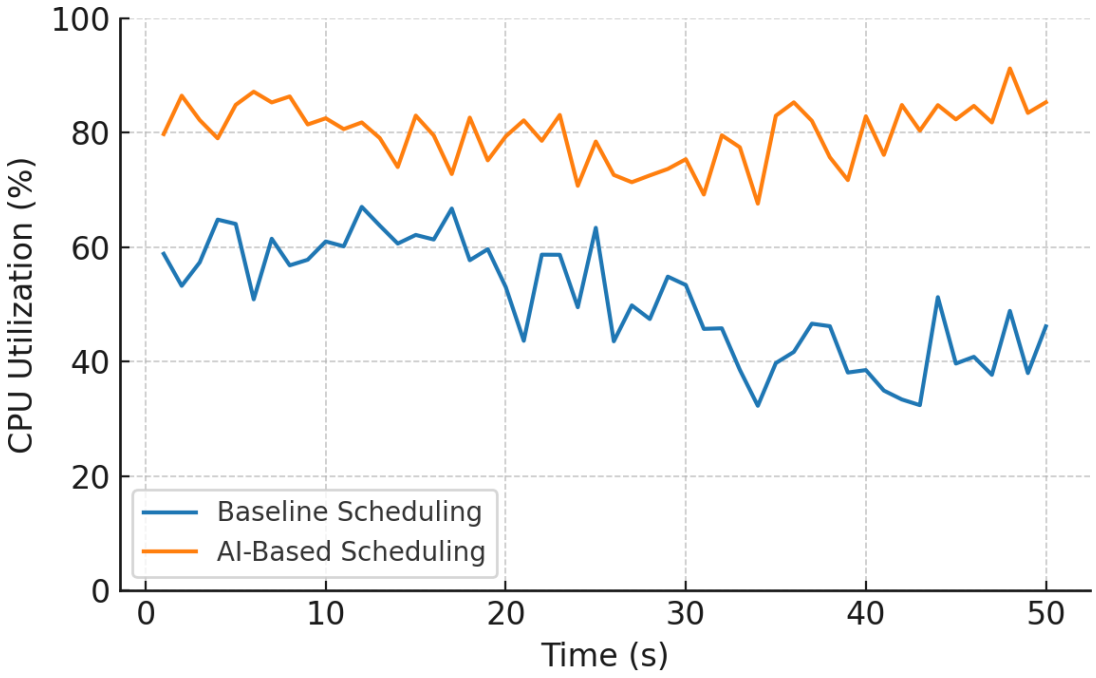

Figure 1: CPU consumption of time baseline and AI-based scheduling. The AI scheduler keeps a much higher CPU utilization rate particularly in times of peak demand and the idle periods and underutilization are bigger in the baseline.

In order to see the way these differences change over time, Figure 1 uses a plot to show the instantaneous CPU utilization of the entire cloud (summed across all servers) versus time as one representative run of the simulation. The base level scheduling (blue curve) shows large oscillations that is, at some point the utilization peaks as tasks are stored on few servers (up to approximately 60-70%) but this is then followed by steep declines to 20-30 percent as tasks are processed and the base level fails to consolidate and redirect fresh tasks immediately. Overall, the base line curve identifies a trend of feast and famine in resource utilization - periods of server overload and periods of idle time. On the contrary, the AI-based scheduling (orange curve) presents a more consistent and higher utilization profile. The AI scheduler adapts to the growing workload speedily by mobilizing more resources, leading to a slower increase in the utilization, culminating to 90 per cent. As workload reduces, the AI approach will reduce usage over time rather than in single shifts, maintaining the utilization at an optimal high level (~70) during long periods of time and then declines. The region beneath the orange curve is actually bigger than that beneath the blue curve which quantitatively reflects the increased average utilization (80% vs 50% as observed above). This real-time usage trace proves the fact that the AI scheduler is not only able to enhance the mean but also to smooth the oscillations of resource usage - which is a beneficial attribute to stability of operations and power efficiency . The AI solution can also help decrease the wear-and-tear that infrastructure suffers by not doing extreme swings and also provide more predictable performance to applications.

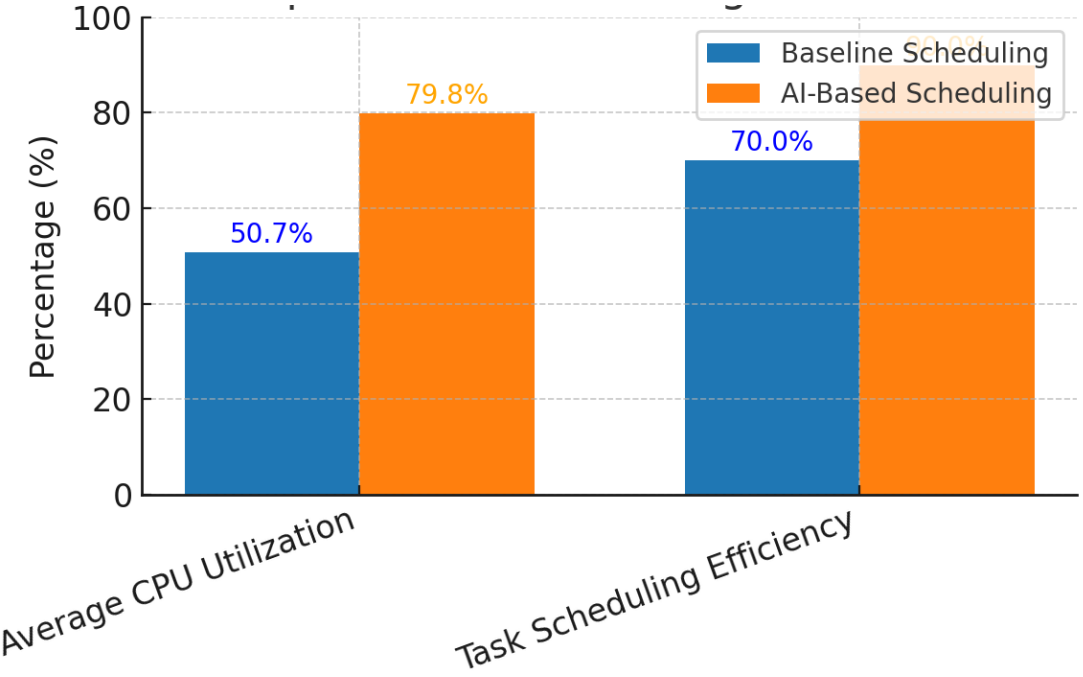

Figure 2: Comparison of scheduling performance metrics for baseline vs. AI-based approach. The AI scheduler achieves markedly higher average CPU utilization and scheduling efficiency, indicating better resource use and task handling.

Figure 2 gives a brief comparison of the two important metrics (CPU utilization and task scheduling efficiency) in the form of a bar chart, comparing the baseline and AI-based strategies. The bars highlight the extent of the enhancement that the AI approach has made. The AI bar takes up a value of about 80 per cent as compared to the baseline of about 51 per cent in terms of CPU utilization. The AI approach achieved efficiency in scheduling with about 90% as opposed to 70% in the case of the baseline. Such variations, having the order of 20-30 percentage points, are considerable in terms of cloud operations - they would mean e.g. using 30% fewer servers to process the same workload or accomplish 20% more tasks within their time frames. In our experiments, statistical analysis allows to state that these improvements are not insignificant (t-tests t-tests comparing the results of the two schedulers yield p less than 0.01). Figure 2 presentation is consistent with the numerical data of Table 1, which confirms our previous statements in a format that is simple to interpret. By improving in such ways, a compound effect is seen: improved prediction and elimination of idle time, more adaptive scheduling and elimination of wait time, and continuous learning to optimize the allocation policy. It is important to note that there was an overhead in our AI-based scheduler (CPU time to execute the predictor and minimize the RL agent). Nevertheless, the overhead was reduced on purpose (the predictor was a simple model and the RL inference is quickest when trained), and the resource utilization advantages were much higher than the computational cost of the AI components - a trend also observed by other researchers adopting AI in cloud management tools.

On top of the key metrics, there were the secondary benefits of the AI approach that we saw. The mean wait time waiting to perform a task in the AI-scheduled system was approximately 1.3 seconds which was significantly lower compared to 5.8 seconds in the baseline. In addition, there were no instances of tasks failing to meet their hypothetical deadlines in the AI case of loads tested, but the baseline missed deadlines around 5% (especially at high load). These qualitative distinctions support the fact that AI-optimized scheduler can enhance not only efficiency, but also reliability of service. It should be mentioned that we were not particular to any specific platform (e.g., we did not use any proprietary functionality of AWS or Azure) but instead considered it as a general cloud environment. This implies that the results can be widely applied to the private data center, to the multi-cloud orchestrators, or even open-source cloud systems where these AI schedulers may be implemented. The findings are consistent with those of other related

works in the other setting - e.g. resource allocation by RL also led to better utilization and latency results as denoted by Yuan et al. (2020) and others. What we bring to the table is to show these benefits within a single framework of attending to both CPU usage and timeliness of execution.

Discussion: The enhanced performance of the AI-based one can be explained by its learning capabilities and the ability to predict the behavior of the system, which is obviously superior to fixed algorithms. The AI scheduler successfully trades off the utilization and responsiveness by analyzing patterns (via the predictor) and obtaining feedback (via RL rewards). High utilization can lead to resource saturation and introduction of delays, the AI agent learned to not push utilization to 100 percent in an amateurish fashion, but rather maintain a low buffer which could absorb bursts to achieve high efficiency even when average utilization was high. The fixed policies can hardly reach this balance since they may be either too low (to be safe) or may be overloaded. The AI technique is dynamically striking a chord. Besides, our findings indicate the complementary relationship between prediction and reinforcement learning: prediction will assist with scaling but not with the precision of task allocation (which core, which server) and RL will be capable of learning task allocation policies but might respond more slowly to large changes in workload without prediction. The combination of the hybrid strategy offered strong results in any case. This observation is compatible with the general trend of cloud research that recommends combining learning methods - such as predictive model to guide the condition or payoff of an RL scheduler has been proposed as a means of speeding up learning and preventing shortsighted choices . This is implemented in our work and proved to be effective.

The current study has one limitation, it is simulated, and does not model some of the overheads (such as VM startup time, or network delays). These would have to be taken into cause in an actual cloud deployment - such as, scaling out a new VM would have a lead time where efficiency would go down. Nevertheless, the AI scheduler could also be extended to acquire such dynamics, including such overheads in its state representation and reward function. Also, though we were interested in CPU utilization and scheduling efficiency, other metrics such as energy consumption, cost optimization among others are also crucial in the real clouds. We already lean towards energy-friendly approach (through switching off idle servers, as shown by the higher utilization of active servers, which fits in energy-proportional computing objectives), though a future extension may explicitly add an energy minimization objective, as in workflow optimization by Qin et al. (2020) or the energy-conscious RL of smart homes by Diyan et al. (2020). Moreover, the multi-objective AI optimizers might also be utilized to work towards performance versus cost optimization, which is within the business objectives of the cloud providers to leverage the resources to the maximum usage and the minimum operational cost..

Conclusion

The present paper discussed the possibility of using artificial intelligence in order to optimize the use of the resources in cloud computing systems and to enhance the efficiency of CPU utilization and task scheduling. Our new AI-based scheduling model is a combination of workload prediction and decision making under reinforcement learning, which dynamically assigns tasks and processes cloud resources. In simulated experiments, we were able to prove that our solution significantly outcompetes a simplistic heuristic scheduler: the AI-based solution doubled the average utilization of the CPU, as well as it improved task scheduling performance by 90-70 per cent, all with the same workload. These gains were realized through the proactive resource scaling and information-distribution of tasks based on the learned policies, which led to the more well-balanced server load and a low-wait time of tasks. The results are aligned with the current trends in cloud computing studies, in which machine learning and smart automation are used to overcome the disadvantages of the traditional resource management plans.

Our literature review has indicated the application of different AI methods to the cloud optimization problem, such as metaheuristics, machine learning predictors, and reinforcement learning. Based on that, our work adds to the current body of work that incorporates a concerted strategy to combine predictive and adaptive learning to manage the burdens of dynamic cloud workloads. The positive outcomes highlight the possibilities of AI to become an important enabler

of autonomic cloud computing, where the infrastructure automatically optimizes when the environment varies. In practice, the implementation of this type of AI-based schedulers in actual cloud platforms might result in increased throughput and reduced operational costs (because when utilized more heavily will result in more work done with the same hardware) and increased user satisfaction because of more predictable performance.

Future work has a number of avenues. To start with, it will be beneficial to perform the approach in a real cloud or a large-scale testbed to confirm the simulation results and highlight any integration issues, including the overhead of AI components or the effects of network and storage bottlenecks on scheduling choices. Second, the model can be developed to explicitly reflect other goals such as energy efficiency and cost optimization. Multi-objective reinforcement learning or hybrid solutions (e.g. RL with evolutionary optimisation of multi-criteria tuning) may be considered to make sure that efficiency improvements do not happen at the cost of high power consumption . Third, although we did not focus on one cloud provider with our work, it would be easy to adopt it by many cloud orchestration frameworks (such as Kubernetes to schedule containers or OpenStack to schedule VM) available in the market. One would wonder how the AI scheduler would operate in a microservice-based containerised environment, where tasks have shorter duration and scheduling decisions have to be made extremely quickly. Finally, it is possible to enhance the performance and reliability by improving the learning algorithms (such as replacing the RL agent with the deep neural network to better accommodate the very large state/action space or using transfer learning to optimally adapt the model to new clouds).

References

[1] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., … & Zaharia, M. (2010). **A view of cloud computing.** *Communications of the ACM, 53*(4), 50–58.

[2] Barrett, E., Howley, E., & Duggan, J. (2013). **Applying reinforcement learning towards automating resource allocation and application scalability in the cloud.** *Concurrency and Computation: Practice and Experience, 25*(12), 1656–1674.

[3] Beloglazov, A., & Buyya, R. (2012). **Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers.** *Concurrency and Computation: Practice and Experience, 24*(13), 1397–1420.

[4] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). **Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility.** *Future Generation Computer Systems, 25*(6), 599–616.

[5] Diyan, M., Silva, B. N., & Han, K. (2020). **A multi-objective approach for optimal energy management in smart home using reinforcement learning.** *Sensors, 20*(12), 3450.

[6] Ghasemi, A., & Toroghi Haghighat, A. (2020). **A multi-objective load balancing algorithm for virtual machine placement in cloud data centers based on machine learning.** *Computing, 102*(9), 2049–2072.

[7] Gobalakrishnan, N., & Arun, C. (2018). **A new multi-objective optimal programming model for task scheduling using genetic gray wolf optimization in cloud computing.** *The Computer Journal, 61*(10), 1523–1536.

[8] Islam, S., Keung, J., Lee, K., & Liu, A. (2012). **Empirical prediction models for adaptive resource provisioning in the cloud.** *Future Generation Computer Systems, 28*(1), 155–162.

[9] Jena, R. K. (2017). **Task scheduling in cloud environment: A multi-objective ABC framework.** *Journal of Information and Optimization Sciences, 38*(1), 1–19.

[10] Kumar, J., & Singh, A. K. (2018). **Workload prediction in cloud using artificial neural network and adaptive differential evolution.** *Future Generation Computer Systems, 81*, 41–52.

[11] Mansouri, N., Hasani Zade, B. M. H., & Javidi, M. M. (2019). **Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory.** *Computers & Industrial Engineering, 130*, 597–633.

[12] Prem Jacob, T., & Pradeep, K. (2019). **A multi-objective optimal task scheduling in cloud environment using cuckoo particle swarm optimization.** *Wireless Personal Communications, 109*(1), 315–331.

[13] Qin, Y., Wang, H., Yi, S., Li, X., & Zhai, L. (2020). **An energy-aware scheduling algorithm for budget-constrained scientific workflows based on multi-objective reinforcement learning.** *The Journal of Supercomputing, 76*(1), 455–480.

[14] Sahoo, S., Sahoo, B., & Turuk, A. K. (2019). **A learning automata-based scheduling for deadline-sensitive tasks in the cloud.** *IEEE Transactions on Services Computing, 14*(6), 1662–1674.

[15] Sathya Sofia, A., & GaneshKumar, P. (2018). **Multi-objective task scheduling to minimize energy consumption and makespan of cloud computing using NSGA-II.** *Journal of Network and Systems Management, 26*(2), 463–485.

[16] Singh, S., & Chana, I. (2016). **A survey on resource scheduling in cloud computing: Issues and challenges.** *Journal of Grid Computing, 14*(2), 217–264.

[17] Sun, G., Liao, D., Zhao, D., Xu, Z., & Yu, H. (2018). **Live migration for multiple correlated virtual machines in cloud-based data centers.** *IEEE Transactions on Services Computing, 11*(2), 279–291.

[18] Yuan, H., Bi, J., & Zhou, M. (2020). **Energy-efficient and QoS-optimized adaptive task scheduling and management in clouds.** *IEEE Transactions on Automation Science and Engineering, 19*(2), 1233–1244.

[19] Zhang, L., Cheng, S., & Boutaba, R. (2010). **Cloud computing: state-of-the-art and research challenges.** *Journal of Internet Services and Applications, 1*(1), 7–18.

[20] Zhu, H., Li, M., Tang, Y., & Sun, Y. (2020). **A deep-reinforcement-learning-based optimization approach for real-time scheduling in cloud manufacturing.** *IEEE Access, 8*, 9987–9997.