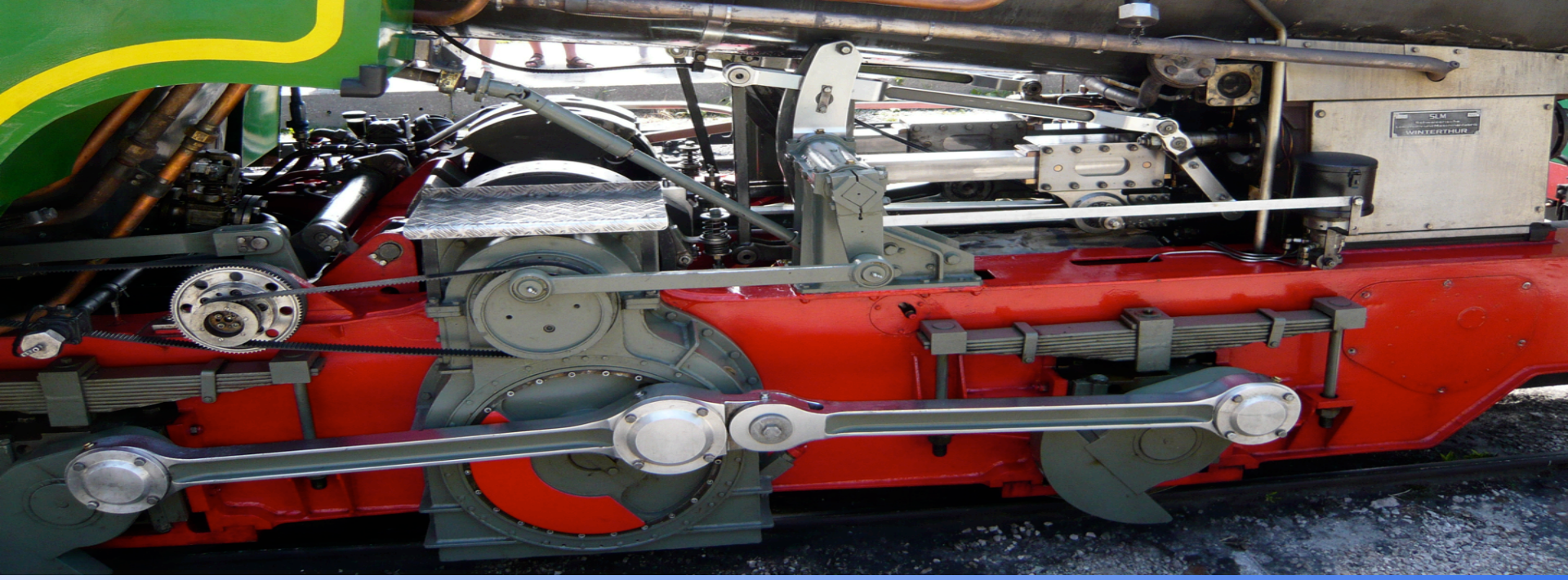


RestFS

Fabrizio Manfredi Furuholmen

Beolink.org

Agenda

RestFS

- Introduction
  - Storage System
  - Storage evolution
- RestFS
  - Goals
  - Architecture
  - Internals
  - Configuration and Deploy
- Samba
  - Interaction
  - Disaster Recovery

Introduction

John H. Terpstra

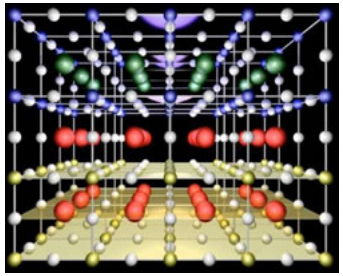
"Data stored globally is expected to grow by 40-60% compounded annually through 2020. Many factors account for this rapid rate of growth, though one thing is clear – the information technology industry needs to rethink how data is shared, stored and managed…"

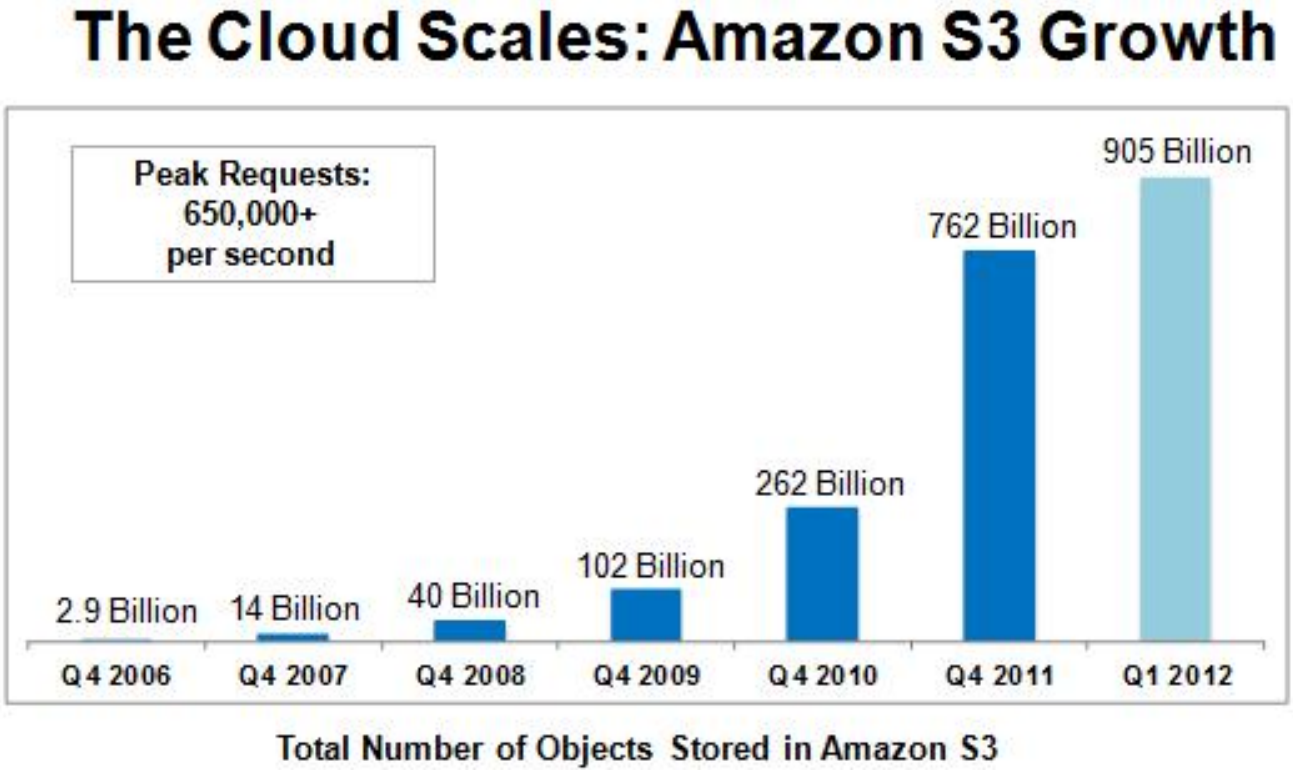
From the Chairman of SambaXP 2012


Web
3.0
Disaster
Recovery
Store It All,
Serve It
Worldwide
PC
New
Device
(TV,Tablet,
mobile)
VM

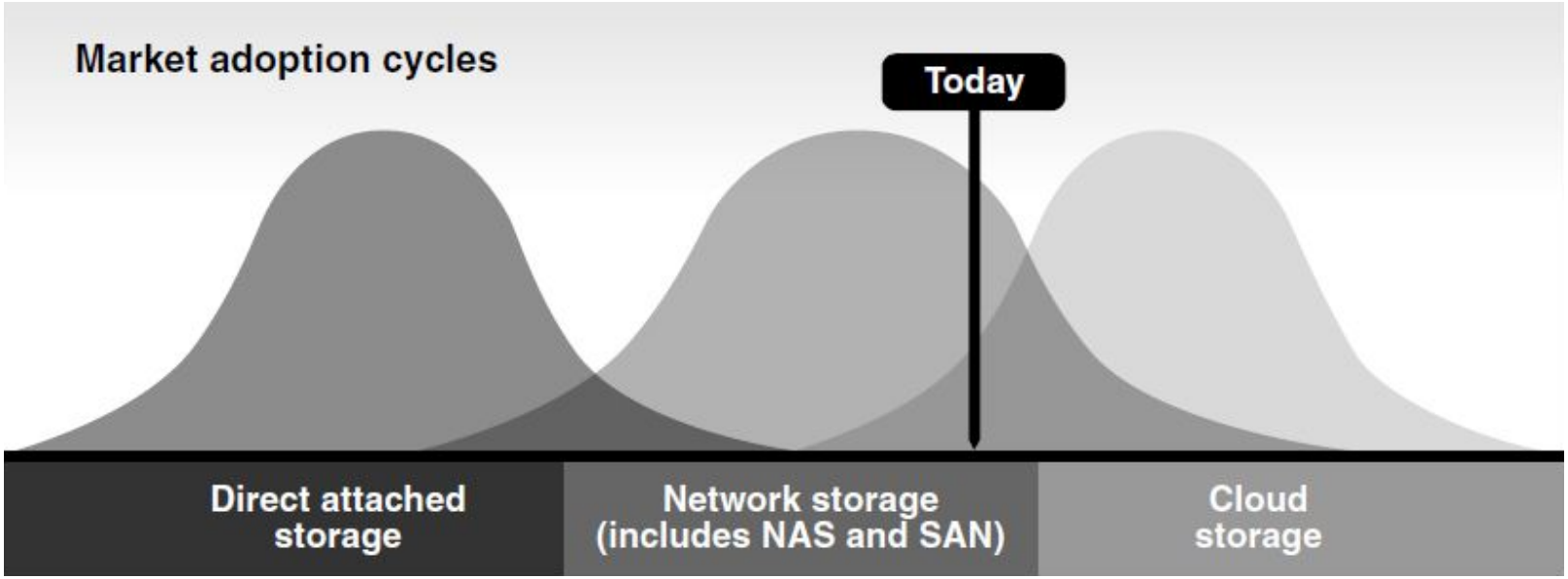
Introduction

70's	Inode
	Tree view
80's	Network filesystem (NFS/OpenAFS)
	RPC
90's	Object Storage (OSD)
	Parallel transfer
00's	Storage Service
	WEB Base
	Key Value




The Cloud Scales: Amazon S3 Growth
Peak Requests:
650,000+
per second
2.9 Billion
14 Billion
40 Billion
102 Billion
262 Billion
762 Billion
905 Billion
Q4 2006
Q4 2007
Q4 2008
Q4 2009
Q4 2010
Q4 2011
Q1 2012
Total Number of Objects Stored in Amazon S3

John's words + new usage + new services + …



The Perfect Solution

Uniform Access

- Global name support

Security

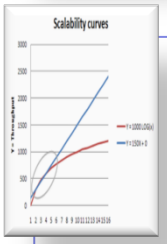
- Global authentication/ authorization

Reliability

- No single point of failure

Availability

- Maintenance without disrupting the user's routines

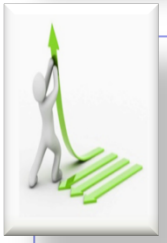


Scalability

- **T**era/Peta/… bytes of data

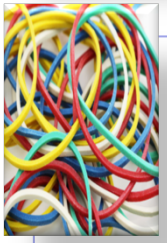
Standard conformance:

Standard semantics



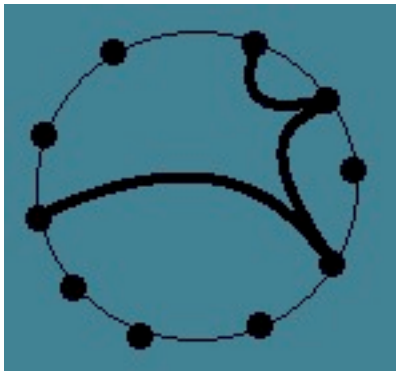
Performance:

- **H**igh performance



Elastic

- Bandwidth and capacity on demand



RestFS

The RestFS is an experimental open-source project with the goal to create a distributed FileSystem for large environments.

It is designed to scale up from a single server to thousand of nodes and delivering a high availability storage system

“Moving Computation is Cheaper than Moving Data”

Main characteristics

Beolink.org

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

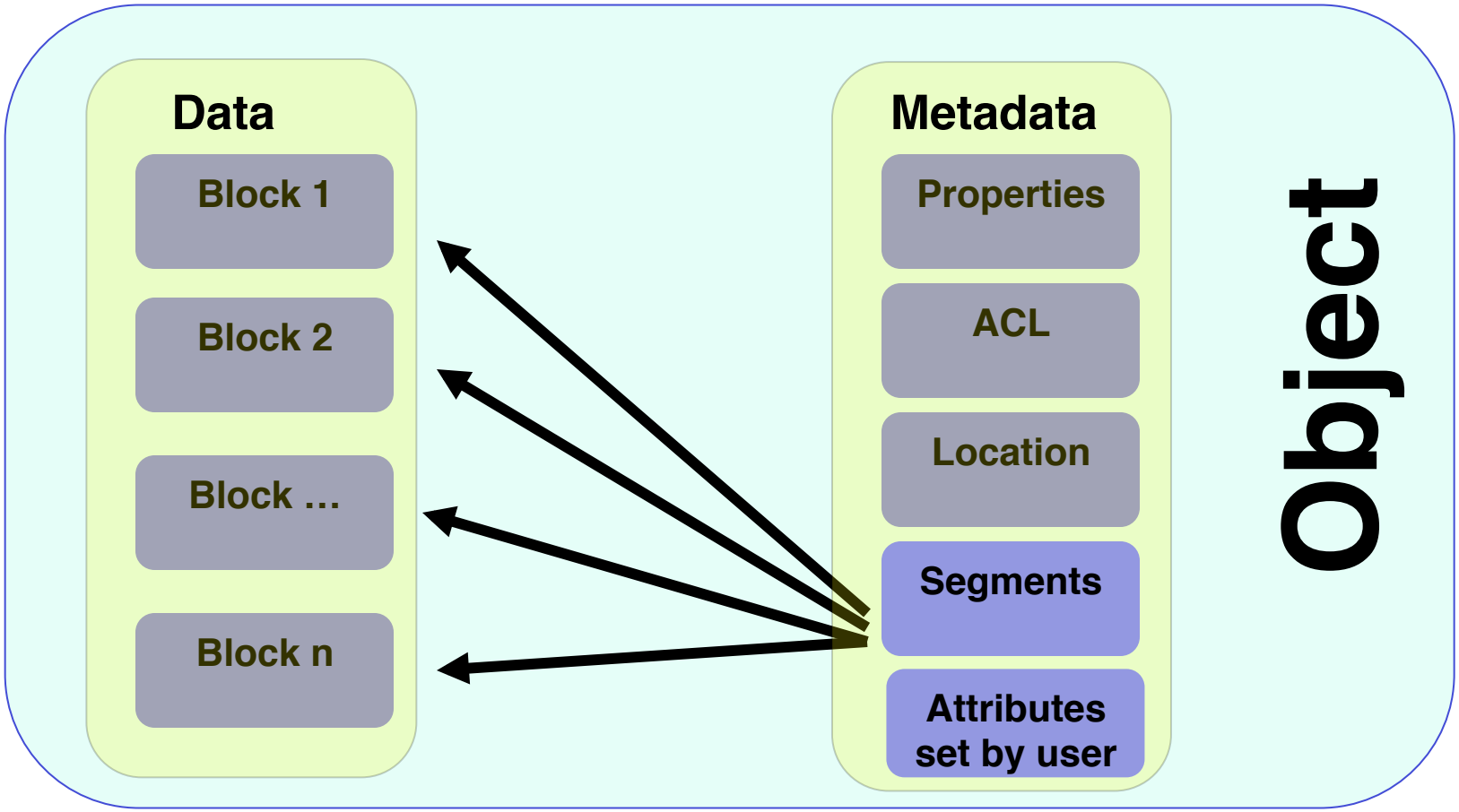
Security

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

RestFS Key Words

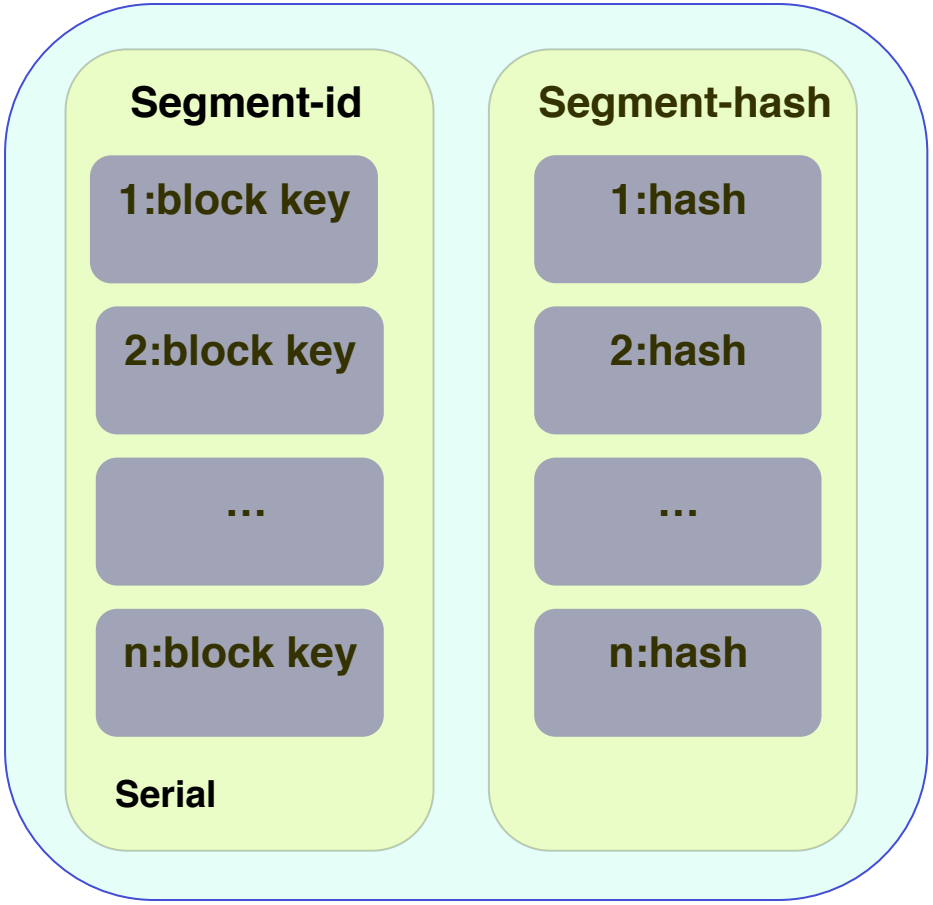
- **Cluster**, collection of servers
- **Bucket**, virtual container (volume)
- **Object**, entity (file, dir, …) contained in a Bucket
  - **Metadata**, collection of attributes owned by Object
  - **Segment**, collection of block owned by Object
  - **Block**, smallest element of data

Object



*The block size is defined by object property

Segment



Segment-id

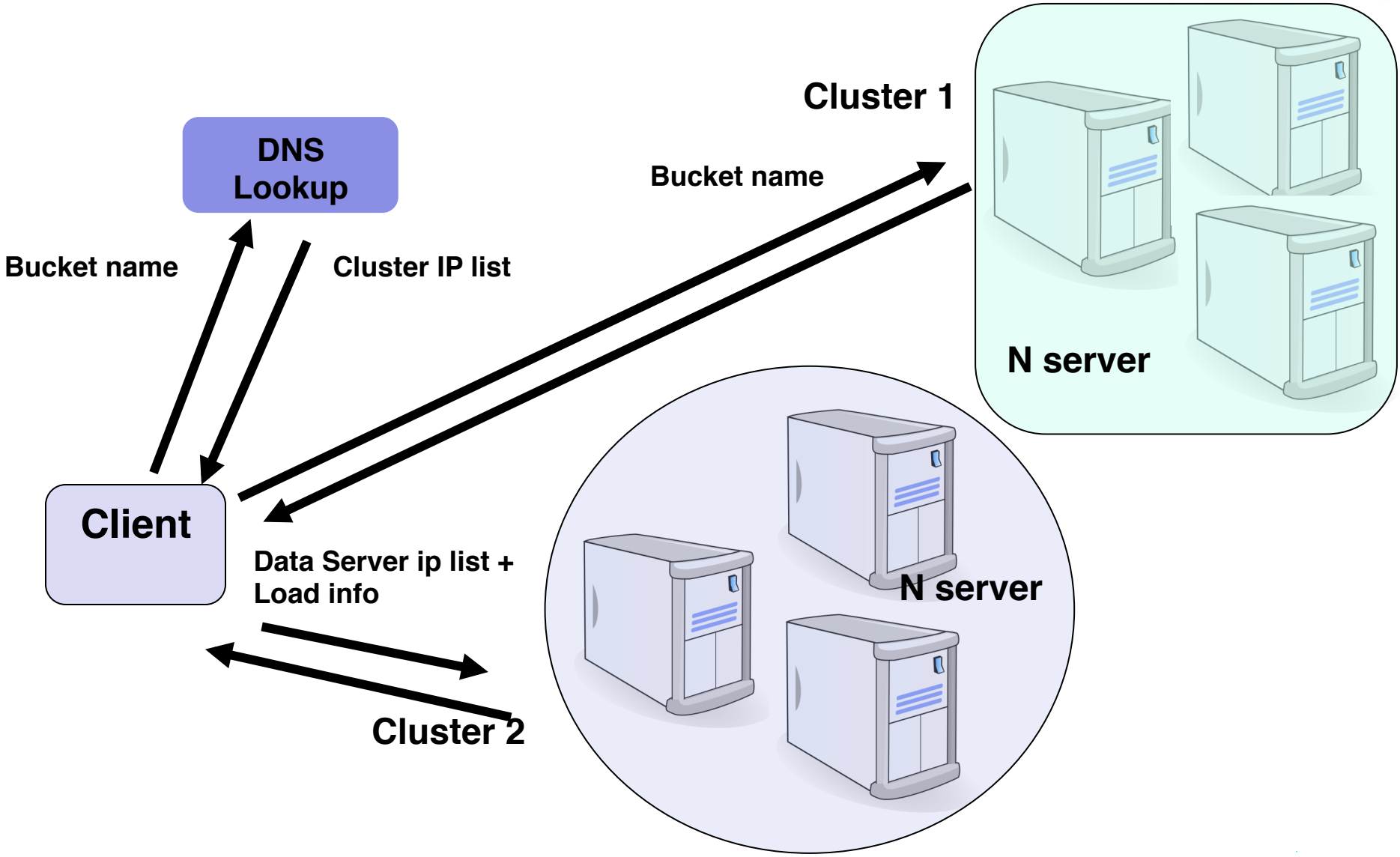
1:zebra.16db0420c9cc29a9d89ff89cd191bd2045e47378
2:zebra.9bcf720b1d5aa9b78eb1bcdbf3d14c353517986c
3:zebra.158aa47df63f79fd5bc227d32d52a97e1451828c
4:zebra.1ee794c0785c7991f986afc199a6eee1fa4
5:zebra.c3c662928ac93e206e025a1b08b14ad02e77b29d
…
vers:1335519328.091779

Segment-hash

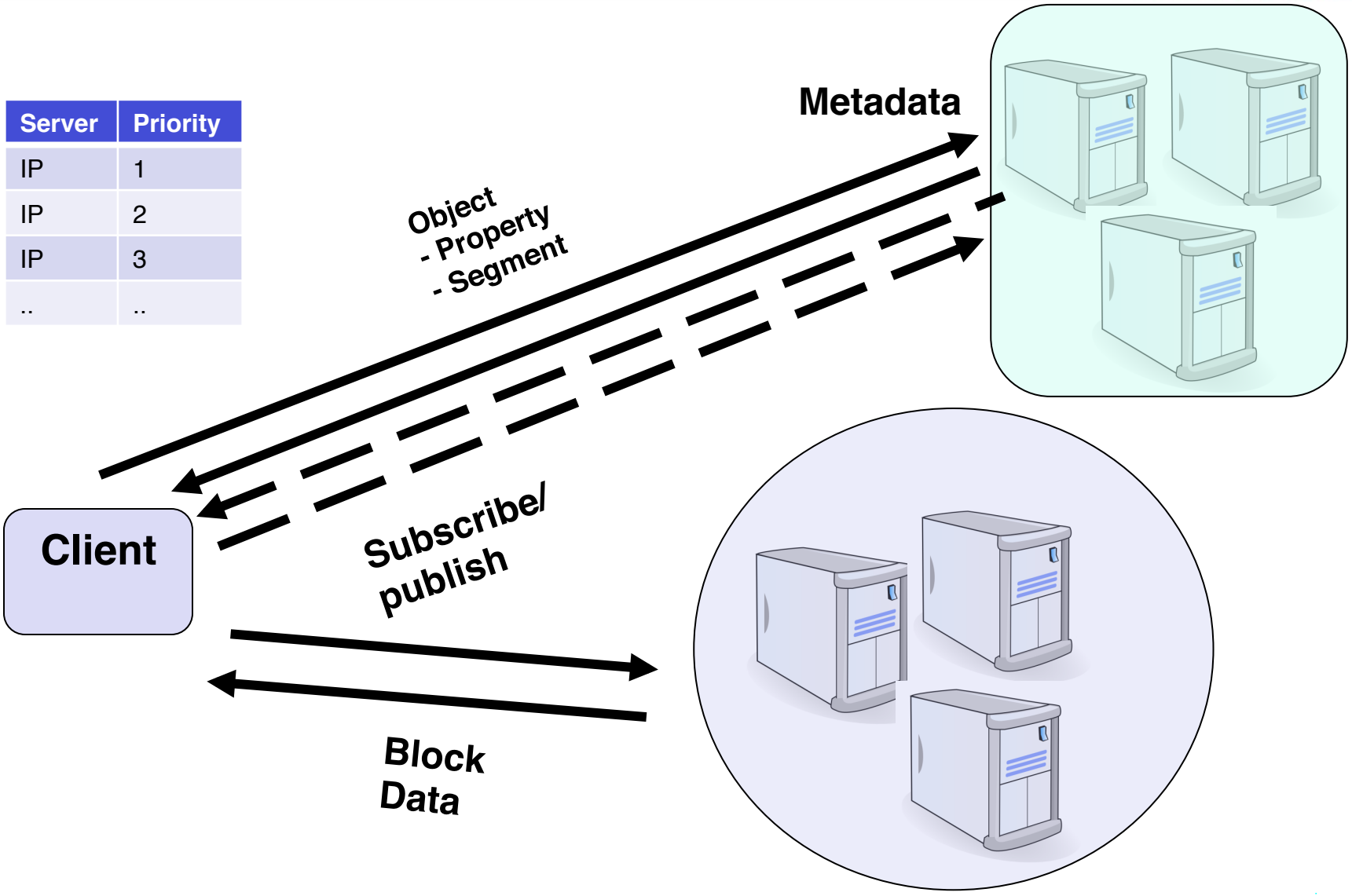
1:7d565defe000db37ad09925996fb407568466ce0
2:cc6c44efcbe4c8899d9ca68b7089506b7435fc74
3:660db9e7cd5b615173c9dc7daf955647db544580
4:fb8a076b04b550ff9d1b14a2bc655a29dcb341c4
5:b2c1ace2823620e8735dd0212e5424da976f27bc
...

*The segment size is defined by object properties

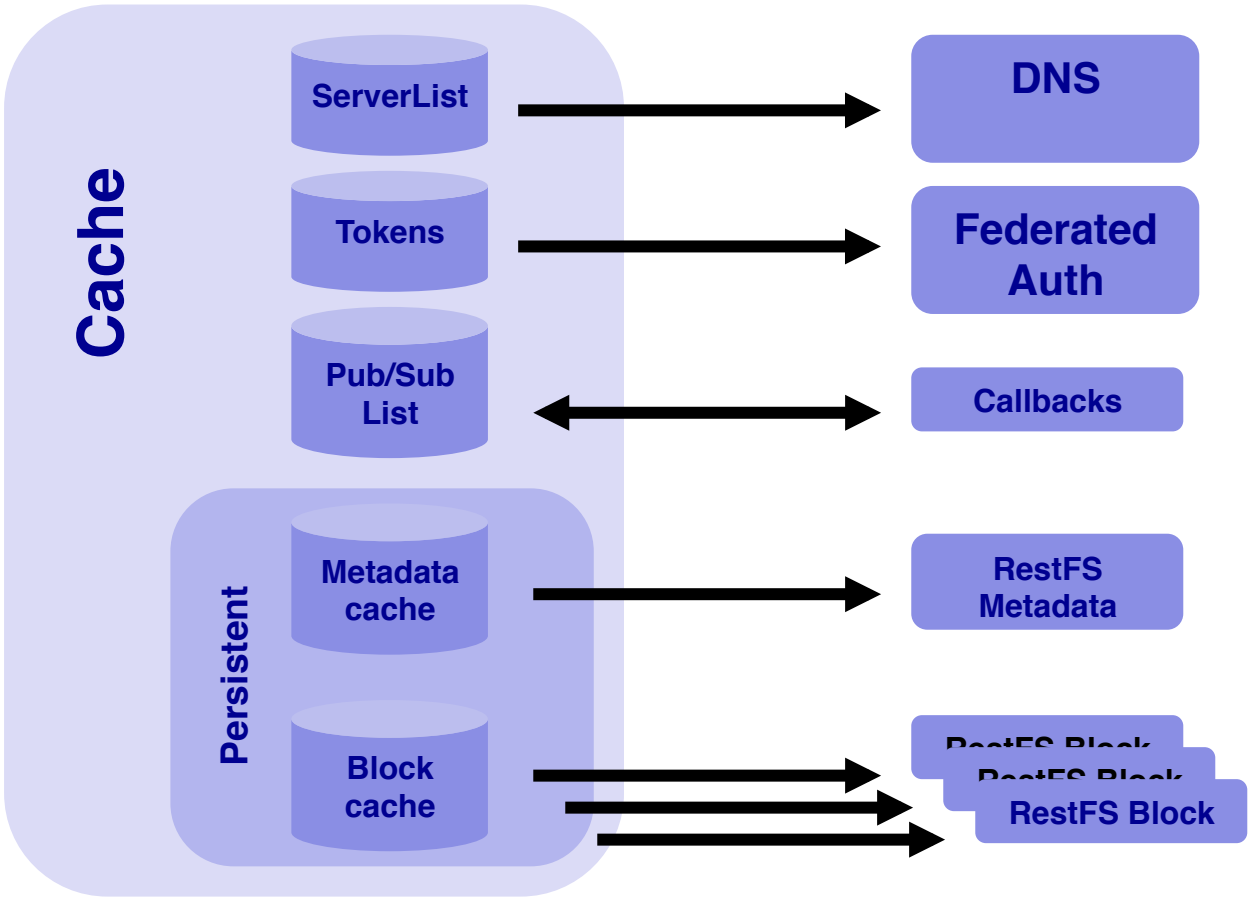
Cluster Discovery



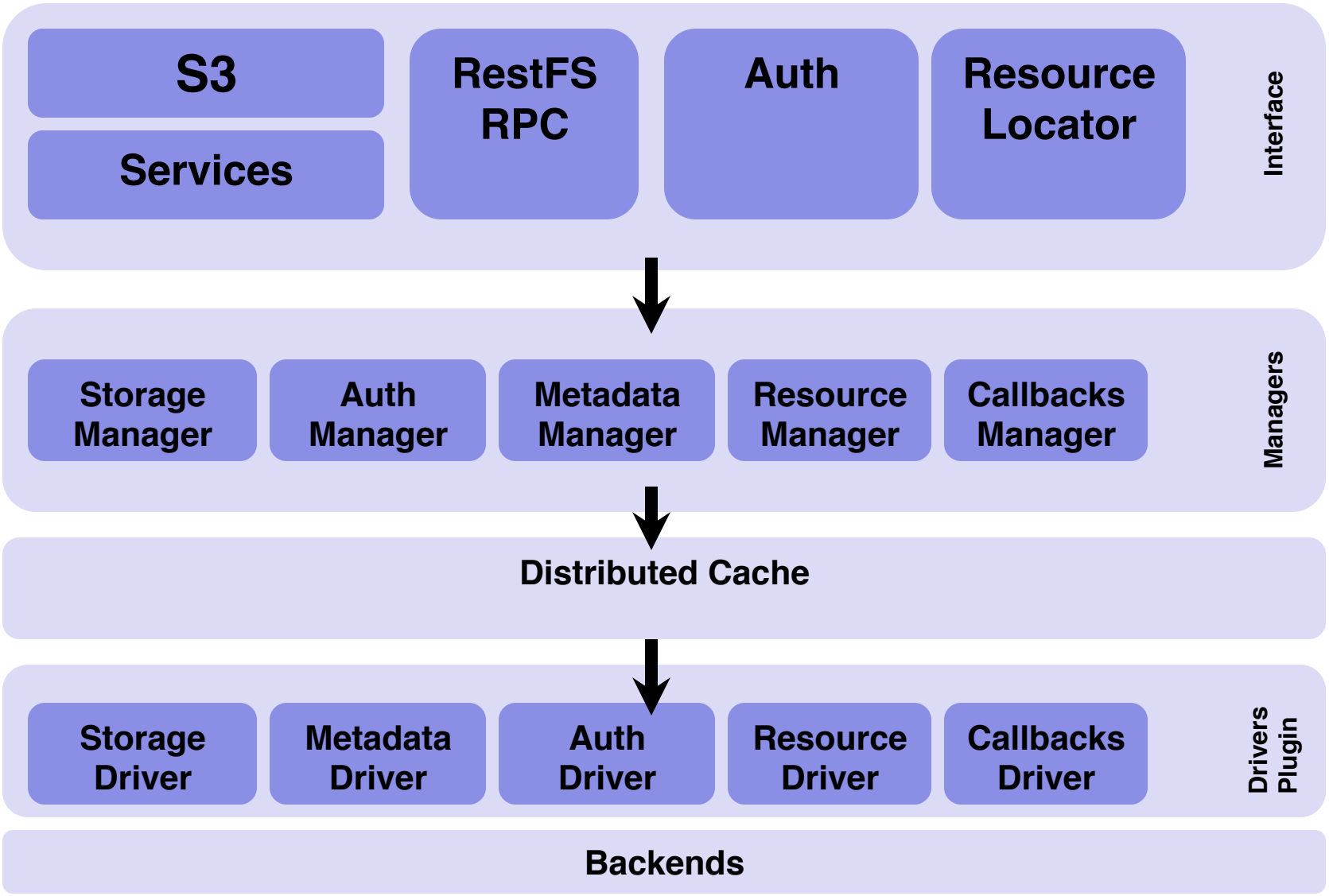
Server	Priority
IP	1
IP	2
IP	3
..	..



Client side with Pub/Sub



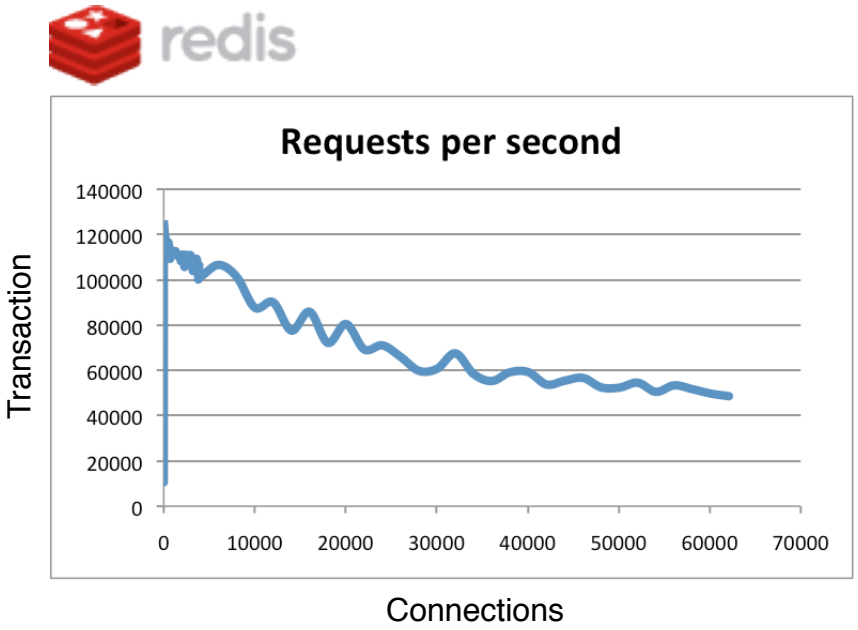
Server Architecture



What we are using

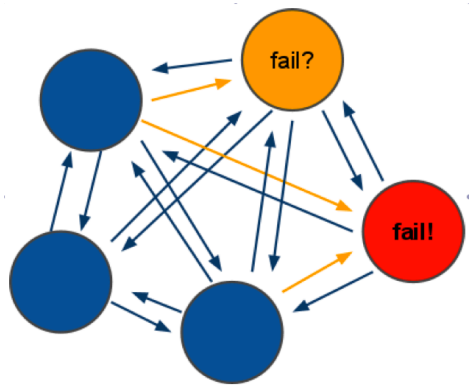
Module	Software
Storage	Filesystem, DHT (kademlia, Pastry*)
Metadata	SQL(mysql,sqlite), Nosql (Redis)
Auth	Oauth(google, twitter, facebook), kerberos*, internal
Protocol	Websocket
Message Format	JSON-RPC 2.0, Amazon S3
Encoding	Plain, bson
CallBack	Subscribe/Publish Websocket/Redis, Async I/O TornadoWeb,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

* are planned



Example of benchmark result

The test was done with 50 simultaneous clients performing 100000 requests.
The value SET and GET is a 256 bytes string.
The Linux box is running Linux 2.6, it's Xeon X3320 2.5 GHz.
Text executed using the loopback interface (127.0.0.1).



Cluster

Multi-master
Auto recovery

Backends Metadata

WebSocket

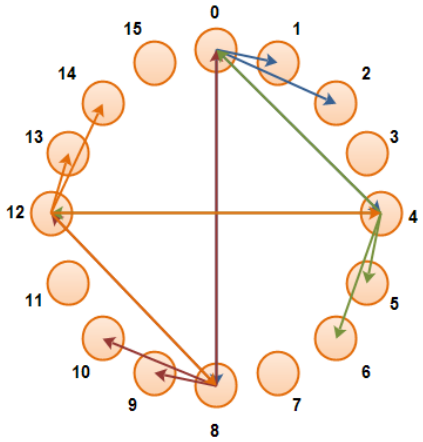
is a web technology providing for multiplexing bi-directional, full-duplex communications channels over a single TCP connection.

This is made possible by providing a standardized way for the server to send content to the browser without being solicited by the client, and allowing for messages to be passed back and forth while keeping the connection open…

Publish–subscribe

"… is a messaging pattern where senders of messages, called publishers, do not program the messages to be sent directly to specific receivers, called subscribers. Published messages are characterized into classes, without knowledge of what, if any, subscribers there may be. Subscribers express interest in one or more classes, and only receive messages that are of interest, without knowledge of what, if any, publishers there are… " Wikipedia

Demo http://www.websocket.org/echo.html



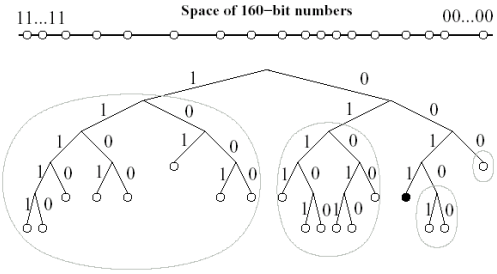


Fig. 1: Kademlia binary tree. The black dot shows the loation of node 0011... in the tree. Grey ovals show subtrees in which node 0011... must have a contact.

Kademlia's XOR distance is easier to calculate.

Kademlia's routing tables makes routing table management a bit easier.

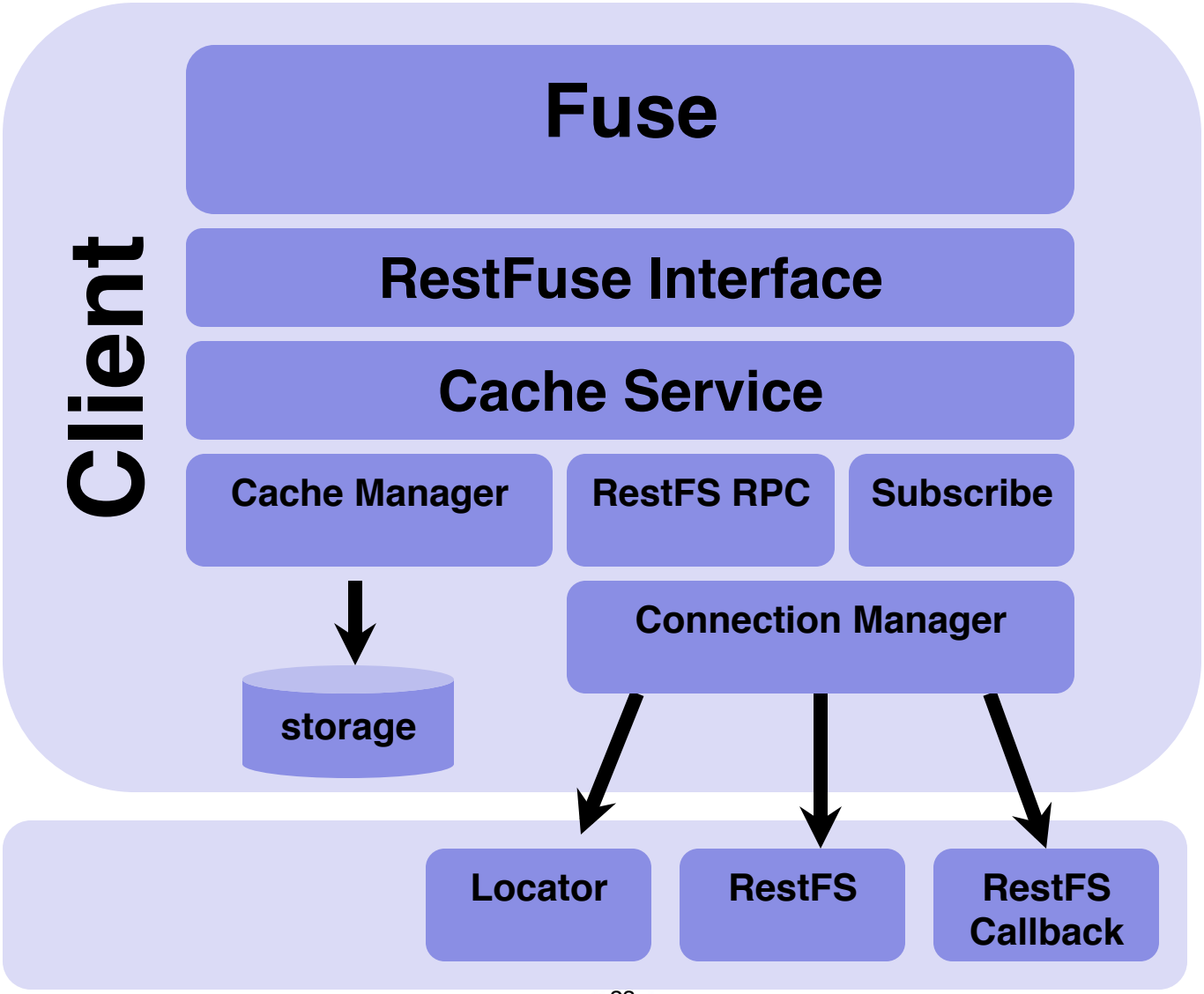
Each node in the network keeps contact information for only log n other nodes

Kademlia implements a "least recently seen" eviction policy, removing contacts that have not been heard from for the longest period of time.

Key/value pair is stored on the node whose 160-bit nodeID is closest to the key

Closest node, send a copy to neighboard

Client Architecture



What we are using

Module	Software
Storage	Filesystem
Metadata	SQL(sqlite), Filesystem (python serialize object)
Auth	internal, kerberos*
Interface	Fuse, RestFS client lib
Message Format	JSON-RPC 2.0
Encoding	Plain, bson
CallBack	Subscribe/Publish Websocket
HASH	Sha-XXX, MD5-XXX, AES
Encryption	SSL, ciphers supported by crypto++
Discovery	DNS, file base

Examples

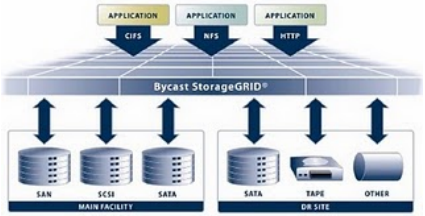
Introduction: Usage

User

- Home directory
- Remote/Internet disks

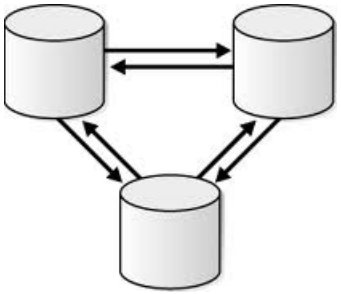
Application

- Object storage
- Shared space
- Virtual Machine



Distribution

- CDN (Multimedia)
- Data replication
- Disaster Recovery

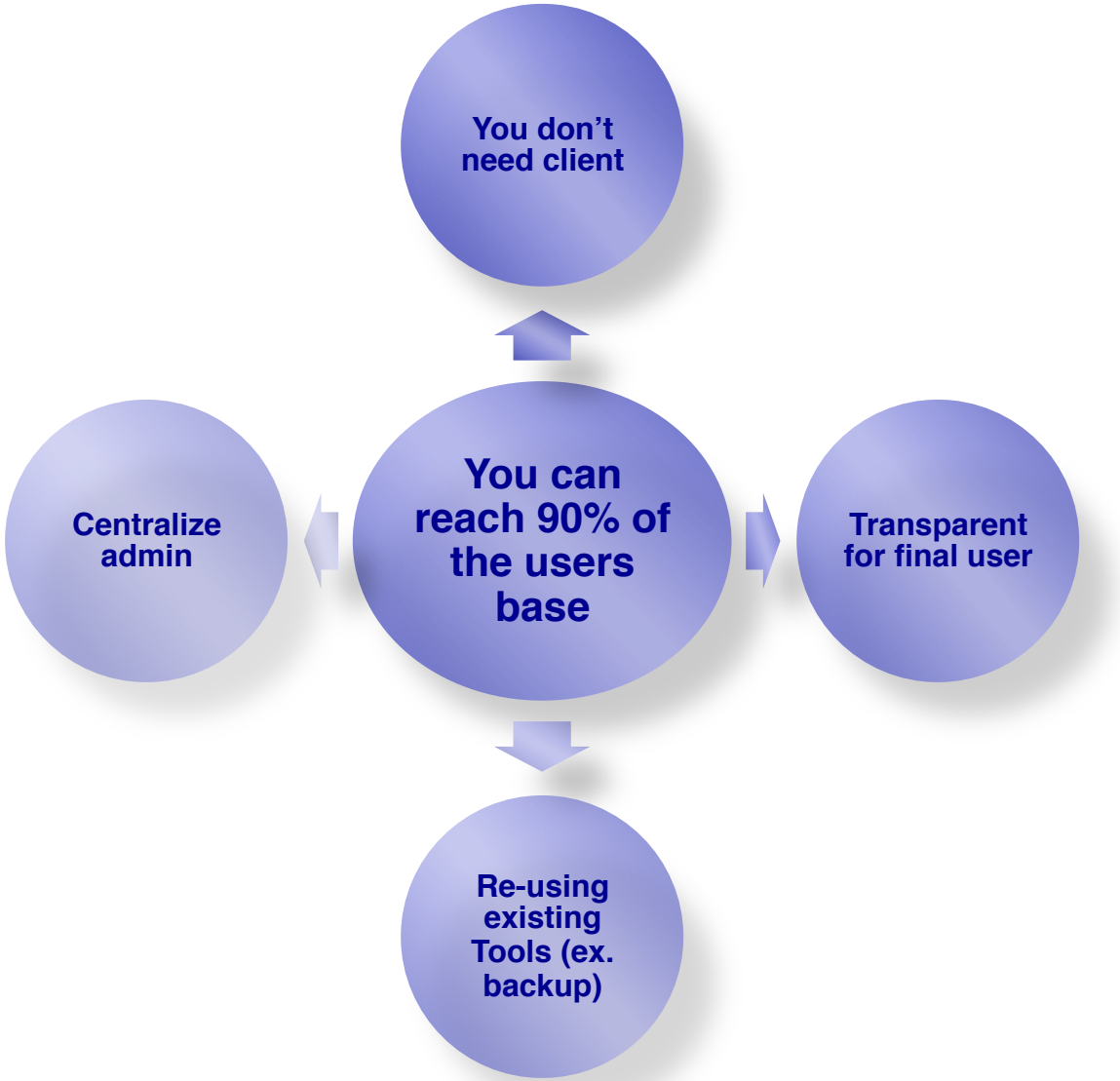


Is everything ok ?

Skip Forward button

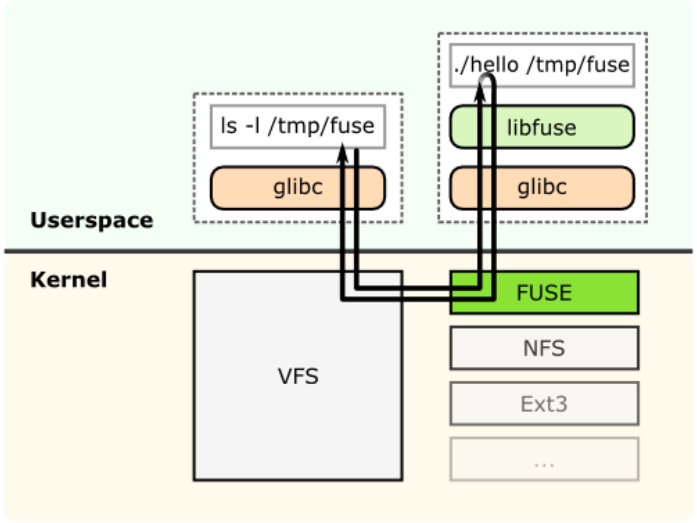

samba

Why samba ?


You don't need client
Centralize admin
You can reach 90% of the users base
Transparent for final user
Re-using existing Tools (ex. backup)

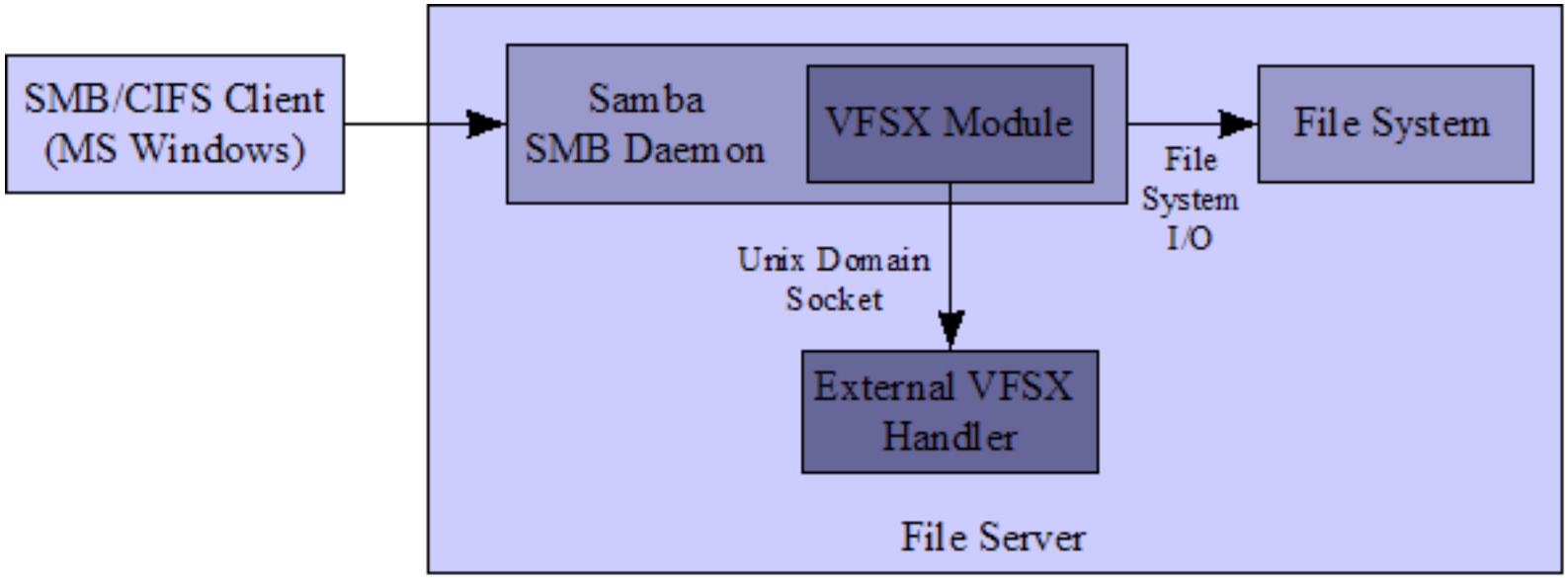
Solution I (simple)

Element	Configuration
Interface	Fuse / Fuse Python
Auth	Server Trust
ACL	Custom Property
Space	One Bucket per Share
Locks	Handle by Samba



Solution II (Experimental)

VFSX is a transparent Samba Virtual File System (VFS) module which forwards operations to a process on the same machine for handing outside of the Samba daemon process…



Solution II (Experimental)

VFS Samba Module

1 Intercept

```
static int vfsx_mkdir(vfs_handle_struct *handle, const char *path, mode_t mode)
{
        int result = -1;
        int count;
        char buf[VFSX_MSG_OUT_SIZE];

        count = snprintf(buf, VFSX_MSG_OUT_SIZE, "mkdir:%s:%s:%s,%d", handle->conn->user, handle->conn->origpath, path, mode);
        if (vfsx_execute(buf, count) == VFSX_SUCCESS_TRANSPARENT) {
                result = SMB_VFS_NEXT_MKDIR(handle, path, mode);
        }
        return result;
}
```

2 Check Socket

```
if (!connected) {
        sd = socket(AF_UNIX, SOCK_STREAM, 0);
        if (sd != -1) {
        strncpy(sa.sun_path, VFSX_SOCKET_FILE,                                  strlen(VFSX_SOCKET_FILE) + 1);
        sa.sun_family = AF_UNIX;
        ret = connect(sd, (struct sockaddr *) &sa, sizeof(sa));
        …
```

3 Write/Read on the socket

```
memset(out, 0, VFSX_MSG_OUT_SIZE);
        strncpy(out, str, strlen(str) + 1);
        ret = write(sd, out, VFSX_MSG_OUT_SIZE);
        if (ret != -1) {
              memset(in, 0, VFSX_MSG_IN_SIZE);
              ret = read(sd, in, VFSX_MSG_IN_SIZE);
              if (ret != -1) {
                        result = atoi(in);
```

Solution II (Experimental)

Smb.conf

```
[myshare]
comment = My share
path = /home/myuser/shared/
valid users = …
….
read only = No
vfs objects = vfsx
```

Samba Conf

Solution II (Experimental)

Python Server

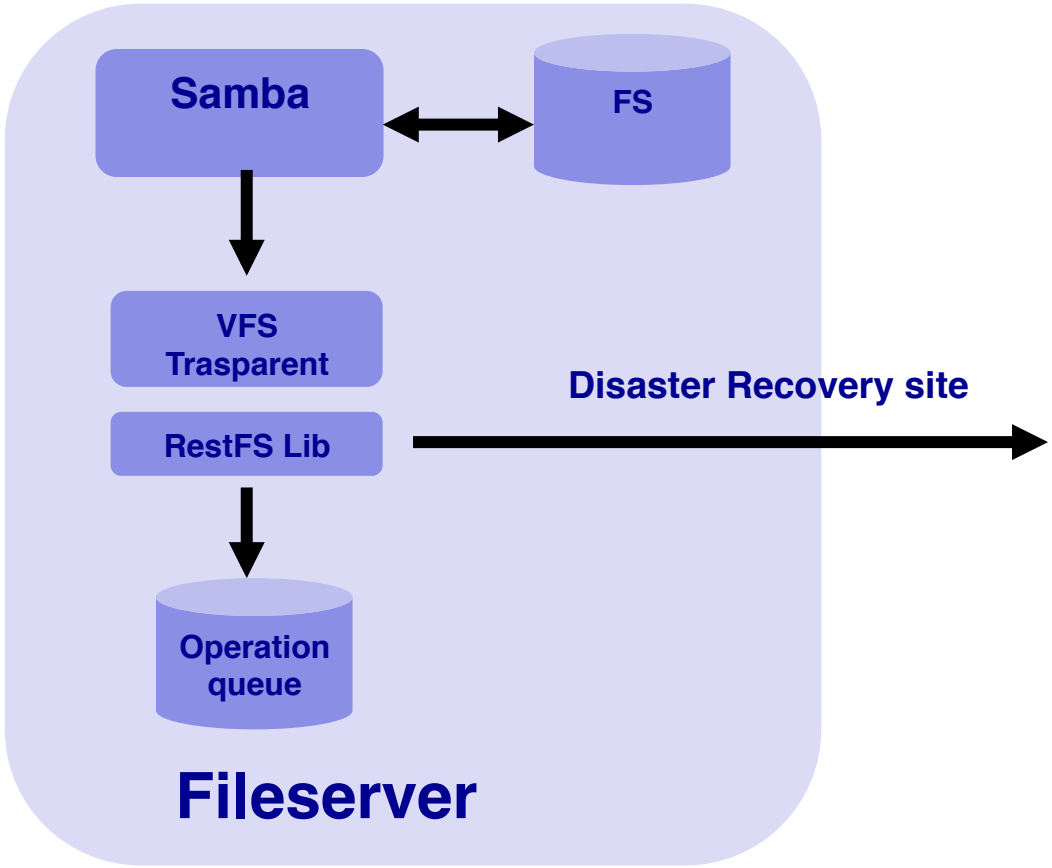
```
…
  while True:
        msg = self.request.recv(512)
        if not msg: break
        log.debug(msg)
        # Handle message-parsing and operation execution error here.
        # Socket communication errors should be propagated.
        try:
              (operation, user, origpath, args) = self.__parseMessage(msg)
              result = self.__callOperation(operation, user, origpath, args)
        except Exception, e:
              result = VFSOperationResult(FAIL_ERROR)
              log.exception(e)
        self.request.send("%d" % result.status)

        # The client probably closed the connection.
        self.request.close()
        log.debug("Close Connection")

def __parseMessage(self, msg):
        parts = msg.split(":")
        (operation, user, origpath) = parts[0:3]
         log.debug("  operation = '%s' user = '%s' origpath = '%s'" %
                                                  (operation, user, origpath))
          args = []
          if len(parts) > 3:
              args = parts[3].split(",")
              log.debug("  args = '%s'" % parts[3])
          return (operation, user, origpath, args)
```

Message Format:

"user:operation:origpath:arg1,arg2,arg3”

Disaster Recovery

Element	Configuration
Interface	VFX
Auth	Samba
ACL	Samba
Cache	Queue mode
Space	One bucket per share



* Under development

Mode 1
Intercept only wr operation

- Attributes, map to RestFS metadata
- Directory, map in RestFS object
- Write file, map to block position
- Return immediately after queue insertion
- Send block to disaster recovery site only on the close operation/flush

Mode 2
Intercept only wr operation

- Attributes, map to RestFS metadata
- Directory, map in RestFS object
- Write file, copy data in block (queue)
- Return immediately after queue insertion

Open Points

- Init Phase (sync)
- Optimization write on close
- Bandwidth Management
- Replication with multiple site
- Sanity Check

* Under development

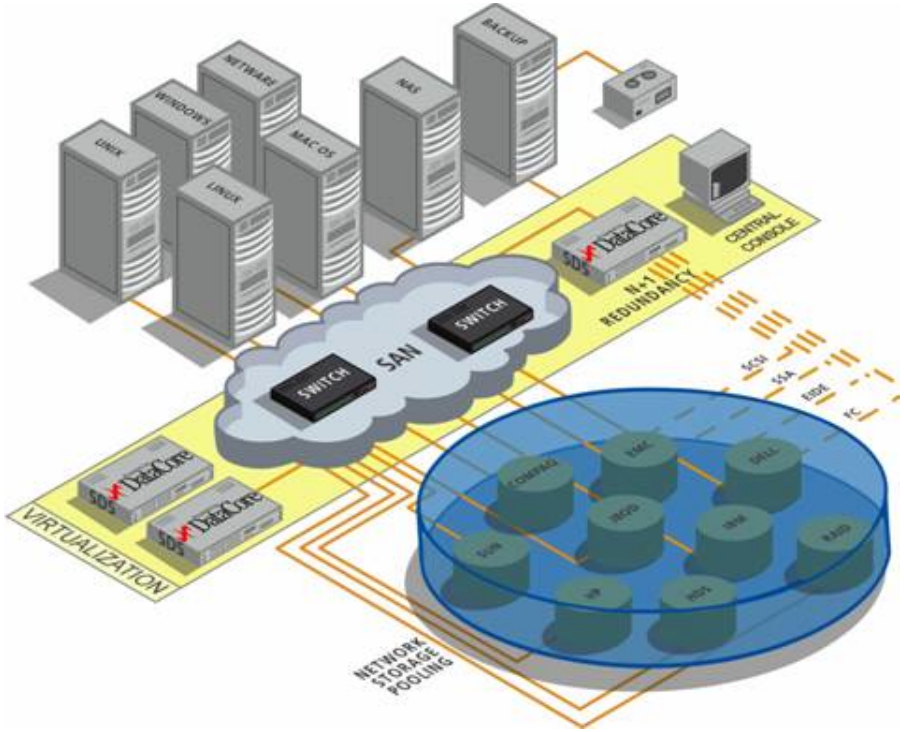
Roadmap

- **0.1 Not Released**
  - Single server on storage (No DHT)
  - FUSE
  - Circular Cache
  - Storage Encryption and compression
  - Federated Authentication
- **0.2 First Public release May (code name WorstFS)**
  - DHT on storage
  - pub/sub
  - ACL
- **Next**

  Clone function, Versioning, Disconnected operation, Logging,Token auth, Locks, Dlocks, Mount Bucket in Bucket, Bucket automate provisioning, Distribution algorithms, Load balancing, samba module, more async i/o, block replication control

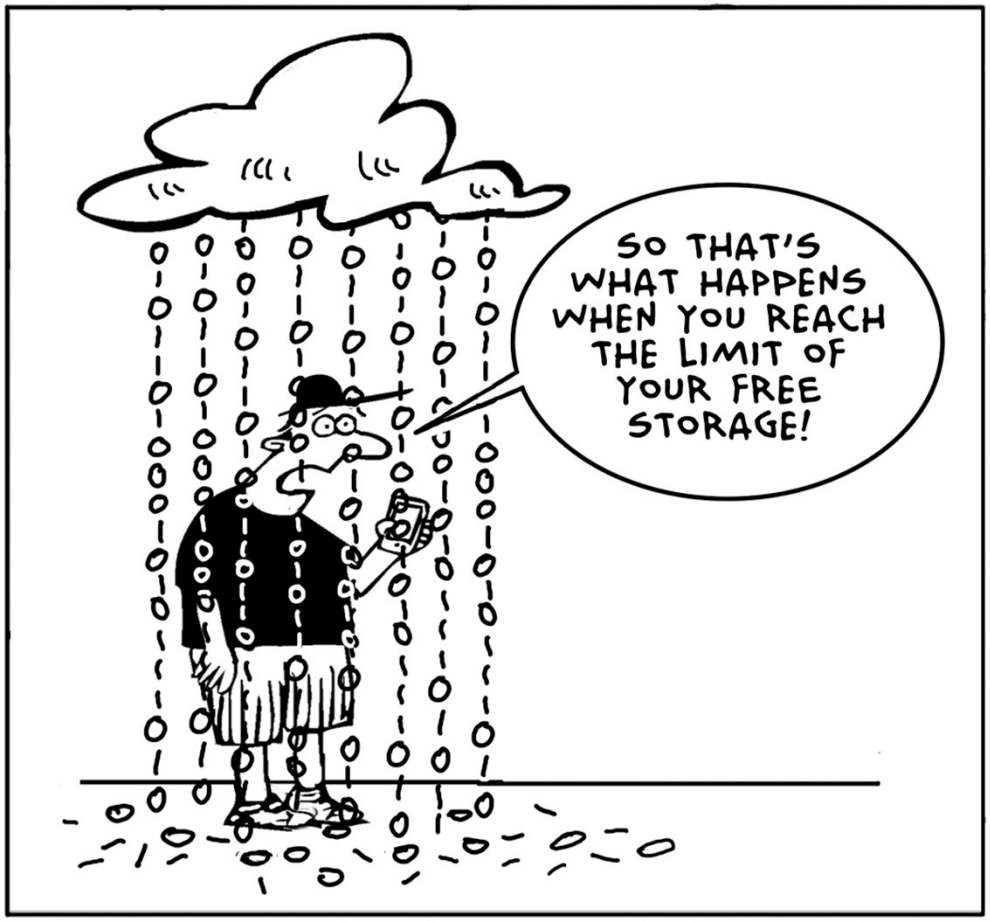
What is the future ?

Beolink.org



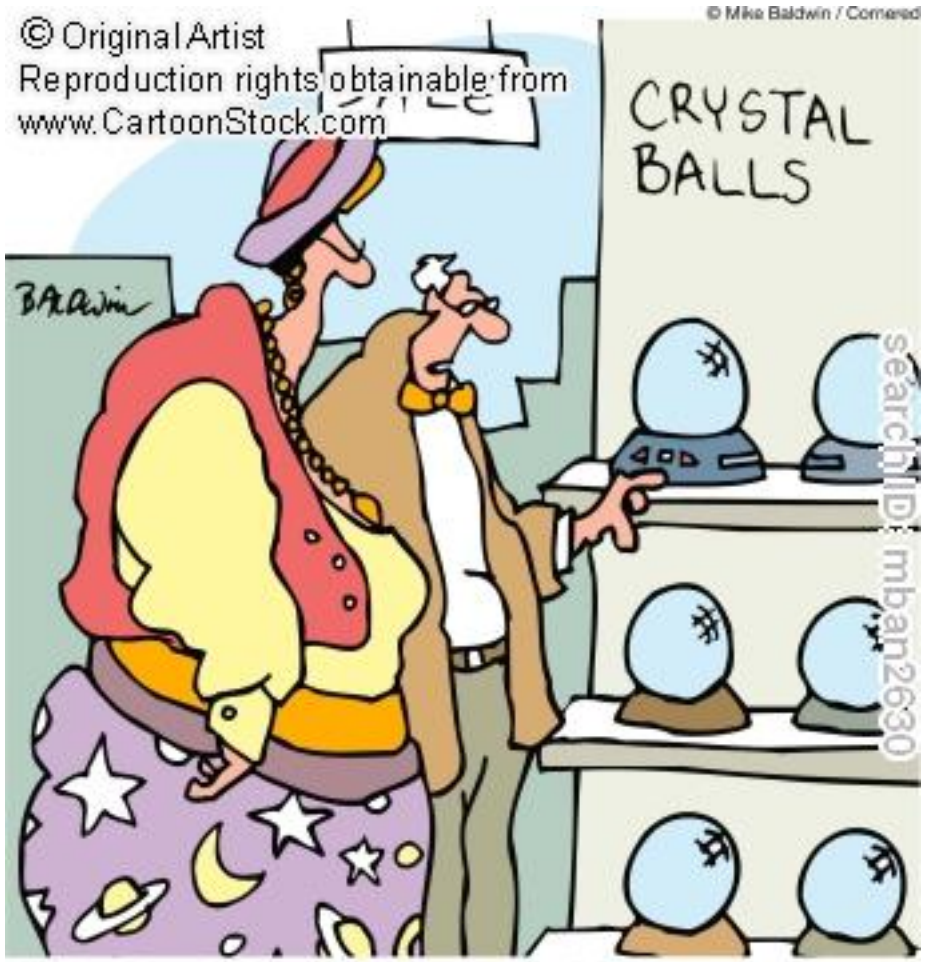
Last question: Space

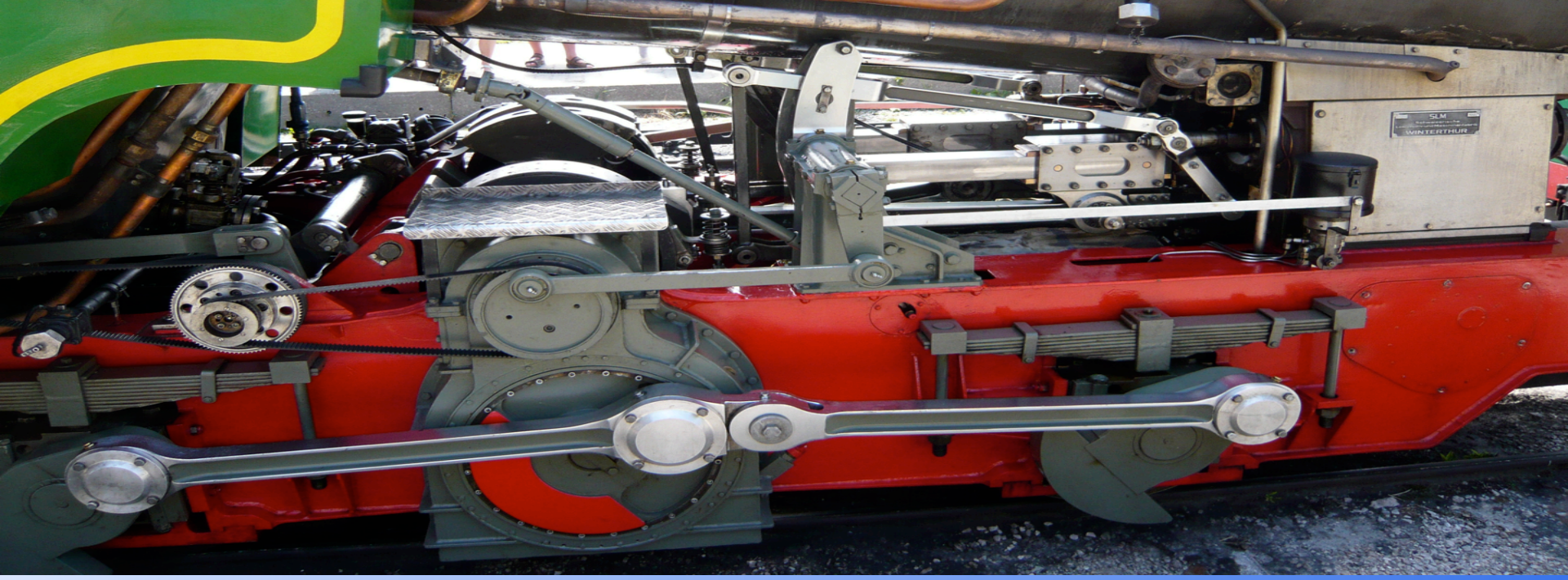
What happens when you have finished the space ?



Do you remember the button of some slide ago ?

Beolink.org



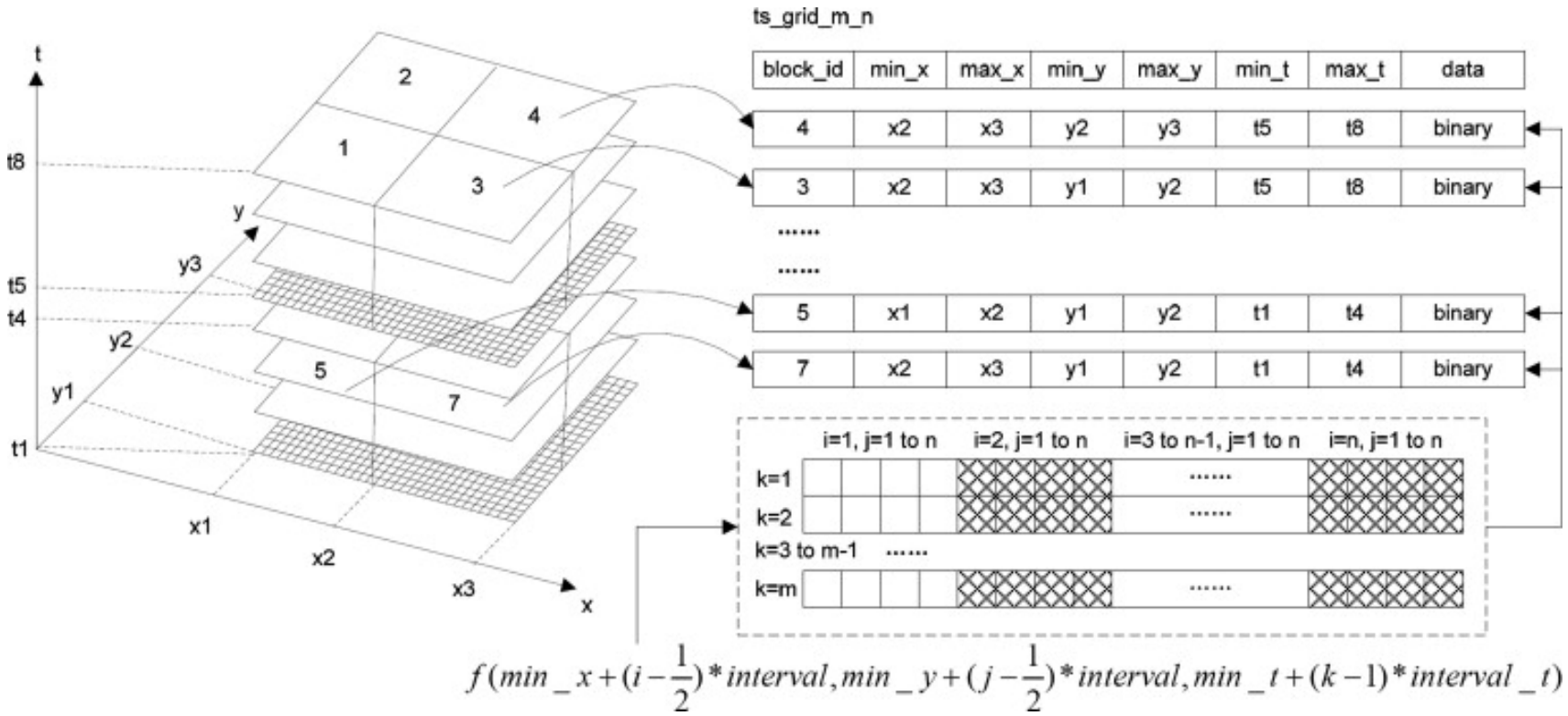


Thank you

http://restfs.beolink.org

manfred.furuholmen@gmail.com

Beolink.org



$$f(min\_x+(i-\frac{1}{2})*interval, min\_y+(j-\frac{1}{2})*interval, min\_t+(k-1)*interval\_t)$$