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

Orna Agmon Ben-Yehuda Muli Ben-Yehuda
Assaf Schuster Dan Tsafrir

Department of Computer Science
Technion — Israel Institute of Technology

Haifux 2012

What will be the New Thing After IaaS?

Recent IaaS Trends:

- The shrinking duration of rental periods
- The increasingly fine-grained resources offered for sale
- Meaningful resource pricing
- Tiered service levels agreements (SLAs)

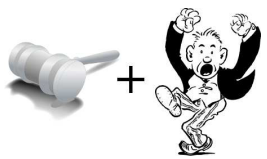
These trends and the economy will drive IaaS to turning into RaaS.

Trend: Granularity of Duration of Rent

- 3 years on average: buying hardware
- Months: web hosting
- Hours: EC2 on-demand (pay-as-you-go)
- 5 minutes: CloudSigma, EC2 Spot Instances (pay-as-you-go)
- 3 minutes: GridSpot
- 1 minute: Profitbricks

Extrapolation: Granularity of Duration of Rent

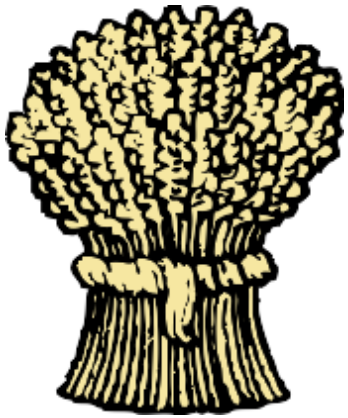
- Clients want to pay for resources only when they need them.
- Clients need extra resources to be allocated within seconds (e.g., when slashdotted)
- Phone charges are advancing from minutes to single seconds.
- Phone companies were driven by consumer pressure and court orders.



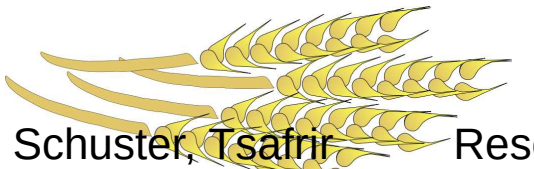
- Car rental (by days) is giving way to car sharing (by the hour).
- We extrapolate that cloud resources will be rented by the second.

Trend: Resource Granularity

- Most cloud providers sell fixed bundles, called “instance types” or “server sizes”.



- Amazon allows adding and removing of “network instances” and “block instances”, thus dynamically changing I/O resources.
- Since August 2012, Amazon also allow clients to set a desired rate on a per-block-instance basis.
- CloudSigma, Gridspot, and ProfitBricks offer clients to compose a flexible bundle.

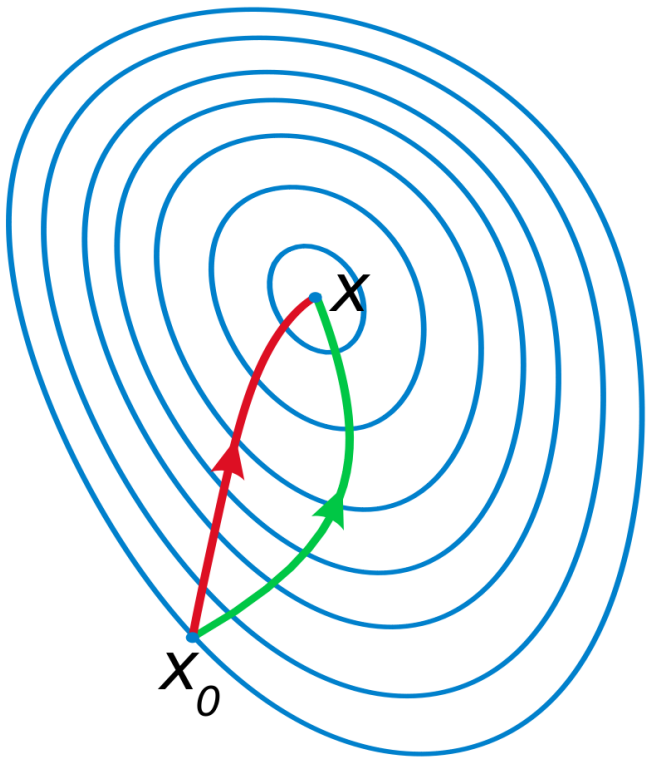


Extrapolation: Resource Granularity

- As physical servers increase, an entire server may be too much for a single client.
- Renting a fixed bundle may waste client resources, even if its requirements stay the same over time. For example, if the client can only use 7 cores, why should it rent 8?
- We extrapolate that clients will rent a seed bundle, and dynamically supplement it with resources in fine granularity.

A job half done

If only the first two trends culminate as described, then clients *can* finally optimize their resource use.
However, this is not enough to guarantee a green, efficient cloud. Would they really optimize? Will they optimize the right target function for a green cloud?



Trend: Service Level Agreements

- Most cloud providers account for rigid availability only (“the machine is accessible”).

- GoGrid and CloudSigma provide guarantees in terms of minimal actual delivered capacity (latency, packet loss and jitter).

Meaningful Resource Pricing

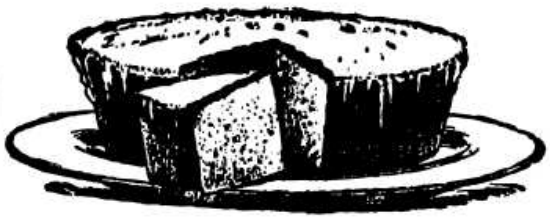
- Benchmarks show great variance in the performance of supposedly similar cloud instances.
- Different clients need different guarantees: a bank will pay for 100% availability. A small business may settle for a 95% guarantee.
- Client valuations of performance and resources differ and are private information.
- Some researchers (Padala'09, Heo'09, Nathuji'10) argue for selling client performance and measuring it. This concept is impossible for a real commercial IaaS black box client.

- IaaS Providers cannot sell performance. They must keep selling resources.

Extrapolation: Service Level Agreements

We extrapolate that:

- Client pressure for efficiency will drive providers to supply levels of quality service.
- Low-QoS clients will be willing to pay less than high-QoS clients.



Tiered Service Levels

How can service levels be tiered?

- Absolute: Unavailability of a minimal X, which is at least a fraction Y of a service period Z. Headroom is still required.
- Relative, like EC2 Spot instances and DotCloud. No headroom is required.

Economic Forces Acting on the Provider

- Commoditization: e.g., OpenStack, adopted by Rackspace, RedHat and even VMWare.
- Economic mechanisms will be required inside a machine.
- The provider must keep spare resources for high-QoS clients.
- The provider can let low-QoS clients use the spare resources, subject to availability.
- The provider must mix low QoS clients with high QoS clients.

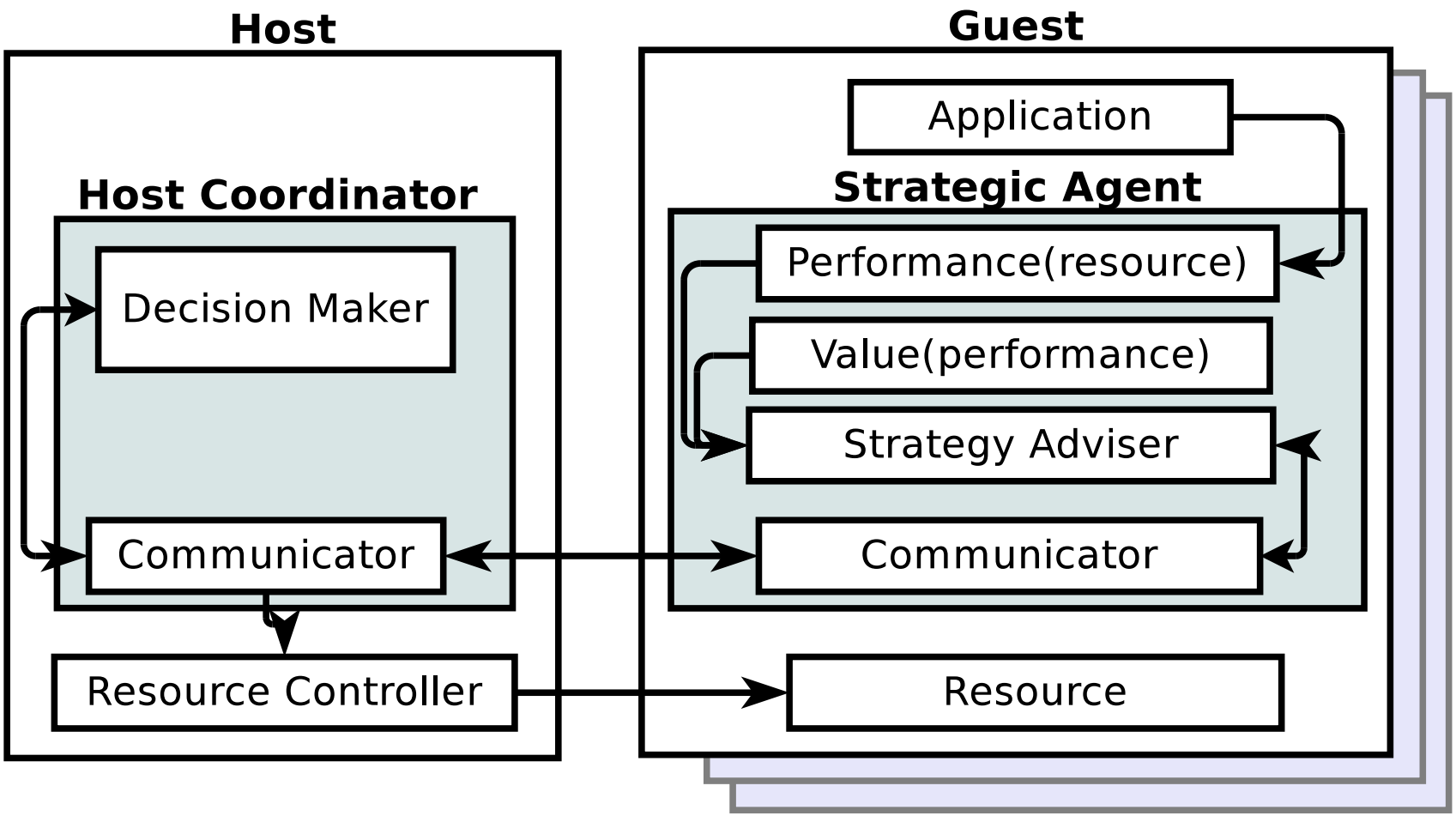
Economic Forces Acting on the Client

- Clients aim to buy exactly what they need, to save on expenses.
- And since providers aim to sell clients what they want to buy, to gain and retain clients...
- CPU is rented by cycles, memory is rented by the page, I/O is rented by bandwidth.

Economic Forces Leading to the RaaS Cloud: Result

- Both clients and providers must continuously decide what and when to rent.
- The fine rent-time granularity and bundle flexibility make decision-making a core function.
- Both providers and clients will use economic software to handle decision making and economic interaction.

The RaaS Cloud

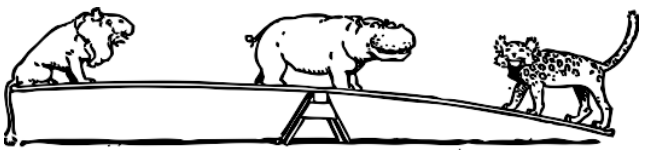


The Guest Agent

- Decides on the size of the seed machine.
- Changes the desired amount of resources on a second-by-second basis.
- Negotiates and bids.
- Trades in the futures market.
- Sublets resources or complete nested virtual machines.
- Is not mandatory: dumb clients are still supported, with the same inefficiency of today's IaaS clouds.

The Host Coordinator: Market Driven Resource Allocation

- Has a view of the global picture (total system resources, change predictions)
- Dictates economic mechanisms and protocols.
- Allocates resources according to agreements.
- Uses the resources to verify that high-QoS clients are satisfied, possibly at the expense of low-QoS clients on the same machine, and given the specific current needs of each client.



Priorities and Host Coordination

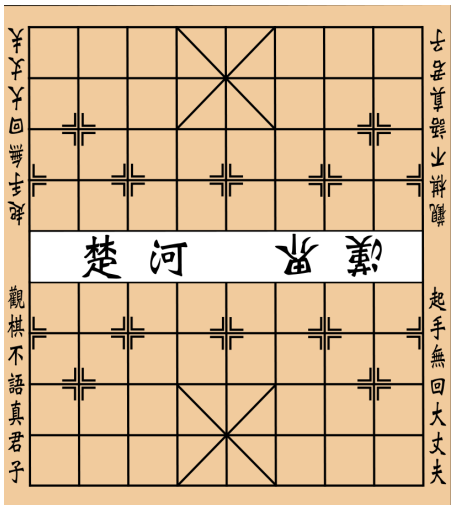
- Priorities for headroom only
- Vertical elasticity: like Robin Hood, in reverse
- A few good neighbors
- Far from the madding crowd

Implications, Challenges, Opportunities

- A client software stack (applications, libraries, OS) that utilizes resources for short durations and trades them off.

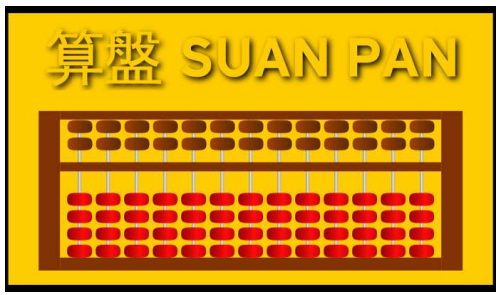
Implications, Challenges, Opportunities

- Economic (game theoretic) mechanisms for multi-resource allocation with different QoS levels.
  - Realistic
  - Incentive compatible
  - Collusion-resistant
  - Computationally efficient at large scale
  - Optimizes the provider's revenue or a social welfare function
  - Minimizes the price of anarchy



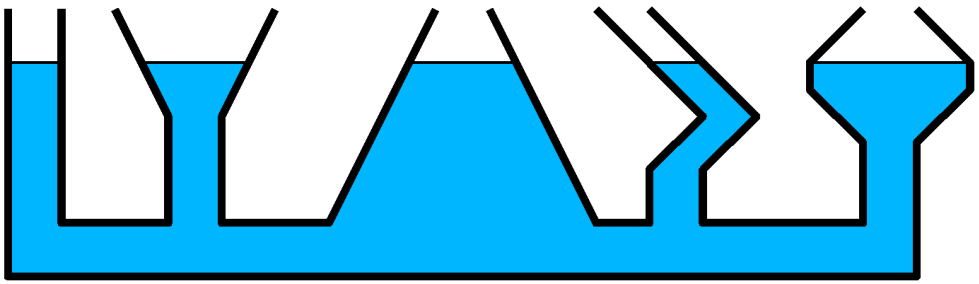
Implications, Challenges, Opportunities

- Technical mechanisms for handling resource (re)allocation, metering and charging:
  - efficient,
  - reliable,
  - and resistant to side channel attacks.



Implications, Challenges, Opportunities

- Balancing guests across a data-center to create heterogeneous mixes of QoS levels on each machine.



Memcached: an application for example

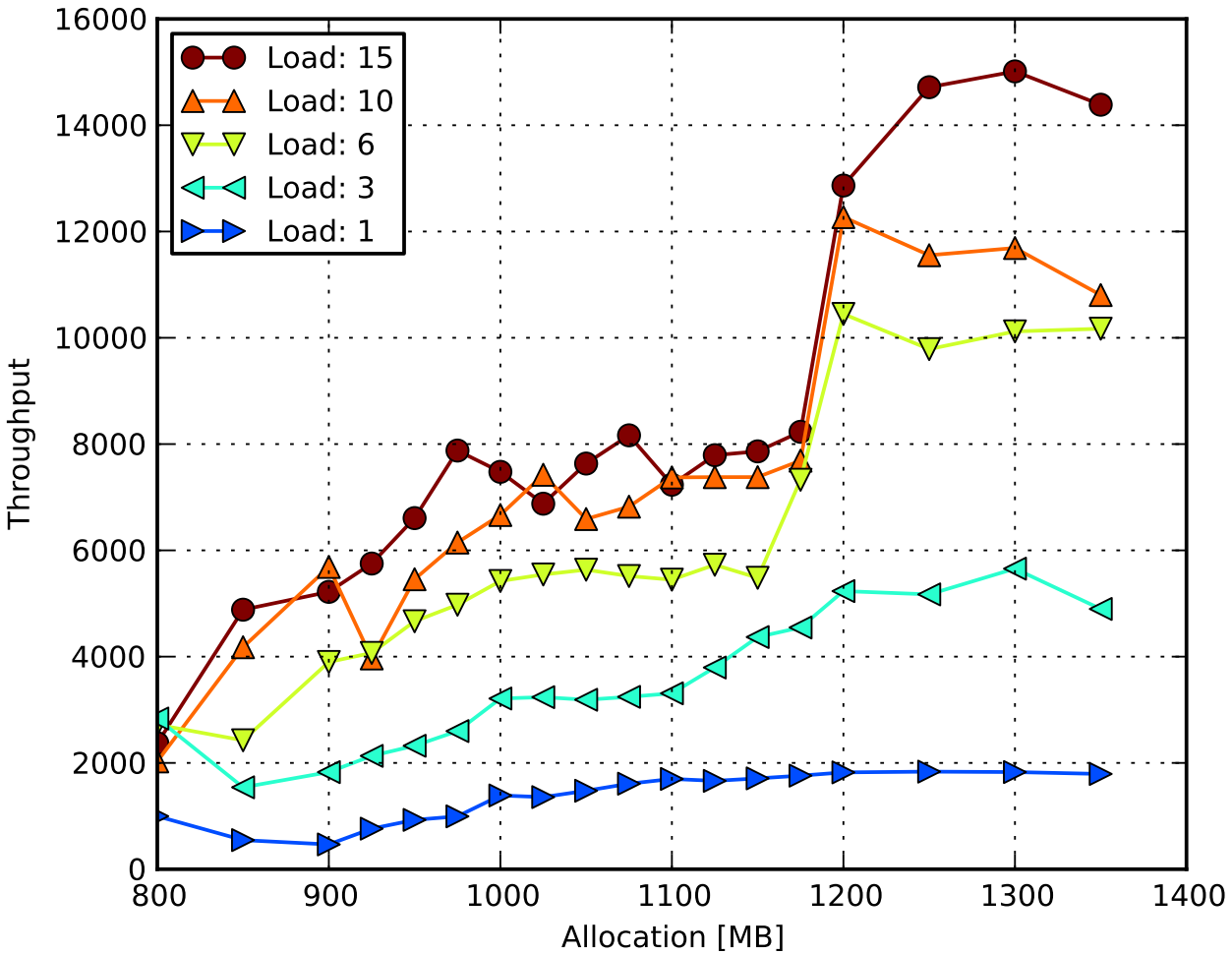


Figure by Eyal Posener.

Dynamic Memcached: an application for example

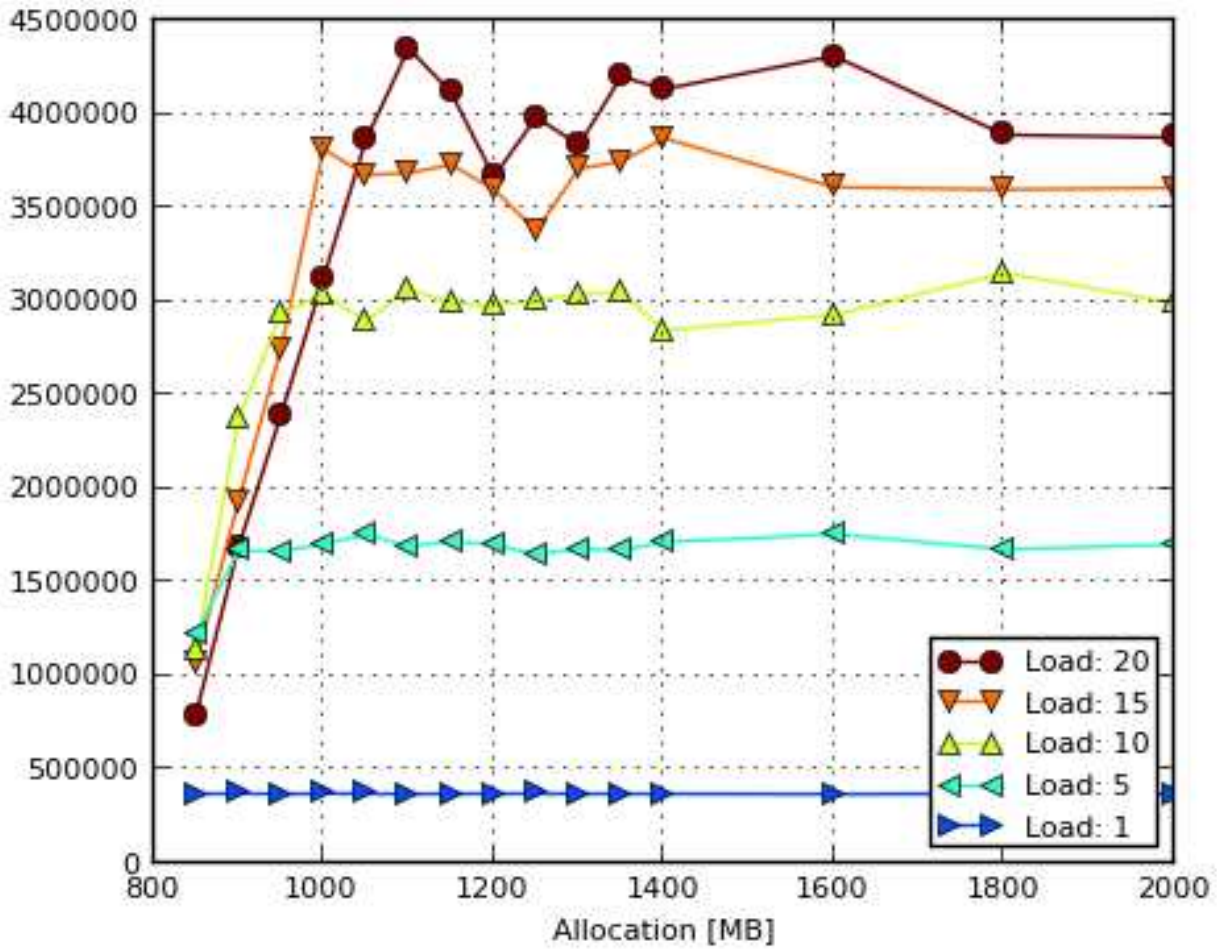


Figure by Eyal Posener.

Questions?

Contact us at:
{ladypine, muli, assaf, dan } at cs.technion.ac.il
Thank You!