# 그리드 환경에서 협업을 위한 워크플로우 프로세스 단위 기반의 애플리케이션 컨텐츠 배포 시스템

문석재* · 허혁* · 최영근*

## Application Contents Deploy System based on workflow process a unit based for collaboration in Grid Environment

Seok-Jae Moon* · Hyuk Heo* · Young-Keun Choi*


### 요 약

그리드 환경에서 워크플로우 프로세스 단위 기반의 애플리케이션 컨텐츠는 특정한 문제 해결을 위한 실질적인 작업이며, 그리드 자원에 분산 배포되고 서로 연관되어 실행되기 때문에 협업 환경을 구성하는 것은 매우 중요한 부분이다. 그리고 대부분의 협업은 워크플로우를 통하여 구체화되고, 그리드 환경에서 협업 처리를 위한 미들웨어로는 Globus toolkit이 대표적이다. 하지만 이 미들웨어는 그리드 환경 구축을 위한 기본 서비스들만을 제공하고, 협업을 할 수 있는 워크플로우 생성, 작업 스케줄링, 프로세스 단위 기반의 애플리케이션 관리 같은 부분은 적용되지 않는다. 또한 Globus Toolkit은 대형화된 그리드 커뮤니티 구성에는 적합하나, 소규모 대다수가 특정한 운영체제로 구성된 대형화된 그리드 커뮤니티 구성에는 적합하지 않고, 소규모 대다수가 특정한 운영체제로 구성된 그리드 커뮤니티 구성에는 부적합하다. 따라서 본 논문에서는 그리드 환경에서 소규모 협업 처리에 효율적인 워크플로우 프로세스 단위 기반의 애플리케이션 배포 시스템을 제안한다. 이 시스템은 애플리케이션 배포, 그리드 커뮤니티에 구성된 자원관리 등의 역할을 통해 효율적인 협업 환경을 지원한다. 또한 워크플로우 프로세스 단위 기반의 애플리케이션간 연관 관계 사전을 만들어 협업에 필요한 애플리케이션간의 정보 및 연관 관계를 표현하여 애플리케이션을 배포하는 기반 정보로도 활용한다.



### ABSTRACT

Application Contents of based on a workflow process unit in grid system is a practical work to solve the problem. To construct cooperation environment is very important because the contents are deployed and executed. And the most of cooperation is executed by workflow. Globus toolkit is the middleware for cooperation process in grid environment. However the middleware only provides basic services for constructing grid, not workflow that can provides cooperation, job scheduling, application contents management. Also Globus toolkit is suitable for huge gird community but small size pc based grid community. Therefore, this paper proposes the efficient application contents deployment system for small sizes cooperation. This system supports efficient cooperation environment with Application contents deployment and resource management provided in grid community. Also it presents information and association relation among application contents with association dictionary among application contents based on workflow process unit, and utilizes this as information for application contents deployment.



키워드

Grid, Workflow, Collaboration, Process


* 광운대학교 컴퓨터과학과

접수일자 : 2007. 8. 9

## Ⅰ. 서 론

그리드 컴퓨팅은 지리적으로 분산되어 있는 컴퓨터, 저장 장치 그리고 실험 장비 등의 자원을 광역 네트워크로 연결하여 상호 운영을 하거나, 특정한 문제 해결(Problem Solving Environment)을 위해서 물리적인 위치에 구애 받지 않고, 자원 교류와 협업 환경을 구성하는 것이다[1]. 그리드 환경에서 수행되는 작업은 대부분이 다수의 작업으로 구성된 복잡한 작업들로서 워크플로우를 이용하면 이러한 작업들을 효율적으로 처리할 수 있다. 최근에는 그리드 환경에서 워크플로우 기술을 적용하여 복잡한 작업들을 관리하기 위한 연구들이 활발하게 진행되고 있다. 대표적인 예로는 Pegasus[2], GridFlow[3] 등이 있다. GridFlow는 다양한 도메인을 통합하여 수시로 자원상태가 변동하는 동적인 그리드 환경에서 시간에 민감한 응용 프로그램들을 스케줄링 하는데 초점을 두고 있다. 그리고 효율적인 스케줄링을 위해 GridFlow는 퍼지 타이밍 기술과 응용 프로그램의 수행시간 예측기술을 사용한다. Pegasus 등의 워크플로우 시스템은 그리드 환경에서 효율적인 워크플로우 관리 기능을 제공하지만 서비스 개념을 제공하지 않는다. 따라서 워크플로우 만으로는 작업을 실행하고 결과물을 얻을 수 없다. 또한 사용자가 작성한 추상화된 워크플로우를 바탕으로 물리적으로 저장되어 있거나 사용자가 작성한 작업과 매핑하여 배포 할 수 있는 기반을 만들어야 한다. 이러한 작업은 프로세스 단위 기반의 애플리케이션을 사용자가 보다 효율적으로 검색하여 워크플로우를 작성하기 위해서는 애플리케이션 컨텐츠의 정보와 더불어 애플리케이션 컨텐츠 간의 연관성이 함께 명시되어 있어야 한다. 또한 애플리케이션을 실행하기 위해서는 자원들을 하나의 VO(Virtual Organization)로 구성 할 수 있는 미들웨어가 필요하다.

현재 그리드를 구성하기 위한 미들웨어로 미국의 아르곤 연구소를 주축으로 한 연구팀은 Globus Toolkit[4]를 제안하였다. Globus는 이 기종 자원의 단일한 접근 방법을 제공하고 필요한 자원의 검색을 위해 디렉터리 서비스를 제공한다. 이 밖에도 원격지 데이터 전송 방법, 사용자의 인증과 보안 방법들을 제공한다. Globus는 그리드 자원들에 대한 단일화된 접근 방법을 제공하지만 상위 계층의 그리드 애플리케이션이 요구하는 다양한 응용 서비스를 모두 충족시킬 수는 없다. 또한 Globus 미들웨어는 휘발성이 강한 PC를 기반으로 하는 그리드 환경을 구성하는 데에는 적합하지 않다. PC를 기반으로 하는 그리드 환경을 구성하기 위해서는 PC에 부담을 주어서는 안되며, 협업을 하기 위한 필요한 정보와 애플리케이션 컨텐츠를 실행 할 수 있는 메커니즘, 데이터를 전송할 수 있는 등 작업 처리를 하기 위하여 최대한 단순화해야 하며, 장애에 대하여 유연하게 대처 할 수 있는 방법이 필요하다. 현재의 PC 기반의 그리드 시스템으로 가장 대표적이라 할 수 있는 SETI@HOME[5], Korea@Home[6] 등의 프로젝트들이 있다. 이들은 대규모 작업을 처리하며, 동일 프로세스에 상이한 데이터들을 분산시켜 실행결과값을 취합한다. 하지만 상이한 다수의 소규모 분산 작업을 처리하기 위해서는 이러한 방식은 적합하지 않다.

본 논문에서는 PC기반의 그리드 환경을 구성하고 다수의 분산 작업을 배포하고 협업 처리 할 수 있는 애플리케이션 서비스 시스템을 제안한다. 애플리케이션 서비스 시스템은 워크플로우 프로세스 단위 기반의 애플리케이션 관리, 그리드 자원 관리, 자원이라 할 수 있는 노드 3가지로 구성 된다. 워크플로우 프로세스 단위 기반의 애플리케이션 관리 서비스는 사용자 또는 애플리케이션 등록자가 애플리케이션을 풀(pool)에 저장하고 이를 작업의 한 단위로 하여 배포하게 하며, 배포하기 위한 스케줄링은 노드로부터 얻은 정보에 의하여 스케줄링 하게 된다. 애플리케이션에 대한 정보, 연관 관계, 저장 위치의 정보 자원들을 상호 연관성에 따라 사전을 구성하고 이를 효율적으로 검색 관리 할 수 있도록 한다. 그리고 실질적인 작업을 실행하고 결과를 만들어내는 자원제공자인 노드는 애플리케이션 서버로부터 애플리케이션과 실행에 필요한 데이터 그리고 작업 흐름 명세를 받아 이를 분석하고 실행하여 결과를 만들어내기 때문에 노드에 대한 부담을 줄이고 장애에 대하여 신속한 대응을 할 수 있다. 본 논문의 구성은 다음과 같다. 2장에서는 관련연구를 설명한다. 3장에서는 워크플로우 프로세스 단위 기반의 애플리케이션 배포 시스템을 기술하고, 4장에서는 시스템 구현 및 성능 분석과 타 시스템과의 비교 분석에 대해 기술한다. 마지막으로 5장에서는 결론 및 향후 연구에 대해서 기술한다.

## Ⅱ. 관련 연구

### 2.1 그리드와 ACS(Application Contents Service)

그리드 애플리케이션이 복잡해질수록 그리드 시스템에 컨텐츠를 디플로이(deploy)하거나 애플리케이션 실행과 관리 작업에 대한 정보를 서술하는 것이 더 어려워 질 수 있다. 또한 애플리케이션이 복잡해지면 취급해야 할 파일들도 일반적으로 많아지게 된다. 그리고 만일 그 애플리케이션이 실행이 되어 끝날 때까지 일괄적으로 관리한다고 한다면 오류가 발생되기도 쉽다. ACS는 복잡한 애플리케이션 컨텐츠들을 일관되게 관리함으로써 디플로이(deploy)를 용이하게 하거나 애플리케이션 실행과 컨텐츠 관리에 대한 오류를 줄여주는 기능을 갖고 있다.

ACS는 현재 애플리케이션 컨텐츠 워킹 그룹에서 Fukui. et. al.를 중심으로 스펙이 연구되고 있다. ACS 시스템은 그리드 시스템에서 이용할 수 있는 하나의 서비스를 위한 시스템이며, 그리드 시스템의 다른 서비스 및 구성 요소들과 연동하며 동작한다. ACS 시스템은 간단히 애플리케이션 저장소라 할 수 있다. 애플리케이션 활동 관리자는 애플리케이션 컨텐츠들을 ACS에 저장하고, 그리드 사용자는 원하는 애플리케이션을 ACS에 요청하여 사용하게 된다. 작업이 제출되면 작업 관리자는 자원 배치 서비스에게 해당 작업을 전달하고, 자원 배치 서비스에서는 ACS로 부터 실행하고자 하는 애플리케이션 가지고 와서 자원에 할당하게 된다. ACS는 애플리케이션 아카이브(application archive)를 관리하기 위해 필요한 것이다. 애플리케이션 아카이브에 포함된 애플리케이션 컨텐츠를 해석하고 실행하는 것은 ACS의 임무가 아니다. ACS로 인한 장점은 이를 이용함으로써 얻을 수 있는 장점을 애플리케이션 관리 측면과 사용자 측면으로 나눌 수 있다. 애플리케이션 관리 측면에서 장점을 살펴보면, 첫째로 ACS는 그리드 활동 관리자와 애플리케이션 개발자의 업무를 분리해 준다. 기존에는 그리드 활용 관리자가 애플리케이션에 대한 상세 사항과 실행이 되기 위해 필요한 정보 등을 파악해서 처리해야 하는 부담이 있었다. 그러나 ACS가 있는 상황이라면 애플리케이션에 대해 더욱 잘 아는 애플리케이션 개발자가 애플리케이션 아카이브 디스크립터(application archive descriptor)에 위에서 언급한 내용을 기술하여 하나의 파일 형태로 그리드 활동 관리자에게 전달하고, 그리드 활동 관리자는 해당 파일을 ACS로 등록하게 된다. 이렇게 됨으로써 그리드 활동 관리자의 부담은 줄어들게 되고, 애플리케이션은 알맞은 조건의 그리드 자원에서 실행되게 된다. 두 번째로 애플리케이션 버전관리를 효율적으로 할 수 있게 된다. 애플리케이션 컨텐츠에는 애플리케이션 실행에 필요한 여러 파일이 포함된다고 언급하였다. 여기에 포함된 파일 중에는 새로운 버전이 나올 때, 변경된 파일도 있을 것이고 아닌 경우도 있을 것이다. 애플리케이션 개발자는 새로운 버전을 만들 때, 변경된 부분만을 그리드 활동 관리자에게 전달하여 등록하면, ACS는 이 부분만을 기존의 애플리케이션 컨텐츠에 추가하여 새로운 버전을 관리할 수 있게 된다. 이렇게 하면 애플리케이션 저장 공간을 줄일 수 있게 되고, 애플리케이션 버전 변경에 대한 기록을 함으로써 구 버전을 원하는 경우에는 대응 할 수 있게 된다. 세 번째로 관리 영역이 다른 그리드 시스템에서 각각 ACS를 생각해 보자. 그리드 사용자가 자신이 속한 그리드 시스템에 원하는 애플리케이션이 없을 경우, 직접 다른 그리드 시스템의 ACS에 등록된 애플리케이션 목록을 조회하는 방식을 통해서 이용할 수 있다. 또한 한쪽 그리드 시스템의 ACS를 새로 구축 또는 복구 등을 위해서도 다른 그리드 시스템에 있는 ACS와의 연동을 통해 애플리케이션 이용이 가능하게 된다.

### 2.2 작업 수행 모델

그리드 환경과 PC기반 그리드의 가장 큰 차이점은 자원의 가용성이라 할 수 있다. 대부분의 그리드 환경은 순수 연구 목적으로 하기 때문에 계산 작업 이외에 기타 작업에서는 자원 사용이 거의 없으나 PC기반 그리드는 기본적으로 사용자의 업무처리를 위해 자원이 사용되다가 자원의 유휴 상태에 따라 제공된다는 것이다. 또한 그리드 환경에서 그리드 애플리케이션이 표현되는 작업의 흐름도 차이가 있다. 병렬적 작업 수행의 경우 각 작업의 결과에 영향을 받지 않기 때문에 작업 완료시간은 문제가 되지 않는다. 그러나 순차적 작업 수행 모델이나 네트워크적 작업 수행 모델은 다른 작업의 결과를 바탕으로 실행 되어야 하기 때문에 이전 작업이 완료되지 않으면 작업을 진행 할 수 없다. 따라서 PC기반의 그리드의 경우 병렬적 작업 수행 모델 형태의 작업을 대상으로 한다.

## Ⅲ. 애플리케이션 컨텐츠 배포 시스템

### 3.1 시스템 개요

본 논문에서 제안하는 애플리케이션 서비스 시스템 모델은 (그림1)과 같다. 시스템 구성은 크게 3가지로 나누어 볼 수 있다. 작업을 선택하고, 선택된 작업의 순서를 정하며, 작업에 대한 결과를 전달 받을 사용자가 있다. 또한 워크플로우 프로세스 단위 기반의 애플리케이션 컨텐츠의 등록, 삭제, 배포, 스케줄링 및 작업 순서 생성을 할 수 있는 애플리케이션과 PC로 구성된 그리드 자원을 관리, 감독 그리고 정보 수집을 하는 그리드 자원관리 하는 애플리케이션 컨텐츠 서버 부분이 있다. 그리고 실질적인 워크플로우 프로세스 단위 기반의 애플리케이션 컨텐츠를 실행하는 그리드 자원 부분이 있다.

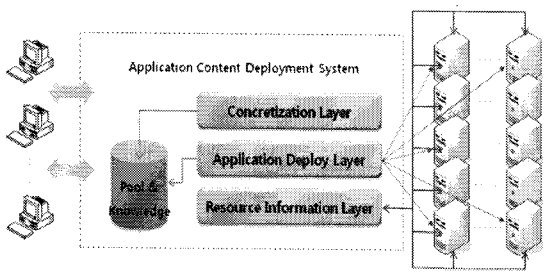


그림 1. 애플리케이션 배포를 위한 PC기반 그리드 구성도
Fig.1. PC based Grid for Application Deploymen

사용자는 작업을 검색하고 요청 할 수 있는 유저인터페이스에 따라 작업 수행 요청을 한다. 그러면 애플리케이션 컨텐츠 배포 서버의 구성 요소 중 자원정보 계층에서 그리드 커뮤니티가 구성된 노드들의 자원 정보와 상태 정보를 수집한다. 이때 작업 스케줄 서비스에서 수집된 자원정보를 토대로 작업 순서의 명세를 만들게 된다. 이렇게 만들어진 명세는 작업 배포 서비스에 의하여 해당하는 노드에 배포되게 된다. 이렇게 배포된 애플리케이션 컨텐츠는 배포된 노드에서 분석되어 실행된 후 스케줄에 따라 결과를 전달하게 된다.

### 3.2 애플리케이션 컨텐츠 배포 서버

본 논문에서 설계 및 구현하고자 하는 멀티 에이전트 시스템의 구조는 (그림2)와 같은 3단계의 계층적 구조를 가지고 있다. 제안한 시스템은 사용자가 처리하고자 하는 작업을 찾고 선택한 정보를 저장하고 구체화시키는 애플리케이션 컨텐츠 정보 계층(Application information Layer), 실질적인 작업이라 할 수 있는 애플리케이션 컨텐츠를 배포하기 위한 모든 기능을 가지고 있는 애플리케이션 배포 계층(Application Deploy Layer), 그리고 PC로 구성된 그리드 자원을 커뮤니티로 이루어진 그리드 자원의 정보를 관리하는 자원 정보 계층(Resource information Layer)으로 구성된다.

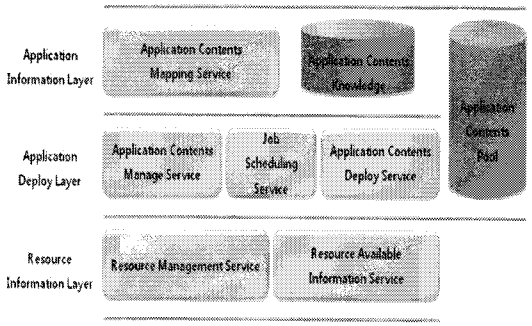


그림 2. 애플리케이션 컨텐츠 배포 시스템 구성도
Fig.2. Application Contents Deployment System

#### 3.2.1 애플리케이션 컨텐츠 정보 계층

애플리케이션 컨텐츠 정보 계층은 사용자가 특정한 문제 해결을 하기 위하여 작업을 검색하고 선택한 정보를 토대로 애플리케이션 컨텐츠의 연관 관계를 찾고 그 실체의 값과 매핑 시키는 역할을 한다. 또한 애플리케이션에 대한 정보를 저장 관리하는 역할을 한다. 사용자가 작업을 분산 작업을 수행하기 위하여 애플리케이션 컨텐츠 검색하여 서비스와 작업흐름 정보를 제공하면, 작업 순서를 만들기 위해서는 애플리케이션 컨텐츠 정보가 필요로 하게 된다. 이 정보는 애플리케이션 컨텐츠 검색 서비스가 애플리케이션 컨텐츠 지식베이스에 저장되어 있는 정보를 가져와 분석하여 보여주면 사용자는 이 정보를 토대로 추상적인 작업 순서 및 흐름을 만들게 된다. 이렇게 작성된 정보는 애플리케이션 컨텐츠 매핑 서비스를 통하여 실질적인 작업을 나열하여 협업 하기 위한 환경을 구성하기 위한 자료로 사용된다.

그림 3. 애플리케이션 컨텐츠 지식베이스
Fig.3. Application Contents Knowledge Base

(그림3)는 애플리케이션 컨텐츠의 지식베이스이다. 지식베이스는 XML문서의 집합이다. 하나의 애플리케이션 컨텐츠 지식베이스 문서는 협업에 필요한 각각의 구성요서의 필요한 모든 정보를 담고 있다. 애플리케이션 컨텐츠 정보, 컨텐츠 간의 연관관계 그리고 실제 컨텐츠가 저장되어 있는 물리적 위치까지 표현되어 있다. 애플리케이션 정보는 이름, 버전, 식별자, 크기, 입력 값, 출력 값이 표현된다. 이러한 정보를 토대로 워크플로우 명세표를 만든다. 작업 순서의 명세표는 아래 (표1)과 같은 항목으로 구성된다. 사용자가 선택한 정보는 (알고리즘 1)을 통하여 애플리케이션 풀에 저장된 실질적인 컨텐츠와 매핑 되어 선택 되어 진다.

표 1. 작업 흐름 명세 항목
Table 1 Job Sequence Specification

| | |
|---|---|
| Job ID | 처음 작업을 작업관리 서버에서 관리하여 배포 할 때 부여된 각각의 작업에 대하여 부여된 유일한 ID. |
| Job 주소 | 해당 작업이 실행되거나 또는 실행되고 있는 IP주소 |
| Job 결과전달주소 | 해당 작업 결과를 전달할 작업 IP주소, 즉 다음 처리해야 할 노드의 IP주소. |
| Job 출력 값 | 작업이 실행 되어서 나온 결과 값, 파일 형태로 만들어짐. |
| Job 입력 값 | 작업을 처리하기 위한 기본이 되는 자료. |
| Job 순서 | 작업을 처리할 순서. |

```
// 애플리케이션 컨텐츠를 배포하기 위한 컨텐츠 매핑
// 사용자로부터 입력 받은 내용을 얻어온다
contents_name = user_input(name);
contents_name = user_input(relation);

// 지식베이스에 맞는 인덱스 정보를 가져온다
Get_knowledgeBase_index();

// 인덱스 정보를 모두 검색한다
While EOF of knowledgeBase_index {

// 사용자가 선택한 정보와 맞는 정보를 찾는다
if (knowledgeBase_index(name) eqals name) {
    application_contents_info                    =
Get_application_contents(name, ID, location);
    deplay_information                           =
save_application_info(application_contents_info);

// 정보가 찾아지면 배포 서비스를 실행한다.
execute_Deploy_Service(deplay_information);
}
if (knowledgebase_index(name) not eqals name) {
    // 매핑되는 컨텐츠가 없을 때 요구자가 애플리케이션을 등록 또는 직접 배포
execute_Deploy_Service(user_request_deplay_information)
}
}
```

알고리즘 1. 어플리케이션 컨텐츠 배포 알고리즘
Algorithm.1. Application Contents Deployment Algorithm

3.2.2 애플리케이션 컨텐츠 배포 계층

애플리케이션 컨텐츠 배포 계층은 애플리케이션 컨텐츠의 등록, 삭제, 갱신의 컨텐츠 자체를 관리하는 애플리케이션 컨텐츠 관리 서비스 (Application Contents Manage Service)와 사용자에 의하여 선택된 정보를 기반으로 작업 처리 스케줄을 생성하는 작업 스케줄링 서비스(Job Scheduling Service) 그리고 실질적인 작업이라 할

수 있는 애플리케이션 컨텐츠 서비스를 스케줄링에 의하여 정해진 노드에 배포 할 수 있는 애플리케이션 컨텐츠 배포 서비스로 구성 되어 진다. (그림4)는 애플리케이션 컨텐츠 배포 시스템에 컨텐츠를 관리하기 위한 구성도이다.

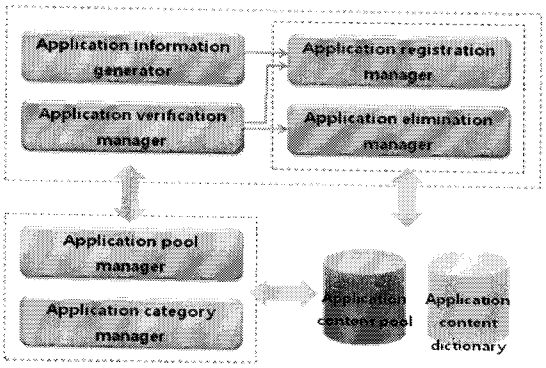


그림 4. 어플리케이션 컨텐츠 관리 구성도
Fig.4. Application Contents Management

애플리케이션 컨텐츠 관리 서비스는 크게 3부분으로 구분한다. 애플리케이션 컨텐츠를 등록, 갱신, 삭제를 하기 위해서는 각각의 애플리케이션에 대한 정보가 필요하다. 같은 역할을 하거나 동일한 컨텐츠가 있는지 검증 관리자를 통하여 확인 후 등록할 때는 애플리케이션 정보 생성기에 의하여 정보를 생성하고, 애플리케이션 등록 관리자에 의하여 등록한다. 이때 검증 관리자는 애플리케이션 풀 관리자와 카테고리 관리자를 통하여 정보를 얻는다. 이렇게 관리된 정보는 작업을 배포하기 위한 기본 정보가 되며, 이러한 정보를 조합하고 컨텐츠에 대한 많은 정보를 합하여 애플리케이션 지식 베이스를 생성하게 된다. 이렇게 등록된 정보들을 토대로 작업에 대한 배포에 근간이 되는 정보를 마련 하였다. 여기에 작업을 배포하기 위하여 필요한 것은 노드의 상태 정보이다. 노드의 상태 정보는 작업을 배포하여 실행 할 때 전체 수행 시간과 작업의 신뢰성을 결정 짓는 중요한 요소이다. 따라서 애플리케이션을 배포하기 위해서는 자원에 대한 가용 정보의 애플리케이션 컨텐츠 정보가 필요하고 이 정보를 토대로 작업을 배포하게 된다.

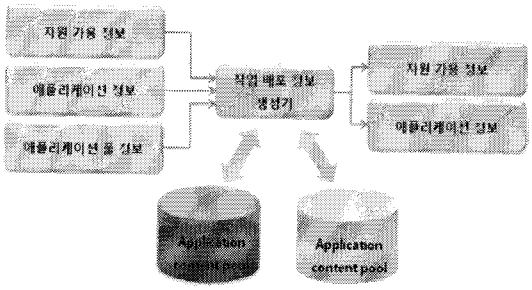


그림 5. 어플리케이션 컨텐츠 배포 구성도
Fig.5. Application Contents Deployment

(그림5)는 애플리케이션 컨텐츠의 배포 구성도이다. (알고리즘2)는 그리드 자원으로 구성 되어 있는 노드 성능의 척도인 유휴 정보에 따라서 배포 스케줄링을 위한 알고리즘이다. 유휴 정보는 표2의 가용정보 항목의 CPU 사용률과 메모리 정보를 바탕으로 정렬하여 사용한다.

```
// 어플리케이션컨텐츠를 배포하기 위한 스케줄링 설정
Scheduling for Application Contents Deploy
{  //현재 가용가능한 노드 정보를 요청
CurrentUSEnodeinfo = REQUEST ALLNodeinformation;

//배포할 작업의 수 만큼의 노드 수 요청
nodeCount = CurrentUSEnodeinfo REQUEST nodeCount;

// 필요한 노드에대한 정보를 만든다
Create NodeInfoListTable
LOOP LAST nodeCount;
// 현재 선택된 노드로부터 작업처리에 필요한 노드의 성능정보를 수집
nodeperformanceinfo = CurrentUSEnodeinfo REQUEST nodeinfo(Time)
NodeInfoListTable[Time] = info REQUEST NodePerformanceAvaiableResource;

Time = Time + 1;
LOOP END;

// 성능정보를 성능별 정렬
REQUEST Sort(NodeInfoListTable, nodeperformanceinfo)
ApplicationContents Job[]
```

```
// 각 노드에 작업을 할당
LOOP LAST nodeCount;
// 작업과 정렬된 순서 노트의 정보를 바탕으로 작업을 배포
REQUEST Jobprocess( Job[], Sort[], NodeInfoListTable);
LOOP END;
```

알고리즘 2. 노드의 유휴 성능을 고려한 배포 스케줄링 알고리즘
Algorithm.2. Deployment Scheduling Algorithm Considering in Terms of Node Availability

스케줄링 할 때 서비스 수행을 위해서 배포되는 애플리케이션 컨텐츠는 전송 순서, 전송 성능을 고려한 스케줄링으로 전체 수행 시간을 단축하는데 있어 매우 중요하게 고려되어야 할 사항이다. 분산 처리의 결과 시간은 배포된 작업이 가장 늦게 완료된 작업 시간이다. 사용자는 인터페이스를 통하여 작업을 선택한다. 그러면 등록할 때 애플리케이션 컨텐츠에 대한 정보 이력을 바탕으로 작업 순서와 작업 처리사양 조합 정보를 산출하여 스케줄을 하게 된다. 이렇게 사용자에 의해서 선택된 정보는 노드의 정보 분석과 작업 흐름을 만들어 줄 수 있는 작업 순서 생성기를 통하여 물리적으로 저장되어 있는 애플리케이션 컨텐츠와 매핑한다. 이렇게 매핑 된 정보를 가지고 스케줄링 알고리즘에 의해서 스케줄을 계획하고 작업을 배포하게 된다. 애플리케이션 작업 배포 서비스는 스케줄링에서 나온 정보로 해당 노드에게 애플리케이션을 전달하는 역할을 한다.

3.2.3 자원 정보 계층

자원 정보 계층은 그리드를 구성하고 있는 자원들에 대한 상태 및 정보를 관리하는 역할을 한다. 그리드를 관리하는데 가장 필요한 요소는 그리드를 이루고 있는 표2와 같은 성능에 대한 정보를 수집하고 관리하여 이러한 정보가 필요한 곳에 정보를 제공한다. 이러한 기능을 그리드 자원관리 서비스와 그리드 가용정보 서비스에서 나누어 처리한다. 애플리케이션 서비스 서버에서 요청하면 자원 제공자는 자산의 노드 정보를 수집하여 전달한다. 이런 자원정보를 추출할 때 가장 중요한 부분이 가용 수치이다. 이 수치는 각각의 자원 노드에서 계산되어서 배포 서버에게 전달한다. 이는 CPU와 메모리 사용률을 계산하여 사용한다.

표 2. 자원 정보 수집 요소
Table 2. Resource Information Elements

| 항목 | 설명 |
|---|---|
| CPU | 해당 노드의 CPU 정보를 가져온다.<br>ex) intel p4 1.8G, AMD Athlon 64 3500+ |
| Memory | 해당 노드의 물리적으로 설치된 메모리 용량을 가져온다.<br>ex) 256MB, 512MB, 1024MB |
| 디스크 용량 | 노드 미들웨어가 설치되어 애플리케이션이 실행되는 하드디스크의 용량을 가져온다. |
| 작업 ID | 현재 실행 중이거나 또는 Job 디렉터리에 있는 작업들의 ID 정보를 가져온다. |
| 가용 수치 | cpu 사용률에 3분간의 프로세스 평균값과 메모리 사용률을 더해 표현한다.<br>- 가용값 = CPU%+(TotalMemory - UseMemory)/TotalMemory)<br>ex) 27% + ((512-323)/512)*100 = 63.914 |
| 운영체제 | 해당 노드에서 사용중인 노드의 OS정보를 가져온다.<br>ex) Winodws XP Pro Service Pack 4, Linux Fedora core |

(표2)에 있는 가용수치 식과 같이 3분간의 CPU사용률의평균값과 남아있는 메모리의 양을 백분율로 계산하여 합산한 수치 값을 가지고 판단한다. 작업ID는 배포되어 현재 실행중인 애플리케이션 컨텐츠들의 ID이다. 이는 현재 노드가 어떠한 작업을 하고 있는지에 대한 명세를 알고자 할 때 사용된다. 디스크 용량은 배포할 때 애플리케이션 컨텐츠를 복제하여 전달하게 되는데 복제할 애플리케이션 컨텐츠와 데이터, 생성될 데이터가 저장될 수 있는지 여부를 판단할 때 사용된다. 그리고 OS정보는 특정 OS에서 실행되어야 하는 애플리케이션은 OS정보와 시스템 정보를 필요로 한다.

### 3.3 PC 기반의 그리드 자원

노드의 구조는 그림9와 같은 소프트웨어 구조를 가지고 있다. 애플리케이션 컨텐츠를 실행하고 결과를 정해진 곳으로 전달하며 자신의 상태 정보와 가용정보 등을

제공하는 것이 노드이다.

3.3.1 노드 구성 요소

노드가 하는 가장 큰 역할은 하나의 자원으로서 동작하는 것이며, 해당 애플리케이션 컨텐츠를 실행하는 것이다. 이런 실행 환경이 구성 되려면 애플리케이션 컨텐츠를 받는 부분과 결과를 보내는 부분이 필요하다. 또한 받은 애플리케이션에 대한 검증하는 부분도 필요하다. 이런 노드는 워크플로우 중에 하나의 구성요소로 동작하는 것이다. 또한 그리드 자원으로써 자기 자신의 가용성 정보, 성능 정보를 그리드 관리 서비스에게 전달해야 한다. 따라서 협업 지원을 하기 위한 노드는 (그림6)과 같은 소프트웨어 구조를 가지게 된다. 애플리케이션 컨텐츠의 무결성은 부여된 ID정보와 파일 크기로 검증한다. 자원 제공자는 애플리케이션 서비스 서버에서 작업 스케줄링을 할 수 있는 기본 자료를 제공해야 한다. 따라서 자원 제공자인 노드는 현재 자기 자신의 CPU, 메모리, 스토리지 등의 정적 자원 정보와 현재 작업 중인 프로세스, 태스크 등의 동적 자원 정보를 제공한다. 이러한 자원 정보를 제공함으로써 동적인 상황에 대해 자원 스케줄링과 작업 스케줄링을 할 수 있게 된다.

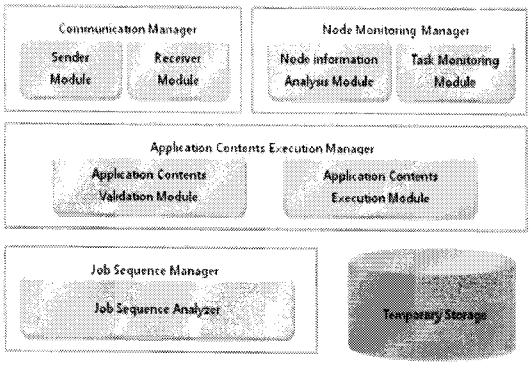


그림 6. 노드의 구성도
Fig.6 Node Architecture

작업 흐름 명세와 애플리케이션 컨텐츠는 애플리케이션 서비스 서버에서 할당 받고 그 이후 작업은 노드들 간의 협업에 의하여 이루어진다. 노드간에는 애플리케이션 컨텐츠와 작업 결과를 주고 받는다. 노드는 할당 받은 애플리케이션 컨텐츠 사용 후 바로 패기 하는 것이 아니라 추후 재사용을 하기 위하여 일정량을 보관하게 된다. 일정량을 초과 하게 되면 최초 할당 받은 순서대로 패기 한다. 최초 할당 받은 애플리케이션 컨텐츠 서비스 서버에서 할당 시키지만 한 번 이상 배포된 애플리케이션 컨텐츠는 다른 노드에서의 애플리케이션 컨텐츠를 복제 받는다. 애플리케이션 컨텐츠에 대한 정보는 서비스 서버의 지식베이스에서 정보를 제공받으며, 애플리케이션 컨텐츠의 식별은 ID정보로 한다. 협업이 시작되면 2개 이상의 애플리케이션 컨텐츠가 연관되어 실행되며, 워크플로우의 순서에 따라 실행된다. 이때 순차와 병렬적 흐름이 혼합된 작업 처리 중에 장애가 발생되면 전달하려는 노드의 애플리케이션 컨텐츠와 같은 노드를 서비스 서버로부터 검색하여 작업 흐름 명세서의 해당 부분을 수정하여 작업을 계속한다. 만약 그런 애플리케이션 컨텐츠를 가진 노드가 없다면 가용성이 가장 좋은 노드를 찾아 애플리케이션 컨텐츠를 할당하고 작업 순서 명세서를 수정 후 계속 진행 한다.

## Ⅳ. 시스템 구현 및 비교 분석

본 논문에서는 PC기반 그리드 환경에서 자동적인 애플리케이션 컨텐츠 배포 시스템을 제안 하였다.

### 4.1 시스템 구현

PC 기반 그리드 환경에서 애플리케이션 컨텐츠 배포 시스템을 구현하기 위한 환경은 Windows 2003 Server를 사용하며, 서버를 구성하기 위한 환경은 VS.NET 2005, .NET Framework 2.0을 사용하였으며, 노드의 실행 환경은 JDK 1.5.0.2를사용 하였다. 또한 보다 정확한 노드의 성능을 측정하기 위하여 JNI와 Windows의 System.DLL과 Visual C++ 6.0의 라이브러리를 사용하여 성능 측정을 데이터를 수집하였다.

(그림7)은 어플리케이션 컨텐츠의 사용자 화면으로 최초 시스템 구동 시 실행되는 화면이다. 이 프로그램을 실행하면 애플리케이션 배포 서버의 정보를 받아와 각각에 필요한 요소를 보여주게 된다. 해당하는 애플리케이션 컨텐츠를 선택하면 애플리케이션 컨텐츠에 대한 입력 값과 출력 값 물리적인 위치 등 애플리케이션 컨텐츠에 대한 상세한 정보를 볼 수 있다.

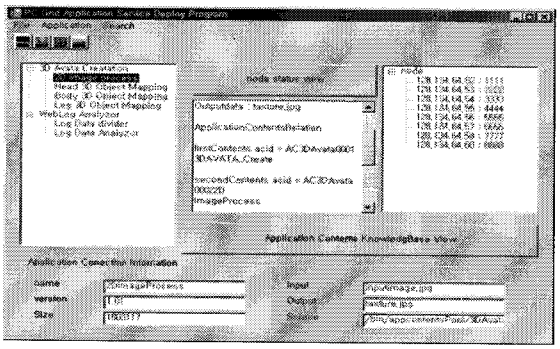

그림 7. 어플리케이션 컨텐츠 배포 시스템 화면
Fig.7. Screen Shot of Application Contents Deployment

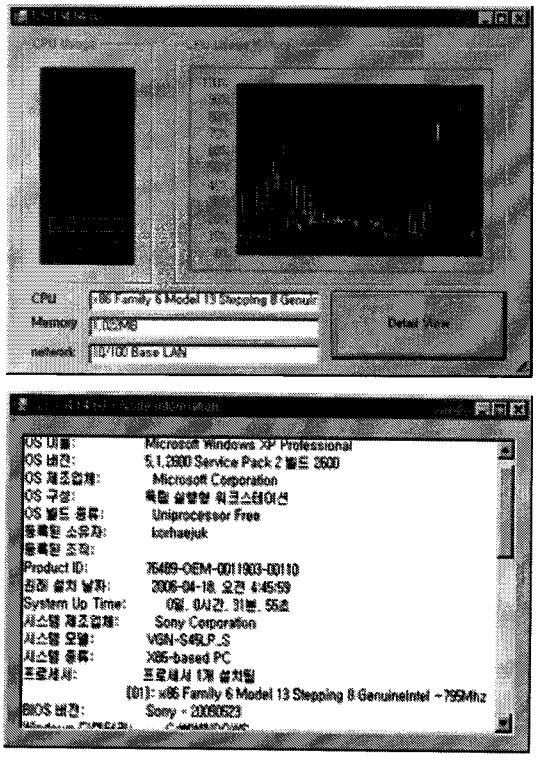


그림 8. 노드 자원 정보 모니터 화면 및 노드 자원 상세 정보
Fig.8. Screenshot of Node Resource Information and Information of Node Resource Specification

(그림8)은 그리드 자원의 개개의 요소를 모니터 하는 화면이다. (그림9)는 사용자가 선택한 애플리케이션 컨텐츠를 배포하는 화면이다. 사용자는 애플리케이션을 선택하고 선택이 완료되면 애플리케이션 컨텐츠를 배포 한다. 배포 할 때 필요한 정보를 하단의 다이얼로그에서 확인 할 수 있다.

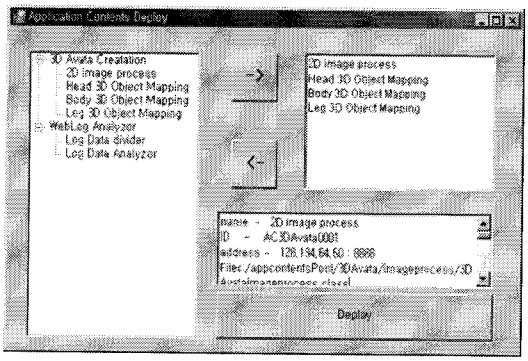


그림 9. 어플리케이션 컨텐츠 배포 화면
Fig.9. Screenshot of Application Contents Deployment

### 4.2 비교 분석

본 논문에서 제안한 시스템의 성능 측정을 위해 다음과 같은 실험을 수행 하였다.

표 3. 성능 측정을 위한 조건
Table 3. condition for performance measurement

| | |
|---|---|
| 실행환경 | - Pentium 4 제온 3.0 dual 1대 : 애플리케이션 컨텐츠 배포 서버<br>- Windows 2000/2003 : 10대<br>- Linux Fedora core : 2대 |
| 실행데이터 | - Matrix 프로세스와 Matrix 데이터를 분배<br>- Matrix 요소는 정수와 실수의 혼용임 |
| 측정을 위한 시스템 자원 제공자 규모 | - 단독수행, 2대, 4대, 8대, 12대 |

위 실험은 계산에 있어 시간적인 요소가 많이 소모되고 계산을 분할하여 처리하기 용이한 Matrix 곱셈 계산을 사용한다. 계산은 임의로 생성된 실수의 8192 by 8192의 행렬4096개의 계산을 실행 하였다. 각각의 실험은 그리드의 자원의 수가 단독수행에서 2대, 4대, 8대, 12대로 실험 하였다. 이 실험은 처리 시간 기준으로 측정하여 분석하였다. (그림10)은 성능 측정 결과이다. 자원노드를 임의로 구성된 자원은 총 18대중 12대만을 선별하여 사

용한다. 이 자원들을 가지고 성능을 고려하여 상위 성능부터 12대를 구성한 것과 무작위로 구성한 것과의 차이를 볼 때 자원의 성능 자체에 따른 성능이 자원 구성 규모가 작을 때는 큰 영향을 미치지만 자원을 제공하는 노드가 많아지면 자원 자체가 연산 하는 양이 적어지기 때문에 크게 최종 결과에 크게 미치지 않는 것으로 판단된다.

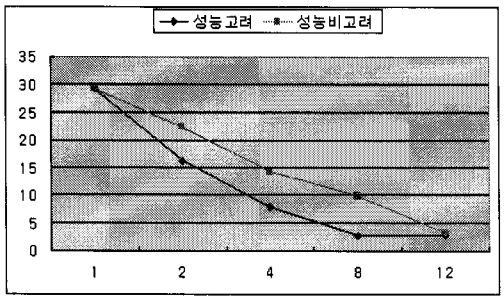


그림 10. 노드의 성능을 고려한 계산 수행 시간 비교
Fig.10. Execution Time of Resource Node Performance

추가적으로 본 논문에서 제안하는 분산 컴퓨팅 방법에 대한 다른 검증 방법으로 자원 가용수치에 따른 성능 분석을 하였다(그림11). 가용수치는 CPU가용률 100, 메모리 가용률 100을 기준으로 자원에서 수집되는 자원 정보를 가지고 수치가 만들어 진다.

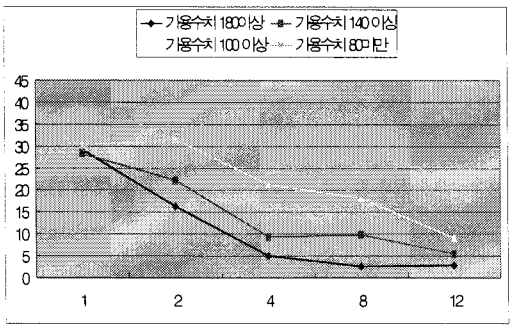


그림 11. 가용수치에 따른 전체수행 시간의 비교
Fig.11. Comparison of Total Execution Time

가용수치는 180, 140, 100, 80 4단계로 나누어 실험을 하였으며, 1가용수치가 180 이상인 경우는 자원 사용의 유휴 한 상태이고140이상은 아주 미비한 정도의 몇 개의 프로세스만이 구동되고 있는 상태이며, 100과 80은 자원이 매우 많은 작업을 수행하고 있는 상태를 말한다. 위 실험에서는 가용수치가 80으로 제한을 두고 시험을 하였다. 가용수치가 50미만인 상태에서 실험을 해보았으나 자원 자체의 문제로 인한 전체의 수행을 정상적으로 수행 할 수가 없었기 때문이다. 위 실험에서 보면 가용수치를 기반으로 한 분산 컴퓨팅을 하는 것이 효율적이라는 것을 볼 수 있다. 따라서 애플리케이션 컨텐츠를 배포할 시 자원의 상태의 가용수치를 기반으로 하여 배포하는 것이 분산처리의 효율성을 더 높일 수 있다. 다음은 기존의 연구와의 정성 분석을 통한 비교분석을 나타낸다.

표 4. 다른 PC기반의 분산컴퓨팅과의 비교분석
Table 4 Comparison of other Distributed Computing

| | SETI@ Home | Korea@ Home | 제안된 시스템 |
|---|---|---|---|
| 어플리케이션 컨텐츠 등록 삭제 | 지원하지 않음 | 부분적 지원 | 지원 |
| 다수 작업 처리 | 부분적 지원 | 부분적 지원 | 지원 |
| 노드 가용을 고려한 우선순위 스케줄링 | 지원하지 않음 | 지원하지 않음 | 지원 |
| 대단위 연산 처리 | 적합 | 적합 | 부적합 |
| 네트워크 규모 | 대규모 | 대규모 | 소규모 |

(표4)는 분산 컴퓨팅을 수행하고 있는 대표적인 @HOME 프로젝트와 비교한 것이다. PC를 기반으로 하는 분산 컴퓨팅 중 외계의 신호를 분석하는 SETI@Home과 우리나라에서 몇 가지 문제를 해결하기 위한 Korea@Home를 비교한 부분은 애플리케이션 컨텐츠 등록 및 배포하는 부분과 다수의 작업 처리 가능 여부, 노드의 가용을 고려한 우선 순위 작업 배포 스케줄링을 하는지에 대해 비교 분석하였다.

## Ⅷ. 결 론

본 논문에서는 PC 기반 그리드 환경에서 소규모 협업

처리를 위한 효율적인 어플리케이션 서비스 시스템을 제시하였다. 이러한 방법으로 인하여 사용자는 추상적인 간단한 방법으로 그리드 자원을 사용하여 처리하고자 하는 작업을 배포하고 수행 할 수 있다. 또한 본 시스템을 사용함으로써 어플리케이션에 대한 관리 능력이 향상 되었고 자동적으로 어플리케이션 컨텐츠가 배포 가능 하게 되었다. 또한 자원들의 가용정보를 토대로 애플리케이션 컨텐츠를 배포하여 결과를 얻기 까지 수행 시간의 감소를 가져왔다. 따라서 이 시스템을 기반으로 하여 PC를 기반으로 구성된 그리드 환경에서 소규모 협업 작업을 처리를 보다 편리하고 용이하게 할 수 있을 것으로 기대 된다.

향후 연구로는 지능적으로 어플리케이션 컨텐츠를 검색 할 수 있는 검색 시스템이 필요하며, 보안적인 측면에서 어플리케이션 컨텐츠의 검증부분이 강화 되어야 할 것 이다. 또한 그리드 자원을 사용하며 분산처리 할 때 작업의 흐름 즉 워크플로우의 효율적인 생성과 워크플로우의 분석에 대한 연구가 필요하다.

## 참고문헌


[1] I. Foster. C. Kesselman, S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations." International J. Supercomputer Application. 15(3). 2001.

[2] Y.Gill, C Kesselman, G.Mehta, S.Patil, M. Su, K. Cahi M. Livny, E. Deelman and J Blythe, "Pagasus Mapping Scientific Workflows onto the Grid." Across Grids Conference 2004.

[3] J. Cao, S. A Jarvis, S. Saini and G. R. Nadd,"GridFlow: WorkFlow Management for Grid Computing." 3rd International Symposium on Cluster Computing and the Grid,pp 12-15, 2003.

[4] Globus Project http://www.globus.org

[5] SETI@home. http://setiathome.ssl.berkeley.edu

[6] Korea@home http://www.koreaathome.org


## 저자소개


**문 석 재(Seok-Jae Moon)**

2002년 독학사 전자계산학 이학사
2004년 광운대학교 컴퓨터소프트웨어학과 공학석사
2004년~2005년 필컴정보시스템 주임연구원
2006년~ 현재 광운대학교 컴퓨터과학과 박사과정
※관심분야 : XMDR, 데이터 그리드, 상호운용성

**허 혁(Hyuk Heo)**

2004년 광운대학교 컴퓨터과학과 공학사
2007년 광운대학교 컴퓨터과학과 공학석사
2007년 ~ 현재 (주)아이센스 연구원
※관심분야 : 그리드 컴퓨팅, 분산객체컴퓨팅, 상호운용성

**최 영 근(Young-Keun Choi)**

1980년 서울대학교 수학교육과 이학사
1982년 서울대학교 계산통계학과 이학석사
1989년 서울대학교 계산통계학과 이학박사
1983년~ 현재 광운대학교 컴퓨터과학과 교수
1992년~2000년 광운대학교 전산정보원 원장
2002년~2005년 광운대학교 교무연구처장
※관심분야 : 객체지향설계, 분산시스템, 이동에이전트, 상호운용성