제 19 회 한국정보처리학회 춘계학술발표대회 논문집 제 10 권 제 1 호 (2003. 5)

대규모 그리드 환경에서 Chord 을 이용한 신뢰성 있는 자원 탐색방법에 관한 연구

정인철*, 함호상*
*한국전자통신연구원, 이동분산처리연구팀
e-mail : jic@etri.re.kr, hsham@etri.re.kr

Serving Reliable Resource Discovery in Grid Environment using a Chord

InCheol Jeong*, Ho-Sang Ham*
*Mobile Distributed Computing Research Team, ETRI

요 약

그리드와 같은 대규모 컴퓨팅 자원이 연결되어 이용되는 분산처리 환경에서 가장 중요한 기능 중에 하나가 자원 탐색(lookup) 기능이다. 찾기 기능은 미리 정해진 형태로 자원을 등록하고 이 정보를 중앙에서 관리하는 단일기관이나 소규모로 분산된 컴퓨팅 자원을 공유하는 환경에서는 문제가 되지 않는다. 그러나 임의로 가입과 탈퇴가 가능하고 대규모로 자원이 연결된 그리드 환경에서는 빠른 자원 탐색이 성능의 기본 요소가 된다. 이 논문에서 그리드 환경에서 자원탐색 방법으로 분산형 자원탐색 방법인 Chord 시스템을 이용하고 자원탐색방법을 신뢰성 있게 유지하기 위해 메시지 큐를 이용하여 시스템을 모델링하고 시스템을 설계한 내용을 기술하였다. 이를 위하여 Chord 시스템에 Message Queue System 을 통합하여 응용 프로그램에서 요청한 자원탐색을 메시지 큐에 저장하여 노드가 failure 되거나 네트워크가 분할이 되더라도 메시지 큐에 저장된 내용을 바탕으로 일관성을 유지할 수 있는 방법을 제시하였다.

1. 서론

컴퓨터 자원과 기술이 발전함에 따라서 정보처리방법이 컴퓨팅 자원을 다양하게 이용할 수 있는 분산처리 기술이 필요하게 되었다. 급격한 기술 발전에 따라서 그리드 환경이 분산처리 환경에서 가장 중요한 기술 중에 하나가 되었다. 그리드라는 용어는 1990 년 중반에 과학과 엔지니어링 분야에서 분산처리 구조로 제안되었다. 기술의 발전에 따라서 그러한 인프라 구조를 만드는 것을 발전시켜 왔으나 아직까지 용어가 혼재한 상태[8]이다. 그리드라는 용어는 발전된 형태의 네트워킹 기술이라는 용어에서부터 인공지능에 이르기 까지 개념이 혼재한다. 그리드 기술은 다양한 개인과 단체, 자원을 가변적으로 공유하여 유연하고 보안성이 높고 조정이 가능한 기술이라고 정의할 수 있다. 따라서 그리드 환경에서는 대규모 컴퓨팅자원이 연결되어 있는 그리드에서 가장 중요한 기능 중에 하나가 자원탐색 방법이다.

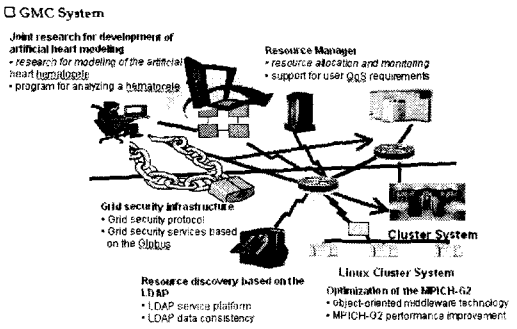

그림 1 : 그리드 시스템

자원 탐색방법은 미리 정해진 형태대로 관련 정보를 등록하는 이 정보를 중앙 집중식으로 관리하는 단일 기관이나 소규모로 분산된 컴퓨팅자원을 공유하는

환경에서는 그다지 문제가 되질 않는다. 그러나 임의로 가입과 탈퇴가 가능하고 대규모의 자원이 연결되어 공유되는 단대 단 그리드 환경에서는 빠른 찾기 기능이 그리드 성능의 기본요소가 된다. 자원을 탐색하는 것은 두 가지 단계를 가진다. 첫번째는 자원을 탐색하는 과정이고 두 번째는 자원과 연결하는 과정이다. 그러나 자원탐색 서버가 대규모의 peer-to-peer 환경에서 중앙 집중식으로 관리된다면 자원탐색은 큰 문제가 아니다. 그런데 전통적인 자원 탐색방법(LDAP 서버)은 중앙 집중식 DBMS 방법에 기반하고 있어 모든 정보(Name, Location, Attribute)가 한 곳에서 유지되고 관리된다. 중앙 집중식 방법에서는 자원탐색시간은 일정하다. 그러나 같은 자원 탐색방법이라고 멤버가 자유롭게 등록하고 삭제하는 조정이 가변적인 환경에서는 자원탐색 시간이 일정하지 않다[7].

Chord 시스템은 peer-to-peer 응용을 위한 분산처리 자원탐색 프로토콜이다. Chord 시스템은 하나의 오퍼레이션만을 지원한다. 키가 주어지면 키를 이용하여 노드와 매핑시킨다. 데이터의 위치는 키와 관련된 데이터를 연결 함으로서 쉽게 구현할 수 있다. 키와 데이터 항목은 노드에 연결된다. Chord 시스템은 시스템을 자유롭게 join 하거나 leave 하는 노드에서 효율적이며 시스템이 지속적으로 변경하는 과정에서도 query를 수행할 수 있다.

대규모 사용자기 연결된 그리드 환경에서 네트워크 분할이나 대규모 노드의 동시 가입과 탈퇴를 효율적으로 대처하기 위해서는 노드의 동시 가입과 탈퇴하는 과정이 원자적(Atomic Operation)으로 수행되어야 한다. 특히 가입과 탈퇴 그리고 네트워크 분할이 자주 발생하는 경우에는 더욱 그러하다. 네트워크가 분할되었다가 다시 회복되는 경우에 recovery 프로토콜을 동작시켜서 연결되는 네트워크가 어디에서 바이딩 되더라도 정확한 위치에 연결될 수 있어야 한다. 그러나 네트워크가 분할되는 어떤 경우도 해결할 수 있는 서비스를 보장하는 것은 어렵다. 따라서 "best-efforts" availability 방법으로 보장한다. 이 방법은 도달 가능한 replication 노드 중에서 적어도 하나의 노드를 접근할 수 있다는 전제 하에 보장하는 방법이다.

이 논문은 분산된 환경에서 Chord 시스템을 이용한 자원의 위치에 관한 정보의 integrity 를 유지할 수 있는 방법을 기술한다. 자원의 관한 정보가 한 곳에 모여 있지 않고 분산되어 있으며 분산되어있는 정보가 어디에 존재하는지를 알 수 없으며, 컴퓨팅 노드가 가입과 탈퇴를 임의로 하기에 자원의 위치에 관한 정보의 integrity 를 보장할 수 없기 때문이다.

2. 관련연구

자원 탐색방법에 관한 연구의 대상이 되는 시스템은 두 가지 종류가 있다 : 첫 번째는 가변적이고 자체적인 네트워크를 보유하고 있는 환경에서 자원탐색 방법, 두 번째는 wide-area 시스템에서 자원탐색 방법이다. 전자의 경우에는 최근의 P2P(Peer-to-Peer)의 인기에 따라서 집중이 되고 있는 파일 공유 시스템이다.

그런 시스템들은 이름을 이용하여 자원을 인식한다. 그리고 파일의 위치를 찾는 방식도 여러 가지가 존재한다. 예를 들면 aggressive flooding 방식을 채택한 Gnutella[5]시스템과 request forward 방식과 automatic 파일 replication 방법을 통합한 시스템인 Freenet[10] 방법과 효율적인 이름 기반 탐색을 채택하고 있는 CAN[13], Chord[4] 와 Tapestry[12] 시스템들이 존재한다. 이 시스템들은 데이터를 위치를 효율적으로 하기 위한 함축되고 신뢰성 있는 유연한 구조를 가지고 있다

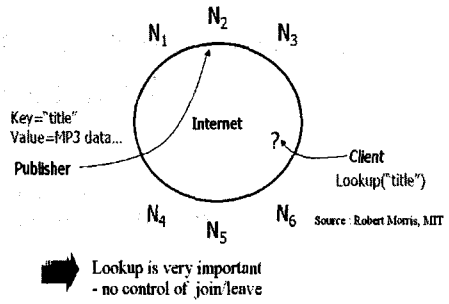

그림 2 : 일반적인 자원 탐색방법

Dynamism 과 scale 과 heterogeneity 문제로 인하여 name 을 기반으로 한 탐색방법은 computational GRID 에서는 실용적이지 못하다. 자원탐색방법 중에서 전역 이름을 이용하지 않는 분산 자원공유 시스템은 Condor 의 matchmaker 라는 시스템이다. 자원 정의와 요청이 matching 이 이루어지는 중앙 서버에서 이루어진다. 중앙 집중식 구조는 Condor 이 처음에 제안한 LAN(local area network)에서는 효율적이다. 그러나 중앙서버에서 동작하도록 조직이 구성되어 있는 경우에만 효율적이다.

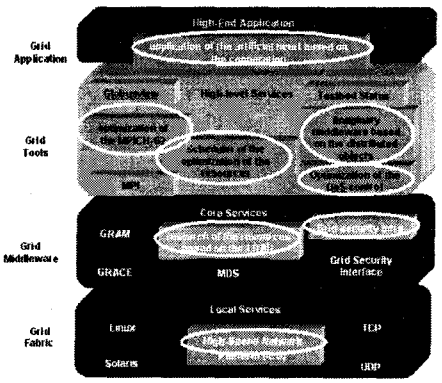

그림 3 : Globus 그리드 계층구조

Globus 는 다른 방법의 구조를 가지고 있는데 MDS(Meta Data Service) 라는 시스템이다. 초기에는 이

서버는 중앙 집중식 구조를 가지고 있었으나 이후에 자원과 사용자의 개수가 많아지자 분산형 구조로 변경하였다[7]. MDS-2 에서는 그리드는 registration 프로토콜을 통하여 인덱스 서버에 기록하는 다중 Information 구조를 가지고 있다. 인덱스 서버는 enquiry 프로토콜을 사용하여 디렉토리 서버에게 query 를 수행하고 Information source 로부터 상세한 정보를 얻는다.

현재까지 peer-to-peer 파일 공유 시스템은 scalability 가 가장 성취하기 어려운 시스템이다. Napster 의 경우에는 중앙 집중형 디렉토리의 구조를 가지지만 중앙의 서버의 single failure 에는 취약하다. Gnutella 시스템은 범위를 결정하는데 broadcast 방법을 이용한다. Freenet 은 replication 문서를 집합적으로 관리하지만 여러 번의 단계를 거치는 문서에 대한 검색이나 변경을 보장해 주지는 않는다. 그러나 Chord 시스템은 이러한 경우에 적합한 자원탐색 응용이다.

3. Chord 시스템 모델

Chord 시스템은 peer-to-peer 응용을 위한 분산처리 자원탐색 프로토콜로, 분산 lookup 서비스를 지원하며 key 를 이용하여 삽입, 탐색, 삭제를 수행한다. Chord 시스템의 key 는 bytes array 형태로 유지되며 유니크한 키를 유지하기 위해서 random m-bit key identifier 를 이용한다.

Function	Description
Insert(key, value)	Inserts a key / value binding at r distinct nodes. Under stable conditions, exactly r nodes contain the key/value binding
Lookup(key)	Returns the value associated with the key.
Update(key, value)	Inserts the key / new Val binding at r nodes Under stable conditions, exactly r nodes contain key / new Val binding.
Join(n)	Causes a node to add itself as a server to the Chord system that node n is part of. Returns success or failure.
Leave()	Leave the Chord system. No Return value.

표 1 : Chord 시스템의 API

Chord 시스템의 API 는 다음과 같은 구성요소를 가진다[4]. insert(key, value)가 호출되면 Chord 시스템은 key/value 를 선택된 노드에 삽입한다. Lookup(key)가 호출되면 Chord 시스템은 시스템의 여러 노드 중에서 가장 알맞은 key/value 를 찾는다. Chord 시스템은 key / value 항목을 변경하는데 originator 에 의해서만 허용한다. Chord 시스템은 delete 오퍼레이션을 가지고 있지 않는데 update(key, value)를 이용하여 구현 가능하다. 또 다른 두 가지 오퍼레이션은 Chord 시스템에 연결하거나 탈퇴하는 명령어이다.

Chord 시스템의 서비스 모델은 "best-effort" 방법을 이용한다. Key 를 저장하고 있는 Chord 네트워크에서 적어도 하나의 노드만이 존재한다면 key / value 는 영속적이다. 만일 Chord 시스템이 어떤 이유로 분할이 된다면 서버는 분할이 된 상태에서 재 조직화하여 분리된 서버 안에서 재조작을 위한 통신을 수행하고 각 바인딩에 저장된 노드 끼리 만 유지한다. 그러나 분할이 고쳐지면 stabilization protocol 을 이용하여 연결된 분할 들끼리 어떤 binding 을 통하여 정확한 위치를 유지하게 한다. 따라서 이를 일관성을 유지하기 위해 tight bound 를 유지하지 않는다. Tight bound 를 유지하기 위해서는 key/value binding 에 일관성을 유지해야 한다. 따라서 노드를 삽입하고 변경하는 과정은 원자성을 보장하지 않는다.

4. 신뢰성 있는 자원탐색

Chord 시스템에서 신뢰성문제는 하나의 COPY 만이 serial 하게 접근할 수 있는 것을 의미한다. 그러나 이 시스템은 효율적인 자원탐색과 join 과 leave 시에 자원 탐색시간이 짧은 것이 특징이다. 또한 join/leave/failure 시에 Lazy update 를 수행하는 것이 특징이다. 그러므로 자원을 leave 하는 과정에서 Successor list 의 다음이나 successor list 이전에서 네트워크가 단절이 된다면 Atomicity 를 만족하지 못하게 되는 문제점이 발생된다.

따라서 실행 중에도 일관성이 유지되어야 한다. 그러나 chord 시스템의 경우에 네트워크가 분할되는 경우와 대규모 사용자가 동시에 접근하는 경우와 Chord 시스템이 비동기적으로 수행할 경우가 생기기 때문에 일관성이 깨지기 때문에 신뢰성 보장 자원탐색 방법이 필요하게 되었다. 이러한 원자성은 여러 사용자가 동시에 쓰기를 수행하는 시스템의 경우나 agreement 나 totally ordered broadcast 방법을 이용하는 서비스인 경우에는 더욱 중요한 요소이다.

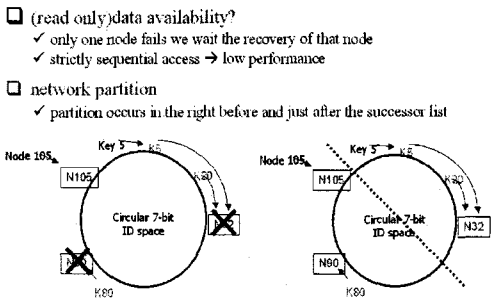

그림 4. Chord 시스템 failure

Chord 시스템의 경우에 노드가 fail 되면 n 을 포함하는 테이블을 가진 노드는 n 의 success 를 찾는다. 그리고 n 의 successor 를 찾으면 key/value 의 복사본을 n 에 저장한다. 그런데 만일 failure 가 발생하면 진행중인 query 오퍼레이션이 동작하지 않게 되고 re-stabilize 을 수행해야 한다. 만일 노드가 fail 되는 경우에 이웃

노드들과의 stabilize 과정이 응답이 없을 수 있다. 이 때 인접노드에게 즉각적으로 노드가 죽었다는 사실을 통보한다. 이런 경우와 네트워크가 분할되는 경우에 실행을 자체적으로 함에 따른 일관성 문제가 발생한다.

4.1 제안 시스템 모델

이런 문제를 해결하기 위해 노드와 노드사이에 전달되는 통신을 위한 메시지 큐를 통합한다. 이를 통하여 원자성을 보장한다.

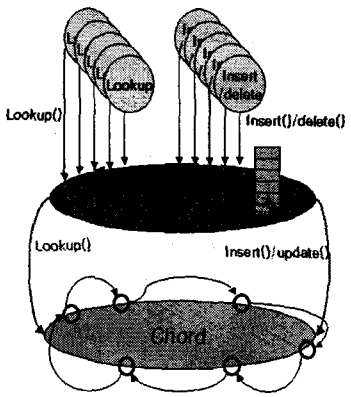

그림 5. 메시지 큐를 이용한 신뢰성 있는 자원탐색

각 노드는 자신의 상태(status)를 유지한다. (joining, active, leaving) 각 노드는 MQS 의 Incoming, Outgoing Queue 에 삽입된 테이블과 대응되는 테이블을 유지한다. 노드가 MQS 의 Incoming Queue 에 삽입을 요청하면 그 내용을 자신에게 미리 복사한다. 삽입과 삭제 요청을 수행하면 MQS 의 Outgoing Queue 에 삽입한다. 따라서 동시에 접근하는 클라이언트의 메시지 Order 문제를 해결할 수 있다.

자원 탐색인 경우 성능상의 문제로 메시지 큐에 탐색하려는 노드를 저장하는데 persistent 모드로 저장하지 않기 때문에 failure 가 발생하면 내용을 잃어버릴 수 있다. 자원 삽입 및 삭제(insert, delete)인 경우 메시지의 큐에 persistent 모드로 send(…, PERSISTENT ,…) 을 요청하여 메시지의 persistent 를 보장한다. 시스템이 failure 이 되면 MQS 의 persistent store 에 저장된 내용을 바탕으로 retry 를 수행하여 일관성을 유지한다. 노드를 시스템에 join 하거나 leave 하는 경우 메시지 큐에 해당 노드를 저장한다.

5. 결론

현재까지는 Chord 의 신뢰성을 유지하기 위한 방법으로 메시지 큐를 통합하는 방법을 제시하였다. 이를 이용하여 그리드 환경(Globus)에 이 시스템을 통합시켜서 제대로 동작하는 지를 검증할 예정이다. 대규모 사용자 환경에서 동시에 접근하는 사용자 시험을 위하여 노드의 개수가 증가함에 일관성이 보장되는지를 시험할 예정이다.

참고문헌

[1] CLARKE, I., SANDBERG, O., WILEY, B., and HONG, T. Freenet : A distributed anonymous information storage and retrieval systems. *In Workshop on Design Issues in* Anonymity and Un-observability(2000)

[2] CZAJKOWSKI, K., FITZGERAND, S., FOSTER, I., and KESSELMAN, C. Grid information services for distributed resource sharing. In 10th IEEE Symposium on High Performance Distributed Computing(2001)

[3] IAMNITCHI, A., AND FOSTER, I. On fully decentralized resource discovery in grid environments. In International Workshop on Grid Computing (Denver, Colorado, November 2001), IEEE

[4] STOICA, I., MORRIS, R., KARGER, D,, KAASHOEK, M. AND BALSKRISHNAN, H. Chord : A Scalable peer-to-peer lookup service for internet applications. In ACM SIGCOMM (2001)

[5] Gnutella protocol specification, http://www.clip2.com/articles.html

[6] Nancy Lynch, Dahlia Malkhi, David Ratajczak , Atomic Data Access in Distributed Hash Tables. Proceedings of the International Peer-to-Peer Symposium, March 2002.

[7] Adriana Iamnitchi, Ian Foster, Daniel C. Nurmi Peer-to-Peer Approach to Resource Discovery in Grid Environment. In High Performance Distributed Computing (Edinbourgh, UK, July 2002)

[8] Ian Foster, Carl kessenlman, Steven Tuecke, The Anatomy of the Grid. In International Journal of Supercomputer Applications, 2001

[9] Napster, http://www.napster.com

[10] Clarke I. , Sandberg O. , Wiley B., and Hong T. W. Freenet : A distributed anonymous information storage and retrieval system. In Proceedings of the ICSI Workshop on Design Issues in Anonymous and Unobservability (Berkeley, California, June 2000). http://freenet.sourceforge.net.

[11] Ohara, Smart decentralized peer-to-peer sharing. http://www.ohara.com/design.html

[12] Zhao, B., Kubiatowicz, J., and Joseph, A., Tapestry : An infrastructure for fault-resilient wide-area location and routing. Tech. Rep. UCB//CSD-010114, U.C. Berkeley, 2001

[13] Ratnasamy, S. , Francis, P., Handley, M. , Karp, R., and Shenker, S., A scalable content addressable network. In ACM SIGCOMM(2001)