논문 2008-45CI-2-2

데이터 그리드를 위한 효율적인 Peer-to-Peer 기반 복제 정책

(An Efficient Peer-to-Peer Based Replication Strategy for Data Grid)

오 상 원*, 이 원 주**, 전 창 호***

(Sang Won Oh, Won Joo Lee, and Chang Ho Jeon)

요 약

본 논문에서는 데이터 그리드 시스템의 성능을 향상 시킬 수 있는 Peer-to-Peer 기반의 효율적인 복제 정책을 제안한다. 이 정책은 기존의 데이터 복제정책에 Peer-to-Peer 개념을 추가하여 계층구조로 인한 데이터 전송 제한을 줄인다. 그리고 클라이언트 계층 노드의 저장장치 일정 부분을 임계구역(critical section)으로 설정하여 데이터 복사본을 저장할 수 있도록 함으로써 클라이언트 계층 노드간의 데이터 복사본 전송과 클라이언트 계층 노드에서 상위 계층의 데이터 복제 서버로 데이터 복사본 전송이 가능하도록 한다. 클라이언트 계층 노드간의 데이터 복사본 전송은 데이터 서버 또는 데이터 복제 서버에서 데이터 복사본을 전송하는 것에 비해 전송시간을 최소화 할 수 있기 때문에 데이터 그리드 시스템의 성능을 높일 수 있다. 본 논문에서는 시뮬레이션을 통하여 제안하는 Peer-to-Peer 기반 데이터 복제 정책이 기존의 복제 정책에 비해 그리드 시스템의 성능을 향상시킨다는 것을 보인다.

Abstract

In this paper, we propose an efficient data replication strategy based on Peer-to-Peer which improves the performance of Data Grid system. The key idea of this replication strategy is to add the Peer-to-Peer concept for reducing data transmission restriction caused by hierarchical topology. And, it makes the nodes can store data replica to set a critical section at the storage of client level nodes. Therefore, it is possible to transmit the data replica between client level nodes and from client level nodes to upper data replication server. It is more effective to transmit data replica between client level nodes than transmitting data replica from data server or data replication server with respect to minimize the transmission time. This results in improving the performance of Data Grid system. Through simulation, we show that the proposed data replication strategy based on Peer-to-Peer improves the performance of entire Data Grid environment compared to previous strategies.

Keywords: 데이터 그리드(Data Grid), 복제정책(Replication Strategy), Peer-to-Peer(P2P)

I. 서 론

데이터 그리드[1]는 지리적으로 분산되어 있는 컴퓨팅 자원과 저장 자원을 공유하는 컴퓨팅 환경이다[2, 3]. 광대역폭 네트워크를 사용하는 유로 데이터 그리드는 데이터 그리드의 한 예이다. 유로 데이터 그리드[4]는 생물정보, 고 에너지 물리 실험, 지구과학 등의 연구에 관련된 데이터 복사본을 분산된 데이터 서버에 효율적으로 배치함으로써 대학 및 연구기관에 특수한 연구 환경을 제공한다[5~10].

데이터 그리드는 그림 1과 같은 계층형 연결구조를 가진다. 이러한 계층구조에서는 상위 계층일수록 저장장치의 용량과 계층 간의 연결 대역폭이 증가한다[11].

계층 1은 데이터 그리드의 모든 데이터를 생성하는 메인 데이터 서버이다. 계층 1의 노드를 제외한 다른 노드에서는 데이터의 생성 및 수정이 불가능하다. 계층 2와 3은 데이터 저장 기능을 하는 중간 노드로 구성한다. 이 계층에서는 클라이언트 노드에서 요청하는 데이

* 정회원, 에어코드 DTV 연구소
(DTV R&D Center, Aircode)
** 정회원-교신저자, 인하공업전문대학
컴퓨터정보공학부 컴퓨터정보과
(Department of Computer Science, Inha Technical College)
*** 평생회원, 한양대학교 전자컴퓨터공학부
(School of Electrical and Computer Engineering, Hanyang University)
접수일자: 2007년11월2일, 수정완료일: 2008년2월29일

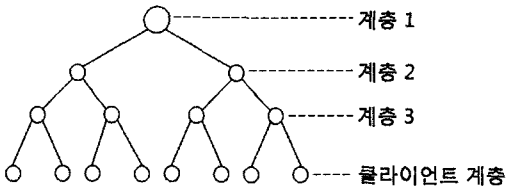

그림 1. 데이터 그리드연결 구조의 예
Fig. 1. Example of Data Grid topology.

터 접속시간을 최소화하기 위해 클라이언트에 근접한 데이터 복제 서버에 데이터 복사본을 저장한다. 클라이언트 계층은 최하위 수준에 위치하며, 계층 1에서 생성된 데이터나 계층 2, 3에서 유지되고 있는 데이터 복사본에 대하여 필요한 데이터 전송을 요청한다.

이러한 연결구조는 각 계층 간의 기능을 엄격하게 구분하고, 구조적 종속성이 존재하기 때문에 상위계층에서 하위계층으로만 데이터 복제가 가능하다는 문제점이 있다. 즉, 그림 1에서 계층 3의 데이터 복제 서버는 계층 2의 데이터 복제 서버의 복사본 중 일부만을 저장할 수 있다. 그리고 클라이언트 계층에서는 데이터 요청만 가능하다. 이러한 제한 사항은 데이터 복제 서버에 저장된 복사본의 대체를 자주 발생시키기 때문에 데이터 그리드의 성능을 저하시키는 요인이 된다. 따라서 데이터 그리드의 성능 향상을 위해서는 데이터 복사본을 효율적으로 유지 관리하는 데이터 복제 정책이 필요하다.

본 논문에서는 기존의 복제 정책에 Peer-to-Peer 개념을 추가하여 데이터 복제 비용을 최소화 할 수 있는 Peer-to-Peer 기반 복제 정책을 제안한다. 이 정책은 하위 계층의 데이터 복제 서버뿐만 아니라 해당 데이터 복사본을 요청한 그룹에서 요청 횟수가 최고인 데이터 복제 서버 혹은 클라이언트 계층에서도 데이터 복사본 작성이 가능하다는 것이다. 즉, 그림 1의 계층 1에 해당하는 데이터 서버를 제외한 각 노드들은 Peer-to-Peer 연결로 데이터를 전송함으로써 계층 구조로 인한 성능 저하 문제점을 최소화한다. 이때 클라이언트 계층 간의 전송은 원거리의 클라이언트 간의 전송이 아니라 동일한 LAN 상에 존재하는 클라이언트 간의 전송이다. 그리고 각 노드의 저장 공간 관리를 위한 오버헤드는 크지 않다고 가정한다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 기존의 데이터 그리드 복제 정책에 대하여 설명한다. 그리고 Ⅲ장에서는 제안하는 복제 정책에 대하여 상세히 기술한다. Ⅳ장에서는 시뮬레이션을 통하여 제안하는 복제 정책이 기존의 복제정책에 비해 우수함을 검증하고, Ⅴ장에서 결론을 맺는다.

Ⅱ. 기존의 데이터 그리드 복제 정책

데이터 그리드를 위한 기존의 복제 정책은 6가지로 분류할 수 있다[11]. 이 정책들은 그림 1의 계층구조를 기반으로 한다. 각 정책에 대한 상세한 설명은 다음과 같다.

1. 무 복제 정책 (No Replication)

무 복제 정책[3]은 데이터 복사본을 작성하지 않는 것이다. 그러므로 이 정책을 적용한 데이터 그리드 시스템은 데이터가 필요할 때 마다 계층 1의 메인 데이터 서버에 해당 데이터를 요청하여 사용한다. 계층 2와 계층 3의 데이터 복제 서버들은 아무 역할도 하지 않는다. 이 정책은 단순하지만 성능이 낮다는 것이 단점이다.

2. 최적 클라이언트 정책(Best Client)

최적 클라이언트 정책[3]은 기존의 계층구조를 탈피하여 데이터의 요청이 가장 많았던 데이터 복제 서버에 데이터 복사본을 배치하는 것이다. 각 노드는 모든 데이터에 대한 히스토리(요청 횟수, 요청자 정보)를 각각 유지·관리한다. 이 히스토리 정보를 통해 데이터 복사본을 복제할 최적의 노드를 찾는다. 해당 데이터의 요청 횟수가 임계값을 초과하면 그 데이터를 가장 많이 요청한 노드에 해당 데이터의 복사본을 복제를 한다. 데이터 복사본을 최적의 위치에 배치하는 이 정책은 6가지 정책들 중에 평균 성능이 가장 우수하다. 그러나 상위 계층의 저장 공간이 낭비되고 하위 계층의 저장 공간에 모든 데이터가 집중되기 때문에 저장 공간을 효율적으로 사용하기 힘들다.

3. 연속형 복제 정책(Cascading Replication)

연속형 복제 정책[3]은 저장 공간의 효율성을 높일 수 있다. 데이터 복사본은 상위 데이터 복제 서버에 먼저 저장되고, 저장 공간이 부족할 경우 하위 데이터 복제 서버에 저장된다. 즉, 그림 1에서 계층 1의 데이터 서버에서 계층 2의 데이터 복제 서버로 데이터 복사본이 전달된다. 계층 2의 데이터 복제 서버의 저장 공간이 부족하면 계층 3의 데이터 복제 서버로 데이터 복제가 일어난다. 이 정책은 다른 정책에 비해 데이터 복제 서버 저장 공간의 효율성이 높다. 하지만 데이터 복사본이 계층구조를 따라 전달되기 때문에 전체 시스템의 성능이 저하되는 단점이 있다.

4. 보통 캐싱 정책(Plain Caching)

보통 캐싱 정책[3]은 클라이언트 노드들이 캐싱 기능을 가지고 있다. 기존 정책의 클라이언트 계층은 요청을 통해 전송 받은 데이터를 사용한 후 그 데이터를 다시 사용하지 않는다. 하지만 이 정책에서는 클라이언트 계층에서 사용했던 데이터가 바로 다음 작업에서 필요할 경우 재전송을 요청하지 않고 바로 사용할 수 있다. 따라서 시간 집약성이 높은 작업을 수행할 경우에 최대의 성능을 얻을 수 있지만 그 이외의 경우에는 성능 면에서 큰 이득이 없다.

5. 캐싱이 추가된 연속형 복제 정책(Caching plus Cascading Replication)

캐싱이 추가된 연속형 복제 정책[3]은 연속형 복제 정책과 보통 캐싱 정책의 장점들을 채택한 정책이다. 기본적으로는 연속형 복제 방식을 따르며 클라이언트 계층에서 사용했던 데이터를 저장하여 재사용할 수 있다. 그리고 서버는 정기적으로 참조 빈도가 높은 데이터의 복사본을 하위 계층으로 전달한다. 이 정책은 보통 캐싱 정책과 연속형 복제 정책의 장점을 모두 가지고 있지만 시간 집약성이 낮은 작업을 수행하면 연속형 복제 정책과 같은 성능을 보인다.

6. 고속 확산 정책(Fast Spread)

고속 확산 정책[3]은 데이터의 복사본을 전 시스템에 빠르게 전송한다. 클라이언트 계층이 메인 데이터 서버에 데이터를 요청하면 해당 데이터는 메인 데이터 서버와 요청자 클라이언트 사이의 경로에 존재하는 모든 데이터 복제 서버에 데이터 복사본을 저장한다. 짧은 시간 안에 데이터 복사본이 확산되어 시스템의 성능을 향상시킬 수 있다. 하지만 클라이언트 노드에서 데이터의 대체가 빈번하게 일어나는 작업을 수행할 경우 데이터 복제 서버에서의 데이터 복사본 대체가 오버헤드가 되어 시스템의 성능을 저하시키는 단점이 있다.

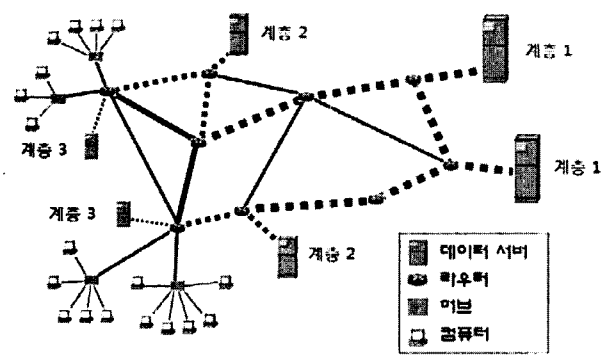

그림 2. 네트워크 연결 구조
Fig. 2. Network topology.

기존의 데이터 그리드 복제 정책을 사용하는 환경에서는 계층 간의 기능적 구분으로 인해 데이터 복사본에 종속성이 존재한다. 즉, 그림 2와 같은 네트워크 연결구조에서는 계층 간의 기능적 구분 때문에 점선으로 표시된 경로를 통하여 상위 계층에서 하위 계층으로만 데이터 복사본을 전송한다. 이때 실선으로 표시된 연결 경로는 사용되지 않기 때문에 네트워크 대역폭을 낭비하게 되는 문제점이 있다.

Ⅲ. 제안하는 Peer-to-Peer 기반의 복제 정책

본 논문에서는 기존의 데이터 그리드 복제 정책의 문제점을 해결하기 위해 동일 수준의 계층에서뿐만 아니라 상/하위 계층 간의 데이터 복사본 이동이 가능한 Peer-to-Peer 기반의 복제 정책을 제안한다. 이 복제 정책은 다음과 같이 두 가지 방법을 적용한다.

첫 번째 방법은 계층의 기능적 구분에 따른 데이터 복사본의 종속성을 제거하는 것이다. 즉, 데이터를 생성하는 계층 1을 제외한 다른 계층의 기능적 구분을 제거함으로써 상위 계층의 데이터 복제 서버에 존재하는 데이터 복사본만을 하위 계층의 데이터 복제 서버에 저장할 수 있도록 데이터 복사본의 종속성을 제거한다.

두 번째 방법은 기존의 그리드 환경에 Peer-to-Peer 전송 방식을 적용하여 동일한 계층의 클라이언트 노드들 간에 데이터 복사본 전송이 가능하도록 한다. 각 클라이언트 노드는 임계구역(critical section)으로 지정된 저장 공간을 가진다. 임계구역은 저장 장치의 일부 영역으로 저장된 데이터 복사본을 수정할 수 없는 공간이다. 이 임계구역을 이용하여 클라이언트 계층 노드간의 데이터 복사본 전송과 클라이언트 계층에서 상위 데이터 복제 서버로의 데이터 복사본 전송을 하게 된다. 클라이언트 계층간의 데이터 복사본 전송은 데이터 서버 또는 데이터 복제 서버에서 데이터 복사본을 전송하는 것에 비해 전송시간을 최소화 할 수 있다.

1. 데이터 복제 정책

데이터 그리드의 각 계층별로 동작 방식에 약간의 차이점이 존재한다. 제안하는 데이터 복제 정책은 그림 3과 그림 4를 예로 들어 설명한다. 그리고 메인 데이터 서버부터 클라이언트까지 각 부분의 기능을 상세히 설

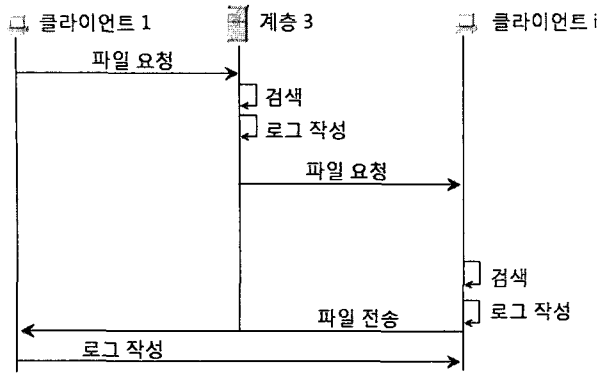

그림 3. Peer-to-Peer를 이용한 파일 전송 예 1
Fig. 3. Example 1 of file transmission using Peer-to-Peer.

명한다. 그리드 미들웨어는 각 노드의 로그 정보를 포함한 메타데이터를 제공한다. 각 노드는 데이터에 대한 요청 수와 요청자 정보에 해당하는 로그 정보를 기록한다. 이 로그 정보는 해당 데이터를 가장 빠르게 복제할 수 있는 노드를 검색할 때 이용된다.

그림 3에서 클라이언트 1은 자신이 속해있는 그룹의 데이터 복제 서버인 계층 3에 데이터 요청을 전송한다. 계층 3은 보유하고 있는 로그 정보를 이용하여 데이터 복사본의 전송시간을 최소화 할 수 있는 노드를 검색한다. 만일, 검색 결과 요청 받은 데이터의 복사본이 계층 3에 존재하면 기존의 정책들과 같은 방법으로 로그 정보를 갱신하고 해당 데이터 복사본을 클라이언트 1로 전송한다. 하지만 계층 3이 아닌 클라이언트 i에서 데이터 복사본을 전송 받는 것이 전송비용을 최소화 할 수 있다면 계층 3에 해당 데이터 복사본이 존재하더라도 클라이언트 i의 해당 데이터 복사본을 클라이언트 1에 전송할 것을 요청한다. 이때 계층 3은 로그 정보를 갱신한다. 클라이언트 i는 요청 받은 데이터의 복사본을

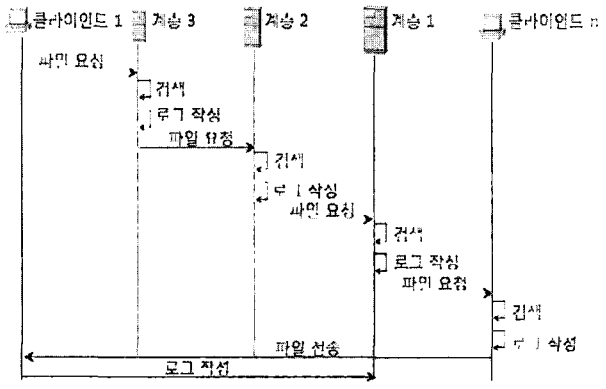

그림 4. Peer-to-Peer를 이용한 파일 전송 예 2
Fig. 4. Example 2 of file transmission using Peer-to-Peer.

클라이언트 1에 전송하고 로그 정보를 작성한다.

계층 3보다 더 상위 계층으로 요청이 전송되는 경우는 그림 4에서 설명한다.

그림 4에서 계층 3까지의 해당 데이터 요청 전달 및 처리 과정은 그림 3의 과정과 동일하다. 하지만 그림 4에서는 계층 3의 로그 정보 검색 결과 해당 데이터 복사본을 찾지 못했기 때문에 데이터 요청은 계층 3 보다 상위 계층이면서 같은 그룹의 데이터 복제 서버인 계층 2로 전달된다. 계층 2는 계층 3과 같은 방법으로 로그 정보를 통해 최적의 노드를 검색한다. 계층 2의 데이터 복제 서버의 로그 정보에서도 검색에 실패하면 다시 그 정보를 메인 데이터 서버인 계층 1에 전송한다. 계층 1에서도 자신이 보유하고 있는 데이터를 무조건 전송하는 것이 아니라 로그 정보를 통해 최적의 노드를 검색한다. 검색 결과 클라이언트 n에 해당 데이터의 복사본이 존재하면 계층 1은 클라이언트 n에게 해당 데이터를 클라이언트 1에 전송하도록 요청한다. 클라이언트 n은 해당 데이터 복사본을 클라이언트 1에 전송한다. 데이터 복사본의 복제가 완료되면 클라이언트 1은 자신 그룹의 데이터 복제 서버들에 갱신된 로그 정보를 전송한다.

2. 데이터 복제 알고리즘

Peer-to-Peer 기반의 데이터 복제 정책은 메인 데이터 서버 관리, 데이터 복제 서버의 데이터 요청, 클라이언트의 데이터 요청 알고리즘으로 구성된다.

2.1 메인 데이터 서버 관리 알고리즘

메인 데이터 서버는 데이터를 생성하여 그리드 환경에 등록한다. 그리고 데이터에 대한 요청을 받으면 해당 데이터의 복사본을 전송하고 관련 정보를 유지 관리하는 기능을 한다. 메인 데이터 서버 관리 알고리즘은

```
void createFile (파일명) // 파일 생성
{
  파일 생성 및 로그 작성
}
int requestFileSend (사이트 명, 파일명)
{
  파일의 최적 위치 검색
  if (파일이 서버에 존재할 경우) {
    해당 클라이언트로 파일 전송
  } else if (파일이 하위 계층에 존재할 경우) {
    클라이언트 or 데이터 복제 서버에서 P2P로 파일 전송
  }
}
```

그림 5. 메인 데이터 서버 관리 알고리즘
Fig. 5. Main data server management algorithm.

그림 5와 같다.

그림 5의 createFile() 함수는 메인 데이터 서버에 존재하지 않는 데이터를 생성하여 저장 장치에 저장하고 그 데이터에 대한 로그를 작성한다. requestFileSend() 함수는 클라이언트혹은 데이터 복제 서버와 같은 하위 계층의 데이터 요청을 처리한다. 데이터 요청 처리는 먼저 그리드 미들웨어에서 제공하는 메타데이터를 이용하여 데이터의 위치를 검색하고 로그를 작성한다. 그리고 파일이 서버에 존재하면 해당 클라이언트로 파일을 전송한다. 그러나 파일이 서버에 존재 않으면 Peer-to-Peer 연결을 이용하여 다른 계층의 클라이언트 또는 데이터 복제 서버에서 데이터를 전송하도록 한다.

2.2 데이터 복제 서버의 데이터 요청 알고리즘

데이터 복제 서버에서는 하위 계층, 동일 계층, 그리고 상위 계층의 데이터 전송 요청을 처리한다. 이때 요청된 데이터 복사본이 존재하는 최적의 노드를 찾아 해당 노드에 데이터 복사본의 전송을 요청하고, 관련 정보를 유지 관리한다. 데이터 복제 서버의 파일 전송 요청 알고리즘은 그림 6과 같다.

그림 6의 ReplicationServer_requestFileToUpper() 함수는 그리드 미들웨어에서 제공하는 메타데이터를 이용하여 파일 로그를 작성하고, 파일의 최적 위치를 검색한다. 그러나 파일이 데이터 복제 서버에 존재하지 않고 하위 계층의 노드에 존재하면 하위 계층 노드에서 파일을 전송한다. 또한 파일이 상위 계층에 존재하면 상위 계층의 노드에서 파일을 전송한다.

또한 데이터 복제 서버에서 로그 정보에 기록된 데이터의 요청 횟수가 일정 한계값을 초과하면 그림 7의 데이터 복사본 요청 알고리즘을 실행한다. 해당 파일의 위치를 검색한 후 파일이 하위 또는 상위 계층에 존재하면 각각 하위 계층 노드와 상위 계층 노드에 데이터

```
int ReplicationServer_requestFileToUpper(사이트명, 파일명)
{
  파일에 대한 로그 작성
  파일의 최적 위치 검색
  if (파일이 데이터 복제 서버에 존재할 경우) {
    해당 클라이언트로 파일 전송
  } else if (파일이 하위 계층에 존재할 경우) {
    하위 계층에 파일 전송 요청
  } else if (파일이 상위 계층에 존재할 경우) {
    상위 계층에 파일 전송 요청
  }
}
```

그림 6. 데이터 복제 서버의 파일 전송 요청 알고리즘
Fig. 6. File transmission request algorithm of data replication server.

```
if (요청 횟수가 임계 값에 도달했을 경우)
{
  파일의 최적 위치 검색
  if (파일이 하위 계층에 존재할 경우) {
    하위 계층에 파일 전송 요청
  } else if (파일이 상위 계층에 존재할 경우) {
    상위 계층에 파일 전송 요청
  }
  if (클라이언트의 저장 공간이 부족할 경우) {
        삭제 파일검색 및 삭제
        로그 갱신 요청
  }
  파일 전송
  로그 갱신 요청
}
```

그림 7. 데이터 복제 서버의 데이터 복사본 요청 알고리즘
Fig. 7. Data replica request algorithm of data replication server.

전송을 요청한다. 이때 데이터를 전송 받은 클라이언트에서는 데이터 복제를 위해 먼저 저장 장치의 여유 공간을 확인하다. 여유 공간이 존재하면 데이터를 복제하고 로그 정보 갱신을 요청한다. 그러나 여유 공간이 부족할 경우에는 요청 횟수가 적은 데이터를 순서에 따라 삭제하여 여유공간을 확보한 후 복제한다. 데이터 복사본 생성이 완료되면 상위 계층의 데이터 복제 서버에 로그 정보 갱신을 요청 한다.

2.3 클라이언트의 데이터 요청 알고리즘

클라이언트 계층은 데이터 복사본 전송을 요청한다. 데이터 복사본의 전송 요청이 일정 한계값을 초과하면 데이터 복사본을 임계구역에 저장하고, 상위 계층의 데이터 복제 서버에 로그 정보 갱신을 요청한다. 클라이언트의 데이터 요청 알고리즘은 그림 8과 같다.

```
int Client_requestFile(사이트명, 파일명)
{
  파일의 최적 위치 검색
  if (파일이 클라이언트에 있는 경우) {
    파일 사용
  } else {
    if (파일이 상위 계층에 있을 경우) {
      상위 계층에 파일 전송 요청
    } else if (파일이 다른 클라이언트에 있을 경우) {
      해당 클라이언트에 P2P를 이용한 파일전송 요청
      }
    if (클라이언트의 저장 공간이 부족할 경우) {
        삭제 파일검색 및 삭제
        로그 갱신 요청
    }
    파일 전송
    로그 갱신 요청
  }
}
```

그림 8. 클라이언트의 데이터 요청 알고리즘
Fig. 8. Data request algorithm of client.

그림 8의 Client_requestFile() 함수는 그리드 미들웨어에서 제공하는 메타데이터를 이용하여 파일의 최적 위치를 검색한다. 파일이 클라이언트에 존재하면 복제하지 않고 해당 파일을 사용한다. 그러나 파일이 클라이언트에 존재하지 않으면 상위 또는 다른 클라이언트 계층에 파일 전송을 요청한다. 특히 다른 클라이언트 계층의 노드는 Peer-to-Peer 연결을 이용하여 데이터를 전송한다. 이때 데이터를 전송 받은 클라이언트에서는 데이터 복제를 위해 먼저 저장 장치의 여유 공간을 확인하다. 여유 공간이 존재하면 데이터를 복제하고 로그 정보 갱신을 요청한다. 그러나 여유 공간이 부족할 경우에는 요청 횟수가 적은 데이터를 순서에 따라 삭제하여 여유 공간을 확보한 후 복제한다.

Ⅳ. 성능 평가

본 논문에서는 시뮬레이션을 통하여 제안한 Peer-to-Peer 기반의 복제 정책이 기존의 복제 정책에 비해 성능이 우수함을 검증한다.

1. 시뮬레이션 환경

본 논문에서는 시뮬레이터로 OptorSim 1.1[12~13] 버전을 사용한다. 시뮬레이션 환경은 표 1과 같다.

표 1. 시뮬레이션 환경
Table 1. Environment of simulation.

네트워크	대역폭: 45Mbps ~ 155Mbps 토폴로지: 트리(그림 1 참조)
컴퓨팅 자원 (사이트 별)	계산소자 수: 0 ~ 1 저장용량: 1GB ~ 1000GB
작업 특성	작업수: 500 각 작업의 실행시간: 1000 작업 크기: 0.5GB 작업 요청율: Random
참여 사이트	데이터 서버 수: 2 라우터 수: 4 클라이언트 수: 11

네트워크 토폴로지는 그림 1와 같이 다중 계층 데이터 그리드로, 2개의 데이터 서버, 4개의 라우터, 11개의 클라이언트로 총 17개의 사이트를 사용한다[14]. 각 계층 간의 네트워크 대역폭은 45 Mbps~155Mbps로 설정한다. 그리고 계층 1, 2, 3과 클라이언트 계층 노드의 검색 시간은 모두 동일하다고 가정한다.

2. 시뮬레이션 결과

본 논문에서는 총작업실행시간, 평균작업실행시간, 외부 참조를 성능 평가 척도로 사용한다. 작업실행시간은 그리드 환경에 할당된 하나의 작업을 처리하는데 소요되는 시간이다. 총작업실행시간은 모든 작업의 실행시간을 합산한 것이다. 평균작업실행시간은 클라이언트의 총작업실행시간을 작업 수로 나누어 구한다. 외부 참조 수는 할당 받은 작업을 수행하면서 다른 클라이언트의 파일을 참조한 횟수이다.

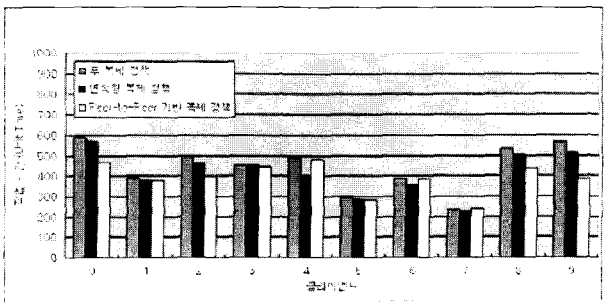

그림 9. 각 클라이언트의 평균작업실행시간
Fig. 9. Mean execution time of an each client.

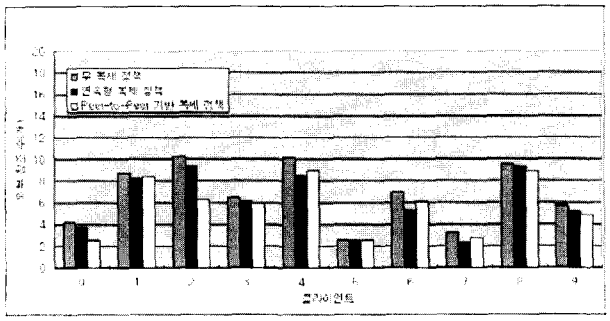

그림 10. 각 클라이언트의 작업평균외부참조 수
Fig. 10. Mean number of remote reads of an each client.

각 클라이언트의 평균작업실행시간을 측정한 결과는 그림 9와 같다.

그림 9를 살펴보면 Peer-to-Peer 기반 복제 정책은 무 복제 정책에 비해 평균작업실행시간 면에서 좋은 성능을 보인다. 하지만 클라이언트 4, 6, 7에서는 연속형 복제 정책의 성능이 가장 우수한 것으로 보인다. 그 이유는 Peer-to-Peer기반 복제 정책이 연속형 복제 정책에 비해 작업 당 평균 외부 참조 횟수가 많았기 때문이다. 이것은 그림 10의 클라이언트별 작업평균외부참조 수를 분석해 보면 알 수 있다.

그림 10은 각 클라이언트의 작업평균외부참조 수를 측정한 결과이다. 작업평균외부참조 수는 각 클라이언트에서 할당 받은 작업을 처리하기 위해 외부 데이터를 참조한 횟수를 작업의 수로 나눈 것이다.

그림 10에서 Peer-to-Peer 기반 복제 정책과 연속형 복제 정책의 외부 참조 횟수가 적은 것으로 나타난다.

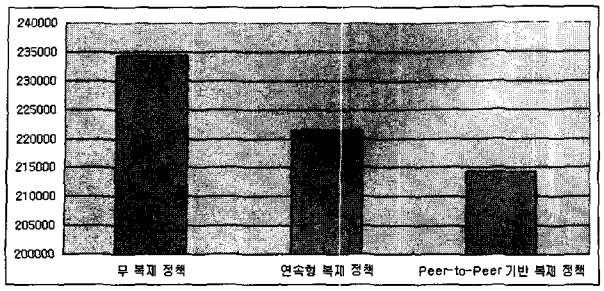

그림 11. 총작업실행시간
Fig. 11. Total job execution time.

클라이언트 4, 6, 7에서는 연속형 복제 정책에 비해 Peer-to-Peer 기반 복제 정책의 작업 당 평균 외부 참조 횟수가 많은 것으로 나타났다. 이러한 외부 참조 수의 증가로 인해 그림 9의 클라이언트 4, 6, 7에서 Peer-to-Peer 기반 복제 정책의 작업 시간이 증가한 것이다. 이것은 동일 그룹에 속한 각 클라이언트에서 여러 형태의 작업을 처리하면서 발생하는 빈번한 데이터 복제로 인한 부하 증가 때문이다.

각 복제 정책에 대한 전체 데이터 그리드의 총작업실행시간은 그림 11과 같다. 그림 11에서 Peer-to-Peer 기반 복제 정책의 총작업실행시간은 무 복제 정책에 비해 9% 줄었고, 연속형 복제 정책에 비해 4% 줄었음을 볼 수 있다. 따라서 Peer-to-Peer 기반 복제 정책의 성능이 가장 우수함을 알 수 있다. Peer-to-Peer 기반 복제 정책은 데이터 복사본 요청이 빈번하게 발생하는 클라이언트에 해당 데이터의 복사본을 배치하기 때문에 다른 복사 정책에 비해 성능이 우수하다.

V. 결 론

본 논문에서는 데이터 그리드의 기존 복제 정책에 각 노드 간의 Peer-to-Peer 연결 구조를 고려하여 데이터 복사본 작성시 전송시간을 최소화하는 노드를 찾아 복사본을 작성하는 정책을 제안하였다. 또한 작업요청 기능만을 갖는 클라이언트 노드에 저장 기능을 부여하여 데이터 복사본을 저장할 수 있게 함으로써 전체적인 저장 용량을 증가시켰다. 이 Peer-to-Peer 기반 복제 정책의 특징은 데이터 복사본이 상위 계층의 데이터 복제 서버 혹은 주 데이터 서버에서만 작성되는 것이 아니라, 데이터의 전송 비용을 최소화하면서 네트워크 대역폭의 점유율을 낮출 수 있는 노드를 찾아 해당 데이터를 복제한다. 이때 기존의 복제 정책과는 달리 클라이언트 노드에도 데이터의 복사본을 저장할 수 있다. 이 정책은 데이터 복제 비용을 감소시키고, 네트워크 대역폭의 점유율을 낮추어 전체적인 데이터 그리드 시스템의 성능을 향상시킨다.

본 논문에서는 시뮬레이션을 통하여 제안하는 Peer-to-Peer 기반 복제 정책이 기존의 데이터 복제 정책에 비해 평균작업시간과 작업평균외부참조 수를 줄이는데 효과가 있음을 보였다. 그리고 총작업실행시간을 줄여 전체 데이터 그리드 시스템의 성능을 향상시킬 수 있음을 보였다.

참 고 문 헌

[1] W. Hoscheck, J. Jaen-Martinez, A.Samar, H. Stockinger, and K. Stockinger, "Data Management in an International Data Grid Project," Proceedings of the First IEEE/ACM International Workshop on Grid Computing (Grid'2000), Lecture Notes In Computer Science, Vol. 1971, ISBN: 3-540-41403-7, pp. 77-90, Bangalore, India, Dec. 2000.

[2] Ian Foster, Carl Kesselman, "The Grid 2: Blueprint for a New Computing Infrastructure," Morgan Kaufmann, 2004.

[3] H. Lamehamedi, Z. Shentu, B. Szymanski and E. Deelman, "Simulation of Dynamic Data Replication Strategies in Data Grids," in Proceeding of the International Parallel and Distributed Processing Symposium, Apr. 2003.

[4] The EU DataGrid Project: *http://eu-datagrid.web.cern.ch/eu-datagrid/*

[5] I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit," *International Journal of Supercomputer* Applications, Vol. 11, No. 2, pp. 115-128, 1997.

[6] 강경우, 박형우, "그리드 연구개발 동향," 한국정보과학회지, 제 20권, 제 2호, 23-34쪽, 2002년 2월

[7] I. Foster and C. Kesselman(eds.) "*The Grid: Blueprint for a new Computing Infrastructure*," Morgan Kaufmann Publishers, 1998.

[8] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, C.Kesselman, S.Meder, V.Nefedova, D. Quesnel and S. Tuecke, "Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing," in Proceeding of the IEEE Mass Storage Conference, 2001.

[9] I. Foster, "The Grid: A New Infrastructure for 21st Century Science," Physics Today, Vol. 54, No. 2, 2002.

[10] R. Stevens, P. Woodward, T. DeFanti, and C.

Catlett, "From the I-WAY to the National Technology Grid," Communications of the ACM, 40(11): 50-61. 1997.
[11] Kavitha Ranganathan and Ian Foster, "Identifying Dynamic Replication Strategies for a High-Performance Data Grid," in Proceeding of International Workshop on Grid Computing, Denver, Nov. 2002.
[12] David G. Cameron, Ruben Carvajal-Schiaffino, A. Paul Millar, Caitriana Nicholson, Kurt Stockinger, and Floriano Zini, "OptorSim c1.0 Installation and User Guide," Feb. 2004.
[13] OptorSim-A Replica Optimizer Simulation: *http://edg-wp2.web.cern.ch/edg-wp2/optimization/optorsim.html.*
[14] M. Tang, Bu-Sung Lee, Chai-Kiat Yeo, and X. Tang, "Dynamic replication algorithms for the multi-tier Data Grid," Future Generation Computer Systems, Vol. 21, Issue 5, pp. 775-790, May 2005.

저 자 소 개

오 상 원(정회원)
2002년 한양대학교 전자컴퓨터공학부 학사.
2005년 한양대학교 컴퓨터공학과 석사.
2005년 3월~현재 (주)에어코드 DTV 연구소 전임연구원.
<주관심분야 : 병렬처리시스템, Grid 컴퓨팅>

이 원 주(정회원)
1989년 한양대학교 전자계산학과 학사.
1991년 한양대학교 전자계산학과 석사.
2004년 한양대학교 컴퓨터공학과 박사.
1999년~2008년 2월 두원공과대학 모바일인터넷과 부교수.
2008년 3월 현재 인하공업전문대학 컴퓨터정보공학부 부교수.
<주관심분야 : 병렬처리시스템, 인터넷 및 모바일 컴퓨팅, Grid 컴퓨팅, 센서네트워크>

전 창 호(평생회원)
1977년 한양대학교 전자공학과 학사.
1982년 Cornell University, 컴퓨터공학과 석사.
1986년 Cornell University, 컴퓨터공학과 박사.
1977년~1979년 전자통신연구소 연구원.
1989년~현재 한양대학교 전자컴퓨터공학부 교수.
<주관심분야 : 병렬처리시스템, 성능분석, Grid 컴퓨팅, 센서네트워크>