RC25504 (WAT1410-085) October 22, 2014
Computer Science

IBM Research Report

A Probabilistic Framework and Statistical Sampling Approach to Optimized Placement in the Cloud

Asser N. Tantawi
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598
USA

Research Division
Almaden – Austin – Beijing – Cambridge – Dublin - Haifa – India – Melbourne - T.J. Watson – Tokyo - Zurich

A Probabilistic Framework and Statistical Sampling Approach to Optimized Placement in the Cloud

Asser N. Tantawi
IBM T.J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598
tantawi@us.ibm.com

Abstract

We consider a cloud system environment, consisting of physical entities, subjected to user requests, consisting of virtual entities with relationship constraints among them, such as location constraints. In this case, the resource allocation problem is a mapping of virtual to physical entities which satisfies the constraints and optimizes an objective function which combines system and user performance. The typical problem size, nature of relationship constraints, complexity and adaptability requirement of the objective function, as well as solution timing budget make traditional techniques for solving this combinatorial optimization problem infeasible. In this paper we outline an efficient technique that is based on random search methods and uses a probabilistic framework and statistical sampling methods. In particular, the proposed technique utilizes (1) importance sampling as a mechanism for describing the optimal solution through marginal distributions, (2) independent sampling via a modified Gibbs sampler with intra-sample dependency, and (3) a jumping distribution that uses conditionals derived from the relationship constraints given in the user request and cloud system topology, and the importance sampling marginal distributions as posterior distributions.

1 Introduction

Cloud services have progressed in recent years from provisioning Virtual Machines (VM) in the physical cloud infrastructure to virtual platforms and virtual applications, which have become the new cloud workload. Open source programmable interfaces have been developed to interact with the cloud management system and define a software environment describing the cloud infrastructure which consists of compute, storage and network nodes, e.g. [2]. Using a document [1], a user specifies a workload consisting of logical entities, such as VMs, data volumes, communication links, and services, and their needs of the underlying physical resources. Moreover, the user specifies requirements on the provisioned topology of such logical entities. Examples of such requirements include physical proximity of the logical entities, availability/reliability concerns, preferred hosting requirements, licensing and cost issues, and migration requirements. The degree with which such requirements are satisfied during provisioning is a user measure of Quality-of Service. On the other hand, the cloud service provider attempts to maximize the use of the physical resources in a way that provides best performance to users, e.g. load balanced resources. When a user request arrives to the cloud management system, the placement engine decides on a mapping of the logical entities in the request to the physical entities in the cloud system, given its current state, in a way to optimize a given objective function

which combines user and provider objectives [4]. This placement optimization problem is quite challenging for several reasons. The size of the problem is quite large, having thousands of physical entities, hundreds of logical entities and hundreds of constraints in a request. Allowing the remapping of existing allocations makes the problem even larger. Typically, the objective function is constructed from policies that users and providers specify. Such policies are not necessarily well behaved, in the mathematical sense, and subject to change and evolution. Hence, the optimization approach cannot assume and/or exploit properties of the objective function. Needless to say that the placement decision is expected to be fast, i.e. sub-second and not multiple seconds or minutes, in order to cope with the cloud workload traffic and the potential need to redistribute resources through migration of logical entities in the cloud.

Briefly, there is a growing research to solving the cloud placement problem. Recently, a technique for attempting to decrease the size of the problem has been proposed [6]. Further, a more promising technique which uses biased sampling along with cross-entropy [8] was introduced [10].

In this paper, we provide a probabilistic framework and statistical sampling method for the algorithm outlined in [10]. The problem is stated as a search problem [9] and a general random search method [7, 3] is sought. An independent Metropolis-Hastings sampling is performed. In particular, the Gibbs sampling [5] technique is modified and restricted to intra-sample dependency. The constraints specified in the user request are used to construct the conditional probabilities and the importance sampling marginal distributions are employed as posterior distributions.

2 Problem Statement

We use the following notation. For vectors and matrices, we use boldface capital letters, e.g. $\mathbf{V}$ and $\mathbf{M}$. A corresponding small letter denotes an element in the vector or matrix, e.g. v and m. A subscripted small letter represents a particular element given by the value of the subscript, e.g. v_i and $m_{i,j}$. For convenience, a subscripted capital letter representing a matrix denotes a vector row in the matrix, where the row number is given by the value of the subscript, e.g. $\mathbf{M}_i$. (We will not need to denote vector columns in matrices.) The 1-norm of a vector is denoted by $\|\mathbf{V}\|$, which is the sum of the absolute values of its elements. For sets, we use a calligraphic capital letter, e.g. $\mathcal{S}$. A normal capital letter is an integer, and its corresponding small letter takes values in the enumeration from one to the value of the capital letter, e.g. I and $i = 1, 2, \cdots, I$.

Define the sets $\mathcal{M} = \{1, 2, \cdots, M\}$ and $\mathcal{N} = \{1, 2, \cdots, N\}$, where $M, N \geq 1$. Let $\mathbf{X} = [x_1 \ x_2 \ \cdots \ x_M]$ be a vector representing variables taking values in $\mathcal{N}$, i.e. $x_m \in \mathcal{N}$, $m \in \mathcal{M}$. We refer to a particular valued vector $\mathbf{A} = [a_1 \ a_2 \ \cdots \ a_M]$, where $a_m \in \mathcal{N}$, $m \in \mathcal{M}$, as an assignment to $\mathbf{X}$.

Define a scalar objective function $f(\mathbf{X})$ with range $\mathbb{R}$, the set of real numbers. The unconstrained state space for variable $\mathbf{X}$ is the cartesian power $\mathcal{N}^M$. Let $\mathcal{S}$ denote a constrained, nonempty state space, $\mathcal{S} \subseteq \mathcal{N}^M$. The optimization problem is stated as,

$$\min_X f(\mathbf{X}), \quad \mathbf{X} \in \mathcal{S}. \tag{1}$$

We use the notation $\mathbf{X}_{<m}$ to denote the variable vector $\mathbf{X}$ excluding the elements $\{x_m, x_{m+1}, \cdots, x_M\}$, where $m = 2, \cdots, M$ and $M \geq 2$.

In relation to the cloud placement (assignment) problem, we have M physical entities, N logical entities in the user request, $\mathbf{X}$ is a variable mapping logical to physical entities, $\mathbf{A}$ is a particular mapping (solution to the problem), and S the set of possible solutions given the requirement constraints specified in the user request. The objective function $f(\mathbf{X})$ combines user and provider objectives. We do not make assumptions about $f(\mathbf{X})$ other than it could be numerically evaluated given $\mathbf{X}$ and the current state of the system.

Define $\mathcal{A}$ as an arbitrary, nonempty subset of $\mathcal{S}$, i.e. $\emptyset \neq \mathcal{A} \subseteq \mathcal{S}$. Let $\mathcal{A}$ contains $L \geq 1$ unique assignments, i.e. $\mathcal{A} = \{\mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_L\}$, where $\mathbf{A}_l = [a_{l1} \ a_{l2} \ \cdots \ a_{lM}]$, $l \in \mathcal{L}$, is an assignment vector. If $L = 1$ then $\mathcal{A}$ represents the set consisting of a unique solution $\mathbf{A}_1$.

Define an $M \times N$ (row) stochastic matrix $\mathbf{P}$, i.e. element $p_{m,n} \in [0,1]$, $\|\mathbf{P}_m\| = 1$, $m \in \mathcal{M}$, and $n \in \mathcal{N}$. We refer to a stochastic matrix $\mathbf{P}$ as *deterministic* if the elements are such that $p_{m,n} \in \{0,1\}$. Let $p_{m,n}$ be the probability that $a_{lm} = n$ over $l \in \mathcal{L}$. Hence, $\mathbf{P}_m$ represents the marginal probability distribution of the m^{th} element of the assignments in $\mathcal{A}$.

3 Solution Approach

Since the objective function $f(\mathbf{X})$ could be quite general, we develop a solution approach that does not rely on properties, such as convexity, nor devise heuristics implied by its shape. Rather, we consider the optimization problem as a general search problem [9] for an optimal $\mathbf{X}^*$ with minimum $f(\mathbf{X}^*)$ in the solution space S. A general random search method [3] consists of the following steps. (A stopping criterion is expected but not specified.)

1. Select a starting point $\mathbf{X}(0) \in \mathcal{S}$ and an initial estimate of the optimal solution $\mathbf{X}^*(0) \in \mathcal{S}$. Let $k = 0$.
2. Generate a candidate solution $\mathbf{X}'(k) \in \mathcal{A}(k)$, where $\mathcal{A}(k)$ defines a neighborhood of solutions around and including $\mathbf{X}(k)$.
3. Determine the next point $\mathbf{X}(k+1) \in \{\mathbf{X}(k), \mathbf{X}'(k)\}$, using $f(\mathbf{X}(k))$ and $f(\mathbf{X}'(k))$.
4. Obtain a new estimate of the optimal solution $\mathbf{X}^*(k+1)$. Let $k = k+1$ and go to step 2.

There area several choices in this general search method: (1) defining the neighborhood $\mathcal{A}(k)$; (2) generating a candidate solution $\mathbf{X}'(k)$; (3) determining a next point $\mathbf{X}(k+1)$; and (4) estimating an optimal solution $\mathbf{X}^*(k)$.

Our solution approach makes the following choices: (1) the neighborhood $\mathcal{A}(k)$ is characterized by the marginal probability distributions of the m^{th} element in $\mathbf{X}$; (2) candidate solution $\mathbf{X}'(k)$ is generated using a modified Gibbs sampling as described below; (3) the next point $\mathbf{X}(k+1)$ is the generated point $\mathbf{X}'(k)$; and (4) an estimate of optimal solution $\mathbf{X}^*(k)$ is generated using importance sampling technique [8].

4 Algorithm

An outline of the placement algorithm follows.

1. $k = 0$. Initially, set the stochastic matrix $\mathbf{P}(0)$ proportional to resource availability, i.e. $p_{m,n}(0) = 1 - u(PM_n)$, $m \in \mathcal{M}$, and $n \in \mathcal{N}$, where $u()$ represents the utilization of the bottleneck resource or a measure of utilization of the multiple resources on a PM.
2. Use $\mathbf{P}(k)$ to generate a number of independent samples (solutions).
   (a) Define the set $\mathcal{R} = \{1, 2, \cdots, R\}$, $R \geq 1$. Let $\mathbb{B} = \{\mathbf{B}(r);\ r \in \mathcal{R}\}$ be a family of R stochastic matrices, each of size $M \times N$. We refer to $\mathbb{B}$ as a set of biasing matrices. For R biasing criteria, $\mathbf{B}(r)$ represents the basing matrix for criterion $r, r \in \mathcal{R}$. Each criterion represents a type of requirement constraint in the user request, e.g. communication, location, target preference, license usage, and cost constraints. Define a weight vector $\mathbf{W}$ of length R, where element $w_r \geq 0$ is a weight associated with $\mathbf{B}(r)$, $r \in \mathcal{R}$.
   (b) A sample $\mathbf{X}$ is constructed incrementally, one element at a time. After $(m-1)$ elements are generated, where $m = 2, \cdots, M$, we have $\mathbf{X}_{<m}$. The element x_m is generated given $\mathbf{X}_{<m}$. in other words, the set $\mathbb{B}$ are filled in as conditional probabilities given $\mathbf{X}_{<m}$. Hence, we generate the elements as per the Gibbs sampling method, except that we remove the dependency on the previous sample. This yields independent samples, rather than a Markov Chain Monte Carlo sequence. In general, the chain resulting from independent samples behaves well if the jumping distribution has a heavier tail than the posterior marginal distributions.
   (c) Given an initial $M \times N$ stochastic matrix $\mathbf{P}(k)$, we evaluate the resulting one-step jumping stochastic matrix $\mathbf{P}'(k)$ as follows.

Let $\mathbf{B}$ denote the weighted product of $\mathbf{B}(r)$, given by

$$\mathbf{B} = \underset{r \in R}{\circ} \mathbf{B}(r)^{w_r},$$

where the symbol $\circ$ represents the Hadamart element-wise product of matrices, and the exponent w_r applies to all elements of matrix $\mathbf{B}(r)$. Then, we write

$$\mathbf{P}'(k) = \mathbf{diag}(\mathbf{C}) \ (\mathbf{P}(k) \circ \mathbf{B}),$$

where $\mathbf{C}$ is a normalization constant vector of length M to make $\mathbf{P}'(k)$ stochastic.

3. Order the generated samples w.r.t. $f()$ and select the best top portion of the samples. Represent each selected sample by its corresponding *deterministic* stochastic matrix, add all such matrices element-wise and normalize to generate a new stochastic matrix $\mathbf{P}(k+1)$. The latter is a characterization of the top generated samples. In such a matrix, $\mathbf{P}_m(k+1)$ represents the marginal probability distribution of the m^{th} element in the optimal solution.

4. $k = k+1$. Go to step 2 until a stopping criterion is satisfied.

5 Results

We briefly describe the setup and present only the performance of the placement algorithm in terms of its execution time. We consider a cloud system which consists of 256 Physical Machines (PM), each with CPU, RAM, and disk storage resources having capacities 32 cores, 512 GB, and 10 TB. The PMs are interconnected with a tree network of height 2, where the link capacities of edge (level 1) and core (level 2) links are 20 Gb/s and 80 Gb/s, respectively. The PMs are arranged in 16 racks, where each rack houses 16 PMs. A user request represents a 3-tier application, where the tiers have 3, 6, and 3 VMs, respectively. The resource demands of the VMs in the 3 tiers are {2 cores, 2 GB, 4TB}, {3 cores, 4 GB, 4TB}, {3 cores, 8 GB, 4TB}, respectively The communication bandwidth demand between a pair of VMs in tiers 1 and 2 is 0.3 Gb/s, and a pair of VMs in tiers 2 and 3 is 0.05 Gb/s. For high availability reasons, the location requirements are such that VMs in tier 2 are to be placed on separate PMs, and VMs in tier 3 are to be placed in different racks. Requests arrive as a Poisson process, and applications have a uniformly distributed lifetime such that the offered load on the CPU is 0.8. The system reached steady state after about 200 request arrivals. The algorithm is coded in Java and runs on a MacBook Pro with 2.4 GHs Intel Core 2 Duo and 4GB RAM, running Mac OS X 10.5 and JVM 1.6.0. Figure 1 depicts the execution of our placement algorithm for 900 requests, starting with an empty system. The average time was 49 msec.

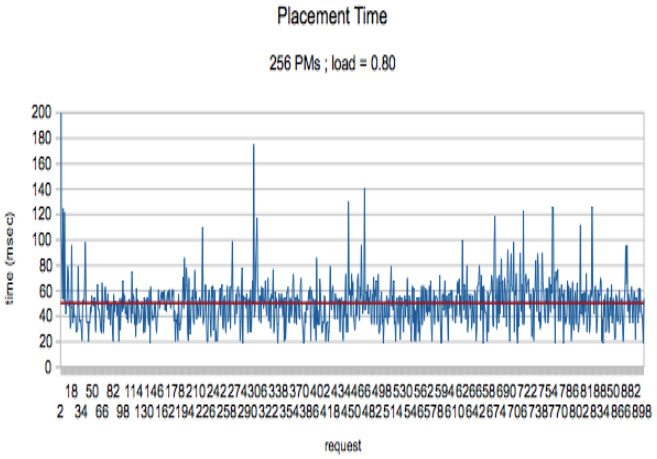

Figure 1: Placement algorithm time.

6 Conclusion

We presented an approach for solving a cloud placement optimization problem. The approach is based on a general random search technique where we employ importance sampling to construct the marginal distribution of the solution, and a modified Gibbs sampling technique restricted to intra-sample dependency using the marginal distributions as posteriors. Several remaining work is underway. We mention, for example, convergence properties of the algorithm,

criterion for selecting parameters of the algorithm, comparison of the algorithm to other techniques.

References

[1] Heat Orchestration Template (HOT) guide.

[2] OpenStack open source cloud computing software.

[3] S. Andradóttir. Accelerating the convergence of random search methods for discrete stochastic optimization. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 9(4):349–380, 1999.

[4] W. Arnold, D. Arroyo, W. Segmuller, M. Spreitzer, M. Steinder, and A. Tantawi. Workload orchestration and optimization for software defined environments. *IBM Journal of Research and Development*, 58(2):1–12, March 2014.

[5] G. Casella and E. I. George. Explaining the gibbs sampler. *The American Statistician*, 46(3):167–174, 1992.

[6] I. Giurgiu, C. Castillo, A. Tantawi, and M. Steinder. Enabling efficient placement of virtual infrastructures in the cloud. In *Middleware 2012*, pages 332–353. Springer, 2012.

[7] T. L. Lai. Sequential analysis: some classical problems and new challenges. *Statistica Sinica*, 11(2):303–350, 2001.

[8] R. Y. Rubinstein and D. P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*. Information in Sciences and Statistics Series. Springer-Verlag New York, LLC, 2004.

[9] J. C. Spall. *Introduction to stochastic search and optimization: estimation, simulation, and control*, volume 65. John Wiley & Sons, 2005.

[10] A. Tantawi. A scalable algorithm for placement of virtual clusters in large data centers. In *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2012 IEEE 20th International Symposium on*, pages 3–10. IEEE, 2012.