StoRM

disk management middleware

CHEP 2004, Interlaken

27th September 2004

Flavia Donno, INFN
Antonia Ghiselli, CNAF-INFN
Luca Magnoni, CNAF-INFN
Riccardo Zappi, CNAF-INFN

Problem statement (1/3)

Storage System

- For applications that are disk-intensive, particular attention should be paid to storage resources, such as those providing **higher capacity**, **redundancy**, **scalability**, etc..
- More than decent **performance** are required to access your data.

Problem statement (2/3)

Space Reservation

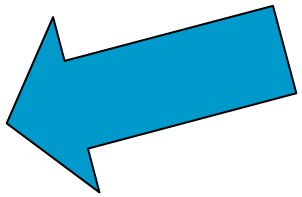
- In a storage system, we can identify two kinds of resources : **file** and **space**.
- It often authorized users on storage acts as **concurrent consumer** of storage resources.
- Concurrent consuming of storage resources may cause jobs failure, specially if there is no certainty of **space availability**.

Problem statement (3/3)

Direct access

- Data-intensive jobs want **direct access on** reserved space with good I/O performance.
- Some data-intensive job might not be able to be adapted to use reserved space through an “external” space management application (**backward compatibility**).

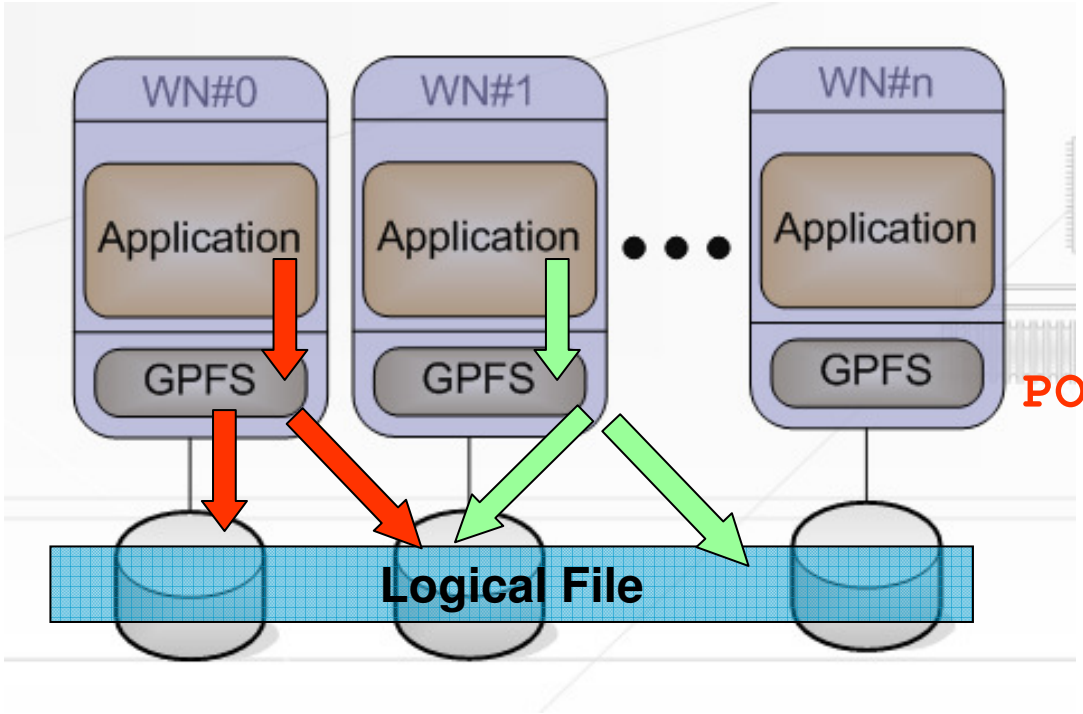
Requirements (1/2)



- **Storage**
  - Efficient I/O
  - Capacity, Fault-tolerance
- **Space Management**
  - Space reservation
  - Direct access to reserved space
  - Management policies

Storage : Efficient I/O
Solution: Parallel File System

- An example of Parallel FS.




```
POSIX I/O
read()
write()
open()
close()
…
```

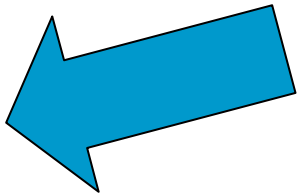
Storage : Scalability, Fault recovery, ..
Solution: GPFS

- The following requirements:
  - Scalability
  - Large files/file-systems
  - Security
  - Failure Recovery

are satisfied by modern parallel file system as GPFS.

Requirements (2/2)

- **Storage**
  - Efficient I/O
  - Capacity, Fault-tolerance
- **Space Management**
  - Space reservation
  - Direct access to reserved space
  - Management policies



Space Management : Interface

- In SRM v2.1 advanced specification there are some new functionalities regarding space management :

srmReserveSpace()
srmReleaseSpace()
srmUpdateSpace()
srmCompactSpace()
...

we have adopted SRM definitions to preserve compatibility with any **Space Management Client** that is **SRM-compliant.**

Space Management : Direct access

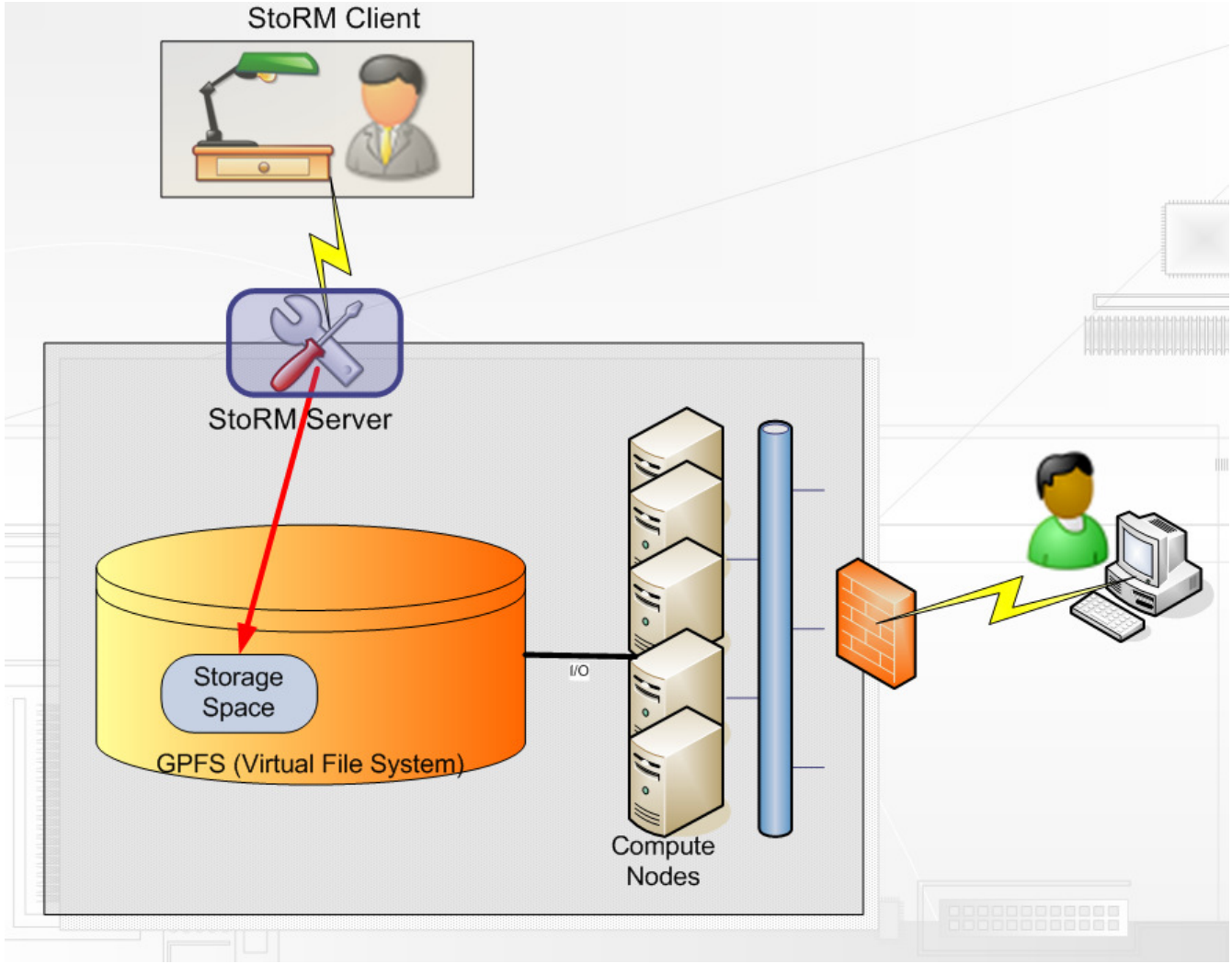
In order to enable applications for **posix access** to the reserved space in a storage, we consider a <path>/<file-name> as a part of *spaceToken*.

path ::= [root] [relative-path]
root ::= [root-directory]
root-directory ::= "/"
relative-path ::= path-element { "/" path-element } ["/"]
path-element ::= name | parent-directory | directory-placeholder
name ::= char { char }
directory-placeholder ::= "."
parent-directory ::= ".."

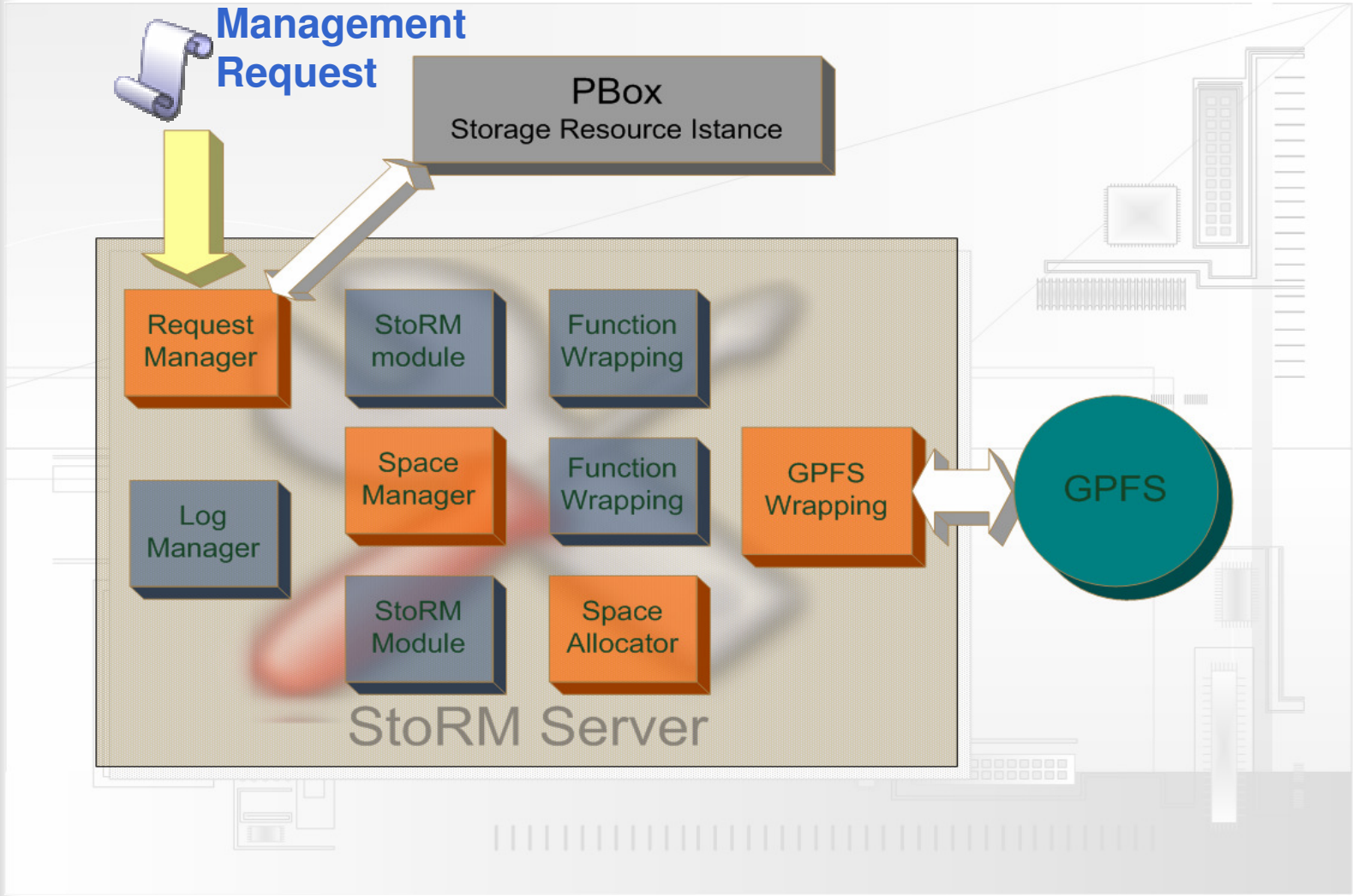
Assembling all the pieces: StoRM

- **Sto**rage **R**esource **M**anager
- StoRM goal is to create a **container** for storage management functionalities as:
  - Space reservation (SRM-compliant)
  - Other SRM functionalities.
  - Management policy enforcement on storage (G-Pbox provided by INFN).

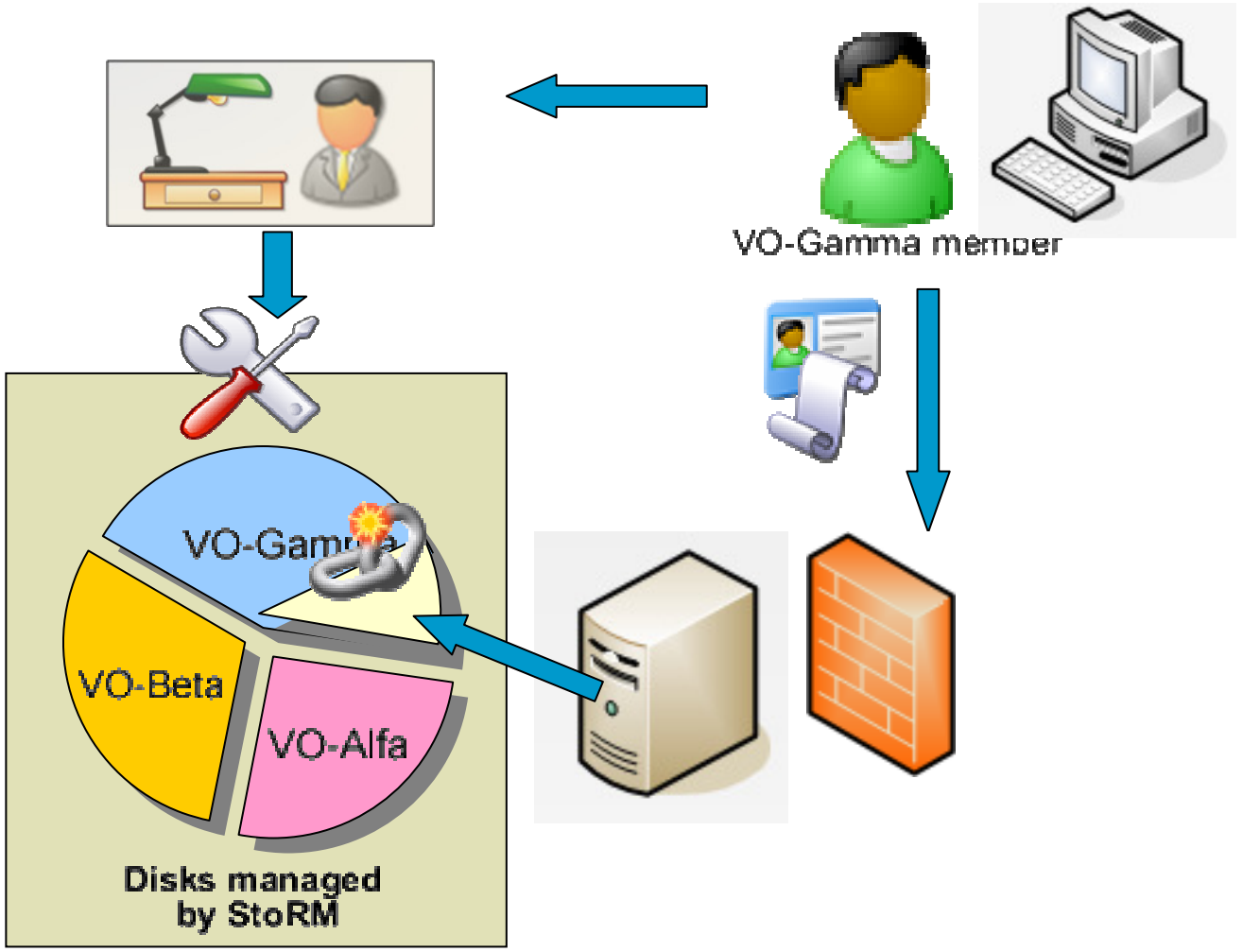
Architecture : access to StoRM.



Architecture : StoRM Server



Use scenario : Use of reserved space



Prototype status and future plan

- Prototype of StoRM implementing reserve-space functionality exists.
- Next steps :
  - Testing of all functionalities defined in SRM advanced involving Space Management.
  - Integration with other Storage Management Client SRM-Compliant (SRM File Transfer functionalities).

Conclusion

- We have presented StoRM, a solution that fulfils the requirements about space management and I/O performance in disk-intensive Grid applications.
- StoRM prototype have shown how to create and use reserved space in efficient mode.
- There is a lot of work to do, but we are confident in a success.

End

- Project references can be found at INFNForge :

  http://infnforge.cnaf.infn.it