Autonomic Management of Cost, Performance, and Resource Uncertainty for Migration of Applications to Infrastructure-as-a-Service (IaaS) Clouds

Ph.D. Dissertation Defense

Wes J. Lloyd
October 27, 2014

Colorado State University, Fort Collins, Colorado USA

Outline

- **Introduction**
  - **Research goals**
  - Challenges
  - Research questions
  - Background
  - Research contributions
- Supporting Infrastructure
- Research Results
  - Performance Modeling for Component Composition
  - VM Placement to Reduce Resource Contention
  - Workload Cost Prediction Methodology
- Conclusions

(VM) Virtual Machine
(PM) Physical Machine

Research Goals

- **Support application migration:**
  VM component composition, dynamic scaling, infrastructure alternatives
  - **Maximize: application throughput**
    Requests per second
  - **Minimize: hosting costs, server occupancy**
    Number of VMs, CPU cores, memory, disk space, hosting costs
  - **Minimize response time**
    Average service execution time (sec/min)

Outline

- **Introduction**
  - Research goals
  - **Challenges**
  - Research questions
  - Background
  - Research contributions
- Supporting Infrastructure
- Research Results
  - Performance Modeling for Component Composition
  - VM Placement to Reduce Resource Contention
  - Workload Cost Prediction Methodology
- Conclusions

Research Challenges – WHERE

Where should infrastructure be provisioned?

Research Challenges – WHERE

Service Isolation

Component Composition

Provisioning Variation

Server Consolidation

Multi-tenancy

Overprovisioning

Resource Contention

Research Challenges - WHAT

What infrastructure should be provisioned?

Performance

Research Challenges - WHAT

Size
Vertical Scaling

Quantity
Horizontal Scaling

Amazon VM types
m1.large
m3.xlarge
m2.2xlarge
m1.small
c1.xlarge
m3.medium
m2.4xlarge
c1.medium
m1.xlarge
c3.large

Qualitative
Resource descriptions

Virtualization Overhead

Virtualization Hypervisors

Performance Heterogeneity

Research Challenges - WHEN

When should infrastructure be provisioned?

Research Challenges - WHEN

Hot Spot Detection | **VM Launch Latency**

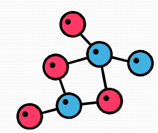
Future Load Prediction | **Pre-provisioning**

WHEN

Outline

- **Introduction**
  - Research goals
  - Challenges
  - **Research questions**
  - Background
  - Research contributions
- Supporting Infrastructure
- Research Results
  - Performance Modeling for Component Composition
  - VM Placement to Reduce Resource Contention
  - Workload Cost Prediction Methodology
- Conclusions

Research Questions (1/3)



DRQ-2: Performance modeling

What are the most important resource utilization variables and modeling techniques for predicting *service oriented application* (SOA) performance?

DRQ-3: Component composition

How does resource utilization and SOA performance vary relative to component composition across VMs?

Research Questions (2/3)

DRQ-4: VM placement implications

When dynamically scaling cloud infrastructure to address demand spikes how does VM placement impact SOA performance?

DRQ-5: Noisy neighbors

How can *noisy neighbors*, multi-tenant VMs that cause resource contention be detected? What performance implications result when ignoring them?

Research Questions (3/3)

DRQ-6: Infrastructure prediction

How effectively can we predict required infrastructure for SOA workload hosting by harnessing resource utilization models and Linux time accounting principles?

Outline

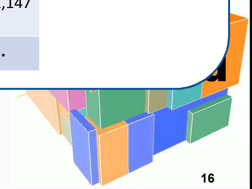
- **Introduction**
  - Research goals
  - Challenges
  - Research questions
  - **Background**
  - Research contributions
- Supporting Infrastructure
- Research Results
  - Performance Modeling for Component Composition
  - VM Placement to Reduce Resource Contention
  - Workload Cost Prediction Methodology
- Conclusions

Virtual Machine (VM) Placement as "Bin Packing Problem"

Bell's Number

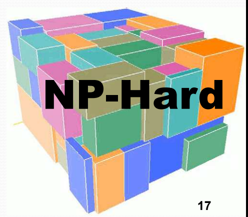
n	k
4	15
5	52
6	203
7	877
8	4,140
9	21,147
n	. . .

- VM resource utilization varies
- Component requirements vary



Virtual Machine (VM) Placement as "Bin Packing Problem"

- Components $_{items}$ → virtual machines (VMs) $_{bins}$
- Virtual machines (VMs) $_{items}$ → physical machines (PMs) $_{bins}$
- Dimensions
  - # CPU cores, CPU clock speed, architecture
  - RAM, hard disk size, # cores
  - Disk read/write throughput
  - Network read/write throughput
- PM capacities vary dynamically
- VM resource utilization varies
- Component requirements vary



Why Gaps Exist

- Public clouds
  - Research is time/cost prohibitive
  - Hardware abstraction: Users are not in control
  - Rapidly changing system implementations
- Private clouds: systems still evolving
- Performance models (large problem space)
- Virtualization misunderstood or overlooked

Outline

- **Introduction**
  - Research goals
  - Challenges
  - Research questions
  - Background
  - **Research contributions**
- Supporting Infrastructure
- Research Contributions
  - Performance Modeling for Component Composition
  - VM Placement to Reduce Resource Contention
  - Workload Cost Prediction Methodology
- Conclusions

Primary Research Contributions

- In the context of SOA migration to IaaS Clouds
  - Resource utilization modeling to predict component composition performance
  - VM placement improvement to reduce contention
    - Private IaaS: LeastBusy VM placement
    - Public/Private IaaS: Noisy-Neighbor Detection, Avoid heterogeneous VM type implementations
  - Workload cost prediction methodology for infrastructure alternatives to reduce hosting costs

Outline

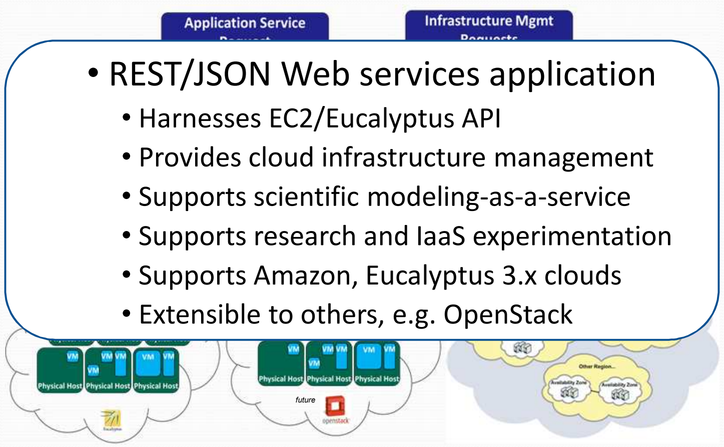
- Introduction
  - Research goals
  - Challenges
  - Research questions
  - Background
  - Research contributions
- **Supporting Infrastructure**
- Research Results
  - Performance Modeling for Component Composition
  - VM Placement to Reduce Resource Contention
  - Workload Cost Prediction Methodology
- Conclusions

Scientific Modeling Workloads

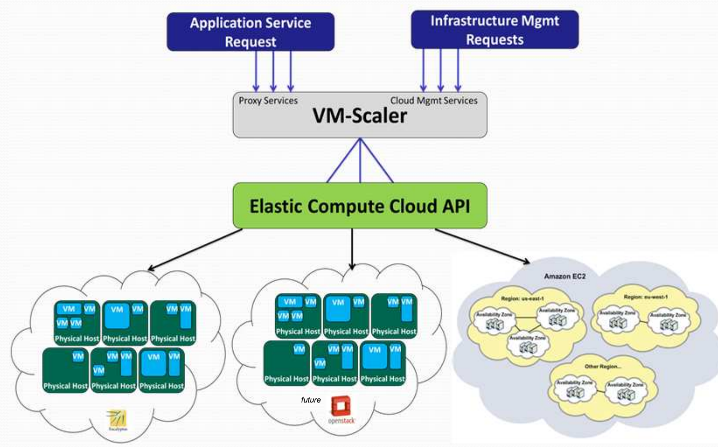
- CSIP: USDA platform for model services
- Service oriented application surrogates
  - RUSLE2 – Soil erosion model
  - WEPS – Wind Erosion Prediction System
  - SWAT-DEG: Stream channel degradation prediction
    Monte carlo workloads
  - Comprehensive Flow Analysis tools
    Load estimator, Load duration curve, Flow duration Curve, Baseflow, Flood analysis, Drought analysis

VM-Scaler

- REST/JSON Web services application
  - Harnesses EC2/Eucalyptus API
  - Provides cloud infrastructure management
  - Supports scientific modeling-as-a-service
  - Supports research and IaaS experimentation
  - Supports Amazon, Eucalyptus 3.x clouds
  - Extensible to others, e.g. OpenStack



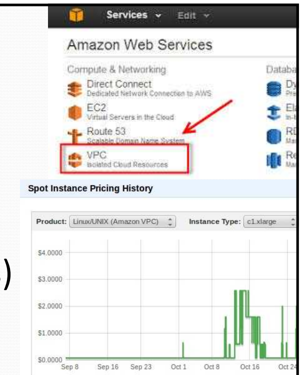
VM-Scaler



Eucalyptus 3.x Private Cloud

- Implemented (2) Private Clouds @ CSU
- Eramscloud: Oracle X6270 blade system
  - Dual Intel Xeon 4core HT 2.8 GHz CPUs
  - 24 GB ram, 146 GB 15k rpm HDDs
  - CentOS 5 & 6 x86_64 (host OS)
  - Ubuntu x86_64 (guest OS)
- Eucalyptus 3.x
  - Amazon EC2 API support
  - 8 Nodes (NC), 1 Cloud Controller (CLC, CC, SC)
  - Managed mode networking with private VLANs
  - XEN hypervisor version 3 & 4, paravirtualization

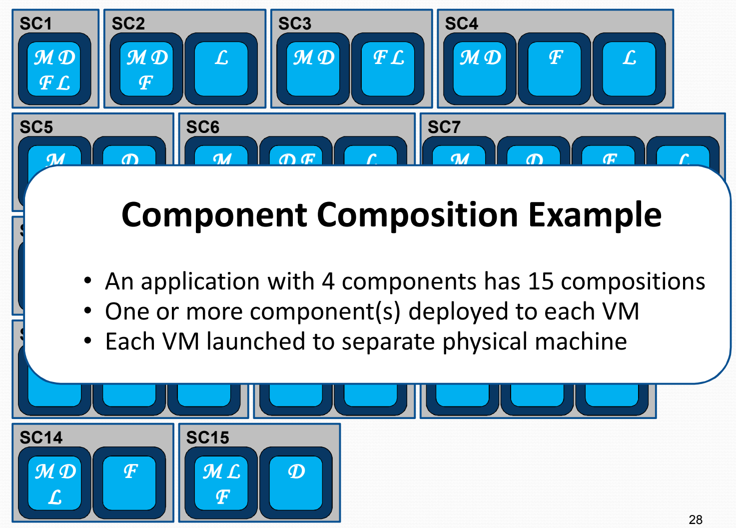
Amazon AWS



- Spot Instances
- Virtual Private Cloud (VPC)
- Ubuntu 9.10/12.04 (guests)
  - Xen virtualization
- 12 VM types, across 3 generations
  - 1st: m1.medium, m1.large, m1.xlarge, c1.medium, c1.xlarge
  - 2nd: m2.xlarge, m2.2xlarge, and m2.4xlarge
  - 3rd: c3.large, c3.xlarge c3.2xlarge, m3.large

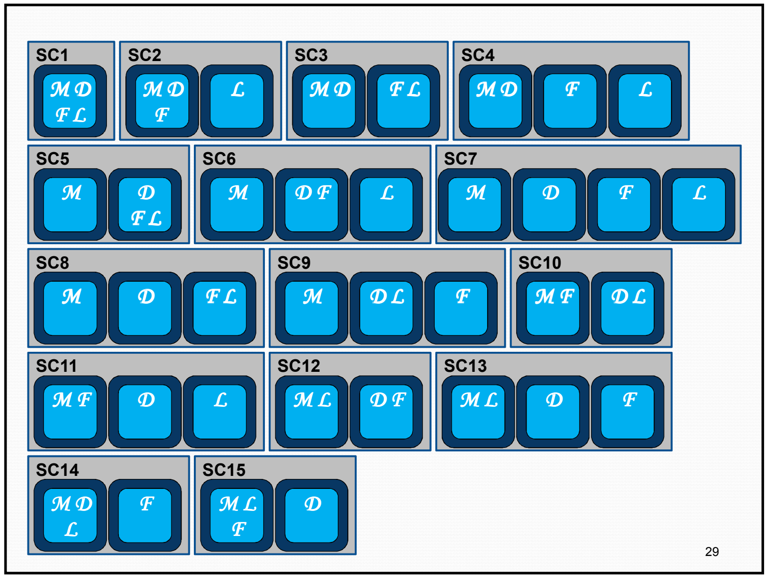
Outline

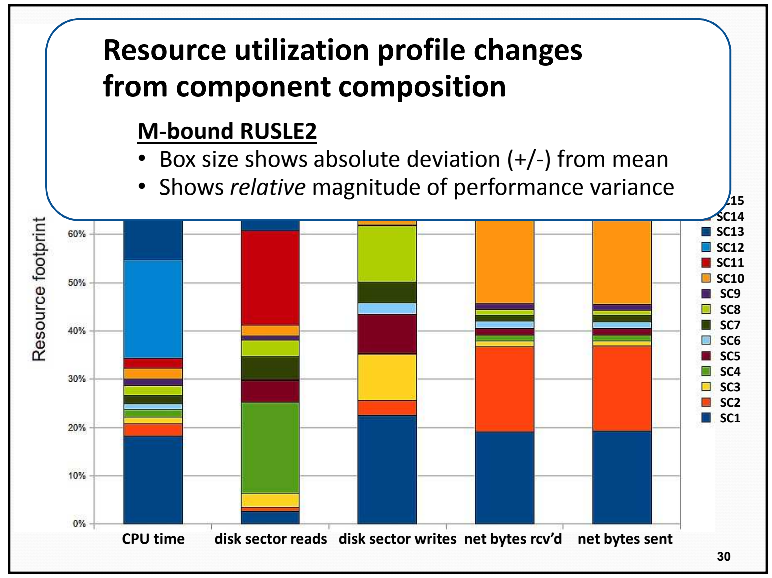
- Introduction
  - Research goals
  - Challenges
  - Research questions
  - Background
  - Research contributions
- Supporting Infrastructure
- **Research Results**
  - **Performance Modeling for Component Composition**
  - VM Placement to Reduce Resource Contention
  - Workload Cost Prediction Methodology
- Conclusions

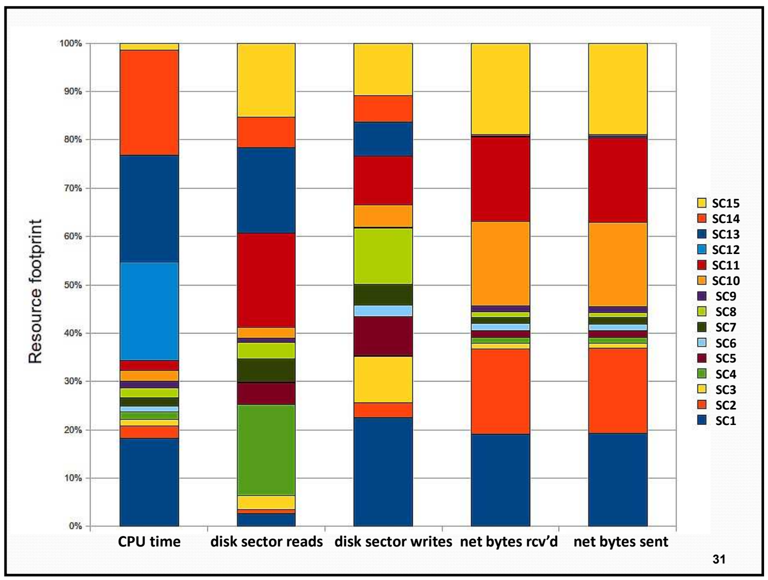


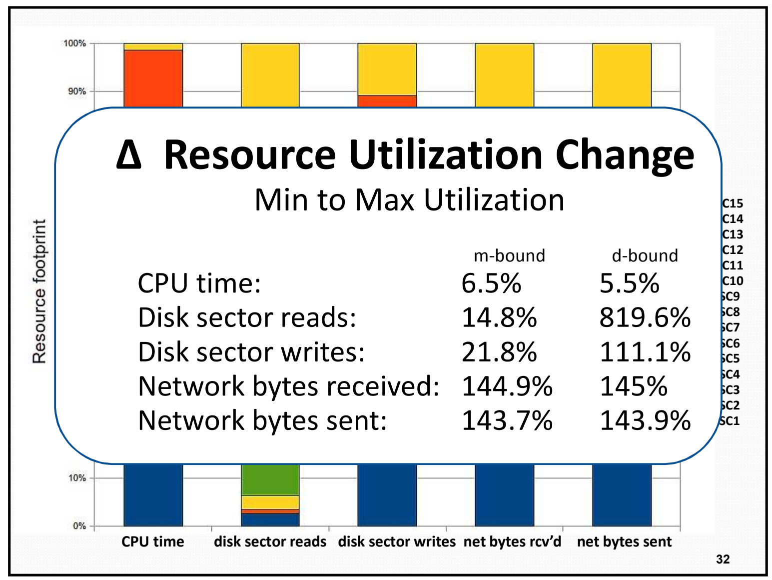
Component Composition Example

- An application with 4 components has 15 compositions
- One or more component(s) deployed to each VM
- Each VM launched to separate physical machine


SC1
M D F L
SC2
M D F
L
SC3
M D
F L
SC4
M D
F
L
SC5
M
D F L
SC6
M
D F
L
SC7
M
D
F
L
SC8
M
D
F L
SC9
M
D L
F
SC10
M F
D L
SC11
M F
D
L
SC12
M L
D F
SC13
M L
D
F
SC14
M D L
F
SC15
M L F
D
29


Resource utilization profile changes from component composition
M-bound RUSLE2
• Box size shows absolute deviation (+/-) from mean
• Shows *relative* magnitude of performance variance
Resource footprint
CPU time
disk sector reads
disk sector writes
net bytes rcv'd
net bytes sent
SC14
SC13
SC12
SC11
SC10
SC9
SC8
SC7
SC6
SC5
SC4
SC3
SC2
SC1
30


Resource footprint
CPU time
disk sector reads
disk sector writes
net bytes rcv'd
net bytes sent
SC15
SC14
SC13
SC12
SC11
SC10
SC9
SC8
SC7
SC6
SC5
SC4
SC3
SC2
SC1
31


Δ Resource Utilization Change
Min to Max Utilization
m-bound
d-bound
CPU time: 6.5% 5.5%
Disk sector reads: 14.8% 819.6%
Disk sector writes: 21.8% 111.1%
Network bytes received: 144.9% 145%
Network bytes sent: 143.7% 143.9%
Resource footprint
CPU time
disk sector reads
disk sector writes
net bytes rcv'd
net bytes sent
32

Resource Utilization Data Collection

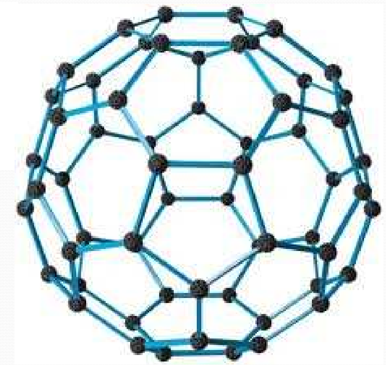
- Resource utilization sensors
  - Sensor on each VM/PM
  - Transmits data to VM-Scaler @ configurable intervals

CPU
- CPU time: (cpuUsr + cpuKrn)
- cpuUsr: CPU time in user mode
- cpuKrn:CPU time in kernel mode
- cpuIdle: CPU idle time
- contextsw: # of context switches
- cpuIoWait: CPU time waiting for I/O
- cpuIntSrvc: CPU time serving interrupts
- cpuSftIntSrvc: CPU time serving soft interrupts
- cpuNice: CPU time executing prioritized processes
- cpuSteal: CPU ticks lost to virtualized guests
- loadavg: (# proc / 60 secs)

Disk
- dsr: disk sector reads
- dsreads: disk sector reads completed
- drm: merged adjacent disk reads
- readtime: time spent reading from disk
- dsw: disk sector writes
- dswrites: disk sector writes completed
- dwm: merged adjacent disk writes
- writetime: time spent writing to disk

Network
- nbs: network bytes sent
- nbr: network bytes received

Can Resource Utilization Statistics Model Application Performance?

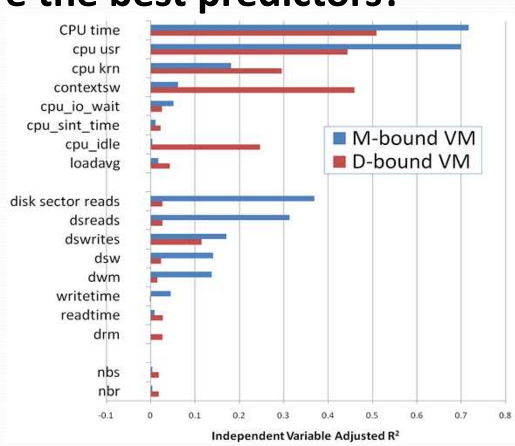


Which resource utilization variables are the best predictors?

CPU

Disk I/O

Network I/O



Which modeling techniques were most effective?

- Multiple Linear Regression (MLR)
- Stepwise Multiple Linear Regression (MLR-step)
- Multivariate Adaptive Regression Splines (MARS)
- Artificial Neural Network (ANNs)

Which modeling techniques were most effective?

Multiple Linear Regression

Stepwise MLR

Multivariate Adaptive Regression Splines

Artifical Neural Network

Model	Type	Adj. R^2	$RMSE_{train}$	$RMSE_{test}$
D-bound	MLR	0.9107	4532.85	44904
M-bound	MLR	0.8546	616.98	807.34
D-bound	MLR-step	0.9118	4589.27	43919
M-bound	MLR-step	0.8571	621.41	799.22
D-bound	MARS	0.918	4472.32	45137
M-bound	MARS	0.8718	596.45	825.34
D-bound	ANN	n/a	4440.03	44094
M-bound	ANN	n/a	595.49	800.71

Which modeling techniques were most effective?

Multiple Linear Regression

Stepwise MLR

Multivariate Adaptive Regression Splines

Artifical Neural Network

Data from each VM_{MDFL} combined to train models.

D-Bound RUSLE2

High $RMSE_{test}$ error (32% avg)

Which modeling techniques were most effective?

Multiple

Model performance did not vary much

Best vs. Worst

D-Bound		M-Bound
.11%	$RMSE_{train}$	.08%
.89%	$RMSE_{test}$	.08%
.40	rank err	.66

Outline

- Introduction
  - Research goals
  - Challenges
  - Research questions
  - Background
  - Research contributions
- Supporting Infrastructure
- **Research Results**
  - Performance Modeling for Component Composition
  - **VM Placement to Reduce Resource Contention**
  - Workload Cost Prediction Methodology
- Conclusions

Least-Busy VM Placement

- Busy-Metric
  - % of resource utilization vs. total capacity @ 1 second intervals
  - RU-sensors report VM Busy-Metric values every 15 secs
  - Units are (average RU/sec)
- PM aggregation
  - Sum VM Busy-Metric values
- Parameter weighting applied to particular RU variables
  - Supports prioritizing key resources for specific SOAs

Resource Utilization Data

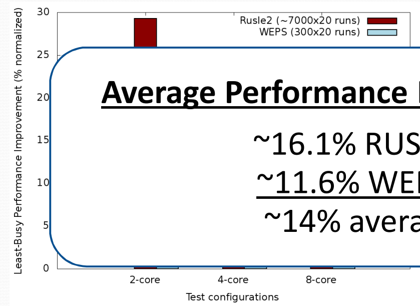
CPU
- Total CPU time
  weighted 2X

Disk
- Disk sector reads (DSR)
- Disk sector writes (DSW)

Network
- Network bytes sent (NBR)
- Network bytes received (NBS)

Virtualization
- Total VM count per host

Application Performance Improvement vs. Round-Robin VM Placement



Average Performance Improvement:

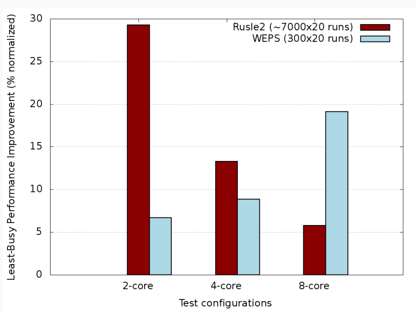
~16.1% RUSLE2
~11.6% WEPS
~14% average

VM size	RUSLE2	WEPS

Normalized % Performance Improvement

Statistical significance

Application Performance Improvement vs. Round-Robin VM Placement

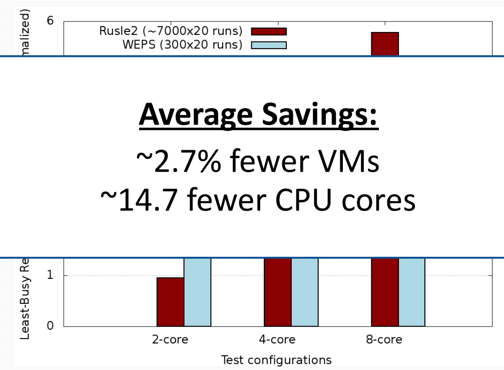


VM size	RUSLE2	WEPS
2-core VMs	*lb<rr* p=.014 df=18.2	*lb<rr* p=.162 *n.s.* [1]
4-core VMs	*lb<rr* p= 0.065 df = 22.7	*lb<rr* p= .035 df = 24.65
8-core VMs	*lb<rr* p=.017 df=24.5	*lb<rr* p=.00003 df=33.796

Normalized % Performance Improvement

Statistical significance

Resource Cost Savings vs. Round-Robin VM Placement



Average Savings:

~2.7% fewer VMs
~14.7 fewer CPU cores

Resource Cost Savings (% Fewer VMs)

CpuSteal Noisy Neighbor Detection Methodology (NN-Detect)

Noise Neighbor Thresholds

Application agnostic:
Minimum of 2x average *cpuSteal* for training workloads

Workload specific:
Select SOA workload which stresses the resource of concern (e.g. CPU-bound, disk-bound, network-bound)
Observe workloads to identify minimum cpuSteal thresholds for performance degradation

A Noisy Neighbor's *cpuSteal* exceeds **both** thresholds.

CpuSteal Noisy Neighbor Detection Methodology (NN-Detect)

- 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 worker VMs.
Step 2: Capture total *cpuSteal* for each worker VM for the workload.
Step 3: Calculate average *cpuSteal* for the workload ($cpuSteal_{avg}$).

Identify NNs using application agnostic and specific thresholds…

Amazon EC2 *CpuSteal* Analysis

VM type	Backing CPU	Average R^2 linear reg.	Average *cpuSteal* per core	% with Noisy Neighbors
us-east-1c				
c3.large-2c	E5-2680v2/10c	.1753	2.35	0%
m3.large-2c	E5-2670v2/10c	-	1.58	0%
m1.large-2c	E5-2650v0/8c	.5568	7.62	12%
m2.xlarge-2c	X5550/4c	.4490	310.25	18%
m1.xlarge-4c	E5-2651v2/12c	.9431	7.25	4%
m3.medium-1c	E5-2670v2/10c	.0646	17683.2[1]	n/a
c1.xlarge-8c	E5-2651v2/12c	.3658	1.86	0%
us-east-1d				
m1.medium-1c	E5-2650v0/8c	.4545	6.2	10%
m2.xlarge-2c	E5-2665v0/8c	.0911	3.14	0%

Amazon EC2 *CpuSteal* Analysis

Key Result #1

4 VM types had $R^2 > 0.44$

m1.large, m2.xlarge, m1.xlarge, m1.medium

Key Result #2

Where *cpuSteal* could not be predicted it did not exist. This hardware tended to be CPU core dense. (e.g. 8, 10, or 12)

EC2 Noisy Neighbor Performance Degradation

VM type	Region	WEPS	RUSLE2
m1.large E5-2650v0/8c	us-east-1c	117.68% df=9.866 $p=6.847\cdot10^{-8}$	125.42% df=9.003 p=.016
m2.xlarge X5550/4c	us-east-1c	107.3% df=19.159 p=.05232	102.76% df=25.34 $p=1.73\cdot10^{-11}$
c1.xlarge E5-2651v2/12c	us-east-1c	100.73% df=9.54 p=.1456	102.91% n.s.
m1.medium E5-2650v0/8c	us-east-1d	111.6% df=13.459 $p=6.25\cdot10^{-8}$	104.32% df=9.196 $p=1.173\cdot10^{-5}$

EC2 Noisy Neighbor Performance Degradation

Key Result #1

Maximum performance loss:

WEPS 18%, RUSLE2 25%

Key Result #2

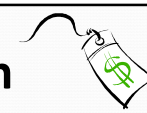
3 VM types with significant performance loss (p <.05)

Average performance loss: WEPS/RUSLE2 ~ 9%

Outline

- Introduction
  - Research goals
  - Challenges
  - Research questions
  - Background
  - Research contributions
- Supporting Infrastructure
- **Research Results**
  - Performance Modeling for Component Composition
  - VM Placement to Reduce Resource Contention
  - **Workload Cost Prediction Methodology**
- Conclusions

Workload Cost Prediction



Example:
Base VM-type: [`5 x c3.xlarge`] = 20 cores

- Scale the number of worker VMs
- Achieve ***equivalent*** performance using any VM type
- Load balance workload across VM pool

`c3.xlarge → c1.medium`
`c3.xlarge → m1.large`
`c3.xlarge → m2.4xlarge`
`c3.xlarge → m2.2xlarge`
`c3.xlarge → m2.xlarge`
`c3.xlarge → m1.xlarge`
`c3.xlarge → m1.medium`

Workload Cost Prediction

- Predict number of VMs of alternate type(s) supporting ***equivalent*** workload execution time
  - Execution within +/- 2 seconds using any base VM type
- Supports use of alternate VM types based on
  - Public cloud: lowest price VM-type
  - Private cloud: Most available or convenient VM-type
- Some VM types may be too slow to be viable

Approach

- Harness Linux CPU time accounting principles

Workload wall clock time can be calculated:

Sum CPU resource utilization variables across the worker VM pool, and divide by total CPU cores

$$\text{Workload}_{\text{time}} = \frac{cpuUsr_T + cpuKrn_T + cpuIdle_T + cpuIoWait_T + cpuIntSrvc_T + cpuSftIntSrvc_T + cpuNice_T + cpuSteal_T}{VM_{cores}}$$

Step 1: Train Resource Utilization Models

c3.xlarge → c1.medium

c3.xlarge → m1.large

c3.xlarge → m2.4xlarge

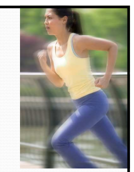
c3.xlarge → m2.2xlarge

c3.xlarge → m2.xlarge

c3.xlarge → m1.xlarge

c3.xlarge → m1.medium

Step 1: Train Resource Utilization Models



- Select representative SOA workloads
- Apples → Apples: Fix the # of CPU cores of worker VM pools
- Benchmark SOA workloads
  - Capture resource utilization profiles
- Train MLR-RU models
  - Models convert RU for different VM-types

VM-type Resource Variable Conversion Multiple Linear Regression

RU variable	adjusted R^2 m1.xlarge LR	adjusted R^2 m1.xlarge MLR	adjusted R^2 c1.medium MLR
cpuUsr	.9924	.9993	.9983
cpuKrn	.9464	.989	.9784
cpuIdle	.7103	.9674	.9498
cpuIoWait	.9205	.9584	.9725

Single Linear Regres. / **Multp Linear Regres.**

RU variable	adjusted R^2 m2.xlarge MLR	adjusted R^2 m3.xlarge MLR
cpuUsr	.9987	.9992
cpuKrn	.967	.9831
cpuIdle	.9235	.9554
cpuIoWait	.9472	.9831

Strong predictability forms the crux of the approach

Step 2: Profile workload resource utilization

- Perform single profiling run to capture resource utilization for a base VM-type (VM_{base} = 5 x c3.xlarge)

$$RU_{w(VM\text{-}base)} \leftarrow (W) \text{ on } n \text{ x } VM_{base}$$
$$n\text{=base \#VMs}$$

Step 3: Convert resource utilization profile

- Convert RU profile (Step 1) to alternate VM types

$$RU_{w(VM\text{-}base)} \rightarrow (M_{all}) \rightarrow RU_w\{n \text{ x } VM_{type1}, .. \; n \text{ x } VM_{type\text{-}j}\}$$
$$n\text{=base \#VMs, j=number VM types}$$

- Example types: {5 x m1.xlarge, 10 x c1.medium, 10 x m2.xlarge, 5 x m3.xlarge}

Step 4: Scale resource utilization profile

- “Virtually” scale up the # of worker VMs Calculate # of VMs required to “fit” workload execution into available wall clock time.
- Application agnostic, application aware heuristics

VMs / cores	wall time-goal	available clock ticks	*cpuUsr*	*cpuKrn*	*cpuIdle*
5 / 20	94.076s	188152	221502	10231	-43581
6 / 24	94.076s	225782	222533	10231	-6982
7 / 28	94.076s	263412	223565	10231	29616
8 / 32	94.076s	301043	224597	10231	66215
9 / 36	94.076s	338673	225629	10231	102813
10 /40	94.076s	376304	226661	10231	139412

Must Scale

Step 5: Select resource utilization profile

- Must select RU profile with sufficient *cpuIdle* time
  - Convert base type *cpuIdle* time, then scale value
  - Application agnostic, application aware heuristics
  - Too low *cpuIdle* suggests not enough wall clock time

VMs / cores	wall time-goal	available clock ticks	*cpuUsr*	*cpuKrn*	*cpuIdle*	
5 / 20	94.076s	188152	221502	10231	-43581	
6 / 24	94.076s	225782	222533	10231	-6982	← Clearly not enough
7 / 28	94.076s	263412	223565	10231	29616	← Possibly not enough
8 / 32	94.076s	301043	224597	10231	66215	← Too much?
9 / 36	94.076s	338673	225629	10231	102813	
10 /40	94.076s	376304	226661	10231	139412	

Step 6: Select VM type to minimize cost

- Resource scaling and profile selection heuristics allow determination of the required # of VMs for different VM types for equivalent performance
- Cost calculation involves plugging in resource costs

VM type	CPU cores	ECUs/core	RAM	Disk	Cost/hr.
c3.xlarge	4	3.5	7.5 GB	2x40 GB SDD	30¢
m1.xlarge	4	2	15 GB	4x420 GB	48¢
c1.medium	2	2.5	1.7 GB	1x350 GB	14¢
m2.xlarge	2	3.25	17.1 GB	1x420 GB	41¢
m3.xlarge	4	3.25	15 GB	2x40 GB SSD	45¢

Multiply by # of VMs

VMs required for equivalent performance

Mean Absolute Error (# VMs)

SOA / VM-type	PS-1 (RS-1)	PS-2 (RS-1)	PS-1 (RS-2)	PS-2 (RS-2)
WEPS	.5	.5	.5	.5
RUSLE2	.25	0	.125	.125
SWATDEG-STOC	.75	.5	.5	.625
SWATDEG-DET	.25	.375	.125	.125
m1.xlarge	.375	.25	.25	.25
c1.medium	.875	.625	.5	.625
m2.xlarge	.25	.25	.25	.25
m3.xlarge	.25	.25	.25	.25
Average	***.4375***	***.34375***	***.3125***	***.34375***

Workload hosting cost prediction

SOA	m1.xlarge	c1.medium	m2.xlarge
WEPS	$3.84	$2.24	$2.46
RUSLE2	$3.84	$2.24	$2.46
SWATDEG-Stoc	n/a	$1.96	$2.46
SWATDEG-Det	$3.84	$2.52	$2.87
Total	***$11.52***	***$8.96***	***$10.25***

	m3.xlarge	Total error
WEPS	$2.70	-$.76
RUSLE2	$2.70	$0
SWATDEG-Stoc	$2.70	-$.86
SWATDEG-Det	$2.70	+$.13
Total	***$10.80***	***-$1.49 (3.59%)***

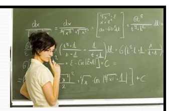
Outline

- Introduction
  - Research goals
  - Challenges
  - Research questions
  - Background
  - Research contributions
- Supporting Infrastructure
- Research Results
  - Performance Modeling for Component Composition
  - VM Placement to Reduce Resource Contention
  - Workload Cost Prediction Methodology
- **Conclusions**

Key Innovations

- Workload cost prediction methodology
  - Infrastructure alternatives to reduce costs
- Resource utilization performance modeling
  - Supports prediction of component compositions
- Noisy neighbor detection method
  - SOA performance improvement
- Least-Busy VM placement
  - Dynamic scaling improvement

Conclusions (1 of 3)



DRQ-2: Performance modeling

Best independent variables vary based on application profile characteristics.

CPU-bound applications : *cpuUsr*, *cpuKrn*, *dswrites.*

I/O-bound applications: *contextsw*, *dsr*, *dsreads*

DRQ-3: Component composition

Intuition is insufficient to determine best performant component compositions.

Magnitude of performance variance depends on application profile characteristics.

Performance variance of at least 15-25% is expected.

Conclusions (2 of 3)

DRQ-4: VM placement implications

Resource utilization spikes occur when launching VMs in parallel degrading application performance.

Careful VM placement reduces infrastructure requirements.

Least-Busy VM placement improves service execution time by 10-15%.

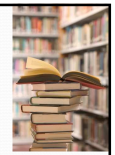
DRQ-5: Noisy neighbors

Analysis of *cpuSteal* supports detection of noisy neighbors.

Performance losses are reproducible for several hours.

Performance degradation from 10-25% is typical.

Conclusions (3 of 3)



DRQ-6: Infrastructure prediction

Workload Cost Prediction Methodology supports infrastructure and cost prediction while achieving equivalent performance

Infrastructure predictions: mean absolute error 0.3125 VMs

Infrastructure cost predictions ($): ~3.59% of actual.

Research Implications

- Infrastructure-as-a-service leads to the simplistic view that resource are homogeneous and scaling can infinitely provide linear performance gains
- Our results provide:

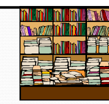
  Methodologies and algorithms to support application performance improvements while reducing infrastructure hosting costs

Do more with less!

Future Work

- White box resource utilization prediction
- Public cloud resource contention study
- Workload cost prediction methodology
  - Automated VM-scaler support
  - Predictive models to support resource scaling and profile selection
  - Workload cost prediction using mixed resources
  - Integration of spot market pricing models

Publications: Journal



1. W. Lloyd, S. Pallickara, O. David, J. Lyon, M. Arabi, and K. W. Rojas, "Performance implications of multi-tier application deployments on Infrastructure-as-a-Service clouds: Towards performance modeling," Future Generation Computer Systems, 2013.
2. O. David, J. C. Ascough II, W. Lloyd, T. R. Green, K. W. Rojas, G. H. Leavesley, and L. R. Ahuja, "A software engineering perspective on environmental modeling framework design: The Object Modeling System," Environ. Model. Softw., vol. 39, pp. 201–213, 2013.
3. W. Lloyd, S. Pallickara, O. David, M. Arabi, and K. W. Rojas, "Improving VM Placements to Mitigate Resource Contention and Heterogeneity in Cloud Settings for Scientific Modeling Services" submitted to the IEEE Transactions on Cloud Computing Journal, special issue:Scientific Cloud Computing (*under review*).
4. W. Lloyd, S. Pallickara, O. David, M. Arabi, and K. W. Rojas, "Demystifying the Clouds: Harnessing Resource Utilization Models for Cost Effective Infrastructure Alternatives" submitted to the IEEE Transactions on Cloud Computing Journal (*under review*).

Publications: Conference

1. W. Lloyd, S. Pallickara, O. David, J. Lyon, M. Arabi, and K. W. Rojas, “Migration of multi-tier applications to infrastructure-as-a-service clouds: An investigation using kernel-based virtual machines,” Proc. - 2011 12th IEEE/ACM Int. Conf. Grid Comput. Grid 2011, pp. 137–144, 2011.
2. W. Lloyd, O. David, J. Lyon, K. W. Rojas, J. C. Ascough II, T. R. Green, and J. Carlson, “The Cloud Services Innovation Platform - Enabling Service-Based Environmental Modeling Using IaaS Cloud Computing,” in Proceedings iEMSs 2012 International Congress on Environmental Modeling and Software, 2012, p. 8.
3. W. Lloyd, S. Pallickara, O. David, J. Lyon, M. Arabi, and K. W. Rojas, “Performance modeling to support multi-tier application deployment to infrastructure-as-a-service clouds,” in Proceedings - 2012 IEEE/ACM 5th International Conference on Utility and Cloud Computing, UCC 2012, 2012, pp. 73–80.
4. W. Lloyd, S. Pallickara, O. David, J. Lyon, M. Arabi, and K. W. Rojas, “Service isolation vs. consolidation: Implications for IaaS cloud application deployment,” in Proceedings of the IEEE International Conference on Cloud Engineering, IC2E 2013, 2013, pp. 21–30.
5. W. Lloyd, S. Pallickara, O. David, M. Arabi, and K. W. Rojas, “Dynamic Scaling for Service Oriented Applications: Implications of Virtual Machine Placement on IaaS Clouds,” in Proceedings of the 2014 IEEE International Conference on Cloud Engineering (IC2E ’14), 2014.
6. W. Lloyd, O. David, M. Arabi, J. C. Ascough II, T. R. Green, J. Carlson, and K. W. Rojas, “The Virtual Machine (VM) Scaler: An Infrastructure Manager Supporting Environmental Modeling on IaaS Clouds,” in Proceedings iEMSs 2014 International Congress on Environmental Modeling and Software, p. 8.
7. O. David, W. Lloyd, K. W. Rojas, M. Arabi, F. Geter, J. Carlson, G. H. Leavesley, J. C. Ascough II, and T. R. Green, “Model as a Service (MaaS) using the Cloud Service Innovation Platform (CSIP),” in Proceedings iEMSs 2014 International Congress on Environmental Modeling and Software, p. 8.
8. T. Wible, W. Lloyd, O. David, and M. Arabi, “Cyberinfrastructure for Scalable Access to Stream Flow Analysis,” in Proceedings iEMSs 2014 International Congress on Environmental Modeling and Software2, p. 6.

Questions