Lustre 파일 시스템을 위한 Purge 기능의 병렬화 구현

권민우*, 윤준원*, 홍태영*, 박찬열*
*한국과학기술정보연구원 슈퍼컴퓨팅센터
e-mail:mwkwon81@kisti.re.kr

A Parallel Implementation of Purge Process for Lustre File System

Min-Woo Kwon*, Jun-Weon Yoon*, Tae-Young Hong*, Chan-Yeol Park*
*Dept. of Supercomputing Center, KISTI

요 약

슈퍼컴퓨터는 대용량의 데이터를 효율적으로 관리하기 위해 Lustre 파일 시스템과 같은 고성능의 병렬 파일 시스템을 이용한다. 한국과학기술정보연구원의 슈퍼컴퓨터 4호기 Tachyon 2차 시스템과 같이 다수의 사용자가 접속하는 슈퍼컴퓨터는 사용자의 데이터가 한없이 누적됨으로 Lustre 파일 시스템의 성능이 저하되는 이슈가 있다. 본 논문에서는 사용자의 데이터가 누적되는 것을 방지하기 위해 장기간 사용하지 않는 데이터를 자동 삭제하는 기능인 Purge기능을 구현하였다. 특히, 기하급수적으로 늘어나는 병렬 파일 시스템의 용량에 대처하기 위해 병렬 컴퓨팅 기술을 이용해 고속 Purge 기능을 구현하였다. 단일 컴퓨팅 노드와 병렬 환경에서 구현한 결과를 비교하였을 때, 단일 컴퓨팅 노드에서는 1,517GB 용량을 지우는데 221.2초가 걸렸으며 16개의 컴퓨팅 노드를 이용한 병렬 환경에서는 49.9초가 걸렸다. 이 결과를 비교했을 때 단일 컴퓨팅 노드에서 구현한 결과 대비 병렬 환경에서 구현했을 때 약 4.4배의 성능향상을 얻을 수 있었다.

1. 서론

오늘날 슈퍼컴퓨터는 데이터의 용량이 기하급수적으로 늘어나면서 한 대의 컴퓨터로 이를 저장하거나 관리하기가 쉽지 않다. 대부분의 슈퍼컴퓨터는 분산파일시스템을 기본으로 하는 Lustre 파일 시스템과 같은 고성능의 병렬 파일 시스템을 이용하여 데이터를 관리한다[1]. 본 논문에서는 Lustre 파일 시스템에서 한없이 누적되는 사용자 데이터를 효율적으로 관리하기 위해 정해진 기간 동안 접근(Access)/수정(Modify)/변경(Change)이 발생하지 않은 파일 및 디렉터리를 자동 삭제하는 기능(Purge)을 병렬 컴퓨팅 기술을 이용하여 구현함으로 고속화하였다.

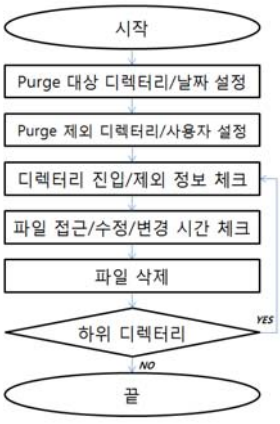
2. 슈퍼컴퓨터 4호기 Tachyon 2차 시스템의 구성

한국과학기술정보연구원(KISTI) 슈퍼컴퓨팅센터의 슈퍼컴퓨터 4호기인 Tachyon 2차 시스템은 3,200대의 인프라 노드와 컴퓨팅 노드로 구성되어 있으며 인피니밴드(Infiniband) 네트워크를 통해 병렬파일시스템과 연결되어 있다. 병렬파일시스템은 크게 3개의 디렉터리로 구성이 되어 있다. 117TB 용량의 홈 디렉터리(/home01)와 117TB 용량의 사용자 어플리케이션을 제공하는 디렉터리(/applic), 2.4PB의 글로벌 스크래치 디렉터리(/scratch2)를 제공하고 있다. 홈 디렉터리와 스크래치 디렉터리의 경우, 사용자 별로 용량 제한을 두고 있는데 홈 디렉터리는 64GB, 스크래치 디렉터리는 100TB이다. 스크래치 디렉터리의 경우 데이터가 한없이 누적되는 것을 막기 위해 15일간 사용하지 않은 데이터는 자동 삭제하는 Purge 기능을 수행하고 있다[2].

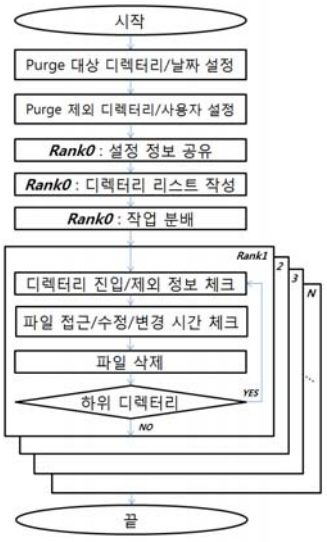
3. 단일 컴퓨팅 노드에서의 Purge 기능의 구현

Purge 프로그램은 설정 정보 파일로 부터 설정정보를 얻어서 동작한다. 설정정보로는 Purge 대상 디렉터리 정보(TargetDir)와 Purge 대상 파일의 기준 날짜 정보(days), Purge 제외 디렉터리 정보(xDir), Purge 제외 사용자 정보(xUsers)가 있다. Tachyon 2차 시스템에서는 '/scratch2' 디렉터리를 TargetDir로 Purge 기능을 수행한다. days는 이 날짜를 기준으로 그 이상 접근/수정/변경이 발생하지 않는 파일이나 디렉터리를 삭제하기 위한 기준 날짜이다. xDir은 Purge 기능에 의해서 삭제되는 것을 방지하기 위한 디렉터리의 목록으로서 시스템 관리를 위한 특수한 목적의 디렉터리와 같이 특별한 경우에 이 항목에 넣어 제외시킬 수 있다. xUser는 Tachyon 2차 사용자 중에 특정 사용자의 계정이 소유하는 디렉터리를 Purge 기능에서 제외시키는 것으로 시스템 관리자 계정과 같은 특별한 사용자가 소유하는 파일과 디렉터리에 대해서 삭제 항목에서 배제시킬 때 사용할 수 있다.

(그림 1)은 단일 컴퓨팅 노드에서 구현된 Purge 기능의 순서도이다. Purge 기능이 동작하면 설정 정보 파일에서 위 정보들을 읽어들인다. 그 후에 TargetDir에 진입한다. xDir, xUser 정보를 이용하여 디렉터리가 제외대상인지를 판별한다. 제외 대상이 아니라면, 디렉터리 내의 모든 파일에 대하여 접근/수정/변경 시간을 조사하여 days 값과 비교하여 모두 이보다 작을 경우 파일 삭제를 진행한다.



(그림 1) 단일 노드 Purge



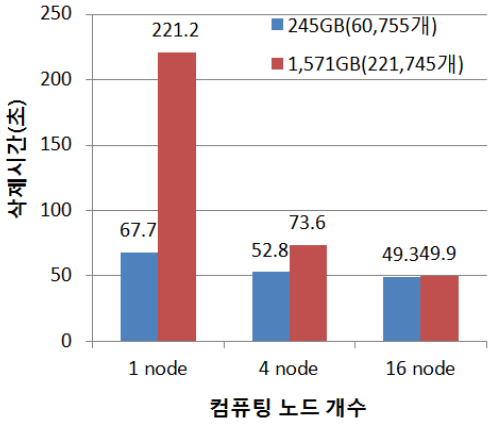
(그림 2) 병렬 환경 Purge

4. 병렬 컴퓨팅을 이용한 고속 Purge 기능의 구현

본 논문에서는 MPI 라이브러리를 이용하여 Purge 기능을 병렬화 구현하였다. (그림 2)는 병렬환경의 컴퓨팅 노드에서 구현된 Purge 기능의 순서도이다. 앞에서 설명한 단일 컴퓨팅 노드에서의 기능 구현과 동일하게 Purge 기능에 관련된 설정 정보들을 입력받는다. 이렇게 입력받은 설정 정보를 루트 노드(*Rank0*)가 나머지 노드(*Rank1*~*RankN*)로 MPI 메시지를 이용해 전달한다[3]. 이후에 *Rank0* 노드는 TargeDir의 하위 디렉터리 경로를 나머지 노드들에게 전달함으로 작업 분배를 한다. 디렉터리 경로를 받은 노드들은 디렉터리에 진입하여 설정정보를 토대로 삭제 여부를 확인한다. 삭제 대상으로 판별될 경우, 접근/수정/변경 시간을 조사하여 days 값과 비교하여 모두 이보다 작을 경우 파일 삭제를 진행한다.

5. 실험 결과 분석 및 결론

(그림 3)은 단일노드(1노드)와 병렬환경(4, 16노드)에서의 실험결과를 보여준다. 동일한 환경에서 실험하기 위해서 두 개의 TargetDir에서 days를 '0'으로 하여 전체 파일을 삭제하는 시간을 측정하였다. 두 디렉토리의 용량과 파일개수는 각각, 245GB(60,755개), 1,571GB(221,745개)이다. 단일노드에서 245GB, 1,571GB용량을 삭제하는데 각각, 67.7초, 221.2초의 시간이 걸렸다. 4개의 노드에서 삭제를 진행했을 때는 각각, 52.8초, 49.3초의 시간이 걸리고, 16개의 노드에는 각각, 73.6초, 49.9초의 시간이 걸렸다. 큰 용량의 데이터에서 테스트를 한 경우가 단일노드 대비 병렬환경을 이용한 구현 결과에서 큰 차이를 보이는 것을 확인할 수 있다. 이는 본 논문에서 구현한 병렬 컴퓨팅 방식이 전처리부를 *Rank0*에서 담당하고 계산부를 병렬화하다 보니 용량이 작은 실험에서는 단일노드에서의 구현과 큰 차이가 발생하지 않은 것이다. 데이터 용량이 적은 실험에서는 4노드와 16노드 간에 결과의 차이도 크게 발생하지 않았다. 그러나 데이터 용량이 큰 실험에서는 결과에서 확인할 수 있는 것처럼 4노드로 동작 시에 단일노드 대비 약 3배, 16노드로 동작 시에 약 4.3배의 결과를 얻었다. 일반적인 슈퍼컴퓨터 환경에서는 본 논문에서 제시하는 수치보다 훨씬 큰 데이터 용량이 제거됨으로 단일노드 구현 대비 병렬환경 구현에서 좋은 성능을 얻을 수 있다. 향후, 병렬화 정도를 향상시켜 성능을 개선하는 연구와 파일크기와 개수를 고려한 최적의 병렬 작업 분배에 관한 연구를 진행하고자 한다.



(그림 3) 단일노드와 병렬환경에서 구현된 Purge 삭제 시간 결과

참고문헌

[1] Feiyi Wang, Sarp Oral, Galen Shipman, Oleg Drokin, Tom Wang and Isaac Huang "Understanding lustre filesystem internals", Oak Ridge National Lab, Technical Report ORNL/TM-2009/117, 2009

[2] National Institute of Supercomputing and Networking, KISTI, http://www.nisn.re.kr

[3] William Gropp, Ewing Lusk, Nathan Doss and Anthony Skjellum "A high-performance, portable implementation of the MPI message passing interface standard" Parallel computing, 22(6), 789-828, 1996