Computational Grid 상에서 네트워크 성능정보를 고려한 효율적인 작업 프로세스 할당 방법

조수현° 김영학
금오공과대학교 대학원 컴퓨터공학과
{shcho°, yhkim}@cespc1.kumoh.ac.kr

Efficient Work Processes Allocation Method Considering Network Performance Information on Computational Grid

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

요 약

그리드 컴퓨팅은 우주과학, 수학적인 큰 문제들을 해결하기 위해 네트워크 상에 분산된 수많은 컴퓨터들의 컴퓨팅 파워와 대용량 저장장치를 공유하여 문제들을 해결할 수 있는 기술이다. 그리도 컴퓨팅의 환경은 WAN으로 구성된 각기 다른 성능과 이질적인 네트워크 상태들로 구성된다. 이런 이질적인 성능요소들을 고려하여 계산 작업에 반영시키는 것이 무엇보다 중요하다. 본 논문에서는 WAN 환경의 네트워크 상태 정보 중 latency, bandwidth, latency-bandwidth 혼합정보들을 고려하여, 노드별로 작업 프로세스 수를 결정하는 방법을 제안한다. 본 논문에서의 네트워크 성능정보 수집은 NWS(Network Weather Service)를 통해 이뤄지며, 평가결과 네트워크 성능정보를 고려하지 않은 균등방식에 비해서 latency, latency-bandwidth 고려한 방법의 결과가 9%, 31% 성능이 향상되었다.

1. 서 론

그리드 컴퓨팅(Grid Computing)은 지역 네트워크를 초월한 인터넷 영역의 가상 집합체들의 자원을 공유하여 방대한 계산 작업을 수행할 수 있는 새로운 패러다임이다. 그리드 컴퓨팅의 성능향상을 위해서는 그리드 환경에서 수행될 응용 프로그램과 실제 계산작업을 수행하는 자원, 정보, 저장 공간 등이 효율적으로 관리되어야 한다.

또한 WAN(Wide Area Network)으로 구성된 그리드 환경은 노드별 컴퓨팅 파워뿐만 아니라, 이질적인 네트워크 상태정보들을 고려하여 반영하는 것이 무엇보다 중요하다. 본 논문에서는 그리드 환경에서 네트워크 상태정보들을 고려한, 각 노드에 생성될 최적의 작업 프로세스 수를 결정하는 방법을 제안한다. 네트워크 상태정보로는 latency, bandwidth, latency-bandwidth를 혼합한 정보를 이용하여 노드별 작업 프로세스 수를 결정한다.

네트워크 성능정보 수집을 위해선 NWS(Network Weather Service)[1]를 이용하고, 성능요소별 분석 및 분류기능을 추가하였다. 분류된 정보를 기반으로 각 노드의 성능비율을 계산한 후, 노드별로 생성될 작업 프로세스 수를 결정한다. 최종적으로 결정된 작업 프로세스 수를 통해 각 노드에 수행할 정보들을 갖고 있는 RSL(Resource Specification Language) 파일을 자동으로 생성하여 실행한다.

본 논문의 구성은 다음과 같다. 2장은 관련연구에 대해 개괄적으로 설명하고, 3장은 제안된 노드별 작업 프로세스 수 계산 방법과, 시스템 구성에 대해 언급한다. 4장에서는 실험결과 및 분석에 대해서, 끝으로 5장에서 결론 및 향후 연구를 설명한다.

2. 관련 연구

WAN영역에 분포하고 있는 많은 자원 및 네트워크 상태는 이질적인 환경에 놓여있다. 이런 이질적인 환경을 고려하기란 쉽지 않기 때문에 응용 프로그램 사용자에게는 많은 노력들이 요구된다. 이를 해결하기 위해 미들웨어인 글로버스(Globus)[2] 툴이 개발되어 어느 정도 문제점들을 해결하고 있다. 하지만 사용자는 자원 및 정보이용에 있어 수동적으로 이용하는 단계에 불가하다. 즉 노드에 대한 위치 및 일반적인 정보(CPU, Memory)만을 이용할 뿐, 네트워크 상태정보에 따른 분석 및 분류 기능 등을 이용할 수 없다. 본 논문에서는 WAN환경의 중요한 성능요소 인 네트워크 상태정보들을 고려하여 성능요소별 분류를 통해 노드별 성능비율을 구한 후, 노드별 작업 프로세스 수를 결정하여 사용자에게 보다 나은 작업 정보를 제공한다.

[3]에서는 네트워크의 성능요소 중 latency를 기반으로 통신 트리를 구성하는 연구들이 제안되었다. 하지만 latency 정보 수집 및 통신 트리 구성 작업들이 응용 프로그램이 수행되는 시점에서 이뤄진다. 즉 전체 수행시간에서 latency 측정시간과 통신 트리 구성 시간 등이 불필요하게 반영되는 문제점이 있다. 본 논문에서는 네트워크 상태 정보수집과 상태 정보에 따른 노드 분류, 노드별 성능비율 계산 등을 응용 프로그램 수행과 별개로 진행하여 문제점들을 해결한다.

3. 제안된 방법

본 절에서는 성능정보를 기반으로 노드별 성능비율 계산과 작업 프로세스 수를 결정하는 방법을 알아본다. 또한 네트워크 성능요소인 latency, bandwidth 정보 수집 및 분류방법과 시스템구조를 살펴본다.

3.1 노드별 작업 프로세스 수 계산 방법

3.1.1 노드별로 균등하게 프로세스 생성

하나의 노드는 작업에 참여하지 않고 요청 및 결과 값만을 확인하는 순수 클라이언트이므로, 작업에 참여한 전체 노드 수

는 N-1이 된다. 사용자가 요청한 작업 프로세스 수를 P라고 할 때, 네트워크 상태정보를 고려하지 않고, 요청한 프로세스 수를 N-1개의 노드 수만큼 균등하게 작업 프로세스들을 생성한다. 다음은 노드별로 균등하게 작업 프로세스 수를 결정하는 수식을 나타낸다.

-node_Ni_process: 노드 i에 생성될 프로세스 수

$$node\_N_i\_process = P / N-1$$

3.1.2 latency를 고려한 노드별 프로세스 수 계산

성능요소별 분류 중 latency 정보를 기반으로 노드별 성능비율을 계산한다. 요청한 프로세스 수를 노드별 성능비율을 적용하여 생성할 작업 프로세스 수를 결정한다. 노드별 성능비율은 latency를 기반으로 각각 오름차순, 내림차순을 수행한 후, 내림차순 한 노드의 latency 값을 오름차순 한 노드의 성능 값이 되어 전체노드의 latency 총합으로 나눈 값이다. 마지막으로, 요청한 프로세스 수에 노드별 성능비율을 적용하면 해당 노드의 작업 프로세스 수가 결정된다. 다음은 노드별 latency를 고려한 작업 프로세스 수를 결정하는 수식을 나타낸다.

-total_latency_sum : 전체 노드들의 latency값의 총합

-latency_descending_sort[i] : 내림차순 한 노드에 대한 포인터

$$node\_N_i\_process = ROUND\ (\ P * (latency\_descending\_sort[i]->latency\ /\ total\_latency\_sum))$$

3.1.3 bandwidth를 고려한 노드별 프로세스 수 계산

bandwidth 정보를 기반으로 노드별 bandwidth 값을 전체 bandwidth 총합으로 나누어 성능비율을 계산한다. 요청한 프로세스 수에 노드별 성능비율을 적용하면 해당 노드의 작업 프로세스 수가 결정된다. 다음은 각 노드별 bandwidth를 고려한 작업 프로세스 수를 결정하는 수식을 나타낸다.

-total_bandwidth_sum: 전체 노드들의 bandwidth값의 총합

-node_Ni_bandwidth: 노드 i의 측정된 bandwidth 값

$$node\_N_i\_process = ROUND\ (\ P * (node\_N_i\_bandwidth\ /\ total\_bandwidth\_sum))$$

3.1.4 latency-bandwidth를 혼합한 프로세스 수 계산

이전에 계산한 노드별 latency, bandwidth 성능비율을 혼합하여 프로세스 수를 결정한다. 혼합한 성능요소의 수를 C라고 하면 본 논문에서는 latency-bandwidth를 혼합하였기에 2가 된다. latency와 bandwidth의 성능비율의 합을 C로 나누면 해당 노드의 latency-bandwidth를 혼합한 성능비율이 계산된다. 따라서 요청한 프로세스 수에 혼합한 성능비율을 적용하면 해당 노드의 작업 프로세스 수가 결정된다. 다음은 노드별 latency-bandwidth를 고려한 작업 프로세스 수를 결정하는 수식을 나타낸다.

$$N_i\_latency\_bandwidth\_sum = N_i\_latency\_percentage + N_i\_bandwidth\_percentage$$

$$N_i\_latency\_bandwidth\_percentage = N_i\_latency\_bandwidth\_sum\ /\ C$$

$$node\_N_i\_process = ROUND\ (\ P * N_i\_latency\_bandwidth\_percentage\ )$$

3.2 시스템 구성

3.2.1 네트워크 상태정보 수집 및 분류

본 논문에서는 네트워크 상태정보 수집을 위해서 NWS를 사용한다. NWS는 상태정보 외 성능정보 예측기능이 포함되어 있다. 그림 1과 같이 각 노드에는 자신의 상태정보를 일정시간 간격으로 수집하는 센서들이 작동한다. 노드별 센서들은 수집한 정보를 메모리 서버로 전송한다. 수집된 정보들은 메모리 서버에서 일관되게 관리되고 사용자들은 이를 이용한다.

하지만 성능정보별 분석 및 분류할 수 있는 기능이 없는 상태다. 본 논문에서는 성능정보별로 노드들을 분류하여 추후에 노드별 성능비율 계산에 사용할 수 있게끔 라이브러리 형태로 구현하여 추가하였다.

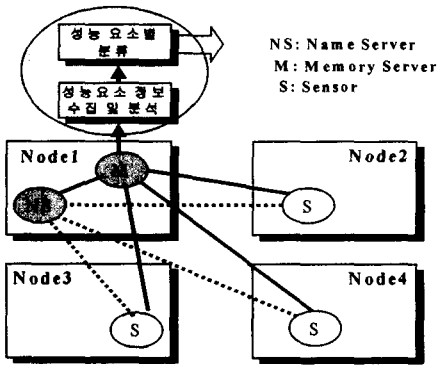

그림 1. NWS기반 성능요소별 노드 분류

3.2.2 시스템 구조

그림 2는 NWS로부터 수집한 정보를 기반으로 성능요소별 분석 및 분류를 하고 노드별 성능비율을 계산한 후, 성능비율에 따른 작업 프로세스 수를 결정하여 최종적인 RSL 파일을 만들어 수행하는 것을 보여준다.

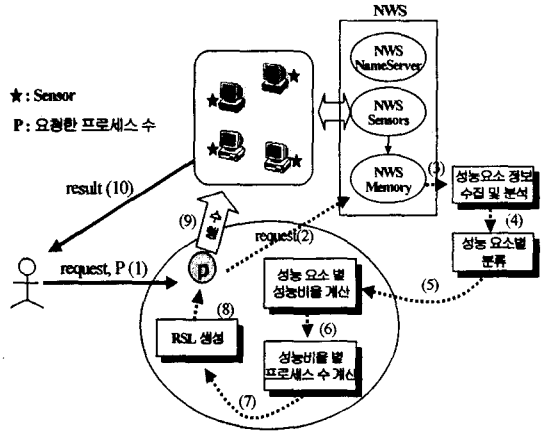

그림 2. 시스템 구조

기존 그리드 시스템 사용자들은 노드별 작업 프로세스 수를 결정하기 위해 일반적인 노드들의 정보만을 이용하여 각 노드에 균등하게 작업 프로세스들을 생성하였다. 또한 노드별 수행할 작업내용을 담고 있는 RSL 파일을 작성하기 위해 특성 명령어를 이용 해야하는 불편함이 있다.

본 논문에서는 사용자가 수행 요청과 함께 계산작업을 위해 필요한 작업 프로세스 수만을 전달한다. NWS를 통해 수집된

네트워크 상태정보인 latency, bandwidth, latency-bandwidth 혼합정보를 기반으로 분류작업을 한 후, 노드별 성능비율을 계산하여 작업 프로세스 수를 결정한다. 마지막 단계에서는, 결정된 작업 프로세스 수를 기반으로 RSL 문서를 자동으로 생성하여 실질적인 계산작업을 수행한다. 또한 2-8 단계 수행절차는 사용자 요청과 별개로 진행하여 네트워크 성능정보 수집 및 분류 등과 같이 불필요하게 소요되는 시간을 단축한다.

4. 실험결과 및 분석

4.1 실험 환경

성능평가를 위한 실험 환경은 다음과 같이 구성된다.

1. 운영체제 : RedHat Linux 9.0(커널버전 : 2.4.20-8)
2. 소프트웨어 : Globus Toolkit 2.2.2, MPICH_G2[4]
   NWS 2.8.1, iperf-1.7.0[5]

실험에 참여한 노드들은 10Mbps 이더넷으로 연결된 3대의 노드들이며 성능평가를 위해 256*256 행렬 곱셈계산을 수행하였다. 실험평가 환경은 2가지 경우로 나눠 수행한다.

o 네트워크 상태가 정적인 경우

본 논문의 실험 평가를 위한 네트워크 환경은 상태변화가 일정한 정적인 상태이다. 그래서 특별히 네트워크에 대한 임의적인 조작 없이 요청 프로세스 수를 증가하면서 균등, latency, bandwidth, latency-bandwidth들을 적용하여 평가한다.

o 네트워크 상태가 동적인 경우

실질적인 WAN환경은 네트워크의 상태변화가 동적으로 변화한다. 본 논문에서는 네트워크 상태변화를 위해 네트워크 트래픽을 발생시키는 iperf를 이용하여 동적인 네트워크 환경을 구성하였다. 평가방법은 정적인 방식과 동일하다.

4.2 실험 결과

그림 3은 정적인 네트워크 상태에서 4가지 평가방법을 적용하여 수행한 결과를 보여준다. 네트워크 변화가 일정한 상태이기에 큰 폭의 성능차이는 보이지 않았다. 특히 요청한 프로세스 수가 4와 같이 작을 경우에는 4가지 방식간의 별다른 성능차이는 없다. 경우에 따라서는 균등한 방식이 우수하였다. 하지만 요청한 프로세스 수가 증가할수록 균등, bandwidth 방식보다 latency, latency-bandwidth를 고려한 방법이 6%, 13% 성능이 향상되었음을 알 수 있다.

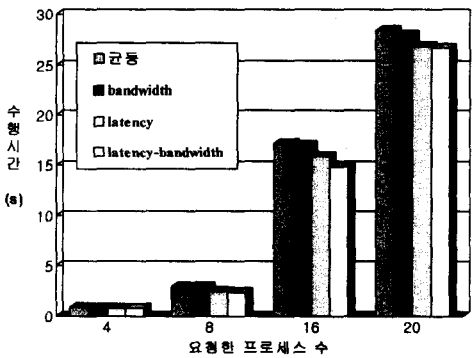

그림 3. 네트워크 상태가 정적인 경우에서의 수행결과

그림 4는 4가지 평가방법을 네트워크 상태가 동적인 환경에서 수행한 결과를 나타낸다. 정적인 상태와 같이 요청한 프로세스 수가 4일 때 별다른 성능차이는 없다. 하지만 프로세스 수가 증가할수록 정적인 환경에 비해 성능향상 폭이 컸으며 latency, latency-bandwidth가 다른 2가지 방법에 비해 9%, 31% 성능이 향상됨을 알 수 있다. 특히 요청 프로세스 수가 16일 때 노드간 네트워크 상태변화가 급변할 경우에는 latency-bandwidth를 혼합한 성능이 탁월함을 알 수 있다.

네트워크 상태가 정적-동적인 환경모두에서, 요청 프로세스 수가 증가할수록 latency, latency-bandwidth 방법의 성능이 향상되었음을 확인하였다. 이는 WAN으로 구성된 그리드 환경에서 네트워크 상태를 고려하는 것이 전체 성능 향상에 많은 부분 반영됨을 실험을 통해 확인할 수 있다.

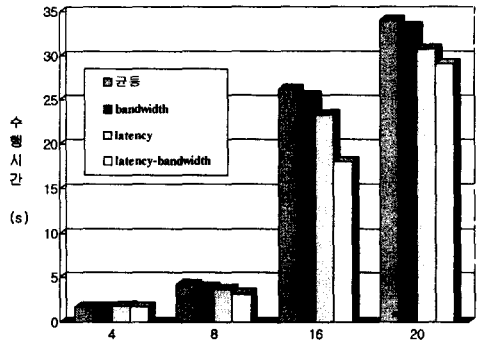

그림 4. 네트워크 상태가 동적인 경우에서의 수행결과

5. 결론 및 향후 연구

그리드 컴퓨팅은 WAN 환경으로 구성되어 네트워크를 통해 작업 및 결과 값이 송수신됨으로 네트워크의 상태정보 반영이 무엇보다 중요하다. 본 논문에서는 네트워크 상태정보 중 latency, bandwidth, latency-bandwidth 혼합 정보를 이용하여 노드별 분류 및 성능비율을 계산한 후, 작업 프로세스 수를 결정하는 방법을 제안한다. 또한 NWS에 의해 수집된 정보를 성능요소별 분석 및 분류 기능을 새롭게 추가하여 적용하였다.

실험결과 정적인 네트워크 환경보다 동적인 환경에서 네트워크 상태정보를 고려하지 않은 균등방식에 비해서, latency, latency-bandwidth를 고려한 방법이 9%, 31% 성능 향상을 보였다. 향후 연구에서는 그리드 환경에서의 노드별 작업 프로세스 수와 수행시간과의 관련성에 대해 연구 할 예정이다.

참고 문헌

[1] Network Weather Service, Online at http://nws.cs.ucsb.edu/

[2] I. Foster, and C. Kesselman, "Globus: A metacomputing infrastructure toolkit", International Journal of Supercomputer Applications, Vol. 11, No. 2, pp.115-128, 1997.

[3] P. B. Bhat, C. S. Raghavendra, and V. K. Prasanna, "Efficient collective communication in distributed heterogeneous systems," 19th IEEE International Conference on Distributed Computing Systems, 1999.

[4] I. Foster, and N. Karonis, "A Grid-Enabled MPI: Message Passing in Heterogeneous Distributed Computing Systems", Proceedings of Supercomputing 98, 1998.

[5] iperf-1.7.0, Online at http://dast.nlanr.net/Projects/Iperf/