Optimizing Cloud Resource Allocation Using Reinforcement Learning

Jalpa Deepak Patel, Sumisha Surendran, Abdulrauf Aremu Gidado and Patrick Amoateng
School of Computer Science, University of Windsor, ON, Canada
Email: {patel2fu, suren111, gidado, amoatenp}@uwindsor.ca

***Abstract*—The increase in digitization of processes in different spectra of life has led to increasing demand and proliferation of applications ranging from web to mobile applications (ubiquitous computing). This leads to the demand to shift computing resources to the cloud for easier, secure and timely access. Hence, one of the current dominating trends in the architecture of modern software systems is the leveraging of cloud computing infrastructures. This also calls for the need to automatically assign resources on-cloud to these computing resources as the needs arise. Thus, there is a need for an automatic decision-making model that synchronously decides computational needs on-time and effectively. Until recently, reinforcement learning models were deployed to a gaming environment. In this work, we proposed a technique for cloud resource allocation and optimization using the long-short term memory (LSTM) on reinforcement learning. We carried out our experiments using Alibaba cloud cluster data. Our experimental result shows that using LSTM to manage the previous decisions taken by a reinforcement learning agent would optimize cloud resources allocation.**

***Key words*—Cloud Resource Allocation, Cloud Resource Optimization, Reinforcement learning, LSTM**

I. Introduction

Cloud computing has emerged as the most popular computing paradigm in today's software industry. Cloud computing is widely accepted as it is used to achieve various tasks like data backup, disaster recovery, big data analytics, cloud storage, testing and development. Cloud computing with virtualization technology enables computational resources in data centers or server clusters to be shared by allocating virtual machines (VMs) in an on-demand manner. [9] One of the current dominating trends in the architecture of modern software systems is the leveraging of cloud computing infrastructures. The key benefits of this strategy include high availability, improved security and resource allocation flexibility. In certain instances, the catalyst of the adoption is the ability to adapt the number of resources used to the individual needs. There is a very convincing pledge to reduce the cost of capital. Unfortunately it requires implementing special measures. The application needs to add or remove more cores, RAM, hard drives (vertical scaling) or more virtual or physical machines (horizontal scaling) without breaking the core functionality. In certain scenarios, where the environment delivers strong and stable seasonal behaviour patterns, the configurations and resources can be estimated by human experience in advance. However, in many other cases, elasticity can only be enabled by automatic scaling, which we can define as a dynamic process that adapts software configurations and hardware resources provisioning.

Due to the time-variance of workloads, it is desirable to perform cloud resource allocation and management in an adaptive way. Most cloud service providers offered services as "pay-as-you-use" model resources need to be allocated profitably from the service providers' perspective. Although a wide variety of mechanisms are emerging to support resource allocation, these approaches do not consider performance optimization of resource allocation. The process of correctly selecting and assigning the best resources to a workload or application is cloud resource optimization. Efficiency is achieved when workload efficiency, compliance, and cost are correctly and consistently balanced in real time against the best-fit infrastructure.

Automatic decision-making methods, such as Reinforcement Learning (RL) is the evolving solution approach for making decisions centred on statistically estimating and maximizing the expected long-term utilities. Until recently, these were mainly linked to relatively simple issues, where it was easy to analyze the whole environment and the amount of possible acts was limited. The recent developments in the field makes it possible to tackle even more complex domains, such as computer games, control of robots. These strategies make it possible to learn complex behaviors through studying an environment directly and communicating with it through pre-defined actions. In certain cases, this method has made it possible to produce outcomes that exceed efficiency based on human decisions.

Apart from various research related to security, privacy and virtualization, there is ongoing research for cloud resource allocation [1]. Cloud resource allocation is a concerning task for both the providers and users of cloud services to efficiently use the services for given time period. A complete resource allocation framework in the cloud computing systems exhibits high dimensions in state and action spaces. For example, a state in the state space may be the Cartesian product of characteristics and current resource utilization level of each server (for hundreds of servers) as well as current workload level (number and characteristics of VMs for allocation) [9].

Cloud resources can be optimized based on various factors, like the number of instances used on every machine. Tasks activity also can be monitored by keeping track of several tasks per container. Computing on the cloud can also be optimized further by keeping track of usage of resources over a period [4]. This paper's principal contribution demonstrates how reinforcement learning can be used for learning utility

functions for making dynamic resource allocation (or any system reconfiguration) decisions in unknown stochastic dynamic environments with very large or continuous state spaces. We analyze how resources are used over time. We model our reinforcement algorithm with the help of the LSTM network. The implementation addresses the brute force method of reinforcement learning, where the maximum resource utilized is noted at each learning stage. To achieve this we use time series data that logs the usage of resources over a period on every container, machine and server.

II. Problem Statement

Cloud computing is widely used for various computing services today. One perfect scenario to explain would be the platform Google Colab. As we are aware that this platform helps us in executing python scripts and perform various Data Science and Machine Learning tasks on Google Cloud. Colab allows users to save data on the cloud and even compute them on the cloud by enabling users to save and share their work on Google Drive. This platform performs all the tasks that a computer helps us to achieve. Since cloud services enable computing and data storage more easily, numerous computing resources are used for the same.

As cloud enables us to perform a broad spectrum of tasks, it is crucial to make it more efficient. For the same, there is research work to address the security and network areas of the cloud. Optimizing resource allocation on cloud is one of the ongoing research tasks, and we address the same with this project. We propose to address it by implementing an LSTM based reinforcement learning model.

A reinforcement learning problem (either the traditional reinforcement learning or the emerging deep reinforcement learning model) consist of an agent and environment parameters. The formal definition of this problem (cloud resource allocation using deep reinforcement learning) can be viewed as: A system consisting of an Agent (i.e., the decision maker which needs to be trained), an environment, a finite state space S, a set of available actions that the agent can take A, and a reward function SxA→R. A continuous interaction exists between the agent and the environment in the space. At each phase k (i.e. decision epoch) the agent will make a decision denoted as k which is based on the current state of the environment denoted as sk. As the agent makes this decision, the environment receives this decision, thereby making corresponding changes by updating it state to sk+1 and then providing the agent with a reward rk for the decision. To state the above in the context of cloud resources allocation. Consider a server cluster consisting of M physical servers, these servers render D types of resources. As jobs arrives to the system, there exist a job broker which assigns jobs to servers in the clusters at arrival time.

III. Motivation

The increased use of cloud resources in our day-to-day activities was our impetus for undertaking this research project. Recently the use of cloud platforms provided by Google and Microsoft to share files has increased. Other cloud service applications like Kaggle and Google Colab enable us to perform Machine Learning related tasks. Running our Machine learning notebooks on cloud made us consider how the cloud service operates to store huge datasets and compute them online using heavy computational resources like GPU. Hence we conducted a related research survey to determine the significance of ongoing research works for optimizing cloud resource allocation.

IV. Related Work

One of the main goals of Artificial Intelligence (AI) is to develop agents which will be completely autonomous and communicate with their environments in order to learn optimal behavior, progress by trial over time [19].

The earliest major research on reinforcement learning is attributed to [16] and this opened a spectrum of other research in the field and the application of reinforcement learning to variety of scenarios such as game theory, resource allocation etc. The simple mechanism of reinforcement learning is that the learning agent observes an input pattern/state, produces an output (usually refers to as an action or control signal), and then receives a scalar known as a reward or reinforcement feedback signal from the environment denoting how good or bad its output was. However, in most practical scenarios, the reward part of the paradigm is delayed (i.e. the reward is given at the end of a long sequence of inputs and outputs). This led to an aspect of research in reinforcement learning known as temporal credit assignment problem which the learner(agent) must solve [19].

[19] proposed a novel approach which attempts to solve the temporal credit assignment aspect of the problem has remained extremely difficult to model. The method proposed is called Temporal Difference (TD) Learning Methods. The basic idea of this method is that learning is based on the difference between temporally successive predictions. TD is a class of incremental learning procedures specialized for prediction problems. i.e. TD method is driven by the error or difference between temporally successive predictions; with them, learning occurs whenever there is a change in prediction over time unlike the conventional learning methods which is driven by the error between the predicted values and the actual outcomes.

[20] applied the above paradigm proposed by [19] to neural network and was experimented with the Backgammons game. According to [2], [20] research Serves as one of the early practical research proving that as the number of hidden layers in a neural network increases, neural model performs better. Also, it shows that neural network is capable of automatic features discovery. This is one of the longstanding goals of game research since the time of Samuel i.e. [16]. [12] conducted a groundbreaking research in the field of deep reinforcement learning (i.e. applying deep learning to reinforcement learning) which was the development of algorithm that could learn to play a series of Atari 2600 video games at a superhuman level. [12]provides solutions for the instability of function

approximation techniques in RL and it is the first research to convincingly demonstrate that RL agents could be trained on raw, high-dimensional observations, solely based on a reward signal [2].

Recently, breakthrough research in deep reinforcement learning (DRL) demonstrates good results that DPL can handle large state space of complicated/complex control problems such as cloud resources allocation. For instance, recently developed systems such as Atari [11] and AlphaGo [18]. The above two systems applied convolutional neural network (CNN) (which is an example of deep neural network (DNN), apart from other examples such as recurrent neural network (RNN)) to effectively extract information from high dimensional image input and build a correlation between each RL's state-action pair (s; a) and the associated value function Q(s; a). The Q(s; a) is the expected cumulative costs that the RL's agents aims to maximize. Also, one of the recent works that specifically applied DRL to cloud resource allocation is [9]. The proposed an hierarchical framework which uses deep reinforcement learning in automating cloud resource allocation.

Non-Markovian reinforcement learning which is the scenario where the agent must learn from hidden states, an example of which is scenario where the same action leads to different state [3]. There are earlier approach towards solving the issue with reinforcement learning, this includes the fixed sized history window approach [7], learning finite state automata [5], and using Recurrent Neural network [17] or learning to set memory bits for the sequence of previous decisions/action [13].

Also, one of the recent works that specifically applied DRL to cloud resource allocation is [9]. The proposed model/algorithm uses LSTM for reinforcement learning in automating cloud resource allocation.

V. Methodology

A. Methods

To optimise resource allocation in cloud services model we propose implementation using an LSTM neural network. Previous studies show that LSTM can be used to implement reinforcement learning [3]. LSTM can be applied for a reinforcement learning task by letting the LSTM neural network learn the cloud resource environment. Later the LSTM model can predict the quantity of resources required. This way the model can record the resources and power used at different states of a cloud computing environment [3]. As the setup is non-Markovian one, the approach of using LSTM to store the resource state seems to be a good fit for the current study [23].

To achieve this task we implement a script to organize the data in a usable state. The data has different time stamps to identify the start and end of task. To learn the usage of the containers, servers and machines over time period, we need to have a uniformity in selecting the time. Hence we calculate the time period for which the instances have been in use. This helps us to shape the data in having the variables as time period, task, task status, identifiers(machine, containers and server), CPU utilized and memory utilized.

B. Algorithms

1) Reinforcement Learning: The basic RL method can learn by itself for auto configuration. The VM-Agent will gradually push the policy to converge to a better one through environmental exploitation and discovery, beginning from any initial policy. Adaptability and scalability are two remaining problems for auto configuration online.

2) Long short-term memory(LSTM): Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video). LSTMs help maintain the error that can be backpropagated through time and layers. By retaining a more constant error, they allow recurrent nets to continue to learn over many time steps usually over 1000, thereby opening a channel to link causes and effects remotely. This is one of the central challenges to machine learning and AI, since algorithms are frequently confronted by environments where reward signals are sparse and delayed, such as life itself. LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, hidden Markov models and other sequence learning methods in numerous applications [24].Gradients are values used to update a neural networks weights. The vanishing gradient problem is when the gradient shrinks as it back propagates through time. If a gradient value becomes extremely small, it doesn't contribute too much learning. However, standard RNN has the gradient vanishing or exploding problems. In order to overcome the issues, Long Short-term Memory network (LSTM) was developed and achieved superior performance (Hochreiter and Schmidhuber, 1997). In the LSTM architecture, there are three gates and a cell memory state. [22] An LSTM has a similar control flow as a recurrent neural network. It processes data passing on information as it propagates forward [6]. The differences are the operations within the LSTM's cells. [14]

3) Brute Force Algorithm for RL: Brute Force Algorithms are exactly what they sound like – straightforward methods of solving a problem that rely on sheer computing power and trying every possibility rather than advanced techniques to improve efficiency. The time complexity of a generic brute force algorithm is O(n*m) . Which means if we were to search for a string of "n" characters in a string of "m" characters using brute force, it would take us n * m tries.

Brute force implementation for LSTM consists learning the quantity of each cloud resource and return the optimal value of resource utilized at each stage in the network. At every stage

of the task the LSTM model remembers the optimal use of resources used and updates it. All the relevant optimal usage of resources can be later used to predict for a new task given the list of available resources [25].

VI. Experiments and Discusssion

A. Datasets

We conducted our experiments using Alibaba cloud cluster data. The cluster data Was collected in 2018 in time intervals of 8 days. The data set includes information of 4000 machines [21]. The data is divided in 6 files/schema as described in Tables 1,2,3,4,5,6. The description of the relevant information in the schema of the data is as follows:
i) Meta and event information of machines
ii) Resource usage of each machine
iii) Meta and event information of containers
iv) Resource usage of each container

MACHINE META	
Field	Description
machine_id	uid of machine
time_stamp	time stamp, in second
failure_domain_1	one level of container failure domain
failure_domain_2	another level of container failure domain
cpu_num	number of cpu on a machine
mem_size	normalized memory size. [0, 100]
status	status of a machine

Table 1 : Machine meta information

MACHINE USAGE	
Field	Description
machine_id	uid of machine
time_stamp	time stamp, in seconds
cpu_util_percent	Percentage of CPU utilized in range of [0, 100]
mem_util_percent	Percentage of memory utilized in range of [0, 100]
mem_gps	normalized memory bandwidth in range of [0, 100]
mkpi	cache miss per thousand instruction
net_in	normalized in coming network traffic in range [0, 100]
net_out	normalized out going network traffic in range [0, 100]
disk_io_percent	Percentage of disk utilized in range of [0, 100], abnormal values are of -1 or 101

Table 2 : Machine usage information

CONTAINER META	
Field	Description
container_id	uid of a container
machine_id	uid of machine
time_stamp	time stamp, in second
app_du	Containers with same app_du belong to same application group
status	Status of machine
cpu_request	Requests made to CPU - 100 is 10 million
cpu_limit	Limit of CPU - 100 is 10 million
mem_size	Normalized memory in range [0,100]

Table 3 : Container meta information

CONTAINER USAGE	
Field	Description
container_id	uid of container
machine_id	uid of container's host machine
time_stamp	time stamp, in second
cpu_util_percent	Percentage of CPU utilized in range [0, 100]
mem_util_percent	Percentage of memory utilized in range [0, 100]
mem_gps	normalized memory bandwidth in range [0, 100]
mkpi	cache miss per thousand instruction
net_in	normalized in coming network traffic measured in range [0, 100]
net_out	normalized out going network traffic measured in range [0, 100]
disk_io_percent	[0, 100], abnormal values are of -1 or 101

Table 4 : Container usage information

BATCH TASK	
Field	Description
task_name	Task name - unique within a job
instance_num	Number of instances
job_name	Job name
task_type	Type of task - generic
status	Status of task
start_time	Start time of the task
end_time	End time of the task
plan_cpu	Quantity of CPU required by the task
plan_mem	Normalized memory size in [0,100]

Table 5 : Information related to tasks - workload considered is batch workload

BATCH INSTANCE	
Field	Description
instance_name	instance name of the instance
task_name	name of task to which the instance belong
job_name	name of job to which the instance belong
task_type	task type
status	instance status
start_time	start time of the instance
end_time	end time of the instance
machine_id	uid of host machine of the instance
seq_no	sequence number of this instance
total_seq_no	total sequence number of this instance
cpu_avg	average cpu used by the instance, 100 is 1 core
cpu_max	average memory used by the instance (normalized)
mem_avg	max cpu used by the instance, 100 is 1 core
mem_max	max memory used by the instance (normalized, [0, 100])

Table 6 : Batch workload instance information

The data set was designed to address the inefficiency of utilisation of resources. [8] The data related to machine utilization as shown in Table 1 and 2 captures the events and resources utilized by a machine. As the data set offers information for both online and offline computing [21], the online task information is present in Tables 4 and 6 and batch workloads data schema is depicted in Tables 5 and 6.

The batch workloads are from a DAG as there are multiple instances which execute different computing techniques and services [10] [15].

B. Experimental Framework

We build a 3 layer model where the input layer and the first hidden layer is the LSTM layer. We use sigmoid as activation function for the two hidden layers and softmax as the output activation function. To build the LSTM model we add a LSTM layer to the Sequential model. The hidden layers help us in remembering the optimal resources allocated for a job. As we want the number of optimal resources as the output the output dimension is 1. The second hidden layer and output layer are fully connected Dense layers.

To compile the model we use adam optimizer. Adam is a optimisation algorithm used for optimising adaptive learning rate. Since we want the model to learn how resources are allocated over a series of time intervals, adam would be a good fit for such adaptive learning. To compute the loss we use Mean squared error. We use 30% of training data for validation and run our experiments for 300 epochs.

We update the weights and save the best weight for every epoch which thus helps us in getting the optimal weight for the input. To further conduct the experiments we split the data in 70-30 ratio. We use the 30% data to test it and have a comparative study for it as shown in the Figure.

To implement the above mentioned algorithms we use Keras library. The dataset requires preprocessing and data cleaning to convert it to an usable format. We use a script to compute the time period of the data given the end time and start time for which the server was used. This helps in determining the time period required for various tasks.

C. Performance Measures

The model loss is computed and as shown in the Fig. 1 the loss decreases over epochs and then stays constant.

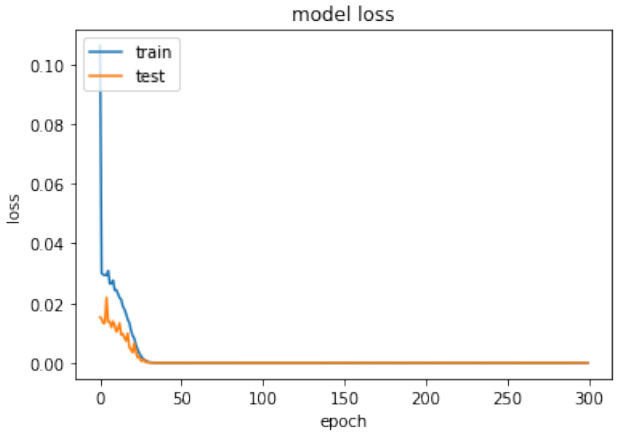

Fig. 1. Model Loss Plot

To compare how the model has learnt to optimise resource we also plot a graph that shows the number of instances assigned for a given time interval. As seen in Fig. 2 below the red dashed line is what the algorithm has predicted and from that we see that it fails to depict the usage of exceptionally high number of instances.

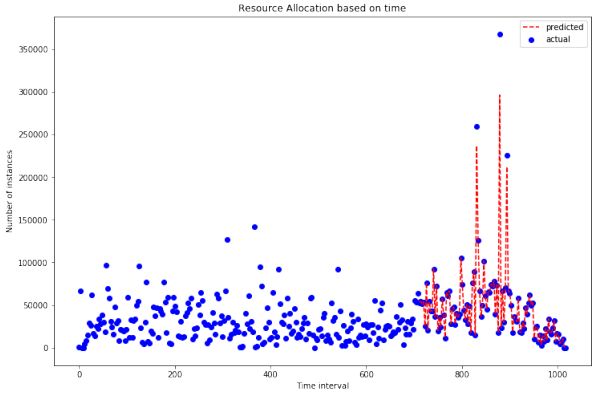

Fig. 2. Prediction of Test data compared to the orignal data

D. Results

We need to determine what variables are the best in order to proceed with optimising the resource allocation. We being

to do so by understanding the usage of online instances over time as shown in the figure below.

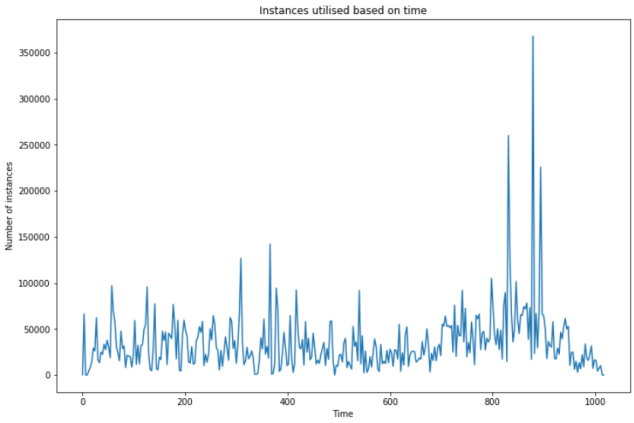

Fig. 3. Instances utilized based on time

As shown in Fig. 4, we have computed CPU utilisation based on status over a given period of time. To achieve the task we group by the status and plot the graph. The graph also depicts the maximum, minimum and average use of CPU. This usage of CPU as a resource is depicted for the status Failed, Interrupted, Running and Terminated. The status for

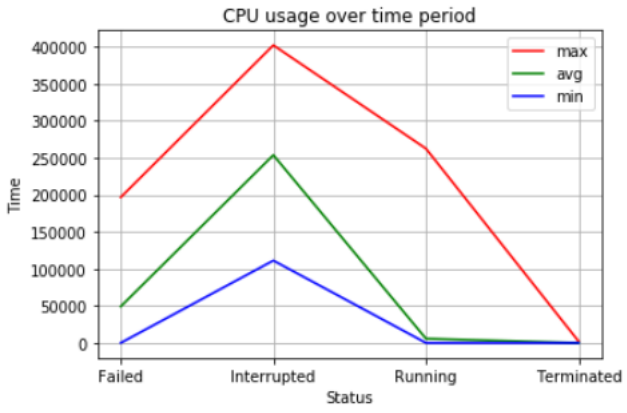

Fig. 4. CPU usage based on status for containers

terminated is always 0 to show that the task is completed and is not using CPU at that point. Also, the maximum utilisation of CPU is done at the status ”Interrupted” and the least utilisation apart from ”Terminated” is for status ”Running”.

Fig. 5 depicts the CPU usage of an instance given the containers to perform the tasks. We can see that we have maximum containers used for 100 CPU tasks.

We also analyse the memory utilized by server over a given time period as depicted in Fig. 6. The tasks with less time period vary highly in memory utilisation but the amount of memory utilisation else remains constant for tasks with a longer time period.

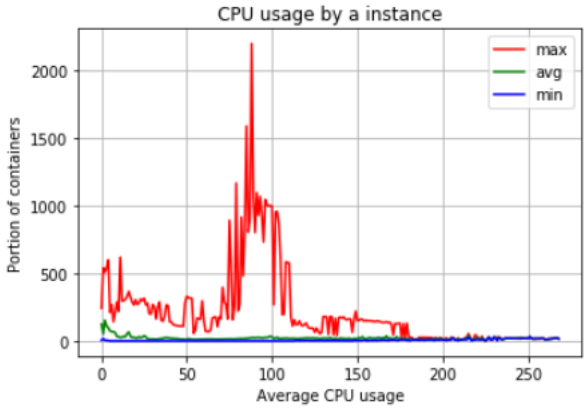

Fig. 5. CPU usage for containers

Fig. 6. Memory utilised by server

Further we also compare the CPU and memory utilisation of Machines at different time intervals as shown Fig. 7 and Fig. 8.

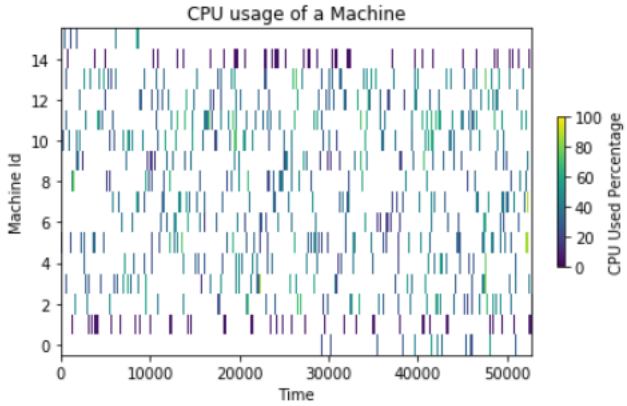

Fig. 7. CPU usage of Machine

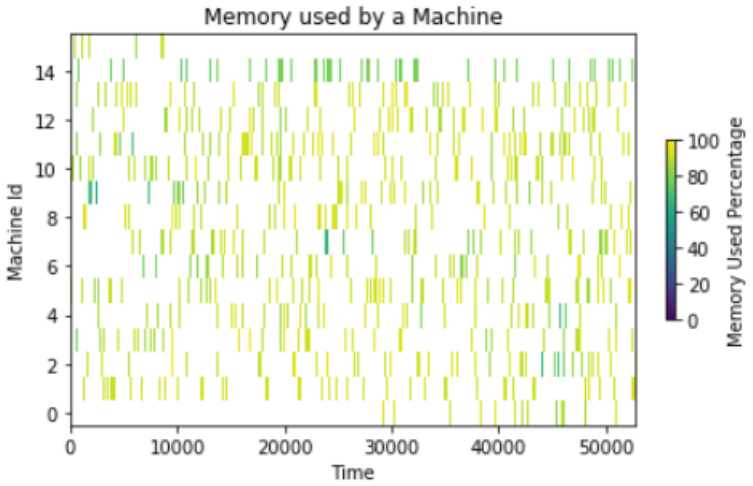

Fig. 8. Memory usage of Machine

VII. Conclusion and Future Work

The attempt to optimize resource allocation for a cloud computing environment helps the cloud providers for their business and to reach out to more users with resources and provide a fast-working space.

We achieved the same by implementing the brute force reinforcement learning algorithm using LSTM. We test the model on the given data to compare how accurately it has learnt to predict the optimum value of resource required. From the results we can say that our model is able to predict accurate allocation of resources to most of the server instances.

We believe our work serves as a foundation for further developments by implementing a Deep Reinforcement learning algorithm. We can also work in the near future to include more variables like power management and latency.

VIII. Acknowledgment

We want to thank Dr. Alioune Ngom for providing us with all the resources and reference materials to help us understand the Machine learning concepts. We are thankful to the professor for guiding us and supporting throughout to pursue this research topic.

REFERENCES

[1] Göran Adamson, Lihui Wang, Magnus Holm, and Philip Moore. Cloud manufacturing–a critical review of recent development and future trends. *International Journal of Computer Integrated Manufacturing*, 30(4-5):347–380, 2017.

[2] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[3] Bram Bakker. Reinforcement learning with long short-term memory. *Advances in neural information processing systems*, 14:1475–1482, 2001.

[4] Yue Cheng, Ali Anwar, and Xuejing Duan. Analyzing alibaba's co-located datacenter workloads. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 292–297. IEEE, 2018.

[5] Lonnie Chrisman. Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In *AAAI*, volume 1992, pages 183–188. Citeseer, 1992.

[6] Hongzimao. hongzimao/deeprm.

[7] Long-Ji Lin and Tom M Mitchell. Reinforcement learning with hidden states. *From animals to animats*, 2:271–280, 1993.

[8] Lioncruise. lioncruise/alibaba-clusterdata-analysis.

[9] Ning Liu, Zhe Li, Jielong Xu, Zhiyuan Xu, Sheng Lin, Qinru Qiu, Jian Tang, and Yanzhi Wang. A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 372–382. IEEE, 2017.

[10] Chengzhi Lu, Kejiang Ye, Guoyao Xu, Cheng-Zhong Xu, and Tongxin Bai. Imbalance in the cloud: An analysis on alibaba cluster trace. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 2884–2892. IEEE, 2017.

[11] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[12] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[13] Leonid Peshkin, Nicolas Meuleau, and Leslie Kaelbling. Learning policies with external memory. *arXiv preprint cs/0103003*, 2001.

[14] Michael Phi. Illustrated guide to lstm's and gru's: A step by step explanation, Jun 2020.

[15] PriyadarshiniDas. Priyadarshinidas/workload-prediction-of-alibaba-cluster-dataset.

[16] Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.

[17] Jürgen Schmidhuber. Networks adjusting networks. In *Proceedings of" Distributed Adaptive Neural Information Processing"*, pages 197–208, 1990.

[18] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[19] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[20] Gerald Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995.

[21] Alibaba Trace. https://github.com/alibaba/clusterdata. 2018.

[22] Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615, 2016.

[23] Daan Wierstra, Alexander Foerster, Jan Peters, and Juergen Schmidhuber. Solving deep memory pomdps with recurrent policy gradients. In *International Conference on Artificial Neural Networks*, pages 697–706. Springer, 2007.

[24] Wikipedia contributors. Long short-term memory — Wikipedia, the free encyclopedia, 2020. [Online; accessed 1-May-2020].

[25] Wikipedia contributors. Reinforcement learning — Wikipedia, the free encyclopedia, 2020. [Online; accessed 21-December-2020].