성능이 다른 워크스테이션을 위한 동적 Load Balancing

O
전상준, 심재홍, 김재훈, 최경희
아주대학교 정보 및 컴퓨터 공학부

Load Balancing for Workstations with Different Speed

O
Sang-Joon Jun, Jae-Hong Shim, Jai-Hoon Kim, Kyung-Hee Choi
Information & Computer Science Department, Ajou University

요 약

본 논문에서는 성능이 서로 다른 워크스테이션들을 효과적으로 사용하기 위한동적인 Load Balancing 방식을 제안한다. 각 워크스테이션의 수행시간에 따라서 load 의양이 결정되며 각 워크스테이션의 성능을 사전에 알아야 할 필요가 없는 방식이다. 이를 Ethernet 으로 연결된 두개의 워크스테이션에서 실험하여 그 성능을 비교 평가한다.

1. 서론

네트워크 기술의 발전에 따라 하나의 독립적인 기계를 사용하기보다는 네트워크에 의해 연결된 여러 개의 워크스테이션들을 이용해서 프로그램을 수행하는 것이 일반화 되어가고 있다. 이를 워크스테이션의 성능이 동일한 경우도 있지만 서로 다른 워크스테이션으로 컴퓨팅 환경을 구축하는 경우도 있다. 워크스테이션들의 성능과 기종이 유사할 경우는 load 의 양을 동일하게 분배하면 되지만 각 워크스데이션들의 성능이 다를 경우에는 load 를 분배함에 있어서 신중을 기해야 한다.

즉, 네트워크로 연결된 여러 개의 워크스테이션들을 이용해서 프로그램을 효과적으로 수행하기 위해서는 워크스테이션의 성능에 따라 다른 양의 작업을 배분하는 과정이 필요하다. 또한 각 워크스테이션들은 서로 다른 성능을 가질 뿐만 아니라 다른 사용자 또는 다른 작업등으로 인하여 그들의 성능 및 load 가 계속적으로 바뀔 수 있기 때문에 각 워크스테이션들의 load 를 측정하고 이에 따라 배분되는 일의 양을 달리하는 연구가 요구된다.

본 논문에서는 워크스테이션들에서 하나의 프로그램을 수행할 때, 워크스테이션들의 성능이 다를 경우와 다른 사용자 또는 다른 작업으로 인하여 그 워크스테이션들의 load 가 변하는 상황에서도 효과적으로 적용되는 load balancing 방법을 제안한다. Ethernet 으로 연결된 두 개의 워크스테이션에서 socket 을 이용하여 제안된 load balancing 방법이 실험되고 이를 이용해서 load balancing 이 적용되지 않은 경우와 비교함으로 이 dynamic load balancing 의 성능향상을 보인다.

2. 관련연구 조사

Load Balancing 은 크게 동적(dynamic)인 방식과 정적(static)인 방식으로 나눌 수 있다. 이것은 각각에 분산되어질 load 의 양이 결정되는 시기에 따른 것으로 동적인 Load Balancing 은 프로그램의 수행중에 Load 의 양이 결정된다.

M. J. Zaki 는 동적 load balancing 을 load balancing 결정에 사용되는 정보에 따라 global 과 local 한 방식으로 나누고 load balancer 가 하나의 프로세서인지 아닌지에 따라 centralized 와 distributed 한 방식으로 나누어서 어떠한 경우에 각 방식이 적절한지를 비교했다[1].

Hamdi 는 이미지(image) 프로세싱을 위한 동적 load balancing 을 제안했는데 이는 하나의 iteration 의 수행이 오래 걸리게 때문에 하나의 iteration 안에서도 각 프로세스의 load 에 따라 일의 양을 조절할 수 있는 방식을 택한다. 여기서 load 를 분배하는 manager 는 각 worker process 의 load 정보를 항상 알고 있어야 한다[4].

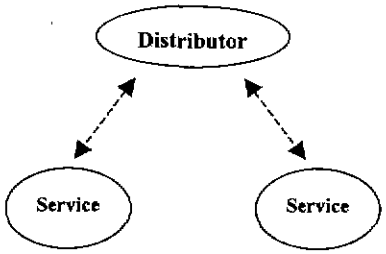
T. Schnekenburger 는 각 프로세스의 속도에 따라 결정되는 프로세스 가중치(weight)에 따라 workload 의 양이 결정된다[5].

3. 동적 load balancing 제안

3.1 프로그램 모델

성능이 서로 다른 워크스테이션들을 효과적으로 사용하기 위해서는 그것들의 성능 및 load에 따른 일의 분배가 이루어져야 한다. 여기에서는 각 워크스테이션의 성능을 판단하는 기준은 초기값으로 주어진 일의 수행시간에 두고 있다.

프로그램으로는 여러 워크스테이션에서 분산되어 실행되기 위한 병렬 루프(Parallel loops)를 가진 것을 이용한다. 구조는 그림과 같이 Distributor 프로세스와 Service 프로세스로 이루어진다. Distributor는 주어진 일을 각 Service 프로세스에 분배하는 역할을 하고 각 Service의 성능에 따라 다음 load를 분배하는 역할을 한다. 각 Service는 Distributor부터 받은 일을 수행하고 다른 메시지를 기다린다. Service 간에는 통신은 이루어지지 않는다.



[그림 1]

3.2 Workload 분배 방법

Workload를 각 워크스테이션의 성능 및 load에 기초하여 다르게 분배하여 최단 시간에 프로그램을 수행하는 것이 load-balancing의 근본 취지이다. 따라서 각 워크스테이션의 성능 및 load를 계산하는 데 걸리는 시간은 짧아야 하며 재분배를 계산함에 있어서도 동일하게 적용된다.

Distributor 프로세스는 다음과 같은 순서로 진행된다.

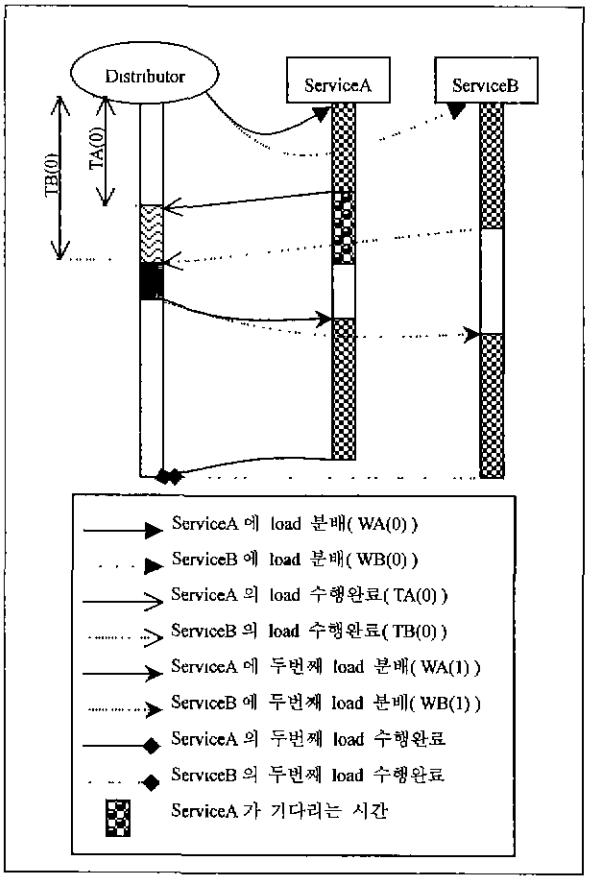
i) Distributor 프로세스는 초기값으로 각 Service 프로세스에게 일정량의 load를 분배한다.

ii) 각 Service 프로세스는 분배받은 load를 수행하고 그 수행시간을 Distributor에게 알린다.

iii) 수행시간에 반비례하게 그 다음 load의 양을 계산하고 계산된 load를 각 워크스테이션에 분배한다.

iv) 주어진 모든 일을 끝낼 때 까지 ii)와 iii)을 반복한다.

서로 다른 성능을 가진 워크스테이션들을 이용해서 프로그램을 수행할때 그 수행시간은 느린 것에 따라 결정되기 때문에 성능이 우수한 워크스테이션의 컴퓨팅 파워를 낭비하게 되고 전체 수행시간도 길어진다. 이를 해결하기 위하여 작업의 양을 작업에 비례하여 각 워크스테이션에 분배하여 수행하게 된다.



[그림 2]

이 load balancing은 사전에 각 워크스테이션에 대한 성능이나 시스템에 대한 어떠한 정보가 없더라도 적용될 수 있다. 워크스테이션들의 성능이 다를 경우에 성능이 느린 기계의 영향을 줄여서 전체 수행시간을 단축시키는 효과를 가진다. 워크스테이션들의 load가 다른 사용자 또는 다른 작업으로 인하여 변화하여도 각 워크스테이션에 주어지는 작업량이 조절되기 때문에 동적으로 load가 변화할 때도 사용될 수 있다.

[그림 2]와 같이 Distributor는 각 Service 들로부터 요청을 받으면 Service 프로세스들에게 같은 양의 workload인

WA(0)와 WB(0)를 분배한다. 각 Service 프로세스가 실행을 마치면 Distributor는 그들의 수행시간에 반비례하게 다음 load를 준다.

$$WA(1) * TA(0) = WB(1) * TB(0)$$

4 구현 및 실험

4.1 실험 환경 및 방법.

성능이 서로 다른 두개의 워크스테이션을 이용해서 실험이 이루어진다. SUN ULTRA SPARC 으로 Solaris 2.5.1 을 사용하는 것과 SUN20 는 10M bps 정도의 Ethernet 으로 연결된다.

다음과 같은 병렬 루프가 두 워크스테이션에서 실행된다. 처음에는 같은 수의 루프를 두 워크스테이션에 분배하고 그 수행시간을 측정하게 된다. 이것이 각 워크스테이션의 초기 성능으로 평가되고 이에 따라 두번째로 분배되는 루프의 양이 결정된다. 즉, 초기값으로 주어진 루프의 수행을 먼저 완료한 성능이 상대적으로 우수한 워크스테이션에는 더 많은 수의 루프가 주어지게 된다.

그들 중 한 워크스테이션의 load가 다른 사용자나 다른 작업으로 인해서 변하는 경우에는 그 단계에 주어진 루프를 수행하는 데 소요되는 시간이 길어진다. 이러한 경우에도 수행시간에 따라 각 워크스테이션의 load를 알 수 있으므로 다음 단계에 수행할 루프의 수를 줄이게 된다. 따라서 각 워크스테이션의 동적인 load 변화에도 성능이 저하되는 것을 방지 할 수 있다.

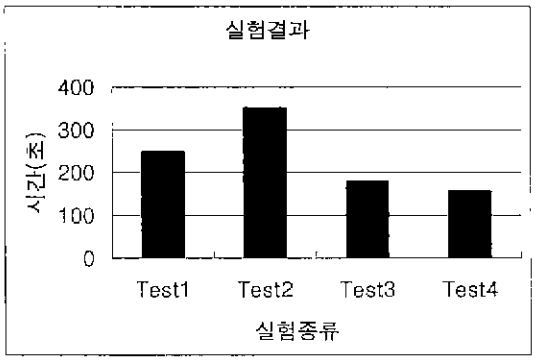
```
for(i=from; i<to; i++)
  for(j=0; j<n; j++)
    for(k=0; k<n; k++)
      for(l=0; l<n; l++)
        value = 1*i + 2*j +3*k + 4*l;
```

4.2 실험 결과

다음과 같이 시험은 4 번의 실험이 실시된다.

- Test 1 : 첫번째 워크스테이션에서만 실행한 경우
- Test 2 : 두번째 워크스테이션에서만 실행한 경우
- Test 3 : 같은 작업량을 두 워크스테이션에 실행한 경우
- Test 4 : 동적 load balancing을 사용한 경우

그 결과는 [그림 3]과 같이 Test 4 의 수행시간이 가장 짧음을 알 수 있다.



[그림 3]

5 결론

동적인 load balancing을 적용한 경우가 그렇지 않은 경우보다 수행시간이 단축됨을 볼 수 있다. 이 방법은 시스템에 대한 사전지식이 필요 없고 또 각 워크스테이션의 load가 다른 사용자 또는 다른 작업수행으로 변화되어도 이에 적응할 수 있는 장점이 있다. 향후 과제로서 이 방식을 이용한 다양한 실험과 적용을 남겨둔다.

참고문헌

[1] M. J. Zaki, W. Li, S. Parthasarathy, "Customized Dynamic Load Balancing for a Network of Workstations," *Univ. of Roachester,* TR602, December 1996.

[2] M. Cierniak, W. Li, and M. J. Zaki, "Loop Scheduling for Heterogeneity," *In 4th IEEE Intl Symposium on High-Performance Distributed Computing, also as URCS-TR 540, CS Dept., Univ. of Rochester*, August 1995.

[3] K. K. Yue and D. J. Lija. "An Effective Processor Allocation Strategy for Multiprogrammed Shared-Memory Multiprocessors," *IEEE Trans. on Parallel and Distributed System,* Vol. 8, No.12, December, 1997.

[4] M. Hamdi, C.K. Lee, "Dynamic load-balancing of image processing applications on clusters of workstations." *Parallel Computing 22(1997)*, pp 1477-1492.

[5] T. Schnekenburger and M. Huber, "Heterogeneous Partitioning in a Workstation Network," *Symposium, Workshop on Heterogeneous Computing,* Mexico. April 1994, pp. 72-77