동질 시스템에서 제한된 복제를 사용하는 스케줄링 알고리즘

차명진° 정진하 윤완오 신광식 최상방
인하대학교 전자공학과
sangbang@inha.ac.kr

Task Scheduling Algorithms Using Constrained Duplication in Homogeneous Systems

Myongjin Cha°, Jinha Cheong, Wanoh Yoon, Kwangsik Shin, and Sangbang Choi
Department of Electronic Engineering, Inha University

요 약

동질 시스템에서 분산 프로그램을 스케줄링할 경우 프로세서 사이의 통신 시간으로 인해 복제를 사용하는 태스크 스케줄링 알고리즘이 복제를 사용하지 않는 방법에 비해 향상된 결과를 얻을 수 있다. 하지만 불필요한 복제로 인해 자원을 낭비하여 프로세서가 제한된 경우 전체 수행시간이 증가하게 된다. 그래서 본 논문에서 조인 노드를 스케줄링할 경우 불필요한 복제가 발생되는 문제점을 해결하는 알고리즘을 제안하였다. 이러한 문제점을 해결함으로써 프로세서의 제한인 있는 경우에 기존의 복제 알고리즘에 비해 전체 수행시간이 개선되는 결과를 얻을 수 있었다.

1. 서 론

다중컴퓨터의 구조 설계가 발전했음에도 불구하고 프로세서 사이의 통신은 분산 프로그램 실행에서 피할 수 없는 문제로 남아있다. 이러한 문제는 서로 다른 프로세서에 할당된 분산 프로그램의 태스크가 데이터를 교환할 때 발생된다. 같은 프로세서에 할당된 태스크 간의 통신 시간은 무시해도 될 정도로 작기 때문에 프로세서 사이에 생기는 통신 시간을 줄이기 위해서 태스크를 복제하는 방법을 사용한다. 이 방법은 분산 프로그램에서 중요한 태스크를 여러 프로세서에 복제하는 것으로 통신 시간으로 인해 기다리는 태스크의 시작 시간이 줄어들 수 있다. 결과적으로 전체 프로그램의 수행시간이 개선될 수 있는 것이다. 복제에 기반한 스케줄링은 높은 통신 시간이나 낮은 대역폭을 가진 워크스테이션의 네트워크와 같은 시스템에 유용하게 사용될 수 있다 [1].

복제에 기반을 둔 스케줄링 방법은 리스트나 클러스터링에 기반을 둔 스케줄링 방법에 비해 대체로 향상된 결과를 얻을 수 있다 [2]. 그러나 CPFD (Critical Path First Duplication) 알고리즘과 같은 경우 주어진 태스크의 시작 시간을 빠르게 하기 위해서 가능한 모든 부모/조상 (parents/ancestors) 노드를 복제하려 하기 때문에 불필요한 복제가 발생되어 많은 자원을 소비하게 된다 [3]. 또한, 태스크 그래프의 크기가 커짐에 따라 필요한 프로세서의 수가 급격하게 증가하므로 크기가 큰 문제에 대해서 실질적인 사용이 제한된다.

이러한 문제를 해결하기 위해서 SD 알고리즘 (Selective Duplication)에서는 부모 노드를 복제하여도 시작 시간이 개선되지 않으면 복제하지 않기 때문에 불필요한 복제가 줄어든다 [4]. 하지만 SD 알고리즘에서도 조인 노드를 할당할 경우 불필요한 복제가 발생되어 자원이 낭비된다. 자원의 낭비는 프로세서의 수가 충분할 경우에는 각 프로세서에서 자원을 확보할 뿐 전체 수행시간이 개선되는 결과를 가져오지 않는다. 하지만 자원이 제한되어 있다면, 즉 프로세서의 수가 충분하지 않다면 불필요한 자원 소비로 인해서 태스크의 시작 시간이 늘어나게 되어 결과적으로 전체 프로그램의 수행시간이 증가하게 된다.

그래서 본 논문에서는 부모 노드가 프로세서에 할당된 상태에서 조인 노드를 다른 프로세서에 할당하려 할 때 부모 노드의 복제가 필요한 경우를 찾아보았다. 이러한 경우 복제되는 위치에 부모 노드를 할당하는 알고리즘을 제안하여 전체 수행시간에는 영향이 없으면서 자원의 낭비를 줄이는 결과를 얻게 되었다. 또한, 태스크 그래프를 프로세서가 제한된 상황에서 SD나 CPFD 알고리즘으로 스케줄링한 경우와 본 논문에서 제안한 알고리즘으로 스케줄링한 경우를 비교해 본 결과 전체 수행시간이 줄어드는 것을 확인할 수 있었다.

2. 복제를 사용하는 스케줄링 알고리즘

병렬 프로그램은 각각의 태스크를 의미하는 노드와 태스크 간의 통신 시간을 나타내는 에지(*edge*)로 구성된 DAG (*direct* acyclic graph)로 표현할 수 있다. 태스크 스케줄링은 이러한 DAG에 있는 모든 노드를 프로세서에 할당하는 것이고 출구 (exit) 노드가 끝나는 시간을 전체 수행시간이라고 한다 [4].

복제 기법을 사용한 알고리즘은 모든 노드가 각각의 프로세서에서 가장 빨리 시작할 수 있는 시간인 est (earliest start time)를 구할 때 필요한 부모/조상 노드를 복제하여서 est를 최대한 작게 하는 방법으로 결과적으로 전체 수행시간이 줄어들게 되는 것이다. 그러므로 프로세서의 수가 충분할 경우 DAG를 CPFD 알고리즘을 사용하여 스케줄링하면 최적의 전체 수행시간을 얻을 수 있다.

복제 기법을 사용하는 스케줄링 방법이 그렇지 않은 것에 비해 향상된 결과를 얻을 수 있는 반면에 많은 자원을 소비한다는 문제점이 있다. 이를 알아보기 위해 포크-조인 (fork-join) 노드로 이루어진 간단한 DAG를 스케줄링하여 간트 차트 (Gantt Chart)로 비교해 보면,

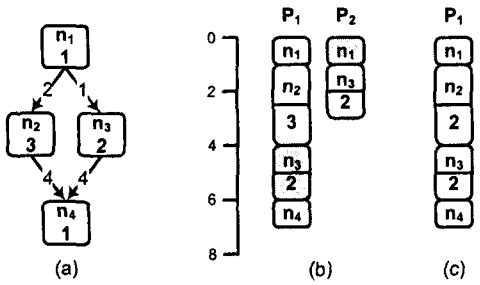

그림 1. 복제 알고리즘의 문제점을 알 수 있는 간단한 DAG. (b) CPFD 알고리즘으로 스케줄링한 경우의 간트 차트, (c) 하나의 프로세서에 모든 노드를 스케줄링한 경우의 간트 차트

그림 1(a)의 DAG를 CPFD 알고리즘을 사용하여 스케줄링하면 그림 1(b)처럼 되고, 프로세서 하나에 모든 노드를 할당하면 그림 1(c)와 같이 된다. 두 그림을 보면 쉽게 알 수 있듯이 전체 수행시간은 7로 같지만 CPFD 알고리즘을 사용할 경우 프로세서를 하나 더 사용하는 것을 알 수 있다. 즉, n3을 est가 가장 작은 p2에 할당하지 않고 p1에 할당하면 프로세서 하나만으로 동일한 전체 수행시간이 걸리게 된다 [5].

CPFD 알고리즘의 문제점은 그림 1(a)와 같은 DAG를 스케줄링할 경우 조인 노드(n4)의 선임 (predecessor) 노드(n2, n3)를 est가 가장 작은 프로세서에 할당하려고 하기 때문에 불필요한 복제가 발생한다. 일반적으로 조인 노드의 선임 노드가 다수의 후임 노드를 가지고 있을 경우에는(그림 2에서 n8의 n2, n3) est가 가장 빠른 프로세서에 할당해야 후임 노드의 시작 시간이 빨라지므로 이 때 발생되는 복제는 반드시 필요하다. 하지만 하나의 후임 노드만 있을 경우는(그림 2에서 n9의 n6, n7, n8) 조인 노드를 할당할 때 복제가 발생된다면 복제되는 위치에 부모 노드를 할당시켜도 조인 노드의 시작 시간이 늦어지지 않기 때문에 전체 결과에는 영향을 미치지 않으면서 복제를 줄일 수 있다.

3. 제안된 알고리즘

3.1 태스크 순서

SD 알고리즘에서 사용하는 태스크 순서에 레벨을 추가한다. 출구 노드가 가장 하위 레벨이고 출구 노드의 부모 노드를 그 다음 레벨로 설정하여 입구 (entry) 노드가 가장 상위 레벨이 된다. 그래서 SD 알고리즘에서 사용하는 태스크 순서로 레벨의 모든 노드를 할당한 후 그 다음 레벨의 태스크 순서대로 할당한다. 이렇게 설정하는 이유는 SD 알고리즘의 태스크 순서로 할당할 경우 조인 노드에 본 알고리즘을 적용할 때 선임 노드의 부모 노드가 할당되지 않은 경우가 발생하기 때문이다. 이런 방식으로 태스크 순서를 정할 경우 그림 2의 DAG에서는 {n1, n3, n2, n5, n4, n7, n6, n8, n9} 순서로 알고리즘을 적용한다.

3.2 Join 알고리즘

Join 알고리즘을 적용하기 위해서 DAG의 출구 노드부터 입구노드까지 검색해 본다. 아래 두 조건을 모두 만족할 경우 조인 노드를 할당할 때 불필요한 복제가 발생될 가능이 존재한다.

1. 다수의 선임 노드 중에서 후임 노드가 하나인 경우 (isolate)가 존재한다.
2. 선임 노드의 수행시간이 통신시간보다 작은 경우가 존재한다.

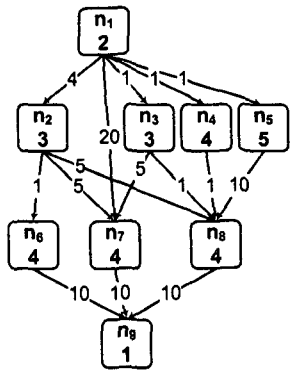

그림 2. 스케줄링 알고리즘을 비교하기 위한 DAG

그림 2의 DAG에서 이와 같이 조건을 만족하는 노드가 n7, n8, n9 라는 것을 알 수 있다. 하지만 n7, n8은 n9로 인해 join 알고리즘을 적용할 수 없다. 왜냐하면 n7을 join 알고리즘을 적용하게 되면 n9를 할당할 때 이미 n7이 할당된 상태이므로 n9를 할당하기 위해 n7이 복제되기 때문이다. 즉, CPFD 알고리즘이나 SD 알고리즘처럼 불필요한 복제가 발생되는 것이다.

조건에 만족하는 조인 노드인 n9를 찾은 후에 태스크 순서대로 SD 알고리즘을 적용하여 스케줄링한다. 스케줄링을 하다가 태스크 순서가 n7이 되었을 때 아래와 같은 표를 구성한다.

표 1. 조인 노드의 est

	P1	P2	P3	P4	est, ect
n6	6	*5	7	6	5, 9
n7	*8 (n2)	8 (n3)	10	10	8, 12
n8	10 (n5)	10 (n5)	* 10	11 (n5)	10, 14
n9	24	24	22	24	DAT
time slot	(12,24)	(9,24)	(7,10) (14,22)	(6,24)	

표 2. 알고리즘에 의해 est를 수정

	P1	P2	P3	P4	est, ect
n6	6	* 5	7	6	5, 9
n7	**8 (n3)**	8 (n2)	* 14	10	14, 18
n8	10 (n5)	10 (n5)	* 10	11 (n5)	10, 14
n9	21	22	* 19	22	19, 20

* 각 노드의 est를 선택

우선 각 프로세서에서 n6, n7, n8의 est를 구한다. 이 때 est만 구하고 할당을 하지 않으며 복제가 필요하다면 소괄호에 노드의 이름을 적는다. 모든 선임 노드의 est를 구하면 태스크 순서대로 가장 작은 est를 결정해야 한다. 태스크 순서가 가장 빠른 n7이 p1을 선택하고 n6는 p2를 선택하고 n8은 p3을 선택하면 된다. 이렇게 선임 노드의 est를 결정하면 각 프로세서에서 n9의 DAT (data arrival time)를 결정할 수 있다. 여기서 선임 노드의 ect (earliest finish time)와 각 프로세서에서 n9의 DAT 사이에 시간 슬롯 (time slot)이 생긴다. 여기까지의 내용이 표 1에 표시되었다.

다음으로 표 1을 통해 조인 노드에 불필요한 복제가 생기는지 알아본다. DAT가 가장 작은 p3에서 조인 노드 n9의 est를 구할 경우 n9의 DAT와 n8의 ect 사이에 (14, 22) 시간 슬롯이 있는 것을 알 수 있다. 여기서 다음 두 조건이 만족하면 불필요한 복제가 발생되는 것이다.

1. p3에서 n9의 miip(most important immediate parent)인 n7의 ect가 시간 슬롯의 종료시간 (finish time) 보다 작다.
2. n7의 수행시간이 시간 슬롯의 크기보다 작다.

조인 노드 n9가 p3에서 두 조건을 만족하므로 n7을 est가 가장 작은 p1에 할당하지 않고 p3에서 복제되는 위치에 할당을 하면 CPFD나 SD 알고리즘의 불필요한 복제를 막을 수 있다.

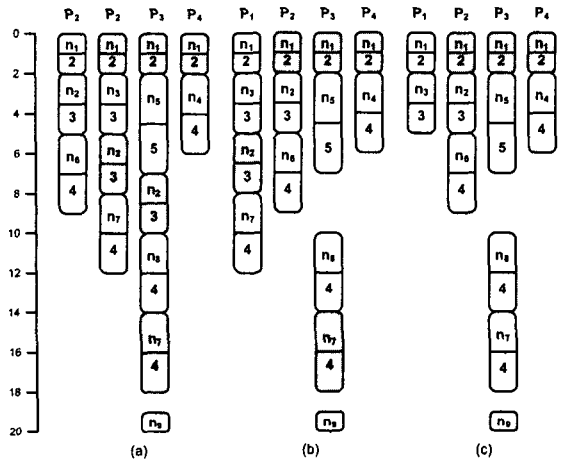

그림 3. 3가지 알고리즘으로 스케줄링한 경우의 간트 차트. (a) CPFD 알고리즘, (b) SD 알고리즘, (c) Join 알고리즘

그림 2의 DAG를 CPFD 알고리즘, SD 알고리즘, 제안된 알고리즘으로 스케줄링한 경우 그림 3와 같이 되는 것을 알 수 있다. 여기서 보면 SD 알고리즘에서 n2가 p3에서 복제되지 않으므로 p3의 자원을 확보하는 것을 볼 수 있고, join 알고리즘을 사용해 n7이 p3에 할당되면 p1의 자원을 확보하는 것을 알 수 있다.

4. 성능 분석

본 논문에서 제안하는 알고리즘은 조인 노드의 선임 노드가 하나의 후임 노드를 가질 경우 (isolate) 불필요한 복제가 발생되지 않게 하여서 자원 낭비를 줄이는 것이다. 자원의 낭비를 줄이는 것은 프로세서가 충분할 경우에는 단지 각각의 프로세서에서 자원을 확보할 뿐 전체 수행시간에는 영향이 없다. 하지만 프로세서가 제한된 경우 그림 1(a)와 같은 DAG가 큰 DAG의 일부이면 자원의 낭비를 줄이는 것은 개선된 결과를 가져온다는 것을 그림 4를 보면 알 수 있다.

그림 4(a)를 CPFD나 SD 알고리즘으로 스케줄링하면 그림 4(b)와 같은 결과를 얻을 수 있다. 그림 4(c)는 join 알고리즘을 통해 불필요한 복제를 예측하고 n3을 p1에 할당하여 2개의 프로세서만으로 전체 수행시간 9가 나오는 것을 볼 수 있다. 하지만 프로세서의 수를 2개로 제한하고 CPFD나 SD 알고리즘으로 스케줄링하면 그림 4(d)와 같이 되어서 전체 수행시간이 10이 나오게 되는 것을 알 수 있다.

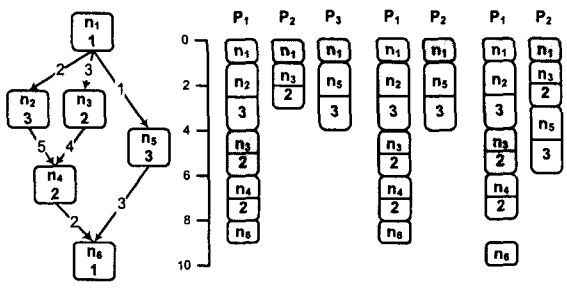

그림 4. 프로세서를 제한한 경우에 성능을 비교하기 위한 DAG와 간트 차트. (b) CPFD나 SD 알고리즘을 사용한 경우, (c) Join 알고리즘을 사용한 경우, (d) 프로세서의 수를 2개로 제한하고 CPFD나 SD 알고리즘을 사용한 경우

본 논문에서 제안한 알고리즘을 사용하면 n2를 할당할 때 후임 노드 n4이 조인 노드인 것을 알 수 있고 n3이 n4가 할당될 때 p1에 복제된다는 것을 알 수 있다. 그러므로 n3를 est가 가장 적은 p2에 할당하지 않고 p1에 할당한 후 n4를 할당하면 p2의 자원을 확보할 수 있어서 n5의 est가 3이 아니라 프로세서의 제한이 없는 경우와 마찬가지로 1이 되어서 n6의 est가 9가 되는 것이다.

5. 결론

본 논문에서는 복제를 사용하는 스케줄링 알고리즘에서 불필요한 복제가 발생되는 것을 개선하였다. 조인 노드의 부모 노드를 할당할 때 복제가 발생된다는 것을 예상하여 복제되는 위치에 부모 노드를 할당함으로써 불필요한 복제를 줄일 수 있었다. 이로 인해 자원을 낭비를 줄여서 프로세서의 수가 제한된 경우 전체 수행시간이 줄어드는 결과를 가져왔다. 또한, 대부분의 복제를 사용하는 스케줄링 알고리즘이 급격히 증가하는 프로세서의 수로 인해 실질적인 사용에 제한이 있었으나 제안된 알고리즘을 적용하면 자원의 소비가 줄어들어 이러한 문제를 해결할 수 있을 것이다.

참고문헌

[1] T.L. Casavant and J.G. Kuhl, " A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems," *IEEE Trans. Software Eng.*, vol. 14, no. 2, pp. 141-154, Feb. 1988.

[2] Y.K. Kwok and I. Ahmad, " Benchmarking and Comparison of the Task Graph Scheduling Algorithms," *J. Parallel and Distributed Computing*, vol. 59, no. 3, pp. 381-422, Dec. 1999.

[3] I. Ahmad and Y.K. Kwok, " On Exploiting Task Duplication in Parallel Program Scheduling," *IEEE Trans. Parallel and Distributed Systems*, vol. 9, no. 9, pp. 872-892, Sept. 1998.

[4] Savina Bansal, Student Member, " An improved duplication strategy for scheduling precedence constrained graphs in multiprocessor systems," *IEEE Trans. Parallel and Distributed Systems*, vol. 14, no. 6, pp.533-544, , June 2003.

[5] C.I. Park and T.Y. Choe, " An Optimal Scheduling Algorithm Based on Task Duplication," *IEEE Trans. Computers*, vol. 51, no. 4, pp. 444-448, Apr. 2002.