2차원 유동장 해석에서 영역분할법에 따른 병렬효율성 검토

A Study on Effect of Domain-Decomposition Method on Parallel Efficiency in 2-D Flow Computations

이상열[1], 허남건[2]
Sangyeul Lee , Nahmkeon Hur

2-D flow fields are studied by using a shared memory parallel computer with a parallel flow analysis program which uses domain decomposition method and MPI library for data exchange at overlapped interface. Especially, effects of directional domain decomposition on parallel efficiency are studied for 2-D Lid-Driven cavity flow and flow through square cavity. It is known from the present study that domain decomposition along the main flow direction gives better parallel efficiency in 1-D partitioning than along the other direction. 2-D partitioning, however, is less sensitive to flow directions and gives good parallel efficiency for most of the cases considered.

1. 서론

Von Neumann 형태의 컴퓨터가 제시된 이후, 단일 프로세서를 사용하는 컴퓨터는 빠른 속도로 발전해 왔다. 하지만 단일 프로세서의 발전이 이제 한계에 도달하고 있으며 또한, 메모리 등의 자원을 확장하는 데에도 문제가 있는 것이 사실이다. 따라서 이러한 문제점의 대안으로 발전한 것이 병렬형 슈퍼컴퓨터이며 많은 계산 량을 필요로 하는 과학 및 공학 분야에서 점차 그 사용이 증가하고 있다.

전산 유체역학 분야에서는 특히, 난류, 다상유동, 연소등 좀 더 높은 컴퓨팅 파워를 필요로 하는 부분에 이용될 수 있으며 현재 많은 연구가 진행중이다. 국외에서는 K. Shimano와 C. Arakawa[1]등은 영역분할법을 이용하여 급확대유동 및 Cavity Flow에 대해서 수치 해석하였고 E. Shima[2]는 비정렬격자계를 사용하는 Navier-Stokes Solver에 영역분할법을 적용하여 병렬화하였다. F. Desprez 및 M. Grbey[3]는 연소문제에 병렬화 기법을 사용하여 수치 해석하였다. 국내의 연구로는 고덕곤[4]이 유동해석 프로그램을 병렬화하여 공기역학문제에 적용하였으며, 곽호상[5]은 비정상 자연대류 문제에 적용하였다. 권장혁[6]등은 3차원 다중격자 DADI방법을 병렬화하여 효율적으로 수행하였다. 또한, 강동진[7]은 workstation cluster를 이용하여 2차원 Navier-Stokes 방정식 solver에 영역분할법을 적용하여 병렬화하였다. 그리고 현재 사용되는 대부분의 상용 코드들이 이미 병렬화 되어 사용되고 있거나 병렬화가 진행 중에 있다.

본 연구에서는 data의 교환을 위한 방법으로 message passing 방법[8]을 이용하였고 문제의 분할방법으로는 영역분할법을 사용하였다. 2차원 유동장 해석에서 영역을 분할할 경우 1-D, 또는 2-D로 분할하는 차원별 분할이 가장 손쉬운 분할법이며 필요에 따라 프로

1) 서강대학교 대학원(e-mail : lesan@euler.sogang.ac.kr)
2) 서강대학교 기계공학과(서울시 마포구 신수동 1번지 , e-mail : nhur@ccs.sogang.ac.kr)

그래머가 임의로 분할할 수도 있다. 이 경우 분할된 영역의 경계에서는 인접한 영역의 계산 결과를 사용하는 임의의 경계 조건을 사용하게 된다. 일반적으로 영역을 분할 할 경우, data의 교환 회수를 줄일 수 있는 방향으로 분할하여야 하며 각 프로세서에 동일한 load를 갖도록 분할하는 것이 유리하다. 또한 분할된 영역의 경계면에서 계산 결과가 실제 얻고자하는 해에 빨리 수렴할수록 계산 효율이 향상될 것이다. 이는 convection이 지배적인 유동장의 경우 영역분할 방향이 병렬 효율에 영향을 미칠 수 있음을 생각해 볼 수 있다. 따라서, 본 연구에서는 몇 가지 2차원 유동장에서 1-D 및 2-D partitioning에 따른 병렬효율성을 검토하고 또한, 1-D partitioning중에서 x축으로의 분할과 y축으로의 분할 등에 따른 병렬 효율성을 검토하였다.

2. 고려된 2차원 유동장 문제

본 연구에서 해석한 2차원 유동문제는 2가지로 하나는 Lid-Driven Cavity Flow이며 또하나는 정사각형의 용기 내에서 좌측 아래부분이 inlet이며 우측 위부분이 outlet으로 주어진 Flow through Square Cavity 문제로 Fig. 1과 Fig. 2에 각각 도시하였다. 사각 용기는 x방향 길이와 y방향 길이가 서로 같다. 계산은 FVM과 SIMPLE Algorithm을 사용하는 유동 해석 프로그램을 병렬화하여 수행하였다. 각 문제에 대해서 Re=100과 Re=1500일 경우로 나누어 계산하여 크게 4가지의 경우로 나누어지며 각 문제에 대해서 단일 프로세서를 이용한 계산과 병렬 프로세서를 이용한 계산을 수행하였다. 단일 프로세서를 이용할 경우, 계산에 사용된 격자는 모두 정렬격자이며 96×96의 격자수를 가진다. 병렬 프로세서를 이용한 계산의 경우, 영역분할법으로 96×96격자계를 사용되는 프로세서의 수로 나누게 된다. 또한, 정렬 격자계를 사용하므로 1-D partitioning중 x축 분할과 y축 분할에서 같은 수의 격자로 분할하였다. 프로세서 4개를 사용할 경우에는 1-D partitioning과 2-D partitioning 모두 동일한 격자수를 가지게 되며 data를 교환하는 격자의 수도 동일하다. Fig. 3에 1-D partitioning 과 2-D partitioning을 나타내었으며 더불어 각각 x, y 축에 대한 분할도 함께 나타내었다.

3. 병렬처리 방법

연구에 사용된 병렬컴퓨터는 본 연구실에서 보유하고 있는 Silicon Graphics Inc.의 Origin 2000으로 메모리를 프로세서들이 공유하는 공유메모리[9],[10] 형태의 병렬형 컴퓨터이다. 총 4개의 R10k 프로세서를 가지고 있으며 512MB의 메모리와 각 프로세서마다 1MB의 외부 캐쉬를 가지고 있다. Storage는 총 18GB이다. 공유메모리임에도 불구하고 본 연구에서 message passing 방법을 이용한 것은 Silicon Graphics Inc. 에서도 MPI library를 제공하고 있으며, 향후 Network Cluster를 이용한 병렬계산을 수행하고자 하며 또한 분산 메모리[9],[10] 형태의 병렬 컴퓨터에서도 계산이 가능하도록 하기 위함이다. Data 교환에 사용된 MPI routine은 blocking 방법으로 메시지를 전달하는 MPI_SEND, MPI_RECV[8]를 사용하였다.

본 연구에서는 정렬격자계를 사용하였으므로 각 프로세서에 동일한 개수의 계산 격자를 가지도록 분할하였다. 또한, 2차 정확도의 공간 이산화를 얻기 위하여 분할면에 대하여 2개의 격자를 중첩시키는 중첩격자계를 이용하였다. 유동 방향과 partitioning 방향과의 관계를 알아보기 위하여 1D partitioning 중에서 x축에 대한 분할과 y축에 대한 분할을 각각 하여 계산하였고 프로세서수가 4개일 경우는 2D partitioning도 함께 계산하였다. 각 분할 영

역은 독립적으로 행렬 문제를 풀게되며 그 계산 결과를 분할면에서 교환하는 방식을 이용하였다. 그 방법을 Fig. 4에 나타내었다. 이런 방법을 이용할 경우 전체 행렬과 같은 결과를 얻을 수는 없지만 정상상태의 문제에서 그 해에 영향을 주지 않는다[6]. 따라서, 본 연구에서는 각 분할영역에서 각각 TDMA방법으로 물리량을 구하고 교환하는 방법을 사용하였다.

영역의 분할은 계산에 사용되는 프로세서의 수와 관계없이 분할이 가능하고 경계조건은 특정 프로세서가 다른 프로세서들에게 전달하는 것이 아니고 각 프로세서들이 자신의 영역을 할당받게 되면 각 영역에 맞게 자동으로 분할되어 계산을 수행하도록 하였다. 또한, 계산이 완료된 이후에는 프로세서들이 할당받은 영역의 결과만 파일로 출력하도록 하였고 후처리를 위하여 다시 각 결과를 하나로 합치는 과정을 수행하도록 하였다. 이는 각 영역에서 계산된 결과를 특정 프로세서가 수집하고 파일로 출력하는데는 많은 시간이 소요되므로 이를 피하기 위함이다.

3. 계산 결과

계산 결과를 측정하는데 사용된 시간은 순수 계산시간과 통신에 사용된 시간을 모두 합한 wall time이다. 또한, 각 문제에 대해서 단일 프로세서를 이용한 경우와 병렬 프로세서를 이용한 경우의 sweep수와 완화계수는 동일하게 하였다.

본 계산에 앞서 관로 유동과 같은 2차원 유동에 대해서 병렬 계산을 수행하였다. 즉, 위와 아랫면에 벽면에 존재하고 좌측면 전체에서 유체가 들어오고 우측면 전체를 통하여 나가는 문제를 1-D partitioning에서 x, y 축에 대해서 영역 분할을 통하여 병렬 계산을 수행한 결과, y축에 대한 분할(Fig. 3b)의 경우 해가 수렴을 하지 않았고 x축에 대한 분할(Fig. 3a)에서만 해가 수렴을 하였다. 이것은 영역 분할을 주유동 방향에 수직하게 분할하는 것이 병렬 효율을 좋게 한다는 것을 뜻한다.

Fig. 5에서 Fig. 8까지는 각 계산 경우에 있어서 speed up을 나타낸 그래프이다. 경우에 따라서 2배, 3배, 4배 이상의 speed up이 나타나고 있는데 이는 본 계산에 사용된 시스템이 크로스바 스위칭을 사용하는 공유 메모리[9],[10] 형태로 각 영역간의 통신이 시스템 구조내에서 이루어지고 있으며 또한, 격자수가 많은 편이 아니므로 통신에 의한 시간 소모보다는 영역분할에 의한 계산 시간 감소가 더 큰 작용을 한 것으로 생각된다. 프로세서를 많이 사용함에 따라 효율이 감소하는 경우는 분할면이 많아짐에 따라 그 만큼의 수렴성이 감소하기 때문인 것으로 생각된다. 이 요인의 정확한 측정을 위하여는 계산 시간과 통신 시간을 따로 측정하여 비교하여 보는 것이 필요하나 본 연구에서는 수행하지 않았다.

Fig. 5와 Fig. 6을 살펴보면 Lid-Driven Cavity Flow(Fig. 1 참고)에 있어서는 Re=100 및 Re=1500일 때 모두 1-D partitioning에서 y축으로의 분할이 x축으로의 분할보다 좋은 병렬 효율을 보이고 있다. 또한, Fig. 7과 Fig. 8을 보면 Flow through Square Cavity (Fig. 2 참고)에서는 Re=100일 때는 y축으로의 분할이 좋은 병렬 효율을 보였고 Re=1500일 때는 x축으로의 분할이 좋은 병렬효율을 보였다. 위의 결과에 대해서 분할면에서의 u방향 속도 성분과 v방향 속도 성분의 절대값을 취하여 서로 비교하여 보았다. Flow through Square Cavity에서는 Re=100에서는 y축에 대한 분할면에서의 v속도 성분이 x축에 대한 분할면에서 u방향의 속도 성분보다 큰 값을 가지는 것을 확인할 수 있었다. 그리고 Re=1500에서는 이와 반대적으로 x축에 대한 분할면에서 u방향의 속도 성분이 y축에 대한 분할면에서 v방향 속도 성분보다 좀 더 큰 값을 가졌다. Lid-Driven Cavity Flow에서는 Re=100 및 Re=1500 모두에서 분할면에서의 u방향 속도 성분이 v방향의 속도 성분보다 큰 값을 가지는

것으로 나타났다. 이는 Lid-Driven Cavity Flow에서는 위 벽면의 움직임에 의한 u방향의 속도 성분이 너무 큰 것으로 생각된다. 따라서, 이 부분을 제외하면 주된 속도 방향이 v방향이 될 것으로 생각된다. 또한, Re 수가 높을수록 특정 partitioning에서 높은 병렬 효율을 가지는 것을 볼 수 있었다. 이는 전체 속도 성분이 증가함에 따라 그 만큼의 분할면에서의 전파성이 커진 것으로 생각된다.

2-D partitioning 에서는 대체적으로 방향성에 상관없이 비교적 좋은 병렬 효율을 가지는 것을 볼 수 있었다. 다만, 유동이 특정 부분으로 치우치는 경향이 생기는 Flow through Square Cavity에서 Re=1500일때 좋지 않은 병렬 효율을 보였다.

4. 결론

본 연구에서는 공유 메모리 형태의 병렬 컴퓨터에서 영역 분할법을 사용하고 중첩 격자계에서 data 교환을 위하여 MPI Library를 이용하는 병렬 유동해석 프로그램을 작성하여 몇 가지 2차원 유동장 문제를 해석하였다. 특히, 차원별 영역분할을 통하여 특정 유동 형태에 따른 x축으로의 1-D partitioning과 y축으로의 1-D partitioning의 병렬 효율을 비교 검토하였고, 2-D partitioning의 병렬 효율과도 서로 비교하여 보았다.

해석결과로부터 1-D partitioning 에서는 특정 유동방향에 대해서 수직하게 분할하는 것이 병렬 효율을 높이는 것을 알 수 있었으며, Convection이 지배적인 높은 Re수의 유동에서는 특정 partitioning 방향에서 높은 병렬 효율을 나타냄을 알 수 있었다. 2-D partitioning는 이에 비해 유동의 방향 등에 영향을 덜 받으며 전체적으로 좋은 병렬 효율을 나타내었다. 따라서, 유동의 방향 및 partitioning 방법이 병렬 효율에 많은 영향을 주고 있음을 알 수 있었다.

실제 병렬유동해석에서는 본 연구에서 연구한 유동방향을 고려한 방법이외에도, 통신시간의 최소화를 위한 분할방법과 sweep수의 조정, 또는 컴파일러의 option 등에 따라서도 병렬 효율성이 많은 영향을 받을 수 있다. 그러므로 이러한 모든 요건을 잘 tuning 시키는 것이 최상의 병렬 효율을 얻을 수 있는 방법이라 생각된다.

후기

본 연구는 서강대학교 1997년도 교내연구비로 수행되었기에 학교측에 감사드린다.

참고문헌

[1] K. Shimano, C. Arakwa, "Numerical Simulation of Incompressible Flow on Parallel computer with the Domain Decomposition Technique," Parallel Computational Fluid Dynamics - New Algorithms and Apllications, Elsevier Science B.V. , North Holland, 1995.

[2] E. Shima,"Domain Decomposition Method for Unstructured Grid Navier-Stokes Solver," Parallel Computational Fluid Dynamics - New Algorithms and Apllications, Elsevier Science B.V. , North Holland, 1995.

[3] F. Desprez, M. Garbey, "Parallel Computation of Combustion Problem," Parallel Computational Fluid Dynamics - New Trends and Advances, Parallel Computational Fluid Dynamics - New Trends and Advances, Elsevier

Science B.V. , North Holland, 1995.
[4] 고덕곤, "비점성 압축성 코드의 병렬화 기법에 의한 슈퍼컴퓨터 CRAY T3E의 성능 분석," 1997년 한국 전산유체 공학회 추계학술대회 논문집, 1997.
[5] 곽호상, "영역분할법에 의한 SIMPLER 기법의 병렬화," 1997년 한국 전산유체 공학회 추계학술대회 논문집, 1997.
[6] 권장혁 외, "3차원 다중격자 DADI방법의 병렬처리," 1998년 한국 전산유체 공학회 춘계학술대회 논문집, 1998.
[7] 강동진, "CFD Experiences on a Cluster of Workstation," 병렬프로그래밍 Workshop proceedings, 1998.
[8] Jack Dongarra, et al. , "MPI: The Complete Reference," MIT Press, 1996.
[9] Ian Foster,"Designing and Building Parallel Programs," Addison-Wesley, 1995.
[10] Lou Baker, et al. ,"Parallel Programming," McGRAW-HILL, 1997.

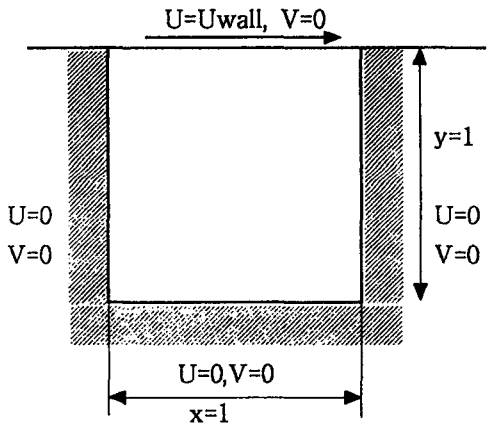


Fig. 1 Lid-Driven Cavity Flow

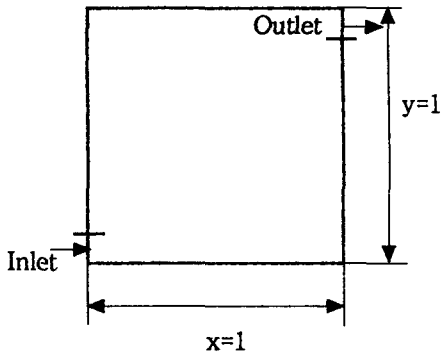
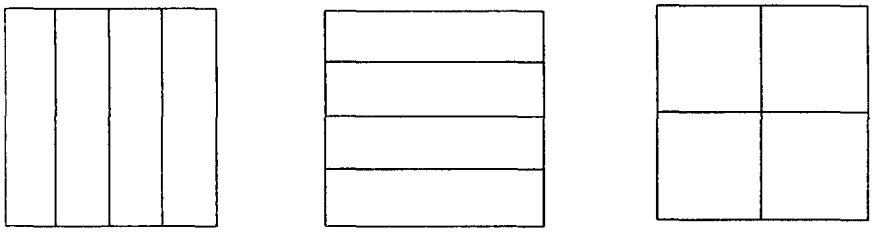


Fig. 2 Flow through Square Cavity



(a) 1-D Partitioning in x (b) 1-D Partitioning in y (c) 2-D Partitioning

Fig. 3 1-D Partitioning and 2-D Partitioning

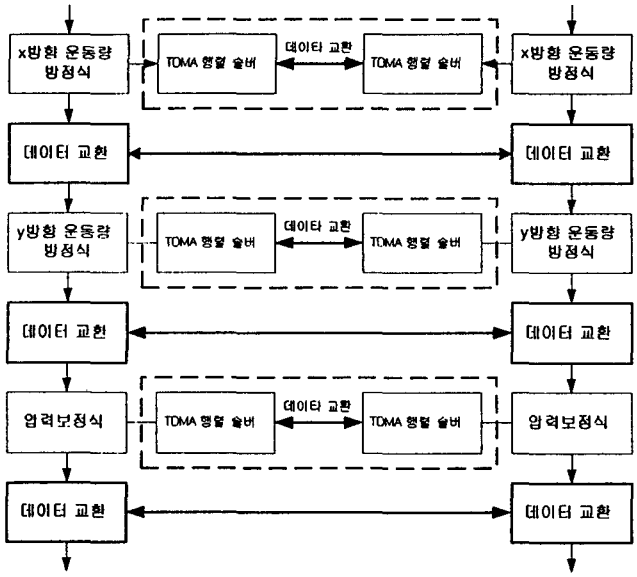


Fig. 4 Flow Chart of Parallel Program

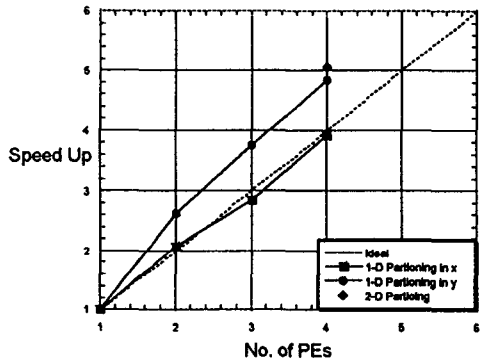


Fig. 5 Speed Up of Lid-Driven Cavity Flow(Re=100)

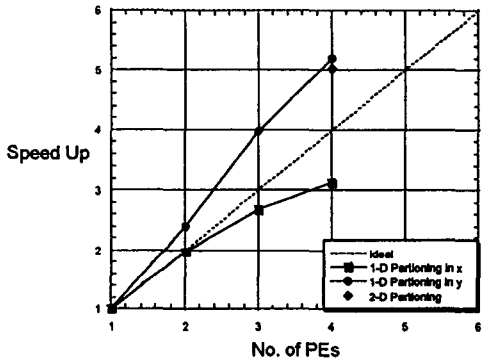


Fig. 6 Speed Up of Lid-Driven Cavity Flow(Re=1500)

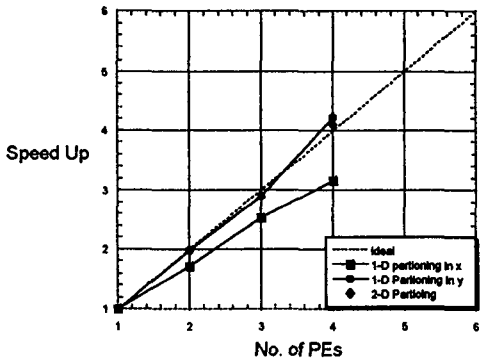


Fig. 7 Speed Up of Flow through Square Cavity(Re=100)

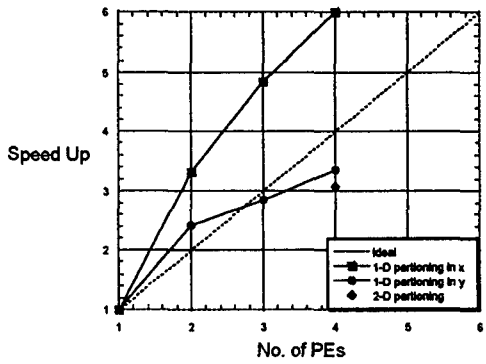


Fig. 8 Speed Up of Flow through Square Cavity(Re=1500)