분산 데이타베이스에서의 동적 화일배정에 관한 연구

황영헌, 김대환, 김영호, 강석호
서울대학교 산업공학과

Abstract

We propose dynamic file allocation method in distributed database management system with changing access patterns.

There are a lot of studies on file allocation problem in D-DBMS, and those studies deal with off-line analysis and optimization. Those works are well for systems with static database access patterns, but are inadequate for systems that have changing access patterns. In these systems, dynamic file allocation along with access pattern is more proper.

In advance, Brunstrom et al. studied on this area, but they dealt a extremely simplified model. So, we make more practical models to simulate real system. In these models, many factors that were disregard in the advance study are considered. These models are composed with the non-replication system and the replication system. In addition to, we deal with CPU workload balancing in such system in order to improve performance of systems. Our methodology is very simple and realistic, therefore we think that it will give a lot of improvement in D-DBMS with changing access pattern.

1. 서론

분산 데이타베이스에서의 화일배정 문제(file allocation problem: FAP)는 시스템의 효율을 높이고, 통신량을 줄이기 위한 수단으로 매우 중요하게 다루어져 왔다. 이러한 화일배정 문제는 데이타베이스의 운용단계 이전인 설계단계에서 다루어졌으며, 데이타베이스에 대한 환경이 바뀌는 경우에 대한 대응도 off-line 상태에서 이루어지는 것이 일반적이었다. 이는 화일배정 문제가 NP-complete로 알려져 있으며[Eswaran], 제안된 휴리스틱들 또한 많은 계산량을 가지고 있기 때문이다. 화일배정을 운용 전 단계에서 수행하기 위해서는 데이타베이스가 운용될 때 각 site에서의 접근비율을 사전에 알 수 있고, 이 접근비율이 고정되어 있다는 가정이 필요하다. 그러나 많은 경우 이러한 가정은 잘 적용되지 않는다. 특히 각 site들로부터의 작업부하(workload)가 시간에 따라 변화하는 경우 화일을 수시로 재배정해 주는 것이 전체 분산 데이타베이스 시스템의 통신부하(communication load)를 줄이고, 질의에 대한 응답시간(response time)을 단축시키는 방법이 될 수 있다.

이러한 경우, off-line 상태에서 전체 화일을 수시로 재배정하는 것은 화일배정 문제가 많은 계산량을 가지고 있으며, 또한 화일을 재배치하기 위해서는 전체 시스템을 사용할 수가 없으므로, 본 연구에서는 각 화일에 대한 배정 문제를 해결한 후 이 화일만 재배정하는 방법을 전체 화일에 대해 수행하는 방법을 취하였다. 즉 전체 시스템의 최적화를 달성하기보다는 각 화일에 대한 부분 최적화를 시도함으로써 문제의 복잡도를 낮추고, 시스템의 가용성(availability)을 높일 수 있게 하는 방법을 말한다.

이러한 연구는 Brunstrom 등에 의해 처음 시도된 바 있다[Brunstrom et al.]. 그러나 이 연구에서는 화일의 동적 재할당이 데이타베이스의 성능을 향상시킨다는 것을 보이기 위해 문제를 지나치게 단순화하였다. 그러나 이들은 화일을 동적으로 재배정하는 것이 그렇지 않은 경우보다 시스템의 성능을 높인다는 것을 보이는데 중점을 두었으며, 이를 실험으로 실증하고 있다.

본 연구에서는 보다 현실을 잘 반영한 동적 화일 재배정 방안을 개발하였으며, 화일의 복사본과(replication)을 CPU의 작업부하를 고려하였다.

2. 분산 데이타베이스의 화일배정 문제

분산 데이타베이스에서의 화일배정 문제는 다수의 site와 화일이 존재하는 분산 데이타베이스에서의 최적 배정대안을 찾는 최적화 문제(optimization problem)이다. 이 때 최적화의 기준은 비용 최소화, 수행도 최대화 두 가지이다[Dowdy and Foster]. 비용이란 저장비용, 검색비용, 갱신비용, 통신비용 등을 가리키며, 수행도는 반응시간(response time), 처리량(throughput) 등을 가리킨다. 또한 제약식으로는 반응시간 제약, 저장용량 제약, 처리용량 제약 등이 있다[Özsu and Valduriez]. 그러나 비용과 수행도를 하나의 목적식으로 공식화(formulation)하는 것과, 많은 제약들을 모두 고려하는 것은 매우 어려운 문제이며, 제안된 대부분의 공식들이 분산 데이타베이스의 특징을 완벽히 고려하고 있지 못하다[Rothnie and Goodman]. 또한 이렇게 만들어진 문제는 비선형 0/1 정수계획법의 형태를 가지게 되어 문제를 해결하는 것도 실제로 가능하지 않다. 이로 인해 많은 휴리스틱들이 제안되고 있으며, 최근에는 유전 알고리즘 등 일반적인 휴리스틱 방법론을 배정문제에 적용하는 연구도 많이 이루어지고 있다.

본 연구에서는 복사본을 허용하지 않는 경우와 허용하는 경우를 모두 고려하였다. 분산 데이타베이스에서의 복사본을 허용하는 것은 수행도와 가용성(availability) 측면의 장점을 가지는 반면 복사본에 대한 동시성 제어(concurrency control)라는 부하(overhead)가 발생한다[Bell and Grimson]. 그러므로 복사본을 허용하는 경우는 이러한 비용과 효과에 대한 고려가 이루어져야 할 것이다. 복사본을 만드는 경우 갱신과 관련한 다양한 프로토콜이 있을 수 있는데, 시스템에 따라 또한 시스템 관리 전략에 따라 이러한 비용은 달라질 수 있을 것이다.

3. 복사본을 허용하지 않는 동적 화일 재배정

동적 화일 재배정 문제를 다루기 위해서는 우선 화일에 대한 각 site에서의 통신량을 계산할 필요가 있다. 이 통신량은 질의와 질의에 대한 결과의 전송, 또는 JOIN 등의 연산(operation)을 위한 전송 모두를 포함한다. 이를 계산하기 위해 각 화일은 전체 시스템으로부터의 통신량을 계산하기 위한 적산기(accumulator)를 가지도록 하였다. 이 적산기는 N개의 화일, M개의 site가 있을 때 모두 N×M개가 존재하게 되며, 각 적산기는 화일이 있는 site에 존재하게 된다.

화일의 재배정을 모든 화일에 대해 주기적으로 수행할 수도 있지만 화일에 대한 접근량을 고려하여 접근이 많은 화일에 대해 보다 자주 재배정을 수행하는 것이 전자 보다 계산량을 줄일 수 있으며, 변화를 빨리 반영할 수 있다는 장점이 있으므로 본 연구에서는 후자의 방법을 선택하였다. 후자의 방법을 사용하기 위해서는 일정한 수준의 통신량을 미리 정해 두고 이 값에 다다른 화일에 대해 재배정 문제를 수행해야 할 것이다.

배정문제를 풀면 현재의 site가 최적인 경우와 새로운 site가 최적인 경우가 생기게 되는데, 전자의 경우는 화일을 움직이지 않고 적산기의 값만 초기화한다. 새로운 site가 결정되면 아래와 같은 순서로 화일을 이동하게 된다.

1. 현재 화일의 snapshot을 생성시킨다. snapshot을 생성한 시점부터 화일을 수정하는 operation에 대한 디퍼렌셜 화일(differential file)을 만든다.
2. 이 snapshot을 새로운 site로 전송한다.
3. 전송이 완료되면 전체 site에 "site change of file f_i" 메시지를 방송(broadcasting)한 후, 현재의 화일에 잠금(lock)을 한다.
4. 기록된 디퍼렌셜 화일을 새로운 site로 전송한 후 이 화일을 통해 snapshot을 수정한다. 이 때 기존의 화일과 디퍼렌셜 화일은 삭제한다.
5. 끝나면 "change commit" 메시지를 전체 site에 방송한다. 적산기를 생성하고 계산을 시작한다.

여기서 디퍼렌셜 화일이란 화일의 내용에 변화를 줄 수 있는 내용만을 기록한 화일을 가르키는데, snapshot을 먼저 전송한 후 나중에 디퍼렌셜 화일을 전송하는 것은 화일전송에 따르는 잠금시간(locking time)을 최소화하기 위한 것이다[이우기]. 배정문제를 푸는 식은 아래와 같이 구성(formulation)할 수 있다.

$$Min_j \ (c_{sj} + a_j)f_i \ + \ \Sigma_m c_{mj}v_{im} \ + \ \delta_j d_i w_i$$

s : 현재 화일이 위치한 site

c_{ij} : i site에서 j site로의 통신비용

$c_{ij}=0$, $i=j$

$c_{ij}>0$, otherwise

a_j : j site의 저장비용

f_i : 화일 i의 크기

v_{im} : 화일 i에 대한 site m으로부터의 통신량

d_i : 화일 i에 대한 잠금비용

w_i : 화일 i에 대한 단위시간당 통신량

$$w_i = \frac{\Sigma_m v_{im}}{\Delta t}$$

Δt : 화일 i에 대한 배정문제 푸는 시간간격

$\delta_j = 0$, $j = s$

$= 1$, otherwise

각 항목의 의미는 다음과 같다. 첫 번째 항은 화일 i를 site s에서 site j로 이동시킬 때 화일의 이동으로 인해 발생하는 통신비용($c_{sj}f_i$)과 저장비용(a_jf_i)에 대한 합이다. site가 변하지 않는 경우에 화일의 이동으로 인한 통신비용은 0이 될 것이다. 두 번째 항은 화일 i에 대한 각 site로부터의 질의(query)에 대한 비용을 의미한다. 여기서 사용된 v_{ik}는 적산기가 계산한 질의에 대한 누적값을 의미하는데, 이 값은 분산 데이타베이스에서 사용되는 중요한 개념인 위치 투명성(location transparency)에 의해 화일이 저장된 site의 위치와 무관하다. 마지막 항은 잠금(locking)과 관련한 비용이다. 이 비용은 화일의 크기보다는 디퍼렌셜 화일의 크기와 질의의 양의 곱에 비례한다. 그러나 디퍼렌션 화일은 화일을 전송하는 동안에 발생하므로 크기를 사전에 알 수 없으며, 디퍼렌셜 화일의 전송시, 즉 잠금이 일어나는 순간에 도착하는 질의의 양을 추정하는 것도 도착 질의의 수를 Poisson 분포를 따른다고 할 때 상대적으로 짧은 시간에서의 추정은 의미가 없다. 그래서 여기서는 잠금과 관련한 비용을 단위시간당 질의의 양에 비례한다고 가정하였다. 또한 화일마다 잠금비용이 다를 수 있다고 가정하였다. 물론 이 비용은 화일이 이동하지 않는 경우에는 발생하지 않으므로 δ_j를 곱하였다.

위의 식에서 구해진 j가 화일 i가 새로 배정될 site가 된다. 이 식에 사용된 가정들은 아래와 같다.

1. 디퍼렌션 화일의 전송량은 다른 자료의 전송량에 비해 충분히 적다고 가정하고 생략하였다. 실제로 디퍼렌션 화일에는 화일의 내용을 갱신하는 경우만 저장하므로 디퍼렌션 화일은 상대적으로 적고, 상대적으로 짧은 시간 동안만 기록되기 때문에 이러한 가정은 의미가 있다.
2. 각 site에는 화일을 저장할 수 있는 충분한 공간이 있다고 가정하였으며, 또한 CPU의 작업균형 문제로 고려하지 않았다. 작업균형을 고려한 문제는 5장에서 다루었다.
3. 화일의 복사본은 허락하지 않았다. 복사본을 허락한 문제는 다음 장에 다루었다.
4. site의 장애나 통신장애는 없거나 무시할 수 있다고 가정하였다. 이러한 가정은 비현실적이지만 이 연구의 목적이 작업부하가 변화하는 경우 동적으로 화일을 재배정하는 것이 옳다는 것을 보이는 것이 목적이기 때문이다.
5. 각 site간 전송경로는 반드시 존재한다고 가정하였다. 이러한 가정은 가정4에서 암시된다. 또한 각 site간 전송비용 및 화일에 대한 저장비용 등은 사전에 계산되었다고 가정하였다.

위의 식을 통해 화일을 배정할 새로운 site를 결정하여 화일을 새로 배정함으로써 통신부하를 줄일 수 있게 된다. 위에서 소개한 식은 site의 수가 n개인 경우 $O(n^2)$의 복잡도를 가지므로 별도의 휴리스틱을 필요로 하지 않는다. 기존의 화일배정 문제가 NP-complete임에도 동적 배정문제가 이러한 복잡도를 가지는 것은 전체적인 최적화(global optimization)를 추구하지 않았고, 시스템의 운용시 빠른 시간에 문제를 풀 수 있도록 하기 위하여 제약식을 만들지 않았기 때문이다. 사실 화일배정 문제는 대개 비선형 0/1 정수 계획법의 형태를 나타내게 되지만, 본 연구에서는 문제를 단순화시켰다. 그

러므로 이 방법론을 통해 화일들이 지나치게 편중되는 경우 전체 시스템의 성능은 떨어질 수도 있으며, 기억장소의 부족 문제도 발생할 수 있을 것이다. 이를 막기 위해서는 별도의 배정문제를 고려해야 하는데, CPU의 작업부하를 고려한 화일 재배정 문제는 5장에 소개되었다.

4. 복사본을 허용하는 화일 재배정

복사본을 허용하는 분산 데이타베이스 시스템에서는 크게 두 가지 방식이 있다. 먼저 주화일(master file)이 별도로 있고, 이를 복사한 종화일(slave file)이 있어서 주화일은 갱신이 가능하지만 종화일은 읽기만 가능하게 하는 경우가 있다. 다음으로 분산 갱신제어(distributed update control)의 경우가 있는데, 이 경우는 모든 복사본에 대해 갱신과 수정을 할 수 있는 경우를 말한다. 그러나 후자의 경우가 보다 보편적이므로 본 연구에서는 이를 택하였다.

이 연구에서는 이러한 복사본의 생성을 화일의 재배정 문제와 같이 동적으로 수행하고자 한다. 즉 변화하는 작업부하를 가진 시스템의 수행도를 증가시키기 위해 필요한 경우 화일의 복사본이 새롭게 생성될 수도, 기존의 복사본이 삭제될 수도 있도록 하겠다는 것이다.

수행 절차는 복사본을 허용하지 않는 경우와 유사하다. 단 복사본이 존재하므로 화일을 전송하는 site는 통신비용이 적은 site를 택한다. 복사본을 허용하는 배정문제를 풀면 복사본의 수가 변하지 않는 경우, 복사본의 수가 줄어드는 경우, 증가하는 경우가 생기게 되는데, 각 경우에 대한 수행절차는 큰 차이가 없다.

화일의 복사본의 수와 복사본이 저장될 site를 결정하기 위해 본 연구에서는 한 번에 계산하지 않고 아래와 같이 복사본의 수를 하나씩 증가시키면서 해결하는 방법을 사용하였다.

1. 복사본이 하나인 경우의 해를 구한 후, 이 때의 값을 C_1이라고 둔다.
2. 복사본의 수를 하나 증가시킨 후(복사본의 수: i) 해를 구한 후, 이 때의 값을 C_i라고 한다.
3. $C_{i-1} > C_i$이면, 2번으로 간다.
4. $C_{i-1} \leq C_i$이면, 복사본의 수를 i-1개로 하고, 이 때 구해진 site에 화일을 배정한다.

이 때 각 복사본들은 자신의 적산기를 가지며, 재배정 문제를 풀 때 복사본에 대한 통신량 데이타는 하나의 논리적 화일(logical file)에 대한 값으로 합산될 수 있으므로 화일에 대한 통신량은 복사본을 허용하지 않을 때와 마찬가지로 v_{ik}(여기서 i는 논리적 화일을 가리킴)를 사용하였다. 식은 아래와 같이 구성된다.

먼저 복사본이 하나일 경우는 아래와 같다.

$$Min_j(\ c_{sj} + a_j)f_i + \Sigma_m c_{mj}v_{im} + \delta_j d_i w_i$$

c_{ij} : 현재 site들로부터 j site로의 최소통신비용
즉 $c_{sj} = Min(\ c_{s_1 j}, c_{s_2 j}, \ldots c_{s_N j})$,
여기서 $s_1, s_2, \ldots$는 복사본이 저장된 site

$\delta_j = 0$, $j = s_1$ or $j = s_2$ or $\ldots$ $j = s_N$
$= 1$, otherwise

이 식은 별도의 복사본 프로토콜(replica protocol)이 필요치 않으므로 3장에서 제시한 식과 유사하다. 단 기존에 복사본이 존재하고 있을 수 있으므로 화일의 전송은 옮겨갈 site과의 통신비용이 가장 작은 site에서 담당하게 하여 이 때의 비용을 통신비용으로 하였다. 이를 위해 c_{sj}를 사용하였다. 또한 δ_j 역시 3장과는 달리 복사본의 경우를 고려하였다.

복사본이 두 개일 경우,

$$Min_{j,k,j+k}(\ c_{sj} + c_{sk} + a_j + a_k)f_i + \Sigma_l\ c_l v_{il} + \delta d_i w_i + k_2 w_i$$

c_l : l site와 복사본이 있는 site와의 최소통신비용,
여기서는 $c_l = Min(c_{jl}, c_{kl})$

$\delta = 0$, $i, j \in \{s_1, s_2, \ldots, s_N\}$
$= 1$, otherwise

k_n : n개 복사본의 동시성 제어를 위한 단위부하

위 식에서 마지막 항은 복사본을 유지하기 위해 추가되는 부하량을 의미한다. 이에 대해 설명하면 다음과 같다.

복사본을 허용하는 경우 복사본을 관리하기 위한 여러 종류의 복사본 프로토콜이 있다. 이런 프로토콜들은 검색질의(read query)와 기록질의(write query) 각각에 대해 수행도를 높이는 동시에 동시성 제어(concurrency control)를 달성할 수 있는 기법들을 제공하고 있다. 분산 데이타베이스 시스템에서는 각 site에 있는 server에게 제출되는 질의는 트랜잭션 조정자(transaction coordinator: TC)에 의해 복사본이 있는 site의 트랜잭션 관리자(transaction manager: TM)로 보내지며 이 때 동시성 제어를 위한 많은 부가적인 통신이 이루어지게 된다. 이러한 부가적인 통신량은 프로토콜에 따라 다르며, 동일한 프로토콜에서도 복사본의 수, 통신량, 검색질의/기록질의 여부에 따라 많은 차이를 보인다. 이러한 모든 것을 고려하기 위해서는 많은 정보가 추가적으로 필요하게 된다. 본 연구에서는 추가적인 정보를 계산하기 위한 기능들을 추가하는 대신 복사본의 수에 따라 증가하는 단위부하 값을 이용함으로써 단순화하였다. 위의 식에서는 단위부하의 값과 화일에 대한 단위시간당 통신량을 곱함으로써 복사본에 대한 부하량을 계산하였다. 그러나 동시성 제어의 방법으로 많이 사용하는 잠금기법(locking method)의 경우, 부하량은 질의량이 어느 한도를 넘어서면 급격하게 증가하는 경향을 나타낸다. 이를 표현하기 위해서는 단순히 단위부하의 값과 통신량을 곱하는 대신 다른 식을 사용할 수 있을 것이다.

복사본이 세 개 이상인 경우는 두 개의 경우와 유사하게 표현할 수 있으므로 생략한다.

이 문제의 복잡도는 site의 수가 n개일 때 $O(2^n)$이나, 실제 분산 데이타베이스 시스템에서는 동시성 제어를 위한 부하의 증가로 복사본의 수가 3개 이상 되는 경우는 그리 많지 않으므로 문제의 복잡도는 크지 않다고 볼 수 있으므로, 역시 휴리스틱을 제안할 필요성은 없다고 본다. 이 문제 역시 CPU의 작업부하 문제를 고려하지 않았다.

5. 작업부하를 고려한 화일 재배정

위에서 제안한 화일 재배정 문제들은 화일들의 편중으로 인한 CPU 작업부하의 불균형 문제를 발생시킬 수 있다는 단점이 있다. 작업부하의 균형문제를 해결하기 위해서는 화일의 배정시 CPU부하를 고려해 주는 방법을 사용할 수 있다. 그러나 데이타베이스에서의 연산이 하나의 화일을 상대로 이루어지는 경우 보다 여러 화일이 관계되는 경우가 많으

며, 또한 여러 응용 프로그램이 동시에 분산 시스템에서 사용되는 경우가 대부분이므로 각 화일과 관련된 CPU 시간을 정확히 계산한다는 것은 불가능하다. 그러므로 본 연구에서는 작업부하가 동적으로 변화하는 분산 데이타베이스 시스템의 특성을 고려해 주기 위해 위에서 계산한 화일에 대한 단위 시간당 통신량(w_i)에 대한 데이타와 시스템이 제공하는 CPU 이용률에 대한 데이타를 이용하여 작업부하 균형을 고려하고자 하였다.

[표1] CPU 작업부하 균형화를 위한 데이타(예)

Site	화일	단위시간 당통신량 (w_i)	전체단위 시간당 통신량 (w_{T_i})	전체 이용률 u_{T_i}:%	DB 이용률 u_{D_i}:%	통신량 대비 이용효율 u_{D_i}/w_{T_i}
A	1	2	10	80	40	4.0
	2	3				
	3	5				
B	4	3	8	60	20	2.5
	5	5				

예를 들어 [표1]과 같은 데이타가 주어질 때 화일 3이 site 1에서 site 2로 옮겨간다고 하면, site 1의 이용률이 20.0(=5×4.0)이 줄어드는 대신, site 2에서의 이용률이 12.5(=5×2.5) 만큼 늘어난다고 계산할 수 있다. 물론 이러한 계산이 실제와는 다소 차이가 있을 수 있지만 이러한 계산을 통해 CPU의 부하 균형화를 달성할 수 있다.

작업부하를 고려한 화일의 재배정 문제는 아래와 같이 공식화할 수 있다. 여기서는 복사본을 허용하지 않는 경우만을 다루었는데, 이는 복사본을 고려하는 경우도 4장에서 설명한 공식을 통하여 비교적 간단히 수정할 수 있기 때문이다.

CPU의 작업부하를 고려한 배정문제를 다루기 위해 본 연구에서는 아래와 같이 벌칙점을 계산하였다. 여기서 $\overline{u_{T_i}}$는 사전에 정해진 최적 CPU 부하 배정 목표를 의미한다. 즉 이 값과의 차이를 줄여주기 위해 아래와 같은 벌칙점을 계산한 것이다.

$$P = p(\Sigma_{m \neq j,s} |u_{T_m} - u_{T_m}| + |\overline{u_{T_i}} - u_{T_i} - \frac{u_{D_i}}{w_{T_i}} \times w_i| + |u_{T_j} - u_{T_j} + \frac{u_{D_i}}{w_{T_j}} \times w_i|)$$

p : 부하 배정 목표와의 차이에 대한 벌칙점 상수
u_{T_i} : 최적 CPU 부하 배정 목표

여기서는 절대치를 사용하였으나 절대치의 제곱근이나, 제곱치를 사용할 수도 있을 것이며, 최적 부하 배정 목표 보다 커지는 경우만 벌칙점을 부여할 수도 있다. u_{T_i}는 각 server의 계산능력이나 다른 응용 프로그램의 상황을 고려하여 구할 수 있다.

작업부하를 고려한 배정문제를 해결하기 위해 기존의 공식에 위에서 구한 벌칙점을 추가하였는데, 이는 아래와 같다.

$$Min_j(c_{sj} + a_j)f_i + \Sigma_m c_{mj} v_{im} + \delta_j d_i w_i + P$$

이러한 방법은 제약식을 추가함으로써 문제를 복잡하게 하는 것을 피하기 위함이며, 이는 짧은 시간에 화일의 재배정 문제를 해결하고자 하는 이 연구의 목적과 일치한다. 여기서는 저장공간을 고려하지 않았지만 저장공간을 고려하는 경우도 위와 같은 방법을 사용할 수 있다.

이 공식을 보면 벌칙점 상수 p의 크기는 중요한 의미를 가지게 된다. 즉 p가 0이 되면 작업부하를 고려하지 않는 시스템이 되며, 반대로 ∞가 되면 화일의 이동이 전적으로 최적 CPU의 부하 배정 목표에 의존하게 되는 시스템이 될 것이다.

6. 결론

이 연구를 통하여 작업부하가 변화하는 분산 데이타베이스 시스템의 통신비용을 줄이기 위한 방안으로 간단한 방법론을 사용한 화일 재배정 문제를 제시하였다. 또한 이 때 발생하는 CPU의 작업부하 편중을 막기 위해 부하 평준화 문제도 다루었다.

본 연구에서는 분산 데이타베이스에서 고려해야 할 많은 문제를 단순화시켰다. 이는 분산 데이타베이스에서 고려되는 많은 문제들은 그 자체만으로도 많은 분량을 차지하기 때문이며, 또한 이 연구의 목적이 짧은 시간에 화일 재배정 문제를 해결하게 하는 것이 목적이므로 모델의 단순화는 불가피하다고 할 수 있다.

앞으로 이 연구에서 제시한 방법이 주는 효과를 검증하기 위해 시뮬레이션 프로그램을 사용하거나 실제 시스템을 사용한 검증과정을 거쳐야 할 것이다. 또한 주로 3장, 5장에서 논의된 다양한 화일 재배정 대안들에 대한 비교분석을 통하여 가장 적합한 배정방법을 찾아야 하며, 객체지향 데이타베이스 시스템(OODBMS)이나 최근 많은 관심을 끌고 있는 객체관계형 데이타베이스 시스템(ORDBMS)들에 적용할 수 있는 방법론이 연구되어야 할 것이다.

참고문헌

1. 이우기, 분산 데이타베이스 구현을 위한 레플리케이션 서버 체계에 관한 연구, 박사학위 논문, 서울대학교 산업공학과, 1996.
2. Bell, D. and J. Grimson, Distributed Database Systems, International Computer Science Series, 1992, pp.201-202.
3. Brunstrom, A., S. T. Leutenegger and R. Simha, "Experimental Evaluation of Dynamic Data Allocation Strategies in a Distributed Database with Changing Workloads", Conference Information and Knowledge Management, Baltimore, 1995, pp.395-402.
4. Dowdy, L. W. and D. V. Foster, "Comparative Models of the File Assignment Problem", ACM Computer Survey, 1982, 14(20), pp.287-313.
5. Eswaran, K., "Placement of Records of a File and File Allocation in a Computer Network", IFIP Conf., 1974, pp.304-307.
6. Rothnie, J. B. and N. Goodman, "A Survey of Research and Development in Distributed Database Management," Proc. 3rd Int. Conf. on VLDB, Tokyo, Japan, 1977, pp.48-62.
7. Özsu, M. T. and P. Valduriez, Principles of Distributed Database Systems, Prentice-Hall, 1991, p.137.