e-Science 환경을 위한 서비스 지향 구조의 그리드 미들웨어

숭실대학교 최재영 · 이상근

1. e-Science와 그리드 컴퓨팅

e-Science란 여러 기관에 있는 연구자들이 대규모 데이터의 처리를 요구하는 과학 연구를 효과적으로 수행하기 위해서 인터넷과 분산 컴퓨팅을 통해서 컴퓨팅 자원과 자료를 서로 공유하고 협력하면서 연구를 수행하는 것을 의미한다[1]. e-Science를 적용하면 실제 실험의 횟수를 줄일 수 있고, 많은 수의 컴퓨터를 활용하여 빠른 결과를 얻을 수 있으며 GUI를 적용한 편리한 연구 환경을 구축할 수 있다. 최근에는 생명 공학, 기상 예측, 항공 우주 공학, 물리학 등의 분야에서 그리드 컴퓨팅을 적용한 e-Science 환경을 구축하기 위한 노력이 국내외에서 활발히 진행되고 있다.

e-Science 환경 구축을 위한 분산 컴퓨팅 기술로 그리드 컴퓨팅이 널리 적용되고 있다. 그리드 컴퓨팅은 컴퓨팅 자원을 가상화하고 통합하여 서비스하는 고도화된 분산컴퓨팅 환경이다. 그리드 컴퓨팅은 기존의 분산 시스템과 달리 원격에 위치하며 각각 다른 방법으로 관리되는 다양한 기종의 시스템들도 통합할 수 있는 기반을 제공한다. 이러한 그리드 컴퓨팅의 등장으로 e-Science는 국가 간의 공동 연구와 컴퓨팅 자원의 공유가 가능한 시스템으로 발전하고 있다.

그리드 컴퓨팅을 적용한 e-Science의 사례들을 살펴보면 다음과 같다. 국내에서는 2002년부터 국가 그리드 과제를 통해서 그리드 기술이 개발되고 있으며, 2010년까지 선진국수준의 e-Science 환경을 구축하기 위한 목적으로 2005년부터 국가 e-Science 구축 사업이 진행되고 있다. 국외의 경우에도 미국의 PPDG[2], iVDGL[3], GriPhyN[4], EOL[5], OSG[6], ESG[7]와 유럽의 EGEE[8], myGrid[9], GridLab[10], LCG[11], 일본의 NAREGI[12] 등 그리드 컴퓨팅을 적용한 e-Science 구축이 전 세계적으로 활발히 진행되고 있다.

† 이 논문은 2005~2006년도 과학기술부의 국가 e-Science 프로젝트 지원으로 수행되었습니다.

그리드 컴퓨팅을 기반으로 e-Science 환경을 원활히 구축하기 위해서는 컴퓨팅 자원과 사용자의 작업을 관리하는 미들웨어의 구조가 효율적이어야 한다. 특히 생명공학 분야에서는 다음과 같은 특징으로 인해서 다른 분야에 비해 복잡한 미들웨어 구조가 사용되고 있다. 우선, MPI와 같은 전용 프로그램을 사용하는 분야와 달리 BLAST 등 UNIX 환경에서 기존에 사용하던 소프트웨어 패키지를 사용한다. 또 단일 소프트웨어의 실행으로 결과를 얻을 수 있는 것이 아니라 연구를 구성하는 각 단계별로 다양한 소프트웨어를 적용하여야 하므로 워크플로우 기술의 적용이 필요하다. 또 유전자 DB, 단백질 DB, 화합물 DB 등 여러 종류의 데이터를 사용하고 이러한 종류의 데이터를 관리할 수 있어야 한다. 따라서 복잡한 미들웨어들을 관리하기 위해서 계층적인 구조가 필요하며, 다양한 소프트웨어와 데이터를 효율적으로 사용하기 위해서 서비스 지향 구조의 적용이 필요하다.

이 논문에서는 국가 e-Science 구축 사업의 일환으로 진행되고 있는 HG2C(Human Genome to Chemicals) 프로젝트[13]의 경험을 바탕으로, 그리드 기반의 e-Science 환경을 구축하기 위한 미들웨어 구조에 대해 설명하고, HG2C를 위한 미들웨어인 MSF[14]와 MAGE[15]를 통합하여 이러한 미들웨어 구조를 효율적으로 구성하는 방법에 대해서도 살펴본다. MSF는 다양한 응용 프로그램을 지원하기 위한 서비스 지향구조를 제공하며 서비스들을 통합하여 워크플로우와 작업을 관리하는 기능을 제공한다. MAGE는 여러 개의 컴포넌트로 구성된 그리드 미들웨어의 효율적인 구성과 재배치를 위한 프레임워크를 제공한다. MSF와 MAGE를 적용하면 기존의 미들웨어 구조에 비해서 효율적으로 미들웨어를 배치하고 구성할 수 있다.

2. 관련연구

HG2C는 인간의 유전자 정보로부터 단백질 구조 모델을 생성하고 화합물이 결합될 수 있는 위치를 탐색

한 다음 여러 화합물을 결합시키는 과정을 거쳐서 신약후보 물질을 도출하는 연구 환경을 제공한다. 또한 인간 유전자 정보와 화합물의 연관 관계를 미리 데이터베이스로 축적하여 새로운 질병의 원인이 되는 유전자가 알려졌을 경우 바로 신약 후보 물질을 선택하여 실험을 할 수 있으므로 신약탐색 시간과 비용을 획기적으로 줄일 수 있다. HG2C와 유사한 국외의 연구내용으로는 미국 샌디애고 슈퍼컴퓨팅센터(SDSC)의 EOL (Encyclopedia of Life) 프로젝트가 있다. EOL 프로젝트는 모든 종의 모든 단백질 정보를 공공의 이익을 위해서 색인화 하는 것을 목표로 하며, 수집된 정보 백과사전처럼 접근할 수 있다. 특히 이러한 단백질 정보를 수집하기 위해서 iGAP(Integrative Genome Annotation Pipeline) 기법을 그리드 환경에서 수행한다. iGAP 포탈[16]의 계층 구조를 살펴보면 그림 1과 같다. 가장 하위에 그리드에서 작업을 수행하기 위한 미들웨어가 위치하고 그 위에 파라미터 처리를 수행하는 APST(AppLeS Parameter Sweep Template)가 위치한다. 최근에는 워크플로우 처리의 필요성 때문에 워크플로우 계층이 추가되었다. 워크플로우 시스템 상단에는 사용자가 접근할 수 있는 데이터들을 저장한 데이터베이스와 최상단의 사용자와 관리자를 위한 인터페이스가 존재한다.

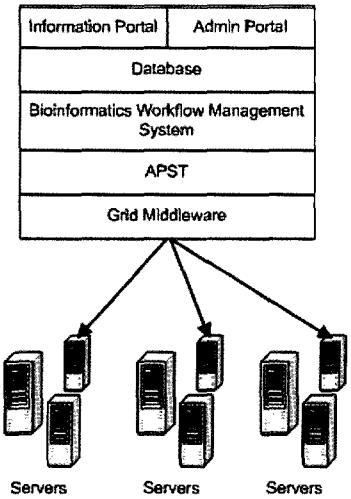

그림 1 iGAP 포탈 구조

e-Science 환경을 성공적으로 구축하기 위해서는 그리드 인프라를 구축하는데 사용되는 Globus Toolkit[17] 등의 저수준 미들웨어와 상위의 사용자 인터페이스를 통합해주는 미들웨어의 성능과 확장성이 중요하다. 이 때 통합을 위한 추상화 기능을 제공하는 것이 서비스 지향 구조(SOA: Service Oriented Architecture)이다. 서비스 지향구조는 서비스의 단위로 구성요소들의 접근 방법을 추상화하여서 기능과는 독립적으로 서비스를 호출하는 방법을 알면 서비스를 호출할 수 있고, 서비스간의 상호작용을 정의하여 서비스를 통합할 수 있는 프레임워크를 제공하는 환경이다. 서비스 지향 구조의 대표적인 예로는 웹서비스가 있다. e-Science 환경에 관한 연구 중에서 서비스 지향구조를 적용한 미들웨어 기술의 완성도가 높은 대표적인 연구로는 myGrid[9]와 GridLab[10]이 있다.

myGrid는 IT 연구자와 BT 연구자들의 협력을 통해서 응용 소프트웨어와 미들웨어 기능들을 전부 웹서비스로 구현하고, 이것들을 워크플로우로 통합하여 연구를 수행할 수 있는 환경을 제공한다. 또 연구에 관한 지식을 온톨로지로 통합하여 서비스를 분류하거나 워크플로우를 작성할 때 오류를 체크하는 등의 지능적인 서비스를 제공한다.

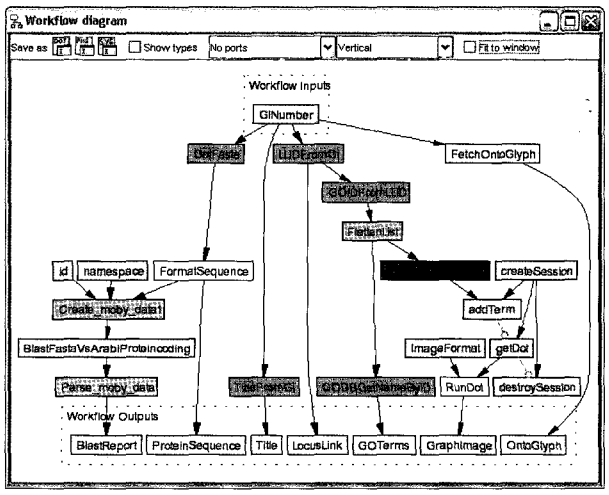

그림 2 myGrid의 Taverna 워크플로우 환경

GridLab은 GAT[18]라는 API를 바탕으로 미들웨어 프레임워크를 설계하고 GRMS[19], Mercury[20] 등의 미들웨어를 통합하여 계층화되고 구조화된 미들웨어 환경을 제공한다. 또 그리드 컴퓨팅이 e-Science에 적용되기 전부터 널리 쓰이던 Triana 워크플로우 시스템[21]을 그리드 환경에 통합하여 기존의 Triana 사용자들이 그리드 환경으로 손쉽게 이전할 수 있도록 하였다.

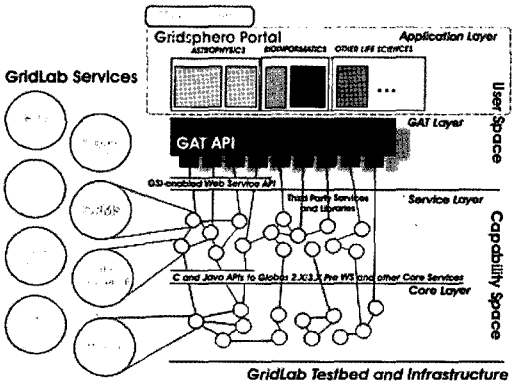

그림 3 GridLab 계층 구조

3. e-Science 환경을 위한 그리드 미들웨어의 구성 요소

3.1 미들웨어 구성 요소

그리드 컴퓨팅을 기반으로 e-Science 환경을 구축하기 위해서 필요한 구성 요소들을 살펴보면 다음과 같다. 우선 사용자가 자신의 과학 연구 과정을 구성하는 단위 실험들을 서비스로 표현할 수 있어야 한다. 그리고 서비스로 표현한 연구 과정들을 통합하고 데이터를 연결할 수 있는 워크플로우 기능이 필수적이다. 또한 사용자가 워크플로우로 기술한 실험 내용을 컴퓨팅 자원에 분배해 주는 작업 스케줄러 및 사용자의 작업이 어떤 자원에 분배되었는지를 알려주는 모니터링 기능을 제공하여야 한다. 뿐만 아니라 컴퓨팅 자원이 쉽게 그리드 환경에 추가되고 통합되어서 사용자가 보다 높은 성능이 필요할 때 성능을 확장할 수 있도록 자원을 통합 관리할 수 있어야 한다. 마지막으로 사용자의 연구 분야나 대상 실험의 특성에 따라 데이터 마이닝, 데이터 복제 관리, 데이터베이스 통합 등의 부가기능이 필요하다.

3.2 미들웨어 계층 구조

그리드 기반 e-Science 환경 구축을 위한 시스템 구조를 계층 별로 살펴보면 그림 4와 같다. 우선 그리드 컴퓨팅을 기반으로 컴퓨팅 자원들을 가상화해주는 자원 가상화 계층이 존재한다. 그 위의 미들웨어/인프라 관리 계층은 자원을 관리하고 작업을 스케줄링하며 시스템을 구성하는 역할을 한다. 마지막으로 서비스 지향 구조 계층은 사용자가 연구내용을 기술하고, 기술한 내용을 해석하여 미들웨어/인프라 계층에 작업의 실행을 요구한다.

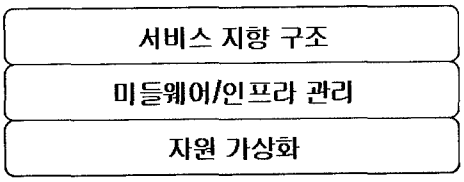

그림 4 그리드 기반 e-Science 환경의 계층 구조

자원 가상화 계층은 응용 소프트웨어, 데이터 등을 가상화하여 통합하고, 이것을 그리드 환경에서 접근할 수 있도록 인터페이스를 제공하는 계층이다. 이 계층에는 Globus Toolkit이 설치되어 사용자 인증, 작업 실행, 데이터 전송 등의 기본 기능을 제공한다. 또 LSF, PBS, Condor 등 클러스터 상의 배치 작업을 관리할 수 있는 스케줄러가 설치되어서 클러스터가 단일한 컴퓨팅 자원처럼 가상화되어 통합된다.

미들웨어 및 인프라 관리 계층은 자원 가상화 계층의 미들웨어들을 그리드 환경에 배치하고 설정 및 관리하여 그리드 환경을 구성하는 계층이다. 자원 가상화 계층에 설치된 미들웨어들의 정보를 수집하고, 이를 바탕으로 사용자의 작업과 데이터를 분배하는 기능을 수행한다. 또 새로운 자원이 추가되거나 특정 자원이 사용이 불가능하게 되는 등의 정보를 수집하여 시스템의 환경을 구성하고 유지 보수하는 역할을 제공하는 계층이다.

서비스 지향 구조 계층은 사용자가 서비스로 제공되는 소프트웨어와 데이터들을 통합하여 응용 연구를 설계할 수 있는 계층이다. 서비스 지향 구조 계층에 필요한 구성 요소들은 연구 내용을 작성하고 수정하여 재사용할 수 있는 워크플로우 편집기, 웹 포탈 등의 사용자 인터페이스와 사용자 인증 등 사용자의 작업을 미들웨어/인프라 계층에 전달하기 위해 필요한 기능들이 위치한다.

그림 4의 계층 구조와 그림 1의 iGAP 계층 구조를 비교해 보면 그리드 미들웨어 계층이 자원 가상화 계층에 해당하며, APST와 BWMS 등이 미들웨어/인프라 관리 계층에 해당한다. 나머지 부분은 사용자를 위한 서비스 지향 구조에 해당한다. 일반적으로 데이터베이스는 자원 가상화 계층에 해당되지만 iGAP에서는 데이터베이스를 사용자가 데이터에 접근하는 방법을 그리드 미들웨어에 대해 독립적으로 추상화하기 위한 목적으로 사용하므로 서비스 지향 구조에 속한다.

4. 현재 기술 수준 및 문제점

4.1 미들웨어의 설치 및 관리 기술

그리드 환경을 구축하기 위해서는 각각의 고유한 설정 방법과 관리 방법을 가지는 다양한 종류의 미들웨어들을 설치하여야 한다. 이러한 어려움을 해결하기 위한 방법으로 공통의 API를 기반으로 미들웨어 프레임워크를 설계하고 그 API를 이용해서 미들웨어를 구현하는 방법이 있다. 미들웨어 프레임워크를 적용한 연구의 대표적인 예로는 앞서 기술하였던 GridLab Project의 GAT를 들 수 있다. GAT는 그리드에서 사용될 수 있는 기능들을 파일 관리, 모니터링 및 이벤트 관리, 정보 교환, 자원 관리, 지원 유틸리티 서비스 등으로 구분하여 추상화한다. 그리고 GAT API를 이용해서 사용자 서비스를 구현하면 하위의 미들웨어를 다른 것으로 바꾸더라도 사용자 서비스의 코드를 새로 작성할 필요가 없다. 그러나 GAT API를 통해 새로운 미들웨어에 접근할 수 있는 모듈(GAT에서는 Provider 모듈)을 개발하여야 한다.

그리드 미들웨어를 설치하고 배포하는 다른 방법으로는 미들웨어를 웹 서비스 및 그리드 서비스로 구현하는 방법이 있다. 웹/그리드 서비스로 미들웨어를 구현하면 XML 형식의 WSDD(web-service deployment descriptor)를 이용해서 미들웨어를 컨테이너에 쉽게 설치할 수 있다. 또 설치된 미들웨어와 웹 서비스로 통신하므로 미들웨어에 접근하는 방법이 표준화되고, 미들웨어를 통합하기가 쉬워진다. 컨테이너에 웹 서비스와 그리드 서비스를 적용하는 경우를 비교해 보면, 웹 서비스로 구현할 경우에는 범용의 웹 서비스 컨테이너를 사용할 수 있는 장점이 있으며, 그리드 서비스로 구현할 경우에는 GSI 인증[22]을 사용할 수 있고, WSRF 23]에서 제공하는 인터페이스를 활용하여 미들웨어의 상태 정보를 수집하고 관리할 수 있는 이점이 있다.

웹/그리드 서비스를 이용하여 미들웨어를 구현하는 방법이 다양한 자원들로 구성된 그리드 환경에서 이상적인 미들웨어 설치/배포 모델을 제시하고 있다. 또 Globus Toolkit 역시 그리드 서비스 기반으로 발전되고 있다. 그러나 성능상의 문제, 명세의 발전 속도 등의 문제로 인해 현재 그리드 기반 e-Science 환경의 구축에 널리 사용되지 못하고 있다. 또 웹/그리드 서비스는 인터페이스를 표준화할 수는 있지만 프로그램 코드가 직접 전송되는 것이 아니기 때문에 그리드 환경에 추가되는 컴퓨팅 자원에 자동으로 배포하고 설치하는 기능을 제공할 수 없다.

4.2 워크플로우 및 자원 관리 미들웨어

Globus Toolkit의 작업 관리자인 GRAM에서 작업의 실행 방법을 표현하는 RSL은 단일 프로그램을 수행하기 위한 정보만을 담고 있다. 따라서 Globus Toolkit에 다수의 데이터를 대상으로 반복 작업을 수행할 때는 Nimrod[24] 등 데이터와 파라미터를 관리하는 미들웨어가 필요하고, 워크플로우 기능이 필요할 때는 워크플로우 엔진이 필요하다. 이러한 불편함을 해결하기 위하여 Java CoG Kit[25]에 Karajan[26]으로 명명된 워크플로우 기능이 추가되었다. Java CoG Kit은 Globus Toolkit에서 제공하는 기능들에 접근할 수 있는 Java API를 제공하는 개발자 라이브러리로 최근에는 Globus Toolkit에 포함된다.

Karajan을 사용하면 작업들 간의 선후 관계, 각각의 자원이 수행될 컴퓨팅 자원, 데이터 전송, 동시 수행 여부, 루프, 메시지 출력 등의 부가 기능, 라디오 버튼 등의 간단한 사용자 인터페이스 등을 표현할 수 있다. Karajan의 장점은 다양한 기본 기능을 제공하여 마치 UNIX 환경에서 쉘 스크립트를 작성하는 것처럼 범용적으로 적용할 수 있는 워크플로우를 제공하는데 있다. 그러나 워크플로우에 작업을 기술하거나 데이터 전송을 기술할 때, 컴퓨팅 자원(구체적으로는 호스트 이름)을 명시적으로 기술하여야 한다. 따라서 그리드 환경에 연결된 호스트의 목록과 각 호스트의 부하 정보에 따라 동적으로 작업을 분배하는 것이 어렵다.

5. HG2C를 위한 e-Science 미들웨어

국내 e-Science 환경의 특징을 살펴보면 해외의 연구들과 비교하여 미들웨어와 응용 소프트웨어 모두에서 웹서비스가 원활히 도입되지 못하고 있다. 그리고 그리드 미들웨어의 설치 및 관리가 어려워서 그리드 기반의 e-Science가 크게 확산되지 못하고 있는 실정이다. 본 장에서는 HG2C를 구축하면서 서비스 지향 구조를 바탕으로 숭실대학교에서 개발하고 있는 e-Science 환경을 위한 미들웨어에 대해 기술한다.

5.1 MAGE (Modular and Adaptive Grid Environment)

MAGE는 그리드 환경에서 모듈 기반의 동적 재구성이 가능한 어플리케이션 프레임워크를 제공하기 위한 목적으로 개발되었다. 서비스 지향 구조를 기반으로 재사용이 가능한 서비스 컴포넌트들을 조합하여 전체 시스템을 구성할 수 있게 해주며, 서비스 컴포넌트간의 의존성을 최소화하고 서비스들 간의 통신 투명성을 제공해 준다. 또 시스템을 재시작하지 않고 미들웨어를 동적으로 재구성할 수 있으며, 미들웨어를 구성하는 모듈 간의 기능적 의존성에 따라 필요한 모듈을 자동으로 설치할 수 있다.

그림 5는 MAGE의 시스템 구조를 보여준다. MAGE는 Request Broker, Service Manager, Component Manager의 3개로 구성된다. 먼저 Request Broker는 클라이언트의 서비스 요구를 받아들이는 통신 지점 역할과 한 개 이상의 통신 프로토콜을 동시에 수행할 수 있도록 관리자 역할을 수행한다. 즉, 서로 다른 통신 프로토콜을 사용해서 들어온 클라이언트의 요구 메시지를 해석해서 MAGE 내부에서 사용되는 형식으로 변환한 후에 적절한 Component Manager에게 전달하는 역할을 수행한다. Service Manager는 MAGE 서비스들을 위한 서비스 저장소로 Request Broker와 Component Manager 사이의 서비스 검색 및 저장을 위한 서비스 브로커 역할을 수행한다. Service Manager에 저장된 서비스에 대한 메타정보를 기반으로 서비스들 간의 의존성 정보를 제공한다. 마지막으로 Component Manager는 컴포넌트의 등록과 해제 및 활성화, 비활성화와 같은 생명주기 관리를 담당하며, 원격에서의 컴포넌트 제어 및 설치, 제거 기능을 제공

한다. Request Broker가 MAGE 모듈을 호출하면 적절한 서비스 컴포넌트를 수행시킨 후, 그 결과를 반환한다. 컴포넌트에 대한 메타정보는 XML 형식의 컴포넌트 디스크립터 파일을 이용해서 정의된다. 이 정보는 MAGE 시작과 동시에 적재되며 원격의 노드에 설치를 요청할 때도 같이 제공된다.

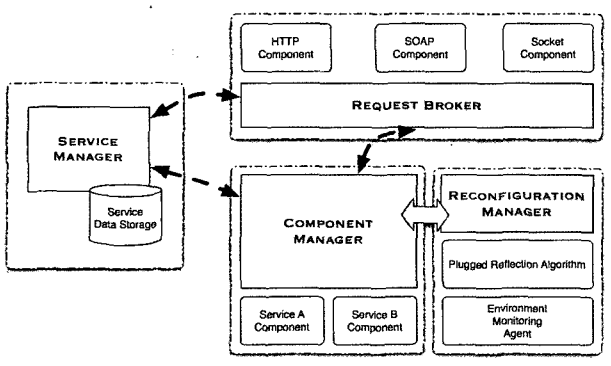

그림 5 MAGE 시스템 구조

MAGE 컴포넌트들은 컴포넌트 제어 연산과 서비스 연산들을 제공한다. 컴포넌트 제어 연산은 컴포넌트를 초기화하고 서비스 시작, 서비스 중지, 서비스 제거 등을 실행하며, 서비스 연산들은 컴포넌트의 속성을 설정하기 위한 연산과 메소드 호출을 위한 executeMethod 인터페이스를 제공한다. 사용자는 이러한 연산들을 호출하여 MAGE 컴포넌트를 설정하고 기능을 호출할 수 있다.

MAGE를 적용해서 미들웨어 인프라를 구성하였을 경우를 4장에서 기술한 GAT나 웹서비스와 비교하면 다음과 같은 장점이 있다.

- 웹서비스에 비해 가볍고 실제 모듈이 네트워크로 전송되므로, 미들웨어나 설정 도구들을 MAGE 모듈로 구현해서 배포가 가능하다.
- GAT가 미들웨어 의존성을 해결하기 위한 추상화 계층에 중점을 두는 것에 반해, 통신의 투명성과 모듈을 그리드 네트워크에 배치하고 설정할 수 있는 기능에 중점을 둔다.
- 통신의 투명성을 보장하므로 추후에 웹서비스/그리드 서비스로 이전이 필요할 때보다 쉽게 이전할 수 있다.

XML로 기술된 메타정보를 통해서 모듈을 동적으로 설정할 수 있다.

5.2 MSF (Meta Services Framework)

MSF는 그리드 환경에서 워크플로우 기능을 제공하고, 메타서비스를 이용해서 워크플로우를 서비스로 재사용이 가능하게 하는 미들웨어로 다음과 같은 특징을 가진다.

- 파라미터 분배 기능, 작업 스케줄링 기능, 워크플로우 엔진은 서로 다른 소프트웨어를 인터페이스 모듈로 통합하기보다는 단일한 미들웨어로 구현한다.
- 미들웨어를 그 기능에 따라 다수의 모듈로 분할하고, 동적으로 변하는 그리드 환경에 적합하게 재배치하고 설정이 가능하도록 한다.
- 워크플로우를 서비스, 플로우, 태스크의 계층 구조로 분할하여 재사용성을 높이고, 웹 서비스, 소프트웨어 스트림 등 다양한 형식의 태스크를 확장할 수 있도록 한다.
- e-Science 환경에서 워크플로우를 편리하게 작성하고, 다른 사용자와 공유할 수 있도록 한다.

MSF 시스템의 구조는 그림 6과 같다. MSF 시스템은 자바로 구현된 다섯 개의 에이전트로 구성된다. AM(Access Manager)은 그리드 환경에서 사용자를 인증하고 어떠한 기능을 사용할 수 있는지를 확인한다. OM(Ontology Manager)은 서비스, 플로우, 태스크 등을 XML 파일로 저장하고 관리한다. SM(Service Manager)은 사용자가 OM에 접속해서 서비스를 검색하고 호출하면 해당 서비스의 메타서비스 정보를 이용해서 워크플로우를 생성한다. RM(Resource Manager)은 생성된 워크플로우를 해석하여 태스크로 분할하고 미리 수집된 자원 정보를 이용해서 스케줄링한다. RM에서 스케줄링한 작업은 EM(Execution Manager)에 할당되어 실행된다.

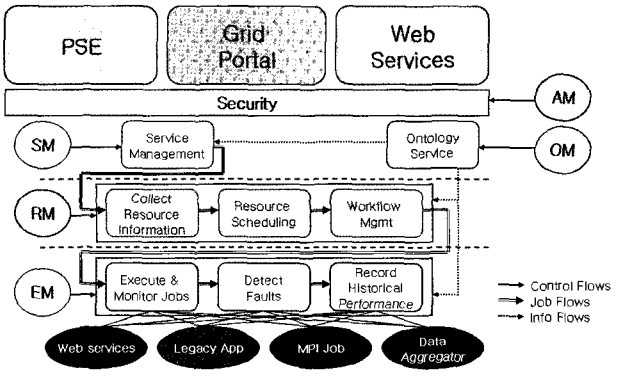

그림 6 MSF 시스템 구조

그림 6의 MSF 시스템의 구조를 그림 4의 계층 구조와 비교하면 EM은 웹서비스, 레거시 응용 프로그램 등의 작업을 가상화하고 실행하는 자원 가상화 계층, RM은 워크플로우의 분석과 스케줄링을 수행하는 미들웨어/인프라 관리 계층, 마지막으로 SM은 서비스 재사용과 공유를 위한 서비스 지향 구조 계층에 각각 대응된다.

그림 7은 MSF를 구성하는 에이전트들을 그리드 환경에 배치하는 예를 보여준다. 그림에서 VO는 그리드 자

원을 그룹으로 묶은 가상 조직(Virtual Organization)을 의미한다. AM, OM, SM은 두 개의 가상 조직에 소속된 그리드 시스템을 통합하기 위해 하나만 설치하였다. 만약 하나의 그리드 자원을 분할하여 여러 개의 포탈을 운영할 경우라면 각각의 에이전트가 둘 이상 설치될 수도 있다. 워크플로우를 스케줄링하는 역할을 하는 RM은 VO별로 설치하였으며 VO1에는 클러스터에서 사용 가능한 CPU들을 2개로 분할하여 관리하기 위해서 두 개의 EM을 사용하였다. 또 VO2에 설치된 EM은 VO1에 설치된 EM에도 작업을 제출할 수 있는 구조로 시스템을 구성하였다. MSF는 기능별로 모듈화가 되어 있어서 새로운 그리드 시스템이나 컴퓨팅 자원이 추가되거나 제거되는 경우에 손쉽게 전체 미들웨어를 재구성할 수 있다.

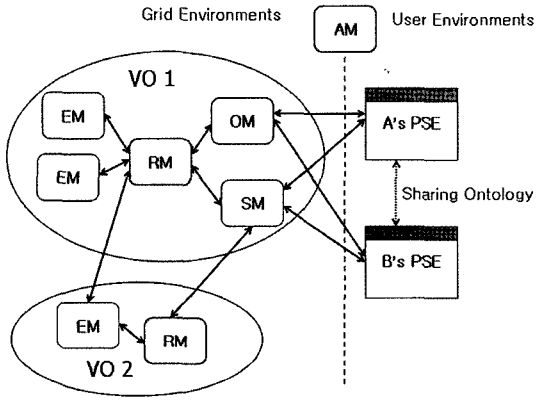

그림 7 MSF 시스템의 그리드 배치

5.3 MAGE를 이용한 MSF 구성 요소의 배치

그림 7과 같이 MSF 시스템을 그리드에 배치하려면, MSF 에이전트를 복사하고 호스트 이름과 포트 번호 등의 설정을 변경해야 한다. 또 태스크를 수행하는 EM이 적합한 RM에 연결되는 등 에이전트 간의 등록과 삭제가 적절히 수행되어야 한다. 이러한 배치 과정에 MAGE 시스템의 컴퍼넌트 분배와 설정 기능을 사용할 수 있으면 자동으로 에이전트를 복사하고 설정하는 것이 가능하다. 또 MSF를 구성하는 AM, OM, RM, EM, SM을 MAGE 컴퍼넌트로 구현하면 우선 해당 에이전트가 필요한 컴퓨터에 에이전트 프로그램을 전송하고, 호스트 이름과 포트 번호 등을 설정한 다음, 에이전트 간의 연결을 위해 각 에이전트에서 제공하는 연산을 자동적으로 수행하여 그리드 시스템을 효율적으로 구성할 수 있다.

6. 결론 및 향후 연구 계획

본 논문에서는 e-Science 환경을 구성하는 미들웨어의 특성과 계층 구조를 살펴보았고, 기존의 미들웨어 프레임워크의 대안으로 MSF와 MAGE를 이용한 미들웨어 구조에 대해서 설명하였다. **MSF는 e-Science 환경의 워크플로우를 자원 가상화, 미들웨어/인프라 관리, 서비스 지향 구조의 3단계로 분류하고, 전체 기능을 다섯 개의 구성요소로 분할한 구조를 가진다. 또 MSF의 구성요소들은 MAGE를 통해서 그리드 환경에 자동적으로 분산되어 설치되고 연결될 수 있다.**

향후에는 MSF를 위한 워크플로우 편집기, 포탈 인터페이스 지원, Globus Toolkit의 구성 요소들을 설정할 수 있는 MAGE 모듈들을 연구하여 보다 편리한 e-Science 구축 환경을 제공하고자 한다.

참고문헌

[1] Reading e-Science Centre, http://www.resc.rdg.ac.uk

[2] PPDG: Particle Physics Data Grid, http://www.ppdg.net

[3] iVDGL: International Virtual Data Grid Laboratory, http://www.ivdgl.org

[4] The GriPhyN Project, http://www.griphyn.org

[5] EOL: The Encyclopedia of Life, http://eol.sdsc.edu

[6] OSG: Open Science Grid, http://www.opensciencegrid.org

[7] ESG: Earth System Grid, http://www.earthsystemgrid.org

[8] EGEE: European Union Flag The Enabling Grids for E-sciencE, http://public.eu-egee.org

[9] R. Stevens, A. Robinson and C. Goble, "myGrid: personalized bioinformatics on the information grid," Bioinformatics, 19(1), pp.302-304, 2003.

[10] G. Allen et al., "Enabling applications on the Grid-a GridLab overview. Intl. Journal on High Performance Computing Applications," 17(4):449-466, 2003.

[11] LCG: LHC Computing Grid Project, http://lcg.web.cern.ch/LCG

[12] NAREGI Project, http://www.naregi.org/index_e.html

[13] HG2C Project, http://www.hg2c.org

[14] 이상근, 최재영, 황석찬, "그리드 환경에서 워크플로우의 서비스 매핑을 위한 메타 서비스", 정보처

리학회논문지A 제12-A권 제 4호, pp.289-296, 2005.
[15] S. Kwon, J. Choi, and J. Lee, "Protocol transparent application framework for Grid," Proceedings of HPC Asia '2005, IEEE Computer Society Press, Page 378-384, Nov. 30-Dec. 3, 2005.
[16] A. Shahab, D. Chuon, T. Suzumura, W. W. Li, R. W. Byrnes, K. Tanaka, L. Ang, S. Matsuoka, P. E. Bourne, M. A. Miller, P. W. Arzberger, "Grid Portal Interface for Interactive Use and Monitoring of High-Throughput Proteome Annotation," Lecture Notes In Computer Science, 3370:53-67.
[17] I. Foster, C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit," Intl J. Supercomputer Applications, 11(2):115-128, 1997.
[18] G. Allen et al., "The Grid Application Toolkit: Towards Generic and Easy Application Programming Interfaces for the Grid," Proceedings of the IEEE, Vol.93(3), pp.534-550, 2005.
[19] GRMS(Gridlab Resource Managemet System), http://gridlab.org/WorkPackages/wp-9/
[20] Mercury, http://www.gridlab.org/WorkPackages/ wp-11/
[21] Triana, http://www.triana.co.uk
[22] I. Foster, C. Kesselman, G. Tsudik, S. Tuecke, "A Security Architecture for Computational Grids," Proc. 5th ACM Conference on Computer and Communications Security Conference, pp. 83-92, 1998.
[23] K. Czajkowski, D. F. Ferguson, I. Foster, J. Frey, S. Graham, I. Sedukhin, D. Snelling, S. Tuecke, W. Vambenepe, "The WS-Resource Framework," March 5, 2004.
[24] D. Abramson, R. Sosic, J. Giddy, and B. Hall. "Nimrod: A tool for performing parameterised simulations using distributed workstations," In Proceedings of the 4th IEEE Symposium on High Performance Distributed Computing(HPDC). IEEE Computer Soceity Press, USA, 1995.
[25] G. von Laszewski, I. Foster, J. Gawor, P. Lane, "A Java Commodity Grid Toolkit," Concurrency: Practice and Experience, 13, 2001.
[26] Java CoG Kit Karajan Workflow Reference Manual, http://wiki.cogkit.org/index.php/Java_CoG_Kit_Karajan_Workflow_Reference_Manual

최 재 영

1984 서울대학교 제어계측공학과(학사)
1986 미국 남가주대학교 컴퓨터공학(석사)
1991 미국 코넬대학교 컴퓨터공학(박사)
1992~1994 미국 국립 오크리지연구소 연구원
1994~1995 미국 테네시 주립대학교 연구교수
2001~2002 미국 국립 수퍼컴퓨팅 응용센터(NCSA) 초빙연구원
1995~현재 숭실대학교 정보과학대학 컴퓨터학부 부교수
관심분야 : 고성능컴퓨팅(HPC), 병렬/분산 처리, 유비쿼터스컴퓨팅
E-mail : choi@ssu.ac.kr

이 상 근

2001 숭실대학교 컴퓨터학부(학사)
2003 숭실대학교 컴퓨터학과(석사)
2003~현재 숭실대학교 컴퓨터학과 박사과정
관심분야 : 그리드컴퓨팅, 고성능컴퓨팅 (HPC), 전자상거래
E-mail : sglee@ss.ssu.ac.kr