Power Minimization in Fat-Tree SDN Datacenter Operation

Min Sang Yoon, and Ahmed E. Kamal
Department of Electrical and Computer Engineering
Iowa State University, IA, 50010
{my222, kamal}@iastate.edu

***Abstract*—The problem of energy consumption in data centers has attracted many researchers interest recently. This paper proposes an optimal energy consumption Software Defined Network (SDN) data center model using the dynamic activation of hosts and switches. We model switches and hosts as queues and formulate a Mixed Integer Linear Programming (MILP) model to minimize energy consumption while guaranteeing Quality of Service (QoS) of data center. Our purpose is minimizing static power, port power, and memory power of data centers. Since the problem is NP-hard, we adopt Simulated Annealing algorithm to obtain the solution. Through numerical experiment, we could observe that our model is able to save reasonable energy compared to the full operation data center model.**

I. Introduction

Power consumption in data centers has recently received significant interest. Increase in data traffic with highend mobile devices and expansion of Cloud computing environment have demanded continuous growth in data center usage. As a result, data centers consumed 91 billion kilowatt-hours of electricity in 2013, which is equivalent to the energy from 34 large coal fire power plants.

Internet traffic arriving to data center is not constant. The traffic is usually increased in the daytime and decreased during the night. Therefore, running data centers continuously will cause wastage of energy because data centers will be wasting many energy to operate unnecessary hosts and switches when there is low traffic input to data centers. Thus, dynamic operation of data center techniques have been suggested by many researchers that control the activation of hosts in data centers. However, they did not consider dynamic activation of switches in data center because of its complexity. Deciding the activation of switches and hosts is very important. If the data center is running a so small number of hosts compared to requests, it will not be able to cover all requests from users. Even if there are enough hosts, if the data center is operating too small number of switches for energy saving purposes, it will increase the latency in transferring data in the data center. Also, it is important to determine which switches we operate. Data centers are usually built with tree architectures, so if we do not activate necessary switches that connect core switches and hosts, this will cause failure of job allocation. We also activate memory dynamically because memory also consumes much energy in data center. According to [1], DRAM consumes 25% of total power in data centers.

In this paper, we develop an optimal energy consumption model in a Fat-tree architecture data center which depends on incoming job requests by using dynamic activation model. Our model guarantees Quality of Service (QoS) of data center by observing delay constraints and performance of data center with architecture constraint.

In terms of memory constraint, our model estimates the number of jobs with high probability in each switch and host, and decides how much memory we need to activate.

Scalability of data centers is an important issue. As traffic requests to data centers increase, many data center operator will face the need for increasing data centers sizes. Most of data centers are designed homogeneously, which means it is not easy to expand data center resource capacity. Our model employs a heterogeneous Fat-tree architecture of data center that is controlled with SDN controler. SDN provides programmable centralized controls plane that controls distribution of jobs and routing paths. Since control plane manages data centers, we can configure more scalable heterogeneous data center. When we upgrade hosts as necessary, we do not have to modify the setting of all configurations of switches. By changing the node information only in the control plane we can scale the data center easily.

We model an optimal SDN controlled data center using Mixed Integer Linear Program (MILP) model. Switches are considered as M/M/1 queues and hosts are considered as M/M/c queues. This model will minimize the operation power by deciding active switches, and hosts and operating memory in each switch and host. We can specify working switches by deciding load distribution from upper layer switches. So the problem will decide the workload distribution in each link as well. We also minimize the switch port power. Ports consume power when they forward jobs. If each switch consumes different port power, we can minimize the power consumed by ports by forwarding packets though inexpensive switch ports.

The rest of paper is organized as follows. Section II discuss the related work on power minimization in data centers. We introduce the system model in Section III. In Section IV, we formulated MILP problem. Since the problem is NP-hard, we use simulated annealing algorithm to obtain heuristic solution of the optimization problem in section V. Section VI introduces the numerical results and we will end this paper with conclusions in Section VII.

II. Related work

Fat-tree architecture was invented by Charles Leiserson in [4], and Mohammad Al-Fares *et al*. applied this architecture to data center network in [5]. This paper showed the scalability and superiority of the Fat-tree architecture in terms of interconnection.

Many researchers propose data center architectures using SDN technology. Yongli et al have configured OpenFlow based SDN datacenter testbed including heterogeneous networks [6]. They evaluated the performance of SDN data center and could observe improvement in allocation efficiency of application and network resources. Abhinava Sadasivarao *et al*. suggest an SDN enabled optical transport architecture between data centers [7]. Since data centers are spread over long distance and are required to communicate with each other with growth of Cloud Computing, they present a strategy that optimizes transport application between data center with SDN technology. Dan Li et al proposed energy aware flow scheduling in data center by using SDN technique [8]. They could save more energy with suggested scheduling algorithm and improve utilization ratio of switches and links.

Energy saving in data center is considered in many aspects

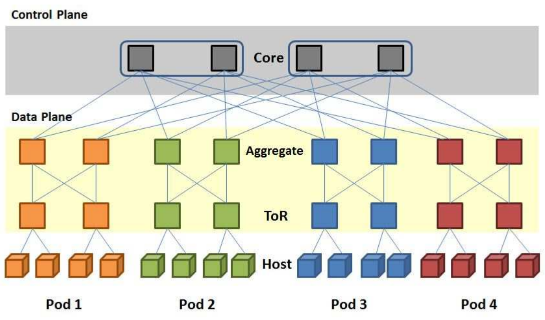

Figure 1: Fat-tree SDN data center architecture (k=4)

because of its importance. Junwei Cao *et al*. propose optimal power allocation and load distribution in a heterogeneous data center environment [9]. They model 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 power optimization in data center network with correlation analysis [10]. They suggest power optimization strategy that leverages workload correlation analysis to jointly minimize total power consumption of servers and data centers. After they analyze the correlation among Virtual Machines (VMs), they consolidate VMs that are not correlated with each other onto the same physical servers. Through this approach they could achieve up to 51.6% of energy saving in their testbed.

III. System Model

There are many data center architectures: Fat-tree, VL2, Bcube, and MDcube. We will use the most popular architecture model, the Fat-tree architecture. We also employ the SDN technique in order to implement our strategy.

A. *Fat-Tree Architecture*

Fat-tree is the most popular data center architecture as many companies are using this architecture. Fat-tree is composed of three switch layers (core, aggregation, and ToR layer) and host layers. All switches have the same number of ports. If we assume the number of ports in each switch is k, we can have the k number of core switches and pods. Core switches are grouped into $k/2$ groups. Since a core switch has k ports, we can say each core switch is connected to all pods. For example, the first core switch in the each group is connected with the first aggregation switch in each pod and the second core switch in each group is connected with the second aggregation switch in each pod. Each pod has $k/2$ number of aggregation and ToR switch. Aggregation and ToR switches have complete bipartite graph. Since a ToR switch uses $k/2$ ports for connecting to aggregation switches they can serve $k/2$ hosts each. Therefore, the data center can have $k^3/4$ number of hosts total.

In Fat-tree architecture data center, we cannot turn off necessary switches to operate required hosts. For example, we cannot turn off the first ToR switch when we want to allocate jobs to the first host in the first pod. Aggregate switches and ToR switches form a complete bipartite connection. So we do not have to turn on the first aggregation switch for the first ToR switch but we need to turn on at least one aggregation switch for allocating workload to the first pod. In our model, the core switch should be able to reach to host with three links and host should reach a core switch with three links as well. So, we will turn on and off switches without violation of this rule to guarantee the proper performance of data center.

B. *SDN Data Center Architecture*

The SDN model is composed of a control plane and a data plane. The control plan plays the role of managing the network and deciding routing paths. The data plane just forwards data according to the forwarding table. In data centers, we define the core switch layer as the control plane and the aggregation and ToR switch become the data plane. Since each core switch plays the role of manager and decides the routing paths, the data center network will be a distributed SDN model. When a user request arrives to the data center, the core switch decides which host will be the best for the request and decide routing path to hosts. If inter-data traffic is used for moving a data set from a host to another host, the host should send a request to core layer as well. In traditional data centers, a host could move a dataset to another host in the same pod by going through just ToR and aggregation switches. However, host should go through core switch any case in this model. Although the SDN data center model will cause a minor disadvantage compare to the traditional data centers in terms of inter-pod communication, it will be a minor issue when we consider efficient management of data centers through the SDN data center architecture.

IV. Problem Formulation

We formulated problem as an MILP. Our objective is to minimize the static power, port power, and memory power consumption in data centers while guaranteeing QoS and architecture of Fat-tree data center. This model decides the active state of switches and hosts by using binary variables and determines load distribution state of switches with continuous variables. Based on allocated workload in each switch, the switch can determine how many memory slots they will need to be on. We assume that jobs arrive to the data center according to a as Poisson distribution. We have problem 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.
- $H_{ij\_T_{mn}}$: working state of i_{th} pod j_{th} host connected to T_{mn} ToR switch; binary variable.
- $P^{static}_{C_{ij}}$, $P^{static}_{A_{ij}}$, $P^{static}_{T_{ij}}$, $P^{static}_{H_{ij\_T_{mn}}}$: Static power consumption of core, aggregate, ToR switches and hosts during a unit time; constant.
- $P^{port}_{C_{ij}}$, $P^{port}_{A_{ij}}$, $P^{port}_{T_{ij}}$: Power consumption of each port in core, aggregate, and ToR switche per job; constant.
- $P^{memory}_{C_{ij}}$, $P^{memory}_{A_{ij}}$, $P^{memory}_{T_{ij}}$, $P^{memory}_{H_{ij\_T_{mn}}}$: Memory power consumption of core, aggregate, ToR switches and hosts during a unit time; constant.
- λ: Job arrival rate to data center; constant.
- $\lambda_{C_{ij}}$, $\lambda_{A_{ij}}$, $\lambda_{T_{ij}}$, $\lambda_{H_{ij\_T_{mn}}}$: 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.
- $\mu_{C_{ij}}$, $\mu_{A_{ij}}$, $\mu_{T_{ij}}$, $\mu_{H_{ij\_T_{mn}}}$: service rate of core switch; constant.
- $M_{C_{ij}}$, $M_{A_{ij}}$, $M_{T_{ij}}$, $M_{H_{ij\_T_{mn}}}$: The number of operating memory in core, aggregate, ToR switches, and hosts; integer variable.

A. *Objective Function*

$$P^{static} + P^{port} + P^{memory} \tag{1}$$

The objective function includes static power, port power, and memory power consumption during a unit time. Static power is constant power by turning on switches or hosts. Port power is the power consumption for transmitting a job through the port. We assume ports keep idle state when they do not transmit jobs and consume no power. So port power is calculated depending on the number of jobs they transmit. Memory power is determined depending on how many memory slots are powered. If we turn on too many memory slots rather than required the number in a switch or a host, it will waste energy. Therefore, we propose to turn on the exact number of required memory based on statistical estimation.

1) Static Power:

$$p^{static} = \sum_{i=1}^{k/2}\sum_{j=1}^{k/2} C_{ij}P_{C_{ij}}^{static} + \sum_{i=1}^{k}\sum_{j=1}^{k/2} A_{ij}P_{A_{ij}}^{static} + \sum_{i=1}^{k}\sum_{j=1}^{k/2} T_{ij}P_{T_{ij}}^{static} + \sum_{i=1,m=1}^{k}\sum_{j=1}^{k/2}\sum_{n=1}^{k/2} H_{ij\_T_{mn}}P_{H_{ij\_T_{mn}}}^{static} \quad (2)$$

We can calculate the static power by multiplying binary variables that represent the state of each switch and the host and power consumption constant of each switch and host.

2) Port Power:

$$p^{port} = \sum_{i=1}^{k/2}\sum_{j=1}^{k/2}\sum_{l=1}^{k} \lambda_{C_{ij\_l}}P_{C_{ij}}^{port} + \sum_{i=1}^{k}\sum_{j=1}^{k/2}\sum_{l=1}^{k/2} \lambda_{A_{ij\_l}}P_{A_{ij}}^{port} + \sum_{i=1}^{k}\sum_{j=1}^{k/2}\sum_{l=1}^{k/2} \lambda_{T_{ij\_l}}P_{T_{ij}}^{port} \quad (3)$$

Port power is need to be considered only in switches. The problem decides how each switch will distribute arrived jobs through which port. Therefore, we can measure the port power consumption by multiplying the amount of jobs passing through each port and power consumption variable of each port.

3) Memory Power:

$$p^{memory} = \sum_{i=1}^{k/2}\sum_{j=1}^{k/2} M_{C_{ij}}P_{C_{ij}}^{memory} + \sum_{i=1}^{k}\sum_{j=1}^{k/2} M_{A_{ij}}P_{A_{ij}}^{memory} + \sum_{i=1}^{k}\sum_{j=1}^{k/2} M_{T_{ij}}P_{T_{ij}}^{memory} + \sum_{i=1,m=1}^{k}\sum_{j=1}^{k/2}\sum_{n=1}^{k/2} M_{H_{ij\_T_{mn}}}P_{H_{ij\_T_{mn}}}^{memory} \quad (4)$$

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

B. Load Distribution Constraint

Workload should be distributed in a rational manner in data centers. Since we are purposing to turn off part of the switches and hosts in the data center, we have to decide exactly which port will transfer jobs the lower layers. 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 port.

1) Load Distribution in each layer: Summation of workload in each layer should be equivalent to λ, the total arriving workload to the data center. This constraint purposes to the allocate workload to working switches and hosts while guaranteeing that we forward all requests to hosts.

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

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

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

$$\sum_{i=1,m=1}^{k}\sum_{j=1}^{k/2}\sum_{n=1}^{k/2} H_{ij\_T_{mn}}\lambda_{H_{ij\_T_{mn}}} = \lambda \quad (8)$$

Equations (5)-(8) are multiplications of binary variables and continuous variables, so it is non-linear equation. Therefore we will linearize the non-linear formulation by using the equation. When x is a binary variable and y is a continuous variable, we can linearize their product by replacing it by a continuous variable t with the constraints shown below.

$$0 \le t \le \max(y)\cdot x \quad (9)$$

$$y - (1-x)\cdot\max(y) \le t \le y \quad (10)$$

The maximum of continuous variables, $\lambda_{C_{ij}}$, $\lambda_{A_{ij}}$, $\lambda_{T_{ij}}$, and $\lambda_{H_{ij\_T_{mn}}}$ is λ because the maximum workload can be allocated to each switch and host 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.

2) Link Distribution: As mentioned before, it is important which port we use for transferring jobs. If we use the port connected with an inactive switch or host for transmitting that job, we will lose request and cannot forward it properly.

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

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

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

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

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

$$\lambda_{T_{ij\_l}} = \lambda_{H_{il\_T_{ij}}}, \forall i,j,T_{ij} \quad (16)$$

Equation (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.

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

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

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 any amount of workload to turned off switches because the summation of workload is still same if the switch is turned off. Therefore, we need constraints that prohibit workloads from being allocated to inactive switches and hosts.

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

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

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

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

C. Performance Constraints

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

1) Latency Constraint in Switch: Switches are modeled as M/M/1 queues. We can use the Cumulative Density Function (CDF) of residual time in M/M/1 queue to obtain the residual time equation in M/M/1 queue.

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

Equation (21) presents the probability that the residual time in the queue will be less than $t_{residual}$. Therefore, we can achieve the residual time in the queue by letting $F_R(t_{residual}) = \alpha$, α is close to 1. If $t_{residual}$ is obtain to satisfy α. $t_{residual}$ will be the residual time in M/M/1 queue that satisfies the probabilistic upper bound on the residual time in queue.

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

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

If we take log base e on both sides, we can obtain the following equation.

$$\ln(1 - \alpha) = -(\mu - \lambda_{switch})t_{residual} \tag{24}$$

Then we can obtain residual time in the queue like below,

$$t_{residual} = -\frac{\ln(1 - \alpha)}{(\mu - \lambda_{switch})} \tag{25}$$

Since the residual time cannot exceed the latency constraint, Latency, we can obtain below constraint.

$$-\frac{\ln(1 - \alpha)}{(\mu - \lambda_{switch})} \leq Latency \tag{26}$$

If we move the denominator in the left side, we can achieve the linear constraint below because λ_{switch} is continuous variable in each switch.

$$-\ln(1 - \alpha) \leq Latency \cdot (\mu - \lambda_{switch}) \tag{27}$$

By using (27), we can obtain latency constraint about all switches.

2) Latency in Host: We use same strategy above to obtain the delay in hosts. The host is modeled as M/M/c queue. The CDF of residual time in M/M/c queue is [2]:

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

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 small number and ρ represents $\frac{\lambda}{\mu}$. Therefore, we can assume p_c is equal to 1 for the simplification of the problem.

$$F_R(t_{residual}) = 1 - \frac{c}{c - \rho} \cdot e^{-(c\mu - \lambda)t_{residual}} \tag{29}$$

If we obtain residual time which covers the probability α, with $\alpha \approx 1$, then

$$1 - \frac{c}{c - \rho} \cdot e^{-(c\mu - \lambda)t_{residual}} = \alpha \tag{30}$$

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

If we take log to base e in both sides,

$$\ln \frac{(1 - \alpha)(1 - \rho)}{c} = -(c\mu - \lambda)t_{residual} \tag{31}$$

Then we can evaluate the residual time in the M/M/c queue.

$$t_{residual} = \frac{\ln \frac{(1-\alpha)(1-\rho)}{c}}{-(c\mu - \lambda)} \tag{32}$$

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

$$\frac{\ln \frac{(1-\alpha)(1-\rho)}{c}}{-(c\mu - \lambda)} \leq Latency \tag{33}$$

Which can also be written by

$$-\ln(1 - \alpha) - \ln(1 - \rho) + \ln c \leq Latency(c\mu - \lambda) \tag{34}$$

(34) is non-linear. So we will linearize it by using the pairwise approximation technique used in [3]. ρ has limited domain between 0 to 1. Therefore, we can linearize them by calculating a linear function in each sub domain. We divide ρ to three sub-domains: [0, 0.75], [0.75, 0.95], and [0.95, 1]. Then, we could obtain a linear function that fits each sub-domain, $f_1(\rho)$, $f_2(\rho)$,and $f_3(\rho)$.

Through approximation we could achieve three linear functions in each sub-domains, $f_1(\rho) = 1.72\rho$, $f_2(\rho) = 7.1679\rho - 4.1698$, $f_3(\rho) = 40.2359\rho - 35.6308$. By replacing $-\ln(1 - \rho)$ in (34) by $\max_y f_y(\rho)$, y represents a linear function in y sub-domain, we can obtain the linear constraint below.

$$\max_y f_y(\rho) - \ln(1 - \alpha) + \ln c \tag{35}$$
$$\leq Latency \cdot (c\mu_{H_{ij\_T_{mn}}} - \lambda_{H_{ij\_T_{mn}}}), \forall i, j, m, n, y$$

3) Service Rate: For the stability of the system, the summation of service rate of each layer should be greater than the total number of jobs.

$$\sum_{i=1}^{k/2} \sum_{j=1}^{k/2} \mu_{C_{ij}} C_{ij} \geq \lambda \tag{36}$$

$$\sum_{i=1}^{k} \sum_{j=1}^{k/2} \mu_{A_{ij}} A_{ij} \geq \lambda \tag{37}$$

$$\sum_{i=1}^{k} \sum_{j=1}^{k/2} \mu_{T_{ij}} T_{ij} \geq \lambda \tag{38}$$

$$\sum_{i=1,m=1}^{k}\sum_{j=1}^{k/2}\sum_{n=1}^{k/2}\mu_{H_{ij\_T_{mn}}}H_{ij\_T_{mn}} \geq \lambda \tag{39}$$

D. Memory Constraint

Main memory consumes almost 25% of total energy consumption in data center. Therefore, diminishing the number of working memory slots will contribute to saving energy consumption in data centers. We decide the number of working memory slots based on the probability that an arriving job is rejected.

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

$$p(loss) = p_{n_q} \tag{40}$$

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

$$p(queuing) = \alpha \tag{41}$$

Then we can get equation below:

$$p(loss) = 1 - p(queuing) = 1 - \alpha \tag{42}$$

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

$$p(loss) = 1 - \sum_{i=0}^{n_q-1} p_i \tag{43}$$

If we substitute p(loss) in (42) with (43),

$$\sum_{i=0}^{n_q-1} p_i = \alpha \tag{44}$$

$p_i = \rho^i(1-\rho)$ in M/M/1 queue. So from (44) we can obtain,

$$1 - \rho^{n_q-1} = \alpha \tag{45}$$

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

$$n_{switch} = \frac{\ln(1-\alpha)}{\ln(\rho)} + 1 \tag{46}$$

Equation (46) represents the number of existing jobs in a switch with very high probability α. Based on the number of jobs in queue, we can obtain the number of memory slots we needed. If we assume the expected size of a job is J and memory slot size is MS, we can make memory constraint like below.

$$n_{switch} \cdot J \leq M_{switch} \cdot MS \tag{47}$$

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

2) Memory Constraint in host: Hosts are considered as M/M/c queues. We can use the approach in (44) but 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:

$$\sum_{i=c}^{\infty} p_i \approx 1 \tag{48}$$

Steady state of probability in M/M/c queue is given by, when $i \geq c$:

$$p_i = \frac{c^c \rho^n}{c!} p_0, \rho = \frac{\lambda_{host}}{c\mu} \tag{49}$$

Initial state probability p_0 can be found by substituting p_i in (48) with (49).

$$\sum_{i=c}^{\infty} \frac{c^c \rho^n}{c!} p \approx 1 \tag{50}$$

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

So we can induce initial probability p_0,

$$p_0 \approx \frac{c!(1-\rho)}{c^c \rho^c} \tag{51}$$

By using (51), we can use steady state probabilities by substituting p_0 in (49).

$$p_i = \frac{c^c}{c!} \rho^i p_0 \tag{52}$$

If we insert (52) into p_i in (44), we can obtain the number of jobs in hosts according to in coming rate λ.

$$n_{host} = \frac{\ln(1-\alpha)}{\ln(\rho)} + c \tag{53}$$

By using (53), we can achieve the required number of memory slots that can accommodate jobs in hosts with the following constraint.

$$n_{host} \cdot J \leq M_{host} \cdot MS \tag{54}$$

However, (47) and (54) 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 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 that achieve 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:

$$f_{n1}(\rho) = \frac{f_n(\rho_1^*)}{\rho_1^*}\rho \tag{55}$$

$$f_{n2}(\rho) = \frac{f_n(\rho_2^*) - f_n^*(\rho_1^*)}{\rho_2^* - \rho_1^*}(\rho - 1) + f_n(\rho_2^*)$$

$$f_{n3}(\rho) = \frac{\mu - f_n(\rho_2^*)}{1-\rho_2^*}\rho - \mu - \frac{\mu - f_n(\rho_2^*)}{1-\rho_2^*}$$

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 (47) and (54) like below in each switch and host:

$$\max_y(f_{ny}(\rho_{C_{ij}})) \leq \frac{M_{host} \cdot MS}{J} \tag{56}$$

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

E. Connectivity Constraint

The architecture constraint is applied to the problem to protect the Fat-tree architecture of data centers. If we turn off necessary switches for connecting to working hosts, it will cause the disconnection between the core layer and hosts. Therefore, we define the least constraints that maintain the architecture of data centers.

$$\frac{\sum_{j=1}^{k/2} T_{ij}}{k} \leq \sum_{j=1}^{k/2} A_{ij}, \forall i \tag{57}$$

The ToR switches and aggregation switches in the same pod have 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. (57) activates at least one aggregation switch if ToR switches are activated in the same pod.

$$\frac{k}{2} \sum_{j=1}^{k/2} H_{ij\_T_{mn}} \leq T_{mn}, \forall i \tag{58}$$

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 host by $2/k$ in order to make the upper bound of summation equal to 1.

V. Simulated Annealing Algorithm

The optimization problem includes binary variables and integer variables in the objective function and constraints: C_{ij}, A_{ij}, T_{ij}, $H_{ij\_T_{mn}}$, $M_{C_{ij}}$, $M_{A_{ij}}$, $M_{T_{ij}}$, $M_{H_{ij\_T_{mn}}}$. The binary variables decide the state of switches and hosts and the integer variables decide the number of working memory slots in each switch and host. As we include integer variables in the problem, this optimization becomes a NP-hard problem. When we consider small size data center, the optimization problem can be solved in a reasonable time. However, data center is usually composed of a large number of hosts and servers. As the size of data center increases, it will consume more time to get the solution. So, we will use Simulated Annealing algorithm, which is one of the popular heuristic algorithms that can find a near optimal solution in reasonable time.

Simulated Annealing is randomized algorithm that provides reasonable global solution in acceptable computation time. For the input parameters, we need the service rates of hosts and switches, total incoming jobs into the data center, parameter β that has value between 0 to 1, and maximum number of 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 very close to service rates. Therefore, we decide the allocation rate that does 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 induce which switch will be needed to operate the data center properly. So 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 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 solution from line 5 to line 20. We find neighbor solution of job allocation into hosts (line 6). In the same method, we can find which switches are required to be operated for working hosts. So we turn on the required switches by using switch() function (line 7). Then, we can calculate the power consumption of neighbor solution in line 8. If power consumption of neighbor solution is less than power consumption of candidate solution, we replace the candidate solution with neighbor solution. Simulated Annealing is a randomized algorithm. We also replace candidate solutions with neighbor solutions with some probability to avoid isolation of solution. 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 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 the value less than 1, to decrease the β in every iteration (line 16 and line 18). The algorithm repeats finding neighbor solution until t reaches to iteration. After finishing iteration, the candidate solution will become a near optimal solution of the problem.

Algorithm 1 Simulated Annealing Algorithm

```
Require: $S_{H_{ij\_T_{il}}}, S_{T_{ij}}, S_{A_{ij}}, S_{C_{ij}}, \lambda, \beta$, iteration
Ensure: $P_{minimum}$, $\lambda_{H_{ij\_T_{il}}}, \lambda_{T_{ij}}, \lambda_{A_{ij}}, \lambda_{C_{ij}}$
 1: We generate initial allocation to hosts $\lambda_{h_{ij\_T_{mn}}}$ 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($\lambda_{h_{ij\_T_{il}}}$)
 4: $P_{minimum}$=$Cal\_power(\lambda_{h_{ij\_T_{il}}}, \lambda_{T_{ij}}, \lambda_{A_{ij}}, \lambda_{C_{ij}})$
 5: while t$\leq$ iteration do
 6:   Search neighbor allocation $\lambda'_{h_{ij\_T_{il}}}$
 7:   $\lambda'_{T_{ij}}, \lambda'_{A_{ij}}, \lambda'_{C_{ij}}$=Switch($\lambda'_{H_{ij\_T_{il}}}$)
 8:   $P_{candidate} = Cal\_power(\lambda'_{H_{ij\_T_{il}}}, \lambda'_{T_{ij}}, \lambda'_{A_{ij}}, \lambda'_{C_{ij}})$
 9:   r←rand()
10:   if $P_{candidate} < P_{minimum}$
      or $e^{\frac{p_{minimum}-p_{candidate}}{\beta}} > r$ then
11:     $\lambda_{H_{ij\_T_{il}}} \leftarrow \lambda'_{H_{ij\_T_{il}}}$
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: $P_{minimum}$, $\lambda_{H_{ij\_T_{il}}}, \lambda_{T_{ij}}, \lambda_{A_{ij}}, \lambda_{C_{ij}}$
```

VI. Numerical Analysis

Optimization problem is solved by CPLEX and Simulated Annealing algorithm is implemented in C. We designed heterogeneous environment data center having different service rate of hosts. However, all switches have same performance in each layer. We set the parameters of the model as shown in Table 1.

Variable	Values
$P^{static}_{C_{ij}}$, $P^{static}_{A_{ij}}$, $P^{static}_{T_{ij}}$	300W, 150W, 150W
$P^{static}_{H_{ij\_T_{mn}}}$	[100, 150, 200, 250]W
$P^{port}_{C_{ij}}$, $P^{port}_{A_{ij}}$, $P^{port}_{T_{ij}}$	0.0005W/job
$P^{memory}_{C_{ij}}$, $P^{memory}_{A_{ij}}$, $P^{memory}_{T_{ij}}$, $P^{memory}_{H_{ij\_T_{mn}}}$	25W
Latency	0.01 sec

Table I: Value of key parameters

Static power of switches is different depends on performance. Since core switches implement path calculation and management as control plane, they consume more energy than aggregation and ToR switches. Hosts have different static power depending on their service rate. Our model includes four types of hosts, so hosts have four types of static power depends on its service rate. High performance hosts consume more power. Identical hosts will be allocated in the same pod. Thus, hosts will be heterogeneous between pods.

The service rates of hosts and switches change depending on the data center size. For example, we set the service rate of a core switch to 100000, aggregation and ToR switch to 500000, and hosts have 5000, 6000, 7000, and 8000 when k is equal to 8. Basically, aggregation and ToR switches have higher service rate than core switch because they just forward jobs. However, core switches calculate path and manage data center network so service rates is lower than other layer switches. Since we have four types of hosts, they have four types of service rate as well.

We set the expected size of each job to 2MB and all switches and hosts have four 512MB memory.

Since the problem is NP-hard, we could not obtain optimal solution about large size data center. Optimal solution of the problem is achieved when k is equal to 4 in reasonable time. However, computation time of the problem increased extremely when we enlarge the size of data center to k that is greater than 8.

We ran Simulated Annealing for 1000 iterations. β is set to 1 and α is 0.9 in Algorithm 1.

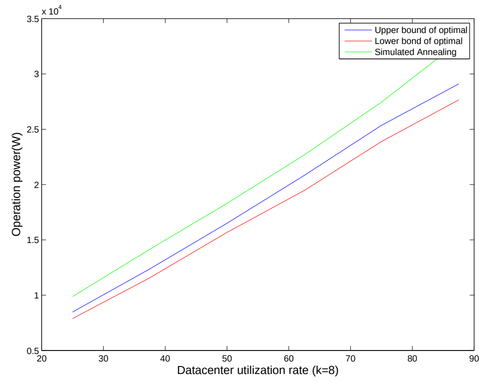

Figure 2: Optimal and heuristic solution comparison

Figure 2 compares the solution of optimal solution and heuristic solution when k is equal to 8. CPLEX uses branch and bound algorithm. Since the computation time of the problem is unrealistic, we compare the upper and lower bound of optimal solution when gap is around 6%. It has taken around 1300 seconds until we reached the 6% gap. Data center utilization rate exhibits incoming rate of jobs compared to maximum capacity of data centers. For stability of data center, incoming rate cannot exceed maximum capacity of data center. The Simulated Annealing solution has around 10% difference with upper bound and 15% difference with lower bound. So we could see the Simulated Annealing algorithm work properly.

Figure 3 shows how much energy we can save with our model. Energy saving rate is calculated by comparing energy consumption of heuristic solution and expected energy when then data center operates every switch and host. We measure how much energy we can save depending on the utilization rate and size of data center. As utilization rate increases, the energy 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 energy saving rate is increased when the size of data center increases.

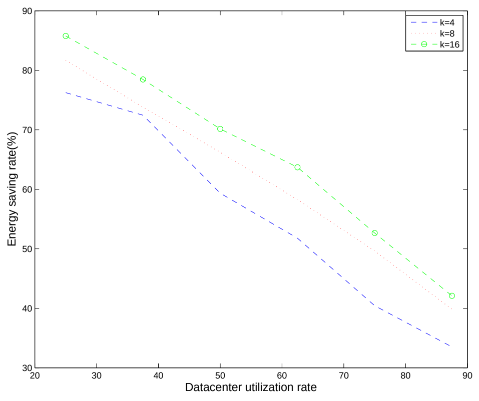

Figure 3: Energy saving rate with Simulated Annealing algorithm

VII. Conclusions

This paper presented a power minimization model in Fat-tree architecture data center. The problem is modeled as a MILP and could achieve optimal solution about small size data center. For scalability of the problem, we employ the Simulated Annealing algorithm. We could achieve high energy saving rate with reasonable computation time with the Simulated Annealing algorithm. The result exhibits that we can save much energy with this model by preventing wastage of resources. Also, we could have additional saving rate with dynamic memory model in high request situation as well.

References

[1] 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.
[2] 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.
[3] 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.
[4] C. E. Leiserson*"Fat-trees: Universal networks for hardware-efficient supercomputing"*, Computers, IEEE Transactions on, vol. C-34, pp.892-901,Oct 1985.
[5] M. Al-Fares, A. Loukissas, A. Vahdat*"A Scalable, Commodity Data Center Network Architecture"*, Proceedings of the ACM SIGCOMM 2008, pp. 63-74, 2008.
[6] 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.
[7] 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.
[8] 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.
[9] 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.
[10] 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.
[11] C. DeCusatis, M. Haley, T. Bundy, R. Cannistra, R. Wallner, J. Parraga, and R. Flaherty*"Dynamic, software-defined service provider network infrastructure and cloud drivers for SDN adoption"*,Communications Workshop (ICC(, 2013 IEEE International Conference on, pp. 235-239, June 2013.
[12] M. Mechtri, I. Houidi, W. Louati, and D. Zeghlache*"SDN for Inter Cloud Networking"*, Future Networks and Services (SDN4FNS), 2013 IEEE SDN for, pp. 1-7, Nov, 2013.
[13] 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.
[14] 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.
[15] Y. Zhang and N. Ansari*"HERO: Hierarchical Energy Optimization for Data Center Networks"*, Systems Journal, IEEE, pp. 1-10, Oct 2013.
[16] 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.