선형계획법을 적용한 임의 분할 불가능한 부하 분배계획

Indivisible load scheduling applied to Linear Programming

손경호, 이달호[1] 김형중[2] , [1]경원대학교 [2]강원대학교
Kyung-Ho Son, Dal-Ho Lee[1] Hyoung-Joog Kim[2]
[1]Kyungwon Univ. [2]Kangwon Univ.

Abstract

There are many studies on arbitrarily divisible load scheduling problem in a distributed computing network consisting of processors interconnected through communication links. It is not efficient to arbitrarily distribute the load that comes into the system. In this paper, how to schedule in case that arbitrarily indivisible load comes into the system is studied. Also, the cases of the divisible load mixed with the indivisible load that come into network were dealt with optimal load distribution in parallel processing system by scheduling applied to linear programming.

1. 서 론

병렬분산처리시스템의 주요한 목적은 '주어진 부하를 네트워크에 연결된 각 프로세서에 얼마만큼 할당하여야 전체 계산을 되도록 짧은 시간에 마치게 할 수 있는가?' 이다. 이를 위해 네트워크상의 프로세서들이 가장 빠른 시간 안에 작업을 하기 위해 처리할 부하를 효과적으로 분배해 줄 센서가 필요하다[1]. [1]에서 센서가 부하를 분배하고, 각 프로세서가 처리하는데 소비되는 시간의 다이어그램(diagram)과 순환적인 부하분배 방정식을 표현하고 있다. [1]에서의 방법은 [2,3]에서의 1단계 트리 네트워크(single-level tree network)와 버스 네트워크(bus network)로 확장되어진다. [1-3]에서는 프로세서에 할당되는 부하는 모든 프로세서가 동시에 처리를 마친다는 가정 하에 얻어진다. 이 가정은 선형 네트워크에서 최적처리시간을 얻는데 충분한 조건과 필요성이 있다는 것을 보여준다[4]. 프로세서가 동등한 능력을 가지고 있다는 개념을 이용하여 버스네트워크에서 최적부하분배의 분석적 증명은 [5]에서 논의되었다. 그러나 그것은 '단지 제한된 의미에서만 사실이 된다는 것'을 증명한다[6]. 선형 네트워크의 경우, [7-9]에서 처리시간의 점근적 분석을 하였다. [6]에서 '프로세서의 처리능력과 각 프로세서를 연결하는 링크의 전송능력의 순열을 탁월한 순으로 하는 것이 전체 계산을 마치는데 소비되는 시간을 줄인다'라는 것을 증명하였다. 본 연구에서 프로세서와 링크의 순열을 고려하여 임의 분할 불가능한 부하와 가능한 부하의 스케줄링 문제를 다룬다. 이전에 보여진 것처럼, 임의 분할 가능한 부하는 거의 항상 최적으로 스케줄링이 될 수 있다. 그러나 임의 분할 불가능한 부하는 현재 최적으로 스케줄링하기 매우 어렵다. 그 이유는 이것은 NP-complete 문제이기 때문이다. 본 연구는 1단계 트리 네트워크에서 최적 스케줄링 조건을 유도한다. 이는 버스, 데이지 체인(daisy-chain) 네트워크 등, 다른 위상에도 확장가능하다. [10]의 선형계획법을 이용하여 여러개의 임의 분할 불가능한 부하를 어떤 프로세서에 할당하여야 네트워크가 전체 계산을 되도록 짧은 시간에 마치게 할 수 있는 지에 대한 방법을 알아본다.

2. 기호의 정의

본 논문의 부하 분배 계획시 자주 사용될 기호를 아래와 같이 정의한다.

- T_{cm} : 일정량의 부하와 하나의 링크를 기준으로 하여 부하를 전송하는데 소비되는 시간
- T_{cp} : 일정량의 부하와 하나의 프로세서를 기준으로 하여 부하를 처리하는데 소비되는 시간
- w_i : 프로세서 p_i가 주어진 부하를 처리할 때, 기준 처리시간(T_{cp})에 대한 처리하는데 소비되는 시간의 비
- z_i : 주어진 부하가 링크(link) i를 통해 전송될 때 기준 전송시간(T_{cm})에 대한 전송되는 시간의 비
- α_i : 프로세서 p_i에 할당되는 부하
- β_i : 프로세서 p_i에 할당되는 임의 분할 불가능한 부하
- γi : 프로세서 p_i에 할당되는 임의 분할 가능한 부하
- q_i : 임의 분할 불가능한 부하
- $T_{\alpha_i}(T_{\beta_i})$: 프로세서 p_i에 할당된 부하의 처리를 마치는 시간
- $T(T_\beta)$: 부하 분배가 최적화된 네트워크에서 전체 계산을 마치는 시간

3. 1단계 트리 네트워크 (single-level tree network)에서 부하 분배

1단계 트리 네트워크의 구조는 그림 1에서 보는 것과 같이 루트 프로세서(root processor) p_0에 자 프로세서(Child processor) p_1, p_2, …, p_m가 m개의 링크로 연결되어 있는 구조이다. 프로세서 p_0가 전체 부하를 받아 자신이 처리할 부하만을 남기고 나머지 부하는 프로세서 p_1, p_2, …, p_m에 적절히 분배한다. 각 프로세서는 할당받은 부하를 받은 즉시 처리를 시작한다. 주 프로세서 p_0는 전치계산기(front end processor) 가 있어 프로세서 p_1, p_2, …, p_m에 부하를 전송하는 시간이 프로세서 p_0가 부하를 처리하는 시간에 포함되지 않는다고 가정한다.

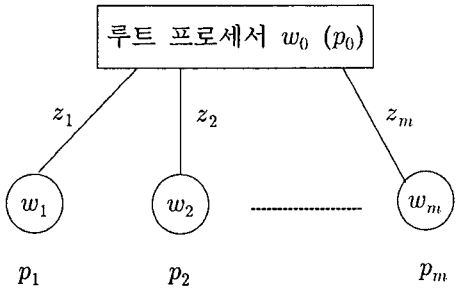

그림1. 1단계 트리 네트워크
(single-level tree network)

프로세서 p_i 가 부하의 처리를 마치는 시간은 식 (1)과 같다.

$$T_{\alpha_i} = \sum_{j=1}^{i} \alpha_j z_j T_{cm} + \alpha_i w_i T_{cp} \quad (i = 1, \cdots, m) \qquad (1)$$

(단, $T_{\alpha_0} = \alpha_0 w_0 T_{cp}$)

3.1. 임의 분할 불가능한 부하의 분배

목적함수와 제약조건을 아래와 같이 선형계획을 한다.

- 목적함수(Objective function) : Minimize $c^T X_\beta$ (2)

($c^T = [1\ 0\ 0\ \cdots\ 0]$, $X_\beta{}^T = [T_\beta\ \beta_0\ \beta_1\ \cdots\ \beta_m]$)

- 제약조건 : $AX_\beta \geq b$, $X_\beta \geq 0$ (3)

T_β는 네트워크가 전체 계산을 마치는 시간이므로 각 프로세서가 처리를 마치는 시간 중 최대값을 갖게 된다. 이 사실과 식(1)로부터 정리되는 다음의 식으로부터 행렬 A 와 b 는 얻어진다.

$$\begin{gathered} T_\beta - \beta_0 w_0 T_{cp} \geq 0 \\ T_\beta - \sum_{j=1}^{i} \beta_j z_j T_{cm} - \beta_i w_i T_{cp} \geq 0, \ i = 1, 2, \cdots, m \\ \beta_0 + \cdots + \beta_m \geq \sum_{i=1}^{n} q_i \end{gathered} \qquad (4)$$

식(2)~(4)로부터 선형계획하여 부하의 최적분배를 하여 T_β 의 값이 최소가 되게 하여야 하는데 β_j 값은 임의 분할 불가능한 부하의 합이므로 위의 선형계획법으로 구한 β_j 값에 맞게 부하를 분배할 수 없다. 따라서 '각각의 β_j 가 어떤 임의 분할 불가능한 부하 q_i 을 선택할 것인가?'에 대한 다음과 같은 선행 과정이 필요하다.

$$[\beta_0\ \beta_1\ \cdots\ \beta_m] = v * \begin{bmatrix} x_{11} & \cdots & x_{1m+1} \\ & \vdots & \\ x_{n1} & \cdots & x_{nm+1} \end{bmatrix} \qquad (5)$$

where $x_{ij} = 0$ or 1 and $\sum_{j=1}^{m+1} x_{ij} = 1$

($i = 1, 2, \cdots, n$, $j = 1, 2, \cdots, m+1$,

$v = [q_1\ q_2\ \cdots\ q_n]$: 임의 분할 불가능한 부하)

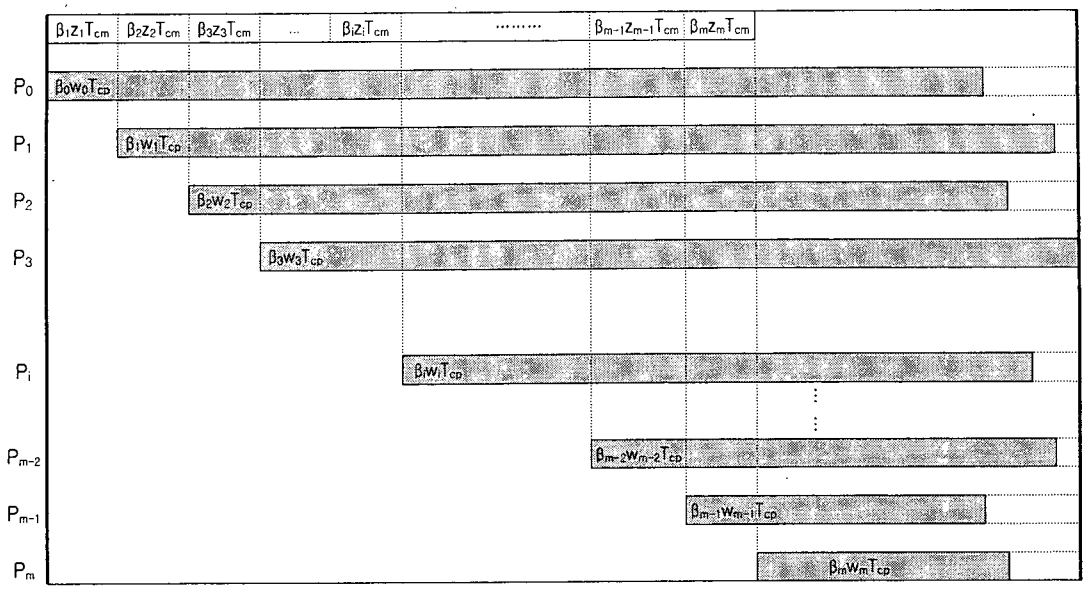

그림 2. 네트워크의 타이밍 다이어그램(timing diagram for network)

식(5)에서 x_{ij}는 β_j가 어떤 q_i을 취할 것인가에 대한 선택연산자(select operate)이다. 식(5)을 식(2)~(4)에 적용하여 선형계획하면 임의 분할 불가능한 부하의 최적분배(optimal load distribution)가 이루어진다.

3.2. 임의 분할 불가능한 부하와 가능한 부하의 분배

임의 분할 불가능한 부하와 가능한 부하가 같이 입력되었을 때, 임의 분할 불가능한 부하는 임의로 분할 할 수 없기 때문에 앞 절의 방법으로 분배 후 그림 2에서처럼 휴면시간(idle time)이 가장 많이 발생하는 순의 프로세서에 처리시간이 T_β을 넘지 않게 적절히 임의 분할 가능한 부하를 할당하고 할당 후 남는 부하는 T_β을 가장 적게 넘게 각 프로세서에 부하를 분배하는 것이 최적화하는 방법이다. 이 같은 방법의 해결은 다음의 선형계획법으로 해결할 수 있다.

- 목적함수(objective function) : Minimize $c^T X_\gamma$ (6)

($c^T = [1\ 0\ 0\ \cdots\ 0]$, $X_\gamma{}^T = [\ T\ \gamma_0\ \gamma_1\ \cdots\ \gamma_m\ ]$)

- 제약조건 : $AX_\gamma \geq -D + b$, $X_\gamma \geq 0$ (7)

($b^T = [0\ 0\ \cdots\ (\sum_{i=0}^{m}\beta_i + \sum_{i=0}^{m}\gamma_i)]$,

$\sum_{i=0}^{m}\beta_i (= \sum_{i=1}^{n} q_i)$: 임의 분할 불가능한 부하의 총합,

$\sum_{i=0}^{m}\gamma_i$: 임의 분할 가능한 부하의 총합,

$D = [\ T_{\beta_0}\ T_{\beta_1}\ T_{\beta_2}\ \cdots\ T_{\beta_m}\ -\sum_{i=0}^{m}\beta_i\ ]$)

식(6)~(7)의 선형계획법으로 임의 분할 가능한 부하의 할당이 결정된다. 각 프로세서에 할당되는 임의 분할 불가능한 부하와 가능한 부하의 합은 $\alpha_i = \beta_i + \gamma_i$ ($i = 0, 1, 2, \ldots, m$)으로 표현된다.

각각의 γ_i , β_i ($i = 0, 1, 2, \ldots, m$)는 임의 분할 가능한 부하와 불가능한 부하가 각각의 프로세서에 얼마만큼 할당되는지를 나타낸다. 그리고 네트워크가 모든 부하의 처리를 마치는데 소비되는 시간은 다음과 같다.

$$T = \max\{T_{\alpha_0},\ T_{\alpha_1},\ T_{\alpha_2},\ \cdots,\ T_{\alpha_m}\}$$

4. 1단계 트리 네트워크에 적용 예

그림 3과 같이 1개의 루트 프로세서와 3개의 자 프로세서가 연결된 구조로 된 네트워크에 앞 절의 알고리즘을 적용하여 최적분배를 한다. 위 네트워크에서 표준 프로세서의 처리시간과 전송시간은 T_{cm}

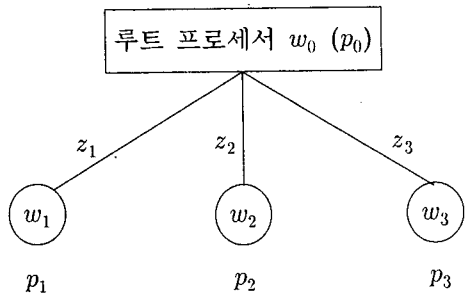

그림 3. 1단계 트리 네트워크
(sing-level tree network)

=1, T_{cp}=1이다. 또, 각 프로세서의 처리시간의 비(ratio processing speed)는 w_0=4.6836, w_1=6.3439, w_2=6.8636, w_3=7.7463이고, 각 프로세서로 전송시간의 비(ratio communication speed)는 z_1=0.8940, z_2=1.0865, z_3=1.2881이다. 근 프로세서에 0.2135, 0.1580, 0.1665, 0.0740, 0.0689의 크기를 갖는 임의 분할 불가능한 부하와 부하의 합이 0.1397인 임의 분할 가능한 부하가 도착되어있다.

이를 바탕으로 임의 분할 불가능한 부하의 분배, 임의 분할 불가능한 부하와 가능한 부하의 혼합 분배시 임의 분할 가능한 부하의 크기에 따른 분배를 한다.

4.1. 임의 분할 불가능한 부하의 분배

아래와 같은 목적함수와 제약조건하에서 선형계획을 한다.

- 목적함수(Objective function)
  : Minimize $c^T X_j$
  (c^T = [1 0 0 0 0], X_j=[T_j β_0 β_1 β_2 β_3])
- 제약조건 : $AX_j \geqq b$, $X_j \geqq 0$

$$(A = \begin{bmatrix} 1 & -4.6836 & 0 & 0 & 0 \\ 1 & 0 & -7.2379 & 0 & 0 \\ 1 & 0 & -0.8940 & -7.9501 & 0 \\ 1 & 0 & -0.8940 & -1.0865 & -9.0344 \\ 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

b^T = [0 0 0 0 0.6809],
v=[0.2135 0.1580 0.1665 0.0740 0.0689]
: 임의 분할 불가능한 부하)

선형계획법으로 구한 v 에 대한 선택연산자는

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

와 같이 구해지므로 임의 분할 불가능한 부하 중 크기가 0.2135 와 0.0740인 것은 프로세서 p_0에, 0.1580인 것은 프로세서 p_2에, 0.1665인 것은 프로세서 p_1에, 0.0689인 것은 프로세서 p_3에 할당이 되고, 각 프로세서가 갖는 임의 분할 불가능한 총 부하는 β_0 = 0.2875, β_1 = 0.1665, β_2 = 0.1580, β_3 = 0.0689이 된다. 이로부터 각 프로세서에서 처리를 마치는 시간은 T_{β_0} = 1.3465, T_{β_1} = 1.2051, T_{β_2} = 1.4050, T_{β_3} = 0.9430와 같이 되고, 처리를 마치는데 가장 오랜 시간을 소비하는 프로세서는 프로세서 p_2인 것을 알 수 있다. 그리고 네트워크가 전체 계산을 마치는데 소비되는 시간은 프로세서 p_2가 처리를 마치는 시간 (T = 1.4050)이 된다.(그림4)

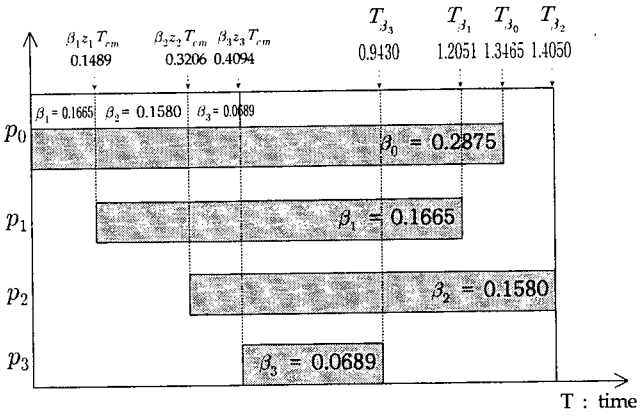

그림 4. 임의 분할 불가능한 부하 분배 타이밍 다이어그램

4.2. 임의 분할 불가능한 부하와 가능한 부하 분배

4.2.1. 임의 분할 가능한 부하가 큰 경우

앞 절에서 임의 분할 불가능한 부하 분배의 결과를 제약조건에 포함시켜 임의 분할 가능한 부하의 할당을 구하면 γ_0 = 0.0336, γ_1 = 0.0413, γ_2 = 0.0078, γ_3 = 0.0571이 된다. 각 프로세서에 할당되는 총 부하는 α_0 = 0.3211, α_1 = 0.2078, α_2 =

0.1658, α_3 = 0.1260이 된다. 각 프로세서에서 부하를 처리하는데 소비되는 시간은 T_{α_0}= 1.5038, T_{α_1}= 1.5038, T_{α_2}= 1.5038, T_{α_3}= 1.5038이 된다. 따라서 시스템이 전체 계산을 마치는데 소비되는 시간은 T = 1.5038이 됨을 알 수 있고, 모든 프로세서가 동시에 처리를 마침을 알 수 있다.(그림5)

이 같은 결과는 임의 분할 가능한 부하가 모든 프로세서가 동시에 처리를 마칠 수 있도록 충분한 부하를 주었기 때문에 가능하다.

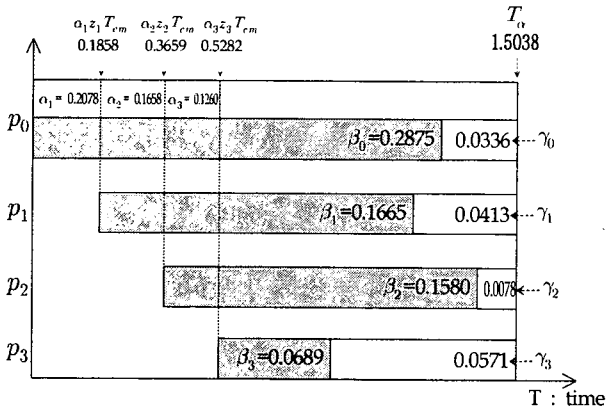

그림 5. 임의 분할 불가능한 부하와 가능한 부하의 분배 타이밍 다이어그램 (임의 분할 가능한 부하가 큰 경우)

4.2.2. 임의 분할 가능한 부하가 작은 경우

앞 절에서 근 프로세서에 도착하는 임의 분할 가능한 부하의 합이 0.0824로 바뀐 경우를 살펴본다. 이 부하의 분배를 구하면 γ_0 = 0.0148, γ_1 = 0.0125, γ_2 = 0, γ_3 = 0.0511이 되고, 각 프로세서에 할당되는 총 부하는 α_0 = 0.3023, α_1 = 0.1790, α_2 = 0.1580, α_3 = 0.1200이 된다. 또 각 프로세서에서 부하를 처리하는데 소비되는 시간은 T_{α_0}= 1.4159, T_{α_1}= 1.2956, T_{α_2}= 1.4161, T_{α_3}= 1.4158이 된다. 그리고 시스템이 모든 임의 분할 불가능한 부하와 임의 분할 가능한 부하의 처리를 마치는데 소비되는 시간이 프로세서 p_2가 처리를 마치는 시간(T = 1.4161)이 된다. 이 같은 경우는 임의 분할 불가능한 부하의 분배를 마쳤을 때, 최대의 처리시간을 갖는 프로세서가 처리를 하는 동안 다른 프로세서들은 휴면시간을 갖게 되는데 이 휴면시간을 갖는 프로세서가 임의 분할 가능한 부하를 분담하여 처리를 하여도 최대 처리시간을 갖는 프로세서가 처리를 마치기 전에 처리를 마치고 휴면시간(idle time)을 갖는 프로세서가 존재하기 때문이다.(그림 6) 또, 그림 6을 보면 프로세서 p_1만이 노는 시간을 갖는 것을 볼 수 있는데, 그 이유는 전송시간을 단축하기 위해 프로세서 p_0, p_3에 더 많은 부하가 분배되기 때문이다.

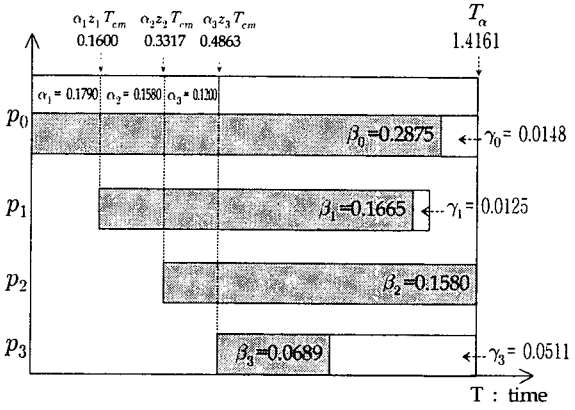

그림 6. 임의 분할 불가능한 부하와 가능한 부하의 분배 타이밍 다이어그램 (임의 분할 가능한 부하가 작은 경우)

5. 결 론

병렬처리시스템에서 최적부하분배(optimal load distribution)는 네트워크상에 연결되어있는 프로세서가 동시에 처리를 마치도록 분배하는 것이 이상적이다. 이것을 위해 각 프로세서의 처리능력과 각 프로세서에 연결되는 링크의 전송속도를 고려하여 각 프로세서가 얼마의 부하를 가질 것인가를 계산할 수는 있지만, 임의 분할 불가능한 부하가 입력될 때에는 이 계산 값대로 부하를 분배할 수 가 없다. 따라서 임의 분할 불가능한 부하가 입력될 때 선형계획법을 통해 처리시간을 최대한 짧게 하는 분배방법을 제시하였다. 또한, 임의 분할 불가능한 부하와 임의 분할 가능한 부하가 혼합되어 입력되었을 경우의 최적 분배방법을 제시하였다.

[참 고 문 헌]

[1] Y.C. Cheng, and T.G. Robertazzi, "Distributed computation with communication delay", *IEEE*

Trans. on Aerospace and Electronic Systems, Vol.24, No.6, pp.700-712, 1988.

[2] S. Bataineh, and T.G. Robertazzi, " Bus oriented load sharing for a network of sensor driven processors", *IEEE Trans. on System Man Cybernetics,* Vol. 21, No.5, pp.1202-1205, 1991.

[3] T.C. Cheng, and T.G. Robertazzi, "Distributed computation for a tree network with communication delays", *IEEE Trans. on AES,* Vol.26, No.3, pp.511-516, 1990.

[4] T.G. Robertazzi, "Processor equivalence for daisy chain load sharing processors", *IEEE Trans. on AES,* Vol.29, No.4, pp.1216-1221, 1993.

[5] J. Sohn, and T.G. Robertazzi, "Optimal divisible job load sharing on bus networks", *IEEE Trans. on AES,* Vol.32, No.1, pp.34-40, 1996.

[6] V. Bharadwaj, D. Ghose, and V. Mani, "Optimal sequencing and arrangement in distributed single-level tree networks with communication delays", *IEEE Trans. on Parallel and Distributed Systems,* Vol.5, No.9, pp.968-976, 1994.

[7] J. Blazewicz, and M. Drozdowski, "The performance limits of a two-dimensional network of load sharing processors", *Foundations of Computing and Decision Sciences,* Vol. 21, No.1, pp.3-15, 1996.

[8] D. Ghose and V. Mani, "Distributed computation with communication delays" Asymptotic performance analysis", *J. of Parallel and Distributed Computing,* Vol.23, No.3, pp.293-305, 1994.

[9] K. Li, "Parallel processing of divisible loads on partitionable static inetconnection networks", *Cluster Computing, Special Issue on : Divisible Load Scheduling,* Vol.6, No.1, pp.47-55, 2003.

[10] D. Ghose, and H.J. Kim, "A generalized linear programming approach to optimal divisible load scheduling", *Technical Report KNU/CI/MSL/001/2004,* Kangwon National University, Korea, 2004.