System Scalability

- Scalability
  - Vertical & Horizontal Scaling
- Horizontal Scalability
  - Replication
  - Services
  - Caching
  - Asynchronous Process
  - Partitioning
- Load Balancing
  - Load Balancers
  - Service Discovery
  - DNS & Geo Load Balancing
- Micro-Services
  - Architecture
  - Transactions
  - SAGA Pattern
  - NoSQL

Performance Vs Scalability

FIXED LOAD

- Performance
  - Low Latency
  - High Throughput
    - Concurrency
      - Single Machine – Multi-Threading
      - Multi Machine – Multi Threading + Multi-Processing = Distributed Processing
    - Capacity

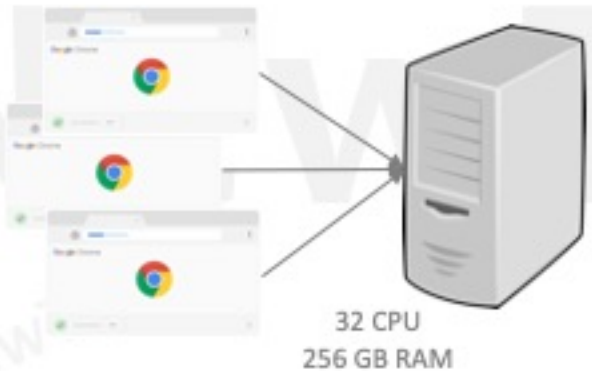
VARIABLE LOAD

- Scalability
  - High Throughput
    - Ability of a system to increase its throughput by adding more hardware capacity
  - Both ways – UP and DOWN

Vertical & Horizontal Scalability

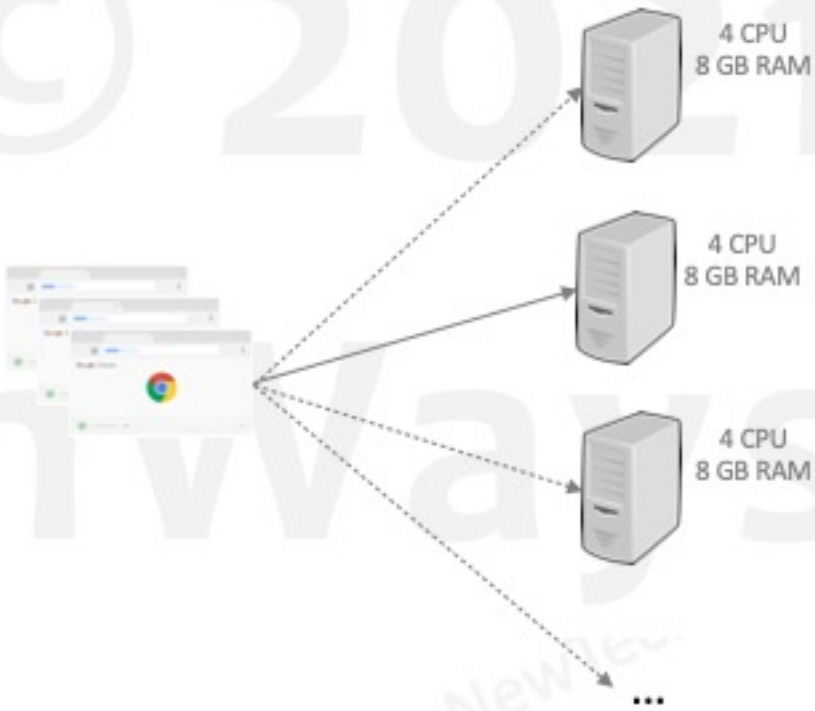
Vertical

- Easier to achieve
- Limited scalability



Horizontal

- Hard to achieve
- Unlimited scalability



NewTechWays

Reverse Proxy

- Client needs to know only about the address of the Reverse Proxy
- Reverse Proxy can also act as a load balancer

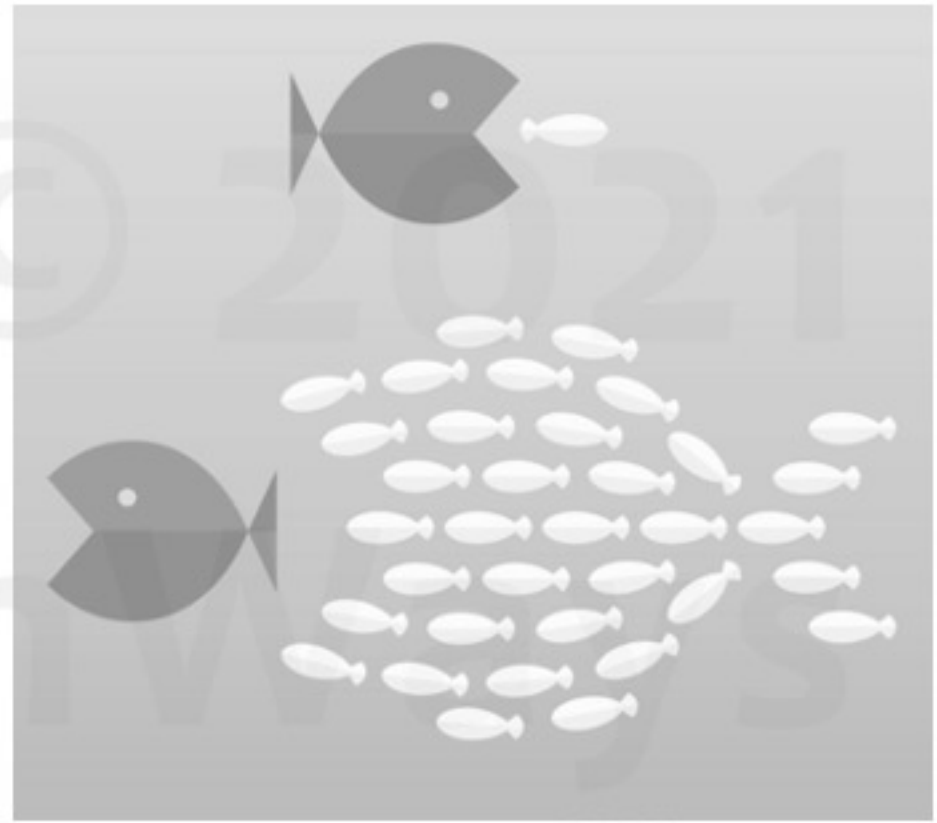
NewTechWays

System For Scalability Discussion

NewTechWays

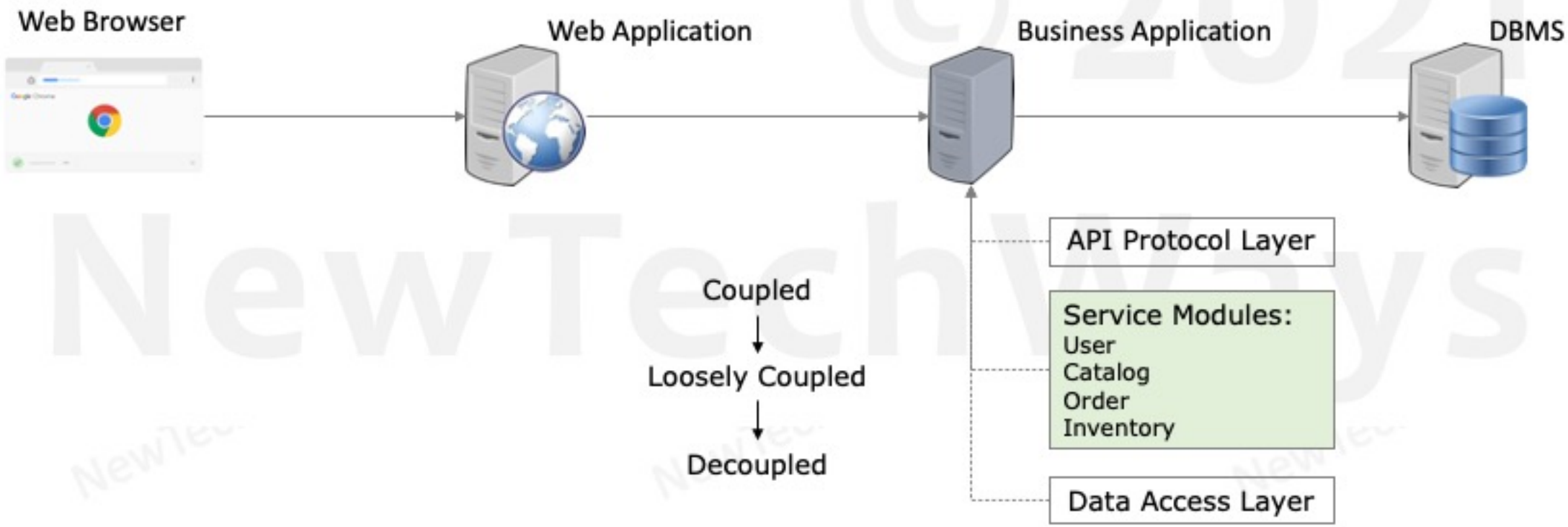
Scalability Principle

- Decentralization – *Monolith is an anti-pattern for Scalability*
  - More workers – Instances, Threads
  - Specialized workers – Services
- Independence
  - Multiple workers are as good as a single worker if they can't work independently
    - They must work concurrently to maximum extent
  - Independence is impeded by
    - Shared resources
    - Shared mutable data



Modularity

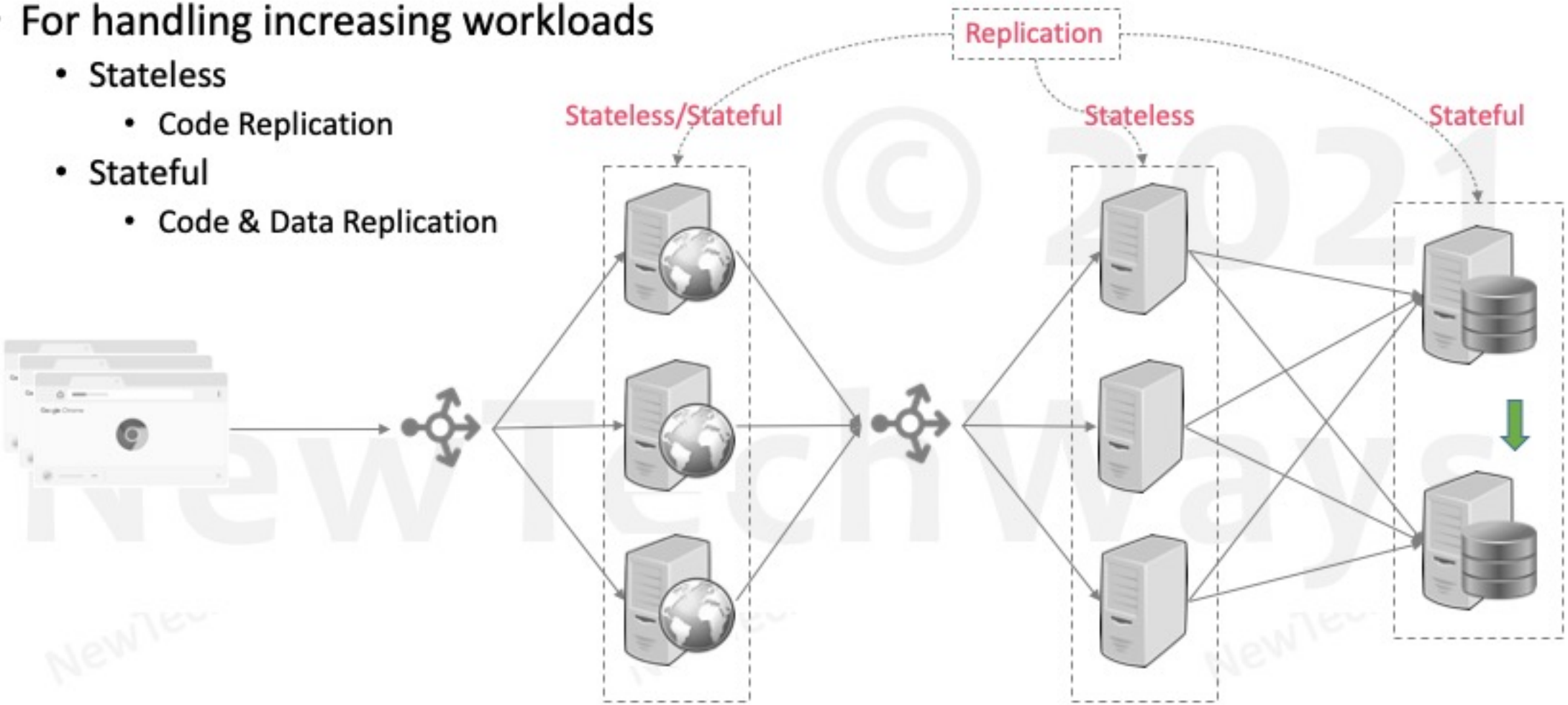
- Scalable architecture starts with modularity
  - Provides the foundation for breaking an application into more specialized functions/services



NewTechWays

Replication

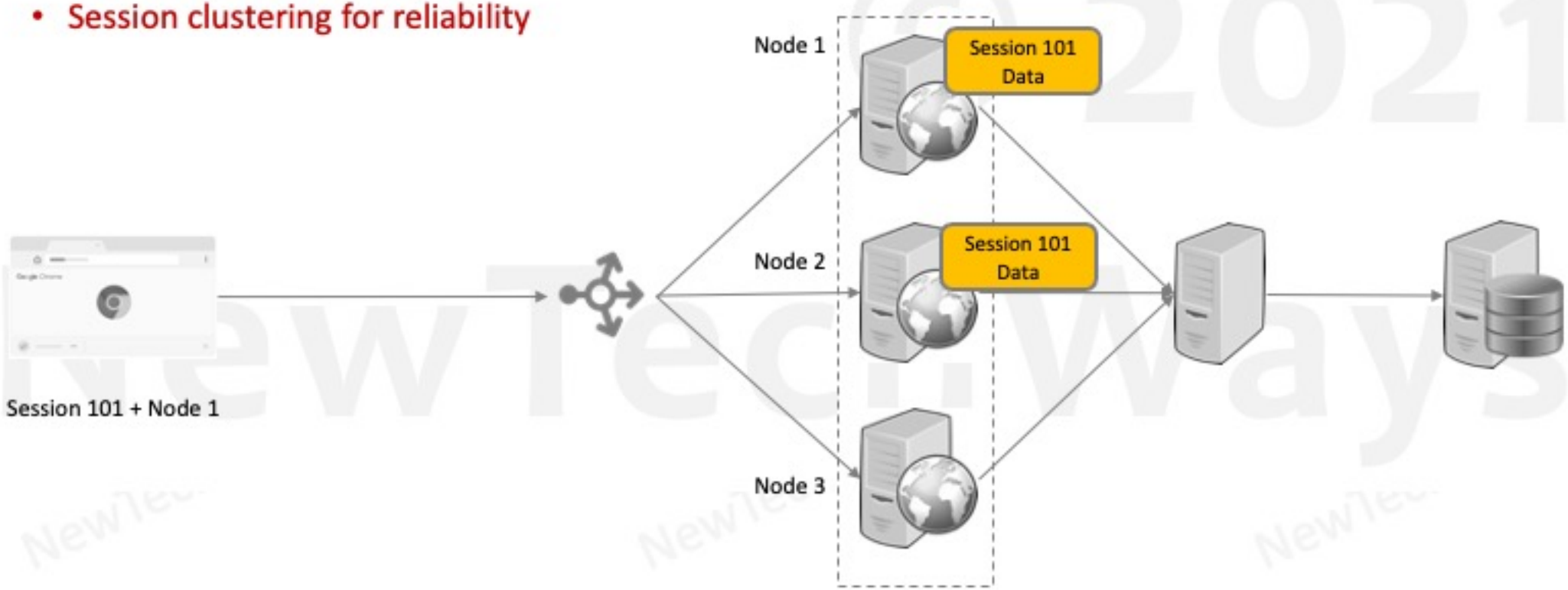
- For handling increasing workloads
  - Stateless
    - Code Replication
  - Stateful
    - Code & Data Replication



NewTechWays

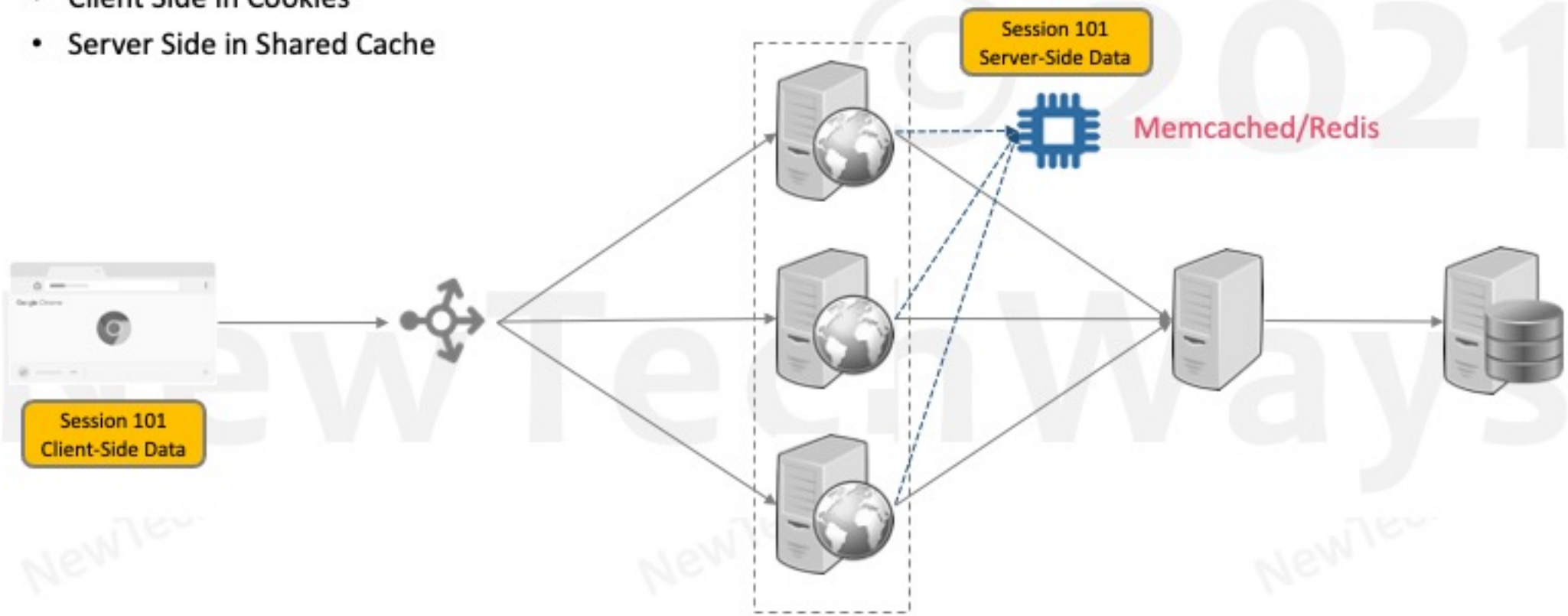
Web Stateful Replication

- When low latency is required
  - Sticky sessions/Session affinity
  - Sessions occupy memory
  - Session clustering for reliability



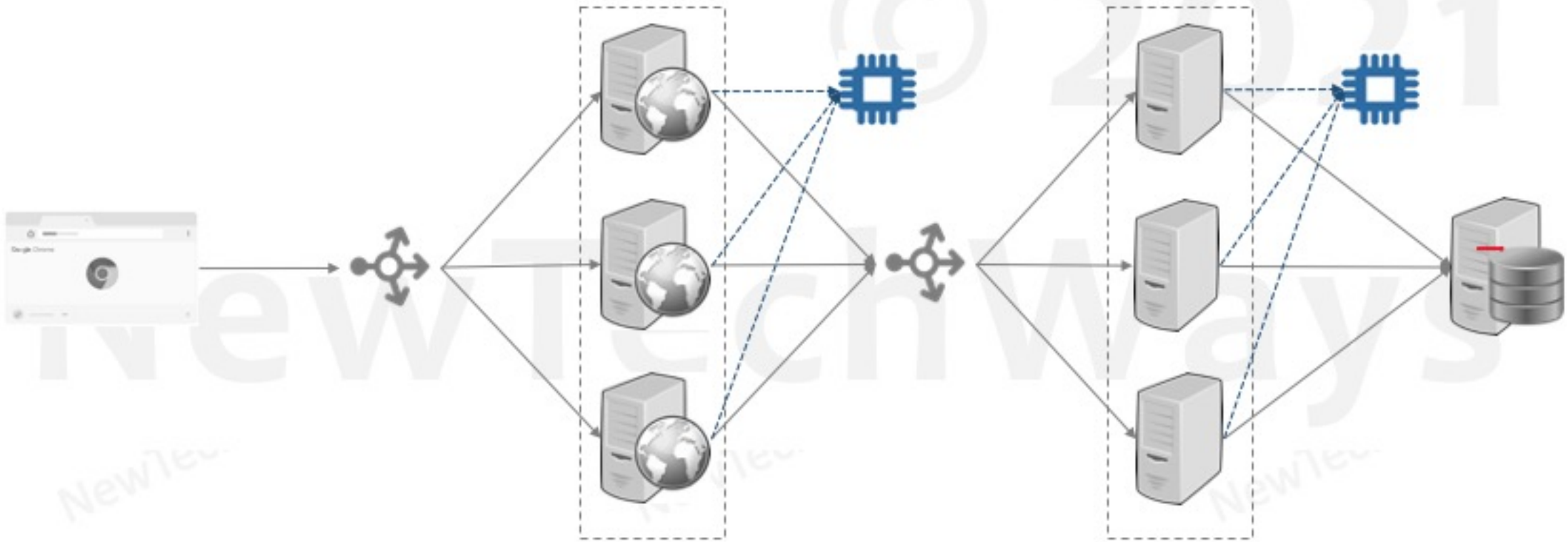
Web Stateless Replication

- For higher scalability at the expense of higher latency
- Session data can be stored on
  - Client Side in Cookies
  - Server Side in Shared Cache



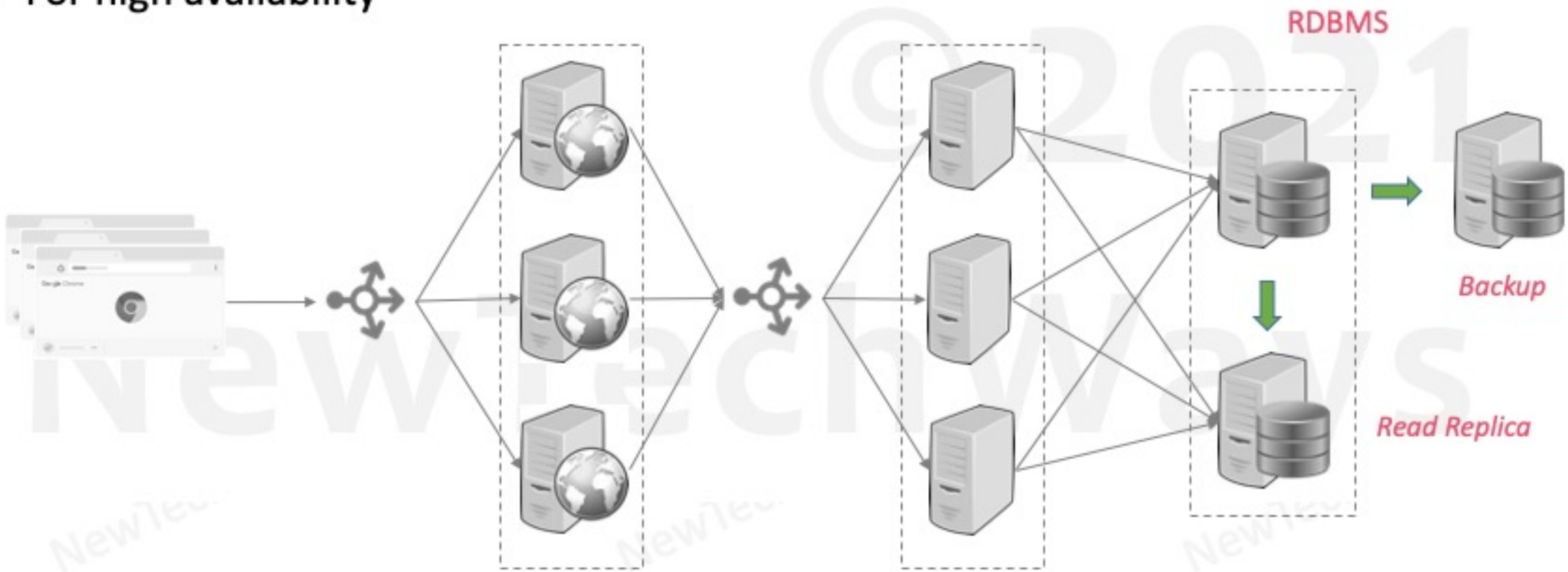
Service Replication

- Stateless replication – same as web stateless



Database Replication

- For higher read scalability
- For high availability



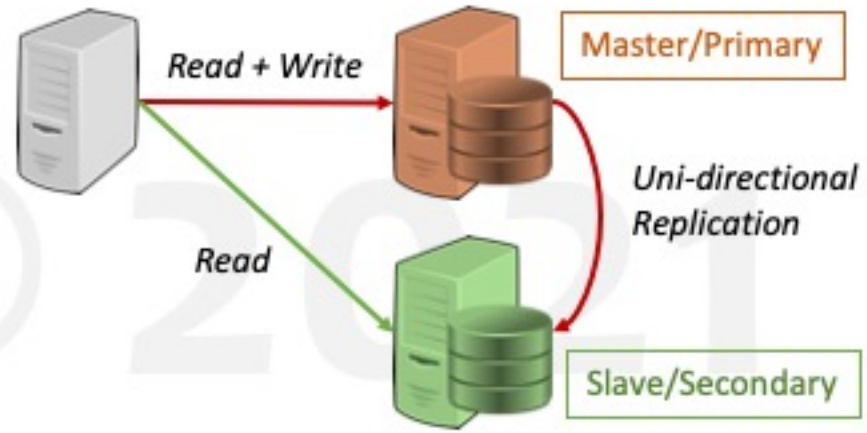
Database Replication

- Master-Slave (Primary-Secondary)
  - Asynchronous
    - Low latency writes
    - Eventually Consistent
    - Data Loss
  - Synchronous
    - Consistent
    - High latency writes
    - Low write availability

READ REPLICA

BACKUP

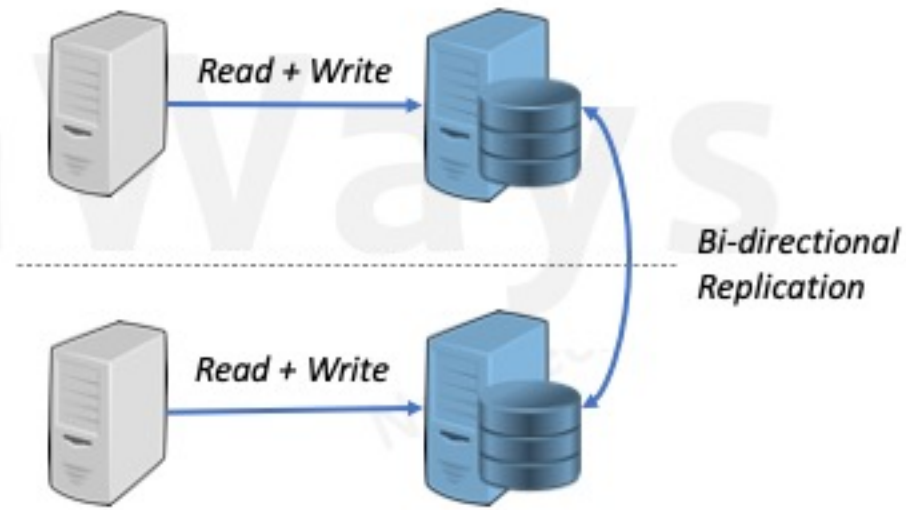
- High Read Scalability
- High Read Availability
- No Write Conflicts



- Master-Master (No-Master/Peer-To-Peer)
  - Asynchronous
    - Write conflicts
    - High availability

MULTI GEOGRAPHY

- High Read Scalability
- High Read Write Availability
- Transaction ordering issues



NewTechWays

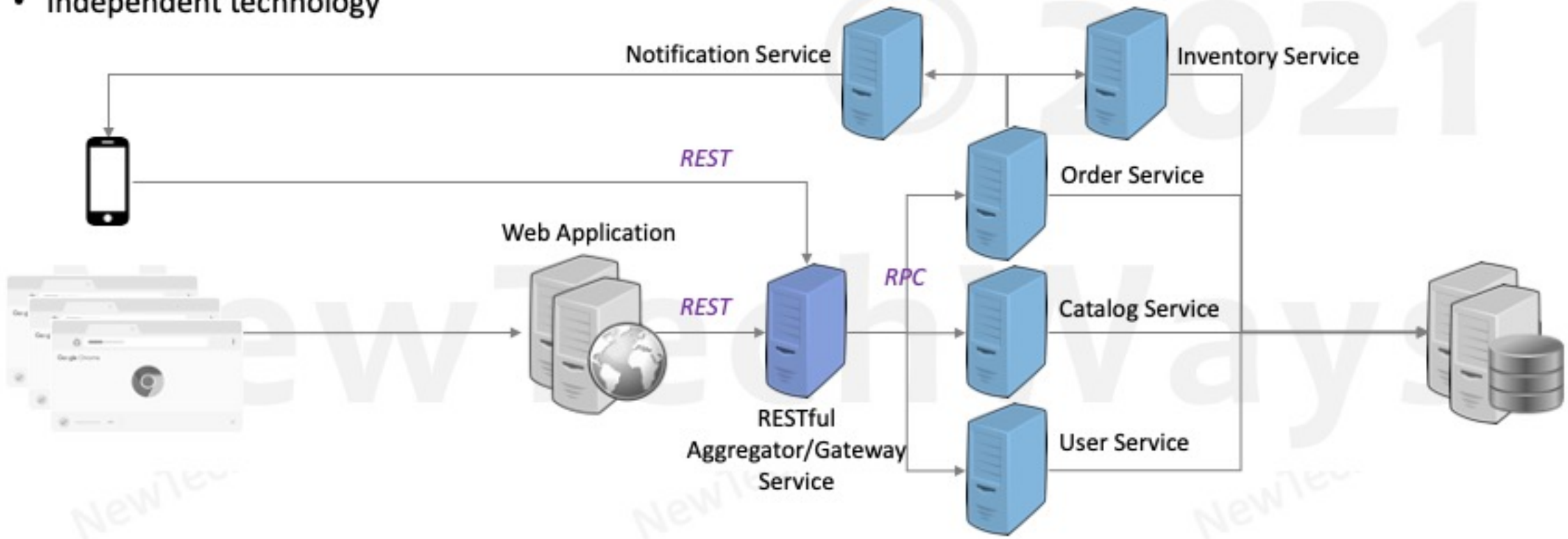
Need For Specialized Services

Service Modules:

- User
- Catalog
- Order
- Inventory
- Notification

Specialized Services (SOAP/REST-Services)

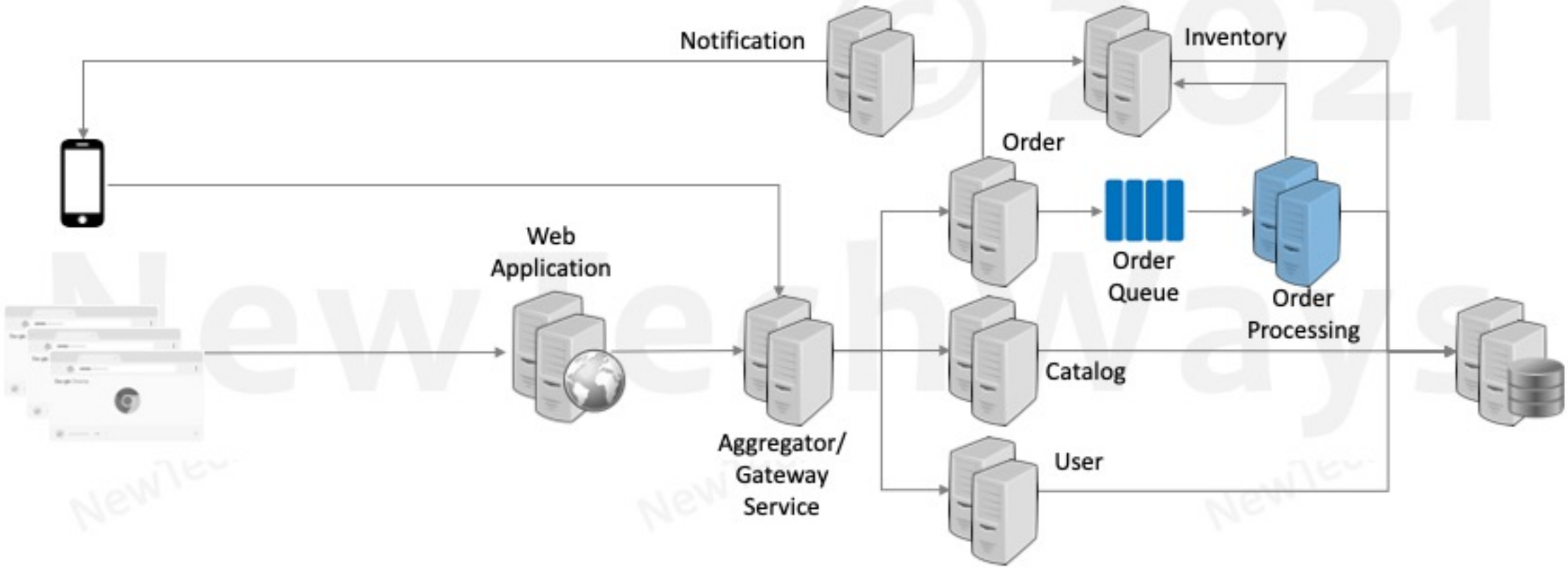
- Partially independent development and deployment
- Independent scalability
- Independent technology



NewTechWays

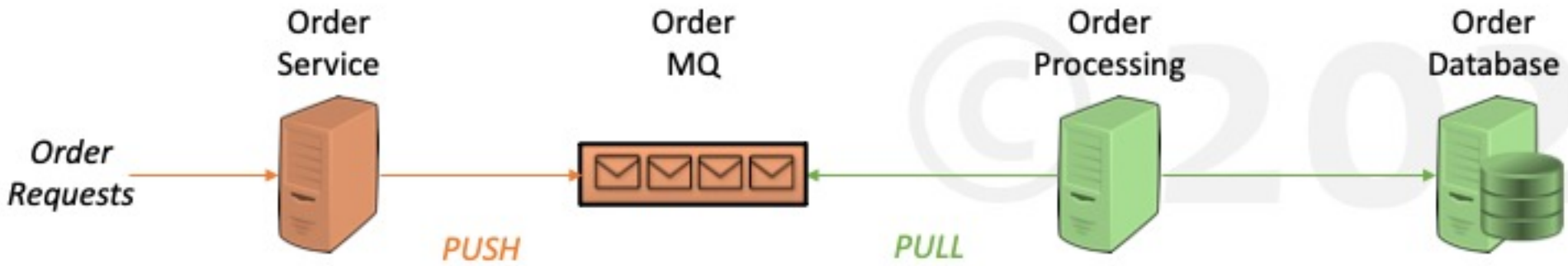
Asynchronous Services

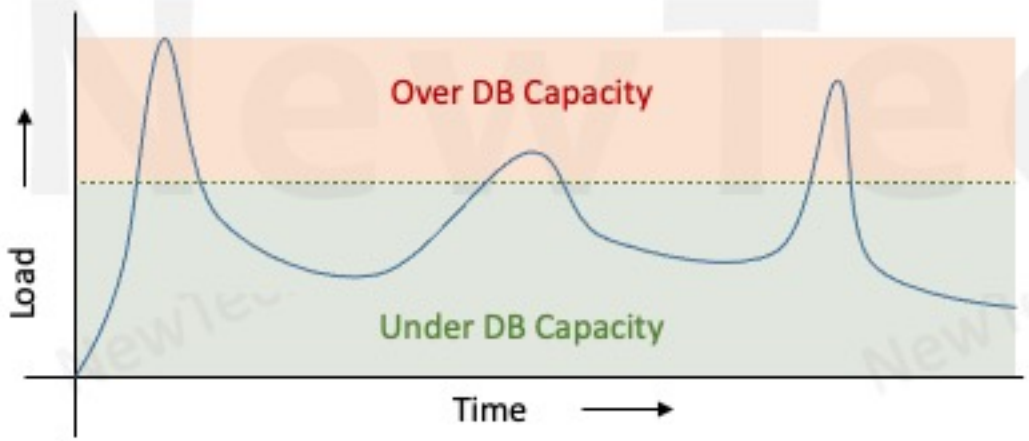
- Async services effectively reduces write load from a database

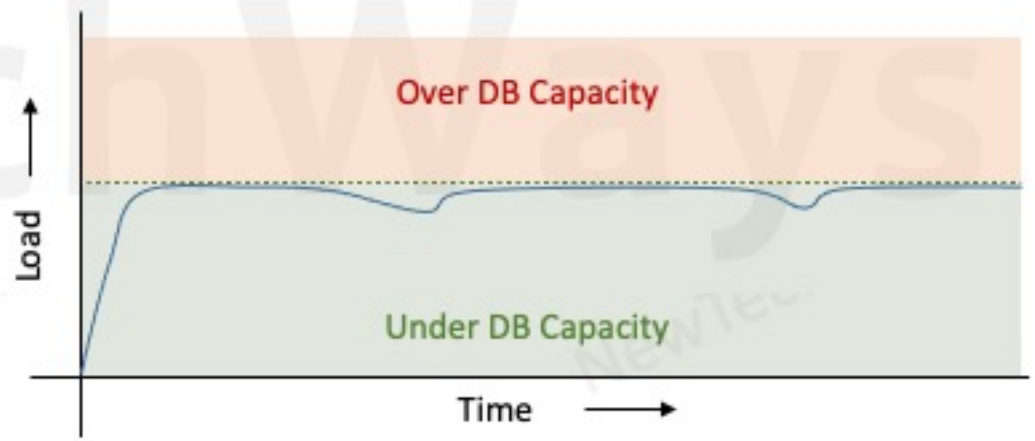


Asynchronous Processing & Scalability

- Async services require infrastructure for average load as opposed to peak load

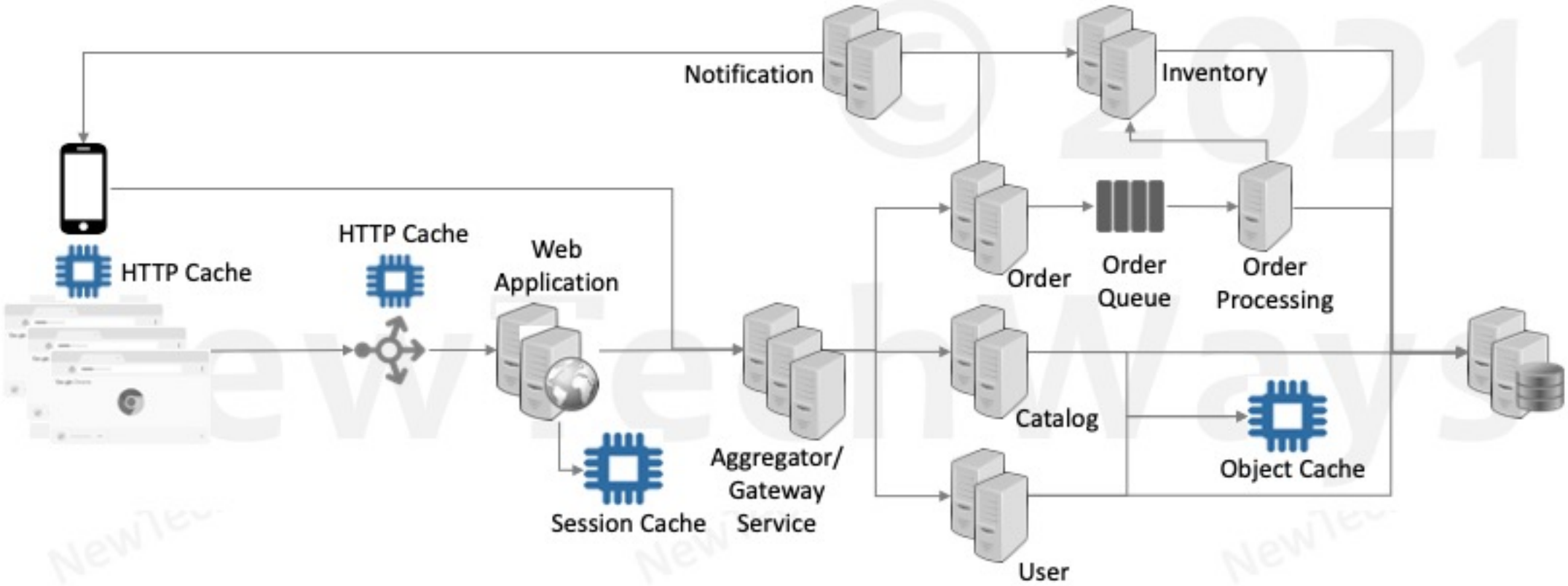






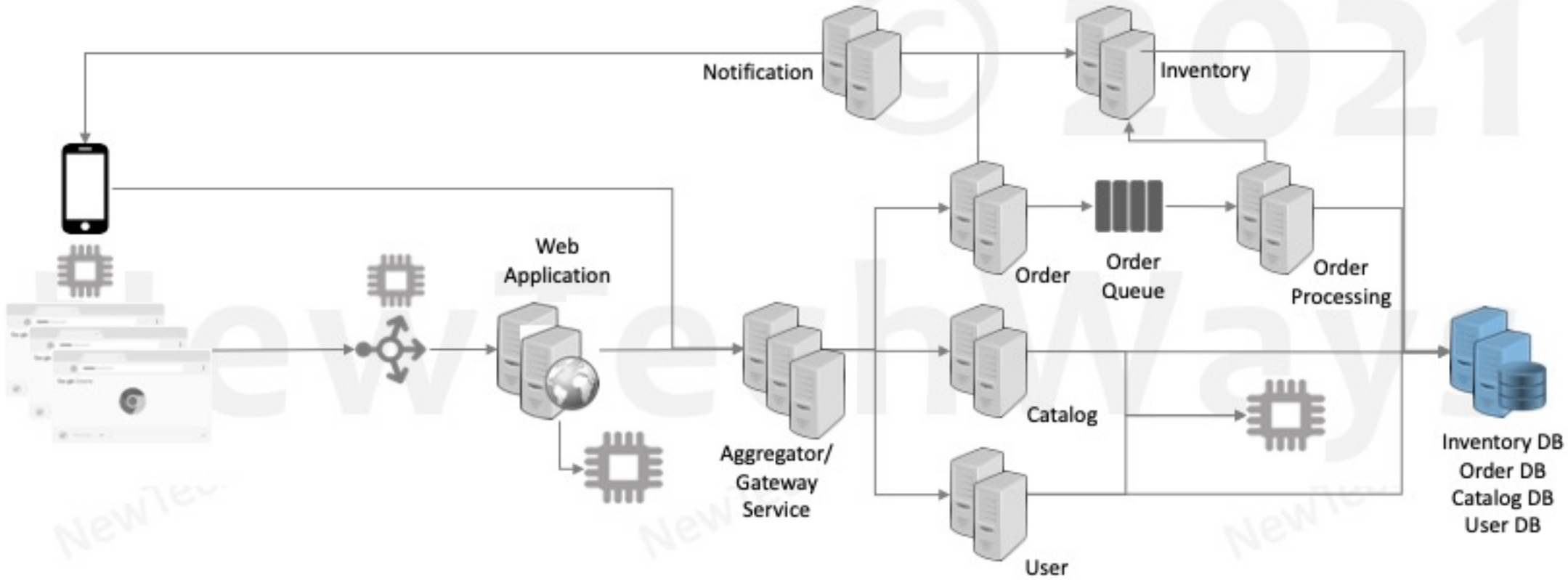
Caching

- Caching reduces latency and reduces overall read load



Vertical Partitioning (Micro-Services)

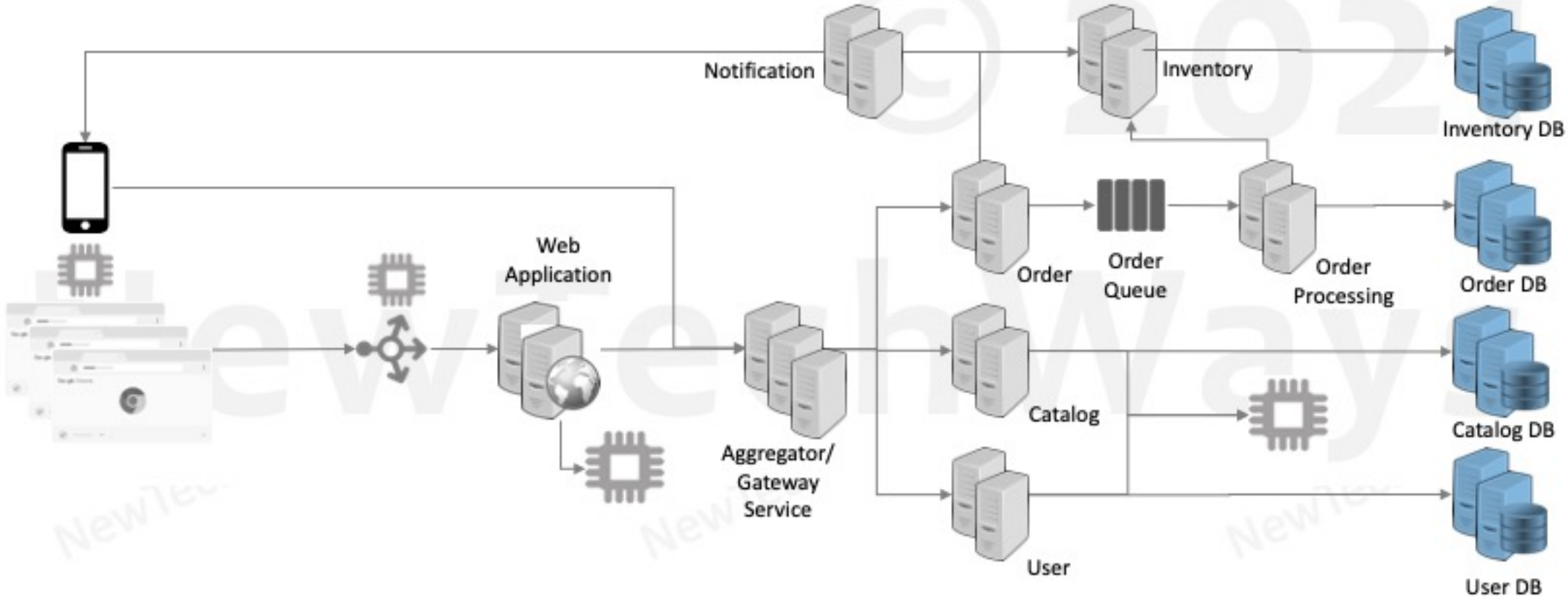
- Micro-Services completely decouples services and databases for higher scalability



NewTechWays

Vertical Partitioning (Micro-Services)

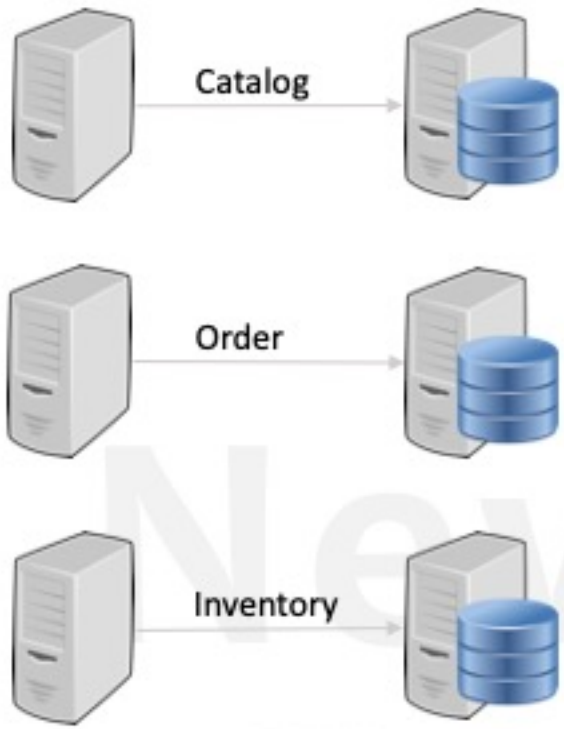
- Micro-Services completely decouples services and databases for higher scalability
  - Can no longer do inter service ACID transactions and need to deal with eventual consistency



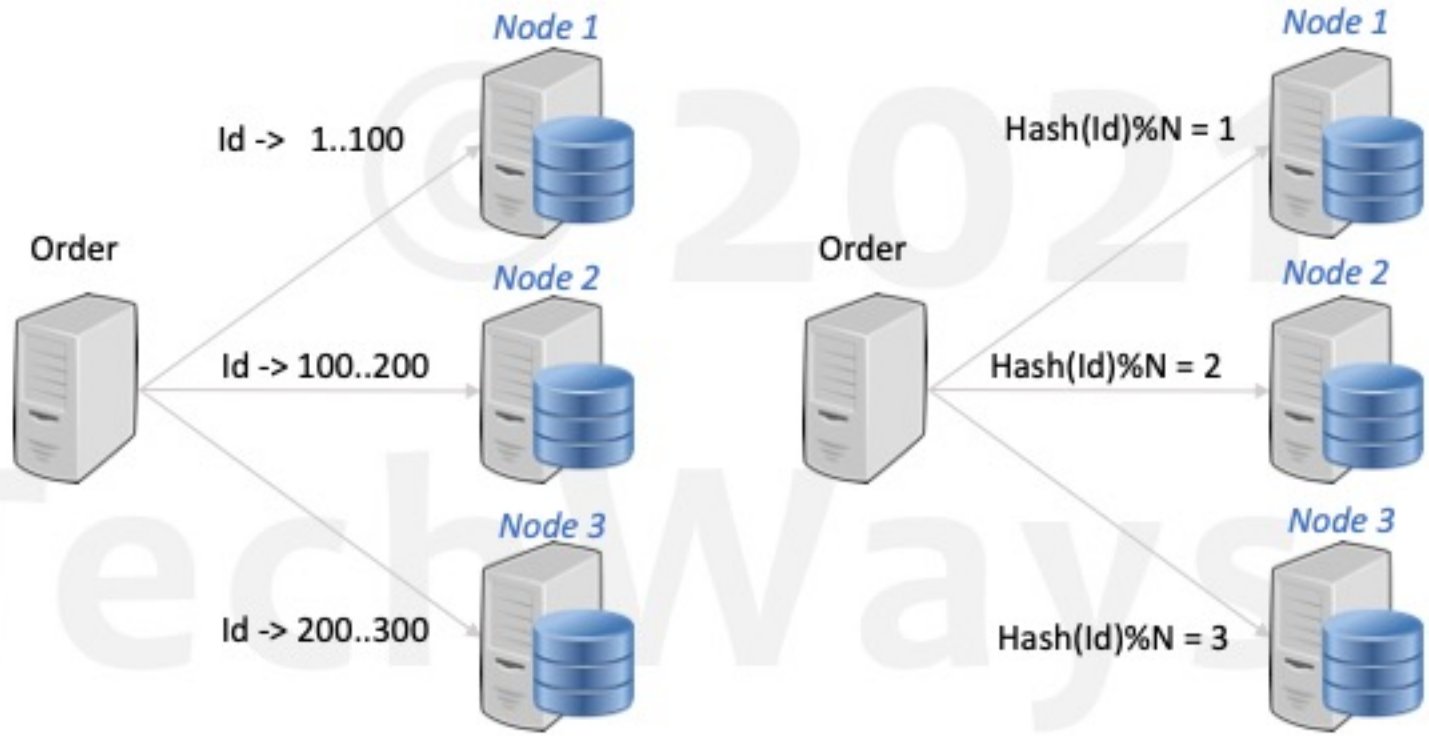
NewTechWays

Database Partitioning

Vertical Partitioning Of System



Horizontal Partitioning Of Database



Range Partitioning

Hash Partitioning

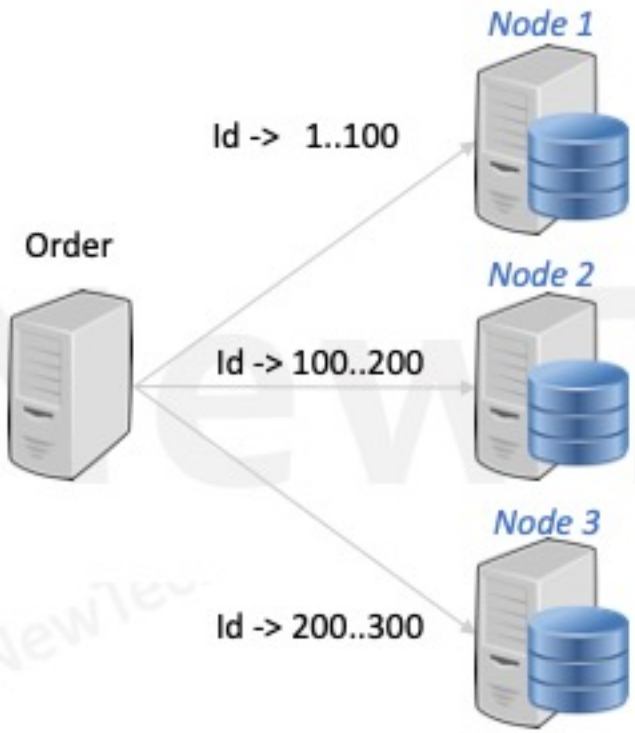
NewTechWays

Database Partitioning Selection

Range Partitioning

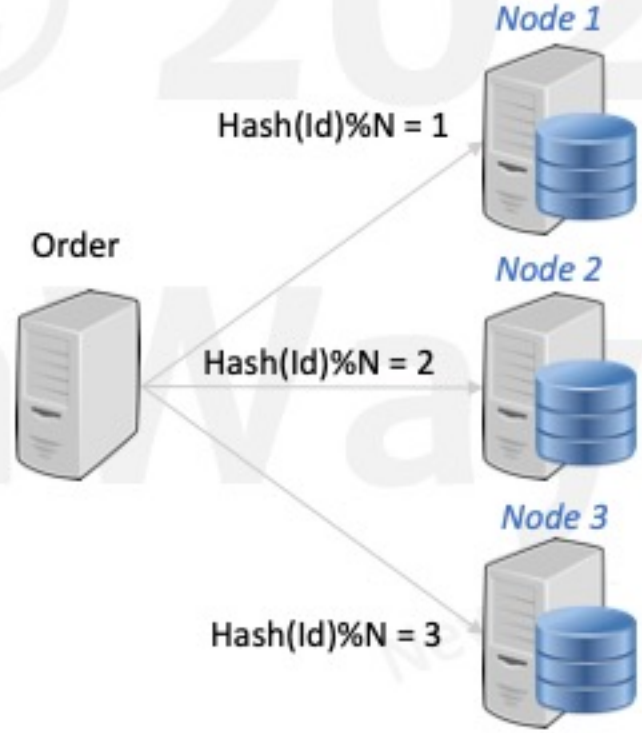
SELECT * FROM Order WHERE id = 150

SELECT * FROM Order WHERE id > 150 AND id < 250



Hash Partitioning

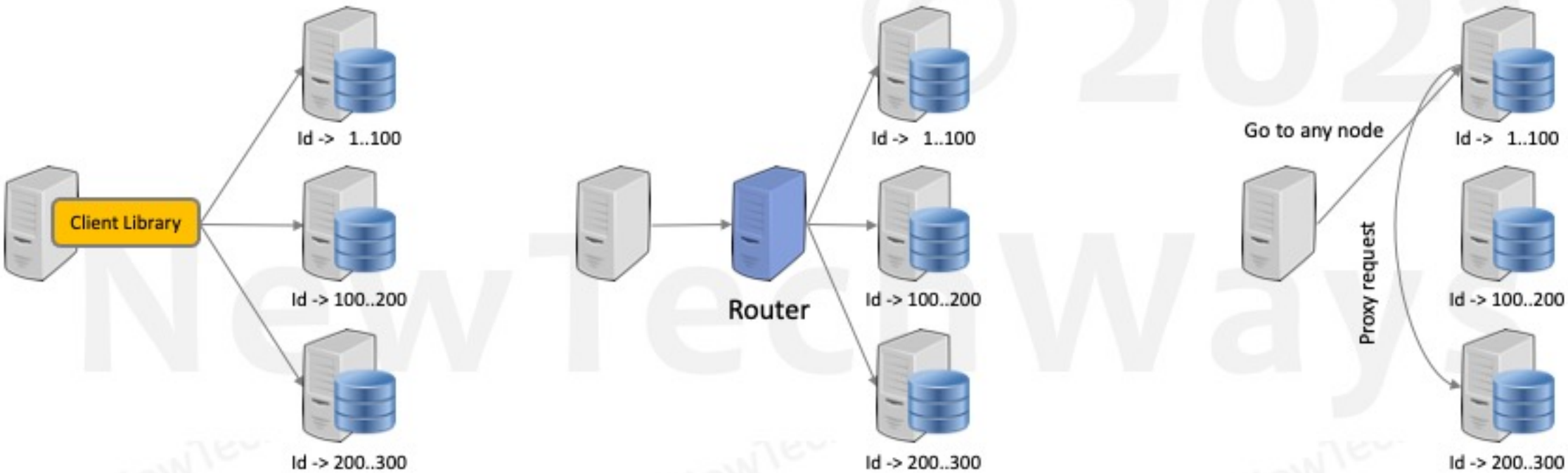
SELECT * FROM Order WHERE id = 150



NewTechWays

Routing with Database Partitioning

Get: Id = 256



NewTechWays

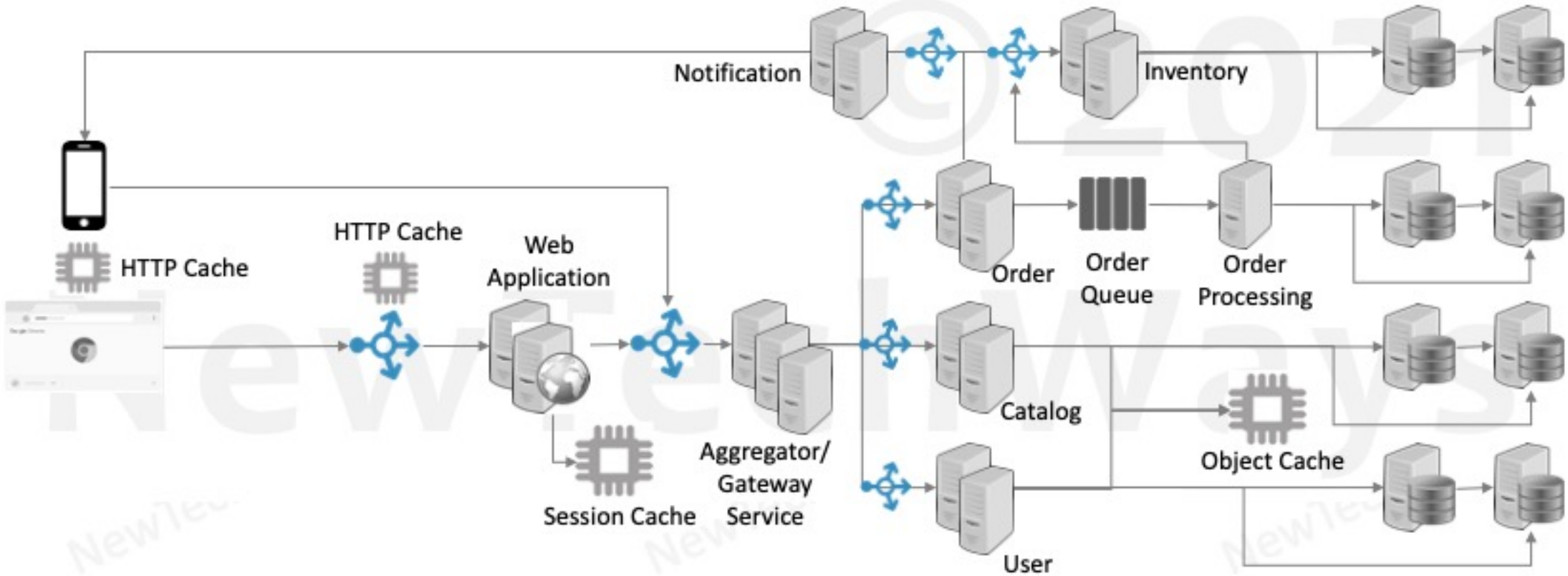
Horizontal Scaling Methods

1. Services
2. Replication
    1. Stateful
    2. Stateless
3. Partitioning
    1. Vertical/Functionality Partitioning
    2. Database Partitioning
4. Asynchronous Calls
5. Caching

Load Balancing

Load Balancing

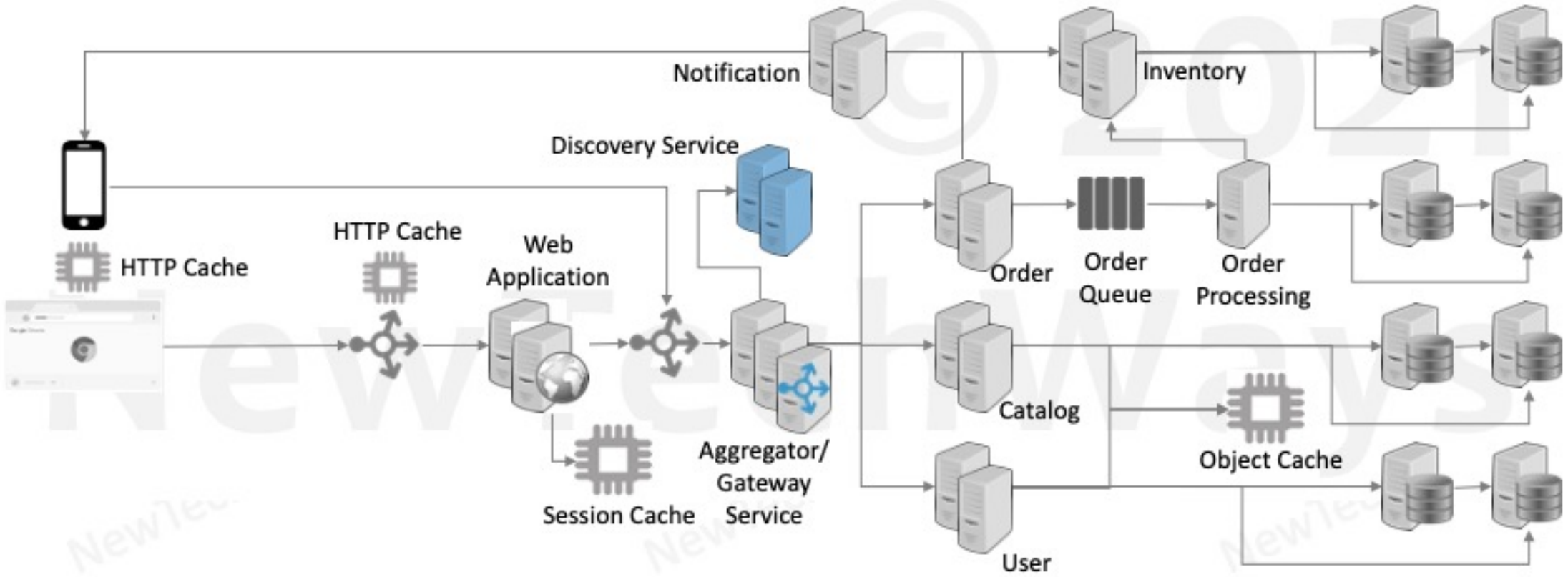
- Single IP address for a Component
- Load Distribution



NewTechWays

Discovery Service & Load Balancing

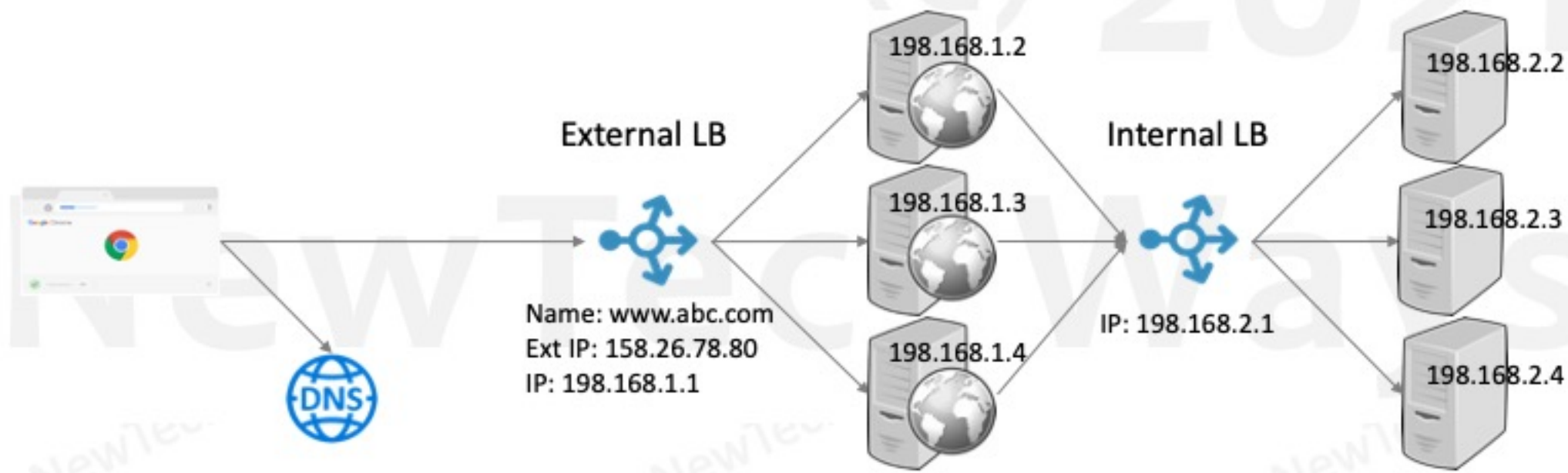
- Discovery – Registry for IP of Healthy Instances



NewTechWays

Load Balancer Discovery

- External Clients – Use DNS to discover the external load balancer
- Internal Clients – Use a local registry/config to discover an internal load balancer



NewTechWays

HLB & SLB

- Hardware Based Load Balancer
  - Load distribution for L4 & L7

2x 14-Core Intel Xeon processors
1.6TB of available use storage space
512GB DDR4 RAM
4x 100G and 8x 40G fiber ports

- F5 Big IP i5000 series
  - Connections: 300 million
  - Throughput: 320/160 Gbps
  - RPS (L7): 10 million

- Software Based Load Balancers
  - Load distribution L7
  - Features
    - Content based Routing
    - Supports SSL Termination
    - Supports Sticky Sessions

Apache

NGINX

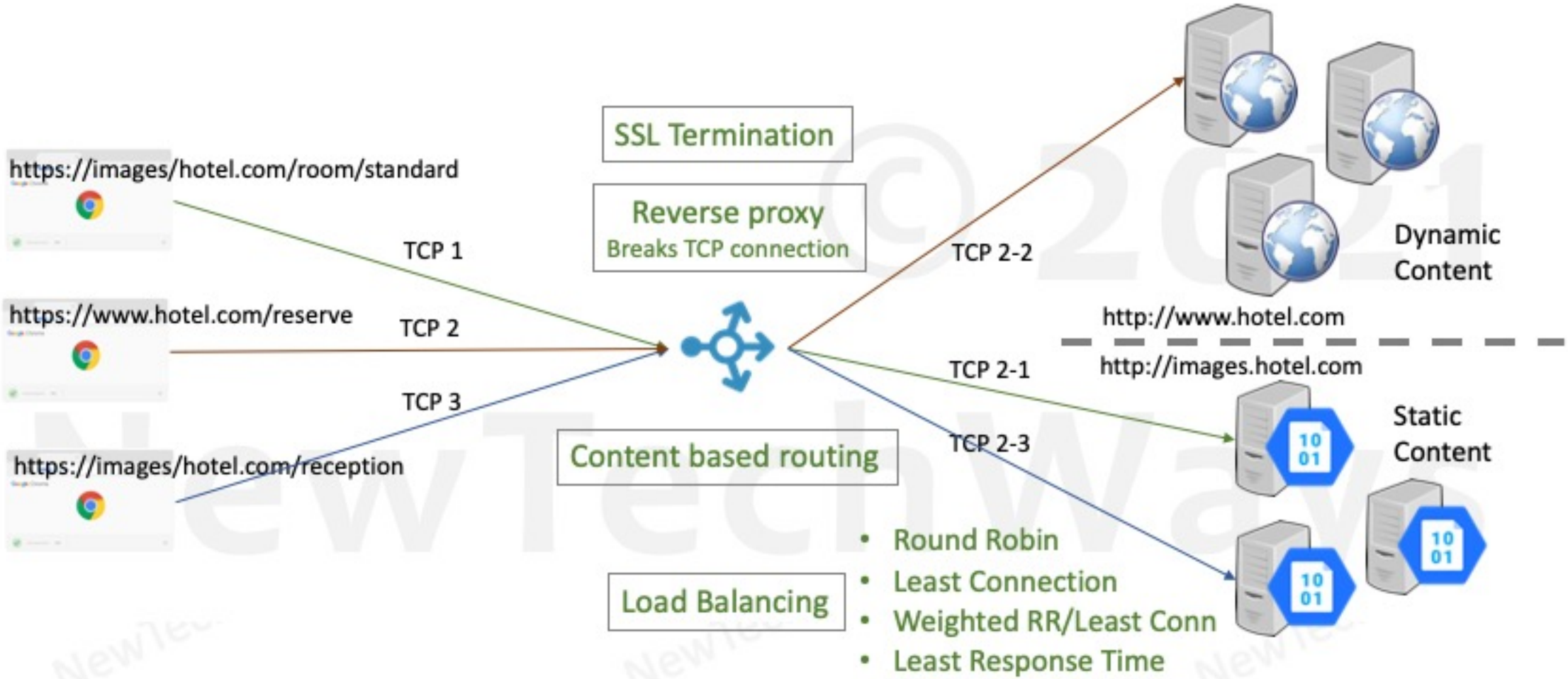
HAPROXY
Powering Your Uptime

TCP/IP Model		OSI Model	
Application Layer	HTTP, HTTPS, SMTP, IMAP, FTP, DNS, NNTP	Application	'Layer 7' Load Balancing
		Presentation	
		Session	
Transport	UDP, TCP, SCTP	Transport	'Layer 4' Load Balancing
Internet		Network	
Network Access (Link)		Data Link	
		Physical	

- NGINX
  - Connections: 225 K
  - Throughput: 70 Gbps
  - RPS: 3 million

NewTechWays

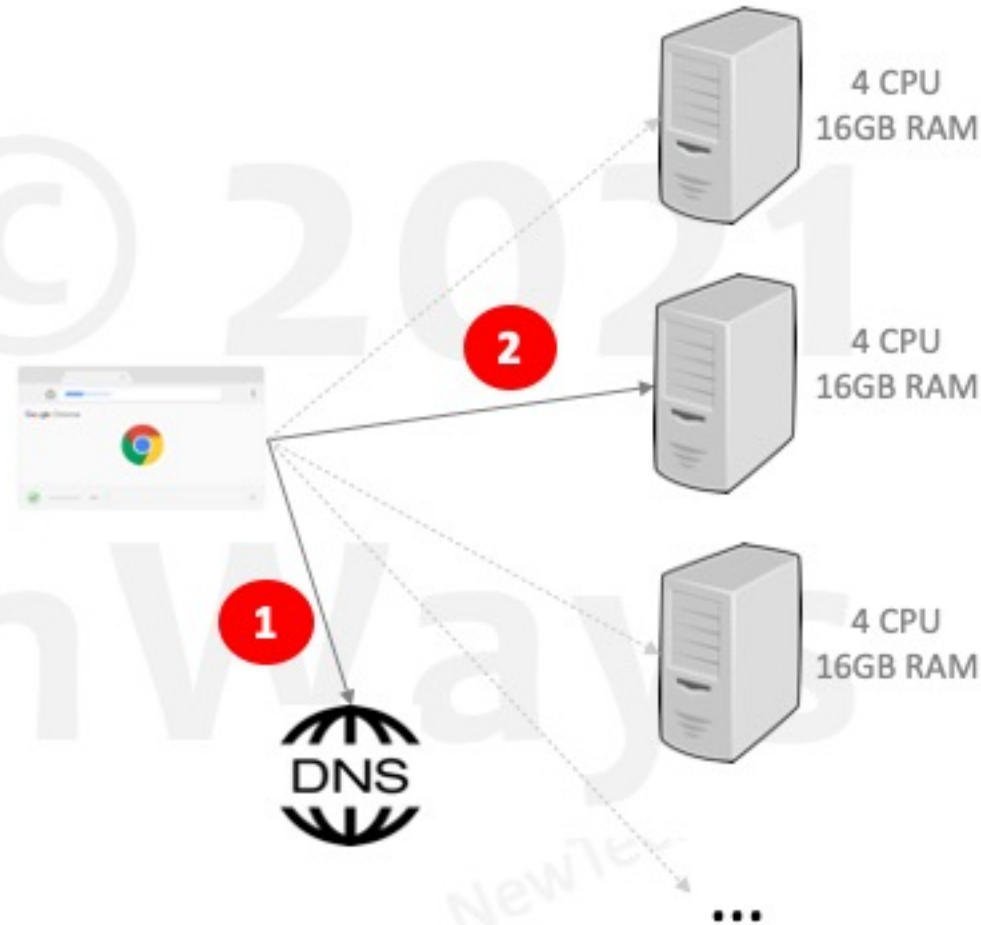
L7 – Load Balancer



NewTechWays

DNS as Load Balancer

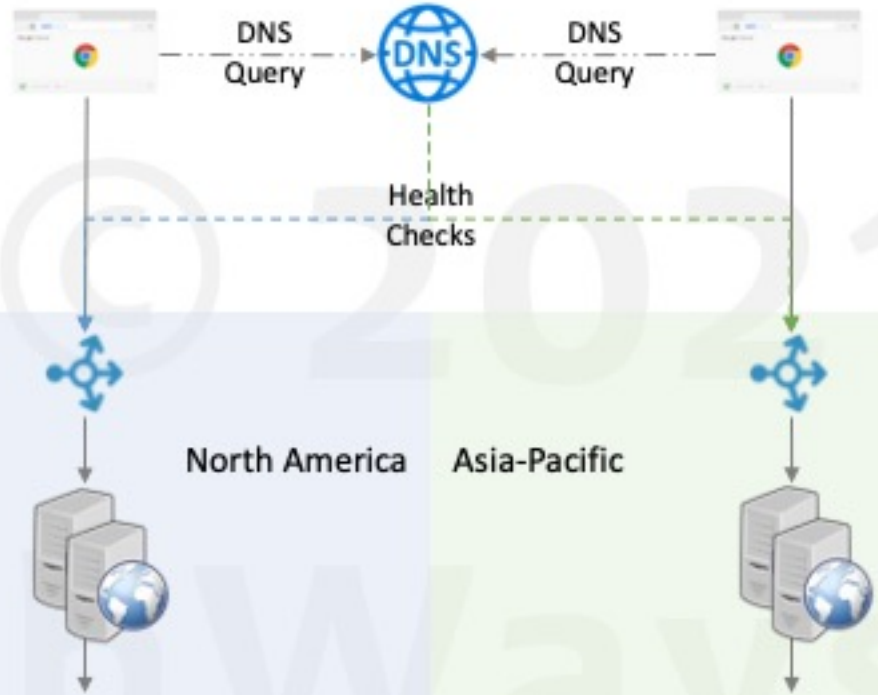
- Configure DNS records with multiple A records
  - Return single IP in a round-robin fashion
  - Return a list of IP
- Cloud based DNS can be configured along with health checks
- Drawbacks
  - Indefinite caching and not respecting TTLs
  - Low or zero TTLs can create a very high load on DNS



NewTechWays

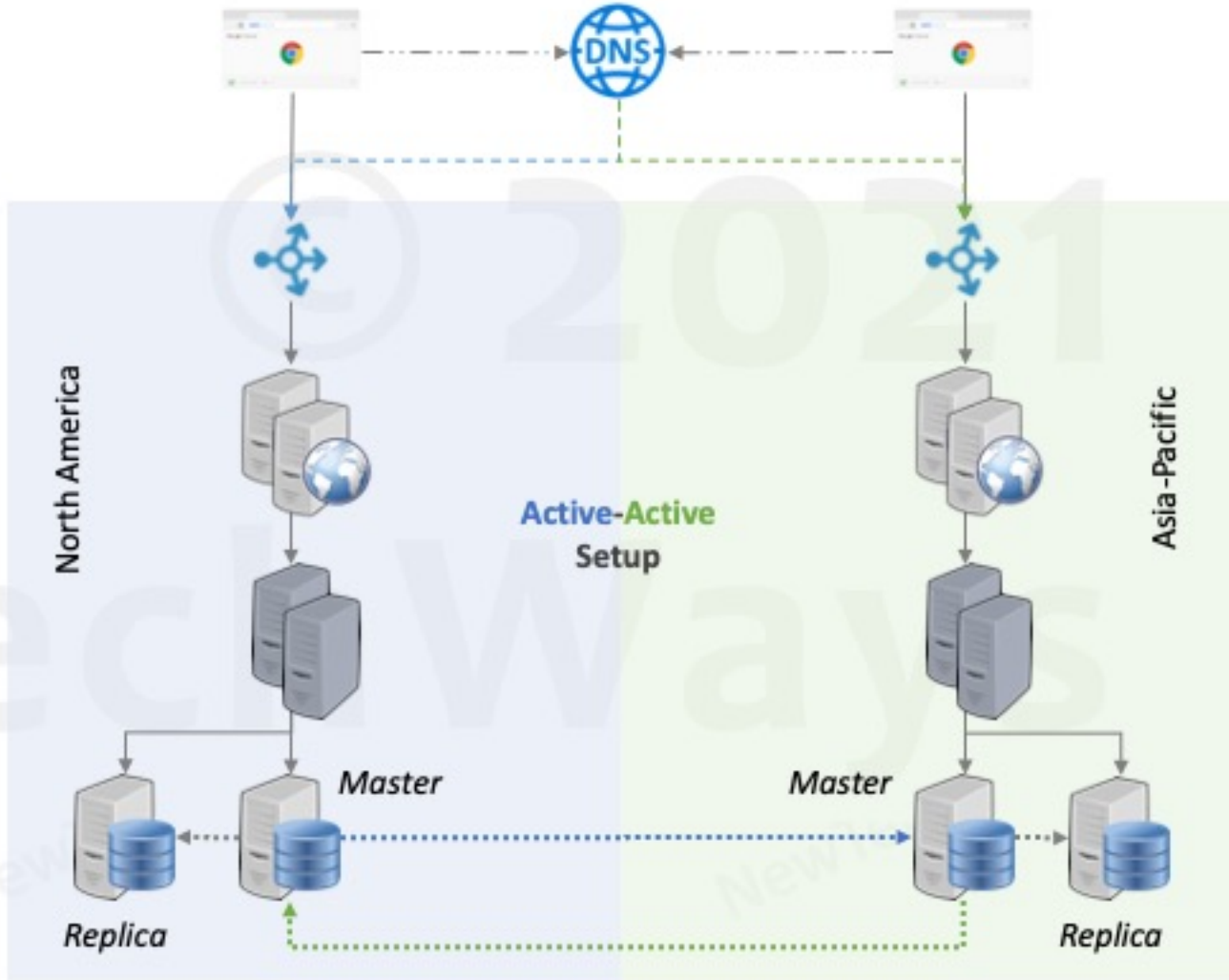
Global Server Load Balancing

- Scalability
  - Routing for multi-geographic systems
- Performance
  - Locality for multi-geographic users
    - Client to Datacenter Latency
    - Client to Datacenter Proximity
    - Datacenter Geography
- High Availability
  - Multi region availability
- Disaster Recovery



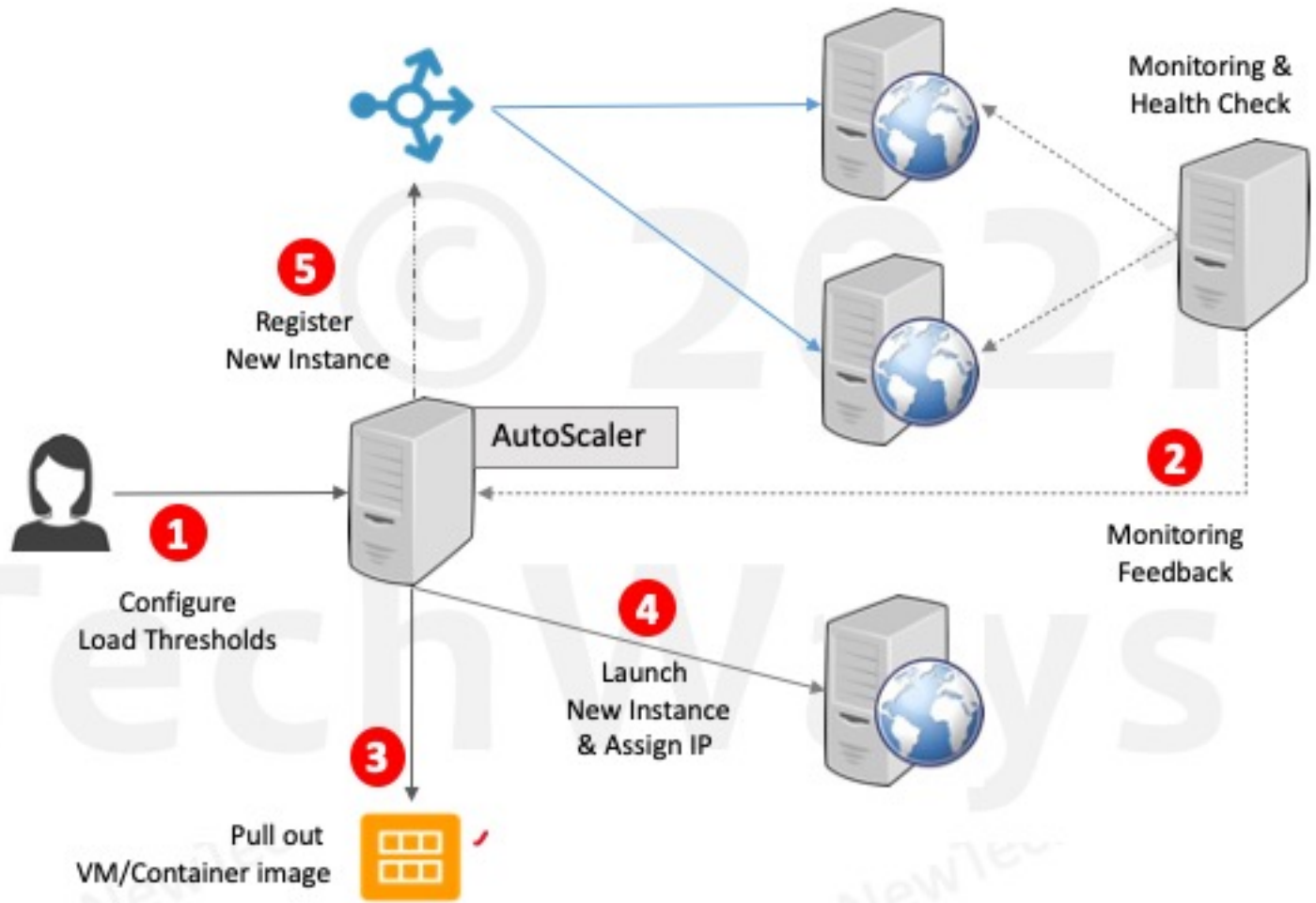
Global Data Replication

- Active-Active Setup
  - All sites active
- Master-Master or Peer-to-Peer replication
  - Mostly asynchronous
- Failover is quick
- Some data loss is a possibility



Auto Scaling

- Monitoring Service
  - Monitor Load
    - CPU
    - Network
    - Disk
  - Monitor Heath
    - Ping
    - Http
- Auto-Scaling Service
  - Configure load thresholds
  - Monitor load
  - Launch New Instance
  - Shutdown Instance



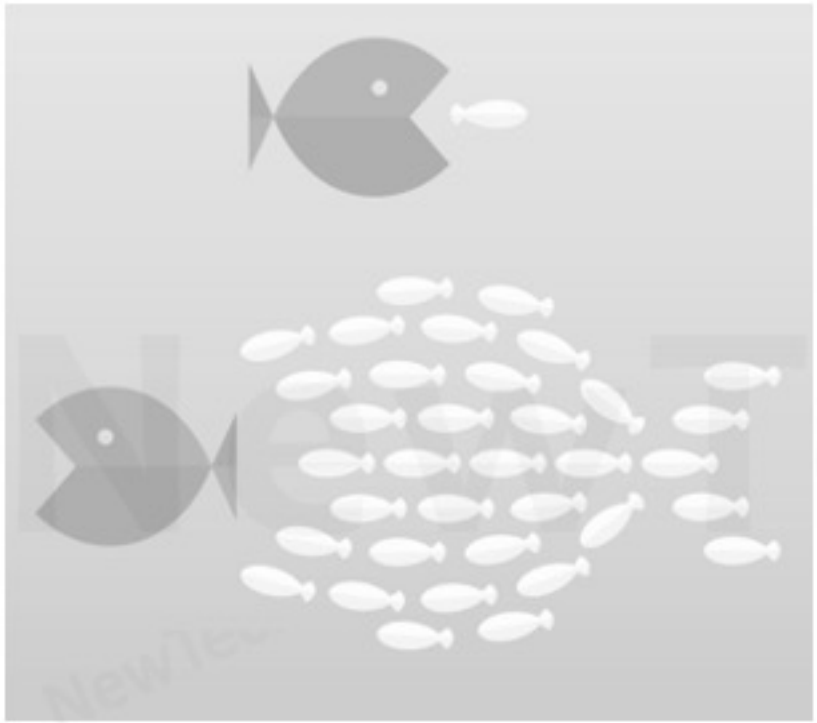
NewTechWays

Micro-Services Architecture

NewTechWays

Micro-Services Motivation

High Scalability
Decentralization & Independence



Frequent Deployment

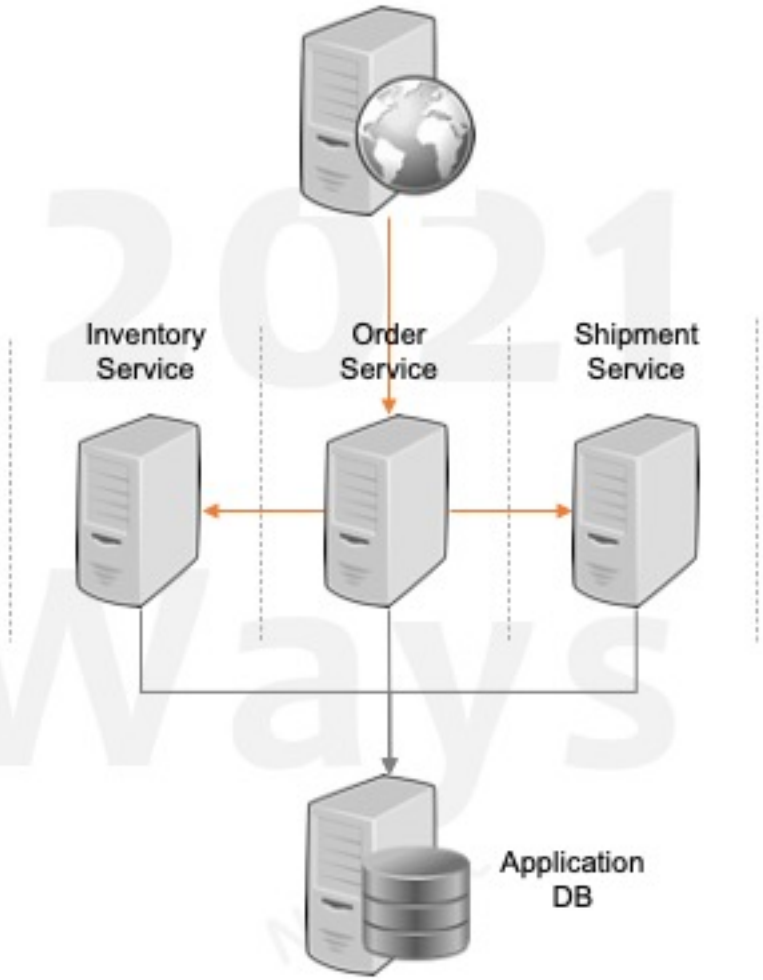
Independent Deployment

Independent Development

Independent Services

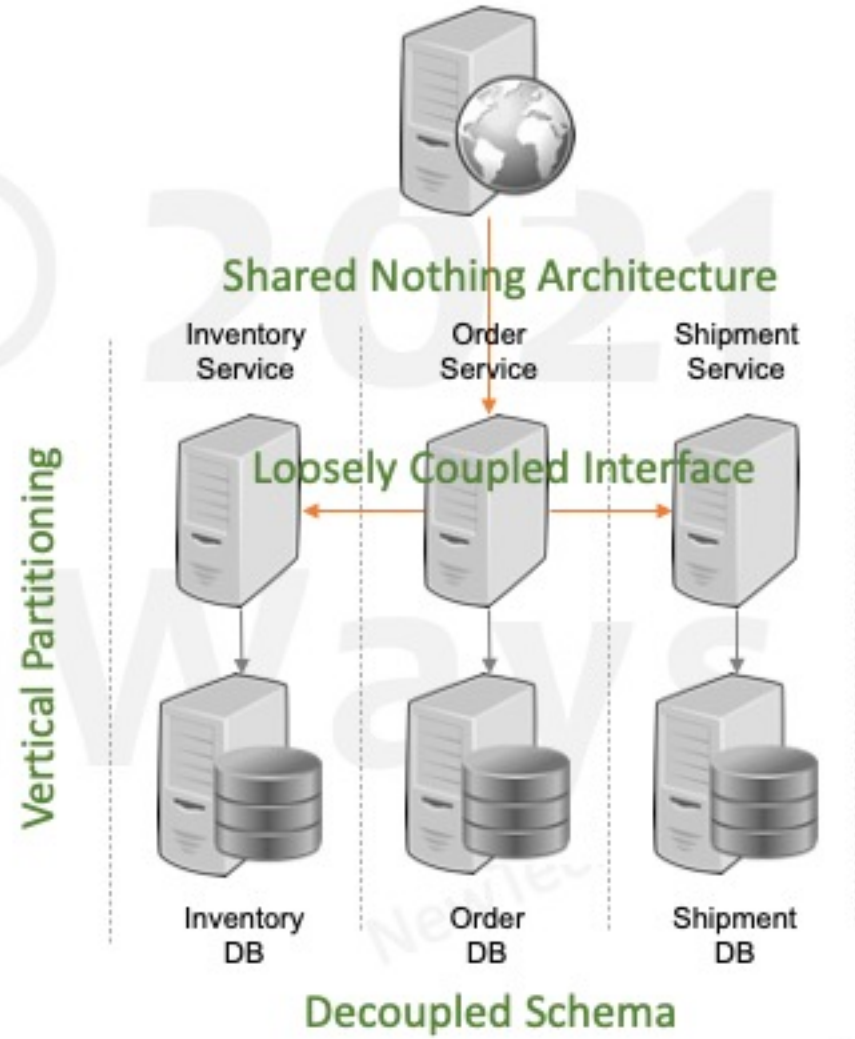
Service Oriented Architecture

- Independent
  - Each service can have its own technology stack, libraries, frameworks etc.
  - Each service can be scaled independently and differently
- Not Independent
  - Common interface schema
    - XML schema
  - Common database schema
    - RDBMS schema
- Issues
  - Service development may be independent but not deployment
  - Single database has scalability limitations



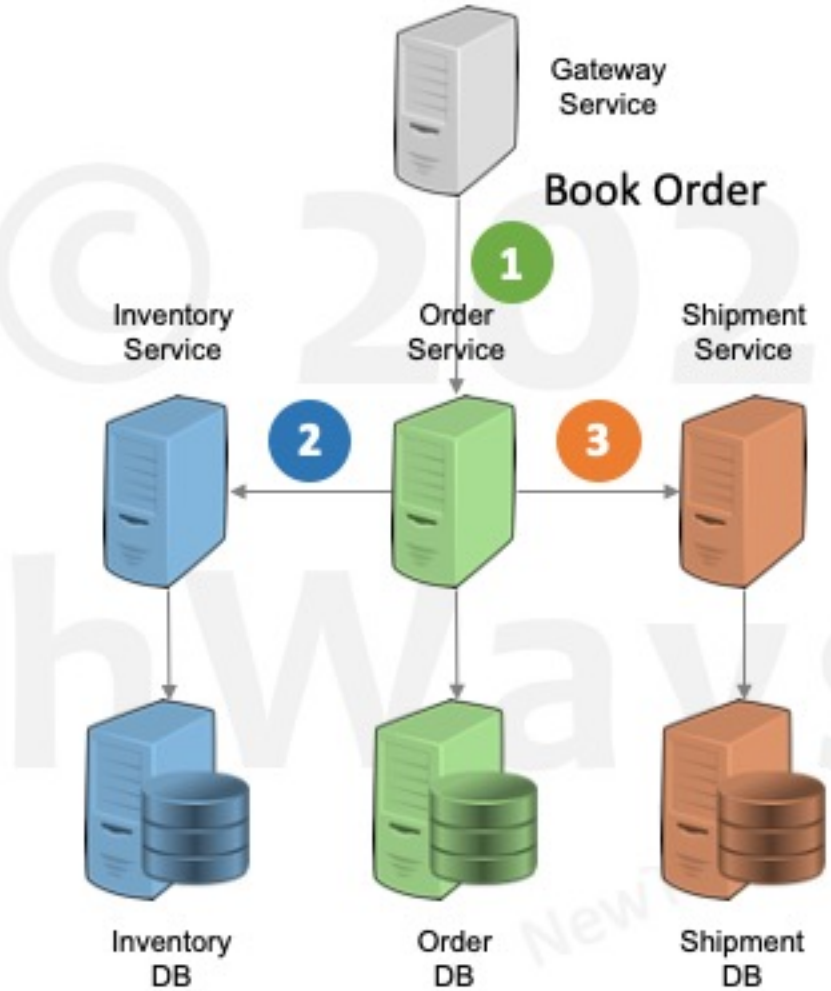
Micro-Services Architecture Style

- Shared Nothing Architecture
  - Services developed and deployed independently
  - Achieved through vertical partitioning
- Vertical/Domain Partitioning
  - Independent schema/database
  - Loosely coupled service interfaces
    - REST interfaces instead of XML/WSDL schemas
  - No reusable libraries except utilities
- Issues
  - Duplicate Codebase
  - Transaction failures
  - Transaction rollbacks



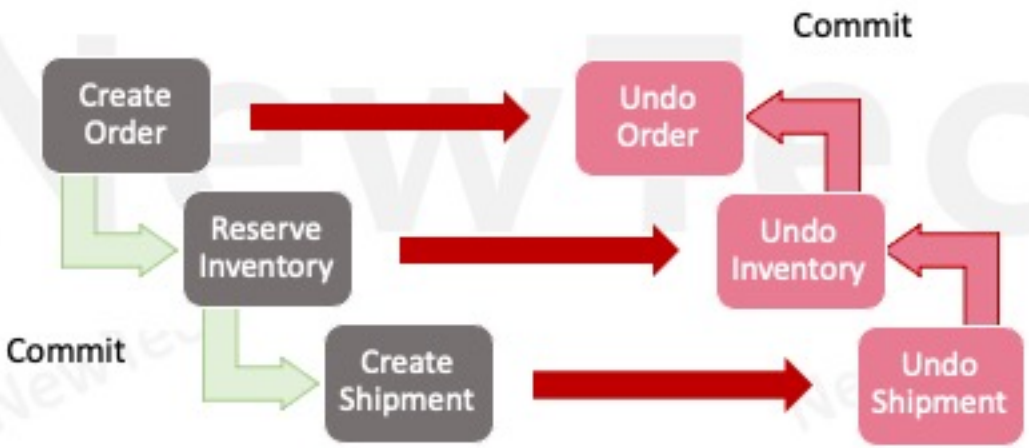
Micro-Services Transactions

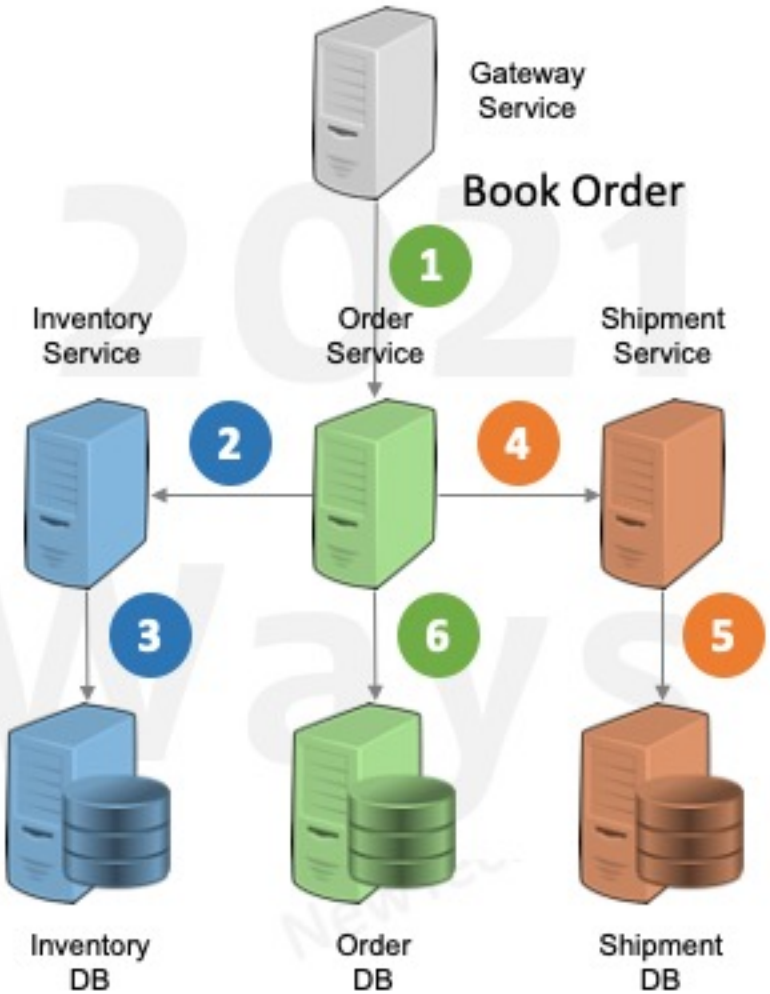
- Transaction involves multiple machines
  - Distributed services with their own DB
  - Local transaction not possible
- Options
  - Distributed ACID Transactions
    - 2PC/3PC
    - Completely ACID
  - Compensating Transactions
    - SAGA pattern
    - Eventually consistent model
      - Relaxes consistency and isolation



Compensating Transactions – SAGA Pattern

- 'Logical Undo' of a partially committed transactions
  - Flow of reversal may not be exactly opposite, and some steps can be executed in parallel
  - Compensation itself can fail. Should be able to restart itself and retry
- Asynchronous processing for reliability

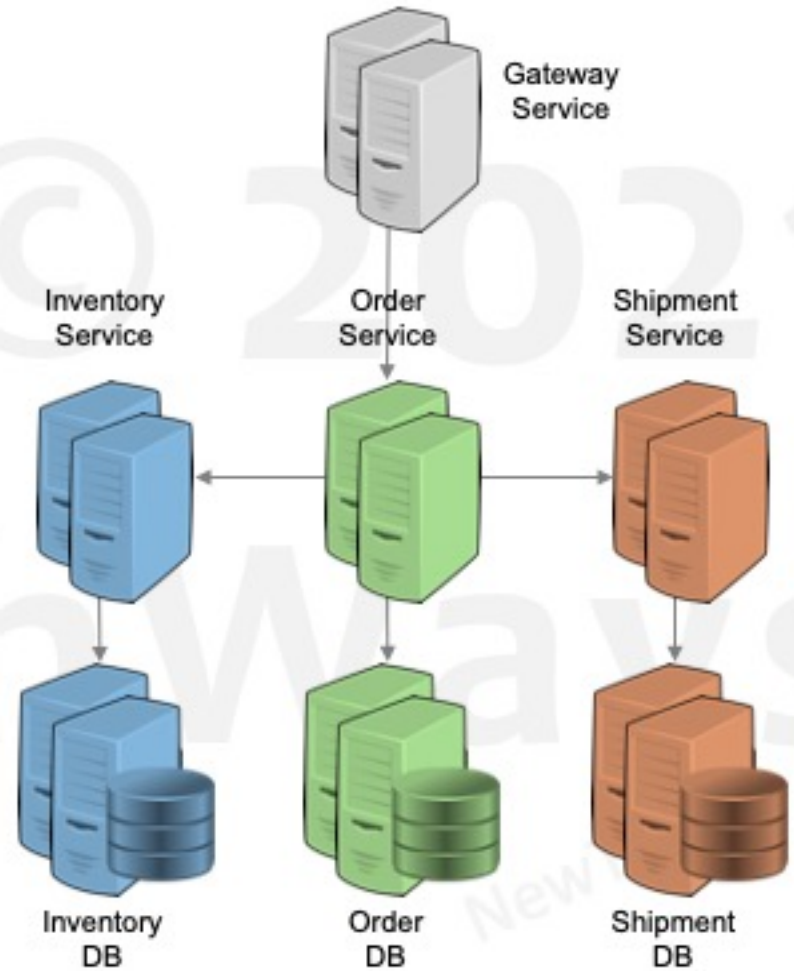




NewTechWays

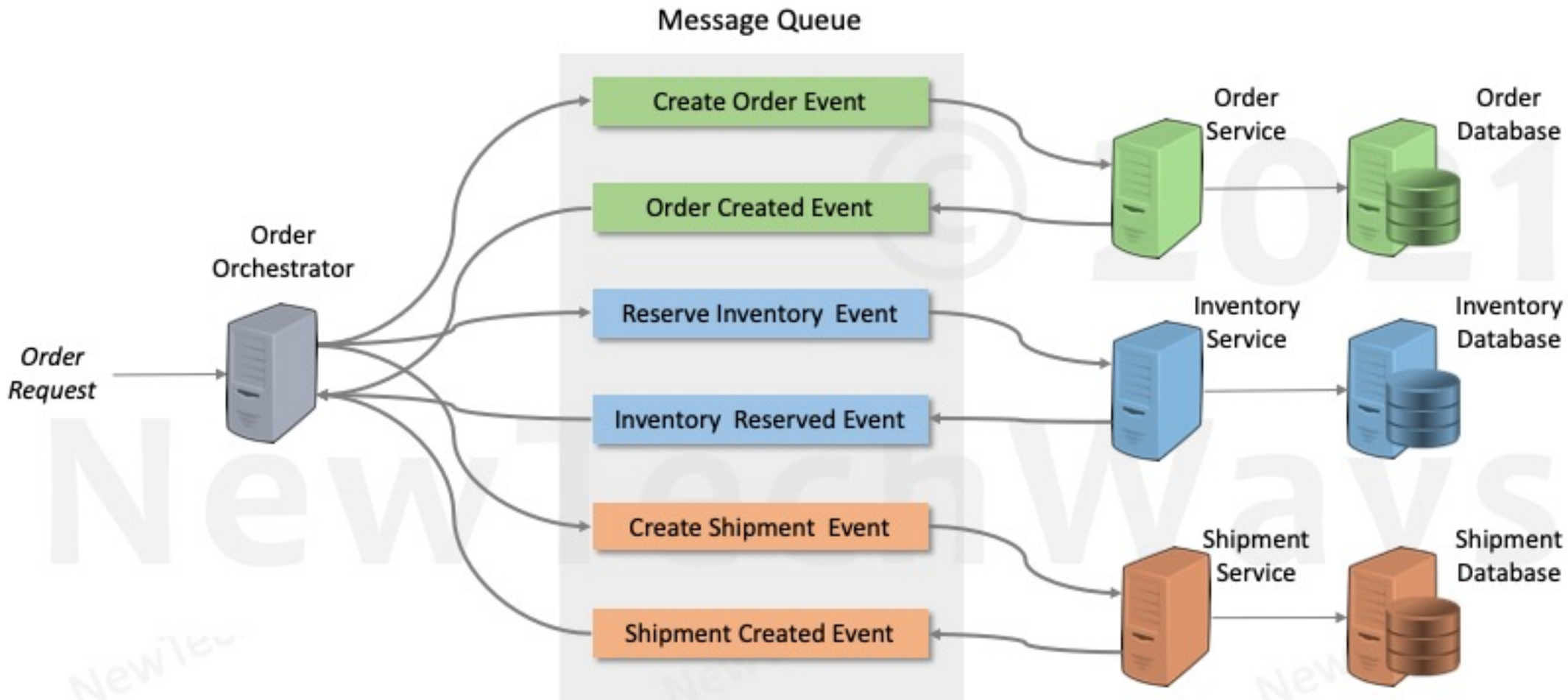
Micro-Services Communication Model

- Synchronous processing
  - Immediate Response
  - For read/query loads
- Asynchronous processing
  - Deferred response
  - For write/transaction loads
  - Higher scalability
  - Higher reliability



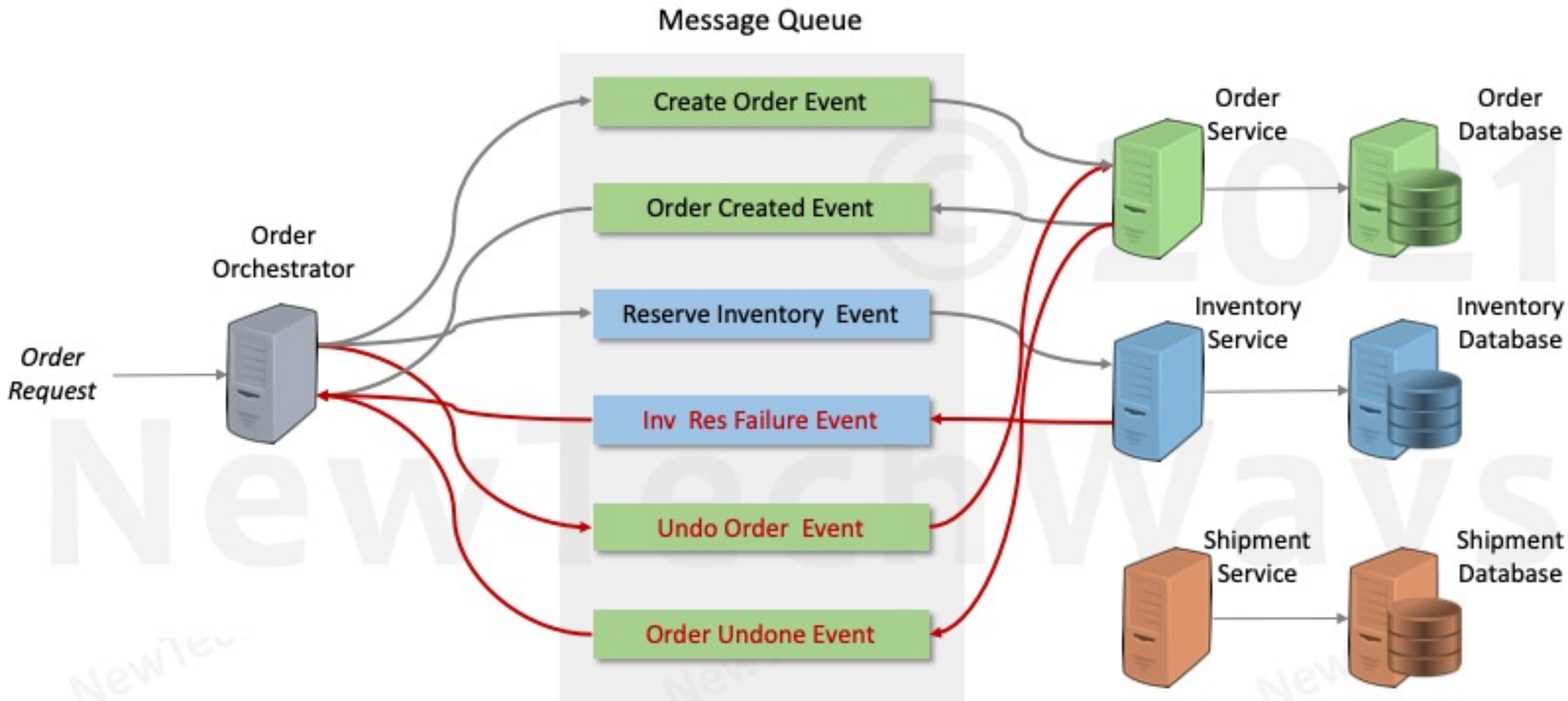
NewTechWays

Micro-Services Event Driven Transactions



NewTechWays

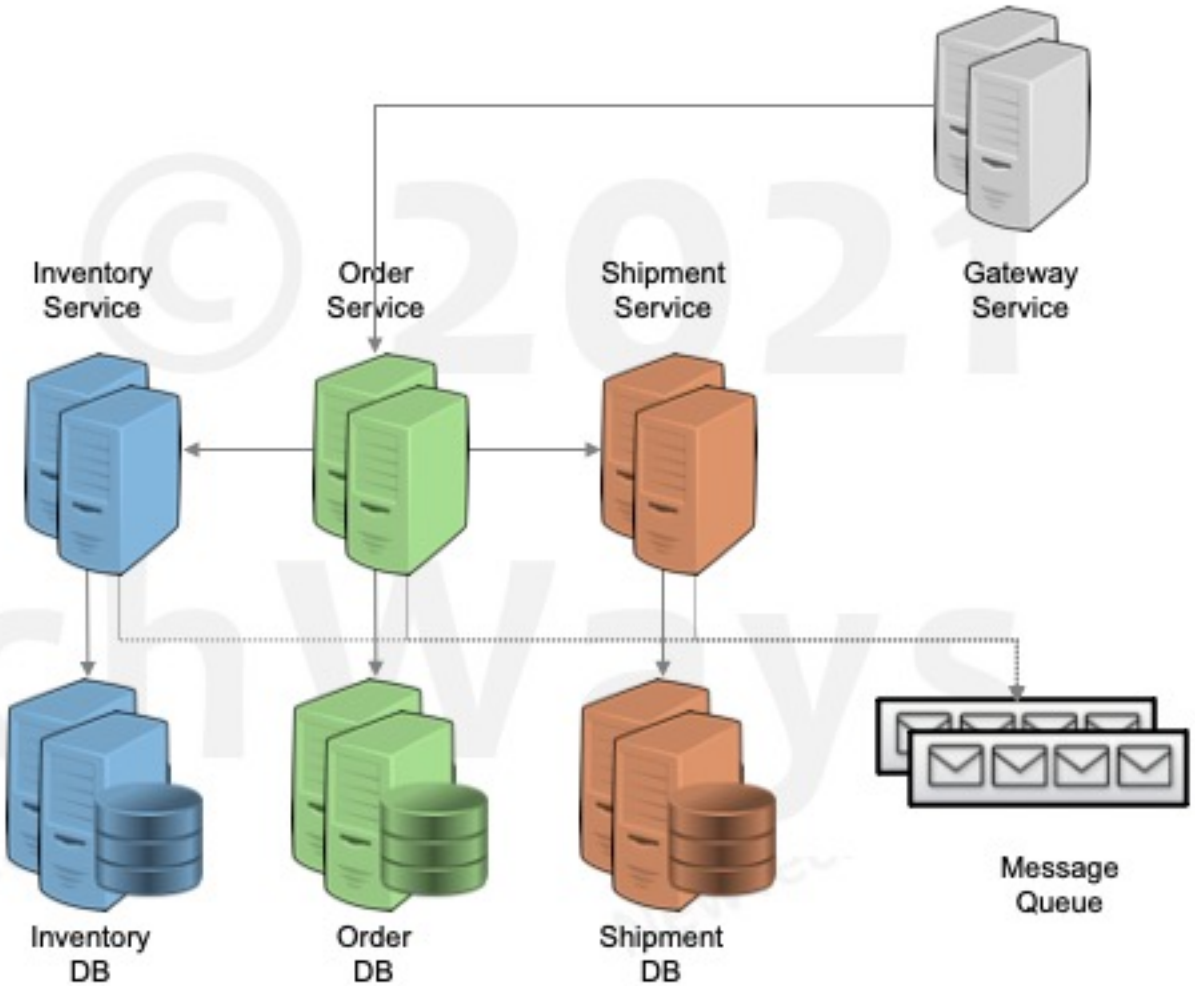
Micro-Services Event Driven Transactions



NewTechWays

Extreme Scalability With NoSQL & Kafka

- Microservices Transactions
  - ACID within Service
  - Compensating Transaction across services
    - Eventually consistent
- NoSQL DB
  - ACID transaction at aggregate level
  - Eventually consistent transactions across aggregates
  - Low latency operations
    - Multiple nodes
  - High scalability
    - Horizontal partitioning
- Kafka
  - Horizontal partitioning of topics



Summary

- Scalable systems are decentralized, and their components function independently
- To make a system scalable:
  - Cache frequently read and rarely mutating data to reduce load on the backend
  - Asynchronous or Event driven processing for distributing load over time
  - Vertical partitioning of functionality into independent, stateless, replicated services
  - Partitioning and replication of state for extreme scalability
- Scalable systems requires infrastructure:
  - Load balancers – Hardware based (L4 + L7) & Software based (L7)
  - Discovery services for service discovery and health checks
  - DNS as load balancers at global scale
- Microservices for extreme scalability
  - Fully vertically partitioned services and databases leads to eventual consistency

Thanks!

NewTechWays

https://www.newtechways.com