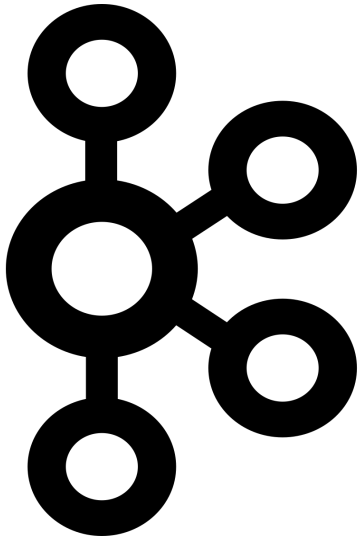


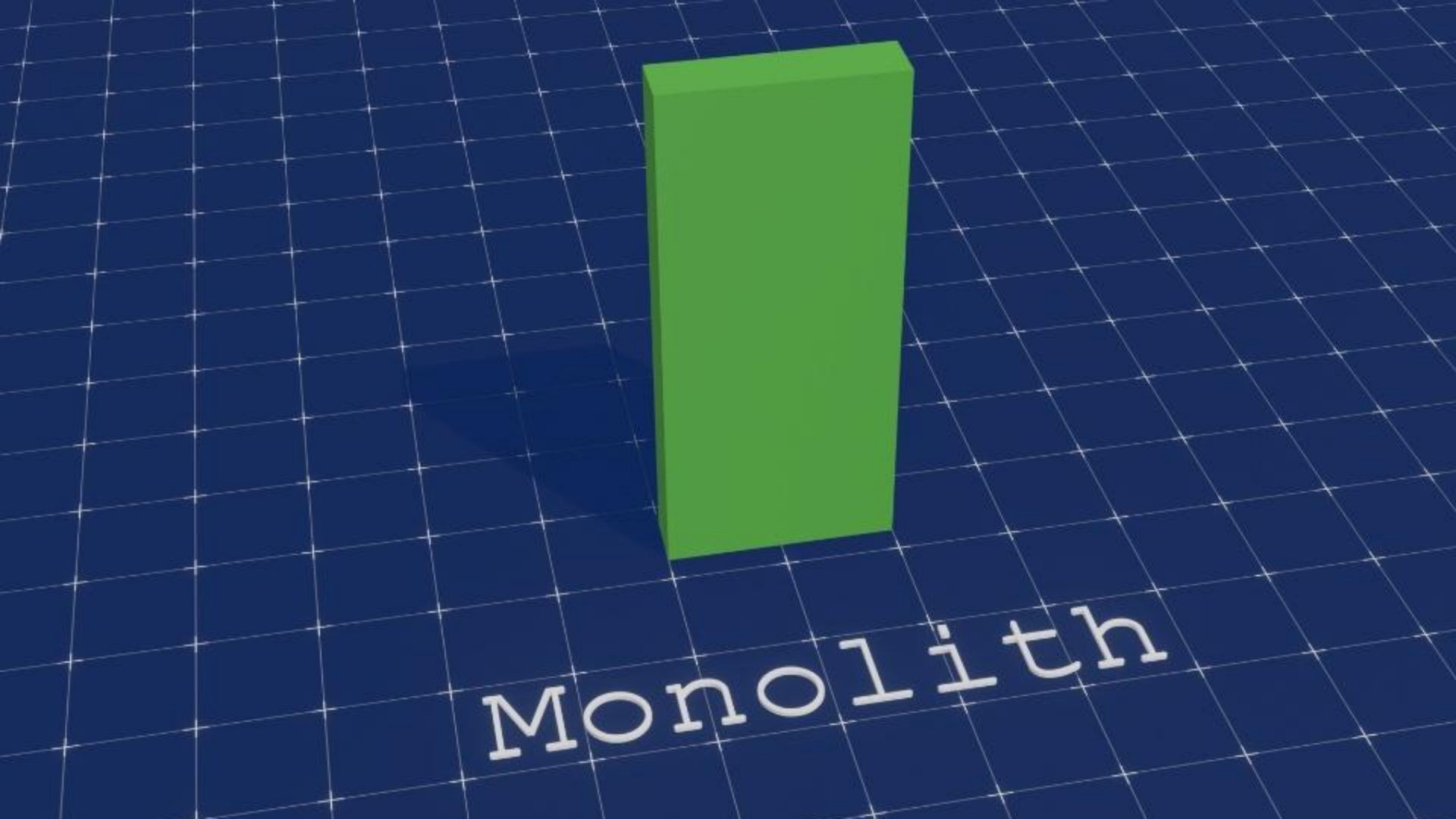
Apache Kafka

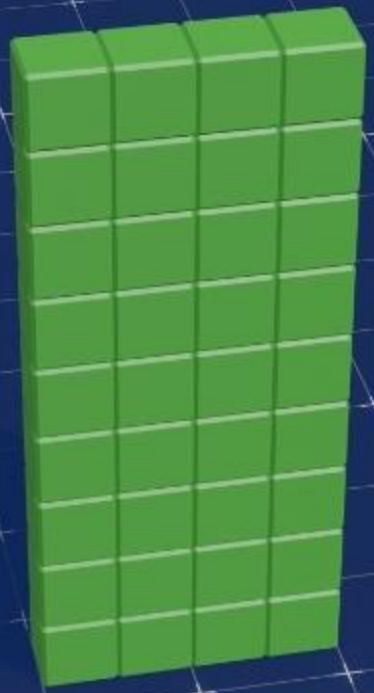
Introduction

An Evolving System will require change

Blocker:

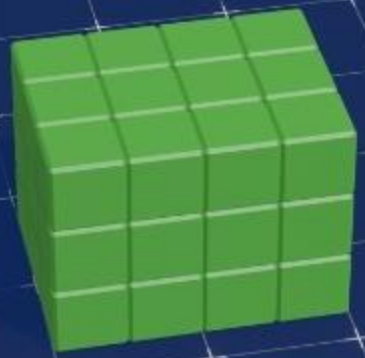
The Monolith Arch


Monolith



Modules

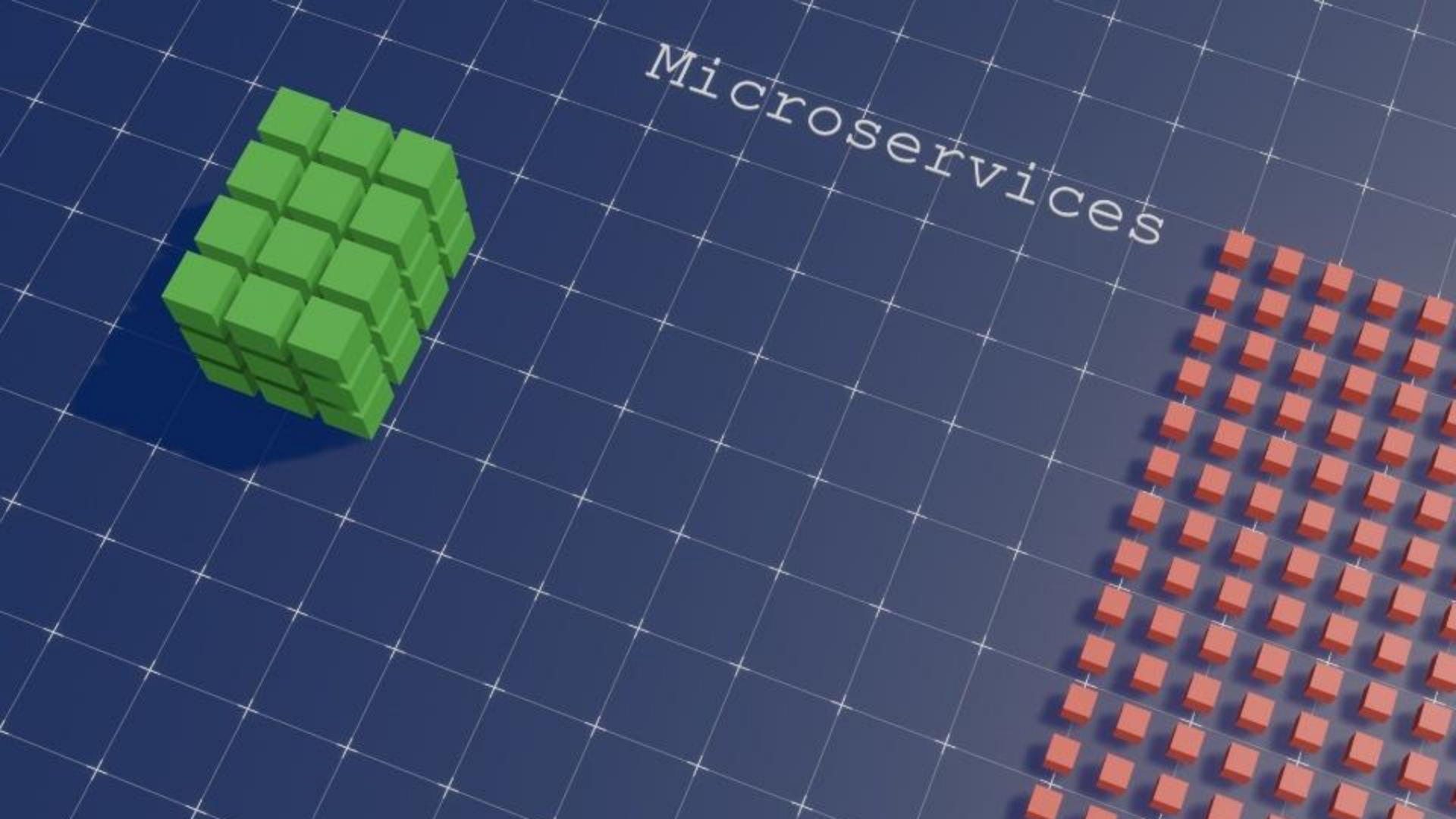
Monolith

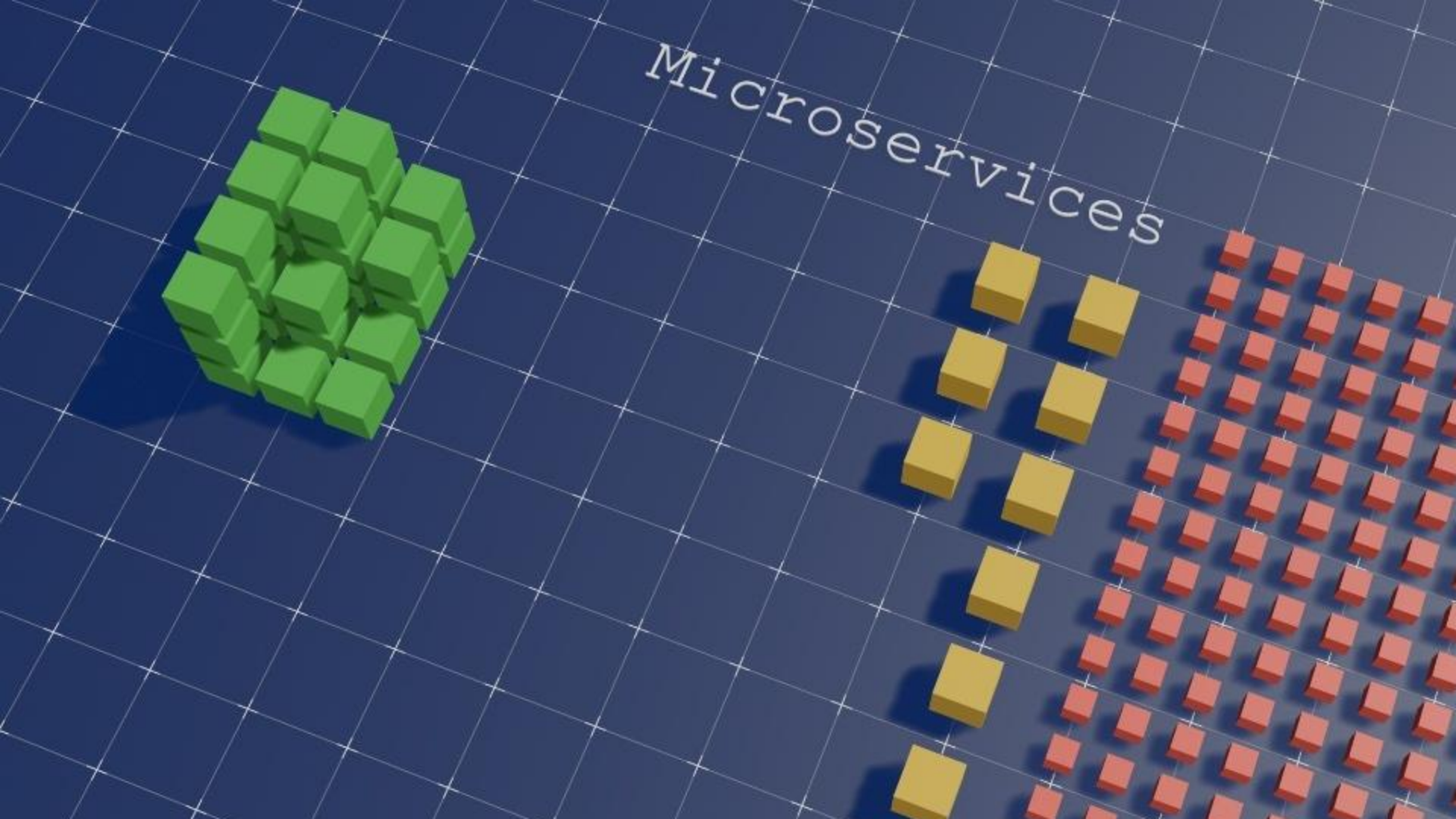


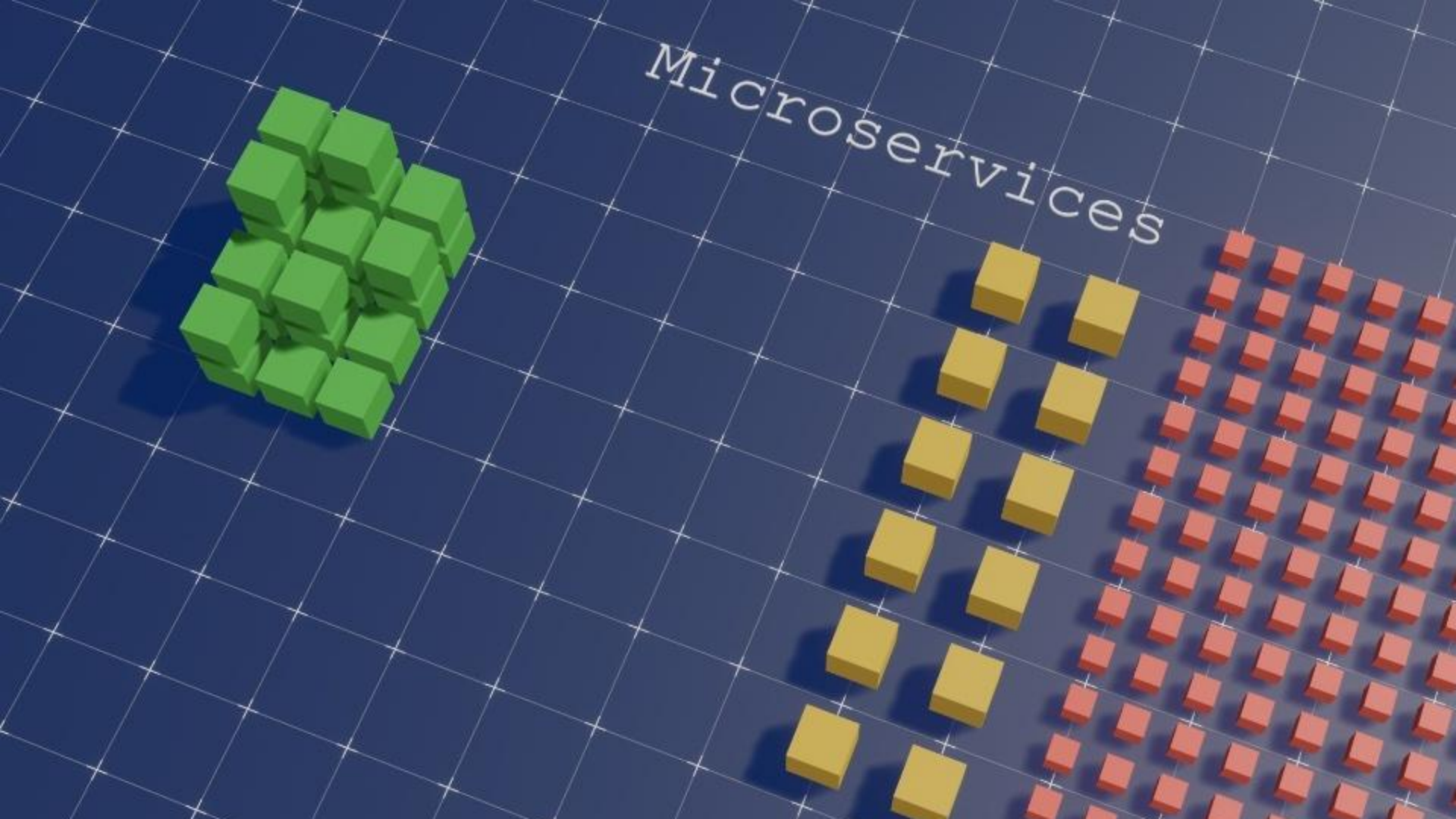
Modules

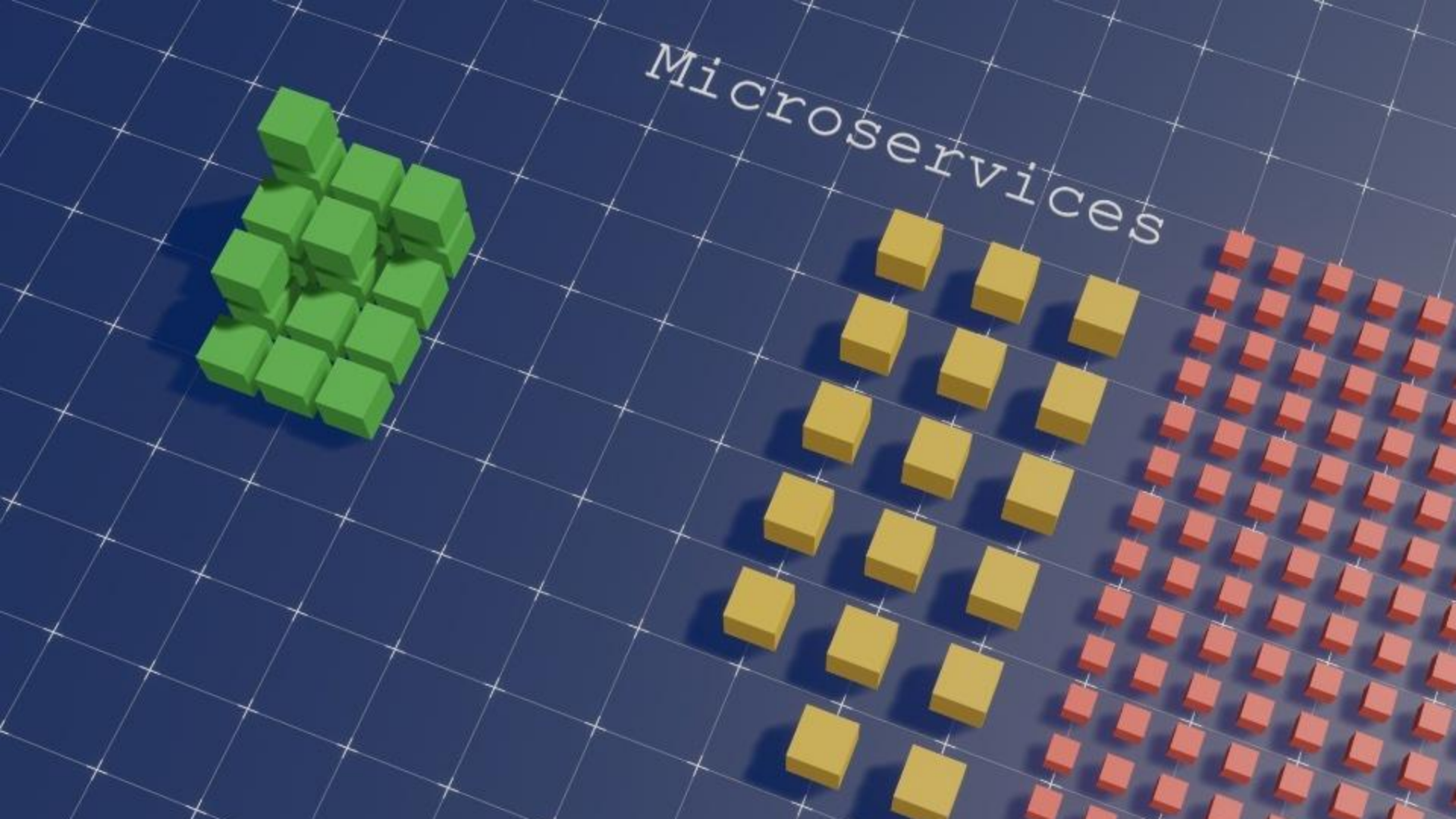
Monolith

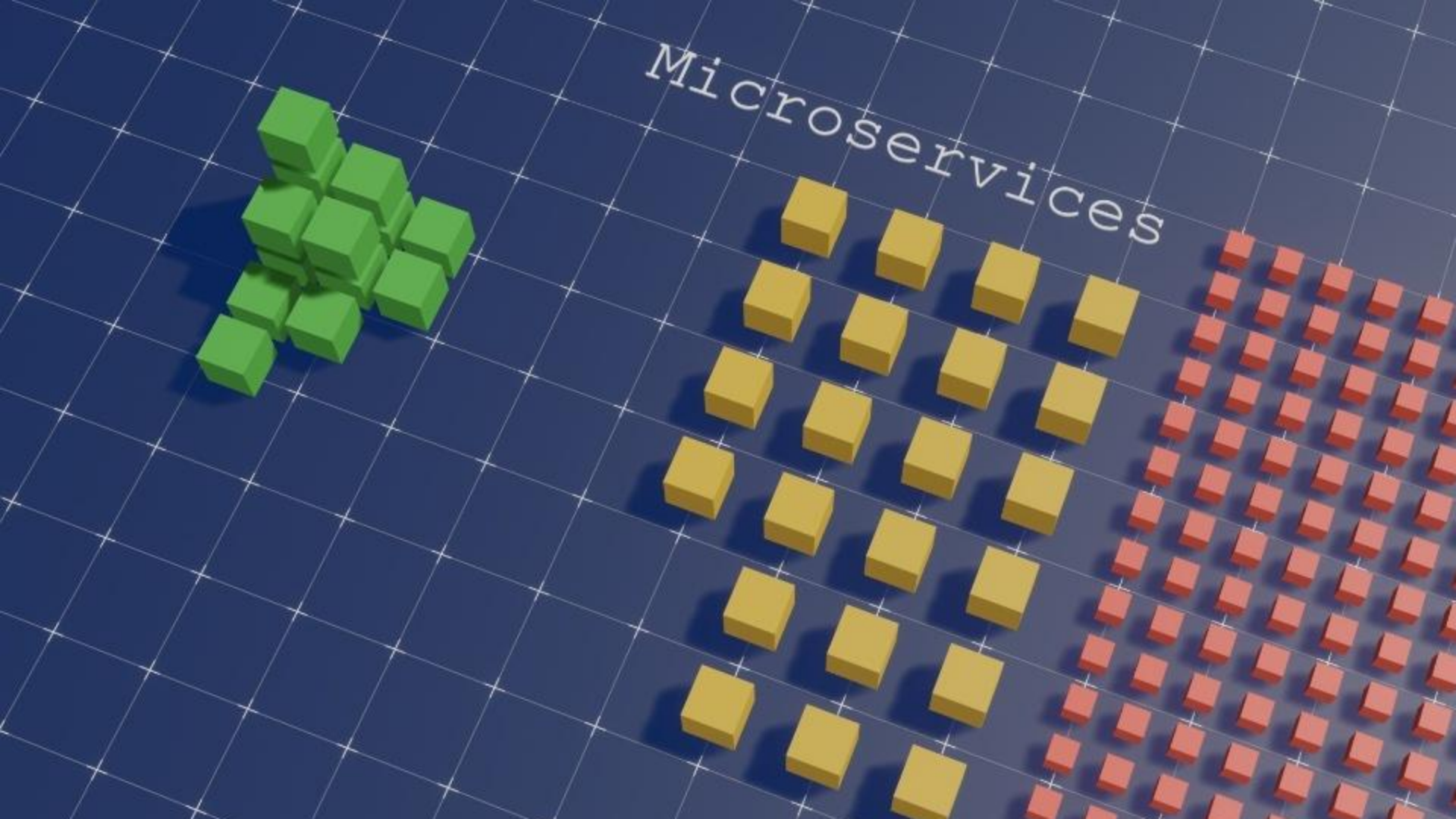
Solution:
The Distributed Systems

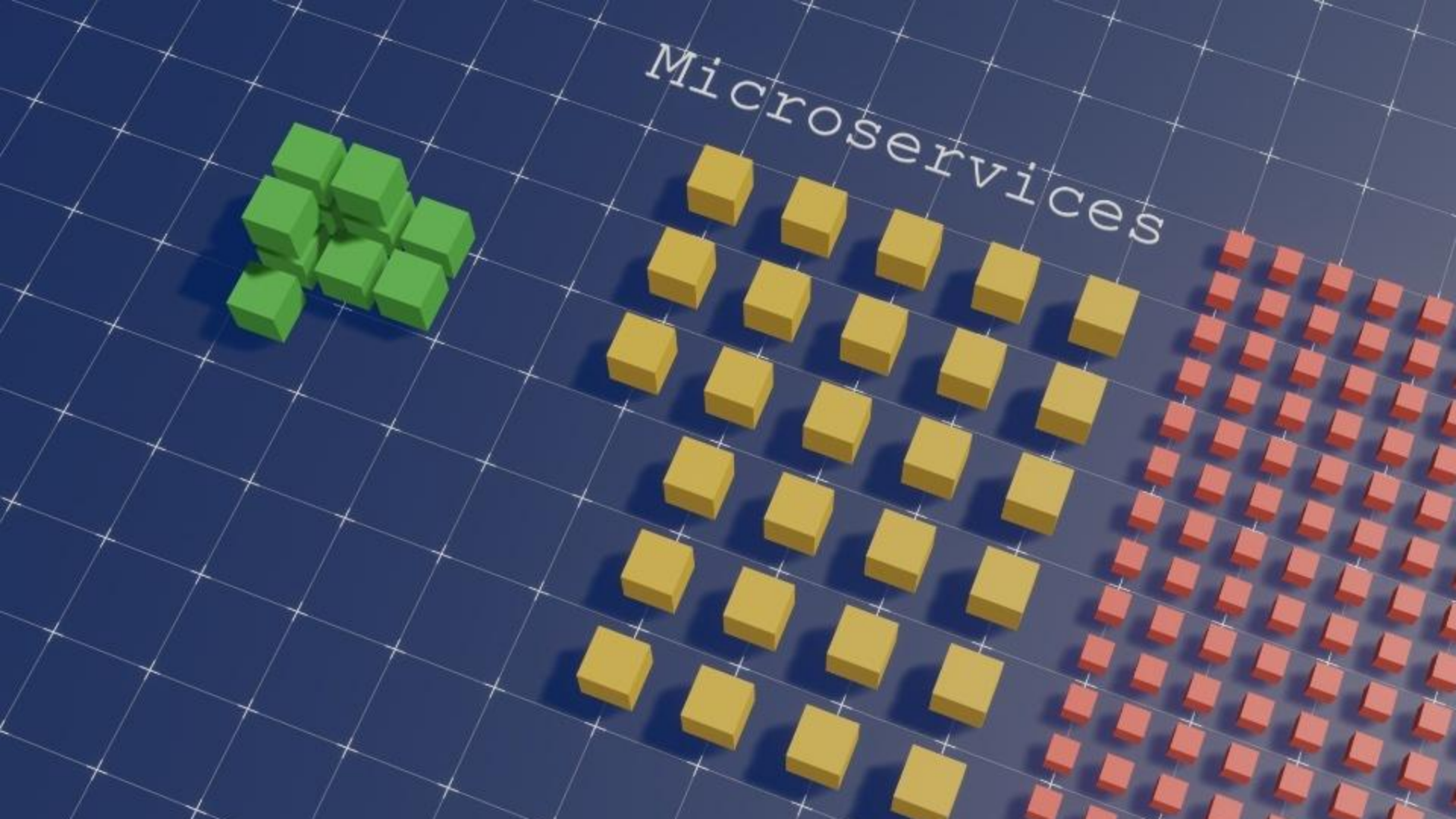

Microservices

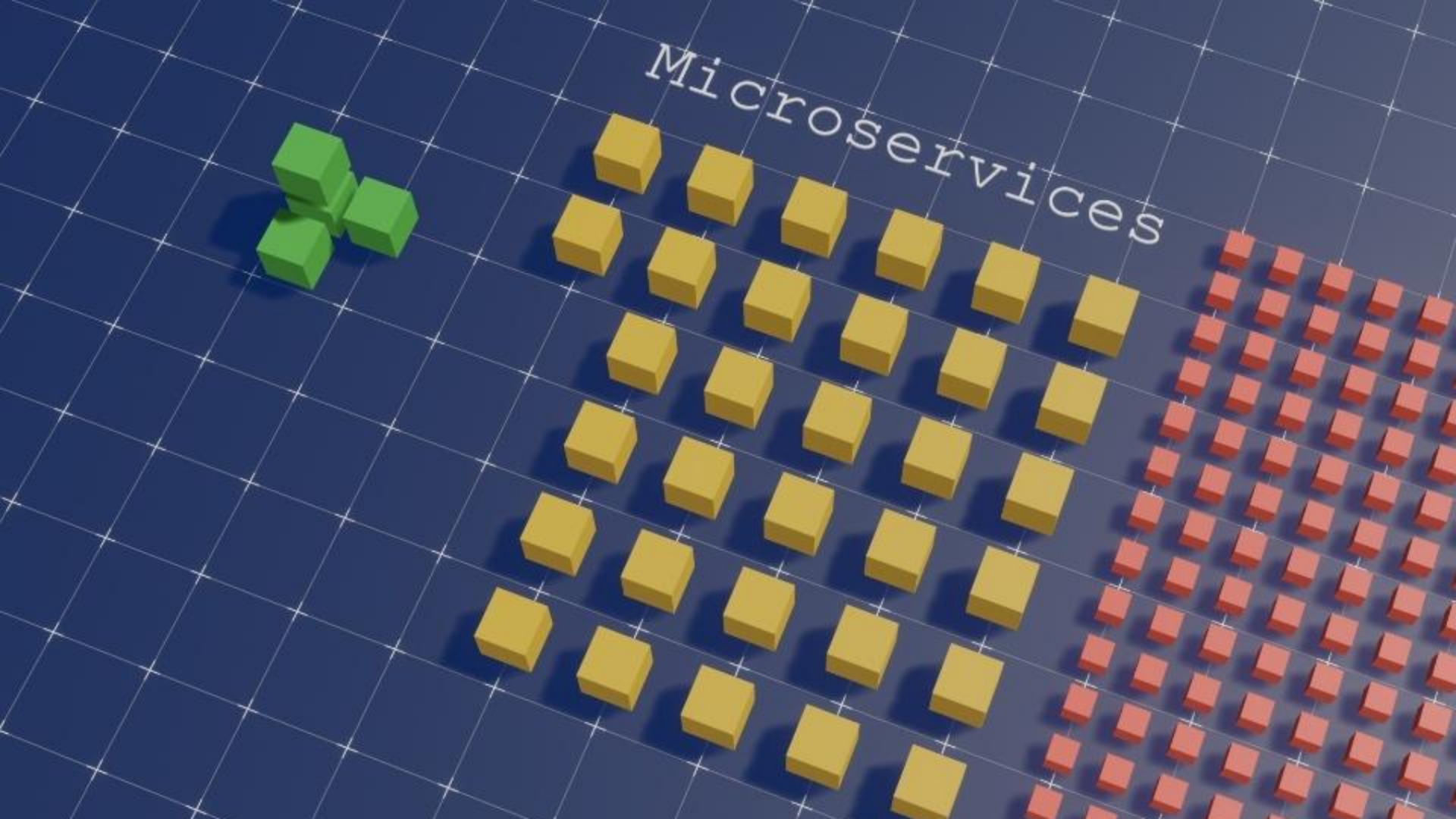

Microservices

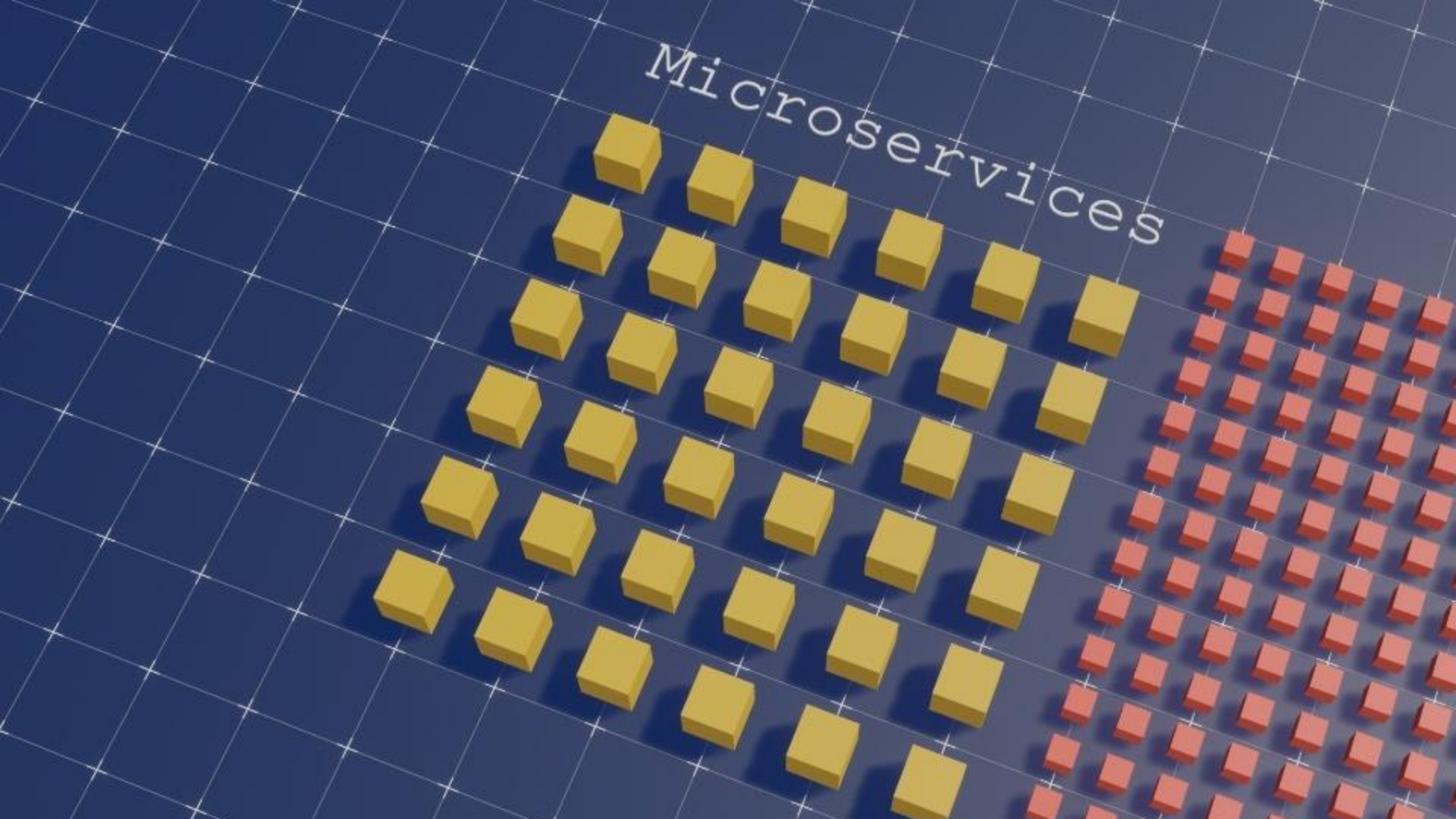

Microservices

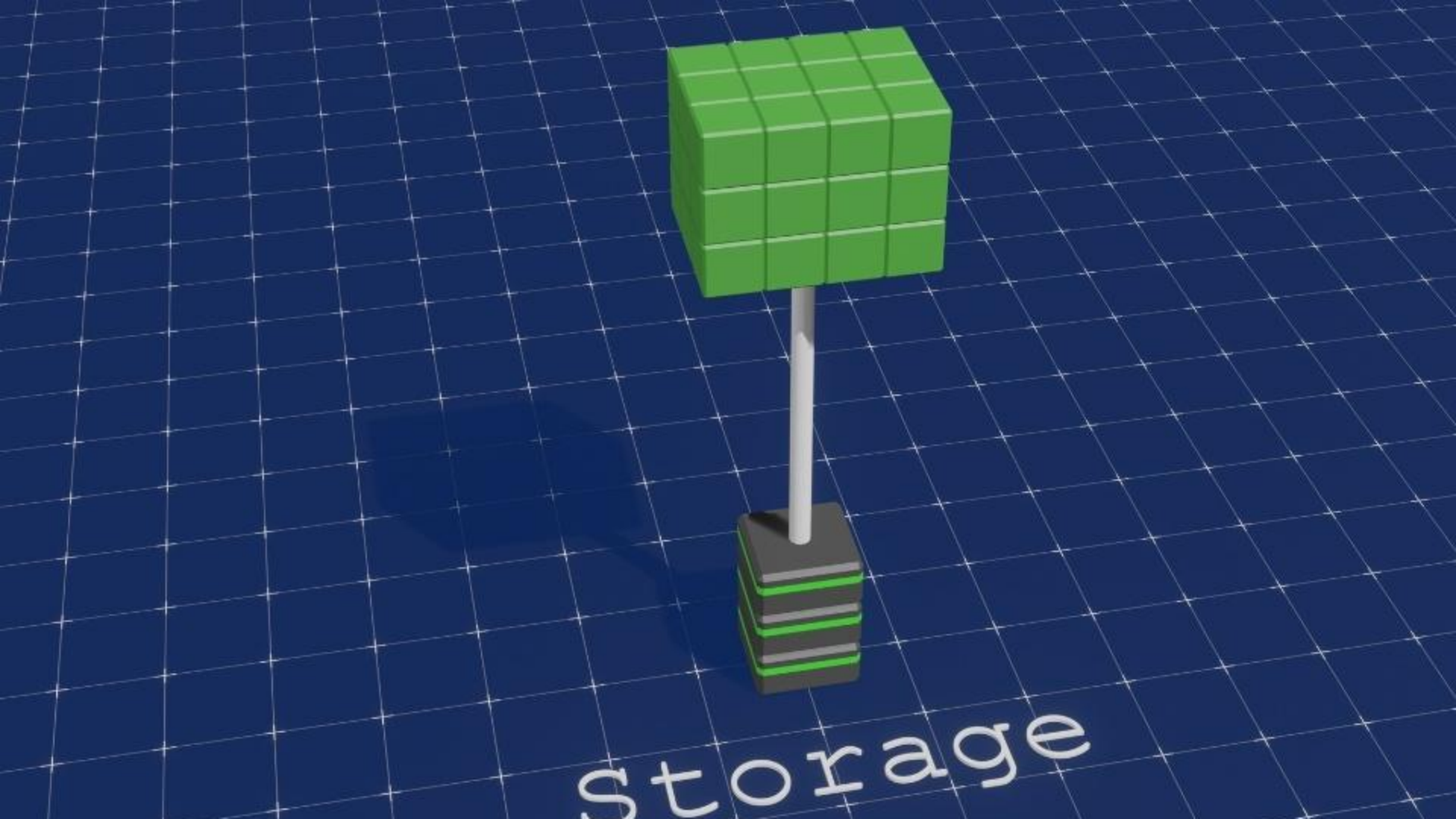

Microservices

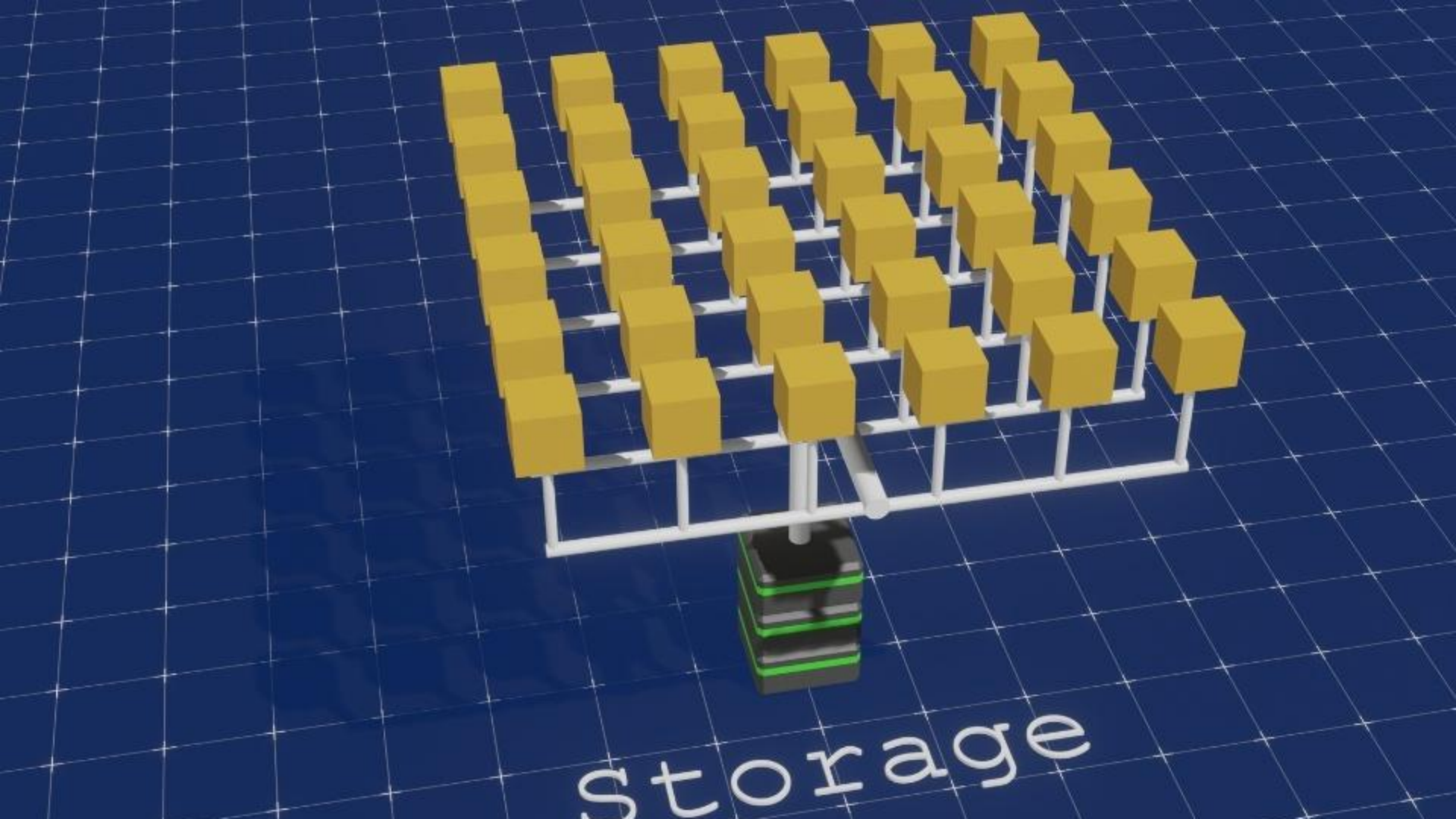

Microservices

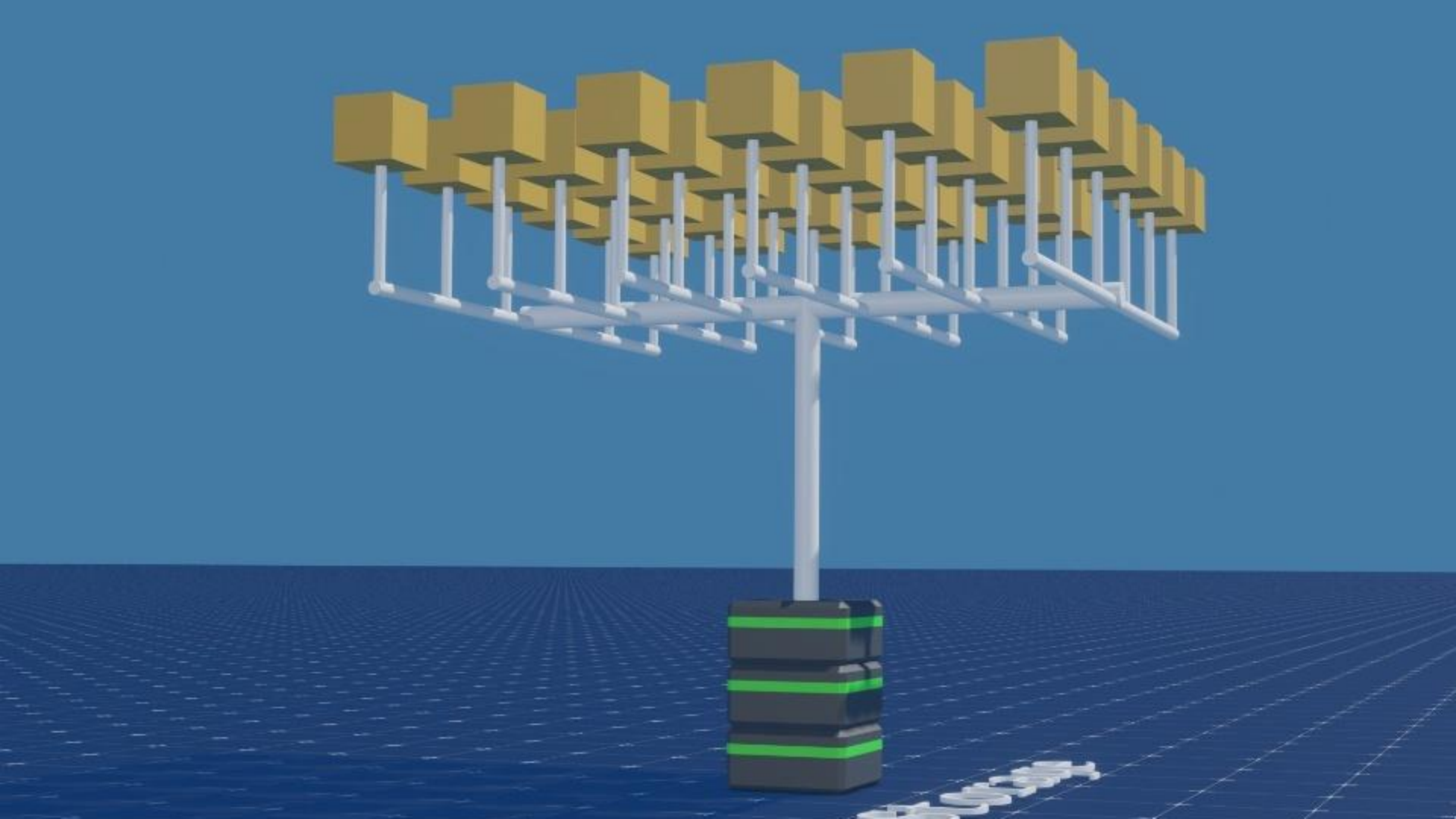

Microservices

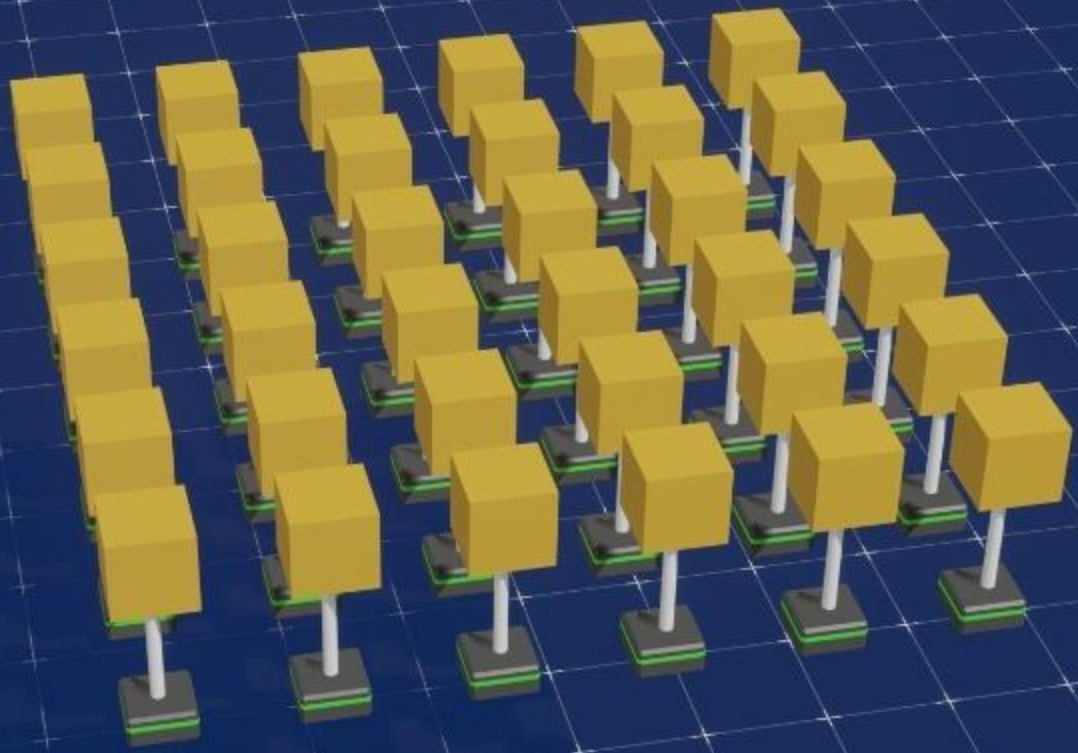

Microservices


Microservices

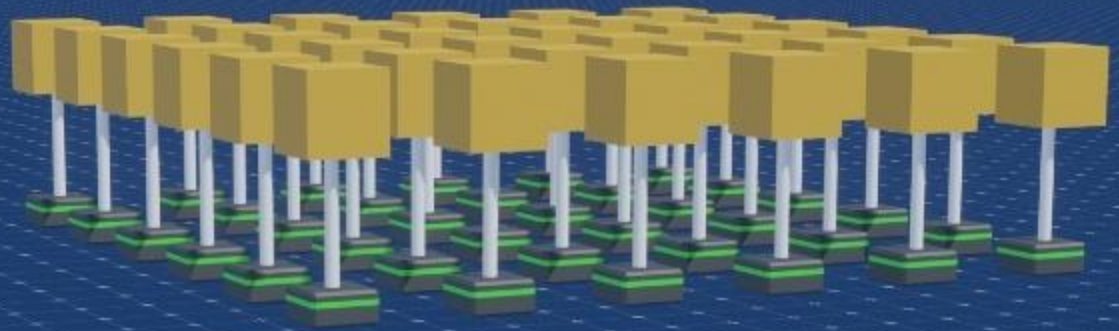

Storage


Storage

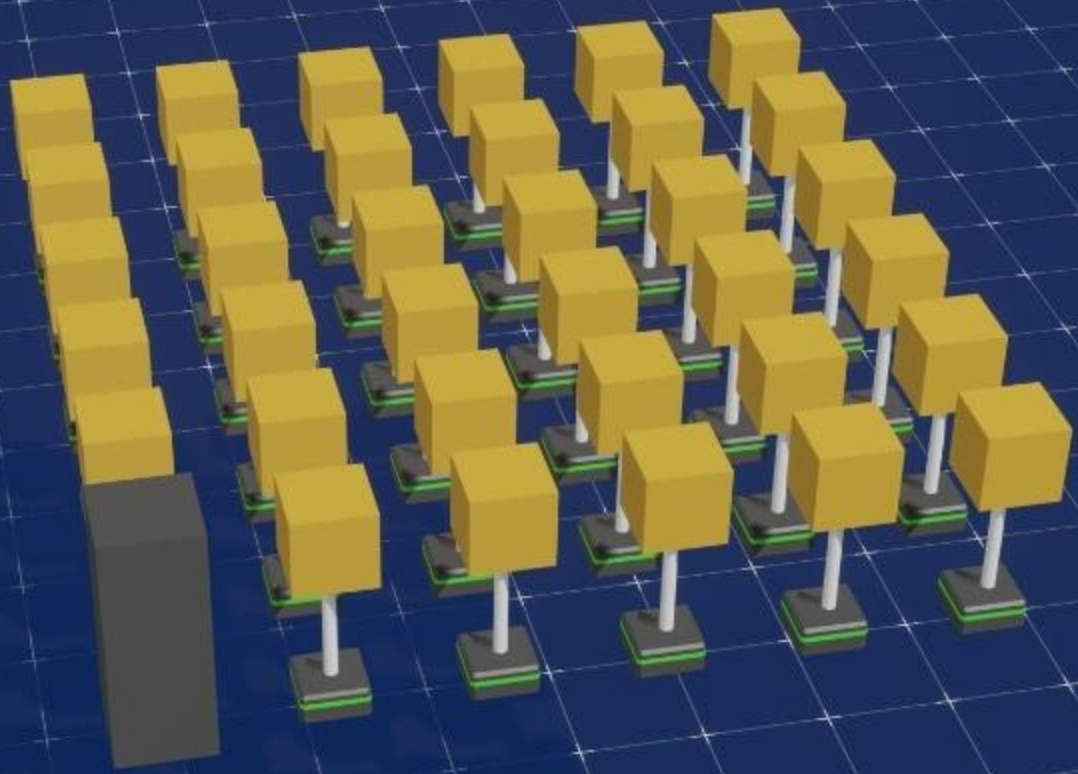




loosley coupled

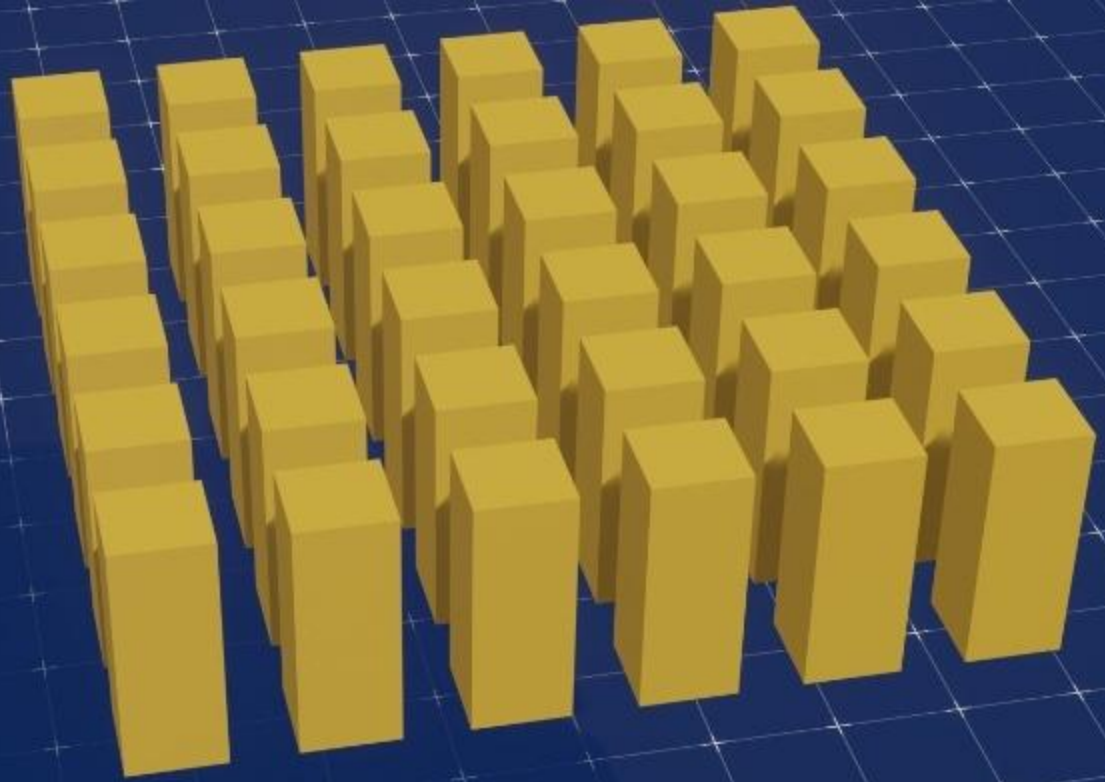


loosley coupled



loosley coupled

loosley coupled



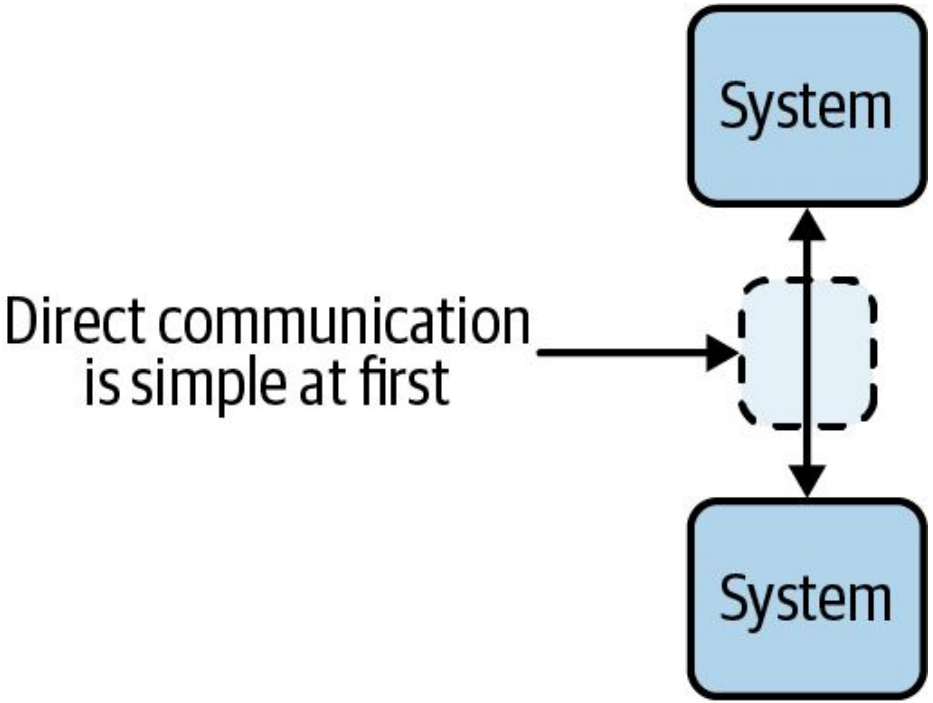
loosley coupled

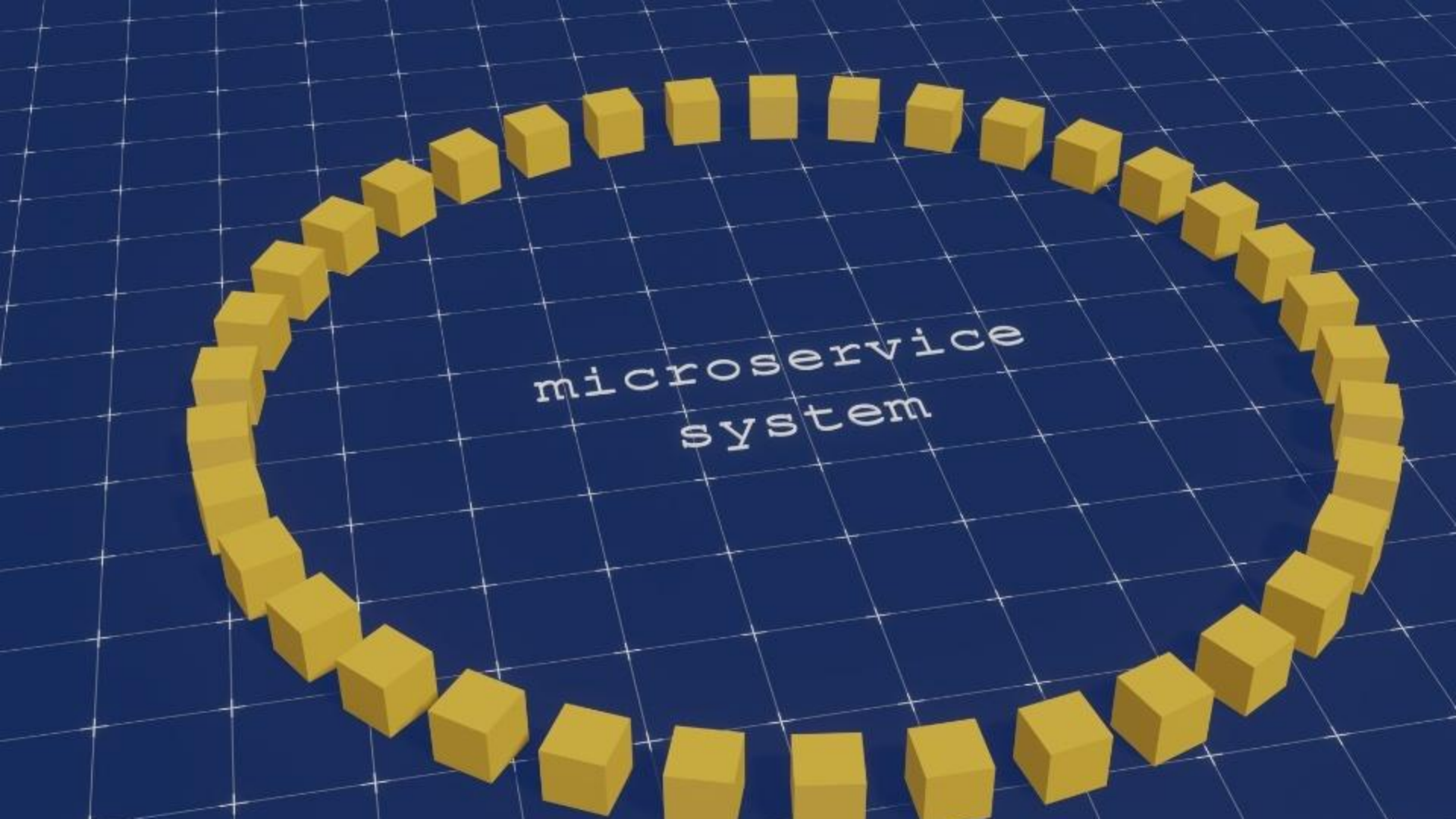
Traditional Communication Model

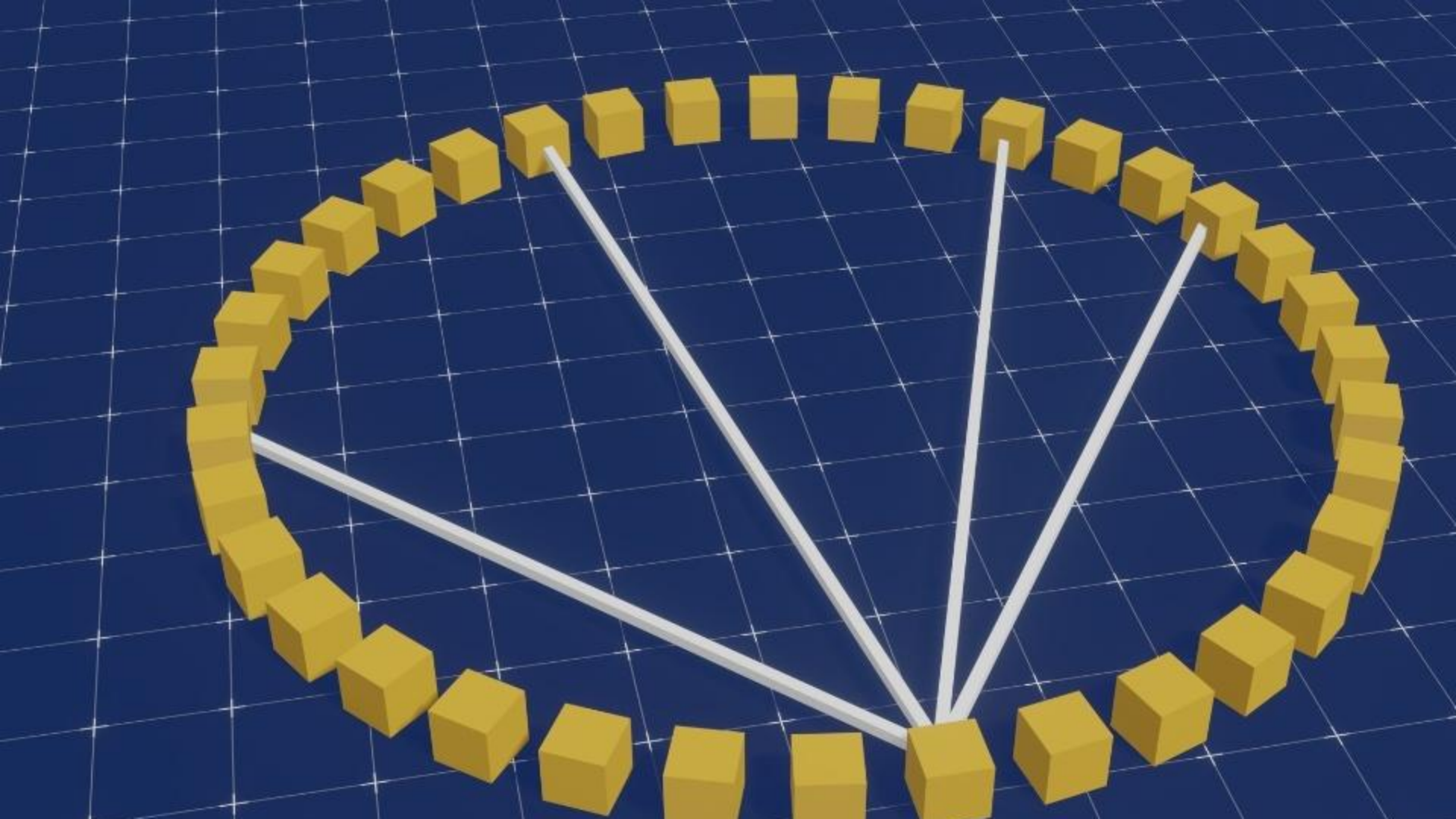
Client-Server Communication

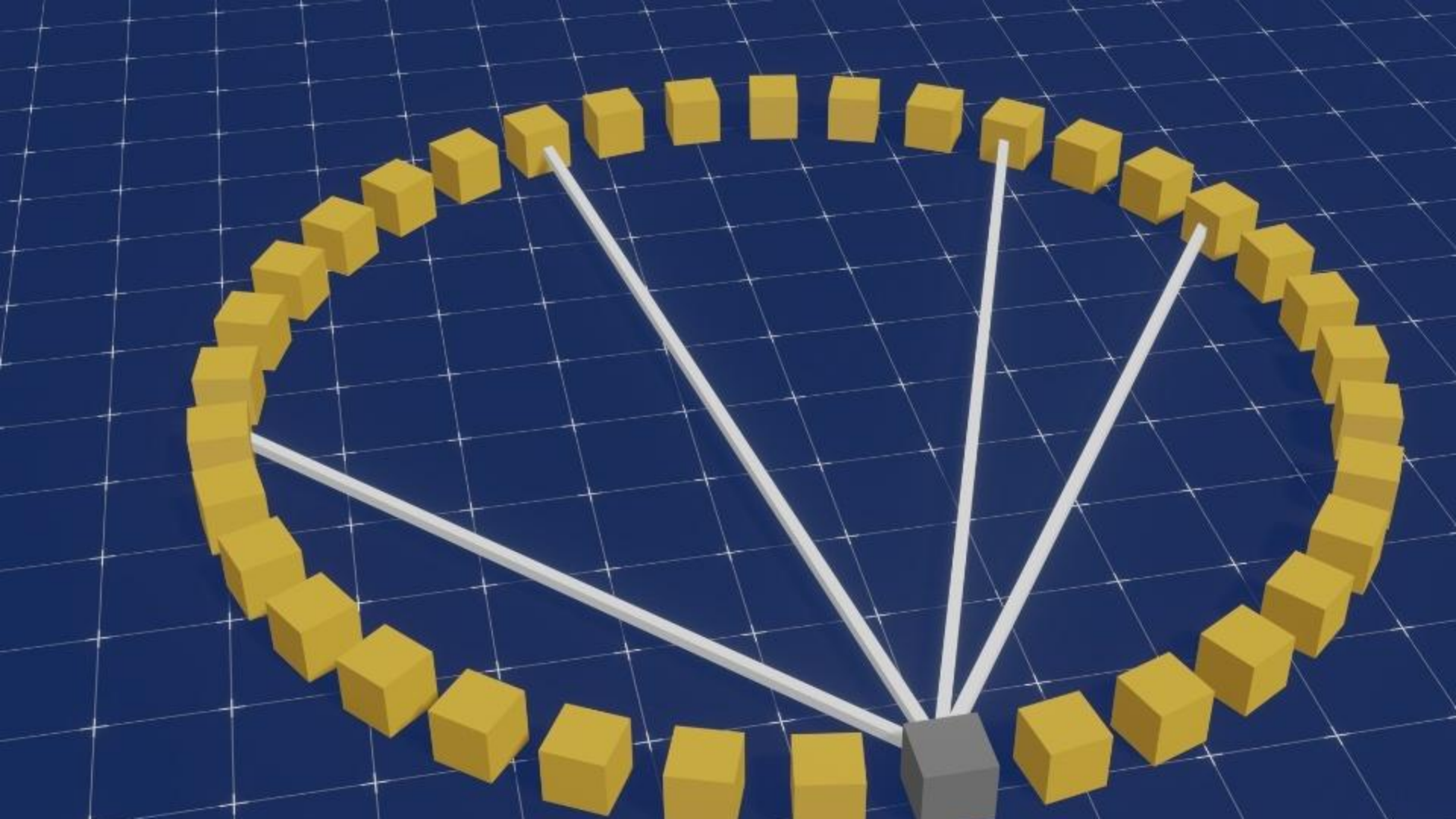
The Client-Server Communication Model

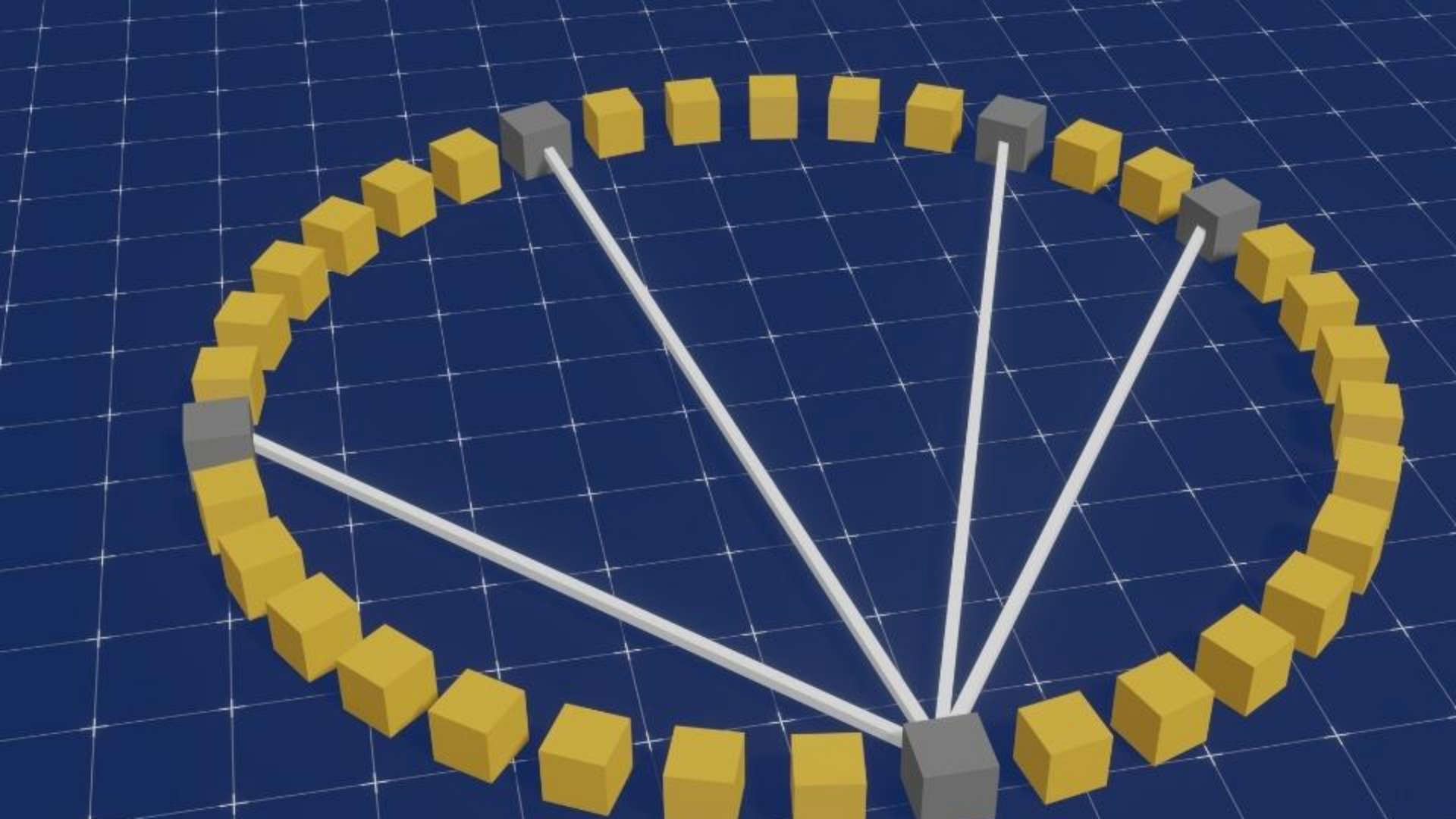
- Common in applications, microservices, and databases
- Simple and direct but becomes complex with scale



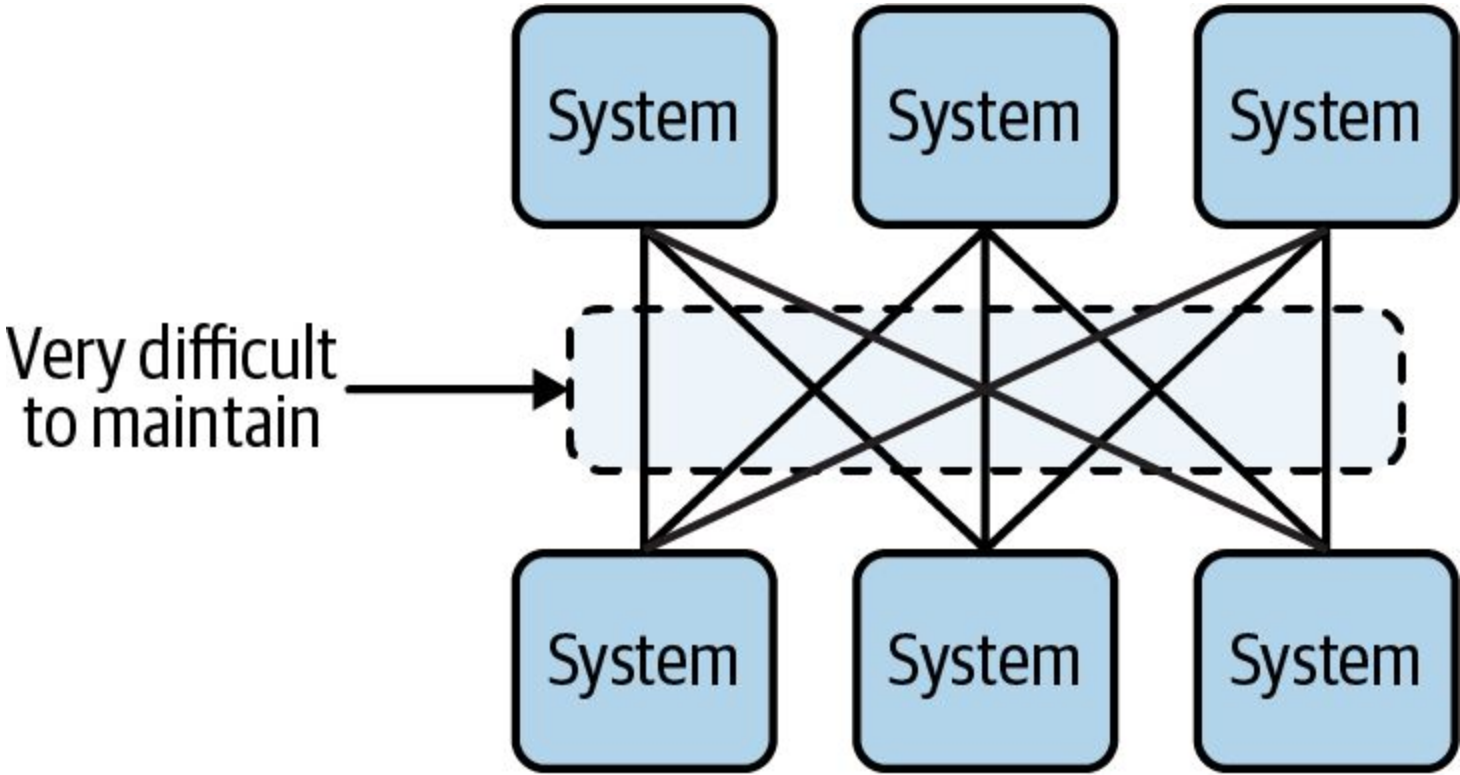

microservice
system







Challenges of scaling point-to-point communications



Drawbacks of Client-Server Model

- Tightly coupled systems, difficult maintenance and updates
- Synchronous pitfalls: No delivery guarantees if a system goes offline
- Communication inconsistencies: Different protocols and strategies
- Data flow issues: Overwhelming receiving systems, lack of replayability

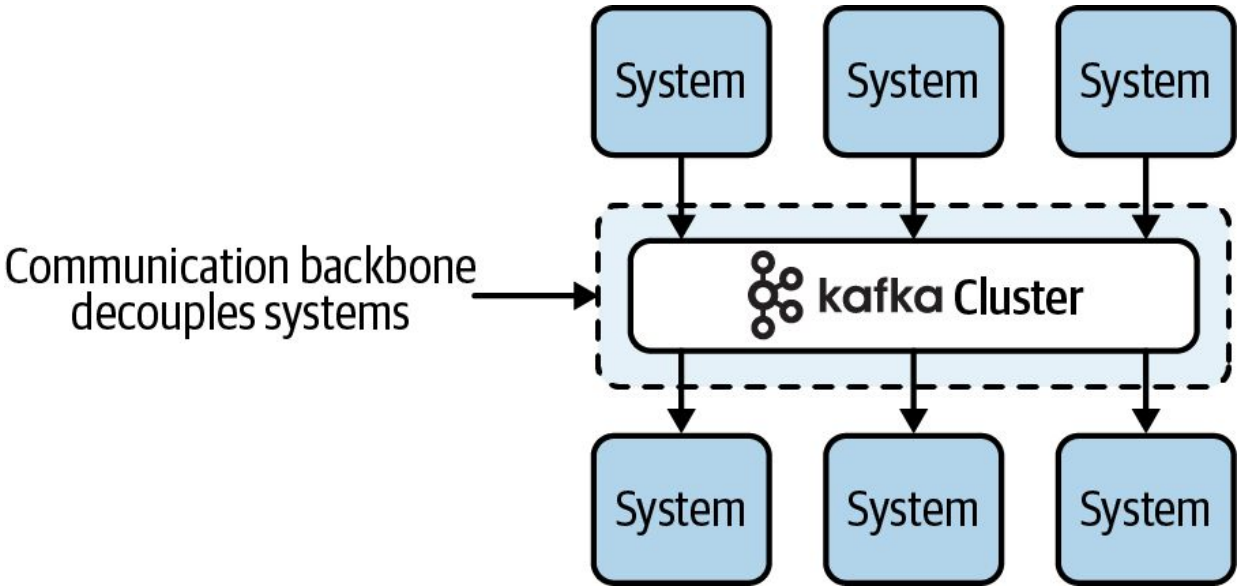
Client-Server Model - challenges

1. Cross language communication
2. Service-discovery
3. Load-balancing
4. Network-latency
5. Reliability
6. Versioning & Compatibility

Introducing Kafka's Communication Model

Kafka as a Central Communication Hub

- Decentralizes and simplifies system communication
- Publish-subscribe model replaces direct interactions

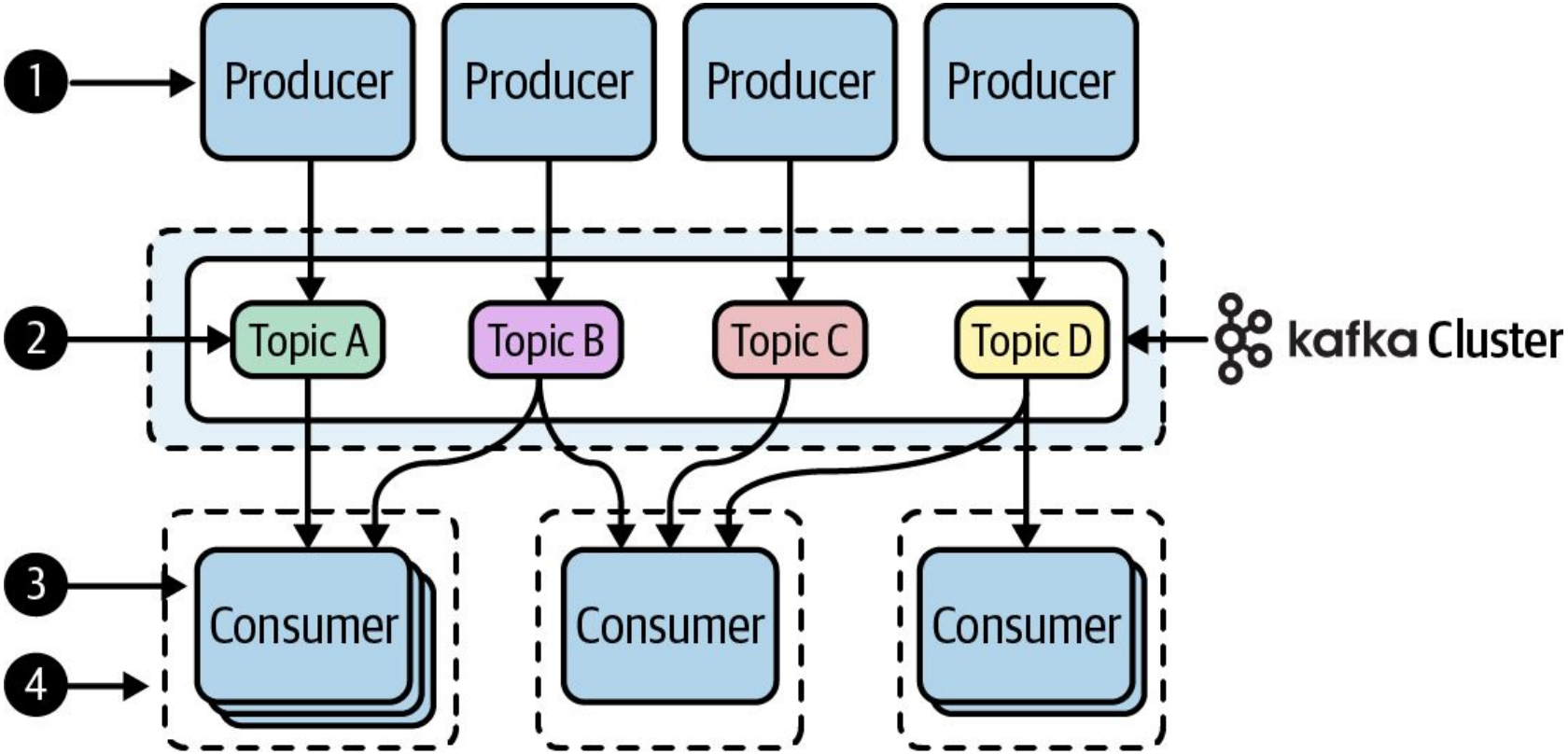


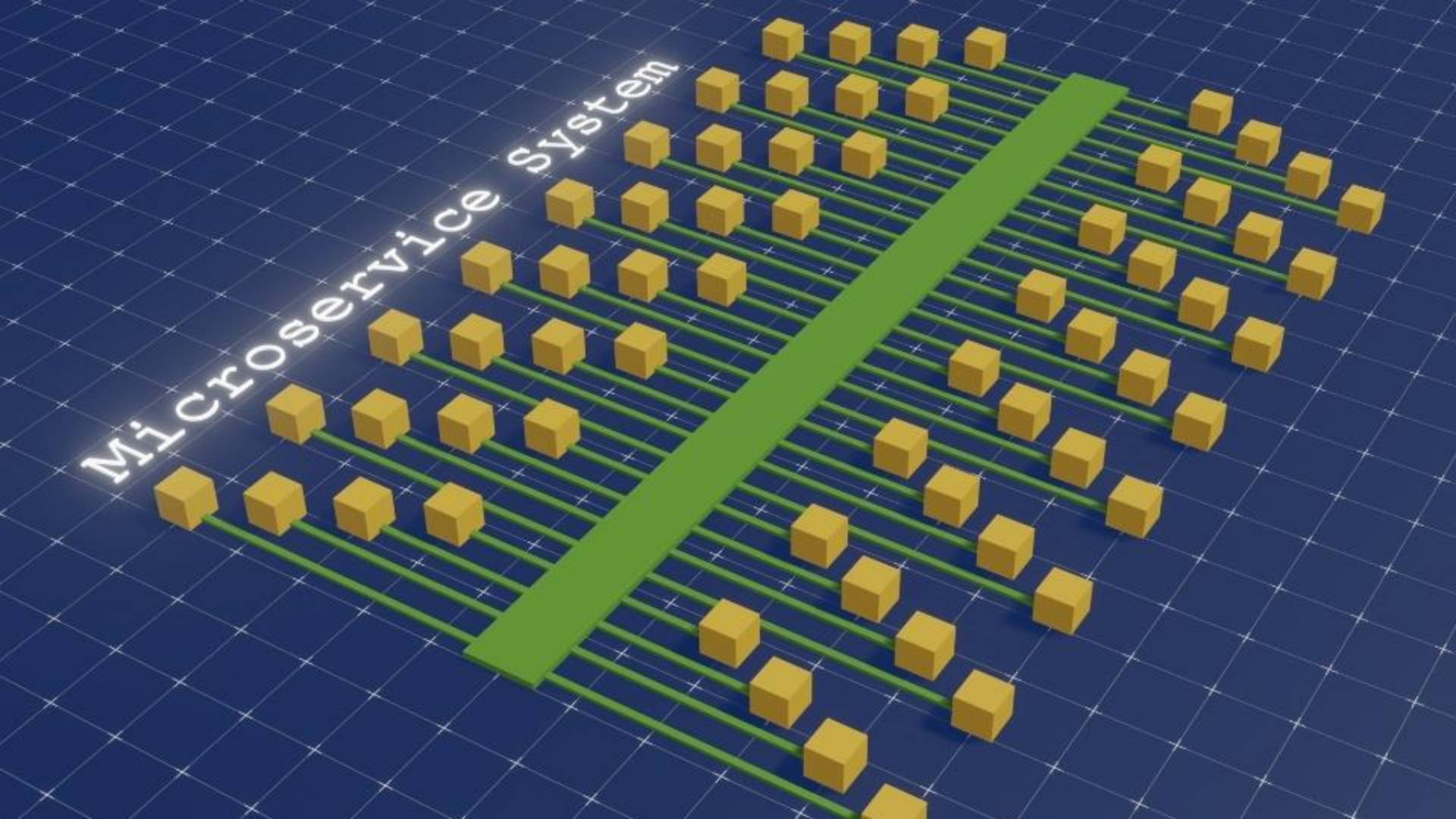
Kafka’s Publish-Subscribe Mechanics

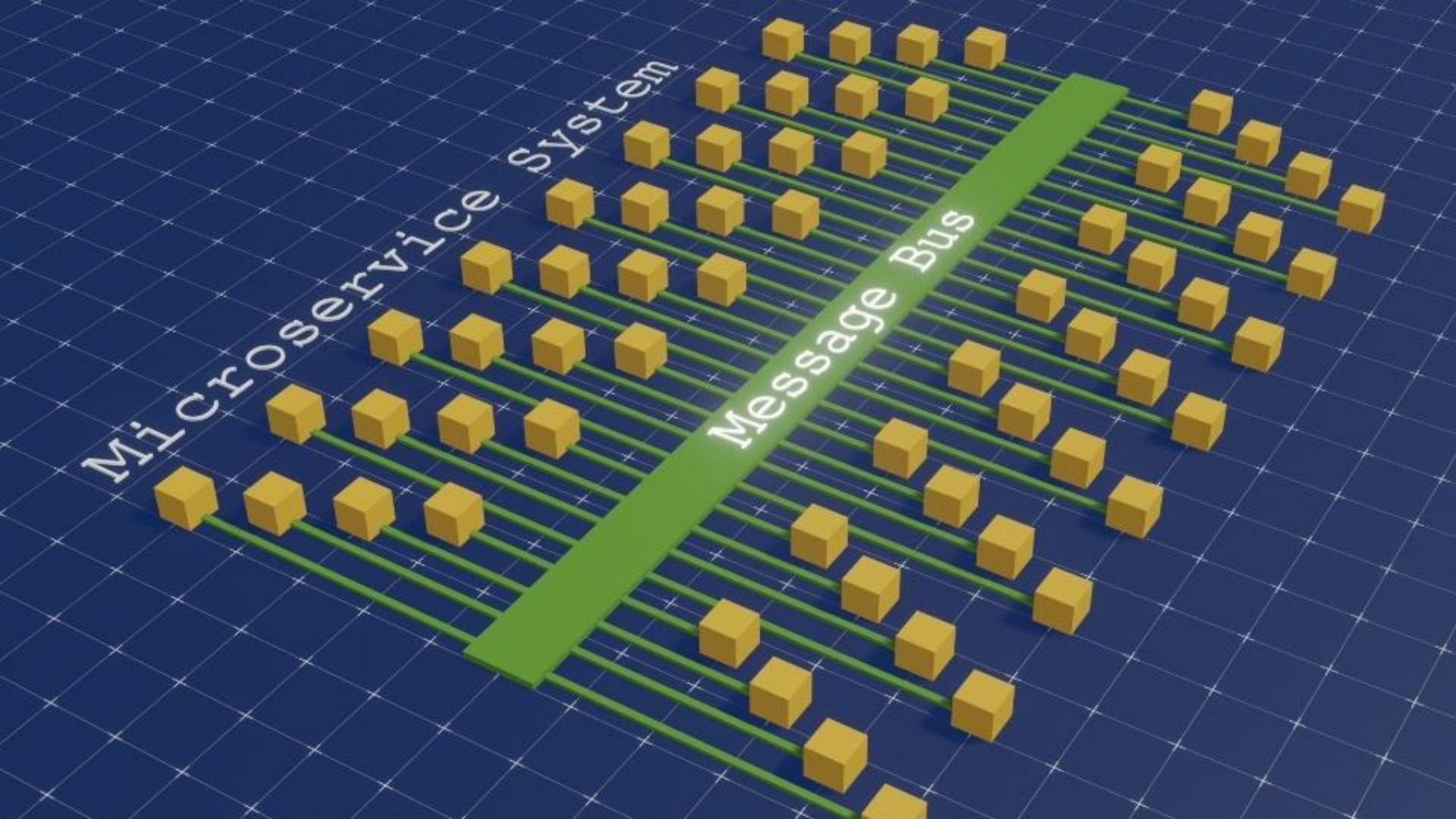
Understanding Kafka's Pub/Sub Model

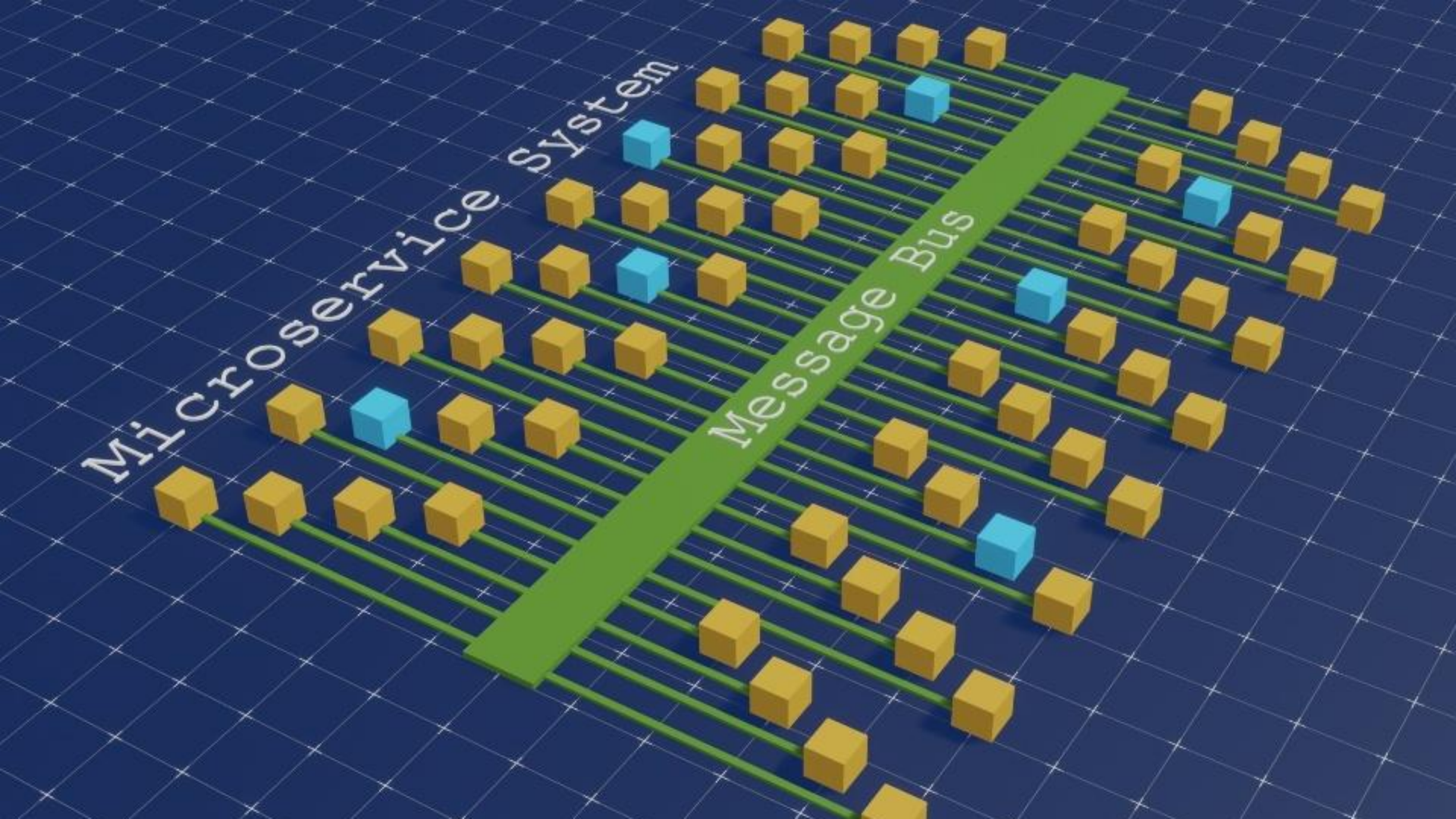
- Producers publish data to topics without knowing the consumers
- Consumers subscribe to topics of interest, decoupling from producers
- Consumer groups for distributed processing and fault tolerance

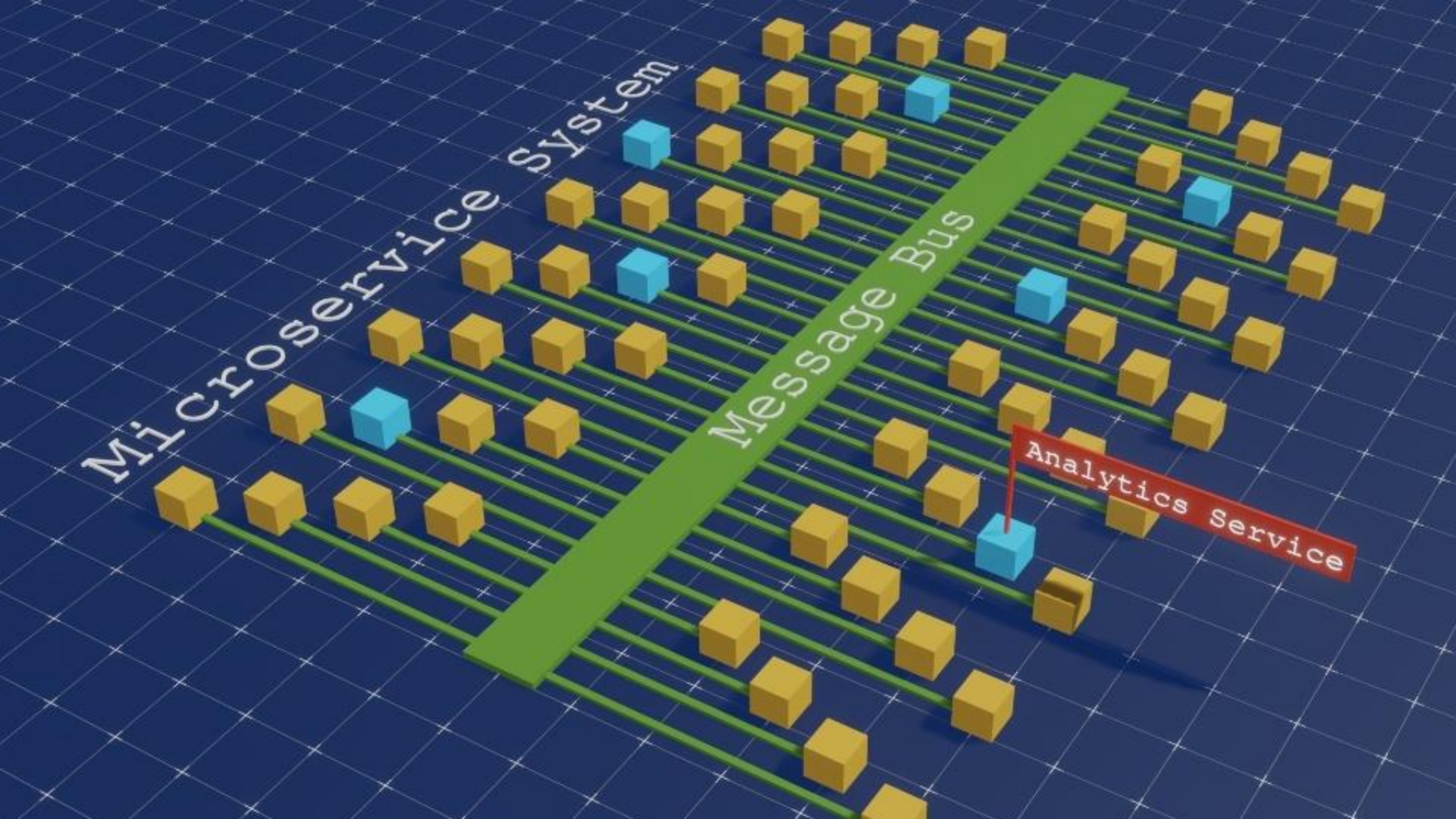
Understanding Kafka's Pub/Sub Model

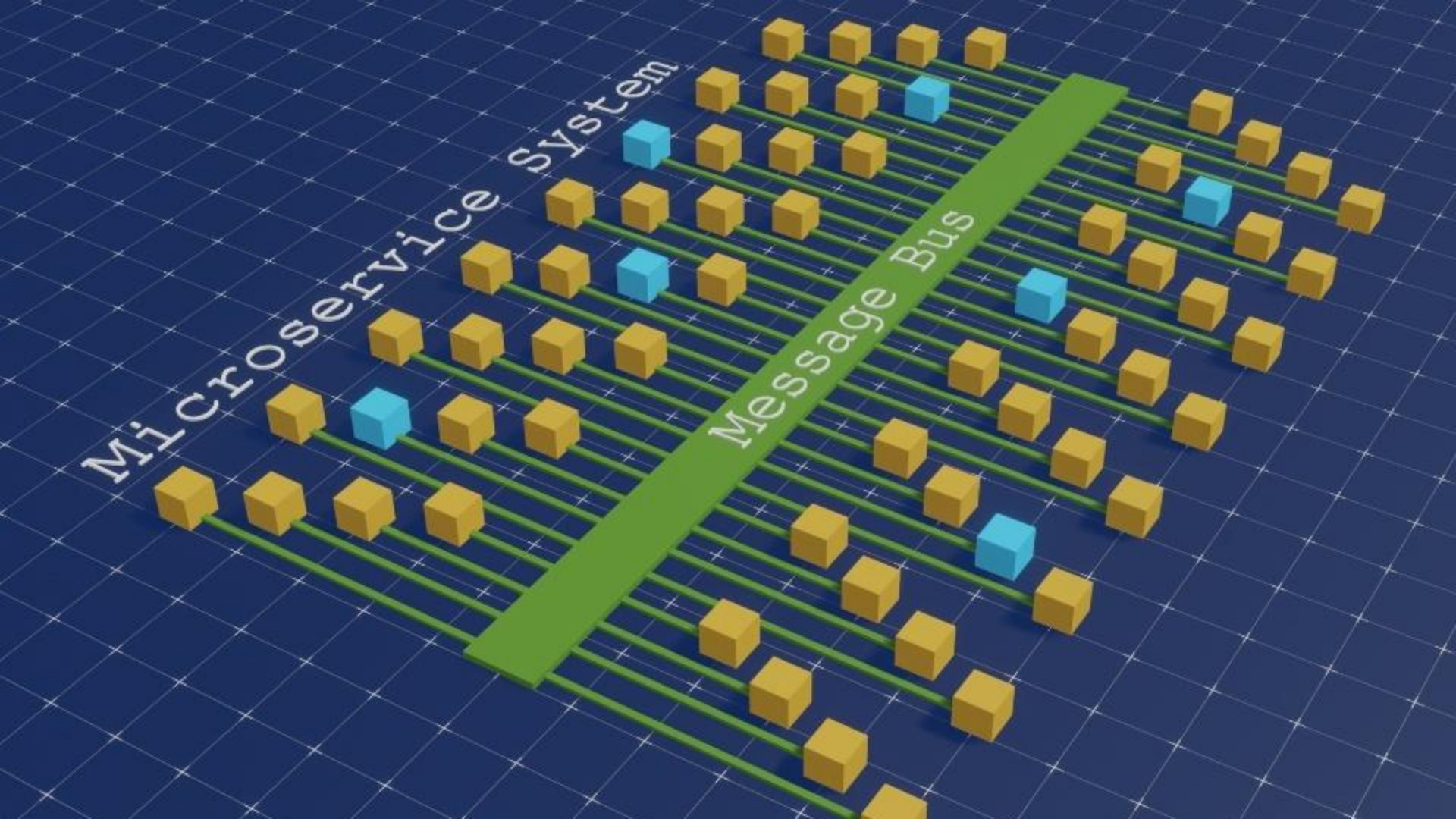


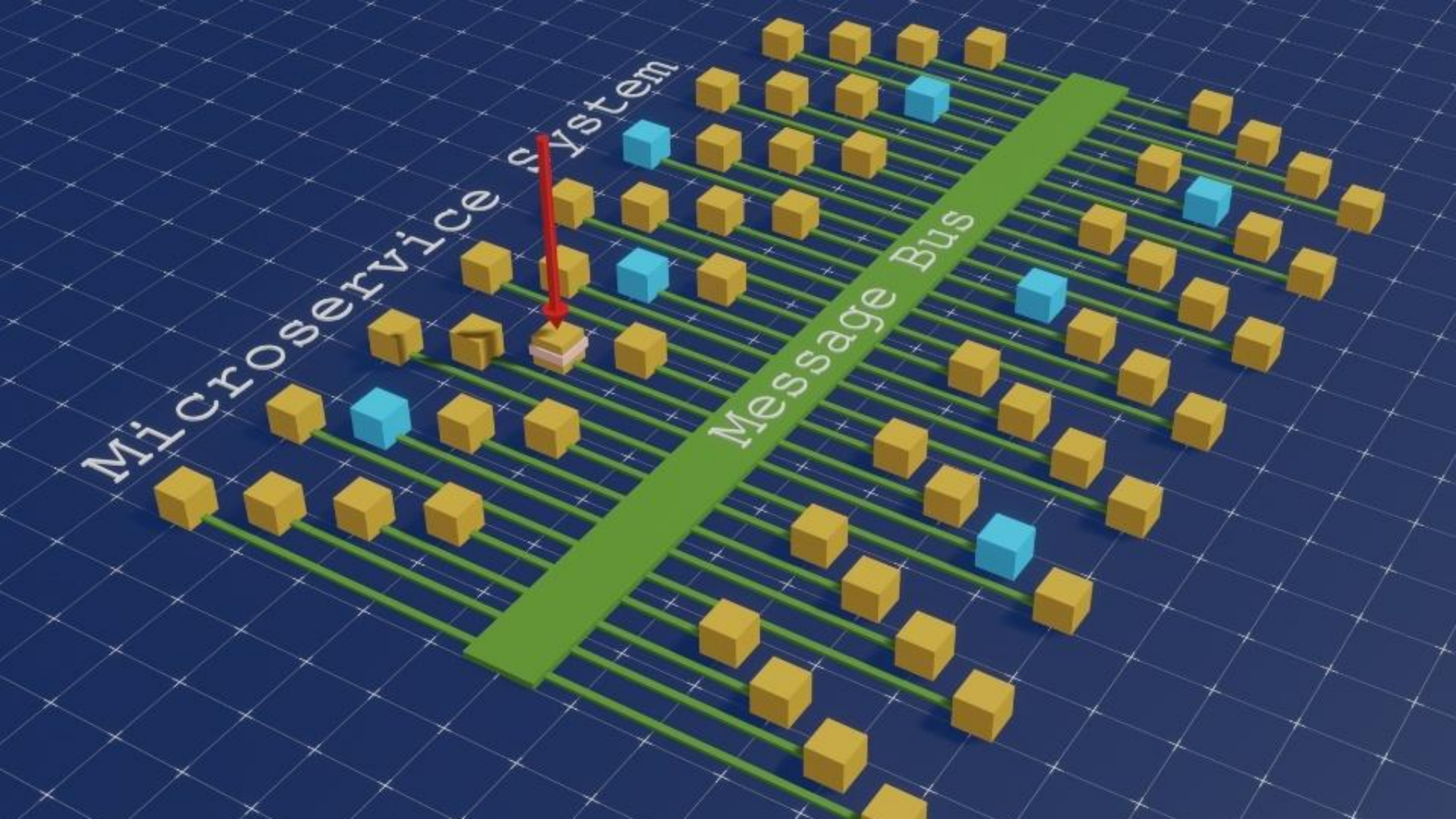

Microservice System

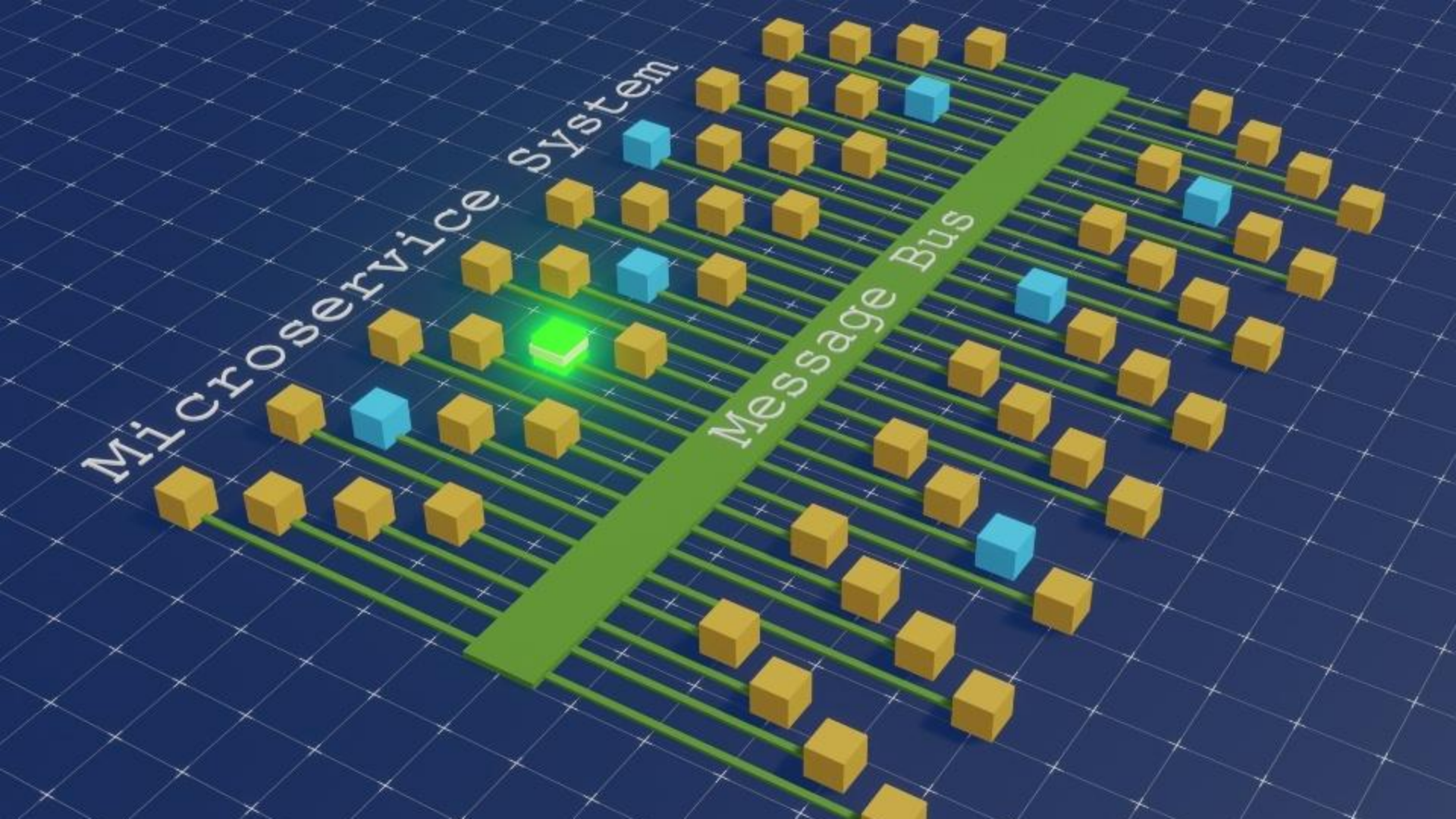

Microservice System
Message Bus

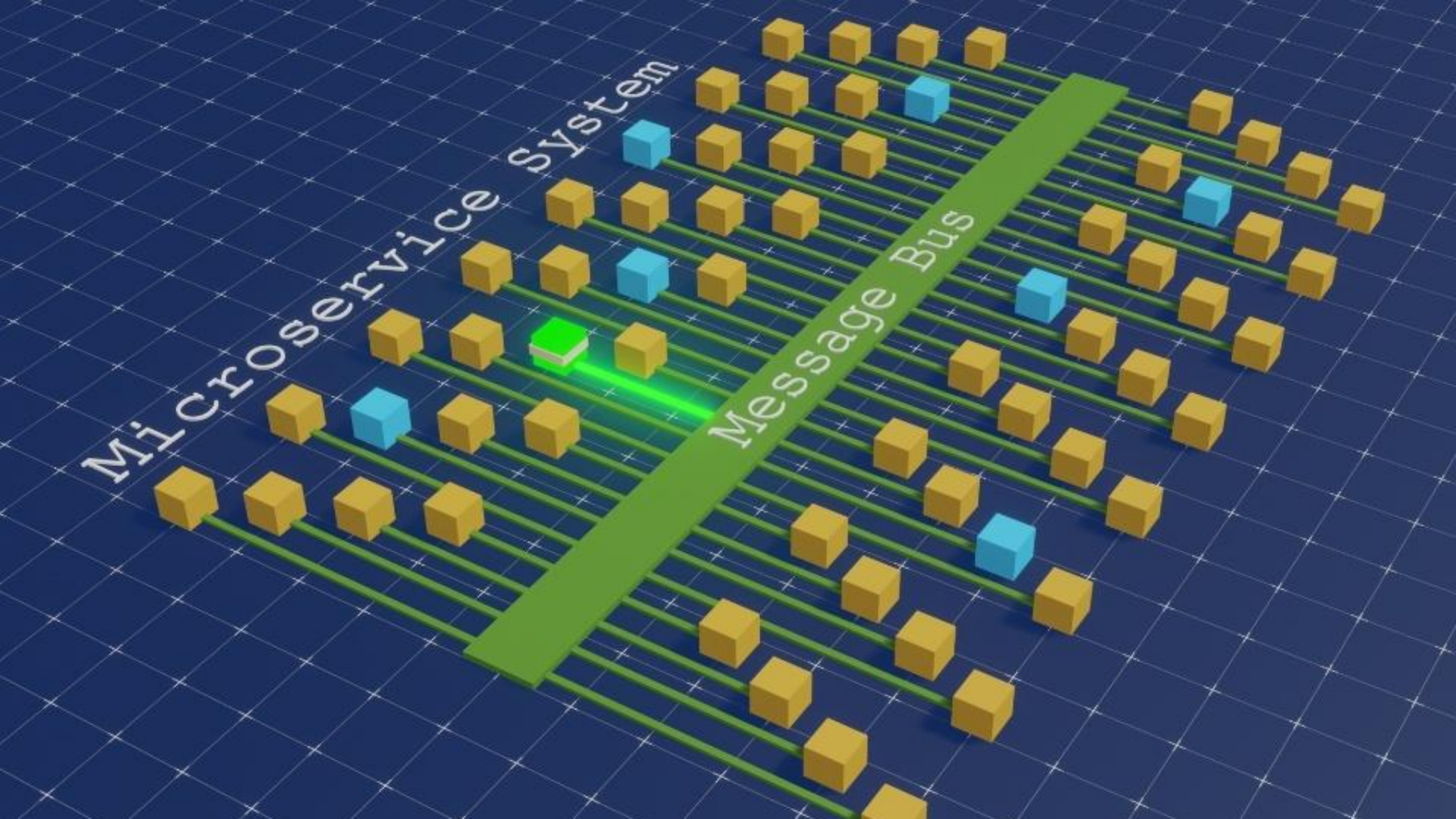

Microservice System
Message Bus

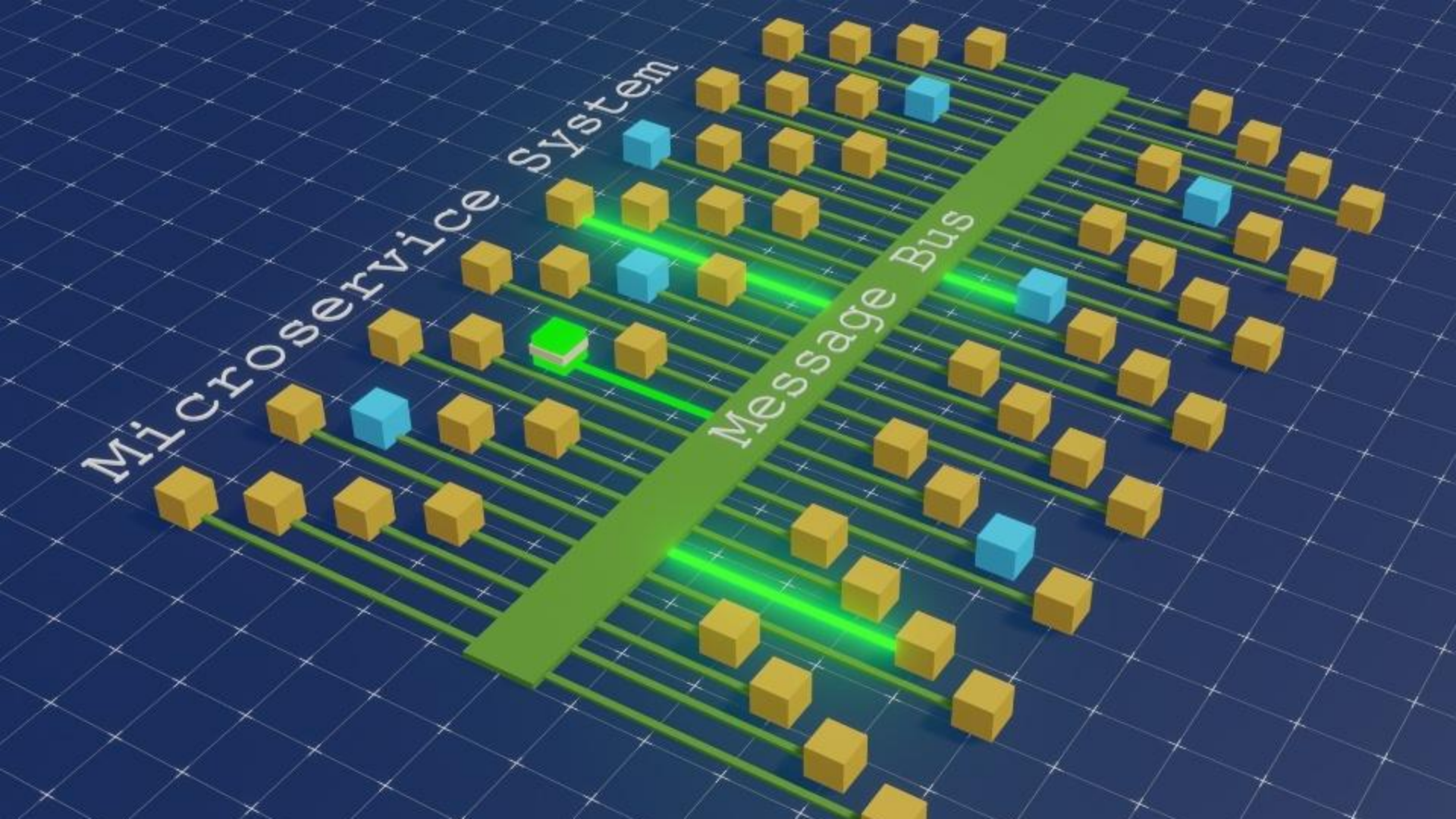

Microservice System
Message Bus
Analytics Service

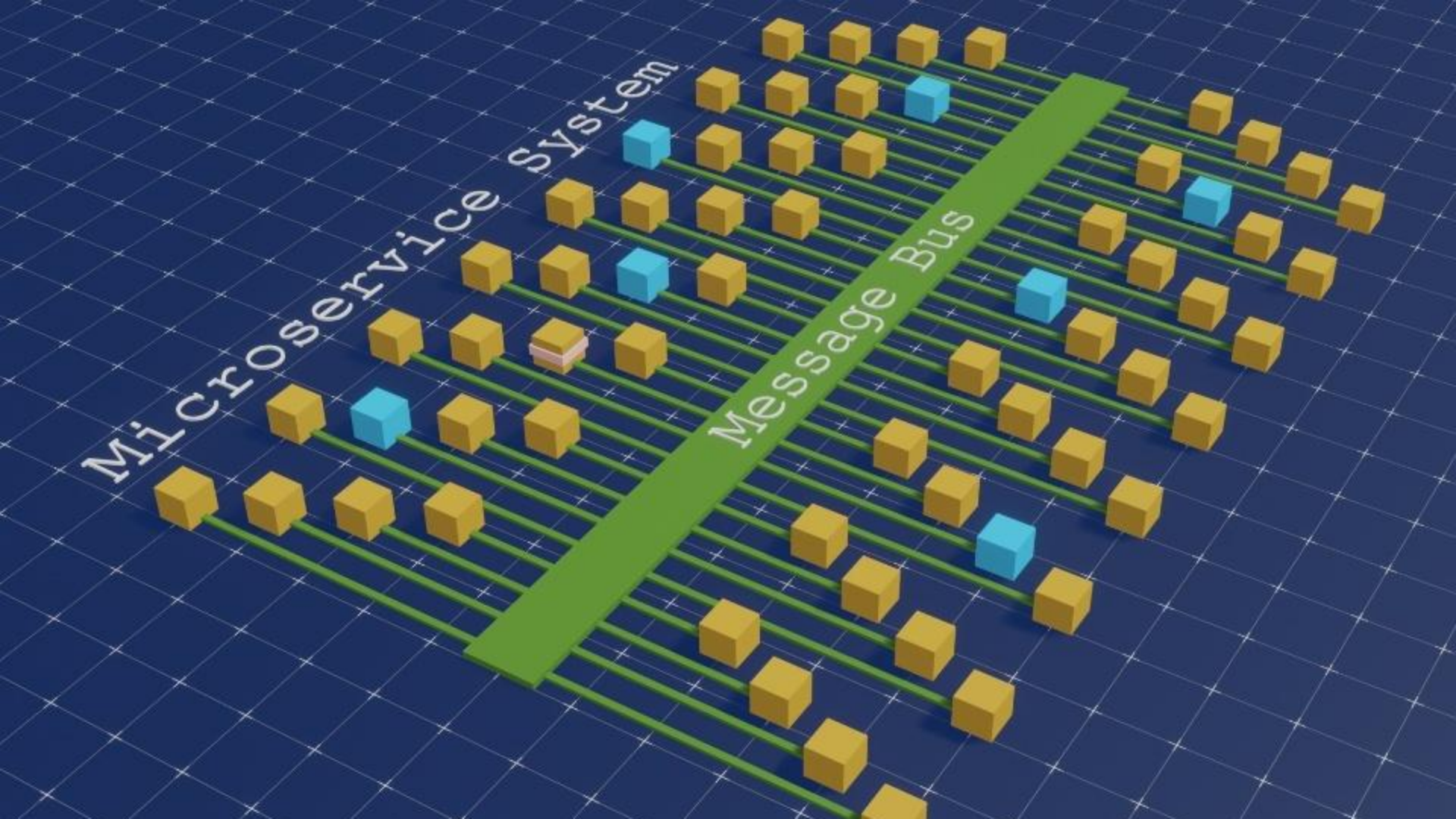

Microservice System
Message Bus

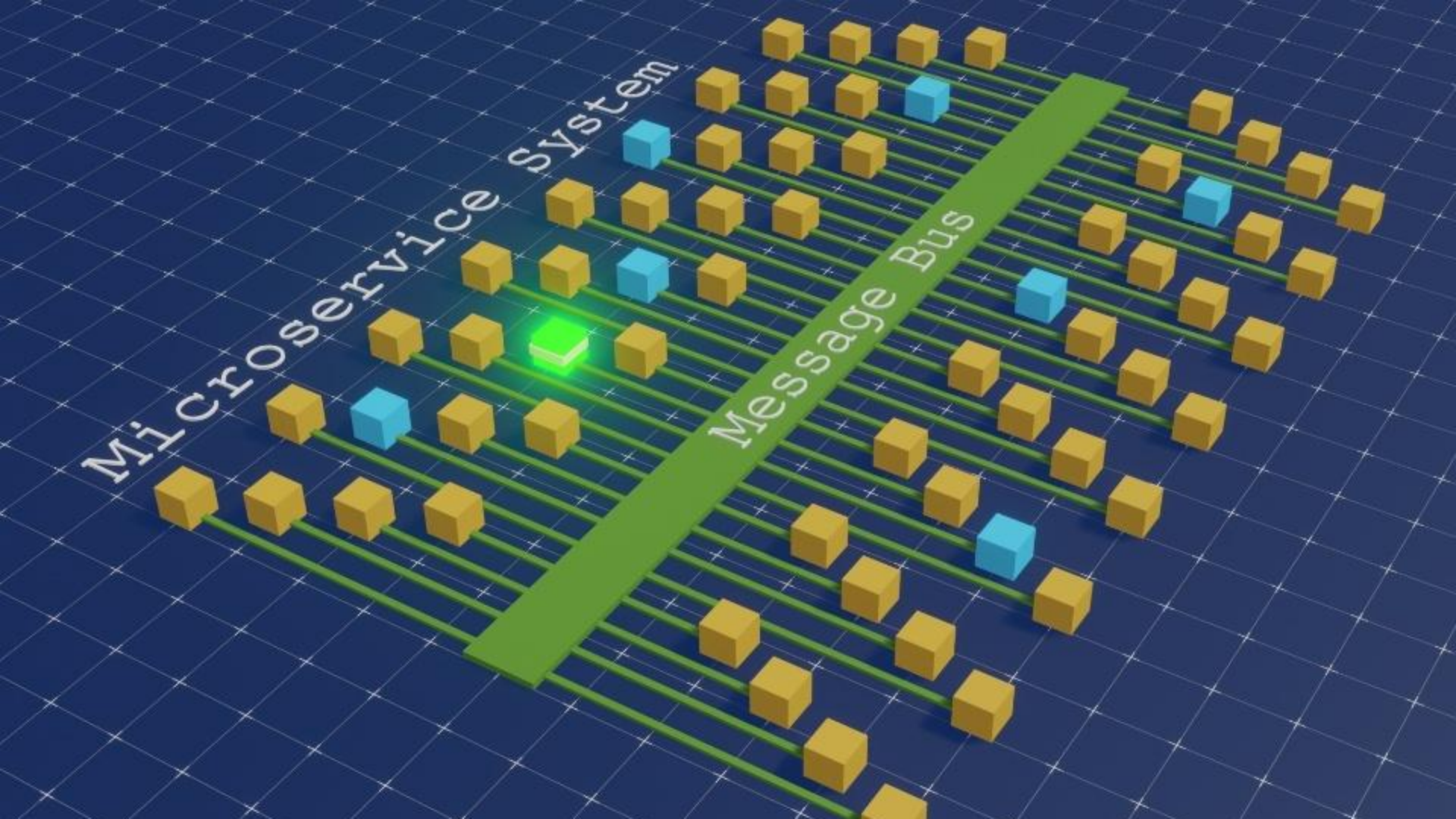

Microservice system
Message Bus

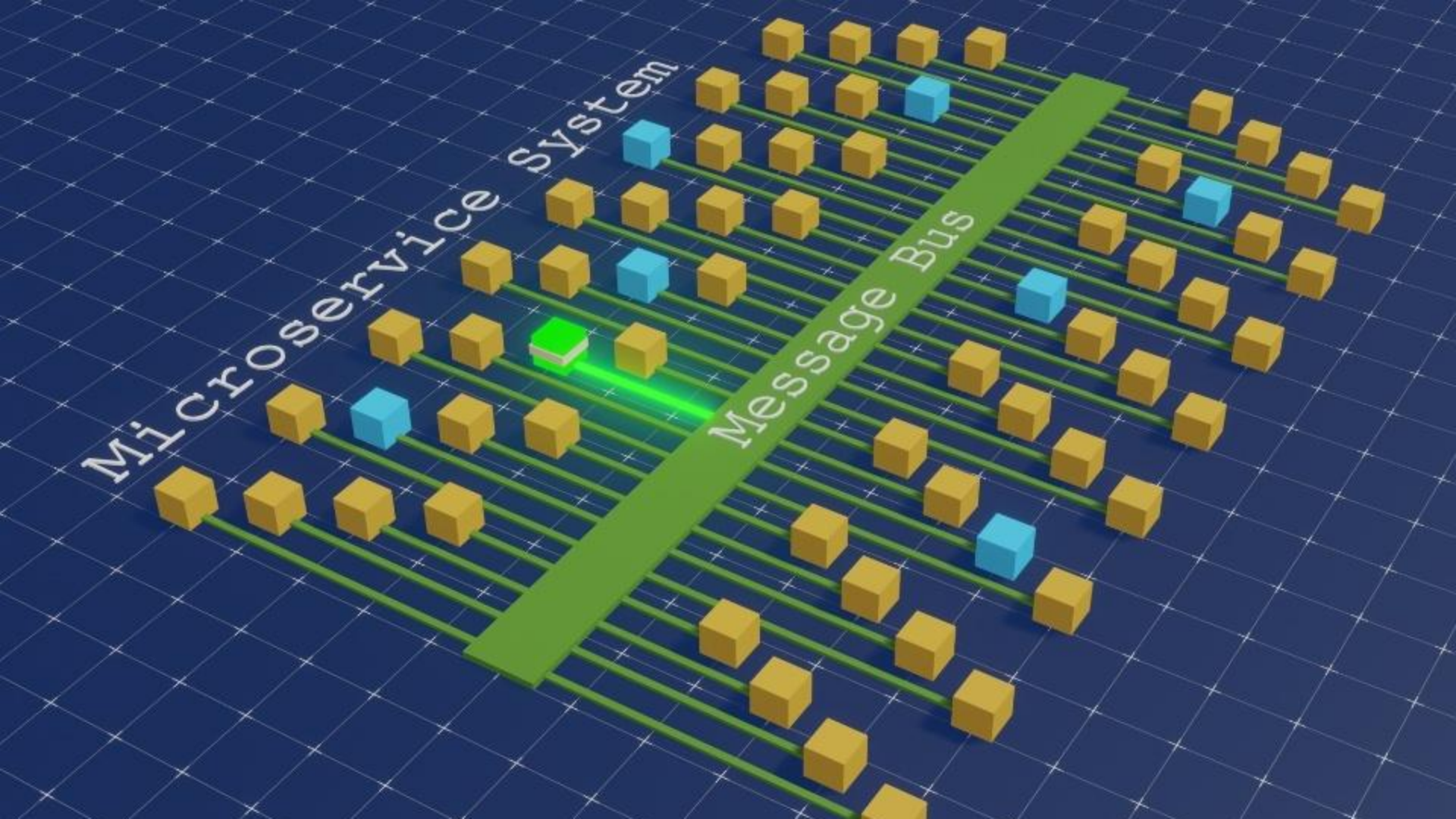

Microservice System
Message Bus

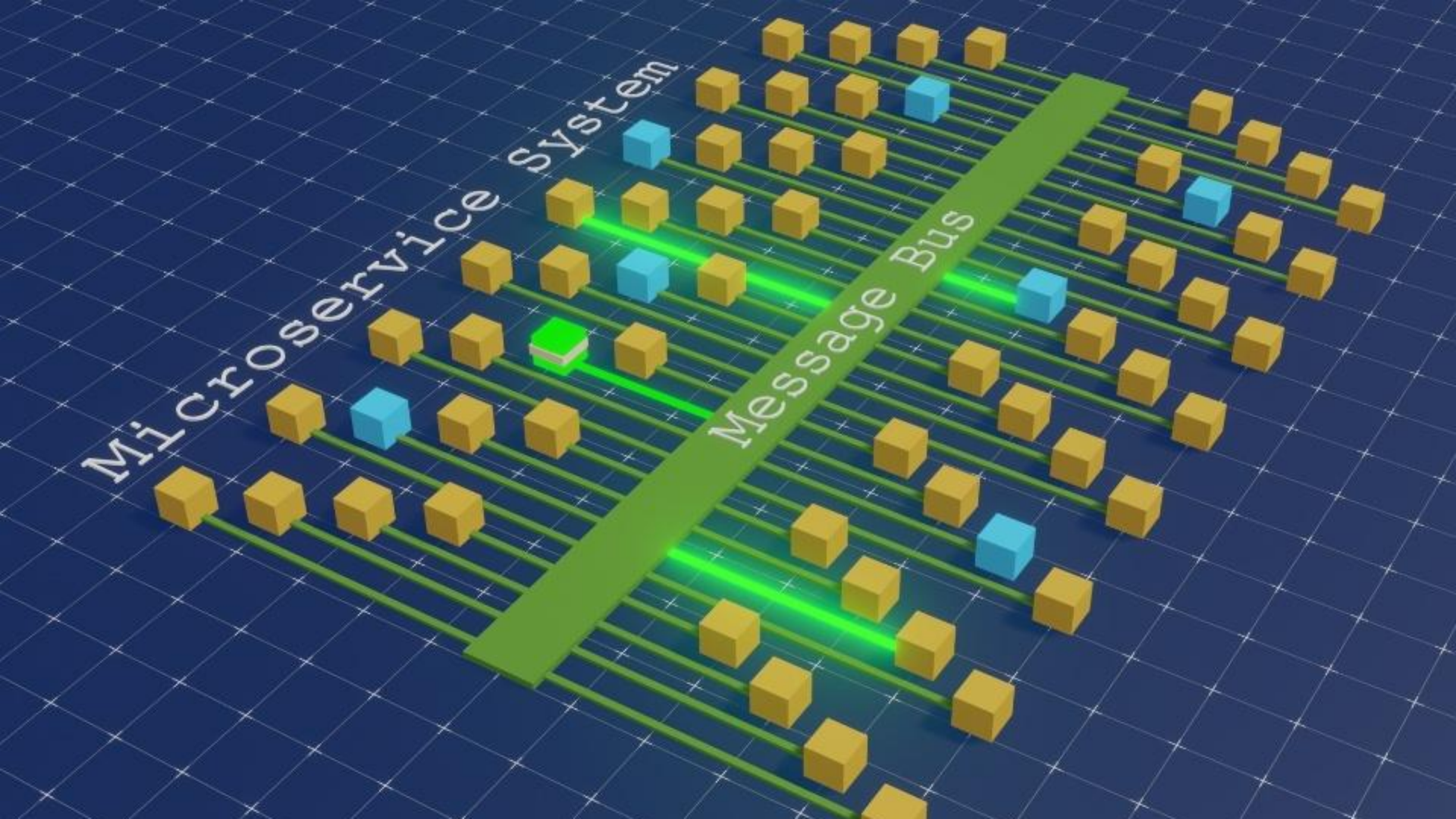

Microservice System
Message Bus

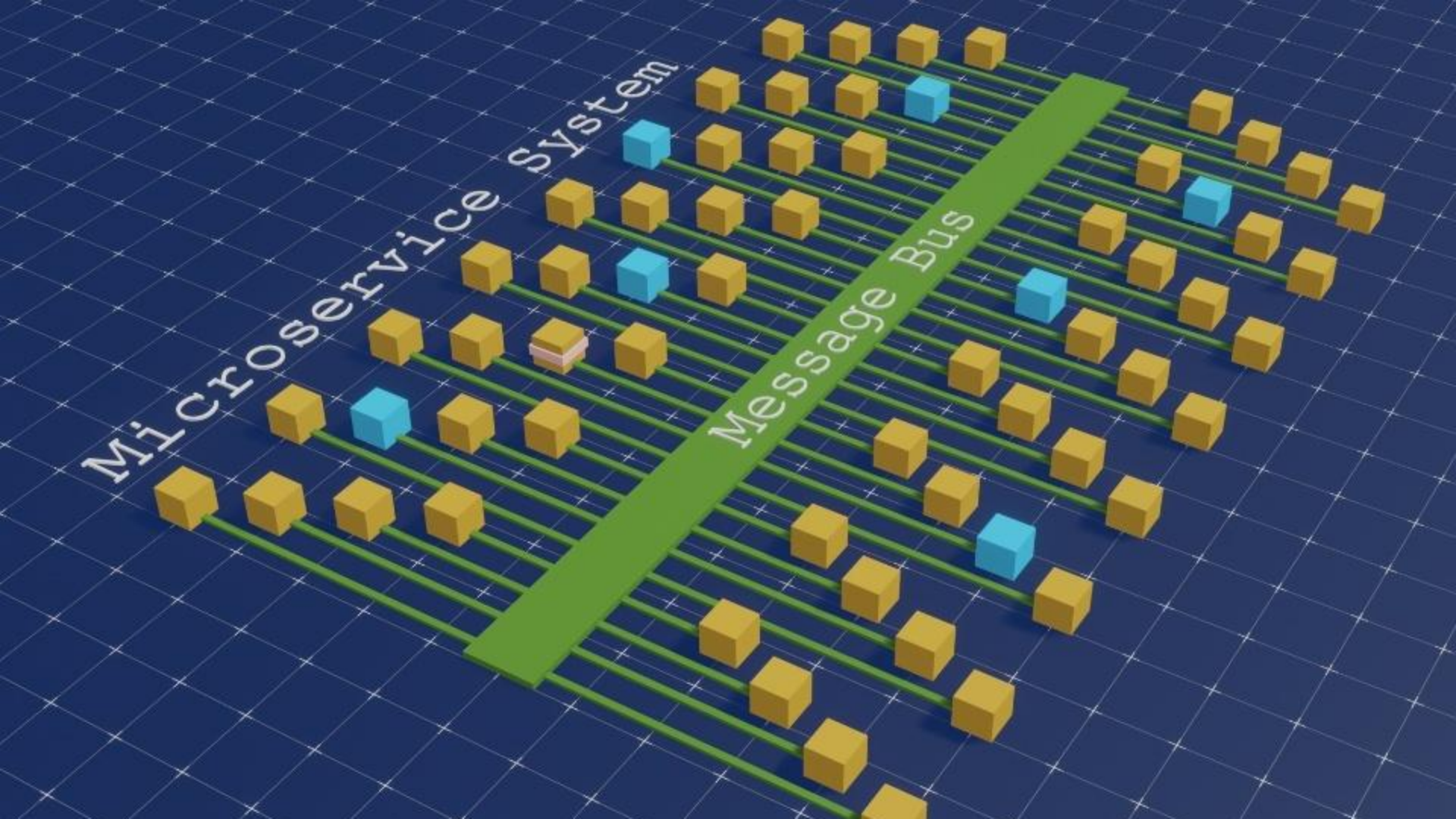

Microservice System
Message Bus

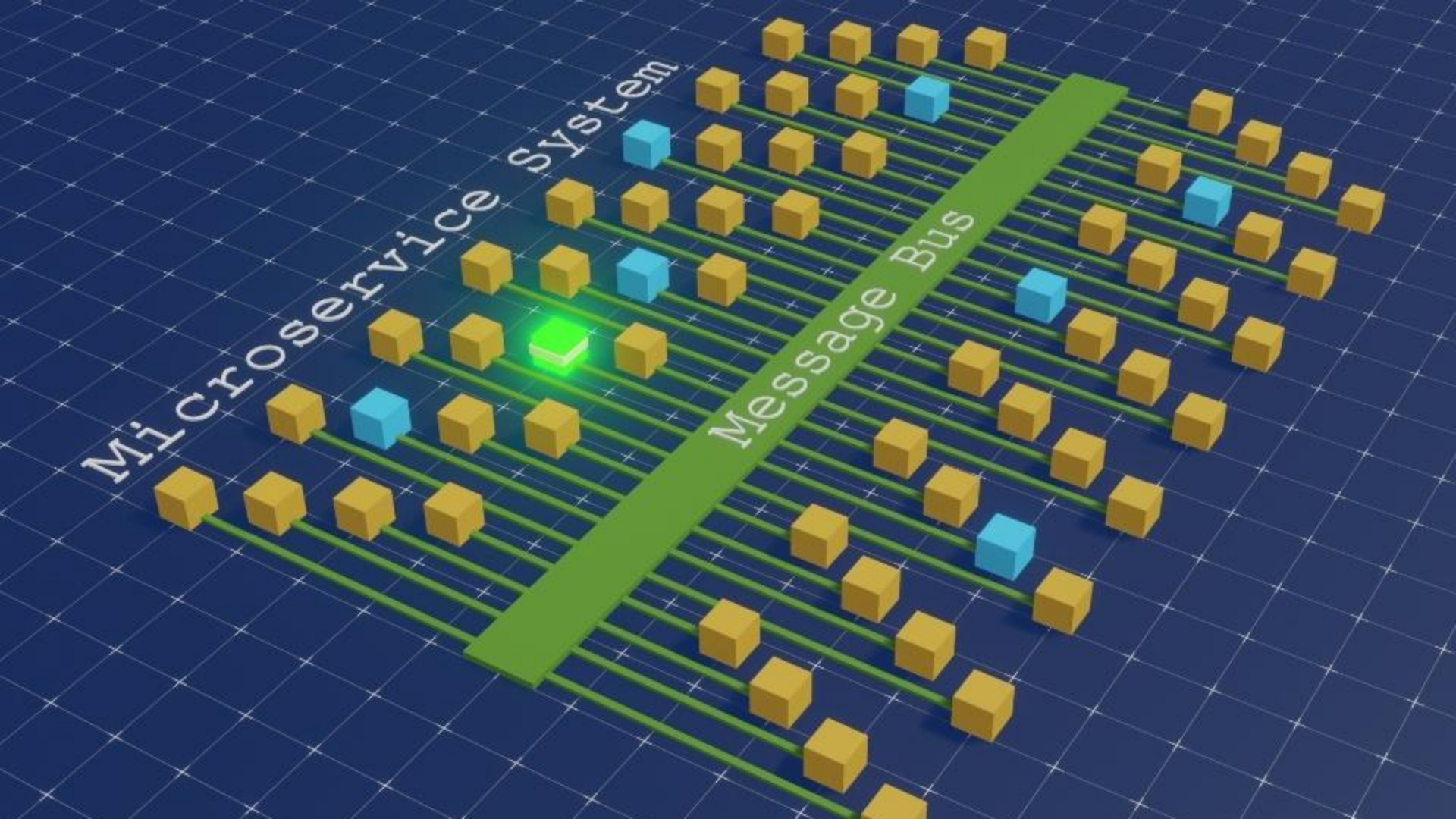

Microservice System
Message Bus

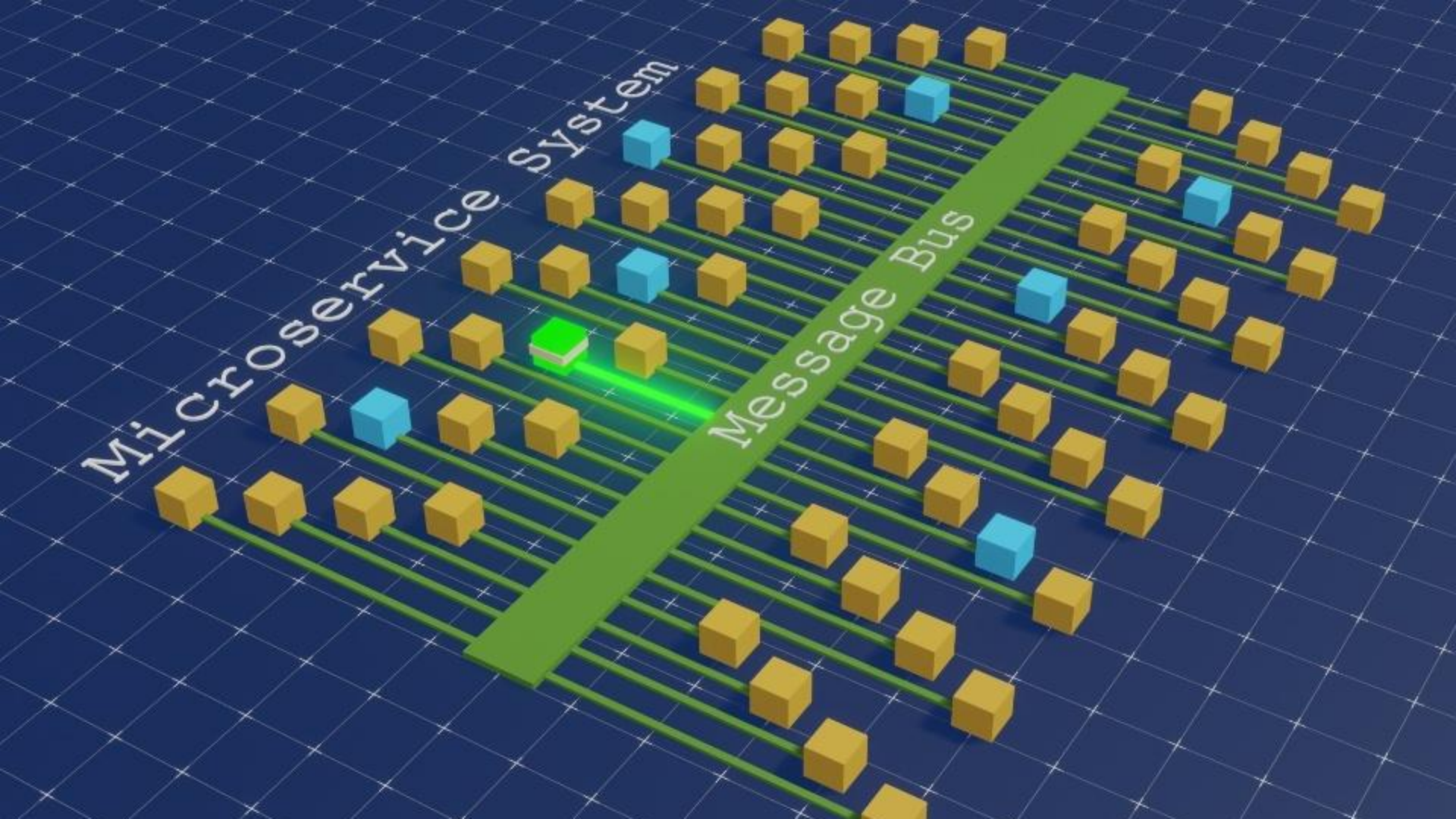

Microservice System
Message Bus

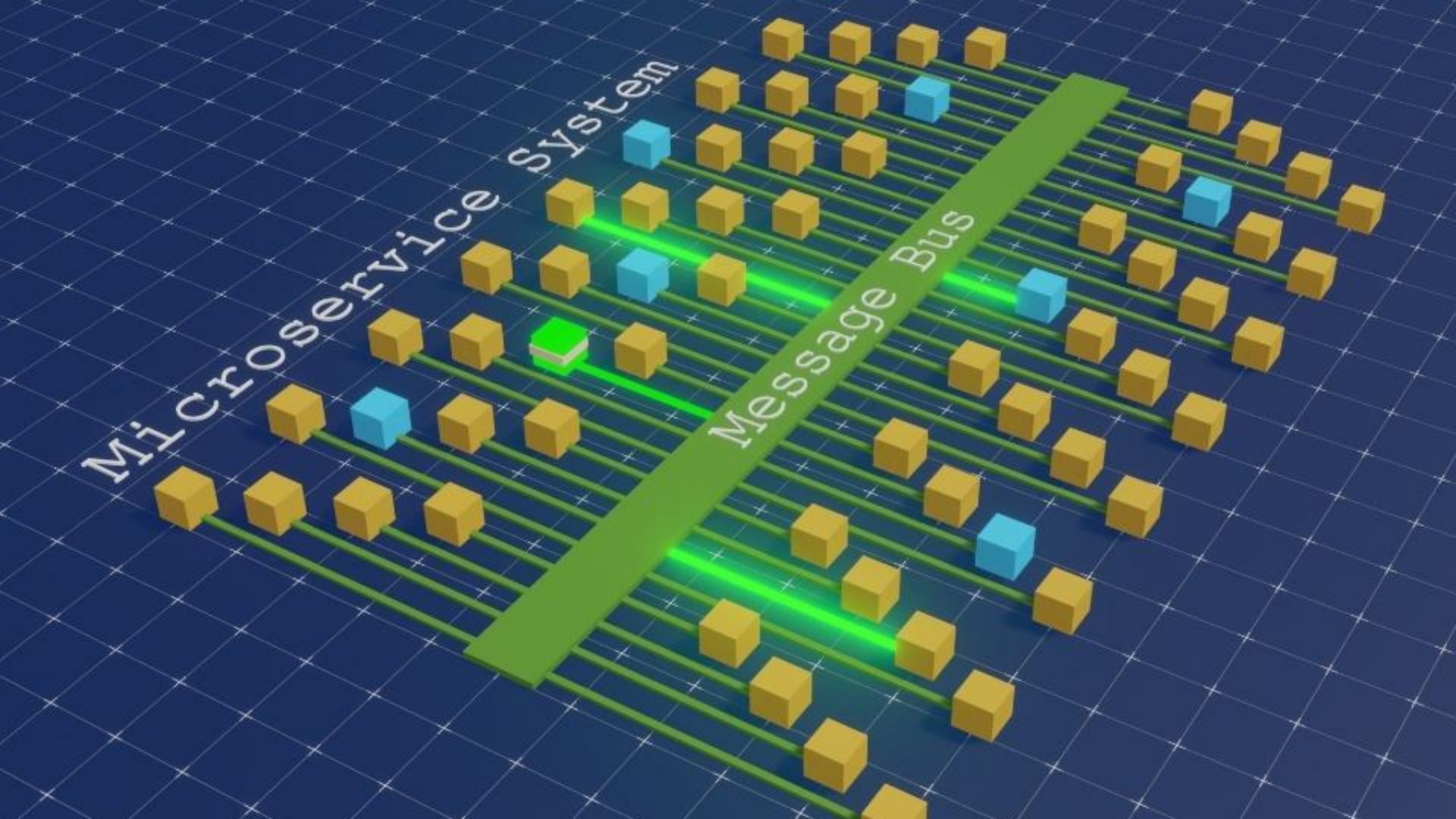

Microservice System
Message Bus

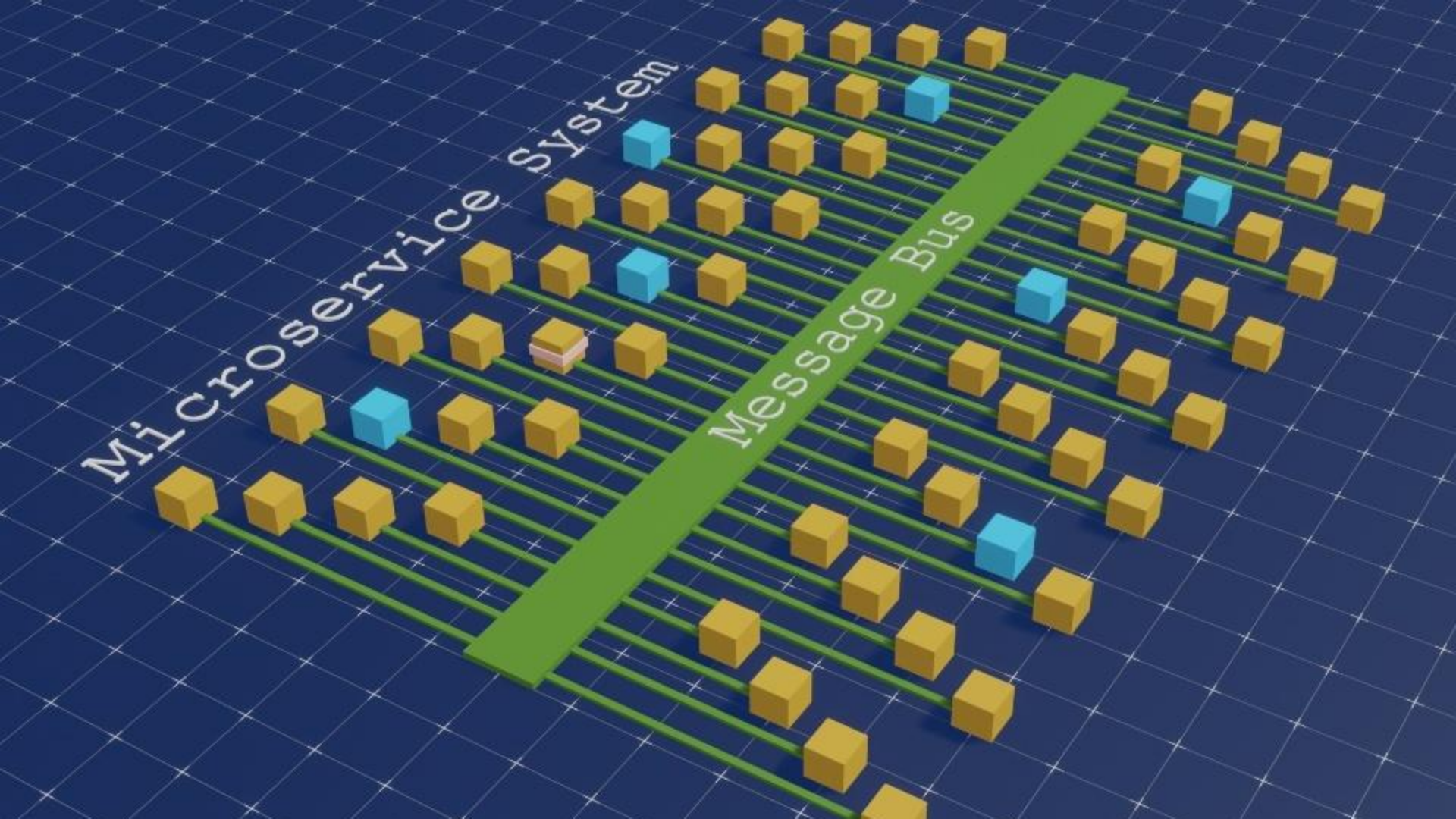

Microservice System
Message Bus

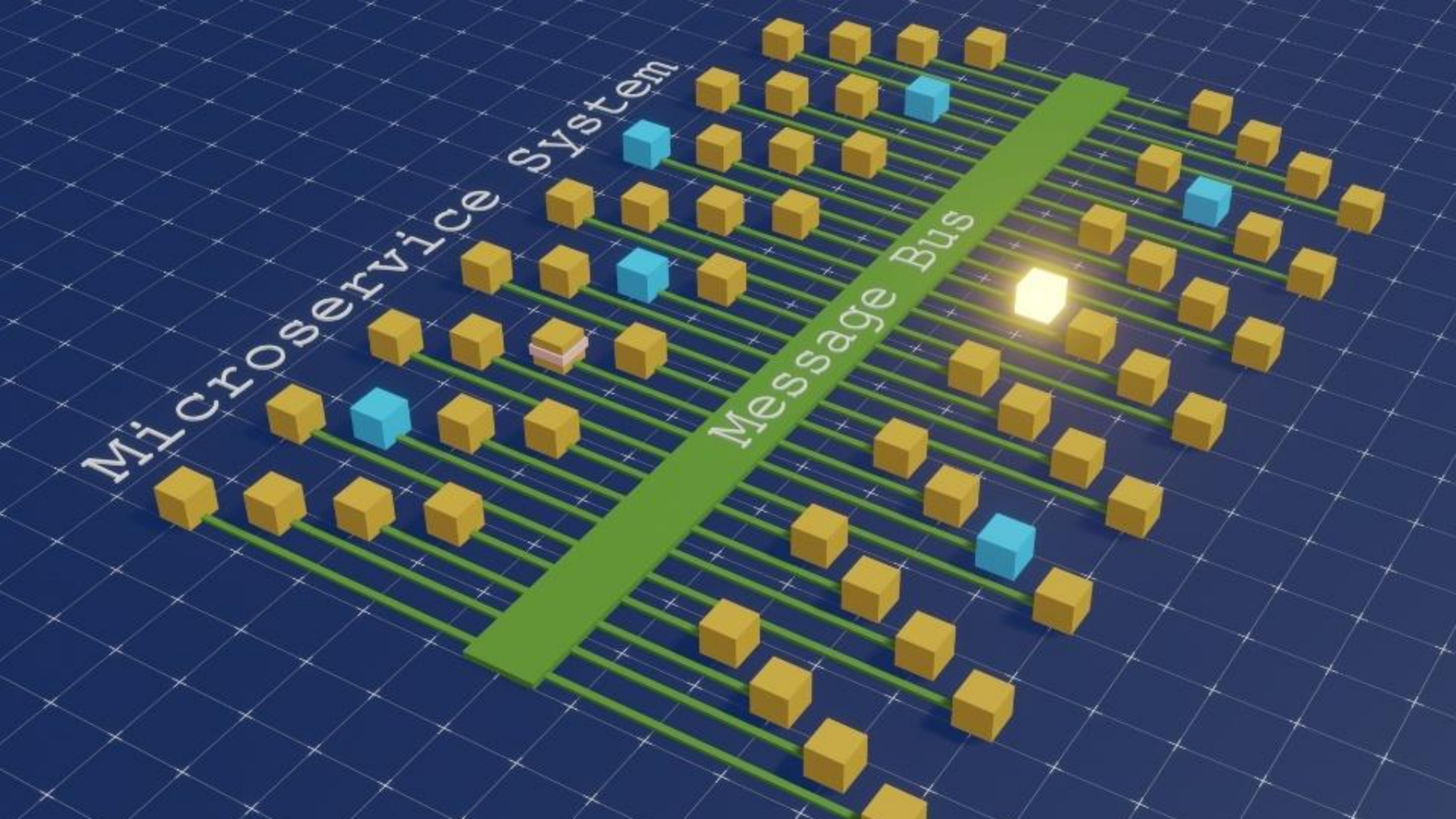

Microservice System
Message Bus

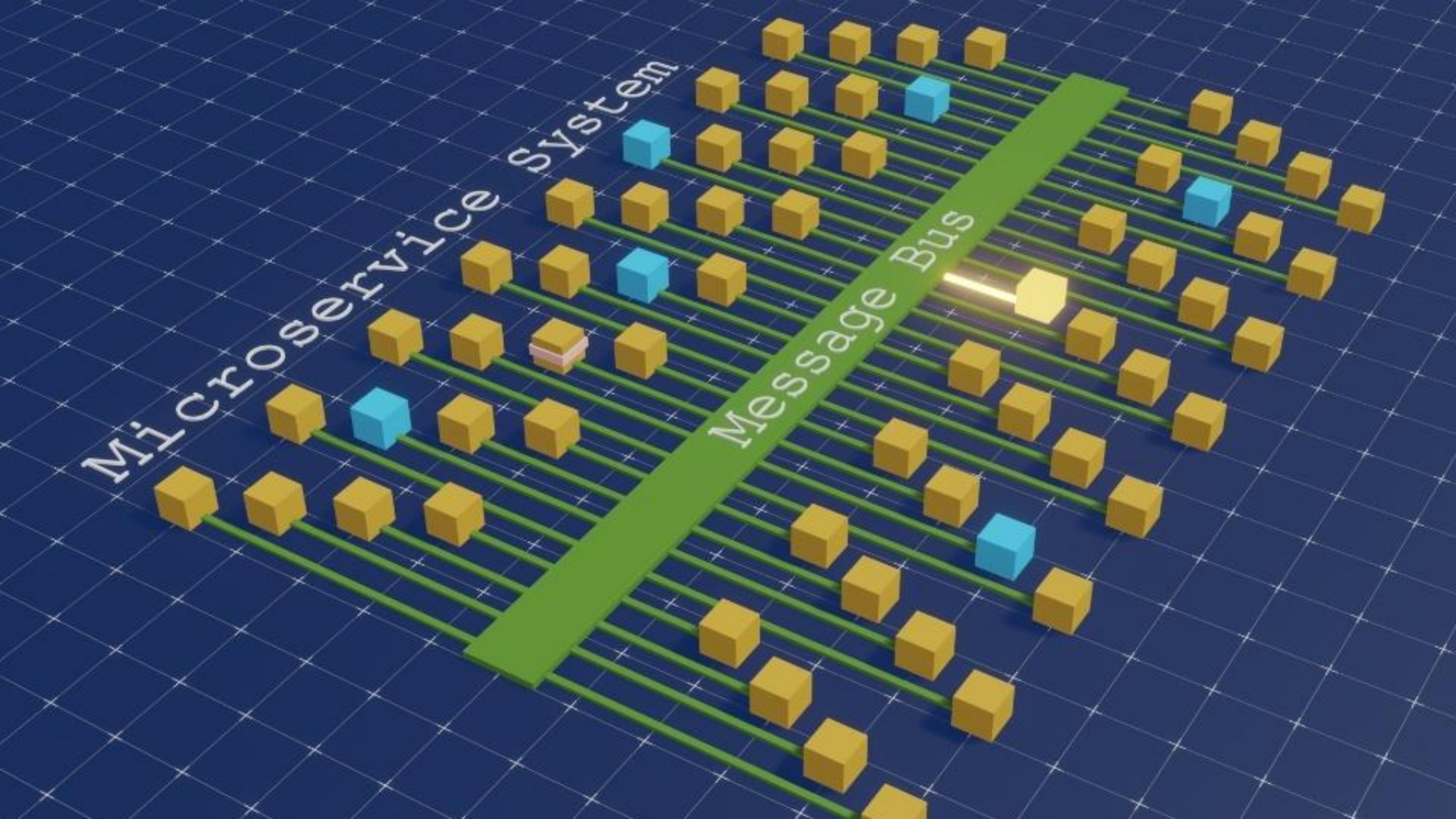

Microservice System
Message Bus

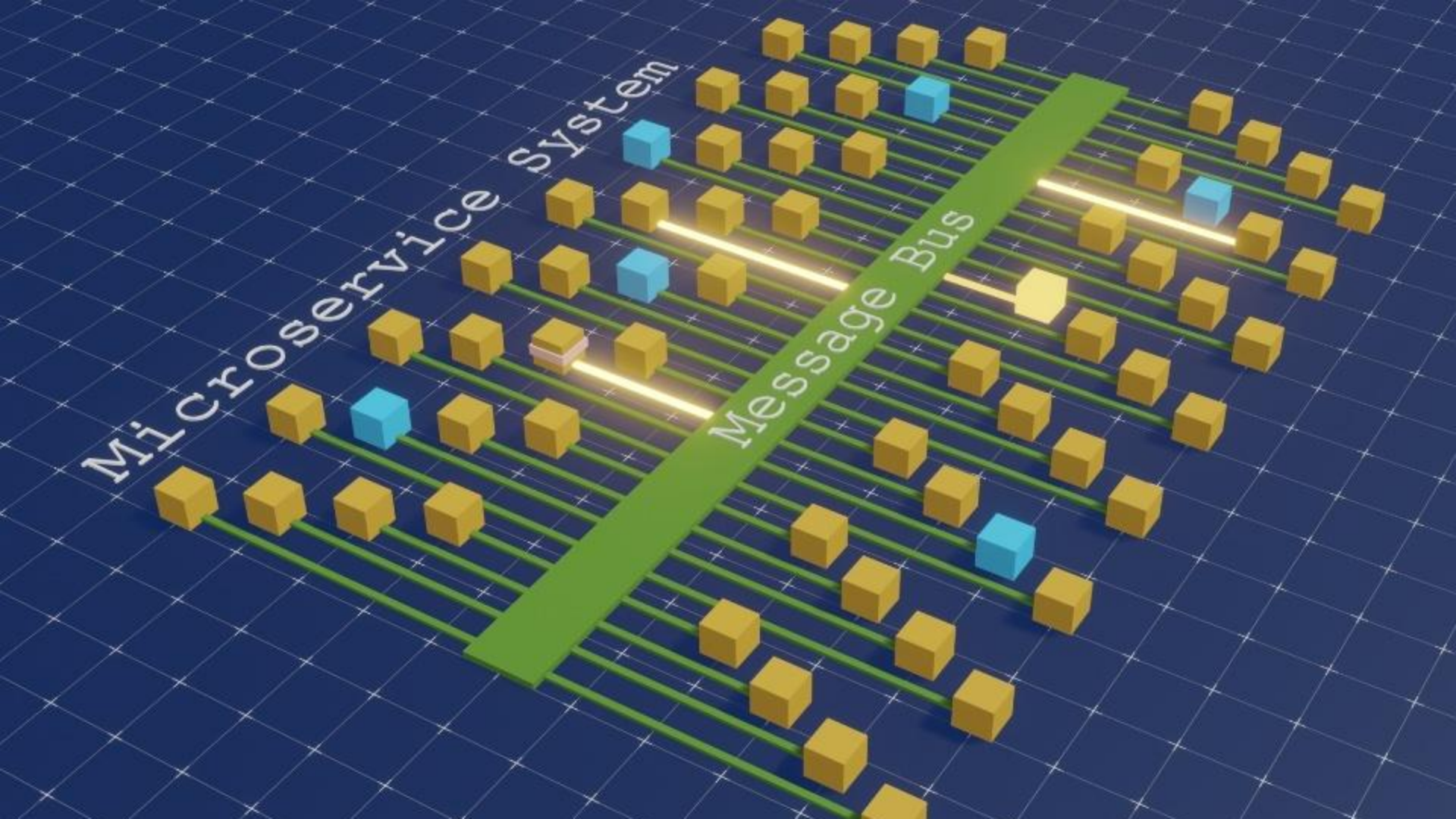

Microservice System
Message Bus

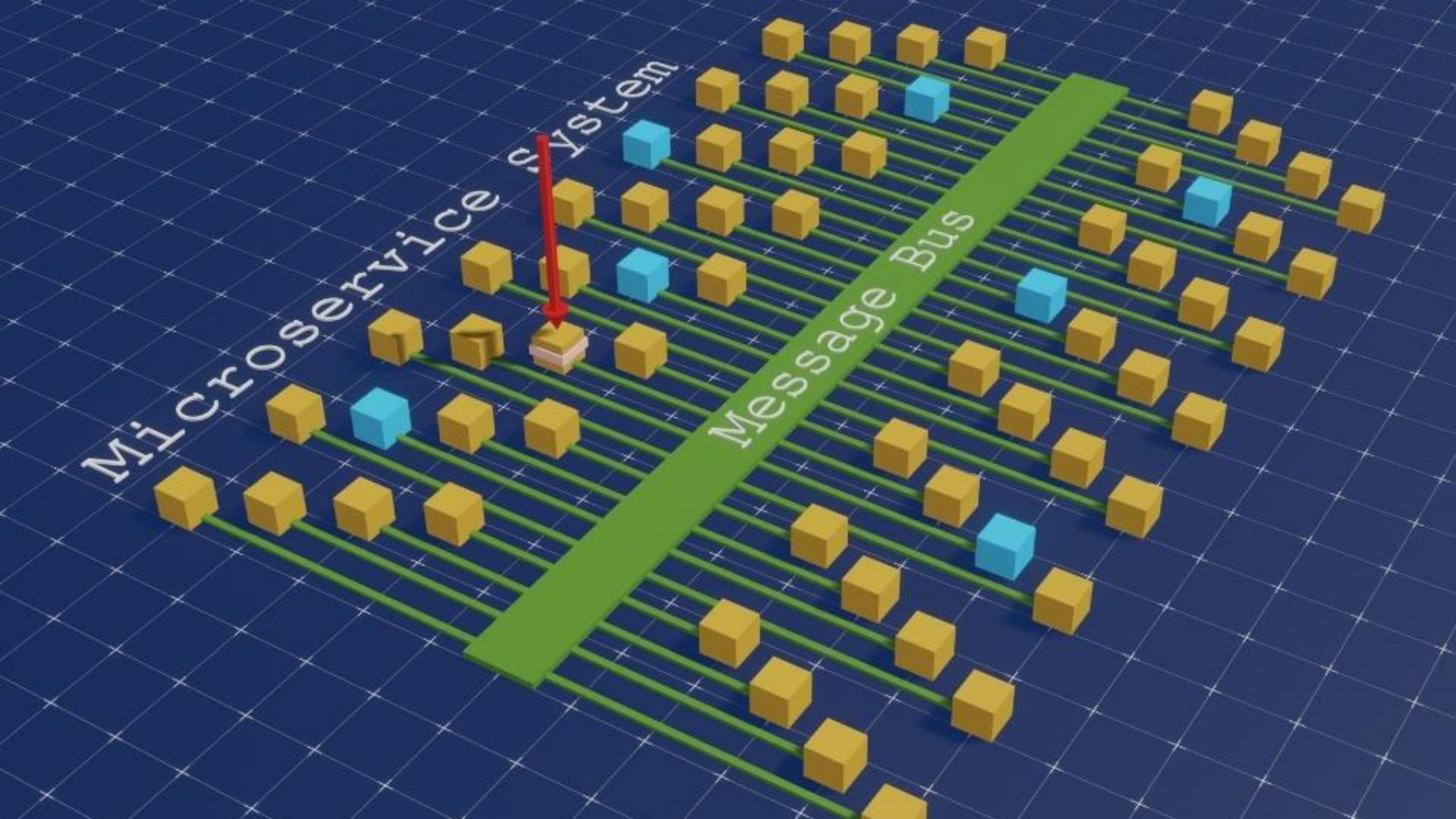

Microservice System
Message Bus

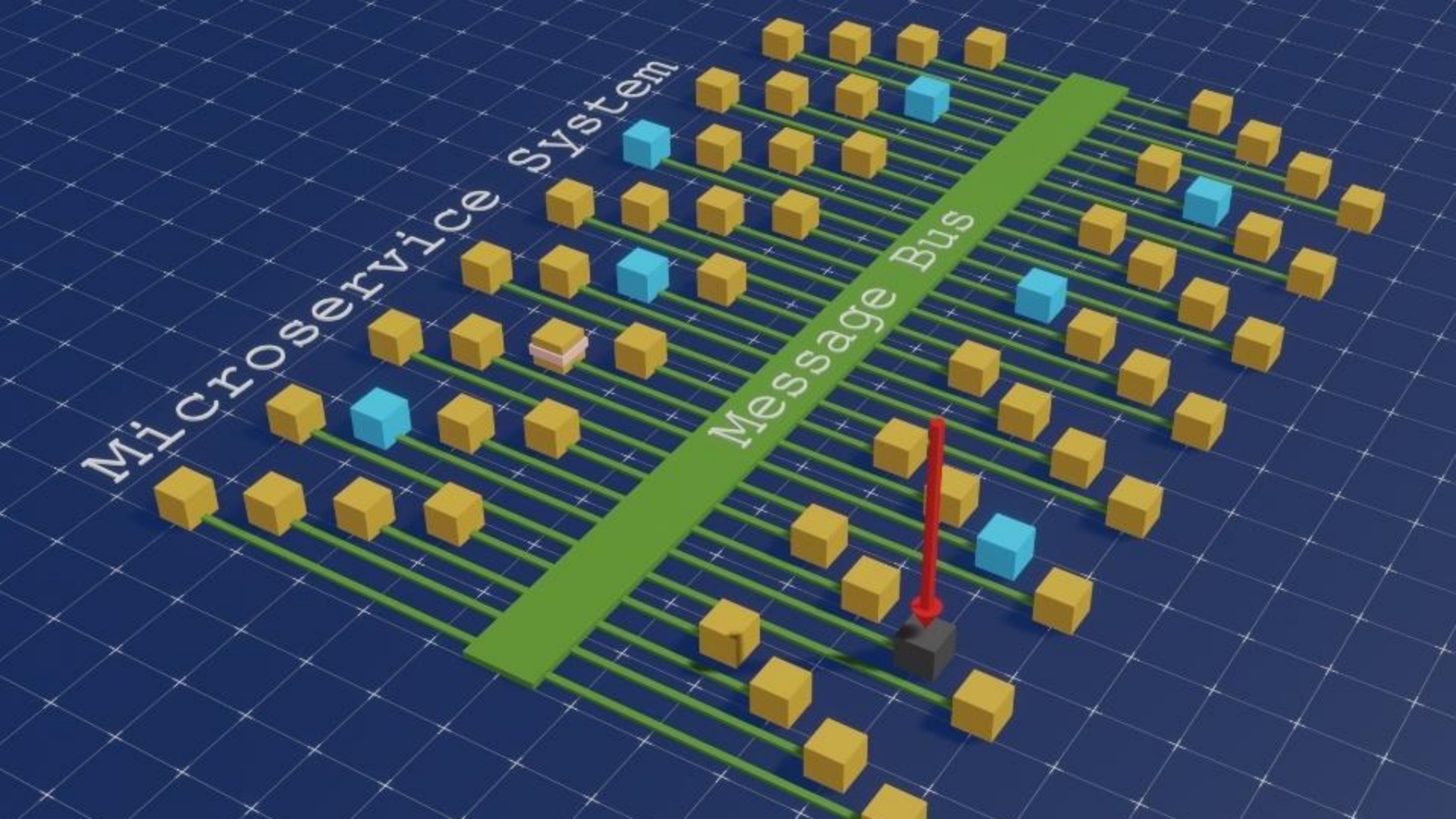

Microservice System
Message Bus

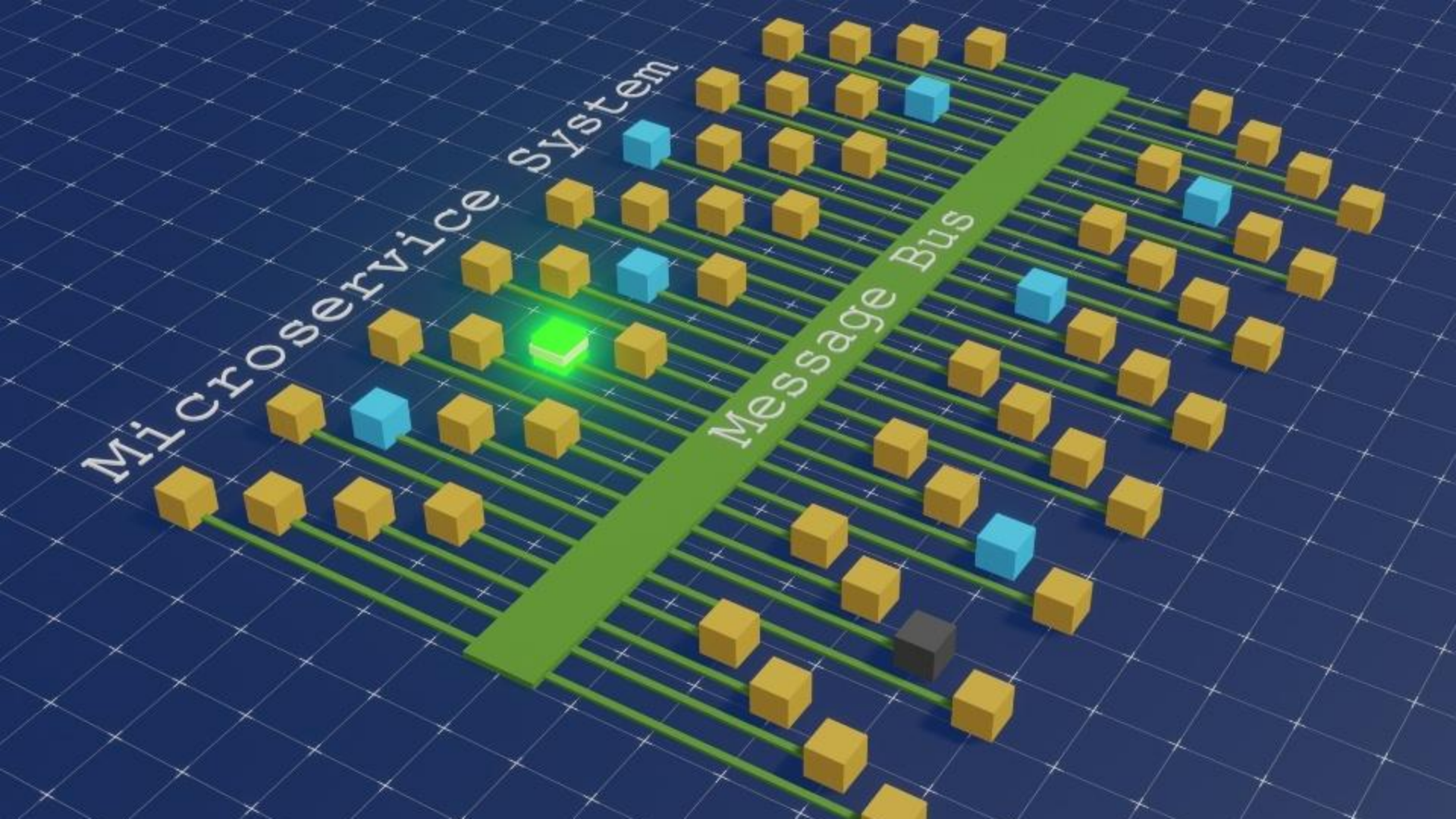

Microservice System
Message Bus

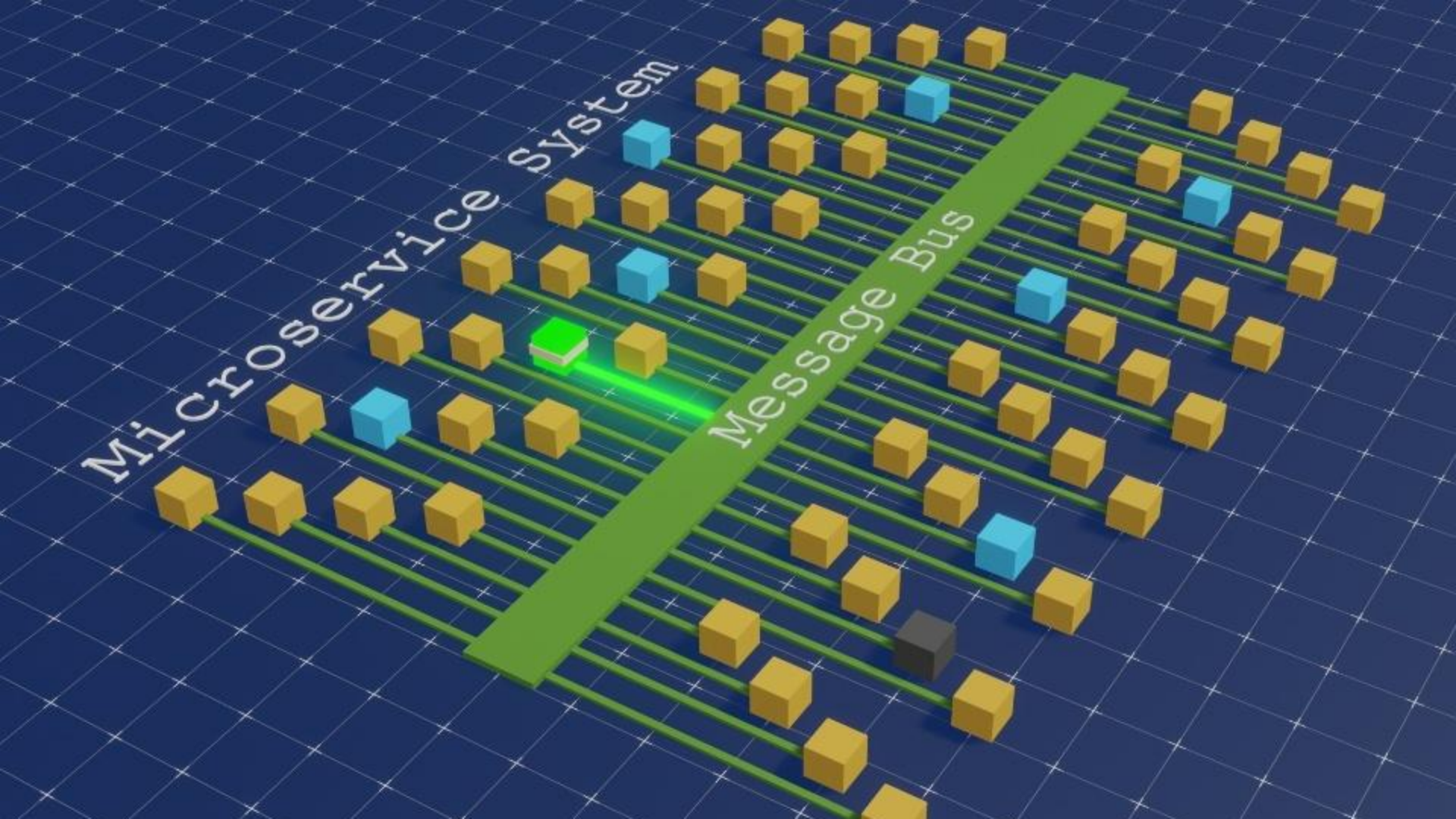

Microservice System
Message Bus

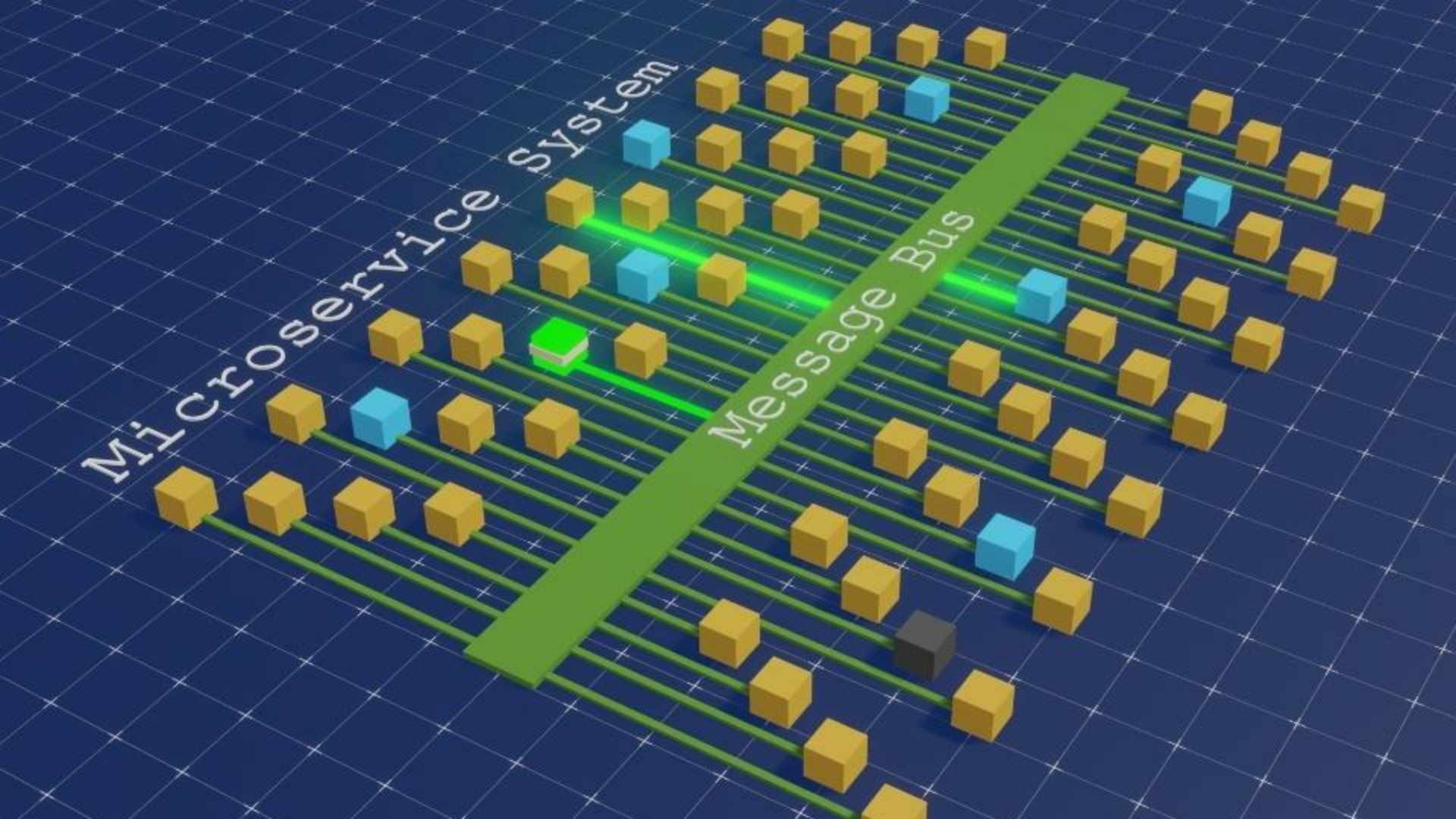

Microservice System
Message Bus

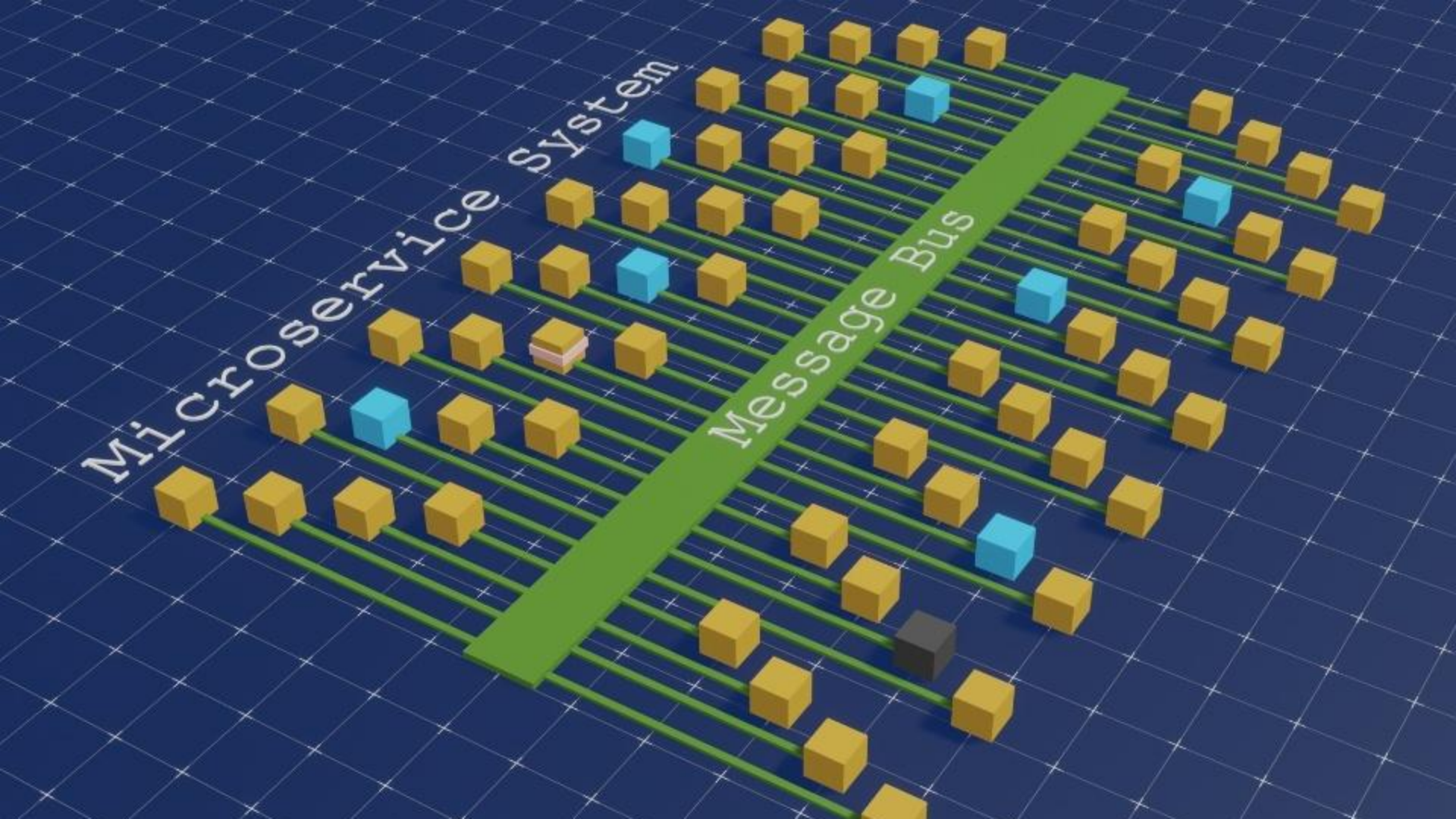

Microservice system
Message Bus

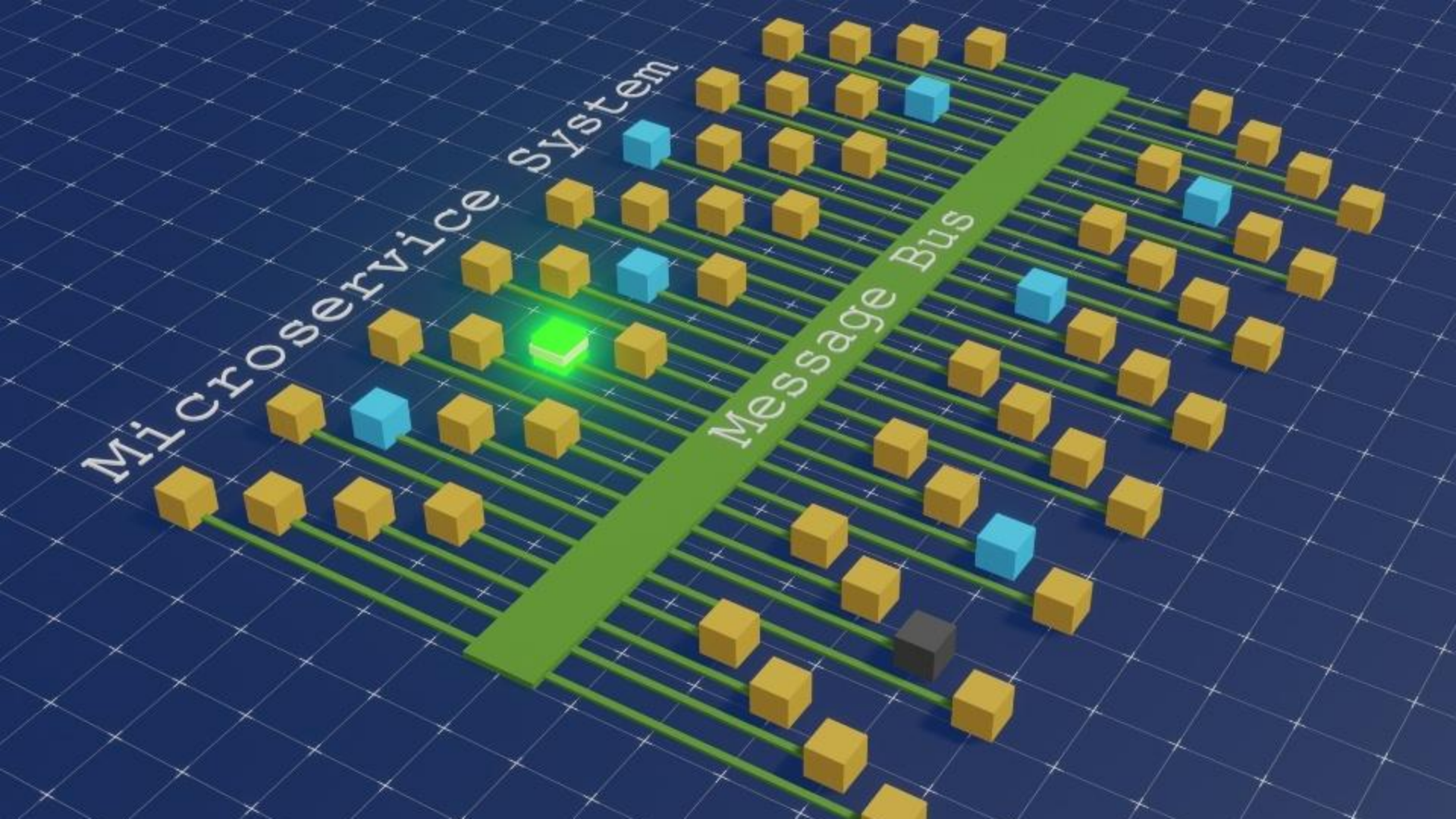

Microservice System
Message Bus

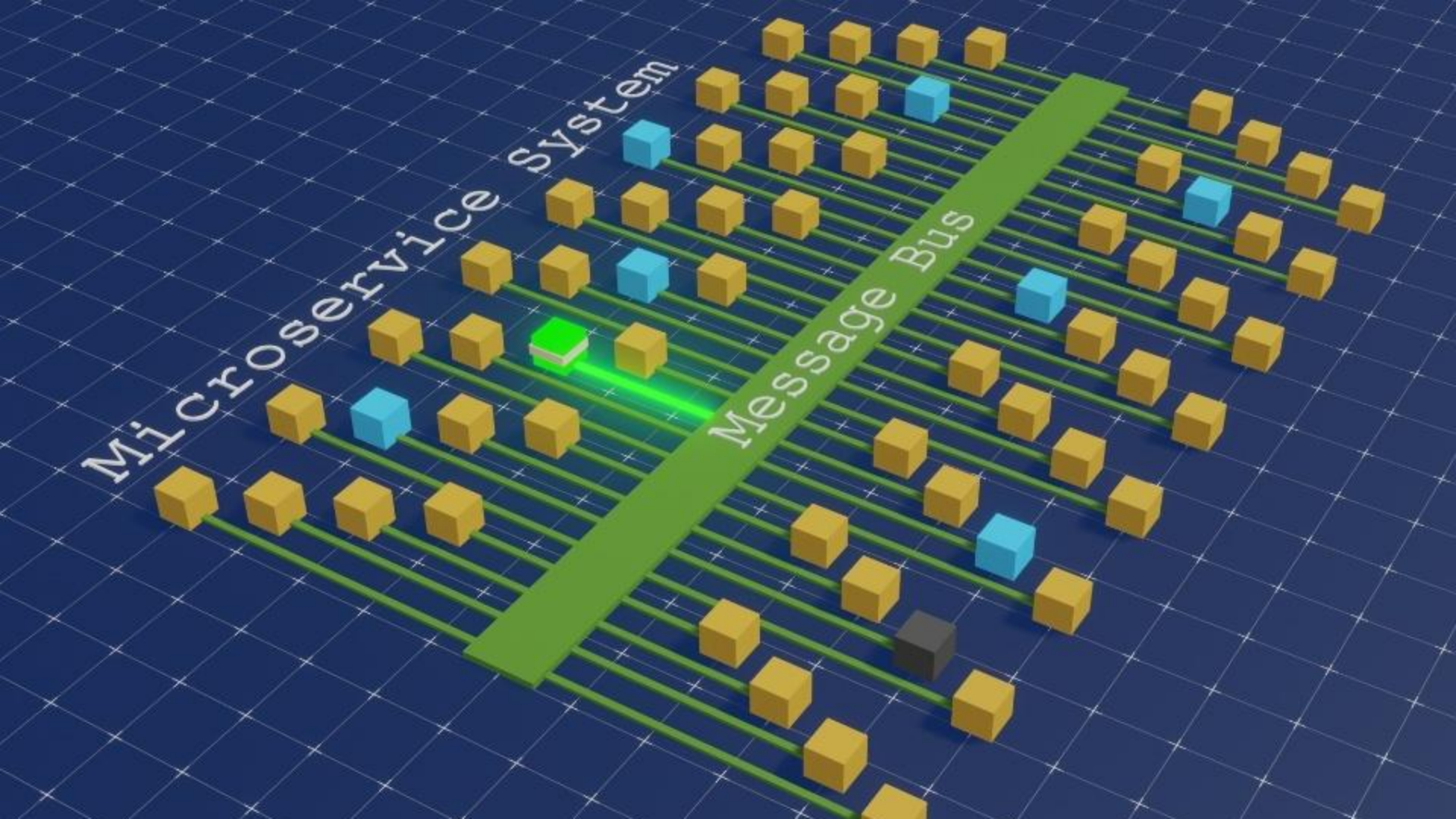

Microservice System
Message Bus

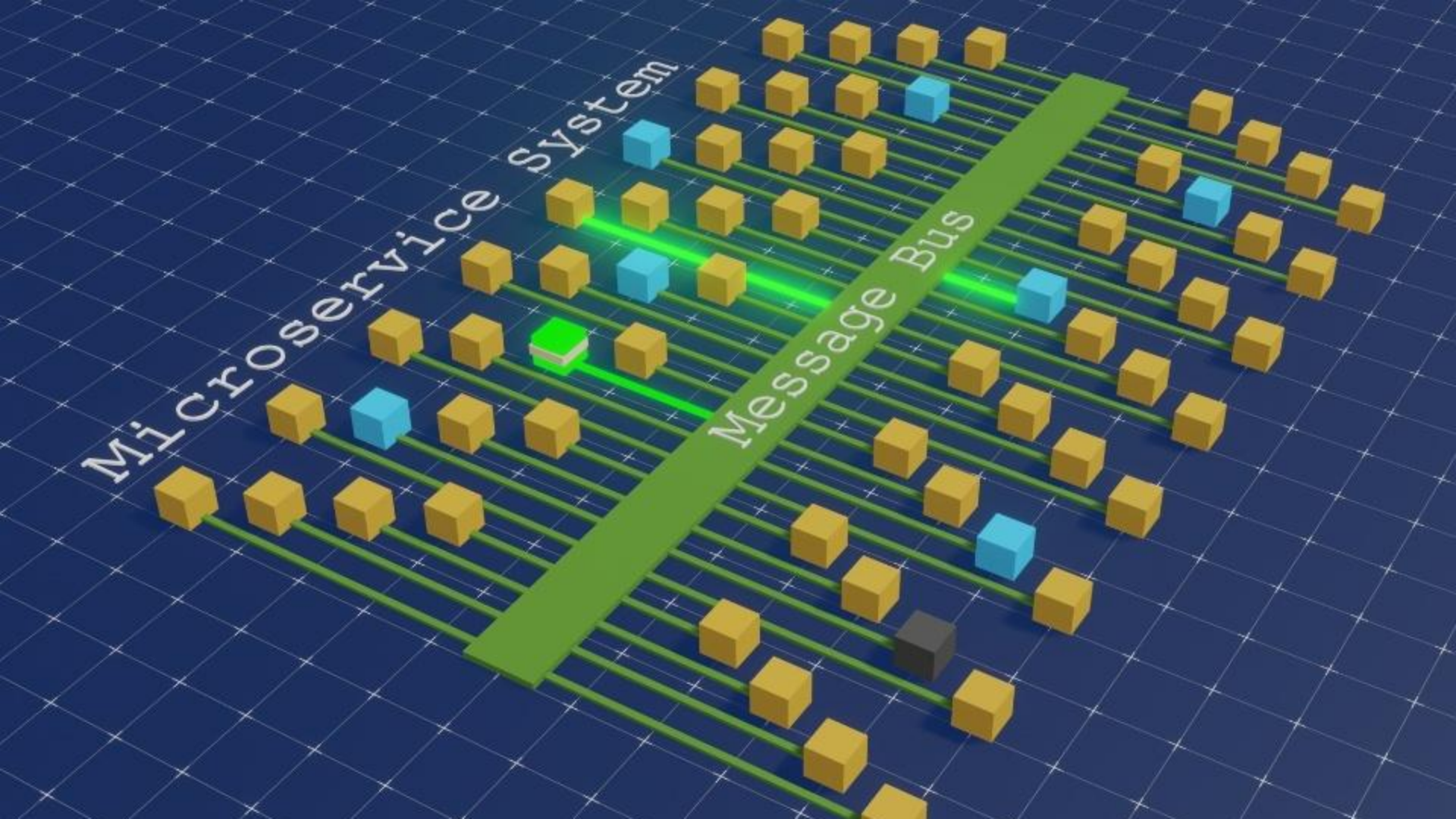

Microservice System
Message Bus

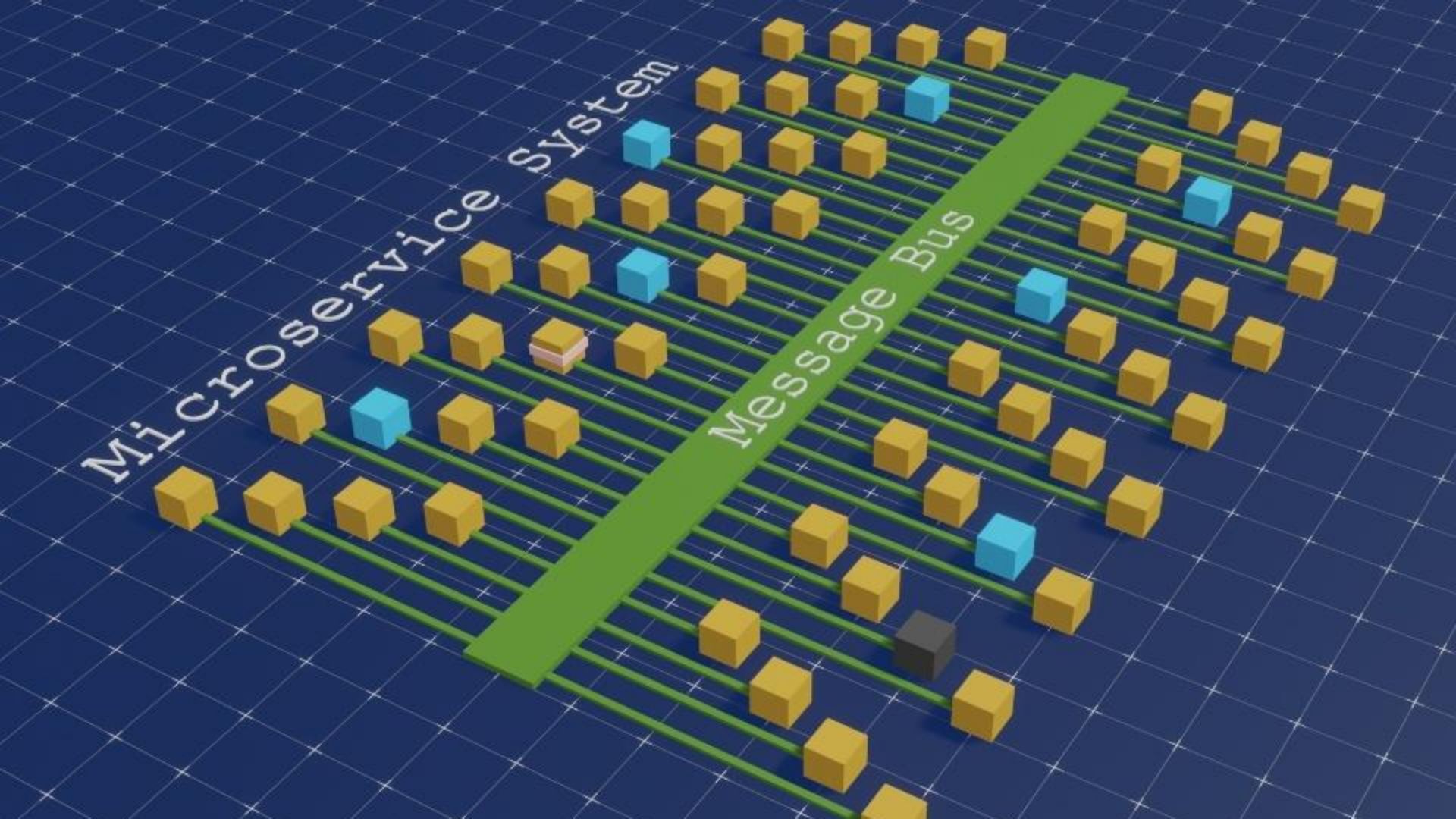

Microservice System
Message Bus

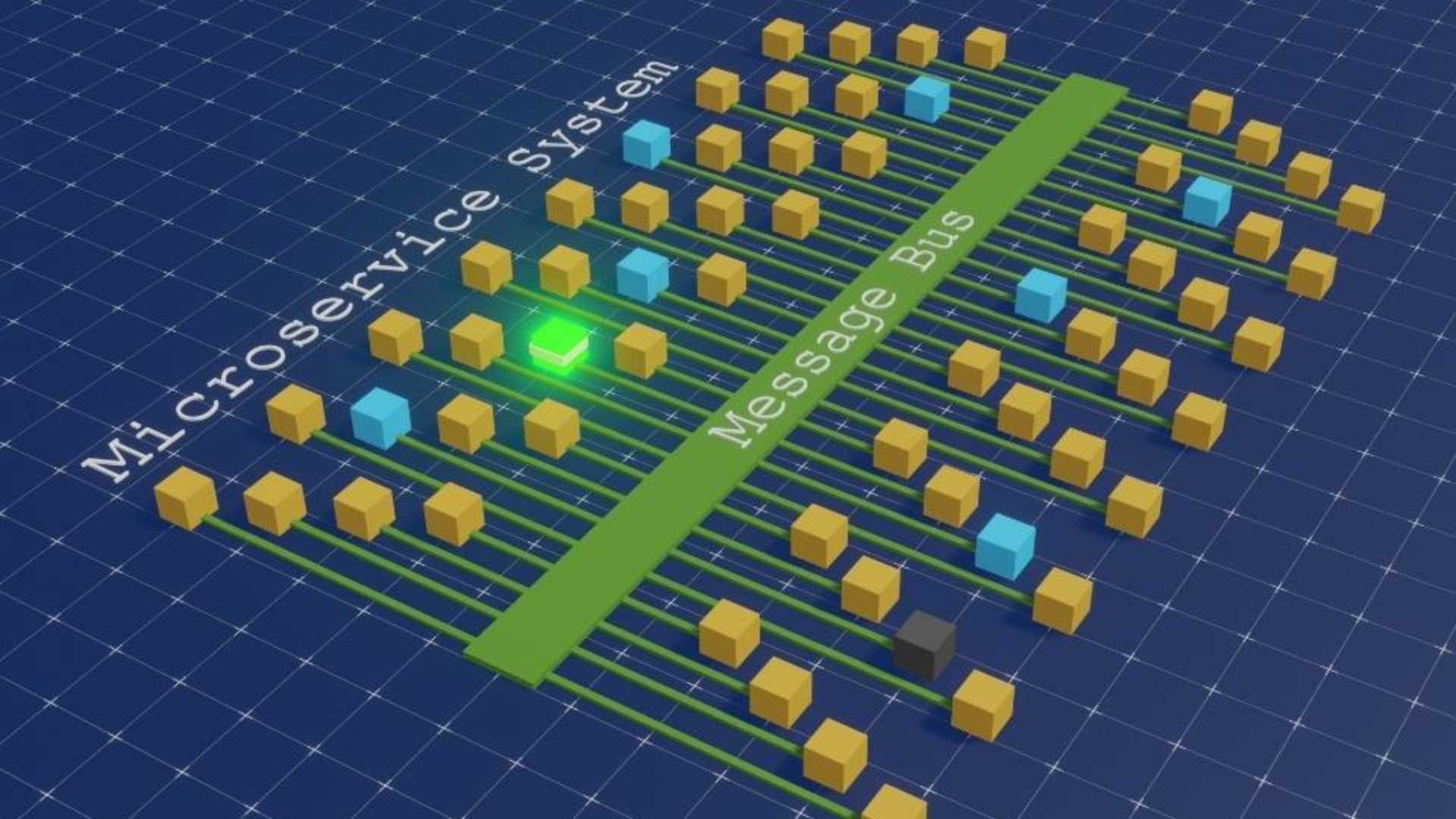

Microservice System
Message Bus

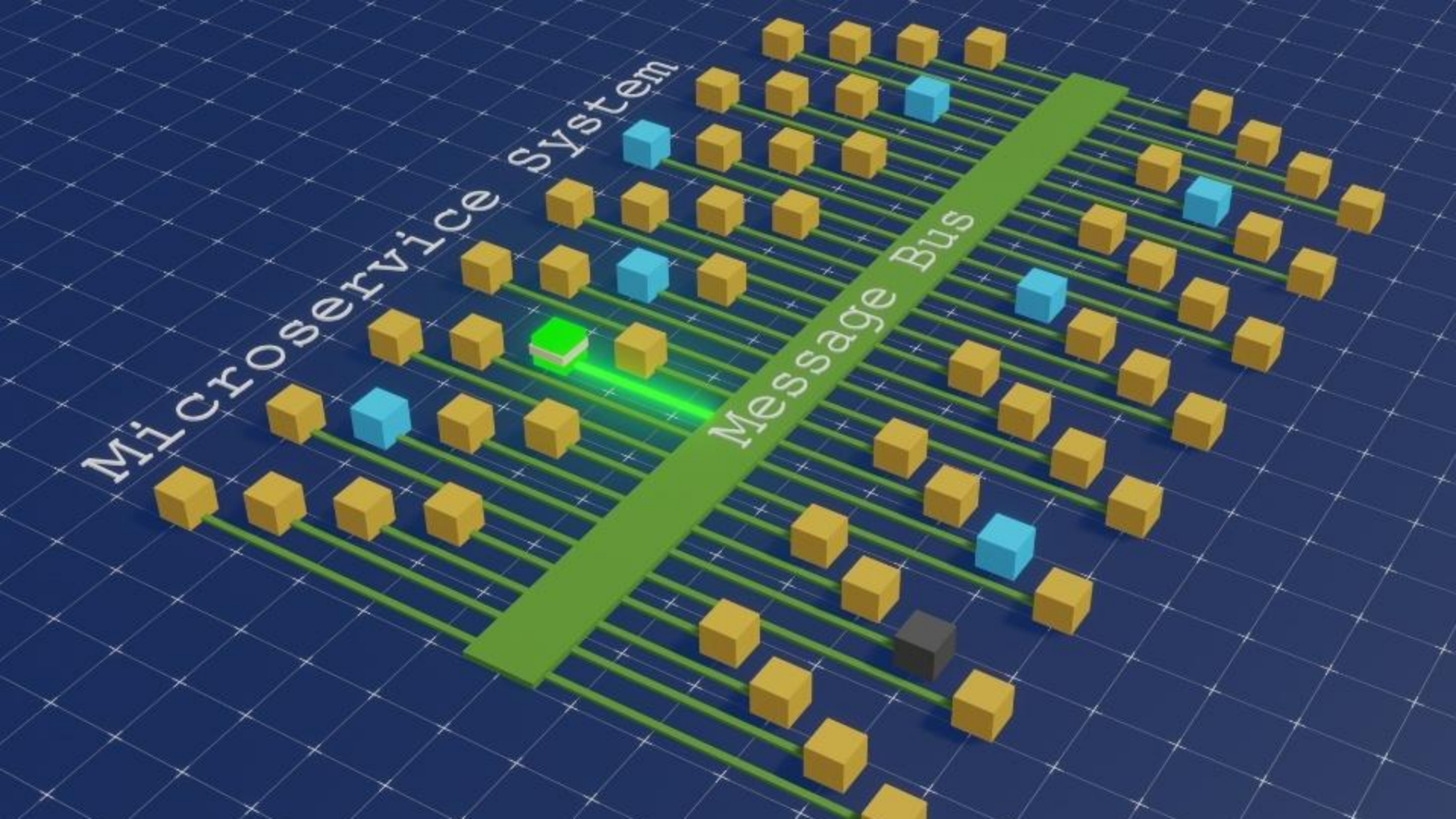

Microservice System
Message Bus

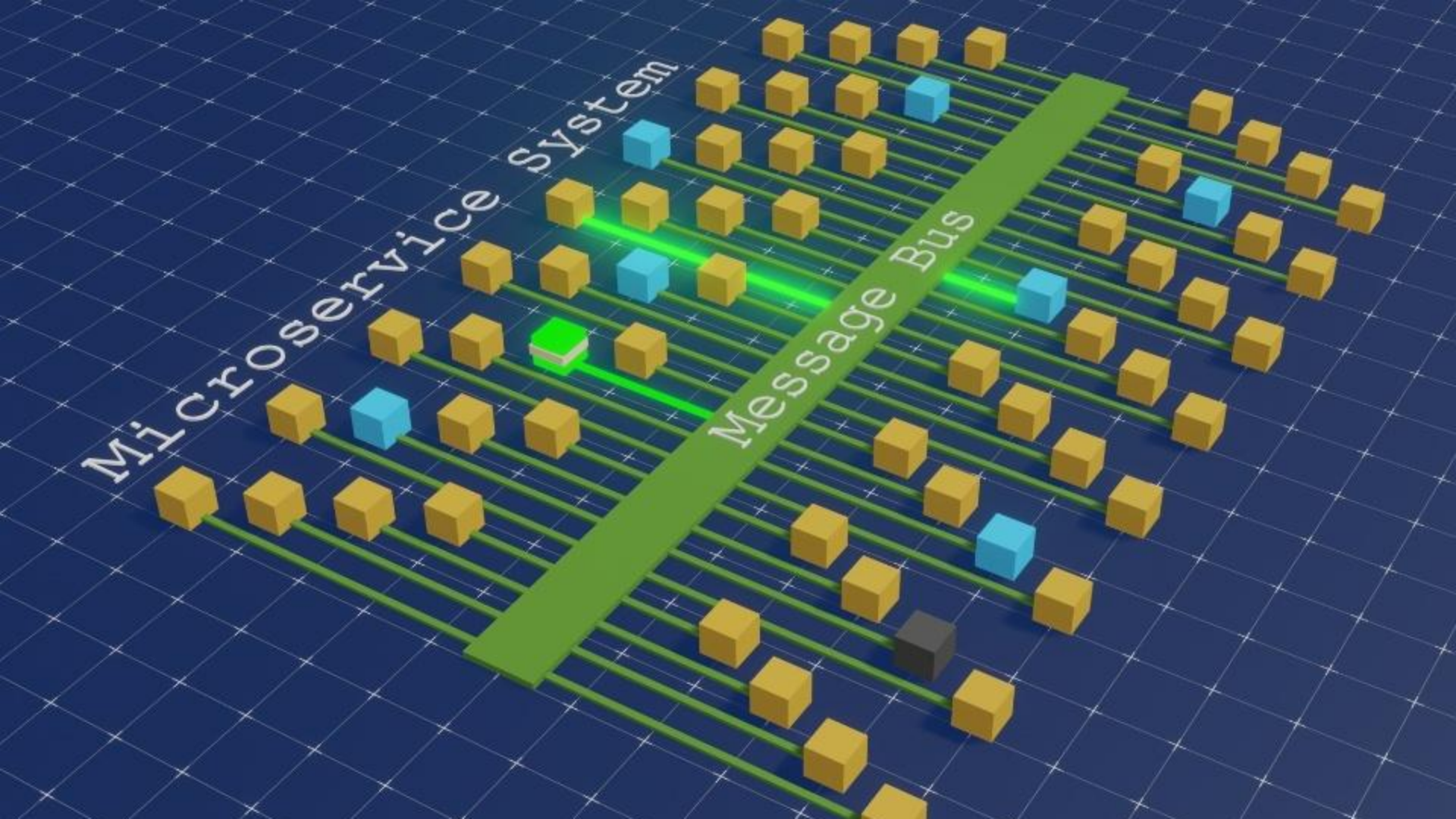

Microservice System
Message Bus

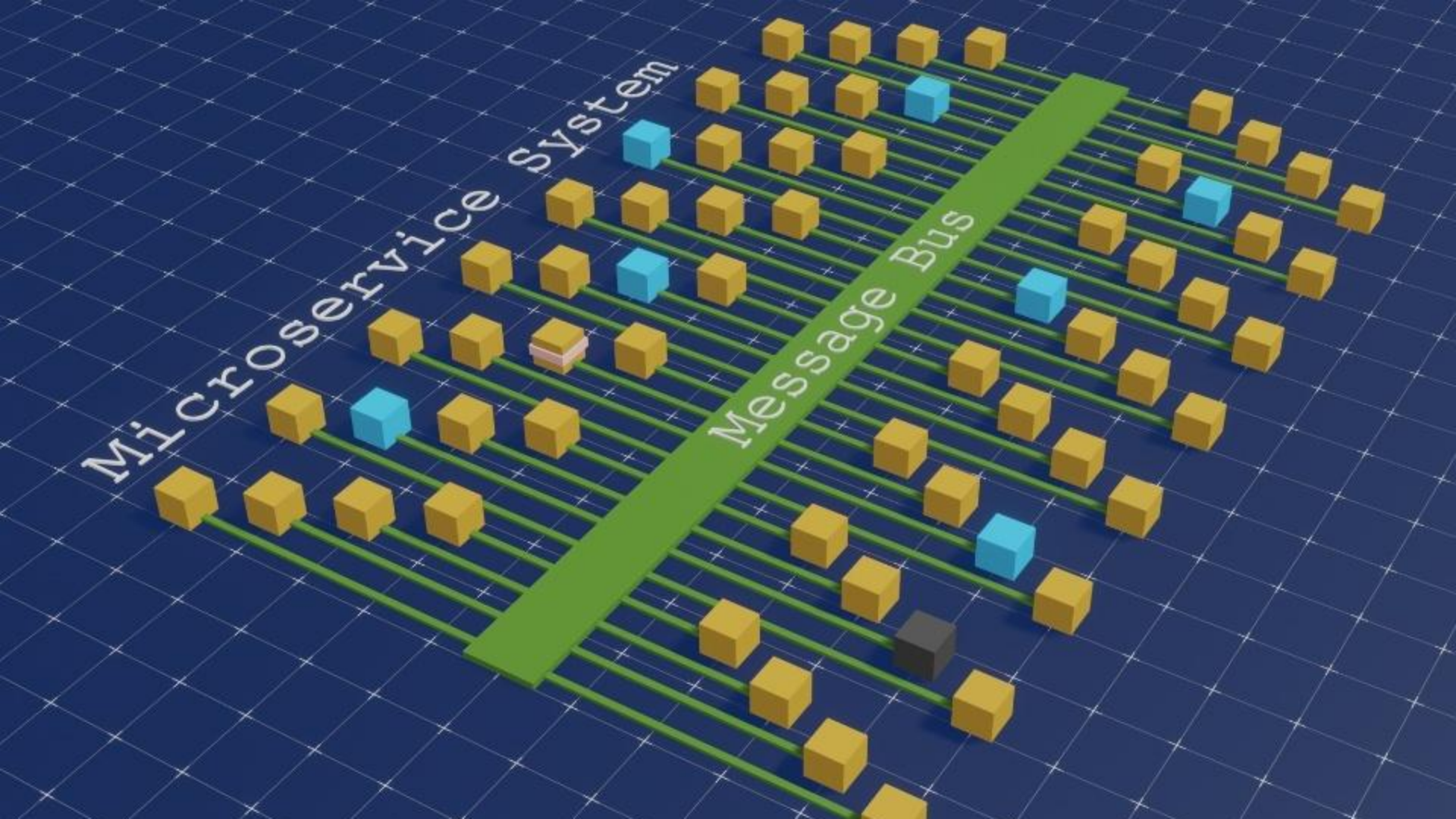

Microservice System
Message Bus

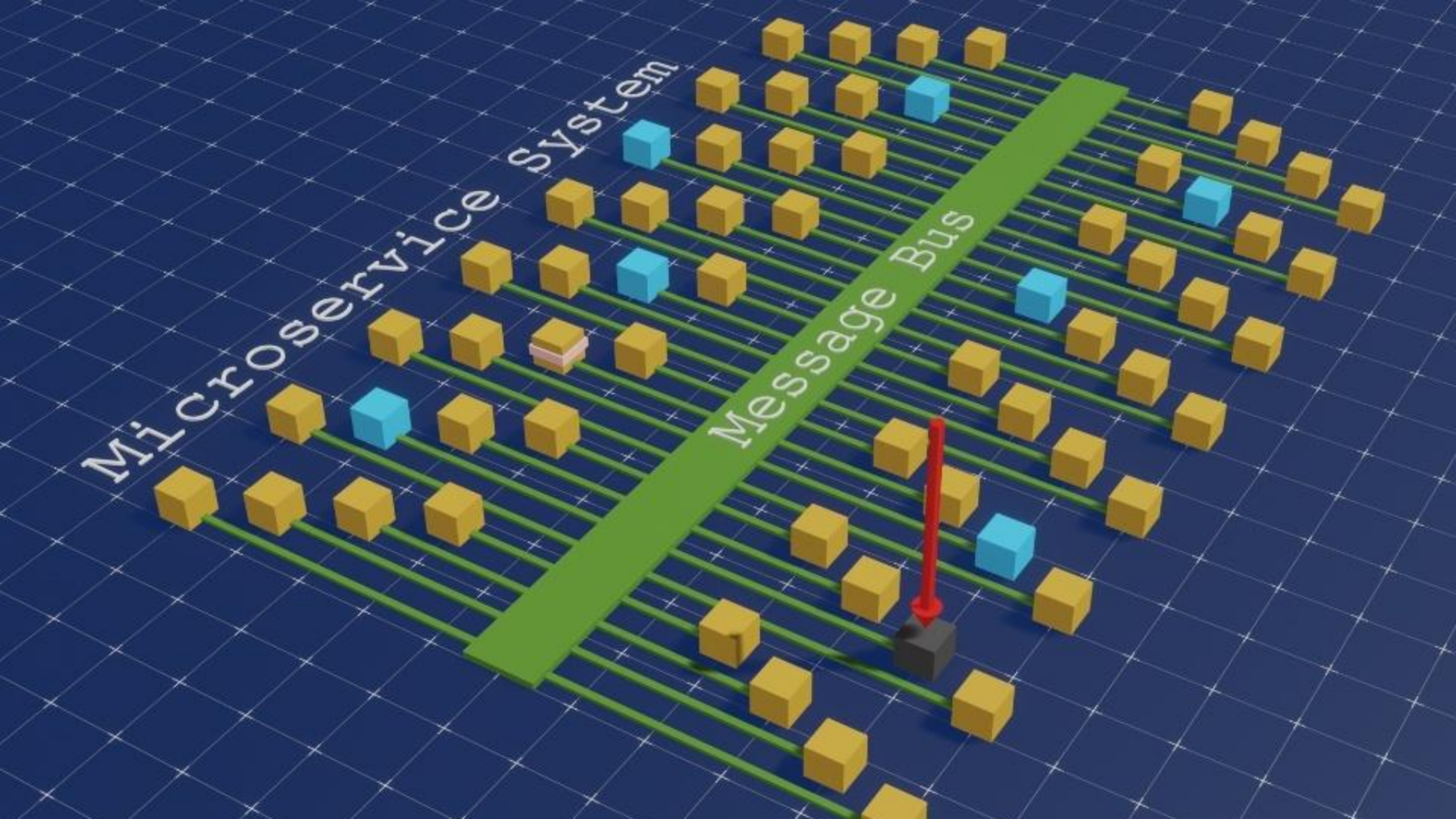

Microservice System
Message Bus

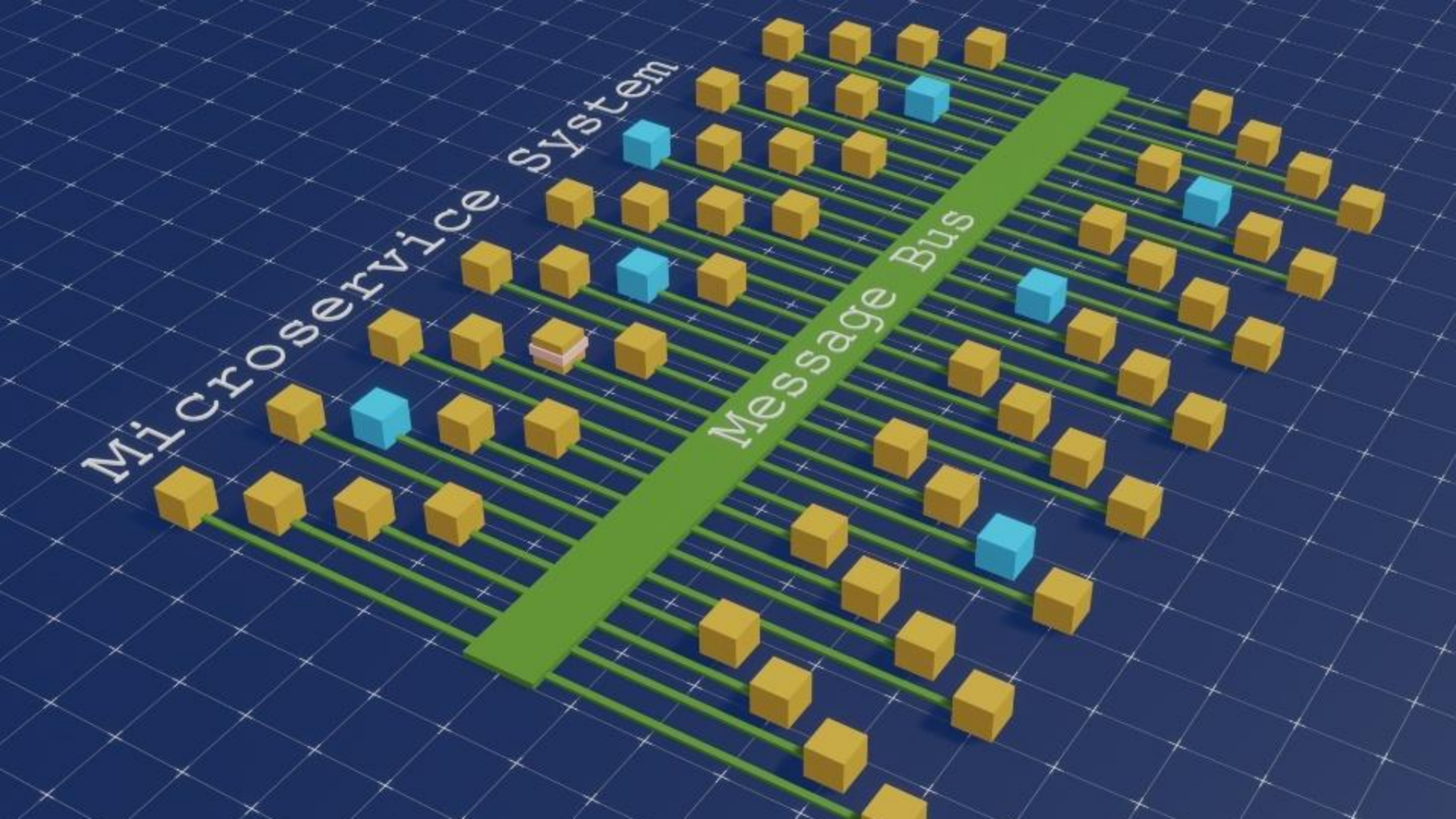

Microservice System
Message Bus

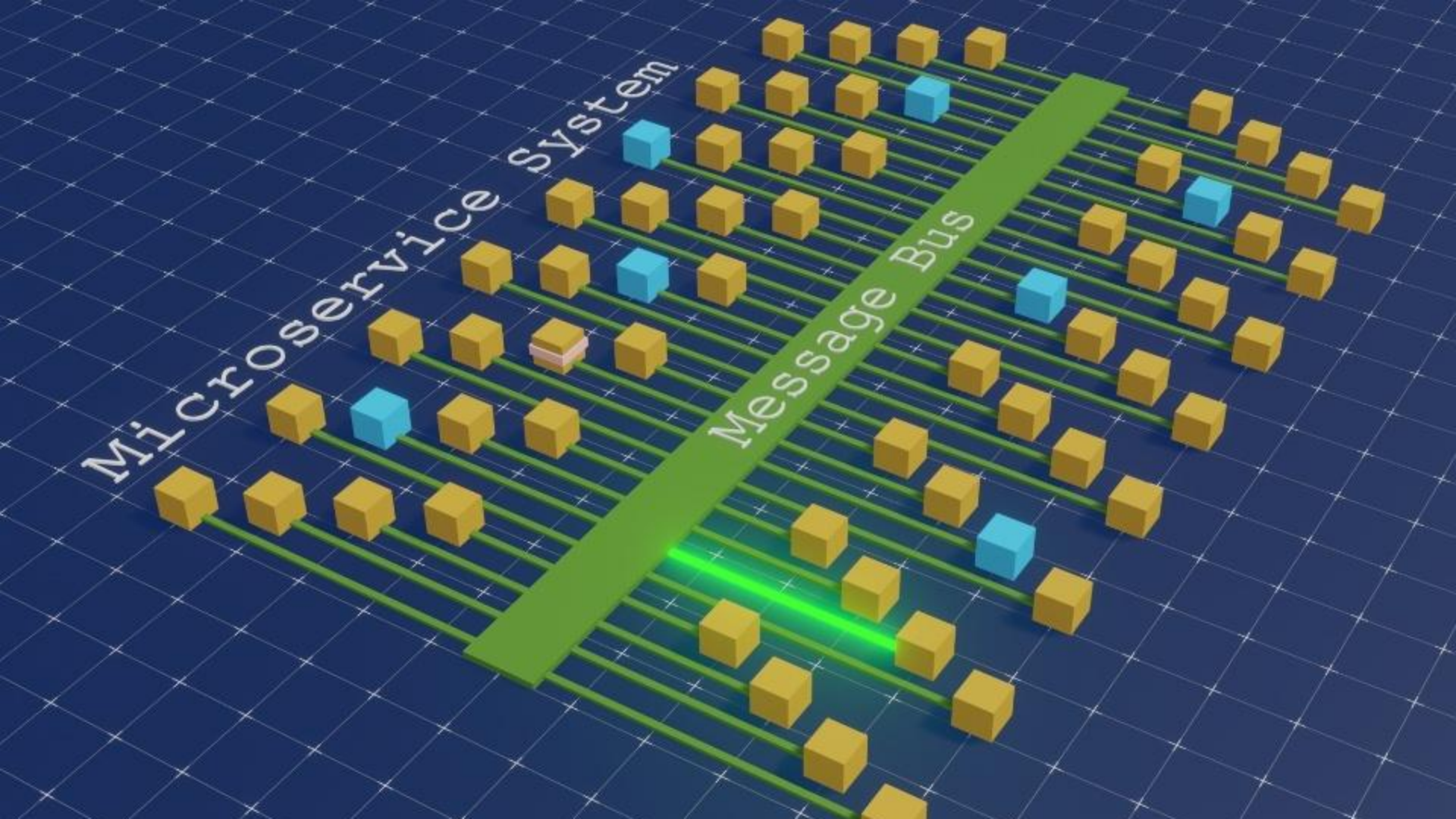

Microservice System
Message Bus

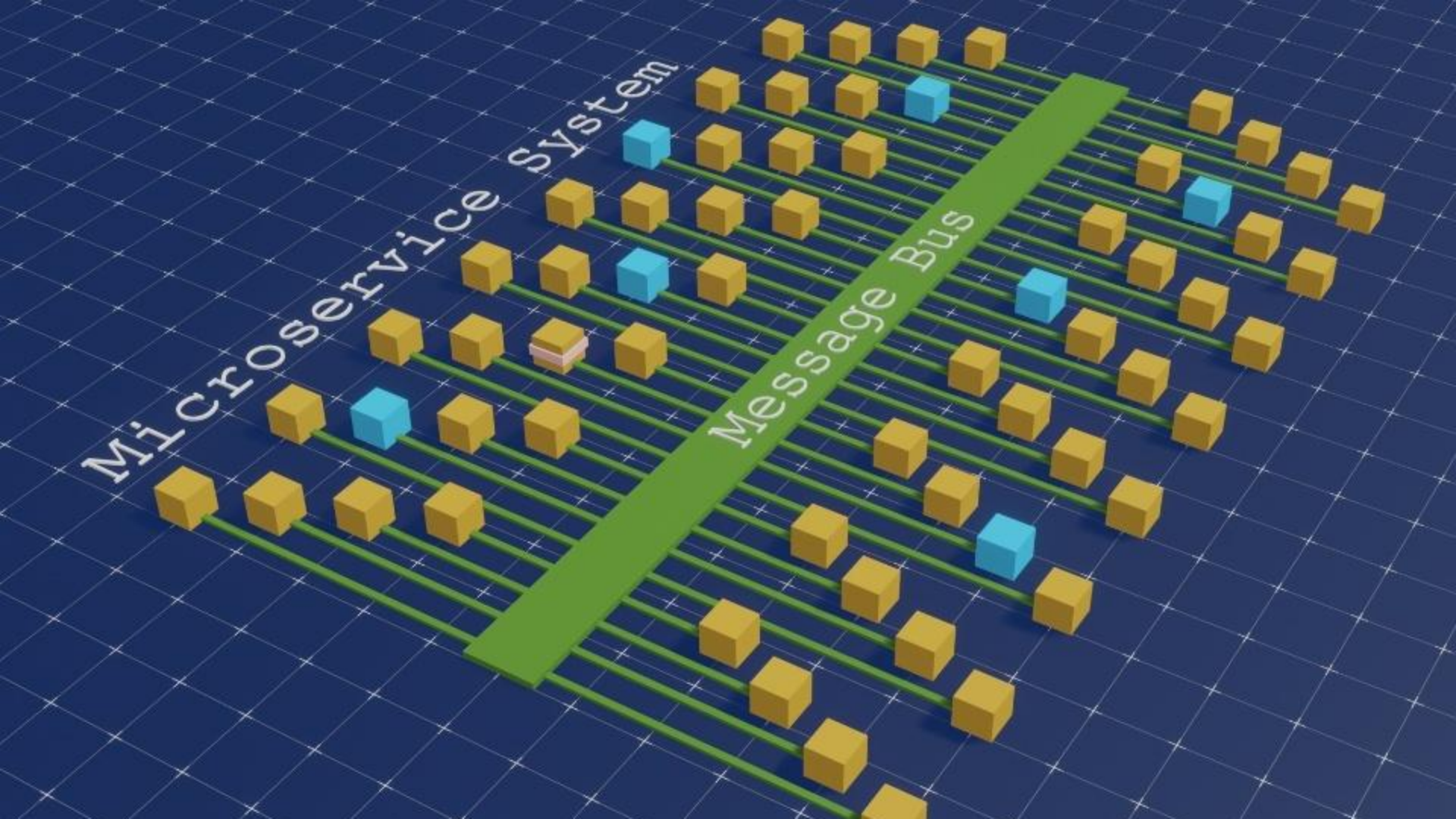

Microservice System
Message Bus

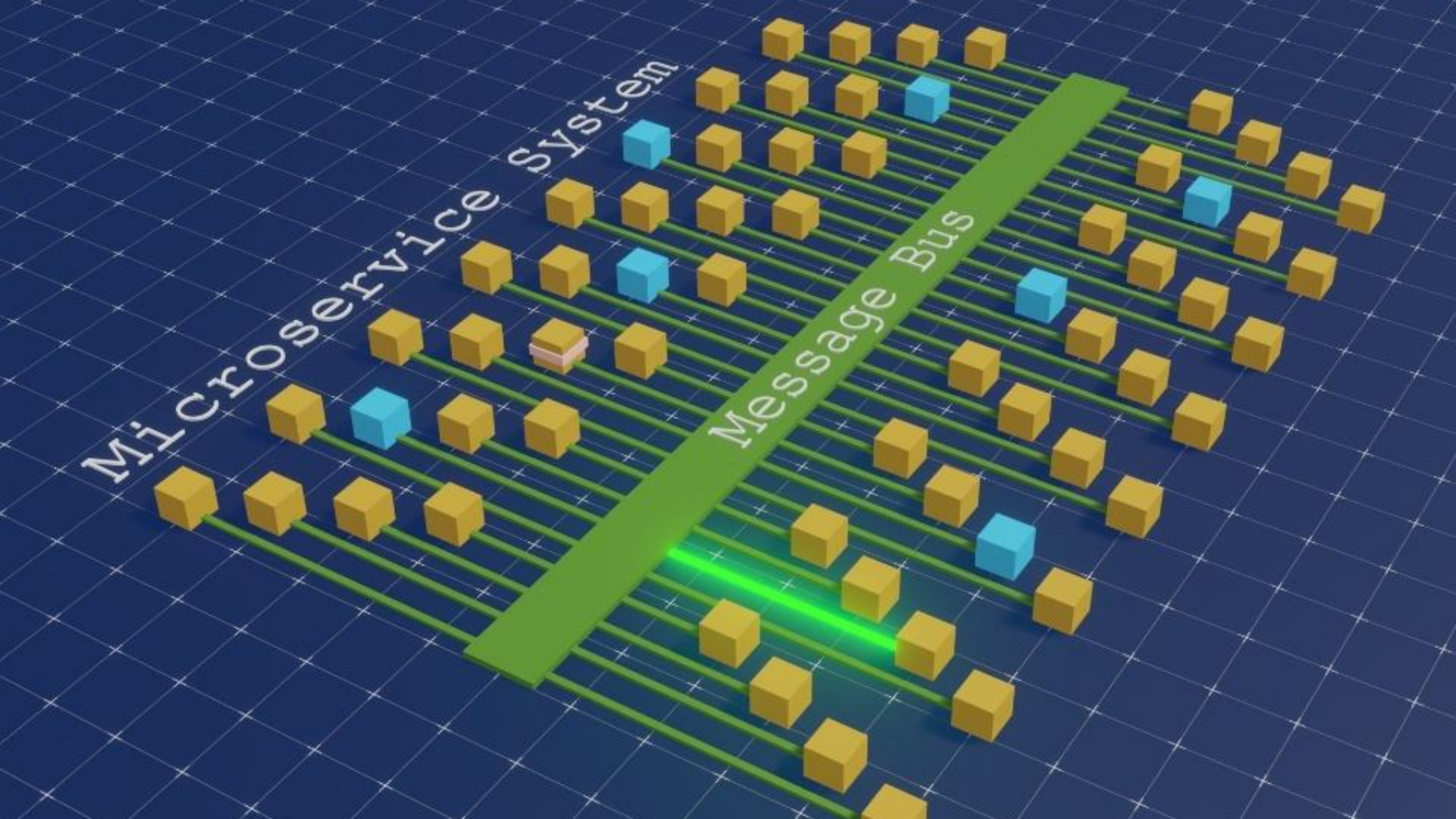

Microservice System
Message Bus

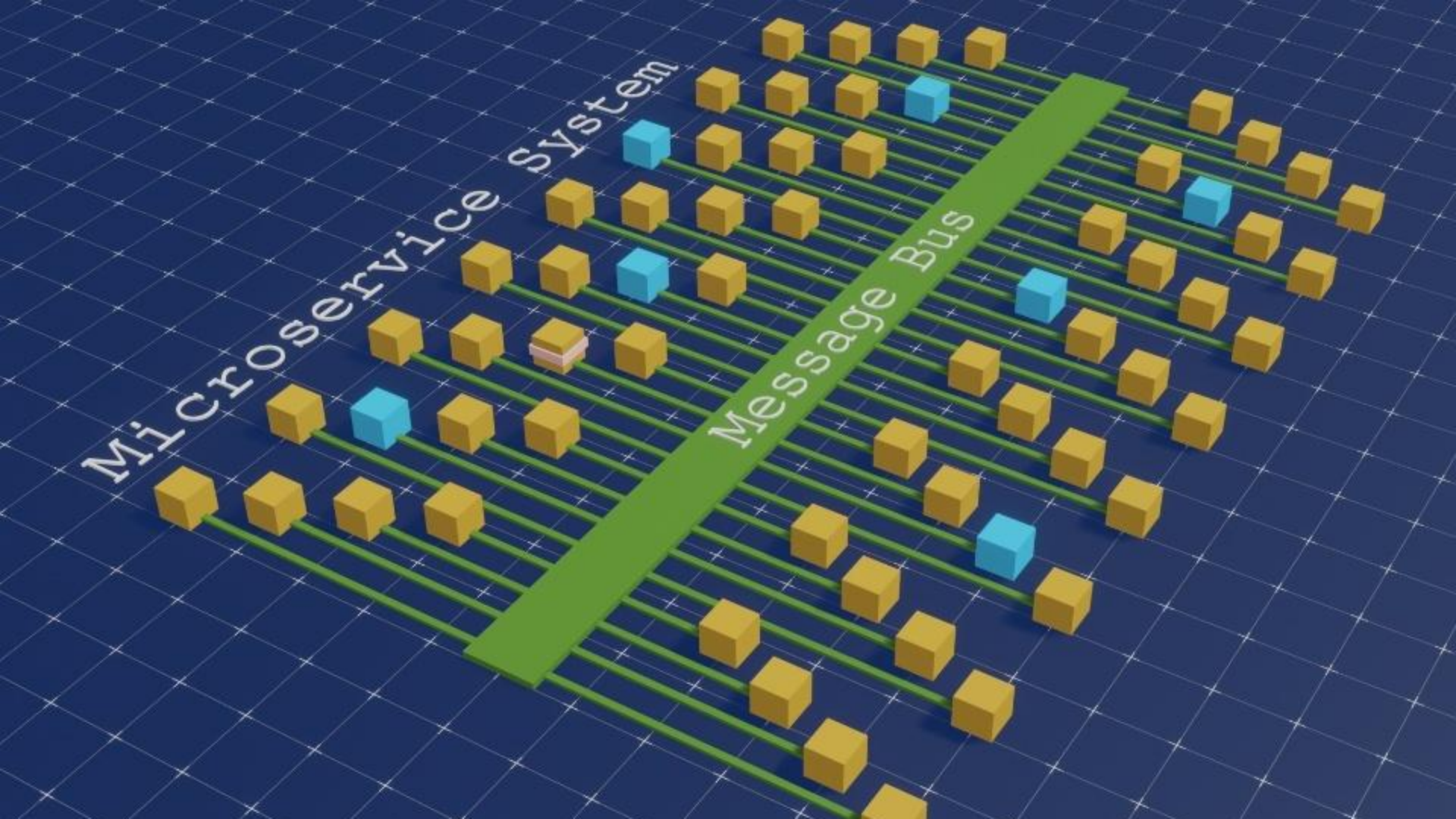

Microservice System
Message Bus

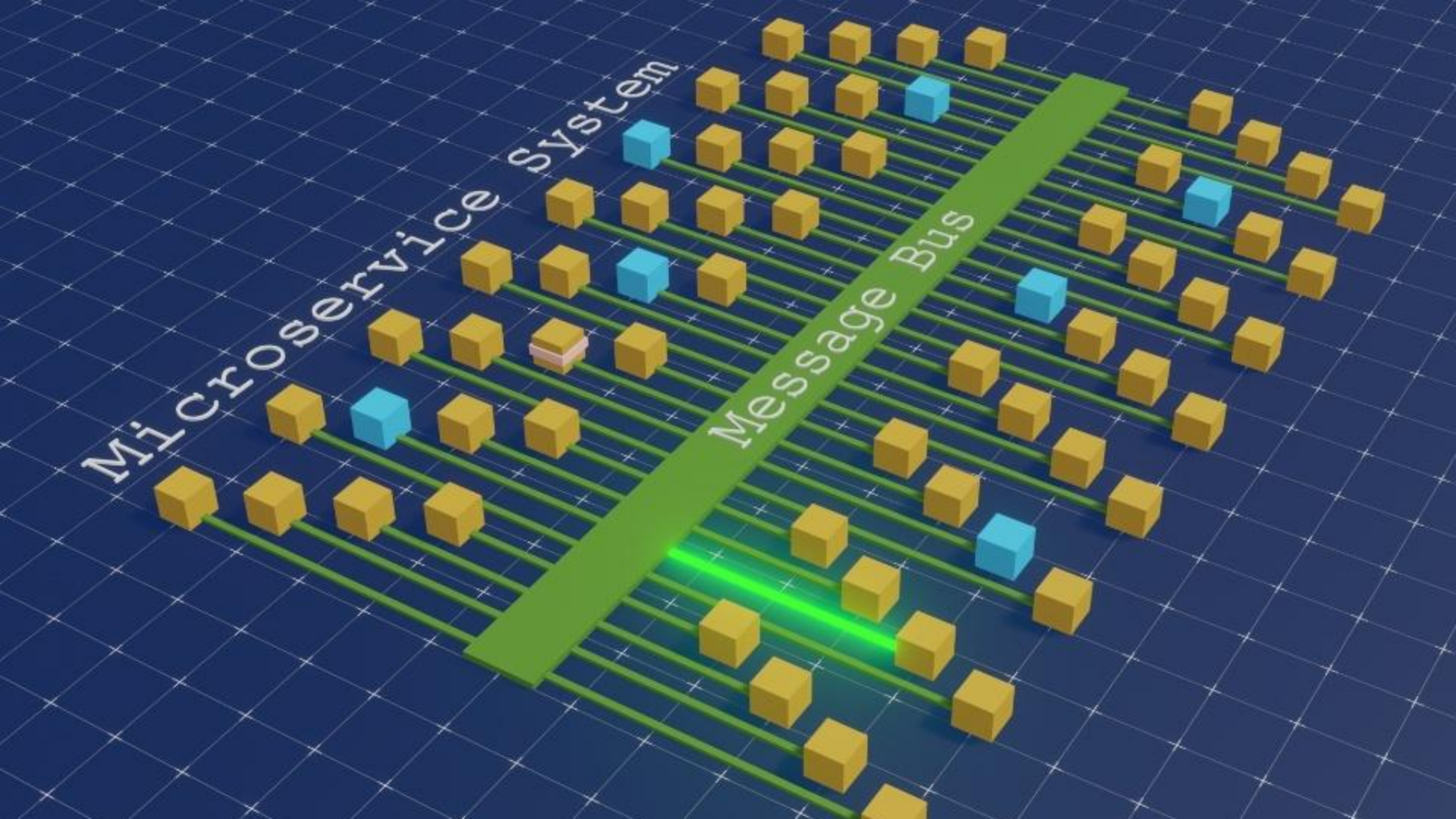

Microservice System
Message Bus

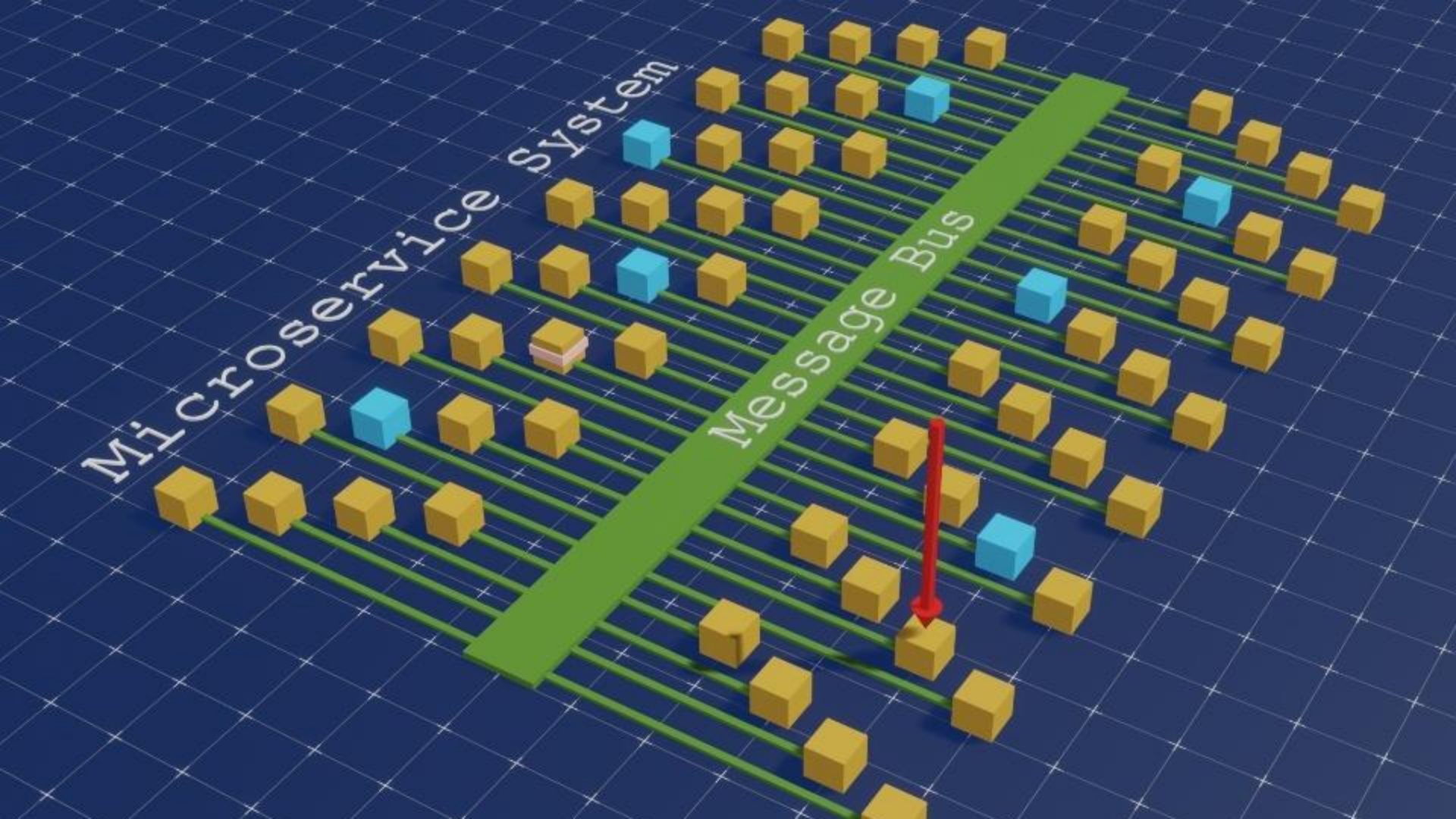

Microservice System
Message Bus


Microservice System
Message Bus


Microservice System
Message Bus


Microservice System
Message Bus

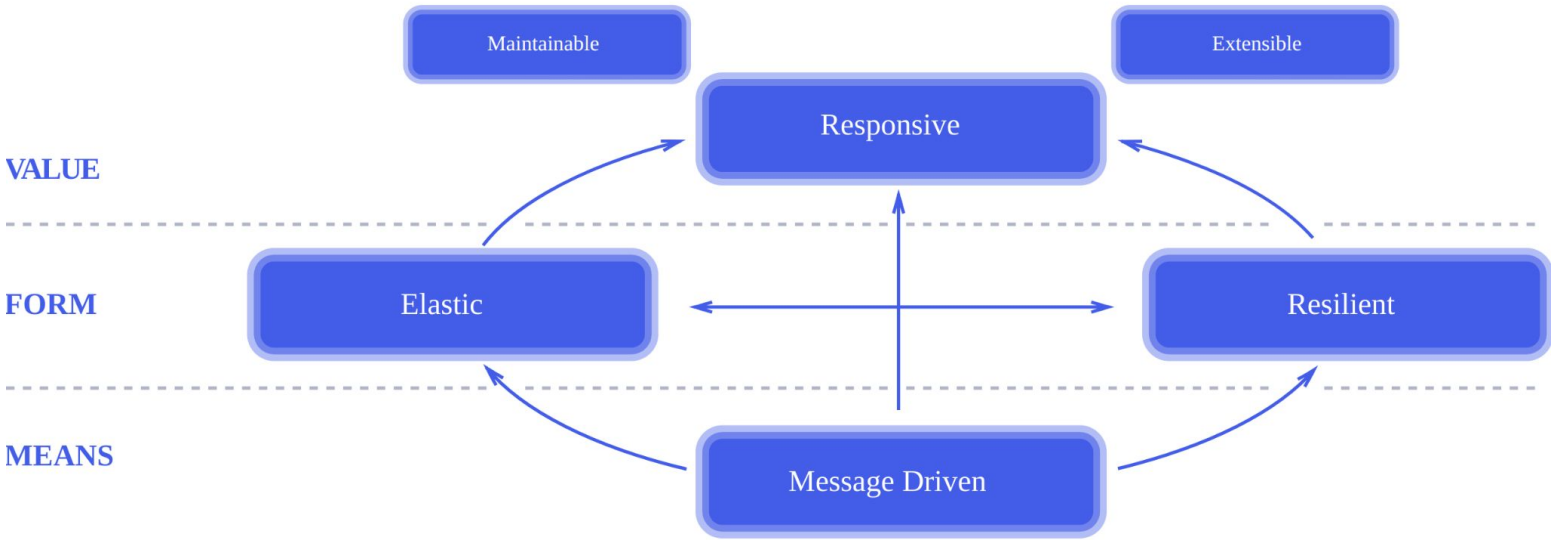

Microservice System
Message Bus


Microservice System
Message Bus

Advantages of Kafka's Communication Approach

- **Decoupled systems:** Easier maintenance and flexibility
- Asynchronous communication with robust delivery guarantees
- Standardized protocols enhance uniformity across systems
- Consumers manage their own data processing pace

The Reactive Manifesto



https://www.reactivemanifesto.org/

Event-Driven - Advantages

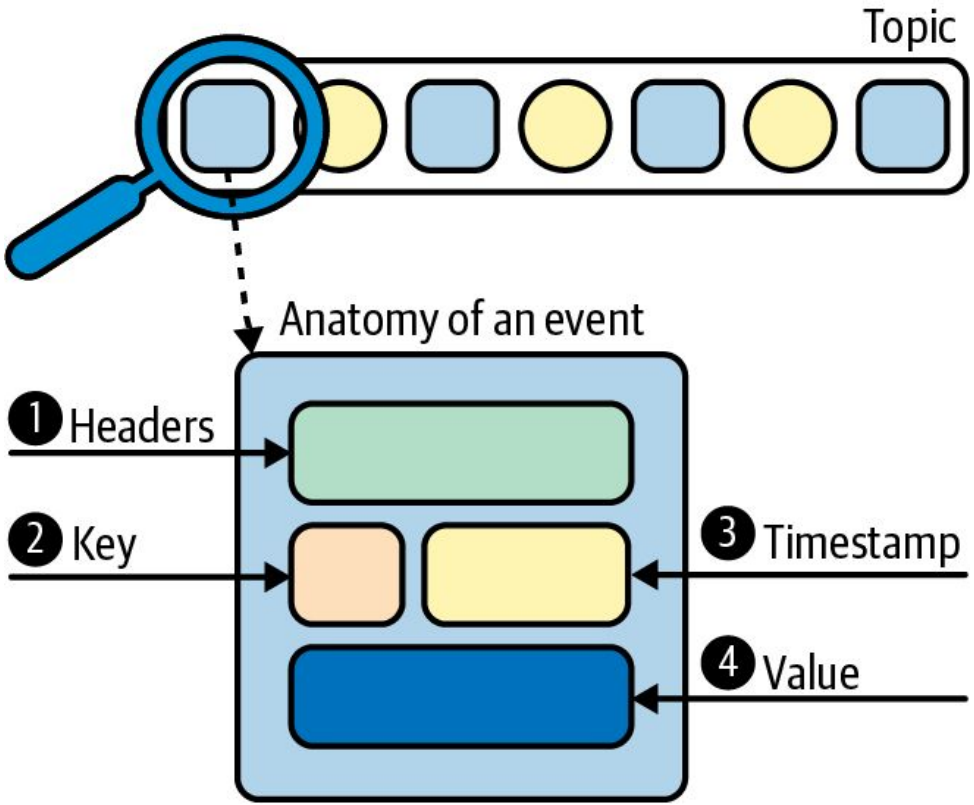
1. Decoupling of services
2. Scalability
3. Resilience and Fault Tolerance
4. Improved Responsiveness & Performance
5. Ease of Integration
6. Enables Reactive Programming Models

Event

Introduction to Data in Kafka Topics

- Kafka uses terms like messages, records, and events interchangeably.
- Preferred term in this context: Event.
- Definition of an event: A timestamped key-value pair that records an occurrence.

Anatomy of an Event in Kafka



Anatomy of an Event in Kafka

- **Application-level Headers:** Optional metadata, not commonly focused on in this text.
- **Keys**: Optional but vital for data distribution across partitions; help in correlating related records.
- **Timestamp**: Associates each event with the time of its occurrence; detailed exploration in later chapters.
- **Value**: Contains the event's actual data, stored as a byte array; requires deserialization by client applications.

Kafka Internal Arch

https://docs.google.com/presentation/d/1oZU7PEcszM1C1HmR7u7EI5-DhRiQJNKi/edit?usp=sharing&ouid=116008769779639876320&rtpof=true&sd=true