

AI for Cloud Ops

at the Boston University **Red Hat Collaboratory**

AYSE K. COSKUN

ALAN LIU

GIANLUCA STRINGHINI

February 16th, 2022



Red Hat

BOSTON
UNIVERSITY



Red Hatters: **Marcel Hild, Steven Huels, Daniel Riek**

IBM Researcher: **Fabio Oliveira**

PhD students: **Anthony Byrne, Mert Toslali, Saad Ullah, Lesley Zhou**



The Next Generation of Cloud Computing

- Most innovations in the cloud are motivated by **efficiency** and **scale**
- Automation reduces the need for expensive expert-dependent monitoring and management
- **Artificially-intelligent** cloud operations are the next logical step
 - Circumvents the slowest and most expensive element: the human operator
 - Recognizes known problems, makes resource management decisions to improve performance and efficiency, and many others

Why AI? Why now?

Machine Learning Models

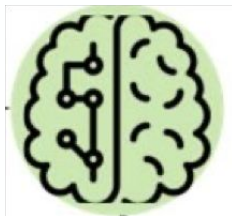
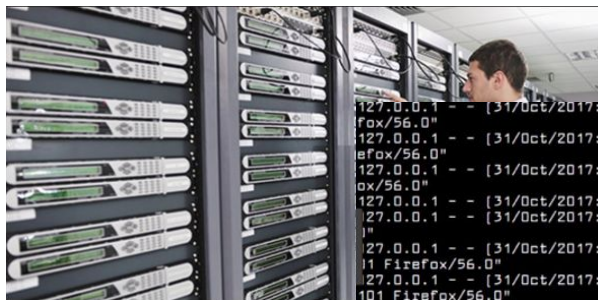


Image recognition
Speech recognition
Robotics
Navigation
... *many others*

Computer System Administration



```
127.0.0.1 - - [31/Oct/2017:11:11:37 +0530] "GET /icons/blank.gif HTTP/1.1" 200
Firefox/56.0"
127.0.0.1 - - [31/Oct/2017:11:11:37 +0530] "GET /icons/folder.gif HTTP/1.1" 200
Firefox/56.0"
127.0.0.1 - - [31/Oct/2017:11:11:37 +0530] "GET /icons/text.gif HTTP/1.1" 200 5
Firefox/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 /tecmin/ HTTP/1.1" 200 787 "ht
j"
127.0.0.1 - - [31/Oct/2017:11:12:05 +0530] "GET /icons/back.gif HTTP/1.1" 200 4
11 Firefox/56.0"
127.0.0.1 - - [31/Oct/2017:11:13:58 +0530] "GET /tecmin/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
a/20100101 Firefox/56.0"
```

File_Name: SIGFILES DKXA64_500500.SYS2

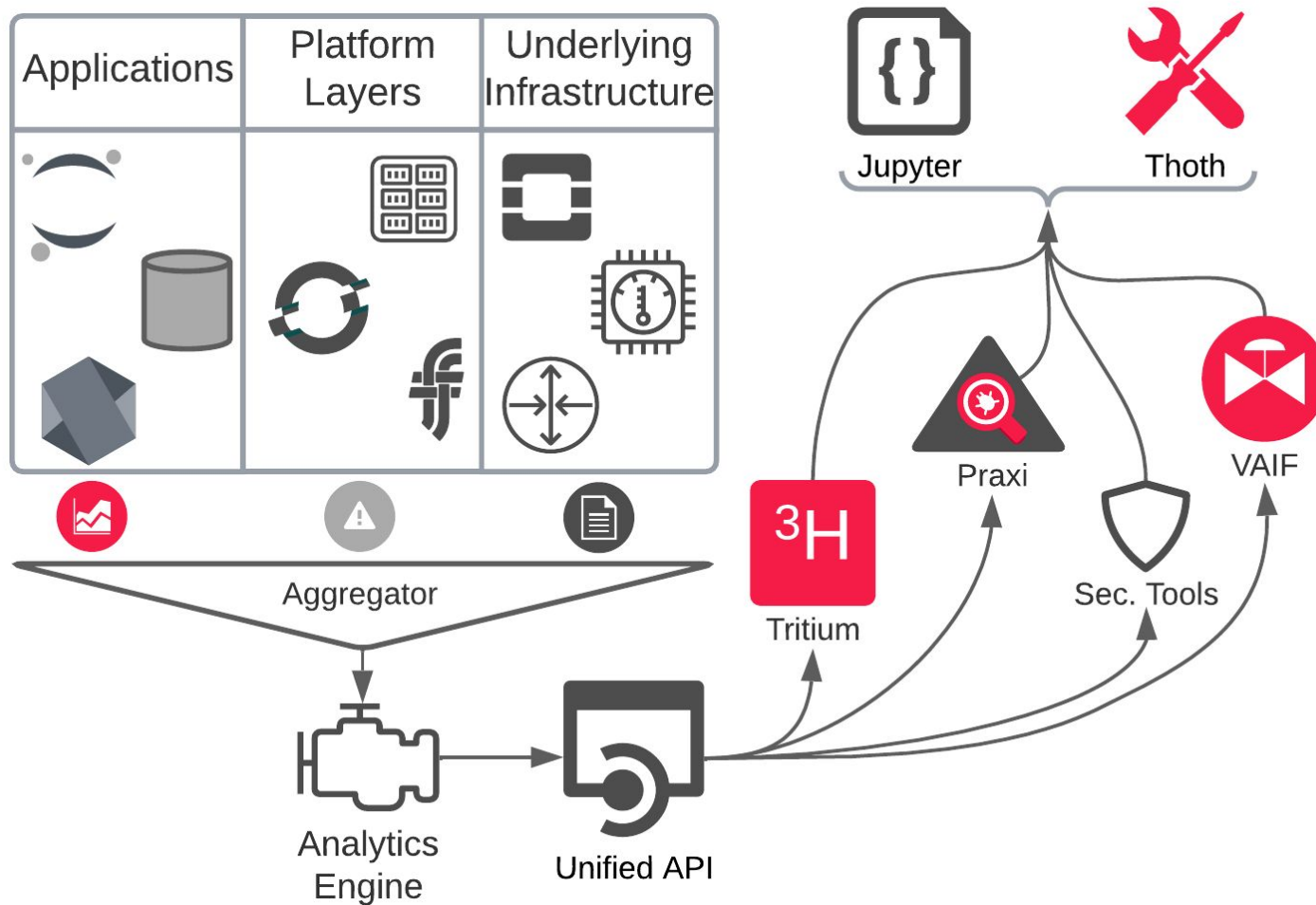
File_Size: ~~100 KB~~ 105KB



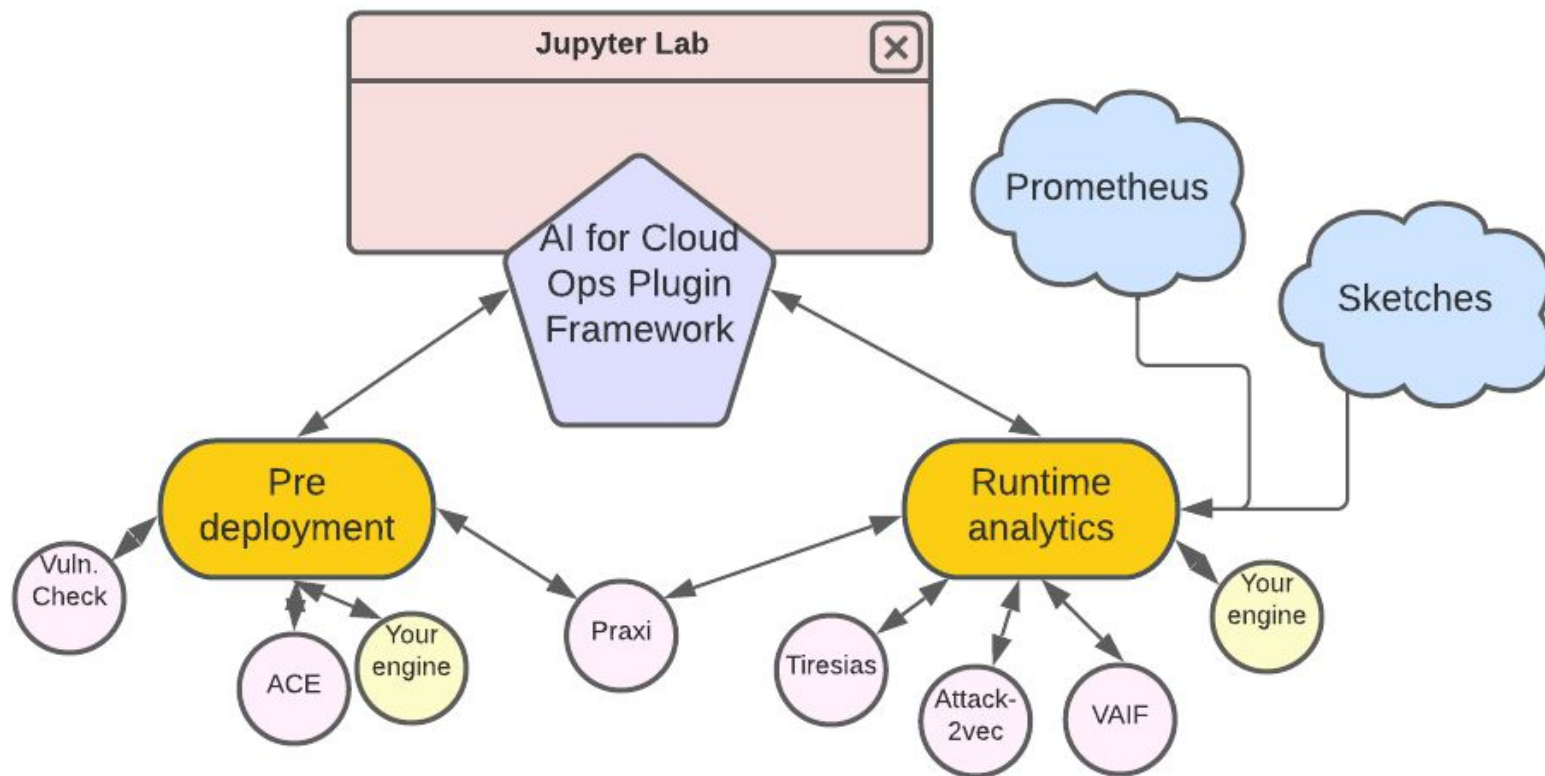
Our Vision

- Cloud computing has already grown beyond what manual operations methods can manage
- We're working on ways to ***automatically*** optimize efficiency and detect potential problems in cloud applications and infrastructure
- Our work will lead to a higher performance, more reliable, and secure future for cloud computing

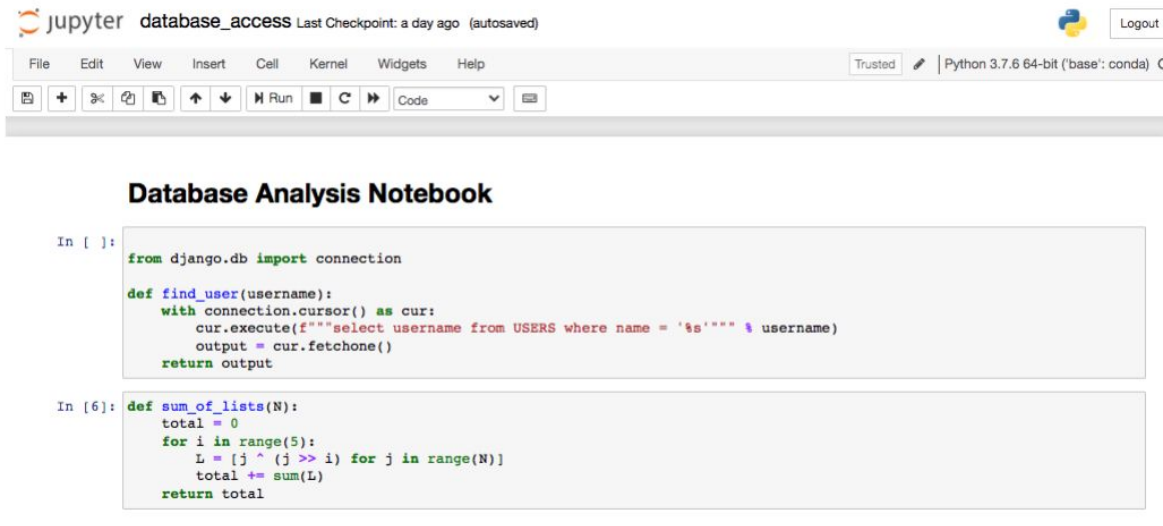
See how we're realizing this vision at: https://research.redhat.com/blog/research_project/ai-for-cloud-ops/
and sign up on: <https://github.com/operate-first/ai-for-cloud-ops/>



An easily-extensible, alive framework for the community



Code analysis
and SW
discovery
before
deployment



jupyter database_access Last Checkpoint: a day ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3.7.6 64-bit ('base': conda)

Database Analysis Notebook

```
In [ ]: from django.db import connection

def find_user(username):
    with connection.cursor() as cur:
        cur.execute(f"select username from USERS where name = '%s'" % username)
        output = cur.fetchone()
    return output

In [6]: def sum_of_lists(N):
        total = 0
        for i in range(5):
            L = [j ^ (j >> i) for j in range(N)]
            total += sum(L)
        return total
```


Code analysis
and SW
discovery
before
deployment

jupyter database_access Last Checkpoint: an hour ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Database Analysis Notebook

```
In [ ]: from django.db import connection

def find_user(username):
    with connection.cursor() as cur:
        cur.execute(f"select username from USERS where name = '%s'" % username)
        output = cur.fetchone()
    return output
```

M7065: Outdated module version in use; click [here](#) to upgrade via pip

W1041: Potential SQL injection vulnerability detected

```
In [ ]: def sum_of_lists(N):
        total = 0
        for i in range(5):
            L = [j ^ (j >> i) for j in range(N)]
            total += sum(L)
        return total
```

P8002: Performance bottleneck detected; showing profiling analysis below

Performance Bottleneck Analysis (Line 4)

Diagnosis: Bulk of execution time spent resolving list comprehension inside `sum_of_lists`

14 function calls in 0.714 sec; peak memory: 100.08 MiB; increment: 61.36 MiB

Details:

Ordered by: internal time

ncalls	tottime	percall	cumtime	percall	filename:lineno(function)
5	0.599	0.120	0.599	0.120	<ipython-input-19>:4(<listcomp>)
5	0.064	0.013	0.064	0.013	{built-in method sum}
1	0.036	0.036	0.699	0.699	<ipython-input-19>:1(sum_of_lists)
1	0.014	0.014	0.714	0.714	<string>:1(<module>)
1	0.000	0.000	0.714	0.714	{built-in method exec}

Code analysis
and SW
discovery
before
deployment

jupyter database_access Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Database Analysis Notebook

```
In [ ]: from django.db import connection

def find_user(username):
    with connection.cursor() as cur:
        cur.execute(f"select username from USERS where name = '%s'" % username)
        output = cur.fetchone()
    return output
```

M7065: Outdated module version in use; click [here](#) to upgrade via pip

W1041: Potential SQL injection vulnerability detected

```
In [ ]: def sum_of_lists(N):
        total = 0
        for i in range(5):
            L = [j ^ (j >> i) for j in range(N)]
            total += sum(L)
        return total
```

P8002: Performance bottleneck detected; showing profiling analysis below

Performance Bottleneck Analysis (Line 4)

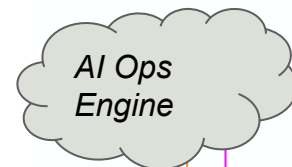
Diagnosis: Bulk of execution time spent resolving list comprehension inside `sum_of_lists`

14 function calls in 0.714 sec; peak memory: 100.08 MiB; increment: 61.36 MiB

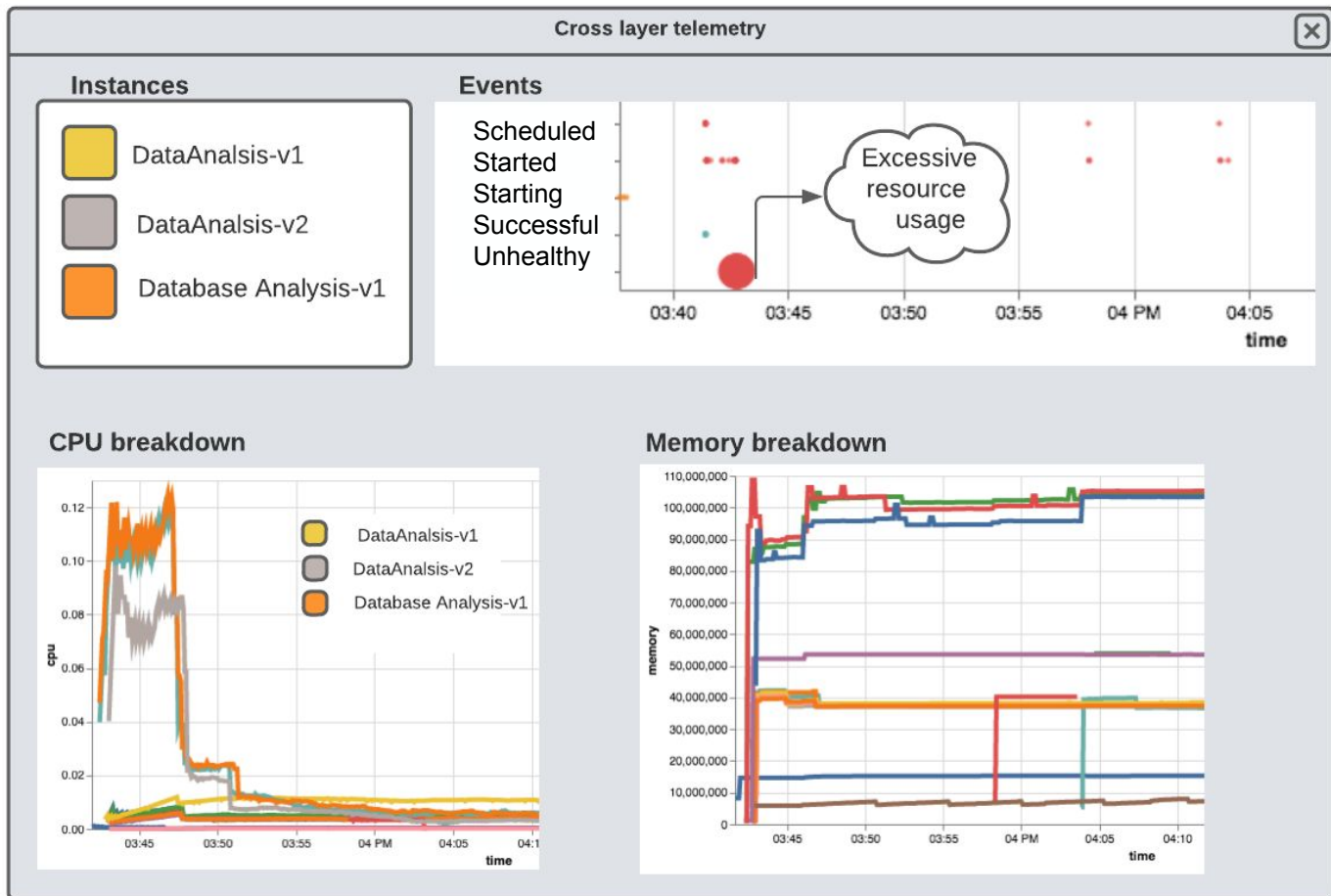
Details:

Ordered by: internal time

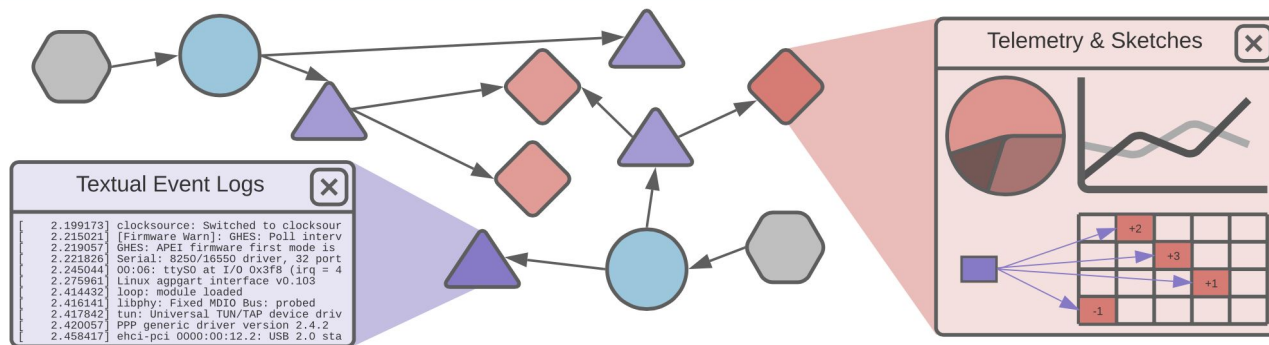
ncalls	tottime	percall	cumtime	percall	filename:lineno(function)
5	0.599	0.120	0.599	0.120	<ipython-input-19>:4(<listcomp>)
5	0.064	0.013	0.064	0.013	{built-in method sum}
1	0.036	0.036	0.699	0.699	<ipython-input-19>:1(sum_of_lists)
1	0.014	0.014	0.714	0.714	<string>:1(<module>)
1	0.000	0.000	0.714	0.714	{built-in method exec}



Cross-layer data fusion and analytics *at runtime*



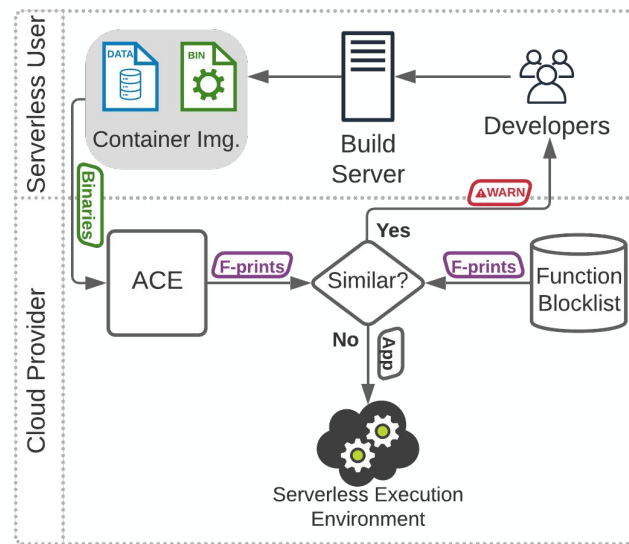
Cross-layer data fusion and analytics at runtime



Runtime Analytics						
Anomalous Runtime Behavior Detected in Instance vd-002.openshiftapps.com						
Example Trace Details						
Request	Method	Hits	Err. Rate	.NET Runtime	Database Runtime	Apache Runtime
test.HttpClientAsync	GET	370	0.0%	0.015s	-	0.343s
test.WebClientAsync	GET	354	10.8%	0.013s	-	0.325s
test.DbTestAsync	POST	368	1.0%	0.029s	0.165s	-

Automated Cloud Software Discovery

- Cloud apps become more complex over time
 - Eventually, manual software audits become impractical
- **Praxi** is a fully-automated, machine-learning-based method of discovering unwanted cloud software
 - Achieves >96% accuracy 14x faster than prev. method
- **ACE** discovers unwanted functions/libraries in opaque binary cloud software before execution
 - Detects known-vulnerable functions with 99% accuracy
 - 5.2x faster than prev. method and no model training req'd



ACE applied to serverless software component discovery

VAIF: Variance-driven Automated Instrumentation Framework

- A logging framework that automatically enables the logs needed to diagnose performance problems
- *Insight*: requests with similar critical paths will have similar response times
 - If not, this behavior may represent performance problems
- *Enabler*: distributed tracing
 - Provides discriminate context needed for effective diagnosis

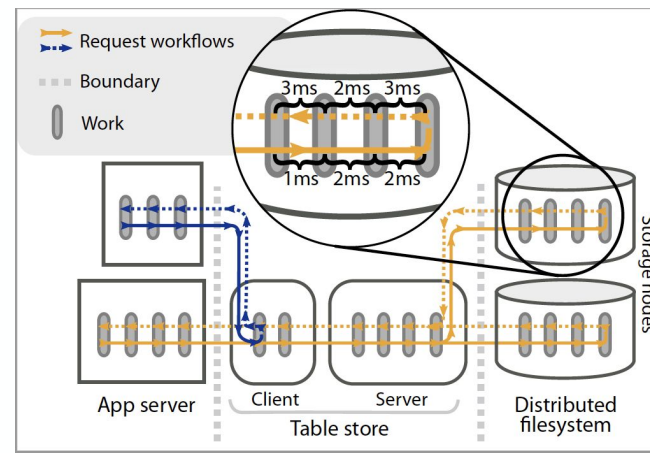
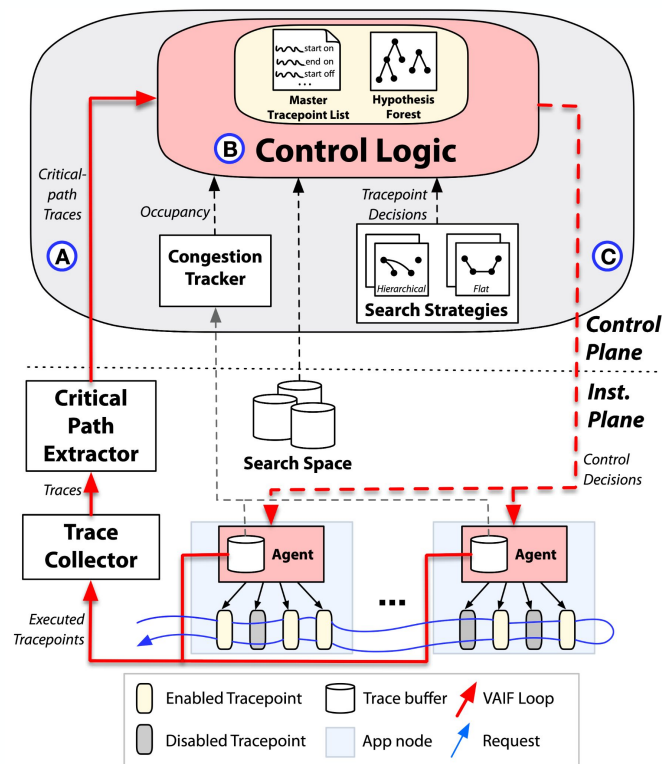
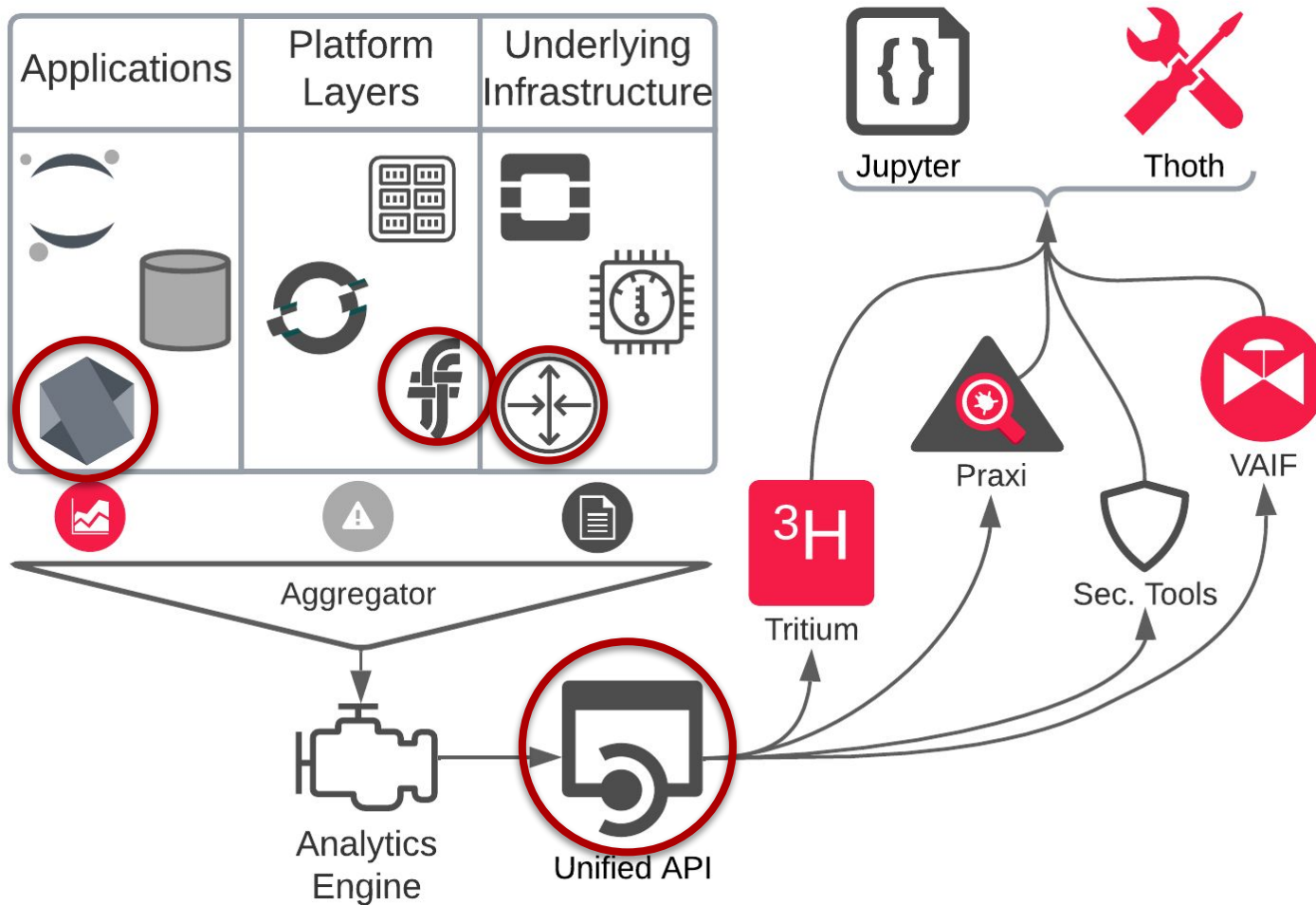


Illustration of distributed tracing

VAIF adjusts tracing in response to performance problems

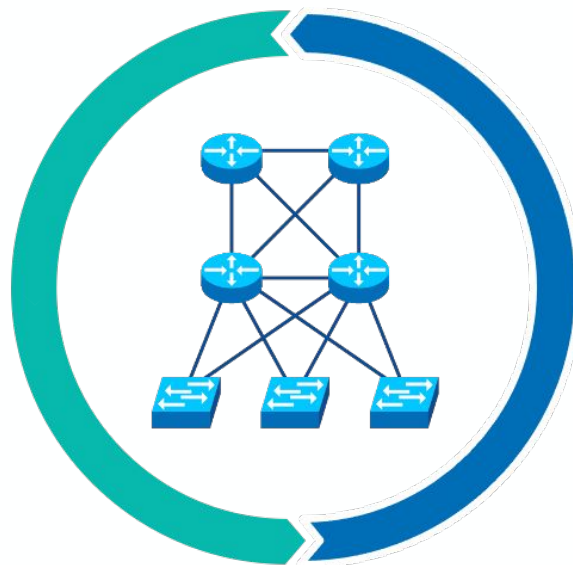
- Automatically enriches traces with additional *tracepoints* to localize problems
- Only enables 3-37% of all possible tracepoints to localize problems in HDFS, Openstack, and DeathStarBench applications
 - Problems include slow code paths, resource contention, and problematic third-party code





Run-Time Network Analytics

Why Run-Time Network Analytics?

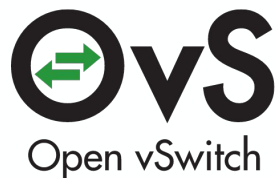


AI-based Control:
Performance Evaluation
Failure Diagnosis
Security Analysis

...

Closed-loop control

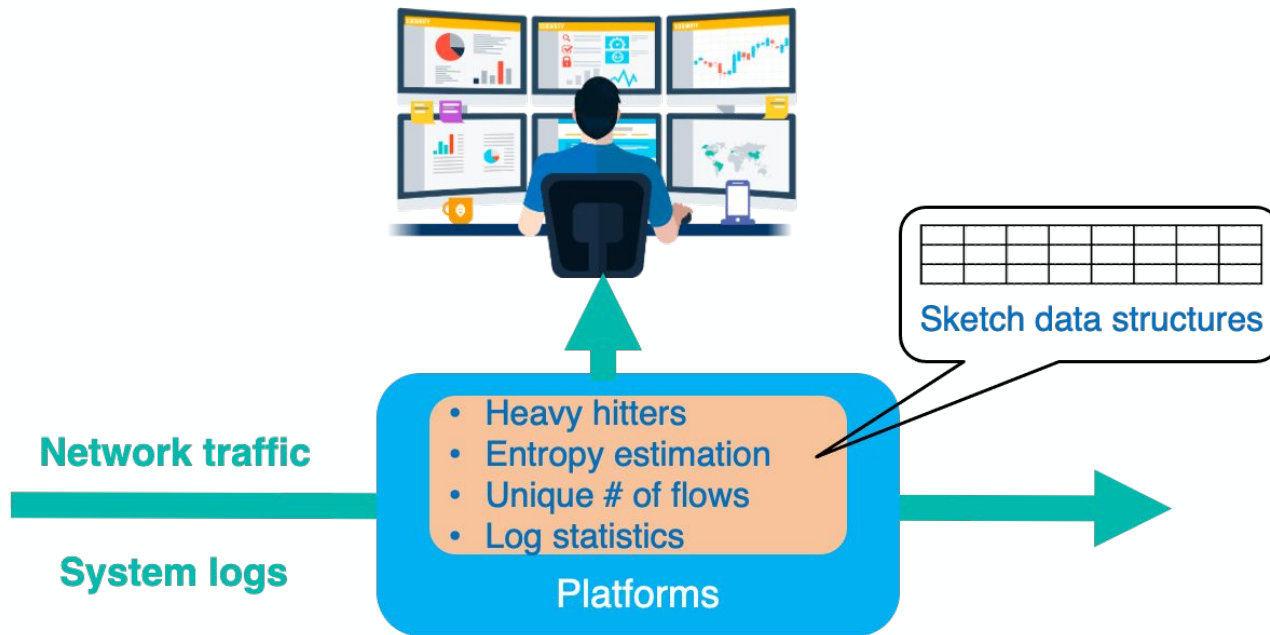
Performing Cross-Layer Network Analytics



- High latency?
- Low throughput?
- Under attack?
- Network failures?

- **Idea:** Cross-layer, fine-grained online analytics engine via sketches to obtain real-time visibility from the network.
- **New:** Does not require offline analysis like NetFlow/sFlow.

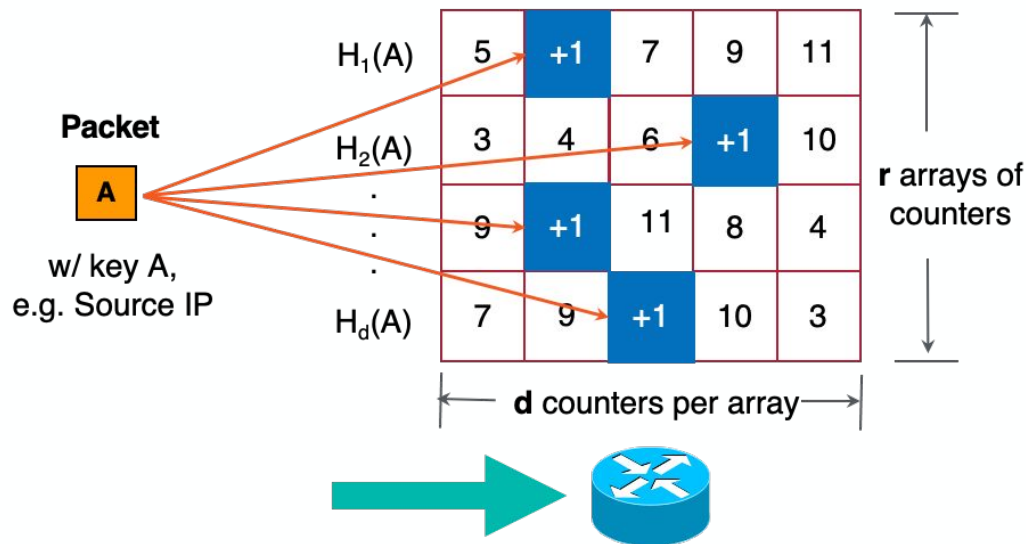
Sketch-based Network Analytics



- Lightweight, online sketch data structures to perform various analytical tasks.
- Collect and analyze sketches from different platforms to obtain aggregated results.

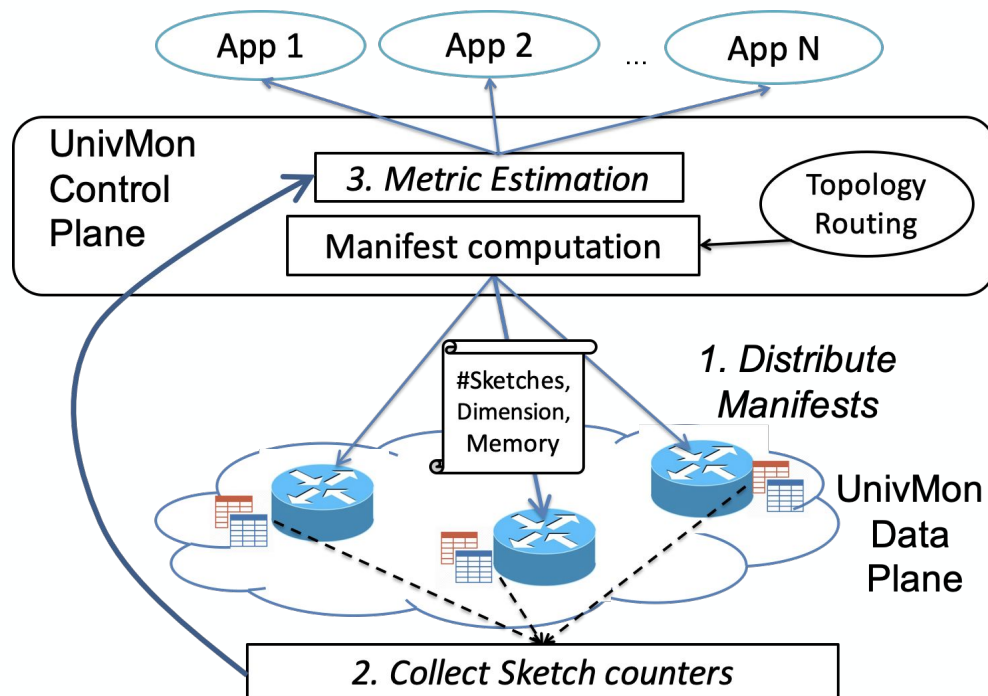
How Simple Sketches work?

Estimate Heavy Hitter Flows (HH)



- High-accuracy, low resource usage with theoretically and empirically proven accuracies.

UnivMon sketch for estimating various metrics



Control plane

- Metric estimation
- Monitoring configuration

Data Plane

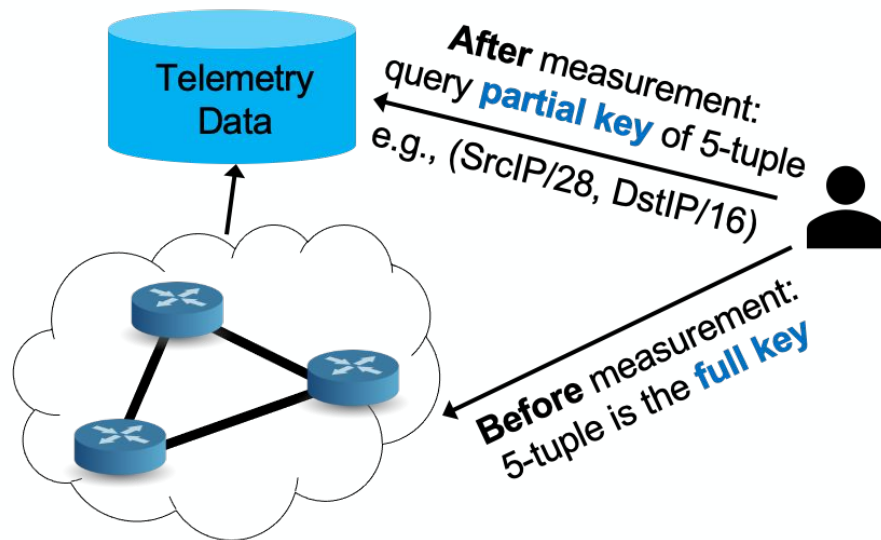
- Multi-data collection
- Sketch counter updates

[Liu et al., SIGCOMM'19]

[Liu et al., SIGCOMM'16]

- For example, **600KB** can estimate a broad range of metrics (e.g., entropy, heavy hitters, histogram) with **>98%** accuracy, processing over **16** million records.

CocoSketch: Multidimensional Queries in the Network

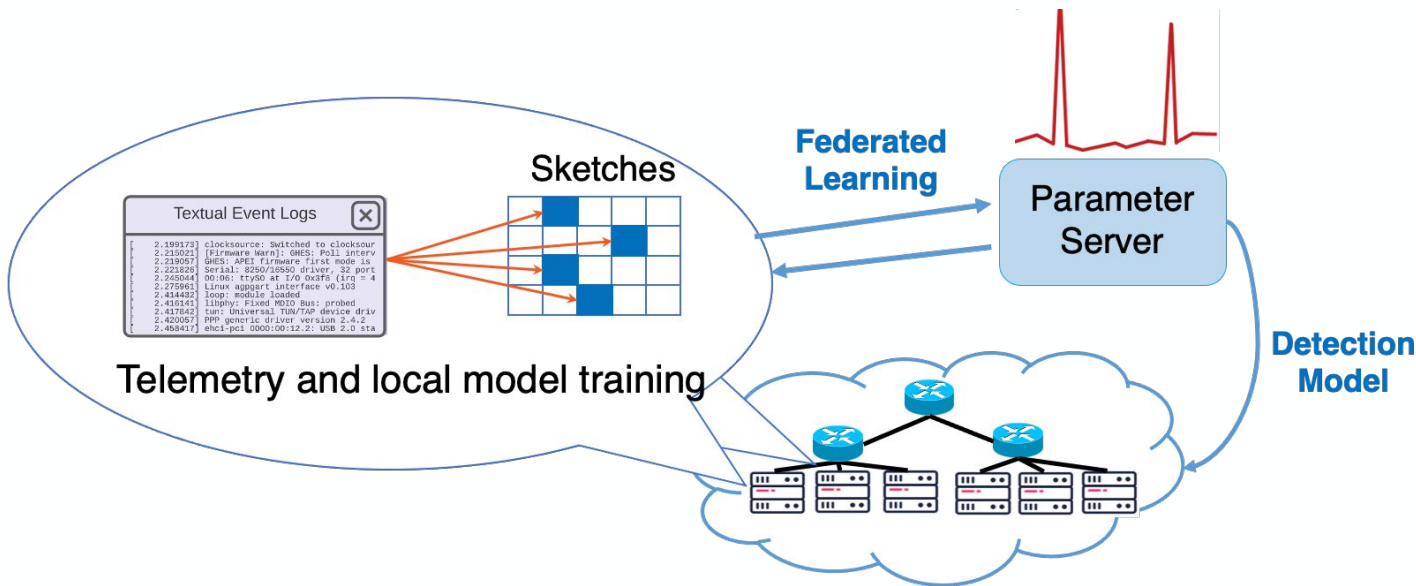


Key benefits

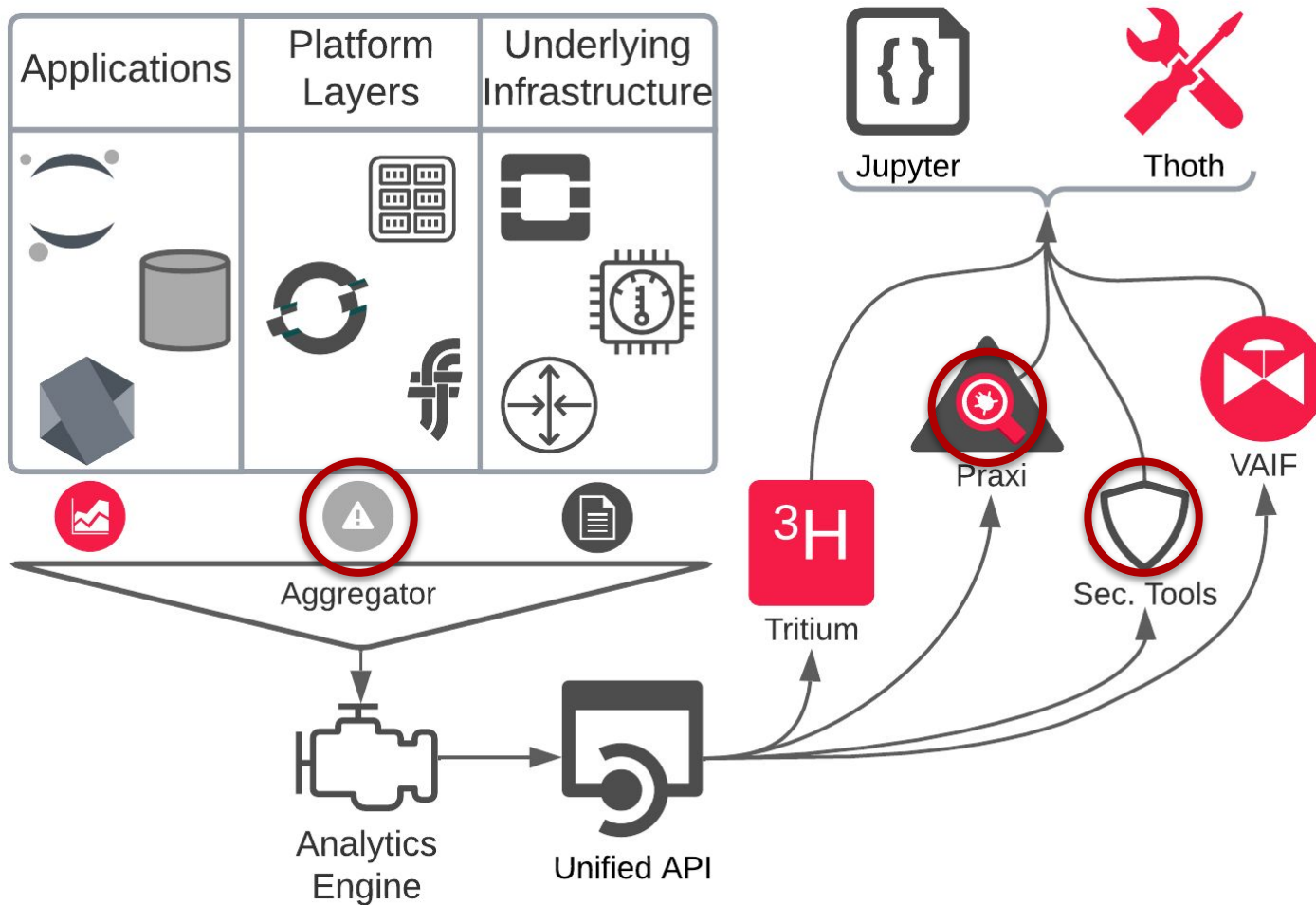
1. Support queries on **multiple** keys.
2. **Without** the need to predefine each key.

- **Key**: any combination of packet-header fields.
- **5-tuple** (SrcIP, DstIP, SrcPort, DstPort, Protocol).
- E.g., can we query the SrcIPs that are sending a lot of traffic to many DstIPs? (**Potential malicious user**)

Failure Detection via Sketch-based Analytics



- Fast failure detection model training via sketches (e.g., network failures).
- Real-time, sketch-based model inference (using sketches as input).



AI Analytics for Security

AI Analytics for Security

Two threat models:

- The users of Jupyter notebooks make mistakes, use insecure modules, and introduce security vulnerabilities that could be exploited by others (e.g., through malicious files)
- A malicious user might write malicious code in a Jupyter notebook to compromise the security of the cloud infrastructure (e.g., perform a DoS attack)

Idea: collect telemetry events generated by Jupyter Lab notebooks, use them to train AI models, and use these models to detect potential attacks or predict and mitigate them ahead of time

What does vulnerable/malicious code look like?

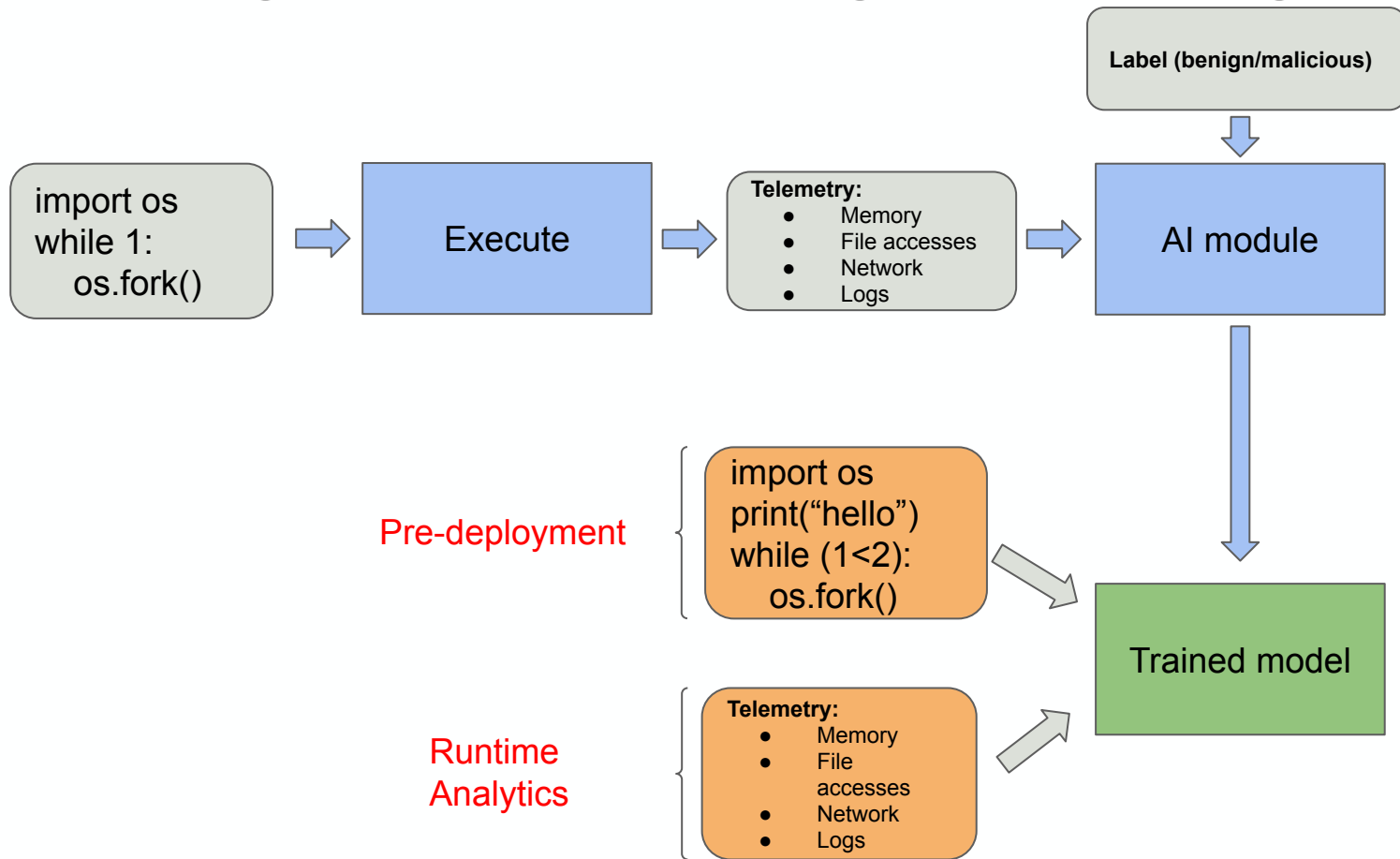
Known vulnerable/insecure modules or functions:

- **Code injection:** eval
- **Untrusted data serialization:** pickle.load
- **Insecure cryptography/network functions**

Insecure/vulnerable/malicious code blocks

```
import os
while 1:
    os.fork()
```

Predicting and understanding attacks using AI



Tiresias - predict upcoming attacks using IPS logs

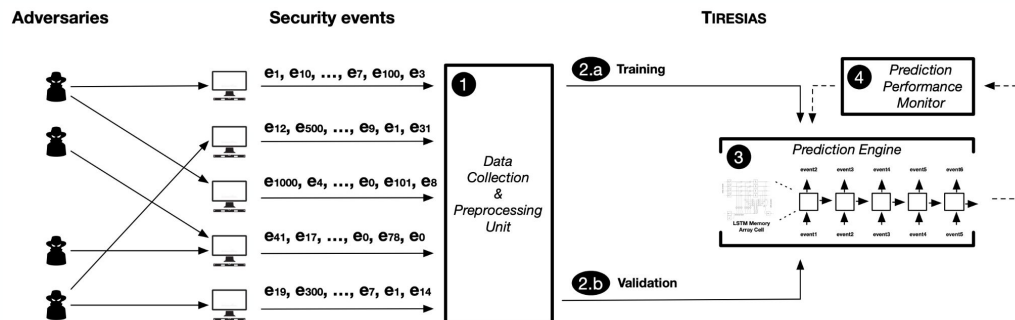
Idea: use sequences of attack steps (i.e., IPS logs)
to train a LSTM model

Use this model to predict the likely
action that an attacker will take next

Goal: develop proactive defenses

4,495 possible security events (IDS alerts)

We make the correct prediction 85% of the time



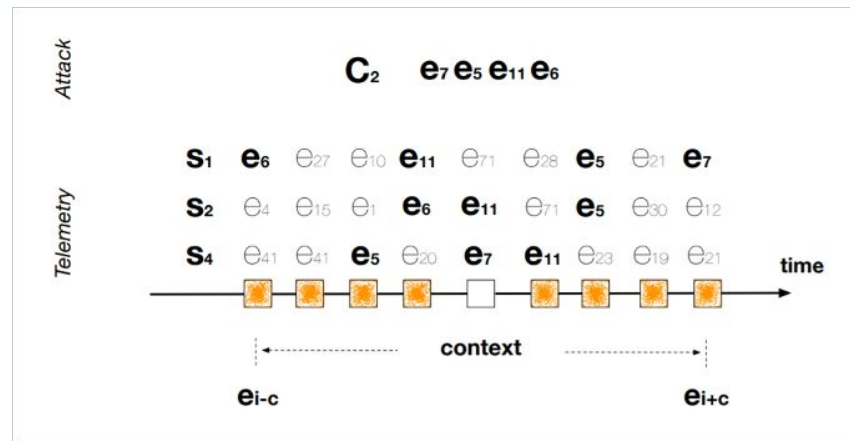
How can we apply these ideas to the cloud?

- Leverage the telemetry generated by the execution of code blocks in Jupyter notebooks to train a model (e.g., LSTM)
- **Pre-deployment mode**: match code blocks written by the user to previously observed ones and alert the user if past code was deemed unsafe/insecure
- **Runtime analytics**: monitor the execution of code blocks and deploy mitigations (kill process, rate limit, ...) if the model believes that the code is about to perform malicious actions

attack2vec - understanding emerging attack trends

Most modern attacks are multi-step

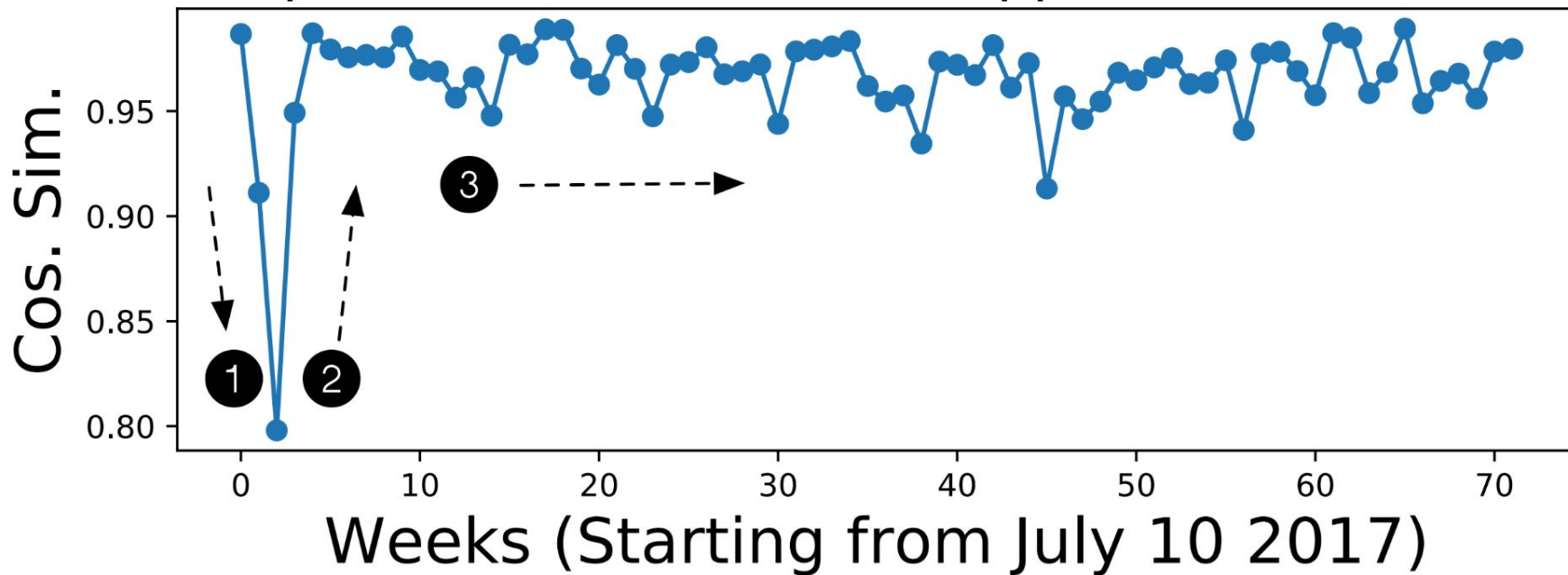
Idea: treat single attack steps as words in a sentence



Train vector embedding models to learn how single steps (e.g., the exploitation of a CVE) are used as part of more complex attacks

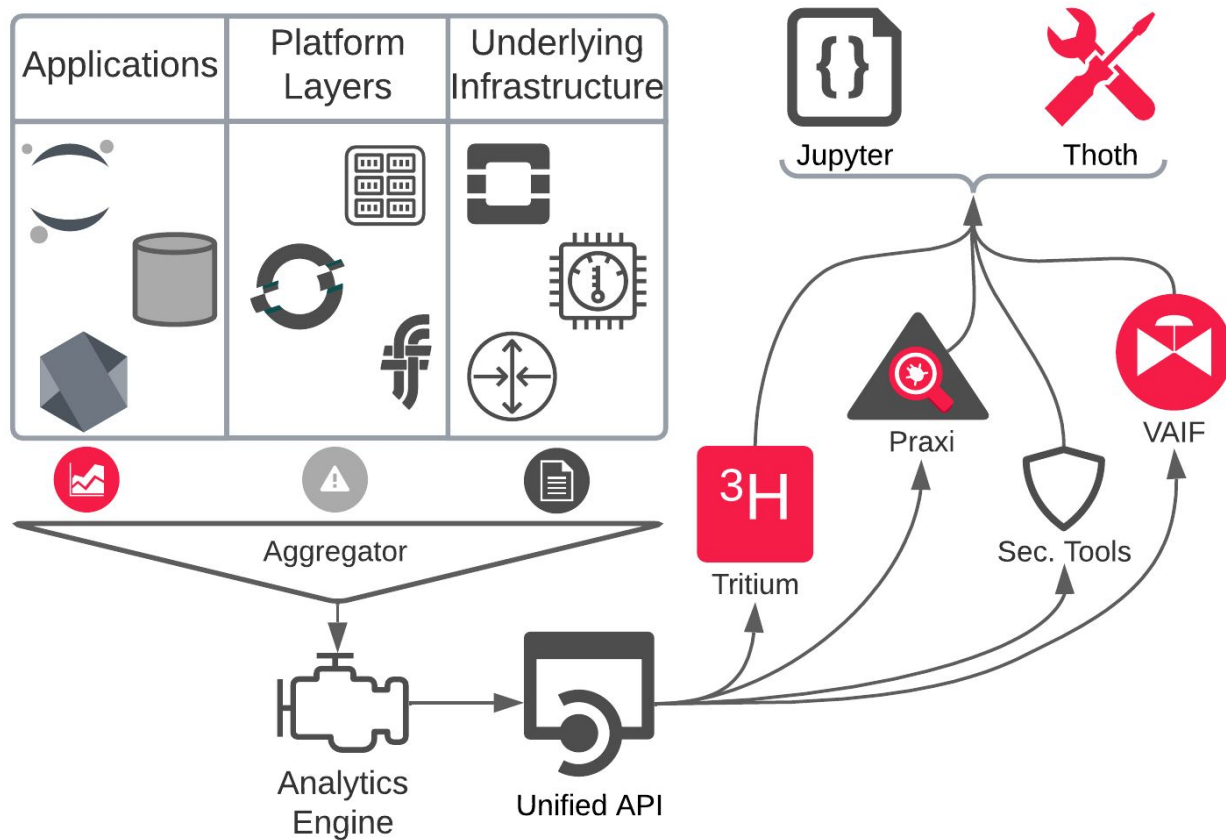
A steep change in the vector embedding for a certain attack step indicates that a new attack strategy has emerged

Apache Struts Showcase App CVE-2017-9791



How can we apply these ideas to the cloud?

- Match code blocks to previously observed ones
- Analyze the **context** in which these code blocks are executed, especially malicious ones
- **Situational awareness**: warn cloud operators when malicious code blocks are used in a new way, as a new defense strategy might be needed



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

Sign up on: <https://github.com/operate-first/ai-for-cloud-ops/>