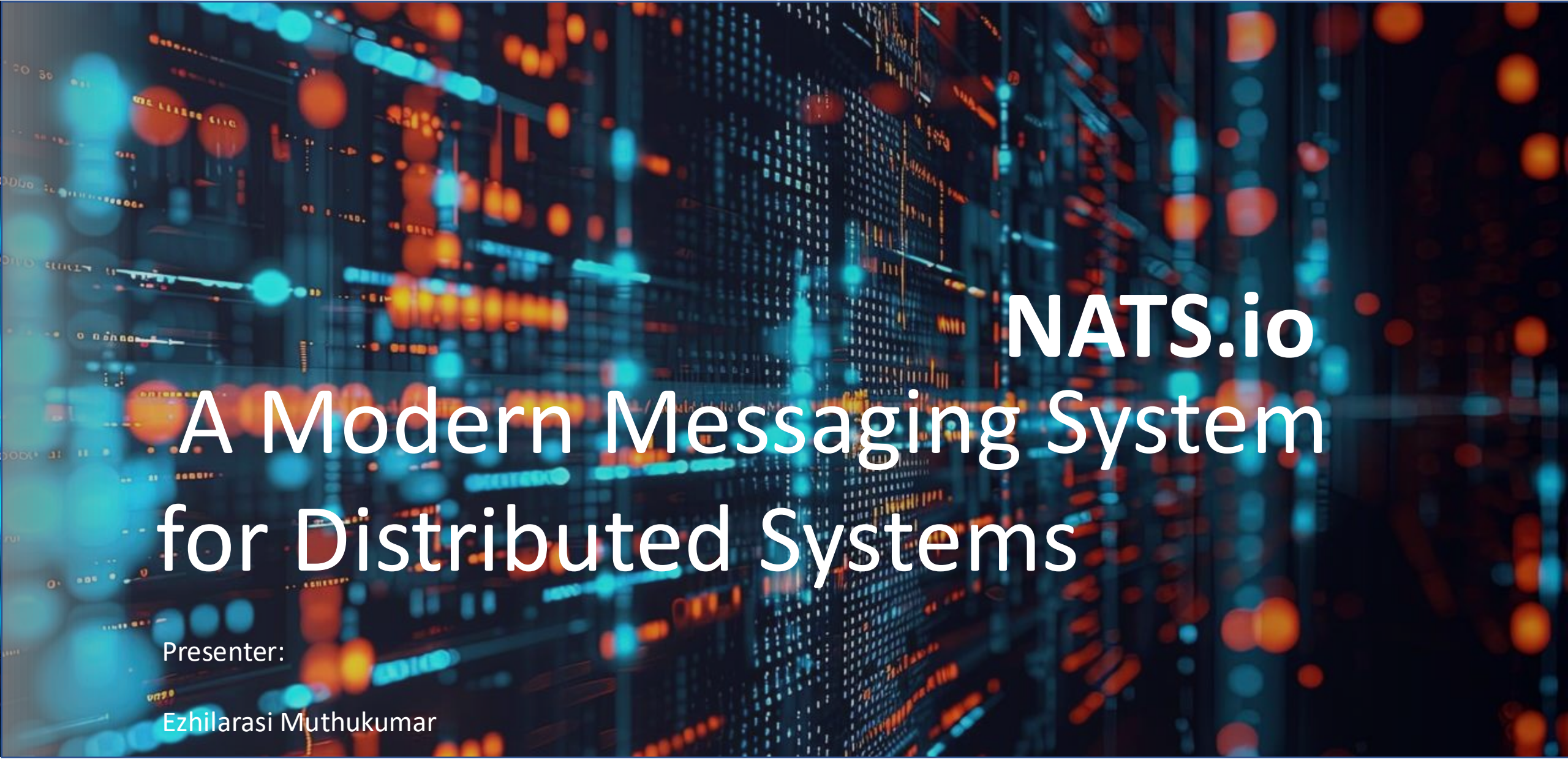


NATS.io
A Modern Messaging System for Distributed Systems

Presenter:

Ezhilarasi Muthukumar

UNIVERSITÉ CÔTE D'AZUR

FRANCE 2030 Initiative d'Excellence

Approved by 3iA Côte d'Azur

International Programs MSc Université Côte d'Azur

Agenda

- Introduction to NATS
- Architecture of NATS
- Use Cases & Problems NATS Solves
- Pros & Cons
- Comparison Against Alternate Technology
- Conclusion

1. Introduction to NATS

NATS (Neural Autonomic Transport System) is a simple, secure, and high-performance open-source messaging system designed for cloud-native applications, IoT messaging, and microservices architectures.

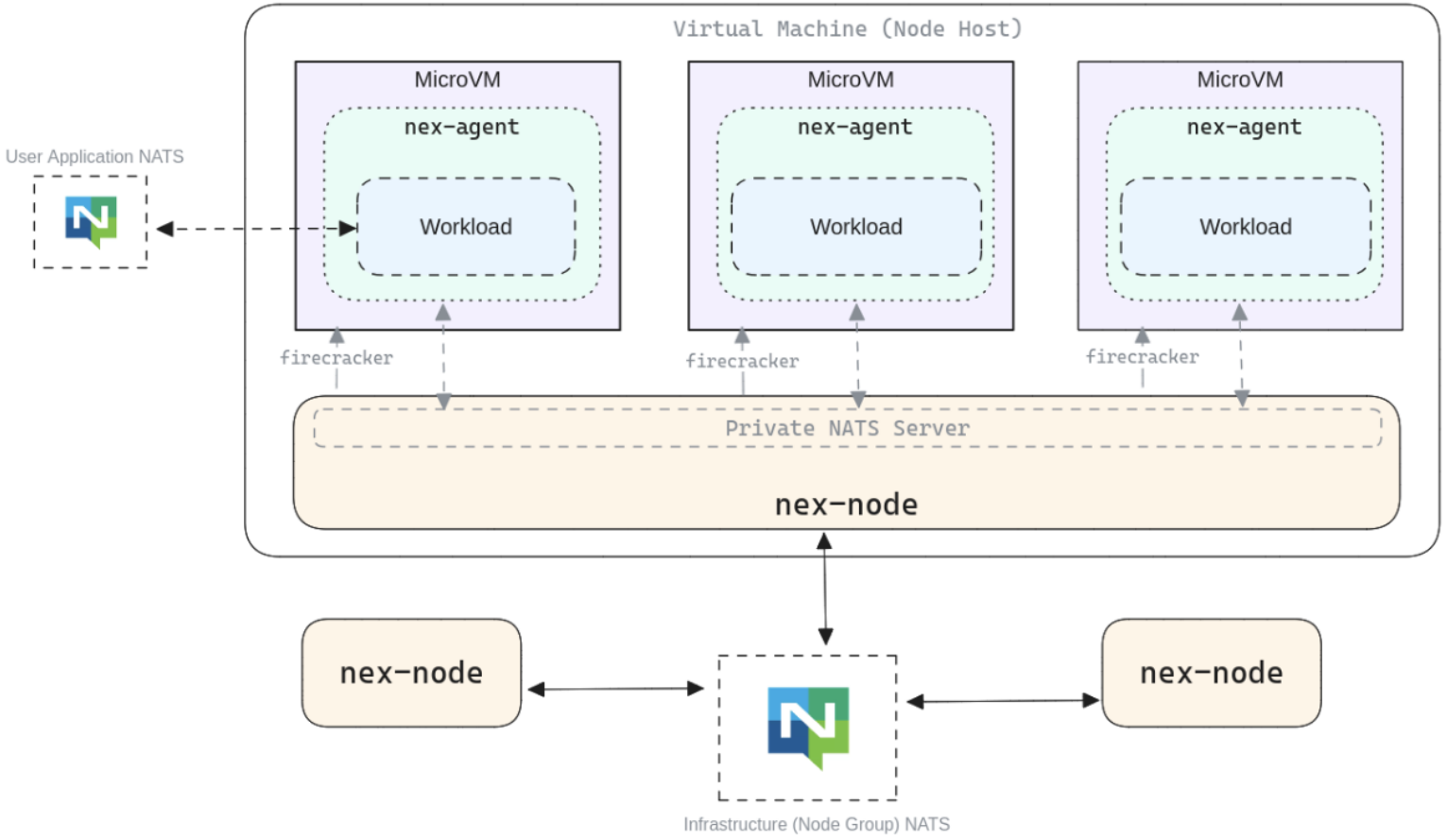
Key Characteristics:

- **Lightweight & Fast**: Written in Go, NATS offers minimal overhead with high throughput
- **Simple Protocol**: Text-based protocol that's easy to understand and implement
- **Cloud-Native**: Built for modern distributed systems and containerized environments
- **Multiple Messaging Patterns**: Supports publish-subscribe, request-reply, and queue groups
- **Location Transparency**: Clients don't need to know where services are located

NATS Ecosystem:

- **NATS Core**: Basic pub-sub messaging
- **NATS JetStream**: Persistence, streaming, and exactly-once delivery
- **NATS KV**: Distributed key-value store
- **NATS Object Store**: Distributed object storage

2. Architecture of NATS



architecture diagram

Architecture

NATS is a publish/subscribe message oriented middleware with an emphasis on simplicity, performance, security, and scalability.

It provides **at-most-once delivery** and supports three primary messaging patterns:
Publish/Subscribe (1:N) – Broadcast messages to multiple subscribers.
Request/Reply (1:1) – RPC-style communication.
Queue Groups – Load-balanced message processing among multiple consumers.

3. Use Cases & Problems NATS Solves

Service Communication Complexity

- **Problem**: Direct service-to-service communication creates tight coupling
- **Solution**: NATS provides a decoupled communication layer between services

Scalability Challenges

- **Problem**: Traditional messaging systems struggle with massive scale
- **Solution**: NATS handles millions of messages per second with low latency

IoT & Edge Computing

- Real-time telemetry from millions of IoT devices
- Edge-to-cloud data streaming
- Device command and control
- **Example**: Smart city sensors sending traffic data to central systems

4. Pros & Cons: Where NATS Excels and Areas for Improvement

Pros - Where NATS Excels:

Performance

- **Ultra-low latency**: Sub-millisecond message delivery
- **High throughput**: Millions of messages per second on commodity hardware
- **Minimal resource footprint**: ~10MB memory for core server

Cons - Areas Needing Improvement:

Limited Enterprise Features

- **No GUI management console**: Relies on CLI and third-party tools
- **Basic monitoring**: Requires external tools (Prometheus, Grafana) for comprehensive monitoring
- **Limited commercial support options**: Compared to enterprise messaging systems

5. Comparison Against Alternative Technologies

NATS vs Apache Kafka

Feature	NATS	Apache Kafka
Primary Use Case	Real-time messaging, microservices	Log aggregation, event streaming
Latency	Sub-millisecond	Milliseconds to seconds
Throughput	Very high (14M msg/sec)	Very high (millions/sec)
Persistence	Optional (JetStream)	Always (disk-based)
Complexity	Low (single binary)	High (Zookeeper/KRaft, multiple components)

Choose NATS when: You need ultra-low latency, simple deployment, and request-reply patterns
Choose Kafka when: You need guaranteed ordering, large-scale data pipelines, and complex stream processing

6. Conclusion

For Modern Distributed Systems, NATS represents an **excellent choice** that balances:

- **Performance** without sacrificing simplicity
- **Open source** without compromising enterprise features
- **Cost efficiency** without limiting scalability
- **Innovation** without breaking stability

Start with NATS if:

- You're building new cloud-native applications
- You value operational simplicity
- You need performance and low latency
- You want to avoid vendor lock-in

The combination of **zero licensing costs**, **low operational overhead**, and **high performance** makes NATS a compelling foundation for modern, distributed application architectures.

THANK YOU