An Approximate Bribe Queueing Model for Bid Advising in Cloud Spot Markets

Bogdan Ghiț*[1][0000−0002−2530−8736] and Asser Tantawi[2][0000−0001−6598−8863]

[1] Databricks Inc., Amsterdam, the Netherlands
bogdan.ghit@databricks.com
[2] IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA
tantawi@us.ibm.com

Abstract. We consider the scheduling system of a container cloud spot market where the user specifies the requested number of containers and their resource requirements, along with a bid value. Jobs are preemptively ordered based on their bid values as the available capacity, which is excess capacity made available for the spot market, may vary over time. Due to this variation, the number of allocated containers to a job may vary during its lifetime, resulting in users experiencing periods of degraded performance, potentially leading to job slowdown. We want to model and analyze such a scheduling system starting from first principles, inspired by the M/M/1 bribe queue. Thus, we introduce a simple, empirical queueing model which parametrically relates job slowdown to bid values given load and bid distribution. We demonstrate the accuracy of our approximation and parameter estimation through simulation.

1 Introduction

Cloud providers make excess resource capacity available at discounted prices through a so called spot market [5, 17, 10]. The unit of sale is typically a Virtual Machine (VM) instance, but variations may also include containers or collection of VM instances in case of a container or batch service, respectively. Users submit their bids for such resource units which have a time-varying price per unit that is controlled by the provider. When the price goes above the user bid, the user loses the corresponding resources. Such a market is attractive to users because spot instance have relatively low prices. However, a major drawback of the spot market is that users need to deal with potential unit revocations [1], which are difficult to anticipate. For both the service provider and users, there is a crucial need for a prediction tool to provide (1) revenue estimates as a function of price and (2) quality of service as a function of bid, respectively.

We consider an enhanced management of excess resource capacity through a scheduling system, where a user specifies a bid value, which acts as a priority level. As the available capacity shrinks or higher priority jobs are submitted, the

* This work was done while B. Ghit was an intern at the IBM T.J. Watson Research Center.

scheduler reclaims resources from lower priority running jobs, by preempting them and putting them back in the queue. From a modeling point of view, we consider a preemptive priority scheduler of jobs using containers. At job submission time, the user specifies the requested number of containers and their resource requirements, along with a bid value. The number of allocated containers may vary during the lifetime of a job, anywhere from zero to the requested number. The tasks of a job are managed by a task scheduler and run on the allocated containers. The job continues to execute, with potential degraded performance, as the number of allocated containers varies. The deallocation of a container causes the currently running task(s) on the container to be aborted. If the number allocated goes down to zero, the job is put back in the queue.

We seek to obtain a simple and empirical expression for the job slowdown as a function of bid value. To this end, we propose a parametric approximate expression inspired by the M/M/1 *bribing queue* [9]. We are also concerned with the dynamic estimation of parameters in order to adapt the queueing model and provide accurate performance predictions in the face of time-varying workloads. We achieve this by employing an extended Kalman filter [12] on the slowdown and bid values measured over a period of time. We validate the accuracy of our approximation through simulation experiments.

The main contributions of this paper are: (1) a simple, empirical, parametric closed-form approximate expression for the job slowdown as a function of bid value for scheduling jobs in a container cloud spot market, and (2) a methodology, based on filtering techniques, for dynamically estimating the model parameters at runtime based on measurements.

2 Problem Description

We consider a container cloud spot market, enhanced with a job scheduler, providing differentiated performance based on bids. As the market price fluctuates, a job may wait in the queue and/or be preempted during its life time. Partial preemption is possible if only a fraction of the containers that a job needs are de-allocated. This is in contrast with a typical spot market (without queueing and only for single instances (containers)), where jobs are either rejected or completely aborted.

We assume a parallel job model such as data analytics applications in a real environment with multiple resources. A job is decomposed into multiple tasks that run on units of the available capacity called *compute slots*. Without loss of generality, we refer to a *compute slot* as a container, which is the unit of allocation. Typically, a worker runs in a container and is responsible for the execution of the tasks assigned to it by the task scheduler. We use the terms worker and container interchangeably throughout the paper. In this paper we are *not* concerned with the task graph of the job, nor the scheduling of individual tasks. Rather, we remain at the job level and consider the job, as a whole, and its allocated workers. We assume that running workers are always busy executing tasks. Because the number of tasks is typically much larger than the allocated

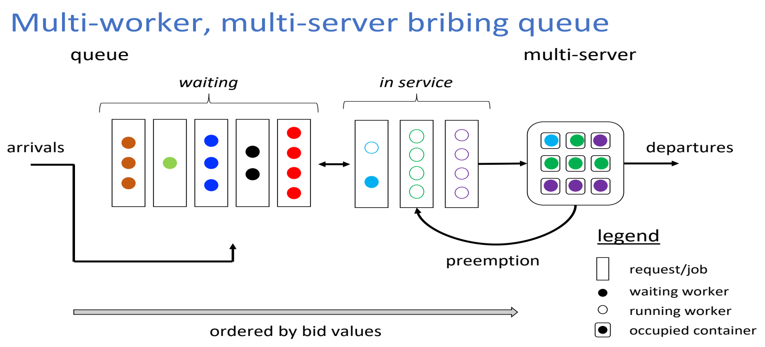


Fig. 1: An overview of our queueing system.

containers, jobs are often multi-waved, thereby running only a fraction of their tasks at a time.

Figure 1 depicts the scheduling of jobs in this environment. Jobs arrive independently according to some stochastic process. A job comes with a bid value and the jobs are ordered based on their bid values, so that lower bid values are in the back of the queue. A job requests some number of containers to run its tasks. Jobs and containers are drawn as rectangles and circles within the rectangles, respectively. The system has a certain container capacity, represented as squares. A circle within a square represents a worker assigned to a container. Jobs wait in the queue until they are allocated at least one container. At such a time they move to the *in service* area. They remain in service until they complete and depart from the system. While *in service*, the number of allocated containers may grow and shrink, depending on container availability and preemption. Allocated and unallocated containers are drawn as hollow and solid circles, respectively. Preempted tasks may need to be re-executed. If all the containers of a job are deallocated, the job goes back to the waiting queue.

A job could be in one of three states, as far as its container allocation is concerned: waiting, executing, or partially executing. A job is in a waiting state if all its requested containers are not allocated yet (all solid circles). Alternatively, a job is in an executing state if all of its requested containers are allocated (all hollow circles). And, a partially executing job is one with a non-zero (and non-one) fraction of its requested containers are allocated. Such a job is progressing with a degraded performance. Thus, jobs in the *in service* area are either executing or partially executing. In general, containers may not be the same size in terms of their allocated resources (CPU and memory). However, in the case of homogeneous containers, we may only have at most one job in the partially executing state.

3 Modeling and Analysis

We seek a functional relationship between job slowdown and bid. In addition to the bid value, the job slowdown depends on many factors such as the current load in the system, the resource requirements of the job, the number of containers allocated to run the job, and the bid values of other jobs. To assess the relationship between the job slowdown and the bid value, given such factors, we need to develop an analytic model which can be used to predict either the slowdown given a bid value, or the bid value which would result in a given slowdown. Both predictions could be valuable to users to set the expectation for job performance and to advise setting a bid value, respectively.

3.1 Definitions and Assumptions

Let job arrivals constitute a Poisson process with rate λ and let X be the random variable representing the job bid value. Without loss of generality, we assume that the bid value is in the set $\mathcal{X} = [0, 1]$. The probability distribution function of X is denoted by $B(x) = Pr[X \leq x], x \in \mathcal{X}$ which is continuous and differentiable.

A job leaves the system after completing all of its work, which we denote by W. Jobs request homogeneous workers that have the same amount of resources. We further assume that it is always possible to divide the remaining work among the running workers, with no running (allocated) worker staying idle. The service time R of a job is the duration over which the running workers execute all the work needed. Thus, if a job is allocated all requested workers the service time is given by $R = W/K$, where K denotes the number of workers. This analysis is for the preemptive-resume case[3]. Let $\mu = 1/\overline{R}$ be the service rate, where $\overline{R}$ is the average job service time. We assume homogeneous slots, i.e. equal amount of resources per slot, and that one worker fits exactly into one slot. Thus, the offered load is given by $\rho = \lambda \overline{K}/N\mu < 1$, where $\overline{K}$ is the average number of workers per job and N is the system capacity.

We denote the average response time of a job with bribe value x by $T(x)$. We further define the job slowdown as the ratio of the average response time $T(x)$ and the average service time $1/\mu$, denoted by $S(x) = T(x)/(1/\mu)$.[4]

3.2 Bribery Queueing Model

The simplest case for our queueing system is when K is fixed at $K = 1$, W is exponentially distributed, and $N = 1$, resulting in the M/M/1 bribing[5] queue [9]. For such a model, the slowdown of a job with bribe value x is given by:

$$S(x) = \frac{1}{(1 - \rho(1 - B(x)))^2}. \tag{1}$$

[3] Note that in the case of exponential service time, the preemptive-repeat and preemptive-resume cases result in similar expressions for the average response time.

[4] Note that we define the slowdown as the ratio of two average values, and not the average of a ratio of two values. The latter alternative definition would have (1) resulted in a more complex derivation and conditional expression on the service time and, more importantly, (2) necessitated *a priori* knowledge of job service time, which may not be available in practice.

[5] We will use the words bribe and bid interchangeably throughout this paper.

Let S be the random variable representing the slowdown across all jobs. The bribe value which yields a given slowdown of $s \in \mathcal{S}$ is obtained by inverting Equation 1, as

$$x(s) = B^{-1}\left(1 - \frac{1 - 1/\sqrt{s}}{\rho}\right). \tag{2}$$

Our queueing system, described in Section 2, along with other *modern* job models in data centers and clouds, are quite challenging to analyze [6]. Though simplistic and limiting, the single-server bribing queue has an appealingly concise expression. We seek an approximate expression for our generalized model by introducing a parameter vector, Θ, consisting of two model parameters, $\Theta = [\theta_0, \theta_1]$, which act as scale and shape parameters, respectively, such that $0 \leq \theta_0 < 1$ and $\theta_1 > 0$. First, θ_0 acts as a (virtual) replacement for the server utilization, ρ, which may not be available to an external observer. Second, θ_1 captures the variation in the expression due to the model features described previously. Hence, we extend Equation 1 and write a closed-form approximate expression for the job slowdown as a function of B, the bid distribution, and Θ, the parameter vector, in addition to x, the bid value, as

$$S(x; B, \Theta) = \frac{1}{\left[1 - \theta_0 (1 - B(x))^{\theta_1}\right]^2}. \tag{3}$$

The bid which results in a given job slowdown may be obtained by inverting the above equation, similar to Equation 2.

3.3 Parameter estimation

This section addresses the issue of dynamically estimating the parameters in our model of the scheduling system. The model has bid values as input and corresponding slowdown values as output. There are two sets of parameters: (1) bidding parameters which characterize the bid distribution, $B(x)$, and (2) model parameters which characterize the queueing model, i.e. the relationship between a bid value x and its corresponding slowdown value $S(x)$. In practice, one is not given such a distribution or parameter values. Thus, we need to have a dynamic estimator which derives them dynamically, based on observations of the job bid sequence $\{x_0, x_1, \cdots, x_i\}$ and corresponding attained slowdown sequence $\{s_0, s_1, \cdots, s_i\}$, $i \gg 0$. Based on such sequences, the estimator builds a model and keeps updating the two sets of parameters dynamically. The model produced by the estimator may be used to predict a slowdown $\tilde{S}(x)$ for a given bid value x, or a bid value $\tilde{x}$ for a desired slowdown value s.

Our design for such a dynamic estimator is depicted in Figure 2. We separate the estimation process into two independent processes: one for estimating the bidding parameters and another for estimating the model parameters. Firstly, we use the job bid sequence $\{x_0, x_1, \cdots, x_i\}$ to derive a bid distribution. Then, using the latter, along with the attained slowdown sequence $\{s_0, s_1, \cdots, s_i\}$, we use a filter to estimate the model parameters dynamically. Both the bid distribution and parameter values could then be used for prediction.

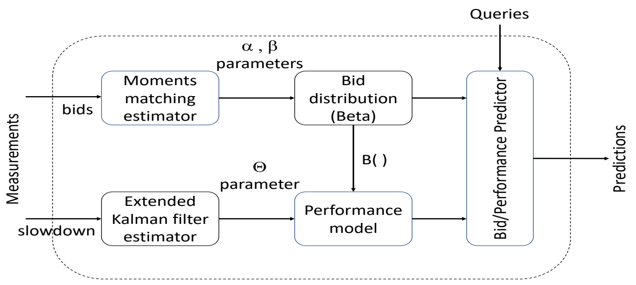


Fig. 2: Our framework for dynamic estimation and prediction.

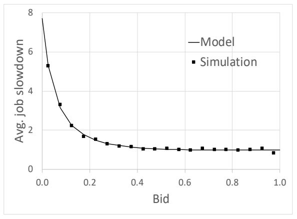
We select a probability distribution for the bid distribution from the first and second moments of the distribution. In general, given the first few moments of a probability distribution over a finite range, the maximum-entropy distribution may be obtained using Lagrange multipliers [3], which may be approximated by the Beta distribution [13]. We characterize the bid distribution with two parameters: α and β, associated with the first and second moments of X. Let $\tilde{r}$ and $\tilde{s}^2$ be the sample average and sample variance of the observed bid values over a given time interval. Thus, using the method of moments [4], we can estimate the parameters $\tilde{\alpha}$ and $\tilde{\beta}$, as $\tilde{\alpha} = \tilde{r}\left(\frac{\tilde{r}(1-\tilde{r})}{\tilde{s}^2} - 1\right)$ and $\tilde{\beta} = (1-\tilde{r})\left(\frac{\tilde{r}(1-\tilde{r})}{\tilde{s}^2} - 1\right)$, respectively.

As changes occur in the system, such as the nature of workload, load intensity, and cluster configuration, the model parameters change accordingly. Hence, we need a method by which an estimate of the parameter vector $\tilde{\Theta}$ is obtained. In particular, we employ a system where the state vector corresponds to the parameter vector, $\tilde{\Theta}$, the observation corresponds to the measured slowdown, and the system environment includes the bid value. The system transfer function is given by the model functional expression which relates the input to the output, which is non-linear. We employ an extended Kalman filter technique [20, 19] to linearize the transfer function by taking first derivatives.

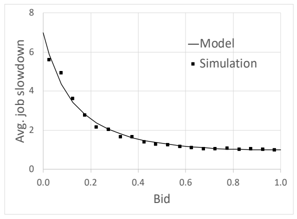
We set the state evolution matrix $\mathbf{F}$ to the identity matrix and the (evolution) covariance matrix $\mathbf{Q}$ to a fraction f of the squared values of the initial state variables. For the (system) covariance matrix $\mathbf{R}$, we use an approximation based on the 95% confidence interval of the t-distribution divided by a factor γ which is a fraction of the actual measurement window in relation to one which yields steady state measurements. We set $f = 5\%$ and $\gamma = 0.5$.

4 Simulations

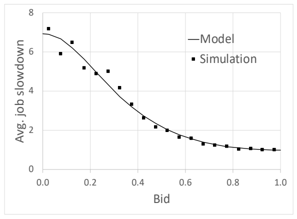
To validate the job slowdown approximation given by Equation 3, we simulate a system with 12 slots over 4,000 secs in steady state. Jobs arrive as a Poisson process with an average rate of 3.2 jobs/sec. The job service time is Gamma distributed with average 1 sec and variance 0.5 sec^2. The requested number of



(a) Linearly decreasing

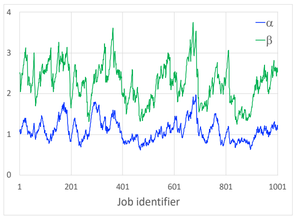


(b) Uniform

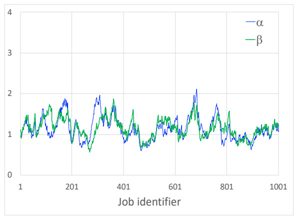


(c) Linearly increasing

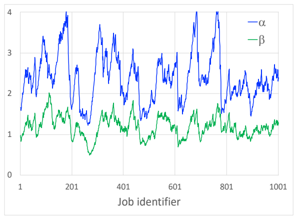
Fig. 3: The average job slowdown versus the bid value with the prediction model and simulations for different bid distributions.



(a) Linearly decreasing



(b) Uniform



(c) Linearly increasing

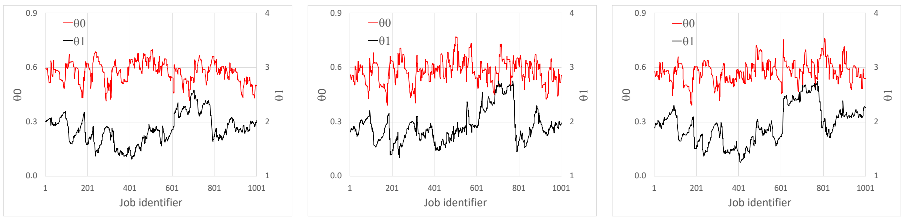
Fig. 4: Estimates of bid distribution parameters for different bid distributions.

slots per job is uniformly distributed between 1 and 5 slots. This constitute an offered load of 80%. As for the bid distribution, we consider three cases: linearly decreasing[6], uniform, and linearly increasing. The density functions $b(x)$ are $2(1-x)$, 1, and $2x$, respectively, $x \in [0, 1]$. And, the distribution functions $B(x)$ are $x(2-x)$, x, and x^2, respectively.

We divide the bid range into 20 bins, each with a width of 0.05, and we calculate the average job slowdown for each such bin. The data points are used in a regression analysis of Equation 3, solving for $[\tilde{\theta}_0, \tilde{\theta}_1]$ that minimizes the mean squared error (mse) between the model and simulation values of the average slowdown. Figure 3 depicts the average slowdown as a function of bid using the model and simulation for three bid distributions. We observe that our model anticipates with high accuracy the simulation results for the entire range of bid values, irrespective of the shape of the bid distribution. The estimates for the three bid distributions were $\tilde{\theta}_0 = 0.64$ and $\tilde{\theta}_1 = 2.44$, with $mse = 3.47 * 10^{-03}$, $\tilde{\theta}_0 = 0.62$ and $\tilde{\theta}_1 = 2.26$, with $mse = 2.56 * 10^{-02}$ and $\tilde{\theta}_0 = 0.62$ and $\tilde{\theta}_1 = 2.27$, with $mse = 3.21 * 10^{-02}$, respectively.

Figure 4 shows the estimates of the three bid distributions. Because our estimator uses samples of the bid distribution, parameters α and β fluctuate around their values, governed by $\alpha/\beta = 0.5$, 1, and 2, respectively.

[6] In practice, users may favor bidding low.



(a) Linearly decreasing (b) Uniform (c) Linearly increasing

Fig. 5: Estimates of model parameters for different bid distributions.

Figure 5 shows the model parameters over time for the three bid distributions. We observe that $\tilde{\theta}_1$ has a catalyzing effect and drops to lower values when $\tilde{\theta}_0$ is overestimated thus adjusting the prediction model.

5 Related Work

Since Amazon EC2 released its spot markets in 2009, a sizable body of research analyzed the operation of such systems in the cloud. The characterization and prediction of spot prices of the AWS spot markets [1] inspired the design of user bidding strategies that optimize cost while also achieving uninterrupted service. Such strategies can be derived either by means of statistical analysis of historical spot prices [8, 18] or through more advanced modeling techniques such as Markov chains [2, 16].

Modeling and predicting the performance of multi-task MapReduce-based applications has been studied in various settings [7, 15]. Common approaches build an estimator by choosing a relationship between an output variable that needs to be predicted and several system properties that can be measured and used for prediction. To provide good predictions, the estimator employs machine learning techniques and needs large amounts of training data based on low-level application performance characteristics [11, 14].

6 Conclusion

We presented a simple, empirical, parametric approximate expression for the job slowdown as a function of bid value for job scheduling in container cloud spot markets. The approximate expression extends the M/M/1 *bribing queue* using two parameters. Further, we provided a methodology for the dynamic estimation of the parameters using the method of moments matching and extended Kalman filtering, a control-theoretic approach. We validated our approximation and prediction methodology using simulation experiments. We incorporated our approximate model in a spot advisor that is employed to either (1) set an expectation for the performance of a job given a particular bid value or (2) suggest a minimum bid value required to attain a given service level.

References

1. O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir. Deconstructing Amazon EC2 Spot Instance Pricing. *ACM Transactions on Economics and Computation*, 1(3):16:1–16:20, 2013.
2. N. Chohan, C. Castillo, M. Spreitzer, M. Steinder, A. N. Tantawi, and C. Krintz. See Spot Run: Using Spot Instances for MapReduce Workflows. *USENIX HotCloud*, 2010.
3. D. Dowson and A. Wragg. Maximum-entropy distributions having prescribed first and second moments (corresp.). *Information Theory, IEEE Transactions on*, 19(5):689–693, 1973.
4. C. Forbes, M. Evans, N. Hastings, and B. Peacock. *Statistical distributions.* John Wiley & Sons, 2011.
5. B. Ghit and D. Epema. Better Safe than Sorry: Grappling with Failures of In-Memory Data Analytics Frameworks. *ACM HPDC*, 2017.
6. M. Harchol-Balter. Open problems in queueing theory inspired by datacenter computing. *Queueing Systems*, pages 1–35, 2021.
7. H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. B. Cetin, and S. Babu. Starfish: A Self-tuning System for Big Data Analytics. *CIDR*, 11(2011):261–272, 2011.
8. B. Javadi, R. K. Thulasiramy, and R. Buyya. Statistical modeling of spot instance prices in public cloud environments. In *IEEE Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*, pages 219–228. IEEE, 2011.
9. L. Kleinrock. Optimum bribing for queue position. *Operations Research*, 15(2):304–318, 1967.
10. H. Liu. Cutting MapReduce Cost with Spot Market. *HotCloud*, 2011.
11. J. Shi, J. Zou, J. Lu, Z. Cao, S. Li, and C. Wang. MRTuner: A Toolkit to Enable Holistic Optimization for MapReduce Jobs. *VLDB Endowment*, 7(13):1319–1330, 2014.
12. D. J. Simon. *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches.* John Wiley and Sons, 2006.
13. M. Unuvar, Y. Doganata, and A. Tantawi. Configuring cloud admission policies under dynamic demand. In *Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), 2013 IEEE 21st International Symposium on*, pages 313–317, Aug 2013.
14. S. Venkataraman, Z. Yang, M. J. Franklin, B. Recht, and I. Stoica. Ernest: Efficient Performance Prediction for Large-Scale Advanced Analytics. *USENIX NSDI*, 2016.
15. A. Verma, L. Cherkasova, and R. H. Campbell. ARIA: Automatic Resource Inference and Allocation for MapReduce Environments. *ACM ICAC*, 2011.
16. M. Zafer, Y. Song, and K.-W. Lee. Optimal Bids for Spot VMs in a Cloud for Deadline Constrained Jobs. *IEEE CLOUD*, 2012.
17. M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, and I. Stoica. Improving MapReduce Performance in Heterogeneous Environments. *USENIX OSDI*, 2008.
18. L. Zheng, C. Joe-Wong, C. W. Tan, M. Chiang, and X. Wang. How to Bid the Cloud. *ACM SIGCOMM Computer Communication Review*, 45(4):71–84, 2015.
19. T. Zheng, M. Woodside, and M. Litoiu. Performance Model Estimation and Tracking Using Optimal Filters. *IEEE Transactions on Software Engineering*, 34(3):391–406, 2008.
20. T. Zheng, J. Yang, M. Woodside, M. Litoiu, and G. Iszlai. Tracking Time-Varying Parameters in Software Systems with Extended Kalman Filters. *IBM Press Centre for Advanced Studies on Collaborative Research*, 2005.