데이터 그리드 환경에서 파일 교체 정책 연구

박 홍 진†

요 약

데이터 그리드는 대용량의 데이터 어플리케이션 처리를 위해 지리적으로 분산되어 있는 저장 자원을 제공한다. 대용량을 처리해야 하는 데이터 그리드 환경에서는 기존 웹 캐싱 정책이나 가상 메모리 캐쉬 교체 정책과는 다른 파일 교체 정책이 필요하다. LRU(Least Recently Used)나 LCB-K(Least Cost Beneficial based on K), EBR(Economic-based cache replacement), LVCT(Least Value-based on Caching Time) 같은 기존의 파일 교체 전략은 파일 교체를 위해 추가적인 자원이 필요하거나 미래를 예측해야한다. 본 논문은 이를 해결하기 위해 파일의 크기에 기반하여 파일 교체를 수행하는 SBR-k(Sized-based replacement-k)을 제안한다. 성능평가 결과 제안한 정책이 기존의 정책보다 더 나은 성능을 나타낸다는 것을 확인하였다.

키워드 : 데이터 그리드, 자원 저장 관리자, 파일교체 정책

A Study of File Replacement Policy in Data Grid Environments

Hong-Jin Park†

ABSTRACT

The data grid computing provides geographically distributed storage resources to solve computational problems with large-scale data. Unlike cache replacement policies in virtual memory or web-caching replacement, an optimal file replacement policy for data grids is the one of the important problems by the fact that file size is very large. The traditional file replacement policies such as LRU(Least Recently Used), LCB-K(Least Cost Beneficial based on K), EBR(Economic-based cache replacement), LVCT(Least Value-based on Caching Time) have the problem that they have to predict requests or need additional resources to file replacement.

To solve theses problems, this paper propose SBR-k(Sized-based replacement-k) that replaces files based on file size. The results of the simulation show that the proposed policy performs better than traditional policies.

Key Words : Data Grid, Resource Storage Manager, File Replacement Policy

1. 서 론

고가의 슈퍼 컴퓨팅 사용에 대해 저비용의 고효율적인 대안으로 그리드 컴퓨팅이 제안되어 활용되고 있다. 그리드 컴퓨팅은 분산되어 있는 고성능 컴퓨팅 자원을 네트워크로 상호 연동하여 조직과 지역에 관계없이 사용할 수 있는 환경을 의미한다. 일반적으로 그리드 컴퓨팅은 컴퓨팅 그리드(computational grid), 데이터 그리드(data grid), 엑세스 그리드(access grid)로 구분된다. 컴퓨터 그리드는 지역적으로 분산되어 있는 컴퓨팅 파워를 공유하여 마치 한대의 고성능 컴퓨터처럼 사용할 수 있는 해주는 그리드이며, 엑세스 그리드는 분산된 지역의 연구원들이 공동으로 프로젝트를 진행할 수 있는 협업 환경을 제공해 주는 그리드이다. 그리드 컴퓨팅에서 데이터 그리드는 고성능의 이질적인 노드와 데이터 저장 자원이 지리적으로 분산되어 있는 플랫폼의 네트워크를 의미한다. 데이터 그리드 컴퓨팅을 이용한 응용 분야는 지질학 연구, 고 에너지 물리학, 항공물리, 기후변화 모델링등 분산되어 있는 대용량의 데이터를 생성, 저장, 처리하는 것과 관련되어 있는 분야이며[1-4], 본 논문은 데이터 그리드의 요소기술에 대한 논문이다.

데이터 그리드에서 중요한 문제 중 하나는 상호 연결된 네트워크의 높은 지연 시간과 저장 장치에 있는 대용량 데이터에 대한 접근과 관련된 문제이다. 이 문제를 해결하기 위한 방법으로는, 원격 접근에 따른 오버헤드를 피하고 자원에 대한 신뢰성을 향상하기 위해 원격 데이터를 근거리(local)로 캐싱하는 기법을 들 수 있다. 캐싱 기술은 컴퓨터 시스템, 데이터베이스, 웹 캐싱등 저장 시스템의 성능을 향상시키기 위해 사용되어 왔다. 그러나, 데이터 그리드에서 사용하는 캐싱 기술은 기존 캐싱 기술과 여러 가지 면에서

† 정 회 원 : 상지대학교 컴퓨터정보공학부 조교수
논문접수 : 2006년 5월 26일, 심사완료 : 2006년 9월 22일

서로 다르다. 예를 들어, 데이터 그리드에서 사용되는 네트워크는 노드 거리가 먼 원거리 네트워크(WAN)에 기반하고 있으며, 파일 크기도 기가바이트 용량이며, 캐쉬 크기도 수백 기가 바이트에서 수십 테라바이트를 수용할 수 있어야한다. 기존 웹에서 사용하는 캐싱 기술은 전송 지연 시간도 몇초에서 몇분 정도에 불과하고, 캐싱 기술이 선택적인 사항임에 반해, 데이터 그리드 환경에서는 데이터 전송 지연 시간이 몇분에서 몇 시간까지 매우 길며, 웹 기술 사용도 필수적이다. <표 1>은 웹 캐싱과 데이터 그리드에서 캐싱과의 보다 세부적인 차이점을 보이고 있다.

데이터 그리드 환경에서는 대용량 데이타 저장 장치에 대한 요청 횟수가 동시에 수십에서 수백, 수천이 될 수 있다. 각각의 요청은 버퍼에 저장되며 수많은 파일 처리 요청 중 먼저 처리해야 파일을 선택해야하며, 이러한 선택을 파일 허가 정책이라 한다. 선택된 파일의 저장 공간 확보를 위해 현재 저장 되어 있는 파일 중 교체될 희생자 파일을 선택해야 하며, 이러한 희생자 선택을 파일 교체 정책이라 한다. 데이터 그리드 환경에서 파일 교체 전략 기법은 <표 1>에서 나타난 것처럼 처리되어야 할 파일의 크기등이 다르기 기존의 가상 메모리 페이징 교체 정책이나 웹 캐싱 정책과는 다른 파일 교체 정책을 사용해야한다[5,6]. 즉, 기존 가상 메모리 페이징 교체 정책에서 사용되는 LRU나 LFU 같은 알고리즘은 너무 단순한 정보를 가지고 교체될 파일을 선택한다. 데이터 그리드를 위한 파일 교체 정책인 LCB-K이나 [5], EBR 정책[7]은 파일 교체를 위해 미래를 예측해야 하는 문제점이 있다. 또한 LVCT 정책[8]은 파일 교체를 위해 반드시 스택을 유지/관리해야 하는 문제점이 존재한다.

본 논문에서 제안한 파일 교체 정책은 스택과 같은 추가적인 자원이 필요 없고, 미래를 예측하지 않고 파일을 교체하는 SBR-k(Size-based replacement-k)를 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 데이터 그리드에서의 시스템 모델을 설명한다. 3장에서는 기존의 파일 교체 정책과 이들의 문제점을 설명한다. 4장에서 본 논문에서 제안한 정책을 설명한다. 5장에서는 기존 정책과의 성능을 비교/평가한 결과를 제시하고, 6장에서 결론을 맺는다.

〈표 1〉 웹 캐싱과 데이터 그리드 캐싱 비교[5]

특성	웹 캐싱	데이터 그리드 캐싱
파일/객체 크기	메가바이트(MB)	기가바이트(GB)
캐쉬 크기	수십에서 수백 메가 바이트	수백 기가에서 수십 테라바이트
전송 시간	몇초에서 몇분	몇초에서 몇시간
캐쉬 요구	선택적인 사항	강제적인 사항
객체 참조 시간	거의 순간적	몇초에서 몇분
배치 요구	전형적으로는 하나의 요구가 하나의 배치 참조	전형적으로 하나의 요구는 수백 파일과 연관
네트워크 대역폭	표준 인터넷 용량	초고속 기가 네트워크 용량

2. 데이터 그리드의 시스템 모델

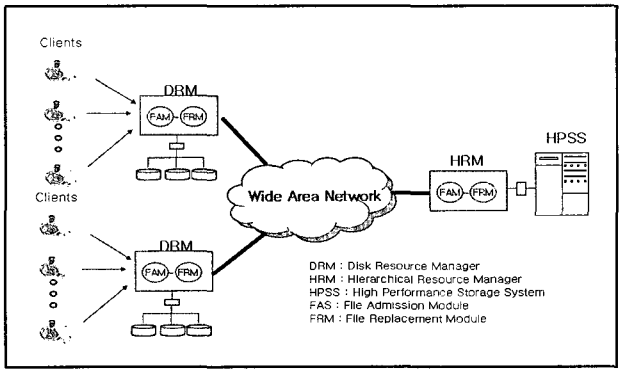
데이터 그리드 환경에서 저장 자원 관리자(SRM : Storage Resource Manager)는 데이터 공유와 자원 저장 관리의 기능을 제공하는 필수적인 미들웨어 컴포넌트이다. 저장 자원 관리자의 주요 기능은 대용량의 정보에 대한 캐싱이다. 저장 자원 관리자는 디스크 자원 관리자(DSM : Disk Resource Manager)나 계층적 자원 관리자(HRM : Hierarchical Resource Manager)로 특성화 될 수 있다[6].

그림 1은 데이터 그리드 환경에서 저장 자원 관리자의 위치를 보이고 있다. 각 사이트에는 하나 이상의 저장 자원 관리자가 있을 수가 있으며, 이들은 서로 원거리 네트워크로 연결되어 있다. 또한, 각 사이트 내에서 자원들은 근거리 네트워크로 구성되어 있다. 각각의 사이트에는 다양한 크기를 지닌 파일들 존재하고 있으며, 임의의 파일은 동시에 데이터 그리드에서 여러 사이트에서 복사되어 존재할 수도 있다.

디스크 자원 관리자는 파일에 대한 메타 데이터 정보 테이블(예를 들어 파일이름, 크기, 위치 정보등)를 유지하고 있으며, 사용자로부터 자원의 대한 요청을 받아 해당의 자원을 제공하는 역할을 수행한다. 먼저, 사용자가 원하는 자원이 근거리에 있는지 메타 데이터 정보 테이블을 통해 확인한 후 근거리에 있으면 자원을 바로 제공해 주며, 근거리에 없으면 원격지로 이를 요청하여 서비스 해준다.

각 사이트에 있는 자원 저장 관리자는 만약 근거리에 새로운 파일이 생성되었다면, 이에 대한 파일의 메타 정보를 (파일이름, 크기등) 다른 모든 사이트에 전송시켜서 후에 파일을 원하는 사이트에서 쉽게 사용할 수 있도록 한다. 또한, 다른 사이트에서 원본 파일을 복사하였으면 이에 대한 정보(메타 정보)도 다른 사이트에 전송 시키며, 메타 정보를 받은 사이트에는 식 (1)을 통해 파일 전송 비용(cost)을 산출할 수 있다. (식 1)에서 v1, v2는 사이트를 나타내며 파일 f에 대한 사이트 v1에서 v2까지의 전송 비용을 나타내고 있다.

$$cost(f, v1, v2) = \text{지연시간} + \text{파일의 크기}(fi) / \text{대역폭}(v1, v2) \quad \text{식 (1)}$$



(그림 1) 데이터 그리드 환경에서 저장 자원 관리자

위 식 (1)을 기반으로 각 사이트의 자원 저장 관리자는 자신의 파일 메타 정보 테이블에서 해당 파일 정보를 변경한다.

계층적 자원 관리자는 고성능 저장 장치(HPSS : High Performance Storage System)에 대한 관리에 대한 역할을 수행한다. 웹 프락시 캐쉬 서버와는 다르게 저장 자원 관리자에 도착된 각각의 요청은 동시에 수백에서 수천 건이 될 수 있다. 데이터 그리드에서는 많은 사용자의 동시 접속 자원 요청을 지원하고 있으며, 이를 위해 저장 자원 관리자는 요청 자원을 요청 버퍼에 큐잉(queuing)시킨다. 큐잉된 자원에 대해 어느 파일이 먼저 처리되어야 할지를 결정해야 하는데(예를 들어, 선입 선출), 이를 "파일 허가 정책(file admission policy)"이라고 한다. 파일 허가 정책은 그림 1에서 자장 자원 관리자 내에 파일 허가 모듈(FAM : File Admission Module)이 수행한다. 허가된 파일 요청을 처리하기 위해 먼저 근거리에서 찾아보고 없을 경우에는 원거리에서 요청 파일을 가져와야한다. 이때 지역 디스크 공간이 충분하지 않으면 요청 파일을 위한 공간을 확보하기 위해 저장되어 있는 파일 중 희생자(victim)를 선정해서 교체해야 하는데 이를 "파일 교체 정책(file replacement policy)이라한다. 저장 자원 관리자 내에 파일 교체 모듈(FRM : File Replacement Module)이 이 같은 일을 수행한다. 본 논문에서는 데이터 그리드 환경의 저장 자원 관리자 내에서 바로 이 파일 교체 정책에 관한 것이다.

3. 관련 연구

운영체제에서 많이 사용되는 페이지 교체 정책에는 LRU(Least Recently Used)와 LFU(Least Frequently Used)가 있으며[9], 이들 외에도 LRU와 LFU를 서로 혼용한 과거 k 참조에 기반하여 LRU을 적용하는 LRU-k[10]가 있다. 이들 정책은 최근 참조된 시점이나 참조되어진 빈도수를 가지고 캐쉬될 파일을 선정하는 알고리즘이다.

LCB-K(Least Cost Beneficial based on K backward reference) 정책은 [5]에서 제안한 데이터 그리드의 파일 교체 정책이다. 이 알고리즘은 교체될 파일 선택을 위해 유틸리티 함수를 사용한다. LCB-K정책에 기반한 유틸리티 함수는 파일들이 향후 참조될 가능성을 예측하여 계산한다. 즉, 미래 도착률을 예측하기 위해 (최대 K 까지의 최근 참조회수*검색 값(cost)/파일의 크기)의 유틸리티 함수를 사용한다. 캐쉬되어야 할 각 파일들의 유틸리티 함수 값으로 상대적인 순위를 매겨 가장 낮은 유틸리티 함수 값을 가진 파일을 교체대상으로 선택한다. 요청 파일을 위한 충분한 공간이 확보 될 때까지 이과정이 반복된다.

EBR(Economic-Based cache Replacement) 정책은 [7]에서 제안하였다. 이 정책은 데이타 그리드 환경에서 파일의 중복을 최적화하기 위한 알고리즘이다. 최적의 중복 결정을 하기 위해 중복된 파일에 대한 최소 값(cost)을 교체하는 경제적인 모델을 이용하고 있다. 파일을 저장하기 위해 충분한 공간이 있으면 새롭게 도착된 파일을 디스크상에 자동으로 저장한다. 만약 디스크가 충분한 공간이 없으면, EBR은 디스크 상에서 최소 값을 가진 파일을 선택한다. 이러한 선택을 하기위해 각 사이트에 구현된 중복 최적기(Replica Optimizer)를 이용한다. 중복 최적기는 이익(profit)을 최대화 시키기 위해 저장되어 있는 각각의 파일의 값을 로그에 유지하면서, 이 값이 미래 소득 예측 함수의 입력 값으로 사용된다. 이 예측 함수는 시간적 연관성, 지리적 연관성, 순서적 연관성을 고려한 과거의 윈도우 W 시간에 요청을 기반하여 앞으로 W 시간에 요청을 미리 예측하여 값을 산출한다.

LVCT(Least Value-based on Caching Time) 정책은 [8]에서 제안하였다. 이 정책은 얼마나 파일이 재접근되어지는가를 고려하여 파일을 교체한다. 유틸리티 함수 값에서 가장 작은 값을 지닌 파일(들)이 선택되며, 유틸리티 함수는 ((1/캐쉉 시간)* cost/파일의 크기)이다. 이 정책은 각 파일의 캐쉉 시간을 유지하기 위해 캐쉉 시간과 파일의 크기를 지닌 캐쉉 시간 스택(caching time stack)을 사용한다. 즉, 각 파일 마다 접근 되는 시간을 고려한 캐쉉 시간과 파일의 크기를 캐쉉 시간 스택에 저장하는 것이다. 예를 들어, 파일 f가 교체되기 위해 접근되면 파일 f을 스택의 꼭대기(top)로 옮긴 후에 파일 f의 캐쉉 시간을 0으로 한다. 계속해서 재접근된 파일을 스택의 꼭대기에 옮김으로써 스택의 바닥(bottom)에 있는 파일은 가장 재접근이 안 된 파일이고, 이 파일(들)을 교체한다는 것이다. 즉, 가장 큰 캐쉉 시간을 지닌 파일이 가장 접근이 되지 않은 파일이고 이 파일(들)을 교체한다.

위에서 기술한 기존 정책들의 문제점을 분석해 보면 다음과 같다. LRU와 LFU, LRU-k 정책은 널리 알려진 정책이긴 하지만 너무 적은(혹은 단순한) 정보를 기반으로 교체 파일을 한다. 또한, LCB-K 정책은 유틸리티 함수가 각 파일에 대한 미래 도착률을 예측해야 한다는 부담과 교체 후보 파일들에 대해 과거 일정 시간(k) 동안의 참조 여부를 추적해야 한다는 단점이 존재한다. 게다가 일정한 시간 밖에 있는 파일들은 유틸리티 함수에 적용되지 않으며 교체 후보도 될 수 없기 때문에 이 정책에서 선택된 파일은 교체하기 위한 최적의 선택이 아닐 가능성이 높다. EBR 정책 단점도 역시 경제적인 모델을 위해 과거의 정보를 기반으로 앞으로의 요청을 미리 예측해야한다는 점이다. LVCT 정책의 단점은 캐쉉 시간을 유지하기 위해 캐쉉 시간과 파일 크기를 지닌 스택을 유지해야한다는 점이다. 접근되는 파일을 스택의 꼭대기로 옮기는 추가적인 작업도 필요하다. 전체적은 보면 공간 확보를 위해 앞으로 상황을 예측해야하며, 필요한 공간을 확보하기위해 요청된 파일 보다 상대적으로 작은 파일들이 모든 교체될 가능성이 있다. 또한, 교체를 위해 스택과 같은 불필요한 자원도 유지하고 있음을 알 수 있다.

4. 제안 알고리즘

본 논문에서 제안한 알고리즘은 SBR-k(Size-based re-

placement-k)이다. 이 알고리즘은 기존 알고리즘들과 달리 불확실한 미래의 상황을 예측하는 부담을 피하면서도 요청된 파일에 대응되는 교체 파일의 개수가 되도록 적게 하기 위해 파일 크기를 고려하는 것이 특징이라 하겠다.

데이터 그리드 환경에서 근거리 저장 자원 관리자에 대한 파일 요청은 동시에 수백내지 수천 건이 들어 올 수 있는데 저장 자원 관리자는 이들을 먼저 큐잉 시킨다. 큐잉된 요청 메세지들은 저장 자원 관리자 내의 파일 허가 모듈(FAM)에 의해 그 처리 순서가 결정되고 처리 관정에서 파일 교체가 필요할 경우 파일 교체 모듈(FRM)에 요청 메시지를 전달한다.

파일 교체 모듈은 요청한 파일이 근거리 디스크에 있는지를 우선 확인한다. 만약, 근거리 디스크에 있으면 요청한 사용자에게 요청한 파일을 전달해 준다. 만약 근거리 디스크에 요청 파일이 없으면 원거리 디스크에 사용자 요청파일을 요구하여 근거리 디스크에 캐싱을 하는데, 이때 근거리 디스크에 공간이 없으면 파일 교체 알고리즘에 의해서 교체될 파일이 선택된다. 현재 근거리에 저장되어 있는 파일들이 선택 후보가 되는데 이를 "unpinned"이라고 하며, 선택 후보 파일 중 교체하기 위해 선택된 파일은 "pinned"이라한다.

본 논문에서 제안한 파일 교체 알고리즘의 동작 과정은 그림 2와 같다. 사용자가 요청한 파일의 크기를 r이라고 할 때, SBR-k 알고리즘은 먼저 지역적으로 현재 저장되어 있는 파일 중 크기가 r인 파일을 찾는다. 만약 r과 같은 크기의 파일이 있으면 가장 먼저 교체 파일로 선택되며 이는 SBR-k 알고리즘에서 최적 교체이다.

만약, 크기가 같은 파일이 없을 경우는 SBR-k 알고리즘에서는 k값을 고려하여 파일 교체 알고리즘을 수행하는데, 여기서 k는 사용자가 요청한 파일 크기와 비례되는 값이다. 예를 들어, 사용자가 요청한 파일의 크기가 1,000MB 이고, k값이 0.1(즉, 10%)이면, k값은 요청한 파일 크기의 10% 비율인 100MB를 의미한다. 지역 디스크에서 같은 크기의 파일이 없고, k값이 0.1일 경우 SBR-k 알고리즘은 요청 파일 크기에 k값을 더한 크기(1,100MB) 이내에서 다른 파일을 찾는다. 이 크기를 q라고 한다면, SBR-k 알고리즘은 지역 디스크에서 r보다 크고 q보다 작은 파일(들)을 찾는 것이다. 이렇게 찾은 파일들 집합을 p라고 하면 SBR-k 알고리즘은 p내의 파일(들) 중 LRU을 고려하여 최종 교체 파일을 선정하여 교체한다.

만약 지역 디스크에서 p파일이 없을 경우, SBR-k 알고리즘은 요청 파일 크기에서 k값을 뺀 크기를 갖는 파일(900MB)을 찾는다. 이 크기를 q'라고 한다면, SBR-k 알고리즘은 지역 디스크에서 r보다 작고 q'보다 큰 파일(들)을 찾는다. 이렇게 찾은 파일들의 집합을 p'라 하면, SBR-k 알고리즘에서는 p'파일(들) 중 LRU을 고려하여 최종 교체 파일을 선정하여 교체한다. 만약 지역 디스크에서 p'파일이 없을 경우에는 교체될 파일이 있을 때까지 k값을 증가하여 SBR-k 알고리즘을 계속 수행한다.

5. 평 가

전통적으로 캐쉬 교체 정책의 효율성을 평가하기 위해 사

```
if(new_requested_file_size == unpinned_file_size)
        then {
             replace file;
             exit;
        }
/* K 값을 교려 */
 k=0;
 while( 1 ){
    /* k의 초기값은 0.1(즉, 10%)이며 10%씩 증가 */
    k += 0.1;
     k_size = new_requested_file_size * k;
     i) new_requested_file_size 보다 크고 (new_requested_file_size + k_size) 보다 작은 file들을 search;
          선택된 파일들에 대해서
                    LRU 알고리즘을 고려하여, 적합한 파일을 선택;
                    replace file;
                    exit;
     ii)  new_requested_file_size 보다 작고 (new_requested_file_size - k_size) 보다 작은
           unpinned file들을 search;
           선택된 파일들에 대해서
                    LRU 알고리즘을 고려하여, 적합한 파일을 선택;
                     replace files;
                     exit;
 }
```

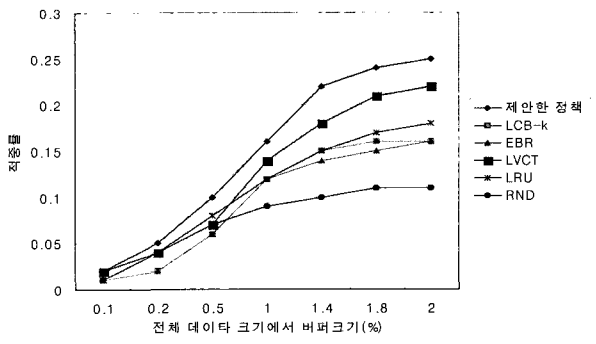
(그림 2) SBR-k 교체 정책의 동작 과정

용되는 평가 방법은 적중률(hit ratio)과 바이트 적중률(byte hit ratio)이다[5, 6]. 적중률은 전체 요청의 회수에서 캐쉬 안에 파일이 있을 비율을 나타내고, 바이트 적중률은 요청한 전체 데이터 양에 대한 캐쉬된 데이터(byte) 양의 비율을 의미한다. 적중률과 바이트 적중률은 캐쉬에서 각각 응답 시간과 대역폭에 대한 효율성 측정 척도로 이용된다.

본 논문에서 사용된 시뮬레이션 도구는 OptorSim인데 OptorSim[11, 12]은 유럽 데이터 그리드 프로젝트(European DataGrid)에서 자바로 구현된 데이터 그리드 시뮬레이션 도구이다(OptorSim을 설치하기 위해 본 논문에서는 자바 1.5.0_04 버전과 ant 1.6.5을 설치하였다.). 시뮬레이션의 환경 설정은 논문 [5, 6]에 기반을 한다. 파일의 크기는 1.0에서 2.1GB 범위로 한다. 요청에 대해서는 평균 10초의 포아송 도착시간을 이용한다. 파일 크기는 5,000,000 바이트와 2,147,000,000 바이트 사이에서 균일하게 분산되어 있다. 임의의 간격으로 파일을 요청하며, 참조는 80-20 규칙의 참조 지역성(locality of reference) 규칙을 적용하였다. 80-20 규칙은 80%의 요청이 파일의 20%을 재 참조하는 것을 의미한다. 전체 20개 사이트에서 1.1TB 저장할 수 있으며, 18개 사이트에서는 50GB를 2개의 사이트에서는 100GB를 저장할 수 있다.

그림 3과 그림 4는 제안한 정책과 LCB-K, EBR, LVCT, LRU, RND(Random)에 대한 각 사이트의 평균 버퍼의 적중률과 바이트 적중률을 나타내고 있다. 전체적으로 버퍼 크기가 작을 경우에는 제안한 정책과 기존 정책을 비교하며 적중률의 차이는 크게 발생하고 있지 않으나, 버퍼의 크기가 커지면서 적중률도 커지고 있다. 그림 3에서 전체 버퍼의 크기가 1%인 11GB까지 비슷한 적중률(제안한 정책 0.16, LCB-k 0.12, EBR 0.12 LVCT 0.14, LRU 0.12, RND 0.09)을 보이고 있으나, 그 이후에 버퍼의 크기를 2%인 22GB에서는 제안한 정책의 적중률(0.25)이 LCB-K(0.16), EBR(0.16), LVCT(0.22), LRU(0.18) RND(0.11)비해 성능이 좋음을 알 수가 있다.

바이트 적중률을 나타내는 그림 4에서도 버퍼 크기가 작으면 비슷한 성능을 나타내나, 버퍼의 크기를 2%일 경우 제안한 정책의 적중률(0.22)이 LCB-K(0.18), EBR(0.17), LVCT(0.2), LRU(0.18) RND(0.14)비해 성능이 좋음을 알 수가 있다. 본 논문에서 제안한 정책은 교체될 파일의 크기를 되도록 새로운 파일의 크기와 비교하여 가장 비슷한 파일을 선정하기 때문이다. 그림 3과 그림 4의 성능 평가에서 임의로 교체 파일을 선택하는 RND 정책의 적중률이 가장 낮았으며 LCB-K나 EBR 정책은 버퍼 크기가 작으면 다른 정책과 비슷한 적중률을 나타내었으나, 버퍼 크기가 증가함에 따라 교체될 파일의 예측 확률 낮아 적중률이 낮게 나타내고 있다.



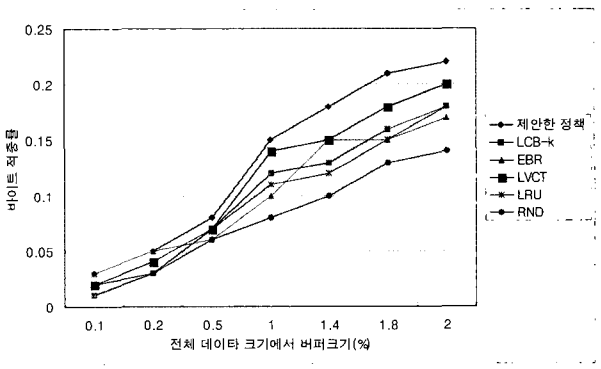
(그림 3) 데이터 그리드에서 적중률

6. 결 론

지리적으로 분산되어 있는 대용량의 데이터를 효율적으로 처리하기 위해 데이터 그리드 컴퓨팅에 대한 연구는 계속 진행되고 있다. 기존의 웹 캐쉬, 가상 메모리에서 파일 교체 정책과는 다르게 데이터 그리드 환경에서의 파일 교체 전략은 파일 크기가 대용량이며, 이에 대한 캐쉬 용량도 대용량이어서 복잡하다.

기존 논문에서 제안되고 있는 LRU나, LCB-K, EBR, LVCT같은 정책은 너무 단순한 정보로 파일을 교체하거나, 추가적인 자원, 불확실한 미래에 대한 예측에 기반하고 있다. 이를 해결하기 위해 본 논문은 파일 크기에 기반하여 교체하는 SBR-k을 제안하였다. 성능평가 결과 캐쉬 크기가 작을 경우에는 적중률이 비슷하나, 캐쉬 크기가 크면 LCB-K, EBR, LVCT, LRU, RND에 비해 본 논문에서 제시한 정책이 우수함을 보였다.

향후 연구로는 본 논문에서 제시한 정책을 파일에 대한 우선순위도 고려하여 파일 교체 정책을 수행하는 일이다. 또한 만약에 파일이 멀티미디어 데이터로 구성되어 있을 경우 제한된 저장 공간을 고려한 미디어 별 교체 정책을 고려해야할 것이다.



(그림 4) 데이터 그리드에서 바이트 적중률

참 고 문 헌

[1] B. Allcock, I. Foster, V. Nefedova, A. Chervenak, D. Deelman etc, "High-performance remote access to climate simulation

data : A challenge problem for data grid technologies," Proceeding of the SuperComputing Conference, Non, 2003.
[2] K. Iloltman, "Data grid system overview and requirements," Technical report, CERN, July, 2001.
[3] M. Russel, G. Allen, G. Daues, I. Foster, E. Seidel, J. Novotny, J. Shalf and G. von Laszewski, "The astophysics simulation collaboratory : A science portal enabling community software deveopment," Cluster Computing, 2002.
[4] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury and S. Tueckes, "The data grid : Towards an architecture for the distributed management and analysis of large scientific data-sets," Journal of Network and Computer Applications, pp.187-200, 2002.
[5] E. J. Otoo, F. Olken and A. Shoshani, "Disk Cache replacement algorithm for storage resource managers in data grids," In The 15th Annual Super Computer Conference, pp.1-15, Nov., 2002.
[6] J.H Abawajy, "File Replacement Algorithm for Storage Resource Managers in Data Grids," LNCS 3038, pp.339-346, 2004.
[7] M. Carman, F. Serafini, L. Stokinger, K. Stockinger, "Towards an Economy-Based Optimisation of File Access and Replication on a Data Grid," In Procedings of 2nd CCGRID, pp.120-126, 2002.
[8] S. Jiang, X. Zhang, "An Efficient Distributed Disk Caching in Data Grid Managemnet," In Proceedings of Cluster Computing, 2003.
[9] S. Aberham, G. Peter Baer, G. Greg, Operating system principles, 7th, Wiley, 2006.
[10] E. J. Otoo, F. O'Neil, G. Weilum, "The LRU-K page repalcement Algorithm for Database Disk Buffering," The 15th Annual Super Computer Conference, Nov., 2002.
[11] Daid G. Gameron, Ruben Carvajal-Schiaffino, Jamie Ferguson, OptorSim v2.0 Installation and User Guid, http://cern.ch/edg-wp2/optimization/opotorsim.html
[12] W.H.Bell, D.G. Gameron, "Optorsim - A grid simulator for studying dynamic data replication strategies," Journal of High Performance Computing Application, 2003.

박 홍 진

e-mail : hjpark1@sangji.ac.kr
1993년 원광대학교 컴퓨터공학과(공학사)
1995년 중앙대학교 컴퓨터공학과(공학석사)
2001년 중앙대학교 컴퓨터공학과(공학박사)
2001년~현재 상지대학교 컴퓨터정보공학부 조교수
관심분야 : 분산 시스템, 운영체제, 무선 모니터링 및 관리