산학기술성공학회논문지
Vol 2, No. 1, pp. 43~51, 2001

CFD 프로그램 개발자를 위한 메타컴퓨팅 시스템

강 경 우*

The Metacomputing System for CFD Program Developer

Kyung-Woo Gang*

요 약 메타컴퓨팅 시스템은 분산된 컴퓨팅 자원들과 가시화 장비들을 통합하여 사용자가 편리하고 신속하게 작업을 처리할 수 있게 해 주는 환경이다. 본 연구에서는 전산유체역학(CFD: Computational Fluid Dynamics) 프로그램 개발자들을 위한 메타컴퓨팅 시스템을 개발하였다. 본 시스템에서 사용자는 프로그램을 컴퓨팅 자원에 분산시키고 분산환경에서 컴파일 할 수 있으며 프로그램구조에 적합한 컴퓨팅 자원에서 그 결과를 동영상을 통하여 얻을 수 있다. 본 연구에서는 다음과 같이 두 가지 연구를 수행하였다: 전산유체역학 프로그램의 구조를 이용한 자원선정에 대한 연구, 요소시스템 통합연구. 컴퓨팅 자원 선정을 위해 전산유체역학 프로그램의 구조적인 특징을 이용하였고 실시간 가시화를 위하여 기존의 가시화 도구를 수정하여 통합하였다.

Abstract Metacomputing system is the environment, which helps the users easily and promptly deal with their jobs, with integration of the distributed computing resources and visualization device. In this research, we have developed a prototype of a special-purpose metacomputing system for simulation in CFD(Computational Fluid Dynamics) field. This system supports the automatic remote compilation, transparent data distribution, the selection of appropriate computing resource, and the realtime visualization. This research can be summarized as following: a study on selecting resource and the integration of component systems. In the research of selecting computing resource, we use the property of CFD algorithm. In the research of realtime visualization, we modify a popular visualizer.

Key Words : Metacomputing, Visualization, CFD, Supercomputing

1. 서 론

네트워크의 발전과 소프트웨어 인프라스트릭처가 발전함에 따라 컴퓨팅 자원과 가시화 장비, 소프트웨어들을 같이 사용할 수 있게 해 주는 시스템인 메타컴퓨팅 시스템이 가능해졌다. 메타컴퓨팅은 네트워크와 소프트웨어로 연결된 분산된 수많은 컴퓨터들을 같이 사용할 수 있게 해 주는 시스템이다. 메타컴퓨팅 시스템의 첫 번째 이점은 사용자들이 강력한 가상 컴퓨터를 이용하는 것과 같은 효과를 낼 수 있다는 것이다[1,2,9,10]. 두 번째 이점은 다양한 대화형 작업들을 허용한다는 것이다[1,9]. 최근 들어 많은 노력이 메타컴퓨팅 시스템 개발에 기울어지고 있다. 그러나, 메타컴퓨팅 시스템을 구성하는 작업은 기술적인 문제뿐만 아니라 정책적인 문제가 결부된 복합적인 일이다. 장애가 되는 주요 기술적 문제들은 다양한 수준에서 구성하는 시스템간에 이질적인 부분이 있다는 것이다. 예를 들면, 하드웨어 구조도 다르고 운영체제도 다르며 각 컴퓨터 시스템들의 운영방식도 다르다. 최근 들어 많은 노력이 메타컴퓨팅 시스템 개발에 기울여지고 있다. 그렇지만 이상적인 메타컴퓨팅 시스템을 개발하기에 아직은 기술적으로 어렵기 때문에 특정 기능을 위한 메타컴퓨팅 시스템들이 개발되고 있다.[3] 본 연구에서는 슈퍼컴퓨터들을 사용하여 전산유체역학 프로그램을 작성하는 사용자들이 슈퍼컴퓨터들을 쉽게 사용할 수 있게 하는 시스템 개발을 목적으로 하였다.

본 연구의 실험환경에는 두 대의 이기종 슈퍼컴퓨터가 있다. 하나는 병렬 슈퍼컴퓨터이고 다른 하나는 벡터 슈퍼컴퓨터이다. 그 외에도 여러 고성능 SMP 컴퓨터들과 linux-cluster가 있다. 사용자들은 자신의 문제에 대해 다양한 컴퓨터의 구조에 맞게 최적화된 프로그램을 작성하여 사용하며 슈퍼컴퓨터의 현재 사용 가능 상태

*천안대학교 정보통신학부

본 논문은 2000년도 연구개발정보센터(KORDIC)에서 수행중인 메타컴퓨팅 시스템 개발에 관한 연구의 일부로서 그 핵심내용은 병렬 CFD 프로그램 개발을 위한 개발환경의 제안이다. (Tel: 041-550-0496)

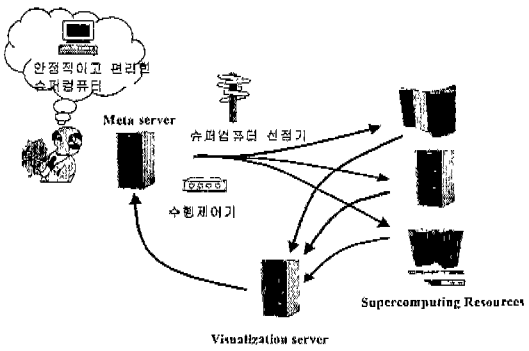

Figure 1. 본 연구의 메타컴퓨팅 시스템 개념도.

에 따라 적당한 컴퓨팅 자원을 정하고 자신의 작업을 수행한다. 그러므로, 자신의 해결한 응용분야의 문제에 대해서 다양한 프로그램들을 가지고 있는 것이다. 작업을 수행 후 사용자는 결과 파일을 워크스테이션으로 옮긴 후에 가시화 소프트웨어를 통해서 결과를 보게 된다. 사용자는 프로그램 수행 중에 자신의 프로그램의 결과를 볼 수 없고 제어할 수도 없다. 본 연구에서는 Figure 1에서 보는 바와 같이 파일 전송을 자동으로 하고 슈퍼컴퓨터 자원 선택에 도움을 주는 정보를 제공하며 실시간으로 가시화 하는 메타컴퓨팅 시스템을 개발하였다. 사용자는 자신의 프로그램과 데이터를 메타서버에 가져다 놓으면 슈퍼컴퓨터 선정기가 사용자 프로그램의 구조와 적합한 슈퍼컴퓨팅 자원을 선정하게 된다. 소스와 데이터는 자동으로 슈퍼컴퓨팅 자원으로 전송되고 컴파일되며 실행된다. 실행 중 결과는 가시화기로 전달되고 가시화기에 의해 동영상으로 보여지게 된다. 전체 흐름의 상세한 사항은 3절에서 다루어진다.

2. 관련 연구

2.1. 전산유체역학(CFD:Computational Fluid Dynamics)

전산유체역학은 유체유동현상을 지배하는 Navier-Stokes방정식을 컴퓨터를 이용하여 근사해를 구하는 방법이다.[5,12] Navier-Stokes방정식은 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.

전산유체역학에서는 연립편미분방정식인 Navier-Stokes방정식을 적절한 가정을 도입하여 컴퓨터에서 해석 가능한 대수적인 연립방정식으로 변환하여 이에 대해서 반복적인 방법으로 해를 구하게 된다. 따라서 전산유체역학 해석 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[5].

Figure 2에 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. 그러므로 전산유체역학 프로그램은 많은 중첩 루프들을 가지는 알고리즘으로 표현할 수 있다. 이와 같은 프로그램의 구조를 살펴보면 가장 바깥의 루프는 시간축을 따라 반복하는 루프로써 Nstep_Max 회 반복된다고 가정하면 전체 계산수행시간은 (한 Step의 계산시간 × Nstep_Max)정도가 된다고 할 수 있다. 그리고 각각의 N루프에서의 계산시간은 프로그램 수행 중에 거의 일정하다.

전산유체역학 해석 프로그램을 슈퍼컴퓨터에서 수행시킬 때 좋은 성능을 얻기 위해서는 inner-most 루프의 벡터화와 영역분할이 중요하다. 벡터 슈퍼컴퓨터가 제공하는 벡터 파이프라인을 충분히 활용하기 위해서는 프로그램의 inner-most 루프들이 벡터화 될 수 있어야 한다. 그리고, 벡터화 된 루프들의 루프 크기가 벡터 파이프라인을 충분히 활용할 만큼 커야 한다. 병렬 슈퍼컴퓨터에서는 영역 분할을 어떻게 하느냐에 따라 프로세서들 간 메시지 전달 양이 달라지고 프로세서가 수행할 계산량이 달라진다[5,12].

Figure 3은 structured 프로그램이고 Figure 4는 비행기의 표면장력을 시뮬레이션 하는 unstructured 프로그램이다. Structured 프로그램에서 영역 분할할 때 고려

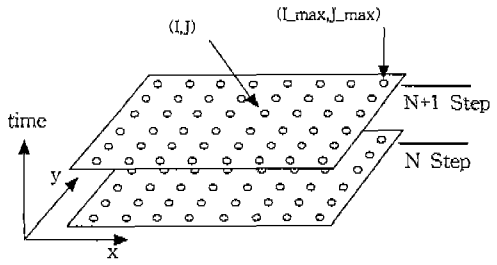

Figure 2. 전산유체역학해석 프로그램의 기본개념.

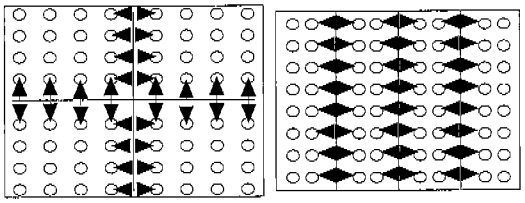

Figure 3. Structured 프로그램에서 영역분할.

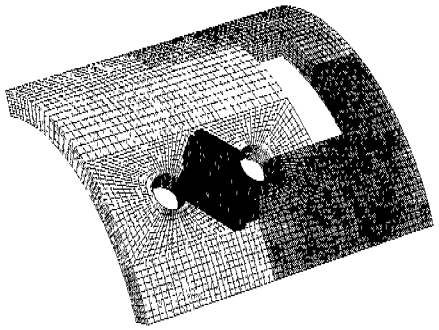

Figure 4. Unstructured 프로그램에서 영역분할.

해야 하는 것은 분할로 인해 발생하는 통신량이다. Figure 3의 경우에도 두 방법이 통신량이 다름을 알 수 있다. 반면에 unstructured 프로그램인 Figure 4에서는 전체 도메인의 모양이 같은 크기로 나누기에 부적합한 형태이다. 그리고, Figure 4와 같은 경우에는 겉으로 보기에 비슷한 크기로 나눈다고 해서 프로세서들의 계산량이 같은 것이 아니기 때문에 분할 시에 통신량 분만 아니라 계산량도 고려해야 한다.

2.2. 메타컴퓨팅 시스템 개발 관련 연구

2.2.1. MOL(Metacomputer Online)

MOL은 WAN환경 상에 연결된 고성능 시스템들을 마치 하나의 컴퓨팅 자원처럼 사용자에게 제공하는 소프트웨어이다. 여러 개의 시스템을 연결하는 것은 하나의 슈퍼컴퓨터에서 해결할 수 없는 큰 크기의 문제를 해결하는데 도움을 준다[1]. MOL 프로젝트의 목표는 기존의 소프트웨어 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[1].

- 사용자가 편리한 사용자 인터페이스
- 서로 다른 메시지 패싱을 이용하는 기존의 코드의 연동을 지원
- Throughput을 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 실행중 작업을 이동시킬 수 있음

2.2.2. Legion

버지니아 대학에서 수행중인 과제로써 객체지향 메타시스템을 개발하는 것이다. Legion 프로젝트에서는 사용자들에게 하나의 가상 컴퓨터를 제공하는 시스템을 만들고자 하고 있다[2]. 개발하는 가상 컴퓨터는 병렬처리를 이용하여 응답시간이 빠를 뿐만 아니라 높은 throughput을 제공하는 것이다. Legion은 Workstation cluster들과 workstation 등, 슈퍼컴퓨터들, 빙렬 컴퓨터들을 대상으로 하고 있다. 이들의 개발 방법은 기존의 객체지향 병렬처리 시스템에 이기종 분산처리 시스템의 특징들을 가미하는 방법을 취하고 있다[4]. 기존의 것으로 활용하는 것은 이들이 이미 개발한 Mentat라는 객체지향 병렬처리 시스템이다. 여기서 개발된 도구를 이용하여 문제를 해결하고자 한다면 사용자는 객체지향 프로그래밍 기법을 충분히 이해하여야 하고, 많은 API들의 사용을 익혀야 한다.

2.2.3. Globus

통신, 자원 할당, 데이터 접근 등 이기종 컴퓨팅 인프라의 요구사항들을 낮은 수준의 메커니즘들을 이용하여 제공한다. 이들 메커니즘은 스케줄러와 병렬프로그래밍 도구 등 상위 레벨의 이기종 컴퓨팅 서비스들을 구현하는데 사용된다. 구성요소들은 각각 통신, 자원을 찾고 할당, 자료 접근, authentication들을 위한 요소들이었다[4,10].

- 통신기능: 메시지 전달, Remote Procedure Call, DSM등 다양한 통신방법을 제공
- 자원할당: 응용프로그램이 필요한 자원들을 표현하고 응용프로그램이 필요에 맞는 자원을 찾고 스케줄링 하는 메커니즘[4]
- 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 파라메터들을 전달하고 새로운 프로세스를 실행시키고 끝내는 일을 맡는 부분
- Authentication interface: 사용자와 자원의 ID를 구분해 주는 모듈

3. CFD처리를 위한 메타컴퓨팅 시스템

3.1. 컴퓨팅 자원 선정 방법

Structured 전산유체역학 프로그램들은 전체 수행시간의 대부분을 소비하는 코드가 있다. 본 연구에서는 structured 전산유체역학 프로그램 내에서 수행시간의 대부분을 소비하는 부분을 조사하고 이를 변형함으로써 전체 수행시간을 예측한다[6-8].

[정의1] 핵심루프

전산유체역학 프로그램 내에는 많은 시간을 소비하는 한 개의 루프가 존재한다. 이 루프를 핵심루프라 한다. 핵심루프의 초기조건은 다음과 같다.

(1) 핵심루프는 수백 번 또는 수천 번 반복한다.

(2) 한번 수행하는데 수십 초의 시간을 필요로 한다.

(3) 핵심루프내의 수행시간은 몇 번째 반복인지에 관계없이 거의 일정하다. 이 특징은 반복 첨자의 값에 상관없이 데이터만 바꾼 채 같은 과정을 이용하기 때문이다.

이와 같은 특징을 고려한다면 핵심루프를 한번 수행했을 때 속도는 핵심루프의 전체 수행 속도를 예측하는 지표가 될 수 있다. 즉, 반복횟수가 1000번이고 한번 루프를 수행하는데 필요한 시간이 30초라면 핵심루프를 수행하는데 필요한 시간은 약 30000초이다. 본 연구에서는 전산유체역학 프로그램을 읽고 핵심루프의 수행횟수를 1번으로 변형한다. 그리고, 슈퍼컴퓨터 군에 있는 사용가능 컴퓨터에서 수행하여 수행시간을 측정한다. 수행 후 제일 먼저 결과를 내는 슈퍼컴퓨터가 원본 프로그램을 수행시켰을 때도 제일 빨리 결과를 낼 수 있다.

프로그램 작성자는 프로그램 내에서 핵심루프가 어디인지 표기해 주어야 한다. 프로그램 내에는 수많은 루프가 있다. 이들 중 시간을 대부분 소모하는 핵심루프는 바르게 판단 될 수 없기 때문에 프로그래머는 핵심루프가 시작하는 지점을 지정해 주어야 한다.

```
      program NS2D
        . . . .
        I = 0
        Start_time = second()
c       do 10 j = 1, jmax
          . . . .
c 10    continue
        end_time = second()
        elapsed_time = end_time  start_time
        call cs_send(myparent, IREALX,
        +       elapsed_time, 1, 1, 9998, ierr
         . . . .
```

Figure 5. 핵심루프를 변형한 형태.

Figure 5와 같이 변형 후 성능예측 시스템은 대상이 되는 슈퍼컴퓨터로 프로그램과 자료를 옮기고 수행하게 된다. 이 과정은 다음 알고리즘으로 요약할 수 있다.

Algorithm3.1: Performance-estimation with kernel-loop (source,data,compile,execution)

/*

1. source: a structured CFD program file
2. data: data file for running the program
3. compile: commands for compiling source
4. execution: execution method after compile

*/

Output: Selected Supercomputer

Phases:

(1) Module_Search /* 프로그램 파일들 중 핵심루프를 포함하는 모듈을 찾음 */

(2) Kernel_Loop_Search /* 모듈을 AST(Abstract Syntax Tree)로 변형하여 핵심루프의 시작과 끝을 찾음 */

(3) Code_Conversion /* 루프를 삭제하고 루프를 한번 수행하는데 걸린 시간을 측정하기 위한 수행시간측정 코드를 삽입 */

(4) File_Transfer /* 슈퍼컴퓨터로 프로그램과 자료를 전송 */

(5) Remote_Compile /* 슈퍼컴퓨터에서 지정된 방법에 따라 컴파일 */

(6) Spawn /* 실행 파일을 spwan */

(7) Supercomputer_Select /* 실행파일이 수행 후 리턴한 기록을 기반으로 슈퍼컴퓨터 선정*/

3.2. 처리 알고리즘

본 연구에서 개발한 메타컴퓨팅 시스템의 전체 시스템의 처리과정은 Figure 6와 같다. 메타컴퓨팅 시스템은 크게 4가지 부분으로 나눌 수 있다: 사용자 인터페이스, 사용자가 제출하는 파일들, 슈퍼컴퓨터 선정 및 수행제어, 응용 프로그래밍(또는 Solver)의 수행 등이다.

3.3. 전산유체역학 프로그램과 하드웨어 구조 사이의 친숙도 분석

3절에서 언급한 전산유체역학 프로그램의 구조를 4개의 PE(processing Element)에서 수행할 경우를 나타내면 Figure 7과 같다. 각 프로세서는 자신에게 할당된 데이터 영역 상에서 계산을 하는데 계산이 끝나면 두 번째 time-step으로 들어가기 전에 이웃하는 프로세서들과 영역분할에 의해 발생한 경계선에 있는 데이터들을

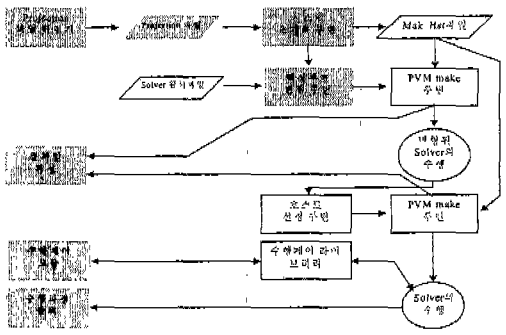

Figure 6. 메타컴퓨팅 시스템의 처리과정.

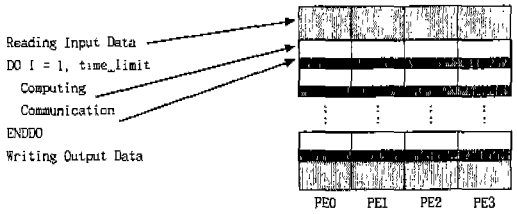

Figure 7. 4개의 PE상에서 수행할 때 CFD 프로그램의 구조.

통신을 통하여 교환한다. 그리고 난 후 모든 프로세서들은 두 번째 time-step으로 들어간다. 이렇게 프로그램을 작성할 때 최적화를 위해 고려해야 하는 것이 두 가지가 있다. 첫째는 계산시간과 통신시간 사이의 균형이다. 일반적으로 통신은 계산에 비해 느리기 때문에 프로그램 작성시 통신량을 작게 해야 한다. 두 번째로 고려해야 하는 것은 프로세서들간 계산량 사이의 균형이다. structured 프로그램인 경우에는 계산량을 균등하게 하는 것은 쉬운 일이지만 *unstructured* 프로그램일 경우에는 균등화하기가 쉽지 않다[6-8].

본 연구에서는 사용자 전산유체역학 프로그램의 핵심부분인 반복 계산 단계의 일부분을 시험수행하고 그 결과를 사용자에게 제공함으로 사용자가 부하를 예측할 수 있게 하였다. 제공하는 정보는 다음 네 가지이다[4].

(1) 벡터화 비율: 소스코드의 inner-most루프들이 벡터화 된 비율을 나타내는 정보로써 컴파일 과정에 알 수 있고 벡터 컴퓨터에서 좋은 성능을 나타낼지 에 대한 척도가 된다.

(2) 벡터 크기: 컴파일 과정에 벡터화 된 루프들이 실제 수행시간에 벡터 파이프라인을 얼마나 사용하는지를 나타내는 정보로써 시험 수행 후 알 수 있으며 벡터 컴퓨터에서 좋은 성능을 나타낼지 에 대한 척도가 된다.

(3) 계산시간에 대한 통신시간의 비율: 통신시간의 비율이 작으면 프로그램은 CPU의 계산시간을 충분히 활용할 수 있기 때문에 좋은 성능을 나타낸다.

(4) 프로세서들 사이에 계산시간 편차: 편차가 작을수록 프로세서들 사이에 작업분배가 잘 이루어 진 것임을 나타내는 것이다.

Table 1은 두 명의 사용자가 작성한 2차원 Navier-Stokes 프로그램에 대한 벡터컴퓨터에서의 정보를 보여주고 있다. NS2D(1)은 벡터화 비율은 크지만 벡터 크기가 작기 때문에 벡터 슈퍼컴퓨터의 특징인 벡터 파이프라인을 충분히 활용하지 못한다. 반면에 NS2D(2)는 벡터화 비율도 클 뿐만 아니라 벡터의 크기도 크기 때문에 벡터 슈퍼컴퓨터를 충분히 활용하며 수행하게 된다. 그러므로, NS2D(1)의 사용자는 벡터 크기를 크게 프로그램을 고치거나 다른 슈퍼컴퓨터를 사용하는 편이 낫다.

Figure 8은 NAS의 병렬 컴퓨터 벤치 마킹 프로그램 중에서 LU 프로그램(C class,162X162X162 격자수)을 CRAY T3E 4개 CPU를 이용하여 계산했을 때 첫 번째 계산 단계에서 핵심루프의 CPU 경과 시간과 통신시간을 프로세서마다 나타내었다. 최적화가 잘 된 병렬 프로그램이기 때문에 프로세서간 편차도 거이 없고 통신시간이 차지하는 비율도 아주 작다.

Figure 9은 사용자가 작성한 structured 프로그램을 8개 CPU상에서 수행 시켰을 경우이다. 2절에서 언급한 바와 같이 structured프로그램은 프로세서간 부하를 균등하게 하기 위해 도메인을 분할하는 것은 쉬운 일이다. Figure에서 보는 바와 같이 프로세서간 계산시간의 편차는 작다. 그렇지만 통신시간이 차지하는 비율은 크다. 사용자는 통신 시간을 줄이도록 프로그램을 수정할 필요가 있다.

Table 1. 벡터 슈퍼컴퓨터와 친숙도 정보

벡터 정보 / 프로그램	벡터화 비율	벡터 크기
NS2D(1)	83.8 %	29
NS2D(2)	90.9 %	117

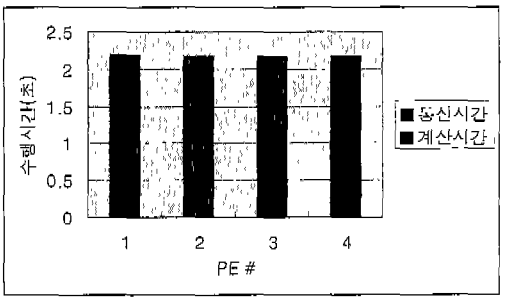

Figure 8. LU Solver의 시험수행.

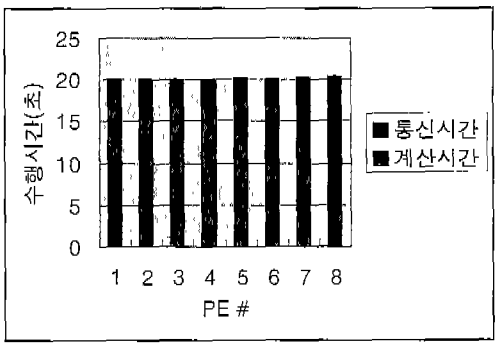

Figure 9. Structured 프로그램의 시험 수행.

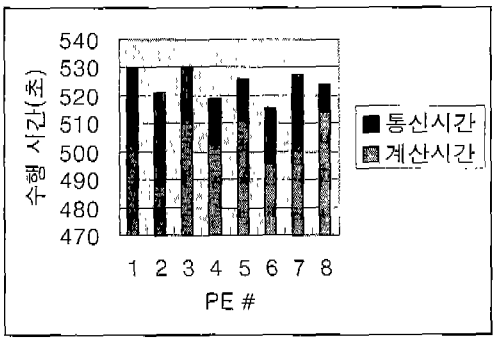

Figure 10. Unstructured 프로그램의 시험 수행.

Figure 10은 사용자가 작성한 unstructured 프로그램을 8개 CPU상에서 수행 시켰을 경우이다. 이런 종류의 프로그램은 프로세서간 부하 균등화도 어렵고 통신 시간의 비율을 줄이는 것도 어렵다. 사용자는 다양한 도메인 분할을 통해 부하 균등화를 시도해야 할 것이다.

고성능 슈퍼컴퓨터 상에서 효율적인 프로그램을 작성하는 것은 쉽지 않다. 일반적으로 작성한 프로그램을 여러 번 수행시키고 그 결과를 바탕으로 다시 최적화하는 방법을 이용한다. 벡터 슈퍼컴퓨터에서는 inner-most loop들이 벡터화 되도록 할 것이고 병렬 슈퍼컴퓨터에서는 프로세서간 계산시간 편차를 작게 하고 통신시간의 비율을 작게 할 것이다.

본 연구에서는 병렬 슈퍼컴퓨터인 CRAY T3E와 벡터 슈퍼컴퓨터인 CRAY C90상에서 전산유체역학 프로그램의 개발을 돕는 도구를 개발하였다. 제안된 방법은 전산유체역학 해석 프로그램의 반복적인 부분을 추출하여 시험수행을 거친다. 시험수행 후 얻어진 벡터화 비율, 벡터 크기, 계산시간 편차, 통신시간 비율 등을 사용자에게 제공함으로 사용자가 프로그램을 고치거나 슈퍼컴퓨터를 선택하도록 돕는다. 본 연구에서 개발된 도구를 이용한다면 프로그램 개발자는 최적화된 프로그램을 더 빨리 개발할 수 있다.

4. 구현 및 실험

4.1. 구 현

메타컴퓨팅 시스템의 구현환경은 Figure 11에 나와 있다. 슈퍼컴퓨터들과 병렬형 컴퓨터들을 컴퓨팅 자원으로 이용하여 구현하였다. 구현된 메타컴퓨팅 시스템은 리눅스가 운영체제인 컴퓨터에서 C언어로 구현되었다. 사용자 인터페이스는 Tcl/Tk 8.3을 이용하여 구현하였다. 메시지 전달과 통신을 위해서 본 연구에서는 PVM을 이용하였다.[12] PVM은 이기종 컴퓨터간 통신과 프로세서들의 생성과 관리를 할 수 있게 해 주는 기능들을 제공한다. 자동 파일 전송과 원격 컴파일을 위해서 pvmmake를 이용하였다. 그리고, Solver의 수행 결과를 가시화 하기 위해 Gnuplot을 이용하였다. Gnuplot은 명령어 위주의 interactive한 함수 plotting 프로그램이다. Gnuplot은 데이터를 입력으로 받아서 정지된 이미지를 생성하는 기능만을 가지고 있는데 본 연구에서는 Gnuplot의 소스를 변경함으로 여러 데이터를 이용한 동적 이미지도 가능하게 하였다. Figure 12, Figure 13, Figure 14는 비행기의 Airfoil주위에 발생하는 압력을 contour를 이용하여 나타낸 것으로써 동영상 중에 한

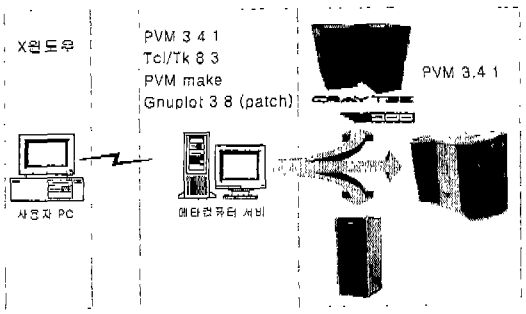

Figure 11. 시스템 구현 환경.

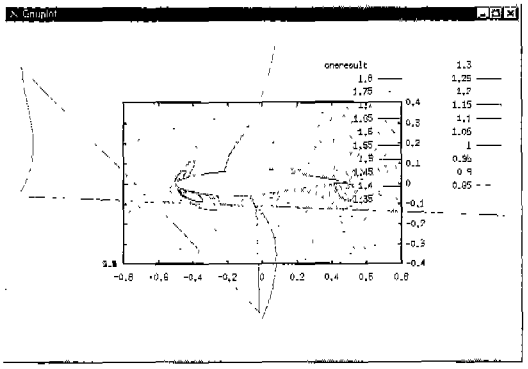

Figure 12. bug 수정 전 contouring 모습.

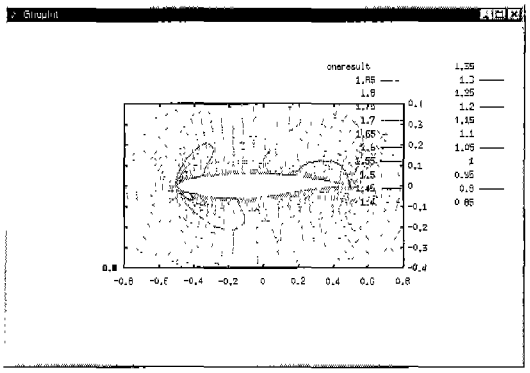

Figure 13. bug 수정 후 contouring 모습.

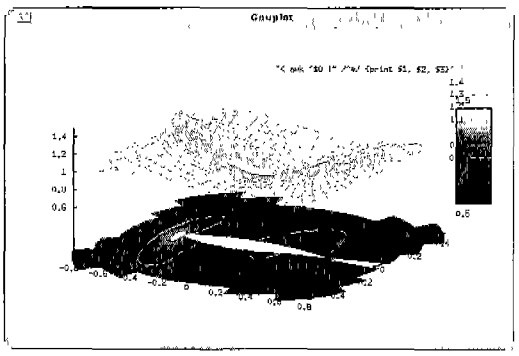

Figure 14. 3차원 simulation 가시화.

순간의 영상을 잡은 것이다. 사용자는 자신의 Solver에 그리기를 원하는 GRID데이터를 출력하면 된다. 동영상을 위해서는 각 프레임 사이에 "@"를 넣어 줌으로 새로운 프레임 출력임을 알게 된다. Gnuplot은 사용자가 출력하는 데이터를 하드디스크에 저장하지 않고 파이프를 통하여 받아들인다. Gnuplot의 소스에서 고친 부분들 중 또 다른 부분은 contour를 그리는 부분이다. Gnuplot이 가지고 있던 bug 중 하나로 그리는 영역을 재조정했을 때 잘못된 형태의 contour를 그린다는 것이다. 본 연구에서는 Figure 13과 Figure 14에서 보는 바와 같이 bug를 수정을 통하여 바로 잡았다.

4.2. 실 험

본 연구에서 개발한 메타컴퓨팅 시스템의 효용성을 위해서 Navier-Stokes 방정식의 해를 위해 작성된 전산유체역학 프로그램을 가지고 실험을 수행하였다. 첫 번째 모델은 NS2D라고 불리는 순차 프로그램이다. 다른 두 개의 모델은 영역분할과 메시지 전달 라이브러리를 이용한 병렬 프로그램인데 각각 MPI와 PVM을 가지고 작성되었다. 각각은 NS2D-PVM, NS2D-MPI 라고 불린다.[5] 모든 모델들은 structured 그리드 시스템 상에서 finite difference (volume) method을 이용하여 수식화 되었다. 결론적으로, 수치모델들의 기본 구조는 서로 비슷하다. 앞 절에서 언급한 자원 선정 방법을 위한 가정을 적용할 수 있다. 그리고, 위의 세 가지 프로그램은 Navier-Stokes 방정식을 기반으로 했지만 적용되는 문제가 서로 다르기 때문에 성능을 서로 비교할 수는 없다. 다음의 Table 2는 슈퍼컴퓨터들 상에서 이들 Solver들을 수행했을 때 성능을 나타내고 있다.

NS2D는 원래 벡터프로세싱을 고려하지 않고 캐쉬를 기반으로 한 RISC 프로세서 상에서 효율적으로 수행될 수 있도록 작성된 프로그램이다. 반면에, NS2D-MPI는 벡터 프로세싱과 병렬 처리, 둘 다를 고려하여 설계되었다. 그렇기 때문에 Cray c90에서 수행했을 때 훨씬 빠른 수행을 한 것이다. 그렇지만, Cray T3E에서도 12개 이상의 프로세서를 사용할 수 있다면 Cray C90에서 보다 빨리 결과를 볼 수 있다. 그렇기 때문에 Solver를 수행할 현재 사용 가능한 프로세서들의 수를 알고 있어야 한다. 그런 정보를 알고 있고 이를 기반으로 더 나은 컴퓨팅 자원을 선택하는 것이다.

전산유체역학 시뮬레이션 프로그램들은 일단 한번 개발되면 자주 사용되고 반복에 무관하게 계산 시간은 일정하다. 이 사실을 입증하기 위해 각 시간 단위마다 측정된 계산시간을 Figure 15와 Figure 16에서 보여주고

Table 2. 슈퍼컴퓨터 상에서 Solver들의 CPU-time (sec)

Supercomputer / Solver	Cray C90	Cray T3E	
	PE	1 PE	12 PEs
NS2D	658.62	644	
NS2D-MPI	369.96	4400	360

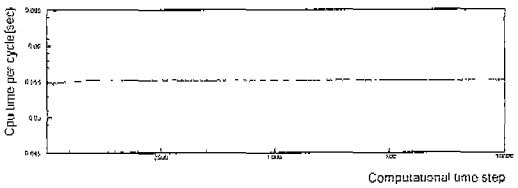

Figure 15. NS2D[lee]-PVM의 각 시간단계에 따른 CPU time.

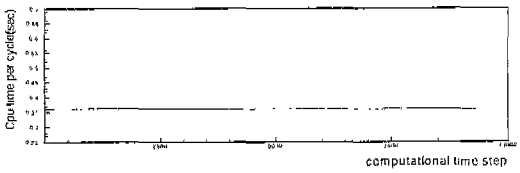

Figure 16. NS2D의 각 시간단계에 따른 CPU time.

있다. 이들 Figure에서 보는 바와 같이 매 시간 단위마다 거이 일정함을 알 수 있다. 그럼에도 불구하고 약간에 오차가 발생하는 것은 프로세서들 간에 통신이 시간대에 따라 차이가 있기 때문이다. 여기서 제안된 방법은 슈퍼컴퓨터의 전체 계산시간을 효율적으로 스케줄 할 때 사용할 수 있지만 또한 사용자의 비용 절약 측면에서도 활용할 수 있다. 일반적으로 슈퍼컴퓨터의 사용료는 자신의 작업에 CPU가 할당된 elapsed CPU time을 기반으로 한다. 전산유체역학 코드일 경우에는 몇 번의 수행으로 전체 수행시간을 예측할 수 있기 때문에 자신의 작업을 수행시킬 때 소요되는 비용을 알 수 있다. 이 정보를 이용하여 사용자는 적당한 슈퍼컴퓨터를 선정하여 사용할 수 있다. 단, 이 방법들의 단점은 항상 시험수행이 있기 때문에 시험수행에 컴퓨팅 시간을 낭비한다는 것이다.

4.3. 관련 연구와의 비교

기존의 메타컴퓨팅 시스템은 각각 고유의 문제를 해결하기 위하여 개발되었다. 2절에서 예를 들었던 대표적인 메타컴퓨팅 시스템인 MOL, Legion, Globus를 살펴보면, 첫째로 MOL과 Globus는 기존의 MPI를 이용하여 프로그래밍을 하는 사용자들에게 도움을 주기 위해 개발된 것이다. 반면에 Legion은 분산환경 프로그램을 위해 개발된 것이다. 이 때문에 응용 프로그램 개발자들이 Legion을 사용하기 위해서는 새로운 프로그래밍 기법과 API들을 배워야 한다. 두 번째로 Globus는 분산 병렬 프로그래밍을 위한 통합환경 이라기 보다 도구들을 제공하는 형태이다. 그렇기 때문에 서론에서 언급했던 편집, 원격컴파일, 수행, 가시화를 통합환경에서 할 수 있는 것은 아니다. 반면에 MOL은 가시화를 제외한 기능들을 통합하고 있다. 이들 관련연구 중에 MOL이 본 연구와 가장 유사하다[1]. 다음 Table 3은 두 가지 시스템을 비교한 결과를 보여준다.

이 표에서 보는 바와 같이 기능상에 차이가 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 throughput을 올리고 실시간 가시화 기능은 필요로 하지 않은 것이다. 반면에 본 연구에서는 지역적으로 근거리이고 컴퓨팅 자원도 주어진 몇 개 시스템을 기반으로 하고 사용자도 전산유체역학 개발자이기 때문에 실시간 가시화 기능이 필요하고 컴퓨팅 자원간 이동은 불필요한 것이다.

5. 결 론

5.1. 연구개발 결과

본 연구에서는 복수의 슈퍼컴퓨터를 운영하는 이기종 전산환경의 이용 효율을 최적화 하고 고성능 네트워크를 이용한 메타컴퓨팅 시스템을 개발하는 것이 목표이다. 본 연구에서 개발된 시스템은 전산자원인 슈퍼컴퓨터들의 하드웨어 특성과 현재 부하에 따라 작업할당을 결정하고 사용자의 편이를 위해 사용자 인터페이스와 실시간 가시화기능을 제공한다.

본 연구에서 메타컴퓨팅 시스템을 개발하며 여러 요소기술을 축적하였다. 이들 기술은 메타컴퓨팅 시스템 연구 분야뿐만 아니라 전산유체역학 연구분야, 가상 실험환경 분야 등에 적용할 수 있다. 이들은 다음과 같다.

(1) 성능 예측기
- 핵심 루프를 이용한 수행 시간 예측 기술
- 프로그램 변형 기술
- 알고리즘과 하드웨어 구조 사이에 친숙도 분석 기술

(2) 수행 제어기
- 프로그램 수행 중 제어 기술
- 구성요소인 subsystem들 간 통신을 안정화하는 통신 시스템 기술

Table 3. MOL과 본 연구의 비교

비 교 항 목	MOL	본 연구
사용자 인터페이스	제공	제공
메시지 패싱	서로 다른 메시지 패싱 이용 가능	MPI, PVM 지원
Throughput	전체관리로 대기시간 줄임으로 throughput 올림	응용 프로그램의 구조와 친숙한 컴퓨팅 자원을 이용함으로 전체 throughput 올림
원격컴파일 및 자료 전송	기능 제공	기능 제공
컴퓨팅 자원 선정	자동 선정	사용자 도움을 기반으로 하는 자동 선정
컴퓨팅 자원간에 작업 이동	수행시간 예측을 기반으로 이동 가능	이동 불가능
실시간 가시화	지원 없음	Gnuplot을 이용하여 실시간 가시화

(3) 사용자 인터페이스

사용자가 슈퍼컴퓨터를 쉽게 사용할 수 있는 인터페이스 설계 기술

- 프로그램의 수행과정을 실시간으로 가시화 하는 가시화 기술

5.2. 향후 연구 계획

구현 방향은 일반적인 응용분야가 아닌 전산유체역학 분야의 사용자들을 위한 편리한 계산환경을 만드는 것이다. 본 시스템은 사용자 인터페이스를 제공하고 응용프로그램에 따라 슈퍼컴퓨팅 자원을 자동으로 선정하고 계산 결과를 제어하기 위한 도구와 실시간으로 보여주는 가시화기를 제공한다. 본 연구에서 개발된 시스템은 전산유체역학 프로그램 작성자들에게 도움을 주는 개발환경으로 활용할 수 있다. 또한, 가상 풍동(Virtual Wind Tunnel)에서 사용할 수 있다. 가상 풍동은 사용자가 시뮬레이션 코드를 지정해 주고 원하는 형상을 그리면 자동으로 시뮬레이션을 해서 그 결과를 형상 주위에 그리는 실험용 소프트웨어이다. 본 연구에서는 시뮬레이션 코드의 자동 수행과 결과를 보여주는 가시화 도구까지 개발했기 때문에 형상을 그리는 도구를 연동시키기만 하면 가상 풍동이 완성될 것이다.

본 연구에서 개발된 요소들 중에 사용자 인터페이스와 원격컴파일은 클러스터를 쉽게 사용할 수 있게 해주는 도구로써 사용할 수 있다. 클러스터를 사용하는 사람들은 클러스터 전체를 하나의 컴퓨터처럼 보게 하는 SSI(Single System Image)의 부족으로 불편을 겪는다[11]. 이런 불편을 해소하기 위해 개발된 요소 시스템들은 적용될 수 있다.

참 고 문 헌

[1] A. Reinefeld et al, The MOL Project: An Open, Extensible Metacomputer, In proc. 6th Heterogeneous Computing Workshop (HCW 97), Geneva, pp. 17-31, 1998.

[2] A. S. Grimshaw, J. B. Weissman, E. A. West, and E. C. Loyot, Jr, "Metasystems: An Approach Combining Parallel Processing and Heterogeneous Distributed Computing Systems", Journal of Parallel and Distributed Computing Vol. 21, 257-270, 1994.

[3] Jinhwen Wang, Songnian Zhou, Khalid Ahmed, and Weihong Long, "LSBATCH: A Distributed Load Sharing Batch System", CSTR-286, CSRI, University of Toronto, April, 1993.

[4] K. Czajkowski, I. Foster, C. Kesselman, S. Martin, W. Smith and S. Tuecke, "A Resource Management Architecture for Metacomputing Systems", Preprint, mathematics and Computer Science Division, ANL, Argonne, Ill., 1997.

[5] Klaus A. Hoffmann, "Computational Fluid Dynamics for Engineers," 1993.

[6] N. B. MacDonald, "Predicting Execution Times of Sequential Scientific Kernels", Proceedings International Workshop on Automatic Distributed Memory Parallelisation, Automatic Data Distribution and Automatic Parallel Performance Prediction, Saarbrucken, Germany, pp: 56-92, March 1-3, 1993.

[7] R. Wolski, N. Spring and C. Peterson, "Implementing a Performance Forecasting System for Metacomputing: The Network Weather Service", SuperComputing '97, UCSD Technical Report TR-CS97-540, pp: 103-130, May 20, 1997.

[8] Thomas Fahringer Automatic Performance Prediction of Parallel Programs, Kluwer Academic Publishers, 1996.

[9] L. Smarr, C. E. Catlett. Metacomputing. Communications of ACM 35,6(1992), p: 45-52

[10] The Globus project. http://www.globus.org/

[11] M. A Baker, G.C. Fox, H. W. Yau. Cluster computing review. Techn. Report, Syracuse Univ., Nov. 1995.

[12] Al Geist et el., "PVM : Parallel Virtual Machine, A Users' Guide and Tutorial for Networked Parallel Computing," The MIT Press, 1994.