2003 춘계학술발표회 논문집
한국원자력학회

삼차원 전노심 코드 DeCART의 병렬처리 특성

Parallelization Characteristics of a Three-Dimensional Whole-Core Code DeCART

조진영, 주한규, 김하용, 이정찬, 장문희
한국원자력연구소
대전광역시 유성구 덕진동 150

요 약

3차원 전노심에 대한 중성자 수송 계산은 상당량의 계산시간 뿐만 아니라 막대한 메모리를 요구한다. 따라서 DeCART 같은 코드에서는 병렬계산 뿐만 아니라 메모리 분산이 필수적이다. 이 논문에서는 프로세스 그룹핑에 기초한 병렬처리 및 메모리 분산방식을 DeCART 코드에 구현하고, C5G7 3차원 벤치마크 문제와 단순화된 3차원 SMART 노심을 풀어 그 성능을 평가한다. 24개 CPU를 사용할 경우 C5G7 3차원 벤치마크 문제의 경우 IBM Regatta 슈퍼컴퓨터에서 MOC 커널의 최대 Speedup이 약 22로, LINUX Cluster의 경우는 약 21로 나타나 병렬 처리능이 아주 우수한 것으로 나타난다. 또한 단순화된 3차원 SMART 노심의 경우 1대의 컴퓨터를 사용할 경우 약 11 GBytes의 메모리가 요구되나 메모리 분산을 할 경우 약 940 MByte가 소요되어 매우 큰 문제도 저가의 LINUX Cluster를 통해 계산할 수 있게 되었다.

Abstract

Neutron transport calculation for three-dimensional core problems requires not only a tremendous amount of computing time but also huge memory. Therefore, whole-core codes such as DeCART need both parallel computation and distributed memory capabilities. This paper is to implement such parallel capabilities based on MPI grouping and memory distribution on the DeCART code, and then to evaluate the performance by solving the C5G7 three-dimensional benchmark and a simplified three-dimensional SMART core problem. In C5G7 problem with 24 CPUs, a speedup of maximum 22 is obtained on IBM Regatta machine and 21 on a LINUX Cluster for the MOC kernel, which indicates good parallel performance of the DeCART code. The

Simplified SMART problem which need about 11 GBytes memory with one processors requires about 940 MBytes, which means that the DeCART code can now solve large core problems on affordable LINUX Clusters.

1. 서 론

3차원 전노심에 대한 중성자 수송 계산[1,2,3]은 컴퓨터의 성능 향상 및 계산 방법론의 진보에 따라 기존의 노심계산 방식, 즉 2차원 집합체 계산 및 집합체 균질화 상수 생산, 3차원 노심계산, 역균질화 등을 수반한 방식이 가지는 한계성을 극복할 새로운 체계로서 꾸준히 연구되고 있다. 그러나 이러한 3차원 전노심에 대한 중성자 수송 계산은 컴퓨터 하드웨어 및 방법론 개발 등 외적 상황의 변화에도 불구하고 여전히 상당량의 계산시간 뿐만 아니라 기억용량을 요구하여 기존의 상용원자로와 같은 대형문제의 해를 구하는 일은 슈퍼컴퓨터에 의존해야만 하는 매우 어려운 일이다. 따라서 이러한 대형 문제에 대한 해를 현실적으로 구하기 위해서는 여러 대의 컴퓨터가 계산량 뿐만 아니라 기억용량을 분산하여 상호 필요한 자료를 주고받으면서 계산하는 병렬 계산 방법이 필수적이다.

DeCART 코드는 3차원 전노심 계산을 위해 3차원 중성자 수송방정식을 반경방향 및 축방향으로 각각 적분하여 유도된 2D/1D 횡방향 중성자 누설 연계방정식을 풀어 3차원 중성자 수송방정식의 근사해를 구하며, 그 해를 효율적으로 구하기 위해 기존의 노달 방법에서 가속기법으로 많이 사용되는 거시격자 유한차분법(CMFD)를 사용한다. DeCART 코드에 구현된 3차원 중성자 수송방정식의 해를 구하는 방법은 전체 계산시간의 약 95% 이상을 2차원 MOC 계산에 소요하며, 전체 기억용량의 상당부분을 MOC 및 다군 CMFD 관련 변수들에 할당한다. 따라서 DeCART 코드에서는 실질적인 병렬계산은 MOC 계산 모듈에서만 수행되며, 메모리 분산은 이들 주요 변수들의 분산에 초점을 두고 있다.

이 논문의 제 2장에서는 이들 DeCART 코드에서 사용하는 프로세스 그룹핑, 병렬계산 방법 및 메모리 분산처리 방법에 대해 자세히 소개한다. 제 3장에서는 이들 메모리 분산처리에 따른 데이터의 전송에 초점을 두어 DeCART 코드에서의 계산 흐름도를 자세히 설명한다. 마지막으로 제 4장에서는 DeCART 코드의 병렬계산 성능을 평가하기 위해 C5G7[4] 문제에 대해 적용하여 그 결과를 분석하며, 단순화된 SMART 3차원 노심에 적용하여 메모리 분산처리 방법의 효용성을 점검한다.

2. 병렬계산 및 메모리 분산처리

DeCART 코드는 병렬계산을 수행하더라도 수렴과정 및 모든 계산 결과가 직렬 계산시와 동일하도록 프로그래밍 되었으며, 메모리 분산처리도 반경방향 MOC 병렬계산의 효율성을 유지하는 범위내에서 이루어졌다. 즉, MOC 계산이 평면별로 계산되므로 평면별 메모리 분산이 최우선 순위를 가지며, 셀 균질화에 따른 등가상수의 저장도 평면별로 분산

되었다. 이에 따라 다군 축방향 NEM 계산 및 다군 CMFD 계산 모듈 등에서는 균질화 상수의 상호 통신이 필요하며, 그 결과의 저장을 위한 통신 또한 필요하다.

표 1은 3차원 핵연료 집합체 문제를 하나의 CPU를 사용하여 풀 경우, DeCART 코드 내 각 모듈별 계산 시간을 정리하여 나타낸 것이다. 표에서 알 수 있듯이, 중성자 수송방정식의 해를 구하기 위한 총 계산시간의 대부분은 MOC 모듈에서 사용하고 있으며 약 97% 이상을 차지한다. 이에 따라 DeCART 코드의 계산상의 병렬처리는 이러한 MOC 모듈의 병렬처리에 초점을 두어야 함을 알 수 있다.

표 1. 주요모듈별 계산시간(12 Plane Full Assembly, Tight Ray)

모듈	Computing Time, min/Fraction	Dependency
MOC + Subroup	144.2 / 97.4 %	δA, Nazi, Npol
NEM	0.2 / 0.1 %	No. of Cells
MG CMFD	1.3 / 0.9 %	No. of Cells
2G CMFD	0.1 / 0.1 %	No. of Cells
T/H	1.0 / 0.7 %	No. of Channels
Miscellaneous	1.2 / 0.8 %	No. of Channels
Total	148.0 / 100 %	

표 2. 주요변수별 메모리 요구량(12 Plane Full Assembly, Tight Ray)

모듈	변수	Memory 요구량 (MBytes)	Dependency
MOC	ϕs(Scalar Flux)	60	Flat Source Region
	Ψ(Angular Flux)	1504	B.D. Radial Surface, δA, Nazi, Npol
	Σeq(Equivalence XS) Niso, ...	57	균일핵단면적 영역수
NEM	$\tilde{\Phi}_1$, $\tilde{\Phi}_2$ (Moments)	14	No. of Cells
MG CMFD	Σ, ϕ, D-tilde, D-hat ...	82	No. of Cells
Library	-	35	무관
기타		20	
Total		1772	

표 2는 표 1과 같은 문제에 대한 주요 변수별 메모리 요구량을 나타낸다. 표에서 알 수 있듯이, 많은 양의 기억용량을 요구하는 주요 변수는 MOC 모듈에서 사용하는 균일 중성

자원 영역의 중성자속 및 문제 최외곽 경계에서의 각 중성자속과, 다군 CMFD 모듈에서 사용하는 셀별 균질화된 등가상수 및 중성자속이다. 이 문제에서는 특별히 각 중성자속은 중성자 추적선을 아주 촘촘히 생성함으로 인해 매우 많은 메모리가 요구되었으나, 실제 계산에서 널리 이용하는 추적선 수의 경우에는 표 2에 비해 약 1/10배 정도로 감소한다. DeCART 코드의 메모리 분산처리는 표 2의 주요변수에 대한 메모리 분산에 중점을 두었다. 이 장에서는 이러한 DeCART 코드의 주요 병렬 특성에 대해 기술한다.

2.1 MPI 프로세스 그룹핑

MPI 프로세서 그룹은 전체 프로세스내 존재하는 프로세스들의 부분집합으로 전체 문제를 부분으로 나누어 각 프로세스 그룹들에게 쉽게 할당하기 위해 만들어진 개념이다. DeCART 코드는 각 평면별 MOC 계산을 몇 개의 프로세스 그룹들에게 할당하며, 프로세스 그룹들은 그룹내 존재하는 모든 프로세스들을 모두 참여시켜 할당된 평면들을 차례대로 풀게 된다. 이들 프로세스 그룹내 존재하는 모든 프로세스들은 같은 양의 메모리를 갖게 되며, MOC 계산시 할당된 방향각에 대해 추적선 계산을 수행한다. 예를 들어, 방향각의 수가 8개이고 프로세스 그룹내 프로세스의 수가 4개이면 각 프로세스는 2개씩의 방향각을 할당받으며 주어진 방향각에 대해 추적선 계산을 수행하게 된다. 이때, 할당받는 2개의 방위각은 경계면에서 서로 반사될 수 있는 대칭각으로 설정된다. 그룹내 존재하는 모든 프로세스들의 추적선 계산이 끝나면, 각 프로세스에서 계산된 영역별 중성자속과 셀 경계면 중성자류의 합산이 수행되고, 이들 합산된 변수값은 모든 프로세스들에 전달되어 기존의 변수값을 대체하게 된다.

프로세스 그룹은 하나의 그룹 Master(GM: Group Master)와 다수의 그룹 Slave (GSlave: Group Slave)로 구성된다. MOC 계산을 제외한 CMFD 계산 및 축방향 NEM 계산시에는 GM들 만이 참여하게 되며, GSlave들은 MOC 계산이 끝난후 이들 계산이 끝나기를 기다리게 된다.

2.2 병렬 계산

DeCART 코드내 병렬 계산은 MOC 계산 모듈, 다군 핵단면적 라이브러리를 사용할 경우의 핵단면적 계산 모듈, 셀 균질화 및 군축약모듈에서만 수행된다. DeCART 코드내 MOC 계산은 축방향 중성자 누설항이 바로 전에 수행된 NEM 계산에 의해 중성자원으로 주어진 상태에서 수행되므로, 각 평면들은 타 평면의 계산결과와는 무관한 독립 계산 체계를 갖게된다. 따라서 MOC 모듈에서는 어떤 평면을 먼저 풀어도 상관이 없게 되는 자연적 병렬계산성이 확보된다.

DeCART 코드에서의 핵단면적 처리 모듈의 특징[5]은 핵단면적 수시 계산체계와 균일 핵단면적 영역으로 대변할 수 있다. 이들 특징들은 전노심 계산시 요구되는 핵단면적 관

련 변수들의 과도한 기억용량을 제거하기 위한 방안으로, 핵단면적 수시 계산체계는 MOC 계산시 필요한 핵단면적을 미리 계산해 두지 않고, 핵단면적이 필요할 때마다 수시로 계산하여 공급하는 방식이다. 그리고 균일 핵단면적 영역 체계는 핵단면적에서 거의 차이가 없을 것으로 판단되는 몇 개의 영역을 묶어 하나의 균일 핵단면적 영역으로 대표하는 방식으로, 핵연료봉 내에서는 봉의 중심을 기준으로 하여 환형으로 만들어진다.

이러한 DeCART 코드의 핵단면적 처리 방식은 기존 코드들이 MOC 계산 전에 모든 영역에서의 핵단면적을 계산하고 이를 저장하는 방식과는 달리, 계산과정 중 수차례에 걸쳐 핵단면적 계산을 수행하므로 기존 코드들에 비해 많은 핵단면적 계산시간을 요구하게 된다. 따라서 병렬계산시에는 이러한 핵단면적 계산도 병렬로 수행할 필요가 있다. 핵단면적 계산은 각 핵단면적 영역별 상호 독립상태에서 수행되므로, 병렬 계산을 손쉽게 수행할 수 있으며, 다만 각 프로세스 그룹내에서의 병렬계산후 계산된 핵단면적들의 전송을 위한 통신시간이 추가로 요구된다.

셀 균질화 계산은 다군 CMFD 계산에 필요한 다군 군정수 및 중성자류 보정인자 생산을 위해 필요하며, MOC 계산으로부터 구해진 영역별 중성자속을 사용하여 모든 셀에서 수행된다. 군축약 계산은 2군 CMFD 계산에 필요한 2군 군정수 및 중성자류 보정인자 생산을 위한 계산으로, 다군 CMFD 계산으로부터 구해진 각 셀별 중성자속 스펙트럼을 이용하여 구해진다. 이들 셀 균질화 계산 및 군 축약 계산은 각 셀별 상호 독립상태에서 수행되므로, 병렬 계산이 손쉽게 수행될 수 있다.

2.3 메모리 분산 처리

DeCART 코드내 메모리 분산은 많은 메모리를 요하는 주요 변수들에 대해서만 수행된다. 이들 주요 변수들은 표 2에서 알 수 있듯이, MOC 모듈에서 사용하는 균일 중성자원 영역의 중성자속 및 문제 최외곽 경계에서의 방향 중성자속, 다군 CMFD 모듈에서 사용하는 셀별 주요 변수, 즉 균질화된 핵단면적, 중성자류 보정인자 및 중성자속 등이다. 이들 주요 변수들은 각 프로세스별로 할당된 평면에 대해서만 메모리가 할당되며, 따라서 프로세스 그룹수에 비례하여 할당된 메모리는 감소하여 최소 1개 평면에 대한 메모리가 된다.

3. 병렬계산 흐름도

그림 1은 DeCART 코드에서의 주요 계산 모듈별 병렬계산 흐름도를 보여준다. CMFD 및 축방향 NEM 계산은 Master 프로세스에 의해서만 수행되며 다른 프로세스들은 메모리 분산을 위해 Master 프로세서에서 필요한 자료 및 결과들을 저장하는 역할을 수행한다.

2군 CMFD 계산을 수행하기 위해서는 모든 평면에 대한 2군 핵단면적 및 중성자속이

필요하다. 따라서 GM들은 2군 CMFD 계산 전에 에너지 군 축약 계산을 수행하여 각 평면별 2군 등가상수 및 중성자속을 생산하고 이를 Master 프로세스에게 전송해 주어야 한다. 그리고 Master 프로세스는 2군 CMFD 계산을 수행하고 그 계산 결과인 2군 중성자속을 GM들에게 전송해 주어야 한다.

다군 CMFD 계산은 먼저 각 GM들이 Master 프로세스로부터 전송 받은 2군 중성자속을 이용하여 다군 중성자속을 갱신하는 작업으로부터 시작한다. 다음으로 각 에너지군에 대해 순차적으로 중성자원 계산, 중성자속 및 중성자원 전송, 1군 CMFD 계산, 계산결과 전송 등의 계산이 진행된다. 각 GM들은 다군 CMFD 계산내에서 1군 CMFD 계산을 수행할 할당된 에너지군이 있으며, 이들 할당된 에너지군 계산시에는 계산전에 타 프로세스들로부터 필요한 자료를 받으며, 계산후에는 계산 결과를 전송한다. 각 GM에 대한 1군 CMFD 계산용 에너지군 할당은 1군 CMFD 계산용 행렬체계 구성시 필요한 등가상수 자료의 빈번한 전송을 막아 통신시간을 줄이기 위한 것으로, 각 에너지군별 1군 CMFD 계산용 행렬체계는 MOC 계산후 수행되는 셀 균질화시 미리 계산되어 각 GM들에 분산 저장된다.

다군 CMFD 계산이 끝나면 Master 프로세스에 의한 T/H 계산이 수행된다. 이때 각 GM들은 할당된 셀에 대한 출력분포를 계산하여 Master 프로세스에 전송해 주어야 하며, Master 프로세스는 T/H 계산 후 그 결과인 냉각재 온도 및 밀도, 핵연료 온도 등을 GM에 전송해 준다. 그러면 각 GM들은 이들 결과들를 다시 GSlave들에게 전송해 준다. T/H 결과의 전송이 모두 끝나면 모든 프로세스들은 이들 T/H 자료들을 바탕으로 냉각재 핵종들의 수밀도를 다시 계산하여 갱신하게 된다.

T/H 계산이 끝나면, Master 프로세스에 의한 축방향 1차원 셀 별 NEM 계산이 수행된다. 각 GM 들은 축방향 NEM 계산을 위한 중성자속 및 중성자속 모멘트, 반경방향 중성자 누설 등을 갱신하며, 이러한 자료들을 핵단면적 자료와 함께 Master 프로세스에 전송한다. Master 프로세스에 의한 1차원 NEM 계산이 끝나면 Master 프로세스는 MOC 계산을 위한 축방향 중성자 누설, CMFD 계산을 위한 축방향 중성자류 보정인자 등을 계산하여 NEM 계산결과와 함께 각 GM 들에게 전송한다.

축방향 NEM 계산이 끝나면, 모든 프로세스들이 참여하는 평면별 MOC 계산이 수행된다. 이를 위해 먼저 각 GM들은 MOC 영역별 중성자속 및 방향 중성자속의 갱신을 위한 곱인자 계산을 수행되며, 이들 인자들을 GSlave에게 전송한다. GSlave는 이들 곱인자를 사용하여 MOC 영역별 중성자속 및 방향 중성자속을 갱신한다. 각 프로세스 그룹들은 할당된 평면에 대해 MOC 계산을 수행하며, 각 그룹내 프로세스들은 할당된 평면내 할당된 방향각에 대해 추적선 계산을 수행한다. 한 평면에 대한 추적선 계산이 끝나면 GM은 그룹내 프로세스들로부터 영역별 중성자속 및 셀 경계면 중성자류를 받아 이를 합산하며, 그 결과를 다시 각 그룹내 프로세스들에게 전송한다.

반경방향 MOC 계산이 계산이 끝나면, 비균질 노심에 대한 영역별 중성자속 및 셀 경계면에서의 중성자류가 구해지며, 이들 결과들을 바탕으로 셀 균질화 및 등가상수 계산이

수행된다. 셀 균질화 및 등가상수 계산에는 GM들만 참여하게 되며 GSlave들은 다시 MOC 계산이 수행될 때까지 기다린다.

셀 균질화 및 등가상수 계산이 끝나면, MOC 계산전에 수행한 축방향 1차원 NEM 계산을 갱신된 등가상수를 사용하여 다시 수행하며, CMFD 계산을 위한 축방향 중성자류 보정인자를 갱신하게 된다. NEM 계산이 끝나면 군축약 과정을 거쳐 앞에서 언급한 2군 CMFD 계산을 다시 수행하게 된다.

4. 병렬계산 결과 및 성능 평가

DeCART 코드의 병렬계산능 평가는 C5G7 3차원 벤치마크 문제 및 단순화된 SMART 3차원 노심를 통해 평가되었다. 이들 중 단순화된 SMART 3차원 노심 문제는 약 11 GByte 이상의 메모리를 요구하여 병렬계산능 평가에는 적절하지 못하며, 메모리 분산능 평가를 위해서만 사용되었다. 병렬계산능 평가에는 C5G7 3차원 벤치마크 문제만이 사용되었다.

표 3 및 그림 2는 C5G7 3차원 벤치마크 문제에 대한 병렬계산능 평가 결과를 나타낸다. IBM Regatta 슈퍼컴퓨터에서는 OpenMP에 의한 계산도 가능하기 때문에 MPI Grouping 계산 결과와 함께 수록하였다. DeCART 코드내 MOC 모듈에서의 최대 Speedup은 24개 프로세스를 사용할 경우 IBM 컴퓨터에서는 약 22 정도가, LINUX Cluster 에서는 약 21 정도가 나타남을 알 수 있다. 그리고 Speedup은 프로세스 그룹수에 대해서는 거의 비례하여 증가함을 알 수 있으나, 그룹당 프로세스의 수 또는 OpenMP Thread 수에 대해서는 상대적으로 낮은 Speedup이 나옴을 알 수 있다. 이는 OpenMP나 그룹당 프로세스 수를 증가시킬 경우 추적선 계산이 끝난후 영역별 중성자속과 셀경계면에서의 중성자류의 합산을 위한 통신시간의 증가가 주 원인으로 풀이된다. MOC 모듈에서의 Speedup과는 달리 총 계산시간에서의 Speedup은 MOC 모듈을 제외한 다른 모듈의 병렬화가 되지 않음으로 인해 MOC 커널에 비해 떨어짐을 알 수 있다. 총 계산시간에서의 Speedup은 24개 프로세스를 사용할 경우 IBM 컴퓨터에서는 약 16, LINUX Cluster에서는 약 14 정도가 나온다. 그리고 IBM 컴퓨터와 LINUX Cluster에서의 통신시간을 비교하면, IBM 컴퓨터가 훨씬 적음을 알 수 있다.

표 4는 단순화된 SMART 330 문제에 대한 3차원 1/4 노심 계산 결과를 요약한 것이다. 이 문제는 한 대의 컴퓨터로 풀 경우 촘촘한 추적선의 경우 약 11 GBytes의 메모리를 요구하여 슈퍼컴퓨터가 아니면 풀기 어려운 문제이다. 그러나 메모리 분산처리의 경우는 표에서 알 수 있듯이 촘촘한 추적선의 경우에도 약 940 MBytes의 메모리를 요하여 Linux Cluster에서도 충분히 계산될 수 있음을 알 수 있으며, 통상적으로 널리 사용되는 추적선의 경우에는 약 580 MBytes의 메모리를 요한다. 계산 결과에서는 일상적인 추적선의 경우 촘촘한 추적선에 비해 k-eff 에서 약 60 pcm 정도의 오차가 났으나, T/H 계산 결과에서는 거의 차이가 없으며, 계산 시간에서는 약 3배정도 빠름을 알 수 있다. 또

한 계산 모듈별 계산시간에서는 일상적인 추적선의 경우에는 공명처리를 포함한 추적선 계산이 전체 계산시간의 약 54 %를, 촘촘한 추적선의 경우에는 약 86 %를 차지하여 아직도 추적선 계산에 상당히 많은 시간이 소요됨을 알 수 있다.

5. 결론

이 연구에서는 DeCART 코드가 상당량의 계산시간과 메모리를 요하는 큰 문제에도 잘 이용될 수 있도록 프로세스 그룹핑에 기초한 병렬처리 및 메모리 분산방식을 DeCART 코드에 이식하고 그 성능을 평가하였다. 24개의 프로세스를 사용할 경우 C5G7 3차원 벤치마크 문제에서 DeCART 코드의 MOC 모듈은 약 21 ~ 22 의 Speedup을 보였으며, 약 11 GBytes의 메모리를 요하는 단순화된 SMART 3차원 노심 문제는 메모리 분산효과로 인해 약 940 MBytes의 메모리로 계산이 가능했다. 따라서 메모리 분산이 포함된 DeCART의 병렬처리 기법은 병렬계산을 통해 계산 시간을 단축한다는 기본적인 의의 이외에 메모리의 크기가 제한적인 저가의 LINUX Cluster 상에서도 매우 큰 문제를 처리할 수 있게 해 준다는 측면에서 그 가치가 있다 하겠다.

감사의 글

이 연구는 과학기술부가 수행하고 있는 선진기술확보사업의 일환으로 수행되었다.

참 고 문 헌

1. N.Z.Cho, G.S.Lee and C.J.Park, "Fusion of Method of Characteristics and Nodal Method for 3-D Whole Core Transport Calculation," *Trans. Am. Nucl. Soc.*, **86**, 322 (2002)

2. T. Takeda, et al., "Development of 3-D FBR Heterogeneous Core Calculation Method based on Characteristics Method," *Proc. Kor. Nucl. Soc., Reac. Phys. 3-rd Joint Korea Japan Session*, Kwangju, Korea, May 2002.

3. 주한규 외, "DeCART 코드의 삼차원 전노심 수송계산 방법과 성능," 한국원자력학회 2002 추계학술발표회 논문집, 용평, 2002.

4. E. E. Lewis, et al., "Benchmark Specification for Deterministic 2-D/3-D MOX Fuel Assembly Transport Calculations without Spatial Homogenization (C5G7MOX)," NEA/NSC/DOC(2001)4.

5. 조진영 외, “DeCART 코드에 의한 SMART 노심 2차원 계산,” 한국원자력학회 2002 추계학술발표회 논문집, 용평, 2002.

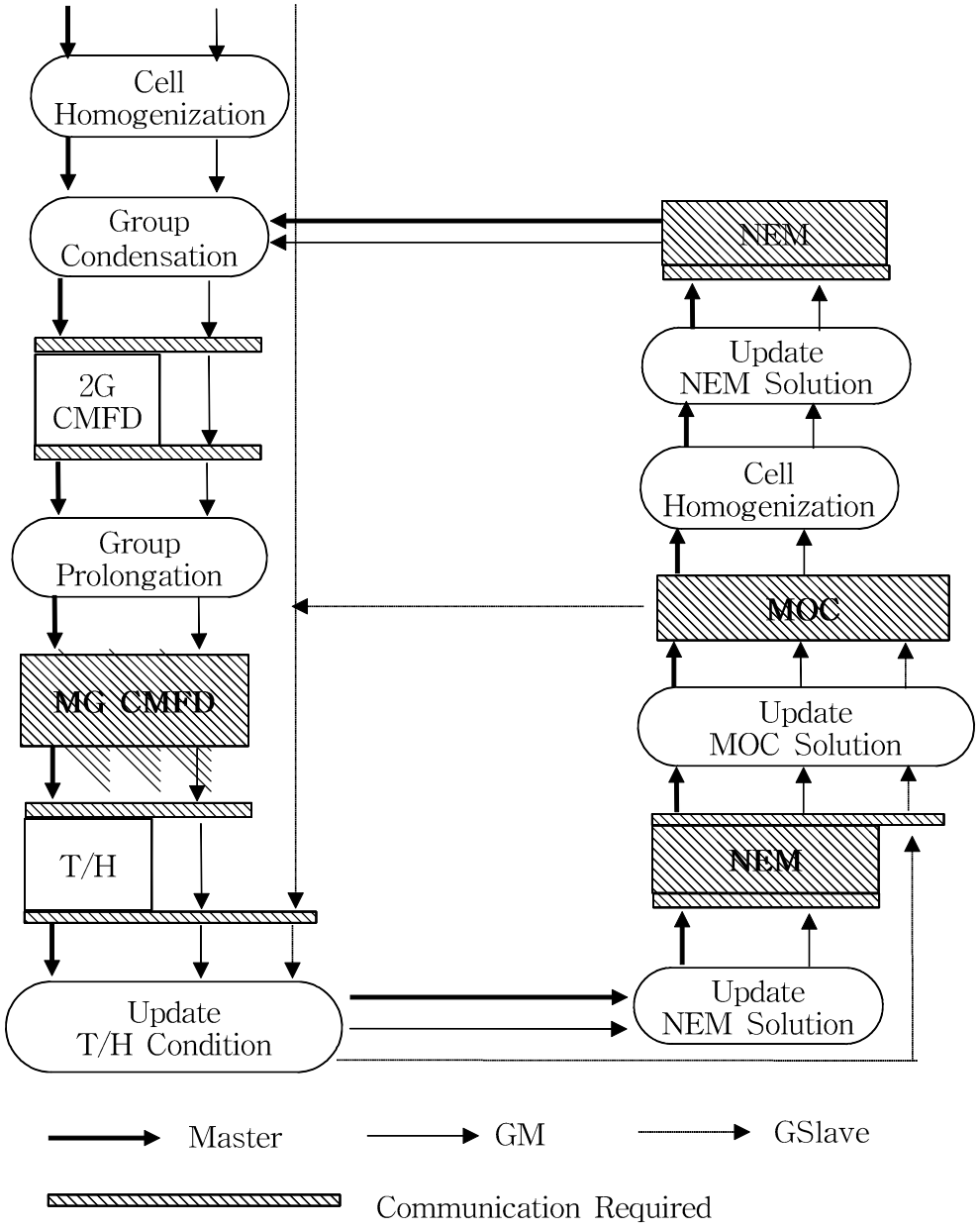

그림 1. DeCART 코드내 병렬계산 흐름도

표 3. C5G7 3차원 벤치마크 문제에 대한 병렬 계산능 결과

Machine	Processor Combination	T_{comm} [1], sec	Ray Tracing Kernel		Total		a/b*100
			a)Time, sec	Speedup	b)Time, sec	Speedup	
IBM Regatta	(1[2],1[3],1[4])	0.0	4298.4	1.00	4381.1	1.00	98.1
	(1,1,2)	0.0	2235.9	1.92	2318.6	1.89	96.4
	(1,1,4)	0.0	1150.5	3.74	1233.2	3.55	93.3
	(1,1,8)	0.0	639.1	6.73	721.8	6.07	88.5
	(2,1,1)	2.0	2137.9	2.01	2220.6	1.97	96.3
	(2,1,2)	1.9	1108.3	3.88	1191.0	3.68	93.1
	(2,1,4)	1.9	569.6	7.55	652.3	6.72	87.3
	(2,1,8)	1.8	343.0	12.53	425.7	10.29	80.6
	(3,1,1)	2.2	1429.0	3.01	1511.7	2.90	94.5
	(3,1,2)	2.2	739.7	5.81	822.4	5.33	89.9
	(3,1,4)	2.0	380.6	11.29	463.3	9.46	82.1
	(4,1,1)	2.4	1086.0	3.96	1168.7	3.75	92.9
	(4,1,2)	2.3	554.3	7.75	637.0	6.88	87.0
	(4,1,4)	2.1	315.5	13.63	398.2	11.00	79.2
	(6,1,1)	2.5	723.0	5.95	805.7	5.44	89.7
	(6,1,2)	2.4	373.4	11.51	456.1	9.61	81.9
	(12,1,1)	2.6	367.3	11.70	450.0	9.74	81.6
	(12,2,1)	4.5	193.6	22.20	276.3	15.86	70.1
LINUX Cluster	(1,1,1)	0.0	11554.3	1.00	11800.1	1.00	97.9
	(2,1,1)	21.0	5870.7	1.97	6116.9	1.93	96.0
	(3,1,1)	28.3	3947.9	2.93	4191.9	2.82	94.2
	(4,1,1)	31.5	3003.1	3.85	3248.2	3.67	92.5
	(6,1,1)	35.1	2009.3	5.75	2254.8	5.23	89.1
	(12,1,1)	40.5	1031.8	11.20	1278.6	9.23	80.7
	(6,2,1)	65.3	1057.4	10.93	1304.0	9.05	81.1
	(12,2,1)	55.6	543.1	21.28	818.5	14.42	66.4
	(6,4,1)	82.0	601.9	19.20	869.1	13.58	69.3

1) 통신시간, sec

2) 프로세스 그룹수

3) 프로세스 그룹당 프로세스수

4) OpenMP Threads 수

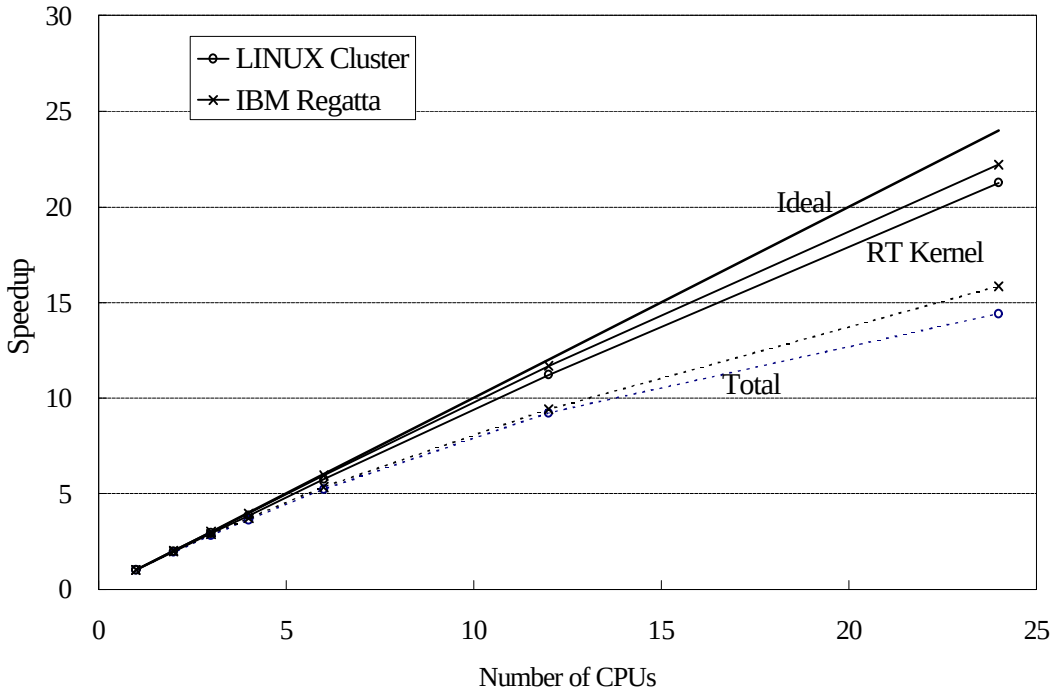

그림 2. C5G7 3차원 벤치마크 문제에 병렬 계산능

표 4. 단순화된 SMART 노심에 대한 병렬 계산 결과
(12 Planes including 10 Fuel Planes, 24 CPUs)

추적선, 간격/방향각/방위각		0.05/4/2		0.02/8/4		Ratio/Diff
메모리, 1 CPU / Total, MBytes		581/6291		941/10611		1.6
k-eff		1.02630		1.02690		-57 pcm
Max. Tf ℃		974.7		975.6		-0.9
Core Average Tm ℃		295.3		295.3		0.0
No. of Ray Sweeping		7		7		0
Time, min/Fraction	Subgroup	23.6	17.9 %	128.1	29.3 %	5.4
	CMFD	32.7	24.7 %	32.8	7.5 %	1.0
	NEM	18.7	14.1 %	18.7	4.3 %	1.0
	Tay Tracing	48.0	36.3 %	248.4	56.8 %	5.2
	T/H+Misc.	9.4	7.1 %	9.3	2.1 %	1.0
	Total	132.4	100.0 %	437.3	100.0 %	3.3
	Communication	38.6	29.1 %	45.4	10.4 %	1.2