On-demand File Caching with GlusterFS

Gustavo Brand
Adrien Lèbre
École des Mines / Scalus EU Project

Nantes – France

Nov 2012

Agenda

- This is an entry level Development talk
  - Different approach using Gluster components
- Problem
  - Investigation experiments
- Where Gluster fits in
  - Current and Future scenarios
  - Fops overview
- Gluster observations and some issues
  - Examples

Problem statement

Marie Curie Initial Training Networks

SCALUS — SCALing by means of Ubiquitous Storage

- Problems:
  - Is it still relevant to use additional nodes to analyze data faster?
  - How **applications** accessing data through the DFSs at LAN/**WAN** levels are **impacted**?
  - Should **locality** be considered as a **major concern** to **design a DFS**?

- In summary: **Validate the assumption** about the impact that **metadata/data traffic and latency** have into **wide area** scenarios when **executing parallel applications**.

Investigation through experimentation

Marie Curie Initial Training Networks

SCALUS – SCALing by means of Ubiquitous Storage

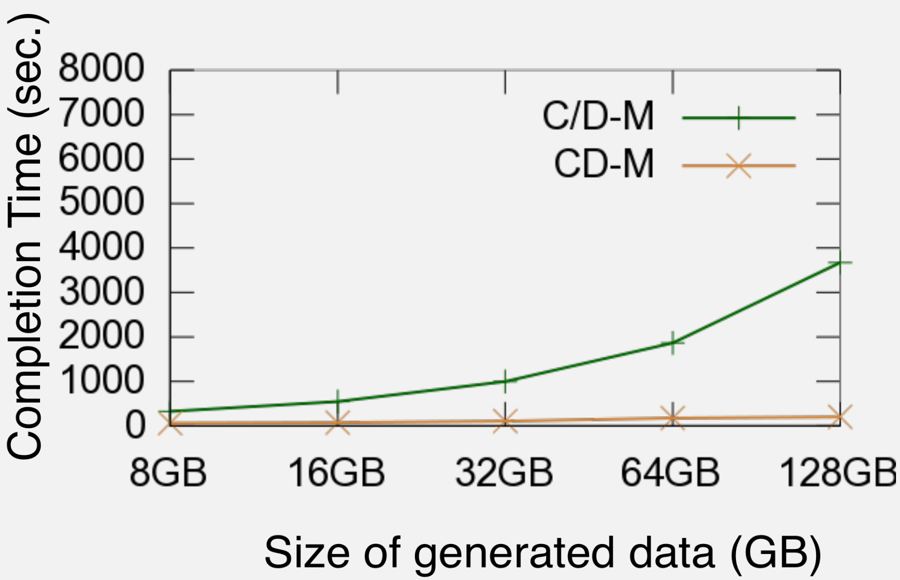
- Investigation method:
  - We conducted experiments with:
    - HDFS and Lustre
    - Different scenarios within **LANs(-)** and **WANs(/)**:
      - CD-M, C-D-M
      - CD/M, C-D/M, CD-M/CD, C-D-M/C, C/D-M
    - Running **3 M/R applications** (grep, writer, sort)
      - Fixed 8GB file size with 16, 32, 64 nodes
      - **Scaled** the file size **8-128GB** with **64 nodes**
    - Main reference measured: **runtime**, among others

Invest. findings

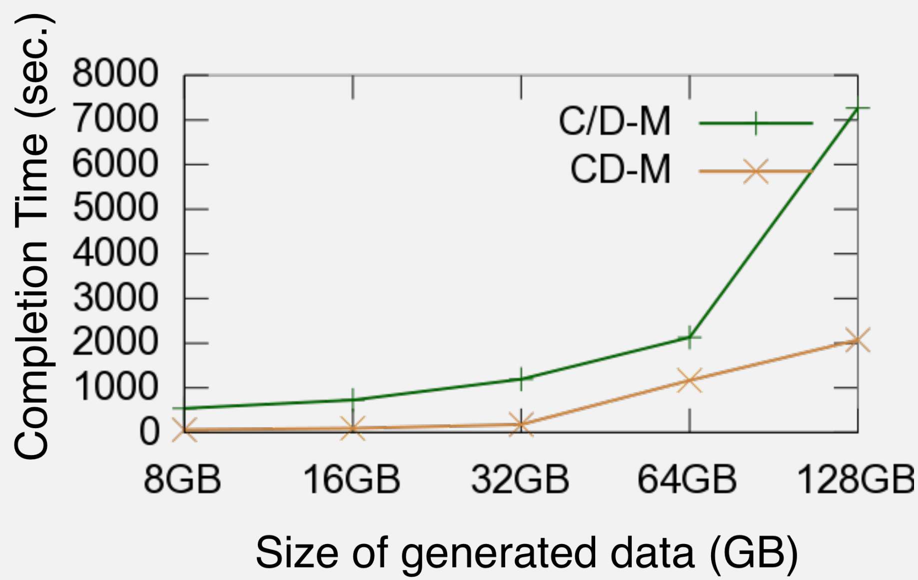
Marie Curie Initial Training Networks

SCALUS — SCALing by means of Ubiquitous Storage

- Exploring **bigger file sizes** with the best and worst scenarios: Sort application



HDFS / Sort



Lustre / Sort

Invest. findings

Marie Curie Initial Training Networks

SCALUS SCALing by means of Ubiquitous Storage

- In most cases, accessing data through **WAN** leads to **worse performance** (as expected…)
- It was **better to use less nodes than** trying to **benefit from external WAN ones**
- The completion time for the **local scenarios with 16 nodes** is **similar** to the **WAN** ones using **2x or 4x** more nodes

Current WiP - GBFS

- Use **implicit striping** (propagation according to the access pattern)
  - **Avoiding unsolicited traffic** through local placement
  - Extending it **group-wide** (stripe into the group)

  - Using the **groups** to manage which data is stored locally (WAN, LAN, local node... you name it)
  - Having a persistent and volatile **LRU persistent caching per group.**

What about Gluster?

Marie Curie Initial Training Networks

SCALUS

SCALing by means of Ubiquitous Storage

- Using some of its translators:
  - **Posix + Server** translators as backend
  - **GBFS + Client** translators at the “client” side.

- Peer to peer way, with each node having a server(+brick) and client(s).

- At GBFS, additional info is provided about other subvols at start time as volume “options” (i.e.: groups each one is related to).

What about Gluster?

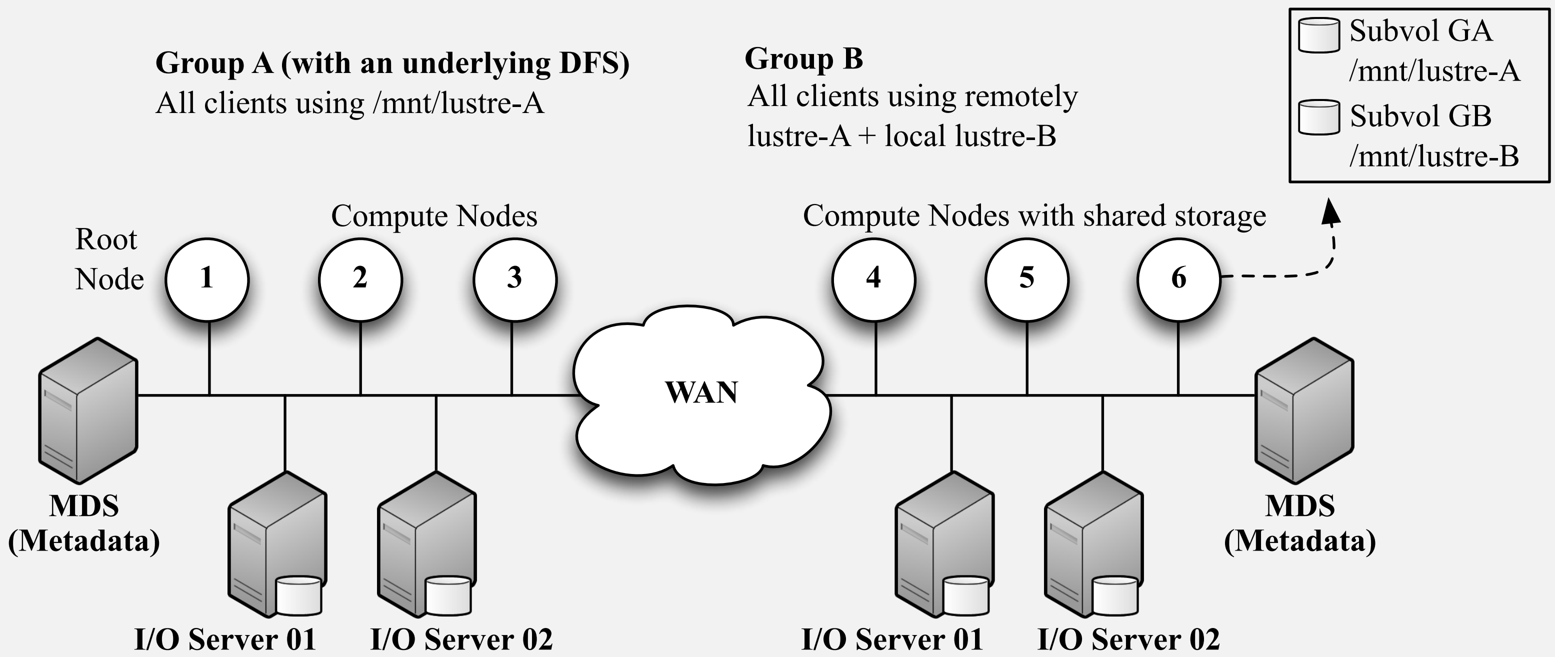
- Advantages: easy access to other nodes through the subvolumes (masking RPC comms)
- Easy installation over FSs/some DFSs
  - xattr size limits can be problematic
- Translators approach enables a good functionality granularity.

Scenario - phase 1

Marie Curie Initial Training Networks

SCALUS
SCALing by means of Ubiquitous Storage

- Local DFSs at each side

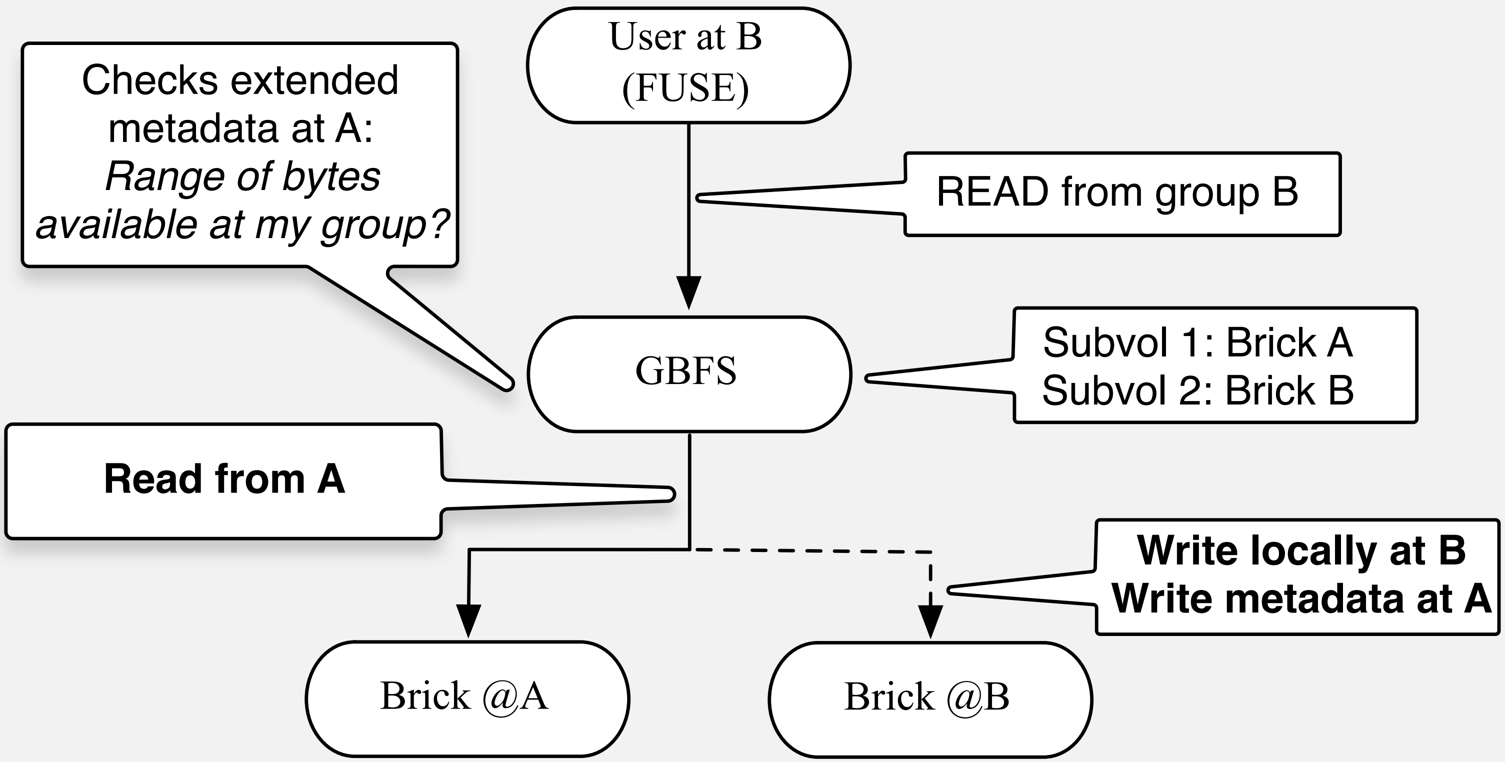


1st read from B

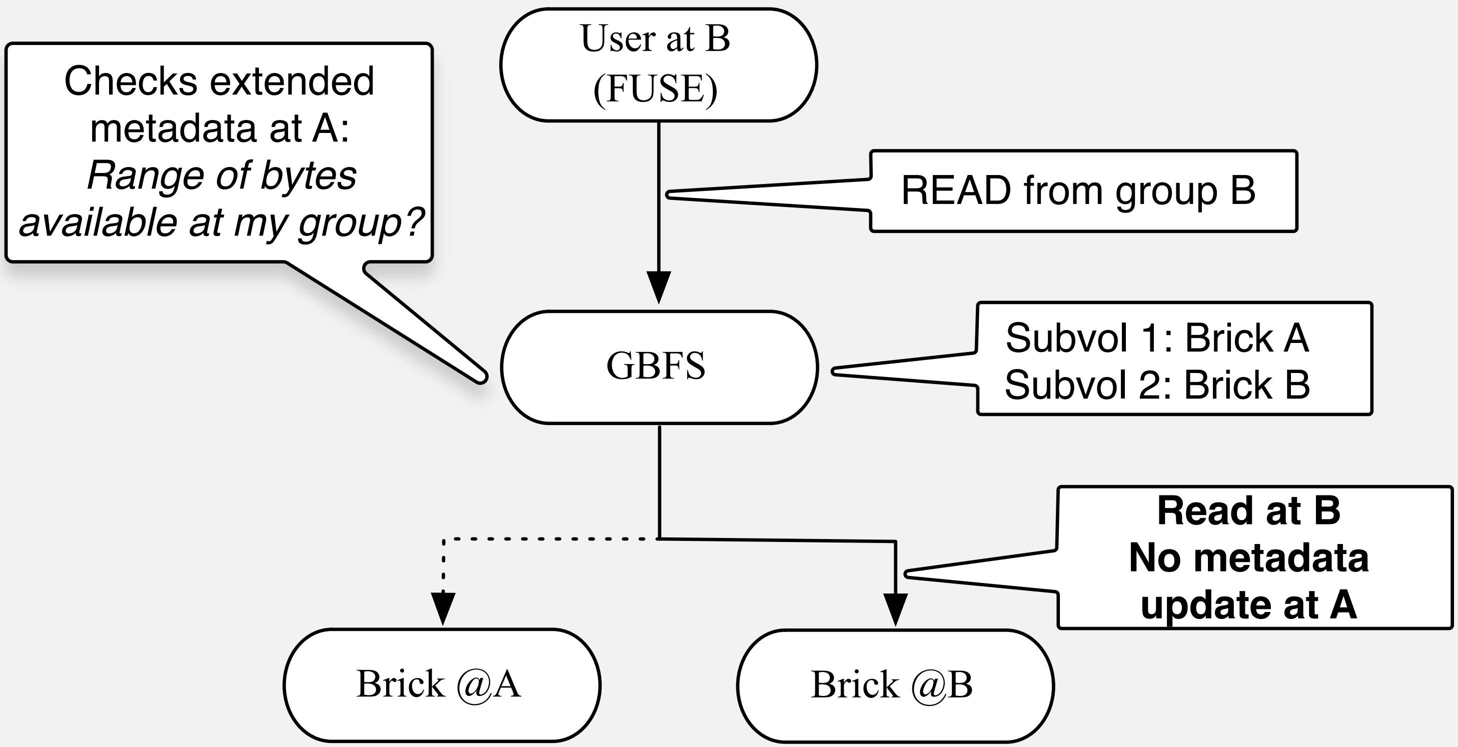
Marie Curie Initial Training Networks

SCALUS

SCALing by means of Ubiquitous Storage



2nd read from B



Write

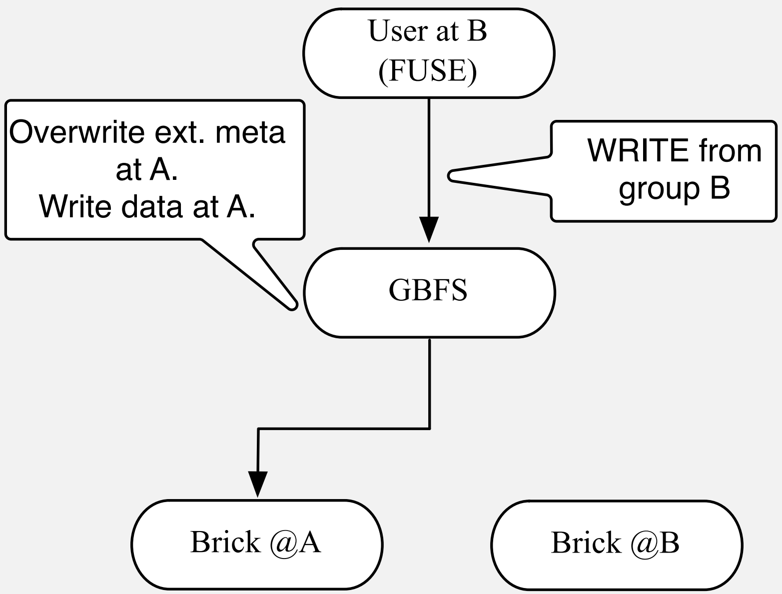
- Using same GFID, with sparse files
- Testing different algorithms/versions:
  - 1st: write centralized
    + overwrite metadata on write at A
  - 2nd: write locally
    + overwrite local metadata on write
    + **validate** local metadata on read
  - 3rd: write locally
    + **invalidate** metadata on write
- Going from centralized to distributed.

Write example

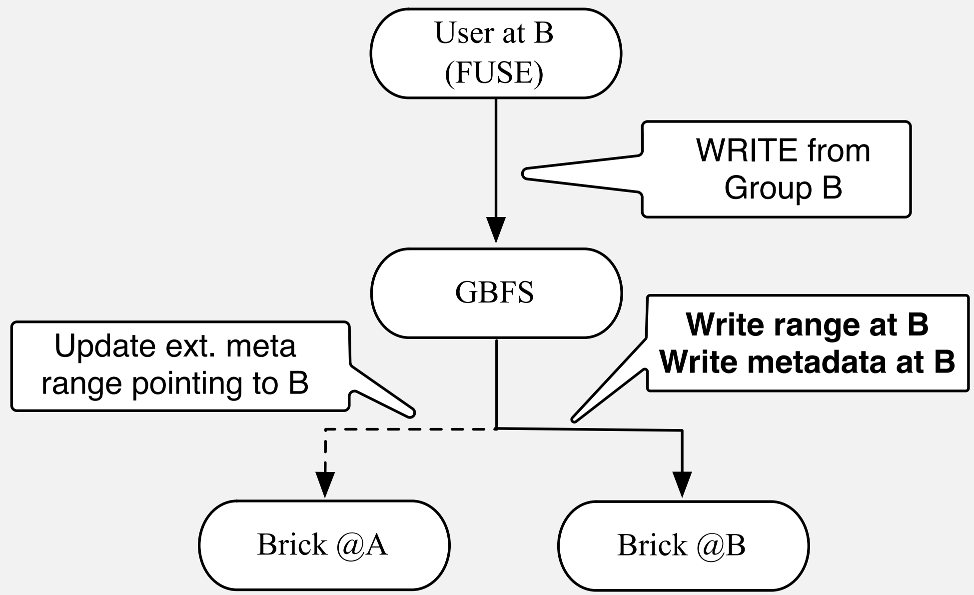
Marie Curie Initial Training Networks

SCALUS
SCALing by means of Ubiquitous Storage

1st simple approach Central write



3rd approach, locally Invalid on write

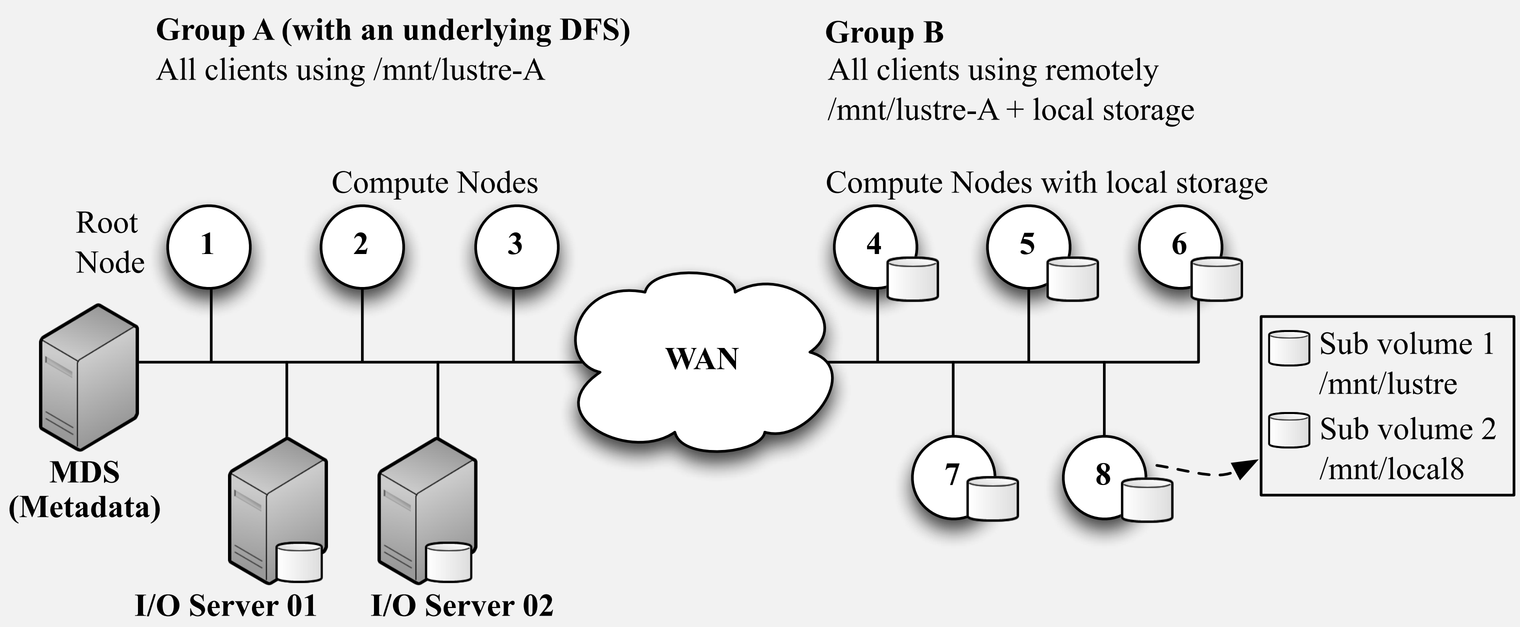


Scenario - phase 2

SCALing by means of Ubiquitous Storage

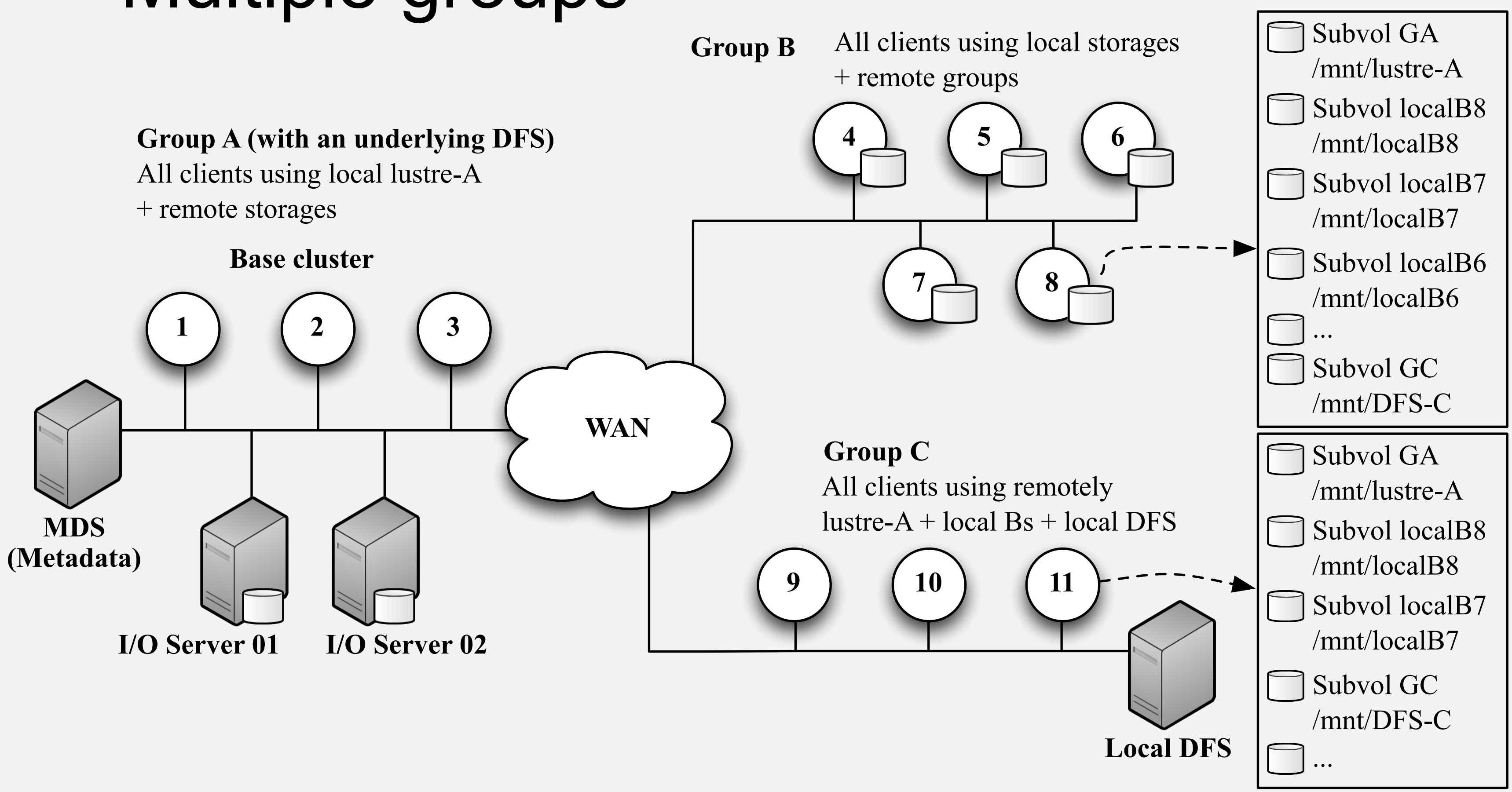
Marie Curie Initial Training Networks

- Local storage at each node (at B)



Bigger group shot

- Multiple groups

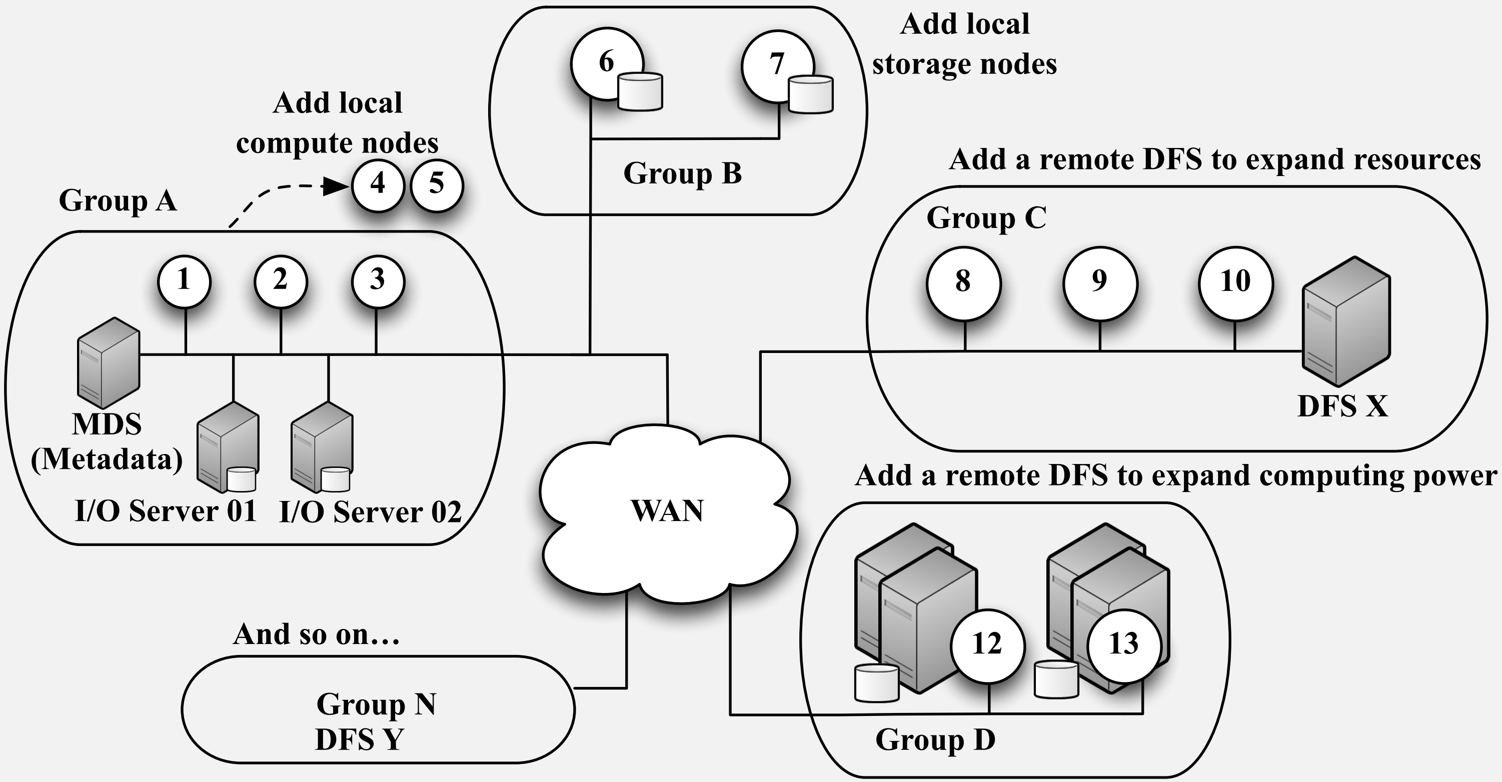


Nodes addition on demand - concept

Marie Curie Initial Training Networks

SCALUS SCALing by means of Ubiquitous Storage

- Having an elastic addition and capacity



Gluster observations

Marie Curie Initial Training Networks

SCALUS

SCALing by means of Ubiquitous Storage

- More dev documentation would be helpful :)
  - “Translator 101” and “xlator_api_2” posts are very good
  - Main resources are Jeff Darcy blog entries, the gluster-devel list and the IRC.
- Few comments all over the code (as of 3.3 at least)
- Overall architecture of gluster communication protocols, client/server/bricks algorithms, etc.

Gluster observations

Marie Curie Initial Training Networks

SCALUS
SCALing by means of Ubiquitous Storage

- Complex translators like DHT and AFR details in terms of routines, schemes, comments would be helpful as well.
  - Ex: Algorithm about how each one keeps a unified namespace, for example (overall view).

- Common develop. techniques details can save a lot of time from contributors/new developers.

Common techniques

- fd_ctx and inode_ctx common usage;
- general use of local / private / *_ctx per translator;
- striping techniques used at current code;
- stubs/sync calls (pending at xlator_api_2 :))
- etc...

Common techniques

Example taken from gluster devel-list

--> Syncop flow:

```
read() {
  // pull in other content
  while(want more) {
    syncop_lookup()
    syncop_open()
    syncop_read()
    syncop_close()
  }
  return iovec / UNWIND
}
```

--> WIND / UNWIND flow:

```
read() {
    while(want more) {
      WIND _lookup()
    }
 }

 _lookup_cbk() {
      wind _open()
 }

 _read_lookup_cbk_open_cbk() {
      wind _read()
      add to priv/fd_ctx/local->iovec
 }

_read_lookup_cbk_open_cbk_read_cbk() {
      wind _close()
      UNWIND
 }
```

- Thanks

  Contact:
  gustavo.bervian_brand@inria.fr
  gbrand at gluster-devel IRC :)