Distributed dynamic speed scaling

Rade Stanojevic
Telefonica Research, Barcelona, Spain

Robert Shorten
Hamilton Institute, NUIM, Ireland

Abstract— **In recent years we have witnessed a great interest in large distributed computing platforms, also known as clouds. While these systems offer enormous computing power, they are however major energy consumers. In the existing data centers CPUs are responsible for approximately half of the energy consumed by the servers. A promising technique for saving CPU energy consumption is dynamic speed scaling, in which the speed at which the processor is ran is adjusted based on demand and performance constraints. In this paper we look at the problem of allocating the demand in the network of processors (each being capable to perform dynamic speed scaling) to minimize the global energy consumption/cost. The nonlinear dependence between the energy consumption and the performance as well as the high variability in the energy prices result in a nontrivial resource allocation. The problem can be abstracted as a fully distributed convex optimization with a linear constraint. On the theoretical side, we propose two low-overhead fully decentralized algorithms for solving the problem of interest and provide closed-form conditions that ensure stability of the algorithms. Then we evaluate the efficacy of the optimal solution using simulations driven by the real-world energy prices. Our findings indicate a possible cost reduction of $10-40\%$ compared to power-oblivious $1/N$ load balancing, for a wide range of load factors.**

I. Introduction

Nowadays, many internet services are structured in a "cloud" around a large number of cloud servers that are distributed worldwide to decrease the costs and to improve content availability, robustness to faults, end-to-end delays, and data transmission rates [15]. Examples include most of Yahoo! and Google services, Amazon's Simple Storage Service (S3) and Elastic Compute Cloud (EC2) as well as Akamai's Content Delivery Network (CDN). Applications that currently rely on these distributed platforms include content distribution, search, remote backup, social networking, etc. Utilizing third party resources (located in the cloud) seems to be a major step towards the new generation of powerful and cheap applications (particularly in enterprise environments), often referred to as software-as-a-service (SaaS) model.

The typical size (and consequently, the power consumption) of a network cloud is massive. The U.S. Environmental Protection Agency estimates the energy consumption by the nation's servers and data centers was 61 billion kilowatt-hours (kWh) in 2006 (1.5 percent of total U.S. electricity consumption), and is projected to reach 100 billion kWh by 2011. It is evident that, apart from the direct costs from the massive power consumption, the environmental cost of data centers is abnormally high. This has led a number of research groups to look at various approaches for reducing energy consumption/cost [1], [8], [14], [19], [33], [36], [37], [43].

The worldwide scale[1] of such systems raises important issues as to how to efficiently control power consumption/cost in those large distributed environments. The following reasons make the design of power-aware resource-control highly nontrivial. First, energy prices exhibit large temporal and geographic variations[2] which in turn implies that the optimal operating point is not fixed, but rather changing based on (hard-to-predict) energy-price dynamics, consequently requiring an online solution. Second, the function that relates the demand and power consumption is highly nonlinear, implying that it is hard to predict what would be the global effects of actions performed locally. Finally, the dimensionality of the problem requires solutions that offer high scalability through low communication overhead, high fault robustness and fast convergence times.

In modern data centers, processors are responsible for approximately 50% of energy consumed by high-end servers [19], [22]. Moreover, every watt saved on the servers, results on the extra 0.5-1 watt saved on cooling and power delivery costs. Consequently, in this paper we investigate how balancing the load among processors with different levels of utilization, as well as different energy prices can reduce the overall energy-related costs. Our work is strongly motivated by the concept of dynamic speed scaling [8]. This is a technique that allows load-aware adaptation of the speed at which processor is run, in order to save the energy. Namely, in order to run a processor at speed s, one needs to supply a power of cs^{α}, with $\alpha = 3$ being the value most commonly used in the literature [1], [8], [46]. Here we look at the problem of demand allocation across a network of processors, with the common goal of minimizing the total cost of power consumed. The key factors affecting the cost gains are (1) the nonlinearity of the performance/energy curve and (2) the large variance of the energy prices across different sites. Thus, the presented framework can be applied to any application that features those two factors.

A. Problem formulation

Let D be the demand intensity (say in requests per second) that needs to be processed by N processor clusters, with cluster i has P_i processors capable of performing dynamic speed scaling based on the offered load. For $i \in \{1, 2, \ldots, N\}$

[1]For example, Google's and Microsoft's services run on up to a million servers distributed worldwide [39], [12], [19] across dozens of data centers, across all continents. Akamai's content distribution network utilizes tens of thousands of servers [15].

[2]For example, in the UK&Ireland energy market, the energy price is created once every 30 minutes with the ratio of daily highest and lowest prices varying between 3 and 10 [9].

the cluster i processes a fraction, $D_i \geq 0$, of total demand, so that

$$\sum_{i=1}^{N} D_i = D. \tag{1}$$

With cluster i serving a load with intensity D_i, the power consumption/cost required for ensuring certain QoS level is a (convex) function of D_i: $f_i(D_i)$. Function f_i depends on the local cost of energy (in $\$/KWh$) and the cluster structure (eg. the number of CPUs, the type of application, etc.); see Section II for more details. Note here that, with given temporal and geographic diversity in the prices, the cost function f_i is location dependent and varies with time (at cluster i) of the day, seasonal changes, etc. Our design objective is obtaining a fully decentralized architecture for power-cost minimization. In terms of communication infrastructure, we allow clusters to exchange local information over the edges of the communication (undirected) graph $G = (\mathbf{N}, E)$, where $\mathbf{N} = \{1, 2, \ldots, N\}$. Given this our performance goal is to design an algorithm that allows nodes (processors) to collaborate without any global information to achieve the minimum aggregate cost:

$$\min_{D_i \geq 0,\ \sum_{i=1}^{N} D_i = D} V(D_1, \ldots, D_N) =$$

$$\min_{D_i \geq 0,\ \sum_{i=1}^{N} D_i = D} \sum_{i=1}^{N} f_i(D_i). \tag{2}$$

Fully distributed algorithms for solving convex problems with multiple linear constraints have been studied in [30]. For convenience, this method is described in Appendix B, and relies on the distributed algorithm for the SUM computation from [31], which is executed at each step of the underlying gradient ascent algorithm. As we shall see, this distributed SUM computation step imposes a large communication overhead that may discourage its usage in commercial settings. In the Section III we present a method that avoids computation of aggregate SUM, and therefore significantly reduces the communication overhead (and consequently reduces the convergence time).

Comment 1: While here we model functions f_i through simple polynomials, in reality, these functions might not be parameterizable through a handful of parameters. That in turn implies that centralized approach would require each node to continuously communicate fine-grained representation of time-varying function f_i to the centralized controller. This approach would not scale to large number of nodes. Therefore, we seek for decentralized solutions that scale well with large number of nodes, through the framework described above. However, in small networks, centralized architecture is feasible, and would provide a solution of the optimization problem in a straightforward manner.

Comment 2: Here we want to stress that the above model ignores the cost of bandwidth. While this is an important consideration for those applications that intensively use WAN, for many applications such as search or media hosting, the major computational cost comes in the CPU intensive operations such as computing the relevant index or rendering. See [7] for a number of case studies of various applications utilizing Amazon's EC2 cloud platform.

B. *Our contributions*

Motivated by the need for reducing power costs through distributed speed scaling, the main concern of this paper is investigation of the potential benefits that power-aware load balancing can have on the electricity costs across a network of distributed servers. In particular we seek for efficient and fully distributed solution for minimization of the aggregate costs (2). Briefly, the main contributions of our work are following:

- We propose a framework for power reduction in the cloud through distributed dynamic speed scaling that takes into account the temporal and spatial variability of the power prices.
- Two fully distributed algorithms (synchronous and asynchronous) for solving the optimization problem of interest are proposed, and sufficient conditions are derived which guarantee that both algorithms drive system to the desired state.
- Several illustrative simulations are presented to evaluate the behavior of the algorithm and support our analytical findings. In particular, we show on a synthetic example driven by the real-world energy prices, that one can expect a reduction of (processor energy related) costs in the range of $10 - 40\%$.

While the presented work has been motivated mainly by concerns related to energy reduction in the cloud, we note that the general problems of type (2) arise in many related domains. In particular load-aware WLAN spectrum sharing [13], distributed throttling of high-bandwidth aggregates for DDoS protections [45], and cost-optimal multihoming [18] share the same underlying problem (see Section VI).

We shall also see that dynamic system underlying our algorithm is nonlinear and implicit. This makes the task of analysis challenging. In particular, the standard theory of distributed coordination algorithms (see [23] and references therein) cannot be employed in our case. The convergence results established in Theorems 1 and 2 are highly nontrivial and represent the main theoretical contribution of this paper.

From the practical side, the proposed solution requires only a few lines of code, is easy to debug, and requires configuration of at most one parameter (η). In contrast, the MRS algorithm from [30] that solves the same type of problems has a set of non-trivial rules for parameter setting, involved logic behind it, making it hard to implement and debug in a real-world setting. Finally because our schemes avoid computation of the global state, the resulting communication overhead can be several orders of magnitude smaller compared to MRS (see Section V-C).

C. Related work

The topic of effective running cloud based services and data centers has lately attracted significant attention in the networking community; see [19], [4] and references therein. A major concern of these efforts include power control as it accounts for the largest part of the non-fixed expenses. Recent efforts for power reduction in data centers and networked systems include: power-aware caching and prefetching [37], sleeping and rate-adaptation [33] and load dispatching [14] of networked and storage systems. Recently, several proposals have appeared promoting the use of a large number of small (micro [21], nano [27]) data centers, instead of few large data centers, as a means to save energy. We refer interested readers to a nice overview of the electricity markets [36], where the author proposes to exploit the diversity of energy prices to save the running costs.

Most of the previous work on dynamic speed scaling has been concerned with scheduling mechanisms needed to efficiently run a single processor [1], [8], [46]. More recently Gandhi et al. [17] looked at problem of load allocation among the processors within a single cluster to maximize the performance under a power budget. Our work in Section II complements their results in that it derives the load allocation that minimizes the cost under a strict performance constraint. However, most of our paper is concerned with power aware demand allocation in the network of processors, in the distributed setting without centralized controller. As we shall see the problem lies in the realm discussed in [30]: namely, of maximizing the sum of (concave) utilities with linear constraints under fully distributed communication infrastructure; see Appendix B for a brief description of algorithm from [30]. In context of distributed flow rate-control, Kelly's framework [24], [42] solves similar problems assuming some form of feedback (packet loss, queueing delay, packet marks, etc.) that contains a global information that is utilized by the agents in order to solve the optimization problem. However, here we assume no such global feedback, and agents communicate between each other sharing only local information.

Load balancing is a common name for a suite of algorithms that allocate demands among a set of servers, to equalize the performance at each of the contributing servers. See [5], [28] for an overview of a standard model of load balancing. Load balancers, in the context of content delivery networks, are described in [38]. Recently, an overlay resource control based on two-random choice [28] load balancing has been proposed in [34]. For load balancing techniques in distributed environments we refer readers to [2], [3]. The problem analyzed here can be seen as an instance of distributed utility maximization load balancing. For a related concept called distributed rate limiting, see [39], [44].

Our method for solving the optimization problem defined in Section I-A can be seen as an instance of the distributed coordination. Distributed coordination has attracted significant attention over last several years being applied in various topics, such as flocking [41], time synchronization, multi-agent coordination [23], sensor, peer-to-peer and ad hoc networks [11]. In most existing applications distributed coordination algorithms can be modeled as positive linear systems, which then allows the elegant theory of nonnegative matrices and Markov chains to be employed to capture the convergence properties of the algorithms. However, little is known about implicit nonlinear coordination problems (see [32]) and one of the main contributions of this paper is the proof of global stability for the implicitly given nonlinear systems describing the dynamics of the algorithms presented in the Section III.

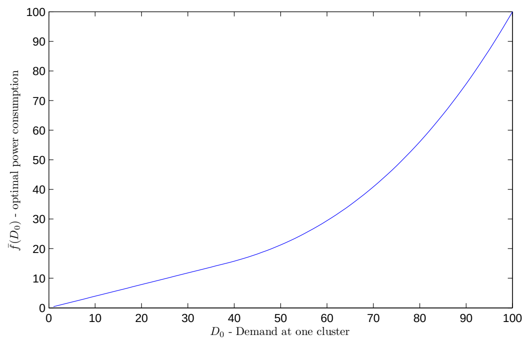

Fig. 1. Local load vs. optimal energy consumption.

II. Load balancing within one cluster

Before we proceed with the problem of distributed power-aware load distribution, we first investigate what can be done locally in terms of load balancing among the processors within one cluster under standard CPU speed vs. power model. Let P be the number of processors within a cluster, each being able to run at speed $s \in (0, 1]$. Dynamic speed scaling literature [1], [8], [46], models the processor power consumption running at speed $s > 0$ as a function of the following form:

$$p(s) = \gamma + (1 - \gamma)s^{\alpha},$$

with the power exponent $\alpha > 1$, most commonly the value $\alpha = 3$ is used. Note also that amortization parameter γ lies in the interval $(0, 1)$, as certain amount of energy is spent for running the processor even when it is idle. We assume also that when the processor is turned off the power consumption is $p(0) = 0$. Since the capacity (maximum speed) of each processor is one unit, the maximum processing speed of the whole cluster is P. For $D_0 \in [0, P]$, lets evaluate the strategy of allocating the demand among those P processors, $(s_1, \dots, s_P)$ (with $s_i \geq 0$ and $\sum_{i=1}^{P} s_i = D_0$) that minimizes the total power:

$$A(s_1, \dots, s_P) = \sum_{i=1}^{P} p(s_i).$$

The optimal strategy might require some of the processors to be turned off. Let k of those P processors be switched on, then without loss of generality we can assume that they are ordered in the following way:

$$s_1 \geq s_2 \geq \ldots \geq s_k > 0 = s_{k+1} = \ldots = s_P.$$

Then, the total power is

$$A(s_1, \ldots, s_P) = \sum_{i=1}^{k}(\gamma+(1-\gamma)s_i^\alpha) \geq k\gamma+(1-\gamma)k\left(\frac{\sum_{i=1}^{k} s_i}{k}\right)$$
$$= k\gamma + (1-\gamma)k^{1-\alpha}D_0^\alpha =: b(k). \quad (3)$$

In (3) we used the following lemma:

Lemma 1: For any nonnegative real numbers $x_1, \ldots, x_k$ the following inequality holds:

$$\frac{\sum_{i=1}^{k} x_i^\alpha}{k} \geq \left(\frac{\sum_{i=1}^{k} x_i}{k}\right)^\alpha. \quad (4)$$

Proof: Notice that $d(t) = t^\alpha$ is a convex function on $[0, \infty)$ for $\alpha \geq 1$. The lemma follows directly from the Jensen's inequality[3] for the convex function $d(t)$:

$$\frac{1}{k}\sum_{i=1}^{k} d(x_i) \geq d\left(\frac{\sum_{i=1}^{k} x_i}{k}\right).$$

■

The above function $b(k)$ has derivative $b'(k) = \gamma+(1-\gamma)(1-\alpha)k^{-\alpha}D_0^\alpha$. Therefore $b(k)$ attains minimum on in the interval $[1, P]$ for

$$k^* = \min(P, D_0\left(\frac{(1-\gamma)(\alpha-1)}{\gamma}\right)^{\frac{1}{\alpha}}). \quad (5)$$

Therefore, we just proved the following proposition:

Proposition 1: In the model described above, the load allocation that minimizes the consumed power at one cluster of P processors is the one for which which k^* (given by (5)[4]) servers are turned on, and each of them processes the same load: $\frac{D}{k^*}$. In that case, the energy consumed by the cluster is given by:

$$\bar{f}(D_0) = b(k^*) = k^*\gamma + (1-\gamma)(k^*)^{1-\alpha}D_0^\alpha. \quad (6)$$

The graph $(D_0, \bar{f}(D_0))$ is depicted in Figure 1 for $\alpha = 3$, $\gamma = 0.1$ on a cluster of $P = 100$ processors.

The previous proposition applies to the case of homogenous processors, for which all the processors have the same relationship between power and speed. In the case of non-homogenous processors, the appropriate load balancing needs to be scaled to account for difference in the weights at different processors.

III. Load balancing across distributed clusters

In the previous section we showed how to balance the load allocated to one cluster in order to minimize the energy consumption at the cluster. In this section we look at the problem formulated in Section I-A: balancing the load D on the set of distributed clusters to minimize the aggregate cost of power (2). For cluster i serving load D_i, the energy consumption is given by $\bar{f}_i(D_i)$ (Proposition 1), and the cost of unit of power is given by ν_i. The power cost is then $f_i(D_i) = \nu_i\bar{f}_i(D_i)$. The key observation of this section is that solving convex problem (2) can be reduced to solving a distributed coordination problem (in which function values among all nodes in the network need to be equalized) that in turn helps us design algorithms with low communication overhead and fast convergence times.

In order to avoid directly enforcing the nonnegativity constraints ($D_i \geq 0$ and $D_i \leq P_i$), we consider the cost functions with a logarithmic barrier

$$h_i(D_i) = f_i(D_i) - \theta\log(D_i) - \theta\log(P_i - D_i),$$

for a small parameter $\theta > 0$. Now, consider the following optimization problem, parameterized with θ:

$$\min_{\sum_{i=1}^{N} D_i = D} V_\theta(\mathbf{D}) = \min_{\sum_{i=1}^{N} D_i = D} \sum_{i=1}^{N} h_i(D_i). \quad (7)$$

For the analysis of the error introduced by this logarithmic barrier see [30]. Basically, for any $\theta > 0$, the solution of (2) is $O(N\theta)$ away from solution of (7). In the evaluation section we set $\theta = 10^{-10}$. For given Lagrange multiplier[5] λ, the Lagrange function is:

$$\Lambda(\lambda, D_1, \ldots, D_N) = V_\theta(D_1, \ldots, D_N) - \lambda(\sum_{i=1}^{N} D_i - D).$$

Therefore the point $\mathbf{D} = (D_1, \ldots, D_N)$ that minimizes V_θ must satisfy:

$$\frac{\partial\Lambda}{\partial D_i} = 0$$

for all i. The last condition is equivalent to

$$h_i'(D_i) = f_i'(D_i) - \frac{\theta}{D_i} + \frac{\theta}{P_i - D_i} = \lambda.$$

Thus, finding the solution of (7) is **equivalent** to finding the point $(D_1, \ldots, D_N)$ for which $\sum_{i=1}^{N} D_i = D$ and

$$\forall(i,j) \quad h_i'(D_i) = h_j'(D_j). \quad (8)$$

Now we describe a fully decentralized algorithm that allows every cluster to obtain the optimal load it needs to process. The observation that finding the optimal point is equivalent to finding a point that satisfies (1) and (8) greatly simplifies the design. The algorithm, that we call Implicit Distributed Coordination (IDC) is given by the pseudocode in Figure 2, and parameterized by parameter η. It is fully decentralized, in that every agent exchanges only local information with its neighbors and has the following logic behind it. Let i and j be nodes connected in the communication graph G. If $h_i'(D_i)$

[3]http://en.wikipedia.org/wiki/Jensen's_-inequality

[4]In case k^* is not an integer, we pick the closest integer instead.

[5]http://en.wikipedia.org/wiki/Lagrangee_-multipliers

```
InitializeDemands()
    for $i = 1 : N$
        $D_i \leftarrow D \frac{P_i}{\sum_{j=1}^{N} P_j}$
    endfor

UpdateDemands()
    Once every $\Delta$ units of time do
        for $i = 1 : N$
            $D_i \leftarrow D_i + \eta \sum_{(i,j)\in E}(h'_j(D_j) - h'_i(D_i))$
        endfor
    enddo
```

Fig. 2. Pseudo-code of IDC

is greater than $h'_j(D_j)$ then this indicates that allocating some demand to node j from node i would potentially decrease the difference between $h'_i(D_i)$ and $h'_j(D_j)$. The parameter $\eta > 0$ determines the responsiveness and stability properties of IDC. While the basic algorithm makes sense intuitively, many questions need to be answered before it can be deployed. Paramount among these concerns under which conditions does the IDC converge to the desired (unique) state and if so, how fast. These questions provide the focus for the investigation presented in the next section.

A. Model and analysis of IDC

In this section we analyze the dynamics of IDC. We provide a bound on η that ensures the stability (convergence) of the system, describing the dynamics of the state vector, $(D_1, \dots, D_N)$, and show that the convergence is exponentially fast (it converges with time t as e^{-at}). We model the IDC system in discrete time t. At the t-th iteration, denote by $D_i(t)$ the fraction of load, served by processor i and by

$$q_i(t) = h'_i(D_i(t)),$$

the value of h'_i at point $D_i(t)$. Being the sum of three convex functions, h_i is itself a convex function. Therefore, h'_i is an increasing function and has a well defined inverse. Let us denote by $g_i = (h'_i)^{-1}$ the inverse function of h'_i. Then

$$D_i(t) = g_i(q_i(t)) = g_i(h'_i(D_i(t))). \tag{9}$$

Equation (9) represents the key relationship between $D_i(t)$ and $q_i(t)$. Given this, the dynamical system describing the evolution of $D_i(t)$, is given by:

$$D_1(0) = D_2(0) = \ldots = D_N(0) = D/N, \tag{10}$$

$$D_i(t+1) = D_i(t) + \eta \sum_{(i,j)\in E} (q_j(t) - q_i(t)). \tag{11}$$

We also denote

$$m(t) = \min_{1\le i\le N} q_i(t),$$

and

$$M(t) = \max_{1\le i\le N} q_i(t).$$

The following lemma is a straightforward consequence of the fact that G is an undirected graph.

Lemma 2: At all times t, the constraint (1) is satisfied: $D_1(t) + D_2(t) + \ldots + D_N(t) = D$.

Proof: For $t = 0$ the statement is true from the definition. Suppose that it is valid for $t = k$, then for $t = k+1$:

$$\sum_{i=1}^{N} D_i(k+1) = \sum_{i=1}^{N} \left[D_i(k) + \eta \sum_{(i,j)\in E} (q_i(k) - q_j(k)) \right] =$$

$$= \sum_{i=1}^{N} D_i(k) + \frac{1}{2}\eta \sum_{(i,j)\in E} ((q_i(k) - q_j(k)) + (q_j(k) - q_i(k)))$$

$$= \sum_{i=1}^{N} D_i(k) = D.$$

■

The following theorem gives a sufficient condition under which the system (10)-(11) converges.

Theorem 1: Let d_i be the degree of node i in the communication graph. If all g_i are differentiable on $(0, \infty)$, then for every η that satisfies:

$$0 < \eta \le \frac{1}{2} \min_{1\le i\le N} \Big(\inf_{y\in[m(0),M(0)]} g'_i(y) \Big) \min_{1\le i\le N} \frac{1}{d_i} \tag{12}$$

the following limits exist

$$\lim_{t\to\infty} D_i(t) =: D_i^*$$

and

$$\lim_{t\to\infty} h'_i(D_i(t)) = \lim_{t\to\infty} h'_j(D_j(t)) =: q^*.$$

The convergence is exponential: there exist real numbers $\theta_1 > 0, \theta_2 > 0$ such that

$$0 \le M(t) - m(t) < \theta_1 e^{-t\theta_2}$$

Proof: See Appendix. ■

Comment 3: The presented algorithm, IDC, can be seen as an instance of distributed equation solving. Suppose that N agents want to solve the following equation in a distributed manner

$$G(x) = \sum_{i=1}^{N} g_i(x) = D.$$

If each agent i is able to solve the equation $g_i(x) = y$ for every y then IDC-like algorithm with appropriate η converges to the solution of the above equation.

B. Asynchronous algorithm

Suppose that nodes i and j, have associated values $h'_i(D_i)$ and $h'_j(D_j)$. If we want to find β for which $h'_i(D_i + \beta) \approx h'_j(D_j - \beta)$, then the first order approximation (using the Taylor series expansion) would be

```
1   InitializeDemands()
2       for $i = 1 : N$
3           $D_i \leftarrow D \frac{P_i}{\sum_{j=1}^{N} P_j}$
4       endfor

5   UpdateDemands()
6       for every edge $(i,j) \in E$
7           Once per time interval $\Delta$ do
8               $\beta \leftarrow \frac{h'_j(D_j) - h'_i(D_i)}{h''_j(D_j) + h''_i(D_i)}$
9               Pick $\eta$ according to the Theorem 2
10              $D_i \leftarrow D_i + \eta\beta$
11              $D_j \leftarrow D_j - \eta\beta$
12          enddo
12      endfor
```

Fig. 3. Pseudo-code of Asynchronous-IDC

$$\beta = \frac{h'_j(D_j) - h'_i(D_i)}{h''_j(D_j) + h''_i(D_i)}.$$

Now, the update step emerges directly from the above approximation, and the pseudocode for the Asynchronous-IDC (A-IDC) is given in Figure 3. However, we need to introduce a gain parameter $\eta \leq 1$ that protects the algorithm from overshooting, and ensures stability. Once the system is close to the steady state, the parameter η is close to 1 (this follows from eq. (13)), meaning that the parameter η is effective mainly in transient periods of searching for steady state. Note that the moment, at which the update over the edge (i,j) is performed, is not pre-specified, and can be anywhere in the interval $(n\Delta, (n+1)\Delta)$.

The following theorem establishes the convergence of A-IDC scheme. The proof can be derived using the same arguments as those from Theorem 1 and only an outline is sketched here.

Theorem 2: Let $D_i(t)$ be the value of D_i after t edges performed updates in A-IDC. Denote by $\eta(t)$ the value of η at time step t at which the updates over edge (i,j) takes place, and by $m_{ij}(t)$ and $M_{ij}(t)$ the minimum and maximum of numbers $h'_i(D_i(t)), h'_j(D_j(t))$, respectively. If $\eta(t)$ satisfies

$$\eta(t) = \min_{s \in \{i,j\}} \inf_{y_1, y_2 \in [m_{ij}(t), M_{ij}(t)]} \frac{h''_s(y_1)}{h''_s(y_2)}, \quad (13)$$

then the following limits exist:

$$\lim_{t\to\infty} D_i(t) = D_i^*$$

and

$$\lim_{t\to\infty} h'_i(D_i(t)) = \lim_{t\to\infty} h'_j(D_j(t)) =: q^*.$$

Proof: As in synchronous case, we define the minimum and maximum of $h'_1(D_1(t)), \ldots, h'_N(D_N(t))$: $m(t)$ and $M(t)$, respectively. The condition (13) ensures that $[m(t), M(t)]$ is a nested sequence of intervals. Once this is established, then following the same lines of the proof of the Theorem 1 one can show that $\lim_{t\to\infty} m(t) = \lim_{t\to\infty} M(t)$. ■

IV. Implementation issues

Comment 4: Here we assume that power management primitives that perform dynamic speed scaling locally are in place at each node. We also assume that there is network infrastructure that would allow shift of load between processors when needed.

Comment 5: The message passing between two nodes can cause some communication delay on a time scale from few milliseconds up to a couple of hundreds of milliseconds. These communication delays could cause some issues related to the stability of the distributed algorithms if the update interval is on some small time scale. However, the time between updates, given by Δ, should be on the order of magnitude of several seconds. This resulting separation of time-scales ensures that effects of the communication delays on the stability of our algorithms may be neglected. Notwithstanding this fact, the issue of delays is a topic for future research.

Comment 6: In our model, we assume that the set of job requests can be split in arbitrary manner. However, in the case where additional constraints exist (eg. certain subset of the demand set must be served by a particular cluster, or the location of clusters is important constraint for serving the requests (like in interactive applications), etc) the extension of the framework presented here to these domain is an open problem.

Comment 7: The presented work does not take into account effects of virtualization. Namely, if one runs several applications on the same hardware, the performance versus power curve becomes much harder to predict and control [43]. Therefore even though the algorithms for cost reduction are directly applicable in this context we are unable to evaluate the potential benefit of the framework presented here in virtualized environments.

Comment 8: Communication between servers is performed via small UDP packets. Each of those packets should contain a field for measuring the performance at the particular cluster, as well as some control overhead to ensure that if a loss of a communication packet occurs no node gains or loses extra demand, and that the constraint (1) is not violated.

V. Evaluation

In this section we present results from several representative simulations to illustrate the behavior of the algorithms discussed above. In particular we investigate the following: (1) The potential benefits in terms of cost reduction of the power-aware load balancing compared to oblivious $\frac{1}{N}$ load balancing; (2) The relationship between the aggregate load of the system and the cost reduction; (3) Scalability to large networks of IDC, A-IDC and MRS.

A. Potential benefits

In our first simulation, we look at potential savings that can be expected if the cloud-based service providers employed

N	number of servers
P_i	number of CPUs per server i
D	aggregate demand
D_i	demand allocated to cluster i
ν_i	cost of unit of power
$\bar{f}_i(D_i)$	energy consumed by cluster i
$f_i(D_i)$	cost of energy consumed by cluster i
μ	aggregate load $(D/(\sum_{i=1}^{N} P_i))$
α	exponent in dynamic speed scaling
γ	amortization parameter
η	gain parameter in IDC
c	overhead factor in MRS

TABLE I

SYMBOL MAP

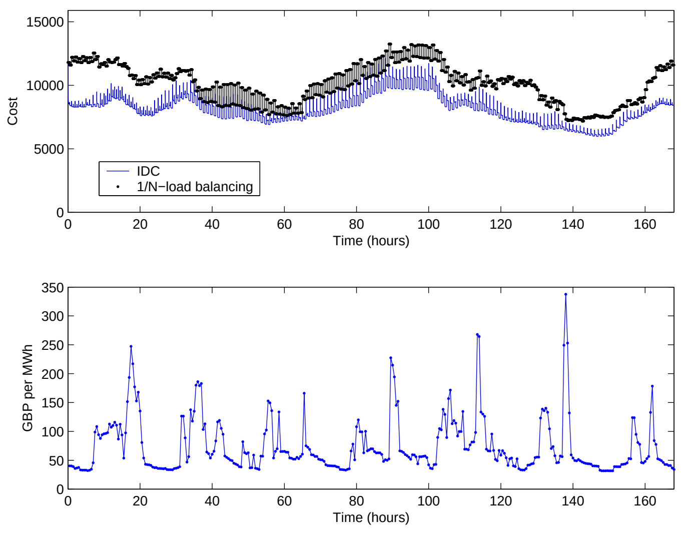

Fig. 4. Bottom: energy prices in GBP per MWh. Top: aggregate cost of running IDC algorithm compared to the aggregate cost of the dynamic speed scaling with $\frac{1}{N}$-load balancing (top). Time span of one week, 168 hours, from Monday 19/01/2009, 00:00:00 (GMT) to Sunday 26/01/2009 23:59:59 (GMT).

power-aware distributed speed scaling. Figure 4 (bottom) contains the graph of energy prices in (GBP per MWh) UK for a week in January 2009 [9]. Energy prices are formed once every 30 minutes, based on the demand and available supply. The setup is following. We consider $N = 12$ identical server clusters, with $P_i = P = 100$ CPUs at each cluster, located in even time zones (GMT, GMT+2,..., GMT+22) serving a constant load $D = 600$. We assume also that the energy-price at local time t is the same across all time zones and are driven by the values depicted in Figure 4 (bottom). We use the cubic power/speed dependance ($\alpha = 3$), with amortization factor $\gamma = 0.1$, with cost at each location being simply the product of energy consumed and power price. We compare the cost of the optimal operating point with the cost of the oblivious load balancing where each location processes $\frac{1}{N}$ of the total demand. The offered load is $\mu = \frac{D}{NP} = 50\%$. The results are depicted in Figure 4 (top).

To illustrate how the load allocated to a cluster evolves with time, and variable energy prices, we plot the load allocated to cluster 1 (located in GMT time zone) in Figure 5 (top). At the

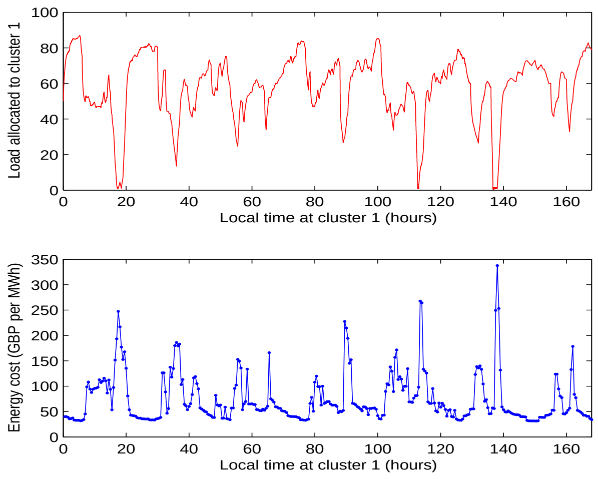

Fig. 5. Bottom: energy prices in GBP per MWh. Top: load allocated to cluster 1 (located in GMT time zone).

same figure (bottom), we depict the local cost of energy. As expected, the low energy price result in the high load allocated to the cluster, and the higher energy prices imply the lower amount of demand allocated to the cluster.

B. Aggregate load vs. cost reduction

In this subsection we evaluate the relationship between the aggregate load and the cost reduction compared to the oblivious $\frac{1}{N}$ load balancing in which each cluster processes $1/N$ of the total demand. For the setup described in Section V-A, we vary the aggregate load μ and plot it against the cost reduction defined as

$$CR(\mu) = 1 - \frac{V(\mathbf{D}^*(\mu))}{V(\frac{D_\mu}{N}, \ldots, \frac{D_\mu}{N})},$$

where $\mathbf{D}^*(\mu)$ is the optimal operating point (solution of (2)) and $(\frac{D_\mu}{N}, \ldots, \frac{D_\mu}{N})$ is the point corresponding to the power-oblivious $1/N$ load allocation. The value of $CR(\mu)$, for scenario described in previous subsection, is depicted in Figure 6 for two values of amortization parameter: $\gamma = 0.1$ and $\gamma = 0.5$. As expected, we can see that for low loads, the improvements are greater than for heavy loads, as in low-load cases, there is more space for balancing and (almost fully) avoiding the expensive clusters. However, we observe a non-monotonic dependance between the load and the cost reduction (in $\gamma = 0.1$ case) for which we do not have an analytic explanation.

C. Communication overhead

In this subsection, we compare the communication overhead of the three fully decentralized schemes that solve (2): MRS, IDC and A-IDC. The setup is following: N clusters process the load D with the communication graph being a 3-regular

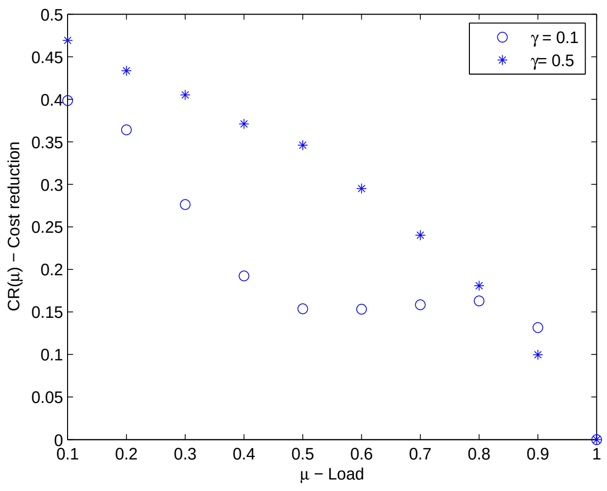

Fig. 6. Cost reduction vs. the aggregate load.

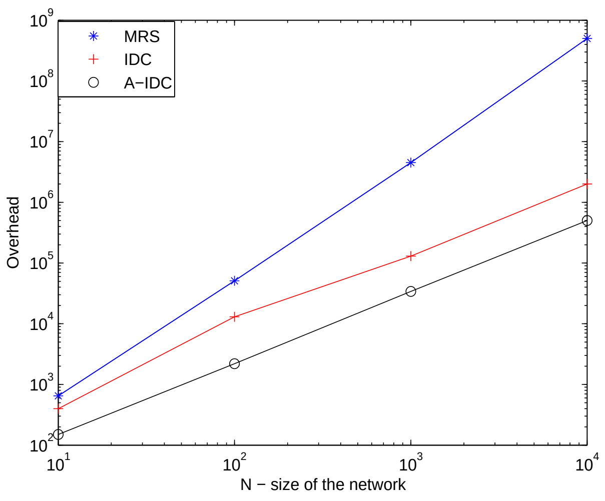

Fig. 7. Communication overhead versus network size, for MRS, IDC and A-IDC.

graph[6]. The cost functions[7] used here are

$$f_i(D_i) = \nu_i D_i^3,$$

where ν_i is drawn uniformly from the interval $(1, 5)$. The parameter η in IDC algorithm is set to the upper bound (12) that guarantees stability. The MRS parameter c, that controls the variance of the algorithm[8], is set to the value that corresponds to the expected relative error $\epsilon = 0.01$, and the remaining parameters of MRS are set following the rules from [30].

We vary the number of nodes in the network N from 10 to 10^4, and evaluate the communication *overhead* for MRS, IDC and A-IDC, defined as:

overhead = Number of IDC-messages exchanged to reach 1%-neighborhood of the optimal state.

We plot our findings in Figure 7. We can observe that the communication overhead of MRS is orders of magnitude greater than in the case of IDC and A-IDC. As we discussed in the previous subsections, the savings in IDC and A-IDC come from the fact that they completely avoid computation of the global state (sum of values from all nodes) which significantly reduces the communication overhead making them feasible for large-scale problems of type (2).

[6]d-regular graph is a graph in which every node has d neighbors.

[7]We stress that the objective here is not to apply algorithms proposed above in the power-minimization framework, but rather to compare the communication overhead of the decentralized control algorithms for one choice of cost functions.

[8]The parameter c represents the number of random variables sent in a single MRS-message used to reduce the variance of the sum estimator. Therefore, for header size of $h = 40$, one MRS-message has amortized cost of $(c + h)/(1 + h)$ IDC messages, that exchange a single value between two nodes.

VI. Conclusions

Issues related to service reliability, service availability, and fault tolerance, have encouraged many service providers in the Internet to shift from traditional centric services to cloud based services. This trend appears to be a sustained mechanism for ensuring robustness of internet services with many "big players", such as Google, Yahoo!, Akamai, Amazon, already offering a suit of cloud-based services.

While offering a huge computational power, those systems are major energy consumers. One of the techniques for CPU energy reduction is dynamic speed scaling, that adjusts the speed at which the processor is run based on current load. In this paper we propose a framework for utilizing dynamic speed scaling in a distributed setting. We propose a distributed algorithm for solving the optimization problem arising in this context that has significantly smaller communication overhead compared to state-of-the-art. Our evaluation supports our analytical findings and shows promise in non-negligible cost savings possible with schemes that exploit temporal and geographical diversity of energy prices and nonlinearity of the performance/power curve. However, the presented evaluation is limited to synthetic scenarios, and for better understanding of the benefits of large scale power-aware load-balancing, we need some realistic traffic models of load/power dependence in real-world implementations of dynamic speed scaling.

We conclude the paper with two recent ideas from two not directly related areas of computer networking, that can benefit from the distributed algorithm proposed here. The first one is load-aware channel width adaptation in wireless LANs. The basic idea is that with existing variability of demand in wireless LANs, allocating wider channels to access points with higher demand can dramatically improve WLAN performance. In the case of full mesh interference graph, the problem of load-aware spectrum allocation can be modeled as the following optimization problem

$$\min_{W_i \geq 0,\ \sum_{i=1}^N W_i = W} \sum_{i=1}^N U_i(W_i). \tag{14}$$

where W_i is the channel width devoted to access point i, W the total available spectrum, and U_i utility function that relates the demand at access point i to W_i. The existing solutions are centralized and are based on combinatorial heuristics for maximizing some specific performance objectives [29], [20].

The second proposal with a similar underlying problem can be found in the security literature. The distributed throttling for protection against DDoS attacks discussed in [45]. Relies on the concept of k-maxmin fairness that partitions the server capacity C among all routers that are k-hops away, such that specific performance goal is met (that can be reformulated as a convex problem of type (2)). The solution proposed there relies on a centralized controller. However we believe that the scheme proposed in this paper can be applied in that context, providing a fully decentralized solution for the problem of distributed throttling.

VII. Acknowledgements

This work is supported by the Science Foundation Ireland grant 07/IN.1/I901.

Appendix

Appendix A: Proof of Theorem 1.

We find it more convenient to write the dynamics of (11) in terms of $q_i(t)$:

$$g_i(q_i(t+1)) = g_i(q_i(t)) + \eta \sum_{(i,j)\in E} (q_j(t) - q_i(t)), \tag{15}$$

Step 1. First we prove that under condition (12) the sequence $m(t)$ is nondecreasing and the sequence $M(t)$ is nonincreasing. Let $q_i(t) = m(t) + \lambda$, for some $\lambda \geq 0$. Then from the equation (15) we have:

$$g_i(q_i(t+1)) = g_i(q_i(t)) + \eta \sum_{(i,j)\in E} (q_j(t) - m(t) - \lambda) \geq$$

$$g_i(q_i(t)) - \eta \sum_{(i,j)\in E} \lambda = g_i(q_i(t)) - \eta\lambda d_i = g_i(m(t)+\lambda) - \eta\lambda d_i. \tag{16}$$

Since g_i is a differentiable, function from the mean value theorem, there exists $q_i^*(t) \in (m(t), m(t)+\lambda) \subset (m(t), M(t))$ such that

$$g_i(m(t)+\lambda) = g_i(m(t)) + g_i'(q_i^*(t))\lambda$$

Combining the last inequality with (16) we get

$$g_i(q_i(t+1)) \geq g_i(m(t)) + g_i'(q_i^*(t))\lambda - d_i\eta\lambda. \tag{17}$$

For $t = 0$, the condition (12) implies that $g_i'(m(0))\lambda \geq d_i\eta\lambda$ implying that $g_i(q_i(1)) \geq g_i(m(0)$. Since g_i is increasing, we have $q_i(1) \geq m(0)$ for all i and thus $m(1) \geq m(0)$. Now we prove, using mathematical induction, that $m(t+1) \geq m(t)$, for all t. For $t = 0$ we just shoved that $m(1) \geq m(0)$. Suppose that for all $k < t$: $m(k+1) \geq m(k)$. Then, for $k = t$, $m(t) \geq m(0)$. Now, the condition (12) and the bound (17) imply:

$$g_i(q_i(t+1)) \geq g_i(m(t)) + g_i'(q_i^*(t)))\lambda - d_i\eta\lambda \geq$$

$$g_i(m(t)) + d_i\eta\lambda - d_i\eta\lambda = g_i(m(t)).$$

The last inequality, together with the fact that g_i is a decreasing function implies

$$m(t+1) = \min_{1\leq i\leq N} q_i(t+1) \geq m(t).$$

One can conclude using similar arguments that for all $t \geq 0$

$$M(t+1) = \max_{1\leq i\leq N} q_i(t+1) \leq M(t).$$

Thus the range of q_i's: $[m(t), M(t)]$ is a nested sequence of closed bounded intervals. In the next step of the proof we use this information to write the dynamics in the more compact form. **Step 2.** In this step we rewrite the dynamics of $q_i(t)$ in a more practical form. Consider the representation of the dynamics of $q_i(t)$ given by (15). From the Lagrange's mean value theorem there exist $r_i(t) \in (q_i(t), q_i(t+1)) \subset (m(0), M(0))$ such that

$$g_i(q_i(t+1)) - g_i(q_i(t)) = (q_i(t+1)) - q_i(t))g_i'(r_i(t)).$$

Thus the recurrence equation (15) can be rewritten as

$$q_i(t+1) = q_i(t) + \frac{\eta}{g_i'(r_i(t))} \sum_{(i,j)\in E} (q_j(t) - q_i(t)).$$

Therefore, the evolution of the state-vector $\mathbf{q}(t) = (q_1(t), \ldots, q_N(t))$ can be written as:

$$\mathbf{q}(t+1) = B(t)\mathbf{q}(t),$$

where the matrix $B(t)$ is given by

$$B(t) = \begin{bmatrix} 1 - \frac{d_1\eta}{g_1'(r_1(t))} & \frac{\eta}{g_1'(r_1(t))}e_{1,2} & \cdots & \frac{\eta}{g_1'(r_1(t))}e_{1,N} \\ \frac{\eta}{g_2'(r_2(t))}e_{2,1} & 1 - \frac{d_2\eta}{g_2'(r_2(t))} & \cdots & \frac{\eta}{g_2'(r_2(t))}e_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\eta}{g_N'(r_N(t))}e_{N,1} & \cdots & \cdots & 1 - \frac{d_N\eta}{g_N'(r_N(t))} \end{bmatrix}$$

with $e_{i,j}$ being the elements of the adjacency matrix of G, ie. if $(i,j) \in E$, then $e_{i,j} = 1$ otherwise $e_{i,j} = 0$.

Step 3. We now use the monotonicity of the sequences $M(t)$ and $m(t)$ that was shown in Step 1, to prove that nonzero elements of $B(t)$ are positive and uniformly bounded away from zero. Indeed, recall that $r_i(t) \in (q_i(t), q_i(t+1))$, and therefore $r_i(t) \geq \min(m(t), m(t+1)) \geq m(0)$, and therefore for the diagonal entries we have that

$$1 - \frac{d_i\eta}{g_i'(r_i(t))} \geq 1 - \eta\frac{\min_{1\leq i\leq N}(d_i)}{\inf_{y\in[m(0),M(0)]} g_i'(y)} \geq 1 - \frac{1}{2} = \frac{1}{2}.$$

For nonzero off-diagonal entries note that $r_i(t) \leq$

$\max(M(t), M(t+1)) \leq M(0)$ and thus

$$\frac{\eta}{g_i'(r_i(t))} \geq \frac{\eta}{\sup_{y \in [m(0), M(0)]} g_i'(y)} =: \delta_i > 0.$$

Take $\delta = \min\{\frac{1}{2}, \delta_1, \ldots, \delta_N\} > 0$. Then, for all t, all nonzero elements of $B(t)$ are not smaller than δ.

Step 4. Finally, we use the fact that G is connected to show that $M(t) - m(t)$ converges to zero. This will imply that $\lim_{t\to\infty} m(t) = \lim_{t\to\infty} M(t) = \lim_{t\to\infty} q_i(t) = q^*$. Let k be the diameter of graph G, i.e. the smallest integer such that there exist a path in G between each two nodes of length not greater than k. Then, for all t:

$$C(t) = B(t+k-1)B(t+k-2)\cdots B(t)$$

is a stochastic matrix[9] with strictly positive entries and each entry of $C(t)$ is greater or equal than δ^k. Denote by j_0 the index for which $q_{j_0}(t) = m(t)$. Then

$$q_i(t+k) = \sum_{j=1}^{N} C_{ij}(t) q_j(t) = \sum_{j \neq j_0} C_{ij}(t) q_j(t) + C_{ij_0}(t) m(t)$$

$$\leq \sum_{j \neq j_0} C_{ij} M(t) + C_{ij_0}(t) m(t) =$$

$$(1 - C_{ij_0}(t)) M(t) + C_{ij_0}(t) m(t) \leq M(t)(1-\delta^k) + m(t)\delta^k.$$

And similarly

$$q_i(t+k) = \sum_{j=1}^{N} C_{ij}(t) q_j(t) \geq m(t)(1-\delta^k) + M(t)\delta^k.$$

Thus

$$M(t+k) - m(t+k) \leq (1 - 2\delta^k)(M(t) - m(t)). \quad (18)$$

Since $M(t) - m(t)$ is a nonincreasing sequence and $\delta > 0$ is independent of t, we conclude that $M(t) - m(t) \to 0$, as $t \to \infty$. Thus

$$\lim_{t\to\infty} M(t) = \lim_{t\to\infty} m(t) = \lim_{t\to\infty} q_i(t) = q^*.$$

Now, the convergence of $D_i(t)$ follows directly from (9) and the continuity of the mappings g_i.

Comment 9: From the bound (18), we can observe that the system converges to the equilibrium exponentially, with a rate bounded above by $(1-2\delta^k)^{\frac{1}{k}}$. Indeed, let us introduce the following quantity:

$$\theta = \frac{1}{1-2\delta^k} \max_{0 \leq s \leq k} (M(s) - m(s)).$$

Then from (18):

$$M(t) - m(t) \leq (1-2\delta^k)^{\lfloor \frac{t}{k} \rfloor}(M(s) - m(s)) \leq$$

$$(1-2\delta^k)^{\frac{t}{k}} \frac{1}{1-2\delta^k}(M(s) - m(s)) \leq \theta \left((1-2\delta^k)^{\frac{1}{k}} \right)^t.$$

[9]A stochastic matrix is a square matrix with nonnegative entries and where sum of each row is 1. Since each of $B(t)$ is stochastic, their product is stochastic as well [10].

Appendix B: A short description of the MRS algorithm.

Here we will briefly describe the main idea of the MRS algorithm [30]. While the basic algorithm is concerned with the case of arbitrary number of linear constraints, here we focus on the problem (2) with a single linear constraint. Basically it is a dual ascent method, relying on the distributed aggregate SUM subroutine for the update step.

The MRS method works as follows. Each node i, tracks the dual variable λ. Let $\lambda(k)$ be the value of the dual variable at the k-th iteration. Then $D_i(k)$, the value of the primal variable at k-th iteration is given by.

$$D_i(k) = \arg \inf_{D_i > 0} (f_i(D_i) - \theta \ln(D_i) + \lambda(k) D_i).$$

A calculation then shows that the gradient:

$$\nabla g(\lambda(k)) = \sum_{i=1}^{N} D_i(k) - D.$$

To compute $\sum_{i=1}^N D_i(k)$, authors use fully distributed algorithm from [31], that produces an (unbiased) estimate $\hat{s}(k)$ of the sum $\sum_{i=1}^N D_i(k)$, with variance that is a decreasing function of the overhead factor c. The dual variable, is then updated using the following rule, and the iterative process proceeds

$$\lambda(k+1) = \lambda(k) + t(\hat{s}(k) - D),$$

where the step size t is governed by the variation in curvature of the Lagrange dual function.

Comment 10: The MRS method has the flavor very similar of the Kempe&McSherry (KM) algorithm for distributed spectrum computation [26]. Basically both MRS and KM are incarnations of well known iterative algorithms: dual ascent in case of MRS and QR algorithm for eigenvalue problem in case of KM. Then in the iterative update step both algorithms use subroutines for aggregate sum computation: MRS uses method from [31], while KM uses [25]. And finally, in both MRS and KM the main technical difficulty arises from the analysis of the conditions that ensure that error induced by the (randomized) aggregate sum subroutine, does not harm the convergence of the iterative algorithms.

References

[1] S. Albers, H. Fujiwara. "Energy-efficient algorithms for flow time minimization". In Proc. of STACS 2006,

[2] A. Anagnostopoulos, A. Kirsch, E. Upfal. "Load Balancing in Arbitrary Network Topologies with Stochastic Adversarial Input". SIAM J. Computing, vol 34(3): 616-639 (2005).

[3] E. Altman, U. Ayesta, B.J. Prabhu. "Optimal load balancing in Processor Sharing systems". Proc. of GameComm 2008.

[4] M. Armbrust, et al. "Above the Clouds: A Berkeley View of Cloud Computing". University of California at Berkeley, Technical Report No. UCB/EECS-2009-28.

[5] Y. Azar, A. Broder, A. Karlin, E. Upfal. "Balanced Allocations". In Proc. of ACM STOC, pp. 593-602, 1994.

[6] Amazon Web Services. http://aws.amazon.com/.

[7] Amazon Web Services, Case Studies. http://aws.amazon.com/solutions/case-studies/.

[8] N. Bansal, K. Pruhs, C. Stein. “Speed scaling for weighted flow times”. In Proc. of SODA 2007.

[9] UK energy prices. http://www.bmreports.com/bw$x_$reporting.htm.

[10] A. Berman, R. Plemmons. ”Nonnegative matrices in the mathematical sciences”. SIAM, 1979.

[11] S. Boyd, A. Ghosh, B. Prabhakar, D. Shah. “Gossip algorithms: Design, analysis and applications”. In Proc. of IEEE Infocom, 2005

[12] D. F. Carr. “How Google works”. Baseline Magazine, July 2006.

[13] R. Chandra, R. Mahajan, T. Moscibroda, R. Raghavendra, P. Bahl. “A Case for Adapting Channel Width in Wireless Networks”. In Proc. of ACM SIGCOMM 2008.

[14] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, F. Zhao. “Energy-aware server provisioning and load dispatching for connection-intensive internet services. In Proc. of NSDI 2008.

[15] Akamai, State of the Internet, 2008. Available online: http://www.akamai.com/stateoftheinternet/

[16] EPA Report on Server and Data Center Energy Efficiency. U.S. Environmental Protection Agency, 2007.

[17] A. Gandhi, M. Harchol-Balter, R. Das, C. Lefurgy. “Optimal Power Allocation in Server Farms”. In Proc. of ACM SIGMETRICS 2009.

[18] D. Goldenberg, L. Qiu, H. Xie, Y. Yang, Y. Zhang. “Optimizing cost and performance, for multihoming”. In Proc. of ACM SIGCOMM 2004.

[19] A. Greenberg, J. Hamilton, D .A. Maltz, P. Patel. ”The Cost of a Cloud: Research Problems in Data Center Networks”. ACM Computer Communications Review, vol 39(1) 2009.

[20] R. Gummadi, H. Balakrishnan. “Wireless Networks Should Spread Spectrum Based On Demands”. In Proc. of Hotnets 2008.

[21] K. Church, A. Greenberg, J. Hamilton. “On Delivering Embarrassingly Distributed Cloud Services”. In proc of Hotnets 2008.

[22] W. Hammond. “Efficient power consumption in the modern Datacenter”. Available online: www.research.ibm.com/aceed/2005/proceedings/hammond.ppt

[23] A. Jadbabaie, J. Lin, and A. S. Morse, “Coordination of groups of mobile autonomous agents using nearest neighbor rules”. IEEE Transactions on Automatic Control, vol. 48(6), 2003

[24] F.P. Kelly, A.K. Maulloo, D.K.H. Tan. “Rate control for communication networks: shadow. prices, proportional fairness and stability”. Journal of the Operational Research Society, Vol. 49 (3), pp. 237-252, March 1998.

[25] D. Kempe, A. Dobra, J. Gehrke. “Gossip-based computation of aggregate information”. In Proc. of IEEE FOCS, 2003.

[26] D. Kempe, F. McSherry. “A decentralized algorithm for spectral analysis”. In Proc. of STOC 2004.

[27] N. Laoutaris, P. Rodriguez, L. Massoulie. “ECHOS: Edge Capacity Hosting Overlays of Nano Data Centers”. ACM/SIGCOMM CCR, vol 38(1) 2008.

[28] M. Mitzenmacher. “The Power of Two Choices in Randomized Load Balancing”. IEEE Trans. on Par. and Dist. Systems, vol. 12(10), 2001.

[29] T. Moscibroda, R. Chandra, Y. Wu, S. Sengupta, P. Bahl. “Load-aware spectrum distribution in wireless LANs”. In Proc. of ICNP 2008.

[30] D. Mosk-Aoyama, T. Roughgarden, D. Shah. “Fully Distributed Algorithms for Convex Optimization Problems”. In Proc. of DISC 2007.

[31] D. Mosk-Aoyama, D. Shah. “Computing separable functions via gossip”. In Proc. of PODC 2006.

[32] L. Moreau. “Stability of multiagent systems with time-dependent communication links”. IEEE Trans. Aut. Control, 2005.

[33] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, D. Wetherall. “Reducing network energy consumption via sleeping and rate-adaptation”. In Proceedings of NSDI, 2008.

[34] H. X. Nguyen, D. Figueiredo, M. Grossglauser, P. Thiran. “Balanced Relay Allocation on Heterogeneous Unstructured Overlays”. In Proceedings of IEEE Infocom 2008, Phoenix, AZ, USA.

[35] A. Odlyzko. “Internet pricing and the history of communications”. Computer Networks, vol. 36, 2001.

[36] A. Qureshi. “Plugging Into Energy Market Diversity”. In Proc. of Hotnets 2008.

[37] A.E. Papathanasiou, M.L. Scott. “Energy Efficient Prefetching and Caching”. In Proc. of USENIX 2004.

[38] G. Peng. “CDN: Content Distribution Network”. Online: http://arxiv.org/abs/cs.NI/0411069.

[39] B. Raghavan, K. Vishwanath, S. Rambhadran, K. Yocum, A. Snoeren. “Cloud Control with Distributed Rate Limiting”. ACM SIGCOMM 2007.

[40] Z.S. Rui, N. McKeown. “Designing a Predictable Internet Backbone with Valiant Load-Balancing”. IWQoS 2005.

[41] R. Olfati-Saber. “Flocking for multi-agent dynamic systems: algorithms and theory”. IEEE Trans. on Auto. Control, 2006.

[42] R. Srikant. “Internet congestion control”. Control theory, 14, Birkhäuser Boston Inc., Boston, MA, 2004.

[43] S. Srikantaiah, A. Kansal, F. Zhao. “ Energy Aware Consolidation for Cloud Computing”. In Proc. of Hotpower 2008.

[44] R. Stanojevic, R. Shorten. “Fully decentralized emulation of best-effort and processor sharing queues”. In Proceedings of ACM Sigmetrics 2008.

[45] C.W. Tan, D.M. Chiu, J.C.S. Lui, D.K.Y. Yau. “A Distributed Throttling Approach for Handling High Bandwidth Aggregates”. IEEE Transactions on Parallel and Distributed Systems, vol 18(7), 2007.

[46] A. Wierman, L.L.H. Andrew, A. Tang. “Power-Aware Speed Scaling in Processor Sharing Systems”. In Proc. of IEEE Infocom 2009.