Robust Performance-Based Resource Provisioning

Ms.Sreeja B P¹, Ms.SarathaDevi², Ms.Narmatha K³

¹²³ Assistant Professor, Department of Information Technology, Karpagam College of Engineering, Coimbatore, Tamilnadu, India.
sreejabp@gmail.com
sarathadevi.techno@gmail.com
knarmatha18.cool@gmail.com

Abstract: Cloud computing has enabled entirely new business models for high-performance computing. It is a dedicated local high-performance computer is still an option for some, but more are turning to cloud computing resources to fulfill their high-performance computing needs. With cloud computing it is possible to tailor your computing infrastructure to perform best for your particular type of workload by selecting the correct number of machines of each type. This paper presents an efficient algorithm to find the best set of computing resources to allocate to the workload. This research is applicable to users provisioning cloud computing resources and to datacenter owners making purchasing decisions about physical hardware. Studies have shown that cloud computing machines have measurable variability in their performance. Some of the causes of performance variability include small changes in architecture, location within the datacenter, and neighboring applications consuming shared network resources. The proposed algorithm models the uncertainty in the computing resources and the variability in the tasks in a many-task computing environment to find a robust number of machines of each type necessary to process the workload. In addition, reward rate, cost, failure rate, and power consumption can be optimized, as desired, to compute Pareto fronts.

Keywords — datacenter, workload, Cloud Computing

1. INTRODUCTION

Some high-performance computing users are turning to cloud providers to complete their work due to the potential cost effectiveness and/or ease of use of cloud computing. The ability to provision hardware on-demand from a pre-defined set of different machine types, known as instance types, is very powerful. In fact, a proof of concept cluster was built by Amazon Web Services from their high performance instance types composed of over 26,000 cores with nodes connected via 10G Ethernet. This cluster ranked 101 on the Top 500 list for November 2014 [1]. The cloud has been successfully employed to process HPC jobs for actual scientific experiments [2]. Recent studies have shown that the performance of small and medium virtual clusters can compete with physical hardware. Cloud infrastructure as a service (IaaS) providers charge for the amount of time a virtual machine, known as an instance, is allocated (idle or active). This means that it is advantageous to terminate some or all instances once the workload has been processed. Leaving instances idle in the cloud is usually not cost effective. Once a new set of work needs to be processed, the decision of what instance types to start can be re-evaluated each time, considering the size and Sciences and Engineering instance types. Selecting the ideal number of instances of each instance type a user needs is challenging. The approach to provisioning computational resources given in this paper not only applies to cloud resource provisioning but also to selecting physical machines to purchase for use within HPC systems. The goal for provisioning HPC systems is to determine how to originally select or upgrade a system in such a way that maximizes the performance of the resultant system while meeting specific requirements that often include a budget constraint. The instance types available in the cloud have widely varying capabilities, by design, so that users can choose the resources that best match their workload and in doing so minimize the cost. For example, there is no need to provision high memory instance types if the workload does not require large amounts of memory. The cost for the smaller memory instance type will often be significantly less and provide nearly identical performance assuming all else is equal. Within a single IaaS provider, instance types vary in the amount of memory, number and type of CPUs, disk capacity and performance, and network performance. All of these properties of instance types affect the performance of the workload executing on the instances [5]. Due to the availability of heterogeneous resources, IaaS is inherently a heterogeneous computing system.

2. DESIGN AND IMPLEMENTATION

A. Performance Results

Stochastic linear programming is an extension of linear programming, where some of the coefficients in the objective and the constraints are random variables. The particular stochastic program we use is the recourse problem (RP) given in standard form as

\[
\text{minimize } c^T x + E \left[ Q(x,\xi) \right]
\]

where \( \xi \) is a random vector representing the uncertain parameters. For the RP in , the first stage decision variable, \( x \) is a flattened version of MB and MS. The second stage decision, \( y \) are flattened versions of the schedule p. The coefficients that are deterministic are incorporated into \( c \) and
the coefficients that are random are incorporated into \( q \). This linear RP is similar to a linear program except for the expectation of the value function, \( Q(x,\xi) \), in the objective. The RP in (13) is known as a two-stage RP. The optimization problem finds the optimal \( x \) that minimizes the sum of the linear function \( c^T x \) and the expectation of \( Q \). The second stage optimization problem finds optimal \( y \) given a fixed realization of \( \xi \). The random vector \( y \) is known as the recourse decision vector. This RP finds a robust solution for \( x \) in the sense that the objective value will on average be minimal when the optimal value of \( x \) is used. The solution \( x \) is robust to unknown values of the parameters. The vector \( x \) is often referred to as a strategy of the RP.

The primary issue with stochastic programming is accurately computing the expectation in the objective function. The SAA approach uses many samples of \( \xi \) to compute the expectation as a sample average. Realizations of \( \xi \) are called scenarios. The process of creating scenarios is known as scenario generation. When using SAA, generating a reasonably small set of representative scenarios is important. Let there be \( K \) scenarios. For scenario \( k \), let the probability of occurring be given by \( p_k \). With SAA, the scenarios are generated by randomly sampling according to its distribution, thus all the samples are equally probable. The expectation operator is linear and is applied to a linear function, namely \( q^T y(\xi) \). Each scenario (i.e., realization of \( \xi \)) defines a set of matrices \( T_k \) and \( W_k \), and vectors \( q_k \) and \( h_k \). Each scenario also introduces a new vector of decision variables \( y_k \) into the problem. The SAA is an unbiased estimator of the mean. In practice, it converges to the mean quickly in \( K \). The DEP can have a large number of variables and constraints. For very large problems a technique called the L-method can be used to exploit the block structure of the constraint matrix to distribute the work of solving the linear program to many nodes. The problem is broken into two coupled decisions. The first is what to provision or purchase, namely \( MB \) and \( MS \). Then the random variables in the problem are realized and the second decision, known as the recourse decision, can be made. For this work, the random variables are the arrival rates, execution times, and power consumption, but virtually any other parameter in the model can be converted to a random variable. The recourse decision involves selecting the schedule \( p \) that is optimal for the actual arrival rates, execution times, and power consumption of the tasks.

\[
\text{minimize } c^T x + p_1 q_1^T y_1 + \cdots + p_K q_K^T y_K \\
\text{subject to: } \begin{align*}
Ax & = b \\
T_1 x + W_1 y_1 & = h_1 \\
& \vdots \\
T_K x + W_K y_K & = h_K \\
x, y_1, \ldots, y_K & \geq 0
\end{align*}
\]

Multi-objective optimization is challenging because there is usually no single solution that is superior to all others. Instead, there is a set of superior feasible solutions that are referred to as the non-dominated solutions [31]. When all objectives are to be minimized, a feasible solution \( x_1 \) dominates a feasible solution \( x_2 \) where \( d \)th objective function. Feasible solutions that are dominated are generally of little interest because one can always find a better solution from the non-dominated set. The non-dominated solutions, also known as outcomes and efficient points, compose the Pareto front. There are many techniques for solving multi-objective optimization problems. For linear optimization problems, there are two primary approaches. The first is known as Benson’s algorithm that iteratively refines the Pareto front. The second is a technique that converts the multi-objective problem into a set of scalar optimization problems through a process called scalarization. There are many scalarization techniques but most are specializations of Pascoletti-Serafini scalarization such as the weighted sum algorithm.

\[
\forall d \quad f_d(x_1) \leq f_d(x_2) \\
\exists d \quad f_d(x_1) < f_d(x_2)
\]

Three very different environments are used to analyze the behavior of the proposed algorithms for resource provisioning. The heuristic-based algorithms H1, H2, and H3, and the RP that uses stochastic programming are compared. The first environment is a small example used to illustrate the behavior of the algorithms. The second environment is a larger environment. The third environment was built based on benchmark data. A complete description of the source code, system parameters, and simulation results are available in the
supplementary material to aid in reproducibility. The data is provided as CSV and JSON files with further details available in the accompanying README.txt file.

The steady-state model and the scenario generation are written in C++. To generate the DEP in (14) the Coin-OR Stochastic Modeling interface (SMI) is used [38]. The underlying linear programming problem is solved with the Coin-OR Linear Programming (CLP) solver [39]. CLP is a high quality open-source solver written in C++. All the simulations where run on an Apple MacBook Pro Mid 2014, 2.2 GHz Intel Core i7. The solver is single threaded so timing results are for one core. To compare MVP, RP, and WS against the heuristics described in Section 7, one must be able to compute all the objectives including reward rate. The reward rate is a function of the steady-state schedule. To allow heuristics to perform as best as possible we use the optimal schedule from the steady-state model by solving a linear programming problem where the MB is fixed by the heuristic and MS = 0. When computing the expected objective values the optimal schedule is used for each scenario.

![Graph showing expected reward rate for different budgets for the E1 environment](image)

(a) Shows the reward rate computed with the mean of the parameters.
(b) Shows the expected reward rate over all uncertainty in the parameters.

### 3. Conclusion

Stochastic programming is a powerful tool that can be applied to make robust decisions in the midst of the inherit uncertainty in computing systems in both IaaS provider clouds and traditional environments. The linear steady-state model and representative stochastic model enables the use of an efficient two-stage stochastic program for solving the machine provisioning problem.

### References


Author Profile

Ms. Sreeja B P,
Working as an Assistant Professor in IT Department KCE from 2011. Has received her M.Tech Degree in Computational Engineering and networks, B.Tech Degree in Information Technology & Her area of working includes Wireless Sensor Networks, Machine Translation, Cloud Computing.

Ms. Saratha Devi G,
Working as an Assistant professor in the department of Information Technology in Karpagam College of Engineering, Coimbatore, Tamil Nadu from 2012. She is doing her Ph.d in Information and Communication Engineering in Anna university, Chennai. She received M.E degree in Karpagam university, Coimbatore. She received her B.E in Avinashilingam university, Coimbatore.

Ms. Narmatha K,
Received my bachelor’s degree from SNS College of Technology, Anna University, Coimbatore, India in Computer Science and Engineering during 2011. Received my Master’s degree in Embedded Systems from Kumaraguru College of Technology, Anna University, Coimbatore, during 2013. I’m currently working as Assistant Professor at the department of Information Technology in Karpagam College of Engineering, Coimbatore, India.