Failures and Resiliency Principles of Distributed Computing

course 2024

arnaud-nauwynck@gmail.com

This document:
https://github.com/Arnaud-Nauwynck/presentations/pres-bigdata/
3-failure-and-resiliency-pinciples-distributed-computing

MTBF

M.T.B.F = Mean Time Between Failure

For HDD ~ 500 000 hours   ≥ 50 years

May looks good at home, to save your data
("Mean" is probabilistic average...
  Real errors will occur before)

MTBF « at Scale »

```
In DataCenter with 10 000 servers x 4 disks

   =>  1 failure every 1h 15mn

    =  19 failures per day


becomes a recurrent task / job
```

When Failure(s) Happens ?

=> Program fails ?

=> System fails ?

=> need relaunch manually ?

=> diagnostic hardware/software ?

=> interrupt all, repair ?

Studying « When Failure » 1/5

=> Program fails ?

=> System fails ?

=> need relaunch manually ?

=> diagnostic hardware/software ?

=> interrupt all, repair ?

When Failure..

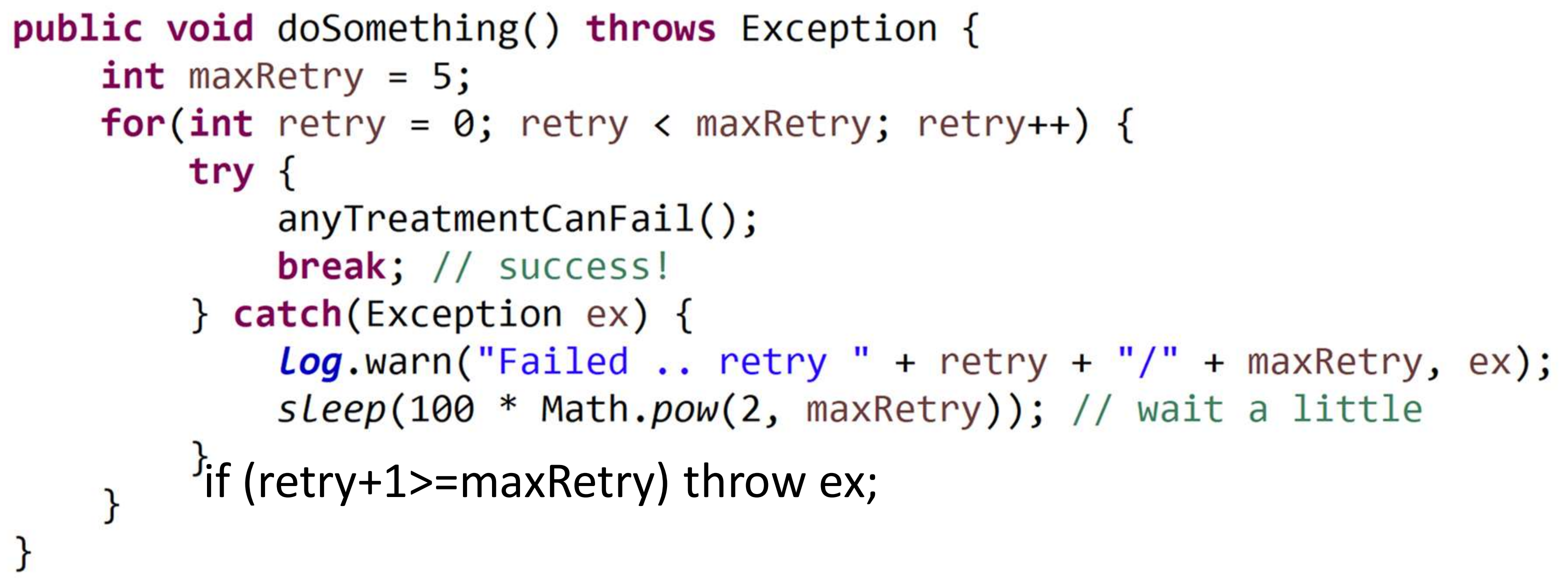
=> Program fails ?

Individual Component : Obviously FAIL

Distributed Architecture : RESILIENT

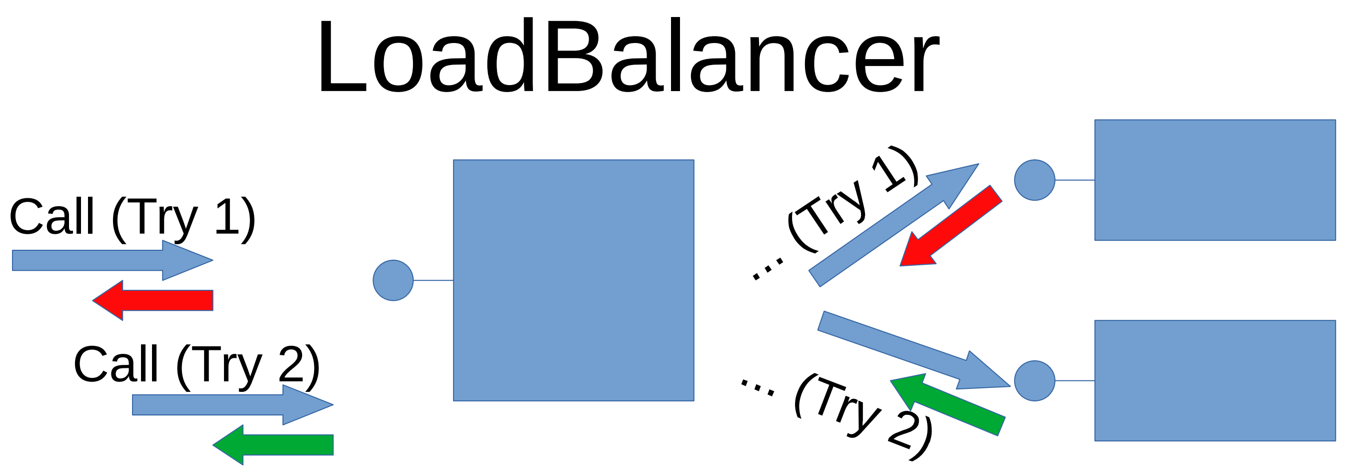
resist (hopefully) to (some) failure

Failure is not « Exceptional » .. is a « normal » path consider it everywhere in code



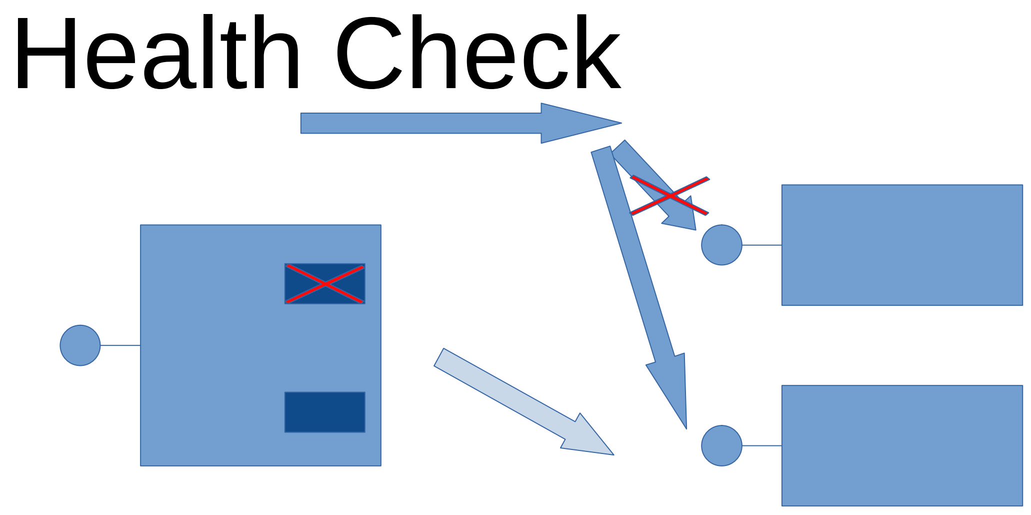
```
public void doSomething() throws Exception {
    int maxRetry = 5;
    for(int retry = 0; retry < maxRetry; retry++) {
        try {
            anyTreatmentCanFail();
            break; // success!
        } catch(Exception ex) {
            log.warn("Failed .. retry " + retry + "/" + maxRetry, ex);
            sleep(100 * Math.pow(2, maxRetry)); // wait a little
        }if (retry+1>=maxRetry) throw ex;
    }
}
```

Retrying with a LoadBalancer .. might just work

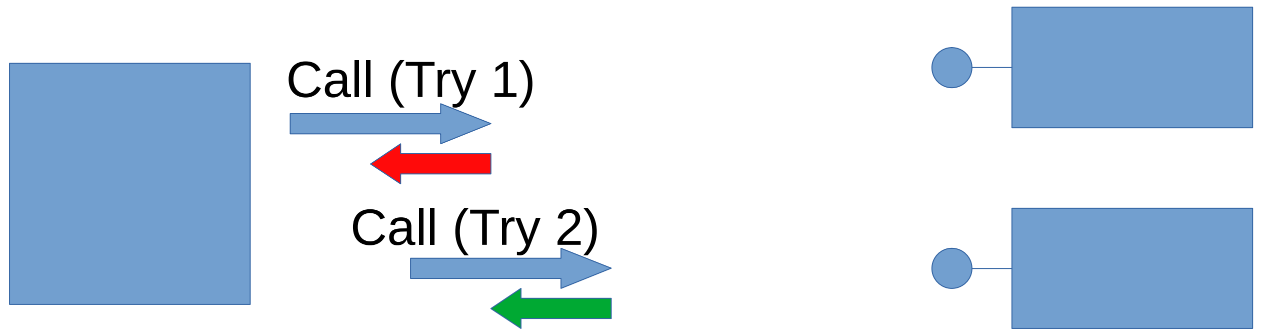


Dispatch to underlying servers
Using Round-Robin

Server Health Check
Temporary evict
from RoundRobin Pool



Client-Side « LB »

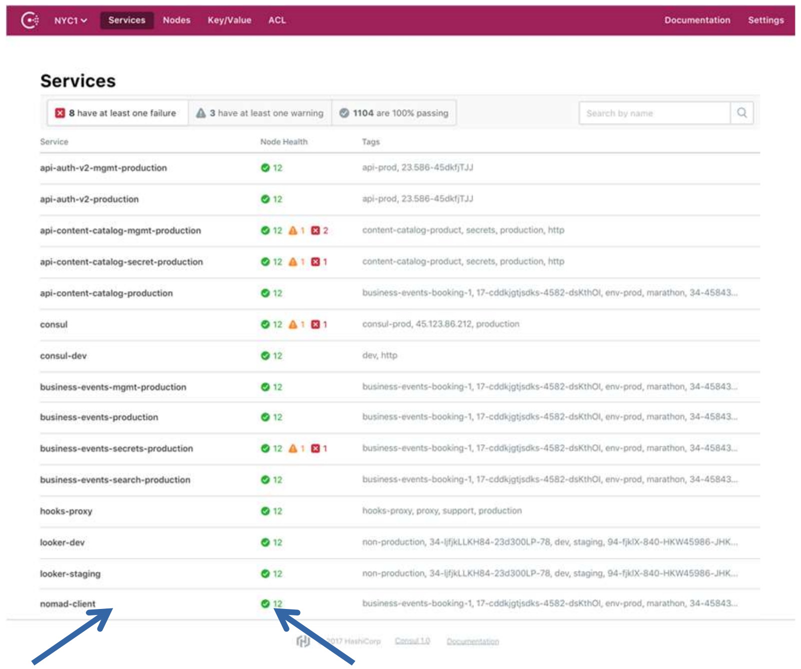


NO « LB »

Need discovery mecanism on client

Service >= Nodes
Examples of Service Discovery

DNS, ServiceMesh, Zookeeper, Consul.io, HAProxy, Kubernetes, ..



Services

8 have at least one failure | 3 have at least one warning | 1104 are 100% passing

Service	Node Health	Tags
api-auth-v2-mgmt-production	✓ 12	api-prod, 23.586-45dkfjTJJ
api-auth-v2-production	✓ 12	api-prod, 23.586-45dkfjTJJ
api-content-catalog-mgmt-production	✓ 12 ⚠ 1 ✗ 2	content-catalog-product, secrets, production, http
api-content-catalog-secret-production	✓ 12 ⚠ 1 ✗ 1	content-catalog-product, secrets, production, http
api-content-catalog-production	✓ 12	business-events-booking-1, 17-cddkjgtjsdks-4582-dsKthOl, env-prod, marathon, 34-45843...
consul	✓ 12 ⚠ 1 ✗ 1	consul-prod, 45.123.86.212, production
consul-dev	✓ 12	dev, http
business-events-mgmt-production	✓ 12	business-events-booking-1, 17-cddkjgtjsdks-4582-dsKthOl, env-prod, marathon, 34-45843...
business-events-production	✓ 12	business-events-booking-1, 17-cddkjgtjsdks-4582-dsKthOl, env-prod, marathon, 34-45843...
business-events-secrets-production	✓ 12 ⚠ 1 ✗ 1	business-events-booking-1, 17-cddkjgtjsdks-4582-dsKthOl, env-prod, marathon, 34-45843...
business-events-search-production	✓ 12	business-events-booking-1, 17-cddkjgtjsdks-4582-dsKthOl, env-prod, marathon, 34-45843...
hooks-proxy	✓ 12	hooks-proxy, proxy, support, production
looker-dev	✓ 12	non-production, 34-ljfjkLLKH84-23d300LP-78, dev, staging, 94-fjklX-840-HKW45986-JHK...
looker-staging	✓ 12	non-production, 34-ljfjkLLKH84-23d300LP-78, dev, staging, 94-fjklX-840-HKW45986-JHK...
nomad-client	✓ 12	business-events-booking-1, 17-cddkjgtjsdks-4582-dsKthOl, env-prod, marathon, 34-45843...

© 2017 HashiCorp Consul 1.0 Documentation

Consul.io Services

Nodes

Studying « When Failure » 2/5

=> Program fails ?

=> System fails ?

=> need relaunch manually ?

=> diagnostic hardware/software ?

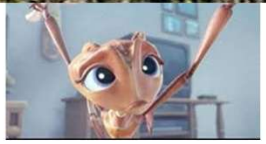
=> interrupt all, repair ?

System = Union of independent components

Each component can be killed
… The system must survive each failure

Be Confident in « System »

Walk on a « Ant » ...



… « Anthill » not in danger

How to Check System does not « Fail » ?

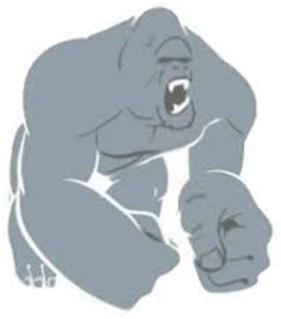
Very difficult to « prove » system correctness

Easy to test : « kill and see »
Not an exhaustive test… Repeat + Changes

Test Kill with Chaos Engineering

Initiated by Netflix

Goal : Randomly kill Process / VM / Datacenter



SLA : 99.99 Up-time ?

= Daily: 8s
Weekly: 1m 0s
Monthly: 4m 22s
Quarterly: 13m 8s
Yearly: 52m 35s

3 nines : 99.999 = Yearly : 5mn15s

4 nines : 99.9999 = Yearly : 31s

Studying « When Failure » 3/5

=> Program fails ?

=> System fails ?

=> need relaunch manually ?

=> diagnostic hardware/software ?

=> interrupt all, repair ?

When Failure ...

=> need relaunch manually ?

Hard-Coded Launch to specific server.. fail

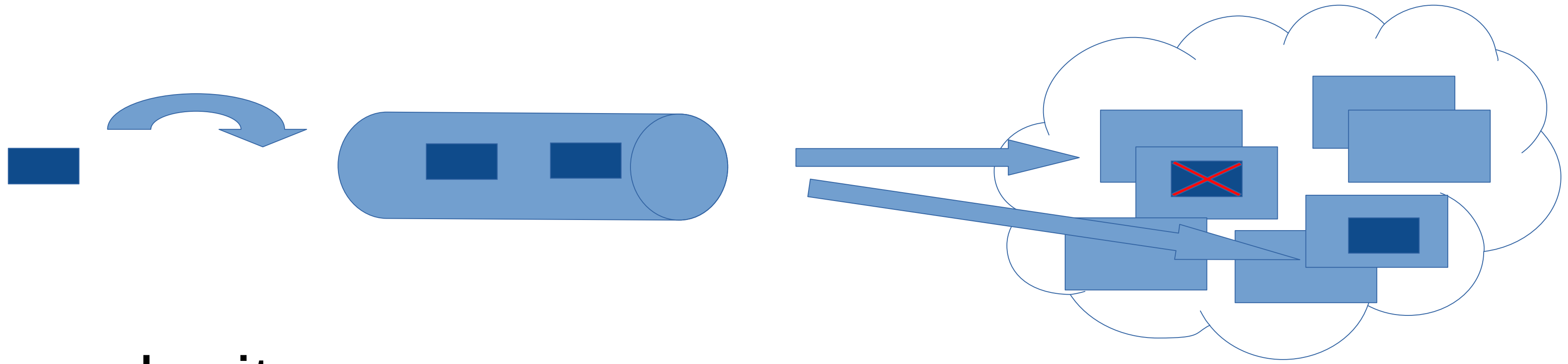
Auto Distributed: RESILIENT
don’t launch manually on a specific server
Let the system select one

Pet vs Cattle

Pet have id=name
You take extra care of it

Cattle >= 1000 : anonymous
Id=Number, interchangeable

Launching => Scheduling on Allocated Resource



You submit
Something to run

You don’t run yourself

internally, it may
wait/allocate resource
do retries...
Not your concern.

Studying « When Failure » 4/5

=> Program fails ?

=> System fails ?

=> need relaunch manually ?

=> diagnostic hardware/software ?

=> interrupt all, repair ?

When Failure ...

=> diagnostic hardware/software ?

NO ? … difficult anyway
Bugs may exist
BUT failure is « not » a bug

Studying « When Failure » 5/5

=> Program fails ?

=> System fails ?

=> need relaunch manually ?

=> diagnostic hardware/software ?

=> interrupt all, repair ?

When Failure ...

=> interrupt all, repair ?

NO …
Immutable-Infrastructure
Do not edit infra once created
Drop and re-create new VMs

When Failure ...

Hardware HOT-PLUG :
unplug old disk, and plug new one
Without interruption

When Failure ...

Idem for Software :
add/remove servers to cluster
(no hard-coded topology/confs)

Duplicate Failable Component for Fewer System Failure

If a component has 0.01 chance to Fail today

Adding another (independent) component…
=> Probability that 2 fail today = 0.01*0.01 = 0.0001

=> Probability that 3 fail today = 0.01^3 = 0.000001 = 1e-6

« If » your system still works with 1 working component out of 2 … better

System ≥ Component
If No Correlated / Dispatch / No Spof

Electric power : when fail … all fails   (correlated failures)

Need reliable network + dispatching to working components (retry/detect failed ones)

Everything between the components can be a « SPOF »
Network may be a « Single Point Of Failure »
as all forgotten components not redundant

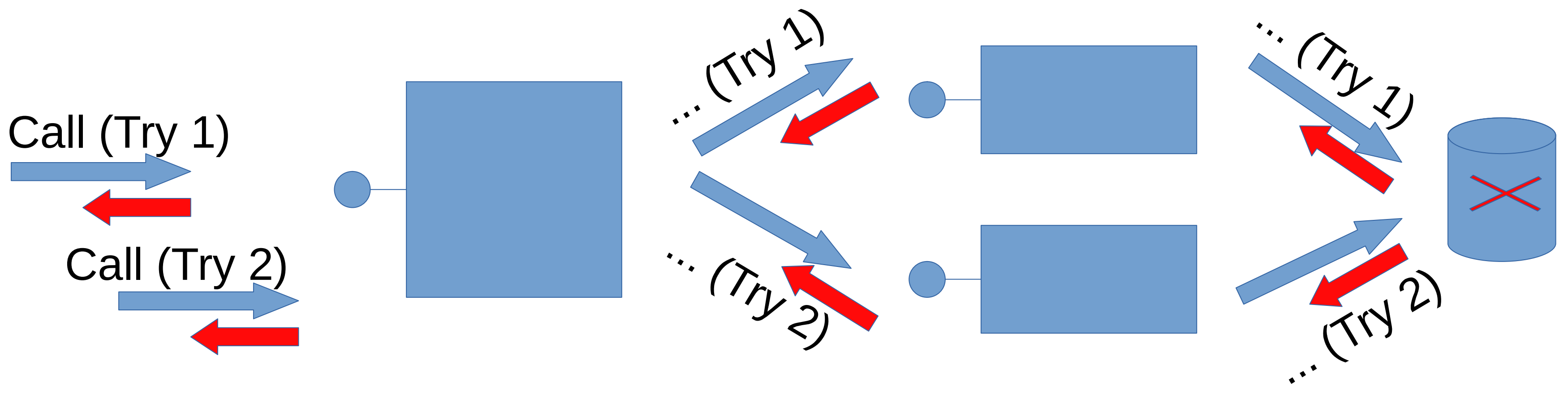
Story of Arianne 501
Duplicated / « Correlated » Errors...

Flight computation performed twice in parallel on 2 isolated hardwares

But using same program…
Both programs throw same «overflow exception » at exact same millis

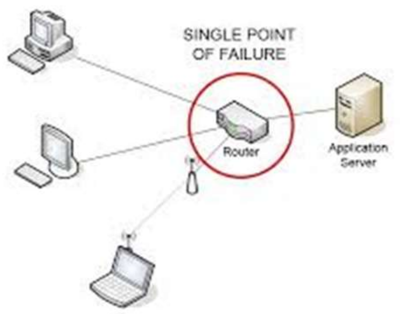
=> Lessons learned : use 2 hardwares + 2 independent softwares + ..

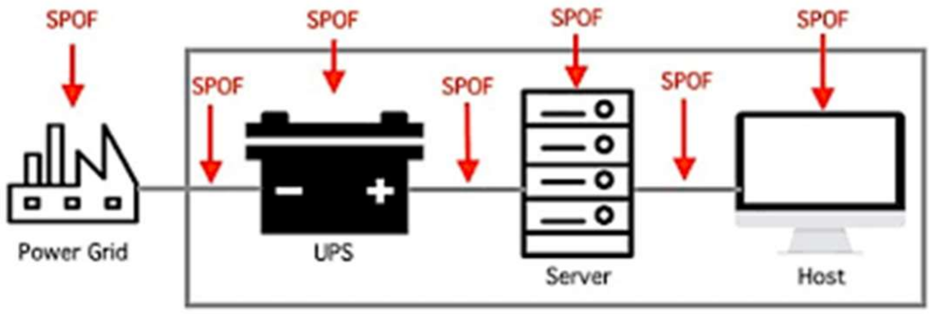
Single Point Of Failure ?

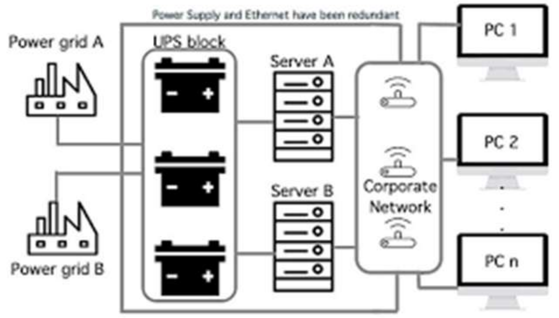


High Availability Servers…
But Shared State / Database .. SPOF

Examples of SPOF





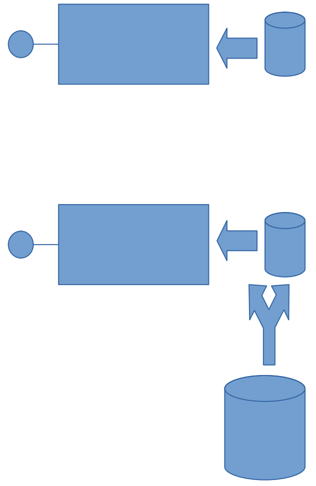


NO SPOF : Duplicate Everything

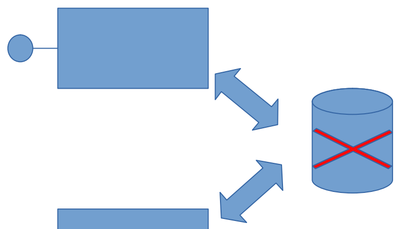
But how to duplicate a single Source of Truth « Data » ?

Copy => stale data / replication / distributed lock ?

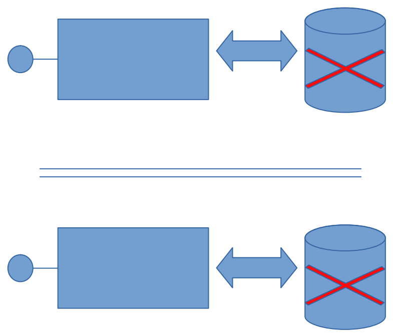
Stateless, Spof, Sharded (easy) vs Statefull (difficult)



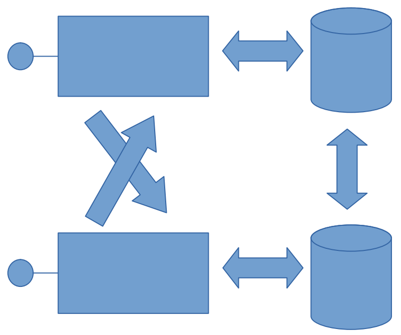
Stateless
(or stale Read-Only
cached data)



Spof



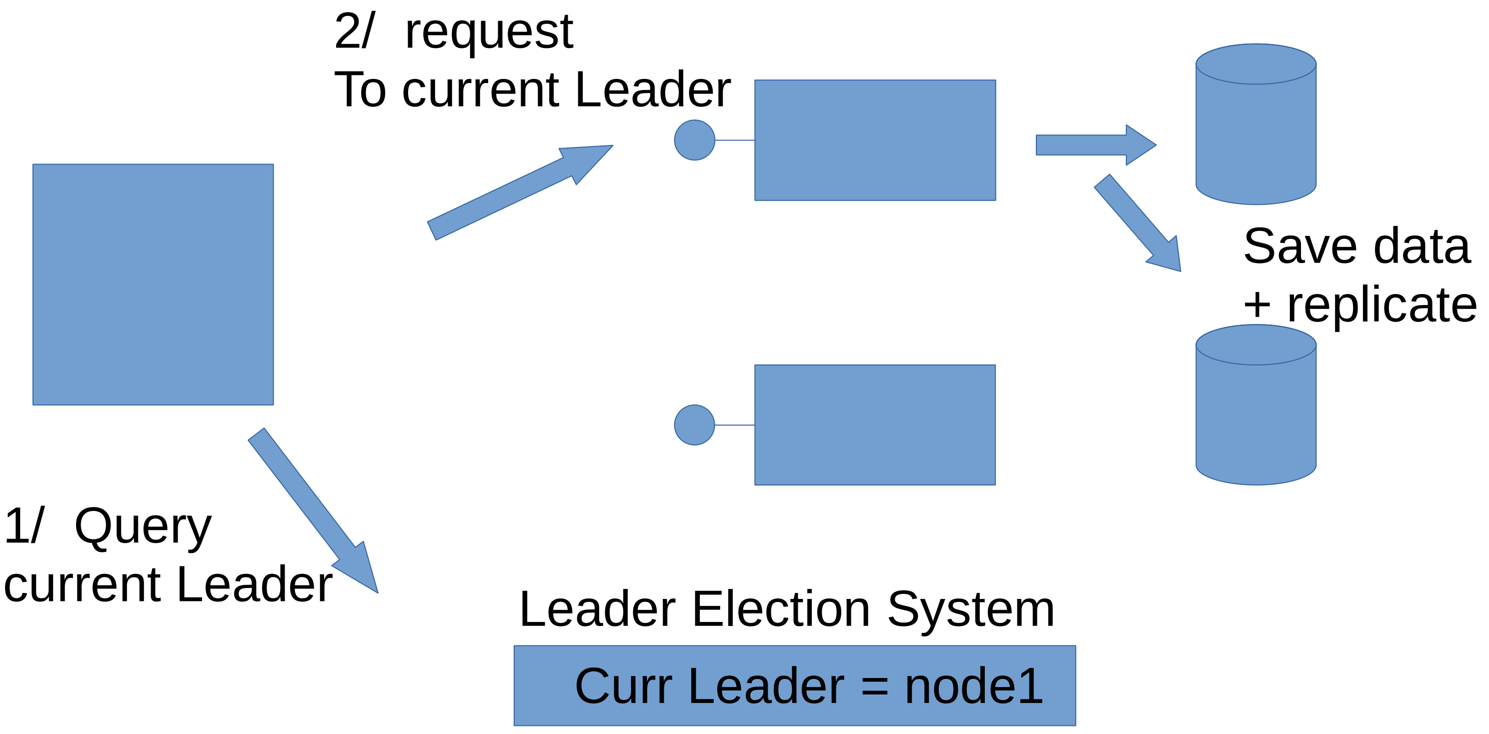
Sharded… 2 spofs
(ex : id modulo 2)
Horizontal scale : OK
But not replicated



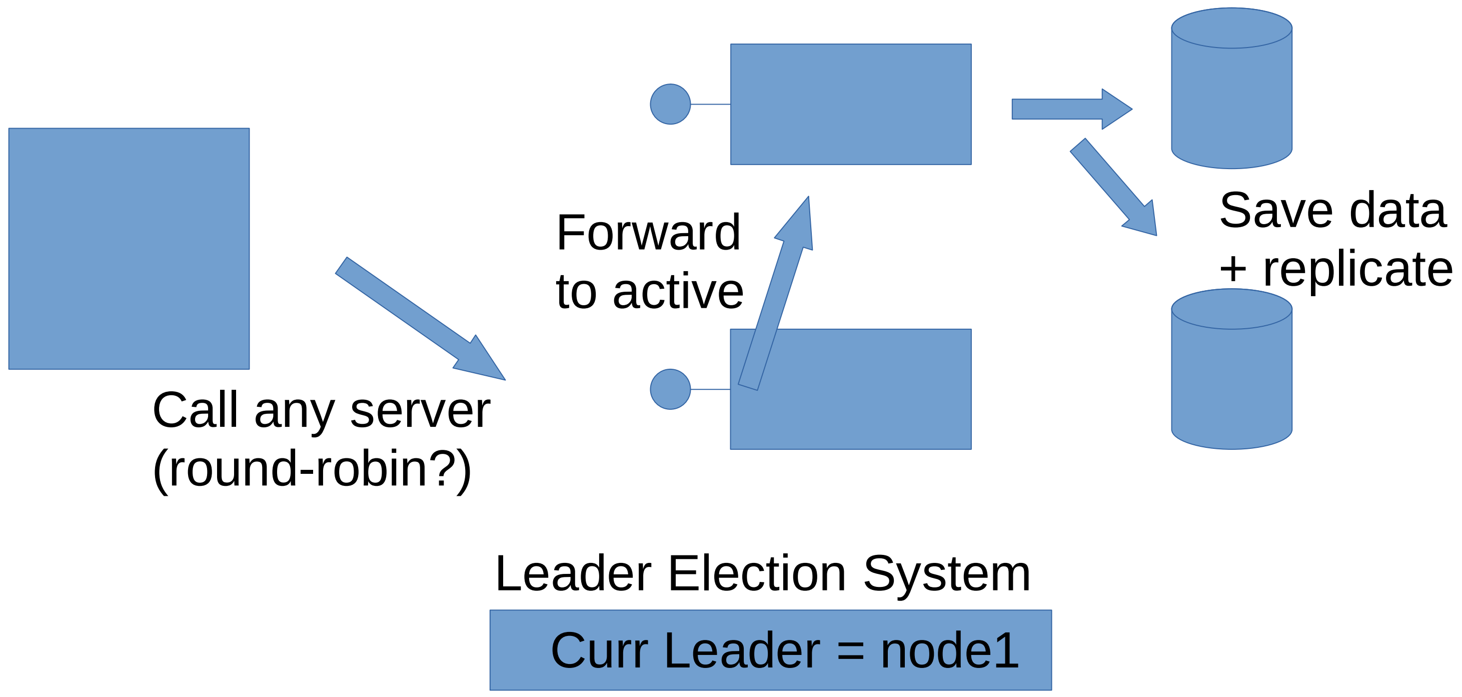
Distributed Lock ?
Replication ?
Source of truth ?

Statefull
Concurrent Distributed Data

« HA » High Availability
Active – Standby + Replication



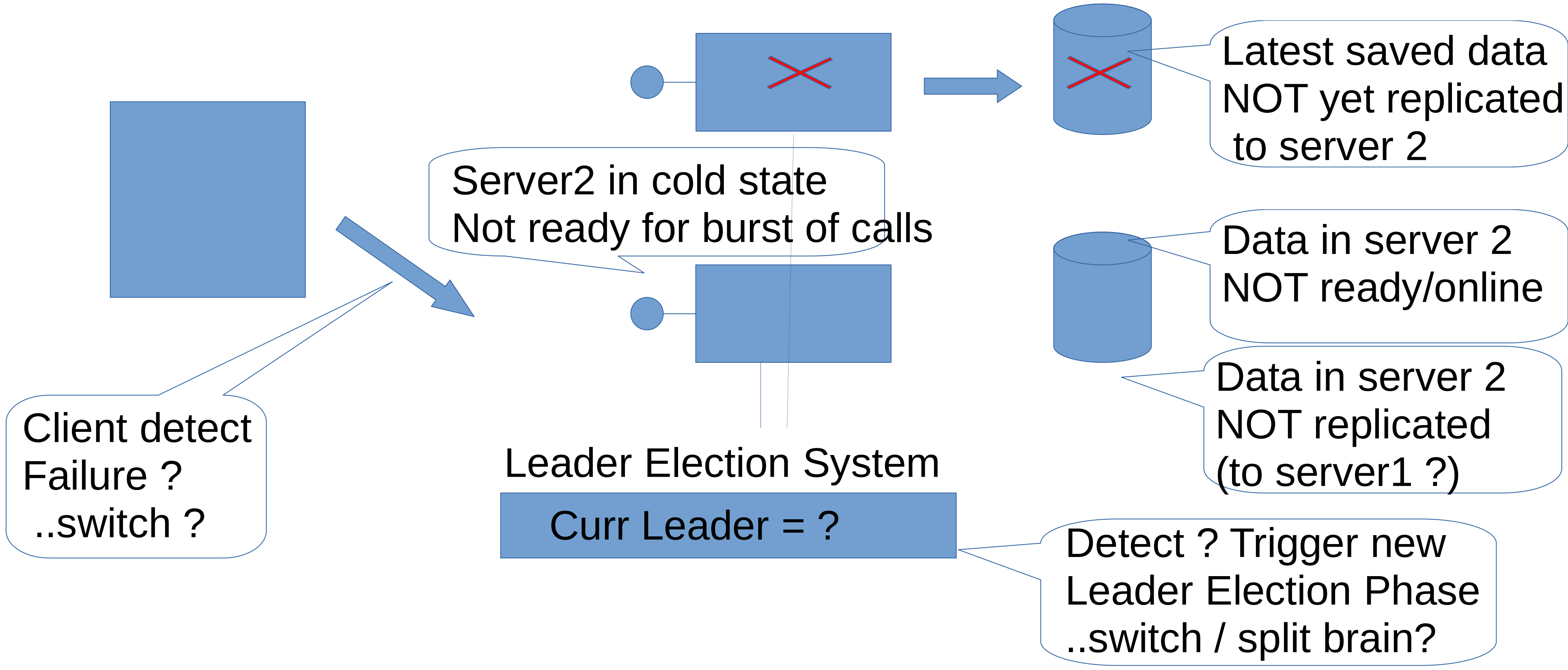
Transparent Delegate to Active



Synonym Terms...

Master – Slaves (not polically correct)
Leader – Followers
Active – StandBys
Primary – Secondaries
Main – Replica
MainSite - Disaster Recovery
MainDisk - BackupDisk

Switching from Active to StandBy



Problem with Leaders same as in Political

works only when there is exactly 1 leader

≥ 2 leaders … conflicts / Split-Brains / Network Partitioning

0 Leader … nothing works

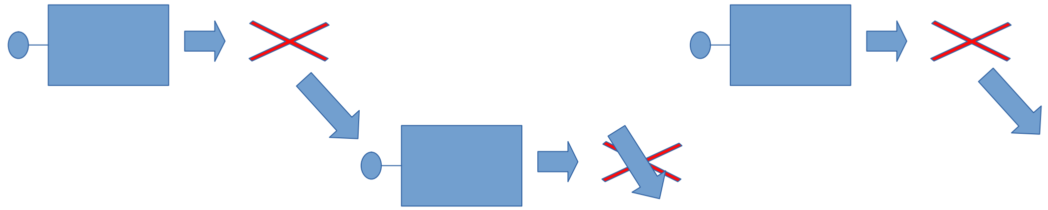
Imposters pretend to be Leader
Leader staying leader too long after new election

Slow to organize election
Trust the result of election ?
Who can organize if there is no leader?

Can’t decide election in 50 % / 50 % equality … need an odd number : 3, 5, 7..
There must be at least 50 % participation (no Abstention)
Otherwise both candidates pretend to majority (not absolute majority) with 50 %

Leader fail to « Start Lead » … re-electing too Often ?

Example of Catastrophic scenario :



1/ Server « N » become Leader

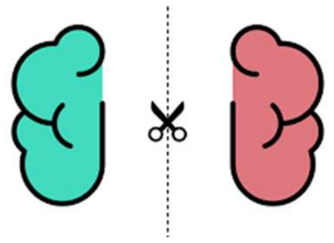
2/ It needs huge memory to serve requests
There are too many requests (burst of waiting requests)

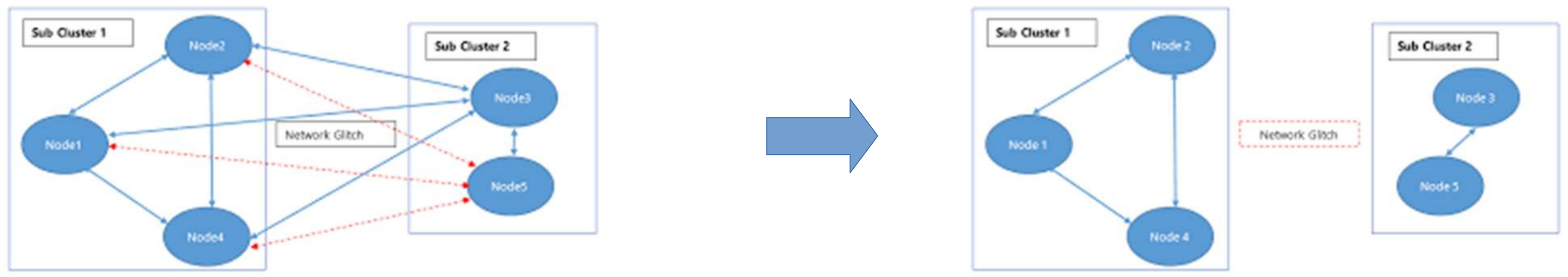
3/ « Full GC » occurs, take >=30s , and « stop-the-world »
30s is the default Timeout for socket connection / read response

… During this time, Leader fails to answer to « Health check »
4/ Leader is considered not healthy => not leader any more

=> 5/ Electing another Leader « N+1 » => back to beginning 1/

Split Brain / Network Partitionning





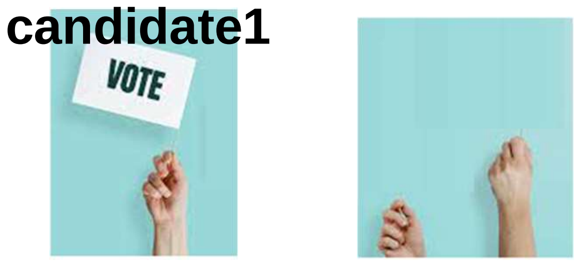
1 cluster, 1 master
status=OK
… but network partition happens

2 clusters fully isolated each
Each have 1 master
Each have status=OK
… data diverge, will discover conflict on merge

Quorum of 50 %

Quorum of >50 % reachs => elect candidate1

NO Quorum of 50 % reachs
=> re-organize new election
(proba=0.5 that 2 of 3 vote for same)



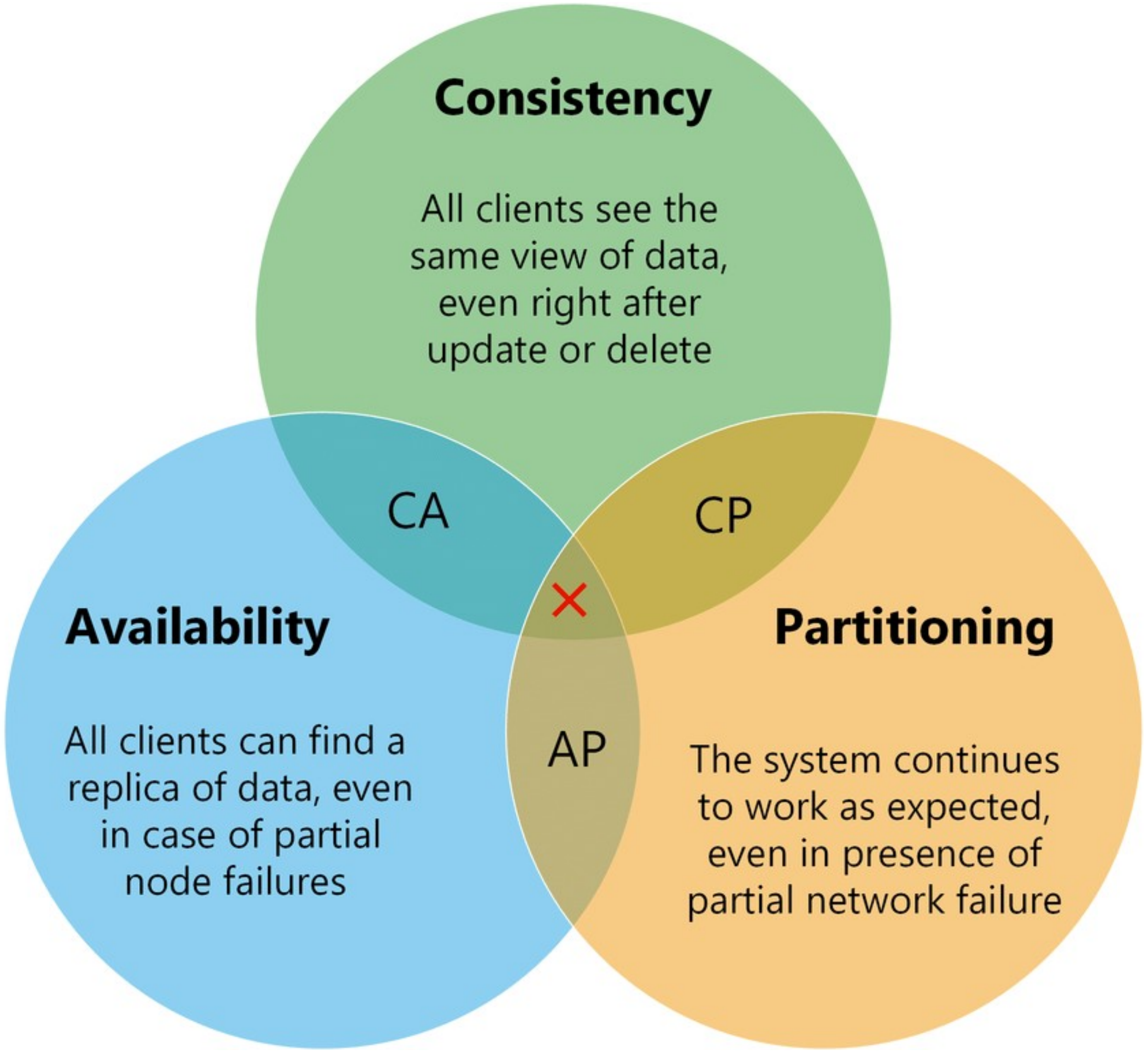
Majority but NO Quorum
… high risk of Split-Brain
(all others may vote and you are not aware)
=> wait enough voters (re-organize / accept vote)

Waiting Quorum
= system « Not Available »

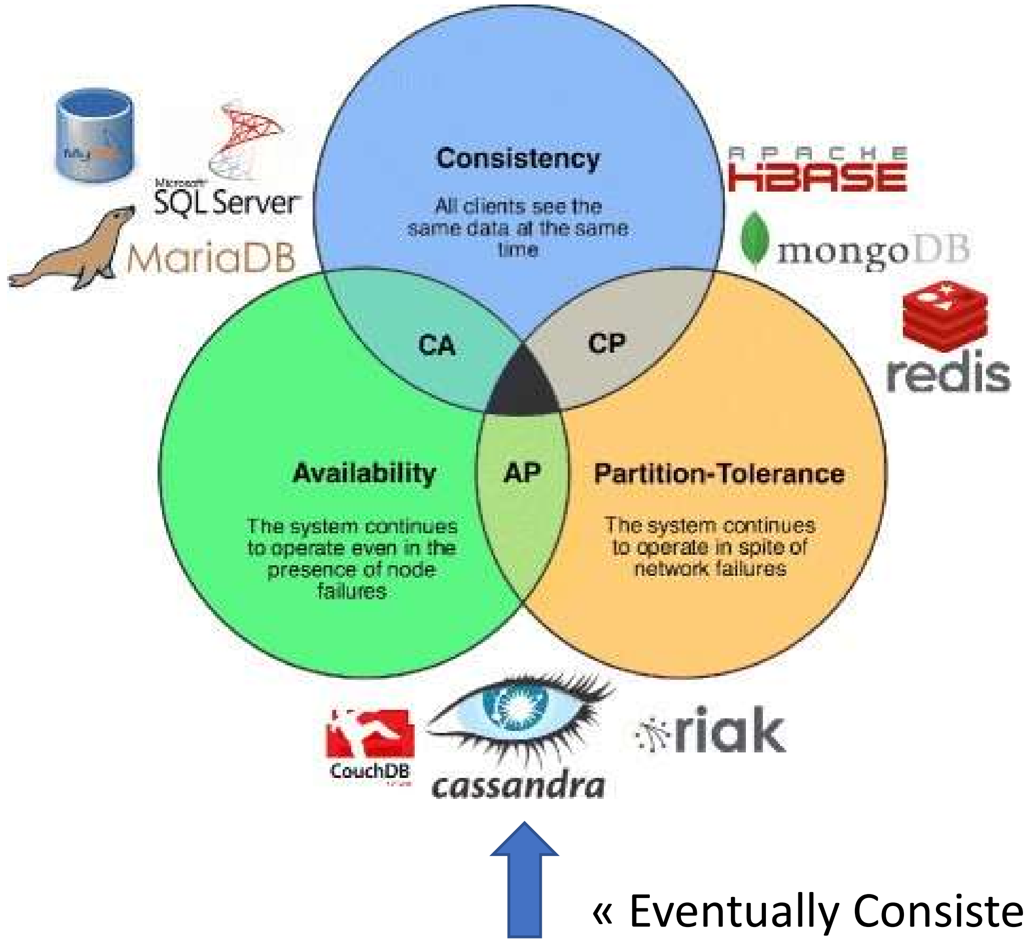
Waiting Quorum

<= 50 %
… means cluster 0 % working

CAP Theorem .. Choose 2 out of 3


Consistency
All clients see the same view of data, even right after update or delete
CA
CP
×
Availability
Partitioning
AP
All clients can find a replica of data, even in case of partial node failures
The system continues to work as expected, even in presence of partial network failure

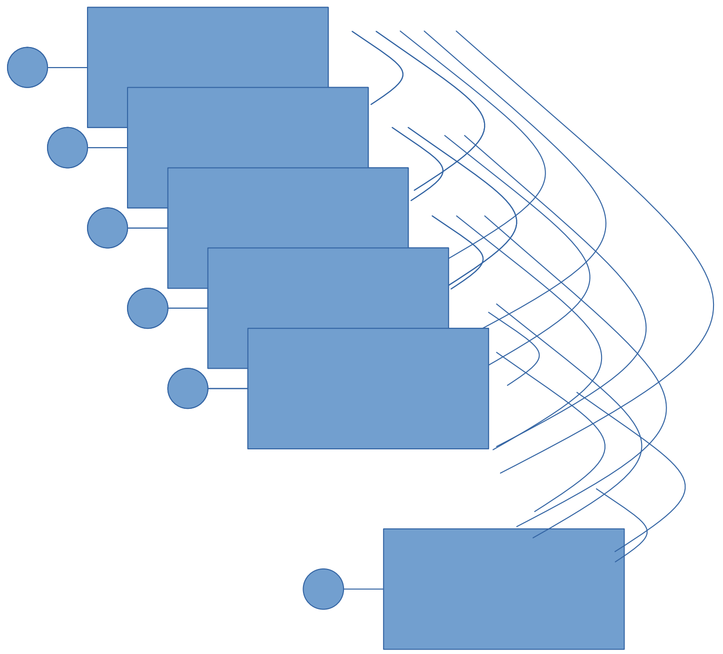
Databases choices

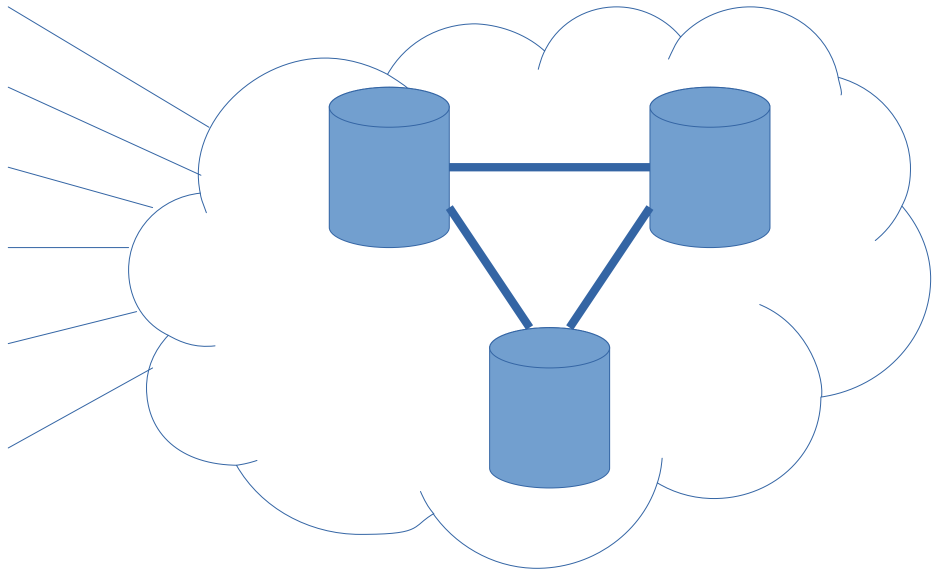


Consistency
All clients see the same data at the same time
Availability
The system continues to operate even in the presence of node failures
Partition-Tolerance
The system continues to operate in spite of network failures
CA
CP
AP
SQL Server
MariaDB
APACHE HBASE
mongoDB
redis
CouchDB
cassandra
riak

« Eventually Consistent »
.. Means « NOT » always consistent … but ends to be

Distributed Coordination Server(s)





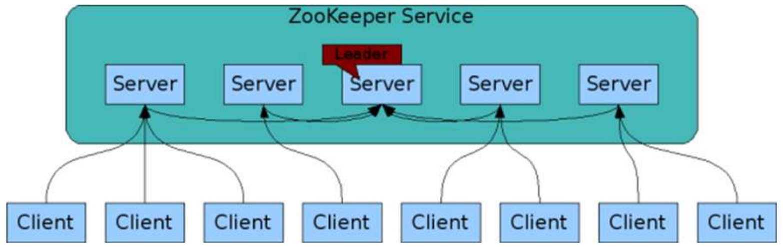
Difficult to synchronize data on N servers
N servers => 1/2 * N * (N-1) connections .. too many
Slow to organize election

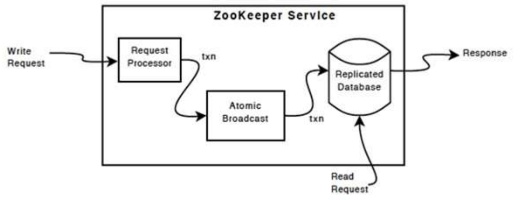
Small subsystem (3,5,7 nodes)
for quorum / election
Storing coordination data
N servers => N+3 connections

ZooKeeper

Because Coordinating Distributed Systems is a Zoo

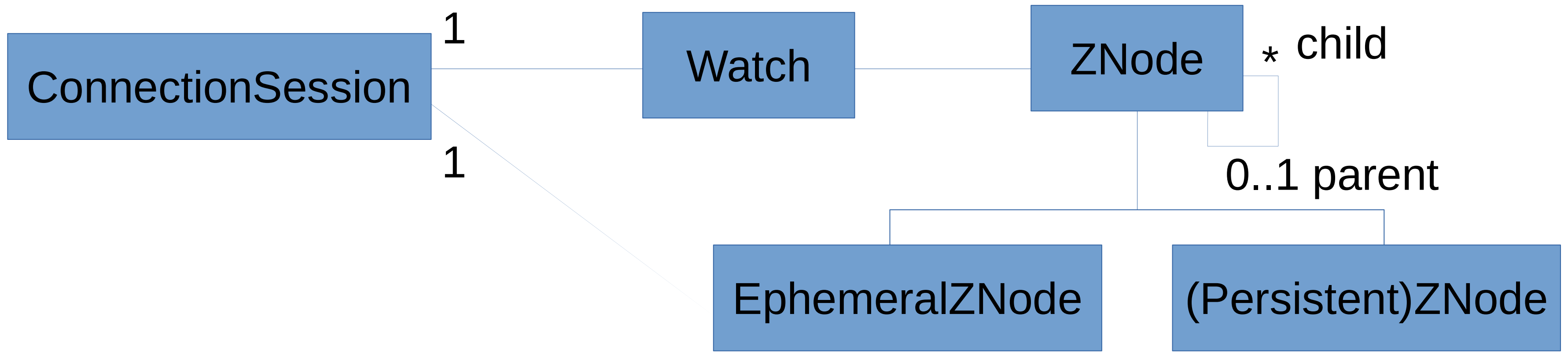
Zookeeper is **CP** (Consistent and Partition Tolerant)
Not **A** (Available)





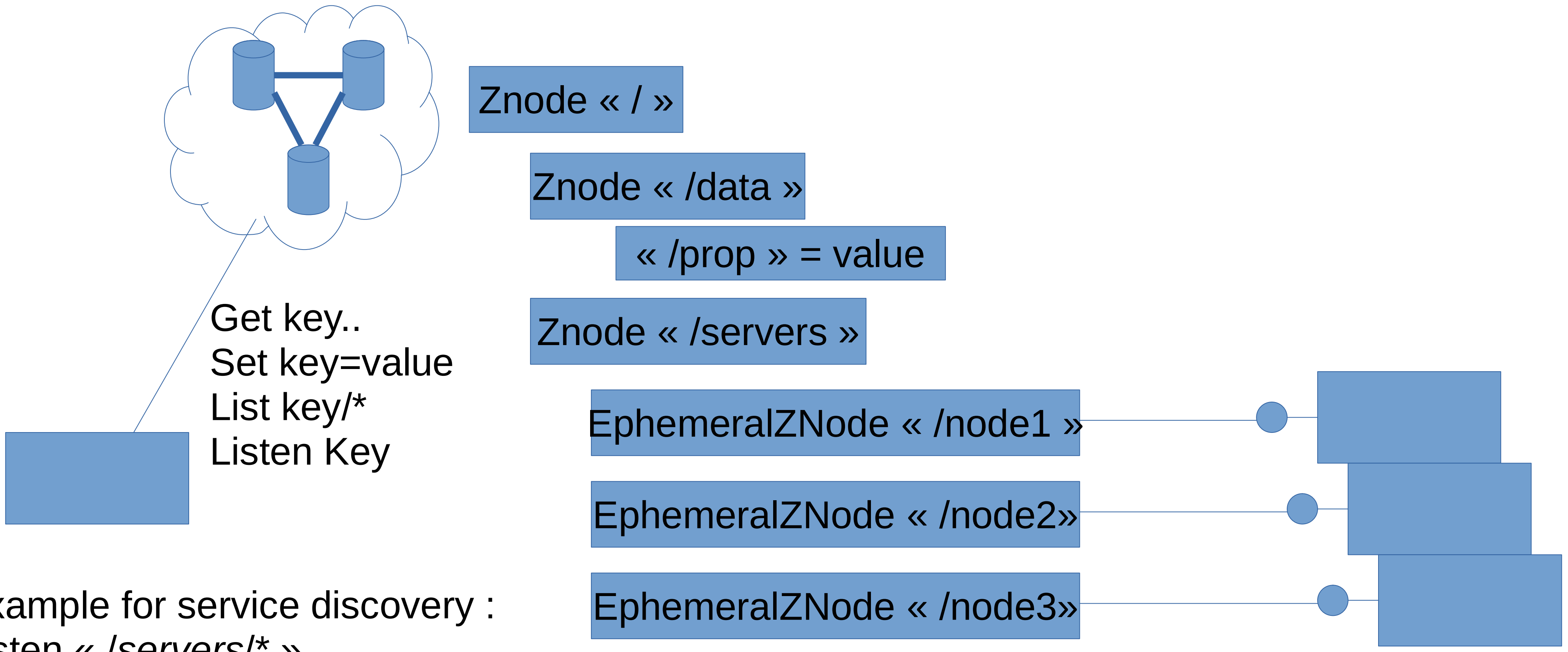
All writes are serialized to Leader (exclusive lock)
+ replicated to quorum for committing

ZooKeeper Features



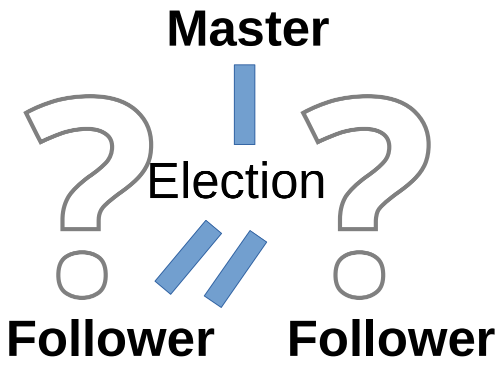
Zookeeper kernel : « key=value » database, with atomic features
Organized as directories hierarchy, called Znode
With Listening capabilities
And Ephemeral Znode for connections

ZooKeeper

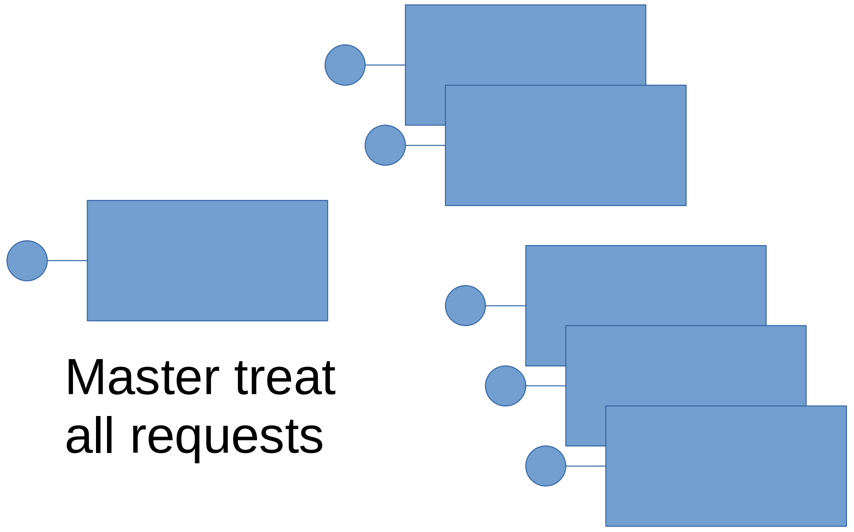


Example for service discovery :
Listen « */servers/** »

ZooKeeper : for Master Election + Persist Topology/Metadata Infos



1 Master for All => does not scale



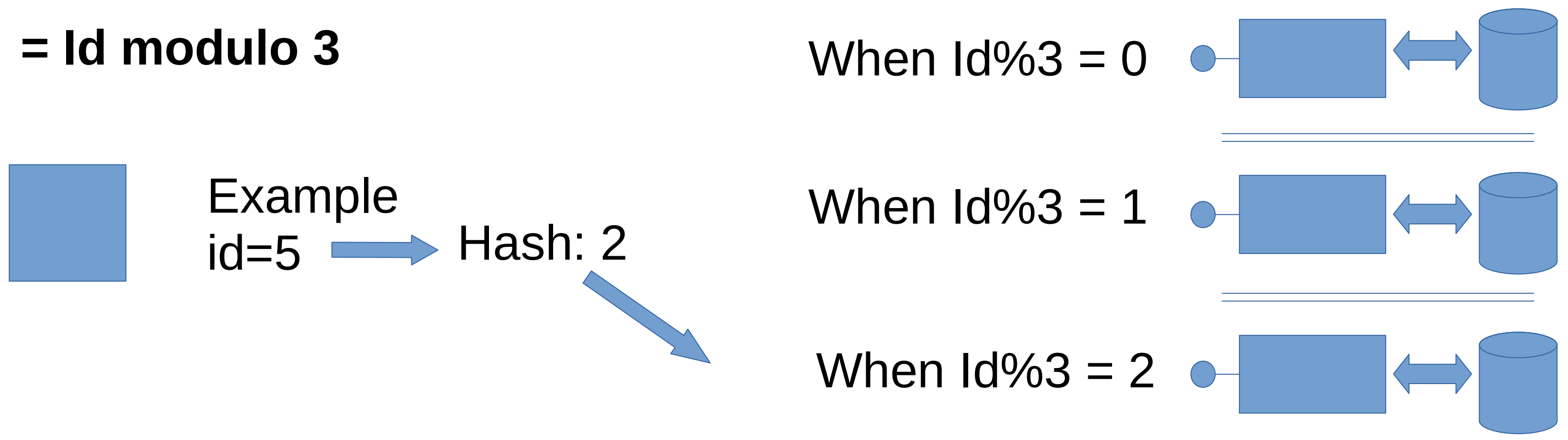
Master treat
all requests

Followers IDDLE
in StandBy Mode
(waste of cpu)

Sharding for Horyzontal Scaling

Sharding = choose responsible server from data Id
(shared rule for clients and servers)

ServerId = Id modulo 3



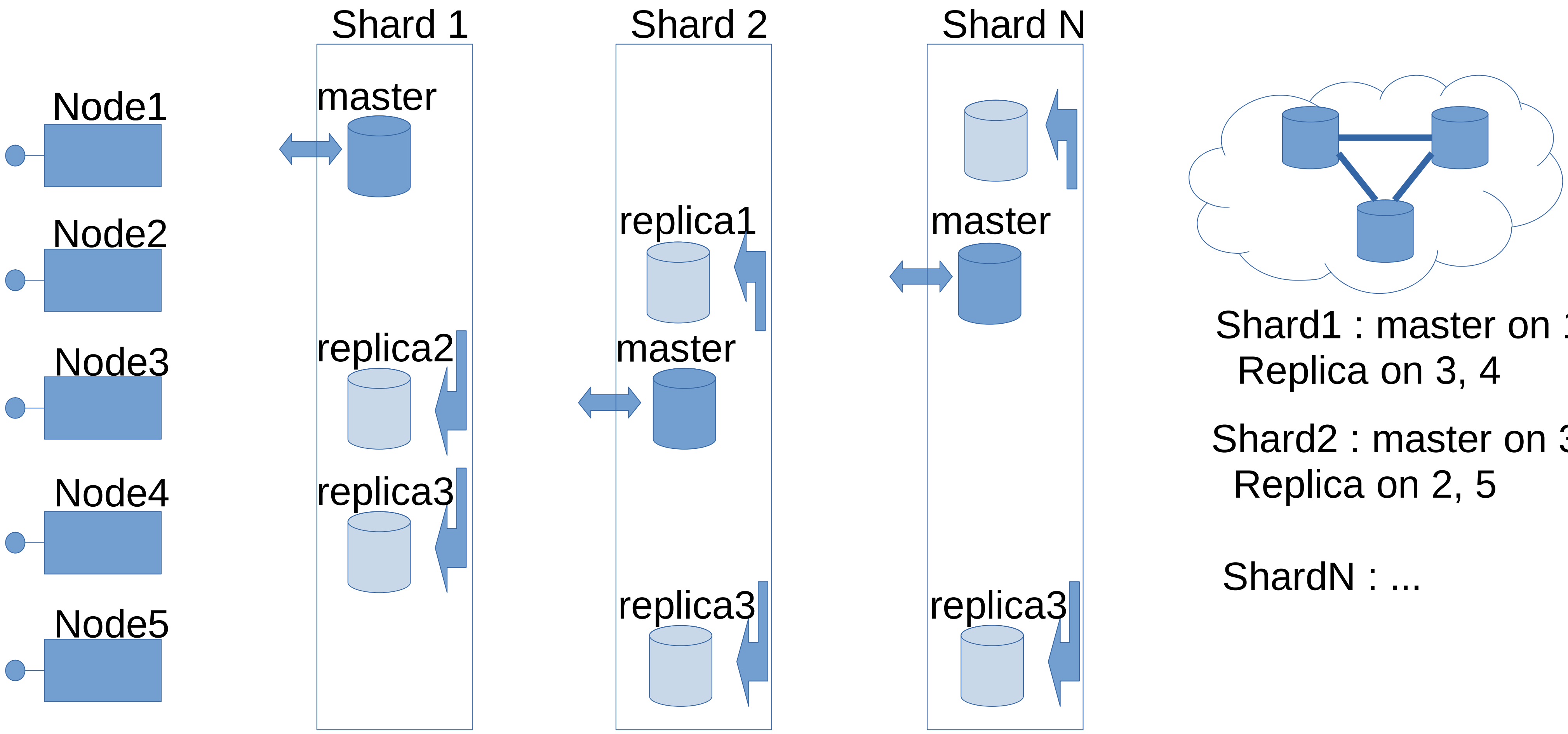
Sharding .. Pros/Cons

Scale very well « linearly » with number of servers

ONLY for request with known « ID »
(not for search / full scan)

Difficult to redimension « N » at runtime… need re-shuffle all data !

1 Master per Shard



Shard1 : master on 1
Replica on 3, 4

Shard2 : master on 3
Replica on 2, 5

ShardN : ...

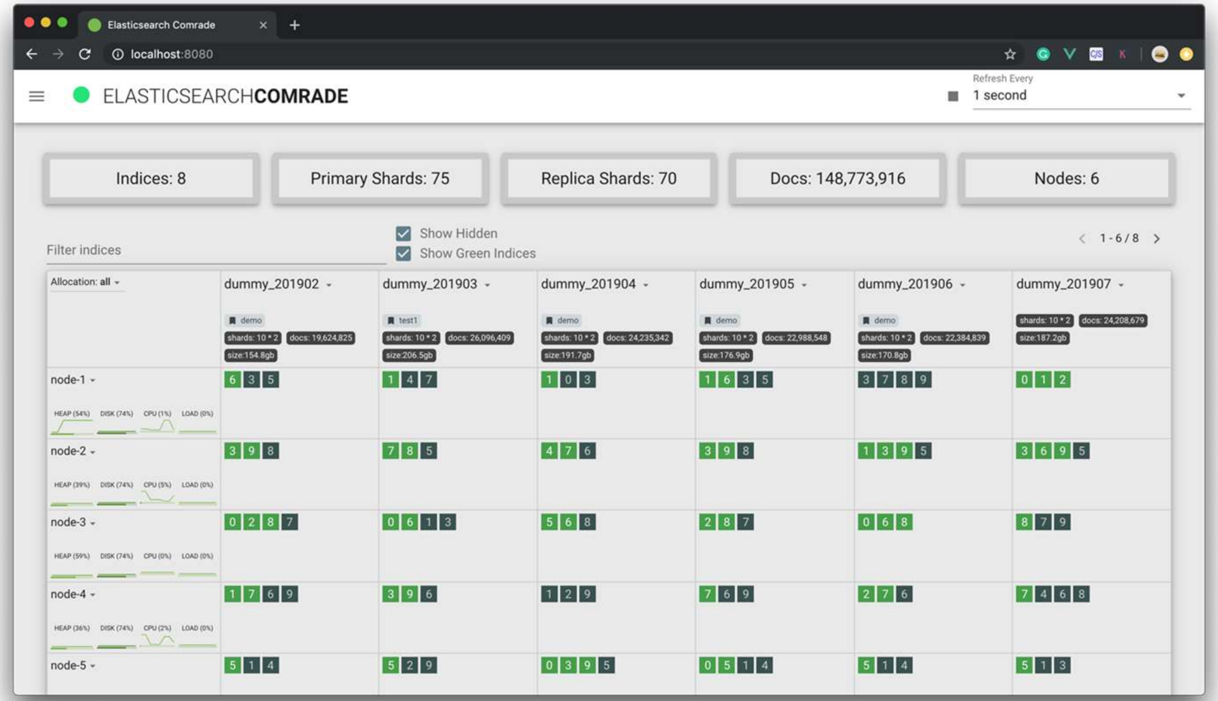
3 Examples & Comparisons for Sharding + Master/Replica

elasticsearch

hadoop HDFS

APACHE HBASE

Example 1/3 : ElasticSearch



Refresh Every: 1 second

Indices: 8 | Primary Shards: 75 | Replica Shards: 70 | Docs: 148,773,916 | Nodes: 6

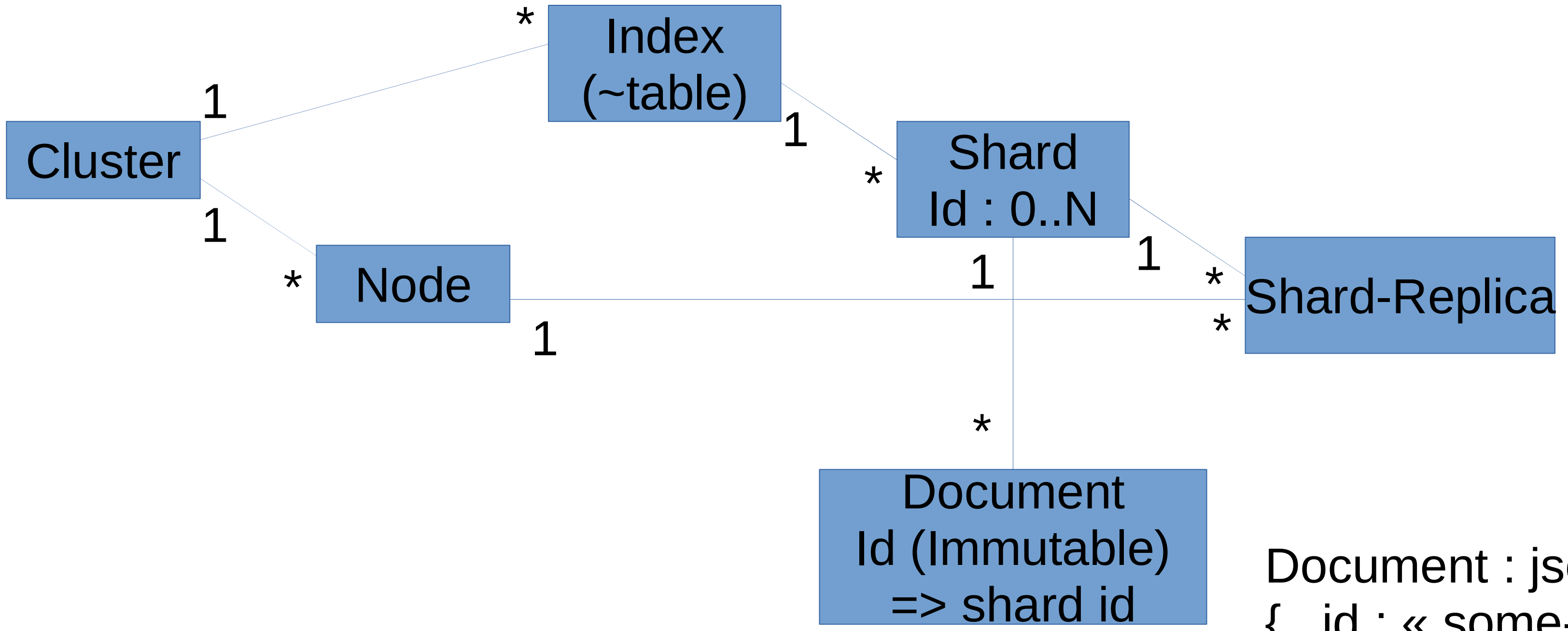
Filter indices

☑ Show Hidden
☑ Show Green Indices

1 - 6 / 8

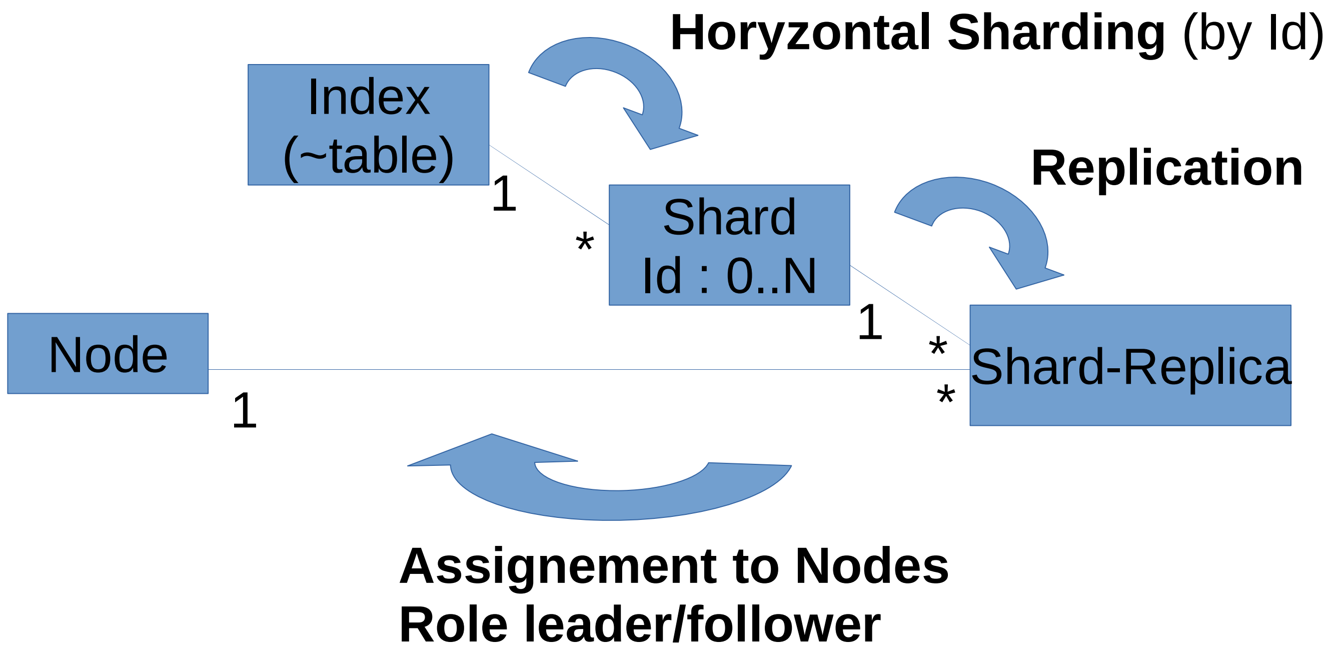
Allocation: all	dummy_201902	dummy_201903	dummy_201904	dummy_201905	dummy_201906	dummy_201907
	demo shards: 10 * 2 docs: 19,624,825 size:154.8gb	test1 shards: 10 * 2 docs: 26,096,409 size:206.5gb	demo shards: 10 * 2 docs: 24,235,342 size:191.7gb	demo shards: 10 * 2 docs: 22,988,548 size:176.9gb	demo shards: 10 * 2 docs: 22,384,839 size:170.8gb	shards: 10 * 2 docs: 24,208,679 size:187.2gb
node-1 HEAP (54%) DISK (74%) CPU (1%) LOAD (0%)	6 3 5	1 4 7	1 0 3	1 6 3 5	3 7 8 9	0 1 2
node-2 HEAP (39%) DISK (74%) CPU (5%) LOAD (0%)	3 9 8	7 8 5	4 7 6	3 9 8	1 3 9 5	3 6 9 5
node-3 HEAP (59%) DISK (74%) CPU (0%) LOAD (0%)	0 2 8 7	0 6 1 3	5 6 8	2 8 7	0 6 8	8 7 9
node-4 HEAP (36%) DISK (74%) CPU (2%) LOAD (0%)	1 7 6 9	3 9 6	1 2 9	7 6 9	2 7 6	7 4 6 8
node-5	5 1 4	5 2 9	0 3 9 5	0 5 1 4	5 1 4	5 1 3

ElasticSearch … UML model

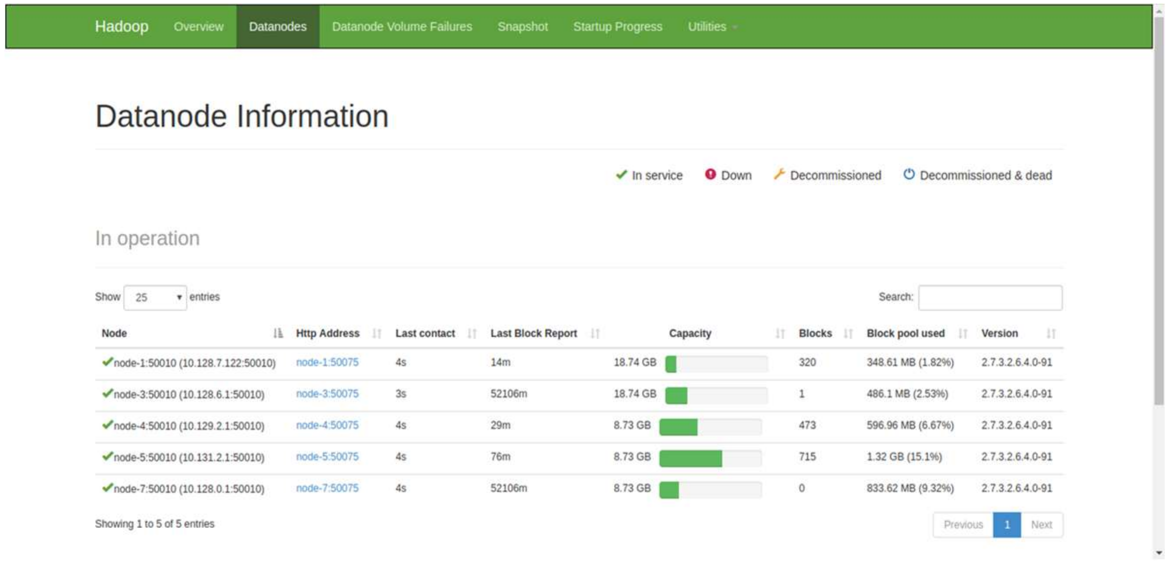


Document : json text
{ _id : « some-id1 »,
  field1 : 123,
  field2 : { subField : [« a »] }
}

ElasticSearch …
Zooming Relations



HDFS



Datanode Information

✓ In service ● Down Decommissioned Decommissioned & dead

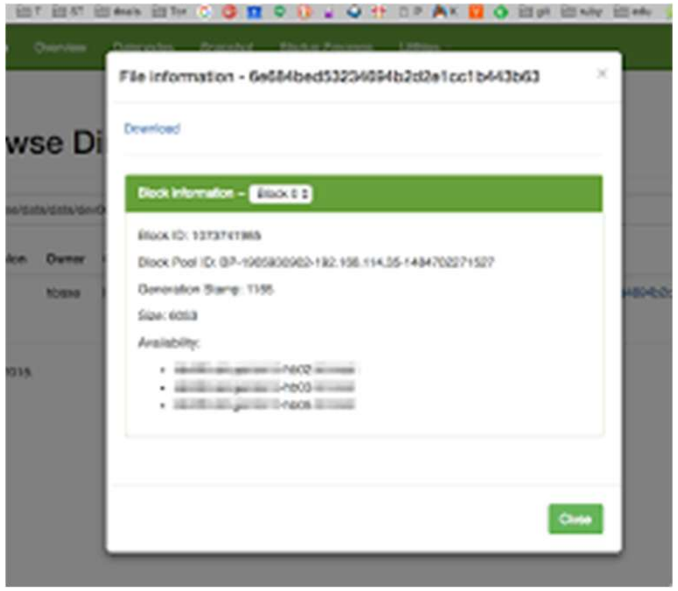
In operation

Show 25 entries Search:

Node	Http Address	Last contact	Last Block Report	Capacity	Blocks	Block pool used	Version
✓node-1:50010 (10.128.7.122:50010)	node-1:50075	4s	14m	18.74 GB	320	348.61 MB (1.82%)	2.7.3.2.6.4.0-91
✓node-3:50010 (10.128.6.1:50010)	node-3:50075	3s	52106m	18.74 GB	1	486.1 MB (2.53%)	2.7.3.2.6.4.0-91
✓node-4:50010 (10.129.2.1:50010)	node-4:50075	4s	29m	8.73 GB	473	596.96 MB (6.67%)	2.7.3.2.6.4.0-91
✓node-5:50010 (10.131.2.1:50010)	node-5:50075	4s	76m	8.73 GB	715	1.32 GB (15.1%)	2.7.3.2.6.4.0-91
✓node-7:50010 (10.128.0.1:50010)	node-7:50075	4s	52106m	8.73 GB	0	833.62 MB (9.32%)	2.7.3.2.6.4.0-91

Showing 1 to 5 of 5 entries

Previous 1 Next



File information - 6e684bed53234094b2d2e1cc1b443b63

Download

Block information – Block 0

Block ID: 1073741865

Block Pool ID: BP-1905800902-192.168.114.35-1484702271527

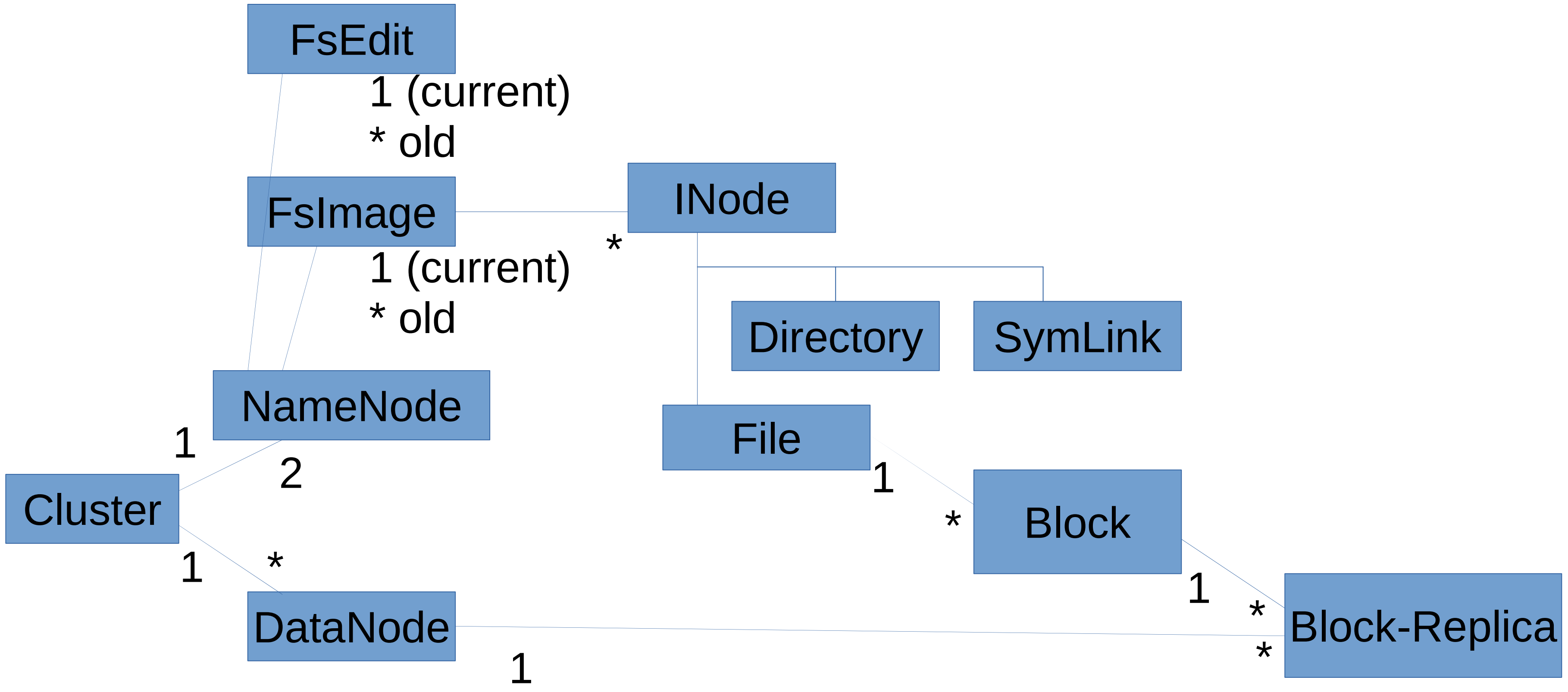
Generation Stamp: 1155

Size: 6053

Availability:

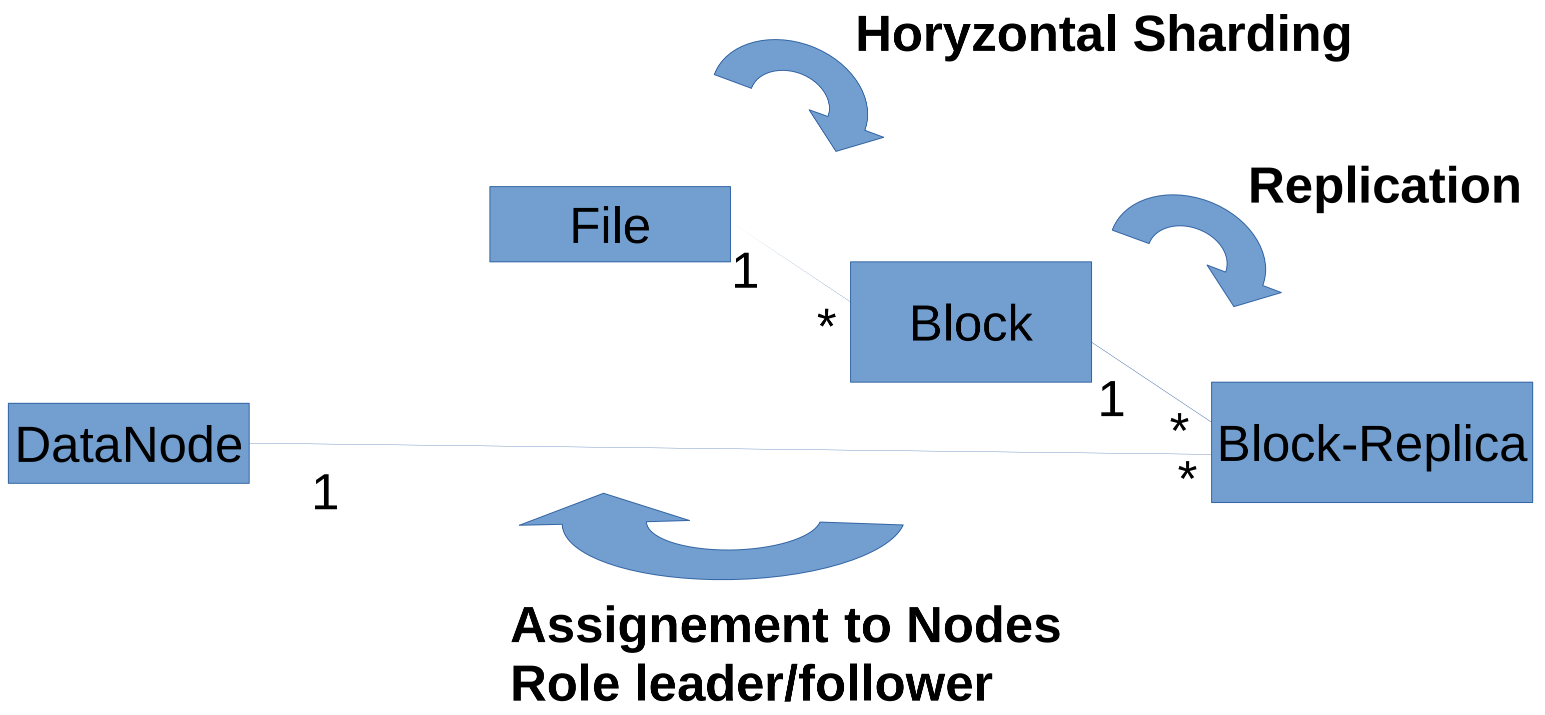
- [redacted]-hb02 [redacted]
- [redacted]-hb03 [redacted]
- [redacted]-hb05 [redacted]

HDFS … UML Model

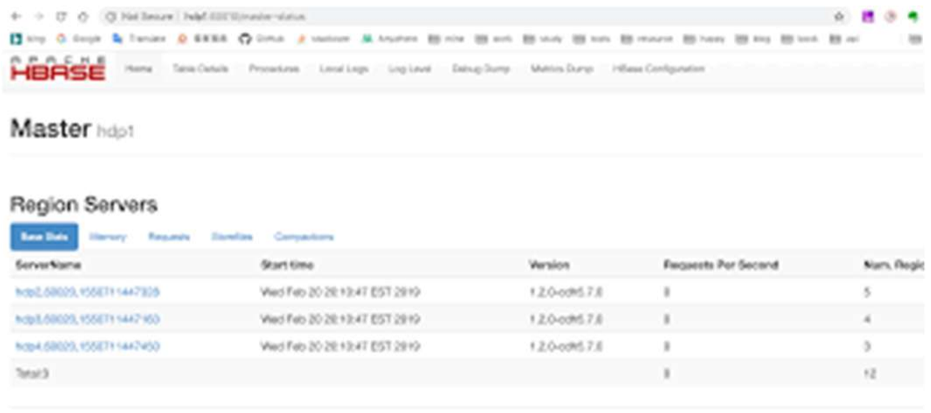


File: binary large object « blob »
= metadata
(owner, group, chmod, acl, ..)
+ list of blocks

HDFS … Zoomin Relations



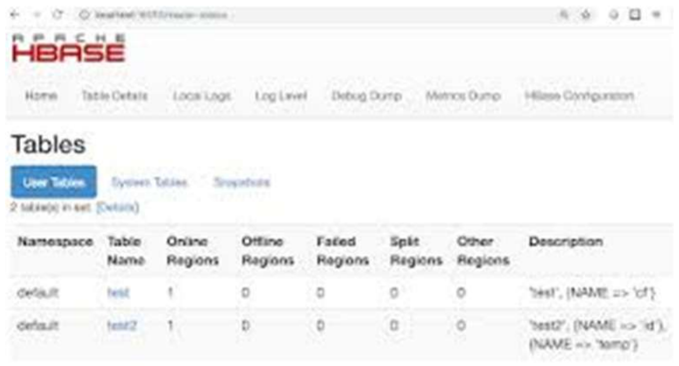
HBase



Master hdp1

Region Servers

ServerName	Start time	Version	Requests Per Second	Num. Regic
hdp2,60020,1550711447326	Wed Feb 20 20:13:47 EST 2019	1.2.0-cdh5.7.6	0	5
hdp3,60020,1550711447160	Wed Feb 20 20:13:47 EST 2019	1.2.0-cdh5.7.6	0	4
hdp4,60020,1550711447450	Wed Feb 20 20:13:47 EST 2019	1.2.0-cdh5.7.6	0	3
Total:3			0	12

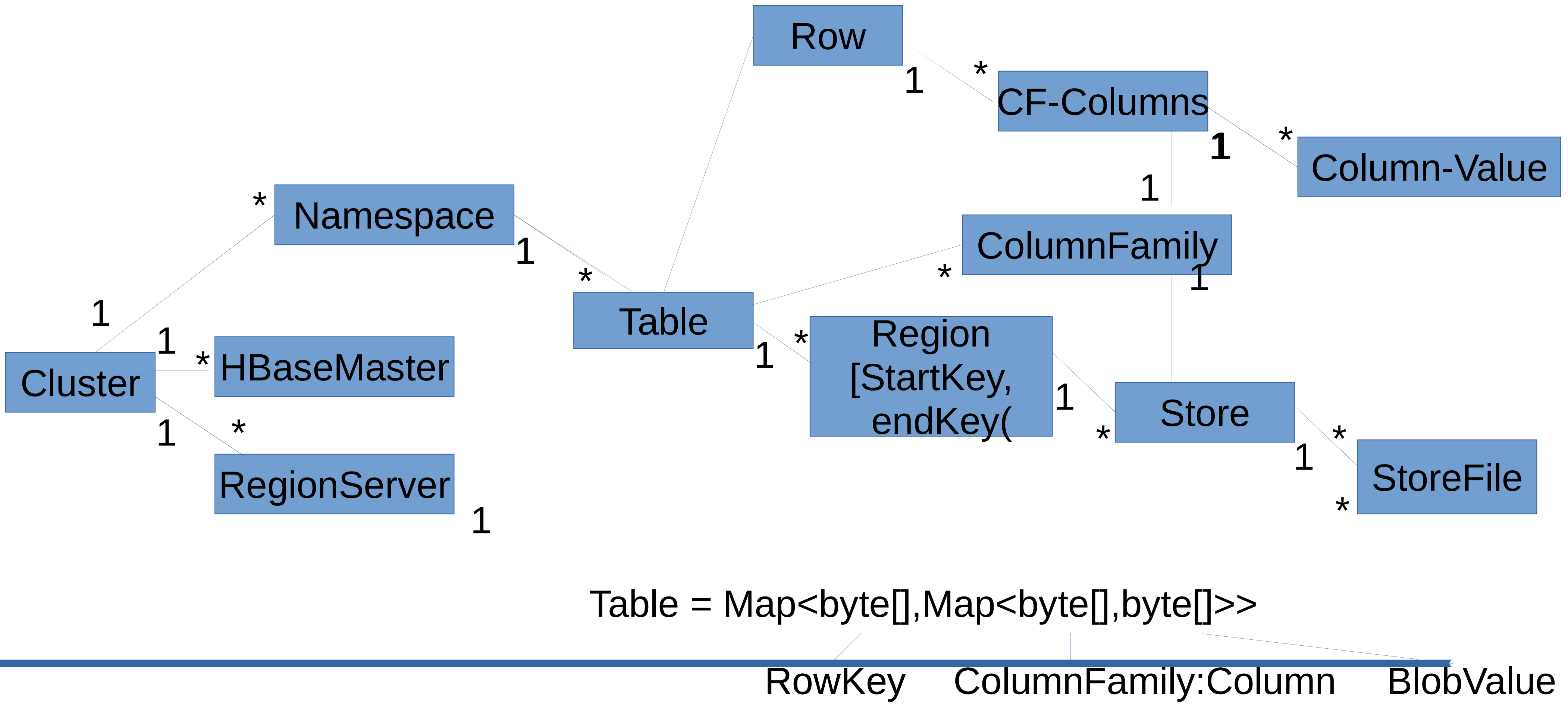


Tables

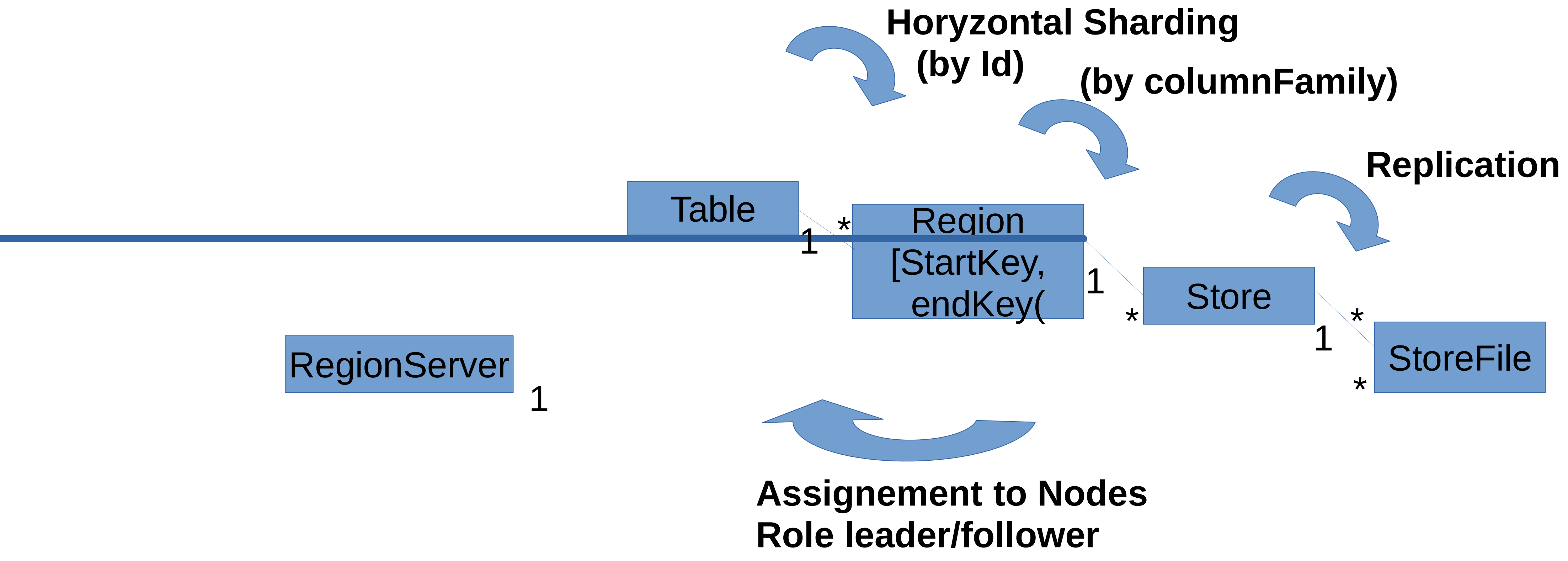
2 table(s) in set. [Details]

Namespace	Table Name	Online Regions	Offline Regions	Failed Regions	Split Regions	Other Regions	Description
default	test	1	0	0	0	0	'test', {NAME => 'cf'}
default	test2	1	0	0	0	0	'test2', {NAME => 'id'}, {NAME => 'temp'}

HBase … UML Model



HBase … Zooming Relations



Questions ?