Web Applications

e.g.
Ryanair
Amazon
eBay

Web Application Architecture
- Three Tier

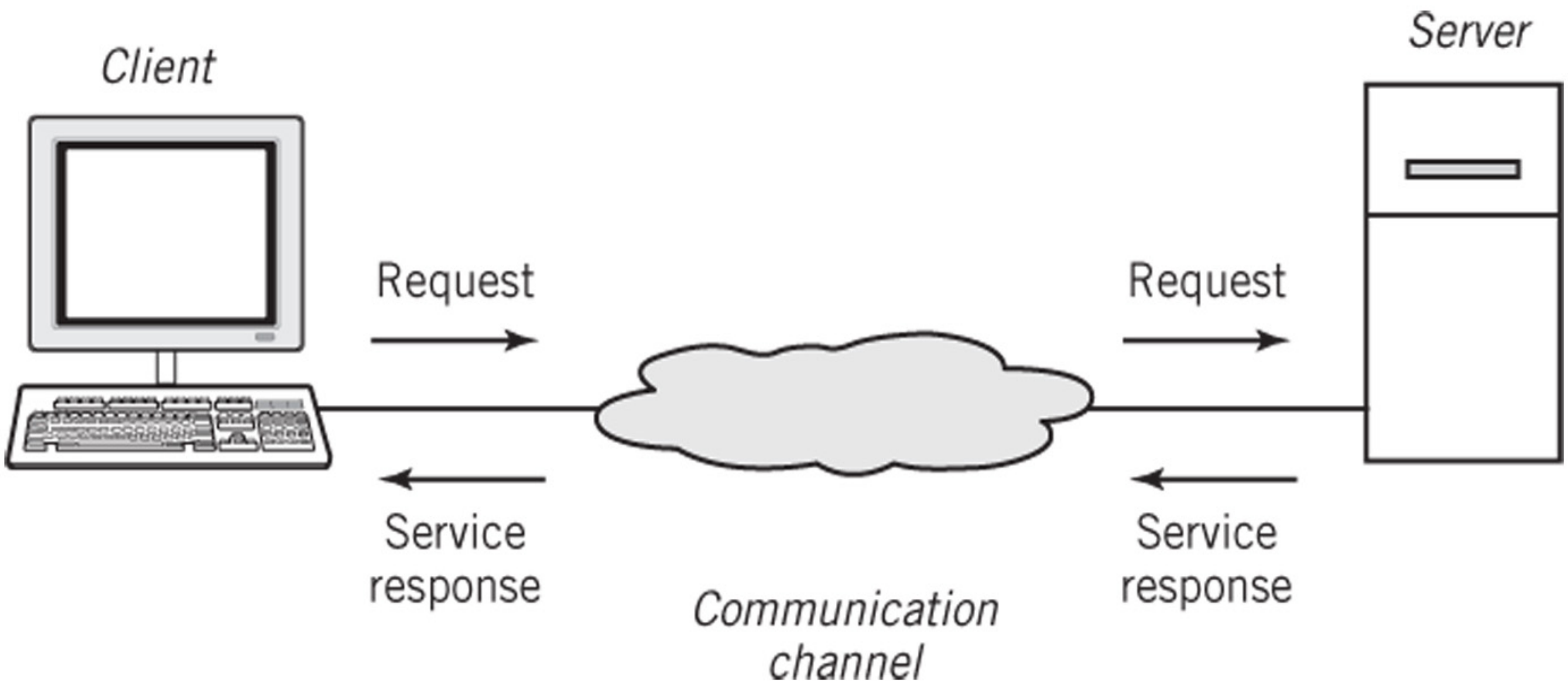
Scaling
- Horizontal versus Vertical
- Load Balancing

Database Replication

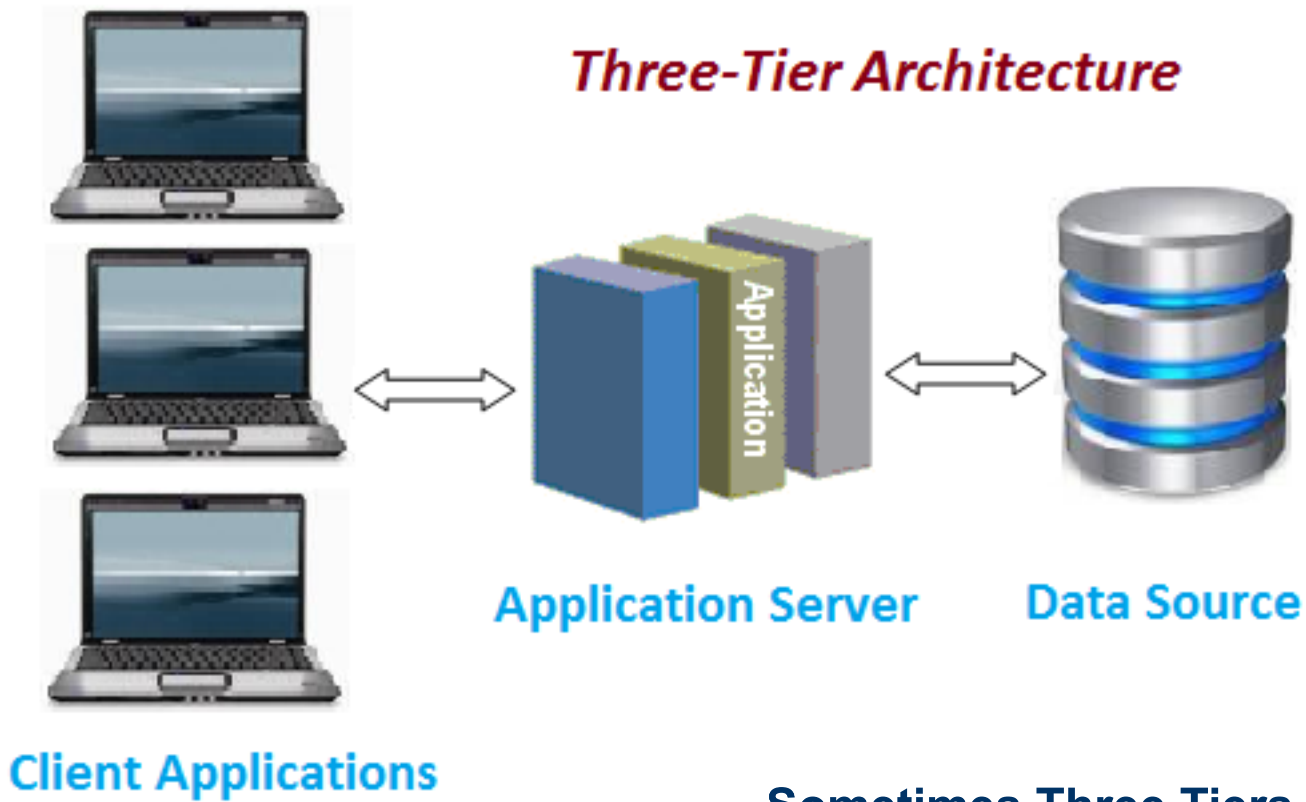
Cacheing

NoSQL Databases

Basic client-server architecture



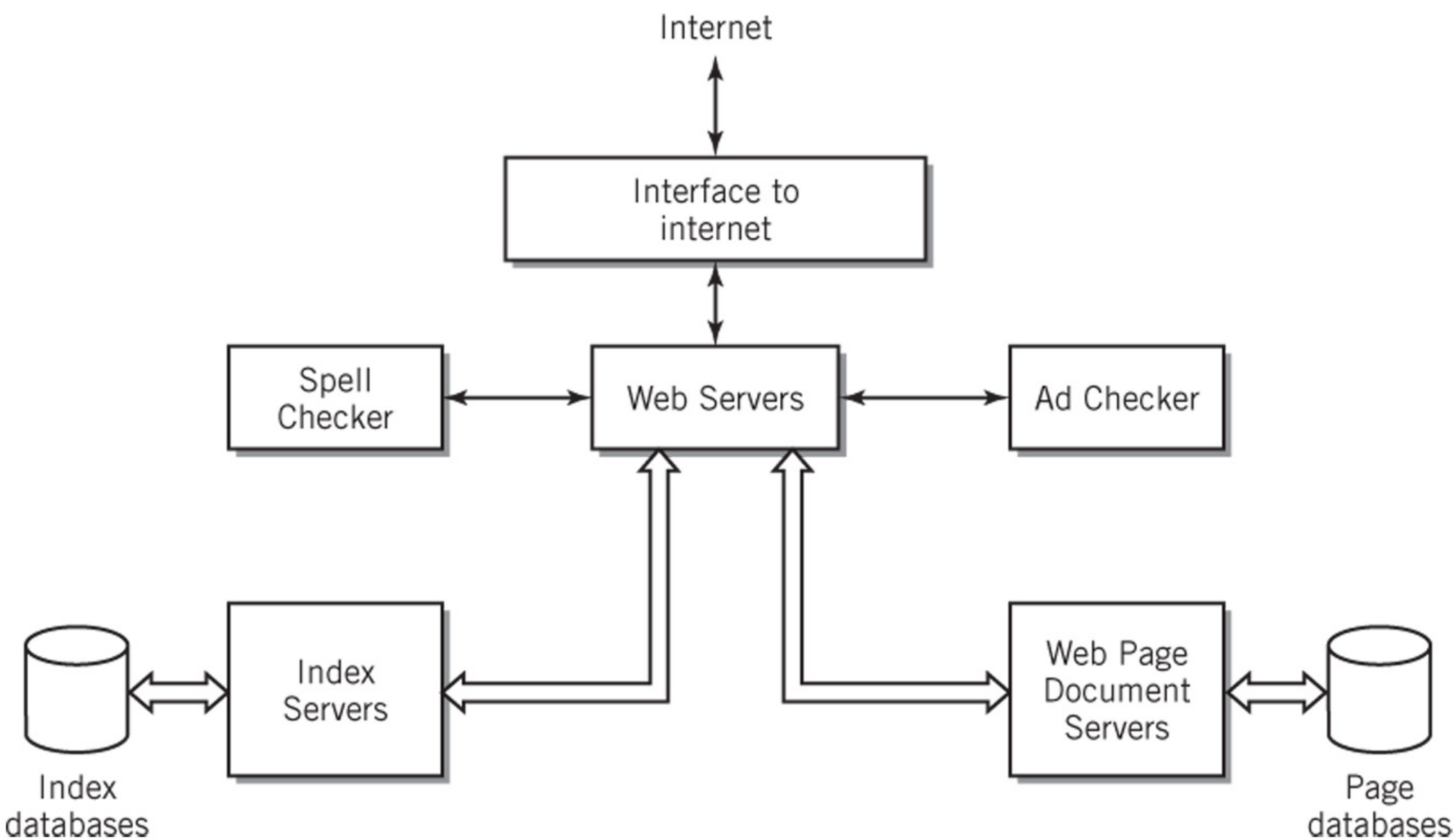
Web application architecture



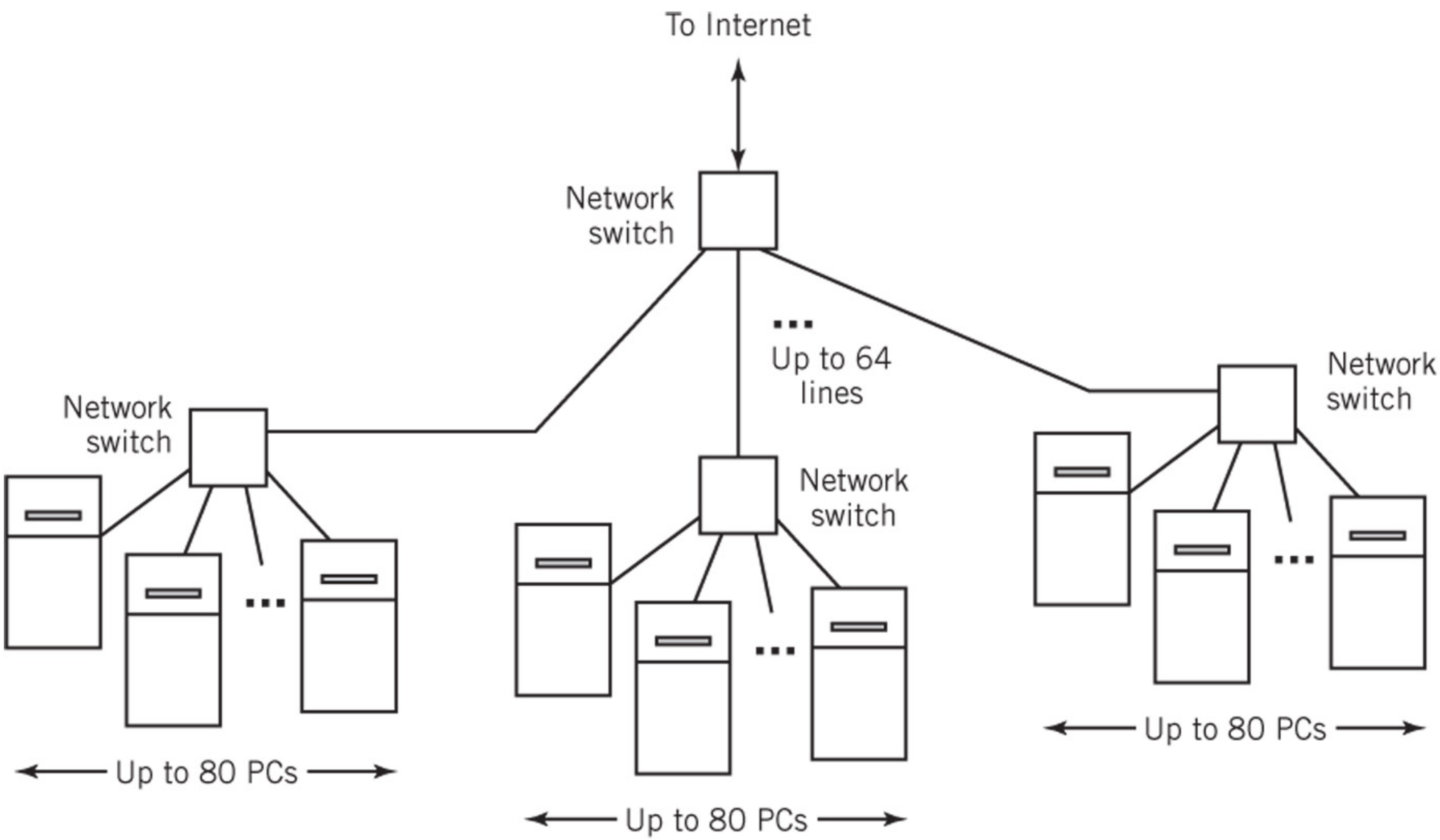
Think
Ryanair
Amazon
eBay

Sometimes Three Tiers are:
1. Client side
2. Web Server + Application
3. Database

Google Data Center Search Application Architecture



Simplified Google System Hardware Architecture



Design Considerations

High availability

Uptime - >revenue

Performance

Speed, user satisfaction ->revenue, retention

Reliability

Consistency

Scalability

How easy to handle additional traffic/data

Manageability

Cost

https://msdn.microsoft.com/en-us/library/ee658084.aspx

Web Application Architecture

Clients

Network/Internet

Web Server
- processes HTTP requests, returns web content
- e.g. Apache (Tomcat)

Application Server
- facilities to create web applications + server environment to run them
- e.g. Java EE, .NET framework

Database
- e.g. MySQL

Challenges

Failures/Errors

- 20% Network
- 10% Web server
- 30% Database server
- 40% Application server
  - Source: Performance, scalability and reliability issues in web applications, INDUSTRIAL MANAGEMENT & DATA SYSTEMS · JUNE 2005

Strategies

- Fault tolerance
- Scalability
- Load balancing
- Caching
- Data replication/sharding

Scalability

Elasticity is about doing this (semi)automatically

Capability of a system (network, process, algorithm etc) to handle a changing (growing) amount of work, or its potential to be enlarged in order to accommodate that growth

Scalable system – performance improves proportionally to resources added

- Each layer in architecture needs to be scalable – database connections, web server requests etc
  - What is scaling/scalability
  - Horizontal vs vertical scaling
  - Caching as scalability solution
  - Proxy Servers
  - Load balancing

Examples

How many concurrent users can your application handle until response time grows beyond x seconds?

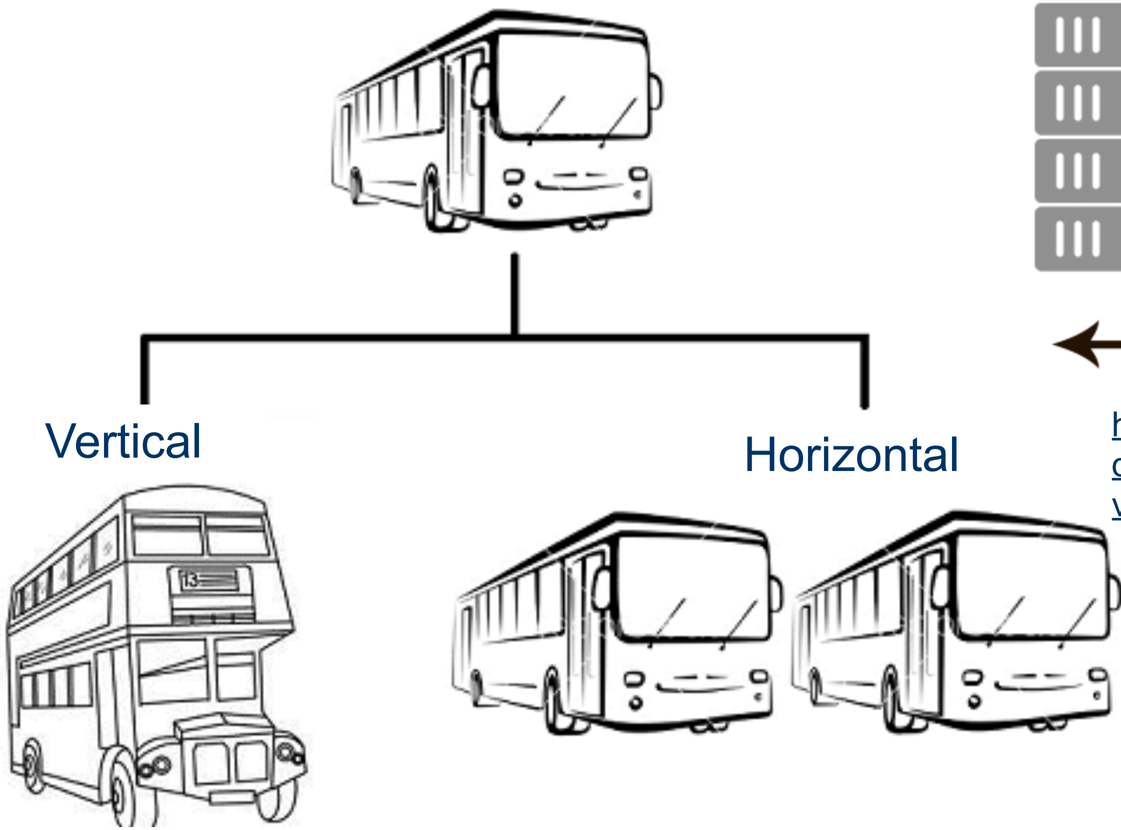
What about if add another CPU? More RAM? Another web server? Another database server?

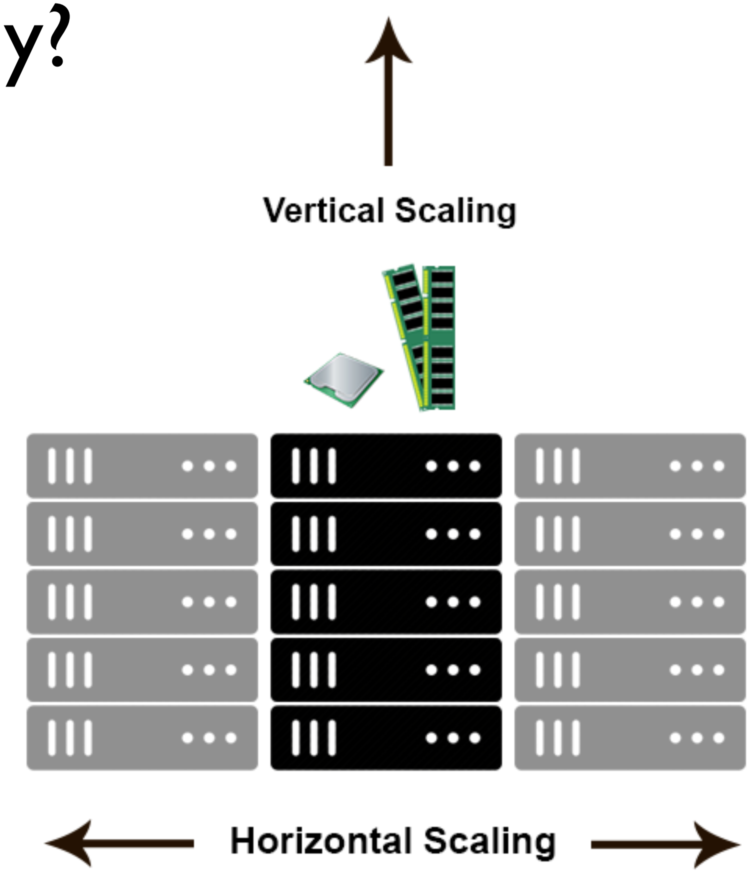
Ryanair case-study

http://www.couchbase.com/binaries/content/assets/us/customers/ryanair-case-study.pdf

> "Over the last 30 years, Ryanair has experienced exponential growth. Since we launched our new mobile app supported by Couchbase, we have increased app performance and decreased flight booking times from 5 minutes to 2 minutes."
> Vladimir Atanasov, Lead Developer,

Scale vertically or horizontally?





http://stackoverflow.com/questions/11707879/difference-between-scaling-horizontally-and-vertically-for-databases

Types of scaling

Vertical/scaling up

- Upgrade the box
- Add additional resources to a single node
  - CPU, RAM, Disk
- Improve existing system to handle more work

Horizontal/scaling out

- Add more nodes to a system
  - Servers
    - Load balancing? Distributed requests across servers
  - Databases
    - Partitioning/sharding – distribute data across databases
- Additional advantage – failover capability and higher availability

Trade offs

Horizontal

Larger number of computers – increased management complexity, more complex programming models, throughput/latency between nodes, consistency

- **But increased availability/failover**

Not all applications can be scaled horizontally – **gains from parallelising smaller**

Cost: linear

Vertical

Almost always directly speeds up the system

Simpler – no changes to application, still running on a single just a more powerful node

However physical limits

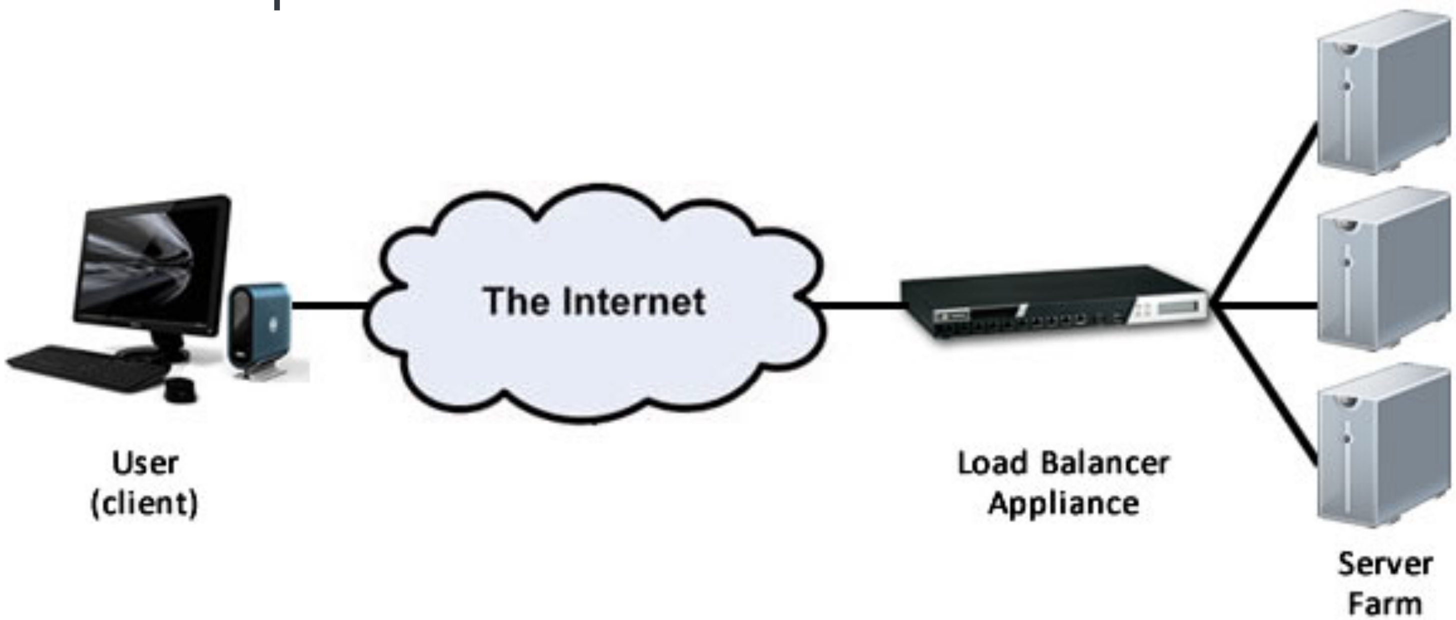
Cost: worse than linear

Load Balancing

Distribute client requests or network load efficiently across multiple servers

Ensures high **availability** and **reliability** by sending requests only to servers that are online

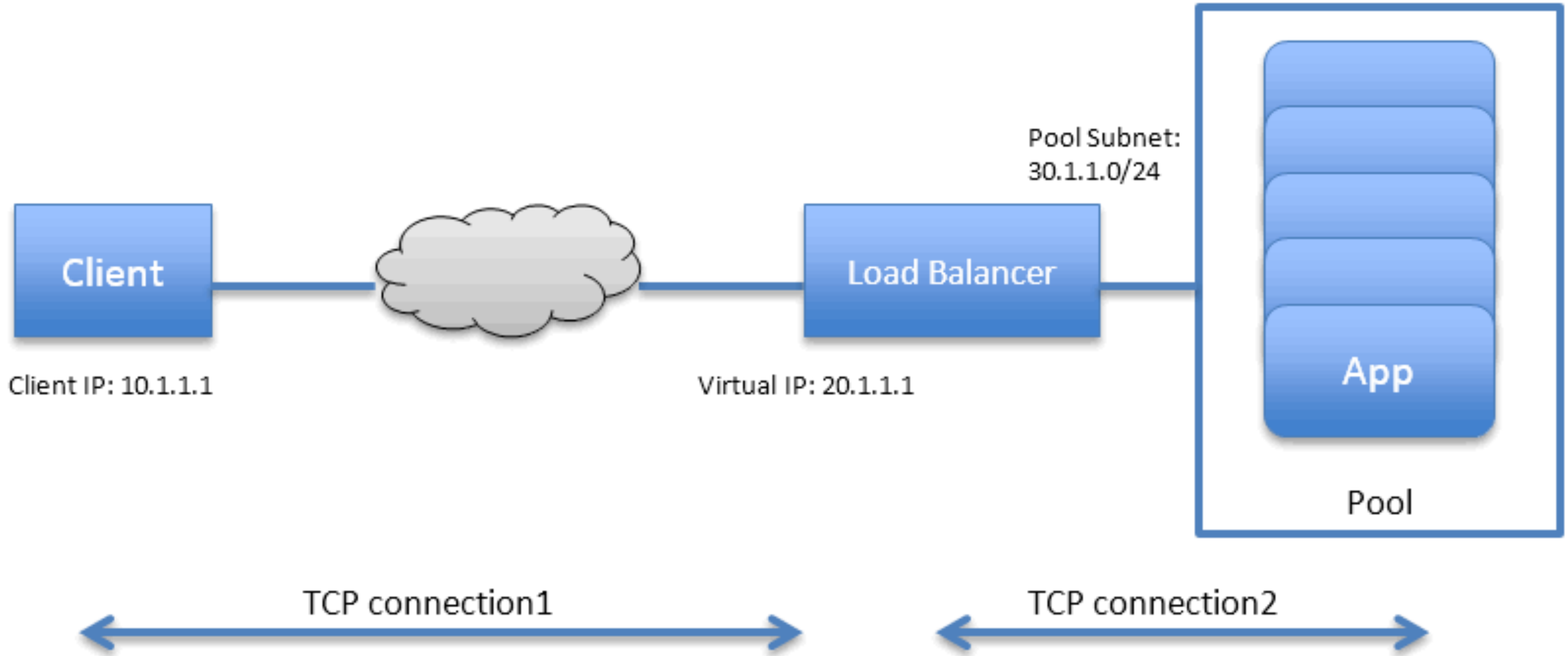
- Detects unresponsive and overloaded nodes



Load Balance where?

Application layer: application logic decides where to direct requests.

Transport layer: Scheduler decides where to route TCP requests



https://www.nginx.com/resources/glossary/layer-7-load-balancing/

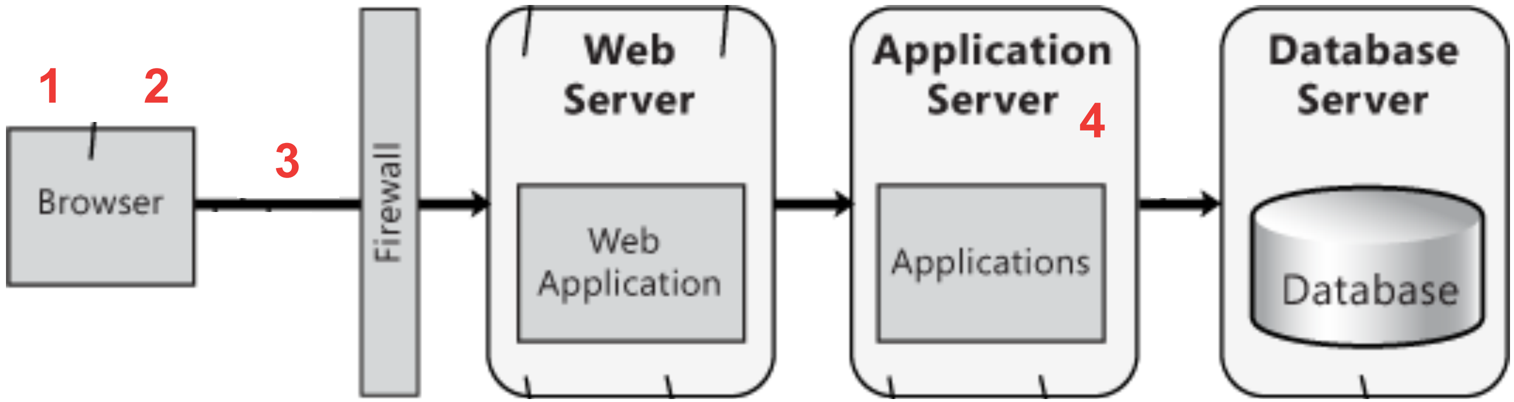
Session Persistence

How to handle information that must be kept across the multiple requests in a user's session

- If stored on one backend server, won't be accessible when routed to next one
- Always route to same server? Failover issues
- Keep session data in a DB? Increases DB load
- If client a browser – cookies

Session persistent and non-session persistent balancers

Caching in Web Applications



Reduce bandwith usage

Reduce server load

Reduce response time

On the other hand – can be complex, need to maintain additional storage

Where?

1. In app
2. In browser
3. Client site proxy
4. Application server

Technology examples - caching

Memcached (http://memcached.org)

distributed memory caching system

Speed up dynamic database-driven websites by caching data in RAM to reduce the number of times database must be read

Youtube, reddit, zynga, **facebook**, twitter, wikipedia

https://www.youtube.com/watch?v=UH7wkvcf0ys

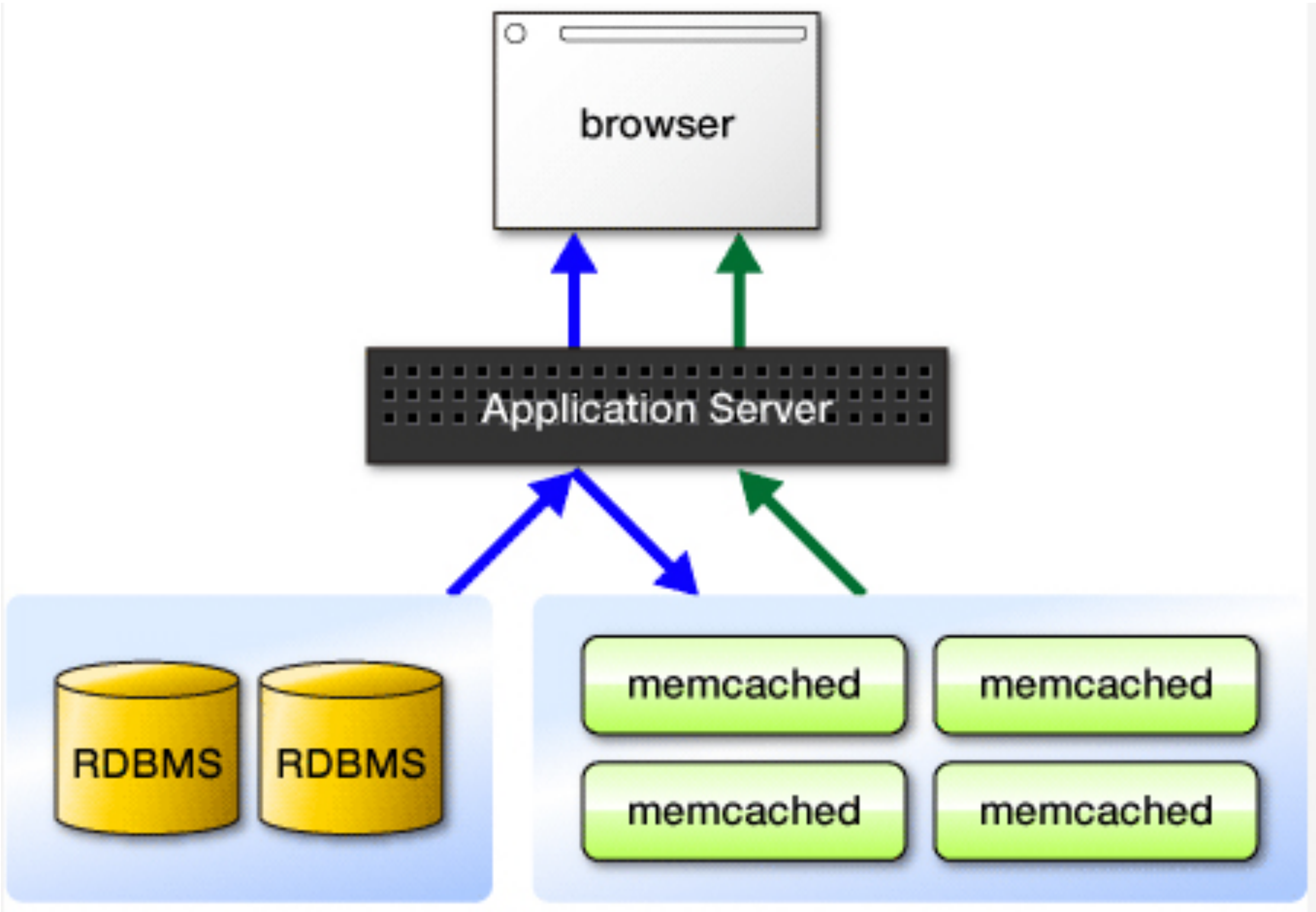
Redis (http://redis.io)

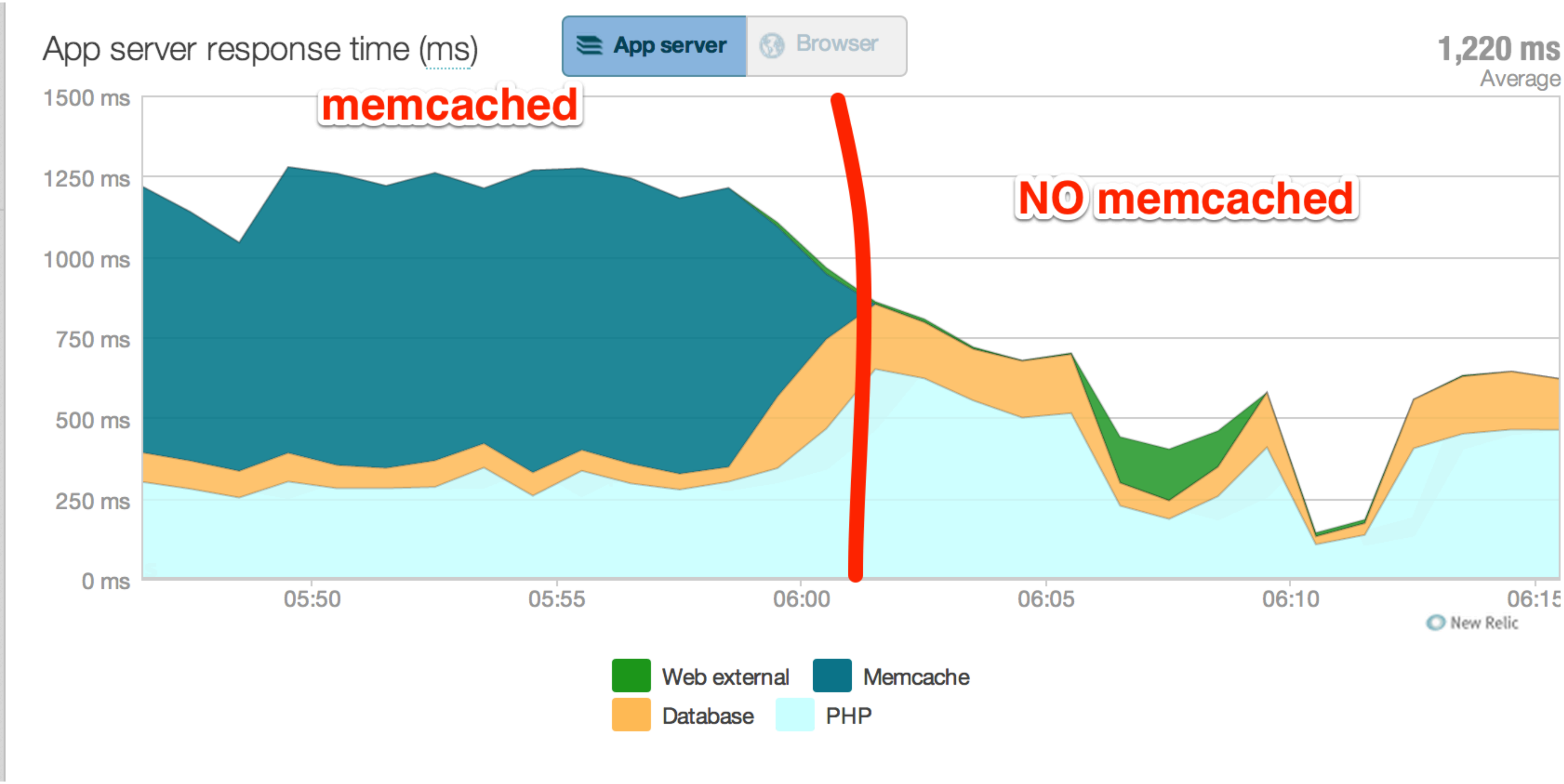
in-memory **data structure store**, used as database, cache and message broker

Both are open source, noSQL in memory key-value data stores

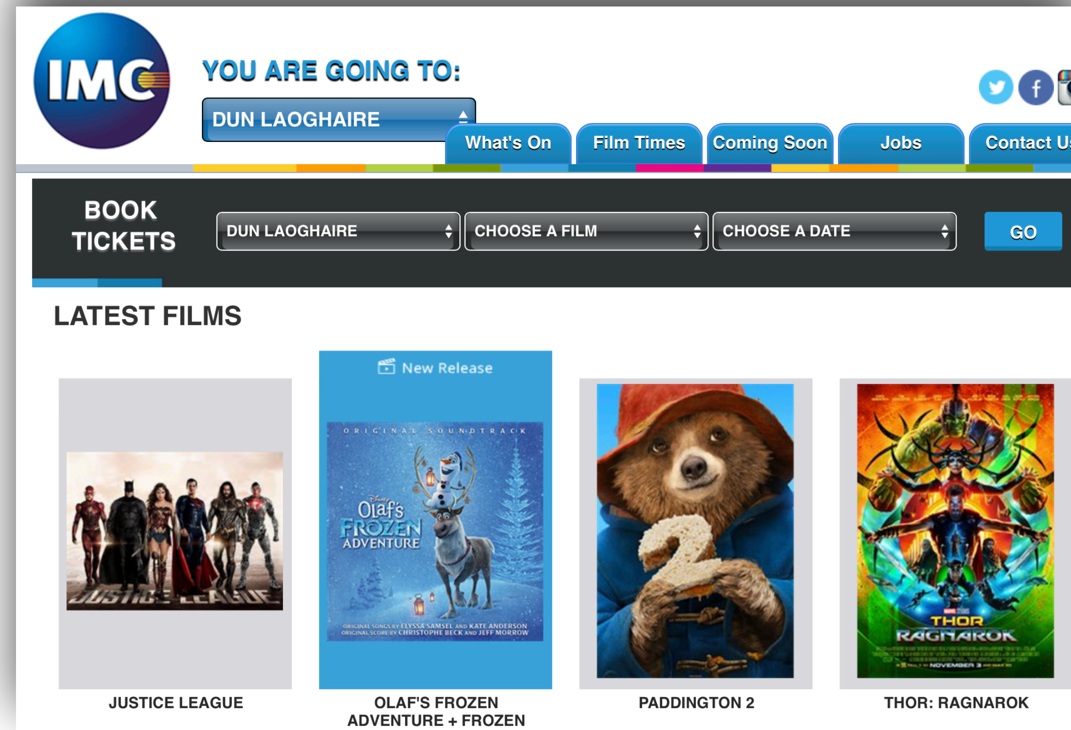
https://medium.com/@Alibaba_Cloud/redis-vs-memcached-in-memory-data-storage-systems-3395279b0941

Memcached (& Redis)




App server response time (ms)
App server
Browser
1,220 ms
Average
memcached
NO memcached
1500 ms
1250 ms
1000 ms
750 ms
500 ms
250 ms
0 ms
05:50
05:55
06:00
06:05
06:10
06:15
New Relic
Web external
Memcache
Database
PHP

Cacheing & Hashing



Memcached uses a hashtable for cacheing

Data stored as <key><value> pairs

Consider a web application for cinema booking

Film times are cached

```
data = memcached_fetch(<key>)
if not data
     data = db_select(<db_query>)
     memcached_add(<key>, data)
```

Hashtables in Python

Hashing in Python done using Dictionaries

Setup

```
FTm = {'DaddysHome2': ['11:00','13:30','16:00','18:30','21:00'],
       'Paddington2' : ['13:00','14:30','16:15','17:00'],
       'JusticeLeague' : ['12:30','15:15','18:00','20:30'],
       'BattleoftheSexes' : ['17:30','20:15']}

FTm
Out[28]:
{'BattleoftheSexes': ['17:30', '20:15'],
 'DaddysHome2': ['11:00' ,13:30', '16:00', '18:30', '21:00'],
 'JusticeLeague': ['12:30', '15:15', '18:00', '20:30'],
 'Paddington2': ['13:00', '14:30' , '16:15', '17:00']}

FTm['JusticeLeague']
Out[29]:
['12:30', '15:15', '18:00', '20:30']
```

Retrieve

http://www.laurentluce.com/posts/python-dictionary-implementation/

Updating the Dictionary

Add

```
FTm ['MurderontheOrientExpress'] = ['14:40','17:20','19:30']

FTm
Out[31]:
{'BattleoftheSexes': ['17:30', '20:15'],
 'DaddysHome2': ['11:00' ,13:30', '16:00', '18:30', '21:00'],
 'JusticeLeague': ['12:30', '15:15', '18:00', '20:30'],
 'MurderontheOrientExpress': ['14:40', '17:20', '19:30'],
 'Paddington2': ['13:00', '14:30' , '16:15', '17:00']}
```

Proxy Server

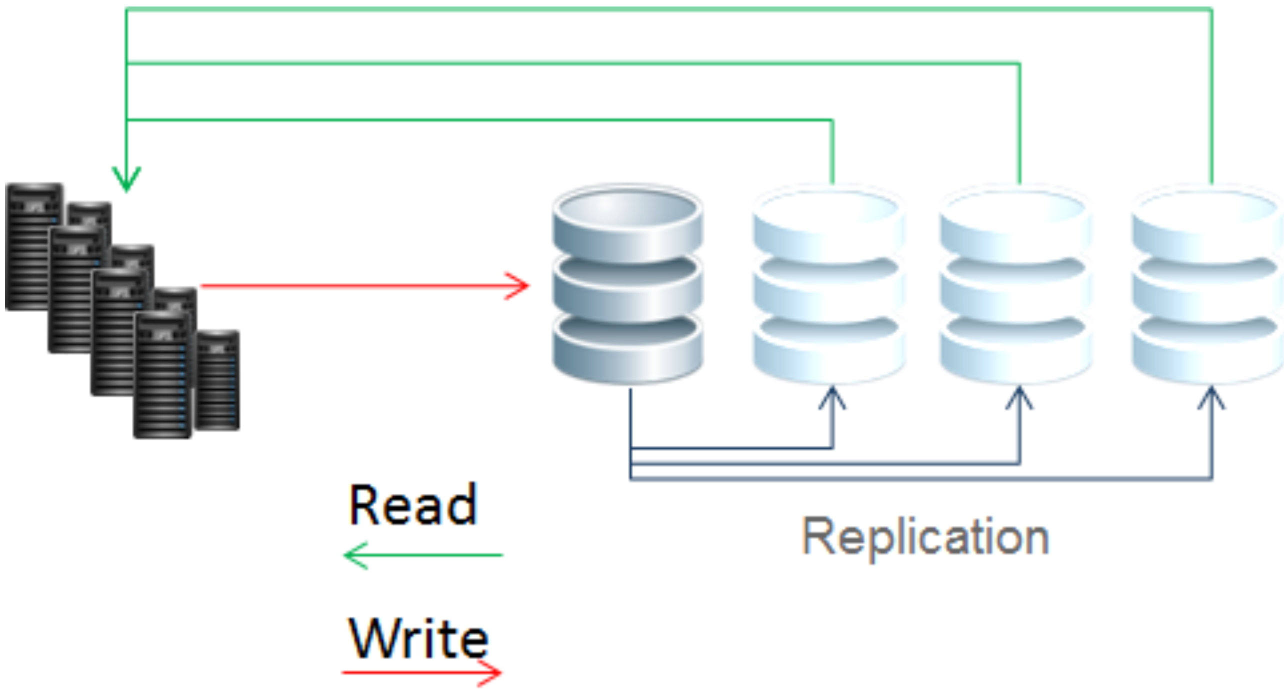
Intermediate piece of software/hardware that receives requests from clients and relays them to backend servers

Filter requests, log requests, transform requests, Anonymity, security, load balancing, caching

Optimize performance – collapse same/similar (for spatially close data) requests together into one request

Database replication

The process of creating and maintaining multiple instances of the same database and the process of sharing data or database design between databases in different locations without having to copy the entire database



Writes go to the Master
Reads can be handled by Slaves

http://database-scalability.blogspot.ie/2012/06/catch-22-of-readwrite-splitting.html#.WC7kUHd0eMQ

Master-slave replication

One database server maintains the master copy of the database – Master

Additional database servers maintain slave copies of the database – Replicas (slaves)

The two or more copies of a single database remain synchronized

- Synchronisation updates only data that has changed

Master + slaves = replica set

Fault Tolerance
When a Master fails promote the most up-to-date Slave to be Master

NoSQL databases

Mechanism for storage and retrieval of data that is modelled in means other than the tabular relations used in relational databases.

Eg key-value, graph, document

Simplicity of design, simpler horizontal scaling/sharding

- Increased use in big data and web apps

MondoDB, Redis, Cassandra, Hbase etc

Web Services

- Web Application Architecture
  - Three Tier
- Scaling
  - Horizontal versus Vertical
  - Load Balancing
- Database Replication
- Cacheing
- NoSQL Databases