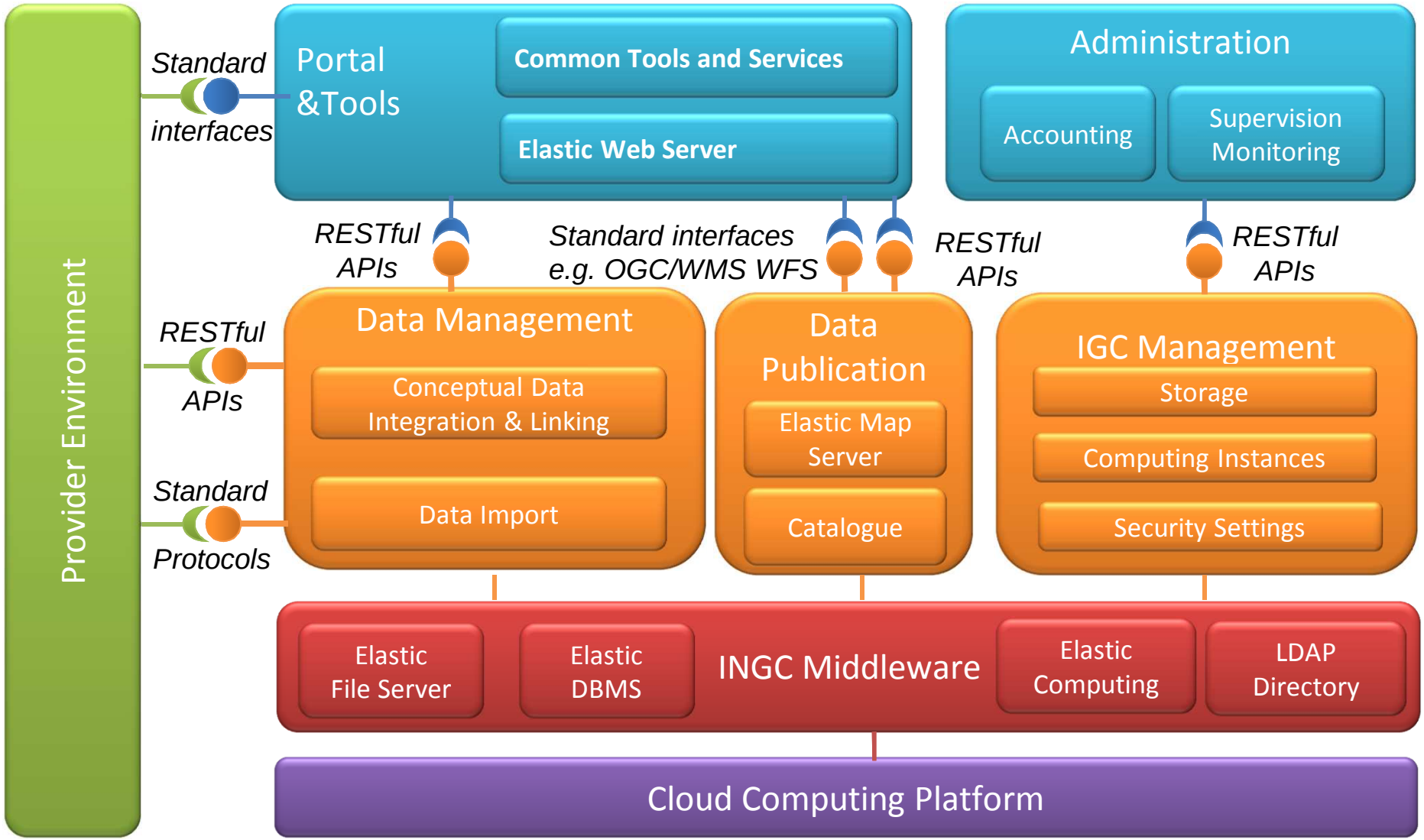
CNR

Insights into the Technical Aspects of the InGeoCloudS platform

Experts Workshop

Nov. 21st, 2013

Overview of the Architecture



Elastic Computing

- Gateway to the Cloud Provider
  - Transparent access
  - Improved portability
- Service exposed (as library functions, not in REST):
  - Virtual instances Management
    - Run a new instance, Stop an instance, attach a storage device, Elastic IP, automatically mount the distributed file system.
  - Auto-Scaling Layer Managment
    - Manage an elastic pool of servers, including reliability and load balancing

Elastic Computing

- ***Dynamic pool of servers*** abstraction
  - A.k.a. scalable group
  - It encompasses aset of servers offering the same service.
- The ***number of servers in the pool can vary*** automatically over time
  - ***Elasticity***
  - Monitoring of aggregated stats (e.g., CPU load) and trigger-based mechanisms
  - A *reference* virtual image is available through some permanent storage facility
- ***Load balancer*** distributes requests among the servers
- Also used to achieve ***reliability***

- ***This is the building block for scalable services***

Geo Processing

- It lets a provider run on demand tasks into the platform
- The configuration of the environment and the installation of the required software is left to the provider
- The managing of the instances is made using the APIs
  - Start, stop, restart, terminate, status actions available
- The cost of those instances are imputed only to the provider

Geo Processing API

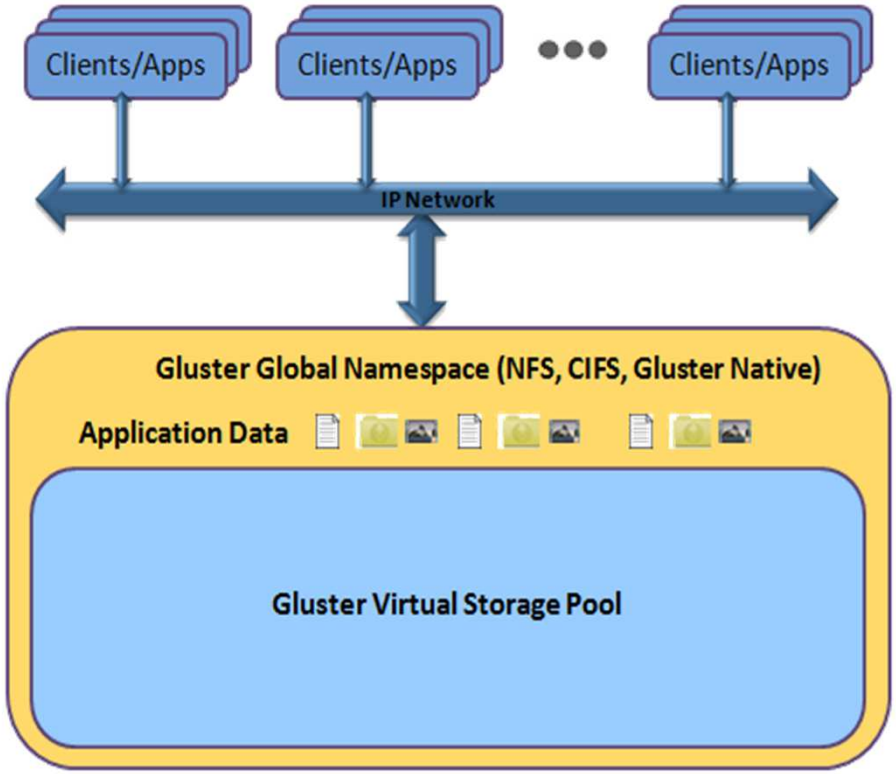
- RESTful services for GeoProcessing management:
  - Start an instance (Provider)
  - Stop an instance (Provider)
  - Terminate an instance (Provider)
  - Restart a stopped instance (Provider)
  - Retrieve the status of an instance (Provider)

ElasticFS

- Elastic File System:
  - Scalable storage capacity
  - Scalable computational needs
  - High availability and performance
  - Automatic mount of the volume on the instances at boot time

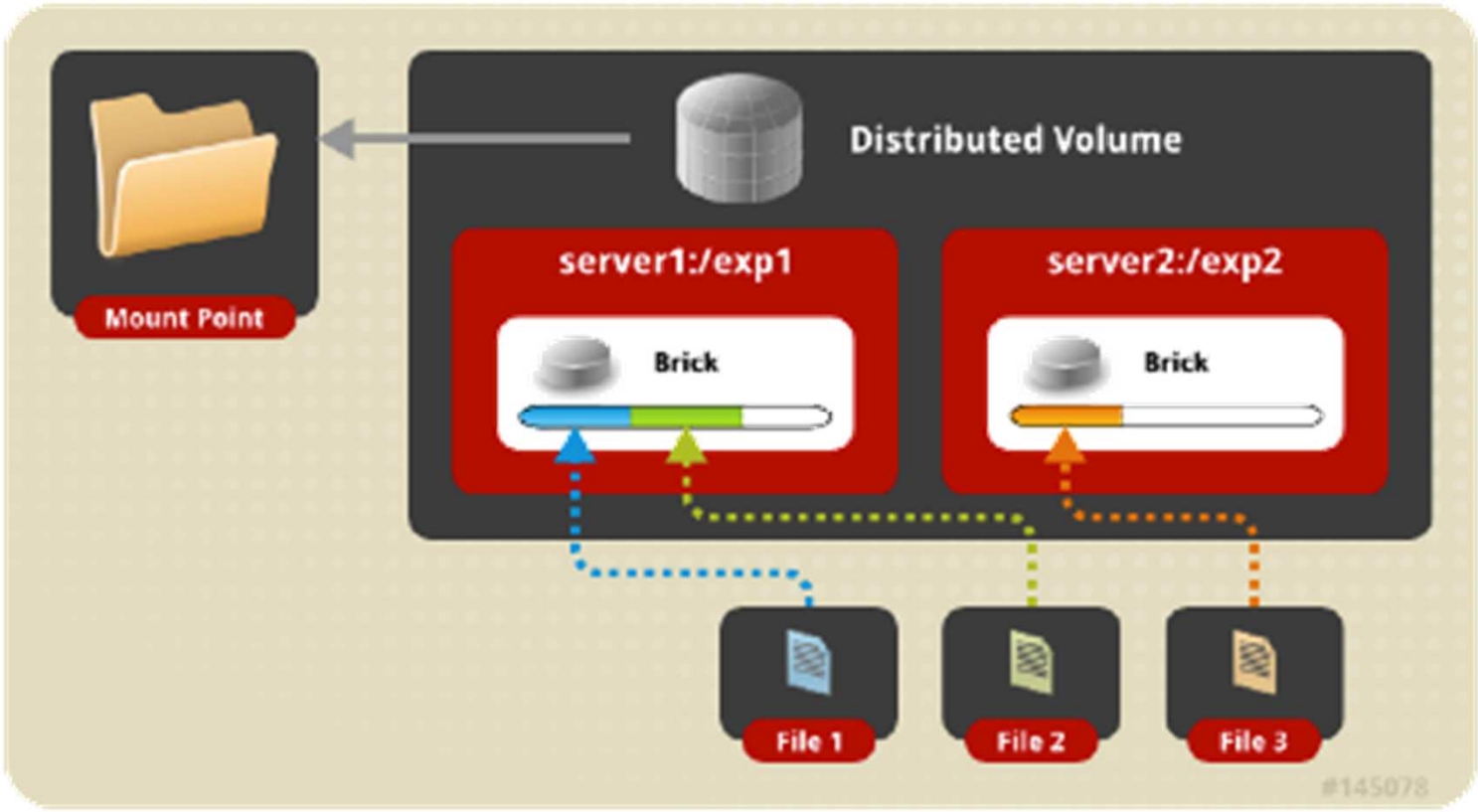
- Relies on top of GlusterFS
  - An open source solution
  - Robust, mature and stable
  - Distributed and decentralized (hash algorithm)
  - Support Replication, Distribution, Striping and a mix of them
  - Export the “volume” as a single global namespace

ElasticFS

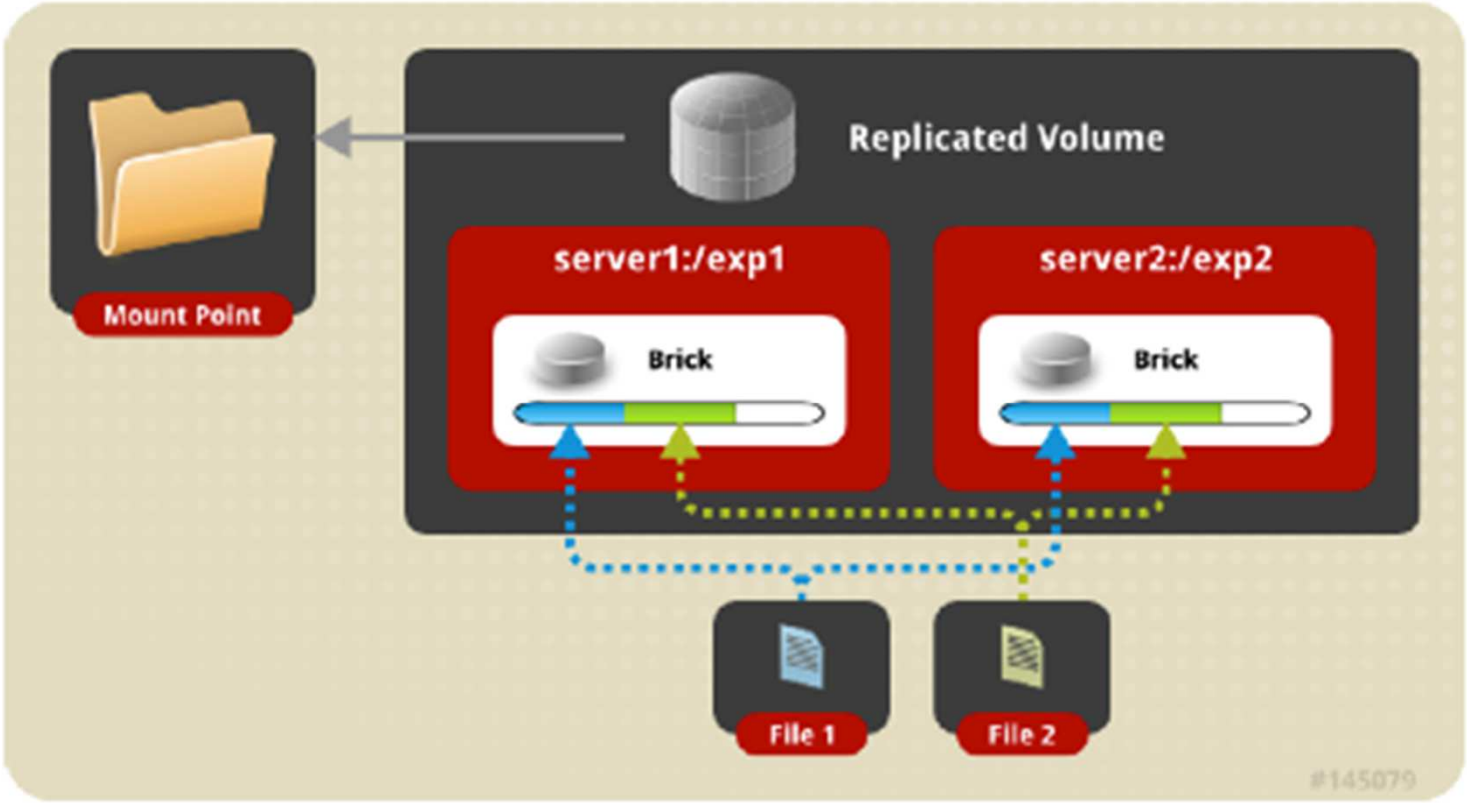


- No single point of failure
  - No file metadata server
- Scalable
  - Add dynamically server
  - Add dynamically volume
- Can use standard protocols
  - E.g. NFS
- Linux authorization mechanism
- Includes: read ahead, write behind, async I/O, scheduling, caching
- Sponsored by RedHat

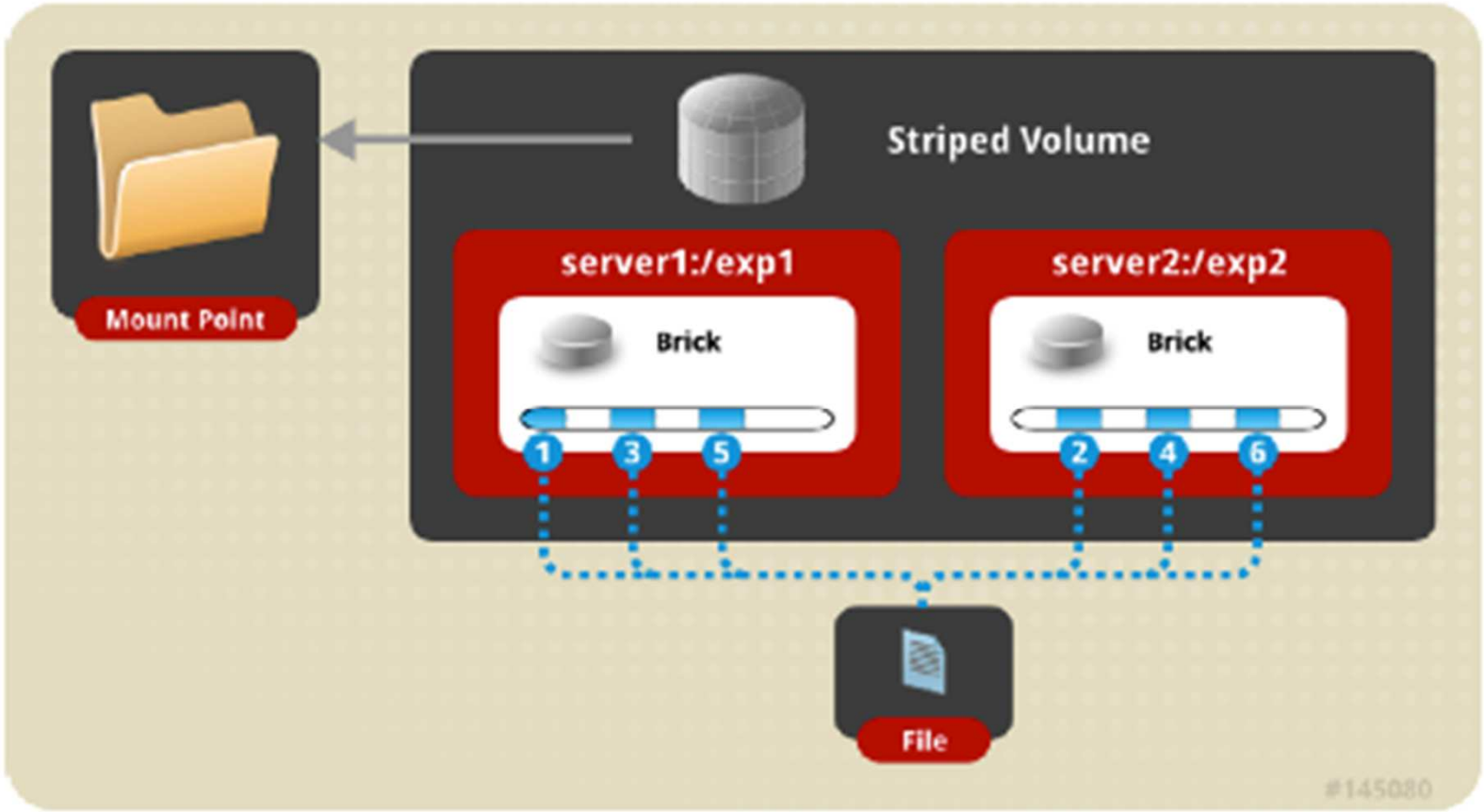
GlusterFS configurations



GlusterFS configurations



GlusterFS configurations



ElasticFS

- Solution adopted in the cloud:
  - Number of EC2 instances according to the load needs
  - Number of EBS volumes according to the storage needs
  - Replication factor of 2
  - Automatic mount of the volume on the instances at boot time

- Automatic failover of a gluster node
  - By replacing it and “rebalancing” the volume

ElasticFS Scalability

ElasticFS API

- RESTful services for FS management:
  - Start the ElasticFS (Administrator)
  - Stop the ElasticFS (Administrator)

- RESTful services for FS monitoring:
  - Check the ElasticFS health (Administrator)
  - Get detailed status on the ElasticFS (Administrator)

- RESTful services for FS maintenance:
  - Backup the entire file system (Administrator)
  - Restore the entire file system (Administrator)

ElasticFS API

- RESTful services for provider management:
  - Create/delete a dedicated workspace for a provider (Administrator)
  - Get information on a provider's workspace (Administrator, Data Provider)

- RESTful services for FS access:
  - Get connection information on a provider's workspace (Administrator, Data provider)

Management

- Coordination of multiple components
  - E.g., start/stop of the platform
- Provide user account creation
  - Create account in LDAP
  - Create private workspace in FS
  - Create private database inside DBMS
- Provide backup/restore facilities
  - By using the backup/restore functionalities provided by the underlying module (FS, DB, LDAP)

Management API

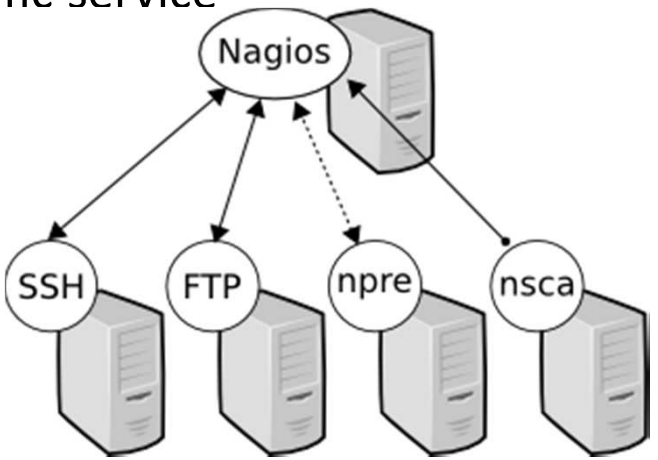
- RESTful services for platform management:
  - Start all the INGC services (Administrator)
  - Stop all the INGC services (Administrator)

- RESTful services for platform management:
  - Backup the entire platform (FS, DB, Accounts, etc, Administrator)
  - Restore the entire platform (FS, DB, Accounts, etc, Administrator)

- RESTful services for user management:
  - Create/modify a Data Provider account

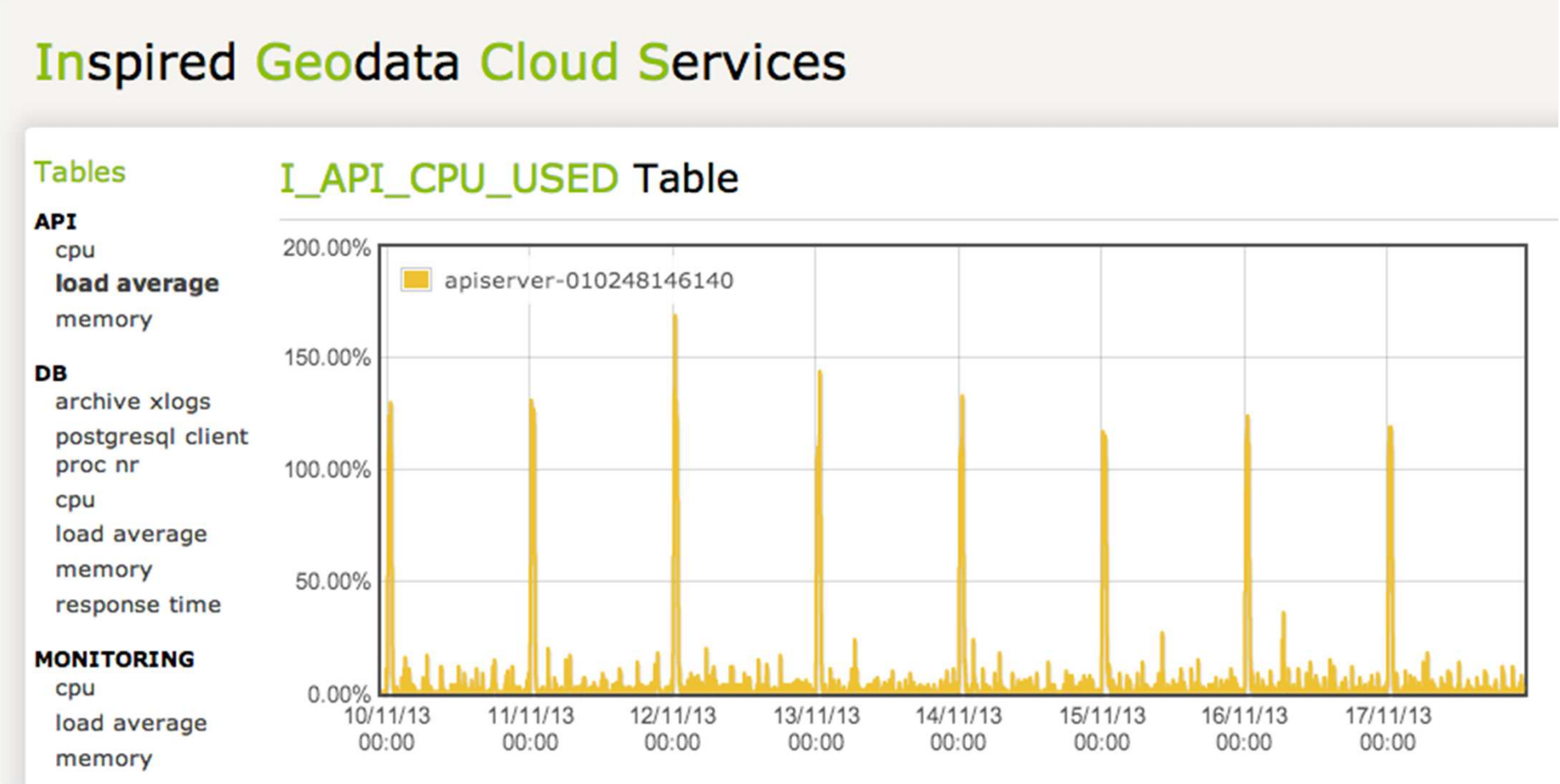
Administration - Monitoring

- Monitoring facilities of both:
  - Platform Services
    - Status
    - Detailed information related to the specific service
  - Cloud Resources
    - Load average
    - CPU
    - Memory usage



- By using a Nagios-based solution
  - With a dedicated instance for harvesting the data
  - Information stored in a database (Amazon RDS)
  - With an interface to query the data

Monitoring


Inspired Geodata Cloud Services
Tables
API
cpu
load average
memory
DB
archive xlogs
postgresql client proc nr
cpu
load average
memory
response time
MONITORING
cpu
load average
memory
I_API_CPU_USED Table
apiserver-010248146140
200.00%
150.00%
100.00%
50.00%
0.00%
10/11/13 00:00
11/11/13 00:00
12/11/13 00:00
13/11/13 00:00
14/11/13 00:00
15/11/13 00:00
16/11/13 00:00
17/11/13 00:00

Administration - Accounting

- Problem:
  - Shared resources
  - Shared services
  - Different usage by each provider
  - Dedicated instances (geo-processing)
- Solution:
  - Measure the storage used by each provider (FS and DB)
  - By application logs to estimate usage shares of indivisible services (e.g., compute hours of Map Server)
- Result is an estimation of:
  - Per-service cost
  - Per-provider cost, service by service

Administration – Accounting API

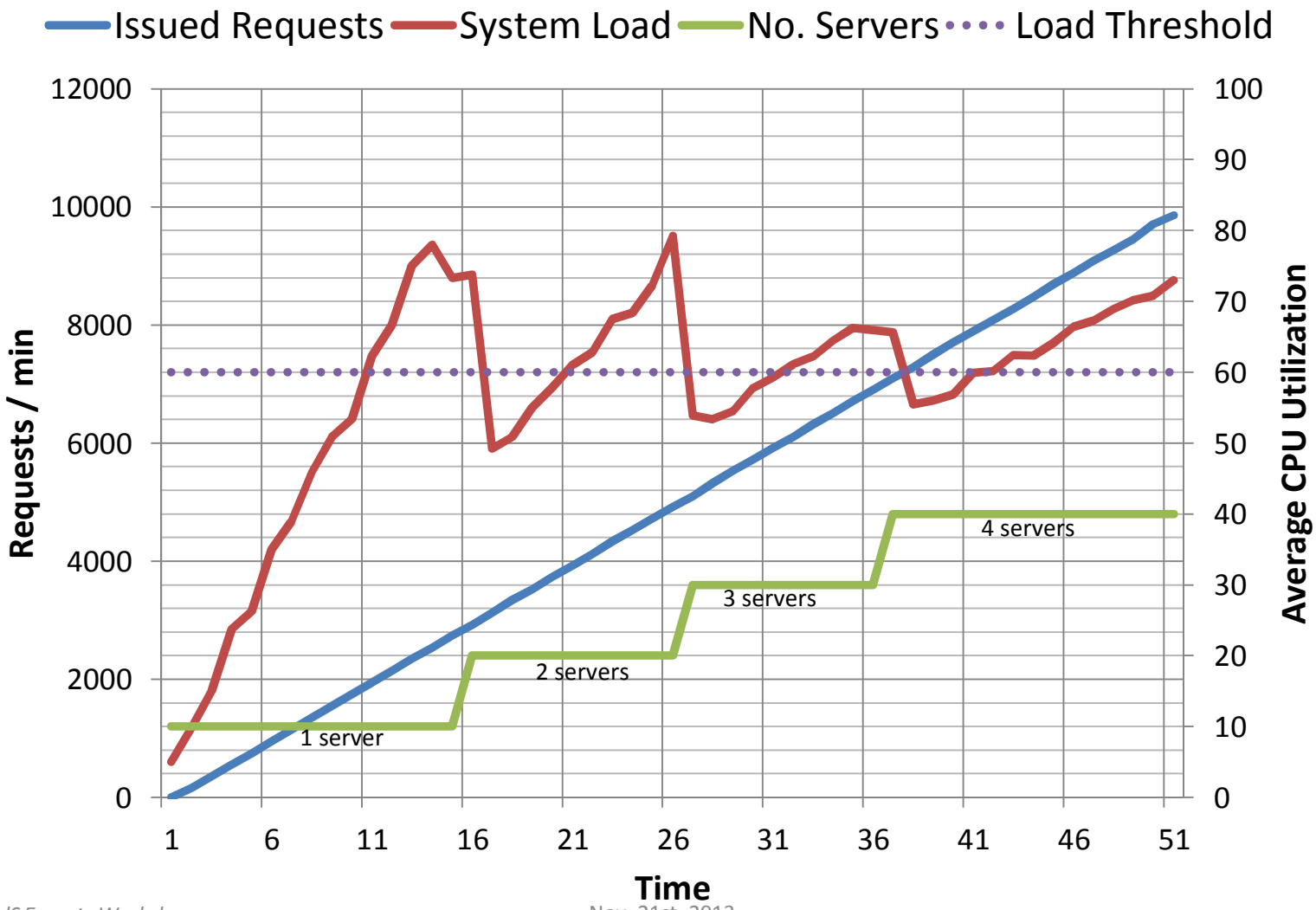
- RESTful services for usage estimation:
  - Per-service usage of each provider (Administrator)
  - Per-service usage of the specific provider (Provider)

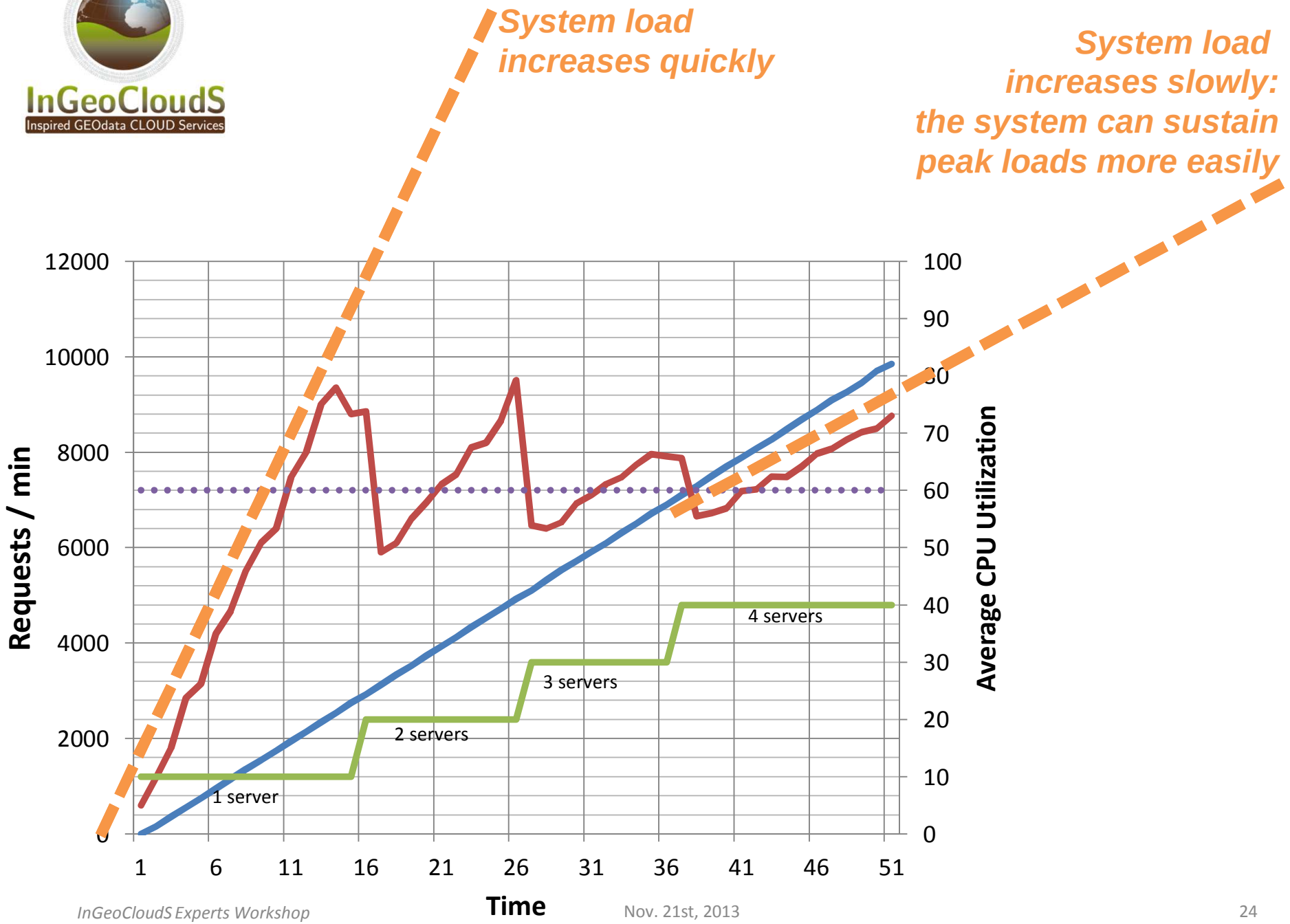
- RESTful services for cost estimation:
  - Per-service cost of each provider (Administrator)
  - Per-service cost of the specific provider (Provider)

Elastic Web Server

- Perfect use-case for demonstrating the feasibility of the cloud
  - By setting a proper load balancer policy (session stickyness)
  - By setting good scaling conditions
  - The service will scale up/down automatically
  - Without service interruptions
  - Preserving a good response time
  - Reducing costs during low platform usage
- The API provides you:
  - The status of the service
  - The number of instances used by the service

Elasticity Experiment: Elastic Web Server




System load increases quickly
System load increases slowly: the system can sustain peak loads more easily
Requests / min
Average CPU Utilization
Time
12000
10000
8000
6000
4000
2000
0
100
90
80
70
60
50
40
30
20
10
0
1
6
11
16
21
26
31
36
41
46
51
1 server
2 servers
3 servers
4 servers

Thanks for your attention