XMDR을 이용한 데이터 그리드 미들웨어의 협력 에이전트 설계

노선택° 문석재 엄영현 국윤규 정계동 최영근
광운대학교 컴퓨터 과학과
{pearl°, msj1568, class76, ykkook, gdjung, ygchoi}@cs.kw.ac.kr

A Design of Collaborative Agent for Data Grid MiddleWare using XMDR

Seontaek Noh° S.J. Moon Y.H. Eum Y.G. Kook G.D. Jung Y.G. Choi
Department of Computer Science, KwangWoon University

요 약

최근 기업환경에서는 분산되어 있는 정보를 통합하여 정보 공유의 필요성이 증가함에 따라 기존 레거시 시스템간의 협업을 하기 위한 상호 운용이 강조되고 있다. 독립적인 레거시 시스템의 상호 연결을 위해서는 플랫폼 이질성, 의미 이질성 등을 극복할 필요가 있다. 이러한 문제를 해결하기 위해 ISO/IEC 11179에서 진행하고 있는 XMDR을 이용하여 미들웨어를 설계하였다. 설계한 미들웨어를 레거시 시스템에 적용하여 데이터 공유 및 통합의 일관성을 유지할 수 있게 되었다. 하지만 설계된 미들웨어는 각 노드의 자원 상황과 작업 상황에 대한 조정기능이 없기 때문에 정보 활용의 효율성을 보장할 수 없다. 따라서 레거시 시스템을 관리하고 조정하는 방안이 필요하다. 본 논문에서는 정보를 요청하는 요청 에이전트와 정보를 제공하는 정보 에이전트간의 정확한 정보 교환을 할 수 있도록 조정하고, 각 레거시 시스템의 정보 모니터링과 작업 분배 및 로컬 노드의 자원 관리를 담당하는 협력 에이전트를 설계함으로써 통합된 정보를 효율적으로 활용할 수 있도록 한다.

1. 서 론

현재 많은 기관이나 기업들은 고유의 업무특성에 맞도록 독립적인 레거시 시스템을 구축하여 사용해 왔다. 이러한 환경에서 응용업무가 복잡해지고 다양해지면서 기존에 구축되어 있는 레거시 시스템을 네트워크를 통해 상호운용하려는 필요성이 증가하게 되었다[9]. 특히 이들 레거시 시스템들내에서 운영되고 있는 데이터베이스는 서로 다른 DBMS를 사용하고 있을 뿐만 아니라 데이터의 형태, 단위, 형식 등이 서로 다르기 때문에 이러한 이질성들을 극복해야 한다.

이러한 데이터 요소들의 이질성은 최근 ISO/IEC 11179에서 제안하는 MDR과 온톨로지를 결합한 XMDR을 이용하여 구축한 데이터 그리드 미들웨어를 통해 해결할 수 있다[8]. 또한 온톨로지와 MDR의 개념을 이용한 시스템에서, 효율적인 정보교환을 하기 위해서는 독립적이며 지능적인 에이전트가 요구된다. 따라서 상호운용성을 지닌 에이전트 기술을 사용함으로서 이질적인 데이터베이스 기반의 레거시 시스템에서 자유로운 정보 교환을 가능하게 한다. 하지만 에이전트를 이용하여 노드간의 데이터를 활용하기 위해서는 에이전트와 각 노드간의 메시지 통신을 효율적으로 해야 할 필요성이 있다. 또한 목표 노드에 작업 요청 에이전트가 동시에 몰릴 경우 요청된 작업을 정확하게 수행되지 않거나 응답하지 않는 경우가 발생하게 된다. 따라서 목표 노드에 작업 부하가 발생하지 않도록 조정할 수 있는 메카니즘을 추가할 필요성이 있다.

본 논문에서는 XMDR을 이용한 데이터 그리드 미들웨어에 적용되는 협력 에이전트(Collaborative Agent:SA) 시스템을 제안한다. 협력 에이전트 시스템은 레거시 시스템에서 추출한 정보 제공을 담당하는 정보 에이전트(Informaion Agent:IA), 그리고 정보를 요청하게 되는 요청 에이전트(Request Agent:RqA)와 정보 에이전트의 사이에서 에이전트 통신을 담당하는 조정 에이전트(Coordinating Agent:CA) 로 구성된다. 요청 에이전트를 통해 커뮤니티에 속해 있는 레거시 시스템에 데이터 정보를 요청할 때, 조정 에이전트는 미들웨어를 통해 추출된 로컬 자원 정보를 바탕으로 요청 에이전트의 프로세싱을 결정함으로써 데이터 그리드 미들웨어가 속해 있는 레거시 시스템의 작업 부하를 조정하게 된다. 따라서 XMDR을 이용하여 구성된 커뮤니티 내에서 각 노드의 작업 부하를 줄일 수 있으며 정보 요청에 대한 신뢰성과 효율성을 보장할 수 있다.

2장에서는 관련 연구에 대해 서술하고, 3장은 XMDR을 이용한 데이터 그리드 미들웨어에 대해 설명한다. 4장에서는 데이터 그리드 미들웨어에 적용되는 협력 에이전트 설계에 대해 설명하며, 5장은 시스템 적용과 비교분석 및 성능 평가에 대해 기술하고, 마지막으로 6장은 결론 및 향후 연구에 대해서 기술한다.

2. 관련 연구

2.1 데이터 그리드 미들웨어 구조

본 논문에서 제안하는 협력 에이전트 시스템이 적용될 데이터 그리드 미들웨어의 구조는 4계층으로 구성된다 [8]. 각 계층은 관리 부분과 응용 부분으로 이루어져 있으며 관리 계층은 다시 네트워크 계층, 시스템 계층, 서비스 계층, 응용계층으로 구성된다. 협력 에이전트 시스템은 협업에 관한 시스템 정책을 관리하고 에이전트를 관리하는 시스템 계층에 구성된다. 에이전트 관리자(Agent Manager)는 같은 계층에 속해 있는 시스템 정책을 바탕으로 서비스 계층과 네트워크 계층사이에서 조정/관리하는 역할을 담당한다. 그림 1은 협력 에이전트 시스템이 적용되는 데이터 그리드 미들웨어의 구조도이다.

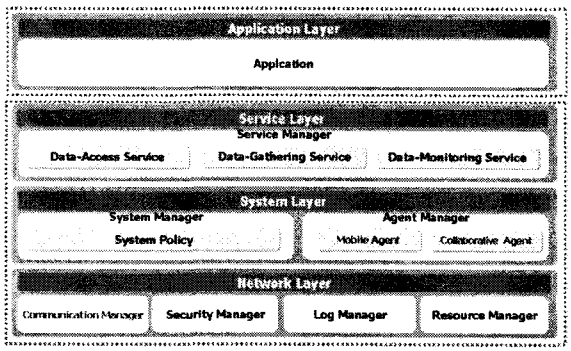

그림 1. 데이터 그리드 미들웨어 구조

2.2 XMDR을 이용한 데이터 그리드 커뮤니티

본 논문에서 설계하는 협력 에이전트는 그림 2와 같은 데이터 그리드 커뮤니티 환경에 적용된다. 그리고 기존에 구축된 레거시 시스템간의 데이터 이질성을 해결하기 위해 XMDR을 구성하게 된다. XMDR은 MDR과 온톨로지를 포함한 확장된 개념을 말한다. MDR은 레거시 시스템들이 실제 사용되고 있는 데이터베이스의 스키마 정보를 수집하여 구성하였으며, 온톨로지는 표준 메타데이터 온톨로지와 위치 메타데이터 온톨로지로 구성된다. 표준 메타데이터 온톨로지는 레거시 시스템간의 메타데이터 공유 및 통합하기 위해 메타데이터의 의미적인 내용과 데이터 요소의 구문을 위한 표준을 제시하며, 위치 메타데이터 온톨로지는 응용 시스템의 위치정보, 데이터베이스 정보, 테이블 정보를 제공해준다.

각 노드간의 통신은 플랫폼에 독립적인 성격을 가지고 있는 이동 에이전트 기술을 이용한다. 이는 플랫폼 이질적인 문제를 해결할 뿐 아니라 연결을 항상 유지할 필요 없이 에이전트를 보낼 때와 결과를 받을 때만 연결되게 함으로써 노드간의 부담을 줄일 수 있다는 장점이 있다. 이런 환경을 바탕으로 협업을 위해 형성된 커뮤니티는 파일 공유, 분산 컴퓨팅, 데이터 공유 및 교환, 정보 검색 등을 함으로써 공동 작업을 수행할 수 있다.

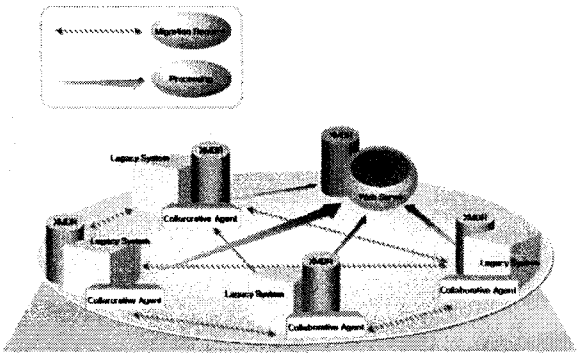

그림 2. 데이터 그리드 커뮤니티 개요

웹서버를 통하여 등록된 각 노드의 XMDR 정보는 XMDR 서버에 저장된다. 또한 각 노드에서 주기적으로 발생하는 자원 정보를 모니터링하여 수시로 갱신함으로서 동적으로 우선순위를 결정하게 된다. 따라서 웹서버는 이주 경로 서비스를 요청한 노드가 수행해야할 작업에 따라 최적의 이주 경로를 서비스하게 된다. 커뮤니티에 속해 있는 노드에 서비스를 요청하게 되면 요청 메시지가 이벤트에 의해 생성되는 요청 에이전트를 통해 각 노드에 통신을 하게 되며, 각 노드로 전달되는 요청 에이전트의 이주 경로는 웹서버에서 결정된 우선순위에 따라 결정된다. 목적 노드에 도착한 요청 에이전트는 데이터 그리드 미들웨어에 구성되어 있는 협력 에이전트에서 수행을 시작하게 된다. 이때 협력 에이전트에 속해 있는 조정 에이전트가 요청 에이전트의 요청 특성에 따라 수행 여부를 결정하게 된다. 또한 로컬 노드에서 추출한 자원 정보 및 작업 현황에 따라 요청 에이전트의 수행 여부를 결정하게 된다.

조정 에이전트에 의해 요청 에이전트의 수행이 결정된 경우에는 에이전트 스레드가 구동되며 요청 에이전트가 수행되어 정보를 제공해주는 정보 에이전트와 통신하게 된다. 수행되는 요청 에이전트는 정보 에이전트에게 요청 메시지만 전달하고 다음 수행 노드로 이동하게 되며 수행 결과는 조정 에이전트에서 생성되는 응답 에이전트(Respond Agent:RpA)를 통해 최초 요청했던 노드로 전달된다. 로컬 노드의 자원 상태나 작업 수행의 양이 한계치 이상이 되는 경우에는 에이전트 정보를 큐에 저장하여 작업이 가능한 경우에 순차적으로 수행하게 하거나 요청 에이전트의 이주 우선순위를 변경하여 다음 노드로 이동시킨다. 로컬 노드의 상태에 따른 관리에 대한 내용은 3장에서 자세히 기술한다.

2.3 협력 에이전트(Collaborative Agent)

협력 에이전트는 협조 에이전트라고도 불리며, 다중 에이전트 시스템을 구성하는 에이전트를 말한다. 다중 에이전트 기반 구조는 다수의 에이전트들을 어떻게 구성하고, 에이전트들이 어떻게 메시지를 주고받을 것인가를

정의하는 부분이다. 다중 에이전트 기반구조는 조정 에이전트를 통하여 메시지를 주고받는 중앙 집중식 구조와 조정 에이전트 없이 각 에이전트들이 다른 에이전트에 대한 정보를 가지고 상호 메시지를 주고받는 분산식 구조로 나눌 수 있다[8]. 본 논문에서 제안하는 협력 에이전트 시스템의 구조는 조정 에이전트를 이용한 중앙 집중식 구조로 되어 있다. 중앙 집중식 구조는 각 응용 에이전트들이 다른 에이전트에 대해 에이전트 위치라든가 제공 서비스를 파악할 필요 없이 원하는 서비스를 조정 에이전트에게 요청하기만 하면 그 이후의 일은 조정 에이전트가 처리해 줌으로써 각 응용 에이전트의 구성이 간단해지고, 조정 에이전트를 통하여 응용 에이전트들을 관리함으로써 에이전트들에 대한 관리가 용이해 진다는 장점이 있다.

3. 협력 에이전트 시스템 설계

3.1 협력 에이전트 시스템 구조

본 논문의 협력 에이전트 시스템은 2계층으로 구성된다. 정보 에이전트는 네트워크 계층에서 추출된 로그 정보나 자원 정보 및 보안 정보, 서비스 계층에서 제공하는 요청된 데이터와 결과 정보를 제공한다. 그리고 조정 에이전트는 정보나 데이터를 요청하는 요청 에이전트와 정보 에이전트간의 통신을 관리한다. 각 에이전트의 수행은 각 에이전트 계층 안에 속해있는 관리자에 의해 수행된다. 그림 3은 협력 에이전트의 구조이다.

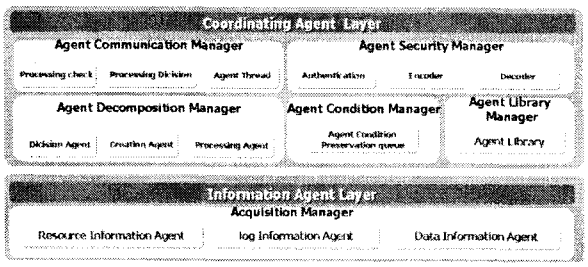

그림 3. 협력 에이전트 시스템의 구조

3.1.1 조정 에이전트 계층

조정 에이전트 계층에는 모두 5개의 관리자로 구성되어 있으며 각각의 관리자는 다른 관리자, 다른 계층의 관리자와 통신을 하며 수행하게 된다.

◦ 에이전트 통신 관리자(Agent Communication Manager) : 요청 에이전트와 정보 에이전트간의 통신을 조정한다. 에이전트간의 통신은 단순한 데이터의 이동이 아닌 프로세스 및 상태, 수행 데이터 등을 이동해야 하기 때문에 프로세스 등을 복제하는 방법의 구현이 필요하다. 이는 복제 프로세스를 같이 전송하는 방법과 다양한 상태 데이터를 직접 전송하는 방법이 있으며, 프로세스가 이동되어 생성된 후 필요한 데이터만 요구하는 방법 등이 있다. 요청 에이전트가 목적 노드에 도착했을 때 로컬 노드의 자원 상태와 작업 현황을 파악하여 프로세스 수행을 결정해야 한다. 만약 로컬 노드의 작업 현황이나 자원 상태가 한계치를 넘게 되었을 경우에는 요청 에이전트에 포함되어 있는 이주 경로를 추출한 뒤, 우선순위를 변경하여 요청 에이전트를 다음 노드로 이동시키거나 에이전트 상태 관리자에 있는 에이전트 상태 보존 큐에 저장하여 작업을 대기시키도록 한다. 이러한 정책들은 그림 3의 시스템 계층에 존재하는 시스템 관리자를 참조하여 결정된다.

◦ 에이전트 보안 관리자(Agent Security Manager) : 에이전트와 에이전트에 포함되어 있는 데이터나 정보에 대한 인증과 암호화/복호화의 기능을 수행한다. 에이전트간의 통신은 프로세스의 이동 및 교환이 주가 되기 때문에 악의적인 프로세스가 포함되었을 경우에는 로컬 노드의 시스템에 막대한 오류를 범할 수 있다. 따라서 로컬 노드로 들어오게 되는 요청 에이전트들의 인증 알고리즘을 관리하는 모듈이 포함된다. 또한 요청 에이전트에 의해 요청된 데이터의 수행 결과, 요청 결과 및 상태 결과에 대해 요청 노드로 전송할시 데이터를 암호화 시키고 해독할 수 있는 기능도 포함되어 있다.

◦ 에이전트 분리 관리자(Agent Decomposition Manager) : 요청 에이전트의 유형 판단과 에이전트 생성 및 서비스별 에이전트 분리에 관한 기능을 수행한다. 에이전트 통신 관리자에 의해 결정된 요청 에이전트는 에이전트 스레드를 통해 프로세스 수행을 시작한다. 에이전트 분리 관리자는 수행되기 시작한 요청 에이전트의 유형을 판단하여 데이터 그리드 미들웨어에서 제공하는 서비스를 결정하게 되며, 요청에 맞는 정보 에이전트와의 통신 및 수행을 하게 된다. 또한 에이전트 분리 관리자는 정보 에이전트를 통해 전달받은 요청 데이터나 상태 정보 메시지를 요청 노드로 전달하기 위해 응답 에이전트를 생성하는 역할도 한다. 이때 생성된 응답 에이전트는 각 정보 에이전트의 서비스에 따라 유형이 결정된다.

◦ 에이전트 상태 관리자(Agent Condition Manager) : 에이전트 통신 관리자에 의해 로컬 노드에 대기하도록 결정된 요청 에이전트를 에이전트 상태 보존 큐에 저장하는 역할을 한다. 이때 저장되는 요청 에이전트는 프로세스 및 각 데이터와 상태정보를 복제하게 된다. 복제가 마친 요청 에이전트는 자신의 복제 에이전트를 큐에 저장하고 다음 노드로 이동하여 수행한다.

◦ 에이전트 라이브러리 관리자(Agent Library Manager) : 이동하는 코드를 라이브러리화 하는 기능을 수행한다. 본 논문에서 제안하는 협력 에이전트 시스템의 요청 에이전트와 응답 에이전트는 이동 에이전트 기술을 활용하였기 때문에 코드와 프로세스가 이동하게 된다. 이동하는 코드나 프로세스 중 자주 사용하는 기능을 라이브러리화 하게 되면 요청 에이전트와 응답 에이전트의 크기를 줄일 수 있다. 또한 라이브러리에 등록되어

있기 때문에 수행 속도를 증가시킬 수 있다.

3.1.2 정보 에이전트 계층

정보 에이전트는 로컬 노드에서 서비스하는 정보와 데이터를 제공하는 역할을 한다. 정보 에이전트 계층에는 수집 관리자에 의해 3개의 에이전트를 관리한다. 데이터 그리드 미들웨어의 자원 관리자에 의해 주기적으로 수집되는 자원 정보를 제공하며 로그 에이전트는 로컬 노드에서 수행하는 각각의 에이전트들의 로그와 데이터베이스의 로그정보 등을 제공한다. 데이터 정보 에이전트는 데이터 그리드 미들웨어에서 제공하는 서비스에 대한 정보를 제공하는 역할을 한다.

3.2 협력 에이전트의 수행 과정

협력 에이전트의 수행과정은 그림 4와 같다. 요청 에이전트가 목적 노드에 도달하게 되면 조정 에이전트는 에이전트 보안 관리자에 의해 수행되는 에이전트 인증을 수행한다. 이때, 인증이 완료된 에이전트는 최초 요청 에이전트를 생성할 시에 암호화 되어있는 메시지 정보를 복호기를 통해 해독하게 된다.

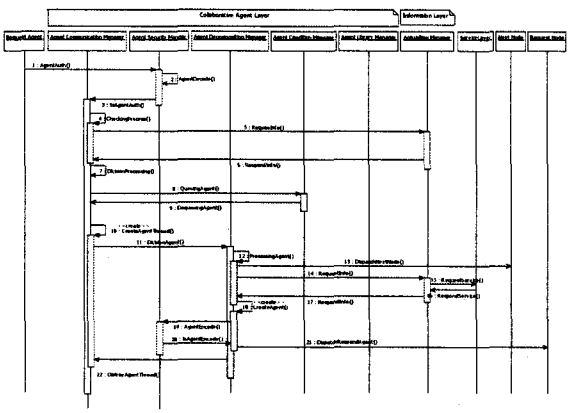

그림 4. 협력 에이전트 관리자 수행 과정

해독이 완료된 에이전트는 에이전트 통신 관리자에게 완료가 되었음을 알리고 에이전트 인증 수행을 완료한다. 에이전트 통신 관리자는 인증 완료된 요청 에이전트를 수행하기 전 로컬 노드의 자원 정보와 작업 수행과정에 대한 정보를 정보 에이전트에게 요청한다. 이때 참조하는 자원 정보는 로컬 노드의 CPU점유율과 메모리 할당 정보 및 에이전트 스레드의 개수 등을 가리킨다. 또한 작업 수행과정에 대한 정보는 미들웨어에 존재하는 로그 관리자에 의해 관리되는 정보를 가리키며 정보 에이전트에 의해 수집된다. 에이전트 통신 관리자는 정보 에이전트에 의해 수집된 자원을 바탕으로 요청 에이전트의 수행을 결정한다. 수행이 결정된 이후의 요청 에이전트의 처리는 다음과 같다

◦ 수행불가(ProcessDisallowance) : 로컬 자원이 허용치를 넘어섰을 경우 로컬 노드내에서의 요청 에이전트 수행은 거부된다. 따라서 조정 에이전트를 통해 요청 에이전트가 가지고 있는 이주 경로를 변경하여 다른 노드부터 수행을 한 뒤 방문할 수 있도록 한다.

◦ 수행대기(ProcessStandingBy) : 로컬 자원은 허용치를 넘지 않았으나 로컬 노드내의 에이전트 스레드가 다수 생성되어 더 이상 작업을 수행할 수 없을 경우 로컬 노드내에서의 요청 에이전트 수행은 대기된다. 수행대기가 된 요청 에이전트는 에이전트 상태 보존 큐에 저장되어 순차적으로 실행된다. 이때 저장되는 요청 에이전트는 자신이 저장되어 실행을 대기하는 것이 아니라 복제하여 큐에 저장한 뒤 다음 노드로 이주하게 된다.

◦ 수행허용(ProcessPermission) : 로컬 자원과 에이전트 스레드가 허용치 이내인 경우 요청 에이전트는 로컬 노드 내에서 수행을 할 수 있다.

에이전트 통신 관리자는 수행이 결정된 요청 에이전트가 동작할 수 있도록 에이전트 스레드를 생성한다. 스레드가 생성되는 동시에 에이전트 분리 관리자는 요청 에이전트의 종류를 판단하게 된다. 요청 에이전트의 종류에 따라 프로세스 할 수 있는 정보 에이전트와 연결을 시켜주며 요청에 대한 수행을 시작한다. 수행이 완료된 후 요청에 대한 응답 정보는 에이전트 분리 관리자에 의해 생성된 응답 에이전트에 저장되며, 에이전트 보안 관리자에 의해 암호화되어 최초 요청 노드로 전달된다. 요청 노드로 전달된 후 에이전트 통신 관리자에게 작업이 완료되었음을 알리고 요청 에이전트의 스레드를 소멸시킨다.

4. 성능 평가

본 논문에서 제안한 협력 에이전트의 성능 측정을 위해 기존에 협력 에이전트를 적용하지 않은 시스템과의 비교 실험을 수행하였다. 기존의 시스템은 목적노드에 요청 에이전트가 도달할 경우 반응적으로 에이전트 스레드가 생성되어 실행하는 방식을 가지고 있다. 따라서 로컬 노드의 자원상황이나 작업현황에 상관없이 요청 에이전트가 실행되며 작업부하에 따른 요청 에이전트의 수행 책임은 고려되지 않는다. 성능 측정을 위한 실행 환경과 방법은 다음과 같다.

실행 환경
- Windows Server 2003 / Window XP professional : 7대
- Linux Fedora Core 4 : 3대
- 데이터베이스 환경 : MS-SQL, Oracle 8i

실행 방법
- 도서 정보 검색 요청 에이전트를 통한 정보 검색 (도서 ISBN, 도서명, 저자등)
- 요청 에이전트 증가에 따른 각 노드의 CPU 점유율 측정
- 요청 에이전트 증가에 따른 각 노드의 응답 실패율 측정

- 요청 에이전트 증가에 따른 각 노드의 응답 시간 측정
(응답 에이전트의 최종 도달 시간 측정)

본 실험에서 요청 에이전트가 요구하는 도서 정보의 데이터는 10개의 노드에 모두 존재한다고 가정하였으며 실험을 통해 측정된 값은 25회의 반복 실험을 통한 평균값을 나타낸다. 각 실험에서 사용되는 요청 에이전트의 요청 정보는 매회 동일하게 설정하였다.

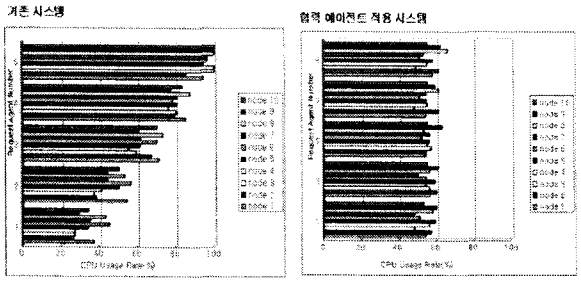

그림 5. 요청 에이전트 수에 따른 각 노드의 CPU 점유율 그래프

그림 5는 요청 에이전트 수에 따른 각 노드의 CPU 점유율을 나타내고 있다. 기존 시스템에서의 CPU 점유율은 로컬노드에 요청 에이전트가 증가할수록 점유율이 선형적으로 증가함을 알 수 있다. 요청 에이전트가 5개 이상이 되면 CPU 점유율이 100%에 다다르게 되며, 요청 에이전트의 개수가 6개 이상이 되면 정상적인 검색 수행이 되지 않거나 시스템자체가 동작하지 않는 현상이 발생하였다. 반면, 협력 에이전트 적용 시스템에서의 CPU 점유율은 조정 에이전트를 통해 CPU등의 로컬 자원의 한계치에 따라 작업 조정을 하고 있기 때문에 50% 내외로 일정하게 유지되고 있다. 따라서 요청 에이전트의 수가 증가하여도 로컬 노드의 CPU 점유율은 항상 일정하게 유지된다.

표 1. 요청 에이전트 수에 따른 각 노드의 응답 결함률

기존 시스템

	node 1	node 2	node 3	node 4	node 5	node 6	node 7	node 8	node 9	node 10
1	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
5	1.8	1.61	2.65	1.53	0	3.23	1.12	0	0	0.21
7	4.98	3.61	2.82	5.52	4.47	5.23	2.62	3.22	5.12	3.42
8	11.21	9.67	10.21	10.11	5.9	9.12	8.55	8.21	13.07	8.8

협력 에이전트 적용 시스템

	node 1	node 2	node 3	node 4	node 5	node 6	node 7	node 8	node 9	node 10
1	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
7	0.76	0	0	0	0.21	0	0.52	0	0.44	0
8	2.56	1.24	0.69	0.44	0.35	0.12	1.55	0.01	0.24	1.77

(단위 : %)

표 1은 요청 에이전트 수에 따른 각 노드의 응답 결함률을 나타낸다. 본 실험에서의 응답 결함률이란 요청 노드에서 요청 에이전트를 통해 도서 정보를 검색하였을 경우 요청에 대한 정보가 요청 노드로 정상적으로 도달하지 못한 경우이거나 목표 노드의 데이터나 정보가 누락된 경우를 나타내는 수치이다.

기존 시스템에서의 응답 결함률은 3개의 요청 에이전트를 수행할 때까지는 정상적인 응답 정보를 얻을 수가 있으나 5개 이상일때부터는 응답 결함이 발생하게 된다. 또한 9개 이상일때부터는 10번의 요청을 하게 되었을 경우 적어도 1번 이상 올바른 정보를 보장하지 못하게 된다. 정보 결함이 발생하는 이유는 요청 에이전트 생성 시 요청 에이전트에 대기 TTL(Time-to-live) 정보를 포함시키기 때문이다. 요청 에이전트의 대기 TTL 정보는 목표 노드로 도달한 요청 에이전트가 목표 노드의 자원 정보나 작업 현황에 따라 에이전트 스레드를 생성하지 못했을 경우 대기하는 시간을 나타내는 정보이다. 대기 TTL에 설정된 시간이 지났을 경우에는 다른 노드로 이주를 하거나 소멸하게 된다. 따라서 기존 시스템에서는 5개 이상의 에이전트 스레드를 발생하였을 경우 무한 대기하는 요청 에이전트가 증가하기 때문에 소멸되거나 올바른 정보를 획득하지 못하는 경우가 발생하게 된다. 반면, 협력 에이전트를 적용한 시스템의 경우에는 조정 에이전트의 에이전트 상태 보존 큐에 프로세싱하지 않고 그대로 저장되기 때문에 대기 TTL에 의해 취소되거나 소멸되는 경우를 줄일 수 있다.

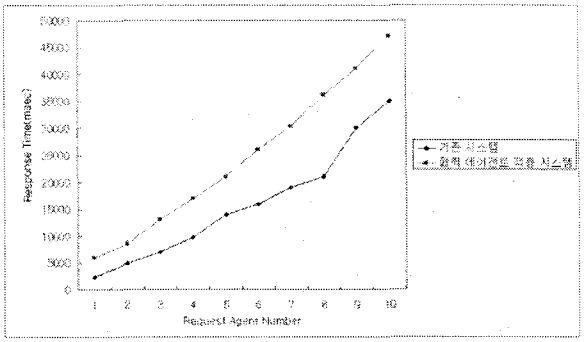

그림 6. 요청 에이전트 수에 따른 응답시간 그래프

본 논문에서 적용하는 데이터 그리드 미들웨어의 통신 메커니즘은 에이전트를 이용하여 수행된다. 요청 에이전트가 요청을 할 경우 각 목표 노드에 요청에 대한 프로세스만을 복제하고 다른 노드로 이주하기 때문에 요청 에이전트의 응답 책임은 배제된다. 응답에 대한 책임은 복제된 프로세스에게 있으며 요청에 대한 정보나 데이터는 응답 에이전트를 통해 최초 요청 노드로 전달되게 된다. 따라서 요청 에이전트 수에 따른 응답시간 측정은 응답 에이전트가 최초 요청 노드에 도착이 완료되는 시간까지의 시간을 측정해야 한다. 그림 6은 요청 에이전트 수에 따른 응답시간을 나타내는 그래프이다. 기존 시스템에서는 요청 에이전트와 정보 에이전트사이에서의 조정시간이 무시되기 때문에 협력 에이전트를 적용시킨 시스템에 비해 비교적 빠른 응답시간을 나타낸다. 협력 에이전트를 적용시킨 시스템에서는 각 요청 에이전트가 도달할 때마다 로컬 노드의 자원정보를 추출하여 판단해야 하며, 대기하고 있는 큐에 저장된 요청 에이전트의

수행까지 마쳐야 하기 때문에 기존 시스템에 비해 늦은 응답시간이 측정되었다.

5. 결론 및 향후 연구

본 논문에서는 이 시스템에 적용시킬 협력 에이전트를 제안함으로써 기존의 시스템보다 효율적인 정보 활용을 할 수 있도록 하였다. 조정 에이전트를 통해 커뮤니티에 속해있는 노드의 자원 관리와 작업 분배로 인한 로컬 노드의 부하를 줄일 수 있다. 하지만 성능 평가를 통해 기존의 시스템보다 늦은 응답시간이 도출되었음을 확인하였다. 조정 에이전트의 조정과 관리로 인한 시간 손실은 효율적인 알고리즘 연구를 통하여 해결해야할 필요성이 있다. 또한 검색 서비스뿐만이 아닌 실시간 엑세스 관리를 할 수 있는 서비스를 제공할 수 있는 연구가 필요하다.

참고 문헌

[1] J. Jang and J. Choi, "A Java-based Agent Management System for Dynamic Invocation of Heterogeneous Agents", The 1999 International Conference on Artificial Intelligence, pp. 324-330, Las Vegas, 1999.
[2] Matthew Hart, Scott Jesse, "Oracle Database 10g: High Availability with RAC, Flashback & Data Guard", Oracle Press, 2005
[3] Maurizio Panti, Luca Spalazzi, Alberto Giretti, "A Case-Based Approach to Information Integration", The VLDB journel, 2000
[4] Bastin Tony Roy Savarimuthu, Maryam Purvis, "A collaborative multi-agent based worflow system", KES2004,
[5] P.Spyns, R. Meersman, and M. Jarrar, "Data modeling versus Intology engineering", SIGMOD Record, Vol 31, 2002
[6] Ray Gates,"Introduction to MED-Tutorial on ISO/IEC11179", Metedata Open Forum 2004, 2004
[7] ISO/IEC 11179, "Informaion technology - Specification and standardization of data elements" , 2003
[8] 노선택, 문석재, 엄영현, 국윤규, 정계동, 최영근, "XMDR을 이용한 데이터 그리드 미들웨어 설계", KCC2006, 2006
[9] 장영욱, 이광로, 민병의, "에이전트 기술", ETRI 전자통신동향분석 12권 6호, 1997
[10] 정의석 "상호운용성 확보를 위한 지식정보 표준화 동향", ITFIND 주간기술동향 제1230호, 2006
[11] 최중민, "Intelligent Agents Overview", 명지대 차세대전력연구센터 초청세미나, 2005