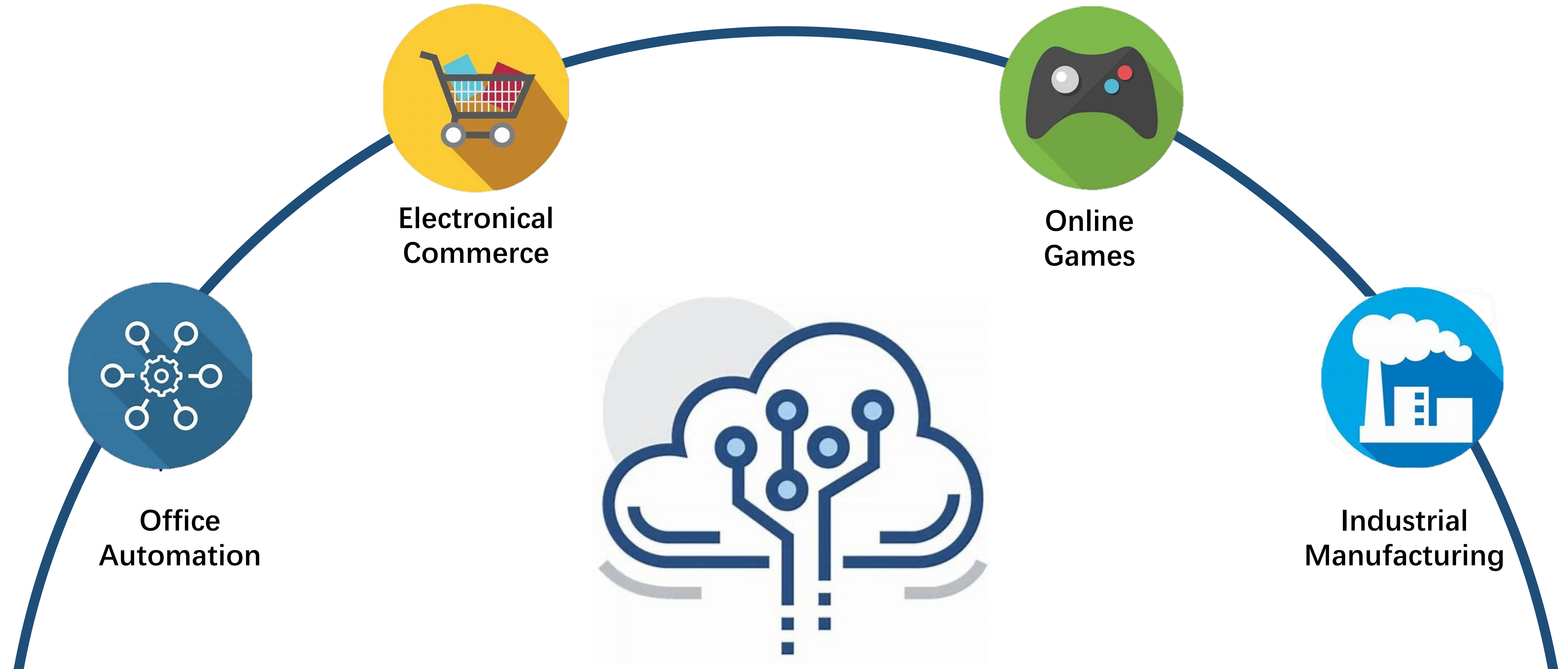
Network-Centric Distributed Tracing with DeepFlow: Troubleshooting Your Microservices in Zero Code

Junxian Shen, Han Zhang, Yang Xiang, Xingang Shi, Xinrui Li,
Yunxi Shen, Zijian Zhang, Yongxiang Wu, Xia Yin, Jilong Wang, Mingwei Xu,
Yahui Li, Jiping Yin, Jianchang Song, Zhuofeng Li, Runjie Nie

清華大學
Tsinghua University

云杉网络
Yunshan Networks



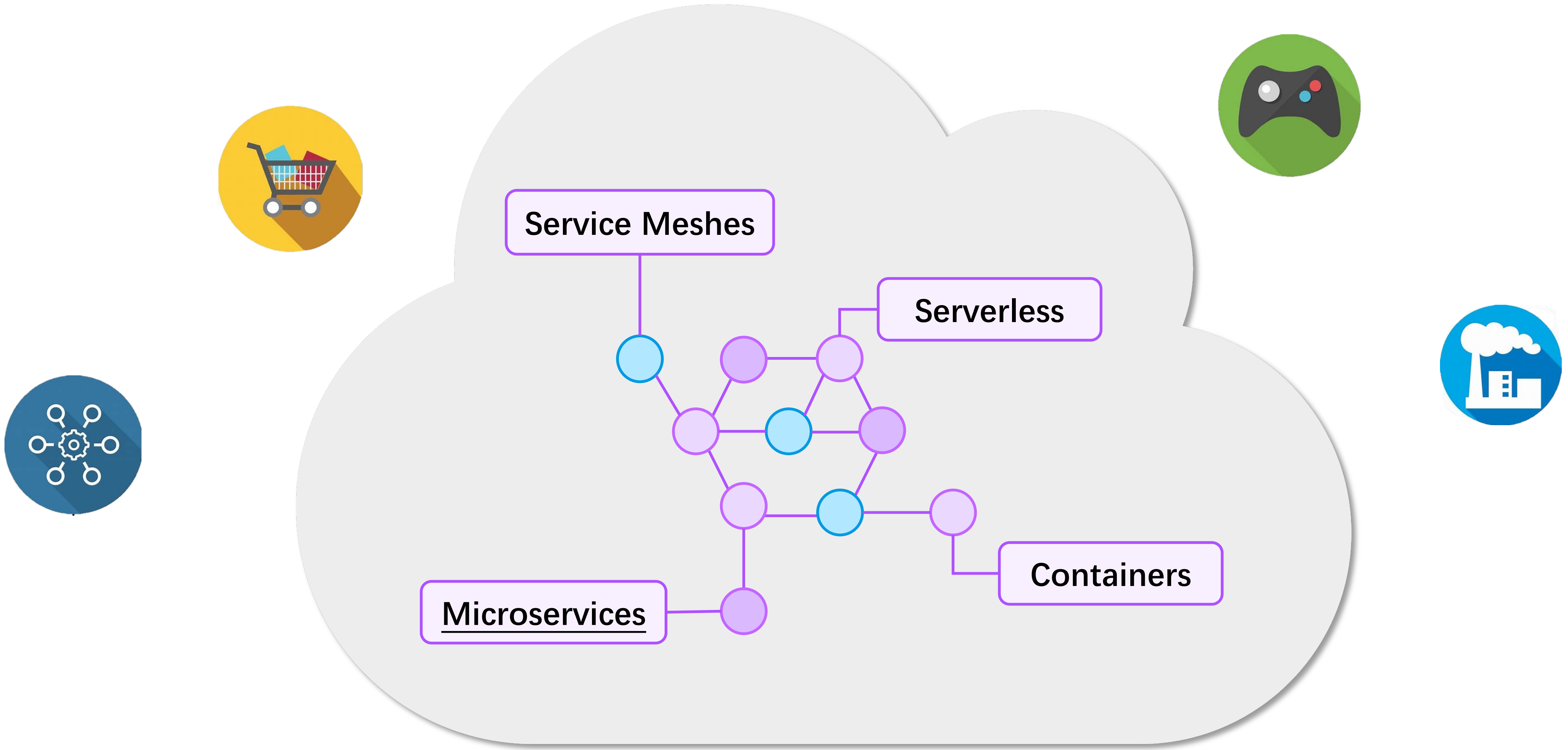
Electronical Commerce

Online Games

Office Automation

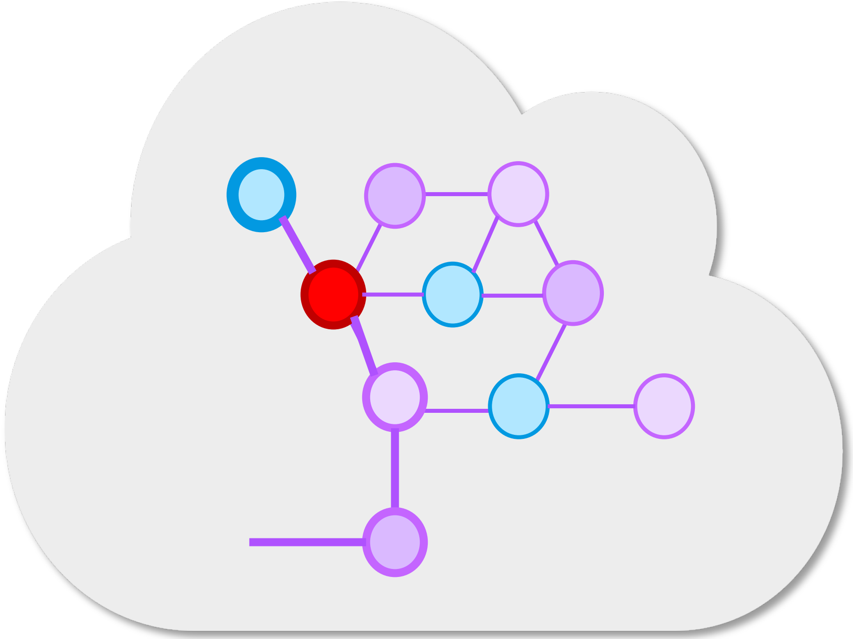
Industrial Manufacturing

Applications in many fields are migrating to Cloud Platforms.

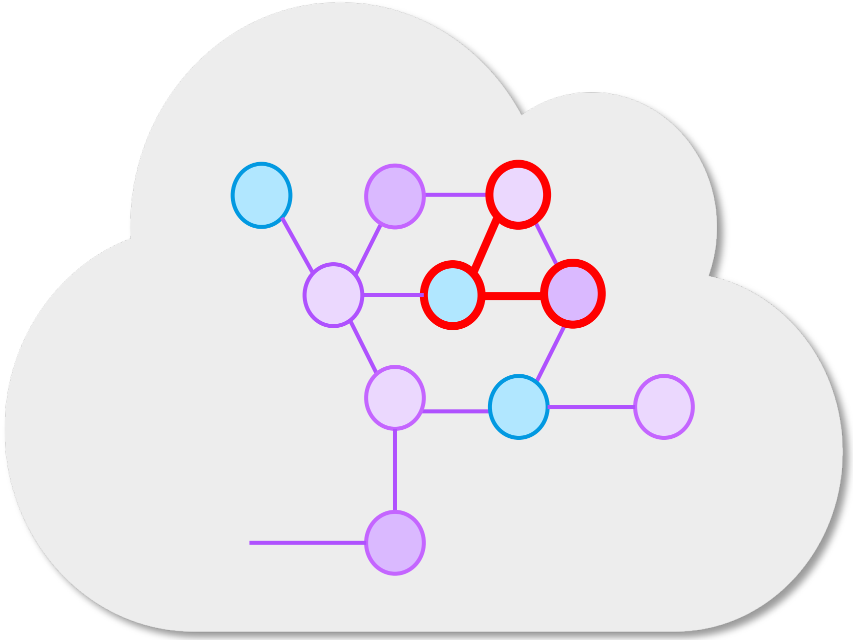


Systems become more Distributed, using various new concepts.

Microservice as a Double-Edged Sword



Identifying a broken point
in a long chain is hard



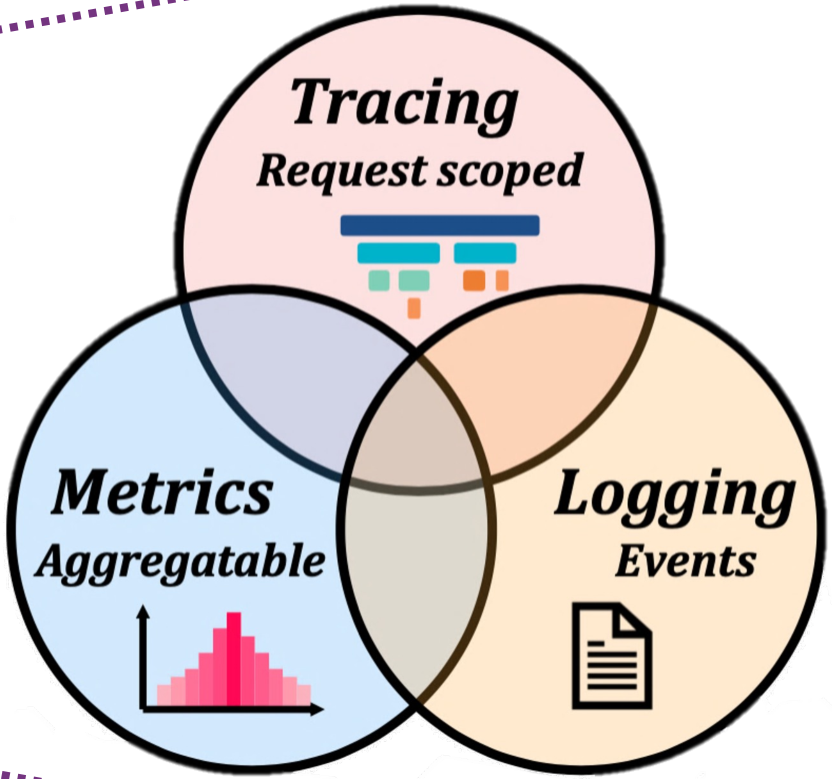
Interaction between components
bring unexpected failures

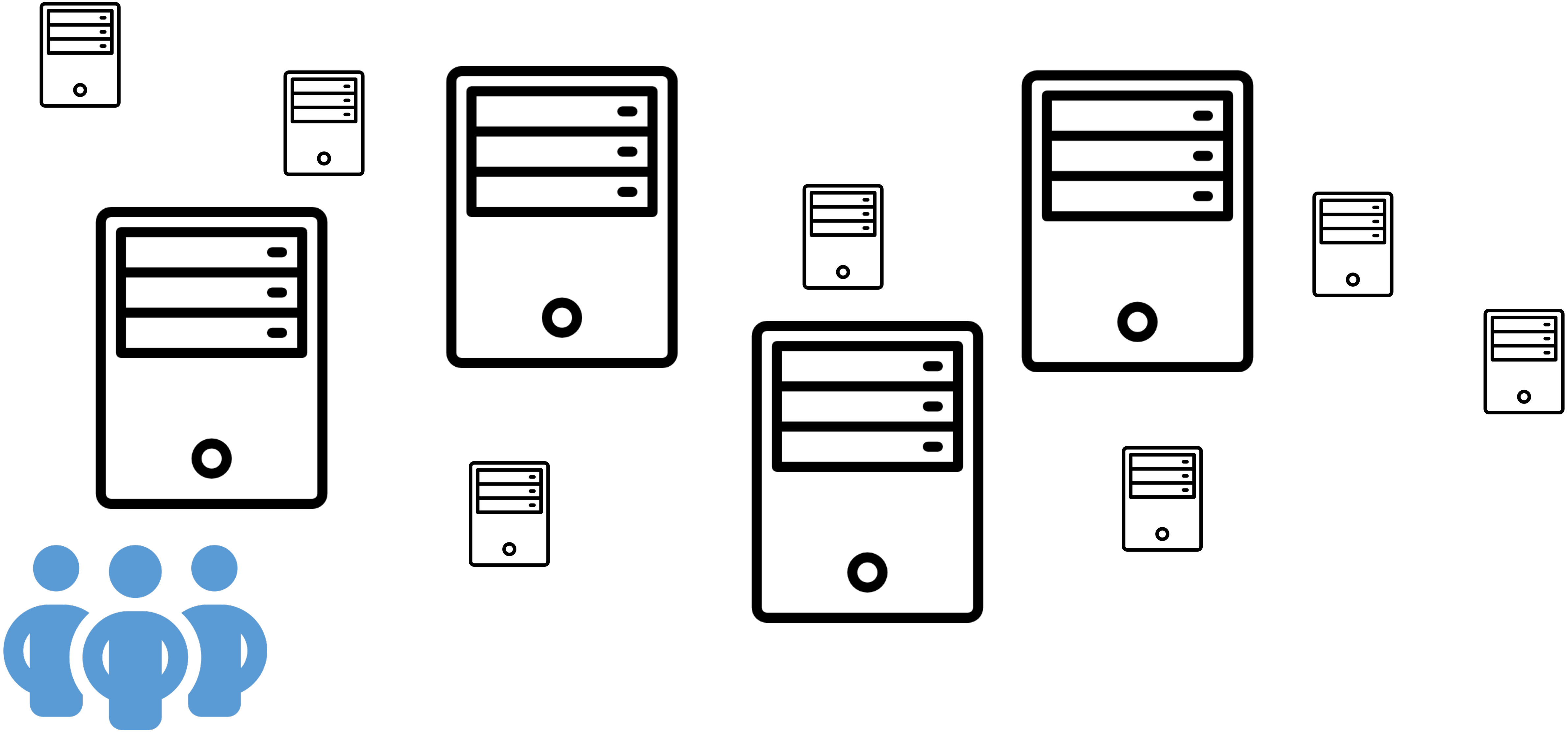
Deep Visibility into modern software systems is required
for faster problem identification and resolution.

Observability

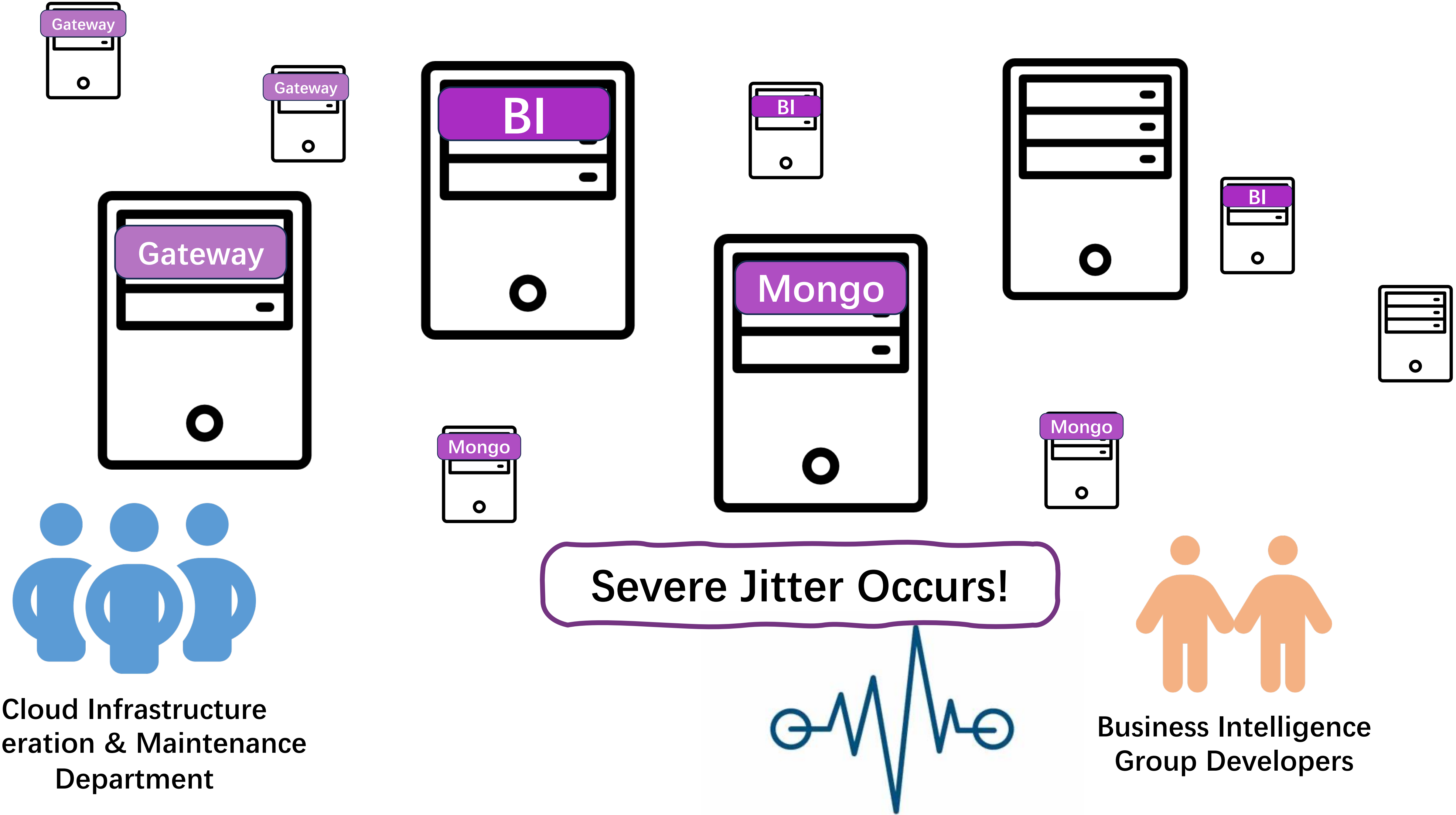
- Control Theory:
  Understand the internal states of a system based on its external outputs.

- Software System:
  Correlate the collected data to provide contextual information throughout the system.





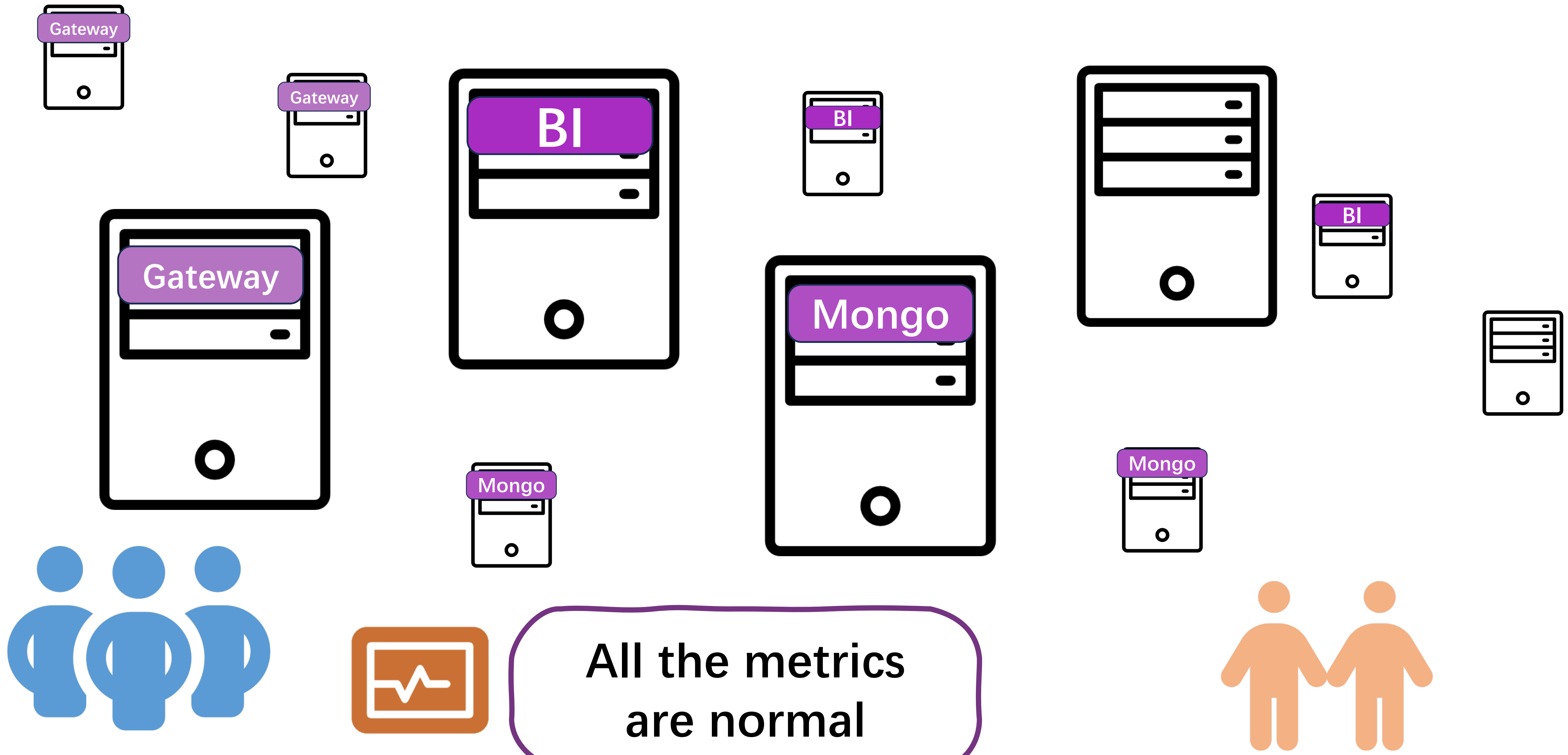
Cloud Infrastructure
Operation & Maintenance
Department



Severe Jitter Occurs!

Cloud Infrastructure
Operation & Maintenance
Department

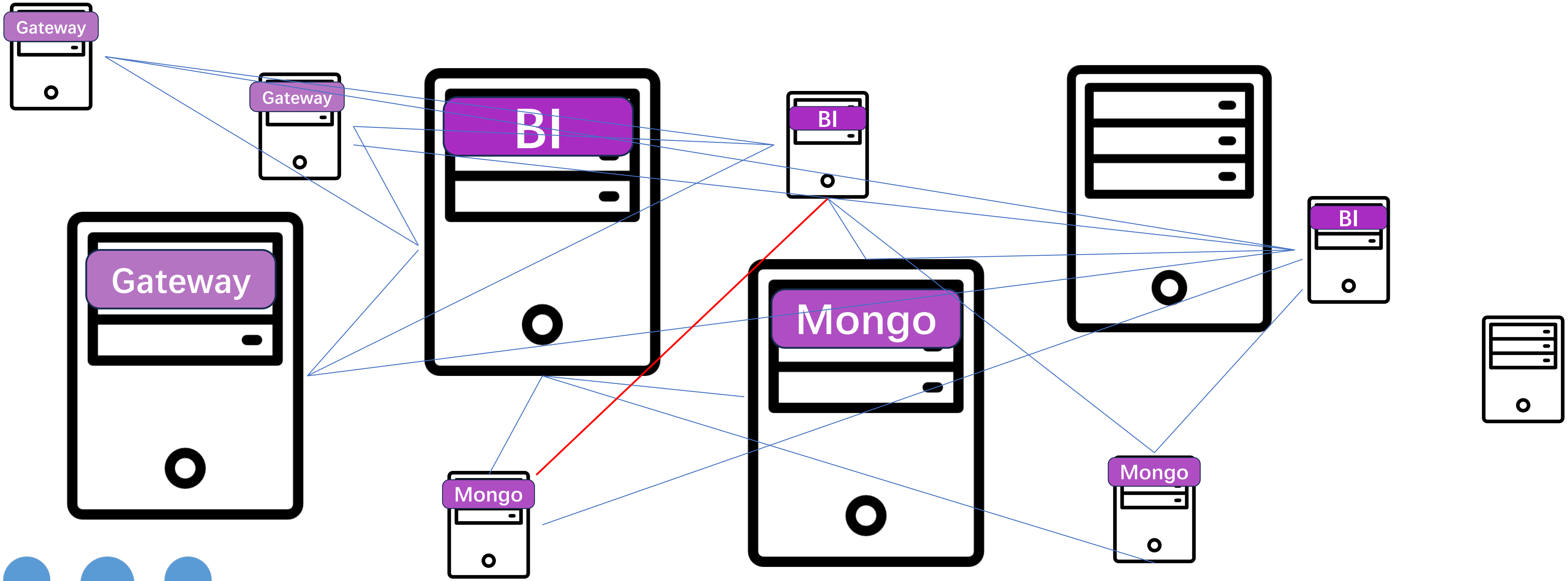
Business Intelligence
Group Developers



All the metrics are normal

Cloud Infrastructure Operation & Maintenance Department

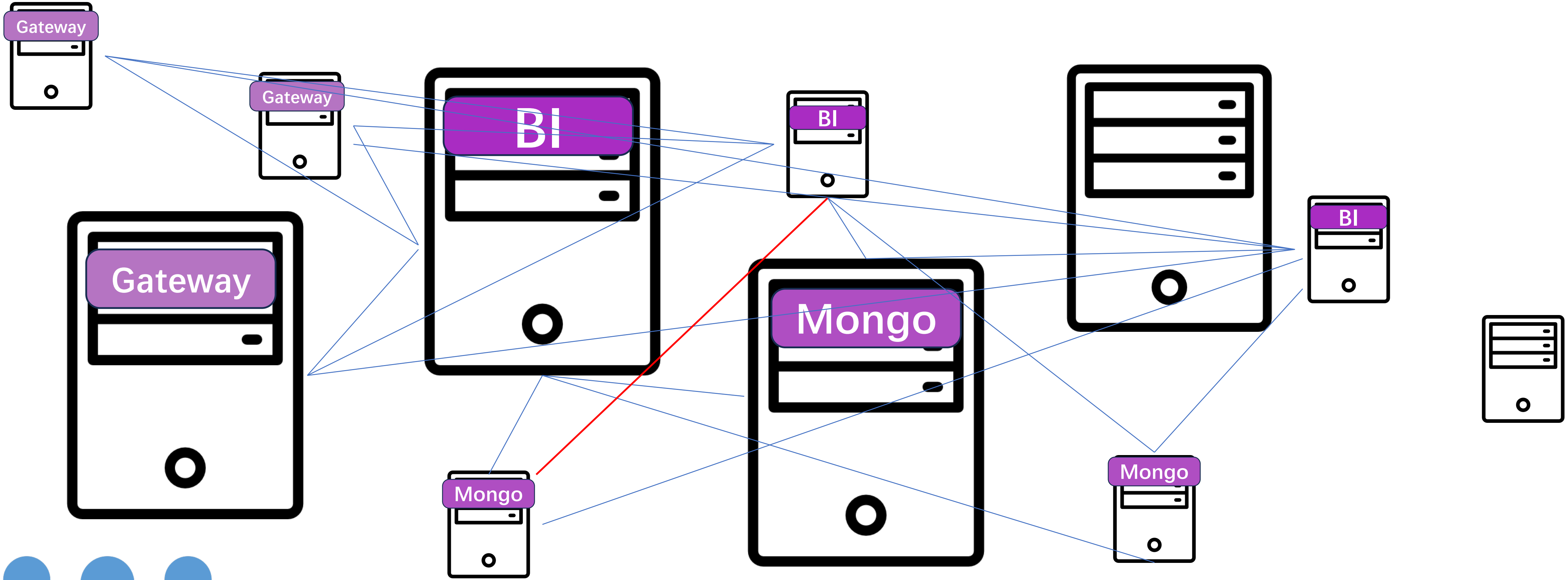
Business Intelligence Group Developers



Cloud Infrastructure
Operation & Maintenance
Department

Mission Impossible
to find unexpected logs

Business Intelligence
Group Developers



Cloud Infrastructure
Operation & Maintenance
Department

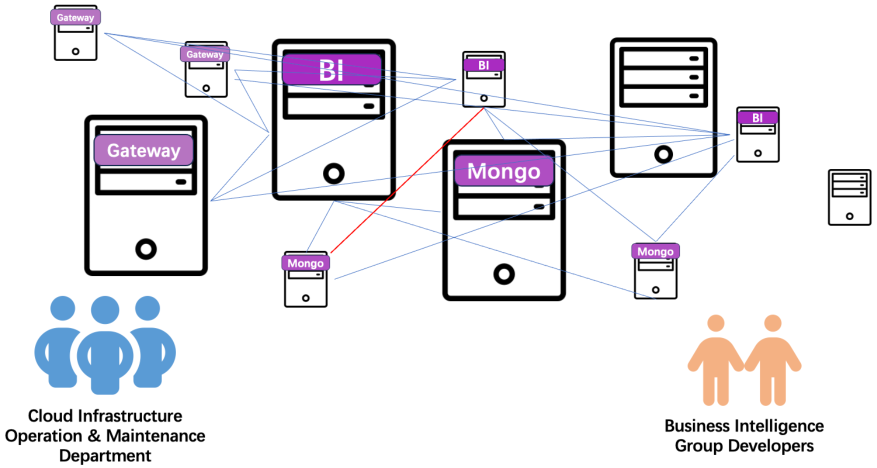
Problem Stalls : (

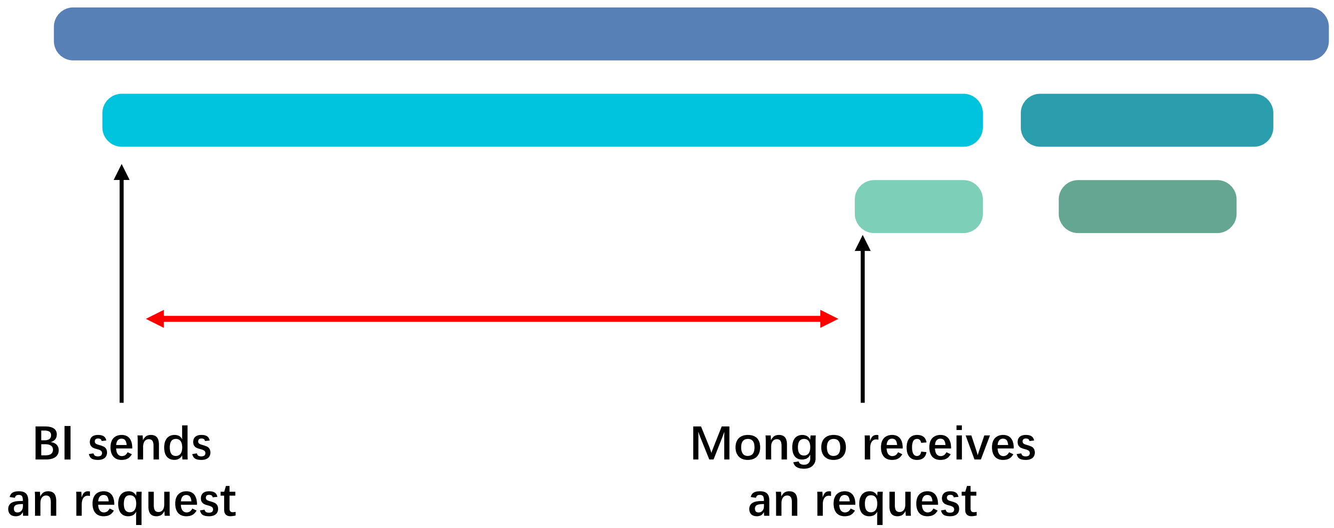
Business Intelligence
Group Developers

JAEGER

√ Problem narrows down from entire request to one hop

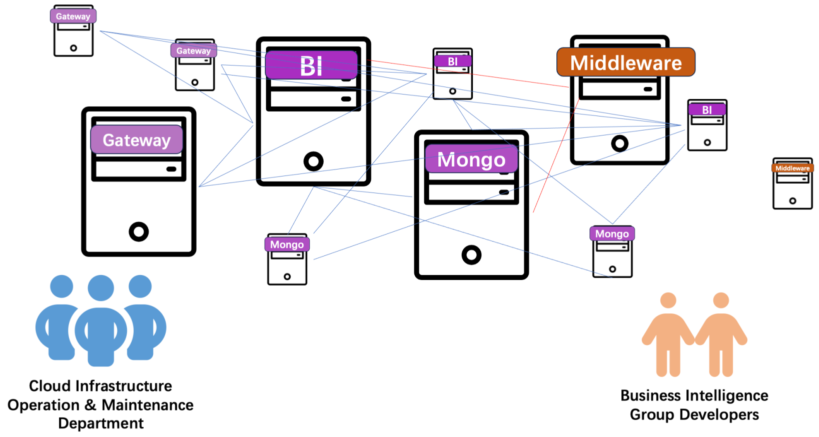
Nothing wrong can be found here X

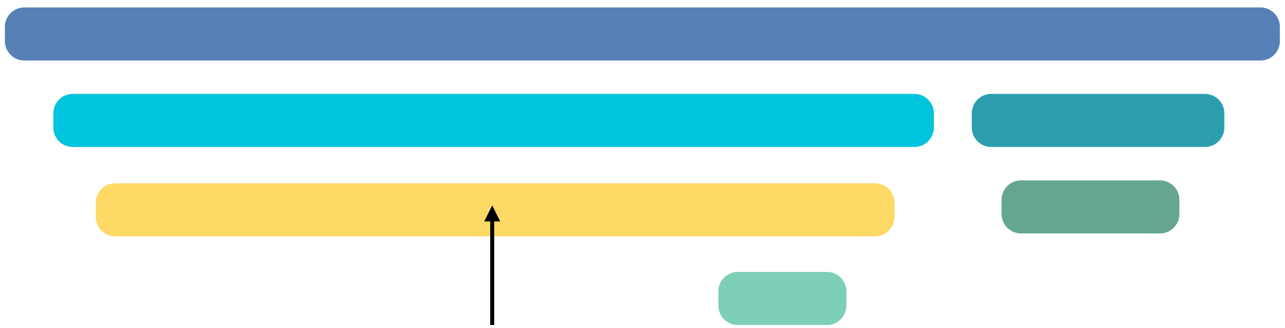

Gateway
Gateway
BI
BI
BI
Gateway
Mongo
Mongo
Mongo
Cloud Infrastructure Operation & Maintenance Department
Business Intelligence Group Developers



What happens between these two points?

DeepFlow


Gateway
Gateway
BI
BI
Middleware
BI
Gateway
Mongo
Middleware
Mongo
Mongo
Cloud Infrastructure Operation & Maintenance Department
Business Intelligence Group Developers



Span for Middleware

A Hidden Service Occurs!

DeepFlow

Non-Intrusive Matters A Lot !

Out-of-the-box Tracing

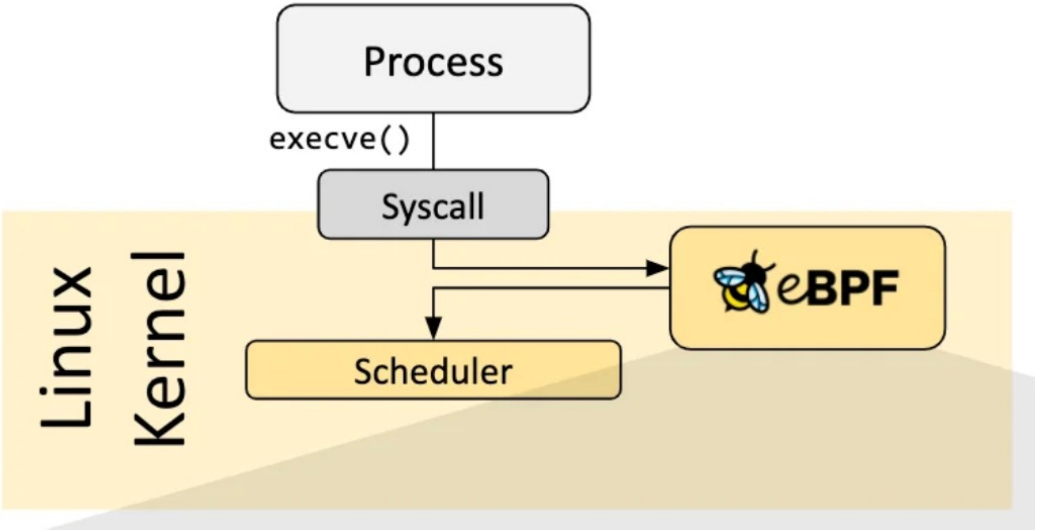
- Convenience
- Portability
- Stability

Rapid Problem Location

- Coverage
- Correlation

How to simultaneously provide rapid problem location and out-of-the-box tracing with the eBPF technology?

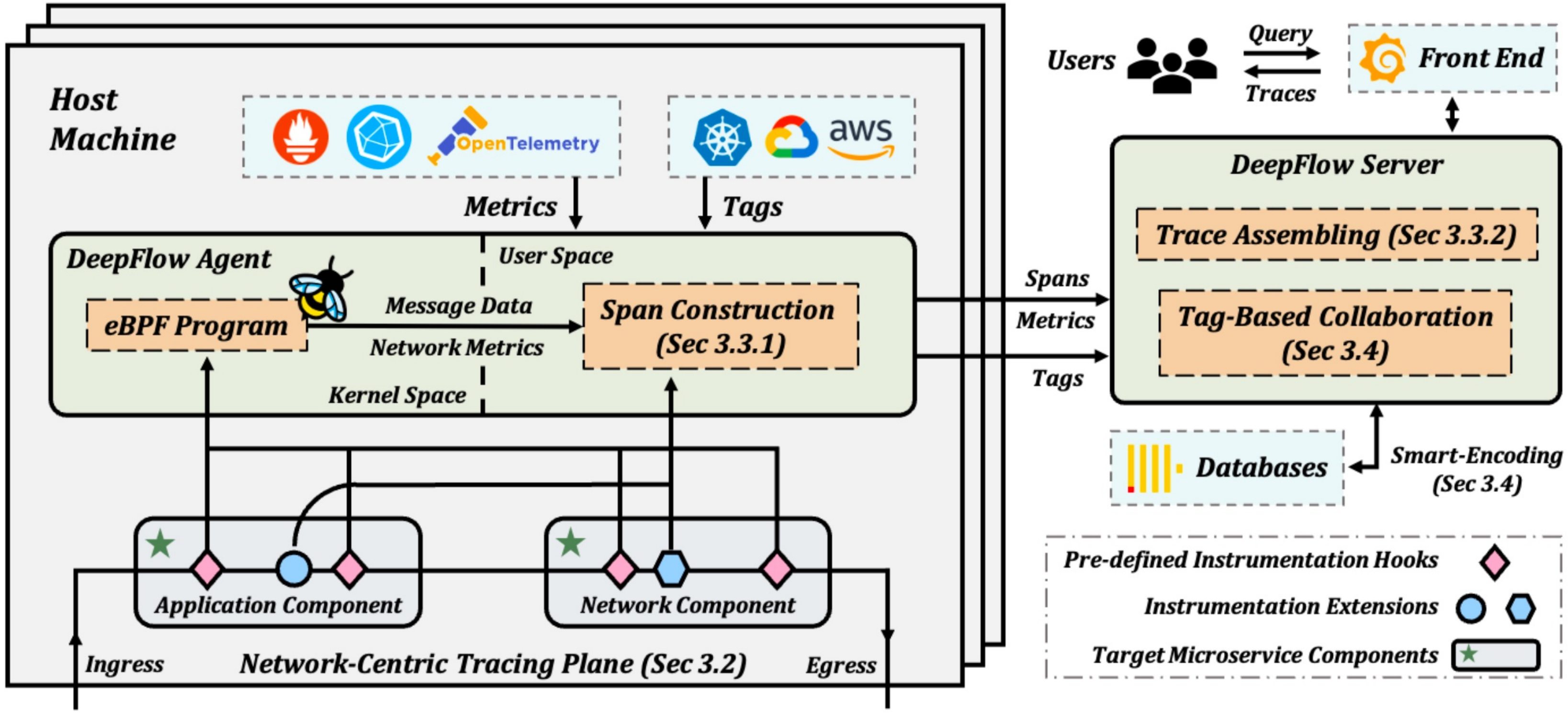
Opportunities by eBPF



- **What is eBPF?**
  - extended Berkeley Packet Filter
  - provides a virtual machine in the kernel that enables the execution of BPF programs written by users.

- **Why eBPF meets our needs?**
  - **Convenience**:
    non-intrusive hook insertion
  - **Stability**:
    eBPF verifier
  - **Correlation**:
    kernel's ability to access network info

Architecture Overview



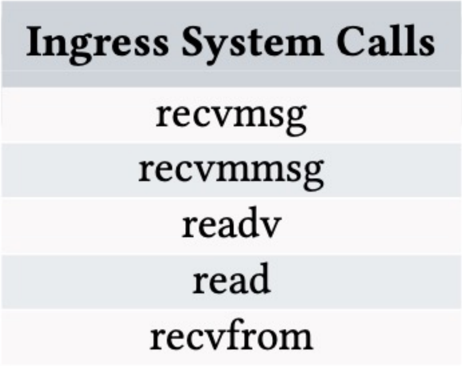
- **Agent**
  Trace Data Capturing
  Span Construction
- **Server**
  Span Storing
  Trace Assembling

Rich Third Party Integration Support

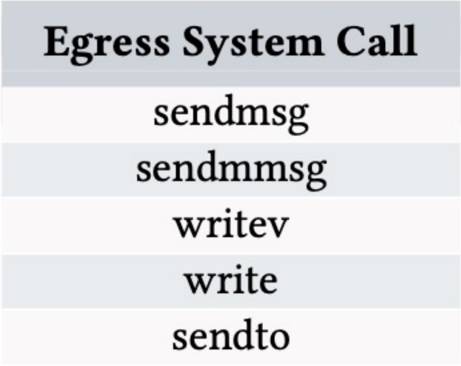
Key Designs

- Where shall we capture trace data from?
- How do we collect trace data using eBPF hooks?
- How to assemble the collected data into traces?

Where shall we capture trace data from?



Ingress System Calls
recvmsg
recvmmsg
readv
read
recvfrom



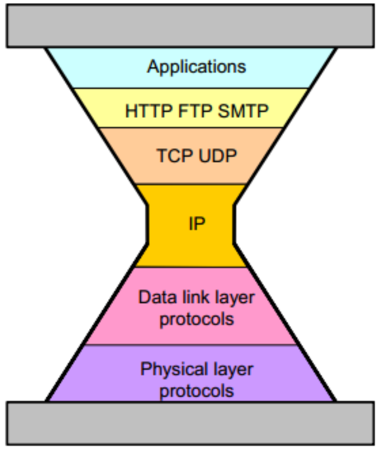
Egress System Call
sendmsg
sendmmsg
writev
write
sendto

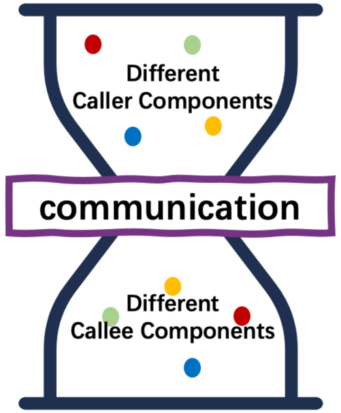
Design 1

A narrow-waist instrumentation model with <u>ingress and egress</u> system calls

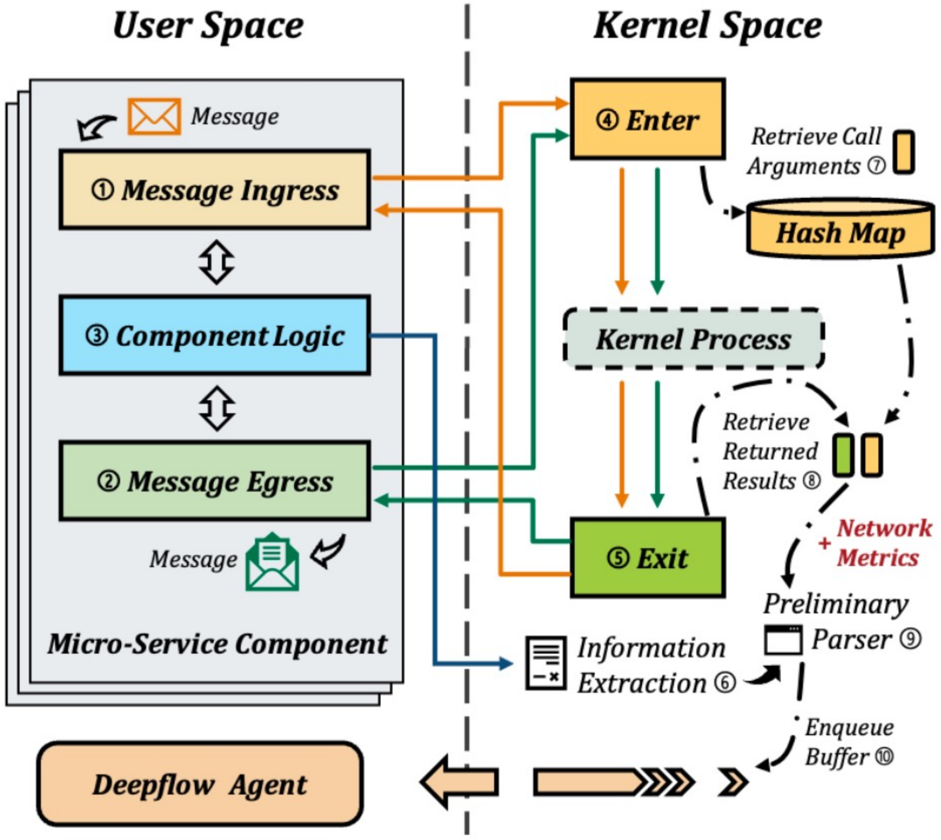
The execution of a microservice component is triggered by its communication.

Instrumenting all data communication system calls is well enough for basic tracing.





How do we collect trace data using eBPF hooks?



Design 2

In-kernel hook-based instrumentation

📌 Store communication information as it enters or exits the kernel.

Combine using hash map and send to Deepflow Agent

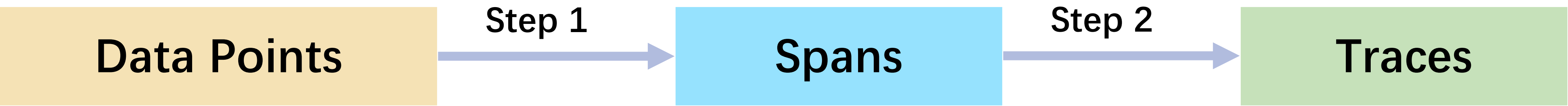
Program Info	Network Info
Tracing Info	Syscall Info

How to assemble the collected data into traces?

Design 3

Implicit context propagation with hierarchical aggregation

The information required for context propagation
is already contained in **<u>network-related data</u>**.

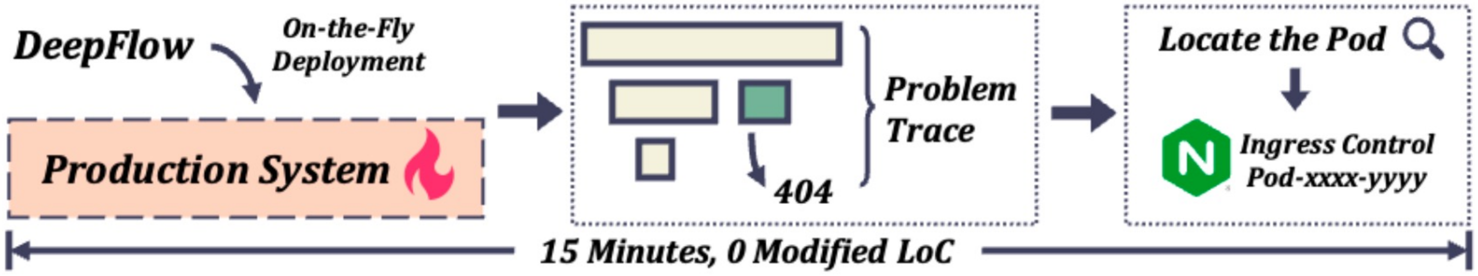


1. Span Construction		2. Trace Construction
Message Type Inference (i.e. Request/Response)	Session Aggregation	Iterative Span Search Parent-Child Relationship
Protocol Payload Format	TCP 5-tuple Specific Protocol Headers	

Production Usage

26 Big company clients

71 Critical performance issues

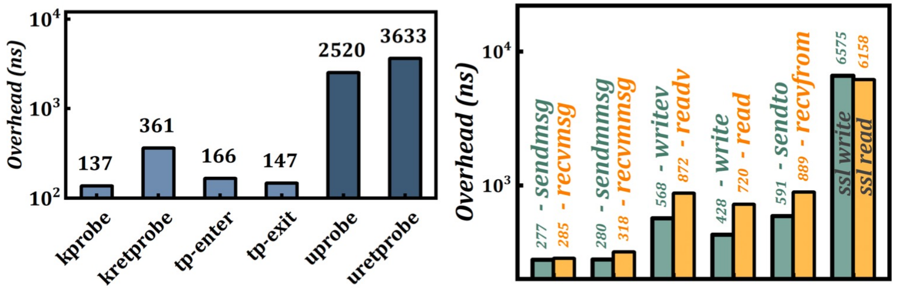


✓ **quick production system performance debugging**

- on-the-fly deployment
- non-intrusive collection

Testbed Evaluation

Trace Collection Overhead

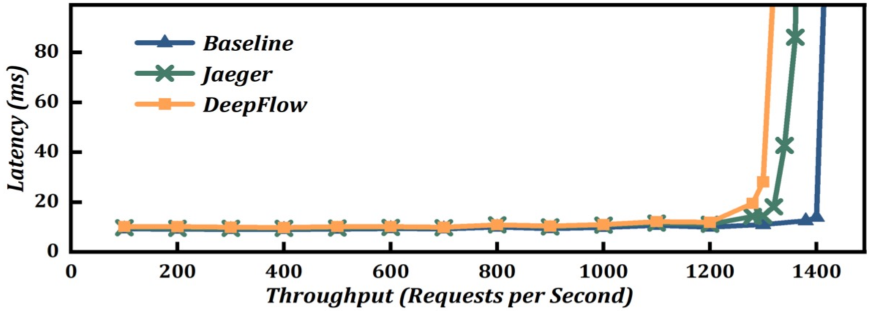


(a) Per-event eBPF instrumentation overhead. **tp** stands for tracepoint.

(b) Overhead of each instrumentation points in DeepFlow.

- ✓ Acceptable extra latency brought by Deepflow Agent.

End-to-End Performance



(a) End-to-end performance evaluation of Spring Boot demo.

- ✓ Marginally inferior performance (<7% overhead on throughput)
- ✓ Significantly more spans per trace (4 vs. 18)

Key Takeaways

DeepFlow®

CLOUD NATIVE LANDSCAPE

eBPF Project Landscape

- Evolving distributed scenarios such as microservices have **new requirements** on tracing frameworks.
- DeepFlow establishes a **network-centric tracing plane** with eBPF in the kernel non-intrusively.
- The collected network metrics are used to achieve **implicit context propagation**.

Try it out: https://deepflow.io/community.html