High Performance Computing 환경을 위한 고성능, 무정지 파일시스템 구현

The development of the high effective and stoppageless file system for high performance computing

박영배, 최승환, 이상호*, 김경수**, 공용준**
전력연구원, 충북대학교**, 이파워게이트(주)**

Abstract

In the current high network-centralized computing and enterprising environment, it is getting essential to transmit data reliably at very high rates. Until now previous client/server model based NFS(Network File System) or AFS(Andrew's Files System) have met the various demands but from now couldn't satisfy those of the today's scalable high-performance computing environment. Not only performance but data sharing service redundancy have risen as a serious problem. In case of NFS, the locking issue and cache cause file system to reboot and make problem when it is used simply as ip-take over for H/A service. In case of AFS, it provides file sharing redundancy but it is not possible until the storage supporting redundancy and equipments are prepared. Lustre is an open source based cluster file system developed to meet both demands. Lustre consists of three types of subsystems : MDS(Meta-Data Server) which offers the meta-data services, OST(Objec Storage Targets) which provide file I/O, and Lustre Clients which interact with OST and MDS. These subsystems with message exchanging and pursuing scalable high-performance file system service. In this paper, we compare the transmission speed of gigabytes file between Lustre and NFS on the basis of concurrent users and also present the high availability of the file system by removing more than one OST in operation

Ⅰ. 서 론

최근 인터넷의 성장으로 인한 수요의 증가 그리고 네트워크의 발전은 기존의 시간적, 공간적 개념을 뛰어넘은 새로운 생활 문화 공간을 만들어 가고 있다. 사용자는 인터넷을 이용하여 원하는 정보를 취득하며 자신의 자료 및 자원을 다른 사용자들과 공유, 활용 될 수 있다. 다양한 사용자가 늘어남에 때라 요구하는 데이터의 종류도 멀티미디어 데이터를 포함한 대용량의 데이터 처리 요구가 급증하고 있다. 다수의 인터넷 사용자에게 웹을 통한 대용량의 멀티미디어 데이터 서비스를 효율적으로 제공해주기 위해서는 서버 시스템내의 저장장치의 구조의 변화가 필요하다. 기존의 클라이언트/서버(예를 들면 Network File System)같은 경우, 동시 접근이 많아지고, 각각의 접근에 따른 데이터의 크기가 커지면서 저장장치의 부하가 증가하여 서버자체적인 용량의 한계와 서버와

클라이언트 간의 잦은 데이터전송에 따른 메모리 복사가 요구되며, 서버의 오류 발생시 모든 클라이언트들이 서비스를 받지 못하는 치명적인 오류를 유발할 수 있는 구조로 되어있다.

네트워크 부착형 저장장치(Network Attached Storage: NAS)와 저장장치 전용 네트워크(Storage Area Network: SAN)는 서버에 종속적으로 연결되었던 저장장치들을 Fiber 채널을 이용한 고속의 전용 네트워크에 직접 연결하여 특정 서버의 제어 없이 네트워크를 통해 직접적인 접근이 가능한, 데이터 중심적인 새로운 저장장치 환경이다. 현재 SAN을 통한 저장장치의 연구가 증가하고 있으며, 이러한 환경에서 많은 수의 상용소프트웨어가 개발되어 사용 중이다.

SAN 기반 클러스터 파일 시스템은 분산 파일 시스템의 기능을 모두 지원하며 또한 직접 연결된 저장장치들을 특정한 서버의 도움 없이 전용 네트워크를 통해 접근할 수 있으므로 기존 분산파일 시스템보다 확장성이나 가용성이 우수하다. 기존 분산 파일 시스템이 하나의 중앙 집중적인 서버에 의한 전체 저장장치 관리라는 서버 시스템의 병목현상 단점을 가지고 있는 반면, SAN 기반 클러스터 파일 시스템은 특정한 서버와의 접근 없이 SAN에 부착된 공유 저장장치들을 자유롭게 접근 할 수 있다는 장점역시 제공한다.

그러나 클러스터내의 각 서버들은 파일 공유를 위해서 기존의 파일 시스템을 사용할 수 없으며 클러스터를 위한 전용 파일 시스템을 운영해야한다. 즉 파일 공유를 위한 동시성제어나 메타 데이터의 관리기능을 할 수 있는 클러스터 파일 시스템으로 대체되어야한다. Lustre파일 시스템은 이와 같은 환경에서 사용할 수 있는 시스템이다.

본 논문은 lustre파일 시스템의 소개와 그와 관련된 새로운 개념인 OST(Object based Storage)에 대한 소개와 그 구현을 통하여 Linux 시스템 내에서의 Cluster File 시스템을 소개하고 더불어 H/A file system 설명하도록 한다.

Ⅱ. 관련연구

1. 클러스터 파일 시스템의 특징 및 기존연구

네트워크의 발전으로 인하여 지역적 파일 시스템은 점차 서버들의 클러스터를 통한 분산 파일 시스템(distributed file system)또는 공유 파일 시스템(shared file system)으로 발전하고 있다. 다수의 원격노드에 존재하는 클라이언트로부터 공유 저장장치가 동시에 마운트 될 수 있으며, 동시에 원격지의 클라이언트들에 의해 동시 접근이 가능한 파일 시스템들이다.

전형적 클라이언트-서버 파일 시스템은 서버에 의해 파일 식별 공간, 파일 접근 허가 권한 등을 제공받으며, 파일 이름과 오프셋을 디스크 블록 어드레스로 Mapping하는 기능을 제공받고 있다. 이와 같은 클라이언트-서버 분산파일 시스템은 각 클라이언트에서 사용되는 지역 파일 시스템 명령어에 의해 분산 저장된 파일을 접근할 수 있는 장점을 가지고 있다. 또한 RPC, XDR, TCP/IP 등과 같은 표준 네트워크 하드웨어와 프로토콜에 대한 독립성을 제공받을 수 있다. 이러한 분산파일시스템의 대표적인 예는 썬 마이크로 시스템사의 NFS(Network File System)과 AT&T사의 RFS(Remote File Sharing System), UCB대학의 AFS(Andrew File System)등이 있다.

다른 종류의 파일 공유 방법은 미들웨어 방식의 공유로서, 파일 공유를 위해 운영체제를 수정하지 않고 응용 프로그램과 운영체제 사이의 관련 요구 사항들을 가로채어 파일 공유가 가능하도록 하는 모듈을 이용하는 방식이다. 이와 같은 기능을 제공하는 제품으로는 IBM Tivoli사의 SANergy를 들 수 있다. 이 제품에서는 파일 공유의 동시성을 제어하기 위해 서버 역할을 하는 메타데이터 제어기 노드를 별도로 운영하여야한다.

궁극적으로 Clustering 되어 있는 서버들 간의 데이터를 공유하기 위해서는 데이터 접근시 어떠한 중앙 집중적 서버의 제어도 받지 않아야 되며, 서버의 오류로 인해 전체 클러스터 내의 노드들이 데이터 서

비스가 중단되어서도 안된다. IBM Tivoli사의 SANergy의 경우 클러스터 간의 파일 공유 Solution을 출시되어 많이 사용되고 있기는 하나 공유데이터 전송에 많은 시간이 걸린다는 단점을 가지고 있다. 또한 서버 역학을 하는 MDC 모듈과 클라이언트 모듈로 구성되는 클라이언트-서버 형식을 취하고 있어, 서버에 오버헤드가 생길 수 있으며 병목현상의 문제점을 안고 있다. 또한 클러스터내의 노드오류로 인한 전체 시스템 서비스의 중단과 이를 처리하는 장애 복구 시간이 길어 고가용성을 제공하지 못한다.

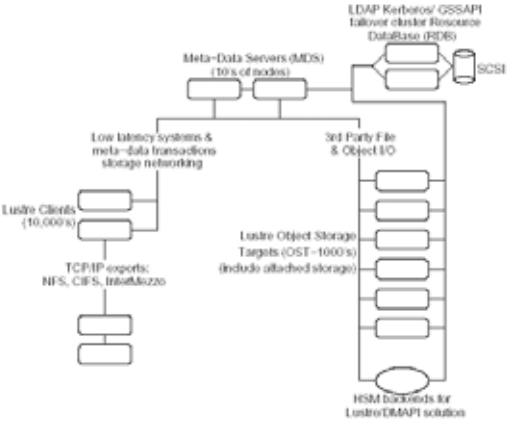
Minnesota 대학의 GFS(Global File System)의 경우 연구용 목적으로 사용되었으나 Sistina라는 기업에서 인수, 개발하여 현재 제품으로 출시되고 있다. 이 공유 파일 시스템은 GFS lock 서버 역할을 하는 노드와 클라이언트 노드들로 구성은 비대칭형구조이다. 이 제품들은 비대칭형구조의 전형적인 문제인 병목현상으로 인한 성능저하 현상을 갖게 되며, rock 서버의 오류로 인하여 전체 시스템 서비스가 중단되는 저가용성 문제를 갖는다.

최근에 출시된 Matrix사의 서버제품은 폴리서브라는 회사의 제품으로 본래 Clustering 솔루션으로 개발되었으나, 최근 공유파일 시스템의 수요가 증가함에 따라 공유 파일 기능이 추가된 경우이다. 최근 출시제품임으로 아직까지 시장 평가가 많이 이루어지지 않은 상태로써 현재 10노드 클러스터 규모를 지원하며 플랫폼 역시 Linux로 제한되어 있다.

썬 마이크로 시스템의 기술진들이 설립한 Veritas에서 출시한 SAN Point Foundation Suite는 클러스터 파일 시스템, 볼륨관리기를 포함한 제품으로 가장 최근 출시되었으며, 아직 오류로 인한 고가용성 제공의 미흡과 비용 및 번들 구성으로 인한 사용자 요구 만족 등의 문제점을 안고 있다

2. Lustre 구조

Lustre는 크게 3가지의 모듈, 클라이언트 OST (Object Storage Targets) 그리고 MDS(Meta-Data Server)로 구성된다. 자세한 것은 아래의 그림에 나타나 있다.



▶▶ 그림 1. Lustre Cluster

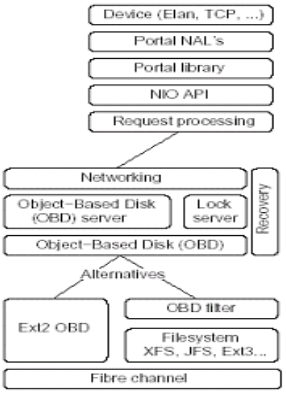
이 3개의 모듈이 서로 다른 시스템에 설치되어 있을 때는 Lustre는 파일 ma메니저를 둔 클러스터파일 시스템과 비슷한 구조를 가지게 된다. 하지만 이 3개의 모듈은 동시에 하나의 서버에서 수행될 수 있는데, 이럴 경우 대칭적 구조를 가지게 된다.

▶▶ 그림 2. Outline of Interaction Between Systems

2.1 OST(Object Storage Targets)

Lustre의 기본 개념은 Object Storage이다. 여기서 object는 파일을 저장하는 i-node로 생각될 수 있다. 각각의 object에 대한 접근은 파일의 I/O를 담당하는 OST에 의해 제공된다. Name space는 lustre의 i-node를 관리하는 meta data서비스가 관리를 하게 된다. 이렇게 될 경우 meta data는 단지 OST에서 관리하는 파일 데이터 object의 reference의 의미만을 가지게 된다.

Lustre의 기본적인 디자인은 OST가 데이터object를 위한 block을 할당하게 되고 이에 따라 분산되거나 확장 가능한 meta data의 설정까지 수행하는 것이다. 그리고 OST는 object에 접근하는 클라이언트에 보안까지 적용할 수 있게 한다. client OST protocol은 파일 처리에 대한 요구를 Remote MDS와 결합한다는 점에선 DAFS와 유사하다. 소프트웨어 모듈은 아래그림에 표시한다.



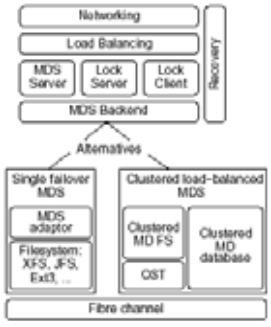
▶▶ 그림 3. OST Software Modules

OST는 다른 object storage에 네트워크 인터페이스를 제공한다. 이것은 Object Storage의 두 번째 Layer로써, 흔히 Object Storage Driver라고 부른다. 이것은 파일이라고 생각할 수 있는 object를 Storage와 함께 관리하게 된다. Object들은 raw ext2 i-node 타입으로도 저장달수 있으며 그렇지 않을 경우 필터링 드라이버에 의해 journaling 파일 타입으로도 저장이 가능하다.

Lustre는 파일에 대한 요구 처리를 Sandia에서 개발된 Portal이라는 API를 통해 처리하게 된다. Portal은 NAL(Network Abstraction Layer)에 의해 많은 네트워크 트랜스폴트를 사용할 수 있게 된다. 이 API는 메시지연결 이벤트 또는 전달을 제공하고 나아가서 RDMA를 사용할 수 있는 향상된 능력도 제공한다.

2.2 MDS(Meta Data Server)

Meta data서버는 lustre시스템에서 가장 복잡한 시스템일 것이다. Meta data서버는 메타데이터 서비스를 위한 벡엔드 저장장치를 제공할 뿐만 아니라 네트워크 인터페이스를 통해 검출되는 모든 메타데이터관련 트랜잭션을 실시간으로 update 한다. 현재는 이를 위해 journal 파일 시스템을 사용한다. 하지만 공유디스크방식 역시 곧 지원할 예정이다. 아래 그림에 위에서 설명한 기능을 나타내었다.



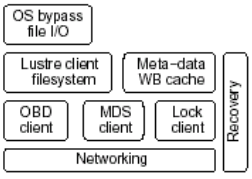
▶▶ 그림 4. Meta-Data Server Software Modules

MDS는 ext3와 XFS와 같은 저널 파일시스템의 기능을 가진 locking 모듈이 탑재되어 있다. Lustre lite 시스템에서는 Meta data 서버는 오직 한대만으로 구성되도록 되어있지만 킴벌라이트(kimberlite)와 같은 솔루션을 single point of Failure는 막을 수가 있다.

Lustre에서 메타데이터의 처리는 적절히 시스템의 작업량을 나눌 수 있는 시스템으로 진화할 것이다. 하지만 네트워크를 통해 외부에서 메타데이터 서비스를 제공하는 기존의 방식은 바뀌지 않을 것이고 대신 메타데이터 서버들 간에 데이터의 복사를 위한 meta-data dispatch 요구를 처리할 수 있는 stackable framework를 구현하는 쪽으로 진행될 것이다.

2.3 Client File System

클라이언트 쪽의 Meta data protocol은 AFS, Coda, intermezzo 파일 시스템에서 가져온 트랜잭션 기반의 모델을 기초로 한다. 이 프로토콜은 Meta data와 실제 파일 데이터의 coherency는 유지하면서 접근 승인과 모든 메타데이터 update를 Caching하는 것은 특징으로 하고 있다. 데이터의 coherency는 만일 application 레벨에서 지원한다면 좀 더 효율적일 수 있다. 클라이언트 모듈은 아래 그림과 같이 다양한 소프트웨어 모듈을 포함하고 있다.



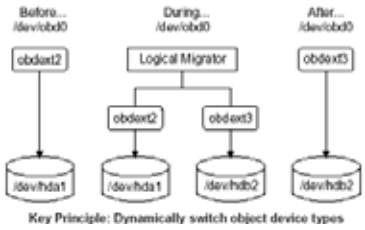
▶▶ 그림 5. Client Software Modules

Lustre와 Lustre Lite 파일시스템은 진보된 기능인 scalable allocation 알고리즘, 보안, Meta-data 제어와 같은 기능을 정확하게 제공한다. IBM의 GPFS와 같은 전통적인 클러스터 파일시스템은 이런 기능을 제공하지만, 독립적인 추상화 대신에 대규모의 단일 파일 시스템의 한 부분으로 존재한다.

2.4 Storage Management

Lustre는 데이터 Migration, snapshot, 향상된 보안, 그리고 데이터 mining을 위한 active disk component와 같은 상당히 향상된 기능을 처리하기 위한 다양한 방법을 제공한다. 이러한 스토리지의 관리는 object 모듈의 상호과정을 통해 실현된다. 일반적인 프레임워크가 드라이버 스택을 동적으로 변화시키고 관리하기 위해 제공된다.

stacking object 모듈의 예는 아래그림에 나타내고 있다. 아래 그림은 data의 hot Migration의 과정을 나타내고 있다.



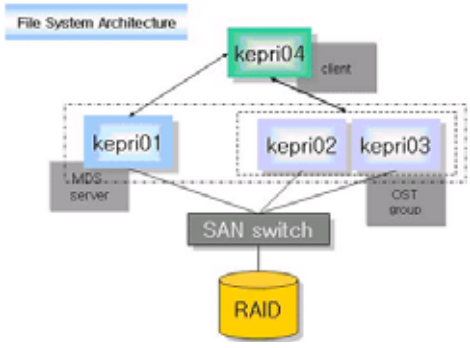
▶▶ 그림 6. Hot Data Migration

Ⅲ. 시스템 설계

본 논문을 위하여 총 4대의 서버를 사용하여 시스템을 구성하였다.

[표 1] 시스템 구축서버 사양

종류	hostname	CPU	RAM	Network	O/S
MDS	kepri01	PIII 800	128	100MB	Linux
OST1	kepri02	PIII 800	128	100MB	Linux
OST2	kepri03	PIII 800	128	100MB	Linux
Client	kepri04	PIII 800	128	100MB	Linux



▶▶ 그림 7. 시스템구성도

1. Lustre OST

서버이름을 각각 kepri02, kepri03으로 지정하였다. 그리고 이들 노드는 SAN switch를 통해 RAID 볼륨중 하나인 /dev/sda2를 공통으로 사용한다. 이들 시스템은 active/ inactive status로 file system I/O를 제공하며, 후에 kepri02나 kepri03 둘 중 하나의 시스템에 장애가 생겼을 경우라도 정상적으로 file

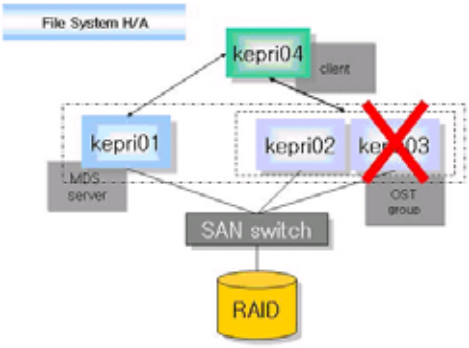
system I/O를 제공하도록 HA mode로 구성하기 위해 RAID 시스템의 같은 파티션(/dev/sda2)을 사용하도록 구성하였다.

2. Lustre OST

kepri01이라고 명명한 시스템은 SAN Switch를 통하여 RAID의 /dev/sda1를 meta data를 위한 파일 시스템으로 사용한다.

3. Lustre Client

kepri04를 lustre file system을 mount하여 사용하는 client로 지정하여 사용한다.



▶▶ 그림 8. HA 시스템 시험 구성도

kepri04 시스템이 lustre file system을 통하여 자신의 /mnt/lustre Directory에 mount하고 있다. 그 동안 에러 상황이라고 가정하여 kepri03 서버를 강제로 shut down 시킨 후, /mnt/lustre Directory에 여전히 read/write가 가능한지를 확인하는 것으로 한다.

Ⅳ. 실험 및 고찰

1. 성능시험

데이터의 write 성능을 보기 위해 전형적인 데이터 공유 시스템인 NFS와 그 성능을 비교해 보았다. 테스트는 kepri01을 NFS 서버로 하고 kepri04에서 NFS로 Mount한 Directory와 Lustre로 Mount한 Directory간의 write 성능을 테스트하였다. 단, 이 테스트의 경우 Lustre 파일 시스템의 경우 RAID 볼륨을 Mount한 것이 아니라 MDS를 kepri01의 로컬 IDE 하드디스크로 OST는 kepri02의 로컬 IDE 하드디스크를 사용하였다. IDE와 RAID 시스템의 속도차를 최소화하기 위하여 위와 같은 방법을 선택하였다.

두 개의 파일시스템에서 100MB의 파일을 생성하는 것을 테스트 해본 결과는 아래와 같다.

[표 2] NFS와 Lustre 속도 비교표

파일 시스템	NFS	Lustre
속도(sec)	8.911	11.128

Client 시스템이 1대일 경우에는 NFS가 훨씬 빨랐다. 이는 Lustre의 경우 단 한대의 Client에서 transaction이나 file update가 일어날 경우 MDS와 OST 모두에게 네트워크 traffic을 발생시키고 MDS와 OST간의 통신으로 인해 약간의 delay가 생성되는 반면, NFS에서는 동일시스템 내에서 모든 것들이 이루어지기 때문에 발생된 결과라고 할 수 있겠다. 만일 MDS와 OST간의 네트워크를 infiniband와 같은 고속의 네트워크를 사용한다면 이 시간차이는 더 줄어 들것이다.

그리고 Client의 수를 더 늘일 수 있었다면 역시 그 결과는 반대가 되었을 것이다.

2. HA 시험

위에서 가정한대로 두 번째 OST 즉 kepri03을 리부팅 시킨 상태에서 파일 Creation 및 Remove를 시도해 보았으나, 이 둘 모두 성공적으로 수행되었다.

3. 고찰

Lustre 파일 시스템의 경우 Linux 기반의 SAN file system이라는 의미 이외에도 Object Based

Storage라는 새로운 개념의 저장기술을 적용한 파일 시스템이라는 데에 그 의미가 크다 하겠다. 그리고 기존의 공유 시스템과는 달리 Meta-data 시스템과 실제 파일의 입출력을 담당하는 시스템을 별도로 구분해 유지할 수 있어서 확장성과Load Balancing에 있어서 탁월한 성능을 발휘할 수 있도록 개발된 시스템이다.

파일 입출력의 성능 역시 기타의 다른 파일시스템과 비교해서 뒤떨어지지 않는 성능을 보였고, fail-over 역시 기타의 다른 상용 파일 시스템에 비해 결코 뒤지지 않는 성능을 가졌다.

다만 아쉬운 것이 있다면, OST 그룹 중 한대에 이상이 생겨서 다운 됐을 때 적절한 조취를 취해 시스템을 정상복구 했다 하더라도 이를 다시 OST 그룹에 추가시키는 것은 불가능하다는 것이다. 만일 다시 OST에 추가시키려할 경우 모든 파일 시스템, MDS, 다른 OST 그리고 마운트하고 있는 클라이언트도 포함한 관련 서비스를 중지시키고 다시 시작해야한다는 것은 상당히 critical한 결점이라고 할 수 있겠다.

그리고 OST나 MDS를 여러 대의 서버로 구현했을 경우 각각의 서버에 Load Balancing을 하는 부분의 알고리즘 역시 아직은 취약한 상태여서 이 부분에서의 성능 및 알고리즘 개발도 필요하다.

▌참고문헌▐

[1] "The Lustre Storage Architecture, A architecture, design and user manual for Lustre", pp. 16-20
[2] "A Scalable, High Performance File System", Cluster File System Inc. pp. 1-2
[3] Peter J.Braam, "Objected based Storage Cluster File System & Parallel I/O", Stelias Computing and Carnegie Mellon University. pp. 34-35
[4] Peter J.Braam, "Lustre Scalable Clustered Object Storage", Stelias Computing and Carnegie Mellon University. pp. 16-17, 31-34
[5] http://www.lustre.org
[6] PeterJ.Braam & Michael J.Callahan, "Lustre : A SAN File System fro Linux", Stelias Computing, inc, 1999
[7] 박영배, "고속수치 연산 클러스터링 기법연구", 한전전력연구원, pp. 35-40, 2004