9-28-2011

Control in Computing Systems: Part I

Chaouki T. Abdallah

Jose Luna

Recommended Citation
Abdallah, Chaouki T. and Jose Luna. "Control in Computing Systems: Part I." (2011): 25-31. doi:DOI 10.1109/CACSD.2011.6044541.

2011 IEEE International Symposium on Computer-Aided Control System Design (CACSD)
Part of 2011 IEEE Multi-Conference on Systems and Control
Denver, CO, USA. September 28-30, 2011

Control in Computing Systems: Part I

José Marcio Luna & C.T. Abdallah

Abstract— **This is the first part of a paper that provides an overview of some applications of control theory to computing systems. With the advent of cloud computing and more affordable computing infrastructures, computing engineers are looking for control theory to provide rigorous tools to analyze and design computing systems and algorithms. On the other hand, control systems have greatly benefited from the advent of distributed and affordable computing infrastructure. In part I we will focus on control applications at the data center level.**

I. Introduction

Control theory has had a productive but limited relationship with computing theory and systems. In recent years, control is being used in problems such as managing power consumption for data centers, smart grids, managing resources in cloud computing applications, congestion control, and networked control systems. The objective of this paper is to review the interplay between computing and control. We will review various application domains where control theory has had an impact while also stressing a variety of control concepts and technologies that may impact computing systems. Our intent is to highlight control problems and solutions from single computer, multi-core processors and servers all the way up to large data centers. Control techniques from classical control designs (PID) to optimal control and model-predictive and adaptive controls will also be highlighted. Recent overview papers on the interaction of control and computing are found in [1], [2]. Specific problems include workload management [2], power and performance control [3], resource allocation [4], [5], load balancing and management [6], congestion control [7], digital rights management [8], [9], etc. This paper is organized as follows. Section II discusses controls applications at the data center level, including workload management and power and performance control. Throughout the paper, various control techniques are illustrated. Section III contains our conclusions.

II. Data Center Level Controls

In this section we discuss control at the data center level. We assume that we have multiple computers working together and sharing resources in order to carry out computations. We focus on two of the current mainstream areas of research namely, *workload management* and *power and performance* [1].

J. M. Luna is with the Department of Electrical and Computer Engineering, The University of New Mexico, MSC 01 1100, 1 University of New Mexico, Albuquerque, NM 87131, USA jmarcio@ece.unm.edu

C.T. Abdallah is with Faculty of the Department of Electrical and Computer Engineering, The University of New Mexico, MSC 01 1100, 1 University of New Mexico, Albuquerque, NM 87131, USA chaouki@ece.unm.edu

A. Workload Management

Data centers have become predominant in enterprise computing. Services such as on-demand computing are crucial for commercial users willing to take advantage of companies with experience in data center administration *e.g.,* Amazon, HP, Microsoft, SalesForce, and Google [10]. Current motivating and leading initiatives such as the RESERVOIR framework [11] have been working on different aspects to improve cloud computing infrastructure. One of the objectives of RESERVOIR is precisely the encouragement of solutions related to the automatic allocation and deployment of resources depending on fluctuation on demands. Subscribing to computing on-demand services reduces the costs of infrastructure acquisition and support [10]. With the implementation of *shared virtualized infrastructure* on multi-tier servers [12], [13] a variety of enterprise computing services became possible. The data centers have become the physical manifestation of cloud computing. Cloud computing allows users to use large resources for storage and computational capabilities [10], [14], [15]. Currently most of the services on demand are available based on *resource capacity* rather than *application performance* [15]. Guaranteeing application performance is still a difficult task given the complexity involved in the management of virtual machines. The time-varying nature of the demands, the dependency of the applications on shared resources, and the varying resource allocation complicates the negotiation of SLAs to guarantee a Quality of service (QoS) [16], [17]. In what follows of the section we will present two approaches to the problem of guaranteeing QoS in data centers.

1) Fluid Approximations and Optimal Control: A multi-tier server system consists of a set of servers laid out in series as shown in Fig 1 which is based on [13]. The k-th server acts as a client for the $(k+1)$-th server. Based on [13] we have that for the k-th server on the series, the incoming and outgoing throughput are represented by T_k^i and T_k^o respectively. Furthermore, the admission control generates a rejection throughput called T_k^r, while the number of concurrent connections, and the multi-programming level (MPL) associated to the k-th server are denoted by N_k and MPL_k respectively. Finally, for the k-th server, let us denote the processing time of a request as S_k and let us call λ_k the probability that a request processed by the server $k-1$ produces a request on server k. Following [13], we assume the following for multi-tier server systems ,

- **Assumption 1:** Every request that arrives at tier $k > 1$, can only originate from tier $k-1$.

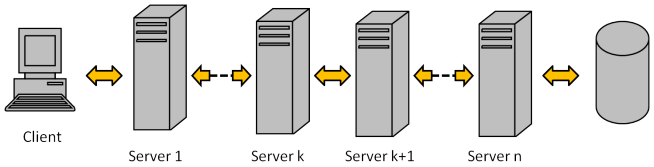

Fig. 1. Multi-tier server system in series.

- **Assumption 2:** We say that a request is processed by tier $k > 1$, when the tier sends a response to tier $k-1$.
- **Assumption 3:** One or more requests *in series*, can be generated at tier $k+1$ from a request being processed at tier k.
- **Assumption 4:** At any specific time t, at most one request can be generated at tier $k+1$ from a request being processed at tier k.
- **Assumption 5:** If a client request is rejected at tier $k > 1$ then the "generating" request will be rejected at tier $k-1$ as well.
- **Assumption 6:** For all concurrent connections we have that $N_i \geq N_{i+1}$ for all i in the interval $[1, n-1]$.

The sever workload can be divided into *workload amount* and *workload mix*. Workload amount corresponds to the number of clients that try to access a server at the same time. Workload mix deals with the distribution of the available variety of interactions supplied by the clients *e.g.,* read-only request mix or read-write request mix. Based on [13] and [18], the most common parameters used to measure the QoS of the server systems are the *latency* ℓ and the *abandon rate* α. Latency is the time the server needs to process a request from the client. The abandon rate of a server is the ratio between the rejected requests and the total number of received requests. Another variable of interest is the *cost* represented by w. The cost is related to the amount of resources involved in a cluster based multi-tier application and may be interpreted as the number of active machines used to host the applications. Reference [13] proposes a fluid approximation model to implement an admission control for multi-tier server-systems in order to guarantee a given QoS objective under high loads. Admission control consists of setting an initial value for the MPL_k of each server.

If the number of concurrent clients overtakes MPL_k, the k-th server rejects incoming requests. Note that in [13] multiple visits of a request to a tier are not considered. Then, defining the model of the multi-tier server system we have that the state vector of the system is given by $(N_1, N_2, \ldots, N_n)^T$, the external input is given by the incoming throughput of the front-end server T_1^i and the control input is given by the vector $(MPL_1, MPL_2, \ldots, MPL_n)^T$. Then, in [13] the author comes up with the following dynamical model of the number of concurrent connections on the server k,

$$\dot{N}_k(t) = T_k^i(t) - T_k^o(t) - \sum_{i=k}^{n} T_i^r(t), \quad \text{with } k = 1, \ldots, n. \quad (1)$$

After considering the assumptions given above the author propose the following recursive expression of T_k^o,

$$T_k^o(t) = (1-\lambda_{k+1})\frac{(N_k(t) - N_{k+1}(t))}{S_k} + \lambda_{k+1} T_{k+1}^o(t), \quad (2)$$

where for the n-th server $T_n^o = \frac{N_n(T)}{S_n}$. Furthermore, the rejection throughput of the server k is defined as,

$$T_k^r = \begin{cases} 0 & \text{if } N_k < \text{MPL}_k \\ T_k^i & \text{otherwise} \end{cases}$$

In [13] the processing time of a request on server k, namely $S_k(t)$ is considered linearly proportional to the number of requests being processed by server k, then

$$S_k(t) = \begin{cases} a_k(N_k(t) - N_{k+1}(t)) + b_k & \text{for } 1 \leq k \leq n-1 \\ a_n N_n(t) + b_n & \text{for } k = n \end{cases}$$

As noted earlier the outputs of the system are the latency ℓ and the abandon rate α. For this case, the average client request latency is considered, and is given by, $\ell = \sum_{k=1}^{n} S_k \cdot \prod_{j=1}^{k} \lambda_j$, and the abandon rate defined as, $\alpha = \frac{T^r}{T_1^i}$, where $T^r = \sum_{k=1}^{n} T_k^r$. The controller input for the admission control consists of finding the optimal input vector $MPL^* = (MPL_1^*, \ldots, MPL_n^*)$ that minimizes T_1^r while maintaining the average latency under a determined threshold $L_{\max}$. Notice that the system is assumed saturated i.e. $\forall k \in [1, n], \quad N_k = MPL_k$, then, from (2) we get that,

$$\begin{aligned} T_1^o &= \sum_{k=2}^{n} \left(\prod_{j=1}^{k-1} \lambda_j \right) (1-\lambda_k) \frac{\text{MPL}_{k-1} - \text{MPL}_k}{S_{k-1}} \\ &+ \left(\prod_{j=1}^{n} \lambda_j \right) \frac{\text{MPL}_n}{S_n}, \end{aligned} \quad (3)$$

Assuming that the system in steady state, we get from (1) that the global rejection problem gives, $T^r = T_1^i - T_1^o$, subject to $\ell \leq \ell_{\max}$, with

$$S_k(t) = \begin{cases} a_k(\text{MPL}_k(t) - \text{MPL}_{k+1}(t)) + b_k & k \leq n-1 \\ a_n \text{MPL}_n(t) + b_n & \text{for } k = n \end{cases}.$$

Lastly the Karush-Kuhn-Tucker (KKT) necessary conditions are used in [13] to find μ^* such that,

$$\begin{aligned} \nabla_{MPL} \mathcal{L}(\text{MPL}^* \cdot \mu^*) &= 0, & (4) \\ \mu^*(\ell \cdot \text{MPL}^* - \ell_{\max}) &= 0, & (5) \\ \mu^* &\geq 0, & (6) \end{aligned}$$

where $\mathcal{L} = T^r + \mu(\ell - \ell_{MAX})$ is the Lagrangian. Differentiating the Lagrangian, the optimal control input MPL^* with $\mu^* \neq 0$ is found. In order to test the performance of the MPL optimization strategy the author proposes a comparison with three additional strategies namely,

- **Strategy 1:** Admission control for front-end server only and $MPL = (50, 50, 50)$.
- **Strategy 2:** Admission control at every tier and $MPL = (50, 40, 30)$.
- **Strategy 3:** Restrictive admission control at every tier and $MPL = (50, 30, 10)$.

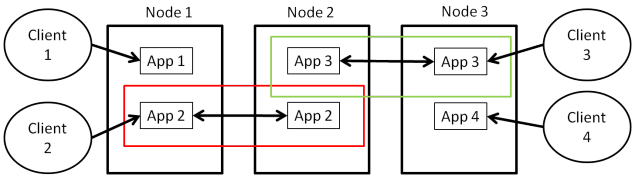

Fig. 2. A realization of a shared virtualized infrastructure.

The author then compares the results of the three strategies above with the optimal MPL. Strategies 2 and 3 present less latency but higher abandon rates as long as the number of clients is relatively low. After a certain number of clients, Strategy 2 results in the smallest abandon rate. Therefore, the best methodology among strategies 1, 2 and 3 is the implementation of admission control at each tier. The optimal strategy MPL^*, performs the best when the workload amount is high, both in terms of latency and abandon rate. However, for light loads strategies 1 and 2 present better performance since the calculations of the optimal MPL assumes that the system is saturated.

2) Model Estimation and Optimal Control: Padala *et. al.,* present in [12] a tool called *AutoControl*, which is composed of an online model estimator and a Multi-Input Multi-Output (MIMO) resource allocation controller. The main goal of this controller is to optimally allocate the virtualized resources on a data center so that the system can fulfill the required Service Level Objective (SLO). From Fig 2, based on [12], every tier of an application resides in a *virtual machine* (VM) and a multi-tier application can run on several nodes. *AutoControl* has been designed in a two-layer distributed architecture. The first layer is composed by a set of node controllers called *NodeControllers* and the second layer is composed by a set of application controllers called *AppControllers*. There is one NodeController for each virtualized node and one AppController for each hosted application.

Notice that the following assumptions are considered in [12],

- **Assumption 1:** There is a separate service for capacity planning to carry out the initial placement of the applications among the nodes,
- **Assumption 2:** the same service takes care of admission control for new applications,
- **Assumption 3:** the workload migration system works at a time scale of minutes or longer.

The AppController is composed of a *model estimator* and an *optimizer*. The model estimator uses an auto-regressive-moving-average (ARMA) model to approximate the relation between the resource allocation of the application $a \in A$, namely $u_a = \{u_{a,r,t} : r \in R, t \in T_a\}$ and the normalized performance for application a namely, $\hat{y}_a = \frac{y_a}{\bar{y}_a}$, where y_a and $\bar{y}_a$ are the actual and the desired performance of the application a respectively. The ARMA model proposed in [12] is given by,

$$\begin{aligned} \hat{y}_a &= a_1(k)\hat{y}_a(k-1) + a_2(k)\hat{y}_a(k-2) \\ &+ {b_0}^T(k)u_a(k) + {b_1}^T(k)u_a(k-1), \end{aligned} \tag{7}$$

$u_{a,r,t}$ is the actual allocation of the resource $r \in R$ for the application $a \in A$ in tier $t \in T_a$. A is the set of all hosted applications, R is the set of all resource types considered and T_a is the set of all tiers associated to application a. The parameters $a_1(k)$ and $a_2(k)$ and the parameter vectors $b_0(k)$ and $b_1(k)$ are estimated through a recursive least square algorithm. The optimizer searches for the required allocation vector $\bar{u}_a$ that guarantees the performance target of the application a while avoiding oscillations in resource allocation. This may be achieved by minimizing the following cost function,

$$J_a = (\hat{y}_a(k) - 1)^2 + q\|u_a - u_a(k-1)\|^2. \tag{8}$$

Note that the function J_a is minimized for $\hat{y}_a(k) \approx 1$ leading the application a to approach the desired performance. Furthermore (8) is minimized if the difference $\|u_a - u_a(k-1)\|$ becomes smaller *i.e.* when large changes in the resource allocation during a sample period are avoided. The optimal resource allocation strategy is therefore given by,

$$\begin{aligned} \bar{u}_a(k) &= (b_0{b_0}^T + qI)^{-1}\left((1 - a_1\hat{y}_a(k-1)\right. \\ &\quad -a_2\hat{y}_a(k-2) - b_1^T u_a(k-1))b_0 \\ &\quad \left. +qu_a(k-1)\right), \end{aligned}$$

with I the identity matrix. Note that , $a_i, b_i; i = 1, 2$ are functions of k. The second layer in Autocontrol consists of the NodeController. This module takes the resource allocation calculated by the AppController and, if the total requested allocation is less than the available capacity of the system, it assigns the resources as the AppController dictates. On the other hand, when the total resource allocation is higher than the available capacity we have *contention*. In such cases, a new optimization problem is carried out in order to minimize the error between the required application performance and the actual one.

In an example proposed in [12], two nodes namely, n_1 and n_2 hosting applications 1 and 2 are considered, and the shared resources are the CPU and disk. Then, from application 1 we get the requests $\bar{u}_{1,cpu,web}$ and $\bar{u}_{1,disk,web}$. Similarly, from application 2 we get the requests $\bar{u}_{2,cpu}$ and $\bar{u}_{2,disk}$. Assuming that node n_1 has enough capacity to fulfill the request for only one of the available resources. Then, modeling the shortage of the resources we get $\Delta u_{1,r,web} = \bar{u}_{1,r,web} - u_{1,r,web}$ and $\Delta u_{2,r} = \bar{u}_{2,r} - u_{2,r}$. This problem

can be expressed as a constrained optimization problem,

$$\begin{aligned} \min J_{n1} &= w_1 \left(\frac{\partial \hat{y}_1}{\partial u_{1,r,web}} \Delta u_{1,r,web} \right)^2 \\ &\quad + w_2 \left(\frac{\partial \hat{y}_2}{\partial u_{2,r}} \Delta u_{2,r} \right)^2, \qquad (9) \end{aligned}$$

subject to

$$\Delta u_{1,r,web} + \Delta u_{2,r} \geq \bar{u}_{1,r,web} + \bar{u}_{1,r,web} + \bar{u}_{2,r} - 1, \qquad (10)$$

$$\Delta u_{1,r,web} \leq 0, \qquad (11)$$

$$\Delta u_{2,r} \leq 0. \qquad (12)$$

where the expression $\frac{\partial \hat{y}_1}{\partial u_{1,r,web}} \Delta u_{1,r,web}$ estimates the error between the requested and actual performance of application 1. Note the similar expression for error estimation of application 2 in the second term in the right hand side of (9). The condition in (10) reflects a capacity constraint that when violated implies contention of one of the resources. The conditions given by (11) and (12) guarantee that no application can exceed its target performance at the detriment of the other. The weights w_1 and w_2 determine the priority of the applications and the lower the weight, the greater the degradation. Some technical details of the computational testbed where Autocontrol has been implemented are given in [12]. Simulations were conducted with four media applications and one RUBiS application. The idea was to induce reduced bottlenecks of the Disk or of the CPU and assess the performance of Autocontrol compared with two state-of-the-art techniques, namely, *work-conserving* and *static* allocation modes [12]. Autocontrol stayed closer to the throughput target even after a bottleneck was introduced.

B. Power and Performance

Based on references cited in [1], in large data centers about 23-50% of the incomes should be invested on energy. Poussot-Vassal *et al.,* assert in [19] that Information Technology (IT) analysts predict that by 2012, up to 40% of the technology budget of a company will be the cost of energy. Furthermore, IT produces around 2% of global CO_2 emissions, an amount equivalent to the emissions of global air traffic. Wang *et. al.,* assert [20] that for every 1 W of power spent on the operation of servers, 0.5–1 W of additional power are required for the cooling equipment. These examples encourage the control community to put some research effort on the study of power and performance in web service systems.

In fact, control systems techniques have been recently proved to be effective in power and performance control for computing system applications [1], [16]. Research related to the unified control of cooling resources and servers may be found in [21], [22], [23], [24]. In what follows, we review methodologies aimed not only to optimally manage the power consumption but to also optimize the general performance of data centers. In section II-B.1 we explore a technique proposed by Fu *et. al.,* to optimize power consumption in data centers by using Linear Quadratic Control. In section II-B.2 we review an optimal control analysis that may be the first formal approach to the unified control for power management, cooling, and workload management for clusters of servers.

1) Linear Quadratic Regulator: A methodology to optimize power consumption in clusters is proposed by Fu *et al.*, in [3] and [25]. Their approach is based on *thermal balancing* rather than *load balancing*, since the thermal dynamics of the processor are not only related to the load but to the past history of processing as well. This approach is called *Control-Theoretic Thermal Balancing* (CTB). A feedback loop monitors the temperature and CPU utilization of different servers in a cluster and redistributes the service requests among the processors in the cluster so that the temperature is balanced. The utilization is included in the actual model because temperature variations are usually slow. The proposed thermal model for processor P_{ri} is given by,

$$\frac{dT_i(t)}{dt} = -c_{i,2}(T_i(t) - T_0) + c_{i,1}P_i(t), \qquad (13)$$

where T_i is the temperature of the processor P_{r_i}, P_i is the actual active power of the processor, $c_{i,1}$, $c_{i,2}$ are constants which depend on the thermal features of the processor and T_0 is the ambient temperature. Defining the error variable $T'i(t) = T_i(t) - T_0$ we have,

$$\frac{dT_i'(t)}{dt} = -c_{i,2}(T_i'(t) - T_0) + c_{i,1}P_i(t). \qquad (14)$$

Assuming that all the processors are homogeneous, the authors propose the following model for the whole system, $\dot{T}'(t) = AT'(t) + BP(t)$, where $T'(t) = [T_1'(t), T_2'(t), \ldots, T_n'(t)]$ and $P(t) = [P_1(t), P_2(t), \ldots, P_n(t)]$ and $A = \text{diag}\{-c_{1,2}, \ldots, -c_{n,2}\}$, $B = \text{diag}\{c_{1,1}, \ldots, c_{n,1}\}$ Applying a bilinear transformation to discretize the system it gives, $T'(k+1) = \Phi T'(k) + \Gamma' P(k)$ where $\Phi = \left(I + \frac{AW}{2}\right)\left(I - \frac{AW}{2}\right)^{-1}$ and $\Gamma' = \left(I - \frac{AW}{2}\right)^{-1} B\sqrt{W}$ and W is the length of sampling interval. Up to this point we have not included the dynamical model of the server cluster. Based on [25] the authors consider that when a server is idle its power decreases to a minimum value P_{sleep}. Furthermore, the average power of the processor P_{r_i} at the sampling time k is $\bar{P}_i(k)$. Finally, there is a set of running tasks, namely, J_i. The estimated utilization of the i-th processor in the k-th sampling may be approximated by the sum of the estimated utilization of each task, $U_i(k) = \sum_{j \in J_i(k)} u_j$. Then the dynamical model for the average power gives,

$$\bar{P}_i(k) = G_i(P_a - P_{sleep})U_i(k) + P_{sleep}, \qquad (15)$$

where $G_i = \frac{\tilde{P}_a - P_{sleep}}{P_a - P_{sleep}}$, $\tilde{P}_a$ is the actual active power and P_a is the estimated active power. Based on [3], approximating the discrete power $P(k)$ by the average power $\bar{P}$ induces an error that can be neglected. Defining $T(k) = T'(k) - \Gamma' P_{sleep}(\Phi - I)^{-1}$ then we get the following thermal model,

$$T(k+1) = \Phi T(k) + \Gamma U(k), \qquad (16)$$

where $\Gamma = G(P_a - P_{sleep})\Gamma'$.

We should therefore shoot for thermal balance while reducing the control cost. The dynamics of the estimated utilization may be defined as,

$$U(k+1) = U(k) + \Delta U(k). \tag{17}$$

Then, expressing (15) in matrix form,

$$\begin{pmatrix} T(k+1) \\ U(k+1) \end{pmatrix} = \begin{pmatrix} \Phi & \Gamma \\ 0 & I \end{pmatrix} \begin{pmatrix} T(k) \\ U(k) \end{pmatrix} + \begin{pmatrix} 0 \\ I \end{pmatrix} \Delta U(k). \tag{18}$$

The LQ controller is given by,

$$\Delta U(k) = - \begin{pmatrix} K_T & K_U \end{pmatrix} \begin{pmatrix} T(k) \\ U(k) \end{pmatrix}, \tag{19}$$

for the LQ cost function given by,

$$\begin{aligned} \min_{k\to\infty} \quad & \sum_k \quad \left\{ \begin{pmatrix} T(k) & U(k) \end{pmatrix} Q_{UT} \begin{pmatrix} T(k) \\ U(k) \end{pmatrix} \right. \\ & + \quad \left. \Delta U^T(k) R_{UT} \Delta U(k) \right\}, \end{aligned} \tag{20}$$

where

$$Q_{UT} = \begin{pmatrix} L & 0 \\ 0 & \lambda \end{pmatrix}^T \begin{pmatrix} L & 0 \\ 0 & \lambda \end{pmatrix}, \tag{21}$$

with $R_{UT} = \rho I$ where ρ is a parameter that can be tuned to achieve the balance between thermal balancing and control cost. L is given by

$$L = \begin{pmatrix} 1-\frac{1}{n} & -\frac{1}{n} & \cdots & -\frac{1}{n} \\ -\frac{1}{n} & 1-\frac{1}{n} & \cdots & -\frac{1}{n} \\ \vdots & \ddots & \cdots & -\frac{1}{n} \\ -\frac{1}{n} & \cdots & \cdots & 1-\frac{1}{n} \end{pmatrix}, \tag{22}$$

and λ determines the degree of thermal balancing ($\lambda < 1$) or load balancing ($\lambda > 1$) in the controller. Remember that n corresponds to the total number of processors. The authors present some simulation results that show a comparative performance of the CTB algorithms. Taking the parameter values in Table I from the technical specifications given for a Pentium IV processor they proceed to simulate the system. The authors compare an open-loop controller, a load balancing controller (LB) that tries to balance the CPU utilization of the available processors, and a Heuristic Temperature Balancing (HTB) algorithm that works under the principle that changes on the temperature of the processor are proportional to changes on the utilization of the processors. The HTB controller does not take into account the thermal dynamics of the system and does not take into consideration the control cost. At $t = 1000s$ the utilization is changed randomly to a value in the interval $[0.8, 1.2]$ with values of $\rho = 1$, $\lambda = 0.005$ for CTB-UT and $\rho = 1$ for CTB-T. Since LB and OPEN do not take into account the control of temperature, the temperatures remain constant and are not reduced while CTB-T, CTB-UT and HTB achieve good temperature control. On the other hand, the HTB controller uses more control effort than the CTB algorithms, since the LQ controller considers the control cost while the HTB does not.

TABLE I
SIMULATION PARAMETERS [3]

Parameter	Value	Description
Sampling Period (P_s)	4 s	
Ambient Temperature (T_a)	45° C	
Active Power (P_a)	51.9 W	
Sleep Power (P_i)	13.3 W	
Thermal Capacity (C)	295.7 J/K	
Thermal Resistance (R)	0.1 K/W	
c_1	0.0034 K/J	$c_1 = \frac{1}{C}$
c_2	0.0338 s	$c_2 = \frac{1}{RC}$

2) Optimal Control: Wang *et. al.,* analyze in [20] the challenges and opportunities of a unified approach for workload, power, and cooling management for data centers. They formulate the problem as an optimal control problem to minimize the power consumed by the IT components, as well as to minimize the power of the facility components denoted $P_{servers}$ and $P_{cooling}$ respectively. Furthermore, the feedback system should satisfy the demand of resources required by the application. Therefore, a threshold for resource utilization $Util_{Ref}$ is determined so that, as long as the resource utilization $Util_{servers}$ is under this threshold, the performance of the application is guaranteed. Another important parameter is the temperature of the server $T_{servers}$. The authors propose an upper bound T_{Ref} which guarantees the satisfactory operation of all components based on the technical specifications of the server. Finally, the total power consumption should be kept below a budget P_{budget}.

Then, based on [20], the problem may be formulated as the following constrained optimization problem,

$$\min \quad \int_0^t (P_{cooling}(\tau) + P_{servers}(\tau)) d\tau,$$

subject to

$$Util_{servers}(t) \leq Util_{Ref}(t), \tag{23}$$

$$T_{servers}(t) \leq T_{Ref}, \tag{24}$$

$$P_{cooling} + P_{servers} \leq P_{budget}. \tag{25}$$

A thermal blade enclosure is proposed as a simplified model of the data center. The main objective is to minimize the cooling power consumption and at the same time to keep the temperature of the server in an operational range. The authors propose the following thermal model for the blade enclosure system,

$$C_1 \frac{dT_{CPU,j}}{dt} = \frac{C_2}{R_j}(T_{amb,j} - T_{CPU,j}) + Q, \tag{26}$$

where T_{amb} is the ambient temperature and T_{CPU} is the temperature of the CPU. $R_j = \frac{C_3}{\dot{V}_j^{n_R}} + C_4$, for any blade j, and corresponds to the thermal resistance between the temperature of the CPU and the ambient temperature sensor. C_k with $k = 1, 2, 3, 4$ are constants which depend on the fluid, the properties of the material, the CPU package and the circulating air itself. $\dot{V}_j$ is the volumetric air flow rate through blade j. Q_j represents the heat transferred per unit

of time between the ambient air and the CPU, which based on [20] may be approximated by the power consumption of the CPU P_{CPU}. P_{CPU} can be expressed in terms of the utilization $Util_j$, $P_{CPU} = g_{CPU} \cdot Util_j + P_{CPU,idle}$ where $P_{CPU,idle}$ corresponds to the average minimum power of the CPU and g_{CPU} is a slope which incorporates the effect of of the power status tunning. Note that even though from (26) the temperature of the CPU is modeled as a first-order linear function, variations in the fan speed due to the workload active control make the system a time-varying nonlinear system. Notice that the objective function and constraints should be applied to different systems such as server components, the servers themselves, groups of servers, as well as the data center. Furthermore, the time constants may range from milliseconds to possibly minutes, then the following hierarchical controller to optimize the management of the workload, power and cooling in the data center is proposed in [20],

- **Efficiency Controller (EC):** This controller modifies the P-states of the CPU to keep the CPU utilization bounded within an interval. This controller operates from sub-seconds to seconds.
- **Fan Controller (FC):** This one controls the fan speeds in the blade enclosure to maintain the server temperatures below a given threshold and optimizes $P_{cooling}$. This controller operates from seconds to tens of seconds.
- **Local Power Capper (LPC):** This controller works on each CPU and tunes the P-states of the CPU to keep the power consumption of the server bounded by a given threshold. This controller operates from sub-seconds to seconds.
- **Group Power Capper (GPC):** Setting a power threshold for all servers, this controller along with the LPC keep the power budget P_{budget} bounded. This controller runs at longer times than LPC.
- **Global Controller (GC):** This controller carries out the migration of VM and turns servers on or off if necessary to minimize the total power consumption and fulfill all the constraints of the optimization problem. This controller should be executed every 10 minutes or even longer.

In order to guarantee the tracking of the utilization or of the power budget the EC, LPC and GPC controllers are integrated. A model predictive control (MPC) with three steps look-ahead horizon is used to introduce feedback in the FC. The following is the optimization problem for the FC,

$$\begin{aligned} J &= \min \sum_i P_{F_i} + w\|FS(k+1) - FS(k)\|^2, \\ &\quad \text{subject to} \\ T_{CPU,j}(k+h) &\leq T_{ref}, \quad h = 1,2,3 \text{ for each blade } j, \\ LB_i &\leq FS_i \leq UB_i \text{ for each fan } i. \end{aligned} \tag{27}$$

where $FS_i(k)$ represents the fan speed at time k. LB_i and UB_i are the lower and upper bounds for the fan speed. Finally, w is a weight that determines how responsive is the system to the fan speed changes.

Given the difficulty of the problem because of the several metrics involved, the cost function in (27) can be modified by adding a penalty term when the temperature rises above the threshold and the constraints are relaxed for the FC. In the case of the GC the problem is even more difficult, since the controller tries to minimize the fan power consumption while satisfying the constraints on temperature, performance and budget. Simulated annealing is used to solve this problem.

The authors carried out simulations combining two different kinds of FC with two different kinds of GC. The two kinds of FC are the *zonal feedback controller*, which controls the fans in a single row of the blade enclosure depending on the biggest difference between the temperature reference and the actual temperature on the row. Model Predictive Control (MPC) is the other option. GC uses simulated annealing in both options but applies *thermal aware GC* to calculate the minimal fan power necessary to keep the temperature under the threshold by an optimization problem. In the other case the minimal fan power was not estimated and that is called *non-thermal aware GC*. The power budget was chosen to be $2200W$ which is in the middle of the maximum and minimum power consumption with no power capping. Every combination of fan and global controllers namely C1, C2, C3 and C4 respectively have the following features,

- **C1** Zonal feedback controller and non-thermal aware GC.
- **C2** MPC and non-thermal aware GC.
- **C3** Zonal feedback controller and thermal aware GC.
- **C4** MPC and thermal aware GC.

Then the authors showed that the mean power consumption remained under the power budget in all cases, and that the percentage of samples that exceeded the budget by 5% are below 1%. It is shown in [20] that the thermal-aware GC had the lowest performance loss, and that the fan power in C2 is less than the fan power of C3 at lower temperatures which means that the capacity utilization in C3 is better than in C2 because of the thermal aware GC.

III. Conclusions

While the applications of control techniques to computing systems have recently exploded, rigorous analysis of computing systems using control theoretic approaches remains limited. In this overview, we have attempted to illustrate how model estimation, optimization, and control approaches that have been developed by control theorists are finding their way into the fast moving area of data center control. Due to space limitations, more mature control applications such as those for congestion control [7] and load balancing [6] are not covered, but neither are more recent contributions of game theoretic approaches to computing.

References

[1] T. Abdelzaher, Y. Diao, J. L. Hellerstein, C. Lu, and X. Zhu, "Introduction to control theory and its application to computing systems," in *SIGMETRICS Tutorial*, Annapolis, MD, June 2008, pp. 185 – 215.

[2] J. L. Hellerstein, S. Singhal, and Q. Wang, "Research challenges in control engineering of computing systems," *IEEE Transactions on Network and Management*, vol. 6, no. 4, pp. 206–211, 2009.

[3] Y. Fu, C. Lu, and H. Wang, "Control-theoretic thermal balancing for clusters," in *International Workshop on Feedback Control Implementation and Design in Computing Systems and Networks (FeBID'09)*, April 2009, invited paper.

[4] M. A. Kjær, M. Kihl, and A. Robertsson, "Resource allocation and disturbance rejection in web servers using slas and virtualized servers," *IEEE Transactions on Network Service Management*, vol. 6, no. 4, pp. 226–239, 2009.

[5] J. Piovesan, C. Abdallah, and H. Tanner, "A hybrid framework for resource allocation among multiple agents moving on discrete environments," *Asian Journal of Control*, vol. 10, no. 2, pp. 171–186, 2008.

[6] Z. Tang, J. Birdwell, J. Chiasson, C. Abdallah, and M. Hayat, "Resource-constrained load balancing controller for a parallel database," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 4, pp. 834–840, 2008.

[7] D. Cavendish, M. Gerla, and S. Mascolo, "A control theoretical approach to congestion control in packet networks," *IEEE/ACM Trans. Networkng*, no. 5, pp. 893–906, 2004.

[8] G. L. Heileman, P. A. Jamkhedkar, J. Khoury, and C. J. Hrncir, "DRM game," in *Proceedings of the Seventh ACM Workshop on Digital Rights Management*, Alexandria, VA, Oct. 2007, pp. 54–62.

[9] G. L. Heileman, H. Jerez, P. A. Jamkhedkar, and J. Khoury, "Indirect DRM evaluation architecture," in *Proceedings of the 5th International Workshop for Technical, Economic and Legal Aspects of Business Models for Virtual Goods*, Koblenz, Germany, Oct. 2007.

[10] D. Hilley, "Cloud computing: A taxonomy of platform and infrastructure-level offerings," Georgia Institute of Technology, Tech. Rep., April 2009.

[11] Reservoir: Resources and services virtualization without barriers. [Online]. Available: http://62.149.240.97

[12] P. Padala, K.-Y. Hou, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, and A. Merchant, "Automated control of multiple virtualized resources," in *Proceedings of the 4th ACM European conference on Computer systems*. Nuremberg, Germany: ACM, March 2009, pp. 13–26.

[13] L. Malrait, "Qos-oriented control of server systems," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 3, pp. 59–64, July 2010.

[14] G. Lee, N. Tolia, P. Ranganathan, and R. H. Katz, "Topology-aware resource allocation for data-intensive workloads," in *Proceedings of the 1st ACM Asia-Pacific Workshop on Systems (ApSys2010)*, New Delhi, India, August 2010.

[15] R. Nathuji, A. Kansal, and A. Ghaffarkhah, "Q-clouds: Managing performance interference effects for qos-aware clouds," in *Proceedings of the ACM European Society in Systems Conference 2010*, Paris, France, April 2010, pp. 237–250.

[16] X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, P. Padala, and K. Shin, "What does control theory bring to system research?" *ACM SIGOPS Operating Systems Review*, vol. 43, no. 1, pp. 62–69, January 2009.

[17] J. Heo, X. Zhu, P. Padala, and Z. Wang, "Memory overbooking and dynamic control of xen virtual machines in consolidated environments," in *Proceedings of the IFIP/IEEE Symposium on Integrated Management (IM'09) mini-conference*, Long Island, NY, June 2009, pp. 630–637.

[18] J. Arnaud, "Automated control of internet service," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 3, pp. 47–52, July 2010.

[19] C. Poussot-Vassal, M. Tanelli, and M. Lovera, "Linear parametrically varying mpc for combined quality of service and energy management in web service systems," in *Proceedings of the 2010 American Control Conference (ACC'10)*, Baltimore, MD, June 2010, pp. 3106–3111.

[20] Z. Wang, N. Tolia, and C. Bash, "Opportunities and challenges to unify workload, power and cooling management in data centers," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 3, pp. 41–46, July 2010.

[21] Y. Chen, A. Das, W. Qin, A.Sivasubramanian, and Q. W. N. Gautam, "Managing server energy and operational costs in hosting centers," in *Proceedings of the ACM Join International Conference on Measurement and Modeling of Computer Systems SIGMETRICS'05*, June 2005.

[22] C. Lefurgy, X. Wang, and W. Ware, "Server-level power control," in *IEEE International Conference on Autonomic Computing ICAC'07*, Jacksonville, FL, July 2007.

[23] R. Ayoub, S. Sharifi, and T. S. Rosing, "Gentlecool: Cooling aware proactive workload scheduling in multi-machine systems," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010*, Dresden, 2010, pp. 295–298.

[24] N. Tolia, Z. Wang, P. Ranganathan, C. Bash, M. Marwah, and X. Zhu, "Unified power and cooling management in server enclosures," in *Proceedings of the ASME/Pacific Rim Electronic Packaging Technical Conference and Exhibition (InterPACK '09)*, San Francisco, CA, july 2009.

[25] Y. Fu, C. Lu, and H. Wang, "Robust control-theoretic thermal balancing for server clusters," in *IEEE International Parallel and Distributed Processing Symposium (IPDPS'10)*, April 2010.