클러스터 시스템에서의 동적 여분 부하 균등화

정훈진 최상방
인하대학교 전자공학과
woosuwar@hitel.net, sangbang@dragon.inha.ac.kr

Dynamic Marginal Load Balancing on Cluster System

Hoon-Jin Jung Sang Bang Choi
Dept. of Electronic Engineering, Inha University

요 약

네트워크의 속도가 빨라짐에 따라 저비용으로 고성능의 성능을 얻고자 하는 클러스터 시스템에 대한 연구가 많아지고 있다. Fast Ethernet, ATM, Myrinet, SCI와 같은 고성능 네트워크 시스템이 클러스터 시스템에 많이 사용되고 있다. 기존 분산시스템에의 부하 불균등 문제가 이런 클러스터 시스템에서도 역시 문제시 되고 있다. 일반적인 동적 부하균등화 방법은 실행시간중에 노드들의 부하량에 대한 정보를 얻어 전체 노드들이 평균의 부하량에 수렴하도록 하는 것을 목적으로 한다. 그러나, 통신비용과 노드 복잡도에 따라 완벽한 부하균등화를 하는 것이 효율적인 부하균등화를 얻는 것이 아니다. 부하균등화 비용이 클때는 완벽한 부하균등화보다는 이득률이 크도록 하는 부분적 부하 균등화가 더 효율적일 수 있다. 본 논문에서는 클러스터 시스템을 모델링하고 이에 적합한 부분적 동적 부하 균등화 모델을 제시한다.

1. 서론

최근 급속한 하드웨어 성능의 발전과 네트워크 기술(ex. Gigabit Ethernet, ATM, Myrinet, SCI)의 발전으로 비싸고 특화된 병렬 슈퍼컴퓨터에서 가격 대 성능비가 높은 클러스터 시스템으로 이동되는 경향이 있어왔다. 대표적인 클러스터 시스템에는 NOW(Network of Workstaions), Beowulf, HPVM(High Performance Virtual Machine), Solaris-MC와 같은 것들이 있다. 이런 시스템의 주된 프로그램밍 방식은 SPMD로서 각 노드에 같은 프로그램을 수행하지만 데이터의 형태는 다른 방식이다. 대표적으로 MPI, PVM과 같은 메시지패싱 방식 프로그램밍 환경이 이를 지원하고 있다.

현재 클러스터 시스템은 분산시스템과 MPP시스템의 중간정도의 위치에 해당하는 시스템으로서 노드의 개수는 보통 100여개 내외[1]이며 주된 네트워크 위상은 Switch 구조이다. 저비용으로는 Ethernet, Fast Ethernet을, 고비용으로는 ATM, Gigabit Ethernet, Myrinet, SCI와 같은 네트워크 시스템을 사용하고 있다.

병렬 컴퓨팅의 주된 성능 장애로서 부하 불균등현상을 들 수 있는데 클러스터 시스템에서도 예외는 아니다. 클러스터 시스템은 loosely-coupled된 분산시스템으로 볼 수 있는데 그 만큼 통신비용이 상대적으로 많이 들므로 통신비용이 적은 효과적인 부하균등화를 하여야 한다. 따라서 클러스터 시스템에서 최대한의 성능을 얻기 위해서는 이런 부하 불균등현상이 일어나지 않도록 각 노드가 동등한 부하량의 태스크를 할당받아야 한다. 그러나 각 태스크의 부하량은 실행 이전에 예측할 수 없으며, 한 태스크가 하위 가지로 새로운 Thread을 생성하고, 또한 태스크간에 의존관계가 있어서 동등한 태스크 할당을 불가능하게 한다. 그러므로 실행시간 중에 수시로 각 프로세서의 부하량을 측정해서 적절한 부하재분배를 이루어야 한다. 이를 동적 부하균등화라고 한다.

통신비용을 적게 드는 경우 동적 부하 균등화는 완벽하게 하는 것이 효율이 높다. 그러나 클러스터 시스템에서와 같이 통신비용이 높고 Switch구조로 가변적인 위상을 가지고 있는 경우에는 완벽한 부하 균등화는 비용이 많이 들며 잦은 부하균등화는 오히려 성능을 감소시킬 수 있다.

본 논문에서는 클러스터 시스템에 적합한 부분적 동적 부하 균등화를 모델링하고 이를 시뮬레이션하였다.

2. 클러스터 시스템내의 부하균등화 알고리즘

클러스터 시스템에서는 SPMD(Single Program Multiple Data) 형태로 프로그램을 작성한다. 즉, 각 노드에 같은 프로그램을 수행하나 그 각각의 데이터는 다른 형태를 띄게 된다. 많은 병렬 응용 프로그램들이 이런 부류에 속한다. 예로 linear algebra problems, partial differential equation solvers, image processing algorithms등이 있다. 이런 프로그램들은 처음에 데이터를 각 노드수에 맞게 쪼게 나눈다. 그러나 데이터가 가진 불균등성에 의해 처리시간의 차이가 오고 미리 예측할 수 없는 부하불균등이 이루어진다. 그러므로 수행시간 중간에 Task을 이동시켜 부하 균등화를 이루어야 한다.

본 논문의 목적에 부합되는 클러스터 시스템에서는 기존 분산시스템에서 Transfer Policy의 Threshold기준으로 주로 사용되는 CPU queue length, CPU utilization, I/O usage와 같은 절대적인 Threshold를 사용할 수 없다. 그러므로 클러스터 시스템에서는 각 노드의 부하를 판단하는 기준으로 각 노드의 Run Queue의 Task의 개수로 판단[2]을 한다. 즉, 현재 수행중인 Run Queue Task 개수를 사용하여 다른 노드와 비교하여 어느 쪽이

더 많은 Task를 가지고 있는지를 통해 부하 균등화를 수행한다. 각 노드의 부하정보를 모아 평균의 Task개수를 구하고 이를 Threshold의 기준으로 삼는다. 본 논문에서는 평균을 중심으로 α%내에 있는 부하를 중간부하라고 하고 평균으로부터 $1/2\alpha$%보다 큰 부하를 가지는 노드는 과부하노드로, 평균보다 $1/2\alpha$%보다 적은 부하를 가지는 노드는 저부하노드로 삼는다. 부하균등화는 모든 노드들의 부하 정보를 모으고 이주정책을 거친 후 과부하 노드에서 저부하 노드로 Task를 이주시키게 된다. 그러나, 통신비용의 크기 변화에 따라 부하균등화에 참여하는 노드개수를 변화시켜서 가장 효율적인 성능향상을 가져오는 부하균등화를 수행해야 한다. 통신비용이 커질수록 정보교환 비용과 이주 비용이 커지므로 중간 노드의 개수를 증가시키도록 하여 이주에 참여하는 노드의 개수를 줄이는 것이 적합하며 이와 반대로 이주 비용이 작아지면 대부분의 노드가 부하균등화에 참여하게 하여 최적의 부하균등화를 이루는 것이 바람직하다. 중간노드 개수가 많아지면 이주에 따른 부하 균등화 이득율이 감소하므로 이는 trade-off관계가 있다.

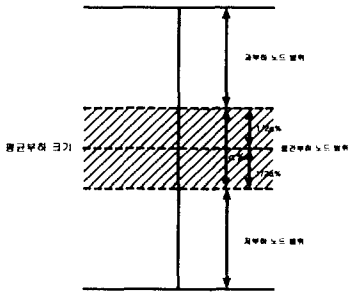

그림 1 노드 부하 결정

3. 부하 균등 과정

시뮬레이션 모델은 Master-Slave 계산모델을 사용하였다. Master가 Slave가 수행할 Task를 생성하고 부하균등화를 담당한다. 각각의 Slave는 받은 Task를 수행하여 그 결과를 Master에게 되돌려주고 모든 Slave로부터 그 결과를 받은 후 Master가 수행을 완료한다.

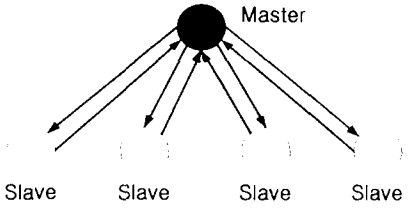

그림 2 마스터-슬레이브 계산 모델

동종의 노드들을 가정하고 임의의 한 노드를 Master Node로서 선정한다. Master Node역시 Task를 수행하나 부하 균등화 작업으로 인한 Overhead가 Slave Node보다 크다. 부하 주기마다 다음과 같은 부하 균등화 과정을 수행한다.

1) Master Node는 Broadcast를 하여 각각의 Slave Node에게 부하 균등화 요청 메시지를 보낸다.

$$C_{master} = (N - 1) \times C_{msg}$$

2) 각각의 Slave Node는 Global Gathering Method을 사용하여 각각의 노드 정보를 Master Node에게 보낸다.

$$C_{response} = (N - 1) \times C_{msg}$$

3) Master Node는 정책에 따라 과부하 노드와 저부하 노드를 결정한다. 평균으로부터 $\pm 1/2\alpha$% 내에 속한 노드를 중간부하라고 결정한다.

4) Master Node는 각각의 Slave Node에게 Broadcast하여 과부하, 중간부하, 저부하 여부와 이주시켜야 할 Task양을 알려준다. 이 때 결과를 통보 받지 않은 노드는 자신이 부하균등화에 참여하지 않는 중간부하임을 알게 된다.

$$C_{master} = (N - 1 - M) \times C_{msg}$$

5) Broadcast로 지시된 명령에 맞게 과부하에서 저부하로 Task가 이동한다. Sender Node는 부하 전달 가능 여부를 묻는 메시지를 보내고, Receiver Node의 응답에 따라 Migration시키므로 통신비용이 추가로 든다. Task개수에 비례하여 통신비용이 증가한다.

$$C_{task} = 단위시간 \times Task개수$$

$$C_{mig} = (2 \times C_{msg}) + C_{task}$$

4. 시뮬레이션

본 논문에서는 제안한 알고리즘의 성능을 분석하기 위하여 시뮬레이션을 하였다. 비교대상으로 Central Dynamic Load Balancing Strategy와 Decentral Dynamic Load Balancing Strategy[5]를 사용하였다.

현존하는 구현된 클러스터 시스템은 대부분 Switch구조로 Interconnection되어 있으므로 아래와 같은 Switch구조의 Topology를 가정한다. 불규칙적인 실제 클러스터 환경을 구성하기 위해 총 31개의 노드를 1, 2, 4, 8, 16으로 나누어 분산시켰다.

LB을 하지 않았을 때 약 10,000단위시간을 수행하는 일을 각 노드당 100개의 Task로 나누어 수행시키고, 시스템 전체적으로 퍼져있는 Task의 평균수행시간을 E라 할 때 부하 정보를 보내기 위해 메시지를 전달하는 데 걸리는 시간(C_{msg})을 E의 1.66%에서 8.33%까지 변화시키고 Task를 이웃 노드로 이동하는 데 걸리는 시간(C_{task})은 C_{msg}의 10배로 하였다. 그룹 내에서의 통신비용을 1이라 할 때 Main Switch를 거치는 즉, 다른 그룹간의 통신비용은 1.4배의 overhead를 추가하였다.

표 1 시뮬레이션 파라메터

Parameter	의 미	사용된 값
Pe	부하 교환 주기	1000단위시간
C_{msg}	부하 정보를 보내고 받는 비용	E의 1.66~8.33%
C_{master}, $C_{receiver}$	Master Node와 Slave Node들간에 수행되는 통신 비용	$(N-1) \times C_{msg}$
C_{task}	Task 이주비용	단위시간 × task size
C_{mig}	Task 이주 전체비용	$(2 \times C_{msg}) + C_{task}$
α	Load Margin for Task Transfer	평균부하량 ± 0.5α%내의 노드
N	노드 개수	31
S	Switch Overhead	C_{msg} × 1.4

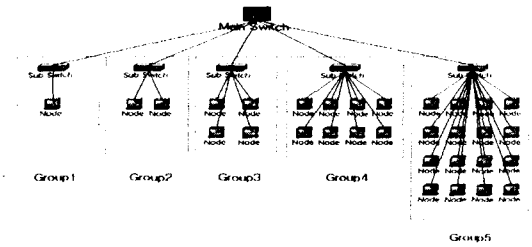

그림 3 네트워크 구조

5. 결과

본 논문에서는 통신비용에 따라 부하균등에 참여하는 부하개수를 변화시키는 동적 부하균등화법을 보였다. 여기서 통신비용은 Task 평균수행시간에 대한 상대적 통신비용을 말한다. 시뮬레이션 결과는 5번 실험을 반복하여 평균값을 사용하였다. 그림 4~7을 보면 그룹내의 노드의 개수가 적어 부하균등화를 위한 정보 교환비용이 적은 Decentral LB이 C_{msg}가 증가하여도 그 변화폭이 완만함을 볼 수 있었다. 그러나 Global Information을 얻을 수 없어 이주를 통한 이득율이 적어 C_{msg} < 3.33% 일때는 Central LB보다 성능이 낮음을 볼 수 있었다. C_{msg}가 1.66~3.33%범위내외에서는 Margin을 쓴 방법이 효율이 적어 이득율이 적었으나 C_{msg} = 3.33%내외를 경계로 Margin을 쓴 방법이 더 효율적임을 알 수 있었다. 또한 통신 비용이 커질수록 불필요한 부하균등화를 하지 않으므로 완만한 경사로 실행시간이 커짐을 볼 수 있었다. 그림 8은 부하균등에 참여하는 노드개수를 변화시켜 본 Speedup 그래프이다. C_{msg} = 3.33%이고 α = 48%일 때 Margin을 쓴 부하균등화가 가장 좋은 Speedup을 보임을 알 수 있었다. 결과적으로 통신비용이 작을 때는 일반적인 동적 부하균등화방법이 유리하며 통신비용이 증가할수록 부하가 큰 일부노드만 부하균등화를 하는 동적여분 부하균등화가 좋은 결과를 보여줌을 알 수 있었다.

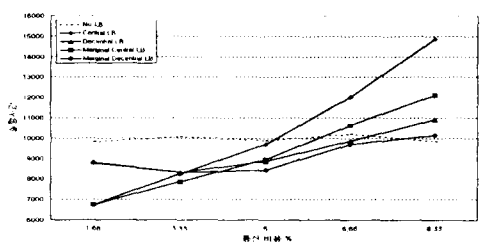

그림 4 통신비용에 따른 실행시간 변화(α = 35%)

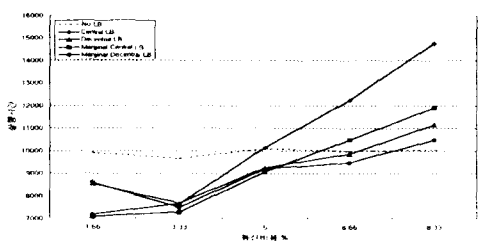

그림 5 통신비용에 따른 실행시간 변화(α = 48%)

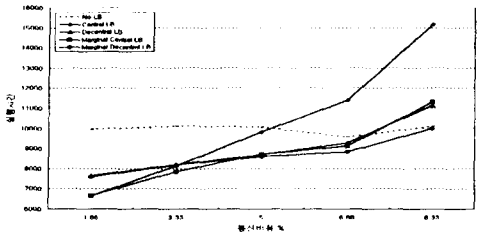

그림 6 통신비용에 따른 실행시간 변화(α = 67%)

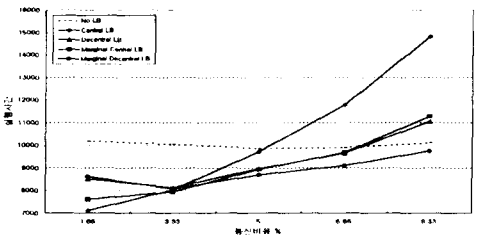

그림 7 통신비용에 따른 실행시간 변화(α = 80%)

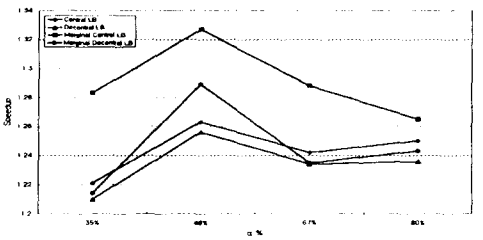

그림 8 α% 변화에 따른 이득율 변화(C_{msg} = 3.3%)

6. 참고 문헌

[1] Rajkumar Buyya, "High Performance Cluster Computing Volume 1 - Architectures and Systems"

[2] M. Cermele, M. Colajanni, G. Necci, "Dynamic Load Balancing of Distributed SPMD Computations with Explicit Message-Passing", IEEE97, ISSN:0-8186-7879-8/97

[3] Wentong Cai, Bu-Sung Lee, Alfred Heng, Li Zhu "A Simulation Study of Dynamic Load Balancing for Network-based Parallel Processing", IEEE 1997, ISSN : 1087-4089/97

[4] Marc H. Willebeek-LeMair, Anthony P. Reeves, "Strategies for Dynamic Load Balancing on Highly Parallel Computers", IEEE Transactions on Parallel and Distibuted Systems, Vol 4, No. 9

[5] 박경우, 김병기, "분산 시스템에서 동적인 혼합 부하균형 알고리즘", 한국정보과학회 논문지 제21권 제4호