그리드 환경하의 효율적 해석을 위한 작업 분할 기법 연구

고 순 흠*[1], 정 명 우[1], 김 종 암[2], 노 오 현[2], 이 상 산[3]

Load Balancing for the Efficient Parallelization in the Grid

Soon-Heum Ko, Myungwoo Jung, Chongam Kim, Oh-Hyun Rho, and Sangsan Lee

The Grid[1] is a communication service that collaborates dispersed high performance computers so that those can be shared and worked together. So, the Grid enables a researcher to analyze a huge-sized problem which was impossible by using local resources. However, diverse communication speeds among computing resources and heterogeneity of computing resources can reduce parallel efficiency in the Grid. The present paper focuses on the development of an efficient load balancing algorithm suitable for the Grid. Proposed algorithm classifies the whole processors into several groups with relatively faster communication speeds. Computational domain is firstly partitioned to each group and then to the processor level considering the performance of each processor. Developed algorithm is validated in the homogeneous system by comparing the present result with the result of equally partitioned meshes and then applied to the heterogeneous system. Additionally, the present algorithm is expanded to be able to solve the decomposed domains and applied to some problems.

Key Words: 그리드(Grid), 병렬화(Parallelization), 작업 분배(Load Balancing)

1. Introduction

무어의 법칙에 따르면[2], 프로세서의 속도는 매 18개월마다 두 배씩 향상된다. 이와 같은 계산 자원의 급격한 발전에 힘입어 전산 유체 역학의 효용 가치는 날로 증대되어 왔다. 그럼에도 불구하고, 아직도 단일 기관의 계산 자원만으로는 해결할 수 없는 거대 규모의 문제들이 존재하는 것이 현실이다. 이와 같은 계산 자원의 한계를 극복하기 위하여 다 기관의 고성능 계산 자원들을 연동하는 'Grid'의 개념이 제안되었으며, 현재 많은 연구가 진행중이다.

Grid의 활용을 통하여 기존에 수행할 수 없었던 많은 문제들에 대한 해석이 가능해짐은 주지의 사실이다. 부가하여 다 기관의 고성능 자원들을 활용하므로 실제 해석 시간의 감소 효과도 기대할 수 있다. 그러나, 해석 시간의 감소가 병렬 효율의 증가에 기인하는 것은 결코 아니며, 오히려 해석에 사용되는 기기들의 이종적 성능과 WAN(Wide Area Network)을 통한 기기간 통신 수행으로 인하여 실제 병렬 효율은 감소하게 된다. 따라서 Grid 환경을 이용한 해석 수행시 이와 같은 병렬 효율의 감소를 최소화할 수 있는 방안 마련이 필수적이라 할 것이다.

본 연구에서는 Grid 환경에서의 효율적 유동 해석 방안을 제시하고자 한다. 병렬 효율 증진을 위하여 Grid 환경에 적합한 작업 분배 알고리즘을 개발하여 실제 계산 환경에 적용하였다. 제안된 알고리즘은 Grid상의 계산 자원의 이종적 성능을 최적으로 활용하고, Grid상에서 발생할 수 있는 자원간 통신 시간의 급격한 증가를 막을 수 있도록 설계되었다. 그리

1 학생회원, 서울대학교 기계항공공학부 대학원
(151-742 서울시 관악구 신림동 Tel : 02-880-1903)
2 종신회원, 서울대학교 기계항공공학부
3 정회원, KISTI 슈퍼컴퓨팅 센터
* E-mail : floydfan@hanmail.net

고 개발된 알고리즘은 다양한 실제 문제에 적용하여 그 타당성을 입증하였다.

2. Governing Equations and Numerical Techniques

지배방정식으로서 3차원 압축성 Euler 방정식이 사용되었다. 공간차분법은 AUSMPW+(modified Advection Upstream Splitting Method Pressure-based Weight function) 이 적용되었다. 시간 적분은 LU-SGS(Lower-Upper Symmetric Gauss Seidel) 기법이 사용되었다. 격자계는 단일 블록으로 구성된 tangent ogive-cylinder 형상 및 다중 블록을 사용하는 boattail 형상을 구현하여 다양한 격자 기법에서의 적용 가능성을 확인하였다.

3. Load Balance Algorithm for the Grid

3.1 Background

병렬 클러스터의 발전과 함께 효율적 병렬 해석을 위한 많은 연구가 수행되어 왔다.[3,4] 그러나 그간 제시된 많은 병렬 효율 증대 기법들을 살펴보면 계산 자원의 성능의 이종성만을 고려하는 작업 할당을 수행하거나 또는 통신 시간의 최적화를 고려한 경우에도 통신량의 축소 방안이나 혹은 계산 자원의 하드웨어적 특성을 적절히 활용하여 통신 시간을 감소하는 데에만 치중하여 왔다. 그러나 Grid망에서는 광역의 자원들을 연동하므로 계산 자원간 통신 속도가 매우 다양하여 통신량의 축소 뿐 아니라 통신 성능이 우수한 기기간에 통신이 수행되도록 하는 적절한 영역 할당이 부가적으로 고려되어야 한다. 따라서 Grid망에서의 효율적 수치 해석을 위하여는 작업량의 균형 및 적절한 작업 배치가 동시에 수행되는 작업 할당 알고리즘의 연구를 수행하여야 한다.

3.2 Development of Algorithm

Grid 환경에서의 Load Balancing을 위하여는 크게 다음의 두 가지 조건이 만족되어야 할 것이다. 우선 WAN상에서 연동되는 기기간 통신은 가능한 최소화되어야 한다. 그리고 전체 작업량은 각 자원의 성능을 고려하여 최적화되도록 분배되어야 한다.

이와 같은 특성을 고려하여 간략한 작업 분할 알고리즘을 제안한다. 알고리즘의 제작을 위하여 다음의 사항들을 가정한다.

1. 서로 다른 병렬 기기의 프로세서간 통신 속도는 동일 기기 내의 자원간 통신 속도보다 느리다.

2. 단일 클러스터 내의 프로세서들은 동일 성능을 가진다.

3. 각 자원에 할당되는 계산 영역은 직사각형(3차원의 경우 직육면체)의 형상이 되도록 한다.

그러므로 동일 클러스터 내의 프로세서간 통신 가능성을 높이기 위하여 각 클러스터를 하나의 그룹이 되도록 영역을 분할한 후 클러스터 내의 프로세서들에 작업을 할당하는 방식을 취한다. 이와 같은 전제로부터 개발된 작업 분할 알고리즘은 다음의 과정을 통하여 구현된다.

3.2.1 각 processor의 성능 테스트

연동하는 자원별 성능을 알아보기 위하여 동일한 격자를 각 processor에 할당한 후 유동 해석자에 대한 몇 회의 반복 계산을 수행하는 동안 각 processor의 해석 시간을 구한다. 그러면 각 processor의 계산 성능은 계산 시간의 역수로 나타난다.

3.2.2 Processor간 통신 테스트

각 processor는 연동되는 다른 모든 processor들과 통신을 수행한다. 그리고 상대적으로 빠른 통신 속도를 가지는 processor들을 단일 group으로 가정하여 이들 processor간 경계 영역의 공유 확률을 높인다.

광역에 분포하는 여러 클러스터간 연동의 경우 각 group은 단일 클러스터와 같게 되며, 동일 LAN(Local Area Network)상의 여러 듀얼 시스템간 연동의 경우 동일 기기 내의 processor간 통신 속도가 훨씬 우수하다는 점에서 각 듀얼 시스템이 하나의 group으로 구성된다.

3.2.3 각 group으로의 계산 영역 할당

각 group의 계산 능력을 고려하여 전체 계산 영역을 각 group으로 할당한다. 우선 각 group의 계산 성능은 group을 구성하는 processor들의 계산 성능의 합으로 나타난다. 이와 같이 구해지는 group간 계산 성능 비로부터 전체 계산 영역을 분할하여 각 group에 할당하게 된다. 여기서 할당되는 계산 영역은 직사각형(3차원의 경우 직육면체) 형상으로 나타나게 된다. 그리고 영역 할당 방식은 전체 격자를 둘로 분할하는 과정의 반복을 통하여 궁극적으로 모든 group이 자신의 계산 영역을 할당받도록 한다.

3.2.4 각 processor로의 계산 영역 할당

각 group의 계산 영역을 processor들에 할당하는 방식은 기본적으로 전체 영역을 각 group에 할당하는 방식과 동일하다. Group에 부여된 계산 영역을 두 영역으로 분할하는 방식을 반복하여 궁극적으로 각 processor에 계산 영역을 할당하게 된다. 그리고 계산 영역의 할당이 완료되면 각 processor는 자신에 할당된 계산 영역의 상대 위치를 고려하여 자신의 경계 조건을 구하게 된다.

3.2.5 유동 해석

이와 같은 영역 분할 과정이 종료되면 각 processor는 병렬 유동 해석을 수행한다.

3.3 Expansion to Decomposed Domains

개발된 알고리즘은 단일 블록을 사용하는 전산유체역학 문제로의 적용 가능성을 염두에 두고 개발되었다. 그러나 Grid 환경에서 해석되는 실제 문제는 복잡한 형상을 가지는 방대한 규모의 문제들이다. 그리고 이와 같은 문제들은 다중 블록이나 겹침 격자 기법과 같은 영역 분할 기법에 의하여 구성되는 격자를 사용하게 된다. 따라서 제안된 작업 할당 기법의 확장을 통하여 다중 블록이나 겹침 격자 기법의 효율적 해석이 가능하도록 하였다.

이와 같은 격자 기법들에 적용하기 위하여 제안된 알고리즘에 각 격자 영역별 자원 할당 과정을 첨가한다. 각 group으로의 계산 영역 할당 과정 전에 수행되는 각 영역별 자원 할당 과정은 다음과 같다.

우선 각 격자 영역의 격자 크기로부터 총 계산 영역의 규모 및 전체 계산량에 대한 각 계산 영역의 계산량의 비를 알 수 있다. 이와 같은 계산량의 비와 가장 근접하도록 group을 배치한다. 이 경우 최적화된 작업 할당을 위하여 단일 group을 분할하여 서로 다른 계산 영역상에 배치하는 과정이 첨가될 수 있다. 그러나, grouping의 목적은 통신 성능이 우수한 processor간 통신 확률을 높이는 데 있으므로 최소의 group의 분할만으로 적절한 작업 균형을 이루도록 한다.

이와 같이 제안되는 작업 할당 기법은 실제 유동 해석자의 반복 계산 후 작업 분할을 수행한다는 점에서 각 사용자의 해석자에 최적의 작업 할당을 해준다는 장점을 가진다. 또한, 단순한 계산 수행을 통하여 작업 분배가 수행된다는 점에서 쉬운 적용 가능성을 가지고 있으며, 영역 분할 기법을 사용하는 격자계로의 적용이 간편하다는 장점을 가진다.

3.4 Computing Resources

사용 자원은 현재 서울대학교 공력시뮬레이션 연구실에 비치된 4 node dual cluster와 다른 PC 4대를 연동하였으며, 다양한 계산 자원 조합에서의 병렬 효율을 분석하였다. 각 계산 자원의 성능은 표 1에 나타난 바와 같다. SPM cluster상의 RAM은 각 node당 크기이며, SPM cluster는 총 8개의 프로세서를 가지고 있다. PE 1 및 2라 이름붙여진 fastcfd2 기기의 경우 역시 dual processor를 사용하고 있다.

표 1 연동 자원 구성

	CPU(s)	RAM	Network
SPM Cluster	P-Ⅲ 933 MHz	512MB	100MBPS fast Ethernet
PE 1, 2 (fastcfd2)	P-Ⅲ 1.4 GHz	1GB	100MBPS fast Ethernet
PE 3 (mana)	P-Ⅲ 850 MHz	256MB	100MBPS fast Ethernet
PE 4 (fastcfd4)	P-Ⅲ 500 MHz	512MB	100MBPS fast Ethernet

4. Results

4.1 Application to Single Block

71×91×81개의 격자를 가지는 3차원 tangent ogive-cylinder 주위의 유동 해석을 수행하였다. 수렴 판정의 기준은 오차 잔류항이 4차 이하가 되는 것으로 하였다. 본 계산의 경우 본 알고리즘의 적용 시 수렴까지 총 2139번의 반복 계산이 수행되었으므로 정확한 비교를 위하여 균등 분할된 격자를 사용하는 경우에도 동일한 수의 반복 계산을 수행하도록 하였다. 그리고 효율적 결과 분석을 위하여 해석자의 계산 수행에 소요된 시간은 크게 pre-processing 과정과 유동 해석 부분으로 나누었다. 그리고, 유동 해석시의 경계면에서의 통신 시간을 알기 위하여 유동 해석을 수행한 후 경계 영역에서의 통신을 배제한 경우의 해석 시간을 알아봄으로써 유동 해석 부분에서 유동 계산 시간과 경계 통신 시간을 분류하였다.

본 알고리즘의 타당성 여부를 확인하기 위하여 성능이 유사한 SPM cluster와 mana PC를 연동하여 총 8개의 processor를 연동하였다. 총 8개의 processor를 가지는 SPM cluster 중 다른 노드에서는 dual CPUs를 모두 사용하고, 2번 노드에서는 하

나의 프로세서만을 사용하도록 하였다. 가장 성능이 낮은 기기는 mana PC이므로 각 경우의 PC의 작업량 변화가 전체적인 효율을 좌우함을 알 수 있다.

표 2 유사한 자원간 연동시의 격자 할당

	할당 격자수	Processor간 계산량 비	등분할과의 비교
Node01	36×46×41 36×46×41	1.06	1.00
Node02	38×46×41	1.12	1.06
Node03	36×46×41 36×46×41	1.06	1.00
Node04	36×46×41 36×46×41	1.06	1.00
mana	34×46×41	1	0.94

표 3에는 현재의 격자 분할시의 해석에 수행된 시간을 등분할된 격자를 사용하는 경우와 비교한 결과를 나타낸다. 우선 격자 할당 부분에 소요되는 시간은 본 알고리즘의 적용시 그 시간이 과도하게 증가하게 된다. 이는, 사전에 분할된 격자를 순수하게 읽어들이기만 하면 되는 등분할 해석시와 달리 각 프로세서의 성능과 프로세서간 통신 시간을 계산하고, 이 결과로부터 그룹화와 영역 분할을 수행하게 되는 본 알고리즘의 수행 과정에 훨씬 많은 계산량이 필요하기 때문이다.

그러나 격자 할당에 필요한 과도한 시간 소모는 순수 계산부에서의 효율성 증대로 인하여 만회된다. 유동 해석부에서 제안된 알고리즘의 사용에 의한 해석 시간 감소는 119초로 나타났으며, 특히 순수 계산에 소요되는 시간은 총 145초의 감소 효과로 나타난다. 이는 표 2에서 제시된 영역 분할 결과와 일맥상통한다. 표 2의 결과에 따르면 가장 성능이 낮은 프로세서는 PC 3이다. 이 결과가 타당하다면 등분할시의 유동 해석에 소요되는 시간은 PC의 성능에 영향을 받을 것이다. 따라서 PC에 할당되는 계산량의 변화만큼 병렬 효율의 증대가 나타나게 된다. 실제로 계산량의 변화를 살펴보면 본 알고리즘의 사용시 5.6%의 계산량 감소가 나타났다. 이 때 계산 시간은 5.6%가 감소되는 것으로 나타나서 본 알고리즘이 매우 정확함을 생각할 수 있다.

그러나 통신 부분에서는 본 알고리즘의 적용에 의하여 총 3.3%의 계산 시간 증가가 나타난다. 이는 현재 최대의 성능을 가지는 것으로 나타난 node 02번의 프로세서에 할당되는 작업량에 기인한다. 이와 같이 특정 프로세서에 할당되는 작업량은 증가하게 되어 이러한 자원들의 경우 경계 영역이 증가하게 되며 이로 인한 통신 시간의 증가가 수반된다. 이는 실제로 개선 불가능한 사항이다.

표 3 유사한 자원간 연동시의 해석 시간

	알고리즘 적용	격자 등분할
총 계산 시간	4331.068	4417.141
격자 할당	37.760	4.505
유동 해석	4293.308	4412.636
(유동 계산)	(3472.852)	(3618.373)
(자원간 통신)	(820.455)	(794.262)

다음으로 fastcfd2, fastcfd4 및 mana의 서로 다른 3대의 PC들을 연동하는 경우의 격자 분할을 나타낸다. 본 문제는 앞의 성능 분석 결과와 달리 연동 자원간 성능 차이가 크게 나타난다. 이로 인해 계산 자원간 계산량의 비를 살펴보면 최저 성능을 나타내는 fastcfd4에 비하여 fastcfd2의 경우 2.62배 많은 계산량을 부여받게 된다. 그리고, 본 알고리즘의 효용성 검증을 위하여 비교 문제로 사용하는 등분배된 계산 격자와 비교할 때 최저 성능의 기기에 할당되는 작업량이 0.52배로 감소하게 된다. 등분배시 병렬 효율은 최저 성능을 가진 processor에 좌우된다는 사실을 생각할 때 본 알고리즘이 최적화된 격자 분할을 이루었다면 결국 본 알고리즘의 적용에 의한 해석 결과는 이상적으로 등분배시에 의한 해석에 비하여 작업량의 감소 비율만큼 해석 시간이 감소할 것이다.

표 4 PC간 연동시의 격자 할당

	할당 격자수	Processor간 계산량 비	등분할과의 비교
fastcfd2	71×41×62	2.62	1.35
	71×40×62	2.55	1.31
mana	71×80×20	1.65	0.85
fastcfd4	71×12×81	1	0.52

표 5에는 본 알고리즘의 적용에 의한 해석 시간과 등분배된 격자의 사용시의 해석 시간에 대한 비교를 나타내었다. 본 결과를 살펴보면 현재의 알고리즘에 의한 계산 효율 증대가 초기의 기대에는 미치지 못함을 알 수 있다. 우선 processor의 성능에 좌우되는 순수 계산 시간을 살펴보면 순수 계산 시간은 등분할에 의한 결과와 비교하여 0.52배의 계산 시간을 나타내어야 하나, 실제 계산 결과는 0.59배가 되어 기

대한 만큼에는 미치지 않는 것을 알 수 있다. 이의 원인은 크게 두 가지가 있다. 우선 유동 계산의 매 iteration당 수행하는 오차 잔류항의 계산시 각 processor에서 계산의 결과로 나타나는 오차를 단일 processor로 전달하고 수렴 판정을 하는 부분은 본 알고리즘의 적용을 통하여 효율 향상을 기대할 수 없다. 실제 각 processor별로 오차를 수합하고 수렴 판정을 수행하는 부분의 경우 균등 분할시와 본 알고리즘의 적용시 동일한 시간을 요구한다. 그리고 메모리 사용량의 증가 또한 현재의 결과의 원인이 될 것이다. 성능이 우수한 processor의 경우 계산량의 증가에 따라 유동 해석시 사용하는 메모리의 양이 증가할 것이다. 이 경우 메모리로부터 저장된 변수의 값을 받아들이는 시간이 증가하게 되어 계산 자체의 성능이 기대 성과에는 미치지 않게 될 것임을 예상할 수 있다.

통신 시간은 균등 분할시의 결과와 비교하여 약 1.83배 증가하였다. 이는 격자 분할시 각 processor에 할당되는 격자의 형태가 상대적으로 경계영역을 넓게 포함하도록 분할이 수행된 점에 기인한다. 또한, 각 processor가 상대 processor와의 통신을 위하여 대기하는 synchronization 문제도 발생하고 있다. 그러나 단일 processor에 할당되는 격자의 규모의 증가에 따라 전 해석 시간에 대한 통신 시간의 비는 줄어들게 되어 지금의 문제의 경우 통신 시간의 증가에 따른 병렬 효율의 저하는 그다지 크게 영향을 미치지는 않는다.

표 5 PC간 연동시의 해석 시간

	알고리즘 적용	격자 등분할
총 계산 시간	7531.097	10758.203
격자 할당	40.388	4.812
유동 해석	7490.709	10753.391
(유동 계산)	(6396.391)	(10155.944)
(자원간 통신)	(1094.318)	(597.447)

4.2 Application to Multiblock

영역 분할 기법을 사용하는 격자계에서의 타당성을 확인하기 위하여 jet가 분출되는 3차원 boattail 형상에 대한 해석을 수행하였다.[5] 그림 1과 같이 총 2개의 블록으로 구성되며, 각 블록은 각각 81×25×81 및 81×25×161개의 격자점을 가진다.

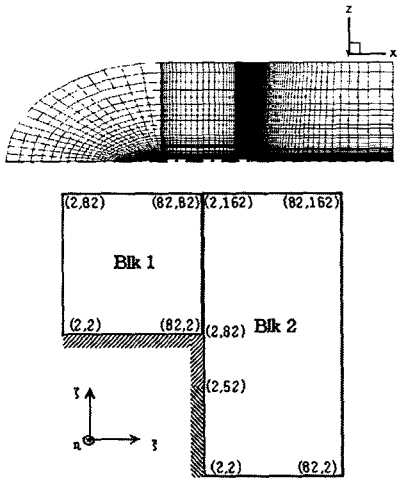

Fig 1 Multiblock Mesh of Boattail Configuration

이와 같은 문제의 해석을 위하여 SPM 클러스터와 mana 및 fastcfd4의 9개의 processor를 연동하였다. 해석 결과는 그림 2와 같이 나타난다. 팽창부의 expansion wave와 afterbody에서의 oblique shock, nozzle 후방의 barrel shock과 같은 물리적 현상에 대한 정확한 예측 결과로부터 본 알고리즘이 해석자에 올바르게 구현되었음을 확인할 수 있다.

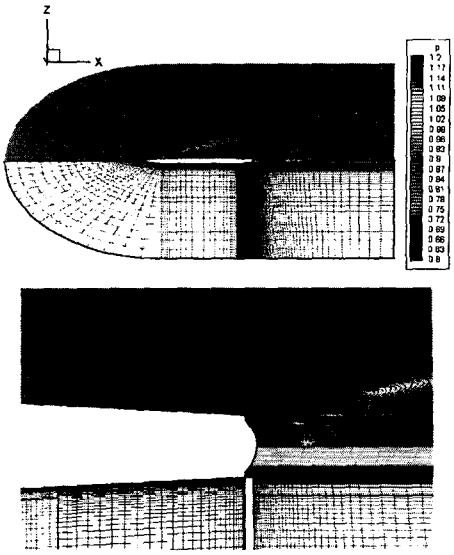

Fig 2 Pressure Contour

본 알고리즘의 적용시 실제 병렬 효율 향상을 알아보기 위하여 실제 할당 격자수를 표 6에 나타낸다. 역시 이 경우에도 최소 사양의 fastcfd4에 비하여 다

른 계산 자원들에 더욱 많은 계산량이 부과됨을 알 수 있다. 단, 이 경우 동일한 자원인 SPM 클러스터 상의 node들에 할당되는 계산량도 조금씩 차이가 남을 알 수 있다. 실제로 4번 node의 processor에 비하여 2번 node의 processor에 할당되는 계산량이 약 6% 더 많은 것을 알 수 있는데, 이는 하나의 processor는 단일 블록 내의 계산 영역을 할당받도록 구성된 현재의 영역 분할 방식에 기인한 것이다. 따라서 현재의 구성에서는 1번 블록의 계산 영역을 할당받는 processor들에 비하여 2번 블록 내에 배치된 processor들이 6% 정도 더 많은 계산 시간을 소요하게 될 것이다. 이와 같은 문제는 연동 자원의 수에 비하여 해석을 수행하는 블록의 수가 늘어날수록, 또 블록간 크기의 비율 차가 클수록 두드러지게 나타나나, 연동 processor의 수가 많아질수록, 연동 자원간 이종성이 커질수록 그 영향은 작아진다.

표 6 다중 블록 격자계에서의 영역 할당

Blk		할당 격자수	Processor간 계산량 비	등분할과의 비교
1	Node04	56×25×41 56×25×41	1.72	1.05
	mana	26×25×81	1.58	0.96
2	Node01	81×25×30	1.82	1.11
	Node02	64×25×38 64×25×37	1.83 1.78	1.11 1.08
	Node03	41×25×59 41×25×59	1.82	1.11
	fastcfd4	18×25×74	1	0.61

위의 구성을 가지는 계산 분배시 실제 해석에 소요되는 계산 시간을 표 7에 나타낸다. 해석자가 수렴하기까지 총 반복 계산은 14431회 수행되었으며, 이와 같이 반복 계산의 횟수가 늘어 전체 계산 시간이 증가함에 따라 초기 격자 분할에 소요된 계산 시간은 전체 병렬 효율에 거의 영향을 미치지 못할 정도로 작아졌다. 유동 계산에 소요되는 시간은 약 26%의 감소 효과를 나타내게 되는데, 장시간의 해석동안 순간적인 시스템의 사용률 변화(다른 사용자의 log in, 시스템 파일의 실행 등), 오차 해석시 동일 시간 소요 등의 원인으로 인하여 이상적인 해석 시간 감소에 비하여 상대적으로 많은 해석 시간을 요구한 것으로 예측할 수 있다. 통신 시간은 균등 분할시와 비교하여 약 14%의 증가를 나타내었는데, 이는 계산량의 증가에 의하여 특정 processor의 경계 영역이 증가함에 기인한 것이다. 본 알고리즘의 적용 결과 전 해석 시간 기준으로 21.1%의 시간 감소 효과를 나타내었으며, 동적 영역 분할(dynamic load balancing) 방식을 적용할 경우 더욱 우수한 효과를 나타낼 것으로 기대된다.

표 7 다중 블록 격자계에서의 해석 시간

	알고리즘 적용	격자 등분할
총 계산 시간	23083.5112	29195.6316
격자 할당	52.3775	2.3784
초기 계산	9.4535	10.8013
유동 해석	23021.6802	29182.4519
(유동 계산)	(18968.2074)	(25629.3232)
(자원간 통신)	(4053.4728)	(3553.1287)

5. 결 론

본 연구를 통하여 Grid 환경에 적합한 작업 분할 알고리즘을 개발하고 이를 실제 문제에 적용하였다. 본 연구는 Grid 환경에서 해석될 복잡한 형상의 다중 블록 문제의 효율적 해석 방안을 확립하였다는 점에 의의를 가진다. 다양한 이종적 환경에서 본 알고리즘의 적용시 상당한 효율 증가를 확인할 수 있었으며, 향후 동적 영역 분할 방식의 적용을 통하여 더욱 우수한 결과를 얻을 수 있을 것으로 기대된다.

후 기

본 연구는 한국과학기술정보연구원의 국가 Grid 사업의 지원으로 수행되었습니다. 연구 지원에 감사드립니다.

참고문헌

[1] M. M. Resch, "Metacomputing in High Performance Computing Center," IEEE 0-7659-0771-9/00, pp. 165-172 (2000)

[2] http://www.intel.com/research/silicon/mooreslaw.htm

[3] Y. P. Chien, et. al, "Load-balancing for parallel computation of fluid dynamics problems," Computer Methods in Applied Mechanics and Engineering, 120, (1995), pp.119-130

[4] J. Rantakokko, "Partitioning Strategies for Structured Multiblock Grids," Parallel Computing, Vol.26, (2000), pp.1661-1680

[5] G. S. Deiwert, "Supersonic Axisymmetric Flow over Boattails Containing a Centered Propulsive Jet," AIAA J., Vol.22-10, (1984), pp. 1358-1365