Performance Prediction for Unseen Virtual Machines

John O'Loughlin and Lee Gillam
Department of Computing, University of Surrey, Guildford, GU2 7XH Surrey, U.K.

Keywords: Cloud Computing, Performance Prediction, Virtualisation, Scheduling.

Abstract: Various papers have reported on the differential performance of virtual machine instances of the same type, and same supposed performance rating, in Public Infrastructure Clouds. It has been established that instance performance is determined in large part by the underlying hardware, and performance variation is due to the heterogeneous nature of large and growing Clouds. Currently, customers have limited ability to request performance levels, and can only identify the physical CPU backing an instance, and so associate CPU models with expected performance levels, once resources have been obtained. Little progress has been made to predict likely performance for instances on such Public Clouds. In this paper, we demonstrate how such performance predictions could be provided for, predicated on knowledge derived empirically from one common Public Infrastructure Cloud.

SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

Infrastructure as a Service (IaaS) Clouds offer resources such as object storage, attachable storage, configurable networks and virtual machines (VMs) to customers over the internet, and are available to anybody with an internet connection and a credit card. Resources are provided on-demand, meaning they are provisioned at the time of the request, and can be 'terminated' when no longer needed. Together with an 'illusion of infinite capacity' (Armbrust et al., 2009) publically available computing environments allow customers to grow and shrink their usage as required and when required, and customers can be charged, typically, on a per-hour basis for resources used. The provider's ability to support this is referred to as elasticity; it is this that differentiates Clouds from dedicated hosting environments where servers can take up to a day to provision, are charged on a per-month basis, and require customers to enter into contracts for a pre-specified period of time irrespective of usage.

Infrastructure Clouds are the first successful attempt at public delivery of compute resources, and naturally there are concerns about migrating on-premise systems to them. Such concerns include: security, availability, vendor lock-in and performance; this latter concern of performance, in particular compute performance, is the focus of this paper.

As the largest, and arguably the de facto standard, of Infrastructure Clouds, performance of instances on EC2 has been widely studied, although mainly for its suitability for HPC and scientific computing. Consequently, differential performance of instances of the same type has been reported on by a number of authors, (Phillips, Engen, and Papay, 2011, and Ou et al., 2012) and it is now established that performance variation is due to the heterogeneous nature of large and growing Clouds. Although EC2 acknowledges heterogeneity, claiming that instances of the same type should deliver a 'consistent and predictable' level of compute (Amazon EC2 FAQS, no date) – despite the published evidence to the contrary– is certainly not ideal.

As users scale their infrastructure, knowing how many instances they should start in order to meet their needs requires an understanding of the deliverable performance. At present users are, potentially, in the unsatisfactory position of either (1) having to guess how many instances to start in order to obtain the desired number of satisfactory ones, with obvious cost implications, or (2) obtaining a number of instances that might not deliver the required performance. EC2 customers who are building services for their own customers on top of these instances may run the risk of failing any SLAs they provide unless they undertake rigorous performance evaluations themselves.

O'Loughlin J. and Gillam L..
Performance Prediction for Unseen Virtual Machines.
DOI: 10.5220/0004840900700077
In *Proceedings of the 4th International Conference on Cloud Computing and Services Science* (CLOSER-2014), pages 70-77
ISBN: 978-989-758-019-2
Copyright © 2014 SCITEPRESS (Science and Technology Publications, Lda.)

In this paper, we consider the problem of determining the expected performance of a set of instances before they are provisioned. The rest of this paper is structured as follows: Section 2 provides background information on Cloud structure, and in section 3 we discuss CPU model distribution in a number of AZs as is relevant to the discussions in this paper. In section 4 we describe our AZ model, and Sections 5-7 discusses simulations of VM offerings. In section 8 we compare our simulated data with real data from Amazon's "us-west-1c" zone. In section 9 we discuss assumptions made in our simulations and how this relates to other scheduling algorithms one could use. In section 10 we review related work, and section 11 concludes and discusses future work.

2 CLOUD INFRASTRUCTURE

EC2, Google Compute Engine (GCE), HP Public Cloud and Rackspace OpenCloud (AWS Global Infrastructure, no date, Google Cloud Platform, no date, HP Public Cloud, no date and Rackspace Global Infrastructure, no date) structure their Clouds into Regions and Zones; and all define these terms in essentially the same manner: Regions are dispersed into geographic areas consisting of Zones, and each Zone is presented as an isolated location. Zones have separate power and network connections and are tied together with high speed interconnects. Consequently, failure of one Zone should not, in theory, disrupt operations in another. EC2 refer to Zones as AZs and we will use these terms interchangeably.

There exist some operational differences between Regions/Zones in different Clouds. For example, EC2 will not automatically replicate any resources placed in one Region into another, whilst GCE state that they make 'no guarantee that project data at rest is kept only in that Region'. However, for the purposes of our present discussion these operational differences are not important.

Clouds offer VMs in a range of sizes known as instance types, with similar types grouped into instance classes. Instances types within the same class typically have the same ratio of quantity of CPU cores to RAM.

Providers give their instance types a performance rating, which expresses the compute power an instance's vCPUs should deliver, with a vCPU rating multiplied by the number of vCPUs offered to provide an overall rating for the type. On Amazon's Elastic Compute Cloud (EC2), this is called the EC2 Compute Unit (ECU), whilst Google Compute Engine has the Google Compute Engine Unit (GCEU). Such ratings should guarantee, if not identical, then certainly very similar, levels of performance of instances of the same type.

However, we have previously shown (Author1 and Author2, 2013) that m1.small instances on EC2 backed by a Xeon E5430 will run a bzip2 compression benchmark on average in 445s, whilst instances backed by a Xeon E5507 will take on average 621s to run the same benchmark. This is a 39% increase in time taken for instances backed by Xeon E5507 compared to the Xeon E5430. These instances are 'rated' the same and the user is charged the same price for each, but a performance-based pricing would clearly be expected to distinguish these.

By a heterogeneous Cloud we mean one where instances of the same type may run a variety of underlying hardware (which we refer to as hosts). Of the providers considered, EC2 is the only one which acknowledges a heterogeneous infrastructure, stating: 'EC2 is built on commodity hardware, over time there may be several different types of physical hardware underlying EC2 instances' (Amazon EC2 FAQs, no date). In such an environment performance variation is to be expected. As Amazon adds new Regions into EC2, and new Availability Zones (AZ) into existing Regions performs hardware refreshes, and as CPU models are retired by manufacturers, heterogeneity is seemingly inevitable. One would assume that, over time, any successful Cloud provider is will end up with a heterogeneous environment.

We have however found a stable association between hardware, as identified by CPU model, and instance classes. That is, an instance of a given class only runs on particular set of hardware. For example, First Generation Standard (FGS) instances run on hardware with Intel Xeon E5430, E5-2650, E5645 and E5507 CPUs. Over the 6 month period, in which we have been running various performance experiments on EC2 (April to September 2013), these sets and associations have not changed.

We would of course expect change over time, for the reasons discussed above. Indeed, in Ou, et al. (2012), the authors find Intel Xeon E5430, E5645, E5507 and AMD Opteron 270 backing FGS instances. They note that the AMD model is present less often in 2012 than compared to results they obtain in 2011. We also find that the hardware associated to different instance classes is distinct from each other. We would assume that hardware associated to different classes has either different

performance characteristics, or a different ratio of CPU cores to RAM. This allows instances from different classes to be differentiated along these lines. Thus, in addition to FGS instance class we have High CPU, High Memory and Second Generation Standard Instance classes. We may logically view AZs as being partitioned into hardware platforms associated with instance classes that are available there.

Amazon is currently the only provider to publish specific CPU model information associated with certain instance types. Generally these are specialised high-performance instance types such as Cluster Compute with 10Gb networking, or GPU instances. These types are only available, at the time of writing, in a limited number of Regions. The recently introduced High Memory class (m3.xlarge and m3.2xlarge) is the only general purpose class of instances globally available on EC2 whereby the CPU model is provided as part of the instance description. Whether EC2 will continue this trend; and specify CPU model for all new, and possibly existing, types, or find that heterogeneity is also inevitable for these new instance types and remove specific information, remains to be seen.

Dividing a Region into AZs, and encouraging their use for load balancing and high availability, tends to suggest that the resources available in the various AZs are likely to be broadly similar. However, we have experimentally determined that the proportions of CPU models backing instances are quite different in different AZs. Indeed, with heterogeneous hardware in the same Zone, even Zone-specific load balancing, and related scalability matters, may necessitate further thought.

In the next section, we present some experimental results showing the distribution of CPU models found in a number of AZs.

3 CPU PERFORMANCE AND DISTRIBUTION IN EC2

In April 2013 (Author1 and Author2 2013), we benchmarked the compute capacity of close to 1300 instances on EC2, across 14 AZs in 6 (out of the 8) Regions. We included 11 (non-specialised) instance types across 4 instance classes: First Generation Standard, High CPU, High Memory and Second Generation Standard. The image id backing the instances was ami-a73264ce which is a para-virtualised image. As such the cpuid instruction (used by the Linux kernel to populate /proc/cpuinfo) does not trap to the hypervisor. Through cpuid we identify the CPU model underlying an instance. And so can determine CPU model to instance class associations, and also the proportions of CPU models found in a given AZ.

We measure an instance's compute capacity by timing the compression of an Ubuntu 10.04 desktop image ISO with bzip2, a compression tool found on Linux systems. Bzip2 is a compute bound application and forms part of the SPECint 2006 CPU benchmark suite. In Table 1 below we record the benchmark times of 540 m1.small instance, broken down by CPU model backing them.

Table 1: Bzip2 compression times on m1.small instances.

Statistic	E5430	E5-2650	E5645	E5507
Mean(s)	445	470	510	621
Sd(s)	14	13	11	28
Min(s)	425	443	488	578
Max(s)	482	519	543	716

From Table 1 we see that, for this particular task, it is clearly preferable to obtain an E5430 than a E5507. These results concur with related work (reviewed in Section 9), which shows that an instance's (compute) performance is primarily determined by the CPU model backing it, and that variation between instances backed by the same CPU is much smaller than the variation between instances backed by different CPUs. As such, determining the expected performance of a set of instances depends, in part, on being able to *predict* which CPU models will back them. Since, in this paper, we seek to elaborate a capability to undertake such predictions, we consider the following problem: *In a request for a number of instances of a given type (from a given AZ), what is the expected number of each CPU model backing the instances?*

We begin with the observation that it will depend (at a minimum) upon (1) the proportions of CPU models found in a given AZ, and (2) the scheduling algorithm being used to place instances onto hosts.

In September 2013 an updated survey was conducted focusing on the AZs us-east-1a and us-east-1c in the Region US East. We ran 150 instances of types m1.small, m1.medium and m1.large in both us-east-1a and us-east-1c, that is 900 instances in total. Table 2 shows the percentages of each model found backing our instances in the 2 AZs. We abbreviate us-east-1a and us-east-1c by 1a and 1c.

Table 2: Proportion of CPU models for FGS instances is us-east-1a and us-east-1c.

Type	AZ	E5430	E5-2650	E5645	E5507
m1.small	1a	0	0	98%	2%
m1.medium	1a	0	85%	15%	0
m1.large	1a	0	67%	33%	0
m1.small	1c	25%	0	26%	49%
m1.medium	1c	24%	1%	39%	36%
m1.large	1c	14%	0	7%	79%

We *do not* interpret the above percentages as an estimate for the actual proportions of hosts with a given CPU type in the Zone, rather as an estimate of the probability of obtaining a particular model. Clearly, one would assume the percentages found to be reflective of actual host proportions in the AZ. From Table 2 we see that in us-east-1a we have very high probability of an m1.small obtaining an E5645, whilst the m1.medium has a much smaller chance. However, we do expect probabilities to change over time, and perhaps even frequently. We discuss this further in section 7.

We also note that without much further consideration a customer (given knowledge of Table 2) may well choose us-east-1a over us-east-1c for certain instance types. However, we also note that us-east-1c may offer better price/performance compared to us-east-1a due to it having a larger proportion of the E5430 model.

4 MODELLING AN AZ

In this section we develop a simple model of an AZ that will allow us to simulate requests and responses. Public Clouds are generally opaque - few architectural details are made available. And so we are guided by knowledge derived from the OpenStack reference architectures (OpenStack, no date), experiments on EC2, and our experiences of running both Eucalyptus and OpenStack Private Clouds at our institution.

To simplify matters we model requests only for FGS instances, the m1.small, m1.medium, m1.large and m1.xlarge instances. As discussed earlier, this class is backed by Intel Xeon E5430, E5-2650, E5645 and E5507 CPUs.

By VM density we refer to the ratio of vCPUs to physical cores that a host presents. For example, a host with 8 cores and VM density = 1 presents 8 vCPU. In this case the host could run, for example, 1 m1.xlarge (requiring 4 vCPU) and 4 m1.small instances (1 vCPU each). However, if the VM density is increased to 1.5 we now have 12 vCPUs presented, and so the host may now run (for example) an additional 2 m1.large instances. Of course, running a Cloud at a higher VM Density will lead to greater resource contention amongst the VMs, resulting in larger performance variation. We set the VM Density = 1.

The scheduler in our model (in common with the OpenStack scheduler) is capable of filtering instances types onto associated hardware. We therefore avoid considering hardware not associated with FGS instance class.

In our model, a host for a given class can run a mix of its instance types. For example, a FGS host with sufficient vCPUs could have an m1.small, m1.medium, m1.large and an m1.xlarge running on it at the same time. There is no obvious reason why this should not also be true on EC2.

We have chosen to use the chance scheduling algorithm. This chooses hosts at random and, if there are available resources, will allocate the VM to the host.

Hence our AZ consists of:

- A number of host machines running a hypervisor in order to host VM instances.
- Heterogeneous hosts (hosts may have different hardware).
- Each host presents a number of vCPUs which are allocated to instances running on it.
- One scheduler, responsible for placement of VM instances onto hosts.
  - The scheduler can be configured to implement a number of scheduling algorithms. We use chance scheduling.
  - The scheduler also filters instance types onto hosts of a given type.
- VM Density = 1.

5 SIMULATING USER REQUESTS

We restate the problem of interest: *In a request for a number of instances of a given type, what is the expected number of each CPU model backing the instances?*

By state of the Zone, we mean the total number of available vCPUs and their arrangement over the host machines. Clearly, the state of the Zone will change over time, potentially for a number of different reasons including demand fluctuation, capacity management, short-term unplanned outages, planned maintenance and hardware replacement and refreshes. Some of these are merely transient, whilst other will change the composition of the Zone.

To generate a state, we need to make an assumption about data centre loading, which we consider to be the number of *in use* vCPUs. Whilst we have no reliable data for actual VM loadings in data centres, and providers do not release such information, it is generally claimed to be high. Indeed, one of the positive arguments for Cloud adoption is high utilisation. Further, providers claim they can maintain high rates and that whilst an individual customer's demand fluctuates; aggregate demand (over all customers) remains fairly constant. This has been referred to as one of the 'Laws' of Cloudonomics (Weinman, 2008). From the discussion above, we believe that scenario 2 would be a reasonable assumption on data centre loading.

Our algorithm for generating a Zone state depends on size of the Zone, the load and the instances we wish to run in it. Assuming 1000 hosts each with 8 cores (without load), so 8000 vCPUs in total, able to run m1.medium, m1.large and m1.xlarge instance, we do as follows:

```
LOADING = randomint(0.7*8000,0.9*8000)
while ( LOADING > 0 ) do
    k = random-select(1, 2, 4)
    schedule(20 instances of k vCPUs)
    LOADING = LOADING - 20*k
```

With all hosts, and so all vCPUs, free, we generate a random loading of 70% to 90% of the vCPU count. Next, we randomly select (random numbers drawn from the uniform distribution) a vCPU size (1, 2, 4 to reflect the vCPU requirement of m1.medium, m1.large and m1.large, respectively) and schedule 20 instances of this type into the Zone. This is repeated until we have the required load. This gives a mix of types running on hosts. In our code the schedule function ensures that a VM is only placed on a host which has a sufficient number of free vCPUs. The schedule function updates the host and VM state after allocation.

We choose a static (Monte Carlo) simulation of user requests, proceeding as follows: We first generate a Zone State as described above, we then simulate a request (over this state) to the scheduler, which allocates the required number of instances onto hosts. We then count the number of CPU models in the hosts backing our instances. An example output of a request for 20 m1.small instances, in a Zone containing CPU models E5430, E5645 and E5507 would be: (5 E5430, 4 E5645, 11 E5507).

For each iteration in the simulation, the requested instance type is constant, the request size is 20 (the maximum on EC2), and Zone state is generated anew so if we are interested in m1.small types then the simulation asks for 20 m1.small instances 50,000 times (each time over a new Zone state). By repeatedly sampling over Zone states a sufficiently large number of times (50,000 in this case) we generate a distribution for the CPU models from which we calculate estimates of the expected number of CPU models per request. In practice there will be some request failures, however for simplicity we assume that all instances are successfully placed in each request.

We investigate two scenarios:

Scenario 1: Zero loading in the data centre.

Scenario 2: 70% to 90% of the cores in the data centre already in use.

To run a simulation, we need to specify the proportions of the CPUs in the Zone. Whilst we have data from our survey of us-east-1c, we are faced with 2 immediate problems. Firstly, as already discussed, the percentages are estimates of the probability of obtaining a model and not estimates of the actual host proportions. Clearly though they are likely to be indicative of them and as there is no obvious way to more accurately determine host proportions we take the percentages as estimates for our model. Next, instance types share common platforms, and yet from Table 2 are not being allocated to them in the same fashion. And so, if we wish to model m1.small requests should we use 49% as the percentage? Or should we average over 49%, 36% and 79%, which would give 55%? When modelling a given type we take the percentage found for the type and not the average.

6 SCENARIO 1

We are requesting 20 FGS instances from the pool of all available hosts. We start by considering the m1.medium instance type which has 1 vCPUs, and so requires 1 physical core by our assumptions. We are using a chance scheduler, which chooses hosts randomly and allocates a VM to it, if it has sufficient resources (if not choose a new host). Thus we are sampling over the set of cores without replacement. In this case the joint probability density function is described by the multivariate hyper-geometric distribution. The marginal distributions are also hyper-geometric, however when the sample size is small compared to the population, as in this case, the hyper-geometric tends to binomial distribution. And, so we have, for example E5430 ~ B(n,p) where n = 20 and p is proportion of E5430 cores expressed as a decimal for m1.medium instance. We also note that, whilst this request is for m1.medium instances, the

argument above, with slight modifications, applies to all FGS instance sizes.

7 SCENARIO 2

The AZ is assumed to have a loading such that 70% to 90% of the cores are already in use. That means the total available core count is between 800 and 2400. The first simulation is with m1.small requests in us-east-1c with CPUs in the following proportions: 25% E5430, 26% E5645 and 49% E5507. We then simulate m1.medium and m1.large using data from Table 2.

We calculate the marginal distributions for our random variables by simply counting the number of times a particular output occurs. To calculate the marginal probability distributions we simply divide the number of occurrences by the 50,000, from this we calculate the expected number of each CPU. In Table 3 below we present the calculated expectations for all CPU models in each simulation, and in Figure 1 we present the frequency distribution for the discrete random variable defined by the number of E5430 models returned in response to a request for 20 m1.large instances.

Table 3: Expected Value of CPU Returns.

	E5430	E5645	E5507
m1.small	5.00	5.197	9.795
m1.medium	5.0114	5.195	9.793
m1.large	5.008	5.192	9.799

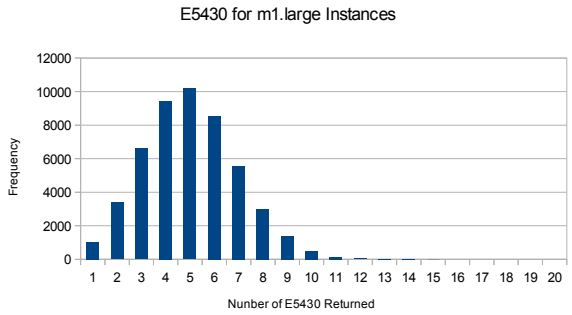

Figure 1: E5430 for m1.large Instances.

We suspect that the distributions are binomial. For the distribution depicted in Figure 1 we perform a Chi-Squared Goodness of Fit Test, where the null hypothesis is E5430 ~ B(20,0.25), and E5430 denotes the discrete random variable as defined in the paragraph above. For the test we generate expected output from B(20,0.25) and then calculate the chi squared statistic as 17.99. With 20 degrees of freedom, at a 5% significance level, we fail to reject the null hypothesis. Similarly, our simulations for m1.small and m1.medium instance types also produce marginal distributions which are binomial.

8 COMPARISONS WITH POST SIMULATION DATA FROM US-EAST-1C

From the above simulation E5430 ~ B(20,0.25). By a direct calculation P(E5430 <=7) – P(E5430 <=2) = 0.673. And so in 2 out of 3 samples of 20 instances we would expect the number of E5430 models obtained to be between 3 and 7.

In a follow on experiment to the one described in section 3, paragraph 6, and with results in Table 2; we requested 3 samples of 20 m1.large instances, with a 10 minute interval between requests and no resources were released until we had concluded the experiment. In Table 4 below we record the results.

Table 4: CPU models returned in request in us-east-1c for 20 m1.large.

Sample	E5430	E5645	E5507
1	3	0	17
2	5	1	14
3	2	0	18

In a final experiment, we launched 100 m1.small instances. Assuming a binomial distribution, and using our data in Table 2 as an estimate for p, the expected numbers (100*p) of each CPU model would be (25,24,49). We obtained (21,29,50).

It is not yet clear to us why instance types within the same class obtained the same CPUs in different proportions. It may well be the case that instances types within the same class do not run on the same host at the same time. That is, a host may run 2 m1.xlarge machines, and if one is terminated will only host another m1.xlarge. However, if both m1.xlarge instances are terminated then the host is available again to all instance types in the class. Over time such a scheme may lead to the distributions we have seen. We intend to pursue this idea further.

9 SCHEDULERS AND AZ STATE

We chose the chance scheduler as our staring point as it is was previously the default scheduling algorithm in OpenStack. There are a number of algorithms that schedule more intelligently than

chance, and reflect the priorities of a site. As an example, minimise the number of hosts in use; whilst keeping the hosts not in use in a low energy state. Others algorithms attempt to make efficient use of resources; and these are generally variations on the bin packing problem, as described in Wilcox, et al. (2010), with the scheduler employing a heuristic to solve the problem. The AZ states and the CPU distributions generated by this scheduler are potentially quite different from those that a chance scheduler would generate. Similarly, schedulers that seek to take advantage of co-locating VMs, for either deduplication of base images or page sharing amongst VMs may also generate AZ states different from both the chance scheduler and bin packing.

10 RELATED WORK

In Armbrust, et al. (2009), the authors describe EC2 Cloud performance as unpredictable. Similarly, Osterman, et al. (2010), describe performance in the Cloud as unreliable. The Magellan Report on Cloud Computing for Science (Yelick, et al., 2011) found that 'applications in the Cloud…experience significant performance variations' and noted that this is in part caused by the heterogeneous nature of EC2. However, EC2 was the only Public Cloud they considered. Similarly, in Phillips, Engen, and Papay, (2011), also discovered differential performance in instances of the same type on EC2 when attempting to predict application performance.

Iosup, et al. (2011), demonstrate that performance variation exists in a range of AWS services, including S3, SDB and SQS, and so performance variation of applications using these services is not solely dependent upon instance compute capability.

Schad, et al. (2010), show that the compute performance of instances sampled from US East N. California and the EU West Dublin Region falls into two distinct performance bands. Upon further investigation they detect two different CPU models backing their instances and speculate that previous results are explained by differences in CPU model. They also found a difference in the amount performance variation in two out of the four AZs used (us-east-1c and us-east-1d).

Ou, et al. (2012), suggest that the heterogeneous nature of Clouds can be exploited by estimating the probability of obtaining a particular CPU model backing an instance. Their method assumes that instances will be randomly sampled from all AZ available in US East N. California, which, as noted by the authors in Schad, et al. (2010) is not necessarily the case. To fully exploit heterogeneity, requests must be made per AZ, and this requires an analysis of the CPU distributions per AZ in the Region of interest. Additionally, they assume that the CPU model obtained in a particular request is *independent* of the one previously obtained. There is no guarantee of this; and VM placement algorithms need to be considered.

There is a large literature base for VM scheduling; for example Bazarbayev, et al. (no date), consider scheduling of VMs with identical or similar base images onto the same hosts. Reig, Alonso and Guitart, (2010), add machine learning capabilities to the scheduler in order for it to be able to predict resources required for a given execution time. However, as far as we aware, there are no papers looking specifically at how scheduling may affect the probability of obtaining particular resources.

11 CONCLUSIONS AND FUTURE WORK

The compute performance of an instance is primarily determined by the CPU model backing it. Currently customers cannot request either a desired level of performance, or, failing that, a particular CPU model. Being able to predict the performance of a set of instances before provisioning them, is therefore an important issue for Cloud users. Being able predict the range of CPU models that may be obtained in response to a request will go someway towards answering this question.

In this paper we therefore address the following problem: *In a request for a number of instances of a given type, what is the expected number of each CPU model backing the instances?*

By create a simple AZ model we showed that in the case of an empty (but sufficiently large) AZ then the number of models of a given type in a request is binomially distributed B(20,p), where p is the proportion of CPU models in the zone.

We then modelled an AZ and under a load of 70% to 90% capacity used, this describes the number of vCPUs in use. We used the uniform distribution to generate the AZ state, and then used a chance scheduling algorithm to allocate FGS instances into our AZ. The simulation samples over the set of possible states, so we estimate a joint probability distribution for our random variables E5430, E5645 and E5507 (denoting the CPU models

found in AZ respectively). From this joint distribution we calculated the marginal distributions and using a chi-squared goodness of fit test we showed that these are drawn from a Binomial distribution.

Therefore, in both scenarios the answer to our original question is this: the expected number of CPUs of a given type is the request size multiplied by the probability of obtaining a CPU (in a request for one instance). Comparing with a (small) set of samples drawn from us-east-1c we find that n*p appears to be reasonably good estimate of the number of models we may obtain.

We intend to investigate commonly found scheduling algorithms and incorporate them into future models, and compare the distributions our models generate with instance sampled from providers.

We recognise that the work involved in implementing such a model, and estimating possibly frequently changing probabilities, may make it impractical for certain users. However, we believe that this is feasible for Infrastructure Cloud brokers; and indeed they are ideally placed to do so.

More generally, we believe this will enable brokers to make performance related pricing offers to customers of the following form: Offer 1: We are 95% sure that at least 15 of the 20 VMs will run the workload in less than or equal to T seconds. Our future work is aimed at advancing this.

REFERENCES

Phillips, S., Engen, V., & Papay, J., 2011. Snow white clouds and the seven dwarfs, in *Proc. of the IEEE International Conference and Workshops on Cloud Computing Technology and Science*, pp738-745, Nov. 2011.

Ou, Z., Zhuang, H., Nurminem, J.K., Yla-Jaaski, A., & Hui, P., 2012. Exploiting Hardware Heterogeneity within the same instance type of Amazon EC2, presented at *4th USENIX Workshop on Hot Topics in Cloud Computing*, Boston, MA. Jun. 2012.

Reig, G., Alonso, J., & Guitart, J., 2010. Prediction of Job Resource Requirement for Deadline Schedulers to Manage High-Level SLAs on the Cloud, in *2012 Ninth IEEE International Symposium on Networking Computing and Applications,* pp 162-167, July 2010.

Amazon EC2 FAQs, no date. Aws.amazon.com. [Online]. Available at: < http://aws.amazon.com/ec2/faqs> [Accessed: 30 September 2013].

Google Cloud Platform, no date. cloud.google.com. [Online]. Available at:https://cloud.google.com [Accessed: 30th September 2013].

HP Public Cloud, no date. www.hpcloud.com. [Online]. Available at: <https://www.hpcloud.com/> [Accessed: 2nd July 2013].

Rackspace Global Infrastructure, no date. www.rackspace.com [Online]. Available at: http:// www.rackspace.com/information/aboutus/datacenters/ [Accessed: 30th September 2013].

AWS Global Infrastructure, no date. aws.amazon.com. [Online]. Available:<http://aws.amazon.com/about-aws/globalinfrastructure/> [Accessed: 30th September 2013].

Armbrust, M., et al, 2009. Above the clouds: a Berkely view of cloud computing, Technical Report EECS-2008-28, EECS Department, University of California, Berkeley.

OpenStack Scheduling, no date. Docs.openstack.org [Online]. Available at: < http://docs.openstack.org/ grizzly/openstack-compute/admin/content/ch_schedu ling.html> [Accessed: 30th September]

Weinman, J., 07/09/2008, 10 laws of Cloudonmics. Giga.com. [Online]. Available at http://gigaom.com /2008/09/07/the-10-laws-of-cloudonomics/ [Accessed: 30th September

Bazarbayev et al., no date. Content based scheduling of Virtual Machines in the Cloud. [Online]. Available at https://www.perform.csl.illinois.edu/Papers/USAN_pa pers/12BAZ01.pdf > [Accessed: 30th September]

Wilcox, D., McNabb, A., Seppi, K., & Flanagan, K., 2010. Probabilistic Virtual Machine Assignment, Cloud Computing 2010: First International Conference on Cloud Computing, Grids and Virtualisation, Lisbon, November 21-16, 2010.

S. Osterman et al, A performance analysis of EC2 cloud computing services for scientific computing, Cloud Computing, Lecture Notes of the Institute for Computer Sciences, *Social-Informatics and Telecommunications Engineering*, vol 34, 2010, pp 115-131.

Iosup, A., Nezih, Y., and Dick, E., 2011. On the performance variability of production cloud services. In Cluster, Cloud and Grid Computing (CCGrid), 2011

Yelick, K., et al, 2011. The Magellan Report on Cloud Computing for Science. [Online]. Available at: http://www/alcf.anl.gov/magellan [Accessed at: 2nd January 2014].

OpenStack, no date. www.openstack.org [Online]. Available at: http://www.openstack.org [Accessed: 1st January 2014].