Multimedia Data를 위한 병렬 파일 시스템

*박시용^{U} *석창규 **박성호 **김영주 **정기동
*부산대학교 일반대학원 멀티미디어협동과정 **부산대학교 전자계산학과
{sypark, cgseok, shpark, yjkim, kdchung}@cs.melon.pusan.ac.kr

Parallel File System for Multimedia Data

*Si-Yong Park^{O} *Chang-Gyu Seok **Seung-Ho Park **Young-Ju Kim **Ki-Dong Chung
*Inter Disciplinary Research Program of Multimedia, Pusan National University
**Dept. of Computer Science, Pusan National University

요 약

본 논문에서는 여러 개의 디스크를 클러스트화한 메시지 전송 기반의 병렬 멀티미디어 파일 시스템(PMFS : Parallel Multimedia File System)을 제안하고 설계, 구현하였다. 본 논문에서 구현한 PMFS는 이식성, 유연성 그리고 확장성을 고려한 멀티미디어 데이터를 지원하는 병렬 파일 시스템으로 2계층 분산 클러스트 구조에 적합하다. 그리고 제어 메시지와 TCP를 기반으로 서버들간에 통신을 하고 다양한 방법의 데이터 배치 기법을 제공한다. PMFS의 성능 평가 결과 데이터들이 임의 시작 블록과 DIS 배치 기법으로 저장된 경우 가장 좋은 성능을 보였다.

1. 서론

최근 인터넷의 대중화와 네트웍 전송기술 및 장비의 발달로 인하여 인터넷 기반의 다양한 멀티미디어 응용 소프트웨어가 개발되어 상용화되고 있다. 이러한 멀티미디어 응용 소프트웨어에서는 고전적인 이미지나 텍스트 파일뿐만 아니라 연속성과 실시간성을 가지는 비디오나 오디오 데이터도 다루고 있다[1]. 멀티미디어 데이터와 같이 혼합된 미디어 데이터를 저장하고 서비스하는 대용량의 멀티미디어 저장 서버에서 발생하는 입출력의 병목 현상을 처리하기 위하여 하나의 객체로 단일 디스크에 저장하지 않고 여러개의 디스크를 클러스트화하여 데이터를 각 디스크에 분산하여 배치한다[2]. 클러스트 서버의 구조 중에서 부하 균등과 대역폭 측면에서 유리한 2계층 서버구조는 저장 서버와 제어 서버간의 통신 오버헤드가 가장 큰 문제점으로 대두되고 있다[3]. 본 논문에서 제안하고 구현한 병렬 멀티미디어 파일 시스템(PMFS : Parallel Multimedia File System)에서는 이러한 문제점을 해결하기 위해서 저장 서버에서 직접 클라이언트로 데이터를 전송한다. 그리고 PMFS는 멀티미디어 저장 서버의 호환성과 이식성을 고려한 효율적인 병렬 파일 시스템 관리를 위해서 개발자가 객체의 블록 크기와 데이터 배치 정책, 사본 정책등을 선택는 기능을 제공한다[4]. 본 논문의 2장에서는 PMFS를 설계하고 구현하는데 필요한 관련 연구를 소개하고 3장에서는 PMFS의 구조를 설명한다. 4장에서는 PMFS의 성능을 평가, 분석하고 마지막으로 5장에서 결론과 향후 연구과제를 언급한다.

2. 관련연구

2.1 병렬 파일 시스템

병렬 파일 시스템은 다양한 목적을 가지고 연구 개발되고 있으며 클러스트 기술을 기본적으로 적용하고 있다. 대표적인 병렬 파일 시스템은 일반적인 목적으로 구현된 Galley[4], PVFS(Parallel Virtual File System)[5], PPFS(Portable Parallel File System), GFS(Global File system) 등이 있으며 특정시스템을 위한 것으로는 PFS(Parallel File System for Intel Paragon), SciFS(SCI Cluster)등이 있다. 그리고 실시간 연속 미디어를 지원하기 위한 병렬 파일 시스템으로는 Symphony, Mitra, Video Server Array, Clockwise등이다. 특히 Symphony와 Clockwise는 실시간 연속 미디어와 비실시간 미디어를 함께 지원하는 멀티미디어 병렬 파일 시스템이다[6].

2.2 클러스트 미디어 저장 서버

클러스트 미디어 저장 서버의 제어 서버와 저장 서버는 물리적 위치에 따라 수평 구조와 2계층 구조로 나눌 수 있고 미디어 객체의 배치 정책에 따라서는 독립 구조와 분산 구조로 나눌 수 있다. 수평 구조는 제어 서버와 저장 서버가 동일 노드상에 존재하고 2계층 구조는 제어 서버와 저장 서버가 서로 다른 노드상에 존재한다. 전자는 통신 오버헤드가 적고 후자의 경우에는 통신 오버헤드가 크며 제어 서버에 부하가 집중되는 경향이 있다. 미디어 객체가 단일 노드상에 저장, 관리되는 독립 구조는 서버의 확장이 용이하지 않은 단점이 있고

미디어 객체가 여러 노드에 분산 저장되는 분산 구조에서는 데이터의 조작을 위해서 서버간의 동기화가 필요하다. 그러나 서버의 확장이 용이한 장점이 있다[3].

2.3 데이터 배치기법

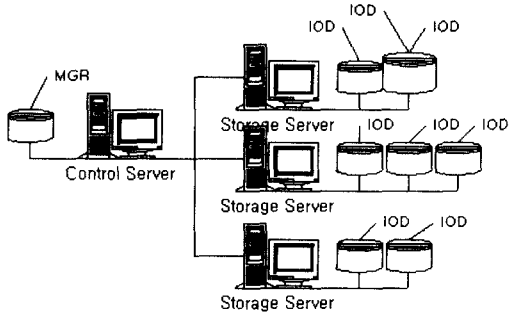
대용량의 연속 미디어 데이터를 효율적으로 저장하기 위해서는 부하균등을 고려하여 여러 개의 디스크에 분산하여 배치한다. 이러한 분산 배치를 토대로 한 기법에는 동일 기종간의 데이터 배치(Round-Robin[7], Staggered Striping, Random Permutation, DIS[8] 등)와 이기종간의 데이터 배치(BSR, Bandwidth Ratio, Storage Space Ratio등)로 크게 나눌 수 있으며 디스크의 결함을 허용하기 위한 배치 기법으로는 복사본을 다른 디스크에 저장시키는 사본 배치 기법과 패리티 비트를 이용한 패리티 기반의 배치 기법을 들 수 있다. 그리고 배치할 데이터의 시작 Block을 어떻게 위치시키는가에 따라서 고정 시작 블록 배치 기법과 임의 시작블록 배치기법으로 나눌 수 있다[8].

3. PMFS(Parallel Multimedia File System)

PMFS는 멀티미디어 데이터를 지원하는 병렬 파일 시스템으로서 아래의 특징을 가진다.

- 커널 레벨이 아닌 상위의 응용 프로그램 레벨에서 구현된 파일시스템으로서 Linux나 FreeBSD, Solaris등과 같은 UNIX계열의 운영체제에서 쉽게 구동될 수 있다.
- 각각의 저장서버에서 읽은 데이터는 제어 서버를 거치지 않고 클라이언트로 전송이 가능하다.
- 실시간 및 비실시간 데이터를 지원하고 실시간 데이터의 경우에는 마감시간(Deadline)을 설정할 수 있다.
- 멀티미디어 데이터를 지원하기 위해서 Round-Robin, Staggered Striping, DIS데이터 배치 기법을 제공하며 시작 블록의 위치를 고정시킬 수도 있고 임의로 배치할 수도 있다.
- 미디어 객체를 시스템 환경에 맞게 블록의 크기, 배치 기법,미디어 형태, 사본 정책등을 설정할 수 있도록하여 객체들을 유연하게 저장할 수 있도록 한다.
- 이기종 물리적 디스크를 논리적인 동일기종으로 구성할 수 있게 하는 방법으로 이기종 디스크를 지원하고 그리고 미디어 사본을 저장할 수 있도록 허락함으로서 디스크의 결함 발생시에도 서비스할 수 있다.

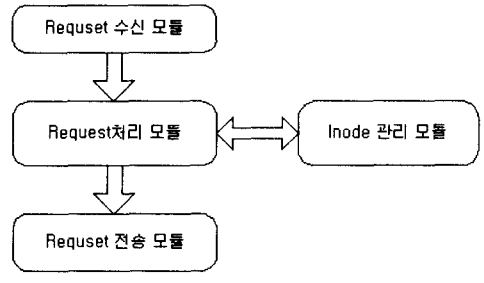
3.1 PMFS 구조



[그림 1] PMFS 구조

위의 [그림 1]에서 보듯이 크게 세 개의 구성 요소를 가진다. 먼저 MGR(Manager Daemon)은 PMFS의 MetaData인 Inode 정보를 관리하고 IOD(I/O Daemon)는 미디어 객체의 데이터를 저장 검색하는 데이터의 입출력을 수행한다. PMFS는 또한 응용 프로그램을 위한 인터페이스로 11개의 사용자 라이브러리 함수를 제공한다. PMFS에서 제공하는 라이브러리는 C로 구현된 C API와 JNI(Java Native Interface)로 구현된 Java API로 구성되어 있다.

3.2 MGR(Manager Daemon)



[그림 2] MGR 구조

MGR의 내부구조를 살펴보면 PMFS 라이브러리로부터 전송되어 오는 Request를 수신하는 Request 수신 모듈, 수신된 Request를 처리하는 Request처리 모듈 그리고 Inode의 정보를 갱신 혹은 검색하는 Inode 관리 모듈이 있다. 그리고 수신된 Request의 처리 결과를 Request 전송 모듈에서 다시 PMFS 라이브러리로 전송한다[그림 2].

PMFS에서 파일의 정보를 저장하는 Inode는 일반적인 파일 시스템이 가지는 정보외에 멀티미디어의 다중 노드 및 결함 허용을 지원하기 위해 몇 가지 정보가 추가적으로 저장된다.

3.3 IOD(I/O Daemon)

PMFS에서 데이터의 입출력을 수행하는 IOD는 Request들을 수신하는 Request Reader와 Job List를 관리하는 Job Scheduler, 전송할 Data의 List를 관리하는 Data List, 크게 3개의 모듈로 나누어 질 수 있다[그림 3].

- Request Reader

입출력 Request를 읽어서 Job List에 Job을 삽입한다. 그러나 쓰기 Request인 경우는 Job List에 삽입하지 않고 Request Reader에서 처리한다.

- Job Scheduler

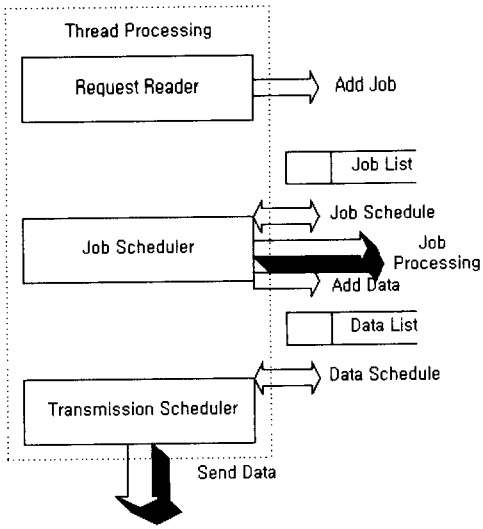
Job List에 있는 Job들을 마감 시간이 설정되어 있는 경우에는 EDF 알고리즘으로 스케줄링하고 만약 마감 시간이 설정되지 않았다면 FIFO순서로 스케줄링하고 Job들을 실행한다.

- Transmission Scheduler

데이터 List에 데이터가 있을 경우 FIFO순서로 데이터를 전송한다.

IOD에서는 데이터를 읽기 위해 Non-Blocking I/O를 하며 정해진 시간 내에 데이터를 읽어 오지 못할 경우 해당 IOD를 실패한 디스크로 표시하고 사본 디스크에서 데이터를 읽어온다. 따라서, 데이터 읽기를 수행하는 도중 하나의 디스크에 결함이 발생하여 서비스가 불가능해졌을 때, 사본 디스크에서 정상적으로 읽어 들이면서 정상적인 서비스를 계속 할 수 있게 된다.

IOD에서 파일 읽기는 각각의 IOD에 읽기 요청을 보내고 IOD가 목표 주소로 데이터를 전송한다. 목표 주소는 PMFS 라이브러리에 의해서 설정된다.



[그림 3] IOD 구조

3.4 PMFS 라이브러리

PMFS에서는 응용 프로그램을 위해서 PMFS 라이브러리 함수를 제공한다[그림 4].

함수 이름	용도
pmfs_create	PMFS 파일을 생성한다.
pmfs_open	PMFS 파일을 연다.
pmfs_write	PMFS 파일에 데이터를 쓴다.
pmfs_read	PMFS 파일을 읽는다.
pmfs_lseek	PMFS 파일포인터를 옮긴다.
pmfs_delete	PMFS 파일을 지운다.
pmfs_close	PMFS 파일을 닫는다.
pmfs_mkdir	PMFS 디렉토리를 만든다.
pmfs_rmdir	PMFS 디렉토리를 제거한다.
pmfs_setdeadline	파일의 마감 시간을 설정한다.
pmfs_settarget	전송할 목표 주소를 설정한다.

[그림 4] PMFS 라이브러리 함수

4. 성능 평가 및 분석

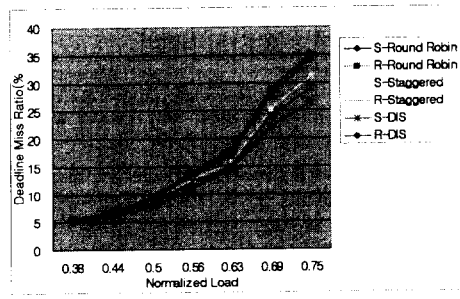
PMFS의 데이터 배치 기법 및 데이터 전송 구조의 타당성을 검증하기 위하여 사용자수에 따른 마감 시간 실패율을 측정하였다.

4.1 환경

1개의 제어 서버와 3개의 저장 서버로 실험 환경을 구축하였고 사용자 입력은 외부 서버(Sun OS,Ultra Sparc 30)에서 가상적으로 발생시켰다. 그리고 성능 평가에 사용된 운영체제는 Linux Kernel2.2.9이다.

4.2 평가 및 분석

임의 시작 블록 배치 기법이 고정 시작 블록 배치기법보다 평균 7~14% 정도 나은 성능을 보였고 DIS배치 기법이 Round-Robin기법이나 Staggered Striping기법 보다 더 좋은 성능을 보였다[그림 5].



(S- : 고정 시작 블록, R- : 임의 시작 블록)

[그림 5] PMFS 성능 평가(파일 읽기)

5. 결론 및 향후 과제

본 논문에서 제안하고 구현한 PMFS는 클러스트 기반의 2계층 분산 구조를 가지고 있으며 제어 서버와 저장 서버, 라이브러리 함수간에는 메시지를 통해서 입출력을 수행한다. 내부 통신 오버헤드와 전송시 제어 서버에 과중한 부하가 발생하는 것을 해결하기 위해서 저장 노드에서 데이터 입출력을 직접 수행할 수 있도록 하였다. 그리고 효율적인 시스템 이용을 위해서 블록 크기 및 데이터 배치 기법등을 제공한다.

다양한 종류의 미디어 서버에 효율적인 파일 시스템을 제공하기 위해서 더 발전된 입출력에 대한 연구와 다양한 실시간 스케쥴링 기법의 도입에 관한 연구가 앞으로 진행되어야 할 것이다.

참고 문헌

[1] Emmanuel Cecchet,"SciFs Technical & Practical Guide",http://sciserv.inrialpes.fr/SciFS/scifs_doc.html, 1999.

[2] Asit Dan,Dinkar Sitaram, "An Online Video Placement Policy on Bandwidth to Space Ratio(BSR)", Proceedings of ACM SIGMOD International Conference on Management of Data, 1995.

[3] 김영주,"확장성을 가진 연속 미디어 저장 서버의 설계 및 성능 분석",부산대학교 일반대학원 이학박사 학위논문,1999

[4] Nils Nieuwejarr, "Galley: A New Parallel File system For Scientific Workloads", Ph.D Dissertation, Dartmouth College, Computer Science Department, 1996

[5] W.B.Ligion III and R.B.Ross,"An Overview of the Virtual File System", Proceeding of the 1999 Extreme Linux Workshop,1999

[6] Heinz Stockinger, "Directory on Parallel Input/Output", Master's Thesis, University of Vienna, Austria, 1998

[7] C. Bernhardt and E. W. Biersack, "The Server Array: A Scalable Video Server Architecture",in High Speed Networking for Multimedia Applications, 1996

[8] 정귀옥, "Clustered NOD 저장 서버에서 데이터 Striping 및 Replica 배치기법", 부산대학교 일반대학원 이학석사 학위논문, 1998