สยามชำนาญกิจ
SIAM CHAMNAN KIT

Redis Workshop

Somkiat Puisungnoen

Somkiat | Home

Somkiat Puisungnoen

Update Info 1 | View Activity Log 10+ | ...

Timeline | About | Friends 3,138 | Photos | More

When did you work at Opendream?

22 Pending Items

Intro

Software Craftsmanship

Software Practitioner at สยามชำนาญกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Post | Photo/Video | Live Video | Life Event

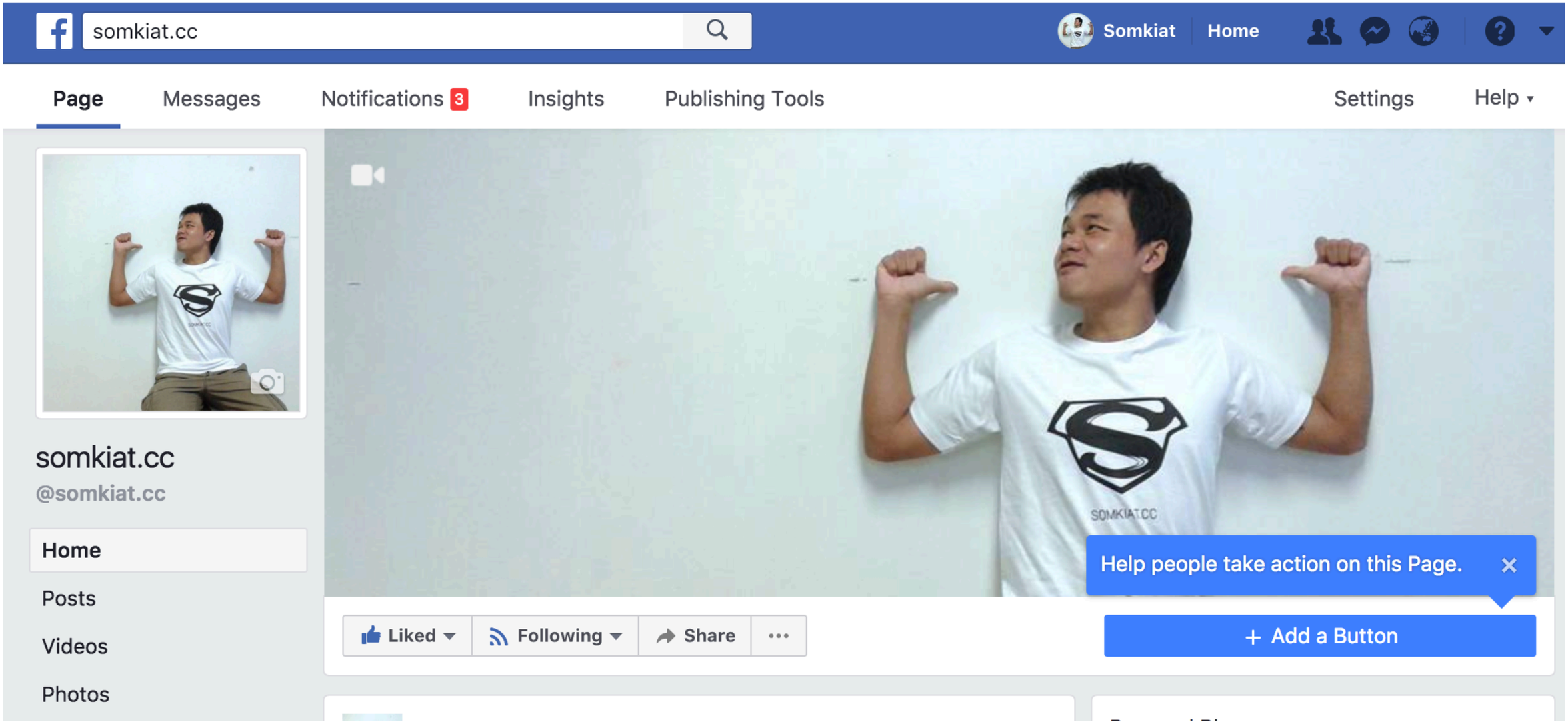
What's on your mind?

Public | Post

Somkiat Puisungnoen

15 mins · Bangkok ·

Java and Bigdata



Agenda

- Single instance
- Sentinel
- Cluster
- Configuration and Monitoring

https://github.com/up1/course-imc-devops-5-days/tree/main/database

Compare with Others

	Redis	Memcached	MongoDB
In-memory	X	X	
Persistent	X		X
Key-value store	X	X	
Supports more than strings	X		X
Multithreaded		X	X
Supports larger-than-memory dataset			X
As fast as	Memory	Memory	Disk

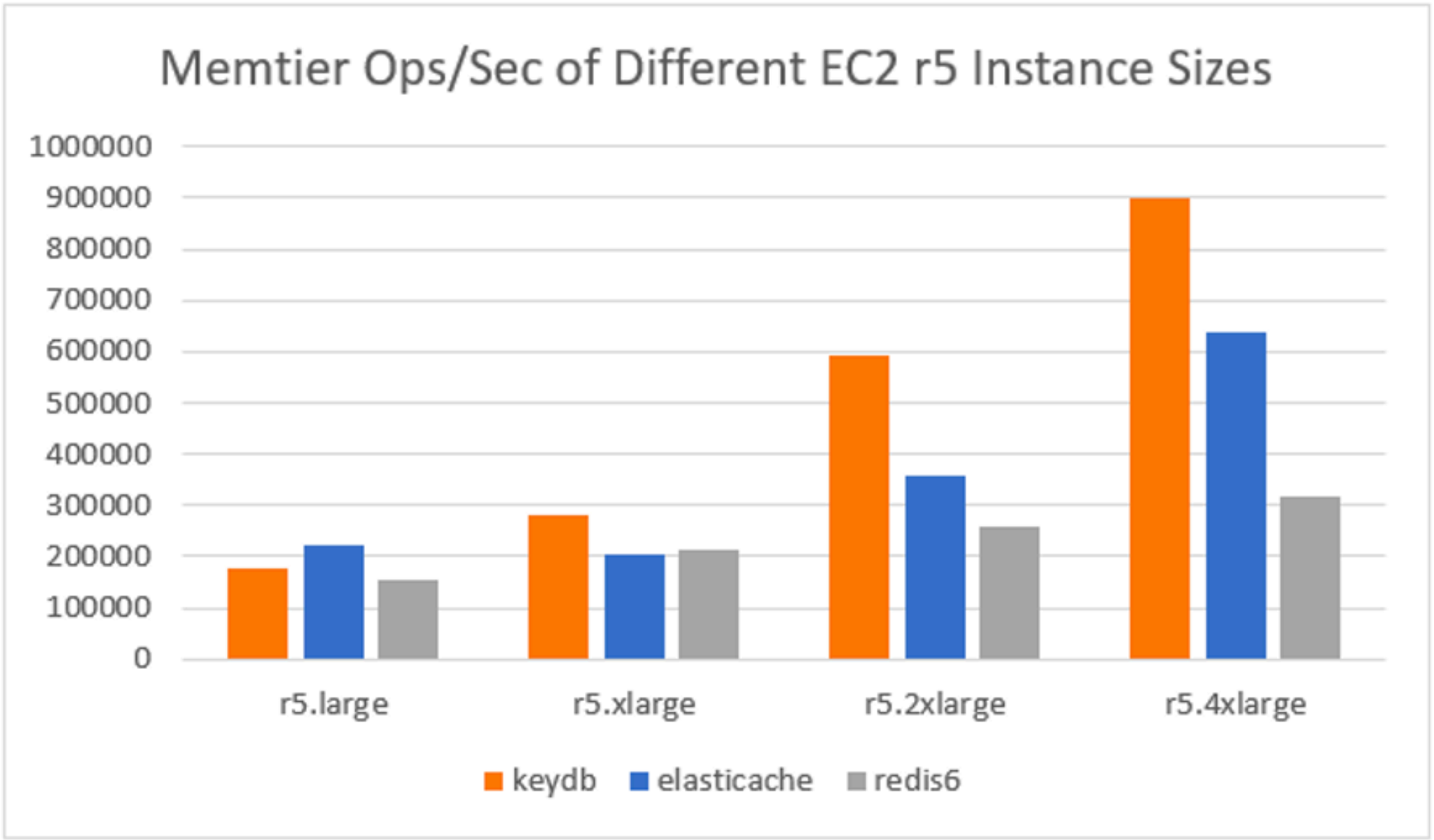
Redis

Single thread

Bottleneck on memory and network

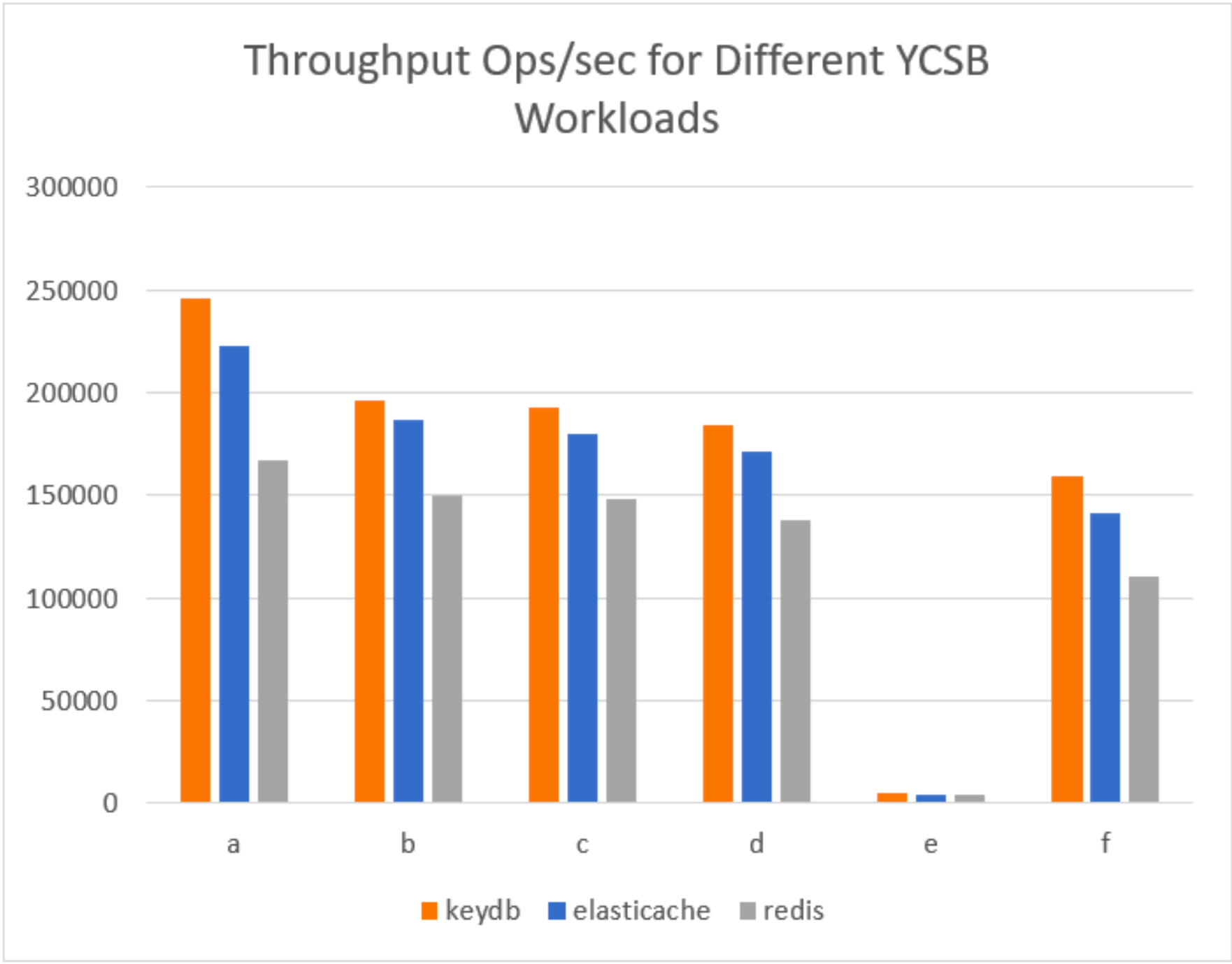
Redis 4.x+ have more threads for I/O

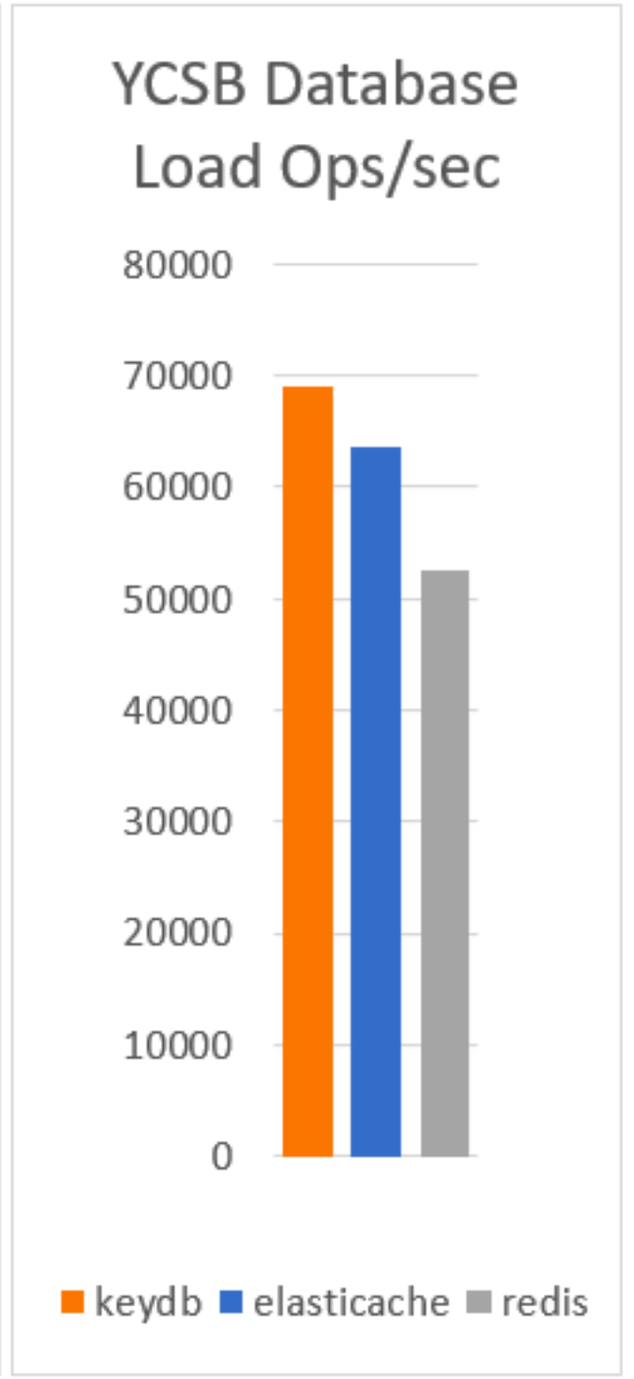
Compare with KeyDB



https://docs.keydb.dev/blog/2020/04/15/blog-post/

Compare with KeyDB

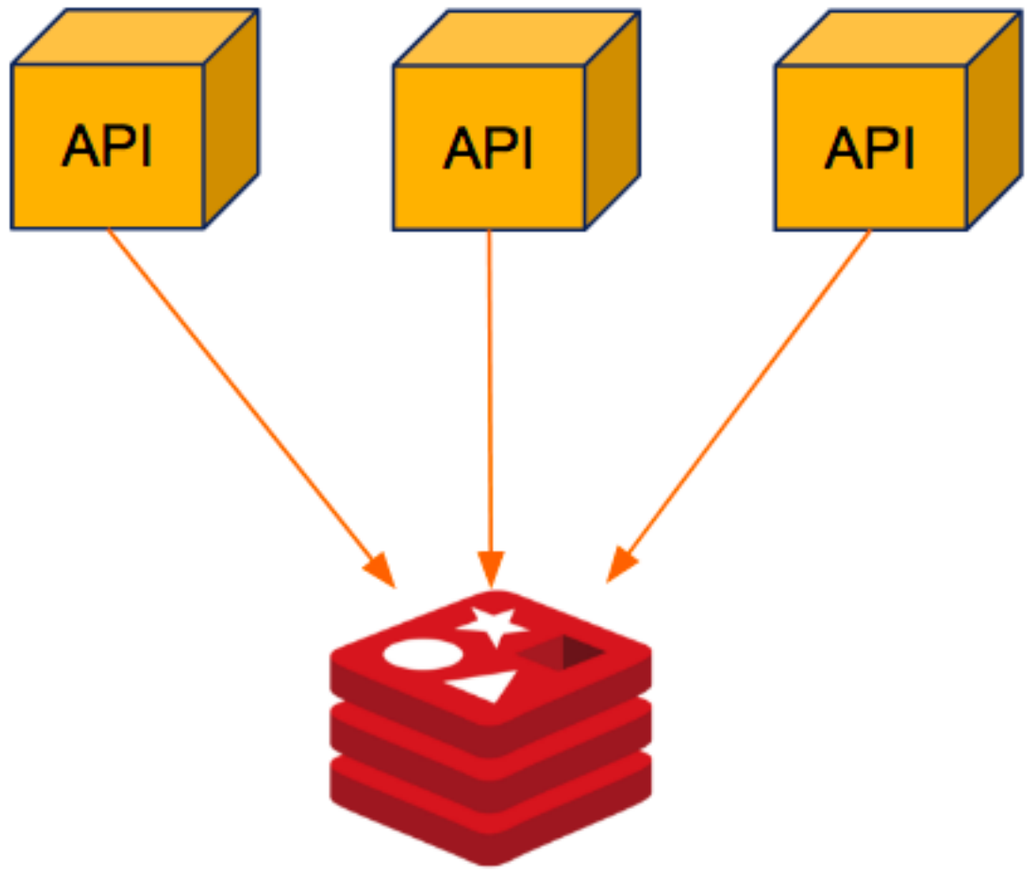




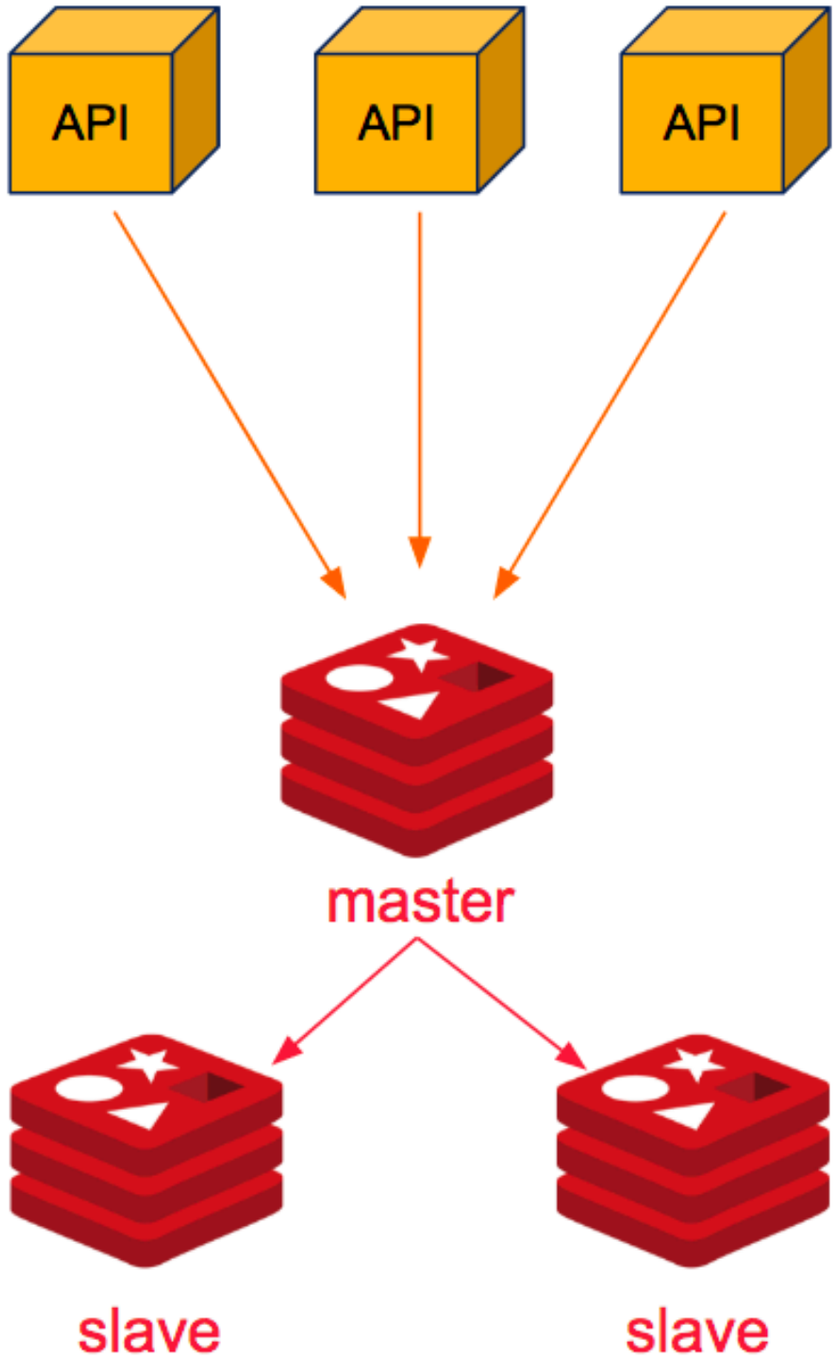
https://docs.keydb.dev/blog/2020/04/15/blog-post/

1. Single instance

Simple in running and configuration
Limited by host's resources (cpu and memory)
Running on port = **6379**



2. Master-slave replication



2. Master-slave replication

Master node = for write data

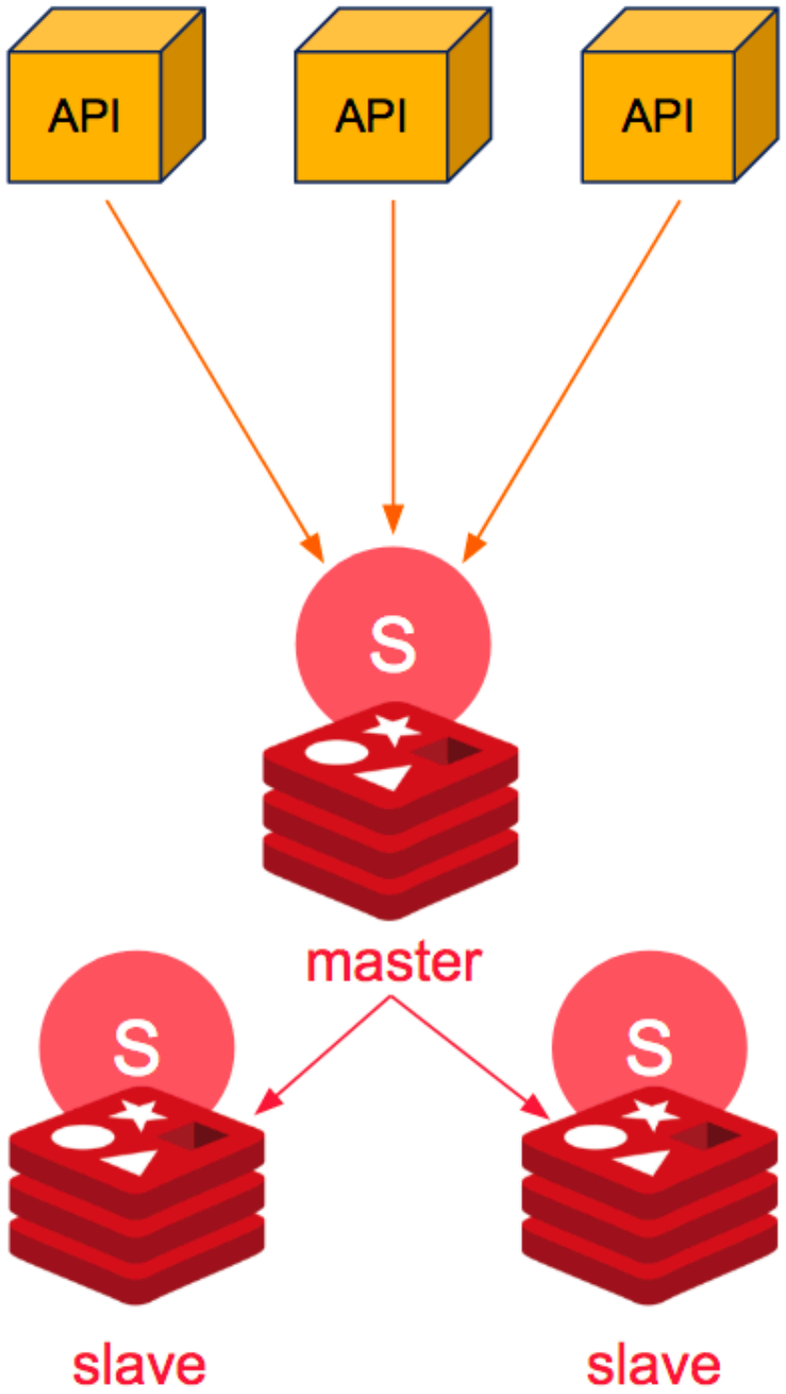
Slave nodes (read only) = for read data

2. Master-slave replication

1. Data will be updated on master node
2. Master node will push changes to slave nodes

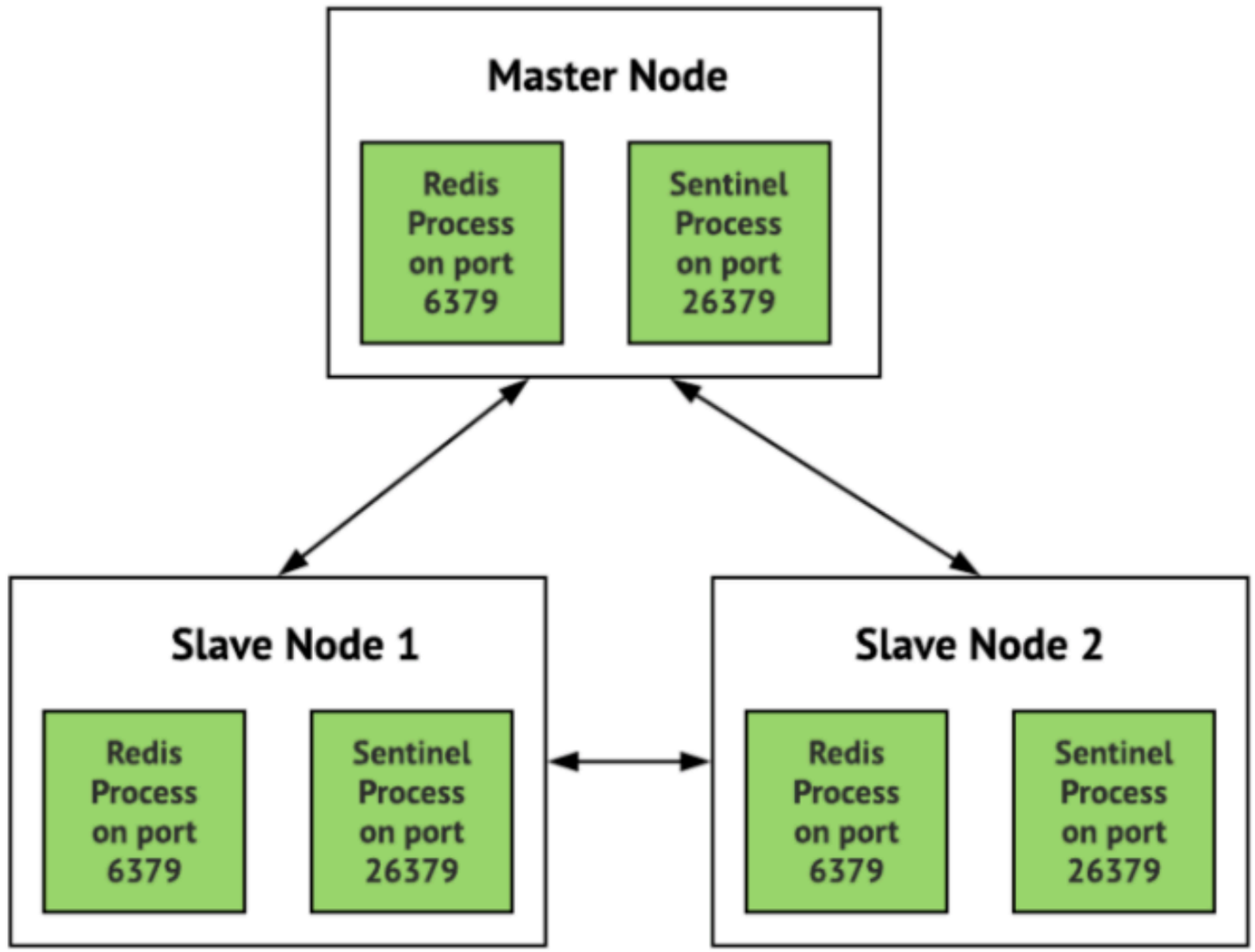
Simple configuration but write operations are limit by master’s resources

3. Redis sentinel



3. Redis sentinel

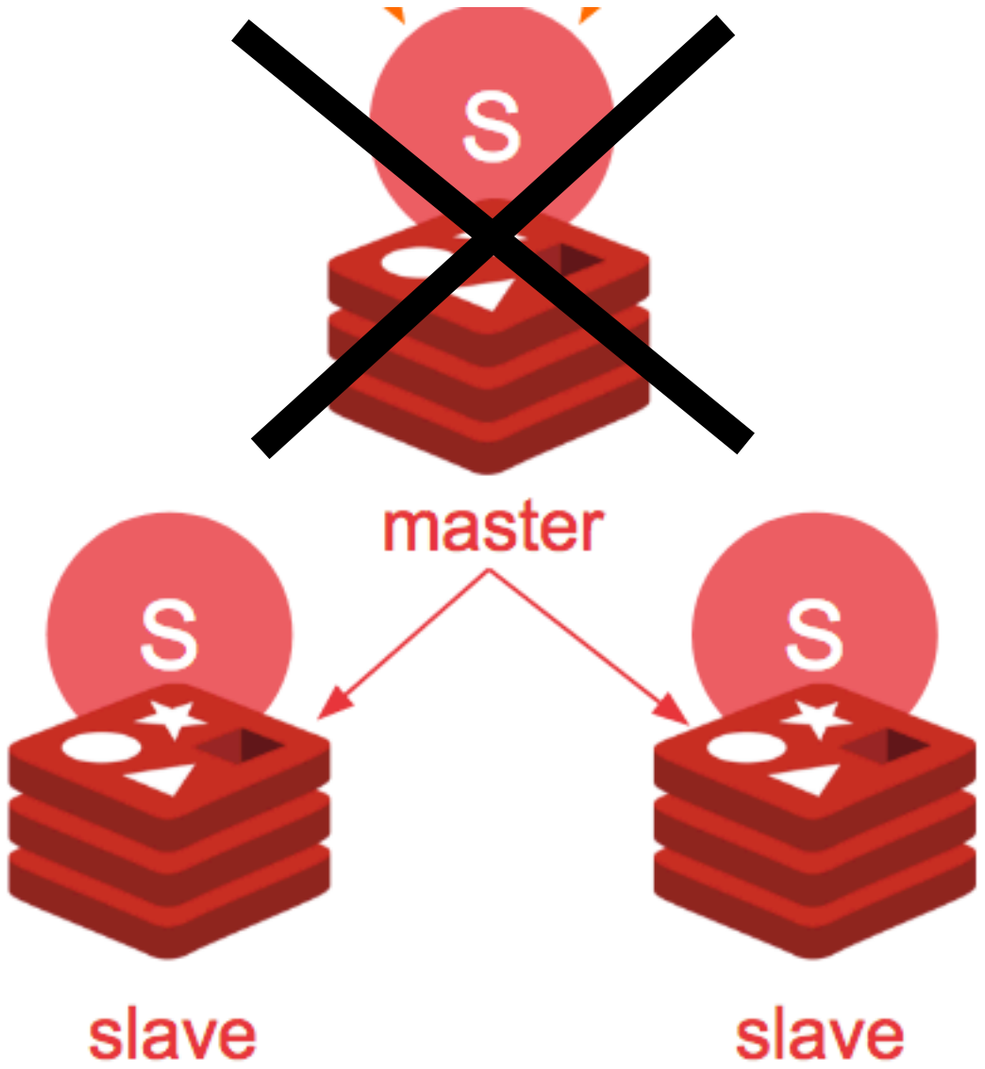
Running sentinel process on port = **26379**



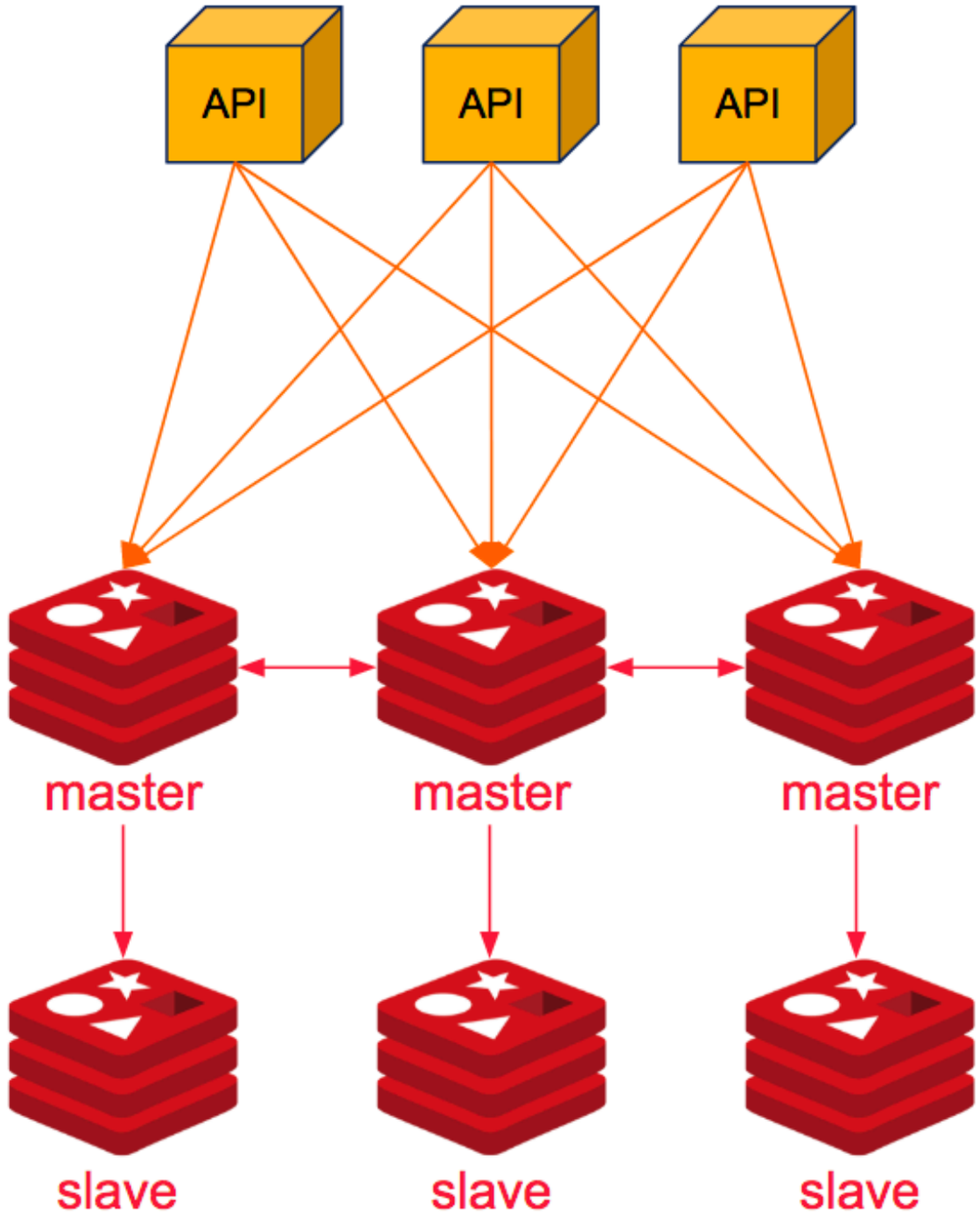
3. Redis sentinel

Similar with Redis replication
Improve from replication (High availability)

Master Down !!



4. Redis cluster



4. Redis cluster

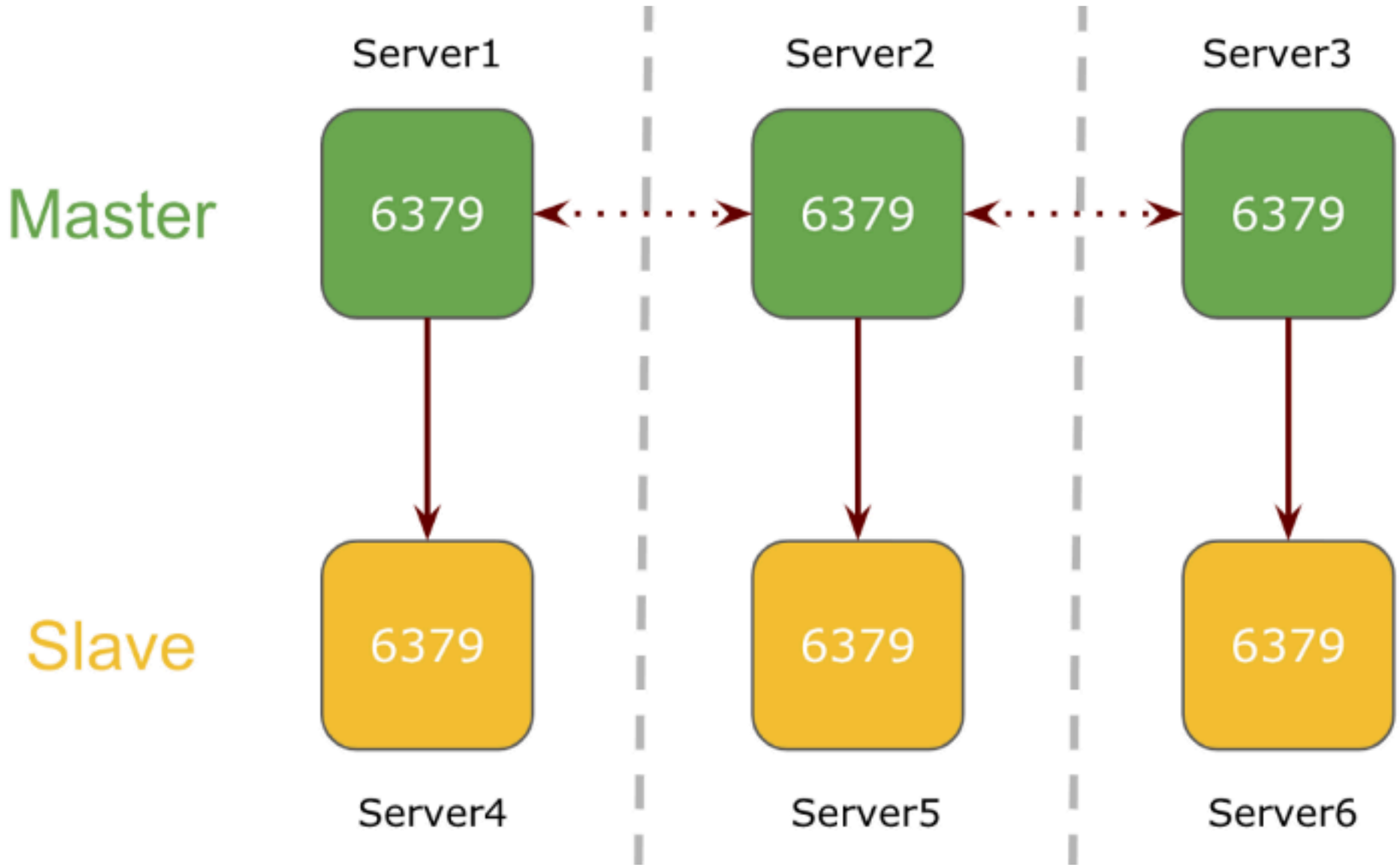
Replication + Cluster + Sharding

4. Redis cluster

Most powerful solution
Minimum 6 redis nodes
3 master nodes
3 slave nodes

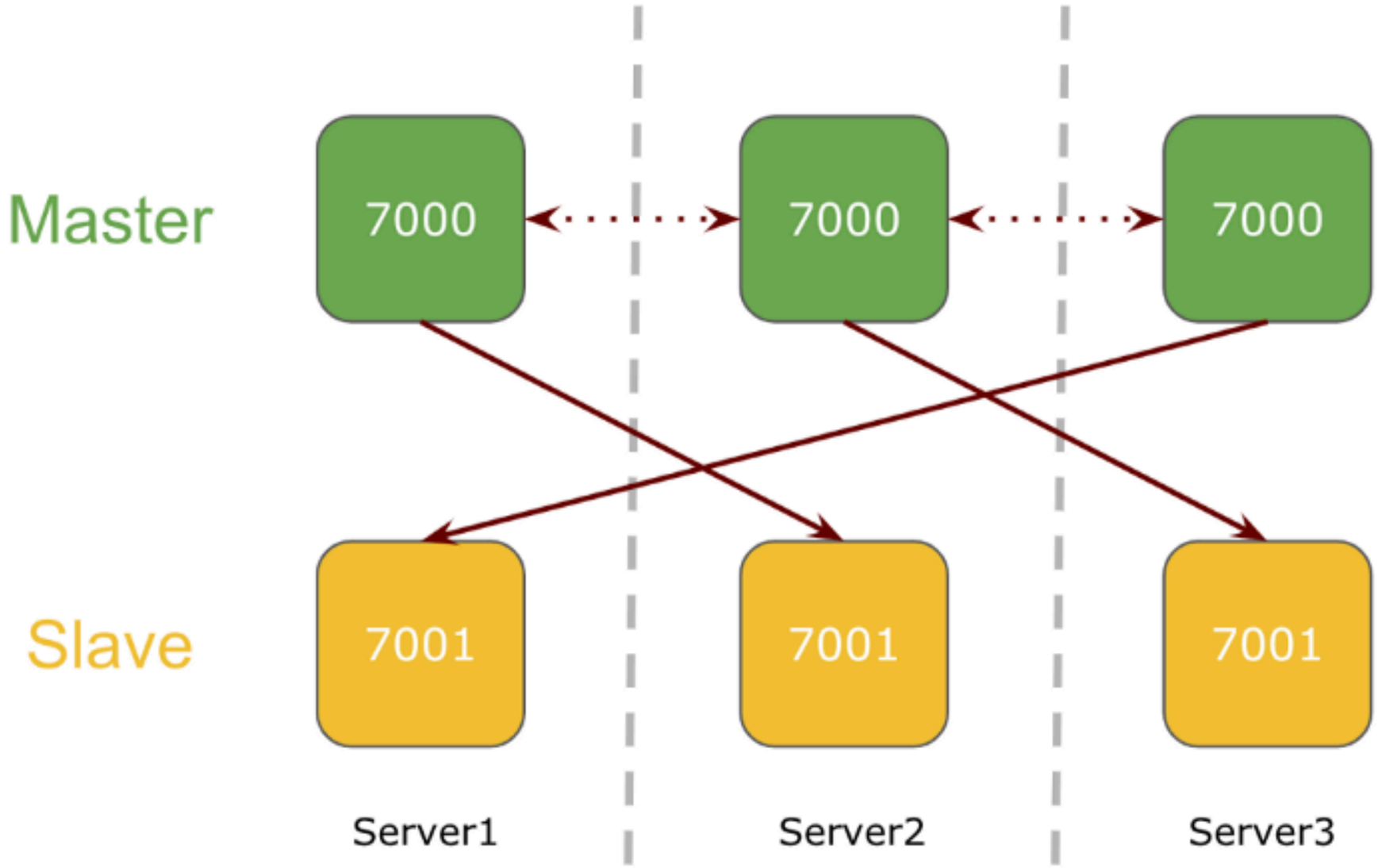
Port = 6379

1. Client connections and communications



Port = 7000,7001

2. Node-to-node communications



Advantages

High performance
High availability
Horizontal and vertical scalability
Native solution (no external proxy)

Limitations

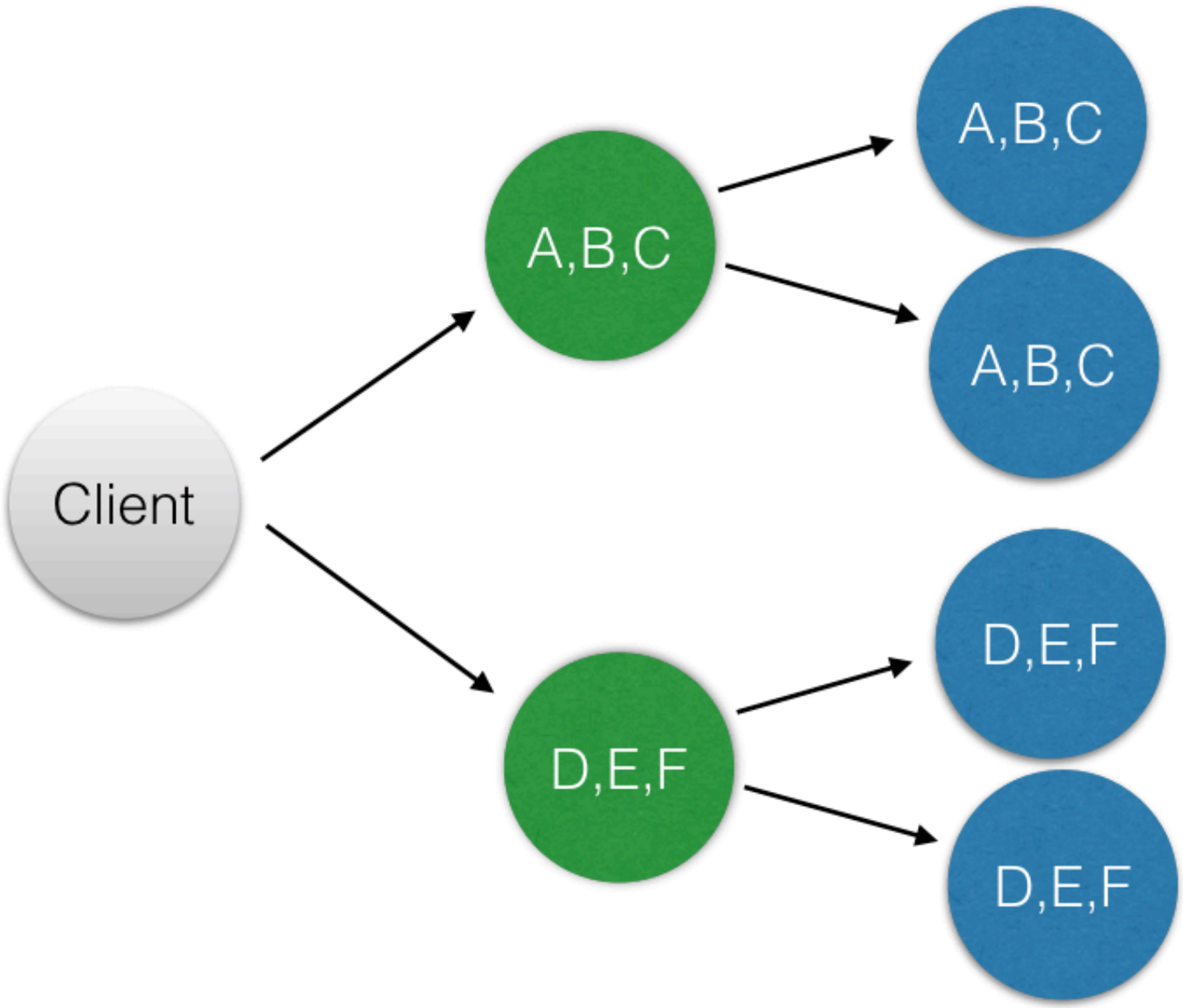
Required client support

Limited **multi-key operation** support

Only supports one database (database = 0)

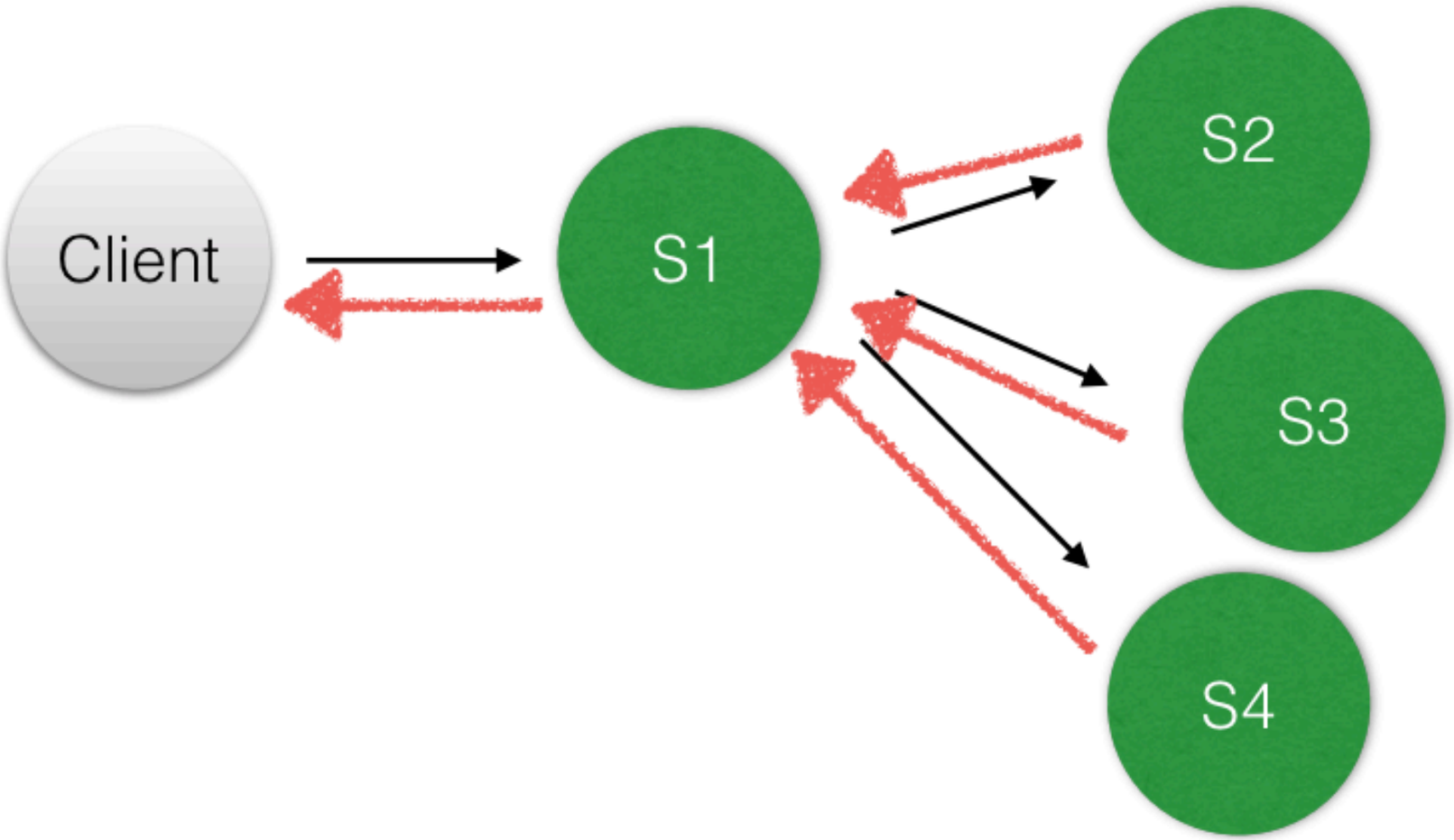
Redis cluster

Sharding and asynchronous **replication**

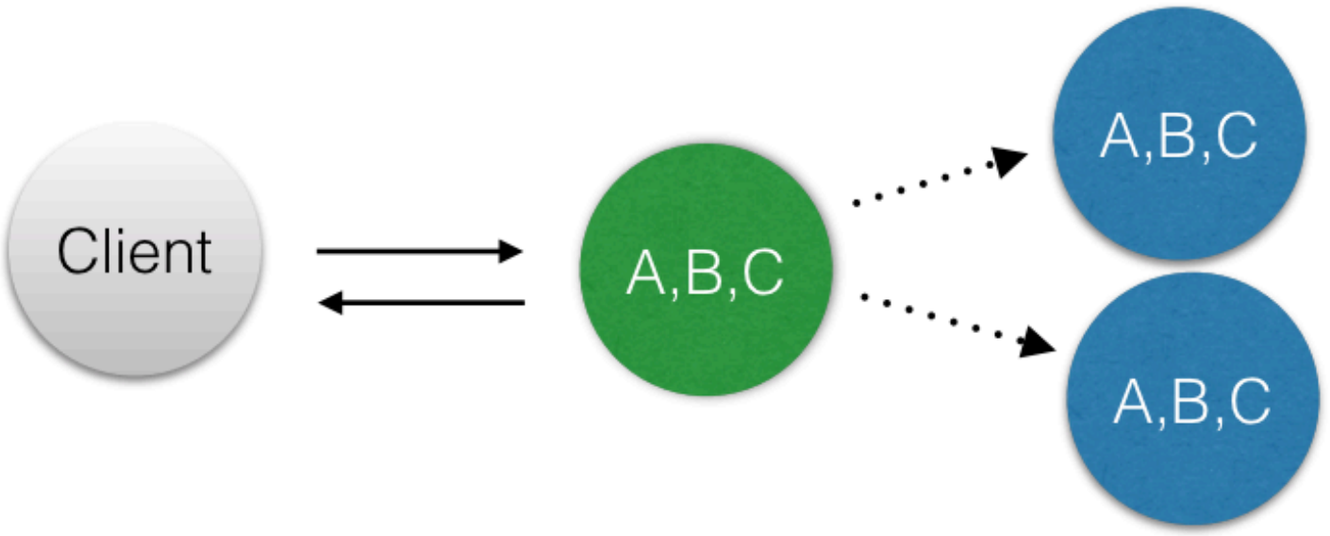


CP system

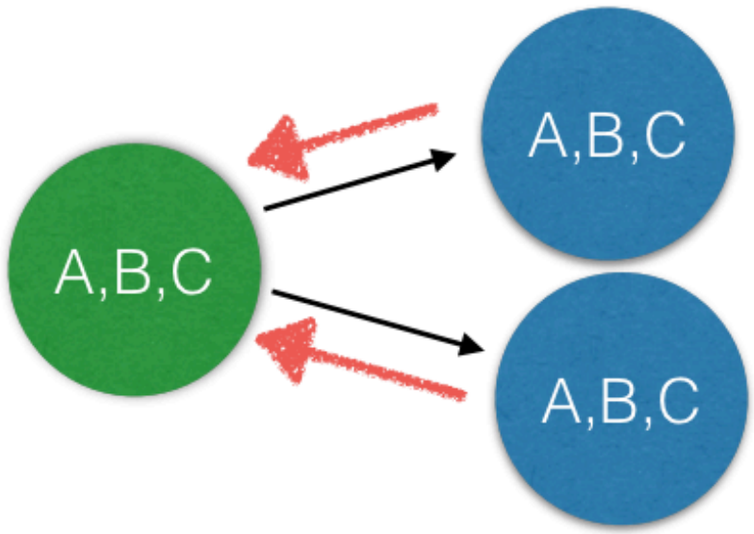
Consistency is added latency !!



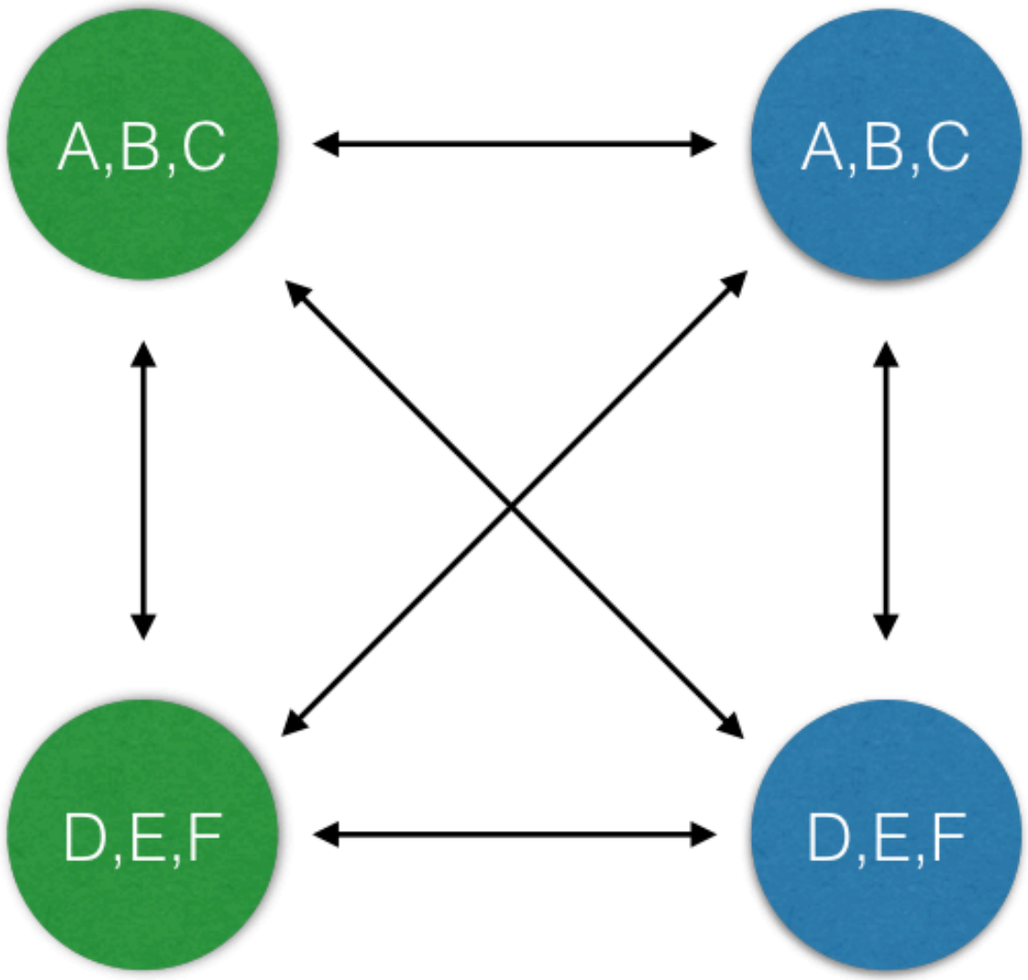
Asynchronous replication



async ACK

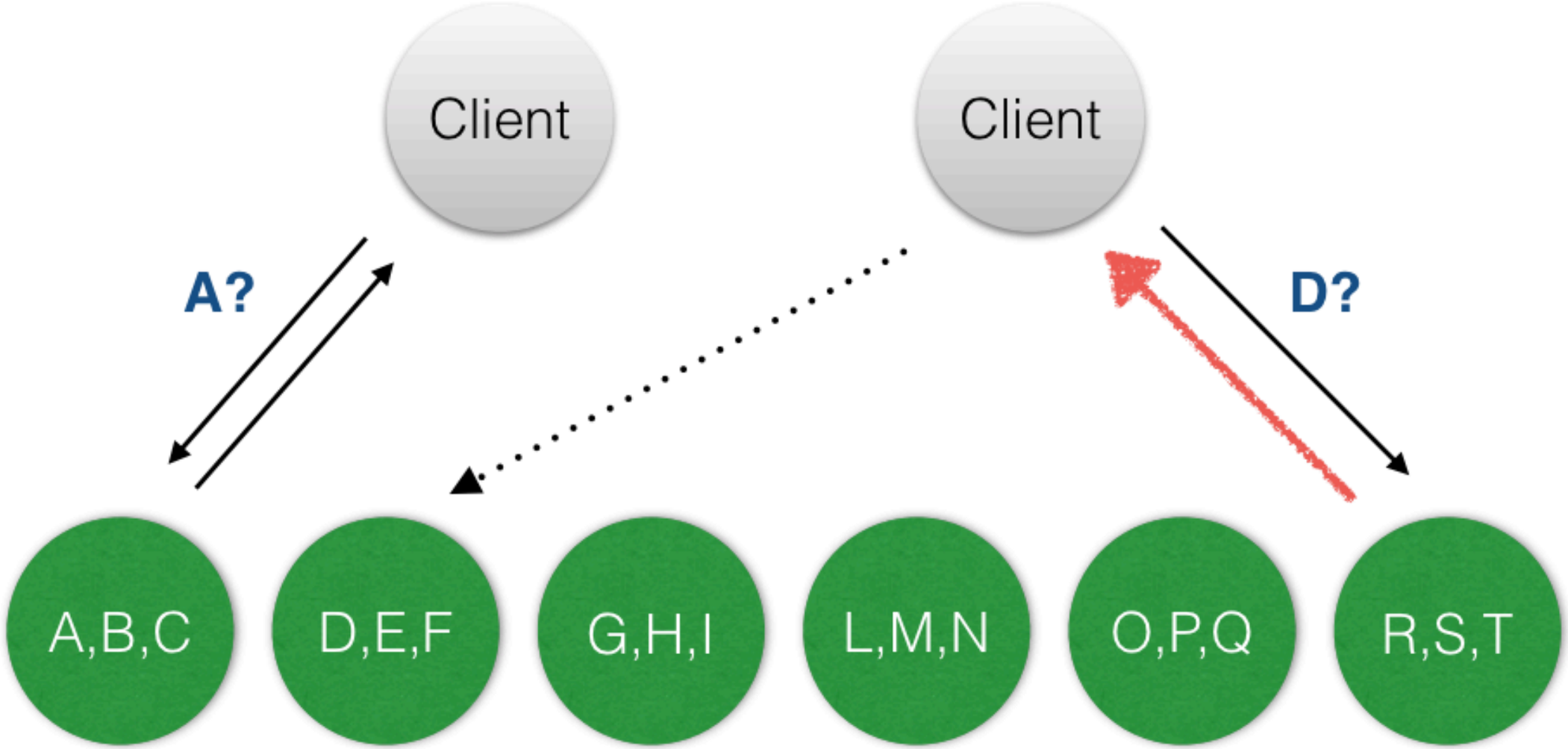


Full mesh



- Heartbeats.
- Nodes gossip.
- Failover auth.
- Config update.

No proxy, but redirection



Sharding

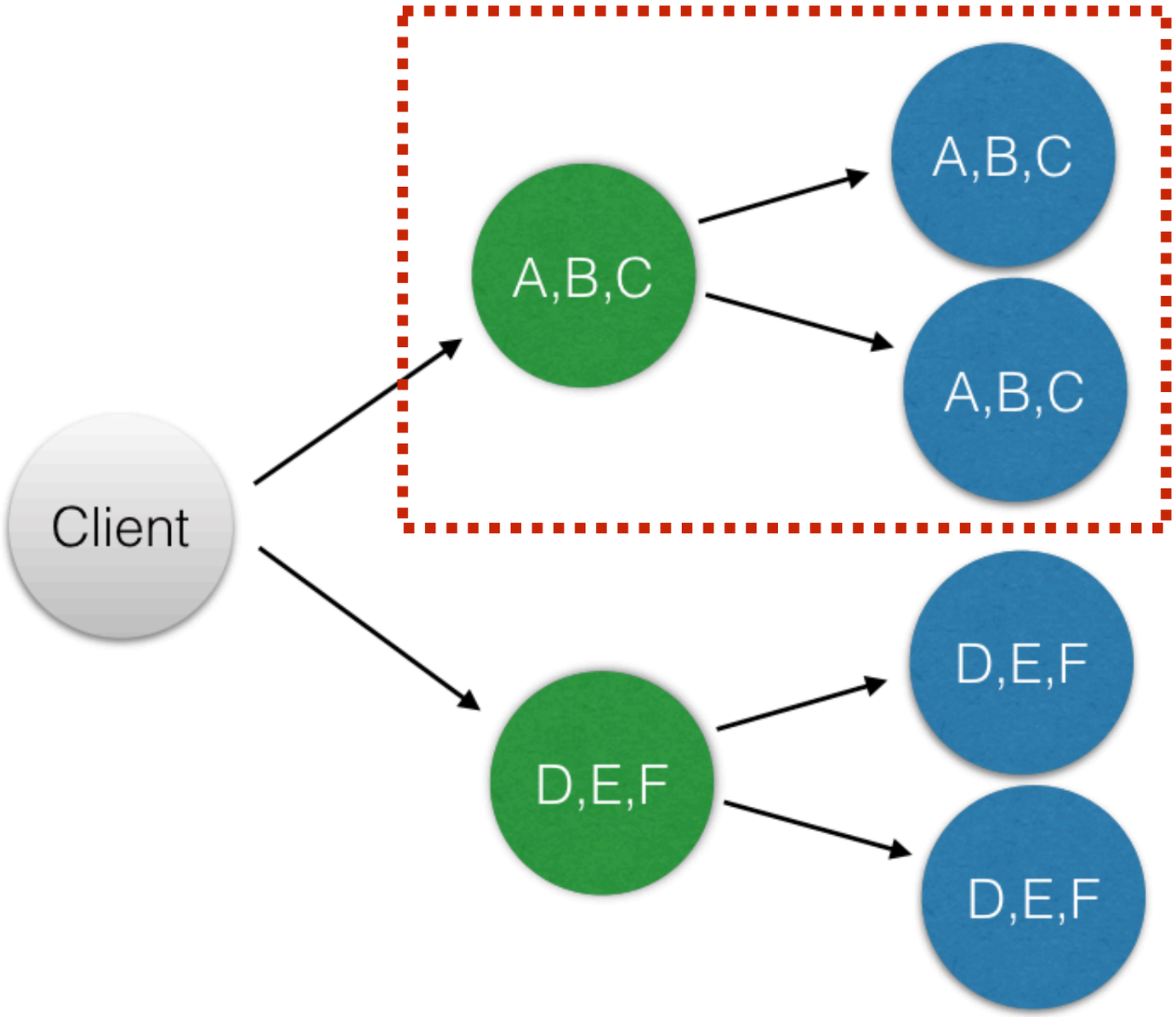
Data is automatically sharded across multiple Redis nodes.

Sharding

The cluster uses hash partitioning to split
the keyspace into **16,384 key slots**,
with each master responsible for a subset of those slots

Sharding

Each slave replicates a specific master



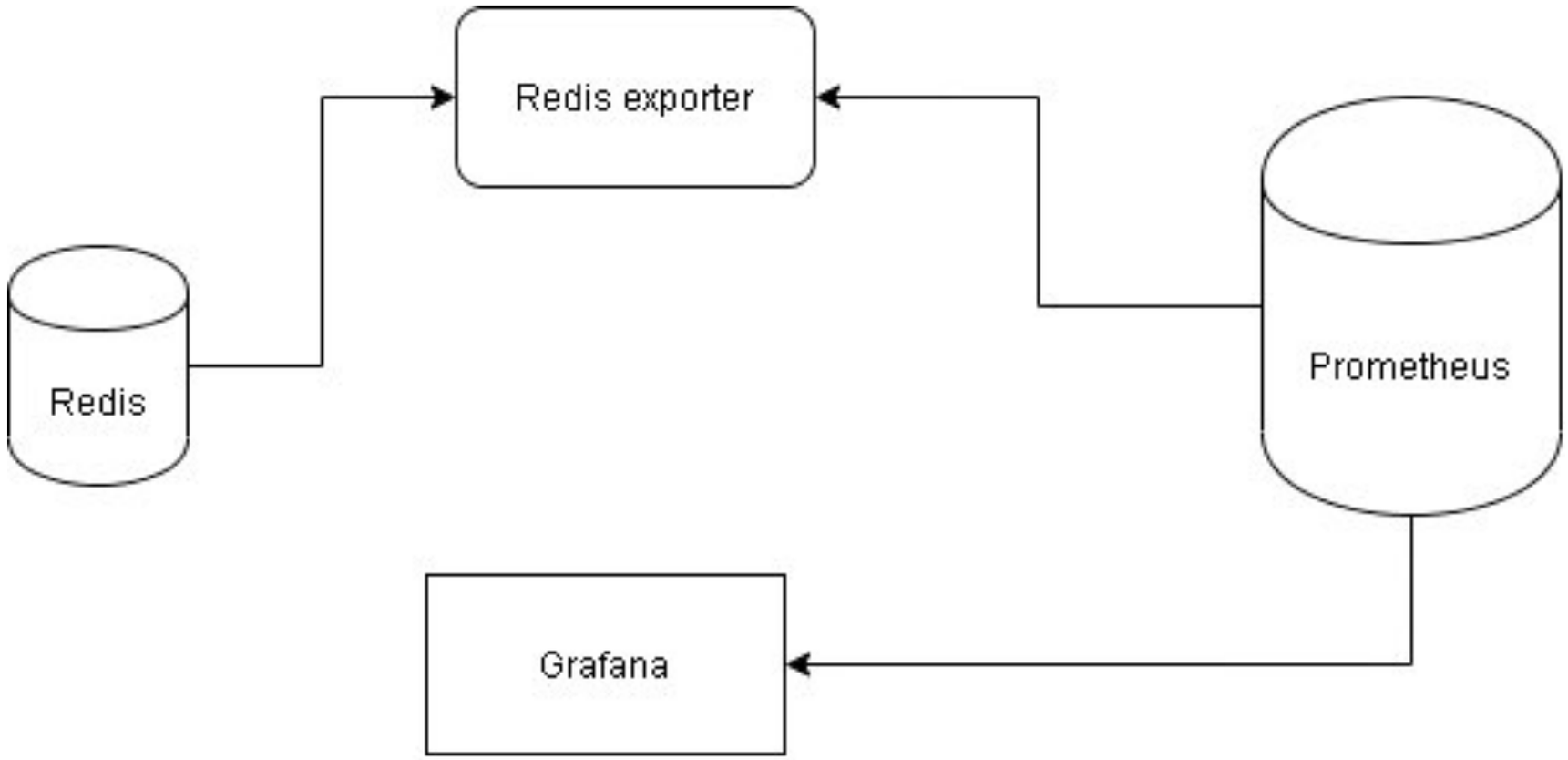
Example Sharding

Hash slot = 16,384
All data are divides into slots

Server	Hash slot (Bucket)
1	0-5,500
2	5501-11,000
3	11,001-16383

Configuration
Performance tuning

Redis exporter



Redis monitoring using Prometheus and Grafana

https://github.com/oliver006/redis_exporter

Workshop