Joint Optimization of Computing and Cooling Energy: Analytic Model and A Machine Room Case Study

Shen Li *, Hieu Le*, Nam Pham†, Jin Heo‡, Tarek Abdelzaher*§

* Department of Computer Science
University of Illinois at Urbana-Champaign, USA
† Lund University, Sweden (Sabbatical Affiliation)
‡ Caterva, Inc.
§ VMWare, Inc.
{shenli3, hieule2, nampham2}@illinois.edu, heoj@vamware.com, zaher@illinois.edu

Abstract—Total energy minimization in data centers (including both computing and cooling energy) requires modeling the interactions between computing decisions (such as load distribution) and heat transfer in the room, since load acts as heat sources whose distribution in space affects cooling energy. This paper presents the first closed-form analytic optimal solution for load distribution in a machine rack that minimizes the sum of computing and cooling energy. We show that by considering actuation knobs on both computing and cooling sides, it is possible to reduce energy cost comparing to state of the art solutions that do not offer holistic energy optimization. The above can be achieved while meeting both throughput requirements and maximum CPU temperature constraints. Using a thorough evaluation on a real testbed of 20 machines, we demonstrate that our simple model adequately captures the thermal behavior and energy consumption of the system. We further show that our approach saves more energy compared to the state of the art in the field.

Keywords—data-center; optimal load allocation; energy optimization; testbed; practical;

I. INTRODUCTION

Understanding, controlling, and optimizing the energy interactions at machine room scale is an important prerequisite to reducing the energy cost and carbon footprint of data centers, which alone are responsible for more than 1.5% of the total energy consumption in the nation. Recent work has started to address joint energy minimization of computing and cooling [1], [2], [3], [4], [5], [6], [7]. Compared to these efforts, ours is the first paper that provides an analytic optimal solution to the problem of combined load distribution and cooling temperature setting in machine rooms (for a simple load and cooler model).

This paper provides a closed-form analytic solution to the optimal load distribution question by developing a simplified model of both computing and cooling components, and uses it to solve the energy-optimal load allocation problem. Besides that, we also present an efficient algorithm for selecting a subset of servers to handle incoming workload, which guarantees optimality under our model. The solution is evaluated empirically on 20-machine testbed. We compare our solution to baselines such as uniform allocation, cool allocation nearest to the floor, and an approach where computing and cooling energy are minimized separately. Experiment results show that total savings in excess of 5% are possible, reaching as far as 18% (even in this very small set-up), over these baselines. We expect that savings in larger systems will be more pronounced, as larger spatial diversity gives rise to more opportunities for optimization.

Our paper focuses on cloud computing workloads, where long computationally-intensive tasks (such as batch processing of click-streams) are distributed among several machines for concurrent handling. In these systems, the total load is steady, and load distribution across machines can be decided by a central load balancer. Since tasks are long-lived, we can ignore initial transients and get away with a steady-state analysis of energy consumption. This approach is not suitable for energy optimization in the presence of dynamic workloads such as those posed by real-time requests on server farms (e.g., the load on live e-shopping, sports, or news websites). In such dynamic situations, changes in load entail changes in server temperature. Hence, servers are never at steady state, and our steady state analysis is not appropriate. With the increasing proliferation of cloud computing applications, however, it is becoming more interesting to look at the type of batch processing loads described in this paper. Hence, the models investigated in our work remain of both wide and increasing applicability.

One should also stress that our goal is to develop and validate a closed-form optimal solution for load distribution across machines (in the sense of total energy minimization). To arrive at closed-form results, we seek simplified models. In particular, it is not our goal to determine the most faithful model of computing and thermal interactions in data centers. Instead, we aim to check whether a simplified model is sufficient to arrive at a solution that achieves a non-trivial improvement in energy savings. This question is addressed by implementing our solution on a real server rack and comparing it with others from literature to ascertain that higher energy savings are indeed attained despite modeling
simplifications. If the models are improved, further savings may become possible, but such refinements are beyond the scope of this work. An advantage of our simplified approach is that our closed-form results offer basic insights into the structure of the solution to the load distribution problem that might be harder to gather from more complex solutions and algorithms.

The rest of this paper is organized as follows. Section II considers the physical models of data centers then formulates the energy optimization problem. Section III proposes a scalable algorithm to to the energy optimization problem. Section IV shows the experimental results from the testbed computing and cooling energy consumption of a machine room, by intelligently sharing load across a subset of machines, due to differences in thermal load distribution across machines, as well as on the input flow temperature. For a given air flow, let us denote the coefficient of air volume, and an air flow out of the air volume. For simplicity, we assume perfect and immediate air mixing. Also, we assume that the heat exchange due to diffusion of the air around the heat source, an air flow into the air volume. The constraints in our model are the total load and maximum allowable CPU core temperature. We show that the optimal solution has a slightly imbalanced load distribution across machines, due to differences in their location in the room and hence differences in thermal properties.

A. Physical Model for a Single Computing Unit

For the purpose of thermal analysis, a computing unit, \( i \) (a rack/blade server), comprises a heat source (the CPU), an air volume around the heat source, an air flow into the air volume, and an air flow out of the air volume. For simplicity, we assume perfect and immediate air mixing. Also, we assume that the heat exchange due to diffusion of the air volume into the outer environment is negligible compared to the heat exchange caused by the intake and outtake air flow. Let \( F_{\text{in}}^i \) denote the intake air flow of the \( i \)th computing unit, \( F_{\text{out}}^i \) denote its mixed output flow, \( T_{\text{in}}^i \) and \( T_{\text{out}}^i \) denote the temperatures of the two flows, respectively, and \( T_{\text{CPU}}^i \) denote the temperature of the CPU. Thus, we can write:

\[
\frac{dT_{\text{CPU}}^i}{dt} = \frac{1}{\mu_{\text{CPU}}^i}(P_i - (T_{\text{CPU}}^i - T_{\text{out}}^i)\varphi_{\text{CPU},box}^i) \tag{1}
\]

where, \( \mu_{\text{CPU}}^i \) is the heat capacity of the CPU, \( P_i \) is the rate at which the \( i \)th computing unit consumes power, and \( \varphi_{\text{CPU},box}^i \) is the heat exchange rate between the CPU and the air volume. We can also write:

\[
\frac{dT_{\text{box}}^i}{dt} = \frac{1}{\rho_{\text{box}}^i}((T_{\text{CPU}}^i - T_{\text{out}}^i)\varphi_{\text{CPU},box}^i + F_{\text{in}}^i c_{\text{air}}T_{\text{in}}^i - F_{\text{out}}^i c_{\text{air}}T_{\text{out}}^i) \tag{2}
\]

where \( \rho_{\text{box}}^i \) be the heat capacity of the air volume inside the computing unit, and \( c_{\text{air}} \) is the specific heat of air. Table I describes physical units of the variables used so far in this section. (The machine index, \( i \), is omitted for clarity.)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Unit</th>
<th>Physical Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_{\text{CPU}} ), ( T_{\text{box}} ), ( T_{\text{in}} )</td>
<td>( K )</td>
<td>(Kevin) Temperature</td>
</tr>
<tr>
<td>( \mu_{\text{CPU}}^i ), ( \rho_{\text{box}}^i )</td>
<td>( JK^{-1} )</td>
<td>Heat Capacity</td>
</tr>
<tr>
<td>( \varphi_{\text{CPU},box}^i )</td>
<td>( JK^{-1}s^{-1} )</td>
<td>Heat Exchange Rate</td>
</tr>
<tr>
<td>( F_{\text{in}}^i ), ( F_{\text{out}}^i )</td>
<td>( m^3s^{-1} )</td>
<td>Air Flow</td>
</tr>
<tr>
<td>( c_{\text{air}} )</td>
<td>( JK^{-1}m^{-3} )</td>
<td>Heat Capacity Density</td>
</tr>
<tr>
<td>( P_{\text{CPU}}^i )</td>
<td>( Js^{-1} )</td>
<td>Heat Producing Rate</td>
</tr>
</tbody>
</table>

Table I

PHYSICAL VARIABLES AND THEIR UNITS

At steady state, we have \( dT_{\text{CPU}}^i / dt = 0 \) and \( dT_{\text{box}}^i / dt = 0 \). Substituting in Equation (1) and Equation (2) respectively, we have:

\[
P_i = (T_{\text{CPU}}^i - T_{\text{out}}^i)\varphi_{\text{CPU},box}^i \tag{3}
\]

and:

\[
P_i = F_{\text{out}}^i c_{\text{air}}T_{\text{box}}^i - F_{\text{in}}^i c_{\text{air}}T_{\text{in}}^i \tag{4}
\]

Solving Equation (3) for \( T_{\text{out}}^i \), then substituting in Equation (4) and rearranging, we get:

\[
T_{\text{CPU}}^i = (\frac{1}{F_{\text{out}}^i c_{\text{air}}} + \frac{1}{\varphi_{\text{CPU},box}^i})P_i + T_{\text{in}}^i \tag{5}
\]

The above equation shows the dependency of \( T_{\text{CPU}}^i \) on the unit’s power consumption (which is related to its computing load), as well as on the input flow temperature. For a given air flow, let us denote the coefficient of \( P_i \) in the above equation by \( \beta_i \):

\[
\beta_i = \frac{1}{F_{\text{out}}^i c_{\text{air}}} + \frac{1}{\varphi_{\text{CPU},box}^i} \tag{6}
\]

It remains to relate the input flow temperature to the output flow temperature of the cooling unit, \( T_{\text{ac}} \). In a large machine room (or data center), this relation depends on how close or far the unit is from the cold air vent. Hence, we set:

\[
T_{\text{in}}^i = \alpha_i T_{\text{ac}} + \gamma_i \tag{7}
\]

Substituting from the above equation into Equation (5), we get:
Finally, from the approximate relationship between energy consumption of a CPU and its load is described by [8]:

\[ P_{i} = w_{1}L_{i} + w_{2} \]  

In this equation, \( P_{i} \) is the power consumption of server \( i \), and \( L_{i} \) is the load on that server. The equation simply states that the power consumption is attributed in part to the load and in part to other (load-independent) components. Equation (8) and Equation (9) are the final model of the computing unit. They are the two equations needed to model the relationship among CPU temperature, computing power, cooling temperature, and computing load in our optimization. Next, we model the power draw of cooling unit.

### B. Model for Cooling System

The air cooler used in our machine room features an internal control loop that manipulates the flow of chilled water in order to maintain air temperature at a specified set point, \( T^{SP} \). The control loop compares exhaust air temperature to the thermal set point, hence effectively controlling temperature of the hot exhaust air returning from the room, as opposed to the temperature of the cooling air, \( T^{ac} \), pumped into the room. This choice closes the control loop, making the temperature control responsive to the thermal load, since it is the exhaust temperature, not the room inlet temperature, that depends on the amount of heat generated in the room. At steady state, the exhaust temperature is kept at the fixed set point temperature, \( T^{SP} \). The power consumed in cooling air from the exhaust temperature, \( T^{SP} \), to the cool air temperature, \( T^{ac} \), is \( c^{air} \cdot f^{ac} \cdot (T^{SP} - T^{ac}) \), where \( c^{air} \) is the heat capacity constant, and \( f^{ac} \) is the flow rate of the cooling unit. In reality, the cooling unit has an efficiency \( \eta < 1 \), since some energy consumed by it simply dissipates. The above equation therefore needs to be divided by \( \eta \) to compute for the actual total power consumption of the unit, denoted \( P^{ac} \). Let us define the constant \( c \) as \( c = c^{air} / \eta \). Thus:

\[ P^{ac} = c \cdot f^{ac} \cdot (T^{SP} - T^{ac}) \]  

### C. Problem Formulation

Based on the above models, the optimal load allocation problem is formulated as follows. Let set \( ON \) be the total set of machines that are turned on. Let \( L_{i} \) be the amount of load allocated to server \( i \in ON \). The problem is to find \( T^{ac} \) and \( L_{i} \) (for all machines \( i \in ON \)) to:

- minimize:

\[ P^{total} = P^{ac} + \sum_{i \in ON} P_{i}, \]

where (from Equation (9) and Equation (10)):

\[ P_{i} = w_{1}L_{i} + w_{2} \]

\[ P^{ac} = c \cdot f^{ac} \cdot (T^{SP} - T^{ac}) \]

Subject to:

\[ \sum_{i \in ON} L_{i} = L \]

and:

\[ \forall i \in ON : T^{cpu}_{i} \leq T^{max} \]

where (from Equation (8)):

\[ T^{cpu}_{i} = \alpha_{i} T^{ac} + \beta_{i} P_{i} + \gamma_{i} \]

III. OPTIMAL SOLUTION

The general procedure to solve the above optimization is the Lagrange’s multiplier method with Kuhn-Tucker conditions. In the following two subsections, we first solve the problem for an arbitrary set of turned-on machines, \( ON \). We then describe an efficient algorithm for deciding the optimal set of machines to keep on. In the following section, all summations over machine index \( i \) implicitly denote the range \( i \in ON \). For clarity, explicit mention of set \( ON \) is omitted when clear from context.

#### A. Optimal Load Distribution

The Lagrangian, \( G \), corresponding to the optimization problem presented in the previous section can be expressed as follows:

\[ G = c \cdot f^{ac} \cdot (T^{SP} - T^{ac}) + \sum_{i} (w_{1}L_{i} + w_{2}) + \lambda (L - \sum_{i} L_{i}) + \sum_{i} \mu_{i} (T^{cpu}_{i} - T^{max}) \]  

While, in principle, it may be interesting to investigate the effect of changing flow \( f^{ac} \) as well, in our actual testbed, the air conditioning unit maintains a fixed flow to keep the rate of air circulation in the room constant. Hence, we do not consider this degree of freedom in our solution. Substituting from Equation (8) for \( T^{cpu}_{i} \):
\[
G = cf^{ac}(T^{SP} - T^{ac}) + \sum_i (w_i L_i + w_2) \\
+ \lambda (L - \sum_i L_i) \\
+ \sum_i \mu_i (\alpha_i T^{ac} + \beta_i P_i + \gamma_i - T^{\text{max}})
\]

(12)

At the optimal point, the partial derivation of the above equation with respect to \(T^{ac}\) and \(L_i\) is zero. Hence, we get the following equations:

\[
\frac{\partial G}{\partial T^{ac}} = -cf^{ac} + \sum_i \mu_i \alpha_i = 0
\]

(13)

and

\[
\forall i : \frac{\partial G}{\partial L_i} = \lambda - \mu_i \beta_i w_1 = 0
\]

(14)

From Equation (14), we get:

\[
\mu_i = \frac{\lambda}{\beta_i w_1}
\]

(15)

Substituting for \(\mu_i\) in Equation (13), we get:

\[
\lambda = \frac{cf^{ac} w_1}{\sum_i (\alpha_i/\beta_i)}
\]

(16)

Substituting for \(\lambda\) in Equation (15), it becomes clear that the Lagrange coefficients \(\mu_i\) are non-zero. In fact, \(\lambda\) and \(\mu_i\) are strictly positive, according to the above equations, because all the quantities on the right-hand-side are positive. Hence, the optimal solution lies at the corresponding constraint boundaries, when \(\partial G/\partial \lambda = 0\) and \(\partial G/\partial \mu_i = 0\). The latter condition yields:

\[
T_{i \text{cpu}}^{\text{opt}} = T^{\text{max}}
\]

(17)

Substituting for \(T_{i \text{cpu}}^{\text{opt}}\) from Equation (8) into Equation (17), then substituting, in the result, for \(P_i\) from Equation (9), and finally solving for \(L_i\), we get:

\[
L_i = \frac{T^{\text{max}} - \alpha_i T^{ac} - \beta_i w_2 - \gamma_i}{\beta_i w_1}
\]

\[
= K_i - \frac{T^{ac}}{w_1 \beta_i}
\]

(18)

where:

\[
K_i = (T^{\text{max}} - \beta_i w_2 - \gamma_i)/(\beta_i w_1)
\]

(19)

Note that \(K_i\) is a constant. Substituting from Equation (18) into the constraint \(\sum_i L_i = L\), we get:

\[
\sum_i K_i - \frac{T^{ac}}{w_1} \sum_i \frac{\alpha_i}{\beta_i} = L
\]

(20)

Solving for \(T^{ac}\), we get the optimal cooling air temperature:

\[
T^{ac} = \frac{(\sum_i K_i - L) w_1}{\sum_i (\alpha_i/\beta_i)}
\]

(21)

where \(K_i\) is given by Equation (19). Finally, substituting for \(T^{ac}\) from Equation (21) into Equation (18), the optimal load for processor \(i\) is:

\[
L_i = K_i - \frac{w_1}{\sum_i (\alpha_i/\beta_i)} (\sum_i K_i - L)
\]

(22)

Equation (21) and Equation (22) (together with the definition of \(K_i\) in Equation (19)) present the closed form solution to the problem of finding optimal cooling temperature and optimal load distribution across a set of turned on machines \(ON\), that obeys both load and CPU temperature constraints. Since load \(L\) is the only variable in the two equations, it takes linear computational complexity (with respect to the number of servers) to derive workload assignment and AC set point.

While it may seem at first that the number of various constants in these equations is overwhelming, these constants (namely, \(w_1, w_2, \alpha_i, \beta_i, \gamma_i,\) and \(K_i\)) can be computed from two sets of experiments. The first empirically determines \(w_1\) and \(w_2\) by fitting machine power and load measurements to Equation (9). The second empirically determines \(\alpha_i, \beta_i,\) and \(\gamma_i\) of each machine by fitting its CPU temperature, power and cooler air temperature to Equation (8). Observe, from Equation (19), that \(K_i\) is computed from the other constants and from \(T^{\text{max}}\) (the maximum CPU temperature constraint).

Note also that the above solution makes no assumptions on where the machines are in the data center. The location of machine \(i\) (e.g., with respect to cool air vents) merely affects the relation between cool air temperature \(T^{ac}\), and the temperature at the inlet of the machine, as captured by constants \(\alpha_i\) and \(\gamma_i\) in Equation (7). The location with respect to vents may also affect the air flow rate through the machine, as captured by constant \(\beta_i\) in Equation (6). Hence, these constants, once profiled for each machine as described above, implicitly account for its placement in the data center. We detail in the evaluation section how profiling was done.

### B. Optimal Load Consolidation

Next, it is desired to find the set of machines to be turned on such that total computing and cooling energy is minimized. At the first glance, the problem of deciding which machines to keep on seems like a Mixed Integer Programming problem. A naive algorithm which checks all possibilities takes \(O(n2^n)\) time to compute the optimal solution. This is obviously not practical even when the cluster contains only a few tens of machines.

To tackle this problem, we first do some preliminaries. Since we already have the equation for \(T^{SP}\), and \(L_i\), the overall power consumption \(P^{\text{total}}\) is:
The following equation
\[ P_{\text{total}} = P_{\text{ac}} + \sum_{i} P_{i} \]
\[ = c f^{\text{ac}}(T S P - w_{1} \left( \sum_{i} K_{i} - L \right)) + \sum_{i} (w_{1} I_{i} + w_{2}) \]
\[ = kw_{2} - \rho \sum_{i} a_{i} - L \sum_{i} b_{i} + \theta \] (23)

where
\[ \rho = c f^{\text{ac}}w_{1} \]
\[ k = [ON] \]
\[ L = c f^{\text{ac}}T S P + w_{1} L \]
\[ a_{i} = K_{i} \]
\[ b_{i} = \alpha_{i} / \beta_{i} \]

Since \( w_{2}, \rho, a_{i}, b_{i} \), are all constants, \( \theta \) is also fixed with a given load \( L \), and \( kw_{2} \) is a simple linear equation with respect to \( k \), the problem reduces to given a set of pairs \( \mathcal{A} = \{(a_{1}, b_{1}), (a_{2}, b_{2}), \ldots, (a_{n}, b_{n})\} \), a positive real number \( L \), and a positive integer \( k \), find a subset \( ON = \{(a_{i1}, b_{i1}), (a_{i2}, b_{i2}), \ldots, (a_{in}, b_{in})\} \) that maximizes the following equation
\[ \sum_{j=1}^{k} a_{ij} - L \sum_{j=1}^{k} b_{ij}. \]

Note that, if we can solve the above problem, it is easy to go back to the original problem by enumerating \( k \) from 1 to \( n \). Call this problem \( \text{select}(\mathcal{A}, k, L) \). Simple heuristics \(^1\) are able to offer some local optimal, but there’s no guarantee that global optimality can be achieved.

The above reduction eliminates irrelevant parameters, and offers an equivalent abstraction of the original problem. However, it still takes exponential time to calculate optimality with naive solutions. Below, we do another manipulation to present a polynomial time algorithm.

Let’s first try to answer a different question: with a given power budget \( P_{b} \), and an integer \( k \), find the maximum load \( L_{\text{max}} \) that the cluster can serve without violating \( P_{b} \) using exactly \( k \) servers. Call it \( \text{maxLo}(\mathcal{A}, P_{b}, k) \) problem. Notice that, \( L_{\text{max}} \) increases monotonously with \( P_{b} \) (may not necessarily be continuously). Therefore, if we can solve \( \text{maxLo}(\mathcal{A}, P_{b}, k) \) problem, we can go back to the \( \text{select}(\mathcal{A}, k, L) \) problem by performing a binary search on \( P_{b} \) to find the minimum power that can serve a given load \( L \). Below, we describe the idea to solve \( \text{maxLo}(\mathcal{A}, P_{b}, k) \) problem.

Given a fixed \( P_{b} \), we define \( t \) such that,
\[ P_{\text{total}} \leq P_{b} = kw_{2} - pt + \theta. \] (24)

Substitute Equation (23) into Equation (24), we have
\[ \sum_{j=1}^{k} a_{ij} - L \sum_{j=1}^{k} b_{ij} \geq t. \] (25)

By doing simple algebra with Equation (25), we get Equation (26):
\[ \sum_{j=1}^{k} x_{ij}(t) \geq L \]
\[ \sum_{j=1}^{k} x_{ij}(t) = a_{ij} + t(-b_{ij}) \]

Now, our goal is to find the largest \( L \). Let us imagine one physical system to demonstrate the meaning of Equation (26). In this system, there are \( n \) particles moving in an one-dimensional space, where \( a_{ij} \) and \( -b_{ij} \) denote the initial coordinate and speed of particle \( i \). Let \( t \) denote time. Then, \( x_{ij}(t) \) represents the coordinate of particle \( i \) at time \( t \). Equation (26) actually checks whether the sum of \( k \) particles’ coordinates is greater than \( L \). Notice that, if the total order of \( n \) particles’ coordinates is fixed, selecting the largest \( k \) coordinates always results in the largest \( \sum_{j=1}^{k} x_{ij}(t) \), which yields largest possible \( L_{\text{max}} \). Therefore, given \( t \), we can compute \( L_{\text{max}} \) in \( O(n \lg n) \) (sorting all coordinates \( x_{ij}(t) \)) time. Since there is a linear relationship between \( t \) and \( P_{b} \) (Equation (24)), by performing binary search on \( P_{b} \), it takes \( O(n \lg n \lg P_{\text{max}}) \) time to compute the optimal solution, where \( P_{\text{max}} \) is the maximum possible power consumption of the whole cluster.

The above algorithm depends on the value of \( P_{\text{max}} \), and it is not fast enough as an on-line algorithm. Below, based on the solution of \( \text{maxLo}(\mathcal{A}, P_{b}, k) \), we further improve the result, and present a strictly polynomial time algorithm as well as a much more efficient on-line query algorithm by explicitly identifying all different possible coordinates’ orders.

Let \( t_{pq} \) denote the time that the event \( (x_{p}(t_{pq}) = x_{q}(t_{pq})) \) occurs, which indicates one particle passes another in the one-dimensional system, and hence the order of all particles’ coordinates changes. There are at most \( O(n^2) \) different events, since \( \binom{n}{2} = O(n^2) \). Let \( T_{c} = (t_{1}, t_{2}, \ldots, t_{m}) \) be a sorted vector of all \( t_{pq} \) \((1 \leq p < q \leq m)\), such that, \( t_{i} \leq t_{i+1} \) \((1 \leq i < n)\). Given an initial order at time 0, as time increases, the order of \( n \) particles’ coordinates changes if and only if one event occurs. An important property is that during each time interval \( (t_{i}, t_{i+1}) \) \((1 \leq i < n)\), the total order of \( n \) particles’ coordinates does not change, because no particle passes any other particles. Therefore, we have at most \( O(n^2) \) different coordinates orders. As we have already stated previously, with fixed order and \( k \), picking the \( k \) largest coordinates always gives us \( L_{\text{max}} \). Hence, altogether, there are \( O(n^3) \) different possibilities, \( O(n^2) \)

\(^1\) E.g., sort \( \mathcal{A} \) by decreasing order of \( \frac{a}{\beta} \), then pick the first \( k \) nodes. Or, first pick the largest \( \frac{a}{\beta} \), then pick the next node to make the result as large as possible, and recursively do this. The example below will make the above two heuristics fail.

\( \mathcal{A} = \{(10, 7), (2, 3), (1, 2), (0, 2, 1.34)\} \).
orders × \Omega(n) different k). An (n = 4, k = 2) example is shown in Figure 1.

Based on the above analysis, we provide an \Omega(n^3 \text{Ign}) off-line pre-processing algorithm to profile all possible possibilities (status), and an \Omega(\text{Ign}) online algorithm which determines the optimal subset of servers to keep on.

Let data be an array of pairs that represents \{x\}. Each element data[i] has two data fields, \((a, b)\), which store \(a_i\) and \(b_i\) respectively. Let Event be a class that contains three data fields, \((t_{pq}, p, q)\), meaning particle \(p\) passes particle \(q\) at time \(t_{pq}\), where \(p < q\). Order is a class that stores the ordered node id in an array with respect to particles’ coordinates and supports deep clone by method \(\text{Order}::\text{clone}()\). Let Status denote a class that has four data fields, \((t_{pq}, k, L_{\max}, P^b)\) which are corresponding to time, the number of nodes considered, maximum supported load with given \(t_{pq}\) and \(k\), and power budget. (Parameter \(P^b\) is enclosed to simplify the explanation below. The algorithm itself does not make use of \(P^b\).) The off-line pre-processing algorithm is described in Algorithm (1).

The For loops starting at line 2, and line 14 take \(O(n^2)\) time, since we have only \(O(n^2)\) different events. The For loop starting at line 18 takes \(O(n^3)\) time, since for each event, there is a total order of \(n\) elements. The most time consuming operation is sorting \(\text{allStatus} (O(n^3)\text{ elements})\) on line 27, which takes \(O(n^3 \text{Ign})\) time. As an off-line algorithm, it is efficient enough. So, altogether, the computational complexity and space complexity of Algorithm (1) are \(O(n^3 \text{Ign})\), and \(O(n^3)\) respectively. Algorithm (1) gives us maximum supported load (\(L_{\max}\)) under all possible status, and the total orders (Orders) of \(n\) particles right after each event. Then, given an \(L\), the optimal consolidation policy can be retrieved in \(O(\text{Ign})\) time as described in Algorithm (2).

![Figure 1](image_url)

**Algorithm 1: Pre-process algorithm**

Input: data: the array contains all \((a_i, b_i)\) pairs
Output: Orders: total orders right after each event, indexed by event time; allStatus: coordinates sum of all possible combinations of different events and node number \(k\)

1. \(\text{Events} \leftarrow \text{new Vector} \langle \text{Event} \rangle ()\)
2. for \(i \leftarrow 1 \sim n\) do
3.     for \(j \leftarrow i + 1 \sim n\) do
4.         \(\text{passTime} \leftarrow \frac{a_i - a_j}{b_i - b_j}\)
5.         if \(\text{passTime} \leq 0\) then
6.             continue
7.         \(\text{event} \leftarrow \text{new Event(}\text{passTime} + \varepsilon, i, j\)\)
8.         \(\text{Events.push_back(event)}\)
9. Sort \text{Events} by Events[:\text{pq}]

10. curOrder \leftarrow \text{new Order}([1, 2, ..., n])
11. Sort curOrder by decreasing order of \text{data[curOrder][]}.a

12. Orders \leftarrow \text{new Hashtable} < \text{Time, Order} ()
13. Orders.insert(0, curOrder.clone())
14. for \(i \leftarrow 1 \sim \text{Events.size()}\) do
15.     curOrder.swap(Events[i].p, Events[i].q)
16.     Orders.insert(Events[i].t, curOrder.clone())

17. allStatus \leftarrow \text{new Vector} \langle \text{Status} \rangle ()
18. for each key \(\text{tmpOrder in Orders} \) do
19.     \text{tmpOrder} \leftarrow \text{Orders.get_value}(t)
20.     \(L_{\max} \leftarrow 0\)
21. for \(i \leftarrow 1 \sim \text{tmpOrder.size()}\) do
22.     \(\text{index} \leftarrow \text{tmpOrder}[i]\)
23.     \(L_{\max} \leftarrow L_{\max} + \text{data[index].a} - t_{pq} \times \text{data[index].b}\)
24.     \(P^b \leftarrow \text{if} \_\text{w2} - P_{pq} + \theta\)
25.     \(\text{status} \leftarrow \text{new Status}(t_{pq}, L_{\max}, i, P^b)\)
26.     allStatus.push_back(status)

27. Sort allStatus by increasing order of \(L_{\max}\)

![Algorithm 1 Diagram](image_url)

**Algorithm 2: Online load consolidation**

Input: Load \(L\), allStatus, Orders
Output: ON: servers to turn on which guarantees optimality

1. Binary search on allStatus[\cdot].\(L_{\max}\) to find the minimum \(L_{\max}\), s.t., \(\text{opt.}L_{\max} > L\)
2. \(\text{optOrder} \leftarrow \text{Orders.get_value} (\text{opt}\_t_{pq})\)
3. \(\text{ON} \leftarrow \text{optOrder.clone}(0, \text{opt.k})\)
4. Return ON

![Algorithm 2 Diagram](image_url)
The basic idea is to perform binary search on allStatus to find the first element allStatus[i] whose L_{max} is greater than L. According to the monotonicity between L_{max} and P^b, opt.P^b is the minimum possible power budget, and the corresponding set of servers form the optimal solution. Line 2 \sim 5 simply retrieve the solution set from Orders.

IV. EXPERIMENTAL EVALUATION

We implemented our solution on a testbed of one rack of 20 Dell Power Edge R210 machines. The computer rack is located in a machine room at the Department of Computer Science, with a controllable cooling system (Liebert Challenger 3000 with linecard) whose controls are given to the authors. The cooling unit supplies cool air from the ceiling of the room. We use Watts up Pro power meters to measure the power consumption of servers.

In this section, we evaluate how good the proposed holistic algorithm is at saving energy compared to other heuristics described in previous literature. Using the 20-machine testbed, we begin by system profiling. The purpose of profiling is to estimate the coefficients used in our modeling equations. These coefficients are then fed into the optimization problem to derive actual AC temperature and load distribution.

A. Profiling

In the model used in our simplified optimization problem formulation, both power consumption and stable computer temperature were linearly dependent on other parameters such as workload and cooling setting point. Hence, our first task is to quantify these dependencies empirically. The involved coefficients can be computed via off-the-shelf linear regression. In our work, we use least squares fitting technique for coefficient estimation.

Profiling the Power Consumption Model: The power consumption coefficients are the same for all machines in our testbed, as they have the same hardware configuration and are running the same software. In our experiments, for workload, we ran a text processing application, resembling data mining applications that currently are one of the prime examples of computations for which large processing power is needed. Our application took html files as input, extracted meaningful text, then produced a word histogram for that text. The capacity of a machine (the maximum number of html files that a machine could process on average per second) was measured before the experiment. We then ran a profiling experiment where load was changed from 0% to 10%, 25%, 50%, and 75% of the capacity computed above. The machines stayed at each load level for 15 minutes and power consumption (in Watts) was measured every second. Before changing the load each time, the machines were left idle (0% load) for a short period of time. The above load and power measurements were then used as input to the regression algorithm that computed the \( w_1 \) and \( w_2 \) coefficients of Equation (9).

Figure 2 compared measured power to predicted values from the computed linear regression model. It can be seen that the model is quite accurate. In the figure, measured data is smoothed by a lower-pass filter to eliminate noise.

![Figure 2. Measured Power Consumption and Predicted Power Consumption Comparison](image)

Profiling Stable CPU Temperature Model: The same experiment was done to estimate the coefficients of the CPU’s stable temperature model, described by Equation (8). Notice that, the thermal model coefficients are different among machines. This is due to the difference in the relative position of machines on our rack (which causes the cool intake air stream to be distributed to each machine differently). The experiments were done with different cooling set-points.

In each experiment, the CPU temperature of a server was measured as load was varied. At each level of load, the server was running until a stable CPU temperature was reached (in about 200 seconds). We used linear regression to estimate the coefficients of Equation 8 for each server. Figure 3 shows an example of the predicted stable temperature (using our linear model) versus the measured temperature (by lm-sensors) for one of the servers (the measured temperature was smoothed using a low-pass filter to eliminate noise). While not perfect, the linear model was able to predict (with a few percent error) the stable temperature of server’s CPU given load and AC’s cool air temperature.

B. Performance Evaluation

In this section, we report evaluation of the proposed method with two different baseline approaches in different settings. The two baselines that we consider are as follows:

- **Even**: In this approach, total load is distributed evenly among all machines in the cluster. This is the standard load balancing practice.
- **Cool load allocation** [1]: This is the state of the art method of cooling-sensitive load allocation. It advocates filling machines up, coolest first. The assumption
is that machines in cooler spots must require less energy
to cool down than machines in warmer spots. Applying
this approach to our testbed, the machines at the bottom
of the rack are assigned first (since they are in a cooler
spot. The process continues until all load is assigned.
Hence, we call it bottom-up, where brevity is needed.

In general, an energy control policy can decide on the
following parameters:

- **Load Distribution**: The distribution of load across
  machines was a direct output of the optimization algo-
  rithm.
- **AC’s Temperature**: The solution chooses AC’s tem-
  perature to further save energy. To set $T_{ac}$, we empiri-
  cally measured the relation between $T_{ac}$ and the set point,
  $T_{SP}$ for the optimal load distribution at different server
  loads. We would then choose the set point that produces
  the needed $T_{ac}$ given the load at hand.
- **Consolidation**: When consolidation was used, servers
  which were not utilized, were turned off to save energy.

We thus can come up with eight evaluation scenarios for
comparison as shown in Figure 4. Below, each solution will
be referred to by its number (as shown in the Figure 4).
For example, in method #5, load is distributed bottom-up
according to the cool load allocation approach [1]), then the
AC temperature is set as high as it could be without violating
temperature constraints.

In solutions without AC control, the AC temperature set-
ing was chosen as the highest temperature that (empirically)
and temperature measurements [9], [10]. Prior work also

**Load Allocation with Consolidation**: Next, we consider
the effect of the optimal load allocation on the total energy
consumption in systems which do use consolidation. We see
that with optimal load allocation, 5% saving in total energy
consumption is possible. Figure 8 details energy savings for
a varied total load. The energy savings under the optimal
load allocation were relatively consistent for different loads
in this setting.

While not shown explicitly in the above evaluation, we
also verified that the temperature constraints, $T_{max}$, were
not violated for any of the CPUs, when using the proposed
optimization algorithm. All CPU temperatures were indeed
below the maximum. Finally, the presented model and
the problem formulation take the total throughput (load,
$L$) as another constraint. We also verified that application
throughput was not affected by the energy saving scheme,
thus satisfying the load constraint. Due the space constraint,
we omit the related temperature and throughput plots.

In summary, a holistic solution which jointly and opti-
mally tunes our control knobs on both the cooling side (AC
temperature) and the computing side (machine load distribu-
tion and consolidation) yields the most energy savings. The
above results show that our solution saves 7% of the total
energy consumption on average over all load scenarios and
is able save up to 18% in the best case compared to the next
best baseline, method #7, that uses the cool job allocation
scheme [1]. Figure 9 and 10 summarize these results.

**V. RELATED WORK**

This paper is the first to suggest an optimal closed-form
solution to the problem of joint optimization of cooling and
computing energy in data centers subject to load and CPU
temperature constraints.

The topic of energy optimization in server farms received
a lot of attention. Several prior efforts addressed energy
and temperature measurements [9], [10]. Prior work also
described models of individual computing units to determine the factors affecting energy consumption [11], [12]. While this paper considers a relatively simple power consumption model (that is shown to be adequate using empirical evaluation), future extensions can delve into details such as separating CPU and memory consumption. Indeed, this separation will be necessary for efforts that extend the current work to machines with DVS capabilities. When the CPU runs at the same frequency, lumping all power consumption into one load-dependent and one load-independent component appears to be adequate.

Unlike the plethora of work on uni-processor and multiprocessors [13], where DVFS was a key knob, in large data centers, energy proportionality (making energy consumption proportional to load) can be easily achieved simply by turning off the right number of machines [14], [15]. The increasing percentage of leakage energy in modern architectures makes it less economic to keep machines on, even at the lowest frequency. Hence, in this paper, we did not address DVFS.

A significant portion of prior work on energy optimization minimized energy of either computing side or cooling side alone [6], [7], [16], [17], [18], [19], [20]. In previous work [21], the authors argued that management of energy knobs in data centers must be done holistically in order to avoid inefficiencies. Such inefficiencies were also occur due to “power struggles” [22] when computing and cooling energy are optimized separately, without taking a holistic “cyber-physical” view.

In order to avoid inefficiencies, the problem of joint optimization of computing and cooling was recently investigated. A formulation somewhat similar to ours was addressed in the context of thermal-aware scheduling [2]. It was cast as a control problem with the purpose of maintaining temperature of racks around an operating point, as opposed to an optimization problem. It was formulated at the rack level, making it much coarser than ours. For example, it would stop at trivially assigning all load to the same rack when only one rack is present. In contrast, we addressed load distribution at the machine level (as well as selection of those machines to power on) within or across racks. A similar problem to ours was solved within a single blade server enclosure [3]. However, no closed-form results where obtained. We could not compare our results to that approach due to differences in control knobs that exist in their enclosure, versus those applicable to our machine rack. The closest comparable work is the cool job allocation [1] that addresses combined computing and cooling energy minimization at the machine level in data centers. It suggests loading machines up coolest first. Load distribution is heuristic in nature, as opposed to our result, which presents a closed-form optimal solution and
a guaranteed optimal load consolidation algorithm under our model. A variation of the cool job allocation scheme defines utility for allocating load to machines depending on how easy or difficult it is to cool them [23]. Our presented experimental evaluation shows that our approach indeed saves more energy by optimally solving the load distribution problem.

Some researchers [4], [5], [24] provided solutions to tackle the energy issue by maximizing the capacity of a cluster with a given power budget. Gandhi et al. [24] compared the response time of different configurations of the same cluster. They provided several criteria that help people to decide which configuration to use upon a given environment. Literature [4] utilizes convex optimization to maximize the throughput of one computationally-intensive map-reduce cluster by taking CPU frequency and CPU temperature into consideration. But the solution only applies to very specific scenarios since the simplification of their optimization is based on properties of map-reduce, and cooling energy is not accounted. Admittedly, maximizing capacity with a given power budget helps to minimizing response time when the cluster is saturated. However, in most cases, the saturation will not happen [25]. Hence, we argue that minimizing energy consumption with given load has more practical significance.

Finally, our emphasis in this paper is on the cyber-physical energy optimization problem. We did not address time constraints. One can imagine other workloads where real-time constraints are present, making more complex solutions necessary. Such investigation is beyond the scope of this paper.

VI. CONCLUSIONS

This paper presented total energy optimization in a machine room, subject to load and CPU temperature constraints. The energy expended in the room is the sum of computing energy of servers and cooling energy of the AC. It was shown that holistic analysis is needed that jointly considers both subsystems in order to minimize total energy consumption. An optimal load distribution, based on holistic energy minimization, was presented and shown experimentally outperforming baseline heuristics including a state of the art approach. It is expected that more savings can be achieved in larger-scale systems, which opens the door for more advanced problem formulations to optimize performance of computing systems that interact in the energy domain.

ACKNOWLEDGMENT

This work was supported in part by NSF grants CNS 09-16028, CNS 09-58314, CNS 10-35736, and CNS 1040380, as well as by the LCCC and eLLIIT centers at Lund University, Sweden.

REFERENCES