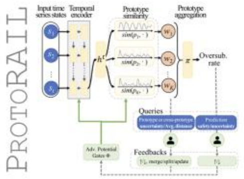
Microsoft

SCIENCE OVER SPECTACLE

ProtoRAIL : A Risk-cognizant Imitation Agent for Adaptive vCPU Oversubscription In the Cloud

Lu Wang, Mayukh Das, Fangkai Yang, Bo Qiao,
Hang Dong, Si Qin, Victor Ruehle, Chetan Bansal, Eli Cortez,
Íñigo Goiri, Saravan Rajmohan, Qingwei Lin, Dongmei Zhang

M365 Research, DKI, Azure Resource Central



Presented at **MLSys 2025**, Santra Clara, CA, USA, 12-15 May, 2025

Outline

1. The Oversubscription Problem
2. Landscape
3. The ProtoRAIL Agent
4. Evaluations
5. Discussion

The Oversubscription Problem

Can we bite more than we can chew? Why? How?

AIOps for Cloud Efficiency

Problem Space

- Optimal mapping problem
- Resource Pool <-> Consumer pool
- **Data-driven** optimal mapping

All Resource types – Compute, Memory, Power, Storage, Cache

All types of consumers – 1P**, 3P, all domains

Runtime Telemetry, Static Metadata, Domain Knowledge

…

Forecasting the future

Forecastable loads is the key feature which makes the problem tractable

Load/Error Forecasting

Resource Usage Forecasting

User Behavior Prediction

Capacity Forecasting

Bandwidth forecasting

…

Packing / Optimization

- Prima facie multi-dimensional stochastic optimization problem
- Exact solution is a NP-Hard problem

Temporal Load Packing

Capacity/Resource/Power Margin optimization

Optimal allocation and packing

Congestion control / optimization

Optimal control / decision-making

…

Competing Objectives

- Utilization Efficiency ∝ 1/Reliability (inverse)
- Baseline reliability/Risk/Perf. guarantees essential

COGS efficiency maximization ↑

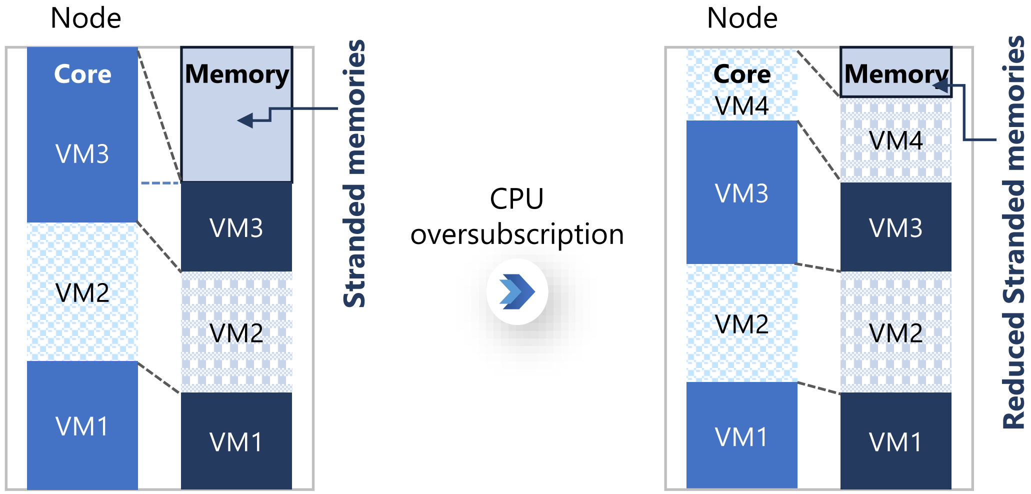
Risk minimization ↓

Reliability Constraints

Workload perf.

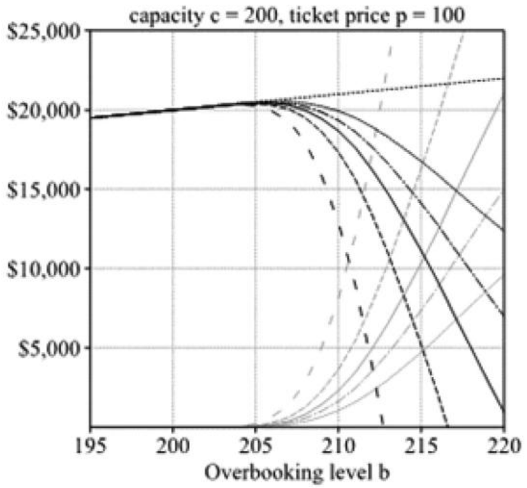
…

Capacity Efficiency



In cloud: vCPU oversubscription

Allocated vCPU < Ask vCPU



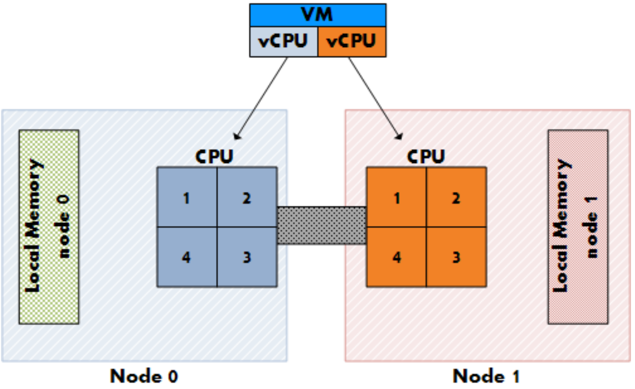
In Transport: Airline Overbooking

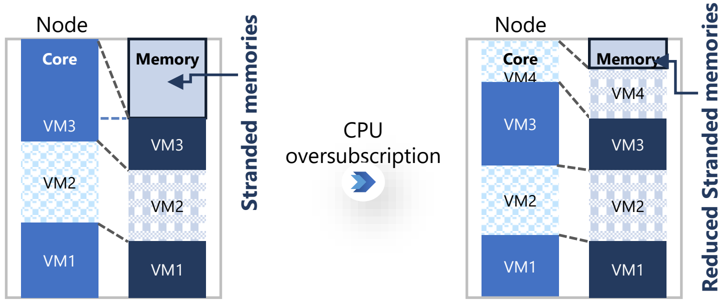
Passengers ticketed > Airline capacity == x<100% of 1 seat per passenger

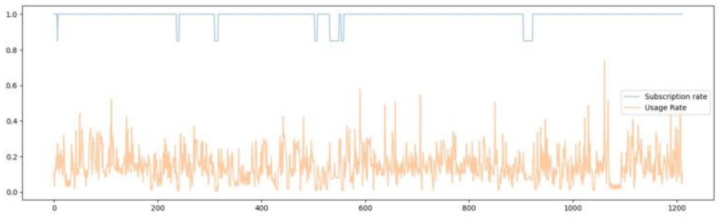
The Oversubscription problem:

- <u>What</u>? System offers more resources than its available capacity, assuming not all users would simultaneously utilize their allocated capacity
- <u>Why</u>? Diminish the sum of unutilized resources and increase gains.
- <u>Impact</u>? COGS (Cost of Goods Sold) reduction → higher revenue margin
- **CPU Oversubscription → minimize stranded cores/memory → Higher capacity efficiency**

The vCPU oversubscription problem







vCPU Oversubscription problem:

- Why Oversubscription?

  User requests 16 #vCore VMs | Physical Node has #60 physical CPU cores

  1 vCore = 1.2 Physical Core

  $$16 * 3 * 1.2 = 57.6 \approx 60\, physical$$

  So node can fit at most 3 VMs

  - VM1 99 %tile usage = 6 vCores
  - VM2 99 %tile usage = 12 vCores
  - VM3 99 %tile usage = 8 vCores

  $$(6 + 12 + 8 = 26) * 1.2 = 31.2 \approx 32\, physical$$

  28 Stranded Cores

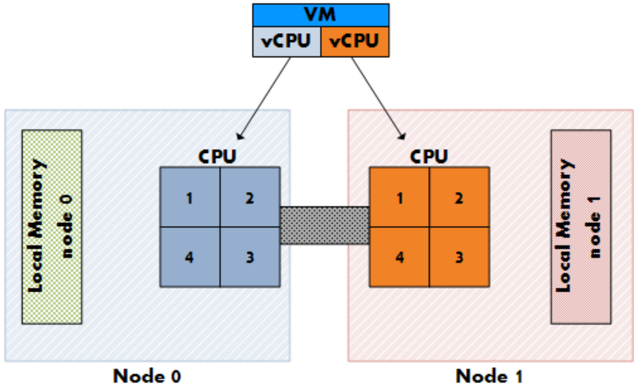
  If we allocate less than requested we can pack more VMs into physical node.

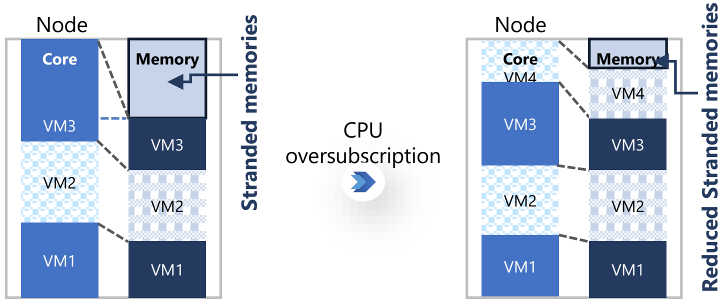
Oversubscription Ratio

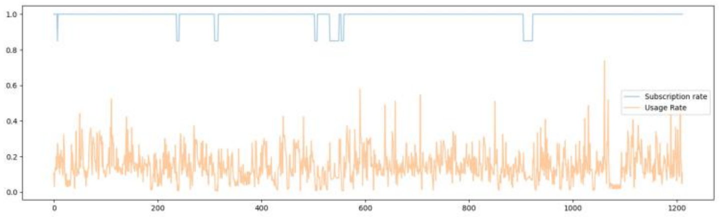
- Should we allocate **50% of requested vCore** to VM1 or is it too tight?
- What if usage peaks higher than usual? Will node have enough extra capacity?
- Does the problem of deciding % and packing VMs be solved together

Intractable

The vCPU oversubscription problem







Oversubscription problem:

- We de-couple for tractable solution.
- Find optimal Oversubscription Ratio $\zeta = \frac{Allocated}{Requested}$ → Assuming VM allocator is doing a fair job (even if not optimal).
- $\underbrace{\underset{\pi}{\operatorname{argmin}}\ \mathbb{E}_{\zeta\sim\pi}[\sum_{nodes} Risk_{\zeta}]}_{\textbf{Risk}} \oplus \underbrace{\underset{\pi}{\operatorname{argmax}}\ \mathbb{E}_{\zeta\sim\pi}[\sum_{nodes} Saved\ \#VCores_{\zeta}]}_{\textbf{Benefit}}$

Challenges:

- Competing objectives
  - Aggressive policy → Contention/JITTER risk;
  - Conservative policy → Stranded cores / memory i.e. Efficiency loss
- Varying demand patterns
- Granularity
- Safety

$\left.\begin{array}{l} \\ \\ \\ \end{array}\right\}$ We need an adaptive solution cognizant of risks

Landscape

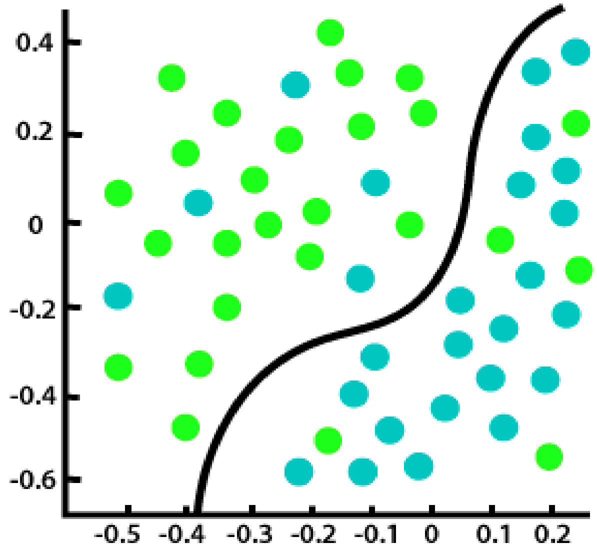
Story so far …

Intelligent Capacity Efficiency Platforms

Azure Resource Central Platform [2]

A Data-driven Resource intelligence platform for all of Azure 1st party

- Data science pipelines | Statistical Models
- Robust VM allocators – based on predictive heuristics
- Utilization Predictions
- Power Predictions



Current naïve solution for oversubscription

- Heuristics-based Binary Usage Classifier
- If Expected Usage <= Threshold, THEN Oversubscription = True/False
- Constant X% oversubscription rate for **all VMs of a client** – across **entire lifetime**
- Trivial savings – Not risk aware

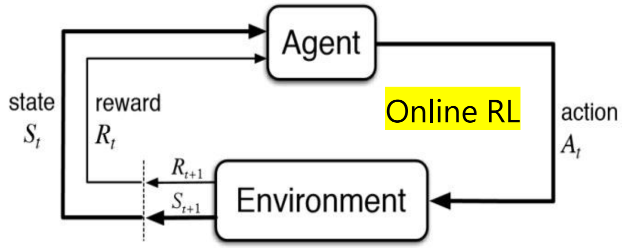
Intelligent Capacity Efficiency Platforms

[Online] Multi-Agent RL based Oversubscription Control [1]

Multi-agent Reinforcement Learning for adaptive oversubscription control
Each user/VM is agent – needs to factor in allocator/packing heuristics

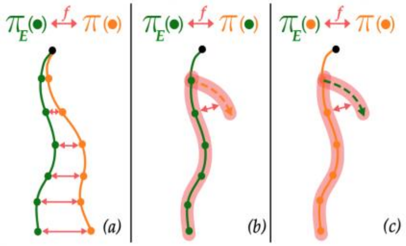
CHALLENGES:

I. How to train RL agents on real environments
II. Suitable reward function – reward sparsity, delayed feedback etc.
III. Granularity of decisions(s) – cluster, group/service, subscription, VMs
IV. Safety against "risk" is not trivial – convergence is tricky in constrained RL formulations.



VS

Offline Imitation Learning



ScroogeVM – LTSM based oversubscription manager [3]

LSTM – based predictor from offline telemetry

COIN – Chance-Constrained Offline Learning from uncertain data [4]

Imitation Learning with stochastic constraints for handling uncertainty

Lot of usage telemetry | Can be trained offline | Safer

BUT: Data is noisy / uncertain. No Ground truth. Granularity of decisions(s) – temporal/spatial

[Resource Level Efficiency]

Key Takeaways

Key problem: Oversubscription

- CPU Oversubscription → minimize stranded cores/memory → Higher capacity efficiency

[Problem] Less physical nodes ←→ More Clients

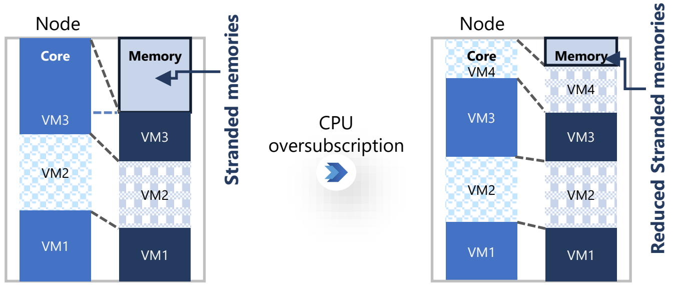
[Forecast] CPU Usage Forecast

[Optimization] Optimize/Decide **oversubscription rate (allocated/requested)** → better VM packing

[Objective] COGS (stranded resources) vs Overloading Risk

In-Prod Heuristics solution

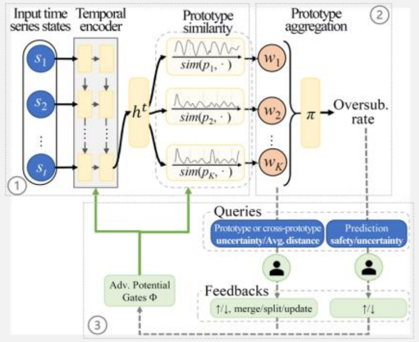
- Usage Classifier
- Oversub = True/False
- Const. X% oversub. rate for all VMs for a client – for entire lifetime
- Trivial savings



Challenges

i. Dynamic demand – NO ground truth;
ii. Granularity VMs / Subscription / Service
iii. Noisy uncertain data - risk

Solution: Smart Imitation Learning (offline mode RL)



- Policy model from usage telemetry
- + Risk min. via domain knowledge
- + Prototype learning for multi-granular decision.

**Fine grained decision & Adaptive solution

8X increase in P95 sellable cores↑

~0% (Negligible) Risk of Jitter↓

Impact: 1) **8X** Savings 2) Multi-million $ **footprint**

Status & Road Map:

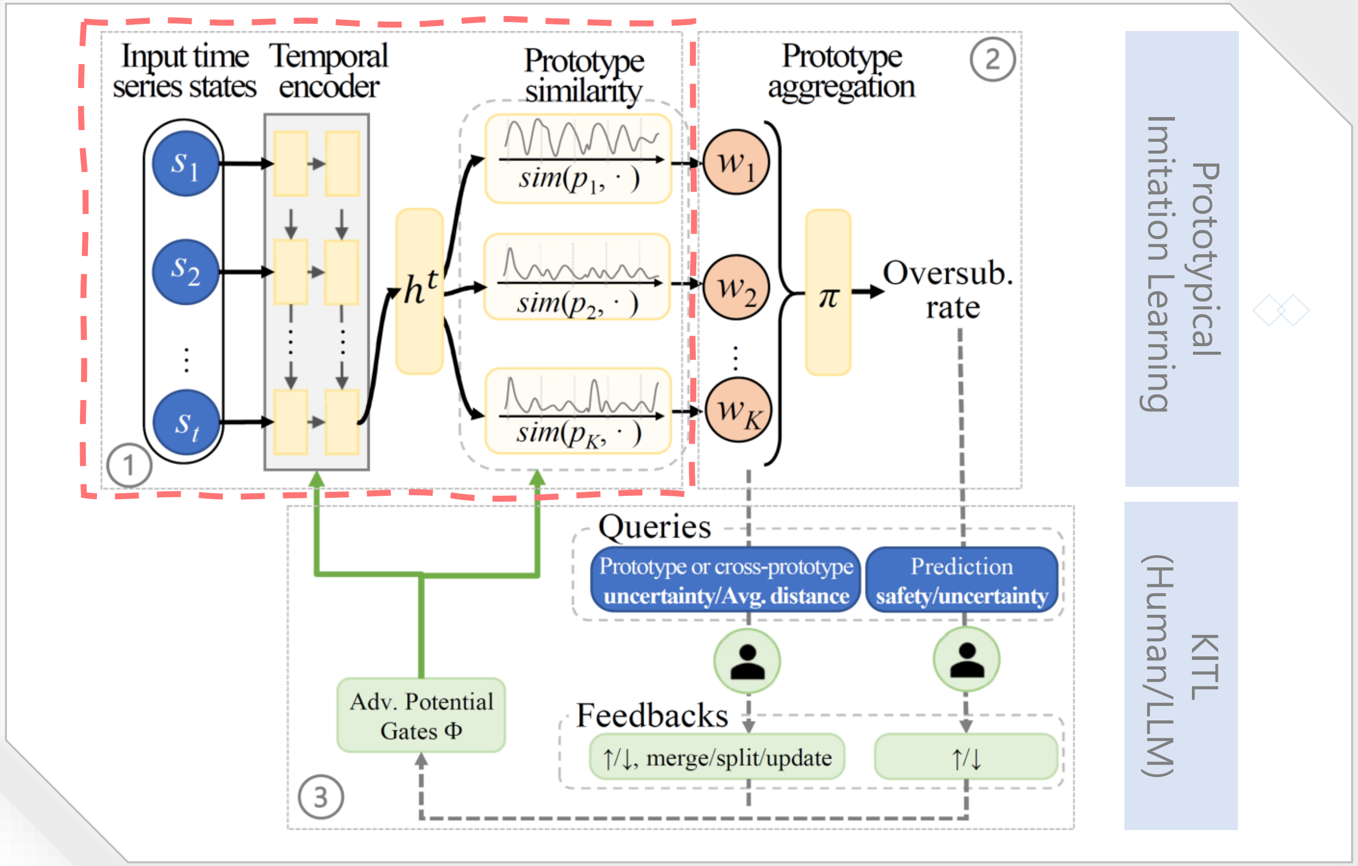
- Model / Algorithmic development as well as POC complete – results promising by deploying on subset of 1st party (internal clients)
- Pilot studies going on for all 1st party VMs
- Full deployment in pipeline
- Adapt to memory / cache and power oversubscription – near future

WWW '23; CIKM '24,

The ProtoRAIL Agent

Intelligent CPU Oversubscription

PROTOtypical **R**isk-cognizant **A**ctive **I**mitation **L**earning (**PROTORAIL**)



Module ①

Policy model from usage telemetry

- Prototype discovery and learning for **multi-granular** decision.
- Prototypes[1] are "**representative points**" for equivalence classes of approximately symmetric patterns
- Temporal Usage trajectories → Encode to embedding space h_t
- Learn N prototype embeddings
- Customized similarity function
- Ensure diversity of learned prototypes

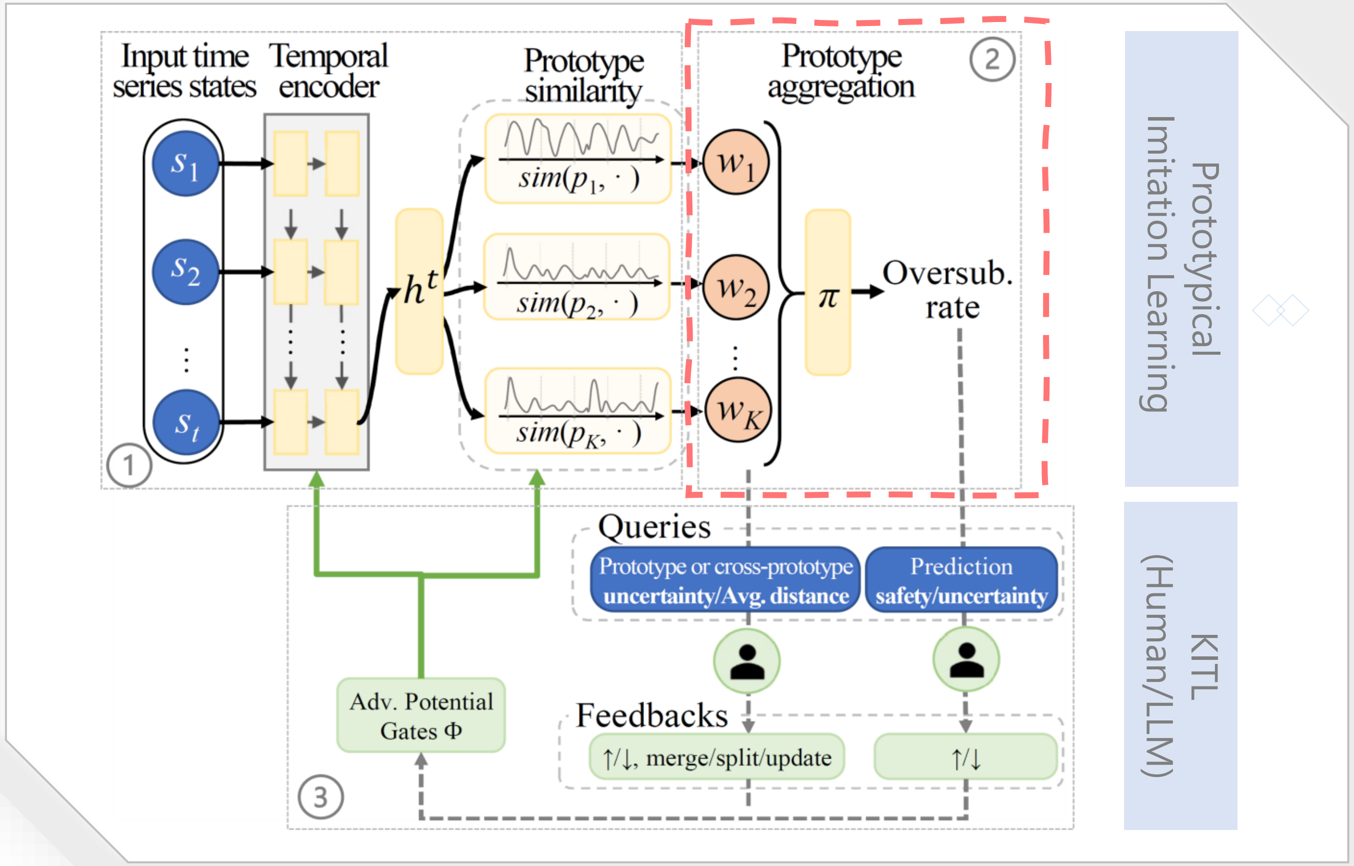
LOSS : $L(\pi_D, \pi_\theta) = [\ell(h, C) + \ell(C_i, C_j)]$

[1] Molnar, C. (2020). Interpretable machine learning. Lulu. com.; Kim et al., NeurIPS 2016

[Resource Level Efficiency]

Intelligent CPU Oversubscription

PROTOtypical **R**isk-cognizant **A**ctive **I**mitation **L**earning (**PROTORAIL**)



Module 2

Policy model from usage telemetry

- Imitation Learning over prototypes
- New sample → h_t^{new} → $sim(h_t^{new}, \{Prototypes\}) = \{w_1, \dots, w_N\}$
- Prediction = aggregate

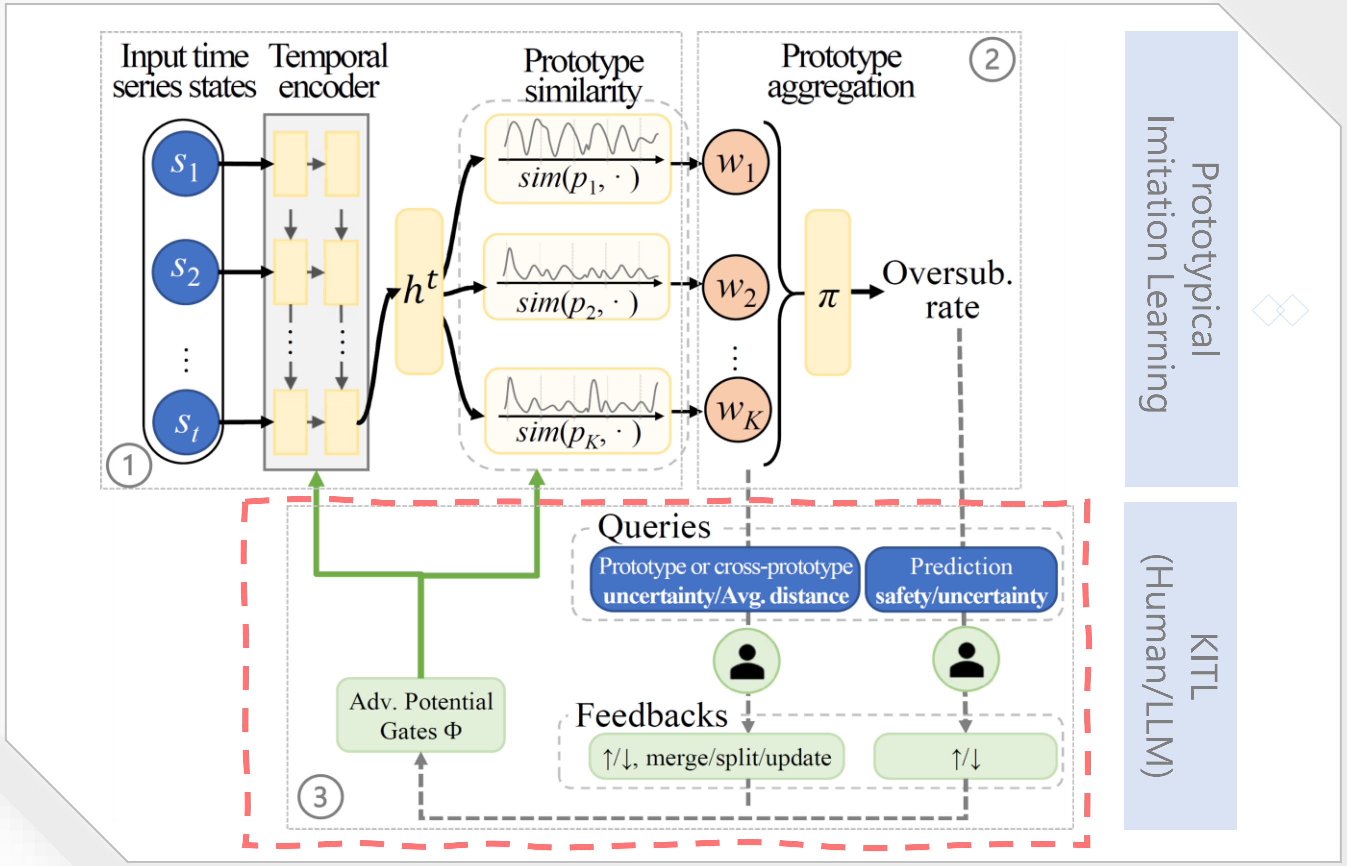
$$\pi_\theta(\bar{\zeta}|.) = \{w_k\}_{k=1}^{N} . \{\pi_\theta^k(\zeta|.)\}_{k=1}^{N}$$

- Optimize with Imitation (behavior cloning) loss

LOSS : $L(\pi_D, \pi_\theta) = \left[\begin{matrix} \ell(\zeta, \hat{\zeta}) + \\ \ell(h, C) + \ell(C_i, C_j) \end{matrix}\right]$

Intelligent CPU Oversubscription

PROTOtypical **R**isk-cognizant **A**ctive **I**mitation **L**earning (**PROTORAIL**)



Module 3

- Risk min. via domain knowledge (KITL)
- **ACTIVE** feedback on quality of prototypes + predictions
- Human Domain Expert / LLM expert
- Feedback = upvote / downvote / merge / split
- Very Efficient – Minimal queries ~6
- Loss scaling with $e^{\alpha.[\Sigma votes]}$
- LLMs are equally good for generic pattern equivalence feedback – Humans better with nuanced **risk related feedback**

LOSS : $L(\pi_D, \pi_\theta) = [\ell(\zeta, \hat{\zeta}) + \ell(h, C) + \ell(C_i, C_j)] \times e^{\alpha.Feedback}$

Evaluation

Does it work, for real?

Evaluation

Prototypical Risk-cognizant Active Imitation Learning

vCPU Oversubscription: Offline Replay Emulation on 2-week long observations of CPU usage for 1st Party Microsoft workloads on the clusters of 2 regions and on 300 clusters.

Flight Tickets data is collection from US DOT database for last 2 decades. Risk here is **involuntary offboarding risk**.

Approach	vCPU Oversubscription		Flight Tickets	
	Hot Node/Risk↓	Core (Benefit)↑	Cost/Risk↓	Profit↑
Grid-search	0%	7450	0M	0M
Moving Average	1.39%	7628	0.96M	6.79M
DDPG	1.47%	5030	12.37M	2.35M
Behavior Cloning	1.19%	7870	1.47M	7.21M
GAIL	1.2%	6980	2.74M	4.56M
Dagger (*20 time steps*)	0.96%	7938	0.47M	6.95M
LSTM	1.27%	7749	1.82M	4.98M
Coop. Multi-Agent RL	0.89%	7897	0.59M	8.17M
PROTORAIL (*w/o KITL*)	**0%**	**8153**	0.31M	8.79M
PROTORAIL	**0%**	**8161**	0.14M	13.65M

"Manual" indicates manually decided oversubscription, "Heuristics" indicates statistical non-adaptive prediction model based on collected heuristics in production.

Method	MSE ↓	VMs w/ β ↑	VMs at Risk ↓	BRR ↑
Manual	–NA–	109	14	7.79
Heuristics	0.065	3502	185	18.95
PROTORAIL	**0.042**	**3542**	113	**31.34**

	Heuristics (Binary)	ProtoRAIL	Total
% savings in core hours	5.2%	20.5%	100%

Deployment Metrics

~ **6-8X increase** in P95 # sellable cores↑

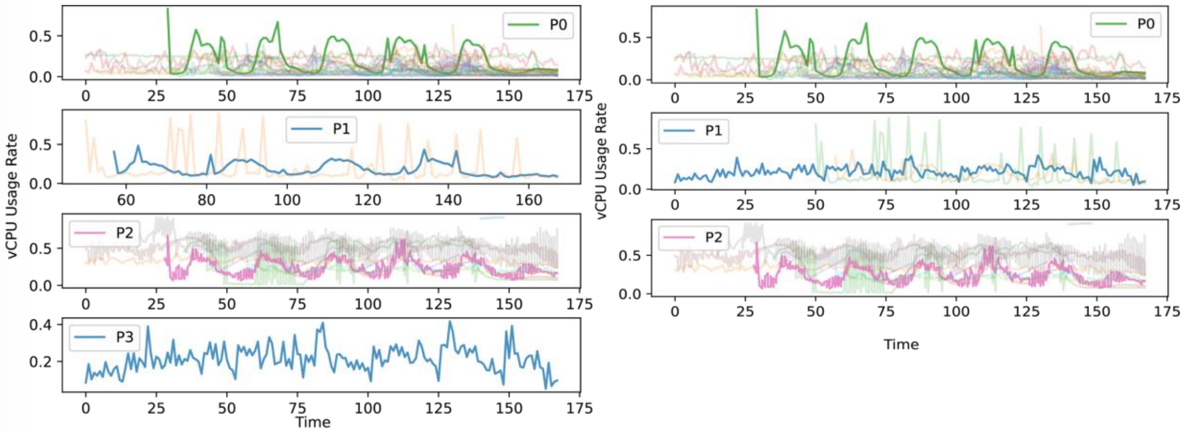
~ **4x increase** in saved (core hours) ↑

~ **0% (Negligible)** Risk of Jitter↓

~ **0.2% Underprediction** Ratio compared to 7% ↓

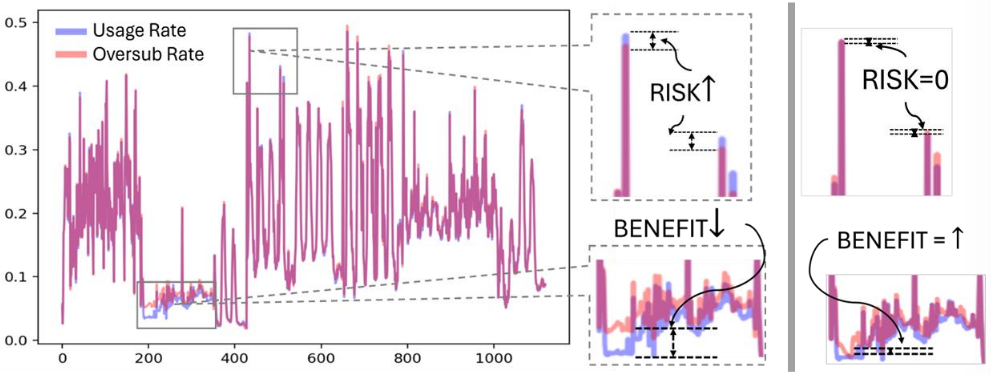
Prototypes Learning / Refinement & Safety

Prototypical Risk-cognizant Active Imitation Learning



Learned Prototypes

After refinement



No KITL

With KITL

Prototypes do capture approximate symmetries in temporal patterns across ANY granularity + Interpretable

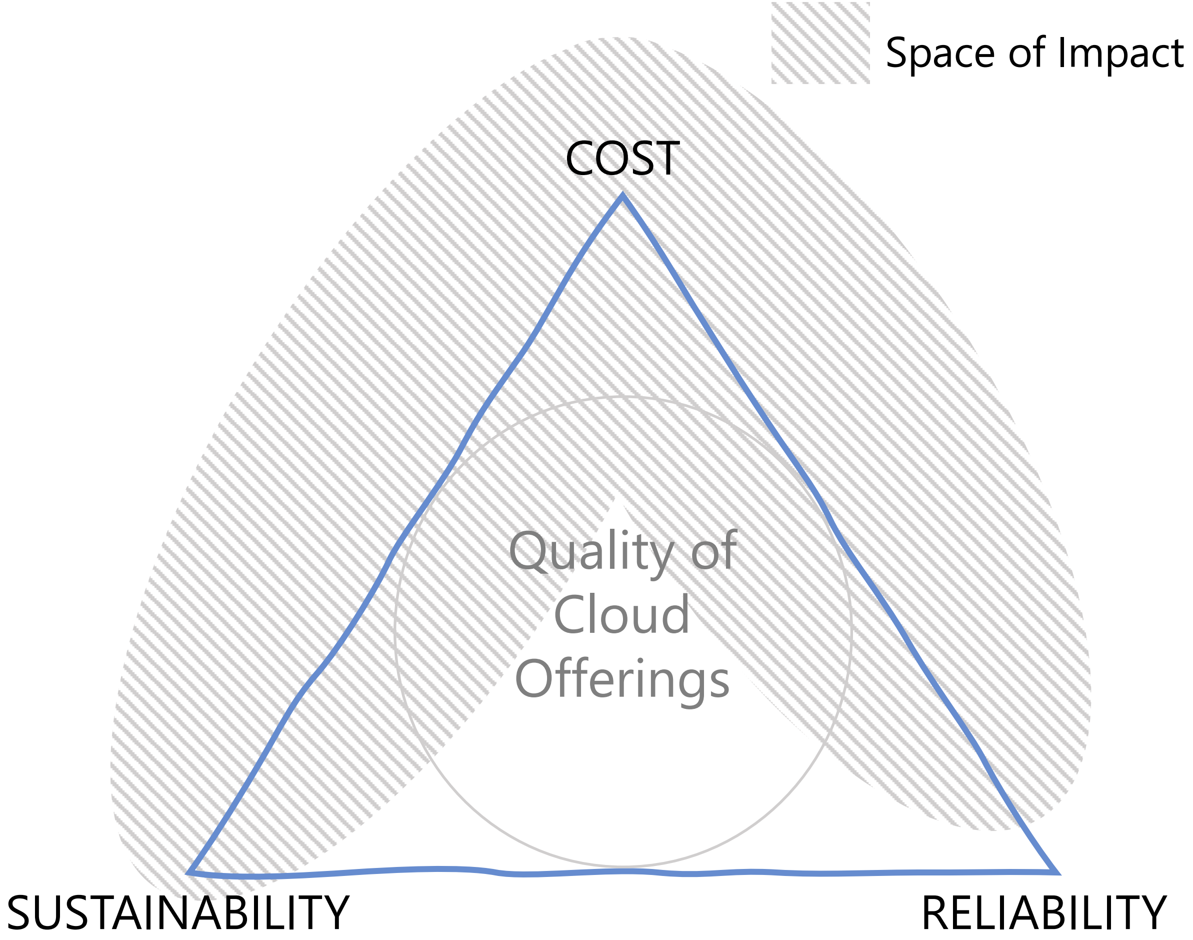
Knowledge in the loop helped get rid of marginally useless prototypes.

Low overhead – Minimal queries

Note how risk gets minimized, Often with Higher Savings, with Knowledge in the loop (KITL)

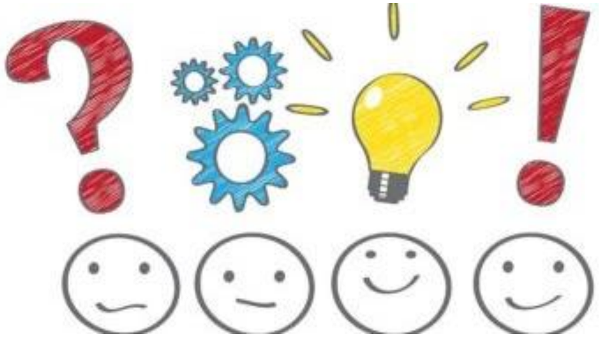
Conclusion and Discussion

- [x] Adaptive Risk-aware Intelligent Multi-granular oversubscription
- [x] Substantially higher sellable cores/ core_hours savings, Minimized overloading risk
- [x] Real tests in internal deployments show significant gains
- [x] Cloud efficiency that is reliable – making cloud operations sustainable
- [x] Novel Prototypical Imitation Learning – Can generalize to any resource optimization problem
- [ ] Full Deployment for all 1st Party
- [ ] Adapting to other oversubscription / overbooking problems
- [ ] Workload characteristics aware policies



A representation of Mundell-Flemming's Trilemma in Macroeconomics applied on cloud business

Thank you!



Link to Paper

Our Research Group M365 Research, Microsoft:
https://www.microsoft.com/en-us/research/group/m365-research/

Systems Innovation Effort at Microsoft:
https://www.microsoft.com/en-us/research/group/systems-innovation/

Acknowledgements: DKI (Microsoft, Beijing), Azure Resource Central (Microsoft)