HA-PVFS : 시간적 지역성에 적응적인 데이터 고가용성을 지원하는 PVFS 파일 시스템

심 상 만[†] · 한 세 영[††] · 박 성 용[†††]

요 약

클러스터 파일 시스템에서 가용성 지원을 위해 파일 복제 방법과 패리티 서버 방식이 사용되어 왔으나, 공간과 시간적 비용이 매우 크고 대량 장애 상황에 적절히 대처하지 못하는 문제가 있다. 따라서 시간적 지역성이 높은 중요한 파일에 대해서만 복제를 하거나 패리티를 생성하게 하여 효율적으로 데이터의 고가용성을 지원하는 HA-PVFS 파일 시스템을 제안한다. 이 파일 시스템에서는 동적으로 주어지는 파일 접근 빈도를 적응적으로 추정해 내기 위하여, 제한적인 정보를 통해 실제 접근 패턴을 정확히 판단하는 알고리즘을 사용하고, 성능 저하를 최소화하기 위해 지연 업데이트 방식과 릴레이식 복제를 사용한다.

키워드 : 클러스터 파일 시스템, 고가용성, 적응적 복제

HA-PVFS : A PVFS File System supporting High Data Availability Adaptive to Temporal Locality

Sangman Sim[†] · Saeyoung Han[††] · Sungyong Park[†††]

ABSTRACT

In cluster file systems, the availability of files has been supported by replicating entire files or generating parities on parity servers. However, those methods require very large temporal and spatial cost, and cannot handle massive failures situation on the file system. So we propose HA-PVFS, a cluster file system supporting high data availability adaptive to temporal locality. HA-PVFS restricts replication or parity generation to some important files, for that it employs an efficient algorithm to estimate file access patterns from limited information. Moreover, in order to minimize the performance degradation of the file system, it uses delayed update method and relay replication.

Key Words : Cluster File System, High Availability, Adaptive Replication

1. 서 론

초기의 클러스터 파일 시스템에서는 네트워크상에 위치하는 특정 파일을 마치 내 장비에 연결된 지역 디스크에 저장된 파일처럼 사용하기 위한 연구가 주류를 이루었다[1, 2]. 그러나 성능에 대한 요구가 더욱 높아짐에 따라, 디스크 자체보다는 저장 서버의 입출력 대역폭에서 병목현상이 일어나게 되므로, 서버를 다중으로 구성하여 입출력 시간을 최소화하는 방향의 연구가 추진되었다[3-6]. 그 결과 클러스터 파일 시스템은 사용자가 요구하는 성능을 제공하면서 저렴한 비용으로 구축할 수 있는 파일 시스템으로 인정받게 되었으며, 나아가 고성능을 필요로 하는 과학 계산용 파일 시스템으로도 사용되게 되었다. 근래에는 기업체에서도 고성능과 저렴함의 이점을 수용하여 여러 가지 클러스터 파일 시스템을 개발하고 있는데, 특히 인터넷 서비스 분야에서 그 효용성을 입증 받고 있다[1, 6].

그러나 상업적인 서비스를 위한 파일 시스템이 개발되면서 이전까지는 중요시 여기지 않았던 시스템 가용성 문제가 새로이 대두되었다. 대부분의 파일 시스템에서는 파일들을 분할해서 저장함으로써 고성능을 구현하는데[6, 7], 이는 가용성 측면에서 보면 아주 위험한 방법이다. 즉, 시스템에 존재하는 여러 디스크 중 하나에 장애가 발생하면 전체 파일 시스템이 동작을 할 수 없게 되어 정상적으로 작동하는 다른 디스크의 데이터도 읽지 못하게 된다. 또한 클러스터 파일 시스템이 단순히 디스크만 여러 개 붙여서 구축하는 것

※ 본 연구는 한국과학재단 목적기초연구(R01-2003-000-10627-0) 지원으로 수행되었음.
† 정 회 원 : 삼성전자 Mobile Platform Sol. Lab 연구원
†† 정 회 원 : (주)이엔지 기술본부 부장
††† 정 회 원 : 서강대학교 컴퓨터학과 부교수
논문접수 : 2005년 8월 23일, 심사완료 : 2006년 3월 22일

이 아니기 때문에 시스템 자체, 네트워크 카드, 네트워크 라인 등의 병렬 사용이 모두 가용성 저하의 원인이 된다. 따라서 상업적인 용도의 클러스터 파일 시스템을 위하여 여러 가지 가용성 지원 방법이 제시되어 왔다. 그러나 기존의 연구에서는 지역 파일 시스템의 가용성을 위해 사용되었던 기존의 알고리즘을 약간 수정한 것이 대부분이어서 태생적 한계를 가지고 있다. 이 중 대표적인 것이 전체 파일 시스템 복제 방식[6]과 RAID[8]에서 사용하던 패리티 디스크 방식[7]이다. 전체 파일 시스템 복제 방식은 실시간으로 전체 파일 시스템의 내용을 복제하는 방법으로 복제용 디스크를 위한 공간적 비용과 실시간 복제에 따른 성능 저하의 단점을 가지고 있다. 패리티 서버 방식은 파일을 분할 저장하면서 패리티 데이터를 생성하는 방식인데, 두 가지 방법 모두 특정 개수 이상의 노드나 디스크에 문제가 생기면 전체 파일 시스템이 동작하지 못하는 문제는 해결하지 못하였다.

상업적인 용도의 클러스터 시스템의 경우 웹서버, 미디어 서버, 데이터베이스 서버처럼 단지 몇 개의 서비스만 수행하는 경우가 많은데, 이런 단일 서비스가 갖는 파일 시스템에 대한 접근 패턴은 그 지역성이 아주 높다[9, 10]. 따라서 본 논문에서는 불필요한 파일을 위한 복제나 패리티 생성을 제한하고, 시간적 지역성이 높은 파일에 대해서만 복제를 하거나 패리티를 생성하게 하여, 가용성 지원에 필요한 공간적이고 시간적인 비용을 충분히 줄일 수 있게 하는 HA-PVFS 파일 시스템을 제안하고자 한다.

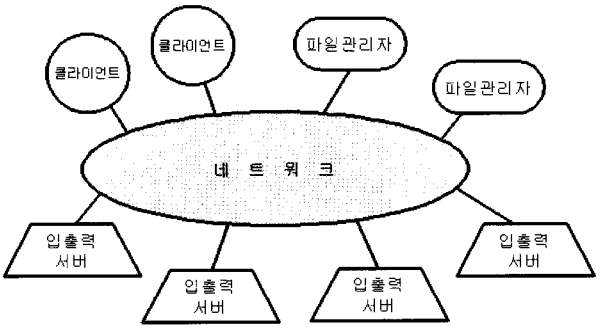
HA-PVFS 파일 시스템은 기존의 PVFS 파일 시스템을 기반으로 데이터의 시간적 지역성에 적응적으로 데이터의 가용성을 보장하도록 하는 파일 시스템이다. 즉, HA-PVFS에서는 파일에 중요도라는 개념을 도입하고 그 중요도는 파일에 대한 접근 빈도, 즉 시간적 지역성(temporal locality)을 통해 측정한다. 이때, 지역성에 대한 정보를 모두 저장할 수 없으므로, 제한적인 정보를 이용해 실제 파일의 접근 패턴을 정확히 판단하는 알고리즘이 필요하다. 본 논문에서는 이를 위해 인터넷 서비스의 접근 패턴을 지수 분포라고 가정하고 판단 최우 측정기를 사용하여 파일의 접근 패턴을 추정한다. 또한 이런 기본 알고리즘과 더불어 HA-PVFS에서는 복제에 의해 나타나는 성능 저하를 최소화하기 위하여 지연 업데이트 방식과 릴레이식 복제 방법을 사용한다.

본 논문의 나머지 구성은 다음과 같다. 2장에서는 클러스터 파일 시스템의 가용성을 지원하기 위한 기존의 연구들과 그 문제점을 기술하고 3장에서는 제한적인 복제를 통해 필요로 하는 파일들만의 가용성을 지원하는 방법을 적용한 HA-PVFS를 설명한다. 4장에서는 HA-PVFS가 기존 시스템의 성능을 떨어뜨리지 않으면서도 가용성을 충분히 보장하고 있음을 실험을 통해 보인다. 마지막 5장에서는 결론과 향후 연구 과제를 논의한다.

2. 관련 연구

2.1 Zebra 파일 시스템

패리티 디스크 방식을 통해 가용성을 구현한 파일 시스템으로 캘리포니아 대학에서 개발한 Zebra 파일 시스템이 있다[7]. (그림 1)과 같이 Zebra 파일 시스템은 클라이언트와 파일 관리자, 그리고 각 저장 장치를 관리하는 입출력 서버들이 네트워크로 연결된다.



(그림 1) Zebra 기본 구조

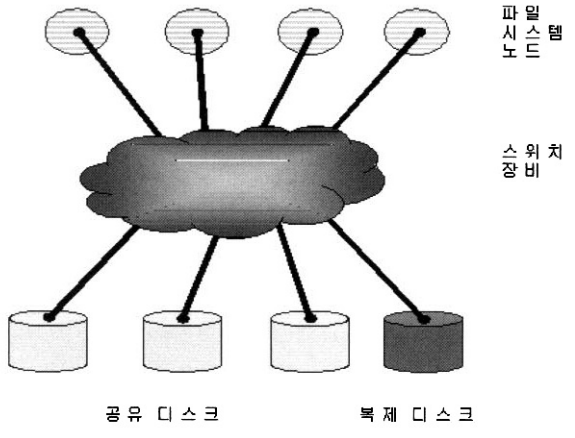
Zebra는 데이터 업데이트 시 데이터를 분할하여 일부의 입출력 서버에 저장하고, 동시에 해당하는 블록의 패리티를 계산해서 패리티 서버에 저장한다. 만일 입출력 서버에 장애가 발생하면 클라이언트는 문제가 생긴 서버의 패리티를 계산하여 원래의 데이터를 읽어낸다. 그러나 문제가 발생한 후에 해당 서버로의 쓰기 작업들은 모두 거부되므로 사용자가 직접 패리티 조각들을 복구시켜주기 전까지는 더 이상의 쓰기 작업은 불가능해 진다. 또한 기존에 패리티를 지원하는 RAID에서도 문제였듯이 특정 개수 이상의 서버에 문제가 발생하면 원래의 데이터를 복구해 내지 못하는 단점이 있다.

2.2 GPFS

GPFS는 IBM에서 자사의 슈퍼컴퓨터와 리눅스 클러스터 시스템에 적재할 목적으로 개발한 (그림 2)와 같은 공유 디스크 환경의 클러스터 파일 시스템으로, 학계에서 연구되어 오던 기존의 연구 성과를 대부분 수용한 고성능의 파일 시스템이다[6].

GPFS는 상업적인 용도의 파일 시스템이었고, 특히 데이터와 메타 데이터 모두를 여러 디스크에 분할해서 저장하기 때문에 특정 디스크의 장애는 전체 파일 시스템의 장애로 직결되는 치명적인 것이었으므로, 가용성을 지원하기 위해 크게 두 가지의 방법을 제안하고 있다. 첫 번째는 이중 연결형의 RAID 제어기의 사용인데, 이는 고기용성을 지원하는 하드웨어로 디스크 자체에 결함이 생기는 문제를 최소화할 수 있으나, 일반적인 파일 시스템에서의 실질적인 해결책은 되지 못한다. 두 번째는 전체 파일 복제 방식을 제안하고 있는데, 사용자가 추가적인 디스크를 설치한 후 파일 시스템에 알려주면 GPFS는 모든 파일을 지정된 하나 이상의 디스크에 복제하고 특정 디스크에 문제가 발생하게 되면 원본 디스크의 기능을 복제 디스크에서 대신하게 한다.

이런 전체 파일 시스템 복제 방식은 시스템의 가용성을 지원하지만, 여분의 복제 디스크가 필요하므로 추가적인 비용이 들고, 특히 GPFS는 고가의 공유 디스크를 사용하는



(그림 2) GPFS 공유 디스크 환경

클러스터를 대상으로 개발되었기 때문에 비용 부분이 무시할 수 있는 항목이 아니었다. 그리고 시스템에서 일정 수 이상의 서버가 동시에 문제를 일으키게 되면 파일 시스템 전체가 동작하지 못하는 문제는 해결하지 못하였다.

2.3 GFS(Google File System)

GFS는 Google이라는 특수한 업체를 고려하여 만들어진 파일 시스템으로, 확장성 있고 관리하기 쉬우며 고가용성을 지원하는 파일 시스템을 위해 만들어 졌다[11]. 기존 시스템들은 설계 시 장애 상황을 상당히 드물게 일어나는 사건이라고 생각하여 접근한 반면, GFS는 장애 상황을 일어날 수 있는 상황으로 받아들이고 이에 대처하기 위한 가용성 부분에 심혈을 기울여 설계하였다

GFS 시스템은 크게 전체 시스템을 관리하는 마스터 노드 하나가 있고 그 외의 모든 서버들은 입출력을 담당하는 청크 서버와 실제 파일 관련 요청을 만들어 내는 클라이언트로 구성된다. GFS가 고가용성을 지원하기 위해서 하는 가장 기본적인 작업은 파일을 청크 단위로 나누는 것이다. 나눠진 청크 조각들은 시스템 내의 임의의 청크 서버에 뿌려지게 되고 지정된 크기와 개수로 복제된다. 복제본이 생성된 후 일어나는 쓰기 작업은 모든 복제본을 차례로 업데이트하면서 이루어진다. 만약 특정 서버에 문제가 발생하게 되면 마스터 서버는 그 노드에 존재하던 복제본들을 더 복제하여 복제본 수를 일정하게 유지하여 가용성을 유지한다. 복제본 자체가 많은 서버들 중 하나에 위치하기 때문에 기존에 문제가 되었던 대량 장애 상황에서 특정 청크의 모든 복제본들이 저장된 서버가 한꺼번에 장애가 일어날 확률은 거의 없게 된다. 물론 복제본의 기본 개수에 따라 원래 파일 용양 보다 n 배의 비용이 들긴 하지만, GFS가 운영되고 있는 시스템 자체가 방대하고 저가의 시스템들을 사용하기 때문에 어느 정도 수용 가능했다. 하지만 이 파일 시스템은 설계 시부터 자신의 시스템에 최적화된 형태로 개발되었기 때문에 기본 가정들이 일반 클러스터 시스템에서는 그 의미를 상실하게 하여, 범용적인 클러스터 파일 시스템으로서 GFS를 도입하기엔 적절치 못하다고 할 수 있다.

3. 시간적 지역성에 적응적인 데이터의 고가용성 지원하기 위한 PVFS 확장, HA-PVFS

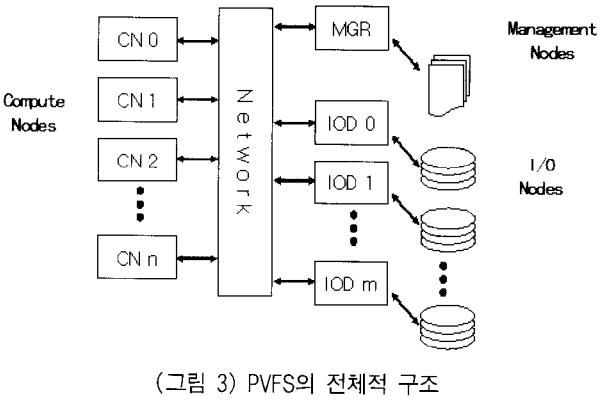
3.1 PVFS

3.1.1 기본 구성 항목과 작업 방법

PVFS는 근래의 대표적인 오픈 소스 병렬 파일 시스템으로 병렬 응용 프로그램의 고속의 파일 데이터 접근을 지원하기 위해 만들어졌다[3]. 이 파일 시스템은 병렬 접근 능력과 더불어 클러스터 단위의 고정된 이름 체계를 지원하고 다중의 입출력 노드들에 저장되는 데이터의 분할 방식을 사용자가 지정할 수 있도록 해준다. 또한 POSIX 표준 입출력 API를 제공하여 현존하는 소프트웨어를 컴파일 과정 없이 그대로 사용할 수 있도록 한다.

PVFS의 전체적 시스템 구조는 (그림 3)과 같다. 직접적인 디스크 입출력을 담당하는 서버로 IOD(I/O Daemon) 노드가 하나 이상 존재하고, 각 PVFS 파일들은 이 IOD 노드에 분할되어 저장된다. 이외에 해당 파일과 관련한 정보인 메타 데이터만을 따로 관리하는 MGR이라는 데몬이 존재하는데 이 데몬은 파일의 생성(Creation), 열기(Open), 닫기(Close) 그리고 지우기(Unlink) 등을 수행한다.

PVFS 시스템은 초기에는 사용자 레벨의 프로그램으로만 제공되었지만, 기존의 소프트웨어들을 컴파일 하지 않고 그대로 사용할 수 있도록 하기 위해서 LINUX VFS 차원의 지원을 하게 되었다. 즉 사용자 프로그램이 기존 시스템 호출 형태로 파일 관련 작업을 명령하면 커널 레벨의 VFS에서 그 명령을 받아 PVFS 커널 모듈에 넘겨준다. 이 명령은 다시 /dev/pvfsd를 통해 유저 레벨의 pvfsd에 전달되어 이 때부터 각 서버들과의 통신으로 PVFS의 작업을 수행하게 된다.



(그림 3) PVFS의 전체적 구조

3.1.2 데이터 저장 방법

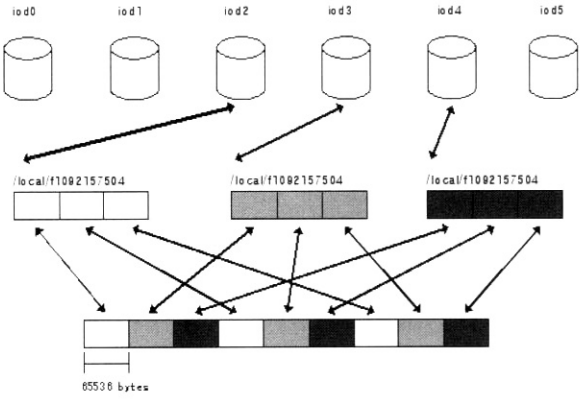
기존의 파일 시스템들은 메타 데이터와 일반 데이터들을 모두 원시 디스크에 그대로 저장하였으나 PVFS에서는 이들 모두를 각각 다른 파일의 형태로 저장한다. 또한 PVFS의 파일들은 모두 하나 이상의 디스크에 분할하여 저장되는데 이를 통해서 병렬 접근의 이득을 얻을 수 있게 된다. 그리고 병렬 프로그램들의 접근 방식을 유연하게 적용하기 위해

분할되는 방식, 즉 어떤 서버에 분할을 하고 어떤 크기로 분할을 할 것인지를 사용자가 설정 가능 하도록 하였다. <표 1>은 이를 지원하기 위한 PVFS의 메타 데이터의 예이다.

〈표 1〉 메타데이터 예제 : /pvfs/foo

inode	1092157504
⋮	⋮
base	2
pcount	3
ssize	65536

여기서 base는 시스템에 존재하는 IOD 중 2번 노드부터 데이터를 저장을 시작할 것이라고 지정하는 역할을 한다. pcount가 3인 것은 2번에서 시작해서 총 2, 3, 4, 세 노드에 분할을 하겠다는 것이고 각 분할의 기본 사이즈를 65546 바이트(ssize)로 지정한다는 의미이다. 결국 이 메타 데이터가 나타내는 바대로 파일을 저장한다면 (그림 4)와 같게 된다.



(그림 4) PVFS 데이터 분할 저장 예제

3.2 HA-PVFS

3.2.1 기본 아이디어와 제반 가정

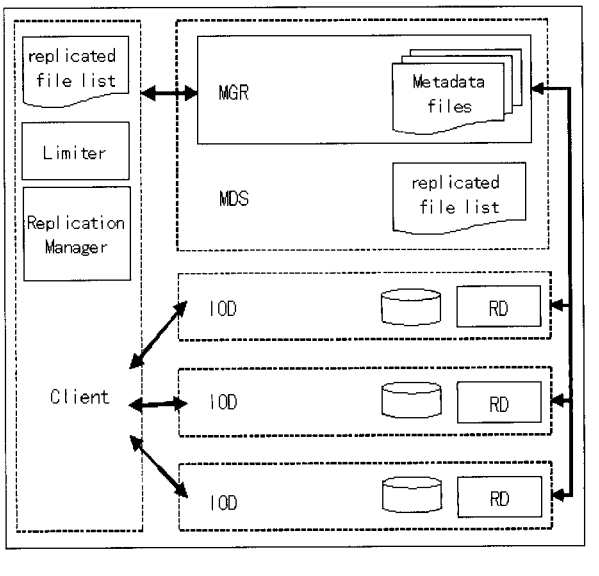
기존의 클러스터 파일 시스템들은 가용성의 대상을 시스템 내의 모든 파일이라고 설정하였다. 하지만 실제로 파일 시스템을 운영하게 되면 그 요청의 상당수는 몇 개의 파일에 몰리게 된다. 특히 상업적인 용도로 사용되는 클러스터 같은 경우 대부분 한두 가지 서비스만 운영하는데 이는 굉장히 높은 지역성을 가지게 된다[9, 10]. 즉, 현재 시스템 내의 파일 중 대부분이 서비스가 지속되는 동안 전혀 접근되지 않고 있음을 의미하는데 기존의 가용성 지원 알고리즘에서는 이 사용되지 않는 파일들이 큰 추가 비용을 초래하게 된다. 따라서 실제로 필요로 하는 파일들에 대한 가용성만을 보장하는 것이 더욱 효율적이고 적절할 것이다. 따라서 본 논문에서 제시하는 고가용성 지원을 위한 알고리즘의 기본 목표는 "파일 시스템의 접근 대상이 되는 파일 중 그 접근 빈도가 높은 파일에 대해서만 가용성을 지원 하겠다"는 것이다.

이후의 절에서는 위의 기본적인 아이디어를 가지고 필요한 파일들만의 가용성을 지원하는 알고리즘을 제시하고 이 알고리즘을 PVFS 파일 시스템에 적용하기 위해 설계한 구성요소에 대해서 설명한다. 이에 앞서 먼저 알고리즘 전개의 편의를 위해 다음의 몇 가지 가정을 하도록 하겠다. 첫째로 제시된 알고리즘이 사용되는 시스템이 인터넷 서비스를 목적으로 구축된다고 가정한다. 즉 파일의 접근에 충분히 높은 시간적 지역성을 가지는 시스템을 가정한다. 두 번째 인터넷 서비스가 나타내는 지역성의 분포가 지수 분포를 따른다고 가정한다. 첫 번째 가정으로 생성된 높은 지역성 특성은 지수 분포가 가지는 특성과 매우 흡사하며 파라미터의 유연한 조정으로 충분히 근사할 수 있음을 유추할 수 있다. 이 가정의 적절성은 4장에서 다시 논의하도록 하겠다. 지수 분포의 몇 가지 좋은 속성이 알고리즘을 수행하는데 매우 큰 시간 및 공간적 이익을 제공한다는 점이 이런 가정의 이유이다.

3.2.2 HA-PVFS 기본 구조

본 논문이 제시하는 가용성 지원 알고리즘을 구현한 HA-PVFS는 복제 관리자(Replication Manager), 한정자(Limiter), 메타데이터 관리 서버(MDS), 복제 파일 리스트(Replicated File List), 다수의 입출력 서버(IOD) 그리고 복제 데몬(RD : Replication Daemon)을 이용하여 동작하며 그 구조는 (그림 5)와 같다.

HA-PVFS 시스템에서 궁극적으로 얻고자 하는 것은 지역성을 갖는 파일들을 클라이언트에서 파악하여 입출력 서버에 복제를 요청하고 이렇게 복제가 되면 자신의 복제 파일 리스트를 갱신하여 다음 작업 때부터 클라이언트가 해당 파일의 복제 여부를 판단하고 특정 서버의 장애 발생 시 장애 없이 동작 하도록 만드는 것이다. 다음절부터는 각 구성요소들이 하는 일들을 설명한다.



(그림 5) HA-PVFS의 기본 구조

3.2.3 복제 관리자(Replication Manager)

복제 관리자가 하는 일은 실시간으로 들어오는 접근 정보와 다음 절에서 설명할 한정자가 제시해 준 한계치를 통해 가용성 지원 정책의 직접적인 명령권을 가진 주체가 된다. 복제 관리자가 구현한 알고리즘은 다음과 같다.

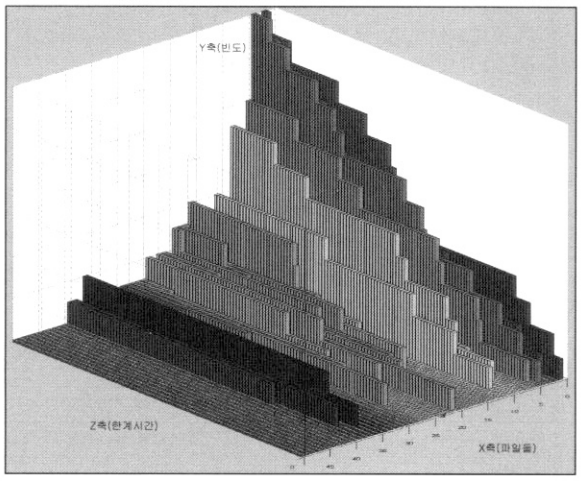
시스템의 파일 시스템 접근 패턴을 정확히 파악하기 위해서는 서비스 시작 시점부터 현재까지의 모든 데이터를 가지고 있어야 한다. 그러나 이는 현실적으로 비용이 너무 높아 불가능하기 때문에 조사 시간에 한계를 두어서 부분적인 최근의 지역성만을 조사한다. 이 시간을 측정 시간 Δt 라고 하면, 접근의 분포나 해당 파일의 크기가 매우 동적이기 때문에 Δt 에 의해 측정된 지역성 분포 또한 변화폭이 매우 크다. (그림 6)은 이 변화를 그래프로 나타낸 것이다.

Δt 를 나타내는 z 축을 따라 그래프를 살펴보면 Δt 가 작을 경우 파일 접근 패턴을 전혀 반영하지 못하는 반면 (실제로 주입한 파일들의 접근 패턴은 파일의 인덱스를 x 축으로 하는 지수분포) Δt 가 크면 클수록 점점 정확하게 지역성을 반영하고 있음을 확인할 수 있다. 하지만 앞에서 말한 바와 같이 무한정 Δt 값을 늘리는 것은 불가능하므로 한계를 지어줄 필요가 있는데 본 논문에서는 이를 한계치로 정의하고 L 로 나타내었다. 이렇게 L 값이 고정되고 나면 식 (1)을 통해 적절한 Δt 값을 찾아낼 수 있다.

$$\Delta t = argmin_{\Delta t}(L - \sum_{A} S(a, \Delta t)) \qquad \text{식 (1)}$$

여기서 함수 S 는 Δt 로 지역성을 조사했을 때 나타나는 파일 a 의 접근 빈도를 말하고 A는 전체 파일의 집합을 의미한다.

이 알고리즘의 구현을 위해서 복제 관리자는 Δt 한계시간을 L 값에 의해 동적으로 설정해야한다. 하지만 파일들의 접근 시간에 따라 순서 연결 리스트를 구성하고 연결 리스트 내 모든 파일 용량의 합이 L 값을 넘지 않게 유지하는 것만으로 충분히 효율적이고 정확한 결과를 낼 수 있었다.

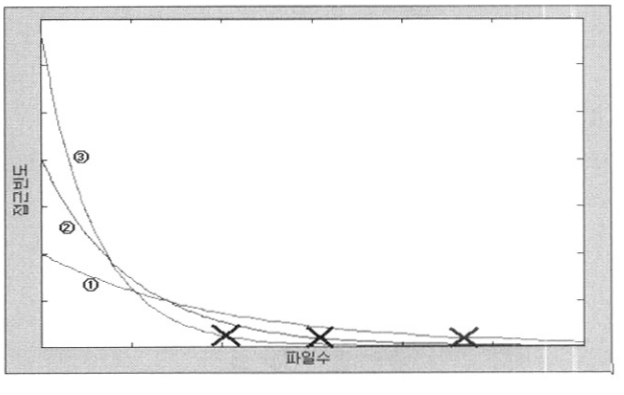


(그림 6) 한계 시간별 지역성

3.2.4 한정자(Limiter)

한정자가 하는 일은 복제 관리자에 의해 필요시 호출되며 현재 시스템 상황을 기반으로 한계치를 재조정하는 것이다. 복제 관리자가 구현한 알고리즘에서 한계치 L 값만 정해지면 적절한 Δt 가 정해지고 정해진 한계시간에 의해서 지역성을 조사할 수 있는 정보를 구해낼 수 있다. 한정자는 이 정보들을 기반으로 실제 접근의 분포 패턴을 추정해 내어 복제할 용량, 즉 한계치를 더 늘릴지 줄일지를 결정하게 된다.

시스템에 들어오는 접근의 분포에 기본적인 지역성이 있다 하더라도 여러 가지 환경에 의해서 그 지역성의 영역은 고정되지 않는다. (그림 7)의 예에서 3가지 서로 다른 접근 분포 중 하나의 패턴이 들어 올 때 시스템은 얼마큼의 파일을 복제해야 하는지를 결정해야 하는데, 그 지표를 수치 하나로 확실히 나타내 줄 수 있어야 한다. 이러한 목적으로 임계치라는 값을 정의 하고 이를 0과 1사이의 실수인 θ 값으로 표시하였다. 이 값이 의미하는 바는 현재 시스템에 접근 빈도가 있는 파일들 중 θ 만큼의 파일들에 대해서만 복제하여 가용성을 유지하겠다는 것이다. 즉 시스템에 총 1000개의 파일이 있는 상황에서 100개의 파일에 접근이 완벽히 몰리는 경우라면 $100 * \theta$ 개 만큼의 파일을 복제하게 된다. (그림 7)의 세 가지 접근 패턴에 나타난 ×지점의 x 축 좌표가 바로 θ에 의해서 결정되는 파일 수가 된다.



(그림 7) 작업 패턴별 가용도 지원 목표

앞에서 시스템에 들어오는 접근은 어떤 지역성을 가지든 지수 분포를 따른다고 가정하였다. 지수 분포는 도착률 λ 라는 하나의 인수로 정의되기 때문에 추정 자체가 매우 쉽다고 알려져 있다. 특히 지수 분포의 판단 최우 추정기(Maximum Likelihood Estimator)가 측정된 표본들의 평균이라는 사실은 널리 알려진 수학적 사실이다. 식 (2)에서 지수 분포의 판단 최우 추정기를 나타내었다.

$$\lambda = \frac{n}{\sum_{i=1}^{n} X_i} \qquad \text{식 (2)}$$

여기서 X_i 는 복제 파일 리스트 내에서 파일들을 접근 빈도에 따라 정렬했을 때 해당 파일의 인덱스를 의미한다. 즉

측정된 인덱스 평균의 역수가 추정 λ가 되는 것이다. 이렇게 구한 추정치를 통해서 입력된 분포라고 예상되는 지수분포를 얻게 되면 임계치 θ와의 관계를 통해 목적하는 곳의 정보를 파악해 낼 수 있는데, <식 3>은 이들의 관계식이다.

$$\theta = \int_0^{x_\theta} \lambda e^{-\lambda t} dt = 1 - e^{-\lambda x_\theta}$$
$$x_\theta = -\frac{1}{\lambda} \ln(1-\theta) \qquad \text{식 (3)}$$

여기서 x_θ는 임계치를 보장하기 위해 가용성을 지켜야 하는 파일의 수를 나타낸다.

한편 지금까지 계산을 통해서 얻고자 하는 것은 한계치 L을 어떻게 조정할 것이냐 하는 것인데 이 값을 찾아내는 것은 앞으로 복제 파일 리스트에 들어올 파일의 용량을 예측하는 일로 확률적인 문제가 된다. 이때 이 변위를 크게 한다면 그만큼 실제 분포를 추정해내는 시간이 줄어들게 되는 대신 안정값에 이르렀을 때의 진동이 우려되고 반대로 아주 작게 변위를 설정하는 경우 적응 시간이 길어지는 단점이 생긴다. HA-PVFS에서는 몇 가지 실험을 통해 식 (4)와 같이 확률적 크기의 차이에 현재의 평균 한계치를 곱해줌으로써 변위를 취하는 중간 정도의 방법을 선택하였다.

$$\begin{aligned} \Delta L &= (F_r(x_c) - F_e(x_\theta))L \\ &= ((1 - f_r(x_c) - (1 - f_e(x_\theta)))L \\ &= (f_e(x_\theta) - f_r(x_c))L \end{aligned} \qquad \text{식 (4)}$$

이때 F 함수는 해당 분포의 CDF를 나타내고 아래 첨자로 표시된 e와 r은 추정한 분포와 실제 복제 파일 리스트에 존재하는 데이터만을 이용해서 얻게 된 분포 함수 있다. 다시 각각을 수식으로 정의 하면 다음의 식 (5)와 같다.

$$f_e(x_\theta) = \lambda e^{-\lambda x_\theta}$$
$$f_r(x_c) = \frac{y}{N} \qquad \text{식 (5)}$$

y는 복제 파일 리스트의 파일들을 접근 빈도에 따라 내림차순으로 정렬했을 때 첫 번째 인덱스부터 더한 파일의 용량이 기존 한계치 L 값의 θ 만큼의 비율이 되는 시점에서 파일 인덱스가 가지는 접근 빈도를 뜻한다. 위의 값들을 모두 구하면 식 (4)에 의해서 한계치의 변위를 구하고 기존 한계치에 더해 새로운 한계치를 구하여 시스템의 복제 정책을 갱신한다.

위에서 제시한 알고리즘은 들어오는 분포에 따라서 적응적으로 시스템의 상한치를 변경한다. 즉 3 메가 크기의 파일 하나에 전체 시스템의 접근이 몰린다면 상한치가 3 메가에 머물게 된다. 그러나 이런 경우 순차적인 파일 시스템 접근에 대해서 빠르게 적응함으로써 오히려 부작용을 낳을 수 있다. 즉 현재 특정 사용자에 의해 a 라는 파일에 대한 접근이 몰린다면 한정자는 이런 상황 파악을 하고 a가 가지는 용량만큼을 시스템 한계치로 설정하려 할 것이다. 다시 일정 시간이 흐른 후 다른 사용자가 b에 대한 접근을 동일한 형태로 하는 경우 다시 시스템은 b에 기준해 상한치를 결정하게 된다. 이 바뀌는 시점의 중간에는 두 파일의 지역성이 비슷한 것으로 파악하여 두 파일 모두를 복제 대상으로 삼게 되지만 일정 시간만 흐르면 자연히 a를 제외하게 된다. 이런 상황이 반복되어 a, b가 계속적으로 대체되어 복제되게 되면 시스템 성능에 해가 될 뿐 아니라 적절한 조치로 보이지 않는다. 이런 상황에 대처하기 위해 한정자는 하한치라는 개념을 추가적으로 정의한다. 즉, 하한치로 특정 용량을 설정해 두면 상한치가 일정 값 이하로 내려가더라도 하한치로 크기를 제한함으로써 적은 양의 파일들이 반복적으로 복제 대상에서 추가되고 삭제되는 것을 방지 할 수 있다.

한정자는 복제 파일 리스트로부터 데이터를 가져와서 위의 알고리즘을 수행하며 수행의 속도를 위해 복제 파일 리스트 내에서 파일 크기순의 리스트를 추가적으로 관리하게 된다.

3.2.5 복제 파일 리스트(Replicated File List)

복제 파일 리스트는 복제된 파일들의 목록을 저장하는 역할을 담당하는데 그 만큼 접근 빈도가 높은 데이터이다. 따라서 이 복제 파일 리스트의 위치가 클라이언트인지 메타 데이터 관리 서버인지에 따라 접근 시간에 큰 차이를 보이게 된다. 현재의 PVFS는 성능 향상을 위해서 읽기/쓰기 작업시 메타 데이터의 중계 없이 바로 입출력 서버에게 접근하게 되는데 이 복제 파일이 메타 데이터 서버에 위치하게 되면 이러한 노력을 해치게 된다. HA-PVFS는 이를 해결하기 위해 클라이언트에 따로 동일한 복제 파일 리스트를 두는 방법을 사용하였다. 이때 메타 데이터 서버에서 복제 파일 리스트를 삭제하지 않는 것은 PVFS가 복수 클라이언트 지원정책을 그대로 수용하였을 때 복제 파일 정보가 전체적으로 관리되어야만 하기 때문이다.

클라이언트의 복제 파일 리스트는 읽기/쓰기 작업 시 항상 접근해야 하는 정보이기 때문에 그 접근 속도에 따라 시스템 성능에 큰 영향을 미친다. 따라서 복제 파일 리스트의 구조를 해시 방식으로 구현하여 접근 속도를 높였다. 그 이외에도 복제 관리자의 편의를 위해서 접근 순서에 따르는 연결 리스트를 추가 하였고 한정자의 편의를 위해서 크기순으로 관리되는 연결 리스트도 제공하였다. <표 2>는 이 복제 파일 리스트의 실제 데이터 구조를 나타낸다.

〈표 2〉 복제 파일 리스트 해시의 데이터 구조

타입	변수명	설명
int64_t	f_ino	파일의 inode 번호
int32_t	f_size	파일의 크기
int32_t	count	접근 빈도 수
int32_t	status	복제 완료 상태
rfl_item_t	*hash_next	해시 충돌 시 다음 항목
rfl_item_t	*next	시간 순서로써 다음
rfl_item_t	*prev	시간 순서로써 이전
rfl_item_t	*up	파일 크기 순서로써 다음
rfl_item_t	*down	파일 크기 순서로써 이전

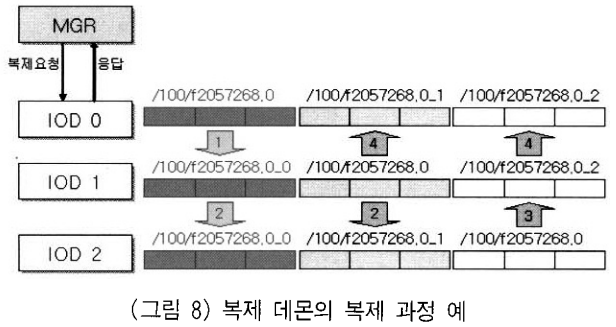
메타 데이터 서버에서 관리하는 복제 파일 리스트도 그 구조가 다르지 않다. 하지만 메타 데이터 서버의 정보는 시스템이 중지되었다 복구 되더라도 정상적인 복제 정보를 알 수 있어야 한다. 따라서 메모리에 관리되는 해시 구조와는 다르게 파일로도 정보를 저장해 주어야 하는데 이 역시 성능 저하의 원인이 될 수 있다. 따라서 본 시스템에서는 실시간으로 파일내의 해당 정보를 검색해서 업데이트 해주는 방식이 아닌 시스템 작업 로그를 파일의 마지막에 추가 시키는 작업으로 대체 시켰다. 이렇게 추가된 로그들은 시스템이 다시 시작하거나 일정 시간이 흐른 후 한 번에 업데이트를 함으로써 실시간 파일 시스템 접근에 의한 성능 저하를 방지하였다.

3.2.6 복제 데몬(RD, Replication Daemon)

복제 데몬은 HA-PVFS의 지연 복제 방식을 지원하기 위해서 추가적으로 만들어진 데몬이다. PVFS는 각 서버 당 하나의 단일 쓰레드로 동작하는데 이 쓰레드가 복제 작업까지 담당하게 되면 기존의 읽기/쓰기 작업을 방해하게 되어 성능 저하를 가져오게 된다. 또한 HA-PVFS는 지연 복제를 지원하는데, 이는 복제의 대상으로 설정되는 시점에 바로 복제 명령을 내리고 그 결과를 기다리는 것이 아니라 복제 명령을 내리고 난 후 그에 대한 작업은 다른 쓰레드에 맡기고 자신은 바로 다른 작업으로 돌아갈 수 있게 하는 것을 말한다. 이를 통해, 최대한 복제의 성공을 지원하면서, 그 복제의 성공을 기다리느라 시스템의 성능을 떨어뜨리지는 않도록 한다. 이 후 일정 시간이 지나 복제가 완료되면 복제 파일 리스트에 메타 데이터 관리자가 데이터를 등록하고, 그 정보가 클라이언트들에게 배포됨으로써 자연스럽게 복제 파일로써 기능하게 된다. 이러한 비동기적인 특성을 지원하기 위해서 복제 데몬이 필요하다.

복제 데몬에게 명령을 내리는 구성 요소는 메타 데이터 관리자이다. 이전에 복제된 일이 없는 파일에 대한 복제 요청이 클라이언트로부터 올 경우 메타 데이터 관리자는 복제 데몬에게 해당 파일의 메타 데이터(분할 정보가 들어있음)와 함께 복제를 요청하게 된다. (그림 8)은 요청을 받은 이후 복제가 이루어지는 예이다.

메타 데이터 서버에서 분할된 파일의 정보를 실제 메타 데이터를 통해서 확인하고 0번 IOD에게 복제를 요청한다. 이와 동시에 메타 데이터 서버는 IOD들의 상태를 첨부하여 보냄으로써 각 IOD들이 어떻게 동작해야 하는 지를 알려주게 된다. 0번 IOD는 요청을 받고 1번 IOD에게 복제를 요청하면서 자신이 가지고 있는 데이터를 보낸다. 1번 IOD는 이 요청과 파일을 받은 후 2번 IOD에게 이전 IOD에서 보낸 모든 데이터를 전달하고 마지막으로 자신이 갖고 있던 파일까지 추가하여 보낸다. 이렇게 순차적으로 전달되어 마지막 IOD에 도착하면, 다시 역순으로 여분의 파일을 자신의 앞 노드로 전달한다. 이렇게 한번 데이터가 전달되고 나면 각 서버들은 동일한 데이터의 복제본을 가지게 되고 동시에 시스템 네트워크 대역폭의 소비를 줄이는 이득도 볼 수 있다. 그리고 각 파일을 일부러 합치지 않는 것은 추가 작업으로 인한 시간지연을 방지하고 기존의 알고리즘 적용을 쉽게 하여 파일에 접근하는 것이 훨씬 수월하기 때문이다.



(그림 8) 복제 데몬의 복제 과정 예

4. 성능 평가

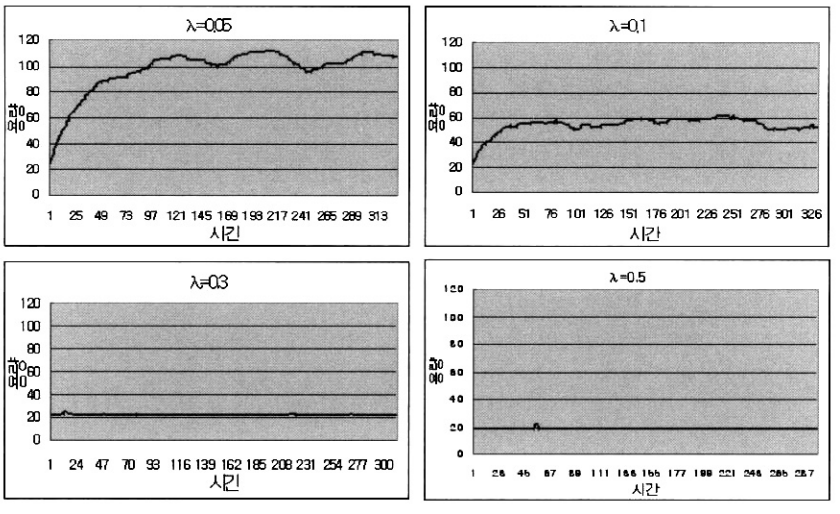
본 장에서는 HA-PVFS의 성능을 측정하고자 입출력을 담당하는 서버 두 대와 메타데이터 관리 서버 1대, 그리고 실제 입출력 작업 명령을 내리는 클라이언트를 분리하여 실험 환경을 구성하였다. 모든 서버와 클라이언트들은 동일한 사양으로 구성되어 있으며 PVFS 시스템의 요구사항 때문에 클라이언트의 커널 버전만 업그레이드하였다.

본 논문이 제시하는 가용성 지원은 기존의 다른 클러스터 파일 시스템에서 제공하는 것과는 다르게 적응적인 변화를 기본으로 하고 있으므로, 입력되는 다양한 분포의 접근 패턴에 대하여 시스템이 어떻게 반응하는 지를 성능의 주요한 측정 대상으로 하고 있다.

접근 패턴에 따른 적응도는 다양한 형태의 지수 분포를 갖는 접근 패턴을 동적으로 생성하여 시스템이 들어오는 분포를 정확하게 추정해 내는 지를 확인하였고, 추정된 분포를 통해 시스템이 한계치를 어떻게 조정하는지를 관찰하였다.

임계값에 따른 적응도 실험을 위하여 요청 패턴 등의 외부 요소를 고정한 후 다양한 임계값을 설정함으로써 임계값에 따라 변하는 시스템의 상황을 관찰하고, 이를 통해 임계값 하나로 시스템의 복제 정책을 유연하게 조절할 수 있음을 확인하였다. 파일크기에 따른 적응도 부분에서는 지역성을 갖는 파일들의 용량이 시스템 인자들에 어떤 영향을 주는지를 측정하였고, 마지막으로 비 지수 분포에 대한 적응도 실험을 통해 본 논문에서 기본적으로 가정하고 있는 "인터넷 서비스 파일 접근 패턴의 지수 분포성"의 적절함을 테스트하였다. 마지막으로 기존 PVFS 파일 시스템과의 응답 시간 비교 측정을 통해 새로 추가된 고가용성 관련 요소가 시스템 성능에 미치는 영향을 측정하였다.

측정의 대부분은 지정된 분포의 파일 시스템 접근 요청을 생성하는 것으로 시작하는데 주로 CDF의 역함수 방법이나 거부법(Rejection Method)을 이용하였다. 이렇게 생성된 연속형의 값들은 특정 파일을 지칭하는 지표가 되어야 하는데 이는 실험에 사용하는 파일들의 이름을 특정 상수로 생성함으로써 사상시킬 수 있도록 하였다. 따라서 각 파일들은 자신의 이름에 나타난 상수부터 그 상수보다 1이 큰 상수까지



(그림 9) 다른 종류의 분포에 대한 한계치 변화

의 확률적 빈도를 대표하게 된다.

4.1 적응도 측정

4.1.1 접근 패턴에 따른 적응도 변화

(그림 9)는 측정된 λ 값을 기본으로 한계치의 변화를 보여주는 그래프이다. 한계치가 어떻게 설정 되느냐에 따라 복제되는 파일의 양도 설정된다. 그래프를 보면 모든 λ 값에 대해 적절한 값으로 수렴한 후 진동하는 모습을 보인다. 여기서 λ가 작다는 것은 그만큼 지역성을 보이는 파일이 많다는 뜻이고, 따라서 그 파일들을 모두 수용하기 위해 설정되는 용량의 한계치가 자연스레 늘어난다. 그래프에서 $\lambda = 0.05$ 일 때 평균 100 메가로 한계치가 설정되는데 비해 $\lambda = 0.1$ 일 경우 60으로 한계치가 설정되는 것은 이런 이유 때문이라고 할 수 있다. 또한 λ 값을 0.3으로 한 분포에서부터 한계치가 특정값 이하로 떨어지지 않는 것은 앞에서 설명한 하한치에 의해 일어나는 현상으로, 일정 수준 이상의 지역성을 가지게 되면 하한치의 범위 내에서 모두 복제가 되어 추가적인 복제 없이 동작하는 것을 보여준다.

이 논문에서 최종적으로 얻고자 하는 것은 접근하는 파일 중 지정된 비율 만큼 가용성을 지원해 주는 것이다. 이를 얼마나 만족하는 지 확인하기 위하여 오차율을 계산했는데, 여기서 오차율이란 실제 들어오는 접근의 분포를 안다고 가정했을 때 주어진 임계치에 의해 보장해 줘야 하는 파일의 수에서 실험적으로 측정된 복제 파일의 수를 나누어 그 비율을 구한 것이다. 이때 실제로 보장해야 하는 파일의 수는 식 (6)과 같다.

$$file\,number = \left\lfloor -\frac{1}{\lambda}\ln(1-\theta) \right\rfloor \qquad \text{식 (6)}$$

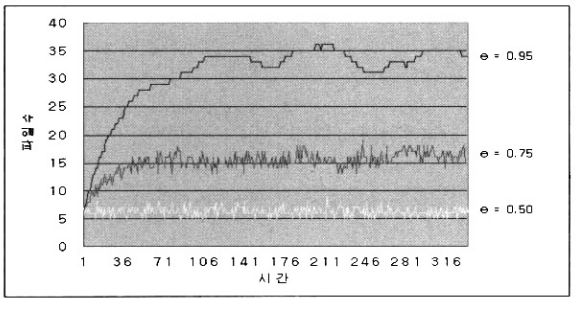
각 분포에 대한 오차율은 위의 <표 3>과 같다. λ 값이 증가하면, 즉 접근의 지역성이 높을수록 원래 의도했던 만큼의 가용성을 보장하게 되지만 지역성이 떨어질수록 원래 지원해야 하는 양에 못 미치는 양만 복제하게 된다. 이러한 현상이 생기는 것은 앞에서 설명한 바와 같이 편중되고 제한된 정보에 의해서 λ 값을 제대로 추정해 내지 못하기 때문이다. 그러나 <표 3>에서 보이는 오차율은 특정 λ 값에 대해서 고정된 값을 보여주는데 이는 추정된 λ 값에 따라 일정 팩터를 곱해줌으로써 지정한 만큼의 가용성을 좀 더 정확히 지원할 수 있음을 알려준다. 그러나 현재 이 논문이 목적하는 바는 지역성의 95% (특정 값) 만큼의 복제 지원이라기보다는 95% 정도의 복제 지원의 의미가 크므로 정확한 값을 맞추기 위해서 비용을 지불하는 것은 큰 의미가 없다고 생각된다.

〈표 3〉 λ 값에 따른 한계치 오차

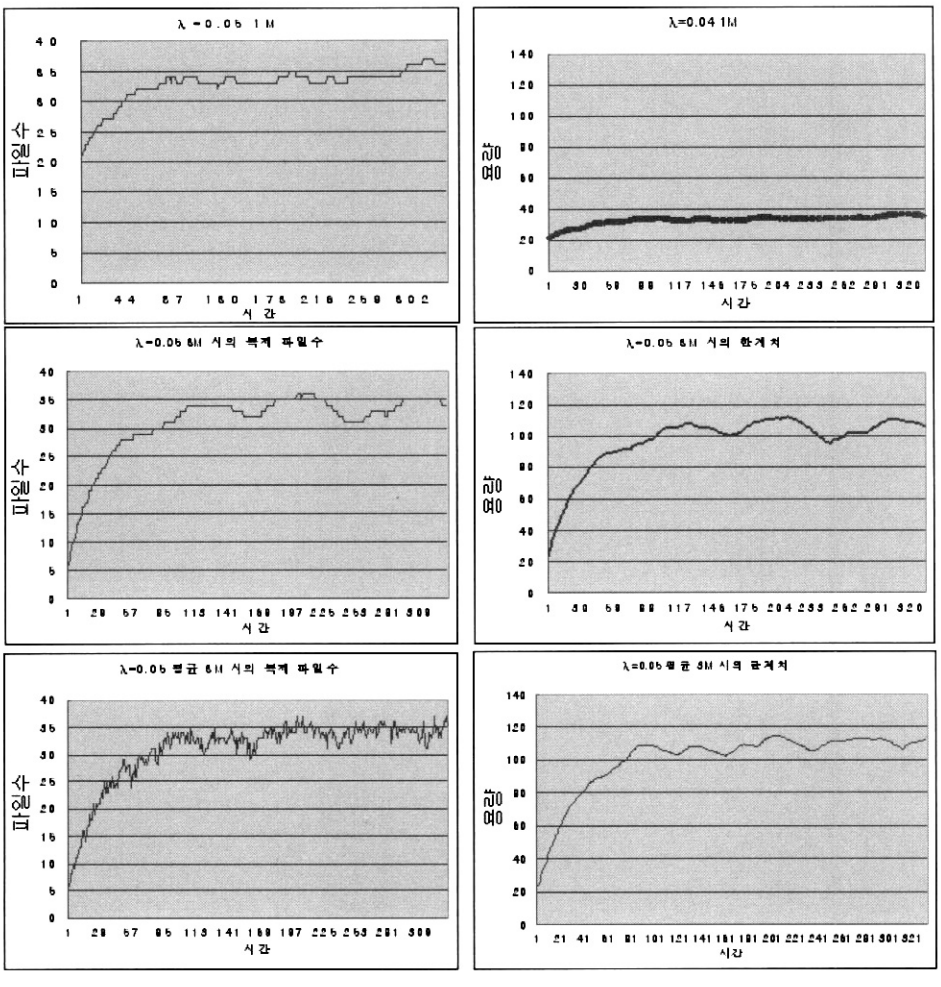
	0.05	0.1	0.2	0.4	0.5
평균파일수	33.5	17.60	9.69	6.22	6.01
보장파일수	59.91	29.96	14.98	7.49	5.99
오차율	1.79	1.70	1.54	1.89	0.99

4.1.2 임계값에 따른 적응도 측정

임계값 θ는 이 값에 의해 시스템에서 사용하는 모든 값들이 규정되고 정의 내려지게 되는, 시스템의 작동 방법을 규정하는 중요한 상수이다. (그림 10)은 θ 값의 변화가 한계값에 미치는 영향을 표시한 것이다. θ가 줄어들수록 유지하고자 하는 지역성의 범위가 줄어드는 것이고 그만큼 적은 양의 파일의 복제를 지시하게 된다.



(그림 10) 다른 분포에 따른 복제 대상 파일 수



(그림 11) 파일 크기에 따른 복제 대상 파일 수와 한계치

4.1.3 파일크기에 따른 적응도 측정

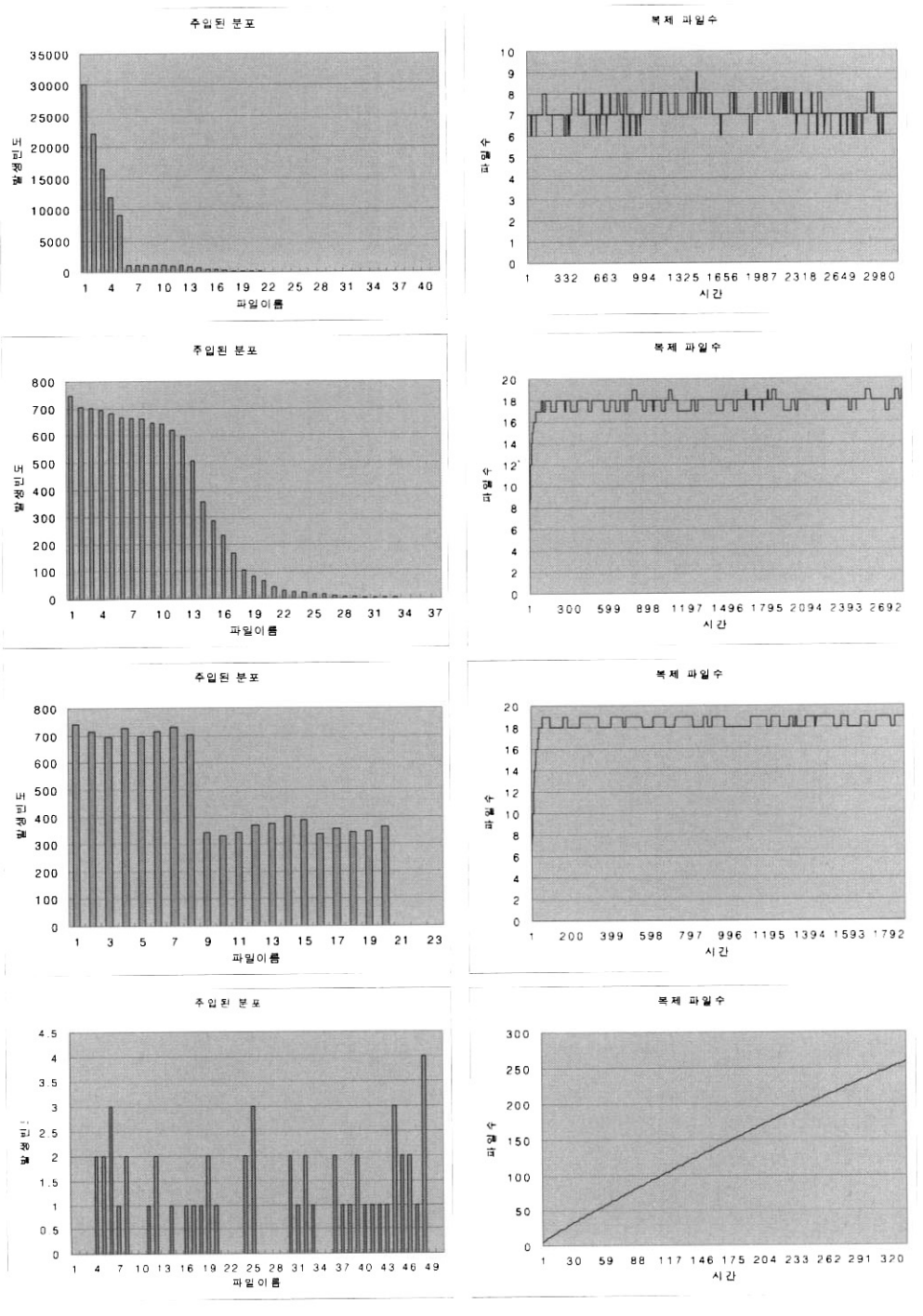
(그림 11)은 시스템에 존재하는 파일들의 크기에 따라서 HA-PVFS가 어떻게 반응 하는지를 표시한 것이다. 좌측의 그래프들은 복제 대상 파일 수를 나타내고, 우측의 그래프는 설정된 한계치를 보여준다. 최상단의 그래프는 파일의 크기를 1MB, 중간의 그래프는 3MB 로 고정하여 실험한 결과이다. 파일수의 측정에서는 두 가지 경우 비슷한데, 이는 HA-PVFS 복제 알고리즘이 파일 단위의 접근 분포를 대상으로 동작하기 때문이다. 그러나 한계치 면에서는 상당한 차이를 보이고 있는데, 복제 대상 파일의 수는 고정되지만, 크기가 큰 파일의 경우 한계치의 용량이 비례해서 커져야 하기 때문이다.

마지막 세 번째의 그래프는 평균이 3MB인 여러 가지 크기의 파일을 균일하게 분포 시키고 실험을 추가로 진행한 결과이다. 파일의 크기를 3MB로 고정시킨 실험의 결과인 중간 그래프와 비교했을 때 수량이나 용량 면에서 거의 차이가 없다. 즉, 복제 파일 리스트의 한계치는 시스템에서 자주 접근되는 파일의 평균 용량에 의해 결정됨을 알 수 있다. 그러나 파일의 크기가 다양한 경우 복제 파일 리스트로 관리하게 되는 파일의 수가 빈번하게 변하게 되는데, 이는 큰 용량의 파일에 대한 접근 빈도가 떨어져 리스트에서 제거되게 되는 경우 작은 용량의 파일들이 여럿 들어올 수 있기 때문이다.

4.1.4 비 지수 분포 접근에 대한 적응도 측정

파일에 대한 접근은 감소함수의 형태를 띠긴 하겠지만 모두가 지수함수라고 한 가정은 사실이 아니다. 따라서 지수분포가 아닌 형태의 접근 방법에 대해서 HA-PVFS 복제 알고리즘이 어떻게 반응 하는지에 대해서 실험해 보았다.

(그림 12)의 왼쪽 그래프들은 입력 분포로 사용할 두 개 이상의 분포를 합친 형태와 계단 형태의 분포를 나타내고, 오른쪽 그래프는 그 결과 복제 파일 리스트에 속하는 파일 수를 나타낸다. 최상단의 그래프는 6개의 파일에 지수적인 접근을 보이다가 갑자기 접근수를 떨어뜨리는 분포인데 복제 파일의 수가 6~7개로 설정된다. 다음 그래프의 분포는 처음 14개 정도의 파일에는 완만한 접근율을 보이다가 떨어지는 형태의 분포인데 17~18개 정도에 적응하게 된다. 다음의 그래프는 가장 극단적인 형태의 분포로 8개까지의 파일에 일정한 접근율을 보이고 8~20개의 파일 사이에는 반정도 되는 접근율을 보이는 계단 형태의 분포로, HA-PVFS

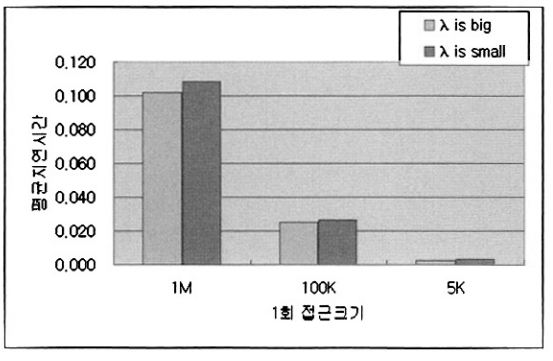


(그림 12) 비 지수 분포 요청에 대한 복제 파일 수 변화

는 18~19개의 파일의 가용성을 보장하고 이것 역시 적절하다고 볼 수 있다. 마지막으로 시스템 전체 파일에 균일한 접근율을 보이는 분포에서는 파일의 개수가 계속적으로 증가한다.

4.2 응답시간 비교

HA-PVFS는 기존의 PVFS 시스템의 내부는 전혀 수정하지 않고 새로운 가용성 관련 코드만 추가한 것이기 때문에 기존 시스템과 비교했을 때 성능 저하가 예상된다. 총 1000개의 파일을 대상으로 요청을 생성하는데 측정 시작 시 모든 파일에 대한 열기 시스템 호출에 의한 지연을 배제하였고 지정된 형태의 분포로 임의 숫자를 생성해서 해당 인덱스의 파일에 특정 크기만큼 지정된 비율(읽기 : 9, 쓰기 : 1)의 작업을 반복 수행하였다. 여기서 비록 읽기, 쓰기 비율을 고정하였지만 해당 알고리즘이 철저히 접근 빈도에 의해서 결정되므로 비율은 큰 의미를 가지지지 않는다. (그림 13)은 그 실험 결과를 나타낸다.



(그림 13) HA-PVFS의 응답 시간

요청이 가지는 지역성에 따라 복제의 수가 결정되기 때문에 지역성이 낮을수록 실제 파일 시스템 작업에 대한 방해가 크리라 생각하고 λ 값을 다르게 하여 요청 분포를 주입시키는 실험을 하였다. $\lambda = 0.5$인 높은 지역성을 보이는 분포와 $\lambda = 0.05$인 산개된 형태의 분포에 대한 응답시간을 측정하였는데, λ 값이 작은 경우 복제되는 파일의 수가 많아져 복제에 대한 영향이 커지게 되고 따라서 지연시간 또한 늘어남을 확인할 수 있었다. 만약 전체 복제 방법을 제반 환경과 동일하게 한 후 측정한다면, 이는 λ 값이 매우 낮게 설정된 상태와 비슷하다고 할 수 있다. 즉 응답시간에 있어서 HA-PVFS의 알고리즘은 지역성이 높은 경우 전체 복제 방법에 비해 충분한 이익을 얻을 수 있음을 확인할 수 있다.

5. 결 론

기존의 데이터 고가용성 지원을 위한 클러스터 파일 시스템들은 대부분 전체 파일 시스템에 대한 가용성만을 고려했기 때문에 불필요한 공간 및 성능적인 비용을 지불해야만 했다. 또한 이런 비용에 대한 부담으로 특정 개수 이상의 디스크 또는 노드의 장애 상황에도 적절하게 대비하지 못하는 문제가 있었다. 본 논문은 이러한 기존 파일 시스템의 단점을 해결하기 위하여 파일에 중요도라는 개념을 도입하고 그 중요도는 파일에 대한 접근 빈도, 즉 지역적 특성을 통해서 측정하였다. 그러나 지역성에 대한 정보를 모두 저장할 수 없기 때문에 제한적인 정보를 통해 실제 접근 패턴을 정확히 판단하기 위해, 인터넷 서비스의 파일 시스템 접근 패턴을 지수 분포라고 가정하고 판단 최우 측정기를 사용하여 그 접근 패턴을 추정하였다. 또한 복제에 의해 나타나는 성능 저하를 최소로 줄이기 위해 지연 업데이트 방식과 릴레이식 복제 방식을 사용하였다.

성능 측정을 통해 제시된 알고리즘이 적절하게 동작하는지 확인하였다. 우선 몇 가지 지수 분포를 주고서 적응도를 측정해 본 결과 오차는 있지만 정해진 방향성을 보이며 적절한 수의 파일을 복제해 가용성을 유지해 주는 모습을 보였다. 또한 비 지수 분포의 일반적인 예에서도 시간적 지역성을 보이는 파일에 대한 복제가 적절히 일어남을 볼 수 있었다. 응답시간 측면에서도 파일들의 시간 지역성이 높은 경우 전체 복제 방법에 비해 좋은 성능을 보임을 알 수 있었다.

위 측정된 결과를 통해서 HA-PVFS가 시스템 내에 존재하는 파일들의 중요도를 지역성의 기준으로 적절하게 판단하고 있음을 확인할 수 있었다. 따라서 파일 시스템에 대한 접근 패턴에 지역성이 아주 높은, 단일 서비스를 제공하는 일반적인 상용 시스템들에서 제안하는 HA-PVFS 파일 시스템이 효율적으로 데이터의 가용성을 제공할 수 있음을 알 수 있다.

앞으로의 연구 과제는, 현재 구현된 HA-PVFS가 다중 클라이언트 환경에서도 효율적으로 가용성을 지원한다는 것을 테스트를 통해 보이는 것이 필요하다. 더 나아가 지금까지의 측정은 모두 가상으로 생성된 요청으로 이루어 졌으므로, 실제 인터넷 서비스를 운용하고 있는 클러스터 시스템에 HA-PVFS를 설치 후, 가용성 지원의 적절성을 실제적으로 파악하는 것이 필요하다.

참 고 문 헌

[1] R. Sandberg, "The SUN Network File System : Design, Implementation and Experience", SUN Microsystems, Inc., pp.119-130, 1985.

[2] Howard J. H. "An Overview of the Andrew File System", Proceedings of the USENIX Winter Technical Conference, pp.23-26, Feb., 1988.

[3] P. Carns et al. "PVFS : A parallel file system for linux clusters." In Proceedings of the 4th Annual Linux Showcase and Conference, pp.317-327, 2000.

[4] Peter J. Braam et al., "The Lustre Storage Architecture", Cluster File System. Inc, Mar. 2003

[5] K. W. Preslan et al., "A 64 Bit, Shared Disk File System for Linux", Proceedings of the 16th IEEE Mass Storage Systems Symposium, pp.22-41, 1999.

[6] F. Schmuck et al. "GPFS : A Shared-Disk File System for Large Computing Clusters", Proceedings of the FAST Conference on File and Storage Technologies, pp.231-234, 2002.

[7] J. H. Hartman et al. "The Zebra Striped Network File System", ACM Transactions on Computer System(TOCS) Vol.13, Issue3, pp.274-310, August, 1995.

[8] D. A. Patterson et al. "A Case for Redundant Arrays of Inexpensive Disks(RAID), Proc. of the ACM Conference on Management of DATA(SIGMOD), pp.109-116, June, 1988.

[9] L. Cherkasova et al "Dynamics and Evolution of Web Sites: Analysis, Metrics and Design Issues", Proceedings of the Sixth IEEE Symposium on Computers and Communications (ISCC), p.64, 2001.

[10] L. Cherkasova et. al "Analysis of Enterprise Media Server Workloads: Access Patterns, Locality, Content Evolution, and Rates of Change", IEEE/ACM Transaction on Networking, Vol.12, No.5, pp.781-794, October, 2004.

[11] S. Ghemawat et al. "The Google File System", Proceedings of the 19th ACM Symposium on Operating System Principles, October, 2003.

심 상 만

e-mail : yeuri@sogang.ac.kr
2003년 서강대학교 컴퓨터학과(공학사)
2005년 서강대학교 컴퓨터학과(공학석사)
2005년~현재 삼성전자 Mobile Platform Sol. Lab 연구원
관심분야 : 클러스터 파일시스템, 분산처리 시스템

한 세 영

e-mail : syhan@sogang.ac.kr
1991년 포항공과대학교 수학과(이학사)
2003년 서강대학교 정보통신대학원 (공학석사)
2004년~현재 서강대학교 컴퓨터학과 박사과정
1996년~현재 (주)이엔지 기술본부 부장
관심분야 : 피어투피어 컴퓨팅, 분산처리 시스템

박 성 용

e-mail : parksy@sogang.ac.kr
1987년 서강대학교 컴퓨터학과(공학사)
1994년 미국 Syracuse University(공학석사)
1998년 미국 Syracuse University(공학박사)
1998년~1999년 미국 Bell Communication Research 연구원
1999년~현재 서강대학교 컴퓨터학과 부교수
관심분야 : Autonomic Computing, Peer to Peer Computing, High Performance Cluster Computing and System