Resource Central: Understanding and Predicting Workloads for Improved Resource Management in Large Cloud Platforms

Marcus Fontoura

Eli Cortez, Anand Bonde, Alexandre Muzio, Thomas Moscibroda, Gabriel Magalhaes, Mark Russinovich, Ricardo Bianchini

Outline

Motivation

Container Scheduler

Characterization Azure VM Workload

Resource Central

Evaluation

Demo

Taxonomy

Conclusions

Machine learning everywhere

ML-based services:

Image recognition in Facebook Moments

Video analysis in YouTube captions

Correlation analysis in movie recommendations

…

ML techniques:

Regression

Classification

Reinforcement learning

…

We can leverage ML techniques to optimize the cloud platforms that run these services

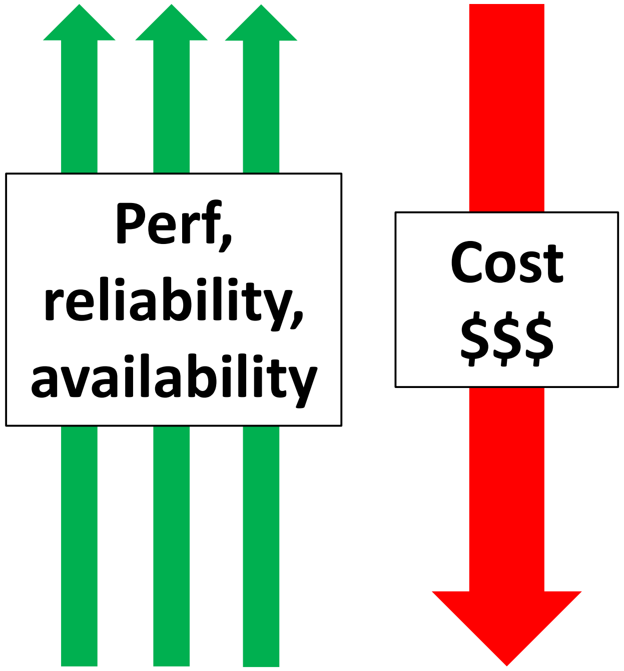
Microsoft

Public cloud platforms

Microsoft Azure

amazon web services

Google Cloud Platform



Lower Costs Via Resource Management

Microsoft

Pack VMs tightly

Oversubscribe resources

Increase server density

Reduce energy consumption

Reduce management overhead

Scavenge idle resources

Practical challenges:

- Complexity and scale
- VM performance impact
- VM availability impact

Lower Costs Via Resource Management

Pack VMs tightly

Oversubscribe resources

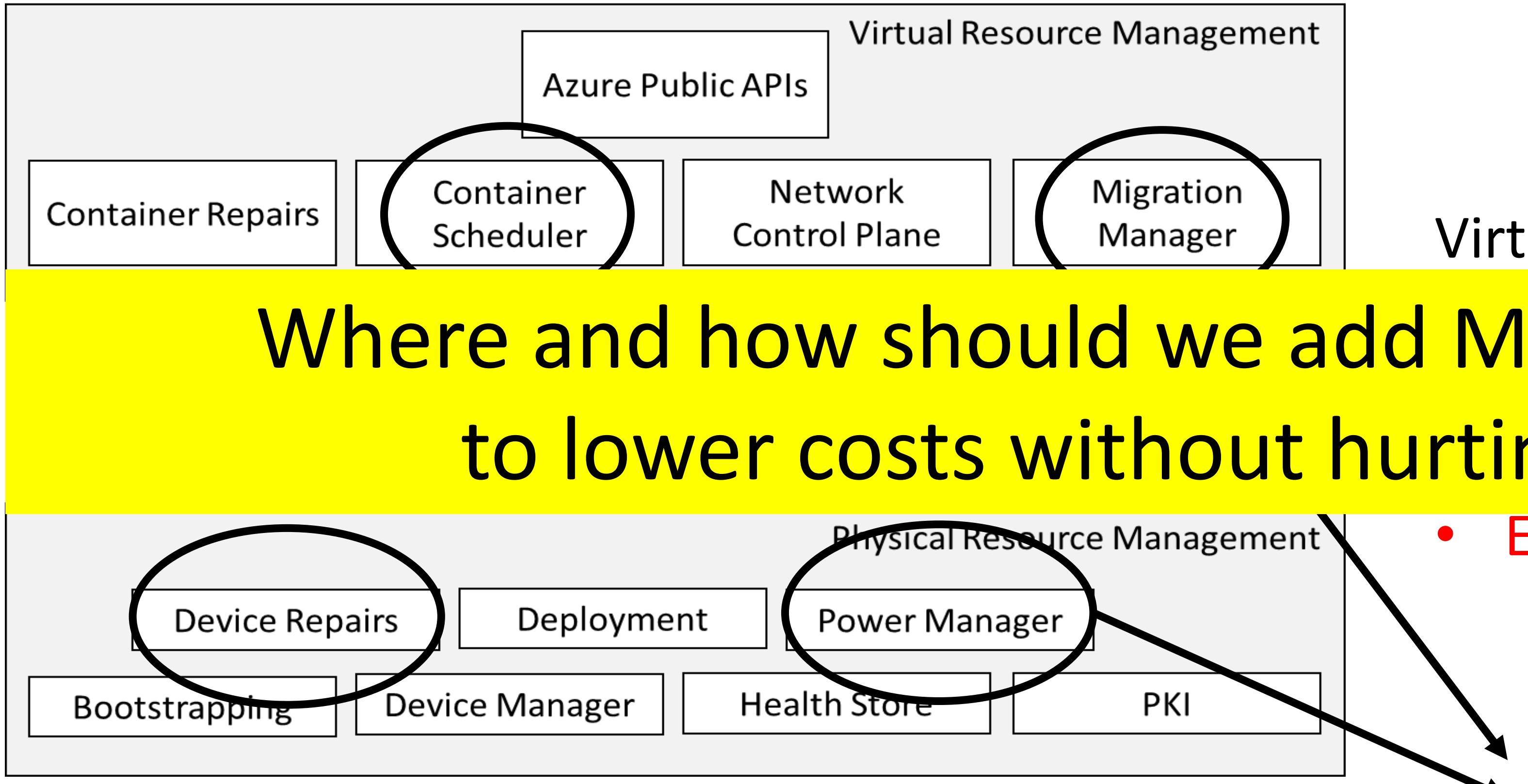
Practical challenges:

We can address these challenges by deeply <u>understanding</u> and <u>predicting</u> the characteristics of the VM workload!

Reduce management overhead

Scavenge idle resources

Overview of the Azure Compute platform



Virtual machine (VM) offerings:

Where and how should we add ML intelligence to lower costs without hurting QoS?

- Expensive to build and operate

Resource managers

Where? Many managers can benefit

Container scheduler

- Pack tightly [ASPLOS'13]
- Oversubscribe [Later, SOSP'17]
- Scavenge [OSDI'16]

Power manager

- Cap power
- Save energy [Google]

Migration manager

- Defragment servers
- Free up misbehaving servers

Practical challenges:

- Complexity and scale
- No info about apps
- Performance impact
- Availability impact

ML can help!

We need a general framework

Outline

Motivation

Container Scheduler

Characterization Azure VM Workload

Resource Central

Evaluation

Demo

Taxonomy

Conclusions

Virtual Machine Types

Azure has several VM families, for instance:

A: High-Value

Type	Cores	RAM
A0	1	0.768
A1	1	1.75
A2	2	3.5
A3	4	7
A4	8	14
A5	2	14
A6	4	28
A7	8	56
A8	8	56
A9	16	112
A10	8	56
A11	16	112

High Memory

Infiniband

Faster CPUs

D: Low-Latency, SSD

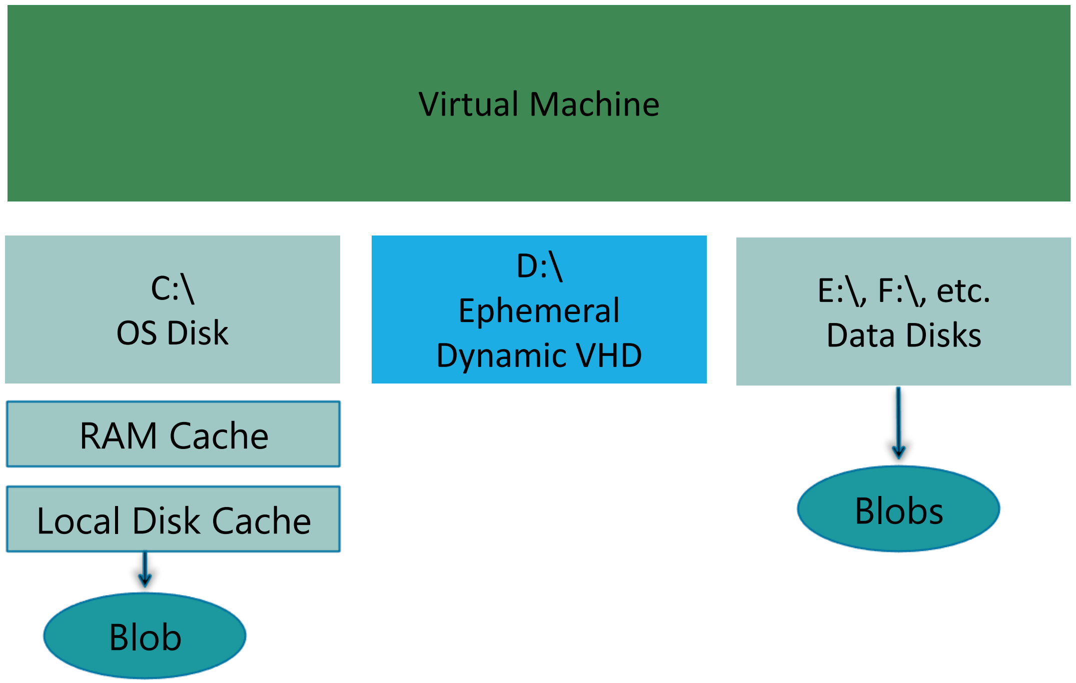
Type	Cores	RAM
D1	1	3.5
D2	2	7
D3	4	14
D4	8	28
D11	2	14
D12	4	28
D13	8	56
D14	16	112

G: Extreme Performance, SSD

Type	Cores	RAM
G1	2	28
G2	4	56
G3	8	112
G4	16	224
G5	32	448

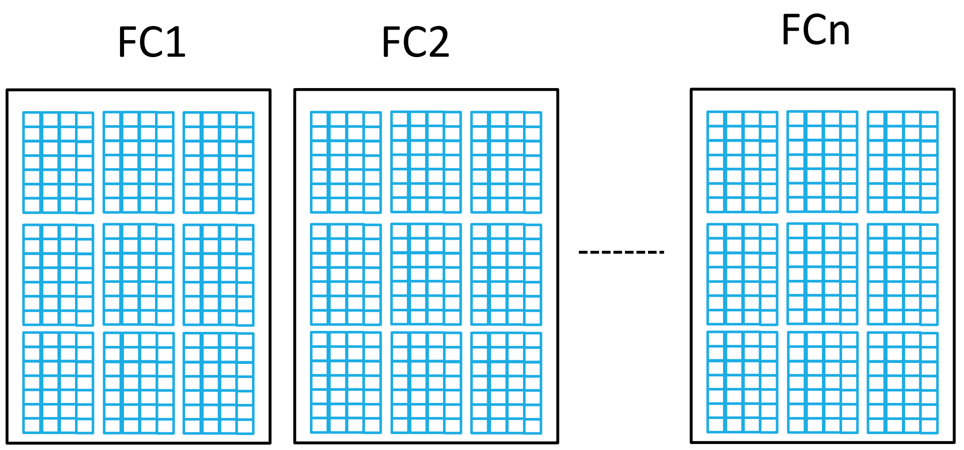
Virtual Machine Architecture

- Network, local and remote storage are additional allocation dimensions
- Ephemeral storage: uses local storage bandwidth and space
  - Backed by local HDD or SSD
- Persistent storage: uses network bandwidth
  - Cached on local server RAM, HDD or SSD
  - Backed by Azure Storage page blobs
  - “S” variants (e.g. “DS14”) can use SSD-backed Premium Storage



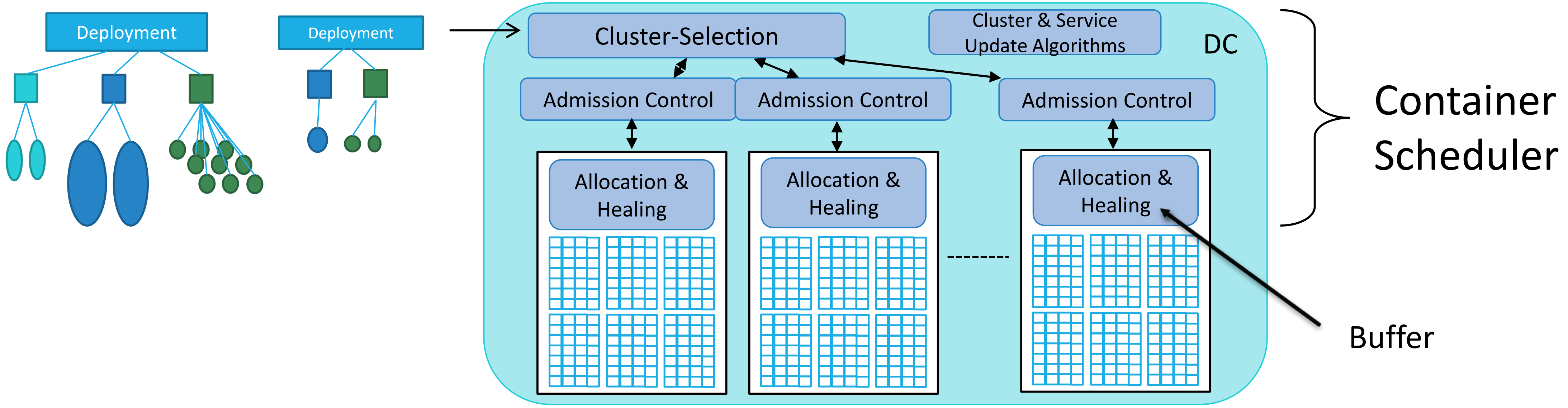
Fabric Clusters

- Fabric Controller: Hardware and VM manager for a “cluster” of servers
  - Uses 5-server Paxos-type replication for high availability
  - Exposes API for deploying, deleting and updating VMs
  - Keeps track of server and VM health
- Fabric Controller can autonomously “heal” a VM
  - Detects server has failed and restarts VM on a healthy server



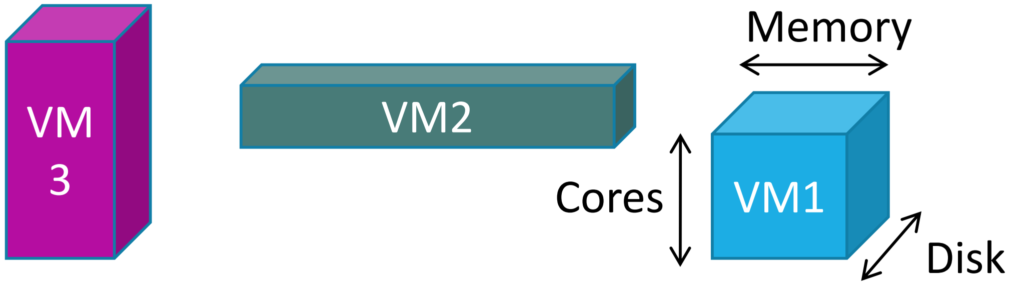
Container Scheduler

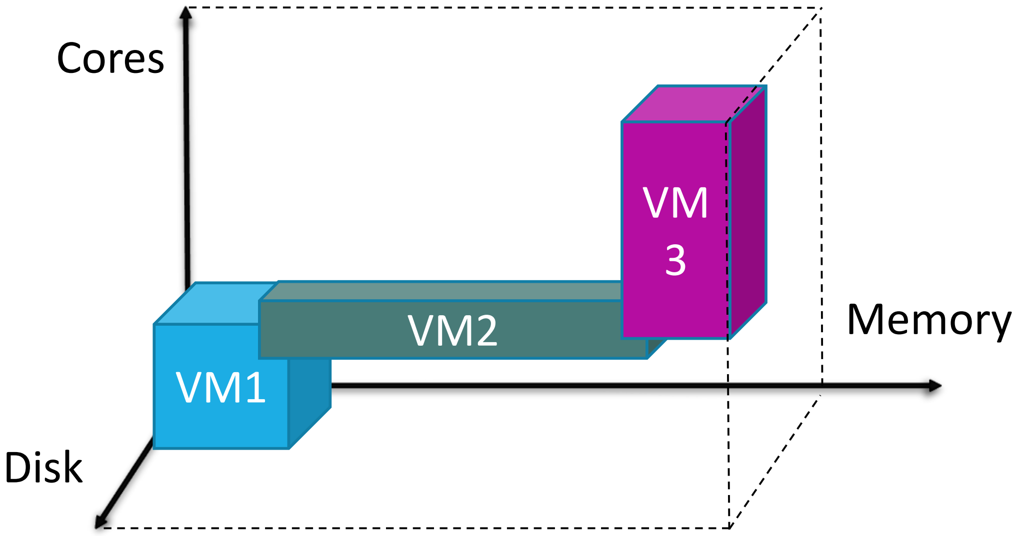
- Composed of cluster-selection, admission-control, and intra-cluster allocation algorithms
- Multi-level:
  - First, select FC cluster
  - Then, FC cluster allocator places VMs on servers



Constraints

- Placement constraints
  - Resource constraints: Sum of resources of all VMs on a node cannot exceed server resources (CPU, memory, disk, SSD, network IO,…)
    → Bin-Packing
  - Failure domain constraint: VMs of the same tenant must be spread across many failure domains
  - Co-location constraints: Certain types of VMs cannot be co-located together



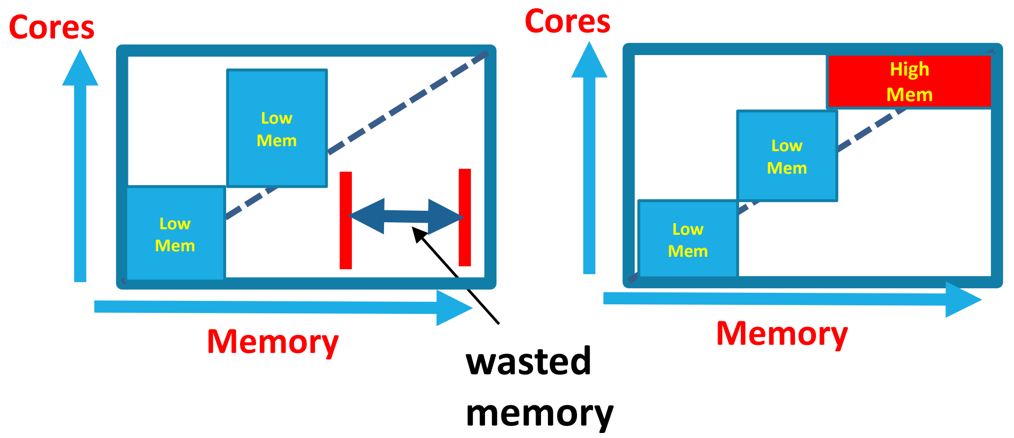


Resource Utilization

- VM Packing should achieve high utilization across all resource dimensions

  Multi-dimensional resource packing

Container scheduler should be aware of Multiple Resource Dimensions:

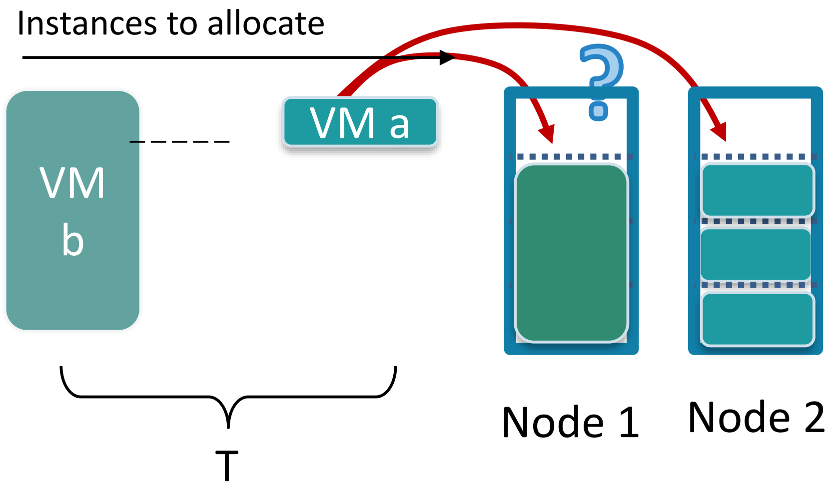


- We use **multi-dimensional best-fit**. [*Heuristics for Vector Bin Packing*, Panigrahy et al., MSR Tech Report 2011]
- Each resource dimension d is assigned a weight w_d → scarcity of the resource.
- r_d is the residual resource of a node
- Allocate the VM to the node that minimizes $\sum_d w_d * r_d$

Multi-Dimension Optimization

- Container scheduling should achieve high utilization across all resource dimensions
  1. Multi-dimensional resource packing
  2. Take into account online nature of service allocation

Container scheduler should be aware of online nature of allocation



- Simple example: Assume every VM has probability of ½ of leaving until time T.
- Probability that we can deploy VM_b ?
  - If new VM is placed on Node 1:

$$\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^3 = \frac{6}{16}$$

  - If new VM is placed on Node 2:

$$\left(\frac{1}{2}\right) + \left(\frac{1}{2}\right)^4 = \frac{9}{16}$$

→ Placing new VM on Node 2 is better !

Resource utilization in Azure

- Each 1% of utilization gain results in millions of $ savings

Container scheduling algorithms are crucial for operating Azure effectively!

Outline

Motivation

Container Scheduler

Characterization Azure VM Workload

Resource Central

Evaluation

Demo

Taxonomy

Conclusions

Background: Main Azure characteristics

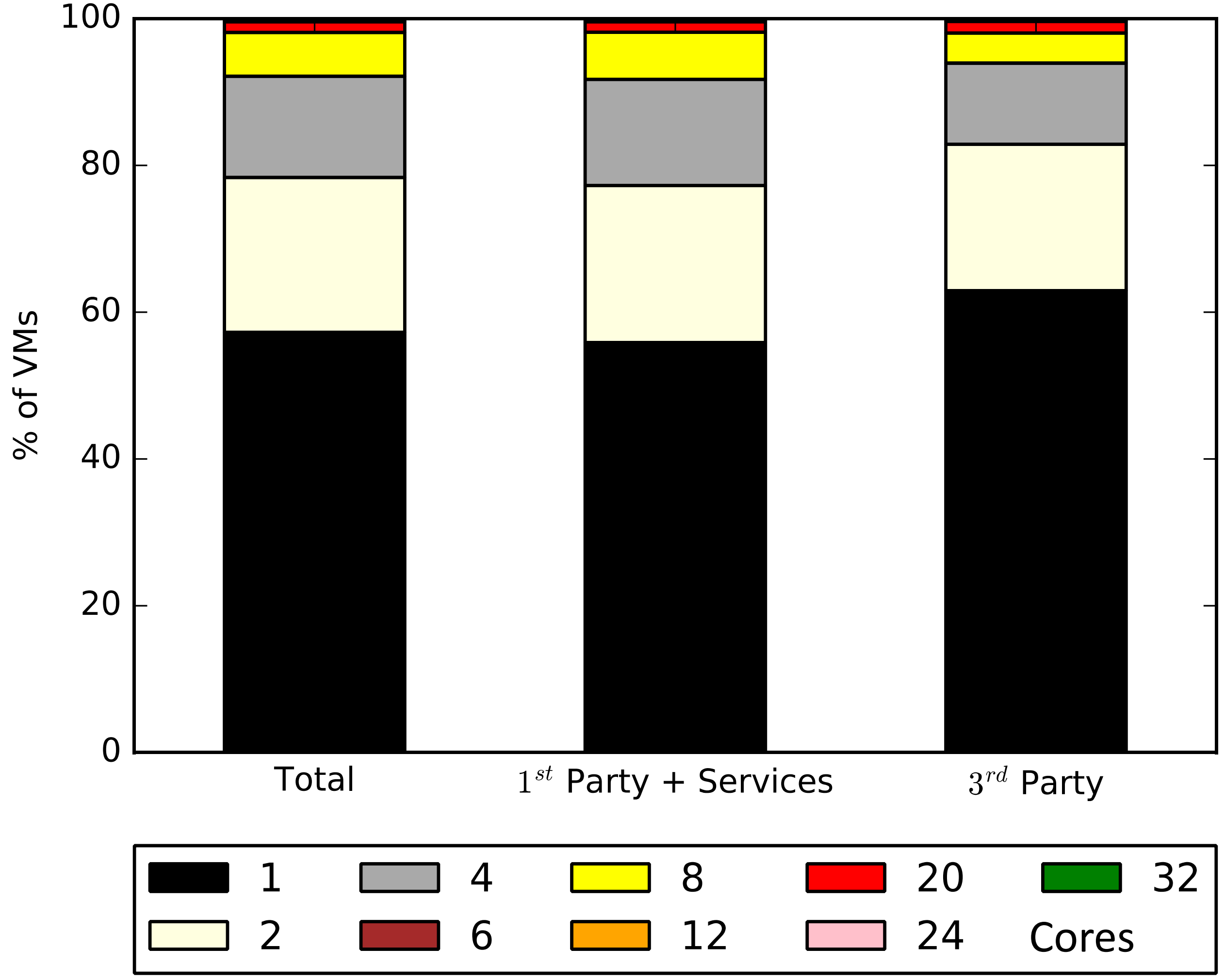
Azure hosts:

- 1st-party VMs – Microsoft dev, test, internal services
- 1st-party services offered to 3rd-party customers – Office 365, Xbox, Skype, …
- 3rd-party VMs – External users' VMs, Daimler, Geico, Adobe, …

Customers create "subscriptions", deploy VMs to regions in "deployments"

Our study: Full VM workload of Azure over 3 months (trace available!)

Characterization – VM size (CPU cores)



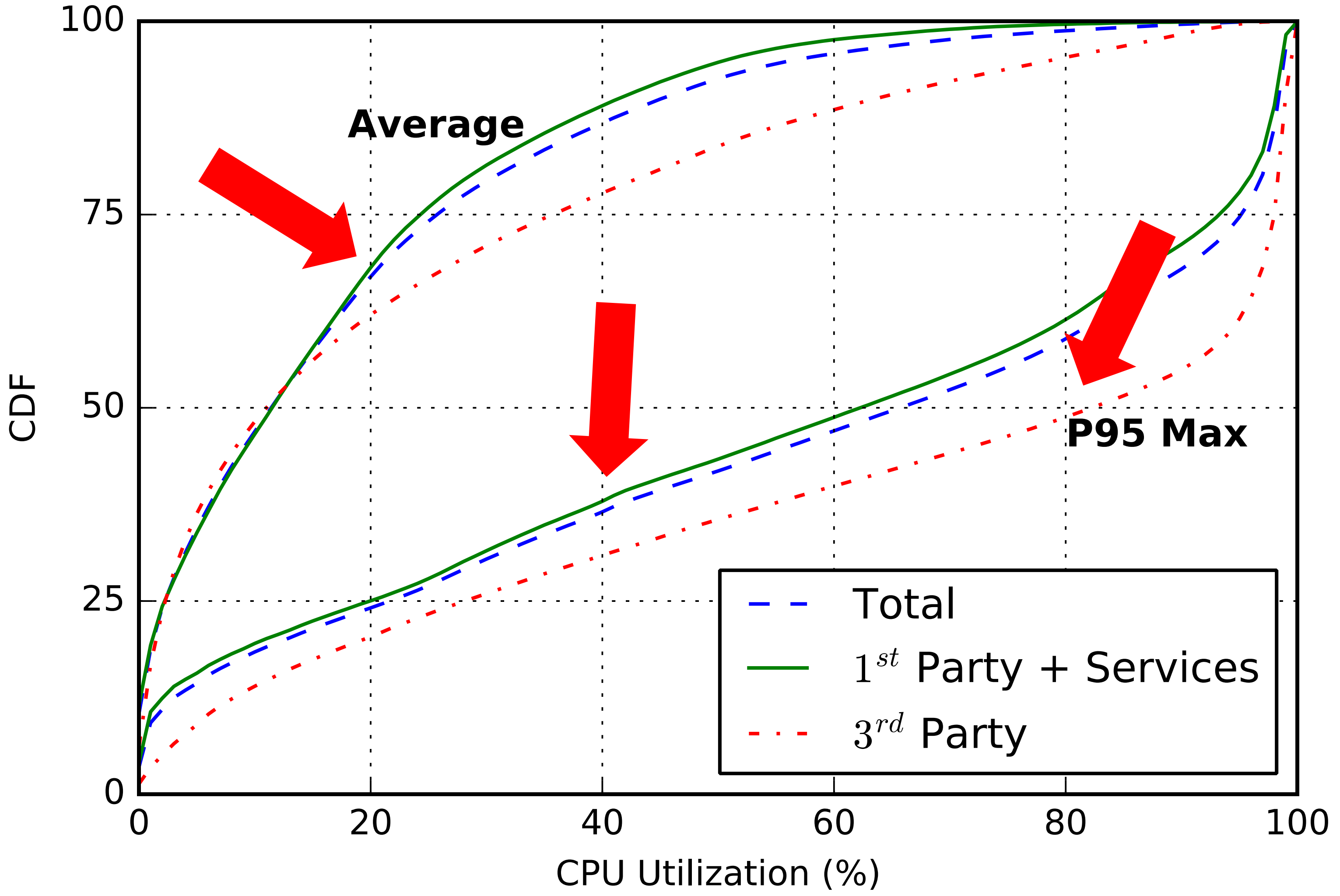
Observations:

- Small VMs with scale-out pattern
- CPU cores and memory are correlated
- 1st- and 3rd-party are similar

Resource management:

- Easier to fill holes

Characterization – VM CPU utilization



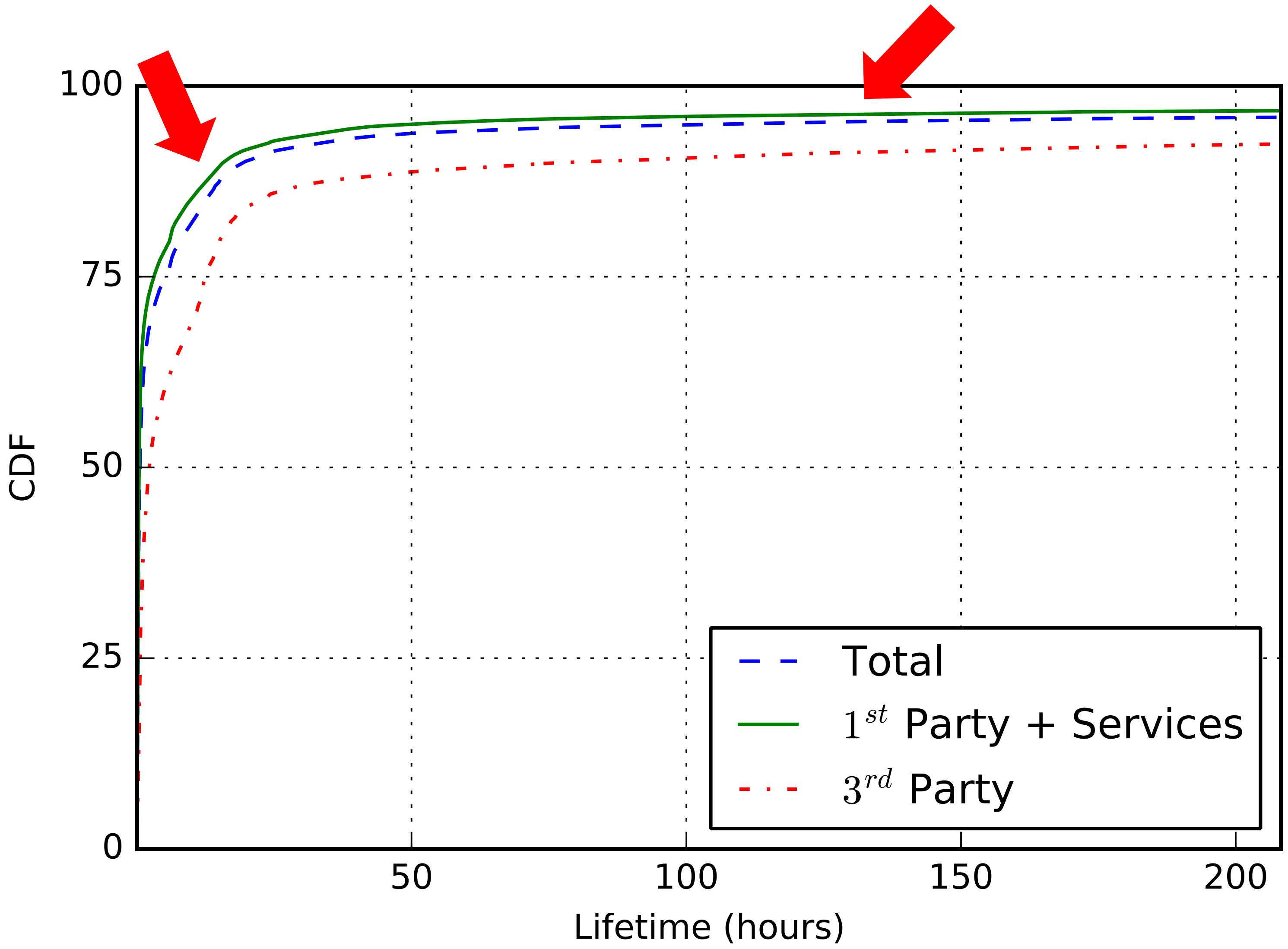
Observations:

- Large % with low avg. utilization
- Large % with high P95 util., esp. 3rd party
- Large % with low utilization even at P95

Resource management:

- High utils ➔ may limit packing
- Low utils ➔ oversubscription is possible

Characterization – VM lifetime



Observations:

- Short VMs dominate, esp. for 1st-party
- Non-trivial percentage of long VMs
- Long VMs = 95% of core hours!

Resource management:

- If VM lasts 1 day, it will live much longer
- Non-urgent maintenance
- Lifetime-aware VM scheduling

Other VM workload characteristics

VM type (IaaS vs PaaS)

VM memory size

VM deployment size

VM arrivals

VM workload class (interactive vs delay-insensitive)

Correlations between characteristics

Please refer to our paper for details

Outline

Motivation

Container Scheduler

Characterization Azure VM Workload

Resource Central

Evaluation

Demo

Taxonomy

Conclusions

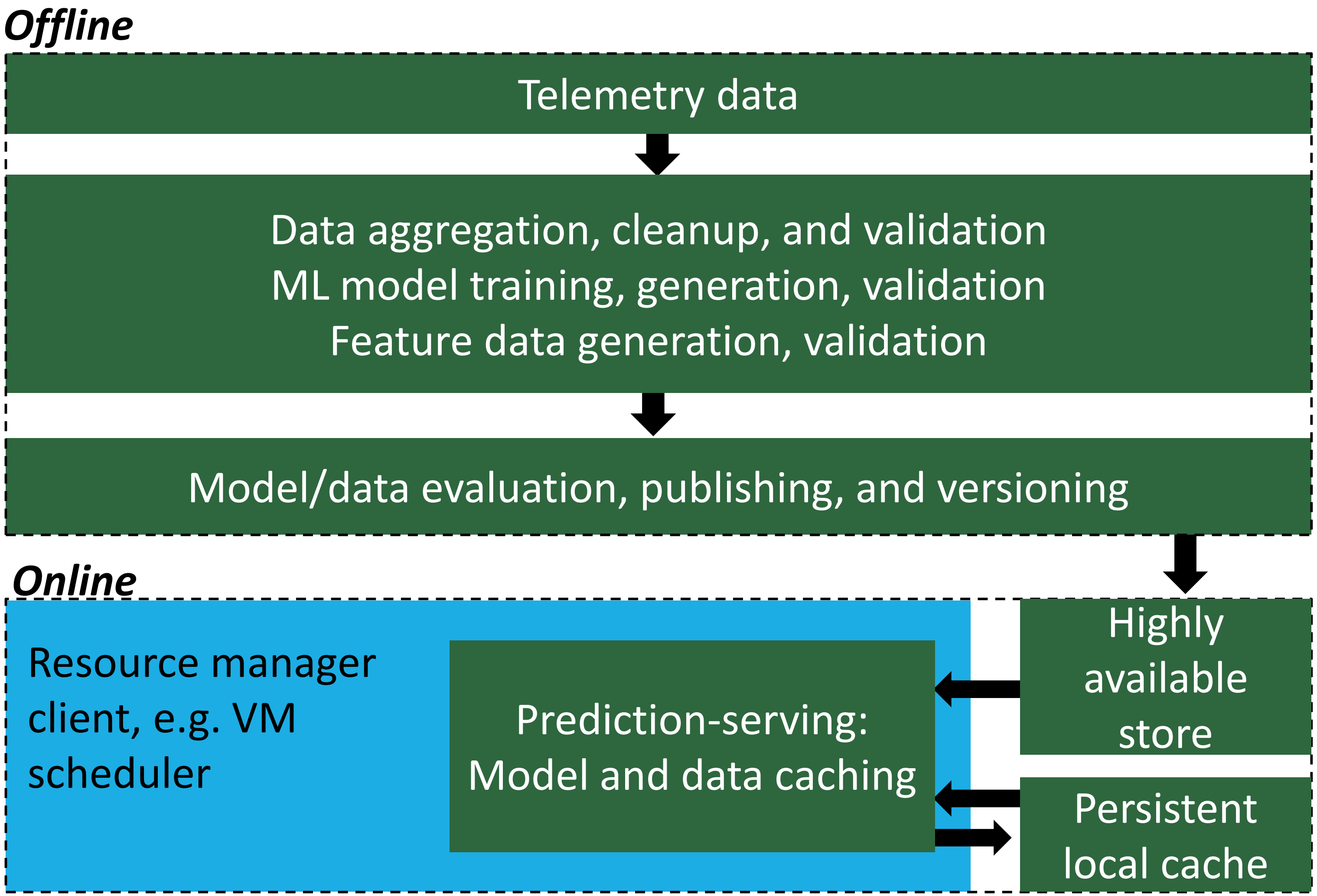
Resource Central

ML and prediction-serving system for improving resource management

Potential RC clients: Platform resource managers

- VM scheduling
- Cluster selection
- Power oversubscription
- Server maintenance
- VM rightsizing recommendation

Resource Central architecture



Design principles:

- Off critical perf & availability paths
- Simple; based on stable systems
- General; easy to use by clients

Status:

- Manually used by engineers
- Clients in production

Current ML models

Metrics	Modeling approaches
CPU utilization	Random Forests
Deployment size	Extreme Gradient Boosting Trees
Lifetime	Extreme Gradient Boosting Trees
Workload class	FFT, Extreme Gradient Boosting Tree

Classification algorithms

- Numeric models predict “buckets”
- Prediction comes with a “confidence score”

Outline

Motivation

Container Scheduler

Characterization Azure VM Workload

Resource Central

Evaluation

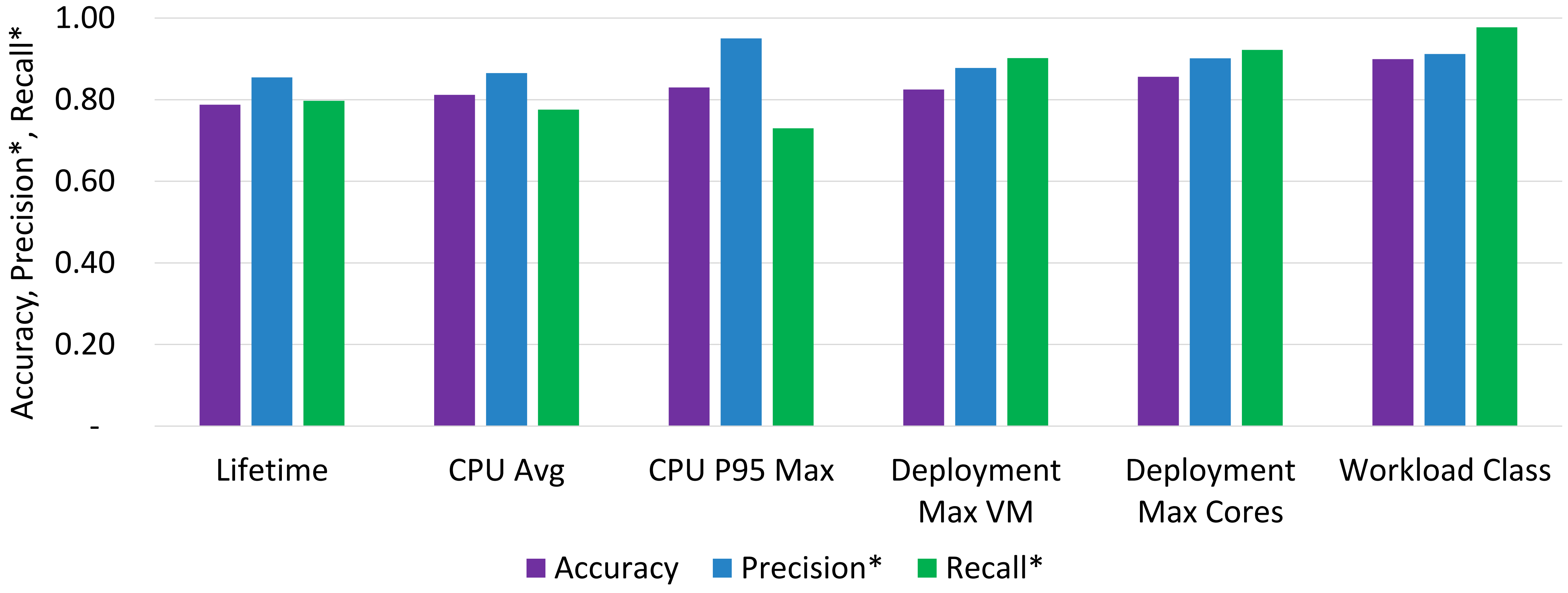
Demo

Taxonomy

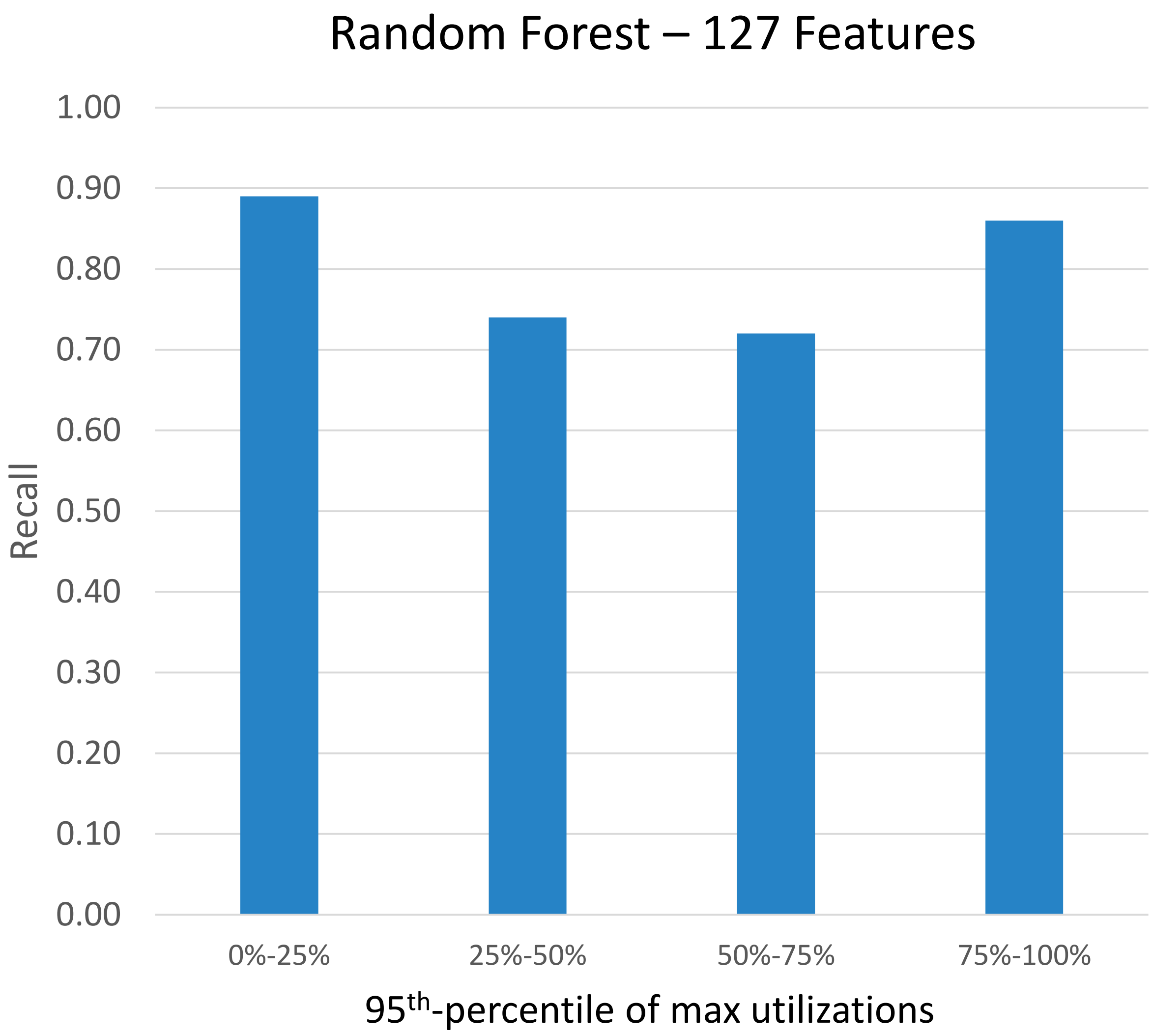
Conclusions

Prediction quality

Accuracy ≥ 79%
$Precision^{\theta}$ ≥ 85%
$Recall^{\theta}$ ≥ 73%



Prediction - VM CPU P95 max



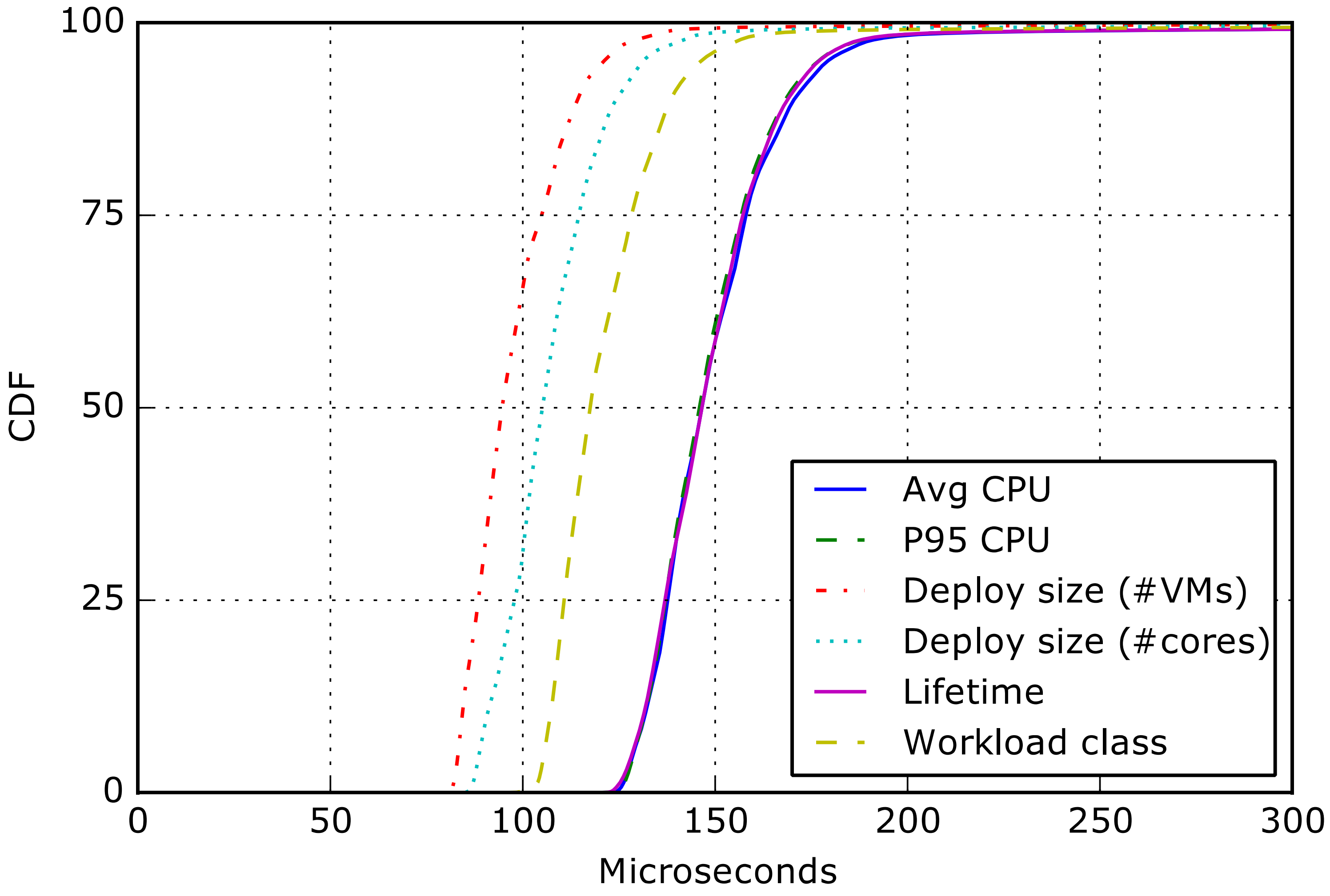
- Overall accuracy = 0.83
- $Precision^{\theta} = 0.94$
- $Recall^{\theta} = 0.73$

Important attributes:

- % previous VMs in bucket (subscription)
- Operating system

Deployment time is irrelevant

Performance – Model Execution



- Low latency
- Predictable
- 99th percentile: 258 µsec max

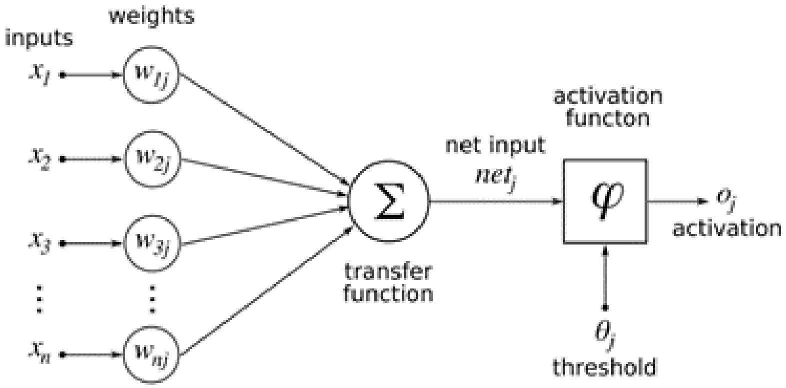
Deep Learning in RC

Azure Workload Information

Telemetry, Metadata

Azure Batch AI

Deep Learning Models



Microsoft

Deep Learning in RC

Task: VM Lifetime Prediction

Neural net

Inputs:
(~500 features)

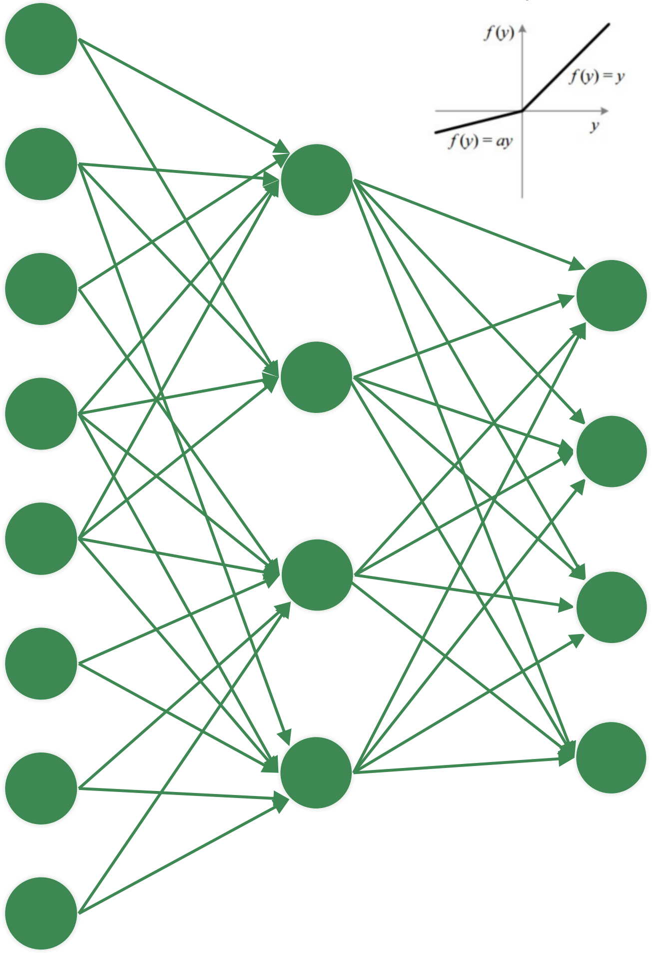
- VP Count
- Memory
- OS
- VM Type
- Subscription

(…)

Output (classification):

VM Lifetime (in 4 buckets)

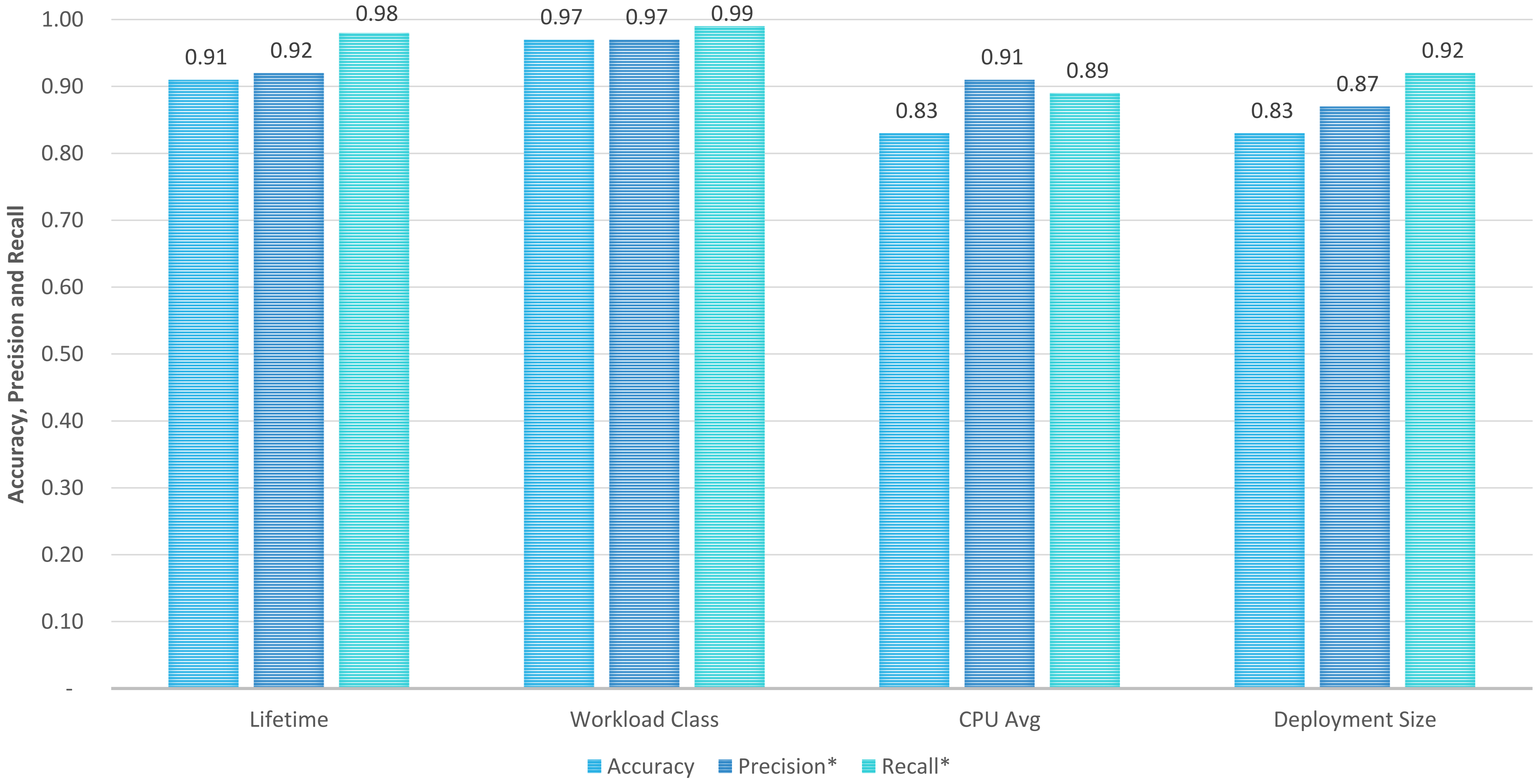
Activation Function:
LeakyRelu



Features

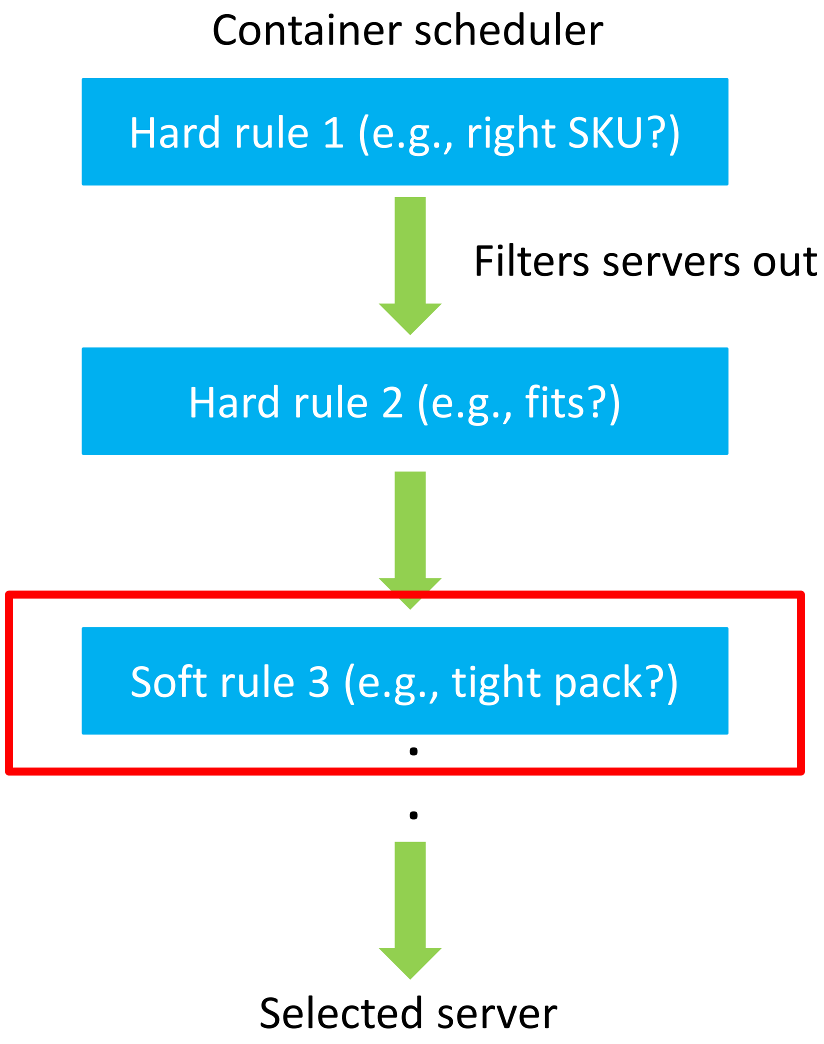
Prediction Quality

$Accuracy \geq 83\%$
$Precision^{\theta} \geq 87\%$
$Recall^{\theta} \geq 89\%$



Case study: Smart CPU oversubscription

Microsoft



Goals:

- Be conservative! Stick with P95, 1st-party loads
- Don't oversubscribe servers running prod VMs
- Oversubscribe other servers up to a percentage over capacity and a max predicted (P95) utilization

New rule checking the sum of the P95 utilizations

Mispredictions: only issue is consistent under-prediction

RC-informed CPU oversubscription

Simulation results

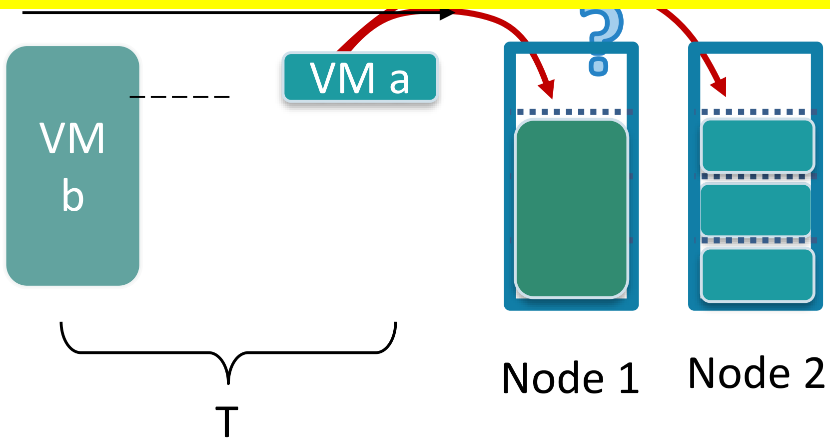
Version	Description	Behavior
Baseline	No oversubscription	Low capacity; many VM allocation failures
Naive	25% oversub without predictions	No failures; 6x resource exhaustion
RC-informed	25% oversub with RC predictions	No failures; rare exhaustion
RC-right	25% oversub with oracle predictions	No failures; same exhaustion

Multi-Dimension Optimization

- Container scheduling should achieve high utilization across all resource dimensions
  1. Multi-dimensional resource packing
  2. Take into account online nature of service allocation

- Simple example: **Assume every VM has**

Lifetime prediction is important for container scheduling



$$\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^3 = \frac{6}{16}$$

- If new VM is placed on Node 2:

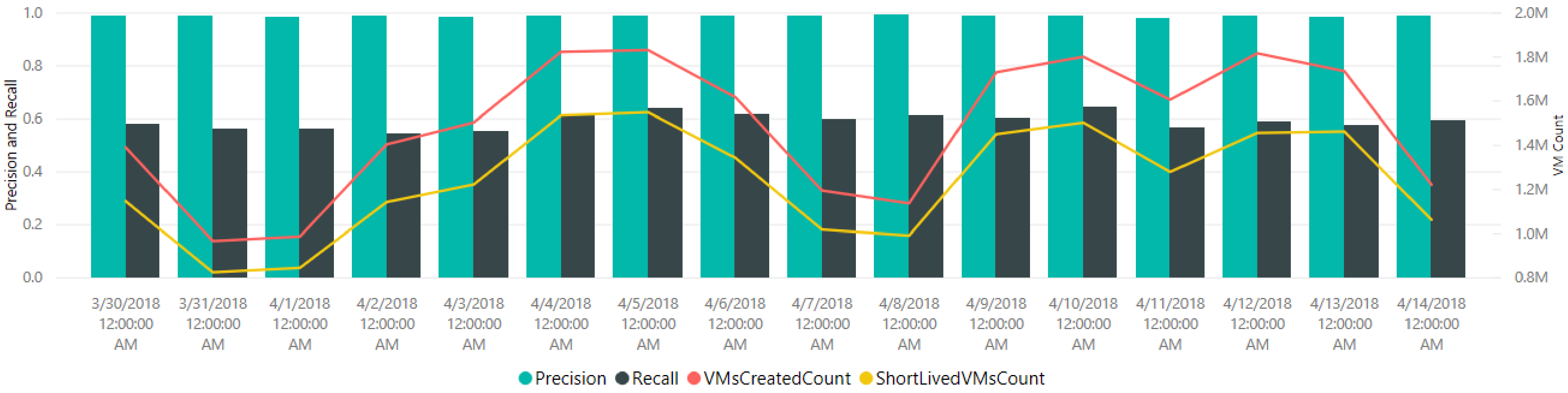
$$\left(\frac{1}{2}\right) + \left(\frac{1}{2}\right)^4 = \frac{9}{16}$$

→ Placing new VM on Node 2 is better !

Production Dashboard

Resource Central - Short Lived VM Prediction Quality in Production

Date	VMsCreatedCount	ShortLivedVMsCount	ShortLivedResourceCentralPredictedCount	ShortLivedAndPredictedCount	Precision	Recall
4/3/2018 12:00:00 AM	1503947	1222903	688535	680188	0.99	0.56
4/4/2018 12:00:00 AM	1823851	1536073	948810	941223	0.99	0.61
4/5/2018 12:00:00 AM	1832033	1549938	1002854	994740	0.99	0.64
4/6/2018 12:00:00 AM	1618960	1344647	838380	828991	0.99	0.62
4/7/2018 12:00:00 AM	1195448	1018937	616786	609763	0.99	0.60
4/8/2018 12:00:00 AM	1137267	991428	611711	607731	0.99	0.61
4/9/2018 12:00:00 AM	1730869	1451170	887340	880931	0.99	0.61
4/10/2018 12:00:00 AM	1801473	1503357	982545	972590	0.99	0.65
4/11/2018 12:00:00 AM	1606677	1280204	740178	728069	0.98	0.57
4/12/2018 12:00:00 AM	1817186	1457029	868355	860817	0.99	0.59
4/13/2018 12:00:00 AM	1736058	1463295	856922	845368	0.99	0.58
4/14/2018 12:00:00 AM	1221487	1062817	641981	634921	0.99	0.60



Demo

Outline

- Motivation
- Container Scheduler
- Characterization Azure VM Workload
- Resource Central
- Evaluation
- Demo
- **Taxonomy**
- Conclusions

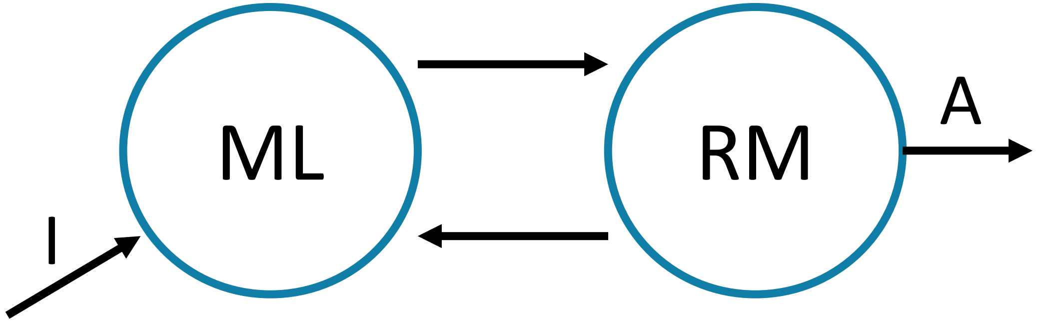
Approaches to adding ML

Passive, external to managers:

Predict load intensity, utilization

Cluster workloads, resources

ML as an insight provider



I = Inputs; A = Actions
RM = Resource Manager

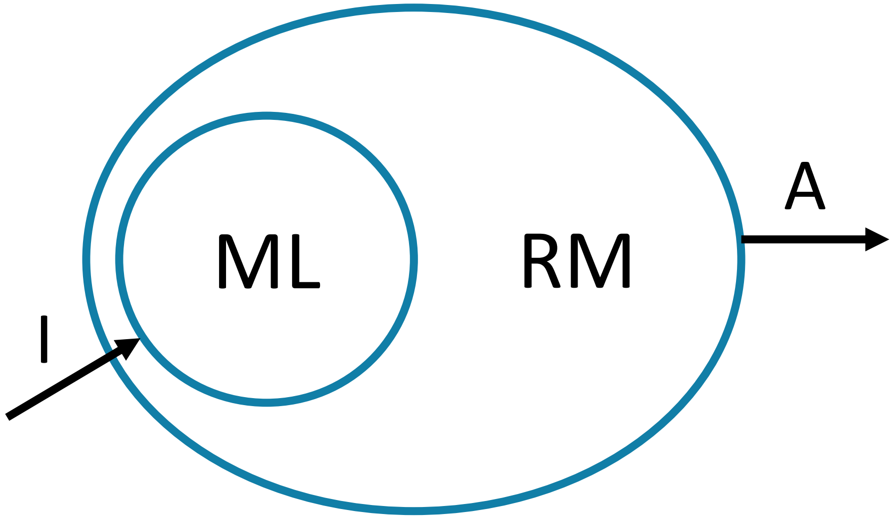
Debuggable; simpler RMs

Active, built into managers:

Adjust parameters of policies

Select actions to be performed

ML has deep knowledge of policies

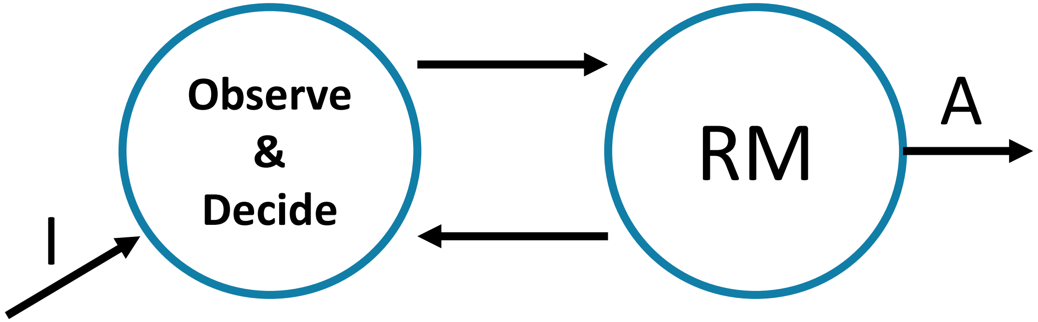


Along a different dimension

Iterative observe and decide:

After each action, observe & decide

Management as a control problem

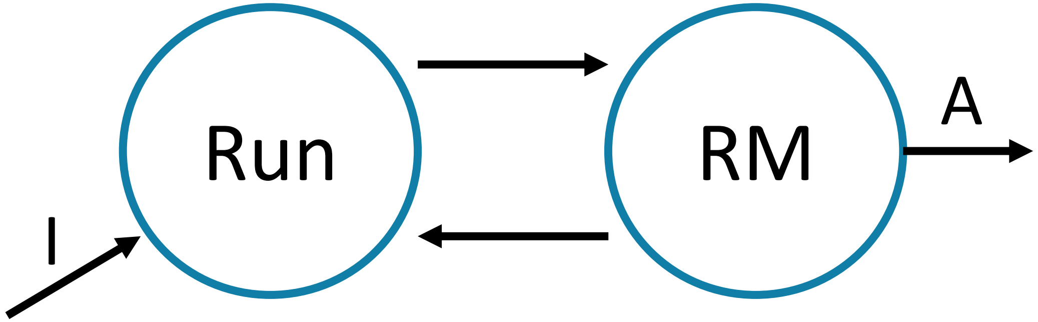


I = Inputs; A = Actions
RM = Resource Manager

Delayed observation:

Generate model offline, run it online

Re-generate model periodically



Simpler

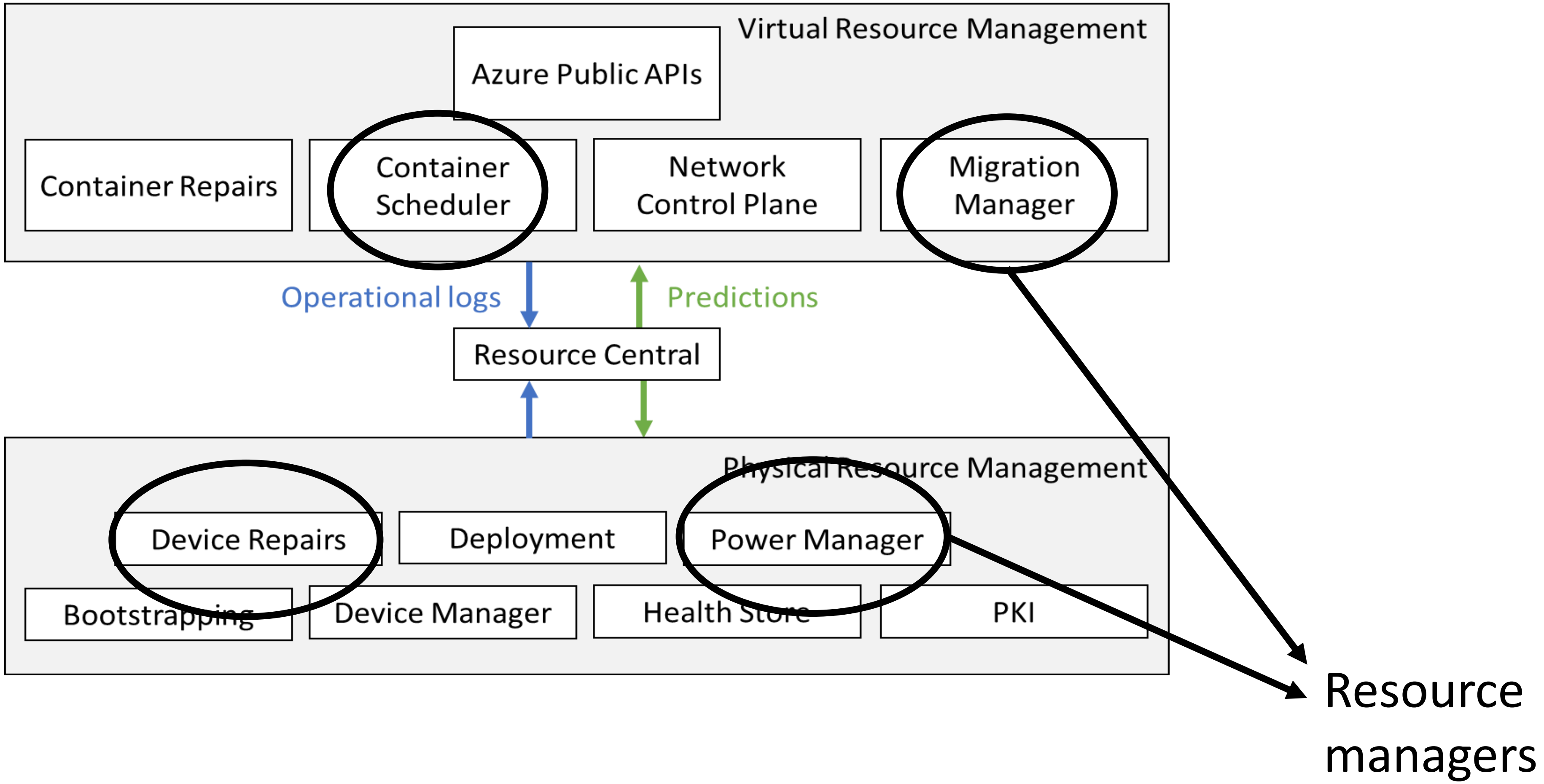
Summary of our approach

A general, passive and delayed-observation framework for all ML tasks

Management	Useful predictions
Container scheduling	VM resource utilization
Scavenging	VM workload class

We are building Resource Central and modifying resource managers to use its predictions in Azure Compute

RC at the center of Azure Compute



Conclusions

ML can improve resource management in cloud platforms

Understanding cloud workload is key for identifying improvements

Resource Central produces high quality workload predictions

Passive and delayed-observation framework is simpler. Scale is the problem!

Predictions enable lower costs while retaining good QoS

Thanks

Resource Central: Understanding and Predicting Workloads for Improved Resource Management in Large Cloud Platforms

VM Traces -- https://github.com/Azure/AzurePublicDataset/

Marcus Fontoura

Eli Cortez, Anand Bonde, Alexandre Muzio, Thomas Moscibroda, Gabriel Magalhaes, Mark Russinovich, Ricardo Bianchini