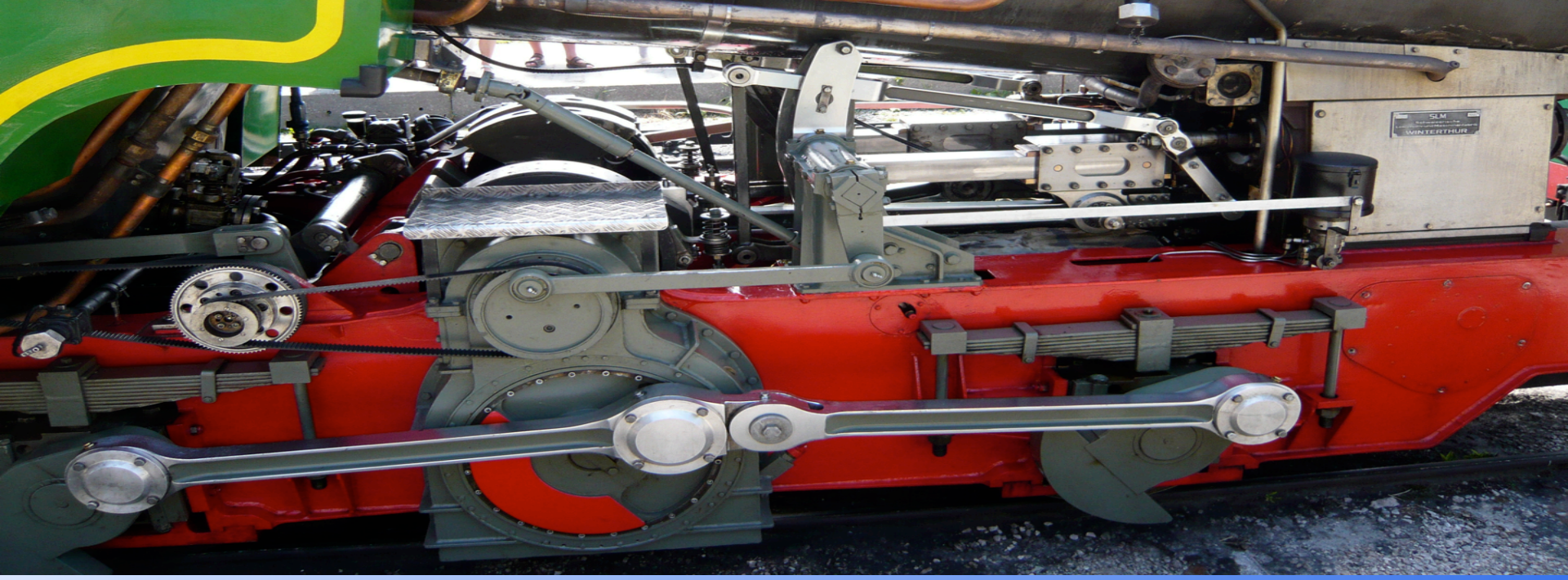


RestFS

Fabrizio Manfredi Furuholmen
Federico Mosca

Beolink.org

Agenda

RestFS

- Introduction
  - Goals
  - Principals
- RestFS
  - Architecture
  - Internals
  - Sub project
- Conclusion
  - Developments

Zetabyte

1000^7 bytes

10^{21} bytes

1,000,000,000,000,000,000,000 bytes

All of the data on Earth today 150GB of data per person

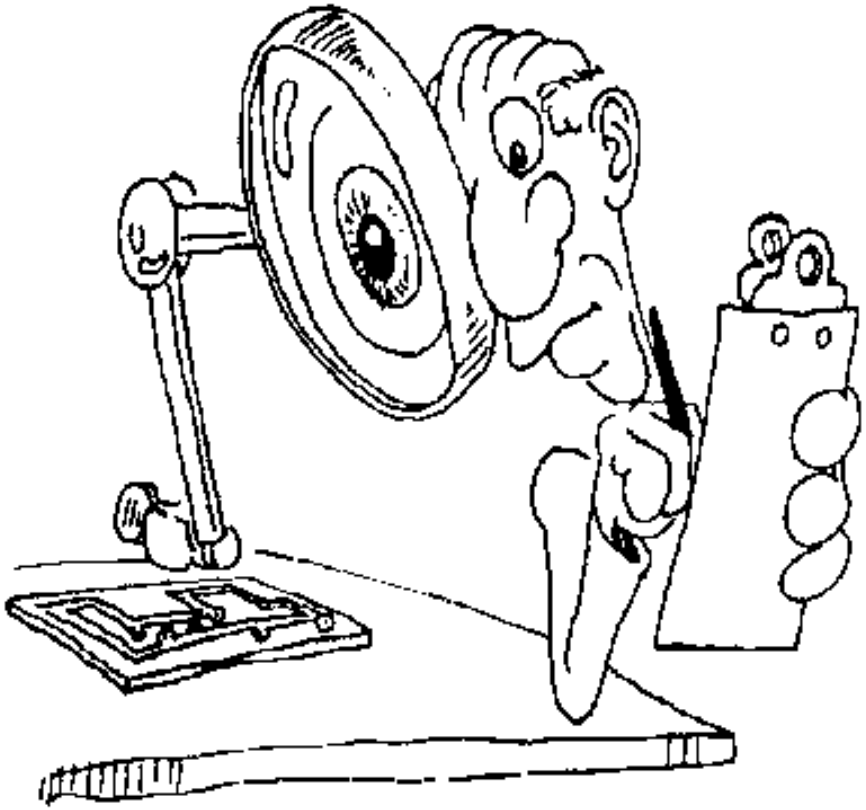
2% of the data on Earth in 2020

Unsolved problem


RestFS

RestFS
is
High scalable, high available
network object storage

Create a framework for **testing** a new **technologies** and **paradigm**



“Moving Computation is Cheaper than Moving Data”

“There is always a failure waiting around the corner”

*Werner Vogel

“Decompose into small loosely coupled, stateless building blocks”

*’ Leaving a Legacy System Revisited’ Chad Fowler

Five pylons

Objects

- Separation btw data and metadata
- Each element is marked with a revision
- Each element is marked with an hash.

Cache

- Client side
- Callback/ Notify
- Persistent

Transmission

- Parallel operation
- Http like protocol
- Compression
- Transfer by difference

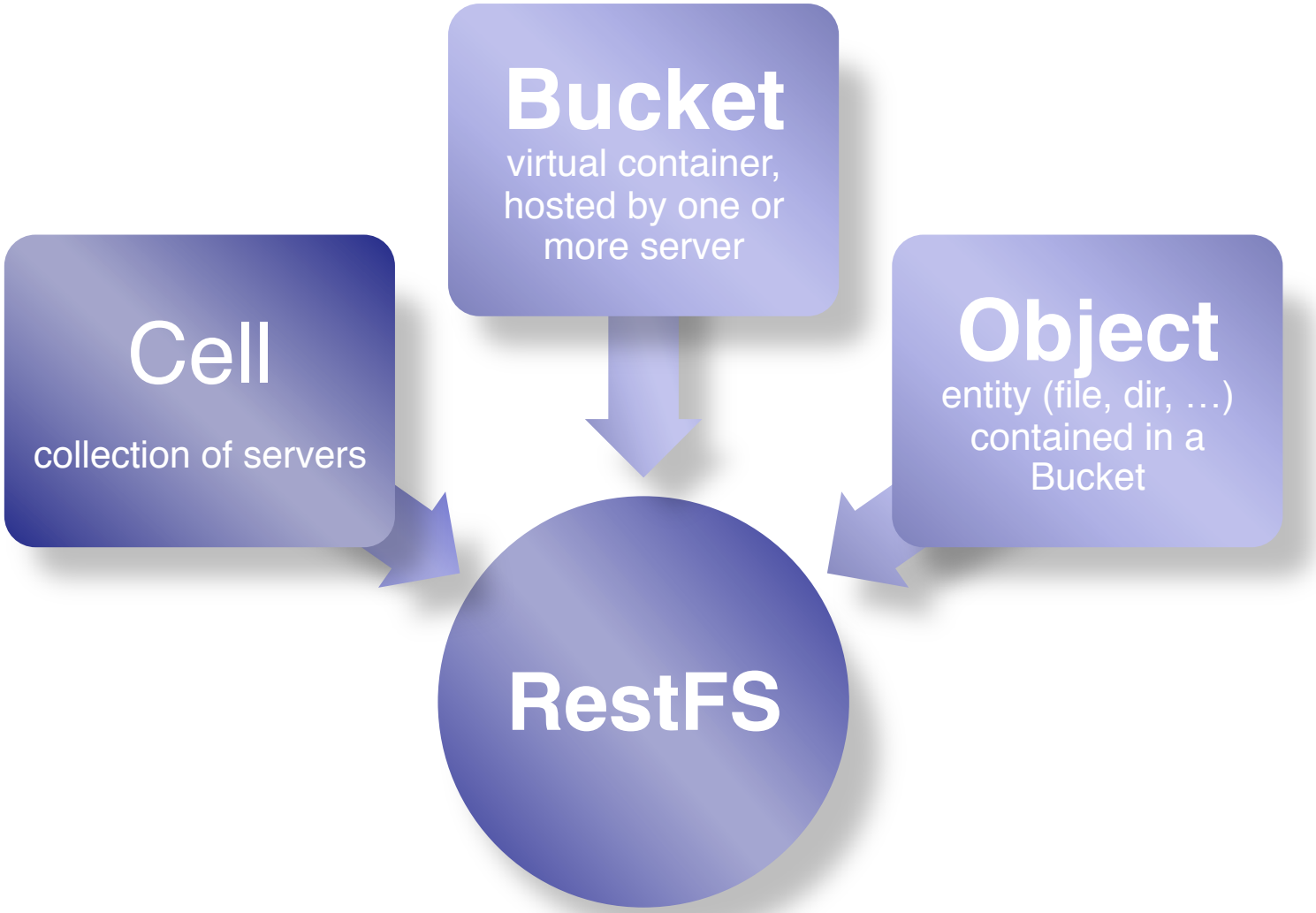
Distribution

- Resource discovery by DNS
- Data spread on multi node cluster
- Decentralize
- Independents cluster
- Data Replication

Security

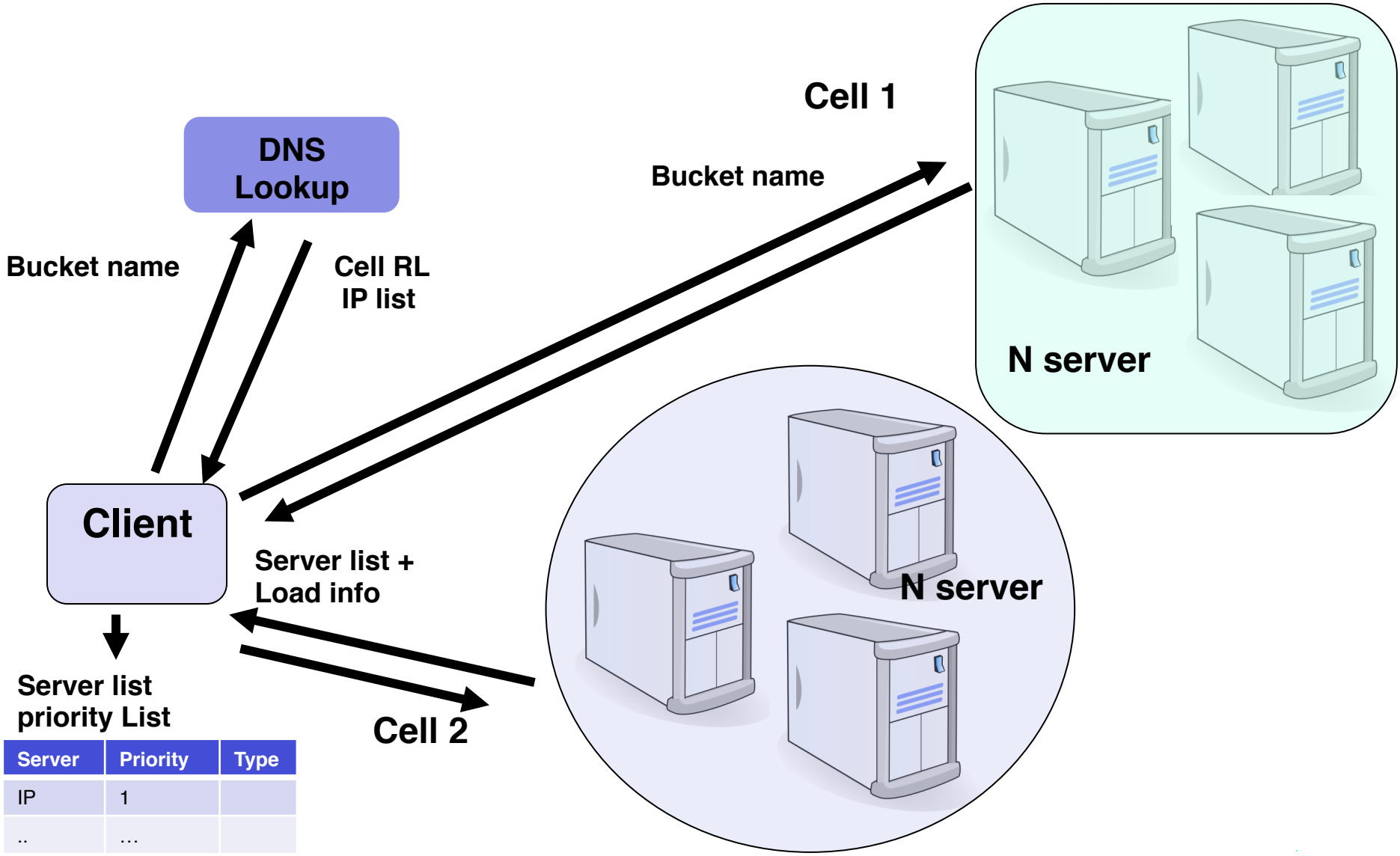
- Secure connection
- Encryption client side,
- Extend ACL
- Delegation/ Federation
- Admin Delegation

RestFS Key Words



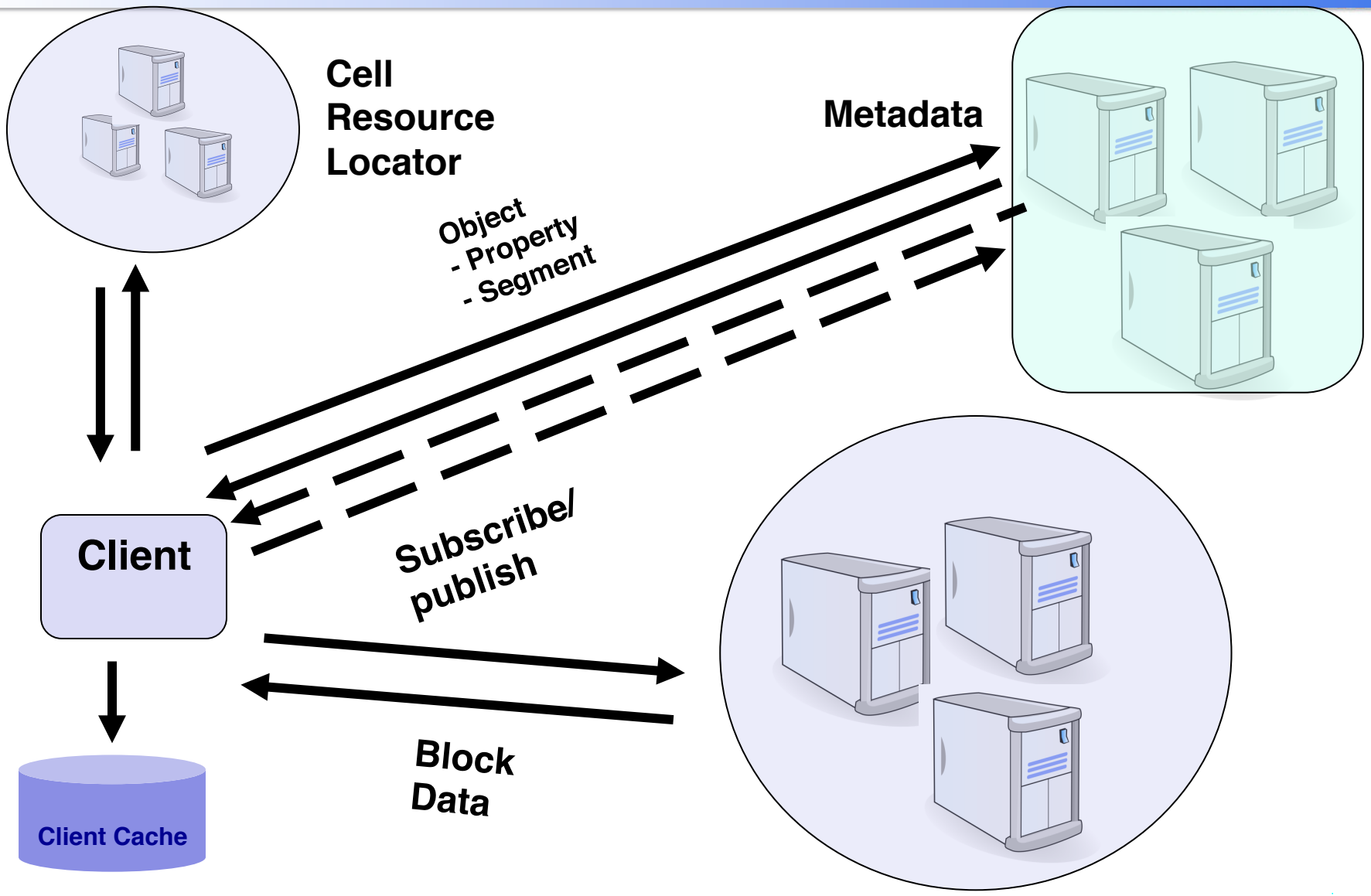
Bucket Discovery

Beolink.org



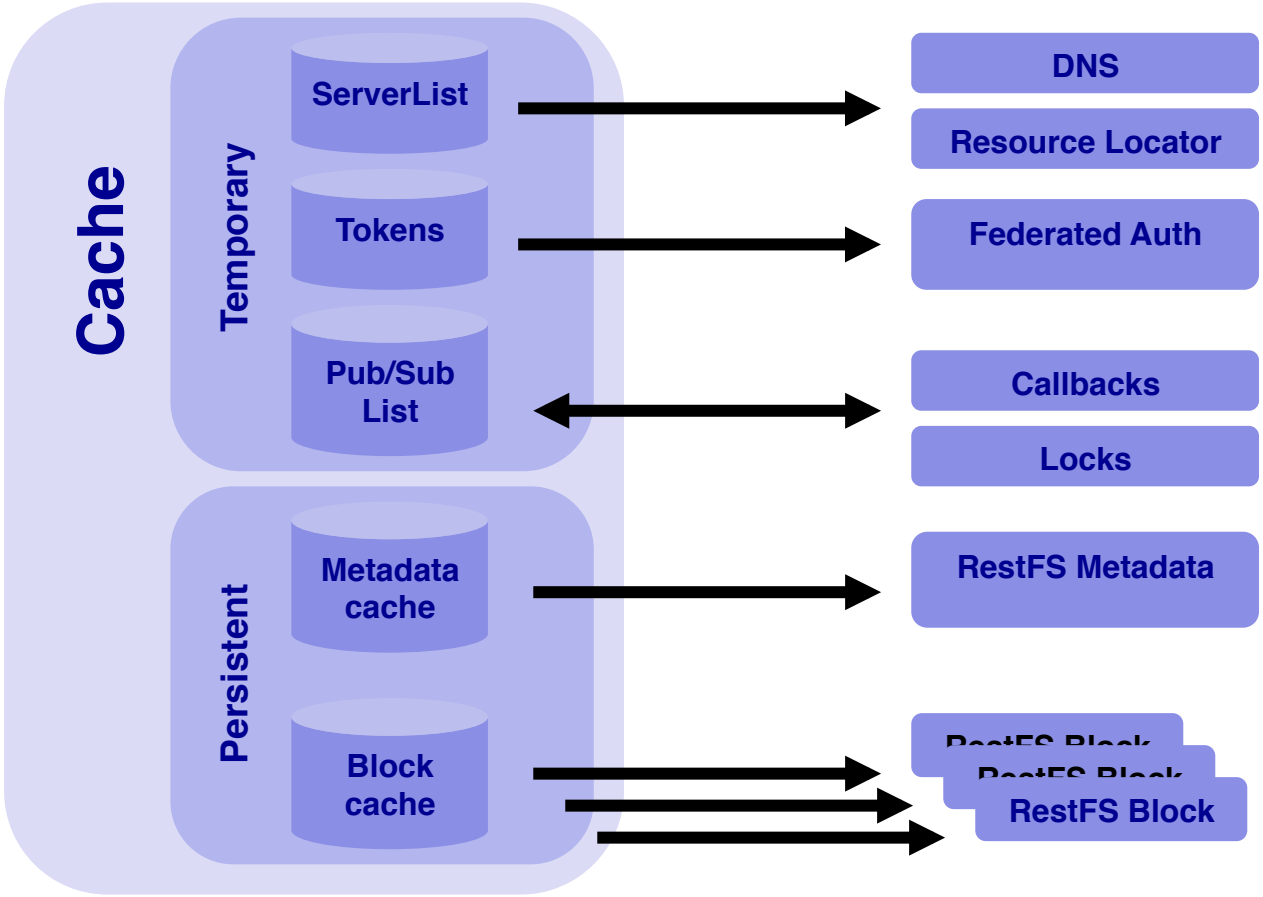
Server	Priority	Type
IP	1	
..	…	

Object Retrieve



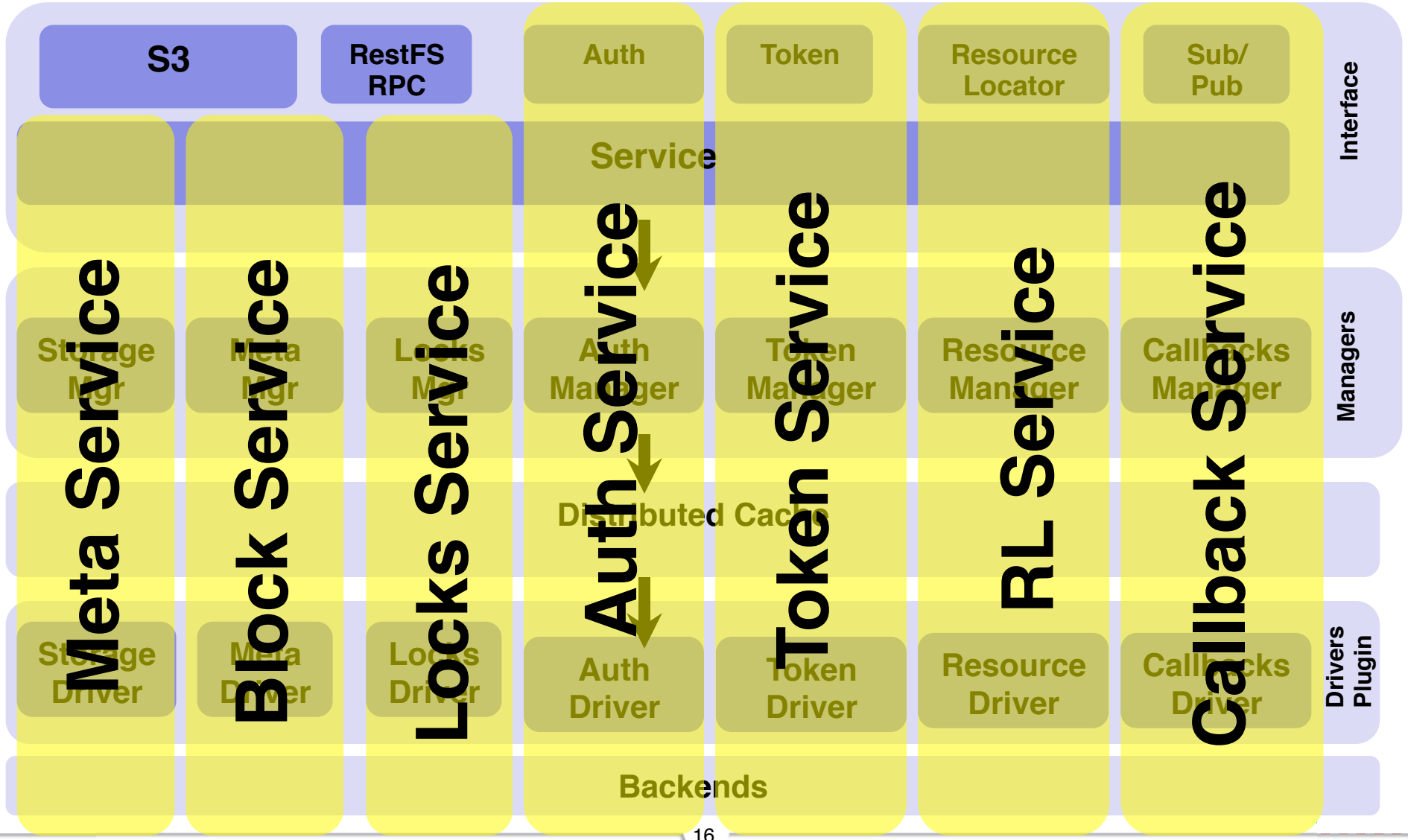
RestFS Cache client side

Beolink.org



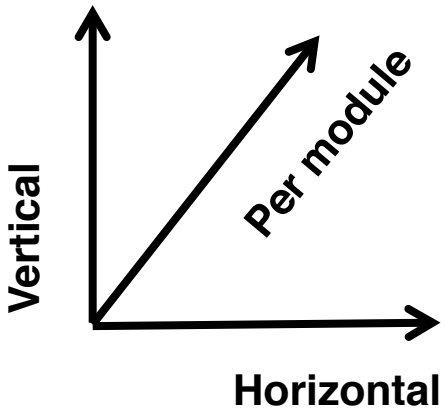
Server Architecture

Beolink.org



Scalability

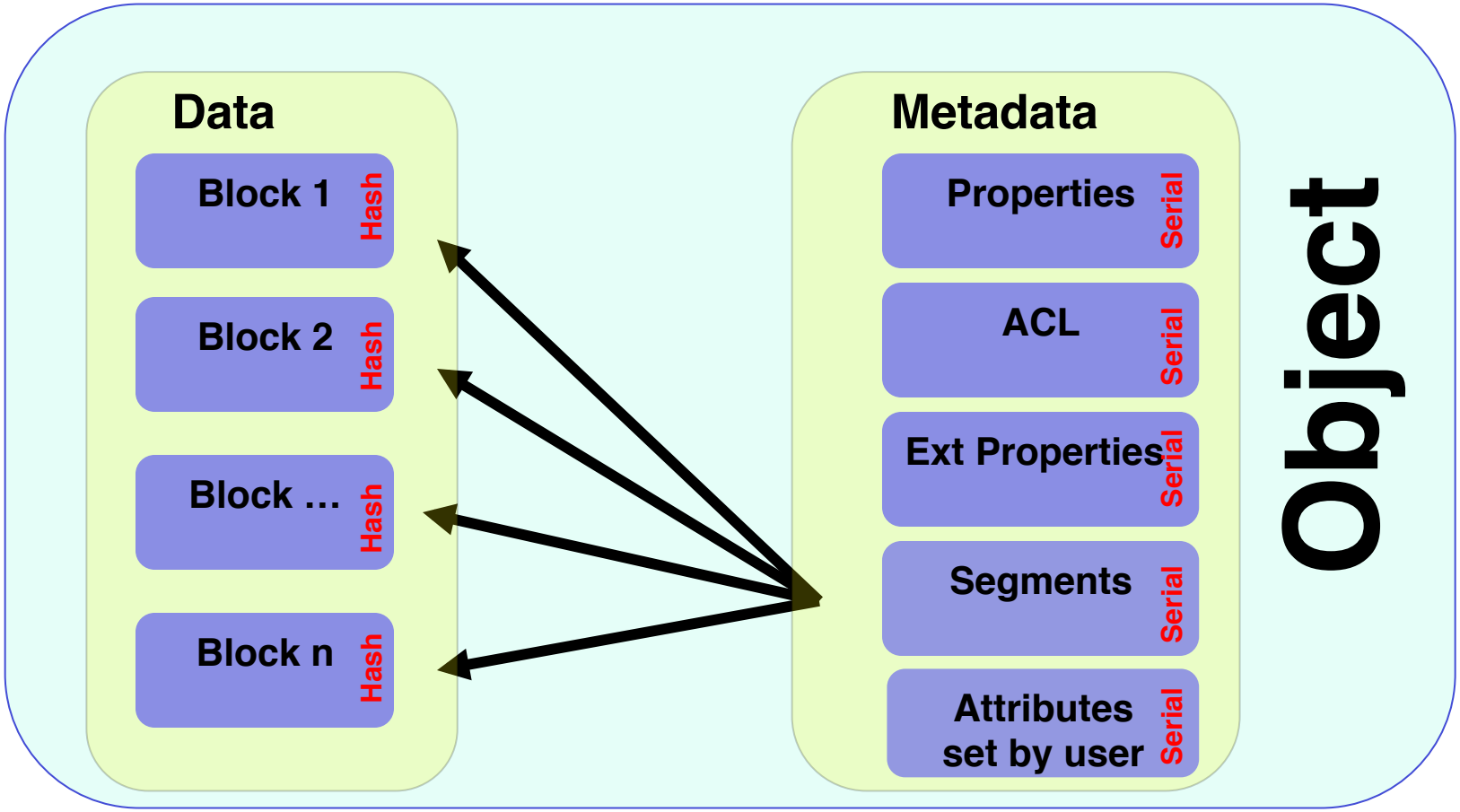
Dedicated storage infrastructure per Function/ Group of functions	→	Decentralization Parallel operations
Key Value pair, as much as possible	→	Native SQL operations (Fast Operation)
Simple query/operation vs complex operation in the DB	→	Parallel operations
Distributed Memory cache	→	Reduce number of operation to the backend



Bucket

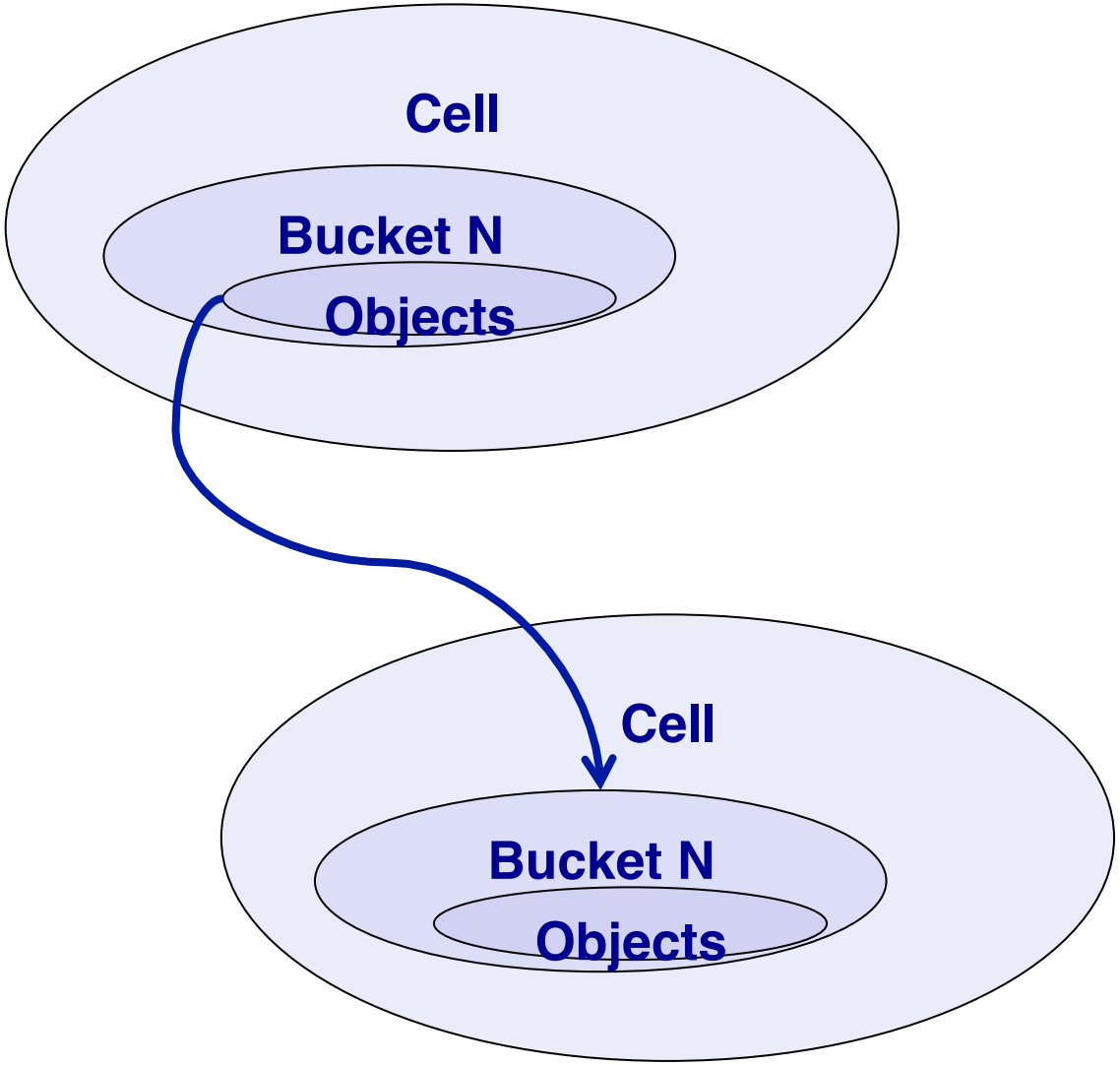
Objects

Object

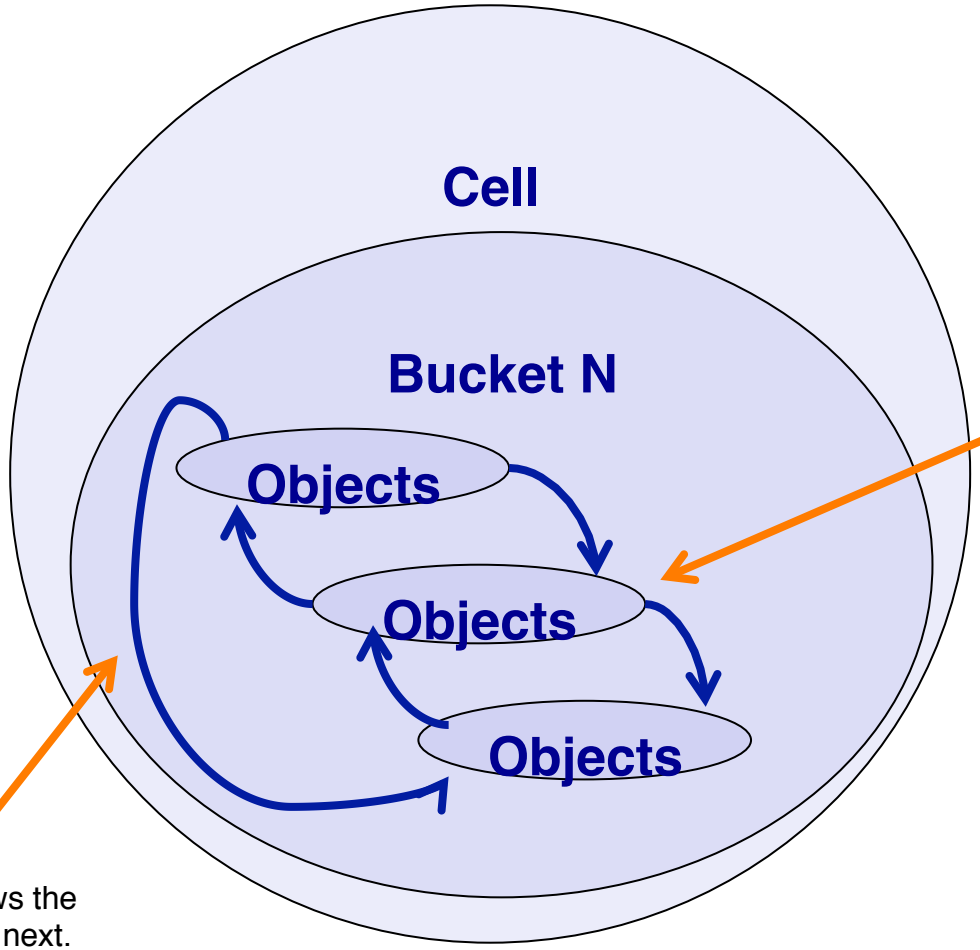

Data
Block 1
Hash
Block 2
Hash
Block …
Hash
Block n
Hash
Metadata
Properties
Serial
ACL
Serial
Ext Properties
Serial
Segments
Serial
Attributes set by user
Serial
Object

Restfs ID

Id Bucket	→	Plain text DNS name
Id Object	→	UUID random generation
Id segment and id block	→	Base on the position of the content
Chunck data on the storage	→	SHA-1 hash of the concatenation of Bucket.object.segment.block_id


Cell
Bucket N
Objects
Cell
Bucket N
Objects

Object Versioning

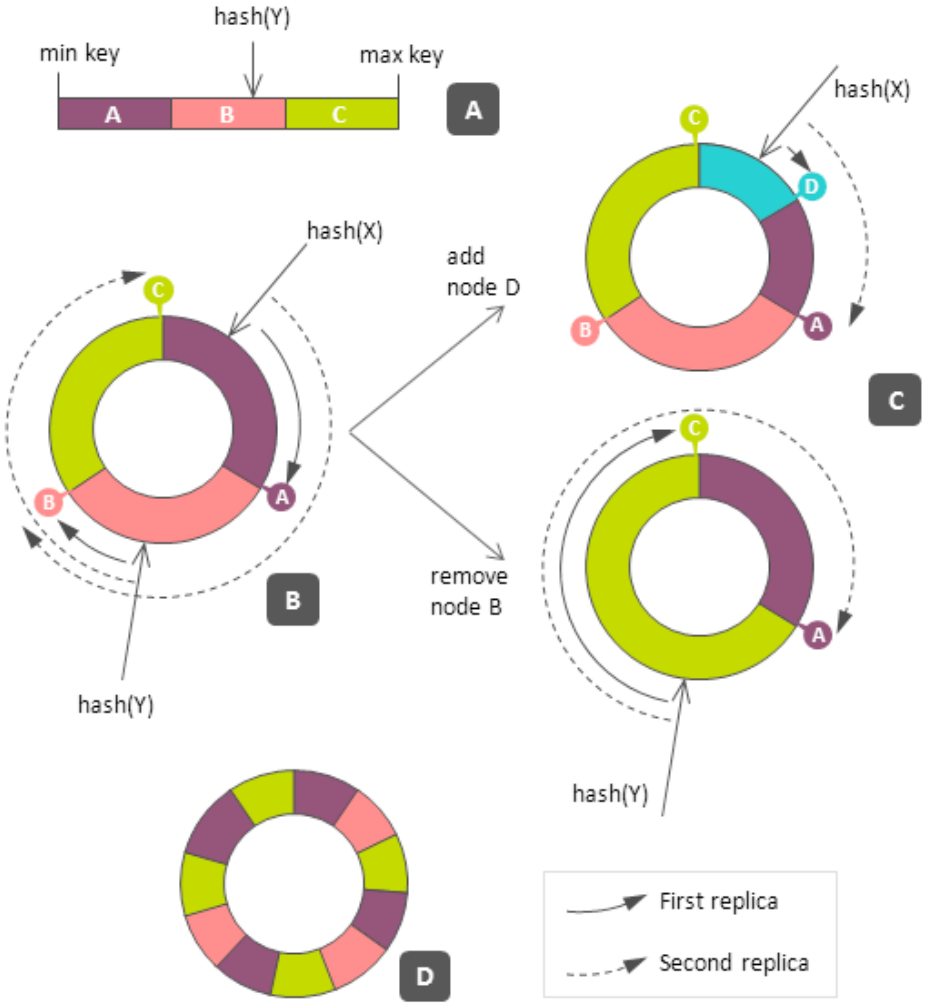


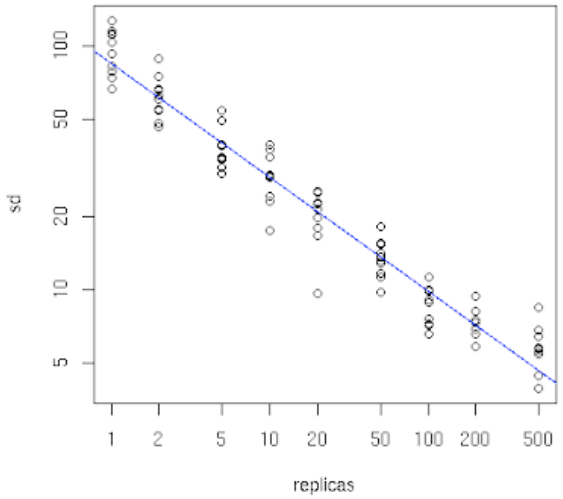
The segment contain the diff to upstream object

Each object knows the previous and the next. The current object knows the previous and the last

Block Storage

Backend: Consistent Hashing





Number of key to move for add/ remove a node :

Keys/Node= keys to relocate

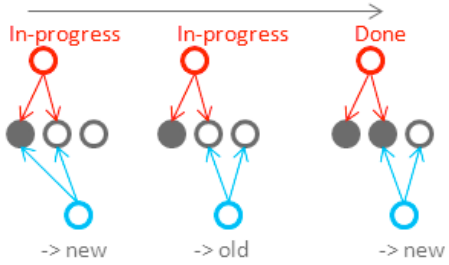
Blocks are collected in shards

http://highlyscalable.wordpress.com/2012/09/18/distributed-algorithms-in-nosql-databases/

3 Copies

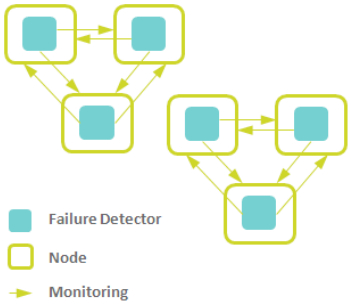
Configurable read and write consistent level and security:

- 2W1R
- 2W2R
- 1W1R
- …



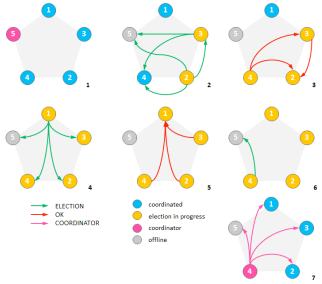
Monitor of neighbored

small cluster of 3 nodes (GOSSIP)

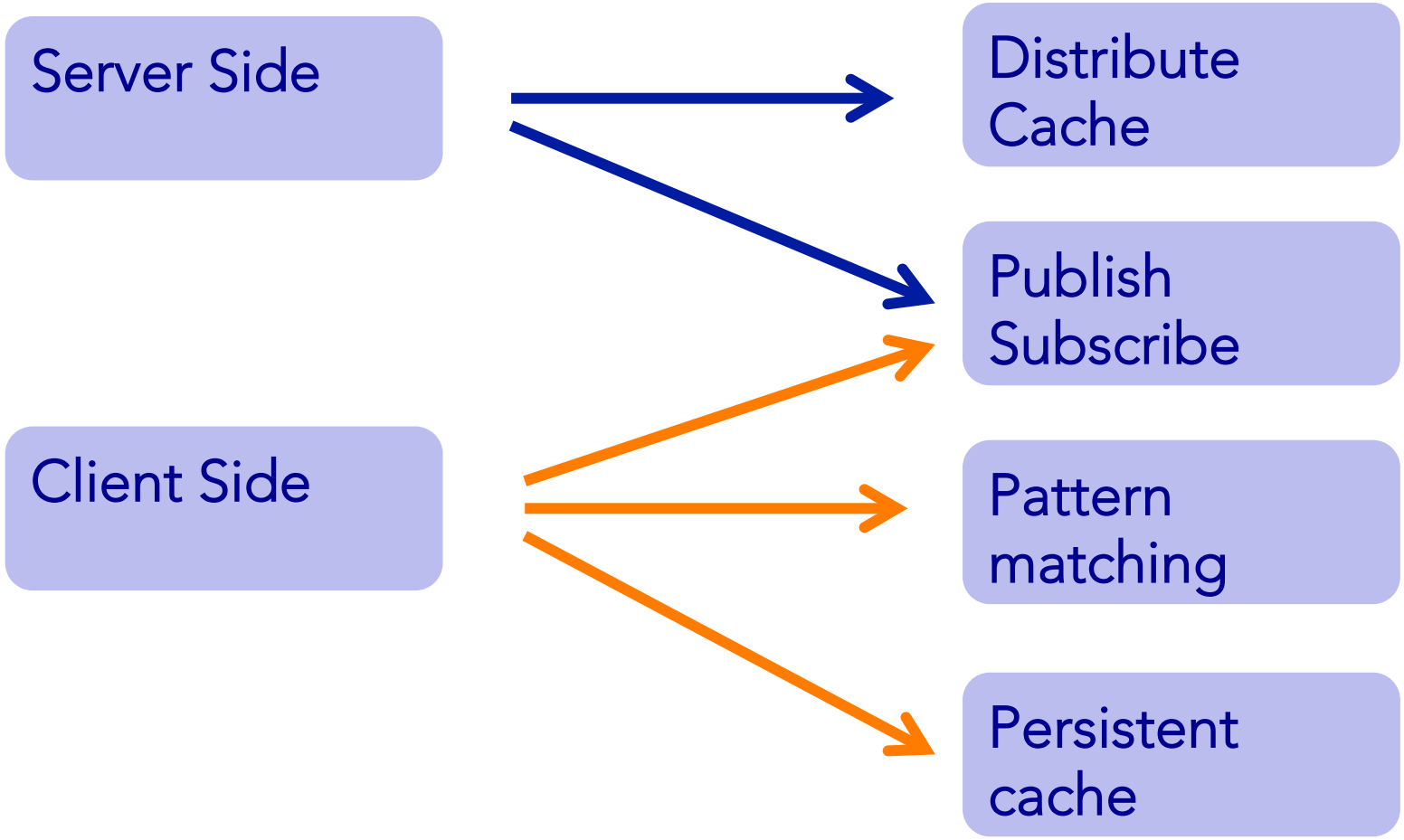


Mini cluster election

key space reclaim for replica coordination, leave join cluster



Cache


Server Side
Client Side
Distribute Cache
Publish Subscribe
Pattern matching
Persistent cache

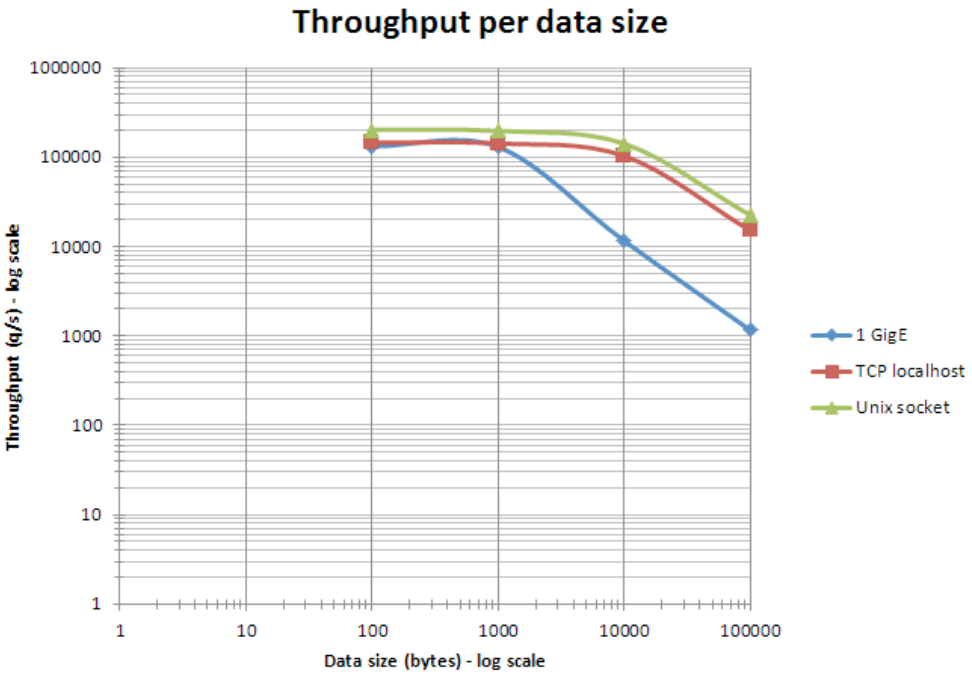
Security

Security


Protocol,
• SSL Protocol
Authentication
• Token for devices (Enrollment)
• Session Token for User
• External password provider
Data Integrity
• Encryption on block level
Authorization
• Extended ACL based on NFS4 ACL
• Admin delegation on the Bucket level

NOSQL **DB**

Redis performance

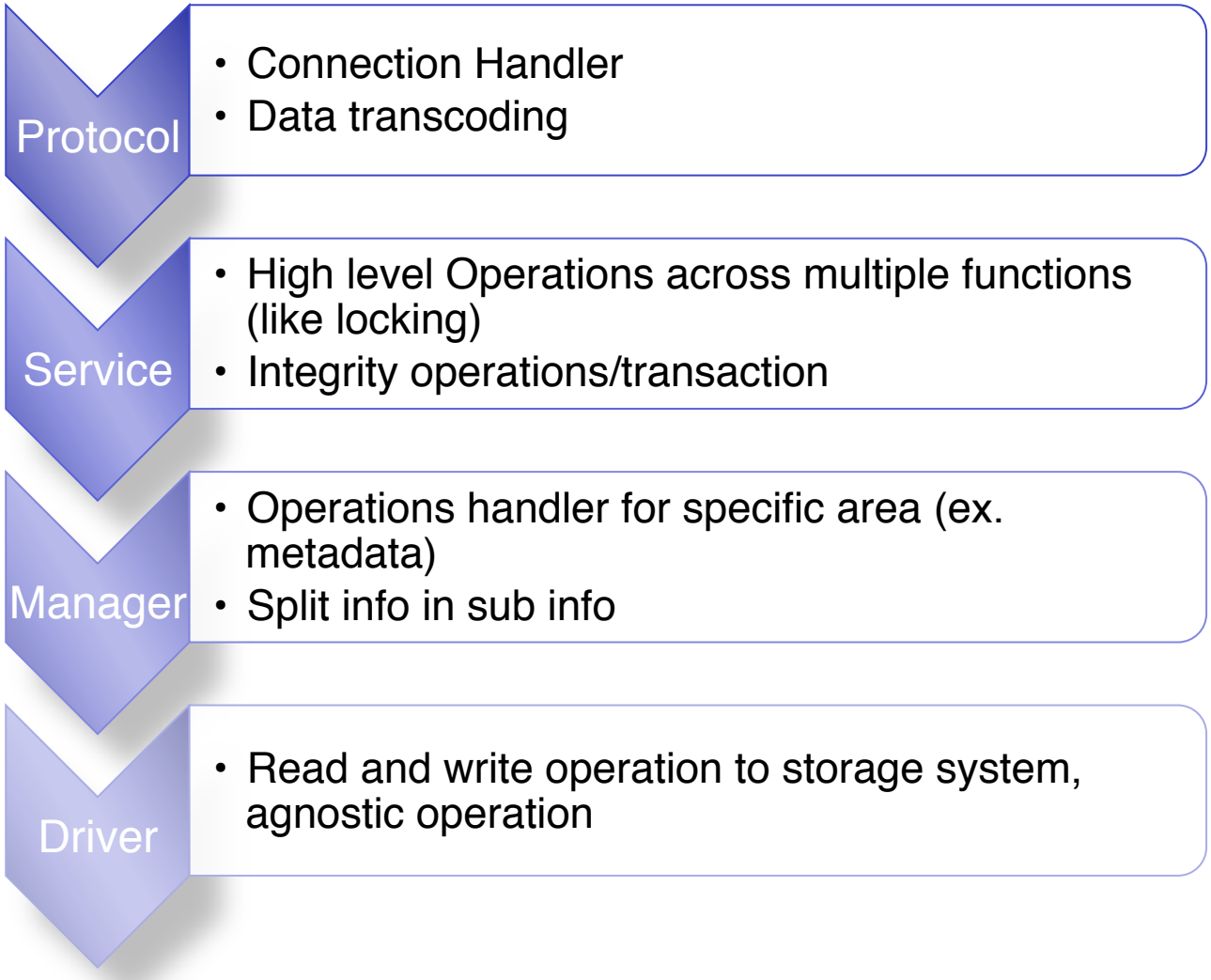


$ ./redis-benchmark -r 1000000 -n 2000000 -t get,set,lpush,lpop -P 16 –q

SET: 552028.75 requests per second
GET: 707463.75 requests per second
LPUSH: 767459.75 requests per second
LPOP: 770119.38 requests per second

Code

Interface, dynamic load



What we are using

Module	Software
Storage	Filesystem, DHT (kademlia, Pastry*)
Metadata	SQL(mysql,sqlite), Nosql (Redis)
Auth	Oauth(google, twitter, facebook), kerberos*, internal
Protocol	Websocket
Message Format	JSON-RPC 2.0, Amazon S3
Encoding	Plain, bson
CallBack	Subscribe/Publish Websocket/Redis, Async I/O TornadoWeb, ZeroMQ*
HASH	Sha-XXX, MD5-XXX, AES
Encryption	SSL, ciphers supported by crypto++
Discovery	DNS, file base

* are planned

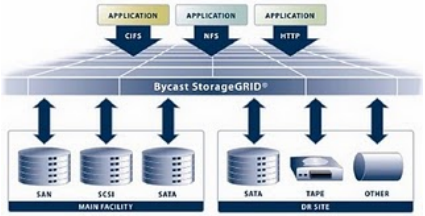
What is it good for ?

User

- Home directory
- Remote/Internet disks

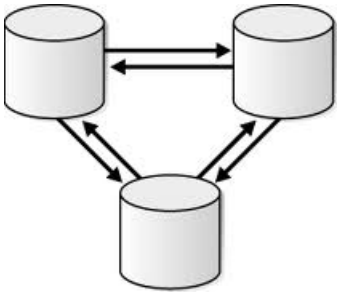
Application

- Object storage
- Shared space
- Virtual Machine



Distribution

- CDN (Multimedia)
- Data replication
- Disaster Recovery



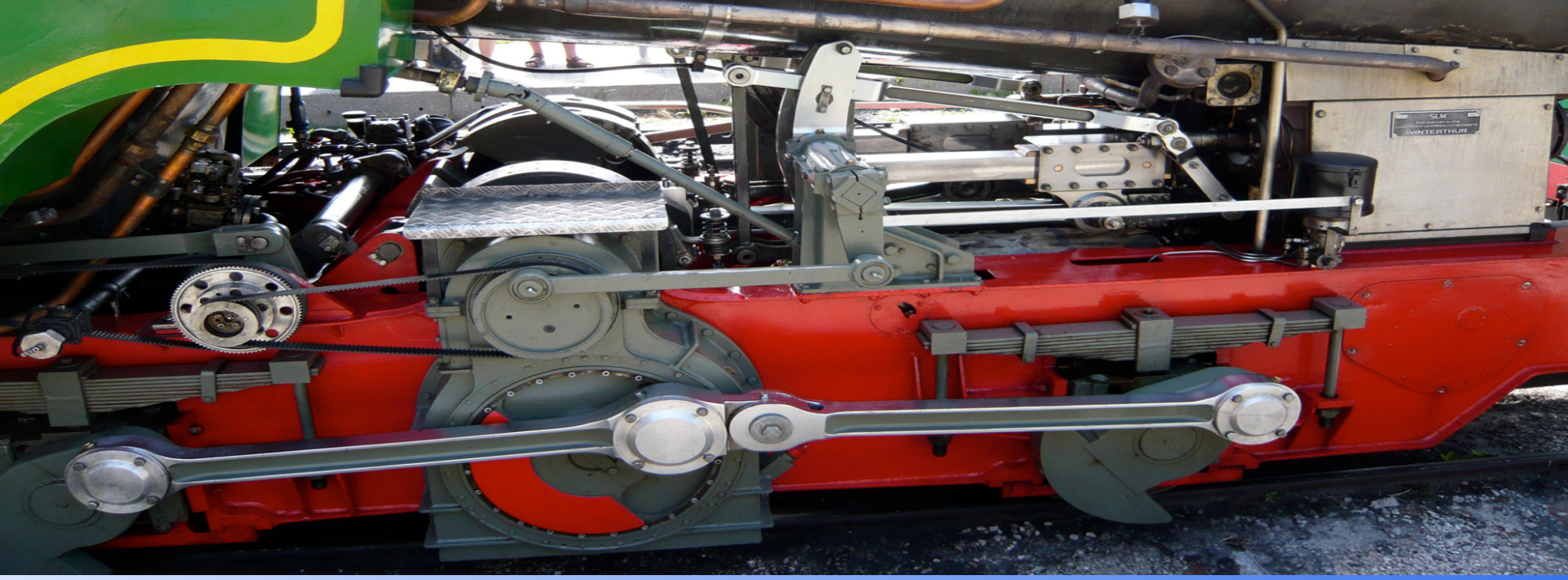
European AFS & Kerberos Conference 2014

26-28 March 2014
CERN Genève

http://indico.cern.ch/conferenceDisplay.py?confId=271400


Donate


FRAGILE



Thank you

http://restfs.beolink.org

manfred.furuholmen@gmail.com
fege85@gmail.com

Beolink.org

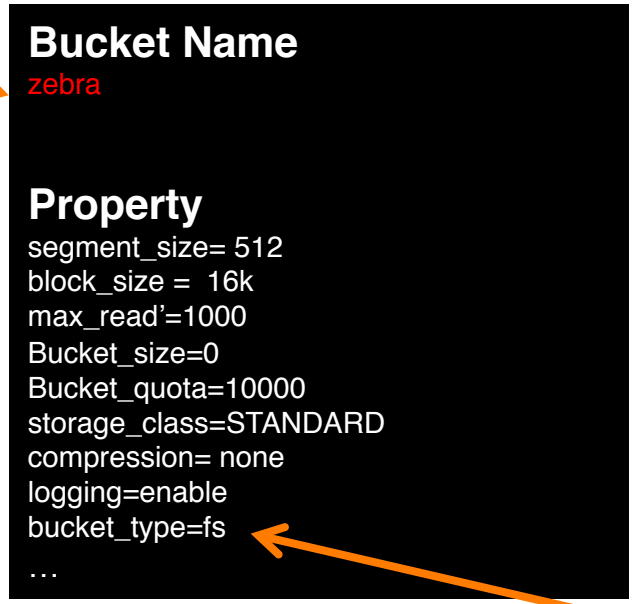
Bucket

Bucket

The bucket has many properties, the property element is a collection of object information, with this element you can retrieve the default value for the bucket (logging level, security level, ect).

Properties objects:
- Property
- Property Ext
- Property ACL
- Property Stats

Bucket Name



Default parameters

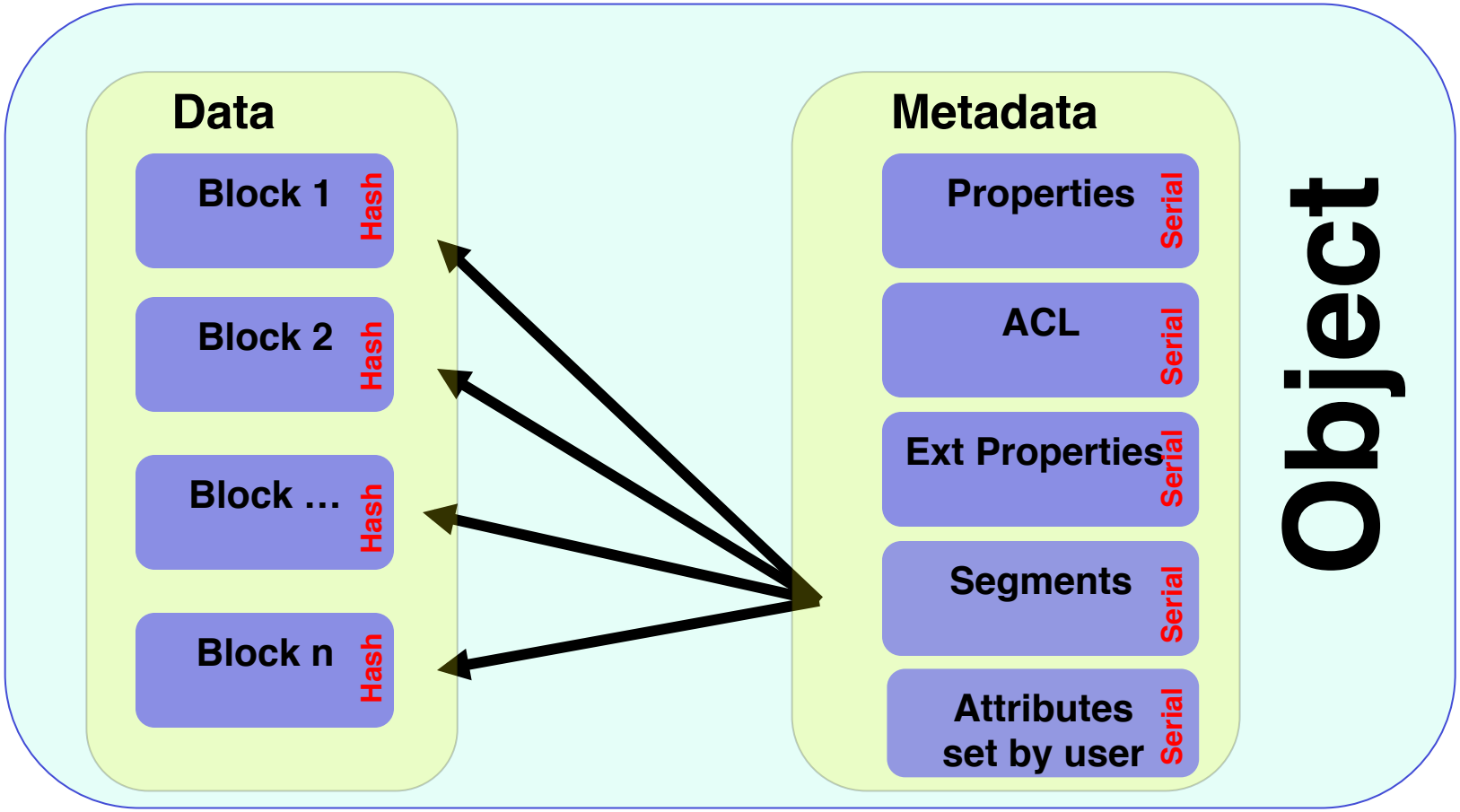
Python Dict

- **Filesystm,** The bucket is used as a filesystem
- **Logging,** Logging operation done on the specific Bucket
- **Replica RO,** Bucket shadow replication

…Custom definition

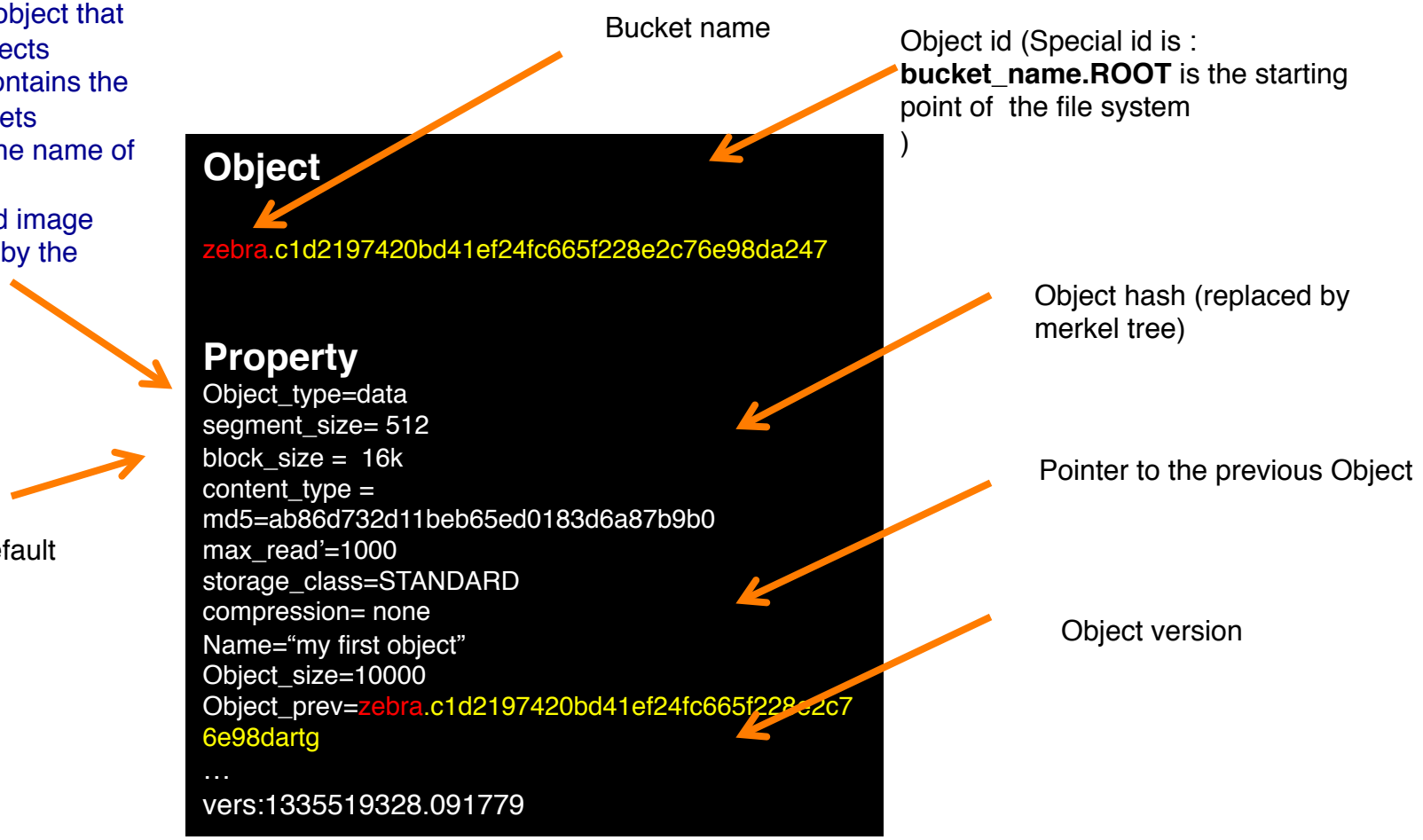
Objects

Object


Data
Block 1
Hash
Block 2
Hash
Block ...
Hash
Block n
Hash
Metadata
Properties
Serial
ACL
Serial
Ext Properties
Serial
Segments
Serial
Attributes set by user
Serial
Object

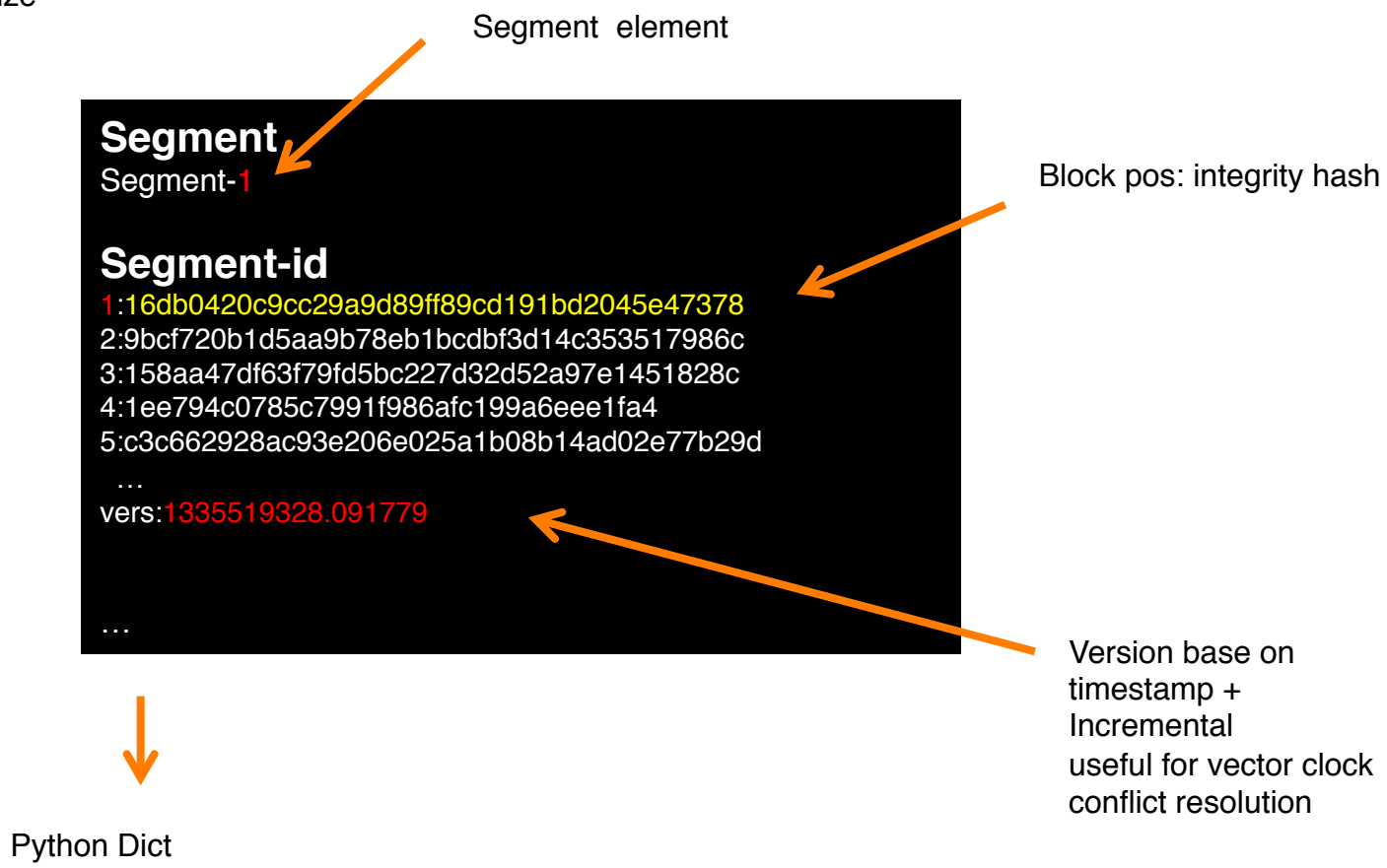
MetaData Properties

Object type:
- **Data,** Contains files
- **Folder,** Special object that contain others objects
- **Mount point,** Contains the name of the buckets
- **Link,** Contains the name of the objects
- **Immutable,** Gold image
Custom, Defined by the users

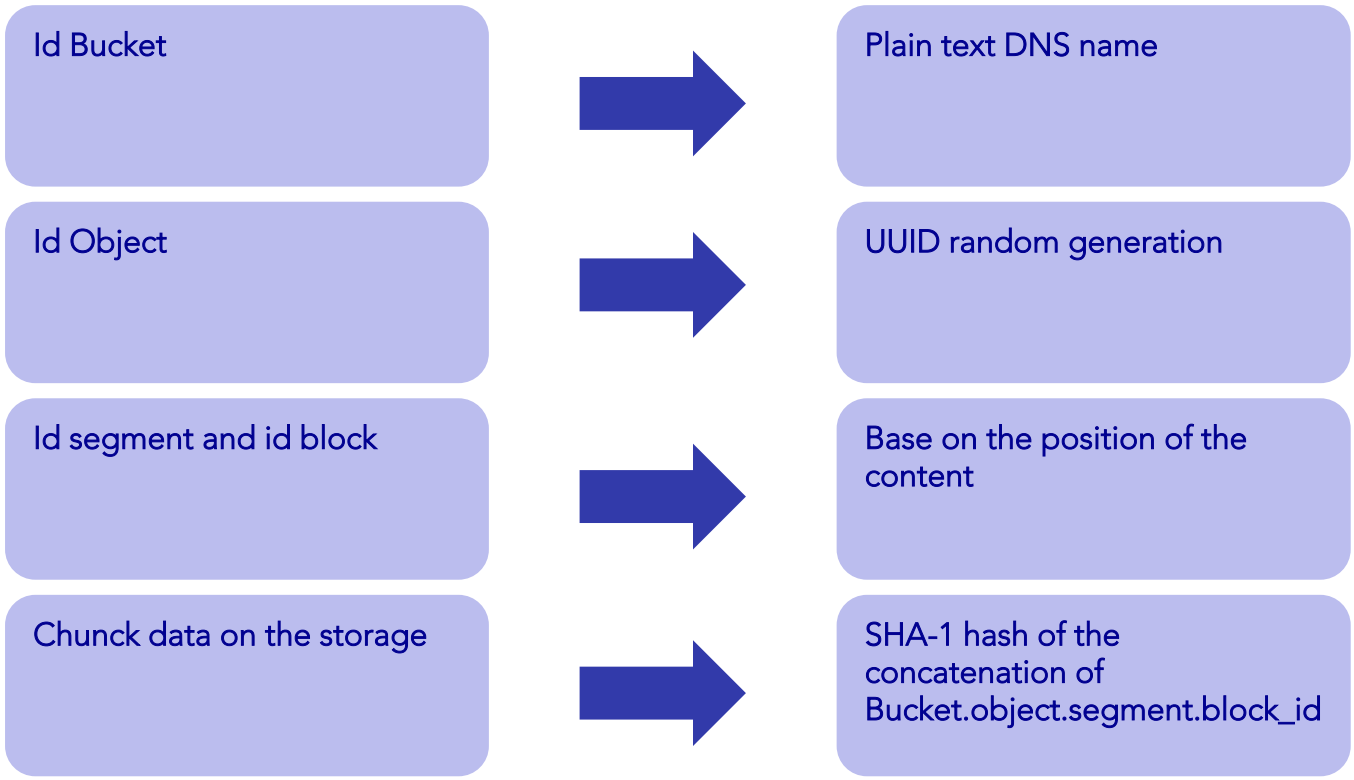


Metadata Segment

Data_size
---------- = Total Segment
block_size*segment_size

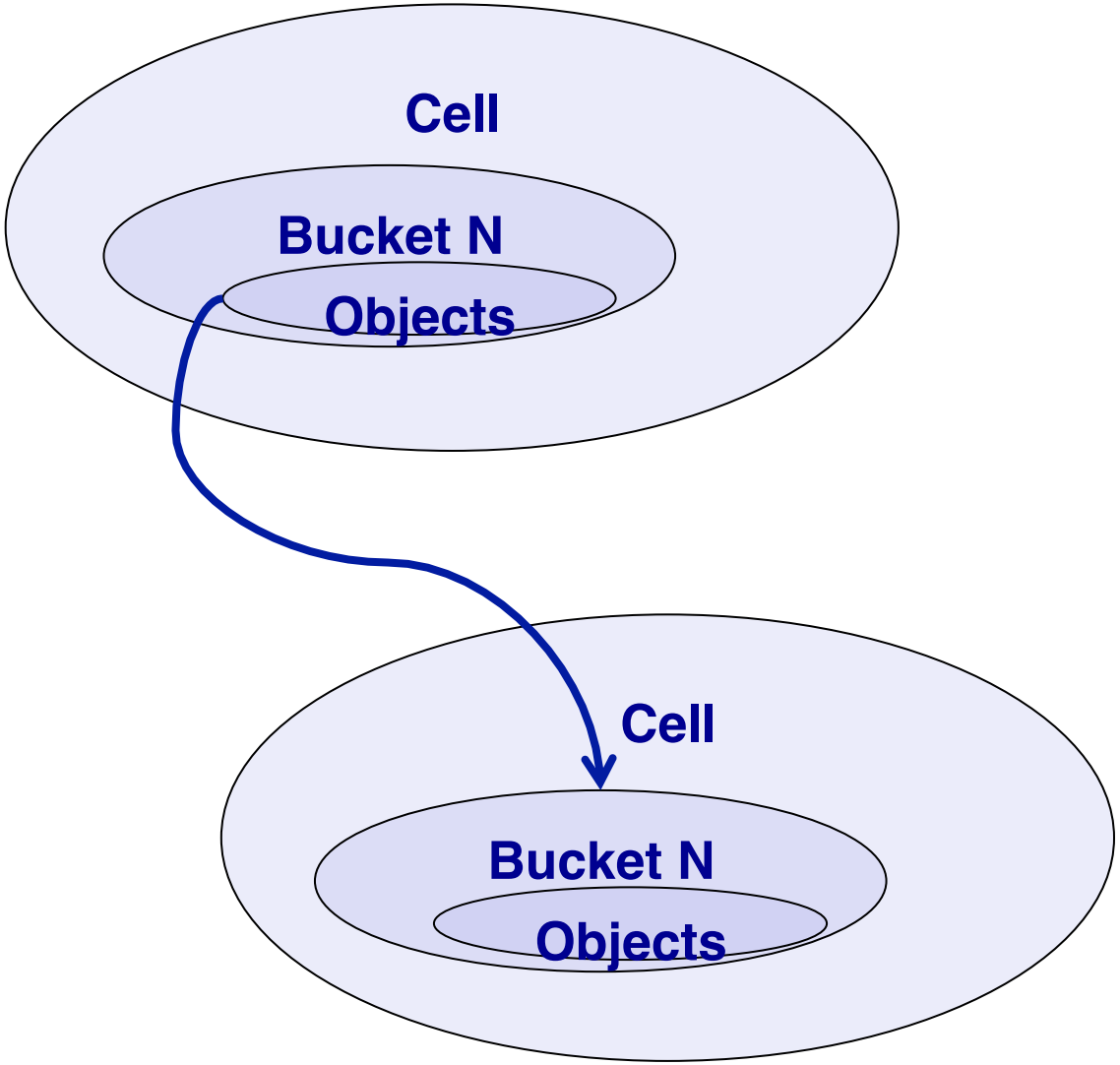


Restfs ID

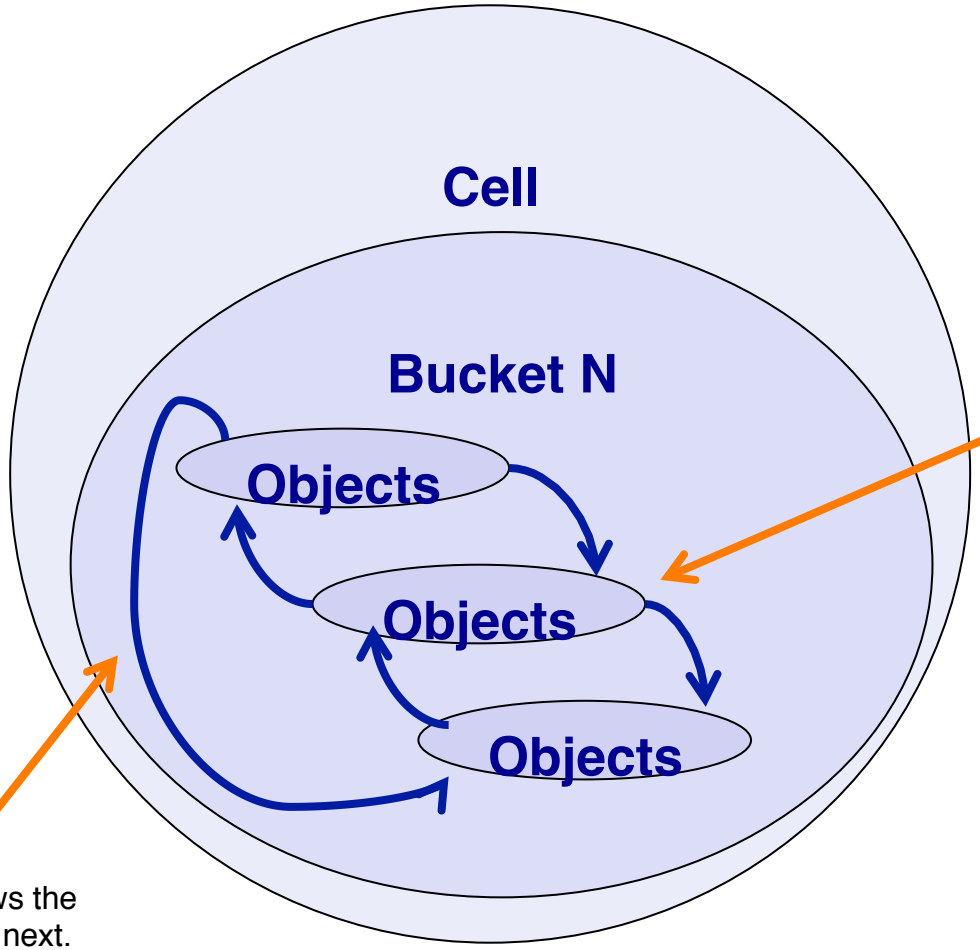


Id Object is unique inside of the Bucket, with bucket name the id is a UUID

Mounting



Object Versioning



The segment contain the diff to upstream object

Each object knows the previous and the next. The current object knows the previous and the last

Protocols

RestFS Protocol

Beolink.org

Standard HTTP/
HTTPS port

```
GET /mychat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==
Sec-WebSocket-Protocol: chat
Sec-WebSocket-Version: 13
Origin: http://example.com
```

WebSocket

is a web technology for multiplexing bi-directional, full-duplex communications channels over a single TCP connection.

Simple to covert in
python dict

```
--> { "method": ”readBlock", "params": [”…"], "id": 1}
<-- { "result": [..], "error": null, "id": 1}
```

JSON-RPC

is lightweight remote procedure call protocol similar to XML-RPC. It's designed to be simple

```
{"hello": "world"}
→
"\x16\x00\x00\x00\x02hello\x00
 \x06\x00\x00\x00world\x00\x00"
```

BSON

short for Binary JSON, is a binary-encoded serialization of JSON-like documents..
BSON can be compared to binary interchange formats

*Compression is a long story…

Protocols Metadata


```
→{ "method": ”readBlock", "params": [“
bucket_name: test,
segment:1   ,
blocks:[1,2,3,4]"],
"id": 1}
```

Collecting per segment

Parallel request per segment

```
→{ "method": ”getSegmentVer", "params": [“
bucket_name: test,
segment:1   ,
, "id": 1}

<-- { "result": [
ver: 1335519328.091779
],
 "error": null, "id": 1}
```

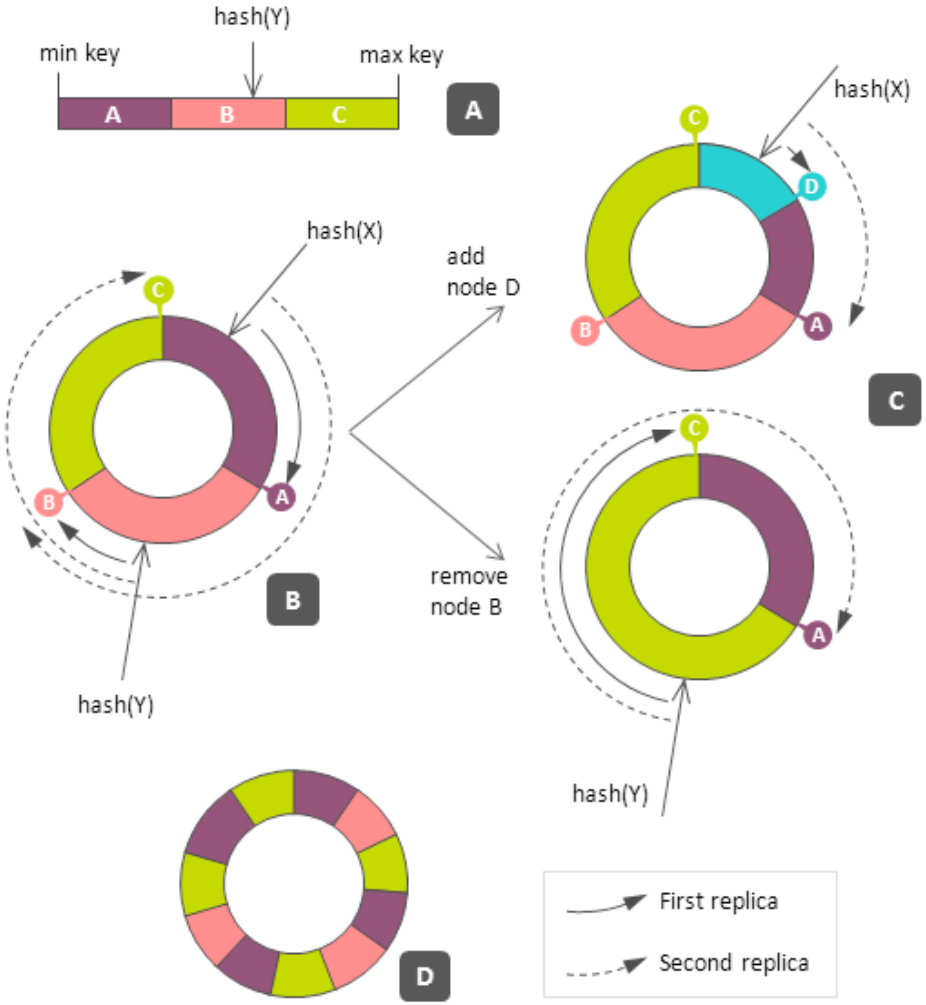
Check cached Data

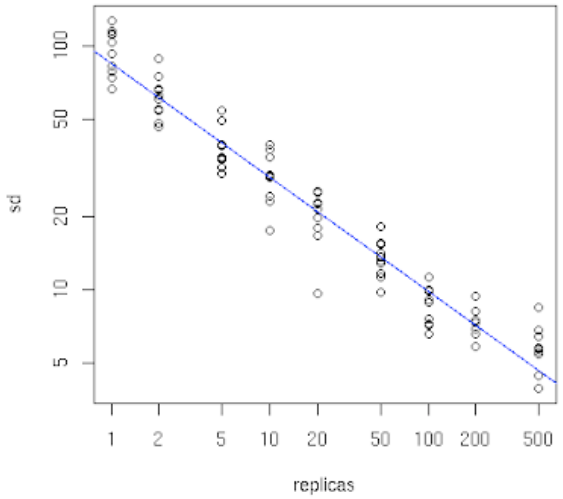
```
→{ "method": ”getSegmentHash", "params": [“
bucket_name: test,
segment:1   ,
, "id": 1}

<-- { "result": [
1:16db0420c9cc29a9d89ff89cd191bd2045e47378
2:9bcf720b1d5aa9b78eb1bcdbf3d14c353517986c
…
],
 "error": null, "id": 1}
```

Block hash list for a specific segment

Block Storage

Backend: Consistent Hashing





Number of key to move for add/ remove a node :

Keys/Node= keys to relocate

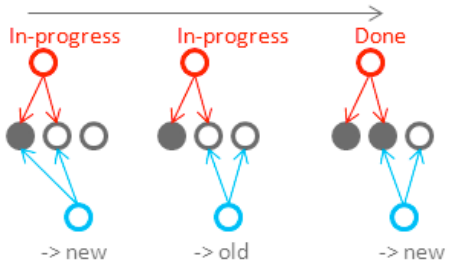
Blocks are collected in shards

http://highlyscalable.wordpress.com/2012/09/18/distributed-algorithms-in-nosql-databases/

3 Copies

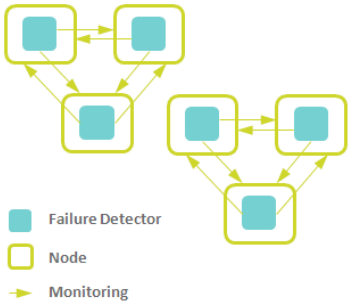
Configurable read and write consistent level and security:

- 2W1R
- 2W2R
- 1W1R
- …



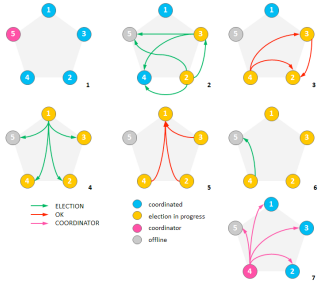
Monitor of neighbored

small cluster of 3 nodes (GOSSIP)

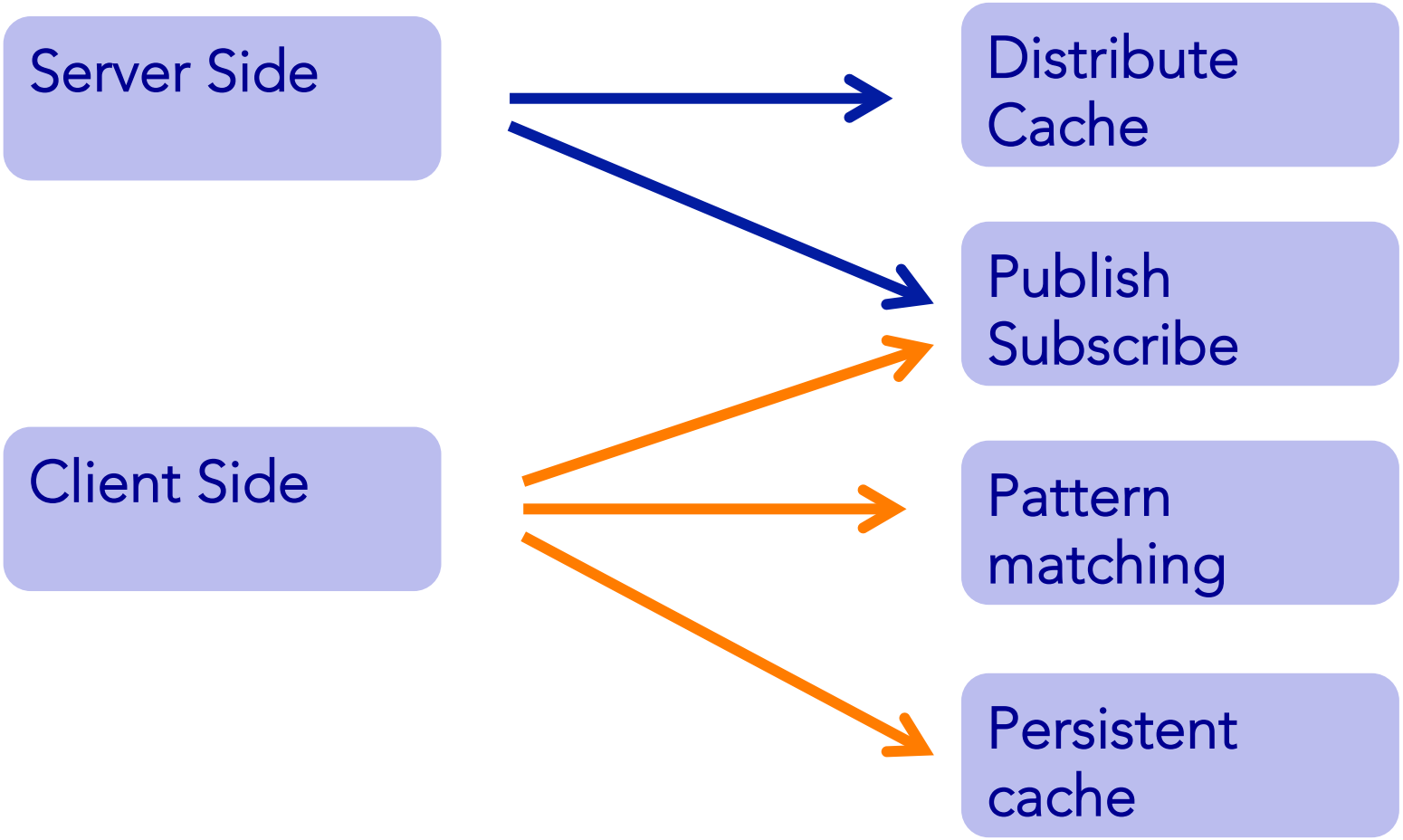


Mini cluster election

key space reclaim for replica coordination, leave join cluster



Cache


Server Side
Client Side
Distribute Cache
Publish Subscribe
Pattern matching
Persistent cache

Security

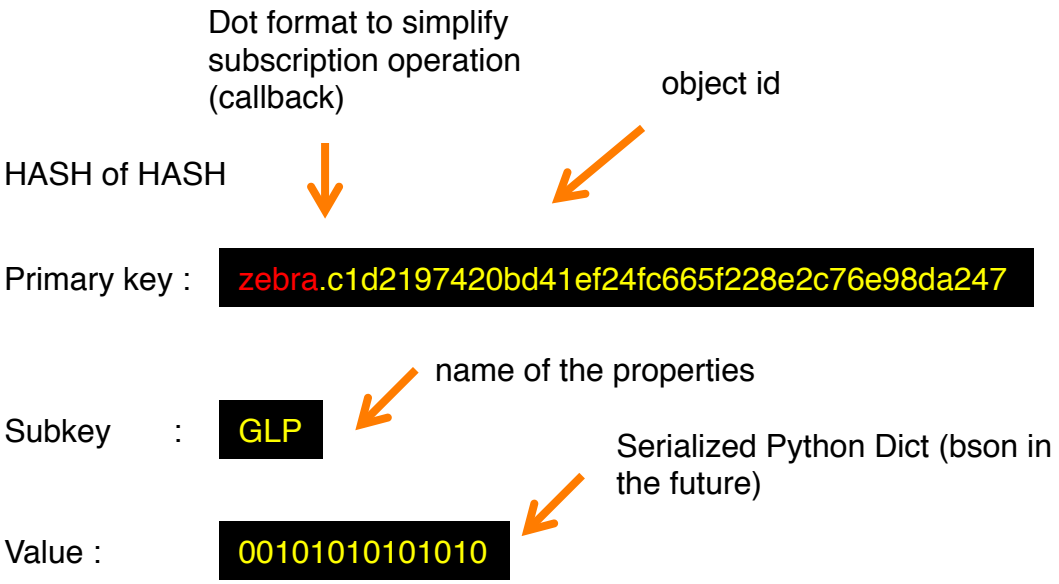
Security


Protocol,
• SSL Protocol
Authentication
• Token for devices (Enrollment)
• Session Token for User
• External password provider
Data Integrity
• Encryption on block level
Authorization
• Extended ACL based on NFS4 ACL
• Admin delegation on the Bucket level

NOSQL **DB**

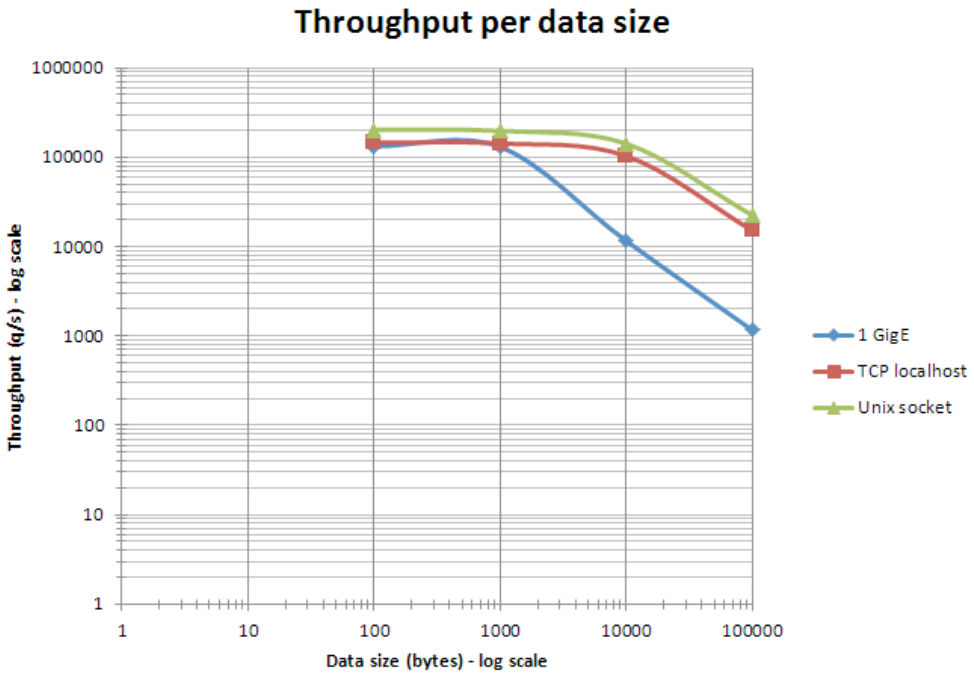
Main characteristics

- Fast
- Store Hash of HASH
- Atomic operation
- Sub/Pub primitives



* Version and Hash of the objects has a dedicated subkey, no serialization

Redis performance

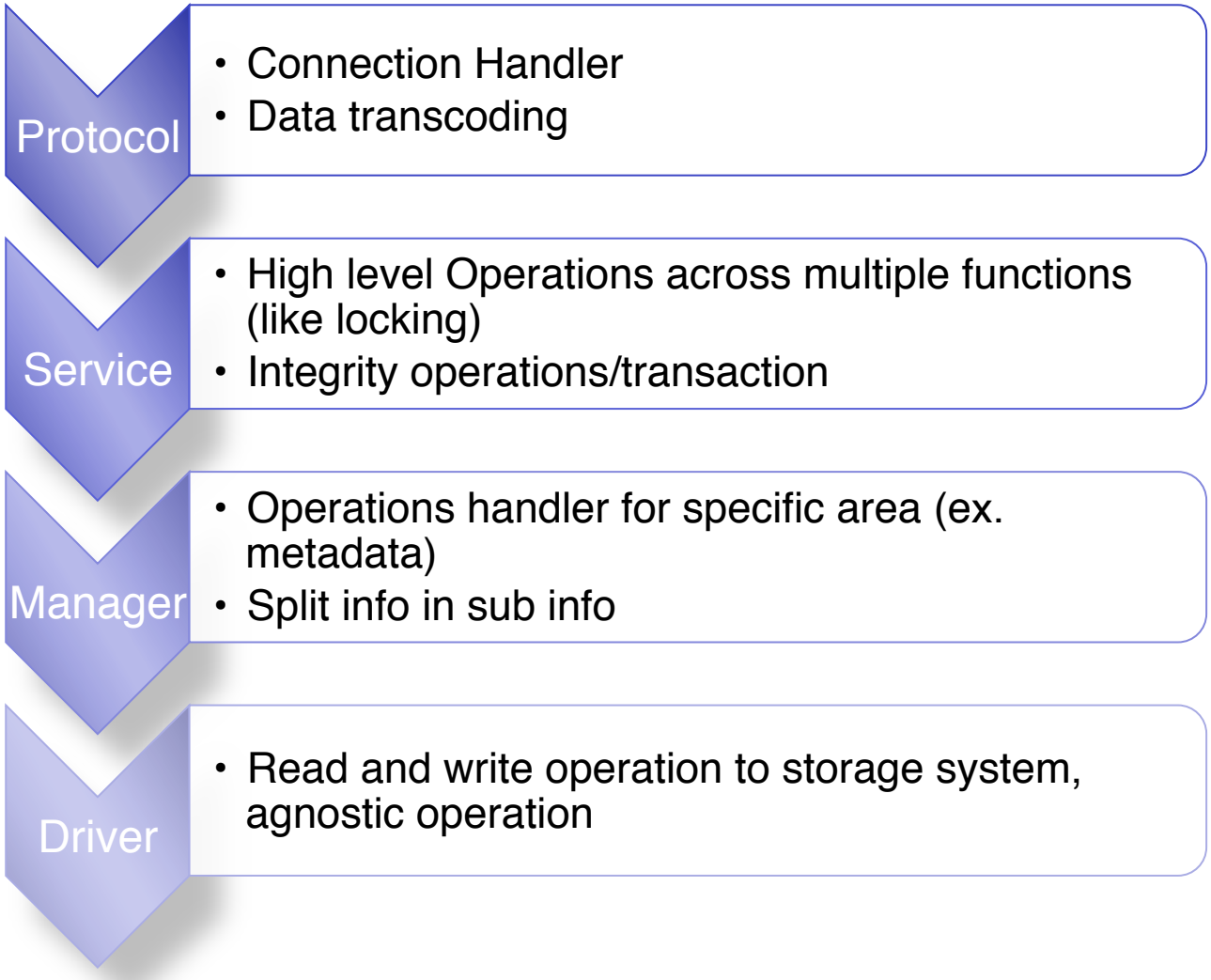


$ ./redis-benchmark -r 1000000 -n 2000000 -t get,set,lpush,lpop -P 16 –q

SET: 552028.75 requests per second
GET: 707463.75 requests per second
LPUSH: 767459.75 requests per second
LPOP: 770119.38 requests per second

Code

Interface, dynamic load



What we are using

Module	Software
Storage	Filesystem, DHT (kademlia, Pastry*)
Metadata	SQL(mysql,sqlite), Nosql (Redis)
Auth	Oauth(google, twitter, facebook), kerberos*, internal
Protocol	Websocket
Message Format	JSON-RPC 2.0, Amazon S3
Encoding	Plain, bson
CallBack	Subscribe/Publish Websocket/Redis, Async I/O TornadoWeb, ZeroMQ*
HASH	Sha-XXX, MD5-XXX, AES
Encryption	SSL, ciphers supported by crypto++
Discovery	DNS, file base

* are planned

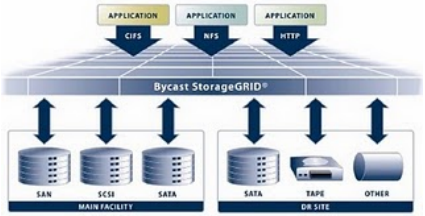
What is it good for ?

User

- Home directory
- Remote/Internet disks

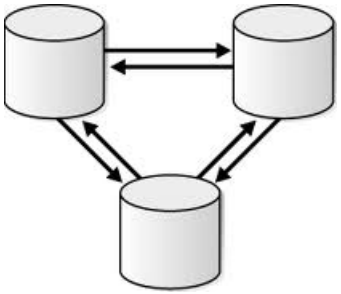
Application

- Object storage
- Shared space
- Virtual Machine

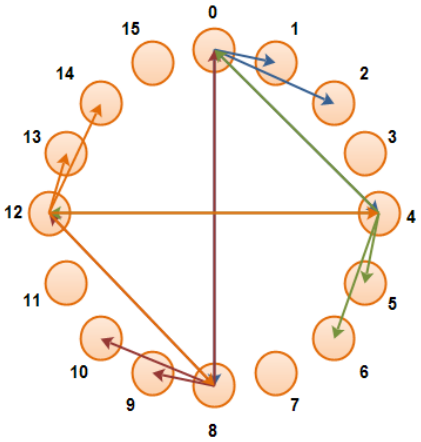


Distribution

- CDN (Multimedia)
- Data replication
- Disaster Recovery



Backend: Storage



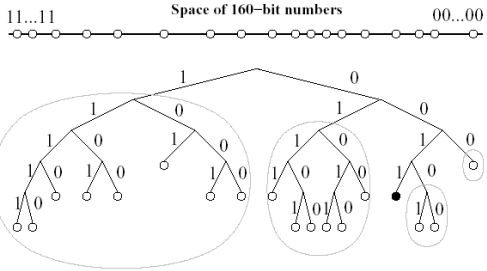


Fig. 1: Kademlia binary tree. The black dot shows the loation of node 0011... in the tree. Grey ovals show subtrees in which node 0011... must have a contact.

Transport Layer ZeroMQ

Storage Compressed DAta