Review:

Is the Same Instance Type Created Equal?

Exploiting Heterogeneity of Public Clouds

Zhonghon Ou et al.
IEEE Trans. On Cloud Computing – 2013

Wes Lloyd
Computer Science
Institute of Technology – UW Tacoma
wlloyd@uw.edu
http://faculty.washington.edu/wlloyd/

Outline

- Problem introduction
- Background / related work
- Approach
- Key contributions
- Results

Problem Introduction

- Cloud systems trend towards hardware heterogeneity over time
  - Some applications become dependent on specific instance types due to significant benchmarking and performance tuning
  - Cloud provider gradually replaces old hardware
  - Legacy hardware no longer available, can not be repaired or replaced
  - Must reimplement infrastructure with new HW

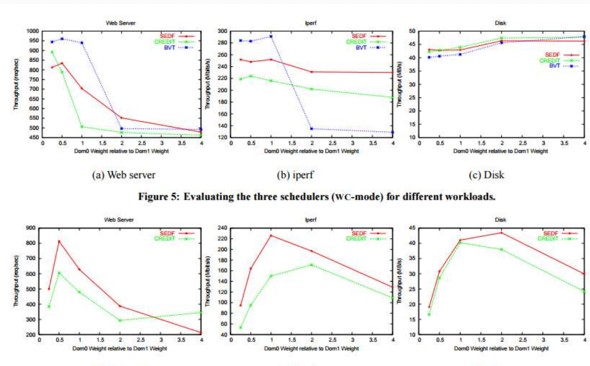
Problem Introduction - 2

- Typical HW lifecycle is 3-5 years
- Over time hardware heterogeneity appears to increase:
  - Problem is quite observerable with 1st and 2nd generations Amazon EC2 VM instance types
- VM's implemented using XEN hypervisor
- Cloud Providers may implement VMs with different XEN CPU Scheduler Configurations

Problem Introduction - 3

- XEN CPU Schedulers
  - SEDF- fixed % allotments of CPU cores to vCPUs
    - No load balancing of vCPUs across CPUs
    - Specifies minimum amount of CPU time for a given period
    - Boolean flag specifies to receive extra, unused cycles
  - Credit- weights CPU cores to vCPUs
    - Supports load balancing vCPUs across CPUs
    - vCPUs are context switched at 30 ms time slices
    - Weight – number of credits
    - Cap- percentage of extra CPU time that can be received

Performance - XEN CPU Scheduling



Background / Related Work

- High-performance computing (HPC) cloud comparison studies
- System performance comparisons
- Exploiting heterogeneity in the cloud

Related Work: HPC/Cloud Comparison

- Walker: 2008 Study, compared EC2 instances with traditional scientific cluster
  - Performance gaps observed
- Jackson et al. 2010, comprehensive comparison of HPC to AWS cloud study
- Zhai et al. 2011, Cloud vs. private cluster for *Message Passing Interface (MPI)* parallel applications

Related Work: System Performance Comparison

- Li et al. 2010, Developed cost comparator (CloudCmp) to measure various cloud services
  - Later developed CloudProphet to predict end-to-end response time of on premise web applications deployed to public cloud
- Lenk et al. 2011, Identified that performance indicators are insufficient to compare IaaS offerings
- Wang and Ng., 2010, Virtualization's impact on network performance
- Schad., 2010, Performance variance study
- Barker., 2010, Evaluation of latency sensitive applications

Related Work: Exploiting heterogeneity

- Suneja et al., 2011, Harness GPU cycles for cloud management and hypervisor tasks
  - Reduce overhead
- Lee et al., 2011, Hadoop scheduler on EC2 instances which considers job progress are resource requirement variation (e.g. CPU, I/O)
- Samih et al., manage and share cluster memory dynamically, swap memory pages to other servers with excess memory capacity

Related Work: Exploiting Heterogeneity - 2

- Farlet et al., 2012, Hardware variation leads to performance variance of instance types (m1.small)
- Evaluation limited to one-week of experimental runs
- Only studied m1.small instance type

Approach

- Examined hardware heterogeneity of Amazon and Rackspace cloud resources
- Considered: memory, disk, CPU
- Investigated impact of hypervisor scheduling
- Comparison runs with a local XEN servers
- Game-theoretic and Nash equilibrium analysis
  - To model random, stochastic events

Key Contributions

- Identified VM type heterogeneity of Amazon and RackSpace Public Clouds – produced data sets
- Benchmarked inner-VM-type performance variations
- Reverse engineered XEN scheduler configurations
  - Determined time share of CPU cores
- Performance and cost improvements:
  trail-and-better VM scheduling

VM-type heterogeneity- Amazon

Instance	CPU(Alias)	GHz	%(2011)	%(2012)
m1.small	**E5645**	**2.0**	3%	30%
	E5430	2.66	34%	38%
	E5507	2.26	45%	12%
	2218HE	2.6	18%	20%
m1.large	**E5645(A1)**	**2.0**	5%	42%
	E5430(A2)	2.66	29%	17%
	E5507(A3)	2.26	58%	40%
	2218HE	2.6	4%	1%
	270	2.0	4%	-
m1.xlarge	**E5645**	**2.0**	40%	48%
	E5430	2.66	27%	46%
	E5507	2.26	31%	6%
	270	2.0	2%	-

m1.xlarge – 4 Core x 2 ECUs

m1.xlarge implementations reported in paper:

m1.xlarge	**E5645**	**2.0**	40%	48%
	E5430	2.66	27%	46%
	E5507	2.26	31%	6%
	270	2.0	2%	-

2014 observed m1.xlarge implementations:

Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz # very common
Intel(R) Xeon(R) CPU E5-2651 v2 @ 1.80GHz # less common
Intel(R) Xeon(R) CPU E5645 @ 2.40GHz # very uncommon

Power Consumption Trends:

E5430	4 cores	20 watts/core	2007
E5645	6 cores	12 watts/core	2010
E5-2650	8 cores	11.875 watts/core	2012
E5-2651	12 cores	8.75 watts/core	2013

VM-type heterogeneity-Rackspace

AMD Opteron CPUs

Instance	Dallas		Chicago	
(Memory)	CPU(Alias)	GHz	CPU(Alias)	GHz
512 MB *1 GB* *2 GB*	4170	2.1	4170	2.1
4 GB *8 GB* *15 GB* *30 GB*	4170 (R1)	2.1	2374 (R2)	2.2

Legend

- Amazon EC2: A1 / A2 / A3 **Intel XEON CPUs**

m1.large	**E5645(A1)**	**2.0**	5%	42%
	E5430(A2)	2.66	29%	17%
	E5507(A3)	2.26	58%	40%

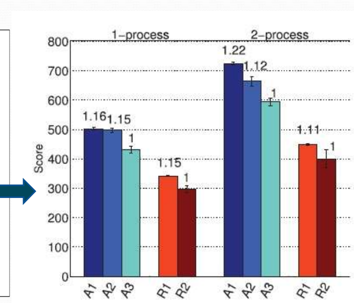
- Rackspace: R1 / R2 **AMD CPUs**

Instance	Dallas		Chicago	
(Memory)	CPU(Alias)	GHz	CPU(Alias)	GHz
512 MB *1 GB* *2 GB*	4170	2.1	4170	2.1
4 GB *8 GB*	4170 (R1)	2.1	2374 (R2)	2.2

UnixBench Scores

Unix Bench is an aggregate normalized measure of multiple metrics

These are aggregated normalized normalized numbers.



UnixBench

- 4.2: CPU Performance
  - UnixBench is used as a "CPU" Benchmark
  - Authors used standard "UnixBench" runs
- Weakness: UnixBench is ***NOT*** a CPU Benchmark!
  - It measures ***all*** aspects of a Unix machine's performance including. . .

UnixBench provides an aggregated normalized value for system performance consisting of these tests:

dhry2reg	Dhrystone 2 using register variables
whetstone-double	Double-Precision Whetstone
syscall	System Call Overhead
pipe	Pipe Throughput
context1	Pipe-based Context Switching
spawn	Process Creation
execl	Execl Throughput
fstime-w	File Write 1024 bufsize 2000 maxblocks
fstime-r	File Read 1024 bufsize 2000 maxblocks
fstime	File Copy 1024 bufsize 2000 maxblocks
fsbuffer-w	File Write 256 bufsize 500 maxblocks
fsbuffer-r	File Read 256 bufsize 500 maxblocks
fsbuffer	File Copy 256 bufsize 500 maxblocks
fsdisk-w	File Write 4096 bufsize 8000 maxblocks
fsdisk-r	File Read 4096 bufsize 8000 maxblocks
fsdisk	File Copy 4096 bufsize 8000 maxblocks
shell1	Shell Scripts (1 concurrent) (runs "looper 60 multi.sh 1")
shell8	Shell Scripts (8 concurrent) (runs "looper 60 multi.sh 8")
shell16	Shell Scripts (8 concurrent) (runs "looper 60 multi.sh 16")

Xen Scheduler Reverse-Engineering

- Call `gettimeofday()` (1) million times
  - Bare metal system: call resolution of µs
- Analyze CPU run/wait time intervals
- Derive VM scheduler acquisition percentages
  - Can extend test to multiple cores
  - Identify when cores are scheduled differently

Legend

- Amazon EC2: A1 / A2 / A3 **Intel XEON CPUs**

	E5645(A1)	**2.0**	5%	42%
	E5430(A2)	2.66	29%	17%
m1.large	E5507(A3)	2.26	58%	40%

- Rackspace: R1 / R2 **AMD CPUs**

Instance	Dallas		Chicago	
(Memory)	CPU(Alias)	GHz	CPU(Alias)	GHz
512 MB *1 GB* *2 GB*	4170	2.1	4170	2.1
4 GB *8 GB*	4170 (R1)	2.1	2374 (R2)	2.2

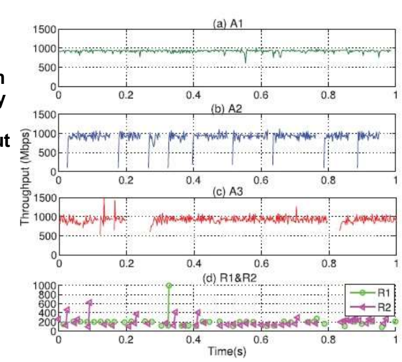
Probable XEN Scheduler Configurations

TABLE 4
CPU Acquisition Percentage of Instances

CPU	1-proc (%)	2-proc (%)	Emulation
A1	99.9	(99.9, 99.6)	Credit (256, 200)
A2	75.7	(75.4, 74.9)	SEDF (30, 40, 0)
A2-s	**99.4**	**(74.6, 74.8)**	**Credit (256, 150)**
A3	72.5	(71.1, 71.9)	SEDF (150, 210, 0)
R1	99.9	(98.1, 98.5)	Credit (512, 200)
R2	99.5	(97.3, 95.7)	Credit (512, 190)

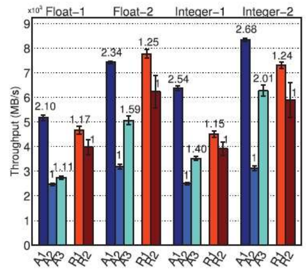
Network I/O Variance

Used TCPBench to quantify network throughput



Memory Throughput Variance

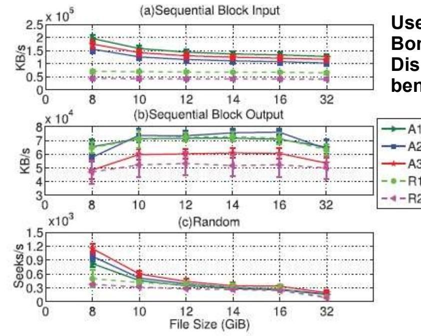
Used the RAMspeed memory benchmark



Higher is better

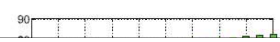
Fig. 4. Memory performance. Float- and Integer-stand for float and integer operations, and 1 and 2 stand for 1 and 2 processes, respectively.

Disk I/O Throughput Variance



Used the Bonnie++ Disk I/O benchmark!

Trial and Better Cost Savings



Theoretical Cost Savings:

Actual Performance Variance

Amazon m1.large 1.2-1.6x

Rackspace 1.2x

Fig. 7. Cost saving analysis. Different p value stands for different percentage of better-performing instances.

Trial and Better Application Testing

TABLE 6
Speedup and Overhead of "Trial-and-Better" Approach

	Speedup	Overhead
Redis	11.5%	1.5%
Httperf	28.8%	1.8%

10 instances, 100 hours
Trial 1 – Random Instances
Trial 2 – Trial-and-Better Instances

Conclusions

- Amazon EC2- Heterogeneity within the same availability zone
- Rackspace – Between different regions
- Hardware diversity produced performance variance:
  - AWS: 20% for CPU, 268% for memory
  - RS: 15% CPU, 75% disk
- VM CPU scheduling: exacerbates performance variance
- Up to 30% performance improvement with "trial and better" instance provisioning

Strengths

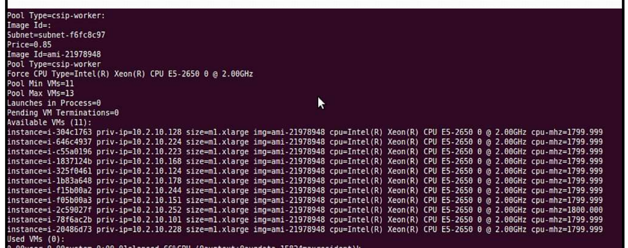
- Multidimensional study
  - Two public cloud providers
  - Many instance types
  - CPU variation
  - CPU scheduling analysis
  - Memory, Disk, Network performance analysis

Weaknesses

- UnixBench cited as a CPU benchmark
- Did not consider application performance
  - No consideration of multi-VM deployments
  - Primarily used well known benchmarks

Trial and Better – VM-Scaler

- **Harness this approach for VM-Pools**
- **Help increase homogeneity of VM pools**
- **Provide more consistent test results for cloud BMs**



Questions

Extra Slides