이기종 워크스테이션 클러스터 상에서 부하 균형을 위한 효과적 작업 분배 방법

Efficient Task Distribution Method for Load Balancing on Clusters of Heterogeneous Workstations

지 병 준* 이 광 모**
Byoung-Jun Ji Kwang-Mo Lee

요 약

이기종 워크스테이션 클러스터링은 병렬 응용 처리에 유용하며 비용 측면에서 효과적이다. 이기종 워크스테이션 클러스터링 환경에서 최소의 전체 작업 반환 시간이 되도록 하기 위해서는 부하 균형 시스템이 필요하다. 다른 사용자들, 그룹들, 다른 작업 요구들, 그리고 다른 처리능력을 가지는 각 워크스테이션들은 클러스터링 환경의 다른 그것과 상대적이다.

기존 방식은 각 워크스테이션의 처리능력에 가중치를 미리 부여하는 정적 방식이거나 각 워크스테이션의 상대적 처리능력을 얻기 위해서 벤치마크 프로그램을 수행하는 동적 방식이다. 수행되는 응용과는 관계없는 벤치마크 프로그램은 계산시간을 소비하고 전체 작업 반환 시간을 지연시킨다.

이 논문은 효과적 작업 분배 방식을 제안하고 이기종 워크스테이션 클러스터링 환경에서 부하 균형 시스템을 구현한다. 이 논문에서 제안한 방식의 전체 작업 반환 시간은 부하 균형을 하지 않은 방식뿐만 아니라 벤치마크 프로그램에 의한 부하 균형 방식과도 비교한다. 실험을 통하여 비교한 방식보다 제안 방식의 결과가 우수함을 보인다.

Abstract

The clustering environment with heterogeneous workstations provides the cost effectiveness and usability for executing applications in parallel. The load balancing is considered as a necessary feature for the clustering of heterogeneous workstations to minimize the turnaround time. Since each workstation may have different users, groups, requests for different tasks, and different processing power, the capability of each processing unit is relative to the others' unit in the clustering environment

Previous works is a static approach which assign a predetermined weight for the processing capability of each workstation or a dynamic approach which executes a benchmark program to get relative processing capability of each workstation. The execution of the benchmark program, which has nothing to do with the application being executed, consumes the computation time and the overall turnaround time is delayed.

In this paper, we present an efficient task distribution method and implementation of load balancing system for the clustering environment with heterogeneous workstations. Turnaround time of the methods presented in this paper is compared with the method without load balancing as well as with the method load balancing with performance evaluation program. The experimental results show that our methods outperform all the other methods that we compared.

1. 서 론

네트워크 중심의 분산 워크스테이션 시스템에서 병렬 응용 프로그램을 처리하게 되면 값비싼 거대 병렬 컴퓨터 시스템 없이도 초고속 통신망에 연결된 워크스테이션들을 하나의 계산 자원으로 구성함으로서 비용측면에서 효과적이고 폭넓은 이식성을 바탕으로 한 이용측면에서도 높은 확장성을 장점으로 갖는다. 네트워크에 연결된 워크스테이션들을 묶어서 하나의 가상 병렬 컴퓨터를 구성하고 병렬 응용 프로그램을 처리하는 도

* 정회원 : 한림정보산업대학 전산정보처리과 부교
bjji@sun.hallym-c.ac.kr
** 비회원 : 한림대학교 정보전자공과대학 컴퓨터공학부 교수
kmlee@sun.hallym.ac.kr

구로 P4, Express, LINDA, PVM 등이 개발되어 있다[1,2,3,4]. 특히, PVM은 이기종 환경에서 병렬처리를 지원하는 워크스테이션 클러스터 시스템 구성을 지원한다. 그러나 병렬처리를 위한 작업 분배를 워크스테이션의 작업부하를 고려하지 않고 라운드 로빈 방식으로 처리하고 있다[10]. 그러므로 워크스테이션 클러스터 시스템 중의 한 워크스테이션이 과부하 상태일지라도 작업을 분배 받게되고 이로 인해 전체 병렬 응용 프로그램의 처리 성능이 저하된다. 이러한 문제를 해결하기 위해서 부하 균형 시스템이 필요하며 이와 관련한 연구는 공유 메모리 다중 프로세서와 분산 시스템으로 구분되어 연구되었고 분산 시스템의 경우는 동기종 프로세서 분산 환경과 이기종 프로세서 분산 환경으로 구분하여 연구되었다. 부하 균형을 위한 작업의 분배 결정 시기는 작업 분배시 이전의 정보를 토대로 분배를 결정하는 정적 방식과 현재의 정보를 토대로 분배된 작업의 이주와 재시작을 수시로 하는 동적 방식이 있다. 이 분배 결정을 분산 시스템 내의 어느 워크스테이션이 하는가에 따라서 주종관계의 중앙식과 모든 워크스테이션들이 결정하는 분산식 방식이 있다. 기존에 가장 많이 개발되어진 부하 균형 시스템은 중앙식 정적 접근 방식 시스템으로 Load Sharing Facility (LSF), Utopia, Distributed Queuing System (DQS), Portable Batch System(PBS), Prospero Resource Manager(PRM) 등이 있다[9].

특히, 이기종 워크스테이션 클러스터 시스템에서 부하 균형 시스템에 적용할 각 워크스테이션의 처리 능력을 계산하는 기존의 연구에서는 구성 워크스테이션의 프로세서 성능을 미리 가중치로 환산하는 정적 방식이거나, 병렬 응용 프로그램 수행과는 무관한 성능 테스트(benchmark) 프로그램을 주기적으로 수행하여[5,6,7,8] 얻은 결과를 이기종 워크스테이션 클러스터 시스템 구성원들의 상대적 처리 능력으로 사용하는 방법이다. 즉 성능 테스트 프로그램 의존적 부하 균형 방식이다.

이 논문은 첫 번째, PVM을 도구로 하여 이기종의 워크스테이션들로 가상의 병렬 컴퓨터를 구성하고 균등한 작업 분배를 위해 각 워크스테이션들의 작업 처리 수행 능력과 부하 정도를 파악하고 취합하는 부하 균형 시스템(load balancing system)을 구성한다. 부하 균형 시스템은 이기종 환경에서 작업 제출 시간에 작업과 워크스테이션이 맵핑되고 동적인 이주는 고려하지 않은 정적 방식으로 구성한다. 자신의 부하 정보를 파악하는 부하 모니터(Load Monitor)와 각 워크스테이션의 부하 모니터에서 보내오는 부하 정보를 취합하여 균형적으로 작업을 분배하는 부하 관리자(Load Manager)를 주종관계의 중앙식으로 구성한다. 또한 병렬 응용 프로그램과 부하 관리자 간의 인터페이스를 제공한다.

두 번째, 부하 균형 시스템에 적용할 이기종 워크스테이션들의 상대적 처리 능력을 성능 테스트 프로그램에 의존하지 않고 동적으로 분석하여 균형적으로 작업을 분배하는 알고리즘을 제안한다. 이기종 환경에서 동적 분석에 기초한 균형적 작업 분배 알고리즘은 서로 다른 사용자들과 그룹들, 다른 외부 작업 수행 요구들, 다른 처리능력을 갖는 프로세서로 구성된 이기종 워크스테이션 클러스터에서 상대적 처리 능력을 비교하고 가중치로 표시된 상대적 처리 능력에 따라서 부하 정도를 파악하여 작업을 분배한다. 기존의 성능 테스트 프로그램 의존적 방식은 그 시간만큼 전체 작업 반환 시간이 길었다. 이 논문은 수행하려는 병렬 응용 프로그램의 동일한 실행 코드의 작업들을 바로 사용하여 이기종의 상대적 처리 능력을 비교하는 성능 테스트 프로그램 독립적 방식의 상대적 처리 능력 측정(Relative Processing Capacity Measurement) 알고리즘과 균등한 작업 분배(Task Distribute) 알고리즘을 제안한다.

세 번째, 이기종 분산 환경에서 응용 프로그램의 동일 실행 코드가 성능 테스트 프로그램 수행의 결과를 대신하여 상대적 처리 능력을 파악 할 수 있음을 보기 위한 실험과 제안한 분배 알고리

즘을 설계한 부하 균형 시스템에 적용하여 기존 방식보다 향상된 실험 결과를 얻었음을 보인다.

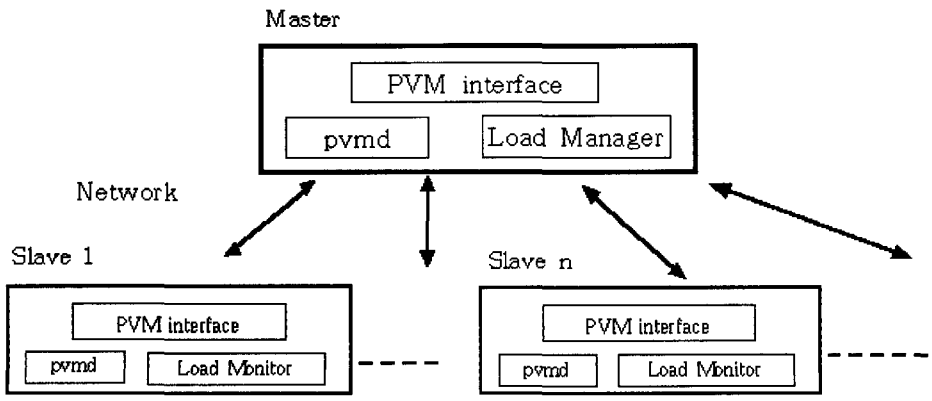
2. 부하 균형 시스템 설계

PVM을 도구로 하여 워크스테이션 클러스터 시스템을 구성하였으나 병렬 응용 프로그램 수행은 단순한 라운드 로빈 방식으로 작업을 분배하고 있다. 효과적인 작업분배를 통하여 워크스테이션 클러스터 시스템의 처리율을 향상시키기 위한 부하 균형 시스템을 설계한다.

PVM에 의해서 운영되는 이기종 워크스테이션 클러스터 시스템은 병행처리 가능한 SPMD(Single Program Multiple Data) 방식으로 동일 실행 코드의 병렬처리 작업들을 주(master) 워크스테이션에서 종(slave) 워크스테이션들에게 분배하고 처리하여 다시 주 워크스테이션이 취합하는 주종관계를 가진다. 이러한 시스템에서 병렬로 수행될 동일 실행 코드의 작업들은 무수히 많을 수 있으며 종 워크스테이션들은 작업 처리를 위한 프로세서의 사용만을 대여하게 된다[12].

부하 균형 시스템은 전송상대를 선택할 때, 조정자(coordinator) 역할을 하는 주 워크스테이션 중심으로 한다. 즉 중앙 집중식의 관리자를 구성하고 전송할 작업의 선정도 맡도록 한다. PVM을 기반으로 한 부하 균형 시스템의 구성은 그림 1과 같다. pvmd 데몬은 워크스테이션 클러스터 시스템을 구성하기 위해서 모든 워크스테이션들에 위치하고 주 워크스테이션에는 부하 관리자(load manager)를 두어 종 워크스테이션들에 위치한 부하 모니터(load monitor)가 보내는 부하 정보를 수집한다. 각 워크스테이션의 부하 정도를 표시하는 방법은 Unix 커널에서 제공하는 uptime이나 top의 평균 부하량(Load Average)이나 vmstat의 user time, system time, idle time을 이용한다. 즉 종 워크스테이션의 부하 모니터는 vmstat을 통해서 프로세서의 유휴(idle) 시간의 비율을 부하 정보로 취하고 주 워크스테이션의 부하 관리자로 보내거나 uptime이나 top의 부하 평균값을 취하여 주 워크스테이션으로 보낸다. 시스템의 부하정도를 표현하는 부하 색인으로서 평균 부하량을 가장 많이 사용하고 있다[11]. 주 워크스테이션의 부하 관리자로 전송이 완료되면 부하 모니터는 일정한 간격으로 부하 정보를 취하여 새로이 부하 관리자와 연결된다.

주 워크스테이션의 부하 관리자는 부하 균형 시스템의 초기화를 수행하고 부하 모니터로부터 부하 정보를 수집, 저장하고 구성 워크스테이션들의 상대적 부하 정도를 평가한다. 그리고 그 값을 기초로 수행할 병렬 작업들이 최소의 반환 시간이 만들어 질 수 있도록 균형적으로 분배한다. 주 워크스테이션의 부하 관리자의 수행과정과 종 워



(그림 1) 부하 균형 시스템

크스테이션의 부하 모니터 수행과정은 표 1, 표 2와 같다.

(표 1) 부하 관리자 수행과정

```
for (j=0; j<# of workstation in Cluster; j++ )
    { workstation[j] 에 초기작업 spawn }
for (j=0; j<# of workstation in Cluster; j++ )
    { j번째 Workstation에서 부하정보를 전송 받아
      load_val[j]에 저장 }
load_avg[i]의 값을 이용한 상대적 처리 능력 측정
부하 균형을 위해 각 워크스테이션에 분배할 작업 결정
for (j=0; j<# of workstation in Cluster; j++ )
    { workstation[j]에 균형 작업 spawn }
```

(표 2) 부하 모니터 수행과정

```
if ( 0 == fork())
    { load_average변화값 측정
      부하관리자에 전송 }
else
    { 작업 수행 }
```

3. 이기종 부하 균형 알고리즘

동기종으로 워크스테이션 클러스터 시스템을 구성하고 부하 균형을 하기 위해서는 가장 일반적인 평균 부하량만을 각 컴퓨터에서 받아서 그 컴퓨터의 현재의 상대적 성능 정도, 즉 부하 색인을 파악할 수 있다. 그러나, 프로세서의 처리 능력 자체가 다른 이기종간에 이루어진 워크스테이션 클러스터의 부하 균형을 위해서는 그 워크스테이션의 현재 평균 부하량만으로는 다른 워크스테이션과의 상대적 처리 능력 정도를 비교할 수 없다. 또한 워크스테이션 클러스터 시스템에 작업이 주어졌을 때, 최소의 반환시간으로 처리될 수 있도록 균형적 작업의 분배 방식도 중요하다.

기존 방식으로는 워크스테이션 클러스터를 구성하는 각 워크스테이션의 프로세서 성능을 미리 가중치로 환산하는 정적 방식이거나, 병렬 프로그램 수행과는 무관한 성능 테스트(benchmark) 프로그램을 주기적으로 수행하여 얻은 결과로 이기종 워크스테이션 클러스터 시스템의 구성원들의 상대적 처리 능력으로 사용하는 방식이므로 그 시간만큼 전체 작업 반환 시간이 길었다.

이기종 환경에서 기존의 많은 부하 균형 알고리즘은 다른 처리 능력을 갖는 프로세서들의 처리능력을 상대적으로 비교하기 위해서 반복적으로 각 프로세서에 처리하고자 하는 병렬 프로그램과는 상관없는 간단한 성능 테스트 프로그램을 수행해서 처리 능력을 측정하고 결과를 스케줄러에 통보하고 있다[6,7,8,9,13]. 이것은 보다 빠른 처리를 요구하는 병렬처리에서 치명적인 부담이 될 수도 있다. 이 논문에서는 이 문제를 보완한 상대적 처리 능력 측정(Relative Processing Capacity Measurement) 알고리즘과 상대적 처리 능력 측정 결과에 따른 병렬 작업 분배(Task Distribute) 알고리즘을 제안한다.

3.1 상대적 처리 능력 측정 알고리즘

동기종의 분산 환경을 통한 병렬처리에서는 각 워크스테이션이 동일한 처리능력을 가지므로 클러스터상의 각 워크스테이션들은 다중프로세서 시스템의 구성 프로세서와 같다고 할 수 있다. 그러나 이기종간의 분산 병렬처리를 위해서는 클러스터의 일원으로 구성된 각각의 프로세서들의 처리 능력이 다르므로 구성원 프로세서들을 한 시점에 상대 비교함으로서 서로의 차이를 구분해야 한다.

PVM에서 수행하는 병렬 처리 응용 프로그램은 SPMD의 구조를 가지는 동일한 실행 코드의 병렬 작업이 상당히 많이 발생하고 이것은 워크스테이션 클러스터 시스템의 각 워크스테이션들로 분산되게 된다. 분산되는 작업의 수는 클러스터의 구성 워크스테이션들의 수보다 상당히 많다. 워크스테이션간 메시지는 연속적으로 교환되고 메시지 크기에 통신부하는 비례하며, 작업의 시작과 끝은

같은 워크스테이션에서 하도록 한다[9].

부하 균형을 위해서는 각 워크스테이션의 처리 능력에 따른 작업 분배가 필수적이며, 기존 연구에서 각 워크스테이션들의 처리 능력을 파악하는 방법으로는 미리 가중치로 정하는 정적 방식을 사용하나 현재의 부하 정도를 파악하기 어려우며, 동적으로 현재의 부하 정도를 상대적으로 파악하기 위한 방법으로 벤치마크 프로그램을 수행하고 그 결과를 워크스테이션들의 상대적 처리 능력으로 처리하는 방법이 있다. 그러나 이 방법은 수행 작업과 무관한 성능 테스트 프로그램을 수행하여 그 만큼의 시간을 낭비할 수 있다. 이러한 문제를 해결하기 위한 상대적 처리 능력 측정 알고리즘은 워크스테이션 클러스터를 구성하는 모든 워크스테이션의 처리 능력을 상대 비교하는 기준자로써 지금 처리하고자 하는 동일 실행 코드의 작업을 분배하고 일정 시간에 각 워크스테이션의 부하 색인값의 변화를 이용하여 현재의 이기종 워크스테이션들의 상대적 처리 능력을 계산한다. 부하 색인값으로는 CPU의 유휴비율이나 평균 부하량 등을 사용할 수 있다. 준비 큐의 평균 대기 작업의 수인 평균 부하값(load average)은 부하균형에 가장 적절한 부하색인이다[11].

워크스테이션 클러스터 시스템을 Φ_{system}이라고 하고 X를 병렬 처리 프로그램에서 발생되는 SPMD의 동일 실행코드의 병렬처리 작업들의 개수라 하자. Φ_{system}내의 각 프로세서 p_i에 분배되는 작업의 수를 x_i라고 한다면

$$X = x_1 + \ldots + x_i + \ldots + x_n$$

이다.

각 프로세서에 분배되는 x_i중에서 하나의 작업만을 상대적 처리 능력을 평가하기 위해서 사용한다. 즉 워크스테이션 클러스터 시스템을 구성하는 n개의 프로세서에 하나씩 작업을 할당하고 병렬 처리 프로그램을 시작하면서 이기종 프로세서간의 상대적인 처리능력 또한 계산한다.

병렬 작업이 분배되어 수행되기 전인 t 시간의 특정 i 프로세서의 평균 부하값 id는 $id_i^{(t)}$로 한다면 분배된 동일 작업을 수행하고 난 후의 $t+1$ 시간의 프로세서 평균 부하값 id는 $id_i^{(t+1)}$로 나타낼 수 있다. 여기서 일정 시간동안의 프로세서 부하의 변화를 $\Delta id_i^{(t+1)}$로 한다.

$$\Delta id_i^{(t+1)} = id_i^{(t+1)} - id_i^{(t)}$$

Φ_{system}의 각 프로세서들은 $\Delta id_i^{(t+1)}$를 가진다. 모든 프로세서들에서 부하의 변화를 측정하였으면, 여기서 변화의 폭이 가장 크다는 것은 Φ_{system}에서 현재 부가된 작업을 처리하는데 가장 많은 부하가 생성되었음을 나타낸다. 즉 현재 가장 처리 능력이 떨어지는 프로세서로 간주할 수 있다. 가장 큰 폭의 변화를 보이는 프로세서를 1로 하고 i 프로세서의 상대적인 처리 능력 rpc_i(relative processing capacity)값을 구한다.

$$rpc_i = \frac{MAX[\Delta id_{i-1}^{(t+1)}, \Delta id_i^{(t+1)}, \Delta id_{i+1}^{(t+1)}, \ldots, \Delta id_n^{(t+1)}]}{\Delta id_i^{(t+1)}}$$

rpc_i는 가상 병렬 컴퓨터를 구성하는 각 호스트들 중에 종 프로세서 즉 부하 모니터로부터 부하 정보를 받아서 중앙의 조정자 역할을 하는 부하 관리자가 처리한다.

상대적 처리 능력 측정 알고리즘에서 얻어진 rpc_i는 Φ_{system}에서 프로세서 p_i의 상대적 처리 능력을 나타낸다.

$$rpc_i = \frac{Capacity(p_i)}{Capacity\ (slowest\ processor\ in\ \Phi_{system})}$$

3.2 작업 분배 알고리즘

작업 분배 알고리즘은 워크스테이션 클러스터 시스템에서 균형적 작업 분배를 통하여 최소의

전체 작업 반환 시간을 얻고자 한다.

Φ_{system}에서 Φ를 현재 사용 가능한 프로세서 집합이라면 Φ내의 각 프로세서 p_i는 병렬 수행할 전체 작업에서

$$\frac{rpc_i}{\sum_{\forall_i \in \Phi} rpc_i}$$

만큼의 비율로 작업을 할당받는다.

X의 수가 $X \geq \Phi$ 만큼이라면 p_i에 할당되는 작업의 실제 개수 $+x_i$는 다음과 같다.

$$x_i = X\left[\frac{rpc_i}{\sum_{\forall_i \in \Phi} rpc_i}\right] = \alpha rpc_i \ , \quad x_i \geq 1$$

여기서 x_i는 이기종의 상대적 처리능력 측정에 사용되는 작업을 포함하여 1 이상의 값을 갖는다. α는 작업의 가능한 최소 반환 시간으로 볼 수 있다. 즉 α는 다음과 같다.

$$\alpha = \frac{X}{\sum rpc_i}$$

또한 x_i와 α는 이상적인 값이므로 실제 구현에 적용하기 위해서는 정수 값을 유도하는 정수 유도 알고리즘이 필요하다.

프로세서 p_i에 할당된 작업의 완료시간을 T_i라 한다면, 실제 구현에 있어서 병렬작업 전체의 최소 반환 시간 $T_{\min}$은 다음과 같다.

$$T_{\min} = \underset{\forall_i \in \Phi}{Max} \{T_i\}$$

그러면 실제 구현을 위한 정수값 x_i를 구하고 $T_{\min}$을 최소화하는 방향으로 워크스테이션 클러스터 시스템의 워크스테이션에 작업을 할당하는 작업 분배 알고리즘의 정수 유도에 의한 분배 단계는 다음의 순서를 따른다.

우선 각각의 프로세서 p_i에 할당할 최적의 작업 개수인 x_i의 하한 정수 값에서 이미 상대적 처리능력을 측정하기 위해 기존의 성능 테스트 역할로 수행된 1개의 작업을 뺀 값을 취한다.

$$x_i' = \lfloor x_i \rfloor - 1 \ , \forall_i \in \Phi$$

다음은 워크스테이션 클러스터 시스템의 구성 프로세서들에게 정수개의 작업 x_i'를 부여하고도 아직 남아있는 작업들을 구한다. x_i'에 포함되지 않은 비할당 작업 개수 y_i'는 병렬 처리 해야할 총 작업의 개수에서 이미 각 프로세서들에 할당된 즉, Φ내의 n개의 작업과 총 할당 작업 x_i'를 뺀 값이다.

$$y_i' = X - (\sum_{\forall_i \in \Phi} x_i' + n)$$

y_i'개의 아직 할당되지 않은 작업을 추가로 프로세서에 분배하기 위해서는 먼저 프로세서에 x_i개의 작업을 분배하고 발생되는 반환시간의 변화를 구해야 한다. 즉 프로세서들에 할당된 추가 작업으로부터 추가로 발생되는 반환 시간은

$$PlusT_i = \frac{(\lceil x_i \rceil - x_i)}{rpc_i}, \ \forall_i \in \Phi$$

이다.

이제 $PlusT_i$에 기초해서 Φ내의 프로세서들 중에 $PlusT_i$가 작은 것에서부터 y_i'개의 작업을 프로세서에 할당한다.

물론 작업이 마지막으로 할당된 프로세서의 완료시간까지 고려된 최종의 병렬 작업의 최소 반환 시간인 $T_{\min}$이 수정된다. 즉 마지막 작업까지 프로세서에 할당한 프로세서 p_i의 최종 작업 개수를 x_{if}'로 한다면 p_i에서 이들 작업들이 완료하는 시간 T_{if}와 병렬처리가 완료된 최종의 최소 반환 시간 $T_{\min}$은 다음과 같다.

$$T_{if} = \alpha + \frac{(x_{if}' - x_i)}{rpc_i} \quad , T_{min} = \underset{\forall_i \in \phi}{Max} \{T_{if}\}$$

3.3 알고리즘의 적용 예

(표 3) 부하 평균값 변화량

	$id_i^{(t)}$	$id_i^{(t+1)}$	$\Delta id_i^{(t+1)}$
P1	0.00	0.02	2
P2	0.01	0.21	20
P3	0.02	0.07	5
P4	0.10	0.17	7

병렬 처리 응용 프로그램의 총 병렬 작업, 즉 X를 20개라고 하고 4대의 워크스테이션으로 클러스터 시스템을 구성한다. 먼저 상대적 처리 능력 측정 알고리즘에 의해서 4개의 프로세서에 동시에 동일 실행 작업이 1개씩 배분된다. 일정 시간의 각 프로세서의 유휴상태 변환은 다음과 같다. 구현 실험에서는 세 번을 반복적으로 유휴상태 변화를 구하여 변화의 평균값을 Δid_i에 적용하였다.

표 3의 경우에 상대적 처리 능력 측정 알고리즘에 의한 각 프로세서의 상대적 처리 능력 $\{ rpc_i \} = \{ 10, 1, 4, 2.86 \}$이 된다.

rpc_i를 토대로 작업 분배 알고리즘에서는 먼저, 가능한 최소 반환시간 α를 다음과 같이 구한다.

$$\alpha = \frac{X}{\sum rpc_i} = \frac{20}{17.86} = 1.119$$

구한 α을 이용하여 각 프로세서에 할당되는 VP개수인 $\{ x_i \}$는

$$\{ x_i \} = \{ rpc_i \ \alpha \}$$
$$= \{ 11.19, 1.119, 4.476, 3.200 \}$$

이다.

다음은 정수 값을 구하기 위한 정수 유도 알고리즘 과정을 수행한다.

첫 번째는 실제 구현에 적용하기 위한 정수값 x_i는

$$x_i' = \lfloor x_i \rfloor - 1 \ , \forall_i \in \phi$$

로 구하면

$$\{ x_i' \} = \{ 10, 0, 3, 2 \}$$

이다.

두 번째는 아직 할당되지 않은 작업 개수인 y_i'를 계산하면

$$y_i' = X - (\sum_{\forall_i \in \phi} x_i' + n)$$
$$= 20 - (15 + 4) = 1$$

이다. 남은 1개의 작업을 어느 프로세서로 분배할지를 결정해야한다.

세 번째 단계에서는 기존 프로세서 처리 능력에 첨부된 작업으로 인해서 발생하는 추가 반환 시간 계산은

$$PlusT_i = \frac{(\lceil x_i \rceil - x_i)}{rpc_i} \quad , \forall_i \in \phi$$

$$\text{(P1 경우)} = \frac{12 - 11.19}{10} = 0.081$$

$$\text{(P2 경우)} = \frac{2 - 1.119}{1} = 0.881$$

$$\text{(P3 경우)} = \frac{5 - 4.476}{4} = 0.131$$

$$\text{(P4 경우)} = \frac{4 - 3.20}{2.86} = 0.279$$

$$\{ PlusT_i \} = \{ 0.081, 0.881, 0.131, 0.279 \}$$

이다.

두 번째와 세 번째 단계의 y_i'와 $PlusT_i$을 가지고 $PlusT_i$가 작은 것에서부터 y_i'를 할당한다. P1의 $PlusT_i$가 가장 작으므로 남은 한 개의 프로세서를 분배한다. 최종적으로 마지막 작업까지 할당하고 난 후의 각 프로세서의 최종 할당 작업 개수는 다음과 같다.

$$\{ x_{if}' \} = \{ 11, 0, 3, 2 \}$$

워크스테이션 클러스터 시스템을 구성하는 P1에서 P4의 프로세서들에 분배되는 병렬 작업은 이기종의 상대적 처리능력 비교를 위해서 미리 분배된 하나씩을 제외하고 P1에 11개, P2에는 분배가 없고 P3에 3개, P4에 2개의 작업을 분배함으로써 현재의 이기종 네트워크 병렬 컴퓨터에서 각 프로세서의 능력이 고려되어 효과적으로 병렬 프로그램이 수행된다.

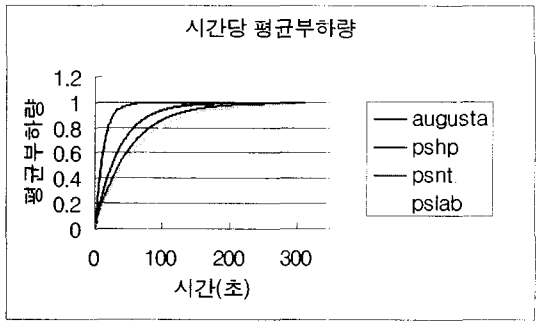
4. 실험 결과

실험을 위한 분산 환경은 Pentium Pro 200 프로세서 장착 psnt 워크스테이션, Pentium II 233 프로세서 장착 pslab 워크스테이션, SUN Sparc 10의 augusta 워크스테이션과 HP 712의 pshp 워크스테이션으로 구성하였으며 각 워크스테이션은 Solaris 2.5.1 운영체제를 가진다. 워크스테이션 클러스터링은 PVM을 기반으로 하여 psnt 워크스테이션을 주 워크스테이션으로 하여 병렬분산 환경을 구성하였다. 워크스테이션 클러스터 시스템을 구성하는 각 워크스테이션의 SPECint와 SPECfp 성능평가의 결과는 표 4와 같다.

(표 4) 처리기 SPEC 성능 평가

CPU Name	호스트 이름	SPECint	SPECfp
Pentium Pro 200	psnt	8.20	6.21
Pentium II 233	pslab	9.38	7.40
SPARCstation 10	augusta	1.13	1.38
HP 712/80	pshp	3.12	3.55

(표 5) 동일 작업 수행시 시간당 평균 부하값 변화



이기종의 각 처리기에 완전 유휴 상태인 0의 평균 부하값 상태에서 동일한 작업 부하를 주었을 때, 시간당 평균 부하량의 변화가 기종간에 차이가 있다면 이 논문에서 제안하고 있는 상대적 처리 능력 측정 알고리즘이 가능함을 증명 할 수 있다. 실험의 결과는 표 5와 같으며 변화율은 표 5의 기울기로 나타나고 그 정도는 시스템의 현재 처리 능력을 나타낸다.

SPEC 성능 평가의 결과를 A라하고 표 5의 기울기를 B라 했을 때 이들의 관계는

$$A = 2.96 \times \varepsilon^{-13.36 \times B}$$

의 비선형적 관계를 가지며 0.88의 상관계수를 가진다.

300×300의 행렬 곱셈 프로그램을 20개의 작업으로 분산 병렬 처리하는 실험을 통하여 제안 알고리즘과 기존 방식을 비교 분석하였다. 첫번째 실험은 psnt 워크스테이션에 병렬 작업 분배 시점에 자체적인 외부작업로 인한 부하가 있는 상황이고 나머지는 유휴(idle)상태이고 실험 2와 3은 자체 외부 작업 유입으로 인한 부하 발생 상황을 워크스테이션을 바꾸어 실험한 내용이다(진하게 표시된 부분은 과부하를 의미함). 실험 4은 모든 워크스테이션들이 유휴상태에서 실험한 결과이다. 또한 각 실험 환경에서 최종 작업의 완료를 의미하는 전체 반환시간을 나타낸다.

먼저 워크스테이션 클러스터 시스템에서 이기종의 성능차이를 무시한 단순 분배 방식인 라운드 로빈의 기존 방식으로 작업을 분배하여 처리된 전체 반환 시간은 표 6와 같다.

작업을 분배할 때, 과부하 워크스테이션을 변화 시키면서 실험한 1, 2, 3 내용 중에서 처리 능력이 떨어지는 augusta의 경우는 현재에 부하가 있는 상태에 부하 균형을 고려하지 않은 동일한 수의 작업 분배로 인하여 전체 작업 반환 시간이 확연히 늦어졌음을 알 수 있다. 또한 실험 4에서는 구성 워크스테이션들 모두가 유휴 상태이라도 가장 떨어지는 성능의 워크스테이션이 처리한 작업 반환 시간이 전체 작업 반환 시간에 영향을 미친다.

부하를 고려한 균형적 작업 분배를 위하여 기존에는 워크스테이션의 성능을 가중치로 미리 설정하여 작업을 분배하고 있다. 모든 구성 워크스테이션들이 유휴 상태일 때에는 좋은 결과를 보이지만 각 컴퓨터의 외부 작업 유입에 따르는 현재의 부하 정도를 반영하지 못한다. 그러므로 기존에는 작업 분배전 각 워크스테이션의 상대적 성능 정도를 파악하기 위해서 별도의 성능 테스트 프로그램을 수행한다. 이기종 처리기의 현재 부하 정도를 파악하기 위해서 기존 방식인 성능 테스트 프로그램에 의한 부하 균형 실험 결과는 표 7과 같다.

이 논문에서 제안한 알고리즘을 적용한 부하 균형 시스템에서 과부하 워크스테이션을 변화시키면서 실험한 전체 반환 시간은 표 8과 같다.

기존의 단순한 라운드 로빈 방식(RR식)의 전체

(표 6) 기존 방식 적용시 전체 반환시간(RR식)

실험 \ 호스트		psnt	augusta	pslab	전체반환시간
1	분배시 평균 부하량	0.65	0.21	0.0	293.85
	분배 타스크수	7	7	6	
2	분배시 평균 부하량	0.02	0.52	0.0	338.83
	분배 타스크수	7	7	6	
3	분배시 평균 부하량	0.0	0.02	0.52	293.00
	분배 타스크수	7	7	6	
4	분재시 평균 부하량	0.02	0.04	0.0	290.5
	분배 타스크수	7	7	6	

(표 7) 성능 테스트 프로그램 수행 후 부하 균형(BLB식)

실험 \ 호스트		psnt	augusta	pslab	전체반환시간
1회	분배시 평균 부하량	0.63	0.02	0.0	268.55
	분배 타스크수	1	6	13	
2회	분배시 평균 부하량	0.04	0.52	0.0	134.25
	분배 타스크수	4	2	14	
3회	분배시 평균 부하량	0.0	0.02	0.60	287.80
	분배 타스크수	6	7	7	
4회	분재시 평균 부하량	0.0	0.02	0.03	130.20
	분배 타스크수	3	3	14	

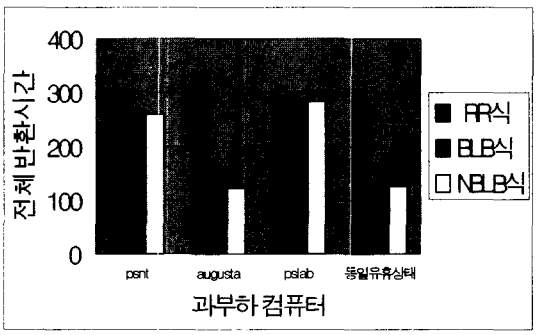
(표 8) 제안 알고리즘 적용 부하 균형(NBLB식)

실험 \ 호스트		psnt	augusta	pslab	전체반환시간
1회	분배시 평균 부하량	0.50	0.0	0.02	259.80
	분배 타스크수	1	6	13	
2회	분배시 평균 부하량	0.02	0.00	0.0	122.10
	분배 타스크수	4	2	14	
3회	분배시 평균 부하량	0.02	0.02	0.60	283.14
	분배 타스크수	6	7	7	
4회	분재시 평균 부하량	0.02	0.02	0.03	125.79
	분배 타스크수	3	3	14	

작업 반환 시간 결과인 표 6과 이기종 분산 환경에서 부하 균형을 지원하기 성능 테스트 프로그램 수행 결과를 이용한 부하 균형 방식(BLB식)의 전체 반환 시간 결과인 표 7과 논문에서 제안한 알고리즘을 적용한 부하 균형 시스템(NBLB식)에서 전체 반환 시간의 결과인 표 8의 비교는 표 9와 같다.

제안 알고리즘에 의한 부하 균형 시스템은 주어진 작업에 대한 전체 반환 시간이 기존의 가상 병렬 컴퓨터 구성시의 구성 순서에 의한 라운드 로빈 방식의 분배보다는 augusta 컴퓨터의 경우는 최고 277.5% 향상됨을 보인다. 또한 기존의 이기종 지원 부하 균형 시스템에서 적용하고 있는 성능 테스트 프로그램 수행 후 균형적 분배 방식과 비교하여도 향상됨을 볼 수 있다.

(표 9) 기존 방식과 제안 방식



5. 결 론

서로 다른 사용자들과 그룹들, 다른 외부 작업 수행 요구들, 다른 처리능력을 갖는 프로세서로 구성된 이기종 워크스테이션 클러스터에서 각 워크스테이션은 병렬 수행을 위한 작업들을 부여받아서 수행하게 된다. 이때 중요한 것은 그 워크스테이션의 부하 정도이며 이것을 관리할 관리 시스템이 필요하다. 이에 본 논문에서는 이기종 워크스테이션 클러스터 시스템을 지원하는 부하 균형 시스템을 중앙식의 주종관계에 의한 부하 관리자와 부하 모니터를 설계 구현하였다. 또한, 상대적 처리 능력을 파악하기 위해서 기존의 설치시 고정된 가중치를 부여하거나 성능 테스트 프로그램 수행에 따른 결과값을 이용하는 방법이 아닌, 응용 프로그램 자체의 동일 실행 작업들에 위한 상대적 처리 능력을 비교하여 부하 균형을 위한 균형적 작업 분배 방법을 제안하여 설계 및 구현한 부하 균형 시스템에 적용하였다. 즉 기존의 방법이 성능 테스트 프로그램 의존적 부하 균형 방법이라면 논문에서 제안한 방법은 자체 응용 프로그램에 의한 성능 테스트 프로그램 독립적 부하 균형 방법이다.

실험 결과는 이기종 워크스테이션 클러스터 시스템 구성 순서에 의한 순차적 작업 분배 방식에 의한 전체 작업 반환 시간에 비하여 제안 알고리

즘을 적용한 부하 균형 시스템에서 전체 작업 반환 시간이 향상됨을 보였다. 특히 구성 워크스테이션 중에서 가장 성능이 떨어지는 워크스테이션에 외부 작업에 의한 부하가 발생한 경우에 부하 균형 시스템은 상당한 전체 작업 반환 시간 단축을 보였다. 또한 이기종의 상대적 처리 능력을 수행 전 미리 가중치로 가지는 정적 방식이 아니 동적 방식의 상대적 처리 능력 분석 방식은 현재 상태의 워크스테이션 부하 정도를 적절히 나타내고 있다. 기존에 사용하던 방식인 성능 테스트 프로그램 수행 후 작업 분배 방식보다 성능 테스트 프로그램 수행에 따르는 부가 시간과 추가 부하가 없는 제안 방식의 부하 균형 시스템의 전체 작업 반환 시간이 단축됨을 보였다.

향후 연구에는 동적 부하균형을 위하여 작업 이주의 시점을 결정할 수 있는 이주 결정 스케줄링과 외부작업 유입으로 인한 평균부하의 변화를 현재 수행중인 병렬분산 작업에 어떻게 하면 효율적으로 적용하는가의 문제이다.

참 고 문 헌

[1] R. Butler, and E. Lusk, "Monitors, messages, and cluster: The p4 parallel programming system", Technical Report MCS-P362-0493, Argonne National Laboratory, Argonne, IL, 1993.

[2] J. Flower, A. Kolawa, and S. Bharadwaj, "The Express way to distributed processing", Supercomputing Review, pp. 54-55, 1991.

[3] N. Carriero, and D. Gelernter, "LINDA in context", Communication of the ACM, 32(4), pp. 444-458, 1989.

[4] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam, 'PVM: Parallel Virtual Machine A User' Guide and Tutorial for Networked Parallel Computing', The MIT Press.1994

[5] J. Casas, D. Clark, P Galbiati, R. Konuru, S. Otto, "MIST:PVM with Transparent Migration and Checkpoint", Dept of computer Science and Engineering Oregon Graduate Institute, 1995

[6] Thomas L. Casavant, "a taxonomy of scheduling in General-purpose Distributed computing systems", IEEE Trans. on Soft Engineering, Vol 14, No 2, 1988.

[7] J. C. Fabero, I. Martin, A. Bautista, S. Molina, "Dynamic Load Balancing in a Heterogeneous Environment under PVM", IEEE Procedings of PDP'96, 1996.

[8] Khaled Al-Saqabi, Steve W. Otto, Jonathan Walpole, "Gang Scheduling in Heterogeneous Distributed System", Oregon Graduate Institute, 1994

[9] T.s Hsu, J.C. Lee, D.R Lopez, "Task Allocation on a Network of Processors", IEEE Trans. on Computers, Vol 49, No 12, Dec 2000

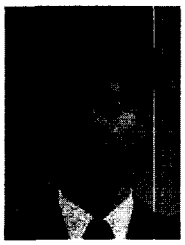
[10] David J Jackson, Chris W Humphres, "A Simple yet effective load balancing extension to the PVM software system", Parallel Computing 22, 1647-1660, 1997.

[11] Domenico Ferrari, "A Study of Load Indices for Load Balancing Schemes", Report No. UCB/CSD 86/262, October 1985

[12] 이 광모 외 2. "병렬 가상 컴퓨터에서 작업 부하를 고려한 타스크 관리자의 설계 및 구현", 병렬처리연구회 학술발표논문집, 제7권 2호, 1996.5

[13] Olivier DALLE, "LoadBuilder:a tool for generating and modeling workloads in distributed workstation environment" proceeding of 9th ISCA 1996.

◐ 저 자 소 개 ◑



지 병 준

1983. 2.~1987. 3. 한림대학교 전산계산학과 (이학사)
1987. 9.~1993. 2. 중앙대학교 컴퓨터공학과 (공학석사)
1995. 9.~1998. 9. 한림대학교 컴퓨터공학과 (박사수료)
1994. 3.~현재 한림정보산업대학 전산정보처리과 부교수
관심분야 : 분산처리, 클러스터 시스템, 웹 분산
E-mail : bjji@sun.hallym-c.ac.kr

이 광 모

1980. 9.~1985. 1. 조선대학교 전자계산학과 조교수
1985. 2.~현재 한림대학교 정보전자공과대학 컴퓨터공학부 교수
1992. 1.~1993. 1. Florida State University 방문교수
관심분야 : 병렬처리. 프로그램 언어, 병렬 컴파일러