분산시스템에서 효율적인 타스크 할당 알고리즘

김 주 만[†] · 박 치 항[††] · 이 철 훈[†††]

요 약

본 논문에서는 분산 시스템에서의 타스크 할당 문제를 다룬다. 타스크 할당에 있어서 부하의 최대 균등과 프로세서 간 통신(IPC : interprocessor communication)의 최소화를 동시에 추구하여야 하나, 특성상 이 두 목적들은 서로 상충되므로, 주어진 타스크 부류에 따라 그들 사이에서 절충을 해야 한다. 대부분의 기존 방법들은 주어진 부하 균등치 조건을 만족하는 범위 내에서 IPC를 최소화한다. 그들은 부하 균등치를 하나의 조건으로 놓고 IPC 최소화만을 목적으로 하기 때문에IPC와 부하 균등 사이에서 절충하기가 어렵다. 그러나 최적의 절충을 취하면서 부하 균등과 IPC 최소화를 동시에 처리하는 것이 바람직하다. 따라서, 본 논문에서는 먼저 이들 두 목적들의 상관 관계를 명확히 표현하는 새로운 비용 함수를 제안한다. 부하 분배에 대한 통계적 편차를 이용하여 프로세서들의 부하 균등치를 표현한다. 또한, 가중치를 체계적으로 조정함으로써 이들 목적 사이에서 절충을 취할 수 있는 정책을 제안한다. 그래프 변환 방법을 사용하여 타스크 할당 문제가 최소 N-컷 문제로 변환됨을 보이고, 이 문제에 대한 휴어리스틱 알고리즘을 제안한다. 여러 종류의 타스크-프로세서 시스템 상에서의 실험을 통하여 본 논문에서 제안한 방법이 기존의 방법보다 성능이 우수함을 보인다.

An Efficient Task Assignment Algorithm in Distributed System

Joo-Man Kim[†] · Chee-Hang Park[††] · Cheol-Hoon Lee[†††]

ABSTRACT

This paper deals with the static task-assignment problem in a distributed computing system. In assigning tasks, we have to pursue maximization of load balancing and minimization of interprocessor communication (IPC) simultaneously. However, since these two goals conflict each other, one has to make a compromise between them according to the given task type. Most of the existing approaches minimize IPC subject to constraints on the degree to which the processors' loads are balanced. Since they consider the minimization of IPC as the only objective while using load balancing just as a constraint, it is difficult to make a tradeoff between IPC and load balancing. However, it is desirable to simultaneously balance loads and minimize IPC by making an optimal tradeoff between the two conflicting goals. We therefore propose a new cost function to evaluate static task assignments, explicitly describing the tradeoff between the two goals. The variance statistics of load distribution are used to represent the degree of load balancing among the processors for a given assignment. Also suggested is a policy which enables the system designer to make a tradeoff between the two goals by systematically adjusting a weighting factor. It is shown that the task-assignment problem can be transformed into the minimum N-cut problem using a graph modification technique. We propose a heuristic algorithm for solving the transformed problem. Simulation results show that our approach outperforms an existing representative approach for a variety of task and processing systems.

† 정 회 원 : ETRI 컴퓨터구조연구실 책임연구원
†† 정 회 원 : 한국전자통신연구원 책임연구원
††† 정 회 원 : 충남대학교 컴퓨터공학과 조교수
논문접수 : 1997년 11월 3일, 심사완료 : 1997년 12월 26일

1. 서 론

VLSI와 컴퓨터 네트워킹 기술의 발전으로 여러 응용 분야에서 분산 처리 시스템이 큰 호응을 얻고 있다. 분산 처리 시스템은 다른 지역에 존재하는 자원이나 데이타를 이용할 수 있는 기능을 제공할 뿐만아니라 다수의 프로세서 및 통신 경로를 가지고 있으므로 시스템 성능과 신뢰성을 향상시킬 수 있다. 본 논문에서는 이와 같은 분산 시스템에서 해결해야 할 가장 중요한 문제들 중의 하나인 타스크 할당 문제를 다룬다.

지금까지 여러 경우에 대해서 타스크 할당 알고리즘들이 많이 제시되어 왔다. Stone과 Bokhari는 전체 수행 및 통신 비용을 최소화하는 타스크 할당 방법들을 제시했다 [1-4, 20-22]. Lee 등은 Stone의 2-프로세서 문제에 대한 네트워크 흐름 (network-flow) 방법을 선형 배열 시스템에서 일반적인 N-프로세서 문제로 확장을 하였다 [13]. Lo는 Stone의 모델을 확장하여 타스크 할당에서 동시성을 향상시키는 알고리즘을 제안하였다 [14]. 이들 중 많은 알고리즘들은 타스크와 프로세서 시스템을 그래프로 표현하여 Ford와 Fulkerson에 의해 처음 제안된 네트워크 흐름 알고리즘을 [7] 이용하여 전체 수행 및 통신 비용이 최소인 할당을 찾는다 [1, 2, 12, 14, 20, 21]. 다른 학자들은 주어진 부하 균등 조건 내에서 프로세서 간의 통신(IPC :interprocesor communication) 비용을 최소화하거나 [5, 15], 또는 전체 수행 시간을 최소화하는 타스크 할당 문제를 다루었다 [19]. 타스크 할당에서 일반적으로 추구하는 목적은 IPC를 최소화하고 또한 부하 균등을 최대화 하는 것이다. 이러한 타스크 할당 문제는 NP-complete 문제이므로, 최적에 가까운 할당을 빨리 찾을 수 있는 효율적인 휴어리스틱 방법에 의존할 수 밖에 없다 [14, 17].

본 논문에서는 분산 처리 시스템 상에서 각 프로세서들의 부하를 균등히 함과 동시에 IPC를 최소화 하는 타스크 할당 문제를 다룬다. IPC를 최소화하기 위해서는 가능한 한 타스크들을 하나 혹은 적은 수의 프로세서에 할당하여야 하고, 반대로 부하 균등을 위해서는 가능한 한 모든 프로세서들에게 균등하게 분배되어야 한다. 이와 같은 두 가지 목적은 서로 상충되므로, 타스크 부류에 따라 이들 사이에서 절충을 취해야만 한다. 예를 들어, 주어진 타스크가 계산-제한적(computation-bound) 또는 통신-제한적(communication-bound)인 경우에는 부하 균등 또는 IPC 최소화에 각각 보다 비중을 두어야 한다.

타스크 할당 문제는, 그래프 상에서 전체 노드 집합을 N 개의 부집합으로 나누되 각 부집합들의 크기를 균등하게 함과 동시에 (즉, 부하 균등) 값이 최소가 되는 (즉, IPC 최소화) N-컷을 찾는 문제와 동일하다. 이것을 균등 최소 N-컷(*balanced minimum N-cut*) 문제로 불리며, 이미 NP-complete으로 증명이 되어 있다 [8, 11]. Kernighan과 Lin은 $N=2$인 경우에 대해서 아주 효율적인 휴어리스틱을 제안하였다 [9]. 이 알고리즘은 $N>2$인 일반적인 N-컷 문제에도 적용이 가능하며, 다른 학자들도 Kernighan-Lin 방법보다 부분적으로 향상된 알고리즘을 제시한 적이 있다 [6, 10, 16, 18]. 그러나 이들은 $N=2$인 경우에 대해서 향상을 보이고, N이 커질수록 성능이 훨씬 떨어진다 [11]. 게다가, 이들은 모두 부하 균등치는 제약 조건으로 주어지고, 단지 N-컷 값만을 최소화하는 알고리즘들이다. 따라서, 다음과 같은 두 가지 이유로 인하여, 그들 알고리즘으로는 부하 균등과 IPC 최소화 사이의 절충을 취하기가 매우 어렵다. 첫째, 주어진 부하 균등치 조건만 만족하면 더 이상 부하 균등을 고려하지 않는다. 따라서, 부하 균등 조건이 느슨할 경우, 상당히 부하가 균등되고 IPC 비용이 조금 더 큰 할당과 부하가 불균등되고 IPC 비용이 조금 더 작은 할당 중 후자를 선택하게 된다. 둘째로, 부하 균등 조건이 만족되지 않은 할당들은 전혀 고려하지 않는다는 점이다. 그러므로, 균등 조건이 엄격할 경우, 이 조건때문에 IPC 비용이 훨씬 큰 할당을 선택할 수 밖에 없다. 그러나, 타스크 할당 문제에 있어서, 부하 균등 및 IPC 최소화는 모두 추구되어야 하는 목적들이고, 따라서 동시에 고려할 수 있어야 한다. 또한 시스템 설계자는 주어진 타스크 부류에 따라 이들 사이에서 적당한 절충을 취할 수 있어야 한다. 따라서 기존의 알고리즘들은 이와 같은 타스크 할당 문제에 효율적으로 적용될 수 없었다.

본 논문에서는 먼저 부하 분배에 대한 통계적 편차를 이용하여 부하 균등치를 정의하고, 부하 균등과 IPC 최소화라는 두 가지 상충되는 목적을 하나의 함수로 표현한 새로운 비용 함수를 제안한다. 이 비용

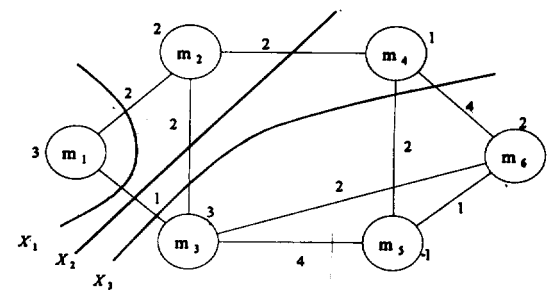
함수를 사용하여 주어진 타스크 부류에 따라 체계적으로 두 목적 사이에 절충을 취할 수 있다. 또한, 타스크 할당 문제가 최소 균등 N-컷 문제가 아닌 최소 N-컷 문제로 변형됨을 보인다. 참고로, 최소 균등 N-컷 문제는 두 가지 목적을 가지는 반면에, 최소 N-컷 문제는 하나의 목적만을 가지면서 부하 균등과 IPC 최소화를 동시에 취급한다. 본 논문의 주 공헌은 부하 균등과 IPC 최소화라는 두 가지 상충되는 목적들 사이의 상호 보완적인 관계를 체계적으로 연구할 수 있는 효율적인 휴어리스틱 알고리즘을 제시하는 것이다.

본 논문의 구성은 다음과 같다. 다음 절에서는 타스크 할당 문제의 모델에 대해서 기술하고, 제 3절에서는 부하 균등과 IPC 최소화라는 두 목적을 하나의 함수로 표현한 새로운 비용 함수를 제안한다. 이들 두 목적 사이에서 체계적으로 절충을 취하는 방법에 대해서 제 4절에서 설명하고, 제 5절에서는 타스크 할당 문제가 그래프 변환 기법을 사용하여 최소 N-컷 문제로 변환됨을 보인다. 제 6절에서는 이 문제에 대한 휴어리스틱 알고리즘을 제시하고, 마지막 두 절에서 제안된 알고리즘의 성능 평가를 하고 결론을 맺는다.

2. 타스크 할당 문제

본 논문에서 고려하는 시스템은 기능적으로 동일한 N개의 프로세서로 구성된 동종시스템이며, 타스크 T는 M개의 모듈 $m_1, m_2, \cdots, m_M$로 구성되며 이들은 각 프로세서 $P_1, P_2, \cdots, P_N$에 할당되어 수행된다. 모듈 m_i의 수행 비용을 R_i라고 하고, 타스크 T의 각 모듈들의 상호 관계는 모듈 상호 그래프(MIG : Module Interaction Graph)로 표현하며 여기에서 노드는 각 모듈을 의미하며, 두 모듈이 서로 통신을 하는 경우 해당하는 두 노드 사이에 엣지(edge)가 존재한다. MIG의 각 엣지 (m_i, m_j)는 엣지값 $W_{ij}(=W_{ji})$을 가지는데 이것은 두 모듈 m_i와 m_j가 서로 다른 프로세서에 할당되었을 경우 발생하는 통신 비용을 의미한다. 만약 두 노드 m_i와 m_j 사이에 엣지가 없을 경우에는 당연히 $W_{ij}=0$가 된다

타스크 할당 문제는 부하 균등을 취하는 동시에 IPC 비용이 최소인 할당을 찾는 문제이다. 그러나, 이들 두 최적화 목적은 서로 상충된다. (그림 1)은 2-프로세서 시스템에서의 MIG의 한 예를 보여준다. 여기



(그림 1) 2-프로세서 시스템에서의 예제 타스크 그래프의 세가지 타스크 할당

(Fig. 1) Three assignments on an example task graph in the two-processor system

에서 IPC를 최소화하기 위해서는 m_4이 P_1에 할당되고, 나머지 모듈들이 모두 P_2에 할당된, 따라서 전체 통신 비용이 3인, $X1$이 가장 좋은 할당이 된다. 그러나, 이 경우 프로세서 P_1과 P_2의 수행 비용이 각각 3과 9이므로 부하 불균등이 심하다. 부하 균등만을 위해서는 최적으로 부하가 균등된 할당 X_3을 선택하여야 한다. 그러나 X_3는 IPC 비용이 9이므로 X_1에 비해 너무 크다. 그러나 X_3에서 P_1에 할당된 m_4를 P_2로 옮길 경우, 즉 할당 X_2, 약간의 부하 불균등으로 IPC 비용을 5로 낮출 수가 있다. 최적의 부하 균등만을 고집하지 않을 경우에는 할당 X_2를 선택하는 것이 바람직하다. 따라서, 시스템 설계자는 수행할 타스크 종류에 따라 이들 두 상충되는 목적 사이에서 적당한 절충을 취할 수가 있어야 한다.

3. 비용 함수

MIG의 각 모듈들의 N개 프로세서에 대한 할당을 함수 $X: m_i \rightarrow P_{X(i)}$로 표현한다. 할당 X에 대한 IPC 비용을 $IPC(X)$라 하고 이것은 다음과 같이 계산할 수 있다:

$$IPC(X)=\sum_{k<l}\sum_{\substack{X(m_i)=P_k\\X(m_j)=P_l}} W_{ij},\ for\ 1\le k,l\le N,\ and\ 1\le i,j\le M.$$

할당하기 전의 각 프로세서의 초기 부하를 I_k라고 하자. 그러면, 임의의 할당에 대해서, 각 프로세서 P_k의

부하 L_k는 초기 부하 I_k와 P_k에 할당된 모듈들의 전체 수행 비용을 합한 값이 된다. 즉,

$$L_k = I_k + \sum_{X(m_i)=P_k} R_i, \; for \; 1 \le i \le M.$$

N개 프로세서의 전체 부하는 $\sum_{k=1}^{N} L_k$이다. 만약 이 부하가 각 프로세서에 균등하게 부과되었을 경우 각 프로세서 부하는 $\bar{L} = L/N$이 된다. 여기에서 L과 $\bar{L}$는 타스크 할당에 관계없이 정해지는 상수 값들이다.

각 할당 X에 대해서, $S^2(X) = \sum_{k=1}^{N} \frac{(L_k - \bar{L})^2}{N} = \sum_{k=1}^{N} \frac{L_k^2}{N} - \bar{L}^2$ 는 그 할당에서의 부하 분포에 대한 편차(variance)이다. 즉,

$$\begin{aligned} S^2(X) &= \sum_{k=1}^{N} \frac{L_k^2}{N} - \bar{L}^2 \\ &= \frac{1}{N}\left(L^2 - 2\sum_{k<l} L_k \cdot L_l\right) - \left(\frac{L}{N}\right)^2 \\ &= \frac{N-1}{N^2} L^2 - \frac{2}{N} \sum_{k<l} L_k \cdot L_l. \end{aligned}$$

여기에서 첫째 항 $(N-1)L^2/N^2$는 주어진 타스크에 대한 상수이고, 두번째 항인 $(2/N)\sum_{k<l} L_k \cdot L_l$은 타스크 할당에 따라 0과 $(N-1)L^2/N^2$ 사이의 값을 가지게 된다. 따라서 다음의 부등식이 성립한다:

$$0 \le S^2(X) \le \frac{(N-1)}{N^2} L^2, \; for \; all \; X.$$

부하 분포의 편차는 부하가 프로세서들 사이에 얼마나 균등하게 분포하고 있는가를 나타낸다. 즉, 편차가 작을수록 부하 분포가 균등함을 의미한다. 편차 $S^2(\cdot)$는 모든 모듈들이 하나의 프로세서에 할당되었을 때 최대값을 가진다. 또한 모든 프로세서의 부하가 균등할 때 $S^2(\cdot)$는 최소값을 가지며, 이때 시스템은 최적으로 부하-균등되었다고 한다. 따라서, 편차 $S^2(X)$을 할당 X의 부하 균등 정도를 나타내 주는 척도로 사용할 수 있다. 간소화를 위해, 본 논문에서는, 할당 X의 부하 균등 척도(degree of load balancing)를 $LB(X)$로 표현하고 다음과 같이 정의한다:

$$LB(X) = 1 - \frac{S^2(X)}{\frac{N-1}{N^2} L^2}$$

$$= \frac{2N}{(N-1)L^2} \sum_{k<l} L_k \cdot I_l.$$

참고로, $LB(X)$는 항상 부등식 $0 \le LB(X) \le 1$을 만족한다.

본 논문에서는 $IPC(\cdot)$와 $S^2(\cdot)$을 사용하여 다음과 같은 비용 함수를 제안한다:

$$\begin{aligned} COST(X) &= IPC(X) + \alpha \cdot S^2(X) \\ &= IPC(X) + \alpha \cdot (1 - LB(X)) \frac{N-1}{N^2} L^2, \end{aligned}$$

여기에서, 가중치 α는 시스템 설계자에 의해 정해지는 비음수(nonnegative) 값이다. 가중치 α를 적당한 값으로 설정함으로써 두 상충되는 최적화 목적 사이에서 절충을 취할 수 있다. 예를들어, 부하 균등에 보다 비중을 두고자 할 경우에는 α를 보다 큰 값으로 설정하면 된다. 반대로, α 값을 낮게 설정함으로써, 부하 균등보다 IPC에 보다 비중을 둘 수 있다. 다음 절에서는 타스크 부류에 따라 α 값을 체계적으로 설정하는 방법에 대해서 다룬다.

4. 가중치

제안된 비용 함수 상에서 α 값을 적당히 설정함으로써 부하 균등과 IPC 비용 사이에서 절충을 할 수 있다. 타스크 할당 문제에는 다음과 같은 두가지 극단적인 경우가 있을 수 있다. 하나는 부하 균등을 전혀 고려하지 않고 IPC 비용을 최소화하는 목적만을 가지고 할당하는 것이고, 다른 하나는 부하 균등이 최대가 되게 할당하는 것이다. 이들 두 경우는 통신-제한적 타스크와 그리고 계산-제한적 타스크인 경우에 각각 발생할 수 있다. 첫번째 경우에는 제안된 비용 함수 상에서 α 값을 0으로 설정함으로써 그 목적을 표현할 수 있다. 그러나 두번째 경우에서는 최대 부하 균등을 얻을 수 있는 최소 값을 예측하여야 한다. 다음 정리는 최대 부하 균등을 보장하는 α 값에 대한 조건을 제시한다.

정리 1: α가 $(N/2) \cdot \sum_{i<j} W_{ij}$ 보다 더 큰 값으로 설정할 경우 최적 할당 X는 부하 균등이 최대이다.

증명: X가 부하 균등이 최대가 아니고, 또다른 할당 X^*가 부하 균등이 최대라고 가정하자. 또한 L_k와 L_k^*를 각각 할당 X와 X^*에서의 프로세서 P_k의 부하라고 하자. 그러면, $LB(X^*) > LB(X)$이다. 다시말해서, 모든 수행 비용은 양수이므로 다음 부등식이 성립는 양수 값 p가 적어도 하나 존재한다:

$$\sum_{k<l} L_k^* \cdot L_l^* \sum_{k<l} L_k \cdot L_l + p,$$

또한 X는 최적 할당이므로 다음 부등식이 성립한다:

$$COST(X) = IPC(X) + \alpha \cdot \left(\frac{N-1}{N^2} L^2 - \frac{2}{N} \sum_{k<l} L_k \cdot L_l\right) \quad (1)$$

$$\leq IPC(X^*) + \alpha \cdot \left(\frac{N-1}{N^2} L^2 - \frac{2}{N} \sum_{k<l} L_k^* \cdot L_l^*\right) \quad (2)$$

$$\leq IPC(X^*) + \alpha \cdot \left(\frac{N-1}{N^2} L^2 - \frac{2}{N} \left(\sum_{k<l} L_k \cdot L_l + p\right)\right). \quad (3)$$

식 (1)과 (3)을 비교하면 다음 부등식을 얻을 수 있다:

$$p \cdot \frac{2\alpha}{N} \leq IPC(X^*) - IPC(X)$$

$$\leq \sum_{i<j} W_{ij}.$$

(마지막 부등식은 임의의 할당 X'에 대해 $0 \leq IPC(X') \leq \sum_{i<j} W_{ij}$이라는 사실에 기인한다.) 이것은 p가 양수이므로, $\alpha > N/2) \cdot \sum_{i<j} W_{ij}$이라는 사실과 모순이 된다. 따라서 X는 부하 균등이 최대이다.

정리 1에 의해서, α를 0과 $(N/2) \cdot \sum_{i<j} W_{ij}$ 사이의 값으로 설정함으로써 부하 균등과 IPC 사이의 절충을 취할 수 있다. 가중치 α 값에 대한 최적 할당을 X_o^α라고 하자. 그러면, 다음 정리에 의해서 $LB(X_o^\alpha)$와 $IPC(X_o^\alpha)$는 모두 α 값의 증가에 대한 비감소(non-decreasing) 함수이다.

정리 2: 모든 $\alpha < \alpha'$에 대해서 $LB(X_o^\alpha) \leq LB(X_o^{\alpha'})$과 $IPC(X_o^\alpha) \leq IPC(X_o^{\alpha'})$이 성립한다.

증명: X_o^α는 가중치 α에 대한 최적 할당이므로 다음 부등식이 성립한다:

$$IPC(X_o^\alpha) + \alpha(1 - LB(X_o^\alpha)) \frac{N-1}{N^2} L^2 \leq IPC(X_o^{\alpha'})$$

$$+ \alpha(1 - LB(X_o^{\alpha'})) \frac{N-1}{N^2} L^2. \quad (4)$$

가중치 α'에 대해서도 마찬가지로 다음 부등식이 성립한다:

$$IPC(X_o^{\alpha'}) + \alpha'(1 - LB(X_o^{\alpha'})) \frac{N-1}{N^2} L^2 \leq IPC(X_o^\alpha)$$

$$+ \alpha'(1 - LB(X_o^\alpha)) \frac{N-1}{N^2} L^2. \quad (5)$$

식 (4)와 (5)의 왼쪽과 오른쪽 항을 각각 더하면, 다음 부등식을 얻는다:

$$(\alpha - \alpha')(LB(X_o^{\alpha'}) - LB(X_o^\alpha)) \leq 0.$$

따라서, $\alpha < \alpha'$이므로 $LB(X_o^\alpha) \leq LB(X_o^{\alpha'})$이다. 또한 식 (4)를 다음과 같이 정리할 수 있다:

$$\alpha(LB(X_o^{\alpha'}) - LB(X_o^\alpha)) \frac{N-1}{N^2} L^2 \leq IPC(X_o^{\alpha'}) - IPC(X_o^\alpha).$$

따라서, $LB(X_o^\alpha) \leq LB(X_o^{\alpha'})$이므로 $IPC(X_o^\alpha) \leq IPC(X_o^{\alpha'})$이다.

최적 할당의 부하 균등치는 α가 증가할수록 최대 부하 균등이 될때까지 계속 증가를 하게된다. 최적 할당이 최대 부하 균등이 되는 최소 가중치 값을 $\hat{\alpha}$라고 하자. 본 논문에서는 $\hat{\alpha}$를 가중치의 "최상값(upper-bound)"이라고 부른다. 그러면 정리 2에 의해서, 임의의 양수 ε과 θ에 대해, $LB(X_o^{\hat{\alpha}-\varepsilon}) < LB(X_o^{\hat{\alpha}}) = LB(X_o^{\hat{\alpha}+\theta})$이다. 또한 정리 1의 가중치 값에 대한 조건은 충분 조건이므로, $\hat{\alpha} \leq (n/2) \cdot \sum_{i<j} W_{ij}$이다.

추론 1: 임의의 양수 ε, θ에 대해서, $IPC(X_o^{\hat{\alpha}-\varepsilon}) < IPC(X_o^{\hat{\alpha}}) = IPC(X_o^{\hat{\alpha}+\theta})$이다.

최적 할당의 IPC 비용이나 부하 균등치는 모두 가중치 α가 최상값 $\hat{\alpha}$가 될 때 최대값에 도달한다. 가중치 α가 최상값 $\hat{\alpha}$이하로 떨어지면 $IPC(X_o^\alpha)$나 $LB(X_o^\alpha)$도 함께 떨어지게 된다. 다시 말해서, α가 작아질수록 IPC 비용 면에서는 이득을 보게 되고, 반면에 부하 균등 면에서는 손실을 보게 된다.

정의 1: 할당 X에 대한 IPC 이득은 $G_{ipc}(X)$으로 표현

하고, 다음과 같이 정의한다:

정의 2: 할당 X에 대한 부하 균등 손실은 $L_{lb}(X)$으로 표현하고, 다음과 같이 정의한다:

$$L_{lb}(X)=1-LB(X).$$

α를 $\hat{\alpha}$보다 작은 값으로 설정함으로써, IPC 비용 면에서는 이득을 보고 부하 균등 면에서는 손실을 보게 된다. 이들 둘 사이에서 절충을 취하기 위해서는 주어진 타스크 부류에 따라 그들 사이의 비중을 따져서, 비용 함수에 반영할 수 있어야 한다. 비용 함수가 부하 균등에 비해 IPC 비용에 δ배의 비중을 더 둘 경우 이 비용 함수를 "δ-정책"이라고 부른다. 이것은 다음 정의 3에서 보다 형식적으로 정의된다.

정의 3: 비용 함수가 다음과 같은 두 가지 조건을 만족할 때 δ-정책이라고 부른다.

$$COST(X_1)<COST(X_2)\ if\ G_{ipc}(X_1)-\delta\cdot L_{lb}(X_1)>G_{ipc}(X_2)-\delta\cdot L_{lb}(X_2),$$
$$COST(X_1)=COST(X_2)\ if\ G_{ipc}(X_1)-\delta\cdot L_{lb}(X_1)=G_{ipc}(X_2)-\delta\cdot L_{lb}(X_2).$$

정리 3: 본 논문에서 제안한 비용 함수는 가중치를 다음과 같이 설정하면 δ-정책이 된다:

$$\alpha=\delta\cdot\frac{IPC(X_o^{\hat{\alpha}})}{\frac{N-1}{N^2}L^2}.$$

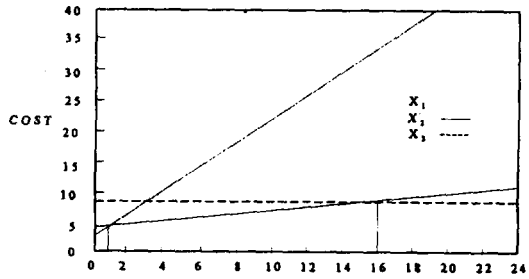
증명: 임의의 두 할당 X_1과 X_2에 대해서, 다음과 같다고 가정하자:

$$G_{ipc}(X_1)-\delta\cdot L_{lb}(X_1)>G_{ipc}(X_2)-\delta\cdot L_{lb}(X_2).$$

그러면,

$$\begin{aligned}&COST(X_1)-COST(X_2)\\&=IPC(X_1)-IPC(X_2)+\delta\cdot IPC(X_o^{\hat{\alpha}})\cdot(LB(X_2)-LB(X_1))\\&=IPC(X_o^{\hat{\alpha}})((G_{ipc}(X_2)-G_{ipc}(X_1))+\delta\cdot(L_{lb}(X_1)-L_{lb}(X_2)))\end{aligned}$$

따라서, $G_{ipc}(X_1)-\delta\cdot L_{lb}(X_1)=G_{ipc}(X_2)-\delta\cdot L_{lb}(X_2)$. 일 때, $COST(X_1)=COST(X_2)$이 된다. 그러므로, 제안된 비용 함수는 δ-정책이다.



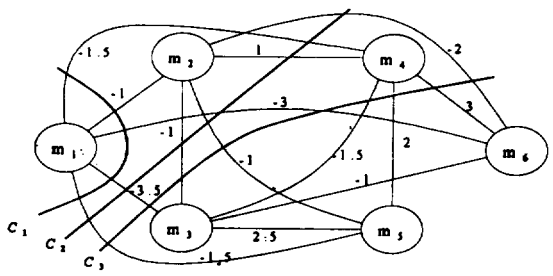
(그림 2) 그림 1의 할당들의 전체 비용
(Fig. 2) The total costs of assignments in Fig. 1

예를들어, (그림 1)의 각 할당 X_1, X_2, X_3의 IPC 이득은 각각 $\frac{6}{9}$, $\frac{4}{9}$, 그리고 0이다. 이들의 부하 균등 손실은 각각 $\frac{3}{4}$, $\frac{35}{36}$, 0이다. (이 예에서 $\hat{\alpha}=4$이고 $X_o^{\hat{\alpha}}=X_3$이다.) 그러므로, 정책에 따라서 우리는 서로 다른 할당을 선택해야 한다. (그림 2)는 δ값 증가에 따른 각 할당들의 전체 비용이 어떻게 변하는 지를 보여주고 있다. 이 그림에서 보듯이, $1\le\delta\le16$일 때 X_2가 가장 좋은 할당이 된다. 최적 할당 $X_o^{\hat{\alpha}}$을 찾는 문제는 최소 N-컷 문제와 [11] 동일하므로, NP-complete이다. 그러나 이 문제에 대한 여러가지 휴어리스틱 알고리즘들이 많이 제안되어 있으므로, 이들을 사용하여 최적에 가까운 할당을 찾을 수 있다. 본 논문에서는 이들 중 가장 효율적인 참고 문헌 [11]에서 제안한 알고리즘을 사용하며, 이에 대해서는 제 6절에서 설명한다.

5. 문제 변환

그래프 G에서의 N-컷이란, 엣지들의 집합으로서 이들을 제거할 경우 그 그래프의 모든 노드들이 N개의 서로 다른 부집합 $P_1, P_2, \cdots, P_N$으로 분리되는 엣지 집합이다. N-컷의 값은 그 집합에 속하는 모든 엣지들의 값의 합이다. 각 부집합 P_k의 크기는 P_k에 있는

모든 노드들의 크기의 합이다. 주어진 타스크의 MIG에서의 각 N-컷들과 타스크 할당들 사이에는 일대일 대응 관계가 성립한다. 즉, N-컷에 의해 노드들이 N개의 부집합 $P_1, P_2, \cdots, P_N$으로 분리될 경우, 각 부집합 P_i에 있는 각 노드들은 (즉, 각 모듈들은) 프로세서 P_i에 할당된다. 그리고, N-컷의 값은 IPC 비용과 같고, N개의 부집합들의 크기 분포에 대한 부하 균등치는 바로 대응하는 할당의 부하 균등치와 같다. 따라서, 본 논문에서의 타스크 할당 문제는 그래프 상에서 부집합들의 크기를 균등하게 함과 동시에 최소값을 가지는 N-컷을 찾는 문제와 일치하게 된다.



(그림 3) 그래프 변환 예
(Fig. 3) An example of graph modification

본래의 MIG 그래프에서의 각 N-컷에는 부집합들의 크기에 대한 정보가 전혀 없다. 따라서, N-컷의 값과 동시에 부집합들의 부하 균등치를 고려하여야 하므로, 본래의 그래프 상에서 효율적인 휴어리스틱 알고리즘을 고안하는 것이 상당히 어렵게 된다. 예를들어, Kernighan-Lin의 알고리즘은 [9] 처음에 균등 분할에서 출발하여 크기가 같은 노드들끼리 쌍-교환 (pairwise-exchange) 방법을 사용하여 부하 균등을 유지하면서 컷 값을 줄여나간다. 다시 말해서 Kernighan-Lin의 알고리즘은 IPC와 부하 균등을 서로 다른 차원에서 처리를 하기때문에 이들 둘 사이의 절충을 취하는 것이 거의 불가능하다. 그러나 그래프를 변형하여 각각의 엣지가 원래의 엣지 값 (즉, IPC 비용) 뿐만아니라 대응하는 두 노드들의 크기에 대한 정보까지 (즉, 수행 비용) 가지게 한다면, 변형된 그래프의 각 N-컷은 대응하는 할당의 IPC 비용 및 수행 비용에 대한 정보가 다 포함되므로 이들 두 상충하는 목적사이의 절충을 취할 수 있는 효율적인 알고리즘을 보다 쉽게 고안할 수 있다. 설명을 간단히 하기 위해서 모든 프로세서들의 초기 부하는 0으로 한다. (즉, $I_k=0$, $\forall\ 1 \le k \le N$.) 그러나 결론에서 언급되겠지만, 이 제한은 쉽게 $I_k > 0$으로 확장될 수 있다.

본 논문에서는 다음과 같이 MIG 그래프를 변형하여 각 N-컷이 IPC 비용 및 수행 비용에 대한 정보를 다 가지도록 하는 그래프 변형 방법을 먼저 제안한다. 아래 정리 4에서 보인 바와 같이, 타스크 할당 문제를 최소 N-컷 문제로 변형하기 위해 임의의 모든 두 노드 사이의 엣지값에서 $-R_i \cdot R_j \cdot (2\alpha/N)$값을 뺀다. 즉, MIG의 임의의 두 노드 m_i와 m_j에 대해서:

만약 그들 사이에 엣지가 존재하면, 그 엣지값을 $W_{ij} - R_i \cdot R_j \cdot (2\alpha/N)$으로 만들고, 그렇지 않으면 그들 사이에 엣지값 $-R_i \cdot R_j \cdot (2\alpha/N)$을 가지는 새로운 엣지를 만든다. 여기에서, $\alpha = \delta \cdot IPC(X_o^{\hat{a}})/(\frac{N-1}{N^2})$이다.

예를들어, (그림 1)의 MIG에 대해서 2-정체으로 변형된 그래프 G가 (그림 3)에 나타나 있다. 이 예에서, $IPC(X_o^{\hat{a}})=9$, 즉 $\alpha=0.5$이다. (그림 3)의 각 2-컷 C_1, C_2, C_3는 (그림 1)의 할당 X_1, X_2, X_3에 대응한다. 변형된 그래프 G 상에서의 각 N-컷은 대응하는 할당의 전체 비용에 대한 정보를 다 가지고 있으므로, 본 논문에서 고려하는 타스크 할당 문제는 그래프에서 최소값 N-컷을 찾는 문제로 (즉, 최소 N-컷 문제) 변형이 된다.

정리 4: 타스크 할당 문제는 최소 N-컷 문제로 변형이 된다.

증명: 변형된 그래프 G에서의 하나의 N-컷 C가 할당 X에 대응한다고 하자. 또한 N-컷 C가 G의 노드들을 N개의 부집합 $P_1, P_2, \cdots, P_N$으로 분리한다고 하자. 각 부집합 P_k는 $\{m_{k_i} : 1 \le i \le p_k\}$라고 한다. 그러면, C의 값 $W(C)$는 다음과 같다:

$$
\begin{aligned}
W(C) &= \sum_{k<l} \left| \sum_{\substack{1 \le i \le p_k \\ 1 \le j \le p_l}} \left| W_{k_i l_j} - R_{k_i} \cdot R_{l_j} \cdot \frac{2\alpha}{N} \right| \right| \\
&= \sum_{k<l} \sum_{\substack{1 \le i \le p_k \\ 1 \le j \le p_l}} W_{k_i l_j} - \left(\sum_{k<l} L_k \cdot L_l \right) \cdot \frac{2\alpha}{N} \\
&= IPC(X) - \left(\sum_{k<l} L_k \cdot L_l \right) \cdot \frac{2}{N} \cdot \alpha
\end{aligned}
$$

$$= COST(X) - \frac{N-1}{N^2} L^2 \cdot \alpha.$$

마지막 식의 두번째 항은 각 타스크마다 정해지는 상수이다. 따라서, 제안된 그래프 병형 방법을 사용하여, 타스크 할당 문제가 그래프에서 최소값 N-컷을 찾는 문제로 변형된다. 예를들어, (그림 4)의 C_1, C_2, C_3의 값은 각각 -10.5, -12.5, -9.0이고, 따라서 C_2가 최소값 2-컷이 된다. 이들과 대응하는 할당의 전체 비용은 다음과 같다:

$$COST(X_1) = W(C_1) + \frac{N-1}{N^2} L^2 \cdot \alpha = -10.5 + 36 \times 0.5 = 7.5,$$

$$COST(X_2) = W(C_2) + \frac{N-1}{N^2} L^2 \cdot \alpha = -12.5 + 36 \times 0.5 = 5.5,$$

$$COST(X_3) = W(C_3) + \frac{N-1}{N^2} L^2 \cdot \alpha = -9.0 + 36 \times 0.5 = 9.$$

따라서, (그림 2)에서 보듯이, $\delta = 2$일 경우 2-컷 C_2에 대응하는 X_2가 가장 좋은 할당이 된다.

6. 할당 알고리즘

δ가 주어지면 가중치 α 값을 결정하기 위해서 먼저 X_o^{δ}를 찾아야 한다. 그러나 그래프에서 이와같은 최적 할당을 찾는 문제는 NP-complete 문제이므로, 휴어리스틱 알고리즘을 사용하여야 한다. 또한, 앞 절에서도 언급한바와 같이, 이 문제는 가중치 α를 최상값인 로 $\hat{\alpha} = (N/2) \cdot \sum_{i<j} W_{ij}$로 설정하고 제안한 그래프 변형을 통해 최소 N-컷 문제로 변환이 된다. 그래프에서 최소 N-컷을 찾는 방법으로 본 논문에서는 참고문헌 [11]에서 제안된 휴어리스틱 알고리즘을 사용하고자 한다. 이 알고리즘에 대한 자세한 설명은 생략하고 간단하게 동작 원리를 설명하면 다음과 같다.

참고문헌 [11]의 N-컷 문제에 대한 기본적인 접근 방법은 먼저 임의의 할당에서 시작하여, 각 프로세서에서 하나의 노드를 선택하여 다른 프로세서로 옮김으로써 보다 개선된 할당을 찾는 작업을 되풀이 하는 것이다. 옮길 노드의 선택은 그 노드를 옮김으로써 N-컷의 값이 최대로 줄어드는 (줄어드는 것이 불가능할 때에는 최소로 늘어나는) 노드를 하나 선택한다. 이 알고리즘은 연속적인 반복 구간으로 구성이 되며, 각 반복 구간에서는 모든 노드들이 정확히 한번씩 옮겨진다. 즉, 각 반복 구간에서 옮겨질 노드는 그 구간에서 아직 옮겨지지 않은 노드들 중에서 하나 선택된다. 각 반복 구간 동안에 M개의 새로운 할당이 얻어지며, 이들 중 가장 N-컷의 값이 최소인 할당을 선택하여, 그 다음 반복 구간의 초기 할당으로 사용한다. N-컷의 값에 더 이상 개선이 없을 때까지 이와같은 반복이 계속된다.

참고문헌 [11]과 본 논문의 차이는 [11]에서는 가중치 값을 최상값 이상으로 설정하여 최적 균등 할당을 찾는 문제를 다루고, 본 논문에서는 [11]의 알고리즘을 사용하여 IPC를 최소화하는 것과 부하 균등이라는 두가지 상충되는 목적 사이에서 체계적으로 최적의 절충을 취하는 방법을 다룬다. 참고 문헌 [11]의 알고리즘은 MINCUT이라고 부르고 M개의 노드로 구성된 그래프 G를 입력으로 받고, 크기 M인 1차원 배열 ASSIGN[1..M]을 출력한다. 여기에서 각 노드 (즉 모듈) m_i는 프로세서 $P_{ASSIGN[i]}$에 할당된다. 알고리즘 MINCUT의 시간 복잡도는 $O(NM^2)$이다.

본 논문에서는 알고리즘 MINCUT을 이용하여 두 상충되는 목적들 사이의 절충을 취한 (즉, δ-정책) 할당을 찾는 알고리즘인 ASSIGNMENT를 (그림 4)와 같이 제안한다. 이 알고리즘은 타스크 그래프인 MIG와 δ 값을 입력으로 받아서 δ-정책 할당을 출력한다.

```
알고리즘 ASSIGNMENT

입력 : 그래프 MIG, δ
출력 : ASSIGN[1..M] // 모듈 m_i 는 프로세서 P_ASSIGN[i] //
변수
    Temp[1..N] : integer; // assignment X_o^δ //
    Ipc : integer; // IPC(X_o^δ), initially zero //

1)  modify graph MIG to G' with α = (N/2)·Σ_{i<j} W_ij

2)  Temp[1..M] = MINCUT(G'); // find X_o^δ //
3)  for i := 1 to N-1 do // calculate IPC(X_o^δ) //
      for j := i+1 to N do
        If Temp[i] ≠ Temp[j] then Ipc = Ipc + c_ij;

4)  modify graph MIG to G with α = δ·Ipc/((N-1)/N^2 L^2);

5)  ASSIGN[1..M] = MINCUT(G); // find the final assignment //
end. // exit with the final assignment ASSIGN[1..M] //
```

(그림 4) 알고리즘 ASSIGNMENT
(Fig. 4) The algorithm ASSIGNMENT

여기에서 가중치 δ 값을 결정하기 위해 먼저 $X_{\hat{\alpha}}^{*}$를 찾아야 하는데, 이것은 (그림 4)에서와 같이, 가중치를 $\hat{\alpha}$로 설정한 상태에서 MIG를 변형한 그래프인 G'상에서 알고리즘 MINCUT을 이용하여 최소 N-컷을 찾으면 된다. 가중치 α 값이 결정되면, 이 값으로 다시 변형된 그래프 G에서 최소 N-컷을 찾으면 이것이 최종 할당이 된다. (그림 5)에서 보듯이, ASSIGNMENT는 MINCUT을 정확히 두번 부르며, 따라서 이 알고리즘의 시간 복잡도 역시 $O(NM^2)$이다.

7. 실험 결과

제안한 할당 알고리즘 ASSIGNMENT의 성능을 측정하기 위하여 여러가지 임의 타스크 그래프에 대해서 프로세서 갯수 N이 2, 4, 10인 경우에 대해서 Kernighan-Lin 알고리즘과 비교 분석하였다. Kernighan-Lin 알고리즘을 포함하여 지금까지 제안된 것들은 IPC 최소화와 부하 균등이라는 두 가지 상충된 목적 사이에서 절충을 취하는 것이 아니고, 주어진 부하 균등치의 범위 안에서 IPC가 최소인 할당을 찾는 알고리즘들이다. 따라서 공정한 비교를 위해서 크기가 동일한 두 노드들 만이 교환 대상이 되는 Kernighan-Lin 알고리즘의 쌍-교환 방법을 다음과 같이 수정하였다. 서로 크기가 다른 두 노드도 교환했을 때 주어진 부하 균등치의 범위를 만족할 경우에는 쌍-교환 대상이 될 수 있다. 임의의 두 부집합 사이의 크기 차이의 최대 허용치를 부하 균등치 조건으로 주어지며, 이것을 "*tolerance*"라고 부른다. 임의의 두 부집합 사이의 크기 차이는 *tolerance*보다 같거나 작아야 한다. (즉, $|L_k - L_l| \leq tolerance$, $\forall\ k, l$.) 프로세서가 각각 2, 4, 10 개인 경우에 대해서 임의의 타스크 그래프를 대상으로 실험을 하였다. 또한 특정한 양수 값 r_s에 대해서 타스크 그래프의 각 노드의 크기를 0과 r_s 사이에서 임의로 설정하였으며 $r_s = 3, 6, 9$인 세 가지 경우에 대해서 실험하였다.

본 논문에서 제안한 δ-정책과 ($\delta = 1, 2, 5$) Kernighan-Lin 알고리즘을 비교하였으며, 각 실험마다 100개의 노드와 엣지값이 1인 약 1500 개의 엣지로 구성된 100개의 임의 그래프를 대상으로 하여 실험하였고, 그 평균치를 〈표 1-3〉에 정리하였다. 여기에서 $diff_{\max}$는 임의의 두 부집합 사이의 최대 크기 차이이다, 즉,

〈표 1〉$N = 2$일 때의 실험 결과
〈Table 1〉 Experimental results for $N = 2$

r_s	δ	ASSIGNMENT			Kernighan-Lin's		
		Average IPC(·)	Average LB(·)	Average $\text{Diff}_{\max}$	Average IPC(·)	Average LB(·)	Average $\text{Diff}_{\max}$
3	1	318.35	0.668	111.80	358.40	0.668	111.80
	2	529.23	0.968	35.21	550.20	0.968	34.95
	5	581.41	0.997	9.45	600.50	0.997	9.35
6	1	324.29	0.734	178.24	371.55	0.735	178.10
	2	499.23	0.964	66.48	535.30	0.964	66.20
	5	562.24	0.997	19.36	574.40	0.997	19.10
9	1	324.85	0.740	261.63	365.85	0.742	260.95
	2	498.35	0.961	97.55	529.75	0.962	97.55
	5	558.19	0.996	28.83	578.15	0.997	27.90

〈표 2〉$N = 4$일 때의 실험 결과
〈Table 2〉 Experimental results for $N = 4$

r_s	δ	ASSIGNMENT			Kernighan-Lin's		
		Average IPC(·)	Average LB(·)	Average $\text{Diff}_{\max}$	Average IPC(·)	Average LB(·)	Average $\text{Diff}_{\max}$
3	1	563.60	0.720	109.18	634.10	0.673	110.10
	2	897.25	0.967	38.79	946.95	0.952	38.80
	5	987.75	0.998	9.35	1002.35	0.997	10.35
6	1	540.85	0.732	187.38	633.75	0.694	188.30
	2	864.39	0.962	70.52	938.75	0.945	71.40
	5	955.80	0.998	20.48	991.90	0.995	21.45
9	1	558.73	0.750	259.28	650.05	0.699	258.50
	2	826.85	0.953	103.32	913.10	0.944	103.05
	5	953.28	0.997	31.50	991.75	0.995	31.40

$diff_{\max} = \max_{k,l} |L_k - L_l|$ 각 임의 그래프에 대해서, ASSIGNMENT를 사용하여 할당을 먼저 구하고, 이 할당에서의 두 부집합 사이의 최대 크기 차이에 1을 더한 값을 Kernighan-Lin 알고리즘의 tolerance로 설정하였다. 프로세서가 2개인 경우의 실험 결과는 〈표 1〉에 나타나 있으며, 여기에서 보듯이 ASSIGNMENT의 IPC 비용이 Kernighan-Lin 알고리즘 보다 평균 7% 이상 작음을 알 수 있다.

그러나, 각 그래프에 대해서 tolerance를 ASSIGN-

〈표 3〉$N = 10$일 때의 실험 결과
〈Table 3〉 Experimental results for $N = 10$

r_s	δ	ASSIGNMENT			Kernighan-Lin's		
		Average IPC(·)	Average LB(·)	Average $Diff_{max}$	Average IPC(·)	Average LB(·)	Average $Diff_{max}$
3	1	552.20	0.687	118.10	557.90	0.533	118.20
	2	1034.70	0.962	40.25	1035.60	0.903	41.75
	5	1179.25	0.999	7.95	1185.80	0.995	8.95
6	1	568.92	0.733	190.84	572.55	0.550	191.15
	2	1017.37	0.957	77.71	1019.30	0.894	78.70
	5	1138.21	0.997	21.69	1149.60	0.990	22.65
9	1	587.92	0.752	258.24	592.60	0.562	259.35
	2	982.49	0.954	111.23	1009.30	0.890	112.15
	5	1123.92	0.996	36.30	1150.40	0.987	37.30

MENT가 찾은 할당의 최대 크기 차이에 1을 더한 값으로 설정하였음에도 불구하고, Kernighan-Lin 알고리즘의 최대 크기 차이의 평균값이 ASSIGNMENT보다 조금 더 작다. 예를들어, $r_s=3$, $\delta=5$인 경우, ASSIGNMENT와 Kernighan-Lin 알고리즘의 평균 $diff_{max}$는 각각 9.45와 9.35이다. 이 경우 Kernighan-Lin 알고리즘에 주어진 평균 *tolerance*는 10.45이다. 이것은 Kernighan-Lin 알고리즘이 $3.95 < diff_{max} \leq 10.45$ 사이에서 더 좋은 할당을 찾지 못했다는 것을 의미한다. 프로세서가 4, 10개인 경우의 결과는 〈표 2, 3〉에 각각 나타나 있다. 이 표들에서 알 수 있듯이, ASSIGNMENT의 평균 IPC 비용이 Kernighan-Lin 알고리즘 보다 훨씬 작다. 평균 최대 크기 차이도 ASSIGNMENT가 Kernighan-Lin 알고리즘 보다 작다. 이와 같은 성능 차이는 부하 균등치 조건이 느슨할수록 (즉, δ 값이 클수록) 더 커진다. 결론적으로 본 논문에서 제안한 타스크 할당 방법이 (즉, IPC 최소화와 부하 균등 사이에 절충을 취하는 방법) 기존의 할당 방법보다 성능이 우수함을 알 수 있다.

8. 결 론

본 논문에서는 타스크 할당에 있어서 추구 해야할 두 가지 상충 되는 목적 (즉, IPC 최소화와 부하 균등) 사이에서 주어진 타스크 부류에 따라 체계적으로 절충을 취할 수 있는 타스크 할당 방법을 제안하였다. 이것은 부하 분포의 통계적 편차를 사용하여 부하 균등치를 정의함으로써 가능하였으며, 이를 이용하여 두 목적 사이의 상호 보완 관계를 하나의 함수로 표현한 새로운 비용 함수를 제안하였다. 또한, 상충되는 두 목적 사이에서 타스크 부류에 따라 체계적으로 절충을 취할 수 있는 δ-정책을 제안하였다. 본 논문에서 다루는 타스크 할당 문제는 최소 N-컷 문제로 변환이 가능함을 증명하였고, 여러 종류의 타스크 그래프에 대한 실험에서 제안한 알고리즘이 기존의 할당 알고리즘들에 비해 성능이 우수함을 보였다.

본 논문에서는 각 프로세서의 초기 부하가 0이라고 가정하였다, 즉, 모든 $1 \leq k \leq N$에 대해서 $I_k=0$. 그러나, 본래의 MIG 그래프를 수정하여 각 프로세서 P_k에 대해 크기가 $-I_k$인 모조 (dummy) 노드 d_k를 하나씩 추가함으로써, 이 가정은 $I_k > 0$으로 확장될 수 있다. 이 경우, 각 모조 노드 d_k는 P_k에 고정이 된다. 즉, 각 반복 구간에서 d_k는 P_k로 부터 다른 프로세서로 이동할 수 없다. 참고로, 어떤 프로세서 P_l의 초기 부하가 0인 경우에는 모조 노드 d_l을 고려하지 않아도 되는데, 그 이유는 이들 노드의 크기가 0이므로 N-컷 값 계산에 아무런 영향을 주지 않기 때문이다.

타스크 할당 문제는 메모리 제한 조건, 실시간 제한, 모듈들 간의 선행 관계 등과 같이 해결해야 할 문제들이 많이 남아 있다. 이들 각 문제들에 대한 연구가 많은 곳에서 진행이 되고 있지만, 본 논문에서 제안한 방법을 확장하여 이들 문제들을 해결하는 것도 흥미있는 연구가 될 것이다.

참 고 문 헌

[1] S. H. Bokhari, "Dual processor scheduling with dynamic reassignment," *IEEE Trans. Software Eng.*, vol. SE-5, pp. 341-349, July 1979.

[2] S. H. Bokhari, "A shortest tree algorithm for optimal assignments across space and time in a distributed processor systems," *IEEE Trans. Software Eng.*, vol. SE-7, pp. 583-589, Nov. 1981.

[3] S. H. Bokhari, *Assignment Problems in Parallel and Distributed Computing*. Kluwer Academic

Publishers, Boston, 1987.

[4] S. H. Bokhari, "Partitioning problems in parallel, pipelined and distributed computing," *IEEE Trans. Comput.*, vol. C-37, pp. 48-57, Jan. 1988.

[5] W. W. Chu and K. K. Leung, "Module replication and assignment for real-time distributed processing systems," *IEEE Proc.*, vol. 75, pp. 547-562, May 1987.

[6] C. M. Fiduccia and R. M. Mattheyses, "A linear-time heuristic for improving network partitions," in *Proc. 19th Design Automation Conf.*, pp. 175-181, June 1982.

[7] L. R. Ford, Jr. and D. R. Fulkerson, *Flows in Networks*. Princeton, NJ: Princeton Univ. Press, 1962.

[8] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979.

[9] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell Syst. Tech. J.*, vol. 49, pp. 291-307, Feb. 1970.

[10] B. Krishnamurthy, "An improved min-cut algorithm for partitioning VLSI networks," *IEEE Trans. Comput.*, vol. C-33, pp. 438-446, May 1984.

[11] C. -H. Lee, C. -I. Park, and M. Kim, "Efficient algorithm for graph-partitioning problem using a problem transformation method," *Computer-Aided Design*, vol. 21, pp. 611-618, Dec. 1989.

[12] C. -H. Lee and Kang G. Shin, "Optimal task assignment in homogeneous systems," *IEEE Trans. Parallel and Dist. Syst.*, vol. 8, no. 2, pp. 119-129. Feb. 1997.

[13] C. -H. Lee, D. Lee, and M. Kim, "Optimal task assignment in linear array networks," *IEEE Trans. Comput.*, vol. C-41, pp. 877-880, July 1992.

[14] V. M. Lo, "Heuristic algorithms for task assignment in distributed systems," *IEEE Trans. Comput.*, vol. C-37, pp. 1384-1397, Nov. 1988.

[15] V. M. Lo, "Algorithms for static task assignment and symmetric contraction in distributed computing systems," *Proc. Int. Conf. on Parallel Processing*, pp. 239-244, 1988.

[16] T. Ng, J. Oldfield, and V. Pitchumani, "Improvements of a mincut partition algorithm," *Proc. Int. Conf. on Computer-Aided Design*, pp. 470-473, 1987.

[17] M. G. Norman and P. Thanisch, "Models of machines and computation for mapping in multicomputers," *ACM Computing Surveys*, vol. 25, pp. 263-302, Sep. 1993.

[18] L. A. Sanchis, "Multiple-way network partitioning," *IEEE Trans. Comput.*, vol. C-38, pp. 62-81, Jan. 1989.

[19] C. C. Shen and W. H. Tsai, "A graph matching approach to optimal task assignment in distributed computing systems using a minimax criterion," *IEEE Trans. Comput.*, vol. C-34, no. 3, pp. 197-203, Mar. 1985.

[20] H. S. Stone, "Multiprocessor scheduling with the aid of network flow algorithms," *IEEE Trans. Software Eng.*, vol. SE-3, pp. 85-93, Jan. 1977.

[21] H. S. Stone, "Critical load factors in distributed computer systems," *IEEE Trans. Software Eng.*, vol. SE-4, pp. 254-258, May 1978.

[22] H. S. Stone and S. H. Bokhari, "Control of distributed processes," *IEEE Computer*, vol. 11, pp. 97-106, July 1978.

김 주 만

1984년 숭실대학교 전자계산학과(공학사)

1997년 충남대학교 대학원 컴퓨터공학과(석사과정)

1995년~1996년 미국 노벨 및 SCO사 국제공동연구 파견근무

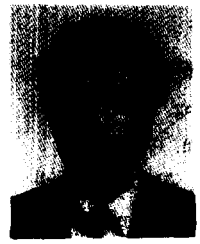
1985년~현재 ETRI 컴퓨터구조연구실 책임연구원

1994년 정보처리 기술사(전자 계산기 조직 응용)

관심분야: 운영체제, 병렬 처리, 결함 허용 시스템

박 치 항

1974년 서울대학교 응용물리학과 졸업(이학사)
1980년 한국과학기술원 전산학과 졸업(이학석사)
1987년 파리 6대학 전산학과(공학박사)
1974년~1978년 한국과학기술연구소 연구원
1993년~1994년 미국 오레곤 주립대학교 객원교수
1978년~현재 한국전자통신연구원 컴퓨터 연구단장
관심분야: 멀티미디어, 분산시스템, 가상현실, 에이전트, 네트워크 가상 컴퓨팅



이 철 훈

1983년 서울대학교 전자공학과(공학사)
1988년 한국과학기술원 전기및 전자공학과(공학석사)
1992년 한국과학기술원 전기및 전자공학과(공학박사)
1983년~1986 삼성전자 컴퓨터 개발실발실 연구원
1992년~1994년 삼성전자 컴퓨터사업부 선임연구원
1994년~1995년 University of Michigan 객원연구원
1995년~현재 충남대학교 컴퓨터공학과 조교수
관심분야: OS, 병렬처리, 결함허용 및 실시간 시스템