A Power Provision and Capping Architecture for Large Scale Systems

Yongpeng Liu*, Hong Zhu†, Kai Lu* and Yongyan Liu‡
*School of Computer Science, National University of Defense Technology, Changsha, P.R.China
Email: liuyp@nudt.edu.cn
†Department of Computing and Communication Technologies, Oxford Brookes University, Oxford, U.K
Email: hzhu@brookes.ac.uk
‡Information Center, Ministry of Science and Technology, Beijing, P.R.China

***Abstract*—The rapid growth of large scale computing systems imposes a grave challenge to their power management, where power provision and capping is essential. In this paper, we propose a new architecture of power provision and capping to control the power consumption of large scale clusters. In this architecture, performance sensitive computation units are distinguished from those having less impact on system performance. A subset of units is monitored and their operation states are controlled in order to maintain whole system's total power consumption under budget. Two policies are designed and implemented to select the target subset of nodes for power regulation. One policy is state-based, which chooses nodes running the most power consuming job for power regulation. The other is change-based, which chooses those nodes that runs a job whose power consumption increases most rapidly among all jobs. Experiments have been conducted on the Tianhe-1A supercomputer system to evaluate the effectiveness of these power capping solutions. The experiments demonstrated that the new architecture can ensure power usage safety with only a negligible decline of performance, which is only about 2%.**

***Keywords*-Large-scale system; Power capping; Power control architecture; Metrics.**

I. Introduction

With the steadily rising of the demands on computing performance from scientific applications, both the number of compute units in high performance computing systems and the system integration density grow rapidly. Consequently, in the past years, the so-called Moore's law of power consumption has been observed; that is, 'the power consumption of computer nodes doubles every 18 months' [1]. Power management has become a grave challenge to the development and operation of large scale computing systems.

A. Motivation

Large scale computing systems consume a tremendous amount of energy. According to the recent TOP500 list of high performance systems [2], the average power consumption of Top10 systems is 4.55 MW. The peak power consumption of the fastest supercomputer, i.e. the K computer, reaches 12.659 MW, which equals the power usage of a middle scale city. In 2006, US servers and data centers consumed around 61 billion kilowatt hours (kWh) at a cost of about 4.5 billion U.S. Dollars. This is about 1.5% of the total U.S. electricity consumption or the output of about 15 typical power plants [3]. Many data center projects have been cancelled or delayed because of being unable to meet such enormous power requirements.

High density power consumption causes overheating, which leads to problems of the reliability and availability of the system. Based on empirical data from leading vendors, Feng found that the failure rate of a computing node doubles with every 10^oC increase in the temperature [1]. Consequently, energy has also had be spent on cooling. For example, 0.7W energy is spent on cooling in order to dissipate every 1.0W of power consumed by the high performance computer system at the Lawrence Livermore National Laboratory [4]. Due to the positive feed-back loop between temperature and power, a computer chipset with higher temperatures consumes more power while running identical computations at the same performance state [5].

Huge construction costs of large scale computer systems are also incurred in order to accommodate the huge amount of energy demand. Statistical data from the American Power Conversion show that 63% of the infrastructure cost is used for power supply and cooling [6]. To save the cost in both construction and operating, computer centers hope to reduce the power provision capability and power budget to such systems. Fortunately, the probability of synchronized power spikes on all components of all nodes in a large scale cluster is zero because of its low resource utilization [7]. There is a clear gap between the maximum power actually used by a large group of nodes and their aggregate theoretical peak usage. It is reasonable to limit the power budget to a level that is lower than the theoretical peak power demand. However, an effective power management solution is required to maintain the safety of system power usage. This observation has led to a major area of concern for researchers and leading vendors of high performance computing systems [8].

In summary, power management is essential for large scale computing systems. It is not only concerned with saving energy, but also determines the economic viability of their construction and operation. Moreover, it is closely related to the reliability and safety of the system. It is what this paper is concerned with. In particular, we will propose

an architecture for power capping according to the limit of power provision capability.

B. Related Works

Power capping is to control the maximal power consumption during the operation of a system, which means keeping the power usage of the system below a certain threshold at any execution time. In general, given a total power consumption budget, power capping techniques consist of power sensing followed by power throttling. Sensing detects the power consumption state of the system. Throttling changes the operation states of the nodes in the system, thus keeps the system running at a safe level of power consumption [9]. The goal of power capping is to distribute the total budget of power consumption to the nodes in the system to achieve the highest possible system performance.

Typically, power budget is allocated in a two-level structure such as in Femal *et al*.'s system [10]. The cluster-level power manager dynamically collects the power consumption information of all nodes and assigns a power budget to each node to ensure that the total power is within the budget. The node-level power manager then allocates its power budget to each device in the node, making sure that its power expenditure is beneath its local threshold.

In clusters, multiple nodes may couple together to run an application and therefore their power states should be managed coordinatedly. Noticed this phenomena, Wang *et al*. [11] proposed a multi-input-multi-output (MIMO) algorithm to control the power consumption of multiple servers simultaneously. In each control cycle, the controller collects the power state and CPU utilization of each server, computes a new CPU frequency for each processor, and directs each processor to change frequency in a coordinated way.

Power capping inevitably affects system's performance. How to control system's power consumption with minimal effect on performance is the key issue of power capping. This is addressed by Ranganathan *et al*. [7] by taking into consideration of service level agreement (SLA). They deployed a management agent on each server to monitor the local power consumption. A global controller periodically collects local readings from all agents and estimates the total power consumption of the cluster. If the total consumption exceeds the predefined power budget, the controller determines which server to throttle based on SLA.

It is worth noting that in the existing approaches to power capping for large scale computing systems, all nodes in the cluster are considered as of the same importance indiscriminately. In such a power capping architecture, all nodes are under the same rules of power control. However, in most uses of large scale systems, some nodes are more important to the system's performance while the others have less impact. Therefore, sensing and throttling the states of such nodes is a waste of computation resources. In this paper, we propose an architecture that distinguishes performance important nodes from others and demonstrate that regulating a subset of nodes' power states is sufficient to maintain the total power consumption below safe threshold. By doing so, we can achieve minimal effect on system performance.

Metrics are essential to quantitatively measure and thereby evaluate the efficiency of energy consumption and the effectiveness of power management techniques. There are many metrics of power consumption proposed by researchers for various purposes [8]. For example, the metric $E \times D^n$ [12] combines energy consumption with performance. $FLOPS/W$ is employed by Green500 [13] as a measurement of energy efficiency w.r.t. computation capability. *Total Cost of Ownership* (*TCO*) [1] reflects the total cost of the system, including construction, operation and maintenance. *Power Usage Efficiency* (*PUE*) [14] estimates the energy efficiency of data centers in operation, and so on. These metrics focus on the energy efficiency of computing systems, but neglect the effect of power overload on system. Thus, they provide little guidance to the selection of the nodes for power capping. In this paper, we define a new metric that measures the damage of overspending power budget over a period of time.

C. Organization of the Paper

The paper is organized as follows. Section II presents our architecture of power management. Section III gives the key algorithms of our power management system. Section IV discusses the various policies to select the target set of nodes for power regulation. Section V reports the implementation of the power management system, the experiments with the system, and the evaluation results. Section VI concludes the paper with a brief discussion of future work.

II. The Proposed Architecture

In this section, we outline the proposed architecture, which is depicted in Figure 1.

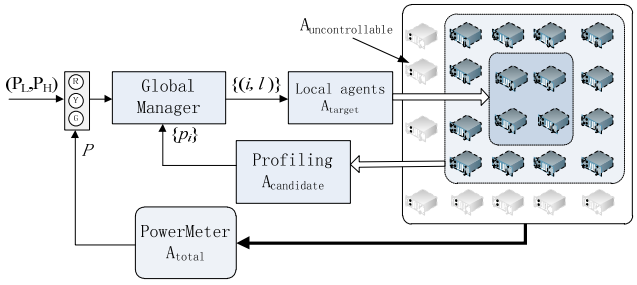


Figure 1. Proposed power capping architecture

A. Classification of Computation Nodes

The architecture is proposed for power management of large scale computing systems. As mentioned in Section I, not all nodes in a large scale computing system should be

considered as the target of power regulation. For instance, some nodes may have no dynamic power management facilities [8]. Some nodes may be running tasks that are urgent, or of high priority in real-time systems, or critical to the system's performance and the response time to service requests. Thus, their degradation will have a significant impact on system's performance, even cause violation of SLA. They should not be degraded. There is no point to monitor their power consumptions, and to waste computation resources to collect their execution states and calculate their power usages. Such nodes (or devices) are regarded as privileged nodes, and regard as uncontrollable in our architecture.

Moreover, in a large scale system, there are a huge number of nodes. Monitoring all nodes for power management could be too costly to be practically usable.

Therefore, in our architecture, we recognize the following sets of computational nodes.

- *Total set* (A_{total}): It contains all nodes in the system that consume power budget. It is the complete set of nodes recognized by the power management system.
- *Privileged Set*: ($A_{uncontrollable}$): It is the set of nodes that are uncontrollable, either because they have no facility to realize power management, or must not be degraded for its vital impact to system's performance.
- *Candidate Set* ($A_{candidate}$): It contains the nodes that are subject to power throttling when need to. It is the set of nodes that equals $A_{total} - A_{uncontrolable}$. The set $A_{candidate}$ may vary during the execution of the system since the tasks running on a node may change. The power management system samples the power states and other execution characteristics of the nodes in this set, and selects a subset of the node to change execution states in order to control the power consumption of the whole system.
- *Target Set* (A_{target}): It is a subset of the candidate set and contains the nodes whose power states will be throttled in a power control loop. It may vary from time to time and determined by the power manager according to the state of the system and a target selection policy.

B. Power Consumption States

Our architecture of power capping for large scale systems employs two thresholds of system's total power consumption: P_H and P_L, where $P_L \leq P_H$

According to these two threshold values, system's power consumption states can be classified into the following three.

- *Green State* (*safe*): The system's power consumption is below the threshold P_L. Thus, it is perceived to be safe and there is no need to reduce system's power consumption.
- *Yellow State* (*warning*): The system's power consumption level is between P_L and P_H. It is still within the power provision, but regarded as too close to the upper limit. Thus, actions must be taken to reduce the power consumption to ensure the system's safety.
- *Red State* (*critical*): The system's power consumption is above P_H. It is considered as critically far too high and it is vital to enforce maximal strength of throttling on the system immediately to prevent permanent physical damage to the system due to overspending the power budget. Without a definite action immediately, the system will run a big risk of physical damage.

By employing two thresholds, we can balance between performance requirements and safety requirements by maintaining the system below (but close to) the threshold P_L yet giving sufficient space to remain in safe without go to the red area from yellow area. Therefore, it is very important to set the right values to these two thresholds. This issue is addressed in subsection III.A.

C. Power Profile Model

The total system power consumption is the key factor to decide whether to throttle system power and the corresponding throttling strength. Thus, we need a high precision system power sampling. This can be achieved by using an appropriate power meter of the whole system. It can be used to directly monitor the real time system power usage.

However, it is sometimes infeasible to fit one power meter for each node in a large scale system because the system may have hundreds and thousands of nodes. Consequently, a power profiling model is needed to indirectly estimate the power usage of each node based on the node's operation mode. We deploy a profiling agent to each node in the candidate set $A_{candidate}$, to profile its local operation state.

In general, the power consumption of a node is composed of two parts, i.e. *static power* and *dynamic power* [8]. For a node with multiple power states, its static power varies according to its power state. Its dynamic power depends on its resource usage. The total power consumption $P(l)$ of a node at power state l can be calculated from its operation mode approximately according to the following formula.

$$P(l) = P_{idle}(l) + Uti_{CPU} \times \sum_{x \in CPU} P_x(l) + \frac{Mem_{used}}{Mem_{total}} \times P_{mem}(l) + \frac{Data_{NIC}}{\tau \times BW_{NIC}} \times P_{NIC}(l) \quad (1)$$

where, for each power level l,

- $P_{idle}(l)$ is the node's power usage when it is idle, i.e., its static power at level l;
- $P_{cpu}(l)$ is the maximal dynamic power consumption of a CPU unit at the corresponding power state, which is the gap between its maximal power and idle power of the CPU unit;
- $P_{mem}(l)$ and $P_{NIC}(l)$ are the maximal dynamic power consumption of the total memory devices and the communication device, respectively;

- Uti_{cpu} is the CPU utilization; the sum on all CPU units is used to obtain the power usage of all CPUs in a node;
- Mem_{used} is the size of memory that is used by the software running on the node;
- Mem_{total} is total size of memory devices in the node;
- τ is the sampling interval;
- $Data_{NIC}$ is the size of total data transmitted through the communication device within the sampling interval;
- BW_{NIC} is the bandwidth of the communication device; thus, $\tau \times BW_{NIC}$ gives the maximal data size of the communication device during a sampling interval.

This power profiling model estimates the power consumption of a computing node based on the operation states of its three key types of devices: CPU, memory and communication device.

D. Underlying Assumptions

Given a large scale system, we assume that the above architecture of power management is applicable and the system is well configured. This assumption is articulated by the following properties.

Controllability: The set $A_{candidate}$ consists of a large enough number of computation units whose power consumption can be controlled (i.e. the number of power consumption levels l is greater than 1) such that when all the nodes in $A_{candidate}$ are set to their lowest power consumption levels, the system's power consumption will certainly be lower than the power provision capability.

All large scale computing systems today consists of a large number of computation units. Each unit may contain a number of different components or devices that can be executed in several different operational modes/states, where operating in different states have different levels of power consumption. Thus, under this assumption, power capping can be implemented by controlling the power consumption at different component levels, such as at device and node levels, which eventually effect on the whole system's power consumption. This assumption is valid as most commodity computing devices provide facilities for power control. Therefore, it is always possible to select a subset of such nodes that satisfies the requirement of controllability.

Observability: The system's total power consumption can be measured directly, and for each node in the set $A_{candidate}$, its power consumption can be measured directly or estimated to a sufficient accuracy in real-time during system's operation.

This assumption is valid because a power meter for the whole system is easy to implement. And, a computing node's power consumption can be estimated from its execution states accurate enough for power management.

Necessity: Power provision capability P_{Max} is less than the theoretical maximal power consumption P_{thy}.

Here, *power provision capability* P_{Max} is the designed capability of power supply subsystem, i.e. the maximal power consumption that the system can afford. In theory, if the number of nodes in the system is N and the maximal power consumption of each node i is P_i, the maximal power consumption of the system P_{thy} is

$$P_{thy} = \sum_{i=0}^{N-1} P_i,$$

which is known as the *theoretical maximal power consumption*. The assumption states that $P_{max} < P_{thy}$. Therefore, it is necessary to manage the system's power consumption for the safety of the power supply subsystem.

This assumption is valid, because it is widely recognized that the probability that all nodes run at their maximal power at the same time is very small. To reduce the construction cost and the operation cost, power supply subsystem is often less capable than the theoretical maximal power consumption.

Operability: The power provision capability is high enough so that the system can function normally without go over the power provision. It is well designed so that it is only occasionally required to throttle system's power to deal with the power spikes.

Here, we assume that, although the power provision capability P_{Max} is often lower than the theoretical maximum, the capability is not too low so that the system is still operable within the power provision. In other words, the power provision is not ridiculously low. In a real environment, the applied power budget should be able to satisfy the power provision in most cases; the throttling of system power is used to deal with the occasional power spikes. Therefore, we believe that this assumption is reasonable and should be valid for most systems.

III. Power Management Algorithms

In our architecture, the power management system invokes three key algorithms to maintain system's power consumption in a stable and safe state. These algorithms are presented in this section.

A. Overall Power Management Cycles

The key algorithms in our system are: (a) threshold setting and adjustment algorithm, (b) power capping algorithm, and (c) candidate set selection algorithm.

The system uses two thresholds P_L and P_H of power consumption level in the power capping algorithm. These thresholds are configurable. They can be set manually by the system administrator based on his empirical knowledge. However, to ensure the effectiveness of power management,

here we propose a simple learning algorithm to set and adjust these parameters. In particular, the system is first run for a relatively long period (say for 24 hours) as the training period with P_L and P_H set according to the following formulas, where the initial value of P_{peak} is set to be the value of P_{max}.

$$P_H = (1 - 7\%) \times P_{peak} = 93\% \times P_{peak}$$
$$P_L = (1 - 16\%) \times p_{peak} = 84\% \times P_{peak}$$

During a training period, the maximal power of the system is recorded and used as the P_{peak} to adjust the next value of P_L and P_H also using the above formulas. After the training period, the observation of the peak power consumption continues through the whole execution period of the system, and the adjustment takes place periodically for every t_p control cycles, where t_p is relatively large so that adjusting the thresholds P_L and P_H is much less frequent than the invocations of the power capping algorithm.

The particular parameters used in the above formula are based on Fan *et al.*'s work [9], which reported that there is a clear gap of 7% - 16% between achieved and theoretical aggregate system power in large scale systems even in the executions of well-balanced applications.

The set of candidate nodes is also a configurable parameter of the power management system. The parameter can be set manually and also be adjusted during the execution of the system according to the impact of the nodes' performance on system's performance as well as the existence of power management facility on the hardware. Details are omitted for the sake of space.

The power capping algorithm employs a target set selection policy to determine the set A_{target}. And, for each node in A_{target}, it assigns a new power state level l to the node. Therefore, the output of the power capping algorithm is a set of ordered pairs (i, l), where i is a node in A_{target}, and l is its target power state level.

The target selection algorithm is periodically invoked by the power management system. If A_{target} returned by the algorithm is not the empty set $\emptyset$, the power manager will send commands to all nodes in the A_{target}, and tell them to regulate their power state to the corresponding target level.

B. Power Capping Algorithm

The power capping algorithm is given in Figure 2. It has the following properties.

1) It is applicable to both heterogeneous and homogeneous systems as far as the power states of a node are discrete, and there are only a finite number of different power levels for each node.
2) It regulates the power states of all nodes in the system synchronously.
3) A timer ($time_g$) is used to record how long the system is in the green state continuously. If the system is in the green state continuously longer than a predefined value T_g, we regard the system as in a *steady green* state. Therefore, the power consumption budget of some degraded nodes can be increased.
4) If the system power is greater than P_L and less than P_H, the system power consumption enters the yellow warning state. A subset of the candidate nodes is selected to decrease the power budget by one level. This mild policy can avoid the heavy effect on system performance by over adjusting. Note that a node in its lowest power state cannot be degraded any further. A valid target set selection policy shall not select an idle node as a target.
5) If the system power consumption enters the red critical state, power management system regards the safety of system power usage as of the highest priority. It pulls the system power consumption down to the safe threshold as quick as possible. All candidate nodes are thus degraded into their lowest power states. Under the assumptions of controllability, this will certainly bring the system back to the safe state. After the system stays in the safe green state for a period of T_g, the system will gradually increase the power budget of the candidate nodes to improve its performance. This allows the system to cool down after overheating due to overspending power budget.

It is worth noting that the effectiveness of the algorithm depends on the target set selection policy. The following section describes two different types of such policies and discusses the rationale behind them.

IV. Target Set Selection Policies

We have identified two types of target set selection policies: state-based policies and change-based policies. The former selects a subset of candidate nodes according to the current state of power consumptions. The latter selects the target set according to the rate of increases in power consumption.

A. State-Based Selection Policies

In a large scale system, especially in a parallel computing environment, multiple nodes are usually coupled to run an application. For a well-balanced application, performance degradation of one node may make this node the bottleneck of the whole system's performance on this application. In other words, the degradation of other related nodes will not cause further loss of system's performance. Throttling all nodes within one application has the same effect on system performance as throttling a single node. However, the former have more effect on system's power consumption than the latter. Therefore, it is reasonable to select the set of nodes on which a job is running as the target set A_{target}.

Algorithm 1. (*Power Capping*)
Variables:

- P: Current system power consumption level.
- $A_{target} = \{(i, l_i)\}_i$: The target set of nodes with target power states.
- $A_{degraded}$: The subset of $A_{candidate}$ that have been degraded. Its initial value is $\emptyset$ when the system starts.
- $Time_g$: A timer recording the length of the period in which the system is continuously in green state. Its initial value is 0.

Begin
  A_{target} := $\emptyset$;
  if $(P < PL)$ {*i.e., in the green state*} **then**
    $Time_g$++;
    if $((Time_g \geq T_g)$ **and** $(A_{degraded} \neq \emptyset))$ {*steady green*} **then**
      A_{target} := $A_{degraded}$;
      for all $(i, l_i) \in A_{target}$ **do**
        Replace (i, l_i) in A_{target} with $(i, l_i + 1)$;
        if $l_i + 1$ is the highest level for node i **then**
          Remove node i from $A_{degraded}$;
        end if
      end for
    end if
  else if $((P \geq PL)$ **and** $(P < PH))$ {*i.e., in the yellow state*} **then**
    $Time_g$:= 0;
    Select $A_{target} \subseteq A_{candidate}$ according to the *Target Set Selection Policy*;
    for all (i, l_i) in A_{target} **do**
      Replace (i, l_i) in A_{target} with $(i, l_i - 1)$;
      Add i into $A_{degraded}$;
    end for
  else if $(P \geq PH)$ {*i.e., in the red state*} **then**
    $Time_g$:= 0;
    for all i in $A_{candidate}$ **do**
      Add (i, l_i) into A_{target}, where l_i is the lowest power state of node i;
      $A_{degraded}$:= $A_{candidate}$;
    end for
  end if
End

Figure 2. Power Capping Algorithm

The question is: which job should the selection policy target at? There is a variety of choices to answer this question, such as:

- the most power consuming job (or jobs);
- the least power consuming job (or jobs);
- the best fit job (or jobs), whose power consumption is closest to the right amount of power budget cut.

Let J be a job running in the system at the time of target selection. Let $Nodes(J)$ be the subset of non-idle candidate nodes on which the job is running on. The power consumption of a job J, denoted by $Power(J)$, is calculated as follows.

$$Power(J) = \sum_{i \in Node(J)} P(i)$$

where $P(i)$ is calculated according to formula (1).

The *most power consuming job* policy (*MPC*) is to select $Nodes(J)$ as the target set A_{target} for the job J such that $Power(J)$ is the largest among all jobs.

It is worth noting that by reducing the power budgets by one level for the set of nodes in $Nodes(J)$ may not be enough to bring the system into the safe green state. However, through several cycles of power management, the system's state will be brought back to the safe green state gradually.

The power consumption state can be brought back to green faster than targeting a single job, if a number of highest power consumption jobs are targeted at. Figure 3 gives the algorithm for the *most power consuming job collection* policy (*MPC-C*).

Algorithm 2.(*Select target set according to MPC-C*)
Begin
  $Saved$:= 0;
  A_{target} := $\emptyset$;
  A := $\emptyset$;
  Order the jobs $J_1, J_2, \cdots, J_k$ in the system such that for all $i = 1, \cdots, k-1$, $Power(J_i) \geq Power(J_{i+1})$;
  for all $i \in \{1, \cdots, k\}$ **do**
    $Saved$:= $Saved + \sum_{x \in Nodes(J_i) - A} [P(x) - P'(x)]$;
    A := $A \cup Nodes(J_i)$;
    if $(Saved \geq P - P_L)$ **then**
      Exit the for-loop
    end if
  end for;
  for all $i \in A$ **do**
    Add (i, l_i) into A_{target}, where l_i is the current power state of node i;
  end for;
  Output(A_{target});
End

Figure 3. MPC-C Target Selection Policy

In Algorithm 2, $P'(x)$ is the estimation of power consumption of node x when the power budget is decreased by one level. It is calculated using Formula (1).

Similar to the MPC and MPC-C policies, we can apply the *least power consuming job* (*LPC*) / *job collection* (*LPC- C*) policies, which selects the target nodes on which the job/job collection runs and they consume the least power. These policies have the slowest effect on power consumption, but least likely to cause swings of power consumption states between green and yellow.

Another alternative is the *best fit job* policy (*BFJ*), i.e. to select the target set to be the set $Nodes(J)$ such that the power consumption saved by reducing the power budget by one level for each node in $Nodes(J)$ is just above the

difference between P and P_L. This policy can be regard as a compromise between the MPC and the LPC policies.

The above policies have a common feature, that is, they all target at the job(s) according to the current power consumption states of the nodes on which the jobs are running. These state-based policies degrades a set of nodes that are associated to certain job(s) when the system enters the yellow warning state without considering which job or jobs actually caused the increase in power consumption. This type of policies is not fair when the targeted job does not cause the problem. This is especially true when the MPC or MPC-C policies are used. In these cases, a large number of nodes could be affected. An alternative is to target directly at the jobs that cause the increases in power consumption, thus the following change-based selection policy.

B. Change-based Selection Policies

The basic idea of change-based policies is to target the job(s) with the highest rate of increase in power consumption. Let $\Delta P^t(x)$ denote the rate of increase in power consumption of node x at time moment t. That is,

$$\Delta P^t(x) = \frac{P^t(x) - P^{t-1}(x)}{P^{t-1}(x)},$$

where both $P^t(x)$ and $P^{t-1}(x)$ are calculated using formula (1) with the parameters as node x's state at time moment t and $t-1$, respectively.

The rate of increase in power consumption due to job J at time moment t, denoted by $\Delta P^t(J)$, is defined as follows.

$$\Delta P^t(J) = \frac{P^t(J) - P^{t-1}(J)}{P^{t-1}(J)},$$

where $Nodes^t(J)$ is the set of nodes on which job J is executing at time moment t, and

$$P^t(J) = \sum_{x \in Nodes^t(J)} P^t(x).$$

Similar to the MPC policy, we can define the *highest rate of increase in power consumption* policy (*HRI*) by targeting at the job whose value of $\Delta P^t(J)$ is the largest among all jobs. It selects the set $Nodes(J)$ as the target set such that the $\Delta P^t(J)$ is the largest among all job.

Using HRI policy, the job directly targeted at for degradation is probably the cause of moving power consumption level above the green line P_L. It is fairer because it punishes the job that cause problem and balances the effect among all nodes of the system. However, the number of target nodes may be less than that of MPC policy. Therefore, using this policy, the power reduction in each control loop usually is less than that of MPC policy. It may be slower to pull the system back to the safe state.

Similarly, as a counterpart of the policy of the most power consuming collection of job, we can also define the policy that selects the collection of jobs that are of highest rates of increases in power consumptions. However, it does not make sense to define the counterparts of the other state-based policies.

V. Implementation and Evaluation

The proposed power management system has been implemented and deployed to an experimental supercomputer environment. It is a variant of the supercomputer Tianhe-1A, which is currently ranked as No. 2 in Top500 list and was ranked as No.1 in the previous Top500 list. Experiments are been carried out in this environment. This section reports the implementation and experiments with the system.

A. Hardware and Software Platform

In the environment, each compute node is constructed on one Tianhe-1A main board. Each node has two Intel Xeon X5670 processors, and each processor contains 6 cores. Each processor is configured with 6 DDR3-133 memory devices. The capacity of each memory device is 4GB. These two processors are in the SMP architecture. A Tianhe-1A high speed communication chipset is embedded in the mainboard. There are 128 nodes in our environment that are connected by Tianhe-1A network through this communication chipset.

The Intel Xeon X5670 processor in the experiment environment supports Dynamical Voltage and Frequency Scale (DVFS) mechanism. It can operate in 10 different working frequencies from 1.6 GHZ to 2.93 GHZ. Lower frequency means lower compute capability, and also lower power consumption. It is the only hardware mechanism in our experiment environment that facilitates power management. We control the power state of a node by regulating the working frequency of its processor cores synchronously. The power consumption of all other devices is indirectly managed in the experiment through decrease the power consumption level of the processors. Each level of node power degradation is implemented by decreasing one level of processor frequency. If its processor core is working at 1.6 GHZ, we consider the node is in lowest power state.

The operating system running on our compute node is Linux. The parameters in our power profiling model, i.e. the values of Uti_{cpu}, Mem_{used}, Mem_{total} in formula (1), are obtained by sampling the devices through invocations of the /proc interface provided by Linux kernel. The value of $Data_{NIC}$ is obtained from the automatic log generated by Tianhe-1A communication chipset.

B. The Evaluation Benchmark

Our experiments used five parallel applications in the *MPI benchmark suite* in the *NAS Parallel Benchmarks* [15] as evaluation jobs. They are EP, CG, LU, BT and SP. In the experiments, their CLASS configuration is set as D. The NPROCS parameter of these jobs varies from 8, 16, 32, 64, 128 to 256.

C. Execution of the experiments

In the experiment, the thresholds P_L and P_H were learned by training executions as described in Section III. The length of training period is 24 hours. During the initial training period, all nodes are running at highest power state without any power management.

The parameter T_g in the power capping algorithm is set to be 10 cycles. Two target selection policies has been implemented and used in the experiments. They are MPC and HRI policies. After the initial train period, the experiment system is executed with one of the target selection policies for 12 hours and the power consumption behavior of the system is measured and recorded.

In the experiment, evaluation jobs were generated at random by first selecting one application from the benchmark, and than set the NPROCS parameter at random to be one of the values 8, 16, 32, 64, 128 to 256. An evaluation job is added to the job queue whenever the queue is empty. They were loaded to the system as soon as the required hardware resource is available. Therefore, at any time during the experiment there may be multiple jobs running on the system and each job have a number of processes.

During the experiments, power consumption and performance behavior are measured and data collected. Two measurements of system's performance are used in order to analyze and compare the effectiveness of our power management architecture.

1) System performance: The performances of the system under different power management policies are calculated according to the following formula:

$$Performance(cap) = \frac{1}{J}\left(\sum_{j=1}^{J}\frac{T_j}{T_{cap,j}}\right),$$

where

- cap is a selection policy used in power capping;
- J is the number of finished jobs in the experiment;
- T_j is the time to finish the job j with highest node performance without any power capping;
- $T_{cap,j}$ is the time to finish job j using a selection policy cap.

Using this performance metric, the greater measurement means the higher performance of the system, while the less means larger performance loss.

2) Count of performance lossless jobs (CPLJ): This metric count the finished jobs whose execution time under power management is equal to the time to execute the job when the system is in full power, i.e. without power management. Therefore, the higher in this measurement means the higher system performance when its power is managed, and the less performance loss.

3) The maximal power (P_{max}): It is the peak value of the power consumption observed during the execution of the system.

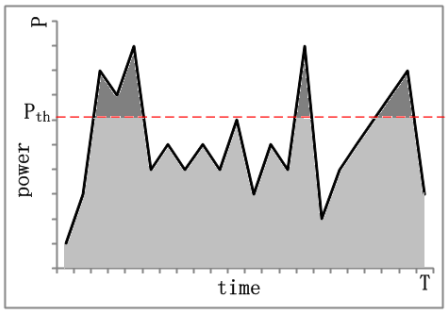


Figure 4. Accumulative effect of overspending

To evaluate the effectiveness of a power management system, we also define the following metric.

4) Accumulative effect of overspending ($\Delta P \times T$): This metric is defined by the following formula.

$$\Delta P \times T = \frac{\int_{p>P_{th}} (P(t) - P_{th})dt}{\int P(t)dt},$$

where

- $P(t)$ is the system's power consumption at time t,
- P_{th} is the power provision capability, i.e. the threshold of power consumption required by the system.

The meaning of the formula is illustrated by Figure 4, where the curve represents the power consumption behavior $P(t)$; the red dashed line represents the system's required power consumption threshold. The dark grey area is the effect of power overspending during the whole execution period from time moment 0 to T. The total grey area including both dark grey and light grey areas is the total amount of heat generated by the execution of the system due to consuming the power. The $\Delta P \times T$ metric calculates the ratio of the dark grey area over the total grey area. It evaluates the effect of power over consumption on a system during the whole execution period.

Therefore, informally, the metric calculates the relative effect of power consumption that is above the required threshold P_{th}. It reflects the accumulative thermal impact caused by overspending power budget, i.e. the extra heat generated by the overspent power during the whole execution period over the total heat generated. This metric captures our intuition that the effect caused by power overload is related to the extent of power consumption above the threshold and the length of the time overspending happened.

D. Experiment Results

To select target nodes wisely, the global power manager need to collect information about the runtime behaviors and the power consumptions of all nodes in the candidate set. The cost of central power management rises with the number of nodes to be monitored. Figure 5 shows that the CPU utilizations of the central management node increases

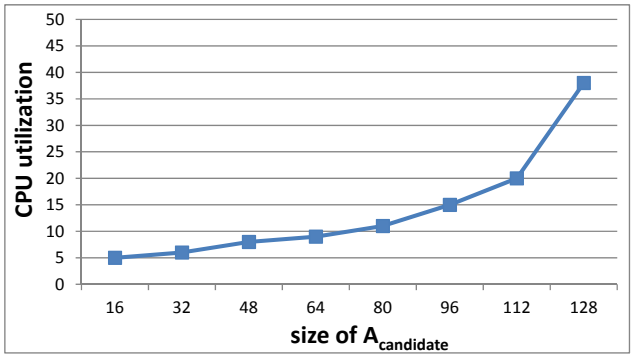
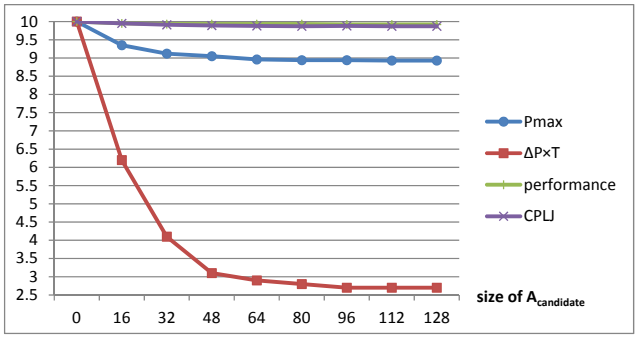
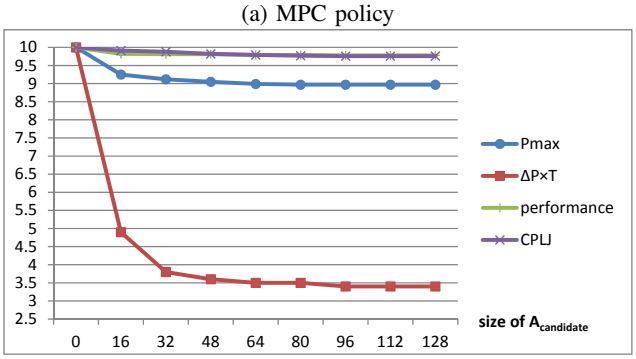


Figure 5. Scalability of global manager



(a) MPC policy



(b) HRI policy

Figure 6. Power capping effect at different size of $A_{candidate}$

non-linearly with the sizes of $A_{candidate}$. It indicates that scalability may become a problem in power management for large scale systems. It is indeed necessary to only profile a subset $A_{candidate}$ of nodes rather than all the nodes in cluster.

With different sizes of the candidate set, we observed the effect of the power capping in the experiment. The results of the experiments are shown in Figure 6, where the values are normalized against the values when the system is executed without any power management (i.e. when the size of $A_{candidate}$ is 0).

From the results shown in Figure 6, we can see that the trend curves of MPC are similar to those of HRI. Both

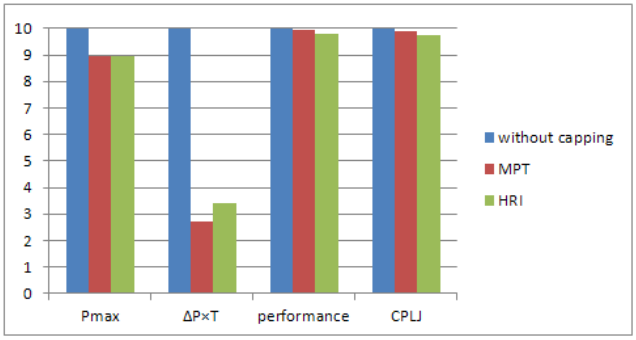


Figure 7. Power capping results of different policies

the maximal power metric and the accumulative effect of overspending metric show that the more nodes covered by $A_{candidate}$, the better effect our power management solution has. However, as discussed above, the larger $A_{candidate}$ means more jobs can be throttled by power capping, and thus incurs more management cost. However, it is worth noting that the increase of effect with the size of $A_{candidate}$ is not linear. When $A_{candidate}$ is large enough (48 nodes in our experiment environment), the increase in effect of power capping diminishes. In this situation, adding more monitored nodes returns little effect but a remarkable increase in cost, as shown in Figure 5. Therefore, a power management solution should trade-off between cost and effect by choosing a suitable size of $A_{candidate}$.

As shown in Figure 7, when all 128 nodes are included in $A_{candidate}$, system performance is lost by about 2% due to power capping when use either of the target set selection policies. However, the maximal power is reduced by about 10%. This means that the spikes of power consumption are successfully controlled by the power management system.

Comparing the target set selection policies MPC and HRI, the experiments shown that their effects on performance and the maximal power are nearly the same. However, when the power consumption behaviors are measured by the $\Delta P \times T$ metrics, the difference between these two policies becomes apparent. The power behavior measured in $\Delta P \times T$ is improved remarkably by both policies, where MPC and HRI reduced $\Delta P \times T$ measures by 73% and 66%, respectively. Moreover, using the MPC policy, the number of jobs without performance loss is greater than (by 1.4%) the same measure when the HRI policy is used. Therefore, MPC policy is more favorable than HRI policy.

It is worth mentioning that in our experiments with power capping, system power is always below P_H no matter which selection policy is used. In other words, the system power consumption has never entered the red critical state when the power is managed. This is another evidence of the validity of the proposed architecture.

VI. Conclusion

In this paper, we proposed a new architecture for power management of high performance computing systems. One of its key characteristics is that for each power capping cycle, it selects a subset of nodes from a set of candidates to reduce their power consumption when power reduction is needed. We have explored two types of policies to select this power capping target set of nodes. The first type of policies is state-based policies, which only consider the current state of power consumption by the nodes that execute a job or a set of jobs. The second type of policies is change-based policies, which selects the target nodes by considering the rate of increase in power consumption for executing a job or a collection of jobs. A power management system has been implemented and deployed to the Tianhe-1A environment with two different policies, one in each type. Experiments have been conducted to evaluate the effectiveness of the architecture and to compare the different policies. The results show that in both policies, the system can successfully manage the power consumption with only 2% degradation of system's performance.

For future work, we are implementing other selection policies and conducting more experiments with new policies to compare their power and performance behaviors.

Acknowledgement

This work is supported by the National High Technology Research and Development Program of China (863 Program) under grant No. 2009AA01A128 and the NSF of China under Grant No. 60603061, No. 60903059 and No.60903044.

References

[1] W. Feng, "Making a case for efficient supercomputing," *ACM Queue*, vol. 1, no. 7, pp. 54–64, 2003.

[2] Top 500, URL: http://www.top500.org, Nov. 2011.

[3] US Environmental Protection Agency, "Report to congress on server and data center energy efficiency". Energy Star Program. URL: http://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Datacenter_Report_Congress_Final.pdf, August 2007.

[4] M. Seager, "What are the future trends in high-performance interconnects for parallel computers?" in *Proceedings of the 12th IEEE Symposium on High-Performance Interconnects*, Stanford, California, U.S.A., Aug. 2004, pp. 3–3.

[5] O. Sarood and L. V. Kale, "A 'cool' load balancer for parallel applications," in *Proceedings of the 24th Supercomputing Conference (SC'11)*, Seattle, Washington, USA, Nov. 2011.

[6] American Power Conversion (APC), "Determining total cost of ownership for data center and network room infrastructure," URL: ftp://www.apcmedia.com/salestools/CMRP-5T9PQG_R2_EN.pdf, Dec. 2003.

[7] P. Ranganathan, P. Leech, D. Irwin, and J. Chase, "Ensemble-level power management for dense blade servers," in *Proceedings of the 33rd International Symposium on Computer Architecture (ISCA'06)*, Boston, MA, USA, Jun. 2006, pp. 66–77.

[8] Y. Liu and H. Zhu, "A survey of the research on power management techniques for high-performance systems," *Software - Practice and Experience*, vol. 40, no. 1, pp. 943–964, Jan. 2010.

[9] X. Fan, W. Weber, and L. Barroso, "Power provisioning for a warehouse-sized computer," in *Proceedings of the 34th International Symposium on Computer Architecture (ISCA'07)*, San Diego, California, USA, Jun. 2007, pp. 13–23.

[10] M. Femal and V. Freech, "Boosting data center performance through non-uniform power allocation," in *Proceedings of the 2nd International Conference on Autonomic Computing (ICAC'05)*, Seattle, WA, USA, Jun. 2005, pp. 250–261.

[11] X. Wang and M. Chen, "Cluster-level feedback power control for performance optimization," in *Proceedings of the 14th IEEE International Symposium on High-Performance Computer Architecture (HPCA'08)*, Salt Lake City, Utah, Feb. 2008, pp. 101–110.

[12] P. Penzes and A. Martin, "Energy-delay efficiency of vlsi computations," in *Proceedings of the 12th ACM Great Lakes Symposium on VLSI (GLSVLSI'02)*, New York, NY, USA, April 2002, pp. 104–111.

[13] Green500, URL: http://www.green500.org/, Nov. 2011.

[14] C. Belady, A. Rawson, J. Pfleuger, and T. Cader, "Green grid data center power efficiency metrics: PUE and DCiE," White Paper No. 6, Nov. 2007, URL: http://www.thegreengrid.org/gg_content/TGG_Data_Center_Power_Efficiency_Metrics_PUE_and_DCiE.pdf.

[15] D. Bailey, E. Barszcz, J. Barton, D. Browning, R. Carter, L. Dagum, R. Fatoohi, S. Fineberg, P. Frederickson, T. Lasinski, R. Schreiber, H. S. V. Venkatakrishnan, and S. Weeratunga, "The NAS parallel benchmarks," RNR Technical Report, RNR-94-007, March 1994, http://www.nas.nasa.gov/publications/npb.html.