Mucura

Prototyping a cloud-based file storage system

Fabio Hernandez

fabio@in2p3.fr

on behalf of the Mucura team

Lyon, March 19th 2012

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE
MINISTÈRE DES AFFAIRES ÉTRANGÈRES ET EUROPÉENNES

中國科學院高能物理研究所
Institute of High Energy Physics
Chinese Academy of Sciences

CCIN2P3

Preamble

- Goal

this presentation is a short overview of the activities we have been conducting regarding the prototyping of a cloud-based storage system

fabio hernandez
fabio@in2p3.fr
CAS/IHEP Computing Centre
CNRS/IN2P3 Computing Centre

Mucura

Big clay container (sort of amphora) used for storing beverages, water, cereals and also for funeral rites by natives of pre-colombian ethnic groups in Colombia and other American countries

Photo: Wikipedia

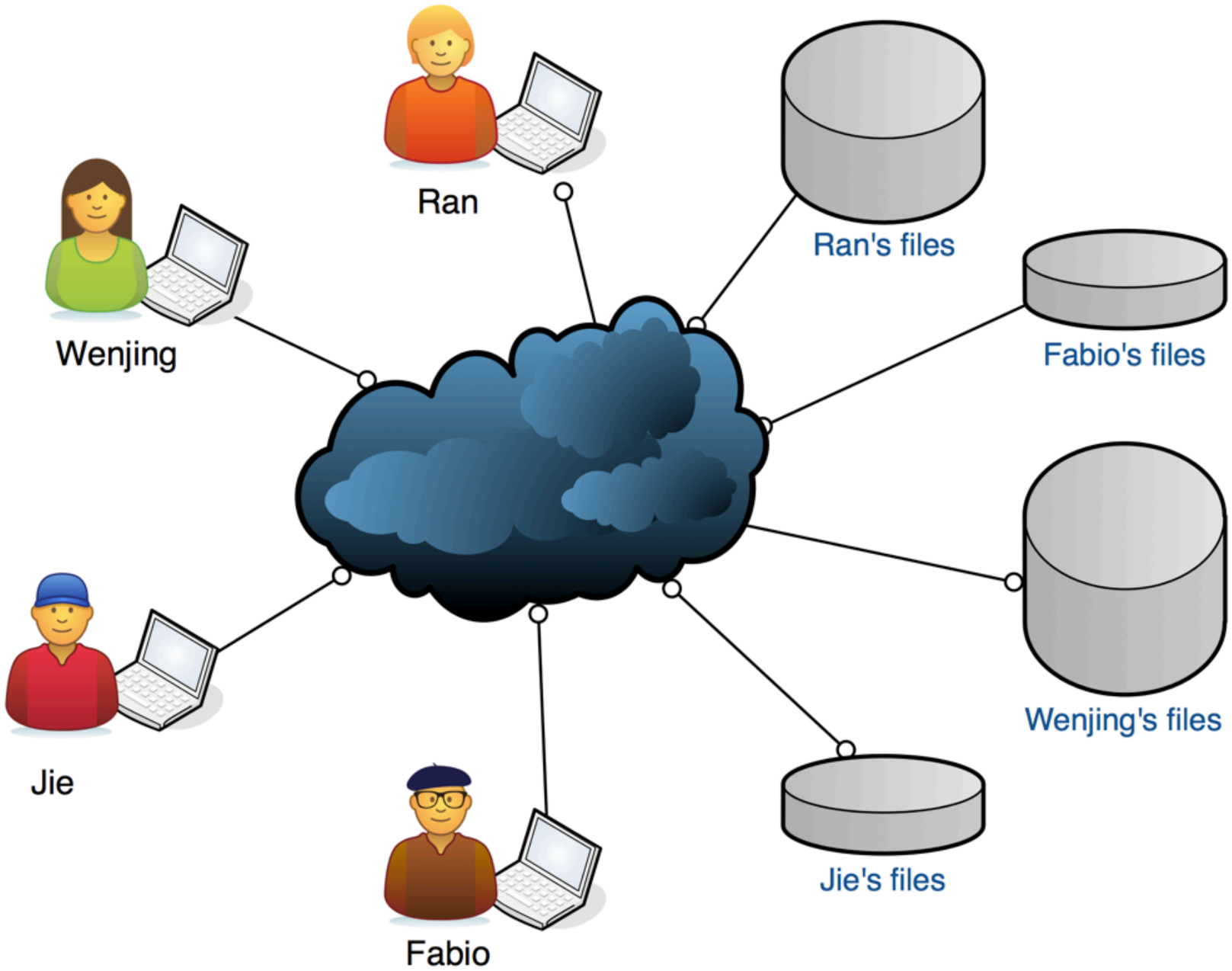
Adapted from the mucura entry in Wikipedia Español

Cloud-based file storage

- Goal: to provide individual researchers a **convenient** service for remotely storing their files

*Think of it as a **personal on-line file repository**, accessible both from the user's personal computing (interactive usage) and from the grid jobs (batch usage)*

Vision



Your files are physically stored on remote servers accessible seamless through the network.

You interact with your remote files as you usually do with your local files, using your desktop's metaphors (file explorer, drag & drop, etc.).

You are free to organize your own storage space.

The system provides you significantly more storage capacity than is locally available in your personal computer.

You can share your files with selected users of the system.

Vision (cont.)

- Who this service is intended for?

  individual *scientists, initially those of the high energy physics community*

- Why would you use this service?

  because it provides a ***convenient*** *way for storing your individual files, for instance, the logs of your (grid) jobs, datasets you are analyzing, etc.*

  because you may want to ***share*** *some files with some colleagues sitting at the next door or across the world*

  because you would have total ***flexibility*** *to manage your own space according to your needs and working methods*

  because it may eventually provide you more storage ***capacity*** *than you have internally on your personal computer*

Vision (cont.)

- Who would operate the service?

computing centres of the high energy physics community: good national and international connectivity, 24x7 service, expertise in running IT services, trustable, reliable, ...

files would be physically located in disk servers managed and operated by one or more of those centres

Use-case & scale

- Use-case profile

  retrieval of files more frequent than storage

  repository used as a high-capacity highly-available archive system: ***not intended to directly serve I/O-intensive applications***

  in addition to access files from the desktop, grid jobs and jobs submitted to a particular site can also interact with the repository for storing/retrieving files

- Scale

  initially to provide each user an individual storage capacity of 1TB to 2TB

  thousands of directories, tens of thousands of files, per user

  file size in the region of 1B to 5GB, but most of the files expected in the area of a few hundreds MB

Features (cont.)

- Reduced set of basic operations
  - ***list, store, retrieve, delete files***
  - ***create, delete directories***

  although the system does not expose complete POSIX semantics, emulation will be possible for compatibility

  compliant to a well-documented standard API

- Service operation

  *the system must be **operator-friendly**: scalable, reduced amount of skilled manpower required to operate it, resilient to hardware failures and to network partitioning (both inter- and intra-centre)*

  *cost-effective: built with **commodity hardware***

  no need to explicitly back the files up

Features (cont.)

- Confidentiality

  use secure channels for transporting the data between the user's personal computer and the remote disk servers

  should the files be stored encrypted?

- Performance

  no stringent I/O performance requirements

  the store is NOT intended to be used as a high-performance file system to serve the data directly to the applications, but rather as resilient and scalable repository for storing files online

  however, some level of interactivity is highly desirable

Building blocks

- Back-end

  *files contents and meta-data stored on **key-value data stores***

  possibility to use a networked file-system for storing file contents

  *exposes an **Amazon S3-compatible API**: HTTPS-based*

- Client-side

  existing Amazon S3-compatible applications and utilities: Cyberduck, Expandrive, s3fs, jetS3t, PersistentFS, s3cmd, ...

  proprietary FUSE-based filesystem layer

Key-value store

- Data store exposing a extremely simplified interface, basically composed of 3 operations

```
bool put(string key, byte[] value);

byte[] get(string key);

bool delete(string key);
```

- Data structure supported by several programming languages, also known as associative arrays

- The semantics of ‘`value`’ are client-defined: no predefined schema

  although the store may provide support for structured format of values, referred to as ‘documents’ (typically JSON)

Key-value store (cont.)

- Several (too many?) implementations, focusing on different features

  in-memory vs. persistent

  single-node vs. distributed

  binary objects vs. document/graph/column-oriented

  latency

  …

Key-value store (cont.)

- Heavily used as a storage back-end for (large scale) web applications

  user profiles, user sessions, shopping carts, blog entries and their associated threaded discussions, streams of data, …

  Amazon, Google, Twitter, Baidu , Facebook, LinkedIn, Yahoo, ...

- Benefits

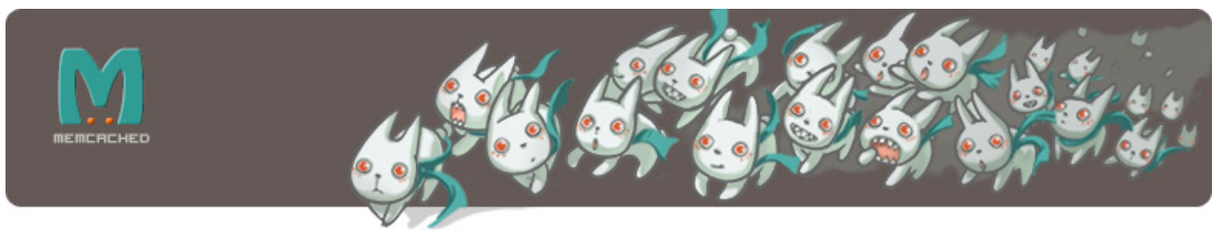
  very fast, scalable, flexible schema, flexible datatypes, programmer-friendliness, …

- Constraints

  require you model your problem domain in terms of key-value mappings

  no transactions, no joins, ...

Key-value stores (cont.)



Distributed hash tables

- For this project, we are particularly interested in distributed key-value stores that are

  ***decentralized**: the nodes collectively form the system, no central coordination*

  ***scalable**: the system works efficiently with hundreds of nodes and thousands of users*

  ***fault tolerant**: to some extent, the system can continue working with nodes failing, leaving and joining the system*

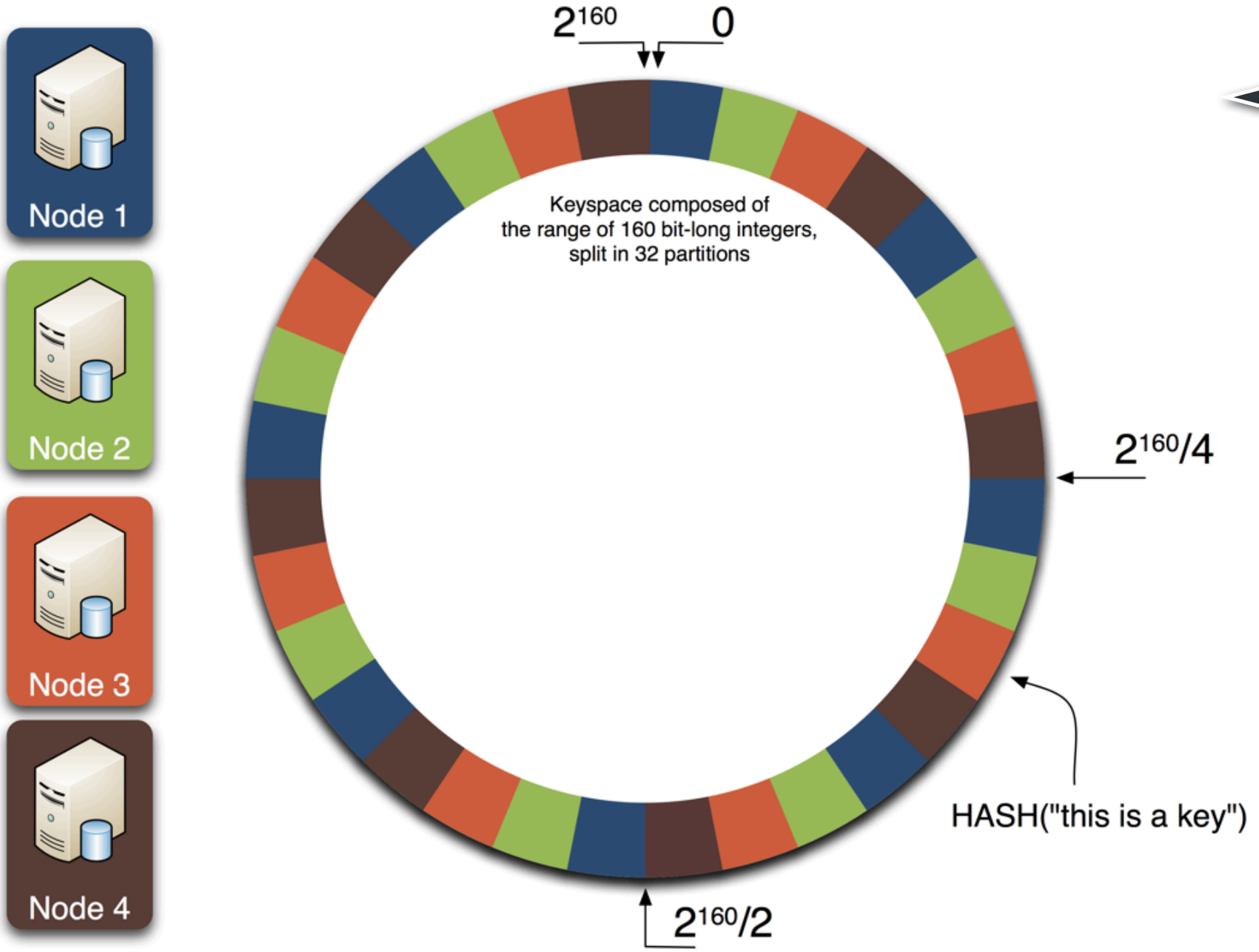
- Nodes are considered trustable

  all the machines are managed by the operator of the service, typically a computing centre

- At least a basic level of authorization is required

  Alice should not be able to remove (nor even access) files owned by Bob, unless Bob explicitly authorizes her to do so

Dynamo: ring



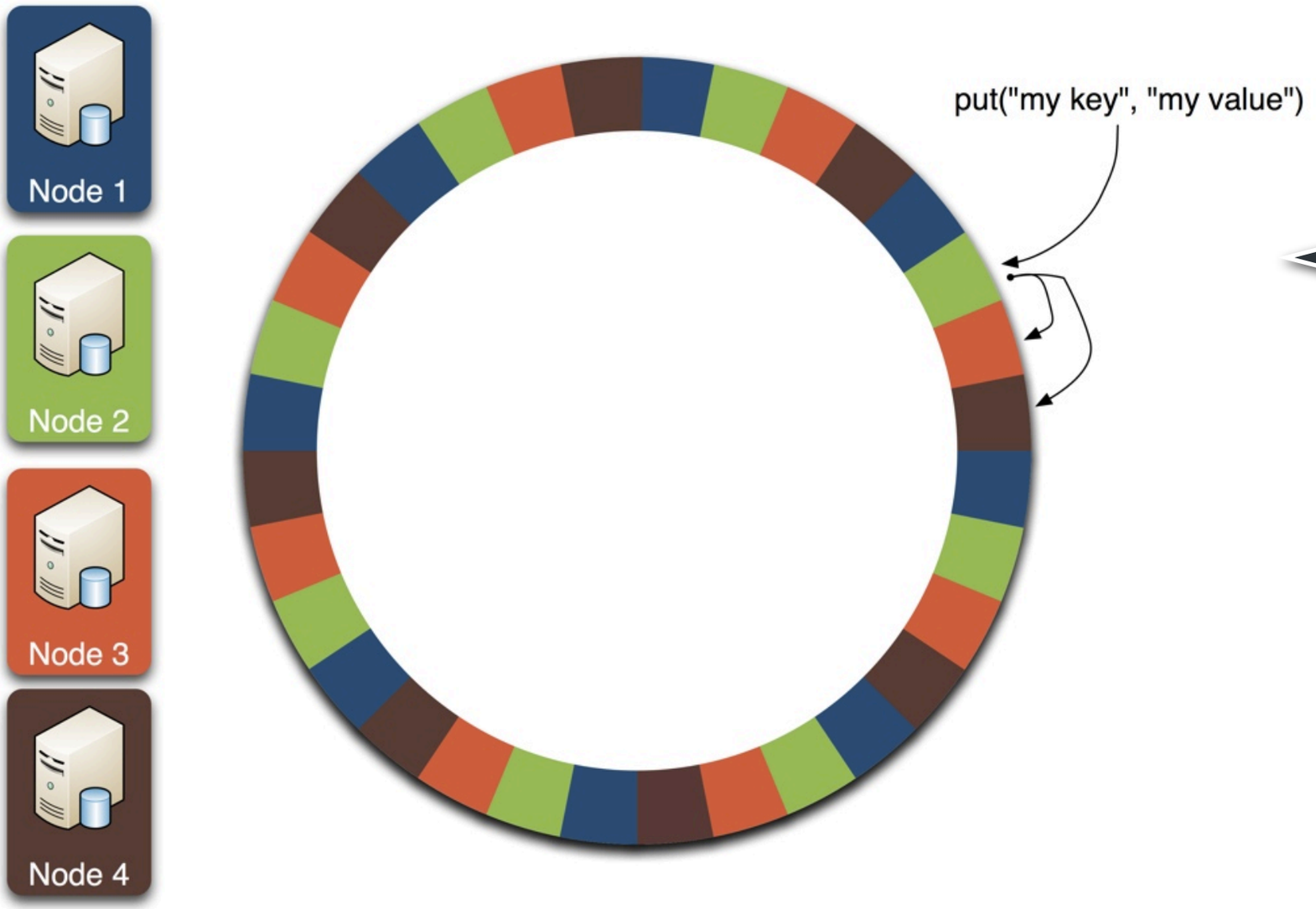
File servers are organized in a logical ring.

Each server is responsible for storing the values associated to one or more partitions in the key space.

Each node in the ring knows which peer node is responsible for storing & retrieving the value associated to a key.

Adapted from : basho.com

Dynamo: replication



The node responsible for handling operations associated to the key is selected as the coordinator of the request.

After storing the `[key,value]` pair locally, it replicates it to its available neighbors (N=3 in this case).

The number of replicas to keep (N) and the minimum number of replicas to read (R) or write (W) to consider a request successful is configurable.

Dynamo: characteristics

- Availability

  data is automatically and asynchronously replicated to multiple file servers within the same data centre or across data centres

- Symmetry & decentralization

  every file server plays an identical role: no single point of failure, no network bottleneck

- Manageability

  failed file servers can be replaced with no downtime

  additional servers can be added without service interruption

- Elasticity

  read and write throughput both increase linearly as more file servers are added to the system

- Heterogeneity

  system exploits the heterogeneity of the infrastructure (network throughput and latencies, file server capacities, etc.). This allows for adding new more powerful file servers without having to upgrade them at once

- Eventual consistency

  replicated copies of the same data may not all be consistent at any given moment in time, but the system is designed for them to be eventually consistent

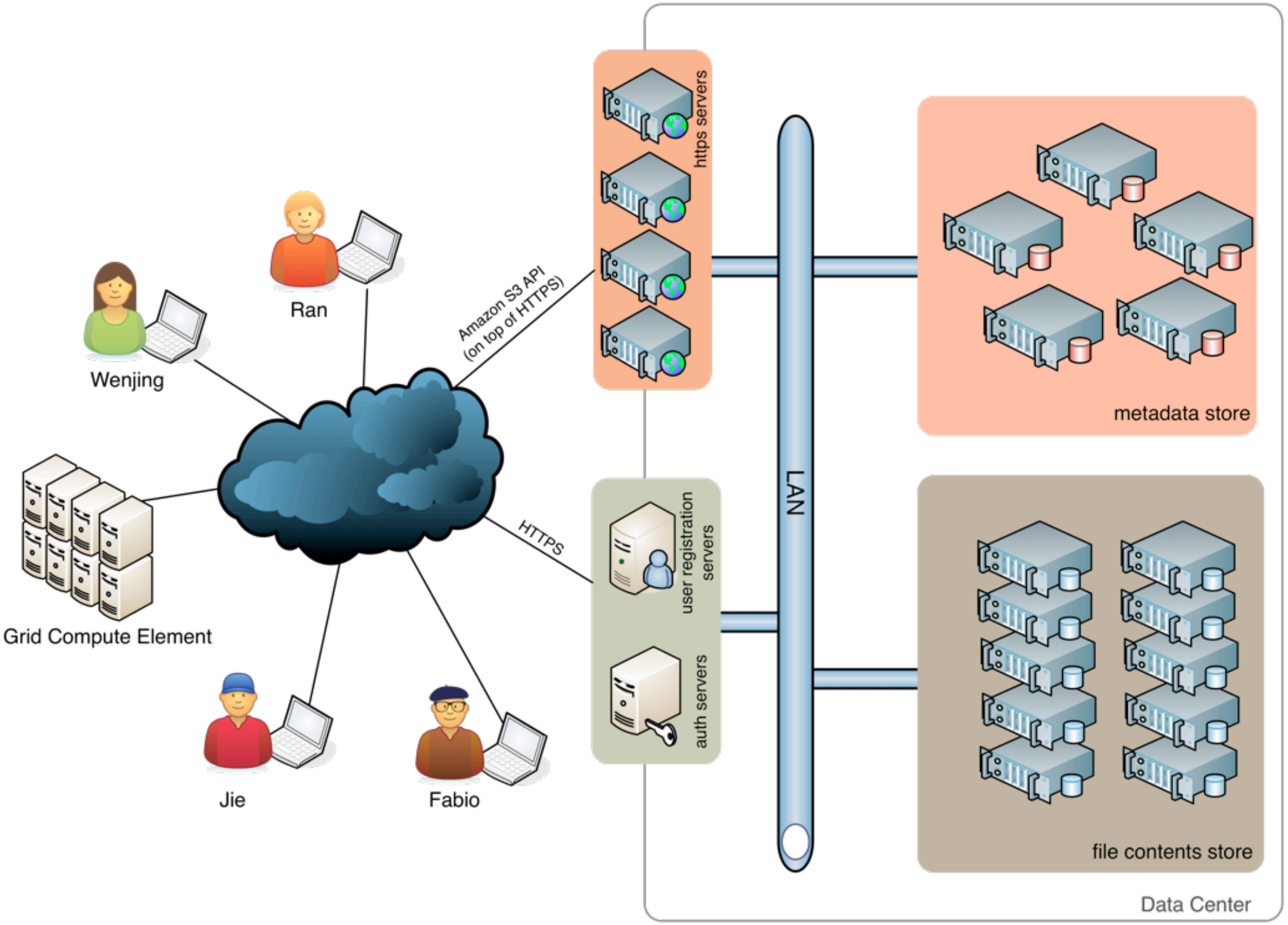
  conflict resolution may be implemented at the store level or at the application level, e.g. "last write wins"

- Data model

  distributed key-value store

  partitioning of data among the file servers is done using consistent hashing

Overall architecture



Demo

Current status

- Meta-data storage

  developed 2 prototypes using Redis (in-memory key-value store) and Cassandra

  written in Python

  measured performance: 2.000+ meta-data requests/second (Redis)

- File contents storage

  developed module to store the file contents on top of a networked file system (Gluster, Lustre, GPFS, ext3, xfs, etc.)

  currently testing OpenStack's Swift as the support for file contents (no quantified results yet)

- Server

  (incomplete) implementation of an HTTPS server compatible with Amazon S3 API, written in Python

- Client

  GUI application Cyberduck works unmodified. Tested also Expandrive and Transmit, but they need modification

  boto (open source Python API speaking S3) works unmodified

- Authorization

  possibility to use Amazon-S3 style credentials (mainly for interactive usage) or temporary credentials generated based on X509 certificates (used by the grid, mainly for batch usage)

  proof of concept of an authorization server delivering temporary credentials for Mucura, based on grid proxies

Perspectives

- Performs tests with OpenStack's Swift back-end

  performance, scalability, manageability

- Perform end-to-end scalability tests

  including when client and server are very distant

- Make sure software compliant to to open source standards

  distribution channels, documentation, web presence, …

- Start alpha tests with early adopters

Questions & Comments

fabio hernandez
fabio@in2p3.fr
CAS/IHEP Computing Centre
CNRS/IN2P3 Computing Centre