WAN data access studies

Paolo Franchini

13 december 2012

Library state-of-art: `libSbNet`

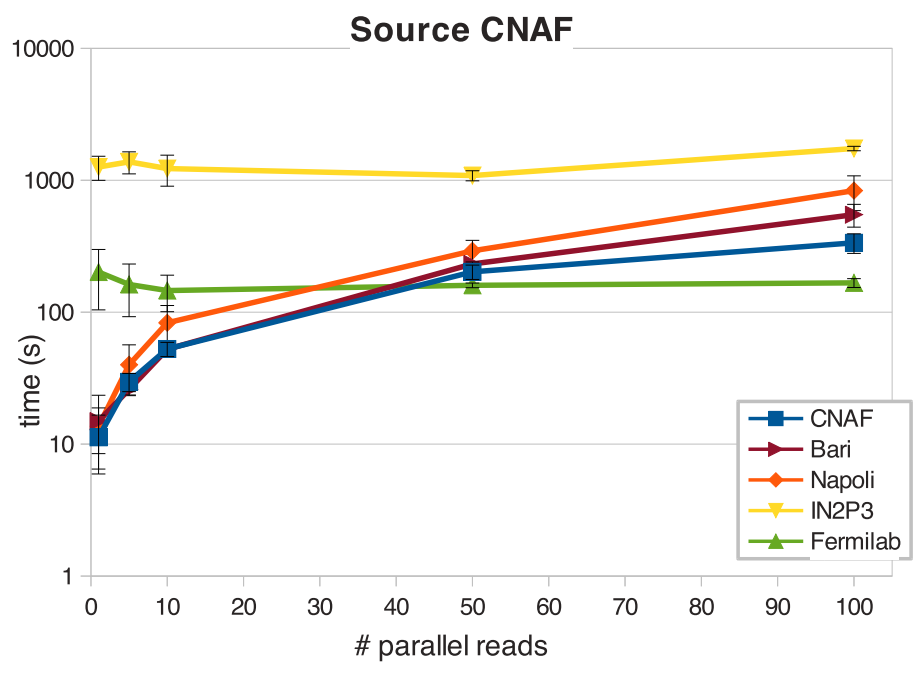
- Grid community interest in WAN data access via http protocol
  - "Towards an HTTP Ecosystem for HEP Data Access" https://indico.cern.ch/conferenceDisplay.py?confId=218328
- The library takes as input a catalog file identifier (lfn://), and returns a TURL identifier, after checking its actual availability.
- The supported protocols are **file**, **gsiftp** and **http**. It is possible to select the protocols and their priority with the configuration file.
- The library first of all tries to use the default SE, with the selected protocols. Then if the default is not defined, builds the list of all SRMs that holds a file copy, in "ping time" order, and tries to obtain the TURL from the "nearest" one. The search continues until a TURL is found or the SRM list ends.

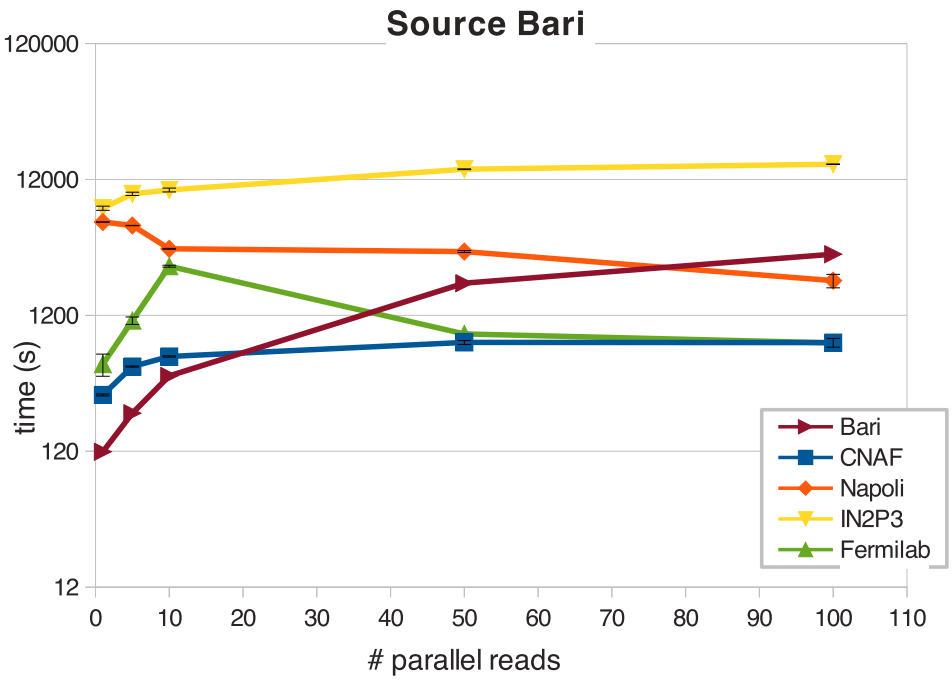
Data access test

- Test goals:
  - measure the latency period due to the increase number of parallel read stream
  - measure the latency period due to the increase of round trip time elapsed between source and destination
  - support the development of a general, experiment wide, data access software layer
  - start the characterization of a concrete WAN scenario, including traffic impact, typical latency, network resource overloading

Test bed validation

- Test layout definition of the preliminary tests:
  - 1, 5, 10, 50 and 100 parallel set of read streams
  - each stream reads a random files according to a trace file obtained from an analysis application
  - 250 compressed root files, 476 MB each
  - data sources: INFN-T1 and INFN-Bari
  - jobs destinations: INFN-T1, INFN-Napoli, INFN-BARI, IN2P3 and FNAL
  - measured the time of the `cURL` execution





Next test desing

- Use ROOT to perform the file accesess using the protocols already implemented:
  - class TFile for *file* protocol
  - class TWebFile for *http* protocol
  - class TNetFile for *root* protocol
- Measure the latency in a real analysis application provided by Elisa Manoni
- More statistical relevance, running each job 10-50 times
- Control over the network (LHCONE infrastructure) using monitors and network performances tools
- Results within February 2013

Test and development on HadoopFS

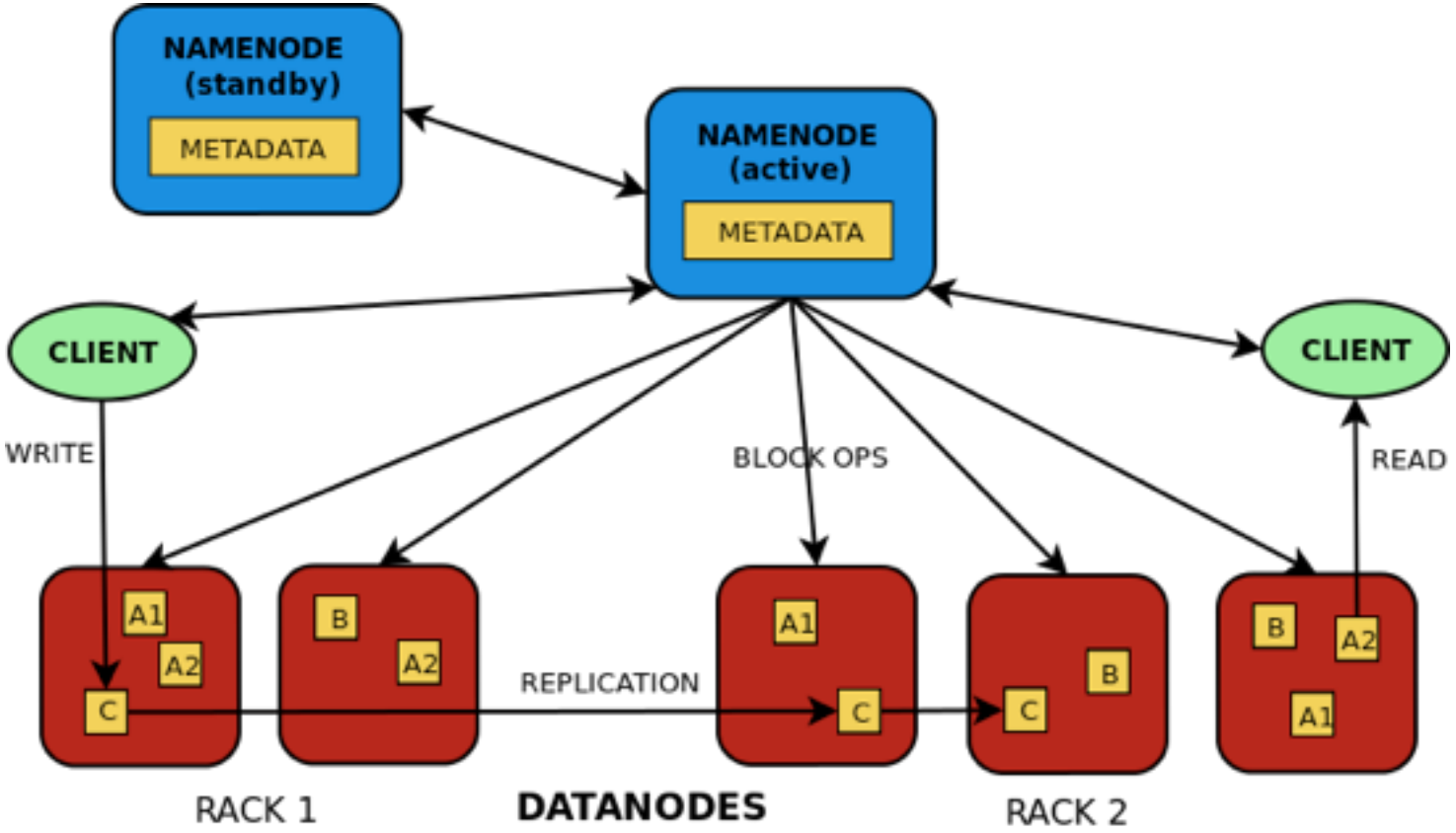
Giacinto DONVITO
ReCaS-INFN
Giovanni MARZULLI
GARR-INFN

HDFS features

Hadoop **D**istributed **F**ile **S**ystem

- Open source
- Large dataset
- Fault tolerance
- Scalability
- Commodity hardware
- Rack awareness

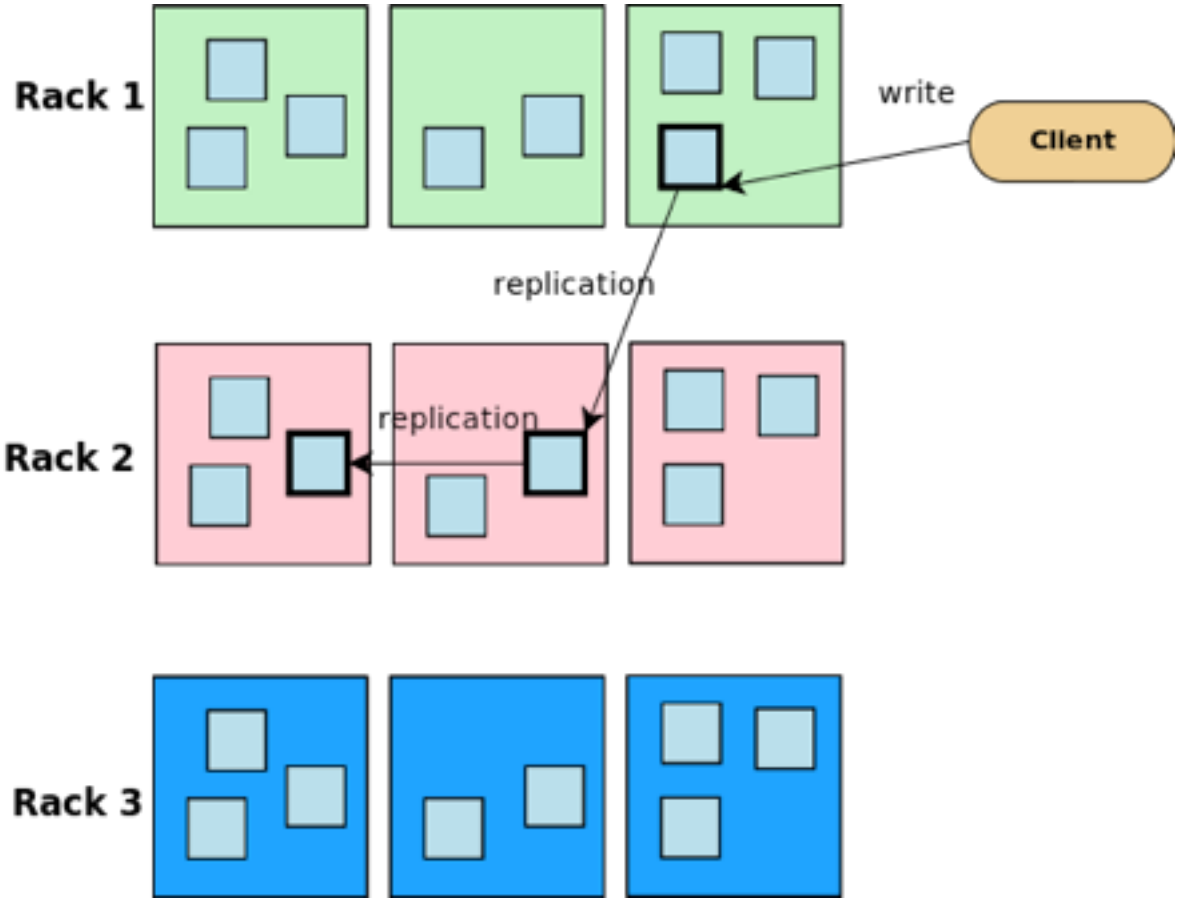
HDFS Architecture



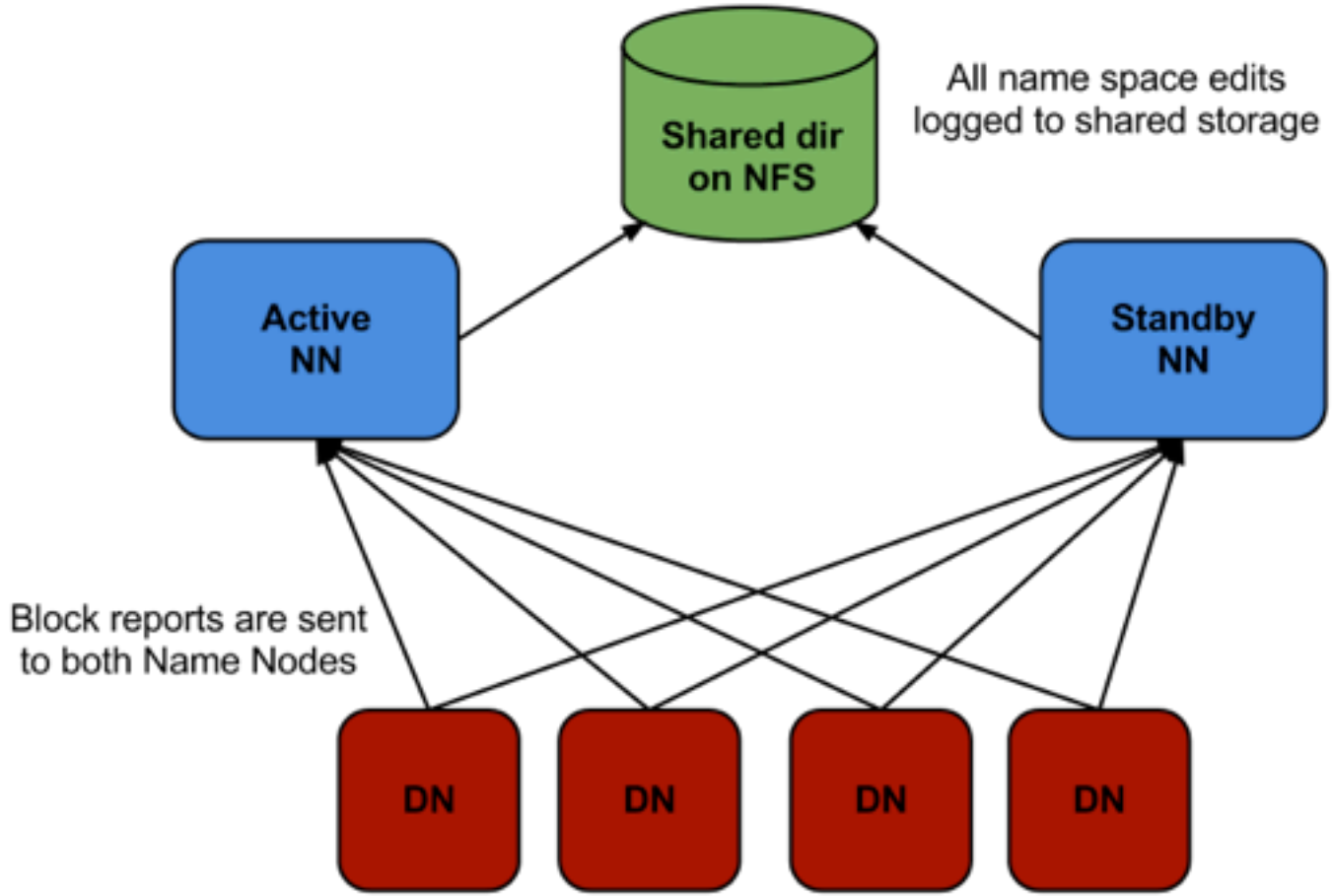
Placement policies

- Default policy
  - 1 replica on a node of local rack, 2 replicas on different nodes in the same remote rack
- Developed policies
  - One Replica per Rack
  - Hierarchical

Default placement policy



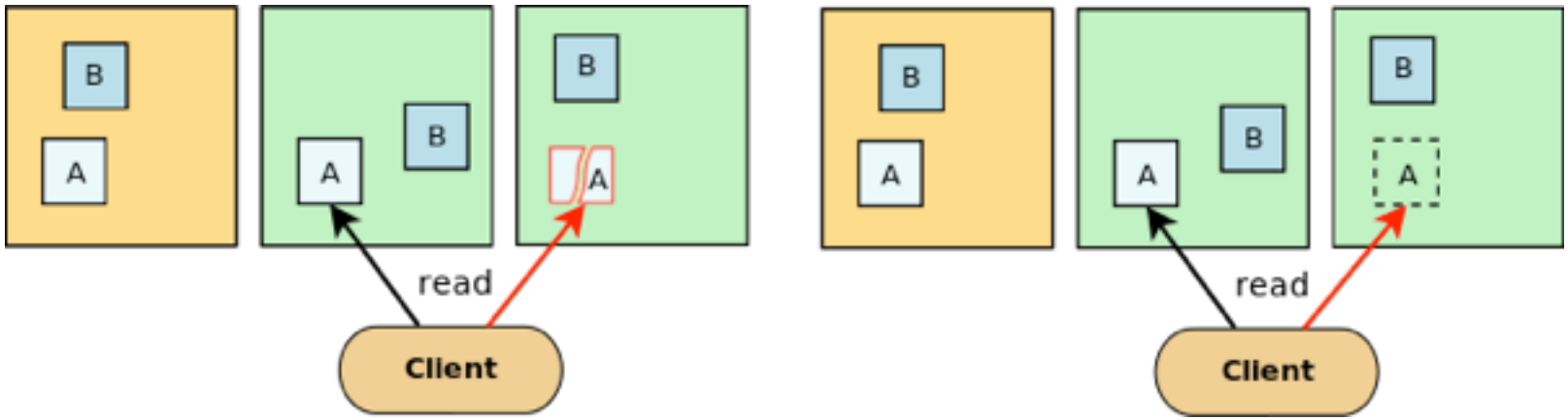
HA namenode


Shared dir
on NFS
All name space edits
logged to shared storage
Active
NN
Standby
NN
Block reports are sent
to both Name Nodes
DN
DN
DN
DN

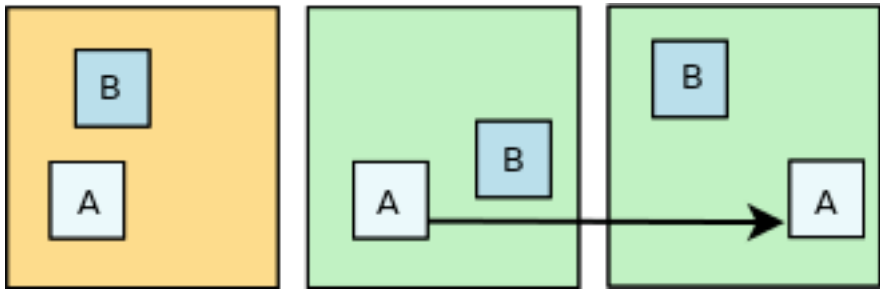
Namenode test

- Metadata corrupted or lost
  - Recovery from secondary namenode
    - `hadoop-daemon start namenode -importCheckpoint`
- Namenode down
  - Waiting of clients and datanodes
  - Failover
    - `hdfs haadmin -failover nn1 nn2`

Datanode test: lost or corruption data



- Automatic recovery



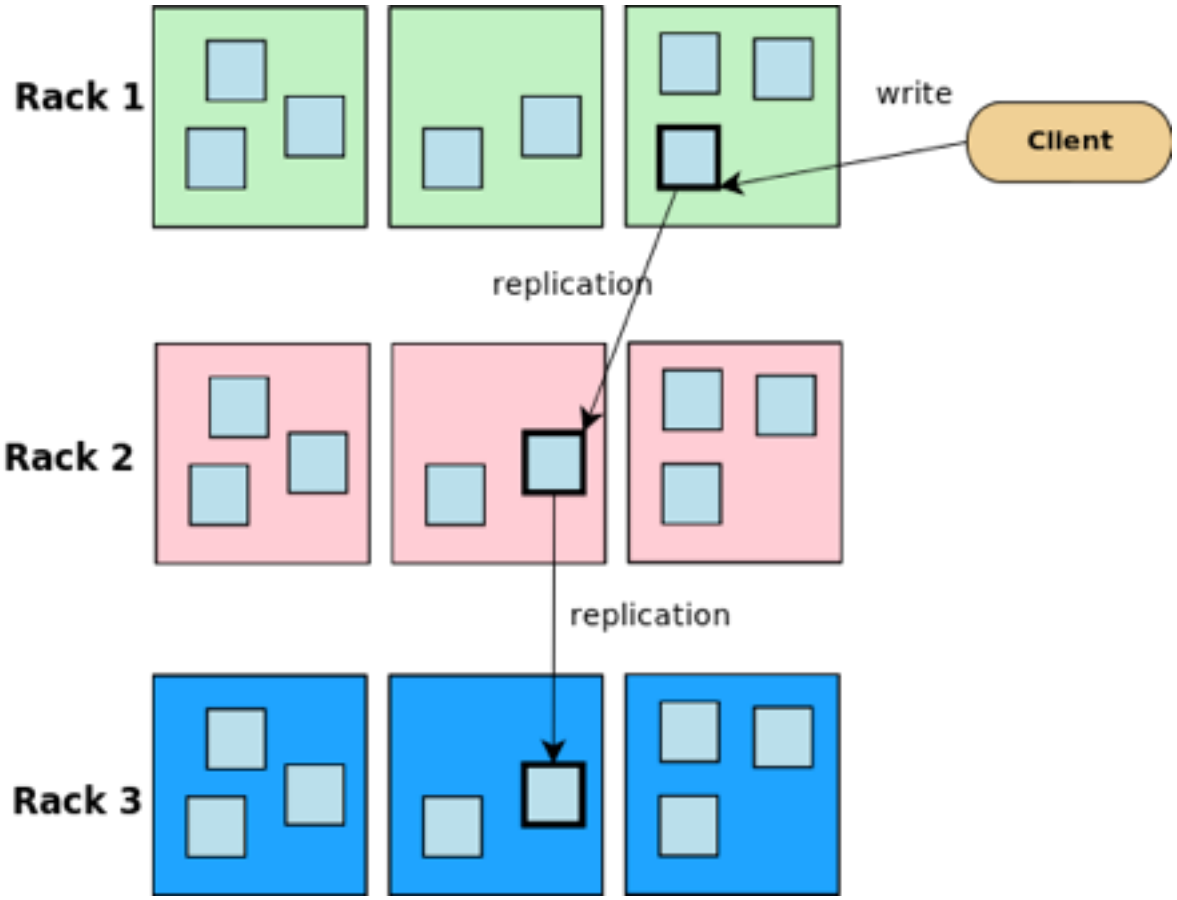
Datanodes test

- Under-replicated blocks
  - After datanode failure
- Over-replicated blocks
  - After recovery and restart datanode
- Mis-replicated blocks
  - Policy violation
- Datanode failure during writing/reading
  - Switch to other live nodes
- Workload

One replica placement policy

- 1 replica per rack
- More reliability
  - 2 racks fault tolerant (if replication factor is 3)
- More data distribution
- Less read cost
  - Reading from nearest replica
- More write cost
  - More data transmission

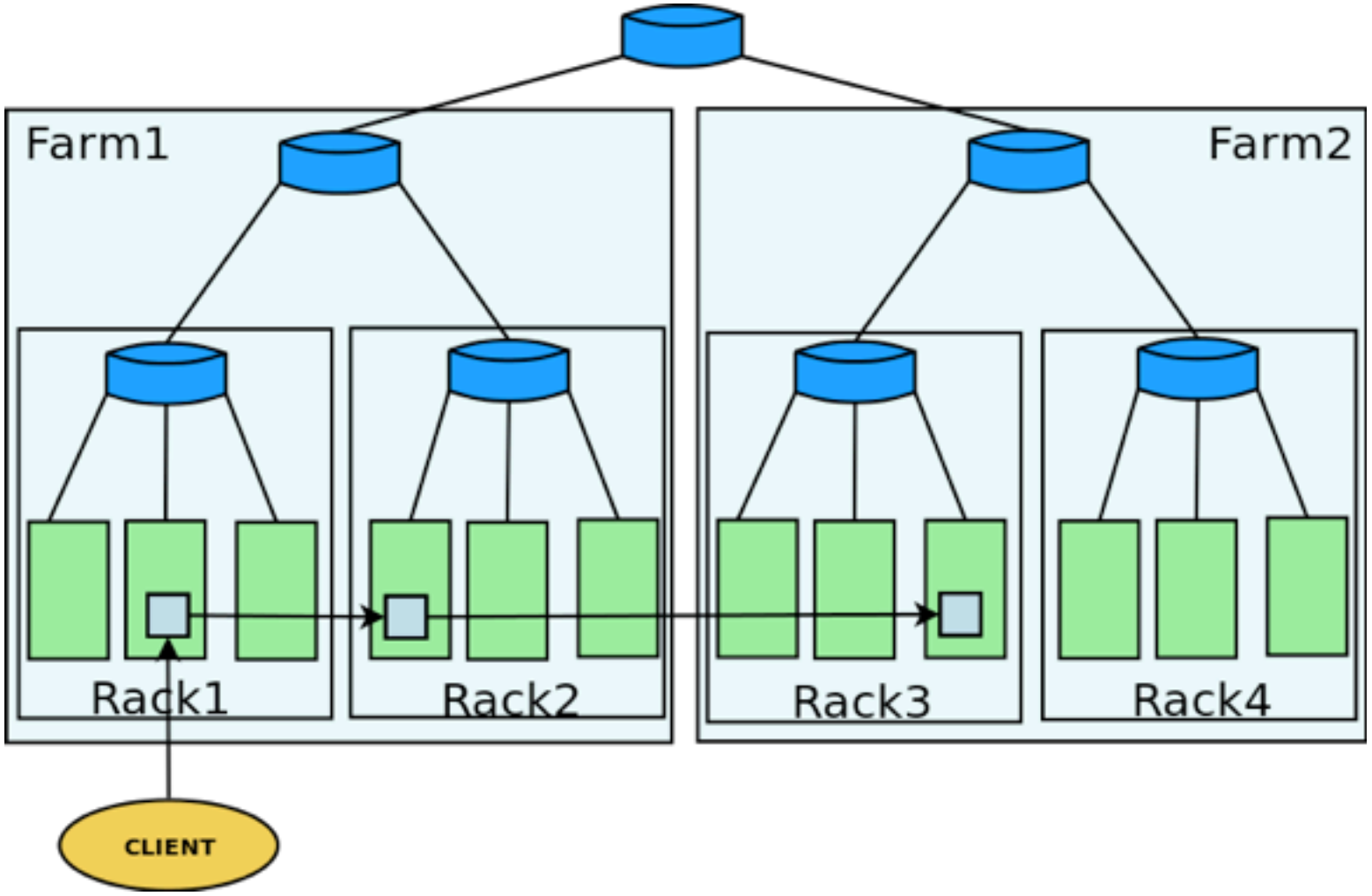
One replica placement policy



Hierarchical placement policy

- Awareness of hierarchical network topology
- 2 replicas in local farm but in different racks
- 1 replica on a rack of remote farm
- Tolerance of whole farm fault

Hierarchical placement policy

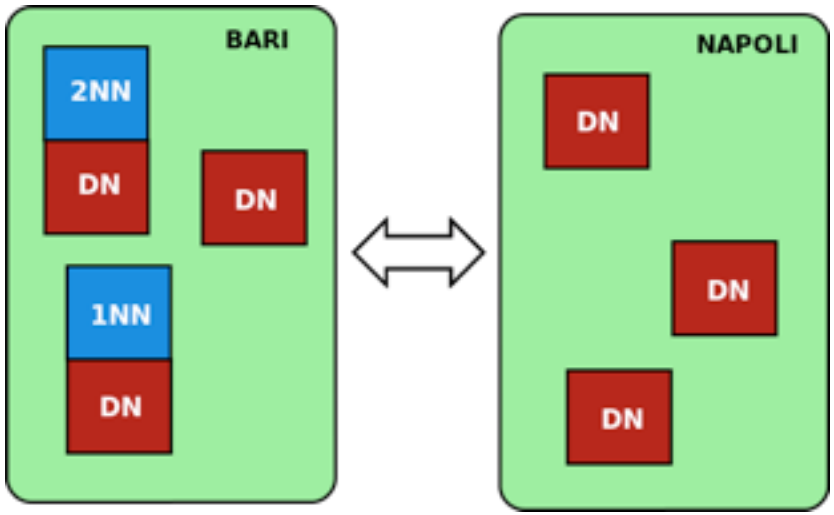


Development of custom policies

- BlockPlacementPolicy Java abstract class
  - Default implementation
    - BlockPlacementPolicyDefault
  - Custom implementations
    - BlockPlacementPolicyOneReplica
    - BlockPlacementPolicyHierarchical
- The Policy is configurable in the configuration file

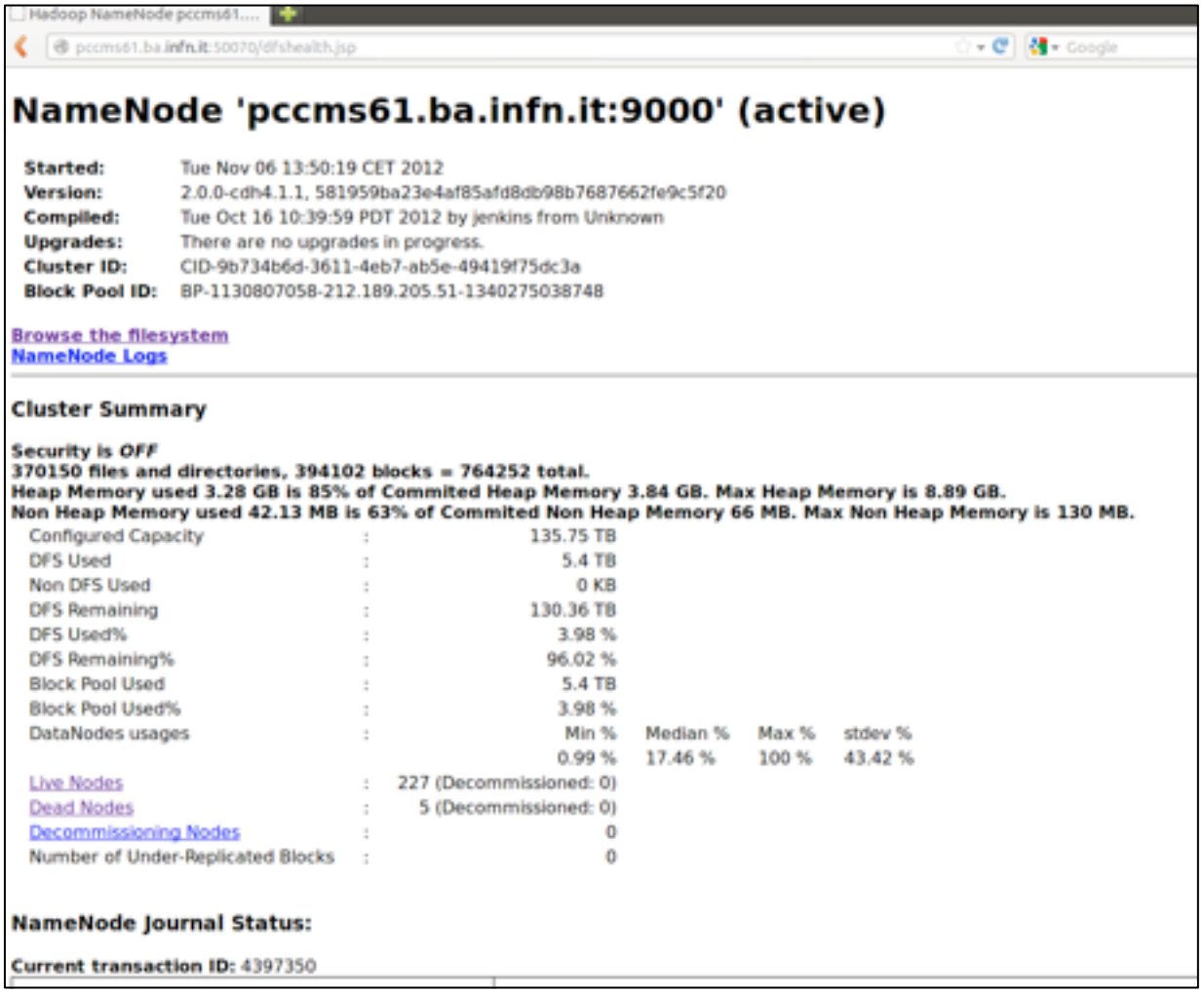
Geographic cluster

- INFN Bari and INFN Napoli (ReCaS sites)

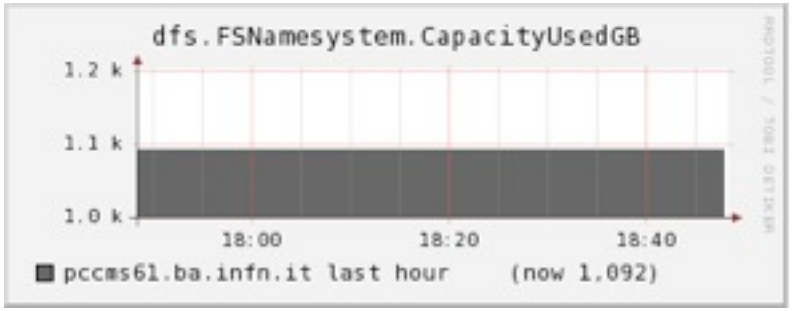


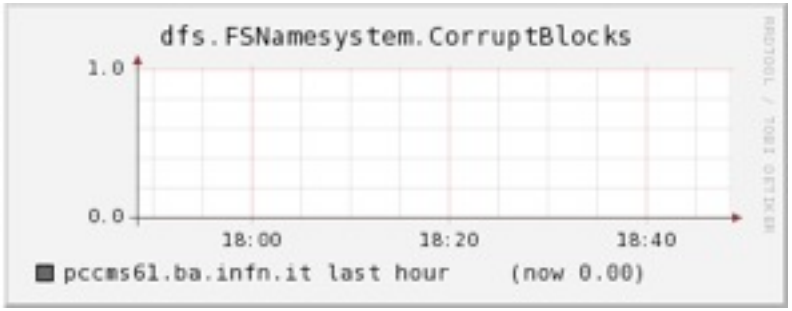
- Functionality test
- Custom policies test

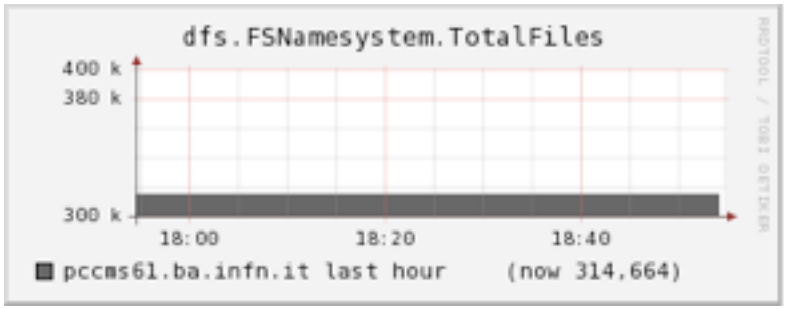
INFN Bari (pre)production cluster

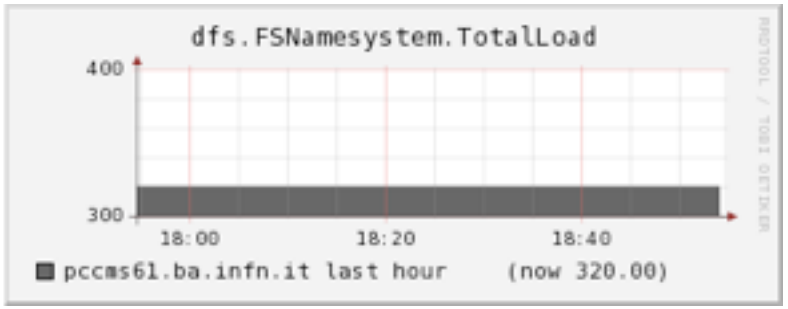


Ganglia monitoring









Custom monitoring

- We developed a monitoring system in order to track:
  - Locations of blocks placements
    - Recent blocks history
    - Corrupted or missing blocks
  - Blocks operations
- Stored in a relational database

Automatic node installation and configuration

- Script procedure to run on each node, that provide
  - Software installation
    - Packages repository
  - Configuration based on nodetype
    - formatting, mounting and assigning unused disks/ partitions to the file system
  - Process restart if node falls

Performance test

- Writing and reading mean rates on 3000 file operations

Test setting

Parameters	values
Datanode	Active, passive
Block size (MB)	64, 128, 256
Client	Hadoop, Fuse-dfs
Replication factor	1,2
File dimension (MB)	4096
Complete dataset	3K File operations

Performance test: statistical results

Writing mean rate (MB/s)

Block size	Replication Factor 1		Replication Factor 2	
	Passive datan.	Active datan.	Passive datan.	Active datan.
64MB	58,48	85,23	46,00	56,98
128MB	64,31	87,86	50,72	55,34
256MB	91,98	83,98	46,61	55,68

Reading mean rate (MB/s)

Block size	Replication	Replication
64MB	61,57	62,29
128MB	67,59	60,87
256MB	66,24	61,84

Performance test: real case

- 600 jobs of Pamela reading ROOT files from HDFS via Fuse-dfs

Future works

- Infrastructure expansion
  - Long-run test on cluster up to 300 nodes and 500TB of disk space
  - Up to 4000 jobs simultaneously running
  - Scalability test
- Geographic test of 3 sites-cluster
  - Add another ReCaS site to the existing cluster
- Research of optimal configuration
  - Block size
  - Fuse-dfs
- Production of namenodes federation

Conclusions

- Strength of data reliability
- Strength of automatic recovery behavior
- Optimization in order to increase reliability and performance
- Positive feedback by first real users

People involved

- Giacinto Donvito (ReCaS -- INFN)
- Giovanni Marzulli (GARR -- INFN)