Journal of The Korea Society of Computer and Information
Vol. 19, No. 6, June 2014
www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2014.19.6.101

다중 서버를 사용하는 병렬 머신 스케줄링을 위한 효율적인 알고리즘

정 균 락*

An efficient algorithm for scheduling parallel machines with multiple servers

Kyun-Rak Chong*

요 약

병렬 머신 스케줄링은 주어진 작업들의 총 완료 시간이 최소가 되도록 작업들을 병렬 머신들에 할당하는 문제로 강철 산업, 반도체 제조, 플라스틱 산업 등 다양한 제조 시스템 분야에서 활용되고 있다. 각 작업들은 준비 과정과 처리 과정을 거치게 되는데, 응용 분야에 따라 제거 과정이 필요한 경우도 있다. 이 중 처리 과정은 병렬 머신만 사용되는데 비해, 준비 과정이나 제거 과정은 서버와 병렬 머신이 동시에 사용된다. 기존의 연구들은 단일 서버를 사용하거나 준비 과정과 처리 과정만을 고려하는 연구가 대부분인데, 단일 서버를 사용하는 경우에는 서버에 병목 현상이 발생하게 되어 총 완료 시간이 늦어지게 되고, 병렬 머신의 수를 증가시키더라도 총 완료 시간은 별로 향상되지 않는 단점을 가지게 된다. 본 연구에서는 다중 서버를 사용하고 준비 과정, 처리 과정, 제거 과정을 모두 고려하는 병렬 머신 스케줄링 알고리즘을 제안하고, 서버의 수와 병렬 머신의 수가 총 완료 시간에 어떤 영향을 미치는지 실험을 통해 분석하였다.

▸ Keywords : 병렬 머신 스케줄링, 총 완료 시간, NP-하드

Abstract

The parallel machine scheduling is to schedule each job to exactly one parallel machine so that the total completion time is minimized. It is used in various manufacturing system areas such as steel industries, semiconductor manufacturing and plastic industries. Each job has a setup phase and a processing phase. A removal phase is needed in some application areas. A processing phase is performed by a parallel machine alone while a setup phase and a removal phase are performed by both a server and a parallel machine simultaneously. Most of previous researches used a single

•제1저자 : 정균락 •교신저자 : 정균락
•투고일 : 2014. 4. 22, 심사일 : 2014. 5. 12, 게재확정일 : 2014. 5. 27.
* 홍익대학교 컴퓨터공학과 (Dept. of Computer Engineering, Honglk University)
※ 이 논문은 2013학년도 홍익대학교 학술연구진흥비에 의하여 지원되었음

server and considered only a setup phase and a processing phase. If a single server is used for scheduling, the bottleneck in the server increases the total completion time. Even though the number of parallel machines is increased, the total completion time is not reduced significantly. In this paper, we have proposed an efficient algorithm for the parallel machine scheduling using multiple servers and considering setup, processing and removal phases. We also have investigated experimentally how the number of servers and the number of parallel machines affect the total completion time.

▸ Keywords : parallel machine scheduling, the total completion time, NP-hard

I. 서 론

병렬 머신 스케줄링은 여러 대의 병렬 머신(parallel machine)들이 있을 때 작업(job)들의 총 완료 시간(total completion time, makespan)이 최소가 되게 작업들을 병렬 머신에 할당하는 문제인데, 강철 산업, 반도체 제조, PCB 제조, 페인팅과 플라스틱 공정, 사출 성형 공정 등 매우 다양한 제조 시스템 분야에서 사용되고 있다[1].

병렬 머신 스케줄링은 목적 함수, 병렬 머신의 특성, 작업의 특성, 일반적인 경우와 특수한 경우 등에 따라 매우 다양한 문제들이 연구되었다. 목적 함수로는 총 완료 시간이 주로 사용되었는데 이 외에도, 총 가중 완료 시간, 총 지연 시간, 총 가중 지연 시간 등이 사용되었다. 병렬 머신 특성에 따른 분류를 보면 작업 J_i의 처리(processing) 요구를 d_i라 하고 머신 M_j가 작업 J_i를 처리할 때의 속도를 v_{ij}라 하면 작업 J_i가 머신 M_j에서 처리되는 시간은 d_i/v_{ij}가 된다. 모든 i, j에 대해 $v_{ij}=1$이면 동일(identical) 머신이라 하고, 모든 i에 대해 $v_{ij}=v_j$이면 균등(uniform) 머신이라 하고, v_{ij}가 임의이면 무관(unrelated)하다고 한다[2]. 각 작업들은 준비(setup) 과정, 처리 과정, 제거(removal) 과정이 있고 작업의 특성으로는 작업들이 독립적으로 처리될 수 있는 지 아니면 작업 간에 우선순위가 있는 지 작업이 시작되면 완료될 때까지 중지할 수 없는 지 중지 했다가 나중에 다시 처리할 수 있는 지 등이 있다[1].

기존의 연구들은 대부분 단일 서버를 사용하고, 각 작업들의 준비 시간과 처리 시간만을 고려하였다[3][4][5]. 단일 서버를 사용하고 준비 시간, 처리 시간, 제거 시간을 모두 고려하는 병렬 머신 스케줄링이 [6]에서 연구되었는데 이 문제가 NP-하드임이 증명되었다. 또 실제적인 응용 예로써 강철 코일(steel coil)의 질을 개선하기 위해 일괄적으로 처리되는 가열냉각(annealing) 공정에 대해 기술하였다. 고정된 받침대에 강철 코일들이 놓여 있는데 먼저 이동식 기중기가 강철 코일을 사용 가능한 용광로에 로드한다. 그러면 바로 용광로 안에서 가열이 시작되고 가열이 끝나면 이동식 기중기가 강철 코일을 용광로에서 언로드 한다. 여기서 이동식 기중기는 서버, 용광로는 병렬 머신, 강철 코일들은 작업으로 생각할 수 있고, 그러면 셋업은 이동식 기중기가 코일을 용광로에 로딩하는 작업, 처리는 용광로가 가열하는 과정, 제거는 이동식 기중기가 코일을 용광로에서 언로딩하는 작업이 된다.

단일 서버를 사용하는 병렬 머신 스케줄링에서 서버는 준비와 제거 과정에 반드시 사용되어야 하므로 병렬 머신의 수가 많아지면 서버에 병목 현상이 발생하여 총 완료 시간이 늦어지게 된다. 또 병렬 머신의 수를 증가 시키더라도 총 완료 시간이 별로 개선되지 않는 문제점을 가지고 있다.

최근에 다중 서버를 사용하는 연구들도 발표되었는데 준비 과정과 처리 과정을 고려하거나 처리 과정과 언로딩(제거)만을 고려하였다[7][8].

본 연구에서는 다중 서버와 동일 머신들을 사용하고, 준비 시간, 처리 시간과 제거 시간을 모두 고려한 병렬 머신 스케줄링 문제를 위한 효율적인 알고리즘을 제안하였다. 또 서버 부하를 고려하여 다양한 데이터를 생성하고 서버의 수와 병렬 머신의 수를 증가시킬 때마다 각 병렬 머신에서 작업들이 완료되는 최종 시간이 어떻게 단축되는지 실험을 통해 분석하였다.

논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대해 살펴보고, 3장에서는 문제를 정의하고 제안된 알고리즘에 대해 기술한다. 4장에서는 실험 결과에 대해 분석하고, 5장에서

결론을 맺는다.

II. 관련 연구

병렬 머신 스케줄링 문제 오래전부터 활발하게 연구되었는데 처음에는 각 작업의 처리 시간만을 고려하다가, 준비 시간을 같이 고려하는 연구가 진행되었고 최근에는 제거 시간도 고려하는 연구가 진행되었다. 또한 서버의 수도 단일 서버에서 다중 서버를 사용하는 연구가 진행되고 있다.

다중 서버를 사용하는 병렬 머신 스케줄링 문제를 보면 [7]에서는 언로딩 서버를 사용하는 연구가 진행되었다. 이 연구에서는 여러 개의 서버를 사용하는데 작업의 준비 시간은 고려하지 않고, 처리 시간과 제거 시간만을 고려하는데 제거 시간은 모든 작업이 같다고 가정하였다. [8]에서는 각 작업의 준비 시간과 처리 시간을 고려하고 다중 서버를 사용하는 연구가 진행되었다. 고정된 수의 병렬 머신과 서버가 주어졌을 때 총 완료 시간을 최소로 하는 스케줄링 문제가 NP-하드임을 보였다. 특수한 경우로 모든 작업의 처리 시간과 준비 시간이 같을 때 문제 해결을 위한 다항 시간 알고리즘을 제안하였고, 모든 작업의 준비 시간이 1이고 병렬 머신의 수가 m개, 서버의 수가 m-1 개일 때는 의사 다항(pseudo polynomial) 시간에 해결됨을 보였다.

단일 서버를 사용하는 병렬 머신 스케줄링 문제는 많은 연구가 진행되었는데 대표적인 연구를 보면 [3]에서는 단일 서버를 사용하고 병렬 머신의 수가 2이고, 준비 시간과 처리 시간을 고려할 때 작업 사이의 머신 유휴 시간의 합을 최소로 하는 스케줄링 문제가 연구되었는데 이 문제가 강한 NP-하드임을 증명하였고 휴리스틱 알고리즘을 제안하였다. [9]에서는 작업의 준비 시간이 모두 1일 때, 스케줄링 문제가 NP-하드임을 보였고 의사 다항 시간 알고리즘을 제안하였다. 병렬 머신의 수가 임의일 때는 강한 NP-하드임을 보였다. [4]에서는 처리 시간이 모두 1일 때 다항 시간 알고리즘을 제안하였고, 준비 시간이 상수인 경우에는 NP-하드임을 증명하였다. [10]에서는 처리 시간이 상수일 때는 NP-하드임을 보였다.

[5]에서는 준비 시간과 처리 시간을 고려하여 총 완료 시간을 최소로 하는 병렬 머신 스케줄링 문제가 강한 NP-하드임을 보이고 이 문제 해결을 위한 정수계획법을 제안하였다. 또 특수한 경우로 처리 시간이 짧은 경우(모든 작업에 대해 처리 시간이 준비 시간보다 작거나 같을 때)와 작업들의 길이(준비 시간과 처리 시간의 합)이 모두 같을 경우를 위한 다항 시간 최적 알고리즘을 제안하였고 일반적인 경우를 위한 휴리스틱도 제안하였다. [11]에서는 처리 시간이 모두 같고 준비 시간이 모두 처리 시간 보다 작거나 같은 경우와 준비 시간이 모두 같고 각 작업의 처리 시간이 다른 작업의 준비 시간과 처리 시간의 합보다 작거나 같은 경우도 NP-완비임을 보이고 휴리스틱들을 제안하였다. [12]에서는 준비 시간과 제거 시간이 모두 1이고 각 작업의 처리가 끝나면 바로 제거되는 경우를 위한 LS와 LPT 휴리스틱을 제안하고 최악의 경우 성능 비율이 12/7과 4/3임을 보였다.

단일 서버를 사용하면서 여러 대의 병렬 머신을 사용하는 스케줄링 문제에 대한 연구를 보면 [13]에서는 일반적인 경우에 대해 휴리스틱들을 제안하고 실험적으로 분석하였다. [14]에서는 혼합 정수 프로그래밍을 사용하고 이를 해결하기 위해, 분기와 평가(branch and price) 방법과 그리디 방법을 사용하는 휴리스틱을 제안하고 실험적으로 분석하였다. [6]에서는 준비 시간, 처리 시간, 제거 시간을 모두 고려한 스케줄링 문제에서 LPT 휴리스틱을 사용하면 최악의 경우 성능 비율이 3/2-1/2m임을 보였다. 이 연구에서는 준비가 끝나면 작업의 처리가 바로 시작되지만, 작업의 처리가 끝나면 제거는 바로 할 필요가 없고 나중에 서버가 한가할 때 실행할 수 있다고 가정하였다. [15]에서는 [6]에서와 같은 조건에서 각 작업들을 m 개의 부분집합으로 분할하고 각 분분집합마다 한 개의 병렬 머신을 배정할 경우 최악의 경우 성능 비율이 2인 다항 시간 근사 알고리즘을 제안하였다.

그 밖에 관련된 연구들을 보면 [16]에서는 각 작업들이 우선순위를 가질 때 각 작업의 밀도를 우선순위를 작업의 처리 시간으로 나눈 것으로 정의하고, 멀티프로세서 서버 스케줄링 문제를 해결하는 데 밀도가 가장 높은 작업부터 선택하는 방법이 실행시간이 가장 짧은 작업부터 선택하는 방법이나 잔여 시간이 가장 짧은 작업부터 선택하는 방법보다 더 효과적임을 보였다. 온라인 환경에서 스케줄링 문제도 연구되었는데 각 작업들이 할당될 수 있는 병렬 머신들의 집합이 주어졌을 때 온라인 스케줄링 알고리즘들에 대한 비교 조사가 [2]에 정리 되어 있다. 단일 서버와 2대의 병렬 머신을 사용하면서 준비 시간과 처리 시간외에 해제 시간(release time)을 고려하는 온라인 스케줄링 문제가 [17]에서 연구되었으며 온라인 LPT 알고리즘을 사용할 때 최악의 경우 성능 비율이 2이고, 준비 시간이 모두 1인 경우에는 성능 비율이 1.5임을 보였다.

각 작업의 처리 속도가 병렬 머신마다 일정하지 않을 때 총 완료 시간을 최소로 하는 스케줄링 문제가 [18]에서 연구되었고, 총 지연(tardiness) 시간을 최소로 하는 문제를 해결하기 위해 타부 탐색이 [1]에서 제안되었고, [19]에서는 총 가중 지연 시간을 최소로 하는 문제가 연구되었다.

III. 문제 정의와 알고리즘

병렬 머신 스케줄링 문제는 다음과 같이 정의 된다. n 개의 작업 J_1, J_2, …, J_n이 있을 때 작업 J_i의 준비 시간은 s_i이며, 처리 시간은 p_i이고, 제거 시간은 r_i이다. 서버는 이 작업들을 병렬 머신에 할당해서 처리하게 된다. 준비 과정과 제거 과정은 서버와 해당 병렬 머신이 동시에 요구되며, 처리 과정은 해당 병렬 머신만 사용된다. 준비 과정이 완료되면 해당 병렬 머신에서 작업의 처리가 바로 시작되고, 서버는 다른 일을 할 수 있다. 작업의 처리가 끝나면 제거 과정을 시행하게 되는데 제거 과정은 바로 시행할 필요는 없고 나중에 서버가 시간이 있을 때 시행하면 된다. 준비, 처리, 제거 과정들은 모두 한번 시작되면 완료될 때까지 실행되어야 하며 중간에 중지시켰다가 다시 시작할 수 없다. 작업은 한 개의 병렬 머신에만 할당될 수 있으며, 병렬 머신은 동시에 두 개 이상의 작업을 처리할 수 없고, 서버는 한 번에 한 개의 작업만 처리할 수 있다. 병렬 머신과 서버들은 시작 시점에 사용 가능하고, 작업의 수와 작업 시간들, 병렬 머신의 수들은 처음부터 고정되어 있고 중간에 변하지 않는다. 서버가 병렬 머신으로 이동하는 시간은 0이라고 가정한다. 이 문제의 목적은 모든 작업들의 총 완료 시간을 최소로 하는 스케줄을 찾는 것이다.

그림 1에 n = 5이고, 각 작업의 (s_i, p_i, r_i)가 J_1=(1, 4, 1), J_2=(1, 3, 1), J_3=(2, 2, 2), J_4=(1, 2, 1), J_5=(1, 1, 1)일 때 병렬 머신 스케줄링의 예가 나타나 있다. 그림 1(a)는 서버의 수(S)가 1이고 병렬 머신의 수(M)이 3일 때 스케줄링의 예로 총 완료 시간은 13이고, 그림 1(b)는 서버의 수가 2이고 병렬 머신의 수가 3일 때 스케줄링의 예로 총 완료 시간은 10이 된다.

slot	1	2	3	4	5	6	7	8	9	10	11	12	13
S1	s1	s2	s3			r1	s4	r2	s5	r3		r4	r5
M1	s1	p1				r1	s4	p4				r4	
M2		s2	p2					r2	s5	p5			r5
M3			s3		p3					r3			

(a) S=1, M=3일 때의 스케줄

slot	1	2	3	4	5	6	7	8	9	10	11	12	13
S1	s1	s3				r1	s5		r4				
S2	s2				r2	s4	r3			r5			
M1	s1	p1				r1	s5	p5		r5			
M2	s2	p3			r2	s4	p4		r4				
M3		s3		p3			r3						

(b) S=2, M=3일 때의 스케줄

그림 1. 병렬 머신 스케줄링의 예
Fig. 1. An Example of Parallel Machine Scheduling

본 연구에서 제안하는 알고리즘의 개요는 다음과 같다. 각 서버들은 사용 가능 시간이 빠른 순서로 우선 순위 큐에 저장되는 데 이 큐를 SQ라 한다. 또 각 병렬 머신들도 사용 가능 시간이 빠른 순서로 우선 순위 큐에 저장되는 데 이 큐를 MQ라 한다.

병렬 머신의 수를 m이라 하면 처음 m 개의 작업이 m 개의 병렬 머신에 할당 된다. 그 다음부터 새로운 작업을 어떤 병렬 머신에 할당하려면 그 병렬 머신에 이미 할당되어 있는 작업의 처리가 끝난 후 그 작업을 제거하여야 한다. avail은 현재 사용 가능한 병렬 머신의 수인데 초기값은 m이고, 병렬 머신에 작업이 할당될 때마다 1씩 감소하고, 어떤 병렬 머신에서 작업이 제거되어 사용 가능한 병렬 머신이 생기게 되면 1씩 증가된다.

알고리즘이 실행되는 과정을 보면 각 작업들은 처리 시간이 긴 것부터 병렬 머신에 할당한다. 현재 할당될 작업을 j라 하자. 먼저 SQ에서 사용 가능한 시간이 가장 빠른 서버를 삭제하는데 이 서버를 S_i라 하고 사용 가능한 시간을 $et(S_i)$라 한다. 그 다음 MQ에서 사용 가능한 시간이 가장 빠른 병렬 머신을 삭제하는데 이 병렬 머신을 M_k라 하고 사용 가능한 시간을 $et(M_k)$라 한다. 또 max를 $et(S_i)$와 $et(M_k)$ 중 최대 값이라 하자. avail = 0이면 현재 사용 가능한 병렬 머신이 없으므로 병렬 머신 M_k에 할당된 작업의 처리가 끝나면 이 작업을 제거하여 다른 작업이 할당될 수 있게 한다. M_k에 할당된 작업을 h라 하고 작업 h의 제거 시간을 r_h라 하면 서버 S_i와 병렬 머신 M_k는 구간 [max, max+r_h]에서 작업 h를 공동으로 제거하게 되고, 서버 S_i와 병렬 머신 M_k의 다음 사용 가능 시간은 둘 다 max+r_h로 변경되며, avail은 1 증가된다. avail > 0이면 서버 S_i와 병렬 머신 M_k가 구간 [max, max+s_j]에서 공동으로 작업 j의 준비 과정을 실행하고, 끝나는 즉시 병렬 머신 M_k는 구간 [max+s_j, max+s_j+p_j]에서 작업 j의 처리 과정을 실행한다. 그러므로 서버 Si의 다음 사용 가능 시간은 max+s_j, 병렬 머신 M_k의 다음 사용 가능 시간은 max+s_j+p_j로 변경된다.

모든 작업이 병렬 머신에 할당되면 각 병렬 머신에 할당된 작업의 처리가 끝나는 데로 제거하게 되고 모든 작업들이 제거되면 종료한다. 제안된 알고리즘이 그림 2에 나타나 있다. insertS()는 우선 순위 큐 SQ에 서버를 삽입하는 함수이고 deleteS()는 우선 순위 큐 SQ에서 사용 가능한 시간이 가장 빠른 서버를 삭제하는 함수이다. insertM()은 우선 순위 큐 MQ에 병렬 머신을 삽입하는 함수이고 deleteM()는 우선 순위 큐 MQ에서 사용 가능한 시간이 가장 빠른 병렬 머신을 삭제하는 함수이다. job(M_k)는 현재 병렬 머신 M_k에 할당되어

있는 작업을 가리킨다.

작업의 수가 n일 때 본 논문에서 제안하는 알고리즘의 시간 복잡도는 O(n)이 된다.

IV. 성능 평가

제안된 알고리즘은 C 언어를 사용해서 Intel(R) Core(TM) i5 CPU를 가진 PC에서 구현되고 실험되었다. 실험을 위해 데이터는 다음과 같이 생성되었다. 작업의 수는 1,000개를 사용하였고, 준비 시간과 처리 시간은 [3]에서와 같이 서버 부하(load)를 사용하여 임의로 생성되었다. 서버 부하는 준비 시간의 합을 처리 시간의 합으로 나눈 값으로 정의 된다. 서버 부하가 L일 때 처리 시간이 구간 [0, P]에서 임의로 생성되면 준비 시간은 구간 [0, P*L]에서 생성된다. 제거 시간은 준비 시간과 같은 구간에서 임의로 생성되었다. 서버 부하는 L=0.05, 0.2, 1.0 세 가지 경우를 고려하였는데, 각 경우에 대해 10개씩 임의로 데이터를 생성하여 평균 결과를 측정하였다. 서버의 수(S)는 1부터 5까지 5가지 경우를 사용하였고 병렬 머신의 수(M)는 10, 20, 30, 40, 50을 사용하였다.

```
Parallel Machine Scheduling Algorithm
{
   initialize SQ and MQ;
   for (each server i) insertS(Si);
   for (each machine k) insertM(Mk);
   avail = m;
   for (each job j) {
      Si = deleteS();
      Mk = deleteM();
      max = max(et(Si), et(Mk));
      if (avail = 0) {
         h = job(Mk);
         max = max+rh;
         et(Si) = max;
         et(Mk) = max;
         avail++;
      }
      et(Si) = max+sj;
      insertS(Si);
      job((Mk) = j;
      et(Mk) = max+si+pj;
      insertM(Mk);
      avail--;
   }
   while (MQ is not empty) {
      Si = deleteS();
      Mk = deleteM();
      max = max(et(Si), et(Mk));
      h = job(Mk);
      max = max+rh;
      et(Si) = max;
      insertS(Si);
   }
   retrurn max∀i∀k(et(Si), et(Mk));
}
```

그림 2 제안된 병렬 머신 스케줄링 알고리즘
Fig. 2 Proposed parallel machine scheduling algorithm

4.1 L = 0.05인 경우

서버 부하가 0.05인 경우는 준비와 제거의 평균 시간이 처리 평균 시간의 20분의 1이 되는 경우로 서버 부하가 낮은 경우이다. 표 1에 서버의 수와 병렬 머신의 수에 대한 총 완료 시간의 비율이 나타나 있다. 서버의 수가 1이고 병렬 머신의 수가 10일 때 총 완료 시간을 1이라 하고 다른 경우에 대한 총 완료 시간의 비율을 표시하였다.

표 1의 결과를 보면 서버의 수가 증가하거나 병렬 머신의 수가 증가하면 총 완료 시간이 감소하고 있음을 알 수 있다. 각 병렬 머신의 수에 따라 서버의 수가 미치는 영향을 보면 M이 10인 경우에는 S가 1일 때에 비해 S가 2일 때 총 완료 시간이 0.139 감소하였고, S가 3, 4, 5일 때는 각각 0.163, 0.172, 0.176 감소하였는데 서버의 수가 증가해도 감소 비율은 2.4%이하로 별로 감소하지 않았다. M이 50인 경우에는 S가 1일 때에 비해 S가 2일 때 총 완료 시간이 0.378(49.3%) 감소하였고 S가 3, 4, 5일 때도 각각 0.497(64.8%), 0.543(70.8%), 0.563(73.4%) 감소하였는데 이 결과를 보면 병렬 머신의 수가 많을 때 서버의 수를 증가시키면 결과가 더 좋은 것을 알 수 있다.

서버 수의 증가에 따른 총 완료 평균 시간을 보면 S가 1일 때에 비해, S가 2, 3, 4, 5인 경우 각각 0.306(37.5%), 0.381(46.6%), 0.408(49.9%), 0.42(51.4%) 감소하였다.

다음에 각 서버의 수에 대해 병렬 머신의 수가 미치는 영향을 보면 S가 1인 경우에는 M이 10에서 20으로 변할 때 총 완료 시간이 0.222 감소하고 그 다음부터는 M이 증가해도 1% 미만 감소하고 있다. S가 5인 경우에는 M이 10에서 20으로 변할 때 총 완료 시간이 0.4(48.5%) 감소하고 그 다음부터 M이 30, 40, 50일 때 각각 0.528(64.1%), 0.588(71.4%), 0.62(75.2%) 감소하였다.

표 1은 필요한 총 완료 시간을 달성하기 위한 서버의 수와 병렬 머신의 수를 결정하는데 사용할 수 있다. 예를 들어 S=1, M=10일 때의 총 완료 시간을 1이라 할 때, 이 시간을 반으로 줄이려면 S=2와 M=20을 사용하면 되고, 삼분의 일로 줄이려면 S=3과 M=30을 사용하면 된다.

표 1. L=0.05일 때 총 완료 시간의 비교
Table 1. The comparison of the total completion times for L=0.05

	M=10	M=20	M=30	M=40	M=50	평균
S=1	1.000	0.778	0.772	0.768	0.767	0.817
S=2	0.861	0.501	0.413	0.392	0.389	0.511
S=3	0.837	0.450	0.335	0.289	0.270	0.436
S=4	0.828	0.432	0.309	0.253	0.224	0.409
S=5	0.824	0.424	0.296	0.236	0.204	0.397

4.2 L = 0.2인 경우

서버 부하가 0.2인 경우는 응용 분야에 따라 약간의 차이가 있지만 서버 부하가 중간 정도 되는 경우로 볼 수 있다. 표 2의 결과를 보면 서버의 수가 증가하면 총 완료 시간이 L=0.05일 때에 비해 더 많이 감소하는 것을 알 수 있고, 반면에 병렬 머신의 수를 증가시켜도 총 완료 시간은 별로 감소하지 않음을 알 수 있다.

서버 수에 대한 결과를 보면 M이 10인 경우에는 S가 1일 때에 비해 S가 2일 때 총 완료 시간이 0.479 감소하였고, S가 3, 4, 5일 때는 각각 0.589, 0.624, 0.639 감소하였다. M이 50인 경우에는 S가 1일 때에 비해 S가 2, 3, 4, 5일 때 총 완료 시간이 각각 0.499, 0.665, 0.749, 0.798 감소하였는데 M이 10일 때에 비해 더 많이 감소하고 있음을 알 수 있다.

서버 수의 증가에 따른 총 완료 평균 시간을 보면 S=1일 때에 비해, S가 2, 3, 4, 5인 경우 각각 0.495, 0.649, 0.721, 0.761 감소하였다.

각 서버의 수에 대해 병렬 머신의 수가 미치는 영향을 보면 S가 1인 경우에는 병렬 머신의 수를 증가시켜도 총 완료 시간이 감소하지 않고 있음을 알 수 있고, S가 2, 3, 4, 5인 경우는 M이 10일 때에 비해 M이 20일 때가 제일 많이 감소하고 그 다음부터는 M이 증가해도 결과는 조금씩만 좋아지는 것으로 나타나 있다. M이 30일 때의 감소 비율이 M이 40, 50일 때와 거의 비슷하여 병렬 머신의 수를 30대 이상 사용하는 것은 무의미함을 나타냈다.

표 2. L=0.2일 때 총 완료 시간의 비교
Table 2. The comparison of the total completion times for L=0.2

	M=10	M=20	M=30	M=40	M=50	평균
S=1	1.000	0.999	0.999	0.999	0.999	0.999
S=2	0.521	0.500	0.500	0.500	0.500	0.504
S=3	0.411	0.336	0.334	0.334	0.334	0.350
S=4	0.376	0.260	0.252	0.251	0.250	0.278
S=5	0.361	0.224	0.203	0.202	0.201	0.238

4.3 L = 1.0인 경우

서버 부하가 1.0인 경우는 준비와 제거의 평균 시간이 처리 평균 시간과 같은 경우로 서버 부하가 매우 높아 병렬 머신들은 서버가 작업을 준비해주고 작업이 끝나면 제거해 줄 때까지 대부분의 시간을 가동하지 않은 상태로 기다리게 된다.

표 3의 결과를 보면 서버의 수가 같을 때는 병렬 머신의 수를 증가시켜도 총 완료 시간이 빨라지지 않음을 알 수 있다. 또 서버의 수를 증가시키면 병렬 머신의 수에 상관없이 거의 같은 비율로 총 완료 시간이 감소하고 있는데, 서버의 수가 k가 되면 총 완료 시간은 1/k로 되는 것을 볼 수 있다.

표 3. L=1.0일 때 총 완료 시간의 비교
Table 3. The comparison of the total completion times for L=1.0

	M=10	M=20	M=30	M=40	M=50	평균
S=1	1.000	1.000	1.000	1.000	1.000	1.000
S=2	0.500	0.500	0.500	0.500	0.500	0.500
S=3	0.334	0.334	0.334	0.334	0.334	0.334
S=4	0.251	0.250	0.250	0.250	0.250	0.250
S=5	0.203	0.200	0.200	0.200	0.200	0.201

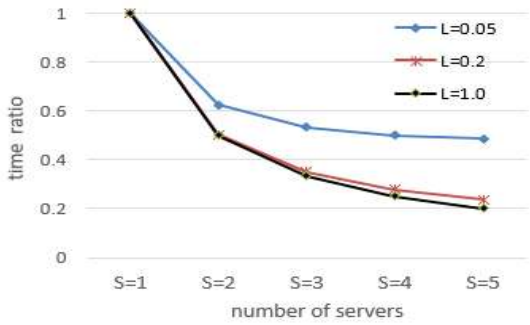

그림 3 서버의 수에 대한 평균 총 완료 시간의 비율
Fig. 3 Ratio of average total completion time on the number of servers

그림 3에 각 L 값에 대해 S가 1인 경우 총 완료 시간의 평균을 1이라 할 때 서버의 수가 2, 3, 4, 5인 경우 평균 총 완료 시간의 비율이 나타나 있다. 그림 3의 결과를 보면 서버 부하가 클수록 서버의 수를 증가시키면 총 완료 시간이 더 많이 단축됨을 알 수 있다.

V. 결 론

병렬 머신 스케줄링은 여러 대의 병렬 머신들이 있을 때 작업들의 총 완료 시간이 최소가 되게 작업들을 병렬 머신에 할당하는 문제로 다양한 응용 분야에서 사용되고 있다. 그 동안의 연구들은 주로 단일 서버를 사용하고, 각 작업들의 준비 시간과 처리 시간만을 고려하였고 최근에 다수의 서버를 사용하거나 작업의 제거 시간도 고려하는 연구가 진행되고 있다. 본 연구에서는 동일한 병렬 머신들을 사용하고 작업의 준비, 처리와 제거 시간을 모두 고려할 때 다수의 서버를 사용하여 총 완료 시간 최소로 하는 스케줄링 알고리즘을 제안하고, 또 서버 부하를 고려하여 다양한 데이터를 생성하여 서버의 수와 병렬 머신의 수에 따라 작업들이 완료되는 최종 시간이 어떻게 단축되는지 실험을 통해 분석하였다.

본 연구에서 얻은 실험 결과는 작업들의 총 완료 시간과 비용을 고려할 때 몇 대의 서버와 병렬 머신들을 사용하는 것이 최적이 되는 가를 결정하는 데도 활용될 것으로 기대된다.

참고문헌

〔1〕 J. H. Lee, J. M. Yu and D. H. Lee, "A tabu search algorithm for unrelated parallel machine scheduling with sequence- and machine-dependent setups: minimizing total tardiness, " Intl. J. of Adv. Manuf. Tech., 2013

〔2〕 K. Lee, J. Y.-T. Leung and M. L. Pinedo, " Makespan minimization in online scheduling with machine eligibility," Ann. Ope. res., vol. 204, pp. 189-222, 2013

〔3〕 C. P. Koulamas, "Scheduling two parallel semiautomatic machines to minimize machine interference," Computers and operation Research, vol. 23, no. 10, pp.945-956, 1996

〔4〕 N. G. Hall, C. N. Potts, and C. Sriskandarajah, "Parallel machine scheduling with a common server,"Discrete Applied Mathematics, vol. 102, pp. 223-243, 2000

〔5〕 A. H. Abdekhodaee and A. Wirth, "Scheduling parallel machines with a single server: some solvable cases and heuristics," Computers and Operation Research 29, pp. 295-315, 2002

〔6〕 X. Xie, H. Zhou, Y. Li, and Y. Zheng, "Scheduling Parallel Machines with a Single Server,"IEEE Intl. Conf. on MIC, pp. 453-456, 2012.

〔7〕 J. Ou, X. Qi, and C.Y. Lee, "Parallel Machine Scheduling with Multiple Unloading Servers," J. of Scheduling, vol. 13, no. 3 pp. 213-226, 2009

〔8〕 F. Werner and S.A. Kravchenko, "Scheduling with Multiple Servers," Automation and Remote Control, vol. 71, no. 10, pp. 2109-2121, 2010

〔9〕 F. Werner and S.A. Kravchenko, "Parallel Machine Scheduling with a Single Server," Mathematicsl and Computer Modelling, vol. 26, pp. 1-11, 1997

〔10〕 P. Brucker, C. Dhaenens-Flipo, S. Knust, S. A. Kravchchenko, and F. Werner, "Complexity results for parallel machine problems with a single server," J. of Scheduling, vol. 5, pp. 429-457, 2002

〔11〕 A. H. Abdekhodaee, A. Wirth and H .S. Gan, "Equal processing and equal setup time cases of scheduling parallel machines with a single server," Computers and Operation Research 31, pp. 1867-1889, 2004

〔12〕 J. Hu, Q. Jhang, J. Dong, and Y. J, "Parallel Machine Scheduling with a Single Server: Loading and Unloading," LNCS 8287, pp. 106-116, 2013

〔13〕 A. H. Abdekhodaee, A. Wirth and H .S. Gan, "Scheduling parallel machines with a single server: the general case," Computers and Operation Research 33, pp. 994-1009, 2006

〔14〕 H .S. Gan, A. A. Wirth and H. Abdekhodaee, "A branch-and-price algorithm for scheduling parallel machines with a single server," Computers and Operation Research 39, pp. 2242-2247, 2012

〔15〕 X. Xie, Y. Li, and Y. Zheng, "Scheduling Parallel Machines with a Single Server: a Dedicated Case,"Fifth Intl. Joint Conf. on Computational Science and Optimization, pp.

146-149, 2012.
〔16〕 C.Bussema and E. Torng, "Greedy multiprocessor server scheduling," Operations Research Letters, vol. 34, pp. 451-458, 2006.
〔17〕 C. Su, "Online LPT algorithms for parallel machines scheduling with a single server," J. Comb. Optim., 26, pp. 480-488, 2013
〔18〕 L. Y. Wang, X. Huang, P. Ji and E. M. Feng, "Unrelated parallel machine scheduling with deteriorating maintenance activities to minimize the total completion time ," Optim. Letters, 2012
〔19〕 C. W. Lin, Y. K. Lin and H. T. Hsieh, "Ant colony optimization for unrelated parallel machine scheduling," Intl. J. of Adv. Manuf. Tech., 2013

저 자 소 개

정 균 락
1980: 한국과학기술원
전자계산학 석사
1991: 미네소타대학교
컴퓨터공학 박사
1991 ~ 현재: 홍익대학교
컴퓨터공학과 교수
관심분야: 네트워크 알고리즘,
이동 통신,
무선 센서 네트워크,
머신 스케줄링 알고리즘,
VLSI 알고리즘
Email : chong@cs.hongik.ac.kr