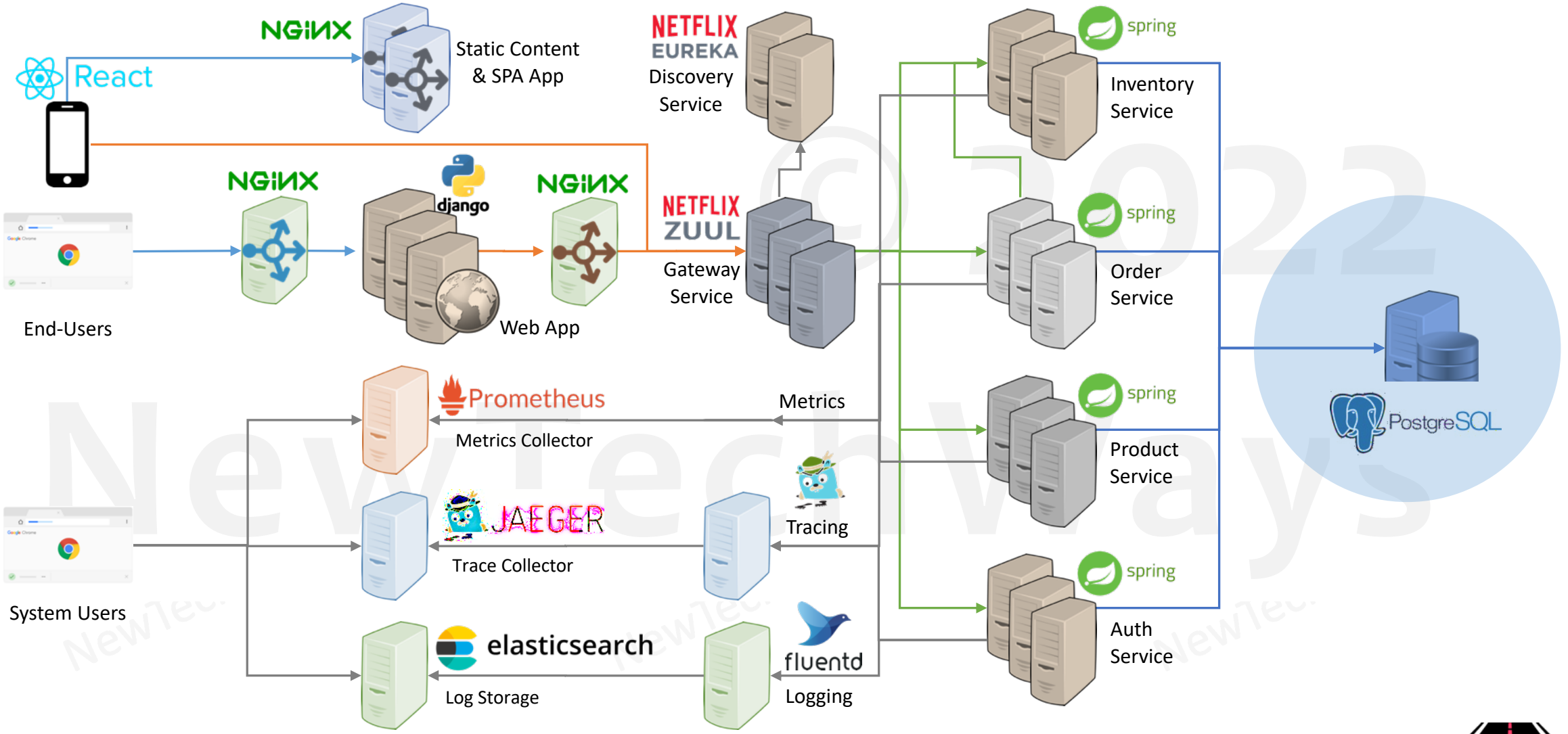
Data Management at High-Scale

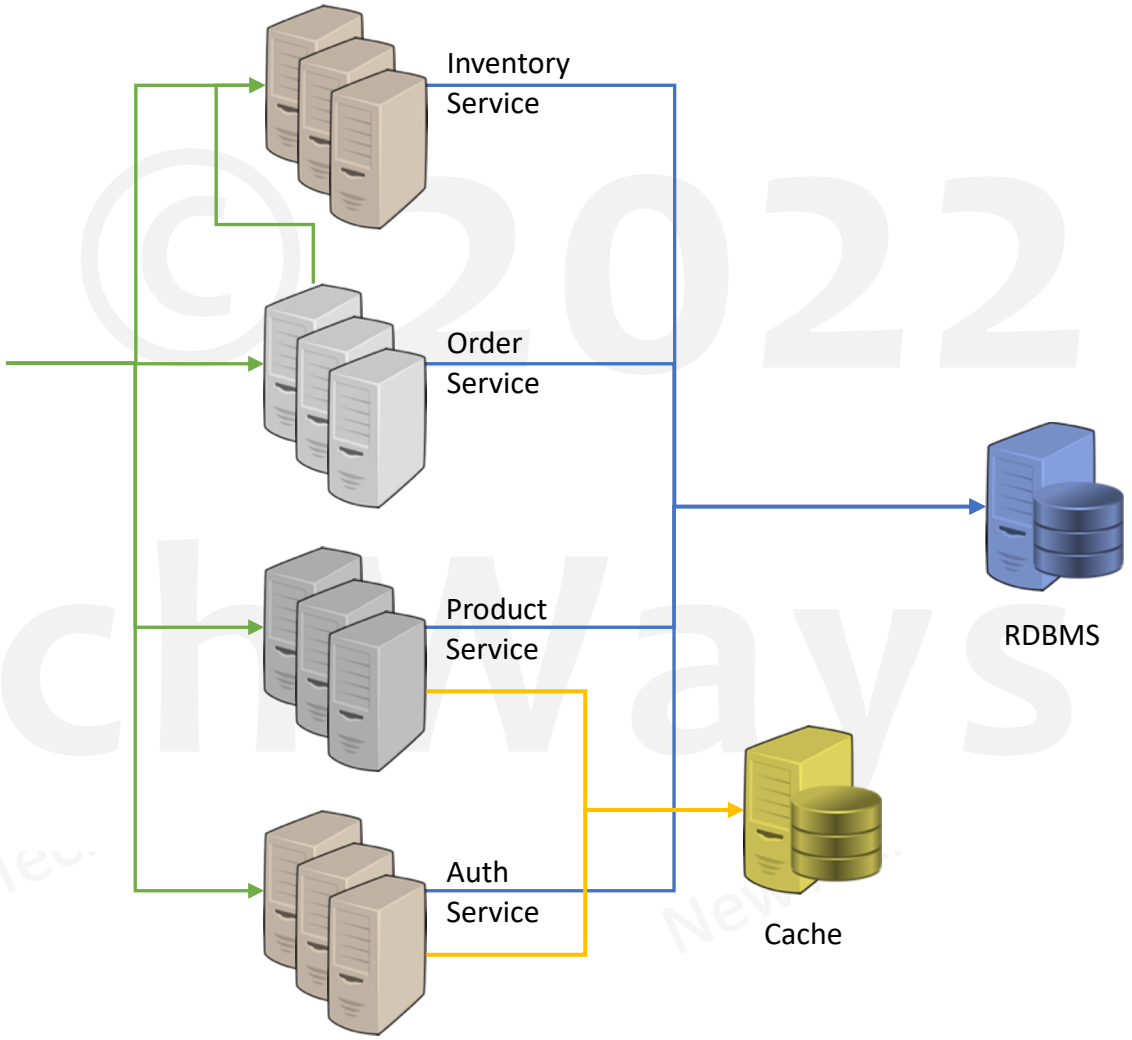
High Data Load



NewTechWays

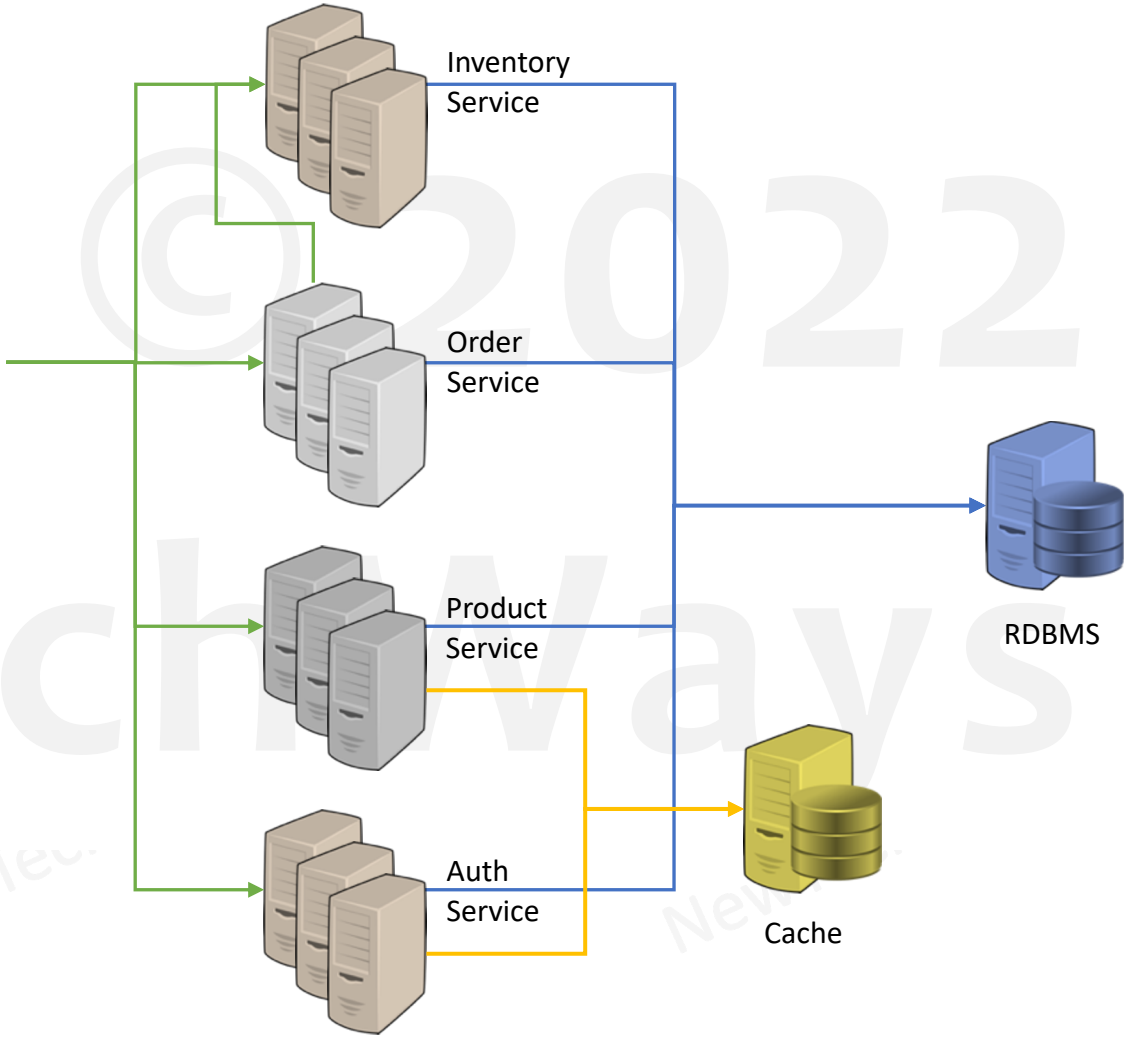
Managing Read Load – Caching

- Reduces data read load from the backend
  - Improve performance of read requests
  - Creates more capacity for write requests
- Challenge is to deal with stale data
  - TTL Values
    - Not ideal for dynamic data
  - Batch update
    - Read origin and update cache
  - Dynamic update
    - Update cache on add/update/delete



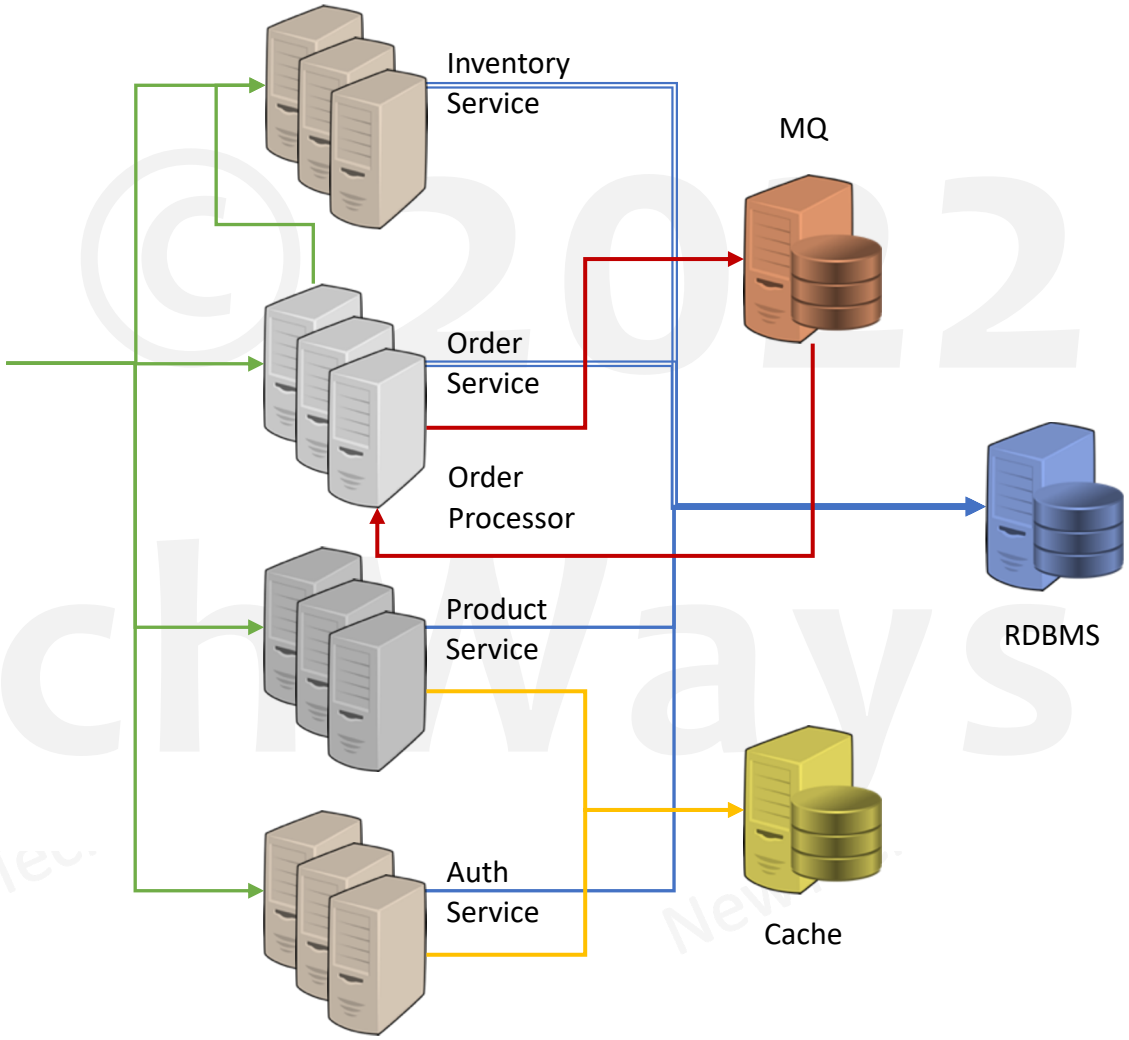
Managing Read Load – Caching

- Eviction Policy
  - LRU/LFU/TTL
    - Datasets that cannot fit in memory
    - View a product, user
  - No Eviction – Like DB
    - Datasets that can fit in memory
    - View all products in a category
- Cache Persistence
  - No frequent cache warmup
    - Restart – Maintenance/Failure



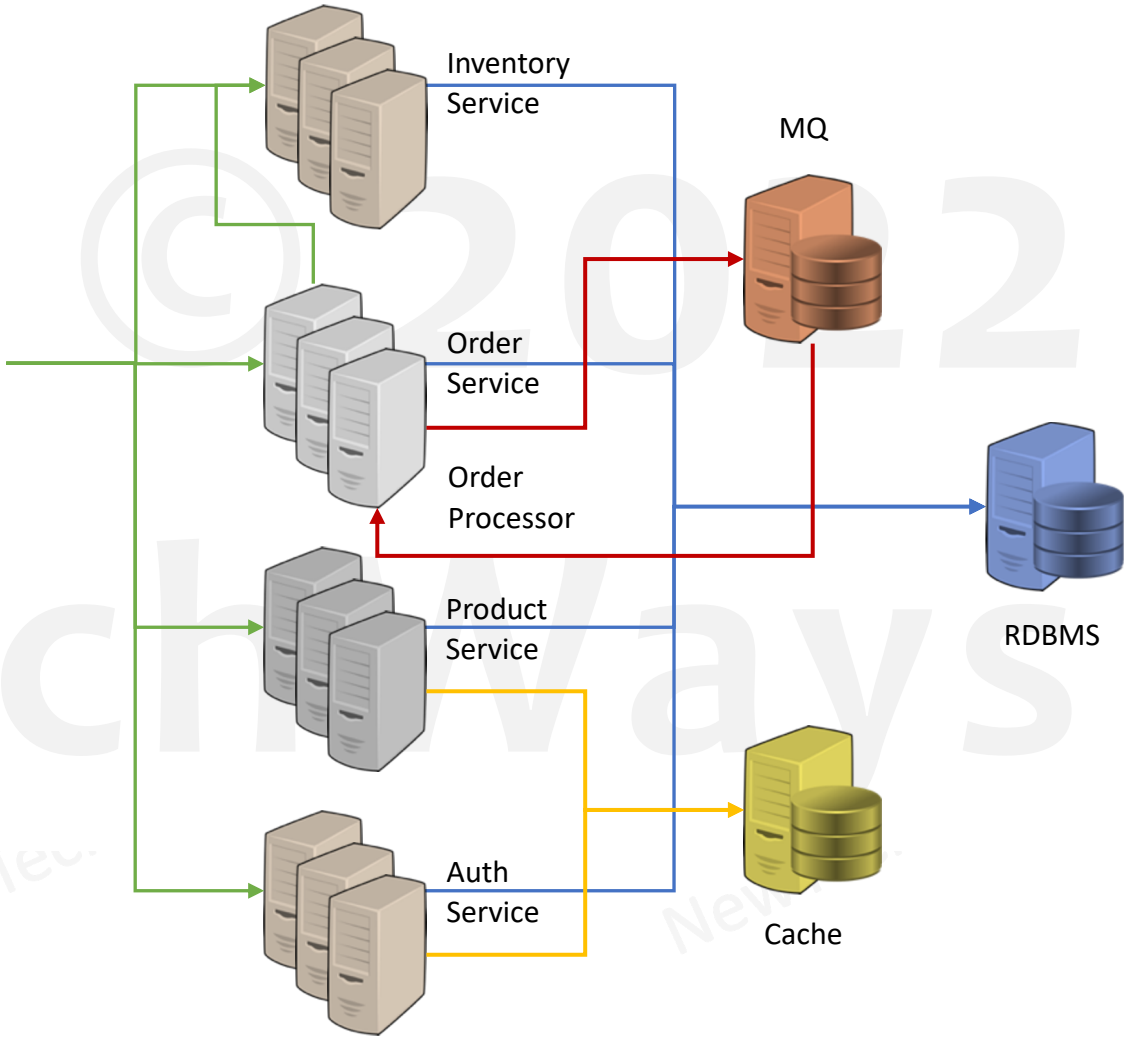
Managing Write Load – Asynchronous Processing

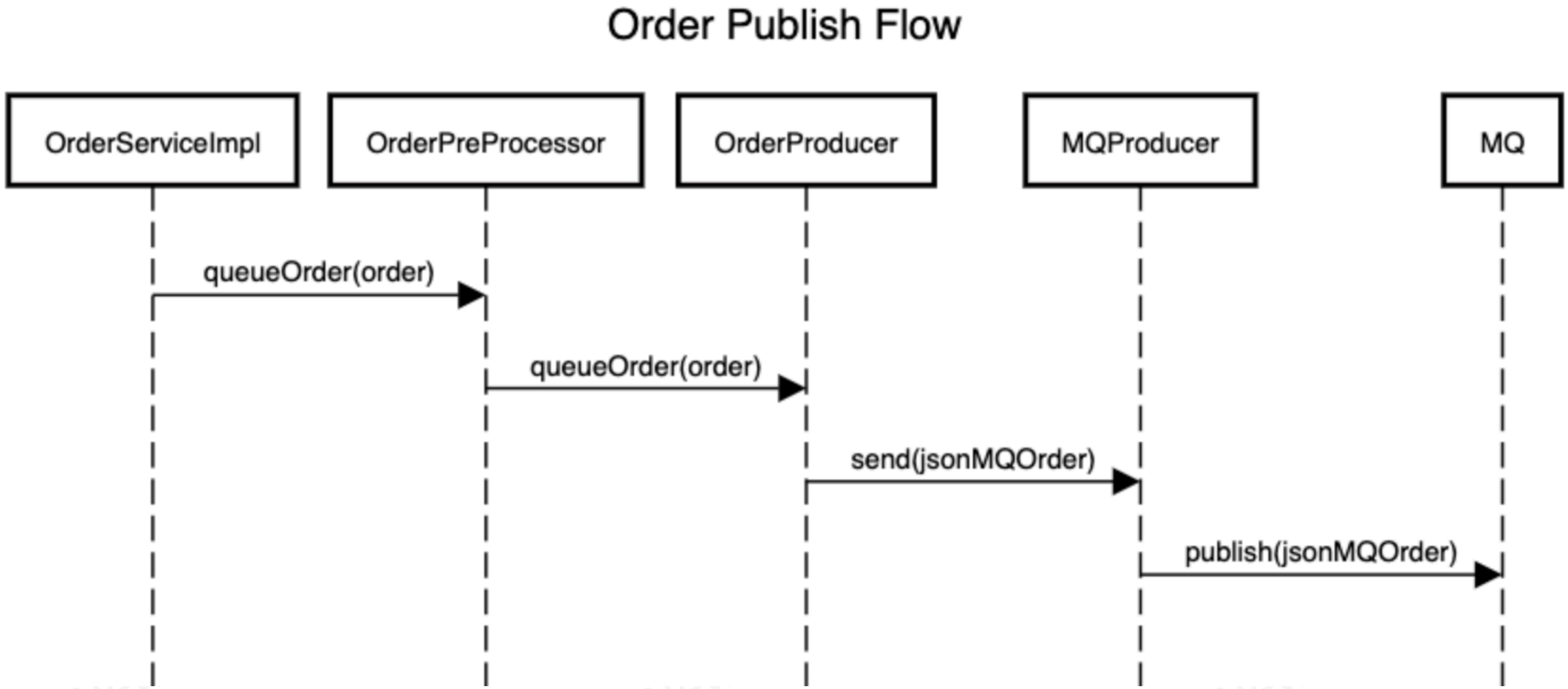
- Decouple write APIs from read APIs
  - Write APIs can be executed asynchronously
- Decouple order capture from order processing
  - Synchronous order capture
    - Low latency
    - Low failure rates
    - Improved user experience
  - Asynchronous order processing
    - Can wait for capacity availability
    - Asynchronous notification on completion



Managing Write Load – Message Queue

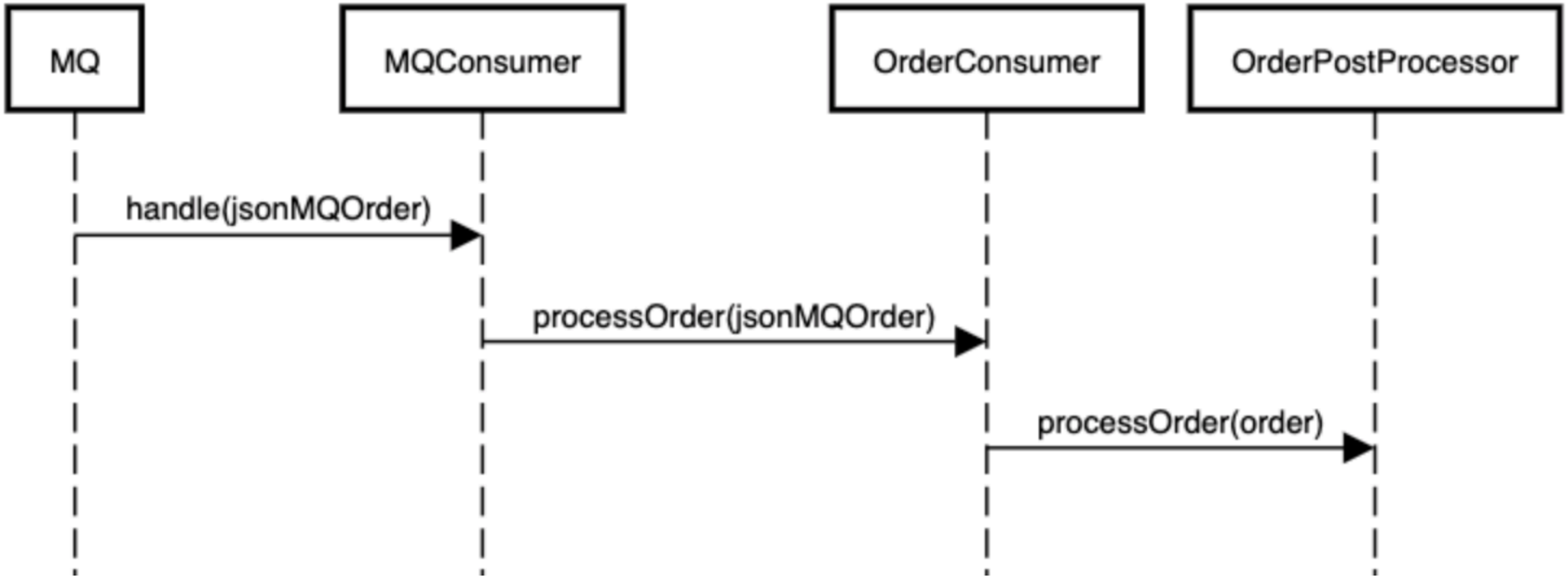
- Kafka is a distributed queue for high scalability
  - Partitioning of queue
- Kafka tradeoffs
  - Complex setup, requires Zookeeper
  - Message order only within a partition
  - No push message delivery
    - Pull requires constant polling





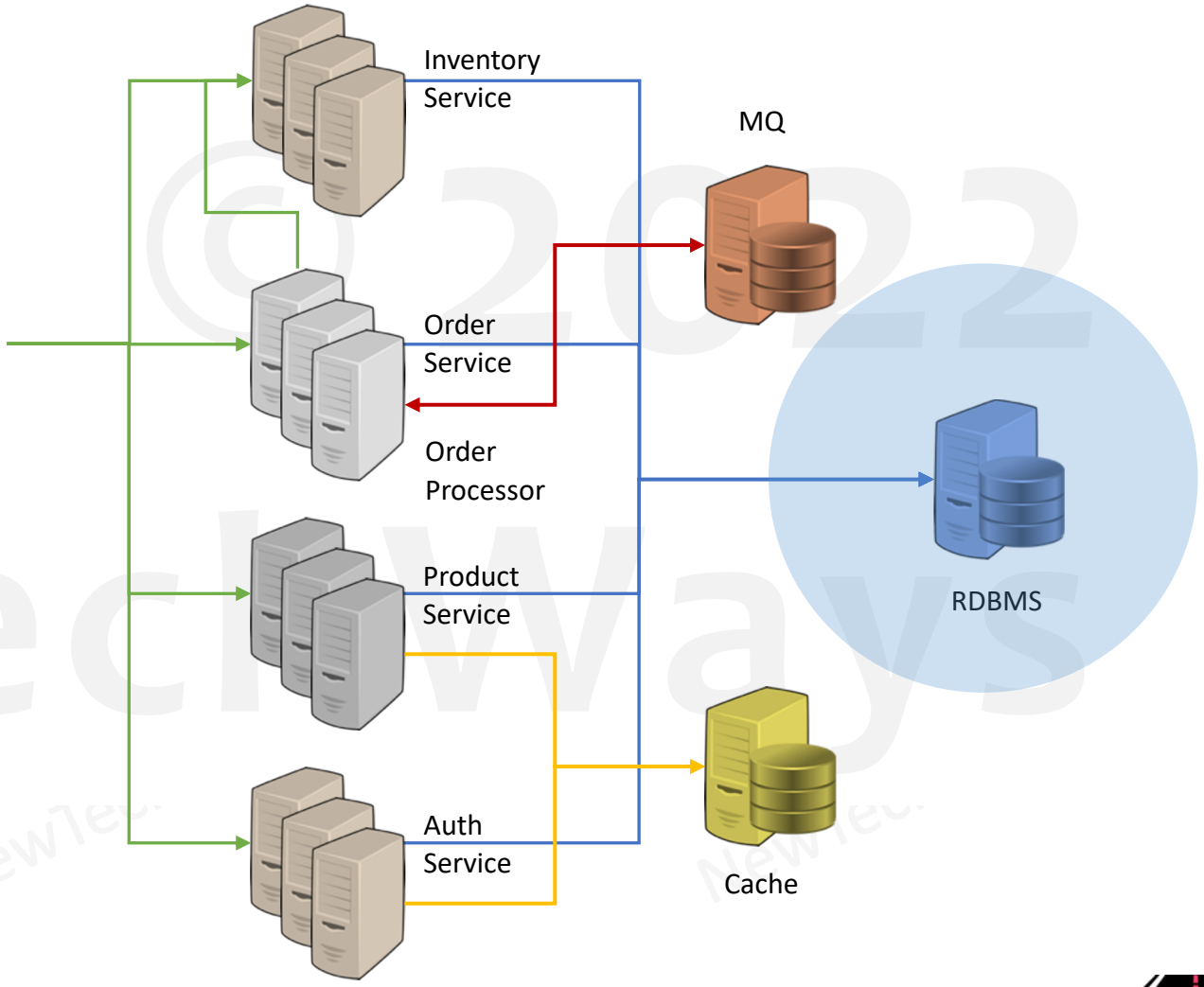
NewTechWays

Order Consumer Flow



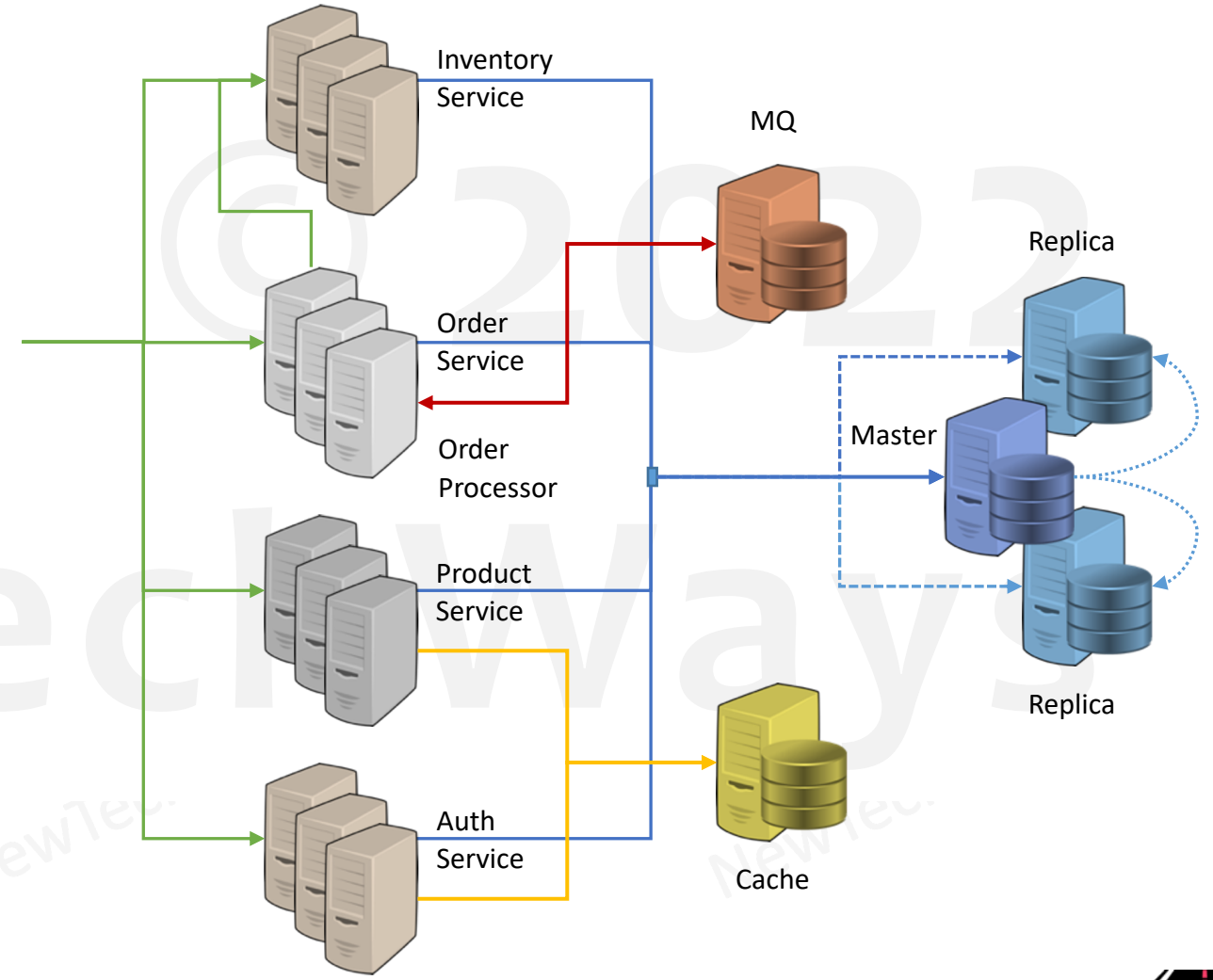
RDBMS Limitations

- RDBMS limitations
  - Single instance
  - < 1~5 TB of data



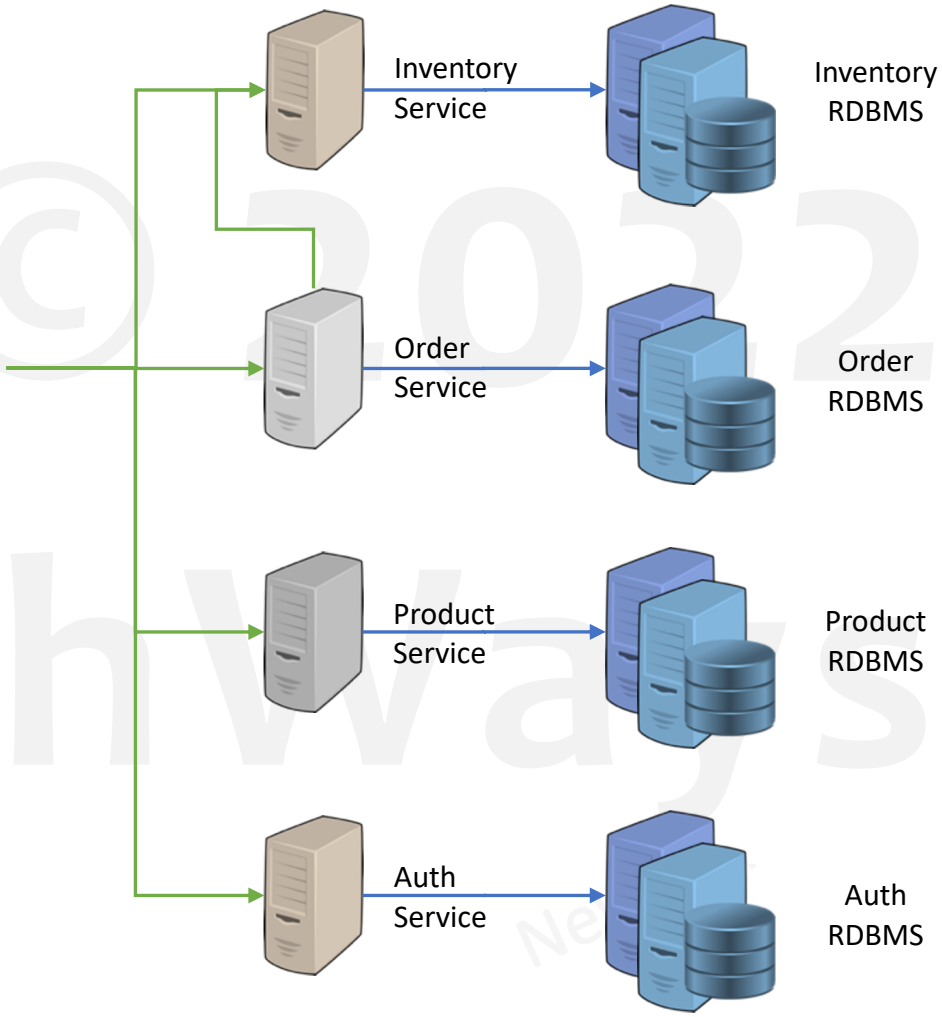
Partitioning Data Load

- Replication
  - High read load
    - Eventually consistent read queries
  - High availability
  - Write load is still a problem



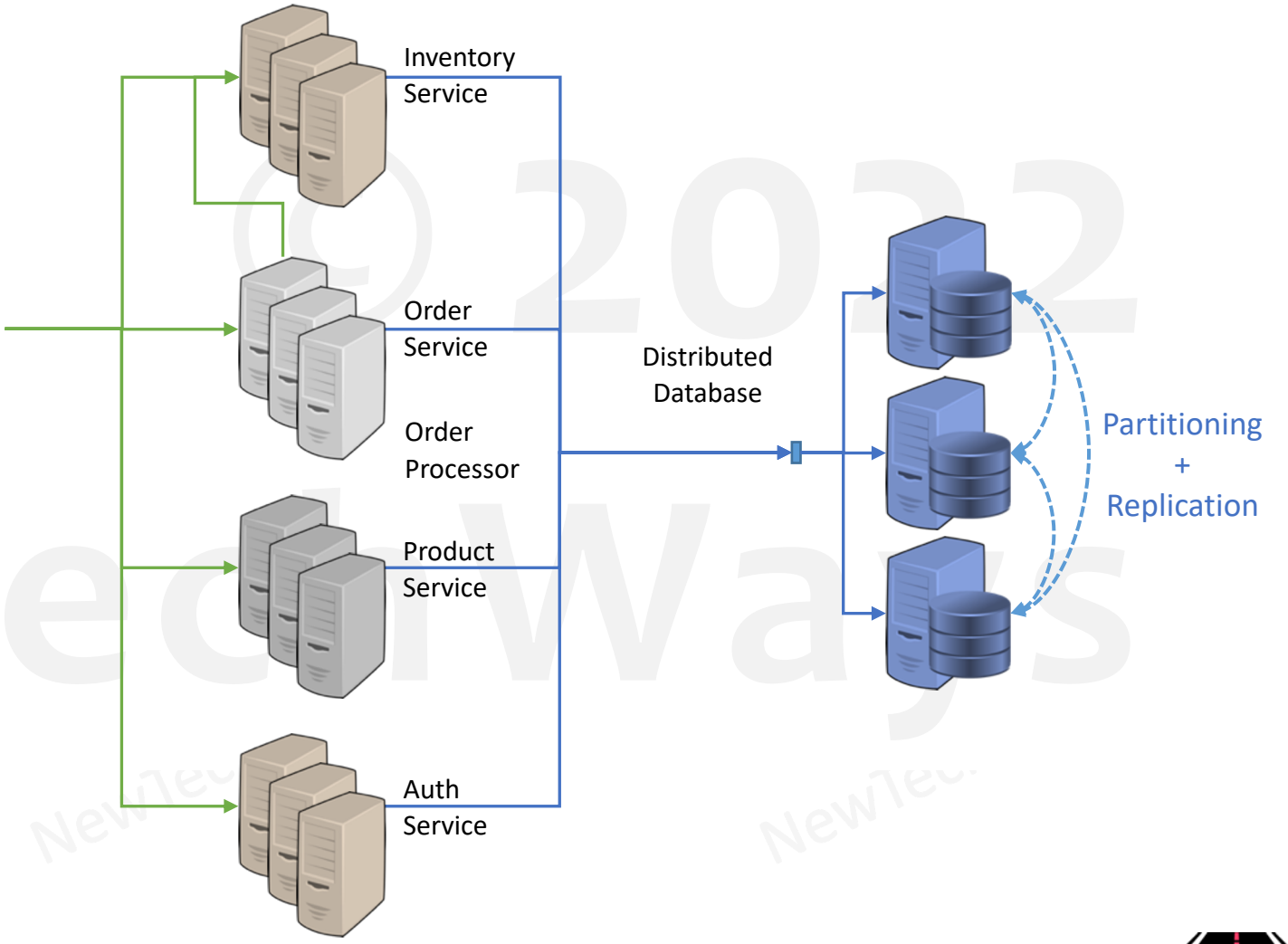
Partitioning – Application Level

- Vertical Partitioning
  - Microservices Architecture
    - Independent development and deployment
- Within a database
  - ACID, Joins, Constraints
- Across databases
  - No integrity constraints
  - No query joins
  - No ACID transactions
- Complexity of maintaining multiple databases



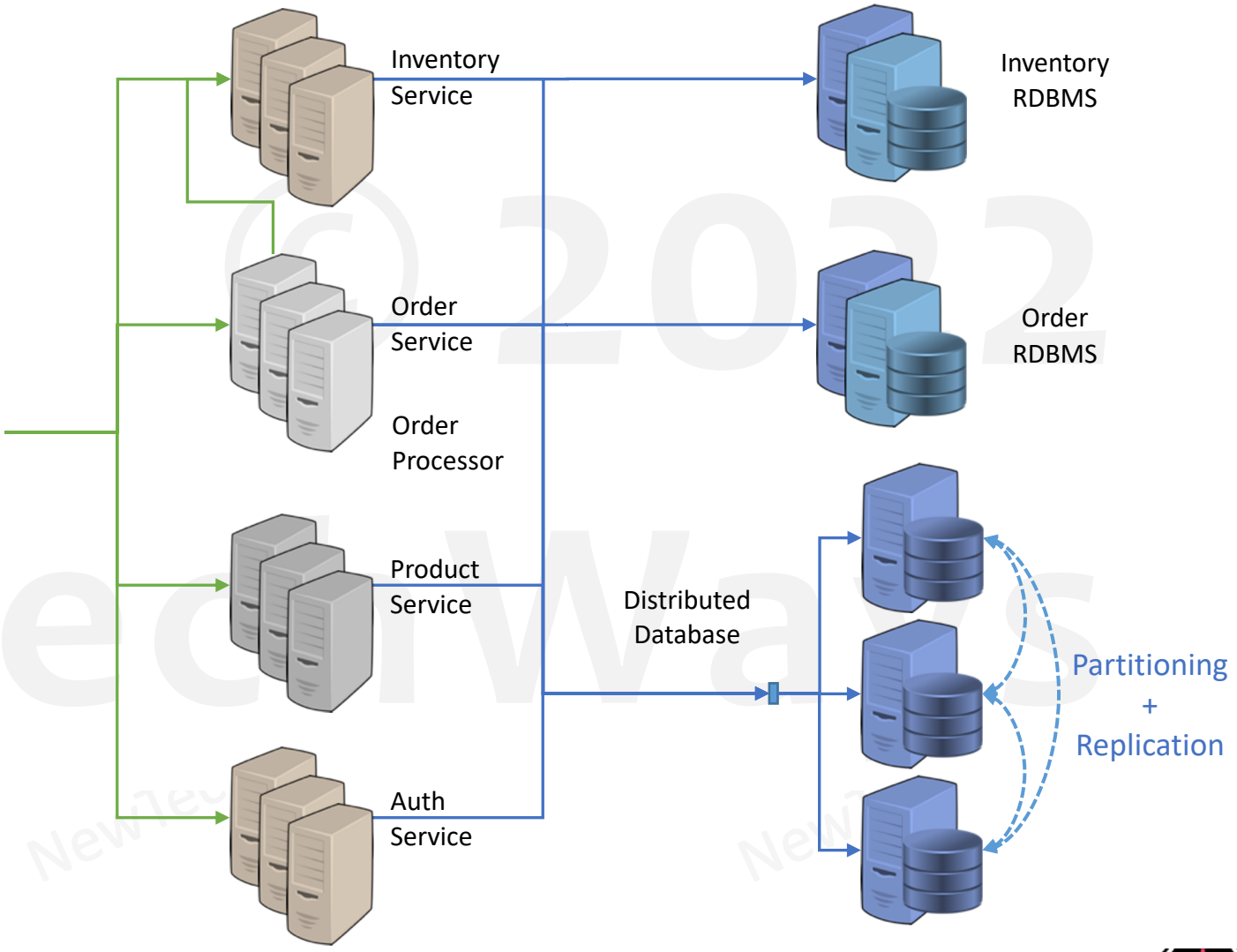
Partitioning – DB Record Level

- Data Partitioning
  - Record level
- Data Replication
- Routing request to serving instances
- Schema on demand
- Limited indexing
  - Puts limit on query patterns



Hybrid Approach

- RDBMS
  - For services that need strong consistency
- Distributed database
  - For services that can be eventually consistent



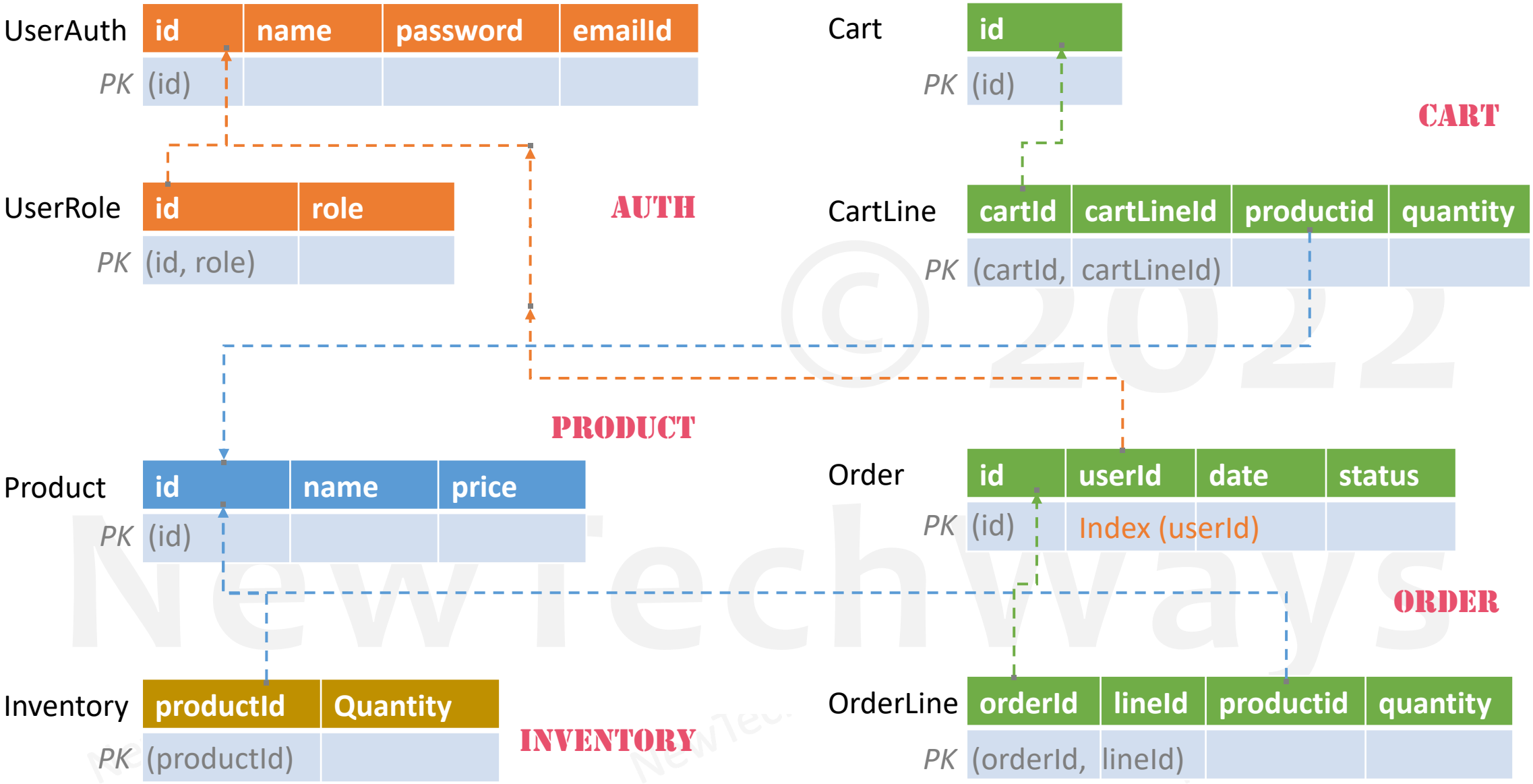
Partitioning Data Load

- MongoDB
  - Document Schema - JSON like
  - Hash Table like indexing
  - Master-Slave architecture
    - Single master for a record
- Cassandra
  - Column Family – Semi Structured Tables
  - Hash Table like indexing
  - Master-Master architecture - Write anywhere
    - Availability over consistency
- HBase
  - Column Family
  - B-Tree like indexing
    - Suitable for range queries
  - Master-Slave architecture

Partitioning Data Load

- MongoDB
  - Document type (b JSON) schema – Unstructured data
  - Hash Table like indexing
  - Master-Slave architecture
    - Write only on Master
- Cassandra
  - Table type schema – Structured data
    - SQL like CQL
  - Hash Table like indexing
  - Master-Master architecture
    - Write anywhere, no downtime
- HBase
  - Column Family type schema – Unstructured data
  - B-Tree like indexing
    - Suitable for analytics
  - Master-Slave architecture

NewTechWays



UserAuth

AUTH

id	name	password	emailId	role

Partition Key (id)

CartLine

CART

cartId	cartLineId	productid	quantity

Partition Key (cartId); *Sort Key* (cartLineId)

Product

PRODUCT

id	name	price

Partition Key (id)

Inventory

INVENTORY

productId	Quantity

Partition Key (productId)

OrderLine

ORDER

userId	id	orderId	lineId	productid	quantity	createdDate	createdTime	status

Partition Key (userId); *Sort Key* (orderId, lineId)

NewTechWays