이더넷과 인피니밴드 네트워크 기반의 분산 메모리 시스템에서 병렬성능 분석

전 병 진,[1] 최 형 권*[2]
[1]서울과학기술대학교 에너지시스템공학과
[2]서울과학기술대학교 기계자동차공학과

PERFORMANCE ANALYSIS OF THE PARALLEL CUPID CODE IN DISTRIBUTED MEMORY SYSTEM BASED ETHERNET AND INFINIBAND NETWORK

B.J. Jeon[1] and H.G. Choi*[2]
[1]Dept. of Energy System, Seoul National Univ. of Science and Technology
[2]Dept. of Mechanical/Automotive Engineering, Seoul National Univ. of Science and Technology

In this study, a parallel performance of CUPID-code has been investigated for both Ethernet and Infiniband network system to examine the effect of cache memory and network-speed. Bi-conjugate gradient solver of CUPID-code has been parallelised by using domain decomposition method and message passing interface (MPI). It is shown that the parallel performance of Ethernet-network system is worse than that of Infiniband-network system due to the slow network-speed and a small cache memory. It is also found that the parallel performance of each system deteriorates for a small problem due to the communication overhead, but the performance of Infiniband-network system is better than Ethernet-network system due to a much faster network-speed. For a large problem, the parallel performance depends less on network system.

Key Words : 이더넷(Ethernet), 인피니밴드(Infiniband), CUPID 코드(CUPID code), 이중공액구배법(Bi-Conjugate Gradient)

1. 서 론

기계, 의료, 원자력 등과 같이 대규모 수치해석 연산이 필요한 분야에서는 빠른 계산 속도가 요구되어지고 있다. CPU의 클럭 주파수를 증가시켜 계산 속도를 올리는 것은 발열 문제로 인하여 제한되었다. 따라서, 하드웨어 업체들은 CPU의 클럭 주파수를 낮추고 하나의 칩에 다수의 코어를 집적하는 멀티코어 방식으로 제품을 개발하고 있다. 또한, 여러 개의 컴퓨터를 네트워크로 연결한 분산 메모리 형태의 클러스터링의 수요는 계속 증가되고 있는 추세이다[1].

2000년대 이전에는 여러 개의 컴퓨터를 기가비트 이더넷(Ethernet) 네트워크 스위치로 연결하여 클러스터링을 구성하였다. Yoon and Chung[2]은 기존의 기가비트 이더넷 카드의 드라이버를 수정하여 전송되는 패킷의 최대 크기를 높이는 연구를 수행하였다. 그 결과로 대역폭은 평균 13MB/s가 향상되고 지연시간은 20%를 감소되었다. 또한, 패킷의 최대 크기를 더 높일 수 있다면 더 좋은 성능을 얻을 수 있을 것이라고 예측하였다. Lee[3]는 이더넷 기반의 리눅스 클러스터 시스템을 이용하여 공액구배법(Conjugate Gradient)과 LU 알고리즘에 대한 병렬 성능 향상 분석을 수행하였다. 그들은 프로세서의 수가 증가함에 따라 공액구배법의 성능이 저하되는 것을 확인하고 병렬 성능을 최대화하기 위해서는 알고리즘의 병렬 특성 파악과 네트워크 장비의 성능 향상이 중요한 요인이라고 보고하였다. 이와 같이 많은 연구자들은 네트워크 스위치로 연결된 클러스터의 계산 성능에 대한 관심이 매우 높았으나 이더넷 기반의 네트워크 장비에서는 한계를 직시하게 되었다.

2000년대 이후, 고속연산을 위한 클러스터링 분야에서는 RDMA (Remote Direct Memory Access) 기술이나 인피니밴드

Received: June 11, 2014, Revised: June 26, 2014, Accepted: June 27, 2014.
* Corresponding author, E-mail: hgchoi@seoultech.ac.kr
DOI http://dx.doi.org/10.6112/kscfe.2014.19.2.024

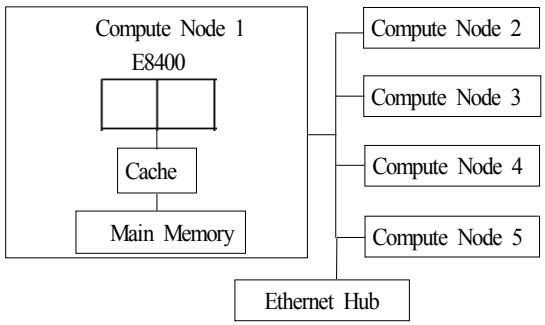
(Infiniband) 시스템 등을 이용하여 대용량 데이터를 빠르게 전송시킬 수 있는 방법에 대해서 큰 관심을 두고 있다. RDMA 기술은 RDMA Network Interface Card가 장착된 기존의 TCP/IP, 이더넷 등 기존 네트워크 시스템을 이용하여 낮은 지연시간과 높은 대역폭 전송을 위해 사용하고 있다[4]. RDMA 기술은 CPU가 개입하지 않고 네트워크가 자체적으로 데이터 전송을 관리하는 방식을 말한다[5]. 인피니밴드 네트워크 시스템은 RDMA 기술과 OS(Operating System) 내의 메시지를 주고받지 않는 기술을 모두 포함하고 있어서 기존의 이더넷 네트워크 시스템보다 통신에 필요한 지연시간이 최소화하였다. 또한, 넓은 대역폭과 빠른 전송속도, 양방향 전송이 가능하여 최근 대규모 클러스터링에 필수 장치로 여겨지고 있다. Liu et al.[6]은 그들이 새롭게 제안한 RDMA 기술이 접목된 인피니밴드 시스템 기반의 클러스터를 이용하여 기존의 인피니밴드 시스템보다 대역폭을 104% 향상시키고 시간지연을 24%를 줄일 수 있었다. Mininni et al.[7]은 인피니밴드 시스템 기반의 클러스터에서 MPI 라이브러리와 OpenMP 라이브러리를 혼합한 Hybrid 병렬 기법을 이용하여 전산유체역학 해석을 수행하였다. 그들은 2만개의 코어를 사용하였을 때 이상적인 병렬 성능의 83%에 해당하는 속도 향상 결과를 보여줬다. 이와 같이 병렬 성능의 향상을 위한 하드웨어 및 네트워크 기술, 병렬 알고리즘의 개발은 계속 진행되고 있다.

인피니밴드 시스템으로 구성된 분산 메모리 형태의 클러스터 구조에서 병렬 계산을 수행하기 위해서는 인피니밴드 시스템을 대상으로 개발된 MPI 라이브러리가 필요하다. MPI 병렬 계산을 위한 라이브러리는 MVAPICH2[8], Open MPI[9] 등과 같이 다양하게 존재한다. 다양한 그룹에서 개발된 MPI 라이브러리는 구조 및 장단점이 다르다. Shipman et al.[5]은 인피니밴드 시스템 기반의 클러스터에서 가장 널리 사용되는 MVAPICH[8]와 OpenMPI[9] 라이브러리의 성능 분석을 수행하였다.

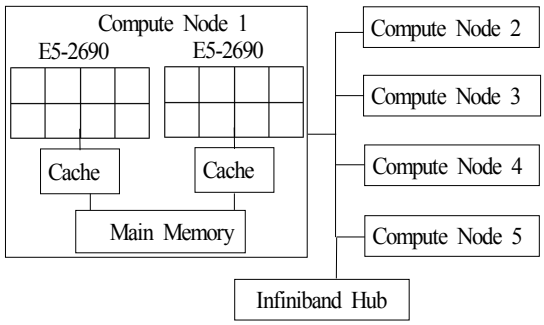
본 연구에서는 이더넷과 인피니밴드 네트워크 시스템 구성된 분산 메모리 형태의 클러스터 특성을 파악하기 위해 가압경수로 주요기기의 고정밀 열수력 해석을 위한 CUPID (Component Unstructured Program for Interfacial Dynamics) 코드의 압력방정식의 병렬연산을 수행하고 그 결과를 분석하고자 한다.

2. 병렬 컴퓨팅 시스템 구성

본 연구에서는 이더넷과 인피니밴드 네트워크 시스템 구성에 따른 CUPID 코드의 병렬 성능을 분석하기 위해서 Red Hat Enterprise 운영체제를 사용하는 클러스터에서 계산을 수행하였다. CUPID 코드의 컴파일은 각각 Intel compiler v10.1과 intel compiler XE 2013을 사용하였다. Fig. 1(a)는 이더넷 네트워크 시스템 방식으로 구성된 클러스터의 개략도이다. 총 5개의 계산 노드로 구성되어 있으며, 각 계산 노드는 2개의 코어를 가지는 Intel Core2 Duo E8400 모델의 CPU가 1개씩 장착되어 있다. CPU는 6MB의 캐쉬 메모리와 6GB의 메인 메모리를 독립적으로 사용하고 있다. Fig. 1(b)는 인피니밴드 네트워



(a) Ethernet Network



(b) Infiniband Network

Fig. 1 Structure of parallel computing system

Table 1 information of each compute node in cluster

		Ethernet Network (1 G)	Infiniband Network (40 G)
CPU	Model	Intel Core2 Duo E8400	Intel Xeon E5-2690
	Num. Core	2	10
	Frequency	3.0GHz	2.9GHz
	Cache Mem	6MB	20MB
	Instruction set	SSE2	AVX
Num. total core		2	20 (10core×2CPU)
Memory		6GB	64GB
OS		Red Hat Enterprise release 4	Red Hat Enterprise release 6
Compiler		intel compiler v10.1	intel compiler XE 2013
MPI Library		MPICH1	MVAPICH2

크 시스템 방식으로 구성된 클러스터의 개략도이다. 총 5개의 계산 노드로 구성되어 있으며, 각 계산 노드는 10개의 코어를 가지는 Intel Xeon E5-2690 모델의 CPU가 2개씩 장착되어 있다. 각 CPU는 20MB의 캐쉬 메모리를 독립적으로 사용하며 64GB의 메인 메모리는 서로 공유하고 있다. 이더넷 네트워크 기반의 경우에는 MPICH1, 인피니밴드 네트워크 기반의 경우에는 MVAPICH2 라이브러리를 사용하였다. 클러스터의 자세한 정보는 Table 1을 참고한다.

3. 영역분할과 알고리즘의 병렬화

본 연구에서는 영역분할법과 MPI 병렬 기법을 사용하여 아래의 행렬로 구성된 CUPID 코드의 압력방정식 해법 부분을 병렬화 하였다.

$$\begin{pmatrix} \left[1+\sum_f CC_6\right]_1 & \cdots & (CC_6)_{1,j} & \cdots & (CC_6)_{1,N} \\ \vdots & \ddots & & & \vdots \\ (CC_6)_{i,1} & \cdots & \left[1+\sum_f CC_6\right]_i & \cdots & (CC_6)_{i,N} \\ \vdots & & & \ddots & \vdots \\ (CC_6)_{N,1} & \cdots & (CC_6)_{N,j} & \cdots & \left[1+\sum_f CC_6\right]_N \end{pmatrix} \begin{pmatrix} \delta P_1 \\ \vdots \\ \delta P_i \\ \vdots \\ \delta P_N \end{pmatrix} = \begin{pmatrix} (BB_6)_1 \\ \vdots \\ (BB_6)_i \\ \vdots \\ (BB_6)_N \end{pmatrix} \tag{1}$$

위 행렬의 자세한 설명은 CUPID 코드의 매뉴얼을 참고한다[10]. 비정렬 격자계에 대해서 각 코어 간의 계산량을 균등하게 분배하기 위한 영역분할은 METIS 라이브러리[11]를 이용하였다. 압력방정식의 해법을 위한 예조건화된(Preconditioned) 이중공액구배법(Bi-Conjugate Gradient) 알고리즘은 수렴할 때까지 다음 식들을 반복하게 된다. 여기서 하첨자 j는 반복 계산 횟수를 의미한다.

$$\alpha_j = \left(\{z_j\}^T\{r_j\}\right)/\left(\{d_j\}^T[A]\{d_j\}\right) \tag{2}$$

$$\{x_{j+1}\} = \{x_j\} + \alpha_j\{d_j\} \tag{3}$$

$$\{r_{j+1}\} = \{r_j\} - \alpha_j[A]\{d_j\} \tag{4}$$

$$\{z_{j+1}\} = [M]^{-1}\{r_{j+1}\} \tag{5}$$

$$\beta_j = \left(\{z_{j+1}\}^T\{r_{j+1}\}\right)/\left(\{z_j\}^T\{r_j\}\right) \tag{6}$$

$$\{d_{j+1}\} = \{z_{j+1}\} + \beta_j\{d_j\} \tag{7}$$

MPI 병렬화를 적용할 때 국소행렬 및 국소벡터는 분할된 영역에서 독립적으로 정의되므로 식 (3), (4), (7)은 각 코어에서 독립적으로 수행된다. 식 (2), (6)은 전체 계산 영역에 대한 값이므로 각 코어에서 독립적으로 계산을 수행한 후 그 값을 다시 모아서 최종적으로 구하게 된다. 식 (5)에서 예조건화 벡터를 구하는 방법은 대각 예조건화 기법을 이용하였다. $[M_n] = Diag[A_n]$이고 $[M_n]^{-1}$은 각 국소영역 내부의 값으로 구성된 대각 성분들의 역수로 각 코어에서 독립적으로 수행할 수 있다[13].

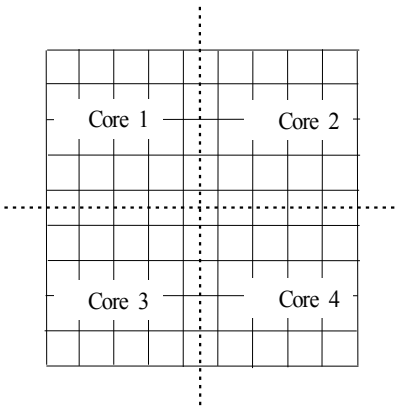


Fig. 2 Decomposition of computational domain[12,13]

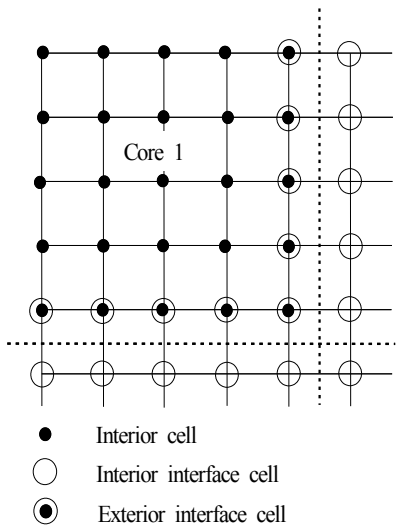


Fig. 3 Classification of cells in a sub-domain[12,13]

Fig. 2는 METIS를 이용하여 1개의 영역을 4개의 영역으로 분할한 예를 간략하게 보여주고 있다. 영역 분할을 하게 되면 Fig. 3에서 보는바와 같이 1번 코어를 기준으로 내부 영역과 경계 영역, 그리고 외부 영역으로 나누어지게 된다. 내부 영역에 위치하는 셀은 외부 영역의 셀과 데이터 교환을 하지 않는다. 그러나, 경계에 위치한 셀들은 외부 영역의 셀과 데

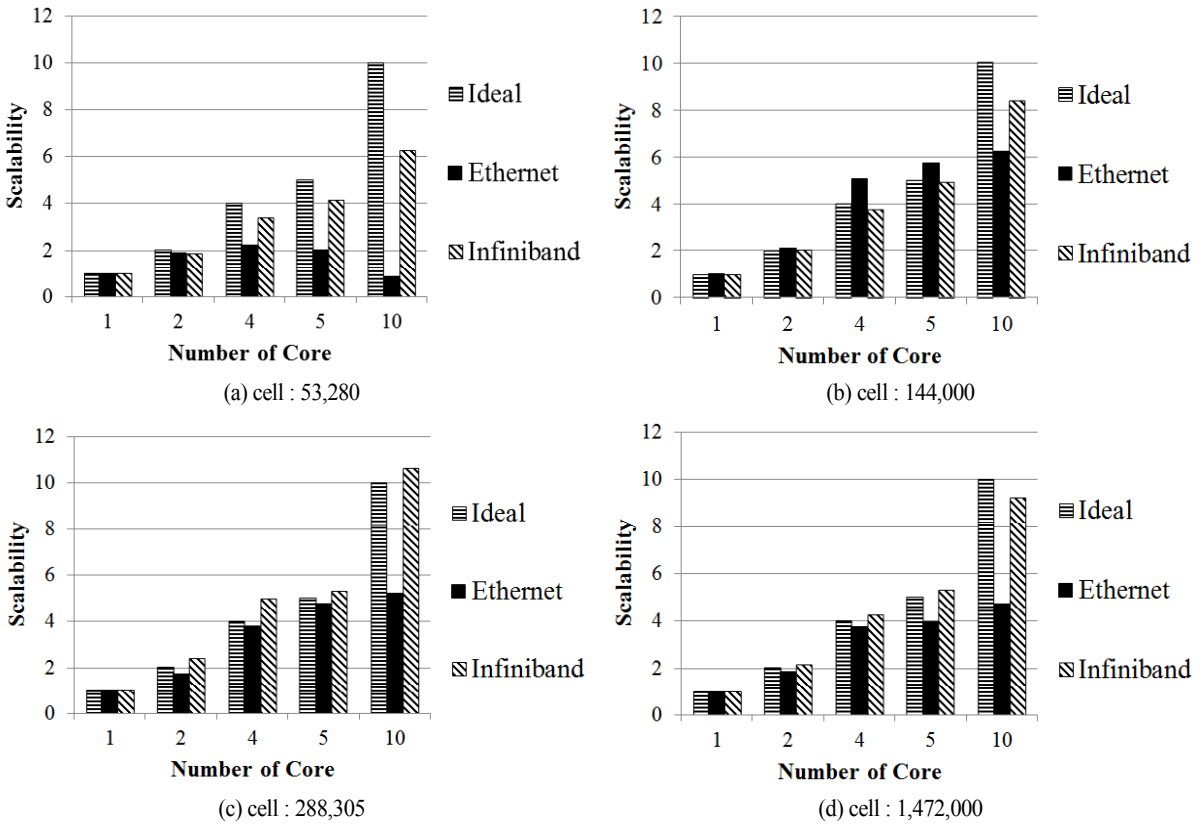


Fig. 4 Parallel performances for various grid

이터 교환이 필요하다. 이 때, 각 코어간의 데이터 교환은 MPI 라이브러리를 이용하여 수행하게 된다. 자세한 내용은 Kang et al.[13]의 논문을 참고한다.

4. 해석 결과 및 토의

본 연구에서는 5개의 계산 노드가 이더넷 또는 인피니밴드 네트워크로 연결된 분산 메모리 기반 클러스터에서 다양한 개수를 가지는 격자들에 대한 병렬 성능을 측정하였다. 1, 2, 4, 5개의 코어를 사용한 경우에는 각 노드당 1개의 코어를 사용하는 방식으로 작업을 할당하여 계산을 수행하였다. 그리고, 10개의 코어를 사용하여 계산한 경우에는 각 노드당 2개의 코어를 사용하였다.

4.1 셀의 개수가 적은 격자계에 대한 병렬 성능 분석

Fig. 4(a)는 네트워크 구성 특성에 대해서 셀의 개수가 53,280개일 때 계산에 관여한 코어 수에 따른 속도 향상 결과를 나타낸 그림이다. 이더넷 네트워크 기반 클러스터에서 코어를 2개 사용한 경우에는 계산 속도가 1.9배 빠르지만, 4개를 사용한 경우에는 2.2배로 성능이 저하되는 것을 볼 수 있다. 그리고 5개의 코어를 사용한 경우의 계산 속도는 4개의 코어를 사용한 경우보다 느려진다. 그 이유는 셀의 개수가 53,280개인 경우에는 계산량이 매우 적어서 MPI를 이용하여 노드 간에 필요한 정보를 주고받는 통신부하량이 차지하는 비율이 상대적으로 커지기 때문이다. 10개의 코어를 사용한 경우에는 2개의 코어가 하나의 캐쉬와 메모리를 공유하여 사용하는 것과 통신부하량 문제로 단일 계산 속도보다 더 느려지는 것을 알 수 있다. 인피니밴드 네트워크 기반 클러스터에서 코어를 2개 사용한 경우의 계산 속도는 1.9배, 4개를 사용한 경우에는 3.4배 빨라진다. 그리고 5개의 코어를 사용한 경우의 계산 속도는 4.1배, 10개의 코어를 사용한 경우에는 6.2배 빨라지는 것을 볼 수 있다. 인피니밴드 네트워크 기반 클러스터의 경우에는 빠른 데이터 전송 특성에 의해서 이더넷 네트워크 기반 클러스터의 성능보다는 높게 나왔지만 계산량이 매우 적은 이유로 작업에 할당된 코어의 개수만큼 계산

속도가 증가하지는 않았다.

4.2 캐쉬 메모리 사용량에 적합한 셀의 개수를 가지는 격자계에 대한 병렬 성능 분석

Fig. 4(b)는 네트워크 구성 특성에 대해서 셀의 개수가 144,000개일 때 계산에 관여한 코어 수에 따른 속도 향상 결과를 나타낸 그림이다. 이더넷 네트워크 기반 클러스터에서 코어를 2개 사용한 경우에는 계산 속도가 2.1배, 4개를 사용한 경우에는 5.1배, 5개를 사용한 경우에는 5.7배로 이상적인 속도 향상 기대치보다 더 빠른 것을 알 수 있다. 그 이유는 캐쉬 메모리의 효과로 설명할 수 있다. 셀의 개수가 144,000인 경우 5개로 영역을 분할하게 되면 1개의 영역당 셀의 개수가 28,800이다. Table 2에서와 같이 셀의 개수가 23,000개인 경우 요구되는 메모리는 약 12MB이며, 이 메모리량은 1개의 CPU에서 사용할 수 있는 캐쉬 메모리의 크기에 2배에 가깝다. 결과적으로 셀의 개수가 144,000인 경우에는 1개의 영역에서 캐쉬 메모리를 최대에 가까운 성능으로 사용하기 때문에 병렬 성능이 매우 좋게 나타나게 된다. 그러나, 10개의 코어를 사용한 경우에는 2개의 코어가 하나의 캐쉬와 메모리를 공유하여 사용하기 때문에 계산 속도가 6.2배로 병렬 성능이 저하되는 것을 볼 수 있다. 인피니밴드 네트워크 기반 클러스터에서 코어를 2개 사용한 경우에는 계산 속도가 2.0배, 4개를 사용한 경우에는 3.8배, 5개를 사용한 경우에는 4.9배, 10개를 사용한 경우에는 8.4배로 작업에 할당된 코어의 개수만큼 계산 속도가 증가하는 것을 볼 수 있다. 그러나, 인피니밴드 기반의 클러스터의 계산 성능은 매우 높기 때문에 셀의 개수가 144,000일 때 상대적으로 계산량이 부족하여 이상적인 속도 향상 기대치보다 더 빠르지는 않은 것을 볼 수 있다.

Fig. 4(c)는 네트워크 구성 특성에 대해서 셀의 개수가 288,305개일 때 계산에 관여한 코어 수에 따른 속도 향상 결과를 나타낸 그림이다. 이더넷 네트워크 기반 클러스터에서 코어를 2개 사용한 경우에는 계산 속도가 1.8배, 4개를 사용한 경우에는 3.8배, 5개를 사용한 경우에는 4.8배로 작업에 할당된 코어의 개수만큼 계산 속도가 증가하는 것을 볼 수 있다. 셀의 개수가 288,305인 경우에는 MPI에 의한 통신부하량보다 계산량이 충분히 많기 때문이다. 그러나, 10개의 코어를 사용한 경우에는 2개의 코어가 하나의 캐쉬와 메모리를 공유하여 사용하기 때문에 계산 속도가 5.3배로 병렬 성능이 저하되는 것을 볼 수 있다. 인피니밴드 네트워크 기반 클러스터에서 코어를 2개 사용한 경우의 계산 속도는 2.4배, 4개를 사용한 경우에는 5배, 5개를 사용한 경우에는 5.3배, 10개를 사용한 경우에는 10.6배로 이상적인 속도 향상 기대치보다 더 빠른 것을 알 수 있다. 그 이유는 캐쉬 메모리의 효과로 설명할 수 있다. 셀의 개수가 288,305인 경우 5개로 영역을 분할하게 되면 1개의 영역당 셀의 개수가 57,661이고 10개의 영역으로 분할하게 되면 1개의 영역당 셀의 개수가 약 28,830이다. Table 2에서 제시한 바와 같이 셀의 개수가 53,280개인 경우에 요구되는 메모리는 약 20MB이며, 23,000개인 경우에는 약 12MB이다. 이 메모리량은 1개의 CPU에서 사용할 수 있는 캐쉬 메모리의 크기와 동일하거나 절반에 가깝다. 결과적으로 셀의 개수가 288,305인 경우에는 1개의 영역에서 캐쉬 메모리를 최대에 가까운 성능으로 사용하기 때문에 병렬 성능이 매우 좋게 나타나게 된다. 이더넷 네트워크 기반 클러스터의 경우에는 1개의 CPU 당 주어진 캐쉬 메모리가 6MB로 계산에서 요구되는 메모리 크기에 비해 매우 적다. 따라서, 캐쉬 메모리에 의한 속도 향상을 기대할 수 없다.

Table 2 Memory requirement for each problem size

Number of cells	23,000	53,280	144,000	288,305	1,472,000
Memory requirement (MB)	12	20	40	120	480

4.3 셀의 개수가 매우 많은 격자계에 대한 병렬 성능 분석

Fig. 4(d)은 네트워크 구성 특성에 대해서 셀의 개수가 1,472,000개일 때 계산에 관여한 코어 수에 따른 속도 향상 결과를 나타낸 그림이다. 이더넷 네트워크 기반 클러스터에서 코어를 2개 사용한 경우에는 계산 속도가 1.9배, 4개를 사용한 경우에는 3.8배로 작업에 할당된 코어의 개수만큼 계산 속도가 증가하는 것을 볼 수 있다. 그러나, 통신부하량 증가에 의해서 5개의 코어를 사용한 경우에는 3.9배가 빨라진다. 10개의 코어를 사용한 경우에는 2개의 코어가 하나의 캐쉬와 메모리를 공유하는 추가적인 문제로 인해 5.3배로 병렬 성능이 급격하게 저하되는 것을 볼 수 있다. 인피니밴드 네트워크 기반 클러스터에서 2개의 코어를 사용한 경우의 계산 속도는 2.2배, 4개를 사용한 경우에는 4.3배, 5개를 사용한 경우에는 5.3배로 이상적인 속도 향상 기대치보다 더 빠른 것을 알 수 있다. 코어를 10개를 사용한 경우에는 9.2배의 속도 향상 결과가 나타났다. 셀의 개수가 1,472,000인 경우에는 계산량이 매우 많고 인피니밴드 네트워크의 특성으로 인해 통신부하량이 상대적으로 매우 적기 때문에 병렬 성능이 매우 좋게 나타난다. 또한, 5개로 영역을 분할하게 되면 1개의 영역당 셀의 개수가 294,400이고 10개의 영역으로 분할하게 되면 1개의 영역당 셀의 개수가 약 147,200이다. Table 2에서와 같이 셀의 개수가 288,305개인 경우 요구되는 메모리는 약 120MB이며, 144,000개인 경우에는 약 40MB이다. 10개의 영역으로 나눈 경우에는 계산에서 요구되는 메모리의 크기와 1개의 노드 내 2개의 CPU에 의한 캐쉬 메모리 크기가 동일하다. 따라서, 캐

쉬 메모리에 의한 효과도 적용된 결과라고 볼 수 있다.

5. 결 론

본 연구에서는 이더넷 또는 인피니밴드 네트워크로 연결된 분산 메모리 기반 클러스터의 병렬 성능을 비교하였다. 다양한 개수를 가지는 격자들에 대해 영역분할 및 MPI에 기반한 병렬 CUPID 코드의 성능을 측정하여 다음의 결론들을 도출하였다.

(1) 이더넷 네트워크 기반의 클러스터의 경우에는 캐쉬 메모리의 부족과 느린 네트워크 전송 속도에 의한 통신부하 문제에 의해서 대부분의 문제의 경우에 인피니밴드 네트워크 기반의 클러스터보다 병렬 성능이 떨어지는 것을 확인하였다.
(2) 캐쉬는 병렬 성능에 매우 중요한 영향을 미친다. 10개의 코어를 사용한 경우의 이더넷은 한 개의 캐쉬를 두 개의 코어가 공유하기 때문에 병렬 성능이 떨어진다.
(3) 두 가지 네트워크 시스템 모두 셀의 수가 적은 경우에는 계산량에 대한 통신부하의 비중이 높기 때문에 병렬 성능이 저하되나, 인피니밴드 시스템의 경우 빠른 전송속도에 의해서 상대적으로 우수한 병렬성능을 보인다. 반면, 두 시스템 모두 문제의 크기가 캐쉬 메모리 크기에 적합한 경우에는 5개 이내의 코어를 사용할 경우 이상적인 성능보다 높은 성능을 보였다.
(4) 두 가지 네트워크 시스템 모두 문제의 크기가 큰 경우에는 통신부하의 비중이 적어서 병렬 성능이 양호하다.

후 기

이 연구는 서울과학기술대학교 교내 학술연구비 지원으로 수행되었습니다.

Note

This paper is a revised version of a paper presented at the KSCFE 2014 Spring Annual meeting, Jeju KAL Hotel, Jeju May 22-23, 2014.

References

[1] 2010, Kirk, D.B. and Hwu, W.W, "Programming Massively Parallel Processors", *Elsevier Inc*, pp.19-27.
[2] 2002, Yoon, I.S. and Chung, S.H, M-VIA "Implementation on a Gigabit Ethernet Card", *Journal of KIISE:Computer Systems and Theory*, Vol.29, pp.648-654.
[3] 1999, Lee, B.S., "Building and Benchmarking of Linux Cluster Systems for Computational Fluid Dynamics", *Proceedings of the Korean Society of Computational Fluid Engineering 1999 Fall Annual Meeting*, pp.201-206.
[4] 2004, Jung, I.H., Chung, S.H. and Park, S.J., "A VIA-based RDMA Mechanism for High Performance PC Cluster Systems", *Journal of KIISE:Computer Systems and Theory*, Vol.31, pp.635-642.
[5] 2006, Shipman, G.M., Woodall, T.S., Graham, R.L. and Maccabe, A.B., "Infiniband Scalability in Open MPI", *Parallel and Distributed Processing Symposium*, pp.1-8.
[6] 2004, Liu, J, Wu, J. and Panda, D.K., "High Performance RDMA-based MPI Implementation over Infiniband", *International Journal of Parallel Programming*, Vol.32, pp.167-198.
[7] 2011, Mininni, P.D., Rosenberg, D., Reddy, R. and Pouquet, A., "A hybrid MPI-OpenMP Scheme for Scalable Parallel Pseudospectral Computations for Fluid Turbulence", *Parallel computing*, Vol.37, pp.316-326.
[8] http://mvapich.cse.ohio-state.edu
[9] http://www.open-mpi.org
[10] 2013, KAERI, CUPID code 1.7 manual, Vol.1, pp.46-53.
[11] http://glaros.dtc.umn.edu/kghome/views/metis
[12] 2014, Jeon, B.J., Lee, J.R., Yoon, H.Y. and Choi, H.G., "Performace Analysis of the Parallel CUPID Code for Various Parallel Programming Models in Symmetric Multi-Processing System", *Trans. Korean Soc. Mech. B*, Vol.38, pp.71-79.
[13] 2003, Kang, S.W., Choi, H.G. and Yoo, J.Y., "Parallel Preconditioner for the Discretized Navier-Stokes Equation", *Trans. Korean Soc. Mech. B*, Vol.27, pp.753-765.