Agent 기반의 부하 분산을 위한 집합 기반 확산 전송 기법*

김중항° 전상현 장형수
서강대학교 컴퓨터학과
{jhkim, shjeon, hschang}@smolab.sogang.ac.kr

Sogang University

Joong Hang Kim Sang Hyun Jeon Hyeong Soo Chang
Department of Computer Science and Engineering, Sogang University
Interdisciplinary Program of Integrated Biotechnology

요 약

본 논문에서는, 부하 분산의 대표적인 기법들 중에서 빠른 부하 분산을 수행하지만 workload의 이동 횟수가 많은 diffusion method와 느린 부하 분산을 수행하지만 workload의 이동 횟수가 매우 적은 messor의 두 가지 방식을 집합을 기반으로 하는 확산 전송 방식으로 융합하여, 효과적으로 부하 분산을 수행할 수 있는 기법을 제안한다. 또한 실험을 통하여, 제안된 기법이 부하 분산의 속도와 workload의 이동 및 이동을 위한 메시지의 증가 사이에서 적절한 trade-off를 제공하는 것을 보인다.

1. 서 론

부하 분산(Load Balancing(LB))[1]이란 distributed computing 환경에서, 각각의 processor들이 처리해야 할 workload의 양이 각기 다른 불균형 상태를 전체 processor들의 workload가 동일한 균형 상태로 만들기 위하여 processor들 사이에서 workload를 적절하게 분배하는 작업을 말한다. 이는 전체적인 distributed network의 작업 효율을 높이는 것뿐만 아니라, 현재의 인터넷 환경과 같이 여러 대의 processor들이 고속의 network상에 구축되어 있는 highly distributed computing 환경에서는 각각의 processor에 요구되는 작업의 수를 예측하기가 매우 어렵기 때문에, 빠른 시간 안에 idle상태의 다른 processor에게 작업을 분산시켜 전체 network의 작업의 효율성을 높이는 LB가 매우 중요하다. 즉 LB는 distributed computing 상에서 풀어야 할 중요한 문제들 중의 하나이며, LB를 위한 알고리즘은 모든 processor들이 균일한 양의 workload를 가지도록 하는 것과 이를 위한 workload의 전송 횟수가 최소화되도록 하는 것을 목적으로 한다[2]. 본 논문에서는, 빠른 LB가 이루어지지만 workload의 이동 횟수가 매우 많은 diffusion method[3]와, 적은 workload의 전송 횟수를 가지지만 diffusion method보다 느린 LB를 수행하는 messor[4]방식 두 가지를 집합 기반 확산 전송(Set-based Load Migration)으로 융합하여 효율적인 LB를 수행할 수 있는 기법을 제안한다. 또한 실험을 통하여, 제안된 기법이 기존의 diffusion 방식보다 적은 workload의 전송 횟수를 가지고 diffusion방식과 비슷한 속도의 LB를 수행하는 것을 보인다.

2. 문제 정의 및 관련 연구

2.1 문제 정의

어떠한 시스템에 저장된 object(data item)는 load로 표현될 수 있다. 이러한 load는 object를 저장하기 위해서 필요한 비트의 수, object의 개수, 또는 object를 사용자에게 제공하기 위한 processor time의 양으로 표현되며 각각의 object는 node들 간에 object를 이동시키기 위하여 필요한 movement cost를 가지고 있으며, 이는 보내는 node와 받는 node 사이에서 일정한 값으로 나타내어진다[2].

시간 t에서 임의의 node i의 workload w_i^t는 해당 node에 저장되어 있는 object들의 load의 합으로 표현된다.

N개의 node를 가진 연결된 network 상에서, 시간 t에서 각 node에 걸린 workload의 전체적인 분포를 $W^t=(w_1^t, w_2^t, \cdots, w_N^t)$라고 하며, 시간 $t\,(t>0)$에서의 전체 network의 workload에 대한 표준 편차 σ^t는

$$\sigma^t=\sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(w_k^t-\overline{W^t}\right)^2}$$

로 나타낼 수 있으며, 이 때 $\overline{W^t}$는 W^t에 대한 평균이다. 본 논문에서는 이러한 σ^t를 전체 network에 걸린 workload가 얼마나 균일하게 분포하는지에 대한 척도로 사용한다.

일반적으로 LB가 가지는 goal[2]은, 전체 network의 효율을 높이기 위하여 초기 시간 $t=0$에서 각 node가 가지는 workload의 분포인 W^0에 대한 불균형을 나타내는 σ^0를 t가 증가함에 따라 0으로 수렴하게 하며, 이에 따라 발생되는 workload의 이동으로 인한 movement cost를 최소화하는 것이다. 본 연구에서는, 초기의 W^0값이 시간에 따른 workload의 이동에 의해서만 변화하는 정적인 네트워크의 환경으로 가정한다.

2.2 관련 연구

앞에서 설명된 LB의 문제를 해결하기 위하여, 정적인 network에서 각각의 node들이 알 수 있는 최소의 지역적 정보인 nearest neighbor를 사용하여 LB를 위한 연산 및 workload의 분산을 수행하는 iterative LB의 방식이 많이 연구되고 있다[1][3][5]. 이는 지역적인 정보만을 가지고 반복적인 LB를 수행하며, 반복적으로 balancing step이 수행됨에 따라 전체 network의 각 node에 균일하게 workload가 분산되는 성질을 가지며[1], 이에 따라 iterative LB는 실제 distributed computing 환경에 매우 적합한 기법이 될 수 있다.

Iterative LB는 크게 deterministic iterative 방식과 stochastic iterative의 두 가지로 나뉜다. 두 가지 방식의 가장 큰 차이점은, deterministic iterative 방식이 미리 정해져 있는 규칙에 따라 LB가 이루어지는 반면에 stochastic iterative 방식은 확률적으로 전체 network의 workload들이 redistribute되는 방식이라는 점이다. 현재로서는 deterministic iterative가 가장 활발히 연구되고 있으며, deterministic iterative의 방식에는 diffusion[3], dimension exchange[5], gradient model[6]이 있다. 이러한 알고리즘들은 기본적으로 각 processor가 자신의 workload를 nearest neighbor들과 비교하여 일정량의 workload를 nearest neighbor에 분산시키거나(Push-mode), nearest neighbor에서 자신에게로 workload를 끌어오는(Pull-mode) 방식을 취하고 있다.

Stochastic iterative 방식은 대표적인 알고리즘인 randomized allocation[7]과 physical optimization[8] 모두가 간단하고 쉬운 아이디어라는 장점을 가지지만, 두 방식 모두 LB에 직접 적용하기 위한 알고리즘의 정형화가 부족하다는 단점[1]이 있다.

최근에는 deterministic iterative외에도 agent를 기반으로 한 LB[4]가 연구되고 있다. [4]는 agent를 사용하여 network를 탐색한 뒤, 자신이 탐색한 node들 중에서 일정한 기준으로 sender와 receiver를 선정하여 workload를 분산하는 방식이다.

일반적으로 deterministic iterative의 방식 중에서는 dimension exchange가 가장 빠른 LB를 수행하는 것으로 알려져 있으며[1], 이러한 사실은 dimension exchange가 diffusion방식보다 적은 workload의 전송 횟수를 가지고 전체 network의 workload를 균일하게 만들 수 있다는 연구 결과[5]가 뒷받침하고 있다.

그러나, dimension exchange방식은 전체 network graph의 edge-coloring 문제가 먼저 해결되어 있다는 것(priori edge coloring)을 가정한다[1]. 이러한 가정은 edge coloring의 문제가 NP-complete라는 사실[9]과 edge coloring의 문제를 해결하기 위해서는 임의의 centralized node에서 전체적인 graph의 연결 구조를 사용한 연산이 불가피하다는 점에서, 실제 distributed computing 환경에는 큰 단점으로 작용할 수 있다.

가장 일반적으로 사용되는 deterministic iterative 방식인 diffusion은 일정한 balancing step 이후에 전체 network에 대한 표준 편차가 0으로 수렴한다는 사실이 증명되어 있으며, 어떠한 구조의 network에서도 잘 동작한다는 것이 알려져 있다[1]. 그러나 diffusion방식이 가지고 있는 가장 큰 단점은 매 balancing step마다 각 node에서 최대 nearest neighbor의 수만큼

* 본 연구는 정보통신부에서 지원하는 대학기초연구지원사업 (B1220-0501-0272)으로 수행되었습니다.

workload의 전송이 일어나기 때문에, 전체적인 workload의 전송 횟수가 빠른 속도로 증가한다는 점이며, 이는 4장에서의 실험 결과를 통하여 확인할 수 있다.

또다른 deterministic방식인 gradient방식은 workload가 큰 node끼리의 전송을 고려하지 않는 약점[1] 때문에 널리 쓰이지 못하고 있다.

Agent를 기반으로 하는 messor[8]는 agent가 node들의 workload에 대한 정보를 저장하며 이동하고, 이렇게 저장된 정보 중에서 가장 workload가 큰 node에서 가장 낮은 node로만 workload의 전송이 일어나기 때문에 workload의 전송 횟수가 매우 적다는 장점이 있다. 그러나 이러한 적은 전송 횟수 때문에, 전체 network의 workload가 균일하게 만들어지는 데에는 다른 알고리즘보다 더 많은 balancing step이 필요하며, 이러한 사실 또한 4장에서의 실험으로 확인할 수 있다.

결론적으로, 전체적인 network의 workload를 빠르게 균일화 시키면서 그에 따른 workload의 전송 횟수를 줄이는 알고리즘이 필요하며, 본 논문에서는 이러한 점들을 고려한 agent 기반의 LB을 제안한다. 제안된 LB는 각 node의 정보를 저장하며 network를 이동하는 messor와 비슷한 방식을 취하고, set을 기반으로 workload가 높은 node와 낮은 node를 구분하며 이를 가지고 diffusion 방식 중 가장 일반적인 Average Neighborhood (AN)[3]이 변형된 형태로 적용되도록 설계되었다.

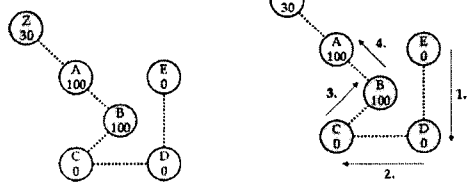
3. Load Balancing via Set-based Diffusive Load Migration

3.1 Basic Idea

본 논문에서 제안된 기법은, 기본적으로는 diffusion방식 중의 하나인 Direct Neighbor Repeated(DNR)[3]의 neighbor 확장 방식을 messor에서의 agent hopping 방식으로 변형하여 nearest neighbor를 구성한 다음, AN을 새롭게 구성된 nearest neighbor에 바로 적용할 경우 생기는 문제점을 해결하기 위하여, set을 기반으로 한 workload의 transfer를 통해 LB를 수행한다.

변형된 agent hopping방식은 agent가 일정 기준에 도달할 때까지 계속해서 다음 node로 이동하는 일반적인 messor에서의 방식과는 달리, agent가 node를 이동하는 hop의 수를 일정하게 제한하고, 일정 hop을 이동하였을 경우 현재까지 지나온 node들을 nearest neighbor로 간주한 뒤 현재 node를 중심으로 하여 LB을 수행한다.

아래의 그림 1에서 원은 node를 뜻하며, 점선은 node와 node가 연결되어 있음을 표시한다. 원 안의 숫자는 해당 node의 workload이며, 알파벳은 node의 이름을 나타낸다. 이 때, 그림 1의 (가)는 일반적인 AN에서 A의 nearest neighbor가 Z와 B로 설정된 그림이며, (나)는 agent의 hop 수가 4로 설정된 agent가 M에서 출발하여 E, D, C, B, A의 차례대로 hopping하여 B, C, D, E가 A의 nearest neighbor로 설정된 상황을 나타내고 있다.



(가) 일반적인 설정 (나) path에 따른 설정

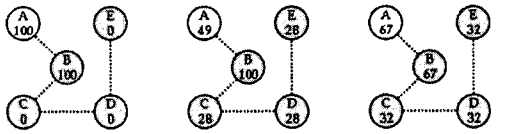
그림 1 : neighbor의 설정 방식 비교

즉, agent는 정해진 hop수만큼을 각 node의 workload에 대한 정보를 수집하면서 이동하고, 정해진 hop수가 되면 자신이 지나온 node들을 nearest neighbor로 삼아 현재 node를 중심으로 하여 LB를 수행한다. 본 논문에서는 이러한 agent기반의 neighbor 확장 방식에nearest neighbor를 통한 AN을 그대로 적용할 경우 생기는 문제점(자세한 내용은 다음 장 참조)을 보완하기 위하여, "Minset-Maxset based Migration(MM)"의 기법을 사용하였다.

MM은 그림 2와 같이 임의의 node i와 j의 nearest neighbor들 중에서, workload가 매우 높은 복수 개의 node가 존재하고 workload가 매우 낮은 복수 개의 node가 존재할 때 기존의 AN으로는 표준 편차를 낮추기 위하여 여러 번의 balancing step이 소요되는 점에 착안하였다. 이러한 단점을 구체적으로 알아보기 위하여, 임의의 두 node 사이에서의 workload의 전송은 한 번의 balancing step에 완료된다고 가정하자. 그림 2의 기호들은 그림 1에서와 같다.

앞에서와 같이, 어떤 agent가 E, D, C, B를 거쳐 A에 도착한 상황을 생각하여 보자. 이에 따라 A는 master node, 즉 workload의 분산을 결정하는 node가 되고 B, C, D, E는 slave node, 즉 A의 nearest neighbor로 설정된 상태이다. 여기서, 일반적인 AN을 따르게 되면 workload가 100인 node가 A와 B 2개임에도 불구하고 master node인 A를 중심으로 LB을 수행하기 때문에, 같은 100의 workload를 가진 slave node B에 대해서는 workload의 차이가 0이므로, 현재 balancing step에서는 workload의 전송이 이루어지지 않는다.

이러한 상황에서 빠른 속도로 전체 network의 workload를 균일하게 만들기 위해서는 자신의 neighbor들까지 함께 고려하여 workload가 일정량보다 높은 다수의 node 각각에서 workload가 일정량보다 낮은 다수의 node로 workload를 분배하면, AN 방식에서 여러 번의 balancing step이 필요한 것에 비하여 balancing step의 횟수를 줄일 수 있게 된다.



표준편차 : 54.77 표준편차 : 31.21 표준편차 : 19.17

(가) 초기 상태 (나) AN 적용 후 (다) MM 적용 후

그림 2. MM과 AN방식의 비교를 위한 Network 구성도

그림 2의 (나)와 (다)는, (가)에서 한 번의 balancing step이 지난 후에 AN과 MM 각각의 LB의 결과를 나타낸다. 그림 2의 (나)는 A에서 C, D, E로 각각 workload의 전송이 이루어진 반면 그림 2의 (다)는 A에서 C, D, E로, 또한 B에서 C, D, E로 workload의 전송이 이루어졌으며, (다)의 방식이 (나)보다 더 빠르게 표준편차가 줄어들도록 만드는 것을 확인할 수 있다.

3.2 MM Migration의 기본 동작 방식

N개의 node를 가진 연결된 network 상에서 각 노드의 이름은 1에서 N사이의 정수 값으로 표현된다고 하자. 같은 node를 중복해서 방문하지 않고, $1 \le d \le N$인 d개의 node를 지나 시간 t에 node i에 도착한 agent θ에 대하여, 자신이 지나온 node들에 대한 workload의 정보가 담긴 queue가 $Q_i^t(\theta) = (w_{N(t-d+1)}^{t-d+1}, \cdots, w_{N(t-1)}^{t-1}, w_{N(t)}^t)$와 같이 나타내어진다고 하자. 이 때 $N(k)$는 시간 k에 agent θ가 방문한 노드의 이름이며, $N(t)=i$이고, 중복되는 값은 없다. 이제 이 $Q_i^t(\theta)$ 중에서 workload가 높은 node와 낮은 node를 구분하기 위하여, $Q_i^t(\theta)$에 대한 평균

$$\overline{Q_i^t(\theta)} = \frac{\sum_{k=t-d+1}^{t} w_{N(k)}^k}{d}$$

를 구한다. 이제 $Q_i^t(\theta)$를 workload가 높은 node의 집합인 $S_{max}^t(\theta)$와 workload가 낮은 node의 집합인 $S_{min}^t(\theta)$로 나누기 위하여($S_{max}^t(\theta) \cup S_{min}^t(\theta) = Q_i^t(\theta)$), $Q_i^t(\theta)$의 각 원소에 대하여 $\overline{Q_i^t(\theta)}$보다 workload가 높은 node는 $S_{max}^t(\theta)$에, workload가 낮은 node는 $S_{min}^t(\theta)$에 포함시킨다. 이제 각각의 $S_{max}^t(\theta)$의 원소에서 $S_{min}^t(\theta)$의 전체 원소들로 전송할 workload의 양을 다음과 같이 계산한다. 이 때, $w_{x,y}$는 node x에서 node y로의 전송량을 뜻하며, $x \in S_{max}^t(\theta)$, $y \in S_{min}^t(\theta)$이다.

$$w_{x,y} = \frac{w_x^K - w_y^K}{|S_{max}^t(\theta)| \cdot (|S_{min}^t(\theta)| + 1)}, \quad K \in [t-d+1, t]$$

해당 $w_{x,y}$가 전송되고, $S_{max}^t(\theta)$의 모든 원소에서의 전송이 끝나면 agent는 다시 $d$hop을 이동하고, 이 과정이 반복된다.

이러한 MM방식의 또 다른 장점은, $|S_{max}^t(\theta)|=1$일 때 자연스럽게 AN의 방식을 따른다는 점에 있다. 이는 MM이 diffusion방식에 기반을 둔 LB를 수행하기 때문이며, 이러한 성질은 AN방식을 그대로 수용할 수 있을 뿐 아니라 AN이 여러 번의 balancing step을 거쳐야 하는 상황을 보완해 줄 수 있다는 점에서 효율적이라 할 수 있다.

전체적인 동작 방식은, 최초에 agent가 일정한 hop수만큼을 각 node들의 정보를 수집하며 이동한다. Agent의 다음 node로의 이동은 nearest neighbor들 중에서, 무작위로 다음 hop에 진행할 node를 결정한다. Agent가 일정 hop이 되면 자신이 지금까지 지나온 node들을 nearest neighbor로 설정하고 MM의 기법을 사용하여 LB를 수행한다. 이러한 작업이 끝나게 되면 agent는 다시 일정한 hop수 d를 진행한 뒤 LB를 수행하며, 이러한 과정이 반복된다.

4. 성능 평가 및 결론

제안된 LB 기법의 성능을 평가하기 위하여, diffusion방식의 대

표적인 기법인 AN과 agent기반의 LB의 대표적인 기법인 Messor를 각각 평가 대상으로 삼았다. Simulation은 agent를 기반으로 한 P2P simulator인 Anthill framework[4]를 사용하였다. Network는 500개의 node로 구성되며, 각 node의 out-degree는 6으로, 각 node는 uniform distribution을 가지고 random하게 다른 node들과 연결된 connected graph 형식의 정적인 P2P network로 설정하였다. Agent를 사용하는 MM 및 Messor의 경우는 초기에 하나의 node에서 하나의 agent를 생성하도록 하였으며, 전체 network의 초기 workload 값은 전체 network의 표준 편차가 2000이 되도록 각 node마다 random하게 설정되었다. 하나의 simulation time은 agent가 한 node에서 다른 node로 이동하는 시간(agent가 1 hop을 이동하는 시간)을 의미한다. 또한 이 시간을 agent가 다른 node로 이동하는 것을 하나의 메시지가 전달되는 것으로 볼 때, 하나의 simulation time은 한 node에서 다른 node로 메시지가 전달되는 시간으로 나타낼 수 있다.

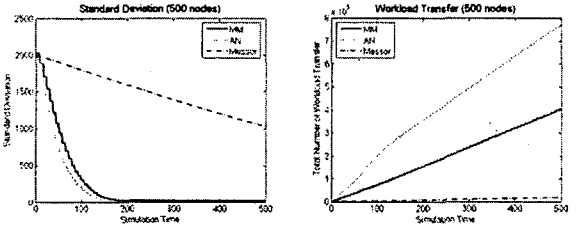


그림 3,4 : 전체 Network의 표준 편차(좌)및 총 전송 횟수(우)

그림 3은 각 알고리즘이 얼마나 빠르게 전체 network의 workload를 균등하게 만드는지를 나타내며, 이를 위한 workload의 총 전송 횟수는 그림 4와 같다.

그림 3에서, MM은 표준 편차가 AN과 비슷한 속도로 감소하지만 그림 4에서와 같이 전체 workload의 전송 횟수는 AN보다 더 적다는 것을 확인할 수 있다.

일반적으로 AN에서는, 각 node가 nearest neighbor들에게 자신의 workload에 대한 정보를 전파하고, 전체 node들이 정보 전파 작업을 끝내면 각자 자신이 받은 정보를 가지고 AN을 수행한다. AN이 수행된 뒤에는 다시 자신의 workload에 대한 정보를 전파해야 하며, 이 과정이 반복된다. 즉, 어떤 node i는 자신의 nearest neighbor에게 자신의 workload에 대한 정보를 전파하기 위하여, agent를 생성하여 nearest neighbor에게 보내게 된다. 이러한 정보 전파에 필요한 agent를 "제어 메시지"로 정의하였다.

MM과 messor에서의 제어 메시지는 agent가 각 node에 대한 정보를 누적하면서 network를 hopping할 때 필요한 메시지를 나타낸다.

그림 5 : 제어 메시지

그림 5에서 볼 수 있듯이 MM와 Messor는 각 node에 대한 workload를 계속해서 누적하면서 이동하기 때문에, 필요한 제어 메시지의 개수는 A에서 B로 이동할 때 1개, B에서 C로 이동할 때 2개(C에 A의 정보를 함께 전파), C에서 D로 이동할 때 3개(D에 A,B의 정보를 함께 전파)의 형태로, agent의 hopping을 제어 메시지를 가지고 나타낼 수 있게 된다.

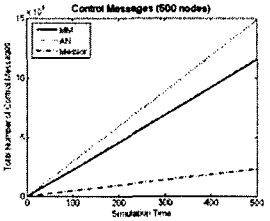
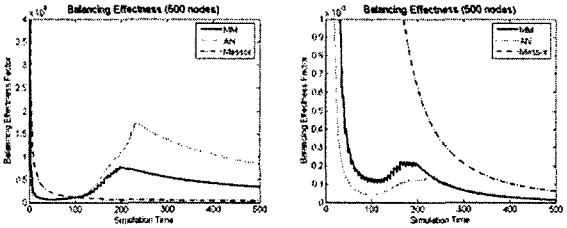


그림 6 : 제어 메시지의 총 전송 횟수

그림 6은 각 알고리즘에 대하여, simulation time이 진행함에 따라 누적된 제어 메시지의 개수를 나타내었다. 이러한 제어 메시지의 개수에 대한 그래프는 workload의 전송 횟수 그래프(그림 4)와 비슷한 모양을 가지며, MM이 AN보다 제어 메시지를 적게 보내면서 LB를 수행하는 것을 확인할 수 있다.

그림 7은 이러한 표준 편차와 전체 메시지와의 관계에 착안하여, 각 simulation time에 전체 메시지의 개수(전송 횟수+제어 메시지의 개수)를 β승 하여 표준 편차를 곱한 값에 역수를 취한 balancing factor라는 값으로 나타낸 그래프이다. 이는 현재 얼마나 효율적으로 LB가 수행되고 있는가를 나타내는 척도가 될 수 있으며 실험에서는 초기에는 MM이 AN보다 높은 효율을 보인 뒤, 200 time이 흐른 뒤에는 MM과 AN 모두 효율이 감소하는 모습을 볼 수 있다. 이는 그림 3과 비교하여 볼 때 전체적인 표준 편차가 매우 작은 값을 가진 후 부터는 MM과 AN의 효율이 감소하게 된다는 사실을 말해 준다. Messor의 경우는 전송 횟수가 매우 적으면서 LB가 수행되기 때문에 초기의 효율은 매우 높은 반면, 표준편차를 감소시키는 시간이 오래 걸리기 때문에 계속해서 효율이 감소하는 모습을 볼 수 있다.



(가) β=1일 경우 (나) β=3일 경우

그림 7 : β값에 따른 Balancing 효율

결과적으로, MM은 AN과 비슷한 성능을 보이면서 더 적은 workload의 전송 횟수와 적은 제어 메시지의 개수를 가지는 것을 확인할 수 있다. 또한 전체 메시지의 개수가 강조된 전송 효율의 측면에서는 AN보다 높은 효율로 LB를 수행하는 것을 확인할 수 있다. 따라서 본 기법은 표준 편차의 감소와 전체 메시지의 증가 사이에서의 적절한 trade-off를 제공하는 것을 알 수 있다.

추후에는 MM이 LB에 대한 속도 면에서 AN보다 약간 뒤처지기 때문에, MM기법을 개선하여 더 빠른 LB를 수행할 수 있도록 만드는 작업이 필요할 것이다.

5. 참고 문헌

[1] C. Xu and F. Lau, *Load Balancing in Parallel Computers*, Kluwer Academic Publishers, 1997.

[2] B. Godfrey, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, "Load Balancing in Dynamic Structured P2P Systems," *IEEE INFOCOM*, pp.2253-2262, 2004.

[3] A. Corradi, L. Leonardi, and F. Zambonelli, "Diffusive Load-Balancing Policies for Dynamic Applications," *IEEE Concurrency*, vol.7, pp.22-31, 1999.

[4] A. Montresor, H. Meling, and Ö. Babaoglu, "Messor: Load-Balancing through a Swarm of Autonomous Agents," *LNCS*, vol.2530/2003, pp.125-137, 2003.

[5] H. Arndt, "Load Balancing: Dimension Exchange on Product Graphs," In *Proc. of 18th Int. Parallel and Distributed Processing Symposium*, pp.20b, IEEE, 2004.

[6] F. C. H. Lin and R. M. Keller, "The Gradient Model Load Balancing Method," *IEEE Trans. on Software Engineering*, vol.13, pp.32-38, 1987.

[7] D. L. Eager, E. D. Lazowska, and J. Zahorjan, "Adaptive Load Sharing in Homogeneous Distributed Systems," *IEEE Trans. on Software Engineering*, vol.12, pp.662-675, 1986.

[8] G. C. Fox, A. Kolawa, and R.Willians, "The Implementation of a Dynamic Load Balancer," In *Proc. of Hypercube Multiprocessors*, pp.114-121, SIAM, 1987.

[9] I. Holyer, "The NP-Completeness of Edge-Colouring," *SIAM J. of Computing*, vol. 10, pp.718-720, 1981.