계산 그리드 상에서 각 노드의 작업 프로세스 수를 결정하기 위한 효율적인 방법

An Efficient Method for Determining Work Process Number of Each Node on Computation Grid

조수현*, 김영학**

금오공과대학교 컴퓨터공학과 대학원*, 금오공과대학교 컴퓨터공학부**

Soo-Hyun Cho(shcho@kumoh.ac.kr)*, Young-Hak Kim(kimyh@kumoh.ac.kr)**

요약

그리드 컴퓨팅은 과학기술 분야의 큰 문제들을 해결하기 위해 네트워크 상에 분산된 수많은 컴퓨터들의 컴퓨팅 파워와 대용량 저장장치를 공유하여 문제들을 해결할 수 있는 기술이다. 그리드 컴퓨팅의 환경은 WAN으로 구성된 각기 다른 성능과 이질적인 네트워크 상태들로 구성된다. 그래서, 이러한 이질적인 성능요소들을 고려하여 계산 작업에 반영시키는 것이 무엇보다 중요하다. 본 논문에서는 네트워크 상태정보를 고려한 노드별 작업 프로세스 수를 결정하는 효율적인 방법을 제안한다. 네트워크 상태정보는 latency, bandwidth, latency-bandwidth 혼합정보를 고려한다. 먼저, 측정된 네트워크 상태정보를 이용하여 노드별 성능비율을 구하고 이를 통해 작업 프로세스 수를 결정한다. 마지막 단계에서는, 결정된 노드별 작업 프로세스 수를 기반으로 자동으로 RSL 파일을 생성하여 작업을 수행한다. 네트워크 성능정보는 NWS(Network Weather Service)에 의해 수집된다. 실험결과에 따르면, 네트워크 성능정보를 고려한 방법이 그렇지 않은 기존의 균등방식보다 작업량, 작업 프로세스 수, 노드 수 관점에서 각각 23%, 31%, 57% 성능이 향상되었다.

■ 중심어 : | 그리드 컴퓨팅 | 계산 그리드 | 작업 프로세스 |

Abstract

The grid computing is a technique to solve big problems such as a field of scientific technique by sharing the computing power and a big storage space of the numerous computers on the distributed network. The environment of the grid computing is composed with the WAN which has a different performance and a heterogeneous network condition. Therefore, it is more important to reflect heterogeneous performance elements to calculation work. In this paper, we propose an efficient method that decides work process number of each node by considering a network state information. The network state information considers the latency, the bandwidth and latency-bandwidth mixture information. First, using information which was measured, we compute the performance ratio and decide work process number of each node. Finally, RSL file was created automatically based on work process number which was decided, and then accomplishes a work. The network performance information is collected by the NWS. According to experimental results, the method which was considered of network performance information is improved respectively 23%, 31%, and 57%, compared to the methods of existing in a viewpoint of work amount, work process number, and node number.

■ keyword : | Grid Computing | Computation Grid | Work Process |

* 본 연구는 2004년도 금오공과대학교 학술연구비 지원에 의해 수행되었습니다.

접수번호 : #041013-001

접수일자 : 2004년 10월 13일

심사완료일 : 2005년 01월 03일

교신저자 : 조수현, e-mail : shcho@kumoh.ac.kr

I. 서 론

기존의 방대한 계산 작업은 슈퍼컴퓨터(Supercomputer), 소규모 지역에서의 클러스터(Cluster) 시스템 등을 이용하여 해결하였다. 하지만 계산 작업량이 증가함에 따라 기존 방법으로는 저장 공간, 컴퓨팅 파워, 메모리 등이 한계가 있다. 그래서 이를 해결하기 위한 그리드 컴퓨팅(Grid Computing)은 지역 네트워크를 초월한 인터넷 영역의 가상 집합체들의 자원을 공유하여 거대한 계산 작업을 수행할 수 있는 새로운 패러다임이다. 최근에 나노기술과 생명과학으로 대변되는 21세기 첨단 과학기술 시대에는 천문학적인 정보 트래픽, 방대한 저장 공간, 강력한 계산능력 등이 네트워크를 통하여 공유될 것이고, 이를 실현하는 유력한 수단으로 그리드 컴퓨팅이 주목받고 있다. 그리드 환경에서의 계산 작업은 참여한 각 노드에 작업 프로세스들이 생성되어 프로세스들에 의해 작업이 계산된 후 네트워크를 통해 결과값이 취합되는 형태를 갖는다. 따라서 그리드 컴퓨팅의 성능향상을 위해서는 그리드 환경에서 수행될 응용 프로그램과 실제 계산 작업에 영향을 끼치는 자원(Resources), 정보(Information) 및 저장(Storages) 공간 등이 효율적으로 관리되어야 한다.

또한 WAN(Wide Area Network)으로 구성된 그리드 환경은 노드별 컴퓨팅 파워뿐만 아니라, 이질적인 네트워크 상태정보들을 고려하여 계산 작업에 반영하는 것이 무엇보다 중요하다. 특히, 네트워크에 연결된 분산 노드에 생성된 작업 프로세스들 간의 계산 작업시 수행할 작업을 수신하거나 결과 값을 전송함에 있어 네트워크의 상태정보 중 latency, bandwidth의 정보가 전체 성능을 좌우한다. 따라서 본 논문에서는 그리드 환경에서의 계산 작업에 네트워크 상태정보들을 고려하여 각 노드에 생성될 최적의 작업 프로세스 수를 결정하는 방법을 제안한다. 고려한 네트워크 상태정보는 latency, band-width, latency-bandwidth를 혼합한 정보를 이용하여 노드별 작업 프로세스 수를 결정한다.

네트워크 성능정보 수집을 위해서는 NWS(Network Weather Service)[1]를 이용하고 성능요소별 분석 및 분류기능을 추가하였다. 또한 분류된 정보를 기반으로 각 노드의 성능비율을 계산한 후, 사용자가 요청한 전체 작업 프로세스 수를 노드별 성능비율에 적용하여 노드별로 생성될 작업 프로세스 수를 결정한다. 최종적으로, 결정된 작업 프로세스 수를 통해 각 노드에 수행할 정보들을 갖고 있는 RSL(Resource Specification Language)[2] 파일을 자동으로 생성하여 실행한다.

본 논문의 구성은 다음과 같다. II장은 관련연구에 대해 개괄적으로 설명하고, III장은 제안된 노드별 작업 프로세스 수 계산방법과 시스템 구성에 대해 언급한다. IV장에서는 실험결과 및 분석에 대해서, 끝으로 V장에서 결론 및 향후 연구를 기술한다.

II. 관련 연구

WAN 영역에 분포하고 있는 많은 자원 및 네트워크 상태는 이질적인 환경에 놓여있다. 이런 이질적인 환경을 고려하기란 쉽지 않기 때문에 기존 응용 프로그램 사용자에게는 많은 노력들이 요구된다. 이를 해결하기 위해 그리드 환경에서의 대표적인 미들웨어인 글로버스(Globus)[3] 툴 킷이 개발되어 어느 정도 문제점들을 해결하고 있다. 하지만 사용자는 자원 및 정보이용에 있어 복잡한 명령어를 통해 수동적으로 이용해야 한다. 즉 각 노드에 대한 위치 및 일반적인 정보(CPU, Memory)만을 이용할 뿐 네트워크 상태정보에 따른 분석 및 분류 기능 등은 이용할 수 없다.

본 논문에서는 WAN 환경에서의 계산 작업 시 중요한 성능요소인 latency, bandwidth를 고려하여 성능요소별 분류를 통해 노드별 성능비율을 구한 후, 노드별 작업 프로세스 수를 결정하여 사용자에게 보다 나은 작업 정보를 제공한다. 그리고 기존 글로버스 툴 킷을 이용한 계산 작업은 최종적으로 각 노드에 어떤 작업들을 수행해야하는지의 정보를 갖고 있는 RSL 문서를 통해 작업 수행이 이루어진다. 그러나 일반 사용자는 RSL 문서를 만들기 위해 직접 스크립트 문서를 만들거나 MPICH-G2[4],[5] 명령어를 이용하여 작성해야 하기에 네트워크 상태 정보들이 반영된 RSL 문서를 만들기는 쉽지 않다. 따라서 본 논문에서는 NWS를 통해 수집

된 네트워크 상태정보를 기반으로 노드분류와 노드별 성능비율을 계산하여 작업 프로세스 수를 결정한다. 마지막 단계에서는 이들 정보들을 RSL 문서에 자동으로 반영시켜 생성될 수 있게끔 새롭게 기능을 추가하였다.

네트워크의 성능요소 중 latency, bandwidth를 기반으로 작업에 참여할 노드들을 계층적으로 구성하는 연구들이 제안되었다[6],[7],[8],[9]. 각 노드들의 네트워크 상태정보를 기반으로 크게 LAN, WAN 영역으로 구분하여 계층적 토폴로지를 구성한 후 네트워크 상태에 맞는 노드간 통신을 함으로서 성능을 향상시키고 있다. 하지만 각 노드에 생성되어 작업을 수행하는 작업 프로세스 수에 관한 내용은 고려하지 않고 있으며, 또한 latency 및 bandwidth 정보 수집 및 계층적 토폴로지 구성 작업들이 응용 프로그램이 수행되는 시점에서 이루어진다. 즉 전체 수행시간에서 latency, bandwidth 측정시간과 계층적 토폴로지 구성 시간 등이 불필요하게 반영되는 문제점이 있다. 그래서 본 논문에서는 수집된 네트워크 상태정보인 latency, bandwidth를 기반으로 노드 분류, 노드별 성능비율 계산 등을 이용하여 노드별 최적의 작업 프로세스 수를 결정하는 방법을 제안한다. 또한 정보수집과 성능비율 계산 작업은 응용 프로그램의 수행과 별개로 진행하여 전체 계산 시간에 불필요하게 반영되는 문제점들을 해결한다.

III. 제안된 방법

본 절에서는 네트워크 상태 정보 중 latency, bandwidth, latency-bandwidth 혼합 정보를 기반으로 노드별 성능비율 계산과 작업 프로세스 수를 결정하는 방법을 알아본다. 또한 네트워크 성능요소인 latency, bandwidth 정보 수집 및 분류 방법과 시스템 구조를 살펴본다.

1. 노드별 작업 프로세스 수 계산 방법

1.1 노드별로 균등하게 프로세스 생성

전체 노드 수를 N이라 할 때 하나의 노드는 작업에 참여하지 않고 요청 및 결과 값만을 확인하는 순수 클라이언트이므로 작업에 참여한 노드 수는 N-1이 된다. 기존 방법에서는 사용자가 요청한 전체 작업 프로세스 수를 P라고 할 때 각 노드에 네트워크 상태정보를 고려하지 않고 요청한 프로세스 수를 N-1개의 노드 수만큼 균등(Uniform)하게 작업 프로세스들을 생성한다. 식 (1)은 노드별로 균등하게 작업 프로세스 수를 결정하는 수식을 나타낸다.

-node_N_i_process: 노드 i에 생성될 프로세스 수

$$node\_N_i\_process = ROUND\ (\ P\ /\ N\text{-}1\ ) \quad (1)$$

1.2 latency를 고려한 노드별 프로세스 수 계산

작업에 참여한 노드 간 latency 정보를 측정하여 시간이 작은 노드 즉 성능비율이 높은 노드에 많은 작업 프로세스를 생성하여 작업을 수행하는 방법이다.

성능요소별 분류 중 latency 정보를 기반으로 노드별 성능비율을 계산한다. 그림 1은 사용자가 요청한 프로세스 수를 노드별 성능비율에 적용하여 생성될 작업 프로세스 수를 결정하는 방법을 나타낸다.

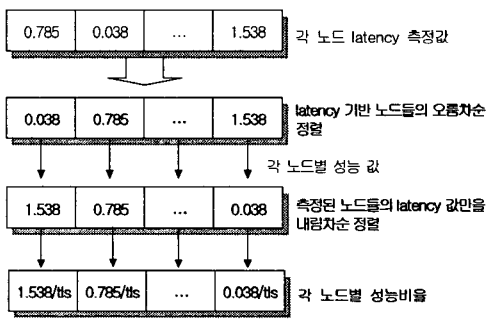


그림 1. latency를 고려한 노드별 프로세스 수 계산방법

latency 값은 작을수록 성능이 우수하므로 노드별 성능비율을 구하기 위해서는 먼저, latency 값을 기반으로 노드들을 오름차순 정렬을 한다. 또한 latency 값이 작은 노드에 높은 성능비율을 반영하기 위해서 측정된 latency 값에 대해서만 내림차순 정렬을 한다. 그래서 내림차순 한 latency 값이 오름차순 한 각 노드의 성능 값이 되어 전체노드의 latency 총합으로 나누면 노드별 성능 비율이 계산된다. 마지막으로, 사용자가 요청한 전

체 작업 프로세스 수를 노드별 성능비율에 적용하면 해당 노드의 작업 프로세스 수가 결정된다. 식 (2)는 노드별 latency를 고려한 작업 프로세스 수를 계산하는 수식을 나타낸다.

-total_latency_sum (tls) :
측정된 전체 노드들의 latency값의 총합
-latency_ascending_sort[i] :
latency 기반 노드들의 오름차순 정렬
-latency_ascending_sort[i].process :
노드 i에 생성될 프로세스 수
-latency_descending_sort[i] :
측정된 노드들의 latency 값만을 내림차순 정렬

$$latency\_ascending\_sort[i].process = ROUND(P * (latency\_descending\_sort[i] / total\_latency\_sum)) \quad (2)$$

1.3 bandwidth를 고려한 노드별 프로세스 수 계산

계산 작업에 참여한 노드 간 bandwidth를 측정하여 값이 큰 노드에는 상대적으로 다른 노드에 비해 많은 작업 프로세스를 생성하여 작업을 수행하는 방법이다.

bandwidth 정보를 기반으로 노드별 bandwidth 값을 전체 bandwidth 총합으로 나누어 성능비율을 계산한다. 즉 그림 2와 같이 요청한 전체 작업 프로세스 수에 노드별 성능비율을 적용하면 해당 노드의 작업 프로세스 수가 된다. 식 (3)은 노드별 bandwidth를 고려한 작업 프로세스 수를 결정하는 수식을 나타낸다.

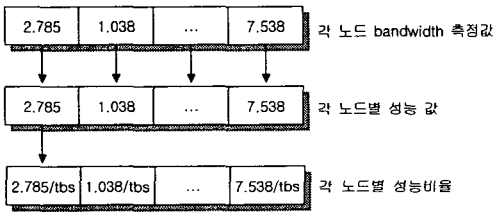


그림 2. bandwidth를 고려한 노드별 프로세스 수 계산방법

- total_bandwidth_sum (tbs) :
전체 노드들의 bandwidth값의 총합
- node_Ni_bandwidth :
노드 i의 측정된 bandwidth 값

$$node\_Ni\_process=ROUND(P * (node\_Ni\_bandwidth/total\_bandwidth\_sum)) \quad (3)$$

1.4 latency-bandwidth를 혼합한 프로세스 수 계산

latency, bandwidth 성능정보를 혼합하여 노드별 작업 프로세스 수를 계산하는 방법이다. 그림 3은 이전에 계산된 노드별 latency, bandwidth 성능비율을 혼합하여 각 노드의 작업 프로세스 수를 결정하는 방법을 나타낸다. 혼합한 성능요소의 수를 C라고 하면 본 논문에서는 latency와 bandwidth를 혼합하였기에 2가 된다. 노드별 latency와 bandwidth의 성능비율의 합을 C로 나누면 해당 노드의 latency-bandwidth를 혼합한 성능비율이 계산된다. 따라서 요청한 전체 작업 프로세스 수에 혼합한 성능비율을 적용하면 해당 노드의 작업 프로세스 수가 된다. 식 (4)는 노드별 latency-bandwidth를 고려한 작업 프로세스 수를 계산하는 수식을 나타낸다.

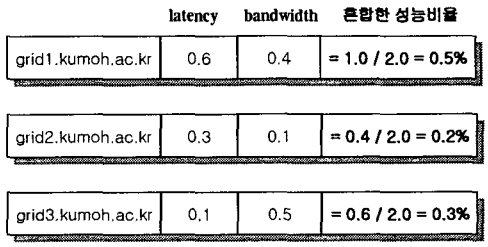


그림 3. latency-bandwidth를 혼합한 프로세스 수 계산 방법

-N_i_latency_bandwidth_sum :
노드 i의 latency-bandwidth의 성능비율의 합
-N_i_latency_percentage :
노드 i의 latency 성능비율 값
-N_i_bandwidth_percentage :
노드 i의 bandwidth 성능비율 값
-N_i_latency_bandwidth_percentage :
노드 i의 latency-bandwidth를 혼합한 성능비율 값

-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 (4)
(P* N_i_latency_bandwidth_percentage)

2. 시스템 구성

2.1 네트워크 상태정보 수집 및 분류

본 논문에서는 네트워크 상태정보 수집을 위해서 NWS를 이용한다. NWS는 상태정보 외 성능정보 예측 기능이 포함되어 있다. 그림 4와 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.

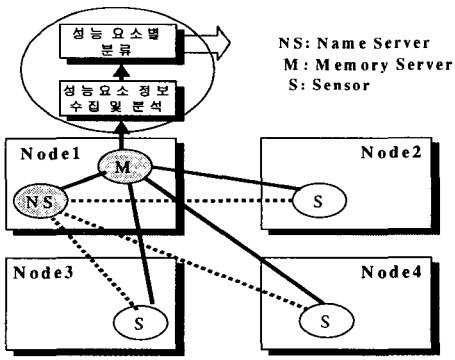


그림 4. NWS기반 성능요소별 노드 분류

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

2.2 시스템 구조

그림 5는 NWS로부터 수집한 정보를 기반으로 성능요소별 분석 및 분류를 통해 노드별 성능비율을 계산한 후, 성능비율에 따른 작업 프로세스 수를 결정하여 최종적인 RSL 파일을 자동으로 만들어 수행하는 것을 보여준다.

기존 그리드 시스템 사용자들은 노드별 작업 프로세스 수를 결정하기 위해 일반적인 노드들의 정보만을 이용하여 각 노드에 균등하게 작업 프로세스들을 생성하였다.물론 노드들의 성능 정보를 스크립트 파일에 반영하기 위해서는 사용자의 또 다른 노력이 필요하다. 그리고 노드별로 수행할 작업내용을 담고 있는 RSL 파일을 생성하기 위해서는 사용자가 그림 6의 (a), (b)와 같이 직접 스크립트 문서를 작성하거나 MPICH-G2 명령어를 통해 그림 6의 (c)와 같은 RSL 파일을 만들어 수행해야하는 불편함이 있다.

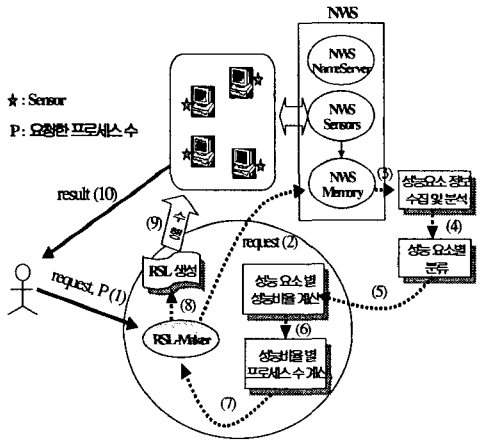


그림 5. 시스템 구조

```
"grid1.kumoh.ac.kr" 10
"grid2.kumoh.ac.kr" 10
```

(a) machines 파일 내용

```
mpirun -dumprsl -np 20 calculation_20 > calculate.rsl
mpirun -globusrsl calculate.rsl
```

(b) RSL 스크립트 파일 생성 및 작업 수행

```
+
( &(resourceManagerContact="grid1.kumoh.ac.kr")
(count=10)
(label="subjob 0")
(environment=(GLOBUS_DUROC_SUBJOB_INDEX 0)
(LD_LIBRARY_PATH /usr/local/globus/lib/))
(directory="/home/shcho/Evaluation")
(executable="/home/shcho/Evaluation/calculate_20")
)
( &(resourceManagerContact="grid2.kumoh.ac.kr")
(count=10)
(label="subjob 10")
(environment=(GLOBUS_DUROC_SUBJOB_INDEX 1)
(LD_LIBRARY_PATH /usr/local/globus/lib/))
(directory="/home/shcho/Evaluation")
(executable="/home/shcho/Evaluation/calculate_20")
)
```

(c) 생성된 RSL 파일

그림 6. 기존 글로버스 툴 킷 수행 방법

그러나 본 논문에서는 사용자가 수행 요청과 함께 계산 작업을 위해 필요한 전체 작업 프로세스 수만을 전달한다. 나머지 모든 단계들은 시스템에서 자동으로 수집된 정보들이 반영된 RSL 문서를 생성하여 작업을 수행한 후 최종적인 결과 값을 사용자에게 전달한다. 즉, 그림 7과 같이 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 그림 8과 같은 RSL 문서를 자동으로 생성하여 실질적인 계산 작업을 수행한다. 또한 그림 5의 2-7 단계 수행절차는 사용자 요청과 별개로 진행하여 네트워크 성능정보 수집 및 분류 등과 같이 순수 계산 작업 시 불필요하게 소요되는 시간을 단축한다.

```
Algorithm Make_Automatic_RSL(Node: 작업에 참여한 노드)
{
  /* 노드별 latency 측정*/
  latency = Get_Latency();

  /* 노드별 bandwidth 측정*/
  bandwidth = Get_Bandwidth();

  /* 측정된 정보 기록*/
  Write_Latency_Bandwidth(latency, bandwidth);

  /* 측정된 latency 평균값 계산*/
  Average_of_Latency(latency, Node);

  /* 측정된 bandwidth 평균값 계산*/
  Average_of_Bandwidth(bandwidth, Node);

  /* 노드별 성능비율과 프로세스 수 계산*/
  Calculate_of_Process(Node);

  /* 성능정보가 반영된 RSL 파일 생성*/
  Make_RSL(rsl, Node)
}
```

그림 7. 네트워크 성능정보 기반 RSL 파일 생성 방법

```
+
( &(resourceManagerContact="grid1.kumoh.ac.kr")
(count=1)
(label="subjob 0")
(environment=(GLOBUS_DUROC_SUBJOB_INDEX 0)
(LD_LIBRARY_PATH /usr/local/globus/lib/))
(directory="/home/shcho/Evaluation")
(executable="/home/shcho/Evaluation/calculate_128")
)
( &(resourceManagerContact="grid2.kumoh.ac.kr")
(count=10)
(label="subjob 1")
(environment=(GLOBUS_DUROC_SUBJOB_INDEX 1)
(LD_LIBRARY_PATH /usr/local/globus/lib/))
(directory="/home/shcho/Evaluation")
(executable="/home/shcho/Evaluation/calculate_128")
)
( &(resourceManagerContact="grid3.kumoh.ac.kr")
(count=5)
(label="subjob 11")
(environment=(GLOBUS_DUROC_SUBJOB_INDEX 2)
(LD_LIBRARY_PATH /usr/local/globus/lib/))
(directory="/home/shcho/Evaluation")
(executable="/home/shcho/Evaluation/calculate_128")
)
```

그림 8. 네트워크 성능정보가 반영되어 자동 생성된 RSL파일

IV. 실험결과 및 분석

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 NWS 2.8.1, iperf-1.7.0 [10]
3. CPU/Memory : P4 1.7G(512M), PIII 666M(128M), PIII 733M(192M)

실험에 참여한 노드들은 10Mbps 이더넷으로 연결된 5대의 노드들이며 성능평가를 위해 다양한 작업 프로세스 수와 행렬 곱셈계산을 수행하였다. 작업 프로세스 수와 작업량에 대한 실험은 3대의 노드에서 수행되었으며 그림 9와 같이 노드 수에 따른 실험은 5대까지 확장하여 실험하였다. 본 논문에서는 기존 연구[11],[12]에서 LAN /WAN 환경에서의 계산 작업 시 각 노드의 성능정보 보다 네트워크 상태 정보를 고려하여 반영하는 것이 전체 성능을 좌우하기 때문에 노드의 성능에 따른 실험은 고려하지 않았다. 따라서 본 논문에서는 네트워크 상태정보를 적용하여 작업 프로세스 수를 결정하기 위한 실험환경을 다음 2가지로 나누어 수행한다.

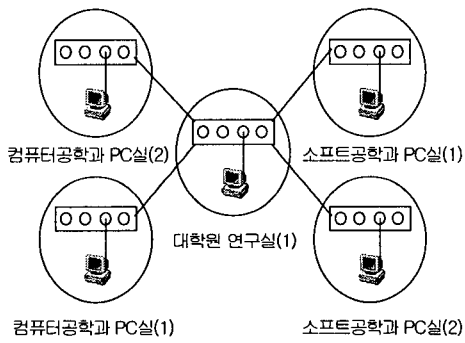


그림 9. 성능평가 실험환경

■ 네트워크 상태가 정적인 경우

본 논문의 실험 평가를 위한 첫 번째 방법은 네트워크 상태변화가 일정한 정적인 상태이다. 그래서 특별히 네트워크 상태에 대한 임의적인 조작 없이 다양한 수의 작업 프로세스에 대한 실험평가와 여러 크기의 작업량에 대한 실험을 균등, latency, bandwidth, latency-bandwidth 방법들을 적용하여 평가한다.

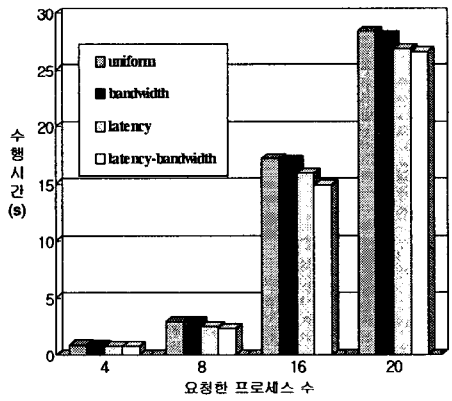
■ 네트워크 상태가 동적인 경우

실질적인 WAN 환경은 네트워크의 상태변화가 동적으로 변화한다. 본 논문에서는 네트워크 상태변화를 위해 네트워크 트래픽을 발생시키는 iperf를 이용하여 동적인 네트워크 환경을 구성하였다. 평가방법은 정적인 방법과 같이 프로세스 수와 작업량에 대한 실험 외 노드 수에 따른 실험을 추가적으로 평가한다.

2. 실험 결과

2.1 작업 프로세스 수에 따른 결과

그림 10은 정적인 네트워크 환경에서 작업량을 16*16 행렬로 고정하여 사용자가 요청한 작업 프로세스 수를 증가하면서 4가지 평가방법을 적용한 결과이다.



작업량 크기 : 16*16

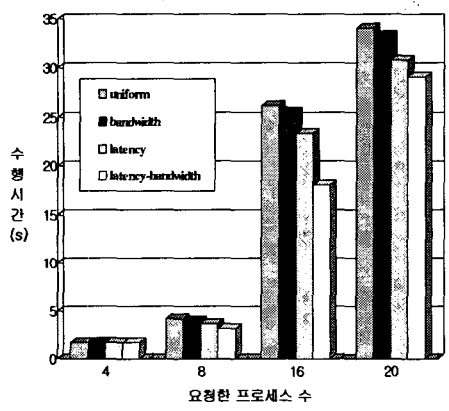
그림 10. 네트워크 상태가 정적인 환경에서의 수행결과

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 각각 6%, 13% 성능이 향상되었음을 알 수 있다. 이는 작업량이 작은 데이터를 프로세스 간 전송을 통해

작업 및 결과 값들을 송수신 하므로 bandwidth 보다 latency를 고려하여 노드별 작업 프로세스 수를 결정하는 것이 성능이 향상됨을 알 수 있다.

그림 11은 네트워크 상태가 동적인 환경에서 사용자가 요청한 프로세스 수를 증가하면서 4가지 평가방법을 적용한 결과를 나타낸다. 정적인 상태와 같이 요청한 프로세스 수가 4일 때 별다른 성능차이는 없다. 그러나 프로세스 수가 증가할수록 정적인 환경에 비해 성능향상 폭이 컸으며 latency, latency-bandwidth를 고려한 경우가 다른 2가지 방법에 비해 각각 9%, 31% 성능이 향상됨을 알 수 있다. 특히 요청 프로세스 수가 16일 때 노드 간 네트워크 상태변화가 급변할 경우에는 latency -band width를 혼합한 성능이 탁월함을 알 수 있다.

작업 프로세스 수 관점에서 네트워크 상태가 정적-동적인 환경모두에서, 수행하는 작업량이 작고 요청 프로세스 수가 증가할수록 latency, latency-bandwidth를 고려한 방법의 성능이 향상되었음을 확인하였다. 이것은 WAN 으로 구성된 그리드 환경에서 작업량이 작을 경우 데이터 전송에 영향을 미치는 bandwidth보다 노드 간 통신에 영향을 미치는 latency를 고려하는 것이 그리드 계산 작업의 전체 성능에 많은 부분 반영되기 때문이다.



작업량 크기 : 16*16

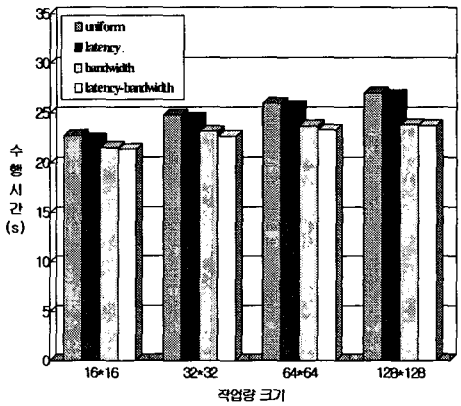
그림 11. 네트워크 상태가 동적인 환경에서의 수행결과

2.2 작업량에 따른 결과

그림 12는 정적인 네트워크 환경에서 작업 프로세스 수가 16일 때, 다양한 크기의 작업량을 증가하면서 4가지 평가방법을 적용한 결과를 보여준다.

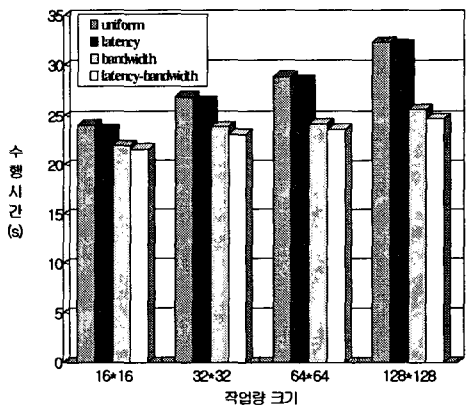
네트워크 변화가 일정한 상태에서 작업량이 작을 경우 큰 폭의 성능차이는 없었으며 때로는 작업량 16*16, 32*32 에서는 균등 방식이 좋은 성능을 보였다. 하지만 점차 작업량을 증가시킬 경우 균등, latency방식보다 bandwidth, latency-bandwidth를 고려한 방법이 8%, 9% 성능이 향상되었음을 알 수 있다.

그림 13은 동적인 네트워크 환경에서 동일한 프로세스 수에 다양한 크기의 작업량에 대해 4가지 평가방법을 적용하여 수행한 결과를 보여준다.



작업 프로세스 수 : 16

그림 12. 네트워크 상태가 정적인 환경에서의 수행결과



작업 프로세스 수 : 16

그림 13. 네트워크 상태가 동적인 환경에서의 수행결과

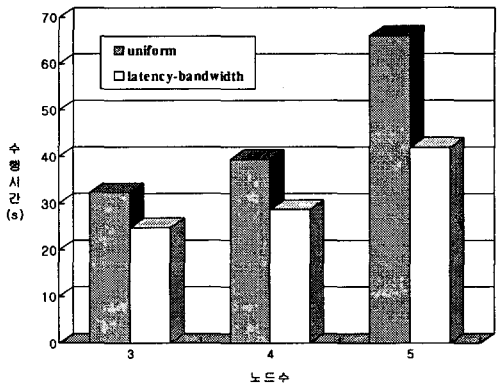
네트워크 상태가 동적인 상태이므로 작업량이 작은 경우에서도 소폭의 성능차이가 있다. 특히 네트워크 상태를 고려하지 않은 균등방식과 latency 방식보다 작업량 전송에 영향을 미치는 bandwidth를 고려한 경우가 성능이 우수하였다. 그리고 점차 작업량이 증가할수록 정적인 환경에 비해 성능향상 폭이 컸으며 균등, latency보다 bandwidth, latency-bandwidth를 고려한 경우가 21%, 23% 성능이 향상됨을 확인 할 수 있다.

정적-동적 네트워크 환경모두에서 동일한 작업 프로세스 수에서 작업량을 증가할수록 bandwidth, latency-bandwidth를 고려한 방법이 다른 방법보다 성능이 향상됨을 알 수 있다. 이는 동일한 프로세스 수에서 수행할 작업량과 계산된 결과 값들이 네트워크를 통해 송수신 된다. 따라서 네트워크를 통해 데이터 전송에 영향을 미치는 bandwidth를 고려하는 것이 그리드 계산 작업의 전체 성능에 많은 부분 반영되기 때문이다.

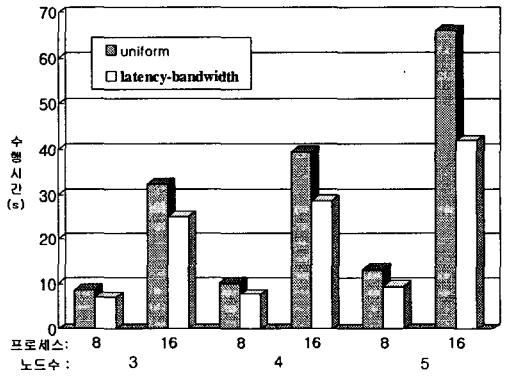
2.3 노드 수에 따른 결과

그림 14는 동적인 네트워크 환경에서 노드 수를 증가하였을 때 작업 프로세스 수와 작업량에 대한 실험결과를 보여준다. 작업 프로세스 수가 16이고 작업량이 128*128인 상황에서 노드 수를 증가 하였을 때 균등방식에 비해 네트워크 상태 정보를 반영하여 적용한 결과가 23~57% 성능이 향상됨을 확인할 수 있다. 또한 노드 수 증가에 따른 작업량의 크기와 작업 프로세스 수의 증가 시에도 균등방식보다 네트워크 상태정보를 고려한 방법이 큰 폭으로 성능이 향상됨을 확인 할 수 있다.

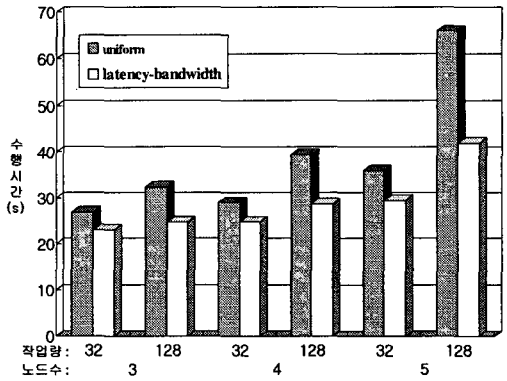
이는 계산 그리드 환경에서 노드 수, 작업량, 작업 프로세스 수의 증가와 네트워크 환경이 동적일 때 각 노드에 생성된 프로세스 간 작업 및 결과 값을 전송함에 있어 통신시간이 전체 수행시간에 많은 부분 반영이 되기 때문이다. 따라서 노드 수가 증가하여 각 노드의 작업 프로세스 수를 결정함에 있어 latency-bandwidth를 고려하는 것이 성능향상에 중요한 영향을 끼친다는 것을 실험을 통해 확인할 수 있다.



(a) 작업프로세스 수 : 16, 작업량 : 128*128



(b) 작업량 : 128*128



(c) 작업 프로세스 수 : 16

그림 14. 네트워크 상태가 동적인 환경에서 노드 수에 따른 수행결과

V. 결론 및 향후 연구

그리드 컴퓨팅은 WAN 환경으로 구성되어 네트워크를 통해 작업 및 결과 값이 송수신 됨으로 네트워크의 상태정보 반영이 무엇보다 중요하다. 본 논문에서는 네트워크 상태정보 중 latency, bandwidth, latency-bandwidth 혼합 정보를 이용하여 노드별 분류 및 성능 비율을 계산한 다음 이를 기반으로 노드별 작업 프로세스 수를 결정한 후 자동으로 RSL 파일을 생성하여 작업을 수행하는 방법을 제안한다. 또한 NWS에 의해 수집된 정보를 성능요소별 분석 및 분류 기능을 새롭게 추가하여 적용하였다.

실험 결과 작업 프로세스 수 관점에서 정적인 네트워크 환경보다 동적인 환경에서 네트워크 상태정보를 고려하지 않은 균등방식보다 latency, latency-bandwidth를 고려한 방법이 각각 9%, 31% 성능 향상을 보였다. 또한 작업량 관점에서도 정적인 네트워크 환경보다 동적인 환경에서 bandwidth, latency-bandwidth를 고려한 방법이 다른 2가지 방법보다 21, 23% 성능이 향상되었다. 그리고 노드수의 증가에 따른 실험에서도 작업량과 작업 프로세스 수가 클수록 균등방식보다 네트워크 상태정보를 반영한 latency-bandwidth 혼합방법이 57% 성능이 향상되었다.

따라서 계산 그리드 환경에서 계산 작업을 함에 있어 작업량이 작고 프로세스 수가 많을 경우 latency를 고려하는 것이 성능향상에 많은 영향을 주고 프로세스 수는 일정하고 작업량이 증가할 경우에는 bandwidth를 고려하는 것이 전체 성능을 좌우한다는 것을 실험을 통해 확인하였다. 또한 노드 수, 작업량, 작업 프로세스 수가 증가 하였을 경우에는 latency, bandwidth를 혼합하여 적용하면 성능이 향상됨을 확인하였다.

향후 연구에서는 그리드 환경에서 작은 수의 작업 프로세스 수를 갖고도 성능을 향상시킬 수 있는 방법에 대해 연구 할 예정이다.

참 고 문 헌

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

[2] RSL,http://www-fp.globus.org/gram/rsl_spec1. html

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

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

[5] MPICH-G2, http://www3.niu.edu/mpi

[6] K. L. Park, H. J. Lee, Y. J. Lee, O. Y. Kwon, S..Y. Park, H. W. Park and S. D. Kim, "An Efficient Collective Communication Method for Grid Scale Networks," International Conference on Computational Science, pp. 819-828, 2003.

[7] T. Kielmann, H. E. Bal and S. Gorlatch, "Bandwidth-efficient Collective Communica-tion for Clustered Wide Area Systems," In Proc. International Parallel and Distributed Processing Symposium, pp. 492-499, 2000.

[8] 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.

[9] N. Karonis, B. de Supinski, I. Foster, W. Gropp, E. Lusk and J. Bresnahan, "Exploiting Hierarchy in Parallel Computer Networks to Optimize Collective Operation Performance," Proceedings of the 14th International Parallel Distributed Processing Symposium (IPDPS '00), pp. 377-384, 2000.

[10] iperf-1.7.0, http://dast.nlanr.net/Projects/iperf/

[11] 조수현, 김영학, "프로세스의 수와 실행시간에 따른 NOW의 성능 분석", 한국콘텐츠학회 논문지, 제2권, 제3호, pp. 135-145, 2002.

[12] 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, 1999.

저 자 소 개

조 수 현(Soo-Hyun Cho) 정회원

- 2000년 : 금오공과대학교 컴퓨터공학과(공학사)
- 2002년 : 금오공과대학교 컴퓨터공학과(공학석사)
- 2002년~현재 : 금오공과대학교 컴퓨터공학과(박사수료)

<관심분야> : 병렬 및 분산처리, 이동 에이전트, 병렬 프로그래밍, Cluster Computing, Grid Computing

김 영 학(Young-Hak Kim) 정회원

- 1984년 : 금오공과대학교 전자공학과(공학사)
- 1989년 : 서강대학교 전자계산학과(공학석사)
- 1997년 : 서강대학교 전자계산학과(공학박사)
- 1989년~1997년 : 해군사관학교 전산과학과 교수
- 1998년~1999년 : 여수대학교 멀티미디어학부 교수
- 1999년~현재 : 금오공과대학교 컴퓨터공학부 부교수

<관심분야> : 병렬 알고리즘, 분산 및 병렬처리, 임베디드 시스템