응용 그리드를 위한 워크플로우 시스템 설계

심규호[0]* 황선태* 정갑주** 박형우***
*국민대학교 컴퓨터학부, **건국대학교 인터넷미디어공학부, ***한국과학기술원
{simsaint*, sthwang*}@cs.kookmin.ac.kr, {jeongk**}@imc.konkuk.ac.kr. {hwpark***}@kisti.re.kr

Design of Worflow System for Application Grids

Gyuho Sim[0]* Suntae Hwang* KarpJoo Jeong** Hyungwoo Park***
*Department of Computer Science, Kookmin University
**College of Information and Communication, Konkuk University
***Korea Institute of Science and Technology and Information

요 약

자연 과학 분야를 연구하는 연구자들은 자신이 연구하는 실험에 대해 전체 혹은 일부를 컴퓨터를 사용하여 실험하게 된다. 이러한 작업의 절차는 각각 연구자마다 다양하며 어느 하나의 방법을 강요할 수 없다. 본 논문은 응용 연구자들이 쉽게 자신의 실험 절차를 정의가 가능하고 보다 유연하게 작업 절차를 정의할 수 있는 워크플로우 모델을 제시한다. 특히 분산 리소스를 통합하여 사용할 수 그리드 환경에서 적용할 수 있는 워크플로우 모델을 제시하여 KISTI 그리드 테스트베드 상에 구축된 MGrid 시스템[1]에 적용하고 다른 그리드 응용 플랫폼에서 적용할 수 있는 방향을 제시해본다.

1. 서 론

최근 분산된 컴퓨팅 자원을 공유하여 사용하기 위한 그리드 컴퓨팅의 기술이 발전하고 있다. 특히 자연과학 분야에서의 엄청난 양의 컴퓨팅 파워와 대용량의 저장매체를 요구는 그리드 컴퓨팅 기술의 발전을 더욱 가속화 시키고 있다. 현재 글로버스 툴킷[2]을 기반으로 하는 미들웨어의 연구는 활발한 반면 그리드 미들웨어를 기반으로 하는 응용 플랫폼의 연구는 미비한 단계이다. 본 논문에서는 바이오/나노 연구자들의 워크플로우를 분석하고 이를 기반으로 하여 그리드 응용 플랫폼인 MGrid 시스템에 적용할 워크플로우 모델을 제시한다.

2. 관련연구

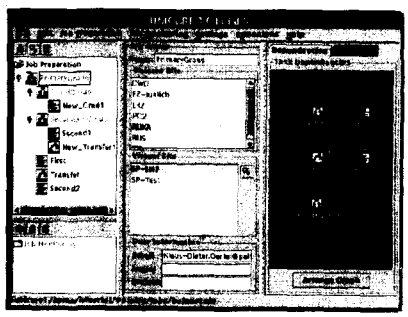
워크플로우는 특정 목표에 도달하기 위한 문서, 정보 혹은 작업 프로세스에 대한 전체 혹은 일부를 이미 정의된 규칙을 통한 자동화로 정의할 수 있다[3]. 현재 많은 시스템에서는 다양한 워크플로우 모델을 정의하고 있으며 각각의 워크플로우 모델은 각각의 특징을 가지고 있다.

2.1 Grid Workflow

Grid Workfow[4]는 Grid Computing Environment Working Group에서 그리드 환경에서 적용할 워크플로우를 정의하고 있으며 워크플로우를 표현하기 위해 XML DTD를 사용하여 XML 다큐먼트 타입을 정의하고 있다. Grid Workflow에서는 Data Transfer, Computation, ResourceQuery, RestartLoop, ForkTask로 컨트롤 흐름을 정의하고 있으며 모든 컨트롤의 흐름을 자동화 한다. 동시에 여러 개의 컨트롤 흐름을 Grid Workflow 모델에서는 제시하지 못하고 있으며 만약 동시에 여러개의 작업을 수행하기 위해서는 서로 분리된 컨트롤 흐름을 정의해야 한다.

2.2 UNICORE

UNICORE(UNiform Interface to COmputing REsource)[5]는 분리된 슈퍼 컴퓨터와 같은 고성능의 컴퓨터 리소스를 통합하려는 취지에서 시작하여 현재 EURO 그리드의 중심축으로 자리잡고 있다. UNICORE는 Job, SubJob, Task의 계층적 구조를 가지는 실행 프로세스를 정의하고 있으며 Job, SubJob은 하위 실행 프로세스에 대해 [그림 1]과 같은 워크플로우 편집기를 통해 실행 순서를 정의할 수 있다. File Import, Processing, File Export등으로 구성된 Plug-in 형태의 커스터마이즈된 Task를 프로그래밍을 통해 생성할 수 있어 보다 유연한 프로세스 모델을 제공한다.



[그림 1] UNICORE Workflow Editor

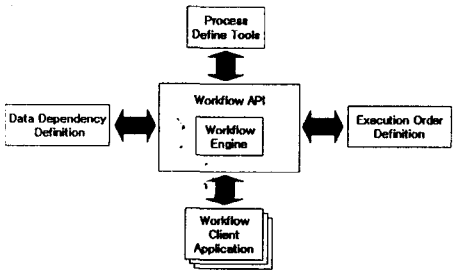
Job, SubJob 내부의 실행 순서는 [그림 1]과 같이 DAG 형태의 그래픽 뷰어를 통해 정의할 수 있으며 각 Task의 실행 스케쥴링은 NJS(Network Job Server)라는 UNICORE 서버에서 수행하게 된다.

2.3 Condor의 DAGMan

DAGMan은 Condor에서 여러 개의 Job들의 실행 순서를 DAG 형태로 정의하고 그 정의된 그 실행순서에 따라 실행시키기 위한 메타 스케쥴러이다. DAG(Directed Acyclic Graph) 형태로 정의된 Job들은 submit되기 전에 입력파일들은 정의되어져 있어야 하며 DAGMan은 Submit 되어진 프로그램 집합에 대해서 스케쥴링, 회복, 리포팅 등을 역할을 수행한다.

3. 워크플로우 설계

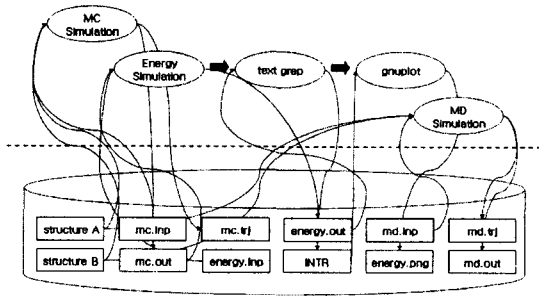
위에서 정의된 워크플로우 모델은 완전 자동화된 컨트롤 흐름 가지고 있으며 컨트롤 흐름 중 중간의 실패가 발생할 경우 다음 컨트롤 흐름은 실패로 간주한다. 이것은 많은 컴퓨팅 리소스를 필요로 하는 응용 연구자들이 자신의 작업 흐름을 정의 할 때 작업의 흐름 중 중간의 작업 정의를 실수할 경우 맨 처음부터 다시 실험을 해야 하는 부담을 가중시킨다. 또한 워크플로우를 통해 정의된 컨트롤 흐름이 진행 중에 연구자의 간섭의 여지를 제공하지 못한다. 본 논문에서는 응용 연구자들의 실험 패턴을 분석해 본 결과 다음의 결론을 얻을 수 있었다.
응용 연구자들은 자신만의 워크플로우를 가지고 있으며 응용 연구자들은 자신만의 워크플로우의 전체 혹은 일부를 컴퓨터를 사용하여 처리하며 각 단계마다 일정한 규칙에 의해서 혹은 연구자들의 직접적인 판단을 통해 워크플로우의 방향이 결정되곤 한다. 이는 응용 연구자의 워크플로우 전체를 자동화 하는 것이 불가능함을 의미한다. 이것은 워크플로우 흐름 중 사용자의 개입이 필요한 부분을 허용하는 워크플로우 모델을 정의해야 함을 의미한다. 이러한 워크플로우의 특성을 정의하기 위해서는 자동화 시킬 수 있는 작업들 간의 흐름과 응용연구자의 개입이 필요한 작업으로 구분된 워크플로우 모델의 설계를 필요로 한다.



[그림 2] 워크플로우 시스템

응용 연구자의 워크플로우의 흐름은 전체 워크플로우의 자동화가 불가능하다. 이는 워크플로우 흐름중 사용자의 개입에 필요한 부분을 앞에서 정의한 워크플로우 모델에서는 정의가 불가능하기 때문이다.

본 논문에서 제시할 워크플로우는 [그림 2]의 구조를 가지는 워크플로우 시스템을 제안한다. 본 워크플로우 시스템은 프로세스 정의, 데이터 의존성 정의, 실행순서의 정의를 분리한다. 본 워크플로우 시스템의 특징은 [그림 3]과 같이 실행 프로세스의 흐름과 데이터 의존성을 서로 다른 공간으로 분리하여 정의한 것이다. 실행 프로세스의 흐름과 데이터 의존성을 서로 다른 공간으로 분리함으로써 사용자에게 보다 직관적이고 유연한 프로세스를 정의할 수 있는 환경을 제공한다. 실행 프로세스의 흐름은 자동화 할 수 있는 모델과 동시에 다수의 실행 프로세스를 실행 시킬 수 있는 모델 모두 정의 하고 있으며 각 실행 프로세스에서 사용되거나 생성하는 데이터를 다른 실행 프로세스에서 사용할 수 있는 가능성을 제공한다.

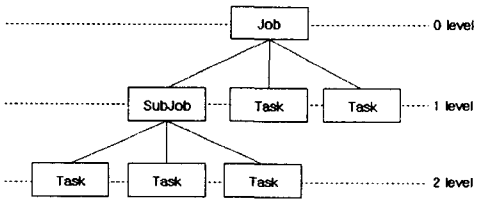


[그림 3] 실행순서와 데이터 의존성 공간 분리

3.1 워크플로우 프로세스 모델

본 논문에서 정의한 워크플로우에서는 다음과 같은 프로세스 모델을 정의한다.

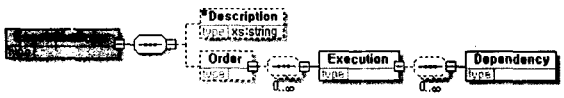
가) Job : 서로 연관된 Task를 관리하는 상위모델로써 Task, SubJob으로 구성되며 각각의 실행은 사용자의 컨트롤을 통해서만 이루어진다.
나) SubJob : 연관된 Task로 구성된 모델로써 SubJob에 속해 있는 Task간에는 실행 순서를 가지며 워크플로우 엔진을 통해 각각 Task가 수행된다.
다) Task : 시뮬레이션, 모니터링, 이벤트 검출을 위한 실행 코드를 포함하며 각 Task가 수행에 필요한 각 입/출력파일들 간의 의존성 정보를 담고 있다.



[그림 4] 2-Level 워크플로우 프로세스

3.2 실행 순서 정의

본 워크플로우 모델은 [그림 4]에 정의된 프로세스 모델에 대해서 1-level의 프로세스는 사용자의 컨트롤에 의존하고 2-level의 프로세스의 흐름은 실행 순서에 따른 자동화 프로세스 흐름으로 정의한다. 다음은 2-level 프로세스의 흐름을 자동화를 정의한 SubJob 내에서 Task간의 실행 순서를 정의한 XML 스키마 문서를 보여준다.



[그림 5] 실행순서를 정의한 XML 스키마

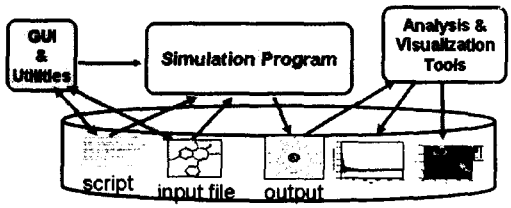
[그림 5]에 정의된 XML 스키마를 보면 ExecutionOrder라는 루트 요소를 정의하고 있으며 하위 요소로써 Description과 Order를 정의하고 있다. Desciption 요소는 Task실행 순서에 대한 설명을 포함하고 있으며 Execution 요소는 각 Task의 실행 의존성을 정의한다. Execution의 하위요소로써 Dependency요소는 각 Execution이 실행하기 위해 이전에 수행되어 종료 되어야 할 Task에한 의존성을 정의한다. [그림 6]은 Task간 실행순서를 정의한 XML 스키마를 기반으로 한 예제로써 SubJob안에서의 Task간의 실행 순서를 정의하고 있다.

```
<?xml version="1.0" encoding="UTF-8"?>
<SubJob xmlns="http://mgrid.or.kr/pse/DAG-Define"
  xmlns:wdl="http://mgrid.or.kr/workflow/ExecutionOrder"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://mgrid.or.kr/pse/JobDefine D:\WorkflowAjob.xsd">
    <wdl:Workflow>
    <wdl:Description>Test </wdl:Description>
    <wdl:Order>
        <wdl:Execution id="task-001"/>
            <wdl:Execution id="task-002">
                <wdl:Dependency ref="task-001"/>
            </wdl:Execution>
            <wdl:Execution id="task-003">
                <wdl:Dependency ref="task-001"/>
                <wdl:Dependency ref="task-002"/>
            </wdl:Execution>
        </wdl:Order>
    </wdl:Workflow>
    <Definition>
        <Task wdl:id="task-001"/>
        <Task wdl:id="task-002"/>
        <Task wdl:id="task-003"/>
    </Definition>
</SubJob>
```

[그림 6] 실행 순서를 정의한 XML 문서

3.3 데이터 의존성 정의

응용 분야에서 필요로 하는 작업들은 기존 데이터를 이용하여 새로운 데이터를 생성한다. 이러한 관계는 실행 프로세스 순서와 맞물려 데이터간의 흐름을 정의할 수 있다. 각 Task는 필요로 하는 입력 파일을 다른 Task에 의해 생성된 파일과의 관계를 통하여 데이터 의존성을 정의할 수 있다. 데이터 의존성은 Active File System[7]에서 정의하고 있어 Task의 실행은 실행에 필요로 하는 입력 파일의 유무에 따라 Task의 실행 여부를 결정할 수 있다. [그림 7]의 구조를 가지는 Active File System은 일반 파일과 그것의 동적인 상태를 표현한 메타 정보로 구성된 논리적인 파일 즉 ActiveFile로 구성된 가상 파일 시스템이다. Active File System에서는 ActiveFile이외에 각 입/출력 파일간의 데이터 의존성을 정의하고 있는 Creator와 ActiveFile의 공통적인 특성을 그룹지은 ProductList를 정의하고 있다.



[그림 7] Active File System

4. 결론 및 향후 과제

본 논문은 Task 실행 순서와 데이터 의존성을 별도의 공간으로 분리한 워크플로우를 제안하였다. 현재 구현중인 MGrid PSE(Problem Solving Environment)는 Active File System 기반으로 하여 Task의 실행순서와 데이터 의존성을 분리된 공간에서 정의하고 있다. 현재 데이터 의존성에 따라 실행 프로세스의 실행을 사용자에 의한 컨트롤을 통해 수행되고 있으며 향후 각 Task간의 실행 순서 정의에 따른 프로세스의 자동화를 구현할 계획이다. 그 외에 실행 순서와 데이터 의존성간의 유효성을 검사하여 Job 생성단계에서의 Task의 유효성 검사와 SubJob 아래에 SubJob이 존재할 수 있는 재귀적 구조를 정의하여 보다 유연한 워크플로우 모델을 제시할 예정이다.

5. 참고 문헌

[1] http://imc.konkuk.ac.kr/~mgrid
[2] http://www globus.org
[3] David Hollingsworth "Workflow Management Coalition The Workflow Reference Model" 19-Jan-95"
[4] Hugh P. Bivens, Grid Computing Environment Working Group, Grid Workflow"
[5] http://www.unicore.org
[6] http://www.w3.org/XML/Schema
[7] 홍상현, 컴퓨팅 그리드에서의 HPC 사용자를 위한 PSE 설계 및 구현, 석사학위 논문, 국민대학교, 전산과학과, 2003