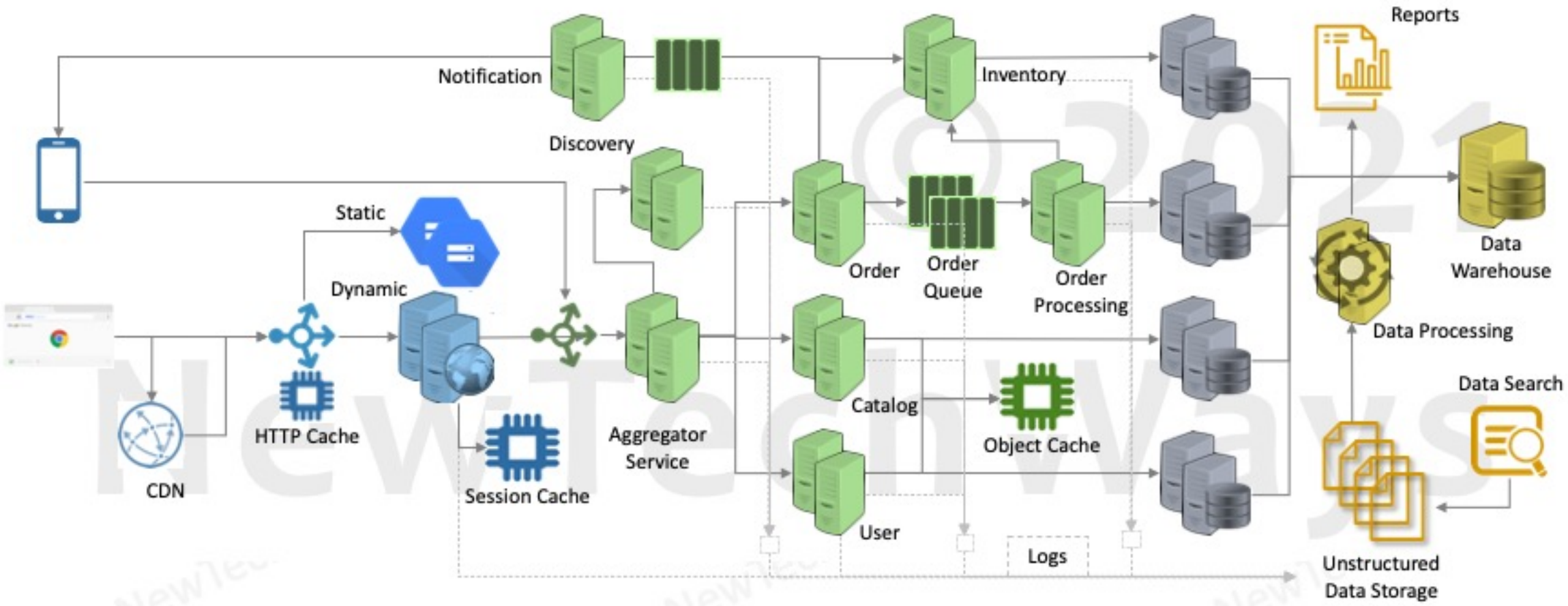
Architecting Solutions With Platforms & Frameworks

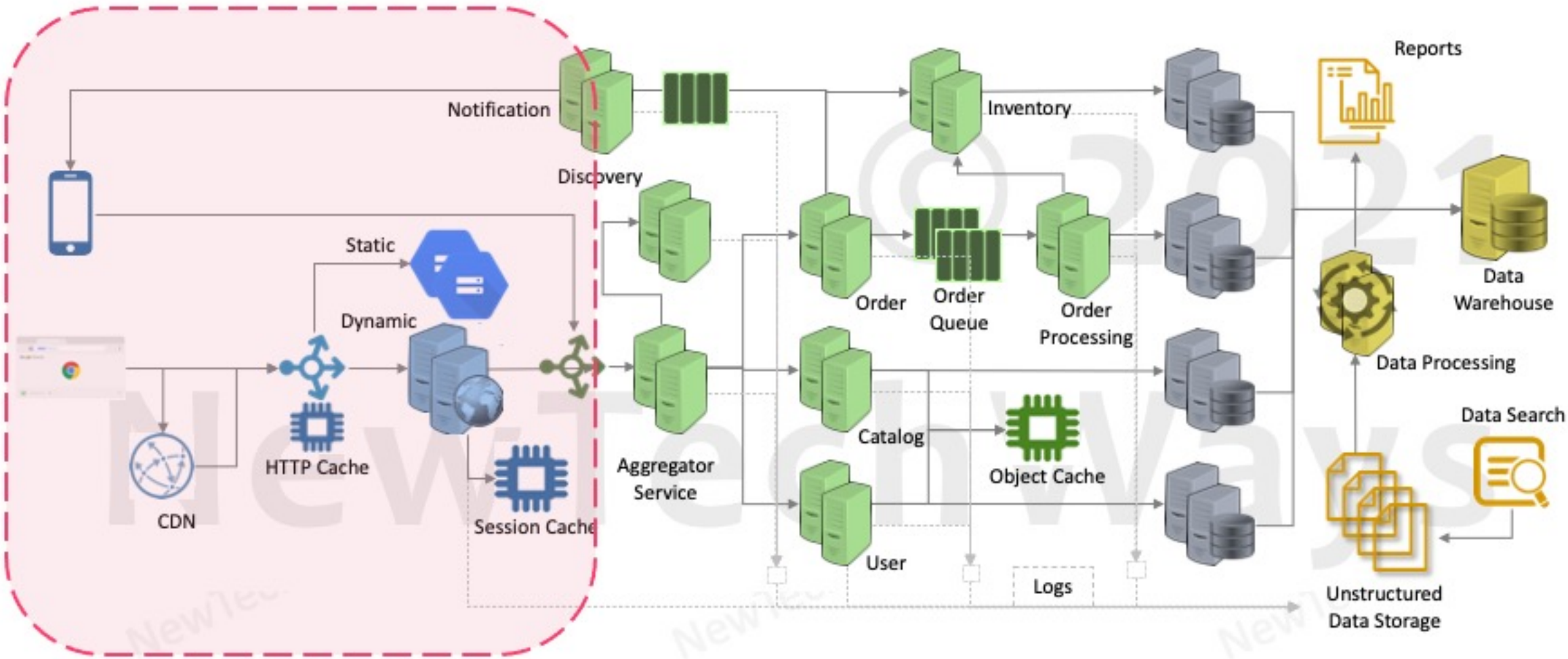
- Platforms for
  - Web Apps
  - Services
  - Datastores
  - Analytics
- Platform Functionality
- Platform Architecture
  - Performance
  - Scalability
  - Reliability
- Platform Use-Cases
- Platform Alternatives
  - Comparison
- Architecting Solution
  - End-to-End

Reference System



NewTechWays

Web Applications



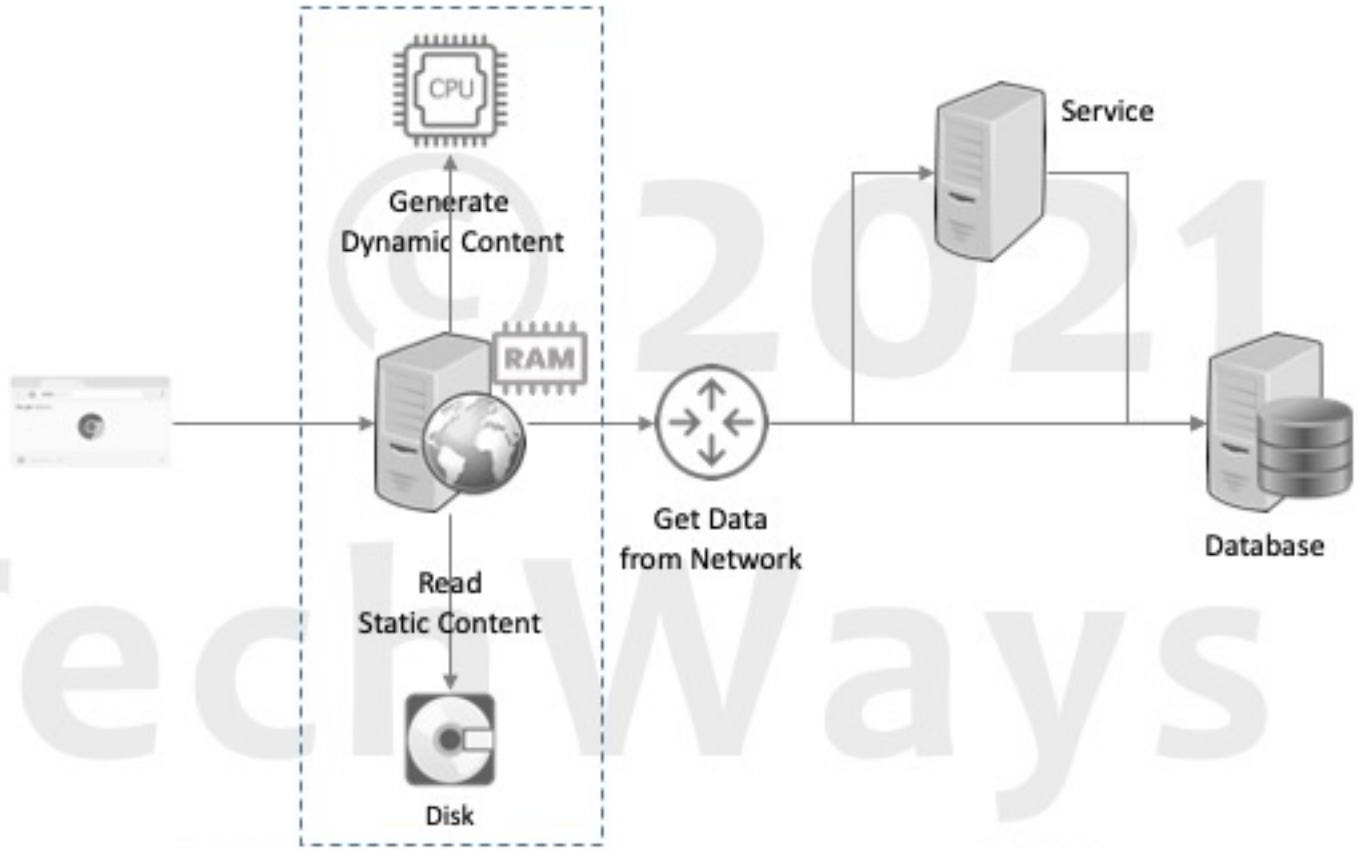
NewTechWays

Web Application – Solutions

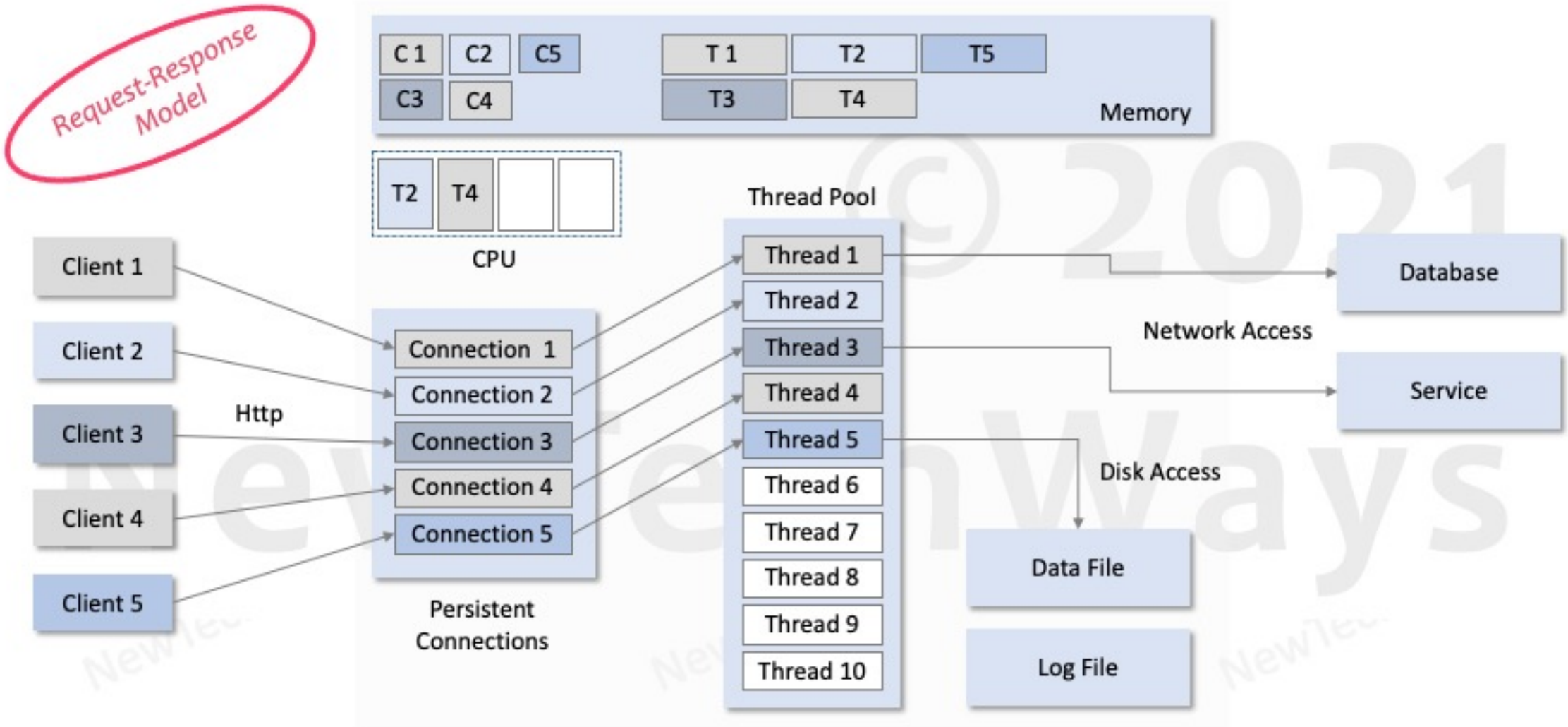
- Static Content
  - Apache Web Server
  - Nginx Web Server
  - Cloud Storage
- Dynamic Content
  - Web Server – Apache HTTPD, NodeJS
  - Java Web Containers – Tomcat, Jetty, Spring-Boot
- Content Caching
  - Nginx
- Content Distribution
  - CDN

Apache Webserver

- Store Static Content
  - Html/CSS/JS files
  - Image files
  - Documents
- Generate Dynamic Content
  - Get data & generate pages dynamically
  - PHP, Python, Perl
  - No JSP/Servlets
- Act as a Reverse Proxy
  - Not great



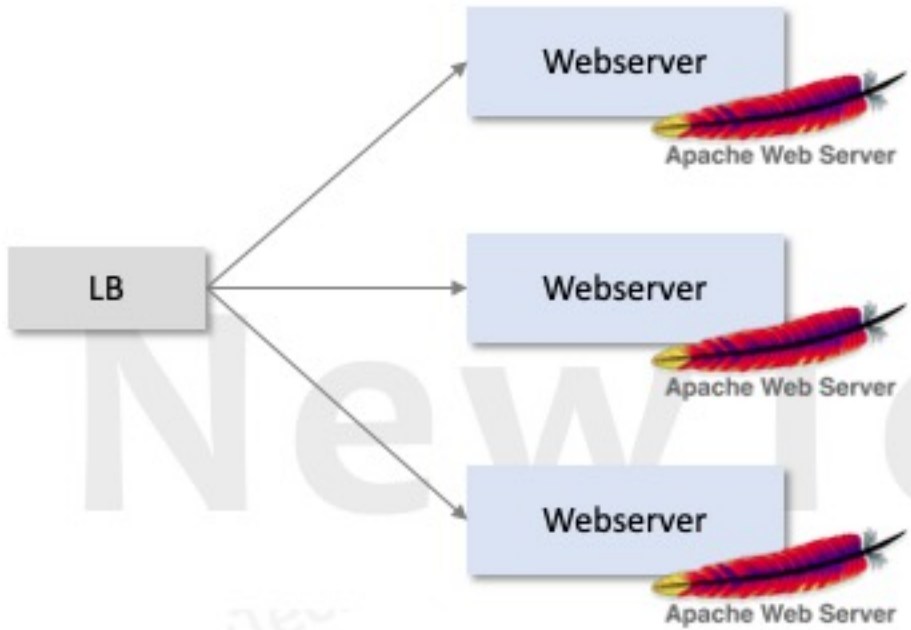
Apache Webserver Architecture



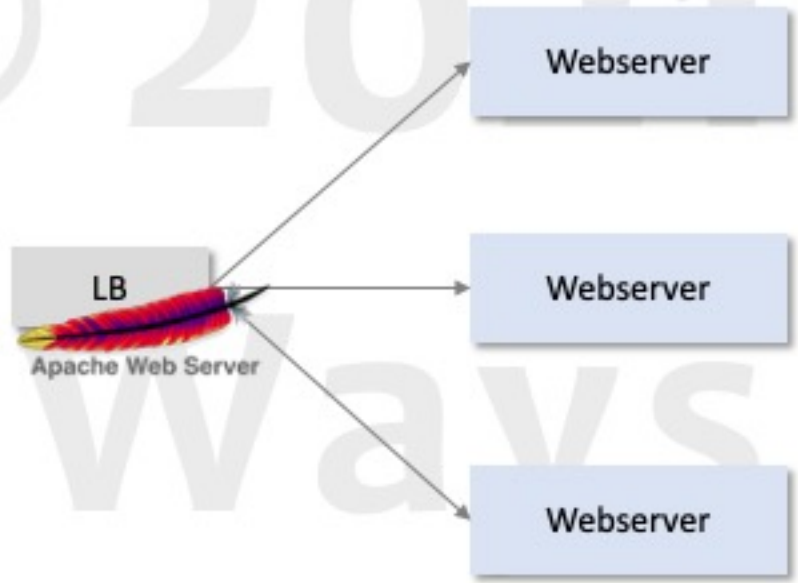
NewTechWays

Apache Webserver Scalability

Apache as Webserver
(CPU Bound)

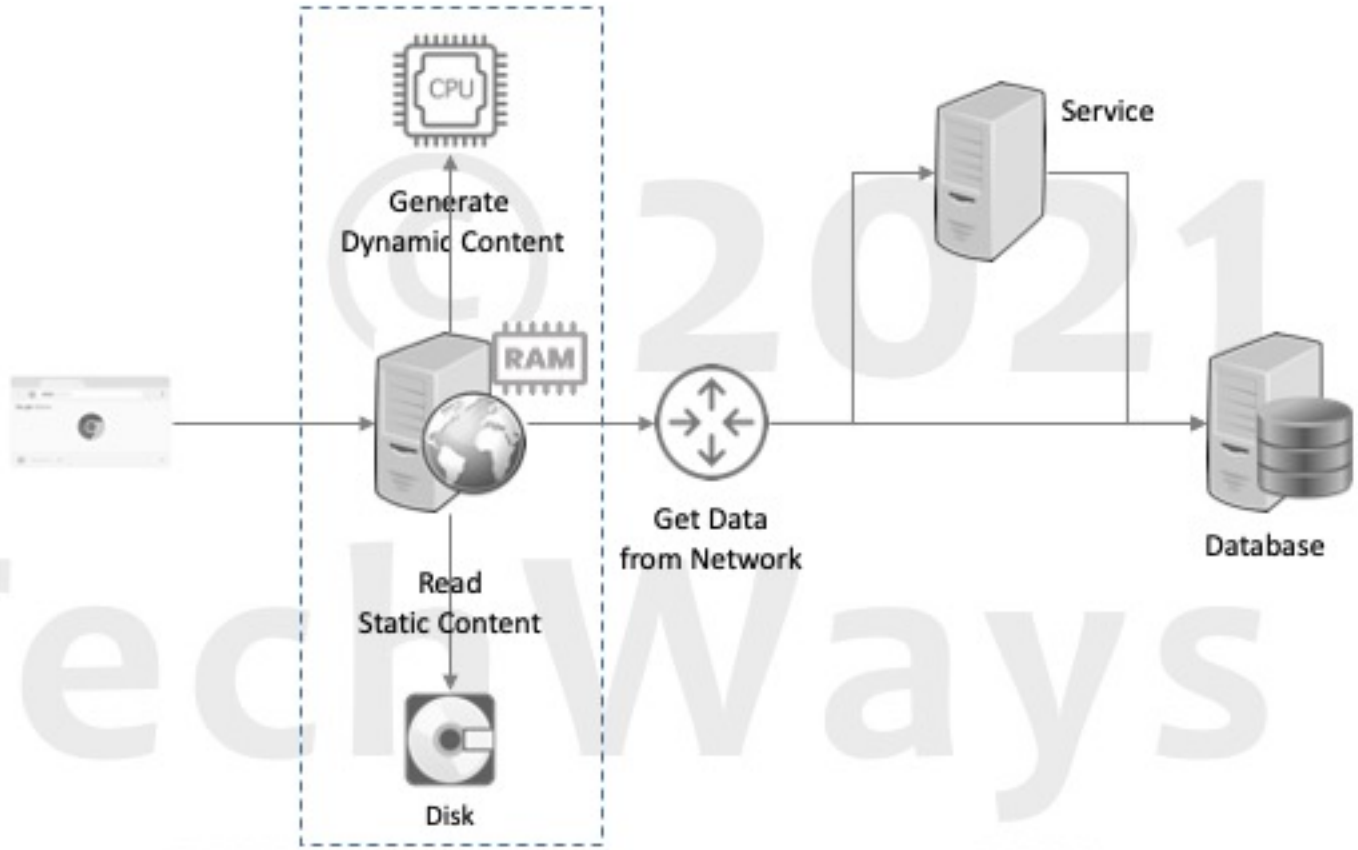


Apache as Reverse Proxy
(IO Bound)

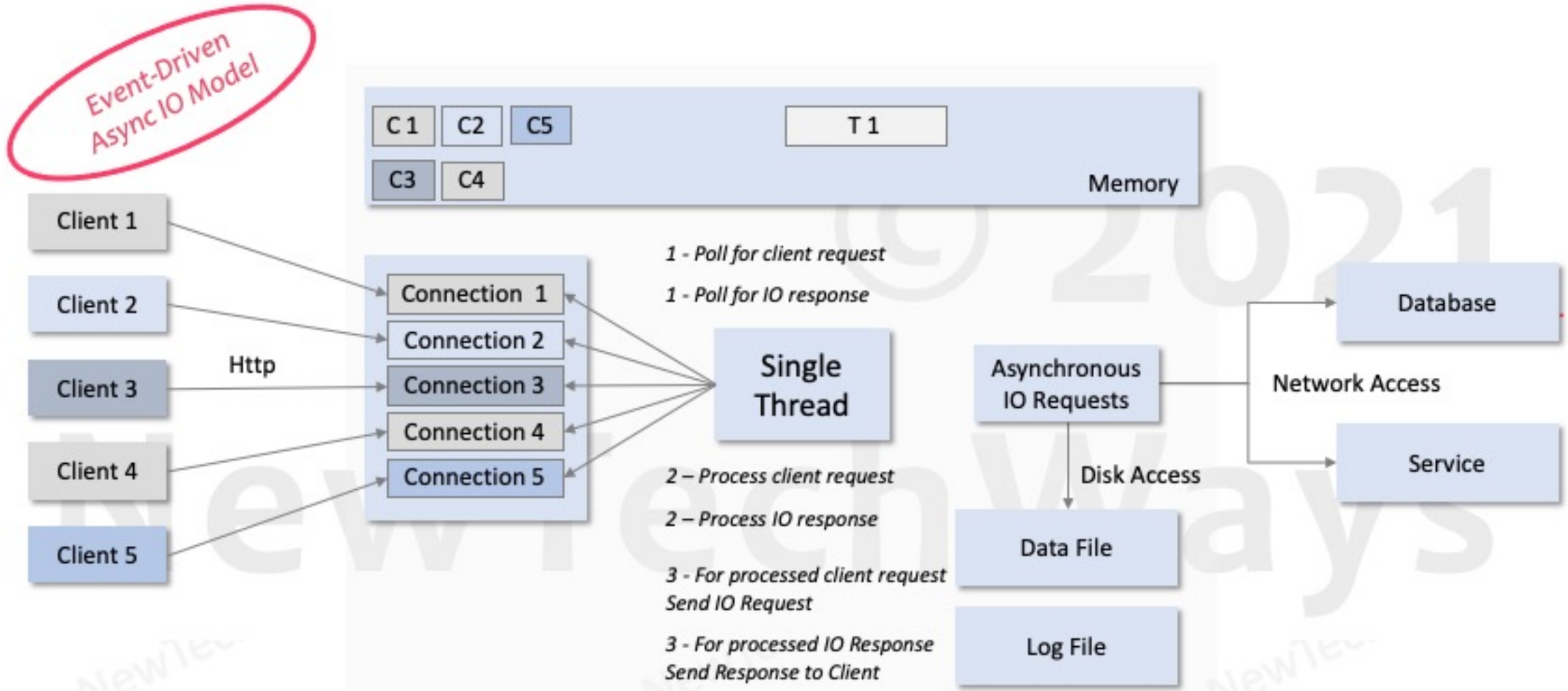


NGINX Webserver

- Store Static Content
  - Html/CSS/JS files
  - Image files
  - Documents
- Generate Dynamic Content
  - Not the best
- Act as a Reverse Proxy
  - Excellent
- Cache Content
  - Good

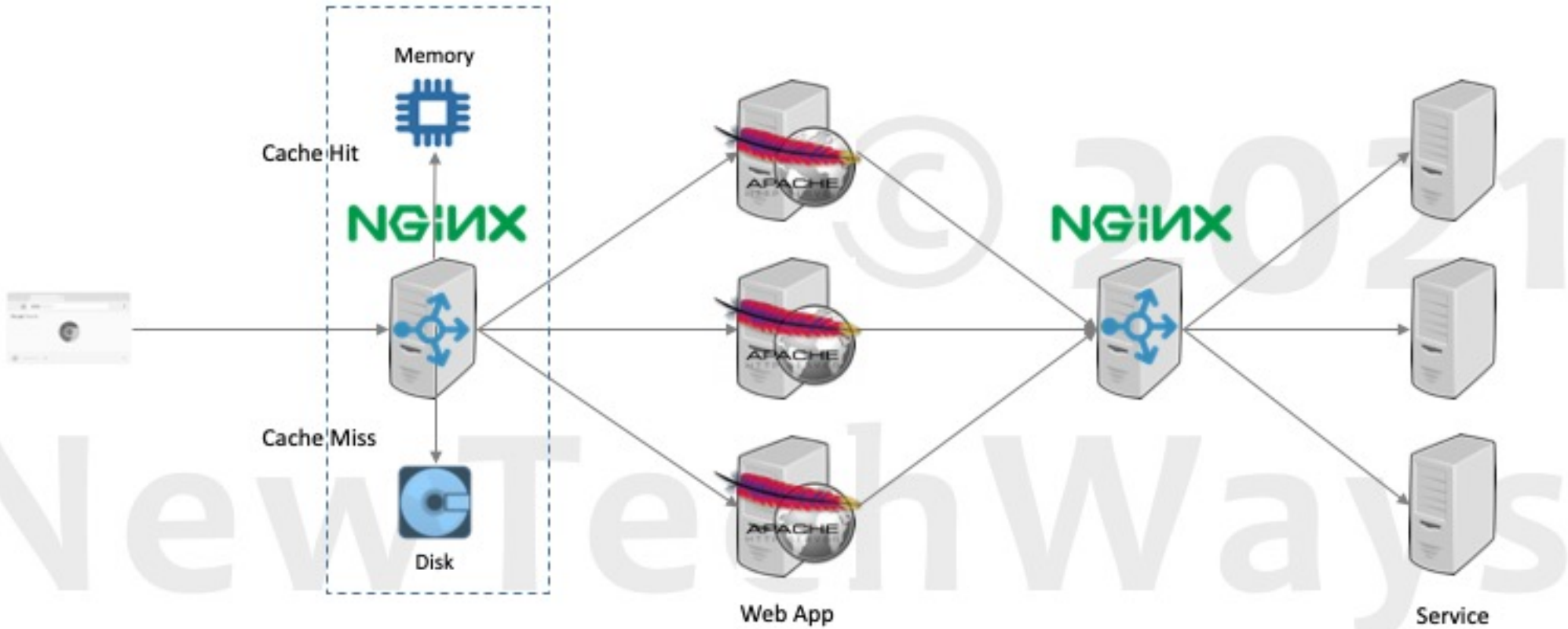


Nginx Architecture



NewTechWays

Nginx as Reverse Proxy & Cache



NewTechWays

Web Containers & Spring Framework

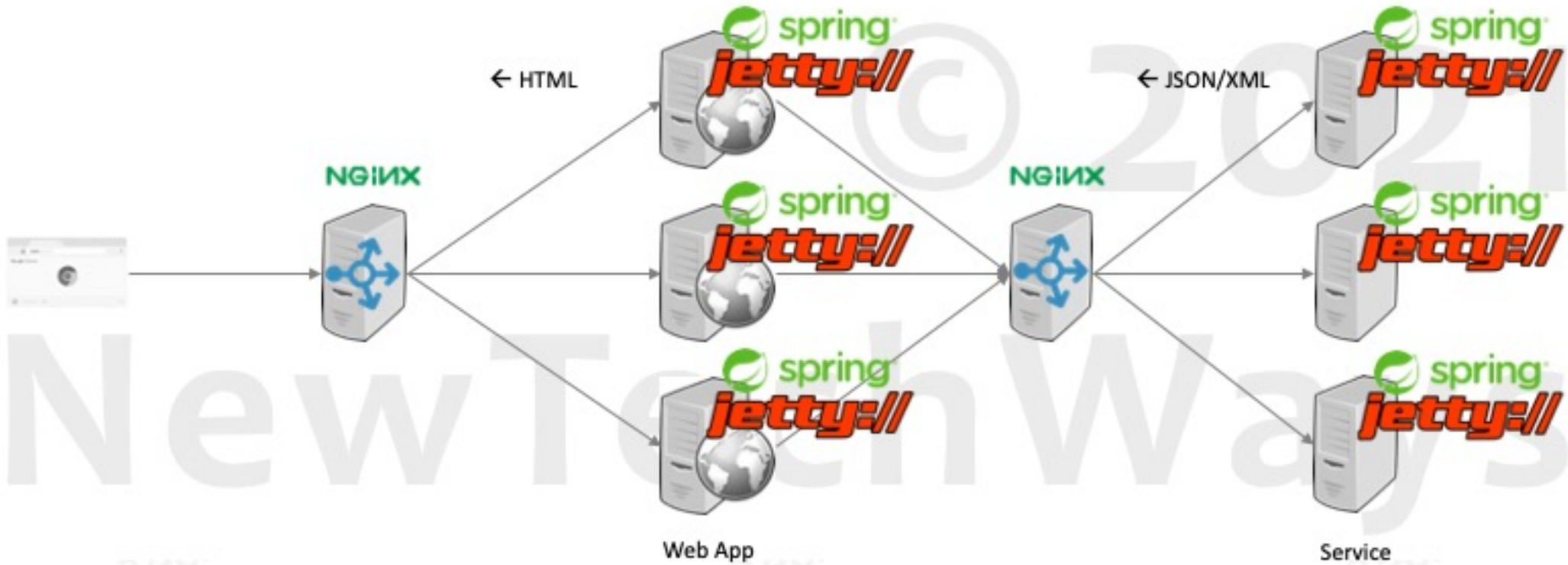
- Dynamic content using Java
  - OO Language for complex logic
- Web Containers
  - Tomcat
  - Jetty

  Servlet Engine
- Application Servers
  - Wildfly/JBoss
  - Weblogic
  - Websphere

  Servlet Engine, EJB Container, Session Clustering, Connection Pools, Caching, JMX, JMS, OSGI, ...
- Spring Boot
  - Runs embedded web container
    - Tomcat, Jetty

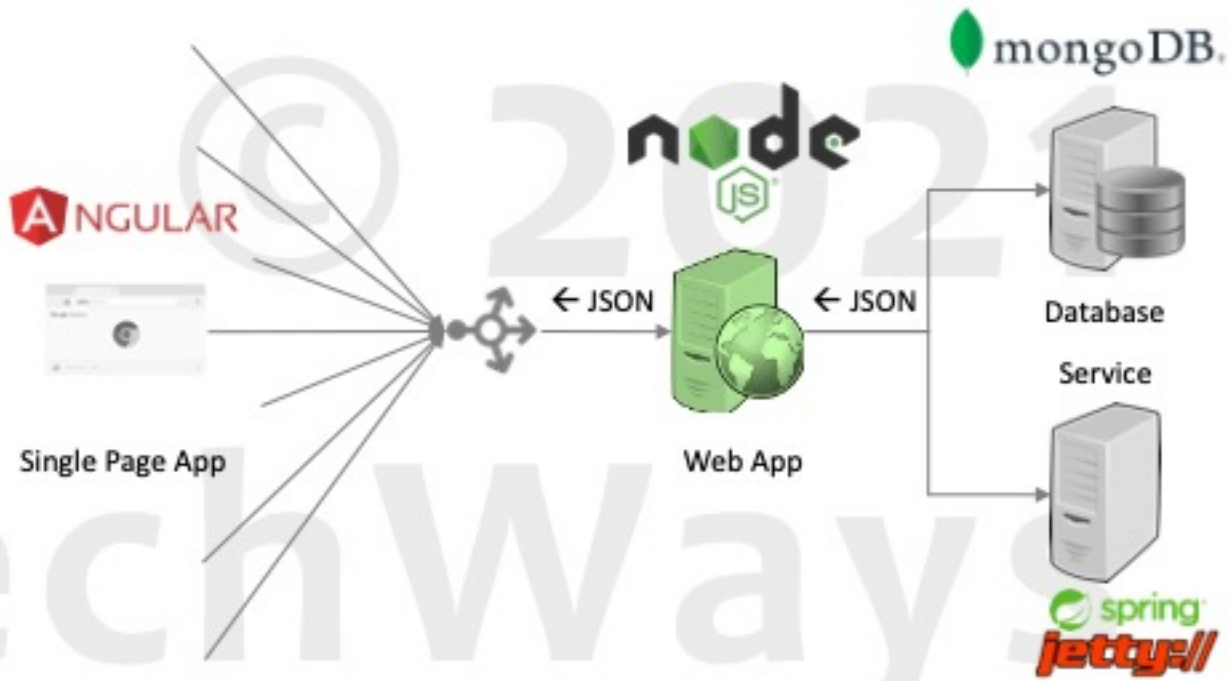
- MVC Architecture
  - Servlets for logic
  - JSP for presentation
- Spring Containers
  - Runs inside a Web container
  - Provides
    - IOC/DI
      - For business logic
    - Model View Controller
      - For frontends
    - JDBC Templates
      - For accessing DB
    - Connection Pools
      - Http, DB

Jetty & Spring

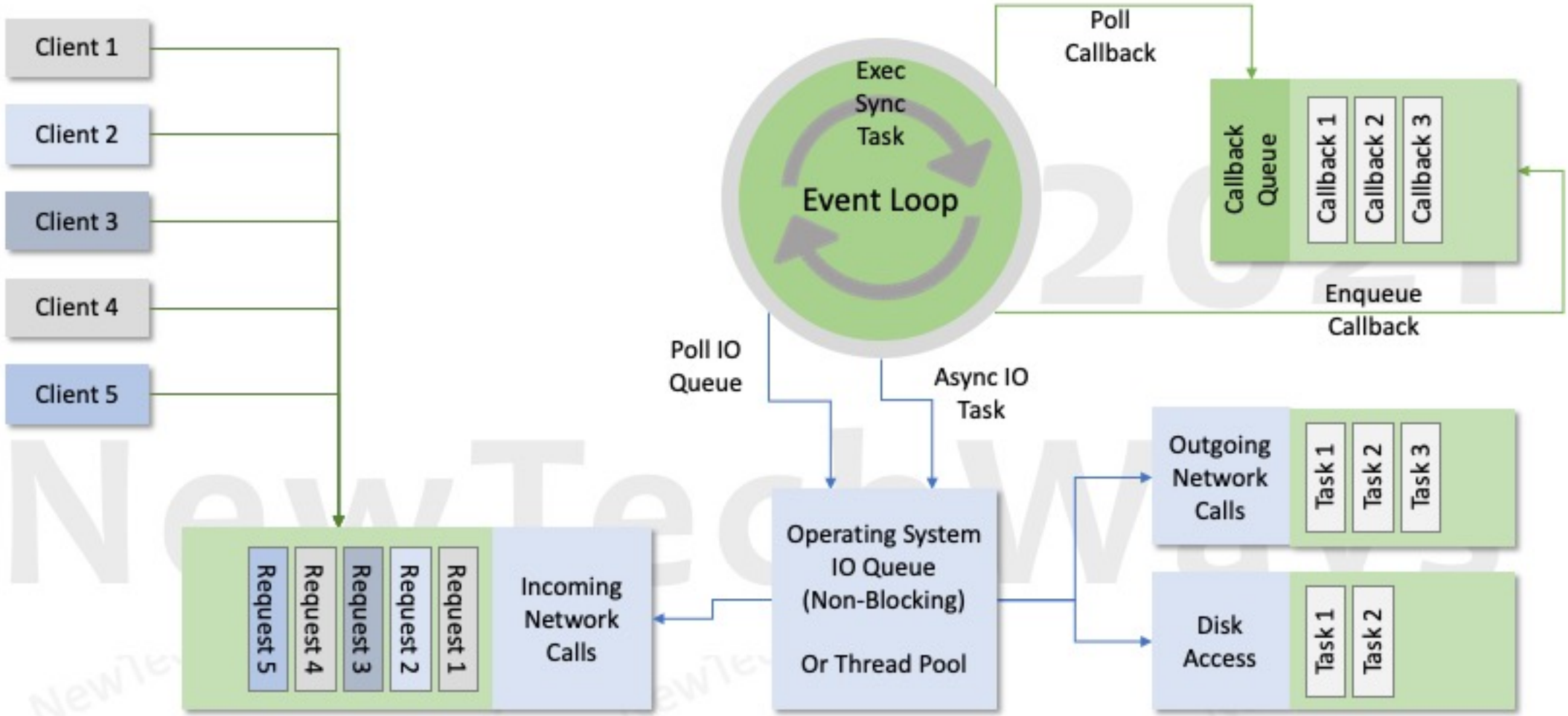


Node.js

- HTTP server
  - Uses JavaScript engine to process requests
- Highly efficient at handling a large-number of connections
  - For requests that are IO bound and not CPU bound
- Single thread to handle all connections
  - Saves memory
  - Avoids context switching



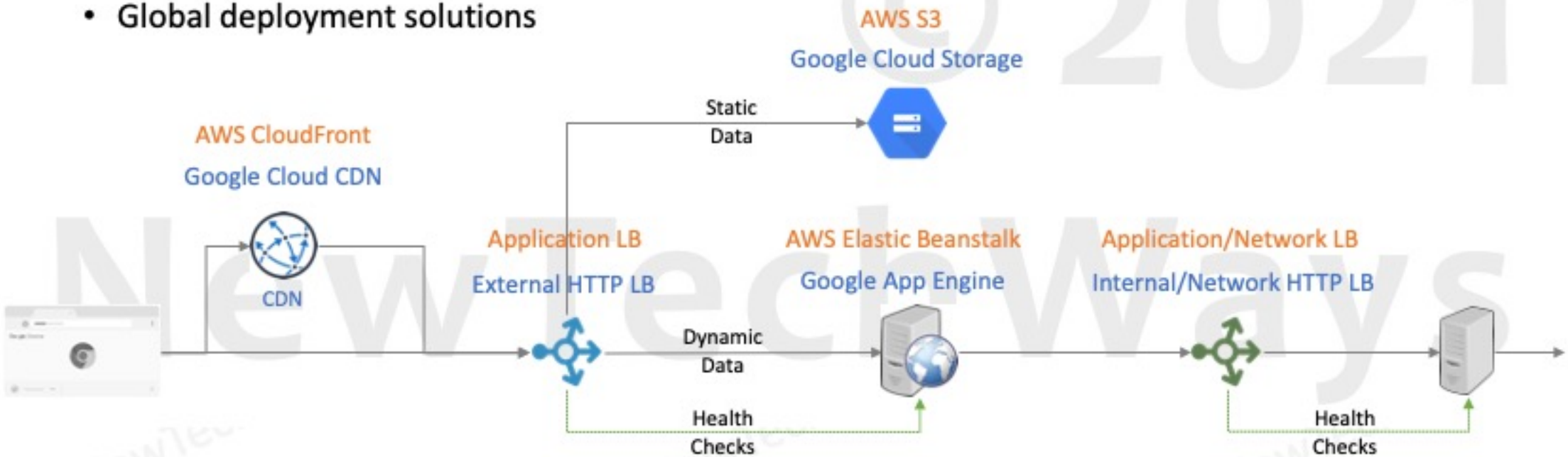
Node.js – Event Loop



NewTechWays

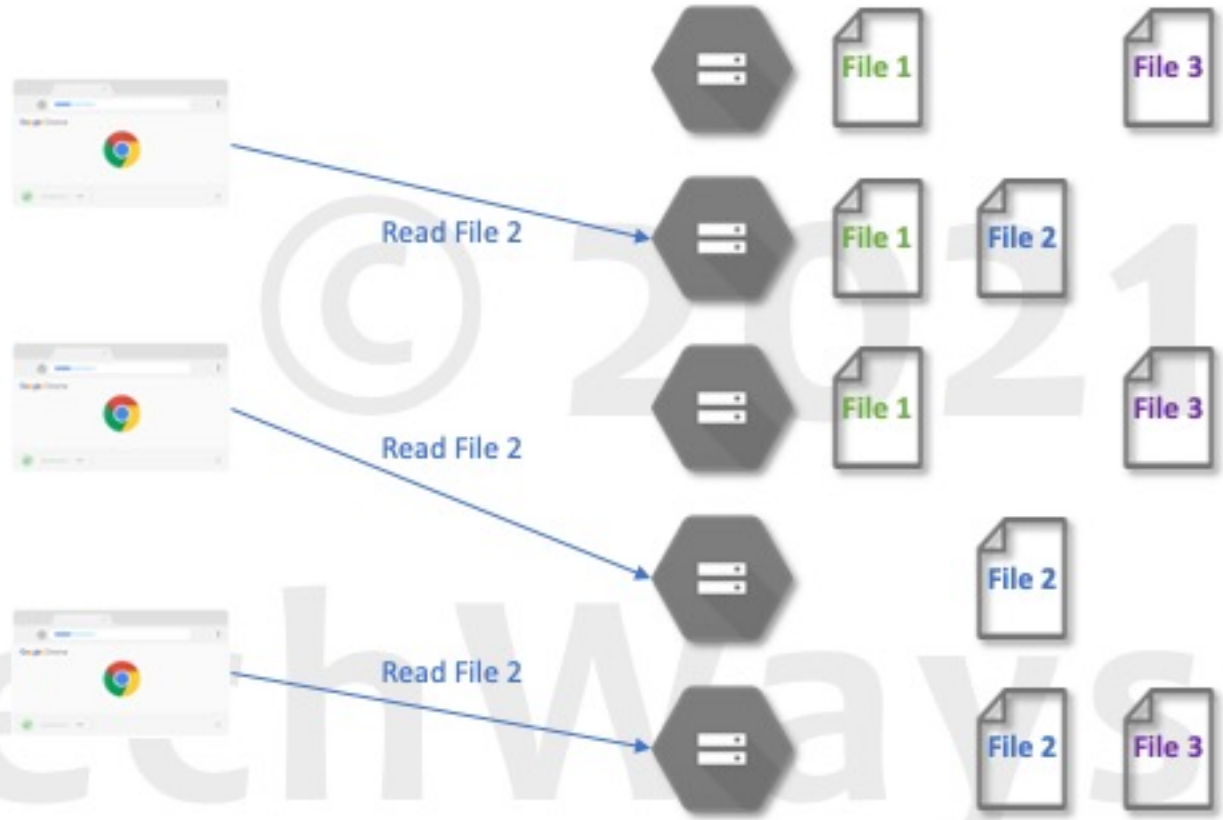
Cloud Solutions For Web

- Managed Services
  - Hardened solutions
  - Automated deployment
  - Built-in scalability & reliability
  - Global deployment solutions



Cloud Storage

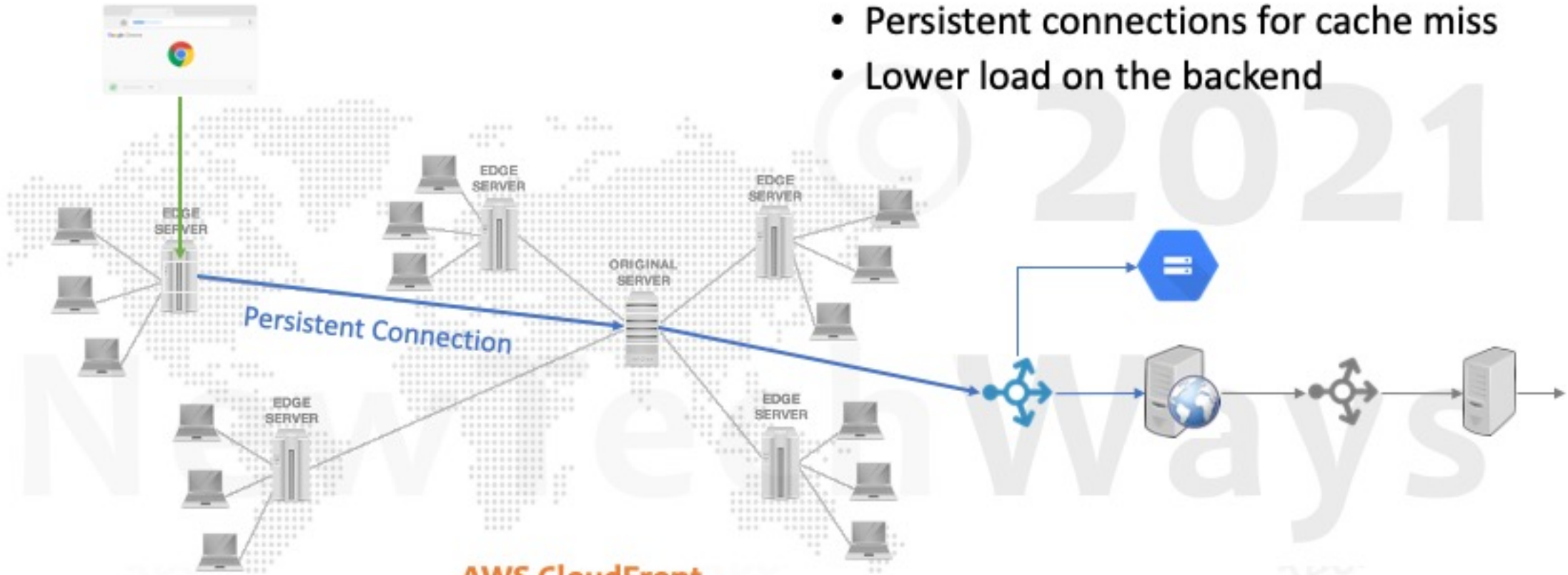
- Unlimited Disk Space
- Version Control
- Access Control
- Low Latency
  - No overhead of Directory structure
- High Throughput
  - Parallel clients
  - Large files can be broken into smaller chunks for parallel read
- High Availability & Reliability
  - Multiple copies
  - Multiple physical locations
- Static Website Creation



NewTechWays

Cloud CDN

- Low latency local access for cache hits
- Persistent connections for cache miss
- Lower load on the backend

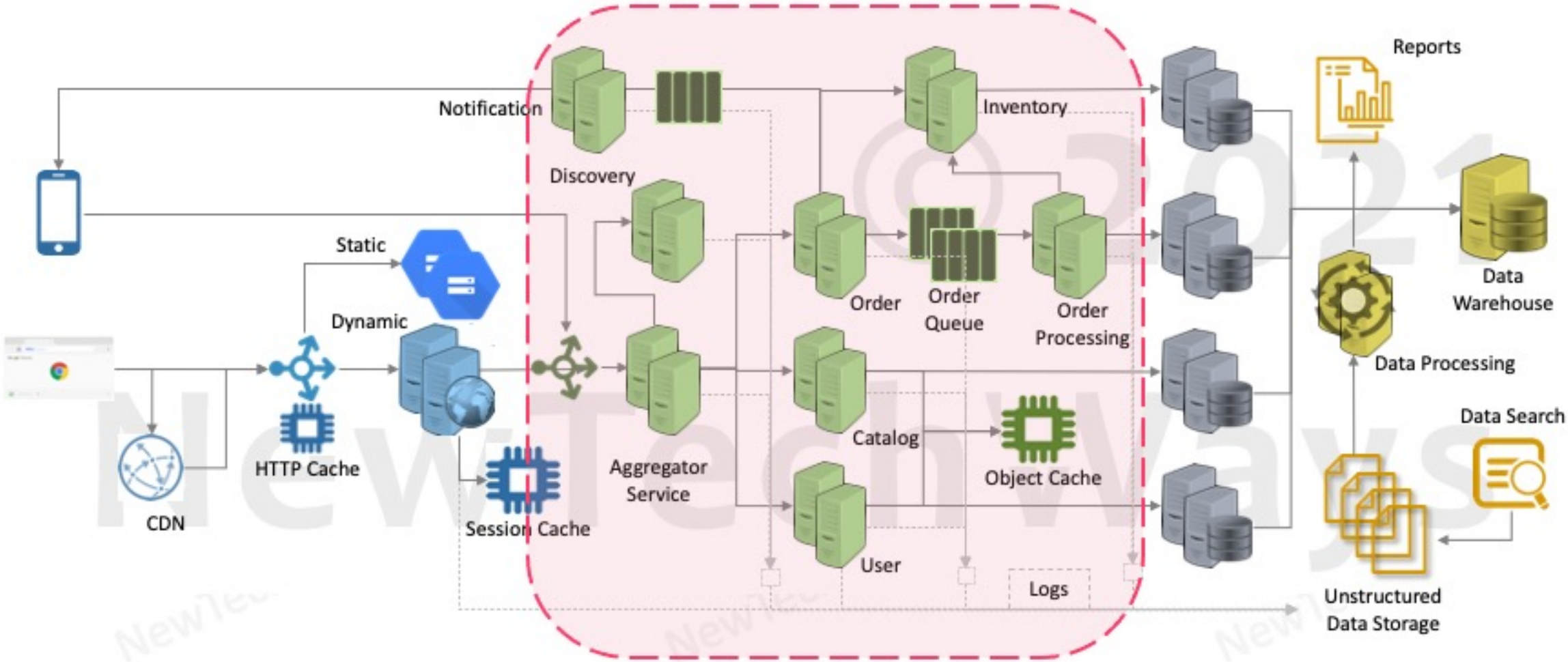


AWS CloudFront

Google Cloud CDN

NewTechWays

Services



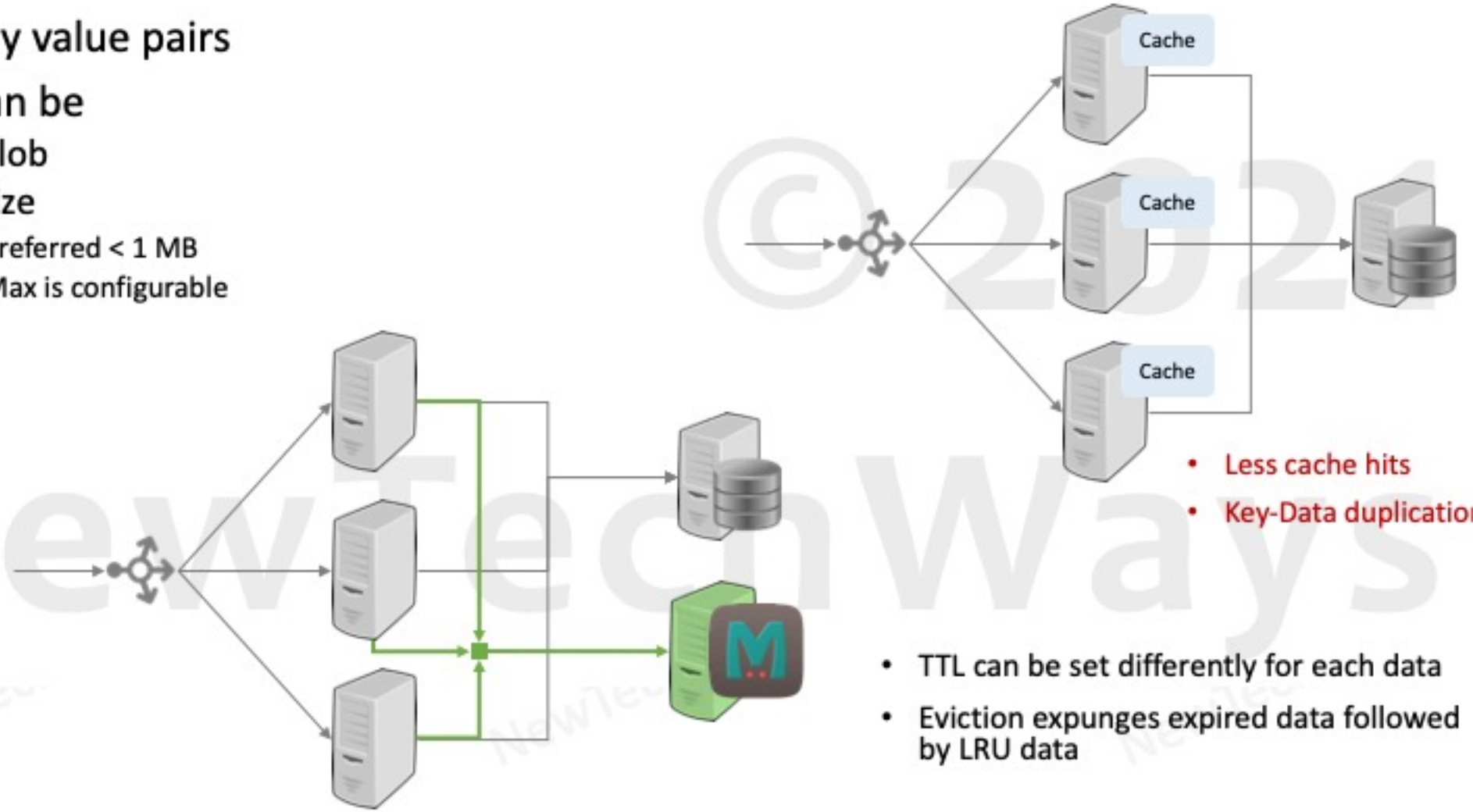
NewTechWays

Services – Solutions

- Web Containers
  - REST & Spring Containers
- Object Caching
  - Memcache
  - Redis
- Asynchronous Messaging
  - Redis
  - RabbitMQ
  - Kafka
- Service Mesh
  - Netflix
  - Istio

Memcached

- Stores key value pairs
- Values can be
  - Any blob
  - Any size
    - Preferred < 1 MB
    - Max is configurable

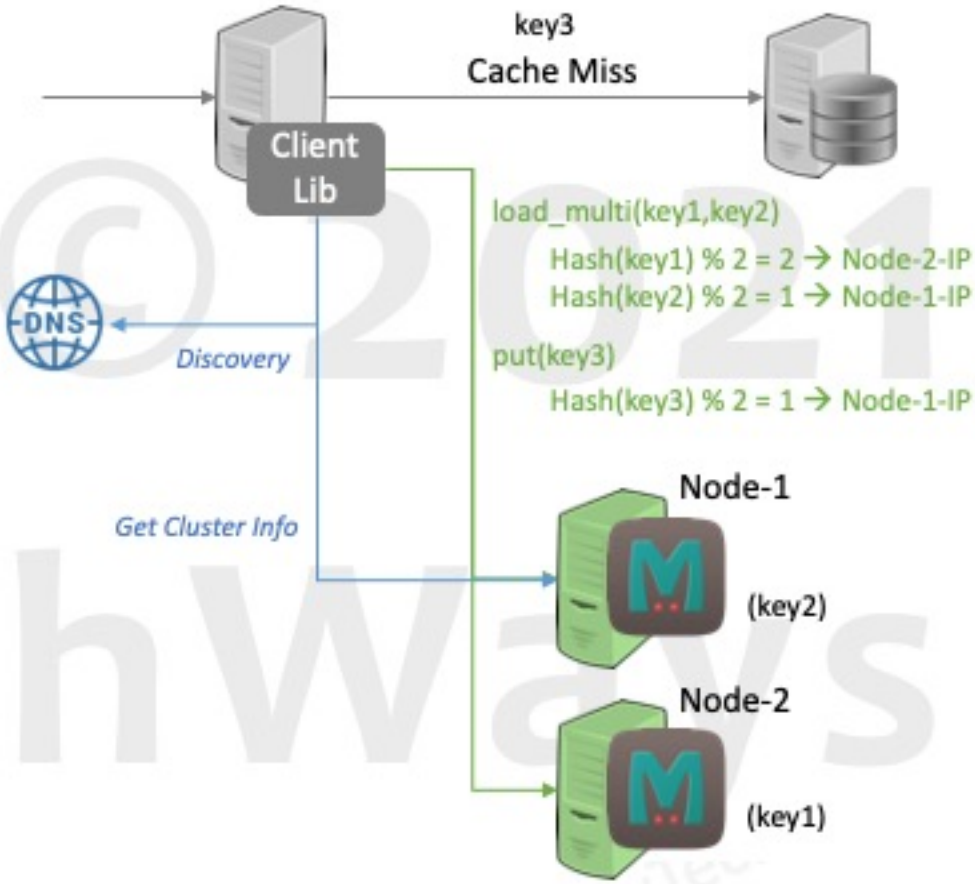


- Less cache hits
- Key-Data duplication

- TTL can be set differently for each data
- Eviction expunges expired data followed by LRU data

Memcached Architecture

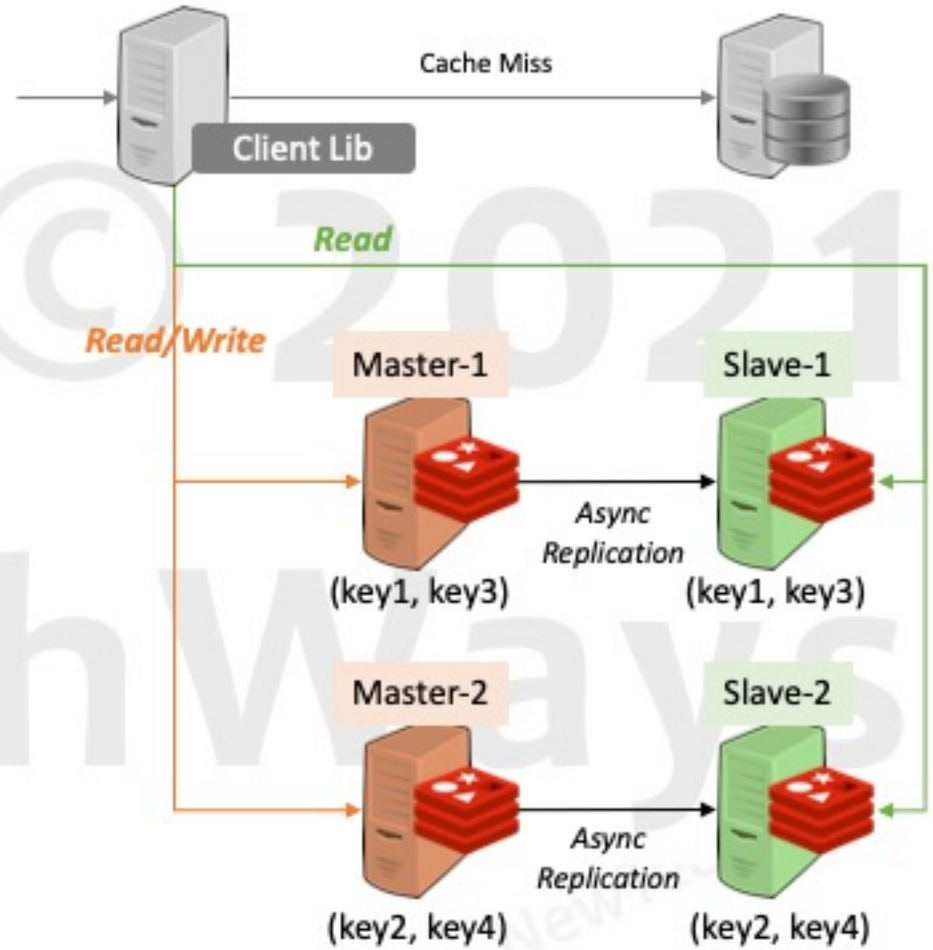
- Cache-aside pattern
- Sub-millisecond Latency
- Horizontally Scalable
  - Data is partitioned
- High throughput
  - Parallel operations
- Cluster aware client library
  - Consistent hashing for resolving a node
- Node failure is treated as a cache miss
  - Use large number of nodes with less data
- Data is lost if a node crashes or restarted



NewTechWays

Redis Cache

- Much like Memcached
- It is a [Key → Data structure] store
  - Strings, Lists, SortedSets, Maps, …
- Data-Store mode for Persistence
  - Stores data on a disk
  - Allows backups
  - Can be started pre-populated with a backup
- Data Replication
  - Asynchronous and synchronous
  - Read load distribution
  - High Availability
- Can also be used as a messaging queue
- Data-Store mode requires fixed number of nodes
  - Cache mode is suitable for node scaling



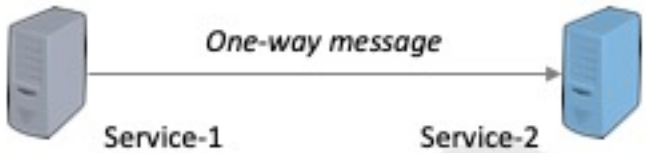
Cloud Caching Solutions

- AWS Elastic Cache
  - Memcached
  - Redis

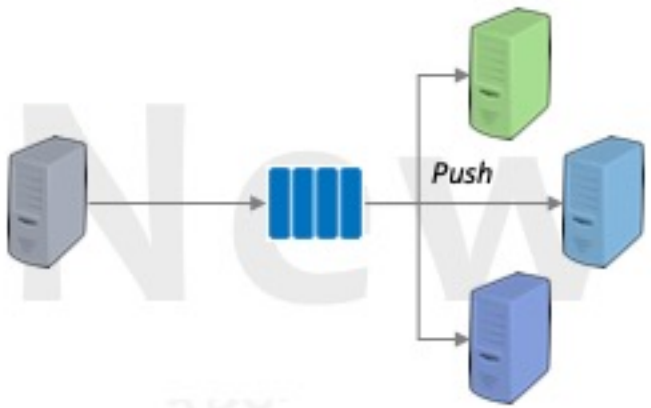
- Google Memorystore
  - Memcached
  - Redis

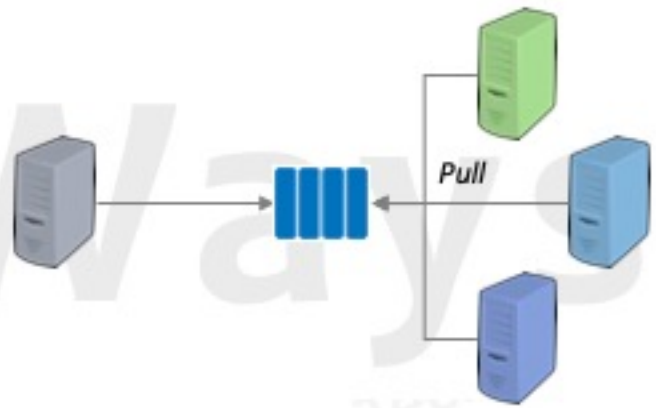
- Fully Managed
- Sub-millisecond latency
- Scalable to 5 TB
- Highly available (99.9%)

RabbitMQ



- Design Goals
  - At-Least-Once Delivery
  - Message Sequencing – FIFO
  - Interface Decoupling
  - Consumer Decoupling
  - Message Rate Decoupling
- Consumer Mode
  - Push
  - Pull
- Use Cases
  - Service Integration
  - Message Buffer





RabbitMQ

kafka

RabbitMQ

NewTechWays

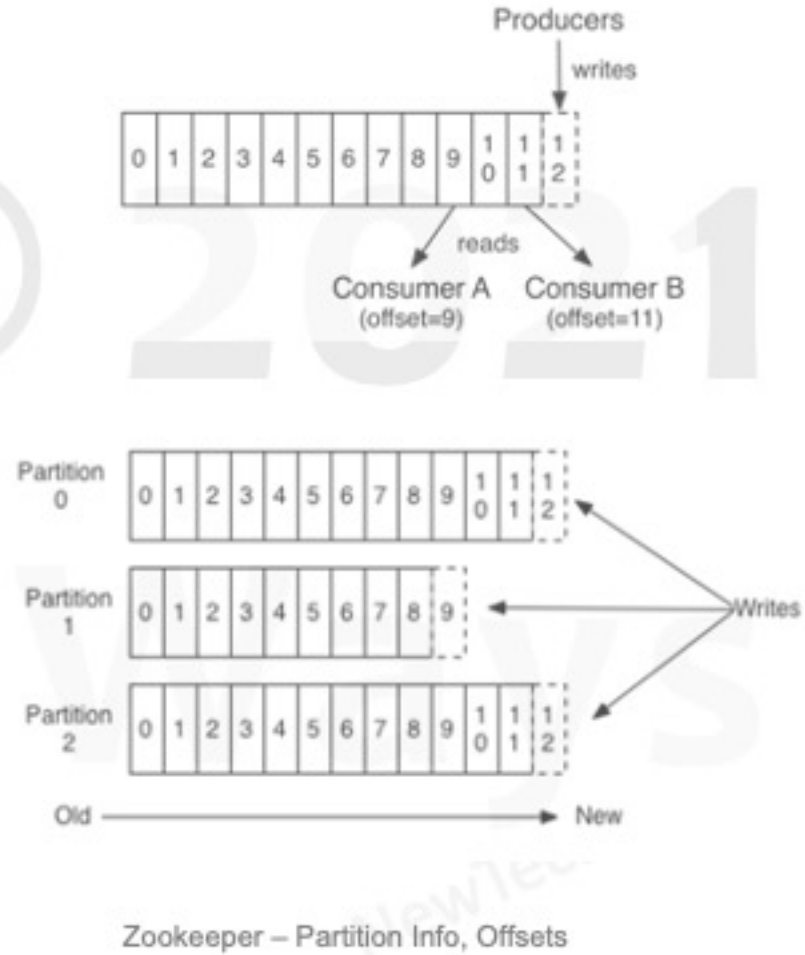
Rabbit MQ

- General purpose message broker
- Messages are pushed to consumers
- Message deleted once acknowledged
- Message ordering is guaranteed
- Useful for asynchronous service integration
- Both persistent and transient messages are supported
- Uses built-in component called exchange for routing

- Scales to 50K messages per second
- Scales well vertically
- Not horizontally scalable
  - Master-Slave replication for high availability
    - Clients can connect to any node
    - All publish, consume operations first go to master

Kafka

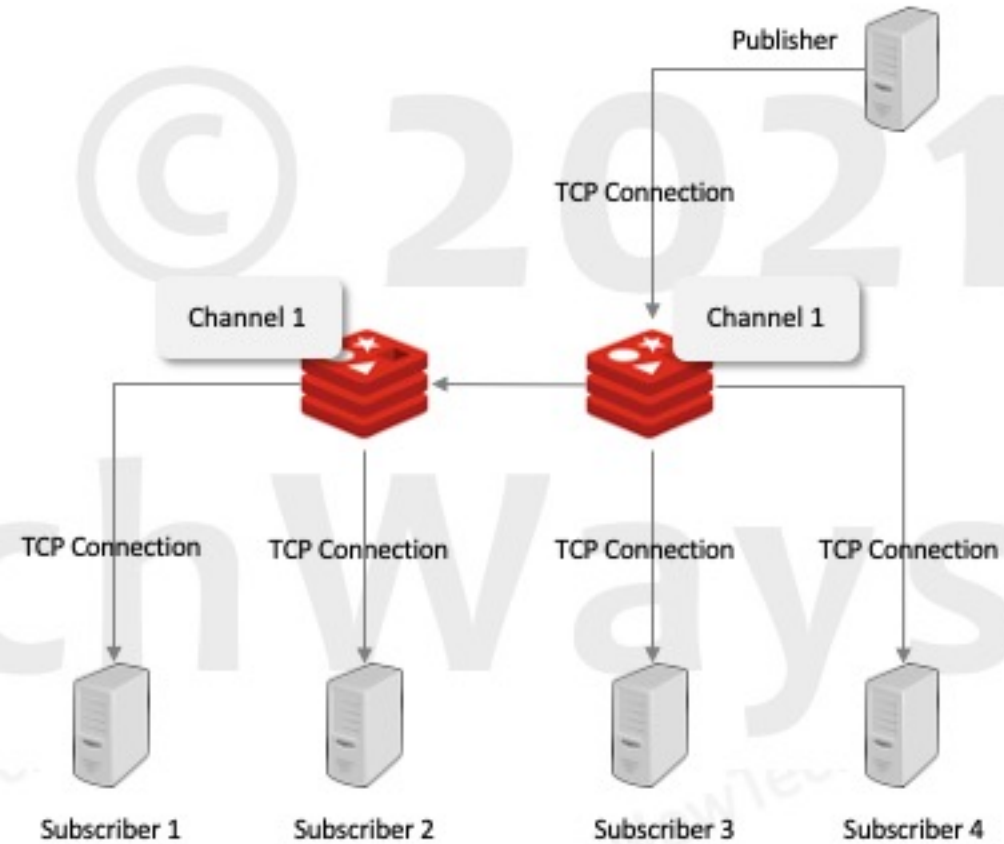
- Performance
  - Million messages per second
  - Sequential writes and reads on log files
  - Relies on page cache for quick reads
- Horizontally Scalable
  - Topics are partitioned
- Order of data guaranteed only within a partition
  - Producer can decide the partition
- Messages are not deleted, so can be replayed
- Consumers can only pull the data
  - No push by Kafka
  - Not designed for service integration
- Useful for streaming analytical workloads
  - Where high throughput is required
  - Click streams, Page views, Logging, Ingestion, Security



Zookeeper – Partition Info, Offsets

Redis Pub/Sub

- For short lived messages with no persistence
  - Much like a synchronous call
  - Fire and forget
    - No delivery guarantee
- Million operations per second
- Useful for making dashboards
  - Leaderboard
- Comparison
  - Kafka
    - No push
    - Writes to log
  - RabbitMQ Transient
    - Delivery acknowledgement
    - Deletion of delivered messages

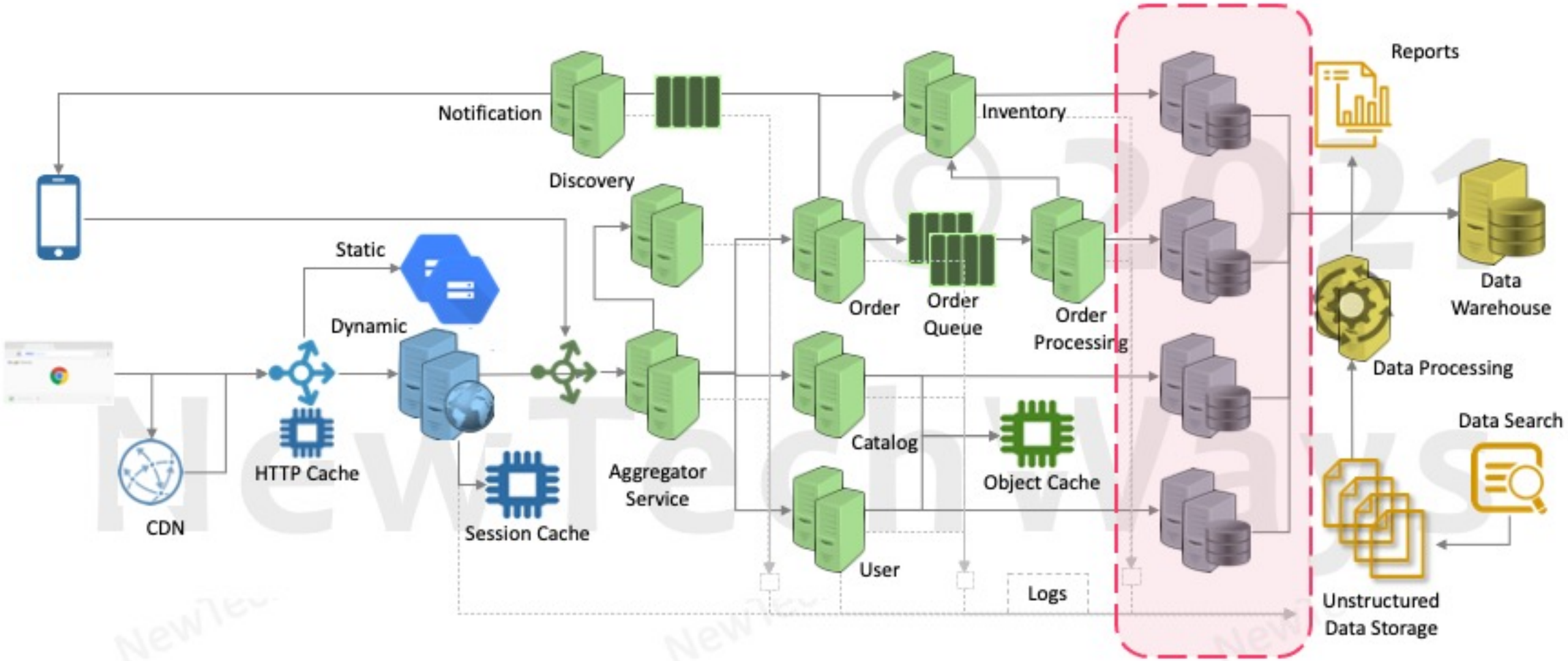


NewTechWays

Cloud MQ Solutions

Community	AWS	Google Cloud
Rabbit MQ 	SQS	Pub/Sub 
Kafka 	Kinesis	

Datastores



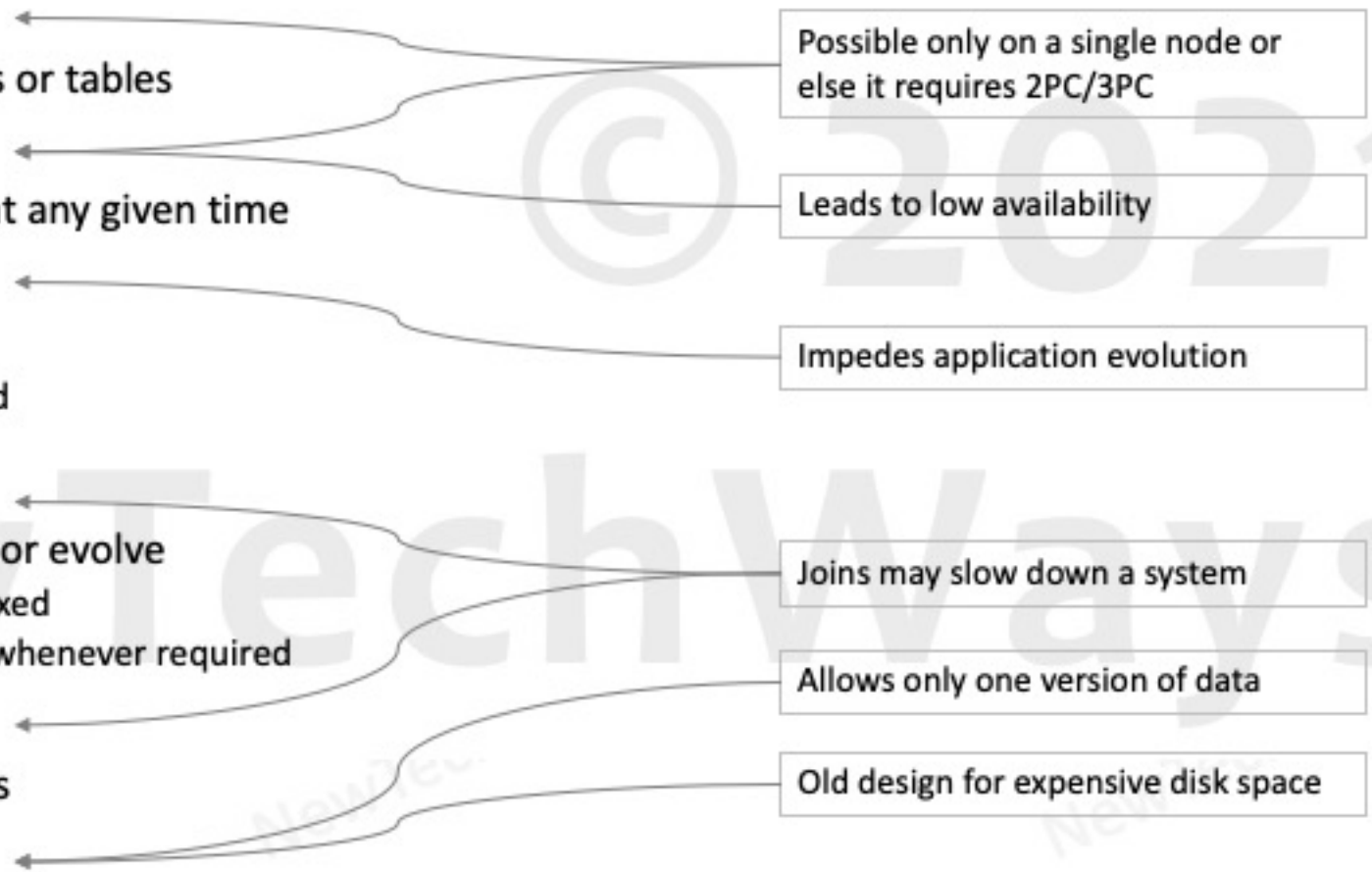
NewTechWays

Datastore – Solutions

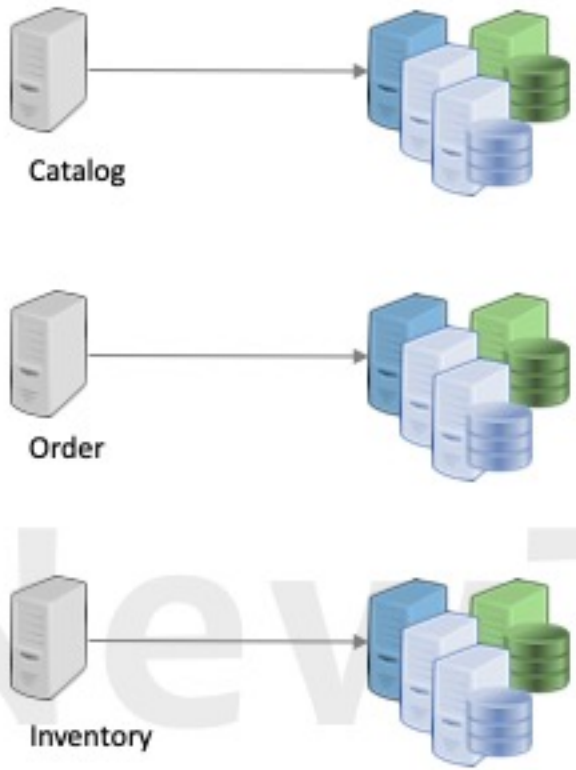
- RDBMS
  - Oracle
  - SQL Server
- Distributed Databases
  - Key-Value
    - Dynamo
  - Column Family
    - Big Table
    - Cassandra
    - HBase
  - Document Oriented
    - MongoDB

RDBMS

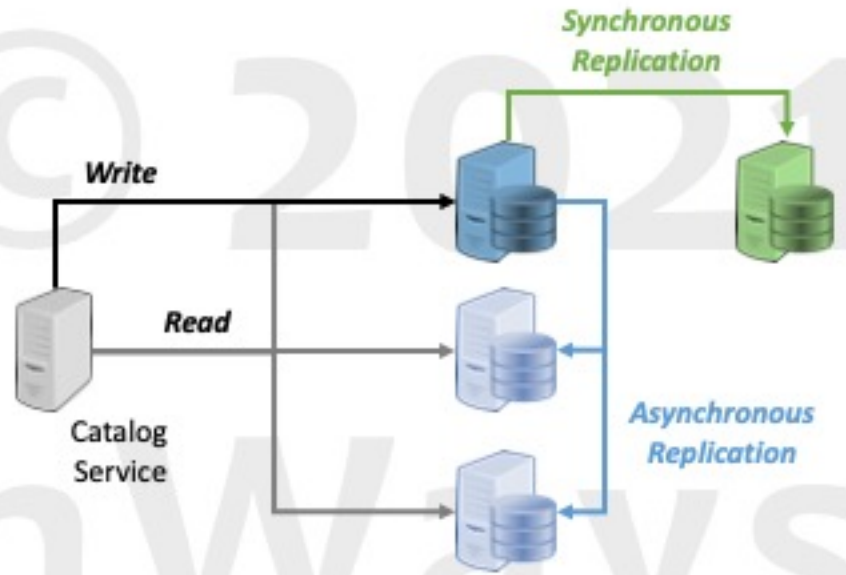
- General purpose database (< 1-5 TB data, 10K connections)
- ACID Transactions
  - Update of multiple records or tables
- Data Consistency
  - Data same for all readers at any given time
- Fixed Schema
  - Data analysis
    - Columns can be selected
    - Rows can be filtered
    - Queries can Join Tables
  - Query pattern can change or evolve
    - Any column can be indexed
    - Indexes can be created whenever required
- Normalized Data
  - Efficient storage and writes
- Overwrite for updates



RDBMS Scalability Architecture



Vertical Partitioning for Scalability



Replication for Reliability/Availability

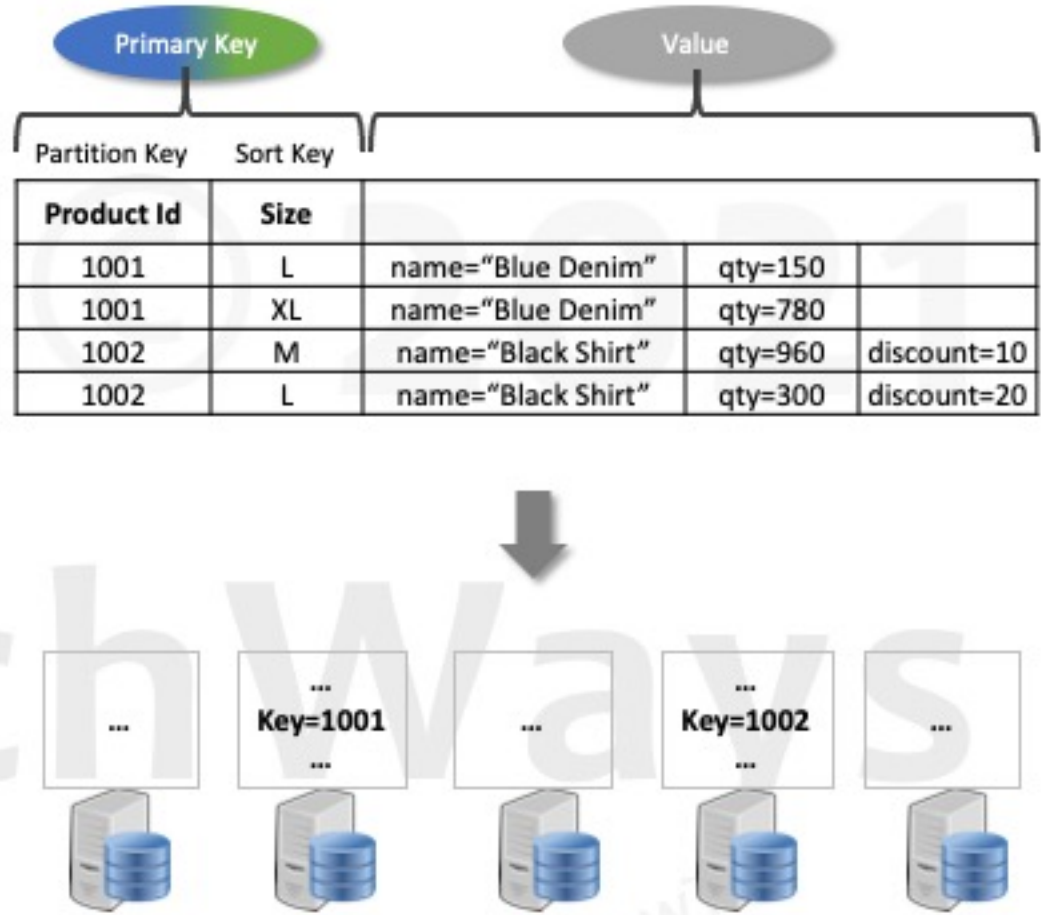
NewTechWays

NoSQL Objectives & Trade-Offs

Objective	Architectural Choice	Trade-Off
Scalability	Horizontal Partitioning, Commodity Hardware	ACID Transactions, Joins
Availability	Data Replication, Eventual Consistency	Data Consistency
Flexible Schema	Key-Value, Column Family, Document-Oriented	SQL, Secondary Indexes, Integrity Constraints
Performance	Aggregate Schema, In-Memory R/W	Normalization, Non-Key Queries

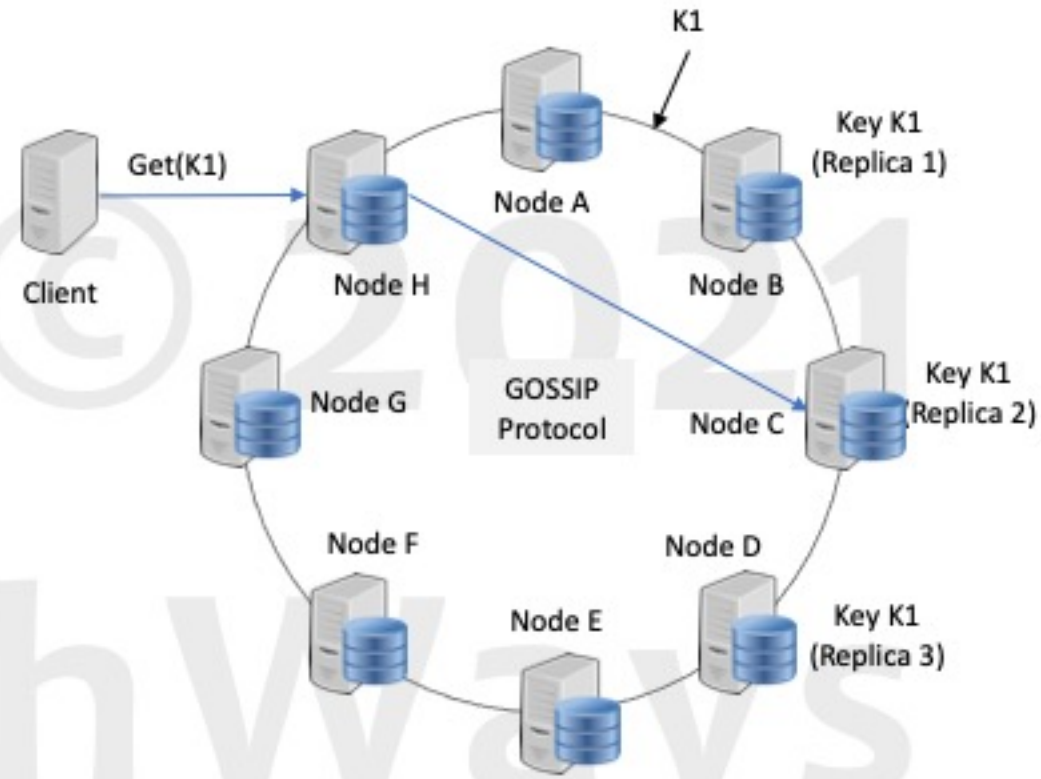
Amazon DynamoDB

- Key-Value Pair Datastore
- Used for High Scalability & Availability
- Table is a Hash-Map
  - Persistent
  - Distributed
- API
  - Put, Get, Update, Delete, Query
- Index Key – has two parts
  - Partition Key
    - Can have only one attribute
    - Key is hashed to determine the partition
  - Sort Key
    - Determines sort order of an item within a partition
    - Can be used for range query <, >, like
- R/W ops for a key are atomic



Dynamo DB Architecture

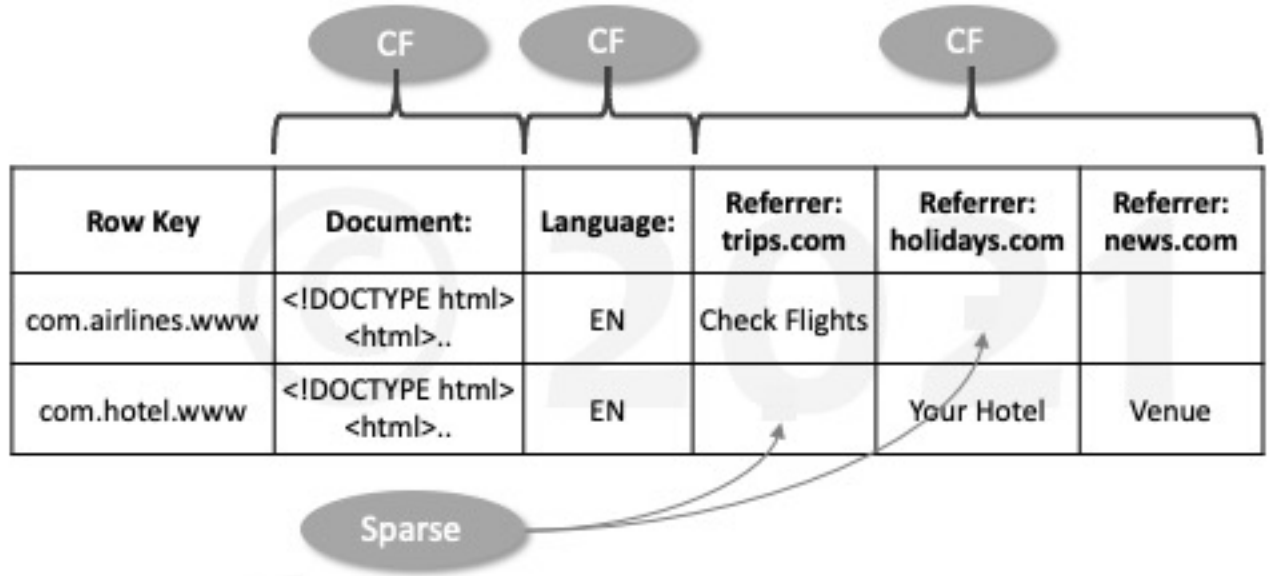
- Suitable for storing small chunks of data
  - Key values less than 1MB
- Peer-To-Peer cluster
  - No master – write on any node
  - Each node responsible for a set of keys
  - Consistent hashing of virtual nodes for key allocation
- Highly Scalable
  - Practically any number of nodes
  - Petabytes of data
  - Can handle 10 million RW ops requests/second
- Highly Available
  - Updates are not rejected even in case of network partitions or server failures
  - Vector clocks and business rules to resolve merge conflicts
- Consistency guarantee is adjustable
  - R+W > N for strong consistency



- Ring represents range of values for a partition key column
- Machine IP/Name is hashed to occupy a position on the ring

Google Big Table

- Basis for Apache HBase
- Column-Family Storage
- Table as Tree-Map
  - Sparse
  - **Sorted**
  - Persistent
  - Distributed
- Limited CF (< 100)
- CFs are compressed
- Unlimited columns
- R/W ops atomic for a key
- Timestamps for versioning



Row Key	Document:	Language:	Referrer: trips.com	Referrer: holidays.com	Referrer: news.com
com.airlines.www	<!DOCTYPE html> <html>..	EN	Check Flights		
com.hotel.www	<!DOCTYPE html> <html>..	EN		Your Hotel	Venue

Index Key: Row Key/Col Family:Col Name/timestamp

Sorted

Row Key	Column Family	Column	Timestamp	Value
com.airlines.www	Document		987654321	<!DOCTYPE html><html>..
com.airlines.www	Language		987654321	EN
com.airlines.www	Referrer	trips.com	987654321	Check Flights
com.hotel.www	Document		987655432	<!DOCTYPE html><html>..
com.hotel.www	Language		987655432	EN
com.hotel.www	Referrer	holidays.com	987655432	Your Hotel
com.hotel.www	Referrer	news.com	987655432	Venue

Bigtable Architecture

- Schema-less, structured (CF), sorted data
- GFS for reliable persistent storage
  - Replicated storage for large files
- In-memory tablet servers
  - High R/W throughput with low latency
  - Single copy of data for read and write
    - No write conflicts
- Horizontally scalable
  - Stores peta/exabytes of data
  - Client load is distributed
- Strongly consistent

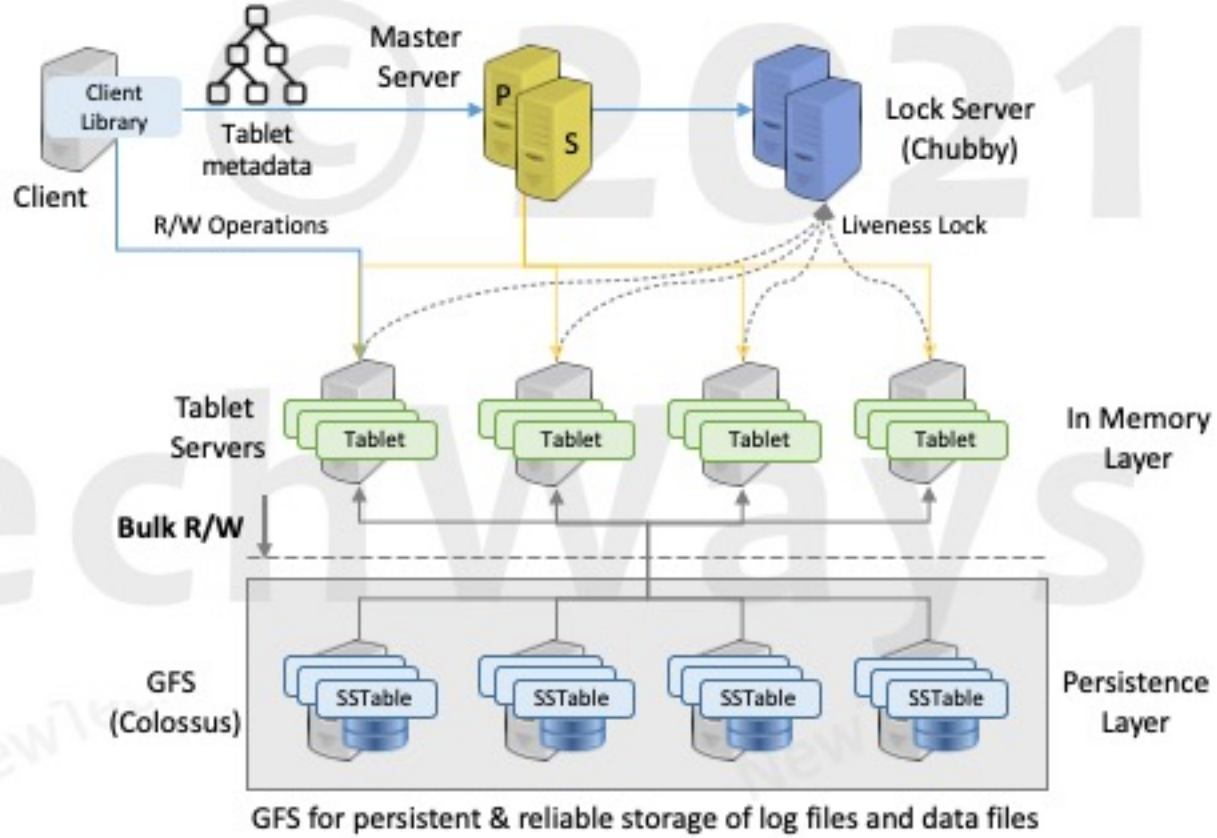
Tablet 1

Tablet 2

Data ranges are split into tablets

Client
- Caches tablet metadata
- directly works with tablet servers

Master
- Checks if tablet servers are live
- Assigns tablets to tablet servers

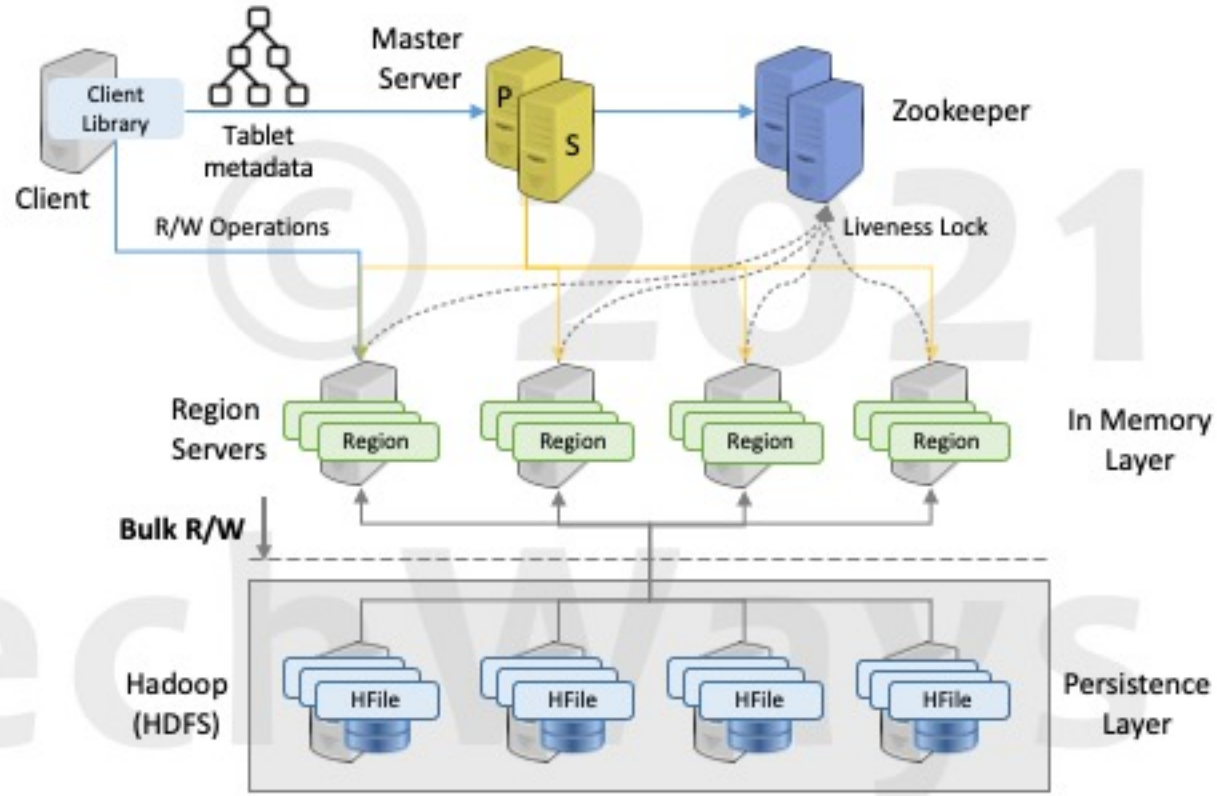


GFS for persistent & reliable storage of log files and data files

NewTechWays

HBase

- Open source implementation of Bigtable
- Column Family schema
- Keys are Range Partitioned
- Storage on Hadoop HDFS
- Strong consistency over high availability
- Highly scalable
- High throughput and low latency writes



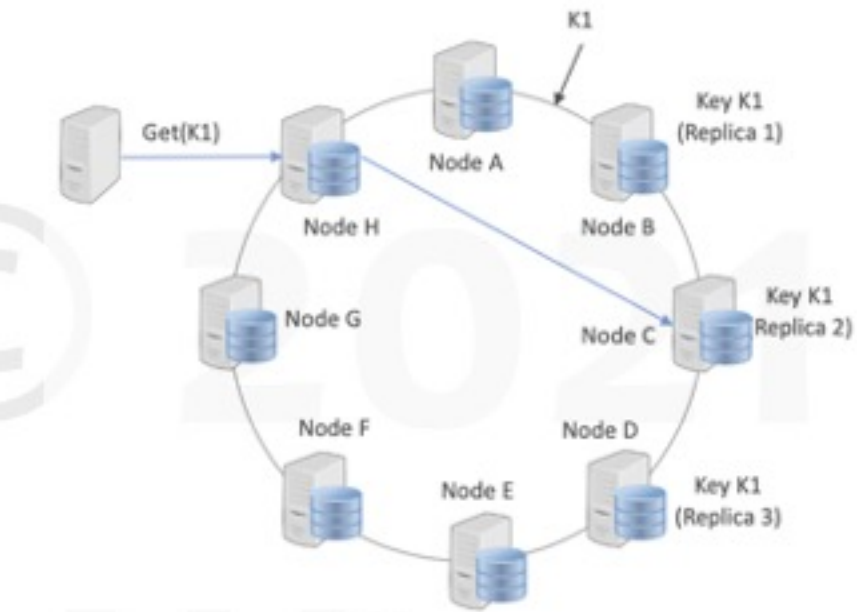
Cassandra

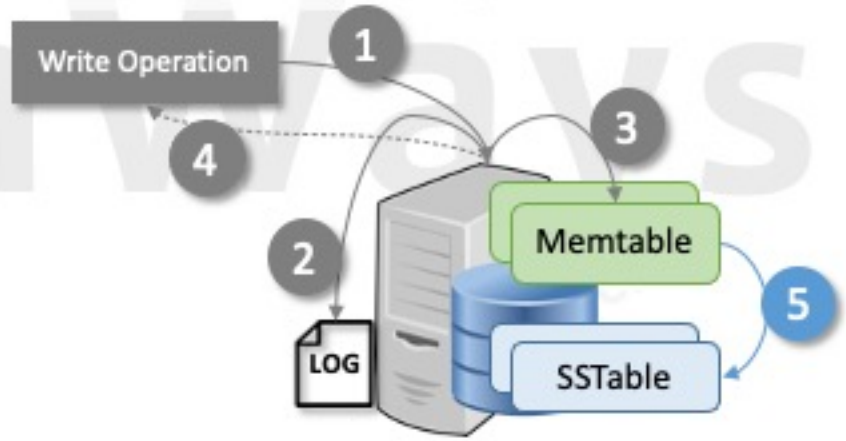
Partition Key | Sort Key | Columns

Product Id	Size	Name	Quantity	Discount
1001	L	Blue Denim	150	
1001	XL	Blue Denim	780	
1002	M	Black Shirt	960	10
1002	L	Black Shirt	300	20

Column Family

	Product Id	Size	Col Name	Col Value
Partition X	1001	L	Name	Blue Denim
	1001	L	Quantity	150
	1001	XL	Name	Blue Denim
	1001	XL	Quantity	780
Partition Y	1002	M	Name	Black Shirt
	1002	M	Quantity	960
	1002	M	Discount	10
	1002	L	Name	Black Shirt
	1002	L	Quantity	300
	1002	L	Discount	20





NewTechWays

Cassandra Features

- Schema-less, Column-Family Structured Data
  - Sparse
  - Persistent
  - Distributed
- Horizontally Scalable
  - Petabytes of data
- Highly Available
  - Even during network partitions
- High throughput R/W operations
- Merge conflicts
  - Vector clocks
  - Timestamps
  - Business rules

Data Model	Column Family	Big Table
Storage	Memtable → SSTable	Big Table
Partitioning	Hash Based	Dynamo
Replication	Peer-to-Peer	Dynamo
Consistency Model	Eventually Consistent	Dynamo
Availability	Highly Available	Dynamo

MongoDB

- Key -> Document
  - In Binary JSON (bson) format
- Columns created dynamically
- Columns can be indexed
- Documents can be queried
  - on id and or column values
- An operation on a single document is atomic
  - 2 PC for multiple documents

```
db.inventory.insertMany([
   { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },
   { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },
   { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" }
]);
```

Insert

```
db.inventory.find( { status: "A", qty: { $lt: 30 } } )
```

Query

```
db.inventory.find( { "size.h": { $lt: 15 } } )
```

Query with nested attribute

```
db.inventory.updateOne(
   { item: "paper" },
   {
     $set: { "size.uom": "cm", status: "P" },
     $currentDate: { lastModified: true }
   }
)
```

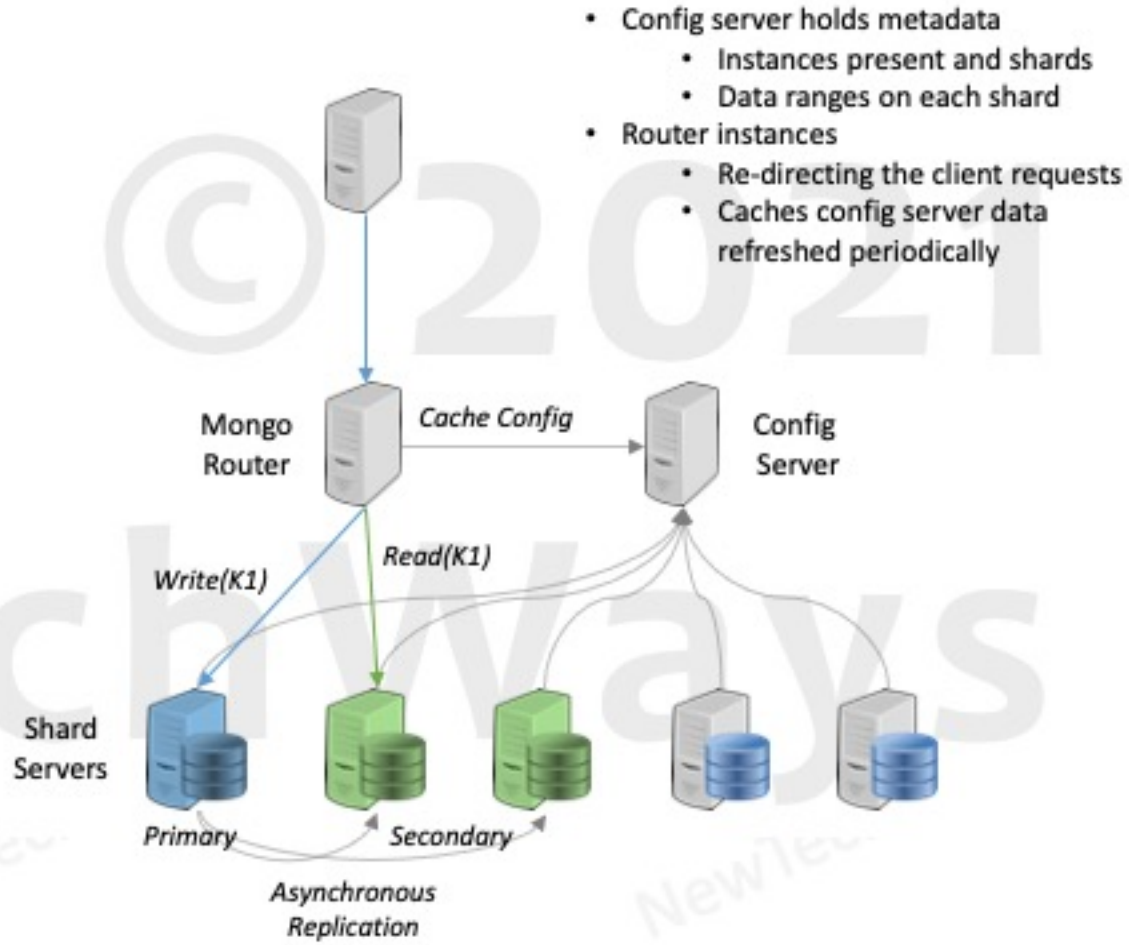
Update

```
db.products.createIndex(
  { item: 1, quantity: -1 }
)
```

Index Creation

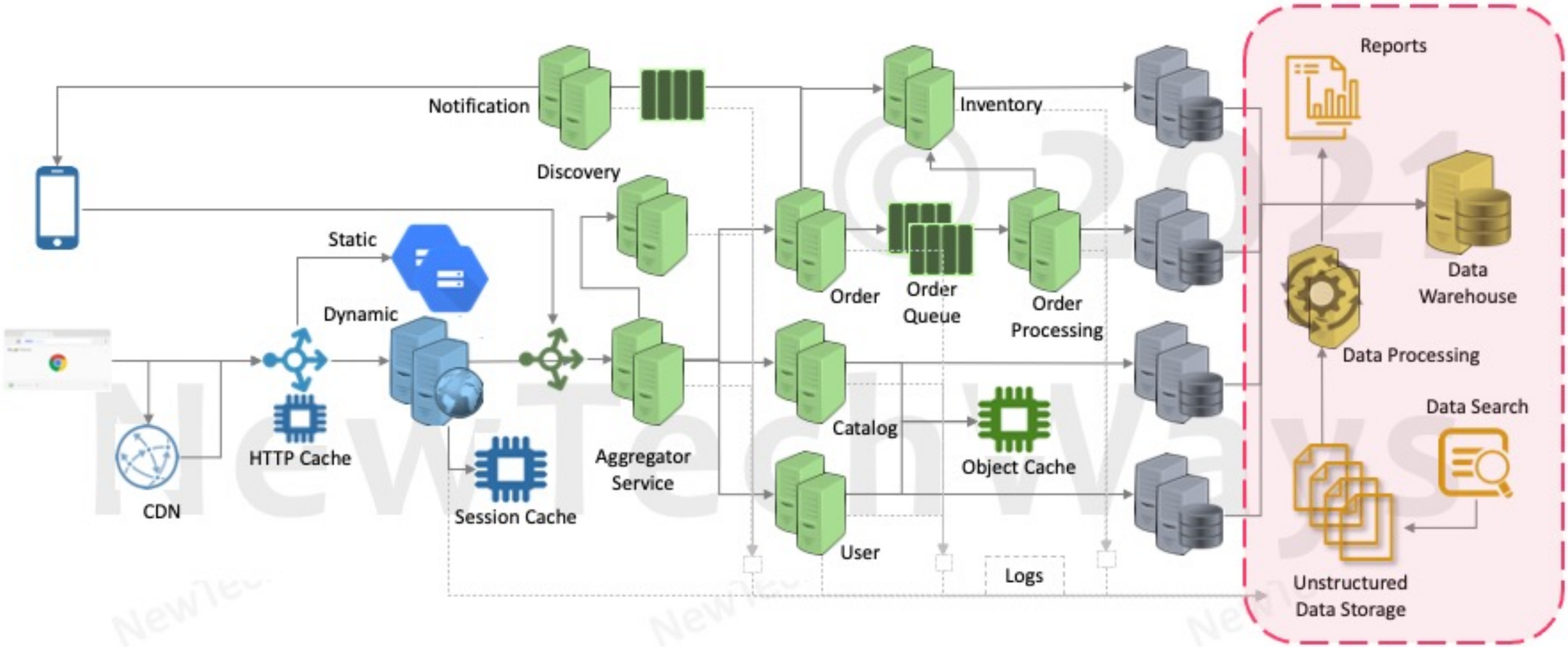
MongoDB Architecture

- Indexing for fast search
  - It's a write overhead
- Sharding for Scalability
  - Range sharding
  - Hash sharding
- Replication
  - Master Slave
    - No write conflicts
  - Asynchronous (default)
    - Eventual consistency
  - Synchronous (on demand)
- Works very well with Node.js
  - Javascript -> Node.js -> Mongo
    - JSON format



NewTechWays

Analytics



NewTechWays

Analytics – Solutions

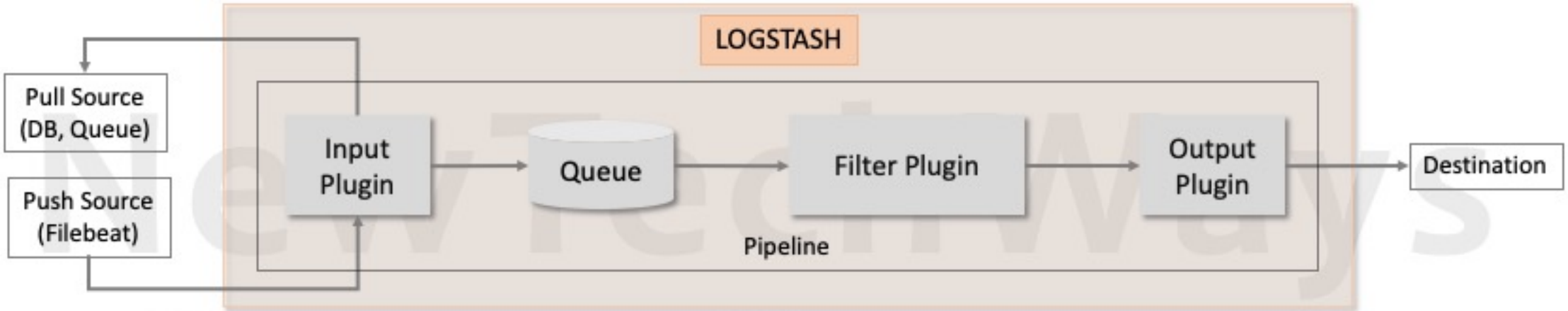
- Data Movement
  - Logstash
  - Fluentd
- Storage
  - Hadoop HDFS
    - Map-Reduce
  - Elastic Search
    - Search
- Stream Processing
  - Kafka
    - Buffer
  - Storm, Flink
    - Processing

Logstash Architecture

- I/O Plugins
  - Collect & Move log/stream events
- Queue
  - Reliable data movement
  - At least once delivery
  - Tracks Acks, Applies Backpressure
- Filter Plugins
  - Filter, Transform, Aggregate

```
127.0.0.1 - - [05/Feb/2012:17:11:55 +0000]
"GET / HTTP/1.1" 200 140 "-" "Mozilla/5.0...
```

```
{
    "host": "127.0.0.1",
    "user": "-",
    "method": "GET",
    "agent": "Mozilla/5.0 (windows…"
}
```

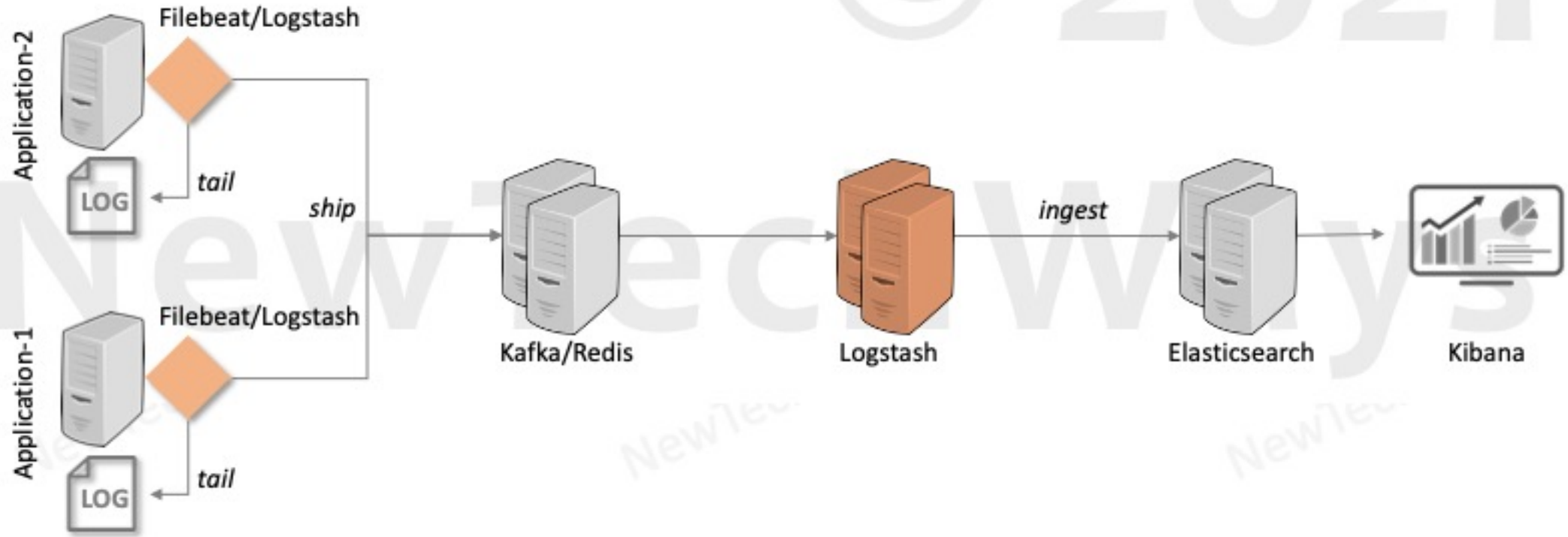


Access Logs App Logs System Logs DB Logs …	File, Beats, Http, JMS, Redis, Kafka, ES, …	Memory File	JSON, CSV, Grok, Dissect, Clone, Drop, Aggregate, …	ES, Redis, Webhdfs, S3, Kafka, File, Email, …	Alerts Analysis Archives

NewTechWays

Logstash Data Streaming Architecture

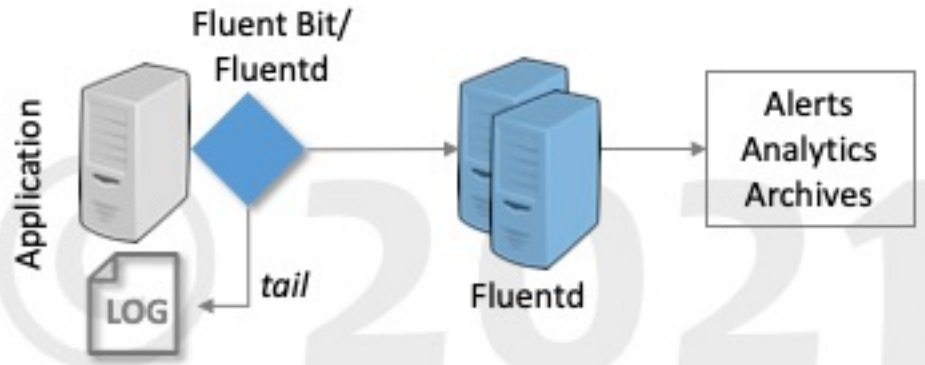
- Streaming log data for Real-Time Analytics
- Horizontally Scalable & Highly Available – Any number of Logstash nodes
- Fault Tolerance – Needs reliable disk storage (RAID, Cloud persistent disks)
- Use Kafka as buffer – For heavy load that Logstash Queue cannot handle



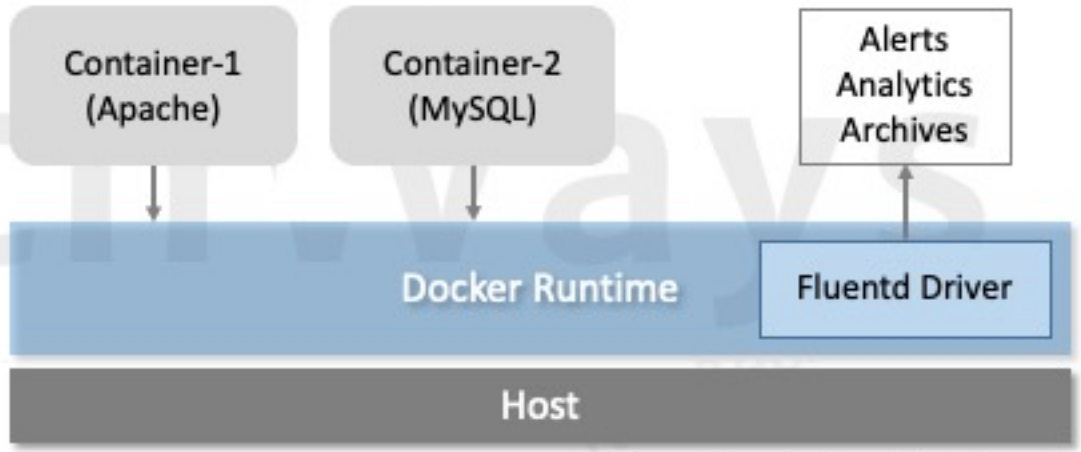
NewTechWays

Fluentd

- Older than Logstash
- Memory Footprint
  - Logstash – Heavyweight (GB)
  - Filebeat – Lightweight (MB)
  - Fluentd – Lightweight (MB)
  - Fluent Bit – Super Lightweight (KB)
- All features of Logstash
- + Routing
  - Tags
- + Docker Logging
  - Picks log events from container console



$> docker run –log-driver=fluentd



$> fluentd –c docker-fluent.conf

Elasticsearch

- Full-Text Search
  - Filter, Group, Aggregate
- Stores JSON Documents
- Document Fetched using id
- Indexes JSON keys and values
- Structure
  - Index -> Database
  - Type -> Table
  - Document -> Row
    - JSON keys are flattened
  - Supports data types
- Users can specify mapping between terms & documents

Documents

ID	Document
Doc-1	Exception in Order module
Doc-2	Your Order is created
Doc-3	Your Order is delivered

Inverted Index

Term	Frequency	Documents
created	1	Doc-2
delivered	1	Doc-3
Exception	1	Doc-1
in	1	Doc-1
is	2	Doc-2, Doc-3
module	1	Doc-1
Order	3	Doc-1, Doc-2, Doc-3
Your	2	Doc-2, Doc-3

```
{
  "_index": "accounts",
  "_type": "person",
  "_id": "1",
  "_source": {
    "name": "John",
    "lastname": "Doe",
    "job_description": "Systems administrator and Linux
      specialist"
  }
}
```

GET localhost:9200/accounts/person/1

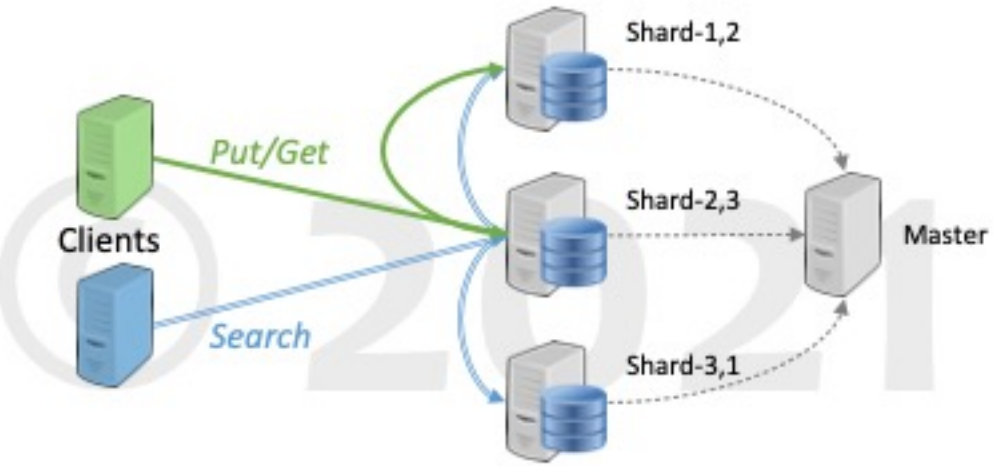
GET localhost:9200/_search?q=john

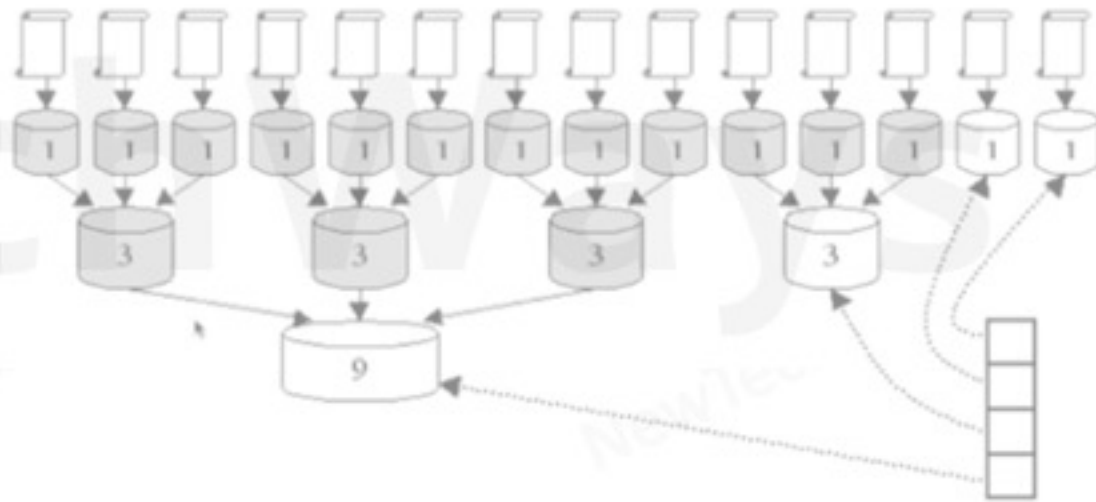
GET localhost:9200/_search?q=job_description:linux

GET localhost:9200/accounts/person/_search?q=job_description:linux

Elasticsearch Architecture

- Document Oriented data-model
- Horizontally Scalable
  - Petabytes of data
  - Data is sharded with key as Document Id
  - Put/Get request goes to specific shard
  - Search queries go to all shards
- Highly Available
  - Shards are replicated
- Index structure is based on merge sort
- Index not updated with every update or insert
- Index maintained in memory
  - Occasionally flushed to disk

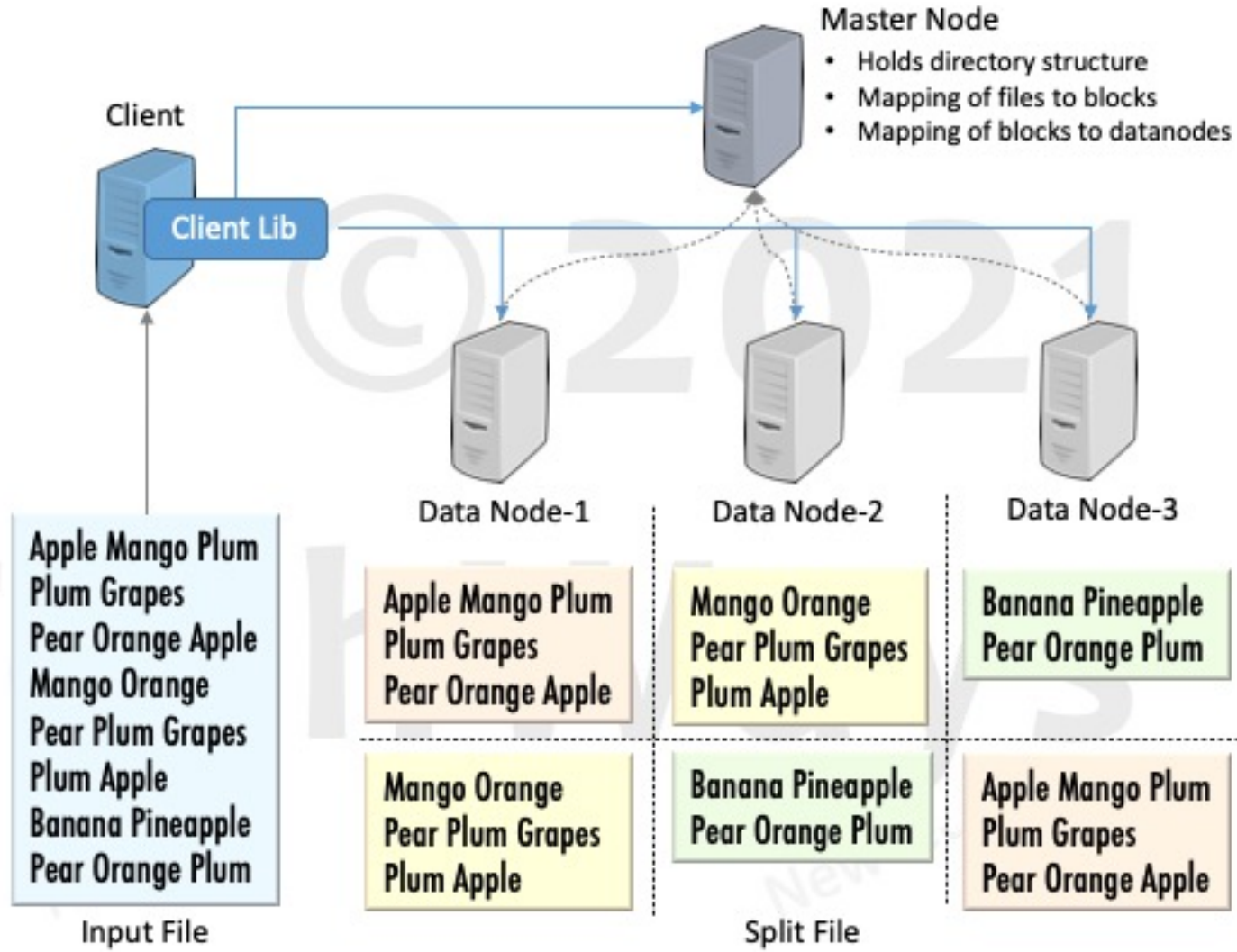




NewTechWays

Hadoop HDFS

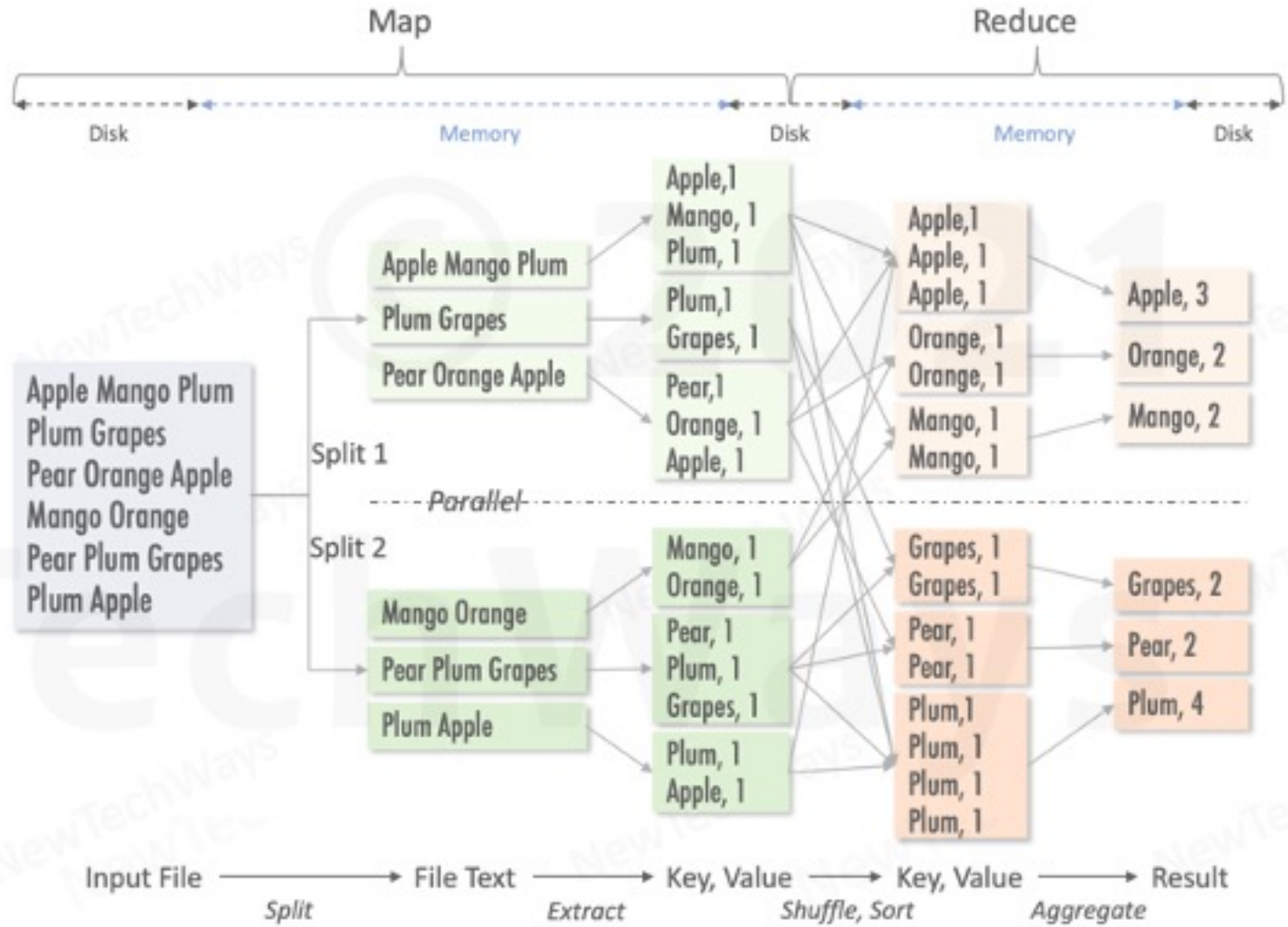
- Distributed File Data Storage
  - Unstructured data files
  - Petabytes of data
  - Large file size > 100MB
- Distributed files
  - Files broken into chunks
  - For parallel reads
    - Map-Reduce
- Sequential Writes – Append
  - Large blocks of data – 64MB
- Replication for reliability



NewTechWays

Map-Reduce

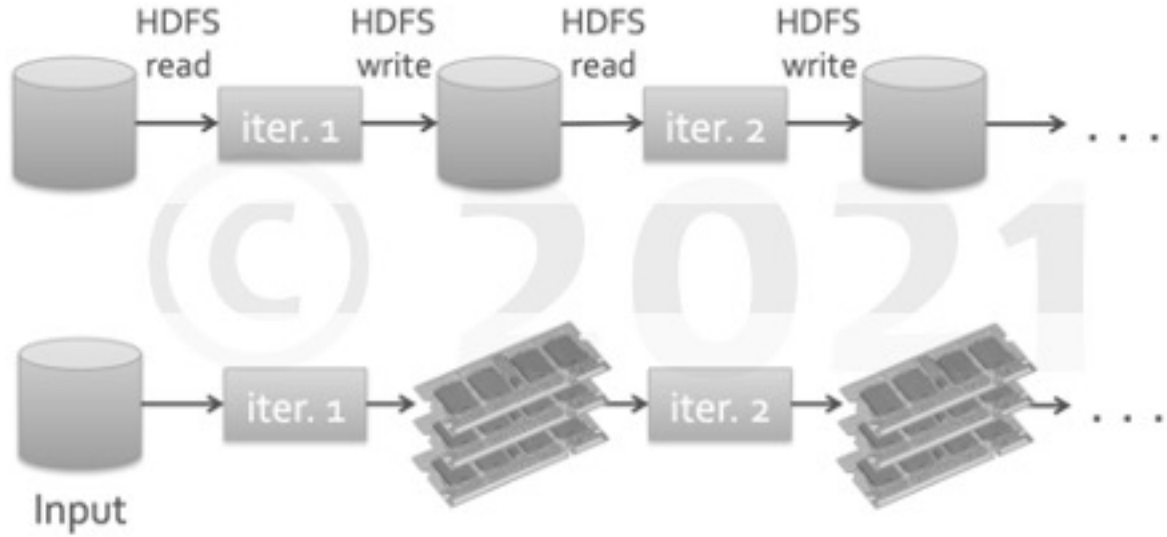
- Parallel file processing on Hadoop cluster
- Processing code executes on datanodes
- Input/Output sources
  - HDFS, HBase, Cassandra
- Map phase
  - Filtering and transformation
- Reduce phase
  - Shuffles map output across nodes
  - Groups related information
  - Computes aggregate data



NewTechWays

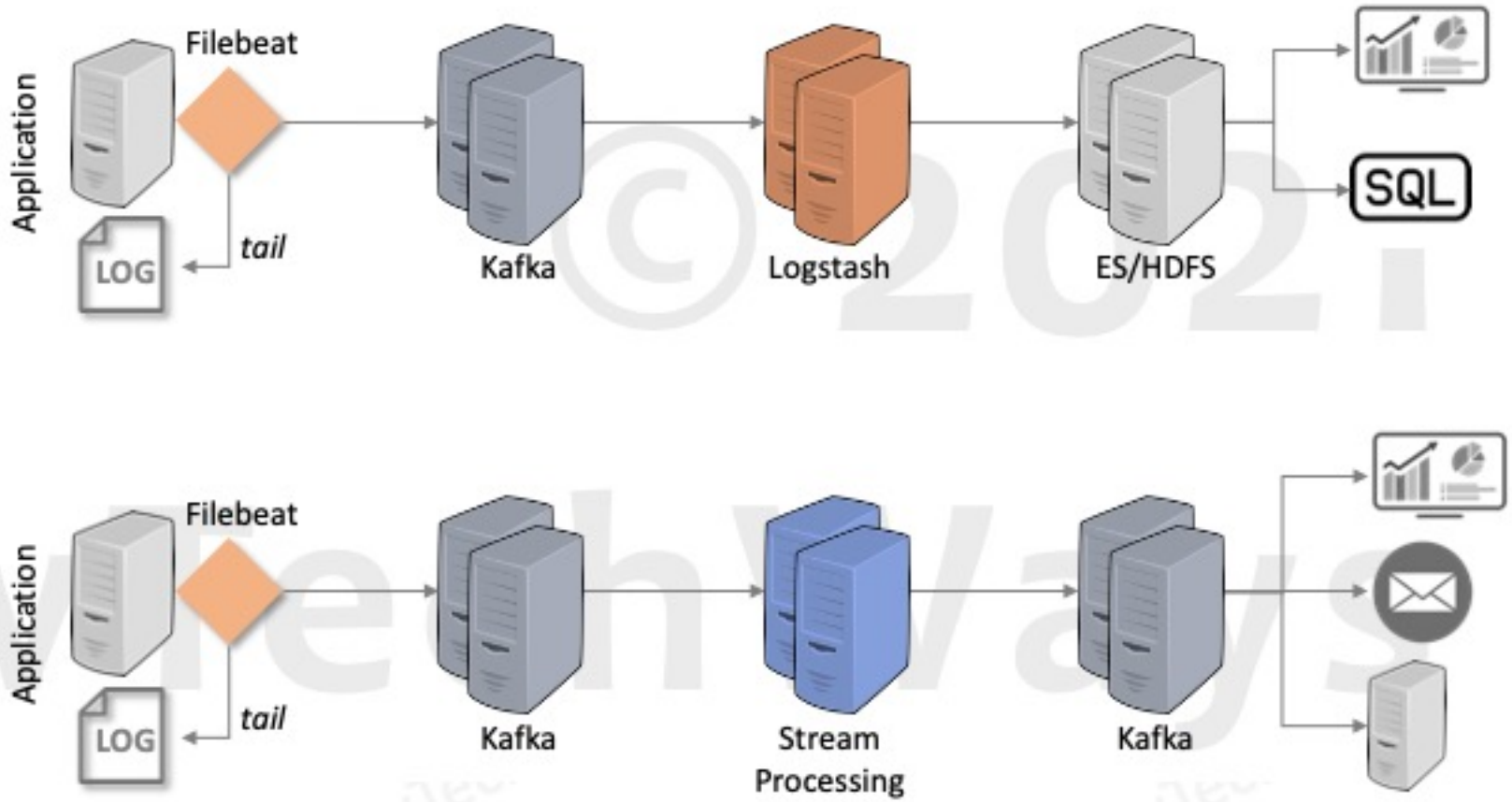
Apache Spark

- Evolution of Map-Reduce
- In memory
  - 10x to 100x times faster
- DAG of operations
  - Multiple operations in a DAG
  - Multiple inbuilt operations
- Interactive
  - Interactive shell in Scala/Python/R
- In built libraries for
  - SQL Interface, Machine Learning, Graph Processing, Streaming



Streaming Processing

- Low latency
  - Keep data moving
- High throughput
  - Multiple sources
- Event Issues
  - Event delay
  - Out-of-Order
  - Missing
- Fault Tolerance
  - Event Replay
- High Availability
- Processing Engine
  - Storm
  - Flink
  - Spark (Micro-Batching)
- Buffer
  - Kafka



NewTechWays

Thanks!

NewTechWays

https://www.newtechways.com