분산 환경에서 CFD 분석 프로그램 수행을 위한 그리드 시스템 META 설계 및 구현

강 경 우† · 우 균††

요 약

본 논문에서는 분산 환경 상에서 CFD(Computational Fluid Dynamics) 분석 프로그램을 편리하게 수행할 수 있도록 하는 그리드 시스템 META(Metacomputing Environment using Test-run of Application)의 설계 및 구현에 관하여 기술한다. 그리드 시스템 META는 CFD 프로그램 개발자들이 네트워크에 분산된 계산 자원들을 단일 시스템처럼 사용할 수 있도록 한다. 그리드 컴퓨팅과 관련하여 연구주제로는 고장허용, 자원 선택, 사용자 인터페이스 설계 등이 있다. 본 논문에서는 MPI(Message Passing Interface)로 작성된 SPMD(Single Program, Multiple Data) 구조의 병렬프로그램을 실행시키기 위한 자동 자원 선택방법을 활용하였다. 본 논문에서 제안한 자원 관리기법은 네트워크상의 전송지연시간과 시험수행을 통해 얻어진 핵심루프의 경과시간을 이용한다. 전송지연시간은 병렬 프로그램이 복수의 시스템에 분산되어 수행될 때 수행성능에 큰 영향을 주는 요인이다. CFD 프로그램들의 공통적인 특성 때문에 핵심루프 경과시간은 전체 수행시간을 예측할 수 있는 지표가 된다. 핵심루프는 CFD 프로그램의 전체 수행시간 중 90% 이상을 차지한다.

키워드 : CFD, 그리드 컴퓨팅, 자동 자원 선택, 메타컴퓨팅

Design and Implementation of a Grid System META for Executing CFD Analysis Programs on Distributed Environment

Kyung-Woo Kang† · Gyun Woo††

ABSTRACT

This paper describes the design and implementation of a grid system META (Metacomputing Environment using Test-run of Application) which facilitates the execution of a CFD (Computational Fluid Dynamics) analysis program on distributed environment. The grid system META allows the CFD program developers can access the computing resources distributed over the network just like one computer system. The research issues involved in the grid computing include fault-tolerance, computing resource selection, and user-interface design. In this paper, we exploits an automatic resource selection scheme for executing the parallel SPMD (Single Program, Multiple Data) application written in MPI (Message Passing Interface). The proposed resource selection scheme is informed from the network latency time and the elapsed time of the kernel loop attained from test-run. The network latency time highly influences the executional performance when a parallel program is distributed and executed over several systems. The elapsed time of the kernel loop can be used as an estimator of the whole execution time of the CFD program due to a common characteristic of CFD programs. The kernel loop consumes over 90% of the whole execution time of a CFD program.

Key Words : CFD, Grid Computing, Automatic Resource Selection, Metacomputing

1. 서 론

슈퍼컴퓨터의 수가 많아지고 네트워크의 속도가 빨라짐에 따라 그리스 시스템에 대한 필요성이 증대되고 있다. 그리드 시스템은 서로 다른 컴퓨터 시스템들을 고속 네트워크로 연결하여 하나의 문제를 해결하는데 활용하는 도구이다. 컴퓨팅 자원들은 같은 건물에 있을 수도 있고 멀리 떨어져 있을 수도 있다. 그러나 그리드 시스템은 네트워크에 연결된 컴퓨팅 자원들을 마치 단일 시스템인 것처럼 사용자에게 제공해야 한다.

그리드 환경에서 과학계산 어플리케이션을 수행하는데 발생하는 기술적인 어려움은 컴퓨팅 자원들 간의 네트워크 속도와 자원 간 성능 차에 기인한다. 네트워크 속도가 빠르면

† 정 회 원 : 천안대학교 정보통신학부 조교수
†† 종신회원 : 부산대학교 정보컴퓨터공학부 조교수(교신저자)
논문접수 : 2005년 10월 13일, 심사완료 : 2006년 9월 5일

성능이 낮은 컴퓨팅 자원에서 수행시켜도 병렬 작업 간에 전송대기시간이 짧기 때문에 빨리 처리할 수 있다. 그러나 네트워크 속도가 느리면 아무리 컴퓨팅 자원의 속도가 빨라도 전송지연시간으로 인해 전체 수행시간은 현저히 떨어진다.

동일하거나 비슷한 네트워크 환경일 경우에는 컴퓨팅 자원의 성능이 전체 수행시간에 영향을 미친다. 그렇지만 단순 성능비교 만으로는 알 수 없다. 작업의 특성에 따라 메모리가 큰 시스템이 적합할 수도 있고 입출력 성능이 좋은 시스템이 적합할 수도 있기 때문이다. 본 논문에서는 네트워크 속도 측정과 컴퓨팅 자원의 성능 측정을 통해 병렬작업을 할당할 컴퓨팅 자원 선택방법을 제안한다. 선택된 사용가능한 컴퓨팅 자원들의 집합은 다른 어떤 집합 사용가능 집합보다 작업을 효율적으로 처리할 수 있다.

최적의 컴퓨팅 자원 집합을 얻기 위해 본 논문에서 제안한 방안은 네트워크 연결 상태를 가중치 그래프(weighted graph)로 표현하는 것이다. 여기서 가중치란 네트워크 지연시간을 의미하는데, 주어진 그래프 상에서 최소 가중치를 갖는 최대 클리크(maximal clique)를 구하면 최적의 컴퓨팅 자원 집합 후보가 될 수 있다. 선택된 후보 집합들 중에 CFD(Computational Fluid Dynamics) 프로그램을 효과적으로 수행할 수 있는 것을 선택한다.

컴퓨팅 자원의 성능 측정을 위해 본 논문에서는 CFD 프로그램의 핵심루프(kernel loop)를 이용한다. CFD 프로그램은 같은 패턴의 계산을 반복하여 수행한다는 특징이 있다. CFD 프로그램에서 핵심루프란 전체수행시간의 대부분을 차지하며 수없이 반복되는 메인루프를 말한다. 본 논문에서는 핵심루프를 자동으로 추출하여 대상이 되는 컴퓨팅 자원에서 시험수행을 거친 결과에 따라 자원을 할당하는 방법을 제시한다. 시험수행에 필요한 시간은 불과 수초에 해당하기 때문에 시험수행으로 인한 오버헤드는 크지 않다. 본 논문에서 개발한 그리드 시스템 META(Metacomputing Environment using Test-run of Application)는 네트워크 상태를 기반으로 최대 크리크 후보들을 선택하고 선택된 자원들 상에서 시험수행을 하여 가장 빠른 자원들을 선택하게 된다.

본 논문의 구조는 다음과 같다. 2절에서는 본 논문의 배경 연구를 기술하고, 3절에서는 2절에서 기술한 내용, 특히 CFD 분석 프로그램에 대한 배경 지식을 바탕으로 자원선택 기법을 제시한다. 4절에서는 META에 대한 설계 및 구현내용을 기술한다. 좀 더 자세히 설명하면, META 시스템 전체 구조와 핵심 루프에 대한 실험, 사용자 인터페이스 내용 및 CFD 프로그램 수행 결과 등을 기술한다. 5절에서는 다른 그리드 컴퓨팅 시스템에 대한 META의 특징을 살펴본다. 특히 META와 유사한 기능을 수행하는 MOL과 다각도에서 비교한다. 끝으로 6절에서 결론을 기술한다.

2. 배경 연구

이 절에서는 META 개발에 배경이 되는 관련 연구를 기술한다. META는 CFD 프로그래머가 그리드 컴퓨팅 시스템 내의 계산 자원들을 마치 하나의 시스템처럼 이용할 수 있도록 한다. 따라서 이 절에서는 그리드 컴퓨팅 시스템 내의 자원을 활용하기 위한 플랫폼을 제시하기 위한 과거 연구를 살펴본다. 또한 META가 목표로 하는 주요 응용 프로그램인 CFD 프로그램의 특징에 대해서도 살펴본다.

2.1 그리드 컴퓨팅 시스템 개발 관련 연구

2.1.1 MOL(Metacomputer Online)

MOL은 WAN환경 상에 연결된 고성능 시스템들을 마치 하나의 컴퓨팅 자원처럼 사용자에게 제공하는 시스템이다[7,8]. 여러 개의 컴퓨팅자원들을 연결하는 것은 하나의 자원에서 해결할 수 없는 큰 크기의 문제를 해결하는데 도움을 준다. MOL 프로젝트의 목표는 기존의 모듈들을 연결하여 하나의 개방형 환경을 만드는 것이다. 현재 개발된 시스템은 고성능 망으로 연결된 Parsytec GC, Intel Paragon, IBM SP2 등 상에서 PVM, MPI, PARIX로 작성된 응용프로그램들을 지원한다. MOL에서 고려한 이슈들은 다음과 같다.

- 편리한 사용자 인터페이스
- 서로 다른 메시지 전달을 이용하는 기존의 코드의 연동을 지원
- 처리율(throughput)을 높이고 대기시간을 줄이기 위해 각 컴퓨팅 시스템들을 글로벌로 관리
- 원격 컴파일 및 데이터의 분산 지원
- 계산에 사용될 적당한 컴퓨팅자원을 자동 선정
- 이기종 컴퓨팅 자원들 상에 사용자 작업의 성능을 예측함으로 실행 중 작업을 이동시킬 수 있음

2.1.2 Legion

버지니아 대학에서 수행중인 과제로서 객체지향 그리드 시스템을 개발하는 것이다[9]. Legion 프로젝트에서는 사용자들에게 하나의 가상 컴퓨터를 제공하는 시스템을 만들고자 하고 있다. 개발하는 가상 컴퓨터는 병렬처리를 이용하여 응답시간이 빠를 뿐만 아니라 높은 처리율을 제공하는 것이다. Legion은 워크스테이션 클러스터들과 워크스테이션들, 슈퍼컴퓨터들, 병렬 컴퓨터들을 대상으로 하고 있다. 이들의 개발 방법은 기존의 객체지향 병렬처리 시스템에 이기종 분산처리 시스템의 특징들을 가미하는 방법을 취하고 있다. 기존의 것으로 활용하는 것은 이들이 이미 개발한 Mentat라는 객체지향 병렬처리 시스템이다. Legion 프로젝트에서 개발된 도구를 이용하여 문제를 해결하고자 한다면, 사용자는 객체지향 프로그래밍 기법을 충분히 이해하여야 하고 많은 API들의 사용법을 익혀야 한다.

2.1.3 Globus

통신, 자원 할당, 데이터 접근 등 이기종 컴퓨팅 인프라의 요구사항들을 낮은 수준의 메커니즘들을 이용하여 제공한다[3]. 이들 메커니즘은 스케줄러와 병렬프로그래밍 도구 등 상위 레벨의 이기종 컴퓨팅 서비스들을 구현하는데 사용된다. 구성요소들은 각각 통신, 자원을 찾고 할당, 자료 접근,

인증(authentication)을 위한 요소들이다.

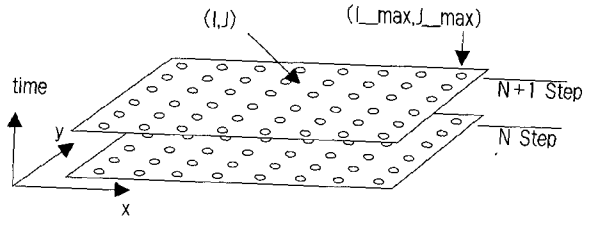
- 통신기능: 메시지 전달, 원격 절차 호출(remote procedure call), DSM 등 다양한 통신방법을 제공
- 자원할당: 응용프로그램이 필요한 자원들을 표현하고 응용프로그램이 필요에 맞는 자원을 찾고 스케줄링 하는 메커니즘
- 통합된 자원정보 서비스: 이기종 컴퓨팅 환경의 구조와 상태에 관한 실시간 정보를 얻는 메커니즘
- 데이터 접근: 원격 데이터와 파일에 고속으로 접근할 수 있게 하는 모듈
- 프로세스 생성: 자원이 할당되면 할당된 자원 상에서 계산을 시작하게 하는 모듈로써 실행파일을 준비하고 매개변수들을 전달하고 새로운 프로세스를 실행시키고 끝내는 일을 맡는 부분
- 인증 인터페이스: 사용자와 자원의 ID를 구분해 주는 모듈

2.2 CFD 분석 프로그램

CFD는 유체유동현상을 지배하는 Navier-Stokes 방정식의 근사해를 컴퓨터를 이용하여 구하는 방법이다. Navier-Stokes 방정식은 연립편미분방정식으로 이에 대한 해석적인 해를 구하는 것은 거의 불가능하다. 이러한 이유로 컴퓨터의 발전과 더불어 이에 대한 수치적인 해를 근사적으로 구하고자 하는 노력이 계속되어 왔으며 현재는 고속의 병렬컴퓨터를 이용하여 단시간에 이에 대한 해를 구하고 있다.

CFD에서는 연립편미분방정식인 Navier-Stokes 방정식을 적절한 가정을 도입하여 컴퓨터에서 해석 가능한 대수적인 연립방정식으로 변환하여 이에 대해서 반복적인 방법으로 해를 구하게 된다. 따라서 CFD 분석 프로그램은 변환된 연립대수방정식에 대한 해를 구하는 프로그램이라고 생각하면 된다. 그러나 이러한 대수방정식은 선형연립방정식이 아니라 각각의 방정식이 서로 연관되어 있는 비선형연립방정식이 된다. 따라서 이에 대한 해석은 시간항과 공간항이 밀접하게 관련이 되어 있어 각 시간단계에 대해서 공간적으로 모든 계산영역에 대해서 반복적으로 해를 구하게 된다.

그림 1에 나타난 바와 같이 해석프로그램은 각 N단계에서 x와 y방향(3차원의 경우 x, y, z)으로 각각의 격자점 (I, J)에서 연립방정식의 해를 구하게 된다. 그리고 이때 각각의 격자점 (I, J)에서의 연립방정식은 적어도 M(M≥4)개의 방정식으로 구성되므로 M×M 행렬의 역행렬을 구해야 한다.



(그림 1) CFD 분석 프로그램의 기본개념

```
      program NS2D
      . . . .
C     Reading Data
      do n = 1, Nstep_Max
      do I = 1, I_Max
      do J = 1, J_Max
      do n1 = 1, M
      do n2 = 1, M
      . . . .
      x(n2) = A(n1,n2) * b(n2)
      . . . .
      enddo
      enddo
      enddo
      enddo
      enddo
      . . . .
C     Writing Data
      stop
      end
```

(그림 2) CFD 프로그램의 일반적인 형태

그러므로 CFD 프로그램은 많은 중첩 루프들을 포함하고 있는 알고리즘으로 표현할 수 있다. 그림 2와 같은 프로그램의 구조를 살펴보면 가장 바깥의 루프는 시간축을 따라 반복하는 루프로서, Nstep_Max 회 반복된다고 가정하면 전체 계산수행시간은 (한 Step의 계산시간 × Nstep_Max) 정도가 된다고 할 수 있다. 게다가 CFD 프로그램은 각각의 N 루프에서의 계산시간이 프로그램 수행 중에 거의 일정하다는 특징이 있다.

3. 자원 선택 기법

3.1 작업집합 후보 선정

그리드 시스템의 네트워크 속도는 CFD 프로그램의 수행성능에 큰 영향을 미친다. 이는 CFD 프로그램의 특성상 개별 병렬 태스크들이 매 시간 단계마다 서로 통신해야 하기 때문이다. 따라서 그리드 환경에 태스크들을 배포할 경우에는 개별 프로세싱 단위의 성능보다 네트워크 속도를 먼저 생각해야 한다. 이 절에서는 그리드 환경에서 CFD 프로그램을 가장 빨리 수행할 수 있는 프로세싱 단위들의 최소 집합을 구하는 알고리즘을 기술한다.

META에서의 컴퓨터 선택 알고리즘을 기술하기 위해 먼저 네트워크 비용 그래프(network latency-time graph)를 생각해 보자. 네트워크 비용 그래프는 다음과 같이 정의된다.

[정의 1] (**네트워크 비용 그래프**) 그리드 시스템의 **네트워크 비용 그래프**는 무방향 가중치 그래프

$$G=(M,E)$$

로 정의되는데 여기서 M은 그리드 시스템 내의 컴퓨터들의 집합이고 E는 모든 컴퓨터 사이에 놓인 개념적인 통신 선로이다. 그리드 시스템이 n개의 컴퓨터로 구성되어 있다고 하면, M과 E는 다음과 같이 생각해 볼 수 있다.

$$M=\{m_1,m_2,\cdots,m_n\}$$
$$E=\{e \mid e \subset M, |e|=2\}$$

네트워크 비용 그래프는 두 개의 함수, P_t($P_t: M \rightarrow \boldsymbol{N}$, 여기서 $\boldsymbol{N}$은 자연수의 집합) 및 L_t($L_t: E \rightarrow \boldsymbol{R}$, 여기서 $\boldsymbol{R}$은 실수의 집합)와 함께 정의되는데, $P_t(m)$은 시각 t에서 컴퓨터 m 내의 가용 프로세싱 단위의 개수를 나타내며 $L_t(\{m_1,m_2\})$는 시각 t에서 m_1과 m_2 사이의 통신 시간을 나타낸다. 시각 t가 현재 시각을 나타내는 경우에는 첨자 t를 생략하고 그냥 $P(m)$ 또는 $L(\{m_1,m_2\})$와 같이 쓴다. 함수 P와 L은 각각 노드 집합 M과 에지 집합 E를 인수로 받도록 확장할 수 있다.

$$P(M)=\sum_{m\in M} P(m)$$

$$L(E)=\sum_{e\in E} L(e)$$

하나의 에지 $e \in E$는 두 개의 다른 노드들의 집합으로 정의되었다는 것에 주의하자. 따라서 $n=|M|$일 때, 에지 개수 $|E|$는 $n(n-1)/2$이다. 즉 그리드 시스템 내의 컴퓨터 수가 2보다 작은 경우, 즉 $|M|<2$인 경우에 에지 집합은 공집합($|E|=0$)이다.

이렇게 네트워크 비용 그래프를 정의하였을 때, META의 목표는 네트워크 전송 시간이 가장 빠른, 네트워크 비용 그래프의 완전 서브그래프(complete subgraph)를 찾는 것이다. 이렇게 찾은 완전 서브 그래프는 CFD 프로그램을 수행하는데 사용될 컴퓨터들의 집합이 된다. 이러한 완전 서브 그래프의 노드들을 우리는 해당 작업에 대한 작업집합(working set)이라고 명명하겠다. 결국 META의 목표는 작업집합 후보를 선정하는 것인데, 작업집합 후보 선정 기준은 통신 시간이 된다. 작업집합 후보 선정 알고리즘을 기술하면 알고리즘 1과 같다.

알고리즘 1의 2행에서는 유효한 작업집합 후보들의 집합 C를 선정하는 부분인데, 여기서 유효한 작업집합이란 해당 작업을 수행하기에 충분한 프로세싱 단위를 포함하고 있는 것을 의미한다. 만약 전체 그리드 시스템 G 내에 해당 작업을 수행하기에 충분한 프로세싱 단위가 없다면 C는 공집합이 되고 따라서 W도 공집합이 되며 4행의 while 루프를 바로 빠져나가게 된다. 이 경우에는 어떤 작업집합 후보도 구할 수 없다. 그러나 통상 전체 그리드 시스템 G에는 충분한 수의 컴퓨터가 있을 것이므로 해당 작업을 수행하는데 필요한, 하나 이상의 작업집합 후보를 선정할 수 있을 것이다. 작업집합 후보가 여러 개인 경우에 작업집합 후보의 개수는, 인수로 주어진 *Number Of Candidates*로 제한된다. 어떤 경우든지 알고리즘 1은 작업집합 후보들로 구성된 유한 집합을 반환하며 종료한다.

알고리즘 1: 작업집합 후보 선정 알고리즘

입력: 네트워크 비용 그래프 $G=(M_G, E_G)$
Number Of PENeeded = 작업 수행에 필요한 프로세싱 단위 개수
Number Of Candidates = 선정할 작업 집합의 최대 개수
출력: 작업집합 후보들의 집합 W(여기서 각 $w \in W$는 $w \subset M_G$이다)

1. $M_G := G$의 노드 집합
2. $C := \{M \mid M \subseteq M_G, P(M) \geq Number of PENeeded\}$
3. $W := \varnothing$
4. **while** $|W| < Number Of Candidates$ **and** $C \neq \varnothing$ **do**
5. $L(\{e \mid e \subset M, |e|=2\})$가 최소가 되는 $M \in C$ 선정
6. $C := C - \{M\}$
7. $W := W \cup \{M\}$
8. **return** W

3.2 핵심루프 모델

2.2절에서 설명한 것처럼 CFD 프로그램들 내에는 전체 수행시간의 대부분을 소비하는 루프가 존재한다. 본 논문에서는 CFD 프로그램 내에서 수행시간의 대부분을 소비하는 부분을 조사하고 이 부분만 추출하여 3.1절에서 뽑은 작업집합 후보에서 시험수행을 거침으로 전체 수행시간을 예측하는 방법을 제시한다.

[정의 2] (**핵심루프**) CFD 프로그램 내에는 많은 시간을 소비하는 한 개의 루프가 존재한다. 이 루프를 **핵심루프(kernel loop)**라 한다. 핵심루프의 조건은 다음과 같다.

(1) 핵심루프는 프로그램 전체 수행 중 수백 번 또는 수천 번 반복된다.
(2) 핵심루프를 한 번 수행하는 데에는 수십 초의 시간이 걸린다.
(3) 핵심루프내의 수행시간은 몇 번째 반복인지에 관계없이 거의 일정하다. 이 특징은 반복 첨자의 값에 상관없이 데이터만 바꾼 채 같은 계산 과정을 거치기 때문이다.

이와 같은 특징을 고려한다면, 핵심루프를 한 번 수행했을 때 속도는 핵심루프의 전체 수행 속도를 예측하는 지표가 될 수 있다는 것을 알 수 있다. 즉, 반복횟수가 1,000번이고 한번 루프를 수행하는데 필요한 시간이 30초라면 핵심루프를 수행하는데 필요한 시간은 약 30,000초이다. META의 CFD 프로그램 분석기는 CFD 프로그램을 입력으로 받아서 핵심루프의 수행 횟수를 1회로 변형한 테스트 프로그램을 생성한다. 테스트 프로그램은 또한 핵심루프의 1회 수행 시간을 측정하기 위한 코드를 삽입한다. (그림 2)에 제시된 CFD 프로그램을 입력으로 받아서 생성한 테스트 프로그램 형태는 (그림 3)과 같다. META는 테스트 프로그램을 시험 수행 후 경과시간을 측정하고 그 결과를 META에게 알린다. 결과는 원본 소스를 각 컴퓨팅 자원에 할당할 때 자원 예약에 활용된다.

META는 (그림 3)과 같이 생성된 테스트 프로그램을 자원 선택기에 전달하여 가장 빠른 작업집합후보를 선택할 수 있도록 한다. 자원 선택기를 통해 가장 빠른 작업집합을 선

```
      program NS2D
      . . . .
C     Reading Data
      start_time = second()
C     do n = 1, Nstep_Max
      . . . .
C     enddo
      end_time = second()
      elapsed_time = end_time - start_time
      call cs_send(myparent, IREALX,
     +       elapsed_time, 1, 1, 9998, ierr)
      . . . .
```

(그림 3) 핵심루프를 변형한 형태

정되면 META는 원본 CFD 프로그램을 작업집합 내의 각 컴퓨터에 전달함으로써 CFD 프로그램의 해를 구한다.

알고리즘 2: 핵심루프를 이용한 최적의 작업집합 선택 알고리즘

입력: P: 사용자가 제출한 CFD 프로그램
DataFile: CFD 프로그램을 수행하기 위한 데이터 파일
CCmd: 프로그램을 컴파일하기 위한 옵션과 컴파일 명령문
W: 작업집합 후보들의 집합
출력: w_{opt}: 최적의 작업집합

1. *KLoop* := *ExtractKernelLoop*(P)
2. P_t := *GenerateTestProgram*(P, *KLoop*);
3. **foreach** $w \in W$ **do**
4. *TransferFiles*(P_t, *DataFile*, w);
5. *RemoteCompile*(*CCmd*, w);
6. t_w := *TestRun*(w);
7. w_{opt} := W 중에서 t_w가 최소인 것을 선택
8. **return** w_{opt}

1행의 *ExtractKernelLoop*은 인수로 주어진 CFD 프로그램 P에서 핵심루프를 찾아 반환한다. 2행의 *GenerateTestProgram*은 P에서 핵심루프 *KLoop*을 삭제하고 루프를 한번 수행하는데 걸린 시간을 측정하기 위한 수행시간측정 코드를 삽입하여 테스트 프로그램 P_t를 생성한다. 3~6행에서는 각 작업집합 후보들에 대해 테스트 프로그램을 수행하는 부분이다. W 내의 각 작업집합 후보 w에 테스트 프로그램 P_t와 *DataFile*을 전송한 후, *CCmd*에 따라 원격 컴파일을 수행하고, 테스트 프로그램 수행 결과로부터 수행시간 t_w를 측정한다. 끝으로, t_w를 바탕으로 최적의 작업집합 w_{opt}를 선정한다.

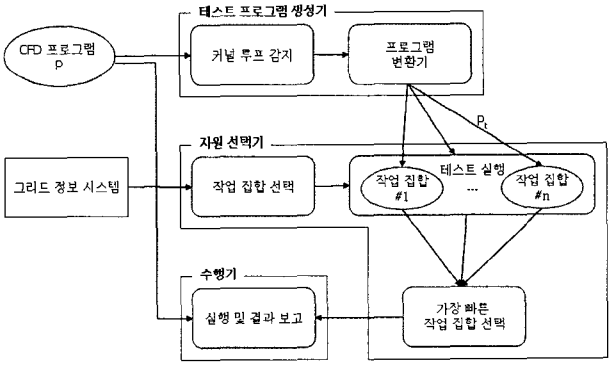
위 알고리즘에는 나타나 있지 않지만, 최적의 작업집합 w_{opt}를 선정한 후에는 여기에 원 CFD 프로그램 P를 전달하여 CFD 프로그램의 해를 구한다. 위 알고리즘은 최적의 작업집합을 선정하는 알고리즘이므로 이 과정을 포함시키지 않았다. 실제 META에는 원 CFD 프로그램을 수행하는 과정이 포함되어 있다. META의 장점은 이상에서 기술한 복잡한 과정을 매우 편리한 인터페이스 뒤에 숨김으로써 CFD 프로그래머로 하여금 마치 독자적인 컴퓨터 시스템을 사용하는 것과 같은 느낌을 줄 수 있다는 것이다.

4. 구현 및 수행결과

4.1 구현

현재 구현되어 있는 META의 구조는 (그림 4)와 같다. META는 세 개의 요소로 구성되어 있는데, 이들 요소는 테스트 프로그램 생성기, 자원 선택기, 수행기이다. 테스트 프로그램 생성기는 원 CFD 계산 프로그램 P로부터 테스트 프로그램 P_t를 생성한다. 자원 선택기는 그리드 정보 시스템으로부터 현재 가용자원들의 정보를 받는다[11, 12]. 받은 정보를 기반으로 알고리즘 1을 통해 작업집합을 선택한다. 선택된 작업집합에서 테스트 실행을 수행한 후, 테스트 실행 성능을 바탕으로 빠른 슈퍼컴퓨터들을 선택한다. 최종적으로 수행기는, 선정된 슈퍼컴퓨터 자원을 이용하여 CFD 계산 프로그램 P를 수행한다.

META는 PVM(Parallel Virtual Machine)과 PVM-make를 이용하여 구현되었다. PVM은, 병렬 혹은 직렬 컴퓨터로 구성된 이종 네트워크(heterogeneous network)를 하나의 계산 자원으로 간주할 수 있도록 해 주는 소프트웨어 패키지이다. PVM은, 프로세스 생성, 프로세스 간 통신, 이기종 네트워크 사이의 프로세스 동기화 등의 기능을 제공한다. CFD 프로그램과 데이터 파일을 각 기계로 전송하기 위해, 또 소스 프로그램을 컴파일하기 위해 PVM-make를 사용하였다.

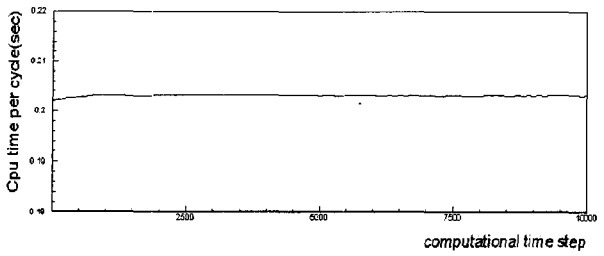


(그림 4) META 시스템 구조

4.2 핵심루프 실험

META의 효과를 살펴보기 위해, Navier-Stokes 방정식을 해결하기 위한 몇몇 CFD 모델에 대해 실험을 수행하였다. 이들 모델은 메시지 전달 및 영역 분할에 기초한 병렬 모델이다(MPI, PVM). 모든 모델은 구조적 그리드 상에서 유한 차분법(finite difference method)을 이용하여 구성한다. 따라서 수치해석 모델들의 기본 틀은 유사하다. 앞 절에서 기술한, 자원 선택 모델을 수립할 때 가정했던 것들도 모두 유효하다. 그러나 이들 Navier-Stokes 솔버는 다른 종류의 문제들을 해결하기 위해 설계된 것이라는 점에 주의하자. 프로그램 구조와 사용된 알고리즘은 모두 다르다.

CFD 시뮬레이션이 재 생산적인 특징을 갖는다는 점, 즉 1회 반복에 드는 시간은 다른 조건에 상관없이 거의 일정하



(그림 5) 단위 사이클 계산에 소요되는 CPU 시간

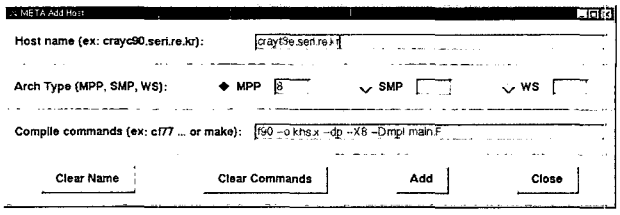
다는 점을 기억하자. 이러한 기본 가정이 옳다는 것을 알아보기 위해 1회 반복에 걸리는 시간을 (그림 5)에 도시하였다. HPC320과 GS320에서 테스트 실행을 수행한 결과 위와 같은 가정이 사실이라는 것을 알 수 있었다.

4.3 사용자 인터페이스 및 수행결과

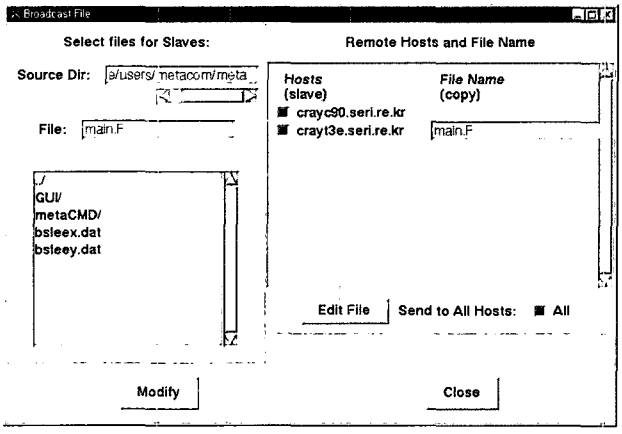
META는 수행할 슈퍼컴퓨터를 선정하고 CFD 프로그램을 전송하여 자동으로 자원을 할당하는 작업을 편리한 인터페이스를 통해 제공한다. (그림 6)은 META가 사용 가능한 호스트를 추가하기 위한 화면을 보여준다. 이 화면에서 사용자는 호스트 IP와 슈퍼컴퓨터 아키텍처 타입, 컴파일 명령어 등을 입력할 수 있다.

(그림 7)은 CFD 소스 프로그램을 선택하는 화면을 보여준다. META는 선택된 호스트에 전송될 소스파일을 사용자 로컬 디스크에서 선택할 수 있도록 하고 있다. META는 필요에 따라서 전송될 파일 이름을 변경할 수 있도록 허용한다.

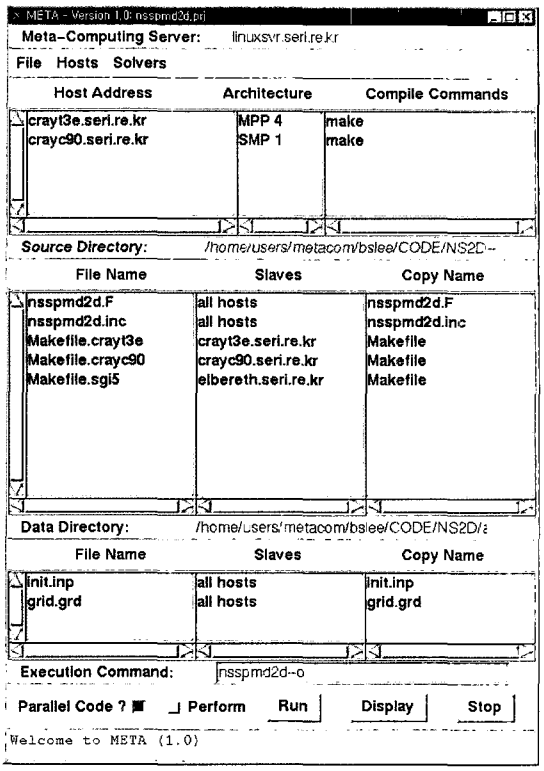
(그림 8)은 META의 메인 화면을 보여준다. 메인 화면의 맨 상단에는 현재 추가되어 있는 슈퍼컴퓨터가 나타나 있다. 중간에는 각 컴퓨터에 전송될 소스 프로그램과 메이크파일



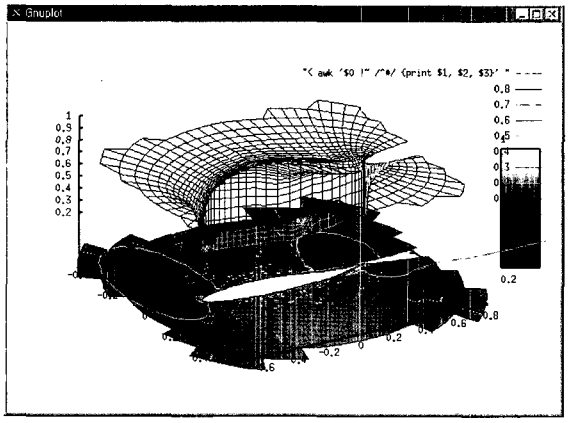
(그림 6) 호스트 추가 사용자 인터페이스



(그림 7) CFD 프로그램 선택 화면



(그림 8) META의 메인 화면



(그림 9) Gnuplot을 이용한 실시간 가시화 결과

등이 나타나있다. 하단에는 각 컴퓨터에 전송될 데이터 파일이 나타나 있다.

이상과 같은 과정을 거쳐 META 설정을 마친 후에는 CFD 프로그램의 해를 구할 수 있다. 네트워크 속도를 고려하여 작업집합 후보를 선정하고, 이들 후보에 대해서 테스트 수행을 통해 최적의 작업집합을 선정하는 과정은 사용자에게 드러나지 않는다. 따라서 사용자는 마치 하나의 컴퓨터 시스템을 사용하는 것처럼 그리드 시스템 자원을 활용할 수 있다.

(그림 9)는 실제 CFD 프로그램의 해를 구한 결과를 보여준다. META는 Gnuplot을 이용하여 CFD 프로그램의 실행결과를 실시간으로 확인할 수 있도록 하는, 실시간 가시화 기능을 제공하고 있다.

5. 다른 그리드 컴퓨팅 시스템과의 비교

기존의 그리드 컴퓨팅 시스템은 각각 고유의 문제를 해결하기 위하여 개발되었다. 대표적인 그리드 컴퓨팅 시스템인 MOL, Legion, Globus를 살펴보면, 첫째로 MOL과 Globus는 기존의 MPI를 이용하여 프로그래밍을 하는 사용자들에게 도움을 주기 위해 개발된 것이다[3,7,8]. 반면에 Legion은 분산 환경 프로그램을 위해 개발된 것이다[9]. 이 때문에 응용 프로그램 개발자들이 Legion을 사용하기 위해서는 새로운 프로그래밍 기법과 API들을 배워야 한다. 두 번째로 Globus는 분산 병렬 프로그래밍을 위한 통합 환경이라기보다 도구들을 제공하는 형태이다. 그렇기 때문에 편집, 원격컴파일, 수행, 가시화를 위한 통합 환경을 제공하지 않는다. 반면에 MOL은 가시화를 제외한 기능들을 통합하고 있다. 이들 관련연구 중에 MOL이 본 연구와 가장 유사하다. 다음 <표 1>은 두 가지 시스템을 비교한 결과를 보여준다.

<표 1>에서 볼 수 있는 바와 같이 두 시스템은 기능상에 있어서 차이점을 보이고 있다. 이와 같은 차이는 두 시스템이 목표로 하는 문제가 다르기 때문이다. MOL은 아주 넓은 지역에 흩어져 있는 많은 컴퓨팅 자원들을 연결하여 문제를 해결하는 것이 목적이기 때문에 대기시간을 줄임으로써 처리율을 올릴 수 있도록 하고 있다. 반면에 본 논문에서 제안한 META는 지역적으로 근거리에 위치한 소규모 컴퓨팅 자원을 기반으로 하고 있으므로 전송지연시간을 줄임으로써 전체 처리율을 올릴 수 있도록 한다. 또한 META는 CFD 프로그램 개발자를 대상으로 하고 있기 때문에 실시간 가시화 기능을 제공하고 있으며, CFD 분야의 특성상 컴퓨팅 자원 간 이동은 불필요하다.

〈표 1〉 MOL과 META의 비교

비교항목	MOL	META
사용자 인터페이스	제공	제공
메시지 전달	서로 다른 메시지 전달 기법 이용 가능	MPI, PVM 지원
처리율	전체관리로 대기시간을 줄임으로써 처리율 올림	전송지연시간 활용하여 컴퓨팅 자원을 할당함으로써 전체 처리율 올림
원격컴파일 및 자료 전송	기능 제공	기능 제공
컴퓨팅 자원 선정	자동 선정	자동 선정
컴퓨팅 자원 간에 작업 이동	수행시간 예측을 기반으로 이동 가능	이동 불가능
실시간 가시화	지원 없음	Gnuplot을 이용하여 실시간 가시화

6. 결 론

슈퍼컴퓨터 수가 늘어나고 네트워크 속도가 향상됨에 따라 슈퍼컴퓨터 상의 그리드 시스템의 필요성은 더욱 증대되고 있다[2-5]. 그리드 시스템의 목적에 따라 그리드 시스템과 관련된 연구 분야는 매우 다양하다. META의 목표는 CFD 사용자가 쉽게 슈퍼컴퓨터를 사용할 수 있도록 하는 것이다. 따라서 본 논문에서는 CFD 프로그램 구조의 특징을 파악하고 가장 적합한 작업 집합을 자동으로 선정하는 것으로 연구 방향을 설정하였다. META에서는 네트워크 속도와 테스트 실행의 속도를 기반으로 하여 최적의 작업 집합을 선정한다.

본 논문에서 우리는 CFD 프로그램을 모델링하기 위한 일반적인 기법으로서 작업 집합 체계와 핵심루프 모델을 제시하였다. 작업 집합 체계는 그리드 시스템에서 몇 개의 완전 서브그래프를 선정하기 위한 것인데, 각 완전 서브그래프는 작업 집합 후보라고 명명하였다. 작업 집합 후보는 네트워크 속도를 기준으로 선정된다. 핵심루프 모델은 CFD 프로그램의 구조가 같은 흐름의 일을 반복하는 형태라는 것을 이용한다. 핵심루프 모델을 통해 작업 집합 후보 중에서 최적의 작업 집합을 선정한다.

향후 연구로는 두 가지 방향을 생각해 볼 수 있다. 첫째, CFD 외의 응용분야에 사용할 수 있도록 META를 확장하는 것이다. META는 현재 CFD 프로그램을 효율적으로 수행하는 것을 목표로 하고 있다. 만약 다른 타입의 병렬 응용 프로그램에 대해서도 META의 기능을 활용할 수 있다면 활용성이 더욱 증대될 것이다. 둘째, META의 작업집합 후보를 선정하는 방법을 개선하는 것이다. 현재는 개념적인 네트워크 연결을 가정하여 네트워크 전송 지연시간을 산출하고 있다. 그러나 실제로 네트워크 연결은 완전그래프가 아니므로 작업집합 후보를 선정할 때 잘못된 정보에 근거하여 선정할 수도 있다. 이를 개선하는 것도 향후 연구로 생각해 볼 수 있겠다.

참 고 문 헌

[1] K.-W. Kang and G. Woo, "A Resource Selection Scheme for Grid Computing System META," *Lecture Notes in Computer Science*, Vol.3251, pp.919-922, 2004.

[2] V. S. Sunderam, "PVM: A Framework for Parallel Distributed Computing," *Concurrency: Practice and Experience*, Vol.2, pp.315-340, 1990.

[3] I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit", *International Journal of Supercomputer Applications*, Vol.11, pp.115-128, 1997.

[4] I. Foster and C. Kesselman, *The Grid: Blueprint for a new Computing Infrastructure*, Morgan Kaufmann Publishers, Inc. 1998.

[5] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman, "Grid Information Services for Distributed Resource Sharing," In *Proceedings of the Tenth*

IEEE International Symposium on High- Performance Distributed Computing (HPDC-10), IEEE Press, pp.181-184, 2001.

[6] K. A. Hoffmann, *Computational Fluid Dynamics for Engineers*, Morgan Kaufmann Publishers, Inc. 1993.

[7] X. Yang and M. Hayes, "Application of Grid Techniques in the CFD Field," *Integrating CFD and Experiments in Aerodynamics*, Glasgow, UK, 2003.

[8] A. Reinefeld, V. Lindenstruth, "How to Build a High-Performance Compute Cluster for the Grid," *2001 International Conference on Parallel Processing Workshops (ICPPW'01)*, pp.221-233, September, 2001.

[9] A. Reinefeld, R. Baraglia, T. Decker, J. Gehring, D. Laforenza, F. Ramme, T. Romke, J. Simon, "The MOL Project: An Open, Extensible Metacomputer," *6th Heterogeneous Computing Workshop (HCW '97)*, pp.17-34, April, 1997.

[10] Legion: Worldwide Virtual Computer, http://www.cs.virginia.edu/~legion/

[11] 김도현, 강경우, 강윤희, 조광문, "그리드 환경에서 NWS를 이용한 네트워크 정보 제공자 구현", 정보처리학회 학술발표논문집, 제9권 제2호, pp.1495-1499, 한국정보처리학회, 2002. 11.

[12] 강윤희, 강경우, 김도현, 조광문, "P2P를 기반으로 한 확장된 그리드 정보서비스 시스템 설계", 정보처리학회 학술발표논문집, 제9권 제2호, pp.205-208, 한국정보처리학회, 2002. 11.

강 경 우

e-mail : kwkang@cheonan.ac.kr
1990년 경성대학교 전산학과(학사)
1992년 한국과학기술원 전산학과(공학석사)
1998년 한국과학기술원 전산학과(공학박사)
현 재 천안대학교 정보통신학부 조교수
관심분야 : 프로그래밍언어 및 컴파일러, 트리패턴매칭, 그리드컴퓨팅 등

우 균

e-mail : woogyun@pusan.ac.kr
1991년 한국과학기술원 전산학과(학사)
1993년 한국과학기술원 전산학과(석사)
2000년 한국과학기술원 전산학과(박사)
현 재 부산대학교 정보컴퓨터공학부 조교수
관심분야 : 프로그래밍언어 및 컴파일러, 함수형 언어, 그리드컴퓨팅, 소프트웨어 메트릭 등