Ayse Coskun
Professor, ECE Department
Director, Center for Information & Systems Engineering
Boston University

AI for Computer System Analytics

BOSTON UNIVERSITY

PEACLAB

Vision: Automating Computer System Operation

Machine Learning Models

- Image & speech recognition
- Healthcare
- Robotics
- Navigation
- … *many others*

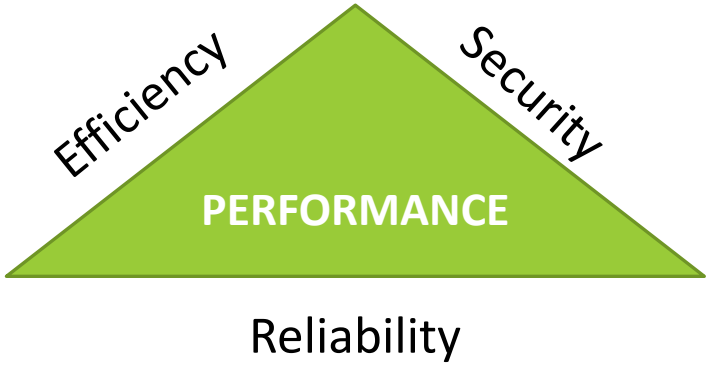
Computer System Administration

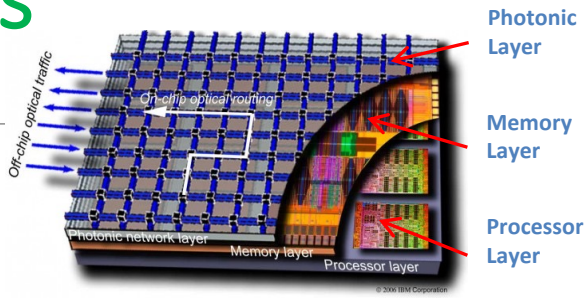
```
127.0.0.1 - - [31/Oct/2017:11:11:37 +0530] "GET /icons/blank.gif HTTP/1.1" 200
fox/56.0"
127.0.0.1 - - [31/Oct/2017:11:11:37 +0530] "GET /icons/folder.gif HTTP/1.1" 200
efox/56.0"
127.0.0.1 - - [31/Oct/2017:11:11:37 +0530] "GET /icons/text.gif HTTP/1.1" 200 5
ox/56.0"
127.0.0.1 - - [31/Oct/2017:11:11:38 +0530] "GET /favicon.ico HTTP/1.1" 404 500
127.0.0.1 - - [31/Oct/2017:11:12:05 +0530] "GET /tecmint/ HTTP/1.1" 200 787 "ht
)"
127.0.0.1 - - [31/Oct/2017:11:12:05 +0530] "GET /icons/back.gif HTTP/1.1" 200 4
)1 Firefox/56.0"
127.0.0.1 - - [31/Oct/2017:11:13:58 +0530] "GET /tecmint/Videos/ HTTP/1.1" 200
101 Firefox/56.0"
127.0.0.1 - - [31/Oct/2017:11:13:58 +0530] "GET /icons/compressed.gif HTTP/1.1"
) Gecko/20100101 Firefox/56.0"
127.0.0.1 - - [31/Oct/2017:11:13:58 +0530] "GET /icons/movie.gif HTTP/1.1" 200
o/20100101 Firefox/56.0"
```

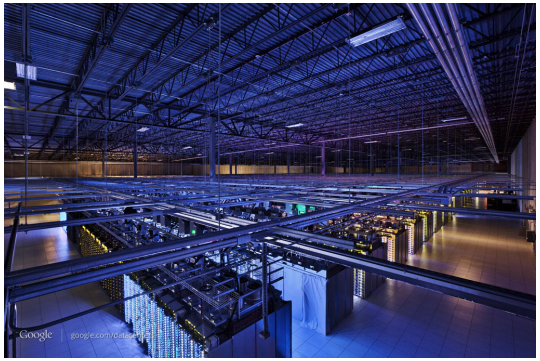
Computer System Design Goals

- Improving quality of life
- Advancing science
- Making money
- …

Important concerns:





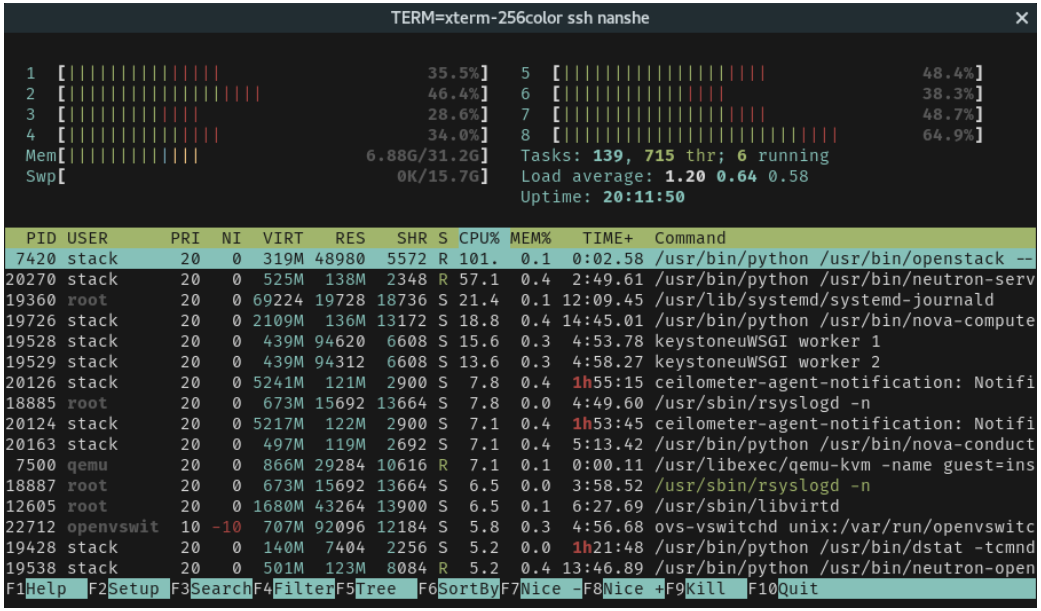


How to Locate or Identify a Problem?

How to Locate or Identify a Problem?

How to locate or identify a problem?

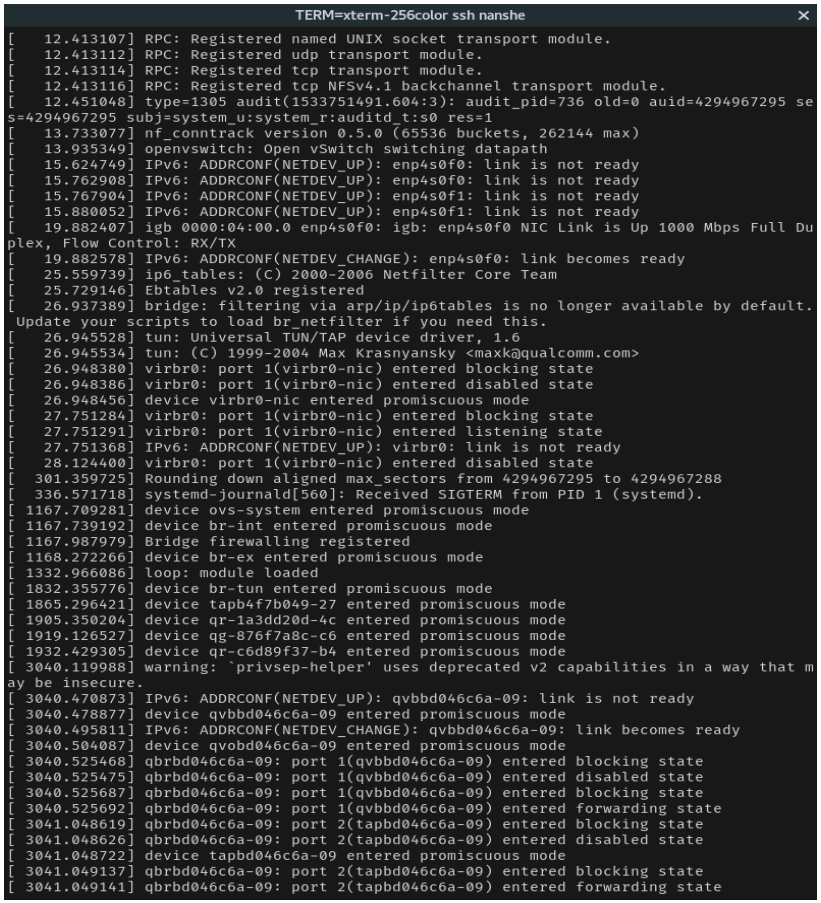
- Numeric data



How to Locate or Identify a Problem?

How to locate or identify a problem?

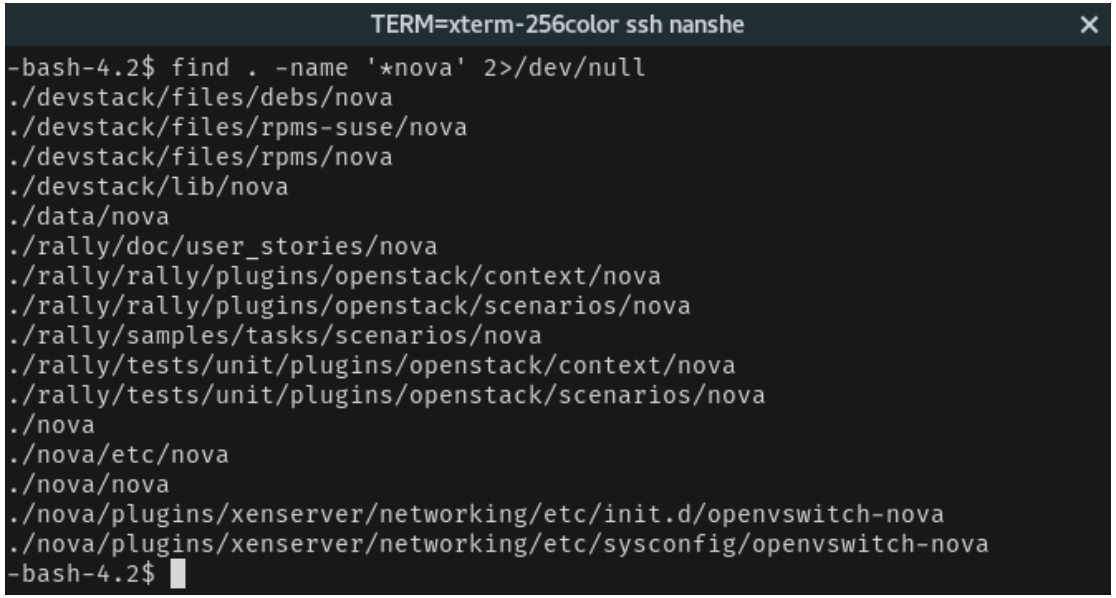
- Numeric data
- Logs



How to Locate or Identify a Problem?

How to locate or identify a problem?

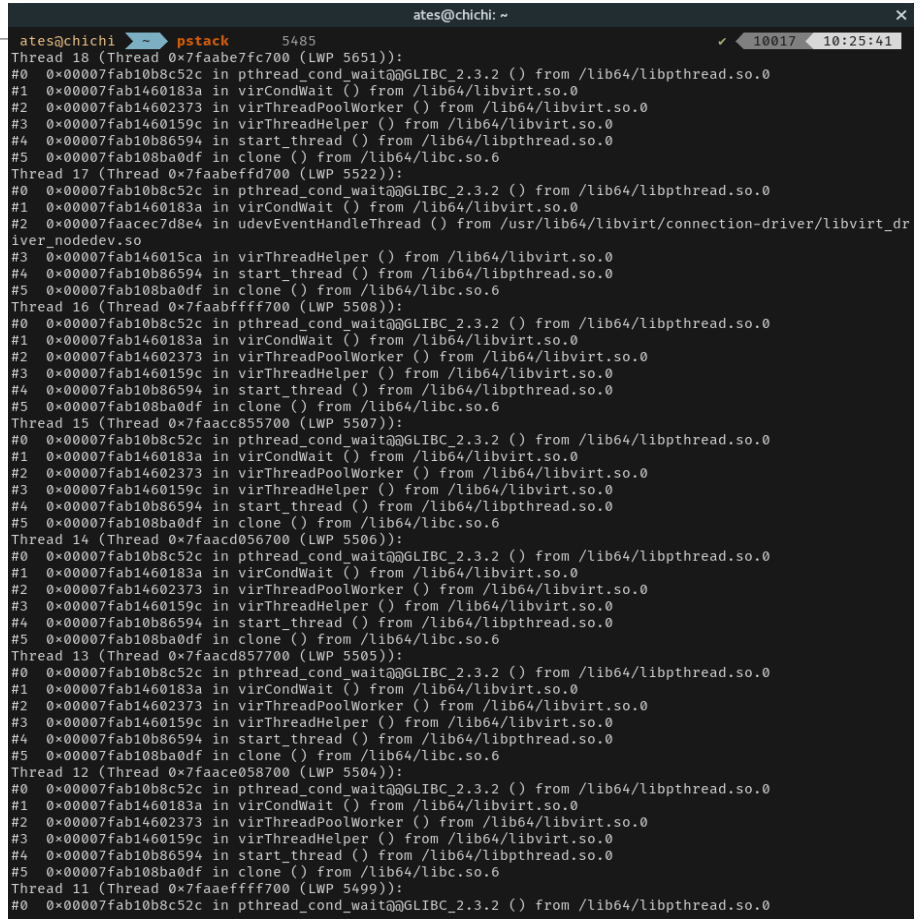
- Numeric data
- Logs
- Files



How to Locate or Identify a Problem?

How to locate or identify a problem?

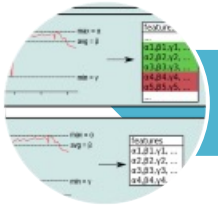
- Numeric data
- Logs
- Files
- Traces
- …



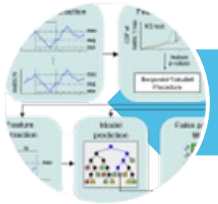
Today's Complex Computer Systems

- How to locate or identify a problem?
  - Terabytes of numeric data
  - Logs from 10,000 systems (or runs)
  - Files in VMs/containers belonging to 100s of different companies
  - Traces of distributed systems
- Automated methods are essential
  - Fast, repeatable
  - Outages and bugs are costly
- Machine learning is a good fit
  - Similar problems often repeat
  - Broadly applicable

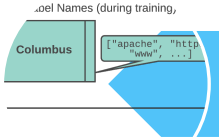
Outline



Diagnosing performance variations



Identifying applications



Towards real-world deployments

Performance Predictability at Large Scale

HPC systems

- Over 1 million cores
- Parallel applications

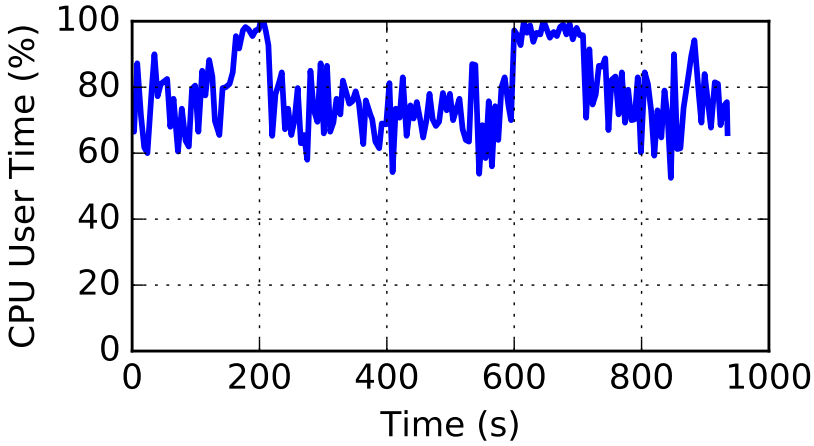
Performance variations

- Same job on the same system → **~8x delay in job execution time** *[Zhang et al., Cluster'20]*
  - Poor scheduling
  - Reduced efficiency
- Much of the variations are caused by **system anomalies**
  - Shared resource contention
  - Orphan processes from previous jobs
  - Hidden hardware problems

Numeric Monitoring Data

Anomalies manifest themselves in performance metrics

- OS Counters
- System Utilization
- Network/Hardware Counters



Challenge:

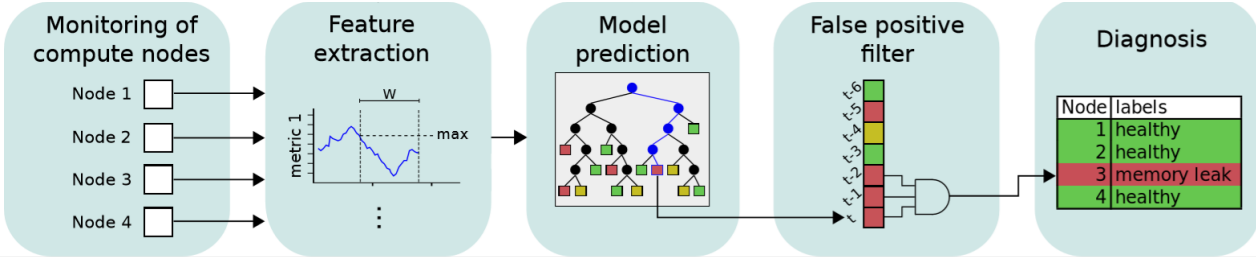
- Vast data volume: TBs per day, 100s of metrics

Advantage:

- Covers many subsystems (CPU, memory, network, …)
- Readily available

Performance Variation Diagnosis

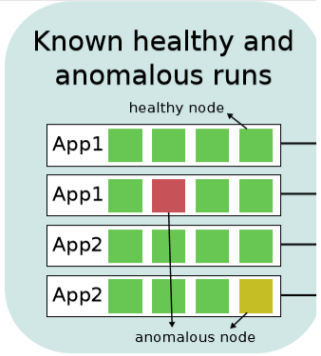
- Automated analytics framework
- Learn anomaly characteristics using labeled historical data



- Sources of data
  - Past anomalous runs labelled by system administrators, users, etc.
  - Controlled experiments

[Tuncer et al., ISC-HPC'17 (Gauss award), Tuncer et al., IEEE TPDS'18]

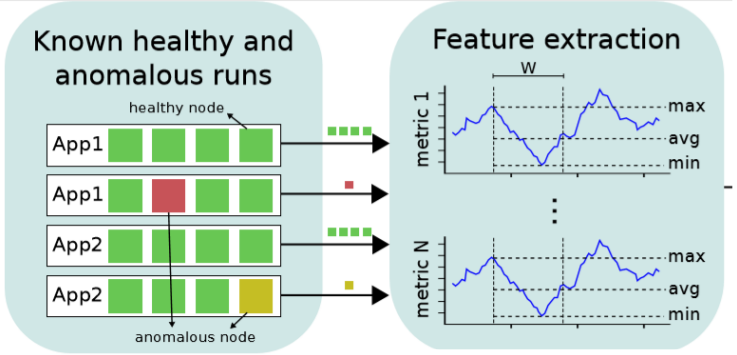
Training Phase: Collecting the Dataset



Insert synthetic anomalies

- Anomalies designed to imitate real anomalies
- Create labeled data
- Each anomaly stresses a specific subsystem

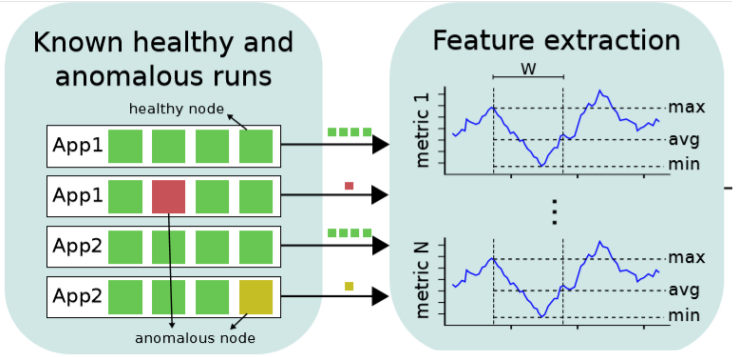
Training Phase: Monitoring



Collect numeric monitoring data

- CPU, memory, network, power, …
- Collected from procfs and hardware counters
- Collection framework already exists in many systems
  - No additional compute overhead at server scale

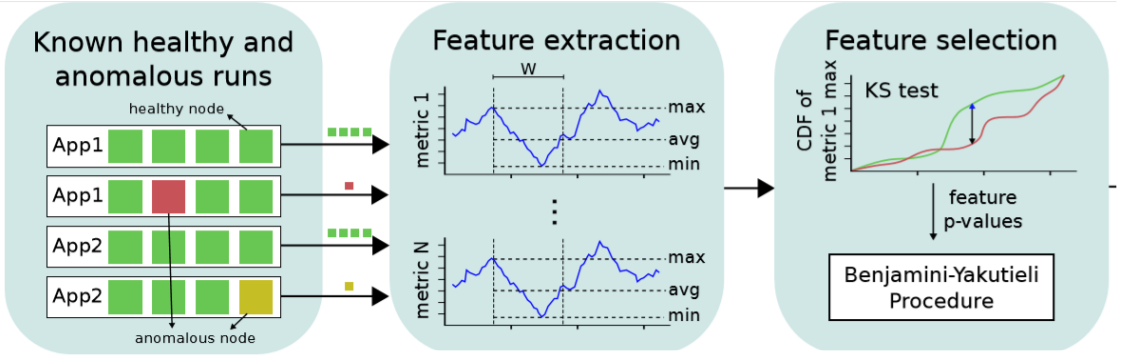
Training Phase: Feature Generation



Features that retain the characteristics while removing noise

- Easy to compute
- Compute features per rolling window
- Stddev, min, max, avg, skew, kurtosis, percentiles, etc.

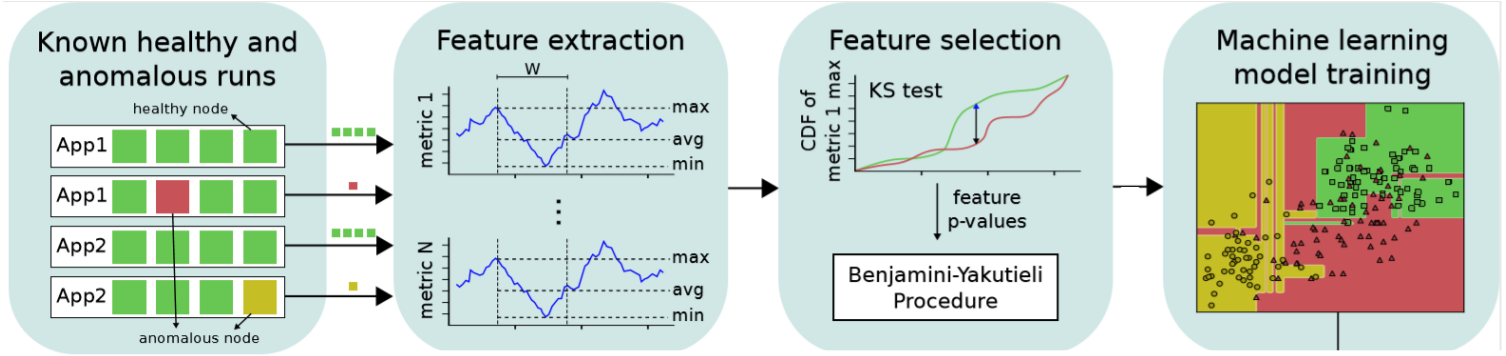
Training Phase: Feature Selection



Some features are not useful for prediction

- Remove features which are not affected by the presence of an anomaly
- Reduces overhead for runtime
- Increases accuracy

Training Phase: Model Training



Train using selected features

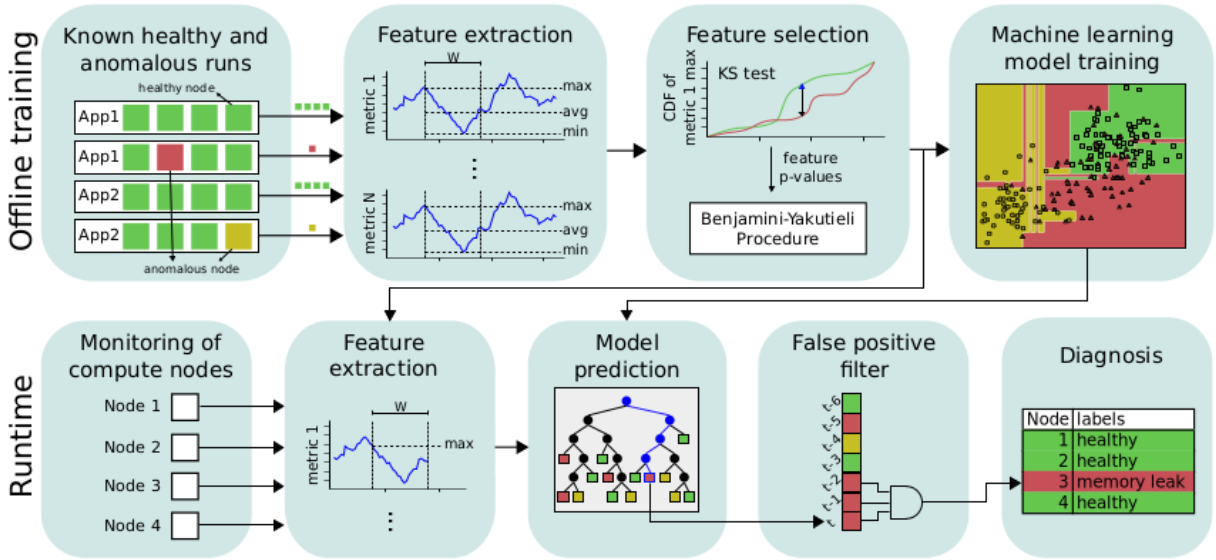
- Tree based algorithms perform better
- Random Forest, Decision Tree, AdaBoost

Complete Framework

At runtime, used previously trained classifiers

Extract only selected features

Filter false positives



Synthetic Anomalies

- Goal: Mimic real-life anomalies in supercomputers
  - Each anomaly has multiple intensity settings and targets a specific subsystem (CPU, memory, etc.)
  - Anomaly runs on *one of the nodes* of the application

Anomaly Name	Subsystem	Operation	Real-Life Event Modeled
Dial	CPU	Floating point operations	CPU intensive orphan process
Dcopy	Cache	Read and write	Cache interference
Linkclog	Network	Inject sleep before MPI calls	Network interference
Memeater	Memory	Allocate, write and realloc()	Memory intensive orphan process
Leak	Memory	Allocate	Memory leak

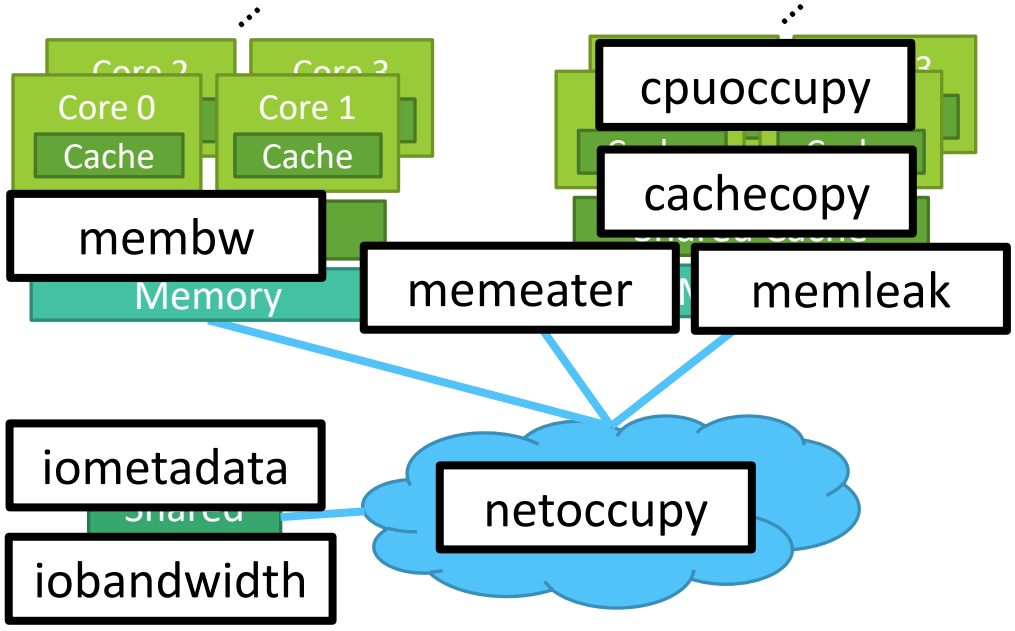
HPAS: An HPC Performance Anomaly Suite

Anomalies targeting different subsystems:

- CPU – cpuoccupy
- Cache – cachecopy
- Memory Bandwidth – membw
- Memory Capacity – memleak, memeater
- Network – netoccupy
- I/O – iometadata, iobandwidth

Implemented in C, Bash

- Separate user-space process
- Minimum impact outside target subsystem
- Tunable intensity



www.github.com/peaclab/hpas

[Ates et al., ICPP'19]

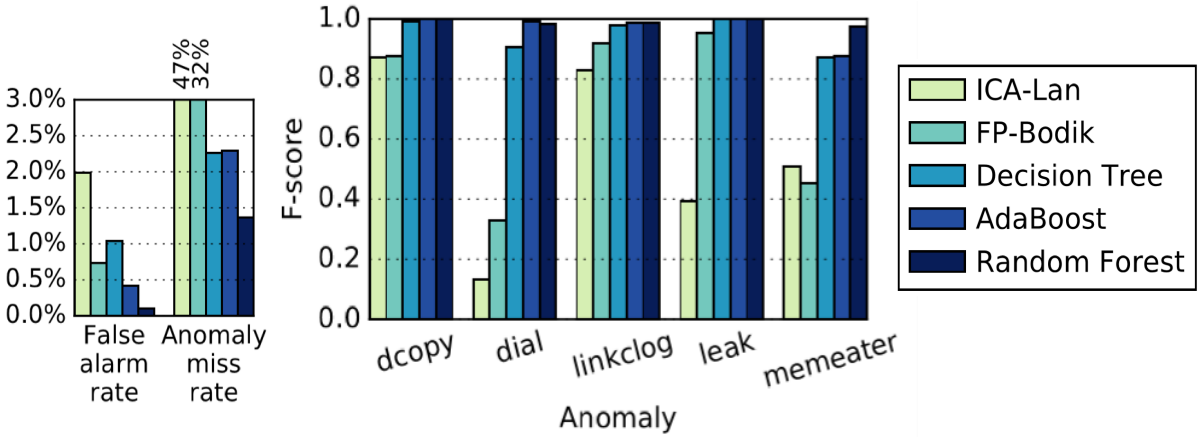
Evaluation

Volta – a Cray XC30m supercomputer

Baselines:

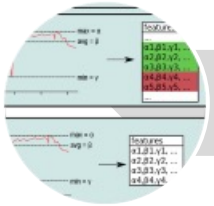
- ICA-Lan, FP-Bodik

Identifies 98% of injected performance variation

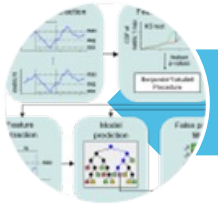


- 0.08% false alarm rate
- Robust against unknown input configurations/anomaly intensities

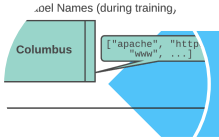
Outline



Diagnosing performance variations



Identifying applications



Towards real-world deployments

Why Identify an Application?

Detect illegal/unwanted applications

- Known malicious or fraudulent apps
- Wasteful apps
- Vulnerable or buggy apps

Detect (types of) applications to improve system management

- Performance
- Energy efficiency
- System design or software optimization

Application name or script name often may not mean much, or may be altered on purpose

US Government Bans

Russian nuclear scientists arrested for 'Bitcoin mining plot'

9 February 2018 Share

Fending off Bitcoin Mining HPC Thieves – Idaho National Lab's Cryptojacking Detector

February 26, 2021 by staff

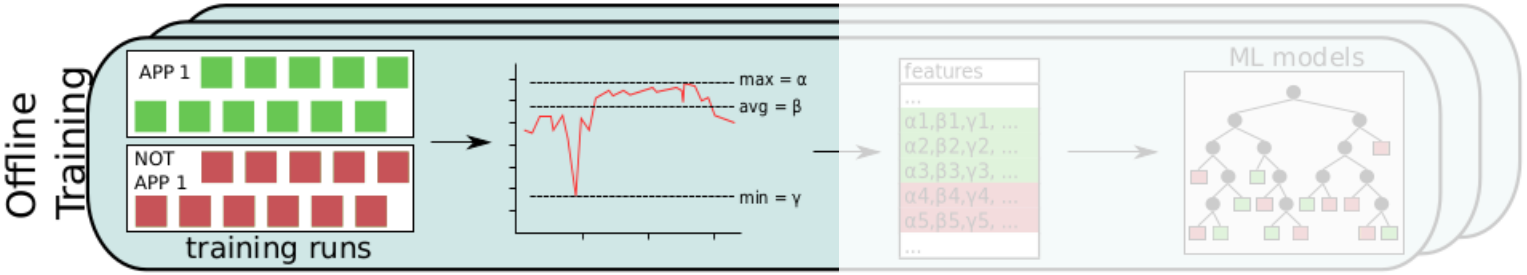
Demand for HPC resources from cryptocurrency miners has only increased with the explosion in the Bitcoin's price – up 65 percent year-to-date. But with that demand has come mining malware, such as Monero and Lighting, algorithms that embed themselves into HPC applications.

Idaho National Laboratory

Idaho National Laboratory (INL) has announced development of a machine translation-based cryptocurrency mining malware detector. On a U.S. Department of Energy contract announcement site, the lab said it seeks a partner to join in a licensing or collaborative research agreement to commercialize the malware detector.

With Bitcoin at $46,000 (it jumped past $50,000 earlier this week), cryptocurrency mining, a highly complex and expensive process, is more of a lure than ever because miners can

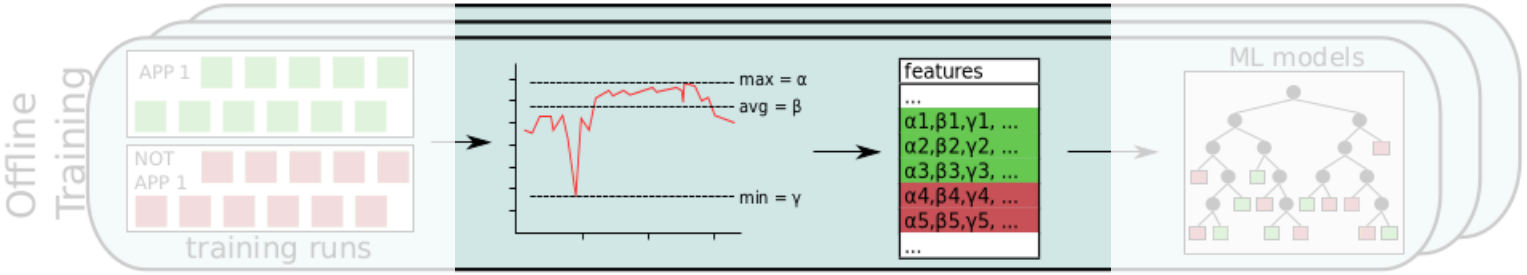
Taxonomist: An Application Detection Framework



- Data is collected from applications of interest
- 100s of timeseries per node

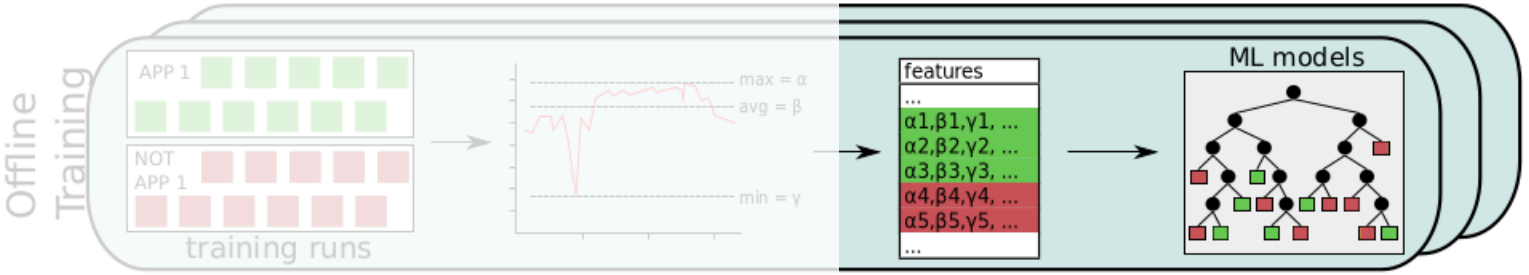
[Ates et al., EuroPar'18 (Best Artifact Award)]

Taxonomist: An Application Detection Framework



- Statistical features are used to summarize the timeseries
  - 5^{th}, 25^{th}, 50^{th}, 75^{th}, 95^{th} percentiles, min, max, skew, kurtosis, standard deviation, mean
- Keep the trend, remove noise

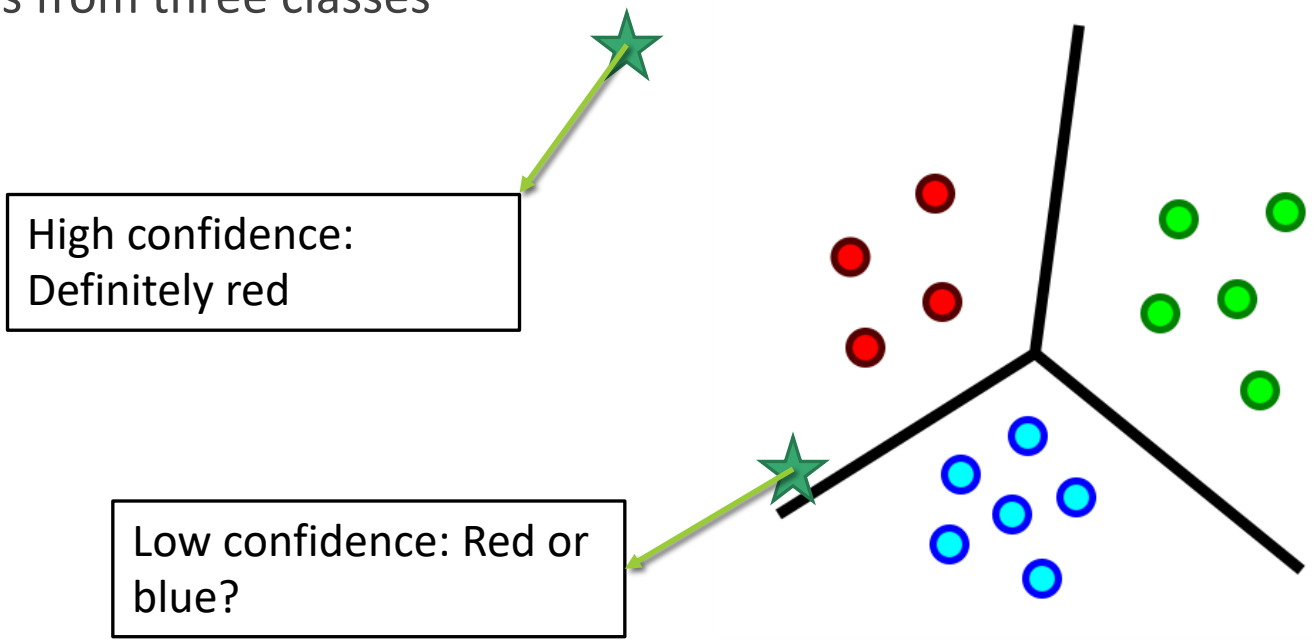
Taxonomist: An Application Detection Framework



- Machine learning models are trained per application
  - One versus rest
- Tested with Random Forest, Decision Tree, SVM, Extra Trees

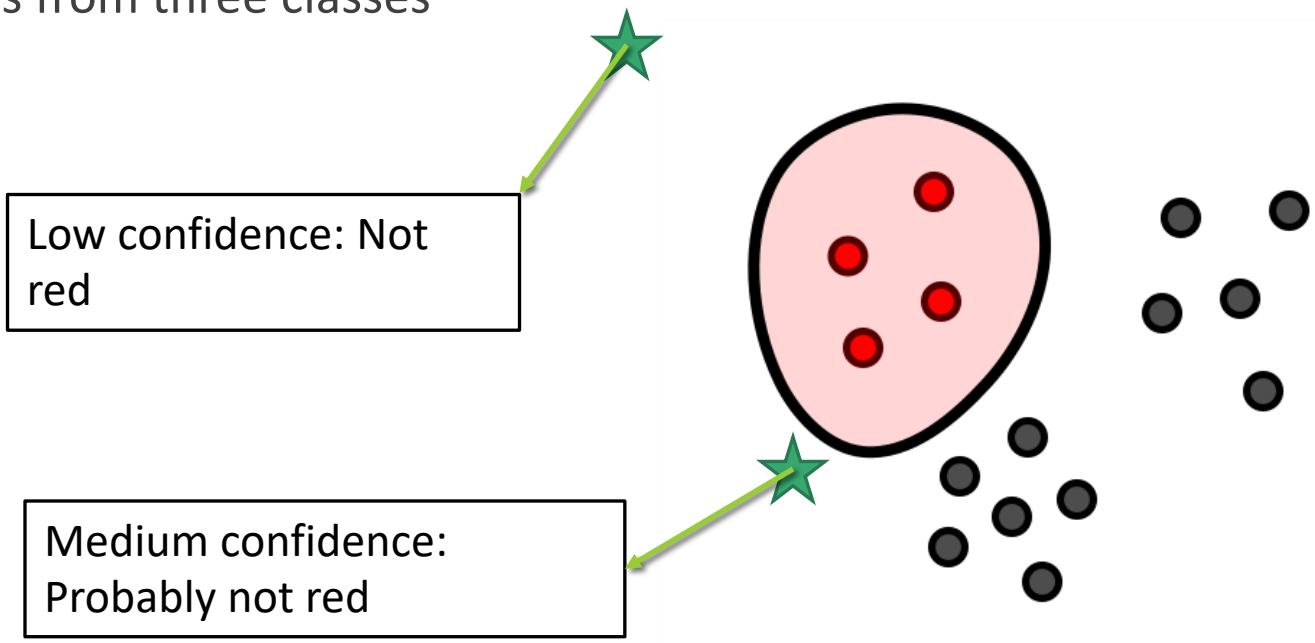
Multiclass Classifier

Observations from three classes

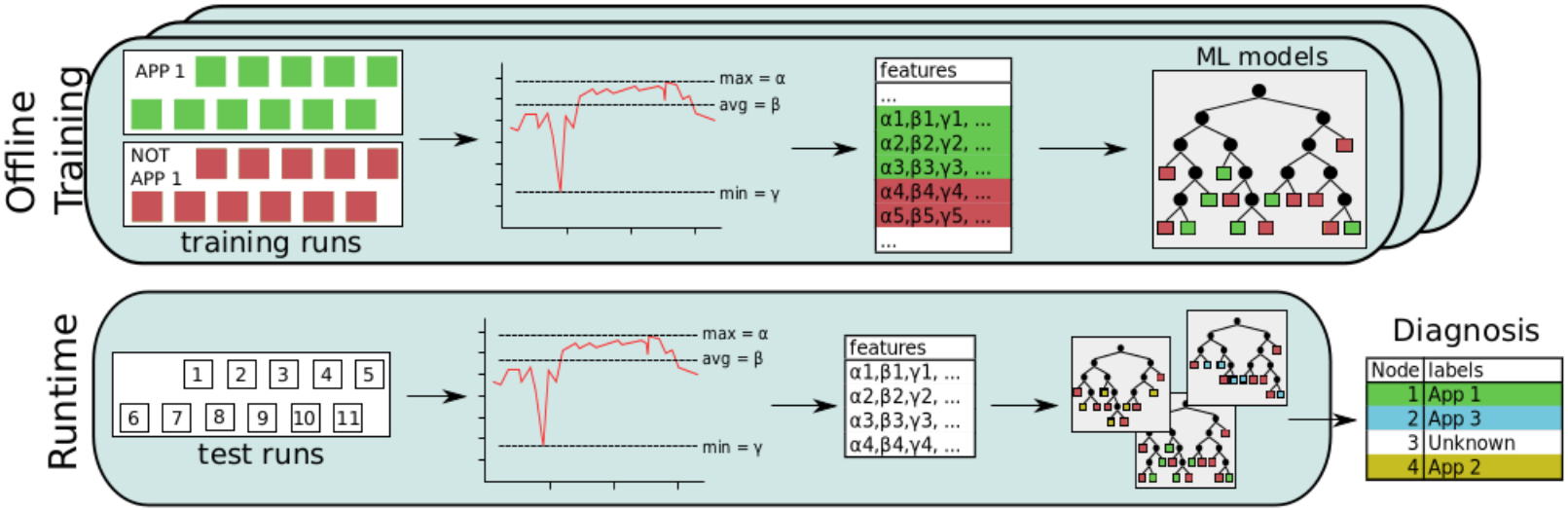


One vs. Rest Classifier

Observations from three classes



Taxonomist: An Application Detection Framework

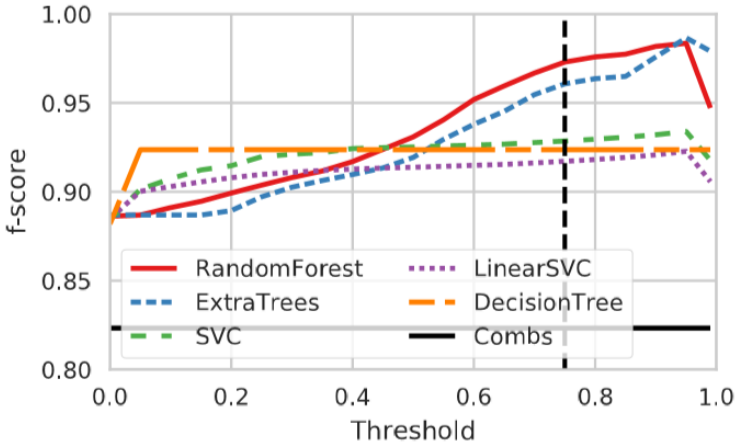


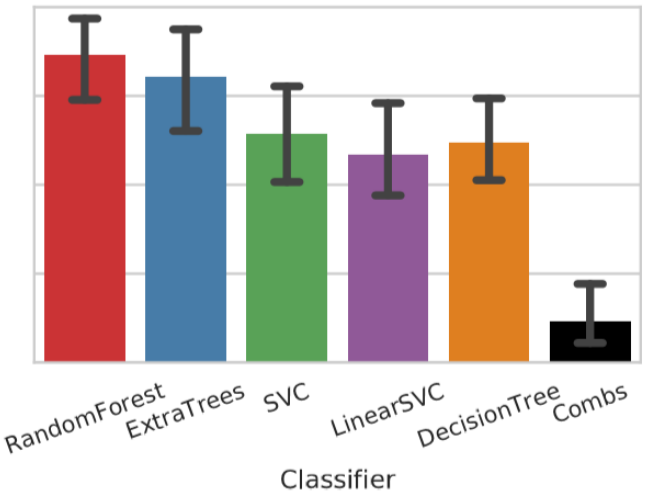
- At runtime, take predictions from every classifier
  - If confidence is under a threshold, mark as *unknown*

Evaluation

Trained with 10 applications, tested with 11 or more

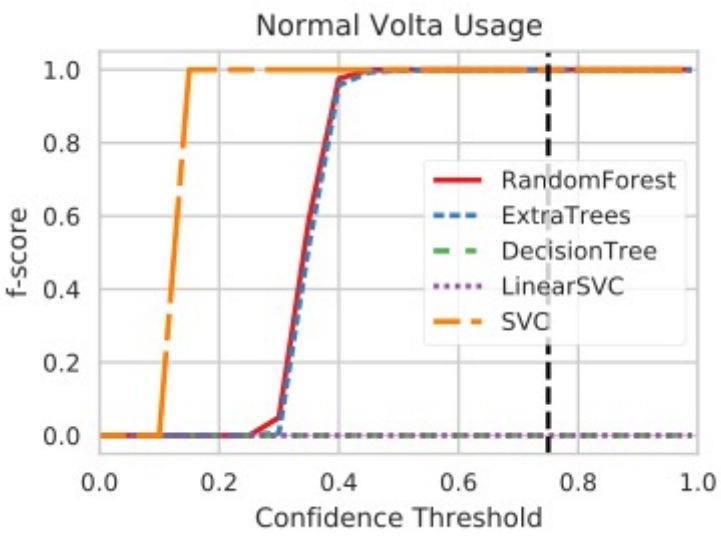
- All applications (including the unknown) correctly identified



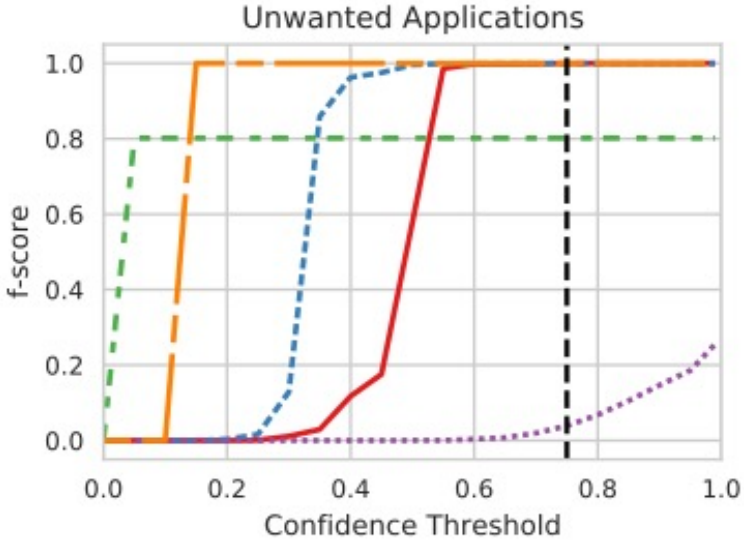


Evaluation

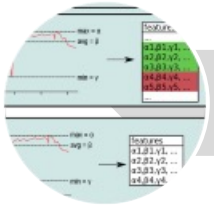
VOLTA USAGE OVER 6 MONTHS



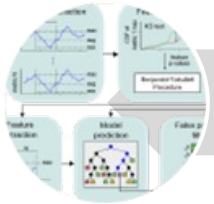
6 CRYPTOCURRENCY MINERS AND PASSWORD CRACKERS



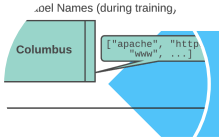
Outline



Diagnosing performance variations



Identifying applications



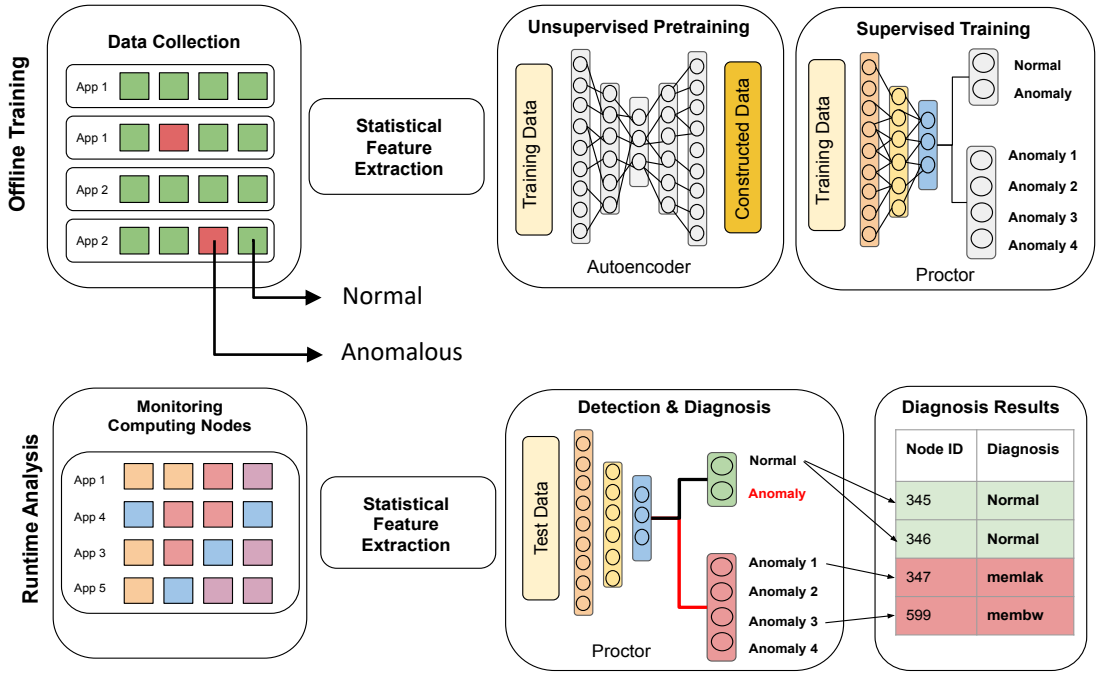
Towards real-world deployments

Proctor: A Semi-Supervised Performance Anomaly Diagnosis Framework

Operates with ***significantly less labeled data*** compared to baselines

Evaluation on a production HPC system and a testbed HPC cluster

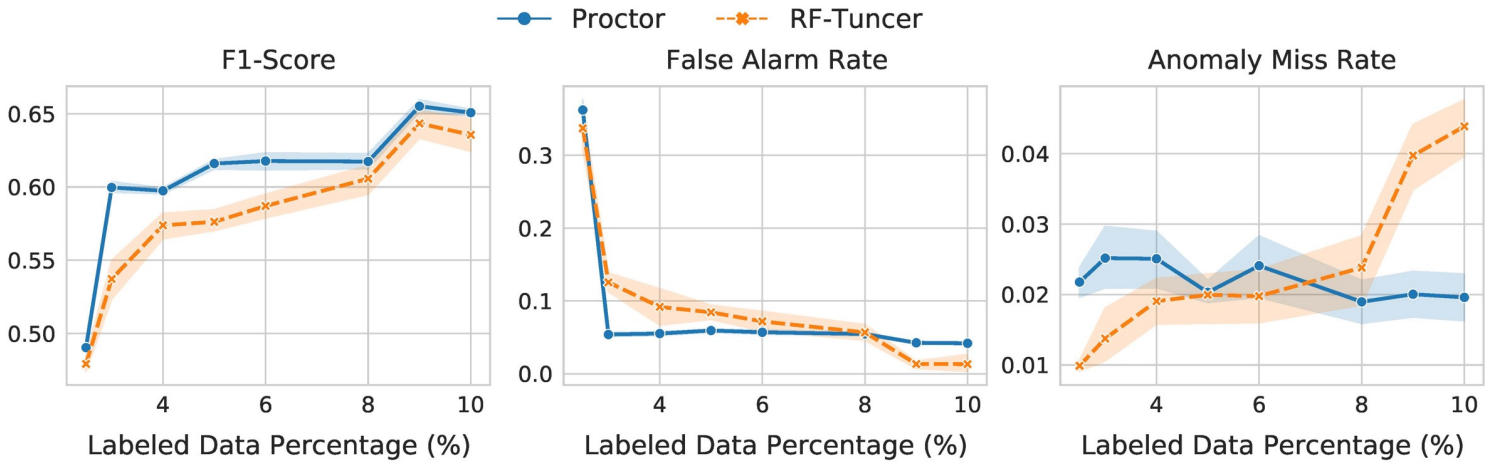
11% better F-score on average



Overview of the Proctor framework

[Aksar et al., ISC'21 - github.com/peaclab/Proctor]

Anomaly Diagnosis with Proctor

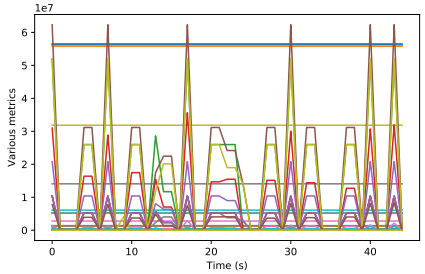


Proctor outperforms RF-Tuncer on average in F1-score

Maintains very low false alarm and anomaly miss rates

RF-Tuncer: Full supervised baseline [Tuncer et al., TPDS'18]

Can We Predict with Even Fewer Samples?



Query the label of a sample

Unlabeled Dataset

Add the sample to the labeled dataset

No

Initially labeled Dataset → Build a model → Evaluate the model on constant test set → Is my model accurate enough?

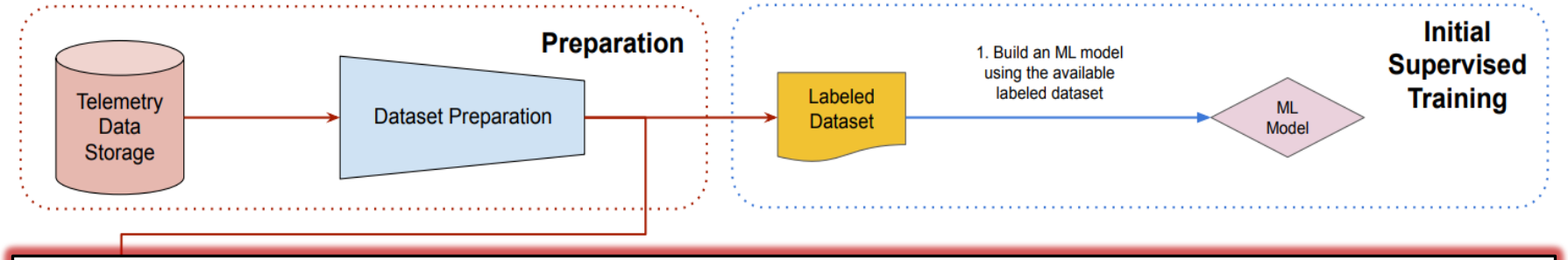
Yes

Terminate

Active Learning Based Anomaly Diagnosis

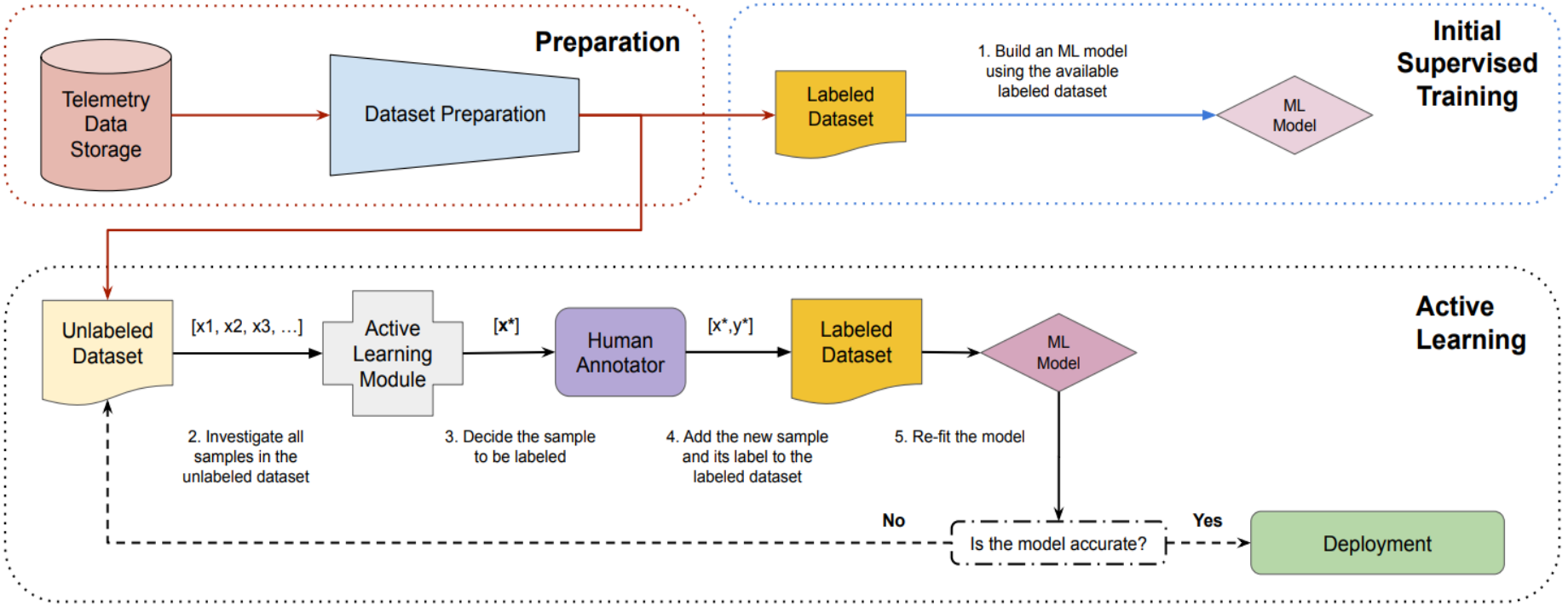
[Aksar et al., CLUSTER'22]

Active Learning Based Anomaly Diagnosis



- Scenario:
  - A few labeled and a large number of unlabeled samples
- Methodology:
  - Start with 1 sample from all application-anomaly pairs in the Eclipse and Volta datasets
    - 30 samples for Eclipse
    - 55 samples for Volta
  - Train a supervised ML model using labeled dataset & make predictions on test dataset

Active Learning Based Anomaly Diagnosis



Active Learning Based Anomaly Diagnosis

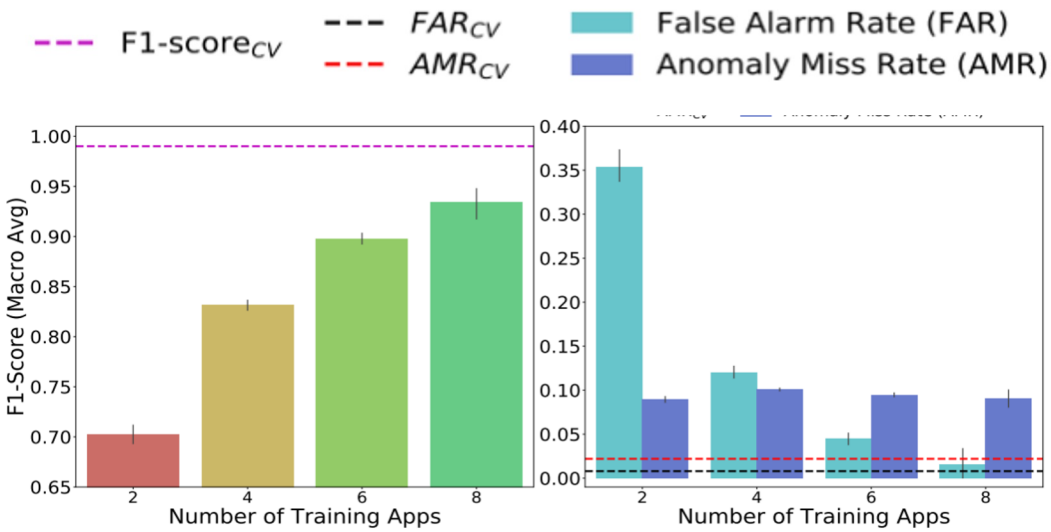
Goal: Show how many samples are necessary to achieve a target F1-score

Dataset	Feature Extraction	The Best Sampling Strategy	Initial Sample Count	Starting F1-score	Starting F1-score: 0.85	F1-score: 0.90	F1-score: 0.95	All Active Learning Training Data F1-score	Max Score (5 fold CV using whole data)
Volta	TSFRESH	Uncertainty	55	0.86	Already >.85	65 Samples	76 Samples	0.95 (6329 Samples)	0.99 (16732 Samples)
Eclipse	MVTS	Margin	30	0.72	57 Samples	150 Samples	230 Samples	0.95 (5619 Samples)	0.99 (19652 Samples)

Active learning achieves the same F1-score as a fully supervised approach using significantly **fewer** data samples

Evaluating Robustness

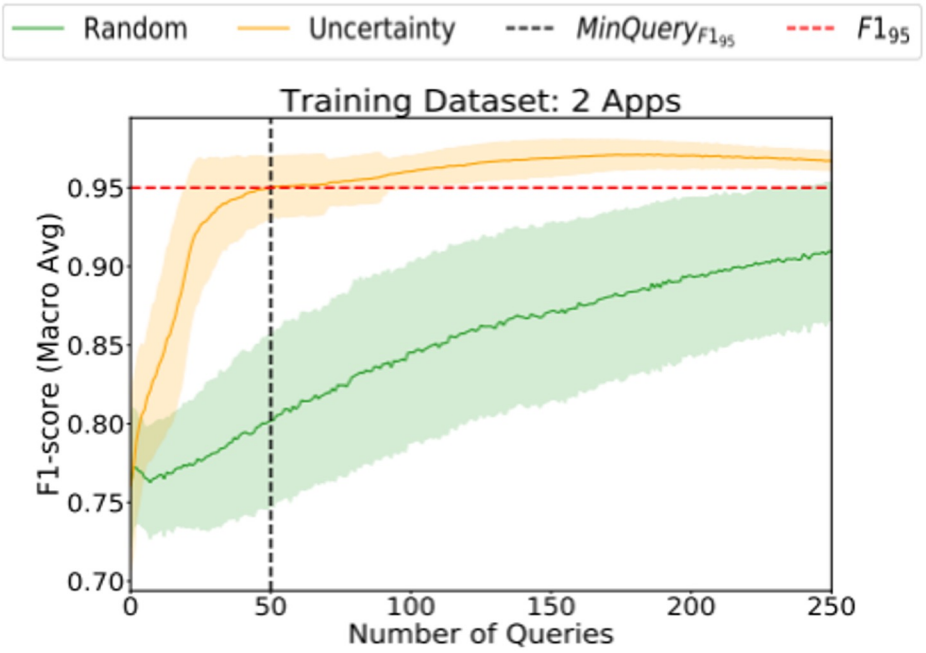
- Robustness:
  - Being able to diagnose anomalies with a small error margin with previously unseen applications and application inputs
- Compared to 5-fold CV results:
  - 30% drop in F1-score
  - 35x increase in FAR



Error bars: 95% confidence interval
Dashed lines: 5-fold CV results

Active Learning With Previously Unseen Apps

- Start with 2 applications in the training dataset, and place the remaining 9 applications to the test dataset
- Uncertainty achieves 0.95 F1-score using **50** additional samples
- Random baseline achieves 0.95 F1-score using **~550** samples

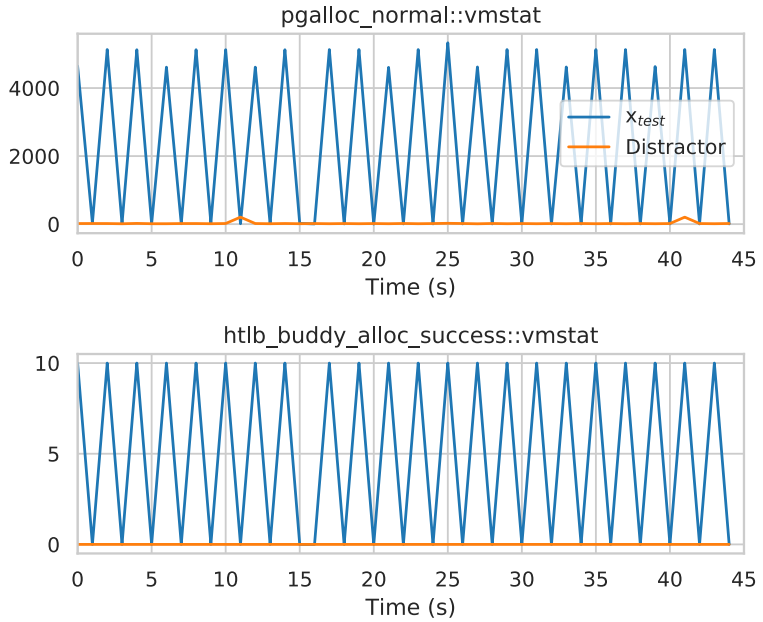


Counterfactual Explanations for Multivariate Time Series (CoMTE)

- Providing explanations is key for real world deployment
- CoMTE provides a small (2-3) number of samples to explain a prediction

2022 R&D 100 Award Finalist:

https://www.youtube.com/watch?v=sUV-gpaZvgg



X_{test}: Correctly classified sample with memory leakage anomaly

Distractor: Correctly classified sample without memory leakage anomaly

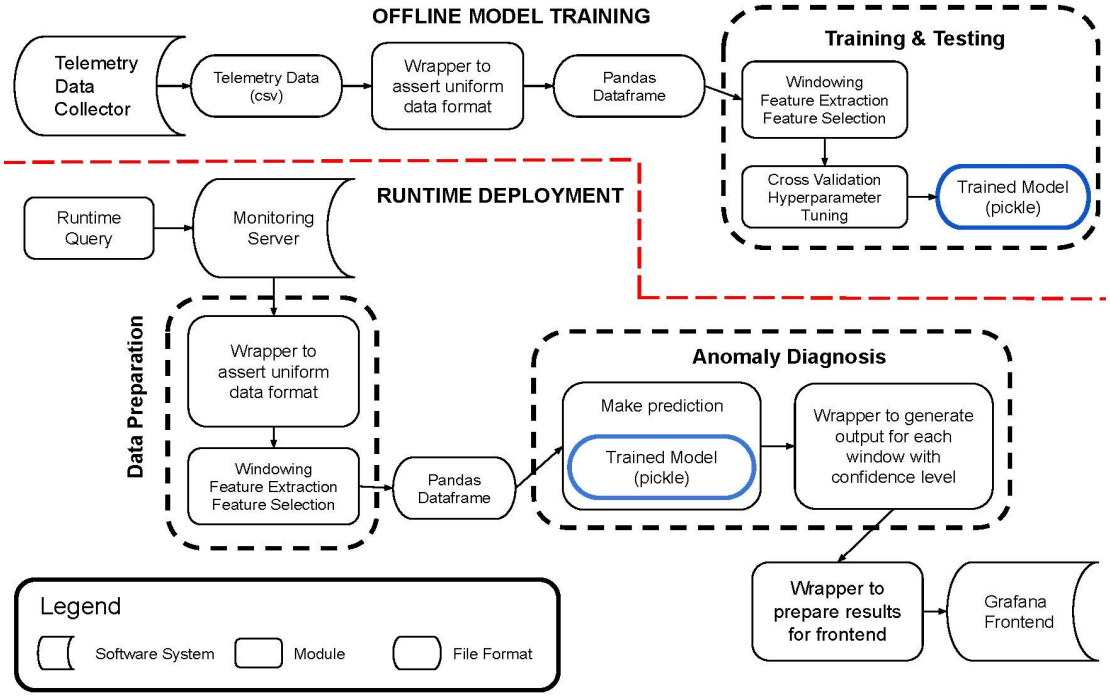
https://github.com/peaclab/CoMTE

[Ates et al., ICAPAI'21]

E2EWatch: An End-to-end Anomaly Diagnosis Framework for Production HPC Systems

Deployment of an end-to-end anomaly diagnosis framework on a 1488-node production HPC system

- Job and node-level analysis
- Delivers results in near-real time
- Customizable and interpretable visualization



Architecture of the E2EWatch framework

[Aksar et al., Euro-Par'21: github.com/peaclab/E2EWatch]

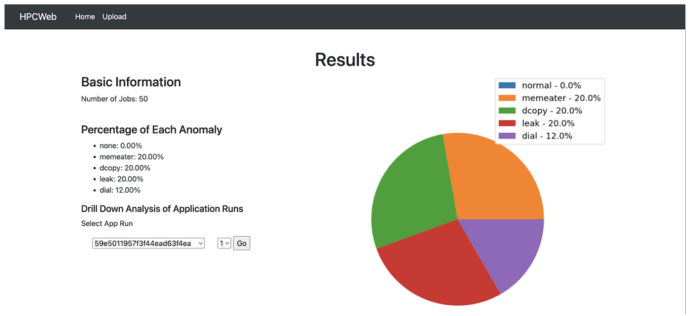
Deployment - Challenges

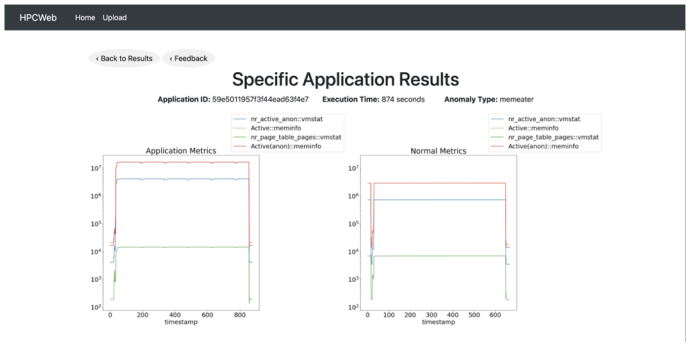
- Compiling applications properly with different input decks requires domain expertise
- Are synthetic anomalies are creating the desired impact?
- Deployment options:
  - Compute node-specific deployment:
    - High training and maintenance cost
      - Hyperparameter tuning
      - Selecting new anomaly threshold for each model
  - Generic deployment:
    - Developing a generic ML model that accurately detects/diagnose anomalies for thousands of compute nodes is hard

Upcoming paper at Supercomputing'23:
Unsupervised anomaly detection framework deployed on a supercomputer

Web Based Anomaly Diagnosis Framework

- Goal: Make our (supervised) ML-based anomaly diagnosis framework widely accessible
- Determines anomalies in user uploaded system telemetry data (or provided data)
- http://ai4hpc.bu.edu/



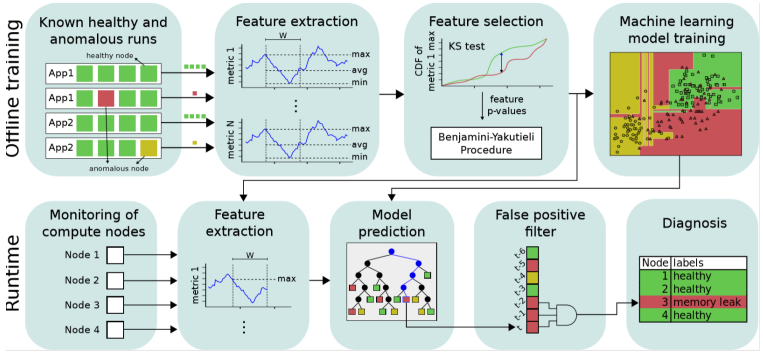


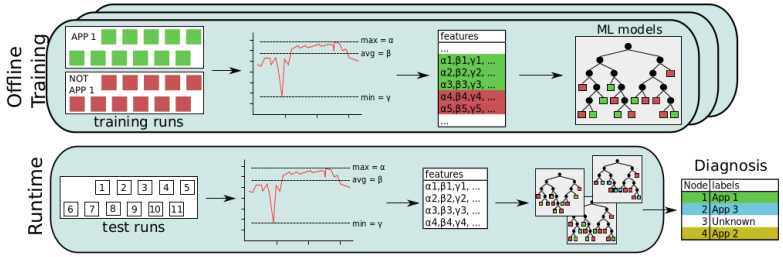
Some Take-Aways

Automated analytics for large scale systems
- **Numeric data**
- File/SW changes
- Systems events

Diagnosing performance variation/problems
- **System level**
- **Application level**
- **Practical solutions**

Application discovery
- At development
- At installation
- **During execution**



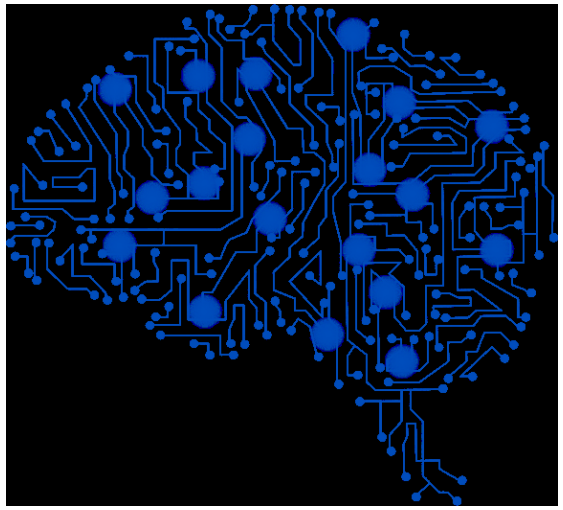


Vision:
ML to Improve and Automate HPC Management

Significant outages, major losses of profit, problems due to widespread vulnerabilities, … are all very common in computing systems

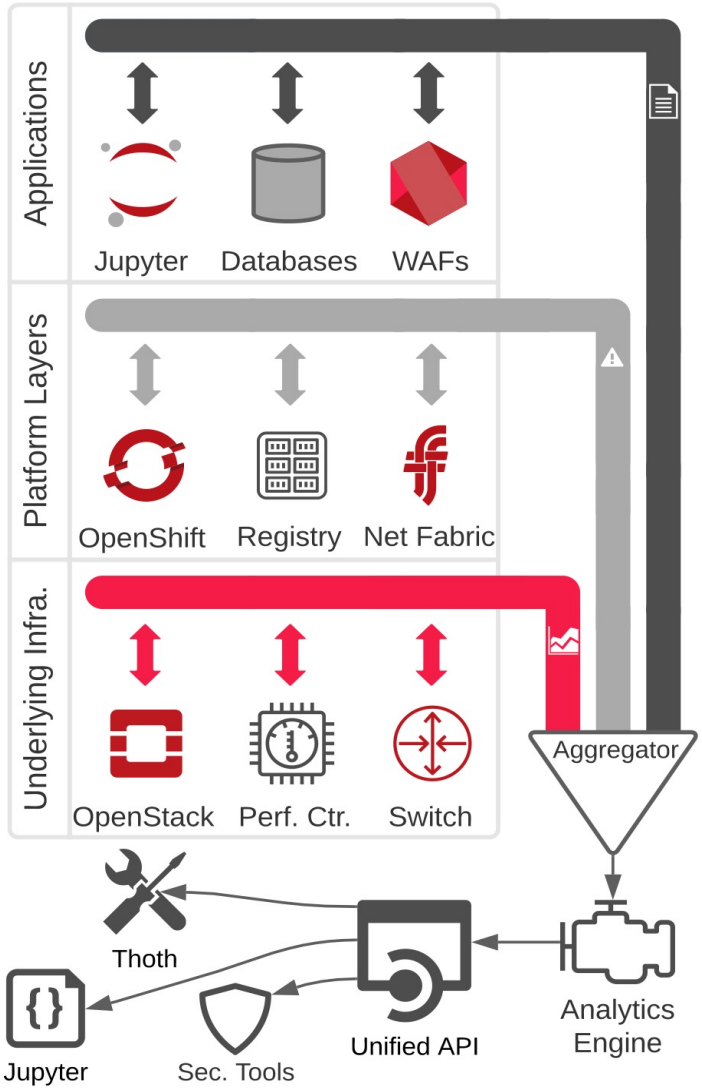
Automated analytics can tremendously help solve or pinpoint many important problems

Many open research problems exist in the design of data-driven management solutions



AI for Cloud Ops

- Modern cloud platforms are far too large/noisy for manual debugging
  - e.g., Kubernetes pod is crashing; parse through these 15 log files to figure out why
- Complex cloud problems require adaptive **cross-layer** diagnostic tools
- We're developing ML-powered tools to help cloud engineers maintain **performance**, **resilience**, and **security**



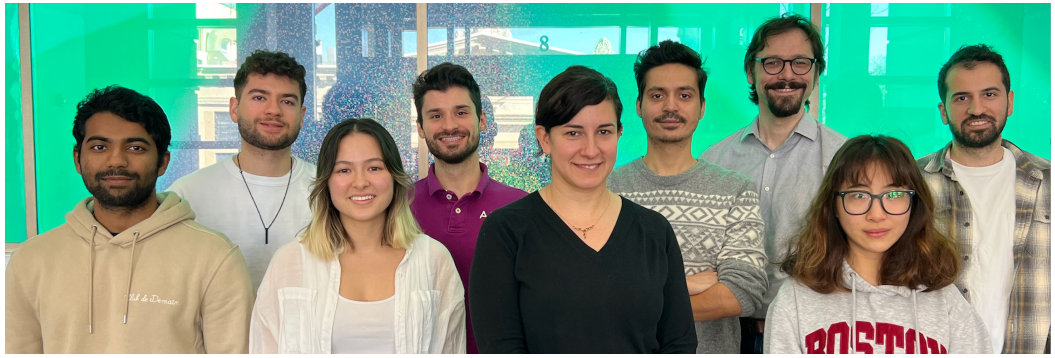
https://research.redhat.com/blog/research_project/ai-for-cloud-ops/

Performance and Energy Aware Computing Laboratory -- http://www.bu.edu/peaclab

Current graduate students:

Burak Aksar, Daniel Wilson, Efe Sencan,

Beste Oztop, Sina Karimi, Amin Khodaverdian



Former Research Scientist: Dr. Ata Turk

Alumni: Dr. Jie Meng, Dr. Can Hankendi, Dr. Hao Chen, Dr. Fulya Kaplan, Dr. Tiansheng Zhang, Dr. Ozan Tuncer, Dr. Onur Sahin, Dr. Emre Ates, Dr. Yijia Zhang, Dr. Aditya Narayan, Dr. Zihao Yuan, Dr. Prachi Shukla, Dr. Mert Toslali

Emika Hammond, Abin George, Carlton Knox, Allen Zou, Yanni Pang, Avery Huang, Cyril Saade, John Knollmeyer, Nathaniel Michener, Ann Lane, Katsu Kawakami, John Furst, Samuel Howes, Jon Bell, Benjamin Havey, Ryan Mullen, many others…

Recent Collaborators:

A. Joshi, M. Egele, Y. Paschalidis @ BU,

P. Vivet & Y. Thonnart @ CEA-Leti,

D. Atienza @ EPFL,

F. Oliveira, S. Nadgowda, S. Parthasarathy @ IBM TJ Watson,

V. Leung and J. Brandt @ Sandia Labs,

S. Reda @ Brown University,

A. Kahng @ UCSD,

E. Salman @ Stony Brook,

R. Sambasivan @ Tufts,

V. Pavlidis @ U. of Manchester,

E. Wang @ MIT.

Many masters students, especially:

Anthony Byrne, Charlie De Vivero, Josh Klein, …

Visitors: Dr. Marina Zapater, Dr. Andrea Bartolini

PeacLab GitHub: https://github.com/peaclab

Recent Funding & Collaborations