태스크 스케줄링 기법의 실험적 구현

강오한° 강상성 김시관
안동대학교 컴퓨터교육과, 금오공과대학교 컴퓨터공학부
{ohkang°, edukang}@andong.ac.kr, sgkim@se.kumoh.ac.kr

Experimental Implementation of Task Scheduling Strategies

Ohhan Kang° Sangsung Kang Sigwan Kim
Dept. of Computer Education, Andong National University

요 약

본 논문에서는 태스크 스케줄링 기법을 PC 클러스터에 구현하여 스케줄링 기법의 성능을 분석한다. 구현된 스케줄링 기법은 태스크 그래프를 입력으로 받아 PC 클러스터로 스케줄링하며, 휴리스틱을 사용하여 태스크를 선택적으로 중복함으로써 병렬연산시간을 단축한다. 실험을 한 결과 본 논문에서 소개한 스케줄링 기법이 비교 기법보다 병렬연산시간 측면에서 성능이 우수함이 확인하였다.

1. 서 론

클러스터 시스템은 여러 대의 PC나 워크스테이션을 고속 네트워크로 연결하여 고성능 서버의 성능을 발휘하는 시스템으로 범용성, 확장성, 가격에 대한 성능의 우수성으로 인하여 병렬 연산을 위한 효과적인 시스템으로 정착되고 있다[1,2]. 클러스터의 성능 저하에 가장 큰 영향을 미치는 요인중의 하나가 프로세서 사이의 통신지연이다. 현재까지 클러스터 환경에서 병렬 연산을 위한 다양한 네트워크 유형(topology)이 사용되지만 구조가 간단한 버스(bus) 구조가 광범위하게 사용되고 있다. 그러나 버스 구조는 네트워크 통신자원을 공유할 수 없으므로 다른 구조와 비교할 때 중요한 한계중의 하나가 통신자원 사용을 위한 비용(cost)이 크다는 것이다.

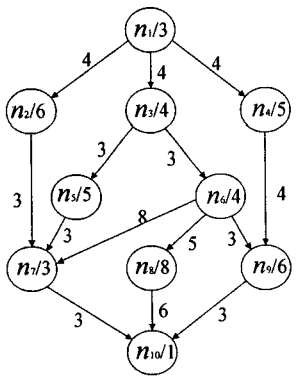
클러스터에서 프로세서의 활용도를 높이고 성능을 향상시키기 위한 태스크 스케줄링 기법과 프로세서간의 통신지연 단축에 관한 연구가 최근에 활발히 추진되고 있다. 현재까지 병렬연산을 위한 다중프로세서나 클러스터 시스템을 위한 다양한 스케줄링 기법 기법들이 제안되어 성능이 평가되었으나 대부분이 시뮬레이션에 의한 비교였다. 클러스터의 보급이 확산되면서 최근에 태스크 스케줄링 기법을 실제 시스템에 구현하여 성능을 비교한 연구들이 발표되고 있다[3].

본 논문에서는 태스크 중복을 기반으로 버스 구조의 클러스터에 적용할 수 있는 휴리스틱 스케줄링 알고리즘을 실제 클러스터 시스템에 구현하여 성능을 비교한다. PC 클러스터에 구현된 스케줄링 기법은 중복할 태스크를 선택할 때 휴리스틱을 사용하여 병렬 시간을 단축할 수 있는 태스크들을 중복시킨다. 스케줄링 기법이 구현된 PC 클러스터는 리눅스가 설치된 6대의 PC와 Gigabit Ethernet을 사용하며, 메시지 교환을 위해 표준화된 메시지 전달방식 통신 라이브러리인 MPI를 사용한다.

본 논문의 2장에서는 논문에서 구현한 스케줄링 기법을 설명한다. 3장에서는 클러스터 구축과 스케줄링 기법의 구현 방법을 설명한다. 4장스케줄링 기법의 성능을 비교하고, 5장에서는 본 논문에 대한 결론을 나타낸다.

2. 태스크 스케줄링 기법

클러스터 시스템에서 태스크 스케줄링의 주된 목적은 태스크를 서로 다른 프로세서에 할당함으로써 응용 프로그램의 병렬연산시간을 단축할 수 있도록 스케줄링 길이를 줄이는 것이다. 이러한 응용 프로그램은 태스크 스케줄링 기법의 입력으로 사용되는 태스크 그래프로 나타낼 수 있다. (그림 1)은 태스크 그래프의 예를 나타낸 것이다.



(그림 1) 태스크 그래프

본 논문에서 소개하고 구현할 스케줄링 기법은 저자가 참고문헌 [4]에서 제안한 스케줄링 기법으로, 본 논문에서는 이를 HTDS(Heuristic Task Duplicaton

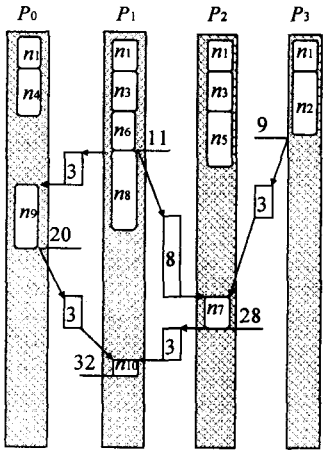
Scheduling) 기법이라고 한다. HTDS 기법은 태스크 그래프를 입력으로 받아서 클래스를 생성하는 부분과 생성된 클래스를 프로세서에 할당하여 각각의 태스크를 스케줄링하는 부분으로 구성된다. 본 논문에서는 HTDS 기법의 성능을 비교하기 위하여 다중프로세서 환경에서 성능이 우수하다고 알려진 태스크 중복기반의 STDS[5] 스케줄링 기법을 버스 구조에 맞도록 수정하였다. 본 논문에서는 이 수정된 스케줄링 기법을 MSTDS(Modified STDS)라 한다.

본 논문에서는 태스크 그래프로 표현된 응용 프로그램의 스케줄링 길이를 구하기 위해 참고문헌 [4]에서 정의한 수식들을 사용한다. <표 1>은 (그림 1)의 태스크 그래프에 대하여 수식들의 값을 구한 것이다.

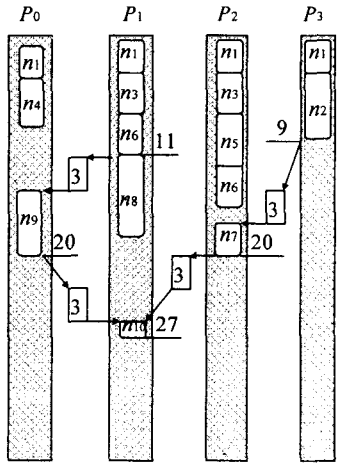
<표 1> (그림 1)의 태스크 그래프에 대한 수식의 값

노 드	*EST*	*ECT*	*LST*	*LCT*	*CPT*	*CCPT*	*level*
(a)	(a)	(a)	(a)	(a)	(a)	(a)	(a)
n_1	0	3	0	3	-	-	20
n_2	3	9	6	12	1	-	10
n_3	3	7	3	7	1	-	17
n_4	3	8	3	8	1	-	12
n_5	7	12	7	12	3	1	9
n_6	7	11	8	12	3	1	13
n_7	15	18	15	18	6	3	4
n_8	11	19	13	21	6	3	9
n_9	12	18	12	18	6	3	7
n_{10}	21	22	21	22	8	6	1

(그림 2)는 (그림 1)의 태스크 그래프에 스케줄링 알고리즘을 적용한 결과를 나타낸 것이다. (그림 2)의 결과를 보면 (그림 1) 태스크 그래프에 MSTDS 기법과 HTDS 기법을 적용한 경우 스케줄링 길이가 각각 32와 27이 된다.



(a) MSTDS 기법을 적용한 경우



(b) HTDS 기법을 적용한 경우

(그림 2) (그림 1)을 스케줄링한 결과

3. PC 클러스터 구축과 스케줄링 기법의 구현

본 논문에서는 PC 클러스터를 구축하고 태스크 스케줄링 기법을 구현하여 성능을 비교하였다. 리눅스(Linux) OS가 설치된 클러스터 PC는 기가비트 이더넷(Gigabit Ethernet)으로 연결되었으며, 메시지 전송을 위하여 병렬 프로그래밍 도구인 MPI를 사용한다. 본 논문에서 구축한 클러스터는 하나의 마스터 PC와 5대의 슬레이브 PC로 구성하였다. 클러스터 구축에 사용된 PC 사양은 Pentium IV CPU, 256M-512M 메모리, 1Gbps NIC로 구성되어 있다. 클러스터 구축에 필요한 소프트웨어는 시스템의 기본 운영환경을 제공하는 운영체제, 각 노드에 IP주소와 커널을 제공하기 위한 서비스 프로그램, 마스터 노드의 디스크를 공유하기 위한 파일시스템, 각 노드들의 계정 정보를 공유하기 위한 서비스 프로그램, 병렬프로그래밍을 위한 API 등으로 구성된다.

본 논문에서는 클러스터 구축에 폭넓게 사용되고 많은 사용자와 지원프로그램을 확보하고 있는 레드햇 리눅스(Redhat Linux)를 PC의 OS로 사용하였다. 슬레이브 노드가 마스터 노드로부터 커널 이미지를 전송받기 위해 파일전송 서비스가 필요한데, 사용자 인증과정 없이 파일전송 서비스를 제공할 수 있는 서비스인 TFTP(Trivial File Transfer Protocol)를 사용하였다. 본 논문에서 구축한 클러스터는 각 노드들간 메시지 및 파일전송, 파일시스템 공유를 위해 TCP/IP 프로토콜을 기반으로 이루어진다. 따라서 클러스터에 참여하는 모든 노드에게 TFTP를 이용해 커널 이미지를 다운로드 받기 전에 TCP/IP를 사용하기 위한 정보가 주어져야 한다. DHCP(Dynamic Host Configuration Protocol)는 마스터 노드에서 슬레이브 노드에게 IP주소를 비롯한 TCP/IP 환경 정보를 전달하기 위한 서비스이다. 이러한 정보의 전달은 슬레이브 노드가 구동한 직후에 이루어진다. 클러스터에 참여하는 노드들 중 마스터 노드는 로컬 디스크에 운영체제

가 설치되지만 하나의 마스터 노드를 제외한 나머지 슬레이브 노드들은 디스크가 없으며 모든 정보는 마스터노드의 디스크를 공유하여 사용하였다. 이를 위해 네트워크로 연결된 호스트의 파일시스템을 로컬 파일시스템처럼 사용할 수 있는 NFS(Network File System)이 필요하다.

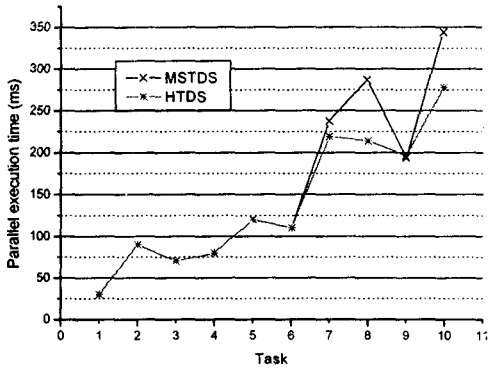
일반적인 응용프로그램은 하나의 CPU 혹은 하나의 호스트에서 실행되도록 개발된다. 클러스터와 같은 병렬 시스템에서 이러한 응용프로그램을 실행한다면 클러스터 규모와는 관계없이 하나의 시스템만 동작하게 된다. 여러 시스템이 협력하여 작업을 처리하도록 하려면 응용프로그램이 병렬처리를 지원하도록 개발되어야하며 병렬프로그래밍을 지원해주는 라이브러리가 필요하다. 병렬프로그래밍 라이브러리는 MPI(Message Passing Interface), PVM(Parallel Virtual Machine) 등이 있으며, MPI는 본 논문에서 구축하는 클러스터와 같이 동일한 플랫폼으로 구성된 클러스터 시스템에 적합하다.

본 논문에서 클러스터 구축을 위하여 커널 환경설정, DHCP 설정, TFTP 설정, NSF 설정, MPI 설정의 작업을 수행하였다. 병렬 프로그래밍 도구인 MPI 설정을 위하여 MPI 설치, 사용자 추가, rlogin과 rsh 설정, rhosts 생성, lam-bhost와 lamhosts 생성의 작업을 수행하였다.

본 논문에서 DAG의 연산비용을 위한 구현은 연산비용에서 제시한 시간(ms)만큼을 지연시켜 실제 작업을 처리하기 위해 시간을 소요하는 것으로 구현하였다. DAG에서 통신비용은 메시지의 크기와 통신의 대역폭에 의해 결정될 수 있다. 정확한 통신비용을 구현하기 위해 먼저 각 노드간의 전송속도를 측정하고 통신비용을 곱하여 실제 전송할 데이터의 크기를 구하게 된다. 속도 측정 프로그램은 두 개의 노드를 지정하면 512Kbyte를 전송한 시간을 계산하고 단위시간(1ms)당 전송량을 계산한다. 프로그램을 사용하여 본 논문에서 구축한 클러스터의 노드간 평균 전송속도를 측정한 결과 약 45Kbytes/ms 정도로 나타났다. 프로그램의 실행시간이나 메시지 전달에 필요한 오버헤드의 영향을 줄이기 위하여 비용의 시간단위를 10ms로 하였다.

4. 성능 평가

본 논문에서는 스케줄링 기법의 성능을 평가하기 위하여 HTDS 기법과 MSTDS 기법을 PC 클러스터에 구현하였다. (그림 3)은 (그림 1)의 태스크 그래프를 실제 PC 클러스터에서 수행시켜 스케줄링되는 과정을 병렬연산시간 측면에서 나타낸 것이다. (그림 3)에서는 본 과제에서 개발한 HTDS 기법이 MSTDS 기법보다 병렬연산시간이 짧아서 HTDS 스케줄링 기법의 성능이 우수함을 보여준다. (그림 1) 태스크 그래프의 병렬연산시간은 태스크 $10(n_{10})$이 종료되는 시간을 비교하면 된다. 응용 프로그램의 실행이 종료되는 태스크 10을 기준으로 HTDS가 MSTDS보다 병렬연산시간 측면에서 16.5%의 성능이 향상되었음을 확인할 수 있다. 이와 같은 결과는 본 과제에서 소개한 스케줄링 기법은 클러스터의 통신특성을 파악하여 통신비용을 줄일 수 있도록 효과적인 휴리스틱을 사용하였기 때문이다.



(그림 3) 병렬연산시간의 비교

5. 결론

본 논문에서는 버스 구조의 클러스터 환경에서 적용할 수 있는 태스크 스케줄링 기법을 소개하고 PC 클러스터에 구현하여 성능을 분석하였다. 구현된 스케줄링 기법은 태스크 중복을 기반으로 하며, 스케줄링 길이를 줄이기 위한 휴리스틱을 사용하여 태스크를 선택적으로 중복한다. 스케줄링 기법을 구현하기 위하여 기가비트 이더넷(Ethernet)으로 연결되고 리눅스와 MPI 환경으로 동작하는 PC 클러스터를 구축하였다. 클러스터에서 스케줄링 알고리즘의 성능을 비교한 결과 본 논문에서 소개한 HTDS 기법이 기존 알고리즘인 MSTDS 기법보다 성능이 향상되었음을 보여주었다.

참고 문헌

1. 특집 클러스터 컴퓨팅, 정보과학회지, 제18권 제3호, 2000.
2. D.E. Culler, et al., A. Mainwaring, R. Martin, C. Yoshikawa, and F. Wong, "Parallel Computing on the Berkeley NOW," Joint Symp. Parallel Processing, 1997.
3. O. Sinnen and L. Sousa, "Experimental Evaluation of Task Scheduling Accuracy: Implications for the Scheduling Model," IEICE Trans. on Information and Systems, Vol. E86-D, No. 9, pp. 1620-1627, 2003.
4. 강오한, 김시관, "버스 기반의 대칭형 다중프로세서 시스템을 위한 태스크 스케줄링 기법," 정보처리학회 논문지, 제9-A권 제4호, pp. 511-518, 2002.
5. 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, 1997, pp. 15-26.