ilionx

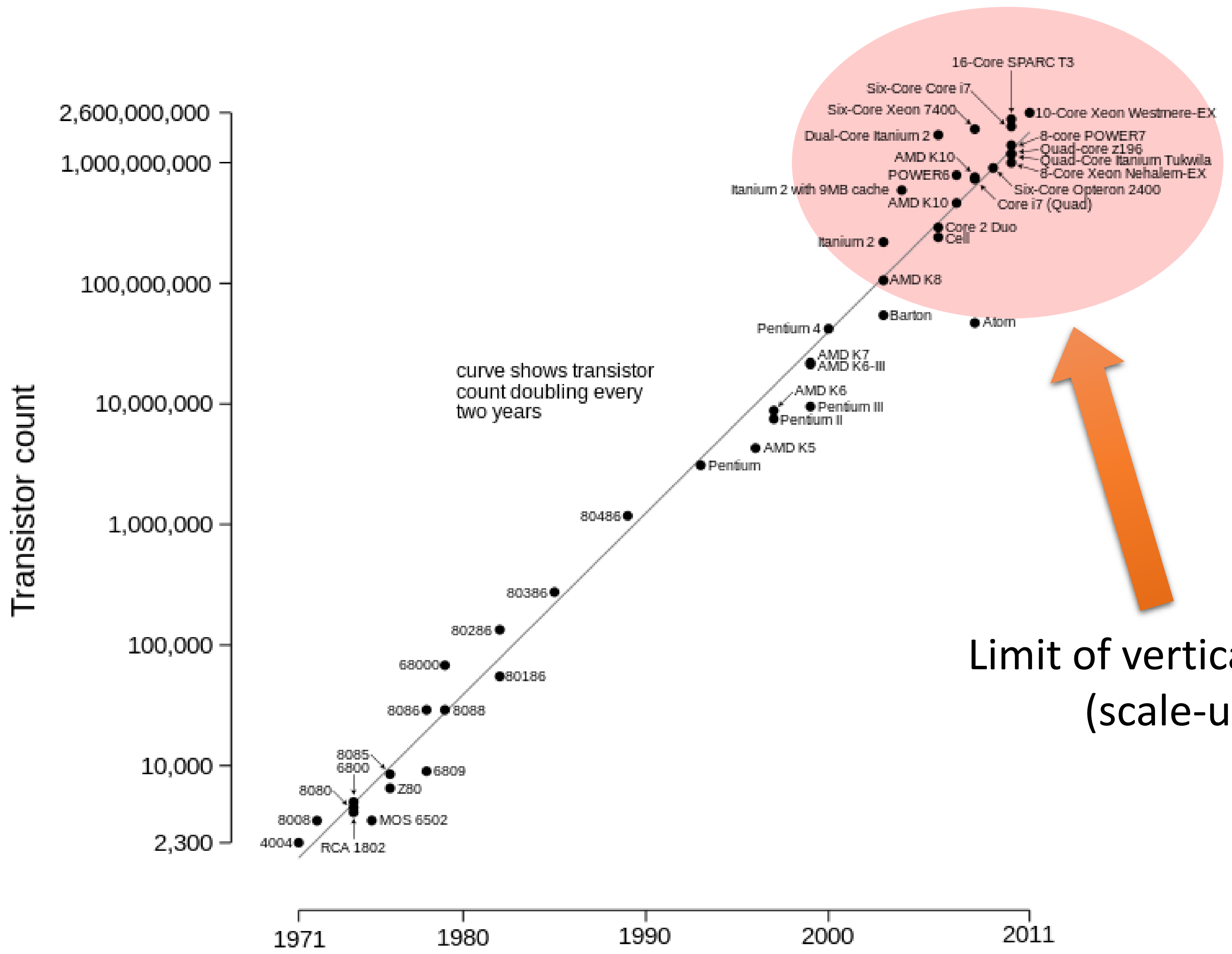
Your partner in digital business

Meetup

Micro Services with Service Fabric

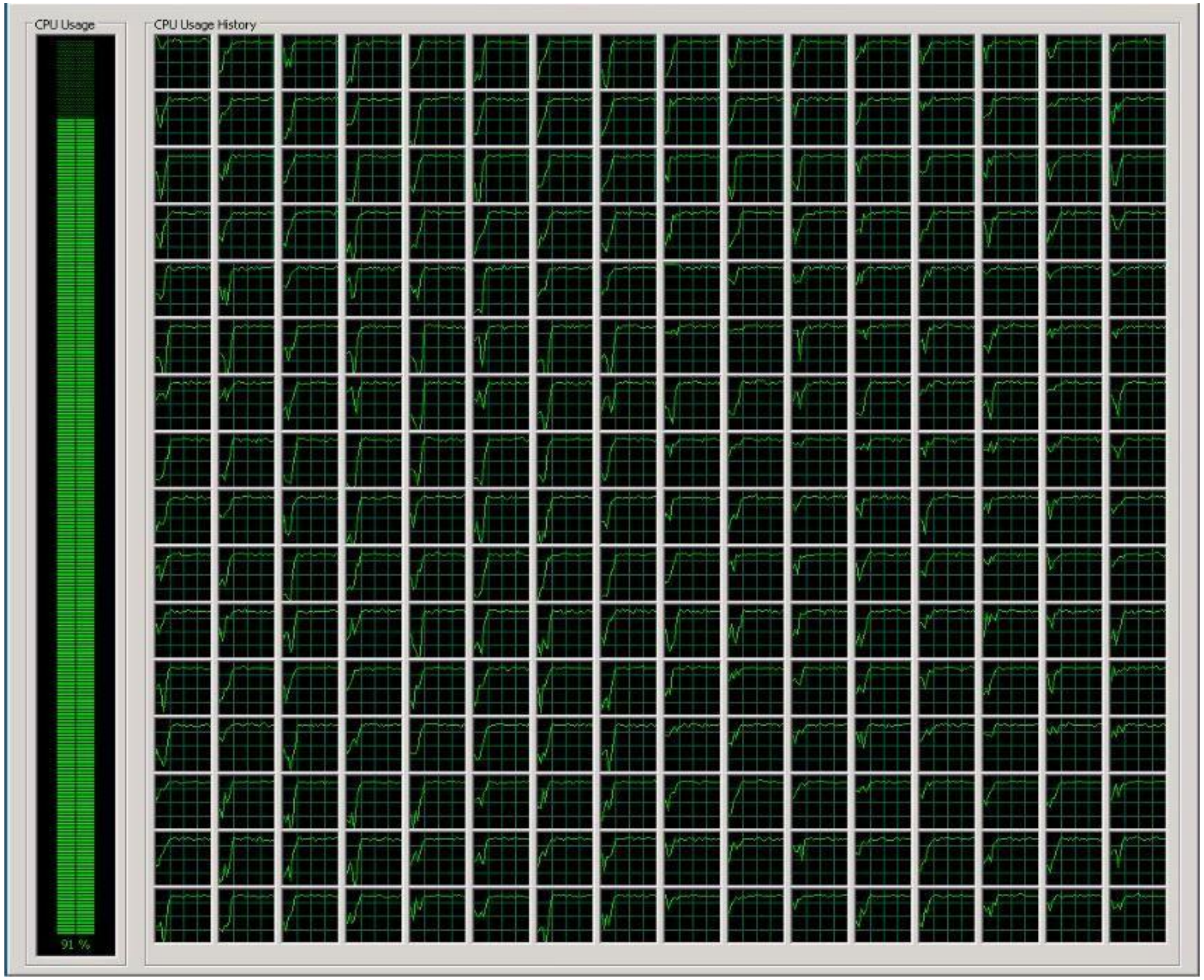
- Probleem gebied
- Micro Services
- Azure Service Fabric
- Demo's

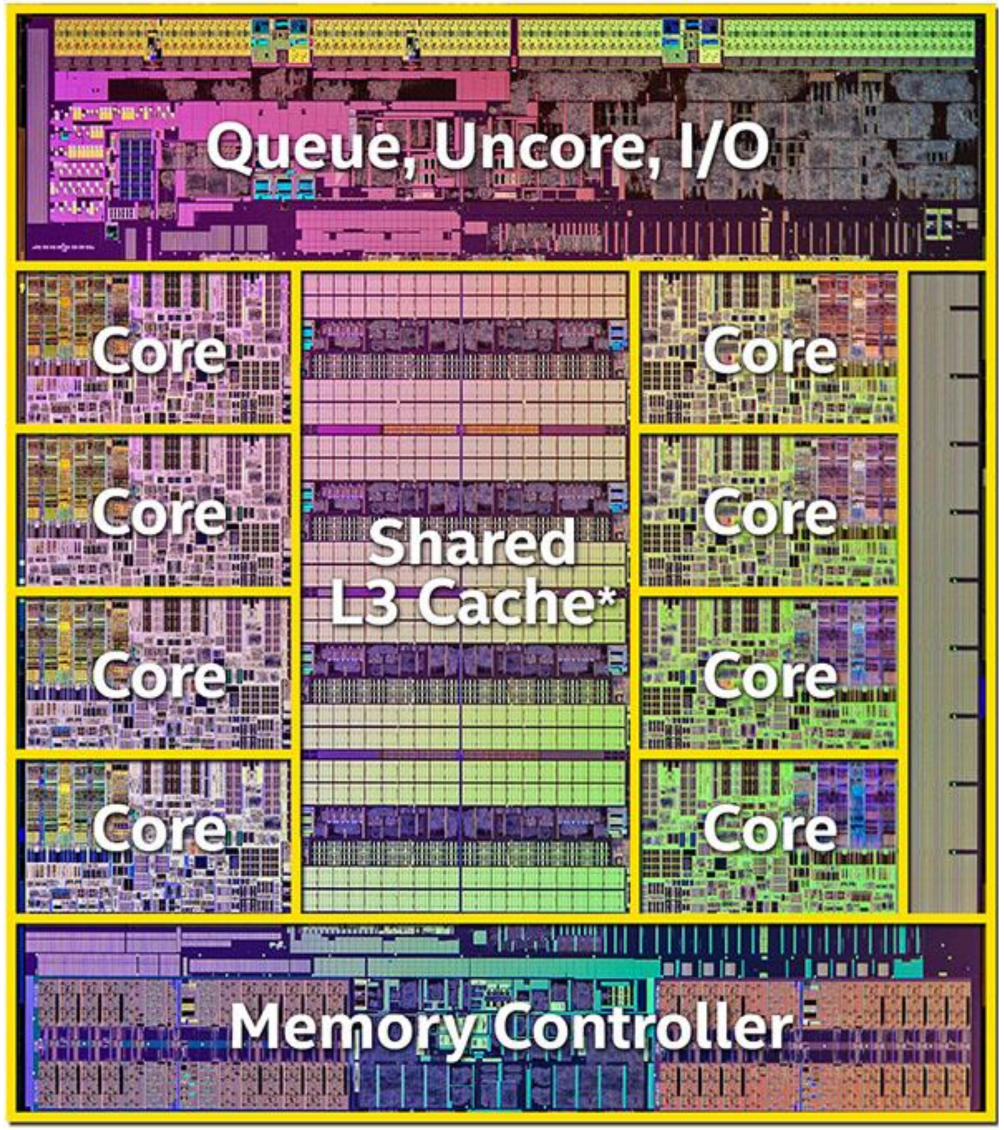
Moore's Law



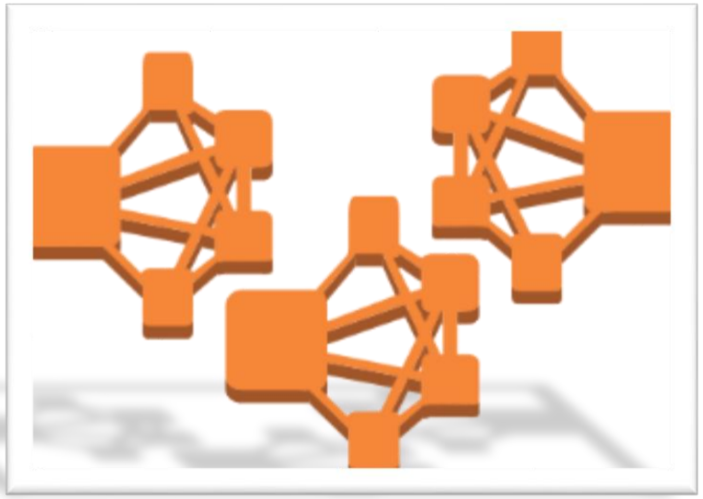
Limit of vertical scale?
(scale-up)

256 core processor





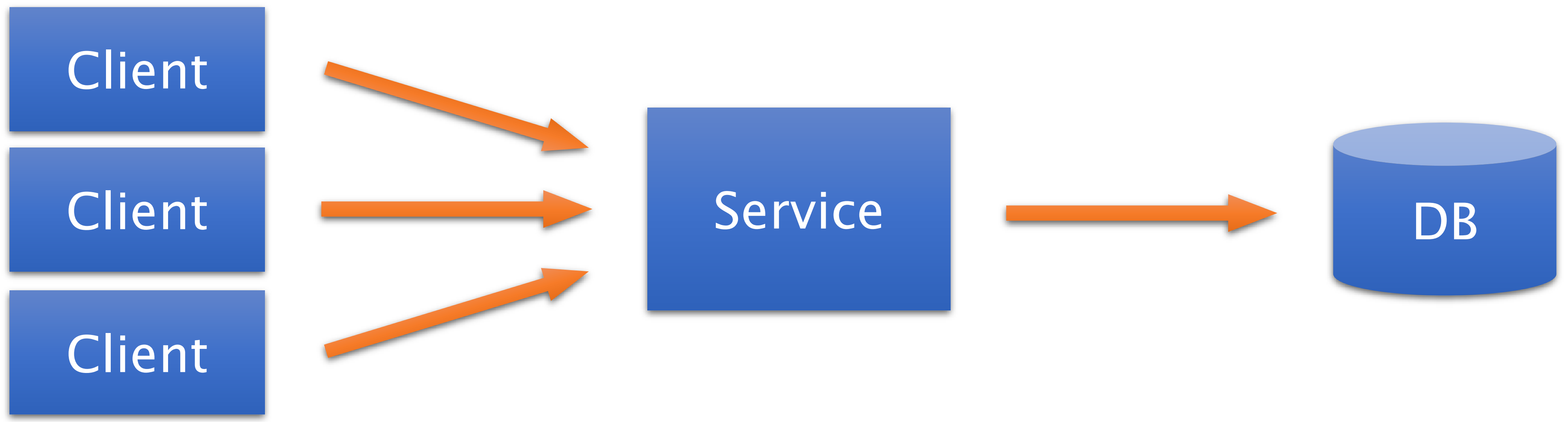
Distributed Systems



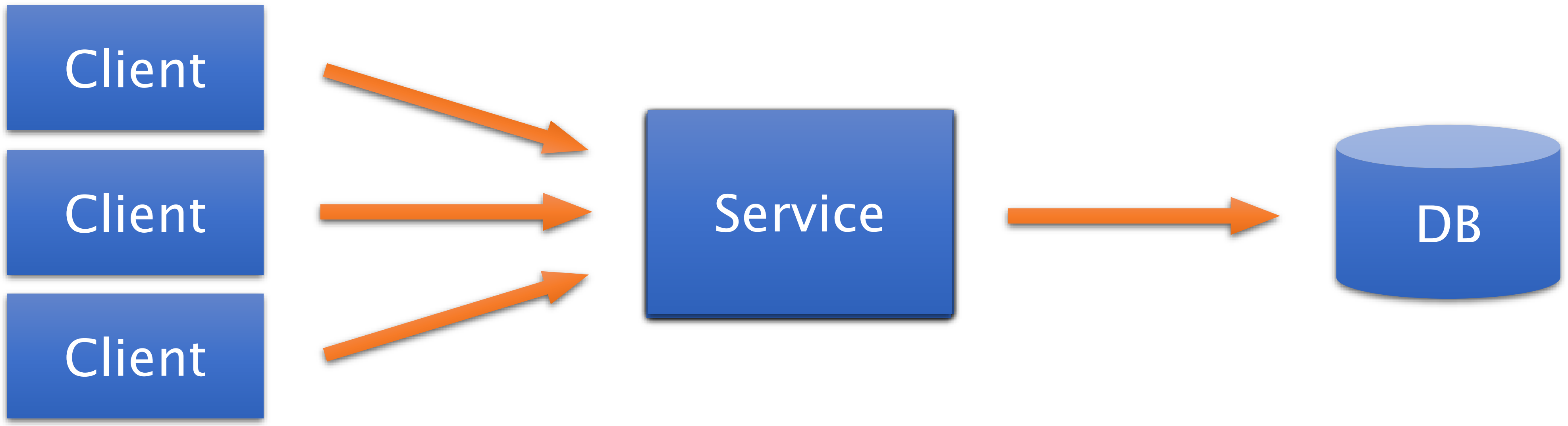
ilionx
Your partner in digital business

Common approach

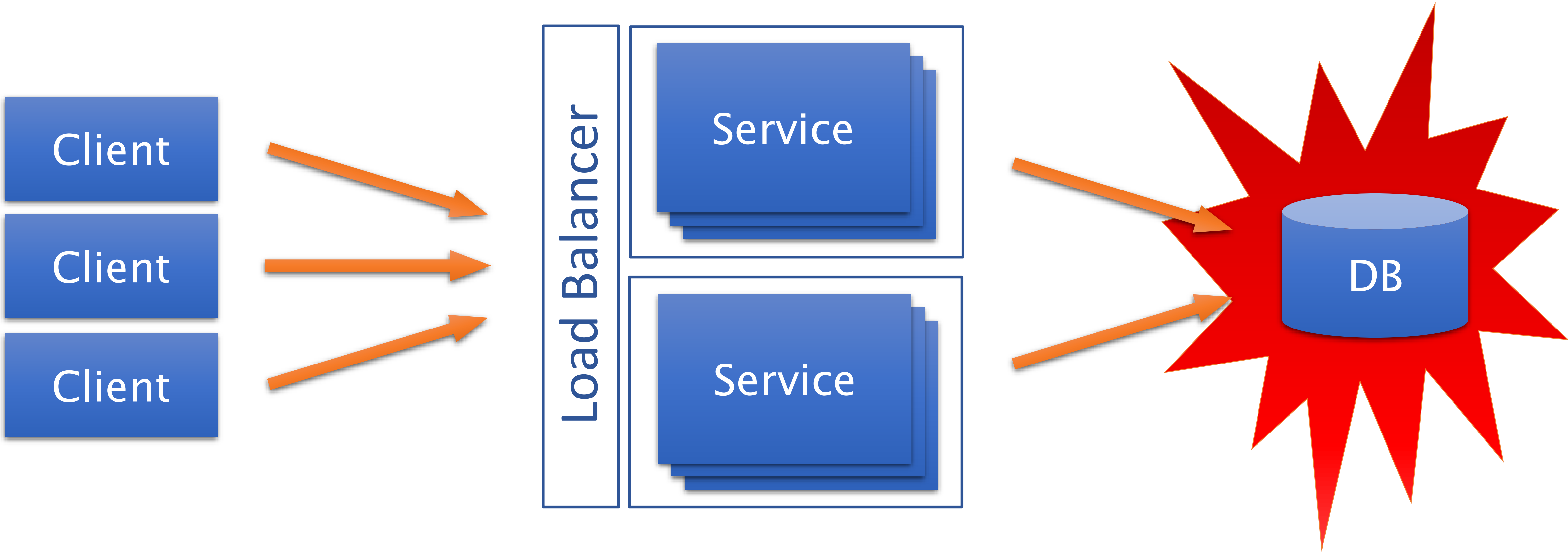
- Client Server model
- 3-tier architecture
- Object Oriented (C#, java, C++, …)
- RPC, DCOM, SOAP, WCF, REST, …



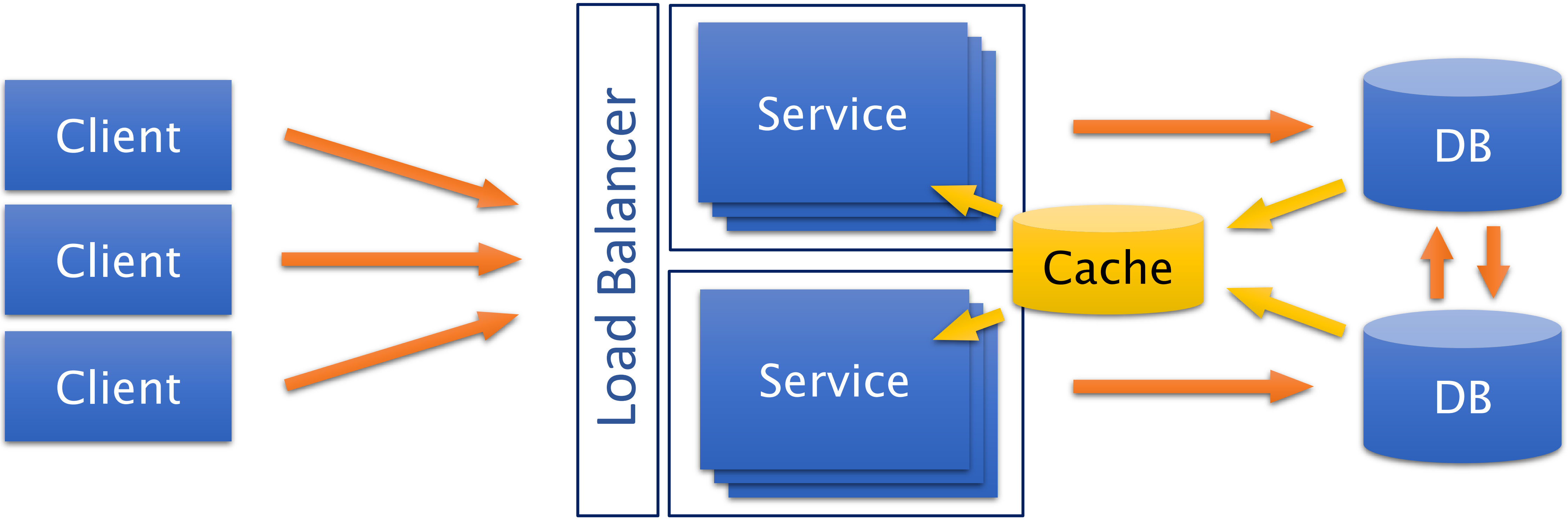
Middleware scale-out



Middleware scale-out

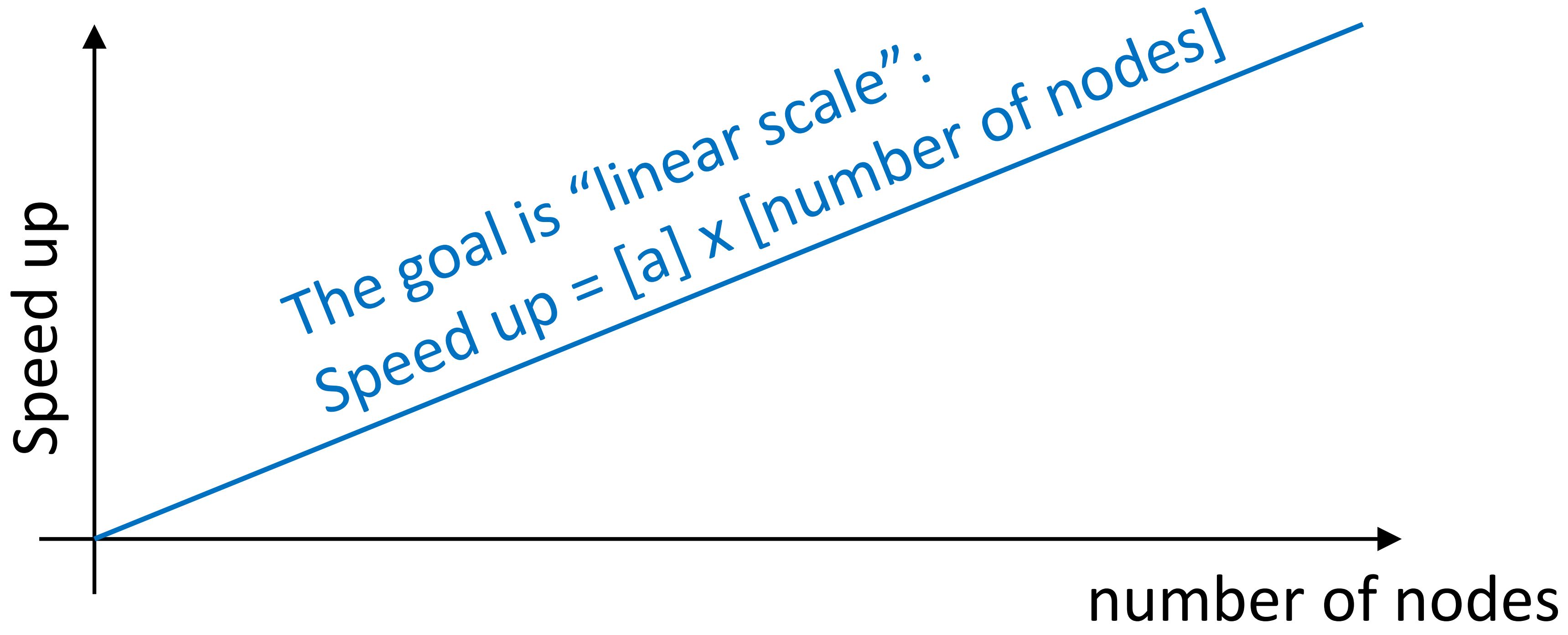


Database partitioning



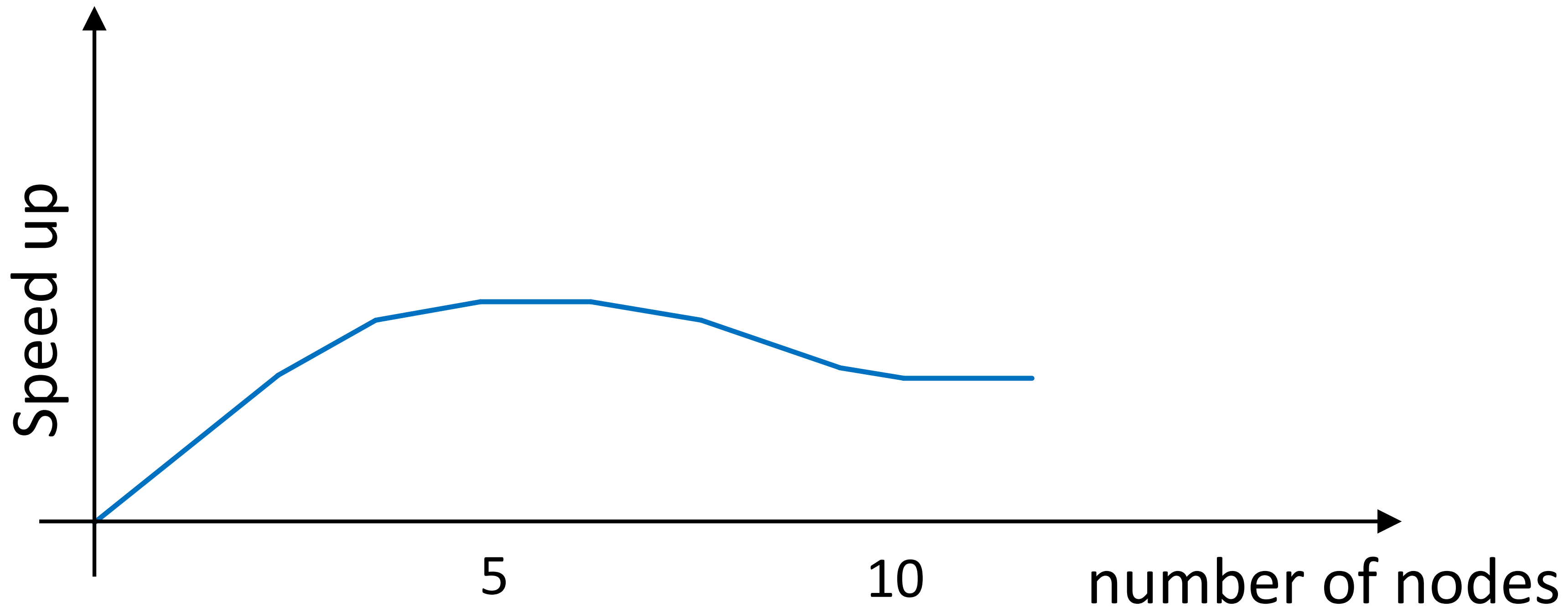
The problem of scale-out

- Concurrency
- Partitioning
- High throughput

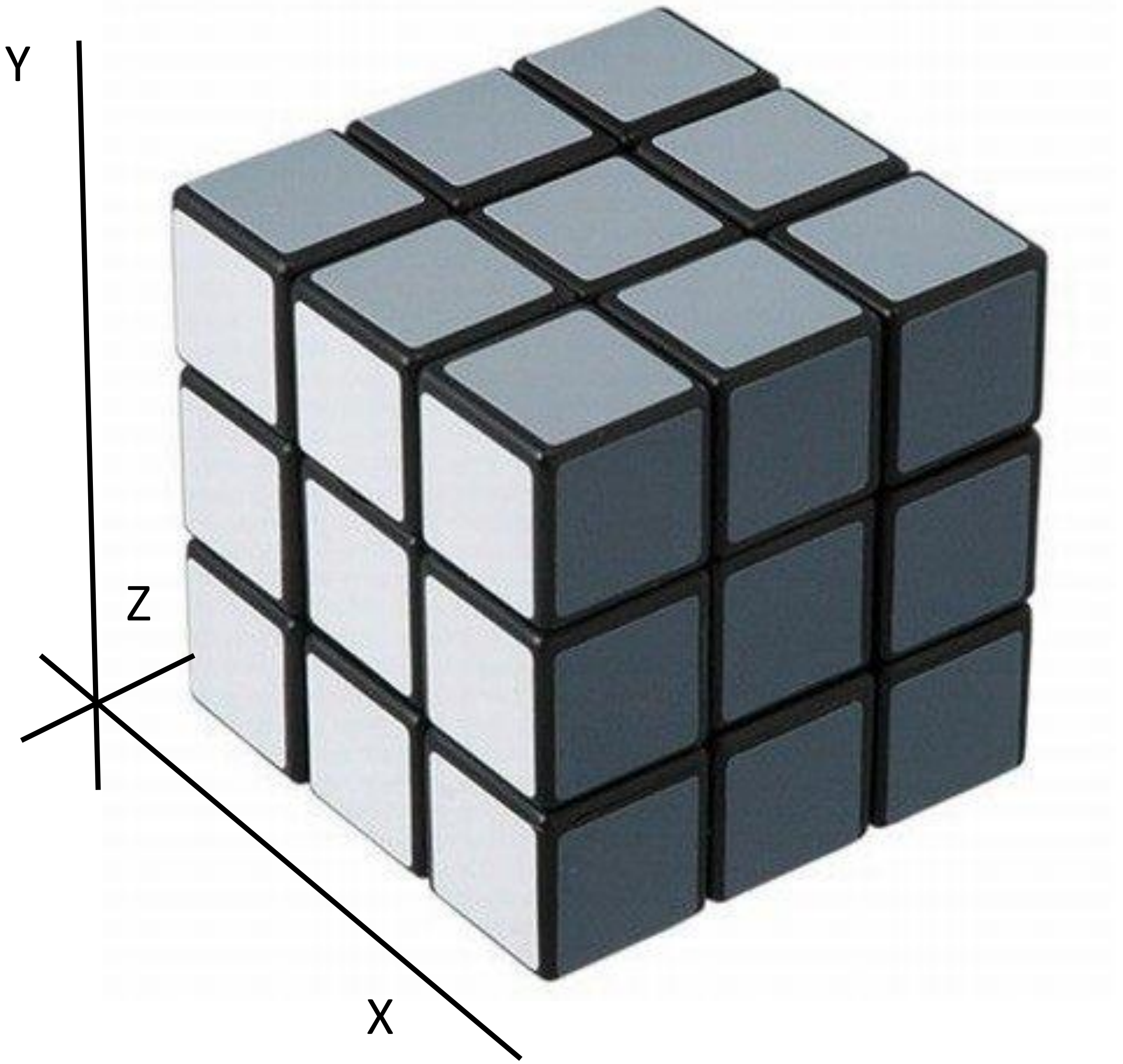


"Speed-up" of traditional approach…

- Reality is non linear
- More nodes does not mean more scale



Need more scaling options → Scale Cube (3D)



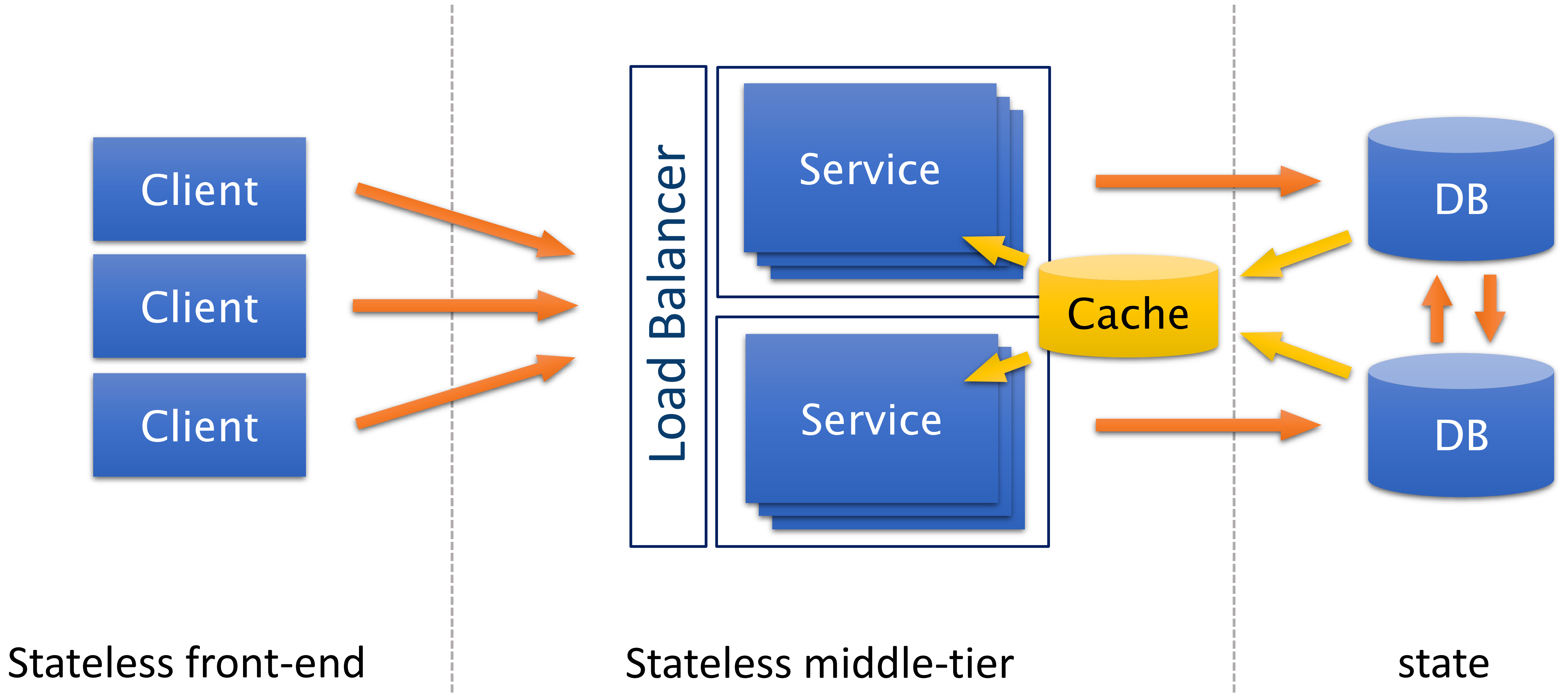
- X
  - Horizontal scale
  - More nodes
  - Network Load Balancing
- Y
  - Functional scale
  - Micro services
- Z
  - Data partitioning
  - Separate tenants
  - Separate by region

Micro Services: the goal

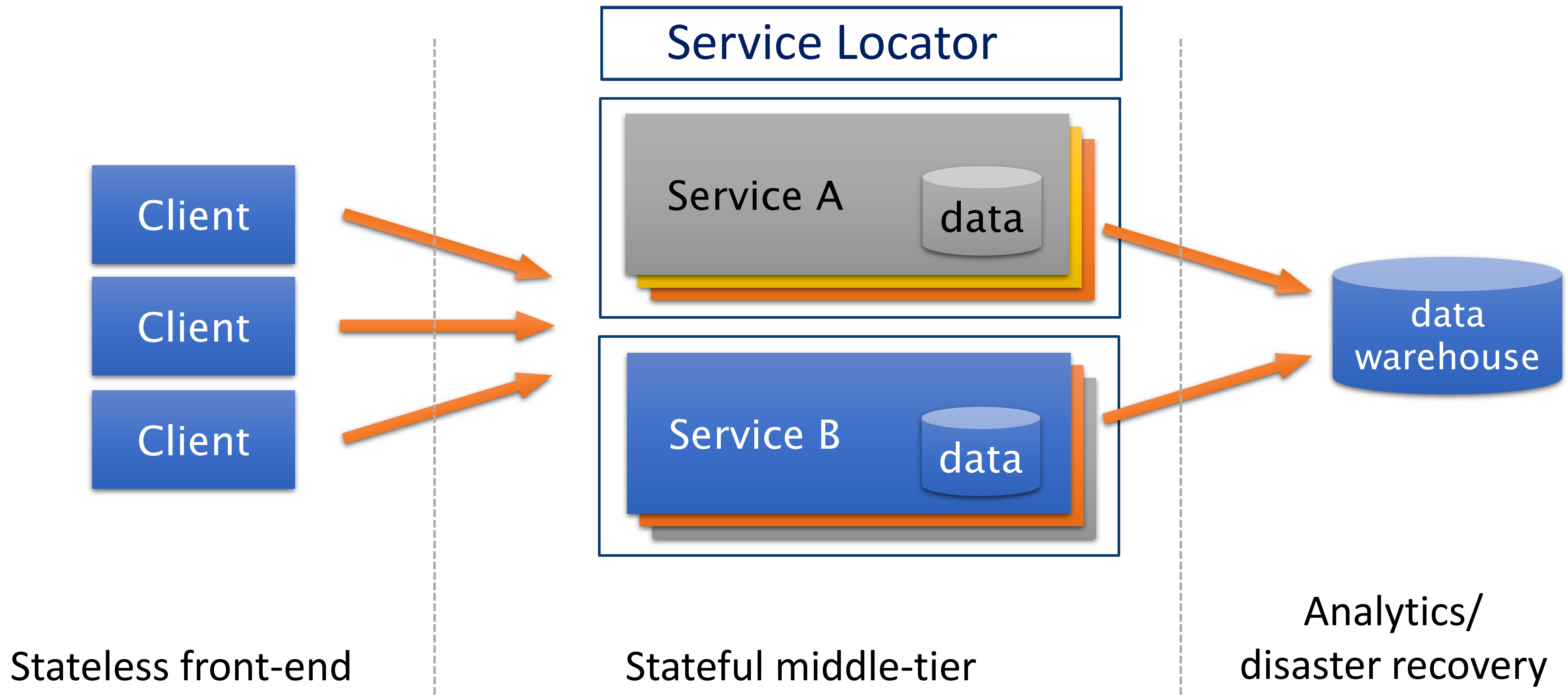
- Build easier Distributed Systems
- Automatic 3D scaling is possible

- Less skilled dev's can build more complex systems
- Save development and operation costs
- Run same code on-premises and in Cloud

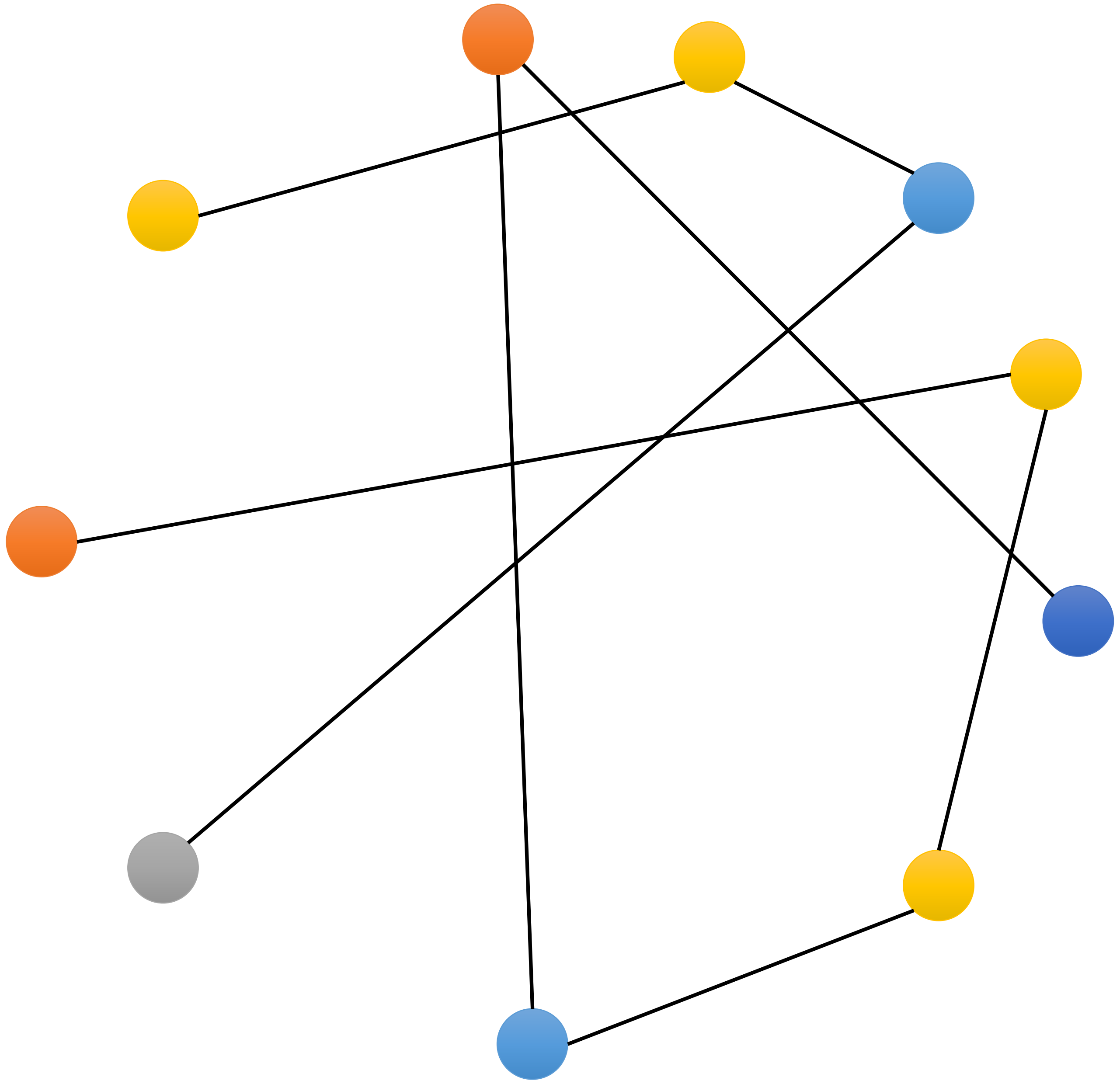
Micro Services



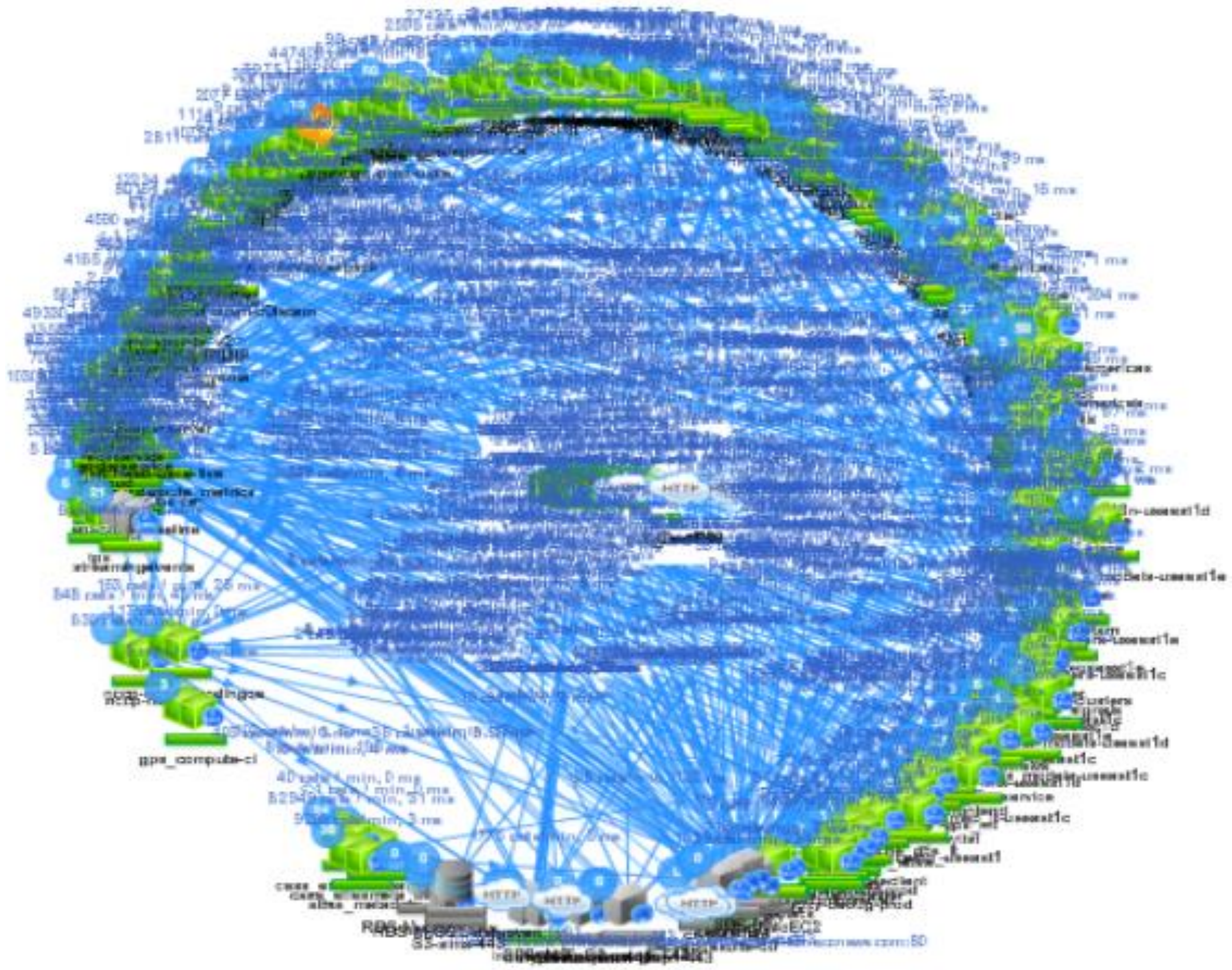
Micro Services



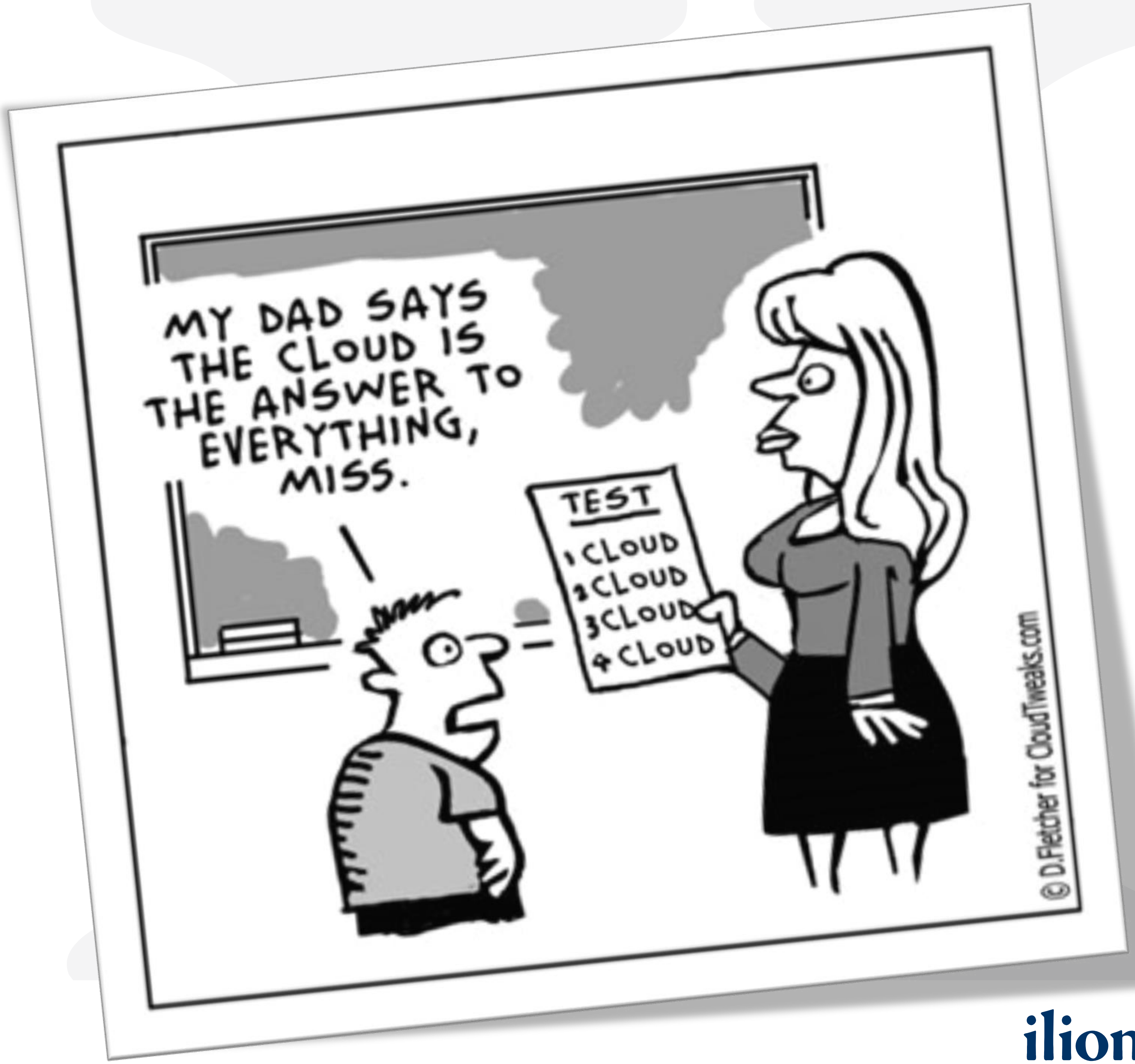
Micro Services



Micro Services bij Netflix

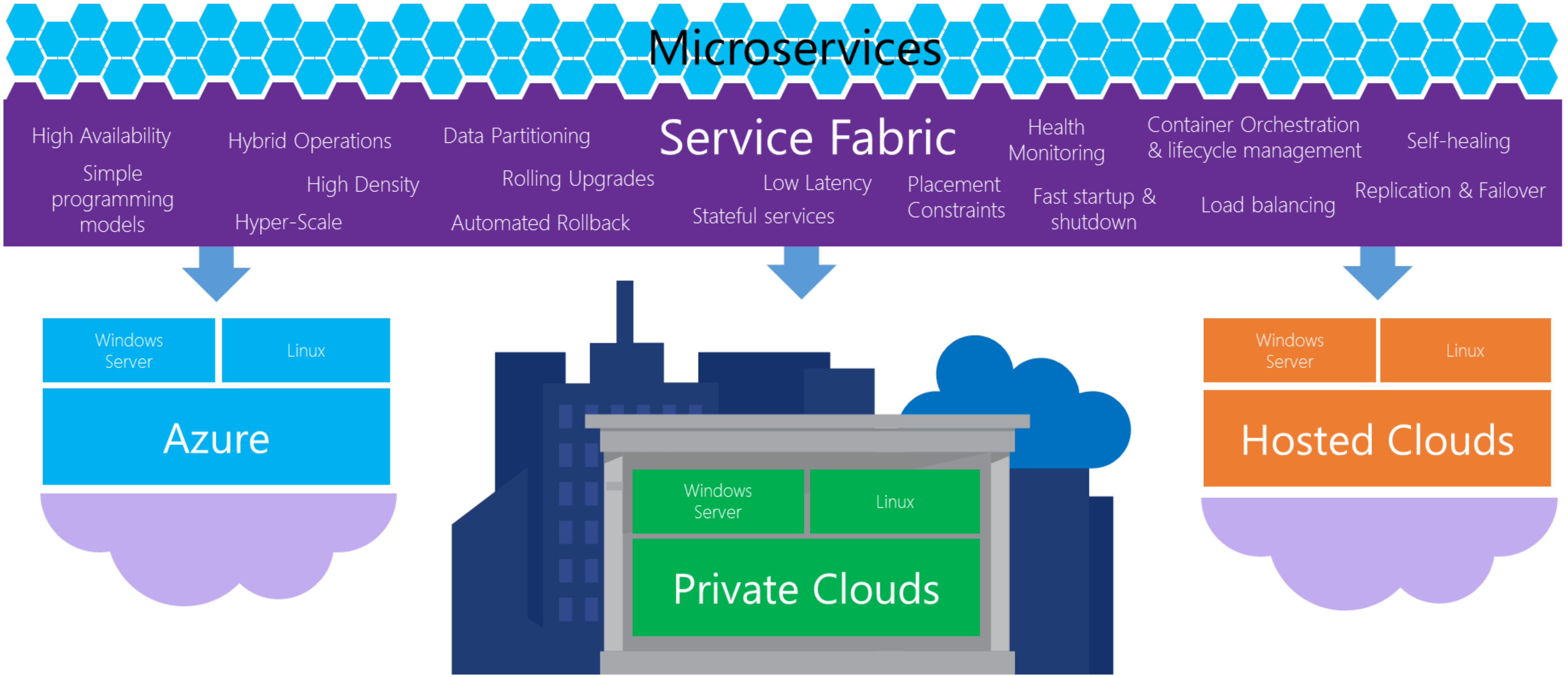


Azure Service Fabric


MY DAD SAYS THE CLOUD IS THE ANSWER TO EVERYTHING, MISS.
TEST
1 CLOUD
2 CLOUD
3 CLOUD
4 CLOUD
© D.Fletcher for CloudTweaks.com

ilionx
Your partner in digital business

Azure Service Fabric



Azure Services build with Service Fabric

Azure Core Infrastructure

thousands of machines

Azure Document DB

billions transactions /week

Intune

800k devices

Skype for Business

Hybrid Ops

Power BI

Azure SQL Database

1.4 million databases

Bing Cortana

500m evals/sec

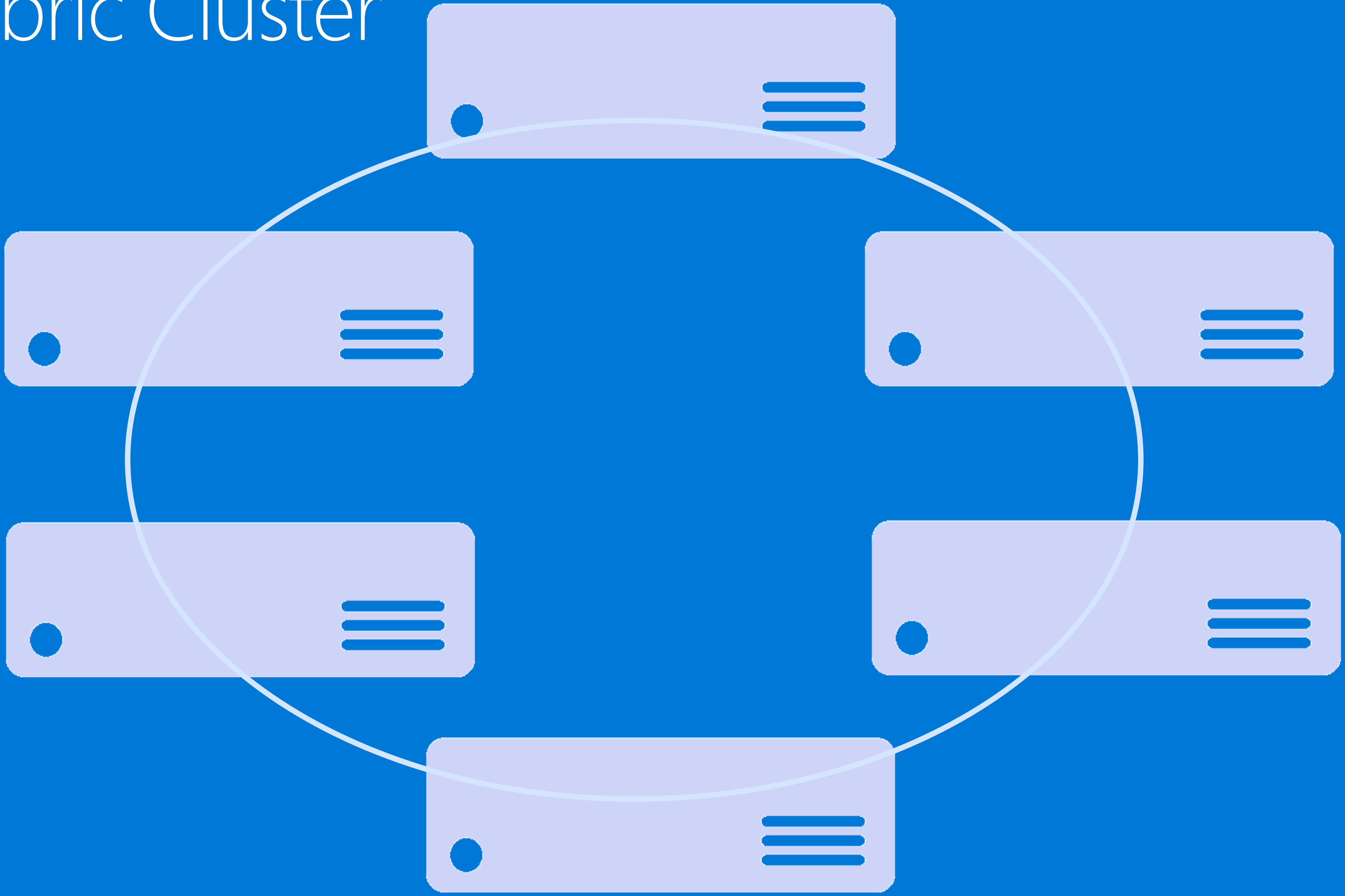
Event Hubs

20bn events/day

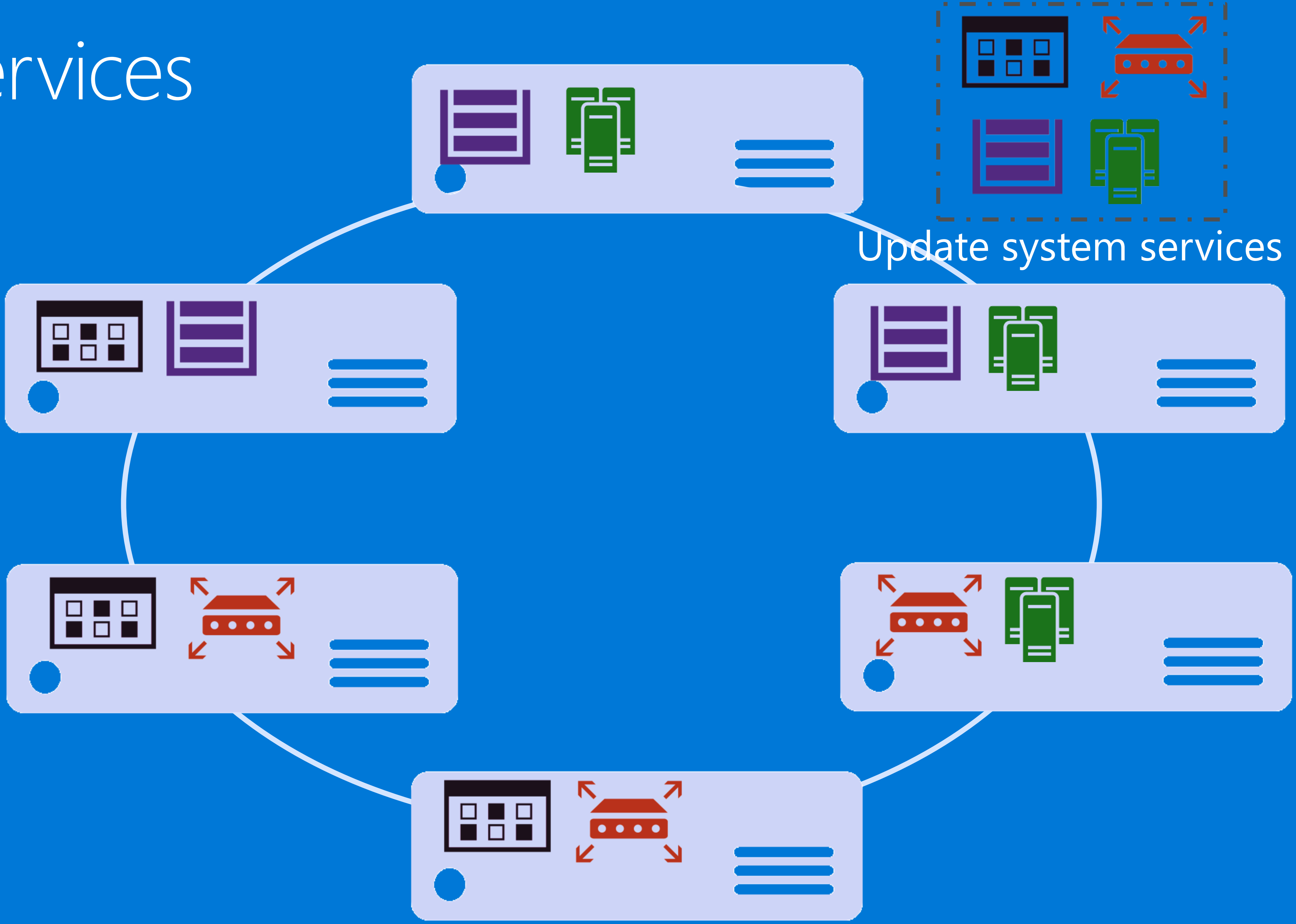
Service Fabric Cluster

A set of machines that Service Fabric stitches together to form a cluster

Clusters can scale to 1000s of machines



System Services



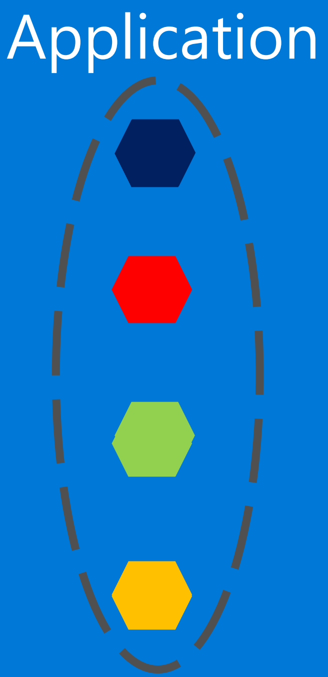
Failover manager

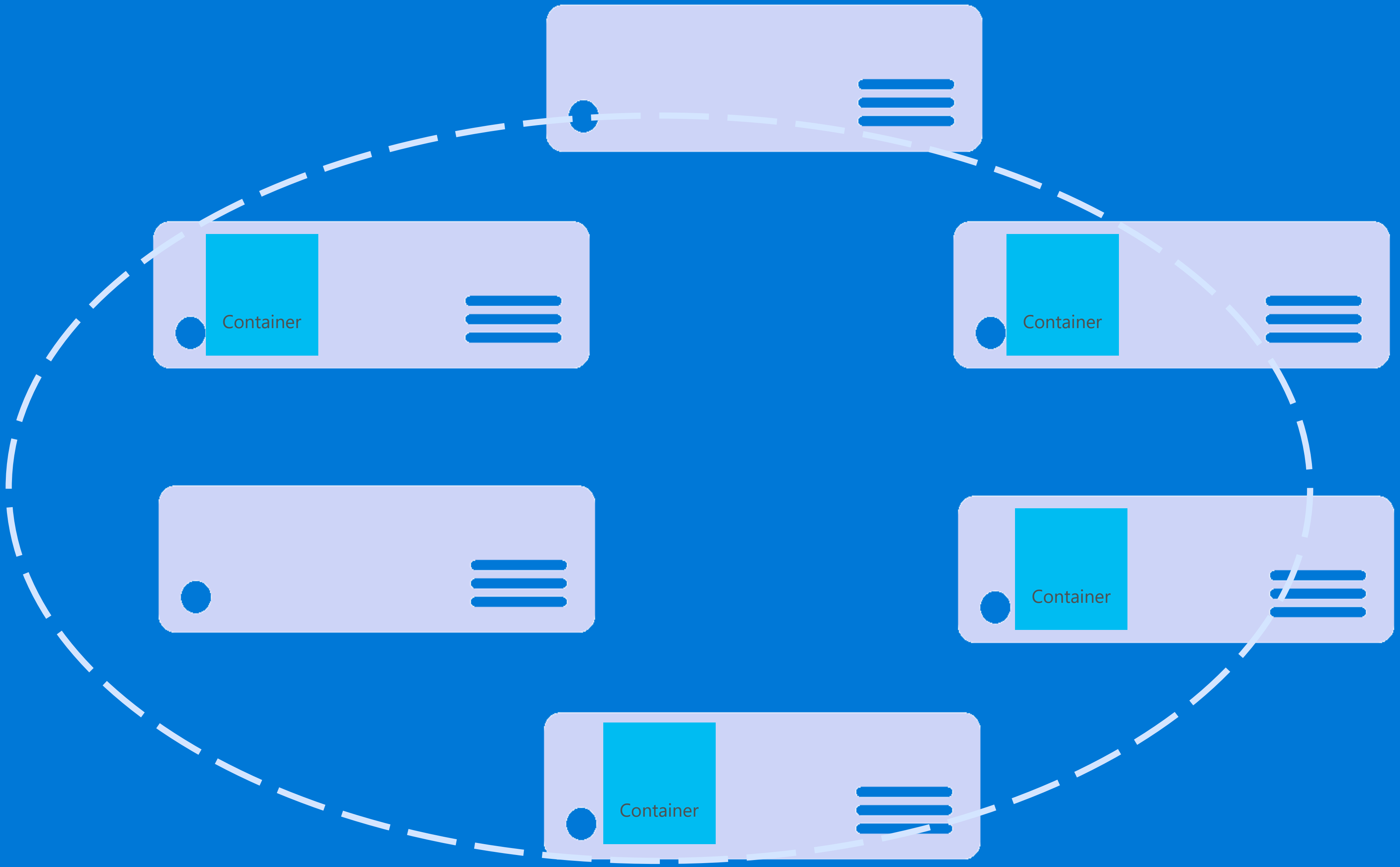
Cluster manager

Naming service

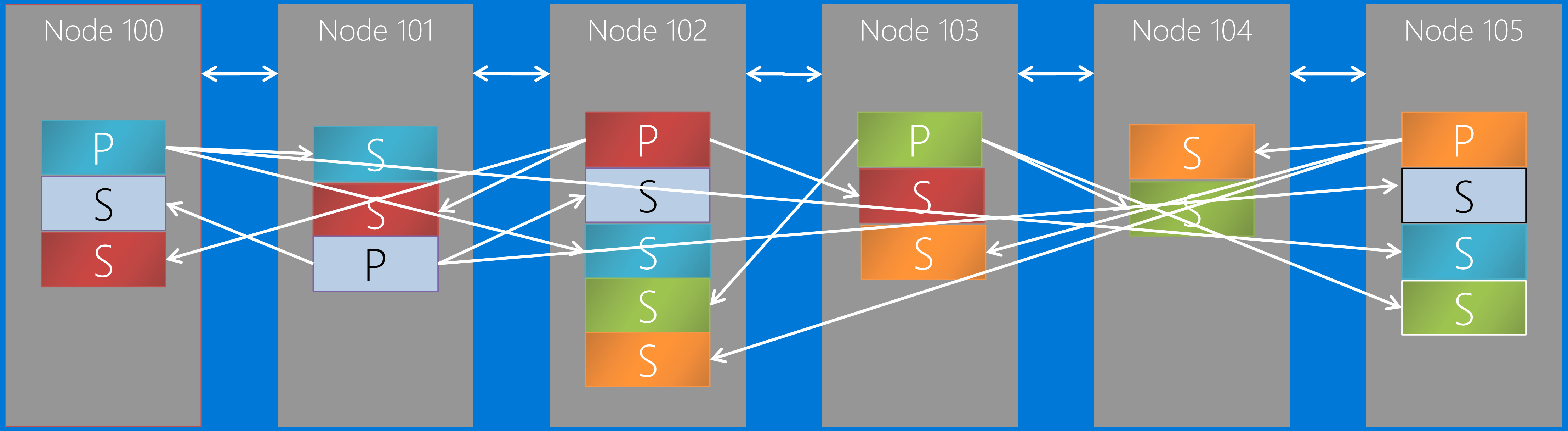
File store service

Application: A group of microservices

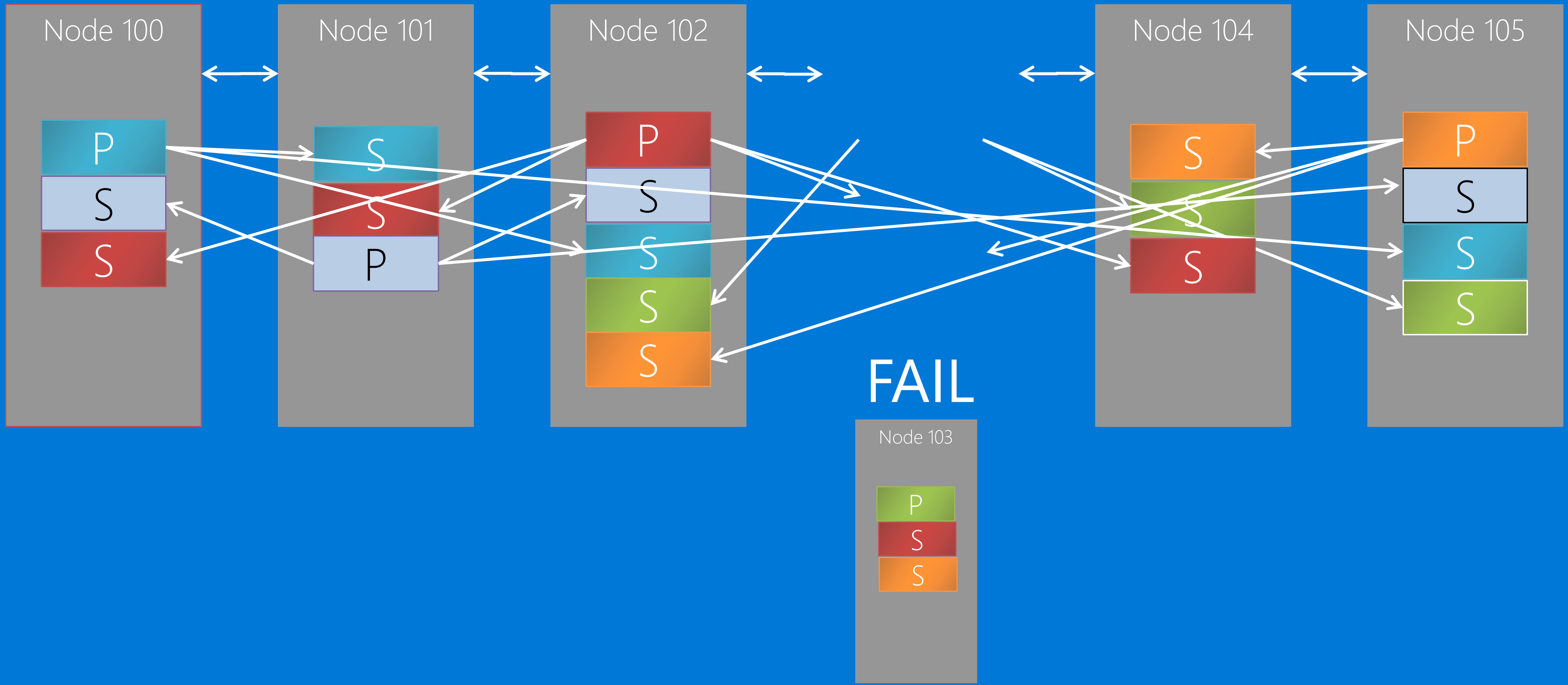




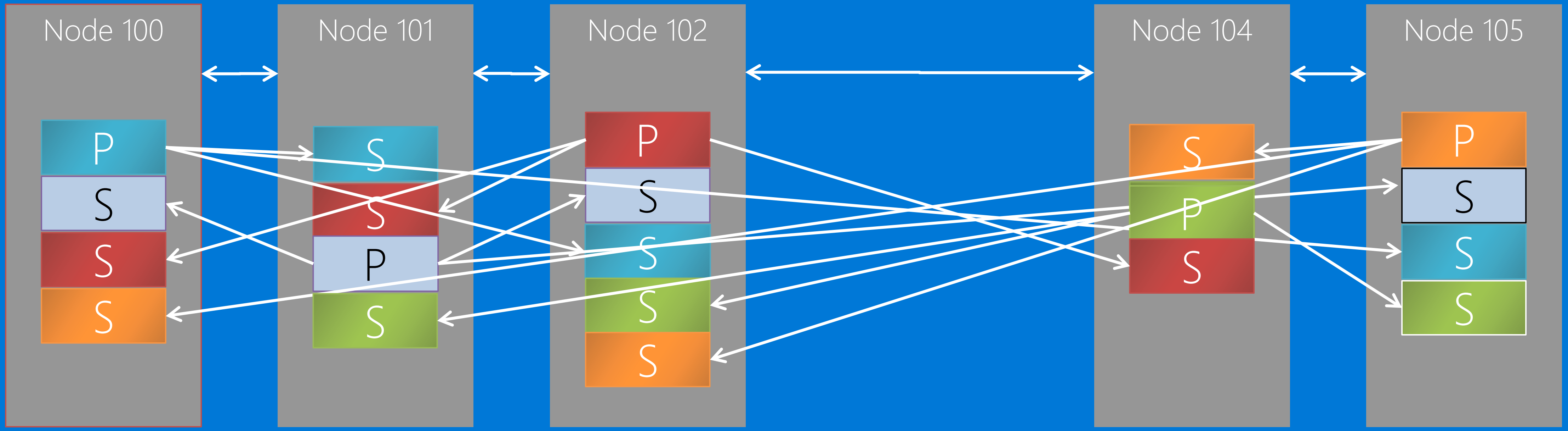
Service partitioning and failover



Service partitioning and failover



Service partitioning and failover



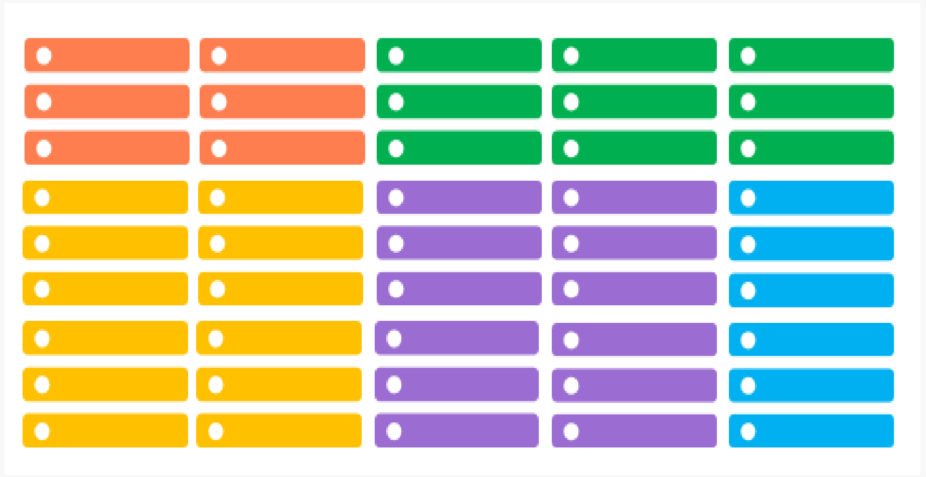
...normal operation...

What is a micro service in Service Fabric?

- Is [*logic + state*] that is independently versioned, deployed, and scaled
- Has a unique name that can be resolved
  - e.g. fabric:/myapplication/myservice
- Interacts with other micro services over well defined interfaces and protocols like REST
- Remains always logically consistent in the presence of failures
- Hosted inside a “container” (code + config)

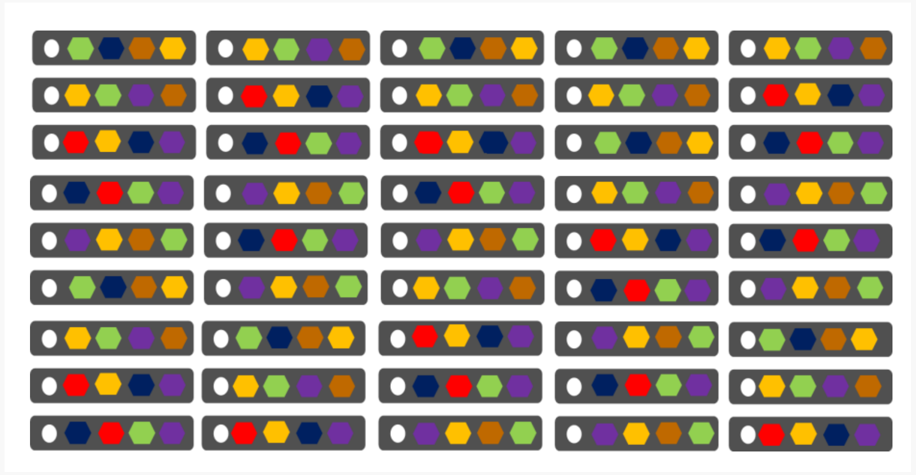
Cloud Services vs Service Fabric

Azure Cloud Services
(Web & Worker Roles)



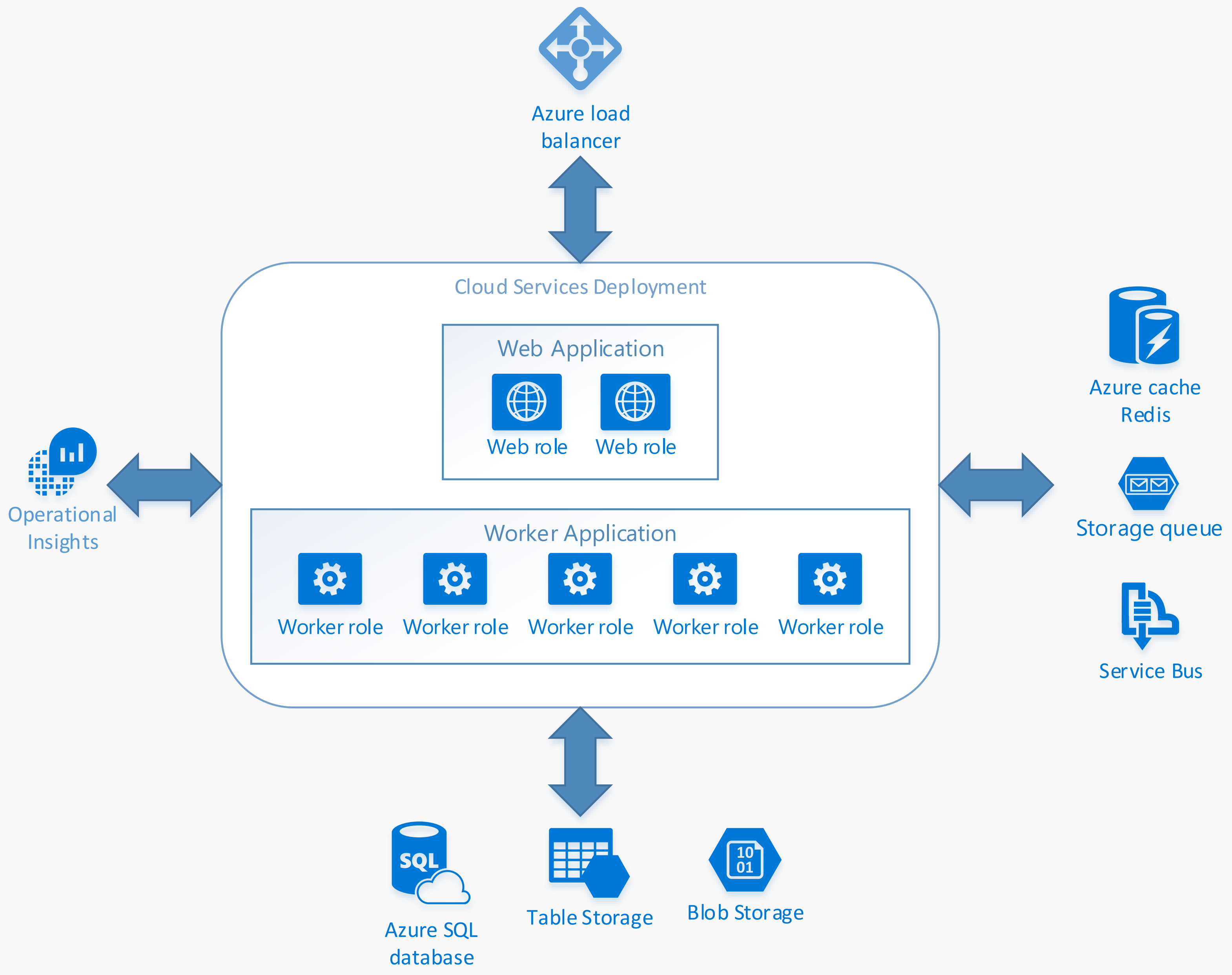
- 1 service instance per VM
- Slow deployment & upgrades
- Slow scaling of roles up/down
- Emulator for development

Azure Service Fabric
(Microservices)

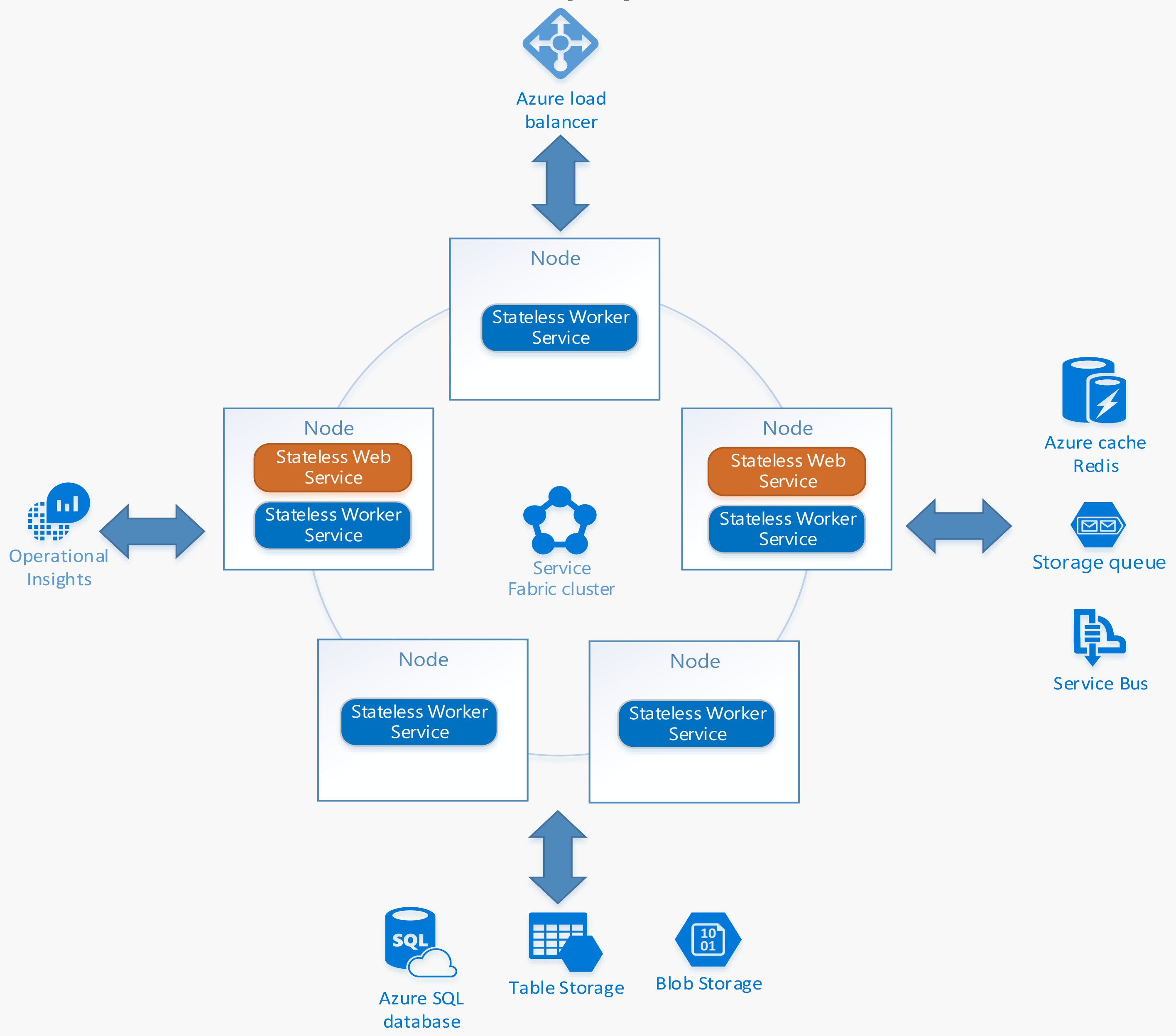


- Many microservices per machine/VM
- Fast deployment & upgrades
- Fast scaling of micro services up/down
- Single machine cluster for development

Cloud Service Application Design



Service Fabric Application Design

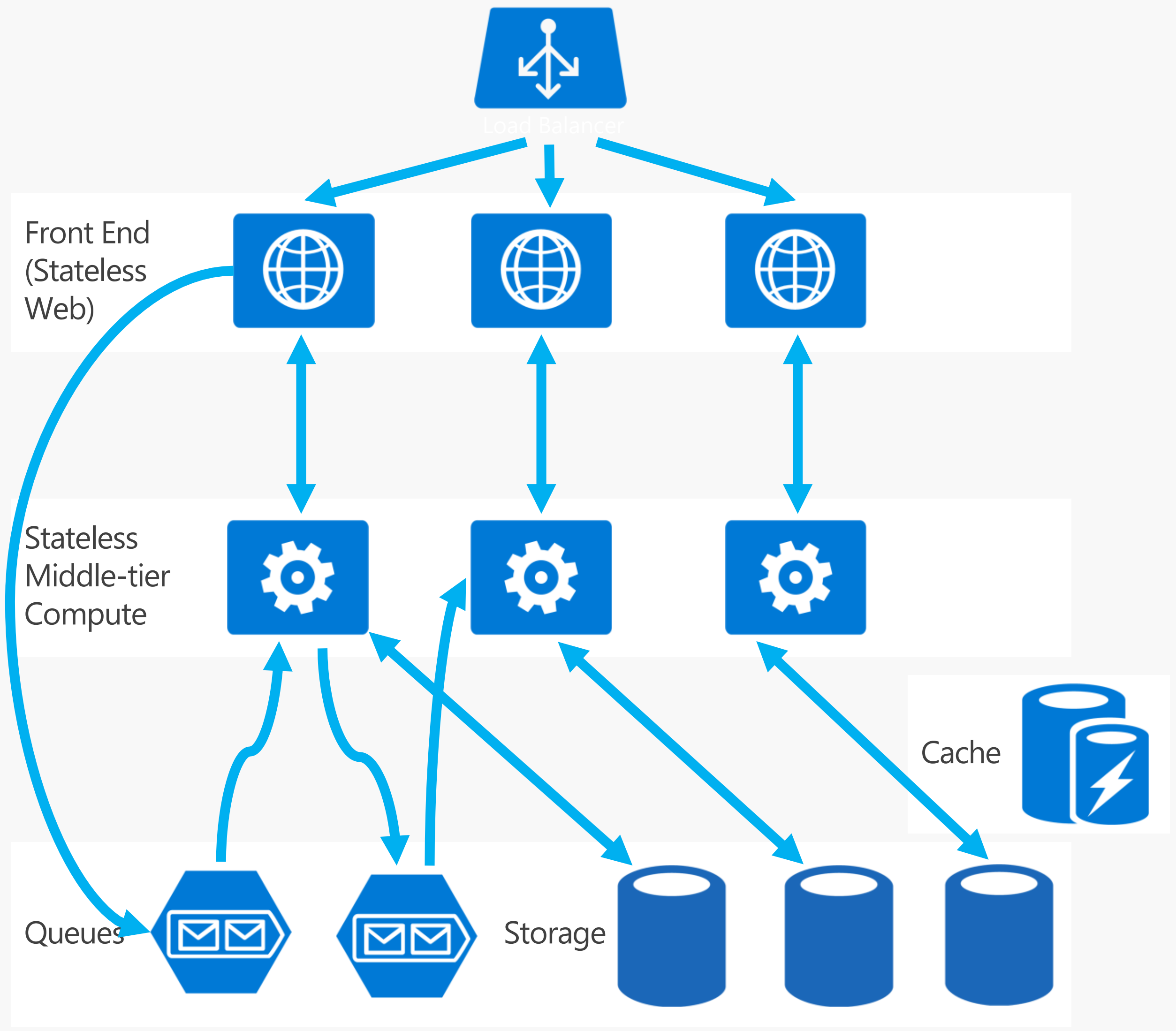


What can you build with Service Fabric?

- Stateless applications
  - A service that has state where the state is persisted to external storage, such as Azure databases or Azure storage
    - e.g. Existing web (ASP.NET) and worker role applications
- Stateful applications
  - Reliability of state through replication and local persistence
  - Reduces latency
  - Reduces the complexity and number of components in traditional three tier architecture
- Existing apps written with other frameworks
  - node.js, Java VMs, any EXE, any Docker container

Stateless Services Pattern

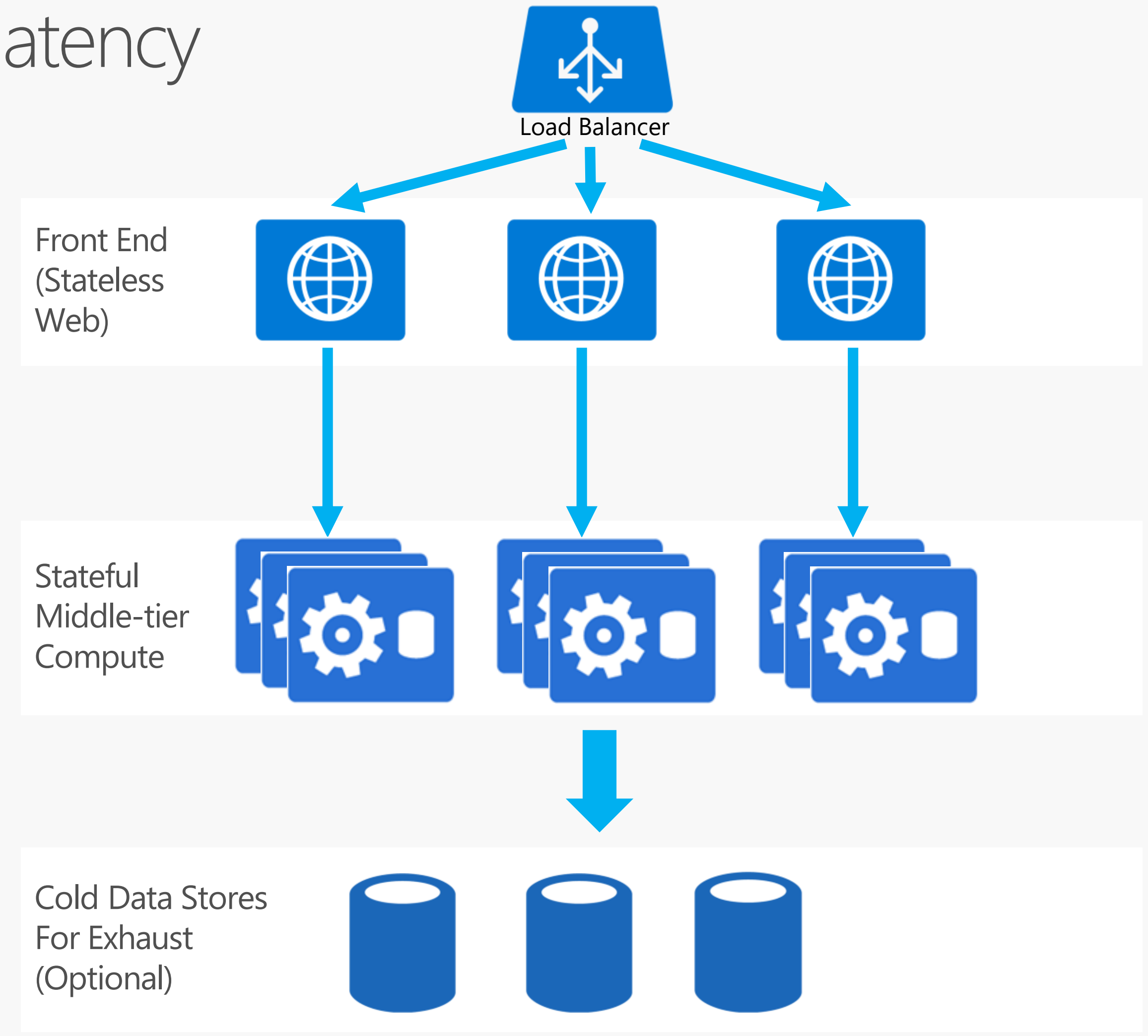
- Scale stateless services backed by partitioned storage
- Increase reliability and ordering with queues
- Reduce read latency with caches
- Manage your own transactions for state consistency
- More moving parts each managed differently



Stateful Services Pattern

Simplify design, reduce latency

- Application state resides in the compute tier
- Low latency reads and writes
- Partitions are first class at the service layer for scale-out
- Built in transactions
- Fewer moving parts
- External stores for exhaust and offline analytics


Load Balancer
Front End (Stateless Web)
Stateful Middle-tier Compute
Cold Data Stores For Exhaust (Optional)

Service Fabric Programming Models

- Reliable Actors API
  - Stateless
  - Stateful

- Reliable Services API
  - Stateless
  - Stateful

- Supported programming frameworks:

Microsoft.NET (full framework)	- runs only on Windows
Microsoft.NET Core	- runs on Windows and Linux
Java	- runs only on Linux

ilionx
Your partner in digital business

Actor programming model

- Introduced in 1973
- An actor is the fundamental unit of computation
  - Does some processing
  - Holds state
  - Communicates with other actors
- Similar to objects in Object Oriented programming

Reliable Services API

- Write services that are reliable, available, scalable and provide consistency
- Use Reliable Collections to persist state
- Manage the concurrency and granularity of state changes using transactions
- Communicate using tech of your choice (REST, SOAP, …)

Contact	:	Arjen Steinhauer asteinhauer@ilionx.com https://www.linkedin.com/in/asteinhauer/
Documentation	:	https://docs.microsoft.com/en-us/azure/service-fabric/
Demo code on GitHub	:	https://github.com/arjensteinhauer/servicefabric-demo

ilionx
Hondiuslaan 46
3528 AB Utrecht
T (088) 05 90 500
E info@ilionx.com

www.ilionx.com