# 그리드 데이터베이스에서 메쉬 연결 구조를 이용한 부하 분산

# (A Load Balancing Method Using Mesh Network Structure in the Grid Database)

이 순 조*
(Lee Soon Jo)

**요 약** 본 논문에서는 그리드 데이터베이스에서의 부하 분산 문제를 복제 데이터 간 메쉬 연결 구조를 이용하여 해결한다. 그리드 데이터베이스의 데이터는 성능 향상을 위해 여러 노드에 복제 저장되어 있다. 따라서 사용자 질의는 목적 데이터를 포함하는 노드들의 작업 부하를 평가하여 노드를 선택함으로써 구성 노드들 간의 부하를 분산하여야 한다. 기존의 기법은 노드의 작업 부하가 한계를 넘게 되었을 때 다른 연결 노드를 선택하여 질의를 처리하게 하는 수동적 부하 분산 기법을 사용하기 때문에 노드의 수가 많고 질의가 유동적인 그리드 데이터베이스에 적용하기에는 비효율적이다. 제안 기법은 각각의 동일 복제본이 포함된 노드들을 하나의 메쉬 구조로 연결하여 사용자 질의가 발생하였을 때 연결 노드 중 부하가 가장 적은 노드를 선택하여 질의 처리를 할 수 있도록 한다. 제안 기법은 성능 평가를 통해 기존의 기법보다 향상된 성능을 가짐을 보였다.

**핵심주제어** : 그리드 데이터베이스, 부하 분산, 메쉬 구조

**Abstract** In this paper, mesh network structure is applied to solve the load balancing problems in the Grid database. Data of the Grid database is replicated to several node for enhanced performance. Therefore, load balancing for user's query is selected node that evaluated workload in it. Existing researches are using passive load balancing method that selected another node after then node overflowed workload. It is inefficient to be applied to Gird database that has a number of node and user's queries almost changes dynamically. The proposed method connected each node which includes the same data through mesh network structure. When user's query occurs, it select node that has the lowest workload. The performance evaluation shows that proposed method performs better than the existing methods.

**Key Words** : Grid Database, Load Balancing, Mesh Network Structure

## 1. 서 론

그리드는 지리적으로 분산된 다양한 컴퓨팅 자원을 초고속 네트워크로 연결하여 고속 연산, 대용량 데이터 처리가 가능하도록 하는 기술이다. 이러한 그리드 환경에서의 데이터를 관리하기 위한 그리드 데이터베이스는 분산된 데이터의 효율적 처리와 사용을 할 수 있도록 한다. 기존의 분산 데이터베이스 시스템의 기

* 서원대학교 컴퓨터공학과

능을 기본적으로 포함하고 있으면서 대용량의 데이터 자원을 공유하고 고속의 트랜잭션 처리를 지원하는 그리드 데이터베이스는 구성하고 있는 각 노드의 데이터 처리 성능과 가용성 향상을 위해 같은 데이터를 서로 다른 위치에 복제 저장한다. 이러한 그리드 데이터베이스가 제공하는 대표적인 기능은 데이터의 연합, 변환, 배치 통합 기능이다[1,2].

그리드 환경에서 발생하는 사용자 질의는 보통 여러 노드에 분산되어 있는 데이터를 필요로 함에 따라 해당 데이터를 포함하는 노드로 질의가 전송되어 처리되어야 한다. 따라서 그리드 데이터베이스는 구성하는 각 노드에서 수행되는 어플리케이션에 따라 필요로 되는 수많은 데이터를 가용성과 성능의 향상을 위해 최적의 성능을 낼 수 있는 노드에 복제본을 두고 있고, 사용자 질의가 적합한 노드를 선택하기 위해 복제 데이터 정보를 갖고 있는 카탈로그 정보를 참조한다.

카탈로그 정보를 참조하여 질의를 노드에 분배할 때, 질의 처리의 최적화를 위해 노드와 노드 사이의 네트워크 비용, 해당 노드의 현재 작업 부하, 사용자 질의의 연산 처리 순서 등을 고려해야 한다. 하지만 그리드 데이터베이스 환경에서 모든 노드의 작업 부하 정보를 얻는 비용이 너무 크기 때문에 이를 개선하기 위한 기존 연구가 있지만 데이터 중심으로 부하 분산을 해야 하는 그리드 데이터베이스 환경에는 적합하지 않다.[3,4,5]

또한 그리드 데이터베이스 환경을 위해 제안된 링 기반 연결 구조 방식은 데이터 복제본이 있는 전체 노드가 아닌 인접 노드만을 고려하고 있어 인접 노드가 부하가 적을 경우에는 효율적이지만 그렇지 않을 경우 적합한 노드를 선택하는데 예측할 수 없는 시간이 소요된다는 것이 단점이다[6].

제안 기법은 데이터의 복제본이 포함된 노드들을 메쉬 구조로 연결하고 사용자 질의 발생 시 적합한 노드의 선택을 위해 복제본에 대한 전체 노드의 정보를 갖고 있는 노드 디스크립터를 이용하여 부하를 분산한다. 메쉬 구조는 그 특성상 복제본을 갖고 있는 모든 노드가 상대 노드에 대한 상태를 파악할 수 있는 구조이기 때문에 가장 적합한 노드를 선택할 수 있다는 장점이 있다.

그리드환경에서 구성하고 있는 모든 노드를 메쉬 구조로 연결하여 노드의 상태를 다른 모든 노드에게 전달한다는 것은 네트워크 트래픽 문제와 노드 자체에 상당한 부하를 주어 무리가 있지만 그리드 데이터베이스 환경에서 모든 노드가 아닌 복제본을 갖고 있는 노드 사이만 메쉬 구조로 연결하는 것은 수용할 수 있다고 본다. 또한 기존의 기법처럼 동일 복제본 그룹의 노드에 대한 상태정보를 주기적으로 점검하여 관리하는 것이 아니라 과부하 노드에서 질의 발생 시 동일 복제본 그룹에 상태 정보를 요청하여 가장 적합한 노드를 가장 최신의 상태 정보를 통해서 결정할 수 있도록 하였다. 이로 인해 상태정보 관리의 부담을 제거하였을 뿐만 아니라 최신의 상태정보를 이용할 수 있도록 하였다.

제안 기법에 대한 성능 평가를 통해 메쉬 구조로 연결하였을 때의 성능이 다른 기존 기법보다 우수함을 보여준다.

본 논문의 구성은 다음과 같다. 2장은 관련 연구로 그리드 데이터베이스와 기본의 부하 분산 기법에 대해 기술하고, 3장에서는 본 논문의 제안 기법인 메쉬 연결 구조를 통한 부하 분산 기법을 설명한다. 4장에서는 제안 기법의 성능 평가를 하고 5장에서 결론을 기술한다.

## 2. 관련연구

본 장에서는 그리드 데이터베이스의 특징 및 기능에 대해 설명하고 기존의 그리드 데이터베이스 시스템에서의 부하 분산 기법을 기술한다.

### 2.1 그리드 데이터베이스

인터넷이 보편화되고 컴퓨터 성능 및 네트워크의 전송 속도가 향상됨에 따라서 분산되어 있는 컴퓨터를 이용한 통합된 작업처리가 이루어 지고 있다. 예로써 암호 해독, 유전자 정보 처리, 비행체나 발사체 설계, 기후와 환경변화 분석, 고 에너지 물리분야의 데이터 분석 등에 이용되고 있다. 이러한 방대한 데이터에 대한 처리 능력 향상을 위한 노력의 일환으로 분

산 데이터베이스와 데이터베이스 클러스터에 대한 연구가 이루어졌고, 이제 또 다른 시도로써 자원 통합에 있어서 동일 기종 데이터베이스뿐만 아니라 이기종 데이터베이스들과 대용량 데이터의 통합을 위해 그리드 데이터베이스가 연구되어지고 있다.

분산 데이터베이스는 컴퓨터 통신망을 이용하여 여러 개의 지역 데이터베이스를 논리적으로 연관시킨 통합된 데이터베이스이다. 물리적으로는 분산되고 논리적으로는 집중되어 있는 형태의 구성으로 단순한 연결이 아닌 각 데이터베이스가 서로 관여를 하는 연결구조이다. 분산 데이터베이스의 장점은 데이터를 분산 배치하므로 장애에 대한 대비가 강하고 다수의 이용자가 대규모의 데이터베이스를 낮은 비용으로 공유할 수 있는 점이다. 분산 데이터베이스는 중앙 집중형 데이터베이스보다 저비용으로 구성이 가능하며, 확장성 및 가용성에 장점이 있다[7].

분산 데이터베이스가 데이터의 공유, 유통 및 투명성에 초점을 맞추어 있다면, 데이터베이스 클러스터는 고속의 네트워크를 연결하여 데이터 처리의 성능, 신뢰성, 가용성을 향상 시키는 목적이 있다. 데이터베이스 클러스터에서는 작업 부하를 분산시키기 위하여 부하 분산 기법을 사용하여 데이터베이스의 작업량을 균형 있게 분배한다.

그리드 데이터베이스는 그리드 컴퓨팅 환경에서 분산된 데이터의 효율적 처리와 사용을 위한 데이터베이스 관리 시스템이다. 그리드 데이터베이스는 기존 분산 데이터베이스 시스템의 기능을 기본적으로 포함하는 동시에 통합, 자동화 및 가상화 등의 기능을 제공한다. 그리드 데이터베이스의 주요한 기능으로는 대용량 자원에 대한 관리와 고속 연산 그리고 데이터 복제를 통한 가용성 향상에 있다. 그리드 데이터베이스를 구성하는 각 노드는 데이터를 처리 성능과 가용성 향상을 위해 서로 다른 위치에 복제하여 저장하며, 데이터 연합, 데이터 변환, 데이터 배치, 데이터 통합 등의 기능을 제공한다[8].

### 2.2 부하 분산 기법

부하 분산 기법은 일종의 스케줄링 문제에 속하며 기법들의 특징에 따라 구분하면 정적기법과 동적기법으로 나뉘어 진다. 정적 기법은 시스템이 진행되기 전에 미리 스케줄링을 완료하여 이후에는 각 노드의 부하에 관계없이 정해진 스케줄링 결과대로 운영하는 기법이며, 반대로 동적 기법은 매 순간마다 그 당시의 시스템의 상태에 따라 스케줄링을 수행하는 기법이다.

기존의 부하 분산 기법 중에서 가장 기본적이고 잘 알려져 있는 기법으로는 입찰기법으로서 질의를 처리시키고자 하는 노드에서 전 노드로 브로드캐스트 하여 응답을 한 노드 중 가장 적합한 노드에 질의를 전달한다. 이 기법은 간단하나 노드를 선택할 때마다 브로드캐스트를 해야 하는 오버헤드가 있으며, 이로 인한 시간 지연 때문에 노드의 상태가 변경되어 잘못된 결정을 내릴 수 있다[9].

그래디언트(gradient) 모델에 기반을 둔 Lin이 제안한 부하 분산 기법은 각 노드가 자기 자신과 바로 연결된 노드들과 정보를 교환한다. 여기서 노드는 저부하, 보통, 과부하의 세 가지 상태로 나뉘어 진다. 각 노드들은 저부하 상태 노드로부터의 최소 거리 값을 가지고, 저부하 노드는 0값을 가지고 인접 노드가 저부하 상태가 아니면 인접 노드의 거리 값은 1을 더한다. 그리고 자신의 값을 다른 인접 노드에게 전파하고 이 전파를 받은 노드는 자신이 저부하 상태가 아니면 그 값에 1을 더하고 인접 노드에게 전파한다. 저부하 상태면 자신의 값을 0으로 하고 모든 인접 노드에 전파한다. 이런 전파과정을 하면 자신의 거리 값은 저부하 상태의 노드로 부터의 최소 거리 값이 된다.

이 기법은 인접 노드들의 거리 값과 부하를 자신과 비교해야 하므로 부하가 변경될 때 인접 노드들의 상태를 메시지로 가져와야 하고 계산 결과를 인접 노드들에게 전파시켜야한다. 이러한 행동이 모든 노드들에서 발생하게 되므로 메시지의 양이 증폭되게 된다. 그리고 사용자 질의가 선택된 노드로 가는 도중에 여러 노드들이 상태 변경을 과하게 발생시키면 네트워크를 떠돌 가능성이 있다[10].

클러스터 시스템에서 부하 분산을 담당하는 하나 이상의 조정자를 두고, 이를 통해 Round Robin(RR) 기법과 Weighted Round -Robin(WRR) 기법 그리고 Least- Connection (LC) 기법을 적용하는 부하 분산 기법도 있다. RR 기법은 노드의 접속 수나 응답 시간을 고려하지 않고 모두 동일하다는 가정 하에 순서적

으로 작업 노드를 선택하는 방식으로서 스케줄링에 소요되는 시간을 단축할 수 있다. WRR 기법은 각 노드에 처리용량에 비례한 가중치를 부여하여 부하를 분산하는 방식이고, LC 기법은 각각의 노드 중에서 가장 적은 접속량을 가진 노드를 선택하여 부하를 분산하는 방식이다. 이 연구에서의 기법은 사용자 질의가 조정자로 먼저 전달되며, 조정자는 정해진 부하 분산 기법을 통해 적합한 노드를 찾게 된다[9].

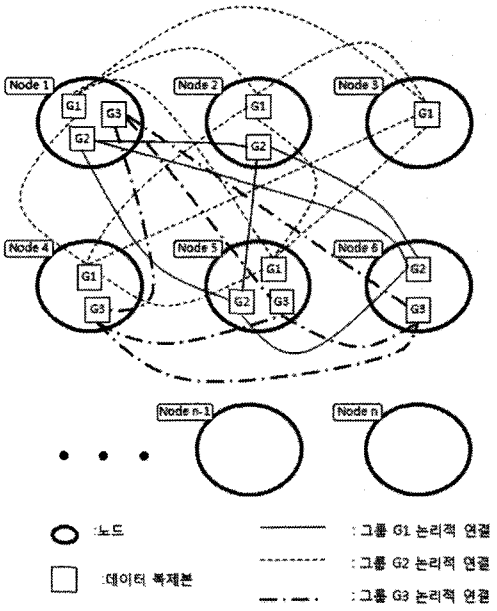
링 연결 기반으로 부하 분산 기법은 동일 복제본이 포함된 노드를 하나의 링 구조로 연결한다. 각각의 복제본들은 정방향 링크와 역방향 링크를 통해 서로 연결되며, 하나의 노드는 자신이 가지는 복제본의 수만큼 연결 링크를 갖게 된다. 임의 노드의 작업 부하가 한계를 넘게 되면 해당 복제본의 정방향 링크가 가리키는 노드로 질의가 전달된다. 그리고 이 노드는 작업 부하가 적어질 때까지 다른 질의에 대한 처리를 중지한다. 또한, 역방향 링크를 통해 메시지를 보내어 자신의 이전 노드가 자신에게 질의를 전달하지 않도록 링크 구조를 수정한다. 작업 부하가 적어지면, 다시 메시지를 이전 노드로 보내어 링크 구조를 원래대로 바꾸며, 질의 처리 과정에 참여한다[6].

## 3. 메쉬 구조를 이용한 부하 분산 기법

본 장에서는 그리드 데이터베이스에서 복제 데이터를 포함하고 있는 노드들을 메쉬 구조로 연결하여 부하를 분산하는 과정을 설명한다.

### 3.1 메쉬 연결 구조와 함수

메쉬 기반 연결 구조는 같은 복제본을 포함하고 있는 모든 노드가 서로 연결되도록 구성한다. 그리드 데이터베이스를 구성하고 있는 모든 노드 중에 같은 복제본을 갖는 노드들로 부분 집합을 만들어 관리함으로써 해당 복제본을 필요로 하는 사용자 질의가 발생할 경우 해당 집합의 노드들 중에 가장 적합한 노드를 선택할 수 있도록 한다. 메쉬 기반 연결 구조로 구성되는 그리드 데이터베이스의 형태는 <그림 1>과 같다.

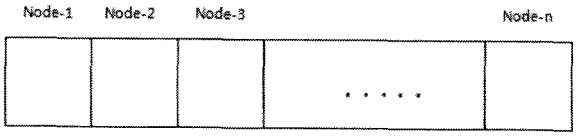



<그림 1> 메쉬 기반 연결 구조

<그림 1>에서는 노드에 포함된 동일 복제본의 논리적인 연결을 선으로 표시한다. Node1, Node2, Node3, Node4, Node5에 있는 복제본 그룹 G1은 메쉬 형태의 논리적 연결을 가짐을 알 수 있다. 각 노드는 동일 복제본의 그룹을 관리하기 위해서 <표 1>과 같은 정보 테이블을 갖는다. 각 그룹은 해당 복제본을 포함하고 있는 노드를 관리하기 위해 해당 노드 디스크립터를 갖는다. 노드 디스크립터는 비트 단위로 해당 노드의 복제본 포함여부를 표시하여 과부하 시에 해당 노드에 메세지를 보내 적합한 노드를 선택할 수 있도록 한다. 노드 디스크립터에 대한 자료구조는 <그림 2>와 같으며 각각은 비트 단위로 각 노드의 동일 복제본 그룹 포함 여부를 표시한다.

<표 1> 복제본 정보 테이블

| 복제본 그룹명 | 복제본 포함 노드 디스크립터 |
|---|---|
| G1 | GD1 |
| G2 | GD2 |
| G3 | GD3 |
| ... | ... |
| Gn | Gn |




<그림 2> 노드 디스크립터 GD의 자료구조

각 노드가 포함하고 있는 노드 디스크립터 정보를 관리하기 위해서는 다음과 같은 함수가 필요하다.

• GD_ZERO(GD_Node *GDnode) : 인자로 전달된 주소의 GD_Node형 변수의 모든 비트를 0으로 초기화 한다.

• GD_SET(int Node, GD_Node *GDnode) : 매개변수 GDnode로 전달된 주소의 변수에 매개변수 Node로 전달된 노드 디스크리터 의 비트를 1로 설정한다. 이 함수는 동일 복제본이 있는 노드를 추가한다.

• GD_UNSET(int Node, GD_Node *GDnode) : 매개변수 GDnode로 전달된 주소의 변수에 매개변수 Node로 전달된 노드 디스크리터 의 비트를 0으로 설정한다. 이 함수는 동일 복제본이 제거된 노드를 삭제 한다.

• GD_ISSET(int Node, GD_Node *GDnode) : 매개변수 GDnode로 전달된 주소의 변수에 매개변수 Node로 전달된 노드 디스크리터 의 비트가 1로 설정되어 있는지 검사한다. 이 함수는 동일 복제본이 있는 노드를 검사한다.

• GD_COUNT(GD_Node *GDnode) : 인자로 전달된 주소의 GD_Node형 변수가 갖고 있는 동일 복제본을 갖는 노드의 수를 반환한다.

• GD_REQUEST(GD_Node *GDnode) : 과부하 노드는 이 함수를 이용하여 동일 복제본을 갖고 있는 노드에게 최대 작업 부하 값과 현재 작업 부하 값을 보내도록 한다.

• GD_RESPONSE(int Node, GD_Info *GDinfo) : GD_REQUEST 함수에의해 요청을 받은 동일 복제본을 포함한 노드는 이 함수를 이용하여 자신의 정보를 매개 변수 Node의 에 해당하는 요청 노드에게 보낸다.

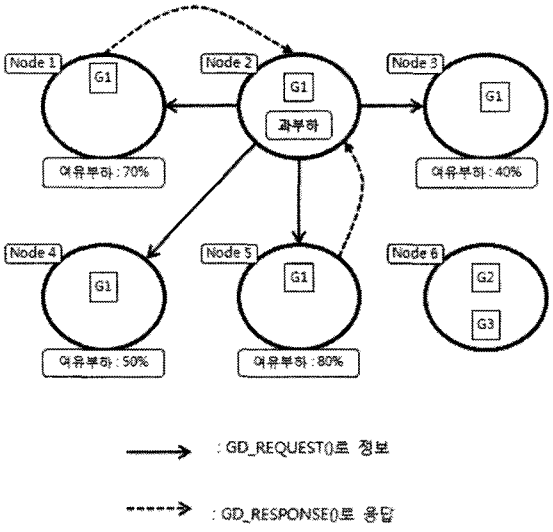
### 3.2 메쉬 연결 구조 기반의 부하 분산 기법

제안 기법이 적용된 그리드 데이터베이스에서 과부하 노드가 발생하면 다음과 같은 절차로 부하를 분산한다.

Step 1: 과부하 노드에 발생한 질의는 해당 질의가 요청하는 동일 복제본 그룹이 있는 노드의 상태를 파악하기 위해 복제본 정보 테이블의 노드 디스크립터를 이용하여 GD_REQUEST()함수를 실행한다.

Step 2: GD_REQUEST() 함수에 의해 요청을 받은 노드는 GD_RESPONSE() 함수를 이용하여 자신의 최대 부하 값에서 현재 작업 부하 값 뺀 나머지 값을 보낸다.

Step 3: GD_RESPONSE() 함수에 의해 동일 복제본이 있는 노드로 부터 응답을 받은 과부하 노드는 동일 복제본의 모든 노드로부터 응답을 다 받을 때까지 대기하지 않고 GD_COUNT() 함수에 의해 파악한 동일 복제본 그룹의 노드 수의 50%가 응답하면 그 중에서 응답 값이 가장 큰 노드로 해당 질의를 보낸다. 노드 수의 50%만 응답을 받는 이유는 첫째, 과부하 노드는 답을 늦게 하거나 하지 않을 것이고 둘째, 과부하 노드가 없더라도 모든 노드의 응답을 기다리기 보다는 먼저 도착한 처리 가능한 일부 노드 중에 선택하는 것이 보다 효율적이기 때문이다.




<그림 3> 적용 예

<그림 3>은 Node 2가 과부하 상태이고 이 노드에서 그룹 G1에 대한 사용자 질의가 발생되었을 경우의 예를 보여준다. 실선 화살표는 그룹 G1이 있는 노드에 GD_REQUEST()함수로 상태 정보를 요청하는 것이고, 점선 화살표는 이에 대해 GD_RESPONSE() 함수로 노드가 응답하는 것을 표시한 것이다. 그룹에서 총 노드에서 자신을 제외한 네 개의 노드 중 두 개의 노드에서 응답이 오면 나머지 노드에 대한 응답을 기다리지 않고 이 응답 결과에서 가장 여유부하가 큰 Node 5를 선택하여 사용자 질의를 보낸다.

## 4. 성능 평가

본 장에서는 제안 기법의 평가를 위한 실험 환경을 설명하고, 노드의 수, 질의 발생도, 동일 복제 노드의 수 등의 변화에 따른 기존 기법과 제안 기법의 성능을 비교 평가한다.

### 4.1 실험 환경

본 논문에서 제안 기법을 평가하기 위해 사용한 시뮬레이션 툴은 분산 환경에서의 컴퓨팅 모델과 알고리즘 평가가 가능한 CSIM를 사용하였고 시뮬레이션 평가를 위한 적용 요소와 값의 범위는 <표2>와 같다[11].

<표 2> 시뮬레이션 적용 요소와 값의 범위

| *적용 요소* | *값의 범위* |
|---|---|
| 실험 시간 | 3000 unit time |
| 노드 수 | 100개 |
| 과부하 노드 수 | 0~50개 |
| 클라이언트 수 | 200개 |
| 동일 복제본 그룹 수 | 1~200개 |
| 질의 처리 시간 | 4 unit time |
| 질의 전송 시간 | 2 unit time |
| 질의 입력 간격 | 0.1 unit time |

실험을 위한 비교 평가는 LIN 방식, RING 방식 그리고 제안 기법인 MESH 방식을 대상으로 한다.

LIN 방식은 각 노드가 자기 자신과 바로 연결된 노드들과 정보를 교환하여 노드를 저부하, 보통, 과부하의 세 가지 상태로 나누어 과부하 노드로 부터 최소 인접 거리에 있는 저부하 노드로 질의를 전송한다.

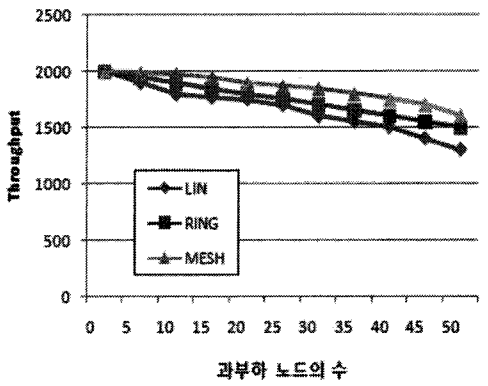
RING 방식은 동일 복제본의 포함된 노드들을 하나의 링 구조로 연결하고 과부하 노드가 발생하면 질의를 정방향 노드로 전달하고 역방향으로 자신에게 질의 전달이 되지 않도록 하는 방식으로 처리한다.

MESH 방식은 본 논문에서 제안한 기법으로 동일 복제본을 포함한 노드들을 모두 연결하여 해당 복제본에 대한 질의 발생 시 질의가 발생한 노드가 과부하 상태이면 연결된 모든 노드 중에서 가장 적합한 노드로 해당 질의를 전송하여 처리한다.

과부하 노드의 기준은 노드의 최대 부하 값에서 현재 부하 값을 감했을 때 10%의 이하인 경우를 과부하 노드로 정하였다.

### 4.2 실험 평가

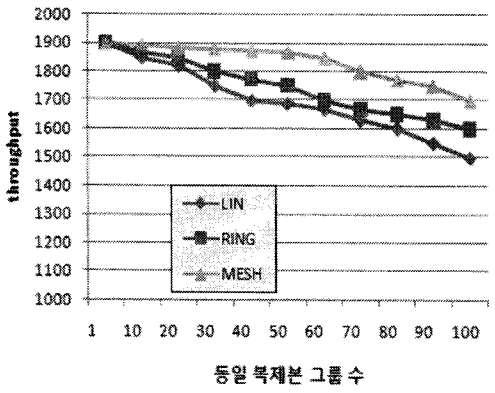
본 실험 평가는 두 가지 경우로 나누어 수행되었다. 첫 번째 실험은 노드의 수와 클라이언트의 수 그리고 동일 복제본 그룹 수 등의 값을 고정시키고 과부하 노드의 수를 변경시키면서 평가를 했고, 두 번째 실험은 노드의 수, 클라이언트 수 그리고 과부하 노드 수 등의 값을 고정시키고 동일 복제본 그룹 수를 증가시키면서 평가하였다. 첫 번째 실험에서 동일 복제본의 그룹 수는 실험의 중간 값인 50으로 하였고 두 번째 실험에서 과부하 노드의 수는 실험의 중간 값인 25로 하였다. 실험 성능의 평가는 처리량(Throughput)을 기준으로 표시하였다.




<그림 4> 과부하 노드 수에 따른 처리성능

<그림 4>는 과부하 노드 수가 증가함에 따라 시간당 처리율이 어떻게 변하는지 보여준다. LIN 방식은 저부하 노드는 0 그렇지 않으면 1이라는 거리 값을 갖기 위해 부하가 변경될 때 인접 노드들의 상태를 메시지로 가져와야 하고 계산 결과를 인접 노드들에 전파시켜야 하는 부담으로 과부하 노드의 수가 증가됨에 따라 반비례해서 처리율이 낮아짐을 알 수 있다. RING 방식은 링크로 연결된 구조로 과부하 노드에서 질의를 인접한 처리 가능한 노드에 보내기 때문에 가장 적합한 노드에 보내지 못한다. 따라서 과부하 노드가 점진적으로 증가할 때 제안 방법 보다는 전반적인 처리율이 낮게 나타남을 알 수 있다. 제안 방법은 다른 기법보다 처리율이 높게 나왔는데 그 이유는 다음과 같다.

첫째 이유는 과부하 노드에서 질의 발생 시 메쉬 방식으로 연결된 모든 노드에 상태 정보를 요구하고 복제본 그룹 노드의 반이 응답을 하면 그중 가장 적합한 노드에게 질의를 보내기 때문으로 처리 속도가 빠르다. 두 번째 이유는 동일 복제본 그룹을 포함한 노드 정보를 주기적으로 점검하거나 상태 정보에 대한 관리 부담이 없기 때문으로 판단된다.




<그림 5> 복제본 그룹 수에 따른 처리성능

<그림 5>는 동일 복제본 그룹의 증가에 따른 시간당 처리성능을 보여주고 있다. 그림에서 LIN 방식은 주어진 과부하 노드의 실험값이 50으로 실험에 사용되는 노드의 반이 처리를 하지 못하는 상황에서 인접 노드에 대한 정보관리의 부담으로 인해 복제본 그룹이 증가 할 수록 처리율이 감소함을 알 수 있다. 본 논문에서 제안하고 있는 MESH 방식은 RING 방식보다 복사본이 많은 상황에서도 좋은 성능을 보이고 있다, 그러한 이유는 동일한 복제본 그룹을 관리하기 위한 부담이 상대적으로 RING 방식보다 적기 때문이고 적합한 노드의 선택도 RING 방식보다 빠르기 때문이다. RING 방식은 복제본 그룹이 증가할수록 관리할 링크 수와 여기에 연결된 노드의 정보를 관리하는 부담이 증가하기 때문으로 보인다.

## 5. 결 론

노드의 수가 많고 사용자 질의가 유동적으로 변하는 그리드 데이터베이스 환경에 적합한 부하 분산을 위해 기존의 기법은 모든 노드에 대한 데이터의 정보를 유지하는 메타 데이터베이스를 관리하거나 링 구조로 구성하여 부하를 분산하는 방법을 사용하여 관리에 부하가 많거나 신속하게 적합한 노드를 결정하는데 많은 시간이 걸리는 문제가 제기되었다.

본 논문에서는 메쉬 기반의 연결 구조를 이용하여 동일한 복제본을 포함한 노드들을 연결하여 과부하 노드에서의 질의 발생 시 가장 적합한 노드를 신속하게 결정하여 처리할 수 있도록 하였고 동일 복제본 그룹의 연결 노드에 대한 상태 정보의 관리 부담이 없도록 하였다.

성능 평가에서 제안 기법은 과부하 노드의 증가 시에 기존 방법 보다 좋은 성능을 보였고 동일 복제본 그룹의 증가 시에도 다른 기법보다 좋은 성능 보여주었다.

## 참 고 문 헌

[1] Fran Berman, Geoffrey Fox, Tony Hey, Grid Computing, John Wiley & Sons, Ltd, 2003

[2] 강철, "On-Demand 환경의 데이터베이스 Grid와 고 가용성," 데이터베이스 연구회, Vol. 00 No. 00 pp. 209-224, 2005

[3] Anastasiadi, S. Kapidakis, C. Nikolaou, and J. Sairamesh, "A computational economy for

dynamic load balancing and data replication," in Proc. 1st International Conference on Information and Computation Economies, pp. 166~180, Charleston, SC, 1998.

[4] J. Gray, P. Helland, P. O'Neil and D. Shasha, "The dangers of replication and a solution," ACM SIGMOD International Conference on Management of Data, Published as SIGMOD RECORD, 25(2): pp. 173~182, ACM, 1996.

[5] Zhiling Lan, Valerie E. Taylor and Greg Bryan, "Dynamic load balancing of SAMR applications on distributed systems," Scientific Programming, 10(4): pp. 319~328, 2002.

[6] 장용일, 신승선, 박순영, 배해영, "그리드 데이터베이스에서 링 기반 연결 구조를 이용한 부하 분산 기법," 멀티미디어학회 논문지, 제9권 제9호, 2006.

[7] S. Ceri and G. Pelagatti, Distributed Databases: Principles & Systems, McGraw-Hill Company, 1984.

[8] Maria A., Nieto-Santisteban, Jim Gray, Alexander S. Szalay, James Annis, Aniruddha R. Thakar and William J. O'Mullane, "When Database System Meet Grid," Proceedings of the 2005 CIDR Conference, pp. 154~161, 2005.

[9] "Job Scheduling Algorithms in Linux Virtual Server," http://www. linuxvirtualserver.org/docs/scheduling.html, Dec. 2003

[10] F.C.H. Lin and R.M. Keller, "The Gradient Model Load Balancing Method," IEEE Trans. on software Engineering, vol. SE-13, No. 1, pp. 32-38, 1987.

[11] Mesquite Software Inc. CSIM19 The Sumulation Engine, 2005, http://www.mesquite.com

**이 순 조** (Lee Soon Jo)

- 정회원
- 인하대학교 전자계산학과 이학사
- 인하대학교 전자계산학과 이학석사
- 인하대학교 전자계산공학과 공학박사
- 서원대학교 컴퓨터공학과 교수
- 관심분야 : 데이터베이스, GIS, 정보보안