DOI:10.1145/2627422

In the RaaS cloud, virtual machines trade in fine-grain resources on the fly.

BY ORNA AGMON BEN-YEHUDA, MULI BEN-YEHUDA, ASSAF SCHUSTER, AND DAN TSAFRIR

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

CLOUD COMPUTING IS taking the computer world by storm. Infrastructure-as-a-Service (IaaS) clouds (such as Amazon Elastic Compute Cloud, and 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. Forrester Research[30] predicted the cloud computing market could top $241 billion in 2020, compared to $40.7 billion in 2010, a sixfold increase. What will the 2020 clouds look like? Given the pace of innovation in cloud computing and other utilities (such as smart grids and wireless spectra), substantial shifts are bound to occur in the way providers design, operate, and sell cloud computing resources and how clients purchase and use them.

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

IaaS Trends

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; the constantly decreasing granularity of resources offered for sale; the increasingly market-driven pricing of resources; and the provisioning of useful service level agreements (SLAs).

Duration of rent and pricing. Before cloud computing emerged over the past decade, the useful lifetime of a purchased server was several 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 for its clients to rent a "server equivalent" for as short a period as one hour. This

» key insights

- **Current trends toward increased flexibility and efficiency in IaaS clouds will force a business model paradigm shift on the cloud storage industry.**
- **The RaaS cloud is the likely result of this shift.**
- **The RaaS cloud uses economic mechanisms within physical machines.**

IMAGE COLLAGE BY ANDRIJ BORYS ASSOCIATES

move was good for the provider—Amazon—because, by incentivizing clients to shut down unneeded instances, the hardware was time-shared better. It also benefited clients, who no longer had to pay for wall-clock time they did not use but only for instance time they did use.

Renting server-equivalents for ever-shorter periods 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, a client might waste up to one hour over the lifetime of every virtual machine (a per-machine penalty). If a partial instance-second were billed as a full second, then the client would waste only up to one second over the lifetime of each virtual machine. Shorter periods of rent and shorter billing units reduce client overhead, opening the cloud for business for shorter workloads. Low overhead encourages horizontal elasticity—changing the number of concurrent virtual machines—and draws clients that require this functionality to the cloud.

The trend toward 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 Amazon bills by the price at the beginning of the hour. CloudSigma, an IaaS cloud provider launched in 2010, reprices its resources exactly every five minutes;[a] see the sidebar "For More on RaaS."

Newer providers charge by even finer time granularity; for example, Gridspot[b] and ProfitBricks,[c] both launched in 2012, charge by three-minute and one-minute chunks, respectively. Meanwhile, Google modified its pricing policies; as of June 2011, Google App Engine bills instances by the minute, with a minimum charge of 15 minutes,[d] and as of May 2013 Google Compute Engine charges by the minute, with a minimum of 10 minutes, instead of by the hour.[e]

We draw an analogy between cloud providers and phone companies that have progressed from billing landlines per several minutes to billing cellphones by the minute and then, due to client pressure or court order, to billing per several seconds and even per second. Likewise, car rental (by the day) is also giving way to car sharing (by the hour), and the U.S. President's Council of Advisors on Science and Technology recommends that wireless spectrum sharing have a shorter period base.[18]

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

Resource granularity. In most IaaS clouds, clients rent a fixed bundle of compute, memory, and I/O resources. Amazon and Rackspace[g] call these bundles "instance types"; GoGrid[h] calls them "server sizes"; and Google Compute Engine[i] calls them "machine types." Selling resources this way gives clients a familiar abstraction of a server equivalent. However, this abstraction is starting to unravel, and in its place are the beginnings of a new trend toward finer resource granularity. In August 2012,[j] Amazon began allowing clients to dynamically change available I/O resources for already-running instances.[k] Google App Engine charges for I/O operations by the million and offers progressive network prices rounded down to small base units before charging (such as 1B, one email message, and one instance-hour).[l] CloudSigma (2010), Gridspot (2012), and ProfitBricks (2012) offer their clients the ability to compose a flexible bundle from varying amounts of resources, similar to building a custom-made server from different combinations of resources (such as CPUs, memory, and I/O devices).

Renting a fixed combination of cloud resources 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 the next 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 clients.

We expect this trend toward increasingly finer resource granularity to continue, so all major resources—compute, memory, and I/O—will be rented and charged for in dynamically changing amounts, not in fixed bundles; clients will buy seed virtual machines with some initial amount of resources, supplementing them with additional resources as needed.

Following these trends, we extrapolate that resources in the near future will be rented separately with fine resource granularity for short periods. As rental periods grow shorter, we expect efficient clients to automate the process by deploying an economic software agent to make decisions in accordance with the current prices of the resources, the changing load the machine should handle, and the client's subjective valuation of the different resources at different times. Such agents are also considered a necessary development in smart grids[29] and in wireless spectrum[40] resource allocation. Two elements are likely to ease adoption of economic agents: client size, whereby larger clients are more likely to invest in systematic ways to save money, which accumulates for them to large amounts, and availability of off-the-shelf and customizable agents (such as open source ones).

Market-driven resource pricing. Virtualization and machine consolidation are beneficial when at least some resources are shared (such as heat sink, bus, and last-level cache) and others are time-shared (such as when a fraction of a CPU is rented or physical memory is overcommitted). However, the performance of a given virtual machine can vary wildly over time due to interference and bottlenecks caused by other virtual machines that share resources whose use is not measured and allocated;[15,24,34] 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,"[m] because consolidation of such applications was hindered by memory bottlenecks. In 2011, Google App Engine was thus driven to charge for memory (by introducing memory-varied bundles). As a result, memory became a measured and allocated resource.

Moreover, interference and bottlenecks depend on the activity of all the virtual machines in the system and are not easily quantified in a live environment in which guests can monitor only their own activity. Even after the guest machine benchmarks its performance as a function of the resource bundle it rented, neighbors sharing the same resources might still cause performance to vary.[34] There is thus a discrepancy between what providers provide and what clients actually prefer; in practice, what clients care about is the subjective performance of their virtual machines.

To bridge this gap, researchers have proposed selling *guest performance* instead of consumed resources.[5,17,24,26] This approach is applicable only where performance is well defined and client applications are fully visible to the provider, as in Software-as-a-Service and Platform-as-a-Service clouds, or the client virtual machines fully cooperate with the provider, as can happen in private IaaS clouds. However, IaaS cloud providers and clients are separate economic entities and do not in general trust one another or cooperate without good reason. Guaranteeing client performance levels is thus not applicable to a public IaaS cloud where allocated resources affect the performance of different applications differently, the very definition of performance is subjective, client virtual machines are opaque, and 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 to motivate the provider to add resources.

More on RaaS

To delve further into the trends behind our vision of the RaaS cloud, see:

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

[b] http://gridspot.com

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

[d] https://developers.google.com/appengine/kb/billing#time_granularity_instance_pricing

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

[f] "50% 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, actor and widely followed Twitter user, London, U.K.; http://tinyurl.com/StephenFrySeconds

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

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

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

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

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

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

[m] Greg D'Alesandre, Google App Engine; http://tinyurl.com/D-Alesandre

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

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

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

[q] http://openstack.org

[r] James Hamilton, Amazon Web Services, slide: "Amazon Cycle of Innovation"; http://tinyurl.com/james-hamilton

[s] http://spotcloud.com

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

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

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

[w] http://tinyurl.com/6fusion-CME

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

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

Public clouds will have to forsake charging users a predefined sum for resource bundles of unknown performance. For high-paying clients, providers can raise prices and forgo overcommitting resources. 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 two possible routes to address the problem of unpredictable resource availability: precisely measure all system resources to quantify the real use each virtual machine makes of them and then charge the clients precisely for the resources they consumed; or 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 matters—the client's subjective valuation of the performance. Clients, in turn, will have to develop their own model to determine the value of a smaller number of monitored resources. Such a model should implicitly factor in virtual-machine interference over non-monitored resources; for example, clients might use a learning algorithm that produces a time-local model of 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. The minimal client model can thus be as simple as a specific sum for a specific amount of resources; below these requirements, the client will not pay at all, and above them, the client will not pay more. The client's willingness to pay affects prices and resource allocation. Unlike previously proposed models,[5,17,24,26] this economic model can accommodate real-world, selfish, rational clients.

Tiered service. Tiered service,[25] in which different clients get different levels of service, is found in certain scientific grids. Jobs of low-priority clients may be preempted (aborted or suspended) by jobs of high-priority clients. Although a decade ago clouds did not offer such prioritized service but supplied service at only one fixed level—on-demand—Amazon has since introduced various priority levels within EC2. Higher priority levels are accorded to 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 price bid by each client. Gridspot (2012) operates in a similar way. As in grids, these priorities are relative, so it is difficult to explicitly define their meaning in terms of absolute availability; for example, availability of on-demand instances depends on demand for reserved instances. The PaaS provider Docker (announced in 2010 as dotCloud)[n] and Google App Engine[o] also offer different SLA levels at different fee levels.

Providers that prioritize clients 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 inexpensive access to computing resources with poorer availability assurances. Mixing high-priority and low-priority clients will allow providers to simultaneously achieve high resource utilization and maintain adequate spare capacity for handling sudden loads.

Extrapolating from the progression of SLA terms we see, clients in the RaaS cloud 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 clients' willingness to pay, then we expect clients to 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*, and *Z*.[5] However, we extrapolate that as more cloud providers adopt flexible SLAs, they will continue the existing trend of relative priorities and not venture into extending absolute SLA language to several tiers.

Economic Dynamics

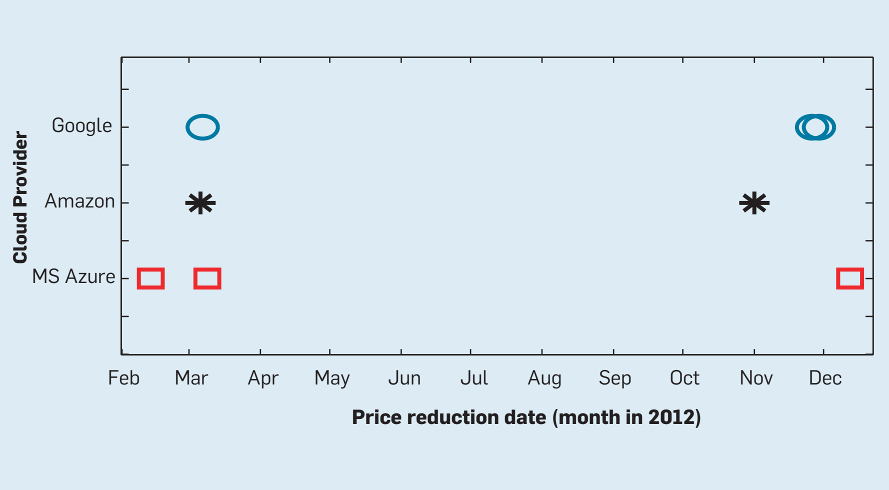
We have considered several ongoing trends, trying to anticipate where they will take the market next. We now survey the economic forces operating on clients and providers, along with their implications. These forces caused the phenomena discussed earlier and will continue pushing today's IaaS clouds along until, inevitably, they undergo a paradigm shift that is likely to turn them into RaaS clouds.

Forces acting on clients. As clients purchase more cloud services, their cloud bill increases. When bills are large, clients seek systematic savings. The best way to do so is by paying only for the resources they need, only when they need them. When clients are able to adjust the resources they rent to match the resources they use, their effective utilization rises, and the cost per utilized resource drops, potentially by 50%–85%, depending on resource utilization.[7] The more flexible the provider offerings, the greater control clients have over their costs and resulting performance. As providers offer increasingly fine-grain resources and service levels, clients are incentivized to develop or adopt resource-provisioning methods. As time scales shorten, manual provisioning methods become tedious, increasing clients' incentive to rely on computerized provisioning agents[38] to act on their behalf.

Forces acting on providers. Competition among IaaS cloud providers is increasing, as indicated by recent cloud price reductions. During the early years—2006–2011—Amazon reduced its prices as it announced new instance types, but by only 15%, while Amazon's hardware costs dropped by 80%.[35] However, the timing of price cuts in 2012 by three major cloud providers was correlated (see the figure here), a phenomenon called a "cloud price war."[p]

Competition is driven in part by commoditization of cloud-computing platforms. Commoditization eases application porting between providers; an example is the open source OpenStack,[q] the foundation of both Rackspace's and Hewlett-Packard's public clouds. OpenStack also offers Amazon EC2/S3-compatible APIs. As changing providers becomes easier, and as hungry new providers enter the market, competition increases and providers are forced to lower their prices.

Correlated cloud-price reduction dates for three major cloud providers, 2012.



Implications of increased competition. As competition increases and prices decrease, providers attempt to cut their costs[r] in an effort to maintain their profit margins. At any moment, given the available revenue-creating client workload, providers seek to minimize their costs (especially power costs) by idling or halting some machines or components[12] by consolidating instances to as few physical machines as reasonably possible. When resources are overcommitted 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 toward tiered service and fine rental granularity is driven in part by the need to reduce costs and accompanying resource pressure. When clients change their resource consumption on the fly, providers continuing to guarantee absolute QoS levels must reserve a conservative amount of headroom for each resource on each physical server. Such spare resources are required just in case all clients simultaneously require all the resources promised them. Clients changing their resource consumption on the fly do not pay for this headroom unless and until they need it, so making it available all the time is wasteful.

Under the fixed-bundles model, if the host (hypervisor) chooses to overcommit resources, some clients would 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 dissipate.

Extending the current absolute SLA language to several tiers reduces only some of the headroom. To eliminate headroom completely, providers must resort to prioritization via tiered service levels, guaranteeing clients only relative QoS. But because relative QoS requires that clients change their approach, it should be introduced gradually, allowing them to control the risk to which they agreed to be exposed.

Here is a concrete example of how a traditional provider might waste its resources and a future provider increase utilization of its powered-up servers and reduce its power costs. Consider a 4GB physical machine running an instance that once required 3GB of memory but now uses only 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 can rent 2GB to the new client. If conflicts arise later due to a shortage of memory, 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 has to be booted.

The RaaS Cloud

We have outlined the distinct trends operating in IaaS clouds, along with the economic forces governing them. Their combined effect is leading to a qualitative transformation of the IaaS cloud into what we call the RaaS cloud. The following is our view of the RaaS cloud, along with possible steps on the path to its realization:

Trading in fine-grain resources. A RaaS cloud machine defines the rules and mechanisms of an economic resource-trading environment, in which the economic entities operate:

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

Fine-grain resources. Resources available for rent include CPU, RAM, and I/O, as well as special resources (such as computational accelerators like GPGPUs, 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 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 single-root input/output virtualization virtual function (SF-IOV VF)[14] or as an emulated[4] or para-virtual device. A dynamic price tag is attached to every resource. Resource rental contracts are set for a minimal fixed period that need not coincide with the repricing period. The host may offer the guest machines renewal of their rental contracts at the same price for an additional fixed period.

In the RaaS cloud, providers leverage the variable willingness of clients to pay a certain price for resources at a given moment (as can be expressed by bids) to decide which client gets which resource.

Host economic coordinator. To facilitate continuous trading, the provider's cloud software includes an economic coordinator representing the provider's interests. It operates an economic mechanism that defines the resource-allocation and billing mechanism—which client gets which resources at what price. Several auctions have been proposed to such ends by, for example, Agmon Ben-Yehuda et al. for the RaaS cloud,[2] as well as Chun and Culler,[8] Kelly,[21] Lazar and Semret,[22] Lubin et al.,[23] and Waldspurger et al.[37] In addition, the coordinator may act as a clearing house and support a secondary market of computing resources inside the physical machine, as SpotCloud[s] does for fixed-bundle virtual machines and Kash et al.[20] proposed for the wireless spectrum.

Guest economic agent. To take part in auctions or trade, clients' virtual machines must include an economic agent representing the client's business interests. 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 co-located 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 neighbor machines' 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 a given time.

Subletting. Clients can secure resources early and sublet them later if they no longer need them. 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 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 among them and inform the coordinator, which transfers the local resources from one guest to another, as Hu et al.[19] did 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,[6] a concept now gaining support. Examples of secondary compute-resource trade exist in the Amazon EC2 Reserved Instance Marketplace,[t] in CloudSigma's reseller option,[u] in Deutsche Börse's vendor-neutral cloud marketplace,[v] in CME Group's plans for an IaaS commodity Exchange,[w] and in Docker, which resells EC2's resources with added value.[x] The subletting option reduces the risk for clients who commit in advance to rent resources. It also partially relieves the provider from having to manage retail sales while improving utilization and possibly increasing revenue through seller fees.[y] Allowing clients to sublet can also be viewed 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 clients to change their business logic. Legacy clients without an economic agent can still function in the RaaS cloud as they do in an IaaS cloud. They simply rent large RaaS seed machines serving as IaaS instances. IaaS virtual machines function in a RaaS cloud as well as they do in an IaaS cloud. However, to realize the RaaS benefits of vertical elasticity and reduced costs, clients must adapt.

Prioritized service levels. The economic mechanisms in the RaaS cloud determine various aspects of the relative service levels:

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

In the RaaS cloud, virtual machines never know the precise amount of resources that will be available to them at any given moment.

Vertical elasticity. RaaS clients are offered on-the-fly, fine-grain, fine-timed vertical elasticity for each instance—the ability to expand and shrink the resource consumption of each virtual machine. The resources required for vertical elasticity are limited by the physical resources in a single machine, because migrating running virtual machines between physical machines is likely to remain less efficient than dynamically balancing available resources between virtual machines coexisting on the same physical machines. Hence, to enable a client to vertically upscale a virtual machine during peak-demand times, the additional resources must be taken from a neighbor.

In the RaaS cloud, providers leverage the variable willingness of clients to pay a certain price for resources at a given moment (as can be expressed through bids) to decide which client gets which resource. Market forces thus dictate the constantly changing prices of resources as well as their allocation. In effect, the RaaS cloud provider does the opposite of Robin Hood by taking from the poor and giving to the rich.

A few good neighbors. The RaaS virtual machine's vertical elasticity is determined through 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. This 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 besides themselves to provide them with greater capacity when their demand peaks. 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 low-priority clients enough aggregate resources for their needs. A low-priority client is thus expected to tolerate a temporary loss of service every so often, but if the physical resources are strictly less than the mean demand, such a client would never get enough resources to make meaningful progress. To retain low-priority clients, the placement algorithm must thus provide them enough resources to make (some) progress.

Full house. The RaaS provider also influences the QoS 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 less than the supply, resources are wasted, but all virtual machines can freely expand. When the "maximal possible aggregate demand" exceeds supply, clients are less likely to succeed in vertical expansion when they need it or might be forced to pay more for the same expansion. RaaS clients are thus willing to pay more to be hosted in a physical machine with lower "maximal possible aggregate demand." This trade-off encourages RaaS providers to expose information about the aggregate demand and supply on the physical machine to its clients.

Implications, Challenges, Opportunities

The RaaS cloud gives rise to a number of implications, challenges, and opportunities for providers and clients alike that did not exist in markets involving entire virtual machines.[3,28,32,33,39] 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 among multiple untrusted, not-necessarily cooperative clients every few seconds.[2] These mechanisms must be efficient and reliable. In particular, they must be resistant to side-channel attacks from malicious clients.[31] Hardware mechanisms are a must for fine-grain resource metering in the RaaS cloud.

The RaaS cloud requires new system software and new applications. Operating systems and applications are generally written under the assumption 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 moment. The software running on those virtual machines must therefore adapt to changing resource availability and exploit whatever resources the software has, when it has them. Assume a client application just got an extra 2Gbps of networking bandwidth at a steal of a price but only for one second. To use it effectively, as it is available, all the software layers, including the operating system, runtime layer, and application, must be aware of it.

The RaaS cloud requires efficient methods of balancing resources within a single physical machine while accounting for the various guaranteed service levels. Bottleneck resource allocation[11,13,16] is a step toward allocation of resource bundles but still requires an algorithm for setting the system share to which each client is entitled.

Resource balancers are most efficient when guest machines with different service levels are co-located on the same physical server. Workload balancers, which balance resources across entire cloud data centers, will need to consider the flexibility and SLA of virtual machines in addition to the current considerations—static resource requests 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 resource pressure on the most stressed machines, allowing clients to vertically expand. Large IaaS providers apparently manage without live migration,[31] as 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 will be required more often. The RaaS cloud will thus 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.[9] Ideally, these models should optimize the provider's revenue or a social welfare function, a function of the benefit of all clients. The clients may measure their benefit in terms of starvation, latency, or throughput, but the mechanisms should optimize the effect of these metrics on the welfare of the clients by, say, maximizing the sum of client benefits or minimizing the unhappiness of the most unsatisfied client.

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

These allocation and pricing mechanisms should be incentive compatible; truth telling regarding private information should be a good course of action for clients so the provider can easily optimize resource allocations. The mechanisms should also 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,[10,36] they should be computationally efficient at large scale, so addressing the resource-allocation problem does not become prohibitive.

The mechanisms should preserve client privacy, as well as minimize the waste incurred by using a distributed mechanism. Moreover, in order to work in the real world, economic mechanisms must accommodate realistic client willingness to pay, which is a function of clients' performance measurements. The mechanism must support such measured functions, which are not necessarily mathematically nice and regular; in particular, they may contain steps.[27] Another real-world demand is simplicity. If researchers combined some of the ideas mentioned here to create a cumbersome mechanism with satisfactory theoretical qualities, that would still not guarantee its acceptance by the market of providers and their clients.

Conclusion

Making the RaaS cloud a reality requires solving problems spanning everything from game theory and economic models to system software and architecture. The onus is on the cloud-computing research community to lead the way, building the mechanisms and policies that will make the RaaS cloud a reality.

Acknowledgment

We thank Sharon Kessler of Pardes-Hanna, Israel for insightful discussions. This research was partially supported by the Prof. A. PaziIAEC-UPBC Joint Research Foundation.

References

1. Agmon Ben-Yehuda, A., Ben-Yehuda, M., Schuster, A., and Tsafrir, D. Deconstructing Amazon EC2 spot instance pricing. *ACM Transactions on Economics and Computation 1*, 3 (Sept. 2013), 1–16.
2. Agmon Ben-Yehuda, A., Posener, E., Ben-Yehuda, M., Schuster, A., and Mu'alem, A. Ginseng: Market-driven memory allocation. In *Proceedings of the 10th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments* (Salt Lake City, UT, Mar.). ACM Press, New York, 2014, 41–52.
3. Altmann, J., Courcoubetis, C., Stamoulis, G., Dramitinos, M., Rayna, T., Risch, M., and Bannink, C. GridEcon: A marketplace for computing resources. In *Proceedings of Grid Economics and Business Models, Volume 5206 of Lecture Notes in Computer Science* (Las Palmas de Gran Canaria, Spain, Aug.). Springer, Berlin/Heidelberg, 2008, 185–196.
4. Amit, N., Ben-Yehuda, M., Tsafrir, D., and Schuster, A. vIOMMU: Efficient IOMMU emulation. In *Proceedings of the USENIX Annual Technical Conference* (Portland, OR, June). USNIX Association, Berkeley, CA, 2011.
5. Baset, S.A. *Cloud SLAs: Present and future. ACM SIGOPS Operating Systems Review 46*, 2 (July 2012), 57–66.
6. 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 *Proceedings of the Symposium on Operating Systems Design and Implementation* (Vancouver, BC). USNIX Association, Berkeley, CA, 2010, 423–436.
7. Chen, Y. and Sion, R. To cloud or not to cloud?: Musings on costs and viability. In *Proceedings of the Second ACM Symposium on Cloud Computing.* ACM Press, New York, 2011.
8. Chun, B.N. and Culler, D.E. *Market-based Proportional Resource Sharing for Clusters. Technical Report.* University of California, Berkeley, 2000; http://www.cs.berkeley.edu/~culler/papers/market.pdf
9. Danak, A. and Mannor, S. Resource allocation with supply adjustment in distributed computing systems. In *Proceedings of the International Conference on Distributed Computing Systems* (Genova, June). IEEE Computer Society, 2010, 498–506.
10. Dobzinski, S. and Nisan, N. Mechanisms for multi-unit auctions. *Journal of Artificial Intelligence Research 37* (2010), 85–98.
11. Dolev, D., Feitelson, D.G., Halpern, J.Y., Kupferman, R., and Linial, N. No justified complaints: On fair sharing of multiple resources. In *Proceedings of the Innovations in Theoretical Computer Science Conference* (Boston). ACM Press, New York, 2012, 68–75.
12. Gandhi, A., Harchol-Balter, M., and Kozuch, M.A. Are sleep states effective in data centers? In *Proceedings of the International Green Computing Conference* (San Jose, CA, June). IEEE Computer Society, 2012, 1–10.
13. Ghodsi, A., Zaharia, M., Hindman, B., Konwinski, A., Shenker, S., and Stoica, I. Dominant resource fairness: Fair allocation of multiple resource types. In *Proceedings of the USENIX Conference on Networked Systems Design and Implementation* (Boston, Mar.). USENIX Association, Berkeley, CA, 2011.
14. 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 *Proceedings of the ACM Conference on Architectural Support for Programming Languages and Operating Systems* (London, U.K., Mar.). ACM Press, New York, 2012, 411–422.
15. 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 *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation* (San Diego, Dec.). USENIX Association, Berkeley, CA, 2008, 309–322.
16. Gutman, A. and Nisan, N. Fair allocation without trade. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems* (Valencia, June). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2012, 719–728.
17. Heo, J., Zhu, X., Padala, P., and Wang, Z. Memory overbooking and dynamic control of Xen virtual machines in consolidated environments. In *Proceedings of the Symposium on Integrated Network Management.* IEEE Computer Society, 2009, 630–637.
18. Holdren, J.P. and Lander, E. *Realizing the Full Potential of Government-held Spectrum to Spur Economic Growth. Technical Report.* The President's Council of Advisors on Science and Technology, Washington, D.C., July 2012; http://www.whitehouse.gov/sites/default/files/microsites/ostp/pcast_spectrum_report_final_july_20_2012.pdf
19. Hu, L., Ryu, K.D., Silva, D.D., and Schwan, K. v-Bundle: Flexible group resource offerings in clouds. In *Proceedings of the International Conference on Distributed Computing Systems* (Macau, June). IEEE Computer Society, 2012, 406–415.
20. Kash, I.A.; Murty, R.; Parkes, D.C., Enabling spectrum sharing in secondary market auctions. *IEEE Transactions on Mobile Computing 13*, 3 (Mar. 2014), 556–568.
21. Kelly, F. Charging and rate control for elastic traffic. *European Transactions on Telecommunications 8*, 1 (Jan.-Feb. 1997), 33–37.
22. Lazar, A. and Semret, N. *Design, Analysis and Simulation of the Progressive Second Price Auction for Network Bandwidth Sharing.* Columbia University, New York, Apr. 1998.
23. Lubin, B., Parkes, D.C., Kephart, J., and Das, R. Expressive power-based resource allocation for data centers. In *Proceedings of the International Joint Conference on Artificial Intelligence* (Pasadena, CA, July 2009), 1451–1456.
24. Nathuji, R., Kansal, A., and Ghaffarkhah, A. Q-Clouds: Managing performance interference effects for QoS-aware clouds. In *Proceedings of the ACM SIGOPS European Conference on Computer Systems* (Paris, Apr.). ACM Press, New York, 2010, 237–250.
25. Odlyzko, A. Paris Metro pricing for the Internet. In *Proceedings of the First ACM Conference on Electronic Commerce* (New York, Nov.). ACM Press, New York, 1999, 140–147.
26. 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 *Proceedings of the ACM SIGOPS European Conference on Computer Systems* (Nuremberg, Germany, Apr.). ACM Press, New York, 2009, 13–26.
27. Parkes, D.C., Procaccia, A.D., and Shah, N. Beyond dominant resource fairness: Extensions, limitations, and indivisibilities. In *Proceedings of the ACM Conference on Electronic Commerce* (Valencia, Spain, June). ACM Press, New York, 2012, 808–825
28. Rahman, M.R., Lu, Y., and Gupta, I. *Risk-Aware Resource Allocation for Clouds. Technical Report.* University of Illinois at Urbana-Champaign, 2011; http://hdl.handle.net/2142/25754
29. 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.
30. Ried, S., Kisker, H., Matzke, P., Bartels, A., and Lisserman, M. *Sizing the Cloud—Understanding and Quantifying the Future of Cloud Computing. Technical Report.* Forrester Research, Cambridge, MA, 2011; http://www.forrester.com/Sizing+The+Cloud/fulltext/-/E-RES58161
31. 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 *Proceedings of the ACM Conference on Computer and Communications Security* (Chicago, Nov.). ACM Press, New York, 2009, 199–212.
32. 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 *Proceedings of the USENIX Workshop on Hot Topics in Operating Systems* (Santa Fe, NM, June). USENIX Association, Berkeley, CA, 2005.
33. 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*, 1 (Mar. 2011), 81–94.
34. Verma, A., Ahuja, P., and Neogi, A. Power-aware dynamic placement of HPC applications. In *Proceedings of the ACM International Conference on Supercomputing* (Island of Kos, Greece, June). ACM Press, New York, 2008, 175–184.
35. Vermeersch, K. *A Broker for Cost-Efficient QoS-aware Resource Allocation in EC2. Master's Thesis.* Universiteit Antwerpen, Antwerp, Belgium; http://www.thesis.kurtvermeersch.com/
36. Vöcking, B. A universally truthful approximation scheme for multi-unit auctions. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms* (Kyoto, Japan, Jan.). Siam, 2012, 846–855.
37. 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 18*, 2 (Feb 1992), 103–117.
38. Yi, S., Kondo, D., and Andrzejak, A. Reducing costs of spot instances via checkpointing in the Amazon Elastic Compute Cloud. In *Proceedings of the IEEE International Conference on Cloud Computing* (Miami, FL, July). IEEE, 2010, 236–243.
39. Zaman, S. and Grosu, D. Combinatorial auction-based dynamic VM provisioning and allocation in clouds. In *Proceedings of the IEEE International Conference on Cloud Computing Technology and Science* (Athens, Greece, Nov.-Dec.). IEEE, 2011, 107–114.
40. Zhou, X., Gandhi, S., Suri, S., and Zheng, H. eBay in the sky: Strategy-proof wireless spectrum auctions. In *Proceedings of the ACM International Conference on Mobile Computing and Networking* (Miami, FL). ACM Press, New York, 2–13.

Orna Agmon Ben-Yehuda (ladypine@cs.technion.ac.il) is a research associate in the Computer Science Department of the Technion - Israel Institute of Technology, Haifa, Israel.

Muli Ben-Yehuda (mulix@mulix.org) is the founder of Hypervisor Technologies and Consulting Ltd., Haifa, Israel, and chief scientist of Stratocale, Herzliya, Israel.

Assaf Schuster (assaf@cs.technion.ac.il) is head of the Technion Center for Computer Engineering and a professor in the Computer Science Department of the Technion - Israel Institute of Technology, Haifa, Israel.

Dan Tsafrir (dan@cs.technion.ac.il) is an assistant professor in the Computer Science Department of the Technion - Israel Institute of Technology, Haifa, Israel.