Power Control by Distribution Tree with Classified Power Capping in Cloud Computing

Zhengkai Wu  
Computer Science  
UCF  
Orlando, US

Jun Wang  
Computer Science  
UCF  
Orlando, Florida

Abstract—Power management is becoming very important in data centers. Cloud computing is also one of the newer promising techniques, that are appealing to many big companies. To apply power management in cloud computing has been proposed and considered as green computing. Cloud computing, due to its dynamic structure and property in online services, differs from current data centers in terms of power management. To better manage the power consumption of web services in cloud computing with dynamic user locations and behaviors, we propose a power budgeting design based on the logical level, using a distribution tree. By setting multiple trees, we can differentiate and analyze the effect of workload types and Service Level Agreements (SLAs, e.g. response time) in terms of power characteristics. Base on these, we introduce classified power capping for different services as the control reference to maximize power saving when there are mixed workloads.

Keywords—Power Budgeting; Logical Level; SLA; Classified Power Capping; Cloud Computing; Green Cloud.

I. INTRODUCTION

Power management is becoming more and more important for modern computer systems, especially for large-scale systems composed of thousands of servers. The advances in micro architecture and fabrication technologies have lead to increased performance capabilities in modern computing platforms, which are being accompanied by a dramatic rise in the power densities exhibited by hardware [2]. Over-consumption of power causes many problems, like electricity consumption, bill cost, and even system failures. As a result, there have been various power saving techniques in response to the increased importance of power saving ([e.g. 4, 6, 12, etc.]).

Cloud computing is another important area of modern computer systems. As a promising super structure for corporate, enterprise and academic computing, cloud computing has the attention of many big companies (e.g. Google, Amazon). A cloud can be defined as a pool of computer resources that can host a variety of different workloads, including batch-style back-end jobs and interactive user applications [9].

The importance of power saving and promising effects of cloud computing bring about the idea of a “green cloud” [5]. For data centers which implement cloud computing, it’s operators still have to purchase and provision servers and other IT equipment as well as maintain the facilities components, therefore it is important for a green cloud to predict demand at least to some extent so that people may not over-provision (building, hardware, software, and other things) for fear of running out of resources due to an unexpected surge in demand. Power budgeting is therefore rendered for online power management in several aspects [2].

The job of power budget management is to ensure that the aggregate power consumption of the platform does not exceed the specified power budget, or that it adheres to some average over time. Accordingly, power budgeting should be flexible and cover the power consumption satisfying certain Service Level Agreement (SLA, e.g. response time). There has been substantial work previously performed on power budgeting, exploiting hardware management capabilities like processor frequency scaling [6, 12].

Power budgeting is important, where reasons include the following (Karsten Schwan, 2010):

- to meet a datacenter-level power constraint, by imposing constraints on certain sets of resources and on the applications that use them; one purpose is to constrain total power usage; another purpose is to ensure fairness across multiple applications (power proportional behavior is a favorite recent topic for e.g., hadoop codes);

- to constrain the resources used by an application, with power being one of the first level metrics desired by datacenter operators, again, to meet center-level power budget constraints, and then, to also have a better basis for charging models;

- to achieve improvements in total power consumption, a key goal for web companies like yahoo and Google, since their total power bills are starting to approach levels similar to the levels of equipment purchase bills.

Effective active power budgeting could differentiate both homogeneous and heterogeneous datacenter resources and
build application-aware VM budgeting [2]. That’s why we think of workload classification by power budgeting for different services in cloud computing. Besides, existing hardware techniques (e.g. queuing techniques, control method using DVFS and MPC [4] and hybrid means [3]) of power management for current data centers do not address enough fault-tolerance and dynamic performance characterization by workload types. Their control references is based on constant power budget distributed to fixed locations and do not adapt to the dynamic user workloads of the web service of a cloud in varying locations.

In this paper, we make several contributions.

1) First, we try to address the power management problem distributed to dynamic locations from the logical level, and present a tree distribution unit as the framework for power budgeting in cloud computing. It takes care of software and network based on the logical level with varying group behavior.

2) Secondly we develop our power model for measurement along with classified power capping for different types of workloads or services in order to apply a feedback control to the power consumption of a web service. Classified power capping outperforms the uniform power capping by 6.8% based on our simulation.

3) We define logical level and physical level of power consumptions, especially for a case when users are running multiple types of service on the machine.

4) In addition, power cost is addressed with user’s awareness of service and performance level by P2P network logic of the tree network.

II. CONTROL UNIT

A. The basic design of the power control unit

The overall goal of tree is to intelligently manage power budgeting to ensure efficiency, fault-tolerance, and proportional power control in the network environment. Take video stream for example, we use a distribution tree (Figure 1) as the power budget unit for the video cloud.

Instead of using even distribution of servers on each level of clusters and groups [4], the tree sets the hierarchy of all servers in a network according to the service type, the importance and priority of the service.

B. Basic operation and system process

The tree model is developed based on the workload or service type, and is usually related to service requirements of the network (e.g. SLA). The cloud service necessitates the building of a network tree for dealing with various types of services, so that a service tree belongs to a certain entity.

Our tree design assumes that the source node has a budget manager in charge of all-level-tree nodes within the network. Each node relates to a server in the network. The process of a tree can be built into a system containing three modules—Optimizer, Controller (for both power and network activity), and Routing, with network connecting them and exchanging information. The system diagram and its detailed function are similar to the dynamic energy tree in data center network [20]. Experimental results of it demonstrate that for datacenter workloads, ElasticTree can save up to 50% of network energy, while maintaining the ability to handle traffic surges. This is an important basis of our work.

![Figure 1 Example distribution tree. Rectangles are peers while circles are relay servers](image)

C. Building distribution trees

1) Strong server close to the root

A node which has a big power budget will be placed close to the root, due to a hierarchical distribution of the power budget tree with the root as the highest priority consumer of the power budget.

2) Joining of peer

A joining user of a certain service is considered as a peer. For joining of peer, when the current tree is full in its capacity based on its current power budget and performance requirement, we consider three cases:

a) Self-support: Self-support is the state that the peer who joins in the network is asked whether he or she would support the adding of the power budget or new power distribution unit introduced by the user. If the peer agrees to pay, then the power budget is added with self-support.

b) Peer-support: When all the adding peers have the same right with current users in the system and there is no policy like first-come-first-served. In that case, all the peers are asked to support the added budget cost. This is the common case when too many users are sharing a service and everyone is asked to pay the increased fees due to system overload.

c) No support: If no users support the increased cost due to join of peer, then the new user probably has to wait for a free space or every user in the network would have a lowered performance under the limit power budget.

3) Leaving of an inner node/server.
A node leaving the tree could represent two situations: users leaving a certain service and possible server fault or power device break in the network.

When a node leaves the tree, the tree manager will first check if there is any waiting node to add into the service tree and then compare the power budget level of the waiting node to determine the adding position.

The above process is illustrated in Figure 2 and it helps building a fault-tolerance structure. The case of multiple leaving nodes could be viewed as the combination of several single nodes leaving the tree.

D. Communication

Communication is important for having the whole network coordinated with it’s users at different levels under control. Figure 3 illustrates the communication process. This way, the centralized controller gets information from the local power manager or decentralized controller in order to balance the whole distribution tree.

III. PROBLEM AND SOLUTION

A. How to categorize tree?--Bird of a feather flock together

Using a tree distribution, the power budget is related to the service network. Consumers simply buy the service with specific SLA requirement levels. Users with similar power budget levels or service types could be put into the same network (e.g. email tree, gaming tree).

B. How to control the network power consumption of the tree?

By feedback control (Figure 4), we could control the power consumption under the given power budget. Instead of measuring the power consumption of a computer/computers distributed on fixed physical group/enclosure, the control loop measures the power consumptions that belong to a particular service of the whole network, which is built into a tree structure. The mechanism could be made into software system for clouds.

C. How to calculate the power budget considering both the network and the physical basis?

To measure the actual power consumption on the web basis, there are two basic steps, namely Identification, and Calculation. Each node is given its own service ID when it is served in the network. When there is an ID match to the network, the total power consumption of a service (denoted as A) is:

\[ P_{serviceA} = \sum_{i serviceA} P_i \]  

\( P_i \) is the power consumption of node i, which belongs to the service A network.

D. Based on the tree unit, how do we calculate and differentiate between the power consumption on the logical level and that on the physical level, especially for computers running multiple webservices?

This is a key point to understand our work. It is possible that, users or servers could run multiple services at the same time. For example, one user could listen to music while using online documents or while playing online games. Several types of work load from different service clouds could contribute to the power consumption of the single server or user computer. Figure 5 shows an illustration example of users running multiple services. There are three users 1, 2, and 3 running two types of network services--A and B. We define two levels of power consumption types:

**Type 1(physical level):** For the same user or server, the power consumptions including all services, are

- P1=P1,1+P1,2 for user 1,
- P2=P2,1+P2,2 for user 2,
P3=Pc3 for user 3.

**Type 2 (logical level):** The power consumptions for classified types of services are

\[ P_e = P_{c1} + P_{c2} + P_{c3} \] for service A,
\[ P_b = P_{c1} + P_{c2} + P_b \] for service B.

In this paper, we focus on Type 2 power control, which is the most important difference of our work from other existing works on physical level.

E. How to set the model and power capping of the tree?

As user workload affects the power consumption, we are able to derive our power model as the basis for determining a power budget:

\[ E_{node} = k_1 N_e + k_2 \]

(2)

\( E_{node} \) is the energy consumption of all nodes in the network tree. \( N_e \) is the number of nodes in the tree. From (2), we could see that more workload nodes lead to more energy consumption. \( k_1 \) relates to the workload power consumption on an individual data center. For the same network distribution, \( k_2 \) relates to the basic network power consumption, and could be viewed stable for a certain type of service, especially when the number of working nodes is significantly large.

Note (2) assumes that all nodes are consuming the same wattage, this applies only to one static type of service, for multiple types of service, the formula change to:

\[ E_{node} = k_1 N_{e1} + k_2 N_{e2} + \ldots + k_n N_{en} + k_{minimal}. \]

(3)

**Classified Power Capping** For single datacenter, the power consumption has the relationship as follows:

\[ P = P_{idle} + P_{service} \]

(4)

In order to have effective feedback control, a control target is needed to provide measure and feedback from actual power consumption. Power capping is introduced as a very promising method to save power and avoid outage cases [19]. From (4), we could see it is better to save power for idle status when there is no service workload running. By DVFS (dynamic voltage/frequency scaling), we could switch power capping from busy status to idle status with a lower level value than normal. We introduce the **Classified Power Capping** on the logical level, which sets different power cappings for classified types of workloads/services.

Later, we try to demonstrate that the classified power capping works better based on tree distribution on logical level in saving power and compare three cases of power capping.

IV. IMPLEMENTATION AND EXPERIMENTS

A. Implementation

In this section, we implement power simulation using defined metrics and provide evaluation of power saving by

![Implementation flow chart](Figure 6)

B. Experiments Methodology

In this paper, we propose three cases of simulation to verify the effectiveness of classified power capping on the logical level (represented by case 1). Calculation of Energy saving results compared to case 2 are calculated and plotted to exploit the classified power capping’s improvements.

- Case 1: Classified power capping according to different service workload type.
- Case 2: Uniform power capping with no frequency or power capping scaling for idle status.
Case3: Uniform Power Capping with power capping scaling for idle status

![Figure 7 Three cases of power capping](image)

Figure 7 show the power savings of three cases using different ways of power capping for power budgeting.

C. Experiments of Overall Energy Consumption

Simulation results are shown in Table I. Figure 8 shows the overall energy savings measured by Power*time. The right graph of it shows the energy consumption between cases for comparison and its data are calculated by:

\[ \text{Energy Saving} = \frac{\text{Power}_{\text{Case1}} - \text{Power}_{\text{Case2}}}{\text{Power}_{\text{Case2}}} \]  \hspace{1cm} (5)

As shown in the results, the energy saving in Case 1 of classified power capping is the highest. For the simulation in this paper, the energy saving of case 1 at the logical level exceeds that without classification by 6.8%. Since it is on percentage level, this number is significant when the basis power consumption is large.

<table>
<thead>
<tr>
<th>Case</th>
<th>Energy cost percentage</th>
<th>Saving Percentage/Case 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>0.7073</td>
<td>Case1-Case2 0.2927</td>
</tr>
<tr>
<td>Case 2</td>
<td>1</td>
<td>Case3-Case2 0.2245</td>
</tr>
<tr>
<td>Case 3</td>
<td>0.7755</td>
<td>Case1-Case3 0.0682</td>
</tr>
</tbody>
</table>

**TABLE I. ENERGY CONSUMPTION COMPARISON**

![Figure 8 Energy-Consumptions Percentage and Energy-Savings Comparison for Three Cases](image)

D. Analysis of Experimental results

Power capping could save power during outages and meet the power requirement at certain times [1]. From the results, power capping with idle status scaling is better for saving more power during an idle status of CPU. Although CPU computes 30% energy of the total power usage by computer, it is significant when user workloads are large in broad scale of cloud. From ElasticTree [20], we know that network energy is reduced too by the optimum power saving of network tree.

Furthermore, classified power capping outperforms uniform power capping by using different power capping values based on various power consumption needs. This works based on the power consumption on logical level by grouping the same network service or workload together in a tree satisfying certain SLA or performance needs.

Large-scale simulations and network implementations of in real clouds, remain to be done in the future.

V. BENEFIT OF THE TREE DISTRIBUTION

Tree distribution unit for power budgeting has several benefits. First of all, it responds to the dynamic structure of network cloud services and builds a fault-tolerant system. The second fundamental advantage is that the tree distribution could be used to classify the relationship between servers by grouping the same service machine together in the network. Categorization of workload power consumption on logical level in the tree helps choose where to build the cloud services (e.g. server level) and apply classified power capping for optimum power saving of mixed workload servers. User incentives are also taken care of when waiting users being added into the network service are aware of their decision of service quality, so that power budget, performance, and cost are related together.
VI. RELATED WORK

Power management has become critical for large scale server systems and HPC machines. Pervious works for datacenter power saving, use methods like turning servers on and off based on demand [16], dynamic voltage and frequency scaling (DVFS) ([4]), and spare servers [15] to manage power. Recently, coordinated design based on multi-level power management for the datacenters is also proposed [1]. Models for performance [17] or power [4] are built to make use of frequency scaling in different power phases. Most power management methods for datacenters are based on power distribution units (PDU)s and RACKs distributed by fixed locations on physical level (groups or enclosures[1]).

On the other hand, power budgeting techniques include platform level controllers that perform fine grain throttling to meet power limits [18], power budgeting processes using non-uniform allocations across nodes in datacenters[6], and a compensation approach using power budgeting during different periods of varying workload to manage power from a VM-centric point of view[2]. Power capping saves power without affecting performance for I/O intensive workloads [19].

Power aware techniques for cloud computing have already been proposed ([10], [9]). Other methods try to manage a cloud’s workload or use VMware in order to lower power consumption. ElasticTree on datacenter network has already been proved as an effective method for saving power [20]. The network tree is actually proposed to improve video quality in cloud computing for the masses, and this paper is similar to [8] for a P2P network, but differs in that it uses tree distribution for the power budgeting in cloud services.

VII. CONCLUSION

In all, we present a control framework of tree distribution for power management in cloud computing so that power budget can be better managed based on workload or service types. A tree could be built for a particular type of service by network connection. Then, we define two types of power consumption on physical level and logical level respectively and the calculation method of power budget in a tree. Based on these, we introduce classified power capping as the control-reference policy in cloud computing on logical level. Simulation demonstrates that classified power capping sets the power budget of mixed network groups more close to actual power needs based on performance requirements and saves more power. Accordingly, power budgeting using the distribution tree provides promising direction for analyzing power characteristics by workload types, aiming at optimum power management in cloud services in the future.

ACKNOWLEDGMENT

We would like to thank the US NSF for sponsoring this work under grants CCF-0811413 and CAREER CCF-0953946. Additionally, we thank the US Department of Energy for sponsoring us under Early Career Principal Investigator Award: DE-FG02-07ER25747.

REFERENCE