RestFS

Fabrizio Manfredi Furuholmen
Federico Mosca
Agenda

- Introduction
  - Storage System
  - Storage Evolution

- RestFS
  - Goals
  - Architecture
  - Internals
  - Sub project

- Demo

- Conclusion
“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...”

John H. Terpstra

By the Chairman of SambaXP 2012
Introduction

Store It All, Serve It Worldwide

- Disaster Recovery
- Web 3.0
- PC
- New Devices (TV, tablet, mobile)
- VM

Europython 2012
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
“Late last week the number of objects stored in Amazon S3 reached one trillion (1,000,000,000,000 or 10^12). That's 142 objects for every person on Planet Earth or 3.3 objects for every star in our Galaxy. If you could count one object per second it would take you 31,710 years to count them all.

We knew this day was coming! Lately, we've seen the object count grow by up to 3.5 billion objects in a single day (that's over 40,000 new objects per second)...”

Amazon Web Services Blog, June 12th
Introduction

John’s words + new usage + new services + ...

Hype Cycle and Technology Adoption Lifecycle Plotted together

"The Chasm"
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.
The Perfect Solution

Goals

Uniform Access
- Global name support

Reliability
- No single point of failure

Scalability
- Tera/Peta/... bytes of data

Performance:
- High performance

Security
- Global authentication/authorization

Availability
- Maintenance without disrupting the user’s routines

Standard conformance:
Standard semantics

Elastic
- Bandwidth and capacity on demand

Beolink.org

Europython 2012
“Moving Computation is Cheaper than Moving Data”
Five main areas

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

Europython 2012
RestFS Key Words

- **Domain**: collection of servers
- **Bucket**: virtual container, hosted by one or more server
- **Object**: entity (file, dir, …) contained in a Bucket
Discovery Bucket

Client

DNS Lookup

Bucket name

Domain RL IP list

Bucket name

Cluster 1

Cluster 2

Server list + Load info

Server list priority List

<table>
<thead>
<tr>
<th>Server</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

N server

Europython 2012
Retrieve Data

Server | Priority
-------|--------
IP     | 1      
IP     | 2      
IP     | 3      
..     | ..     

Client

Block Data

Subscribe/publish

Metadata

Object
- Property
- Segment

Client Cache

Beolink.org

Europython 2012
Cache client side

Cache

Temporary
- ServerList
- Tokens
- Pub/Sub List

Persistent
- Metadata cache
- Block cache

Dependencies:
- DNS
- Resource Locator
- Federated Auth
- Callbacks
- RestFS Metadata
- RestFS Block
- RestFS Block
Client Architecture
Is everything ok?
Demo
Object

**Key Value Pair**

Key for everything

- Metadata: BUCKET_NAME.UUID
- Block: BUCKET_NAME.UUID

**HASH**

Each block has an hash to identify the content

**Serial**

Each element has a version which is identified by a serial.

**Property**

The property element is a collection of object information, with this element you can retrieve the most important metadata.

**Object Serialized**

Backends agnostic on information stored

---

**Object**

zebra.c1d2197420bd41ef24fc665f228e2c76e98da247

**Segment-id**

1:zebra.16db0420c9cc29a9d89ff89cd191bd2045e47378  
2:zebra.9bcf720b1d5aa9b78eb1bcdbf3d14c353517986c  
3:zebra.158aa47df63f79fd5bc227d32d52a97e1451828c  
4:zebra.1ee794c0785c7991f986af199a6e6e1fafa4  
5:zebra.c3c662928ac93e206e025a1b08b14ad02e77b29d  

...  

**vers:**1335519328.091779

**Segment-hash**

1:7d565defe000db37ad09925996fb407568466ce0  
2:ccc664efcbe4c8899d9ca68b7089506b7435fc74  
3:66db9e7cdd5b615173c9dc7daf955647db544580  
4:fb8a076b04b550ff9d1b14a2bc655a29dcb341c4  
5:b2c1ace2823620e8735dd0212e5424da976f27bc  

...  

**Property**

segment_size= 512  
block_size = 16k  
content_type =  
md5=ab86d732d11beb65ed0183d6a87b9b0  
max_read'=1000  
storage_class=STANDARD  
compression= none  
...
**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

**Pattern matching**
Clients may subscribe to glob-style patterns in order to receive all the messages sent to channel names matching a given pattern.

**Distributed Cache**
For server side the server share information over distributed cache to reduce the use of backend

**Client Cache**
Pre allocated block with circular cache write-through cache

---

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

<table>
<thead>
<tr>
<th>Use case</th>
<th>Accepted client</th>
<th>Received messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1,000</td>
<td>16,000</td>
</tr>
<tr>
<td>B</td>
<td>10,000</td>
<td>160,000</td>
</tr>
<tr>
<td>C</td>
<td>100,000</td>
<td>1,600,000</td>
</tr>
</tbody>
</table>
WebSocket
is a web technology 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…

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

BSON
short for Binary JSON,
is a binary-encoded serialization of JSON-like documents. Like JSON, BSON supports the embedding of documents and arrays within other documents and arrays.
BSON can be compared to binary interchange formats

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

HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: HSmrc0sMIYUkAGmm5OPpG2HaGWk=
Sec-WebSocket-Protocol: chat

--> { "method": "echo", "params": ["Hello JSON-RPC"], "id": 1 }
<-- { "result": "Hello JSON-RPC", "error": null, "id": 1 }

{"hello": "world"}
→
\x16\x00\x00\x00\x02hello\x00 \x06\x00\x00\x00\x00world\x00\x00\x00
Security Channel
Communication over S3 protocol is based on SSL
Communication over RestFS RPC is based on WSS

Identification
Internal or through identity provider like Google, Facebook..

Token
For each identity is possible to assign more devices with different token. This operation permits to exclude a stolen device or enable a time period based one.

Authorization
The security control is based on extended ACL (nfs4)

Encryption
The user can encrypt the data with personal password, share information through gnuPG framework (under development)
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
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 neighbor.
Open points: Space

What happens when you have finished the space?

SO THAT'S WHAT HAPPENS WHEN YOU REACH THE LIMIT OF YOUR FREE STORAGE!
### What we are using

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

* 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
Subproject: Disaster Recovery

<table>
<thead>
<tr>
<th>Element</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>VFX</td>
</tr>
<tr>
<td>Auth</td>
<td>Samba</td>
</tr>
<tr>
<td>ACL</td>
<td>Samba</td>
</tr>
<tr>
<td>Cache</td>
<td>Queue mode</td>
</tr>
<tr>
<td>Space</td>
<td>One bucket per share</td>
</tr>
</tbody>
</table>

*S Under development*
## Advantages

<table>
<thead>
<tr>
<th>High reliability</th>
<th>Distributed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Decentralized</td>
</tr>
<tr>
<td></td>
<td>Data replication</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Nearly unlimited scalability</th>
<th>Horizontal scalability</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Multi tier scalability</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cost-efficient</th>
<th>Cheap HW</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Optimized resource usage</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Flexible</th>
<th>User Property</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Extended values and info</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enhanced security</th>
<th>Extended ACL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OAUTH / Federation</td>
</tr>
<tr>
<td></td>
<td>Encryption</td>
</tr>
<tr>
<td></td>
<td>Token for single device</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Simple to Extend</th>
<th>Plugin</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bricks</td>
</tr>
</tbody>
</table>
Roadmap

- **0.1 Released Today**
  - Single server on storage (No DHT)
  - S3 Interface
  - Federated Authentication

- **0.2 Release September (codename WorstFS)**
  - DHT on storage
  - Storage Encryption and compression
  - FUSE

- **0.3 Release TBD (codename WorstFS++)**
  - Deduplication
  - pub/sub
  - ACL
  - ...

- **Next**

  **Clone function, Versioning, Disconnected operation, Logging, Locks, Dlocks, Mount Bucket in Bucket, Bucket automate provisioning, Distribution algorithms, Load balancing, samba module, more async i/o, block replication control, negative cache, index, user defined index**
Thanks

Thanks to Zeropiù for the support
Code, ideas, testing, insults ... everything
Froscon 2012

25./26. August 2012
Sankt Augustin
Cologne DE
www.froscon.de
Thank you

http://restfs.beolink.org

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