메시지 패싱 시스템의 통신 특성을 고려한 개선된 태스크 스케줄링 기법

(Improved Task Scheduling Algorithm Considering the Successive Communication Features of Heterogeneous Message-passing System)

노 두 호 † 김 성 천 ††
(Dooho Noh) (Sungchun Kim)

요 약 본 논문에서는 메시지 패싱 시스템에서의 태스크 스케줄링에 대해 다룬다. 병렬/분산 시스템의 어플리케이션의 태스크에 대한 적절한 스케줄링이 이루어지지 않는 경우, 병렬/분산 처리를 이용한 이득을 기대하기는 어렵기 때문에 이 주제에 대한 연구는 컴퓨터 아키텍처의 발달과 함께 지속되고 있으며, 많은 연구들이 태스크 스케줄링에 대한 다양한 기법들을 제안하고 있다. 기존의 연구들은 공유 메모리 시스템을 가정하여 이루어졌기 때문에, 메시지 패싱 시스템에 기존의 기법을 적용하기가 힘들다. 본 논문에서는 기존 연구의 모델과 메시지 패싱 시스템의 통신 모델의 차이점으로 발생하는 통신비용의 누적을 고려하여 리스트 스케줄링 기법에 기초한 개선된 우선 순위 함수와 새로운 프로세서 선택 기준을 제안한다. 이들 두 가지 제안을 적용한 태스크 스케줄링 기법은 통신비용의 누적을 고려하지 않아 발생하는 비효율적인 스케줄링을 개선한다.

키워드 : 태스크 스케줄링, 메시지 패싱, 통신 비용

Abstract This thesis deals with a task scheduling on a message-passing system. Scheduling and allocation are very important issues since the inappropriate scheduling of tasks cannot exploit the true potential of the system and it can offset the grain from parallelization. It is difficult to apply previous schemes to message-passing system, because previous schemes assume the shared memory system. This thesis proposes an modified priority function and processor selection technique that consider the problems caused by the difference between previous models and message-passing environments. The priority function includes the cumulative communication cost which causes task execution to be delayed. The processor selection technique avoids the situation that a child task is assigned to the same processor allocated to its parent task that has other unscheduled child tasks. We showed by some simulations that our modified features of task scheduling algorithm can make the better scheduling results than the previous algorithms.

Key words : message passing, task scheduling, comunicaion cost

1. 서 론

태스크 스케줄링은 태스크의 실행 순서와 태스크가 실행될 프로세서를 태스크들의 연관성을 고려하여 전체적인 어플리케이션의 실행 시간을 단축시킬 수 있도록 결정하는 것이다[1,2]. 병렬/분산 처리에 있어서 태스크 스케줄링이 적절하게 이루어지지 못한 경우 시스템의 성능과 병렬 처리에 의한 이득을 얻기 힘들기 때문에 아키텍처의 발전과 함께 태스크 스케줄링 기법에 대한 연구가 지속되고 있다.

최적의 실행 시간을 보장하는 태스크 스케줄링을 찾는 문제는 몇 가지 제한된 경우를 제외하곤 NP-complete이다[1,2]. 그렇기 때문에 많은 연구들이 각기 연구 대상으로 삼고 있는 병렬/분산 어플리케이션이나 병렬/분산 처리 시스템에 적합한 휴리스틱 기법을 제안하고 있다.

· 본 연구는 한국과학재단 목적 기초연구 (R01-2001-000-00356-0) 지원으로 수행되었음
† 비 회 원 : 서강대학교 컴퓨터학과
stdu@nate.com
†† 종신회원 : 서강대학교 컴퓨터학과 교수
ksc@arqlab1.sogang.ac.kr
논문접수 : 2003년 2월 3일
심사완료 : 2004년 3월 16일

태스크 스케줄링에 대한 초기 연구에서는 태스크간의 통신에 의한 지연을 고려하지 않았다[1]. 그러나 병렬 시스템은 DSM, SMP, MPP, 클러스터 시스템 등의 다양한 방향으로 연구가 진행되고 있으며[3], 이러한 시스템에서 통신은 이전에 비해 보다 증가하게 되었고 태스크의 실행순서와 태스크가 시작되는 시간에도 영향을 주게 되었다. 따라서 초기의 연구들이 제안한 태스크 스케줄 기법을 그대로 사용할 수 없게 되었다.

태스크간의 통신에 의한 지연을 고려한 스케줄링 기법들은 리스트 스케줄링 기법, 클러스터 기법, 태스크 복제 기법들이 있다[4]. 이들 기법은 태스크의 우선 순위를 결정하는 방법을 달리하거나 통신으로 인한 지연시간을 최소화한다[1,2,4]. 하지만 이들 기법들을 DSM과 같은 메모리를 공유하는 시스템을 대상으로 하는 기법들이며, 클러스터 시스템과 같은 메시지 패싱 기반의 시스템에서는 이득을 기대할 수 없다.

따라서 클러스터 시스템과 같은 메시지 패싱 환경에서의 통신을 고려한 태스크 스케줄링 기법이 연구되어야 한다. 본 논문은 메시지 패싱 환경에서의 통신 특성인, 동기식 통신이 순차적으로 일어나는 환경을 고려하여 개선된 우선 순위 함수와 새로운 프로세서 선택 기준을 포함하는 새로운 스케줄 기법을 제안한다. 새로운 기법은 기존 기법이 메시지 패싱 시스템에서 사용될 때 발생하는 비효율적 스케줄링을 개선한다.

본 논문의 구조는 다음과 같다. 2장에서는 태스크 스케줄 기법의 기본적인 개념을 정리하며, 기존 연구들이 제안한 기법들의 문제점을 살펴본다. 3장에서는 개선된 우선 순위 함수와 새로운 프로세서 선택 기준이 포함된 스케줄링 기법을 제안한다. 4장에서는 실험을 통하여 성능을 분석하며, 5장에서는 본 논문의 결론을 내린다.

2. 기존의 태스크 스케줄링 기법

이 장에서는 태스크 스케줄링 기법에 사용되는 용어에 대해 설명하며, 기존 연구들이 제안한 기법들을 간략하게 살펴본다. 또한 기존의 연구들이 가정한 모델이 메시지 패싱 기반의 환경에 부적합한 이유를 설명한다.

2.1 DAG

태스크 스케줄링 기법에서 어플리케이션은 DAG (Directed Acyclic Graph)로 표현된다. DAG는 어플리케이션의 태스크와 통신을 각각 버텍스(vertex)와 에지(edge)로 표현한다. 이것의 정의는 다음과 같다. $G=(V, E)$, V와 E는 각각 태스크 t와 통신 e의 집합이고, w_i는 t_i의 계산비용을 나타내며, $c_{i,j}$는 t_i와 t_j사이의 통신비용을 나타낸다.

2.2 기존의 태스크 스케줄링 기법

초기의 태스크 스케줄링 기법은 태스크간의 통신을 고려하지 않고 있으며 기본적인 아이디어는 다음과 같다. 그래프에 존재하는 모든 태스크에 우선 순위를 부여하고, 우선 순위에 따라서 태스크 리스트가 작성되며, 리스트의 모든 태스크가 할당 될 때까지 다음 과정이 반복된다.

1) 태스크 리스트에서 가장 우선하는 준비(ready) 상태의 태스크를 태스크가 실행될 유휴(Idle) 상태의 프로세서를 선정하여 할당한다.

2) 태스크 리스트에서 부모 프로세스의 실행이 모두 종료된 태스크를 준비 상태로 바꾼다.

태스크의 우선 순위는 태스크가 갖는 속성 값에 의하며, 가장 대표적인 속성은 레벨(*level*)과 코레벨(*co-level*)이 있다. 이들 속성은 각기 바텀 레벨(*bottom-level*)과 탑 레벨(*top-level*)이라고 불리기도 한다. 이들 속성을 이용하여 HLFET (highest levels first with estimated times), RANDOM, SCFET(smallest co-levels first with estimated times) 등과 같은 휴리스틱 기법들이 제안되었으며, HLFET가 가장 우수한 결과를 보여준다.[1] 레벨에 대한 정의는 다음과 같다.

레벨(Level) *DAG에서 어떤 태스크의 레벨은 그 태스크에서부터 종료 태스크까지의 가장 긴 경로의 길이이다. 경로의 길이는 경로상에 놓인 태스크의 계산비용의 합이다[1].*

초기의 기법들은 통신을 고려하지 않았으나, 병렬/분산 아키텍처가 발전하여 통신비용이 보다 높은 비중을 차지하게 됨에 따라 통신비용에 대한 고려가 필요하게 되었다. 따라서 통신을 고려한 기법으로 통신을 고려한 리스트 스케줄링 기법, 클러스터 기법, 태스크 복제 기법 등이 제안되었다[1,4].

이 기법들은 기존의 레벨이나 코레벨을 개선시켜 태스크 그래프에서 통신비용이 우선 순위에 영향을 주도록 하였다. 또한 이들 기법들은 통신비용을 최소화하기 위해 불필요한 태스크의 병렬화(parallelism)를 제거하고 되도록 통신이 같은 프로세서에서 이루어지도록 한다. 하지만 이들 기법은 메모리를 공유하는 시스템에 적합한 기법으로 클러스터 시스템과 같은 메시지 패싱 기반의 환경에는 적합하지 않다.

2.3 기존 기법의 문제점

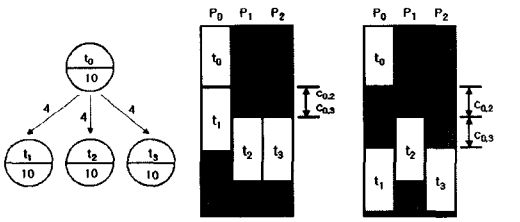
메시지 패싱 환경에서 병렬/분산 어플리케이션은 다른 프로세서가 갖고 있는 데이타에 접근하기 위해 시스템이 제공하는 서비스를 이용한다. 이때 기존의 모델과는 달리 메시지 패싱 시스템은 통신이 이루어질 때 자식 태스크가 할당된 프로세서뿐만 아니라 부모 태스크가 할당된 프로세서에도 영향을 준다. 따라서 동기식의 통신이 순차적으로 이루어지며, 기존의 기법들은 이러한 통신 특성을 반영하지 못하기 때문에 이에 대한 개선이

필요하다.

태스크 스케줄링과 관련된 최근의 연구들[5-7]은 이러한 메시지 패싱 환경의 통신 특성을 고려하고 있다. 하지만 이들 연구들은 메시지 패싱 환경에서 통신이 순차적으로 이루어지는 특성과 동기식으로 이루어지는 특성 모두를 동시에 고려하고 있지는 않다. 본 논문에서는 이러한 두 가지 특성을 모두를 고려하여 기존의 태스크 스케줄링 기법을 개선한다.

3. 개선된 스케줄링 기법

메시지 패싱 시스템에서의 통신은 동기식으로 이루어지고, 다수의 태스크에 각기 다른 메시지를 전달해야하는 경우 각각의 태스크와 순차적으로 통신이 이루어진다. 이러한 순차적인 통신 모델과 기존의 통신모델의 가장 큰 차이점은 순차적 통신에 의한 지연이 통신과 관계된 자식 태스크의 실행뿐만 아니라 다른 자식 태스크들의 실행까지 지연시킨다는 점이다. 그림 1은 기존 연구의 모델과 메시지 패싱을 사용하는 클러스터 시스템에서 통신의 차이점을 보여준다.



(a) DAG (b) 기존 시스템에서의 통신 (c) 클러스터 시스템에서의 통신

그림 1 기존 시스템과 클러스터 시스템에서의 통신

3.1 태스크 레벨의 개선

어떤 태스크 t_i의 레벨은 t_i에서 종료 태스크까지의 가장 긴 경로의 길이이다[1,2]. 초기에는 계산비용의 합으로만 계산되었으나 통신의 비중이 커짐에 따라 통신비용까지 함께 계산되게 되었다. 통신비용까지 고려한 레벨과 구분하여 이전의 레벨은 정적(static) 레벨이라 불린다. 통신비용을 고려한 레벨은 다음과 같이 정의된다.

$$level(t_i) = w_i + \max(c_{i,j} + level(t_j)),$$
$$t_j \in \{t_k \mid t_k \text{ is child of } t_i\} \quad (1)$$

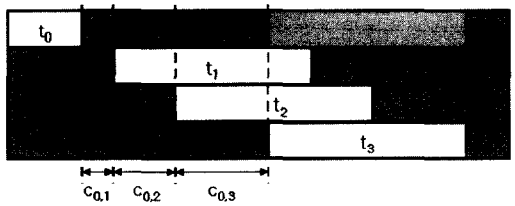
자식 태스크의 통신비용과 레벨 값의 합에서 최대값을 취한 것은 각각의 통신이 다른 통신에 영향을 주지 않고 개별적으로 이루어지기 때문이다. 하지만 본 논문의 모델에서는 통신이 순차적으로 이루어지며 다른 태스크의 실행이나 통신에 영향을 준다. 본 논문에서는 순차적 통신을 고려하여 레벨을 다음과 같이 개선하였다.

$$level(t_i) = w_i + \sum_{t_k} c_{i,k} + \max(level(t_k)),$$
$$t_k \in \{t_j \mid t_j \text{ is child of } t_i\} \quad (2)$$

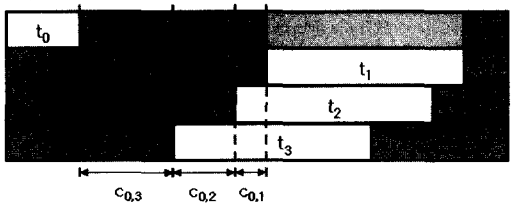
자식 태스크들과의 통신비용의 합은 다수의 태스크와 통신이 이루어지는 경우 통신비용이 누적되는 것을 반영한다. 같은 프로세서에 태스크가 할당돼서 태스크간의 통신비용이 제거되는 것을 고려하지 않을 때, 태스크 t_i와 자식 태스크 t_j의 간에 생길 수 있는 가장 큰 통신비용은 모든 자식 태스크와의 통신비용의 합이기 때문이다.

3.2 우선 순위 함수의 개선

그림 2는 자식 태스크의 스케줄링 순서에 따른 태스크의 시작 시간을 보여준다. 부모 태스크와의 통신비용이 $c_{01} < c_{02} < c_{03}$일 때, (a)는 t_1, t_2, t_3의 순서로 태스크가 할당된 경우이며 (b)는 t_3, t_2, t_1의 순서로 할당된 경우이다. 그림 2.를 통해서 통신비용이 작은 태스크를 먼저 스케줄링하는 것이 통신비용이 큰 태스크를 먼저 스케줄링하는 것보다 전체적인 태스크 시작 시간에 유리하다는 것을 알 수 있다.



(a) 통신비용이 작은 태스크가 먼저 할당된 경우



(b) 통신비용이 큰 태스크가 먼저 할당된 경우

그림 2 태스크 스케줄 순서에 따른 태스크 시작 시간의 변화

이러한 특성을 반영하기 위해서 본 논문에서는 태스크의 크기가 비슷한 경우 통신 시간의 길이가 짧은 태스크가 먼저 할당되도록 $SCF(t_i)$ 함수를 정의하였다.

$$C_j = \sum_{t_k} c_{j,k},\ t_k \in \{t_n \mid t_n \text{ is child of } t_j\} \quad (3)$$

$$SCF(t_i) = C_j - c_{i,j},\ t_j \text{ is parent of } t_i \quad (4)$$

t_j가 t_i의 부모 태스크일 때, $SCF(t_i)$는 t_i와 부모 태스크 t_j간의 통신비용 $c_{i,j}$을 제외한 부모 태스크 t_j의 나머지 자식 태스크 사이의 통신비용의 합이다. 위 식에서 $SCF(t_i)$의 값이 크면 t_i의 통신비용은 작은 것을 의미하

며, 이것은 t_j의 다른 자식 태스크들이 t_i와 t_j의 통신으로 이해서 지연되는 시간이 줄어드는 것을 의미한다. 앞에서 정의한 $level(t_i)$와 $SCF(t_i)$에 의해서 본 논문의 알고리즘에서 사용되는 우선 순위 $priority(t_i)$는 다음과 같이 정의된다.

$$priority(t_i) = level(t_i) + SCF(t_i) \quad (5)$$

3.3 개선된 리스트 스케줄링 기법

본 논문에서 제안하는 태스크 스케줄링 기법은 리스트 스케줄링 기법을 기본으로 한다. 스케줄링 기법은 태스크 순서화 과정과 프로세서 선택 과정으로 나뉜다. 모든 태스크의 계산비용과 통신비용은 미리 계산되어 있다고 가정한다. 태스크 순서화 과정에서는 먼저 DAG의 태스크들을 순서화 하기 위해서 각 태스크의 *priority* 값이 계산된다. 필요한 모든 계산이 끝나면 태스크들을 *priority* 값에 대한 내림차순으로 정렬하여 태스크 리스트를 작성한다.

프로세서 선택 과정은 태스크 리스트에 더 이상 스케줄링 되지 않은 태스크가 없을 때까지 반복된다. 매 과정에서 리스트의 가장 앞에 위치하는 태스크 중 준비 상태의 태스크가 스케줄링 대상으로 선정된다. 태스크가 선정된 후에는 프로세서들 중에서 할당되기에 프로세서 선택 기준을 적용하여 적당한 프로세서를 찾는다. 마지막으로 선택된 프로세서에 태스크를 할당하고 할당된 태스크와의 의존관계에 의해서 준비상태가 아니었던 태스크들의 상태를 준비상태로 바꾼다.

기존 연구에서는 프로세서 선택 기준으로 태스크가 가장 빨리 시작될 수 있는 시간이 사용되었다. 프로세서의 처리 능력이 각기 다른 경우 각 프로세서에서의 수행시간을 각각 계산한 후 가장 빨리 태스크 수행을 종료할 수 있는 프로세서가 선택된다[4]. 이 경우 같은 프로세서 내에서의 통신은 비용이 제거되기 때문에 부모 태스크가 할당된 프로세서가 가장 먼저 선택되는 경향이 있다.

하지만 본 논문에서는 어떤 태스크가 다수의 태스크에 정보를 전달하고자 할 경우 순차적으로 통신이 이루어지는 것을 가정한다. 따라서 선정된 태스크가 부모 태스크의 유일한 자식 태스크가 아닌 경우에 가장 먼저 선정되는 태스크가 부모 태스크와 같은 프로세서에 할당되게 되면 자식 태스크는 부모 태스크가 다른 통신들이 끝날 때까지 태스크 실행이 지연된다. 따라서 제안하는 알고리즘에서는 태스크가 할당되지 않은 자식 태스크를 갖는 부모 태스크와 같은 프로세서에 할당되는 것을 회피한다. 새로운 프로세서 선택 기준은 다음과 같다.

프로세서 선택 기준 :

1) *태스크가 하나의 부모 태스크를 갖으며, 태스크가 부모 태스크의 유일한 자식 태스크인 경우 부모 태스크와 같은 프로세서를 선택한다. 태스크가 부모 태스크의 유일한 자식 태스크가 아닌 경우 부모 태스크가 할당된 프로세서를 제외한 프로세서 중 가장 빨리 태스크를 실행할 수 있는 프로세서를 선택한다.*
2) *태스크가 둘 이상의 부모 태스크를 갖는 경우 현재 태스크를 유일한 자식 태스크로 갖는 부모 태스크가 할당된 프로세서가 유휴상태가 되는 시간과 가장 빨리 태스크를 실행할 수 있는 프로세서에서의 태스크 시작 시간 중 더 작은 값을 갖는 프로세서를 선택한다.*

이러한 개선점을 포함한 개선된 리스트 스케줄링 기법은 기존의 기법이 통신비용의 누적을 고려하지 못하여 발생하는 비효율적인 스케줄링을 개선한다.

본 논문이 제안한 개선된 리스트 스케줄링 기법은 기존의 리스트 스케줄 기법보다는 좀더 많은 비용을 필요로 한다. 하지만 복잡도는 기존의 $O(v^2 \times p)$와 같으며, 정적 스케줄링 기법의 특성상 스케줄링 과정에서 얻는 시간적 이득보다는 스케줄링을 통하여 얻을 수 있는 어플리케이션 실행에 있어서의 시간적 이득이 중요하기 때문에 큰 문제가 되지 않는다.

4. 실험 및 결과 분석

기존의 리스트 스케줄링 기법과 제안기법을 구현한 스케줄러를 사용하여 스케줄링 결과를 비교한다. 각 스케줄러는 스케줄링 된 순서와 시작 시간에 따라 태스크와 통신을 할당하며, 할당된 각 태스크와 통신의 시작 시간과 소요 시간의 기록을 비교하게 된다. 각 기법의 스케줄러는 C언어로 구현하였다.

태스크 스케줄링 기법의 평가를 위해 전체 어플리케이션이 실행되는 시간을 비교하며, 이것은 태스크들이 프로세서에 스케줄링 된 길이, *makespan*으로 측정된다. 하지만 많은 수의 태스크를 갖는 경우 *makespan*으로 성능을 비교하기가 곤란해진다. 이러한 경우 CP(critical path) 상의 태스크들의 계산비용의 합에 대한 *makespan*의 길이로 성능을 평가하며 스케줄 길이 비율(Schedule length ratio, SLR)이라 불린다. *SLR*은 다음과 같이 정의된다.

$$SLR = \frac{makespan}{\sum_{t_i} w_i}, \; t_i \in CP \quad (6)$$

이 실험에 사용된 어플리케이션 모델은 가우스 소거법(Gaussian Elimination)의 태스크 그래프를 사용하였다. 가우스 소거법은 연립방정식의 해를 구하는 방법으로 실세계의 많은 문제를 해결하는데 사용되고 있으며, [4], [8] 등의 여러 연구에서 스케줄링 기법의 비교를 위한 어플리케이션 모델로 사용되고 있다.

4.1 프로세서 변화에 따른 성능 비교

프로세서의 수를 4에서 20까지 증가시키면서 실험하였

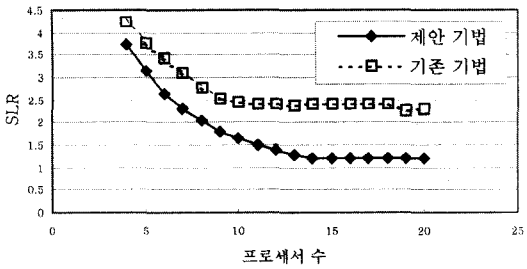


그림 3 프로세서 수에 따른 SLR

으며, 같은 프로세서의 수에 대하여 각기 200번의 시행을 통하여 평균값을 얻어냈다. 사용된 태스크 그래프는 행렬의 크기가 16인 가우스 소거법의 태스크 그래프이고 *CCR*(communication to computation ratio)은 0.4이다.

그림 3에서 기존기법은 프로세서 수가 10일 때부터 SLR 값이 수렴하지만 제안 기법은 프로세서 수가 15일 때부터 수렴한다. 기존기법에 비하여 보다 프로세서를 효율적으로 활용하는 것을 알 수 있다.

4.2 태스크 규모에 따른 성능 비교

이 실험에서 행렬 크기를 5에서 21까지 변화 시켰으며, 이에 따라서 태스크는 14에서 230개까지 변화하였다. 또한 *CCR*의 값이 0.15와 0.7이 되도록 하였다. 프로세서의 수는 8개라고 가정하였으며, 실험 결과는 행렬의 크기에 대해서 각각 500번의 시행을 통하여 평균값을 산출하였다.

그림 4에서 CCR이 0.15일 때 기존기법에 비해 제안 기법이 보다 짧은 실행시간을 보인다. 특히 행렬의 크기가 프로세서 수보다 적을 때 제안기법의 *SLR* 값은 1에 근접한다. 이 실험에서 제안 기법은 기존 기법에 대해서 평균적으로 약 37% 짧은 스케줄 길이를 갖는 스케줄링 결과를 얻을 수 있었다.

그림 5에서는 두 기법 모두 CCR이 작을 때보다 나쁜 성능을 보인다. 하지만 제안 기법이 평균적으로 기존기법에 비해 97% 짧은 스케줄링 결과를 보여준다. 또한 행렬 크기가 프로세서 수보다 적을 때 *SLR*은 최적화에 가까운 성능을 보인다. 앞의 *CCR*이 0.15일 때와 비교

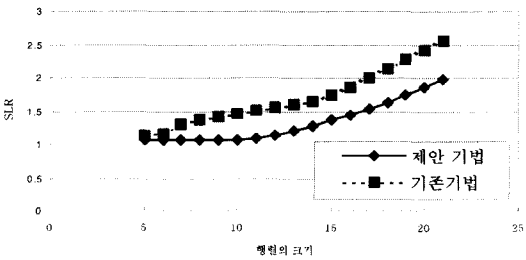


그림 4 *CCR*=0.15에서의 행렬의 크기 변화에 따른 *SLR*의 변화

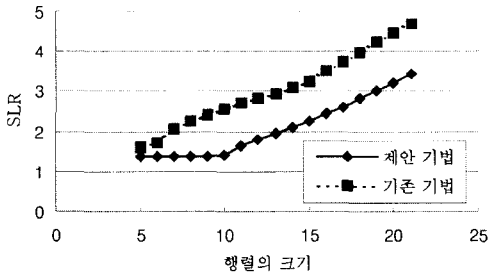


그림 5 *CCR*=0.7에서의 행렬 크기 변화에 따른 *SLR*의 변화

하면, 기존 기법과 제안 기법 모두 *CCR*이 증가함에 따라 성능이 저하되고 있음을 알 수 있지만, 성능 감소가 기존 기법에 비하여 적은 것을 알 수 있다.

5. 결 론

본 논문은 동기식 통신이 순차적으로 발생하는 메시지 패싱 시스템에 적합하도록 리스크 스케줄 기법을 개선하였다. 개선된 우선 순위 함수에서는 기존의 레벨을 개선하여 통신비용이 누적되는 것을 고려하였다. 또한 프로세서 선정 과정을 개선하여 스케줄링 되지 않은 자식 태스크가 남아있는 경우 부모 태스크와 같은 프로세서로 태스크가 할당되는 것을 회피하도록 하였다. 이들 두 가지 제안을 통하여 기존의 태스크 스케줄링 기법이 클러스터 시스템에서 통신비용의 누적을 고려하지 않아 발생하는 비효율적인 스케줄링을 개선하였다.

실험 결과 프로세서가 증가함에 따라서 기존기법보다 짧은 스케줄링 길이를 갖는 결과를 얻을 수 있었다. 기존기법에 비해서 통신비용에 의한 영향을 더 적게 받는 것으로 나타났으며, 제안기법이 좀더 효율적인 스케줄링을 수행하는 것을 알 수 있었다. 따라서 메시지 패싱 환경의 통신 특성을 고려하지 않아 발생한 비효율적인 스케줄링이 개선되었음을 확인 할 수 있었다.

본 논문이 개선한 리스트 스케줄링 기법은 다른 많은 휴리스틱 기법의 기초가 되는 기법으로 제안된 태스크의 레벨 값과 프로세서 선택 과정은 다른 많은 기법에 적용할 수 있을 것으로 생각되며, 이것을 이용한 보다 최적화된 기법이 향후 연구로써 진행되어야 할 것이다.

참 고 문 헌

[1] H. El-Rewini, T. G. Lewis and H. H. Ali, *Task Scheduling in Parallel and Distributed Systems*, PTR Prentice Hall, USA, 1994, pp.17-81.

[2] Y.-K. Kowk and I. Ahmad, "Static Scheduling Algorithms for Allocating Directed Task Graphs to Multiprocessors," ACM Computing Surveys,

vol. 31, no. 4, 1999, pp.406-471.
[3] K. Hwang and Z. Xu, *Scalable Parallel Computing*, McGraw-Hill, 1998, pp.13-36.
[4] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing," IEEE Transactions on parallel and distributed system. vol. 13, no. 3, March 2002, pp.260-274.
[5] T. Hsu and D. R. Lopez, "Task Allocation on a Network of Processors," IEEE Transactions on computers, vol. 49, no. 12, December 2000, pp.1339-1353.
[6] B. R. Arafeh, "Clustering Algorithm for Scheduling Parallel Programs on NOWs with Synchronization Requirements at the Application Level," proceedings of IPDPS'02, Fort Lauderdale, California, April 2002, http://www.computer.org/ proceedings/ipdps/1573/symposium/1573toc.htm
[7] D. Kadamuddi and J. J. P. Tsai, "Clustering Algorithm for Parallelizing Software Systems in Multiprocessors Environment," IEEE Transactions on Software Engineering, vol. 26, no. 4, April 2000, pp.340-361.
[8] I. Ahmad and Y.-K. Kwok, "On Parallelizing the Multiprocessor Scheduling Problem," IEEE Transactions on parallel and distributed system. vol. 10, no. 4, April 1999, pp.414-432.

노 두 호
2001년 서강대학교 컴퓨터학과 공학학사
2003년 서강대학교 컴퓨터학과 공학석사
관심분야는 태스크 스케줄링, 메시지 패싱, 분산, 병렬처리

김 성 천
1975년 서울대학교 공과대학 공학사. 1979년 Wayne State University, M.S. 1982년 Wayne State University, Ph.D. 1985년 서강대학교 컴퓨터학과 교수
관심분야는 통신, 컴퓨터, 분산, 병렬처리