데이터 그리드를 위한 Peer-to-Peer 기반 복제 정책

오상원°, 이원주*, 전창호**
한양대학교 전자컴퓨터공학부°, 두원공과대학 인터넷프로그래밍과*, 한양대학교 전자컴퓨터공학부**
swoh@cse.hanyang.ac.kr°, wonjoo@doowon.ac.kr*, chjeon@cse.hanyang.ac.kr**

A Peer-to-Peer based Replication Strategy for Data Grid

Sangwon Oh°, Wonjoo Lee*, Changho Jeon**
School of Electrical and Computer Engineering, Hanyang University°,
Department of Internet Programming, Doowon Technical College*,
School of Electrical and Computer Engineering, Hanyang University**

요 약

데이터 그리드를 위한 기존의 복제 정책은 계층적 구조를 기반으로 하고 있기 때문에 상위계층에서 하위계층으로만 데이터를 복제할 수 있어 비효율적이다. 따라서 본 논문에서는 기존 데이터 그리드의 계층적 구조에 P2P(Peer-to-Peer) 시스템을 적용하여 효율적으로 복제본을 유지할 수 있는 새로운 복제 정책을 제안한다. 이 정책의 특징은 클라이언트 노드의 저장 공간 일정 부분을 임계 구역(critical section)으로 지정하여 데이터 조작을 방지함으로써 클라이언트 노드도 데이터 복제본을 저장할 수 있는 기능을 가지도록 한 것이다. 따라서 계층적 구조에서 상위 계층뿐만 아니라 동일 계층 또는 클라이언트 노드들 간의 데이터 전송이 가능하기 때문에 데이터 그리드의 성능을 향상시킬 수 있다.

1. 서론

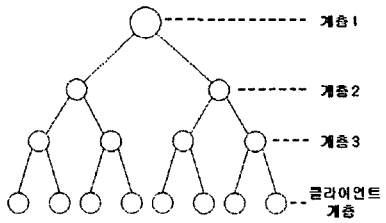
데이터 그리드는 지리적으로 분산되어있는 컴퓨터와 저장 자원을 연결하여 데이터와 자원들을 공유한다[1]. 이러한 환경을 이용하면 전 세계에 흩어져 있는 여러 대학 및 연구 기관들의 학자들간에 상호 협력이 가능해지기 때문에 연구 및 데이터 분석에 대한 효율성이 높아진다[1, 2]. 일반적인 컴퓨터 구조에서 하드 디스크와 CPU 사이에 메인 메모리와 캐시 메모리를 두어 데이터의 접근시간을 줄이 듯이 데이터 그리드 역시 메인 데이터 서버와 클라이언트 노드들 사이에 중간 노드들을 두어 데이터 관리의 효율성을 높여왔다. 하지만 상위/하위가 분명한 계층형 연결구조이기 때문에 상위계층에서 하위계층으로만 데이터 복제가 가능하다는 문제점이 있다.

본 논문에서는 기존의 복제 정책에 P2P 개념을 추가하여 복제 비용을 최소화 할 수 있는 복제 정책을 제시한다. 이 정책은 P2P를 이용한 데이터 분산의 효율성을 고려하여 데이터의 복제 비용이 최소인 노드를 찾아 데이터를 복제함으로써 전체적인 데이터 그리드의 성능을 향상시킨다.

본 논문의 구성은 다음과 같다. 2장에서는 데이터 그리드의 연결 구조와 기존 복제 정책에 대하여 설명한다. 3장에서는 제안하는 복제 정책에 대하여 자세히 설명한다. 그리고 4장에서 결론을 맺는다.

2. 관련 연구

기존의 데이터 그리드 복제 정책을 시뮬레이션하기 위해 사용하는 데이터 그리드의 연결 구조는 <그림 1>과 같다.



<그림 1> 데이터 그리드의 연결 구조

<그림 1>에서 계층 1은 메인 데이터 서버로 구성한다. 메인 데이터 서버는 데이터 그리드에 존재하는 모든 데이터를 생성하는 기능을 한다. 다른 계층의 노드에서는 데이터의 생성 및 수정이 불가능하다. 계층 2와, 계층 3은 데이터 저장 기능을 하는 중간 노드들로 구성된다. 이 계층에서는 효율적인 데이터 복제 정책을 사용하여 복제 데이터를 클라이언트에 근접한 서버에 저장하는 것이 중요하다. 클라이언트 계층은 그리드 계층구조의 최 하위 수준에 위치하며, 계층 1에서 생성된 데이터나 계층 2, 3에서 유지되고 있는 복제 데이터에 대하여 필요한 데이터 전송을 요청한다. 이러한 계층구조에서는 상위 계층일수록 저장장치의 용량과 계층간의 연결 대역폭이 증가한다.

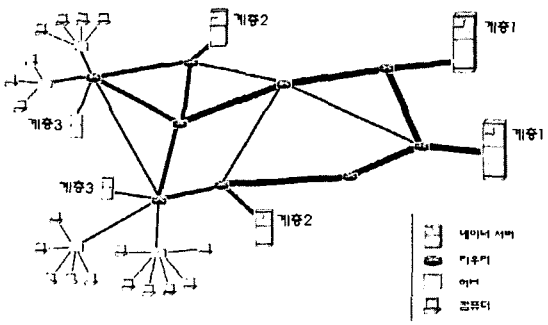
데이터 그리드의 데이터 요청은 시간 집약성(temporal locality), 지리 집약성(geographical locality), 공간 집약성(spatial locality)으로 분류된다[3]. 시간 집약성 데이터 요청은 최근에 접근된 파일이 다시 접근 될 확률이 높다는 것이다. 지리 집약성 데이터 요청은 지리적으로 근접한 클라이언트에 의해 다시 접근 될 확률이 높다는 것이고, 공간 집약성 데이터 요청은 최근에 접근

된 파일의 주변에 있는 파일들이 다시 접근 될 확률이 높다는 것이다. 기존의 데이터 그리드에서 복제 정책은 아래의 <표 1>과 같이 6가지로 분류할 수 있다[3].

<표 1> 기존의 데이터 그리드 복제 정책

정책	특징
무 복제 (No Replication)	데이터를 계층1의 메인 데이터 서버로부터 필요할 때 마다 해당 클라이언트에 복제
최적 클라이언트 (Best Client)	각 노드들은 각 파일에 대한 히스토리인 요청 횟수와 요청자 정보 보관 그 히스토리가 임계값(threshold)을 초과하면 그 파일에 대한 요청이 가장 많았던 노드를 최적 클라이언트로 선정해 해당 파일 복제
연속형 복제 (Cascading Replication)	데이터의 흐름이 계층1에서 하위계층으로 차례로 내려가는 모습에 기인하여 이름 붙여진 정책으로 상위의 중간 노드의 저장 공간이 다 차면 하위의 노드에 파일 복제
보통 캐싱 (Plain Caching)	파일을 요청한 클라이언트가 자신에게 그 파일을 저장할 충분한 공간이 있다면 실행하는 동안 저장하여 사용하고 다른 파일이 필요하다면 대체
캐싱이 추가된 연속형 복제 (Caching plus Cascading Replication)	위의 연속형 복제와 보통 캐싱 정책들을 혼합한 것으로 클라이언트는 필요한 파일을 내부에 캐싱하고, 서버는 정기적으로 인기있는(popular) 파일을 하위 구조로 전달하여 복제
고속 확산 (Fast Spread)	파일을 요청한 클라이언트에 복제를 할 때 그 파일이 지나는 경로에 있는 각 계층의 노드들에 파일 복제

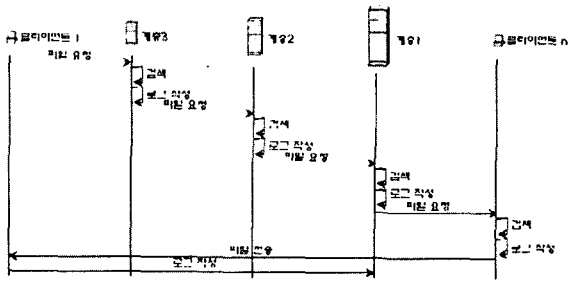
<표 1> 기존의 데이터 그리드 복제정책에서 복제 데이터의 흐름은 상위 계층에서 하위 계층으로만 이루어진다. 즉, 클라이언트 계층의 말단 노드에 데이터가 존재하지만 상위 계층에 데이터 전송을 요청함으로써 복제 비용이 증가하는 문제점이 있다.



<그림 2> 네트워크 연결 구조

3. 제안하는 복제 정책

본 논문에서는 기존 데이터 그리드 복제 정책의 문제점을 줄이기 위해 P2P 방식을 이용한 새로운 복제 정책을 제안한다. 이 복제 정책은 클라이언트의 저장장치 일정 부분을 임계구역(critical section)으로 지정하여 파일 조작을 방지함으로써 클라이언트도 데이터를 저장할 수 있는 기능을 갖도록 한다. 또한 데이터 그리드의 연결 구조에 <그림 2>와 같은 네트워크 연결 구조를 반영함으로써 P2P 연결 방식을 적용할 수 있다. 따라서 상위 계층뿐만 아니라 동일 계층 또는 클라이언트 사이의 데이터 전송이 가능하다.



<그림 3> P2P를 이용한 파일 전송 예

<그림 3>을 예로 들어 제안하는 복제 정책을 설명한다. 각 노드는 데이터에 대한 요청 수와 요청지 정보에 해당하는 로그 정보를 기록한다. 이 로그 정보는 해당 데이터를 가장 빠르게 복제할 수 있는 노드를 검색할 때 사용한다. 이 검색된 노드는 상위 계층의 노드 뿐만 아니라 클라이언트도 포함된다. 따라서 다른 클라이언트 혹은 다른 계층의 노드들로부터 데이터 요청을 받으면 데이터를 전송한다. 클라이언트로부터 데이터 요청을 받았을 경우 데이터 요청 알고리즘은 <그림 4>와 같다.

```
int Client_requestFile (string stieName, string fileName)
{
  fileLocation = whereIs (fileName);
  if (fileLocation == local)        // 파일이 로컬에 있는 경우
  {
    그 파일 사용
  }
  else if (fileLocation == upper) // 파일이 상위 계층에 있을 경우
  {
    status = requestFileToUpper (siteName, fileName);// 파일 요청
  }
  else                              // 파일이 하위 계층에 있는 경우
  {
    해당 클라이언트에 P2P를 이용한 파일전송 요청
  }

  if (Lack of Storage Capacity)     // 저장 공간이 부족할 경우
  {
    dFileName = deleteFile();       // 삭제할 파일 검색 및 삭제
    requestUpdateLog (siteName, dFileName);// 기록 수정 요청
  }
  updateStorage (fileName);                 // 파일 복사
  requestWriteLog (siteName, fileName); // 기록 수정 요청
}
```

<그림 4> 클라이언트의 데이터 요청 알고리즘

<그림 4>에서는 먼저 파일의 위치를 검색한다. 그 결과 파일

이 클라이언트에 존재하면 복제하지 않고, 존재하지 않을 경우에는 상위 계층에 요청한다. 만약 데이터를 복제할 경우에는 저장 장치의 여유 공간을 확인한다. 여유 공간이 존재하면 데이터를 복제하지만 부족할 경우에는 삭제 정책에 따라 데이터를 삭제한 후 복제한다. 최하위 노드의 데이터는 P2P 연결을 이용하여 다른 노드로 전송한다.

<그림 3>을 예로 들어 클라이언트의 데이터 요청 알고리즘을 자세히 설명한다. 클라이언트 1에서 요청하는 데이터와 로그는 계층 1의 메인 데이터 서버와 클라이언트 n에만 존재한다. 그리고 클라이언트 1과 클라이언트 n은 다른 그룹에 위치한다고 가정한다. 클라이언트 1은 상위 계층인 계층 3의 데이터 복제 서버에 필요한 데이터를 요청한다. 이때 계층 3의 데이터 복제 서버는 데이터의 존재여부를 검색하고 로그를 작성한다. 검색 결과 데이터가 계층 3의 그룹에 존재하지 않으면 계층 2의 데이터 복제 서버에 데이터를 요청한다. 계층 2의 데이터 복제 서버에도 데이터가 존재하지 않으면 계층 1의 메인 데이터 서버에 요청한다. 이때 계층 1의 메인 데이터 서버와 복제본을 가진 다른 클라이언트 중에 데이터 전송시간을 최소화할 수 있는 것을 찾는다. 만약 클라이언트 n에서 데이터 전송시간이 최소일 경우 데이터 서버는 로그를 작성하고 클라이언트 1로 데이터를 전송하도록 클라이언트 n에 요청한다. 이때 클라이언트 n과 클라이언트 1사이에는 P2P 연결이 생성되어 데이터를 전송한다.

데이터를 생성하고, 데이터 요청을 처리하는 메인 데이터 서버 관리 알고리즘은 <그림 5>와 같다.

```
void createFile (string fileName)          // 파일 생성
{
  placeFile (fileName);                    // 파일 배치
  writeLog (fileName);                     // 파일에 대한 로그 작성
}

int requestFileToUpper (string stieName, string fileName)
{
  fileLocation = whereIs (fileName);   // 파일 위치 검색
  writeLog (fileName);                 // 파일에 대한 로그 작성
  if (fileLocation == local)           // 파일이 로컬에 존재할 경우
  {
    해당 클라이언트에 파일 전송
  }
  else if (fileLocation == lower)    // 파일이 하위 계층에 존재할 경우
  {
    해당 클라이언트에 P2P를 이용한 파일 전송 요청
    그 파일이 삭제된 경우
      requestFileToUpper (stieName, fileName);  // 파일 요청
  }
}
```

<그림 5> 메인 데이터 서버 관리 알고리즘

<그림 5> createFile() 모듈에서는 메인 데이터 서버에 존재하지 않는 데이터를 생성하여 저장 장치에 배치한다. 그리고 그 데이터에 대한 로그를 작성한다. 또한 메인 데이터 서버는 requestFileToUpper() 모듈을 이용하여 클라이언트의 데이터 요청을 처리한다. 그 절차는 먼저 데이터의 위치를 확인하고 로그를 작성한다. 메인 데이터 서버에서 데이터를 전송하는 것이 전송시간을 최소화 한다면 클라이언트는 메인 데이터 서버로부터 데이터를 받는다. 하지만 반대의 경우에는 P2P를 이용하여 다른 계층의 클라이언트 또는 데이터 복제 서버로부터 데이터를 전송하도록 한다.

4. 결론

본 논문에서는 기존의 데이터 그리드의 연결 구조에 P2P 시스템을 적용한 새로운 복제 정책을 제안하였다. 이 복제 정책의 특징은 필요한 데이터 복제 비용을 최소화 할 수 있는 노드를 찾아 해당 데이터를 복제한다. 즉, 클라이언트 노드의 저장 공간 일정 부분을 임계 구역(critical section)으로 지정하여 데이터 조작을 방지함으로써 클라이언트 노드도 데이터 복제본을 저장할 수 있는 기능을 가지도록 한 것이다. 따라서 계층적 구조에서 상위 계층뿐만 아니라 동일 계층 또는 클라이언트 노드들 간의 데이터 전송이 가능하기 때문에 데이터 복제시간을 줄일 수 있다.

향후 연구는 시뮬레이션을 통하여 제안한 복제 정책을 검증하는 것이다.

5. 참고 문헌

1)I. Foster, C. Kesselman (eds.), "The Grid: Blueprint for a New Computing Infrastructure", Morgan Kaufmann, 1999.

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

3)Kavitha Ranganathan, Ian Foster, "Identifying Dynamic Replication Strategies for a High-Performance Data Grid", Proceeding of International Workshop on Grid Computing, Denver, Nov. 2002.

4)S. Vazhkudai, J. Schopf, I. Foster, "Predicting the Performance of Wide-Data Transfers", 16th Int'l Parallel and Distributed Processing Symposium (IPDPS 2002), Fort Lauderdale, Florida, Apr. 2002.