이기종 클러스터를 위한 수정된 GSS 부하 분할 알고리즘

구 본 근†

요 약

클러스터는 컴퓨터 네트워크로 연결되어 있는 컴퓨터들로 구성된 비용대비 효과적인 병렬 처리 환경이다. 클러스터의 특징으로는 노드의 이기종성, 부하의 다양성, 네트워크 부하의 다양성 등이다. 이러한 특징들은 병렬 프로그램의 수행 성능에 영향을 주기 때문에 클러스터를 위한 부하 분할은 병렬 프로그램의 성능에 많은 영향을 준다. 본 논문에서는 부하 분할 알고리즘인 GSS를 수정한 aGSS 알고리즘을 제안한다. 본 논문에서 제안하는 aGSS 알고리즘에서는 각 노드가 처리할 태스크의 크기를 결정할 때 각 노드의 BogoMIPS를 이용한다. 실험 결과에 의하면 제안된 aGSS 알고리즘이 이기종으로 구성된 클러스터에서 효과적으로 부하를 분할하며, 따라서 병렬 프로그램의 수행 시간을 감소시킬 수 있다.

Adapted GSS Load Sharing Algorithm for Heterogeneous Cluster

Bongeun Goo†

ABSTRACT

Cluster is the cost-effective parallel processing environment, and consists of the off-the-shelf computers connected by the computer networks. The characteristics of cluster are the node heterogeneity, the variety of node load, and the variety of network load. Because these characteristics influence the performance of parallel program executions, the load sharing for cluster is important, and by using the proper load sharing strategy, we can reduce the execution time of parallel programs. In this paper, we propose modified GSS algorithm, aGSS. In the proposed load sharing algorithms aGSS, the size of tasks are decided using the BogoMIPS of node. From the result of out experiments, we conclude that the proposed aGSS algorithm is effective in the heterogeneous cluster.

키워드 : 클러스터(Cluster), 부하 분할(Load Sharing)

1. 서 론

클러스터 기술은 공학적 문제에 대한 시뮬레이션, 기상 예보 시스템, DNA 구조 모델링 등과 같은 강력한 계산 능력(computing power)을 필요로 하는 분야에서 비용 대비 효과적인 병렬 처리 능력을 제공한다[1]. 병렬 처리는 공유 메모리 시스템 등과 같은 병렬 컴퓨터에서 제공할 수 있지만, 병렬 컴퓨터의 높은 가격, 낮은 효용성 등으로 인해 최근에 클러스터를 이용한 병렬 처리에 많은 연구가 이루어지고 있으며, 많은 분야에서 이용되고 있다.

클러스터는 독자적으로 운영되는 여러 컴퓨터들의 집단으로 각 컴퓨터의 계산 능력과 저장 공간을 통합적으로 이용하여 병렬 처리를 하는 시스템이다. 클러스터를 구성하는 컴퓨터를 노드라고 하며, 각 노드는 병렬 처리를 위한 메시지 교환을 위해 ethernet, fast ethernet, Myrinet 등과 같은 컴퓨터 네트워크를 이용한다. 따라서, 클러스터는 다음과 같은 특징을 갖는다[2, 3].

- 노드의 이기종성 - 클러스터의 각 노드는 기종, 성능, 메모리 용량 등이 다를 수 있다.
- 노드 부하의 다양성 - 각 노드는 독자적으로 운용되므로 각 노드가 실행하고 있는 프로세스의 수와 상태 등이 같지 않다.
- 다양한 네트워크 부하 - 클러스터의 네트워크는 병렬 처리를 위한 메시지뿐만 아니라 각 노드에서 수행되는 프로세스에서 요구하는 일반적 목적의 메시지 전송을 위해 사용되므로 네트워크의 부하를 예측하기가 어렵다.

클러스터의 이러한 특징들로 인해 다양한 파라메터들이

† 정 회 원 : 충주대학교 컴퓨터공학과 교수
논문접수 : 2002년 10월 14일, 심사완료 : 2003년 6월 14일

클러스터에서 수행되는 병렬 프로그램의 수행 성능에 영향을 준다. 따라서, 병렬 프로그램의 수행 성능을 향상시키기 위하여 병렬 프로그램에서 수행해야 하는 태스크들을 각 노드에게 적절하게 할당하는 부하 분할(load sharing)은 클러스터에서 중요한 분야이다. 클러스터에서의 부하 분할과 관련한 최근의 연구는 작업(job) 단위의 부하 분할[4, 5]과 태스크 단위의 부하 분할[2, 6-8]이 있다.

작업 단위의 부하 분할은 클러스터에서 수행하는 작업의 수행 패턴을 이용하여 그 작업을 수행할 노드를 결정하는 것으로, 새로운 작업이 노드에 도착하면 노드의 시스템 지표(system metric)에 따라 작업을 노드에서 지역적으로 수행하거나, 적절한 다른 노드에서 원격 수행하는 것을 결정한다. 또 이 부하 분할 정책에 따라 노드에 도착한 새로운 작업을 노드에서 수행하기 위해 이미 노드에서 수행되고 있는 작업을 선택하여 다른 노드로 이주시키기도 한다[4, 5].

태스크 단위의 부하 분할은 클러스터의 각 노드에 할당된 프로세스가 처리할 태스크의 크기(granularity of tasks)를 결정하는 부하 균등(load balancing) 정책이다. 이러한 태스크 단위의 부하 분할 알고리즘은 주 노드-종속 노드 정책(master-slave strategy)을 기반으로 하는 병렬 프로그램에서 주 노드(master node)가 종속 노드(slave node)에서 처리할 태스크의 크기를 결정하는 방법에 따라 고정 태스크 크기(fixed task granularity), 가변 태스크 크기(variable task granularity), 적응적 태스크 크기(adaptive task granularity)로 분류할 수 있다. 고정 태스크 크기 알고리즘은 태스크 크기가 고정되어 있으며, 가변 태스크 크기는 태스크들이 처리되어 짐에 따라 점차적으로 태스크의 크기를 감소시켜 나가는 정책이다[2, 6]. 적응적 태스크 크기 알고리즘은 종속 노드의 상태에 따라 태스크 크기에 변화를 주는 부하 분할 알고리즘이다[2, 6]. 이러한 알고리즘을 비교, 분석한 기존의 연구에서는 클러스터를 구성하는 노드의 성능 차이가 적은 비교적 동기종에 가까운 클러스터를 대상으로 수행하였다[2, 6, 7]. 하지만, 노드의 성능과 구성에 많은 차이가 있는 이기종(heterogeneous) 클러스터인 경우에는 여러 형태의 병목 현상으로 인하여 전체 클러스터의 성능을 감소시킬 수 있다.

본 논문에서는 이기종의 노드들로 구성된 클러스터에서 고정 태스크 크기 알고리즘인 Send 알고리즘과 가변 태스크 크기 알고리즘인 GSS의 문제점을 기술하고, 이를 해결하고 전체 클러스터의 성능을 향상시키 위해 각 노드의 처리 능력을 기준으로 하여 태스크 크기를 결정하는 수정된 GSS 부하 분할 알고리즘인 aGSS 부하 분할 알고리즘을 제안하고자 한다. 본 논문에서 제안하는 aGSS(Adpated GSS) 알고리즘의 성능을 비교, 평가하기 위해 720×720 크기의 행렬 곱셈을 수행하는 병렬 프로그램을 작성하였으며, 각각 Send, GSS 알고리즘과 비교, 분석하였다.

본 논문의 구성은 다음과 같다. 제 2장에서는 부하 분할 알고리즘에 대한 기존의 연구들 중에서 고정 태스크 크기 알고리즘과 가변 태스크 크기 알고리즘을 소개하며, 제 3장에서는 이기종의 노드들로 구성된 클러스터에서 고정 태스크 크기 알고리즘과 가변 태스크 크기 알고리즘의 문제점을 기술한다. 그리고, 제 4장에서는 이기종의 노드들로 구성된 클러스터에서 성능 향상을 위해 GSS 알고리즘을 개선한 aGSS 알고리즘을 제안하며, 그 성능을 기존의 알고리즘과 비교, 분석하며, 제 5장에서 결론과 추후 연구 과제에 대해 기술한다.

2. 부하 분할 알고리즘

병렬 프로그램에서 처리할 전체 태스크의 크기를 S라고 하고, 클러스터의 노드의 수가 N이라고 할 때, 각 노드가 처리할 태스크의 크기 t를 S/N으로 할 경우에는 부하 분할 알고리즘은 필요가 없다. 이러한 방법의 태스크 크기 결정은 클러스터를 구성하는 노드의 상태에 따라 병렬 프로그램의 수행 성능을 감소시킬 수 있다. 이기종으로 구성될 수 있다는 클러스터의 특성상 낮은 성능의 노드에 할당된 크기 $t(=S/N)$인 태스크를 처리하는데 많은 시간이 필요하며, 클러스터 전체적으로 볼 때 수행 성능은 감소된다. 이를 해결하기 위한 방법으로 작은 크기의 태스크를 각 노드가 반복적으로 처리하도록 하여 낮은 성능의 노드에 의한 전체 클러스터의 성능 감소를 해소할 필요가 있다. 이를 위해 부하 분할 알고리즘이 필요하다.

부하 분할 알고리즘을 비교, 평가한 연구에서 부하 분할 알고리즘은 고정 태스크 크기, 가변 태스크 크기, 적응적 태스크 크기 알고리즘으로 분류하였다. Piotrowski의 연구[2]에서는 반응 시간을 기준으로 한 적응적 태스크 크기 알고리즘이 다른 두 종류의 알고리즘 보다 낮은 성능을 보인다는 실험 결과에 대해 기술하고 있으므로 이 장에서는 본 논문에서 주요 비교 대상으로 정한 고정 태스크 크기, 가변 태스크 크기 알고리즘에 대해 기술한다.

2.1 고정 태스크 크기 알고리즘

고정 태스크 크기 알고리즘은 병렬 프로그램을 수행하는 종속 노드에서 처리하는 태스크 크기가 미리 정해져 있으며, 이 크기에 따라 부하 분할을 하는 알고리즘이다. Send 알고리즘은 주 노드에서 고정된 크기의 태스크를 종속 노드에게 전송을 한 후 종속 노드로부터 그 처리 결과를 수신할 때까지 대기하였다가 처리 결과를 수신하면 다음 태스크를 종속 노드에게 전송하는 알고리즘으로 미리 정해져

있는 태스크 크기를 F라고 할 때, 부하 분할을 위한 태스크의 크기 t는 다음과 같이 결정된다.

$$t = F$$

2.2 가변 태스크 크기 알고리즘

가변 태스크 크기 알고리즘은 부하 분할 알고리즘에 의해 결정되는 태스크 크기가 시간에 따라 감소되며, 이것은 남아 있는 태스크 크기에 따라 태스크 크기를 감소시킴으로써 클러스터의 주노드가 마지막 결과를 종속 노드로부터 전송받기 위해 기다려야 하는 시간을 감소시킨다. 이러한 가변 태스크 크기 알고리즘에는 GSS(Guided Self-Scheduling), Factoring, TSS(Trapezoidal Self-Scheduling) 등이 있다.

Piotrowski의 연구에서는 이들 알고리즘 중에서 GSS가 Factoing과 TSS보다 높은 성능을 보인다는 실험 결과를 기술하고 있으므로 이 절에서는 본 논문에서 주요 비교 대상인 GSS에 대해 기술한다.

GSS는 남아 있는 태스크 크기에 대한 일정한 비 $1/G$를 이용하여 태스크 크기 t를 다음과 같이 결정한다. t_{remain}을 남아있는 태스크 크기라고 할 때 태스크 크기 t는 다음과 같이 결정된다.

$$t = t_{remain} \times \frac{1}{G}$$

3. 이기종 클러스터에서 Send와 GSS 알고리즘

비교적 성능의 차이가 적은 펜티엄과 펜티엄 Pro 시스템으로 구성된 환경에서 수행된 Piotrowski의 부하 분할 알고리즘의 비교 연구에서는 낮은 성능을 갖는 노드에서의 처리 지연이 전체 클러스터의 성능에 영향을 적게 주거나, 주지 않는다. 하지만 성능과 구성에 있어 많은 차이가 있는 노드들로 구성된 클러스터인 경우에는 낮은 성능을 갖는 노드에서의 처리 지연이 클러스터 성능에 많은 영향을 준다. 이 장에서는 성능과 구성에 있어 많은 차이가 있는 노드들로 구성된 이기종 클러스터에서 Send와 GSS 알고리즘에 의해 태스크 크기를 결정할 때에 발생하는 문제점을 기술한다. 이를 위해 본 논문에서는 여러 병렬 처리 응용분야에서 많이 사용되는 연산 중의 하나인 행렬 곱셈을 수행하는 병렬 프로그램을 작성하여 그 수행 시간을 비교, 분석한다.

3.1 실험 환경

실험에서는 720×720 크기의 두 개의 행렬을 곱셈을 수행하는 병렬 프로그램을 작성하였으며, 실험에 사용한 이기종 클러스터 환경은 일반적인 10Mbps의 속도를 갖는 LAN 환경에 다섯 대의 서로 다른 CPU와 구성을 갖는 노드들로 구성하였다. 이들 노드들 중 주 노드로 사용한 노드는 펜티엄 4(1.5Ghz) 프로세서와 256MB의 주기억장치로 구성되어 있는 일반적인 PC이고, 종속 노드로 사용할 네 대의 구성은 각각 다음과 같다. 종속 노드 1은 펜티엄 4(1.5Ghz) 프로세서와 256MB의 주기억장치, 종속 노드 2는 펜티엄 3(450Mhz) 프로세서와 256MB의 주기억장치, 종속 노드 3은 펜티엄 2 (256Mhz) 프로세서와 128MB의 주기억장치, 종속 노드 4는 선마이크로시스템의 스팍 프로세서와 64MB의 주기억장치로 구성되어 있다. 이들 노드들은 공개용 운영체제인 리눅스를 운영체제로 하고 있으며, 클러스터를 위한 전용의 노드가 아닌 웹서버, NFS 서버 및 클라이언트, Samba 서버, 학생 실습을 위한 telnet 서버, ftp 서버 등과 같은 일반적인 서비스도 함께 제공하고 있다.

또, 작성된 병렬 프로그램은 노드간의 통신을 위해 PVM 라이브러리를 사용하였으며, 노드간 통신의 패턴을 확인하기 위해 PVM 패키지에 포함되어 있는 xpvm 프로그램을 사용하였다.

3.2 이기종 클러스터에서의 Send 알고리즘

본 연구를 위해 구축한 클러스터 환경에서 send 알고리즘은 곱셈을 할 두 개의 720×720 행렬 중 첫 번째 행렬의 전체 데이터를 주 노드가 각 종속 노드에게 전송한 뒤 결정된 태스크의 크기에 따라 행렬 곱셈을 위한 두 번째 행렬의 서브 행렬을 종속 노드에게 전송한다. 예를 들어, 태스크의 크기가 1이라면 주 노드는 두 번째 행렬에서 720×1의 크기를 갖는 서브 행렬을, 태스크 크기가 16이라면 720×16의 크기를 갖는 서브 행렬을 종속 노드에게 전송하여 행렬 곱셈을 수행하도록 한다.

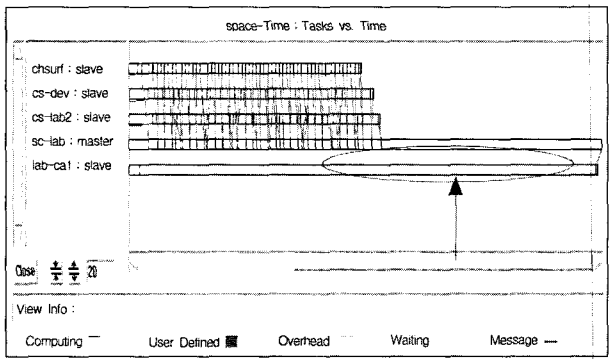
본 논문에서는 고정된 태스크의 크기를 1, 2, 4, 8, 16, 32인 경우에 대하여 각각 20회씩 수행하여 그 수행 시간의 평균을 이용하여 비교, 평가하였다. 병렬 프로그램을 실행하는 시간대에 따른 실험 결과에 미치는 영향을 감소시키기 위해 태스크 크기 결정 방법에 따라 각 알고리즘들을 -뒤에서 기술하는 GSS 알고리즘과 aGSS 알고리즘을 포함하여- 번갈아 가며 각각의 태스크 크기 또는 파라미터를 이용하여 20회 반복 수행하였다. 각 경우에 대한 평균 실행 시간은 <표 1>과 같다. <표 1>에서 통신의 수는 태스크 크기에 따라 주 노드가 종속 노드에게 서브 행렬을 전송하는 전송의 수를 나타낸다.

<표 1>에서 보인 실험 결과에 의하면 태스크 크기 2를 기준으로 그 크기가 증가할수록 평균 수행 시간이 증가됨을 보이고 있다. 이것은 클러스터를 구성하고 있는 노드들 중에서 상대적으로 성능이 낮은 노드에서의 처리 지연이 전

〈표 1〉 Send 알고리즘을 이용한 행렬 곱셈 수행 시간

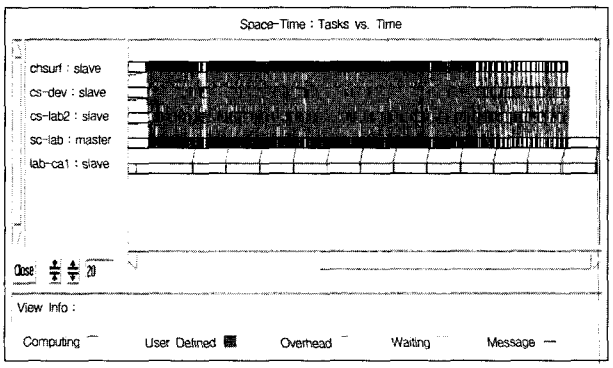
태스크 크기	평균 수행 시간(초)	전송의 수
1	18.9	720
2	18.7	360
4	19.3	180
8	19.5	90
16	28.5	45
32	46.3	23

체 클러스터의 성능에 영향을 줌을 보이고 있다. 이러한 낮은 성능을 갖는 노드에서의 처리 지연은 태스크 크기가 16인 경우에 병렬 프로그램의 수행 패턴를 나타낸 (그림 1)에서 화살표로 표시한 부분에서 확인할 수 있다.



(그림 1) t = 16인 경우의 수행 패턴

(그림 1)에서 왼쪽 부분에 표시한 각 노드의 이름들 중에서 종속 노드 *chsurf*가 클러스터를 구성하는 노드들 중에서 성능이 가장 높으며, 종속 노드 *lab-ca1*가 성능이 가장 낮다.



(그림 2) t = 1인 경우의 수행 패턴

태스크 크기가 1인 경우에 태스크 크기가 2인 경우보다 평균 수행 시간이 긴 것은 주 노드에서 종속 노드에게 데이터를 전송하는 전송의 수가 많아 전송에 따른 지연이 병렬 프로그램의 수행 시간에 영향을 준 것으로 판단된다. 전송의 수가 많은 경우에 병렬 프로그램의 수행 패턴은 (그림 2)에서 나타낸 것과 같다. (그림 2)에서 나타낸 수행 패턴은 병렬 프로그램 수행을 위한 프로세스 생성 등과 같은 전처리를 완료한 후 종속 노드에서 실제 계산을 수행하는 것만을 나타내었다.

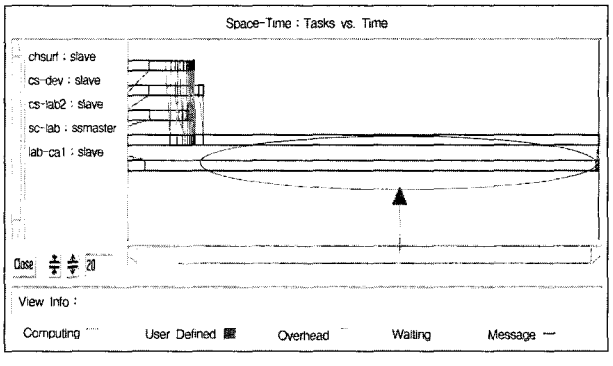
3.3 이기종 클러스터에서의 GSS 알고리즘

GSS 알고리즘에서는 남아있는 태스크에 대한 일정한 비 ($\frac{1}{G}$)를 이용하여 태스크 크기를 결정한다. 본 논문에서는 두 개의 720×720 행렬 곱셈을 위한 G값으로 4, 8, 16, 32, 64, 128을 사용하여, 각각 앞에서 기술한 방법으로 20회씩 수행하여 그 수행 시간을 측정하였다. 각 경우에 대한 평균 수행 시간은 <표 2>와 같다.

GSS에서 G값이 4인 경우에 본 논문에서 실험한 다른 경우에 비해 주 노드에서 종속 노드로의 데이터 전송의 수는 적지만 태스크 크기가 상대적으로 크기 때문에 성능이 낮은 노드에서의 지연이 전체 클러스터의 성능에 심각한 영향을 준다. 이 경우에 병렬 프로그램의 수행 패턴은 (그림 3)에서 표시한 것과 같다. (그림 3)에서 화살표로 표시한 부분이 전체 클러스터의 성능에 심각한 영향을 주는 낮은 성능의 종속 노드에서의 처리 지연을 나타내고 있다.

〈표 2〉 GSS 알고리즘을 이용한 행렬 곱셈 수행 시간

G	평균 수행 시간(초)	전송의 수
4	97.0	20
8	78.5	38
16	52.0	68
32	33.0	116
64	21.6	193
128	18.6	304

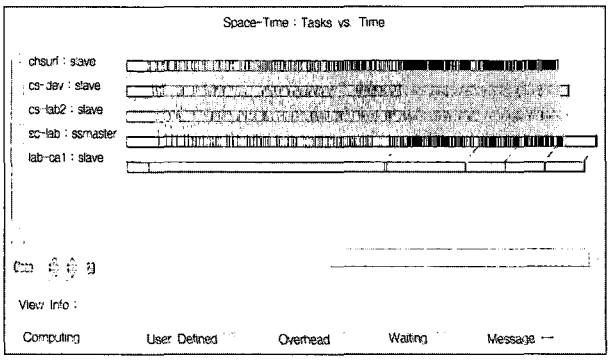


(그림 3) G = 4인 경우의 수행 패턴

GSS에서 G의 값이 큰 경우에는 상대적으로 결정된 태스크 크기가 작아 낮은 성능의 노드에서의 지연으로 인한 전체 클러스터 성능 저하에 상대적으로 적은 영향을 주지만 태스크 크기 결정이 효율적이지 않음을 알 수 있다. (그림

4)는 G의 값이 128일 때 병렬 프로그램의 수행 패턴을 보이고 있다.

본 논문에서의 실험 결과가 Piotrowski의 연구 결과와는 다르게 Send 알고리즘보다 GSS 알고리즘을 적용한 병렬 프로그램의 평균 수행 시간이 긴 것은 다양한 성능과 구성을 갖는 이기종의 노드들로 구축된 클러스터인 경우에 GSS 알고리즘이 태스크를 각 노드에 효과적으로 분산시키지 못하며, 따라서 전체 클러스터의 성능을 저하시키는 것으로 판단된다. 이런 결과의 원인은 GSS가 클러스터의 각 노드의 성능을 고려하지 않고 G값에 의해서만 태스크 크기를 결정하기 때문이다.



(그림 4) G = 128인 경우 수행 패턴

4. 수정된 GSS 부하 분할 알고리즘 : aGSS

4.1 aGSS 알고리즘

고정 부하 분할 알고리즘인 Send 알고리즘에서 태스크 크기를 작게 한 경우에는 주 노드와 종속 노드 사이의 메시지 전송의 수가 많아지므로 네트워크의 상태에 따라 수행 시간의 편차가 증가하며, 태스크 크기를 크게 한 경우에는 낮은 성능을 갖는 노드에서의 처리 지연으로 인하여 클러스터 성능이 떨어진다. 또, 가변 부하 분할 알고리즘인 GSS는 파라미터 G의 값에 의해서 태스크 크기를 결정하므로, G의 값이 큰 경우에는 결정되는 태스크의 크기가 상대적으로 작으므로 낮은 성능의 노드에서 처리 지연이 클러스터의 성능에 미치는 영향이 적지만 전송의 수가 증가한다. 또, G의 값이 작으면 태스크의 크기가 상대적으로 증가하므로 낮은 성능의 노드에서의 처리 지연이 클러스터 성능에 미치는 영향이 크다. 이것은 Send와 GSS 알고리즘이 노드의 성능을 고려하지 않고 정해진 태스크 크기 또는 파라미터에 의해 태스크 크기를 결정함으로 부하를 클러스터의 모든 노드에 적절히 분산시키기 못하기 때문이다.

본 논문에서는 Send와 GSS 알고리즘의 이러한 문제점을 해결하기 위해 GSS 알고리즘에서 이용한 파라미터 G에 의해 결정된 태스크 크기를 노드의 성능을 고려하여 재조정함으로써 낮은 성능의 노드에는 상대적으로 적은 태스크를 할당하여 낮은 성능의 노드에서의 처리 지연으로 인한 전체 클러스터의 성능 감소를 해결하고자 한다. 이 장에서는 노드의 성능에 따라 G에 의해 결정된 태스크 크기를 재조정하는 수정된 GSS(Adapted GSS, aGSS) 알고리즘을 제안하고, 두 개의 720×720 행렬을 곱하는 병렬 프로그램의 수행 시간 비교를 통해 aGSS 알고리즘이 효과적으로 부하를 분할함을 보인다.

4.1.1 노드의 성능 지표 : BogoMIPS

클러스터를 구성하고 있는 노드의 성능을 평가하기 위한 성능 지표로는 CPU 처리 능력, 주기억장치의 용량, 시스템 부하 등이 있다. 최근에 일반적으로 판매되는 컴퓨터는 반도체 메모리의 가격 하락과 집적도의 증가 등으로 인하여 주기억장치가 수십 MB에서 수백 MB의 기억용량을 가지고 있으며, 또한 운영체제의 효율적인 페이징 정책 등으로 인하여 많은 작업을 효율적으로 처리하고 있다. 이러한 컴퓨터로 구성한 클러스터도 작업단위의 부하 분할을 할 경우에 주기억장치의 용량 또는 운영체제의 페이징 정책이 클러스터의 성능에 영향을 준다. 하지만, 본 논문에서 고려하는 태스크 단위의 부하 분할의 경우에는 종속 노드에 전송되어 처리되는 태스크의 크기가 비교적 적기 때문에 클러스터의 성능에 주기억장치의 용량이 미치는 영향은 비교적 적다.

또 각 노드의 시스템 부하도 클러스터의 성능에 비교적 적은 영향을 준다. 이것은 각 노드가 작동을 하는 대부분의 시간을 10% 이하의 CPU 사용율을 보이며, 보통의 노드에 존재하는 많은 프로세스 중에서 실행 상태 또는 준비 상태에 있는 프로세스의 수는 많지 않다. 본 논문에서 구축한 클러스터에 포함된 노드의 경우 웹서버, NFS 서버, telnet 서버, ftp 서버 등의 서비스를 제공하지만 클러스터의 동작에 많은 영향을 주지 않았다. 따라서, 본 논문에서는 노드의 성능을 평가하기 위한 성능 지표로 CPU의 처리 능력만을 사용하였다.

본 논문에서는 CPU의 처리 능력을 위한 성능 지표로 리눅스가 제공하는 BogoMIPS를 이용하였다. 이 BogoMIPS는 CPU의 종류와 동작 클럭 속도에 따라 리눅스 운영체제가 계산하여 제공하는 것으로 실질적인 CPU의 처리 능력을 나타내지는 못한다. 하지만, 이기종으로 구성된 클러스터에서 각 CPU의 상대적 처리 능력을 비교할 경우에는 충분한 가치가 있다. 예를 들어, 본 논문에서 구축한 클러스터의 노드들에 사용된 CPU중 가장 높은 성능의 펜티엄 4(1.5GHz)는 약 2994 BogoMIPS이며, 펜티엄 3(450MHz)은 약 897 BogoMIPS, 펜티엄2(256MHz)는 약 529 BogoMIPS이다. 그리고, 가장 낮은 성능을 갖는 스팍 프로세서는 약

69 BogoMIPS이다.

4.1.2 αGSS 알고리즘의 태스크 크기 결정

αGSS 알고리즘은 기본적으로 GSS 알고리즘과 동일한 방법으로 동작을 하지만 태스크 크기 t는 노드의 CPU 성능 지표로 사용한 BogoMIPS에 따라 다음과 같이 결정한다.

$$t = t_{remain} \times \frac{1}{G} \times \frac{BogoMIPS_{node}}{BogoMIPS_{average}}$$

t_{remain}은 남아 있는 태스크의 크기이며, G는 αGSS 알고리즘을 위한 파라미터로 GSS 알고리즘의 파라미터 G와 같은 의미를 갖는다. $BogoMIPS_{average}$는 클러스터를 구성하는 모든 노드들의 BogoMIPS 평균을 나타내며, $BogoMIPS_{node}$는 태스크를 할당받으려는 노드의 BogoMIPS를 나타낸다. 각 종속 노드의 BogoMIPS 정보는 주 노드가 종속 노드에서 수행할 프로세스를 생성하면, 각 종속 노드에서 수행되는 프로세스가 지역 노드의 BogoMIPS를 주 노드에서 전송을 함으로써 얻을 수 있다. 주 노드는 이를 이용하여 클러스터의 평균 BogoMIPS를 계산하고, 각 노드를 위한 태스크의 크기를 결정한다.

αGSS 알고리즘의 이러한 태스크 크기 결정 방법은 태스크 크기를 결정할 때 각 노드의 상대적 CPU 성능을 이용하여 성능이 높은 노드에는 비교적 많은 태스크를 할당하며, 성능이 낮은 노드에는 비교적 적은 태스크를 할당한다. 따라서, αGSS에서는 높은 성능을 갖는 노드에는 많은 태스크를 할당하여 전체 처리 시간을 감소시키며, 낮은 성능의 노드에는 적은 태스크를 할당함으로써 낮은 성능의 노드에서의 처리 지연이 전체 클러스터의 성능 저하에 영향을 주는 것을 감소시키고자 하였다. 동기종(homogeneous) 클러스터인 경우에는 모든 노드의 CPU 성능이 동일하므로 αGSS는 GSS와 같은 크기의 태스크를 각 종속 노드에게 할당한다.

4.2 실험 결과 및 분석

본 논문에서 제안한 αGSS의 성능을 이기종 클러스터 상에서 Send, GSS 알고리즘의 성능과 비교하기 위해 앞에서 기술한 실험 환경에서 720×720 크기의 두 행렬 곱셈을 수행하는 병렬 프로그램을 작성하였다. 본 논문에서 αGSS를 위한 파라미터 G의 값으로 4, 8, 16, 32, 64, 128을 사용하여, 각각 20회씩 αGSS 알고리즘을 사용하는 병렬 프로그램을 수행하여, 그 수행 시간의 평균을 계산하였다. 각 경우에 대한 평균 수행 시간은 <표 3>과 같다.

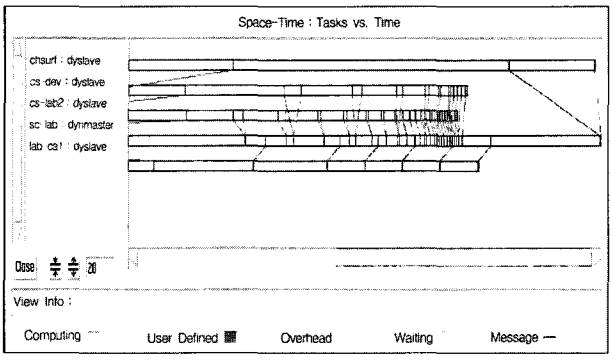
<표 3>의 수행 시간에 의하면 αGSS 알고리즘은 파리미터 G의 값에 따른 수행 시간의 편차가 Send 알고리즘과 GSS 알고리즘의 편차보다 작다. 이것은 αGSS 알고리즘이 부하 분할을 위한 파라미터를 결정하는 것이 다른 알고리즘에 비해 자유롭다는 것을 의미한다. 즉, Send와 GSS의 경우 가장 짧은 수행 시간을 보이는 파라미터를 찾기 위한 시도가 필요하지만 αGSS 알고리즘은 적당한 파라미터 값을 이용하여 병렬 프로그램을 수행하더라도 가장 짧은 수행 시간과 많은 차이가 나지 않는다.

〈표 3〉 αGSS 알고리즘을 이용한 행렬 곱셈 수행 시간

G	평균 수행 시간(초)
4	22.3
8	20.1
16	19.4
32	18.9
64	18.9
128	18.7

αGSS의 이러한 결과는 다른 알고리즘에 비해 각 종속 노드에 효과적으로 태스크를 배분하며, 이를 통해 낮은 성능의 노드에서는 적은 태스크를 처리하게 하여 낮은 성능의 노드에서의 처리 지연으로 인한 클러스터의 성능 저하를 방지함과 동시에 높은 성능의 노드에는 많은 태스크를 처리하도록 하여 전체 클러스터의 성능을 향상시킨다.

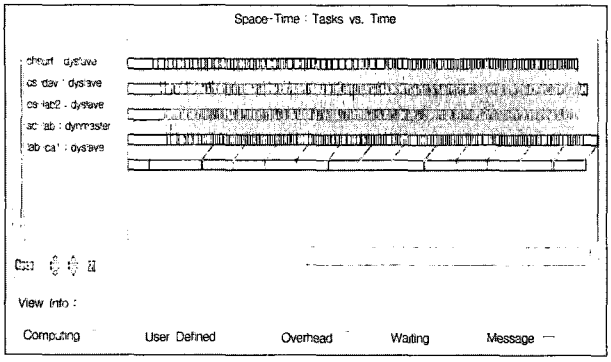
(그림 5)는 αGSS의 수행 시간 중에 가장 긴 수행 시간을 가진 G = 4의 경우에 병렬 프로그램의 수행 패턴을 나타낸 것이다. (그림 5)에 의하면 Send와 GSS 알고리즘과 달리 높은 성능의 노드가 많은 양의 태스크 처리를 담당함으로써 낮은 성능의 노드가 전체 클러스터의 성능에 병목이 되지 않음을 확인할 수 있다. 또, 결정되는 태스크의 크기가 크기 때문에 주 노드에서의 종속 노드로의 전송 수는 작다.



(그림 5) G = 4인 경우 aGSS의 수행 패턴

(그림 6)은 G = 128인 경우에 αGSS 알고리즘을 이용하는 병렬 프로그램의 수행 패턴을 나타낸 것이다. 이 경우 G 값에 의한 초기 태스크 크기가 작기 때문에 CPU 성능에 의한 태스크 크기 조정의 효과가 작다. 하지만, 태스크 크기가 작기 때문에 낮은 성능의 노드에서의 처리 지연으로

인한 병목 현상이 클러스터의 성능에 영향을 주지 않음을 보이고 있다.



(그림 6) G = 128인 경우 aGSS의 수행 패턴

5. 결 론

최근의 컴퓨터 기술은 아주 빠른 속도로 발전하고 있으며, 이러한 기술의 발전으로 인해 컴퓨터는 저가격, 고성능을 이룰 수 있었다. 이러한 저가격, 고성능 컴퓨터는 강력한 계산 능력을 요구하는 다양한 분야에서 활용되었다. 하지만, 많은 컴퓨터는 대부분의 시간을 휴지 상태로 있거나 낮은 시스템 부하를 유지하고 있어 컴퓨터를 강력한 계산 능력을 활용하지 못하고 있다.

공학적 문제에 대한 시뮬레이션, 기상 예보 시스템, DNA 구조 모델링 등과 같은 분야의 문제는 현재의 컴퓨터 기술이 제공할 수 있는 계산 능력보다 더욱 강력한 계산 능력이 요구된다. 이러한 문제를 적절한 시간 내에 해결하기 위해 여러 개의 프로세서를 이용하여 문제를 해결하는 병렬 처리 기술을 이용한다. 공유 메모리 시스템 등과 같은 병렬 컴퓨터는 병렬 처리를 위한 전용의 하드웨어, 소프트웨어 등을 이용하므로 효율적으로 병렬 처리를 할 수 있다. 하지만, 병렬 컴퓨터의 높은 가격, 낮은 효용성 등으로 인하여 최근에는 클러스터를 이용한 병렬 처리에 많은 연구와 이용이 이루어 지고 있다.

클러스터는 대부분의 시간을 휴지 상태로 있거나 낮은 시스템 부하를 갖는 저가격의 고성능 컴퓨터들을 컴퓨터 네트워크로 연결하여 이용되지 않고 있는 컴퓨터의 계산 능력을 이용하여 병렬 처리를 수행하는 비용 대비 효과적인 병렬 처리 시스템이다. 이러한 클러스터의 특징으로는 노드의 이기종성, 노드 부하의 다양성, 다양한 네트워크 부하 등이 있으며, 이러한 특징으로 인하여 각 노드가 처리할 태스크의 크기를 적절히 결정하여 전체적으로는 병렬 프로그램의 수행 성능을 높이는 부하 분할 알고리즘이 필요하다.

본 논문에서는 노드의 성능에 따라 태스크의 크기를 결정하는 수정된 부하 분할 알고리즘 aGSS를 제안하였다. 이 알고리즘은 GSS와 같이 파라미터 G에 의해 태스크 크기를 결정한 후 각 노드의 CPU 성능에 따라 태스크 크기를 재조정한다. 본 논문에서 제안한 aGSS 부하 분할 알고리즘이 부하를 각 종속 노드에 적절히 분산시키는 데 효과가 있음을 보이기 위해 서로 다른 성능을 갖는 다섯 대의 노드를 이용하여 이기종 클러스터를 구축하였으며, 각각 Send, GSS 부하 분할 알고리즘과 비교, 분석하였다. 각 알고리즘을 비교하기 위한 두 개의 720×720 행렬을 곱하는 병렬 프로그램을 작성하여 그 수행 시간을 이용하였다.

비교 분석의 결과로는 Send와 GSS 알고리즘은 낮은 성능을 갖는 노드에서의 처리 지연으로 인한 클러스터의 성능 감소가 있지만, 본 논문에서 제안한 aGSS 알고리즘은 각 종속 노드에 태스크를 적절하게 분할함으로써 낮은 성능을 갖는 노드의 처리 지연이 클러스터의 성능에 영향을 주는 정도를 최소화하였다.

또, Send와 GSS의 경우 고정 태스크의 크기나 파라미터 G의 값에 따른 클러스터 상에서의 병렬 프로그램 수행 시간의 변화가 심하지만, 본 논문에서 제안한 aGSS의 경우에는 파라미터 G의 변화에 대한 수행 시간의 변화가 심하지 않아 최적의 수행 시간을 얻기 위한 파라미터 결정에 있어 상대적으로 자유롭다.

추후 연구 과제로는 노드의 CPU 성능뿐만 아니라 태스크 크기 결정에 사용할 수 있는 노드의 성능 지표에 대한 연구와 함께 적용 방법에 대한 계속적인 연구가 필요하다.

참 고 문 헌

[1] B. Wilkinson and M. Allen, "Parallel Programming : Technique and Applications Using Networked Workstations and Parallel Computers," Prentice Hall, 1999.

[2] A. Piotrowski and S. Dandamudi, "A Comparative Study of Load Shaing on Networks of Workstations," Proc. Int. Conf. Parallel and Distributed Computing System, New Orleans, Oct., 1997.

[3] T. Anderson, D. Culler, D. Patterson and the NOW team, "A Case for NOW," IEEE Micro, Vol.15, No.2, pp.54-64, Feb., 1995.

[4] L. Xiao, S. Chen and X. Zhang, "Dynamic Cluster Resource Allocations for jobs with Known and Unknown Memory Demands," IEEE Tr. On Parallel and Distributed Systems, Vol.13, No.3, pp.223-240, 2002.

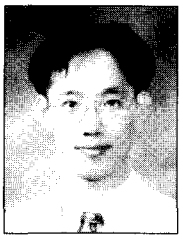
[5] L. Xiao, X. Zhang and Y. Qu, "Effective Load Sharing on Heterogenous Networks of Workstations," Proc. International Parallel and Distributed Processing Symposium(IP

DPS '2000), May, 2000.

[6] A. Piotrowski and S. Dandamudi, "Performance of a Parallel Application on Network of Workstations," 11th Int. Symp. High Performance Computing Systems, Winnipeg, pp.429-440, July, 1997.

[7] 구본근, "NOW 환경에서 개선된 고정 분할 단위 알고리즘", 정보처리학회논문지A, 제8-A권 제2호, pp.117-124, 2001.

[8] 구본근, "NOW 환경에서 preSend 알고리즘의 Java 구현", 충주대학교논문지, 제36집 제2호, 2001.



구 본 근

e-mail : bggoo@gukwon.chungju.ac.kr

1991년 인제대학교 전산학과(이학사)

1993년 부산외국어대학교 컴퓨터공학과 (공학석사)

1998년 경북대학교 컴퓨터공학과(공학박사)

1998년~2000년 충주대학교 컴퓨터공학과 전임강사

2000년~현재 충주대학교 컴퓨터공학과 조교수

관심분야 : 컴퓨터구조, 클러스터, 그리드 컴퓨팅, 병렬처리 시스템