작업생명주기를 고려한 부하균등정책

○
박 수, 홍영식
동국대학교 컴퓨터공학과

Load balancing with Process lifetime Distribution

Park su , Hong young sik
Dept. of Computer Engineering, Dongguk Univ.

요 약

분산시스템에서는 각 노드에서 작업이 생성되어 수행되고 소멸한다. 하지만 특정노드에 집중되어 작업이 수행됨으로써 전체적인 시스템에 부하의 불균형이 초래된다. 각 노드에 작업을 균등하게 나누고 이용률을 극대화시킴으로써 전체적인 시스템의 부하균형을 해결하려는 연구가 활발히 진행되고 있다. 대표적인 방법으로 배치와 이전이 있는데 배치는 작업이 생성될 때만 저부하 노드에서 수행되고 과부하상태가 되어도 다른 곳으로 이전할 수 없다. 반면 이전은 능동적으로 과부하가 되면 작업이전이 되지만 노드가 과부하가 될 때까지 이전정책이 적용되지 않는다. 본 논문에서는 작업이름에 대한 사전정보와 현재 부하상태를 고려한 이전정책을 적용시킴으로써 부하균등을 이루는 적응적인 이전정책을 제안한다.

1. 서론

분산시스템에서 각 노드는 다른 노드에 존재하는 다른 자원을 자원 관리자를 통해 접근한다. 각 노드에서 요구되는 자원을 충족시켜주지 못할 때, 노드는 자원의 할당을 기다리게되어 작업의 수행이 지연된다. 특정노드에 작업이 집중되어 이러한 현상이 나타나고, 다른 노드들의 작업량이 적은 상태로 남게되면 시스템의 부하가 불균형한 상태가 된다. 부하불균형상태가 되면 과부하 상태인 노드에서 수행되는 작업은 자원을 가지고 수행할 수 있는 기회가 줄어들기 때문에 상대적으로 작업을 종료할 때까지 긴 시간이 요구된다. 따라서 시스템의 전체적인 효율성이 떨어지게 된다. 이러한 부하불균형을 해결하기 위한 방법으로 이전정책[2]과 배치정책(placement)[1,5]이 있다. 이전정책은 현재 노드가 과부하 상태가 되면 노드에 부담을 주는 작업을 선정하여 저부하(light load)상태의 노드로 이동시켜 계속 수행함으로써 과부하 상태의 노드 부담을 줄이는 것이다. 배치정책은 작업이 생성될 때 작업의 사전정보나 노드의 상태에 따라서 저부하 노드로 작업을 이동시켜 수행시키는 방법이다. 이전이나 배치의 장점은 과부하상태 노드의 작업을 저부하 상태 노드로 이동시킴으로써 각 작업의 수행시간을 단축시킬 수 있다는 것이다. 즉, 부하를 시스템에 전체적으로 퍼지도록 함으로서 분산시스템 전체의 효율성을 극대화시킬 수 있다.

본 논문에서는 작업이 생성될 때 과거의 행동패턴을 주시하여 원격실행을 결정하는 배치정책과 과부하 노드의 작업을 저부하 노드로 이동시키는 이전정책을 적용시킨 적응적 이전정책 모델을 제안한다. 2장에서는 이전과 배치에 관한 관련연구를 기술하고, 3장에서는 본 연구에서 제안하는 적응적 이전정책을 기술한다. 4장에서는 실험 및 결과를 분석하고, 5장에서 결론 및 향후과제를 제시한다.

2. 기존연구

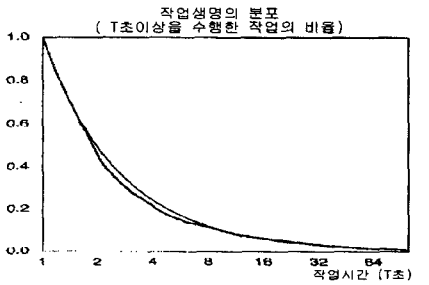
작업생명주기를 고려한 이전정책[2]은 노드가 과부하 상태일 때 현재의 노드들의 상태를 고려하여 작업을 선정한 후 이전시킨다. 이전을 수행하기 위한 기준으로 작업의 수행시간을 이용한다. 작업이 실행되어 수행시간이 임계값을 넘어가게 되면 다른 노드로 이전된다. 이전을 하기에 적당한 선정기준은 다음 식[1]과 같다. (Prun은 작업의 수행시간이고, MC는 이전비용이다. n은 발생노드의 작업개수이고, m은 목적노드의 작업개수다.)

$$Prun > \frac{1}{(n-m)} \cdot MC \qquad [식1]$$

위 식에서 n, m에 의해서 이전을 수행하기 위한 발생노드와 목적노드를 결정할 수 있다. 그리고 이전시키기에 적당한 작업은 수식 우측의 값보다 더 큰 작업수행시간을 요구하는 작업이다. 식[1]에서 필요한 변수들은 모두 현재 노드들의 상태이므로 이전정책을 위해서 추가 변수가 필요 없다.

이전이 빈번히 발생하면 이전비용이 매우 크므로 시스템에 부담을 줄 수 있다. 그러나 작업의 생명분포를 조

사한 결과[2]로 많은 작업이 생성, 수행되지만 대부분이 짧은 수행시간을 가지고 있다는 것을 알 수 있다([그림 1]참조). 즉, [그림1]에서 전체성능 저하의 주원인이 되며 매우 긴 작업수행시간을 갖는 작업들이 전체작업개수에 비해 매우 적다는 것을 알 수 있다. 따라서 이전될 가능성이 높은 장기간 수행되는 작업들의 개수가 단기 수행 작업들보다 상대적으로 매우 적으므로 이전이 많이 발생하지 않는다.



<그림 1> 작업생명분포도[2]

그림[1]에서 알 수 있는 또 다른 사실은 장기간 수행되는 작업들의 숫자는 적지만 장기간 수행이 지속되므로 전체시스템에 부하를 주는 영향이 가장 크다. 따라서 영향이 큰 장기간 수행되는 작업을 다른 노드로 이전시킴으로서, 전체 시스템의 성능을 높일 수 있다.

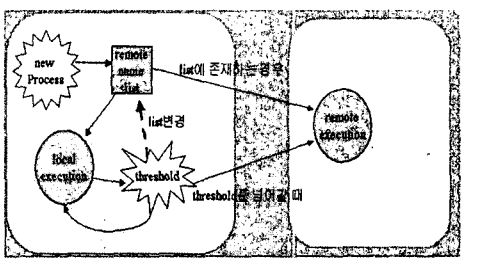
부하균등정책으로 배치를 이용하는 UTOPIA[5]는 사용자가 원격작업명단을 직접 변경하면서 유지 관리한다. 원격작업명단에 속하는 작업이 생성되면 배치정책으로 원격 실행을 행한다. 작업명단을 따로 관리하므로 시스템 커널의 변경이 없다. 하지만 원격작업명단을 정적으로 유지하게 되므로 시스템의 상태를 올바르게 예측하여 표현하는 것은 매우 어려운 일이고, 만약 작업명단의 작성이 작업생성 패턴과 일치하지 않는다면 배치정책이 적용되지 못하여 성능저하를 가져올 수 있다.

작업명단을 유지하는 다른 정책으로 과거정보를 이용한 배치정책이 있다[1]. 각 노드에서 수행되는 모든 작업의 이름과 평균수행시간의 정보를 유지하여 새로운 작업이 생성되었을 때, 평균수행시간이 시스템에서 정의한 임계값을 넘어가게 되면 원격실행 한다. 과거의 작업패턴을 고려하는 방법이므로 현재 노드가 과부하상태가 아니더라도 임계값을 초과하는 작업들은 전체 노드들 중 최저부하인 노드에서 실행시킨다. 하지만 노드에서 생성된 모든 작업에 관한 정보를 유지해야하므로 작업명단의 크기가 매우 커지는 단점이 있다.

3. 적응적인 이전정책

본 논문에서 제안한 모델은 배치정책과 이전정책을 병행 수행하여 부하균등을 이루는 정책이다. 작업이 생성될 때 먼저 배치정책을 적용하여 원격실행이 요구되는 작업을 원격실행 하고, 그렇지 않은 작업은 지역실행 한다. 배치정책을 통해 지역실행이 결정되었다 하더라도 작업수행시간이 임계값을 초과하게 되면 다른 노드로 이전하는 이전정책을 적용한다. 따라서 항상 전체시스템을 저부하 상태로 유지할 수 있다.

배치정책은 작업이 생성될 때, 작업명단을 조사하여 이전정책보다 먼저 수행된다. 새로운 작업이 생성될 때 작업명단을 검색하여 이름이 존재하는지 확인한다. 작업명단에 존재하는 작업이면, 과거 실행패턴으로 지역실행 시 과부하를 유발할 수 있음을 의미하므로 원격 실행시킨다. 작업명단은 작업이름과 작업의 평균수행시간을 포함한 레코드를 유지하는데 초기상태에는 작업명단이 비어있다. 만약 작업명단에 존재하지 않으면 지역실행을 한다. 현재 노드가 과부하 상태가 되면, 노드에서 [식1]을 만족시키는 작업을 선정하여 저부하 상태의 노드로 이동시킨 후 계속 수행한다. 이전하도록 결정된 작업은 다음에 다시 생성되어도 또 이전될 가능성이 높으므로 작업명단에 추가한다. 메모리한계에 도달하여 더 이상 작업명단에 기록할 수 없는 경우, 작업명단에서 작업의 평균수행시간이 제일 작았던 작업이름과 교체를 한다. ([그림2]참조)



<그림 2> Adaptive Migration (배치 + 이전)

```
if process name exist in DNameList then
  start placement_strategy(process, host, target);
endif;

if node is high load then
  migrant = find_migrant(host, target);
  if no migrant exist then
      End;
  endif;
  Update_DNameList( migrant->name );
  target = choose_target_host ();
  start migration_strategy(migrant, host, target);
endif;
```

<그림3> 적응적 이전정책 알고리즘

적응적 이전정책 알고리즘은 위의 [그림3]과 같다. 현재 실행되려는 작업이름이 존재하면 배치를 시키고, 그렇지 않으면 수행을 한다. 수행도중에 다른 노드가 고부하가 되면 이전을 수행한다. 본 논문에서 노드가 과부하 상태인지 조사해서 이전을 수행할지를 검사하는 때는 주기적으로 검사하는 것이 아니라, 새로운 작업이 생성될 때만 수행한다.

4 설계 및 결과 분석

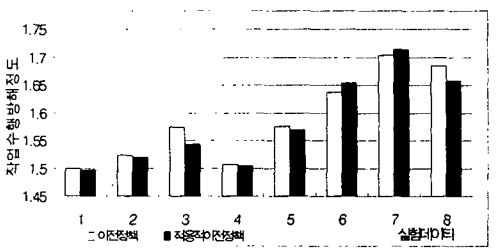
이전정책에 배치를 추가하여 전체적인 부하균등이 이루어지는 정도를 측정을 위해서 methusela[2] 시뮬레이터를 통하여 실험하였다. 위 실험에서 사용한 작업 데이타는 시뮬레이터에서 제공되고, 여섯 개의 동일한 컴퓨터에서 생성, 수행된 작업의 정보들을 한시간 단위로 잘라서 8개의 데이터로 구성되어 있다.

현재 수행되는 작업이 다른 작업들에 의하여 받는 영향을 정량적으로 알아보기 위해 작업수행방해정도(SDP)를 [식2]에서 정의한다[2].

[식2] $SDP(P) =$

전체수행시간(p) / CPU사용시간(p)

다음 [그림4]는 작업의 평균 작업수행방해정도를 나타내는 그림이다.



<그림 4> 평균 Slowdown

작업수행방해정도의 수치는 이상적인 경우 하나의 작업이 중앙처리장치를 100% 점유할 때 1의 값이 나온다. 따라서, 반드시 1의 값을 넘게된다. 작업수행방해정도의 숫자가 커질수록 현재 노드가 과부하상태를 유지하기 때문에 작업을 제대로 수행시키지 못하여 작업종료시간이 늦추어지게 됨을 뜻한다. [그림4]의 결과는 이전정책이 배치정책에 비해 1에 가까운 수치여서 부하균형이 매우 잘 이루어지는 정책이라는 것을 알수 있다. 하지만 이전정책을 수행하기 전에 미리 과부하를 만들 수 있는 가능성을 가진 작업을 먼저 배치시킴으로써 노드가 과부하가 될 기회를 줄여 조금 더 나은 부하균등을 수행하고 있음을 알 수 있다. [그림4]의 실험데이터 6번과 7번 데이터는 적응적 이전정책이 낮은 성능을 보이고 있다. 이유는 배치과정에서 이전되지 말아야할 작업들이 작업명단에 선정되어서 이전되기 때문에, 불필요한 이전비용이 증가되기 때문이다.

5. 결과 및 향후과제

배치정책은 작업이 생성될 때를 제외하고는 부하균등을 할 수 없고, 이전정책은 노드가 과부하상태가 될 때까지 기다려야 하는 단점이 있다. 본 논문은 배치의 작업이 생성될 때 부하가 적은 곳에서 수행하도록 하는 장점과, 수행한 이후에 노드가 과부하가 되었을 때 이전을 통해서 시스템의 부하균형을 이루도록 하는 이전정책의 장점을 살린 모델을 제안하고 실험하였다. 결과로 적응적 이전정책은 과부하상태가 되어 이전정책을 통해서 이동되던 작업을 배치작업을 통해서 좀더 빠르게 부하균등을 수행할 수 있었다. 본 논문에서는 중앙처리장치 자원만을 사용하는 작업으로 정의하고 실험하였지만, 입출력장치와 같은 자원도 동시에 이용하는 작업들도 많이 존재하므로, 향후 연구과제로 여러 자원을 이용하는 이전정책이 더 연구되어야 할 것이다. 그리고 한 작업이 하나의 노드에서만 수행되는 것을 알아보았는데 여러 개의 노드에 걸쳐서 동시에 작업이 수행되는 것에 대한 이전연구도 추가되어야 할 것이다.

참고문헌

[1] Svensson, A. 1990. History, an intelligent load sharing filter. In the IEEE 10th International Conference on Distributed Computing Systems. IEEE, New York, 546-553.

[2] Mor harchol-balter and Allen B. Downey, 1997, Exploiting process lifetime distributions for Dynamic load balancing, ACM Transactions on Computer Systems, vol. 15, No. 3, page 253-285

[3] Krueger, P. And Livny, M. 1988. A Comparison of preemptive and non-preemptive load distributing. In the 8th International Conference on Distributed Computing Systems. IEEE, New York, 123-130

[4] Leland, W.E. and Ott, T.J. 1986. Load-balancing heuristics and process behavior. In Proceedings of Performance '86 and ACM SGIMETRICS. Vol. 14. ACM, New York, 54-69..

[5] Songnian Zhou, Jingwen Wang, Xiaohu Zheng, and Pierre Delise, 1993, UTOPIA: A Load sharing Facility for Large, Heterogeneous Distributed Computer Systems, Software- Practice and Experience. 23, 2 (Dec), 1305-1336