Stratus
Cost-aware container scheduling in the public cloud

Andrew Chung
Jun Woo Park, Greg Ganger

PARALLEL DATA LABORATORY
Carnegie Mellon University
Motivation

- IaaS CSPs provide per-time VM rental of diverse offerings
  - VM types and sizes
  - Contract types (e.g., reliable/on-demand, dynamically-priced/spot,…)
- Can add/remove VMs from virtual cluster (VC) any time
  - VMs paid-for by-the-second while rented
  - Pay for full VM even if only partially used!
- Mgmt complex, **but** sched research has not focused on **both**
  1. Dynamically-sized clusters
  2. Clusters with wide diversity of instance types, sizes, and contracts
Motivation

• IaaS CSPs provide per-time VM rental of diverse offerings
  • VM types and sizes
  • Contract types (e.g., reliable/on-demand, dynamically-priced/spot, …)

How can we take advantage of diverse offerings and virtual cluster elasticity to lower cost of executing batch workloads?

• Mgmt complex, **but** sched research has not focused on **both**
  1. Dynamically-sized clusters
  2. Clusters with wide diversity of instance types, sizes, and contracts
Public cloud sched properties

- **Property 1**: Wasted resource-time is wasted money

- **Money-saving key**: Minimize resource-time “bubbles”
  1. **Resource-cost-awareness**: Pick right-sized, cost-eff VMs
  2. **Efficiently using rental time**: Keep VMs highly utilized when rented, release VMs if no pending tasks

Empty VM

---

Now

Carnegie Mellon
Parallel Data Laboratory

http://www.pdl.cmu.edu/

Andrew Chung, SoCC 2018
Public cloud sched properties

- **Property 1:** Wasted resource-time is wasted money
  - **Money-saving key:** Minimize resource-time “bubbles”
    1. **Resource-cost-awareness:** Pick right-sized, cost-eff VMs
    2. **Efficiently using rental time:** Keep VMs highly utilized when rented, release VMs if no pending tasks

Example where VM resource-time is wasted

```
Now                      Time
+------------------------+-------------------+
| Task A | Task B | Task C |
+------------------------+-------------------+
```

Carnegie Mellon
Parallel Data Laboratory

http://www.pdl.cmu.edu/

Andrew Chung, SoCC 2018
Public cloud sched properties

- **Property 1:** Wasted resource-time is wasted money

  - **Money-saving key:** Minimize resource-time “bubbles”
    1. **Resource-cost-awareness:** Pick right-sized, cost-eff VMs
    2. **Efficiently using rental time:** Keep VMs highly utilized when rented, release VMs if no pending tasks

**Example where VM resource-time is wasted**

Now

Looks well-packed here, but…
Public cloud sched properties

- **Property 1**: Wasted resource-time is wasted money
  - **Money-saving key**: Minimize *resource-time “bubbles”*
    1. **Resource-cost-awareness**: Pick right-sized, cost-eff VMs
    2. **Efficiently using rental time**: Keep VMs highly utilized when rented, release VMs if no pending tasks

Example where VM resource-time is wasted
Public cloud sched properties

- **Property 1**: Wasted resource-time is wasted money
  - **Money-saving key**: Minimize *resource-time “bubbles”*
    1. **Resource-cost-awareness**: Pick right-sized, cost-eff VMs
    2. **Efficiently using rental time**: Keep VMs highly utilized when rented, release VMs if no pending tasks

**Example where VM resource-time is wasted**

<table>
<thead>
<tr>
<th>Task A</th>
<th>Task B</th>
<th>Task C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Public cloud sched properties

- **Property 1:** Wasted resource-time is wasted money
  - Money-saving key: Minimize *resource-time “bubbles”*
    1. **Resource-cost-awareness:** Pick right-sized, cost-eff VMs
    2. **Efficiently using rental time:** Keep VMs highly utilized when rented, release VMs if no pending tasks

- **Property 2:** Possible to have no task queue time
  - Replaced by VM spin-up time
  - Allows bounded workload latency
Overview and goals

- **Stratus**: VC sched middleware for public clouds
  - Suited for collections of batch jobs
  - How to size VC and where to place tasks

- **Goals**: Lower the cost of executing batch workloads with minimum makespan impact
  - Cost-efficiency by reducing “resource bubbles”
  - Makespan-minimization by sched tasks as they arrive
Efficiently using rental time

- Ideally, all tasks assigned to VM finish at the same time
  - 0% utilized (new) → 100% utilized → 0% utilized → released
- Stratus packs tasks on VMs to align task runtimes
  - Does so with a new technique: runtime binning

Stratus: aligning task runtimes

Now

Time
Efficiently using rental time

- Ideally, all tasks assigned to VM finish at same time
  - 0% utilized (new) → 100% utilized → 0% utilized → released
- Stratus packs tasks on VMs to align task runtimes
  - Does so with a new technique: runtime binning

Bad alignment of task runtimes

<table>
<thead>
<tr>
<th>Task A</th>
<th>Task B</th>
<th>Task C</th>
</tr>
</thead>
</table>

Now

Time

Bubbles
Runtime (RT) binning

- RT bins: logical bins of disjoint time intervals sized $\exp(x)$
  - $[\text{now} = 0, 1), [1, 2), [2, 4), [4, 8), [8, 16), \ldots$, and so on
- Task assigned to bin according to remaining runtime \textit{from now}
  - Ex: Task A, which runs for 11 more time units, in blue bin ([8, 16))
Runtime (RT) binning

- RT bins: logical bins of disjoint time intervals sized $\exp$ (now = 0, 1), [1, 2), [2, 4), [4, 8), [8, 16), ..., and so on
- Task assigned to bin according to remaining runtime \textit{from now}
  - Ex: Task A, which runs for 11 more time units, in blue bin ([8, 16))
- VM assigned to bin based on longest remaining task RT
  - Ex: VM with only Task A assigned to blue bin $\rightarrow$ blue border
Runtime (RT) binning

- RT bins: logical bins of disjoint time intervals sized \( \exp\) of the form \([0, 1), [1, 2), [2, 4), [4, 8), [8, 16), \ldots\), and so on
- Task assigned to bin according to remaining runtime from now
  - Ex: Task A, which runs for 11 more time units, in blue bin ([8, 16))
- VM assigned to bin based on longest remaining task RT
  - Ex: VM with only Task A assigned to blue bin → blue border
Runtime (RT) binning

- RT bins: logical bins of disjoint time intervals sized $\exp$ 
  - $[\text{now} = 0, 1), [1, 2), [2, 4), [4, 8), [8, 16), \ldots$, and so on 
- Task assigned to bin according to remaining runtime from now 
  - Ex: Task A, which runs for 11 more time units, in blue bin ($[8, 16)$) 
- VM assigned to bin based on longest remaining task RT 
  - Ex: VM with only Task A assigned to blue bin → blue border
Runtime (RT) binning

- RT bins: logical bins of disjoint time intervals sized \( \exp \)
  - \([\text{now} = 0, 1), [1, 2), [2, 4), [4, 8), [8, 16), \ldots\), and so on
- Task assigned to bin according to remaining runtime \textit{from now}
  - Ex: Task A, which runs for 11 more time units, in blue bin \([8, 16))\)
- VM assigned to bin based on longest remaining task RT
  - Ex: VM with only Task A assigned to blue bin → blue border
Packing tasks to VMs

- Packing preference for task in runtime bin $\beta$
  - VM in $\beta$ > VM in greater RT bins > VM in lesser RT bins
  - Least impact to extend VM time-to-release

Now 1 2 4 8

Task A
Packing tasks to VMs

- Packing preference for task in runtime bin $\beta$
  - VM in $\beta$ > VM in greater RT bins > VM in lesser RT bins
  - Least impact to extend VM time-to-release

Now 1 2 4 8

Task A
Packing tasks to VMs

- Packing preference for task in runtime bin $\beta$
  - VM in $\beta$ > VM in greater RT bins > VM in lesser RT bins
  - Least impact to extend VM time-to-release

Now 1 2 4 8

Task A

Full
Packing tasks to VMs

- Packing preference for task in runtime bin $\beta$
  - VM in $\beta >$ VM in greater RT bins > VM in lesser RT bins
  - Least impact to extend VM time-to-release
Packing tasks to VMs

- Packing preference for task in runtime bin $\beta$
  - VM in $\beta >$ VM in greater RT bins $> $ VM in lesser RT bins
  - Least impact to extend VM time-to-release
Packing tasks to VMs

- Packing preference for task in runtime bin $\beta$
  - VM in $\beta >$ VM in greater RT bins $>$ VM in lesser RT bins
  - Least impact to extend VM time-to-release

- Only scale-out as last resort

---

<table>
<thead>
<tr>
<th>Now</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Full</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Full</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task A</td>
<td></td>
<td></td>
<td>Full</td>
<td>Scale-out</td>
</tr>
</tbody>
</table>
Resource-cost-awareness

- Stratus performs dynamic selection of VC composition
  - Acquire new VMs only if tasks don’t fit on any VMs
  - Release VMs as soon as they become empty
- Recall: diverse offerings and dynamic pricing of VMs
- **Key**: Resource-cost-aware scale-out that considers *both* packing of pending tasks & dynamic rental costs
  - Eval packing of combinations of tasks in *same runtime bin* on to candidate VMs based on cost-per-resource-utilized
  - Packing/scaling in isolation with another increases cost
Acquiring new VMS

Pending tasks
Acquiring new VMS
Acquiring new VMS

Pending tasks

$-\text{per-resource-utilized}: \$0.9$
Acquiring new VMS

$-\text{per-resource-utilized}: \$0.9$
Acquiring new VMS

Pending tasks

$\text{-per-resource-utilized: } $0.5
Acquiring new VMS

Pending tasks

$-per-resource-utilized: $2.0

Now 1 2 4 8

http://www.pdl.cmu.edu/
Acquiring new VMS

Pending tasks

$-per-resource-utilized: $0.5
Acquiring new VMS

Pending tasks

$-per-resource-utilized: $0.9
Acquiring new VMS

$\text{-per-resource-utilized: } $0.9

Pending tasks

<table>
<thead>
<tr>
<th>Now</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
</tr>
</thead>
</table>

Carnegie Mellon
Parallel Data Laboratory

http://www.pdl.cmu.edu/

Andrew Chung, SoCC 2018
Acquiring new VMS

Pending tasks

Now 1 2 4 8

Carnegie Mellon
Parallel Data Laboratory
http://www.pdl.cmu.edu/
Acquiring new VMS
Runtime mis-estimates

- Mis-estimates can lead to low resource utilization

Example: 4 tasks on 2 instances

- Task A (over-est)
- Task B (correct-est)
- Task C (under-est)
- Task D (correct-est)
Runtime mis-estimates

- Mis-estimates can lead to low resource utilization

Example: 4 tasks on 2 instances

Bubble

- Task B (correct-est)

- Task C (under-est)

- Task D (correct-est)
Runtime mis-estimates

- Mis-estimates can lead to low resource utilization

Example: 4 tasks on 2 instances

```
Now 1 2 4 8
```

- Task C (new est)
- Bubble

Released
Runtime mis-estimates

• Mis-estimates can lead to low resource utilization
• RT binning mitigates mis-estimates to some degree

Adjusting mis-estimates

• Over-estimates: No adjustment necessary (task done)
• Under-estimates: Assume task has run for half of its runtime

Instance-clearing: If VM experiences low utilization for extended period of time, migrate tasks and re-distribute
Experimental setup

- Simulation-based experiments
  - Workloads: Google and TwoSigma cluster traces
- Focus on batch jobs
  - Filter out jobs running > 1 day
- EC2 spot market for dynamically-priced markets
  - Same family VMs for comparable perf
Evaluation: Normalized cost

Fleet (Spot Fleet + ECS, Amazon offerings)

- LowestPrice + BinPack policy

Note: Comparisons vs other state-of-the-art-schedulers left out for brevity
Stratus lowers cost vs each compared scheduler by at least 17%
Evaluation: Normalized cost

**Stratus**

- 17% (Google) and 22% cost reduction (TwoSigma)

Note: Comparisons vs other state-of-the-art-schedulers left out for brevity. Stratus lowers cost vs each compared scheduler by at least 17%.
Summary

- Packing/scaling heuristics based on runtime binning
  - Allows for high utilization of resources during rental period
- Scale VC by simultaneous consideration of possible packings and available instance types and prices
  - Independent consideration of packing/scaling leads to higher cost
- ~17% cost reduction on Google and TwoSigma traces compared to next-best evaluated scheduler
  - Attains high resource utilization