DiBA: Distributed Power Budget Allocation for Large-Scale Computing Clusters

Masoud Badiei[1]
SEAS
Harvard University
mbadieik@seas.harvard.edu

Xin Zhan[2]
School of Engineering
Brown University
xin_zhan@brown.edu

Reza Azimi
School of Engineering
Brown University
reza_azimi@brown.edu

Sherief Reda
School of Engineering
Brown University
sherief_reda@brown.edu

Na Li
SEAS
Harvard University
nali@seas.harvard.edu

***Abstract*—Power management has become a central issue in large-scale computing clusters where a considerable amount of energy is consumed and a large operational cost is incurred annually. Traditional power management techniques have a centralized design that creates challenges for scalability of computing clusters. In this work, we develop a framework for distributed power budget allocation that maximizes the utility of computing nodes subject to a total power budget constraint.**

To eliminate the role of central coordinator in the primal-dual technique, we propose a distributed power budget allocation algorithm (DiBA) which maximizes the combined performance of a cluster subject to a power budget constraint in a distributed fashion. Specifically, DiBA is a consensus-based algorithm in which each server determines its optimal power consumption locally by communicating its state with neighbors (connected nodes) in a cluster. We characterize a synchronous primal-dual technique to obtain a benchmark for comparison with the distributed algorithm that we propose. We demonstrate numerically that DiBA is a scalable algorithm that outperforms the conventional primal-dual method on large scale clusters in terms of convergence time. Further, DiBA eliminates the communication bottleneck in the primal-dual method. We thoroughly evaluate the characteristics of DiBA through simulations of large-scale clusters. Furthermore, we provide results from a proof-of-concept implementation on a real experimental cluster.

***Index Terms*- Distributed algorithm, Primal-dual algorithm, Power management, Consensus algorithm.**

I. Introduction

The advent of cloud computing and new online services such as software as a service (SaaS) over the past decade demands a significant scaling of computing clusters. This demand has led to the development of very large-scale clusters that rely on scale-out models (horizontal scaling), which expand capabilities by incorporating new server nodes rather than increasing the capabilities of existing servers [1]. Furthermore, there is a keen interest in dynamically controlling the total power consumption of clusters to enable participation in demand response energy programs and to improve energy efficiency in the presence of variations in cluster utilization.

Incorporating new server nodes is only practical if the computing cluster has a modular architecture. Currently used power management methods are not aligned with such an architecture as they have a centralized design, *i.e.*, there is a central controller aggregating information from all active computing nodes to allocate power efficiently [2]. Specifically, in conventional power management techniques, local data of all computing nodes, such as workload, application priority, and thermal footprint, are constantly monitored to efficiently allocate power budget among them in a centralized manner. Therefore, adding new computing nodes to a cluster necessitates reconfiguration of power management hardware and software which drives up the scaling cost.

To tackle this challenge, we propose a distributed power budgeting framework that scales gracefully for large clusters. In particular, our contributions are as follows:

- We propose DiBA, a novel algorithm for distributed power budget allocation to maximize the combined utility of the entire cluster subject to a given total power budget. The utility of each computing node is a metric for its throughput that depends on the benchmark of the workload as well as server specifications. In our distributed framework, each server exchanges its decision to increase/decrease its power usage with neighboring nodes in the cluster through rounds of communications. After each round, the local parameters of each node are updated in the form of a state-space model. In this phase, the local actions of neighbors are implemented as a control input. Further, the actions of servers take into account the global constraint that must be satisfied.
- We compare the performance of DiBA with a centralized power budgeting scheme in terms of total and per iteration communication/computation time. We note that there is a communication bottleneck at the central coordinator in the centralized method for large clusters. The primal-dual (PD) method also suffers from the same issue since it relies on the same architecture to disseminate information among nodes. In contrast, DiBA provides a fully decentralized mechanism which eliminates the role of the central coordinator. As a result, we demonstrate that for large scale clusters with thousands of nodes, the communication time of DiBA is substantially less than that of centralized scheme and primal-dual method.
- We also examine the impact of a number of issues on the performance of DiBA, including dynamic workloads and dynamic power budgets. We also analyze the impact of topology of communication among the distributed computations, and we show that DiBA can provide fast convergence with topologies as simple as a ring.
- We also implement a working prototype of DiBA on a

[1,2] These authors contributed equally to this work.

real experimental cluster and demonstrates its ability in meeting dynamic total power budget in a fully distributed fashion.

The organization of this paper is as follows. Section II provides a brief overview of related work. Section III motivates the need for distributed power management techniques, and Section IV provides the detailed of our proposed DiBA algorithm. Our experimental results are given in Section V. Finally, we summarize our conclusions and directions for future work in Section VI.

II. Related Works

The problem of efficient power management in computing clusters has been studied extensively from different perspectives, for a recent survey see [3]. Different practices can be applied to reduce power consumption. Nevertheless, they can be divided into one of the following categories:

- Schemes based on dynamic voltage/frequency scaling (DVFS) [2], [4], [5], [6], [7].
- Load balancing and workload scheduling techniques [8], [9].

For memory-intensive workloads, dynamic voltage and frequency scaling (DVFS) during memory-bound phases of execution have been shown to provide power savings with minimal impact on the quality of service (QoS) [10]. Based on this premise, the power management framework we present in this paper utilizes the DVFS technique and thus belongs to the first class. Using the same technique, Beloglazov *et al.* [4] have proposed some heuristics for energy management under strict service level agreement (SLA) constraint. Specifically, by dynamically migrating virtual machines (VMs) across physical machines, workloads are consolidated and subsequently idle resources are put on a low-power state using DVFS. In conjunction with the DVFS technique, Elnozahy *et al.* [5] have evaluated five different policies for cluster-wide power management in server farms. In particular, in the independent voltage scaling policy, the authors consider a mechanism in which each node independently manages its own power consumption using dynamic voltage scaling. Nevertheless, this policy is different from our proposed distributed scheme (*i.e.* DiBA) in that there is no hard constraint on the power budget of the entire cluster and therefore, there is no need for coordination among computing nodes to meet the power constraint.

In the same line of work on DVFS techniques, distributed power management schemes have also been studied [6]. Specifically, in [6], the problem of minimizing the power consumption in a three tier computing cluster under the end-to-end SLA constraint has been investigated. To solve this problem, two strategies have been investigated. In the first strategy dubbed as *OptiTuner*, a primal-dual decomposition method has been employed where each tier of servers updates its power consumption by communicating with a central unit. Also, as the second strategy, a Linear-quadratic regulator (LQR) control method has been employed. Despite having a distributed structure, OptiTuner suffers from the Single Point of Failure (SPF) problem. That is, it requires a central node to coordinate among computing nodes, and therefore, is susceptible to node failure. Moreover, OptiTuner has a homogeneous power distribution at the tier level, *i.e.*, all machines at each tier are assumed to operate at the same frequency. In contrast, DiBA optimizes the performance of each individual server node and thus results in a better overall performance.

Along the same line of work, a centralized, feedback based, power controller for satisfying end-to-end delay is also studied [2]. In the proposed scheme, a central application performance monitor measures the latency of delay sensitive workloads. It then passes the information to a centralized DVFS controller to adjust allocated power for each server. Clearly, such a centralized scheme does not suite large-scale clusters as in addition to SPF problem, the centralized controller must solve a complex, high-dimensional optimization problem to determine the power usage of each computing node.

In the context of load balancing techniques for power management, a "bidding" mechanism akin to the economic models for managing shared resources has been studied [8]. In the proposed framework, services bid for resources and negotiate for SLA based on the offered price and their QoS requirement. Therefore, SLA is assumed to be a flexible parameter. While load balancing improves the system performance, load concentration (unbalancing) can save energy by making servers idle and thus consume less energy [9]. To optimize the trade-off between performance versus power saving, a load balancing and unbalancing algorithm is proposed [9] that takes into account both the total load imposed on the cluster and the power and performance implications of turning nodes off.

III. Motivation

Large clusters are often constrained by power consumption due to the large operational costs for energy consumption. Furthermore, dynamic constraints such as those arising from demand response energy programs or from variations in utilization require cluster operators to find ways to meet changing total power budget throughout daily operation.

Although current employed power management techniques are effective in their objective of reducing power usage, they have a centralized design. Those techniques require a central controller to measure their workload and throughput and determine the optimal DVFS of processors of active servers. Nevertheless, a centralized approach has the following shortcomings:

- *Fault isolation*: Centralized control creates a Single Point of Failure (SPF) and also a performance bottleneck. In particular, the malfunction in controller or communication breakdown between controller and local servers renders the SLA violations. To avoid such a scenario in centralized methods, the local servers are set to automatically default to maximum power [2]. We present a decentralized method in which each server determines its action locally. Thus, the failure in one or few servers or the communication breakdown can be mitigated as the overall performance of the system does not hinge on a particular unit.

- *Real-time monitoring*: In centralized power budgeting techniques, the state of each server is measured by a centralized monitoring unit. This introduces additional latency and deteriorates the quality of service (QoS) due to data aggregation and communication delay. In comparison, the decentralized technique obviates the centralized monitoring unit as each computing node measures its performance metrics locally. For variable internet traffic such as search and social networking, the localized performance monitoring provided by the decentralized control approach results in a faster response time to workload and power management.
- *Scalability*: A centralized power management is an inflexible architecture for horizontal scaling, in the sense that adding new racks to the cluster must be complemented with the increase in the communication bandwidth and power management reconfiguration of the central power controller. In comparison, a decentralized architecture facilitates a modular design. Particularly, DiBA can scale to large clusters with distributed communication requirements that are as simple as a ring and that are carried over regular cluster network infrastructure.

IV. Power Management Methodology

We formulate an optimization problem for maximizing the throughput of a cluster with N server nodes, where we assume a power budget of P for the whole cluster. Thus,

$$\textbf{Problem 1}: \max_{\{p_i\}_{i=1}^N} \sum_{i=1}^N r_i(p_i) \tag{1}$$

$$\sum_{i=1}^N p_i \le P, \tag{2}$$

$$p_i^{\text{idle}} \le p_i \le p_i^{\max}, \quad i = 1, 2, \cdots, N, \tag{3}$$

where p_i^{idle} and $p_i^{\max}$ denote the power that is used by each server in the idle mode and the maximum possible power usage of server i, respectively. We use the simple structure of Problem 1 to exhibit our distributed optimization schemes clearly. In this formulation, the utility metric $r_i(\cdot) : \mathbb{R} \to \mathbb{R}_{\ge 0}$ corresponds to the throughput of each server node that can be increased/decreased by adjusting its capacity (e.g., increasing/decreasing the frequency of the processors using DVFS). We note that the throughput of server i as a function of its power usage p_i depends on the characteristic of the workload that is being executed. We defer further details about modeling a proper throughput function to Section V.

Solving the optimization Problem 1 in a decentralized fashion requires coordination among servers, since the power consumptions of servers are coupled via the constraint (2) in Problem 1. To achieve coordination among nodes, our distributed algorithm creates a consensus framework for each node.

In this section, we consider two distributed schemes that are structurally different. First, we analyze the more conventional primal-dual decomposition approach and discuss its limitations for practical implementation in large clusters. We then propose an alternative algorithm, DiBA, that is fully distributed that outperforms the primal-dual method in terms of convergence speed.

A. Primal-dual decomposition algorithm

The algorithm that we present here is based on the primal-dual decomposition method which is a well-known scheme for distributed optimization and has been extensively studied in the context of TCP/IP and congestion control in networks with elastic traffic [11]. We also remark that the primal-dual decomposition method has been investigated in [6] in the context of power management of computing clusters. However, we use this algorithm as the benchmark for comparison with the performance of DiBA we present in subsequent subsections. Hence, we give a detailed analysis of the primal-dual method in the following.

We describe the simple structure of the asynchronous primal-dual algorithm when it is characterized for Problem 1. Consider the dual form of the optimization problem that is formulated in (1)-(3),

$$\min_{\lambda \ge 0} g(\lambda) \tag{4}$$

where λ is the dual variable, and $g(\lambda)$ is defined as

$$g(\lambda) \stackrel{\text{def}}{=} \sup_{\{p_i\}_{i=1}^N} \sum_{i=1}^N r_i(p_i) + \lambda \left(P - \sum_{i=1}^N p_i \right).$$

Let $\lambda^\star$ be the optimal solution of the dual optimization problem (4). Given the optimal Lagrangian multiplier $\lambda^\star$, each server can compute its optimal power budget $p_i^\star, i \in \{1, 2, \cdots, N\}$ via solving the following *local* optimization problem

$$p_i^\star = \arg \max_{p_i^{\text{idle}} \le p_i \le p_i^{\max}} r_i(p_i) - \lambda^\star p_i.$$

In this formulation, $\lambda^\star$ can be interpreted as the price of using the shared power budget P. In particular, in the case that power demands exceed the amount of shared power budget P, the central coordinator increases the price $\lambda^\star$ to reduce the demand.

A simple scheme to compute the optimal value of $\lambda^\star$ can be achieved by an iterative update rule in which the price λ is computed by a coordinator unit, and then communicated to all local servers. In particular, the coordinator updates the price using the following update formula; see [11],

$$\lambda^{t+1} = \left[\lambda^t - \epsilon \left(P - \sum_{i=1}^N p_i^t \right) \right]^+, \tag{5}$$

where $[z]^+ \stackrel{\text{def}}{=} \max\{0, z\}$, $\epsilon > 0$ is a small step size.

By having the Lagrangian update λ^{t+1}, the i-th server can compute its power consumption $p_i^\star$ as follows

$$p_i^{t+1} = \arg \max_{p_i^{\text{idle}} \le p_i \le p_i^{\max}} r_i(p_i) - \lambda^t p_i. \tag{6}$$

The structure of the primal-dual decomposition algorithm for Problem 1 is described in Algorithm 1.

We make few remarks about Algorithm 1.

Algorithm 1 PRIMAL-DUAL ALGORITHM

1: **Initialize** $p_i^0, i = 1, 2, \cdots, N$, λ^0, and $\epsilon \in \mathbb{R}_{\geq 0}$.
2: **for** $t = 1, 2, \cdots$ **do**
AT THE CENTRAL COORDINATOR:
3: update λ^t according to (5)
AT THE i^{th} SERVER:
4: update p_i^t according to (6)
5: transmit p_i^t to the central coordinator.
6: **end for**

1) Although the primal-dual decomposition method is scalable, it is not efficient in terms of communication requirements. For large clusters, the central coordinator must be equipped with a high communication bandwidth to accommodate communication with all servers. Moreover, since network switches support a limited number of servers, for a large cluster of thousands server nodes, multiple network switches must be employed which is costly and adds additional latency to the network.
2) The existence of a central coordinator in the primal-dual approach is also undesirable since it creates the SPF problem. In the case of failure of the central coordinator, the PD power management technique fails. Therefore, PD does not provide the robustness that is necessary to power management in data centers.
3) The PD algorithm we considered is synchronized in the sense that servers update their power usage information simultaneously. This level of synchronization between different servers is usually provided through Network Time Protocol (NTP) in computer systems [12].

B. DiBA: Distributed Power Budget Allocation Algorithm

In this section we propose and analyze our proposed distributed algorithm called DiBA to control the power consumption of each computing node to maximize the throughput of the cluster subject to total power budgets. We can qualitatively describe DiBA from two different perspectives. From a game design perspective, DiBA is a gradient algorithm that achieves the pure Nash equilibrium corresponding to the state based game defined by Problem 1 [13].

From the view point of distributed optimization, DiBA is a consensus based optimization algorithm. In particular, DiBA establishes the consensus on the estimation from the coupling constraint (2) by augmenting the local utility function of each server node with a barrier (penalty) function which when minimized/maximized, gradually reduces the difference between local variables of computing nodes.

Naturally, to create such a consensus, the communication graph between servers needs to be connected. The connectivity requirement is naturally facilitated in clusters, where clusters can communicate using the cluster's network infrastructure.

For each i-th server, we introduce the estimation e_i term for the coupled constraint (2), i.e.,

$$e_i \sim \sum_{i=1}^{N} p_i - P. \tag{7}$$

During each round of DiBA algorithm, the i-th server transmits its local measurement and also receives the local variables of all its neighbors. Specifically, lets $\hat{e}_{i\to k}$ ($\hat{e}_{i\leftarrow k}$) denote the estimates from (resp. to) the i-th server to (resp. from) the k-th server, where $k \in \mathscr{S}_i$ and $\mathscr{S}_i \subseteq \{1, 2, \cdots, N\}$ denotes the set of servers that are connected to the i-th server.

By aggregating these estimates, the i-th server updates its state variable as well as its estimate from the coupled system constraint. Specifically, based on a state-space control model, we have the following update rule during the t-th iteration of DiBA algorithm,

$$p_i^{t+1} = p_i^t + \hat{p}_i^t \tag{8}$$

$$e_i^{t+1} = e_i^t + \hat{p}_i^t + \sum_{k\in\mathscr{S}_i} \hat{e}_{k\leftarrow i}^t - \sum_{k\in\mathscr{S}_i} \hat{e}_{i\to k}^t, \tag{9}$$

where p_i^t is the power state of the i-th server, and $\hat{p}_i$ is the power control input that takes values from the action space

$$\mathscr{A}_i(p_i^t, e_i^t) \stackrel{\text{def}}{=} \Bigg\{ (\hat{p}_i, \hat{e}_i) : p_i^t + \hat{p}_i \in [p_i^{\text{idle}}, p_i^{\max}], \hat{e}_{i\to k} < 0, \\ e_i^t + \hat{p}_i^t - \sum_{k\in\mathscr{S}_i} \hat{e}_{i\to k} < 0 \Bigg\}.$$

The action space guarantees that the controller inputs $(\hat{p}_i, \hat{e}_i)$ in the state-space model does not violate the constraint on the maximum power and idle power of each computing node, and also satisfy the coupled constraint (2). For each node, we now define a local utility function for node $i = 1, 2, \cdots, N$ as follows

$$R_i(p_i^t, \{e_j^t\}_{j\in\mathscr{N}_i}) = r_i(p_i^t) - \eta \sum_{j\in\mathscr{S}_i} \log(-e_j^t), \tag{10}$$

where $\eta \in \mathbb{R}_{\geq 0}$. The second term $\sum_{j\in\mathscr{S}_i} \log(-e_j^t)$ is a penalty factor that plays a role on the cost function only when $e_j^t > 0$, which translates into the constraint violation condition $\sum_{i=1}^{N} p_i^t > P$.

We note that the speed of reaching the consensus is controlled by the value of η. By choosing a small positive value for η in (10), we can assure that the solution of DiBA is close to the optimal solution of Problem 1. But, choosing a small η will cause numerical difficulty which may lead to slower convergence.

In addition, the barrier function in (10) enforces the constraint to be satisfied. By employing this barrier function, we thus assure that the resultant answer from the above algorithm fulfills the system power constraint. We now have all the machinery to describe DiBA. The steps of DiBA are highlighted in Algorithm 2.

It is imperative to calculate the communication overhead introduced from decentralizing the power management controller. The communication overhead of each control epoch can be approximated by the product of the number of communications required during each round and the number of iterations to converge. In a cluster with N servers,

Algorithm 2 DISTRIBUTED POWER BUDGET ALLOCATION (DIBA)

1: **Initialize** $\eta \in \mathbb{R}_{\geq 0}$ and a non-increasing sequence $\epsilon_i^t \in \mathbb{R}_{\geq 0}$, $i = 1, 2, \cdots, N$. Choose p_i^0, e_i^0 such that $\sum_{i=1}^N p_i^0 \leq P$.
2: **for** $t = 1, 2, \cdots$ **do**
AT THE i^{th} SERVER:
3: compute

$$\hat{p}_i^t = \beta^t \left(-\epsilon_i^t \frac{\partial R_i}{\partial \hat{p}_i}\Big|_{\hat{p}_i=0} \right)$$

$$\hat{e}_{i\to j}^t = \beta^t \min \left(0, -\epsilon_i^t \frac{\partial R_i}{\partial \hat{e}_{i\to j}}\Big|_{\hat{e}_{i\to j}=0} \right),$$

where β^t is computed by backtracking to the constraint $(\hat{p}_i^t, \hat{e}_i^t) \in \mathscr{A}_i(p_i^t, e_i^t)$, and ϵ_i^t is the gradient step size.
4: update p_i^{t+1}, e_i^{t+1} according to (8)-(9).
5: **end for**

1) for primal-dual decomposition algorithm, each computing node sends one packet and then receives one packet from the central coordinator during each iteration. Thus, total number of $2N$ packets are communicated per iteration.
2) for DiBA, each computing node sends packets only to its local neighbors which in case of a ring communication topology is two and thus results in the total number of $2N$ packets that are communicated in parallel among the nodes. For a general graph, dN communications per iteration is required where d is the average degree of each node.

Although it may appear that DiBA has the same communication complexity as the primal-dual method, in practice it has a lower communication latency and packet loss. In particular, if we consider the network overhead in terms of the bottleneck of communication topology, the central coordinator in the primal-dual decomposition method must communicate with all N computing servers which has a degree of N. In comparison, the degree of each node in DiBA is a small fixed number (e.g., two in case of ring topology), the packets from the nodes to their neighbors proceed in parallel. As a result, packet loss is less likely to happen in DiBA architecture. We will show in the next section that DiBA also shows better performance compared to the primal-dual decomposition method in terms of communication latency.

V. NUMERICAL EVALUATION

A. Setup

Workloads: To evaluate the performance of our distributed power management architecture, we select 10 HPC benchmarks. Eight benchmarks from NAS Parallel Benchmarks (NPB) suite [14] (`BT`, `CG`, `EP`, `FT`, `IS`, `LU`, `MG`, and `SP`) and two benchmarks from High Performance Computing Challenge (HPCC) benchmark (`HPL`, `MPI-RA`) and the description of each benchmark is listed in Table I.

name	description
`BT` (NPB)	Block Tri-diagonal solver
`CG` (NPB)	Conjugate Gradient
`EP` (NPB)	Embarrassingly Parallel
`FT` (NPB)	discrete 3D fast Fourier Transform
`IS` (NPB)	Integer Sort
`LU` (NPB)	Lower-Upper Gauss-Seidel solver
`MG` (NPB)	Multi-Grid on a sequence of meshes
`SP` (NPB)	Scalar Penta-diagonal solver
`HPL` (HPCC)	High performance Linpack benchmark
`RA` (HPCC)	Integer random access of memory

TABLE I
LIST OF THE SELECTED BENCHMARKS AND CORRESPONDING DESCRIPTION.

Infrastructure: We evaluate DiBA using numerical simulations and using a real server cluster. The numerical simulations enable us to study DiBA's performance for large-scale clusters, while the real cluster implementation demonstrates a proof-of-concept for DiBA.

1) **Server cluster infrastructure**: Our cluster consists of 12 Dell PowerEdge C1100 servers, where each server has two Xeon L5520 quad-core processors, 40 GB of memory and 10 Gbe network controller. The servers are connected to a Mellanox SX1012 10Gbe switch. The frequency of each processor can scale from 1.60 GHz to 2.27 GHz. To collect performance counters values from all servers, the `perfmon2` [15] tool is used. We created a power measurement appartus to measure the supplied AC current to each server and its power consumption correspondingly. To enforce the local power targets for the individual servers, we implement a software feedback controller on each server. The DVFS-based controller adjusts the DVFS up or down according to the difference between the power target and the current power consumption, with positive difference triggering an increase in DVFS, while negative differences triggering a decrease in DVFS [16]. The controller enables us to apply the local constraints on each server as derived from the DiBA.
2) **Simulation infrastructure**: To evaluate the performance and overhead of DiBA on large clusters, we have developed a simulation tool as follows. We first run each workload in Table I on one of our servers and collect the throughput and power consumption p_i of ith workload under various DVFS levels, and then fit a throughput function $r_i(p_i)$ on the attained data. The power consumed by each workload has a range that is determined by the lowest and highest DVFS levels. The throughput functions are then used to emulate the behavior of the applications running on the server nodes. To simulate a cluster with arbitrary number of servers N, we draw the throughput functions from a uniform distribution such that each server hosts at least one type of workload and such that the entire server cluster is fully utilized. To mitigate the effect of randomness in our simulation results, we average the convergence results we obtain for DiBA over 10 trials. The network architecture of our simulation environment is based on a two-tier star topology. Each rack consists of 40 servers which are

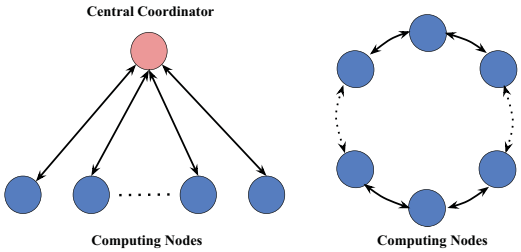

Fig. 1. The communication topology of the decentralized algorithms. Left: Primal-Dual Decomposition. Right: DiBA algorithm.

connected with one top-of-rack 10 Gbe Ethernet switch. Further, all racks are connected via a higher-layer core switch. We model the network traffic using queuing systems in our simulation, where the network service times are derived from measurements on our real cluster.

Algorithm Topology: For the primal-dual method, each computing node is connected with the central coordinator as shown in Fig. 1. Furthermore, for DiBA algorithm, we adopt a ring communication network as shown in Fig. 1. Note that the algorithm communication topology runs on top of our cluster network, which naturally facilitates any two servers to communicate.

Throughput Estimation: The throughput function $r_i(p_i)$ of each server is dependent on the nature of the workload. Moreover, we create a framework in which the local controller learns the throughput function $r_i(\cdot)$ on-the-fly as the workload changes on the server. Specifically, we can learn the throughput function of each workload in our benchmark suite of Table I by applying different DVFS power levels and subsequently measuring the resultant throughput value. We then interpolate a quadratic throughput function. Fig. 2 illustrates a number of throughput functions for our workloads on our servers. This process is repeated for each workload to derive its throughput function when it is executed on a server. As we observe from Fig. 2, the throughput functions for all the benchmarks are concave in power which justifies the use of convex optimization toolbox `CVX` in our simulations.

System Performance metrics: To ensure fairness in our power allocation scheme, we normalize the throughput of each workload by its peak value of r_i^{max}. The application normalized performance (ANP) of each server is computed, where ANP is the ratio of ideal runtime to actual runtime of the workload on the server [17]. In our case, the ANP of application i under power budget p_i is defined as the ratio of actual throughput to its peak throughput:

$$\text{ANP}_i(p_i) = \frac{r_i(p_i)}{r_i^{max}}$$

Then to quantify the performance of the entire cluster, the system normalized performance (SNP) is computed as the arithmetic mean of the ANPs of all M workloads that are running on the cluster. That is,

$$\text{SNP} = \frac{\sum_{i=1}^{M} \text{ANP}_i(p_i)}{M}.$$

We next describe the results from our simulation infrastructure in Subsection V-B and the results from our implementation on the real cluster in Subsection V-C.

B. Simulation results

To evaluate the performance, scalability and resiliency of DiBA, we organize the following experiments.

1. Static experiment of allocating power budget to maximize the system's performance.
2. The scalability of the DiBA on clusters with large number of nodes, considering the communication and computation overhead.
3. Dynamic simulations that evaluate how DiBA reacts to the power budget reallocation and workload dynamics.
4. Experiments that evaluate the effect of communication topology of the distributed computations of DiBA.

1. Static power budgeting results: In this experiment we demonstrate the value of power budget algorithms that use the utility functions of the servers to maximize performance subject to a total power budget. We simulate 1000 servers that are serving 1000 instances of the benchmarks under total power budgets that range from $P = 166$ kW to $P = 186$ kW with the incremental step size of 4 kW. The results of our

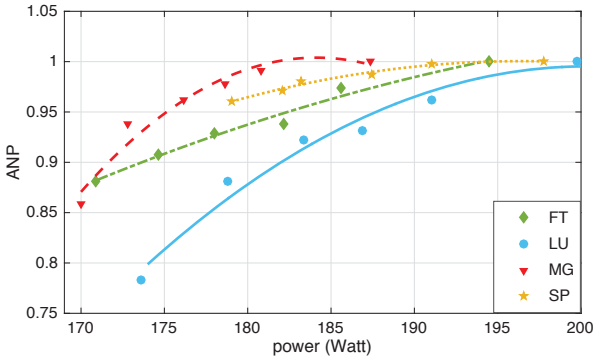

Fig. 2. The normalized throughput function of 4 workloads.

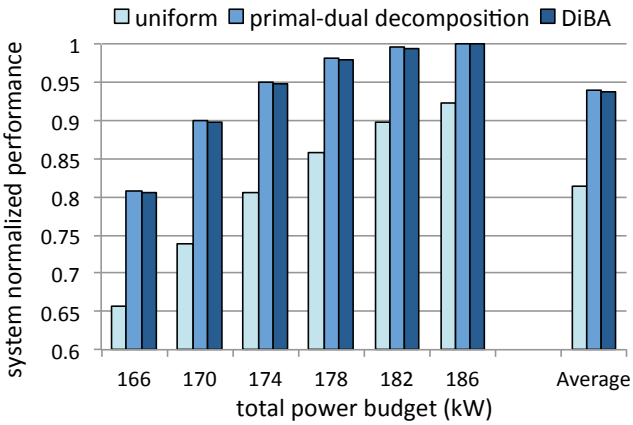

Fig. 3. The system normalized performance of 1000 servers under different power budgets.

# of nodes	centralized		primal-dual decomposition		DiBA	
	Computation (ms)	Communication (ms)	Computation (ms)	Communication (ms)	Computation (ms)	Communication (ms)
400	432.76	86.25	1.47	517.52	0.07	28.00
800	477.25	170.37	1.44	1022.22	0.11	26.20
1600	693.40	339.83	1.41	2038.99	0.43	28.00
3200	1041.50	675.95	1.55	4055.70	0.80	27.20
6400	1882.52	1362.50	1.58	8175.00	1.78	28.40

TABLE II
THE BREAKDOWN OF ALGORITHM RUNTIME WITH DIFFERENT NUMBER OF SERVER NODES.

decentralized algorithms are compared against the method of allocating power budget uniformly. The SNP results under each total power budget are given in Fig. 3. The primal-dual decomposition algorithm improves the SNP by 14.7% over uniform power allocation in average, while DiBA achieves 14.5% improvement. The performance gap between uniform and DiBA shrinks from 22.6% to 8.2% depending on the amount of total power allocated. Clearly, when the total power budget is sufficiently large, all servers receive enough power for their workloads, and hence the performance differences between DiBA and uniform power allocation diminishes.

2. Scalability: To show the advantage of DiBA for large cluster sizes, we compare its performance to that of the centralized approach and primal-dual method. For centralized method, we use `CVX` toolbox [18] to solve the optimization problem formulated in Eqs. (1)-(3). Specifically, in the centralized method, the computing servers transmit their utility functions to the centralized power management unit to calculate the optimal allocation of the total power budget across all servers. Then the power budget of each server is sent back and applied by the servers. By decentralizing the power budgeting unit, the utility functions are optimized locally and the optimization is accelerated. However, to assess the optimization performance, the communication latency between of each iteration has to be taken into account. In this experiment, we quantify the runtime of each algorithm with different number of server nodes. The average latency of reading and write a packets to TCP sockets between two nodes in our cluster is measured to be approximately 200μs and 10μs respectively. We used these results as the service times in our network queuing model, which is used to model the communication time at the central coordinator in the centralized and primal-dual methods. In the 'uplink' phase, when nodes transmit their packets to the central coordinator, the packets arrival time are drawn from the Poisson distribution with average inter-arrival time of 200μs. In the 'downlink' phase, *i.e.*, transmitting from the central coordinator to the nodes in the centralized and primal-dual methods, the communication time is assumed to be the total time to send all the packets in a serial fashion. In contrast, in DiBA architecture, each node only communicates with two neighbor nodes in parallel, thus the communication time for each iteration is time to send and receive a packet.

We compute the runtime of each algorithm with different number of nodes and break the total runtime between computation time and communication time in Table II. We take the result from `CVX` solver as the base line and set the convergence condition of the primal-dual decomposition algorithm and DiBA as achieving 99% of the base line performance. In particular, the initial power setting for DiBA is set as uniform allocation. The convergence time for DiBA is the first time that the utility function obtained from DiBA satisfies the following inequality

$$\frac{|\text{Optimal Utility} - \text{Utility of DiBA}|}{|\text{Optimal Utility}|} < 0.01, \qquad (11)$$

where the optimal utility is the utility that is computed by solving (1) centrally. The convergence of PD algorithm is defined similarly.

The total computation and communication time of the primal-dual decomposition and DiBA are computed by taking into account the number of iterations required for each algorithm to converge. The communication time of each iteration is modeled by our simulation network architecture and the computation time of each iteration is the real measurement from our simulation system. DiBA is completely distributed and all communications and computations are assumed to be carried out in parallel for each iteration. For primal-dual, the computation of all the nodes are done in parallel but need to be added to the computation time of the central coordinator.

From Table II, we observe that PD method improves upon the centralized power allocation strategy in terms of computation time. This is due to the fact that in the PD method, the optimization problem is solved in a decentralized fashion. Nevertheless, in the overall performance, the PD method performs poorly when the communication time is taken into account. As we mentioned in Section IV-A, this increase in latency is due to the communication bottleneck at the central coordinator.

In contrast, we observe that the communication time of DiBA on a ring network increases only slightly. Consequently, DiBA outperforms both the PD and the centralized method in the overall run-time. DiBA also outperforms both the primal-dual and centralized method in the computation time for small to mid range size clusters.

3. Dynamic simulation results: In the previous experiment, we considered a static power budget. Here, we examine a dynamic case where the power budget changes every minute during operation as possibly the case in demand-response programs. We demonstrate how DiBA re-allocates the power budget. Specifically, we evaluate DiBA for cluster of $N = 1000$ servers with dynamic power budget in Fig. 4. We observe that DiBA provides a near optimal performance without power budget violation. To demonstrate a more detailed response of

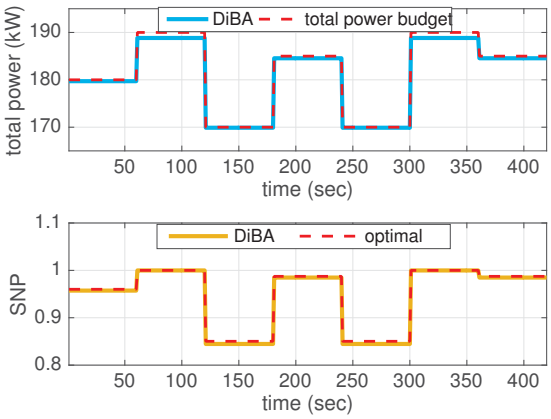

Fig. 4. Dynamic simulation of total power budget reallocation.

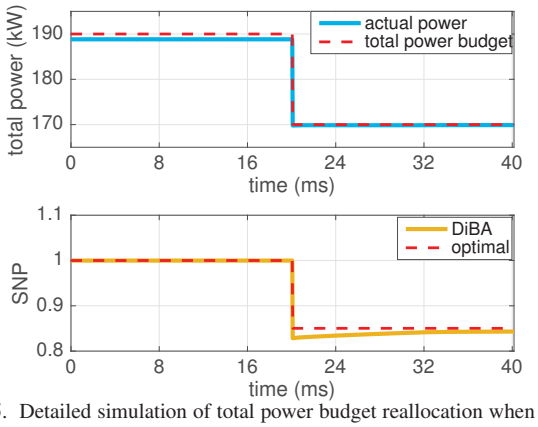

Fig. 5. Detailed simulation of total power budget reallocation when total power budget drops from high to low.

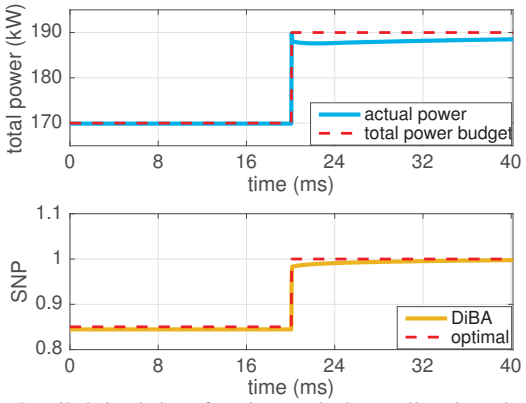

Fig. 6. Detailed simulation of total power budget reallocation when total power budget jumps from low to high.

DiBA when there is an adjustment in the total power budget, we consider cases where the power budget decreases/increases between $P = 190$kW and $P = 170$kW power levels as depicted in Fig. 5 and Fig. 6, respectively. In both cases, DiBA immediately adapts to the new power budget.

DiBA is also adaptive to workload variations on servers. DiBA dynamically recomputes the power usage of each server as the workloads change on the servers. We simulate a cluster with $N = 1000$ servers at a fixed total power budget of $P = 180$kW. The servers host different instances of the testing workloads that are presented in Table I. After a workload is completed, a new workload is randomly drawn from the workload pool in Table I and is launched on the free server. We simulated this process for 80 minutes. The resulting power consumption and SNP is given in Fig. 7. We observe that despite variations in the server's workload, the SNP metric of DiBA is close to optimal. Furthermore, our simulation has shown that under variable workload scenario, the total power consumption is strictly below the power limit.

We now analyze a scenario in which the utility of a single server node undergoes an abrupt change in its utility function. This scenario can occur for example when a node serves a workload and begins processing a new workload from a different benchmark. Under this circumstance, it is interesting to observe the effect of workload changes in a single server on the power states of other servers in the network.

First we examine how the estimation e_i of the affected node propagates through the ring network. More precisely, we consider a ring network of size $N = 100$ nodes. Furthermore, we assume that the coefficients of the quadratic utility function of the node $i = 50$ undergoes a sudden change. For this scenario, the estimation vector $\mathbf{e} \stackrel{\text{def}}{=} (e_1, \cdots, e_N)$ is shown in Fig. 8 at different phases of DiBA algorithm. As shown in Fig. 8 initially, all nodes in the network, except for the perturbed node, have negligible error estimations $e_i, i \neq 50$. After few iterations, the estimation error propagates through the network while the magnitude of the absolute error decreases. More interestingly, as depicted in Fig. 9, after convergence to the new equilibrium point, only few nodes in the vicinity of the perturbed server node need to adjust their power consumption states. More specifically, Fig. 9 shows the difference between power usage before and after utility perturbation at the node $i = 50$. It can bee seen that fluctuation in the utility function has a desirable *local* effect on power consumptions of computing nodes.

4. The effect of algorithm communication topology: The communication topology of of the distributed algorithms can be optimized to improve their convergence. A ring topology is particularly ideal for DiBA due to its low degree and symmetry. In practice, to guarantee the connectivity of the distributed algorithm in the case of multiple node failures, the ring topology must be equipped with a few chords.

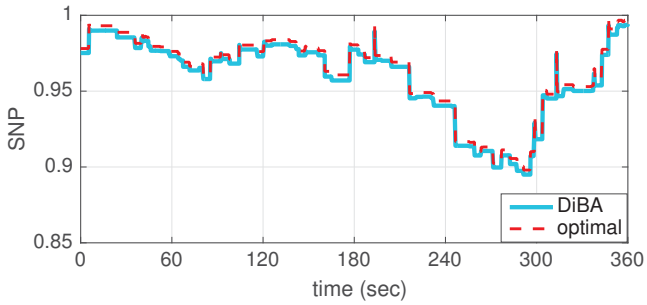

Fig. 7. The performance of DiBA on a cluster with dynamic workloads.

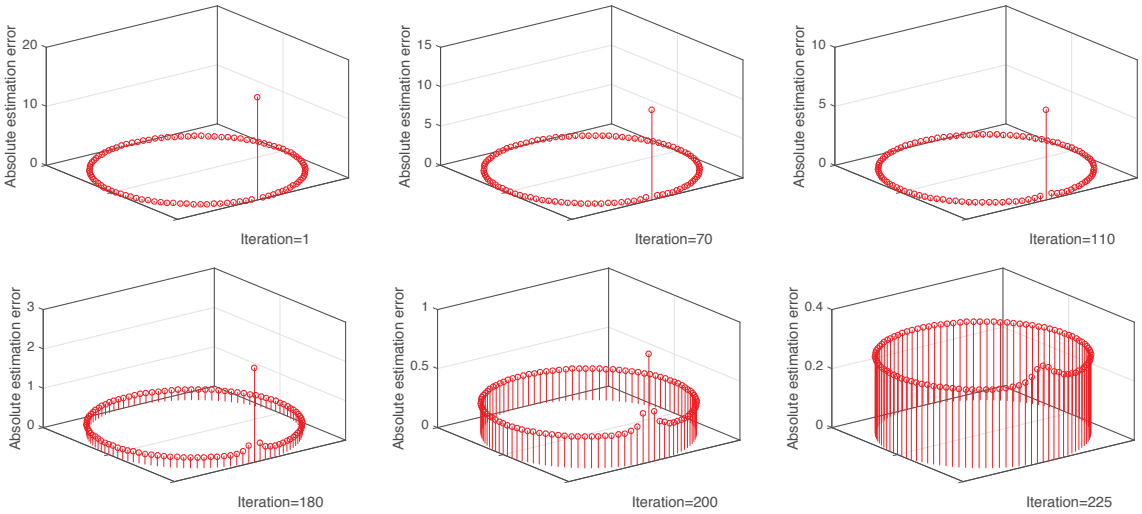

Fig. 8. The *absolute* estimation error of server nodes in the case of utility change of node $i = 50$.

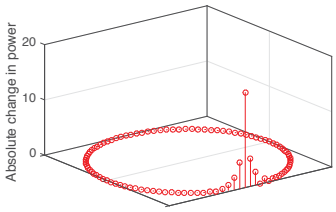

Fig. 9. The *absolute* change in the power consumption states $(p_1, \cdots, p_N)$ after settling at the new equilibrium.

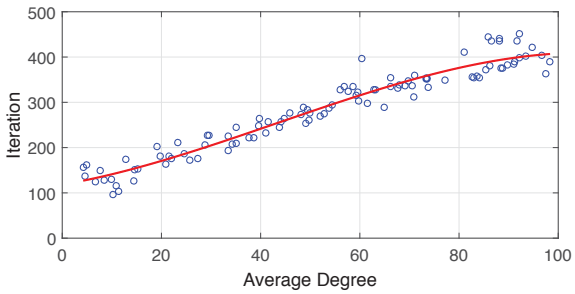

Fig. 10. The iteration numbers of 100 samples of Erdös-Réyni random graphs with $N = 100$ versus the average degree. The red line shows the 3rd order polynomial regression on the samples.

To investigate the effect of connectivity of the distributed computation on the convergence speed of DiBA, we created 100 instances of connected Erdös-Réyni random graphs [19], where the number of computing nodes is fixed at $N = 100$. In the Erdös-Réyni random graph model, a graph is chosen uniformly at random from the set of all possible graphs with N nodes and a certain number of edges. Note that Erdös-Réyni random graph model is degree homogeneous since its *degree distribution* (i.e., the probability of a node to have a certain degree) decays exponentially fast. In our simulation experiment, the number of edges varies which results in graphs with different average degrees per computing node.

For each random graph, we have computed the average degree and the number of iterations. The termination criterion for the iterations is defined to be the time at which DiBA achieves 99% of the optimal utility value (cf. Eq. (11)). The result of this analysis is shown in Fig. 10. We observe that DiBA's convergence time has strong correlation with the average degree of connectivity. The appropriate setting for the average communication degree is then a choice of the system administrator, who can assess the required fault tolerance and adjust the topology of the distributed computation accordingly. It is also worth mentioning the convergence time of DiBA can be adjusted by tuning the free parameters ϵ_i^t as well as μ in the structure of Algorithm 2.

C. Results from Real Cluster Implementation

In this subsection we demonstrate a fully operational implementation of DiBA on our cluster of 12 servers, which are connected using the top-of-the-rack 10 Gbe switch. Each server runs a DiBA power manager that implements our algorithm and the managers communicate with each other using a ring topology with standard socket interfaces. We chose the ring topology since our simulation results identified this simple topology as an effective way to communicate among the DiBA managers with low latency. We also implement a software feedback controller to enforce local power budgets. Once a local DiBA manager computes a local power target, the DVFS-based controller adjusts the DVFS up or down (in increments of 1.3 GHz) depending on the difference between the power target and the current power consumption of the server, with positive difference triggering an increase in DVFS, while negative differences triggering a decrease in DVFS. Our implementation of DiBA takes about 700 lines of C++ code. In our experiments, we launch the MG application on 6 servers, and launch HPL application on the remaining six servers. These two benchmarks were selected because of their different characteristics: HPL is a CPU-intensive benchmark, while MG is a memory-bound application.

To demonstrate that DiBA operates in a practical scenario,

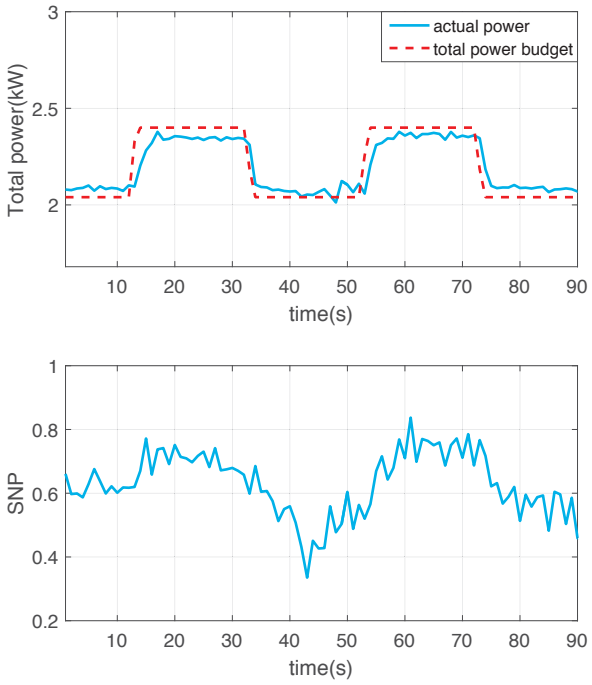

Fig. 11. Real implementation of DiBA on an experimental cluster of 12 servers.

we consider a dynamic power budget to verify that DiBA satisfies the power budget restriction. SNP was calculated by measuring retired floating point operation of the servers. Figure 11 depicts the cluster power consumption and SNP of DiBA as the power budget alternates between 2.0 kW to 2.4 kW over 90 seconds. The figure shows that DiBA is able to meet the total power target. We note the slight oscillations in power measurement are due to the discrete nature of DVFS settings that forces the local DVFS controllers to alternate between two settings to effectively capture an intermediate power level that is otherwise not directly attainable by DVFS. Also the benchmarks have different phases of operations, which result in slight oscillations in power and performance. The SNP results show the expected trend that the performance of the cluster will be reduced when subjected to the lower power budgets.

VI. Conclusions

We studied the problem of utility maximization in large-scale server clusters with the power budget constraint. We proposed DiBA, a framework to compute the optimal power usage of each node locally. In this framework, each node must exchange its power consumption state to the neighboring nodes in the cluster. We investigated various performance metrics associated with DiBA.

In particular, we showed that DiBA outperforms the centralized power budgeting scheme as well the as primal-dual algorithm in terms of computation/communication time. The fast convergence of DiBA is particularly ideal for dynamic workloads as well as dynamic power budget with fast time scales. We demonstrated this characteristic of DiBA through our simulation experiments and experiments on the real cluster using dynamic power budget.

As a future work, it is interesting to modify the structure of DiBA to include delay sensitive applications. In the presence of an end-to-end delay constraint on workloads, computing nodes must coordinate not only their power consumption data, but also their latency information, which leads to a challenging problem formulation.

Acknowledgment

The research of Xin Zhan, Reza Azimi and Sherief Reda is partially supported by NSF grants 1305148 and 1438958. The research of Masoud Badiei and Na Li is supported by NSF CAREER grant 1553407 and Harvard Center for Green Buildings and Cities.

References

[1] A. Vahdat, M. Al-Fares, N. Farrington, R. N. Mysore, G. Porter, and S. Radhakrishnan, "Scale-out networking in the data center," *IEEE Micro*, vol. 30, no. 4, pp. 29–41, 2010.

[2] D. Lo, L. Cheng, R. Govindaraju, L. A. Barroso, and C. Kozyrakis, "Towards energy proportionality for large-scale latency-critical workloads," in *Proceeding of the 41st annual international symposium on Computer architecuture*, pp. 301–312, 2014.

[3] A. Beloglazov, R. Buyya, Y. C. Lee, A. Zomaya, *et al.*, "A taxonomy and survey of energy-efficient data centers and cloud computing systems," *Advances in computers*, vol. 82, no. 2, pp. 47–111, 2011.

[4] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future generation computer systems*, vol. 28, no. 5, pp. 755–768, 2012.

[5] E. M. Elnozahy, M. Kistler, and R. Rajamony, *Power-Aware Computer Systems*. Springer, 2003.

[6] J. Heo, P. Jayachandran, I. Shin, D. Wang, T. Abdelzaher, and X. Liu, "Optituner: On performance composition and server farm energy minimization application," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 11, pp. 1871–1878, 2011.

[7] X. Zhan and S. Reda, "Power budgeting techniques for data centers," *IEEE Transactions on Computers*, vol. 64, pp. 2267–2278, Aug 2015.

[8] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, "Managing energy and server resources in hosting centers," in *ACM SIGOPS Operating Systems Review*, vol. 35, pp. 103–116, 2001.

[9] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath, "Load balancing and unbalancing for power and performance in cluster-based systems," tech. rep., Rutgers University, 5 2001.

[10] H. David, C. Fallin, E. Gorbatov, U. R. Hanebutte, and O. Mutlu, "Memory power management via dynamic voltage/frequency scaling," in *Proceedings of the 8th ACM international conference on Autonomic computing*, pp. 31–40, ACM, 2011.

[11] S. H. Low and D. E. Lapsley, "Optimization flow control-I: basic algorithm and convergence," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 861–874, 1999.

[12] D. L. Mills, "Internet time synchronization: the network time protocol," *Communications, IEEE Transactions on*, vol. 39, no. 10, pp. 1482–1493, 1991.

[13] N. Li and J. R. Marden, "Decoupling coupled constraints through utility design," *Automatic Control, IEEE Transactions on*, vol. 59, no. 8, pp. 2289–2294, 2014.

[14] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, H. D. Simon, V. Venkatakrishnan, and S. K. Weeratunga, "The NAS parallel benchmarks," 1991.

[15] S. Eranian, "Perfmon2: a flexible performance monitoring interface for linux," in *Proc. of the 2006 Ottawa Linux Symposium*, pp. 269–288, Citeseer, 2006.

[16] R. Cochran, C. Hankendi, A. Coskun, and S. Reda, "Pack & Cap: Adaptive DVFS and Thread Packing Under Power Caps," in *ACM/IEEE International Symposium on Microarchitecture*, pp. 175–185, 2011.

[17] X. Zhang, S. Dwarkadas, G. Folkmanis, and K. Shen, "Processor hardware counter statistics as a first-class system resource," in *Proceedings of the 11th USENIX Workshop on Hot Topics in Operating Systems*, pp. 14:1–14:6, 2007.

[18] M. Grant, S. Boyd, and Y. Ye, "CVX: Matlab software for disciplined convex programming," 2008.

[19] B. Bollobás, *Random graphs*. Springer, 1998.