Jua : Customer Facing System to Improve User Experience in Public Clouds

Arjun Singhvi
singhvi2@wisc.edu

Shreya Kamath
skamath2@wisc.edu

Shruthi Racha
shruthir@wisc.edu

***Abstract*—Public cloud providers lease virtual machines(VM) on a server to customers. This model makes it easier for cloud providers to manage their infrastructure. However, on the other hand, it presents the customers with difficulty in terms of manually selecting the VM instance type. The inherent performance heterogeneity observed in public clouds further leads to the uncertainty of VM performance observed.**

We propose *Jua*, a customer facing system that solves the above problems. It consists of two main modules - VM Instance Selector and VM placement Gamer. The former performs the task of choosing the appropriate VM instance type adopting a machine learning methodology and the latter module implements a number of placement gaming strategies to ensure that the customers get the best performance relative to the cost they incur. Initial evaluation of *Jua* confirms the viability of using a machine learning based approach for the selection of instance type. Finally, we develop a simulator to model and implement the various placement strategies and evaluate the same using a wide variety of synthetic benchmarks. We achieve a maximum performance improvement of 70%.

1 INTRODUCTION

Public cloud providers use an easy to understand coarse-grained billing model in which customers pay a flat fee per time quantum to use a virtual machine (VM) pre-configured with a bundle of virtualized resources. For example, Amazon's Elastic Compute Cloud (EC2) charges customers hourly and allows customers to choose from a wide variety of VM instance types that mainly differ based on the virtualized resources that they offer [1]. Various other public cloud providers also follow suit [2, 3]. Adopting such an approach for billing makes managing the cloud easier for the cloud providers by limiting the ways in which the cloud resources can be used by customers. However, there are two main shortcomings of the aforementioned billing approach from the point of view of the customers.

Firstly, the customers are presented with a wide range of VM instance types to choose from. Each VM instance type is characterized by a number of low level technical specifications such as CPU, memory, storage and networking capacity. In an effort to provide more flexibility to customers, each VM instance type is normally offered in multiple sizes. As a result, the customers are faced with the non-trivial task of choosing the appropriate VM instance type for their application based on these specifications. The selection process involves customers mapping their high-level application specific service level attributes (HL-SLAs) to a set of low level technical specifications leading to choosing the specific VM instance type. However, the entire mapping process can be a complex task for customers as it is non-intuitive for them to translate their application requirements to low level technical specifications.

Secondly, since the billing model does not allow customers to have a fine-grained control over the physical resources allocated for hosting their VM, not all instances of a given type offer the same performance. The main reason for this performance heterogeneity can be attributed to the fact that data centers contain multiple generations of hardware. VMs of the same type are not guaranteed to be hosted on the same underlying hardware. Extensive prior work [4, 5, 6, 7] in this domain indeed confirms that performance heterogeneity is a commonly observed phenomenon in public clouds today. In [8], the authors state and verify that the three main causes for the observed performance heterogeneity are due to inter-architecture heterogeneity i.e differences due to processor architecture or system configuration, intra-architecture heterogeneity i.e differences within a processor architecture or system configuration and temporal heterogeneity i.e differences within a single machine over time.

As a part of this work, we propose methodologies to solve the above two problems. The former problem is tackled using a module devel-

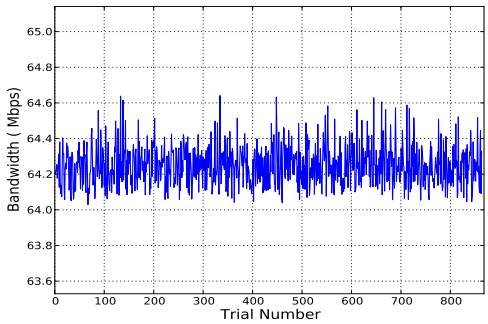

Figure 1: Temporal network performance heterogeneity with microbenchmark iPerf3

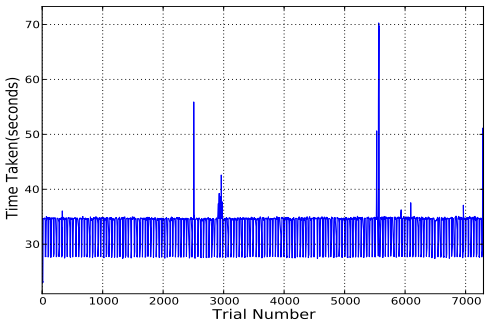

Figure 2: Temporal CPU performance heterogeneity with microbenchmark NQueens

Benchmark	Measured resources
NQueens	CPU
iPerf3	Network bandwidth
ApacheBench	CPU and Network bandwidth

Table 1. Microbenchmarks for EC2 performance evaluation

oped by us that automatically translates application specific HL-SLAs to an appropriate VM instance type. The latter problem is tackled by introducing a number of customer-controlled strategies for selecting instances in order to mitigate the negative impact of the existing performance heterogeneity in public clouds. The proposed strategies are largely based on the strategies mentioned in [8].

Our main contributions can be summarized as follows-

- Verification of the existence of performance heterogeneity in public clouds
- *Jua*, a customer facing system that ensures
  - Automatic VM instance type selection based on HL-SLAs
  - Customers get the best performance relative to the cost they incur by using a number of VM placement strategies

The rest of the paper is organized as follows: Section 2 gives details about the experiments carried out on Amazon's EC2 to verify the existence of performance heterogeneity in public clouds. The details regarding the proposed solution are present in Section 3. Section 4 gives details regarding the extensions incorporated by *Jua* in the context of the various VM placements strategies. We present the key insights from the evaluation of *Jua* in Section 5 and demonstrate a performance improvement upto a maximum of 70%. Section 6 gives details regarding prior work done in the same domain of research. Section 7 lists the future enhancements based on which *Jua* can be extended. Section 8 summarizes the work put forth in this paper.

2 EC2 PERFORMANCE EVALUATION

2.1 Methodology

The main purpose of this performance evaluation was to verify if performance heterogeneity due to the factors mentioned in [8] still exist. We carried out our evaluation in the US-West region of Amazon's EC2. We restricted the scope of our evaluation to investigate whether or not performance variation exists in *t2.micro* instances, which are guaranteed to have a balance of compute, network and memory resources. We aim to extensively study the nature and range of performance variation across the different available instance types in the future.

In order to verify if temporal heterogeneity still exists we use two benchmarks - NQueens and iPerf3 [9]. The former benchmark focuses on benchmarking CPU performance whereas the latter focuses on benchmarking network performance. We used Apache web server [10] to verify if performance heterogeneity exists due to inter-architecture heterogeneity and intra-architecture heterogeneity. Table 1 details the various benchmarks and application used to evaluate the performance of the chosen instance.

2.2 Temporal Performance Heterogeneity

In order to verify the existence of temporal heterogeneity, we ran iPerf3 [9] as well as NQueens for

CPU	Count	Freq [GHz]	Cache [MB]	Mem [GiB]
Intel E5-2670	29	2.50	25	1
Intel E5-2676	21	2.40	30	1

Table 2. Examples of processor heterogeneity in Amazon EC2

Sl. No.	Requests/s	Time per request (ms)	Transfer rate (KBps)
1	1073.85	1.02	12,356.64
2	1042.62	1.16	11,997.29
3	725.54	1.77	8,348.62

Table 3. Performance heterogeneity observed on 3 *t2.micro* instances that use the same processor

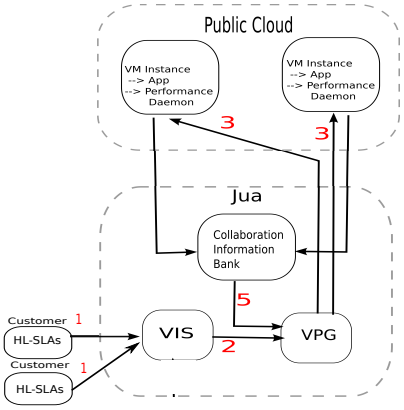

Figure 3: Proposed Design of Jua

24 hours on a *t2.micro* instance and recorded the relevant metrics reported by the benchmarks. As seen from Figures 1 and 2, there is performance variation observed across time - the execution time for the NQueens benchmark varies across multiple runs while the bandwidth reported by iPerf3 also varies. One probable reason for the observed heterogeneity could be due to the presence of competing workloads on the physical machine on which the VM under test is hosted.

2.3 Performance Heterogeneity due to Processor Heterogeneity

As reported in [8], one of the major causes of performance heterogeneity is the presence of different processor architectures in the public cloud. Newer generation processor architectures tend to perform better than the older generation processor architectures due to incorporation of the latest technological performance enhancements in the newer generations. To verify the presence of different processors in Amazon's EC2, we launched a number of instances and recorded the details of the underlying hardware on which the VM is hosted. Table 2 lists out the different processors that we encountered while randomly launching fifty instances of *t2.micro*.

CPU	Requests/s	Time per request (ms)	Transfer rate (KBps)
Intel E5-2670	927.59	1.12	10673.69
Intel E5-2676	947.95	1.05	12792.84

Table 4. Performance heterogeneity observed on 2 *t2.micro* instances that use different processors

In order to verify the presence of intra-architecture and inter-architecture performance heterogeneity, we benchmarked the performance of the Apache web server [10] using ApacheBench(ab) [11]. As seen in Tables 3 and 4, performance heterogeneity due to processor heterogeneity still exists. Probable reasons for the observed variation could be due to different peripherals being associated with the various VM instances as well as the presence of different competing workloads.

3 Design

We propose *Jua*, a customer facing system that resolves the above mentioned issues. As can be seen from Figure 3, the system consists of the following three modules:

- VM Instance Selector (VIS)
- VM Placement Gamer (VPG)
- Collaboration Information Bank (CIB)

3.1 Jua Workflow

We envision a customer to install *Jua* on a Management VM via which *Jua* can appropriately choose and monitor the VMs allocated for the application of the customer while ensuring that the customer gets the best deal with respect to the ease of deciding the instance type as well as performance relative to the cost.

A customer interacting with *Jua*, would specify as input to the system their HL-SLAs which satisfy the requirements of the application they would like to host in the public cloud. For example, a customer who wants to host a web server like Apache can specify the HL-SLAs in terms

of metrics such as requests per second, request concurrency level and so on. The aforementioned group of HL-SLAs are used by the VIS module to predict the appropriate VM instance type that would best meet the customer's requirements. The output of the VIS (i.e VM instance type) is passed as input to the VPG.

Based on the output of the VIS module and the VM placement strategy selected by the customer, *Jua* launches a number of VMs in the public cloud. Post which, *Jua* also ensures that a performance monitoring daemon that monitors the application's performance in terms of the customer specified HL-SLAs, is launched in the VM. At the end of each quantum, these daemons will communicate the measured performance statistics along with the details of the underlying hardware to the centralized CIB. The various strategies of the VPG module use the performance statistics to decide whether a particular VM should be retained for the next quantum or not. The following subsections give a more detailed implementation of the modules of Jua.

3.2 VM Instance Selector

The VIS is responsible for mapping the customer's application needs to the various low level technical specifications and then choosing the appropriate VM instance type. This module takes in as input the customer's application requirements in terms of a vector of HL-SLAs and outputs a specific VM instance type.

A machine learning based approach is used to solve the task of automatic instance type selection based on the customer's HL-SLAs. Internally, the VIS consists of a machine learning based classifier per application type that does the required mapping task for customer application level requirements to VM instance type. All applications that have the same type of HL-SLAs are assumed to be of the same application type. Based on the customer's HL-SLAs, the appropriate classifier is chosen to do the required mapping. Each of the classifier is trained with data that has the required mapping for the entire range of permissible values of the HL-SLAs vector. The training data as described above can be collected based on the decisions taken by the customer or by launching a benchmarking phase to collect the required data. More details about the latter are given in the evaluation section.

3.3 VM Placement Gamer

This module is responsible for ensuring that the customers get the best performance relative to the cost they incur while mitigating the performance heterogeneity observed due to the various factors mentioned earlier. Internally, the VPG implements a number of VM placement gaming strategies with the objective of maximizing performance relative to cost. Currently, *Jua* supports and extends all the strategies proposed in [8]. This was achieved by ensuring that *Jua* has the capability of launching performance monitoring daemons on each of the customer's VMs. The VPG in-turn decides the next step to be taken, based on the strategy chosen by the customer as well as the metrics it has aggregated over the last quantum from the various daemons running on the VMs of the customer.

The VPG leverages the fact that cloud providers allow customers to launch instances programmatically using the the provider's Application Programming Interface(API) as well as return VMs at the end of each quantum. However, the task of the VPG is harder than it might seem primarily due to the fact that customers have a coarse-grained control over the physical hardware on which the VM instances are launched, i.e, the customers cannot select the underlying physical hardware on which on the VM should be hosted.

3.4 Collaboration Information Bank

The collaboration information bank is a centralized repository of performance measurements that enable customers to simultaneously learn about the distribution of VMs in the public cloud along with ascertaining the performance difference between the various instance types by taking the underlying hardware into consideration. Performance monitoring daemons running on the customer's VMs will feed back to the CIB the details of the hardware on which their VM is running, along with the measured values of the HL-SLAs at the end of each quantum.

The details regarding how the various strategies of VPG leverage the CIB are described in the next section.

4 Core VM Placement Strategies:

4.1 Background

A placement strategy embodies a trade-off between exploration and exploitation. The basis of exploitation based strategies is to continue using a particular VM that gives the required performance. On the other hand, the basis of exploitation startegies is to launch new VM instances with the hope of discovering a better performing VM. Prior work [8] in this domain have considered a restricted space of placement strategies called (A,B)-strategies. These run at least A servers in every quantum and launch an additional B "exploratory" instances for one quantum each at some point during the job execution. Two general mechanisms useful in building (A, B)-strategies:

- Up-front exploration: aims to find high performing VMs early so that they can be used for a longer duration. An (A, B)-strategy that uses up-front exploration launches all B "exploratory" instances at the start of a job. At the end of the first quantum, the highest performing A instances are retained and the remaining B instances are shut down.
- Opportunistic replacement: By migrating an instance (shutting it down and replacing it by a new one to continue its work) we can seek out better performing instances or adapt to slowdowns in performance. An (A, B)-strategy that uses opportunistic replacement will retain any VM that is deemed a high performer for a given time quantum and migrate any VM that is a low performer.

In the "CPU" strategy the high performing instances are determined based on their processor type. On the other hand, in the "PERF" strategy the high performing instances are determined based on VM performance that is measured. The above strategies are extended to perform opportunistic replacement by launching additional VMs in each time quantum in order to find a better performing VM instance. The extensions are abbrebriviated as "CPU_OPREP" (CPU with opportunistic replacement) and "PERF_OPREP" (PERF with opportunistic replacement).

4.2 Customer Collaboration

The strategies based on opportunistic replacement proposed in [8] have been extended to leverage the performance statistics present in the CIB. Specifically, when the collaboration based strategies are being executed by the VPG module, the module has the ability to kill an allocated VM immediately if the reported performance from the CIB does not seem promising and launches a new VM instead. However, the proposed strategy can have a negative impact on the customer if the strategy keeps preemptively killing VMs and fails to find a better VM. Reason being, the customer would be charged even for the VMs that are being killed preemptively by the VPG module. Thus, there is a need for a mechanism by which customers can control the aggressiveness of the collaboration based strategies so that the customers do not incur additional costs. The aggressiveness of the strategies can be controlled manually by customers by setting two parameters defined by us - *thrift index* and *preemptive index*.

Thrift index is defined as the minimum probability required for finding a VM that will be hosted on the underlying hardware for which best performance has been reported by other customers using the CIB. A lower thrift index implies that the customer is ready to take a higher risk in anticipation for a better performing VM. On the other hand, a higher thrift index corresponds to a less aggressive strategy as the preemptive killing of VMs would kick in only if there is a high probability of finding a better performing VM.

Preemptive index is defined as the maximum number of VMs that can be killed preemptively per quantum in the hope of finding a better performing VM. A higher preemptive index would imply a more aggressive strategy whereas a lower value would imply a relatively conservative strategy.

By incorporating the above two parameters *Jua* gives a customer the flexibility to control the aggressiveness of the collaboration based strategies. Currently, the customers have to manually set the value of these parameters based on their economic constraints. We plan to come up with a module in the future that would automatically tune these parameters based on a customer's desired monetary budget.

4.3 Application-Level Metrics Monitoring Support

Jua allows customers to specify multiple application specific HL-SLAs that need to be satisfied by VMs hosting the application. The strategies implemented as a part of VPG take all the metrics into account and make the scheduling decisions. *Jua* achieves support for the same by launching measurement daemons that give statistics corresponding to the HL-SLAs on every active VM. The main advantage of using HL-SLAs is that the customers can immediately relate to the decision taken by the strategy in action as opposed to a scenario wherein the strategy bases its decisions on lower-level metrics such as CPU performance, bandwidth usage etc.

5 EVALUATION

This section gives details regarding the evaluation of the two main modules of Jua - VIS and VPG.

5.1 VM Instance Selector

5.1.1 Methodology

In order to evaluate the effectiveness of the VIS, we took the example of web servers as an application type. The customer's HL-SLAs considered corresponding to web servers are: time(s), requests per second, time per request(ms) and transfer rate (KBps). The VIS based on these HL-SLAs selects the appropriate classifier that would choose the appropriate VM instance type. The effectiveness of a machine learning based classifier depends on the data used to train the classifier. The collection of training data involved launching three web servers - Apache [10], NGINX [12] and Lighttpd [13], and benchmarking their performance using ApacheBench requesting for a file of size 100 MB. ApacheBench reported the results of benchmarking in terms of the HL-SLAs selected.

More specifically, the three web servers were launched on four different VM instance types - *t2.nano, t2.micro, t2.small and t2.large*. Using ApacheBench, we obtained a corpus of data with which the accuracy of the VIS could be measured.

5.1.2 Accuracy

In order to select the best performing classifier, we evaluated the classification performance of the following machine learning models: Random Forest(RF), Multinomial Naive Bayes(MNB), Decision Tree(DT), Linear Support Vector Cluster(LSVC), and Support Vector Cluster(SVC). The training data consists of data gotten from benchmarking Apache and NGINX web servers whereas the data collected from Lighttpd was used as the testing data. From Figure 4, we can see that Random Forest gives the best accuracy of 80% in comparison to the other models considered.

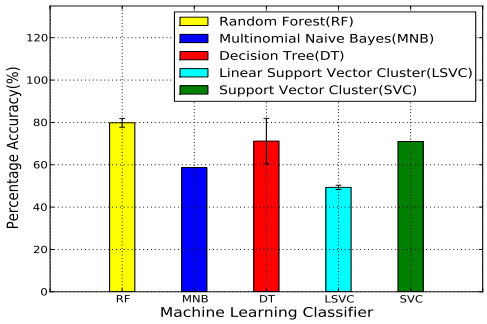

Figure 4: Machine Learning Accuracy

5.2 VM Placement Gamer

5.2.1 Methodology

In order to evaluate the effectiveness of the various strategies used in VPG, we developed a simulator to quickly evaluate the performance of these strategies under different cloud provider configurations. A cloud provider configuration basically indicates the distribution fraction of the different kinds of machines along with their expected performance in terms of a mean rate and the associated standard deviation.

At the end of each quantum, when a VM needs to be chosen, the simulator chooses an instance randomly using the weights corresponding to the distribution fraction of the various machines. The VM's per-quantum performance is selected as an independent normal random variable with the mean and standard deviation of the VM in question. The input to the simulator consists of

- S - strategy to run
- T - total simulator execution time
- q - time quantum(in seconds)
- A - number of machines required to run the job

- B - number of extra machines that can be used by the placement gamer
- p - migration penalty (in seconds)
- tIn - *thrift index*
- pIn - *preemptive index*

The simulator calculates the total work done by each strategy in the same way as proposed in [8]. The relative speedup of each strategy is calculated by comparing its performance to the null strategy in which the required number of VMs are randomly chosen at the beginning of the first quantum and used till the end of the job.

5.2.2 Synthetic Simulations

We have used three different synthetic cloud provider configurations corresponding to the three features of the cloud environment that play an important role in determining the effectiveness of the various placement strategies. The first synthetic simulation evaluates the performance of the strategies when the difference in performance between machines is varied (architecture variation). The next synthetic simulation examines the impact of performance variability observed by the customers (instance variation) on the various strategies. Lastly, we examine the behavior of the various strategies when the distribution of the different performing machines in the cloud is varied (architecture mix).

To limit the scope of our parameter search, we fixed the value of T to 24, A to 30 and B to 20 to indicate a workload that ran for 24 hours with 10 VMs computing at all times with the flexibility of launching an additional 20 ”exploratory” VMs during the job run time for. For simplicity, we have assumed the public cloud to consist of only two types of machines, ”good” machines and ”bad” machines with an even likelihood of each, unless mentioned otherwise.

As stated earlier, the effectiveness of the collaboration strategies depends on the values of the thrift index as well as the preemptive index. We performed a parameter sweep of these two parameters by varying the former from 0.1 to 1.0 and the latter from 1 to 10, and recording the performance of the strategies in each case. The preexisting strategies have been compared with the collaboration strategies when the two parameters have been set to maximize the performance.

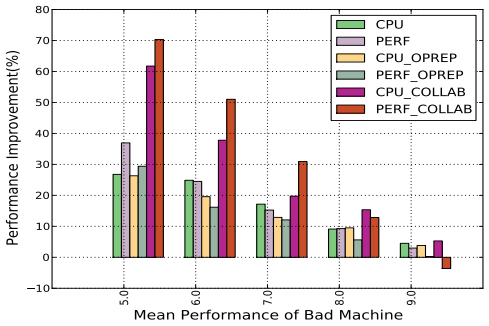

Figure 5: Observe performance improvement for the simulation of architecture variation

Mean Performance of Bad Machine	Thrift Index	pIn_PERF	pIn_CPU
5.0	0.25	9	10
6.0	0.25	3	10
7.0	0.25	8	10
8.0	0.25	2	7
9.0	0.25	8	6

Table 5. Optimal thrift index(tIn) and preemptive index(pIn) for simulation of varying architecture

Architecture variation. In this simulation, we hold the performance variability (standard deviation) steady at 5% of the average performance (a number chosen to match the variation reported in [8]) and evenly spread machines between the two distributions. The performance gap between “good” machines and “bad” machines was varied from 10% to 50%.

As seen in Figure 5, the collaboration based strategies (CPU_COLLAB and PERF_COLLAB) consistently outperform the ones void of collaboration. As the gap between the ”good” and ”bad” machines increases the performance benefits achieved from collaboration also increase. This behavior is consistent with our expectation. Reason being, the collaboration based strategies strive to get hold of the best performing VM and when the gap is maximum, there is more benefit to migrate to a ”good” machine from a relatively ”bad” machine. The values for the thrift index and preemptive index corresponding to the above measurements have been presented in Table 5.

It mush be noted that the collaboration strategies give little or no benefit when there is no performance gap between the different machines.

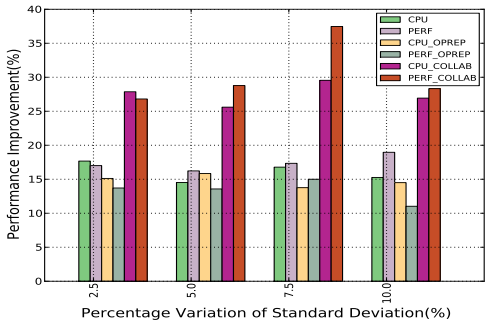

Figure 6: Observe performance improvement for the simulation of instance variation

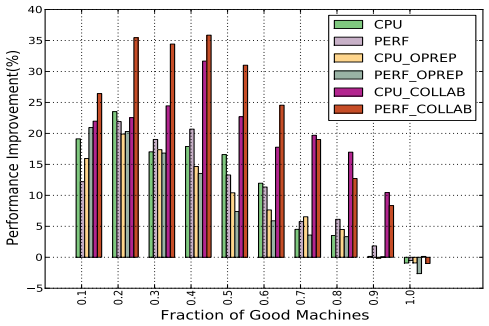

Figure 7: Observe performance improvement for the simulation of architecture mix

This is due to the fact that in such a scenario the benefit of running on a better VM is outweighed by the migration cost.

Instance variation. We then proceeded to evaluate the collaboration based strategies on instances with temporal heterogeneity. For this we enforced variability within a machine's performance by holding steady the average performance difference between "good" and "bad" instances at 30% (again, a number reported in [8]) and instead varying the standard deviation of performance within that instance type. We vary the standard deviation from 2.5% up to 10%.

As seen in Figure 6, the collaboration based strategies continue to outperform the other strategies as the performance variability is varied. The minimum percentage improvement across all values of performance variability is 26% (when variability is 2.5%). That said, even such a modest improvement can ensure that customers get a better value for the cost they incur. However, as one can notice, the degree of variability does not have a significant impact on the performance of the collaboration based strategies. This is due to the fact that the performance improvement achieved by using these strategies largely depends on the benefit of migrating to a "good" machine and the availability of "good" machines. For the above mentioned performance measurements, the thrift index was set to 0.25 and the preemptive index was set to 2 as this combination corresponds to when maximum improvements were noticed.

Architecture Mix. Lastly, we look at how the distribution of machines affect each strategy in order to determine how each strategy fares when the cards are stacked in its favor, or against it. The performance gap between "good" and "bad" machines is set to 30% and the performance variability is fixed at 5%.

Mean of good machines	Thrift index	pIn_PERF	pIn_CPU
0.1	0.05	2	1
0.2	0.1	4	4
0.3	0.15	4	10
0.4	0.2	4	2
0.5	0.25	6	10
0.6	0.25	5	5
0.7	0.35	10	4
0.8	0.4	8	6
0.9	0.45	3	6
1.0	0.5	3	7

Table 6. Optimal thrift index(tIn) and preemptive index(pIn) for simulation of architecture mix

As seen from Figure 7, we continue to observe the same trend of the collaboration based strategies outperforming the other ones. When the fraction of "good" machines is 1.0 which basically implies that all the machines in the public cloud offer the same performance, none of the strategies give a performance improvement as the customers will incur unnecessary migration costs while the performance they get remains the same. The values for thrift index and preemptive index corresponding to the above performance measurements have been presented in Table 6.

Another trend observed is that initially when the fraction of good machines is less, PERF_COLLAB outperforms CPU_COLLAB. However, when the fraction of good machines is high (70% onwards), the CPU_COLLAB performs better as the strategy makes decisions

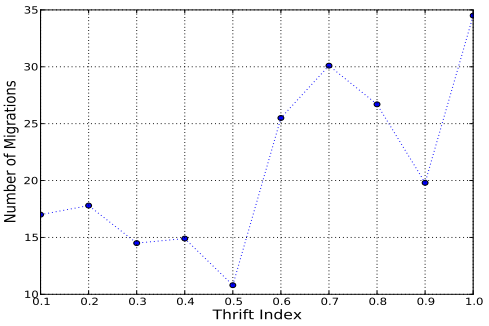

Figure 8: Variation of thrift index(tIn) for a fixed distribution(50%) of good machines

to preemptively kill VMs based on the CPU type and not the performance observed in the previous quantum. Preemptive killing of VMs based on CPU types has greater benefits as compared to PERF_COLLAB that does preemptive killing only if it observes a drop in performance, when the fraction of finding the "good" machine is high. Reason being, the latter may stick to a comparatively less performing machine if there is no significant drop in the observed performance. On the other hand CPU_COLLAB always strives for the best of the available machines.

From the above evaluation of *Jua* in exhaustive simulated cloud distribution scenarios it is evident that the proposed collaboration based strategies outperform the strategies mentioned in [8]. A maximum performance improvement of 70% is achieved.

5.2.3 Customer Collaboration

As stated before, the effectiveness of the collaboration based strategies depends on the values of the thrift index as well as the preemptive index set by the customer. The main idea behind these strategies is to get a VM whose underlying hardware has given the best performance to previous customers. In order to see the overhead involved in the these strategies, we carried out a simple experiment to see the number of migrations done when the collaboration strategy kicks in as compared to when it is not in operation.

For this experiment, the probability of finding a "good" machine was set to be equal to the probability of finding a "bad" machine, i.e, the fraction of each type in the public cloud was 0.5. The preemptive index was set to 20 and the thrift index was varied from 0.1 to 1.0. In such a setting, preemptive killing of VMs is observed only when the probability of finding a "good" machine was greater than or equal to the customer specified thrift index. On the other hand, no preemptive killing of VMs is observed when the thrift index is higher than the probability of finding a "good" machine. In such a scenario, the strategy just falls back to the default strategy without any leveraging of the CIB.

In terms of the overhead of the collaboration based strategies, it can be seen from Figure 8 that the number of migrations are far lesser when collaboration based strategies are operational (when thrift index lies between 0.1 and 0.5) in comparison to the migrations observed when the default strategy is operational (when the thrift index in greater than 0.5). This simple experiment brings out the fact that the collaboration based strategies perform better when compared to the other strategies and also provide this improvement at a far lower cost due to the reduced number of migrations involved. The main reason for the reduced number of migrations is due to the fact that the collaboration based strategies preemptively keep migrating until the best of the available VMs is assigned and post that it does not carry out any migrations.

6 RELATED WORK

Public cloud providers present their clients with a user-friendly billing model and a pre-defined set of minimal primitives in-order to have autonomous control over remote virtual machines(VMs). Due to inter-architecture heterogeneity, intra-architecture heterogeneity and temporal heterogeneity, VM instances of the same configuration in a public cloud environment do not guarantee identical performance. Extensive prior work[14, 15, 16, 17, 18] in this domain have addressed this as a resource scheduling problem among virtual machines(VM) in the cloud. Recent works[8, 19, 20] have fueled research interest to guarantee homogeneous performance across identical VM instances by incorporating scheduling mechanisms on client side.

Paragon [15] uses collaborative filtering techniques to quickly and accurately classify an unknown incoming application, by identifying similarity to previously scheduled applications. Quasar

[16] on the other hand does not solely rely on user specified resource reservations as that can lead to under-utilization due to the fact that users are oblivious to the actual workload requirements. Instead Quasar uses classification techniques tailored to application categories to perform resource allocation and assignment. The VIS in *Jua* draws inspiration from the above two ideologies and leverages application specific machine learning classifiers to map user defined service level agreements(SLA) to a VM instance in the public cloud.

VM placement gaming policies proposed in [8] use exploratory and exploitative placement strategies. These strategies choose the next candidate VM based on either the CPU performance or the system performance. The aim is to provide the best user experience given a customer's SLA. However the work in [8] supported a single SLA and no customer collaboration support. On the other hand, *Jua*, supports multiple application centric SLAs. In addition it also provides customers the flexibility of collaborating among themselves in order to obtain the best performance relative to the cost they incur.

ClouDiA [19] is a client side deployment advisor that selects application node deployments which minimize either largest latency between application nodes or the longest critical path among all application nodes. Their search deployment selects and allocates an instance VM based on the nature of the path between the VM and the application node. *Jua*, on the other hand selects an instance VM based on the customer satisfaction quotient, i.e. the number of customer provided HL-SLAs which are fulfilled by an instance.

7 FUTURE WORK

Though Jua has been able to ensure customers enjoy the benefits of considerable amount of performance improvements, there are a number of aspects of Jua that can be extended to ensure far more benefits to the customer. Jua can be enhanced through the following extensions -

- **Automatic selection of thrift index as well as preemptive index** - Currently, the customers are required to manually set these two parameters. However, as seen in the previous section the performance benefits achieved heavily depends on the values of these two parameters. We plan to have a module that would perform the task of setting the optimal values of these parameters based on the customer's cost budget.
- **VIS correction strategy** - Currently, Jua assumes that the instance type chosen by VIS is always right. However, it may so happen that the instance chosen by the VIS is wrong. We plan to introduce a new strategy in the VPG that would validate the correctness of the VIS selection and may even go to the extent of rectifying the mistake of VIS.

8 CONCLUSION

At present, customers get a raw deal when it comes to using VMs in public clouds. Firstly, the customers need to perform the non-trivial task of choosing the VM instance type manually. Secondly, the customers experience performance heterogeneity due to the fact that they cannot control the underlying physical hardware on which the VMs are hosted. We propose *Jua*, a customer facing system to solve the above problems. Initial evaluation of *Jua* confirms the viability of using a machine learning approach to select a VM instance. Also, the evaluation of *Jua* using a simulator shows a maximum performance improvement of 70% for the proposed collaboration based strategies. This brings out the fact that the proposed extensions to the placement strategies are indeed beneficial.

ACKNOWLEDGEMENT

We would like to thank Prof. Michael Swift for his guidance in helping us complete this project.

References

[1] Amazon Web Services. Amazon EC2 instance types. http://aws.amazon.com/ec2/instance-types/

[2] Microsoft Corp. Windows azure: Pricing details. http: //www.windowsazure.com/en-us/pricing/details/

[3] Rackspace Inc. How we price cloud servers.http://www.rackspace.com/cloud/cloud_hosting_products/servers/pricing

[4] El-Khamra, Yaakoub, Hyunjoo Kim, Shantenu Jha, and Manish Parashar. "Exploring the performance fluctuations of hpc workloads on clouds." *In Cloud Computing Technology and Science (CloudCom)*, pp. 383-387. IEEE, 2010.

[5] Dave Mangot. EC2 variability: The numbers revealed. http://tech.mangot.com/roller/dave/entry/ec2_variability_

the_numbers_revealed, May 2009.

[6] Zhonghong Ou, Hao Zhuang, Jukka K. Nurminen, Antti Ylä-Jääski, and Pan Hui. "Exploiting hardware heterogeneity within the same instance type of amazon EC2". *In 4th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*, 2012.

[7] Schad, J., Dittrich, J. and Quiané-Ruiz, J.A. "Runtime measurements in the cloud: observing, analyzing, and reducing variance". *Proceedings of the VLDB Endowment*, 3(1-2), pp.460-471. VLDB, 2010.

[8] Benjamin Farley, Ari Juels, Venkatanathan Varadarajan, Thomas Ristenpart, Kevin D. Bowers, Michael M. Swift, "More for your money: exploiting performance heterogeneity in public clouds", *In Proc. of the 3rd ACM Symp. on Cloud Computing*, 2012.

[9] A. Tirumala, F. Qin, J Dugan, J. Ferguson, and K. Gibbs. Iperf: The TCP/UDP bandwidth measurement tool, version 2.0.5. http://sourceforge.net/projects/iperf/, 2010.

[10] The Apache HTTP server. https://httpd.apache.org/

[11] The Apache HTTP server Benchmarking (ab) tool. https://httpd.apache.org/docs/2.4/programs/ab.html

[12] The Nginx Web Server https://www.nginx.com/

[13] The Lighttpd Web Serverhttps://www.lighttpd.net/

[14] David Lo , Liqun Cheng , Rama Govindaraju , Parthasarathy Ranganathan , Christos Kozyrakis, "Heracles: improving resource efficiency at scale", *Proceedings of the 42nd Annual International Symposium on Computer Architecture*, June 13-17, 2015, Portland, Oregon.

[15] C. Delimitrou and C. Kozyrakis. "Paragon: QoS-Aware Scheduling for Heterogeneous Datacenters". *In Proc. of the 18th intl. conf. on Architectural Support for Programming Languages and Operating Systems*, 2013.

[16] C. Delimitrou and C. Kozyrakis. "Quasar: Resource-Efficient and QoS-Aware Cluster Management". *In Proc. of the 19th intl. conf. on Architectural Support for Programming Languages and Operating Systems*, 2014.

[17] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, and A. Goldberg. *Quincy: fair scheduling for distributed computing clusters*. In Proc. of the 22nd Symp. on Operating System Principles, 2009.

[18] M. Schwarzkopf, A. Konwinski, M. Abd-El-Malek, and J. Wilkes. "Omega: flexible, scalable schedulers for large compute clusters". *In Proc. of the EuroSys Conf.*, 2013.

[19] Tao Zou , Ronan Le Bras , Marcos Vaz Salles , Alan Demers , Johannes Gehrke. "ClouDiA: A Deployment Advisor for Public Clouds". *In Proc. of the VLDB Endowment*, Vol. 6, No. 2,2012

[20] D. Novakovic, N. Vasic, S. Novakovic, D. Kostic, and R. Bianchini. "DeepDive: Transparently Identifying and Managing Performance in Virtualized Environments". *In Proc. of the USENIX Annual Technical Conference*, 2013.