2000년 한국정보처리학회 추계 학술발표논문집 제7권 제2호

파일 자원 사용 빈도를 고려한 클러스터 파일시스템의 성능 향상

최창열, 정지영, 김성수
아주대학교 정보통신전문대학원 정보통신공학과
e-mail : {clchoi, abback, sskim}@madang.ajou.ac.kr

Performance Improvement of Cluster File System Regarding Usage Frequency of File Resources

Changyeol Choi, Ji Yung Chung and Sungsoo Kim
Professional Graduate School of Informtion and Communication Technology,
Ajou University

요약

공유 저장장치에 접근할 수 있는 경로를 다중화함으로써 클러스터 파일시스템 내 한 노드에서 발생하는 결함을 허용할 수 있다. 공유 저장장치의 다중접근은 파일 이동과 파일 중복으로 이루어진다. 본 논문에서는 전역 파일 시스템의 평균 응답 시간을 줄이기 위해 파일 자원의 접근률에 따라 우선 순위를 두어 파일 이동과 파일 중복 시 발생할 수 있는 클러스터 파일 시스템의 성능 저하 및 서비스 중단을 막기 위한 방법을 제안한다. 파일 자원의 접근률은 파일 처리 시에 사용되는 블록, 아이노드(inode), 슈퍼블록과 같은 자원 사용 빈도를 통해 얻어질 수 있다.

1. 서론

인터넷, e-비지니스와 같은 분야에서 서비스 제공자와 사용자수가 증가함에 따라 처리해야할 데이터량이 점차 증가하고 있다. HPFS, EXT2와 같은 기존 파일 시스템의 구조와 성능으로는 이러한 요구사항을 만족시킬 수 없으며 이로 인하여 부각되고 있는 것이 클러스터 파일 시스템이다. 클러스터 파일 시스템이란 기존 PC 또는 워크스테이션을 네트워크로 연결하여 고성능 서버의 성능을 발휘하는 시스템을 제작하는 것이다. 클러스터 파일 시스템이 만족해야 하는 조건으로 고성능을 비롯하여 효율적인 자원 관리와 클러스터 시스템의 확장성, 가용성, 신뢰성이 높아야 한다[1]. 현재 이러한 클러스터 파일 시스템의 특성을 만족시키기 위한 다양한 형태의 클러스터 파일 시스템이 연구, 개발 중이다. 네트워크 매체의 전송 속도의 한계로 인하여 노드와 노드 사이의 병목현상이 발생하는 것에 대처하기 위해 제안된 것이 공유 저장장치 파일 시스템이다[2]. 그러나 지금까지 클러스터 파일 시스템의 일관성(consistency) 유지를 위한 것과 클러스터 파일 시스템 내 한 노드에 결함이 발생하였을 경우 효율적인 에러 복구를 위한 연구는 많이 진행되었으나 클러스터 파일 시스템의 성능 저하와 결함을 예측하여 결함이 발생하지 않도록 하는 연구는 미미한 실정이다. 본 논문에서는 전역 파일 시스템의 평균 응답 시간을 줄이기 위해 파일의 접근률에 따라 우선 순위를 두어 파일 이동과 파일 중복 시 발생할 수 있는 클러스터 파일 시스템의 성능 저하 및 서비스 중단을 막기 위한 방법을 제안한다. 본 논문의 2장에서는 관련 연구를 언급하고, 3장에서는 자원 사용 빈도에 기반한 파일 이동과 파일 중복을 설명한다. 4장에서는 시뮬레이션을 통하여 본 논문에서 제안한 방법의 성능 평가를 수행하며 5장에서 결론을 내린다.

This work is supported in part by the Ministry of Information & Communication of Korea.("Support Project of University Foundation Research<2000>" supervised by IITA)

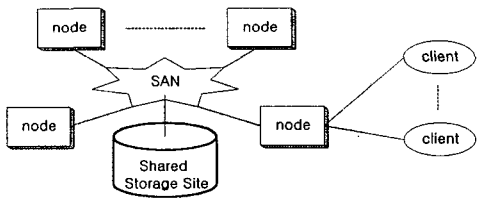
2. 관련연구

2.1 클러스터 파일 시스템

클러스터 시스템을 지원하기 위한 분산 파일 시스템은 크게 메시지기반 파일 시스템과 공유 저장장치 파일 시스템으로 나눌 수 있다. 메시지기반 파일 시스템에는 NFS(Network File System), Sprite 파일 시

스템, AFS(Andrew File System), Coda 파일 시스템, xFS 등이 있으며, 공유 저장장치 파일 시스템은 VAXcluster VMS, SFS(Serverless File System), GFS(Global File System) 등이 있다. 공유 저장장치 파일 시스템은 메시지기반 파일 시스템 보다 확장성에서는 떨어지나 클러스터 시스템 내의 부하균등(load balancing)과 지역성(locality)을 유지하기가 쉽다.

GFS(Global File System)는 미네소타 대학 중심 그룹에서 개발된 저장장치 기반 Serverless 클러스터 파일 시스템이다[2]. GFS는 대용량 저장장치와 대역폭을 이용하는 분야를 목표로 개발되었다. GFS 클러스터 환경은 그림 1과 같다. 그림 1에서 공유 저장장치 사이트는 상이한 공유 저장 장치들을 논리적으로 하나의 저장장치로 통합하는 드라이버인 NSP(Network Storage Pool)로 관리가 되어진다. NSP는 Atomic read-modify-write 오퍼레이션을 통한 locking 메커니즘을 이용하여 데이터 및 파일의 일관성을 유지하여 클러스터의 동시성 메커니즘도 제공하고 있다. 또한 NSP는 여러 개의 하위 풀(subpool)로 구성되어 있으며, 하위 풀은 디스크 배열(disk array), 디스크 스트라이핑(disk striping)으로 이루어져 있다.



(그림 1) 공유 저장장치 전역파일시스템 환경

2.2 파일 이동과 파일 중복

공유 저장장치에 접근할 수 있는 경로를 복수 개로 다중화함으로써 클러스터 파일시스템 내 한 노드에서 발생하는 결함을 허용할 수 있다. 다중 접근(multiple access)은 복수 개의 접근 경로를 통하여 복수 개의 시스템에서 공유 저장장치의 자료를 사용할 수 있도록 한다. 이 기능은 클러스터 시스템 내 한 노드에서 발생하는 결함으로 인하여 서비스가 중단되는 것을 방지할 수 있다[5]. 공유 저장장치에 서로 다른 복수 개의 노드에서 동시에 접근할 수 있도록 접근 경로를 설정함으로써 한 노드에 결함이 발생할 경우 다른 노드를 통하여 같은 파일에 대한 접근이 가능하도록 하는 것이다. 한 노드에 결함이 발생하면 다중 접근 경로가 설정되어 있는 다른 정상적인 노드로 결함이 발생한 노드에서 수행하던 작업을 이동시켜 주는 기능이 있어야 한다. 시스템이 동작하면서 이용하게 되는 중요한 자원은 파일이다. 따라서 파일에 대한 정보와 파일 관련 자원을 복제함으로써 파일 결함에 의한 손실을 줄일 수 있다. 이것은 분산된 다른 노드에서도 자원을 보관함으로써 한 노드의 결함 발생으로 인한 서비스 중단을 막을 수 있다. 이를 다중 자원(multiple resource)이라 한다. 이와 같이 다중 자원 방법을 이용하면 동일한 파일에 대한 캐쉬 자원을 다른 노드에서도 보관할 수 있다. 또한 공유 저장장치 내의 파일을 메모리 안에 있는 캐쉬에 보관함으로써 디스크 접근에 따른 오버헤드를 줄일 수 있어 파일 처리에 대한 평균 응답 시간도 줄일 수 있다. 파일에 대해서 자원을 다중화하는 것은 파일 이동과 파일 중복에 의해서 이루어진다. 파일 이동과 파일 중복을 함께 수행하여 클러스터 파일 시스템의 요구사항인 성능과 가용성 향상을 시키는 것을 Kure, Lui Sheng 그리고 Hurley 등이 제안하였다. Kure, Lui Sheng은 현재 시스템의 상태 정보를 이용하는 방법[3]을 제안하였고 Hurley는 큐의 길이를 고려한 방법[4]을 제안하였다. Kure, Lui Sheng이 제안한 방법은 시스템의 동작을 평가하여 파일 이동과 파일 중복을 수행할 노드를 선택하기 위한 계산이 너무 복잡하고 시간이 많이 걸린다. Hurley가 제안한 방법은 시간 복잡도에서 향상된 알고리즘이나 파일 이동과 파일 중복 시 단지 큐의 길이와 파일 요청 수만을 고려했기 때문에 한 노드에 파일 이동이 집중되는 단점이 있다. 본 논문에서는 파일 자원의 접근률에 따른 손실에 따라 우선순위를 두어 한 노드에 집중되는 현상을 제어하기 위한 방법을 제안한다.

3. 파일 이동과 파일 중복 전략

파일 이동(file migration)은 한 노드에서 처리하던 파일을 다른 노드에서 수행하도록 하는 과정이다. 파일 이동은 한 노드에 결함이 발생하였을 때 서비스를 계속 수행할 수 있도록 하기 위해 진행되거나 클러스터 시스템의 성능을 향상시키기 위해 진행된다. 파일 중복(file replication)은 파일 이동과 비슷한 과정으로 진행되는데 원본 파일을 지우지 않는다는 점에서 다르다. 파일 이동과 파일 중복 정책에서 고려해야 하는 것은 언제, 어느 파일을, 어느 노드로 진행

할 것인가를 결정하는 것이다. 평균 응답 시간을 최소로 하는 노드로 파일 이동과 파일 중복을 수행함으로써 클러스터 파일 시스템의 성능을 향상시킬 수 있다. 평균 응답 시간에서 얻을 수 있는 순수한 이득(NG_i)은 큐 대기 시간을 줄여 얻을 수 있는 이득(R_i)과 파일 이동이 한 노드에 집중됨에 따른 손실시간(L_i)을 빼서 구한다.

$$NG_i = R_i - L_i \quad (1)$$

현재 서비스 중인 노드보다 다른 노드의 큐 대기 시간이 적다면 파일 처리를 그 노드에 이동하여 처리하도록 함으로써 평균 응답 시간을 줄이는 것이다. 일반적으로 파일 이동을 위한 처리에 대한 비용은 파일 처리를 위한 비용보다 적기 때문에 평균 응답 시간에서 이득을 얻을 수 있다. 파일 이동을 위한 비용은 파일 처리를 위한 비용의 5%에 지나지 않는다. 원래 노드의 파일 요청 대기를 위한 큐의 길이를 l_s, 목적 노드의 파일 요청 대기를 위한 큐의 길이를 l_d, 원래 노드에서의 파일 i에 대한 총 파일 요청 수를 n_i, 파일 i에 대한 j번째 파일 요청을 위한 큐의 대기 위치를 $X_{i,j}$, 파일 요청을 처리하기 위해 걸리는 평균 시간을 C_p, 파일 이동을 수행하는데 걸리는 평균 시간을 C_m이라고 할 때, 큐 대기 시간을 줄여 얻을 수 있는 이득(R_i)을 계산하는 것은 식 (2)와 같다.

$$\begin{aligned} R_i &= C_p \sum_{j=1}^{n_i} [(X_{i,j+1} - X_{i,j} - 1)j + (X_{i,j} - l_d - j)] \\ &\quad - C_m(l_s + l_d) \\ &= C_p n_i(l_s - l_d - n_i) - C_m(l_s + l_d) \end{aligned} \quad (2)$$

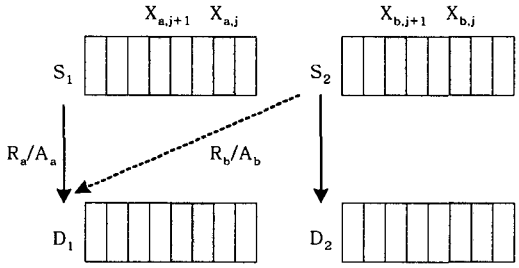
노드의 수가 증가할수록 한 시점에 R_i값이 동일한 노드가 발생할 수 있다. R_i값이 동일하므로 두 개 이상의 노드에서 한 노드로 파일을 이동하는 부하 집중 현상이 발생한다. 그림 2에서와 같이 노드 S_1과 노드 D_1의 R_i값과 노드 S_2와 노드 D_1의 R_j값이 같다면 노드 S_1과 노드 S_2는 노드 D_1으로 파일을 이동하게 된다. 따라서 노드 S_1, S_2 중 한 노드는 R_i값 만큼 이득을 볼 수 있지만 다른 노드는 R_j값 보다 적은 이득을 보거나 최악의 경우 손해를 볼 수 있다. 두 노드 이상에서 한 노드로 집중됨으로써 발생하는 손실 비용(C_a)에 파일 자원의 접근률(A_i)에 따라 가중치를 부여해 손실 비용이 적은 노드만 파일 이동을 수행하게 한다. 여러 노드에서 한 노드로 집중됨으로써 발생하는 손실(L_i)은 파일 자원의 접근률에 따라 가중치를 부여해 구한다.

$$L_i = C_a * \frac{1}{A_i} \quad (3)$$

파일 자원 접근률(A_i)은 파일 처리 시 사용되는 자원인 블록, 아이노드, 슈퍼블록의 사용 빈도수에 의해서 구할 수 있다. 파일 관련 자원에 대한 정보는 로그 파일에 저장한다. 식 (1)에 R_i와 L_i를 대입하면 평균 응답 시간에서 얻을 수 있는 순수한 이득(NG_i)을 식 (4)와 같이 구할 수 있다.

$$NG_i = C_p n_i(l_s - l_d - n_i) - C_m(l_s + l_d) - C_a * \frac{1}{A_i} \quad (4)$$

그림 2에서 파일 a에 대한 접근률(A_a)이 파일 b에 대한 접근률(A_b)보다 크다고 가정하자. 또한 S_1과 D_1 사이의 큐의 길이에 의한 평균 응답 시간의 이득(R_a)과 S_2와 D_1 사이의 큐의 길이에 의한 평균 응답 시간의 이득(R_a)이 동일하다고 가정하자. R_a와 R_i가 동일하므로 로그 파일에서 파일 자원의 접근률에 대한 정보를 얻어 NG_i를 구한다. 접근률이 높은 노드 S_1은 손실(L_i)이 S_2보다 작으므로 노드 D_1로 파일 이동을 수행할 권리를 부여받게 된다. 노드 S_2는 노드 D_2로 파일 이동을 수행하게 된다. 따라서 노드 D_1로 파일 이동을 수행하는 것 보다 파일 응답 시간이 줄어들게 된다.



(그림 2) 큐의 길이와 파일 접근률을 고려한 파일 이동의 예

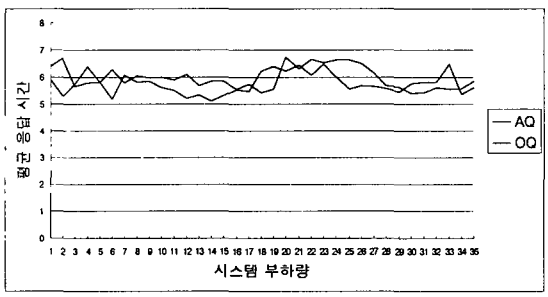
4. 성능 평가

이 장에서는 큐의 길이와 파일 자원의 접근률을 고려한 파일 이동 정책의 성능 평가를 위하여 시뮬레이션을 수행한다. 파일 요청에 대한 분포는 지수분포를 가진다고 가정하였다. 파일 이동을 위한 평균 시간, 파일 처리를 위한 평균 시간과 한 노드에 파일 이동이 집중되어 생기는 손실 비용은 각각 C_m, C_p, C_a를 평균값으로 갖는 지수분포를 가진다고 가정하였다. 새로운 파일 요청이 들어왔을 때, 파일 요청 처리가 다 되었을 때, 파일 이동이 발생하였을 때 큐의 길이인 l_s와 l_d에 변화가 생긴다. 따라서 l_s와 l_d도 지수 분포를 가진다. 또한 파일 자원의 접근률(A_i)에 대한 분포도 지수분포를 따른다고 가정하였다. 파일 자원의 접근률

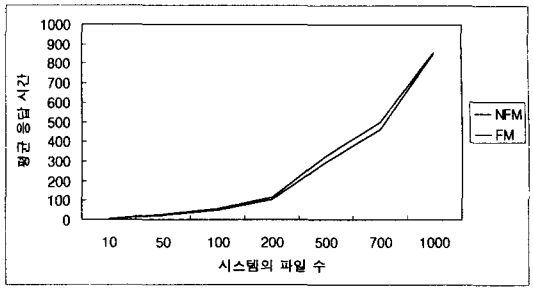
은 블록, 아이노드와 슈퍼블록 각각의 사용 빈도수를 합한 것과 같다. 시뮬레이션에서 얻고자 하는 성능 평가 기준은 파일 요청 후 처리되는 시간인 평균 응답 시간이다. 클러스터 파일 시스템 내의 파일 수는 고정적이라고 가정하였다. 즉, 새로운 파일이 생성되지 않는다고 가정한다.

그림 3은 클러스터 파일 시스템의 노드 수가 4이고 시스템에서 처리되고 있는 파일수가 10일 때 클러스터 파일 시스템의 부하량에 따른 평균 응답 시간을 나타낸다. 그림 3에서 파일 이동으로 인하여 얻을 수 있는 이득에 큐의 길이만을 고려한 방법에 대한 평균 응답 시간의 변화를 나타낸 것이 AQ이고 파일의 사용 빈도와 큐의 길이를 고려한 것이 OQ이다. 그림 3은 대체적으로 파일 자원의 사용 빈도와 큐의 길이를 고려한 방법이 평균 응답 시간이 적음을 보여준다. 파일 이동으로 인하여 얻을 수 있는 이득에 큐의 길이만을 고려한 방법에 대한 응답 시간의 평균(AQ)은 5.92998이었고 파일의 사용 빈도와 큐의 길이를 고려한 방법에 대한 응답 시간의 평균(OQ)은 5.79303였다.

그림 4는 시스템 내의 파일 수에 따른 평균 응답 시간의 변화이다. NFM은 파일 이동을 수행하지 않은 시스템이고 FM은 파일 이동을 수행하는 것이다. 그림 4에서 보는 것과 같이 시스템 내의 파일 수가 증가할수록 평균 응답 시간도 증가하게 된다. 시스템 내의 파일 수가 적을수록 하나의 파일에 대한 요청들의 간격이 없거나 작다. 따라서 하나의 파일 처리를 위한 평균 응답 시간이 적게 된다. 또한 그림 4는 파일 이동을 수행하는 시스템과 수행하지 않는 시스템의 평균 응답 시간을 나타내고 있다. 큐의 대기시간을 줄일 수 있는 노드로 파일 이동을 수행하는 것이 파일 이동을 위한 시간이 소비되더라도 평균 응답 시간에서 이득을 얻는다.



(그림 3) 시스템의 부하량에 따른 평균 응답 시간



(그림 4) 시스템의 파일 수에 따른 평균 응답 시간

5. 결론

본 논문은 클러스터 파일 시스템의 성능을 향상시키기 위한 방법으로 큐의 대기시간과 파일 자원의 접근률을 고려한 파일 이동에 관하여 연구하였다. 특히 파일 이동을 수행할 때 파일 자원의 접근률을 통해 두 노드에서 한 노드로 이동하여 시스템의 성능을 저하시킬 수 있는 문제점을 해결하였다.

향후 연구 과제로 파일 이동을 결정하기 위한 주기가 평균 응답 시간에 미치는 영향에 관한 연구가 필요하다. 또한 파일 이동 시 발생하는 서비스 중단을 막기 위해 수행되는 파일 중복에 따른 클러스터 파일 시스템의 성능 변화에 관한 연구와 가용성 분석에 관한 연구가 필요하다.

참고문헌

[1] High-Availability Linux Project, http://linux.ha.org

[2] R. Soltis, M. Ruwart and T. Okeefe, "The Global File System," NASA Goddard Space Flight Center Conference on Mass Storage System and Technologies, pp. 319-342, Sep. 1996.

[3] Li Sheng, "Models for Dynamic File Migration in Distributed Computer Systems, PhD Thesis, University of Rochester, 1986.

[4] R. Hurley and S. Yeap, "File Migration and File Replication: A Symbiotic Relationship," IEEE Transactions on Parallel and Distributed Systems, Vol. 7, No. 6, pp. 578-586, June 1996.

[5] 장승주, "고장 감내 전역 파일 시스템 설계," 정보과학회논문지, 제 24권, 제 10호, pp. 970-978, 1997. 10.