The Rise of RaaS: the Resource-as-a-Service Cloud

Orna Agmon Ben-Yehuda[1] Muli Ben-Yehuda[1,2] Assaf Schuster[1] Dan Tsafrir[1]
[1]Technion—Israel Institute of Technology [2]Hypervisor Consulting and Technologies
{ladypine,muli,assaf,dan}@cs.technion.ac.il

ABSTRACT

Infrastructure-as-a-Service (IaaS) cloud providers typically sell virtual machines that bundle a fixed amount of resources, such as the core count, the memory size, and the I/O bandwidth. The resource bundles are usually unchanging throughout the lifetime of the virtual machines. We foresee that this type of rigid resource allocation will change in the near future. Instead of fixed bundles, cloud providers will increasingly sell resources individually, reprice them, and adjust their quantity every few seconds in accordance with market-driven supply-and-demand conditions; virtual machines will accordingly purchase and utilize the changing resources dynamically, while they are running. We term this nascent economic model of cloud computing the Resource-as-a-Service (RaaS) cloud, and we contend that its rise is the likely culmination of recent trends in the construction of IaaS clouds and of the economic forces operating on cloud providers and clients.

> "When the quantity of any commodity which is brought to market falls short of the effectual demand, [...] some [...] will be willing to give more."
> (Adam Smith, *An Inquiry into the Nature and Causes of the Wealth of Nations*)

Categories and Subject Descriptors

K.4.4 [**Computing Milieux**]: Computers and Society—*Electronic Commerce*

General Terms

Economics, Human Factors, Design

Keywords

cloud, trends

1. INTRODUCTION

Cloud computing is taking the computer world by storm. Today, Infrastructure-as-a-Service (IaaS) clouds, such as Amazon EC2, allow anyone with a credit card to tap into a seemingly unlimited fountain of computing resources by renting virtual machines for several cents or dollars per hour. According to a Forrester Research report [31], the yearly cloud computing market is expected to top $241 billion in 2020, compared to $40.7 billion in 2010, a sixfold increase. What will these 2020 clouds look like? Given the current pace of innovation in cloud computing and in other utilities such as smart grids and wireless spectra, substantial shifts are bound to occur in how providers design, operate, and sell cloud computing resources, and in how clients purchase and use those resources.

IaaS cloud providers sell fixed bundles of CPU, memory, and I/O resources packaged as server-equivalent virtual machines. We foresee that, instead, providers will continuously reprice and adjust the quantity of the individual resources with a time granularity as fine as seconds; the software stack within the virtual machines will accordingly evolve to productively operate in this dynamic, ever-changing environment. We call this new model of cloud computing the *Resource-as-a-Service (RaaS)* cloud. In a RaaS cloud, provider-governed economic mechanisms will control clients' access to resources. Hence, clients will deploy economic agents that will continuously buy and sell computing resources in accordance with the provider's current supplies and other clients' current demands.

We identify four existing trends in the operation of IaaS cloud computing platforms, that underlie the transition we foresee: the shrinking duration of rental, billing, and pricing periods (Section 2.1), the increasingly fine-grained resources offered for sale (Section 2.2), the increasingly market-driven pricing of resources (Section 2.3), and the provisioning of useful service level agreements (SLAs) (Section 2.4). We believe the economic forces operating on both providers and clients (Section 3) will continue pushing these trends forward. Eventually, as the trends near their culmination, today's IaaS cloud models will be pushed to the limit, and a paradigm shift will be required. We believe that these economic forces will unify today's IaaS cloud computing models into a single economic model of cloud computing that we call RaaS (Section 4). We conclude by outlining the challenges and opportunities the RaaS cloud presents (Section 5).

2. RECENT IAAS TRENDS

2.1 Duration of Rent and Pricing

Before cloud computing, the average useful lifetime of a purchased server was approximately three years. With the advent of Web hosting, clients could rent a server on a monthly basis. With the introduction of on-demand EC2 instances in 2006, Amazon radically changed the time granularity of server rental, making it possible to rent a server equivalent for as little as one hour. This move was good for the provider, because, by incentivizing the clients to shut down unneeded instances, it allowed for better time-sharing of the hardware. It also benefited the clients, who no longer needed to pay for *wall clock time* they did not use, but only for *instance time* that they did use.

This trend—of renting server-equivalents for increasingly shorter time durations—is driven by economic forces that keep pushing clients to improve efficiency and minimize waste: if a partial instance-hour is billed as a full hour, you might waste up to an hour over the lifetime of every virtual machine (a *per-machine penalty*). If a partial instance-second is billed as a full second, then you will only waste up to a second over the lifetime of every virtual machine. Thus, shorter durations of rent and shorter billing units reduce client overhead and open the cloud for business for shorter workloads. Notably, low overheads encourage *horizontal elasticity*— changing the number of concurrent virtual machines—and draw clients who require this functionality to the cloud.

The trend towards shorter times is also gaining ground with regard to pricing periods. Amazon spot-instances, announced in 2009, may be repriced as often as every five minutes [1], although they bill by the price at the beginning of the hour. CloudSigma, announced in 2010, reprices its resources exactly every five minutes.[1]

New providers charge by even finer time granularity: Gridspot[2] and ProfitBricks,[3] both launched in July 2012, charge by three-minute and one-minute chunks, respectively. Google App Engine's new policy is to bill instances by the minute, with a minimum charge of 15 minutes,[4] and as of May 2013 Google Compute Engine charges by the minute with a minimum of ten minutes instead of by hours.[5]

We draw an analogy between cloud providers and phone companies, which have progressed over the years from billing landlines per several minutes to billing cell phones by the minute, and then, due to customer pressure or court orders, to billing per several seconds and even per second. Similarly, car rental (by the day) is also giving way to car sharing (by the hour), and it is recommended that wireless spectrum sharing have a shorter period base [17].

We expect this trend of shortening times to continue such that eventually, cloud providers will reprice computing resources every few seconds and charge for them by the second. Providers might compensate themselves for overheads by charging a minimal amount or using *progressive prices* (higher unit-prices for shorter rental times). Such durations are consistent with peak demands that can change over seconds when a site is "slashdotted" (linked from a high-profile Web site).[6]

2.2 Resource Granularity

In most IaaS clouds, clients rent a fixed bundle of compute, memory, and I/O resources. Amazon and Rackspace[7] call these bundles "instance types," GoGrid[8] calls them "server sizes," and Google Compute Engine[9] calls them "machine types." Selling resources this way provides clients with a familiar abstraction of a server-equivalent. This abstraction is starting to unravel, and in its place we see the beginnings of a new trend towards finer and finer resource granularity. In August 2012,[10] Amazon began allowing clients to dynamically change the available I/O resources for already-running instances.[11] Google App Engine charges I/O operations by the million and offers progressive network prices, which are rounded down to small base units before charging (1 byte, 1 email, 1 instance-hour).[12] CloudSigma (2010), Gridspot (2012), and ProfitBricks (2012) offer clients the ability to compose a flexible bundle from varying amounts of resources, similar to building a custom-made server out of different mixtures of resources such as CPUs, memory, and I/O devices.

Renting a fixed combination of cloud resources cannot and does not reflect the interests of clients. First, as server size is likely to continue to increase—hundreds of cores and hundreds of gigabytes of memory per server in a few years—an entire server-equivalent may be too large for some customer needs. Second, selling a fixed combination of resources is only efficient when the load customers need to handle is both known in advance and constant. As neither condition is likely, the ability to dynamically mix-and-match different amounts of compute, memory, and I/O resources benefits the clients.

We expect this trend towards finer resource granularity to continue, such that all of the major resources (compute, memory, and I/O) will be rented and charged for in dynamically changing amounts and not in fixed bundles: clients will buy seed virtual machines with some initial amount of resources, and then supplement these initial allocations with additional resources as needed.

Studying these trends, we extrapolate that, in the near future, resources will be rented separately with fine resource granularity for short durations. As rental durations grow shorter, we expect efficient clients to automate the process by deploying an economic agent (described in Section 4), which will make decisions in accordance with the current prices of those resources, the changing load the machine should handle, and the client's subjective valuation of those different resources at different points in time. Such agents are also considered a necessary development in smart grids [30] and wireless spectrum [41] resource allocation. Two elements are likely to ease the adoption of economic agents: client size (larger clients are more likely to invest in systematic savings, which accumulate for them to large numbers), and the availability of agents that are off-the-shelf and customizable (e.g., open source).

2.3 Market-Driven Resource Pricing

Virtualization and machine consolidation are beneficial when at least some resources are shared (e.g., heat sink, bus, last-level cache), and others are time shared (e.g., when a fraction of a CPU is rented, or physical memory is overcommited). However, the performance of a given virtual machine can vary wildly at different times due to interference and bottle-

necks caused by other virtual machines that share resources whose use is not measured and allocated [14, 24, 35]. For example, Google App Engine's preliminary model, charging for CPU time only and not for memory, made the scaling of applications that use a lot of memory and little CPU time "cost-prohibitive to Google,"[13] because consolidation of such applications was hindered by memory bottlenecks. Hence, in 2011, Google App Engine was driven to charge for memory (by introducing memory-varied bundles), which became, as a result, a measured and allocated resource.

Moreover, interference and bottlenecks depend on the activity of all the virtual machines involved, and are not easily quantified in a live environment in which the guest can only monitor its own activity. Even after the guest benchmarks its performance as a function of the resource bundle it rented, neighbors sharing those same resources might still cause that performance to vary [35]. Thus, there is a discrepancy between what providers provide and what clients would actually like: in practice, what clients care about is their virtual machines' subjective performance.

To bridge this gap, researchers have proposed to sell client performance instead of consumed resources [4, 16, 24, 27]. This approach is only applicable where performance is well defined, and where the client applications are fully visible to the provider (as is the case in Software-as-a-Service (SaaS) and Platform-as-a-Service (PaaS) clouds), or the client virtual machines fully cooperate with the provider, as may be the case in private IaaS clouds. However, IaaS cloud providers and clients are separate economic entities. In general, they do not trust each other, and do not cooperate without good reason. Hence, guaranteeing client performance levels is not applicable to a public IaaS cloud, where allocated resources affect the performance of different applications differently, where the very definition of performance is subjective, where client virtual machines are opaque, and where the provider cannot rely on clients to tell the truth with regard to their desired and achieved performance. If the provider guarantees a certain performance level, it is in the client's interest to claim the performance is still too low, so that the provider will add resources.

We believe that public clouds will have to forsake the approach of charging users a predefined sum for resource bundles of unknown performance. For high-paying clients, providers can raise prices and forgo resource overcommitment. For low-paying clients, a cheap or free tier of unknown performance can be offered. However, for mid-range clients, providers will have to follow one of the following routes to handle the problem of unpredictable resource availability: (1) tackle the hard task of precisely measuring all the system's resources to quantify the real use each virtual machine made of them, and then charge the clients precisely for the resources they consumed, or (2) switch to a market-driven model.

A market-driven model is based on how clients value the few monitored resources. It does not necessitate precise measurement of resource use on the part of the provider—only the final outcome, the client's subjective valuation of its performance, matters. Clients, in turn, will have to develop their own model to determine the value of a smaller number of monitored resources. The model needs to implicitly factor in virtual machine interference over nonmonitored resources. For example, clients might use a learning algorithm that produces a time-local model for the connection between monitored resources and client performance. Though highly expressive, the client's model need not be complicated: it is enough that the client can adjust the model to the required accuracy level. Hence, the minimal client model can be as simple as a specific sum for a specific amount of resources: below these requirements, the client will not pay. Above them, the client will not add money. The client willingness to pay will affect the prices and the resource allocation. Unlike previously proposed models, this economic model can accommodate real-world, selfish, rational clients.

2.4 Tiered Service Levels

Tiered service [25], where different clients get different levels of service, can be found in certain scientific grids. Jobs of clients with low privileges may be preempted (aborted or suspended) by jobs of clients with higher privileges. Although clouds did not, at first, offer such prioritized service but rather supplied service at only one fixed level (on-demand), Amazon has since introduced different priority levels within EC2. The higher priority levels are accorded to the reserved (introduced March 2009) and on-demand instances. Spot instances (introduced December 2009) provide a continuum of lower service levels, since Amazon prioritizes spot instances according to the bid price stated by each client. Gridspot (2012) operates in a similar manner. As in grids, these priorities are relative, so it is hard to explicitly define their meaning in terms of absolute availability. For example, the availability of on-demand instances depends on the demand for reserved instances. The PaaS provider Dotcloud (announced in 2010)[14] and Google App Engine[15] also offer different SLA levels for different fees.

Having clients with different priorities is useful to the provider, who can provide high-priority clients with elasticity and availability at the expense of lower-priority clients, while simultaneously renting out currently-spare resources to low-priority clients when high-priority clients do not need them. Likewise, different priorities allow budget-constrained cloud clients cheap access to computing resources with poorer availability. Mixing clients of different relative priorities will allow the providers to simultaneously achieve high resource utilization and maintain adequate spare capacity for handling sudden loads.

Extrapolating from the progression of SLA terms we have seen to-date, we expect that in the RaaS cloud clients will be able to define their own priority level, choosing from a relatively priced continuum. Moreover, if prices are market driven, and priority levels reflect the client's willingness to pay, then we expect that clients will be able to change their desired priority levels as often as prices change.

It is possible to extend the prevalent SLA language—"unavailability of a minimal period X, which is at least a fraction Y of a service period Z"—to express different absolute levels by controlling the parameters X,Y,Z [4]. Yet, we extrapolate that as more cloud providers adopt flexible SLAs, they will continue the existing trend of relative priorities, and not venture into extending the absolute SLA language to several tiers.

3. ECONOMIC DYNAMICS

In the previous section, we surveyed several ongoing trends and tried to surmise where they will lead us next. We now survey the economic forces operating on clients and on providers and their implications. We believe these forces caused the phenomena previously discussed and will continue pushing today's IaaS clouds forward until they will have to undergo a paradigm shift. We believe this paradigm shift is likely to turn them into RaaS clouds.

3.1 Forces Acting on Clients

As clients purchase more cloud services, their bill increases. When bills are large, clients seek systematic savings. The best way to achieve this is by paying only for the resources they need, and only when they need them. When clients are able to adjust the resources they rent to match the resources they utilize, their effective utilization rises, and the cost per utilized resource drops, potentially by $50\% - 85\%$, depending on the resource utilization [6]. The more flexible the provider offerings, the better control clients have over their costs and the resulting performance. As providers offer increasingly fine-grained resources and service levels, clients are incentivized to develop or adopt resource provisioning methods. As the time scales involved shorten, manual provisioning methods become tedious, increasing the clients' incentive to rely on computerized provisioning agents [39] to act on their behalf.

3.2 Forces Acting on Providers

Competition between IaaS cloud providers is increasing, as indicated by recent cloud price reductions. During previous years, Amazon reduced its prices in correlation with new instance type announcements, and only by 15%, while hardware costs dropped by 80% [36]. However, as shown in Figure 1, the timing of price cuts in 2012 by three major cloud providers is correlated, a phenomenon referred to as a "cloud price war".[16]

The competition is aided by the commoditization of cloud computing platforms. Commoditization eases application porting between providers. An example for such commoditization is the open source OpenStack,[17] which is the foundation of both Rackspace's public cloud and HP's. OpenStack also offers Amazon EC2/S3 compatible APIs. As changing providers becomes easier, and as hungry new providers join the fray, competition increases and providers are forced to lower prices.

3.3 Implications of Increased Competition

As competition increases and prices decrease, providers attempt to cut their costs,[18] in an effort to maintain their profit margins. At any moment, given the available revenue-creating client workload, the provider seeks to minimize its costs (in particular, power costs) by idling or halting some machines or parts thereof [11]. It does so by consolidating instances to as few physical machines as reasonably possible. When resources are overcommited due to consolidation and clients suddenly wish to use more resources than are physically available on the machine, the result is resource pressure.

The move towards tiered service and fine rental granularity is driven, in part, by the need to reduce costs and the accompanying resource pressure. When clients change their resource consumption on the fly, providers who continue to guarantee absolute QoS levels have to reserve a conservative amount of headroom for each resource on each physical server. This headroom—spare resources—is required just in case all clients simultaneously require all the resources promised them. Clients who change their resource consumption on-the-fly do not pay for this headroom unless and until they need it, so keeping it around all the time is wasteful.

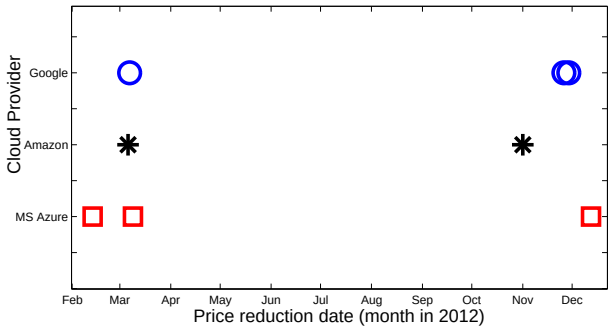


Figure 1: Correlated cloud price reduction dates for three major cloud providers during 2012

Under the fixed bundles model, if the host chooses to overcommit resources, some clients will get less than the bundle they paid for. If the headroom is too small and there is resource pressure, this underprovisioning will be felt by the client in the form of reduced performance, and the illusion of a fixed bundle will be broken.

Extending the current absolute SLA language to several tiers only reduces some of the headroom. To get rid of the headroom completely, providers must resort to prioritization via tiered service levels, which only guarantees clients relative QoS. Relative QoS requires that clients change their approach. Relative QoS should thus be introduced gradually, allowing clients to control the risk to which they agree to be exposed.

Here is a concrete example of how a provider might nowadays waste its resources, and how a future provider might increase the utilization of its powered-up servers and reduce its power costs. Let us consider a 4GB physical machine, running an instance that once required 3GB of memory, and now only uses 2GB. A new client would like to rent an instance with 2GB. Under the IaaS model, the new client cannot be accommodated on this machine. 1GB goes unsold, and 2GB go unused. With tiered SLAs and dynamic resources, the first client can temporarily reduce its holdings to 2GB, and the provider then can rent 2GB to the new client. If conflicts arise later due to memory shortage, the provider can choose how much memory to rent to each client on the basis of economic considerations. No memory goes unused, and no extra physical server needs to be booted.

4. THE RAAS CLOUD

We have presented the distinct trends operating in IaaS clouds, along with the economic forces that govern them. We believe that the combined effect of all these trends and forces is leading to a qualitative transformation of the IaaS cloud

into what we call the *Resource-as-a-Service (RaaS) cloud*. We present here our unified view of the RaaS cloud, and discuss possible steps on the path to its realization.

4.1 Trading in Fine-Grained Resources

Seed virtual machine. In RaaS clouds, the client purchases upon admittance a seed virtual machine. The seed virtual machine has a minimal initial amount of dedicated resources. All other resources needed for the efficient intended operation of the virtual machine are continuously rented. This combination of resource rental schemes—prepurchasing and multiple on-demand levels—benefits the clients with the flexibility of choice.

Fine-grained resources. The resources available for rent include CPU, RAM, and I/O resources, as well as emerging resources such as computational accelerators (e.g., GPGPUs and FPGAs) and Flash devices. CPU capacity is sold on a hardware-thread basis, or even as number of cycles per unit of time; RAM is sold on the basis of memory frames; I/O is sold on the basis of subsets of I/O devices with associated I/O bandwidth and latency guarantees. Such devices include network interfaces and block interfaces. Accelerators are sold both as I/O devices and as CPUs. A subset of an I/O device may be presented to the virtual machine as a direct-assigned SR-IOV Virtual Function(VF) [13] or as an emulated [3] or para-virtual device. Every resource comes with a dynamically changing price tag. Resource rental contracts are set for a minimal fixed period, which does not have to coincide with the repricing period. The host may offer the guests renewal of their rental contract at the same price for an additional fixed period.

Host economic coordinator. To facilitate continuous trading, the provider's cloud software includes an economic coordinator representing the provider's interests. This coordinator operates an economic mechanism which defines the resource allocation and billing mechanism: which client gets which resources and at what price. Several auctions were proposed to such ends, e.g., by Kelly [20], Chun and Culler [7], Lazar and Semret [22], Waldspurger et al. [38], and Lubin et al. [23]. In addition, the coordinator may act as a clearing house and support a secondary market of computing resources inside the physical machine, as SpotCloud[19] offers to do for fixed-bundle virtual machines and as Kash et al. [19] propose to do for the wireless spectrum.

Guest economic agent. To take part in auctions or trade, clients' virtual machines must include an economic agent. This agent represents the client's business needs. It rents the necessary resources—given current requirements, load and costs—at the best possible prices, from either the provider or its *neighbors*—virtual machines collocated on the same physical machine, possibly belonging to different clients. When demand outstrips supply, the agent changes its *bidding strategy* (in cases where the provider runs an *auction*) or negotiates with neighbors' agents, mediating between the client's requirements and the resources available in the system, ultimately deciding how much to offer to pay for each resource at any given time.

Subletting. Clients can secure resources early and sublet them later if they no longer need them. The resource securing can be done either by actively renting resources long term or by negotiating a future contract with the host. Either way, resource subletting also lays the ground for resource futures markets among clients. Clients can sublet to other clients on the same physical machine using infrastructure provided by the host's coordinator: the clients agree to redivide resources between them and inform the coordinator, who transfers the local resources from one guest to another (as Hu et al. [18] do for bandwidth resources). In addition to trading with a limited number of neighbors, clients can sublet excess resources to anyone, in the form of nested full virtual machines [5], a concept which is gaining more and more support. Examples resembling subletting exist today in the Amazon EC2 Reserved Instance Marketplace,[20] in CloudSigma's reseller option,[21] in the Deutsche Börse's Industry-First, Vendor-Neutral Cloud Marketplace,[22] and in DotCloud, which is reselling EC2's resources with an added value.[23] The subletting option reduces the risk for clients who commit in advance to rent resources. It also partially relieves the provider of the burden of retail sales, improves its utilization, and can increase its revenue through seller fees.[24] Allowing clients to sublet can also be considered as a loss leader: a feature that attracts clients by reducing their financial risk.

Legacy clients. IaaS providers can introduce RaaS capabilities gradually, without forcing their clients to change their business logic. Legacy clients, without an economic agent, can still function in the RaaS cloud just as they do in an IaaS cloud. They can simply rent large RaaS seed machines, which serve as IaaS instances. IaaS virtual machines function in a RaaS cloud just as well as they do in an IaaS cloud. However, to get the RaaS benefits of vertical elasticity and reduced costs, clients will need to adapt.

4.2 Prioritized Service Levels

Priorities for headroom only. In the RaaS cloud, the client gets an absolute guarantee (for receiving the resources and for the price paid) only for its minimal consumption, which is constant. Additional resources are provided on a priority basis in market prices. A risk-averse client can prepay for a larger amount of constant resources, trading low costs for peace of mind. From the provider's point of view, the aggregate constant consumption provides a steady income source. Only resources which may go unused (the headroom) are allocated on the basis of market competition.

Vertical elasticity: Robin Hood in reverse. RaaS clients are offered on-the-fly, fine-grained, fine-timed *vertical elasticity* for each instance: the ability to expand and shrink the resource consumption of each virtual machine. The resources required for this vertical elasticity are limited by the physical resources contained in a single machine, because mi-

grating running virtual machines from one physical machine to another will likely remain less efficient than dynamically balancing the available resources between virtual machines co-existing on the same physical machines. Hence, during peak demand times, to enable one client to vertically upscale a virtual machine, the additional resources must be taken from a *neighbor*. Instead of static priorities, in the RaaS cloud providers use the willingness of clients to pay a certain price for resources at a given moment (e.g., bids) to decide which client gets which resource. Thus market forces dictate both the constantly changing prices of resources and their allocation. In effect, the RaaS cloud provider does the opposite of Robin Hood: it takes from the poor and gives to the rich.

A few good neighbors. The RaaS virtual machine's vertical elasticity is determined, via a market mechanism, by its neighbors' willingness to pay. The neighbors also determine the cost of the elastic expansion. Due to the inherent inefficiencies of live virtual machine migration, RaaS clouds must include an algorithm for placing client virtual machines on physical machines. The algorithm should achieve the right mixture of clients with different SLAs on each physical machine in the cloud, such that high-priority clients always have low-priority clients beside them, to provide them with more capacity when their demands peak. The low-paying clients can use the high-paying clients' leftover resources when they do not need them, keeping the provider's machines constantly utilized. Another objective of the allocation algorithm is to allow the low-priority clients enough aggregate resources for their needs. A low priority client can be expected to tolerate a temporary loss of service every so often, but if the physical resources are strictly smaller than the mean demand, such a client will never get enough resources to make meaningful progress. Therefore, to retain the low-priority clients, the placement algorithm must provide them with enough resources to make (some) progress.

Full house. The RaaS provider also influences the quality of service that the RaaS client experiences by limiting the *maximal possible aggregate demand* for physical resources on the machine. Demand can be limited by controlling the number of virtual machines per physical machine and the maximal vertical elasticity to which each virtual machine is entitled. When the maximal possible aggregate demand is lower than the supply, resources are wasted, but all virtual machines can freely expand. As the maximal possible aggregate demand exceeds the supply, clients will be less likely to succeed in vertical expansion when they need it, or might be forced to pay more for the same expansion. Hence, RaaS clients are willing to pay more to be hosted in a physical machine with lower maximal possible aggregate demand. This encourages RaaS providers to expose information about the aggregate demand and supply on the physical machine to its clients.

5. IMPLICATIONS, CHALLENGES, OPPORTUNITIES

The RaaS cloud gives rise to a number of implications, challenges, and opportunities for both providers and clients, which did not exist in markets of entire virtual machines [2, 26, 29, 33, 34, 40]. Broadly speaking, the new research areas can be divided into two categories: technical mechanisms and policies.

The RaaS cloud requires new mechanisms for allocating, metering, charging for, reclaiming, and redistributing CPU, memory and I/O resources between untrusted, not-necessarily-cooperative clients every few seconds. These mechanisms must be efficient and reliable. In particular, they must be resistant to side-channel attacks from malicious clients [32]. Hardware mechanisms are especially needed for fine-grained resource metering in the RaaS cloud.

The RaaS cloud requires new system software and new applications. Usually, current operating systems and applications are written under the assumption that their underlying resources are fixed and always available. In the RaaS cloud, virtual machines never know the precise amount of resources that will be available to them at any given second. Thus, the software running in those virtual machines must adapt to changing resource availability and exploit whatever resources the software has, when it has them. Assume a client application that just got an extra 2Gbps of networking bandwidth at a steal of a price, but only for one second. To use it effectively while it is available, all the software layers, including the operating system, run-time layer, and application must be aware of it.

The RaaS cloud requires efficient methods of balancing resources within a single physical machine, while taking into consideration the different guaranteed service levels. Bottleneck-resource allocation [10, 12, 15] is a step towards allocation of resource bundles, but it still requires an algorithm for setting the system share to which each client is entitled.

The resource balancers are most efficient when guests with different service levels are collocated on the same physical server. Hence, workload balancers, which balance resources across entire cloud data centers, will need to consider the virtual machines, flexibility and SLA in addition to the current considerations (static resource rquests only).

Under dynamic conditions, the intra-machine RaaS mechanisms will quickly respond to flexibility needs, holding the fort until the slower live migration can take place. However, live migration must take place to mitigate the resource pressure on the effectively most stressed machines, and allow clients to change their flexibility bounds. Large IaaS providers apparently manage without live migration [32]: the high rate of initialization and shutdown of virtual machines makes the initial balancer effective enough. However, the fine time granularity of the changes in the RaaS cloud means live migration is going to be required more often. Hence, the RaaS cloud will require efficient methods for live migration of virtual machines and for network virtualization.

On the policy side, the RaaS cloud requires new economic models for deciding what to allocate, when to allocate it, and at what prices [8]. Ideally, they should optimize the provider's revenue or a social welfare function: a function of the benefit of all the clients. The clients may measure their benefit in terms of starvation, latency, or throughput, but the mechanisms should optimize the impact of those performance metrics on the welfare of the clients, for example

by maximizing the sum of client benefits or by minimizing the unhappiness of the most unsatisfied client.

These new economic models should also consider that resources may complete or substitute one another in different ways for different clients. For one client resources might be *economic complements*: if, for each thread the application requires 1GB RAM and 1 core, a client renting 2GB and 2 cores will only be interested in adding a combination of 1GB and 1 core. For another client, resources might be *economic substitutes*: every additional GB allows the application to cache enough previous results to require one core less. So when cores are expensive, a client that is renting 2GB and 2 cores will be able to release one core and rent another GB instead.

These mechanisms should be incentive compatible: truth telling regarding private information should be a good course of action for the clients, so that the provider can easily optimize the resource allocations. The mechanisms should be collusion-resistant: a virtual machine should not suffer if several of the virtual machines it is co-located with happen to belong to the same client. Like approximation algorithms for multi-unit auctions [9, 37], they should be computationally efficient at large scale, so that solving the resource allocation problem does not become prohibitive.

The mechanisms should preserve the clients' privacy as well as minimize the price-of-anarchy [21]: the waste incurred by using a distributed mechanism. Moreover, in order to work in the real world, the economic mechanisms must accommodate realistic clients' willingness to pay, which is a function of their performance measurements. The mechanism must support such measured functions, which are not necessarily mathematically nice and regular (e.g., contain steps [28]). Another real-world demand is simplicity. If researchers combined some of the works mentioned above to create a cumbersome mechanism with satisfactory theoretical qualities, that still would not guarantee its acceptance by the market: the providers and the clients.

In conclusion, making the RaaS cloud a reality will require solving hard problems spanning the entire gamut from game theory and economic models to system software and architecture. The onus is now on us, the cloud computing research community, to lead the way and build the mechanisms and policies that will make the RaaS cloud a reality.

6. ACKNOWLEDGMENTS

We thank Sharon Kessler for insightful discussions.

7. REFERENCES

[1] Agmon Ben-Yehuda, O., Ben-Yehuda, M., Schuster, A., and Tsafrir, D. Deconstructing Amazon EC2 spot instance pricing. In *IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom)* (2011).

[2] Altmann, J., Courcoubetis, C., Stamoulis, G., Dramitinos, M., Rayna, T., Risch, M., and Bannink, C. GridEcon: A market place for computing resources. In *Grid Economics and Business Models*, vol. 5206 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2008, pp. 185–196.

[3] Amit, N., Ben-Yehuda, M., Tsafrir, D., and Schuster, A. vIOMMU: efficient IOMMU emulation. In *USENIX Annual Technical Conference (ATC)* (2011).

[4] Baset, S. A. Cloud SLAs: Present and future. *ACM SIGOPS Operating Systems Review (OSR) 46*, 2 (Jul 2012).

[5] Ben-Yehuda, M., Day, M. D., Dubitzky, Z., Factor, M., Har'El, N., Gordon, A., Liguori, A., Wasserman, O., and Yassour, B.-A. The Turtles project: Design and implementation of nested virtualization. In *Symposium on Operating Systems Design & Implementation (OSDI)* (2010), pp. 423–436.

[6] Chen, Y., and Sion, R. To cloud or not to cloud?: musings on costs and viability. In *ACM Symposium on Cloud Computing (SOCC)* (2011).

[7] Chun, B. N., and Culler, D. E. Market-based proportional resource sharing for clusters. Tech. rep., Berkeley, CA, USA, 2000.

[8] Danak, A., and Mannor, S. Resource allocation with supply adjustment in distributed computing systems. In *International Conference on Distributed Computing Systems (ICDCS)* (2010).

[9] Dobzinski, S., and Nisan, N. Mechanisms for multi-unit auctions. *Journal of Artificial Intelligence Research 37* (2010), 85–98.

[10] Dolev, D., Feitelson, D. G., Halpern, J. Y., Kupferman, R., and Linial, N. No justified complaints: on fair sharing of multiple resources. In *Innovations in Theoretical Computer Science Conference* (2012).

[11] Gandhi, A., Harchol-Balter, M., and Kozuch, M. A. Are sleep states effective in data centers? In *International Green Computing Conference (IGCC)* (2012).

[12] Ghodsi, A., Zaharia, M., Hindman, B., Konwinski, A., Shenker, S., and Stoica, I. Dominant resource fairness: Fair allocation of multiple resource types. In *USENIX Conference on Networked Systems Design and Implementation (NSDI)* (2011).

[13] Gordon, A., Amit, N., Har'El, N., Ben-Yehuda, M., Landau, A., Tsafrir, D., and Schuster, A. ELI: Bare-metal performance for I/O virtualization. In *ACM Architectural Support for Programming Languages & Operating Systems (ASPLOS)* (2012).

[14] Gupta, D., Lee, S., Vrable, M., Savage, S., Snoeren, A. C., Varghese, G., Voelker, G. M., and Vahdat, A. Difference engine: harnessing memory redundancy in virtual machines. In *Symposium on Operating Systems Design & Implementation (OSDI)* (2008).

[15] Gutman, A., and Nisan, N. Fair allocation without trade. In *Intl. Conference on Autonomous Agents and Multiagent Systems (AAMAS)* (2012).

[16] Heo, J., Zhu, X., Padala, P., and Wang, Z. Memory overbooking and dynamic control of Xen virtual machines in consolidated environments. In *Symposium on Integrated Network Management (IM)* (2009), pp. 630–637.

[17] Holdren, J. P., and Lander, E. Realizing the full potential of government-held spectrum to spur economic growth. Tech. rep., The President's Council of Advisors on Science and Technology, 2012.

[18] Hu, L., Ryu, K. D., Silva, D. D., and Schwan, K. v-bundle: Flexible group resource offerings in clouds. In *International Conference on Distributed Computing Systems (ICDCS)* (2012).

[19] Kash, I. A., Murty, R., and Parkes, D. C. Enabling spectrum sharing in secondary market auctions. In *Workshop on the Economics of Networks, Systems, and Computation* (2011).

[20] Kelly, F. Charging and rate control for elastic traffic. *European Transactions on Telecommunications 8* (1997).

[21] Koutsoupias, E., and Papadimitriou, C. Worst-case equilibria. In *Symposium on Theoretical Aspects of Computer Science* (1999), pp. 404–413.

[22] Lazar, A., and Semret, N. Design and analysis of the progressive second price auction for network bandwidth sharing. *Telecommunication Systems—Special Issue on Network Economics* (1999).

[23] Lubin, B., Parkes, D. C., Kephart, J., and Das, R. Expressive power-based resource allocation for data centers. In *International Joint Conference on Artificial Intelligence (IJCAI)* (2009).

[24] Nathuji, R., Kansal, A., and Ghaffarkhah, A. Q-Clouds: Managing performance interference effects for QoS-aware clouds. In *ACM SIGOPS European Conference on Computer Systems (EuroSys)* (2010).

[25] Odlyzko, A. Paris metro pricing for the internet. In *Proceedings of the 1st ACM Conference on Electronic Commerce* (New York, NY, USA, 1999), EC '99, ACM, pp. 140–147.

[26] Ortuno, F. M., and Harder, U. Stochastic calculus model for the spot price of computing power. In *Annual UK Performance Engineering Workshop (UKPEW)* (2010).

[27] Padala, P., Hou, K.-Y., Shin, K. G., Zhu, X., Uysal, M., Wang, Z., Singhal, S., and Merchant, A. Automated control of multiple virtualized resources. In *ACM SIGOPS European Conference on Computer Systems (EuroSys)* (2009).

[28] Parkes, D. C., Procaccia, A. D., and Shah, N. Beyond dominant resource fairness: Extensions, limitations, and indivisibilities. In *The ACM Conference on Electronic Commerce (EC)* (2012).

[29] Rahman, M. R., Lu, Y., and Gupta, I. Risk aware resource allocation for clouds. Tech. rep., University of Illinois at Urbana-Champaign, 2011.

[30] Ramchurn, S. D., Vytelingum, P., Rogers, A., and Jennings, N. R. Putting the 'smarts' into the smart grid: a grand challenge for artificial intelligence. *Commun. ACM 55*, 4 (Apr. 2012), 86–97.

[31] Ried, S., Kisker, H., Matzke, P., Bartels, A., and Lisserman, M. Sizing the cloud—understanding and quantifying the future of cloud computing. Tech. rep., Forrester, 2011.

[32] Ristenpart, T., Tromer, E., Shacham, H., and Savage, S. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *ACM Conference on Computer and Communications Security (CCS)* (2009).

[33] Shneidman, J., Ng, C., Parkes, D. C., Auyoung, A., Snoeren, A. C., Vahdat, A., and Chun, B. Why markets could (but don't currently) solve resource allocation problems in systems. In *USENIX Workshop on Hot Topics in Operating Systems (HOTOS)* (2005), p. 7.

[34] Vanmechelen, K., Depoorter, W., and Broeckhove, J. Combining futures and spot markets: A hybrid market approach to economic grid resource management. *Journal of Grid Computing 9* (2011), 81–94.

[35] Verma, A., Ahuja, P., and Neogi, A. Power-aware dynamic placement of hpc applications. In *ACM Int'l Conference on Supercomputing (ICS)* (2008).

[36] Vermeersch, K. A broker for cost-efficient qos aware resource allocation in EC2. Master's thesis, Universiteit Antwerpen, 2011.

[37] Vöcking, B. A universally-truthful approximation scheme for multi-unit auctions. In *Annual ACM-SIAM Symposium on Discrete Algorithms* (2012).

[38] Waldspurger, C. A., Hogg, T., Huberman, B. A., Kephart, J. O., and Stornetta, W. S. Spawn: A distributed computational economy. *IEEE Transactions on Software Engineering* (Feb 1992).

[39] Yi, S., Kondo, D., and Andrzejak, A. Reducing costs of spot instances via checkpointing in the Amazon Elastic Compute Cloud. In *IEEE International Conference on Cloud Computing (CLOUD)* (2010).

[40] Zaman, S., and Grosu, D. Combinatorial auction-based dynamic vm provisioning and allocation in clouds. In *IEEE International Conference on Cloud Computing Technology and Science (CloudCom)* (2011).

[41] Zhou, X., G, S., Suri, S., and Zheng, H. eBay in the sky: Strategy-proof wireless spectrum auctions. In *ACM International Conference on Mobile Computing and Networking (MobiCom)* (2008).

Notes

[1] http://www.cloudsigma.com

[2] http://gridspot.com

[3] http://www.profitbricks.com

[4] https://developers.google.com/appengine/kb/billing#time_granularity_instance_pricing, accessed December 2012.

[5] https://cloud.google.com/pricing/compute-engine, accessed May 2013.

[6] "Fifty percent of the time the site is down in seconds—even when we've contacted site owners and they've told us everything will be fine. It's often an unprecedented amount of traffic, and they don't have the required capacity."–Stephen Fry, http://tinyurl.com/StephenFrySeconds.

[7] http://www.rackspace.com/cloud/public/servers/techdetails/

[8] http://www.gogrid.com

[9] https://cloud.google.com/pricing/compute-engine

[10] http://aws.amazon.com/about-aws/newsletters/2012/08/14/august-2012/

[11] http://aws.amazon.com/ebs/

[12] https://developers.google.com/appengine/kb/billing

[13] Greg D'Alesandre, http://tinyurl.com/D-Alesandre .

[14] https://www.dotcloud.com/pricing.html

[15] https://cloud.google.com/pricing/

[16] http://tinyurl.com/cloud-price-war

[17] http://openstack.org

[18] James Hamilton, "Amazon cycle of innovation" slide, http://tinyurl.com/james-hamilton

[19] http://spotcloud.com

[20] http://aws.amazon.com/ec2/reserved-instances/marketplace/

[21] http://www.cloudsigma.com/cloud-computing/what-is-the-cloud/171

[22] http://www.cloudsigma.com/about-us/press-releases/242

[23] http://docs.dotcloud.com/0.9/faq/

[24] http://aws.amazon.com/ec2/reserved-instances/marketplace/