NOW 환경에서 작업자 프로세스의 수가 수행시간에 미치는 영향분석

조 수 현° 김 영 학
금 오 공 과 대 학 교 대 학 원 컴 퓨 터 공 학 과
{shcho, yhkim}@cespc1.kumoh.ac.kr

An Effect Analysis between the Number of Worker Processes and Execution Time on a NOW

Soo-Hyun Cho° Young-Hak Kim
Dept. of Computer Engineering, Kumoh National University of Technology

요 약

최근 컴퓨터 하드웨어 기술의 비약적인 발전으로 인하여 고가의 슈퍼컴퓨터를 대신하여 저가의 컴퓨터들을 이용한 NOW 환경에서 복잡한 응용 문제들이 효율적으로 해결되고 있다. 일반적인 NOW 환경에서의 성능은 개개의 컴퓨터들의 계산능력과 통신시간에 좌우된다. 본 논문에서는 실제 NOW 환경에서 성능향상을 위한 여러 요소들을 살펴보고, 특히 작업자 프로세스 수가 전체 수행시간에 어떤 영향을 미치는가를 실험적인 성능평가를 통해 분석한다. 성능 평가들은 NOW 환경에서 LAM/MPI를 이용하여 측정하였다.

1. 서론

비약적인 컴퓨터 기술의 발전으로 슈퍼컴퓨터들의 전유물 이였던 강력한 계산능력이 네트워크에 연결된 컴퓨터들의 자원을 이용한 NOW(Network of Workstations)로서 이 문제를 어느 정도 해결할 수가 있다. NOW는 PC, 워크스테이션 등과 같은 비교적 저가의 컴퓨터들을 네트워크에 연결하여 서로의 계산 능력을 공유함으로서 강력한 계산 능력을 제공하는 기술이다.

최근에 병렬처리를 위해 네트워크에 연결된 워크스테이션 자원과 MPI(Message Passing Interface)[1], PVM(Parallel Virtual Machine)을 이용한 연구가 활발히 진행되고 있다.

NOW 환경에서 전체 수행시간을 향상시킬 수 있는 요소는 다음과 같다.

- o 워크스테이션 자원의 유휴시간(Idle time)을 줄인다.
- o 워크스테이션 부하를 분할(*Load sharing*)한다.
- o 전송하는 데이터의 크기가 클 경우 워크스테이션의 대기시간(Latency)이 길어지므로 알맞은 크기의 데이터를 결정한다.
- o 병렬처리를 위해 효율적으로 데이터를 전송한다.
- o 네트워크에 연결된 워크스테이션은 성능의 차이를 보일 수 있으므로 그에 맞는 부하 분할과 전송 데이터 크기를 고려해야 한다.

위와 같은 일반적인 요소에 대한 연구는 많이 진행되었지만 실제 계산을 하는 Worker Processes(이하 작업자 프로세스)수와 관련해서는 현재 연구가 되지 않고 있다. 본 논문은 NOW 환경에서 일반적인 성능향상에 영향을 미치는 요소들과 작업자 프로세스 증가가 수행시간에 어떤 영향을 미치는가에 대해서 실험을 통하여 성능을 분석하였다. 그리고 모든 성능평가를 위해서는 행렬 곱셈 계산식과, LAM/MPI를 이용하였다.

본 논문의 구성은 다음과 같다. 2장은 관련연구에 대해 개괄적으로 설명하고, 3장은 브로드캐스팅(Broadcasting) 알고리즘에 대해 언급하며, 4장에서는 부하 분할 알고리즘을, 5장은 실험결과 및 분석에 대해서, 끝으로 6장에서 결론 및 향후 연구과제를 설명한다.

2. 관련 연구

MPI는 워크스테이션 클러스터에서 사용할 수 있는 표준화된 메시지 전달 함수의 라이브러리로, 동기 및 비동기의 점대점 통신함수(MPI_Send, MPI_Ssend)와 집단화 통신(Collective Communication) 함수(MPI_Bcast, MPI_Reduce)를 제공한다.

일반적으로 널리 사용되는 라이브러리는 PVM, MPICH, LAM/MPI [2]가 있다. 본 논문에서는 미국 노트르담(Notre Dame) 대학에서 개발한 LAM/MPI를 이용하여 MPI_Bcast함수에 대한 성능평가를 한다.

기존의 MPI_Bcast 함수는 참여하는 작업자 프로세스 수에 따른 점대점 통신 (Point to point)으로 각각의 노드들에게 순차적으로 메시지를 보내는 방식과(이하 마스터/슬레이브), binomial tree(이하 트리)를 이용한 LogP 단계 이후에는 전체노드에게 전송되는 방식으로 구성되어 있다. 하지만 기존 방식들은 근거리를 기반으로 작성된 라이브러리들이기 때문에 실제 원거리(Wide Area) 전송에는 접합하지 않다.

최근에 T.Kielmann 등에 의한 MAGPIE[3]가 제안되었는데 이는 원거리 집단화 통신을 효율적으로 하기 위해 개발된 라이브러리이다. 이 함수는 노드들을 일정 비율로 그룹을 만들면 각 그룹의 조정자(Coordinators)를 통해 통신이 이루어진다. 처음 마스터(Master) 노드는 각 그룹의 조정자들에게 점대점 통신을

하고, 그 다음 각 그룹내의 조정자를 통해 트리 방식으로 전송한다. 그러나 노드관점의 브로드캐스팅이지 본 논문에서 언급할 작업자 프로세스 수에 대해서는 고려하지 않았다.

3. 브로드캐스팅 알고리즘

기존의 브로드캐스팅 알고리즘들은 점대점 통신으로 순차적으로 메시지를 보내는 마스터/슬레이브 방식과 매번 수신되는 노드들의 수를 2배씩 확대해 나가는 트리 방식이 있다. 이는 근거리 통신 기반이며 동일한 노드에 중복 데이터를 전송해 불필요한 통신시간들이 생겨 원거리 통신에는 적합하지 않다.

3.1 기존 방법

마스터/슬레이브 방식은 데이터를 전송하는 마스터 프로세스(이하 마스터)가 전송 받는 나머지 슬레이브(Slave) 프로세스(이하 슬레이브)들에게 순차적으로 전송한다. 단순한 전송방법이며 전송하는 시간과 수신하는 시간과의 차이가 클 경우, 즉 원거리 데이터 전송에 적합한 방법이다. 하지만 전송하는 데이터의 크기가 커질수록 통신시간이 길어짐으로 전체 수행시간 또한 길어진다.

트리 방식에서 마스터는 전송 데이터를 임의의 프로세스에게 보내고 현재 데이터를 받은 모든 프로세스가 그렇지 않은 프로세스에게 전달하여, 재귀적인 방법으로 수신 프로세스의 수를 2배씩 확대해 나가는데 결국 LogP 단계 후에는 전체 프로세스에게 전송된다. 전송하는 시간과 수신되는 시간과의 차이가 작을수록 효과적인 방법이다. 그래서 일반적으로 근거리 데이터 전송에 최적의 방식이다. 그러나 원거리 전송에 있어 동일한 노드의 프로세스들에게 같은 내용의 데이터를 여러 번 전송하기 때문에 불필요한 통신시간이 길린다.

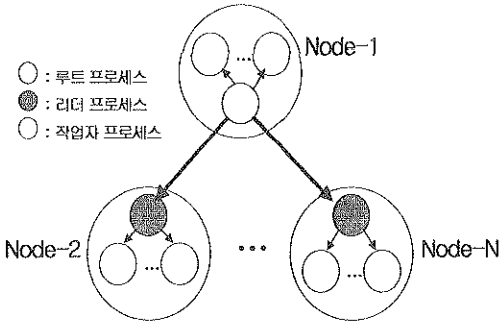

그림 1. 제안된 브로드캐스팅

3.2 제안된 방법

그림 1과 같이 두 번의 단계를 거치게 되는데 첫 번째로 마스터는 각 그룹의 리더(Leader) 프로세스에게 데이터를 전송한다. 그 다음 리더 프로세스는 자신의 그룹에 속한 나머지 슬레이브들에게 데이터 값을 전송하게 된다. 이는 마스터/슬레이브, 트리 방식에서의 같은 그룹에 있는 프로세스들에게 동일 데이터를 여러 번 전송하는 시간을 고려한 것으로 특히 원거리 전송에 있어 월등한 성능향상이 있을 것을 예상할 수 있다.

본 논문에서는 조정자를 사용하지 않고 리더 프로세스를 사용한다. 리더의 기능은 계산작업은 하지 않으며 마스터로부터 전송된 데이터를 자신의 그룹 슬레이브들에게 전송한다.

4. 부하 분할 알고리즘

실제 계산작업에 있어 사용되는 부하분할 알고리즘은 NLS(No Load Sharing), LS(Load Sharing), Pre-LS가 있다. 본 논문에서는 각 노드에 하나씩 작업자 프로세스를 생성하여 각 알고리즘들을 적용한 것과 작업자 프로세스 증가에 대해 성능평가를 한다. 마스터 노드의 루트 프로세스는 작업 배분과 결과값 취합만을 수행하고, 이러한 알고리즘들은 행렬 곱셈문제를 사용하여 평가한다.

4.1 NLS (No Load Sharing) 알고리즘

먼저, 마스터는 다른 슬레이브들에게 첫 번째 행렬식을 전송한다. 그런 다음 두 번째 행렬식에서의 일정비율(C/S)크기의 데이터를 각 슬레이브들에게 전송한다. C는 두 번째 행렬식의 전체 행(Row)수를 말하며 S는 슬레이브 수를 나타낸다. 마스터가 계산수행에 참여할 경우에 S는 S+1이 된다.

이 알고리즘에서 전체 수행시간은 참여하는 노드들의 성능에 좌우되며, 전송하는 데이터의 크기가 클 경우 마스터/슬레이브간의 통신 대기시간을 길게 하는 원인이 된다.

4.2 LS (Load Sharing) 알고리즘

이 알고리즘은 "Work Pool"을 기반으로 한 알고리즘이다. 각 슬레이브는 자신이 수행할 작업(Task)을 마스터로부터 받아 수행하며 그 결과 값을 다시 마스터에게 전송한다. 마스터는 슬레이브들이 전송한 결과 값을 받아 처리하고 전송한 슬레이브들에게 다음 작업을 전송하게 된다. 이런 작업은 전송할 두 번째 행렬식의 전체 행수만큼 반복된다.

이 알고리즘은 4.1 절에서의 단점인 노드들의 성능 의존성을 해결할 수 있다. 하지만 그렇게 하기 위해서는 각 슬레이브들에게 전송할 알맞은 작업의 크기를 결정해야 하며, 각 슬레이브들이 결과 값을 마스터에게 전송하고 다음 작업을 하기 위해서는 마스터로부터 작업을 전송 받아야만 하는데 그로 인한 통신 대기시간이 걸리게 된다.

4.3 Pre-LS 알고리즘

4.2 알고리즘의 단점을 해결코자 마스터는 각 슬레이브들에게 작업들을 전송하며 슬레이브들로부터 결과값 수신여부와 관계없이 일정시간 간격으로 슬레이브들에게 미리 다음 작업을 전송하는 방식이다. 즉 통신시간과 계산시간을 중복되게 함으로 자원의 유휴시간을 최대한 줄여보자는 취지와 일맥상통하는 것이다.

4.4 작업자 프로세스 증가에 따른 Pre-LS 기반 알고리즘

이전 3가지 알고리즘은 각 노드에 하나의 작업자 프로세스가 생성된 상태지만 이 알고리즘에서는 작업자 프로세스를 증가하여 수행을 한다. 부하 분할은 Pre-LS 알고리즘을 기반으로 하여 성능평가를 하였다. 또한 작업자 프로세스 관점과 다른 요소(데이터 크기, 브로드캐스팅)들간의 관계에 대해서도 성능평가를 한다.

5. 실험결과 및 분석

5.1 실험 환경

성능평가를 위한 실험환경은 다음과 같이 구성되어 있다.

1. 운영체제 : Solaris 2.5.1
2. 소프트웨어 : LAM/MPI 6.3.2
3. CPU/Memory : SUN SPARC 110MHz, 32M

실험에 참여한 노드들은 10Mbps 이더넷으로 연결된 9대의 워크스테이션들이며 성능평가를 위해 다양한 크기의 행렬 곱셈계산을 수행하였다.

5.2 실험 결과

5.2.1 브로드캐스팅 결과 비교

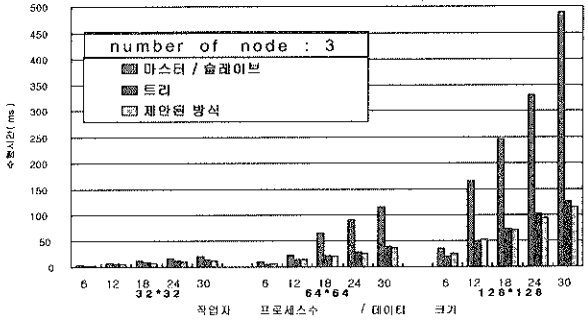

그림 2. 작업자 프로세스 수에 따른 브로드캐스팅 결과 비교

일반적으로 브로드캐스팅 함수인 MPI_Bcast 함수는 마스터와 슬레이브들 모두 이 함수를 호출하여 수행되었을 때 비로서 데이터 전송이 이루어진다. 즉 마스터/슬레이브들의 동기화가 이루어졌을 때 전송된다는 의미이다.

그림 2는 고정된 데이터 크기에서 작업자 프로세스 수에 따른 브로드캐스팅 결과 비교를 보여주고 있다. 본 논문에서 사용된 LAM/MPI의 마스터/슬레이브, 트리 방식은 생성된 작업자 프로세스 수에 따라 구분된다. 작업자 프로세스수가 6개 이하인 경우 별다른 차이가 없으며, 그 이상일 경우 많은 성능차이를 보였다. 또 다른 성능차이의 원인으로 브로드캐스팅 되는 데이터의 크기다. 데이터의 크기가 클수록 마스터/슬레이브 방식과 나머지 방식들간에 아주 큰 성능차이가 있었으며 트리와 제안된 방식에서도 점차적인 성능 차이를 보였다. 마지막으로 노드수에 따른 성능비교에서도 같은 성능차이를 확인할 수 있었다.

5.2.2 부하 분할 결과 비교

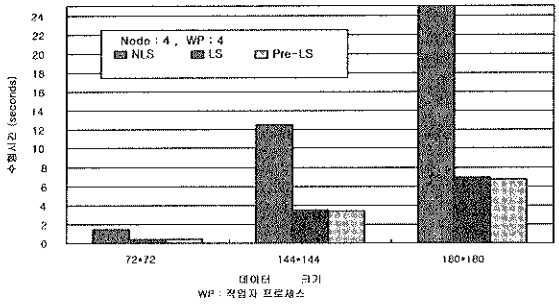

그림 3. 부하 분할 방식들간의 결과 비교

그림 3에서 NLS 방식은 데이터의 크기가 증가할수록 다른 방법들과 비교해 현저한 성능저하를 보여준다. 이는 데이터를 처리하는 시간은 일정하다고 볼 때 전체 수행시간에 영향을 미치는 통신 시간이 길어지기에 이런 결과가 나온 것으로 판단된다. LS와 Pre-LS간의 성능비교에서도 차이를 보이는데 이것은 마스터가 작업할 데이터를 보내고 각 슬레이브는 처리한 결과를 다시 마스터에게 보내는데 그때 마스터로부터의 다음 작업 데이터를 받기까지 통신대기 시간에 있기 때문이다. Pre-LS 방식에서 주의할 점은 각 슬레이브의 결과값 수신여부와 관계없이 전송되는 일정시간 간격을 고려해야 한다.

5.2.3 작업자 프로세스 수에 따른 결과 비교

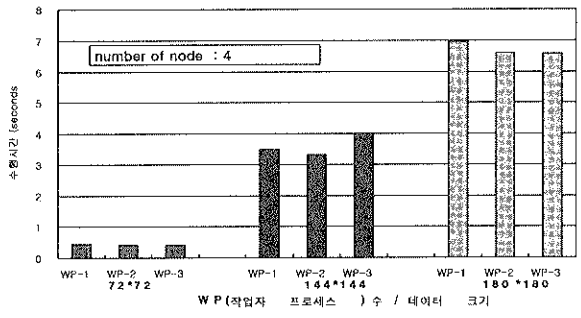

그림 4. 실제 계산에 있어 작업자 프로세스 수에 따른 결과 비교

그림 4는 노드수가 4개이며, 마스터는 단지 작업을 배분하여 전송하고 각 슬레이브로부터의 결과 값을 취합하는 작업만 한다. 작업자 프로세스 수와 전체 수행시간과의 관계에서 각 슬레이브에 전송된 데이터의 처리시간과 마스터로의 결과값 전송시간은 일정하였는데 이는 마스터에서 각 슬레이브로 데이터를 전송하는 시간이 전체 수행시간을 좌우한다는 것을 의미한다. 또한 마스터가 각 슬레이브들에게 전송하는 작업 데이터의 크기에 따라 성능차이를 보였다.

작업자 프로세스 수는 효과적인 브로드캐스팅과 알맞은 작업 데이터 크기로 배분한 경우 성능향상을 보인다. 즉 작업자 프로세스수가 증가한다고 해서 성능이 향상되는 것이 아니라 그에 따라 고려할 사항들이 있음을 말한다. 이것은 전체수행 시간에 있어 그 만큼 작업 계산시간보다 통신시간에 따라 전체성능이 좌우됨을 확인할 수 있다.

6. 결론 및 향후 연구과제

NOW 환경은 이질적인 작업환경으로 통신 대기시간, 유휴시간, 부하분할, 전송방법 등을 고려해야 한다. 전송방법에서는 마스터/슬레이브, 트리, 제안된 방식이 전송되는 데이터의 크기와, 작업자 프로세스수가 증가할수록 수행시간이 증가하였으며 제안된 방식이 가장 효율적인 전송방법임을 알 수 있었다.

부하분할 방법은 최대한 노드들의 유휴시간을 줄여 전체 수행시간을 단축시키고자 한 것이다. Pre-LS방식이 가장 좋은 성능을 보인 것은 그만큼 노드들의 유휴시간을 줄였으며 다시 말해 통신시간과 계산시간을 중복시킨 결과이다.

마지막으로 위의 두 가지 방법을 취합하여 실제 작업을 수행하는 작업자 프로세스 수와의 관계에서는 작업자 프로세스 수를 증가하기보다 효율적인 전송방법, 부하분할, 데이터 크기와 함께 고려되어야 성능이 향상됨을 알 수 있다. 또한 NOW 환경에서 전체 수행시간은 통신시간이 주요하게 좌우된다는 것을 확인할 수 있었다.

향후 연구과제는 이질적인 작업환경에서 각 노드의 성능에 따른 부하분할과 다른 요소들과의 관계를 알아보는 것이다.

참고문헌

[1] MPI Forum, MPI-2: Extensions to the Message-Passing Interface, Http://www.mpi-forum.org/docs/mpi-20-html/mpi2-report.html

[2] LAM/MPI Parallel Computing, Http://www.mpi.nd.edu/lam/

[3] T. Kielmann, R.F.H.Hofman, H.E.Bal, A. Plaat, and R.A.F.Bhoedjang. "MAGPIE: MPI's Collective Communication Operations for Clustered Wide Area Systems. In Proc. Symposium on Principles and Practice of Parallel Programming(PPoPP), pp.131-140, Atlanta, GA, 1999.