Adaptive Controller for Dynamic Power and Performance Management in the Virtualized Computing Systems

Chengjian Wen1, Xiang Long2, Yifen Mu3
1 Department of Computer Science and Engineering, Beihang University, Beijing, People’s Republic of China, 2 Department of Computer Science and Engineering, Beihang University, Beijing, People’s Republic of China, 3 Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, People’s Republic of China

Abstract

Power and performance management problem in large scale computing systems like data centers has attracted a lot of interests from both enterprises and academic researchers as power saving has become more and more important in many fields. Because of the multiple objectives, multiple influential factors and hierarchical structure in the system, the problem is indeed complex and hard. In this paper, the problem will be investigated in a virtualized computing system. Specifically, it is formulated as a power optimization problem with some constraints on performance. Then, the adaptive controller based on least-square self-tuning regulator (LS-STR) is designed to track performance in the first step; and the resource solved by the controller is allocated in order to minimize the power consumption as the second step. Some simulations are designed to test the effectiveness of this method and to compare it with some other controllers. The simulation results show that the adaptive controller is generally effective: it is applicable for different performance metrics, for different workloads, and for single and multiple workloads; it can track the performance requirement effectively and save the power consumption significantly.

Introduction

In the past years, more and more interests have been paid to the power and performance management problem in the computer science. The problem is definitely important especially when we realize how much power have been consumed by the data centers worldwide in one year [1]. In this paper, we will study the problem in the virtualized computing system, as the virtualization technology provides an important approach to save energy consumption for a single machine [2] and data center [3]. Based on the adaptive control theory, we will use the least square self-tuning regulator to track the performance, so as to maintain the power consumption at the lowest level which satisfies the performance demand. And we will show that this method is generally effective for different performance metrics, for different workloads, and for single and multiple workloads via simulation.

Although the problem is important undoubtedly, it is indeed complex and hard. This is because of multiple objectives (performance, power, load balance, etc.), multiple influential factors and the nonlinear relationship between them. Literature from the industrial and academic community tried to solve the problem from different aspects. A review on energy-efficient algorithms can be seen in [4].

From the perspective of modeling method, previous work can be classified into several categories as the following: (1). to optimize one objective given constraint on another one. This kind of work might optimize the performance under a power budget, or track the load balance between virtual machines, see [3] [5–6] etc.. For example, [3] tracks the utilization using the model predictive control based on different prediction algorithms. [5] considers the correlation between the load balance, the performance, and power and designs a two-layer control structure: first to control the load balance between the virtual machines to track a same performance level by a multi-input-multi-output control approach; then [5] manipulates the frequency. (2). to optimize a new objective, which might integrate several objectives such as the performance, the power, or the balance between different machines, see [7–8] etc.. [7] formulates the control problem as a profit maximization problem by integrating the SLA function, which represents the performance, the power consumption, and switching cost. Then two-level control hierarchy is introduced, where one level is a faster control, the other level is a slower control. [8] defines a cost function which integrates the performance and power consumption.

From the perspective of regulating method, previous work can be classified into several kinds too: (1). via control theory, like the optimal control, see [3] [5–8] etc.. Multiple kinds of controller are designed, such the feedback controller, the optimal controller, the Proportional-Integral-Derivative (PID) controller, the Model-Predictive Controller etc.. For example, [3] compares the results by different controllers and find that the predictive controller performs better with some self-learning behavior. [6] uses the PID controller and model predictive controller at the same time, (2). via the heuristic algorithms, see [9–10] etc.. For example, [10] defines a cost function for the long run, which includes both of the reward and penalty in the future, then [10] uses the reinforcement learning method together with fuzzy rule bases to achieve the...
defined objective. Among these methods, control theory has been applied more and more as it can provide an unified framework and a rigorous controller design. Different from the above engineering aspect, [11–12] investigate the abstraction of load balance problem: the balls into bins, and investigate it from a theoretical point based on a probability framework.

In this paper, we will investigate the performance and power problem in the virtualized environment based on adaptive control theory. To be specific, we will formulate the problem to optimize (minimize) the power consumption, and make sure that the performance satisfies a given requirement. That is quite practical in a real system. Then the adaptive controller based on the least-square self-tuning regulator (LS-STR) will be designed for the first time to adaptively and dynamically track the performance. We will show the effectiveness of this method by simulation, in which, the performance model and the power model will be built based on data collected from real machines. The simulation results show that the adaptive LS-STR is generally effective and has obvious advantages to other controllers: it is very general and flexible and is easy to implement; it can track different performance metrics for single or multiple different jobs effectively, which is better than the predictive controller; it can save power consumption considerably compared to the open loop controllers; and finally, it needs only the input and output information while the system model and the workload info is not required, thus it is applicable for a lot of practical problems.

The paper makes special contributions in the following points:

1. By defining the integrated resource for each applications, the problem can be solved in two steps: the performance tracking and power minimization. Then, to track the different performance metrics, we design the adaptive controller based on LS-STR for general systems, which can estimate the system model and track the performance at the same time.

2. In order to test the effectiveness of the adaptive controller, we design a series of simulations. In simulation, there can be single or multiple workloads, the performance metric can be different too. Then the simulation results show that the adaptive controller is generally effective to track the performance and thus save the power.

Analysis

The system and resource

A typical virtualized computing system can be illustrated by Figure 1, from which we can see a clear hierarchical architecture between the resources.

On the top level, there are $M$ physical machines (PM), denoted as $P Mn$. Each PM has certain number of cores. Assume there are $N$ cores all together, denoted each as $C_n$.

Each core has its feasible frequencies when it is running, which might be different from each other. Usually, the feasible frequencies form an arithmetic sequence, i.e., the frequency can be $0, f_0, g_0 + A_{f_0} \cdot k_0, \ldots, g_0 + A_{f_0} \cdot k_n = f_n$. Generally, $f_0, A_{f_0}, k_0, f_n$ are same for the cores on the same PM, but can be different from cores on different PMs. When the frequency is 0 while the PM is still powering on, the PM is called to be at the idle state, at which the power consumption is positive.

Now the virtual machines can be defined and built on the cores. A virtual machine (VM) is an abstraction of the physical machine, which can be run on a PM and can be migrated between different PMs. Usually, a VM is supposed to carry out a specific kind of service or applications (we call them a job), such as the website request, computing demand and so on. The VM makes it possible to share computing resources on a PM/core among multiple applications and shut down the idle PMs thus can save power consumption.

Suppose $J$ jobs are arriving and waiting for resources. The virtual machine for the $j$th job on the $m$th core can be denoted as $VM_{jm}$, if it exists. To complete the $j$th job on $VM_{jm}$, two factors of a core $C_m$ will mainly influence the performance: the frequency of the core $f_m$ and the CPU share $w_{jm} \in [0,100\%]$ of $C_m$, both of them can be regulated. Obviously, the CPU share satisfy $\sum_{jm} w_{jm} = 1, \forall n$.

To summarize briefly, we can define the resource vector $res_j(t) = (f_{jm}(t), w_{jm}(t)), n$ as the resource allocated to the $j$th job on the $m$th core. And $res_j(t) = (f_{jm}(t), w_{jm}(t)), n$ captures the main influential factors to complete the job.

The above definitions can be illustrated by Figure 1 in detail. In Figure 1, two PMs are in the system with two cores on each PM. So the cores are denoted as $c_1, c_2, \ldots, c_4$. There are two jobs demanding the resource. To complete the first job, the VMs are built on each core and the resource vector for the first job is $res_1(t) = (f_{1,0}, (2f_{1,0}, 2f_{1,0}, 2f_{1,0}, 0.3)).$

Problem statement

In the real problems, the strength of the $j$th job, which is called the workload and denoted as $l_j(t)$, is usually time-varying, stochastic and sometimes is periodic. Thus it is necessary to regulate the resource vector dynamically in order to gain performance and save power.

There are several criteria to represent the performance depending on the property of specific jobs. The most popular criteria in the literature are the response time $r_j(t)$ and the throughput $TP_j(t)$, both of which is measurable in real time. The smaller response time implies the fast processing rate, while the bigger throughput implies the bigger processing capability. Both of them imply a good performance.

The power consumption can be represented by the power value $Pm(t)$ kilo-hour for each physical machine and it can be measured in real time by power meters ([13]).

Generally, a tradeoff exists between high performance and low power consumption: high performance means more energy consumption. In the real problems, we usually place the performance demand prior to the power consumption. For
example, usually we require that \( r_j(t) \leq r_j^* \), or \( TP_j(t) \geq TP_j^* \). In order to save power as much as possible, we try to make the performance to exactly satisfy the requirement. Hence, the problem can be formulated as

\[
\min \sum_{m} P_m(t) \tag{1}
\]

subject to \( r_j(t) = r_j^* \), or \( TP_j(t) = TP_j^* \).

In the real problems, the dynamical regulation of the resource vector \( \text{Res}_j(t) \) can be carried out from bottom level to the top level of Figure 1: first, to regulate the CPU share; second, to regulate the frequency; third, to turn on/off the VMs; and finally, to turn on/off the physical machine. Apparently, it is not easy to solve such a problem.

**Methods**

We will solve the problem (1) by two steps: first, to track the performance; second, to minimize the power consumption. To make this method feasible, we define the integrated resource as

\[
\text{Res}_j(t) = \sum_n f_{a}(t) \cdot \text{vms}_{j,n}(t), \tag{2}
\]

which is a scalar and will be the bridge between the two steps. In the following we will give the solutions of the two steps. First, we will track the performance.

In many cases of real problems, the workload \( \lambda_j(t) \) cannot be measured because of the time delay or some system constraint. For this case, we can apply the adaptive control theory to track the performance. In this section, we will use \( y_j(t) \) and \( y_j^* \) to represent the performance \( r_j(t), TP_j(t) \) and performance requirement \( r_j^*, TP_j^* \).

With the definition of \( \text{Res}_j(t) \), we can build the linear regression model between the resource and the performance as below:

\[
y_j(t+1) = -a_j^1 y_j(t) - \ldots - a_j^p y_j(t-p) + b_j^1 \text{Res}_j(t+1) + \ldots + b_j^q \text{Res}_j(t-q) + v(t+1), \tag{3}
\]

where \( a_j^1, \ldots, a_j^p, b_j^1, \ldots, b_j^q \) are coefficients, \( v(t) \) is the noise which covers the stochastic and other effects. Denote \( \theta_j = (a_j^1, \ldots, a_j^p, b_j^1, \ldots, b_j^q)' \) and \( \varphi_j(t) = (y_j(t), \ldots, y_j(t-p+1), \text{Res}_j(t), \ldots, \text{Res}_j(t-q+1))' \), then (3) can be simplified as

\[
y_j(t+1) = \theta_j^T \varphi_j(t) + v(t+1).
\]

The linear regression model (3) above is a very general model. It models the relationship between the input \( \text{Res}_j(t), t=0,1,2,\ldots \) and the output \( y_j(t), t=0,1,2,\ldots \), and the relationship can be definitely nonlinear. On the other hand, it is a simple model, which make it easy to design the adaptive tracker for the model.

For the system (3), we can design the least square self-tuning regulator (LS-STR) to track the output \( y_j^* \):

\[
\text{Res}_j(t) = \frac{1}{b_j^0(t)} \{ y_j^* + a_j^1(t)y_j(t) + \ldots + a_j^p(t)y_j(t-p+1) \}
- b_j^1(t)\text{Res}_j(t-1) - \ldots - b_j^q(t)\text{Res}_j(t-q+1).
\tag{4}
\]

where \( a_j^1(t), \ldots, a_j^p(t), b_j^1(t), \ldots, b_j^q(t) \) are the estimates of the parameters which are obtained by the iterative least square algorithm as below

\[
\theta_j(t+1) = \theta_j(t) + z_j(t)P_j(t)\varphi_j(t)[y_j(t+1) - \varphi_j(t)^T \theta_j(t)],
\]

\[
P_j(t+1) = P_j(t) - z_j(t)P_j(t)\varphi_j(t)\varphi_j(t)^T P_j(t), \tag{5}
\]

\[
z_j(t) = [1 + \varphi_j(t)^T P_j(t) \varphi_j(t)]^{-1}.
\]

The initial values \( \theta_j(0), P_j(0) > 0 \) can be taken arbitrarily.

From [14], under some weak and natural conditions on \( \theta_j \), the output \( y_j(t) \) is optimal in the sense that the accumulated close-loop tracking error \( R_j(T) = \sum_{t=1}^{T} (y_j(t) - y_j^*)^2 \) holds that

\[
\frac{R_j(T)}{T} \rightarrow 0 \quad \text{with respect to} \quad T \rightarrow \infty.
\]

To make it clear, the idea of self-tuning regulator dates back to 60 years ago[15] and was discussed much in the following, like [16] etc. Then [14][17–19] proved the global stability and optimality of the LS-STR and describe the convergence rate of it. Since it is very flexible and very easy to implement, it has found the application in many fields, such as the steel rolling, paper making, metallurgy, automatic pilot of mammoth tanker, etc. ([19]). This paper can be regarded as a new application of LS-STR.

So far, by defining the integrated resource \( \text{Res}_j(t) \) and build the linear regression model (3), the adaptive controller based on LS-STR (4) (5) can be designed which can adaptively track the reference output \( y_j^* \) by regulating the integrated resource while estimating the parameters on line at the same time.

In the simulation, for all the jobs, we take \( p=q=2 \) base on several trials.

Now suppose we have gotten \( \text{Res}_j(t) \) computed by LS-STR (4). Then there might be multiple \( \text{res}_j(t) \) satisfying \( \text{Res}_j(t) = \sum_{n} f_{a}(t) \cdot \text{vms}_{j,n}(t) \) while they result in different power consumptions. Thus we need to find the ‘good’ \( \text{res}_j(t) \) to minimize power consumption. This can be formulated as an optimization problem below:

\[
\min \sum_{m} P_m(t), \tag{6}
\]

subject to \( \sum_j \text{Res}_j(t) \leq \sum_n f_{a}^n \); \tag{7}

\[
\sum_n f_{a}(t) \cdot \text{vms}_{j,n}(t) \geq \text{Res}_j(t), \forall j;
\]
In (6), $\sum_{j} vms_{j,n}(t) \geq Res_{j}(t)$ since the performance constraint must be satisfied. The solution to the problem (6) will be taken as the solution to the original problem (1).

Experiments

Basic settings

To test the effect of the adaptive strategy to manage the performance and power, we design some simulations with single different workloads, different performance criteria, and multiple workloads.

Suppose in a virtualized computing environment, there are two physical machines, with one core on each machine. For the sake of simplicity, we assume that each core has the same four feasible frequencies: \( f_{i}(t) = 0, 1.6 \text{ g}, 2.2 \text{ g}, 2.8 \text{ g}, i = 1, 2 \). This setting is simple but remains the generality and difficulty of the problem.

Performance model

The performance model will be used to generate the system in the simulation. Meanwhile, the performance model is necessary to design the predictive controller to track the performance.

In Section 3.2, we have stated that there are two typical performance criteria which are usually used in the literature: the response time and the throughput. Both of them will be adopted in the simulation.

To build the performance model, first note that for a given resource \( Res_{j}^{0} \), both the response time and the throughput will encounter a critical value \( \lambda_{j}^{0} \), which can be regarded as the maximal capacity corresponding to \( Res_{j}^{0} \).

Then, concerning the response time of a specific job, when the workload is below the maximal capacity, a liner model can be built to approximate the relationship between the performance, the resource and the workload. When the workload is very large, the response time will increase significantly ([7]).

Suppose the maximal capacity of a core/PM with the resource being \( Res_{j}^{0} (= 2.8 \text{ here}) \) is \( \lambda_{j}^{0} \), the corresponding response time is \( r_{j}^{0} \) (taken as 3 seconds in simulation). Then, with the workload \( \lambda_{j}(t) \) and the resource \( Res_j(t) \), the response time can be roughly modeled as

\[
    r_j(t) = \frac{r_{j}^{0} \cdot Res_{j}^{0} \cdot \lambda_j(t)}{\lambda_j^{0} \cdot Res_j(t)}.
\]

Concerning the throughput, when the workload is below the maximal capacity, all the workload can be dealt with. When the workload is larger than the maximal capacity, then the workload which is beyond the maximal capacity will be abandoned. Thus, the throughput can be modeled as below:

\[
    \sum_{j} vms_{j,n}(t) = 1, \forall n.
\]
It is notable that the maximal capacity \( l_0 \) obviously is related to the special PM and the job's type. And in simulation, we assume the workload is always below the maximal capacity.

Additionally, from the performance model (8) the response time is inverse to the resource \( \text{Res}_j(t) \), so when the response time is considered, we will take \( y_j(t) = \frac{TP_j(t)}{\lambda(t)} \), as the performance output in (3) in order to avoid bad tracking effects. Similarly, when the throughput is considered, we will take \( y_j(t) = \frac{TP_j(t)}{\lambda(t)} \) as the performance output in (3).

**Power model**

The power model is the base to solve the optimization problem (6). Roughly, the current power of the physical machine, \( P_m(t) \), mainly depends on the normalized CPU share of the physical machine \( PM_m \) ([20]), which is defined as

\[
u_m = \frac{\sum_{n \in PM_m} f_n}{n_m \cdot \bar{f}_m} \tag{10}
\]

and includes both the CPU share and the CPU frequency together. Obviously, \( u_m \leq [0, n_m \cdot 100\%] \), \( n_m \) is the number of cores of \( PM_m \).

Then the power for a physical machine can be modeled to be linear with \( u_m(t) \):

\[
P_m(t) = z_m + \beta_m \cdot u_m(t). \tag{11}
\]

The coefficients \( z_m, \beta_m \) can be obtained by regressing the data of the measured power \( P_m(t) \) and CPU utilization \( u_m(t) \).

In the simulation, data are collected from a Dell R510 and the power models are taken as

\[
P_1(t) = 180 + 51 \cdot u_1(t), \tag{12}
\]

\[
P_2(t) = 175 + 81 \cdot u_2(t), \tag{13}
\]

where \( u_m(t) = \frac{f_m(t)}{2.8} \), \( f_m(t) \) is the frequency of the core. It is easy to see that in a system, the power consumption can vary significantly with a same resource configuration.

**Workloads**

In the simulation, we will choose two different types of workloads as the testcases. They are the website request rates from a university [21] for two different traffics. Data are collected...
Figure 4. Load B to be completed, using LS-STR. When Load B is performed the power and performance are traced. Meanwhile the ideal needed resource and the actual resource allocated by the LS-STR controller are both shown in this figure. doi:10.1371/journal.pone.0057551.g004

Figure 5. Load A to be completed, using LS-STR. When Load A is performed the power and performance are traced. Meanwhile the ideal needed resource and the actual resource allocated by the LS-STR controller are both shown in this figure. doi:10.1371/journal.pone.0057551.g005
every 5 minutes for 2 days, so there are 574 data for each workload.

Figure 2 below shows the sequence of the two workloads: load A, load B. And in the following, we just write as $j = A, B$ when necessary. Apparently, they are very different: although neither of them is stationary, load A is more stationary than load B; load B is nearly periodic along with the time. In the simulation, we adopt different performance metrics for them: for load A, we consider the throughput; for load B, we consider the response time.

When the workload can be measured, however, that is rarely true in real problems, it can be predicted by suitable model. Then the predictive controller can be designed to track the performance. In the simulation, we will use the AR(2) model to predict the workload:

$$\hat{\lambda}_j(t) = \beta_1^j \lambda_j(t-1) + \beta_2^j \lambda_j(t-2),$$

$\hat{\lambda}_j(t)$ is the prediction. The coefficients $\beta_1^j, \beta_2^j$ can be estimated by online or off line algorithms, see [22].

For instance, when we use the first 100 statistics to estimate $\beta_1^j, \beta_2^j, j = B$, for load B by off-line algorithm, we get $\beta_1^B = 1.128, \beta_2^B = -0.127, j = B$. Then, together with the response time model (8), we can figure out the desired resource to ensure the performance requirement:

$$Res_j(t) = \frac{\chi_j^* - 3.28 \hat{\lambda}_j(t)}{\lambda_j^0}.$$  

![Figure 6. Two loads to be completed using LS-STR.](image)

When Load A and load B are performed the power and performance are traced. Meanwhile the ideal needed resource and the actual resource allocated by the LS-STR controller are both shown in this figure.

doi:10.1371/journal.pone.0057551.g006

| controller  | perf mean | perf variance | perf satisfy | power mean | power save |
|-------------|-----------|---------------|--------------|------------|------------|
| open-loop   | 1.1111    | 0.3095        | 574          | 487        | 0          |
| predictive controller | 2.4946    | 0.8861        | 369          | 253.9633   | 47.85%     |
| LS-STR      | 2.3529    | 0.7725        | 479          | 272.3558   | 44.07%     |

When load B is performed the performance metric is response time. Performance and power comparisons are listed under these three controllers. LS-STR controller has the best power saving result.

doi:10.1371/journal.pone.0057551.t001
Here AR(2) model is adopted because it is sufficient to make prediction and it is simple enough.

Results and Discussion

Figure 3, Figure 4, Figure 5, Figure 6 below show the simulation results. We will illustrate them in the following to show that the adaptive controller based on LS-STR is suitable and effective to track the performance and save power, whatever the cases is when there are one single workload or multiple workloads, or when the different performance metrics are adopted.

A: single load: load B, to track the response time

Now since there is only one type of workload: load B, we have $v_{ms,j} = 1, j = B$. The parameters in (8) are taken as $\text{Res}^0_j = 2.8, C_3^0 = 100, r_j^0 = 3$.

Now the response time $r_j(t)$ is adopted as the performance metric, and $y_j(t) = \frac{1}{r_j(t)}$ is the performance output in the regression model (3). And the controllers are designed to track the reference output $y_j^* = \frac{1}{r_j^*} = 1$.

We will use three types of controllers: the open loop controller, the predictive controller and the adaptive controller based on LS-STR to track the performance. When the open loop controller is used, the core will always be running with the highest frequency, and the resource will be obtained from (15). When the the predictive controller is used, the resource is obtained from (4)(5).

When the predictive controller is used, the averaged response time is 1.1111 with the variance being 0.3095. The performance requirement is satisfied at each time. However, the power is the maximal 407 at each time.

Using the open-loop controller, the averaged response time is 2.4946, with the variance being 0.8861. There are 369 times satisfying the demand. The mean power consumption is 253.9633 and is saved by 47.85% compared to the open loop controller.

Considering that the accurate performance model can not be obtained in the real experiment, the predictive controller is very possible to perform worse.

(2) Using the adaptive controller based on LS-STR, the average response time is 2.3529 with the variance being 0.7725, both of them are smaller than that using predictive controller. And in the response time sequences, those which satisfy the demand has a number of 479, which is much larger than using predictive controller. Now the averaged power consumption is 272.3558, and is saved by 44.07% compared to the open loop controller. Note that this reduction is quite considerable since the total power consumption can be very large according to [1].

So, to summarize briefly, LS-STR can achieve a much better performance than the predictive controller. At the same time, LS-STR can save power consumption as big as 44.07% compared to the open loop controller, which is as good as the predictive controller. On the other hand, LS-STR does not need to know the accurate performance model or to measure the workload, which is the situation in many real problems and thus it can be applied to a lot of systems. These show the great advantage of the adaptive controller based on LS-STR to other controllers like open loop controller and predictive controller.

B: single workload: load A, to track the throughput, using LS-STR

Still, since there is only one type of workload: load A, we have $v_{ms,j} = 1, j = A$. The parameters in (9) are taken as $\text{Res}^0_j = 2.8, C_3^0 = 300$.

Now the throughput $TP_j(t)$ is adopted as the performance metric, and $y_j(t) = \frac{TP_j(t)}{\lambda_j(t)}$ is the performance output in the regression model (3). Now the LS-STR is designed to track the reference output $y_j^* = 1$.

Figure 5 and Table 2 below show the simulation result using the adaptive controller based on LS-STR.

From Figure 5 and Table 2, we can see that the adaptive controller based on LS-STR is effective to track the throughput, which is quite different from the response time metric in both definition and the applicable scope. To be specific, when we use the LS-STR to track the throughput of load A with its mean being 301.3564, the throughput sequence $TP_0(t)$ can get a mean being 296.6247, while the performance output $y_j(t) = \frac{TP_j(t)}{\lambda_j(t)}$ has a mean being 0.9886 with a variance 0.0420, which is very near the

| Table 2. The performance and power using LS-STR: load A, to track the throughput. |
|-----------------|--------|--------|--------|--------|--------|--------|
| workload | mean($TP_j$) | mean($y_j$) | var($y_j$) | perf satisfy | power save |
| load A | 301.3564 | 296.6247 | 0.9886 | 0.0420 | 509 | 17.97% |

When load A is performed the performance metric is throughput. LS-STR controller shows 17.97% power saving.

doi:10.1371/journal.pone.0057551.t002

| Table 3. The performance and power using LS-STR: two loads, two performance metrics. |
|-----------------|--------|--------|--------|--------|--------|
| workload | perf mean | perf variance | perf satisfy | power save |
| load A | mean($y_j$) = 0.9962 | var($y_j$) = 0.0013 | 549 | 16.43% |
| load B | mean($y_j$) = 2.9905 | var($y_j$) = 1.6145 | 332 | 16.43% |

Load A and load B are performed at the same time and LS-STR controller shows 16.43% power saving.

doi:10.1371/journal.pone.0057551.t003
required value 1. And along the time, for as many as 309 times, the throughput equals the workload, i.e., no workload is abandoned. And this controller can save power 17.97% compared with the open loop controller.

C: two workloads, two performance metrics, using LS-STR

Now, since there are two types of workload to be dealt with in the system: load A and load B. So we have \( m_{\text{res}_A}(t) + m_{\text{res}_B}(t) = 1, \forall n \). Thus now the \( m_{\text{res}_A}(t) \) can be taken as the real number in the interval \([0,1]\). Now the parameters in (9) are taken as \( \text{Res}_0 = 2.8, A_B = 500, \text{res}_B = 200, r_B = 3 \).

Now the throughput \( TP_A(t) \) is adopted as the performance metric for load A, and the response time \( r_A(t) \) is adopted as the performance metric for load B, and \( y_A(t) = \frac{TP_A(t)}{r_A(t)} \), and \( y_B(t) = \frac{1}{r_B(t)} \) are the respective performance output in the regression model (3). And the reference output is \( y_A^* = 1 \) and \( y_B^* = 1/3 \).

In the simulation, to solve the optimization problem (6), we first get \( \text{Res}_j(t) = A_B \), then we get the total resource \( \text{Res}(t) = \text{Res}_A(t) + \text{Res}_B(t) \). Note that such a \( \text{Res}(t) \) can be out of the feasible set since the frequency set is discrete here. Then we find the \( \text{Res}(t) \) belonging to the feasible resource set which is nearest to and bigger than the required \( \text{Res}(t) \). Then the resource \( \text{res}(t) \) will be allocated to the jobs according to the proportion, i.e.

\[
\text{res}_j(t) = \frac{\text{Res}_j(t)}{\text{Res}(t)} \times \text{Res}(t)
\]

Figure 6 and Table 3 below show the simulation result using the adaptive controller based on LS-STR.

From the Figure 6 and Table 3, we can see that the adaptive controller based on LS-STR is also effective to deal with the situation where there are two workloads in the system and two performance metrics in the system. To be specific,

1. the LS-STR can track the throughput of load A pretty well: the sequence \( y_A(t) = \frac{TP_A(t)}{r_A(t)} \) has an averaged value 0.9962 with a variance 0.0013, and along the time, for as many as 349 times, the throughput equals the workload, i.e., no workload is abandoned.
2. the LS-STR can track the response time of load B well too: the response time sequence \( r_B(t) \) has an averaged value of 2.9905 with a variance 1.6145, and along the time, for as many as 332 times, the response time is smaller than 3 seconds, implying that the performance requirement is satisfied.

And finally, the LS-STR can save 16.43% power consumption compared with the open loop controller.

Of course, the curve of the changing \( m_{\text{res}_A, \text{B}} \) can also be drawn to see the details of the simulation process, which is omitted here.

Conclusions

Performance and power management in the virtualized environment is a fundamental, important and difficult problem. In this paper, by designing the adaptive controller based on at least square self-tuning regulator (LS-STR), we can dynamically regulate the resources and thus track the required performance and keep the power at a lower level as desired. Simulation results show that this method is very effective and general: it can deal with the problem when there are one application or multiple applications; it is also effective for different performance metrics.

Much work are worthy to do to complete the result and improve the solution in the future. For example, we can study how the parameters influence the effect of LS-STR. And when the maximal capacity of the system is not enough for the applications, there exists conflicts and games between the applications, so the applications might struggle for the resource. We can also take the time-delay effect and the switching cost when we turn on/off the PM into account. Moreover, in the real systems, both the performance and power models can be different among the physical machines and the topological structure between the physical machines also influence the performance and power. All these things will make the problem challenging in theory as well as engineering.

Acknowledgments

We thank Chao Li for his help during this research work. We also thank the reviewers for their useful comments and suggestions.

Author Contributions

Checked the writing errors: CW. Conceived and designed the experiments: CW XL YM. Performed the experiments: CW YM. Analyzed the data: XL YM. Contributed reagents/materials/analysis tools: CW YM. Wrote the paper: CW YM.

References

1. Growth in data center electricity use 2005 to 2010. Available: http://www.analyticpress.com/datacenters.html. Accessed 2012 May 1.
2. Wen G, He J, Zhang J, Long X (2010) PCS: Power Credit Based Fair Scheduler Under DVFS for Multicore Virtualization Platform. Proceedings of the 2010 IEEE/ACM Int’l Conference on Green Computing and Communications & Int’l Conference on Cyber, Physical and Social Computing: 163–170.
3. Xu W, Zhu X, Singhal S, Wang Z (2006) Predictive Control for Dynamic Resource Allocation in Enterprise Data Centers. Proceedings of the 10th IEEE/ IFIP Network Operations and Management Symposium: 115–126.
4. Albers S (2010) Energy-efficient algorithms. Communications of the ACM 53(5): 88–96.
5. Wang Y, Wang X, Chen M, Zhu X (2008) Power-Efficient Response Time Guarantees for Virtualized Enterprise Servers. Proceedings of 2008 Real-Time Systems Symposium: 303–312.
6. Lim H, Kanwal A, Liu J (2011) Power Budgeting for Virtualized Data Centers. Proceedings of the 2011 USENIX conference on USENIX annual technical conference: 5–15.
7. Kusuc D, Kephart JO, Hanson JE, Kandassamy N, Jiang G (2009) Power and Performance Management of Virtualized Computing Environments Via Lookahead Control. Journal Cluster Computing: 12(1): 1–15.
8. Wang X, Chen M (2008) Cluster-level feedback power control for performance optimization. Proceedings of the 14th IEEE International Symposium on High Performance Computer Architecture: 101–110.
9. Xu J, Fortes JAB (2010) Multi-Objective Virtual Machine Placement in Virtualized Data Center Environments. Proceedings of the 2010 IEEE/ACM Int’l Conference on Green Computing and Communications & Int’l Conference on Cyber, Physical and Social Computing: 179–181.
10. Vengerov D (2007) A reinforcement learning approach to dynamic resource allocation. Journal Engineering Applications of Artificial Intelligence 20(3): 383–390.
11. Bender MA, Rabin MO (2000) Scheduling cilk multithreaded parallel programs on processors of different speeds. Proceedings of the twelfth annual ACM symposium on Parallel algorithms and architectures: 13–21.
12. Wieder U (2007) Balanced allocation with heterogeneous bins. Proceedings of the nineteenth annual ACM symposium on Parallel algorithms and architectures: 188–193.
13. Krishman B, Amur H, Gavrilova A, Schwank K (2010) VM power metering: feasibility and challenges. ACM SIGMETRICS Performance Evaluation Review archive 38(3): 56–60.
14. Guo L (1995) Convergence and logarithm laws of self-tuning regulators. Automatica 31(3): 435–450.
15. Kalman RE (1958) Design of a self-optimizing control system. Trans. ASME 70: 486–478.
16. Astrom KJ, Wittenmark B (1973) On self-tuning regulators. Automatica 9:185–199.
17. Guo L., Chen H. (1991) The Astrom-Wittenmark self-tuning regulator revisited and ELS-based adaptive trackers. IEEE Trans. Automatic Control 36(7):802–812.
18. Guo L. (1994) Further results on least square based adaptive minimum variance control. SIAM J. Control and Optimization 32(1): 167–212.
19. Astrom K.J. (1983) Theory and appllication of adaptive control—a survey. Automatica 19:471–486.
20. Spec website. Available: http://www.spec.org/power_ssj2008/results/res2010q4/power_ssj2008-20100921-00294.html. Accessed 2012 Jun 6.
21. Beihang University Website. Available: http://mrtg.buaa.edu.cn. Accessed 2012 Jun 10.
22. Ljung L. (1999) System identification: Theory for the User. N.J: Prentice Hall. 20 p.