Focus period and Workshop in Cloud Control
Lund University, Sweden, May 6-10, 2014

Capacity Management in IaaS Cloud

David Breitgand, IBM Haifa Research Lab

Based on joint works with many collaborators

Speaker Background

- Distributed Computing, Hebrew University, Israel
- Networking, Technion, Israel
- Applied Mathematics, Polytechnic University, Novosibirsk, Russia

- Technical Lead of Cloud Operating System Technology Group @IBM Research -- Haifa
- Technical Lead of Software Defined Manufacturing Group @IBM Research -- Haifa
- 10 years with IBM research, working on algorithms, performance analysis, load balancing, root cause analysis, data center optimization, SLA management, SAN performance management, Cloud computing, VM migration optimization, capacity management.

- **Overarching research theme:** studying, modeling, and optimizing tradeoffs between the costs of management and benefits accrued

Outline

- Introduction
- Problems
- Results
- Future: where does this become really hard and should we go there?
- Q&A

Cloud success == three things:

Business Model

Agility

Simplicity

Do I need [long term] Capacity Management in an IaaS Cloud?

Capacity [long term] Management on a IaaS Cloud?

If you are a **customer** of a public cloud

Capacity [long term] Management on a IaaS Cloud?

If you are a **provider** of a public cloud

Why? And what [long term] “capacity management” means?

- IaaS commoditize → IaaS providers operate under perfect competition
- Provider needs to be competitive → Pressure to lower prices
- Pressure to lower prices → Pressure “to do more with less”

- Capacity management is required to achieve that

- The most obvious link to this workshop: cost efficient elasticity

How to do more with less?

- By increasing *over-commit ratio* to improve *cost efficiency*

- But if I over-commit too much, I will *degrade* cost-efficiency, won't I?

What does “cost-efficiency” means?

- Cost efficiency [first usage in 1970]
  - “Cost effective describes something that is of a good value, where the benefits and usage are worth at least what is paid for them”
- Cost efficiency is a relative concept describing relationship between ***at least*** two options
- Example 1:
  - Use of PSTN line versus VoIP for long distance call
- Example 2:
  - Static resource provisioning versus on-demand provisioning from shared multiplexed pool

The Gist of “capacity management” for provider

- Find minimal physical resources configuration to provide service to the customer, where the value of using the service worthy at least of what the customer pays for it
- Service in IaaS:
  - Provide VM, storage, networks (i.e., resource collections)
- Trade-off:
  - Risk of resource congestion vs. cost of using successfully acquired resources

There is a problem, though

- How do you know the value a customer assigns to using the service???

If you expected an answer…

Second Price Auctions?

- Vickrey-Clark-Groves
- Heterogeneous goods
- Dominant strategy is to report the true value for the goods

- Beautiful game-theoretic mechanism
- Rarely used in practice
- Complex implementation
- Close to zero seller revenue

Zaman, Sharrukh, "Combinatorial Auction-Based Virtual Machine Provisioning And Allocation In Clouds" (2013). *Wayne State University Dissertations*. Paper 720.

Recent Ph.D. thesis in designing Combinatorial Auctions for VM allocation

So, we have the input problem

- A provider optimizing capacity has a problem with the quality of the input
- Capacity is planned in practice solely based on the observed historical demand, error rate, and a forecast
- Forecast quality quickly deteriorates with:
  - Prediction horizon going further into the future
  - Historic data going back into the past
- Performance of forecasting often depends on fine tuning

SLAs

- A mechanism used absence of the real input on consumer valuation of the service
- Essentially a guess
- Essentially a declaration of intentions
- Perceived as due diligence by customers seeking to avoid risk
- Essentially a means to differentiate and compete under perfect competition conditions

Do we have SLAs today in an IaaS Cloud?

rackspace HOSTING :

“**Finally, Real SLAs for Cloud Computing**
The SLA adopted for Cloud Servers™ is just as [illegible]ssive as the one Rackspace provides for traditional host[illegible]ervers. It provides remedies for any downtime event caused [illegible] the network, data center infrastructure, the physical host server, or the migration of

607 words

amazon web services™ :

AWS will use commercially reasonable efforts to make [illegible]mazon EC2 available with an Annual Uptime Percentage[illegible]d below) of at least **99.95%** during the Service Y[illegible] the event Amazon EC2 does not meet the Annual [illegible]e Percentage commitment, you will be eligible to rece[illegible] a Service Credit as described below.

1500 words

GOGRID

10,000% Guaranteed, **100% Uptime** Service Level Agreement (“[illegible]”) supplements the Terms of Service and together such documents, and othe[illegible]nced in the Terms of Service, form a binding agreement (the "**Agreement**") betw[illegible]**oGrid** and Customer”.

….

“A "10,000% Service Credit" is a credit equivalent to **one hundred (100) times Customer's fees** for the impacted Service feature for the duration of the Failure. (For example, where applicable: a Failure lasting seven (7) hours would result in credit of seven hundred (700) hours of free service for the feature in question”

3700 words

What do we **really** get? (assorted examples)

"We guaranty that **our data center network** will be available **100%** of the time in any given monthly billing period, excluding scheduled maintenance.

We guaranty that **data center HVAC and power** will be functioning **100%** of the time in any given monthly billing period, excluding scheduled maintenance. Infrastructure downtime exists when Cloud Servers downtime occurs as a result of power or heat problems.

"Annu
the pe
Amaz
been
is still
the ser

We guaranty the functioni
hypervisor. If a cloud
complete **within**

"The **minimum period of Failure eligible for a credit is 15 minutes**, and **shorter periods will not be aggregated**. The **maximum credit for any single Failure is one month's Service fees**. In the event that multiple periods of Failure overlap in time, credits will not be aggregated, and Customer **will receive credit only for the longest such period of Failure**".

Current Cloud SLA Practices Summary

- Inflated promises
- Standard SLA with “one size fits all” availability SLO
- Obfuscation of provider commitments:
  - Being non-specific about maximum maintenance time per billing period
  - Being non-specific about maintenance head-up warning
  - Unavailability periods aggregation trickery
  - Requiring customers to understand failures on the vendor side
  - Being non-specific about availability tests to verify SLA compliance
  - Placing the onus of damage proof on the customer
- Refund policies that do not compensate the loss of value to the business due to unavailability

Impact of downtime on business

- Loss of profits
- Impact on stock price
- Loss of cash flow from debtors
- Loss of customers (lifetime value of each)
- Market share loss of product
- Cost of fixing / replacing equipment
- Cost of fixing / replacing software
- Salaries paid to staff unable to undertake productive work
- Salaries paid to staff to recover work backlog and maintain deadlines
- Cost of re-creation and recovery of lost data
- Interest value on deferred billings
- Additional cost of credit through reduced credit rating
- Fines and penalties for non-compliance
- Liability claims
- Additional cost of advertising, PR and marketing to reassure customers and prospects to retain market share
- Additional cost of working; administrative costs; travel, etc.
- Cloud: “we will give you a free service next month”

To remind us about our problem

- Find minimal physical resources configuration to provide service to the customer, where the value of using the service worthy at least of what the customer pays for it

- SLAs are an approximation of the customer valuation (being arbitrarily far from the true consumer valuation)

- **Takeaway:** SLA construction is intimately related to capacity management
- Short term (e.g., feedback loop control) resource management is done w.r.t. SLAs that might be suboptimal

- **In this talk: we don't discuss deriving optimal SLAs**
- The focus will be on an easier problem: make SLAs **more specific** and optimize capacity w.r.t. them

…and there is also another input problem

- Find minimal physical resources configuration to provide service to the customer, where the value of using the service worthy at least of what the customer pays for it

- Environment is very dynamic
- There are long running services and short running services, tenants come and go, hardware changes, failures happen, disasters happen…
- The environment changes as capacity optimization cycle completes…

…and, yeah, there is one more “small” input problem

- Administrators don’t like your tool monitoring their production environment:
  - Too much storage overhead
  - Too much network overhead
  - Too much disturbance to services normal operation

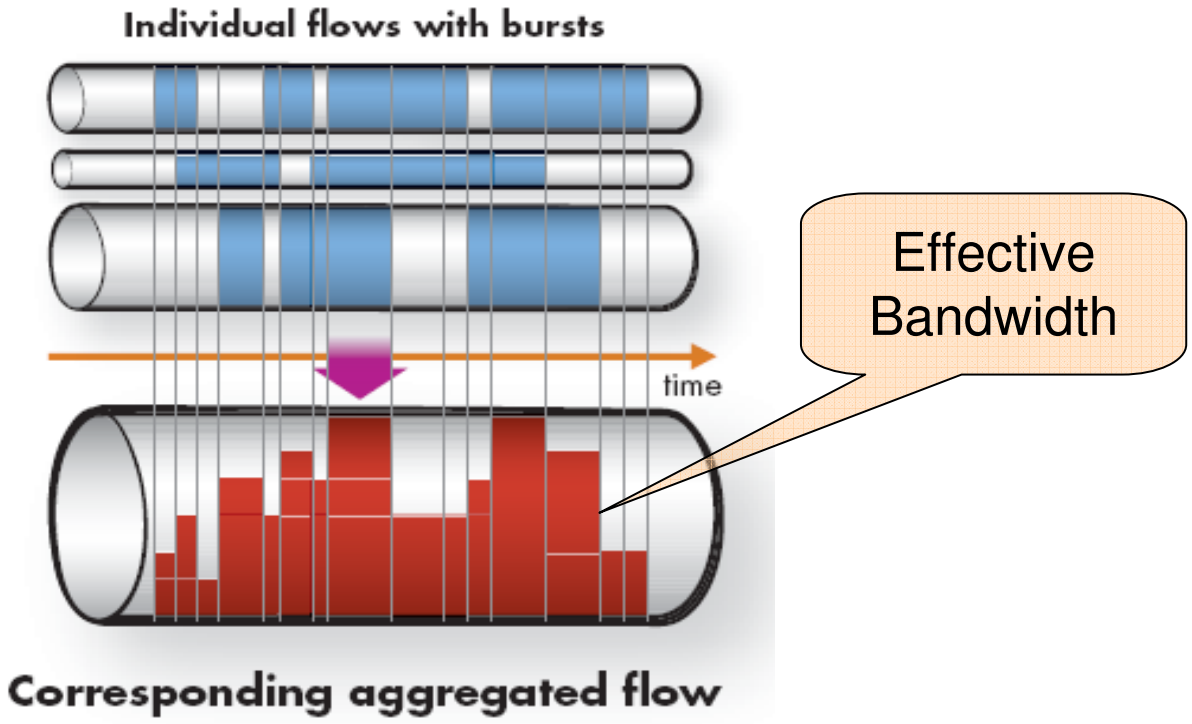
So….

- No input on true customer valuations
- No stable VM populations
- More often than not no meaningful SLAs
- Scarce resources for capacity management

Before moving forward with theory: **a disclaimer**

- **A distance between theory and practice is shorter in theory than in practice**
- Human administrator is the bottleneck for anything “smart”
- **In practice:** capacity management helps **slowing down** procurement cycle, but nobody [that I know] is looking for squeezing the last drop of optimality and/or accuracy

Cloud is cost-efficient thanks to Statistical Multiplexing

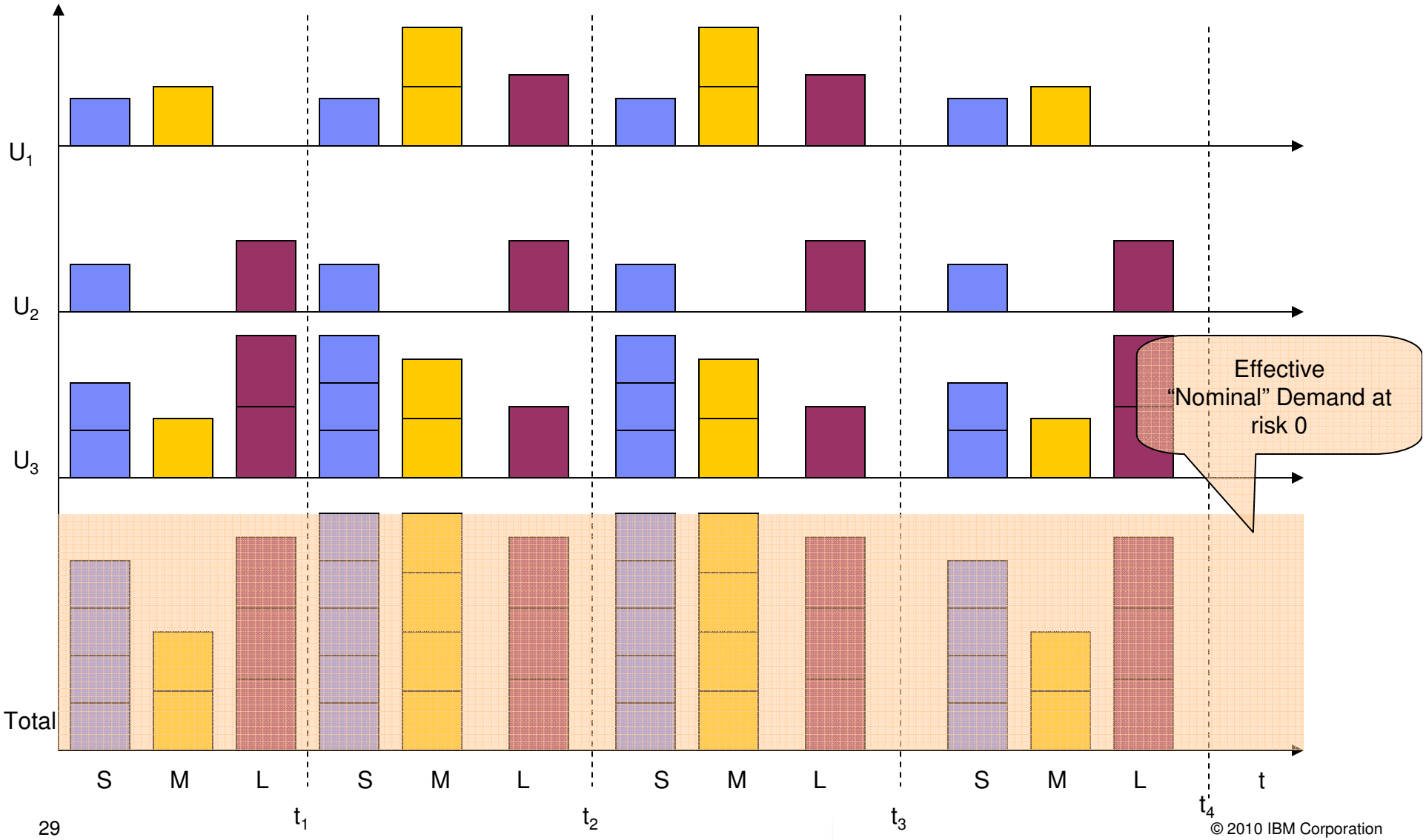


Over-commit: the total capacity of the shared resource is allowed to be *much* smaller than the maximum total demand

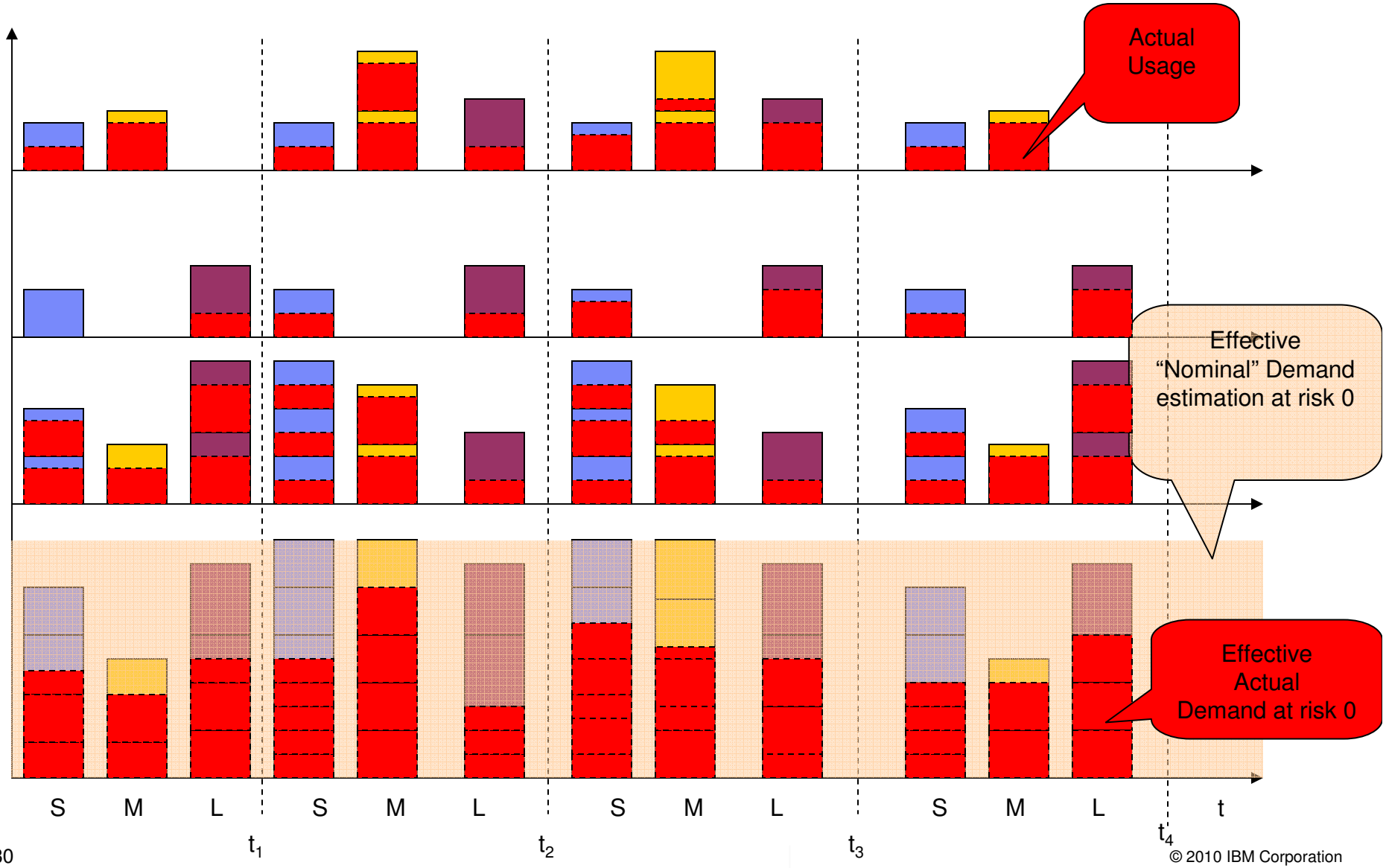
Resource Demand

- Can be expressed
  - In terms of “nominal allocations”
  - In terms of “actual utilization”

Over-Commit in IaaS:
"Pool Level": birth/death of VMs



Over-commit in IaaS: “Host Level”: actual utilization per VM



Highlights

Over-commit on Nominal Demand

- David Breitgand, Zvi Dubizky, Alex Glikson, Amir Epstein, Inbar Shapira, “SLA-aware Resource Over-Commit in an IaaS Cloud”, CNSM’12, Las Vegas, USA

Over-Commit on Actual Demand

- David Breitgand and Amir Epstein, “Improving Consolidation of Virtual Machines with Risk-aware Bandwidth Oversubscription in Compute Clouds”, INFOCOM’12, March 25-30.

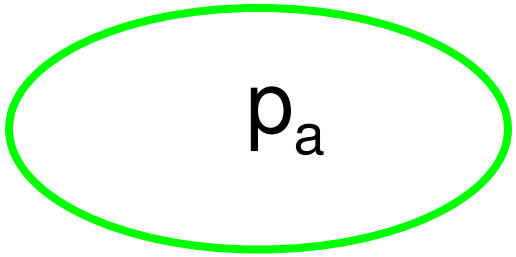
Challenges in Respecting Nominal Allocations cost-efficiently

- No statistics about actual usage of a VM are available
- High dynamics with VMs arriving and departing: no constant VM population
- A resource should be provided as an indivisible “whole” (e.g., bare metal cloud server)

Standard SLA clause

- Standard availability clause in IaaS SLA:

Percentile of the billing period when **pinging** VM succeeds



IBM

Extended Availability SLA

- Assume virtual hardware discrete types a_1, a_2, …, a_n
- For simplicity, assume a single elasticity range for all services: R = [R_{min}, R_{max}]
- Each service S_i is guaranteed to successfully assume configuration Gi = < a_1, a_2, …, a_n > subject to <elasticity range>, with probability <p> computed over <billing period>
- $p \leq p_a$, where p_a is the standard availability clause probability
- Rationale: guarantee on assuming desired configuration
- Interpretation: guarantee on probability to launch a VM of any type, subject to elasticity range

SLA-aware cloud over-commit (CNSM'12)

- Let ***n*** be the total number of workloads (services)
- Let ***i*** be number of VM discrete types
- Let $\boldsymbol{Y_{i,j}}$ be the random variable representing number of VM instances of type ***i*** used by workload ***j***
- $\boldsymbol{X_i} = \sum_{j=1}^{n} Y_{j,i}$ is the total number of VM instances of type *i* in the cloud
- Definition 1: **Nominal Demand** is $\overline{X} = (X_1, X_2, \ldots, X_l)$
- Definition 2: **Effective Nominal Demand**
  - Minimal vector $\overline{\boldsymbol{D}}$, such that

$$Pr(\bigwedge_i (X_i \leq D_i)) \geq p$$

- Union bound: $Pr(\vee_i (X_i > D_i)) \leq \sum_{i=1}^{l} Pr(X_i > D_i)$

p_i

Critical observation

- We do not know distributions of $Y_{i,j}$
- Small contributions to X_i
- For large clouds effect of dependencies diminishes
- Can be treated as independent
- Central Limit Theorem: X_i asymptotically converges to normal distribution

- Should also be identically distributed, but we ignore that in this work

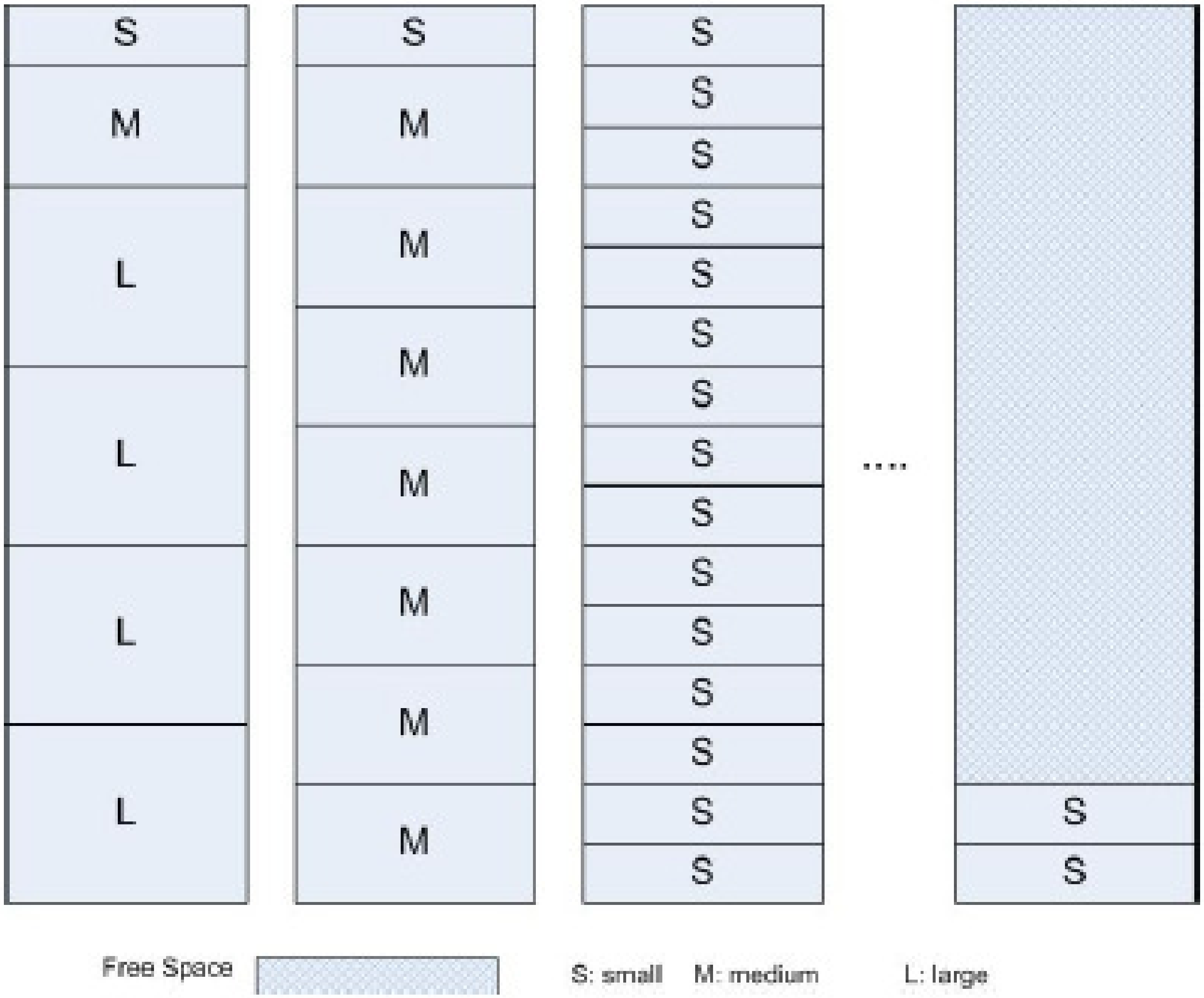
Not so fast ☺

- The variables are discrete
- Normal distribution is truncated normal

$$D_i = \lceil \mu_i' + Z_{p_i} \sigma_i' \rceil$$

$$\mu_i' = \mu_i + \sigma_i \frac{\phi(-\frac{\mu_i}{\sigma_i})}{1 - \Phi(-\frac{\mu_i}{\sigma_i})} \qquad \sigma_i' = \sigma_i \sqrt{1 - \frac{\phi(-\frac{\mu_i}{\sigma_i})}{1 - \Phi(-\frac{\mu_i}{\sigma_i})} \left(\frac{\phi(-\frac{\mu_i}{\sigma_i})}{1 - \Phi(-\frac{\mu_i}{\sigma_i})} + \frac{\mu_i}{\sigma_i} \right)}$$

Illustration: Sample VM slots reservation



Main Virtues

- Computationally and storage efficient
- Extremely simple
- Simple placement
- Explicit calculation of capacity based on risk perceived as tolerable → easy to transform into a policy
- Fast: easy to do “what-if?” analysis to extract **providers’** true valuations of resource congestion

- **A framework is more important than a specific algorithm**

Things missed by nominal strategy

- Nominal capacity allocation strategy treats VMs/resources as indivisible “wholes”
- When should we expect it to produce best results?
- **Gives best results when most of the time actual resource utilization of most of VMs on all of resource types is close to nominal discrete configuration of the VMs**

- **If this is not the case, nominal strategy protects quality of experience (performance), but might be wasteful on resources**

Actual Utilization Over-Commit Strategy to the rescue!

- David Breitgand and Amir Epstein, “Improving Consolidation of Virtual Machines with Risk-aware Bandwidth Oversubscription in Compute Clouds”, INFOCOM’12, March 25-30.

Stochastic Bin Packing Problem (SBP)

- $S=\{X_1,\ldots, X_n\}$ – Set of items
- X_i – random variable representing the size (bandwidth demand) of item i
- p – overflow probability
- Goal: Partition the set S into the smallest number of subsets (bins) $S_1,\ldots,S_k$ such that

$$\Pr[\sum_{i:X_i \in S_j} X_i > 1] \leq p \quad \text{for} \quad 1 \leq j \leq k$$

p represents an SLA-stipulated value

Related Work – Bin Packing

- The problem is NP-hard
- Bin packing is hard to approximate to a factor better than 3/2 unless P=NP.
- First Fit Decreasing (FFD) has asymptotic approximation ratio of 11/9 and (absolute) approximation ratio of 3/2.
- MFFD algorithm has asymptotic approximation ratio of 71/60.
- AFPTAS exists.
- Online bin packing
  - First Fit (FF) has competitive ratio of 17/10.
  - Best upper and lower bounds are 1.58899 and 1.54014, respectively.

Related Work – Stochastic Bin Packing

- $O\left(\sqrt{\frac{\log p^{-1}}{\log\log p^{-1}}}\right)$ -approximation for SBP with Bernoulli variables [Kleinberg et. al 1997]
- SBP with Poisson, Exponential and Bernoulli variables [Goel and Indik 1999]
  - PTAS exists for Poisson and exponential distributions.
  - Quasi-PTAS exists for Bernoulli variables.
  - These results relax bin capacity and overflow probability constraints by a factor 1+ε.
- $(1+\sqrt{2})(1+\varepsilon)$ - competitive algorithm for SBP with normal variables [Wang et. al 2011]

Our Results (Breitgand and Epstein INFOCOM'12)

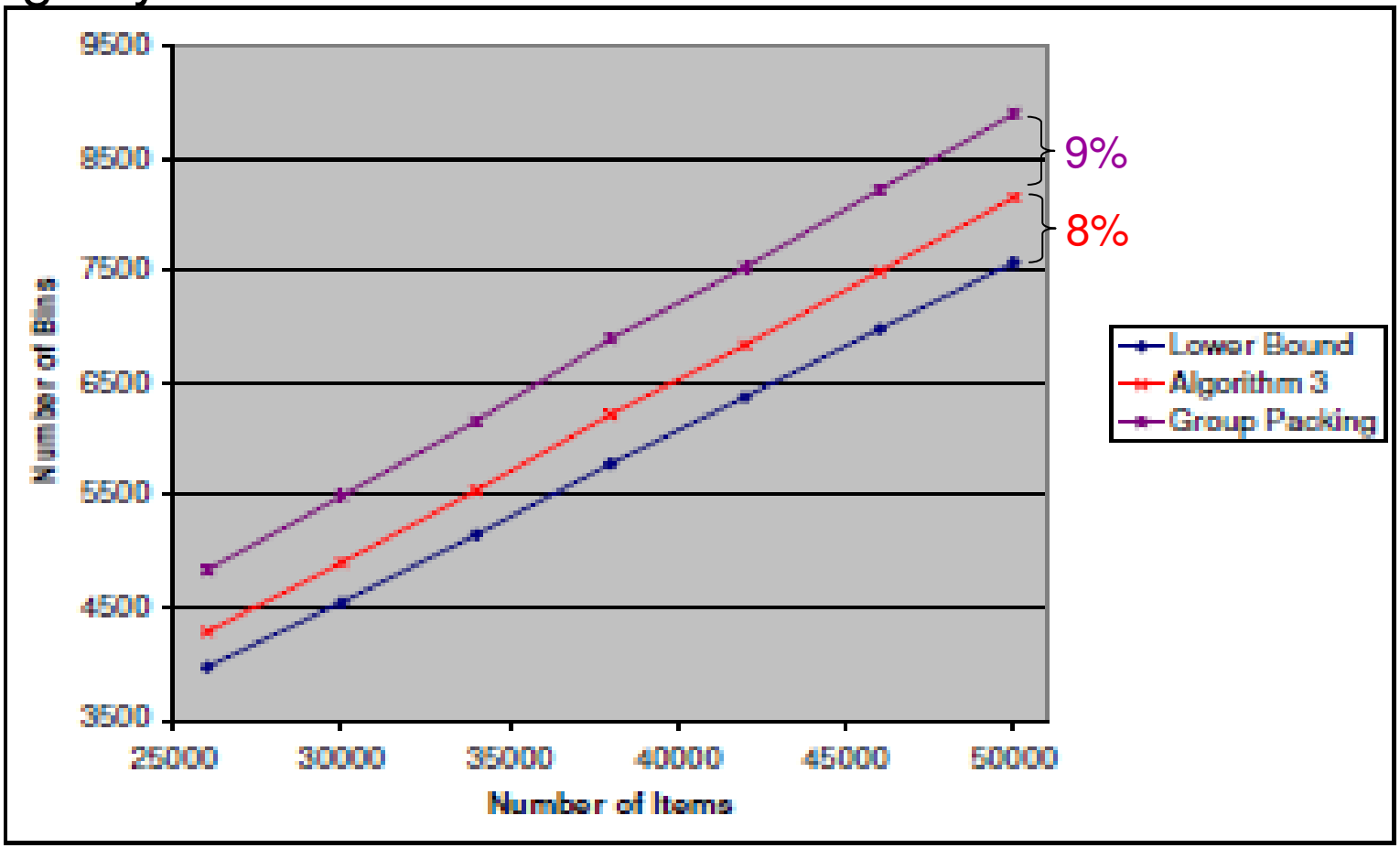
- 2-approximation algorithm for SBP with normal variables
- (2+ε)-competitive algorithm for online SBP with normal variables
  - Best known

Intuition

- Collocating “bursty” items (VMs) together **reduces** effective size
- Normality assumption: relatively small number of VMs per host
- For large hosts (e.g., large IBM Power machines), normality assumption can be dropped

Online Algorithms

- Large synthetic instances based on scaled real workloads



Real Instance

p	Algorithm 3 (Online)	Algorithm 1 (Approx.)	Group Packing	FFD	FF	Algorithm 2 (L.B)
0.1	164	146	595	332	334	144
0.01	215	195	785	519	522	192
0.001	263	243	881	656	662	237

Summary of the methodology

- Make SLA meaningful – starting point
- Calculate effective capacity with respect to observed demand and target resource congestion probability
- Calculate placement for effective capacity
- Continuously update effective capacity
- Recalculate placement only when a significant (affecting target congestion probability) is spotted

Conclusions & Future Research: where does this become really hard?

- Placement Constraints: cannot use simple BP anymore
  - David Breitgand and Amir Epstein, "SLA-aware placement of multi-virtual machine elastic services in compute clouds", IFIP/IEEE Integrated Network Management (IM'11), pp. 161-168, Dublin, Ireland
  - Extends: B. Urgaonkar, A. L. Rosenberg, and P. J. Shenoy, "Application placement on a cluster of servers," *Int. J. Found. Comput. Sci.*, vol. 18, no. 5, pp. 1023–1041, 2007.
  - Elastic Services Placement Problem (ESPP) generalizes GAP, but does not admit constant factor approximation $m^{1/2-\epsilon}$ for any constant $\epsilon > 0$.
- Performance degradation due to non-virtualized resources
  - L2 cache? Bus? Not visible metrics
- **Personal appreciation 1**: these two problems are the core ones impeding progress on **systematically** improving cost-efficiency in IaaS
- **Personal appreciation 2**: solution is likely in apps collaborating with infrastructure provider