Service Level Agreement-Based Joint Application Environment Assignment and Resource Allocation in Cloud Computing Systems

Yanzhi Wang, Shuang Chen and Massoud Pedram
Department of Electrical Engineering
University of Southern California
Los Angeles, USA
{yanzhiwa, shuangc, pedram}@usc.edu

***Abstract*—Cloud computing have attracted a lot of attention recently due to increasing demand for high performance computing and storage. Resource allocation is one of the most important challenges in the cloud computing system especially when the clients have some Service Level Agreements (SLAs) and the total profit depends on how the system can meet these SLAs. Moreover, a data center typically hosts and manages a suite of application environments and a fixed number of servers that are allocated to these application environments in a way that maximizes a certain utility function. In this paper, we consider the problem of SLA-based joint optimization of application environment assignment, request dispatching from the clients to the servers, as well as resource allocation in a data center comprised of heterogeneous servers. The objective is to maximize the total profit, which is the total price gained from serving the clients subtracted by the operation cost of the data center. The total price depends on the average service request response time for each client as defined in their utility functions, while the operating cost is related to the total energy consumption. We propose a near-optimal solution of the joint optimization problem based on the Hungarian algorithm for the assignment problem, as well as convex optimization techniques, in a way that is similar to the constructive partitioning algorithm in VLSI computer-aided design (CAD). Experimental results demonstrate that the proposed near-optimal joint application environment assignment and resource allocation algorithm outperforms baseline algorithms by up to 65.7%.**

Keywords-cloud computing; application environment; resource allocation; assignment problem

I. INTRODUCTION

Cloud computing has been envisioned as the next-generation computing paradigm for its advantages in on-demand service, ubiquitous network access, location independent resource pooling, and transference of risk [1]. Cloud computing shifts the computation and storage resources from the network edges to a "Cloud" from which businesses and users are able to access applications from anywhere in the world on demand [2][3][4]. In cloud computing, the capabilities of business applications are exposed as sophisticated services that can be accessed over a network. Cloud service providers are incentivized by the profits by charging the clients for accessing these services. Clients are attracted by the opportunity for reducing or eliminating the costs associated with "in-house" provision of these services. It is essential that the clients have guarantees from service providers on service delivery. Typically, these are provided through Service Level Agreements (SLAs) brokered between the providers and consumers. The SLAs include computing power, storage space, network bandwidth, availability and security, etc.

The underlying infrastructure of cloud computing consists of data centers and clusters of servers that are monitored and maintained by the cloud service providers [6]. Service providers often end up over-provisioning their resources in these servers in order to meet the clients' SLAs [5]. Such over-provisioning may increase the cost incurred on the servers in terms of both the electrical energy cost and the carbon emission. Therefore, optimal provisioning or allocation of the resources is imperative in order to reduce the cost incurred on the servers as well as the environmental impact while satisfying the clients' SLAs. The problem of optimal resource allocation in the cloud computing framework for serving the service requests of each client is therefore crucial and has been investigated in [7][8][9][10]. The more general problem of resource allocation and management in distributed computing system has been an active research topic in the recent ten years. There is a number of papers discussing the resource allocation problem in grid computing systems [11][12], in the framework of electronic commerce [13], in autonomic computing systems [14][15], in clusters of servers [16], and in hosting centers [17].

A data center typically hosts and manages a suite of complex *Application Environments* with diverse requirements and dynamic characteristics. For example, some Web applications experience highly bursty traffic whose workload intensity varies dramatically during different time of day, or day of week. The data center also has a fixed number of servers that are (dynamically) allocated to these various application environments in a way that maximizes a certain utility function, as discussed in [20]. References [18][19][20][21] introduce the *application environment assignment* problem, i.e., determining which subset of the available servers should be allocated to run

*This research is sponsored in part by the Software and Hardware Foundations program of the NSF's Directorate for Computer & Information Science & Engineering.

978-0-7695-4966-8/13 $26.00 © 2013 IEEE
DOI 10.1109/GreenTech.2013.33

each application environment, where each server can run at most one application environment. However, they suffer from either the scalability problem, or the lack of exploiting the opportunity of joint application environment assignment and resource allocation to maximize the total profit. The latter opportunity is especially important since the results of application environment assignment and resource allocation affect each other in an interactive manner.

In this paper, we consider the problem of SLA-based joint application environment assignment and resource allocation optimization in a data center comprised of heterogeneous servers. Multiple clients exist in this framework, each generating service requests in a different rate. A client runs one or multiple types of application software, which require processing, data storage, and communication resources in the cloud computing system. Each client in this system has a pre-defined *utility function* based on its response time requirements. The data center consists of multiple potentially heterogeneous servers. The total profit in this cloud computing system is the total price gained from serving the clients subtracted by the cost of operating the turned on servers in the system, where the operating cost of turned on servers is proportional to their energy consumptions.

Different from the prior work, we propose a joint optimization framework considering the optimal application environment assignment, request dispatching from the clients to the servers, as well as the optimal resource allocation in the servers. We propose a near-optimal solution of this joint optimization problem, which is motivated by the VLSI computer-aided design (CAD) algorithm for constructive partitioning (partitioning by clustering) of circuit netlists [28][29]. We define and compute an *affinity* value for each application environment-server pair based on the resource allocation optimization results, where a higher affinity value indicates that it is more preferable to assign the corresponding application environment to the server. We allocate a subset of servers based on the affinity values, one for each application environment. We recalculate the affinity values and proceed with this assignment procedure. The near-optimal solution is based on the Hungarian algorithm for the assignment problem [33], as well as convex optimization techniques [31][32]. Experimental results demonstrate that the proposed near-optimal joint application environment assignment and resource allocation algorithm outperforms baseline algorithms by up to 65.7%.

The rest of this paper is organized as follows. The system model for joint application environment assignment and resource allocation in the cloud computing system is introduced in Section II. The optimization problem formulation is provided in Section III. The proposed near-optimal algorithm is provided in Section IV. Experimental results and conclusion are presented in Section V and Section VI, respectively.

II. System Model and Problem Formulation

Figure 1 shows the structure of the target cloud computing resource allocation system with a set of N clients, a data center, as well as a central resource management node. The data center hosts M application environments and has K potentially heterogeneous servers that should be allocated to run the application environments. The central manager has information about the data center as well as the clients.

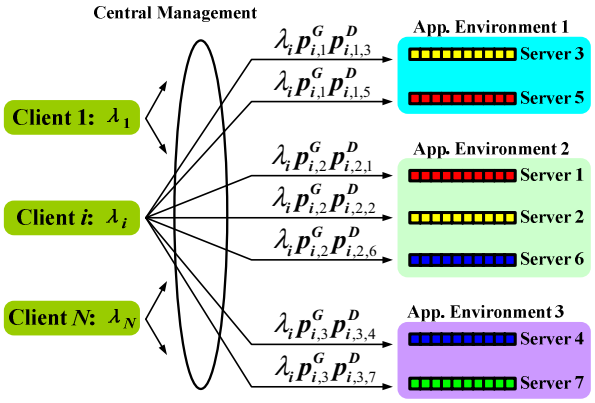

Figure 1. Conceptual structure of the application environment assignment and resource allocation system in cloud computing, with $M = 3$ and $K = 7$.

Each client in the system is identified by a unique ID, represented by index i. Each application environment in the data center is represented by index j. Similarly, each server in the data center is captured by index k. Each server can be allocated by the central manager to run at most one application environment. An application environment can be assigned to multiple servers. Let $AE(k)$ denote the application environment assigned to the k-th server in the data center. Let $\mathbf{SS}(j)$ denote the set of servers allocated to run the j-th application environment.

A client runs one or more types of applications and generates service requests to be served in the data center. Let $\mathbf{APP}(i)$ denote the set of applications running in the i-th client. In order to find the analytical form of the response time, the service requests generated from each i-th client are assumed to follow a Poisson process with an average generating rate of λ_i (predicted based on the behavior of the client.) We assume that a portion $p^{\mathbf{G}}_{i,j}$ of these service requests is generated from the j-th application, where the superscript $\mathbf{G}$ stands for 'Generating'. We have $p^{\mathbf{G}}_{i,j} = 0$ if the j-th application is not running in the i-th client, i.e., $j \notin \mathbf{APP}(i)$. Then according to the properties of the Poisson process, the service requests that are generated from the j-th application of the i-th client follow a Poisson process with an average rate of $\lambda_{i,j} = p^{\mathbf{G}}_{i,j} \cdot \lambda_i$ [24].

Service requests generated by a single application in a single client can be assigned to multiple servers in the data

center, as long as these servers are allocated to run such application environment. The request dispatcher assigns a request generated from the *j*-th application of the *i*-th client to the *k*-th server in the data center with probability $p^{\mathbf{D}}_{i,j,k}$, where the superscript **D** stands for 'Dispatching'. Please note that we have $p^{\mathbf{D}}_{i,j,k} = 0$ as long as $j \neq AE(k)$. These $p^{\mathbf{D}}_{i,j,k}$ probability values are the optimization variables in the resource allocation optimization framework. According to the properties of the Poisson distribution [24], service requests that are generated from the *j*-th application of the *i*-th client and dispatched to the *k*-th server in the data center follow a Poisson process with an average rate of $\lambda_{i,j} \cdot p^{\mathbf{D}}_{i,j,k}$. As long as a service request is dispatched to a server, the server creates a dedicated virtual machine (VM) for that service request, loads the application executable and starts execution [27].

To model the multi-class queues in the cloud computing system, the Generalized Processor Sharing model [25][26] is used. It has been shown that the GPS can be implemented by weighted fair queueing if the service times of packets are not too large. Let $\phi_{i,k}$ denote the portion of computation resources of the *k*-th server that is allocated for the *i*-th client.

All multi-class single server queues can be replaced by single class single server queues using the GPS model. To model the response time of the service requests in the cloud computing system, by using the well-known formula in M/M/1 queues [30], the average response time of the service requests generated from the *i*-th client and dispatched to the *k*-th server (i.e., the application type $j = AE(k)$) is given by:

$$R_{i,k}\left(p^{\mathbf{D}}_{i,AE(k),k}; \phi_{i,k}\right) = \begin{cases} \dfrac{1}{\mu_{AE(k),k} \cdot \phi_{i,k} - p^{\mathbf{D}}_{i,AE(k),k} \cdot \lambda_{i,AE(k)}}, & \text{if } p^{\mathbf{D}}_{i,AE(k),k} > 0 \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

for $\lambda_{i,AE(k)} > 0$ (i.e., $AE(k) \in \mathbf{APP}(i)$). In Eqn. (1), $\mu_{AE(k),k}$ denotes the average processing speed of the *k*-th server in processing service requests from application type $AE(k)$, when all the resources in the server has been allocated. The overall average response time of client *i* can be calculated as:

$$R_i = \sum_{AE(k) \in \mathbf{APP}(i)} p^{\mathbf{G}}_{i,AE(k)} \cdot p^{\mathbf{D}}_{i,AE(k),k} \cdot R_{i,k}\left(p^{\mathbf{D}}_{i,AE(k),k}; \phi_{i,k}\right) \quad (2)$$

The power consumption of each *k*-th server if it is turned ON is modeled as a constant term plus another component linearly related to the utilization of the server in the processing domain, given by:

$$P_{k,const} + P_{k,lin} \cdot \sum_{i=1}^{N} \phi_{i,k} \quad (3)$$

The objective of the joint application environment assignment and resource management problem is to maximize the total profit of the data center from serving the clients. In this system, decision making intervals can be defined based on the behavior of the dynamic parameters in the system. This is because the solution found by the presented algorithm is acceptable only when the parameters used to find the solution are approximately valid. Although some small changes in the parameters can be effectively tracked and responded to by proper reactions of the central resource manager in the data center, large changes cannot be handled in this way. In the remainder of this paper, the joint application environment assignment and resource allocation problem at each decision epoch is presented and a solution is provided, but we do not discuss the estimation, prediction, and dynamic changes in the system because these issues are out of the scope of this paper.

III. Optimization Problem Fomulation

Let $U_i(R)$ denote the non-increasing utility function of the *i*-th client with the average service request response time equal to R. In the rest of this paper, we use a linear-form decreasing utility function in the optimization problem, i.e., $U_i(R) = \beta_i - \alpha_i \cdot R$, similar to the utility function exploited in [7][8]. Let x_k denote the pseudo-Boolean integer to represent if the *k*-th server is turned ON ($x_k = 1$) or OFF ($x_k = 0$). $AE(k)$ and $\phi_{i,k}$ can be arbitrary value if $x_k = 0$, i.e., we are only interested in the $AE(k)$ and $\phi_{i,k}$ values for $x_k = 1$. In this optimization problem, x_k, $AE(k)$, $p^{\mathbf{D}}_{i,AE(k),k}$, and $\phi_{i,k}$ values are the optimization variables. The other parameters are either constants or functions of these optimization variables.

The overall joint optimization problem of application environment assignment, request dispatching, and resource allocation for the data center is formulated as a profit maximization problem as below:

Find the optimal x_k's, $AE(k)$'s, $p^{\mathbf{D}}_{i,AE(k),k}$'s, and $\phi_{i,k}$'s.

Maximize:

$$\sum_{i=1}^{N} \lambda_i \cdot (\beta_i - \alpha_i \cdot R_i) - Price \cdot \sum_{k=1}^{K} x_k \cdot \left(P_{k,const} + P_{k,lin} \cdot \sum_{i=1}^{N} \phi_{i,k}\right) \quad (4)$$

Subject to:

$$x_k \in \{0,1\}, \text{ for } \forall k \in \{1,2,\dots,K\}, \quad (5)$$

$$0 \leq p^{\mathbf{D}}_{i,AE(k),k} \leq 1, \text{ for } \forall i,k, \quad (6)$$

$$0 \leq \phi_{i,k} \leq 1, \text{ for } \forall i,k, \quad (7)$$

$$p^{\mathbf{D}}_{i,AE(k),k} \cdot \lambda_{i,AE(k)} \leq \mu_{AE(k),k} \cdot \phi_{i,k}, \text{ for } \forall i,k, \quad (8)$$

$$\sum_{k:AE(k)=j} p^{\mathbf{D}}_{i,j,k} = 1, \text{ for } \forall i,j, \quad (9)$$

$$\sum_{i=1}^{N} \phi_{i,k} \le 1, \quad \text{for } \forall k \in \{1,2,\dots,K\}, \tag{10}$$

$$x_k \ge \sum_{i=1}^{N} \phi_{i,k}, \quad \text{for } \forall k, \tag{11}$$

where $Price$ is the *unit energy price* at this decision epoch.

In the objective function (4), the first term is the total price gained from serving all the service requests, and the second term is the total energy cost for operating the data center. Constraints (5) - (7) specify the domains of the optimization variables. Constraint (8) shows a lower limit on the resource allocation in each server. Constraint (9) ensures that all requests generated by a client are served. Constraint (10) limits the total amount of allocated resource in each server. Constraint (11) determines the set of turned ON servers based on the allocated resources.

The overall joint application environment assignment and resource allocation problem is a mixed integer nonlinear programming problem. The problem cannot be solved using the conventional convex optimization methods [31][32] since the objective function (4) is neither convex nor concave even if the optimal values of the integer variables x_k's and $AE(k)$'s are given in prior, i.e., the set of servers that are turned ON and the application environment assignment results are given in prior.

IV. Optimization Methods

The joint application environment assignment and resource allocation problem presented in the previous section is a hard problem due to the non-convexity of the objective function as well as the existence of integer variables x_k's and $AE(k)$'s. The simple problem solvers cannot solve this problem except in the case of very small input size by running exhaustive search or by using stochastic optimization methods such as the Simulated Annealing or Genetic Algorithm. In this section, a near-optimal solution is presented for this problem.

The proposed near-optimal solution is motivated by the algorithm of constructive partitioning, i.e., partitioning by clustering, of circuit netlists in VLSI CAD [28][29]. More specifically, the near-optimal solution consists of two steps. In the **first step**, we perform effective application environment assignment and find the near-optimal integer values x_k and $AE(k)$. We define and compute an affinity value $Aff(j,k)$ for each application *j* and server *k*, where a higher affinity value indicates that it is more preferable to assign the corresponding *j*-th application environment to the *k*-th server. We allocate a subset of *M* servers based on the affinity values, one for each application environment. We recalculate these affinity values for the remaining servers and proceed with this assignment procedure. The first step is based on the Hungarian algorithm for the assignment problem [33], as well as convex optimization techniques [31][32]. In the **second step**, we perform effective service request dispatching and resource allocation, i.e., finding the near-optimal $p^{\mathbf{D}}_{i,AE(k),k}$ and $\phi_{i,k}$ values, based on the results of application environment assignment. The second step is based on iterative optimization and convex programming techniques. We will elaborate the details of the near-optimal algorithm in the following three subsections.

A. Calculating the Affinity Values

We introduce in the following the proposed method to calculate the affinity values $Aff(j,k)$'s. A higher affinity value $Aff(j,k)$ indicates that it is more preferable to assign the *j*-th application environment to the *k*-th server. We formally present the calculation procedure of the affinity values as follows.

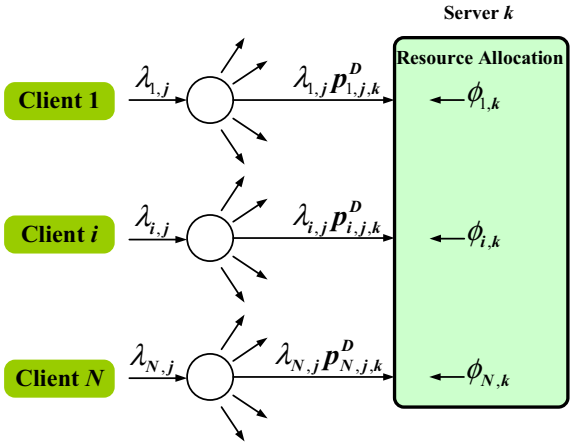

Figure 2. Conceptual structure illustrating the calculation of affinity value $Aff(j,k)$.

In order to calculate the $Aff(j,k)$ value, we only consider dispatching service requests of the *j*-th application environment to the *k*-th server. As illustrated in Figure 2, the service requests of the *j*-th application environment are generated from the 1st, 2nd, …., *N*-th clients with average rate of $\lambda_{1,j}$, $\lambda_{2,j}$, … , and $\lambda_{N,j}$, respectively. We dispatch a portion $p^{\mathbf{D}}_{i,j,k}$ of such service requests from each *i*-th client to the *k*-th server. We allocate a portion $\phi_{i,k}$ of resources in the *k*-th server to serve the requests generated from the *i*-th client. The objective is to maximize the ratio of the total price gained by the *k*-th server from serving the requests to the energy cost, which is given by:

$$\frac{\sum_{i:j\in \mathbf{APP}(i)} \lambda_{i,j} \cdot p^{\mathbf{D}}_{i,j,k} \cdot \left(\beta_i - \alpha_i \cdot R_{i,k}\left(p^{\mathbf{D}}_{i,j,k}; \phi_{i,k}\right)\right)}{Price \cdot \left(P_{k,const} + P_{k,lin} \cdot \sum_{i:j\in \mathbf{APP}(i)} \phi_{i,k}\right)} \tag{12}$$

The optimization variables are $p^{\mathbf{D}}_{i,j,k}$'s and $\phi_{i,k}$'s. The constraints are given as follows:

$$0 \le p^{\mathbf{D}}_{i,j,k} \le 1, \text{ for } \forall i, \quad (13)$$

$$0 \le \phi_{i,k} \le 1, \text{ for } \forall i, \quad (14)$$

$$p^{\mathbf{D}}_{i,j,k} \cdot \lambda_{i,j} \le \mu_{j,k} \cdot \phi_{i,k}, \text{ for } \forall i, \quad (15)$$

$$\sum_{i:j \in \mathbf{APP}(i)} \phi_{i,k} \le 1, \quad (16)$$

We name this optimization problem the *Maximum Ratio Optimization* (MRO) problem. Suppose that the MRO problem has been optimally solved. Let $p^{\mathbf{D}*}_{i,j,k}$'s and $\phi^*_{i,k}$'s denote the optimal values of the optimization variables in the MRO problem. Then the affinity value $Aff(j,k)$ is defined as the corresponding total profit, which is given by:

$$Aff(j,k) = \sum_{i:j \in \mathbf{APP}(i)} \lambda_{i,j} \cdot p^{\mathbf{D}*}_{i,j,k} \left(\beta_i - \alpha_i R_{i,k}\left(p^{\mathbf{D}*}_{i,j,k}; \phi^*_{i,k}\right)\right) - Price \cdot \left(P_{k,const} + P_{k,lin} \cdot \sum_{i:j \in \mathbf{APP}(i)} \phi^*_{i,k}\right) \quad (17)$$

We will provide the solution of the MRO problem as follows.

The MRO problem is a non-convex optimization problem with continuous optimization variables, because the objective function (12) is neither convex nor concave with respect to the optimization variables $p^{\mathbf{D}}_{i,j,k}$'s and $\phi_{i,k}$'s. It cannot be solved optimally using conventional convex optimization techniques. We propose an iterative near-optimal solution of the MRO problem and calculation of $Aff(j,k)$ as shown in Algorithm 1. In each iteration, the near-optimal solution consists of an *optimal resource allocation phase* and an *optimal request dispatching phase* as follows.

The Optimal Resource Allocation Phase: In this phase, the controller finds the optimal $\phi_{i,k}$ values in order to maximize the objective function (12) when the $p^{\mathbf{D}}_{i,j,k}$ values are given. The constraints are (14), (15), and (16). This optimization problem is a quasi-convex optimization problem [31] since (i) the nominator of the objective function (12) to maximize is a concave function of $\phi_{i,k}$'s when the $p^{\mathbf{D}}_{i,j,k}$ values are given, whereas the denominator of (12) is a linear function of $\phi_{i,k}$'s; (ii) the constraints (14), (15), and (16) are linear inequality constraints. Such quasi-convex optimization problem can be transformed into a set of standard convex optimization problem and solved optimally within polynomial time complexity [31].

The Optimal Request Dispatching Phase: In this phase, the controller finds the optimal $p^{\mathbf{D}}_{i,j,k}$ values so as to maximize the objective function (12) when the $\phi_{i,k}$ values are given. The constraints are (13) and (15). This problem is also a convex optimization problem since (i) the objective function (12) is a concave function of $p^{\mathbf{D}}_{i,j,k}$'s when the $\phi_{i,k}$ values are given, and (ii) the constraints (13) and (15) are linear inequality constraints. Therefore, it could be solved optimally with polynomial time complexity using standard convex optimization techniques [31][32].

Algorithm 1: Near-Optimal Solution of the MRO Problem and Calculating the affinity value $Aff(j,k)$.

Initialize the $p^{\mathbf{D}}_{i,j,k}$ values.

Do the following procedure iteratively:

> Optimal resource allocation: Find the optimal $\phi_{i,k}$ values that maximize the objective function (12) based on the derived $p^{\mathbf{D}}_{i,j,k}$ values, using quasi-convex optimization methods.
>
> Optimal request dispatching: Find the optimal $p^{\mathbf{D}}_{i,j,k}$ values that maximize the objective function (12) based on the derived $\phi_{i,k}$ values, using convex optimization methods.

Until the solution converges.

Calculate the $Aff(j,k)$ value using (17) based on the near-optimal solution of the MRO problem.

B. Optimization of Application Environment Assignment

After we have calculated the $Aff(j,k)$ values, we allocate a subset of M servers to execute the application environments, one for each application environment. Essentially, we find a map $f(j)$ from each application environment j to a server index, such that the following summation of affinity values is maximized:

$$\sum_{j=1}^{M} Aff(j, f(j)) \quad (18)$$

This is the well-known assignment problem with unequal numbers of "agents" and "tasks" [33], since the number of servers is larger than the number of application environments (i.e., $M < K$) in this case. We add dummy nodes to the set of M application environment to make the total number of elements the same as the total number of servers, i.e., K. Then we use standard solution of the assignment problem such as the Hungarian method to optimally solve this problem with polynomial time complexity [33].

After the first round, only M servers have been allocated to run the application environments. We update the affinity values for the application environment-server pairs of the $K - M$ unallocated servers, based on the remaining service requests (without dispatched to the servers) that are generated from each j-th application environment. We again allocate another M servers using the Hungarian algorithm based on the updated affinity values. We proceed until there are no remaining service requests from any application

environment. In this way, we obtain the near-optimal application environment assignment results and the set of turned ON servers, i.e., the $AE(k)$ and x_k values. A brief summary of the proposed near-optimal solution for application environment assignment is provided in Algorithm 2. Implementation details are omitted due to space limitation.

Algorithm 2: Near-Optimal Solution for Application Environment Assignment.

While there are remaining service requests without dispatched to the servers:

> Calculate the $Aff(j,k)$ values for the unallocated servers.
>
> Allocate M servers to run the application environments to maximize the total affinity value, using the Hungarian algorithm.
>
> Set the $AE(k)$ values of the M servers, and set their x_k values to 1 (turned ON.)
>
> Dispatch the corresponding portions of service requests from the application environments to the servers, and calculate the remaining amounts of service requests of each application environment.

End.

C. Optimization of Request Dispatching and Resource Allocation

In the last phase of the overall optimization algorithm, near-optimal values of the integer variables x_k and $AE(k)$ are given from the prior application environment assignment procedure. The objective is to perform near-optimal service request dispatching as well as resource allocation, i.e., finding the near-optimal $p^{\mathbf{D}}_{i,AE(k),k}$ and $\phi_{i,k}$ values, so as to maximize the objective function (4). The constraints are (6) – (10). We name this profit maximization problem the *Resource Allocation and Request Dispatching* (RARD) optimization problem. The optimization variables of the RARD problem are the $p^{\mathbf{D}}_{i,AE(k),k}$ and $\phi_{i,k}$ values, which are all continuous variables.

The RARD problem cannot be solved using conventional convex optimization methods since the objective function (4) is still neither convex nor concave even when the integer variable values x_k's and $AE(k)$'s are given in prior. We propose an iterative near-optimal solution of this optimization problem as shown in Algorithm 3. In each iteration, Algorithm 3 has an *optimal resource allocation phase* as well as an *optimal request dispatching phase* as follows:

The Optimal Resource Allocation Phase: In this phase, the controller finds the optimal $\phi_{i,k}$ values in order to maximize the objective function (4) when the $p^{\mathbf{D}}_{i,AE(k),k}$ values are given. The constraints are (7), (8), and (10). This problem is a convex optimization problem since the objective function (4) is a concave function of $\phi_{i,k}$'s when the $p^{\mathbf{D}}_{i,AE(k),k}$ values are given, and constraints (7), (8), and (10) are linear inequality constraints. It can be solved optimally within polynomial time complexity using standard convex optimization techniques [31][32].

The Optimal Request Dispatching Phase: In this phase, the controller finds the optimal $p^{\mathbf{D}}_{i,AE(k),k}$ values so as to maximize the objective function (4) when the $\phi_{i,k}$ values are given. The constraints are (6), (8), and (9). This problem is also a convex optimization problem since the objective function (4) is a concave function of $p^{\mathbf{D}}_{i,AE(k),k}$'s when the $\phi_{i,k}$ values are given, and therefore, it could be solved optimally with polynomial time complexity using standard convex optimization techniques [31][32].

Algorithm 3: Near-Optimal Solution of the RARD Problem.

Initialize the $p^{\mathbf{D}}_{i,AE(k),k}$ values.

Do the following procedure iteratively:

> Optimal resource allocation: Find the optimal $\phi_{i,k}$ values that maximize the objective function (4) based on the derived $p^{\mathbf{D}}_{i,AE(k),k}$ values.
>
> Optimal request dispatching: Find the optimal $p^{\mathbf{D}}_{i,AE(k),k}$ values that maximize the objective function (4) based on the derived $\phi_{i,k}$ values.

Until the solution converges.

V. Experimental Results

In this section, we implement the joint application assignment and resource allocation optimization framework and compare the proposed near-optimal algorithm with baseline resource allocation algorithms.

We consider a data center of 20 heterogeneous servers (we will change this number later in the experiments.) We consider 4 clients generating service requests from 3 types of application environments in the cloud computing system. We use normalized amounts of most of the parameters in the cloud computing system instead of their real values. The average service request generating rate $\lambda_{i,j}$ from each *j*-th application of each *i*-th client is a uniformly distributed random variable between 1 and 4. The maximum average processing speed of each *k*-th server for processing service requests from application type *j*, denoted by $\mu_{j,k}$, is a uniformly distributed random variable between 2 and 20. For the power consumption in each *k*-th server, the $P_{k,const}$ term is a uniformly distributed random variable between 1 and 5, whereas the $P_{k,lin}$ term is a uniformly distributed random variable between 5 and 15. For the utility functions, each α_i value is assumed to be a uniformly distributed random variable between 2 and 3, and the β_i values are assumed to be equal to 7. The unit energy price $Price$ is initialized to be

1. We will change the *Price* parameter later in the experiments.

We compare the total profit of the cloud computing system achieved by the proposed near-optimal algorithm with that achieved by the baseline algorithm. In the baseline algorithm, the application environments are randomly assigned to the servers in the cloud computing system. The baseline algorithm performs near-optimal request dispatching and resource allocation using the iterative algorithm described in Section IV.C.

In the first experiment, we change the unit energy price *Price* from 0.4 to 1.6 and compare the total profit achieved by the proposed algorithm and the baseline algorithm. Figure 3 illustrates the normalized total profit versus the unit energy price of the proposed near-optimal algorithm and the baseline algorithm. We can observe from Figure 3 that the proposed near-optimal algorithm consistently outperforms the baseline algorithm. Moreover, it can be observed that the proposed near-optimal algorithm achieves higher improvement compared with the baseline algorithm when the unit energy price is higher. For example, when the unit energy price is 1.0, the total profit obtained by the proposed algorithm is 37.2% higher than the baseline algorithm. When the unit energy price is 1.6, the total profit obtained by the proposed algorithm is 65.7% higher than the baseline algorithm.

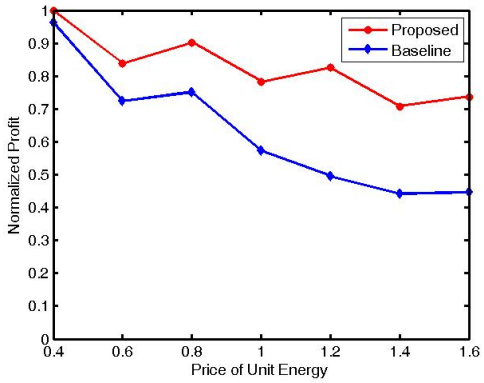

Figure 3. The normalized total profit versus the unit energy price of the proposed near-optimal algorithm and the baseline algorithm.

In the second experiment, we change the number of servers in the cloud computing system from 15 to 30, and compare the total profit achieved by the proposed algorithm and the baseline algorithm. In this experiment, we fix the unit energy price *Price* to be 1. Figure 4 illustrates the normalized total profit versus the number of servers in the cloud computing system of the proposed near-optimal algorithm and the baseline algorithm. We can observe from Figure 4 that the proposed near-optimal algorithm consistently outperforms the baseline algorithm. Moreover, the proposed algorithm achieves higher improvement compared with the baseline algorithm when there are more servers in the cloud computing system. This is partially because the proposed algorithm has more freedom in performing near-optimal application environment assignment in this case, compared with baseline algorithm.

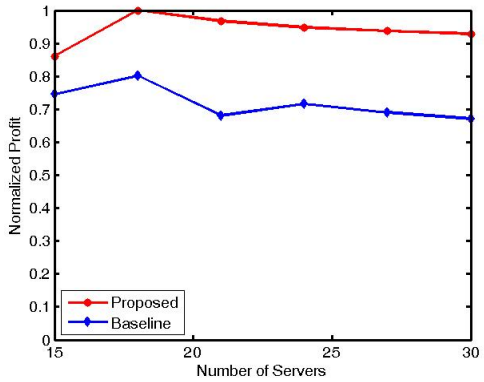

Figure 4. The normalized total profit versus the number of servers of the proposed near-optimal algorithm and the baseline algorithm.

VI. CONCLUSION

In this paper, we consider the problem of SLA-based joint application environment assignment and resource allocation optimization in a data center in the cloud computing framework. The objective is to maximize the total profit, which is the total price gained from serving the clients subtracted by the energy cost of the data center. The total price depends on the average service request response time for each client as defined in their utility functions. We propose a near-optimal solution of the joint optimization problem based on the Hungarian algorithm for the assignment problem, as well as convex optimization techniques, in a way that is similar to the constructive partitioning algorithm in VLSI CAD. We also provide a distributed version of the near-optimal solution comprised of a central resource manager and distributed local agents, in order to enhance the scalability of the solution. Experimental results demonstrate that the proposed near-optimal joint application environment assignment and resource allocation algorithm consistently outperforms baseline algorithms.

REFERENCES

[1] B. Hayes, "Cloud Computing," *Communications of the ACM*, 2008.

[2] R. Buyya, "Market-oriented cloud computing: vision, hype, and reality of delivering computing as the 5th utility," in *9th IEEE/ACM International Symposium on Cluster Computing and the Grid* (CCGrid), 2009.

[3] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Pabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Communications of the ACM*, 2010.

[4] M. Pedram, "Energy-efficient datacenters", *IEEE Trans. on CAD*, 2012.

[5] L. A. Barroso and U. Holzle, "The case for energy-proportional computing," *IEEE Computer*, 2007.

[6] R. H. Katz, "Tech Titans Building Boon," *IEEE Spectrum*, 2009.

[7] H. Goudarzi and M. Pedram, "Multi-dimensional SLA-based resource allocation for multi-tier cloud computing systems," *Proc. of IEEE Cloud*, 2011.

[8] Y. Wang, S. Chen, H. Goudarzi, and M. Pedram, "Resource allocation and consolidation in a multi-core server cluster using a Markov decision process model," *Proc. of ISQED*, 2013.

[9] Y. Wang, X. Lin, and M. Pedram, "A sequential game perspective and optimization of the smart grid with distributed data centers," *Proc. of IEEE ISGT*, 2013.

[10] G. Wei, A. V. Vasilakos, Y. Zheng, and N. Xiong, "A game-theoretic method for fair resource allocation for cloud computing services," *The Journal of Supercomputing*, 2010.

[11] R. Buyya and M. Murshed, "GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *Concurrency and Computation Practice & Experience*, 2002.

[12] K. Krauter, R. Buyya, and M. Maheswaran, "A taxonomy and survey of grid resource management systems for distributed computing," *Software Practice and Experience*, 2002.

[13] Z. Liu, M. S. Squillante, and J. L. Wolf, "On maximizing service-level agreement profits," in *3rd ACM Conference on Electronic Commerce*, 2001.

[14] L. Zhang and D. Ardagna, "SLA based profit optimization in autonomic computing systems," in *2nd Int. Conf. on Service Oriented Computing*, 2004.

[15] D. Ardagna, M. Trubian, and L. Zhang, "SLA based resource allocation policies in autonomic environments," *Journal of Parallel and Distributed Computing*, 2007.

[16] A. Chandra, W. Gongt, and P. Shenoy, "Dynamic resource allocation for shared clusters using online measurements," *International Conference on Measurement and Modeling of Computer Systems* (SIGMETRICS), 2003.

[17] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, "Managing energy and server resources in hosting centers," in *Proc. of the 18th ACM Symposium on Operating Systems Principles* (SOSP'01), 2001.

[18] X. Zhu and S. Singhal, "Optimal resource assignment in Internet data centers," in *Proc. of the 9th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems*, 2001.

[19] C. Santos, X. Zhu, and H. Crowder, "A mathematical optimization approach for resource allocation in large scale clusters," Technical Report HPL-2002-64, HP Labs, March 2002.

[20] W. E. Walsh, G. Tesauro, J. O. Kephart, and R. Das, "Utility functions in autonomic computing," in *Proc. of IEEE International Conf. on Autonomic Copmuting*, 2004.

[21] M. N. Bennani and D. A. Menasce, "Resource allocation for autonomic clusters using analytic performance models," in *Proc. of the 2nd Int. Conf. on Autonomic Computing*, 2005.

[22] S. Srikantaiah, A. Kansal, and F. Zhao, "Energy aware consolidation for cloud computing," in *Workshop on Power Aware Computing and Systems* (HotPower'08), 2008.

[23] I. Hwang, T. Kam, and M. Pedram, "A study of the effectiveness of CPU consolidation in a virtualized multi-core server system," in *Proc. of International Symposium on Low Power Electronics and Design* (ISLPED), 2012.

[24] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, 3rd edition, 1991.

[25] A. K. Parekh, "A generalized processor sharing approach to flow control in integrated services networks," Ph.D. Thesis, Department of Electrical Engineering and Computer Science, MIT, Febrary 1992.

[26] Z. Zhang, D. Towsley, and J. Kurose, "Statistical analysis of generalized processor sharing scheduling discipline," *ACM SIGCOMM'94 Conf. on Communications Architectures, Protocols and Applications*.

[27] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *Proc. of the 19th ACM Symposium on Operating System Principles* (SOSP), 2003.

[28] C. Alpert and A. B. Kahng, "Recent directions in netlist partitioning: a survey," in *Integration, the VLSI Journal*, 1995.

[29] S. H. Gerez, *Algorithms for VLSI Design Automation*, John Wiley & Sons, 1998.

[30] L. Kleinrock, *Queueing Systems, Volume I: Theory*, New York: Wiley, 1975.

[31] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

[32] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 1.21." http://cvxr.com/cvx, Feb, 2011.

[33] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, Vol. 2, Issue 1 – 2, pp. 83 – 97, March 1955.