Grid y Computación
de Altas Prestaciones
GRyCAP

DFSgc

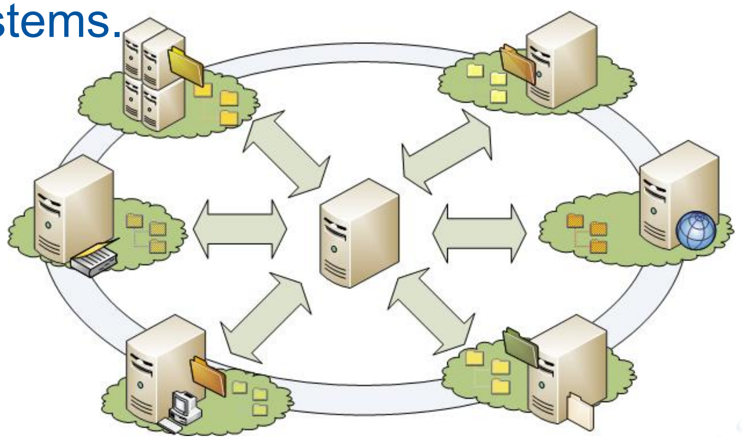
Distributed File System for Multipurpose Grid Applications and Cloud Computing

UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

ITACA
INSTITUTO DE APLICACIONES DE LAS TECNOLOGÍAS DE LA INFORMACIÓN Y DE LAS COMUNICACIONES AVANZADAS

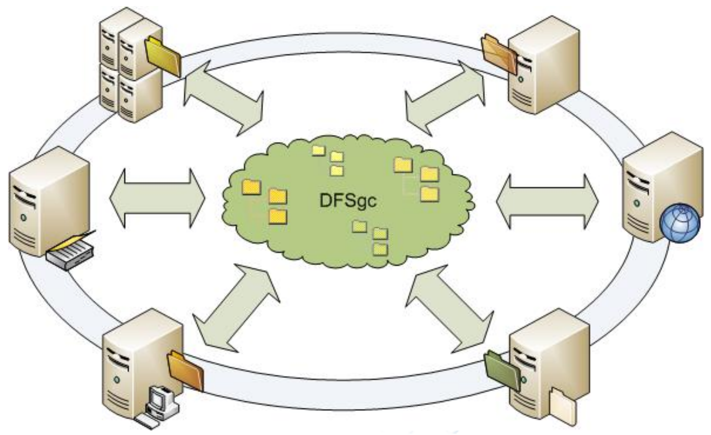
Introduction to DFSgc.

- **Motivation:** Grid Computing currently needs support for managing huge quantities of storage.
- **Lacks of current systems:**
  - Locality of storage: replicas are far from the consumer application. So they invest time in network transferences.
  - Addressing: the applications need to know where are their files placed.
  - Centralization: a central server manages the entire catalogue of files .
  - Partial functionality: storage systems are implemented for a specific purpose and so it is difficult to adapt to multipurpose systems.



Introduction to DFSgc.

- **Features of DFSgc:**
  - Virtualization: there is a global namespace, thus hiding physical locations.
  - Abstraction: simple interfaces, operations like a local storage system.
  - Distributed: storage and catalog systems are distributed.
  - Intelligent storage: data files are closer to where they are needed.
  - Replication: synchronous and asynchronous replication.
  - Coherence: the replicas of a file are always updated in real time.
  - Fault tolerance: the whole catalog is recovered when a failure occur.



Distributed File System for Multipurpose Aplications Grid and Cloud Computing: DFSgc

Some approaches.

File System	Distributed catalogue	Replication support	Fault Tolerant	Grid oriented	Replica consistency	Deterministic discovery	Heuristic Storage
LCG File Catalog	Each VO has a local catalog	Yes	Middle	Yes	Yes	Yes	No
Replica Location Service, Data Replication Service Globus.	LRC, indexed by RLI	Yes	Middle	Yes	No	No. multiple RLI	No
Hadoop Distributed File System.	No	Yes	Low secondary server	No	Yes	Yes	No
Microsoft's DFS.	No	Yes	Low Centralization	No	Yes	Yes	No
Past, peer-to-peer storage systems.	Yes	Yes	High	No	Yes	Yes	No

Distributed File System for Multipurpose Aplications Grid and Cloud Computing: DFSgc

Description of the Design.

- **Architecture DFSgc.**
  - Interface Layer:

    - A simple API enables Grid applications to manage files as a local storage file system.

    - The applications know the name of a file, a global namespace virtualizes the effective locations.
  - Logic Layer:

    - Keeps updated replicas (in real time) to ensure a coherent replication system.

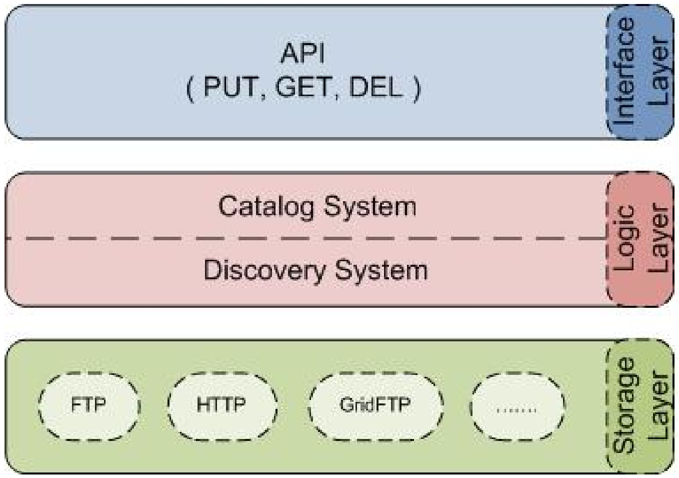
    - Applies heuristics methods to take files near to the applications.

    - Manages a distributed catalog system independent of the underlying storage system.

    - A discovery system is used to find metadata information about a file.
  - Storage Layer:

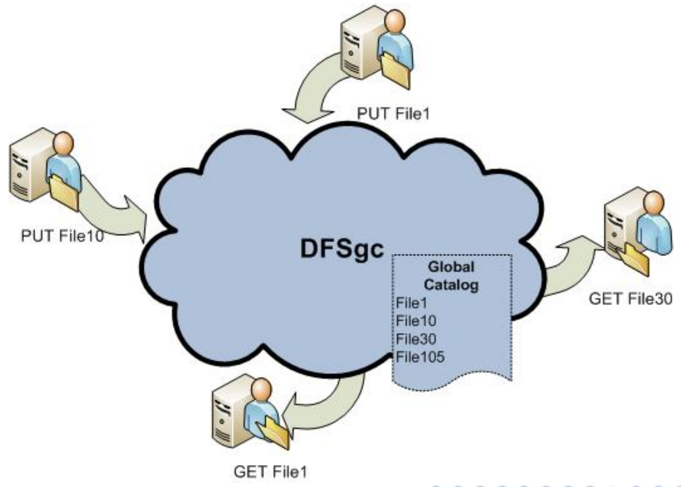
    - Abstracts the storage of a file, whether it is remote or local storage.

    - Decides which transfer protocol will be used by the applications (ftp, http, etc).



Interface Level.

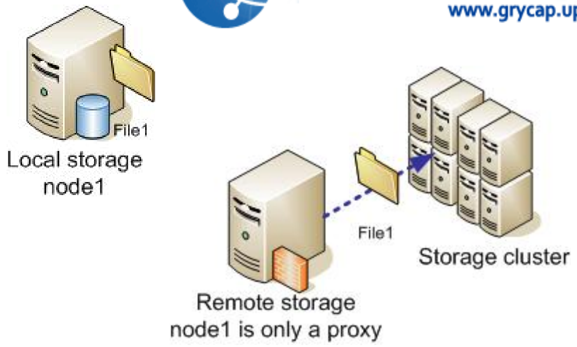
- **API and Namespace.**
  - Specifies three basic operations to manage files by users (PUT, GET, DEL)
  - The Grid applications operates directly with known nodes near of them.
  - Applications are able to decide as much replicas as needed, but DFSgc establishes where to effectively store the files in the system (according to several policies).
  - A single global namespace specifies the way as the files are named.
    - Names for File1: root/File1 or Dom1.SubDom1.File1 or FA3XSD.



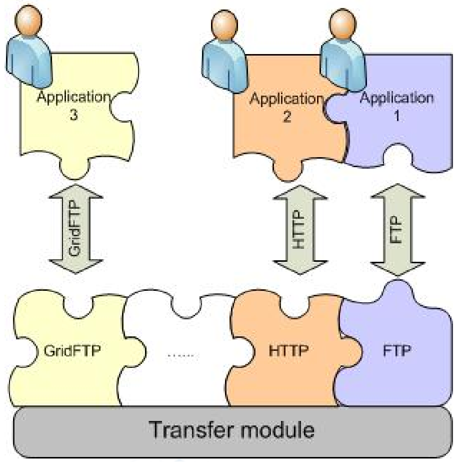
Distributed File System for Multipurpose Aplications Grid and Cloud Computing: DFSgc

Storage Level.

- **Storage.**
  - This layer abstracts the storage infrastructure.
  - The files can be stored in local file systems or in remote servers.



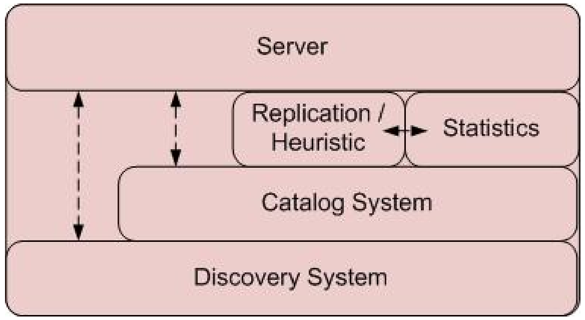
- **Transfer.**
  - It is possible to use several standard transference protocols (ftp, http, Gridftp).
  - DFSgc enables a Plug-in-like model, so that others transference protocols may be added.
  - The application and DFSgc decide which protocol will be used.



Logic Level.

▪Components of DFSgc nodes

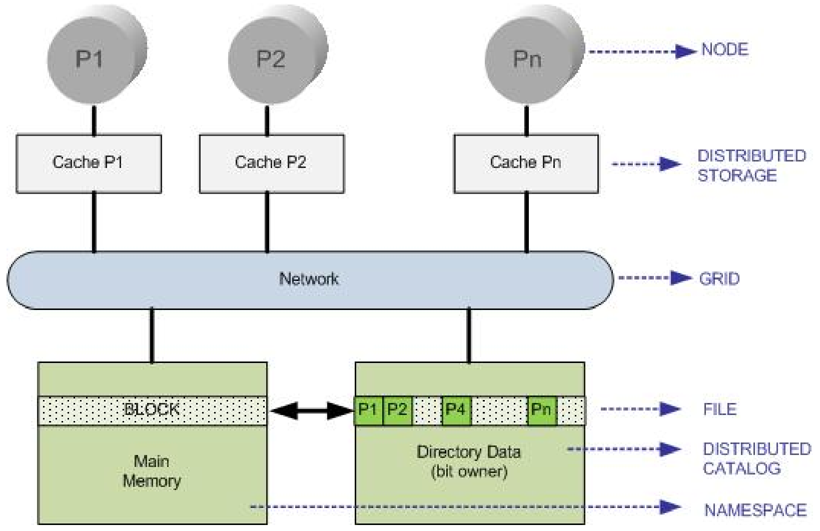
- **Server:**
  - Waits for new requests and coordinates the rest of modules to serve them.
- **Replication:**
  - Decides the number and the correct location to put a file using a heuristic model.
- **Statistics:**
  - Stores historic information about accesses to the files.
- **Catalog System:**
  - Keeps the entire metadata information for each file in the global catalog (replica locations, valid stored files, etc).
- **Discovery System:**
  - If a file exist then the discovery system returns its owner node.



Distributed File System for Multipurpose Aplications Grid and Cloud Computing: DFSgc

Logic Level. Replication System.

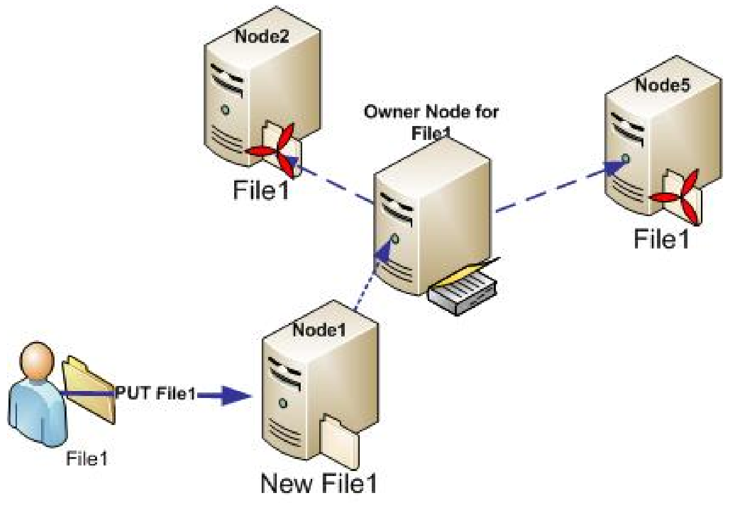
- **Motivation:**
  - The files may be near of the applications → replicas scattered throughout the system.
- **Problem:**
  - Multiple access points → replication system could make appear incoherent copies .
- **Solution:**
  - A distributed file system is similar to a multiprocessing system.
  - DFSgc applies cache coherence techniques to ensure the integrity of the replicas.
  - The file system is considered as a global memory, which is formed by the aggregation of partial file systems.

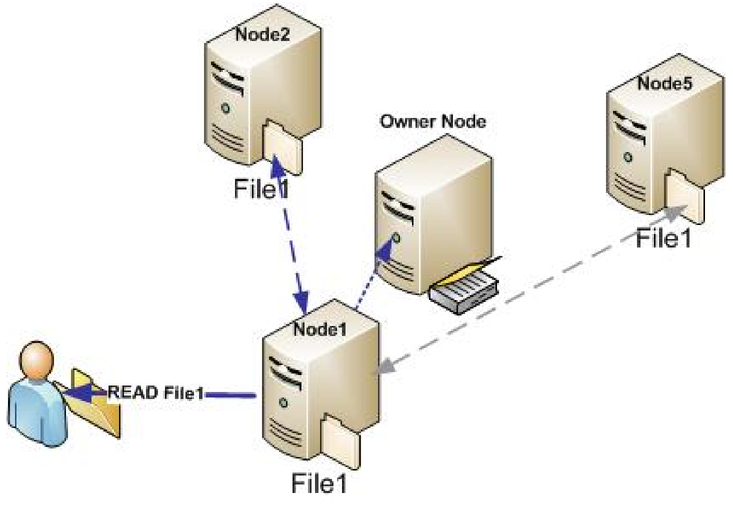


Distributed File System for Multipurpose Aplications Grid and Cloud Computing: DFSgc

Logic Level. Coherence Replication.

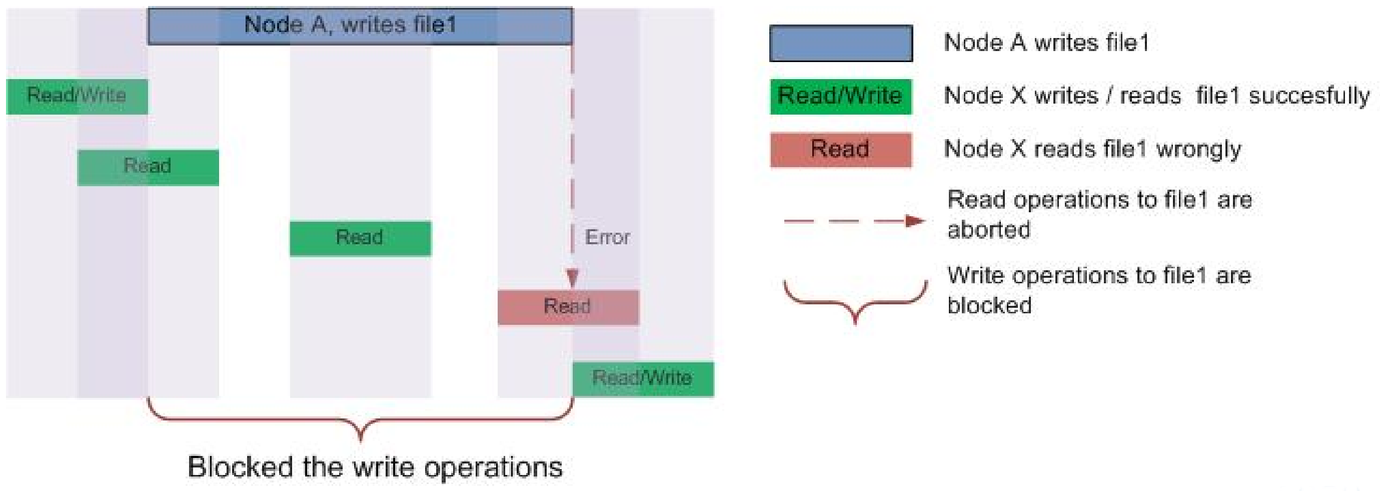
- **Replication system, directory-based coherence:**
  - Any file belongs to a single owner node which manages its metadata information.
  - Write operations invalidate any copies of the file.
  - Read operations create new replicas in the storage system and the file is returned to the application.
  - “Dirty” Bit avoids unnecessary transferences.





Logic Level. Multiple Access to a File.

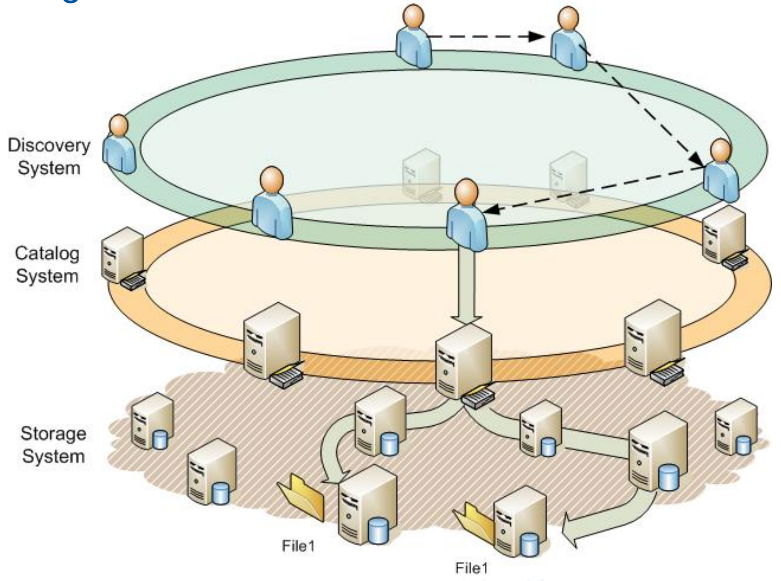
- **Concurrent access to files:**
  - Applications access concurrently to the same file from distinct copies which are distributed among the network.
  - While an application writes a file, DFSgc blocks the writing for the rest of replicas.
  - When a write operation is successfully completed the current reading operations are cancelled. READ operations are allowed in any other interval of time.


Node A, writes file1
Read/Write
Read
Read
Error
Read
Read/Write
Blocked the write operations
Node A writes file1
Read/Write
Node X writes / reads file1 succesfully
Read
Node X reads file1 wrongly
Read operations to file1 are aborted
Write operations to file1 are blocked

Distributed File System for Multipurpose Aplications Grid and Cloud Computing: DFSgc

Logic Level. Catalog and Discovery System.

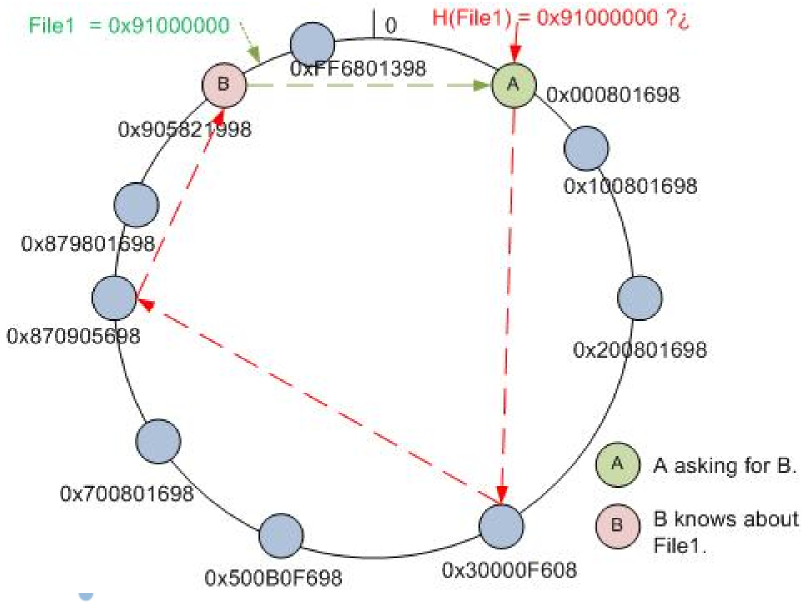
- **Motivation:**
  - Does anyone know anything about a file? → Discovery system to find the owner file.
  - Where are the replicas of the file? → Catalogue system to manage the replica locations.
- **Problem:**
  - If a file exist, the discovery system must return the owner node (deterministic search).
- **Solution:**
  - Discovery system uses a peer to peer model to distribute the catalogue → Distributed Hash Tables.
  - Catalogue system is based on a peer to peer model → High tolerance to fault.
  - physic location ≠ catalogue location.

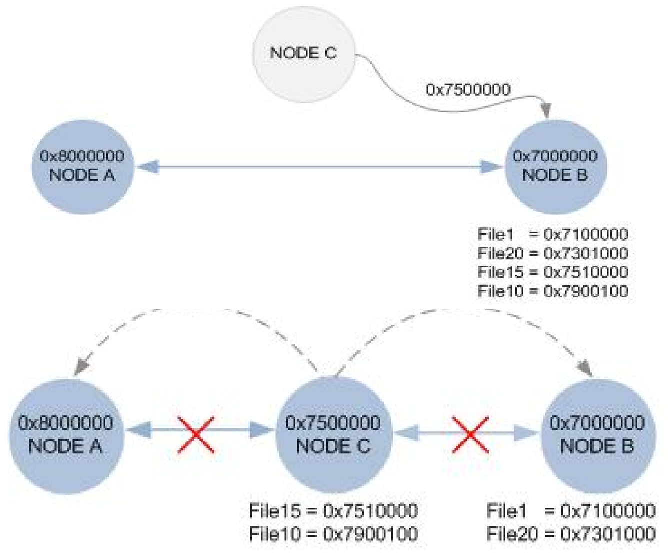


Distributed File System for Multipurpose Aplications Grid and Cloud Computing: DFSgc

Logic Level. Discovery System.

- **Discovery system:**
  - There are several peer to peer discovery systems: Pastry, Chord, Tapestry, etc.
  - Each node has a unique identifier. Hash(IP_node1) = 851243.
  - Each file applies a hash function to generate his own identifier. Hash(File1) = 86000.
  - Nodes are ordered from minor to major identifiers → Ring topology (enables deterministic search and stability to the topology).
  - Every file belongs to the nearest node (owner node). The node manages the metadata for the file.
  - Every node knows its nearest neighbors → deterministic search.

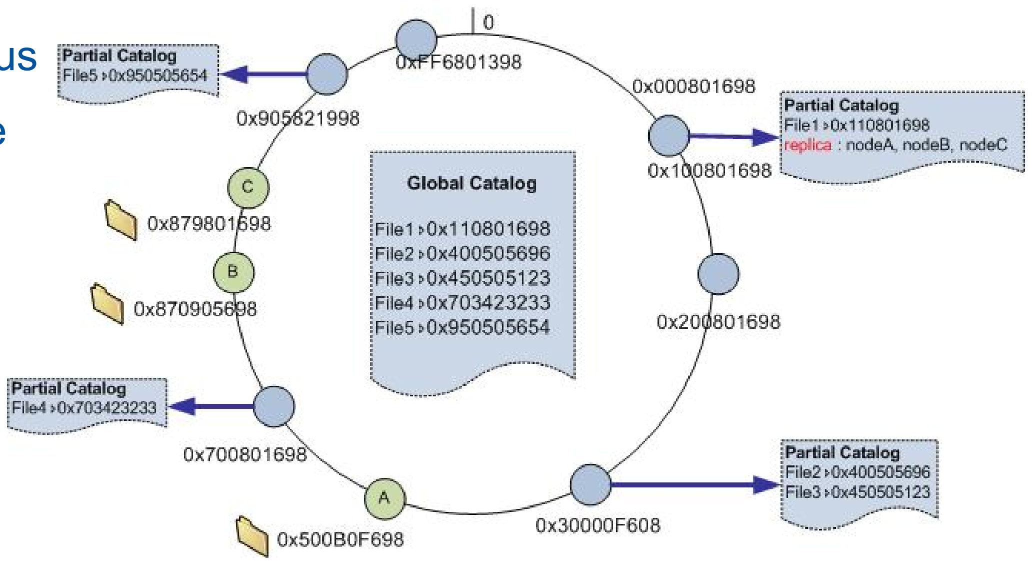




Distributed File System for Multipurpose Aplications Grid and Cloud Computing: DFSgc

Logic Level. Catalog System.

- **Catalogue System:**
  - Every node manages a partial portion of the global catalog (owner node).
  - A partial catalogue stores metadata about managed files: replica location.
  - It manages extra data for maintaining coherence of files: invalidation, dirty and block bits.
  - Any owner node makes synchronous backup copies of neighbors catalogs to increase failure tolerance.
  - Hash function enables that contiguous nodes are not necessarily in the same subnet. It also increases failure tolerance.



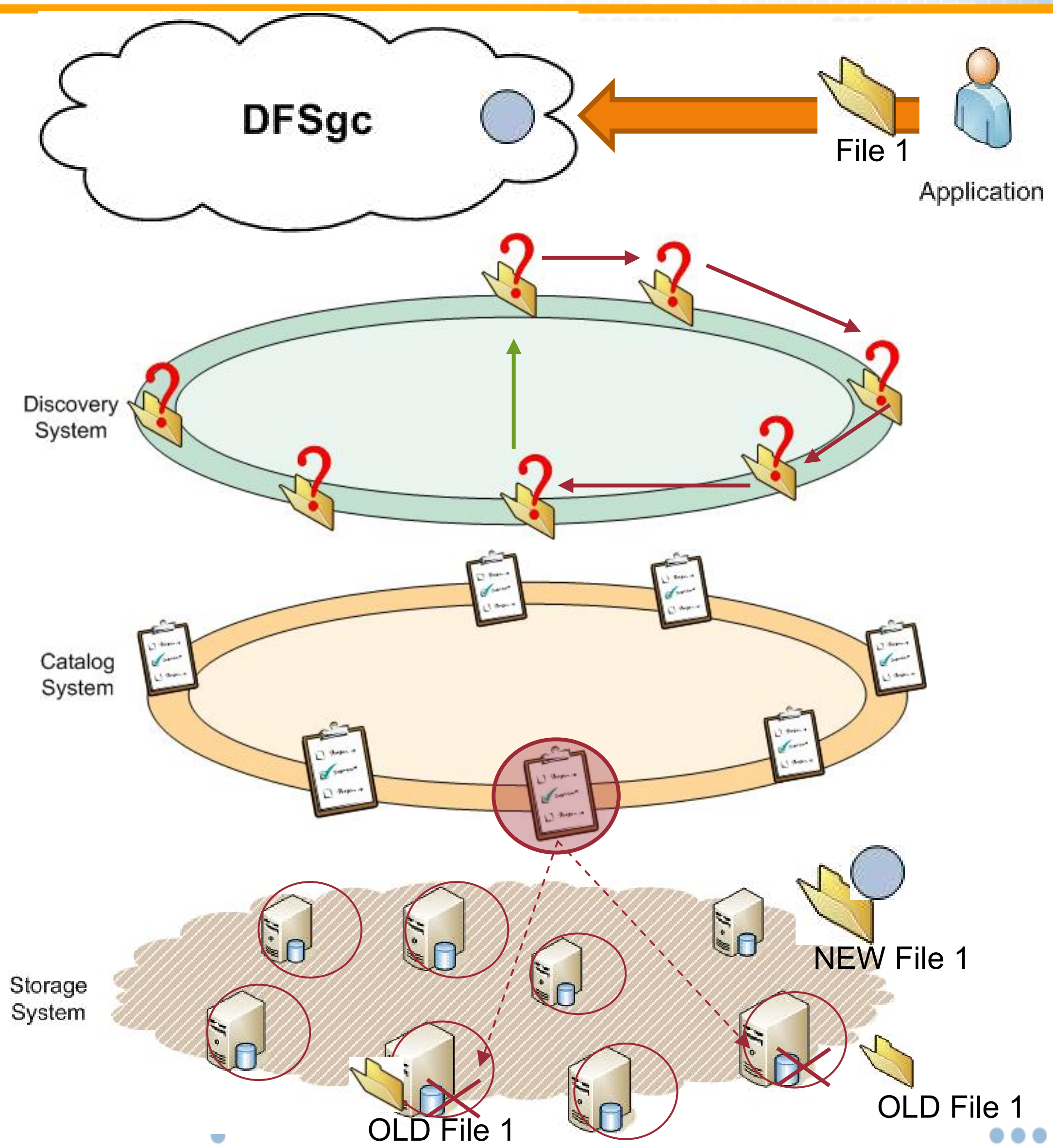
Distributed File System for Multipurpose Aplications Grid and Cloud Computing: DFSgc

Logic Level. Heuristic and Statistic Module.

- **Statistics module:**
  - Every node registers historic information about the access to the files.
  - The replication system uses this information to infer the best location of the replicas in order to make the copies.
  - The server is able to use distinct heuristic methods depending to the access to the files (distance in the network, redundancy, etc.)

Distributed File System for Multipurpose Aplications Grid and Cloud Computing: DFSgc

Use Case – Writing a File.



An application makes PUT operation in a known node (node1) of DFSgc.
The Application does not take care about anything else.

The node1 needs to find the owner node for File1, so it generates the file identifier and start the searching with your neighbour.
Hash(File1) = 100

The owner node manages the catalog of File1. Node1 writes File1 in the storage system while the owner node blocks new uploads of File1.

When node1 writes File1 successfully then the owner node invalidates the rest of replicas and it unblocks File1.

Conclusions and Future Works.

- **Conclusions:**
  - DFSgc is a robust storage system that uses peer to peer techniques and Grid catalogues. It may be used in Grid and ubiquitous systems.

- **Future Works:**
  - We have a first implementation but we are doing …

    Better performances.

    News heuristic models for the replication system.

    Security module.

    ….

Distributed File System for Multipurpose Aplications Grid and Cloud Computing: DFSgc

Thank you !!.