병렬 파일 시스템에서의 가용 입출력 대역폭을 고려한 테이블 비교 선반입 정책

김재열 석성우 조종현 서대화
경 북 대 학 교 전 자 공 학 과
{vada, swsok, alfcom}@palgong.knu.ac.kr dwseo@ee.knu.ac.kr

Table Comparison Prefetching using Available I/O Bandwidth in Parallel File System

Chei-Yol Kim Song-Woo Sok Jong-Hyun Cho Dae-Wha Seo
Dept. of Electronic Engineering, Kyungpook National University

요 약

과도한 파일 입출력이 요구되는 병렬파일 시스템의 성능을 결정하는 중요한 요소로서 캐쉬과 선반입을 들 수 있다. 본 논문은 캐쉬의 크기에 비해 상대적으로 큰 파일을 요청하는 경우에 시스템 성능에 막대한 영향을 미치는 선반입에 대해서 선반입할 데이터를 결정하는 알고리즘으로 테이블 비교법을 제안하고, 이와 더불어 예측된 데이터의 선반입 여부와 선반입 시기를 결정하는 경우 현재의 가용 입출력 대역폭을 고려하는 기법을 제안한다. 제안하는 선반입 알고리즘은 시뮬레이션을 통하여 기타 선반입 알고리즘과 비교해 본 결과 파일 시스템 성능이 향상되었음을 보여준다.

1. 서론

급속한 프로세서의 처리속도 향상으로 인하여 입출력 장치는 컴퓨팅 환경의 병목이 되었다. 이러한 병목을 해소하는 하나의 방법으로 입출력 장치에 병렬성을 도입하였으며 이 결과로 현재 많은 클러스터링 컴퓨팅 환경에서 병렬 파일 시스템이 사용되고 있다. 병렬 파일 시스템의 도입으로 파일의 입출력 대역폭은 향상이 되었으나 여전히 파일 요청시의 디스크 접근 지연시간을 줄일 수는 없었다. 이러한 문제의 해결책으로 이전에 사용한 데이터 블록 중 다음에 사용될 확률이 높다고 생각되는 것들을 메모리에 남겨두는 캐쉬과 다음에 사용될 것으로 예측되는 블록을 미리 메모리로 읽어 들여 파일 서비스 요청 시 지연시간을 최소화 하는 선반입 기법이 있다. 이 중 과학 계산용 병렬 응용프로그램에 사용되는 파일의 경우에는, 대부분의 파일 크기가 캐쉬 메모리 보다 크며 요구하는 데이터의 양도 일반적인 파일 시스템에 비해 큰 경우가 대부분이다[4][5]. 이러한 경우 캐쉬 보다는 선반입 기법이 파일 시스템 성능에 훨씬 많은 영향을 끼치게 된다.

대부분의 선반입 기법에 대한 이전의 연구는 다음에 선반입할 대상 블록을 찾는 알고리즘이 대부분이었다. 하지만 실제 파일 시스템 성능면에서 본다면 파일 요청이 많은 경우에는 선반입을 하지 않는 것이 성능에 유리한 경우도 발생한다. 이를 고려해 본 논문에서는 선반입할 데이터를 결정하는 알고리즘으로 두 개의 테이블을 비교해서 패턴을 찾아내는 테이블 비교 기법을 소개하고, 결정된 데이터의 선반입 여부와 선반입 시기를 결정하는 데 있어 응용프로그램들의 각 파일에 대한 단위시간당 데이터 요구량을 산출하여, 현재 사용 가능한 입출력 대역폭을 계산하고, 이를 기준으로 선반입 시기를 결정하는 가용 입출력 대역폭을 고려한 선반입 정책을 소개한다.

2. 관련연구

2.1. 병렬 응용프로그램의 파일 접근 패턴

최근에 와서야 진행된 클러스터링 환경에서의 병렬 응용프로그램의 파일 접근 패턴 연구[4][5]에 따르면 대부분의 접근 패턴은 크게 3 가지로 나누어 진다. 첫째, 순차적 접근(sequential)으로 하나의 계산 노드가 자신의 개인 파일을 순차적으로 접근하는 경우이며, 둘째로는 하나의 파일을 여러 노드가 동시에 접근하여 자신이 맡은 해당 부분을 읽어가는 경우(interleaved)이며, 마지막으로 위의 두 가지 경우를 혼합한 경우(mixed)로 파일의 헤더 부분은 하나의 노드가 순차적으로 접근하고 나머지 부분은 두 번째 접근 패턴으로 여러 노드가 동시에 접근하는 형태이다.

2.2. 병렬파일 시스템에서의 선반입 기법

이상적인 경우의 선반입 알고리즘은 응용프로그램이 다음에 요구할 데이터를 정확히 알고 선반입하는 경우이다. 이러한 접근 방향의 연구로서 응용프로그램이 제공하는 힌트를 사용해서 선반입하는 방식이 있다[1]. 그러나 힌트를 사용하는 방식은 응용 프로그래머의 부가적인 작업이 필요하고, 기존 응용프로그램에 대한 호환성이 없다는 이유로 널리 사용되지 않고 있다. 이와는 달리 응용프로그램이 제공하는 파일 접근 패턴에 대한 정보 없이 현재의 파일 접근 기록이나 과거의 파일 접근 기록을 가지고 다음에 사용될 데이터를 예측하는 방식의 연구가 있다. 이러한 연구로는 Griffioen과 Appleton[2] 이 제안한 확률 그래프를 사용하는 방법이나 Cortes 가 제안한 ISG(Interval -and-Size-Graph) 방법[3]등이 있다. 확률 그래프는 파일 크기가 대부분 매우 작은 일반 파일 시스템에서 선반입 단위를 파일로 보고 만든 것으로써 병렬파일 시스템에 적용하기는 적절하지 않다. ISG방식은 기타 다른 방식에 비해 유지 해야 하는 자료의

양을 줄였음에도 불구하고, 여전히 패턴이 일정하지 않은 경우에는 그래프를 유지하는 비용이 과도하게 커질 수 있어 시스템의 과부하로 작용할 여지가 있다.

위와 같은 이유로 기존의 알고리즘들은 2.1 에서 보여준 병렬 응용프로그램의 파일 접근 패턴에 대해서는 과도한 메모리와 CPU 자원을 사용하여 효율성이 떨어진다.

3. 테이블 비교 선반입 기법

테이블 비교 선반입 기법은 응용프로그램으로부터 파일요청이 들어오면 각 파일에 대하여 그림1에 보이는 테이블 A,B 두 개를 각각 생성한다. 테이블 A,B 는 요청이 들어 올 때마다 번갈아 갱신된다. request_size 는 요구한 블록의 개수를 나타내며, requets_interval 은 이전 요청의 마지막 블록과 현 요청의 첫번째 블록간의 간격을 기록한다. start_block_num은 요청한 블록들의 첫번째 블록 번호를 나타내며 이는 다음 요청 때의 request_interval 을 계산할 때 사용된다. 즉, request_interval_A(B) = start_block_num_A(B) - (start_block_num_B(A) + request_size_B(A) □ 1) 이다.

이렇게 기록된 두개의 테이블을 비교하여 request_size와 request_interval이 일치하는 경우에 일정한 패턴이라고 판단하고 다음에 선반입할 데이터 블록을 결정한다.

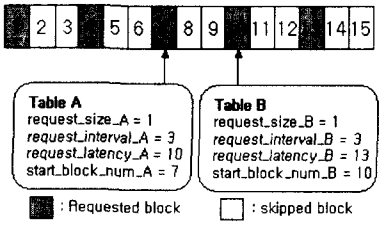


그림 1. simple strided 패턴에 대한 테이블 비교 선반입의 예

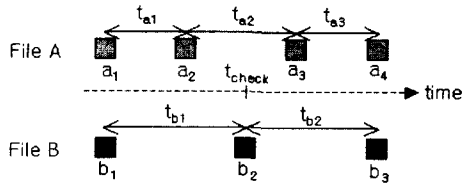
그림1은 파일 요청이 1,4,7,· · ·의 순으로 블록을 요청하는 simple strided 패턴에 대한 테이블 비교법의 예를 들고 있다. 그림1에서의 판단 결과는 Table A 와 Table B 의 request_size와 request_interval 이 동일 하므로 13 번을 10 번 다음에 요청될 블록으로 결정한다. 특별한 경우로 request_interval이 1 인 경우는 순차적 요청을 나타낸다.

기존의 선반입 알고리즘[2][3]은 블록 예측을 위하여 매우 크고 복잡한 자료구조를 사용하고 있다. 그러나 병렬 응용프로그램의 파일 접근 패턴[4][5]을 분석해 보았을 때 2.1절에서 언급한 3 가지 형태의 파일 접근이 주류를 이루고 있으며, 이외의 패턴은 전체 요구의 1% 이하이다. 이러한 이유로 제시한 테이블 비교법은 간단한 자료구조만으로도 대부분의 병렬 응용프로그램의 파일 접근 패턴에 대해서 충분한 성능을 발휘한다.

4. 가용 대역폭을 고려한 선반입 여부 및 시기 결정

병렬 응용프로그램에서 사용하는 파일은 대부분이 크기가 커서 응용프로그램은 한 번에 하나의 파일 전체를 요구하지는 않는다. 응용프로그램은 메모리에 일정한 크기의 버퍼를 할당 받고 이를 통하여 파일 데이터를 반복적으로 요청하여 연산작업을 수행한다. 파일 전체에 대해서 이렇듯 동일한 크기의 데이터를 요청하지는 않으나 부분적으로는 이러한 경향을 따른다. 본 논문에서는 이러한 응용프로그램의 파일 접근 경향을 이용해 대강의 사용 가능한 입출력 대역폭을 계산하여 선반입 여부와 시기를 결정하는데 사용한다.

그림2는 응용프로그램이 두개의 파일 A, B 를 요청하는 경우를 나타낸다. 시스템의 가용 대역폭을 계산하기 위해서는



- a_n, b_n : request size
- t_{an}, t_{bn} : request latency (n=1,2,3, …)

그림 2. 입출력 대역폭 계산을 위한 파일 요청의 예

먼저 현재 응용프로그램이 사용하고 있는 입출력 대역폭을 계산하는 과정이 필요하다. 그림2의 요구 지연시간(request latency)은 그림1의 테이블에 정의된 request_latency 항목과 같은 것으로 각 요청이 들어올 때마다 테이블에 기록된다. 그림2의 블록 데이터 b_2가 요청되었을 때의 시간 t_{check}에서의 가용 입출력 대역폭을 계산해 보자. 먼저 응용프로그램들이 현재 사용하고 있는 입출력 대역폭은 가장 최근의 지연시간과 요구 데이터 크기를 이용해서 구해진다.

따라서, $BW_{used} = a_2/t_{a1} + b_2/t_{b1}$ 가 되며

파일 시스템의 최대 대역폭을 BW_{max} 라고 하면 가용 대역폭 BW_{avail}는 아래와 같다.

$BW_{avail} = BW_{max} - BW_{used}$

이와 같이 계산된 가용 대역폭 BW_{avail}은 아래와 같이 선반입 여부와 시기를 결정하게 된다.

① if ($BW_{avail} < 0$)
  No prefetching
② else if ($BW_{avail} < 1/2\ BW_{max}$)
  if (request_interval == 1)
    Prefetching with current request
  else
    Push prefetching request into prefetching queue
③ else if ($BW_{avail} > 1/2\ BW_{max}$)
  Push prefetching request into prefetching queue

①의 경우는 현재 응용프로그램이 요구하는 입출력 대역폭이 시스템이 제공할 수 있는 능력을 넘어서는 것으로 판단하고 선반입을 중지하게 된다.

②의 경우 request_interval = 1 이면 파일 요청이 순차적이라는 의미며, 이때는 연속적으로 현재 요청한 블록과 선반입할 블록을 동시에 읽어 들인다. 충분한 입출력 대역폭이 남아 있지 않기 때문에 가능하면 파일 시스템의 성능을 효과적으로 이용하기 위하여 연속된 블록은 한 번에 읽어온다.

③의 경우는 입출력 대역폭이 충분하기 때문에 모든 선반입을 입출력 요청이 없을 때 하도록 한다.

이처럼 계산된 가용 대역폭은 시스템에 많은 부하를 주지 않으면서도 필요한 정도의 정확성을 가질 수 있다는 것이 입출력 가용 대역폭을 고려한 선반입 기법의 특징이다.

5. 시뮬레이션

5.1. 시뮬레이션 환경

제안한 선반입 알고리즘의 성능을 측정하기 위한 시뮬레이션 프로그램은 리눅스 운영체제 환경에서 C 언어를 사용하여 작성되었다.

[표 1]. 시뮬레이션 변수

Local disk		Cache memory	
Thruput	4MB/s	Thruput	400MB/s
Latency	8ms	Latency	1μs

표1은 실험에 사용된 매개 변수를 나타낸다. 이때 사용된 입출력 노드의 개수는 4 대로 가정하였고 네트웍의 대역폭은 충분히 크다고 보았다. 또한 하나의 블록의 크기는 8KB 이다. 각 파일 블록은 4 대의 노드에 스트라이핑 방식으로 저장된다. 따라서 BW_{max}는 아래와 같이 계산될 수 있다.

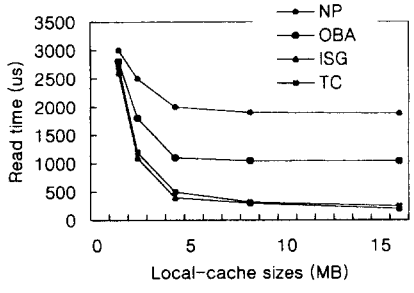
BW_{max} = Thruput(disk)×node개수 = 4MB/s × 4 = 16MB/s

실험에 사용된 작업부하(workload)는 병렬 응용프로그램의 주요 파일 접근 패턴인 sequential, interleaved, mixed 를 바탕으로 발생시켰다. 작업부하는 workload A , B, 2 가지를 발생시켰다. 두개의 작업부하는 접근 패턴 자체는 동일하나 데이터의 요구량, 즉 요구 대역폭은 서로 다르다. workload A 의 평균 데이터 요구량은 8MB/s 이고 workload B의 경우는 12MB/s 이다.

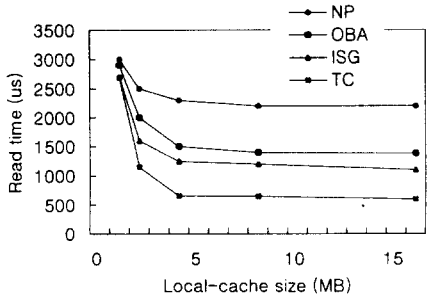
5.2 시뮬레이션 결과

실험에는 제안한 선반입 기법 외에 총 4 개의 선반입 기법을 대상으로 하였다.

- NP : No prefetching
- OBA : One-Block Ahead
- ISG : Interval-and-Size-Graph
- TC : Table Comparision



(a) workload A - 평균 데이터 요구량이 8MB/s 인 경우



(b) workload B □ 평균 데이터 요구량이 12MB/s 인 경우

그림 3. 두개의 작업부하에 대한 선반입 기법의 평균 읽기 시간

그림3은 선반입 기법들에 따르는 각 요청에 대한 평균 읽기 시간을 나타내고 있다. 시스템의 성능을 측정하는 방법으로는 캐쉬의 히트율을 사용 할 수도 있으나 보다 실제적인 시스템의 성능을 검증하기 위해서는 총 입출력 시간을 보는 것이 타당하므로 본 논문에서는 시뮬레이션 결과로 평균 읽기에 사용된 시간을 제시한다. Workload A 와 B 는 서로 패턴은 같으나 평균 데이터 요구량이 서로 다르다. (a) 의 경우에는 입출력 대역폭에 비해서 요구하는 대역폭이 절반밖에 되지 않으므로 충분한 여유를 가지고 있는 경우이다. 이런 경우는 현재의 가용 대역폭을 고려하지 않는 기존의 기법들 중 ISG 와 테이블 비교법이 비슷한 성능을 보인다. 하지만 (b) 의 경우처럼 파일 요청이 많은 경우에는 가용 대역폭을 고려한 테이블 비교법이 적은 시스템 자원을 사용하면서도 ISG 기법과 비교해 더욱 우수한 성능을 나타내고 있다. 이는 파일 요청이 빈번하지 않은 환경에서는 가용 대역폭을 고려하지 않아도 성능에 큰 영향을 미치지 않지만, 파일 요청이 빈번한 병렬파일 시스템에서는 가용 입출력 대역폭을 고려하는 것이 타당하다는 것을 보여주고 있다.

6. 결론 및 향후 과제

데이터 선반입 기법은 앞으로 사용될 데이터를 응용프로그램이 요구하기 전에 미리 캐쉬 메모리에 옮겨 파일 시스템의 성능을 향상시키는 방법이다. 기존의 선반입 연구들 [2][3]은 대부분이 어떤 블록을 미리 가져 올 것인가에 대해서만 초점을 맞추었다. 그러나 본 논문에서는 데이터를 예측하는 방법으로, 병렬 응용프로그램의 파일 접근 패턴 [4][5]에 대해서 간단하면서도 효과적인 테이블 비교법을 제안하였고, 이와 더불어 선반입 여부와 선반입 시기를 결정함에 있어 파일 시스템의 가용 대역폭을 고려하였다.

제안한 테이블 비교법과 가용 대역폭을 고려한 선반입 기법을 사용함으로써 파일 시스템의 성능을 향상 시킬 수 있음을 시뮬레이션 실험을 통해서 보여주었다. 이러한 결과는 파일 시스템의 상황을 고려한 효율적인 정책을 사용 함으로서 시스템의 성능을 향상시킬 수 있음을 보여준다.

향후 과제로는 현재의 입출력 상황을 보다 정확히 반영하여 파일 시스템의 성능을 높일 수 있는 방향의 연구가 필요하며, 제안한 선반입 기법을 실제 파일 시스템에 구현하고, 이를 정확히 분석함으로써 실제 구현상의 개선점을 찾아낼 필요가 있다.

7. 참고 문헌

[1]. Prasenjit Sarkar and John Hartman, "Efficient Cooperative Caching using Hints, " Proc. of the USENIX 1996, 2nd Symposium on Systems Design and Implementation, Seattle, Washington, Oct 28-31, 1996.

[2]. J. Griffioen and R. Appleton, "Performance Measurements of Automatic Prefetching," Parallel and Distributed Computing Systems, pages 165--170. IEEE, Sept. 1995.

[3]. Toni Cortes, "Cooperative Caching and Prefetching in Parallel/Distributed File Systems, " PhD thesis, UPC: Universitat Politècnica de Catalunya, Barcelona, Spain, 1997.

[4]. Evgenia smirni and Daniel A. Reed, "Workload Characterization of Input/Output Intensive Parallel Applications, " Proc. of the Conference on Modeling Techniques and Tools for Computer Performance Evaluation, Springer-Verlag Lecture Notes in Computer Science, pp. 169-180, Vol. 1245, June 1997.

[5]. David Kotz and Nils Nieuwejaar, "File-Access Characteristics of Parallel Scientific Workloads, " *in* IEEE Parallel and Distributed Technology, Spring 1995, pages 51-60. *in* IEEE Parallel and Distributed Technology, Spring 1995, pages 51-60.