Adaptive Data Center Activation with User Request Prediction

Min Sang Yoon, *Student Member, IEEE,* Ahmed E. Kamal, *Fellow, IEEE* and Zhengyuan Zhu

Abstract—The problem of energy saving in data centers has recently attracted significant interest within the research community, and the adaptive data center activation model has emerged as a promising technique to save energy. However, this model has not integrated adaptive activation of switches and hosts in data centers because of its complexity. This paper proposes an adaptive data center activation model that consolidates adaptive activation of switches and hosts simultaneously integrated with a statistical request prediction algorithm. The learning algorithm predicts user requests in a predetermined interval by using a cyclic window learning algorithm. Then the data center activates an optimal number of switches and hosts in order to minimize power consumption that is based on prediction. We designed an adaptive data center activation model by using a cognitive cycle composed of three steps: data collection, prediction, and activation. In the request prediction step, the prediction algorithm forecasts a Poisson distribution parameter λ in every predetermined interval by using Maximum Likelihood Estimation (MLE) and Local Linear Regression (LLR) methods. Then, adaptive activation of the data center is implemented with the predicted parameter in every interval. The adaptive activation model is formulated as a Mixed Integer Linear Programming (MILP) model. Switches and hosts are modeled as M/M/1 and M/M/c queues. In order to minimize power consumption of data centers, the model minimizes the number of activated switches, hosts, and memory modules while guaranteeing Quality of Service (QoS). Since the problem is NP-hard, we use the Simulated Annealing algorithm to solve the model. We employ Google cluster trace data to simulate our prediction model. Then, the predicted data is employed to test the adaptive activation model and observe energy saving rate in every interval. In the experiment, we could observe that the adaptive activation model saves 30 to 50% of energy compared to the full operation state of data centers in practical operating conditions of data centers.

Index Terms—Data center, operational power minimization, Fat-tree data center, traffic prediction, machine learning, queuing theory, Mixed Integer Linear Programming.

1 Introduction

Increase in data traffic with mobile devices and expansion of cloud computing environment will demand continuous growth in data center utilization and energy consumption. Data centers consumed 91 billion kilowatt-hours of electricity in 2013, which is equivalent to energy from 34 large coal fire power plants. This huge amount of energy consumption in data centers has attracted interest from many researchers, and many energy saving models are proposed. Among many proposals, the adaptive data center activation model has emerged as the most promising energy-saving strategies that controls the activation of hosts and network switches to save energy. However, the complexity of the problem has prevented researchers from developing cooperative activation strategy of switches and servers simultaneously. As a result, only a few studies have considered the joint adaptive operation of data center network and server layers [1][2]. However, the latency and service quality constraints are not considered in the previous study of [1].

With the goal of standardizing energy-aware network management, the European Telecommunications Standards Institute (ETSI) published ETSI Standard (ES) 203 237, 'the Green Abstraction Layer (GAL)', which is a framework to manage energy efficient networks of the future [3], [4], [5]. The GAL is an architecture interface that provides information exchange between the power managed data plane and the control plane. The GAL supports discovery, provisioning, and monitoring of functionalities in the networks, and regulates QoS of the network. The overall network power consumption through adaptation of Network-wide Control Policies (NCP) and Local Control Policy (LCP). We follow the GAL scheme in this paper. The NCP is applied by the controller in the data center and the LCP controls each server and switch state, such as ports, memory, and node states, according to the control policy of controller.

In this paper, we propose an advanced interactive adaptive data center activation model that controls switch and host layers' activation simultaneously while satisfying Quality of Services (QoS) requirements. This model will be implemented as an adaptive data center activation model with a three step cognitive cycle: data collection, requests prediction, and data center activation. An accurate prediction of user requests should support the implementation of the adaptive activation model properly. According to [6], in 2013 50% of the energy is wasted in data centers due to lack of awareness of the traffic. This data shows how the accurate prediction of the requests will be increasingly important in the future with an indisputable increase in energy consumption. Also, the accurate request prediction has an effect on the performance of data centers. Excessive deactivation of computing or network devices for energy saving will cause a bottleneck and delay in handling requests.

For the prediction step, we collect the history of requests arriving at the data center and observe the histogram. Based on the histogram analysis, we decide on a probability distribution model to fit to the collected data. Then, parameters of the probability distribution model are inferred using the Maximum Likelihood Estimation (MLE) method and saved for the prediction. After enough data is collected for prediction, the prediction model estimates parameters of the probability distribution with Local Linear Regression (LLR) by using a cyclic window approach. Parameters of the probability distribution are time-dependent on data, which means the parameters change with time. Therefore, the

parameters have different patterns during every interval, but they exhibit the same pattern during the same period in every day or every week.

Arriving traffic is not constant. The traffic usually increases during daytime and decreases during night. Therefore, powering all components of data center will waste unnecessary energy when there is low traffic. Deciding on the number of activated switches and hosts is important and depends on arriving job rates. If a data center is running insufficient hosts compared to requests, it will not be able to serve all requests from users. Even if there are enough hosts, if the data center is not operating enough switches in order to save energy, it will cause latency in transferring data. Most of data centers architectures are constructed with a tree architecture, so if the data center does not activate necessary switches for connecting core switches to hosts, it will fail to transfer requests to hosts.

We develop an optimal energy consumption model for the Fat-tree based data center depending on the incoming rate λ while guaranteeing QoS and connectivity.

For optimal energy consumption of data centers, we control the activation of switches, server nodes, switch ports, and memory modules. The server power consumption is estimated as 40 to 55% and switch layer power consumption is estimated as 10 to 20% from the total data center energy consumption [1]. Thus, deactivating unneeded hosts and switches is expected to save a significant portion of energy. Turning off unneeded switch ports will result additional energy savings. Since we deactivate servers and switches, it is possible to turn off ports connected to inactive nodes. We also dynamically activate memory of switches and hosts because the memory also consumes a significant portion of energy. According to [7], DRAM consumes 25% of total power in data centers.

Scalability of data centers is an important issue. As traffic requests to data centers increase, many data center operators face the need for increasing data centers sizes. Most of data centers are designed homogeneously, which means that it is not easy to increase the capacity of data center resources. Our model employs a heterogeneous Fat-tree architecture of data centers that is controlled with a Software Defined Network (SDN) controller.

We model an optimal SDN controlled data center using a Mixed Integer Linear Program (MILP). Switches are modeled as M/M/1 queue and servers are modeled as M/M/c queue. This model will minimize the operational power by deciding active switches, hosts, and operating memory. The problem is NP-hard. Therefore, Simulated Annealing is used to find a near optimal solution of the problem within reasonable computation time.

Google cluster-trace data, which is real measurements of usage requests in Google cluster, is employed to test our prediction model and predict λ for the adaptive activation model.

To summarize, the contributions of this paper are: i) A request prediction algorithm is developed by using a cyclic window learning approach. The algorithm predicts the probability distribution parameters of requests during every predetermined period. ii) An optimal adaptive data center activation model is modeled as MILP. The data center operates the minimum number of switches and hosts while guaranteeing the QoS requests submitted to the data center. iii) We designed an adaptive data center activation model which activates the data center in every predetermined period based on predicted traffic.

The rest of the paper is organized as follows. Previous work is reviewed in Section II. The system model for adaptive activation model is introduced in Section III. We introduce the requests prediction algorithm in Section IV and adaptive data center activation model in Section V. In Section VI, the Simulated Annealing algorithm is presented to solve the NP-hard optimization problem. Simulation results of the prediction model and adaptive activation model are presented in Section VII and we end this paper with conclusions in Section VIII.

2 Related work

2.1 Traffic prediction in cloud systems

Request prediction in cloud and data center has been studied by many researchers for the adaptive scaling of systems.

Akindele A. Bankole *et al*. employ the machine learning for predictive resource provisioning in the clouds [8]. Their prediction model is achieved with machine learning techniques including: Neural Network, Linear Regression, and Support Vector Machine. They predict the CPU utilization, response time, and throughput based on collected data from virtual machines web servers and database servers. The prediction model generates prediction values in every given minute with machine learning techniques and measure error rate with Mean Absolute Percentage Error (MAPE) and Root Mean Squared Error (RMSE). However, the prediction model did not show a high prediction accuracy, and their results show 24% prediction error in predicting the CPU utilization at a certain point of time and a 21% error in the response time prediction.

Sedeka Islam *et al*. present more advanced machine learning techniques for predicting resource usage in [9]. Error Correction Neural Network (ECNN) and the Linear Regression techniques are employed for the prediction. Then they included a sliding window method to reflect the current state of the system. They generated prediction values based on window sizes and evaluated them with MAPE, PRED(25), RMSE, and R^2prediction accuracy. The CPU utilization data is collected from the Amazon EC2 cloud through the TPC-W benchmark and prediction values are generated with the ECNN and Linear Regression method. The prediction values of CPU utilization have around 19% error rate without the sliding window and have a minimum 18.6% error rate when they employ the sliding window.

Many statistical approaches are also applied to traffic prediction in clouds. Bruno Lopes Dalmazo *et al*. propose a traffic prediction approach based on a statistical model where observations are weighted with Poisson distribution inside a moving window [10]. They consider the past information by means of a sliding window of size λ and this window is applied by weighting the past observations according to the Poisson distribution with parameter λ. Dropbox trace data is employed for testing their prediction model and Normalized Mean Square Error (NMSE) evaluation method is utilized for error measurement. The prediction model could achieve NMSE values between 0.044 and 0.112. The prediction is ideal when NMSE value is equal

to zero and worse when it is greater than one. This approach achieves a reasonably accurate prediction.

They also propose a traffic prediction model with a dynamic window approach [11]. The sliding window size is changed based on variance of the previous window size. A small variance indicates the predicted data is close to the actual data while a high variance means the predicted data is spread out from the mean. Therefore, they update the window size in every prediction interval by considering the size of the variance in the previous prediction. The prediction accuracy is improved from 7.28% to 495.51% compared to the previous statistical model.

2.2 Cloud system modeling using queuing theory

Jordi Vilaplana *et al* studied the computer service QoS in cloud computing based on queuing theory [12]. Cloud platforms are modeled by an open Jackson network which consists of multiple nodes, where each node corresponds to a queue in which the service rates are node dependent. They modeled multi-server system with Entering Server (ES) and Processing Server (PS). ESs are modeled by M/M/1 queues, which works for the load balancer, PSs are modeled as M/M/m queues. However, all queues are assumed to have the same service rate so heterogeneity of systems is not considered. By using the queuing system analysis, they analyzed the response time of the global cloud architecture with different parameters.

Wei-Hua Bai *et al* investigated heterogeneous modern data centers and the service process in data centers by using queuing theory [13]. They built a complex queuing model composed of the master server and computing nodes. The master node works as the main scheduler to allocate resources for tasks to dispatch a node to execute tasks, which is modeled as M/M/1/K queue. Computing nodes which are a multi-core server are modeled as an M/M/c queue because each multicore execution server can parallel-process multiple tasks. By using queuing theory analysis, they investigate system metrics such as the mean response time, the mean waiting time, and other important performance indicators. Although they investigated the system performance of heterogeneous data centers efficiently, they did not include the switch node in their consideration. Since the switch architecture and connections with servers significantly affect data centers performance, switches are also required to be considered when we analyze the data center performance.

Junwei Cao *et al* studied the problem of optimal multi-server configuration for profit maximization in cloud computing environments [14]. They included the amount of service, the workload of an application environment, the configuration of multiserver system, the service level agreement, the satisfaction of a consumer, the quality of service, the penalty of a low-quality service, the cost of renting, the cost of energy consumption, and the service provider's profit margin in their profit modeling and maximized their profit through optimal multiserver configuration. They modeled a multiserver system as an M/M/m queuing model so that the optimization problem can be formulated and solved analytically. For more general analysis, they derived the density function of the waiting time of a newly arriving service request and the expected service charge to a service request is also calculated.

2.3 Power saving in cloud systems

Energy saving in data centers is considered in many aspects because of its importance. Junwei Cao *et al*. propose optimal power allocation and load distribution in a heterogeneous data center environment [15]. They model a multicore server processor as a queuing system with multiple servers and prove optimal server setting for two different core speed models, idle speed model and the constant speed model.

Kuangyu Zheng *et al*. propose joint power optimization of data center network and servers with correlation analysis [1]. They propose a power optimization strategy that leverages workload correlation analysis to jointly minimize total power consumption of servers and data center network. After they analyze the correlation between Virtual Machines (VMs), they consolidate VMs that are not correlated with each other onto the same physical servers to save server power consumption and consolidate network traffic to save data center network power. Through this approach, they could achieve up to 51.6% of energy saving in their testbed. This paper has a similarity with our proposal in terms of joint optimization of data center networks and servers. However, the authors consolidate VMs based on correlation analysis between VMs, and then activates traffic flow between VMS that are correlated, which means the activation of servers and switches are not jointly considered. Also, the traffic flow is consolidated between correlated VMs. Thus, this strategy cannot guarantee connectivity between physical servers due to data center network layer deactivation.

This paper has differences from all previous works in the following aspects: Our prediction model predicts the probability distribution parameter in a certain period instead of predicting quantified value of requests. Therefore, we achieve a more flexible prediction by deciding the desired probability of the predicted distribution.

The dynamic data center activation model integrates the dynamic operation of switch layer and server layer while guaranteeing the performance of the data center network. Since we jointly decide activated switches and hosts, we could save more energy compared to other dynamic activation models which determine operating hosts and switches independently.

3 System Model

3.1 Distribution parameter adaptive data center activation model

The cognitive cycle of the adaptive data center activation, which is shown in Fig. 1, is composed of three phases: data collection, request prediction, and data center activation. For the first phase, the control plane collects the number of incoming tasks and refines collected data to utilize it for the prediction. The collected data is saved in the prediction data set and the prediction model employs it to forecast the future requests of users by using a cyclic window learning algorithm. According to the predicted requests, the control plane solves the adaptive activation problem and activates the optimal set of switches and hosts, and keeps unnecessary components in the idle state. This cycle is repeated periodically in every predetermined period. For example, if we set the duration to 30 minutes, the system repeats one cycle every 30 minutes. Thus, the system can reconfigure the

data center every 30 minutes in order to reduce the waste of resources.

In adaptive data center activation, the system setup time is not a negligible factor of system performance. The inactive devices are kept in standby or low power idle states as described in [3], [4]. If the devices are put in the low power idle state for power saving purpose, a longer wake up time will be required when the device is activated. By using the cognitive cycle of the system, the wake up delay of devices can be minimized through the predictive system activation. The prediction process is executed in advance of the system activation. Based on the prediction result, the controller can put newly activated devices in the standby state. So the system wake up time will be minimized according to the prediction and activation schemes.

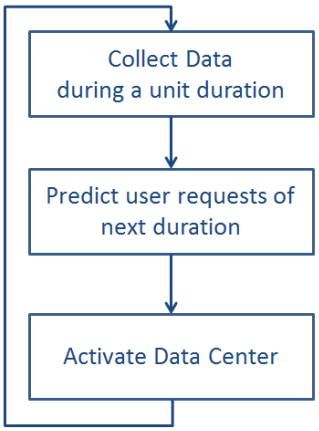

Figure 1: Cognitive cycle of system

3.2 Fat-Tree Data Center

There are many data center architectures: Fat-tree, VL2, Bcube, and so on. In this paper, we adopt the Fat-tree architecture, the most widely used data center architecture by many cloud service providers [16], [17].

Fat-tree is composed of three switch layers (core, aggregation, and Top of Rack (ToR) layer) and a host layer as shown in Fig. 2. All switches have the same number of ports. If we assume the number of ports in each switch is k, we have k core switches and k Points of Delivery (POD). Core switches are grouped into $k/2$ groups. Since a core switch has k ports, each core switch is connected to all PODs. For example, in Fig. 2, the first core switch in each group is connected to the first aggregation switch in each POD and the second core switch in each group is connected to the second aggregation switch in each POD. Each POD has $k/2$ aggregation switches and $k/2$ ToR switches. Aggregation, ToR switches, and their interconnection network forms a complete bipartite graph. Since a ToR switch uses $k/2$ ports for connecting aggregation switches, they can serve $k/2$ hosts each. Therefore, the data center has a total of $k^3/4$ hosts.

The Fat-tree architecture is widely used because of its scalability and interconnection capability [16]. Due to the scalable characteristic of Fat-tree data center, it is able to configure large size data center with lower Capital Expenditure (CAPEX). Also, interconnections between hosts in the same data centers are easy with full bandwidth.

In the Fat-tree architecture, we cannot deactivate necessary switches to operate required hosts because of its connectivity. For example, the first ToR switch cannot be turned off when we want to allocate jobs to the first host in the first POD. In our model, the core switch should be able to reach any host using three links and a host should be able to reach a core switch using three links as well. Therefore, we will turn on and off switches without violation of this rule to guarantee connectivity.

3.3 SDN based data center

For comprehensive management of network management, we employ an SDN based data center model. The SDN based data center is composed of the control plane and the data plane. The control plane plays the role of managing the network and deciding routing paths. The data plane just forwards data according to the values in the forwarding table.

The control plane should have access to data plane switches and needs to receive system status information from all data plane switches. All switches implement the data plane functionality since they have to forward data. In addition, the core switch layer implements the control plane.

Since the workload distribution and resource allocation are determined by the control plane switches, the data center can activate only a limited number of switches and hosts by allocating workloads optimally.

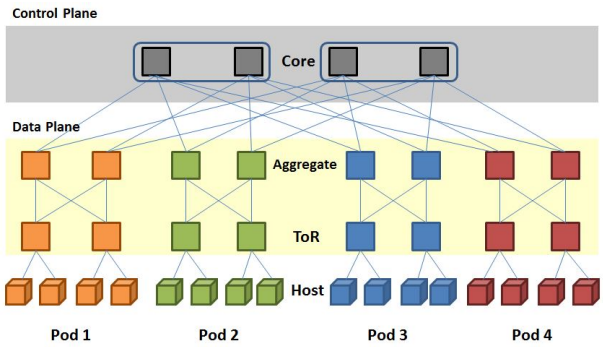

Figure 2: SDN fat-tree data center model (k=4)

4 Request Prediction Algorithm

The prediction model estimates parameters of the probability distribution of future user requests in every predetermined period. For the traffic analysis, one week traffic data is selected from Google Cluster trace data in [18]. Since the data is selected starting from a random point of the whole data, we do not know the exact data and time of the traffic measurement. However, we could find that user requests have obvious patterns and the patterns are repeated periodically, as shown in the Fig. 3.

The hourly pattern shows high peaks for the first 4 hours and the last 10 hours. So, we can assume daytime starts around the 10th hour from the measurement and end at the 4th hour in Fig. 3. If we observe the daily pattern, the first two days and the last two days show higher requests. In the same way, we can assume weekdays start on the 4th day of the measurement by assuming requests increases during

weekdays rather than the weekend. The daily pattern analysis of task arrivals presents obvious patterns of the incoming requests. Therefore, the cyclic data collection should be an effective approach for the prediction. Based on our analysis, the prediction model adopts periodic history data at the same time point in order to make predictions about future data center loads.

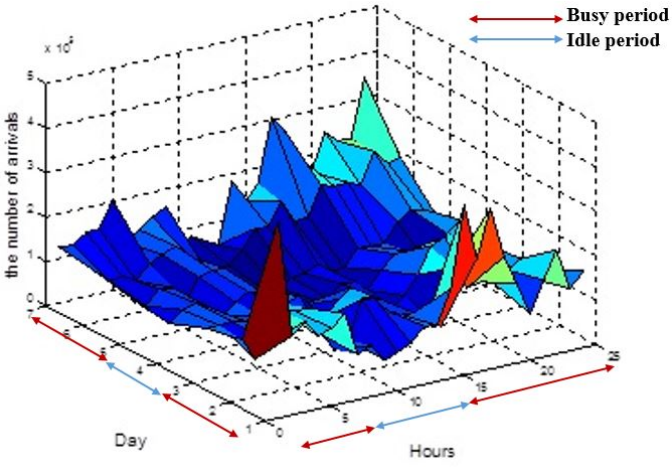

Figure 3: Prediction data set

We introduce three time scales for the prediction periods: Pattern Period (PP), Target Period (TP) and Utilization Period (UP). The PP is a cyclic interval that exhibits pattern repetition. The TP is a unit duration for which we want to make a prediction. The UP is a cyclic window that we use for predicting the activities in TP. In the example, we predict the request distribution during a certain Monday by assuming the same pattern is repeated every week. Then, we can set the TP to a day and the PP to a week because we assume the pattern of a day is repeated every week. If we only use the past Mondays' data for the prediction, the UP becomes one day.

Since patterns are repeated on every PP, any time duration for the prediction corresponds to a certain TP on the PP. Therefore, we can predict the request distribution during any time interval by correlating them to a certain TP on the PP. For precise prediction, data is accumulated for several PPs. Although patterns of the traffic distributions will be similar at the same time point in every PP, the traffic amount will be different. In other words, we can say the distribution of the traffic shows similar pattern in every Monday, but we cannot ensure that the amounts of traffic will be the same. Therefore, the prediction model can achieve higher prediction accuracy by accumulating data during several PPs.

To implement prediction, the data set saves the past data in an $m \times l$ matrix. m represents the number of TPs on the PP and l denotes the number of PPs we accumulate. In Fig. 4, each vertical block corresponds to saved parameters of the probability distribution in each TP during a PP. We start to stack the data from the first block of the first iteration. If the data set is filled until the TP_m's block, which is the last TP, we move to the second iteration and stack the data from the first block of the second iteration, which means we have saved data during a PP. When the matrix is full, the data set goes back to the first block of the first iteration

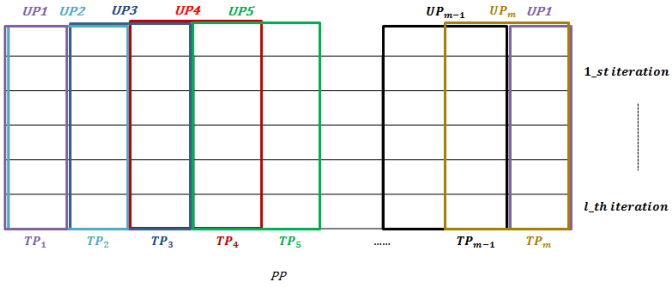

Figure 4: Prediction data set

and replaces the old data to reflect the tendency of recent requests.

Any time duration that we want to predict the traffic distribution can be related to a certain TP on the matrix. In order to make a prediction, the UP data is employed. In Fig. 4, we can see that distribution parameters of TP_m can be predicted by using UP_m. We can set the size of UP depending on how many previous TPs will affect the state of the current TP. For example, UP is set to two days and TP is one day, e.g., previous Sunday and Monday's history parameters are utilized to predict next Monday's request distribution.

In this paper, the prediction model forecasts the number of task arrivals during each target period. In the first step of prediction, the prediction model constructs a histogram of user requests in every target period to observe distributions of requests. Then, it fits a probability distribution to the distribution of requests. When the probability distribution kernel is decided for fitting, we adopt MLE in order to obtain parameters of the probability distribution in every observed period and the parameters are saved on the data set. After the data set accumulates enough data for prediction, it is able to predict parameters of a following TP. LLR will be employed to predict parameters of the future requests. LLR is one of the kernel smoother techniques for estimating a real valued function, when no parametric model for this function is known. Since the data set is updated and solved in every predetermined period, the prediction model is a dynamical operation problem. The detailed process of the algorithm is included in [18].

5 Adaptive Data Center Activation Model

We formulate an adaptive data center activation problem as an MILP. Our objective is to minimize the activation power, port power, and memory power consumption in the data center while guaranteeing QoS. This model decides the active state of switches and hosts by using binary variables and determines load distribution over switches and hosts with continuous variables. Based on allocated workload in switches and servers, controllers can determine how many memory modules need to be on. We assume that jobs arrive at the data center according to a Poisson distribution with rate λ. We define the following variables:

- C_{ij}: working state of i_{th} group j_{th} core switch; binary variable.
- A_{ij}: working state of i_{th} POD j_{th} aggregation switch; binary variable.
- T_{ij}: working state of i_{th} POD j_{th} ToR switch; binary variable.
- $H_{ij\_ToR_{mn}}$: working state of i_{th} POD j_{th} host connected to ToR_{mn} ToR switch; binary variable.
- $M_{C_{ij}}$, $M_{A_{ij}}$, $M_{T_{ij}}$, $M_{H_{ij\_ToR_{mn}}}$: The number of operating memory

modules in core, aggregate, ToR switches, and hosts; integer variables.
• $\lambda_{C_{ij}}, \lambda_{A_{ij}}, \lambda_{T_{ij}}, \lambda_{H_{ij\_ToR_{mn}}}$: Job arrival rate to core, aggregate, ToR switches and hosts; continuous variable.
• $\lambda_{C_{ij\_l}}$: Job arrival rate to i_{th} POD j_{th} core switch l_{th} port; continuous variable.
• $\lambda_{A_{ij\_l}}$: Job arrival rate to i_{th} POD j_{th} aggregation switch l_{th} port; continuous variable.
• $\lambda_{T_{ij\_l}}$: Job arrival rate to i_{th} POD j_{th} ToR switch l_{th} port; continuous variable.
• $P^{static}_{C_{ij}}, P^{static}_{A_{ij}}, P^{static}_{T_{ij}}, P^{static}_{H_{ij\_ToR_{mn}}}$: Static power consumption of core, aggregate, ToR switches and hosts during a unit time; constant.
• $P^{dynamic}_{C_{ij}}, P^{dynamic}_{A_{ij}}, P^{dynamic}_{T_{ij}}, P^{dynamic}_{H_{ij\_ToR_{mn}}}$: Dynamic power consumption of core, aggregate, ToR switches and hosts on the full utilization state during a unit time; constants.
• $P^{port}_{C_{ij}}, P^{port}_{A_{ij}}, P^{port}_{T_{ij}}$: Power consumption of each port in core, aggregate, and ToR switche per job; constants.
• $P^{memory}_{C_{ij}}, P^{memory}_{A_{ij}}, P^{memory}_{T_{ij}}, P^{memory}_{H_{ij\_ToR_{mn}}}$: Memory power consumption of core, aggregate, ToR switches and hosts during a unit time; constants.
• $\mu_{C_{ij}}, \mu_{A_{ij}}, \mu_{T_{ij}}, \mu_{H_{ij\_ToR_{mn}}}$: service rate of core switch; constants.
• λ: Job arrival rate to data center; constant.

5.1 Objective Function

$$Minimize \quad p^{activation} + p^{port} + p^{memory} \tag{1}$$

The objective function includes activation power, port power, and memory power consumption during a unit time. Activation power is the power consumption necessary for operating switches or hosts. Port power is the power consumption for transmitting a job through the port. We assume that ports are in the idle state when they do not transmit jobs and do not consume power in the idle state. So port power is calculated depending on the number of jobs they transmit. Memory power is determined depending on how many memory modules are powered. If we turn on more memory modules than those required in a switch or a host, this will waste energy. Therefore, we propose to turn on the appropriate number of required memory modules based on statistical estimation.

5.1.1 Activation Power

$$\begin{aligned} p^{activation} = & \sum_{i=1}^{k/2}\sum_{j=1}^{k/2} C_{ij}(P^{static}_{C_{ij}} + P^{dynamic}_{C_{ij}}\frac{\lambda_{C_{ij}}}{\mu_{C_{ij}}}) \\ & + \sum_{i=1}^{k}\sum_{j=1}^{k/2} A_{ij}(P^{static}_{A_{ij}} + P^{dynamic}_{A_{ij}}\frac{\lambda_{A_{ij}}}{\mu_{A_{ij}}}) \\ & + \sum_{i=1}^{k}\sum_{j=1}^{k/2} T_{ij}(P^{static}_{T_{ij}} + P^{dynamic}_{T_{ij}}\frac{\lambda_{T_{ij}}}{\mu_{T_{ij}}}) \\ & + \sum_{i=1,m=1}^{k}\sum_{j=1}^{k/2}\sum_{n=1}^{k/2} H_{ij\_ToR_{mn}}(P^{static}_{H_{ij\_ToR_{mn}}} \\ & \qquad + P^{dynamic}_{H_{ij\_ToR_{mn}}}\frac{\lambda_{H_{ij\_ToR_{mn}}}}{\mu_{H_{ij\_ToR_{mn}}}}) \end{aligned} \tag{2}$$

We can calculate the activation power by considering the static and dynamic power of switches and hosts. The static power is a constant power required for activating switches and hosts. The static power can be calculated by multiplying the binary variables that represent the state of each switch and host by the static power consumption constant of each switch and host. The dynamic power increases proportionally to the utilization rate of switch and host. Thus, the dynamic power can be computed by multiplying the utilization, $\frac{\lambda}{\mu}$, and dynamic power consumption constant of each switch and host. The dynamic power expression includes a non-linear term, corresponding to the multiplication of the binary variable and continuous variable. This non-linear term can be linearized by using a simple transformation introduced in Section 5.2.1.

5.1.2 Port Power

$$\begin{aligned} p^{port} = & \sum_{i=1}^{k/2}\sum_{j=1}^{k/2}\sum_{l=1}^{k} \lambda_{C_{ij\_l}} P^{port}_{C_{ij}} + \sum_{i=1}^{k}\sum_{j=1}^{k/2}\sum_{l=1}^{k/2} \lambda_{A_{ij\_l}} P^{port}_{A_{ij}} \\ & + \sum_{i=1}^{k}\sum_{j=1}^{k/2}\sum_{l=1}^{k/2} \lambda_{T_{ij\_l}} P^{port}_{T_{ij}} \end{aligned} \tag{3}$$

Port power should only be considered in switches. The problem decides how each switch will distribute arriving jobs through which port. Therefore, we can measure the port power consumption by multiplying the amount of jobs passing through each port and the power consumption per job of each port.

5.1.3 Memory Power

$$\begin{aligned} p^{memory} = & \sum_{i=1}^{k/2}\sum_{j=1}^{k/2} M_{C_{ij}} P^{memory}_{C_{ij}} \\ & + \sum_{i=1}^{k}\sum_{j=1}^{k/2} M_{A_{ij}} P^{memory}_{A_{ij}} + \sum_{i=1}^{k}\sum_{j=1}^{k/2} M_{T_{ij}} P^{memory}_{T_{ij}} \\ & + \sum_{i=1,m=1}^{k}\sum_{j=1}^{k/2}\sum_{n=1}^{k/2} M_{H_{ij\_T_{mn}}} P^{memory}_{H_{ij\_ToR_{mn}}} \end{aligned} \tag{4}$$

We will estimate the number of jobs in each switch and host memory by using a queuing model. Then we can decide how many memory modules we need to operate based on the estimated number of jobs in each switch and host by queuing theory. After we decide the number of working memory modules, we can evaluate memory power consumption by multiplying the number of memory modules and memory power consumption of each memory module in each switch and host.

5.2 Load Distribution Constraints

Workload should be distributed rationally in the data center. Since we are proposing to turn off parts of the switches and hosts, we have to decide exactly which port will transfer jobs to the lower layers in order to prevent job losses. For example, if the core switch transfers jobs to an inactive aggregation switch, we will not be able to allocate those jobs to hosts. Thus, load distribution constraints decide how we distribute the workload to low layer switches through which ports.

5.2.1 Load Distribution in Each Layer

The summation of workload in each layer should be equivalent to λ, the total arriving workload to the data center. The inter-POD traffic does not need to go through the core switch layer in traditional data centers. However, all traffic goes through the core switch layer in SDN data center architecture. Therefore, the summation of traffic in each layer will be the same. The purpose of this constraint is to allocate the workload to working switches and hosts while guaranteeing that we forward all requests to hosts.

$$\sum_{i=1}^{k/2}\sum_{j=1}^{k/2} C_{ij}\lambda_{C_{ij}} = \lambda \tag{5}$$

$$\sum_{i=1}^{k}\sum_{j=1}^{k/2} A_{ij}\lambda_{A_{ij}} = \lambda \tag{6}$$

$$\sum_{i=1}^{k}\sum_{j=1}^{k/2} T_{ij}\lambda_{T_{ij}} = \lambda \tag{7}$$

$$\sum_{i=1,m=1}^{k}\sum_{j=1}^{k/2}\sum_{n=1}^{k/2} H_{ij\_ToR_{mn}}\lambda_{H_{ij\_ToR_{mn}}} = \lambda \tag{8}$$

Equations (5)-(8) are multiplications of binary variables and continuous variables, so they are non-linear equation. Therefore we linearize those non-linear constraints. When x is a binary variable and y is a continuous variable that is greater than or equal to zero, we can linearize their product by replacing it by a continuous variable t with the constraints shown below.

$$0 \le t \le \max(y)\cdot x \tag{9}$$

$$y - (1-x)\cdot\max(y) \le t \le y \tag{10}$$

The maximum of continuous variables, $\lambda_{C_{ij}}$, $\lambda_{A_{ij}}$, $\lambda_{T_{ij}}$, and $\lambda_{H_{ij\_T_{mn}}}$ is λ because it is the maximum workload that can be allocated to each switch and host, and is the summation of all workloads in the same layer. Therefore, we can linearize non-linear constraints (5), (6), (7), and (8) with this linearization technique.

5.2.2 Link Distribution

As mentioned before, it is important to determine which port to use for transferring jobs. If we use a port connected to an inactive switch or host for transmitting that job, we will lose the request.

$$\lambda_{C_{ij}} = \sum_{l=1}^{k}\lambda_{C_{ij\_l}}, \forall i,j \tag{11}$$

$$\lambda_{A_{ij}} = \sum_{l=1}^{k}\lambda_{C_{lj\_i}}, \forall i,j \tag{12}$$

$$\lambda_{A_{ij}} = \sum_{l=1}^{k/2}\lambda_{A_{ij\_l}}, \forall i,j \tag{13}$$

$$\lambda_{T_{ij}} = \sum_{l=1}^{k/2}\lambda_{A_{il\_j}}, \forall i,j \tag{14}$$

$$\lambda_{T_{ij}} = \sum_{l=1}^{k/2}\lambda_{T_{ij\_l}}, \forall i,j \tag{15}$$

$$\lambda_{T_{ij\_l}} = \lambda_{H_{il\_ToR_{ij}}}, \forall i,j,ToR_{ij} \tag{16}$$

Equations (11), (13), and (15) regulate the distribution from each switch to lower layers. Arriving job requests to each switch should be equivalent to the summation of workloads to go out through their ports.

Equations (12), (14), and (16) decide how arriving jobs are handled by each switch. Summation of the number of jobs going into switches should be equal to the summation of arriving jobs from upper layer through connected port.

Through this constraint, we can decide which port will transfer how many jobs in each link. We can obtain an optimal workload distribution strategy with these constraints.

5.2.3 Distribution Restriction

Jobs cannot be distributed to inactive switches and hosts. Constraints (5)-(8) regulate the summation of jobs in each layer but it can allocate jobs to deactivated switches because the summation of workload is still the same if the switch is turned off. Therefore, we need constraints that prohibit workloads from being allocated to inactive switches and hosts, and these are as follows:

$$\lambda_{C_{ij}} \le \lambda\cdot C_{ij}, \forall i,j \tag{17}$$

$$\lambda_{A_{ij}} \le \lambda\cdot A_{ij}, \forall i,j \tag{18}$$

$$\lambda_{T_{ij}} \le \lambda\cdot T_{ij}, \forall i,j \tag{19}$$

$$\lambda_{H_{ij\_ToR_{mn}}} \le \lambda\cdot H_{ij\_ToR_{mn}}, \forall i,j,m,n \tag{20}$$

These constraints force the workload allocated to a switch or a host to zero if the binary variable corresponding to the switch or host is zero.

5.3 Performance Constraints

In order to guarantee the QoS of data centers, latency constraints will be applied to each switch and host. Switches are assumed to be M/M/1 queues and hosts are M/M/c queues, where c is equivalent to the number of cores of the host using the queuing model, we can obtain the distribution of residual time in the queue. Residual time of a job in queue corresponds to how long each job will stay in switches and hosts. Thus, we will use the residual time distribution to obtain the latency constraint.

5.3.1 Latency in a Switch

Switches are modeled as M/M/1 queues. Using the Cumulative Density Function (CDF) of the residual time in M/M/1 queue to obtain the residual time equation in M/M/1 queue, we have

$$F_R(t_{residual}) = 1 - e^{-(\mu-\lambda_{switch})t_{residual}} \tag{21}$$

Equation (21) represents the probability that the residual time in the queue will be less than $t_{residual}$. Therefore, we can obtain the maximum residual time in the queue by letting $F_R(t_{residual}) = \alpha$, where α is close to 1 and $t_{residual}$ satisfies.

$$1 - e^{-(\mu-\lambda_{switch})t_{residual}} = \alpha \tag{22}$$

Slightly manipulating (22) we have the upper bound on the residual time in the queue as,

$$t_{residual} = -\frac{\ln(1-\alpha)}{(\mu - \lambda_{switch})} \tag{23}$$

Since the residual time cannot exceed the latency constraint, Latency, we have the following constraint.

$$-\frac{\ln(1-\alpha)}{(\mu - \lambda_{switch})} \leq Latency \tag{24}$$

which can be expressed in the following linear form, where

$$-\ln(1-\alpha) \leq Latency \cdot (\mu - \lambda_{switch}) \tag{25}$$

By using (25), we can obtain the latency constraint for all switches.

5.3.2 Latency in a Host

We use a strategy similar to the above to obtain the delay in hosts. A host is modeled as an M/M/c queue, and the CDF of the residual time in an M/M/c queue is [19]:

$$F_R(t_{residual}) = 1 - \frac{c}{c-\rho} \cdot p_c \cdot e^{-(c\mu-\lambda)t_{residual}} \tag{26}$$

p_c is the probability that there are c jobs in a queue, c represents the number of cores in our model, which is usually a small number and ρ represents $\frac{\lambda}{\mu}$. For the most cases of practical interest, all cores will be active and we can assume p_c to be equal to 1 for the purpose of simplifying the problem. Therefore,

$$F_R(t_{residual}) = 1 - \frac{c}{c-\rho} \cdot e^{-(c\mu-\lambda)t_{residual}} \tag{27}$$

Again, if the residual time satisfies (27) with probability α, with $\alpha \approx 1$, then

$$1 - \frac{c}{c-\rho} \cdot e^{-(c\mu-\lambda)t_{residual}} = \alpha \tag{28}$$

After simple manipulations of (28) we obtain

$$t_{residual} = \frac{\ln \frac{(1-\alpha)(1-\rho)}{c}}{-(c\mu - \lambda)} \tag{29}$$

Since the residual time in the queue should be less than the latency constraint, we can get constraint below

$$\frac{\ln \frac{(1-\alpha)(1-\rho)}{c}}{-(c\mu - \lambda)} \leq Latency \tag{30}$$

Which can also be written as

$$-\ln(1-\alpha) - \ln(1-\rho) + \ln c \leq Latency(c\mu - \lambda) \tag{31}$$

Equation (31) is non-linear because it includes the non-linear term, $-ln(1-\rho)$. We will linearize (31) by using the piecewise approximation technique used in [24]. Since ρ has a limited domain between 0 to 1, then we can linearize $-ln(1-\rho)$ by calculating a linear function which lower bounds $-ln(1-\rho)$ in each of a number of sub-domains of ρ. We divide ρ to three sub-domains: [0, 0.75], [0.75, 0.95], and [0.95, 1]. Then, we obtain a linear function that fits each sub-domain, $f_1(\rho)$, $f_2(\rho)$,and $f_3(\rho)$, which lower bounds $-ln(1-\rho)$.

Through approximation, we could achieve three linear functions in each sub-domain, $f_1(\rho) = 1.72\rho$, $f_2(\rho) = 7.1679\rho - 4.1698$, $f_3(\rho) = 40.2359\rho - 35.6308$. By replacing $-\ln(1-\rho)$ in (31) by $\max_y f_y(\rho)$, y represents a linear function in the y sub-domain, and we obtain the linear constraint below.

$$\max_y f_y(\rho) - \ln(1-\alpha) + \ln c \leq Latency \cdot (c\mu_{H_{ij\_ToR_{mn}}} - \lambda_{H_{ij\_ToR_{mn}}}), \forall i, j, m, n, y \tag{32}$$

5.3.3 Service Rate

For the stability of the system, the summation of service rates of each layer should be greater than the total arrival rates jobs. That is,

$$\sum_{i=1}^{k/2}\sum_{j=1}^{k/2} \mu_{C_{ij}} C_{ij} \geq \lambda \tag{33}$$

$$\sum_{i=1}^{k}\sum_{j=1}^{k/2} \mu_{A_{ij}} A_{ij} \geq \lambda \tag{34}$$

$$\sum_{i=1}^{k}\sum_{j=1}^{k/2} \mu_{T_{ij}} T_{ij} \geq \lambda \tag{35}$$

$$\sum_{i=1,m=1}^{k}\sum_{j=1}^{k/2}\sum_{n=1}^{k/2} \mu_{H_{ij\_ToR_{mn}}} H_{ij\_ToR_{mn}} \geq \lambda \tag{36}$$

5.4 Memory Constraints

Main memory consumes almost 25% of total energy consumption in the data center. Therefore, minimizing the number of working memory modules will contribute to saving energy consumption in data centers. We decide the number of working memory modules based on the probability that an arriving job is rejected. We assume the job is rejected only when there is not enough memory in the queue.

5.4.1 Memory in Switch

Switches are modeled as M/M/1 queue. If we assume the $n_q + 1$ job is rejected in the queue, the probability of losing a job is equal to probability of n_q jobs in the queue. So we can calculate the probability of loss like below:

$$p(loss) = p_{n_q} \tag{37}$$

We set the probability that an arriving job is successfully queued to $\alpha \approx 1$.

$$p(queuing) = \alpha \tag{38}$$

Then we can get the equation below:

$$p(loss) = 1 - p(queuing) = 1 - \alpha \tag{39}$$

$p(loss)$ is when there are n_q jobs in queue. So we can calculate the probability of loss like below:

$$p(loss) = 1 - \sum_{i=0}^{n_q-1} p_i \tag{40}$$

If we substitute $p(loss)$ in (39) with (40) to obtain $p_i = \rho^i(1-\rho)$ in the M/M/1 queue.

$$\sum_{i=0}^{n_q-1} p_i = \alpha \tag{41}$$

So from (41) we can obtain,

$$1 - \rho^{n_q - 1} = \alpha \tag{42}$$

If we take the log on both sides, we can achieve:

$$n_{switch} = \frac{\ln(1-\alpha)}{\ln(\rho)} + 1 \tag{43}$$

Equation (43) represents the number of existing jobs in a switch with very high probability α. Based on the number of jobs in the queue, we can obtain the number of memory modules we need. If we assume the expected size of a job is J and memory module size is MS, we can establish the following memory constraint.

$$n_{switch} \cdot J \leq M_{switch} \cdot MS \tag{44}$$

Since all switches are modeled as M/M/1 queues, we can apply this constraint to all switches and decide the number of working memory modules depending on their allocated workloads.

5.4.2 Memory in Host

Hosts are modeled as M/M/c queues. We can use the same approach used with the switch, but instead using the steady state probability of the M/M/c queue. As mentioned before, we assume hosts are always busy when they are turned on. Since c represents the number of cores in a host, we can consider there will be always more than c jobs in the host. Therefore we can use the following equation to obtain steady state probability:

$$\sum_{i=c}^{\infty} p_i \approx 1 \tag{45}$$

The steady state of probability in M/M/c queue is given for $i \geq c$,

$$p_i = \frac{c^c \rho^n}{c!} p_0 \tag{46}$$

where $\rho = \frac{\lambda_{host}}{c\mu}$.

The initial state probability p_0 can be found by substituting p_i in (45) by (46).

$$\sum_{i=c}^{\infty} \frac{c^c \rho^n}{c!} p_0 \approx 1 \tag{47}$$
$$\frac{c^c \rho^c}{c!(1-\rho)} p_0 \approx 1$$

So, the initial probability p_0 is given by,

$$p_0 \approx \frac{c!(1-\rho)}{c^c \rho^c} \tag{48}$$

By using (48), we can use steady state probabilities by substituting p_0 in (46).

If we substitute (46) in (41), we can obtain the number of jobs in hosts in terms of utilization.

$$n_{host} = \frac{\ln(1-\alpha)}{\ln(\rho)} + c \tag{49}$$

By using (49), we can achieve the required number of memory modules that can accommodate jobs in hosts with the following constraint.

$$n_{host} \cdot J \leq M_{host} \cdot MS \tag{50}$$

However, (44) and (50) are non-linear. Similar to the above piecewise linear approximation, we will divide them into three sub-domains: [0, ρ_1^*], [ρ_1^*, ρ_2^*], and [ρ_2^*, 1]. Then we could get a linear function in each sub-domain. The approximation function is different depending on the service rate of switches and hosts. So, if we define the functions that obtain the number of jobs in each switch and host,n_{switch} and n_{host} as $f_n(\rho)$, we can calculate sub-linear functions as:

$$f_{n1}(\rho) = \frac{f_n(\rho_1^*)}{\rho_1^*}\rho \tag{51}$$
$$f_{n2}(\rho) = \frac{f_n(\rho_2^*) - f_n(\rho_1^*)}{\rho_2^* - \rho_1^*}(\rho - \rho_2^*) + f_n(\rho_2^*)$$
$$f_{n3}(\rho) = \frac{\mu - f_n(\rho_2^*)}{1 - \rho_2^*}\rho + \frac{f_n(\rho_2^*)}{1 - \rho_2^*}$$

μ is the service rate of each switch and host. Since we approximate the number of jobs by a linear function, we can linearize the constraints (44) and (50) like below in each switch and host:

$$\max_y(f_{ny}(\rho_{C_{ij}})) \leq \frac{M_{host} \cdot MS}{J} \tag{52}$$

y represents the number of approximation functions and M is the number of memory module that are powered, which is an integer variable.

5.5 Connectivity Constraints

The connectivity constraints are applied to the problem to protect the connectivity of the data center. If we deactivate necessary switches for connecting to working hosts, it will cause the disconnection between the core layer and hosts. Therefore, we define the connectivity constraints that maintain the architecture of data centers.

$$\frac{\sum_{j=1}^{k/2} T_{ij}}{k} \leq \sum_{j=1}^{k/2} A_{ij}, \forall i \tag{53}$$

ToR switches and aggregation switches in the same POD have a complete bipartite connection. Therefore, ToR switch can be connected to the core switch layer even if there is only one working aggregation switch in the same POD. (53) activates at least one aggregation switch if ToR switches are activated in the same POD.

$$\frac{k}{2} \sum_{j=1}^{k/2} H_{ij\_ToR_{mn}} \leq T_{mn}, \forall i \tag{54}$$

If a host is turned on and is connected to T_{mn} ToR switch, T_{mn} should be turned on as well. Since T_{mn} is a binary variable, the right hand side should not exceed 1. So, we divide the number of turned on hosts by $2/k$ in order to make the upper bound of summation equal to 1.

6 Simulated Annealing Algorithm

The optimization problem includes binary variables and integer variables in the objective function and constraints. The binary variables decide the state of switches and hosts and the integer variables decide the number of working memory modules in each switch and host. As we include integer variables in the problem, this optimization becomes

NP-hard. When we consider a small size data center, the optimization problem can be solved in a reasonable time. However, data centers usually include a large number of servers and switches. In such cases, it will take a long time to solve. Therefore, we will use the Simulated Annealing algorithm which, is a popular randomized heuristic algorithm that can find a near optimal solution in a reasonable time.

Algorithm 1 Simulated Annealing Algorithm

Require: $S_{H_{ij\_ToR_{mn}}}, S_{T_{ij}}, S_{A_{ij}}, S_{C_{ij}}, \lambda, \beta, iteration$
Ensure: $P_{minimum}, \lambda_{H_{ij\_ToR_{mn}}}, \lambda_{T_{ij}}, \lambda_{A_{ij}}, \lambda_{C_{ij}}$
1: We generate initial allocation to hosts $\lambda_{H_{ij\_ToR_{mn}}}$ depends on service rate of hosts
2: Decide ToR, aggregation and core switches based on allocation into hosts.
3: $\lambda_{T_{ij}}, \lambda_{A_{ij}}, \lambda_{C_{ij}}$=Switch($\lambda_{h_{ij\_ToR_{mn}}}$)
4: $P_{minimum}$=$Cal\_power(\lambda_{h_{ij\_ToR_{mn}}}, \lambda_{T_{ij}}, \lambda_{A_{ij}}, \lambda_{C_{ij}})$
5: **while** t≤ iteration **do**
6: Search neighbor allocation $\lambda^{'}_{h_{ij\_ToR_{mn}}}$
7: $\lambda^{'}_{T_{ij}}, \lambda^{'}_{A_{ij}}, \lambda^{'}_{C_{ij}}$=Switch($\lambda^{'}_{H_{ij\_ToR_{mn}}}$)
8: $P_{candidate} = Cal\_power(\lambda^{'}_{H_{ij\_ToR_{mn}}}, \lambda^{'}_{T_{ij}}, \lambda^{'}_{A_{ij}}, \lambda^{'}_{C_{ij}})$
9: r←rand()
10: **if** $P_{candidate} < P_{minimum}$ or $e^{\frac{p_{minimum}-p_{candidate}}{\beta}} > r$ **then**
11: $\lambda_{H_{ij\_ToR_{mn}}} \leftarrow \lambda_{H_{ij\_ToR_{mn}}}$
12: $\lambda_{T_{ij}} \leftarrow \lambda^{'}_{T_{ij}}$
13: $\lambda_{A_{ij}} \leftarrow \lambda^{'}_{A_{ij}}$
14: $\lambda_{C_{ij}} \leftarrow \lambda^{'}_{C_{ij}}$
15: $P_{minimum} \leftarrow P_{candidate}$
16: t++, $\beta \leftarrow \beta \cdot \alpha$
17: **else**
18: t++, $\beta \leftarrow \beta \cdot \alpha$
19: **end if**
20: **end while**
21: OUTPUT: $P_{minimum}, \lambda_{H_{ij\_ToR_{mn}}}, \lambda_{T_{ij}}, \lambda_{A_{ij}}, \lambda_{C_{ij}}$

The algorithm requires input parameters: the service rates (jobs/sec) of hosts and switches ($S_{H_{ij\_ToR_{mn}}}, S_{T_{ij}}, S_{A_{ij}}, S_{C_{ij}}$), total incoming jobs into the data center (λ), parameter β that has a value between 0 to 1, and the maximum number of iterations, $iteration$. We generate the initial allocation of λ into hosts while guaranteeing the latency constraint and capacity (line 1). The latency goes to infinity when the allocation to hosts is close to service rates. Therefore, we decide the allocation rates that do not violate the latency constraint and let the allocation of jobs not exceed that ratio. After we decide the initial allocation to hosts, we can determine which switches will be needed to operate the data center properly. So then the Switch() function finds which switches are required and how many jobs can be allocated to switches based on the assignment result of hosts (line 3). After we decide the allocation of jobs over the entire data center, we calculate the power consumption of the data center based on the result of allocations (line 4). Then we set that solution and the power as a candidate solution. We start finding neighbor solutions in lines 5 to 20. We find the neighbor solution of job allocation into hosts (line 6). In the same method, we can find which switches are required to be activated to reach working hosts. So, we turn on the required switches by using Switch() function (line 7). Then, we can calculate the power consumption of a neighbor solution in line 8. If the power consumption of the neighbor solution is less than the power consumption of candidate solution, we replace the candidate solution with the neighbor solution. We also replace candidate solutions with neighbor solutions with some probability to avoid isolation of solution. A random number r is generated between 0 and 1. If $e^{p_{minimum}-p_{candidate}/\beta}$ is greater than random number r, we replace candidate solution with the neighbor solution even if power consumption of neighbor solution is greater than power consumption of candidate solution (line 10 - line 15). In every iteration, β is multiplied by α, which has a value less than 1, to decrease the β in every iteration (line 16 and line 18). The algorithm repeatedly finds neighbor solutions for t iterations. Then, the candidate solution will become a near optimal solution of the problem.

7 Experiments

7.1 Google Cluster Data

The experiments reported in this section are implemented using Google cluster-usage traces data [20]. Google cluster is a set of computing resources composed of thousands of machines. A job is composed of several tasks which can be processed separately. So each task will be a unit processing. We consider the number of task arrivals. Each task has a time-stamp which represents when the task arrives at the cluster. Therefore, the distribution of the number of task arrivals can be observed by using the time-stamp. The cluster starts measurement 600 seconds after the system is operated and has accumulated data for one month approximately. We select a random point to collect data for the prediction model. One week data is sampled as a training data set for the prediction modeling and the following week data is employed to test the accuracy of the prediction model.

7.2 Request Prediction

7.2.1 Time Dependent Parameter Prediction

Predicted parameters are obtained from the accumulated data within blocks of 30 minutes each. Based on our observation of Google cluster data, the data is fitted to a Poisson distribution with parameter λ which is evaluated using the data in every predetermined period, values of λ are then saved in prediction the data set. Parameter values are stacked in data set for the several periods. Therefore, it is possible to estimate the parameter values in future periods more accurately as we accumulate more data. Following parameter values are achieved by implementing LLR over corresponding UP. For example, if we implement LLR over UP including the $1st$ to the $10th$ TPs, the last point value of LLR function becomes the prediction value of the eleventh TP. The prediction values change depending on the UP we set or bandwidth value that we use for LLR.

Fig. 5 represents the prediction of Poisson distribution parameter λ of arriving tasks during a week. Since we set the TP to 30 minutes, we have 336 target periods during the week. Blue points represent parameter values of training data set in each TP and green points are parameter values of test data set in each TP. Solid lines represent parameter prediction values for different values of bandwidth. We set the UP to 25 hours in Fig. 5, which means that prediction value is obtained based on the last $25th$ hours data. Parameter λ is equivalent to mean the number of arrivals during 30 minutes. We can observe that the graph has a regularly repeated pattern. It has seven high peaks in the

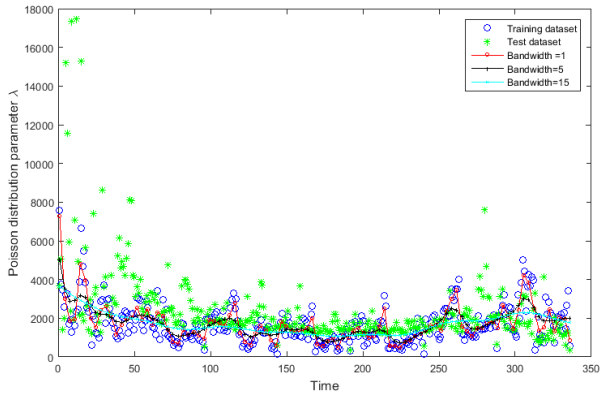

Figure 5: Poisson distribution parameter λ estimation of arrival tasks

graph, which means similar patterns repeated during the week.

7.2.2 Error Assessment

In order to quantify the accuracy of prediction, we measure Mean Absolute Percentage Error (MAPE) between prediction model and test data set. MAPE is a measurement of the accuracy of prediction for constructing fitted time series value. MAPE expresses error rate as the percentage.

$$MAPE = \frac{1}{n}\sum_{j=1}^{n}\frac{|P_j - T_j|}{T_j} \tag{55}$$

P_j is the predicted value of the actual target value T_j. MAPE value is equal to zero when the prediction model is a perfect fit to target value and increased when the prediction is not properly fitted to target values.

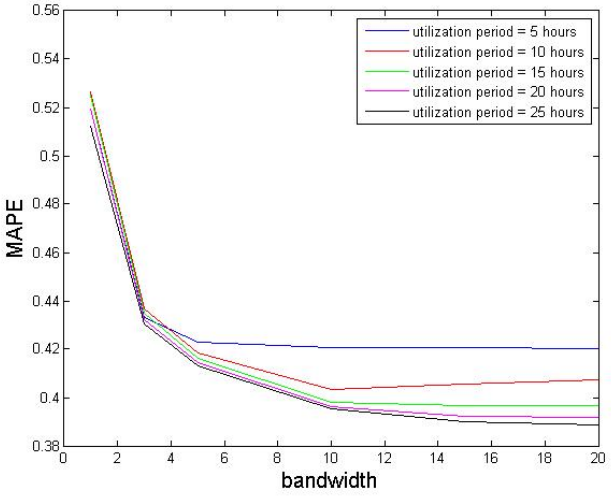

Figure 6: MAPE measurement of arrival tasks prediction

Fig. 6 is the MAPE measurement graph of the Poisson distribution parameter λ. Poisson distribution parameter λ has MAPE value between 38.84% and 52.49% in Fig. 6. The prediction model could achieve higher accuracy with longer UP, which is equivalent to the window size, because increasing the UP means employing more data from history for prediction. However, the large UP requires the more complex computation. In other words, proper selection of utilization period is required to satisfy both prediction accuracy and calculation time. Choosing the best bandwidth value is also an important issue in order to reduce prediction error. Too small bandwidth causes very spiky estimates while large bandwidth leads to over-smoothing. If data values are spread widely, smaller bandwidth will not result in higher prediction accuracy. In contrast, the regression achieves higher accuracy with small bandwidth if data values are compacted. Fig. 6 shows higher accuracy with the increase in bandwidth because parameter data points are more spread as we can see in Fig. 5.

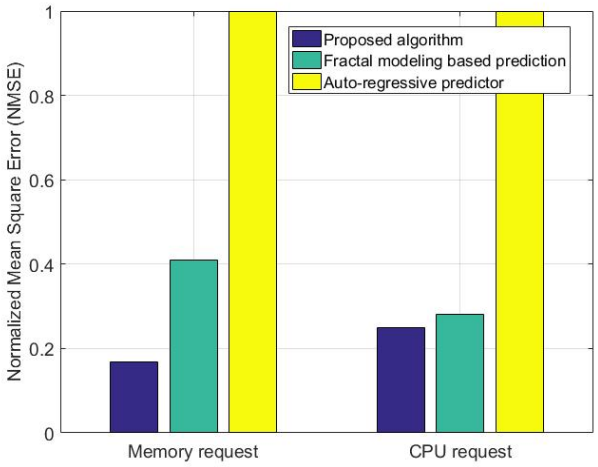

Figure 7: NMSE comparison

The proposed prediction algorithm is compared with the fractal differential equation modeling based prediction method proposed in [21]. The Mean Square Error (MSE) of the proposed algorithm and comparison prediction model is measured for CPU and memory requests and normalized by baseline prediction algorithm. The auto-regressive predictor is used for baseline model by employing 16 previous time slot values, which is equivalent to using 16 UP in the proposed algorithm. In memory request prediction, the proposed model achieves 25% reduction MSE compare to the fractional modeling based predictor and 84% reduced MSE than the auto-regressive predictor. In CPU request prediction, the proposed algorithm shows a slightly enhanced prediction accuracy than fractal modeling based predictor, 3% reduced MSE but it achieves 75% MSE reduction compared to the baseline prediction model.

7.3 Adaptive Data Center Activation

The optimization problem is solved using CPLEX for small data centers and also using the Simulated Annealing algorithm implemented in C. We consider heterogeneous environment data centers having different service rates of hosts. However, all switches have the same performance in each layer. We set the parameters of the model as shown in Table 1.

Power consumption parameters of switches are estimated with practical power consumption of data center switch, HP Altoline 6712 Switch Series [22]. Since core

Table 1: Value of key parameters

Variables	Values
$P^{static}_{C_{ij}}$, $P^{static}_{A_{ij}}$, $P^{static}_{T_{ij}}$	100W, 50W, 50W
$P^{dynamic}_{C_{ij}}$, $P^{dynamic}_{A_{ij}}$, $P^{dynamic}_{T_{ij}}$	300W, 150W, 150W
$P^{static}_{H_{ij\_ToR_{mn}}}$	[30, 50, 70, 90]W
$P^{dynamic}_{H_{ij\_ToR_{mn}}}$	[100, 150, 200, 250]W
$P^{port}_{C_{ij}}$, $P^{port}_{A_{ij}}$, $P^{port}_{T_{ij}}$	0.0005W/job
$P^{memory}_{C_{ij}}$, $P^{memory}_{A_{ij}}$, $P^{memory}_{T_{ij}}$, $P^{memory}_{H_{ij\_ToR_{mn}}}$	25W

switches implement path calculation and management as control plane, they consume more energy than aggregation and ToR switches in the data plane. Switches in the data plane are set to consume half energy consumed by core switches. Hosts have different activation power parameters proportional to their service rate. Our model includes four types of hosts, and therefore hosts have four levels of activation power consumption depending on their service rates with high performance hosts consuming more power. Identical hosts are allocated to the same POD. Thus, hosts heterogeneity will be across PODs.

The size of requests are randomly generated by normal distribution with μ=20MB. All switches have two 512MB memory modules and hosts have two 1GB memory modules. The latency constraint of each switch is set to $1\mu s$ based on the reference switch data sheet and the latency constraint of hosts is set to $1ms$ to satisfy the QoS of data center.

The service rate of a core switches are set to 1000 tasks per unit time, and aggregation and ToR switch have the ability to handle 5000 tasks per a unit time. Since we predict the traffic distribution every 30 minutes and decide system activation states, a unit time is 30 minutes in our simulation. Hosts have 80, 100, 120, and 140 service rates (jobs/sec) which are corresponding processing speed between 1.6 GHz, lower performance host, to 2.4 GHz, high performance host. Data plane switches, aggregation and ToR switches, have higher service rate than control plane switch,core switches, because they just work for forwarding tasks.

Since the problem is NP-hard, we could not obtain the optimal solution for large size data centers. Optimal solutions only when k is equal to 4 were obtained within a reasonable time using CPLEX. However, the computation time of the problem increased significantly when we increase the size of te data center to k that is greater than 8.

Therefore, we employ Simulated Annealing to obtain a near optimal solution of the problem. The algorithm uses 1000 iterations, β is set to 1, and α is 0.9 in Algorithm 1.

Fig. 8 compares the solution of the optimal solution and heuristic solutions when k is equal to 8. Since the computation time of the problem is unrealistic, we compare the upper and lower bounds of the optimal solution when the gap is around 6%. It has taken around 1300 seconds until we reached the 6% gap and could not get to less than 6% gap even after 3 hours of computation. Data center utilization rate exhibits the incoming rate of tasks compared to maximum service rate capacity of data centers. For stability of data centers, the incoming rate cannot exceed the maximum service capacity of the data center. The Simulated Annealing

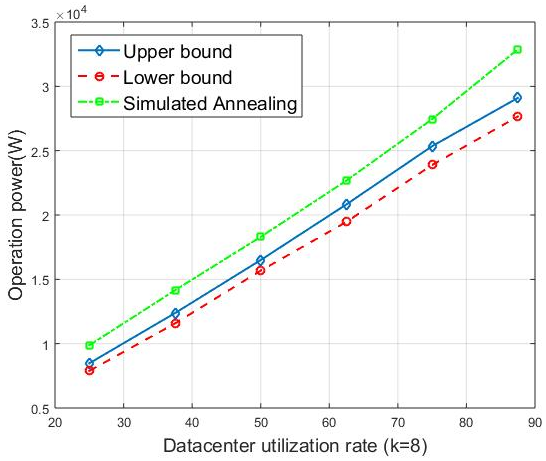

Figure 8: Optimal and heuristic solution comparison

solution has around 10% difference with the upper bound and 15% difference with the lower bound.

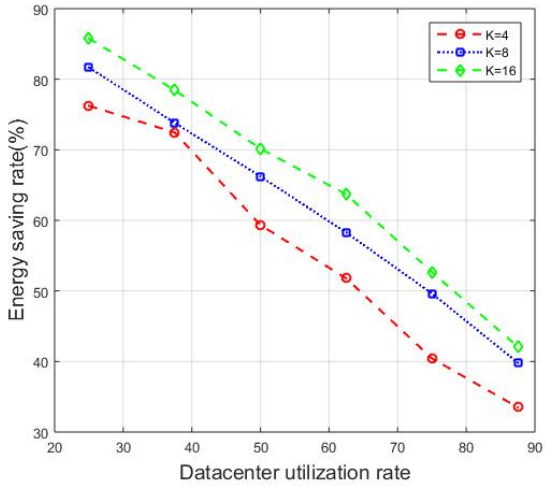

Figure 9: Power saving rate with Simulated Annealing algorithm

Fig. 9 shows how much energy we can save with our model. The power saving rate is calculated by comparing the energy consumption of the heuristic solution and full operation power consumption of data center when the data center operates every switch and host. We measure how much energy we can save depending on the utilization rate and the size of the data center. As the utilization rate increases, the power saving rate is decreased because we cannot turn off many switches and hosts to guarantee the performance of the data center. Also, we can observe that the power saving rate is increased when the size of data center increases.

The predicted Poisson distribution parameters which are obtained in Fig. 5 are employed as the input to the data center in Fig. 10. The predicted parameters include 336 prediction points during a week. So each point corresponds to Poisson distribution parameter for 30 minutes. We have made a modification in the data center parameter in order to make it close to Google cluster environment. The port number k is set to 24, and therefore the data center has 24 core switches, 48 ToR and Aggregation switches, and 3456

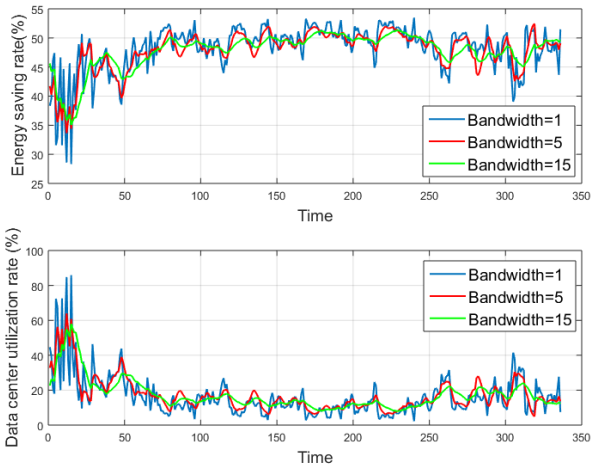

Figure 10: Power saving rate with Simulated Annealing algorithm for different bandwidth values

hosts. Energy consumption and latency parameters are the same as in Table 1 and service rates of switches and hosts are assumed as the same condition.

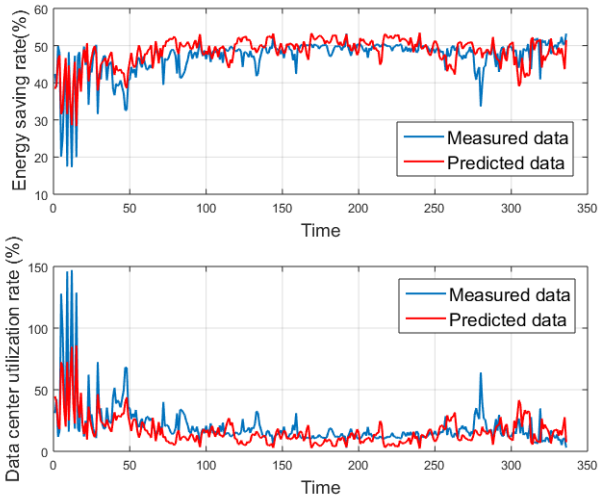

Figure 11: Power saving rate comparison between predicted and measured user requests

The first graph in Fig. 10 presents predicted parameters during a week and the second graph shows how power saving rate is changed depends on λ in every 30 minutes. In Fig. 10, we can observe the data center power saving rate is decreased when the number of arriving tasks increases and it increases when the data center receives fewer requests. The algorithm saves average 47.7 to 48.2% operation energy compared to the full operation state depending on the bandwidth.

Fig. 11 shows the difference between actual parameters and predicted parameters. The prediction model makes accurate prediction generally. However, we can observe the prediction model has a weakness in predicting traffic bursts. If the number of requests soars in a short time, the prediction model could not follow that variation. Power saving rate of the data center model also presents the similar patterns. However, the data center activation model activates switches and hosts by forecasting the high probability traffic of given probability distribution. For example, the α is set to 0.95 in (22), (28), and (38) in the experiment. Therefore, the data center could cover in some degree of traffic burst.

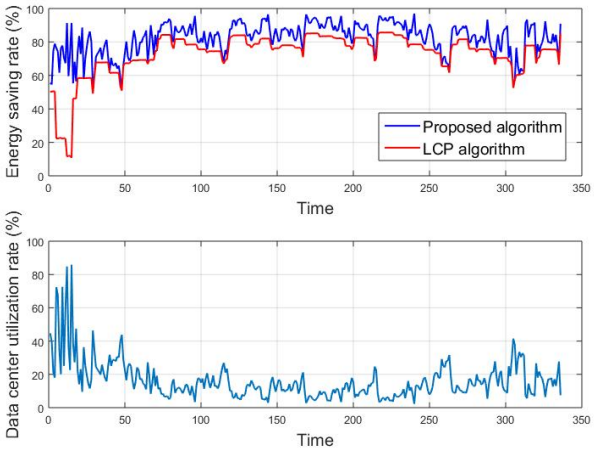

Figure 12: Power saving rate comparison with LCP algorithm

Power saving rates of proposed dynamic data center activation mode algorithm are compared with Lazy Capacity Provisioning (LCP) algorithm proposed in [40]. The LCP algorithm decides the optimal number of activated servers by considering operating cost and transition cost, the cost of server on-off transition. Since the LCP algorithm has been developed for homogeneous server condition and does not affect to activation of switch layer, we modified our simulation model to a homogeneous data center. Heterogeneous server service rates are replaced by a unique service rate and power consumption of servers are set to a single value but the total capacity of the homogeneous data center is set to same as the heterogeneous data center model. Since the LCP algorithm does not control the activation of switch layer, all switches are activated if at least one server is activated in the same POD but switches are deactivated if none of the servers are activated in the same POD. In Fig. 12, power saving rate of the proposed algorithm and the LCP algorithm are compared. The proposed algorithm saves an average of 9.1% more power compared to the LCP algorithm. Also, we can observe that the LCP algorithm does not perform well for traffic burst. During the time slot between 5 to 25, we can observe that the data center utilization rates vary a lot. Since the LCP algorithm minimizes operation cost and transition cost, the data center tries to keep the current state of activated servers. Since the traffic burst is a common situation in cloud computing environments, considering transition cost is not the appropriate model for power saving purposes.

8 Conclusions

This paper proposed adaptive data center activation model with request prediction algorithm. With accurate prediction model, the data center could save the energy in the more efficient way. The prediction model exhibits accurate prediction compared to another prediction algorithms and the adaptive activation model presents flexible energy saving

rate depending on incoming rates of requests. When the model is tested with predicted data based on real data, we could save 30 to 50 percent of energy compared to full operation environment.

REFERENCES

[1] K. Zheng, X. Wang, L. Lie,and X. Wang"*Joint power optimization of data center network and servers with correlation analysis*", INFOCOM, 2014 Proceedings IEEE, pp. 2598-2606, Apr. 2014.

[2] H. Jin, T. Cheocherngngarn, D. Levy, A. Smith, D. Pan, J. Liu, N. Pissinou , *"Joint Host-Network Optimization for Energy-Efficient Data Center Networking"*, in IPDPS, 2013.

[3] *Green Abstraction Layer standard to manage energy consumption of telecom networks*, ETSI Standard (ES) 202 237, http://www.etsi.org/news-events/news/822-2014-09-news-green-abstraction-layer-standard-to-manage-energy-consumption-of-telecom-networks.

[4] Bolla, R. , ,Bruschi, R. , Davoli, F.a , Donadio, P. , Fialho, L. , Collier, M. , Lombardo, A. , Reforgiato, D. , Riccobene, V. , Szemethy, T., *A northbound interface for power management in next generation network devices*, IEEE Communications Magazine, vol. 52, Issue: 1, January 2014.

[5] Bolla, R. , ,Bruschi, R. , Davoli, F.a , Donadio, P. , Fialho, L. , Collier, M. , Lombardo, A. , Reforgiato, D. , Riccobene, V. , Szemethy, T., *The green abstraction layer: A standard power-management interface for next-generation network devices*, IEEE Internet Computing, vol. 17, No. 2, March - April 2013.

[6] J. Whitney and P. Delforge, *"Data center efficiency assessment"*, Issue paper on NRDC (The Natural Resource Defense Council). August 2014.

[7] T. Zhang, K. Chen, C. Xu,G. Sun, T. Wang, and Y. Xie, *"Half-DRAM: a High-bandwidth and Lower-power DRAM Architecture from the Rethinking of Fine-grained Activation"*, Computer Architecture (ISCA) ACM/IEEE 41st International Sumposium on, pp.349-360, June 2014.

[8] A. A. Bankole and S. A. Ajila, *"Predicting cloud resource provisioning using machine learning techniques"*,Proceeding on IEEE Electrical and Computer Engineering (CCECE), pp 1-4. May 2013.

[9] S. Islam, J. Keung, K. Lee, and A. Liu, *Empirical prediction models for adaptive resource provisioning in the cloud*, Future Generation Computer Systems, vol 28, pp 155-162,January 2012.

[10] B. L. Dalmazo, J. P. Vilela, and M. Curado, *"Predicting traffic in the cloud: a statistical approach"*, Proceeding on IEEE Cloud and Green Computing (CGC), pp 121-126. October 2013.

[11] B. L. Dalmazo, J. P. Vilela, and M. Curado, *Onlinetraffic prediction in the cloud: a dynamic window approach*, Proceeding on IEEE Cloud and Green Computing (CGC), pp 9-14, August 2014.

[12] J. Vilaplana, F. Solsona, I. Teixido, J. Mateo, F. Abella, J. Rius, *A queuing theory model for cloud computing*, The Journal of Supercomputing, vol. 69, no. 1, pp 492-507, April 2014.

[13] W.-H. Bai, J.-Q. Xi, J.-X. Zhu, S.-W. Huang, *Performance Analysis of Heterogeneous Data Centers in Cloud Computing Using a Complex Queuing Model*, Mathematical Problems in Engineering, vol. June 2015, Article ID 980945.

[14] J. Cao, K. Hwang, K. Li, A. Y. Zomaya, *Optimal multiserver configuration for profit maximization in cloud computing*, IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 6, pp. 1087-1096, June 2013.

[15] J. Cao, K. Li, and I. Stojmenovic*"Optimal Power Allocation and Load Distribution for Multiple Heterogeneous Multicore Server Processors across Clouds and Data Centers"*, Computers, IEEE Transactionson, vol. 63, pp. 45-58, May 2013.

[16] M. Al-Fares, A. Loukissas, A. Vahdat*"A Scalable, Commodity Data Center Network Architecture"*, Proceedings of the ACM SIGCOMM 2008, pp. 63-74, 2008.

[17] *"Cisco's Massively Scalable Data Center"*,http://www.cisco.com/c/dam/en/us/td/docs/solutions/Enterprise/Data_Center/MSDC/10/MSDC_AAG_1.pdf.

[18] M. Yoon, A. Kamal, Z. Zhu, *"Request prediction in Cloud with a Cyclic Window Learning Algorithm"*, Proceedings of 2016 IEEE GLOBECOM Workshops (GC Wkshps), pp. 1-6, Dec 2016.

[19] W.-C. Chan and Y.-B. Lin *"Waiting time distribution for the M/M/m queue"*, Communications IEEE Proceedings, vol. 150, pp.159-162, June 2003.

[20] Google cluster-trace data, https://code.google.com/p/google clusterdata/

[21] S. Chen, M. Ghorbani, Y. Wang, P. Bogdan, M. Pedram, *Trace-based analysis and prediction of cloud computing user behavior using the fractal modeling technique*, 2014 IEEE International Congress on Big Data, pp 733-739. June/July 2014.

[22] *"HP Altoline 6712 switch series data sheet"*, http://h20195.www2.hp.com/v2/default.aspx?cc=us&lc=en&oid=8566121.

[23] M. Yoon and A. E. Kamal*"Power Minimizationin Fat-Tree SDN Datacenter Operation"*, Proceedings of 2015 IEEE Global Communications Conference (GLOBECOM), pp. 1-7, Dec 2015.

[24] A. M. Hamad and A. E. Kamal*"Power Aware Connection Provisioning For All-Optical Multicast Traffic in WDM Networks"*, Optical Communication and Networking, IEEE/OSA Journals of, vol. 2,pp 481-495, July 2010.

[25] C. E. Leiserson*"Fat-trees: Universal networks for hardware-efficient supercomputing"*, Computers, IEEE Transactions on, vol. C-34, pp.892-901,Oct 1985.

[26] Y. Zhao, J. Zhang, Hui. Yang, and X. Yu*"Data center optimal networks (DCON) with OpenFlow based Software Defined Networking (SDN)"*, Communications and Networking in China (CHINACOM), pp. 771-775, Aug 2013.

[27] A. Sadasivarao, S. Syed, P. Pan, I. Monga, C. Guok, and A. Lake*"Bursting Data Between Data Centers: Case for Trnasport SDN"*, High-Performance Interconnects (HOTI), pp. 87-90, Aug 2013.

[28] D. Li, Y. Shang, and C. Chen*"Software Defined Green Data Center Network with Exclusive Routing"*, INFOCOM, 2014 Proceedings IEEE, pp. 1743-1751, April 2014.

[29] R. Ghosh, F. Longo, R. Xia, V. K. Naik, and K. S. Trivedi*"Stochastic Model Driven Capacity Planning for an Infrastructure-as-a-Service Cloud"*, Services Computing, IEEE Transactions on, vol. 7, pp. 667-680, Sep 2013.

[30] W. Ni, C. Huang, J. Wu*"Provisioning high-availability datacenter networks for full bandwidth communication"*, Communications and Networking in the Cloud, vol. 68, pp. 71-94, Aug 2014.

[31] Y. Zhang and N. Ansari*"HERO: Hierarchical Energy Optimization for Data Center Networks"*, Systems Journal, IEEE, pp. 1-10, Oct 2013.

[32] S. Paul, R. Jain, M. Samaka, and J. Pan*"Application delivery in multi-cloud environments using software defined networking"*, Communication and Networking in the Cloud, vol 68, pp. 166-186, Aug 2014.

[33] J. Fan, *"Local linear regression smoothers and their minmax efficiencies"*, Ann. Statist, vol 21, No. 1, pp 196-216. November 1993.

[34] E. Caron and F. Desprez, *"Forecasting for grid and cloud computing on demand resources based on pattern matching"*, Proceeding on IEEE Cloud Computing Technology and Science (CloudCom), pp 456-463. November 2010.

[35] M. Li and S. C. Lim, *"Modeling network traffic using generalized Cauchy process"*, Physica A: Statistical Mechanics and its application, vol 387, pp 2584-2594. April 2008.

[36] B. L. Dalmazo, J. P. Vilela, and M. Curado, *"Online traffic prediction in the cloud: a dynamic window approach"*, Proceeding on IEEE Cloud and Green Computing (CGC), pp 9-14. August 2014.

[37] T. Wood, L. Cherkasova, K. Ozonat, and P. Shenoy, *"Profiling and modeling resource usage of virtualized application"*,Proceeding of the 9th ACM/IFIP/USENIX International conference on middleware, pp 386-387. December 2008.

[38] S. Andrade-Morelli, E. Ruiz-Snchez, E. Granell, J. Lloret, *"Energy consumption of wireless network access points, Green Communication and Networking"*, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering Volume 113, 2013, pp 81-91.

[39] W. Fang, Z. Wang, J. LIoret, D. Zhang, Z. Yang, *"Optimising data placement and traffic routing for energy saving in Backbone Networks"*, Transactions on Emerging Telecommunications Technologies 25 (9), 914-925. 2014

[40] M. Lin, A. Weirman, L.L.H.Andres, E. Thereska, *"Dynamic right-sizing for power-proportional data centers"*, Proceedings on IEEE INFOCOM 2011, pp 1098-1106. April 2011.