# Cost-Efficient Elastic Stream Processing Using Application-Agnostic Performance Prediction

Shigeru Imai, Stacy Patterson, and Carlos A. Varela
*Department of Computer Science*
*Rensselaer Polytechnic Institute*
$\{$*imais,sep,cvarela*$\}$*@cs.rpi.edu*

***Abstract*—Cloud computing adds great on-demand scalability to stream processing systems with its pay-per-use cost model. However, to promise service level agreements to users while keeping resource allocation cost low is a challenging task due to uncertainties coming from various sources, such as the target application's scalability, future computational demand, and the target cloud infrastructure's performance variability. To deal with these uncertainties, it is essential to create accurate application performance prediction models. In cloud computing, the current state of the art in performance modelling remains application-specific. We propose an application-agnostic performance modeling that is applicable to a wide range of applications. We also propose an extension to probabilistic performance prediction. This paper reports the progress we have made so far.**

***Keywords*-cloud computing; performance prediction; resource allocation**

## I. Introduction

Stream processing has been successfully used with cloud computing in various applications including online advertisement analytics, anomaly detection, and Twitter trend analysis. Cloud computing adds great on-demand scalability to stream processing systems with its pay-per-use cost model. However, to promise service level agreements (SLAs) to users while keeping resource allocation cost low is a challenging task due to uncertainties coming from various sources, such as the target application's scalability, future computational demand, and the target cloud infrastructure's performance variability. The performance of virtual machines (VMs) from public cloud providers wildly varies because of resource contention due to co-located instances [1]. Also, Amazon EC2 reportedly offers different hardware configurations under the same instance type and the performance differs up to 60% [2].

To adaptively scale up and down virtual machines in the cloud, various *auto-scaling* techniques have been proposed [3]. These techniques mostly focus on how to scale virtual machines (VMs) to keep up with fluctuating demand either reactively or proactively; however, cost-aware resource scheduling for elastic stream processing has not been fully explored yet, unlike its batch processing alternatives. Ishii and Suzumura first attempted to explicitly optimize monetary cost for hybrid clouds using linear programming [4]. Han et al. developed a queuing theory based performance model for web services and applied it to find a cost-efficient allocation of servers [5]. Both previous works use *application-specific* performance models created offline to predict performance for streaming applications. Offline models are created from benchmarking the target application against all available VM types. Moreover, since it is application-specific, we need to repeat the benchmarking process every time we test new applications and/or environments. Another way of creating models is a hybrid approach, where we create models offline using a training data set and adapt them online afterwards. The hybrid approach does not need to repeat the benchmarking process, and it has ability to adapt to new test environments. AROMA [6] trains performance prediction models from various Hadoop jobs, and then it selects an appropriate model for a newly submitted job using its resource utilization signature. Hybrid approaches have been explored in the context of database management systems [7], web applications [8], HPC applications [9], [10] and embedded systems [11] as well.

To the best of our knowledge, hybrid approaches used in cloud computing to date are all application-specific; however, if we could create performance models in an *application-agnostic* way, the hybrid approach is potentially applicable to a wide range of applications. Moreover, instead of predicting performance $y$ deterministically by a function $y = f(\mathbf{x})$, we can probabilistically predict it by a distribution $P(y \mid \mathbf{x})$. By adding this probabilistic prediction capability to the prediction model, it can address public cloud computing's inherent performance variability and could lead to a better prediction accuracy. The created models will be used to probabilistically satisfy a user-specified SLA (*e.g.*, latency violation must be at most X %) while keeping the cost as low as possible. Also, to rapidly adapt to unexpected demand changes, resource scheduling algorithms should not use computationally heavy optimization algorithms. This is a completely different requirement from batch processing, where the use of computationally expensive algorithms is meaningful for a long one-shot batch.

In summary, we would like to achieve the following goals for elastic stream processing:

- We develop a hybrid performance model creation method that is application-agnostic. Our approaches only uses general resource usage information, and thus can potentially be applicable to a broad range of cloud applications.

- We further enhance the application-agnostic modelling method by adding probabilistic prediction capability for better prediction accuracy.
- We develop a light-weight resource scheduling algorithm suitable for stream processing and evaluate the monetary cost efficiency.

This paper reports progress we have made so far toward the above goals. The rest of the paper is organized as follows. Section II describes a general framework of cost-efficient elastic stream processing. Section III summarizes our initial VM scheduling heuristic for elastic stream processing and its preliminary results. Section IV describes our proposed application-agnostic performance prediction model creation method. Finally, Section V concludes the paper.

## II. Cost-Efficient Elastic Stream Processing

Figure 1 shows the architecture of the general elastic stream processing system. For streaming systems, throughput is one of the most important performance measures, but too high throughput is not possible or too expensive to achieve. So, the user first needs to set a reasonable maximum throughput $\eta_{\max}$ as an SLA to the system. The Application Monitor monitors the application and collects information such as the number of processing requests in the queue and current throughput. The Resource Monitor monitors application-independent resource usage information such as CPU, memory, disk, and network utilization. The Resource Controller collects information from the Application and Resource Monitors, and make scaling decisions based on predicted future demand from the Time Series Prediction Model and predicted throughput from the Throughput Prediction Model.

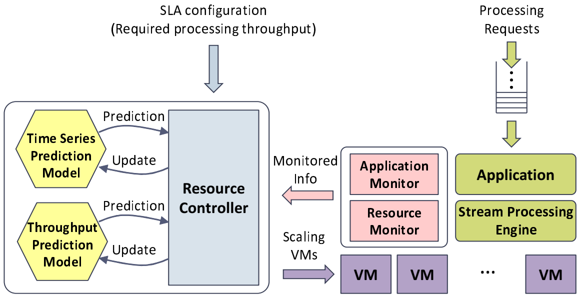


Figure 1. Architecture of the elastic stream processing system

The Resource Controller makes scaling decisions every $\Delta t$ time. At each decision time step $t$, it obtains a set of predicted maximum throughput $\mathcal{H}(t) = \{\hat{\eta}_1(t), \hat{\eta}_2(t), ...., \hat{\eta}_N(t)\}$ for $N$ different VM configurations from the Throughput Prediction Model. $\hat{\eta}_c(t)$ is a predicted throughput at time $t$ using VM configuration $c$. Similarly, the Resource Controller receives a set of predicted demand for the next $T$ time steps $\mathcal{D}(t) = \{\hat{d}(t), \hat{d}(t+\Delta t), ..., \hat{d}(t+T\cdot\Delta t)\}$ from the Throughput Prediction Model. Using $\mathcal{H}(t)$ and $\mathcal{D}(t)$, we can determine the cost minimum configuration $c$ for time $t$ as follows.

$$\min_c COST(c) \text{ s.t. } \min\big(\hat{d}(t),\ \eta_{\max}\big) \le \hat{\eta}_c(t), \quad (1)$$

where $COST(c)$ is a function to compute the cost of VM configuration $c$ when using it until the next decision time step. We can repeat the same operation for the next $T$ time steps to obtain a sequence of optimal configurations $c(t), c(t+\Delta t), ..., c(t+T\cdot\Delta t)$. Public cloud providers such as Amazon EC2 charge one full hour even if actual VM usage is just one minute. Therefore, to reduce cost, we should adjust the sequence of VM configurations accordingly so that useless VM termination and restart can be avoided. This idea is known as *smart kill* [12].

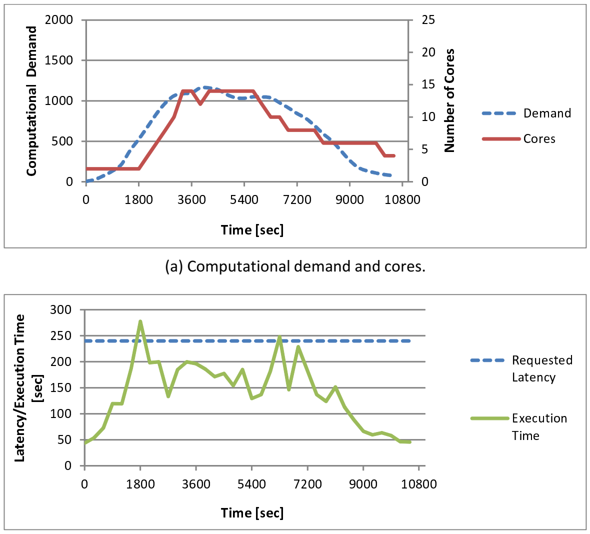
## III. Preliminary Heuristic and Its Evaluation

As an initial attempt to realize cost-efficient elastic stream processing, we implemented an elastic VM scheduling heuristic with future demand time series prediction and VM performance prediction (for details, see [13]). The scheduler receives a request of solving integer linear programming problem every five minutes and is expected to finish it within four minutes. The heuristic is evaluated on Amazon EC2 and is aware of billing cycle of currently allocated VMs so that it does not waste cost and computing power.

The result of an experiment with gradually changing demand is shown in Figure 2. Due to inaccuracy of the demand predictor, execution time exceeded the requested latency at 1800 and 6300 seconds respectively, but overall, it follows demand changes relatively well. Figure 3 shows a VM allocation sequence generated from the heuristic. We can see that the heuristic gradually allocated VMs from c4.large instances, and once it reached the c4.large's allocation limit (five instances by default), allocated a c4.xlarge instance. Table I shows a comparison with Amazon EC2's threshold-based Auto Scaling. Since Auto Scaling only reacted to CPU utilization threshold values, it failed to allocate enough VMs and ended up with higher latency violations than our heuristic.

Table I
VM hours, cost, and latency violations for Amazon EC2's auto scaling and the deterministic elastic scheduling.

| Policy | Cores | VM hours [core·hour] | Cost [USD] | Violations [%] |
|---|---|---|---|---|
| Auto Scaling | 2 to 8 | 15.96 | 0.88 | 25 |
| Elastic Scaling | 2 to 14 | 18.33 | 1.01 | 5.56 |



(a) Computational demand and cores.

(b) Requested latency and execution time.

Figure 2. Preliminary experimental result for deterministic elastic scheduling.

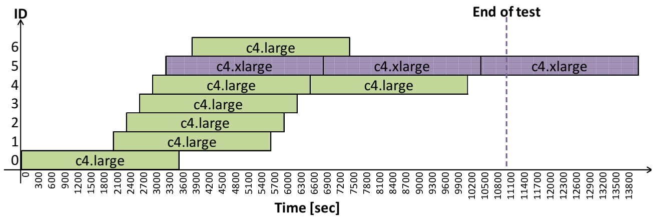


Figure 3. VM allocation sequence for generated from the deterministic elastic scheduling.

## IV. Toward Cost-Efficient, Application-Agnostic Elastic Stream Processing

Accurate application throughput estimation is key to cost-efficiency of elastic stream processing. We plan to develop the application-agnostic performance estimation in two steps, as shown in Figure 4. In Step 1, we plan to achieve throughput prediction deterministically (*i.e.*, $y = f(\mathbf{x})$), and then in Step 2, we plan to further enhance the developed prediction method to generate a probability density function to account for cloud computing's inherent performance variance (*i.e.*, $P(y \mid \mathbf{x})$).

We describe the process of deterministic throughput prediction in Step 1 by following Figure 4. First, in the offline phase, we create an application-agnostic throughput prediction model $g_i(\mathbf{p}, d, c_j)$ for each VM configuration $c_i (i = 1, ..., N)$, where $\mathbf{p}$ is a resource usage profile vector, $d$ is computational demand, and $c_k$ is a target VM configuration. The training set $D_{\text{train}}$ is created in such a way that $g_i$ can return relative throughput between $c_i$ and all other VM configurations other than $c_i$. That is, $g_i(\mathbf{p}, d, c_j)$ gives relative throughput $\eta_j / \eta_i$, where $\eta_j$ and $\eta_k$ are throughput obtained from $c_j$ and $c_k$ respectively. Next, in the online phase, we run the target application on any VM configuration $c_k$ one time and record $\mathbf{p}, d$, and $\eta_k$. When predicting throughput for $c_l$ $(\neq c_k)$, we can estimate the throughput for $c_l$ using the relative throughput estimated by $g_k$ as follows:

$$\hat{\eta_l} = g_k(\mathbf{p}, d, c_l) \cdot \eta_k \tag{2}$$

In Step 2, we apply a corresponding performance variability profile to the deterministic prediction generated from Step 1 and obtain a probability density function of throughput.

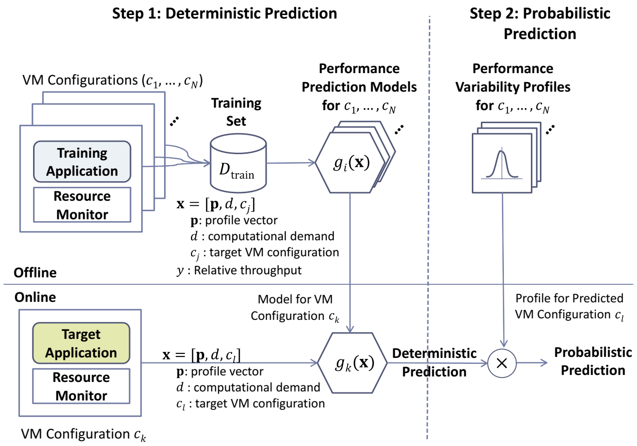

Step 1: Deterministic Prediction
Step 2: Probabilistic Prediction
VM Configurations $(c_1, \dots, c_N)$
Training Set
Performance Prediction Models for $c_1, \dots, c_N$
Performance Variability Profiles for $c_1, \dots, c_N$
Training Application
Resource Monitor
$D_{\text{train}}$
$g_i(\mathbf{x})$
$\mathbf{x} = [\mathbf{p}, d, c_j]$
$\mathbf{p}$: profile vector
$d$ : computational demand
$c_j$ : target VM configuration
$y$ : Relative throughput
Offline
Online
Model for VM Configuration $c_k$
Profile for Predicted VM Configuration $c_l$
Target Application
Resource Monitor
$\mathbf{x} = [\mathbf{p}, d, c_l]$
$\mathbf{p}$: profile vector
$d$ : computational demand
$c_l$ : target VM configuration
$g_k(\mathbf{x})$
Deterministic Prediction
×
Probabilistic Prediction
VM Configuration $c_k$

Figure 4. Overview of application-agnostic throughput prediction.

Contributions and evaluation plans of the proposed approaches are summarized as follows:

- In cloud computing, it is common to create application performance models offline only [14], [15], [4], [16] or in a completely adaptive manner [17]. There are few studies combining offline training and online adaptation for predicting performance of Hadoop jobs [6] and queries for database management systems [7]. Unlike these application-specific approaches, we take an application-agnostic approach, where the performance predictor only uses general resource usage information. Our approach can potentially be applicable to a broad range of cloud applications. We will evaluate several categories of applications and investigate the applicability and limitations of this approach.
- As reported from several studies [1], [2], the performance of VMs varies due to resource contention. There are several studies on cost-efficient resource management that address the performance variance [18], [19]; however, their models are application-specific. Taking the performance variation of VMs into consideration, we will be able to obtain probabilistic performance models for a wide range of applications. By using a probability distribution, we can estimate the per-

formance probabilistically (*e.g.*, X% of chance, the application finishes within Y seconds). This would give the users a chance to more precisely control the performance through an SLA (*e.g.*, latency violation must be at most Z%).
- We will develop a light-weight resource scheduling algorithm suitable for stream processing and evaluate the monetary cost efficiency. We will collect real application resource usage from public cloud service providers and use it with CloudSim [20] for simulation-based evaluation.

## V. Conclusion

We showed a preliminary heuristic for elastic stream processing and described our application-agnostic performance prediction modelling method, which can be widely applicable to many applications. The created models will be used to probabilistically satisfy a user-specified SLA while keeping the cost as low as possible. We plan to first formalize the approach in detail and verify the effectiveness of the approach on CloudSim [20], and then move to real cloud environments.

## Acknowledgments

This research is partially supported by the DDDAS program of the Air Force Office of Scientific Research, Grant No. FA9550-15-1-0214 and NSF Awards, Grant No. 1462342, 1553340, and 1527287. The authors would like to thank an Amazon Web Services educational research grant and a Google Cloud Credits Award.

## References

[1] A. Gandhi, P. Dube, A. Karve, A. Kochut, and H. Ellanti, "The Unobservability Problem in Clouds," in *Cloud and Autonomic Computing, 2015 International Conference on*. IEEE, 2015, pp. 13–20.

[2] Z. Ou, H. Zhuang, J. K. Nurminen, A. Yl-jski, and P. Hui, "Exploiting Hardware Heterogeneity within the Same Instance Type of Amazon EC2," in *in Proc. 4th USENIX Conference on Hot Topics in Cloud Computing*, 2012, p. 4.

[3] T. Lorido-Botran, J. Miguel-Alonso, and J. a. Lozano, "A Review of Auto-scaling Techniques for Elastic Applications in Cloud Environments," *Journal of Grid Computing*, vol. 12, no. 4, pp. 559–592, 2014.

[4] A. Ishii and T. Suzumura, "Elastic Stream Computing with Clouds," *2011 IEEE 4th International Conference on Cloud Computing*, pp. 195–202, 2011.

[5] R. Han, M. M. Ghanem, L. Guo, Y. Guo, and M. Osmond, "Enabling Cost-Aware and Adaptive Elasticity of Multi-Tier Cloud Applications," *Future Generation Computer Systems*, vol. 32, pp. 82–98, 2014.

[6] P. Lama and X. Zhou, "AROMA: Automated Resource Allocation and Configuration of MapReduce Environment in the Cloud," *Proceedings of the 9th international conference on Autonomic computing*, p. 63, 2012.

[7] M. B. Sheikh, U. F. Minhas, O. Z. Khan, A. Aboulnaga, P. Poupart, and D. J. Taylor, "A Bayesian Approach to Online Performance Modeling for Database Appliances Using Gaussian Models," in *Proceedings of the 8th ACM international conference on Autonomic computing*. ACM, 2011, pp. 121–130.

[8] A. Li, X. Zong, S. Kandula, X. Yang, and M. Zhang, "CloudProphet: Towards Application Performance Prediction in Cloud," in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4. ACM, 2011, pp. 426–427.

[9] L. Carrington, M. A. Laurenzano, and A. Tiwari, "Inferring Large-Scale Computation Behavior via Trace Extrapolation," in *Parallel and Distributed Processing Symposium Workshops & PhD Forum, 2013 IEEE 27th International*. IEEE, 2013, pp. 1667–1674.

[10] C. Rosas, J. Giménez, and J. Labarta, "Scalability Prediction For Fundamental Performance Factors," *Supercomputing frontiers and innovations*, vol. 1, no. 2, pp. 4–19, 2014.

[11] X. Zheng, P. Ravikumar, L. K. John, and A. Gerstlauer, "Learning-based analytical cross-platform performance prediction," in *Embedded Computer Systems: Architectures, Modeling, and Simulation, 2015 International Conference on*. IEEE, 2015, pp. 52–59.

[12] J. Kupferman, J. Silverman, P. Jara, and J. Browne, "Scaling into the Cloud," *Tech. rep., University of California, Santa Barbara; CS270-advanced operating systems*, 2009. [Online]. Available: http://www.cs.ucsb.edu/~jbrowne/files/ScalingIntoTheClouds.pdf

[13] S. Imai, S. Patterson, and C. A. Varela, "Elastic Virtual Machine Scheduling for Continuous Air Traffic Optimization," in *16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, May 2016.

[14] S. Imai, T. Chestna, and C. A. Varela, "Accurate Resource Prediction for Hybrid IaaS Clouds Using Workload-Tailored Elastic Compute Units," in *6th IEEE/ACM International Conference on Utility and Cloud Computing*, December 2013.

[15] S. Abrishami, M. Naghibzadeh, and D. H. J. Epema, "Deadline-Constrained Workflow Scheduling Algorithms for Infrastructure as a Service Clouds," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 158–169, 2013.

[16] A. Verma, L. Cherkasova, and R. Campbell, "ARIA: Automatic Resource Inference and Allocation for MapReduce Environments," *8th ACM international conference on Autonomic computing*, pp. 235–244, 2011.

[17] S.-M. Park and M. Humphrey, "Self-Tuning Virtual Machines for Predictable Escience," in *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*. IEEE Computer Society, 2009, pp. 356–363.

[18] D. Poola, S. K. Garg, R. Buyya, Y. Yang, and K. Ramamohanarao, "Robust Scheduling of Scientific Workflows with Deadline and Budget Constraints in Clouds," *2014 IEEE 28th International Conference on Advanced Information Networking and Applications*, pp. 858–865, 2014.

[19] L. C. Canon and E. Jeannot, "Evaluation and Optimization of the Robustness of Dag Schedules in Heterogeneous Environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 4, pp. 532–546, 2010.

[20] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.