Interview questions - **File Systems and Distributed Storage**

---

1. What is the difference between a file system and a distributed file system?

Answer:
A traditional file system (like ext4 or NTFS) manages files on a single physical disk or volume. It's limited by the storage and processing capacity of a single machine.

A **distributed file system (DFS)** like HDFS or CephFS, on the other hand, spreads data across multiple machines (nodes), providing:

- Scalability
- Fault tolerance
- High availability
  It allows multiple clients to access files as if they were local, even though data is distributed.

---

2. How does HDFS ensure fault tolerance and reliability?

Answer:
HDFS ensures fault tolerance through **replication**:

- Each file is split into blocks (default: 128MB or 256MB).
- Each block is replicated (default: 3 copies) across different DataNodes.
- If a node fails, HDFS can retrieve the file from other replicas.
- The **NameNode** keeps track of block locations but not the data itself.

This design ensures data remains accessible even if nodes crash.

---

3. What roles do NameNode and DataNodes play in distributed file systems?

Answer:

- **NameNode**: The central metadata server. It stores information about file hierarchy, block locations, and replication. It does not store actual file data.

- **DataNodes**: These store the actual data blocks. They periodically send heartbeat signals and block reports to the NameNode.

Together, they form the core of systems like HDFS, with NameNode as the coordinator and DataNodes as the storage workhorses.

---

4. Explain the trade-offs between latency and throughput in distributed storage systems.

Answer:

- **Latency** is the time it takes to retrieve or write a piece of data. In distributed systems, latency may be higher due to:
    - Network hops
    - Coordination overhead (e.g., NameNode communication)
- **Throughput** is the total amount of data processed over time. Distributed systems are optimized for **high throughput** with parallel reads/writes.

Trade-off:
Optimizing for throughput (e.g., in big data analytics) may increase latency for small, frequent reads (OLTP). Choosing the right system depends on the workload.

---

5. In what scenarios would you use CephFS or GlusterFS over HDFS?

Answer:

- **CephFS** is suitable when you need:
    - POSIX-compliant file system access
    - Fine-grained metadata control
    - Integration with object/block storage (Ceph is unified)

- **GlusterFS** is chosen for:
    - Simpler setup with commodity hardware
    - Scale-out file storage with minimal configuration
    - Use cases like serving media files or shared volumes in container environments
- Use **HDFS** when you're primarily focused on batch analytics workloads (e.g., Hadoop, Spark) with write-once-read-many patterns.

---

6. How would you handle scaling storage in a high-throughput analytics system?

Answer:

- Use a distributed file system like **HDFS or CephFS**.
- **Partition and replicate data** to distribute load and prevent bottlenecks.
- **Scale horizontally** by adding new DataNodes.
- Implement **data locality** awareness—run compute jobs close to where the data resides.
- Use **compression and efficient file formats** (e.g., Parquet, ORC) to reduce I/O.

Also, monitor usage patterns and employ auto-scaling techniques where supported.