Mitigating Resource Contention and Heterogeneity in Public Clouds for Scientific Modeling Services

Wes Lloyd, Shrideep Pallickara, Olaf David, Mazdak Arabi, Ken Rojas
April 6, 2017
Institute of Technology, University of Washington, Tacoma, Washington USA

IC2E 2017: IEEE International Conference on Cloud Engineering

Outline

- Background
- Research Questions
- Experimental Workloads
- Experiments/Evaluation
- Conclusions

Rosetta Protein Folding

- Computational methods for accurate design of new hyperstable constrained peptides
- In 53 hours, using 5,904 EC2 compute cores:
  - Generated 5.2 million peptide structures
  - $3,400 spot instances
  - Upfront cost of physical cluster to achieve same result in ~53 hours: $857,752
- Cloud enables adhoc large-scale experimentation

Research Challenges

- How can we improve performance and costs for hosting scientific application workloads on the cloud?
  - Resource heterogeneity
  - Resource contention
- Relative to:
  - HPC
  - Compute clusters

VM-type heterogeneity- Amazon EC2

From: Is The Same Instance Type Created Equal
2013 IEEE Transactions on Cloud Computing

<table>
<thead>
<tr>
<th>Instance Type</th>
<th>CPU Allocation</th>
<th>C/E</th>
<th>% (2011)</th>
<th>% (2012)</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1.small</td>
<td>ES45</td>
<td>2.0</td>
<td>5%</td>
<td>30%</td>
</tr>
<tr>
<td>ES430</td>
<td>2.66</td>
<td>34%</td>
<td>30%</td>
<td></td>
</tr>
<tr>
<td>ES507</td>
<td>2.36</td>
<td>5%</td>
<td>5%</td>
<td></td>
</tr>
<tr>
<td>2218HE</td>
<td>2.6</td>
<td>18%</td>
<td>20%</td>
<td></td>
</tr>
<tr>
<td>m1.large</td>
<td>ES645(A1)</td>
<td>2.0</td>
<td>5%</td>
<td>42%</td>
</tr>
<tr>
<td>ES430(A2)</td>
<td>2.66</td>
<td>29%</td>
<td>17%</td>
<td></td>
</tr>
<tr>
<td>ES507(A3)</td>
<td>2.26</td>
<td>58%</td>
<td>40%</td>
<td></td>
</tr>
<tr>
<td>2218HE</td>
<td>2.6</td>
<td>4%</td>
<td>1%</td>
<td></td>
</tr>
<tr>
<td>270</td>
<td>2.0</td>
<td>4%</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>m2.xlarge</td>
<td>ES645</td>
<td>2.0</td>
<td>40%</td>
<td>48%</td>
</tr>
<tr>
<td>ES430</td>
<td>2.66</td>
<td>46%</td>
<td>46%</td>
<td></td>
</tr>
<tr>
<td>ES507</td>
<td>2.26</td>
<td>31%</td>
<td>6%</td>
<td></td>
</tr>
<tr>
<td>270</td>
<td>2.0</td>
<td>2%</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>
Trial-and-better Resource Provisioning

- Z. Ou et al., 2013 IEEE Trans. on Cloud Computing
- Using Amazon EC2
  1. Provision instances
  2. Perform trial(s) - VM testing
  3. Keep desired instances
  4. Replace undesirable instances

Test: Underlying CPU Type

VM-Scaler

- Harness this approach for VM-Pools
- Ensure every VM has same backing CPU
- Provide more consistent test results

Resource Utilization Data Collection

- Profile resource utilization for scientific workloads running across many VMs
- Sensor on every VM
  - Transmits data to VM-Scaler

Trial and Better – VM-Scaler

- Web services application
  - Rest-based/JSON
  - Harnesses EC2 API
  - Manages virtual cloud infrastructure
  - Supports scientific modeling-as-a-service
  - Supports Amazon, Eucalyptus clouds

CpuSteal

- CpuSteal: VM's CPU core is ready to execute but the physical CPU core is busy
- Symptom of over provisioning physical servers
- Factors which cause CpuSteal:
  1. Processors shared by too many busy VMs
  2. Hypervisor kernel (Xen dom0) is occupying the CPU
  3. VM's CPU time share <100% for 1 or more cores, and 100% is needed for a CPU intensive workload.
Mitigating Resource Contention and Heterogeneity in Public Clouds for Scientific Modeling Services

Outline

- Background
- Research Questions
- Experimental Workloads
- Experiments/Evaluation
- Conclusions

Research Questions

RQ1: How common is public cloud VM-type implementation heterogeneity?

RQ2: What performance implications result from VM-type heterogeneity for hosting scientific application workloads?

Research Questions - 2

RQ3: How effective is cpuSteal at identifying VMs with high resource contention due to multi-tenancy (e.g. noisy neighbor VMs) in a public cloud?

RQ4: What are the performance implications of hosting scientific modeling workloads on worker VMs with consistently high cpuSteal measurements in a public cloud? Is there a pattern to cpuSteal behavior across worker VMs over time?

CSIP Model Services

- Cloud Services Innovation Platform
  - Java-based framework to support development of scientific model services (modeling-as-a-service)
  - Increase availability and throughput of models
  - Harness scalable cloud infrastructure
  - Cloud virtualization supports variety of legacy software required for scientific applications
    - (e.g. FORTRAN, Visual C++ 6.0, etc.)

Scientific Application Workloads

- Rusle2
  - Soil erosion from water
  - Median runtime ~1.89s

- WEPS
  - Soil erosion from wind
  - Median runtime ~55s
  - Years weather data * Years of crop rotation
Testing for VM Type Heterogeneity

- Identified CPU by checking `/proc/cpuinfo`
  - Launched 50 VMs of a given type
  - If there was heterogeneity, launched 50 more
- Tested 12 VM types, across 3 generations
  - 1st: m1.medium, m1.large, m1.xlarge, c1.medium, c1.xlarge
  - 2nd: m2.xlarge, m2.2xlarge, m2.4xlarge
  - 3rd: c3.large, c3.xlarge, c3.2xlarge, m3.large

Amazon EC2 VM Type Heterogeneity

<table>
<thead>
<tr>
<th>VM type</th>
<th>Region</th>
<th>Backing CPU</th>
<th>Backing CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1.medium</td>
<td>us-east-1</td>
<td>Intel E5-2650 v0 8c, 95w, 96%</td>
<td></td>
</tr>
<tr>
<td>m2.xlarge</td>
<td>us-east-1</td>
<td>Intel Xeon X5550 4c, 95w, 48%</td>
<td></td>
</tr>
<tr>
<td>m1.large</td>
<td>us-east-1</td>
<td>Intel Xeon E5-2651 v2 12c, 115w, 19%</td>
<td></td>
</tr>
<tr>
<td>m2.xlarge</td>
<td>us-east-1</td>
<td>Intel Xeon E5-2665 v0 8c, 115w, 42%</td>
<td></td>
</tr>
</tbody>
</table>

VM Type Heterogeneity Performance Implications

- Tested small 5 VM pools
- Compared the two most abundant hardware implementations
  - m1.large - Intel Xeon
    - E5-2650 v0, 8 cores, 95 w vs. E5-2651 v2, 12 cores, 105 w
  - m2.xlarge - Intel Xeon
    - E5-2665 v0, 8 cores, 115 w vs. X5550, 4 cores, 95 w
- Workloads
  - WEPS: 10 x 100 runs
  - RUSLE2: 10 x 660 runs
Noisy Neighbor (NN-Detect) Detection Methodology

- Noisy neighbors cause resource contention and degrade performance of worker VMs
- Identify noisy neighbors by analyzing cpuSteal

Detection method:
Step 1: Execute processor intensive workload across pool of VMs.
Step 2: Capture total cpuSteal for each VM for the workload.
Step 3: Calculate average cpuSteal for the workload (cpuSteal_{avg}).

Identify NNs using statistical outliers, and assigning application specific thresholds through observation...

Amazon EC2 CpuSteal Analysis

<table>
<thead>
<tr>
<th>VM Type</th>
<th>Host CPU</th>
<th>Avg CPU</th>
<th>% with Noisy Neighbors</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1.large-1c</td>
<td>E5-2680v2/12c</td>
<td>1.86</td>
<td>0%</td>
</tr>
<tr>
<td>m2.large-2c</td>
<td>E5-2650v0/8c</td>
<td>3.14</td>
<td>0%</td>
</tr>
<tr>
<td>m3.medium-1c</td>
<td>X5550/4c</td>
<td>7.25</td>
<td>4%</td>
</tr>
<tr>
<td>c1.xlarge-8c</td>
<td>E5-2670v2/12c</td>
<td>7.62</td>
<td>12%</td>
</tr>
</tbody>
</table>

Step 1: Execute processor intensive workload across pool of VMs.
Step 2: Capture total cpuSteal for each VM for the workload.
Step 3: Calculate average cpuSteal for the workload (cpuSteal_{avg}).

Identify NNs using statistical outliers, and assigning application specific thresholds through observation...

Noisy Neighbor Performance Degradation

- Compared performance of small 5 VM pools
  - 5 Noisy-Neighbor VMs
  - 5 regular VMs
- WEPS: 10 x 100 runs
- RUSLE2: 10 x 660 runs
- Normalized results to VM pools w/o NN’s
EC2 Noisy Neighbor Performance Degradation

<table>
<thead>
<tr>
<th>VM type</th>
<th>Region</th>
<th>WEPS</th>
<th>RUSLE2</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1.large</td>
<td>us-east-1c</td>
<td>117.68%</td>
<td>125.42%</td>
</tr>
<tr>
<td>E5-2650v0/8c</td>
<td></td>
<td>p=6.847·10^-8</td>
<td>p=0.016</td>
</tr>
<tr>
<td>m2.xlarge</td>
<td>us-east-1c</td>
<td>107.3%</td>
<td>102.76%</td>
</tr>
<tr>
<td>X5550/4c</td>
<td></td>
<td>p=0.05232</td>
<td>p=1.73·10^-11</td>
</tr>
<tr>
<td>c1.xlarge</td>
<td>us-east-1c</td>
<td>100.73%</td>
<td>102.91%</td>
</tr>
<tr>
<td>E5-2651v2/12c</td>
<td></td>
<td></td>
<td>n.s.</td>
</tr>
<tr>
<td>m1.medium</td>
<td>us-east-1d</td>
<td>111.6%</td>
<td>104.32%</td>
</tr>
<tr>
<td>E5-2650v0/8c</td>
<td></td>
<td>p=6.25·10^-8</td>
<td>p=1.173·10^-5</td>
</tr>
</tbody>
</table>

Key Result #1
Maximum performance loss:
WEPS 18%, RUSLE2 25%

Key Result #2
3 VM types with significant performance loss (p <0.05)
Average performance loss: WEPS/RUSLE2 ~ 9%

Outline
- Background
- Research Questions
- Experimental Workloads
- Experiments/Evaluation
- Conclusions

Conclusions
- Hardware (CPU-type) heterogeneity is more prolific for legacy instance implementations
- Performance variance up to ~14%
- For large VM pools, 4 of 9 instance types showed had noisy neighbors which produced statistically significant performance variance
- Performance variance up to ~25%

Future Work
- VM-Scaler: Trail-and-better VM pool creation
  - Provide cpuSteal Noisy Neighbor detection
  - Consider are we detecting ourselves? other users?
- Extend noisy neighbor detection techniques
  - Memory, disk, network resource contention
- Evaluate new instance types, public clouds, and science applications

EC2 Noisy Neighbor Performance Degradation

Questions