Pay-as-you-go model

- easy to acquire and release resources
- time-varying resource demands
- when to get resources?
- how many to get?
- need for proper resource scaling mechanism

(a) Peak load allocation

(b) Average load allocation

(c) Ideal allocation
Auto-Scaling support from cloud providers

- Amazon\(^1\), RightScale\(^2\), Windows Azure\(^3\)
  - define metric, threshold, control period
  - define action (amount of resources to add/remove)
  - add \(n\) VMs if \(CPU\) load > 60%
- the burden of scaling still lies with user
- the user has to be an application expert
- proper testing and configuration

---

\(^1\) Amazon auto scaling service, [http://aws.amazon.com/autoscaling/](http://aws.amazon.com/autoscaling/)

\(^2\) Rightscale. Set up Autoscaling using Alert Escalations, [http://support.rightscale.com/](http://support.rightscale.com/)

A new model of buying and selling cloud computing

- fixed bundles (predefined VM templates, widely used model)
  - small (1vCPU, 1.7GB RAM)
  - medium (1vCPU, 3.75GB RAM)
  - large (2vCPU, 7.5GB RAM)
  - what if I need 2 vCPU, 5 GB?
  - 1 hour billing cycle

- flexible resource bundles
  - user can define VM size
    - CloudSigma
  - smaller billing cycles
    - CloudSigma (5min)
    - Google cloud (1min)

- more space for optimization

---

4 CloudSigma, IaaS provider, http://www.cloudsigma.com/
Outline

1. Challenges
2. Resource scaling aspects
3. Modeling techniques
4. Contribution
5. Future work
Challenges

- lack of knowledge about the application in the cloud
  - user does not have expertise knowledge
  - cloud provider cannot access the application
- time varying workload
  - demand for individual resource also changes
- mapping performance requirements (SLA) to available resources
  - how much CPU assign to WS VM to provide 100 ms response time?
- application updates
  - model change detection
  - scaling policy adaptation
- cluster-wide correlation
  - CPU saturation of DB leads higher memory usage of WS
  - resource allocation across all tiers
Cloud applications

- **interactive applications**
  - web applications serving HTTP clients
  - layered architecture, consist of tiers (WS, AS, DB)
  - demand may disproportionately impact a specific layer
  - sensitive to under-provisioning
  - metric: response time

- **batch workloads**
  - map reduce like apps
  - long running jobs
  - resource intensive
  - under-provisioning is not vital
  - metric: throughput
Scaling horizons

- **horizontal**
  - add/remove VM(server)
  - start VM < 1 minute\(^7\)
  - not everything may be scaled horizontally (PostgreSQL, Hadoop DataNode)
  - license fee

- **vertical**
  - resize VM
  - CPU (Xen cap, CPU hotplug; KVM cgroups)
  - RAM (Xen, KVM memory ballooning)
  - disk, network bandwidth control (cgroups)
  - plug CPU < 1 second\(^7\)
  - RAM becomes cheaper\(^8\)
  - limited by the host capacity

---

\(^7\) Dutta et al., “SmartScale: Automatic Application Scaling in Enterprise Clouds”, CLOUD ’12

\(^8\) Rowstron et al., “Nobody ever got fired for using Hadoop on a cluster”, HotCDP ’12
Managing capacity overload

- admission control
  - redirect requests (LB)
  - horizontal scaling

- prioritization
  - VM migration\(^9\)
  - performance degradation\(^{10,11}\)

- resources stealing \(^{12}\)
  - some servers run 24 h
    - MemCached, DB
  - use underutilized resource

---

\(^9\) Shen et al., “CloudScale: elastic resource scaling for multi-tenant cloud systems”, SOCC ’11

\(^{10}\) Nathuji, Kansal, and Ghaffarkhah, “Q-clouds: Managing Performance Interference Effects for QoS-aware Clouds”, EuroSys ’10

\(^{11}\) Yazdanov and Fetzer, “Vertical Scaling for Prioritized VMs Provisioning”, CGC ’12

\(^{12}\) Gandhi et al., “SOFTScale: stealing opportunistically for transient scaling”, Middleware ’12
When to provision

- predictive provisioning
  - workload patterns (daily, weekly, seasonal)
- reactive provisioning
  - flash crowds (slashdotting)
  - unplanned events (earthquake, flooding)
  - handling prediction errors
  - rule based approaches Amazon, RightScale
- combined
  - reactive + predictive
System modeling

Application

workload

resource entitlement

desired performance

resource utilization

QoS (response time, throughput)

Time

Demand
System modeling

- **offline**
  - high model accuracy
  - time consuming
  - changes on the controlled system require redesign of scaling policy

- **online**
  - live system
    - low resource overhead
    - careful system identification
    - long learning time
  - sand-boxing \(^{13,14}\)
    - does not affect production system
    - quick model adaptation
    - resource overhead (we need to host the 'sand-box')
    - infrastructure management complexity

---

\(^{13}\) Zheng et al., “JustRunIt: experiment-based management of virtualized data centers”, USENIX’09

\(^{14}\) Vasić et al., “DejaVu: accelerating resource allocation in virtualized environments”, ASPLOS XVII
Modeling techniques

- Control theory
- Queuing theory
- Time series
- Reinforcement learning
Control theory

- obtain "first principles"
  - vary inputs (resource capacity, workload)
  - observe output (app performance)
- controlled experiments
- high accuracy within control bounds
- adaptive control (online)
- offline model design
- requires controlled environment
- assumes linear performance model
Control theory (Related work)

- **Adaptive resource control**\(^{15}\)  
  - CPU allocation for Web application  
  - regulate CPU utilization to 80% to achieve desired RT

- **WS provisioning**\(^{16}\)  
  - CPU and memory scaling  
  - SISO model per each resource  
  - regulate resource utilization

- **AutoControl**\(^{17}\)  
  - Multi-tier application provisioning  
  - CPU and disk I/O scaling  
  - MIMO controller  
  - ARMA model

---

\(^{15}\) Padala et al., “Adaptive control of virtualized resources in utility computing environments”, EuroSys ’07

\(^{16}\) Heo et al., “Memory overbooking and dynamic control of Xen virtual machines in consolidated environments”, IM’09

\(^{17}\) Padala et al., “Automated control of multiple virtualized resources”, EuroSys ’09
Queuing theory

- well fit for systems with stationary nature
- hard assumptions
  - arrival rate
  - service rate
- requires deep knowledge about system internals

Related work
- Web and App tier scaling\textsuperscript{18}
  - peak load provisioning
- App tier scaling \textsuperscript{19}
  - regression based CPU usage approximation


\textsuperscript{19} Zhang, Cherkasova, and Smirni, “A Regression-Based Analytic Model for Dynamic Resource Provisioning of Multi-Tier Applications”, ICAC '07
Time series

- CPU, RAM usage traces
- repeating patterns, future value prediction
  - AR, MA, ARMA, FFT
- low overhead
- simple model
- long historical data
- sensitive to parameters values
Time series (Related work)

- **SmartScale**\(^7\)
  - polynomial approximation, workload prediction, VM (AS scaling) resizing (CPU, RAM)
- **CloudScale**\(^9\)
  - FFT, resource usage prediction (CPU or RAM)
  - VM migration
- **Press**\(^{20}\)
  - FFT, resource usage prediction (CPU)
- **AGILE**\(^{21}\)
  - wavelets transform, resource usage prediction (CPU)

---

\(^{7}\) Dutta et al., “SmartScale: Automatic Application Scaling in Enterprise Clouds”, CLOUD ’12

\(^{9}\) Shen et al., “CloudScale: elastic resource scaling for multi-tenant cloud systems”, SOCC ’11

\(^{20}\) Gong, Gu, and Wilkes, “PRESS: PRedictive Elastic ReSource Scaling for cloud systems”, CNSM ’10

\(^{21}\) Nguyen et al., “AGILE: Elastic Distributed Resource Scaling for Infrastructure-as-a-Service”, ICAC ’13
Reinforcement learning (RL)

- trial-and-error search
  - agent learns from experience
  - agent takes actions and observes environment
- Q-learning algorithm
  - exploration (ε-greedy policy)
  - exploitation

![Diagram of reinforcement learning](image)
RL. Q-learning, initialization

Horizontal scaling

Legend:
- state
  - U, W - VMs allocated
  - W - workload
- transition
  - [a, q, r]
  - a - action, add/release n VMs
  - q - value
  - r - reward
RL. Q-learning, exploration

action selection:
- random
- guided exploration

Q-value update:
$$Q(s,a) = Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

- $\alpha$ - learning rate (0.5)
- $r$ - reward
- $\gamma \in [0, 1]$ - discount factor (0.8)

Reward:
- application performance
- resource usage

Example:
- $r = 0.3, \alpha = 1$
- $q = 0 + \alpha (0.3 + \gamma 0 - 0) = 0.3$
RL. Q-learning, exploitation

- **ε - greedy policy**
  - perform exploration with $\epsilon$ probability

- **State complexity**
  - N- variations for resource (1 to 10 VMs)
  - M- variations for workload (10, 20, 30,.., 100 req/sec)
  - states number N*M

- **Vertical scaling**
  - (CPU, RAM, Workload) → states number N*K*M
Reinforcement learning (cont)

- no a priori knowledge is required
- large state space
- long learning time

Related work

- VCONF\textsuperscript{22} (CPU scaling, NN approximation)
- VirtRL\textsuperscript{23} (AS scaling)
- CoTuner\textsuperscript{24} (CPU, RAM scaling, downhill simplex method, less states)
- URL\textsuperscript{25} (CPU, RAM scaling, NN approximation)

---

\textsuperscript{22} Rao et al., “VCONF: a reinforcement learning approach to virtual machines auto-configuration”, ICAC ’09

\textsuperscript{23} Dutreilh et al., “From Data Center Resource Allocation to Control Theory and Back”, CLOUD ’10

\textsuperscript{24} Bu, Rao, and Xu, “A Model-free Learning Approach for Coordinated Configuration of Virtual Machines and Appliances”, MASCOTS ’11

### Related work summary

Legend:
WS(Web server), AS(Application server), DB(database)
RR(request rate), RT(response time)

<table>
<thead>
<tr>
<th>Ref</th>
<th>Auto-scaling technique</th>
<th>H/V Scaling</th>
<th>Target tier</th>
<th>Metric</th>
<th>Workload</th>
<th>Sys. Modeling</th>
<th>Benchmark</th>
</tr>
</thead>
<tbody>
<tr>
<td>[22]</td>
<td>RL</td>
<td>V(CPU)</td>
<td>AS</td>
<td>CPU, RT</td>
<td>Synt.</td>
<td>Online</td>
<td>RUBiS, TPC-W</td>
</tr>
</tbody>
</table>
Model change detection

- SLA violation
  - first occurrence\(^{13}\)
- workload classification\(^{14}\)
  - bidding, browsing
- repeated misdirection\(^{20}\)

---

\(^{13}\) Zheng et al., “JustRunIt: experiment-based management of virtualized data centers”, USENIX’09

\(^{14}\) Vasić et al., “DejaVu: accelerating resource allocation in virtualized environments”, ASPLOS XVII

\(^{20}\) Gong, Gu, and Wilkes, “PRESS: PRedictive Elastic ReSource Scaling for cloud systems”, CNSM ’10
interactive and batch applications
- interactive has high priority
  - rent resources from neighbor VM
  - give back unused resources
- resource: CPU
- AR based short-term CPU usage prediction

---

11 Yazdanov and Fetzer, “Vertical Scaling for Prioritized VMs Provisioning”, CGC ’12
interactive application
state(MEM, CPU) - VM capacity
action(add, keep remove)
parallel learning with assumption
- use observation from new state
  - resource usage
  - application performance
- update transitions to the states with enough resources

\[\text{State 1 [768:40]}\]
\[\text{State 2 [768:55]}\]
\[\text{State 3 [768:45]}\]
\[\text{State 4 [768:50]}\]
\[\text{State 5 [768:35]}\]

Action 1: (keep, add 15)
Action 2: (keep, add 5)
Action 3: (keep, add 10)
Action 4: (keep, remove 5)

---

\[\text{26 Yazdanov and Fetzer, “VScaler: Autonomic Virtual Machine Scaling”, CLOUD’13}\]
VScaler: Autonomic VM scaling (cont.)

- StandardRL
  - long convergence time
- VScalerRL
  - quickly finds optimal policy

![Cost per time step ($) vs Time (min)](chart.png)
VScalerLight

(a) CPU vs response time

(b) RAM vs response time

\(^{27}\)accepted CLOUD 2014)
interactive application

- 2-tiers: WS, DB

- state-space size reduction from $N \times M$ complexity to $N$ and $M$

- models updated in parallel

- CPU, response time control

- RAM, avoid swapping
Future work

- multi-tier web application
- short time scaling (seconds)
  - adjust VM resource allocation (vertical)
- long term scaling (hours)
  - horizontal scaling
Future work

- map-reduce applications resource allocation
  - slower VM less costs
  - job completion time control
Thank you