그리드 환경에서 워크플로우의 서비스 매핑을 위한 메타 서비스

이 상 근[†] · 최 재 영[††] · 황 석 찬[†††]

요 약

그리드 환경에서 수행되는 작업 중 많은 작업들은 연관된 작업들이 서로 결합된 워크플로우의 형태로 수행된다. 이러한 워크플로우를 서비스로 추상화한다면 사용자는 보다 손쉽게 워크플로우로 구성된 작업을 수행할 수 있다. 본 논문에서는 워크플로우를 서비스로 매핑하기 위한 메타 서비스를 정의하였다. 이 메타 서비스를 사용하면 워크플로우를 포탈 서비스, 그리드 서비스, 웹 서비스 등으로 쉽게 변환할 수 있다. 또한 워크플로우 사용자들 간에 워크플로우를 서비스의 형태로 공유하는 것도 역시 가능해진다. 마지막으로 과거의 성능 데이터 등의 서비스를 효율적으로 수행할 수 있는 정보들을 제공하여 QoS를 향상시킬 수 있다.

키워드 : 그리드, 그리드 포탈, 워크플로우, 웹 서비스, 메타 서비스

Meta Service: Mapping of a Service Request to a Workflow in Grid Environments

Sangkeon Lee[†] · Jaeyoung Choi[††] · Seogchan Hwang[†††]

ABSTRACT

Many jobs in Grid environments consist of several subtasks, and these subtasks can be represented by a workflow, which is executed effectively on a Grid. In this paper, we present Meta services which describe a mapping from a service request to a workflow in Grid environments. By using Meta services, a workflow in Grid environments could adapts various service concepts such as portal services, Grid services, and Web services. And the workflow can be shared and reused among workflow users. Furthermore, historical performance data can be included in Meta services, so effective scheduling of the workflow is also possible.

Key Words : Grid, Grid Portal, Workflow, Web Services, Meta Services

1. 서 론

그리드 컴퓨팅[1]은 다양한 자원들을 공유할 수 있는 분산 컴퓨팅 환경이다. 그리드 환경에서 공유하는 자원들은 종류가 다양하며, 자원의 지리적 위치 등 물리적 제약에 영향을 적게 받는다. 그리드 시스템을 이용하면 기존의 분산 시스템과 비교하여 더 큰 규모의 자원을 공유할 수 있으므로 사용자들에게 높은 성능을 제공할 수 있고, 그리드 환경에서 다양한 서비스를 제공받을 수 있으므로 보다 발전된 응용 소프트웨어의 개발이 가능하다.

미국 아르곤 국립 연구소, 시카고 대학 등에서 주도적으로 개발한 Globus Toolkit[2]은 그리드 시스템을 구축하기 위한 미들웨어로, API 및 클라이언트 도구들을 함께 제공한다. Globus Toolkit은 자원 할당, 원격 작업 제출 및 관리, 메타데이터 관리, 공개키 기반의 보안 등 그리드 환경 구축을 위한 기본적인 서비스들을 제공하지만, 워크플로우 관리나 자원 할당 등의 고급 기능은 제공하지 않는다.

그리드 환경에서 수행되는 작업은 대부분이 여러 개의 작업으로 구성된 복잡한 작업들이다. 워크플로우 기술을 이용하면 이러한 작업들을 효율적으로 통합할 수 있다. 최근에는 그리드 환경에서 워크플로우 기술을 적용하여 복잡한 작업들을 관리하기 위한 연구들이 활발하게 진행되고 있으며, 대표적인 연구로는 Pegasus[3], DAGMan[4], GridFlow[5] 등이 있다. 이에 대한 자세한 내용은 2장의 관련연구에서 다룬다.

MSF(Meta Scheduling Framework)[6]는 워크플로우를 GUI 기반의 편집기로 손쉽게 작성할 수 있으며, 완성된 워크플로우를 Globus Toolkit으로 구축된 그리드 시스템에서

※ 이 연구는 2005년 국가 e-Science 구축 사업의 지원으로 연구되었음.
† 준 회 원 : 숭실대학교 컴퓨터학과 박사과정
†† 종신회원 : 숭실대학교 정보과학대학 컴퓨터학부 부교수
††† 정 회 원 : 한국과학기술정보연구원 선임연구원
논문접수 : 2005년 1월 3일, 심사완료 : 2005년 6월 10일

효율적으로 실행시킬 수 있는 시스템이다. MSF에서는 워크플로우 기술을 위한 마크업 언어인 JCML(Job Control Markup Language)과 워크플로우 관리 시스템을 제공한다. 우리는 MSF 시스템을 이용하여 바이오 인포메틱스를 위한 그리드 포탈을 구축하였다. 포탈을 구축하여 운영해 본 우리의 경험을 비추어볼 때, 보다 쉽게 그리드 포탈을 구축하기 위해서는 다음과 같은 기능들이 추가적으로 요구되었다. 먼저 워크플로우를 손쉽게 호출하기 위해서는 사용하려는 워크플로우를 서비스로 추상화시킬 수 있어야 한다. 포탈 사용자 간에 워크플로우를 공유할 수 있도록 사용 권한을 제한할 수 있어야 하며, 마지막으로 그리드 자원을 보다 효과적으로 스케줄링할 수 있도록 과거의 성능 정보를 관리할 수 있어야 한다. 이러한 기능들은 MSF 시스템 이외의 워크플로우 관리 시스템에도 적용이 가능하므로, 메타 서비스의 개념은 범용성을 가진다.

본 논문에서는 사용자의 서비스 요구를 워크플로우로 매핑시키고 효율적으로 수행시킬 수 있는 메타 서비스에 대해 설명한다. 또 메타 서비스를 기술하기 위한 MSML(Meta Service Markup Language)과 메타 서비스를 MSF 시스템에 적용하여 확장한 시스템인 MSSF(Meta Service and Scheduling Framework) 시스템에 대해서도 설명한다. MSF 시스템에 메타 서비스를 적용한 결과 워크플로우 시스템을 크게 변경하지 않으면서도 서비스를 통한 추상화와 서비스 공유, 그리고 효율적인 자원 할당을 위한 시스템으로 확장할 수 있었다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 관련연구들을 살펴본다. 3장에서는 메타 서비스에 대해 설명하고, 4장에서는 MSML에 대해 설명한다. 5장에서는 메타 서비스와 MSML을 적용한 MSSF에 대해 설명한다. 마지막 장인 6장에서는 결론 및 향후과제에 대해 논의한다.

2. 관련 연구

그리드 자원에서 작업들을 스케줄링하고 작업들을 통합하여 워크플로우로 관리하기 위한 연구들을 살펴보면 다음과 같다. Condor[7]는 계산 집중적인 작업들을 위한 워크로드 관리와 DAGMan을 이용한 의존성 있는 자원들의 스케줄링 기능을 제공한다. Pegasus는 워크플로우 추상적인 응용 프로그램 의존적인 워크플로우들을 DAGMan에서 수행 가능한 워크플로우로 매핑하고 수행하기 위한 시스템이다. GridFlow는 에이전트 기반의 자원 관리 시스템과 지역 자원 스케줄링 시스템을 적용한 워크플로우 관리 시스템이다. 이 시스템은 다양한 도메인을 통합하여 수시로 자원의 상태가 변동하는 동적인 그리드 환경에서 시간에 민감한 그리드 응용 프로그램들을 스케줄링하는데 초점을 두고 있다. 효율적인 스케줄링을 위해 GridFlow는 퍼지 타이밍 기술과 응용 프로그램의 수행 시간 예측 기술[8]을 사용한다. 그러나 DAGMan이나 Pegasus 등의 워크플로우 시스템은 그리드 환경에서 효율적인 워크플로우 관리 기능을 제공하지만 서비스 개념을 제공하지 않는다.

그리드 환경을 위한 고수준 미들웨어 기능들을 서비스로 통합하여 제공하기 위한 연구들로는 MyGrid와 GridLab이 대표적이다. MyGrid[9]는 통합된 그리드 환경을 제공하기 위해 자원 발견, 워크플로우 실행[10], 분산 DB에서의 질의(query) 처리 등 온톨로지 기반의 서비스들을 제공한다. 이 시스템은 바이오 그리드 환경을 지원하기 위한 미들웨어 연구이다. MyGrid 시스템은 성능보다는 웹 서비스를 이용한 시스템의 통합에 중점을 두고 있다. GridLab 프로젝트[11]는 그리드 환경을 위한 미들웨어 서비스들과 응용 프로그램 도구들을 개발하고 있다. GridLab에서 제공하고 있는 서비스로는 자원 브로커링[12], 데이터 관리, 적응성 있는 서비스 등이 있다. GridLab의 미들웨어 서비스들은 GAT(GridLab Abstract Toolkit)[13]라는 단일한 API를 통해 접근할 수 있다. 그러나 myGrid 시스템과 GridLab 시스템의 경우에는 서비스 모델을 제공하지만 많은 시스템과 연관되어 있고, 워크플로우 모델을 범용적으로 기존의 워크플로우를 쉽게 통합시킬 수 없다.

마지막으로 과거의 성능 데이터가 효율적인 그리드 자원의 할당에 사용될 수 있음을 보여주는 연구로는 스미스의 연구[14]를 들 수 있다. 또한 Gridflow와 Pegasus 등의 많은 시스템에서도 과거의 성능 데이터를 이용하여 성능을 예측하고, 자원을 할당하고 있다. 그러나 기존의 연구에서는 포탈을 통한 서비스의 공유 기능이나 과거의 성능 정보를 서비스에 통합하기 위한 방법이 정의되지 않았다.

본 논문에서는 위에서 열거한 문제점들을 해결하기 위하여 다양한 워크플로우를 서비스로 통합할 수 있고, 사용자 간의 워크플로우 공유를 가능하게 하며 과거의 성능 정보를 통합할 수 있는 메타 서비스를 제안하였다.

3. 메타 서비스

3.1 메타 서비스의 필요성

메타 서비스는 아래의 세가지 필요성을 충족시켜야 한다.

첫째, 사용자의 서비스 요구를 실제 동작하는 워크플로우들로 매핑시킬 수 있는 명세가 필요하다. 일반적으로 워크플로우 안에는 많은 작업들과 데이터의 흐름이 복잡하게 연결되어 있다. 일단 워크플로우가 기술되고, 기술된 워크플로우가 그것을 수행하는 WfMS(Workflow Management System)에 저장되면, 사용자들은 이것들을 서비스의 형태로 사용하기를 원한다. 또 웹 서비스, 그리드 서비스, 포탈 서비스로 변환되어 수행되기를 원한다.

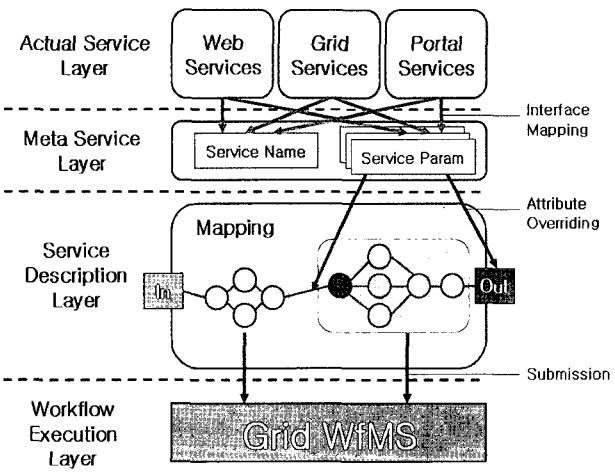
둘째, 사용자는 서비스의 접근 권한을 제어할 수 있어야 한다. 포탈은 포탈 사용자에게 공통적인 서비스를 제공하므로, 사용자가 작성한 서비스 중에서 공통적으로 유용하게 사용될 수 있는 서비스가 있다면, 그러한 서비스들을 포탈 서비스로 공유할 필요가 있다. 그러나 서비스가 특정 그룹에게만 제공되어야 하는 경우도 있으며, 권한이 없는 사용자에 보안이 필요한 경우도 있을 것이다. 이러한 경우에 메

타 서비스는 모든 사용자가 사용할 수 있는 서비스로 선언되거나, 특정한 사용자에게만 사용이 허락된 제한된 서비스로 모두 선언될 수 있어야 한다.

셋째, 사용자는 서비스가 실행될 자원이 할당되는 과정에서 최소한의 영향력을 행사할 수 있어야 한다. 평균적으로 사용자의 서비스를 빠른 속도로 수행할 수 있는 자원일지라도, 가끔씩 오류를 발생시키는 자원이 있다면, 사용자는 평균적인 수행 속도가 느리더라도 안정적인 자원을 선호할 수 있다. 또 특정 자원을 관리하는 기관이나 관리자의 신뢰도 때문에 특정 자원을 보다 신뢰할 수도 있으며, 특정 기관에서 운영하는 특정 자원을 불신하는 경우도 있을 수 있다. 기존의 서비스들은 이러한 사용자의 선호도를 반영할 수 없었다. 또한 그리드 환경에서 효율적으로 자원을 선택할 수 있도록 과거의 성능 정보를 기록하고 활용할 수 있어야 한다. 동일한 서비스 명세에서 생성된 서비스 인스턴스는 데이터나 파라미터의 값이 다르지만 데이터의 형식이 동일하고, 서비스를 구성하는 응용프로그램도 일반적으로 동일하므로, 동일한 자원에서 효율적으로 수행될 확률이 높다. 만약 과거에 효율적으로 서비스를 수행하였던 자원에 대한 성능 데이터가 수집되어 메타 서비스에 기록된다면, 스케줄러가 이러한 자원들을 우선적으로 할당하거나 높은 우선 순위를 적용하여 효과적인 작업 수행이 가능할 것이다.

3.2 메타 서비스의 구조

메타 서비스는 *{서비스 이름, 서비스 파라미터들, 서비스 명세}*의 집합으로 선언되며, 사용자의 서비스 요청과 해당 워크플로우의 매핑을 정의한다. 서비스가 이름과 파라미터들을 이용해 호출될 수 있기 때문에 워크플로우에 비해 보다 쉽게 호출될 수 있다. 그리고 메타 서비스도 서비스와 마찬가지로 이름과 파라미터로 호출되기 때문에 하나의 포트 타입과 하나의 오퍼레이션을 가지는 단순한 웹 서비스나 그리드 서비스의 형태로 손쉽게 변환될 수 있다. 나아가 CGI나 Java 서블릿 형태의 포탈 서비스로도 쉽게 변환될 수 있다.



(그림 1) 메타 서비스를 이용한 서비스 요청과 워크플로우 매핑

메타 서비스가 워크플로우로 매핑되는 과정을 살펴보면 (그림 1)과 같다. 서비스 이름은 서비스를 구분하는 구분자 역할을 하며, 서비스 파라미터는 서비스 내부에서 호출되는 워크플로우의 속성을 변경한다. 서비스 명세는 워크플로우들로 구성된 서비스 로직을 기술한다.

(그림 1)에서 보듯이 워크플로우와 서비스는 서비스 명세 계층에서 매핑된다. 서비스 명세에는 이 외에도 사용 권한 정보와 자원 정보를 포함한다. 메타 서비스의 서비스 명세에 포함된 내용들은 다음과 같다.

- 워크플로우 명세
  - 워크플로우 유닛의 선언
  - 워크플로우 유닛들로 구성된 서비스 로직
- 워크플로우의 공유를 위한 정보
  - 사용 권한
- 효율적인 스케줄링을 위한 자원 정보
  - 사용자가 명시한 자원 할당 제약 정보
  - 과거의 성능 정보

메타 서비스의 서비스 로직은 워크플로우를 최소 단위로 구성되는데, 이 최소 단위의 워크플로우를 워크플로우 유닛으로 정의한다. 워크플로우 유닛은 워크플로우가 포함하는 여러 작업들과 데이터 흐름 등을 객체의 속성으로 취급하고, 워크플로우 자체를 객체로 선언한다. 또 워크플로우 유닛은 하나의 메타 서비스에 여러 개 선언될 수 있다. 그리고 워크플로우 유닛들이 루프나 조건 내부에서 서로 연결되어 서비스 로직을 구성하며, 단순한 메타 서비스의 경우 하나의 워크플로우 유닛만을 호출할 수도 있다.

메타 서비스의 워크플로우 명세 부분에서는 사용자가 사용하기를 원하는 워크플로우 유닛을 여러 개 선언할 수 있다. 그리고 워크플로우 유닛들이 루프나 조건문 내부에서 서로 연결되어 수행될 수도 있으며, 극단적인 경우에는 메타 서비스는 단순히 하나의 워크플로우만을 호출할 수도 있다. 메타 서비스가 호출될 때 워크플로우의 속성이 서비스 파라미터와 사용자가 정의한 값에 따라 변경된다. 서비스가 수행되는 순간에는 워크플로우에 대해 변경된 속성 값을 가지는 새로운 워크플로우 인스턴스가 생성된다. 워크플로우가 서비스 호출에 의해 생성되기 때문에 사용자는 손쉽게 워크플로우를 생성할 수 있다. 또한 조건과 루프는 작업과 데이터 흐름에 비해 자주 변경되므로 이를 워크플로우 유닛에서 분리하여 워크플로우 유닛의 재사용성을 높일 수 있다.

메타 서비스를 이용하면 사용자들이 공통적으로 사용하는 서비스들을 포탈 서비스로 공유할 수 있다. 허락된 사용자는 해당 메타 서비스 파일을 읽고 실행할 수 있으며, 그 사용자의 저장소에 인스턴스가 생성되어 저장된다. 사용자가 서비스를 수정하기를 원한다면 서비스의 복사본을 이용해 서비스 내용을 수정할 수 있다.

자원 할당 제약 정보 부분에는 그리드 환경에서 작업의 효율적인 수행을 위한 자원 정보가 기록된다. 이 부분에 사

용자가 작업을 수행할 때 선호하는 자원이나 싫어하는 자원 정보를 등록할 수 있다. 사용자가 동일한 메타 서비스를 여러 번 수행하였을 경우, 사용자는 특정한 자원에서 서비스가 효율적으로 수행된다거나 혹은 다른 특정 자원에서 수행 도중에 오류가 발생되는 상황을 발견할 수도 있다. 과거의 성능 정보 부분에는 MER(most effective resources) 목록과 LER(less effective resources) 목록의 형식으로 과거의 성능 데이터가 기록된다. 사용자는 워크플로우 엔진에 의해서 기록되는 MER과 LER 목록의 수를 조정할 수 있다.

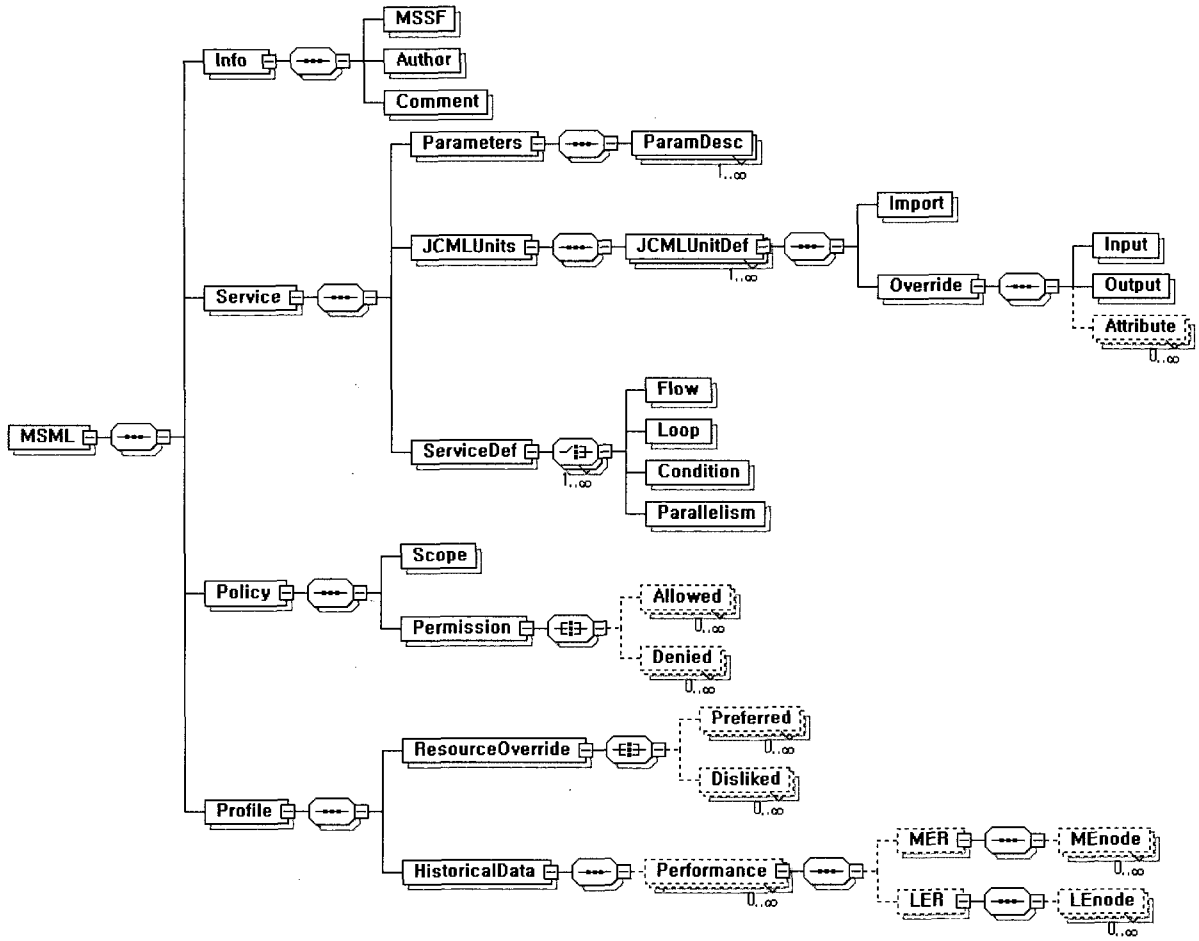
4. Meta Service Markup Language

MSML(Meta Service Markup Language)는 메타 서비스를 기술하기 위한 XML 형식의 마크업 언어이다. MSML의 목적은 MSF에서 이용되는 워크플로우를 메타 서비스로 통합하는 것이다. 메타 서비스를 기술하는 부분은 information, service description, policy, profile의 네 부분으로 구성된다. MSML의 XML 스키마는 (그림 2)와 같다.

우선 Info 부분은 MSSF 시스템 버전, MSML의 저자 등 MSML에 관한 정보를 기술한다. 서비스 부분에서는 메타 서비스를 정의하는데, 우선 서비스의 파라미터를 선언한 다음 서비스를 구성할 워크플로우를 워크플로우 유닛으로 정의한 다음, 마지막으로 서비스 내용을 기술한다.

서비스 정의 부분은 단순히 워크플로우 유닛을 호출하는 것에서부터 여러 워크플로우 유닛이 루프, 조건, 반복, 병렬 수행 등으로 복잡하게 연결된 것까지 다양한 수준의 서비스 내용을 포함하고 있다. 반복 회수나 조건 등은 작업들의 연결 상태 등의 워크플로우 구조보다는 빈번하게 변경될 수 있으므로, 이를 기본 단위에서 제외하는 것이 설계를 단순하게 하고 재사용성을 높일 수 있다. 따라서 워크플로우 언어에서는 조건, 반복 등을 포함하지 않고 데이터와 작업들의 흐름과 연결만을 기술하고 메타 서비스에서 서비스로 통합하면서 조건 및 반복 등을 수행한다.

정책 부분에서 Scope 부분에서는 서비스가 재사용될 수 있는 범위를 선언한다. Scope 부분에는 메타 서비스에서 몇몇 워크플로우 유닛이 다른 서비스로 재사용이 가능하도록 선언될 수도 있고, 서비스 전체를 재사용하도록 선언될 수 있다. Permission 부분에서는 다른 사용자가 서비스를 사용할 수 있는 권한을 선언한다. 특정 사용자나 그룹에게 읽기, 수정, 재사용 등이 허가될 수 있는지를 선언한다. Permission 부분이 비어있으면 기본 값으로 작성자만 접근하여 읽기, 수정, 재사용이 가능하다.



(그림 2) MSML 스키마

```
- <MSML>
  - <Info>
      <MSSF version="1.0" />
      <Author name="S.K.Lee" system="test1.ssu.ac.kr" date="2004.11.5" />
      <Comment>This service simply calls a JCML workflow named docking.jcml with some overrided value</Comment>
    </Info>
  - <Service>
    - <Parameters>
        <ParamDesc direction="in" name="protein" type="string" note="PDBQS file" />
        <ParamDesc direction="in" name="ligand" type="string" note="MOL2 file" />
        <ParamDesc direction="out" name="result" type="string" note="Autodock log file" />
      </Parameters>
    - <JCMLUnits>
      - <JCMLUnitDef name="docking">
          <Import type="jcml" location="/var/repository/portal/seventy9/jcml" name="docking.jcml" />
        - <Override>
            <Input taskid="a" dataid="aia" target="name" override="$ligand" />
            <Input taskid="c" dataid="cib" target="name" override="$protein" />
            <Output taskid="f" dataid="foa" target="name" override="$result" />
          </Override>
        </JCMLUnitDef>
      </JCMLUnits>
    </Service>
  - <Policy>
      <Scope />
      <Permission read="true" type="user">griduser1</Permission>
    </Policy>
  - <Profile>
    - <ResourceOverride>
        <Preferred type="PBSCluster" node="test1.ssu.ac.kr" />
      </ResourceOverride>
    - <HistoricalData recordPerf="true" MERsize="3" MFRsize="0">
      - <Performance>
        - <MER>
            <Enode time="153s">test2.ssu.ac.kr</Enode>
            <Enode time="245s">test3.ssu.ac.kr</Enode>
            <Enode time="288s">test3.ssu.ac.kr</Enode>
          </MER>
        </Performance>
      </HistoricalData>
    </Profile>
</MSML>
```

(그림 3) MSML 파일의 예

마지막 프로파일의 ResourceOverride 부분에서는 사용자가 특정 자원에 대한 선호도를 기술할 수 있다. 사용자가 서비스를 여러 번 수행하였을 경우, 사용자의 작업을 빠른 시간에 정확히 수행하는 자원과 잦은 오류를 발생시킨 자원을 구별할 수 있고, 이를 자원 할당에 반영시키기 위해 이 정보를 스케줄러가 사용할 수 있게 한다. MSML에 과거의 성능 데이터를 모두 기록하는 것은 비효율적이므로, 가장 성능이 높았던 자원들과 가장 성능이 낮았던 자원들을 사용자가 각각 정의한 만큼 WfMS에서 업데이트할 수 있도록 하였다.

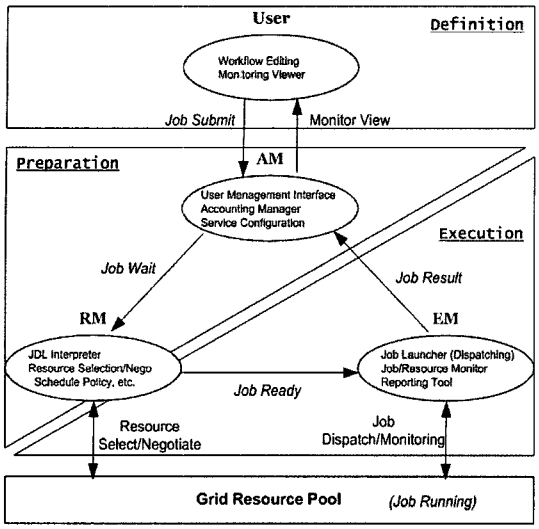
(그림 3)은 MSML 문서의 간략한 예를 보여준다. 이 MSML은 docking.jcml이라는 MSF 워크플로우 파일을 세 개의 인자를 가지는 WorkflowUnit으로 정의한다. 현재 MSML에서는 MSF 시스템의 워크플로우만을 지원한다. 그리고 서비스 정의부분에서 메타 서비스의 인자인 ligand, protein, result의 값으로 호출한다. 결과적으로 작업 a의 첫 번째 입력 데이터, 작업 c의 두 번째 입력 데이터, 작업 f의 첫 번째 출력 데이터를 각각 메타 서비스의 파라미터인 ligand, protein, result의 값으로 변경한다. Permission 부분에서는 워크플로우 작성자 이외에 사용자 id가 "griduser1"인 사용자가 해당 워크플로우에 대해 읽기 권한이 있다는 것을 명시하고 있다. Profile 부분에서는 PBS 클러스터 형식의 자원인 "test1.ssu.ac.kr"이 선호하는 자원으로 등록되었으며, 작업이 수행될 때 부하가 크지 않다면 우선적으로 할당된다. MER에는 3개의 자원이 등록되어 있는데, 이 자원들은 스케줄링되면서 우선 순위에 가중치를 가지게 된다.

5. 메타 서비스를 적용한 MSF 시스템의 확장

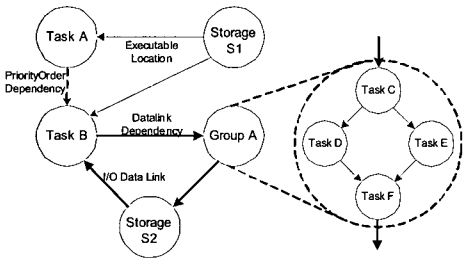
5.1 MSF 시스템의 개요 및 특징

3.1에서 설명한 메타 서비스의 요구사항은 MSF 시스템뿐만 아니라 다른 워크플로우 시스템에도 유용하게 적용될 수 있다. 그러나 본 연구에서는 MSF 시스템의 워크플로우를 서비스로 확장하는 것을 목적으로 한다. 그러므로 설계된 메타 서비스는 MSF의 워크플로우 명세와 워크플로우 시스템의 변경을 최소화하면서도 효과적으로 앞서 설명한 세가지 사항을 반영할 수 있어야 한다.

MSF 시스템은 AM(Access Manager), RM(Resource Manager), EM(Execution Manager), IR(Information Repository)의 구성 요소로 구성되며 그 구조는 (그림 4)와 같다.

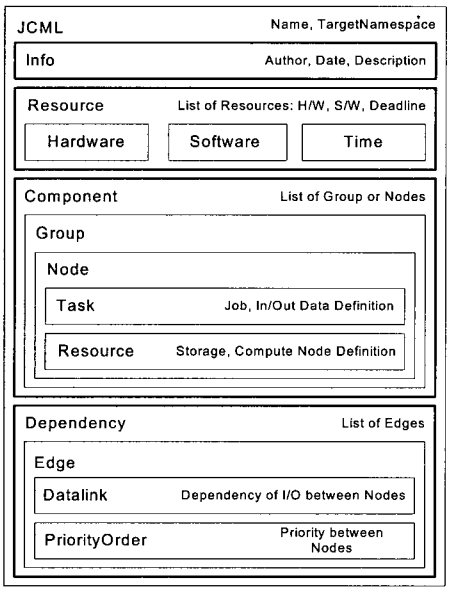


(그림 4) MSF 시스템의 구조



(그림 5) MSF 시스템의 워크플로우 모델

AM은 사용자 인증과 환경 설정, 시스템 접근 등의 서비스를 제공한다. RM은 자원을 발견하고 자원을 할당하는 역할을 한다. EM은 작업을 그리드 환경에 제출하고, 상태를 모니터링하며 결과를 확인한다. IR은 AM, RM, EM에서 사용되는 메타 데이터 들을 통합적으로 관리하는 역할을 한다.



(그림 6) MSF 워크플로우 명세의 형식

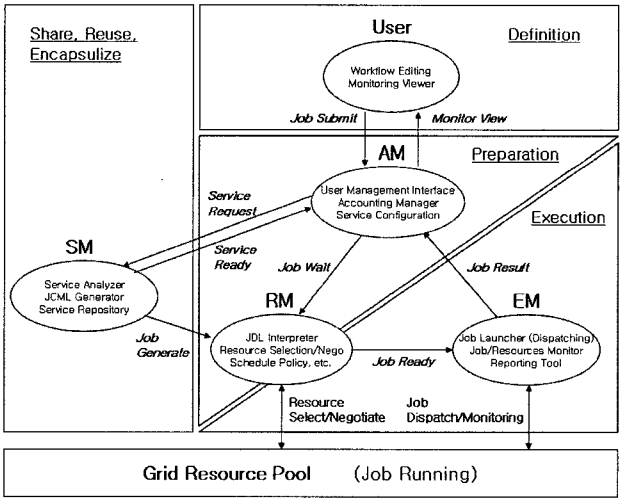
MSF 시스템의 워크플로우 모델은 (그림 5)와 같이 작업과 작업 그룹, 데이터 흐름으로 구성된다. 이 워크플로우 모델은 XML 형식의 워크플로우 명세 파일로 저장되는데, 그 형식은 (그림 6)과 같다.

워크플로우의 작업들은 실행 파일, 경로 등의 속성을 가지는 객체로 취급될 수 있으며, 데이터의 흐름 역시 마찬가지로 객체로 취급될 수 있다. 또 XML 문서는 JAXB(Java Architecture for XML Binding)[15] 기술 등을 통해 일반적으로 객체로 쉽게 매핑이 가능하다. 따라서 메타 서비스는 MSF의 워크플로우가 아닌 다른 워크플로우일지라도 XML로 변환할 수 있다면 적용이 가능하다.

5.2 MSSF 시스템

MSSF 시스템은 메타 서비스를 적용하여 MSF 시스템을 확장한 시스템이다. 메타 서비스를 통합하기 위해, 서비스를 관리하는 SM(Service Manager)이 추가되었다. SM의 구조는 (그림 7)과 같다. 새로 추가된 SM(Service Manager)은 메타 서비스를 관리하고 작업이 실행될 때 메타 서비스를 MSF 워크플로우로 변환하는 역할을 한다. 메타 서비스는 MSSF 시스템의 저장소에 MSML 파일로 저장된다. 그리고 사용자는 메타 서비스의 목록을 살펴보고, 서비스를 검색하고, 서비스 이름과 파라미터들로 서비스를 호출할 수 있다. 이 외에도 AM과 IR이 SM과의 통합을 위해 그 기능이 확장되었다. AM은 사용자의 작업 접근 뿐 아니라 서비스 접근도 관리할 수 있도록 확장되었다. IR은 SM의 메타데이터를 관리하기 위해 확장되었으며, (그림 7)에서는 단순성을 위해 생략되었다. MSF 시스템에 대한 자세한 정보는[16]에 있다.

MSSF 시스템의 사용자 시나리오를 살펴보면 다음과 같다. 먼저 사용자가 메타 서비스를 수행하기 위해 요청을 보내기 위해서는 AM을 통해 인증을 마친 다음, SM에서 해당 서비스를 검색한다. 사용자가 원하는 서비스를 발견하였다면, 서비스를 수행하거나 필요하다면 서비스 기술 내용을



(그림 7) MSSF 시스템 구조

수정하고 서비스를 제출한다. 사용자의 인증이 유효하다면, AM은 서비스 요청을 SM에 전달하고 서비스의 수행이 시작되기를 기다린다. 그 다음 SM은 서비스의 MSML을 파싱하여 JCML로 기술된 워크플로우로 변환한다. JCML 파일이 생성된 다음, SM은 서비스가 준비되었음을 알린 다음, RM에 JCML 워크플로우를 제출한다. JCML을 수행하는 이후의 과정은 MSF 시스템에서 작업을 수행하는 과정과 동일하다. MSF의 확장 예에서 볼 수 있듯이 메타 서비스를 적용하면 서비스를 관리하는 구성요소를 구현하고 통합하는 것만으로 기존의 워크플로우 시스템을 손쉽게 확장할 수 있다.

6. 결론 및 향후 연구

본 논문에서는 워크플로우들을 서비스와 통합할 수 있는 메타 서비스와 메타 서비스를 기술할 수 있는 MSML을 정의하였다. 또 메타 서비스를 적용하여 기존의 그리드 환경을 위한 워크플로우 시스템인 MSF 시스템을 MSSF 시스템으로 확장하였다. MSSF 시스템을 통해 사용자는 보다 쉽게 워크플로우를 사용할 수 있고, 그리드 포탈에 보다 용이하게 워크플로우 엔진을 통합할 수 있다.

MSML을 통해 정의된 서비스들은 기존의 워크플로우 서비스에 비해 많은 장점을 가진다. 메타 서비스를 통해 워크플로우를 그리드 서비스, 웹 서비스, 포탈 서비스 등의 다양한 서비스로 쉽게 이식시킬 수 있으며, 워크플로우를 단순하게 유지하여 워크플로우의 재사용성을 높일 수 있다. 또한 사용자 간의 서비스 공유를 가능하게 하고, 사용자가 서비스가 실행될 자원들을 제약하여 원하지 않는 자원에서 자원이 실행되는 것을 막을 수 있다. 마지막으로 과거의 성능 정보를 기록하여 그리드 환경에서 자원을 보다 효율적으로 스케줄링할 수 있도록 한다.

메타 서비스와 향후 관련되어 계획하고 있는 연구들은 다음과 같다. 우선 MSML에서는 물리적인 서비스 인터페이스들과의 통합만을 지원하고 있으나, 메타 서비스의 상위 계층에 온톨로지를 적용한다면 서비스의 의미적 오류를 검증할 수 있을 것이다. 그리고 현재 MSML에서는 MSF 시스템의 워크플로우 만을 워크플로우 유닛으로 사용할 수 있지만 다양한 워크플로우들을 워크플로우 유닛으로 사용하게 하면, 여러 종류의 상이한 워크플로우 유닛을 통합하여 사용할 수 있는 환경도 제공할 수 있을 것이다. 또한 MSF에서 사용되었던 워크플로우 편집기와 같이 MSML을 손쉽게 편집하고 사용할 수 있는 편집기를 개발하면 MSML을 사용자가 손쉽게 편집할 수 있을 것이다. 마지막으로, MSSF 시스템에서 제공하는 메타 서비스와 워크플로우에는 응용 프로그램들을 그리드 상에서 설치하고 관리하는 기능이 없다. 따라서 이러한 패키지 기능을 MSSF에 통합한다면 그리드 환경에서 서비스, 워크플로우, 응용프로그램을 통합하여 관리할 수 있으므로, 그리드 시스템 구축에 보다 유용하게 사용될 수 있을 것이다.

참 고 문 헌

[1] I. Foster and C. Kesselman, ed., "The Grid: Blueprint for a New Computing Infrastructure," Morgan Kaufmann, 1998.

[2] I. Foster, C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit," Intl J. Supercomputer Applications, 11(2):115-128, 1997.

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

[4] DAGMan, http://www.cs.wisc.edu/condor/dagman.

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

[6] Seogchan Hwang, Jaeyoung Choi and Hyeongwoo Park, "Meta Scheduling Framework for Workflow Service on the Grids," International Conference on Computational Science 2004, pp.445-448.

[7] M. Litzkow, M. Livny and M. Mutka, "Condor-A Hunter of Idle Workstations," 8th International Conference of Distributed Computing Systems, pp.13-17, 1998.

[8] G. R. Nudd et al., "PACE-A Toolset for the Performance Prediction of Parallel and Distributed Systems", Int. J. High Performance Computing Applications, Special Issues on Performance Modelling-Part I, Vol.14, No.3, pp.228-251, 2000.

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

[10] T. Oinn et al., "Taverna: A tool for the composition and enactment of bioinformatics workflows," Bioinformatics Journal 20(17) pp.3045-3054, 2004,

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

[12] GRMS(Gridlab Resource Managemet System), http://gridlab.org/WorkPackages/wp-9/index.html.

[13] 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.

[14] W. Smith, I. Foster, V. Taylor, "Predicting Application Run Times Using Historical Information," Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing, 1998.

[15] JAXB(Java Architecture for XML Binding), http://java.sun.com/xml/jaxb.

[16] 황석찬, 최재영, "그리드 환경에서 워크플로우 서비스를 제공하기 위한 메타 스케줄링 프레임워크," 정보과학회논문지: 컴퓨팅의 실제 제10권 제5호, pp.375-384, 2004.

이 상 근

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

최 재 영

e-mail : choi@ssu.ac.kr
1984년 서울대학교 제어계측공학과(학사)
1986년 미국 남가주대학교 전기공학과(석사)
1991년 미국 코넬대학교 전기공학과(박사)
1992년~1994년 미국 국립오크리지연구소 연구원
1994년~1995년 미국 테네시 주립대학교 연구교수
2001년~2002년 미국 국립 수퍼컴퓨팅 응용센터 (NCSA) 초빙연구원
1995년~현재 숭실대학교 정보과학대학 컴퓨터학부 부교수
관심분야 : 고성능컴퓨팅(HPC), 병렬/분산처리, 유비쿼터스 컴퓨팅

황 석 찬

e-mail : seogchan@kisti.re.kr
1998년 숭실대학교 컴퓨터학과(석사)
2003년 숭실대학교 컴퓨터학과(박사)
2003년~2004년 숭실대학교 시스템소프트웨어연구실 박사후과정
2004년~현재 한국과학기술정보연구원 선임연구원
관심분야 : 분산/병렬 컴퓨팅, 그리드 컴퓨팅, 시스템 소프트웨어, 미들웨어