韓國 OA學會 論文誌
第4卷 第4號, 1999.12.

99-4-4-1-6

다중프로세서에서 비순환 타스크 그래프의 최적 스케쥴링에 관한 연구

조 민 환*

A Study on Optimal Scheduling with Directed Acyclic Graphs Task onto Multiprocessors

Cho Meen Hwan*

요 약

병렬 처리시스템 환경에서 효율적인 타스크 스케쥴링에 관한 연구로서 타스크 전체 수행 시간을 단축시는데 목적을 두고 있다. 멀티프로세서 시스템에서 선행 조건 을 갖는 타스크 그래프의 타스크 스케쥴링은 시스템 처리시간에 많은 영향을 준다. 이 문제는 NP-hard로 알려져 있으며, 많은 사람들이 heuristic 방법으로 최적해에 접근하려고 노력 하고 있다. 우리는 기존 여러 방법들 (swapping, MH, DL)과 개선된 critical path schedule 방법과 상호 비교하였다. 다수개의 root와 다수개의 terminate를 가지는 방향성 비순환 그래프(Directed Acyclic Graph : DAG)를 Random 생성하여 시뮬레이션 한 결과 프로세서 수를 증가한 경우 개선된 Critical Path알고리즘이 실행 타스크의 탐색 시간 개선에 더 우수한 것으로 판명되었다.

Abstract

The task scheduling has an effect on system execution time in a precedence constrained task graph onto the multiprocessor system. This problem is known to be NP-hard, many people made an effort to obtain near optimal schedule. We compared modified critical path schedule with many other methods(CP, MH, DL Swapping) For testing this subject, we created randomly a directed acyclic task graph with many root nodes and terminal nodes simulation result convinced for us that the modified critical path algorithm is superior to the other scheduling algorithm.

* 창원전문재학 정보통신계열 정보처리전공 조교수
논문접수:1999.10.12. 심사완료:1999.12.4

Ⅰ. 서 론

최근 다중프로세스 시스템 환경에서 타스크 스케쥴링에 대한 연구가 많이 진행 되었다[3][8]. 여러 가지 선행 제약 조건을 갖는 타스크 상호 관계로 인하여 타스크 그래프에서 최적 타스크 할당을 구현하는 스케쥴러는 NP-hard로 알려져 있다 [9] [10] [11]. 휴리스틱 방법은 일반적으로 타스크 간 통신을 무시하는 것[4] [5] [6]과 스케쥴 결정에 타스크간 통신을 매우 중요시한 것 [3][7]으로 분류 된다. 본 논문에서는 다중 프로세서상의 중요한 타스크간 통신을 포함하는 타스크 그래프의 타스크들의 기존 프로세서 할당 알고리즘들을 분석하고 개선된 Critical Path 알고리즘과 비교를 위한 새롭고 효율적인 휴리스틱 알고리즘을 구현하여 분석 한다.

Ⅱ장에서는 멀티프로세서 스케쥴링 문제를 정의하고 Ⅲ장은 타스크 스케쥴링 알고리즘인 실행시간만 적용한 알고리즘(CPE), 실행시간 과 통신시간을 적용한 알고리즘(DL) , Mapping Heuristic 알고리즘을 Ⅳ장은 개선된 Critical Path 알고리즘을 간단히 소개 하고, Ⅴ장에서는 Ⅲ, Ⅳ, 장에서 제시 된 알고리즘의 성능을 시뮬레이션 결과를 통해 실험 하는데 마지막 Ⅵ장에서는 결론을 제시한다.

Ⅱ. 다중프로세서의 스케쥴링 문제

병렬 프로그램은 다음과 같이 가중치 있는 방향성순환 타스크 그래프로서 특징 지어 진다. 즉 E={eij}는 V상에 선행 제한 혹은 부분 순서를 정의하는 방향성 에지 들의 집합이며 ri는 vi의 계산(실행) 시간 을 나타내고 가중치 {ri}를 가지는 프로 그램의 타스크 집합을 나타내는 V={vi},,i=1.2....N를 가지는 가중치 있는 직접 비순환 타스크 그래프로서 특징 지어진다. vi에서 vj로 향하는 G에 있는 에지 eij는 타스크 vi가 임의의 스케쥴에서 타스크 vj보다 선행해야 한다는 것을 의미한다. 내부 타스크 통신을 무시할 수 있는 타스크 그래프 경우에는 어떤 가중치도 E에 있는 에지와 관련이 없다. 우리는 무시할 수 없는 내부 타스크 통신을 가지는 타스크 그래프를 고려하고, cij가 vi에서 vj로의 통신량을 나타내는 가중치를 각 에지 eij와 결합한다.

타스크 그래프 G에서는 선행하는 것이 없는 타스크들이 입력 타스크라고 하며, 후행하는 것 없는 타스크들을 탈출 타스크라고한다. 우리는 G가 정확히 두 개의 입력 타스크와 세개의 탈출 다스크를 가진다고 가정한다. 10개의 타스크들을 가진 타스크 그래프 의 예가 [그림 1]에 있다. 이것은 비 순환(acyclic) 그래프이며 타스크 Ti에서 타스크 Tj로의 arc는 두

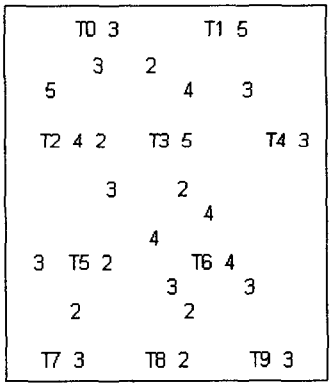


그림 1. 타스크 그래프

타스크 간의 선행관계를 나타낸다. Ti는 Tj의 부모(parent) 타스크이고 Tj는 Ti의 자식(child) 타스크이다. 한 타스크는 자신의 모든 부모 타스크가 처리를 완료한 후에 실행될 수 있다.

타스크 그래프들이 스케쥴될 수 있는 멀티프로세서 시스템을 위한 가능한 구조는 하이퍼 큐브와 그물, 스타, 링 등을 포함한다. 멀티프로세서는 동질이라고 가정하며, 그것의 통신 채널은 반 이중이라고 가정한다. 또한, 각 프로세서는 병렬적 계산과 타스크간 통신 모두를 수행하게 하는 분리된 통신 하드웨어를 포함한다. 이것은 한 프로세서의 계산과 통신 서브 시스템사이의 적당한 버퍼 공간이 있다는 것과 통신 서브 시스템이 동시에 모든 채널을 제어할 수 있다고 가정한다. 후자의 가정은 논쟁 때문에 대기 행렬의 지연을 고려할 필요성을 방지하며, 단지 대기 행렬 지연의 원인은 우리의 알고리즘에 의해

명백히 다룬다. 우선권 제약 타스크 그래프와 멀티프로세서가 주어진다면, 멀티프로세서 스케쥴링 문제는 완료 시간을 최소화하기 위한 방법으로 멀티프로세서상에 타스크 그래프의 비선점 스케쥴을 얻는 것이다. 이런 스케쥴링 방법은 NP-hard 로 알려져 잇고, 결론적으로, 휴리스틱 알고리즘은 적당한 계산 시간내에 만족할 만한 해결책을 산출해낼 수있다[2][13].

Ⅲ. 타스크 스케쥴링 알고리즘

3-1. Dynamic Level 알고리즘

Global clock이 없는 동적 Level 스케쥴러로써 프로세서 스케쥴링과 채널 스케쥴링을 동시에 고려한스케쥴링이다[1][15].

각 스케쥴링 단계에서는, 동적 레벨 스케쥴러가 타스크 그래프와 현재 스케쥴링 단계에 있는 부분 스케쥴과 멀티프로세서의 위상을 고려함으로써 각 프로세서에 대해 각각의 준비 타스크를 위한 우선 순위 리스트를 구성한다. 이것은 각 준비 타스크 프로세서 조합을 위해 정의되고 동적 레벨이라고 부르는 수량에 의해 주어지는 우선 순위이다. 현재의 스케쥴링 단계 Σ에서 하나의 프로세서 pj에 대해 주어진 부분 스케쥴을 가지는 타스크 vi의동적 레벨은 DL(vi, pj, Σ)라고 하며, 타스크 vi와 프로세서 pj 사이의 한쌍의 특징을 반영하며 다음과 같이 주어진다.

$$DL(vi,pj,\Sigma) = SL(Vi) - \max\{TF(pj,\Sigma), DA(vi, pj,\Sigma)\}$$

여기서 SL(vi)는 vi의 정적 레벨이라고 부르며, 타스크 vi에서 탈출 타스크까지의 가장 긴 경로의 길이이다. 한 경로의 길이는 경로상의 모든 타스크들의 실행 시간의 합으로서 정의 된다. TF(pj, Σ)는 부분 스케쥴 Σ가 주어질 때 프로세서 pj에 마지막 타스크가 스케쥴된 시간이며, 실행을 끝낸다. DA(vi, pj, Σ)는 부분 스케쥴 Σ가 주어질 때 모든 자료가 타스크 vi에 의해 요구되는 시간이 프로세서 pj에서 이용 가능하다는 것을 나타낸다. 타스크 Vm의 바로 인접한 선행자로부터 라우팅 알고리즘에 의한 채널

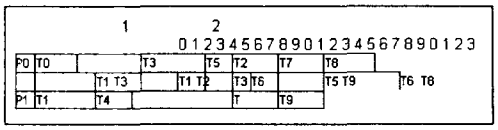


그림 2. Gantt Chart of Fig.1 DAG by Dynamic Level Algorithm

까지 모든 통신을 스케쥴한 후, 가장 일찍 끝나는 프로세서 pn상의 타스크 Vm을 스케쥴한다[1].

3-2. Mapping Heuristic 알고리즘

타스크 그래프에 있는 정보에 기초한 스케쥴링 과정으로 개시 전에 우선 순위를 계산한다. 이것은 하나의 타스크의 우선 순위가 그 타스크로부터 탈출 타스크 까지의 가장 긴 경로의 길이로서 정의되는 레벨에 의해 경정되는 타스크 그래프에 있는 각 타스크와 우선 순위를 결합한다. 여기에서, 경로의 길이는 경로에 있는 모든 에지들의 통신 시간의 합에 경로상의 모든 타스크들의 실행 시간의 합을 더한 것으로서 정의된다. 타스크들은 또 다른 타스크 보다 더 높은 레벨을 가지는 타스크가 후자보다 높은 우선 순위를 가질 것이라는 유일한 우선 순위가 할당된다. 두 개의 타스크가 동일 레벨을 가질 경우, 바로 인접한 후행자를 가진는 타스크에 보다 높은 우선 순위를 할당함으로써 일치하지 않게 한다. 만약 두 개의 타스크가 같은 레벨을 가지면 후행자의 수가 동일하다면, 타스크 중 하나에 높은 우선 순위를 무작위로 할당함으로써 불일치 시킨다[1][12].

3-3. Swapping 알고리즘

어떤 프로세서가 유휴(Idle) 상태 일 때 다른 프로세서의 큐에 대기 중인 첫 번째 ready 타스크의 (스케쥴된) 처리 시작 시간과 통신시간 합이 자신의 큐에 대기 중인 첫 번째 타스크의 (스케쥴된) 처리 시작 시간 보다 작거나 같으면 두큐의 타스크들을 서로 교체(swap)한다[15].

선행 관계를 고려할 때 각 프로세서는 다른 큐에 있는 타스크가 처리될 때 까지 기다릴 필요가 있을 수 있다. 이 때 그 프로세서는 유휴 시간(idle time)을 갖게 되는데 이것은 자신의 큐에 있는 전체 타스크들의 처리를 완료하는 시간을 늦추게 된다. 이러한 문제를 완화 시키는 한가지 방법은 각 프로세서들을 계속 busy상태로 유지하는 것이다. 어떤 프로세서가 유휴(idle) 상태이고, 또 다

른 큐 상에 ready 상태의 타스크가 존재할 경우, 그 프로세서가 ready 타스크를 실행하게 한다면 좋을 것이다.

타스크 교체는 유휴 상태의 프로세서가 존재할 때에만 고려된다. P0을 시간 t에서 유휴 상대의 프로세서, P1를 busy 상태의 프로세서라 하자. P0이 유휴 상태이므로 첫 번째 ready 타스크를 찾기 위해서 Q1를 검색할 것이다. 만약 P0이 Q1의 첫 번째 ready 타스크인 T2r을 찾았다면 두 큐의 타스크를 교체 함으로써 원래의 타스크 수정하게 될 것이다.

다스크 스와핑은 Q1 상의 타스크 T2r과 뒤따르는 모든 타스크들을 Q2으로 이동시킨다. 그리고 원래 Q0에 남아있던 모든 타스크들을 Q1의 뒤에 붙인다. 선행관계를 유지하기 위해서 각 타스크의 스케쥴된 처리 시작 시간과 완료 시간은 변경되지 않는다. 타스크 T2r이 Q1의 첫 번재 타스크가 아닐 경우, 원래 Q2에 있던 타스크들만이 Q0으로 교체된다는 것에 주의해야 한다.

Queue에 Ready 상태인 타스크가 존재 할 경우 스케쥴될 프로세서(pi)가 busy 상태이면 다른 프로세서 중 유휴(idle) 상태인 프로세서를 선택하여 Ready 상태인 타스크의 처리시간과 통신 시간의 합을 계산한다. 유휴 상태인 프로세서와 비교하여 작거나 같은 경우 교체(Swapping) 하여 이미 할당된 프로세서중 위 조건에 부합되지 않으면 새로운 프로세서(pi+1)에

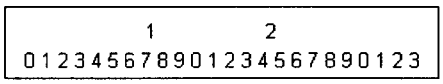


그림 3. Gantt Chart of Fig.1 DAG by swapping Algorithm

스케쥴링 하는 방법으로 타스크 스케쥴링을 향상시킨다.

Ⅳ. 개선된 Critical Path 알고리즘

모든 리스트 스케쥴러는 타스크 그래프를 스케쥴링 하기위해 N개의 스케쥴링 단계를 가진다. 각 스케쥴링 단계에서 준비 타스크 집합을 계산하며 우선순위 리스트를 정열하며, 가장 높은 우선순위를 가지는 요소를 선택한 다음에 요소 의해 정의되는 것으로서 스케쥴링 동작을 수행한다.

실행시간 Critical Path 알고리즘 (Execution Time Ceitical Path Algolithm : ECP) 을 대기 중인 타스크에서 실행시간과 출구까지 실행시간의 비를 계산하여 최대인 타스크를 스케쥴링 하는 것이다. ETi 는 준비된 타스크의 실행시간 이고 Vi 는 스케쥴링을

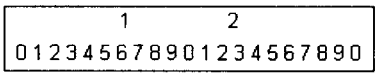


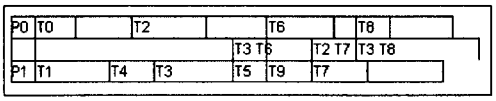
그림 4. Gantt Char of Fig.1 DAG by Modified (ECP) Algorithm

대기하고 있는 타스크이며 CPE(Vi,Pj,å)는 타스크 Vi 에서 출구 타스크 까지의 가장 긴 경로의 길이 이다.

ECP(Vi,Pj,å) = Eti / CPE(Vi,Pj,å)

여기서 ECP((Vi,Pj,å)의 값이 최대인 타스크를 선택하고 프로세서Pj는 D=Vi+C를 계산하여 D값이 최소인 프로세서(Pj)에 스케쥴링한다. C는 통신시간(비용) 이된다. [그림1] 타스크 그래프를 개선된 알고리즘(ECP)으로 스케쥴링한 결과 [그림4]와 같다.

Ⅴ. 시뮬레이션과 결과

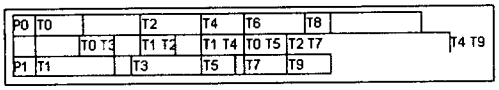


3, 4장에서 소개된 알고리즘들의 성능을 비교하기 위하여 동일한 조건 환경에서 Random하게 생성한 타스크 그래프를 파라메타인 타스크수(N), 프로세서수(P), 통신시간을 1(최소) ~ 5(최대) 까지, 타스크 처리시간을 1(최소) ~ 10(최대) 까지 고정하고, 타스크 수를 10, 20, 30, 40, 50, 60, 70, 80, 90, 100으로 이때 타스크 Level 수는 5, 10, 15, 20, 25, 30, 35, 40, 45, 50를 하고 edge 범위는 15~20, 30~40, 50~60, 60~70, 70~80, 80~90, 90~100, 100~110, 110~120, 프로세서 수 (P) = 2, 3, 4, 5 까지 시뮬레이션한 알고리즘들을 정리하면

알고리즘0 : Ready 타스크 중 실행시간이 최대인 Critical Path 타스크를 선택하여 스케쥴링 알고리즘 알고리즘1 : Ready 타스크중 실행시간과 통신시간이 최대인Critical Path 타스크를 선택한 스케쥴링 알고리즘

알고리즘2 : Mapping Heuristic 스케쥴링 알고리즘 알고리즘3 : Modified Critical Path 알고리즘 등이다.[그림5]은 알고리즘0 방법으로 실행한 결과이고, [그림6]은 알고리즘3 방법으로 실행한 결과 이다.

알고리즘0,1,2,3 방법 모두 프로세서수를 2~5개로



준비하여 타스크 수를 10~100까지 10개식 증가 시켜 실행한 결과, [그림7]은 프로세서를 2개를 실행 했을 경우 알고리즘 0과1이 우수하며 [그림8]은 프로세서를 5개로 실행 했을 경우 알고리즘 2와 3의 성능이 향상되었다.

VI. 결론

멀티프로세서 스케쥴링 문제는 완료 시간을 최소화 하기위한 방법으로 멀티프로세서의 프로세서들 상에 선행제한 타스크 그래프의 타스크들을 스케쥴링 하는 문제이다. 이런 문제는 NP-hard로 알려져 있기 때문에, 최적 스케쥴을 얻는 휴리스틱 알고리즘에 많은 연구가 이루어지고 있다.

Random 하게 생성된 DAG에 대한 휴리스틱 알고리즘들을 적용하여 비교분석 한 결과 root node

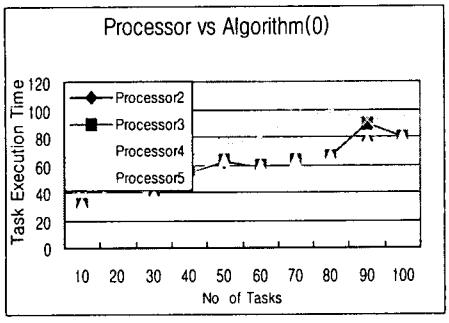


그림 5. 알고리즘0의 실행시간과 타스크수

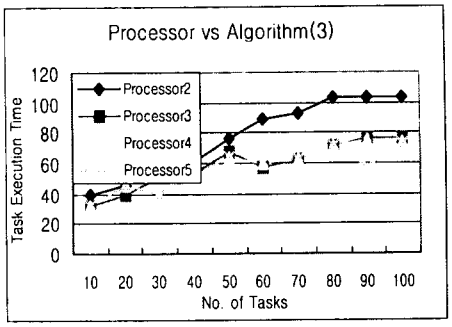


그림 6. 알고리즘3의 실행시간과 타스크수

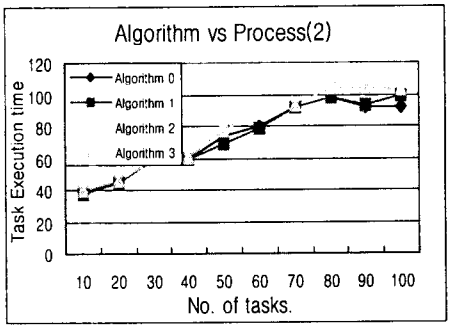


그림 7. 프로세서수2의 실행시간과 타스크수

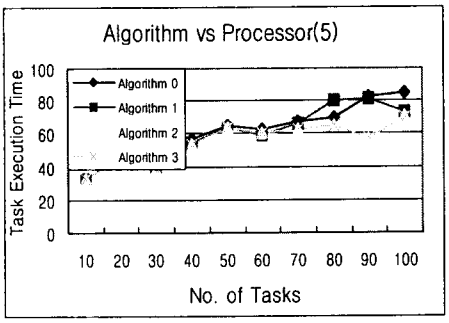


그림 8. 프로세서수5의 실행시간과 타스크수

혹은 Terminal node의 수보다 프로세서의 수가 작은 경우에는 Critical Path Algorithm 0,1 이 우수하고 프로세서가 root node 와 Terminal node수보다 많을 경우 Mapping Heuristic 알고리즘 과 개선된 Critical Path 알고리즘이 우수함을 알 수 있다. 전체적 평균 실행시간은 거의 동일하나 구현 Algorithm은 개선된

Critical Path 알고리즘이 가장 간단하여 전체 실행시간 단축에 기여 할 것으로 생각 된다.

참고문헌

[1] S. Selvakumar and C. Siva Ram Murthy. "Scheduling Precedence Constrained Task Graphs with Non-Negligible Intertask Communication onto Multiprocessors", IEEE Transaction on Parallel and Distributed systems, vol. 5. no. 3, March 1994, pp. 328-336

[2] H. El-Rewini and T. G. Lewis. "Scheduling parallel program tasks onto arbitrary target machines." J. Parallel Dist. Comput., vol. 9. no. 2, pp. 138-153, 1990.

[3] G. C. Sih and E. A. lee, "Scheduling to account for interprocessor communication within interconnection-constrained processor networks." Proc. Int. Conf. Parallel Proc. 1990.

[4] B. Shirazi. M. Wang and G. Pathak, " Analysis and evaluation of heuristic methods for static task scheduling," J. Parallel Distr. Comput., vol. 10. no. 3, pp. 222-232, 1990.

[5] H. Kasahara and S. Narita, "Practical multiprocessor scheduling algorithms for efficient parallel processing," IEEE Trans. comput., vol. C-33. no. 11, pp. 1023-1029, Nov. 1984.

[6] C. L. Chen, C. S. G. Lee, and E. S. H. Hou, "Efficient scheduling algorithms for robot inverse dynamics computation on a multiprocessor system." IEEE Trans. Syst. Man Cybern., vol. SMC-18. no. 5, pp. 729-743, Sept-Oct. 1988.

[7] C. Siva Ram Murthy and V. Rajaraman. "Task assignment in a multiprocessor system," Microprocessing and Microprogramming, vol. 26, pp. 63-71, 1989.

[8] S. J. Kim, "A general approach to multiprocessor scheduling," Ph. D. dissertation. Univ. Texas at Austin. 1988.

[9] C. C. Price and M. A. Salama. "Scheduling of precedence-constrained tasks on multi -processors," The Comput. J., vol. 33. no. 3, pp. 219-229, 1990.

[10] T. L. Kunii, S.Nshimura. and T. Noma, "The design of a parallel processing system for computer graphics." in Parallel Processing for Computer Vision and Display, P. M. Dew, R. A. Eamshaw. and T. R. Heywood. Eds. Reading. MA: Addison-Wesley, 1989.

[11] M. Y. Wu and D. D. Gajski, "Hypertool: A programming aid for message-passing systems."IEEE Trans. Parallel Distr. Syst., vol. 1. no. 3, pp. 330-343, 1990.

[12] C. S. G. Lee and C. L. Chen. "Efficient mapping algorithms for scheduling robot inverse dynamics computation on a multiprocessor system." IEEE Trans. Syst. Man Cybern., vol. 20. mo. 3, pp. 582-595, May-June 1990.

[13] S. Selvakumar and C. Siva Ram Murthy, "Scheduling precedence constrained task graphs with nonnegligible intertask communication onto multiprocessors." Tech. Rep., Dept. Comput. Sci. and Eng., indian inst. of Technol., Madras., Madras, India, Mar. 1991.

[14] K. Y. Kwok. and I. Ahmad, "A Static Scheduling Algorithm using Dynamic Critical Path for Assigning Parallel Algorithms onto Multiprocessors", International conference on parallel

processing, 1994, pp. II-155~159.

[15] Jiahuang Ji and Menkae Jeng, "Dynamic task allocation on shared memory multiprocessor system", International Conference on Parallel Processing, 1990,

저 자 소 개

조 민 환

77.2.~92.10. (주)쌍룡컴퓨터 근무

92.11.~94.2. 한진정보통신(주) 근무

86.9.~89.2. 숭실대학교 산업대학원 전자계산학과 (공학석사)

94.3.~현재 창원전문대학 정보통신계열 정보처리과정 조교수