PC 클러스터를 위한 정렬 중첩 격자의 병렬처리

김 유 진[1*], 권 장 혁[2]

PARALLEL IMPROVEMENT IN STRUCTURED CHIMERA GRID ASSEMBLY FOR PC CLUSTER

Eugene Kim, Jang Hyuk Kwon

Parallel implementation and performance assessment of the grid assembly in a structured chimera grid approach is studied. The grid assembly process, involving hole cutting and searching donor, is parallelized on the PC cluster. A message passing programming model based on the MPI library is implemented using the single program multiple data(SPMD) paradigm. The coarse-grained communication is optimized with the minimized memory allocation because that the parallel grid assembly can access the decomposed geometry data in other processors by only message passing in the distributed memory system such as a PC cluster. The grid assembly workload is based on the static load balancing tied to flow solver. A goal of this work is a development of parallelized grid assembly that is suited for handling multiple moving body problems with large grid size.

Key Words: 병렬 처리(Parallel Processing), 정렬 중첩 격자(structured Chimera grid), PC 클러스터

1. 서 론

비행중인 항공기로부터의 연료탱크의 분리, 다단계 로켓의 분리 등 물체간의 상대운동이 있는 문제는 전산유체역학이 실험의 한계를 대체하는 큰 부분 중의 하나라고 할 수 있으며, 이에 대한 많은 접근들이 이루어지고 있다[1,2]. 이와 같은 문제의 접근에 있어 가장 많이 사용되어지는 방법이 비정렬 격자기법과 Chimera 기법이라 불리는 중첩 격자기법이다[3]. 전자의 경우 점성유동의 제한성과 움직이는 격자계에 대한 전체 격자의 생성에 대한 제약점 때문에 그 응용범위가 제한되고 있다. 반면 후자의 경우는 매 시간 단계에서 격자를 새롭게 구성하지 않으므로 적은 삽간량을 갖게 되는 장점을 가지고 있다.

이와 같은 중첩격자계를 이용한 상대운동 해석은 비점성 계산에 국한되어져 왔으나, 최근 들어 난류 유동장 또는 다물체(multi-body)의 분리유동 해석 등에 적용되어지고 있다. 이러한 유동장은 구성된 격자계의 조밀도와 상대운동 모사를 위해 충분히 넓은 영역을 격자계로 표현해야 하는 이유로 상당한 계산비용을 요구하며, 유동해석자의 경우는 이러한 방대한 계산의 효율성 증진을 위해 병렬화 기법에 대한 연구와 시도가 많이 이루어져 왔다. 이에 따른 Chimera 기법에 대한 병렬화 또한 시도 되고 있으나, 격자계 구성시간이 전체 계산시간의 10% 내외로 실제 해석자의 계산시간에 비해 비교적 작은 부분을 차지하기 때문에 격자구성 모듈의 명확한 병렬화와 그 성능 검증은 많지 않다.

Steger 등에 의해 처음 제안된 일반적인 Chimera 기법은 크게 고체경계내부의 계산점을 제외시키는 홀 절단, 삽간점의 탐색 그리고 영역 연결 과정으로 구성된다[3]. 정렬 격자에서의 Chimera 격자 구성은 다중 블록(multi-block)등을 고려한 작업분배에 따라 각 노드마다 다른 수의 블록을 가지게 되며, 격자 구성 시 사용되어 지는 알고리즘은 블록 단위 또는 고체경계를 구성하는 물체의 단위로 이루어진다.

본 연구에서는 일반적인 Chimera 정렬 격자계에서 가장 많이 사용되는 Hole-Map 알고리즘[2]과, stencil-walk 방법과 구배찾기(gradient search) 방법을 함께 사용하는 2단계탐색과정[2]을 통한 중첩격자 구성 모듈의 병렬처리 알고리즘을 제안한다. 개발된 알고리즘은 비점성 정상 상태 계산을 통하여 검증되고, 개별적으로 격자 구성 모듈의 성능은 다물체의 중첩격자 구성을 통해 측정되었다.

1 학생회원, 한국과학기술원, 항공우주공학전공
2. 종신회원, 한국과학기술원, 항공우주공학전공
* Corresponding author E-mail: jhkwon@kaist.ac.kr

2. Chimera 기법의 병렬처리

2.1 PC 클러스터 환경

방대한 계산에 대한 필요성은 고성능 컴퓨터의 필요성을 가속화 시켰으며, 최근 들어 이러한 고성능 컴퓨터의 경향은 대규모 단일 시스템으로부터 클러스터 시스템으로 급선회 하고 있다[4].

범용 프로세서를 탑재한 단위 컴퓨팅 노드들을 연결망으로 묶어 하나의 시스템으로 활용하는 기술인 클러스터 컴퓨터는 가격대 성능비가 우수하다는 가장 큰 장점과 함께 계산에 적합한 시스템의 구성할 수 있는 유연성과, 컴퓨팅 노드의 교체, 추가 등에 대한 확장성이 뛰어난 장점이 있다. 특히 linux 기반의 PC를 연결한 PC 클러스터의 경우 일반적인 연구 환경 하에서 흔히 볼 수 있는 컴퓨팅 자원이 되었다.

이 시스템의 가장 큰 특징은 SMP(Symmetric MultiProcessor) 시스템과 같은 공유 메모리 구조가 아니라, 각각의 계산노드가 자신의 메모리에 각각 다른 정보들(예를 들어, 격자 정보 등)을 가지며, 다른 노드에 존재하는 정보에 대한 접근은 MPI 등의 범용 라이브러리들이 지원하는 통신(Message Passing)을 통해서만 이루어질 수 있다는 점이다. 유동 해석자의 경우 전처리기 등을 통하여 이루어진 정적 부하 분산(static load-balancing)에 따라 각각의 노드에서 유동장 계산을 따로 수행하고, 정해진 경계에서만 통신을 통한 갱신이 이루어진다. 따라서 증가한 노드 수에 따라 증가한 병렬 경계에서의 정의된 통신만이 늘어나게 된다. 그러나 격자 구성 모듈의 경우는 격자와 관련된 정보들의 통신이 대부분을 차지하며, 빈번하고 복잡하게 얽힌 통신이 이루어지게 된다. 이러한 이유로 과도한 지연과 통신시간을 소비하게 되는 경우가 발생한다. 따라서 동기화에 따른 부하가 계산속도에 비해 크기 때문에 되도록 통신의 횟수를 줄이는 Coarse-grained 병렬성을 지향하고, CPU의 성능과 통신 성능에 대한 적절한 분석을 통해 적합한 알고리즘의 개발이 필요하다. 본 연구는 일반적인 PC 클러스터의 특징을 고려하여, 가능한 한 적은 메모리의 사용과 과도한 통신을 줄이는 알고리즘을 개발하였다.

2.1 정렬 Chimera 격자 기법의 병렬처리

Chimera 기법의 병렬화는 1990년도 초부터 널리 알려진 유동해석코드들의 병렬화에 발맞춰 개발되어져 왔다. 가장 쉬운 방법으로 병렬 프로세서를 FE(Front-End)와 BE(Back-End)로 구분한 후, Chimera 격자계 구성은 FE의 단일 프로세서를 이용하고 해석자는 BE의 다중 프로세서를 이용하는 방법이 있다[5]. 이러한 방법은 직접적인 병렬화라고 할 수는 없으며, 추가적인 전송 시간이 필요하게 된다. 그러나, 기존의 코드를 그대로 적용할 수 있는 방편으로 흔히 이용되고 있다. 격자 구성 모듈의 부하 분산은 최근 들어 Prewitt 등[6]에 의해 이루어졌으며, 비정상 격자 구성 모듈을 위해 해석자와 별개의 메모리 구조를 가지는 동적 부하 분산(dynamic load balancing)을 도입하였다. 그러나 이와 같은 방법은 자원이나 작업 정보를 업데이트 하는데 쓰이는 오버헤드가 과도할 수 있으며, 많은 메모리의 필요성이 발생한다. 본 연구에서는 앞서 말한바와 같이 상당부분의 계산시간을 차지하는 유동해석자의 부하 분산을 따르는 Wissink와 Meakin의 접근 방법을 기초로 한다[7].

격자 생성 모듈은 병렬화 된 유동해석자 KFLOW[8]를 기준으로 병렬화 되었다. 그림 1은 다중블록을 고려하여 모든 노드에 블록들의 순서를 ipblk라는 global-block으로 재 정의하는 과정으로, 격자 생성 모듈의 모든 과정들은 이러한 global-block을 기본으로 이루어진다. 각 ipblk들은 자신이 속한 body 또는 고체 경계를 뜻하는 idblk라는 변수를 가지게 된다.

2.2.1 Hole cutting

크게 두 부분으로 나누어지는 Chimera격자 모듈에서 첫 번째는 고체 경계내부에 속해 있는 계산점들을 계산에서 제외시키기 위한 홀 절단 과정이다. 고체 경계 내부에 존재하는 점을 파악하기 위한 방법으로 다양한 알고리즘이 제시되어 있으나, 대부분의 정렬 Chimera 격자들이 사용하는 Hole-Map 알고리즘을 선택하였다[2]. 이는 고체 경계를 포함하는 Cartesian 격자의 생성을 통해 단순히 값의 산술적 비교만으로 홀점을 판별하는 매우 빠르고 강건한 방법이며, 벡터화가 가능한 장점으로 인해 널리 사용되어져 왔다.

그러나 이 방법의 가장 큰 단점은 모든 고체경계가 닫힌(closed) 경계를 구성해야만 안전성을 보장할 수 있다는 것이다. 따라서 하나의 고체경계(body)를 구성하는 모든 블록들의 facet(홀 절단 경계 요소)들을 함께 모아야만 한다. 또한 각 body의 facet 요소들의 최대/최소값을 판별하는 과정이 필요하다. 이를 위해 각 body를 구성하는 요소들을 위한 communicator가 새로이 정의된다. 그림 2는 기존의 5개의 노드에서 각각 계산되어지던 정보들을 icolor(idblk) 에 따라 구성된 두개의 새로운 Newcomm과 New Rank들을 보여준다. 새로운 communicator는 MPI_COMM_SPLIT 라이브러리로 쉽게 생성시킬 수 있다. facet들의 병합은 MPI_ALLGATHERV(그림 3) 라이브러리를 사용한다. 이를 통해 다른 크기의 배열구조를 하나의 연속된 배열구조에 손쉽게 병합할 수 있다.

홀 절단을 위해 각 body에서 구성된 Hole-Map은 다른 body를 구성하는 노드들로 전송되어져야 한다. 그러나 Cartesian 격자 정보인 Hole-Map의 직접적인 전송은 상당한 통신시간을 요구하게 된다. 이를 보완하기 위해 각 body에서 생성된 격자 정보의 최대/최소값과 facet들 만을 모든 노드들이 공유하도록

한다. 실제로 Chimera 격자계에서의 facet들은 고체 경계를 구성하는 부분으로, 전체 격자계의 아주 작은 부분을 차지하며 이들은 전송은 비교적 적은 통신시간을 소비한다. 또한 facet들을 이용하여 이후 탐색과정에서 고체 경계에서의 처리에 이용할 수도 있다. 이를 위해 각 facet point들의 global block과 index등을 정의한 배열을 설정하고 마찬가지로 모든 노드들이 공유하도록 한다.

이후 정리된 facet 정보들로부터 각각의 노드들은 Hole-Map을 구성하고, 홀 절단을 하게 된다. 이로부터 생성된 삽간점의 정보들은 먼저 초기의 순서대로 itag라는 index를 정의하고, 최초 정의된 기부자점의 후보block이 있는 node로 전송된다. itag는 병렬처리 과정에서 삽간점과 기부자점간의 연관성을 이어주는 최소한의 index가 된다. 따라서 전송되는 정보들은 itag, Take RANK(삽간점이 있는 RANK), INTPs(x,y,z)로 이루어진다.

2.2.2 Searching of Donor points

각각의 node들은 전송받은 삽간점(x,y,z) 정보를 이용하여, 자신의 격자계에서 기부자점 탐색을 수행한다. 기부자점의 탐색은 가장 근접한 점을 찾는 stencil-walk과 삽간점들로 둘러싸여 있는지를 판별하는 구배찾기로 이루어진다. 이 과정 중 병렬경계를 만나는 경우 그림 4와 같이 다음 block으로 넘어가도록 한다. 또한 고체경계를 만나는 경우 그림 5와 같이 facet정보들을 이용하여 그 고체경계에서 가장 근접한 점을 찾아가게 된다. 자신의 node에 존재하는 block이 아닌 다른 node의 block으로 넘어갈 경우, 이러한 점들은 모든 node들의 탐색이 끝난 후, 다시 정리되어 다른 node들로 전송되어지며 다음 순차의 탐색으로 이어지게 된다. tri-linear 삽간을 위해 3차원의 경우 8개의 point로 구성되는 삽간점 중 hole점이 4개 이상일 경우, 또는 자신의 block에서 마땅한 기부자점을 찾지 못하였을 경우, 입력조건으로 주어지는 다음 body로 탐색을 수행하게 된다. 이 때, 다른 body를 구성하는 block중 가장 빠른 ipblk 번호를 가진 block이 새로운 후보 block이 된다.

2.2.3 삽간점과 기부자점의 연결

병렬처리 과정에서 삽간점과 기부자점의 정보(i,j,k index와 block)는 각각의 node들만이 알고 있으면 되기 때문에, 별개의 두 정보들을 연결하는 과정이 필요하다. 각각의 정보들은 상대방의 RANK에 따라 그룹을 형성하게 되고, 이를 itag에 따라 정리함으로 두 정보가 정확히 일치하게 된다. 그림 6은 1번 node의 삽간점과 2번 node의 기부자점의 연결 과정을 나타내고 있다. 정보들의 정렬에는 알려져 있는 알고리즘 중 Heap-sort 알고리즘을 사용하였다. 이는 $O(nlogn)$의 적은 연산을 하며, 최악의 조건(worst case)에도 좋은 성능을 보인다. 또한 추가적인 배열이 필요치 않는 장점이 있다.

3. 결과 및 분석

3.1 격자 생성 모듈의 성능 분석

2장에서 설명한 과정들은 그림 7의 도표로 정리되어 있으며, 병렬화 알고리즘의 효율을 측정하기 위해 3개의 구를 포함한 총 4개의 body로 구성된 격자계의 Chimera 격자 구성을 수행하였다.

구성된 격자계는 그림 8과 같이 후방 격자계(113^3), 각각의 구 격자계($91 \times 65 \times 81$)로 총 288만개 가량의 노드수를 가지며, 약 64,000개의 삽간점을 가진다. 사용된 linux PC 클러스터는 각각 P-4 2.6 GHz의 CPU를 장착하고, 10/100 Mbps ethernet LAN으로 연결되어 있다.

표 1은 각 과정의 계산 시간과 통신 시간을 비교한 것으로 총 계산시간(실시간)은 wall time으로 나타내고 그 중 통신 시간은 communication time으로 따로 표시하였다. 100번의 반복계산을 통해 평균값을 나타내었다. 참고로 기부자점의 탐색과정에서의 통신시간은 정렬시간을 포함한 시간이다. node수가 증가 할수록 실제 계산시간(CPU time)은 감소하지만, 과도한 통신시간이 생겨남을 알 수 있다. 이에 따른 총 소모시간은 증가하는 추세를 보인다. 특히 홀 절단 과정에서 생성된 Hole-map의 전송시 상당한 시간을 소비하게 된다. 표 2는 이를 개선하여 facet과 min/max 값만을 MPI_ALLGATERV를 이용하여 취합하고 각각의 node가 별개로 Hole-Map을 구성한 결과이다. CPU time은 다소 증가하지만, 통신시간의 감소로 전체 계산시간은 감소하고 있다. 이러한 특징은 전송 시 소요되는 지연시간(latency)을 포함한 통신성능이 CPU의 계산 성능보다 훨씬 떨어지기 때문에 발생하는 현상이며, 보통의 Ethernet network으로 연결된 PC 클러스터의 특징이라고 할 수 있다. 최근의 값비싼 network 장비들은 지연시간이 다소 줄어들고 있으나, 여전히 데이터 전송보다는 직접적인 계산과정이 이득을 보이고 있다. 그림 9는 표 2의 결과를 그래프로 나타낸 것이다. 총 24프로세서를 사용한 경우 CPU time은 80%의 감소, wall time은 20% 정도 감소한다.

3.2 정상상태 비점성 유동 계산

제안된 알고리즘을 이용하여 구성된 Chimera 격자 모듈은 유동해석자와 결합되어 2차원 두개의 NACA0012 날개단면과 과 3차원 ONERA M6 날개와 타원체 스토어의 계산으로 검증되어졌다.

유동해석자에 사용된 기법들을 간략히 설명하면 다음과 같다. 요소중심 유한 체적법(cell-centered finite volume method)을 사용하여 제어체적에 대해 이산화 하고, 1차의 공간 정확도를 갖는 Roe의 FDS(Flux Difference Splitting)방법에 MUSCL(Monotone Upwind Scheme for Conservation Law)기법을

적용하여 경계면에서 고차의 이산화 오차를 갖는 수치유량을 구성하였다. 시간 전진 기법으로는 Diagonalized ADI 기법을 사용하였다.

2차원 날개단면 경우는 $M_\infty = 0.55, \alpha = 0$의 자유류 조건을 가지며, Chimera 격자의 정확도를 판별하기 위해 5개의 multi-block 으로 구성된 격자를 이용한 결과와 비교하였다. 사용된 격자계는 그림 10과 같다. 그림 11은 각 경우의 수렴성과 표면압력 분포를 보여주는 것으로 Chimera 격자(serial and parallel)의 결과와 일반적인 multi-block 결과가 잘 일치함을 보여주고 있다.

3차원 계산에는 그림 12와 같은 격자가 사용되었다. 날개에 대해 C-H-유형의 $129 \times 33 \times 33$ 격자점이, 스토어에 대해 O-유형의 $65 \times 25 \times 33$ 격자점이 사용되었다. 자유류 조건은 $M_\infty = 0.84, \alpha = 3.06°$로 주어진다. Serial 계산과 12 프로세서를 이용한 계산결과가 그림 13과 같이 참고문헌[2]의 계산결과와 비교되었다. Store의 영향이 크게 나타나는 span 방향으로 44%와 65%에서 날개표면의 압력분포를 나타내었다.

병렬처리를 통한 결과는 serial 계산의 결과와 아주 잘 일치하고 있고, 참고문헌의 결과와는 전반적으로 비슷한 결과를 보여주고 있다. 날개 윗면의 앞전부근에서 약간의 차이를 보이고 있으나, 이와 같은 차이는 유동 해석자에서 사용된 수치기법의 차이로 볼 수 있으며, 그림 14에서 ONERA M6 날개만을 해석한 결과를 실험치와 함께 비교해 보면, 현재의 유동해석자가 조금 더 실험치와 비슷한 결과를 보여줌을 알 수 있다.

4. 결 론

일반적인 PC 클러스터에 적합한 정렬격자계를 이용한 Chimera 격자 구성 모듈의 병렬화를 수행하였다. 병렬화를 통하여 CPU time을 확실히 줄일 수 있으며, 홀 절단 부분과 기부자점의 탐색과정에서 최적화 된 Coarse-grained 통신으로 프로세서의 증가에 따른 부가적인 통신시간을 최소화 하였다. 이러한 병렬처리는 일반적인 병렬 유동해석자의 정적 부하 분산을 그대로 이용하여 추가적인 선처리기(preprocessor)가 필요하지 않으며, 불필요한 저장 공간 사용을 막을 수 있다.

개발된 알고리즘은 많은 격자를 가지는 비정상 상대운동 해석에 사용될 수 있을 것이다.

참고문헌

[1] L.E. Lijewski and N.E. Suhs, 1994, "Time-Accurate Computational Fluid Dynamics Approach to Transonic Store Separation Trajectory Prediction," Journal of Aircraft, Vol.31, No.4, July-Aug.

[2] 조금원, 2000, "상대운동이 있는 3차원 비정상 유동해석을 위한 효율적인 중첩 격자계 개발,", *박사학위논문*, 한국과학기술원.

[3] J.L. Steger and F.C. Dougherty and J.A. Benek, 1985, "A Chimera Grid Scheme," In Advances in Grid Generation, p.59-69, *ASME* FED-Vol.5, NY.

[4] www.top500.org

[5] N.C. Prewitt, D.M. Belk and W. Shyy, 1998, "Parallel Grid Assembly Within the Beggar Code," *The 4th Symposium on Overset Composite Grid and Solution Technology.*

[6] N.C. Prewitt, D.M. Belk and W. Shyy, 2002, "Improvements in Parallel Chimera Grid Assembly," *AIAA Journal*, Vol.40, No.3, p.497-500.

[7] A.M. Wissink and R.L. Meakin, 1997, "On Parallel Implementations of Dynamic Overset Grid Methods." *SC97: High Performance Networking and Computing.*

[8] Y. Kim, S.H. Park and J.H. Kwon, 2004, "Drag Prediction of DLR-F6 Using the Turbulent Navier-Stokes Calculations with Multigrid," *AIAA Paper 2004-0397*, 42nd Aerospace Science Meeting and Exhibit, Reno, NV.

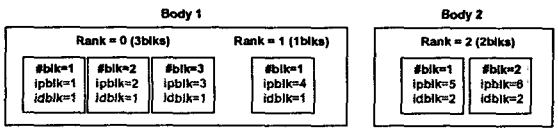


Fig. 1 Global block numbering and body indexing

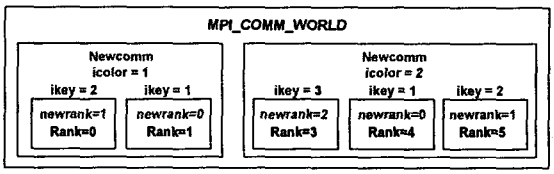


Fig. 2 Define new communicator (MPI_COMM_SPLIT)

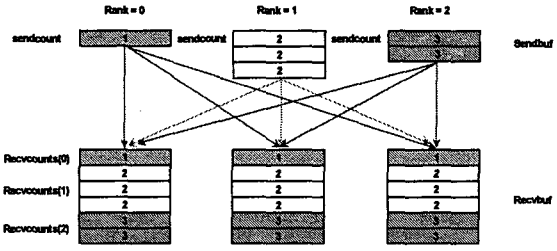


Fig. 3 MPI ALLGATHERV

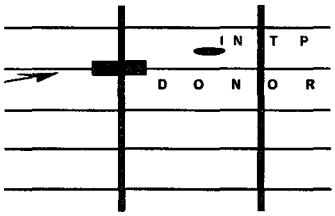


Fig. 4 Jumping through contiguous boundary

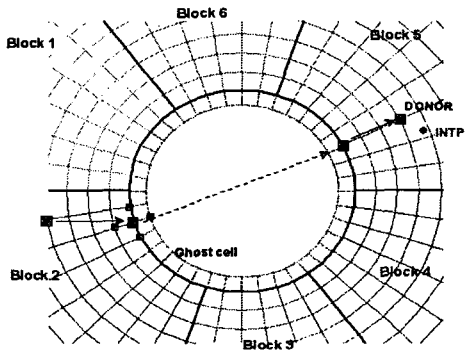


Fig. 5 Jumping to nearest points on solid boundary

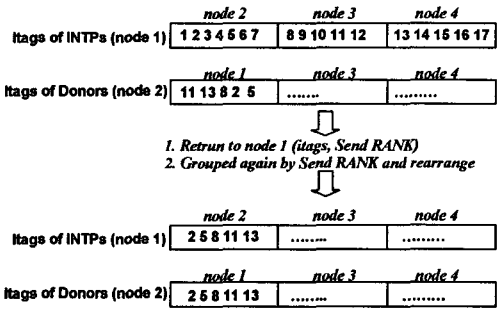


Fig. 6 Coupling of INTS and Donors

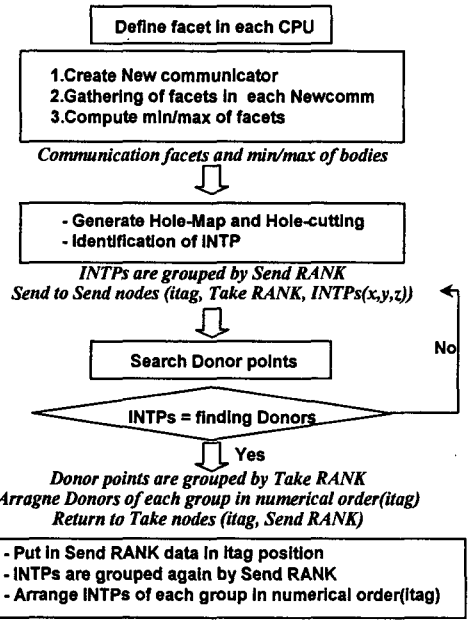


Fig. 7 Algorithm for parallel grid assembly

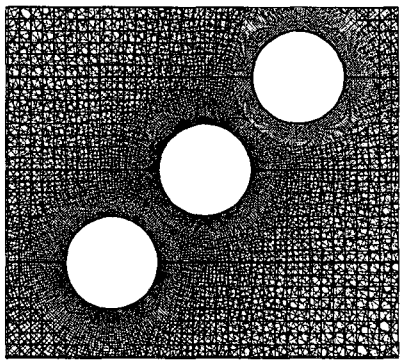


Fig. 8 Chimera grid in symmetric section

Table. 1 Performance of Chimera assembly : average time
(Hole-map communication)

Node	Total time		Hole cutting		Donor-Searching	
	Wall time	CPU time	Wall time	Comm.time	Wall time	Comm.time
1	4.352	4.331	1.271	-	2.528	-
4	5.829	1.801	3.516	2.883	1.900	0.202
7	6.883	0.921	4.077	4.053	1.959	0.379
14	7.072	0.867	5.033	4.476	1.866	0.449
24	7.229	0.661	5.669	5.059	1.832	0.594

Talbe. 2 Performance of Chimera assembly : average time (Hole-map computing)

Node	Total time		Hole cutting		Donor-Searching	
	Wall time	CPU time	Wall time	Comm.time	Wall time	Comm.time
1	4.372	4.367	1.247	-	2.551	-
4	3.624	2.043	1.244	0.442	1.948	0.203
7	3.594	1.125	1.360	0.627	1.980	0.374
14	3.327	1.037	1.266	0.896	1.866	0.443
24	3.175	0.889	1.151	0.769	1.833	0.595

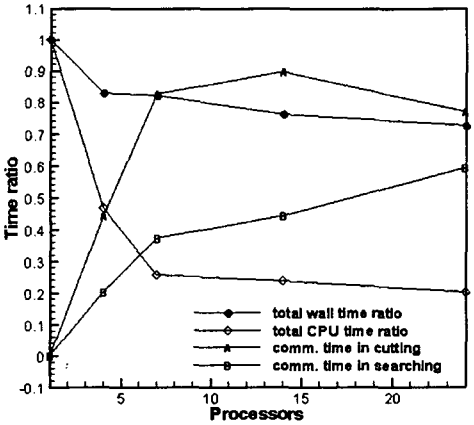


Fig. 9 Performance of Chimera assembly

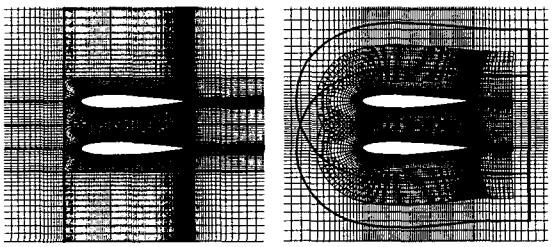


Fig. 10 Two NACA 0012 airfoil grid (multi block and Chimera)

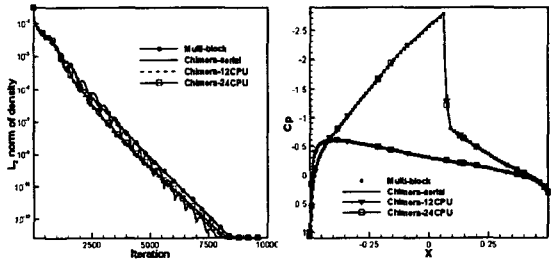


Fig. 11 Convergence History and Pressure Coefficient Distribution (Two NACA0012 airfoil)

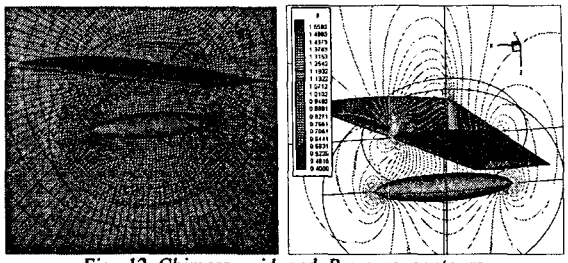
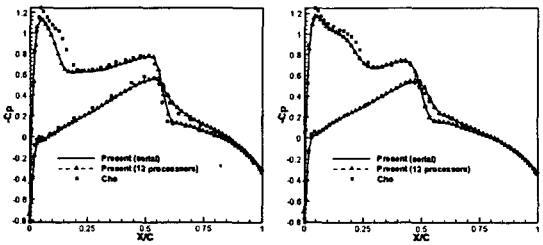
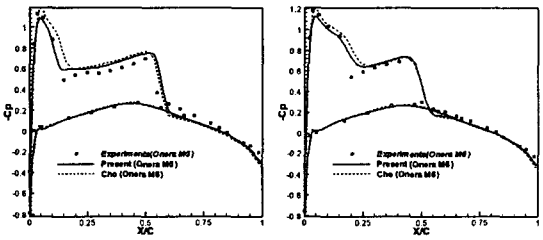


Fig. 12 Chimera grid and Pressure contours (ONERA M6 wing + store)



(a) Span station : 44% (b) Span station : 65%

Fig. 13 Pressure coefficient distributions, Onera M6 + Store



(a) Span station : 44% (b) Span station : 65%

Fig. 14 Pressure coefficient distributions, Onera M6 wing