InfiniStor:

THE Most Feature-Rich Scale-Out NAS Storage System with Cloud Storage Services in Mind

WHITE PAPER

2013. December

1. Introduction

The value proposition of InfiniStor Storage System is that it is highly scalable storage system while providing advanced storage features that are essential for enterprise NAS and Cloud Services. The system provides an abstracted unified view to a collection of multiple, generally heterogeneous, Storage Services and DISK subsystems. This unified view can be accessed through various Remote File Protocols. Internally the system manages files data and metadata in a separated but fully distributed manner. InfiniStor Storage System is designed with the following major goals in mind:

- Linear or Hyper Linear Scalability in Capacity and Performance
- High Availability
- Parallel Access
- Adaptive Behavior
- Automatic Configuration
- Feature-Rich
- Cloud Services

2. Components

InfiniStor File System, short for IFS, is the heart of InfiniStor Storage System. IFS is distributed file system consists of a collection of multiple instances of loosely coupled components or modules, each carrying its own functionally: the IFS Client is responsible for the client communication, the Metadata Server, or MDS, handles all the Client requests and issues, and the IFS SS handles the File System or DISK I/O. Special Directory Server, or DIR, is used to distribute file system user directories among the MDSs and their DISKs. Separate Management Server, or MGS, is used to manage and monitor entire systems.

2.1 IFS Client

The InfiniStor Storage System provides POSIX compliance OS Client Modules. Currently, InfiniStor provides Client Modules for all versions of MS-Windows (Windows Vista or later) and Linux (kernel 2.6.9 or higher). The Client communicates with MDS (and DIR) and IFS Storage Servers to request and communicate IO to and from IFS Storage Servers. In

addition to the POSIX compliance Client Modules, InfiniStor Storage System also provides NFS, CIFS, and REST-HTTP Gateway Services. These Gateway Services may be used with any combinations and any number of Gateways.

IFS provides software development kit for users who want to develop their own applications or services that requires features of InfiniStor built-in to their products. SDK is available in Application Programming Interface (API) format. Currently, JAVA, C in Linux OS, C in MS-Windows, .NET in MS-Windows, and Python in Linux and MS-Windows. All programming language bindings behave identical regardless of languages or Operating Systems. Application or service developers wanting to use InfiniStor as a default backend storage system may use default IFS POSIX Client mounted directory or use one of IFS Client SDKs. For instance, InfiniStor REST Services called “InfiniStor Cloud Storage Service Platform” is developed using Linux C library. Figure 1 describes what kinds of access method IFS provides and their uses.

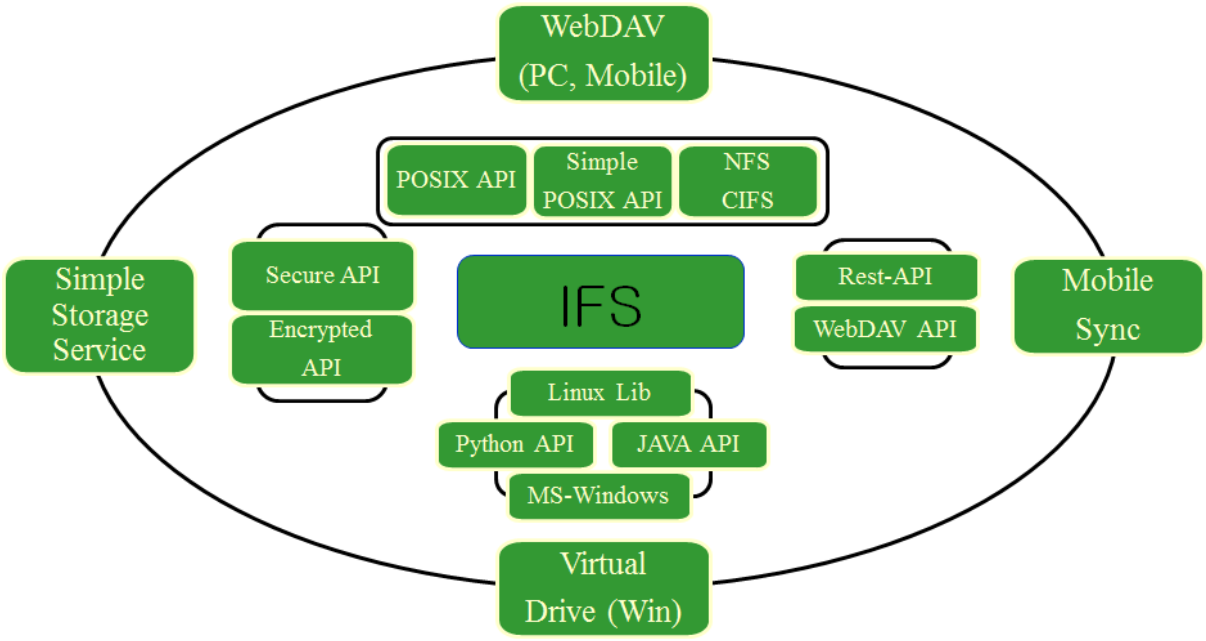


Figure 1. InfiniStor Client Access Methods

2.2 Metadata Services with Directory Services

The Metadata Server Service, MDS, implements all the file system services and additional advanced features on a single group of processes. The MDS Service contains many processes including Metadata Service, Multiple Metadata Server management and directory load balancing, Logging, Metering, Advanced Feature Execution, and more. Local Inter-Module calls within the same MDS are handled as local procedure calls with minimal

overhead. Since Multiple Metadata Servers involved, separate DIR services handle distribution of user directories among the MDS Servers. IFS Client IO requests are handled in Metadata Service Handler only. All operations are non-preemptive threads to simplify the synchronization considerably, and enhance the performance by reducing unnecessary context switch overhead to a minimum.

To take advantage of SMP architecture, multiple instances of MDS Service Threads are created in all Metadata Servers, so that each one serves a different subdomain, and runs almost exclusively on a single CPU Core. As a result, there's almost no overhead on kernel context switching. Figure 2 describes the IFS MDS programs and how they interact with each other.

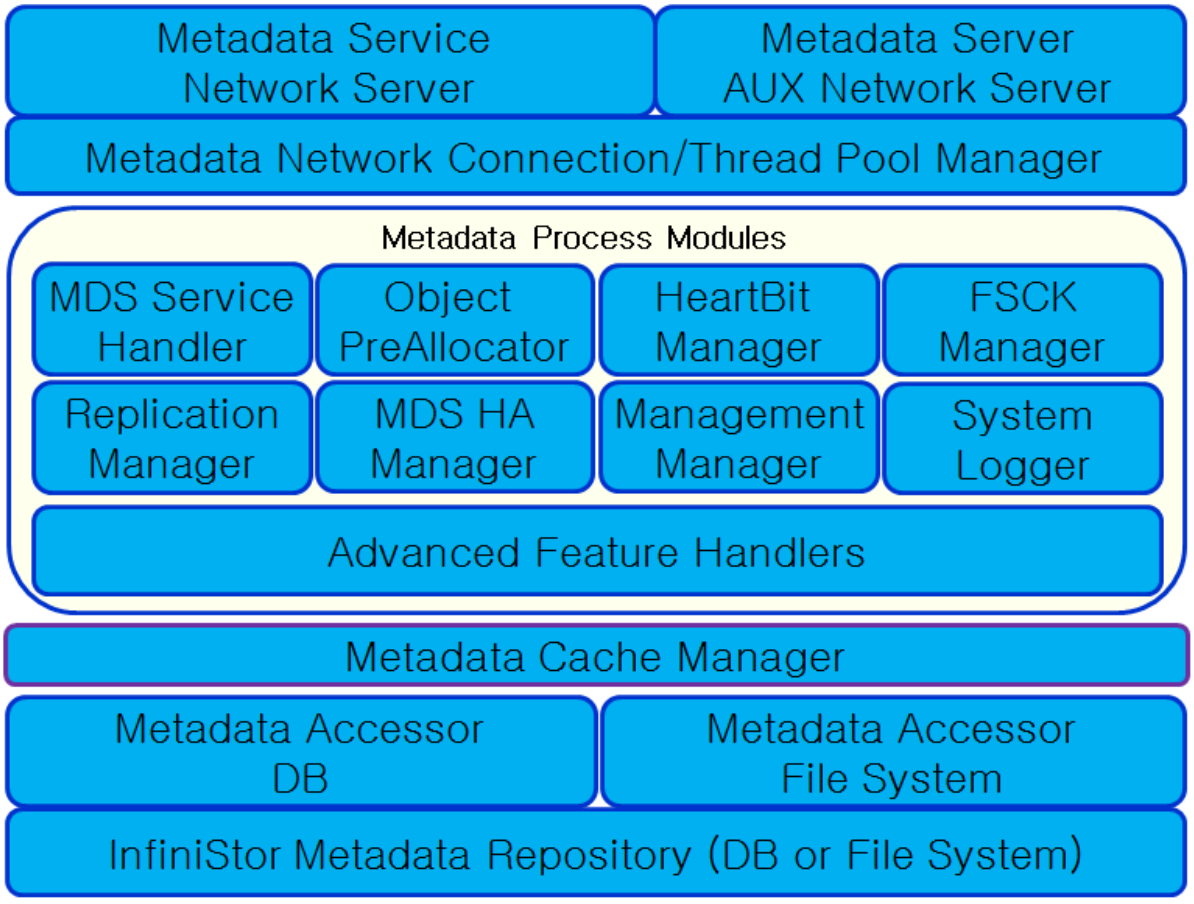


Figure 2. Metadata Services

그림 다시 그릴 것 (DIR 추가)

The following are major components of Metadata Services.

DIR Services

DIR Service is directory or user path distribution services. In IFS3, each user directory gets

its own ID called DIRID. This DIRID is 8 Hexadecimal digit ID and is unique in a volume. DIR Service is responsible for creating / querying / removing DIRIDs for user request. The most frequently and important use of the DIR Service is query. Basically, user requests the location of the PATHNAME in one of the DISKs in one of the MDS Servers. That location contains metadata for PATHs and files in the PATHNAME

The user (mostly IFS3 Client) request DIRID Info for a given "PATHNAME" in String format. Then, DIR Service returns DIRID_INFO that contains MDS_IP, MDSDISK_ID and DIR_ID. MDS_IP is IP of MDS Server and MDSDISK_ID is ID of DISK in MDS that contains the PATH location. Then the DIR_ID is actual path that contains all the files and directories of "PATHNAME" path. DIR Service is an independent service that may be running in an independent server or may reside in other IFS3 service server. For large systems, it should run in its own machine. However, for most cases, DIR Service may run in one of MDS Servers.

IFS3 DIR Service caches most of the recently accessed DIR Info in order the improve the performance of the DIR Service. Also, libifs caches roughly 10,000 DIR Info locally to improve the DIR lookup services.

Metadata Service or Metadata Service Handler

The Metadata server is a plain repository for file metadata. File metadata consists of 2 parts. The first part is file attributes information. File attributes are information regarding files and directories. Attributes include, but not limited to, name, size, type, ownership, permission, inode number, checksum value, and other more. The other part is location of the Object that contains the contents of the file.

Major role of Metadata service is handling POSIX or POSIX like file request and handling the processes. IFS3 is fully POSIX compliance file system. Thus, IFS3 Metadata Service is the brain of IFS3 that handles all of request handling from libifs, FSCK, and other IFS management tools.

Then libifs request FSTAT, querying for file Attribute, MDS Service would look for the file Metadata Info in the attribute part of the Metadata then returns the attributes information to libifs. When libifs request for file READ, MDS Service should look for the Object_INFO part of the metadata and return the Object_INFO to the client. Then, libifs should use the Object_INFO to find where the Object are located and communicate with SS Services for READING contents of the file/

In IFS3, Metadata information may be stored in flat-file, database, or in cache system. It is flexible that some combination of difference repository methods may be combined.

MGS Services

MGS Service is IFS3 Management service. MGS handles 3 main roles. The first is IFS3 System Configuration Management, call it MGS_Conf. MGS_Conf configures and manages DIR Server, MDS Servers, SS Servers, DISKs in MDS, DISKs in SS, and all connected libifs clients. MGS_Conf provides management tools and methods to add/remove MDS / SS Server and DISKs in the servers.

The second role of MGS is running and managing HeartBit server. The details of HeartBit Service are described later.

The third role is managing services. Managing Service process includes start/stop/pause MDS / SS / DIR Services and DISKs associated with them. Also, IFS_Util management tools are provided to manage IFS3 easily. Such tools include monitoring DISK performance, Network IO Performance, Moving Objects from one DISK other other DISK, create user and others.

Several IFS3 programs use DBMS such as MySQL, SQLLite, or others. MGS does not manage nor monitor the status of DBMSs.

FSCK

FSCK is a process of ensuring the correctness of the file metadata and the chuck files associated with the metadata information.

FSCK process requires being run when certain conditions are met. Such conditions include, but not limited to, after rebooting one or more Storage Servers are restarted, after metadata server is restarted, after system failure, and when system administrators wish. FSCK process may run while the InfiniStor system is operating.

Replication Manager

Replication manager is a process of creating CHUNK replicas. Replication occurs when a file (or CHUNK) is created or modified.

However, IFS3 provide replication in real time. It means as libifs write data to Object, the portion of the data is written to its replica Object at the same time. Therefore, Replication Manager seems is not necessary. However, Replication Manager is necessary when IFS3 and FSCK recovers the Objects from the failed DISKs.

DR-Sync Manager

DR-Sync manager is a process of managing 2 or more InfiniStor Storage Systems always sync in near real-time. DR-Sync is not mandatory configuration. However, DR-Sync is usually configured for the systems that require non-stop storage services in the event of an

entire IFS3 system down or inaccessible.

HA-Manager

HA-Manager is a process 1) making sure Metadata is stored in multiple MDS DISKs, 2) providing MDS Service continuously. In IFS3, all Metadata in a MDSDISK is replicated in other MDSDISK which is in StandBy Mode. When update in Metadata happens, same update should be applied to corresponding StandBy MDSDISK. With HA-Manager configuration, IFS3 HA-Manager should detect failure of a MDS Server or DISK, then migrate the service to other corresponding MDS Server or DISK. HA-Manager configuration is not required as it requires extra servers. However, to prepare from the event of catastrophic failure in a main Metadata server, it is highly recommended to configure the system with Metadata server HA.

HeartBit Manager

HeartBit Manager is a server process that receives heart bit messages from all Storage Servers. Along with heart bit message, Storage Server's performance matric such as Network IO, DISK IO, CPU & Memory Usages, and more. HeartBit information is closely related to many other managers in Metadata and Management Manager as well.

When HeartBit Checker detects any malfunctions in any components in IFS3, it should provide necessary actions. For example, if BeartBit detects one of the SS fails, it will report to MGS_Conf about the status. Then MGS_Conf will notify to all MDS to stop allocating further Objects from the DISKs in the failed SS. Furthermore, it will execute necessary FSCK process to recover Objects that were stored in the filed DISKs.

Metadata Cache

Metadata Cache service is a layer of service that contains much file Metadata Information in Memory. Thus Metadata Service Handler does not have look up the actual Metadata file to retrieve file meta information. The layer does not define how file meta information to hold in cache. However, each file meta information requires less than 512Bytes. Thus, it can cache upto 1 Million files of metadata information per 1GB of main memory.

Later version of InfiniStor that will have upto 16 clustered Metadata Servers will be able to share metadata cache information among the 16 Metadata Servers. Such configuration can cache over 1 Billion files metadata information and resulting in close to 100% of all metadata service operations will be handled in cached data.

2.3 IFS Storage Server

The IFS Storage Server is responsible to STORE and READ data to and from a DISK Sub System. The DISK Sub System may be local hard disks, RAID Storage, iSCSI or FC connected storage, or even a NFS mounted volume.

The IFS3 Storage Server service works like low-level IO sub system over IP network. All IFS3 Client IO operations are done using the IFS3 Storage Server operations.

Each IFS3 Storage Server may contain more than 1 DISK of maybe different types. Each disk may be associated with a volume. A volume may have multiple DISK from 1 or more IFS3 Storage Servers.

One of the advanced features of InfiniStor is Encrypted Data Store to DISK. All encryption and decryption is done solely in Storage Server. IFS3 Storage Server uses AES-NI technology that is built-in latest Intel Xeon CPUs. With AES-NI, IFS3 does not suffer from performance penalties as it is using the native CPU instruction and IFS3 does not make extra memory copy operations.

IFS3 Storage Server understands Network status. When network operations seems behave abnormally, IFS3 takes precautionary action before it actually creates problems.

2.4 All Together

Figure 3 provides a general overview of the InfiniStor Storage System components in a set of 2 Metadata Servers, 4 Storage Servers, and various Client Gateways. The Metadata services running on a set of Metadata Servers with HA-Manager using HA Network and Service Network. Figure 3 shows how all components of InfiniStor Storage System interact with each other.

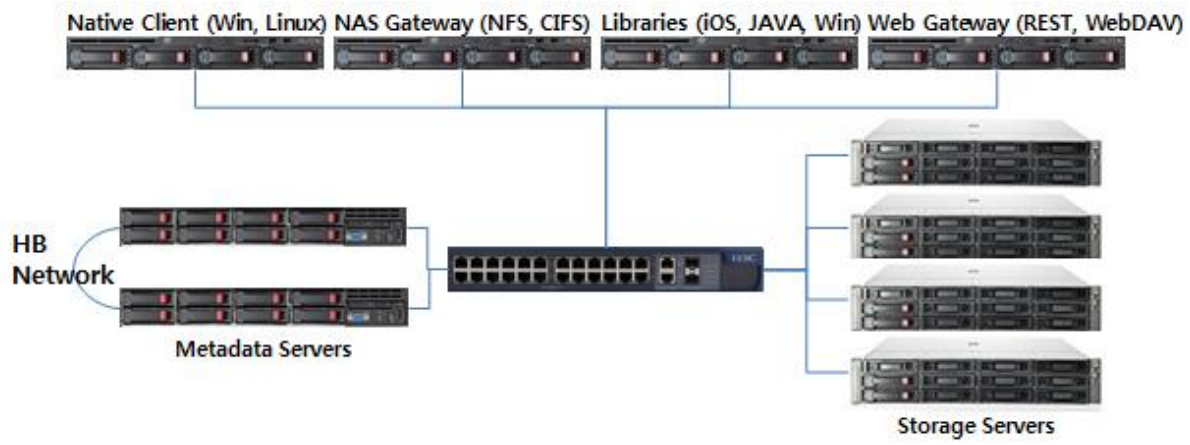
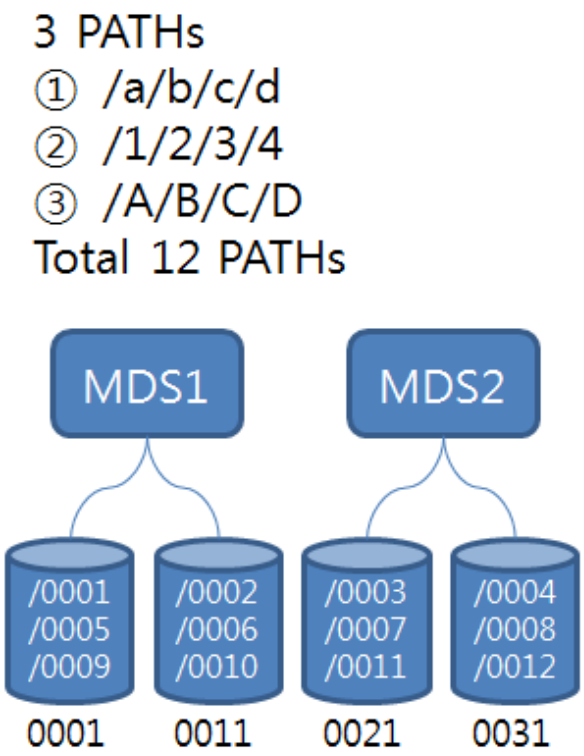


Figure 3. InfiniStor Communication among the Components

그림 다시 그릴 것 (DIR, MGS 추가), 설명도 다시

2.5 File and Object Placement

As most other file systems, the IFS3 has 2 kinds of user date information: PATH and File. In IFS3, PATHs information are stored and managed by DIR Service and Filess Information are stored in MDS Services. As described before, DIR Service handles where to create/store PATH information among the MDS Servers and MDS DISKs. After Client gets MDS IP, MDS DISKID and PATH's DIRID, the clients make request to the MDS to get the file Metadata information which is located in DIR ID in the MDS DISKID. Figure 4 describes how PATH Information is stored in DIR Service and in the MDS Servers.



How PATHs are stored in DIR Service

PATH Name	MDS DISKID	DIRID
/a	0001	0001
/a/b	0011	0002
a/b/c	0021	0003
/a/b/c/d	0031	0004
/1	0001	0005
/1/2	0011	0006
/1/2/3	0021	0007
/1/2/3/4	0031	0008
/A	0001	0009
/A/B	0011	0010
/A/B/C	0021	0011
/A/B/C/D	0031	0012

Figure 4: How PATH information is stored in MDS Servers

Note that DIR Service does not keep the MDS Server's IP Addresses. It is very possible case that MDS Server may change its IP Address. Then the entire DIR Information will be invalidated. Instead, DIR Service must keep a mapping table that translates MDS DISKID into MDS Server's IP Address and local DISKID. Normally, this mapping table is located in memory to improve the speed.

Other part of any file system is managing FILE information. The IFS3 System distributes files to a single or multiple Objects evenly to Storage Servers and DISKs. The size of an Object is user definable. This even distribution improves the balance among system components, hence improving performance and resource utilization, which leads to better

ROI and TCO. Like many other systems that stores a file to an object, we call uniform distribution, is easy and simple to implement. Traditional uniform distribution actually has some advantages such as backup and restores processes. However, even distribution method enables automatic load balancing in capacity as well as performance, reducing bottleneck points when a few files are very hot (very heavy access), and easy to recover in case of failure.

Also, unlike traditional uniform distribution or even distribution, IFS3's advanced technology delivers no slowdown restores or backup processes. In fact, the better use of cache technology enables not just linear performance scale-up, but Hyper Linear Scale-Up performance. Therefore, IFS3's File placement is much more superior than any other technologies on market today.

To prevent from IFS3 Storage Server failures, IFS create object replication via Replication Manager. The number of replica to create is system administrator definable as well as API definable. Individual DISK failures can be protected by DISK's RAID controller.

However, although unlikely, in the event of entire Storage Server failure, all Objects stored in the Storage Server may not be recoverable, resulting in permanent loss of files. In order to prevent from such a catastrophic event, IFS provides Object replica. Thus, even in the case of loss of Storage Server, IFS3 can protect all files from loss and can prevent from stopping storage services. In IFS3, Replication process is done in real time that when files are written for the first time or updated, the portion of the data is replicated to the replica at the same time. The replication process is done entirely by the Storage Servers. When Client detects the Storage Server that contains the Primary Object is failed, the Client continues write to its replica from the point that write process is failed.

2.6 Data Flow

2.6.1 Read Access

When an IFS3 Client issues a data READ request, it is first handled by the DIR Service. The READ request will provide PATH Name and File Name in the PATH. Client first will request to DIR Service with the PATH Name and gets which MDS Server and which MDS either from Metadata Cache Layer or from the Metadata information repository (most likes Metadata file or DB). Then, Metadata Service returns the file Metadata Information to the Client. The file Metadata contains all information about how the file is stored to which IFS Storage Servers.

Then, the Client receives the file Metadata and analyzes which IFS Storage Server stores

the part of file that the Client needs. Then the Client requests to the IFS Storage Server for the data it needs. The request contains name of object the IFS Storage Server has, offset, and the length of the data. The IFS Storage Server receives the request and validates the requests. If validation is successful, IFS Storage Server read from Cache or DISK and returns the data the Client request.

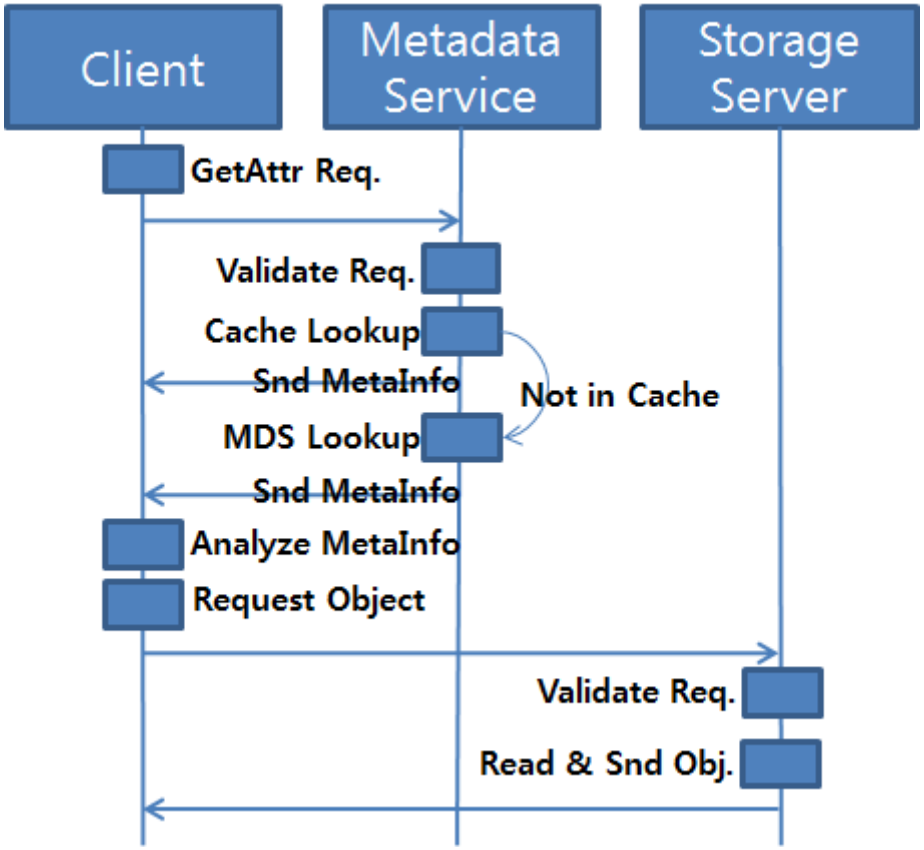


Figure 5: InfiniStor READ operation flow after DIR operations

2.6.2 Write access

File WRITE request is initiated by IFS3 Client libifs. After libifs gets PATH Information from DIR Services, libifs requests WRITE request to the Metadata Service handler of the MDS Server and MDS DISKID with FILE name. Then Metadata Service Handler creates initial metadata for the FILE and returns basic metadata information back to the libifs. The basic metadata information contains (but not limited to) to write Object information. The Object information contains 2 Storage Server's DISK ID where the Object (and its replica) should be stored, Object (and replica object) names to write FILE data. If IFS3 Client finishes writing the FILE, IFS3 Client send request to Metadata Service Handler for notifying the end of file

writing. Then, the Metadata Service Handler processes the post WRITE works. The post process includes (but not limited to) requesting 1) update file attributes information including, but not limited to, size and dates, 2) MDS HA-Manager for handling Metadata HA, 3) DR-Sync Manager for creating DR, 4) End to End CHECKSUM process to make sure if the write process has been successfully and correctly completed. This End-to-End Checksum process may be processes at other stages of the WRITE operation.

To improve the performance, IFS3 Metadata Service Handler prefetches Objects from the DISKs and IFS Storage Servers.

When IFS Storage Server receives data to store to the Object, IFS Storage Server may choose to store the data in Cache (or Main Memory) or to the DISK (in Sync or VFS Cached Sync). For the performance, IFS Storage Server stores the data in Cache. However, for safety, the data must be stored to the DISK in Sync mode. It is upto the system administrators which mode to choose. However, we strongly recommend using Sync mode DISK as the performance advantage of using Cache is not very big.

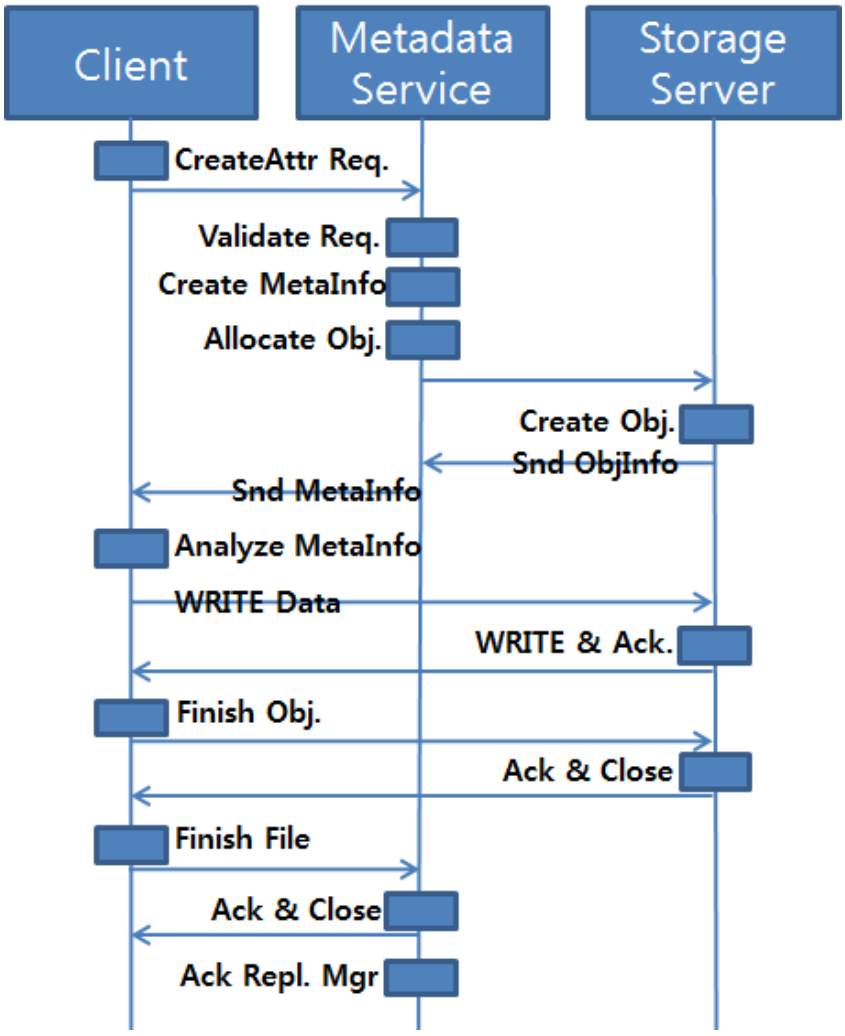


Figure 6: InfiniStor WRITE operation flow after DIR operations

2.7 Degraded Mode and Recovery

2.7.1 Degraded Mode Operation

IFS3 system can be turned to degraded mode for the number of reasons. To define degraded mode, it is state of IFS3 that the system cannot be running at the best possible conditions. Best possible condition means all hardware components including network, server, hard drives are running in good conditions. In addition to the hardware condition, there should be enough space left in the hard drives to store forthcoming data as objects.

IFS3 turns into Degraded mode when 1) hard drive space runs out of space, 2) hard drive fails, and 3) Storage Server is out of reach for some duration of time.

When IFS3 turns into Degraded mode, IFS3 does number of jobs to make the system normal again. When the system runs out of physical hard drive space on one or more (even all), IFS3 trys to execute capacity rebalance operations. Capacity rebalance operation is steps of operations of moving objects from more used hard drives to other less used hard drives. As a result, capacity rebalance will make the IFS3 Storage servers and DISKs evenly used systems and runs the overall system smoother. If the system is full even after Capacity rebalance, IFS3 turns itself into READ_REMOVE_ONLY mode. No more create or update operations will be permitted. Normally, IFS3 turns into degrade mode when the system is 95% full. However, the number is configurable by administrators.

If IFS3 turns into Degraded mode due to the failure of hard drive, IFS3 immediately executes FSCK for the hard drive and alert to all other Storage Servers about the state of the failed hard drive. Then, FSCK and Replication managers will recover all the objects that were stored in the failed hard drives.

If Storage Server becomes out of reach, it could mean a number of reasons. Server failure, network busy, or server is too busy to response. Depending on the cause of the heartbit failure, IFS3 should proceed with necessary procedures and the procedures are heavely dependent on the actual hardware configurations and user environment.

2.7.2 Recovery Operation

IFS3 ensures that the file system should provide storage services even in the Degraded mode as much as possible.

As described in 2.5, when a Storage Server fails, IFS3 ensures no loss of files and no stopping services using Object Replication features. Unlike some other distributed storage

systems or file systems, IFS3 does not use peer Storage Server or Peer DISK which means that individual Object stored in a DISK may be replicated to any other DISK unless they are in the same Storage Server. Thus, recovering the Objects in the DISKs in the failed Storage Server does not move the operations to a specific DISK or Storage Servers, but the recovering process are distributed to all DISK in all Storage Servers. Storage services during the degraded mode are also distributed to all Storage Servers. Therefore, unlike other storage systems, InfiniStor pays fewer penalties during the degraded time while recovery process is much faster and simpler.

IFS3, Objects know which other DISK contains its own Replica. It could be primary object or secondary object. Based on that information, Storage Servers can freely recover the failed DISKs without MDS Services' control. After recovery is made, Replication Manager should update Object's Metadata information accordingly.

2.8 Advanced Features

2.8.1 DeDuplication

InfiniStor™ DeDuplication, call it InfiniDeDupe a service for de-duplicating the stored files within IFS3. InfiniDeDupe™ is not limited to deduplication within a DISK or Storage Server, but it works over entire file system. InfiniDeDupe™ finds files with identical contents and removes redundant copies from the storage system. InfiniDeDupe™ works as a post processing. In other words it does not inline dudplication process. Advantage of InfiniDeDupe™ over other DeDuplication solutions is that InfiniDeDupe™ does not slows down IFS3 Storage Services. Unlike other DeDuplication systems that use only 1 or 2 dedicated appliance servers, IFS3's patented DeDuplication technology uses all available CPU powers of the its Storage Servers. As a result, DeDuplication process takes almost no extra time or CPU power for READ and WRITES processes. On the contrary, InfiniDeDupe™ may improve the performance by better utilizing the cache memory in Storage Servers.

2.8.2 Automatic Storage Tiering or Information Lifecycle Optimization

Files have life-cycle. More recently created files tend to be more frequently used than old files. Naturally, as time passes, files are used less and less frequently. This feature is called InfiniILM™ technology understands this File Life-Cycle and manages the files accordingly.

InfiniILM ™ creates fool of Storage Servers as ArchiveStorageServers. Then, InfiniILM™ stores old and less used files to those ArchiveStorageServers. Often times, ArchiveStorageServers has many large capacity hard drives such as SATA HDD to reduce the hardware cost.

2.8.3 Native Cloud Storage Service Support

IFS3 provides native support for Cloud Storage Service Platform, CSSP for short. CSSP consists of

4 components: 1) Storage Service Gateways (SSG), 2) Identification and Access control Management Services (IAM Service), 3) Operations Support Services (OSS), and 4) Business Support Services (BSS). SSG is service gateways for various storage protocol services.

Currently, CSSP SSG provides REST-APIs, NFS, CIFS, and WebDAV protocols. Other protocols such as FTP are planned. IAM service is user accounts management and their access control management. CSSP IAM service provides hierarchical Identification services that IAM have Tenant Accounts and Tenant User Accounts. CSSP IAM uses separate sign-in credential and access credential for maximum security and flexibility. For Identification service, IAM can work with separate LDAP services for interact with existing user management service. CSSP OSS include service monitoring service, service management, patch management, external monitoring API services. OSS is very flexible to use and configure with customer's existing monitoring and management systems. CSSP BSS is primarily Logging Service. CSSP keep log of all the customer use of the CSSP Services. It include when, where, what, how, and exact amount of data.

2.9 Summary

The InfiniStor Storage System3 implements a distributed, highly scalable, and feature-rich file system and storage system using the file system. The system provides an abstract and unified view of a multiple DISKs of various types and sizes that are installed in multiple Servers. The system is potentially scalable to unlimited capacity and performance. This virtualized view of storage service systems can be accesses through various methods including (and expanding) NFS, CISF, Native Client, REST-API, and programming language bind libraries (Linux System, MS-Windows System, Python, JAVA, and growing). Internally, the system manages files and metadata associated with the files in fully distributed manner. Designed with high frequency level of hardware failure, data protection and service continuity is essential of InfiniStor.

InfiniStor Storage System3 incorporates many advanced features that are frequently used in real world system, but mostly available as a third part system. Current available features like DeDuplication, Information Lifecycle Management, DISK and Storage Tiering, Hot-Content Optimization, and DR-Sync are very useful features to operate large scale storage system. Without these features, the operator would require to have more administrators to manage system and spend more for the external products. Additional features such as SSD Cache Tiering, Tiering for CDN will give even more advantages to the service providers for better data management and more efficient services.