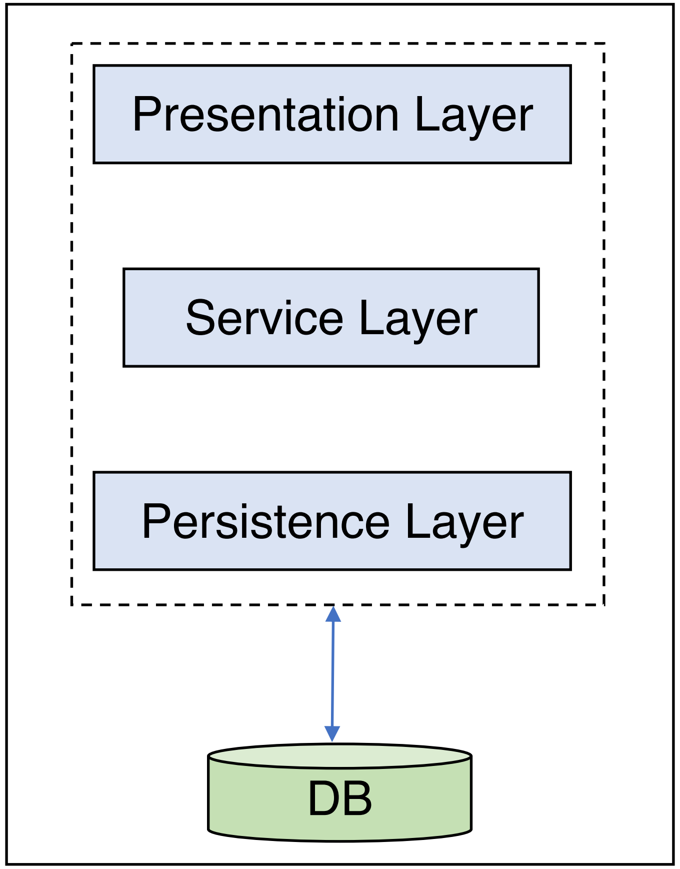
A Case For Cross-Domain Observability to Debug Performance Issues in Microservices

Ranjitha K, Praveen Tammana, **Pravein Govindan Kannan**, Priyanka Naik

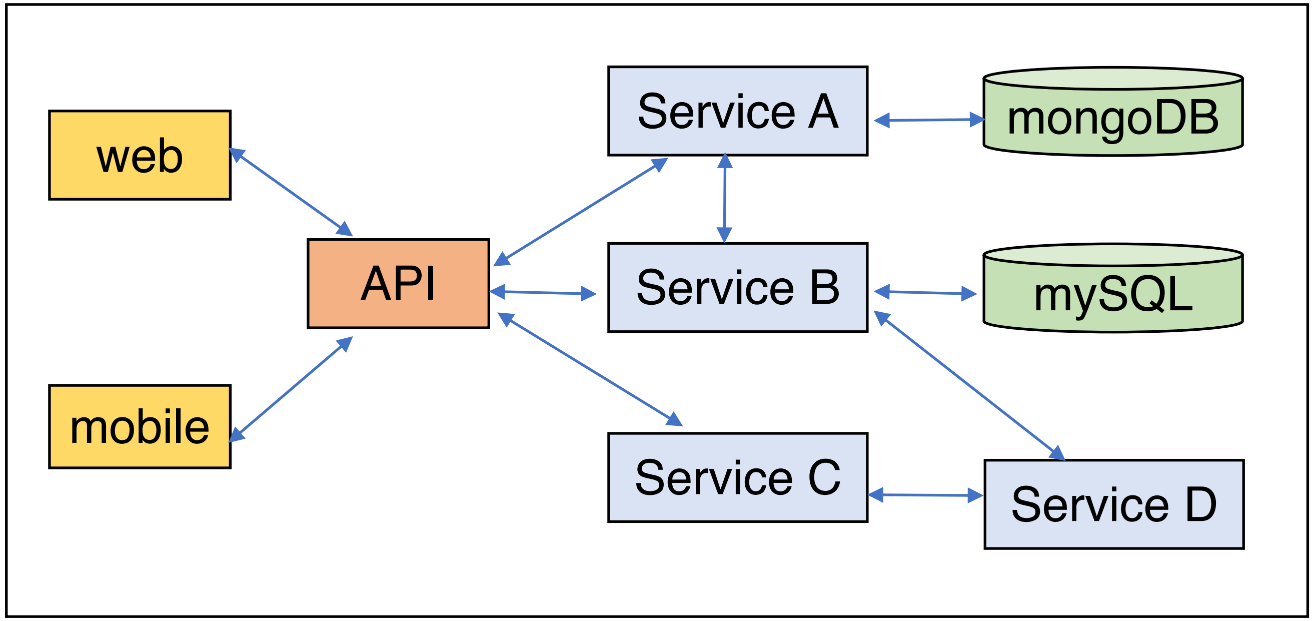
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

IBM Research

Cloud Deployments - Microservices

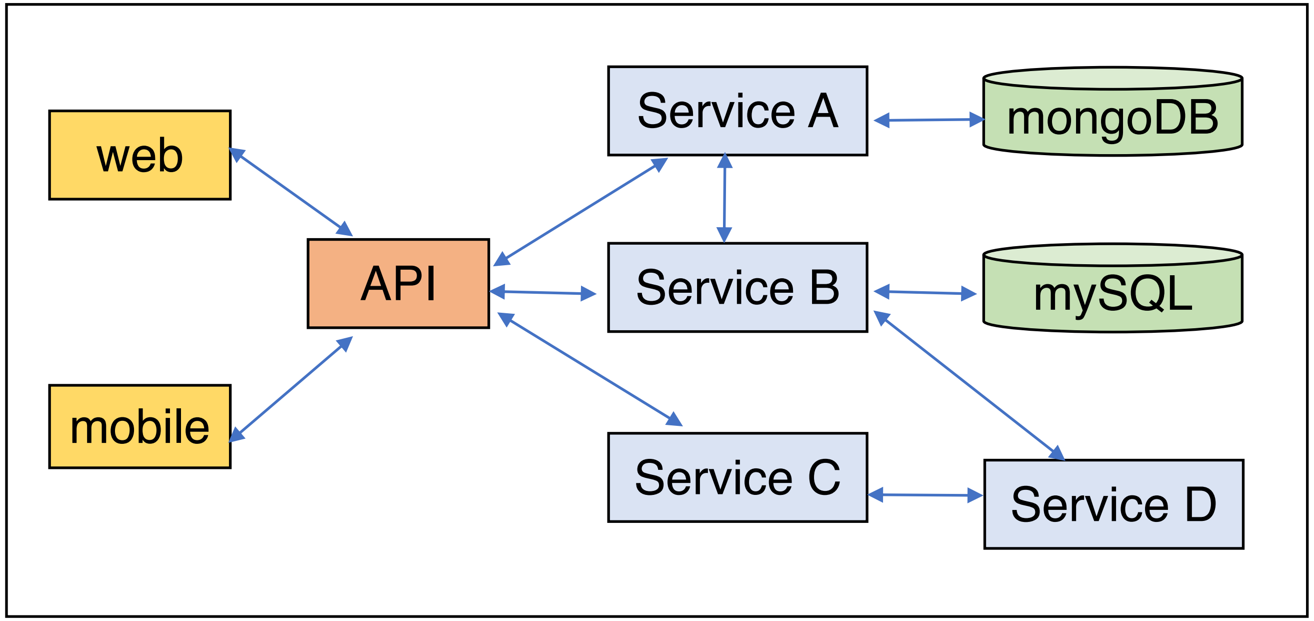


Monolithic Architecture



Microservices Architecture

Cloud Deployments – SLA Violations!

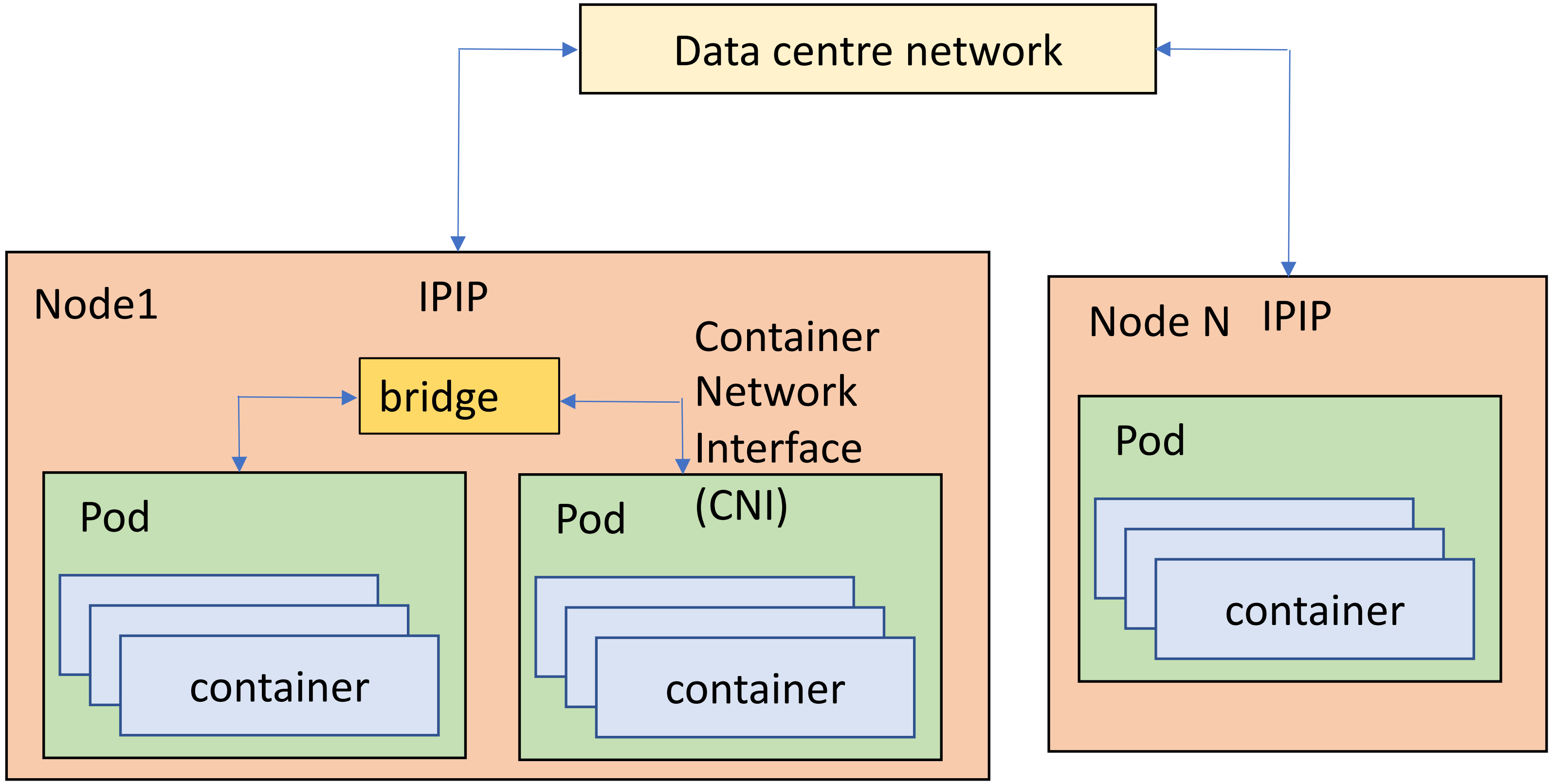


Kubernetes

OPENSHIFT

openstack.

Network Connectivity in Microservices



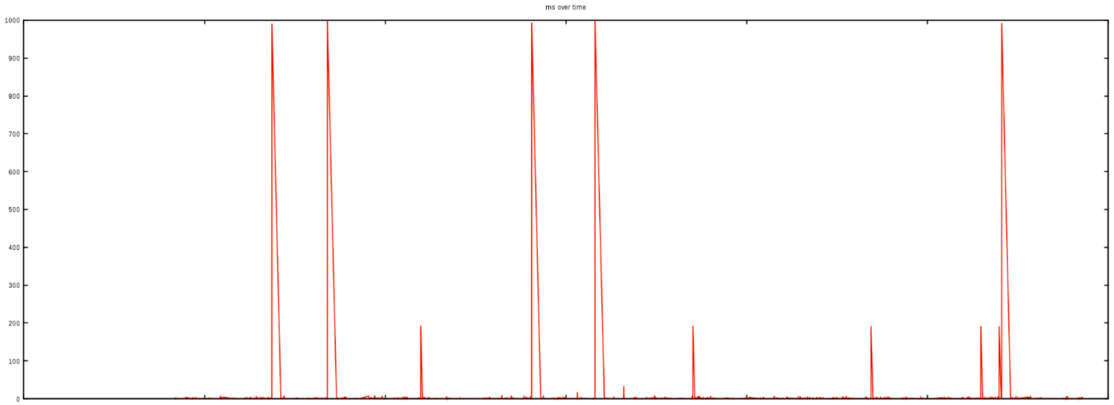
Kubernetes

OPENSHIFT

openstack

Performance Issues

- Sporadic increase in latencies
- 36% of performance anomalies are Transient [Bufscope, NSDI ‘22]
- Reasons could be :
  - On any of the nodes involved :
    - NAT, load-balancer, sender, receiver, etc.
      - IPTables configuration
      - CPU scheduling
      - NIC Queueing
    - Network links
      - Congestion
      - Microbursts
      - Link Failures
      - Packet corruption

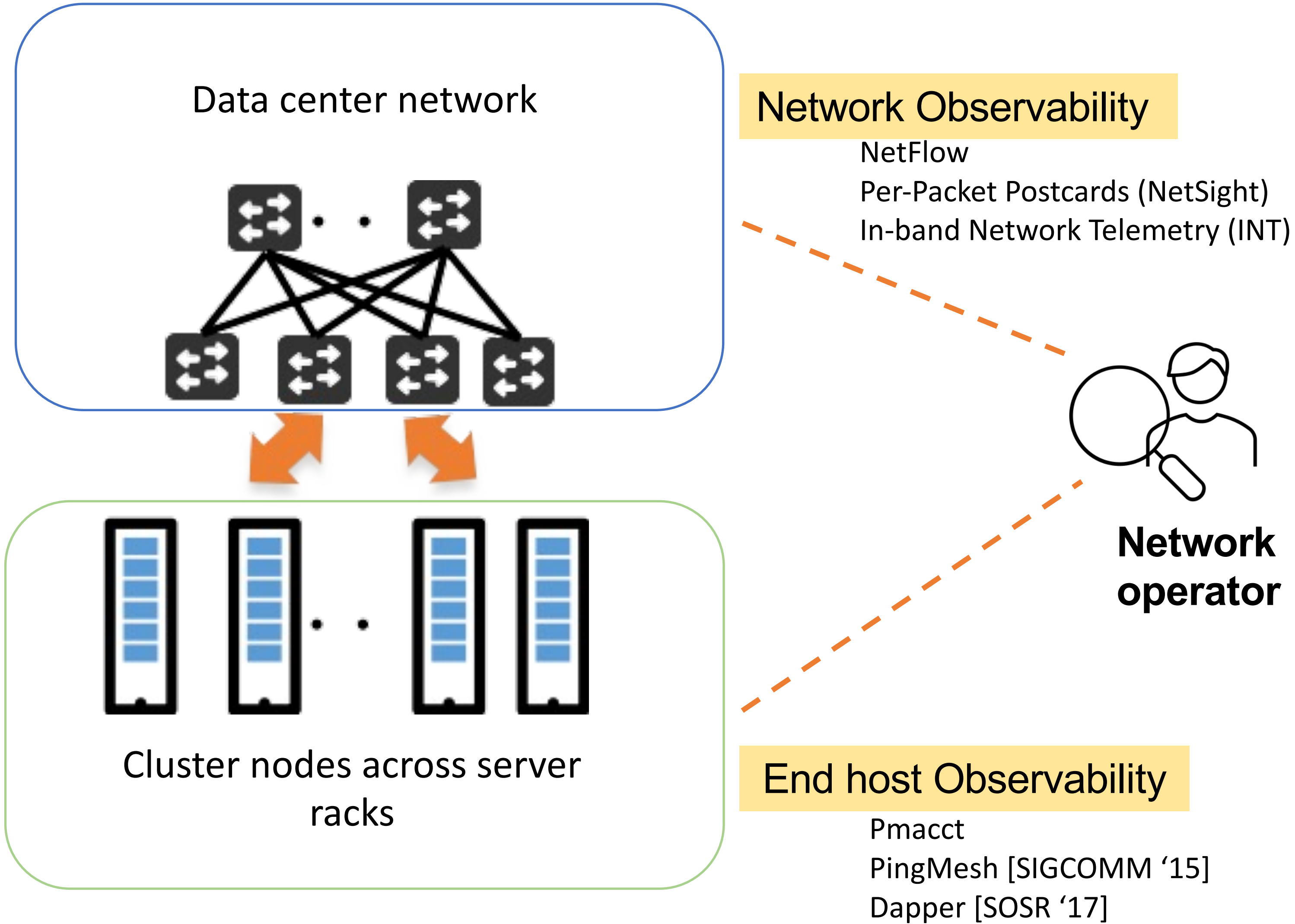


Facebook Microbursts [IMC’17]

https://github.blog/2019-11-21-debugging-network-stalls-on-kubernetes/

https://blog.cloudflare.com/the-story-of-one-latency-spike/

Need for end-to-end observability



Data center network

Cluster nodes across server racks

Network operator

Network Observability

NetFlow
Per-Packet Postcards (NetSight)
In-band Network Telemetry (INT)

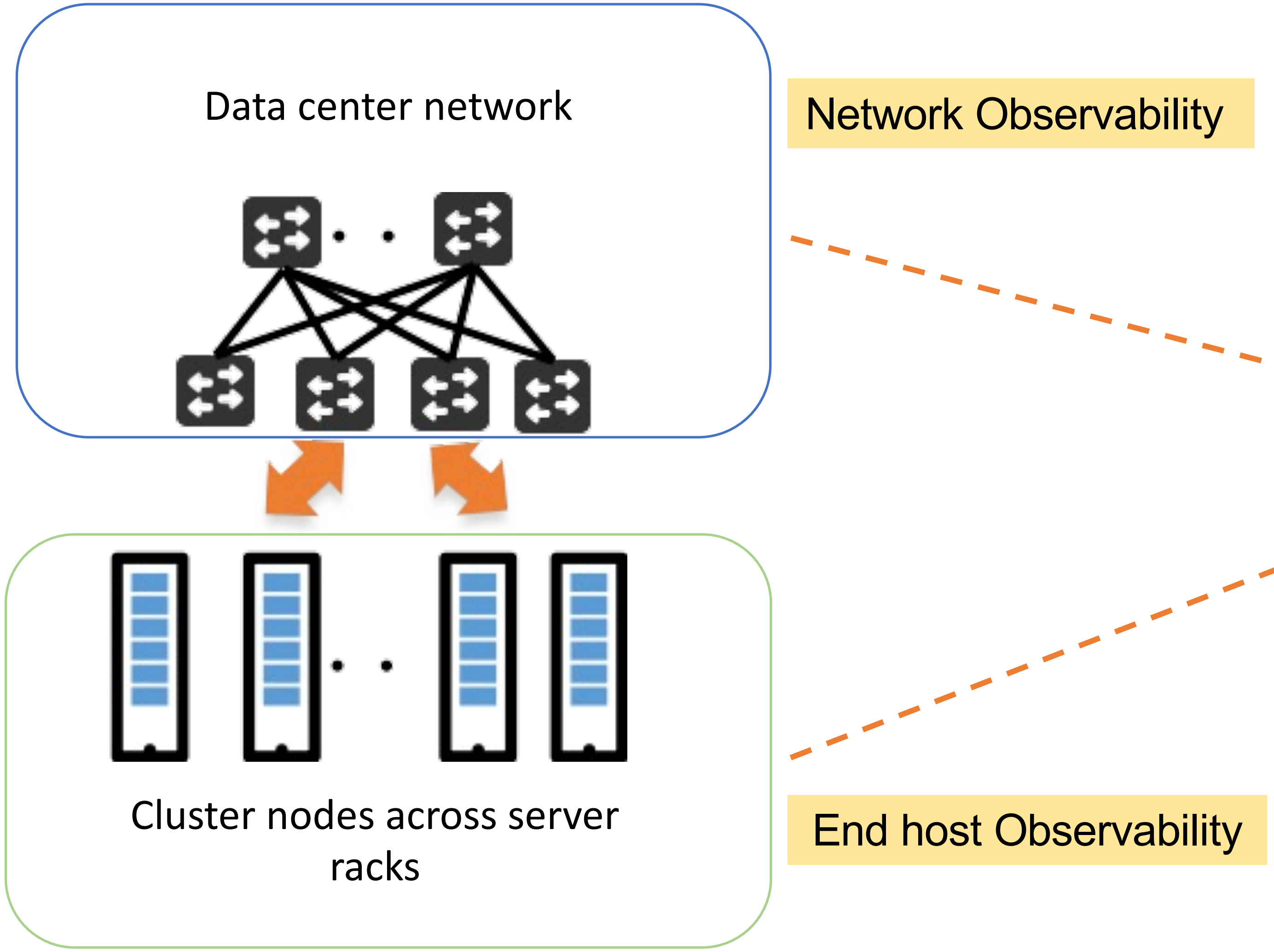
End host Observability

Pmacct
PingMesh [SIGCOMM ‘15]
Dapper [SOSR ‘17]

Limitations

- Disaggregated
- Network-level abstraction
  - Service Mesh, Proxies
- Flow IDs does not match
  - VXLANs, NAT
- Lack end-to-end visibility

Aggregating information and performing root cause analysis can be slow, inaccurate and misleading.

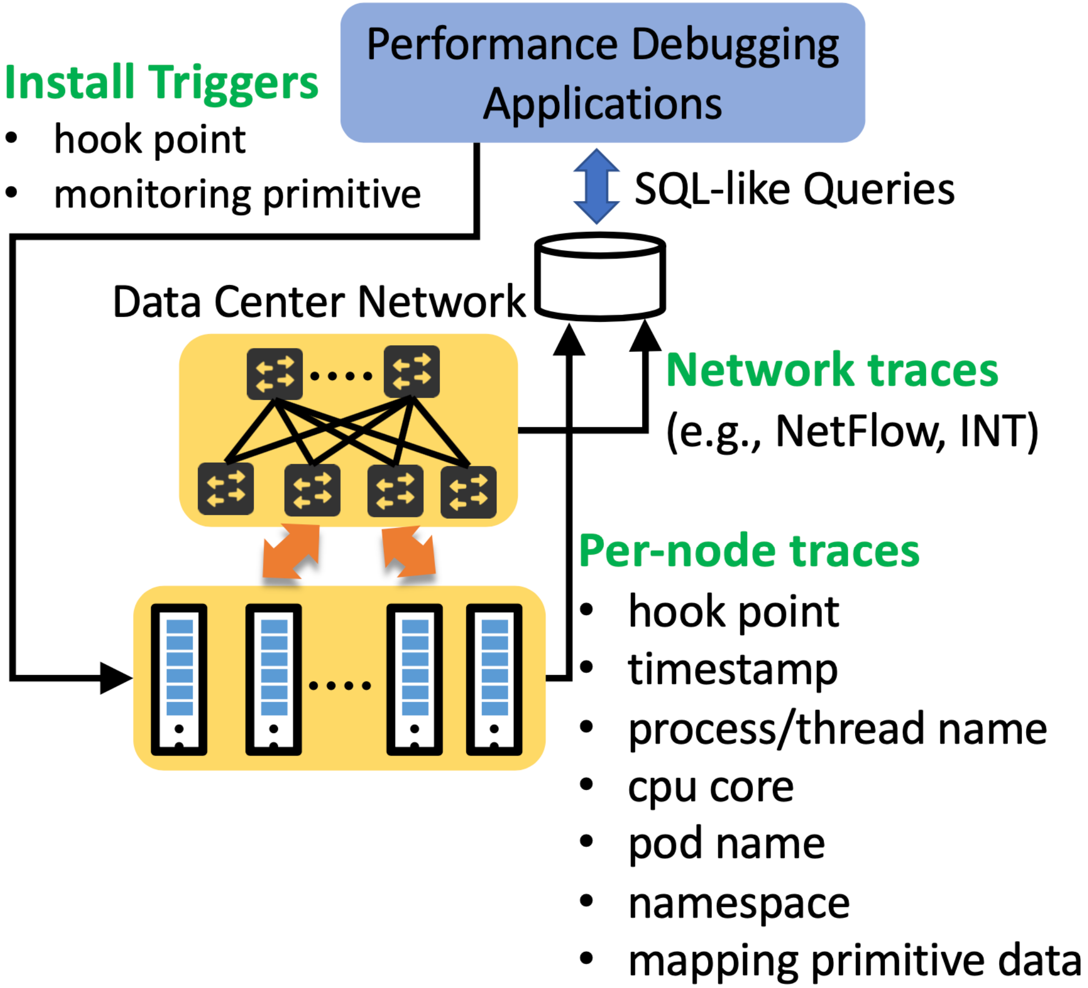


Is it possible to design and efficient performance monitoring framework that can achieve **end-to-end (cross-domain) Observability**?

Design

Enhance Host-observability:

- ***Monitoring Primitive***
  - RTT increase
  - Packet Drops
- ***Tracer***
  - Collect Host-metrics (TPs, Socket, TC, etc)
  - Maintain recent history
- ***Mapping Primitive***
  - Container flow-IDs to Node flow-IDs



Performance Debugging Applications

Install Triggers

- hook point
- monitoring primitive

SQL-like Queries

Data Center Network

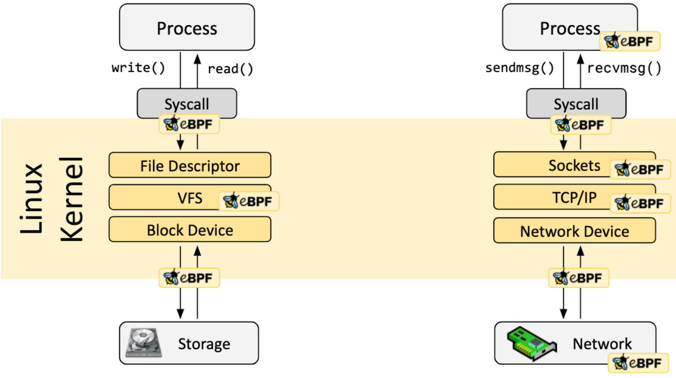
Network traces
(e.g., NetFlow, INT)

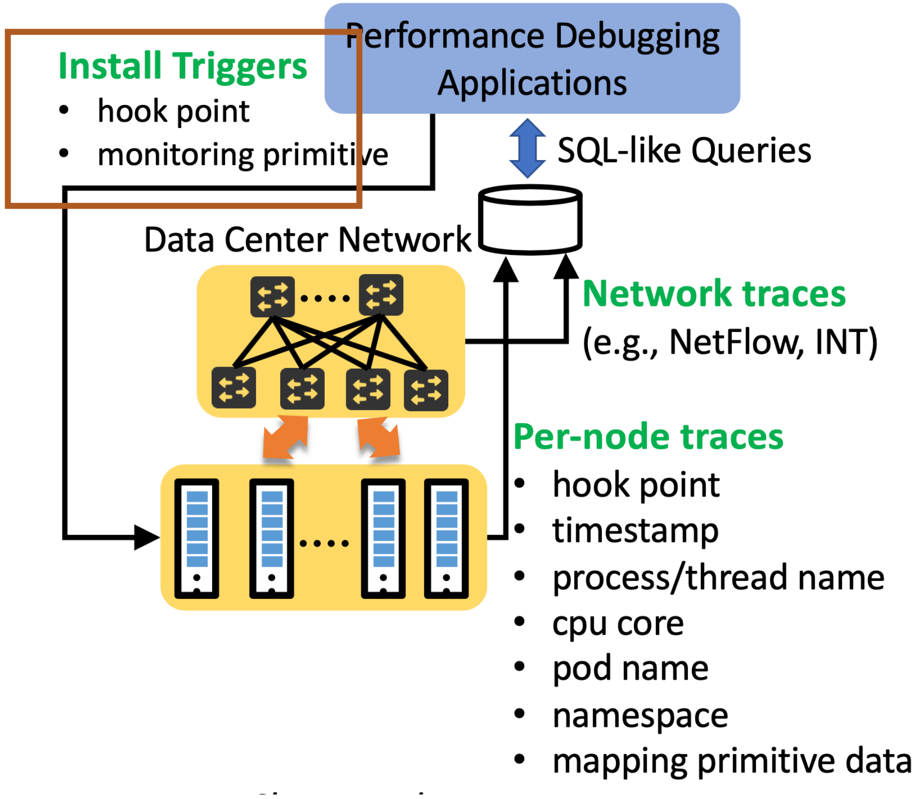
Per-node traces

- hook point
- timestamp
- process/thread name
- cpu core
- pod name
- namespace
- mapping primitive data

Prototype Implementation

- ***Monitoring Primitive [eBPF[1]-based]***
  - RTT monitoring for TCP Flows
  - Stateful monitoring of Seq/ack-seq
  - Per-CPU LRU Hash to maintain <Seq, timestamp>
  - Per-flow Moving average of RTT
  - **Trigger:**
    - Upon Increase of avg RTT by x%
    - Threshold




Install Triggers
- hook point
- monitoring primitive

Performance Debugging Applications

SQL-like Queries

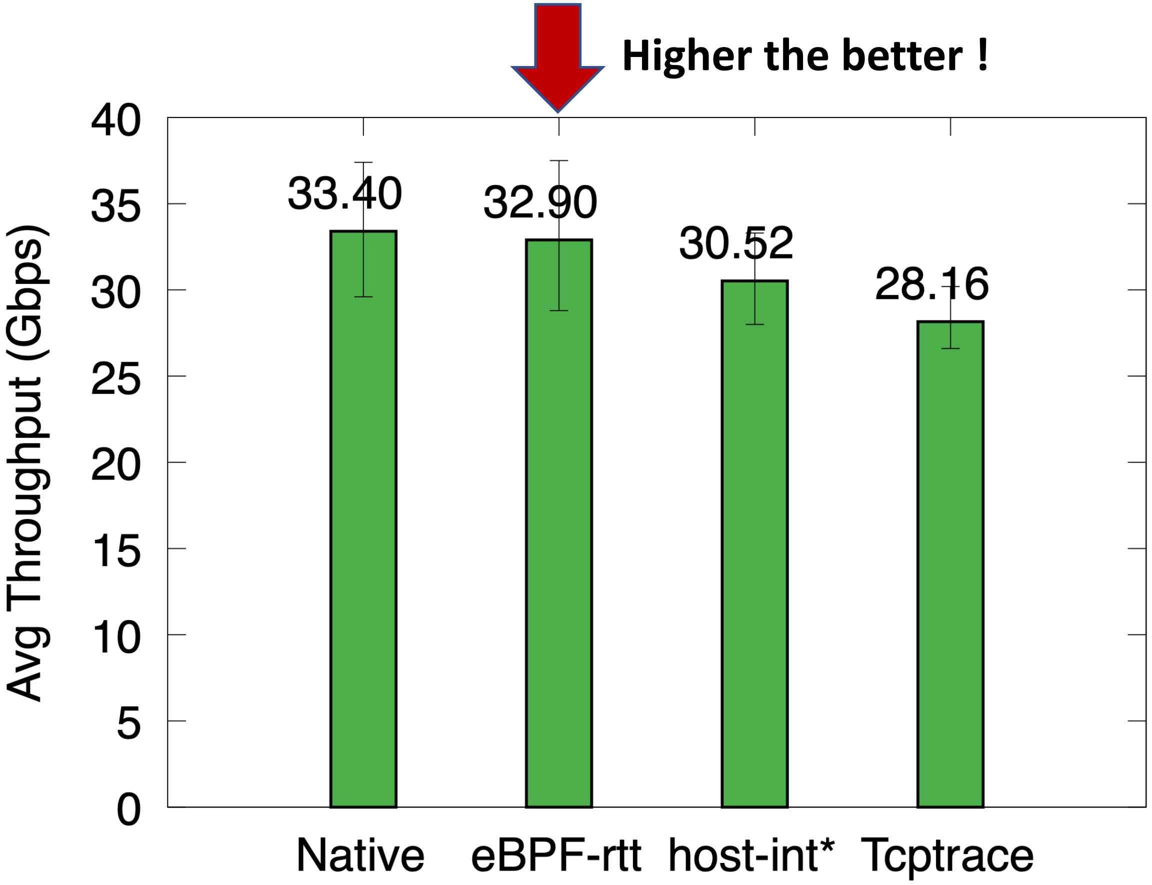
Data Center Network

Network traces
(e.g., NetFlow, INT)

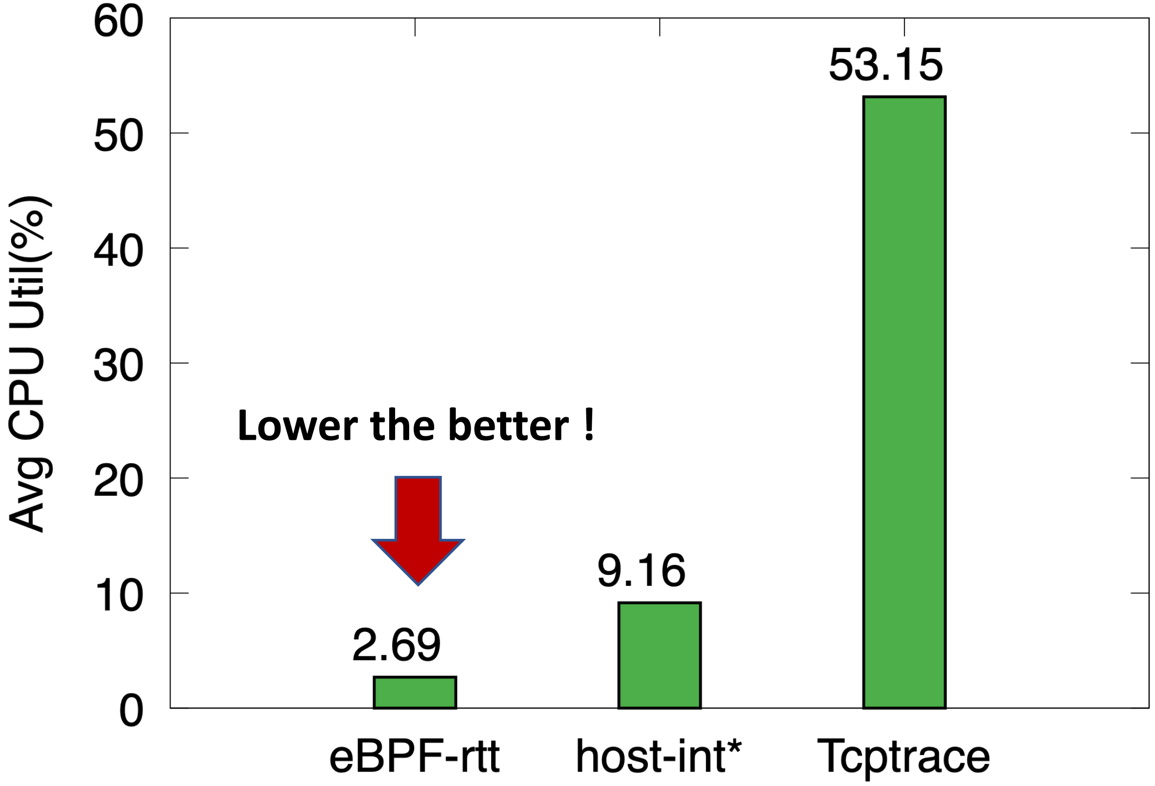
Per-node traces
- hook point
- timestamp
- process/thread name
- cpu core
- pod name
- namespace
- mapping primitive data

[1] ebpf.io

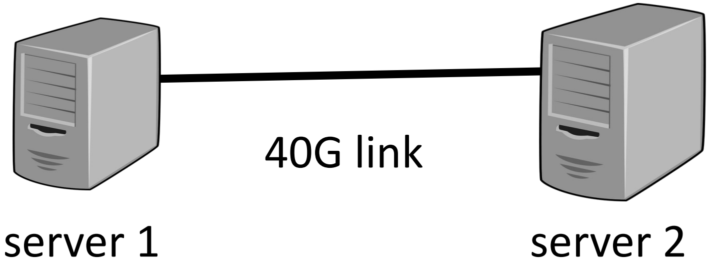
Evaluation



Higher the better !



Lower the better !



Ongoing Work

- *Tracer :*
  - *Maintains continuous list of events (syscalls, timestamps)*
  - *Ringbuffer-based recent history*
  - *eBPF/Intel-PT*
- *Mapping Primitive*
  - *eBPF-based flow mapping*
  - *Monitor vETHs and outgoing interfaces*
- *Evaluate on a larger setup*

Conclusion

- We present a case to build cross-domain observability framework to debug performance issues.

- Feasibility of the system by implementing monitoring primitive.

- eBPF-based RTT monitoring with low overhead.

Thank You !

Contact : Pravein.Govindan.Kannan@ibm.com