버스 기반의 워크스테이션 네트워크에서 통신비용의 효과

강오한[0] 김시관 송미경 남경임
안동대학교 컴퓨터교육과, 멀티미디어전공
{ohkang, sgkim99}@andong.ac.kr

The Effect of Communication Cost on Bus-Based Network of Workstations

Oh-Han Kang[0] Si-Gwan Kim Mi-Kyung Song Kyung-Im Nam
Dept. of Computer Education, Dept. of Multimedia Engineering, Andong National University

요 약

워크스테이션 네트워크 (NOW: Network of Workstations)은 고성능의 병렬 연산을 위한 중요하고 효과적인 기반환경을 제공하고 있다. NOW 환경에서 통신과 동기화 비용은 다중프로세서 시스템보다 상대적으로 매우 크다. NOW에서 병렬 태스크와 통신을 위한 스케줄링 기법의 선택은 시스템의 활용도와 성능에 큰 영향을 미치므로 효과적으로 스케줄링 알고리즘에 대한 연구가 필요하다.

본 논문에서는 버스 기반의 NOW에서 병렬 태스크를 위한 스케줄링 알고리즘을 제안하고, 시뮬레이션을 통하여 통신비용의 변화에 따른 시스템의 성능을 비교한다. 알고리즘은 태스크 중복을 기반으로 하며 통신에 따른 스케줄링 길이를 줄이기 위하여 휴리스틱을 사용한다.

1. 서 론

NOW 환경은 경제성과 신축성(flexibility) 측면에서 다중 프로세서보다 우수하여 병렬 연산을 위한 기반으로 개발되고 있다 [1-5]. 다중 프로세서 시스템과 비교할 때 NOW 환경의 중요한 한계중의 하나가 프로세서 사이의 통신을 위한 비용이 크다는 것이다. NOW 환경에서 병렬 연산을 위한 효과적인 태스크 분할과 스케줄링 알고리즘을 사용함으로서 이러한 문제점을 줄일 수 있다. 버스 기반의 NOW 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 NOW 환경에서 통신과 동기화 비용은 다중프로세서 시스템보다 상대적으로 매우 크다. 본 논문에서는 태스크 중복을 기반으로 버스 기반의 NOW 환경에서 적용할 수 있는 스케줄링 알고리즘을 소개하며 통신비용에 따른 스케줄링 길이의 변화를 조사한다. 알고리즘은 태스크 중복을 기반으로 하며 통신에 따른 스케줄링 길이를 줄이기 위하여 휴리스틱을 사용한다. 시뮬레이션에서는 본 논문에서 소개한 알고리즘을 이용하여 통신비용의 변화에 따른 스케줄링 길이의 변화를 측정하고 분석한다.

본 논문의 2장에서는 태스크 그래프 모델과 용어를 정의하고 관련 연구를 소개한다. 3장에서는 NOW 환경에 적합한 태스크 스케줄링 알고리즘을 설명하고, 예제 태스크 그래프에 대한 스케줄링 결과를 설명한다. 4장에서는 통신비용의 변화에 따른 스케줄링 길이의 변화를 측정한 시뮬레이션 결과를 보여준다. 5장에서는 본 논문에 대한 결론을 나타낸다.

2. 용어 정의 및 관련 연구

2.1 용어 정의

병렬처리를 위한 다중 프로세서 시스템에서 응용 프로그램은 태스크 그래프로 나타낼 수 있다.

※ 본 연구는 한국과학재단 목적기초연구(R02-2000-00279) 지원으로 수행되었음.

그림 1은 태스크 그래프 (DAG: Directed Acyclic Graph)의 예제로서, 태스크 i에 대한 노드 번호는 n_i로 나타낸다.

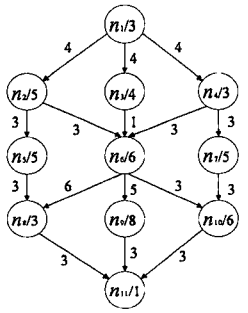

그림 1. 예제 태스크 그래프

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 태스크 i가 실행을 시작할 수 있는 가장 빠른 시간과 실행을 완료할 수 있는 가장 빠른 시간을 각각 $est(i)$ (earliest start time)과 $ect(i)$ (earliest completion time)로 나타낸다. $ect(i)$는 $est(i)$에 태스크 i의 연산비용을 합한 것이다. 태스크 그래프에서 각 노드에 대한 est의 계산은 시작 노드부터 종료 노드까지 하향식(top-down) 방식으로 진행된다. 태스크 i의 모든 부모 노드 j에 대하여 $\{ect(j) + c_{ji}\}$ 값이 최대인 노드를 $cpt(i)$ (critical parent)라고 한다. 태스크 i가 지연되어 완료될 수 있는 가장 늦은 시각과 실행을 시작할 수 있는 가장 늦은 시각을 각각 $lct(i)$ (latest completion time)와 $lst(i)$ (latest start time)로 나타낸다. $lst(i)$는 $ect(i)$에서 태스크 i의 연산비용을 뺀 것이다. 각 노드에 대한 lst의 계산은 종료 노드에서 시작 노드 방향으로

상향식(bottom-up) 방식으로 진행한다. 태스크 그래프에서 노드 i의 레벨(level)은 노드 i에서 종료 노드까지의 연산비용을 합한 값 중에서 가장 큰 값을 나타낸다. 다음은 소개할 스케줄링 알고리즘을 위한 용어들을 정의한다.

정의 1 스케줄링이 완료되었을 때 태스크 i의 확정된 시작 시간과 종료 시간을 각각 $rst(i)$ (real start time)과 $rct(i)$ (real completion time)로 정의한다. $rct(i)$는 $rst(i)$에 태스크 i의 연산비용을 합한 것이다.

정의 2 태스크 i의 $ccpt(i)$ (critical parent of cpt)는 $cpt(i)$의 cpt로 정의한다. 그림 1에서 $cpt(8)$은 n_6 노드이며 $ccpt(8)$은 n_2 노드이다.

정의 3 태스크 그래프에서 태스크의 통신비용의 합한 값을 연산비용의 합한 값으로 나눈 것을 ccr (communication to computation ratio)로 정의한다.

각 노드에 대한 rst와 rct는 스케줄링 길이를 계산하는데 사용되며 스케줄링이 완료된 후 각 태스크의 실제 시작 시간과 종료 시간을 나타낸다. cpt는 결합(join) 노드를 스케줄링할 때 결합 노드와 같은 프로세서에 할당할 부모 노드를 선택하기 위하여 사용하는 것으로, 결합 노드와 cpt 노드를 같은 프로세서에 할당함으로써 병렬 시간을 줄이기 위한 것이다. $ccpt$는 현재 스케줄링하고 있는 노드와 cpt 노드의 중복 여부를 결정하기 위한 휴리스틱에서 사용하는 것이다. 위의 설명과 정의에 따라 그림 1 그래프에 대한 파라미터 값은 표 1과 같다.

표 1. 태스크 그래프에 대한 파라미터 값

노 드	*est*	*ect*	*lst*	*lct*	*cpt*	*ccpt*	*level*
1	0	3	0	3	-	-	23
2	3	8	4	9	1	-	20
3	3	7	4	8	1	-	19
4	3	6	3	6	1	-	18
5	8	13	10	15	2	1	9
6	9	15	9	15	2	1	15
7	6	22	7	12	4	1	12
8	11	18	18	21	6	2	4
9	15	23	16	24	6	2	9
10	15	21	15	21	6	2	7
11	24	25	24	25	9	6	1

2.2 태스크 스케줄링 알고리즘

Darbha[6]의 연구에서와 같이 다중 프로세서 환경에서 태스크 중복을 기반으로 하는 스케줄링 알고리즘들은 완전 연결 통신망을 기본으로 두 개 이상의 태스크가 동시에 통신할 수 있는 구조를 가정하였다. 따라서 다중 프로세서를 기반으로 하는 이들 알고리즘들은 네트워크 통신 자원의 충돌로 독립된 두 개 이상의 태스크가 동시에 통신할 수 없는 버스 기반의 NOW 환경에는 적당하지 않다. 서로 다른 워크스테이션에 할당된 태스크 사이의 통신을 위하여 네트워크 통신 자원을 우선 배정하여 스케줄링 함으로써 네트워크 충돌을 방지한다. 본 논문에서 제안한 알고리즘에서는 중복할 태스크를 선택할 때 다음과 같은 휴리스틱을 사용함으로써 스케줄링 길이를 단축하여 병렬 시간을 줄인다.

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.

위에서 첫 번째 것은 어떤 노드의 부모 노드가 하나만 존재하는 경우로 부모 노드를 포함하여야 클러스터의 생성이 가능하다. 따라서 이 부모 노드가 이미 다른 클러스터에 포함되었더라고 cpt 노드인 이 부모 노드를 현재 스케줄링 하는 노드가 포함된 클러스터에 중복한다.

태스크 그래프에서 결합 노드에 대한 스케줄링이 병렬 시간에 가장 큰 영향을 미친다. 위의 휴리스틱에서 두 번째와 세 번째 내용은 결합 노드와 어떤 부모(parent) 노드를 동일한 워크스테이션에 할당할 것인지를 결정한다. 알고리즘의 중요한 개념은 결합 노드와 스케줄링 길이에 결정적인 영향을 줄 수 있는 cpt 노드를 선택적으로 중복하여 동일한 워크스테이션에 배정함으로써 스케줄링 길이와 병렬 시간을 단축하는 것이다. 태스크 그래프에서 하나의 노드가 두 개 이상의 노드에 대한 cpt 노드로 나타날 수 있다. 이때 cpt 노드를 계속적으로 중복하여 클러스터에 할당하는 것은 스케줄링 길이를 연장할 수 있는 가능성이 있다. 따라서 본 논문에서는 위의 휴리스틱에서 두 번째 방법을 사용하여 cpt 노드를 선택적으로 중복한다. 이 방법은 결합 노드의 cpt 노드 중복을 위하여 cpt 노드의 cpt인 $ccpt$ 노드가 동일한 클러스터에 할당 가능한지를 체크한다. 결합 노드와 $ccpt$ 노드가 동일할 클러스터에 할당될 수 있으면 결합 노드, cpt, $ccpt$ 노드를 하나의 클러스터에 할당한다. 위에서 세 번째 경우는 결합 노드에서 모든 부모 노드가 이미 다른 클러스터에 할당된 상태에서 새로운 클러스터 생성을 위하여 부모 노드 중에서 하나를 선택하는 경우이다. 따라서 결합 노드와 cpt 노드를 동일한 클러스터에 할당함으로써 스케줄링 길이를 단축할 수 있다.

3. 알고리즘 구조

본 논문에서 제안한 스케줄링 알고리즘의 단계 1에서는 2장에서 나타낸 설명과 정의들을 이용하여 알고리즘에서 사용할 필요한 파라미터 값들을 구한다. 이 단계는 다중 프로세서 환경의 스케줄링 알고리즘과 유사하다. 단계 2에서는 단계 1에서 구한 값들을 이용하여 중복된 태스크로 구성된 태스크 클러스터를 생성한다. 태스크 클러스터는 태스크 그래프에서 레벨이 가장 낮은 노드에서 시작하여 각 노드의 cpt 노드를 동일한 클러스터에 할당하면서 진행한다. 각 노드의 중복 여부는 휴리스틱에 의하여 결정되는 것으로, 클러스터 생성을 위해 반드시 필요한 노드와 스케줄링 길이를 줄일 수 있는 노드를 클러스터에 중복한다. 스케줄링하는 현재 노드의 cpt 노드가 이미 다른 클러스터에 할당된 경우에는 정의 2의 $ccpt$ 개념을 적용하여 cpt 노드의 중복을 위한 휴리스틱을 사용한다.

단계 3에서는 단계 2의 결과를 이용하여 태스크의 rst와 rct를 확정하고, 병렬 처리 시간을 예측하기 위한 스케줄링 길이를 계산한다. 이 단계에서 알고리즘은 버스 기반의 NOW 특성을 고려하여 클러스터에 배정된 태스크를 네트워크 자원의 충돌이 발생되지 않도록 스케줄링한다. 네트워크 통신 자원을 독립된 두 개의 통신 태스크가 동시에 사용할 때 충돌이 발생되므로 스케줄링 알고리즘에서 이를 방지하도록 통신 자원을 할당한다. 이를 위하여 알고리즘에서는 슬롯(slot) 개념을 사용한다. 두 개의 태스크가 통신하는 시간 동안을 하나의 슬롯으로 보며, 프로그램이 시작되기 전에 모든 슬롯은 비어있다고 가정한다. 알고리즘에서는 태스크가 통신하기 전에 통신비용에 해당하는 길이의 사용하지 않는 첫 번째 슬롯을 태스크에 배정한다. 각 클러스터에 저장된 태스크를 레벨이 낮은 것부터 스케줄링하며, 스케줄링 할 태스크의 데이터 선행 관계가 만족되지 않으면 다음 클러스터의 태스크를 스케줄링한다. 스케줄링 할 노드의 모든 부모 노드의 스케줄링이 완료되어 데이터 선행 관계가 만족되는 경우에는 네트워크 통신 자원의 충돌이 발생

되지 않도록 스케줄링한다. 그림 2는 그림 1의 태스크 그래프를 사용하여 본 논문에서 제안한 알고리즘을 적용하여 스케줄링한 결과를 나타낸 것이다.

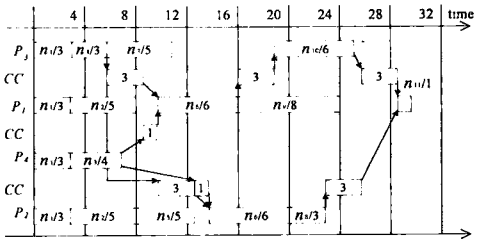

그림 2. 알고리즘의 스케줄링 결과

그림 2에서 P_i는 워크스테이션을 나타내며 *NCR*은 네트워크 통신 자원의 사용을 나타낸다. 그림에서 첫 번째 워크스테이션인 P_1에는 n_{11}, n_9, n_6, n_2, n_1 노드들이 포함된 첫 번째 태스크 클러스터가 할당되어 있으며, 각 노드는 1, 8, 6, 5, 3의 연산비용을 갖는 것을 나타낸다. 따라서 본 논문에서 제안한 알고리즘을 적용하면 스케줄링 길이는 30이며, 알고리즘에서 필요로 하는 워크스테이션의 수는 4이다.

4. 통신 효과 분석

본 논문에서는 *ccr*의 변화에 따른 스케줄링 길이의 변화를 측정하기 위하여 임의의 태스크 그래프와 실제 응용 프로그램의 태스크 그래프를 사용하였다. 시뮬레이션에서 사용한 임의의 태스크 그래프는 40개의 노드로 구성되었으며 *ccr*의 값을 증가시키면서 스케줄링 길이의 변화를 측정하였다. 그림 3은 *ccr*의 증가에 따른 스케줄링 길이의 변화를 나타낸 것이다.

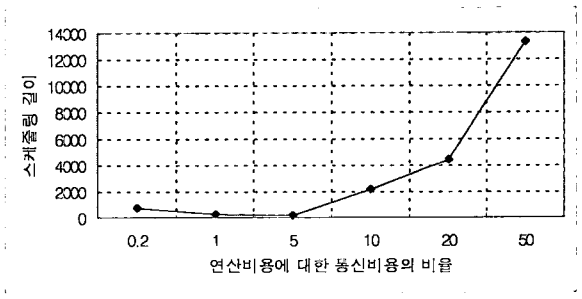

그림 3. *ccr*에 따른 스케줄링 길이의 변화

그림 3에서 *ccr*의 값이 증가하면 스케줄링의 길이가 증가함을 확인할 수 있으며, *ccr* 값의 증가율이 커지면 스케줄링 길이도 급격히 증가함을 알 수 있다. 따라서 NOW 환경에서 통신량의 증가는 병렬 프로그램을 처리하는 시스템의 성능에 큰 영향을 준다고 할 수 있다.

두 번째 시뮬레이션에서는 2개의 실제 응용 프로그램인 Systolic과 Master-Slave 알고리즘의 태스크 그래프를 사용하여 *ccr*의 변화에 따른 스케줄링 길이를 측정하였다. 각 응용 프로그램의 태스크 그래프에서 태스크의 수는 2,502와 2,262개이며, 링크의 수는 4,902와 6,711개이다. 이들 태스크 그래프의 초기 *ccr* 값은 각각 0.068과 0.076이며, 그림 4는 *ccr*의 변화에 따른 스케줄링 길이의 변화를 나타낸 것이다.

그림 4에서 *ccr*의 값이 매우 적은 경우에는 스케줄링의 길이가 늘어나는 것을 알 수 있다. 이것은 태스크 중복을 기본으로 하는 스케줄링 알고리즘의 공통적인 특성이다. 그림 4에서 확인할 수 있는 중요한 내용은 *ccr*이 일정한 값 이상으로 증가하면 스케줄링의 길이가 급격히 늘어난다는 것이다. 병렬 응용 프로그램을 NOW 환경에서 실행할 때 가장 심각한 문제중의 하나가 통신 지연에 관한 것이다.

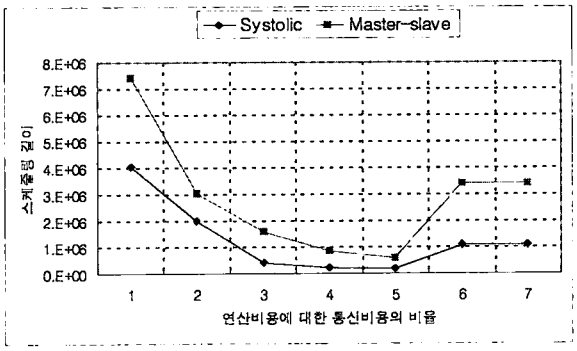

그림 4. *ccr*에 따른 스케줄링 길이의 변화

5. 결 론

본 논문에서는 버스 기반의 NOW 환경에서 적용할 수 있는 태스크 스케줄링 알고리즘을 제안하였다. 알고리즘에서는 서로 다른 워크스테이션에 할당된 태스크 사이의 통신을 위하여 네트워크 통신 자원을 우선 배정하여 스케줄링 함으로써 네트워크 충돌을 방지한다. 또한 본 논문에서는 제안된 알고리즘을 사용하여 NOW 환경에서 통신비용의 변화에 따른 스케줄링 길이의 변화를 조사하였다.

참고 문헌

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

[2] Y. Dong, X. Du, and X. Zhang, "Characterizing and Scheduling Communication Interactions of Parallel and Local Jobs on Networks of Workstations," *Computer Comm.s*, Vol. 21, No. 5, pp. 170-184, 1998.

[3] X. Du and X. Zhang, "Coordinating Parallel Processes on Networks of Workstations," *Journal of Parallel and Distributed Computing*, Vol. 46, pp. 125-135, 1997.

[4] X. Du, Y. Dong, and X. Zhang, "Effectively Scheduling Parallel Tasks and Communications on Networks of Workstations," *Proc. of Euro-Par'97*, 1997.

[5] S. Nagar, A. Banerjee, A. Siva., and C.R. Das, "An Experimental Study of Scheduling Strategies for a Networks of Workstations," *Technical Report CSE-98-009*, Jul. 1997.

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