TeraCluster에서 CFD 코드의 병렬 성능 분석

Performance Analysis of a CFD code in the TeraCluster Parallel System

(○) 조 금 원[1], 이 상 산[2]
K.W. Cho, S. Lee

At the end of 1999, the TeraCluster project has started in the KORDIC Supercomputer center to study PC clusters for parallel computing. The aim is to replace the Cray T3E with a new cluster system in 2002. The PC cluster without a fast network is well suited for applications which do not require large amount of global communications. Since CFD problems are not very communication intensive, whole test cluster may be efficiently used. As an example of practical CFD simulations, the flow past the ONERA M6 wing and the flow past infinite wing are simulated on a cluster of Linux workstations.

1. 서 론

연구개발정보센터(KORDIC) 슈퍼컴퓨팅 사업단은 현재 TeraCluster로 명명된 프로젝트를 수행 중에 있다. 본 프로젝트의 목적은 2002년도에 KORDIC에서 운영중인 Cray T3E(450MHz EV5 알파프로세서)를 대체하기 위한 PC cluster를 개발하는 것이다[1]. 이의 1단계 과정으로 DS10(466MHz EV6 알파 프로세서) 65개(1개의 노드는 Front-End로 사용)와 통신을 위한 스택커블 스위치(stackable switch), Fast Ethernet 카드 그리고 Linux OS(kernel 2.2.0)등으로 이루어진 PC cluster를 구성하였다. 구성된 시스템과 Cray T3E의 latency는 각각 140 μsec와 13 μsec 정도이며, bandwidth는 각각 8.9 *Mbyte*/sec와 157 *Mbyte*/sec정도이다. 각 측정값은 MPICH[2]를 이용하여 소프트웨어적으로 측정한 값이다. 구성된 시스템에 대한 성능분석은 크게 두 분야로 나뉘어 수행되어졌다. 첫 번째는 시스템의 기본 특성을 파악하기 위해 전산분야 시험이 이루어졌다. 이것은 시스템간의 통신 및 MPI 라이브러리등의 기본 성능을 분석하기 위한 것으로 NPB(NAS Parallel Benchmarks)[3]와 PMB(Pallas MPI Benchmarks)[2] 시험이 수행되어 Cray T3E에서의 결과와 비교되었다[1]. 두 번째로 주요 슈퍼컴퓨터 사용자 그룹 분야[구조, 기상, 물리, 열유체, 화학]에서 실제 사용되는 코드를 이용하여 성능을 분석하였다. 이중에서 본 연구는 열유체 분야 코드를 구성된 DS10 시스템에 적용하여 특성을 파악하였으며 마찬가지로 Cray T3E에서의 결과와 비교하였다. 사용된 수치해석 방법은 압축성 유한체적방법[4]이며, 시간적분으로 AF-ADI[4], 공간차분방법으로 Roe의 FDS[5]와 3차 정확도를 갖는 MUSCL 기법[6]이 사용되었다.

본 연구에서는 일반적으로 저렴한 가격과 구입이 용이한 부품으로 시스템을 구성하도록 노력하였으며, 구성된 시스템에 대해 열유체 분야 코드가 어떠한 성능을 갖는지를 파악하고자 하였다. 성능분석은 프로그램의 입출력과정, 해석과정, 자료교환과정 그리고 후처리 과정 등으로 세분화하여 각 부분의 특성을 파악하고자 하였으며, 각 부분에 대해 가능한 좋은 성능을 얻을 수 있는 방법을 제시하고자 하였다. 본 논문은 제 2장에서는 병렬화 기법, 3장에서는 병렬 성능, 4, 5장에서는 결론 및 참고문헌을 나타내었다.

1) 연구개발정보센터 슈퍼컴퓨팅사업단 선임연구원
2) 연구개발정보센터 슈퍼컴퓨팅사업단 책임연구원

2. 병렬화 기법

DS10 시스템에 대한 병렬 성능 분석에 대한 주요한 시험은 확장성의 검토이다. 이를 검증을 위해 이루어진 각 부분별 특징은 다음과 같다.

2.1 영역분할

본 연구에서는 전체 해석영역 격자계를 미리 분할하고 각 프로세서가 해당 영역 격자계를 입력으로 받는 SPMD(Single Program Multiple Data) 기법을 사용하였으며, 이러한 병렬처리를 행하는데 있어서 다음의 두 가지 경우를 시험하였다.

경우 1(C1): 전체는 동일한 격자수를 유지하고 CPU가 증가할 때 해당하는 격자수는 감소하게 하는 방법.
경우 2(C2): 단일 CPU에 해당하는 격자수가 항상 동일하게 유지하도록 전체 격자수를 증감하는 방법.

경우 1은 CPU수가 증가할 때 자료교환이 상대적으로 증가하게 됨으로 주어진 시스템에 대해 계산에 따른 통신성능을 비교하게 되며, 경우 2는 실제적으로 격자수가 증가함에 따라 병렬 성능이 확장성을 가지고 이루어지는지를 판별하게 된다.

2.2 프로그램 구성

유동 해석 과정은 입출력 단계, 해석 사전 단계, 해석 단계, 영역간의 자료교환단계, 수렴성 확인 단계, 경계조건처리 단계 그리고 실제 값을 계산하기 위한 후처리 단계로 구성되어져 있다. 각 과정에 대한 병렬화 과정을 설명하면 다음과 같다.

2.2.1 입출력 과정

단일 또는 다중 격자계를 Front-End 노드에 위치시킨 후(각 노드들은 100Mbps의 Fast Ethernet을 사용하여 NFS(Network File System)으로 연결되었음), 해당 노드들이 해당 자료를 입력하는 형태를 취하게 하였다. 이 방법은 입력자료를 분할시킬 필요 없이 쉽게 자료를 보관 및 분산시킬 수 있다. 그리고 노드 수가 적은 경우 또는 네트웍 성능이 좋은 경우에는 그 시간이 적게 걸리는 반면 노드 수가 많아질 경우 현재의 네트워크 환경에서는 성능이 저하될 것임을 예상할 수 있다. 대안의 방법이 각 노드에 해당되는 격자계 입력 파일을 만든 후 이를 서로 독립적으로 입력하는 방식이다. 각 자료의 출력은 해당 노드가 자신에 속한 격자계와 해석결과를 분리해서 출력하도록 하였다. 자세한 결과는 3장 병렬 성능분석에서 나타내었다.

2.2.2 해석 전후 처리 단계 및 수렴성 확인 단계

본 연구에서는 해석 시 기본이 되는 변수들(예: 각 방향 격자 수, 노드당 다중블럭 수, 전체 격자계 수, 각 격자계의 해당 노드 번호 등)을 공통적으로 가지고 있도록 하였다. 압력계수를 포함한 힘과 모멘트 계수를 계산하기 위해 namelist 입력 자료로 주어진 경계조건에서 해당되는 격자 경계만을 뽑아내어 계산한 후 MPI_ALLREDUCE[7]를 사용하여 계산하였다. 그리고 각 노드에서 계산한 RMS 오차를 MPI_ALLREDUCE로 계산하여 전체 수렴성을 확인하도록 하였다.

2.2.3 경계조건 처리

본 연구에서 병렬 경계처리를 위해 해당 경계면에 대해 상대편 경계면을 지정하게 함으로써 병렬 경계 처리가 격자의 형태에 무관하게 이루어지게 하였다. 또한, 임의의 CPU가 단일 및 다중블럭(multi-block) 격자를 포함할 수 있게 하였다. 실제로 2.1절의 경우에 64개의 다중블럭을 구성한 후 주어진 CPU는 각각 다수의 격자계를 포함하도록 하였다.(예로 4개의 CPU를 사용할 경우 1개의 CPU는 각각 16개의 다중블럭 격자를 포함한다.)

2.3 자료교환 방법

해석자(Solver)의 정확도를 유지하도록 자료교환을 수행했으며[8], 자료교환방법으로 SEND/RECV 방법(경우 3(C3))과 IRECV/SEND 방법(경우 4(C4))을 시험하였다. 이를 표[1]에 나타내었다. 각 경우의 알고리즘을 설명하면 다음과 같다. 경우 3은 PingPong[2]시험과 유사 한 형태를 가지며, 경우 4는 PingPing[2]의 형태를 갖는다. 경우 4의 라이브러리 형태를 좀 더 세분하게 보면 표[2]와 같다.

표 [1] 자료교환 방법의 예

경우 3(C3)	경우 4(C4)
do n = 1,nsurf call MPI_BARRIER call MPI SEND call MPI RECV end do	do n = 1,nsurf call MPI IRECV call MPI SEND call MPI_WAIT end do

표 [2] IRECV/SEND 자료교환 알고리즘

```
call MPI_IRECV(buf,icount,MPI_REAL,ncpu,tag,comm,ireq,ierr)
call MPI_SEND(buf,icount,MPI_REAL,ncpu,tag,comm,ierr)
call MPI_WAIT(ireq,istatus,ierr)
```

표[2]에서 point-to-point 자료교환의 특징을 이용하면 ncpu 값은 각 경우에 동일하게 된다. 위의 두 경우에 대해 2.1절의 경우 1(C1), 경우 2(C2)를 시험하였다. 이는 Cray T3E와 DS10에서 자료교환의 실제적 특성을 파악하는데 중요한 역할을 할 것이라 생각되며, 이것이 일반적으로 PC cluster에서 병렬 처리 코드를 작성할 때 대두되는 병렬 성능 및 확장성을 보장하기 위한 방법을 제시하여 줄 것이다.

3. 병렬 성능 분석

3.1 해석 예제

2.1절에서 언급한 경우 1의 격자계를 그림 1에 나타내었으며, 단일 블록의 격자 수는 17×17×9개, 총 블록수는 64개로 전체 격자수는 166,464개이다. 해석에 사용된 형상은 O-H 격자 형태의 ONERA M6 날개[10]로 자유흐름 마하수가 0.84 받음각이 3.06도인 경우이다.

2.1절에서 언급한 경우 2에 대한 단일 CPU의 격자계는 그림 2에 나타나 있다. 경우 2에서 단일 블록의 격자수는 140,481(129×33×33)개로 1개의 CPU에서 해석되며, 64개 CPU를 사용할 경우의 격자수는 1개의 CPU를 사용한 격자수에 비해 64배 많은 8,990,784개이다. 해석에 사용된 형상은 경우 1과 같이 O-H 격자형태로 ONERA M6 날개 뿌리(root)의 에어포일 형상을 갖고 스팬(span)방향으로 유한 길이를 갖는 날개이다. 격자계는 KGRID[10]를 사용하여 구성하였으며, 경우 2의 예제는 DS10에서만 해석이 수행되었다. Cray T3E에서 프로그램 메모리 한계로 경우 2의 예제는 해석이 수행되지 못했다.

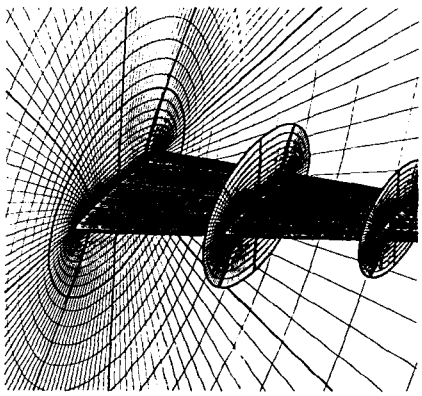

그림 1: 64개의 영역으로 분할된 ONERA M6 날개 형상: Case 1

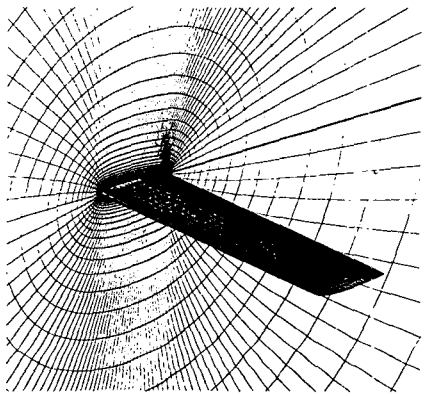

그림 2: 단일 영역의 유한 날개 형상: Case 2

3.2 해석 결과 검증

해석 결과의 타당성을 검증하기 위하여 DC10과 Cray T3E에서 64개 CPU를 사용하여 ONERA M6 날개를 해석하였다. 그림 3에 등압력 선도를 나타내었다.

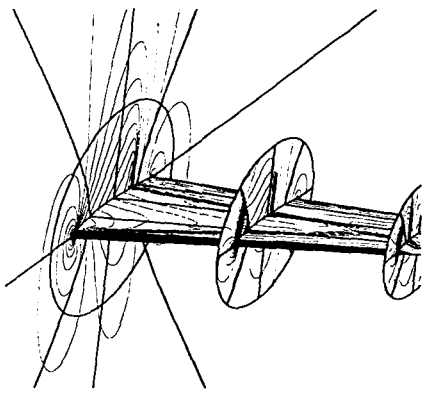

그림 3: ONERA M6 날개상의 등압력 선도(Case 1): $M_\infty = 0.84,\ \alpha = 0^\circ$

3.3 병렬 계산 성능

3.3.1 경우 1(C1)의 해석 결과

그림 4와 5에서 각각 해석에 소요된 시간과 병렬 성능(Speedup)을 계산하였다. 각 경우의 해석 결과는 해석자와 자료교환에 소요된 시간만을 포함한 결과이며, 입출력 및 전후 처리등에 대한 결과는 분리하여 성능을 검토하였다.

그림 6 ~ 9는 그림 4의 계산시간에 대한 해석자와 자료교환의 점유율을 나타내고 있다. DO 루프(loop)에서 MPI_BARRIER를 사용하여 SEND/RECV를 사용할 경우(C3), 해석 코드를 작성하기가 쉬운 반면, 성능이 저하됨을 볼 수 있다. 이는 당연한 결과로써 노드간의 네트워크 형태에 따른 latency와 bandwidth가 크게 차이가 나며 이는 병렬성능을 결정하는 주요한 요인이다. DS10에서 사용된 Fast Ethernet을 사용할 경우 그 차이가 매우 심함을 볼 수 있으며, Cray T3E는 네트워크 성능이 매우 뛰어나 적절한 프로그램에 대해서 우수한 성능을 보이고 있다. Latency를 줄이기 위한 IRECV/SEND

방법을 사용할 경우 자료교환에 대한 시간은 매우 적게 나타남을 볼 수 있다. 이에 대한 좀 더 세분된 결과는 PMB[2]를 사용한 결과[1]에서도 볼 수 있다.

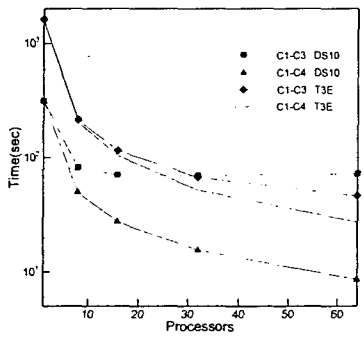

그림 4: 해석 수행 시간 비교

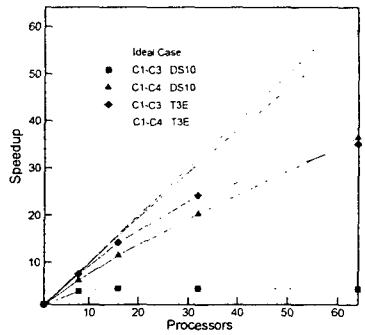

그림 5: 병렬 성능 비교

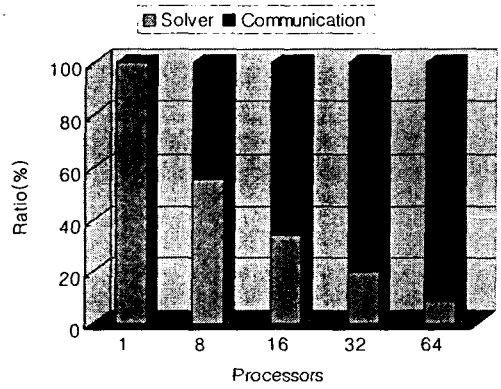

그림 6: 해석 성능 비교
(C1-C3: DS10 경우)

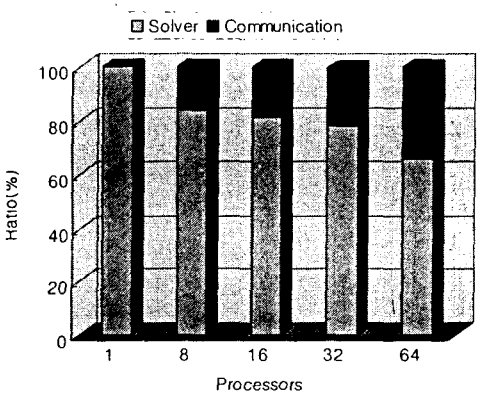

그림 7: 해석 성능 비교
(C1-C4: DS10 경우)

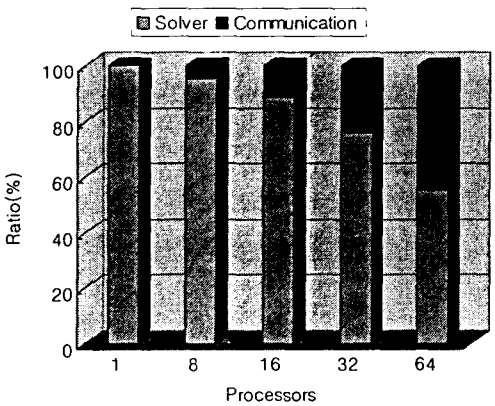

그림 8: 해석 성능 비교
(C1-C3: T3E 경우)

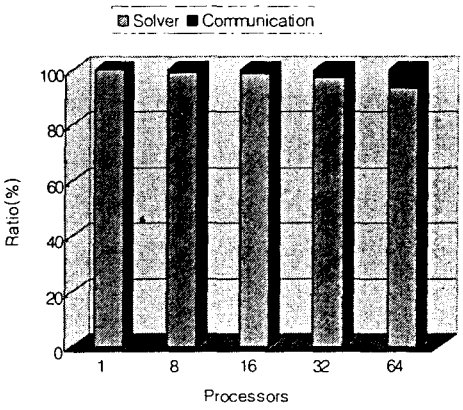

그림 9: 해석 성능 비교
(C1-C4: T3E 경우)

그림 10, 11은 각 프로세서 수에 따른 격자 입력 시간과 해석자에 대한 전 처리과정 시간을 나타내었다. CPU가 증가함에 따라 격자수가 감소함에도 불구하고 DS10에서는 32개의 CPU 이상을 사용할 때 네트워크 성능이 급격히 저하됨을 볼 수 있는 반면, Cray T3E에서는 점진적으로 성능이 저하됨을 보이고 있다. 현재의 결과에 대해 고려해야 할 사항은 두 가지이다. 첫째는 하나의 파일을 각 CPU가 동시에 접근하여 자신에 맞는 격자계를 읽게 됨으로, CPU 수가 증가함에 따라 각 CPU는 반복 횟수가 증가하게 된다. 둘째는 NFS로 연결된 네트워크의 성능이다. DS10의 경우 CPU속도는 T3E에 비해 빠르므로 반복횟수의 계산시간은 단축될 수 있으나 네트워크 성능의 영향이 큰 영향을 미쳐 성능의 저하를 나타내게 된다. 이러한 현상을 해결하기 위해 가능한 파일 입출력을 독립된 노드에서 행하고 횟수를 줄이는 방법을 모색하여야 한다. 그림 12에 해석자만의 계산시간을 나타내었다. 그림에서 각 경우 CPU수를 증가시킴에 따라 비례적으로 계산시간이 줄어듦을 볼 수 있으며, DS10이 단일 CPU에서 Cray T3E 보다 약 4.5배정도 빠르다. 그림 13은 각 경우에 대한 자료교환 시간을 나타낸 것으로 SEND/IRECV를 사용한 경우 전체 계산시간은 감소함을 보이고 있으며, 반면 SEND/RECV를 사용한 경우는 계산시간이 증가함을 볼 수 있다. 그림 14는 공력계수(aerodynamic coefficient)들을 구하는 과정으로 각 CPU에서 해당되는 경계(고체 경계면)가 있으면 이에 대해 계산을 실행하고 계산된 값은 MPI_ALLREDUCE를 사용하여 전체 값을 계산하게 된다. 따라서 32개의 CPU 이상에서 적절한 값으로 수렴함을 볼 수 있다.

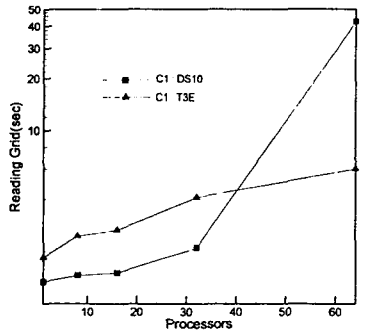

그림 10: 프로세서 수에 따른 입력 시간

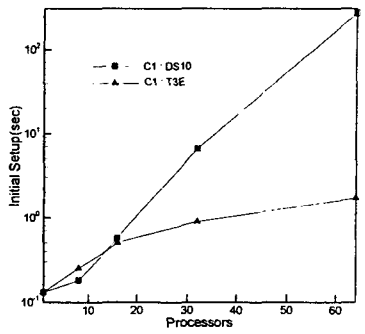

그림 11: 프로세서 수에 따른 전처리 시간

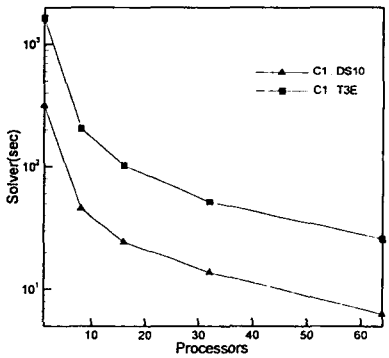

그림 12: 프로세서 수에 따른 해석 시간

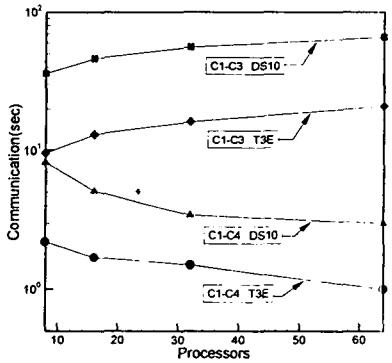

그림 13: 프로세서 수에 따른 자료교환시간

3.3.2 경우 1(C2)의 해석 결과

그림 4, 5와 같이 그림 15, 16은 각각 해석에 소요된 시간과 병렬 성능(Speedup)을 나타

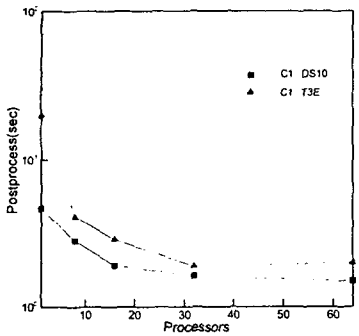

그림 14: 프로세서 수에 따른 후처리 시간

낸다. 각 경우의 해석 결과는 해석자와 자료교환에 소요된 시간만을 포함한 결과이며, 입출력 및 전후 처리 등에 대한 결과는 분리하여 성능을 검토하였다.

그림 15에서 C2-C4(IRECV/SEND)를 결합한 예제에서 격자수가 해당되는 CPU 수만큼 증가하여도 계산시간은 일정한 경향을 보이는 반면, C2-C3(SEND/RECV)를 사용한 경우는 자료교환에 따른 latency 영향으로 계산시간이 증가함을 볼 수 있다. 각 경우에 대해 그림 16에서 보면 C2-C3를 사용할 경우 16개 이상의 CPU를 사용할 경우 성능이 급격히 저하됨을 볼 수 있으며 C2-C4를 사용할 경우 매우 우수한 병렬 성능을 보이고 있다. 그림 17, 18은 그림 15의 계산시간에 대한 해석자와 자료교환의 점유율을 나타내고 있다. SEND/RECV를 사용할 경우(C3)는 경우 1(C1)에서와 같이 IRECV/SEND 방법을 사용할 경우에 비해 latency 값의 증가로 CPU 수가 증가하면 해석시간에 비해 자료교환의 비율이 크게 증가함을 볼 수 있다.

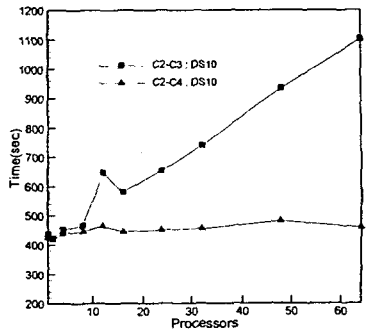

그림 15: 해석 수행 시간 비교

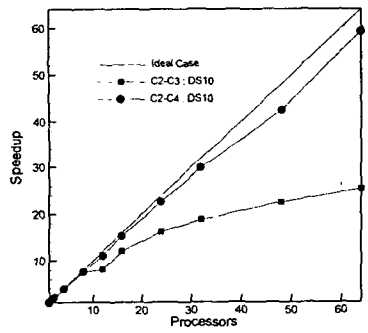

그림 16: 병렬 성능 비교

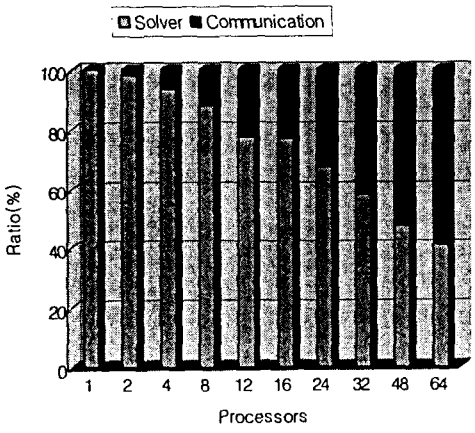

그림 17: 해석 성능 비교
(C2-C3: T3E 경우)

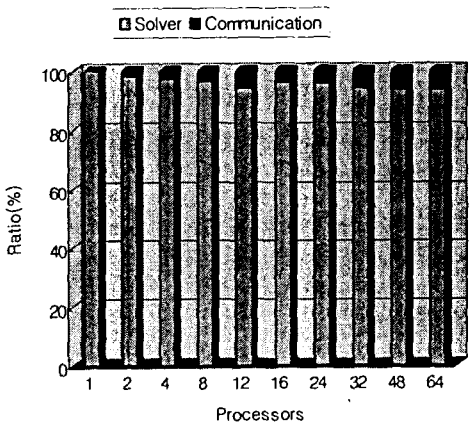

그림 18: 해석 성능 비교
(C2-C4: T3E 경우)

그림 19에서 CPU와 격자수의 증가에 따른 입력 시간을 나타내었으며 그림 20, 21에서 각각 전/후처리에 소요된 시간을 나타내었다. 경우 1의 예제의 경우는 CPU수가 증가함에 따라 격자수가 감소하는 반면, 경우 2의 예제는 CPU수가 증가함에 따라 격자수가 비례하여 증가한다. 따라서 전반적으로 계산시간은 점진적으로 증가하게 되며 이는 네트워크의 성능이 CPU수가 증가함에 따라 저하됨을 알 수 있다. 그림 22에서 해석에만 소요된 시간을 나타내었으며 격자수의 증가에 따라 일정한 값을 보이고 있다. 그림 23은 CPU수 증가에 따른 자료교환에 소요되는 시간을 나타내었다. 그림 13의 경우는 격자수가 CPU수의 증가함에 따라 감소함으로 IRECV/SEND를 사용하면 전체 계산시간은 감소함을 보이는 반면, 현재의 경우는 격자수가 동시에 증가함으로 IRECV/SEND를 사용해도 전체 계산시간이 증감함을 보이고 있다. 그러나 이 값은 SEND/RECV를 사용할 경우 지속적으로 증가하는 반면 16개 이상의 CPU수에서 거의 일정한 값을 나타내고 있음을 볼 수 있다. 그림 15 ~ 23사이에서 12개의 CPU를 사용할 경우, 해석의 결과가 예상치 못한 경향성을 보이고 있다. 이것에 대한 확실한 원인은 판별하지 못하였으나 동시사용자의 CPU 점유에 따른 일시적 현상으로 판단하고 있다.

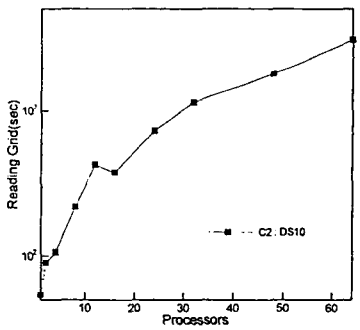

그림 19: 프로세서 수에 따른 입력 시간

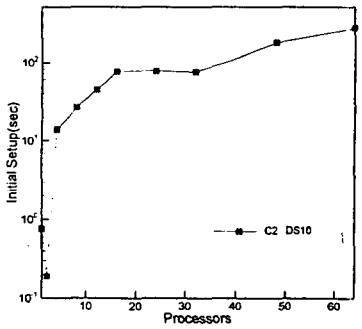

그림 20: 프로세서 수에 따른 전처리 시간

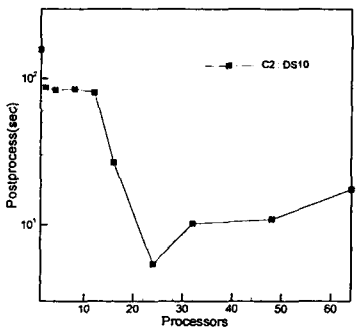

그림 21: 프로세서 수에 따른 후처리 시간

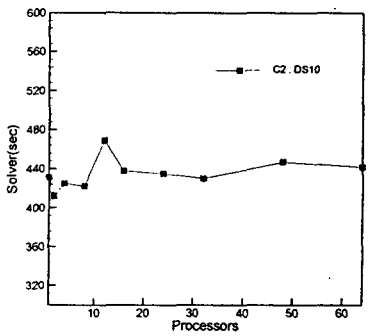

그림 22: 프로세서 수에 따른 해석 시간

4. 결 론

KORDIC 슈퍼컴퓨터 센터에서 수행중인 TeraCluster 프로젝트의 1단계로 DS10 시스템으로 구성된 병렬컴퓨터에 대해 실제 CFD 코드를 성능분석하였다. 사용된 CFD 코드는 구성된 시스템에 대해 전반적으로 우수한 해석 성능은 보이고 있으며, 64개의 CPU를 사용한

경우 60배 정도의 성능을 얻을 수 있었다. 중요하게 고려되어야 할 사항은 입출력과 자료교환 방식의 사용이다. Cray T3E에 비해 단일 CPU 성능은 우수하나 네트워크 성능이 빈약하여 latency가 크므로, 가능한 MPI_BARRIER 또는 대용량의 Global Communication을 제거한 후 병렬코드를 작성함이 유리하다. 또한 대용량의 자료를 동시에 접근할 경우 Fast Ethernet 카드로 구성된 NFS는 64개의 CPU를 동시에 작동시키는데 많은 시간을 소비한다. 따라서 각 CPU가 그 일을 담당하도록 프로그램을 하는 것이 유리하리라 생각된다. 이러한 약점을 제거하기 위해 Front-End와 각 노드간에는 Gigabit 카드를 사용하는 방법이 고려될 수 있다. 자료교환 방식은 경계조건을 단순화하는 point-to-point 접근 방법과 Non-blocking send/recv를 사용하여 자료교환이 모든 CPU에 대해 연속적으로 진행되도록 프로그램하는 것이 중요하며, 조건들이 만족될 때 보다 큰 수의 CPU를 갖는 PC cluster에 대해서도 확장성을 가질 것이라 예측된다. 본 연구에서 분석 검토된 자료는 차후 보다 효과적인 cluster 구축에 이용될 수 있으며, 현재 많이 보급되어 있는 실험실 단위의 병렬컴퓨터 성능을 분석할 수 있는 도구로 이용될 수 있다.

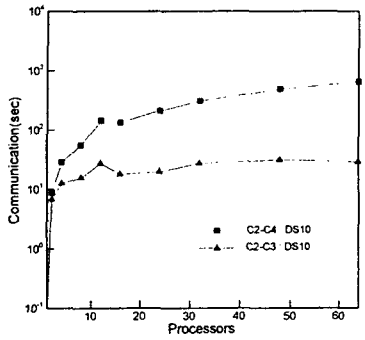

그림 23: 프로세서 수에 따른 자료교환 시간

5. 참고문헌

[1] 이상산 외 8인, "TeraCluster 구축과 실험 1차 보고서 ", KORDIC 슈퍼컴퓨팅센터 내부 보고서, 2000.

[2] Pallas MPI Benchmarks - PMB, Part MPI-1.

[3] http://www.nas.nasa.gov/Research/Software/swdescription.html

[4] C.Hirsch, "Numeraical Computation of Internal and External Flows 1, 2", John Wiley

[5] P.L.Roe, "Approximated Riemann Solvers, Parameter Vectors and Difference Scheme", Journal of Physics, Vol.43, pp.357-372,1981

[6] van Leer, "Towards the Ultimate Conservative Difference Scheme IV. A New Approach to Numerical Convection", Journal of Physics, Vol. 23, pp.276-299, 1977

[7] M. Snir, S.Otto, S.Huss-Lederman, D.Walker and J. Dongarra, "MPI: The Complete Reference", The MIT Press, 1996.

[8] 조금원, 권장혁, 이승수, "비정상 Euler 방정식을 이용한 Chimera 기법의 병렬처리에 관한 연구", 한국전산유체공학회지, 제4권 제 3호, 1999년 12월, pp. 52-62.

[9] http://amber.aae.uiuc.edu/~m-selig/ads/coord_database.html

[10] 김윤식, "KGRID Developers' Guide", 한국과학기술원 항공우주공학과, 1999.