인터넷상의 대규모 유휴 PC들을 사용한 효율적인 PC Cluster의 설계

김영균° 최종욱 김용호 오길호
금오공과대학교 컴퓨터공학부
{ygkim°, jwchoi, kimyh, gilho}@cespc1.kumoh.ac.kr

Design of an Efficient PC Cluster using a Very Large Idle PCs on Internet

Young-Gyun Kim° Jong-Wook Choi Yong-Ho Kim Gil-Ho Oh
School of Computer Engineering, Kumoh national Institute of Technology

요 약

본 논문에서는 인터넷 상에서 대규모의 유휴 PC들을 연산에 활용하는 PC Cluster시스템에 대해 연구하였다. 인터넷상의 노드는 i)이질적이고, ii)각 노드의 신뢰성이 확보되지 않는 특징을 갖는다. 또한 각각의 노드는 iii)지역 작업을 가질 수 있다. 작업 할당시 연산에 참여하는 노드의 이력 정보와 유휴상태를 함께 고려하여 작업을 할당함으로서 i),ii)를 고려한 PC클러스터링 방법을 제안하였다. 작업 할당시 노드의 이전 작업들의 처리에 대한 신뢰성과, 작업 반환 시간을 고려하여 작업을 노드에 할당함으로써 신뢰성이 떨어지는 노드와 평균 반환 시간이 상대적으로 다른 노드들에 비해 큰 노드에 대해서 작업 할당을 가능한 줄임으로써 전체 작업의 신뢰성을 높일 수 있고 평균 작업 반환 시간을 단축시킬 수 있다. 제안한 방법에서는 iii)의 지역 작업을 가질 수 있는 인터넷상의 서로 다른 성능의 이질적인 PC로 Cluster를 구축할 경우 보다 적합한 Self-Scheduling을 사용 함으로서 효율적으로 작업 할당이 가능하도록 하였다. 제안한 방법은 지역 작업을 갖고 있는 인터넷상의 대규모 PC들로 구성된 신뢰성 있는 PC클러스터 구축을 가능하도록 한다.

1.서론

최근 PC Cluster에 대해 활발한 연구가 진행되고, 저비용 고성능의 컴퓨팅 플랫폼에 대한 해결책으로 사용되고 있다[2,3].

본 논문에서는 인터넷상에서 대규모의 유휴한 PC들을 연산에 활용하는 효율적인 PC Cluster에 대해 연구하였다. 인터넷은 특성상 인터넷에 참여하는 노드(Node)의 성능과 기종이 서로 다른 이질적인 특징을 갖고 있으며, 각 노드의 신뢰성이 확보되지 않는다. 또한 각각의 노드는 지역 작업을 가지고 있다. 인터넷의 이러한 특징을 반영할 수 있도록 작업 할당시 연산에 참여하는 노드의 이력정보를 고려 함으로서 신뢰성을 확보할 수 있도록 하였다. 각 노드는 서로 다른 성능을 고려할 수 있도록 유휴(Idle)상태에 Master에 작업을 요청하는 Self-Scheduling[4]방식을 사용하여 연산에 참여 한다. 또한 Master는 유휴상태의 노드가 작업 요청시 평균 반환 시간을 고려하여 작업을 할당함으로써 성능에 따른 작업 할당이 가능하도록 하였다. Self-Scheduling을 효율적으로 진행하기 위해 각 노드는 Master노드에 자신의 IP를 등록한 후, 노드의 상태를 감시하여 유휴 상태일 때, 마스터 노드로 유휴상태를 통보하고 작업 할당을 요청한다. 인터넷상의 대규모의 이질적인 PC로 구성된 클러스터를 손쉽게 모니터링하고 작업할당 할 수 있도록 자바 기반의 이동 에이전트 시스템인 IBM의 Aglet을 사용하였다[1].

2.관련연구

2.1 PC Cluster

높은 비용의 전용 대형컴퓨터를 사용하는 것보다 낮은 비용의 PC들을 네트워크로 연결 함으로서 비슷하거나 훨씬 뛰어난 성능을 발휘할 수 있는 클러스터(Cluster)시스템들에 대한 연구가 활발히 진행되어 왔다[2,3]. 클러스터 시스템은 전용의 워크스테이션으로 구성된 경우와 비전용의 PC 클러스터로 구성된 경우, 이 두 가지 형태가 혼합된 형태가 있다. 네트워크 상의 분산된 컴퓨팅 자원을 효율적으로 사용하기 위해 최근 많이 연구되고 있다.

2.2 이동 에이전트(Mobile agent)

기존의 분산처리에서는 작업을 분산처리하기 위해 주로 데이터의 이동(Data Migration)으로 네트워크상의 각 노드로 작업을 분산 처리하였다. 이동 에이전트 시스템에서는 분산처리의 단위가 코드와 데이터의 이동(Code and Data Migration)으로 이루어 진다. 이동 에이전트는 네트워크상의 각 노드를 자율적으로 이동해가면서 특정 작업을 수행할 수 있는 프로그램 코드이다. 대표적인 이동 에이전트 시스템으로서는 IBM의 Aglet시스템이 있다[1].

2.3 Self-Scheduling Schemes

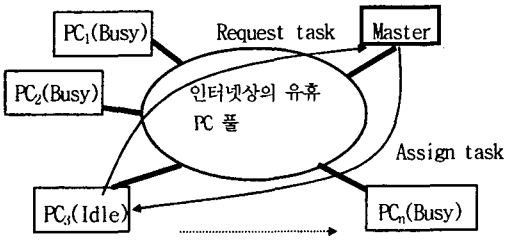
Master-Slave Model에서의 작업 할당시 기존 분산처리 방법은 Master 각 Slave Model의 작업 부하를 고려하여 강제적으로 작업을 할당하는

방식이었다. 이러한 방법은 Master-Slave Model에 기초한 소규모의 프로세스로 구성된 전용의 클러스터 시스템을 구성할 경우 효율적일 수 있으나, 인터넷과 같이 지역 작업을 가질 수 있는 서로 다른 성능을 갖는 노드로 구성된 경우 Master 주도형의 작업 스케쥴링 방식은 비효율적이다. 따라서 서로 다른 성능을 가질 경우 Slave가 유휴 상태일 때 Master에게 작업을 요청하는 방법인 Self-Scheduling[4]방법을 사용하는 것이 보다 효율적이다.

3.시스템의 설계

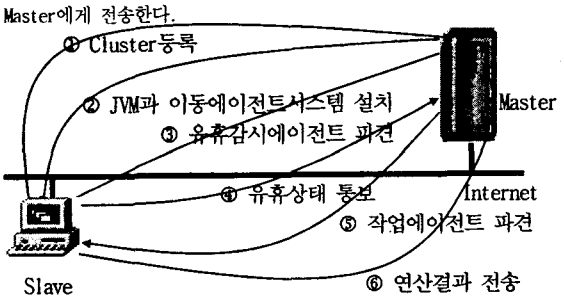
3.1 시스템의 구성

제안한 방법을 사용하는 시스템은 TCP/IP 프로토콜을 사용하는 인터넷환경으로서 그림2와 같이 구성되어 진다.



<그림1. 제안한 방법을 사용하는 시스템의 구성>

연산에 참여하고자 하는 인터넷상의 노드는 1) Master에게 접속하여 Cluster연산에 참여하기 위해 등록을 한다. 2) 자바 가상머쉰과 이동 에이전트 시스템을 다운로드 받아 설치한다. 3) Master노드는 연산에 참여하고자 신청한 노드에게 노드의 유휴상태를 감시하는 이동 에이전트 코드를 파견하고 해당 노드에서 수행하도록 한다. 4) 노드에서 수행중인 유휴상태 감시 에이전트는 노드의 유휴상태 발생시 Master노드에게 유휴상태를 통보하고 작업 할당을 요청한다. 5) Master는 연산을 수행할 이동 에이전트 코드를 작업을 요청한 유휴노드로 파견한다. 6) 유휴노드에 도착한 이동 에이전트 코드는 연산을 수행하고 결과를 Master에게 전송한다.



<그림2. 제안한 시스템의 수행 과정>

연산에 참여하는 노드의 유휴상태는 현재 노드가 일정 시간 수행되는 프로세스(Process)가 없을 때와 노드의 지역 사용자가 일정시간 노드를 사용하지 않는다고 유휴상태로 설정한 경우에 발생한다. 두 가지 경우에 노드의 유휴상태를 감시하는 에이전트가 Master에게 현재 노드가 유휴상태라는 것을 통보한다.

3.2 작업할당 방법

Master는 유휴 Slave노드로부터 작업 할당 요청 메시지를 수신한 경우 Master_Alloc알고리즘과 같이 유휴노드의 평균 작업 반환시간, 신뢰도, 유휴상태를 고려하여 작업을 할당한다.

1) Master의 작업 할당 알고리즘

노드로부터 유휴 상태를 나타내는 메시지가 Master에게 도착했을 때 수행 된다.

```
Algorithm Master_Alloc
Input :
  H(ip) : 주소가 ip인 노드의 History정보를 담고 있는 테이블
Output :
   작업 할당 여부를 나타내는 True, False값을 리턴.
Begin
   /* 노드의 평균반환시간이 경계값 이상인지 확인 */
   If H(ip).AVR_TrunAroundTime ≥ δTime Then
         return FALSE;
   /* 노드의 신뢰도가 경계값 이상인지 확인 */
   If H(ip).Reliability ≤ δReliability Then
         return FALSE;
   /* 노드가 유휴상태이면 작업 할당 가능 */
   If NOT ( H(ip).Idle ) Then return TRUE
      Else return FALSE;
End.
```

2) Slave의 작업 할당 요청 알고리즘

각 노드는 유휴 상태일 때 Master에게 Slave_Request알고리즘을 수행하여 작업을 요청한다.

```
Algorithm Slave_Request
Input :
   Master_IP : Master의 위치를 나타내는 IP address
   Current_Node.Idle : 현재 노드의 유휴 상태
Output :
   작업 할당 여부를 나타내는 True, False값을 리턴.
Begin
  While ( Current_Node.Idle )
   Begin
     /* Master에게 작업 할당 요청 */
    Send_Task_request(Master_IP);
    /* 작업할당이 성공적이면 현재 노드를 Busy상태로 변경 */
    If ( Wait_Job_Assignment(Master_IP) )
          Then Current_Node.Idle=False;
  End While
End.
```

3) Slave의 유휴상태 감시 알고리즘

Slave 노드에서 일정한 주기로 현재 노드가 유휴상태인지 확인하기 위해 알고리즘 Slave_Idle로 주어진 절차를 각 Slave 노드에서 수행 한다.

```
Algorithm Slave_Idle
 Input :
    Current_Node.Process : 현재 노드의 프로세스 수
    User_Idle : 사용자의 유휴상태 설정
 Output : Current_Node.Idle : 현재 노드의 유휴상태
 Begin
    /* 사용자가 유휴상태로 설정한 경우 확인 */
  If ( User_Node ) Then  Current_Node.Idle = True;
   /* 현재 노드의 프로세스 수가 1이하인지 확인*/
  If ( Current_Node.Process > 1
```

```
                    AND NOT Current_Node.Idle )
  Then
      Current_Node.Idle = False;
  Else
     /* 일정시간이 경과되었는지 확인 */
     While ( 0 ≤ δTime AND NOT Current_Node.Idle )
     Begin
        If ( Current_Node.Process ≤ 1 )
                    Then Current_Node.Idle = True;
        δTime = δTime - 1;    /* 시간 값 감소 */
     End While
     δTime = αTime      /* Time상수 재초기화  */
  End If
End.
```

3.3 History 테이블의 구조

연산에 참여하는 인터넷상의 각 노드에 대한 이력(History)정보가 테이블 1과 같은 구조로서 Master노드에서 유지된다. 각 노드의 IP address와 각 노드에 할당된 작업들의 평균 반환시간과 작업을 마치고 결과를 Master로 반환한 작업의 수를 노드에 할당된 작업수로 나눈 신뢰도 지수가 저장되고, 현재 유휴상태에 있는지 여부를 나타내는 값이 테이블에 저장된다.

< 표1. Master노드의 History테이블의 구조 >

No.	IP address	평균반환시간	신뢰도	유휴상태
1	202.31.130.91	30Min	0.999	Idle
...	...	...	...	...
n				

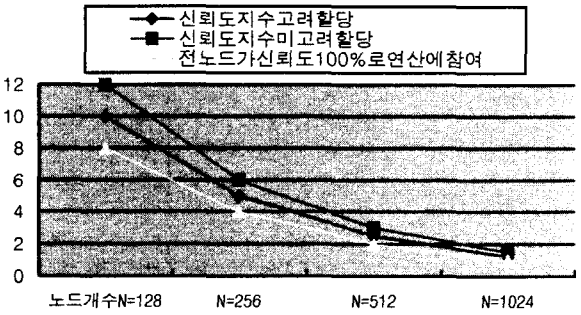
노드의 평균반환시간과 신뢰도는 (식1)과 (식2)와 같이 구할 수 있다.

평균반환시간 = (Σ반환된작업들의반환시간)/할당된작업의수 (식1)
신뢰도 = 노드의반환된작업의수/노드에할당된작업의수 (식2)

각 노드의 사용자가 연산에 참여하는 노드를 등록한 직후 해당 노드의 평균 반환시간과 신뢰도를 측정하기 위해 행렬연산을 수행하는 노드 테스트 작업을 할당한다. 노드 테스트 작업을 수행한 후 노드의 초기 평균반환시간과 신뢰도 지수를 계산한다. 노드의 평균반환시간과 신뢰도를 사용하는 방법은 이질적인 성능의 노드에 대해 상세한 하드웨어적인 지식이 없이 각 노드의 처리 능력과 통신상의 신뢰성을 경험적으로 결정 할 수 있다는 장점을 갖는다.

4. 제안한 방법의 평가 및 고려 사항

그림 3은 연산에 참여하는 노드의 연산결과 미반환율 α=0.2일때 노드에 따른 할당 가능한 동일한 크기의 1024개 Job에 대해 연산시간을 보여 준다. N=128인 경우 α=0.2를 고려하면, 실제 연산에 참여하는 노드는 N_{Real} = 102개가 된다. 신뢰도 지수를 고려했을 때, 미반환 노드에 대해 작업을 재배치하여 연산결과를 받는 만큼의 시간이 줄어 좀 더 빠르다는 것을 알 수 있다. 제안한 방법은 연산에 참여하는 연산노드(PC)의 수가 증가할수록 Master노드의 부하가 증가되어 병목현상을 유발할 수 있다. 따라서 적절한 규모의 연산노드에 대해 Sub-Master를 두는 계층적인 분산 스케줄러를 사용하는 것이 보다 바람직하다. Sub-Master를 둘 경우, Sub-Master로 지정된 노드의 경우 클러스터 전용으로 확보해야 하는 단점을 갖는다. 또한 각 노드의 연산노드에서 결함이 발생한 경우에 대해 중간 계산 결과를 적절히 사용할 수 있도록 체크포인트(check-point) 기법을 적용할 수 있다[5]. 이 경우에 Master 노드에 체크 포인트 기능을 구현할 경우 연산 노드의 개수가 증가함에 따라 Master노드에 체크포인트 및 복구에 소요되는 부하가 가중 될 수 있고 체크포인트로 인한 통신량이 집중되는 문제점을 효과적으로 다룰 수 있도록 계층적인 분산 체크포인트(Check-point)기법이 도입되어야 한다.



<그림3. α=0.2일때의 노드 수에 따른 연산시간(Hours)>

5.결론 및 향후 연구방향

본 논문에서는 이질적인 성능을 가지는 컴퓨터들로 구성된 인터넷상의 유휴 PC를 사용한 대규모 PC클러스터링 시스템에 대해서 연구하였다. 제안한 방법으로 PC클러스터링 시스템을 구성할 경우 신뢰성을 확보하고, 평균 작업 반환 시간을 줄일 수 있는 인터넷상에서 대규모의 PC 클러스터를 쉽게 구성할 수 있다. 지역 작업을 갖는 대규모의 PC를 클러스터링할 경우 Master노드에게 통신 메시지와 작업 분산 스케줄링 부하가 집중되는 현상을 효율적으로 방지할 수 있는 방법과 각 노드에 분산 되는 작업의 크기가 큰 경우 연산 노드의 결함(Fault) 발생시 효율적으로 연산과 체크 포인트(Check-Point)를 중복 할 수 있는 방법에 대해 연구 해 보겠다.

참고문헌

[1] Danny B. Lange , Mitsuru Oshima, Programming And Deploying Java Mobile Agents With Aglets

[2] Putchong Uthayopas, Surachai Phaisithbenchapol, Krisana Chongbarirux, "Building a Resources Monitoring System for SMILE Beowulf Cluster", Proceeding of High Performance Computing, Asia '99, 1999

[3] Rajkumar Buyya, "PARMON: a portable and scalable monitoring system for clusters", SOFTWARE-PRACTICE AND EXPERIENCE, Softw. Pract. Exper. 2000; 30:1-17

[4] Anthony T. Chronopoulos, Razvan Andonie, "A Class of Loop Self-Scheduling for Heterogeneous Clusters", Proceedings of the 2001 IEEE International Conference on Cluster Computing(CLUSTER'01)

[5] Kam Hong Shum, "Fault Tolerant Cluster Computing through Replication", Proceedings of the 1997 International Conference on Parallel and Distributed Systems(ICPADS '97)

[6] Partha Dasgupta, Zvi M. Kedem, Michael O. Rabin, "Parallel Processing on Networks of Workstations: A Fault-Tolerant, High Performance Approach", 15th Intl. Conference on Distributed Computing Systems, May 1995, Vancouver, BC, Canada