|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **HDFS(**Hadoop Distributed File System under Apache licence 2.0 developed by the  Apache Software Foundation**)** | **GLUSTERFS**  **(**open source (GPL) distributed file system developed by the gluster  core team.**)** | **LUSTER (**available for Linux and is under GPL licence.**)** | **CEPH**(open source (LGPL) distributed file system developed by Sage Weil.) |
| **ARCHITECTURE** | Centralized architecture, metadata stored in namenode, data split into blocks, distributed and replicated in datanodes. Secondary namenode is a persistent copy | Decentralized, It has a client-server design in which there is no metadata server. Instead, GlusterFS stores data and metadata on several devices attached to different servers. | Centralized, It stores metadata on a shared storage called Metadata Target (MDT) attached to two Metadata Servers (MDS). Data is split into objects and distributed at several shared Object Storage Target (OST) which can be attached to several Object Storage Servers (OSS). | Distributed, provides a dynamic distributed metadata management using a metadata cluster (MDS) and stores data and metadata in Object Storage Devices (OSD). |
| **NAMING** | INDEXING, inodes hold the metadata, namenode manages namespace and metadata, namenode performs mapping between filename and file block | Locates files algorithmically using the Elastic Hashing Algorithm(EHA) | INDEXING.  The Lustre's single global name space is provided to user by the MDS. Lustre uses inodes. | Data are striped into blocks which are assigned to objects. These objects are identified by the same inode number plus an object number, and placed on storage devices using a function called CRUSH. |
| **API AND CLIENT ACCESS** | REST, FUSE (filesystem in userspace). | REST and mount | FUSE, first ask the MDS to locate data and then directly perform I/O with the appropriate OSTs. | REST and kernel mount |
| **FAULT DETECTION** | Nodes are fully connected. Every three seconds, data nodes send heartbeats to the namenode to confirm their  availability. | When a server becomes unavailable, it is removed from the system and no I/O operations to it can be perform. | Faults are detected during client's' operations (metadata queries or data transfers) instead of being detected during servers' communications. | Nodes are fully connected  and CEPH uses heartbeats to check the nodes. |
| **CACHE CONSISTENCY** | WORM(write only read many) when a file is created, data written to it, and then the file is closed, it cannot be modified anymore. | Does not use client side caching | Lustre implements a Distributed Lock Manager (DLM) to manage cache consistency. This is a sophisticated lock mechanism which allows Lustre to support concurrent read and write access. | Reads must reflect an up-to-date version of data and writes must reflect a particular order of occurrences. It allows concurrent read and write accesses using a simple locking mechanism called capabilities. |
| **REPLICATION** | Asynchronous | Synchronous | Lustre does not ensure replication. | Synchronous |
| **LOAD BALANCING** | HDFS defines the utilisation of a server (or cluster) as the ratio of the space used to the  total capacity of the server (or cluster), and fixes a threshold value in the range of (0,1). It is automatic | GlusterFS affects each  logical storage to a physical one according to the used disk space.  It is manual | Lustre provides simple tools for load balancing. When an OST is unbalanced, that is, it uses more space disk than other OSTs, the MDS will choose OSTs with more free space to store new data. | Firstly, it implements a  counter for each metadata which allows it to know their access frequency. Secondly, Ceph uses weighted devices. |

* Decentralised architectures seem to scale better than a centralised one, because of the distributed workload management.
* For performance, an asynchronous replication and the use of an index to maintain the namespace are preferable whereas a decentralised architecture is better for managing large amounts of data and requests.

**CRITERION FOR SELECTION:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **HDFS** | **GLUSTERFS** | **LUSTER** | **CEPH** |
| **SCALABILITY** | suitable to store small quantity of big files | both small  and big data. | suitable to store small quantity of big files | both small  and big data. Dynamic adjustment of namespace |
| **ACCESS TRANSPARENCY** | Indexing for namespaces, easy to maintain | use an algorithm to calculate data's location, reduces metadata server load | Indexing for namespaces, easy to maintain | use an algorithm to calculate data's location, reduces metadata server load |
| **FAULT DETECTION** | fully connected | detected | manual | fully connected |
| **FAULT TOLERANCE** | Asynchronous replication, | Synchronous replication | no replication | Synchronous replication |