Characterizing Public Cloud Resource Contention to Support Virtual Machine Co-residency Prediction

Xinlei (Edward) Han, Raymond Schooley,
Delvin Mackenzie, Olaf David, Wes Lloyd

April 22, 2020

School of Engineering and Technology
University of Washington Tacoma
IC2E 2020: IEEE International Conference on Cloud Engineering

Outline

- Background
- Research Questions
- Benchmarking Suite
- Testing Infrastructure
- Experiments/Evaluation
- Conclusions

Public Cloud Virtualization Improvements

- Virtualization innovations have enabled HW virtualization of nearly all system components:
  - CPU, memory, network I/O, storage I/O, interrupts, timers
- Improvements have drastically reduced performance overhead of VMs
  - AWS Nitro virtualization claims overhead less than 1%
  - Indistinguishable from performance variance
  - Ex: Genomic sequencing ~5% variance on c5.2xlarge
- See Brendan Gregg's EC2 virtualization blog:
  - http://www.brendangregg.com/blog/2017-11-29/aws-ec2-virtualization-2017.html

Public Cloud VM Density

- Cloud servers have become increasingly core dense
- **Feb 2010**: Amazon EC2 m2.xlarge (2 vCPUs)
  - Original based on dual-CPU four-core Intel Xeon X5550 (45 nm)
  - Allowed maximum of 8 x 2 vCPU guests per host (16 vCPUs)
- **Nov 2017**: Amazon EC2 m5.large (2 vCPUs)
  - Octa-processor 24-core Intel Xeon Platinum 8175M (14 nm)
  - Presumably used in dual CPU configuration
  - Up to 48 x 2 vCPU guests per host (96 vCPUs)
- **2010 → 2017**: VMs may co-reside with 6x as many guests

Public Cloud VM Density

- Cloud servers have become increasingly core dense

What performance implications result from this increasing public cloud VM density?

  - Presumably used in dual CPU configuration
  - Up to 48 x 2 vCPU guests per host (96 vCPUs)
- **2010 → 2017**: VMs may co-reside with 6x as many guests

VM Co-residency

- Public cloud VM launch policies influence VM placement
- **EC2 spread placement**: place VM on distinct rack
- Guarantees isolation from user's own VMs
- Does not avoid resource contention from other user's VMs

- **Example**: Genomic sequencing workflow, ~90 min avg runtime
- CPU bound w/ network/disk I/O across separate phases
- Run 10 instances in parallel across separate VMs

- **Observed min/max performance Δ's:**
- Isolated VM (c5.2xlarge), ec2 dedicated host: **0.5%**
- 10 public VMs (c5.2xlarge): **9.5%** (contention from user + public VMs)
- 10 spread VMs (c5.2xlarge): **7.9%** (contention from public VMs)

VM Co-residency

Genomic sequencing: up to ~10% performance degradation
(long running batch job, public cloud VMs)

How can we infer VM co-residency to identify VMs on busy hosts to avoid resource contention ?

- 10 spread VMs (c5.2xlarge): **7.9%** (contention from public VMs)

April 22, 2020 IC2E 2020: *Characterizing Public Cloud Resource Contention to Support Virtual Machine Co-residency Prediction* 7

Outline

- Background
- Research Questions
- Benchmarking Suite
- Testing Infrastructure
- Experiments/Evaluation
- Conclusions

April 22, 2020 IC2E 2020: *Characterizing Public Cloud Resource Contention to Support Virtual Machine Co-residency Prediction* 8

Research Questions

RQ1: (Resource Contention)
What extent of performance degradation (e.g. CPU, disk, network) results from VM co-location when running identical benchmarks in parallel on a public cloud?

How is public cloud resource contention impacted by recent advancements in virtualization hypervisors and hardware?

April 22, 2020 IC2E 2020: *Characterizing Public Cloud Resource Contention to Support Virtual Machine Co-residency Prediction* 9

Research Questions

RQ2: (VM Co-residency Prediction)
How effective are performance metrics derived from CPU, disk, and network benchmarks run in parallel across VMs as independent variables to predict VM co-residency on physical hosts?

How accurate are VM co-residency predictions from multiple linear regression and random forest models trained using these features?

April 22, 2020 IC2E 2020: *Characterizing Public Cloud Resource Contention to Support Virtual Machine Co-residency Prediction* 10

Outline

- Background
- Research Questions
- Benchmarking Suite
- Testing Infrastructure
- Experiments/Evaluation
- Conclusions

Benchmarking Suite

- Automate running common system benchmarks concurrently across public cloud VM pools
- **Goals:**
- Stress the same resources at the same time
- Workload scaling: gradually increase or decrease VMs in pool that concurrently run a benchmark
  - Observe incremental performance changes
- Fast execution of entire benchmark suite
  - Support rapid assessment of VM resource contention

Benchmarking Suite - 2

- Python-based testing suite
- Leverages ssh/pssh, scp, ntp, crontab, EC2 APIs
- VMs pre-provisioned before running benchmarks
- Suite automatically includes tagged user VMs in test
- VM clocks synchronized with **ntp**
- Scripts facilitate scheduling concurrent job execution across large VM pools using **crontab**
- Scripts capture output to generate CSV data
- Data analyzed with R / Python in Jupyter notebooks

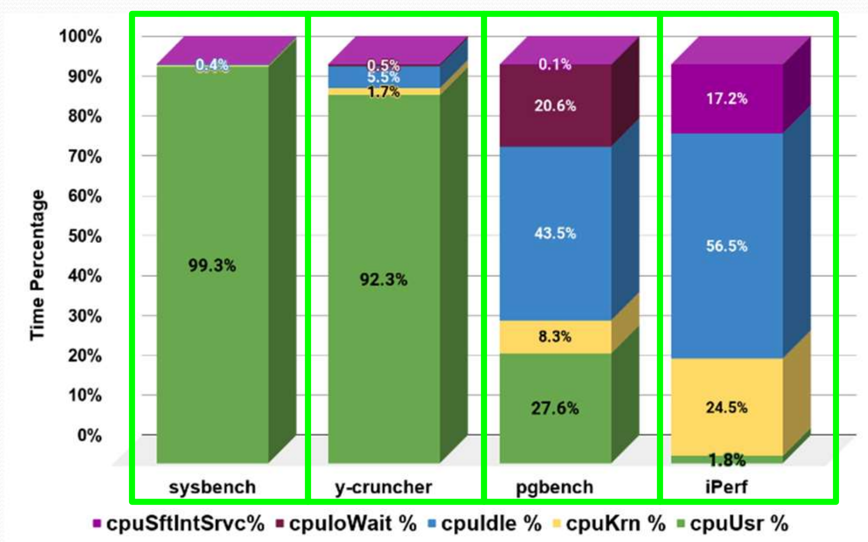
Benchmark Selection

- Emphasis on performing identical, repeatable, deterministic tasks
- Measure runtime or throughput of tasks to assess performance
- Benchmarks had to be scriptable
- Cover a broad range of resource utilization
- Benchmark should be long enough to assess behavior, but short enough to assess quickly

Benchmarks - 2

Benchmark	Description
sysbench CPU	CPU stress test: generate first 2 million prime numbers, w/ 2 threads, 10x
Y-cruncher CPU+memory	CPU+memory stress test: calculate PI to 25 million decimal digits, w/ 2 threads
pgbench CPU+memory+disk +network	PostgreSQL relational database benchmark: measured total number of transactions performed in 60 seconds (select, update, insert queries) to derive transactions per second, w/ 10 threads
iPerf network	Measure bandwidth of concurrent data transfer between client and server for 15-sec test runs Requires 2 VMs: client and server

Benchmark Resource Utilization



Outline

- Background
- Research Questions
- Benchmarking Suite
- Testing Infrastructure
- Experiments/Evaluation
- Conclusions

Public Cloud Testing

- Investigation on 3rd/4th(XEN), and 5th (AWS Nitro) generation ec2 instances
- Used EC2 dedicated hosts: isolated hosts that allow controlled placement of VMs (*rent the entire host)*
- Launched 2-vCPU instances to maximize potential for resource contention
  - Tests ranged from 16 (c3/c4) to 48 (m5d) co-resident VMs
- Instances featured 3 types of cloud storage
  - Local SSD (c3 instances)
  - Elastic Block Store (EBS) volumes-network storage (c4 instances)
  - Local nVME SSD (z1d, m5d instances)

Challenge: How to create disk contention on EBS volumes?

- **Default AWS IOPS quotas:** deter disk contention: i.e. *the provided pipe is bigger than the allowed volume*
- *EBS GP2 volume: 3000 IOPS per 2 vCPU VM Host has ~64,000 estimated IOPS total, 16 VMs/host*
- **Solution:** use provisioned IOPS volumes (16 VMs x 5000) Total 80,000 IOPS, exceeded host capacity by ~16,000 IOPS
- Technique produced performance degradation in pgbench
- Downside: benchmark somewhat expensive- *must delete volumes quickly…*
- Creating disk I/O contention only an issue on instances without local storage option

Key Performance Indicators: ec2 dedicated hosts

KPI	c3	c4	z1d, m5d
Xeon CPU model	E5-2680v2	E5-2666v3	Platinum 8191(z1d), Platinum 8175m(m5d)
family/microns/yr	Ivy Bridge-EP/ 22nm/Sep2013	Haswell-EP / 22nm/Nov2014	Skylake-SP / 14nm/Jul2017
vCPUs/host	40	40	48 (z1d), 96 (m5d)
physical CPU cores/host	20	20	24 (z1d), 48 (m5d)
Base clock MHz	2800	2900	3400 (z1d), 2500 (m5d)
Burst clock MHz (single / all)	3600/3100	3500/3200 [30]	4000/4000 (z1d), 3100/3500 (m5d) [31]
Hypervisor / virtualization-type	XEN / full	XEN / full	AWS Nitro (KVM/full)
Max # of 2 vCPU instances/host	16 x c3.large	16 x c4.large	24 x z1d.large, 48 x m5d.large
Pg db storage	16 GB local shared SSD	100GB io1 EBS volume, 5k iops	75GB local shared NVMe
Network capacity/instance	"Moderate" ~550 Mbps	"Moderate" ~550 Mbps	Up to 10 Gbps
Host price/hr	$1.848	$1.75	$4.91 (z1d), $5.97 (m5d)
VM price/hr	$.1155	$.109375	$.205 (z1d), $.124 (m5d)

Outline

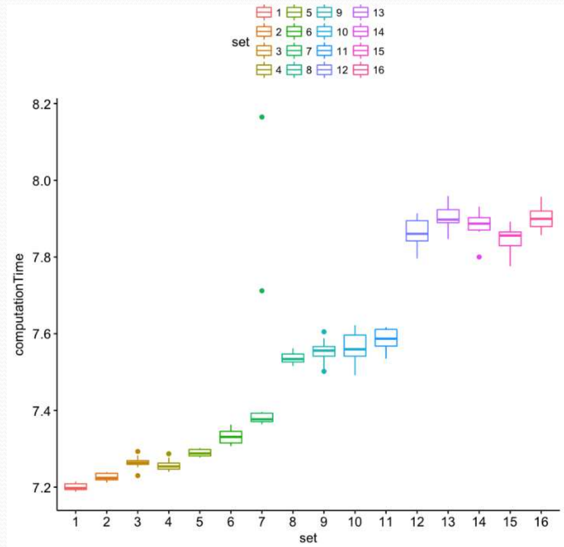
- Background
- Research Questions
- Benchmarking Suite
- Testing Infrastructure
- Experiments/Evaluation
- Conclusions

RQ-1: Resource Contention

What extent of performance degradation (e.g. CPU, disk, network) results from VM co-location when running identical benchmarks in parallel on a public cloud?

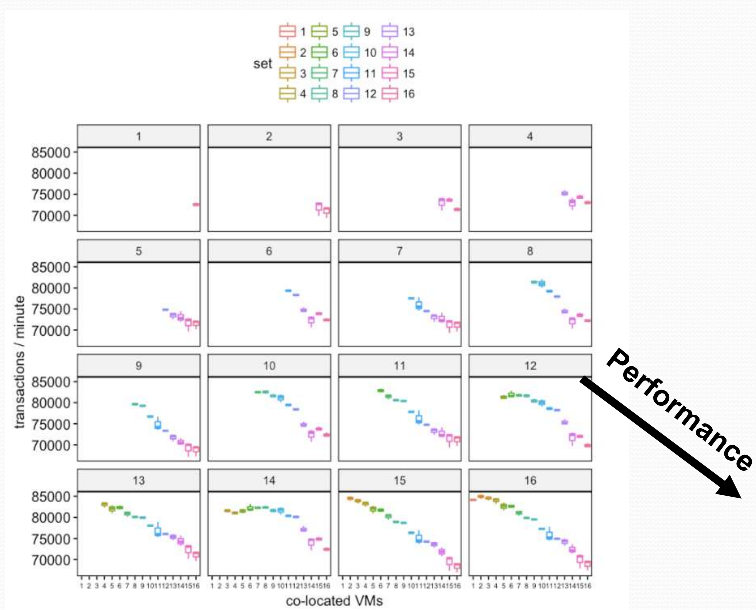
Y-cruncher Performance

- C4 dedicated host
- Each test adds an additional VM
- Scale: 1 to 16 VMs
- Performance boxplots



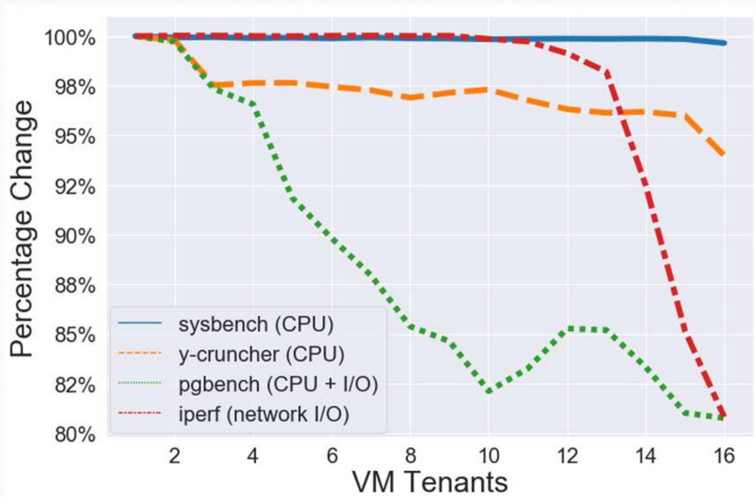
April 22, 2020 IC2E 2020: *Characterizing Public Cloud Resource Contention to Support Virtual Machine Co-residency Prediction* 23

Pgbench transaction throughput



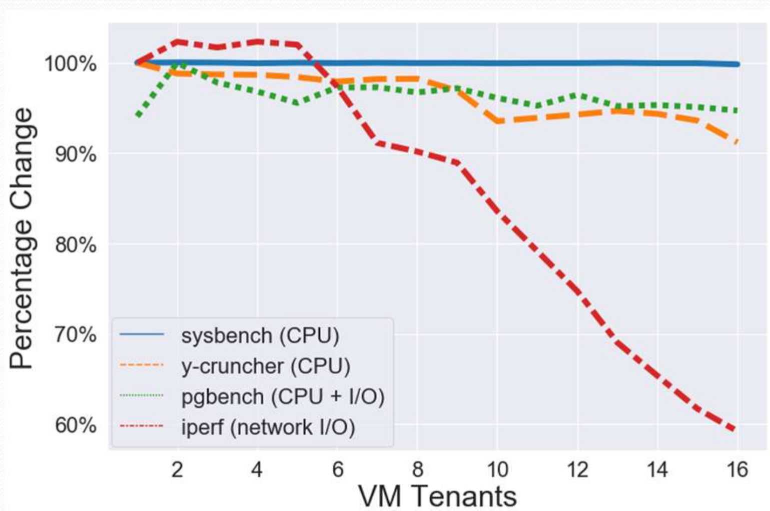
April 22, 2020 IC2E 2020: *Characterizing Public Cloud Resource Contention to Support Virtual Machine Co-residency Prediction* 24

16 x c3.large instances (3rd gen XEN)



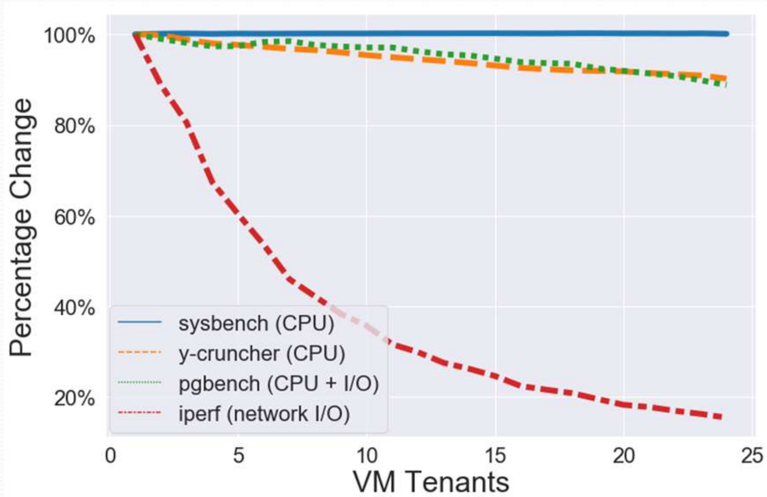
April 22, 2020 IC2E 2020: *Characterizing Public Cloud Resource Contention to Support Virtual Machine Co-residency Prediction* **25**

16 x c4.large instances (4th gen XEN)



April 22, 2020 IC2E 2020: *Characterizing Public Cloud Resource Contention to Support Virtual Machine Co-residency Prediction* **26**

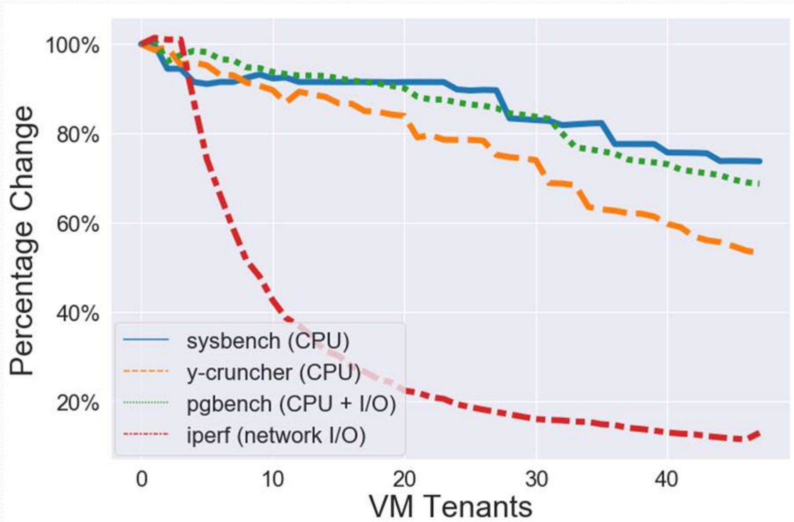
24 x z1d.instances (5th gen AWS Nitro)



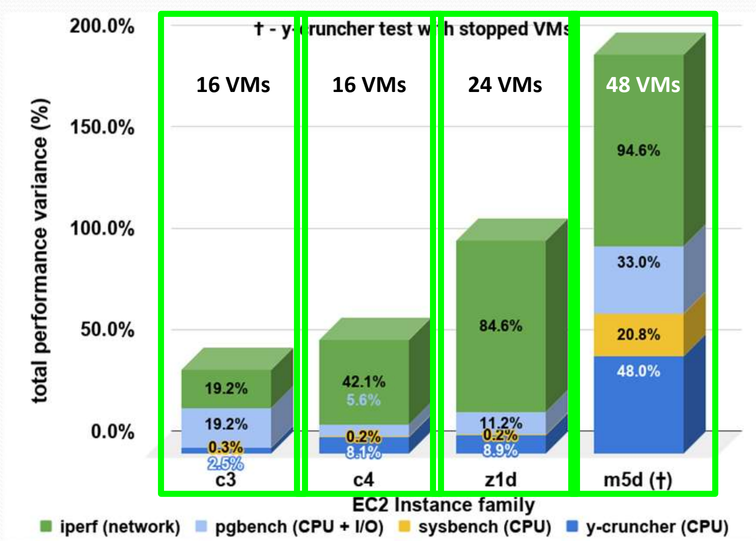
Performance Implications of Idle VMs

- **M5d.large:** (up to 48 co-resident VMs)
- **Y-cruncher**: initial performance Δ 14.96% (5.67s to 6.52s)
  - Value seemed low
  - Refactored scripts to suspend idle Ubuntu Linux VMs
  - Performance Δ increased to 47.97% (4.4s to 6.51s)
  - 47 idle instances of Linux contributed 1.27s (32.42%) slowdown in Y-cruncher runtime
- **Sysbench**: initial performance Δ: ~0.18%
  - Suspending idle VMs increased performance Δ to 20.81%
- **Z1d.large:** (up to 24 co-resident VMs)
  - Y-cruncher: idle VM shutdown increased performance Δ by 4.47%
  - Sysbench: no increase in performance Δ
- **c3.large/c4.large:** (up to 16 VMs) - no increase in performance Δ

48 x m5d.large instances (5th gen AWS Nitro)



Total observed performance change: min to max VM co-residency



RQ-2: VM Co-residency Prediction

Feature Evaluation

How effective are performance metrics derived from CPU, disk, and network benchmarks as independent variables?

31

RQ-2: VM Co-residency Prediction

Prediction Accuracy

How accurate are VM co-residency predictions from multiple linear regression and random forest models?

32

VM Co-residency Prediction

- Focus: predict number of co-resident m5d.large instances
  - Largest possible range from 1 to 48 VMs
- Trained random forest and multiple regression models
  - For multiple regression had to normalize data
- **Training data:** m5d dedicated host
  4 benchmarks x ~10 runs x 48 VM co-residency configurations
- Trained random forest and multiple regression models
- **Testing data:** m5d dedicated host:
  Reran entire benchmark suite to obtain fresh performance measurements several weeks later

VM Co-residency Model Evaluation

Independent Variable Evaluation – Random Forest

Independent Variable	Increase in Node Purity	% Increase in MSE
iPerf (throughput in MB/sec)	70,725	9.268
Sysbench (runtime in sec)	106,714	31.064
Y-cruncher (runtime in sec)	135,112	49.908
Pgbench (transactions/min)	146,221	56.220

Feature Importance Decreases

VM co-residency model evaluation

Evaluation Metric	Random Forest with raw data	MLR with normalized data
R^2	.9755	.9423
Root Mean Squared Error (RMSE)	2.479	2.175
Mean Absolute Error (MAE)	1.950	1.608
Min Prediction	4.537	-0.480
Max Prediction	47.479	49.543

Lower Predictive Error

VM Co-residency Model Evaluation

Independent Variable Evaluation – Random Forest

iF
S
Y
P

VM

Multiple regression model forecasted VM co-residency on average to within ± 1.61 VMs

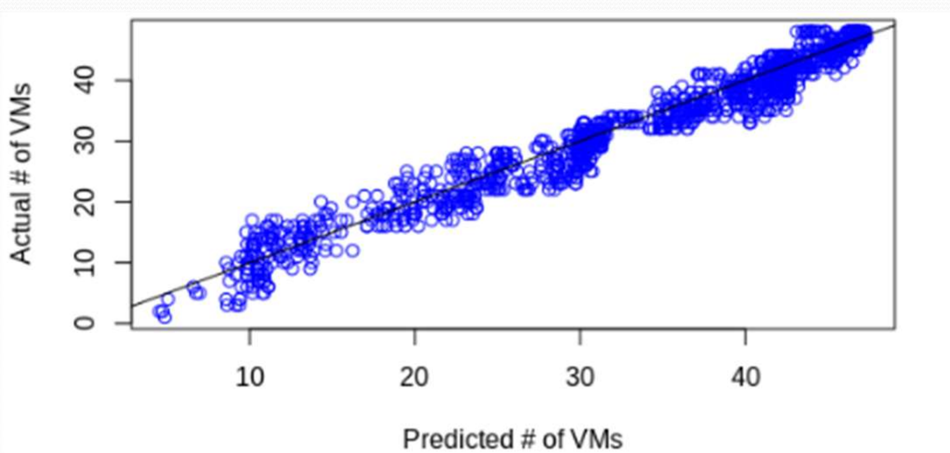
R^2	.9755	.9423
Root Mean Squared Error (RMSE)	2.479	2.175
Mean Absolure Error (MAE)	1.950	1.608
Min Prediction	4.537	-0.480
Max Prediction	47.479	49.543

Lower Predictive Error

April 22, 2020 IC2E 2020: *Characterizing Public Cloud Resource Contention to Support Virtual Machine Co-residency Prediction* 35

Observed vs. predicted co-located VMs: Random Forest Model

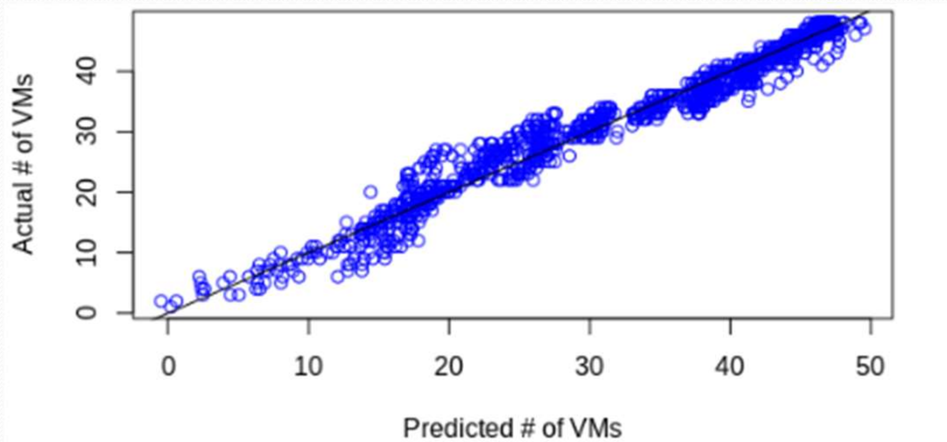
Line represents actual # of VMs



April 22, 2020 IC2E 2020: *Characterizing Public Cloud Resource Contention to Support Virtual Machine Co-residency Prediction* 36

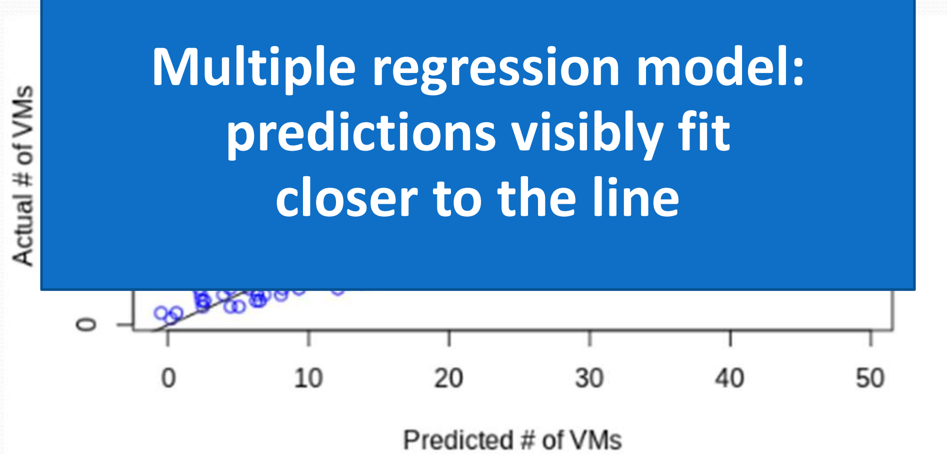
Observed vs. predicted co-located VMs: Multiple Regression Model

Line represents actual # of VMs



Observed vs. predicted co-located VMs: Multiple Regression Model

Line represents actual # of VMs

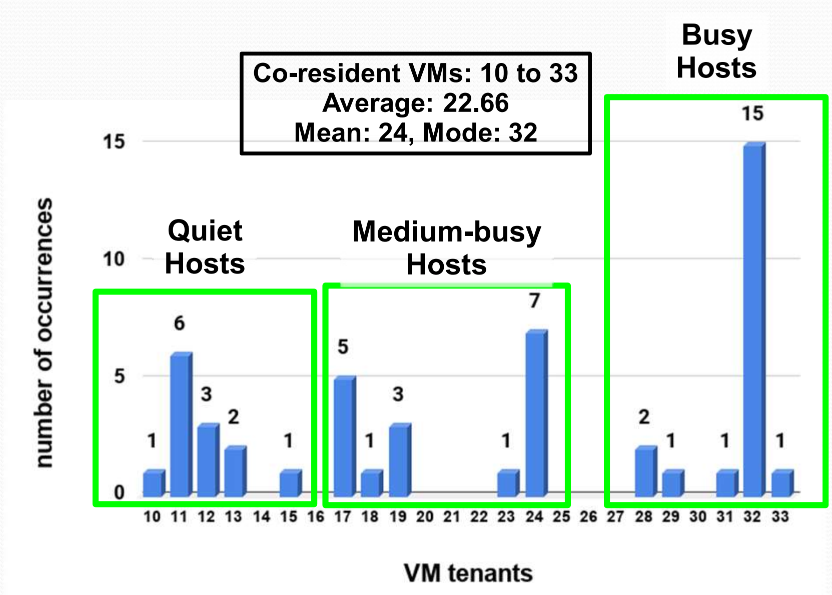


Multiple regression model: predictions visibly fit closer to the line

VM Co-residency Public Cloud Experiment

- Launched 100 x m5d.large 2 vCPUs VMs on the "open" public cloud (not dedicated host)
  - 50 VMs served as iPerf clients
- Ran benchmark suite to obtain test data
- Provided test data to trained multiple regression model
- Public cloud VMs co-reside with other user VMs that could run any workload
- These workloads exhibit CPU, disk, network contention

Public Cloud VM Co-residency Predictions



Public Cloud VM Co-residency Predictions

Busy Hosts

15

While public cloud predictions cannot be directly evaluated, our approach can infer activity level of the host

10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33

VM tenants

Outline

- Background
- Research Questions
- Benchmarking Suite
- Testing Infrastructure
- Experiments/Evaluation
- Conclusions

Conclusions

- **RQ-1 Resource Contention**:
  HW w/ high CPU core density produced greatest performance Δ for benchmarks
- **m5d.large:**
  y-cruncher (48%), pgbench (33%), sysbench (20.8%), iPerf (94.6%).
- Performance Δ's increased across VM generations w/ VM density:
  c3 (42%) to c4 (56%) to z1d (104.9%) to m5d (196.4%)
- Shutting down idle Linux VMs increased benchmark performance
  - **y-cruncher** increased by 32.4% (m5d) and 4.5% (z1d) hosts
  - **Sysbench** increased by 20.8% (m5d) hosts
- Network performance suffered the most from VM co-residency averaging 60% across all VM generations

Conclusions

- **RQ-2 Feature Evaluation**:
  Pg-bench was the most important feature, followed by Y-cruncher
- **RQ-2 Prediction Accuracy**:
  Multiple regression provided slightly more accurate predictions than random forest on evaluation with dedicated hosts
- VM prediction accuracy: +/- 1.61 VMs
- Average busyness of public cloud equivalent to ~22.66 VMs running our benchmark suite concurrently

Questions

Email: wlloyd@uw.edu
Webpage: http://faculty.washington.edu/wlloyd/

Questions

Thank You !

Email: wlloyd@uw.edu
Webpage: http://faculty.washington.edu/wlloyd/