Temperature Aware Workload Management in Geo-distributed Data Centers

Hong Xu, *Member, IEEE,* Chen Feng, *Member, IEEE,* and Baochun Li, *Senior Member, IEEE*

Abstract—Lately, for geo-distributed data centers, a workload management approach that routes user requests to locations with cheaper and cleaner electricity has been developed to reduce energy consumption and cost. We consider two key aspects that have not been explored in this approach. First, through empirical studies, we find that the energy efficiency of cooling systems depends critically on the ambient temperature, which exhibits significant geographical diversity. Temperature diversity can be used to reduce the cooling energy overhead. Second, energy consumption comes from not only interactive workloads driven by user requests, but also delay tolerant batch workloads that run at the back-end. The elastic nature of batch workloads can be exploited to further reduce the energy cost.

In this paper, we propose to make workload management *temperature aware*. We formulate the problem as a joint optimization of request routing for interactive workloads and capacity allocation for batch workloads. We develop a distributed algorithm based on an m-block *alternating direction method of multipliers* (ADMM) algorithm that extends the classical 2-block algorithm. We prove the convergence and rate of convergence results under general assumptions. Through trace-driven simulations, we find that our approach consistently provides 15%–20% cooling energy reduction, and 5%–20% overall cost reduction over existing methods.

Index Terms—Data centers, energy, workload management, cooling efficiency, distributed optimization, ADMM

1 Introduction

Geo-distributed data centers are the powerhouses behind many Internet-scale services. They are deployed across the globe to provide better latency and redundancy. These data centers run hundreds of thousands of servers, consume megawatts of power with massive carbon footprint, and incur millions of dollars of electricity cost [14], [27]. Thus, the topic of reducing their energy consumption and cost has received significant attention [6], [10], [11], [13], [14], [21], [22], [27], [28], [31].

Energy consumption of individual data centers can be reduced with more efficient hardware and integrated thermal management [6], [10], [13], [31]. Recently, important progress has been made on a new *workload management* approach that instead focuses on the overall energy cost of geo-distributed data centers. It exploits the geographical diversity of electricity prices by optimizing the *request routing* algorithm to route user requests to locations with cheaper and cleaner electricity [14], [21], [22], [27], [28].

In this paper, we consider two key aspects of geo-distributed data centers that have not been explored in the existing literature.

First, cooling systems, consuming 30% to 50% of the total energy [26], [31], are often modeled with a constant and location-independent energy efficiency factor in existing work. This tends to be an over-simplification in reality. Through a study of a state-of-the-art production cooling system (Sec. 2), we find that temperature has direct and profound impact on cooling energy efficiency. This is especially true with *outside air cooling* technology, which has seen increasing adoption in mission-critical data centers [1]–[3]. As we will show, its partial PUE (power usage effectiveness), defined as the sum of server power and cooling overhead divided by server power, varies from 1.30 to 1.05 when temperature drops from 35 °C to -3.9 °C .

Through an extensive empirical analysis of daily and hourly climate data for 13 Google data centers, we also find that temperature varies significantly across both time and location, which is intuitive to understand. These observations suggest that data centers at different locations have distinct and time-varying cooling energy efficiencies. This establishes a strong case for making workload management *temperature aware*, where such temperature diversity can be used along with price diversity in making request routing decisions to reduce the overall cooling energy.

Second, energy consumption comes not only from interactive workloads driven by user requests, but also from delay-tolerant batch workloads, such as indexing and data mining jobs, that run at the back-end. Such a mixed nature of data center workloads, verified by measurement studies [29], provides more opportunities to utilize the energy cost diversity. The key observation is that batch workloads are elastic to resource allocations, whereas interactive workloads are highly sensitive to latency and have more pro-

• *Some preliminary results were presented at USENIX ICAC 2013, San Jose, California, June 2013.* *Hong Xu is with Department of Computer Science, City University of Hong Kong, Hong Kong, China. Email: henry.xu@cityu.edu.hk.* *His work is supported in part by City University of Hong Kong Project No. 7200366.* *Chen Feng and Baochun Li are with Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ontario M5S 3G4, Canada. Email: {cfeng,bli}@eecg.toronto.edu.*

found impact on revenue [20]. Thus at times when one location is comparatively cost efficient, we can increase the capacity for interactive workloads by reducing the resources reserved for batch jobs. More requests can then be routed to and processed at this location, and the cost saving can be more substantial. We thus advocate a holistic workload management approach, where *capacity allocation* between interactive and batch workloads is dynamically optimized with request routing. Such dynamic capacity allocation is also technically feasible because batch jobs run on large-scale distributed systems.

Towards temperature aware workload management, we first propose a general framework to capture the important trade-offs involved (Sec. 3). We model both energy cost and utility loss, which corresponds to performance-related revenue reduction. We develop an empirical cooling efficiency model based on a production system. The problem is formulated as a joint optimization of request routing and capacity allocation. The technical challenge is to develop a distributed algorithm for the large-scale optimization with tens of millions of variables for a production geo-distributed cloud. Dual decomposition with subgradient methods are often used to develop distributed optimization algorithms. They, however, require delicate adjustments of step sizes, which makes convergence difficult to achieve for large-scale problems.

We draw upon the *alternating direction method of multipliers* (ADMM), a simple yet powerful algorithm that recently has found practical use in many large-scale distributed convex optimization problems [9]. It works for problems whose objective and variables can be divided into *two* disjoint parts, by iteratively optimizing one part of the objective with one block of variables only. Our formulation, as we shall explain in Sec. 3, has three blocks of variables instead of two, and falls into the more general and difficult case of m-block ($m \geq 3$) ADMM. Little is known about m-block ADMM until recently with two exceptions [15], [18]. [15] establishes the convergence of m-block ADMM for strongly convex objective functions; [18] further shows the linear convergence of m-block ADMM under the assumption that the relation matrix is full column rank, which is, however, not the case in our formation. This motivates us to refine the framework in [18] so that it can be applied to our setup.

In particular, in Sec. 4 we construct a novel proof for the m-block ADMM algorithm in [18] to show that by replacing the full-rank assumption with some mild assumptions on the objective functions, we can obtain the same convergence and rate of convergence results as in [18]. Our contribution is important in that it extends the applicability of the m-block ADMM to the more general case where the full-rank assumption does not hold. For our problem, we further develop a distributed algorithm in Sec. 5, which is amenable to a parallel implementation in data centers.

We finally conduct extensive trace-driven simulations with an empirical cooling efficiency model, electricity prices, and temperature data to realistically assess the potential of our approach (Sec. 6). We find that temperature aware workload management is consistently able to deliver a 15%–20% cooling energy reduction and a 5%–20% overall cost reduction for geo-distributed data centers compared to existing methods. The distributed ADMM algorithm converges quickly within 70 iterations, while a dual decomposition approach with subgradient methods fails to converge within 200 iterations.

2 Background and Motivation

Before making a case for temperature aware workload management, we introduce some background of data center cooling, and empirically assess the geographical diversity of temperature.

2.1 Data center Cooling

Data center cooling is provided by the computer room air conditioners (CRACs) placed on the raised floor of the facility. The CRACs cool down the hot air exhausted from server racks by forcing it to travel through a cooling coil. Heat is often extracted by chilled water in the cooling coil, and the returned hot water is cooled through mechanical refrigeration cycles in an outside chiller plant continuously. The compressor of a chiller consumes a massive amount of energy, and accounts for the majority of the overall cooling cost [31]. The result is an energy-gobbling cooling system that typically consumes a significant portion ($\sim$30%) of the total data center power [31].

2.2 Outside Air Cooling

To improve energy efficiency, various so-called free cooling technologies that operate without mechanical chillers have recently been adopted. In this paper, we focus on a more economically viable technology called *outside air cooling* that uses an air-side economizer to direct cold outside air into the data center to cool down servers. The hot exhaust air is simply rejected out. The advantage of outside air cooling can be significant: Intel ran a 10-month experiment using 900 blade servers, and reported that 67% of the cooling energy can be saved with only slightly increased hardware failure rates [19]. Companies like Google [1], Facebook [2], and HP [3] have been operating their data centers with up to 100% outside air cooling, which reduces the average PUE to below 1.2 and saves millions of dollars on an annual basis.

The energy efficiency of outside air cooling heavily depends on ambient temperature among other factors. When temperature is lower, less air is needed for heat exchange, and the air handler fan speed can be reduced. Thus, a CRAC with an air-side economizer usually operates in three modes: mechanical,

hybrid, and outside air. Table 1 shows the empirical COP[1] and partial PUE (pPUE)[2] of a state-of-the-art CRAC with an air-side economizer. Clearly, as ambient temperature drops, the CRAC relies more on outside air cooling, and switches the operating mode from mechanical to hybrid and then outside air. As a result the COP improves six-fold from 3.3 to 19.5, and the pPUE decreases from 1.30 to 1.05. Due to the sheer amount of energy a data center draws, the numbers imply huge monetary savings for the energy bill.

Outdoor ambient	Cooling mode	COP	pPUE
35°C(90°F)	Mechanical	3.3	1.30
21.1°C(70°F)	Mechanical	4.7	1.21
15.6°C(60°F)	Hybrid	5.9	1.17
10°C(50°F)	Outside air	10.4	1.1
-3.9°C(25°F)	Outside air	19.5	1.05

TABLE 1: Efficiency of Emerson's DSE™ cooling system with an EconoPhase air-side economizer [12]. Return air is set at 29.4°C(85°F).

With the increasing use of outside air cooling, this finding motivates our proposal to make workload management temperature aware. Our idea also applies to data centers using mechanical cooling, because as shown in Table 1, the chiller energy efficiency also depends on outside temperature, albeit milder.

2.3 An Empirical Climate Study

Our idea hinges upon the geographical diversity of temperature. We substantiate this claim with an empirical analysis of historical climate data.

We use Google's data center locations for our study [4]. Google has 6 data centers in the U.S., 1 in South America, 3 in Europe, and 3 in Asia. We acquire historical temperature data from various data repositories of the National Climate Data Center [5] for all 13 locations, covering the entire one-year period of 2011.

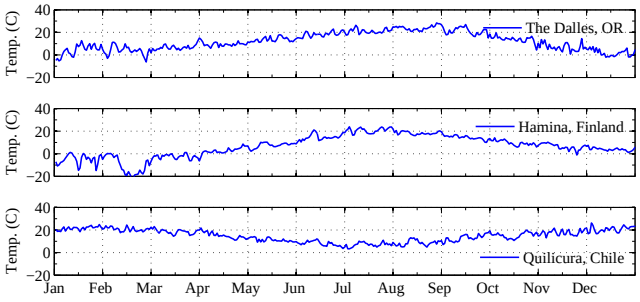


Fig. 1: Daily average temperature at three Google data center locations [5]. Time is in UTC.

Figure 1 plots the daily average temperatures for three select locations. Geographical diversity exists despite the clear seasonal pattern shared among all locations. Diversity is more salient for locations in different hemispheres (e.g. Chile). We also observe a significant amount of day-to-day volatility, suggesting that the availability and capability of outside air cooling constantly varies across regions, and there is no single location that is always cooling efficient.

We then examine short-term temperature volatility. As shown in Figure 2, the hourly temperature variations are more dramatic and highly correlated with time-of-day, which is intuitive to understand. Further, the highs and lows do not occur at the same time for different regions due to time differences.

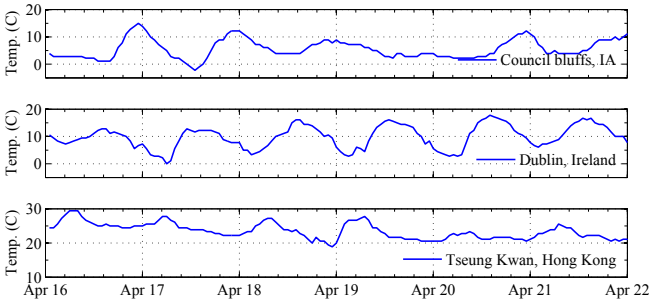


Fig. 2: Hourly temperature variations at three Google data center locations [5]. Time is in UTC.

The analysis reveals that for globally deployed data centers, local temperature at individual locations exhibits both time and geographical diversity. Thus, a carefully designed workload management scheme is both important and necessary to dynamically adjust data center operations to the ambient conditions, and to save energy costs. We also study the correlations of temperature across data centers as shown in Appendix A, and find it rather mild. Correlation structures do not impact problem formulation and algorithm design, as long as the temperature differences result in tangible pPUE differences and warrant a temperature-aware approach.

3 Model

We introduce our model and formulation of the temperature aware workload management problem here.

3.1 System Model

We consider a discrete time model where the length of a time slot matches the time scale at which request routing and capacity allocation decisions are made, e.g., hourly. We consider a provider that runs a set of data centers $\mathcal{J}$ in different regions. Each data center $j \in \mathcal{J}$ has a fixed capacity C_j in terms of the number of servers. To model data center operating costs, we consider *energy cost* and *utility loss* as detailed below.

3.2 Energy Cost and Cooling Efficiency

We focus on servers and the cooling system in our energy model. For servers, we adopt the empirical model from [13] that calculates the individual server power consumption as an affine function of CPU utilization,

1. COP, coefficient of performance, is defined for a cooling device as the ratio between cooling capacity and power.

2. pPUE is defined as the sum of cooling capacity and cooling power divided by cooling capacity. Nearly all the power delivered to servers translates to heat, which matches the CRAC cooling capacity.

$P_{\text{idle}} + (P_{\text{peak}} - P_{\text{idle}})\, u$. P_{idle} is the server power when idle, P_{peak} is the server power when fully utilized, and u is the CPU load. This model is especially accurate for calculating the aggregated power of a large number of servers [13]. It was reported in [13] with measurements from thousands of production servers that, the single activity signal of CPU utilization produces very accurate power estimation results. The reason is that CPU and memory are the main contributors to the dynamic power, and other components either have small dynamic ranges or their power usage correlates well with CPU. Thus it is unnecessary to use more complex models. Many related studies also adopt the same power model [14], [21], [22], [27], [28].

Thus, assuming that workloads are perfectly dispatched and servers have a uniform utilization as in the literature [14], [21], [22], [27], [28], the server power of data center j can be modeled as $C_j P_{\text{idle}} + (P_{\text{peak}} - P_{\text{idle}})\, W_j$, where W_j denotes the total workload in terms of the number of servers required.

For the cooling system, we take an empirical approach based on energy efficiency data of production CRACs. As noted in Sec. 2.2, a production CRAC automatically switches modes according to the ambient condition. Therefore, we study CRACs as a black box, with outside temperature as the input, and its overall energy efficiency as the output.

Specifically, we use partial PUE (pPUE)[3]. As in Sec. 2.2, pPUE is defined as

$$\text{pPUE} = \frac{\text{Server power} + \text{Cooling power}}{\text{Server power}}.$$

A smaller value indicates a more efficient system with less overhead. We apply regression techniques to the empirical pPUE data of the Emerson CRAC [12] introduced in Table 1. We find that the best fitting model—which minimizes the square error among all polynomial models—describes pPUE as a quadratic function of the outside temperature:

$$\text{pPUE} = 7.1705 \times 10^{-5} T^2 + 0.0041 T + 1.0743$$

A plot of the function with the original data points can be found in Appendix B.

Given the outside temperature T_j, the total data center energy as a function of the workload W_j can be expressed as

$$E_j(W_j) = (C_j P_{\text{idle}} + (P_{\text{peak}} - P_{\text{idle}})\, W_j) \cdot \text{pPUE}(T_j) \cdot t, \tag{1}$$

where t is the duration of one time slot. Here we implicitly assume that T_j is known a priori and do not include it as the function variable. This is valid since short-term weather forecast is fairly accurate and accessible.

A data center's electricity price is denoted as P_j. In reality, electricity can be purchased from local day-ahead or hour-ahead forward markets at a predetermined price [27]. Thus, we assume that P_j is known a priori and remains fixed for the duration of a time slot. The total energy cost, including server and cooling power, is simply $P_j E_j(W_j)$.

3. The conventional PUE metric reflects the energy efficiency of the entire facility.

3.3 Utility Loss

Request routing. The concept of utility loss captures the lost revenue due to the user-perceived latency for request routing decisions. Latency is arguably the most important performance metric for most cloud services. We focus on the end-to-end propagation latency, which largely accounts for the user-perceived latency [24]. The provider obtains the propagation latency L_{ij} between user i and data center j through active measurements [23] or other means.

We use α_{ij} to denote the volume of requests routed to data center j from user $i \in \mathcal{I}$, and D_i to denote the demand of each user that can be predicted [25]. Here, a user is an aggregated group of customers from a common geographical region, which may be identified by a unique IP prefix. The lost revenue from user i then depends on the *average* propagation latency $\sum_j \alpha_{ij} L_{ij} / D_i$ through a generic delay utility loss function U_i. U_i can take various forms depending on the cloud service. Our algorithm and proof work for general utility loss functions as long as U_i is increasing, differentiable, and convex.

We comment that differentiability and convexity are standard assumptions made in the literature [14], [21], [22], [27], [28] to make the problems tractable. While the empirical utility loss functions may not necessarily satisfy these assumptions, convexification procedures may be applied to approximate them. We also note that user utility loss may depend on other factors, such as a competitor's latency performance. We choose not to consider them to avoid additional modeling assumptions and complexity. Appendix C has more discussion on the utility loss model.

As a case study, here we use a quadratic function to model user's increased tendency to leave the service with increased latency.

$$U_i(\alpha_i) = q D_i \left(\sum_{j \in \mathcal{J}} \alpha_{ij} L_{ij} / D_i \right)^2, \tag{2}$$

where q is the delay price that translates latency to monetary terms, and $\alpha_i = (\alpha_{i1}, \ldots, \alpha_{i|\mathcal{J}|})^T$. Utility loss is clearly zero when latency is zero between the user and the data center.

Capacity allocation. We denote the utility loss of allocating β_j servers for batch workloads as a differentiable, decreasing, and convex function $V_j(\beta_j)$, since allocating more resources increases the performance of batch jobs. Unlike interactive services, batch jobs are delay tolerant and resource elastic. To model the utility loss of resource allocation, since the loss is zero when

the capacity is fully allocated to batch jobs, an intuitive definition can be of the following form:

$$V_j(\beta_j) = r(\log C_j - \log \beta_j), \tag{3}$$

where r is the utility price that converts the loss to monetary terms. Observe that it captures the intuition that increasing resources results in a decreasing marginal reduction of utility loss.

3.4 Problem Formulation

We now formulate the temperature aware workload management problem. For a given request routing scheme α, the total cost associated with interactive workloads can be written as

$$\sum_{j\in\mathcal{J}} E_j\left(\sum_{i\in\mathcal{I}} \alpha_{ij}\right) P_j + \sum_{i\in\mathcal{I}} U_i\left(\alpha_i\right). \tag{4}$$

For a given capacity allocation decision β, the total cost associated with batch workloads is:

$$\sum_{j\in\mathcal{J}} E_j(\beta_j) P_j + \sum_{j\in\mathcal{J}} V_j(\beta_j). \tag{5}$$

The optimization can be formulated as:

$$\begin{aligned}
\text{minimize} \quad & (4) + (5) && (6)\\
\text{subject to:} \quad & \forall i : \sum_{j\in\mathcal{J}} \alpha_{ij} = D_i, && (7)\\
& \forall j : \sum_{i\in\mathcal{I}} \alpha_{ij} \le C_j - \beta_j, && (8)\\
& \alpha, \beta \succeq 0, && (9)\\
\text{variables:} \quad & \alpha \in \mathbb{R}^{|\mathcal{I}|\times|\mathcal{J}|}, \beta \in \mathbb{R}^{|\mathcal{J}|}.
\end{aligned}$$

(6) is the objective function that jointly considers the cost of request routing and capacity allocation. (7) is the workload conservation constraint to ensure the user demand is satisfied. (8) is the data center capacity constraint, and (9) is the nonnegativity constraint.

We assume that requests can be directed to any data center in the formulation. In practice, typically a request can only be routed to a few locations where the associated data is available, i.e. data locality constraints. This can be readily accommodated as additional constraints in the formulation. From an optimization point of view, the only difference is that the convex constraint set is different. For more details see Appendix D.

3.5 Transforming to the ADMM Form

Problem (6) is a large-scale convex optimization problem. The number of users, i.e., unique IP prefixes, is $O(10^5)$–$O(10^6)$ for production systems. Hence, our problem can have tens of millions of variables and millions of constraints. In such a setting, a distributed algorithm is preferable to fully utilize the computing resources of data centers. Traditionally, dual decomposition with subgradient methods [8] are often used for this purpose. However, they suffer from the curse of step sizes. For the final output to be close to the optimum, we need to strategically pick the step size at each iteration, leading to the well-known problem of slow convergence with large problem sizes.

Alternating direction method of multipliers is a simple yet powerful algorithm that is able to overcome the drawbacks of dual decomposition methods, and is well suited to large-scale distributed convex optimization. Though developed in the 1970s [7], ADMM has recently received renewed interest, and found practical use in many large-scale distributed convex optimization problems in statistics, machine learning, etc. [9]. Before illustrating our new convergence proof and distributed algorithm that extend the classical framework, we introduce the basics of ADMM, followed by a transformation of (6) to the ADMM form.

ADMM solves problems in the form

$$\begin{aligned}
\min \quad & f_1(x_1) + f_2(x_2) && (10)\\
\text{s.t.} \quad & A_1x_1 + A_2x_2 = b,\\
& x_1 \in \mathcal{C}_1, x_2 \in \mathcal{C}_2,
\end{aligned}$$

with variables $x_\ell \in \mathbb{R}^{n_\ell}$, where $A_\ell \in \mathbb{R}^{p\times n_\ell}$, $b \in \mathbb{R}^p$, f_ℓ's are convex functions, and $\mathcal{C}_\ell$'s are non-empty polyhedral sets. Thus, the objective function is *separable* over *two* sets of variables, which are coupled through an equality constraint.

We can form the augmented Lagrangian [17] by introducing an extra L-2 norm term $\|A_1x_1+A_2x_2-b\|_2^2$ to the objective:

$$\begin{aligned}
L_\rho(x_1, x_2; y) = f_1(x_1) + f_2(x_2) + y^T(A_1x_1 + A_2x_2 - b)\\
+ (\rho/2)\|A_1x_1 + A_2x_2 - b\|_2^2.
\end{aligned}$$

Here, $\rho > 0$ is the penalty parameter (L_0 is the standard Lagrangian for the problem). Clearly the problem with the penalty term is equivalent to the original problem (10), since for any feasible x_ℓ the penalty term added to the objective is zero. The benefits of introducing the penalty term are improved numerical stability and faster convergence in practice [9].

ADMM solves the dual problem with the iterations:

$$\begin{aligned}
x_1^{t+1} &:= \operatorname*{argmin}_{x_1\in\mathcal{C}_1} L_\rho(x_1, x_2^t; y^t),\\
x_2^{t+1} &:= \operatorname*{argmin}_{x_2\in\mathcal{C}_2} L_\rho(x_1^{t+1}, x_2; y^t),\\
y^{t+1} &:= y^t + \rho(A_1x_1^{t+1} + A_2x_2^{t+1} - b).
\end{aligned}$$

Note the step size for the dual update is simply the penalty parameter ρ. Thus, x_1 and x_2 are updated in an alternating or sequential fashion, which accounts for the term *alternating direction*. Separating the minimization over variables is precisely what allows for decomposition when f_ℓ is separable, which will prove to be useful in our algorithm design.

The optimality and convergence of 2-block ADMM can be guaranteed under mild technical assumptions [7]. In practice, it is often the case that ADMM converges to modest accuracy within a few tens of iterations [9], which makes it attractive in practical use.

Our formulation (6) has a separable objective function. However, the request routing decision α and capacity allocation decision β are coupled by an inequality constraint rather than an equality constraint as in ADMM problems. Thus we introduce a slack variable $\gamma \in \mathbb{R}^{|\mathcal{J}|}$, and transform (6) to the following

$$\begin{aligned} \text{minimize} \quad & (4) + (5) + I_{\mathbb{R}_+^{|\mathcal{J}|}}(\gamma) && (11) \\ \text{subject to:} \quad & (7),\ (9), \\ & \forall j : \sum_i \alpha_{ij} + \beta_j + \gamma_j = C_j, && (12) \\ \text{variables:} \quad & \alpha \in \mathbb{R}^{|\mathcal{I}|\times|\mathcal{J}|}, \beta \in \mathbb{R}^{|\mathcal{J}|}, \gamma \in \mathbb{R}^{|\mathcal{J}|}. \end{aligned}$$

Here, $I_{\mathbb{R}_+^{|\mathcal{J}|}}(\gamma)$ is an indicator function defined as

$$I_{\mathbb{R}_+^{|\mathcal{J}|}}(\gamma) = \begin{cases} 0, & \gamma \succeq 0, \\ +\infty, & \text{otherwise.} \end{cases} \quad (13)$$

The new formulation (11) is equivalent to (6), since for any feasible α and β, $\gamma \succeq 0$ holds, and the indicator function in the objective values to zero. Clearly, it is in the ADMM form, with a key difference that it has three sets of variables in the objective function and equality constraint (12). The convergence of the generalized m-block ADMM, where $m \geq 3$, has long remained an open problem. Though it seems natural to directly extend the classical 2-block algorithm to the m-block case, such an algorithm may not converge unless some additional back-substitution step is taken [16]. Recently, some progresses have been made by [15], [18] that prove the convergence of m-block ADMM for strongly convex objective functions and the linear convergence of m-block ADMM under a full-column-rank relation matrix. However, the relation matrix in our setup is not full column rank. Thus, we need a new proof for the linear convergence under a general relation matrix, together with a distributed algorithm inspired by the proof.

4 THEORY

This section first introduces a generalized m-block ADMM algorithm inspired by [15], [18]. Then a new convergence proof is presented, which replaces the full column rank assumption with some mild assumptions on the objective function, and further simplifies the proof in [18]. The notations and discussions in this section are made intentionally independent of the other parts of the paper in order to present the proof in a mathematically general way.

4.1 Algorithm

We consider a convex optimization problem

$$\begin{aligned} \min \quad & \sum_{i=1}^m f_i(x_i) && (14) \\ \text{s.t.} \quad & \sum_{i=1}^m A_i x_i = b \end{aligned}$$

with variables $x_i \in \mathbb{R}^{n_i}$ $(i = 1, \ldots, m)$, where $f_i : \mathbb{R}^{n_i} \to \mathbb{R}$ $(i = 1, \ldots, m)$ are closed proper convex functions; $A_i \in \mathbb{R}^{l\times n_i}$ $(i = 1, \ldots, m)$ are given matrices; and $b \in \mathbb{R}^l$ is a given vector.

We form the augmented Lagrangian

$$L_\rho(x_1, \ldots, x_m; y) = \sum_{i=1}^m f_i(x_i) + y^T(\sum_{i=1}^m A_i x_i - b) + (\rho/2)\|\sum_{i=1}^m A_i x_i - b\|_2^2. \quad (15)$$

As in [18], a generalized ADMM algorithm has the following:

$$\begin{aligned} x_i^{k+1} &= \operatorname*{argmin}_{x_i} L_\rho(x_1^{k+1}, \ldots, x_{i-1}^{k+1}, x_i, x_{i+1}^k, \ldots, x_m^k; y^k), \\ &\qquad\qquad i = 1, \ldots, m, \\ y^{k+1} &= y^k + \varrho(\sum_{i=1}^m A_i x_i^{k+1} - b), \end{aligned}$$

where $\varrho > 0$ is the step size for the dual update. Note that the step size ϱ is different from the penalty parameter ρ in the generalized m-block ADMM algorithm, for otherwise it may not converge [16].

4.2 Assumptions

We present two assumptions on the objective functions, based on which we are able to show the convergence of the generalized m-block ADMM algorithm.

Assumption 1: The objective functions f_i $(i = 1, \ldots, m)$ are strongly convex.

Note that strong convexity is quite reasonable in engineering practice. This is because a convex function $f(x)$ can be always well-approximated by a strongly convex function $\bar{f}(x)$. For instance, if we choose $\bar{f}(x) = f(x)+\epsilon\|x\|_2^2$ for some sufficiently small $\epsilon > 0$, then $\bar{f}(x)$ is strongly convex.

Assumption 2: The gradients ∇f_i $(i = 1, \ldots, m)$ are Lipschitz continuous.

Assumption 2 says that, for each i, there exists some constant $\kappa_i > 0$ such that for all $x_1, x_2 \in \mathbb{R}^{n_i}$,

$$\|\nabla f_i(x_1) - \nabla f_i(x_2)\|_2 \leq \kappa_i\|x_1 - x_2\|_2,$$

which is again reasonable in practice, since κ_i can be made sufficiently large.

4.3 Convergence

In this section, we prove the convergence of the generalized ADMM algorithm. We outline the main idea of the analysis here. Some technical details are deferred in the Appendix for better readability.

Define the primal and dual optimality gaps as

$$\begin{aligned} \Delta_p^k &= L_\rho(x^{k+1}; y^k) - d(y^k), \\ \Delta_d^k &= d^* - d(y^k), \end{aligned}$$

respectively. By the definition of $d(y)$, $\Delta_p^k \geq 0$. Similarly, $\Delta_d^k \geq 0$. Define

$$V^k = \Delta_p^k + \Delta_d^k.$$

We will see that V^k is a *Lyapunov function* for the algorithm, *i.e.*, a nonnegative quantity that decreases in each iteration.

Our proof relies on three technical lemmas.

Lemma 1: There exists a constant $\vartheta > 0$ such that

$$V^k \leq V^{k-1} - \varrho\|A\bar{x}^{k+1} - b\|_2^2 - \vartheta\|x^{k+1} - x^k\|_2^2, \quad (16)$$

in each iteration, where $\bar{x}^{k+1} = \operatorname{argmin}_x L_\rho(x; y^k)$.

Proof: See Appendix G. □

Lemma 2: For any given $\delta > 0$, there exists a constant $\tau > 0$ (depending on δ) such that for any (x, y) satisfying $\|x\| + \|y\| \leq 2\delta$, the following inequality holds

$$\|x - \bar{x}(y)\| \leq \tau\|\nabla_x L_\rho(x; y)\|, \quad (17)$$

where $\bar{x}(y) = \arg\min_x L_\rho(x; y)$.

Proof: See Appendix F. □

Lemma 3: There exists a constant $\eta > 0$ such that

$$\|\nabla_x L_\rho(x^k; y^k)\|_2 \leq \eta\|x^k - x^{k+1}\|_2. \quad (18)$$

Proof: See Appendix E. □

By Lemma 1, we have

$$\sum_{k=0}^{\infty} \left(\varrho\|A\bar{x}^{k+1} - b\|_2^2 + \vartheta\|x^{k+1} - x^k\|_2^2\right) \leq V^0.$$

Hence, $\|A\bar{x}^{k+1} - b\|_2^2 \to 0$ and $\|x^{k+1} - x^k\|_2^2 \to 0$, as $k \to \infty$. Suppose that the level set of $\Delta_p + \Delta_d$ is bounded. Then by the Bolzano-Weierstrass theorem, the sequence $\{x^k, y^k\}$ has a convergent subsequence, *i.e.*,

$$\lim_{k \in \mathcal{R}, k \to \infty} (x^k, y^k) = (\tilde{x}, \tilde{y}),$$

for some subsequence $\mathcal{R}$, where $(\tilde{x}, \tilde{y})$ denotes the limit point. By using Lemma 2 and Lemma 3, we can show that the limit point $(\tilde{x}, \tilde{y})$ is an optimal primal-dual solution. Hence,

$$\lim_{k \in \mathcal{R}, k \to \infty} V^k = \lim_{k \in \mathcal{R}, k \to \infty} \Delta_p^k + \Delta_d^k = 0.$$

Since V^k decreases in each iteration, the convergence of a subsequence of V^k implies the convergence of V^k, and we have

$$\lim_{k \to \infty} \Delta_p^k + \Delta_d^k = 0.$$

This further implies that Δ_p^k and Δ_d^k converge to 0.

To sum up, we have the following convergence theorem for the generalized ADMM algorithm.

Theorem 1: Suppose Assumptions 1,and 2 hold and that the level set of $\Delta_p + \Delta_d$ is bounded. Then both the primal gap Δ_p^k and the dual gap Δ_d^k converge to 0.

Remark 1: By some additional argument, we can show that the sequence $\{\Delta_p^k + \Delta_d^k\}$ converges to zero Q-linearly, and that both Δ_p^k and Δ_d^k converge to zero R-linearly.

The key step is to show that $\{\Delta_p^k + \Delta_d^k\}$ contracts geometrically, i.e.,

$$\Delta_p^{k+1} + \Delta_d^{k+1} \leq \mu(\Delta_p^k + \Delta_d^k)$$

for some $\mu \in (0, 1)$. Since the key step can be established by using a similar argument in the proof of Theorem 3.1 in [18], we omit the proof here due to space constraint.

Remark 2: The linear convergence of m-block ADMM is difficult, if not impossible, to show under the framework of [15]. Thus, our result extends the result in [15] from convergence to linear convergence.

5 A Distributed Algorithm

We now develop a distributed solution algorithm based on the generalized ADMM algorithm in Sec. 4.1. Directly applying the algorithm to our problem (11) will lead to a centralized algorithm. The reason is that when the augmented Lagrangian is minimized over α, the penalty term $\sum_j \left(\sum_i \alpha_{ij} + \beta_j + \gamma_j - C_j\right)^2$ couples α_{ij}'s across i, and the utility loss $\sum_i U_i(\alpha_i)$ couples α_{ij}'s across j. The joint optimization of utility loss and the quadratic penalty is particularly difficult to solve, especially when the number of users is large, since $U_i(\alpha_i)$ can take any general form. If they can be separated, then we will have a distributed algorithm where each $U_i(\alpha_i)$ is optimized in parallel, and the quadratic penalty term is optimized efficiently with existing methods.

Towards this end, we introduce a new set of auxiliary variables $a_{ij} = \alpha_{ij}$, and re-formulate the problem (11) as follows:

$$\begin{aligned}
\text{minimize} \quad & \sum_j E_j(\sum_i a_{ij})P_j + \sum_i U_i(\alpha_i) + (5) + I_{\mathbb{R}_+^{|\mathcal{J}|}}(\gamma) \\
\text{subject to:} \quad & (7),\ (9), \\
& \forall j : \sum_i a_{ij} + \beta_j + \gamma_j = C_j, \\
& \forall i, j : a_{ij} = \alpha_{ij}, \\
\text{variables:} \quad & a, \alpha \in \mathbb{R}^{|\mathcal{I}| \times |\mathcal{J}|}, \beta, \gamma \in \mathbb{R}^{|\mathcal{J}|}. \quad (19)
\end{aligned}$$

This is a 4-block ADMM problem, where a_{ij} replaces α_{ij} in the objective function and constraint (12) when the coupling happens across users i. This is the key step that enables the decomposition of the α-minimization problem. The augmented Lagrangian can then be readily obtained from (15). By omitting the irrelevant terms, we can see that at each iteration $k+1$, the α-minimization problem is

$$\begin{aligned}
\min \quad & \sum_i U_i(\alpha_i) - \sum_j \sum_i \left(\varphi_{ij}\alpha_{ij} - \frac{\rho}{2}(\alpha_{ij}^2 - 2\alpha_{ij}a_{ij}^k)\right) \\
\text{s.t.} \quad & \forall i : \sum_j \alpha_{ij} = D_i, \alpha_i \succeq 0, \quad (20)
\end{aligned}$$

where φ_{ij} is the dual variable for the equality constraint $a_{ij} = \alpha_{ij}$. This is clearly decomposable over i into $|\mathcal{I}|$ per-user sub-problems since the objective

function and constraint are separable over i. The per-user sub-problem is of a much smaller scale with only $|\mathcal{J}|$ variables and $|\mathcal{J}|+1$ constraints, and is easy to solve even though it is a non-linear problem for a general U_i.

Some may now wonder if the auxiliary variable a is hard to solve for. The a-minimization problem is

$$\begin{aligned}\min\ & \sum_j \Big(\rho\sum_i a_{ij}(\beta_j^k+\gamma_j^k-C_j+0.5a_{ij}-\alpha_{ij}^{k+1}) \\ & + E_j\Big(\sum_i a_{ij}\Big)P_j+\sum_i a_{ij}(\lambda_j^k+\varphi_{ij}^k)+\frac{\rho}{2}(\sum_i a_{ij})^2\Big) \\ \text{s.t.}\ & a\succeq 0, \end{aligned} \tag{21}$$

where λ_j is the dual variable for the capacity constraint (8). This is decomposable over j into $|\mathcal{J}|$ per-data center sub-problems. Moreover, each per-data center sub-problem is a standard quadratic program. Though it is large-scale, it can be transformed into a second-order cone program and solved efficiently (see Appendix H).

β- and γ-minimization steps are clearly decomposable over j. The entire procedure is summarized below.

Distributed 4-block ADMM.

Initialize $a,\alpha,\beta,\gamma,\lambda,\varphi$ to 0. For $k=0,1,\ldots,$ repeat

1) **α-minimization:** Each user solves the following sub-problem for α_i^{k+1}:

$$\begin{aligned}\min\ & U_i(\alpha_i)-\sum_j\Big(\varphi_{ij}\alpha_{ij}-\frac{\rho}{2}(\alpha_{ij}^2-2\alpha_{ij}a_{ij}^k)\Big) \\ \text{s.t.}\ & \sum_j\alpha_{ij}=D_i,\alpha_i\succeq 0. \end{aligned} \tag{22}$$

2) **a-minimization:** Each data center solves the following sub-problem for $a_j^{k+1}=(a_{1j}^{k+1},\ldots,a_{|\mathcal{I}|j}^{k+1})^T$:

$$\begin{aligned}\min\ & E_j\Big(\sum_i a_{ij}\Big)P_j+\sum_i a_{ij}(\lambda_j^k+\varphi_{ij}^k)+\frac{\rho}{2}(\sum_i a_{ij})^2 \\ & +\rho\Big(\sum_i a_{ij}(\beta_j^k+\gamma_j^k-C_j+0.5a_{ij}-\alpha_{ij}^{k+1})\Big) \\ \text{s.t.}\ & a_j\succeq 0. \end{aligned} \tag{23}$$

3) **β-minimization:** Each data center solves the following sub-problem for β_j^{k+1}:

$$\begin{aligned}\min\ & E_j(\beta_j)P_j+V_j(\beta_j)+\lambda_j^k\beta_j \\ & +\frac{\rho}{2}\Big(\sum_i a_{ij}^{k+1}+\beta_j+\gamma_j^k-C_j\Big)^2 \\ \text{s.t.}\ & \beta_j\geq 0. \end{aligned}$$

4) **γ-minimization:**

$$\gamma_j^{k+1}=\max\left\{0,C_j-\frac{\lambda_j}{\rho}-\sum_i a_{ij}^{k+1}-\beta_j^{k+1}\right\},\ \forall j.$$

5) **Dual update:** Each data center updates λ_j for the capacity constraint (8):

$$\lambda_j^{k+1}=\lambda_j^k+\varrho\Big(\sum_i a_{ij}^{k+1}+\beta_j^{k+1}+\gamma_j^{k+1}-C_j\Big).$$

Each user updates φ_{ij} for the equality constraint $a_{ij}=\alpha_{ij}$:

$$\varphi_{ij}^{k+1}=\varphi_{ij}^k+\varrho(a_{ij}^{k+1}-\alpha_{ij}^{k+1}),\ \forall j.$$

The distributed nature of our algorithm allows for an efficient parallel implementation in data centers. In step 1, the per-user sub-problem (22) can be solved in parallel on each server. Since (22) is a small-scale convex optimization as discussed above, the complexity is low. A multi-threaded implementation can further speed up the algorithm with multi-core hardware. The penalty parameter ρ and utility loss function U_i can be configured at each server before the algorithm runs. Step 2 and 3 involve solving $|\mathcal{J}|$ per-data center sub-problems respectively, which can also be implemented in parallel with only $|\mathcal{J}|$ servers.

Due to space limit, more discussions about the algorithm including its convergence, are in Appendix I.

6 Evaluation

We perform trace-driven simulations to realistically assess the potential of temperature aware workload management.

6.1 Setup

We rely on the Wikipedia request traces [30] to represent the interactive workloads of a cloud service. The dataset we use contains, among other things, 10% of all user requests issued to Wikipedia from the 24-hour period of January 1–2, 2008 UTC. The workloads are normalized to a number of servers, assuming that each request requires 10% of a server's CPU. The traces reflect the diurnal pattern of real-world interactive workloads. The prediction of workloads can be done accurately [25], and we do not consider prediction error here. The optimization is solved hourly.

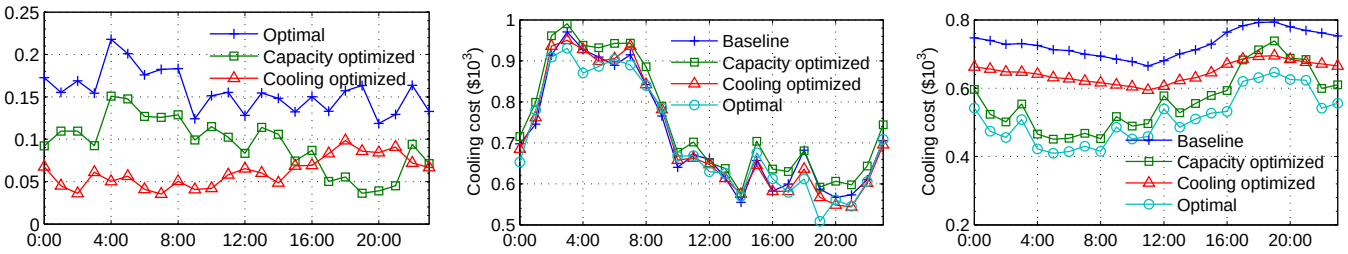
We consider Google's infrastructure [4] as in Sec. 2.3. Each data center's capacity C_j is uniformly distributed between 10K to 20K servers. The empirical CRAC efficiency model developed in Sec. 3.2 is used to derive the total energy consumption of all 13 locations under different temperatures. We use empirical power prices from the U.S. and other markets as detailed in Appendix J. The servers have peak power $P_{\text{peak}}=200$ W, and consume 50% power at idle. These numbers represent state-of-the-art data center hardware [13], [27].

To calculate the utility loss of interactive workloads, we rely on latency measurements from iPlane [23]. More details can be found in Appendix J. We set the number of users $|\mathcal{I}|=10^5$. We use utility loss functions defined in (2) and (3). The delay price $q=4\times10^{-6}$, and the utility loss price for batch jobs $r=500$.

6.2 Benchmarks

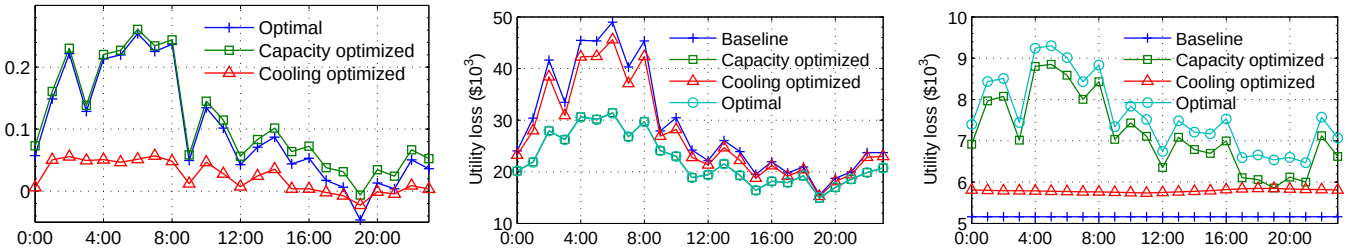
We benchmark our ADMM algorithm, referred to as *Optimal*, against three strategies, which use different amounts of information in managing workloads.

The first benchmark, called *Baseline*, is a temperature agnostic strategy that separately considers capacity allocation and request routing. It first allocates capacity to batch jobs by minimizing the back-end total cost with (5) as the objective. The remaining capacity is used to solve the request routing optimization with (4) as the objective. Only the electricity price diversity is



(a) Overall improvement. (b) Interactive workloads. (c) Batch workloads.

Fig. 3: Cooling energy savings. Time is in UTC.



(a) Overall improvement. (b) Interactive workloads. (c) Batch workloads.

Fig. 4: Utility loss reductions. Time is in UTC.

used, and cooling energy is calculated with a constant pPUE of 1.2. Though naive, such an approach is widely used in current Internet services. It also allows an implicit comparison with prior work [14], [21], [22], [27], [28].

The second benchmark, called *Capacity Optimized*, improves upon *Baseline* by jointly solving capacity allocation and request routing, but still ignores the cooling energy efficiency diversity. This demonstrates the impact of capacity allocation.

The third benchmark, called *Cooling Optimized*, improves upon *Baseline* by exploiting the temperature and cooling efficiency diversity in minimizing cost, but does not adopt joint management of the interactive and batch workloads. This demonstrates the impact of being temperature aware.

We run the four benchmarks with our 24-hour traces at each day of January 2011, using the empirical hourly temperature data we collected in Sec. 2.3. The distributed ADMM algorithm is used to solve them until convergence is achieved. The results are thus averaged over 31 runs.

6.3 Cooling energy savings

We examine the effectiveness of our approach by comparing the cooling energy consumption first. Figure 3 shows the results.

In particular, Figure 3a shows that overall, *Optimal* saves 15%–20% cooling energy compared to *Baseline*. A breakdown of the saving shown in the same figure reveals that dynamic capacity allocation provides 10%–15% saving, and cooling efficiency diversity provides 5%–10% saving, respectively. Note that the cost saving is achieved with cutting-edge CRACs whose efficiency has been substantially improved already. The results confirm that our approach further optimizes the cooling efficiency and cost of geo-distributed data centers.

Figure 3b and 3c show a detailed breakdown of cooling energy cost. Cooling cost attributed to interactive workloads, as in Figure 3b, exhibits a diurnal pattern and peaks between 2:00 and 8:00 UTC (21:00 to 3:00 EST, 18:00 to 0:00 PST), implying that most of the Wikipedia traffic originates from the U.S. The four strategies perform fairly closely, while *Baseline* and *Capacity optimized* consistently incur more cooling energy cost due to their cooling agnostic nature.

Cooling cost attributed to batch workloads is shown in Figure 3c. *Baseline* incurs the highest cost since it underestimates the energy cost, and runs more batch workloads than necessary. *Cooling optimized* improves *Baseline* by taking into account cooling efficiency diversity and reducing batch workloads as a result. Both strategies fail to exploit the trade-off with interactive workloads. Thus their cooling cost closely follows the daily temperature trend in that it gradually decreases from 0:00 to 12:00 UTC (19:00 to 7:00 EST) and then slowly increases from 12:00 to 20:00 UTC (7:00 to 15:00 EST). *Capacity optimized* adjusts capacity allocation with request routing, and further reduces batch workloads in order to allocate more resources for interactive workloads. *Optimal* combines temperature aware cooling optimization with holistic workload management, and has the lowest cooling cost with least batch workloads. Though this increases the back-end utility loss, the overall effect is a net reduction of total cost since interactive workloads enjoy lower latency as will be observed soon.

6.4 Utility loss reductions

Another component of data center cost is utility loss. From Figure 4a, the relative reduction follows the

interactive workloads and also has a visible diurnal pattern. *Optimal* and *Capacity optimized* provide the most significant utility loss reductions from 5% to 25%, while *Cooling optimized* provides a modest 5% reduction compared to *Baseline*. To study the reasons for the varying degrees of reductions, Figure 4b and 4c show the respective utility loss of interactive and batch workloads. We observe that interactive workloads incur most of the utility loss, reflecting its importance compared to batch workloads. *Baseline* and *Cooling optimized* have much higher utility loss from interactive workloads as shown in Figure 4b, because of the separate management of two workloads. The average latency performances under these two strategies are also worse as demonstrated in Appendix K.

Capacity optimized and *Optimal* outperform the two by allocating more capacity to interactive workloads at cost-efficient locations while reducing batch workloads (recall Figure 3c). This is especially effective during peak hours as shown in Figure 4b. *Capacity optimized* and *Optimal* do have higher utility loss from batch workloads as seen in Figure 4c. However since interactive workloads attribute to the majority of the provider's utility and revenue, the overall effect of joint workload management is positive.

We also evaluate performance of all approaches in different seasons. The results indicate that temperature aware workload management offers consistent cost benefits throughout the year. More details can be found in Appendix L. Evaluation of the convergence of our algorithm is in Appendix M, where we show our distributed ADMM algorithm converges much faster than subgradient methods.

7 Conclusion

We propose temperature aware workload management, which explores geographical diversity of temperatures and cooling efficiency, together with dynamical capacity allocation between batch and interactive workloads. We formulate the joint optimization under a general framework with an empirical cooling efficiency model. To solve large-scale problems for production systems, we rely on a distributed m-block ADMM algorithm. Extensive simulations highlight that temperature aware workload management saves 15%–20% cooling energy and 5%–20% overall energy cost, and distributed ADMM is practical to solve large-scale workload management problems with only tens of iterations.

References

[1] http://tinyurl.com/89ros64.
[2] http://tinyurl.com/8ulxfzp.
[3] http://tinyurl.com/bpqv6tl.
[4] https://www.google.com/about/datacenters/inside/locations/.
[5] National climate data center (NCDC). http://www.ncdc.noaa.gov.
[6] C. Bash and G. Forman. Cool job allocation: Measuring the power savings of placing jobs at cooling-efficient locations in the data center. In *Proc. USENIX ATC*, 2007.
[7] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.
[8] S. Boyd and A. Mutapcic. Subgradient methods. Lecture notes of EE364b, Stanford University, Winter Quarter 2006-2007. http://www.stanford.edu/class/ee364b/notes/subgrad_method_notes.pdf.
[9] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2010.
[10] Y. Chen, D. Gmach, C. Hyser, Z. Wang, C. Bash, C. Hoover, and S. Singhal. Integrated management of application performance, power and cooling in datacenters. In *Proc. NOMS*, 2010.
[11] N. El-Sayed, I. Stefanovici, G. Amvrosiadis, and A. A. Hwang. Temperature management in data centers: Why some (might) like it hot. In *Proc. ACM Sigmetrics*, 2012.
[12] Emerson Network Power. Liebert® DSE™ precision cooling system. http://tinyurl.com/c7e8qxz, 2012.
[13] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In *Proc. ACM/IEEE Intl. Symp. Computer Architecture (ISCA)*, 2007.
[14] P. X. Gao, A. R. Curtis, B. Wong, and S. Keshav. It's not easy being green. In *Proc. ACM SIGCOMM*, 2012.
[15] D. Han and X. Yuan. A note on the alternating direction method of multipliers. *J. Optim. Theory Appl.*, 155:227–238, 2012.
[16] B. S. He, M. Tao, and X. M. Yuan. Alternating direction method with Gaussian back substitution for separable convex programming. *SIAM J. Optim.*, 22:313–340, 2012.
[17] M. R. Hestenes. Multiplier and gradient methods. *J. Optim. Theory Appl.*, 4(5):303–320, 1969.
[18] M. Hong and Z.-Q. Luo. On the linear convergence of the alternating direction method of multipliers, August 2012.
[19] Intel Inc. Reducing data center cost with an air economizer, August 2008.
[20] R. Kohavi, R. M. Henne, and D. Sommerfield. Practical guide to controlled experiments on the web: Listen to your customers not to the hippo. In *Proc. ACM SIGKDD*, 2007.
[21] M. Lin, A. Wierman, L. L. H. Andrew, and E. Thereska. Dynamic right-sizing for power-proportional data centers. In *Proc. IEEE INFOCOM*, 2011.
[22] Z. Liu, M. Lin, A. Wierman, S. H. Low, and L. L. Andrew. Greening geographical load balancing. In *Proc. ACM Sigmetrics*, 2011.
[23] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: An information plane for distributed services. In *Proc. USENIX OSDI*, 2006.
[24] S. Narayana, J. W. Jiang, J. Rexford, and M. Chiang. To coordinate or not to coordinate? Wide-Area traffic management for data centers. Technical report, Princeton University, 2012.
[25] D. Niu, H. Xu, B. Li, and S. Zhao. Quality-assured cloud bandwidth auto-scaling for video-on-demand applications. In *Proc. IEEE INFOCOM*, 2012.
[26] S. Pelley, D. Meisner, T. F. Wenisch, and J. W. VanGilder. Understanding and abstracting total data center power. In *Proc. Workshop on Energy Efficient Design (WEED)*, 2009.
[27] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs. Cutting the electricity bill for Internet-scale systems. In *Proc. ACM SIGCOMM*, 2009.
[28] L. Rao, X. Liu, L. Xie, and W. Liu. Minimizing electricity cost: Optimization of distributed Internet data centers in a multi-electricity-market environment. In *Proc. IEEE INFOCOM*, 2010.
[29] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In *Proc. ACM SoCC*, 2012.
[30] G. Urdaneta, G. Pierre, and M. van Steen. Wikipedia workload analysis for decentralized hosting. *Elsevier Computer Networks*, 53(11):1830–1845, July 2009.
[31] R. Zhou, Z. Wang, A. McReynolds, C. Bash, T. Christian, and R. Shih. Optimization and control of cooling microgrids for data centers. In *Proc. IEEE ITherm*, 2012.

Hong Xu received the B.E. degree from the Department of Information Engineering, The Chinese University of Hong Kong, in 2007, and the M.A.Sc. and Ph.D. degrees from the Department of Electrical and Computer Engineering, University of Toronto. He joined the Department of Computer Science, City University of Hong Kong in August 2013, where he is currently an assistant professor. His research interests include data center networking, cloud computing, network economics, and wireless networking. He is a member of ACM and IEEE.

Chen Feng is currently a PhD candidate in the Department of Electrical and Computer Engineering, University of Toronto. He received his B.E. degree from Shanghai Jiao Tong University in 2006, and his M.A.Sc degree from the University of Toronto in 2009. His research interests are in network coding, coding theory, information theory, and their applications to computer networking. He is a recipient of the Chinese Government Award for Outstanding Students Abroad in 2012, the Shahid U.H. Qureshi Memorial Scholarship in 2013, and the Graduate Student Endowment Fund Scholarship in 2013.

Baochun Li received the B.E. degree from the Department of Computer Science and Technology, Tsinghua University, China, in 1995 and the M.S. and Ph.D. degrees from the Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, in 1997 and 2000.

Since 2000, he has been with the Department of Electrical and Computer Engineering at the University of Toronto, where he is currently a Professor. He holds the Nortel Networks Junior Chair in Network Architecture and Services from October 2003 to June 2005, and the Bell Canada Endowed Chair in Computer Engineering since August 2005. His research interests include large-scale distributed systems, cloud computing, peer-to-peer networks, applications of network coding, and wireless networks.

Dr. Li was the recipient of the IEEE Communications Society Leonard G. Abraham Award in the Field of Communications Systems in 2000. In 2009, he was a recipient of the Multimedia Communications Best Paper Award from the IEEE Communications Society, and a recipient of the University of Toronto McLean Award. He is a senior member of IEEE and IEEE Computer Society and a member of ACM.