대칭형 다중프로세서 시스템에서 태스크 중복기반의 클러스터링과 스케줄링

강오한° 조경미 김기남 김시관+
안동대학교 컴퓨터교육과, +금오공과대학교 컴퓨터공학부
°ohkang@andong.ac.kr +sgkim@se.kumoh.ac.kr

Task Duplication Based Clustering and Scheduling on Symmetric Multiprocessor Systems

Oh-Han Kang° Kyung-Mi Cho Ki-Nam Kim Si-Gwan Kim+
°Andong National University +Kumoh National Institute of Technology

요 약

대칭형 다중프로세서 (SMP: Symmetric Multiprocessors) 시스템은 고성능의 병렬 연산을 위한 중요하고 효과적인 기반환경을 제공하고 있다. SMP에서 태스크 클러스터링과 스케줄링 기법은 시스템의 성능에 큰 영향을 미친다. 본 논문에서는 버스 기반의 SMP에서 사용할 수 있는 태스크 중복 기반의 클러스터링과 스케줄링 기법을 소개한다. 본 논문에서 제안한 클러스터링 기법에서는 휴리스틱을 사용하여 중복할 태스크를 선택한 후 프로세서에 할당하고, 스케줄링 기법에서는 잠재하는 통신 충돌을 방지하기 위하여 네트워크 통신 자원을 사전에 할당한다. 새로운 클러스터링과 스케줄링 기법의 성능을 확인하기 위하여 시뮬레이션에서는 통신비용의 변화에 대한 병렬연산시간을 비교하였다.

1. 서 론

다양한 고성능 워크스테이션의 출현으로 워크스테이션 클러스터를 사용한 병렬 연산 기반(platform)의 개발이 가능하게 되었다. 향후 워크스테이션은 하나 이상의 프로세서로 구성된 SMP로 개발될 것이며, SMP 시스템은 SMP 클러스터를 구성하여 사용될 것이다[1]. 다중 프로세서 시스템과 비교할 때 버스 기반의 SMP 환경에서 중요한 한계중의 하나가 프로세서 사이의 통신을 위한 비용이 크다는 것이다. SMP 환경에서 병렬 연산을 위한 효과적인 클러스터링과 스케줄링 기법을 사용함으로서 이러한 문제점을 줄일 수 있다. 버스 기반의 SMP 환경에서 태스크 스케줄링은 완전연결 네트워크에서의 스케줄링과 차이가 있다. 이 환경에서는 네트워크의 통신 자원을 공유할 수 없으므로 서로 독립된 두 개의 통신 태스크가 동시에 수행될 수 없다. 따라서 버스를 기반으로 하는 SMP 환경에서 통신과 동기화 비용은 다중프로세서 시스템보다 상대적으로 매우 크다. 현재까지 다중 프로세서시스템을 위한 다양한 태스크 스케줄링 기법들이 제안되었다. 다중 프로세서 시스템에서 태스크 스케줄링은 휴리스틱을 기반으로 하는 기법 [2]과 태스크 중복(duplication)을 기반으로 하는 기법[3]이 제안되었다.
본 논문에서는 버스 기반의 SMP 환경에서 적용할 수 있는 클러스터링과 스케줄링 알고리즘을 제안하며 통신비용에 따른 병렬연산시간 변화를 조사한다.

본 논문의 2장에서는 태스크 그래프를 설명하고, 태스크 클러스터링과 스케줄링 기법에서 사용할 수식을 정의한다. 3장에서는 SMP에 적용할 수 있는 태스크 클러스터링과 스케줄링 기법을 설명하고, 예제 태스크 그래프에 대한 클러스터링과 스케줄링 결과를 설명한다. 4장에서는 통신비용의 변화에 따른 병렬연산시간의 변화를 측정한 시뮬레이션 결과를 보여준다. 5장에서는 본 논문에 대한 결론을 나타낸다.

본 연구는 한국과학재단 목적기초연구(R05-2000-000-00279-0) 지원으로 수행되었음.

2. 태스크 그래프

병렬처리를 위한 다중 프로세서 시스템에서 응용 프로그램은 태스크 그래프로 나타낼 수 있다. (그림 1)은 태스크 그래프를 DAG(Directed Acyclic Graph)로 나타낸 예제로서, 태스크 i에 대한 노드 번호는 v_i로 나타낸다. 노드 i에서 '|' 다음의 숫자는 태스크 i의 연산비용을 나타낸다.

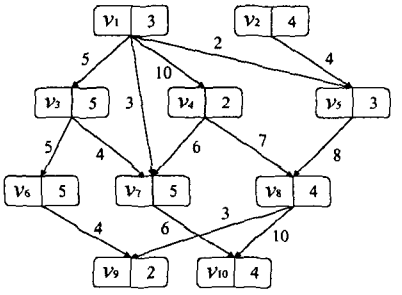

그림 1. 태스크 그래프

태스크 그래프는 V가 태스크 노드이고, E가 통신 링크일 때 튜플(tuple) (V, E, t, c)로 정의할 수 있다. 여기서 집합 t는 연산비용(computation cost)으로 구성되며, 각각의 태스크 $i \in V$는 $t(i)$로 표시하는 연산비용을 갖는다. c는 통신비용(communication cost)의 집합으로 구성되며, 태스크 i에서 태스크 j로 연결되는 링크 $e_{ij} \in E$는 통신비용 c_{ij}를 갖는다.
본 논문에서는 태스크의 클러스터링과 스케줄링을 위하여 다음과 같은 수식들을 사용한다. 위의 수식들을 사용하여 (그림 1)의 태스크 그래프에 대한 값을 구하면 (표 1)과 같다.

$$pred(i)=\{j|e_{ji}\in E\} \quad (1)$$

$$suce(i)=\{j|e_{ij}\in E\} \quad (2)$$

$$est(i)=0 \quad \text{if } pred(i) \text{ is empty} \quad (3)$$

$$est(i)=\min\left\{\max_{j\in pred(i),\ k\in pred(i),\ k\neq j}(ect(j),\ ect(k)+c_{ki})\right\} \quad \text{if } pred(i) \text{ is not empty} \quad (4)$$

$$ect(i)=est(i)+t(i) \quad (5)$$

$$cpt(i)=\{j|(ect(j)+c_{ji}\geq(ect(k)+c_{ki})\} \quad \forall_j\in pred(i);\ k\in pred(i), k\neq j \quad (6)$$

$$ccpt(i)=cpt(cpt(i)) \quad (7)$$

$$lct(i)=ect(i) \quad \text{if } suce(i) \text{ is empty} \quad (8)$$

$$lct(i)=\min\left\{\min_{j\in suce(i),\ i\neq CPT(j)}(lst(j)-c_{ij}),\ \min_{j\in suce(i),\ i=cpt(j)}lst(j)\right\} \quad (9)$$

$$lst(i)=lct(i)-t(i) \quad (10)$$

$$level(i)=t(i) \quad \text{if } suce(i) \text{ is empty} \quad (11)$$

$$level(i)=\max_{k\in suce(i)}(level(k))+t(i) \quad \text{if } suce(i) \text{is not empty} \quad (12)$$

표 1. 태스크 그래프에 대한 파라미터 값

노 드	est	ect	lst	lct	cpt	ccpt	level
1	0	3	0	3	-	-	17
2	0	4	1	5	-	-	15
3	3	8	3	8	1	-	14
4	3	5	3	5	1	-	11
5	5	8	5	8	2	-	11
6	8	13	8	13	3	1	7
7	11	16	11	16	3	1	9
8	12	16	13	17	5	2	8
9	17	19	17	19	8	5	2
10	22	26	22	26	8	5	4

3. 태스크 클러스터링과 스케줄링

3.1 클러스터링과 스케줄링 기법

본 논문에서 제안한 클러스터링 기법의 단계 1에서는 먼저 2장에서 나타낸 수식들을 이용하여 클러스터링에서 사용할 필요한 파라미터 값들을 구한다. 클러스터링의 단계 2에서는 단계 1에서 구한 값들을 이용하여 중복된 태스크로 구성된 태스크 클러스터를 생성한다. 태스크 클러스터 기법은 태스크 그래프에서 레벨이 가장 낮은 노드에서 시작하여 각 노드의 *cpt* 노드를 동일한 클러스터에 할당하면서 진행한다. 노드의 중복 여부는 다음과 같은 휴리스틱에 의하여 결정되는 것으로, 클러스터 생성을 위해 반드시 필요한 노드와 병렬연산시간을 단축할 수 있는 노드를 클러스터에 중복한다.

① *cpt* 노드가 현재 스케줄링하는 노드의 유일한 부모 노드이면 *cpt* 노드를 중복한다.

② 스케줄링하고 있는 결합 노드와 동일한 클러스터에 배정할 *cpt* 노드가 이미 다른 클러스터에 배정된 경우에는 결합 노드의 *ccpt* 노드가 결합 노드와 같은 클러스터에 배정될 수 있는 경우에만 *cpt* 노드를 중복한다.

③ 결합 노드에서 모든 부모 노드가 이미 다른 워크스테이션에 배정된 경우에는 *cpt* 노드를 중복한다.

스케줄링 과정은 클러스터링의 결과를 이용하여 각 태스크의 실제 시작시간과 완료시간을 확정하고, 병렬 처리 시간을 예측하기 위한 스케줄링 길이를 계산한다. 이 단계에서 알고리즘은 버스 기반의 SMP 특성을 고려하여 클러스터에 배정된 태스크를 네트워크 자원의 충돌이 발생되지 않도록 스케줄링한다.
각 클러스터에 저장된 태스크를 레벨이 낮은 것부터 스케줄링하며, 스케줄링 할 태스크의 데이터 선행 관계가 만족되지 않으면 다음 클러스터의 태스크를 스케줄링한다. 스케줄링할 노드의 모든 부모 노드의 스케줄링이 완료되어 데이터 선행 관계가 만족되는 경우에는 네트워크 통신 자원의 충돌이 발생되지 않도록 스케줄링한다.

3.2 동작 예제

태스크 중복을 기반으로 하는 클러스터링과 스케줄링 기법들은 병렬연산시간을 단축하기 위하여 태스크를 여러 개의 프로세서에 중복하여 배정한다. 이들 알고리즘들은 특히 태스크 그래프에서 결합 노드와 그 노드의 스케줄링에 가장 큰 영향을 주는 부모 노드를 동일한 프로세서에 할당하며, 노드의 중복이 가능하다. 버스 기반의 SMP 환경에서는 독립된 태스크의 통신 자원에 대한 동시 사용이 불가능하여 태스크 중복으로 인한 스케줄링 지연이 발생될 수 있다.

(그림 2)는 (그림 1)의 태스크 그래프를 사용하여 본 논문에서 제안한 클러스터링과 스케줄링 기법을 적용하였을 때 스케줄링한 결과를 나타낸 것이다. 태스크 클러스터링 기법을 적용한 결과 4개의 프로세서가 필요하며, 각각의 프로세서에 할당된 태스크들의 구성은 $P_0(v_6, v_3, v_1)$, $P_1(v_9, v_8, v_5, v_2)$, $P_2(v_{10}, v_7, v_3, v_1)$, $P_3(v_4, v_1)$ 와 같다. 또 본 논문에서 버스 기반의 SMP 환경을 가정하므로 제안한 스케줄링 기법을 적용함으로써 (그림 2)에서와 같이 통신자원의 충돌이 발생하지 않도록 태스크를 스케줄링할 수 있다.

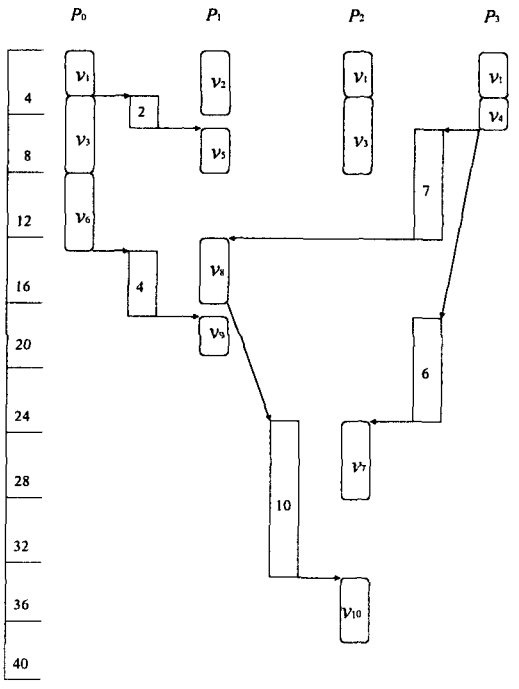

그림 2. 클러스터링과 스케줄링 결과

4. 병렬연산시간의 비교

Darbha[3]와 같이 현재까지 연구된 다중 프로세서 환경에서 태스크 중복을 기반으로 하는 스케줄링 알고리즘들은 완전 연결 통신망을 기본으로 두 개 이상의 태스크가 동시에 통신할 수 있는 구조를 가정하였다. 본 논문에서는 제안한 기법의 성능을 비교하기 위하여 Darbha 알고리즘을 버스기반의 SMP 환경에 맞도록 알고리즘을 수정하여 성능을 비교하였다.

본 논문에서는 *ccr*(communication to computation ratio)의 변화에 따른 병렬연산시간의 변화를 측정하기 위하여 ALPES 프로젝트[4]에서 사용한 4가지 태스크 그래프를 이용하였다. 태스크 그래프에서 태스크의 통신비용의 합한 값을 연산비용의 합한 값으로 나눈 것을 *ccr* 로 정의한다. (그림 3)은 각 태스크 그래프에 대하여 *ccr*의 증가에 따른 병렬연산시간의 변화를 나타낸 것이다. 병렬연산시간 측면에서 본 논문에서 제안한 기법(S3MP)이 Darbha 기법(STDS)보다 성능이 우수함을 알 수 있다.

5. 결 론

본 논문에서는 버스 기반의 SMP 환경에서 태스크 중복이 가능할 때 적용할 수 있는 태스크 클러스터링과 스케줄링 기법을 제안하였다. 클러스터링 단계에서는 휴리스틱을 사용하여 태스크를 프로세서에 할당하고, 스케줄링 단계에서는 네트워크 통신 자원을 우선 배정하여 스케줄링 함으로써 네트워크 충돌을 방지한다. 본 논문에서는 버스 기반의 SMP 환경에서 연산비용에 대한 통신비용의 변화에 따른 병렬연산시간을 측정하여 성능을 비교하였다.

참고 문헌

[1] D.E. Culler et. al., "Parallel Computing on the Berkeley NOW," *9th Joint Symposium. on Parallel Processing*, Japan 1997.

[2] C. L. McCreary, A. A. Khan, J. J. Thompson, and M. E. McArdle, "A Comparison of Heuristics for Scheduling DAGs on Multiprocessor," *Proc. of Eighth Int. Conf. on parallel Processing*, pp. 461-451, 1994.

[3] S. Darbha and D.P. Agrawal, "A Task Duplication Based Scalable Scheduling Algorithm for Distributed Memory Systems," *Journal of Parallel and Distributed Computing*, Vol. 46, pp. 15-26, 1997.

[4] J.P. Kitajma and B. Plateau, "Building Synthetic Parallel Programs: The Project(ALPES)," *Pro. of IFIP WG 10.3 Workshop on Programming Env. for Parallel Computing*, pp. 161-170, 1992.

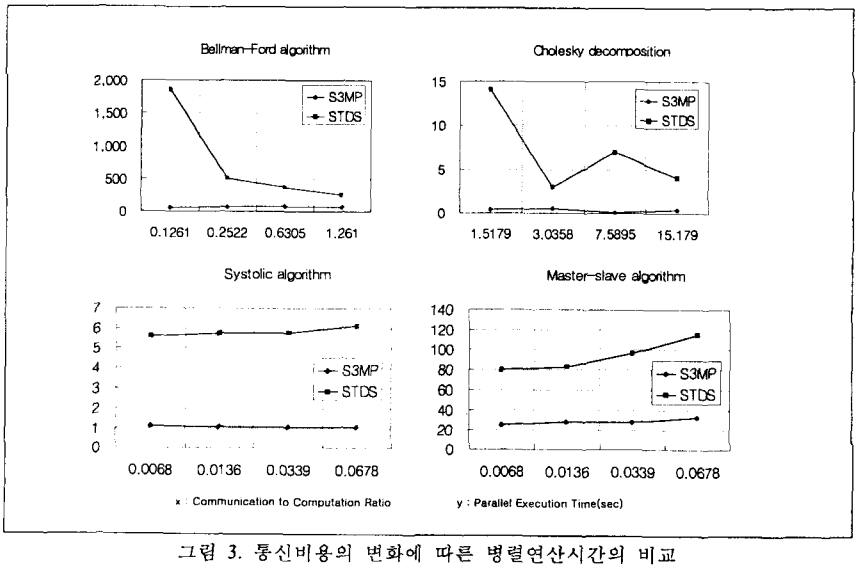

그림 3. 통신비용의 변화에 따른 병렬연산시간의 비교