자원인자 기반 스케줄링 프레임워크†

(Resource Scheduling Framework based on Resource Parameter Graph)†

배재환*, 권성호**, 김덕수***, 이강우****
(Jae-Hwan Bae, Cheng-Hao Quan, Deok-Soo Kim, Kang-Woo Lee)

요 약 대규모 환경의 고성능 그리드 구현을 위해서는 기존 그리드 자원 스케줄링 파라다임이 갖는 성능 확장성 측면의 제한성을 극복할 수 있는 새로운 자원 스케줄링 프레임워크가가 요구된다. 본 연구에서는 자원 스케줄링 성능 최적화를 목표로 자원인자 그래프(Resource Parameter Graph), 자원인자 인덱스 트리(Resource Parameter Index Tree), 그리고 정적 자원 정보 리포지터리로 구성되는 자원인자 스케줄링 프레임워크를 제안한다. 자원인자 그래프는 자원간의 관계 및 자원의 계층적 구성을 나타낼 수 있는 자원표현 기법이며 이러한 표현을 기술하기 위한 XML 기반 자원정보 및 자원요청 기술 스키마를 설계하였다. 또한 자원인자 인덱스 트리는 자원 스케줄링의 자원탐색 및 자원할당, 상태정보 공지 등의 알고리즘의 효율적인 지원을 위한 메모리 기반 인덱스의 데이터 구조이다. 본 논문에서는 이러한 자원인자 스케줄링 프레임워크의 구성 내용에 대하여 기술한다.

핵심주제어: 자원 스케줄링, 자원 탐색

Abstract For the implementation of large scale GRID systems, the performance scalability in resource scheduling is clearly to be addressed. In this research, we analyzed existing scheduling frameworks from the viewpoint of the performance and propose a novel resource scheduling framework called resource parameter based scheduling. Proposed scheduling framework consists of three components. The first is the resource parameter graph that expresses resource information via inter-resource relation and the composition base on the hierarchical structure. The second component is the resource parameter tree to be used for the implementation of the memory-based index of resource information. The third component is the resource information repository which mostly consists of static data to be used for the general resource information services. This paper presents the details of the framework.

Key Words: Resource Scheduling, Resource Discovery

1. 서 론

현재 기초과학과 산업기술 연구는 고속 연산, 대량의 데이터 처리, 첨단 장비의 공유 및 분산 협업 등이 필수적으로 수반된다. 이러한 요구를 만족시켜줄 수 있는 수단으로써 Grid는 지리적으로 분산된 고성능 컴퓨터, 대용량 DB 및 첨단 장비 등의 정보통신 자원을 고속 네트워크로 연동하여 상호 공유할 수 있도록 하는 컴퓨팅 방식이다[1,2,3]. 이러한 대규모 그리드 환경에서 자원은 다양한 형태를 갖으며 광범위한 위치, 그리고 다양한 관리정책에 준하여 관리되는 기기들의 집합이다.

† 본 연구는 2001, 2002년 정보통신기초기술연구지원사업에 의하여 지원받았음.
* 탐라대학교 출판미디어 학부 전임강사
** 대구대학교 정보통신공학과 박사과정
*** 계명문화대학 문헌정보과 겸임교수
**** 동국대학교 정보통신과 교수

등록된 자원들은 그 가용성 측면에서 동적으로 변동하는 특성을 갖으며, 또한 신규 자원의 추가를 통한 가용한 자원의 증가도 발생할 수 있다. 자원 스케줄링(Resource Scheduling)은 어플리케이션 또는 사용자가 자원에 대한 일반적 정보와 가용성(Availability) 정보 등을 제공하는 제반 관련 서비스로 구성된다. 이러한 자원 스케줄링과 관련된 서비스는 크게 다음과 같이 분류할 수 있다.

첫째, 자원정보 서비스로서 그리드 내의 자원들에 대한 통계, 특정 자원에 대한 사양 및 현재 상태 등에 관한 정보를 제공하는 서비스이다. 이러한 서비스는 관리자 또는 사용자가 특정 관리 작업을 위하여 자원에 대한 일반적인 정보가 필요할 경우에 사용하게 된다. 이 서비스의 특징은 일반적으로 특정 자원을 지정한 후 해당 자원에 대한 필요한 정보를 질의와 응답 형태로 갖는다는 점과 이 서비스의 결과로서 그리드 내의 자원의 할당 등과 같은 정보는 변경되지 않는다는 점이다. 따라서 자원정보 서비스는 기본적으로 자원에 대한 목록을 구축한 후 목록의 각 구성요소별로 해당 정보를 저장하는 논리적 구조를 도입하여 효율적으로 구현할 수 있다[4,5,6,7]. 둘째, 자원탐색 서비스에서 자원탐색은 응용프로그램에서 필요한 자원들의 목록과 사양에 준하여 현재 가용한 자원목록에 대하여 질의를 수행하고 그 결과로서 해당 자원들을 가용성 여부 및 가용한 경우 해당 자원들의 목록을 응답으로 제공하는 서비스이다. 자원탐색 서비스는 원칙적으로 탐색된 자원들을 해당 응용프로그램에서 할당하여 그 구동에 사용할 목적으로 제공된다. 따라서 자원탐색 서비스에 있어서 이후 자원할당 시에 동일한 자원탐색을 재수행하지 않도록 하는 메카니즘이 요구된다[8]. 셋째, 자원 상태정보 공지 서비스는 자원 할당을 위해서는 가용성 여부, 가용한 용량 등과 같이 자원 할당과 관련되어 동적으로 변화하는 자원 상태정보의 추적(Tracking)이 요구된다. 이러한 정보는 자원으로부터 주기적으로 또는 요청에 준하여 스케줄러 엔티티(Entity)에 제공되어야 한다. 이러한 정보 제공을 자원정보 공지(Resource Information Publication)라하며 이를 위해서는 실시간으로 상태정보를 갱신할 수 있는 효율적인 저장구조의 설계가 수반되어야 한다. 넷째, 자원할당 서비스에서 자원 할당은 응용프로그램에서 필요한 자원들의 목록과 사양에 준하여 현재 가용한 자원목록에 대한 자원탐색을 마친 후에 응용프로그램의 수행을 위하여 해당 자원들을 할당받는 서비스로서 정의된다. 자원탐색과 자원할당은 순서적 개념에서 이해되어야 하며 이에 따른 상호 연동 관계가 정립되어 그 구현에 반영되어야 한다[9,10,11,12,13,14].

앞에서 기술한 그리드 자원관리에 있어 도출된 주요 문제, 즉 자원 제공 기관의 자원관리 정책과의 호환성 유지 (Site autonomy), 각 사이트마다 서로 다른 지역 자원관리자 사용에 따른 문제 (Heterogeneous Substrate), 자원사용정책의 확장성 (Policy Extensibility) 보장, 자원 동시할당 (Co-allocation), 동적 변동에 따른 온라인 제어 (Online Control) 보장을 지적하였다. 기존의 자원 스케줄링에 관련된 연구개발은 이러한 문제의 해결에 초점을 두고 수행되어 왔다[Fos99]. 하지만 이러한 문제들은 전반적으로 자원 스케줄링의 기능적 측면의 요구사항으로서 성능 측면의 요구사항을 포함하고 있지 않다. 그러므로 기존의 연구개발은 그리드의 자원관리의 기능 및 구현가능성(Fuction & Feasibility)에 초점을 두고 있다고 평가할 수 있다.

현재 그리드에 관한 연구개발은 그리드 미들웨어의 프로토타입 개발을 완료한 후 이를 기반으로 하여 다양한 응용 분야의 그리드를 구축하는 단계에 있다는 점을 앞에서 지적한 바 있다. 이에 따라 기존 개발된 그리드 미들웨어의 구현성, 복잡성, 적합성 등에 대한 구체적인 연구결과가 도출되기 시작하고 있다. 이러한 연구결과들은 현재의 그리드 미들웨어가 각종 응용 그리드 구축을 위한 필요한 제반 기능들을 제공하고 있다는 사실을 보여주며 또한 그 구현성(Feasibility), 복잡성(Complexity)에 대한 비교적 충분한 이해를 제공한다. 하지만 실용화 그리드 환경 하에서의 성능에 대한 연구개발은 아직 제대로 이루어지지 않고 있다.

현재 진행되고 있는 각종 그리드 시험시스템 구축 작업과 그리드 구조의 표준화 작업이 완료되는 시점에는 향후 1, 2년 후에는 성능 측면에서의 최적화에 대한 요구 및 관심이 증대될 것으로 예상된다. 따라서 대규모 그리드 구축을 위해서는 기존 자원 스케줄링 구조가 갖는 다음과 같은 주요 문제점을 필수적으로 해결하여야 한다.

먼저 자원정보를 관리하는 그리드 정보 서비스(GIS : Grid Information Service)가 운영체제 버전과 같은 자원에 대한 정적(Static) 정보와 서버의 부하와 같은 동적(Dynamic) 정보를 X.500 DIT(Directory Information Tree) 기반의 디렉토리에 함께 저장·관리하고 있다. 이러한 설계는 자원정보 공지 및 스케줄

링 기능에 있어 자원의 동적 변화를 갱신하는 부문에 있어 많은 부하를 초래하게 되므로 실용화 환경에서는 성능저하가 수반될 가능성이 매우 높다. 한편 동시 자원요청이 매우 많은 상황에서는 성능 확장성(Performance Scalability)에 제한을 받게 된다.

한편 디렉토리 기반의 자원정보 관리는 자원탐색의 경우 주어진 요청에 대하여 쿼리 기반으로 가용한 자원을 찾게 되는데 이러한 경우 최적화 탐색 알고리즘의 구현이 매우 어렵다 또한 가용 자원을 찾은 경우 자원요청을 한 사용자 프로그램(User Application)이 가용 자원에 대하여 동시할당자(Co-allocator)에게 할당을 요구하게 되는데 이러한 방식은 요청된 자원 중에서 하나라도 할당이 거절될 경우 자원브로커(Resource broker)에게 처음부터 다시 자원요청을 시작하여 동일한 절차를 반복해야 한다. 따라서 자원탐색에 있어 백트랙킹(Backtracking)과 같은 효과적인 알고리즘의 적용이 가능하지 않은 문제점을 갖는다.

2. 자원인자 스케줄링 프레임워크

자원공유라는 개념을 바탕으로 하는 그리드에 있어 자원 스케줄링은 그 구동의 중심축으로서의 역할을 갖으며 미들웨어의 핵심 기능으로서의 위치를 점한다. 자원 스케줄링의 핵심이 되는 요소로서 자원탐색, 자원할당 등의 알고리즘으로 구성된다. 그러므로 자원 스케줄링은 전역/지역 자원관리자, 사용자 자원 요청 기제(Request mechanism) 그리드 정보 서비스 등 그리드의 미들웨어의 많은 구성 요소들과 상호 밀접한 관계를 갖는다. 따라서 효과적인 자원 스케줄링의 구현은 궁극적으로는 이러한 각 요소에 대한 충분한 이해를 필요로 하고 나아가 상호 동작 및 성능에 영향을 미치는 요소들을 고려해야 하기 때문에 함으로 매우 복잡하고 난해한 문제라 할 수 있다.

자원 스케줄링의 제반 알고리즘은 상호 독립적인 역할을 갖지만 그리드의 상태정보를 매개로 하여 상호 긴밀하게 연결되어 있으므로 상태정보는 위의 두 알고리즘에 있어 매우 중요하다. 즉 상태정보에 대한 자료구조 표현은 즉 그리드 상에서 자원상태 공지 및 자원 선택의 작업의 효율에 매우 중요한 영향을 미친다. 그러므로 자원 스케줄링의 제반 알고리즘에 개발에 있어 먼저 자원 스케줄링 프레임워크로서 사용할 상태정보의 자료구조 표현 및 표현된 자료구조에 대하여 구현된 알고리즘들의 연산 및 동작 내용에 대한 효율성 분석이 요구된다.

앞에서 사용자 또는 응용프로그램 측면의 자원관리 및 스케줄링 서비스는 크게 자원정보서비스, 자원탐색 서비스, 자원할당서비스로 구분될 수 있다는 점과 이러한 서비스들은 자원 정보의 갱신 여부에 따른 다양한 요구를 수반한다는 점을 살펴보았다. 이러한 점을 고려하여 본 연구에서는 자원정보를 크게 두 개의 부분으로 구분해서 독립적인 관리정책을 적용한다.

- 스케줄링 자원 정보 베이스(SRIB : Schduling Resource Information Base)는 자원탐색, 자원할당, 그리고 자원공지 작업에 활용되는 제반 자원정보들로 구성된다. SRIB는 자원인자 인덱스 트리로 구현되어 관리되며 주로 각 작업의 결과 해당 내용이 변경되는 동적 변경의 특성을 갖는다.
- 비스케줄링 자원 정보 베이스(NSRIB : Non-Scheduling Resource Information Base)는 자원정보 서비스에 주로 활용되는 제반 자원정보들로 구성된다. NSRIB에 의하여 관리되는 자원정보는 주로 자원의 할당 등의 제반 작업 과정에 변경되지 않는 정적 데이터라는 특성을 갖는다. 특히 스케줄링에는 사용되지 않는 점을 주의하여야 한다.

스케줄링 정보와 비스케줄링 정보를 구분하여 별도의 저장구조로 구현 및 관리를 수행하는 것은 타 연구와 구별되는 대표적인 특징 중의 하나이다. 이러한 특징은 크게 다음과 같은 장점을 갖는다.

자원 스케줄링(Resource Scheduling)과 자원정보 서비스(Resource Information Service)를 분리시켜 자원 스케줄링과 자원정보서비스를 독립적인 모듈로서 해당 모듈에 필요한 차별화된 최적화 구현을 추구할 수 있다.

정적인 자원정보를 독립적으로 관리함으로써 스케줄러에 필요한 자원정보 만을 관리함으로써 스케줄링에 필요한 자원정보의 양을 감소시켜 전반적인 성능향상을 도모할 수 있다. 위와 같은 논의에 기초하는 자원인자 스케줄링 프레임워크는 크게 다음과 같은 세 개의 엔티티를 그 주된 구성요소로 갖는다.

- 자원인자그래프는 지역자원관리자(LRM : Local Resource Manager)로부터 등록된 자원에 대한 정보를 조합하여 집합적으로 나타내는 역할을 갖는다. 독립자원의 구성과 자원간의 상호관계를 나타내는 표현모델로서 자원요청, 자원등록, 그리고 상태정보의 공지에 사용된다.
- 자원인자 인덱스 트리는 스케줄링 자원 정보 베이스(SRIB)의 저장모델로서 등록된 자원의 관리 및 자원탐색,

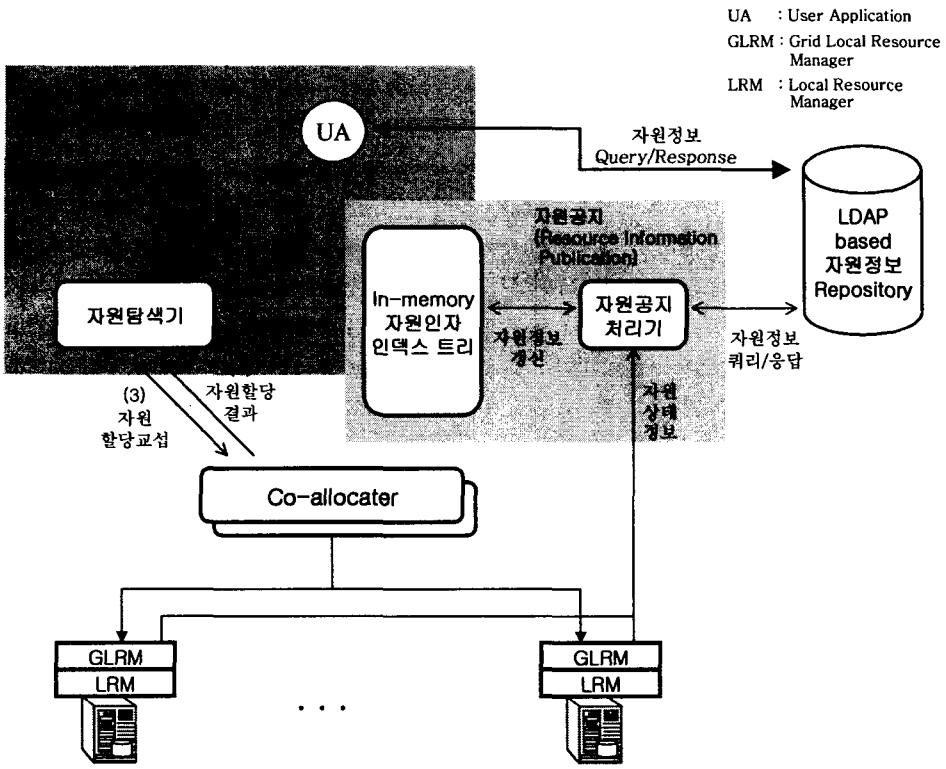


그림 1. 자원인자 자원 스케줄링 프레임워크

자원 정보 갱신 등의 사용자 요청에 이용될 수 있는 내부 메모리 기반 저장구조로서 자원에 대한 탐색 및 제반 연산의 효율성을 고려한 인덱스로서 역할을 갖는다. 이는 아래와 같은 특성을 가진다. 첫째, 자원 정보의 추가 및 변경에 대한 연산의 효율성 고려하고 둘째, 자원에 대한 탐색 및 할당 등과 같은 연산(Operation)에 대한 효율성 고려하고 셋째, 자원인자 인덱스 트리를 기반으로 구현한다.

- 자원정보 리포지터리는 비스케줄링 자원 정보 베이스(NSRIB)를 저장 및 관리하기 위하여 도입되었으며 분산 디렉토리를 기반으로 구현된다.

2.1. 스케줄링 아키텍처

앞에서 설명한 바와 같이 자원인자 스케줄링 프레임워크는 크게 자원인자그래프 기반 자원기술(Resource Description), 자원인자트리 기반 메모리 자원정보 인덱스, 그리고 분산 디렉토리 기반 그리드 자원정보 리포지터리의 세 개의 요소로 구성된다. (그림 1)은 이 요소들을 기반으로 하는 자원 스케줄러의 개념적 구성을 보여주며 각 요소의 역할은 다음과 같다.

자원정보기술언어는 자원요청 및 등록을 위한 자원 사양을 표현하기 위한 언어로서 준구조적 방식인 XML을 채택하여 설계하였다.

자원정보 리포지터리는 자원등록 시에 사용자로부터 등록된 자원에 대한 정보를 집합적으로 모아놓은 저장소로서 LDAP를 사용하여 구현을 하였다. 자원정보 리포지터는 기존 그릭드 정보 서비스 구축에 사용하는 디렉토리와 유사한 역할을 갖으나 기본적으로 자원에 대한 정적 정보(Static Information)를 주로 저장하고 사용자 또는 어프리케이션의 쿼리를 처리하는 하다는 점에서 차이를 갖는다. 특히 자원 스케줄링과 관련해서는 전혀 특별한 부가 정보를 지니고 않다.

자원인자 인덱스 트리는 자원인자 인덱스 트리는 효율적인 자원탐색을 위하여 메모리 기반 인덱스이다. 인덱스는 자원의 속성들 중에서 지정된 속성들에 대해서만 인덱싱(Indexing) 작업이 수행된다. 4장에서 살펴본 바와같이 자원의 온라인 추가 및 삭제가 기존 인덱스 트리를 변형시키지 않고 가능하며 다차원 이진탐색 트리 구조는 자원탐색과 자원 등록 및 삭제 등의 동작을 매우 효율적으로 지원할 수 있는 장점을 갖는다. 따라서 자원공지와 자원탐색이 본 자원인자 인덱스 트리 상에서 동시에 수행될 수 있도록 설계되었다.

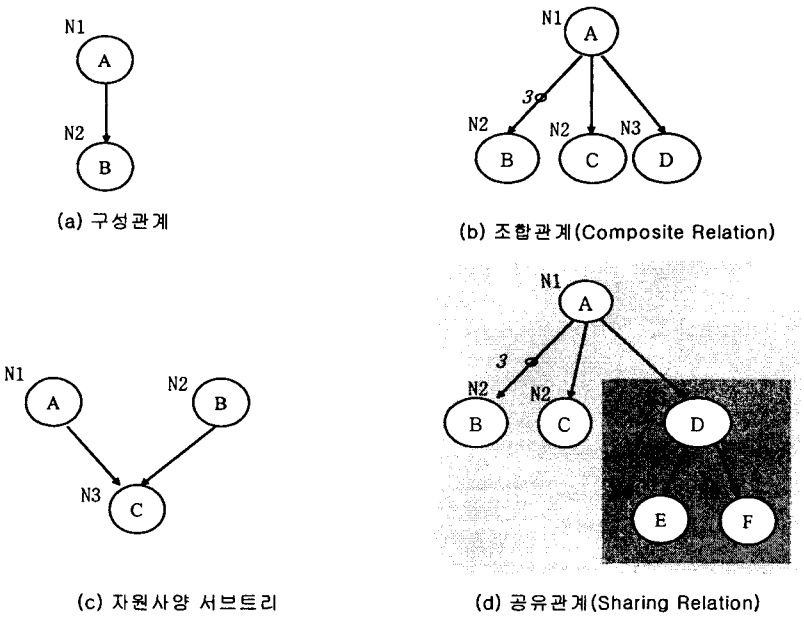


그림 2. 자원인자그래프 Semantics 설명 사례

위의 기반 요소에 준하여 자원 스케줄링의 제반 기능이 수행되며 이러한 기능적 측면의 소프트웨어 모듈은 다음과 같다.

- 자원탐색기는 응용프로그램으로부터 자원요청을 받아서 자원인자 인덱스 트리로부터 요청된 자원의 가용성 여부를 결정하는 기능과 자원탐색 처리하는 기능과 원할당 요청에 대하여 응용프로그램과 동시할당기(Co-allocator) 사이에 중계 역할의 기능을 갖는다. 그 세부 내용은 다음과 같다. 사용자 자원탐색 요청에 대하여 자원인자 인덱스 트리로부터 자원의 가용성 여부와 가용한 경우 가용 자원집합을 사용자에게 보내준다. 이때 자원탐색은 단일자원 또는 다중자원 요청이 가능하다. 탐색을 요청한 자원의 가용하지 않을 경우, 어플리케이션은 요청 내용을 수정하여 신규로 탐색을 요청하거나 또는 탐색을 중단한다 탐색 요청한 자원들이 가용할 경우에 어플리케이션은 탐색된 자원의 할당을 자원탐색기에 요청한다. 자원탐색기는 동시할당기(Co-allocator)에 자원할당을 요청하게 된다. 자원할당을 결과는 다시 자원탐색기에 보내지며 이를 다시 응용프로그램에 전달하게 된다. 여기서 자원탐색기는 자원할당에 있어 응용프로그램과 동시할당기 사이의 중계 역할을 하는데 자원탐색기는 이러한 중계를 통하여 얻은 할당정보를 자원인자 인덱스 트리에 할당정보를 갱신하여 추후 자원탐색에 반영될 수 있도록 한다.
- 자원정보공지 처리기는 자원의 상태를 동적으로 변화시키는 모듈로서 지역자원관리자(Local Resource Manager)로부터 자원의 상태정보를 받아서 자원인자 인덱스 트리에 그 내용을 갱신시킨다.
- 자원정보 리포지터리 서비스 처리기는 자원에 대한 정적(Static) 또는 준정적(Quasi-Static) 정보를 저장하는 LDAP 기반의 자원정보 리포지터리(Repository)는 사용자 또는 응용프로그램에게 자원에 대한 정보서비스를 제공하는 기능을 갖는다. 또한 리포지터리 정보 중에서 자원이자 인덱스 트리의 인덱싱 속성값들은 자원인자 인덱스 트리의 초기화 데이터로 사용된다. 또한 자원인자 인덱스 트리에서 주기적으로 스케줄링과 관련되어 변경된 내용 중에서 필요한 경우 보관(Backup)을 위하여 자원정보 리포지터리에 저장된다. 이 자원정보 리포지터리 서비스 처리기는 사용자 또는 응용프로그램으로부터의 질의와 해당 응답을 제공하는 동작 방식으로 서비스를 제공하며 자원인자 인덱스트리 서비스 모듈과 연동되어 동작한다.

기존 그리드 정보 서비스 구축에 사용하는 디렉토리와 유사한 역할을 갖으나 기본적으로 자원에 대한 정적 정보(Static Information)를 주로 저장하고 사용자 또는 응용프로그램의 쿼리를 처리하다는 점에서 차이를

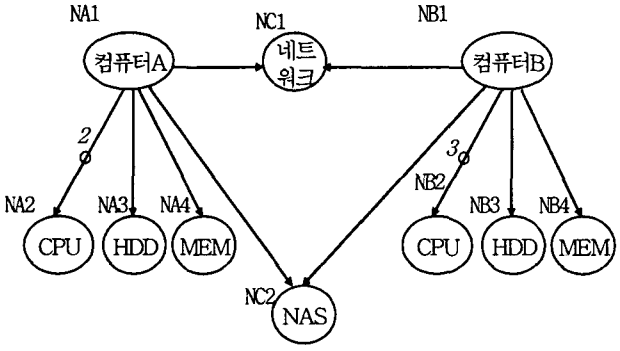


그림 3. 자원인자그래프 Semantics 설명 사례

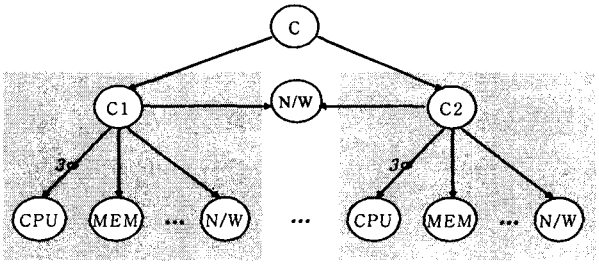


그림 4. 자원인자그래프 사례(2)

갖는다. 자원 스케줄링과 관련해서는 전혀 특별한 부가 정보를 지니고 않다.

3. 자원인자그래프(RPG : Resource Parameter Graph)

자원인자그래프란 각 자원을 노드로 갖으며 각 노드에는 자원의 사양과 연결 네트워크의 용량 등과 같은 정적 자원정보와 시스템 부하 등과 같이 동적으로 변동하는 상태 정보들의 인수 값이 속성(Attribute)으로 저장된다. 한편 그래프의 가지(Edge)는 자원간의 상호 관계(Relation)가 존재함을 나타내며 그 크기를 가지(Edge)의 가중치(Weight)로 나타낸다. 가지는 물리적 실체(Entity)를 나타낼 수 있으며 예를 들어서 연결된 네트워크의 용량(Bandwidth)에 따른 연결 상태를 표시할 수 있다. 또한 자원인자그래프의 가지는 물리적인 아닌 논리적 실체(Entity)를 나타낼 수도 있다. 예를 들면 서버를 나타내는 노드에서 상태정보를 공지해야하는 대상노드들간의 관계를 가지(Edge)로 표현할 수 있다.

보다 구체적으로 설명하면, 자언인자그래프(G)는 가중치를 갖으며 방향성을 갖는 그래프(Weighted and Directed Graph)로서 G={N,E}로 나타내며 여기서 (N)은 노드의 집합이며 E는 방향성을 같는 가지(Edge)의 집합이며 각 가지는 가중치(Weight)를 갖는다. 주어진 가지 E(N1,N2)는 노드 N1과 N2를 연결하며 N1으로부터 N2로 향하는 방향성을 갖는다. 이때 N1을 편의상 트리(Tree)의 용어(Terminology)를 사용하여 N2의 Parent 노드라고 하고 N2를 N1의 Child 노드라고 한다.

노드와 가지에 대하여 좀더 살펴보면 먼저 노드(Node)는 자원을 나타낸다. 가지(Edge)는 자원과 자원간의 관계(Relation)를 나타낸다. Parent 노드와 Child 노드 간에 Child 노드는 Parent 노드의 구성요소가 됨을 나타낸다. 즉 Child 노드에 의하여 표현되는 자원은 Parent 노드에 의하여 표현되는 자원의 구성요소가 된다. 가지에는 정수값을 갖는 가중치가 설정되며 이 가중치는 Child 노드의 Instance의 개수를 나타내다. 즉 동일한 Child 노드가 여러 개 존재할 경우에 그 개수를 가지의 가중치로 나타내며 Child 노드는 한개만 존재하게 된다. 단 가중치가 1인 경우는 디폴트 값으로서 자원인자그래프에는 나타내지 않고 생략할 수 있다.

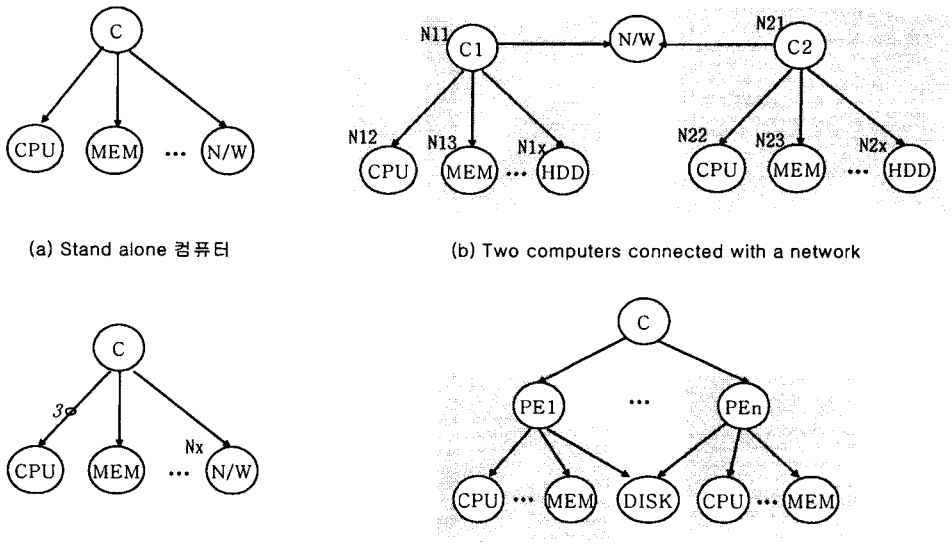
위와 같은 자원인자그래프의 노드 및 가지에 대한 정의에 준하여 아래와 같은 용어 및 개념이 정의된다.

개념 1 : 구성관계 (Container-Component Relation)

주어진 노드 N1과 N2를 연결하며 N1에서 N2로 방향을 갖는 가지 E1(N1,N2)에 대하여 노드 N1에 대한 자원을 R1, 노드 N2에 대한 자원 R2라고 할 때 R1과 R2는 구성관계에 있다고 하며 이때 R1은 컨테이너(Container) 자원, R2는 컴포넌트(Component) 자원이라고 부르며 컴포넌트 R2는 컨테이너 R1의 구성요소가 된다. 이 구성관계는 자원인자그래프 노드 측면에서는 "N1 → N2"로 표시하며 자원 측면에서는 "R1 ∋ R2"로 표시한다. 위 (그림 2.a)는 두 개의 자원 A와 B가 상호 구성관계(A → B)를 형성하고 있는 사례로 볼 수 있으며 자원 B는 자원 A의 컴포넌트이며 자원 A는 자원 B의 컨테이너임을 나타낸다.

개념 2 : 조합관계 (Composition Relation)

주어진 노드 N에 대한 자원은 N의 모든 자식노드들의 집합으로 나타나는 자원으로 구성된다. 이를 조합관계로 정의한다. 예를 들어 노드 N이 M개의 자식노드를 갖고 있으며 각 i번째 자식노드를 NCi(0≤i≤m)라



(a) Stand alone 컴퓨터

(b) Two computers connected with a network

(c) SMP(Shared Memory Multiprocessor

(d) Distributed memory multiprocessor

그림 5. 자원인자그래프 사례(1)

고 하면 그 조합관계는 아래와 같이 표시하며 Wi는 컴포넌트의 개수를 나타낸다.

N = NC1w1 o NC2w2 o ... o NCmwm

(그림 2.b)는 자원 A가 3 종류의 컴포넌트 자원으로 구성되고 있음을 볼 수 있으며 "A = B3 o C o D" 관계로 표현할 수 있다. 특히 여기서 자원 A의 컴포넌트인 B는 가지의 가중치가 3 이므로 3 개의 동일한 자원 B가 A의 컴포넌트로 존재하고 있음을 알 수 있다.

개념 3 : 자원공유관계 (Sharing Relation)

주어진 노드 N1, N2, N 간에 "*N1 → N*" 과 "*N2 → N*" 의 관계가 성립할 경우에 노드 N은 N1과 N2에 의하여 공유되는 컴포넌트로 정의된다. 위의 (그림 2.c)에서 N3는 자원 N1과 N2에 의하여 공유되는 컴포넌트임을 알 수 있다.

개념 4 : 자원사양 서브트리 (Resource Specification Subtree)

주어진 노드 N은 자원을 나타내며 이 자원의 세부 구성 사양은 N을 루트(Root)로 하는 서브트리로 표현된다. 이 서브트리를 노드의 자원사양(Resource Specification)이라고 하며 Subtree(N)으로 표현한다. (그림 2.d)에서 자원 A의 상세사양은 노드 N1을 Root로 하는 서브트리, 즉 옅은 회색조로 표시한 Subtree(N1)로 구성된다. 또한 자원 D의 상세사양은 노드 N3를 Root로 하는 서브트리, 즉 진한 회색조로 표시한 부분인 Subtree(N3)로 구성된다.

위의 개념들에 대한 보다 분명한 이해를 위하여 (그림 3)을 사례로 들어 설명한다. (그림 3)은 네트워크로 연결된 두 개의 시스템에 대한 자원인자그래프를 나타내고 있다. 이 자원인자그래프는 10개의 노드로 구성된다.

위 그림에서 두 개의 컴퓨터는 각각 NA1과 NA2 노드가 나타내고 있으며 구성관계(Composition Relation) 정의에 준하여 각 시스템의 구성요소는 아래와 같다.

컴퓨터 A : NA1 노드가 5개의 Child 노드(NA2, NA3, NA4, NC1, NC2)에 대한 컨테이너 노드로서 아래와 같은 컴포넌트 조합으로 표현된다.

NA1 = NA2 o NA3 o NA4 o NC1 o NC2

한편 컴퓨터 A에 대한 자원사양 서브트리는 NA1을 Root 노드로 하고 5개의 Child 노드(NA2, NA3, NA4, NC1, NC2)로 구성되는 트리로서 정의된다.

컴퓨터 B : NB1 노드가 5개의 Child 노드(NB2, NB3, NB4, NC1, NC2)에 대한 컨테이너 노드로서 아래와 같은 컴포넌트 조합으로 표현됩니다.

NB1 = NB2 o NB3 o NB4 o NC1 o NC2

한편 컴퓨터 B에 대한 자원사양 서브트리는 NA1을 Root 노드로 하고 5개의 Child 노드(NB2, NB3, NB4, NC1, NC2)로 구성되는 트리로서 정의된다. 위의 자원인자그래프에서 NC1과 NC2는 컴퓨터 A와 컴퓨터 B에 의하여 공유되는 자원을 나타내는 노드들로서 컴퓨터 A와 컴퓨터 B가 동일 네트워크에 연결되어 있으며 네트워크 스토리지를 공유하고 있음을 나타낸다.

위에서 자원인자그래프에서 노드는 자원을 나타낸다는 점을 보았다. 조합관계위에(Composition Relation)과

자원사양(Resource Specification) 서브트리의 정의에 준하여 자원인자그래프에서는 노드는 아래와 같은 두 가지 유형이 존재한다.

- 조합노드(Composite Node) : Child 노드를 갖는 노드로서 세부 컴포넌트 자원들의 집합으로 구성되며 그 상세 사양은 그 자원사양 서브트리로 표현된다. 조합노드는 자원사양 서브트리의 Non-leaf 노드가 된다.
- 단일노드(Simple Node) : Child 노드를 갖지 않는 노드로서 세부컴포넌트자원의 구성으로 표시되지 않는 자원을 나타낸다. 단일노드는 일반적으로 자원사양을 나타내는 모든 서브트리의 Leaf 노드가 된다.

위에서 살펴본 자원인자그래프의 구성에 준하여 자언인자그래프는 다음과 같은 특징을 갖는다. 먼저 자원인자그래프는 계층적 구조를 갖는 그래프이며 자원인자그래프에는 Circle이 존재하지 않는다. 그리고 자원인자그래프는 AND-OR Relation을 표현한다.

(그림 4)와 (그림 5)는 컴퓨팅 자원을 자원인자그래프를 활용하여 표현한 구체적인 사례를 보여 준다. (그림 4.a)는 CPU와 메모리, 그리고 네트워크 등으로 구성된 단일 컴퓨터에 대한 자원인자그래프이다. (그림 4.b)는 지역 네트워크로 연결된 2개의 컴퓨터에 대한 자원인자그래프이다. 각 컴퓨터는 CPU와 메모리, 그리고 하드디스크 등으로 구성되어 있다는 것을 알 수 있다. (그림 4.c)는 3개의 CPU로 구성된 공유메모리 병렬컴퓨터를 나타내는 자원인자그래프이다. 마지막으로 (그림 4.d)는분산메모리 기반 병렬컴퓨터를 나타내는 자원인자그래프이다. 이 시스템은 N 개의 PE(Processing Element)로 구성되어 있음을 알 수 있다. (그림 5)는 네트워크로 연결되어 있는 여러 대의 공유메모리 기반 병렬 컴퓨터에 대한 자원인자그래프로서 이러한 구조는 병렬컴퓨터가 논리적으로 하나의 클러스터 시스템으로 구성된다는 점을 알 수 있다.

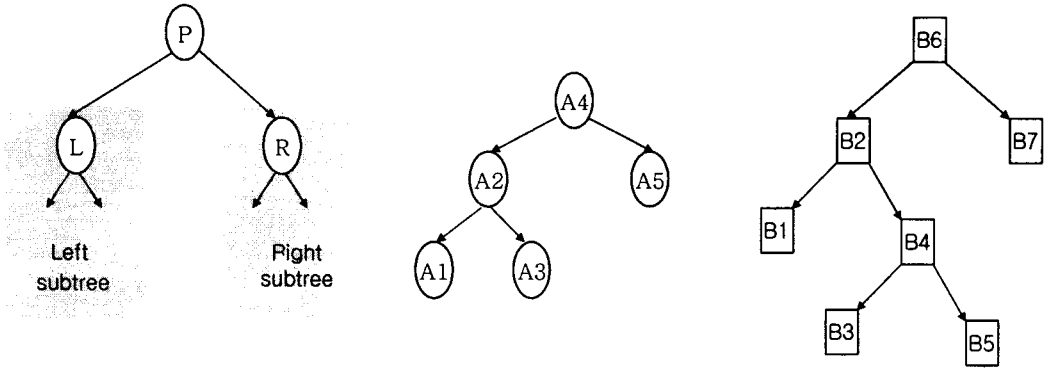
자원 정보를 위의 자원인자그래프로 표현할 경우 다양한 장점을 얻을 수 있다. 먼저 자원간의 관계가 그래프의 가지로 표현되며 또한 자원의 세부 구성이 계층적으로 표현되므로 자원공지 및 탐색을 위한 제반 정보를 충분히 제공한다. 따라서 이러한 정보의 제공은 효과적인 자원 스케줄링을 위한 저장구조, 자원정보 및 상태정보를 도출할 수 있도록 한다. 본 연구에서 자원인자그래프는 XML 기반으로 표현되며 각 자원정보 기술, 자원인자 인덱스 트리의 생성 등에 활용된다.

4. 자원인자 인덱스 트리(Resource Parameter Index Tree)

그리드 시스템의 특징인 대규모 자원과 빈번한 데이터 갱신, 그리고 신속한 자원탐색을 위해서는 자원정보에 대한 효율적인 인덱스 생성이 필수적으로 요구된다. 하지만 앞에서 디렉토리 기반의 기존 자원 스케줄링 구조는 실시간 자원공지 및 탐색에 기본적으로 요구되는 효율적인 동적 인덱스의 생성에 대한 지원이 결여되어 있다는 점을 보았다. 본 소절에서는 효율적인 동적 인덱스 구조인 자원인자 인덱스 트리의 개념, 구현방법에 대하여 설명한다.

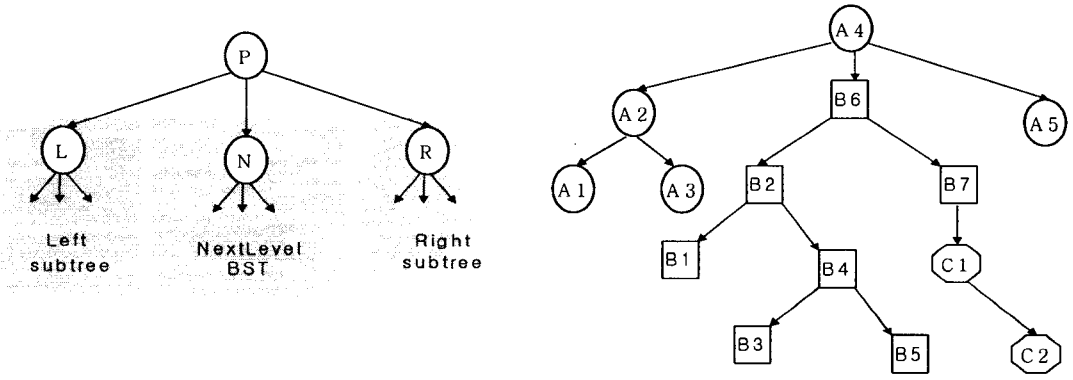
일반적으로 인덱스 생성을 위해서는 해쉬 테이블을 이용하는 방식을 고려해 볼 수 있다. 이 방식은 인덱스 필드 값을 입력으로 하여 해쉬 함수 값을 계산하여 해당 해쉬 엔트리를 지정한다. 이 방식은 메모리 상에 해쉬 테이블을 상주 시킬 수 있으므로 메모리 활용도를 제고시킬 수 있는 장점이 있다. 또한 해쉬 데이터 구조 구조의 특성에 기인하여 주어진 입력 값에 대한 해당 해쉬 엔트리의 탐색이 매우 효율적이라는 장점을 갖는다[17]. 하지만 해쉬 테이블을 이용한 인덱스 생성 및 관리 방법은 성능과 유연성 측면에서 문제점을 갖는다. 먼저 계층적 구조의 멀티레벨 구조의 필드들에 대한 인덱스 생성은 많은 경우 해쉬 엔트리의 충돌(Collision)을 발생할 가능성이 높으며, 각 차원에 대한 해쉬 엔트리들을 각각 산출한 후 이들에 대한 조인(Join) 작업을 수행해야 하므로 적합하지 않다[18].

한편, 계층적 구조를 갖는 데이터들의 저장에 있어 매우 효율적인 구조인 트라이(Trie) 데이터 구조를 이용하는 방식 또한 고려해 볼 수 있다. 트라이 구조는 계층적 구조에서 각 차원의 데이터는 노드를 원소로 갖는 배열로 저장한다. 각 노드는 하나의 필드 값을 나타내며 해당 노드에 다음 차원의 데이터에 해당하는 노드의 배열을 연결시켜 전체적으로는 트리 형식의 구조를 갖는다. 즉 이 방식은 트리의 각 레벨은 특정 차원에 해당하는 필드 값을 대한 인덱스를 형성한다. 논리적으로 각 노드는 <[value, link]*>로 구성되는데 여기서 value는 필드 값이며 link는 자식노드들에 대한 포인터이다. 앞에서 지적한 바와 같이 트라이 구조는 트리 구조를 효과적으로 표현하므로 계층적 구조에 대하여 점진적 탐색(Incremental)이 가능하므로 매우 효율적인 방식이라 할 수 있다. 하지만 성능측면에서 살펴보면 각 노드 내의 주어진 값의 탐색은 기본적으로 순차적



(a) 기본 BST의 구성 (b) 기본 BST 사례 I (c) 기본 BST의 사례 II

그림 6. 이진탐색트리 구성 및 사례



(a) 멀티레벨 BST의 구성 (b) 멀티레벨 BST 사례

그림 7. 멀티레벨 이진탐색트리(MBST : Multi-level Binary Search Tree) 개념

비교가 요구되므로 O(n)의 작업으로 비효율적이다. 또 다른 문제점으로는 동적 인덱스 생성에 필수적으로 요구되는 엔트리의 추가·삭제가 트라이 구조에서는 형제 노드를 상대번지로 저장하기 때문에 추가가 현실적으로 가능하지 않은 단점을 가지고 있다[18].

위의 문제점을 해결할 수 있는 자원정보 인덱스 저장 구조로서 자원인자 인덱스 트리를 개발하였다. 자원인자 인덱스 트리는 메모리 기반 데이터 구조로서 멀티레벨 이진탐색트리(MBST : Multi-level Binary Search Tree)를 기반으로 구축된다. 멀티레벨 이진탐색트리는 이진탐색트리(Binary Search Tree)을 변형된 형태이다. N 레벨 이진탐색트리는 N개의 레벨로 구성되는 이진탐색트리이며 한 개의 레벨에 대한 이진탐색트리는 기본적인 이진탐색트리와 동일하다.

4.1. 기본 이진탐색트리 (Basic Binary Search Tree)

(그림 6.a)는 기본적인 이진탐색트리(Binary Search Tree)의 구성을 보여준다. 이진탐색트리의 각 노드 P는 2개의 자식노드, 즉 왼쪽자식 노드 L과 오른쪽 자식노드 R을 갖는다. 노드 X의 값을 X.value라고 할 때 이진탐색트리는 항상 아래의 관계가 성립한다.

$$L.value < P.value < R.value$$

위의 관계를 서브트리 레벨로 확장하면 왼쪽 서브트리와 오른쪽 서브트리의 노드들은 아래와 같은 관계가 성립하게 된다.

{X, Y | X ∈ left-subtree(P), Y ∈ right-subtree(P), X.value < P.value < Y.vaue}

표 1. 자원집합 사례

자원집합	O/S Type	CPU (Hz)	Memory (MByte)	Free Disk Space (Byte)	N/W (Mbps)
컴퓨터 1	2	1G	300	2G	20
컴퓨터 2	1	0.3G	200	0.4G	10
컴퓨터 3	1	0.3G	300	0.3G	20
컴퓨터 4	1	0.3G	400	0.8G	10
컴퓨터 5	1	0.5G	200	0.5G	20
컴퓨터 6	1	2G	300	10G	10
컴퓨터 7	1	1G	100	1.2G	100
컴퓨터 8	1	1G	400	1.4G	10
컴퓨터 9	1	1G	500	0.9G	10
컴퓨터10	1	2G	700	8G	10
컴퓨터11	3	0.5G	100	1.2G	100
컴퓨터12	3	0.5G	500	0.5G	200
컴퓨터13	3	0.5G	700	0.6G	200
컴퓨터14	3	0.5G	900	0.3G	20
컴퓨터15	3	0.5G	1000	1.8G	20
컴퓨터16	3	1G	700	3G	1544

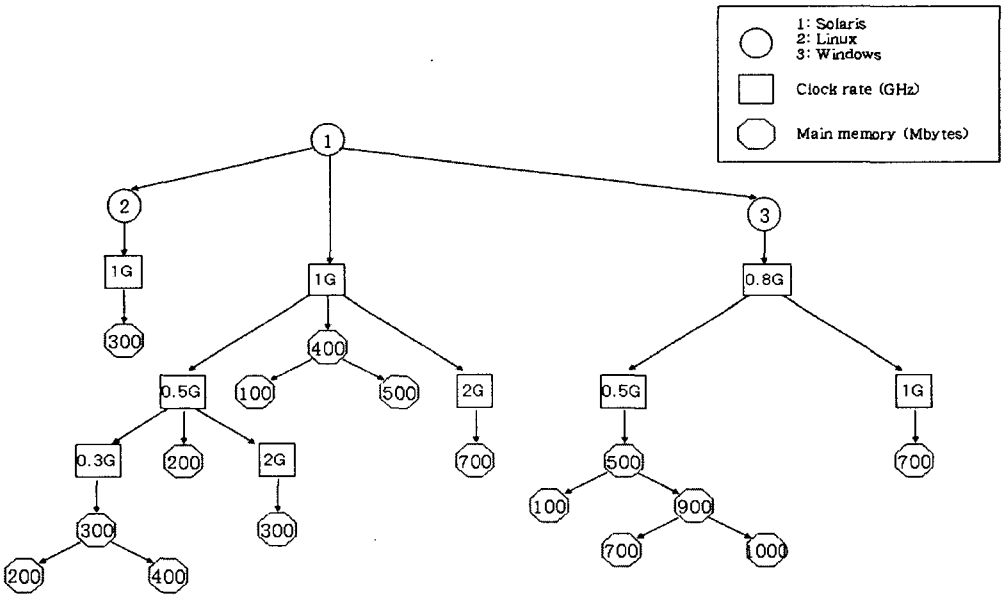


그림 8. 자원인자 인덱스 트리 사례

(그림 6.b)와 (그림 6.c)는 기본 이진탐색트리의 사례를 보여준다. 이진탐색트리의 성격에 준하여 이 사례들에서 노드들의 값은 각각 아래와 같은 관계가 성립하게 된다.

A1.value < A2.value < A3.value < A4.value < A5.value B1.value < B2.value < B3.value < B4.value < B5.value < B6.value < B7.vale

4.2. 멀티레벨 이진탐색트리 (Multi-Level Binary Search Tree)

앞에서 기본 이진탐색트리에 대하여 살펴보았다. 멀티레벨 이진탐색트리는 기본 이진탐색트리를 확장한 형태로서 기본 이진탐색트리를 하나의 차원으로 간주할 때 여러 개의 차원을 계층적으로 나타낼 수 있도록 구성한다.

(그림 7.a)는 멀티레벨 이진탐색트리(Binary Search Tree)의 구성을 보여준다. 멀티레벨 이진탐색트리의 각 노드 P는 세 개의 자식노드, L(Left), N(Next level),그리고 R(Right)을 갖는다. 노드 L과 R은 왼쪽 자식 노드 L과 오른쪽 자식노드 R로서 그 역할은 기본 이진탐색트리에서와 동일하다. 노드 L과 R 중간에 위치하는 자식노드인 N은 다음 레벨의 BST의 Root 노드의 역할을 갖는다. 즉 노드 P가 N 레벨 BST에 속해 있으면 N+1 레벨 BST의 루트 노드를 중간 자식 노드로 갖게 된다.

(그림 7.b)는 멀티레벨 BST의 사례를 보여준다. 이 사례는 3 개의 레벨로 구성된다. 이 사례에서 레벨 1의 노드의 집합은 L1()={A1, A2, A3, A4, A5}이며, fp벨 1의 노드 A4를 부모 노드로 갖는 레벨 2의 노드는 집합은 L2(A4)={B1, B2, B3, B4, B5, B6, B7}가 된다. 한편 레벨 2의 노드 B7을 부모 노드로 갖는 노드의 집합은 L3(B7)={C1, C2}가 된다.

주어진 BST를 구성하는 노드의 집합이 {A1, A2, A3}이며 그 노드값이 "A1.value < A2.value < A3.value" 의 관계를 갖을 때 이 BST를 [A1→A2→A3]으로 표시한다. 그러므로 위의 실례에서 세 개의 BST, 즉 [A1→A2→A3→A4→A5], {B1→B2→B3→B4→B5→B6→B7], 그리고 [C1→C2] 가 존재하는 것을 알 수 있다. 한편 멀티레벨 이진탐색트리에 속하는 BST 중에서 노드 X의 중간 노드를 루트로 하는 BST를 BST(X)로 표시한다. 따라서 위의 사례에서 BST(A4)={B1→B2→B3→B4→B5→B6→B7], BST(B7)=[C1→C2]가 된다.

앞으로 주어진 멀티레벨 이진탐색트리 M에서 주어진 노드 x와 y가 동일 BST에 속해 있을 경우, x ↔ y로 표시하고 주어진 BST S1과 S2가 동일한 레벨에 속해 있을 경우, S1 ⇔ S2로 표시한다. 위의 사례의 경우에는 A1 ↔ A2 ↔ A3 ↔ A4 ↔ A5, B1 ↔ B2 ↔ B3 ↔ B4 ↔ B5 ↔ B6 ↔ B7, C1 ↔ C2로 그 관계를 나타낼 수 있다.

한편 멀티레벨 이진탐색트리에서 레벨 N의 노드 X로부터 레벨 N+1의 노드 Y로의 경로는 X ⇒ Y로 표시하며 이때 Y는 노드 X의 Next-level BST에 속해 있어야 한다. 즉 Y ∈ BST(X)가 항상 성립해야 한다. 이러한 표시 형식에 준하여 (그림 7.b)의 MBST에서의 경로의 사례로서 "A4 ⇒ B7 ⇒ C1" 또는 "A4 ⇒ B7 ⇒ C2"를 들을 수 있다.

4.3. 자원인자 인덱스 트리 사례

자원인자 인덱스 트리는 그리드내의 모든 자원들에 대한 메모리 기반 인덱스로서 자원탐색, 자원상태 공지, 자원할당에서 사용된다. 자원인자 인덱스 트리는 먼저 모든 자원을 주어진 순서를 갖는 속성들의 집합에 대하여 생성한 인덱스로서의 역할을 가지므로 기본적으로 주어진 자원에 대한 신속한 탐색 지원을 가장 중요한 장점으로 갖는다. 나아가 온라인 자원의 등록 및 삭제, 그리고 유연한 자원탐색 및 할당 알고리즘의 구현성 제공 등의 장점을 지니고 있다.

자원인자 인덱스 트리는 앞에서 설명한 멀티레벨 이진탐색트리를 기반으로 구현되며 본 소절에서는 그 구성에 대하여 간략한 사례를 들어 설명한다. <표 1.>은 컴퓨팅 자원으로 구성된 자원집합의 사례를 보여준다. 자원의 사양은 운영체제, CPU 클럭 값, 메인 메모리 용량, 가용한 하드디스크 용량, 연결된 네트워크용량으로 구성되어 있다. 이러한 사례에 대하여 자원 스케줄링을 지원하기 위한 자원인자 인덱스 트리의 구성방법은 아래와 같다. 먼저 인덱싱 대상이 되는 집합인 인덱싱 속성집합을 정의한 후, 각 속성에 대하여 순서적으로 한 개의 레벨을 할당한 후 각 자원별로 순서적으로 하나씩 자원인자 인덱스 트리에 추가한다.

(그림 8)는 위의 사례에 대한 자원인자 인덱스 트리를 나타낸다. 위 그림에 의하면 <운영체제, CPU 클럭 크기, 메모리 용량> 3개의 인덱스 속성에 대하여 자원인자 인덱스 트리를 구성하였다. 본 사례는 자원인자 인덱스 트리의 개념적 구성에 초점을 두고 있어 실용적 측면의 구체성은 지니고 있지 않지만, <운영체제, CPU 클럭 크기, 메모리 용량>에 대한 조건을 입력으로 받아 해당하는 자원의 탐색이 매우 신속하게 이루어질 수 있다는 점을 시사하고 있다.

5. 결 론

대규모 환경의 고성능 그리드 구현을 위해서는 기존 그리드 자원 스케줄링 파라다임에 대한 성능확장성 측면의 제한성을 극복할 수 있는 자원 스케줄링 프레임워크가 요구된다. 본 연구에서는 자원 스케줄링 성능 최적화를 목표로 자원인자 그래프(Resource Parameter Graph), 자원인자 인덱스 트리(Resource Parameter Index Tree), 그리고 정적 자원 정보 리포지터리로 구성되는 자원인자 스케줄링 프레임워크를 제안하였다.

제안하는 자원인자 스케줄링 프레임워크는 대규모 GRID 그리드 구축을 위한 성능확장성에 초점을 둔 연구라는 점에서 구현성(Feasibility)에 초점을 둔 기존 그리드 자원 스케줄링에 관한 연구와 구별된다. 본 논문에서는 정성적인 분석을 통하여 자원인자 스케줄링 프레임워크는 자원 스케줄링 알고리즘의 구현에 있어 높은 성능과 유연성 보장의 장점을 갖는다는 점을 보였다. 따라서 향후 과제로서 분석모델 및 모의 수행을 기반으로 한 정성적 성능분석과 실용화를 위한 견고한 구현에 대한 구체적인 연구가 요망된다.

References

[1] I. Foster, C. Kesselman, S. Tuecke. The Anatomy of the Grid: Enabling scalable virtual organizations. Intl. Journal of Supercomputing Applications. (to appear) 2001.

[2] K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman. Gird Information Service for Distributed Resource Sharing. Proc. 10th IEEE International Symposium on High Performance Distributed Computer(HPDC-10), IEEE Press. 2001.

[3] I. Foster and C. Kesselman, "The grid: Blueprint for a New Computing Infrastructure", Morgan-Kaufmann, 1999.

[4] K. Czajkowski, at al, "grid Information Services for Distributed Resource Sharing", In Proc. of 10th IEEE International Symposium on High Performance Distributed(HDPC-10), 2001

[5] W. Dinda, B. Plale, "A Unified Relational Approach to grid Information Services", grid Forum Informational Draft GWD-GIS-012-1, 2001

[6] W. Fisher, "Relational Model Approach to grid Information Services", grid Forum Informational Draft GWD-GIS-012, 2002

[7] B. Plale, et al, Key Concepts and Services of a grid Information Service, GWD-GIS-018, 2001

[8] M. Harchol-Balter, T. Leighton, and D. Lewin, "Resource discovery in distributed networks", ACM Symposium on Principles of Distributed Computing, May 1999, pp. 229-237.

[9] H. Casanova, M. Kim, J. S. Plank and J. Dongarra, "Adaptive scheduling for task farming with grid middleware", International Journal of Supercomputer Applications and High-Performance Computing, pp. 231-240, Volume 13, Number 3, Fall 1999.

[10] K. Czajkowski at al, "A Resource Management Architecture for Metacomputing Systems", In Proc of 4th Workshop on Job Scheduling strategies for Parallel Processing, pp 62-82, 1998

[11] L. A. Hall, "Approximation algorithms for scheduling", in Dorit S. Hochbaum (ed), "Approximation algorithms for NP-hard problems", PWS Publishing Company, 1997.

[12] A. Keller, A. Reinefeld, "Anatomy of a Resource Management System for HPC Clusters", Annual Review of Scalable Computing, vol 3, pp 1-31, 2001

[13] R. Raman, M. Livny, and M. Solomon, "Matchmaking: Distributed resource management for high throughput computing", 7th IEEE International Symposium on High Performance Distributed Computing, 1988, pp. 28-31.

[14] G. Shao, R. Wolski and F. Berman, "Performance effects of scheduling strategies for Master/Slave distributed applications", Technical Report TR-CS98-598, University of California, San Diego, September 1998.

[15] I. Foster and C. Kesselman. Globus: A Meatacomputing Infrastructure Toolkit. International Journal of Supercomputing applications. 11(2): 115-128, 1997.

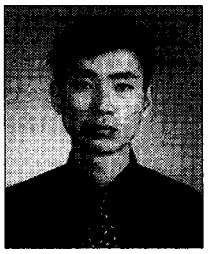
[16] The Globus Project homepage http://www.globus.org

[17] D. Comer,H. Du, "Dynamic Hashing Schemes", ACM Computing Surversy, Vol. 20, No. 2, pp. 85-113, 1988

[18] S. Lee, J. Song, at al, "A Database Index Structure for the Korean Electronic Dictionalry", Journal of Korea Information Science Society, Vol. 22, No. 1, pp. 3-12, Jan, 1995

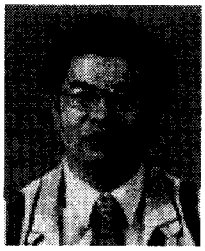
배 재 환 (Jae-hwan Bae)

1992: 경일대 산업공학과 (학사)
1997: 대구대학교 산업정보대학 정보통신학과 (석사)
2000~현재: 대구대학교 정보통신학과 박사과정
2000~현재: 탐라대 출판미디어학부 전임강사
관심분야: 3차원 게임엔진, 병렬 컴퓨터



권 성 호 (Cheng-Hao Quan)
1992: (중)연변대 전자공학(학사)
2001: 대구대학교 정보통신공학과 (석사)
2001~현재: 대구대 정보통신공학과 박사과정

관심분야: 컴퓨터 구조, 병렬처리, 지식관리 시스템



김 덕 수 (Deok-Soo Kim)

1992: 광주대 문헌정보학과 (학사)
1995: 대구대학교 산업정보대학 산업정보학 (석사)
2003: 대구대학교 정보공학 (박사)
2002~현재: 계명문화대학 문헌정보과 겸임교수
관심분야: 정보관리, 전자상거래, 이동통신

이 강 우 (Kang-Woo Lee)
1985: 연세대 전자공학 (학사)
1993: Univ. Southern California (Computer Eng. M.S.)
1997: Univ. Southern California (Computer Eng. Ph.D.)
1998~현재: 동국대 정보통신공학부 조교수
관심분야: 컴퓨터 구조, 병렬처리