Journal of Cloud Computing
a SpringerOpen Journal

RESEARCH **Open Access**

Options, forwards and provision-point contracts in improving cloud infrastructure utilisation

Owen Rogers* and Dave Cliff

* Correspondence:
csorr@bristol.ac.uk
Department of Computer Science, University of Bristol, Merchant Venturers Building, Bristol, UK

Abstract

Cloud computing takes away much of the need to plan future IT demands from the consumer, and puts it in the hands of the provider. Consumers don't need to give advance notice to start or terminate virtual machines, and can do so in real time to reflect changing business objectives. It is the task of the cloud IaaS provider to optimise the use of her infrastructure, and ensure there are enough resources available. Achieving optimum server utilisation in the data centre is particularly challenging – advance notification can help the provider to schedule workloads more efficiently, but this is at odds with one of the key benefits of cloud computing. In this paper, we propose a pricing method that combines options contracts with on-demand purchasing. We show that the method can provide consumers with the flexibility and cost-benefits afforded by cloud computing, and can benefit the provider by improving server utilisation and therefore reducing energy costs. Furthermore, we show how provision-point contracts, often used by deal-of-the-day websites such as Groupon, can further improve the method, making it even more attractive to the provider.

Keywords: Utility computing, Assurance contracts, Energy-efficient computing, Scheduling

Introduction

Cloud computing provides consumers with unprecedented levels of consumption flexibility. In principle they can provision new resources as and when required without providing advance notice, nor specifying for how long those resources will be required. As users do not have to engage in the capital expenditure of building their own infrastructure, hiring IT systems support staff, or investing in maintenance of physical machinery, it is generally accepted that on-demand pricing for cloud computing resources offers financial benefits to consumers [1,2].

But does this flexibility benefit the provider of cloud infrastructure? The provider must manage her hardware (amongst other elements in her infrastructure), ensuring there is enough capability to meet demand while at the same time controlling her costs. The on-demand nature of enterprise cloud computing means that consumers are not required to supply information to the provider in advance. Planning without this information is a challenge for the provider, and poor forecasting can be costly.

In the short term, it can be difficult to schedule customer instances to servers efficiently without advance notification of usage. Most cloud computing providers require no duration of execution to be stipulated when the instance is started, and hence efficient scheduling of virtual machine instances that reduces or minimises the number of powered servers is a tough challenge [3,4]. With electricity costs being a source of major expenditure, keeping the number of active servers as low as possible is a priority for both cost-saving and carbon-footprint reduction.

In the longer term, the provider must ensure capability is available to meet demand, but when do they invest in new infrastructure? As manufacturing processes improve and economies of scale increase, the real cost of infrastructure decreases while its technological capability increases. For that reason, it is better for the provider to wait for as long as possible before investing in additional capability, so that they get the best value for money [5]. But how can the provider know when is the best time?

In this paper, we first describe how the use of a type of financial instrument called an *options contract*, used in combination with on-demand purchasing, can increase server utilisation and therefore reduce energy costs. Furthermore, we show that the method can provide cost-savings to the consumer while retaining the flexibility afforded by cloud computing.

We extend our method to act as *provision-point contracts*, often used by deal-of-the-day websites such as Groupon. We show that our combination of provision-point contracts and options contracts can protect the provider's revenue in situations where no advantage is gained as a result of offering such advance reservations. Crucially, our methods do not increase the provider's downside risk. That is, in the worst case the final novel method described here is cost-neutral, and when it is not cost-neutral then the provider makes a saving.

This paper contributes to the literature on cloud and grid economics, by detailing a number of pricing schemes that offer commercial benefits to both provider and consumer, while retaining the flexibility of on-demand pricing. Currently, there is limited research that considers cloud pricing models that can be used in conjunction with on-demand pricing schemes to provide additional benefits. We believe our schemes can ease capacity planning issues for the provider, and allow her to make best use of her physical infrastructure.

Enterprise pay-as-you-go pricing schemes

Currently, most infrastructure-as-a-service (IaaS) providers allow users to start virtual machines as soon as required, and to be subsequently billed for the period the machine was running. Furthermore, most providers allow their customers to choose from a number of different sized instances of virtual machines on which applications can be built – these are referred to as *instance types*.

Different applications have different requirements for memory, central processing units (CPUs), and storage. For example, a computationally intensive application (such as image processing) may require more CPU capability than a web application. For the provider, this simply means partitioning up the infrastructure on individual servers into *computational units* that can then be combined to form different instance types. The number of computational units determines the *size* of the virtual CPU (*'vCPU'*) available to the virtual machine. This is relatively simple for memory and storage which can be partitioned by dividing the total available in the server by the number of units required.

However, splitting up a CPU chip between multiple virtual machines isn't quite so straightforward due to the complex scheduling and architecture of multi-core CPUs.

The definition of a computational unit varies between providers of public cloud services. Some providers choose to divide a server into a large number of computational units so that consumers have a finer level of granularity when specifying their CPU requirements. In this case, the computational unit is often defined as a fraction of a core or as the number of actual clock cycles available.

Other providers may choose to offer a limited number of computational units for easier management. One option is to assign one CPU core to one computational unit, with each unit getting an equal share of memory and storage.

This variation between providers means that there is no standard measure of a computational unit. As such, a fair comparison of cloud performance cannot be made on published metrics alone, and many providers suggest benchmarking applications on many different sized instances before choosing a final size [6].

Providers are reluctant to publish details on their datacentre infrastructure and internal measure of a computational unit. If consumers compare providers using a published standard, they will naturally choose the most cost-effective. Providers would rather engage with consumers in the first instance, and then attempt to persuade them to choose their service, than not be approached at all.

The largest provider of cloud IaaS is Amazon Web Services (AWS), who offer a maximum instance size of 8 computational units. In a typical blade-server motherboard housing two quad-core CPUs, eight CPU cores are available. If we assume that AWS would wish to offer a range of instance sizes up to the maximum available, then one computational unit is equivalent to one physical core. In this paper, we assume the same relationship, but simulations exploring alternative allocations, such as more than one unit per physical core, is a topic we plan to explore in future work.

The total capacity of units on a server can be split amongst several virtual machines of varying sizes. In many commercial deployments, a virtual machine must be fully contained on a server for it to be able to access the resources assigned to it.

When users purchase virtual machines on-demand, these are started immediately. The provider might choose to start an instance on the first server where free space is available, with the objective of keeping the numbers of servers in operation to a minimum – a *first-fit* algorithm [7]. This strategy might be adopted to reduce power costs that vary with the number of powered servers, or to reduce the management overhead of monitoring and managing unused servers. Packing these differently sized virtual machines into the smallest number of servers is an instance of the combinatorial NP-hard *bin-packing problem*.

The provider wants to obtain the maximum utilisation possible across the cloud infrastructure so that all servers are all being used to their full capacity. As a result, this will keep the number of servers that are powered at a minimum, thereby reducing electricity costs. However, the order of arrival and distribution of virtual machine sizes demanded affects the utilisation of the cloud infrastructure.

The first-fit algorithm has been shown to use no more than $2 + 1.7b$ bins, where b is the minimum number of bins used in the optimum solution [7]. An *online* bin-packing algorithm places an item before subsequent items are placed – the first-fit algorithm has been shown to be the optimum online bin-packing algorithm [8]. This is essentially the model used by on-demand provisioning: in the absence of any fore-knowledge of future demand for resources (nor for how long resources will be required), virtual machines are placed as soon as each request is received.

It is possible to migrate running virtual machines (VMs) from server to server without affecting the machine's performance. In principle, this can facilitate more efficient use of servers by shifting VMs from lightly loaded servers onto higher-laden servers until the lightly-loaded server is unutilised and can be switched into a sleep or off state. However, performing this on large-scale datacentres is not an option due to the vast quantities of virtual machines and servers in use, and the potential for network bottlenecks as a result of the transfer of huge amounts of data [9].

However, if the IaaS provider has a forecast of future usage, she can schedule more efficiently by sorting customer requirements in descending order of size, and then assigning these to the first server with available space (*offline* bin-packing). This is often referred to as the *first fit decreasing algorithm*, and has been shown to allocate items using no more than $1 + (11/9)b$ bins [10]. An alternative method, known as the lower bound and reduction (LBR) procedure was proposed by Martello and Toth, which they claimed aids in reaching an optimum solution [11].

However, obtaining such a forecast has a number of issues. Firstly, how can we incentivise users to provide a forecast instead of just purchasing the resource on-demand? The obvious solution is to provide them with a benefit, such as cost reduction, offered in return for their forecast.

Secondly, how do users know what they are likely to use? One of the main benefits of cloud computing is for users to purchase resources on-demand for immediate execution without such a forecast being required. Does forecasting negate one of the major benefits of the cloud?

One solution could be to combine forecasting with on-demand computing. Consumers who can provide some commitment to future resources are rewarded with a cost benefit. Should they need more resources at a later date, they can simply buy more resources on-demand and utilise or integrate these with their reserved resources using the rapid-scalability and integration capability cloud computing provides.

In the finance literature, the advance-reservation purchase of resources for future use is known as a *forward contract,* which gives buyers guaranteed access to the resource in advance of when it is delivered. The buyer is *obliged* to take ownership of the resource, so she must be confident she will require the resource at the date of delivery. This level of commitment may discourage users to forecast.

An alternative is to allow users to purchase *options contracts*. These give buyers the legal right (but *not* an obligation) to purchase a resource (or *underlying asset*) for an agreed price on some later delivery date. The benefit of options is that the users can pay a small "deposit" to purchase guaranteed future access to a resource without committing to pay any more should they subsequently find that they do not require the resources [12].

Wu, Zhang, and Huberman [13] explored the use of options contracts as a pricing scheme for cloud services, and suggested a two-period model (which we refer to hereafter as the WZH model) that uses an independent third party called the Coordinator. The Coordinator solicits from each user i a probability estimate p_i that is the user's estimate of the probability that they will want to use a resource in the next period. The

Coordinator uses this information to purchase resources from the provider. In Wu *et al.*'s original work, they showed that their model was profitable for the resource-consumers and the Coordinator, but did not discuss if the resource-provider could benefit from such a scheme.

In this paper, we show how a provider can utilise the model directly, offering options contracts to its users without an intermediary party, and subsequently use this information to improve the utilisation of virtual machines running on a scalable cloud infrastructure.

Furthermore, through the use of provision-point contracts commonly used for deal-of-the-day websites such as Groupon, we improve the model so that customers only receive a discount if the anticipated demands of the user-population are likely to provide the provider with a cost-saving. This protects the provider's revenue, while offering the prospect of reducing power costs through the reduction of active servers.

Section 3 of this paper presents results from simulation studies where allocating VMs to servers is conducted via options contracts; the primary finding of section 3 is that options can offer significant improvements in efficiency, but only for usage patterns than involve larger instance sizes; for smaller sizes, we find that the method can generate reductions in efficiency, which is obviously unattractive. Furthermore, customers receive a discount regardless of whether a reduction in utilisation is achieved. As a result, the provider loses revenue without any benefit being gained.

In Section 4 we demonstrate that this problem can be reduced or eliminated by combining option and forward contracts with provision-point contracts, and that the combined method using forward contracts does not lead to any reductions in efficiency: the efficiency of allocations is either unchanged or improved, but never worsened. Furthermore, the method protects the provider's revenue by only awarding a discount where a utilisation improvement is realised.

One important factor to bear in mind when reviewing our results is that the very large numbers of servers in a typical cloud-provider's datacentre means that relative improvements expressed as single-digit positive percentages, which might on face value appear to be trivial, can result in very significant savings when measured in dollar terms. For example, a reduction of 1% in the number of servers required used may sound tiny, but in a major cloud data-centre this could translate into reducing infrastructure needs by thousands of physical server machines. We revisit this point in Section 5 where, purely for illustrative purposes, we translate the percentage improvements generated by our methods into plausible dollar-valued savings for a major cloud infrastructure-as-a-service (IaaS) provider.

Reservations and probability-based options

In the first period of the WZH model, users purchase an options contract that gives them the right to subsequently buy a resource for a pre-agreed strike price, with the "delivery date" being the start of the second period. In the second period, users may then choose to exercise their option by paying the pre-agreed strike price. If they do not wish to execute their right, they pay nothing (apart from the price of the contract already paid in the first period, which is not refunded).

For each user, the price of the contract and the pre-agreed strike price are calculated using a probability in the range [0,1]. The probability q_i is submitted by each user i to the Coordinator in the first period. User *i*'s submitted value q_i is user *i*'s statement of the probability that she will want to use a resource in the second period. Their submitted probability, q_i, does not have to been their actual probability, p_i. The Coordinator aggregates these probabilities and purchases resources on the users' behalf.

Wu *et al.* showed via theoretical analysis that, when the WZH model's parameters were set with appropriate values:

- The Coordinator could make a profit by providing the service;
- Users are encouraged to submit their probabilities honestly; and
- Users who submit honest probabilities will reduce their costs over time, despite sometimes purchasing an options contract which is subsequently not executed.

In this paper, we use the truth-telling mechanism proposed by Wu *et al.* to derive forecasts of future demand for computing resources from users, which is subsequently used to schedule virtual machines. The mechanism works as follows:

The price paid by user i for a resource is:

$$\begin{cases} f(q_i) \text{ if she needs one unit of resource} \\ g(q_i) \text{ if she does not need the resource} \end{cases}$$

Each user submits her true probability so that she expects to pay the least amount later. User i would expect to pay:

$$w(q_i) = p_i f(q_i) + (1 - p_i) g(q_i)$$

Her optimal submission $q_i{}^*$ is determined by the following first order condition:

$$w'(q_i^*) = p_i f'(q_i^*) + (1 - p_i) g'(q_i^*) = 0$$

For an honest forecast, $p_i = q_i$

$$w'(p_i) = p_i f'(p_i) + (1 - p_i) g'(p_i) = 0$$

It is clear that $f'(p) = -k(1 - p)$ and $g'(p) = kp$ satisfies this condition, where $k > 0$.

Therefore, we obtain the following pricing equations that meet the truth telling condition:

$$f(p) = 1 + \frac{k}{2} - kp + \frac{kp^2}{2}$$
$$g(p) = \frac{kp^2}{2}$$

This can be considered an option if *g(p)* is paid in the first period to reserve the resource, and *f(p)* – *g(p)* is paid in the second period should the user wish to use the resource.

In previous work [14], we explored and validated Wu *et al.'s* claims though simulation experiments and showed that a simple evolutionary optimization process operating on an initially maximally dishonest pool of users results in the pool of users becoming more honest over time when interacting with the WZH system.

In [15], we demonstrated how a broker (a third party who buys and sells resources on her clients behalf) can use the WZH model as a basis for determining when to invest in longer-term access to cloud computing resources. We showed that the broker can profit in a number of situations, using a publicly available cloud computing service as the provider.

In [15], we focussed on the benefits of the model to the brokers and consumers. We assumed the provider would benefit as a result of increased uptake of their services. In this paper, we use the same pricing scheme as used in our previous work. However, we now investigate if the WZH model can directly benefit the provider through better virtual machine scheduling.

We propose that the provider can benefit by offering both probability-based options directly and on-demand pricing, without the need for a third party intermediary. The mechanism used by the provider is discussed in the next section.

Mechanism

In our *'combined pricing'* method described below, the provider allows users to purchase options 24 hours in advance (the *reservation phase*) of when they will be executed (the *execution phase*). Options provide one hour of access to the virtual machine, and as *European Options*, they can only be executed at the expiry date.

Users can also purchase on-demand virtual machines which provides one hour of access to the resource, starting the moment the order is made. We use the pricing equations originally proposed by Wu *et al.* but employ them to define the price of a computational unit. Thus, the cost of a virtual machine of size *n* is the cost of *n* computational units.

Reservation phase

1. In the reservation phase, users may choose to purchase an options contract for a virtual machine by paying a premium prior to execution phase. They pay:

$$P_{res} = ng(p) = n\frac{kp^2}{2}$$

   where *n* is the number of computational units the virtual machine will require (the *'instance size'*), *p* is the probability it will be required, and *k* is a constant introduced by Wu *et al.* to tune pricing. We use the same value of $k = 1.5$ used by Wu *et al.* for consistency. We also assume that due to the truth-telling condition being satisfied earlier in this section, users submit honest probabilities.
2. For each instance size, the provider aggregates the probabilities from the entire user population that a virtual machine of this size will be required at the time of expiry. This gives the provider the basis of a forecast of how many of each size of virtual machine will be required.
3. The provider trials a number of bin-packing algorithms with the objective of finding the method that uses the lowest number of servers that contains all requests from all users as submitted in the first period. In our simulations, we use the following algorithms:

   a. First fit decreasing algorithm (FFD)
   b. Martello and Toth's lower bound and reduction procedure (LBR) [11], which was chosen due to its significantly verified performance and easily replicated algorithm

4. The best-performing algorithm is used to map the forecasted demand for instances to servers ready for use in the Execution Phase.

5. This forecast could potentially be used to reduce variable costs, for example, by purchasing electricity usage in advance for a discount [16].

Execution phase

6. At the point of execution, users can choose to execute their option, i.e. claim their right to use a resource. If they want to do this then they pay:

$$P_{exec} = n[f(p) - g(p)] = n\left[1 + \frac{k}{2} - kp\right]$$

7. The provider supplies the user with access to the size of virtual machine requested previously, using the map created in the Reservation Phase. If the provider has not mapped enough virtual machines to meet demand, it will start virtual machines where they will fit using the first-fit algorithm.
8. If a user needs further resources, they will purchase on-demand virtual machines. The on-demand price is the price of executing an options contract with probability 0, as this was not forecasted by the user thus they pay the highest price in our model:

$$P_{od} = nf(0) = n\left[1 + \frac{k}{2}\right]$$

9. A rational user will choose to exercise their option on a resource over purchasing a unit of the same resource on-demand, because the on-demand instance will cost more than the combined prices of the option's purchase price and its strike price (Figure 1).

For the method to be of benefit, it must:

- reduce costs for the users, thus encouraging them to forecast; and
- reduce the number of active servers in use, thus reducing costs for the provider and therefore encouraging her to offer discounts for options.

To test this method, a simulation of this market for cloud-computing resources via options and on-demand purchases was written in Python and deployed on a commercial infrastructure-as-a-service environment.

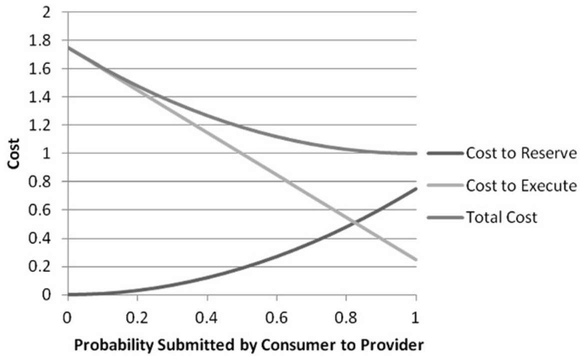

Figure 1 Cost per computational unit of reserving and executing probability-based options contracts.

Simulation study of allocation via of forward and option contracts

In this section, we present results from simulation studies that compare the efficiency of allocations via our combined pricing method to the efficiency of allocations using conventional on-demand scheduling. We find that using options contracts can offer significant efficiency improvements, but only when usage is above a particular threshold: when it is below that threshold, using options can actually make things worse. Nevertheless in Section 4 we go on to show how options can be combined with provision-point contracts to create an improved method that reduces utilisation losses. Furthermore, we show how provision-point forward contracts can deliver significant gains in efficiency, while preventing discounts being awarded where no gain in efficiency is realised.

Scenario

Users purchase options up to 24 hours in advance of when they will be exercised. Potentially, other regimes could be used but we have chosen this so that users have a fair change of predicting future resource requirements, and the provider has enough notice to benefit from this prediction (e.g. by scheduling workloads, purchasing electricity in advance, etc.). Options specify one hour of resource-usage, and can be used in combination with on-demand resources.

For comparison purposes only, we also run a simulation where users may purchase forward contracts instead of options. This is essentially an advance reservation where users are obliged to pay for the resource.

In the simulations in this paper, the provider operates a datacentre of homogeneous servers, due to the cost benefits of bulk purchasing hardware and the convenience of having a single stock of servers from which replacements can be obtained. The provider partitions these servers up into computational units which can then be aggregated together to offer a range of different instance types.

Users can purchase multiple virtual machines instances of any integer size from 1 to 8 computational units; one computational unit provides access to one CPU core for one hour. We typically simulate a run of 28 consecutive 24-hour days, with no difference in demand between week-days and weekends: 28 days therefore could be interpreted as one calendar month. A month duration of 28 days was chosen so that it would be easier to conduct future work on reservation periods of 7 days.

It is possible that the provider may have different physical server types to meet specific needs, such as having servers containing Graphical Processing Units (GPUs) for image processing applications. In this case, the provider could partition these servers into computational units too, and therefore offer the consumer a choice of technology, with a choice of instance types to run upon it. In our experiments, we assume that the provider uses a single technology and has access to a homogenous datacentre of servers with 2 × Quad-Core CPUs.

User behaviour

A user's behaviour is determined by two factors: her *market demand profile*, and her *product demand split*.

A user's *market demand profile* determines the resource requirements she will experience at each hour throughout the simulation. Demand varies smoothly from 0 to 1, where a demand of 1 means that a user's maximum demand requirement (40 VMs in the simulations shown here) will be needed, 0.5 represents that the user's demand is 50% of its maximum, and zero equates to no demand. The atomic unit of consumption is one computational unit for one hour. To model the dynamic variation in demand across each working day, and also across longer-term time-scales, five different *demand profiles* are used in the simulations. Each demand profile defines the pattern of demand variation over one 28-day month (672 hourly time-periods):

- *Flat* profile represents where demand is constant, and hence trivially easy to predict;
- *Random* profile represents stochastically unpredictable demand;
- *Sine* profiles (with period of 24 hours) are an approximation to daily rhythms, where demand varies sinusoidally, peaking in the middle of the day and at a minimum in the middle of the night. More precisely, in our simulations this sinusoidal demand pattern peaks around mid-day, and demand can never be negative, so a function of the form $1 + \cos(2\pi h/24)$ is used, where h is the hour-number in the day. We have explored three variations of these sinusoid patterns:

  - *Flat Sine* represents constant a constant baseline of demand with periodic variations across each day;
  - *Growing Sine* represents daily periodic demand, with the baseline increasing steadily across the month;
  - *Shrinking Sine* represents daily periodic demand, shrinking through the month.

The demand profiles are illustrated in Figure 2

A user's *product demand split* (PDS) defines the probability, for each instance type, that that instance type will be required in a time period by the user. A user may require

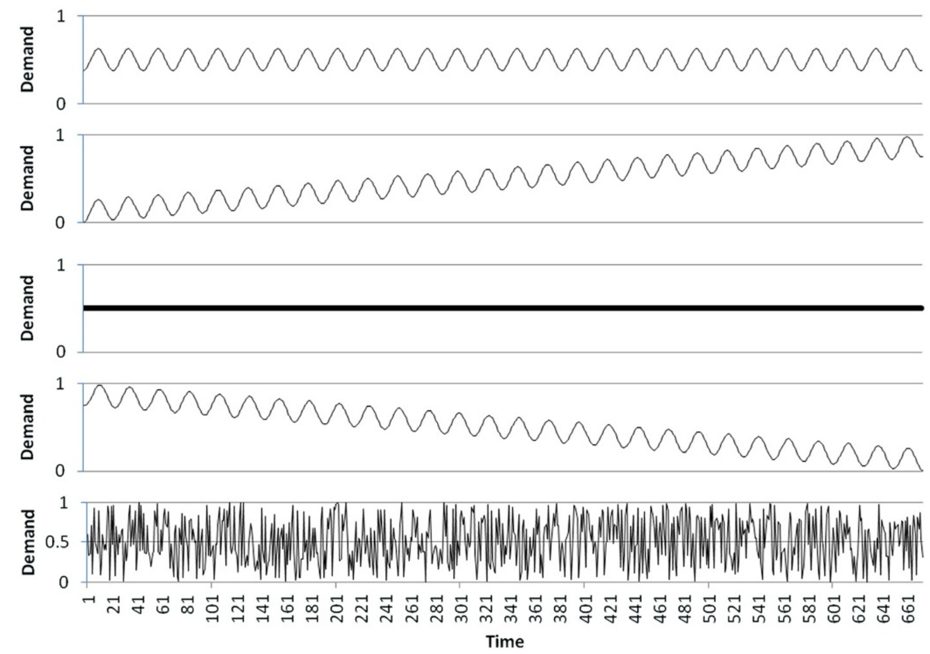

Figure 2 Demand profiles simulated over time period. Demand profiles from top to bottom are Flat Sine, Increasing Sine, Flat, Decreasing Sine and Random.

many different types of instance, of different quantities, in a time period. This is to simulate a population where some users might generally require smaller instances, while others may frequently require larger ones.

In this simulation, we assume that user *i* has a product demand split that is generated from a normal distribution defined by its mean μ_i, and its standard deviation σ_I, with the value then thresholded to clip it within the range [0,8], i.e.: $PDS_i = \min(\max(0,N(\mu_i,\sigma_i)),8)$. The mean represents the instance size that is most likely to be demanded by user *i*. The standard deviation represents the degree of variation in the distribution of instance sizes demanded by user *i*.

This method means that within any time period, a consumer will require a number of virtual machines of different types. For a user with $\mu_i = 4$ and $\sigma_i = 0.25$, it is likely that a large number of instance types of size 4 will be required in any time period. However, it is also possible they will require other instance types in the same time period, but due to the small standard deviation this will only occur rarely.

For a user with $\mu_i = 3$ and $\sigma_i = 6.0$, instances types of size 6 will be demanded most frequently in each time period. However, instance types of size 4 or 5 will also be demanded with roughly the same frequency as size 6 in any time period, due to the larger standard deviation

Each user's requirements are generated at random from the thresholded normal distribution defined by their individual PDS_i function. Although the demand patterns are smooth, individual user's PDS_i functions inject noise into the simulation as a result of the random selection of product demands from the normal distribution

This is a suitable function for modelling demand across virtual machine sizes, to a first approximation, for the following reasons. Consumers have two objectives when choosing a size of virtual machine: first, to ensure the virtual machine can meet the performance of the application running upon it; and second, to pay the least price to meet this performance. This reflects the fact that many cloud-computing customers have broadly similar performance requirements due to the prevalence of standard "stacks" of applications running on standard operating systems (such as the well-known "LAMP stack", involving Linux, Apache, MySQL, and Perl/PHP/Python), so it is probable that many consumers would choose the same virtual instance size that meets an acceptable level of performance for the least cost. Nevertheless, some consumers may need better performance, while others may need to reduce cost further; that is, some users have a preference for performance over price while others value price over performance. Consumers could start with a standard size of VM and then increase or reduce that, to meet their performance or cost requirements.

To determine their probability of future resource demands, our model users employ a simple prediction based on their immediate history of past usage: each user considers if a resource was needed at that hour over the past working three days. For example, if a resource was required at 11 am every day for the past 3 days, the user will reserve a resource with probability 1.0 that it will be used at 11 am tomorrow. If a resource was used two out of three periods, the user will purchase an options contract with probability 0.66 and so on. If the provider offers forward contracts rather than options, then resources can only be reserved with a probability of 1.0 if it has been used in all three periods.

The provider aggregates probabilities for each instance-size and assigns the expected number of virtual machines to each server.

Execution

The simulation was deployed on a commercial infrastructure-as-a-service service to allow parallel processing of multiple experiments[1].

To obtain a comparison with an on-demand only provisioning method, the simulation processed exactly the same requests for resources as for the combined-pricing method. However, each user's requests were randomly re-ordered to simulate the scenario of users purchasing on-demand resources, where the provider must start virtual machines immediately wherever space is available.

Results

To judge the performance of our model, we monitor two measures: the mean reduction in cost per computational unit when the users employ combined options and on-demand purchases; and the reduction in the number of servers needed to service the combined demands of the users. We express both measures as percentages in comparison to results from the same simulation model when configured to run a purely on-demand purchasing system. In this manner, we compare the performance of the models against each other, rather than analysing the absolute values of costs and active servers for each model.

When viewing our results for reduction in server usage, it is important to realise that the large numbers of servers in a typical cloud data-centre mean that a small percentage value, such as a one or two percent reduction, can nevertheless involve the freeing up of hundreds or thousands of server machines.

Figures 3 and 4 shows the mean server reduction and mean cost saving respectively using combined pricing via options contracts, as a percentage increase/decrease of the same measure from on-demand-only pricing, over the course of the simulation for each product demand split μ and market profile. Figures 5 and 6 show the same measures, but for combined pricing using forward contracts.

Server usage reduction

From Figure 3, it can be seen that the combined pricing method via options contracts offers improvements in server utilisation compared to on-demand only pricing, but

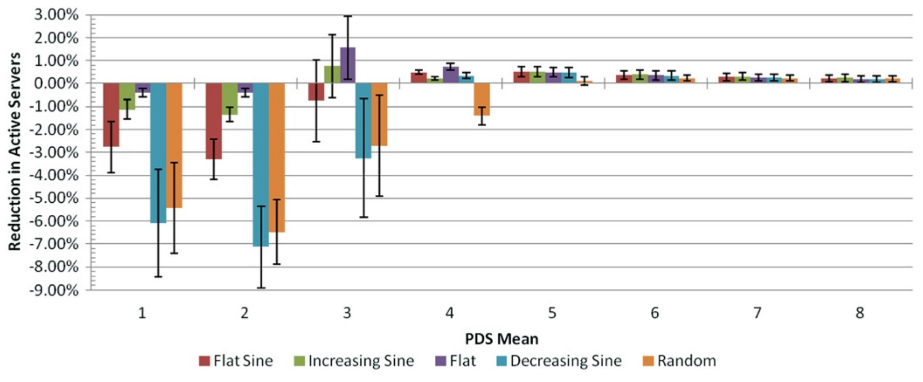

Figure 3 Mean server reduction using combined pricing with options-contracts compared to on-demand only pricing.

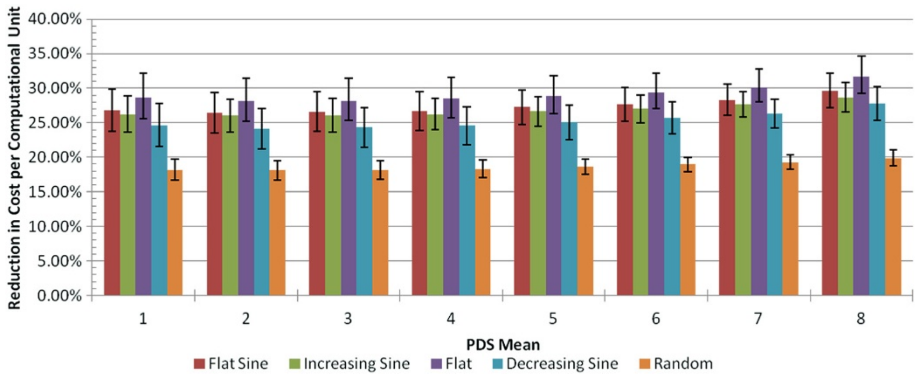

Figure 4 Mean cost saving using combined pricing with options-contracts compared to on-demand only pricing.

only when the mean product demand split is greater than three. This pattern is also seen in the case of combined pricing via forwards contracts (Figure 5). Furthermore, the combined pricing method using options contracts also makes improvements in server utilisation compared to traditional advance reservations (forward contracts) when the product demand split has a mean greater than three.

When the mean product demand split is three or less, the magnitude of the loss of utilisation for smaller instances is far greater than the gain in utilisation achieved when advance-scheduling larger instances. So the combined method using options contracts does have a benefit over both on-demand and advance reservations, but only when larger sized virtual machines are demanded by the user population. The provider must be confident that demand for larger resources is greater than that for smaller resources, or risk worsening the overall server utilisation.

Consider a partially utilised datacentre, receiving requests for instances in a random, on-demand fashion. Figure 7 shows such a scenario, where a datacentre of five servers is partially utilised by virtual machines of varying sizes. In this scenario, an instance of size three can be placed on any server. An instance of size four can only be placed on three of the already utilised servers. The instance of size five cannot fit on any partially utilised server, and a new server must be powered-on to support it.

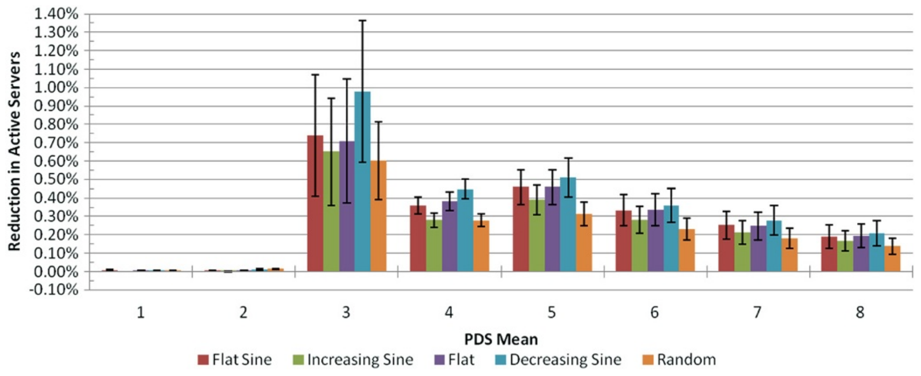

Figure 5 Mean server reduction using combined pricing with forwards-contracts compared to on-demand only pricing.

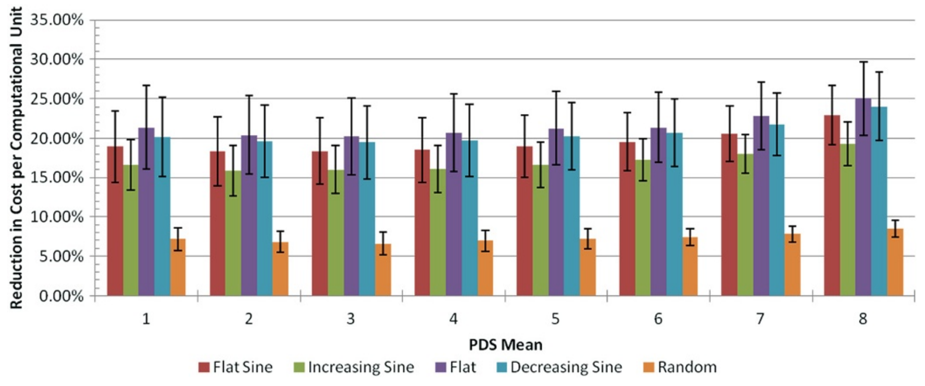

Figure 6 Mean cost saving using combined pricing with forwards-contracts compared to on-demand only pricing.

In a similar manner, when there is an abundance of smaller on-demand virtual machine requests arriving in random order, many of these smaller instances can be placed on already partially-utilised servers. However, when there is an abundance of larger on-demand virtual machine requests, it is more difficult to place them and additional servers must be activated.

As a result of this, the online first-fit algorithm schedules very efficiently when smaller instances are dominant.

The offline first-fit decreasing algorithm also performs very efficiently in advance scheduling of virtual machines. However, in the execution phase, some users may choose not to execute their options contracts. When this occurs, many servers are left under-utilised. The net effect is a drop in utilisation using combined-pricing compared to on-demand only pricing.

In these experiments, there is a 'tipping-point' when the product demand split mean is 4 where the loss in utilisation caused by user's choosing not to execute their options is less than the utilisation gained by advance scheduling. We believe the tipping-point is related

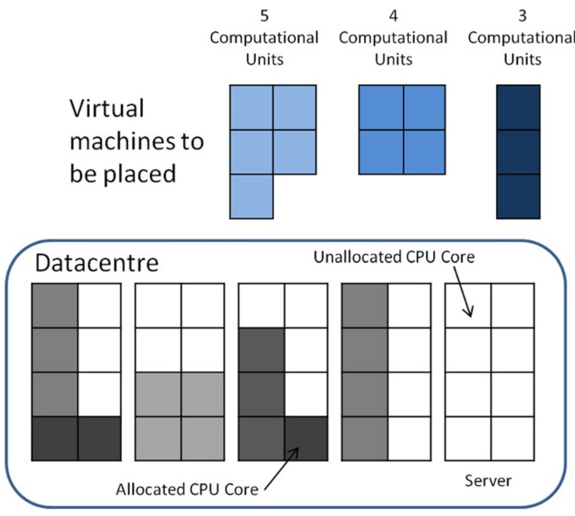

Figure 7 Schematic of on-line virtual machine scheduling in datacentres.

to the point at which there is a large quantity of instances of a size greater than 50% of the maximum server size. Instances that are larger than 50% of the maximum server size cannot be scheduled together, so efficiencies gained by advance scheduling these are likely to be greater that for instances smaller than 50% of the maximum server size.

Figure 3 shows that options contracts achieve better utilisation than forward contracts (Figure 5) for larger instance sizes. When options contracts are deployed, instances are allocated to servers in advance. Subsequently, some users will choose not to execute their contracts and their allocated instances will not be utilised. However, users requiring on-demand resources can be given access to these unused instances, which have been previously mapped in an efficient manner using the bin packing algorithms. In forward contracts, however, it is not possible to assign requests for on-demand resources to unutilised instances. As such, new servers must be started for these new instances and the random order of on-demand instances causes an inefficient mapping.

As the provider aggregates probabilities from the entire user population, if a user chooses not to execute their options, there is a good chance the mapped space will be filled by another consumer purchasing on-demand virtual machines.

Consumer cost reduction

Figures 4 and 6 show that consumers make a cost-saving regardless of profile, product demand split or instrument. However, consumers make a greater cost saving using options contracts (Figure 4) than forwards contracts (Figure 6), implying consumers would prefer to use the combined pricing model utilising options contracts.

The cost reduction experienced by consumers is equivalent to the loss in revenue to the provider. If cost saved through better utilisation (and therefore less server requirements) is greater than the revenue lost, then the provider will make a net improvement in profits. As we discuss in Section 5, the pricing equations can be changed to ensure a net improvement is obtained. However, the provider often loses revenue as a result of offering discounts to consumers which subsequently fail to deliver improved utilisation. As a result, the provider will lose revenue without gaining a tangible benefit in some situations, which is clearly undesirable.

Tweaking the pricing parameters is unlikely to remedy this, as the nature of the method is that consumer's requests must be fulfilled by the provider. In the next section, we develop an extension to this method, using provision-point mechanisms, as a method of protecting against lost revenue.

Provision-point contracts

In a *provision-point contract* (also known as an *assurance contract*), members of a group pledge to contribute to an action if some pre-specified threshold condition is met. Once the threshold point is passed, the action is taken and the public goods are provided; otherwise no party is required to carry out the action and the monies paid by the party are refunded [17].

Such a mechanism is used by deal-of-the-day website Groupon. Users make requests for special offers by purchasing a coupon. When a threshold is reached, the deal is profitable to the provider and the offer is confirmed. If the threshold is not reached, consumers are refunded whatever they initially paid for the coupon.

We propose that such a provision-point mechanism (PPM) can be used for increasing server utilisation by only confirming options or forward contracts if they are beneficial to the provider. In the next section, we describe the mechanism in more detail.

Mechanism

We propose adding a third period between the previous Reservation and Execution phase, whereby the provider confirms all contracts only if it is to her benefit.

Period 1 – Contract request

Users make requests for cloud resources to be consumed in the third period, paying a price P_{res} to make the request, as defined previously.

Period 2 – Contract confirmation

As in our original method, for each instance size, the provider aggregates the probabilities from the entire user population that a virtual machine of this size will be required at the time of expiry and uses a number of bin packing algorithms to determine the least number of servers required. However, additionally the provider attempts a random order fit.

If the provider finds that the randomised order uses fewer servers than the FFD or LBR algorithms, it is likely the distribution of resources is such that pre-scheduling is not beneficial. In this case, the provider will reject all requests and refund any monies paid in the first period.

Otherwise, the provider confirms all requests and contracts are established.

Period 3 – contract execution

If contracts were established, these users have a right to use the resource, and pay a price P_{exec} to access it. They may also buy on-demand instances at cost P_{od}

If only forward contracts were offered in the first period, users must pay for their resource whether they wish to utilize it not.

The provider supplies the user with access to the size of virtual machine requested as per the best-case allocation found in the second period.

If the provider has not mapped enough virtual machines to meet demand, it will start virtual machines where they will fit using the first-fit algorithm.

Unlike a traditional provision-point mechanism, the discounted rate is only available to those who submitted a reservation request previously.

Results

Figure 8 shows that when provision-point options contracts are offered, the mean server utilisation improves compared to using standard options when smaller instances are demanded. This is as a result of the provider rejecting contracts in some time periods where a loss is predicted, thereby improving the utilisation of smaller resources

Figure 8 also shows that when provision-point options contracts are offered, the mean server utilisation decreases slightly compared to using standard options when larger instances are demanded. This is due to the provider occasionally being overzealous

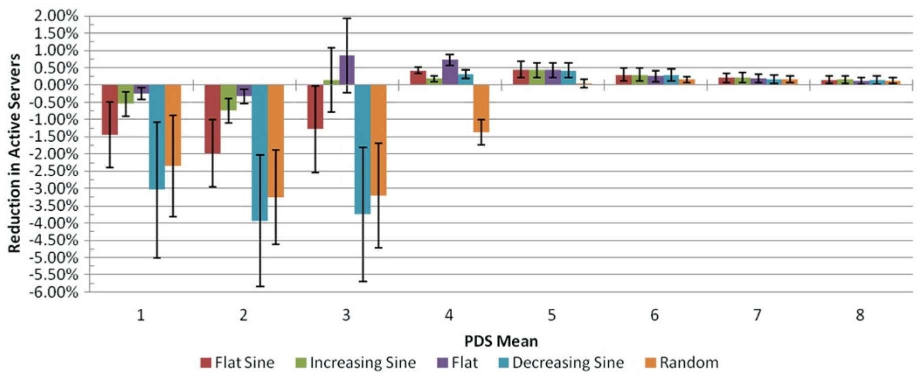

Figure 8 **Mean server reduction using combined pricing with provision point options-contracts compared to on-demand only pricing.**

and rejecting contracts where a benefit can be achieved, hence slightly reducing the utilisation achieved for larger resources.

When provision-point forward contracts are offered (Figure 9), the mean server utilisation improves compared to standard forwards. The provider rejects some contracts in some time periods which it believes will not be beneficial in terms of utilisation, thereby improving utilisation.

But does the use of provision-point contracts protect the provider's revenue? In Figure 10, it can be seen that when options or forward contracts result in improved server utilisation, consumers receive a larger discount. When these contracts are found not to be beneficial to the provider, consumers receive a significantly smaller discount. This protects the provider from reducing her revenue without gaining an advantage in terms of server utilisation.

Ideally, where options and forwards are not advantageous to the provider, consumers should receive no discount. However, the provider's method for determining if contracts should be established is not perfect, and occasionally the provider confirms contracts where it is subsequently found that no scheduling advantage is realised. This could potentially be improved by running more random trials prior to confirming contracts, or using a number of other bin packing algorithms.

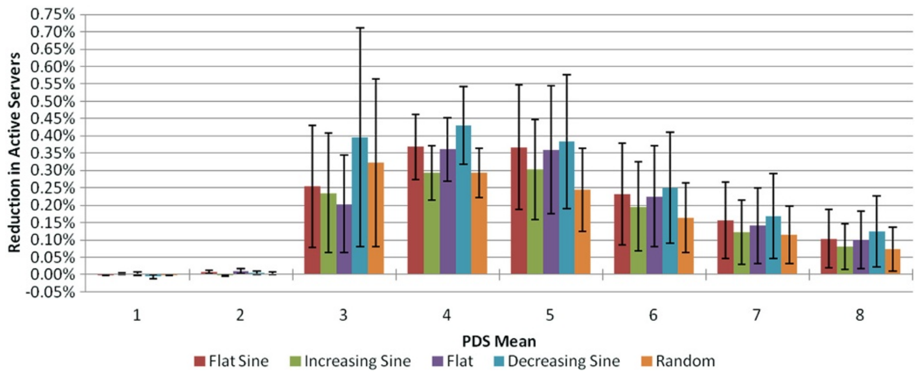

Figure 9 **Mean server reduction using combined pricing with provision point forward contracts compared to on-demand only pricing.**

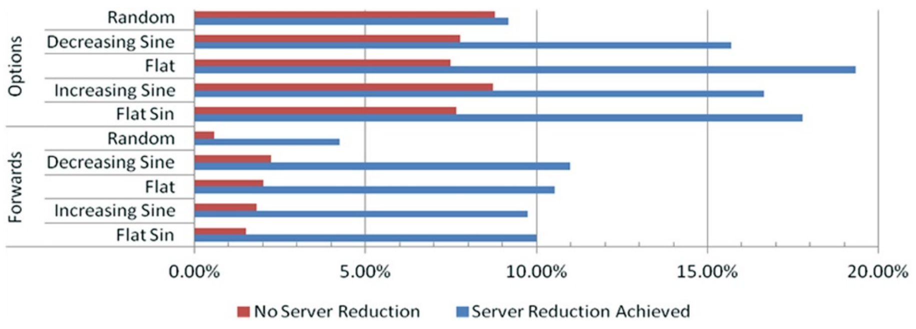

Figure 10 Summary of mean cost reduction using provision-point contracts over on-demand-only pricing.

Consumers receive a discount regardless of market profile, product demand split or contract type. This discount is greatest using provision-point contracts via options (Figure 11), which suggest consumers would prefer to use this scheme. However, it was particularly interesting to note the cost savings achieved by the consumers for pivot-point future contracts in Figure 12. Consumers are required to pay for the resource if they have reserved it, which might imply that they that pay more on average as a result of reserving resources which they subsequently do not use. However, the results show that consumers will make a saving per computation unit required in spite of sometimes reserving instances that they subsequently do not use. Significant cost savings are achieved by consumers for all markets. This indicates that consumers are likely to take advantage of the method.

Discussion

The modest percentage improvement in server utilisation is unlikely to benefit smaller providers of cloud services, who have a limited number of servers. However, these small percentage savings could translate into significant financial benefit for larger providers.

Furthermore, as consumers paid less per computational unit using combined pricing, it is likely that this would act as an incentive, drawing consumers into using the scheme, and giving the provider more accurate information on future demand.

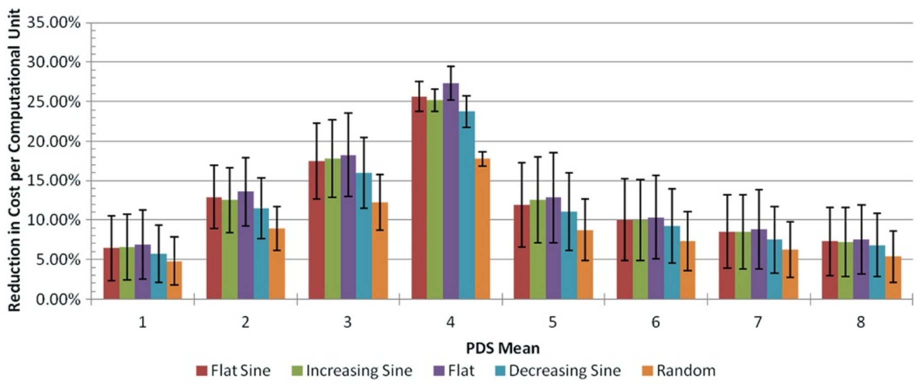

Figure 11 Mean cost saving using combined pricing with provision point options-contracts compared to on-demand only pricing.

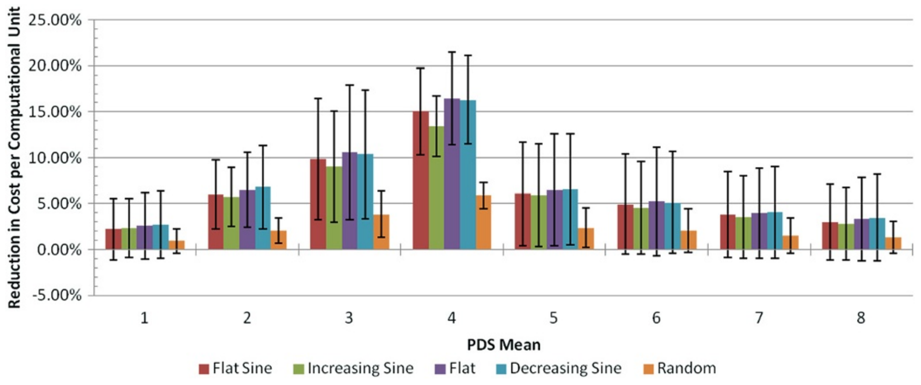

Figure 12 Mean cost saving using combined pricing with provision point futures contracts compared to on-demand only pricing.

The provider can use this information to reduce cost by:

(black circle) reducing the number of *physically located* servers in the datacentre;
(black circle) reducing the number of *powered-on* servers in the datacentre.

The former carries more risk than the latter. The provider would prefer to ensure they have enough servers to meet unexpected demand by some margin initially. However, over time as the provider understands patterns of increased growth, she could deploy more servers to meet this demand, which should be less than the demand before combined pricing was implemented.

The latter option is the easiest to implement and is suitable for providers who have already invested in infrastructure. Most commercial cloud providers, including the largest IaaS provider, Amazon Web Services (AWS), are reluctant to publish details about the size of their infrastructure. However, in a recent blog post, Huan Liu at Stanford University reported results from studies of IP address allocations to estimate the number of servers utilised across AWS [18]. He estimated AWS' total number of servers across the world at around 450,000, although many assumptions were made. Combined pricing would reduce the number of powered servers by 7065 in our best case scenario of a mean 1.57% reduction.

Using the online Hewlett-Packard electricity consumption calculator [19], a typical HP Proliant 140 server with two CPUs would use about 380 W at peak. At the typical cost of USD0.062 per kilowatt-hour [20], this translates to running costs on direct electricity consumption (excluding air conditioning, facilities, etc.) of around $206 per server per year. The combined pricing scheme used to power-off unused servers would save over $1.4 million per annum on direct electricity alone. This is equivalent to a reduction in coal-powered carbon production of 23,500 metric tons per annum [21].

In our full set of results, a saving of 5.58% was achieved using PPM with options contracts where the product split mean and standard deviation was 3 and 1 respectively. In this scenario, the saving would be $5.3 million per annum. This could increase significantly if other variable costs such as air conditioning and staffing are also reduced as a result of the pricing scheme's better matching of demand to supply.

Of course, the provider will lose some revenue as a result of offering a discount to consumers. If the cost reduction gained through the reduction in servers is greater than the revenue lost in discounts, then the method is worth implementing. The cost model can be simply tweaked such that consumers are incentivised whilst the provider makes a net saving through server consolidation, either by changing parameters or changing the pricing equations themselves. The key attribute of the pricing equations is that consumers are incentivised to submit truthfully – a detailed discussion of this can be found in [13].

Conclusions and future work

In this paper, we have shown how pricing methods based on probability-based options contracts can improve datacentre server utilisation, thereby decreasing the number of active servers required to support demand. This can have significant impact on electricity costs and/or carbon footprint.

In the initial approach of using only options contracts, we showed that this could benefit the provider when larger instances are demanded by the user population. However, when smaller instances are demanded there is a large drop in utilisation when using combined-pricing compared to on-demand-only pricing. This is due to inaccuracies in advance scheduling as a result of consumers choosing not to execute options contracts. In this scenario, the provider also loses revenue as a result of offering discounts to customers, which subsequently fails to provide a tangible benefit.

The addition of provision-point contracts, whereby the provider has the opportunity to analyse the requirements of the user population before committing herself to advance reservations, was shown to protect the provider's revenue to some degree. This is achieved by restricting the discount offered to consumers where the distribution of demand across the user population was not found to create better utilisation. Provision-point options contracts were found to suffer the same issue as standard options contracts, where dominance of smaller instances creates poor server utilisation. However, provision-point futures contracts were found to perform well for everyone. Consumers benefit from a mean cost-saving per computational unit, despite using a simplistic method to forecast usage and occasionally purchasing resources they subsequently do not use. The provider benefits through better utilisation regardless of the demand distribution from the population, and the provider does not lose a large amount of revenue when options and forwards contracts are unlikely to improve server utilisation.

We believe standard options and provision-point options contracts could benefit all parties, but the provider must be confident that larger virtual machine sizes are generally in more demand than smaller sizes, or she may suffer losses in utilisation.

The standard options contract scheme is the most attractive for the consumer, as they have confirmed access to a resource as soon as they make a request. However, the provider does not have to offer both schemes. If the provider finds that provision-point futures are more beneficial than options contracts, she can choose to only implement provision-point futures commercially. In this scenario, consumers are likely to use the scheme as it is the only method of obtaining cheaper resources from that provider. If the provider wishes to offer both schemes, she could set cheaper prices for the less attractive provision-point futures pricing scheme as an incentive. This is an area for future work.

Provision-point futures contracts seem to benefit all parties, with very little risk if pricing equations and parameters are chosen correctly such that a net cost reduction is made.

Currently, non-determinism in the simulation is introduced through the random selection of instance sizes chosen from the product demand split normal distribution. It would be interesting to add stochastic noise directly to the demand profile to see how unexpected changes in market demand affect the accuracy of user's forecasts, and subsequently the utilisation of resources. A range of noise distributions and their parameters could be assessed to determine at what point, if any, our pricing model ceases to offer its benefits.

In this paper, we focused our attention on the case where one computational unit is defined as one physical core. It would be useful to investigate the model where there are multiple computational units per core. In the most granular case, consumers could even specify exactly how many CPU cycles are required by a virtual machine. Using a typical 2 GHz CPU, consumers could specify how many clock cycles per second their virtual machine needs, between 1 and 2,000,000,000. Further analysis should also be conducted on how the model performs in heterogeneous datacentres where different server specifications are used.

The model needs to be tested thoroughly using a variety of user models, product demand splits and demand profiles. We believe our work could provide the basis for a commercial implementation.

Endnotes

We were initially surprised by the results of our simulations, and so verified the behaviour of the mechanism by manually running the algorithm for a smaller number of users and virtual machines, on a spreadsheet model. Through this laborious process, we found similar results. Furthermore, we compared log files created from different parts of the simulation code to ensure data integrity. We are confident that our results are realistic, and have released the code as open-source from http://www.lsctis.org/downloads to allow other researchers to review and/or re-use our work.

Competing interests
Owen Rogers' PhD studentship is part-funded by Hewlett-Packard Research Labs. Primary funding for both Rogers' and Cliff's work on this project was provided by a research grant from the UK Engineering and Physical Sciences Research Council. The authors list these sources of funding for completeness, and as a mark of acknowledgement and grateful thanks. Neither author considers these sources of funding to constitute competing or competing interests in the work.

Authors' contributions
OR designed and implemented the simulations, analysed the results, and wrote the first complete draft of this paper. The novels aspects of the combined pricing methods described in this paper are entirely OR's contribution. DC provided direction and advice, proposed research areas and reviewed and edited earlier drafts of this paper. Both authors read and approved the final manuscript.

Authors' information
Owen Rogers is a doctoral researcher at the University of Bristol. Owen graduated from Cardiff University with a MEng in Computer Systems Engineering in 2005. Following this, he joined telecommunications firm Cable & Wireless, first as a Graduate Engineer, and then as Product Development Manager for Managed Security Services. In 2009, Owen joined managed services provider Claranet as Product Portfolio Manager, before returning to academia in 2010. He is currently researching financial mechanisms for improving cloud computing's potential as a tradeable commodity. Owen is a Chartered Engineer through the British Computer Society.
Dave Cliff is a professor of computer science at the University of Bristol. He has previously held faculty positions at the universities of Sussex and Southampton in the UK, and at the MIT Artificial Intelligence Lab, USA. He also spent seven years working in industrial technology research, first for Hewlett-Packard Labs and then for Deutsche Bank's Foreign Exchange Complex Risk Group. Since 2005, Dave has been Director of the UK Large-Scale Complex IT Systems Initiative (www.lscits.org). He is a Fellow of the British Computer Society.

Acknowledgements
We thank the UK Engineering and Physical Sciences Research Council (EPSRC) for funding the Large-Scale Complex IT Systems Initiative (www.lscits.org) as well as HP Labs Adaptive Infrastructure Lab for providing additional financial support. We thank Claranet for use of their Virtual Data Centre cloud service.

Received: 17 August 2012 Accepted: 7 December 2012
Published: 12 December 2012

References
1. Weinman J (2011) Time is Money: The Value of 'On-Demand, pp 1–29, http://joeweinman.com/Resources/Joe_Weinman_Time_Is_Money.pdf." [Accessed 23-Apr-2012]
2. Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, Zaharia M (2010) Above the clouds: A Berkeley view of cloud computing cloud computing: an old idea whose time has (finally) come. Comm ACM 53(4):50–58
3. Sandholm T, Lai K, Andrade J, Odeberg J (2006) Market-based resource allocation using price prediction in a high performance computing grid for scientific applications. 15th IEEE Int Symp High Perform Distr Comput :132–143
4. Prodan R, Wieczorek M, Fard HM (Oct. 2011) Double auction-based scheduling of scientific applications in distributed grid and cloud environments. Journal of Grid Computing 9(4):531–548
5. Varian H, Shapiro C, Farrell J (2004) The Economics of Information Technology – An Introduction. Cambridge Publishing
6. Li A, Yang X (2010) CloudCmp: comparing public cloud providers, in IMC '10 Proceedings of the 10th ACM SIGCOMM conference on Internet measurement, pp 1–14
7. Johnson DS, Ullman JD, Gareyi MR, Grahamii RL (1974) Worst-case performance bounds for simple one-dimensional packing algorithms. SIAM J Comput 3(4):299–325
8. Coffman E, Garey M, Johnson D (1983) Dynamic bin packing. SIAM J Comput 12(2):227–258
9. Stage A, Setzer T (2009) Network-aware migration control and scheduling of differentiated virtual machine workloads, in 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing, pp 9–14
10. Dosa G (2007) The Tight Bound of First Fit Decreasing Bin-Packing Algorithm Is F F D (I) ≤ 11 / 9OP T (I) + 6 / 9. Combinatorics 2007(I):1–11
11. Martello S, Toth P (1990) Lower bounds and reduction procedures for the bin packing problem. Discrete Appl Math 28:59–70
12. Hull JC (2008) Fundamentals of Futures and Options Markets, 6th edn. Prentice Hall
13. Wu F, Zhang L, Huberman B (2008) Truth-telling reservations. Algorithmica 52(1):65–79
14. Rogers O, Cliff D (2010) The Effects of Truthfulness on a Computing Resource Options Market, in 2nd Annual International Conference on Advances in Distributed and Parallel Computing, pp 330–335
15. Rogers O, Cliff D (2012) A financial brokerage model for cloud computing. Journal of Cloud Computing: Advances, Systems and Applications 1(1):2
16. Jaillet P, Ronn EI, Tompaidis S (Jul. 2004) Valuation of commodity-based swing options. Manag Sci 50(7):909–921
17. Bagnolli M, Lipman B (1989) Provision of public goods: fully implementing the core through private contributions. Rev Econ Stud 56:583–601
18. Liu H (2102) Huan Liu's Blog. [Online]. Available: http://huanliu.wordpress.com/2012/03/13/amazon-data-center-size/. [Accessed: 23-Apr-2012]
19. HP Online Electricity Consumption Calculator. [Online]. Available: http://h20000.www2.hp.com/bizsupport/TechSupport/Document.jsp?lang=en&cc=us&taskId=125&prodSeriesId=428936&prodTypeId=15351&objectID=c01510445. [Accessed: 23-Apr-2012]
20. Barroso LA, Hölzle U (Jan. 2009) The datacenter as a computer: an introduction to the design of warehouse-scale machines. Synthesis Lectures on Computer Architecture 4(1):1–108
21. (2006) UK Parliamentary Office of Science and Technology, "Postnote: Carbon Footprint of Electricity Generation., London, Available: www.parliament.uk/documents/post/postpn268.pdf [Accessed 23-Apr-2012]

doi:10.1186/2192-113X-1-21
Cite this article as: Rogers and Cliff: **Options, forwards and provision-point contracts in improving cloud infrastructure utilisation.** *Journal of Cloud Computing: Advances, Systems and Applications* 2012 **1**:21.