부하균형을 위한 부하상태 전파 알고리즘

이옥빈* 김성열** 정일용*** 이상호*
* 충북대학교 컴퓨터과학과
** 울산과학대학 컴퓨터공학부
*** 조선대학교 전자계산학과

An Algorithm for sending Workload Information in Distributed Load Balancing

Ok-Bin Lee Seong-Yeol Kim Il-Yong Chung Sang-Ho Lee
E-mail : lobin@hyun.chosun.ac.kr

요 약

분산시스템에서 부하균형을 유지하기 위해서는 네트워크상의 각 노드는 다른 노드들의 부하상태정보를 가져야만 한다. 네트워크상에 v 개의 노드가 존재할 때 모든 노드간에 부하상태정보를 교환하기 위해서는 $O(v^2)$의 트래픽 오버헤드가 필요하게 된다. 이 논문에서는 분산된 노드간에 동기적으로 동작하는 부하균형 알고리즘을 제시한다. 이를 위해 먼저 SBIBD(symmetric balanced incomplete block design)에 근거하여 $(v,k,1)$-configuration에 의해 $v=k^2-k+1$개의 노드를 갖는 네트워크 토폴로지를 구성하였다. 이 망에서 동작하도록 고안된 부하상태정보 전파 알고리즘은 $O(v\sqrt{v})$의 메시지 오버헤드를 가지면서 각각의 노드가 v개의 모든 노드에 대한 부하상태정보를 가지도록 한다. 또한 이 부하균형 알고리즘은 모든 링크가 부하상태정보 전송을 위해 동일한 트래픽을 갖도록 설계되었다.

1. 서론

분산시스템에서 어떤 시스템은 과부하 되는가 하면 또 어떤 시스템은 부하가 적거나 유휴상태를 나타내기도 한다. 그러므로 이렇게 불균형한 부하를 갖는 시스템간에 부하균형이 유지되도록 부하를 분산함으로써 시스템의 활용도를 높이고 응답시간을 줄일 수가 있다. 따라서 부하분산 기법은 부하상태에 대한 정보의 유지 및 전파, 프로세스이동시점의 결정, 이동시킬 프로세스 선택, 프로세스가 이동될 컴퓨터의 결정, 전송된 프로세스의 상태복구 등을 고려해야한다. 이때 컴퓨터의 부하상태를 측정하거나 부하상태정보를 전파하기 위한 방식은 최소의 오버헤드를 가져야한다. 컴퓨터의 작업량을 나타내는 지수는 CPU의 대기열에 의해 표현되는 것이 가장 효과적이라고 알려져 있다 [2]. 부하균형알고리즘은 정적(static)이거나 동적(dynamic)일 수 있다. 분산시스템이 v개의 컴퓨터로 구성되어 있을 때 정적인 분산시스템에서는 i번째 작업을 $i \pmod v$ 번째 컴퓨터에 할당한다. 동적인 분산시스템에서는 각 시스템의 부하상태 정보를 이용하여 부하가 적은 컴퓨터에 작업을 할당한다. 또한 작업량 분산 방식은 특정 시스템에서 모든 노드에 대한 부하정보를 가지고 있으면서 불균형된 노드간에 부하를 분산하도록 중재하는 집중형과 각각의 노드가 다른 노드에 대한 부하정보를 가지고 스스로 부하분산을 실행하는 분산형으로 구분될 수 있다. 분산시스템에서 부하 분산을 위해서는 분산형의 동적인 알고리즘을 사용하는 것이 바람직하다. 부하 분산을 위해 각각의 노드에게 주어지는 부하상태정보는 최신의 정보이어야 할 필요가 있다. 유효시간이 지나버린 정보는 각각의 노드에서 시스템의 전체에 대한 상황을 올바로 파악할 수 없게 하고 더불어 올바른 부하분산을 수행할 수 없게 한다. 이렇게 최신의 정보를 유지하기 위해서는 주기적으로 갱신되는 정보전송을 위한 통신비용이 심각한 문제가 될 수 있다.

이 논문에서는 v 개의 노드와 $v\times(k-1)$ 개의 링크를 가지며 각각의 노드는 균등하게 $2(k-1)$ 개의 인

접노드를 갖는 정규그래프 형태의 네트워크 토폴로지를 설계하였다.

이와 같은 네트워크상에서 각각의 노드는 $k-1$개의 인접노드로 $k-1$개의 노드에 대한 부하상태정보를 전송한다. 즉, 각각의 노드는 $k-1$개의 노드로부터 정보를 수신하는데 각각 하나의 노드로부터 $k-1$개의 노드에 대한 정보를 수신한다. 이렇게 수신된 모든 정보는 정보 발생지로부터 2단계 이내에서 수신 완료된 것이다. 또한 이렇게 수신된 정보에는 중복이 존재하지 않는다. 따라서 각각의 노드는 매 주기마다 $(k-1)^2$개의 노드에 대한 상태정보를 수신하게된다. 또한 시간주기 T_{2t}와 T_{2t+1}에 수신하는 정보의 차(difference)가 $k-1$가 됨으로써 각 노드는 네트워크상의 모든 노드에 대한 부하상태정보를 얻을 수 있게 된다. 또한 이 알고리즘은 부하상태정보 전송을 위해 각각의 링크가 균등한 트래픽 오버헤드를 갖도록 설계되었다.

2. SBIBD

$V=\{v_0, v_1, \ldots, v_{v-1}\}$를 v개의 원소를 갖는 집합이라고 하자. 그리고 B를 다음과 같이 k개의 원소를 갖는 부분집합들로 구성된 집합족 이라고 하자. $B=\{B_0, B_1, \ldots, B_{b-1}\}$, $B_i=\{b_0, b_1, \ldots, b_{k-1}\}$ 유한결합구조 $\sigma=\{V, B\}$에 대하여 σ가 다음조건을 만족할 때 이는 BIBD(balanced incomplete block design)이다. 그리고 이를 (b, v, r, k, λ)-configuration이라 한다[1][3].

(1) B는 블럭이라 불리는 b개의 부분집합으로 구성되며 각각의 부분집합은 V에 속하는 k개의 원소를 갖는다.

(2) V의 원소 각각은 r개의 블록에만 존재한다.

(3) V의 원소중 임의의 두 원소 쌍(pair)은 B에서 λ번 나타난다.

(4) $k<v$

임의의 (b, v, r, k, λ)-configuration이 다음 조건을 만족하면 이는 SBIBD(symmetric balanced incomplete block design)이고 이를 (v, k, λ)-configuration 이라 한다[1][3].

(1) $k=r$

(2) $b=v$

BIBD를 구성하기 위한 파라미터 b, v, r, k, λ 간에는 특별한 관계가 존재한다.

예를 들면 (b, v, r, k, λ)-configuration에서는 $bk=vr$이 성립하고 $r(k-1)=\lambda(v-1)$이 성립한다. 그리고 (v, k, λ)-configuration의 경우 모든 두 블록 사이에는 λ개의 교집합이 존재한다.

임의의 (v, k, λ)-configuration을 생성하기 위해서는 (v, k, λ)-차집합을 이용할 수 있다[3].

$|V|=v$일 때 $D=\{d_0, d_1, \ldots, d_{k-1}\} \subseteq V$를 k개로 이루어진 부분집합 이라고 하자 (단, $v>k\geq 2$).

모든 원소 $a \in A$, $a \neq 0$에 대하여

$a = d_i - d_j$, $(d_i - d_j \in D)$

인 순서쌍 (d_i, d_j)의 개수가 일정한 정수 λ와 같을 때 D를 V의 (v, k, λ)-차집합이라 한다.

예를 들면 $V=\{0, 1, 2, 3, 4, 5, 6\}$일 때 차집합 $D=\{0, 1, 3\}$은 (7,3,1)-차집합이다. 실재로 V의 원소 1, 2, 3, 4, 5, 6은 다음과 같이 D의 두 원소의 차로 나타내어지며 나타내는 방법은 단 한가지 뿐이다.

1 = 1-0	4 = -3 = 0-3
2 = 3-1	5 = -2 = 1-3
3 = 3-0	6 = -1 = 0-1

이러한 차집합으로부터 [표 1]과 같이 (7,3,1)-configuration을 구성할 수 있다.

[표 1] (7,3,1)-configuration의 예

B_0 = { 0, 1, 3 }
B_1 = { 1, 2, 4 }
B_2 = { 2, 3, 5 }
B_3 = { 3, 4, 6 }
B_4 = { 4, 5, 0 }
B_5 = { 5, 6, 1 }
B_6 = { 6, 0, 2 }

3. 분산시스템의 부하균형알고리즘

3.1 알고리즘 1 - 분산시스템의 네트워크 토폴로지 설계

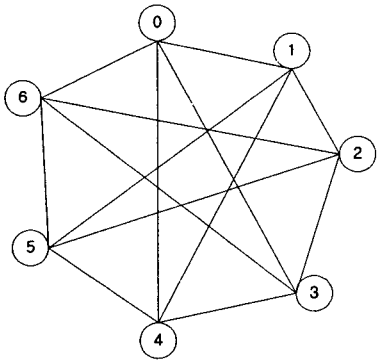
분산시스템의 부하분산을 위해 최소의 링크비용과 트래픽 오버헤드를 갖는 네트워크 토폴로지 설계를 위해 (v,k,λ)-configuration을 도입하기로 한다. 어떤 유한결합구조 $L=\{V,B\}$가 (v,k,λ)-configuration이고 G를 L에 대한 이진결합행렬이라 하자. 이때 행렬 G는 그래프 $G=\{V,E\}$의 인접행렬로 변환될 수 있다. 이러한 접근법에 의해 (v,k,λ)-configuration 으로부터 네트워크 토폴로지를 얻을 수 있다.

(1) $V=\{0,1,2,\ldots,v-1\}$ 에 대하여 $(v,k,1)$-차집합으로부터 $(v,k,1)$-configuration을 만족하는 유한결합구조 $L=\{V,B\}$을 생성한다.

(2) L의 이진결합행렬 M을 생성한다.

(3) L 로부터 다음과 같이 그래프 G의 인접행렬 $A(G)$를 생성한다. 여기에서 G는 v개의 프로세서들로 이루어진 네트워크 토폴로지가 된다.

$$a(i,j)=\begin{cases} 1 & \begin{array}{ll} \text{if } i\neq j & \text{and} \\ \text{if } i\in L[j] & \text{or } j\in L[i] \end{array} \\ 0 & \begin{array}{ll} \text{if } i=j & \text{or} \\ \text{if } i\notin L[j] & \text{and } j\notin L[i] \end{array} \end{cases}$$



[그림 1] L로부터 생성된 그래프 G

예를 들면 알고리즘 1에 의해 [표 1]로부터 생성된 네트워크 토폴로지는 [그림 1]과 같다.

정리 2. 알고리즘 1에 의해 얻어진 네트워크 G는 v개의 노드와 $v\times(k-1)$개의 링크를 가지며 각 노드의 차수가 $2(k-1)$인 정규그래프이다.

증명 : G는 $(v,k,1)$-configuration으로부터 생성되었으므로 노드의 수는 v가 된다. 알고리즘 1-(1)으로부터 L을 구성하는 각각의 블럭 $B[i]$는 k개의 원소 중 한 원소는 i임을 알 수 있다. 알고리즘 1-(3)에 의해 각 노드의 링크가 결정되는데, 식에서 보는바와 같이 노드 i는 $B[i]$ 중 자신을 제외한 $k-1$개의 원소와 연결되어있고 동시에 블럭 $B[j]$가 노드 i를 포함하는 경우의 $k-1$개 노드 j와 연결되어 있다. 그러므로 SBIBD의 정의에 의하여 노드 i는 $2(k-1)$개의 링크를 갖는다. 따라서 네트워크의 링크의 수는 $(2(k-1)\times v)/2 = v(k-1)$임을 알 수 있다.

3.2 알고리즘 2 - $2(k-1)$차 정규그래프상의 부하 상태정보전파 알고리즘

(1) 네트워크상의 각각의 노드 n은 다음과 같이 집합 S_n과 R_n을 정의한다. 여기에서 S_n은 시간 T_{2t}에 노드 n으로부터 상태정보를 수신하는 n에 인접한 $k-1$개의 노드이다. 그리고 R_n은 T_{2t}에 노드 n에 정보를 제공하는 n에 인접한 또다른 $k-1$개의 노드이다. 노드집합이다. 시간 T_{2t+1}에 정보전송의 방향을 거꾸로 한다.

$$S_n=\{v \mid v\in L[n] \text{ and } n\neq v\}$$

$$R_n=\{v \mid n\in L[v] \text{ and } n\neq v\}$$

(2) 네트워크상의 각각의 노드 n은 다음과 같은 집합족 SF_n과 RF_n을 정의한다.

$$SF_n = \{ SF_n(i) \mid i \in S_n, \quad SF_n(i) = \{\{n\} \cup S_n - \{i\}\}$$

$$RF_n = \{ RF_n(i) \mid i \in R_n, \quad RF_n(i) = \{\{n\} \cup R_n - \{i\}\}$$

(3) 각 노드 n은 부하상태정보를 갱신하고 노드집합 S_n의 각 원소 j 에게 노드집합 $SF(j)$ 의 상태정보를 전송한다.

(4) 각 노드 n은 부하상태정보를 갱신하고 노드집합 R_n의 각 원소 j 에게 노드집합 $RF(j)$ 의 상태정보를 전송한다.

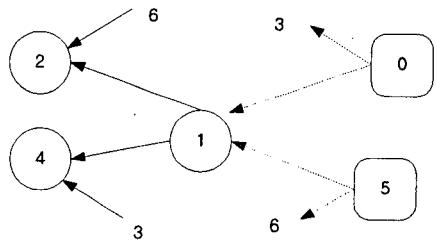
(5) (3)을 반복한다.

정리 3. *알고리즘 2에 의해 각각의 노드는 매 전송 주기마다 $(k-1)^2$개의 노드에 대한 부하상태정보를 얻는다.*

증명 : 노드 n에 대하여, 노드 n으로부터 T_{2t}에 정보를 수신하는 $k-1$개 노드들의 집합을 $S_n = \{s_1, s_2, \ldots, s_{k-1}\}$라 하자. 그러면 $R_n = \{r_1, r_2, \ldots, r_{k-1}\}$는 T_{2t}에 노드 n에 정보를 제공하는 노드집합이된다. 왜냐하면 r_i는 n을 원소로 갖는 $B[i]$로부터 생성되었기 때문이다. 그러므로 시간 T_{2t}에 r_i는 $k-1$개의 노드집합 $SF_{r_i}(n)$ 에 대한 부하상태정보를 노드 n에 전송한다.

그런데 $k-1$개의 노드 R_n으로부터 전송되는 정보에는 중복이 존재하지 않는다. 왜냐하면 SBIBD에는 λ개의 공통요소가 존재하는데 $SF_{r_i}(n)$에 존재하는 공통의 요소는 바로 n이기 때문이다. 따라서 노드 n은 시간 T_{2t}에 $(k-1)^2$개의 노드에 대한 부하상태정보를 수신한다. 또한 T_{2t+1} 에도 마찬가지로 동작한다.

예를 들면 노드 1을 중심으로 T_{2t}에 전송되는 정보의 흐름은 [그림 2]와 같다. 시간 T_{2t+1} 에는 정보전송의 방향이 거꾸로 된다. 그리하여 노드 1의 부하상태정보 수신상태는 [표 2]과 같다.



[그림 2] 시간 T_{2t}에서 노드 1의 정보 송수신

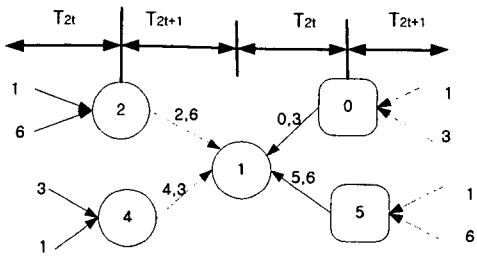
[표 2] 노드 1의 부하상태정보 수신상태

T_{2t}	T_{2t+1}
from 0 => 0, 3	from 2 => 2, 6
from 5 => 5, 6	from 4=> 4, 3

정리 4. *각 노드에서 수신한 부하상태정보의 도달거리는 2 이내이다.*

증명 : 노드 n이 노드 r_i로 부터 T_{2t}에 수신한 $k-1$개의 부하상태정보는 T_{2t-1}에 생성된 r_i에 대한 상태정보와 T_{2t-2}에 생성된 $SF_{r_i}(n) - \{r_i\}$에 대한 상태정보로 구성되어있다. 그러므로 정보도달 거리는 2 이내임을 알 수 있다.

예를 들면 [표 2]에서 노드 1이 수신한 정보의 도달거리는 [그림 3]과 같다.



[그림 3] 노드 1의 수신상태와 수신정보의 도달거리

정리 5. 각각의 노드는 알고리즘 3에 의해 네트워크 상의 모든 노드에 대한 상태정보를 수신한다.

증명 : 정리 3에 의하면 노드 n은 T_{2t}와 T_{2t+1}에 $k-1$개의 노드에 대한 부하상태정보를 수신하지 못한다. 여기에서 우리는 T_{2t}에 수신하지 못한 $k-1$개의 정보를 T_{2t+1}에 수신함을 증명함으로써 각각의 노드가 모든 노드에 대한 상태정보를 수신함을 증명하려한다.

$S_n = \{s_1, s_2, \ldots, s_k\}$는

$L[n] = \{n, s_1, s_2, \ldots, s_k\}$

으로부터 유도된 것이다.

그리고 $R_i = \{r_1, r_2, \ldots, r_k\}$에 의해 다음과 같은 블럭이 존재했음을 알 수 있다.

$$L[r_1] = \{n, r_1, \ldots\}$$

$$L[r_2] = \{n, r_2, \ldots\}$$

$$\vdots$$

$$L[r_k] = \{n, r_k, \ldots\}$$

여기에서 이미 $L[n]$에 n과 s_i의 쌍이 존재하므로 상이 $L[i_j]$에는 s_i가 나올 수 없다. 물론 반대의 경우도 성립한다. 따라서 T_{2t}에 수신하지 못한 k개의 정보를 T_{2t+1}에 수신하게 됨으로써 각 노드는 모든 노드에 대한 부하상태정보를 얻게됨을 알 수 있다.

4. 결론

이 논문에서는 분산시스템의 부하균형을 위하여 블럭디자인을 이용하여 $2(k-1)$차 정규그래프 형태를 갖는 네트워크 토폴로지를 설계하고 이 네트워크상에서 2단계 전송에 의하여 각각의 모든 노드에 대한 부하상태정보를 얻을 수 있는 알고리즘을 제시하였다. 이 알고리즘에서는 이렇듯 모든 노드의 부하상태정보를 전달해주기 위해 각각의 노드와 링크가 동일한 트래픽 오버헤드를 갖는다. 또한 이 알고리즘은 최소의 링크비용과 메시지 오버헤드를 보장하는데, 이는 모든 링크가 동일한 트래픽을 가지며 동시에 전송된 정보에 중복이 없다는 사실로부터 증명될 수 있다.

[참고문헌]

[1] C.L.Liu, Block designs in Introduction to Combinational Mathamatics, McGraw-Hill, pp359-383, 1968

[2] R.Knuz, The Influence of Different Workload Description on a Heuristic Load Balancing Scheme, IEEE Trans. on Software Eng., Vol. 17, No. 7, July 1991. pp.725-730.

[3] M. K. Bennett, Affine and projective geometry, Wiley & Sons, 1995.

[4] Behrooz A. Shirazi, Scheduling and load balancing in parallel and distributed systems, IEEE Computer Society Press, 1995.