이기종 클러스터를 위한 수정된 GSS 알고리즘

구본근*

*충주대학교

Modified GSS Algorithm for Heterogeneous Cluster

Bon-geun Goo*

*Chungju National University

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

요 약

클러스터는 컴퓨터 네트워크로 연결되어 있는 컴퓨터들로 구성된 비용대비 효과적인 병렬 처리 환경이다. 클러스터의 특징으로는 노드의 이기종성, 로드의 다양성, 네트워크 로드의 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.

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

cluser, load sharing, heterogeneous, load balancing

Ⅰ. 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 노드의 이기종성, 노드 부하의 다양성, 다양한 네트워크 부하와 같은 특징을 갖는다[2,3].

클러스터의 이러한 특징들로 인해 클러스터에서 수행되는 병렬 프로그램의 수행 성능에는 다양한 파라메

터들이 영향을 준다. 따라서, 병렬 프로그램의 수행 성능을 향상시키기 위해 병렬 프로그램에서 수행해야 하는 태스크들을 각 노드에게 적절하게 분할하는 부하 분할(load sharing)은 클러스터에서 중요한 분야이다.

본 논문에서는 이기종의 노드들로 구성된 클러스터에서 가변 태스크 크기 알고리즘인 GSS의 문제점을 기술하고, 이를 해결하기 위해 GSS의 알고리즘을 수정하여 각 노드의 처리 능력을 기준으로 하여 태스크 크기를 결정하여 전체 클러스터의 성능을 향상시키는 적응적 부하 분할 정책인 aGSS 알고리즘을 제안하고자 한다. 본 논문에서 제안한 aGSS(Adpative GSS) 알고리즘의 성능을 비교하기 위해 720x720 크기의 행렬 곱셈을 수행하는 병렬 프로그램을 작성하였으며, GSS 알고리즘과 비교, 분석하였다.

본 논문의 구성은 다음과 같다. 제 2장에서는 이기종의 노드들로 구성된 클러스터에서 가변 태스크 크기 알고리즘의 문제점을 기술하고, 이러한 환경하에서 클러스터에서 성능 향상을 위해 GSS 알고리즘을 수정한 aGSS 알고리즘을 제안한다. 제 3장에서 aGSS 알고리즘의 성능을 기존의 알고리즘과 비교하여 그 효과를 기술하며, 제 4장에서 결론과 추후 연구 과제에 대해 기술한다

II. GSS알고리즘과 aGSS알고리즘

1. 실험환경

이 장에서는 이기종의 노드들로 구성된 클러스터에서 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.

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 펜티엄4(1.5Ghz) 프로세서와 256MB의 주기억장치로 구성되어 있는 일반적인 PC이고, 종속 노드로 사용할 네 대의 구성은 각각 다음과 같다. 종속 노드1은 펜티엄4(1.5Ghz) 프로세서와 256MB의 주기억장치, 종속 노드2는 펜티엄3(450Mhz) 프로세서와 256MB의 주기억장치, 종속 노드3은 펜티엄2(256Mhz) 프로세서와 128MB의 주기억장치, 종속 노드4는 선마이크로시스템의 스팍 프로세서와 64MB의 주기억장치로 구성되어 있다. 이들 노드들은 리눅스를 운영체제로 하고 있다.

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

2. GSS 알고리즘

GSS 알고리즘에서 태스크 크기는 남아있는 태스크에 대한 일정한 비($\frac{1}{G}$)를 이용하여 결정된다. 본 논문에서는 두 개의 720×720 행렬 곱셈을 위한 G 값으로 4, 8, 16, 32, 64, 128을 사용하여, 각각 5회씩 수행하여 그 수행 시간을 측정하였다. 각 경우에 대한 평균 수행 시간은 표 1과 같다.

GSS에서 G 값이 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 1에서 표시한 것과 같다. 그림 1에서 원으로 표시한 부분이 전체 클러스터의 성능에 심각한 영향을 주는 낮은 성능의 종속 노드에서의 처리 지연을 나타내고 있다.

표 1. GSS을 이용한 행렬 곱셈 수행 시간

G	평균 수행 시간(초)	전송의 수
4	97.23	20
8	78.81	38
16	52.37	68
32	33.32	116
64	21.87	193
128	19.17	304

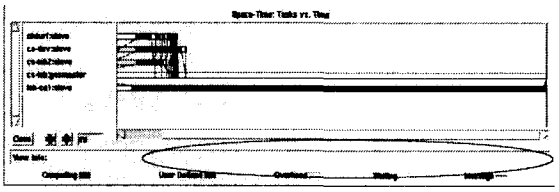

그림 1. G=4인 경우의 수행 패턴

3. aGSS 알고리즘

가. 노드의 성능 지표 : BogoMIPS

클러스터를 구성하고 있는 노드의 성능을 평가하기 위한 성능 지표로는 CPU 처리 능력, 주기억장치의 용량, 시스템 부하(load) 등이 있다. 본 논문에서 고려하

는 태스크 단위의 부하 분할의 경우에는 종속 노드에 전송되어 처리되는 태스크의 크기가 비교적 적기 때문에 주기억장치의 용량이 클러스터의 성능에 비교적 적은 영향을 주는 것으로 판단된다.

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, NFS서버, Samba 서버 등의 서비스를 제공함에도 불구하고 클러스터의 동작에 많은 영향을 주지는 않았다. 따라서, 본 논문에서는 노드의 성능을 평가하기 위한 성능 지표로 CPU의 처리 능력만을 사용하였다.

본 논문에서는 CPU의 처리 능력을 리눅스 운영체제가 제공하는 BogoMIPS를 이용하였다. 이 BogoMIPS는 CPU의 종류와 동작 클럭 속도에 따라 리눅스 운영체제가 계산하여 제공하는 것으로 실질적인 CPU의 처리 능력을 나타내지는 못한다. 하지만, 각 CPU의 상대적 처리 능력을 비교할 경우에는 충분한 가치가 있다. 예를 들어, 본 논문에서 구축한 클러스터의 노드들에 사용된 CPU중 가장 높은 성능인 펜티엄4는 약 2994 BogoMIPS이며, 가장 낮은 성능인 스팍 프로세서는 약 69 BogoMIPS이다.

나. aGSS 알고리즘의 태스크 크기 결정

aGSS 알고리즘은 기본적으로 GSS 알고리즘과 동일한 방법으로 동작을 하지만 태스크 크기 t는 다음과 같이 결정한다.

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

tremain은 남아 있는 태스크의 크기이며, G는 aGSS 알고리즘을 위한 파라미터로 GSS 알고리즘의 G와 같은 의미를 갖는다. BogoMIPSaverage는 클러스터를 구성하는 모든 노드들의 BogoMIPS 평균을 나타내며, BogoMIPSnode는 태스크를 할당받으려는 노드의 BogoMIPS를 나타낸다.

aGSS에서 주 노드는 각 종속 노드를 위한 프로세스를 생성한 후 각 종속 노드에서 수행되는 프로세스로부터 각 노드의 BogoMIPS를 전달받으며, 전달받은 BogoMIPS를 이용하여 평균 BogoMIPS와 각 종속 노드를 위한 태스크의 크기를 결정한다. 동기종(homogeneous) 클러스터의 경우에는 모든 노드의 CPU 성능이 동일하므로 aGSS는 GSS와 동일한 크기의 태스크를 각 종속 노드에게 할당한다.

aGSS 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, aGSS에서는 낮은 성능의 노드에서의 지연이 전체 클러스터의 성능 저하에 영향을 주는 것을 감소시키고자 하였다.

III. gGSS의 실험 결과

본 논문에서는 GSS와의 비교를 위해 aGSS를 위한 파라미터 G의 값으로 4, 8, 16, 32, 64, 128을 사용하여, 각각 5회씩 aGSS 알고리즘을 사용하는 병렬 프로그램을 수행하여, 그 수행 시간의 평균을 계산하였다. 각 경우에 대한 평균 수행 시간은 표 2와 같다.

표 2의 수행 시간에 의하면 aGSS 알고리즘은 파리미터 G의 값에 따른 수행 시간의 편차가 Send 알고리즘과 GSS 알고리즘의 편차보다 작다. 이것은 aGSS 알고리즘이 부하 분할을 위한 파라미터를 결정하는 것이 다른 알고리즘에 비해 자유롭다는 것을 의미한다. 즉, Send와 GSS의 경우 최적의 수행 시간을 위한 파리미터를 찾기 위한 시도가 필요하지만 aGSS 알고리즘은 적당한 파라미터 값을 이용하여 병렬 프로그램을 수행하더라도 최적의 수행 시간과 많은 차이가 나지 않는다.

표 2. aGSS 알고리즘수행 시간

G	평균 수행 시간(초)
4	21.68
8	20.36
16	19.13
32	19.03
64	18.92
128	19.19

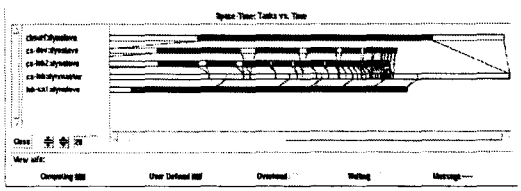

그림 2. G=4인 경우 aGSS의 수행 패턴

aGSS의 이러한 결과는 다른 알고리즘에 비해 각 종속 노드에 효과적으로 태스크를 배분하며, 이를 통

해 전체 클러스터의 성능을 향상시킨다. 그림 2는 aGSS의 수행 시간 중에 가장 긴 수행 시간을 가진 G=4의 경우에 병렬 프로그램의 수행 패턴을 나타낸 것이다. 그림 2에 의하면 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.

IV. 결론

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 연구뿐만 아니라 이용되고 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 적응적 부하 분할 알고리즘 aGSS를 제안하였다. 이 알고리즘은 GSS와 같이 파라미터 G에 의해 태스크 크기를 결정한 후 각 노드의 성능에 따라 태스크 크기를 재조정하는 수정된 GSS 알고리즘이다. 본 논문에서 제안한 aGSS의 효과성을 입증하기 위해 서로 다른 성능을 갖는 다섯 대의 노드를 이용하여 이기종 클러스터를 구축하였으며, GSS 알고리즘과 비교, 분석하였다. 각 알고리즘을 비교하기 위한 두 개의 720×720 행렬을 곱하는 연산을 수행하는 병렬 프로그램을 작성하여 그 수행 시간을 이용하였다.

비교 분석의 결과로는 GSS 알고리즘은 낮은 성능의 노드에 의한 클러스터의 성능 감소가 있지만, 본 논문에서 제안한 aGSS 알고리즘은 태스크를 각 노드에 적절하게 분할함으로써 낮은 성능의 노드가 클러스터의 성능에 영향을 주는 정도를 최소화하였다.

또, GSS의 경우 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 파라미터 결정에 있어 상대적으로 자유롭다.

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

참고문헌

[1] 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, 15(2), pp.54-64, Feb. 1995.

[4] 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.

[5] 구본근, "NOW 환경에서 개선된 고정 분할 단위 알고리즘", 정보처리학회논문지A, 제8-A권, 제2호, pp.117-124, 2001년 6월.