韓國 컴퓨터情報學會 論文誌
第8卷 第1號, 2003. 3.

2003-8-1-1-8

적응적 중복 객체 알고리즘을 이용한 객체 복제본 관리 연구

박 종 선* 장 용 철** 오 수 열***

The Study of the Object Replication Management using Adaptive Duplication Object Algorithm

Jong-sun Park* Yong-chul Jang** Soo-yeol Oh***

요 약

분산 객체 복제 시스템에서 노드들이 공유하는 객체는 동일 내용을 복수 노드에 위치시키는 것이 효율적이다. 노드들은 시스템에 접근시 접근 정보를 자신의 지역 캐시에 저장해 두었다가 필요시에 인출해서 사용한다. 그러나 시간이 지나감에 따라 다른 노드들에 의해서 데이터의 갱신이 이루어지기 때문에 일관성 문제가 발생한다. 따라서 시스템의 일관성 유지를 통해 성능 및 가용성을 높이기 위해서는 객체를 효율적으로 관리하는 메커니즘이 필요하다.

본 논문에서는 공유 메모리 환경에서 일관성 유지를 위해 객체 관리시 기존의 중복 기법에서 사용하는 일관성 비용 외에 부가 비용이 없이도 제한적으로 병렬 수행의 효과를 얻으며, 또한 중복 기법에서 가장 큰 오버헤드로 알려진 일관성 유지비용을 최소화시키기 위하여 이 비용을 결정하는 가장 핵심적인 요소인 객체 복제본의 수와 위치 그리고 각 객체 사이의 상관도를 고려하여 객체를 효율적으로 관리하고, 전체 수행 시간을 개선시키는 적응적 중복 객체 관리 메커니즘을 연구한다.

Abstract

It is effective to be located in the double nodes in the distributed object replication systems, then object which nodes share is the same contents. The nodes store an access information on their local cache as it access to the system. and then the nodes fetch and use it, when it needed. But with time the coherence problems will happen because a data can be updated by other nodes. So keeping the coherence of the system we need a

* 목포과학대학 컴퓨터정보과 조교수
** 목포과학대학 컴퓨터정보과 부교수
*** 목포대학교 정보공학부 컴퓨터공학전공 교수

접수일자 : 2003. 1.20
심사완료 : 2003. 3. 8

mechanism that we managed the object to improve the performance and availability of the system effectively.

In this paper to keep coherence in the shared memory condition, we can get the limited parallel performance without the additional cost except the coherence cost using it to keep the object at the proposed adaptive duplication object(ADO) algorithms. Also to minimize the coherence maintenance cost which is the biggest overhead in the duplication method, we must manage the object effectively for the number of replication and location of the object replica which is the most important points, and then it determines the cost, And that we must study the adaptive duplication object management mechanism which will improve the entire run time.

Ⅰ. 서론

다중 프로세서(multi-processer) 시스템이 최근의 컴퓨터 분야에서 중요한 기술로 부각된 것은 고성능과 그에 비하여 저렴한 비용, 실생활의 응용 프로그램들에서의 생산성 등에 대한 계속적인 요구 증가에 기인한 것이다.[1].

일반적으로 다중프로세서 시스템은 프로세서들 간이나 프로세서와 메모리간에 통신하는 방법에 따라 메시지 전달 구조와 공유 메모리 구조로 분류된다. 메시지 전달 방식의 시스템에서는 분리된 각각의 컴퓨터들이 서로간에 명시적으로 메시지를 전달하는 방법을 통해서만 통신을 한다.

공유 메모리 방식의 시스템에서는 프로세서들은 강결합성으로 연결되어 있고 모든 프로세서들은 메모리에 접근할 수 있고 통신은 공유 메모리에 놓여지는 공유 변수나 메시지에 의해 이루어진다.[2]

최근 연구에서는 분산 공유 메모리 모델에서 물리적인 분산 메모리 모델을 논리적인 공유 메모리 모델처럼 구현함으로써 두 방식의 장점을 유지하는 시스템의 효율적인 객체 복제 관리 시스템이 제안되고 있다[3]. 이는 다른 프로세서와의 통신을 숨김으로써 분산 메모리 시스템의 장점인 확장성을 유지하고, 또한 공유 데이터 객체에 대하여 캐시의 기능을 제공함으로써 불필요한 통신을 줄여주기 때문이다.

본 논문에서 분산 공유 메모리 모델에서 제안한 적응적 중복 객체(ADO : Adaptive Duplication Object) 알고리즘에 의한 데이터 객체 복제 관리를 통한 일관성을 유지하는 방안이다.

분산 객체 시스템에서 객체를 어느 노드에 배치시킬 것인가 하는 것으로 해당 객체를 사용하는 응용 프로그램의 성능 및 신뢰도의 요구 조건을 만족시키기 위하여 객체의 복제본을 2개 이상 유지하는 방법으로 요구 수준을 충족시키는데, 이 때 각 복제본들은 그 내용에 있어서 원본과의 일치를 유지해야 하는 일관성 유지 문제 해결을 통해 시스템 성능을 개선하고자 한다.

Ⅱ. 객체 복제 관리 시스템

데이터의 중복성을 가진 복제 방법은 각각의 클라이언트들이 각 인접 서버에게 서비스를 요청하고 각 인접 서버들은 특정 서버에게 서비스를 요청하게 되고, 특정 서버는 각각의 인접 서버에게 서비스 객체를 복제하여 주게 되며, 각 노드들은 인접 서버로부터 서비스를 제공받는다. 따라서 각각의 노드들은 각각의 인접 서버로부터 서비스를 제공받아 서비스 속도의 지연과 특정 서버의 오버헤드를 줄일 수 있게 되며, 병목현상도 방지할 수 있다.[4]

객체 복제 방법을 이용한 분산 객체 시스템은 광역 네트워크 망에 연결된 노드들은 자신이 속해 있는 지역 서버에 접근하여 객체 복제 기법에 의해서 구축된 객체 정보를 이용하여 최신의 정보를 획득하고, 지역 서버를 통해서 클라이언트가 원하는 데이터가 존재하는 최단 거리의 서버로 접근하게 된다[9]. 객체 복제 관리 시스템은 객체 관리 모듈을 두고 복제된 객체를 관리한다. 객체 관리 시스템에서 일관성 유지 정책을 결정하는 데 가장 큰 영향을 미치는 요인은 공유 메모리와 프로세서 사이의 연결 망의 형태이다.

본 논문에서는 Wilson에 의해 제안된 계층적 캐시 버스의 일관성 유지 모델을 사용하며, 이는 프로세서에 직접적으로 연결된 작지만 빠른 속도의 하위 레벨 캐시들이 공유버스로 연결되어 하나의 클러스터를 구성한다. 이 클러스터들이 모여 약간 느리지만 커다란 크기의 상위 레벨 캐시에 연결되어 전역 버스를 통해서 메모리 모듈과 연결된다. 다단계 캐시 버스 구조는 [그림 1]과 같다[5].

계층 레벨 캐시에서 프로세서에 의한 참조는 그 프로세서에서 가장 가까운 캐시가 제공한다. 모든 접근은 실제 주소에 의해서 이루어진다고 가정하면 다중 레벨 캐시 계층의 특성을 갖게 되는데 계층적 캐시로 구성된 네트워크 구조에서 캐시 일관성의 복잡도를 감소시키는데, 필수적인 특성으로 상위 레벨 캐시의 내용은 입출력과 캐시 일관성 간섭으로부터 하위 레벨 캐시들을 보호할 수 있도록 하위 레벨 캐시의 내용을 포함해야 한다.

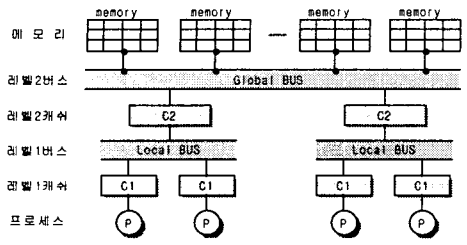

그림 1. 일관성 유지 모델
Fig. 1 Consistency maintenance model

Ⅲ. 적응적 중복 객체 알고리즘 설계

객체 복제 관리 시스템에서 중복 기법은 동일한 내용의 객체를 시스템 내에 하나 이상 유지시키는 방법으로 복수 개로 유지되는 객체 중 처음부터 존재하던 객체를 원본이라 하고 복제되어 유지되는 객체를 복제본이라 한다. 작업이 발생 시 각 작업들은 여러 곳의 객체들을 이용하게 된다. 순수한 작업 수행을 제외한 노드의 검색 및 접근 비용을 통신비용 tcomm이라 하면, 원격지에 존재하는 객체를 사용하는 비용 tremote은 식 (1)과 같다.

$$tremote = texec + tcomm \quad (1)$$

여기서 texec는 해당 객체의 연산을 수행하는데 소요비용으로 객체의 연산이 수행되는 노드의 복제본 수가 많을수록 통신비용은 적게 소요된다. 만약 사용할 객체가 작업을 수행하는 노드 내에 존재한다면 tcomm=0 이므로, tlocal은 texec와 같을 것이다. 또한 k개의 노드 n1, n2, n3 · · · nk로 이루어지는 시스템에서 ni에 이상이 발생할 확률을 f(ni)이라 한다면(여기서 $0<f(ni)\leq 1$이다) n1이 존재하는 객체 On1에 이상이 발생할 확률 또한 f(ni)으로 볼 수 있다.

이 객체의 복제본을 n1, n2, n3, n4에 위치하여 둔 경우는 노드 n1에 이상이 발생하더라도 나머지 노드들만으로 작업의 수행이 가능하며 최대 전체 복제본의 수-1개의 노드에 결함이 발생하여도 다른 정상 노드에서는 객체를 사용할 수 있게 된다. 그러나 n1,n2, n3, n4의 노드에 모두 이상이 발생하였을 경우에는 해당 객체를 사용할 수 없게 되는데 이 때의 확률은 식 (2)와 같이 매우 작게 나타나며, 따라서 높은 가용성과 결함 허용을 통하여 성능의 향상을 기대할 수 있다.

$$f(n1, n2, n3, n4) = n1 \cdot n2 \cdot n3 \cdot n4 \quad (2)$$

객체의 중복을 이용하게 되면 네트워크의 단절 현상이 발생한 경우에도 불구하고, 부분적으로는 비 연결 연산의 수행을 가능하게 한다.

객체를 사용하는 작업이 발생한 노드 상에 해당 객체의 복제본이 존재할 확률이 높기 때문에 노드간에 이루어지는 원격지 호출 비율이 낮아지므로 노드 내의 객체에 대한 호출이 하나의 노드로 집중되기보다는 복제본을 유지하고 있는 복수 개의 각 노드들로 분산되기 때문에 부수적으로 노드간의 부하 균등 효과를 기대할 수 있다. 예를 들어 병렬 수행되는 비율을 $\lambda\rho$라 하면, 읽기 작업의 전체 수행의 기대 시간 tread은 식 (3)과 같다.

$$tread \fallingdotseq (1 - \lambda\rho) \cdot texec \quad (3)$$

일관성 유지비용은 해당 시스템에서 사용하는 일관성 유지 기법이나 시스템에서 수행되는 작업의 특성, 서로 다른 노드들에 존재하는 복제본의 수, 복제본을 가지는 노드들의 부하 및 성능, 분산 시스템의 구성 형태 등에 따라 크게 달라지므로 정확한 측정값을 얻기는 어렵지만 이들 중 가장 큰 영향을 미치는 항목은 복제본의 수와 각 중복 노드의 부하이며 일반적으로 일관성 관리비용은 복제본의 수에 따라 기하급수적으로 증가한다.

작업의 갱신 비율에 있어서 잠금 기법의 비용과 일관성 유지의 비용을 구분하여 고려하는 것이 큰 의미를 가지지 못하므로 편의상 두 비용의 합을 일관성 유지비용이라 한다. 갱신 작업에 있어서 기대되는 작업의 수행시간 twrite은 호출 작업의 수행시간 texec과 일관성 유지비용 tcons에 영향을 받으므로 식 (4)와 같다.

$$twrite = texec + tcons \quad (4)$$

각 노드들이 같은 객체를 호출하면서도 어떤 시점에서는 읽기 전용 작업들이 주를 이룰 것이고 다른 시점에서는 쓰기 작업이 주를 이루거나 읽기·쓰기 작업이 상호 교대로 일어날 수도 있다. 작업의 동적 특성을 무시하고 전

체적인 평균 갱신 비율을 가지고 해당 시스템의 효율적인 복제본의 수를 결정하는 것은 결코 바람직한 방법이라고 할 수 없다[7].

일관성 유지비용의 측면에서 볼 때 효율적인 복제본의 수는 정적으로 정해지는 것이 아니라 객체의 이용 상태에 따라 동적으로 결정되어져야 한다[8]. 객체의 메소드를 수행하는 데에는 일관성 유지비용만이 영향을 미치는 것이 아니므로 다른 요소를 고려하여 전체 수행 시간과 복제본의 수와의 관계를 살펴보기로 하자.

객체 O에 발생하는 메소드 호출 중 갱신을 요하지 않는 작업 Mr이 발생할 확률을 λread라 할 때, 갱신을 요하는 작업 Mw가 발생할 확률은 $1-\lambda$read이다. Mr의 수행 시간, Er은 노드 내 호출인 경우와 원격 호출인 경우로 분류, $\rho 1$을 작업이 발생한 노드에 객체의 복제본이 존재할 확률, Ttr을 객체 호출이 발생한 노드와 실제 객체가 존재하는 노드 사이에 메시지를 전송에 소요되는 시간, Tr을 메소드 수행하는데 실제 필요한 시간, Mr의 수행시간 Er은 식 (5)와 같다.

$$Er = P1(Tr) + (1 - P1)(2Ttr + Tr) \quad (5)$$

Mw는 갱신 작업으로서 모두 중재자를 통하여 수행하게 되고 잠금 비용과 일관성 유지비용의 부가 비용이 발생한다. Tw를 호출한 메소드의 실제 수행시간으로 정의하고, Ln은 갱신의 직렬화를 위하여 n개의 노드에 잠금 관리를 하기 위한 시간, Cn은 n개 중복 노드 사이에 일관성을 유지시키기 위해 소요되는 시간이라 할 때, Mw의 수행 시간인 Ew는 식 (6)과 같다.

$$Ew = Ln + Tw + Cn \quad (6)$$

식 (6)을 기초로 하여 전체 수행 시간을 나타내면 식 (7)과 같다.

$$.E \fallingdotseq (\lambda\text{read})(1-\lambda\rho)(K1\text{-}2\rho 1Ttr)+(1-\lambda\text{read})(Ln+Cn+K2) \quad (7)$$

중복 기법을 적용하는데 있어서 복제본의 수를 증가시킬수록 작업이 발생한 임의의 노드에 해당 객체가 존재할 확률 $\rho 1$이 높아지고, 또한 읽기 모드 호출이 병렬로 수행될 확률 $\lambda\rho$도 증가하므로 결국 전체 읽기 작업 수행 시간 ΣEr은 감소하는 반면에 중복 관리비용(Ln+Cn)은 증가하여 전체 갱신 작업 수행 시간 ΣEw는 크게 증가한다.

읽기 연산 비율 λread가 높아짐에 따라 전체 수행 시간에서 읽기 모드 연산의 수행 시간이 차지하는 비중이 커지고, 반대로 λread가 작아질수록 전체 작업 시간의 대부분을 갱신 연산이 차지하게 된다. Er에 의해 전체 수행 시간이 크게 영향을 받는 상태인 λread가 큰 상태에서는 복제본의 수 n을 될 수 있는 대로 작은 값으로 유지하는 것이 성능 향상에 바람직할 것이다. 현 시스템의 λread의 값을 구하여 해당 λread에 알맞은 복제본의 수 n을 유지한다면 항상 최상의 성능을 유지할 수 있다.

```
get the read ratio λread
if (λread − λprevious read) > threshold        ····①
   increase number of replica
   direction ← increased
else if (λprevious read − λread) > threshold   ····②
   decrease number of replica
   if number of replica < 1 then number of replica← 1
      direction ← decreased
endif
get the response time tresponse of object
if tresponse > tprevious response                ····③
   if direction = increased then
         decrease number of replica
   else if direction = decreased then
                  increased number of replica
direction = null
```

그림 2. 제안된 ADO 알고리즘
Fig. 2 Proposed ADO algorithm

최적의 복제본의 수 n을 구하기 위해 필요한 Ln, Cn은 일관성 유지비용을 위해 각 노드의 상태나 사용되는 중복 기법 등에 크게 영향을 받는 값이다. 따라서 Ln, Cn의 일관성 유지비용을 그대로 사용하지는 않으나 이와 밀접한 관계를 가지는 λread를 이용하여 간접적으로 복제본의 수 n을 결정하는 제안된 ADO 알고리즘은 [그림 2]와 같다. 이 방식을 통하여 정확한 Ln, Cn을 구하는 작업이 없어도 최적의 n값에 근사한 n의 값을 적응적으로 결정할 수 있다.

이 기법을 적용하기 위해서 필요한 시스템 정보는 각 객체의 읽기 연산 비율과 응답 시간이다. 이 값들은 시스템 내에 객체가 하나만 존재하는 경우는 구하기 쉬우나 여러 개가 분산되어 저장된 경우에는 각 노드에 존재하는 객체들에 대한 읽기 연산 비율과 응답 시간을 종합하여 평가할 필요가 있다. 이러한 읽기 비율에 의한 n값의 동적 결정을 위한 개념은 [그림 3]과 같다.

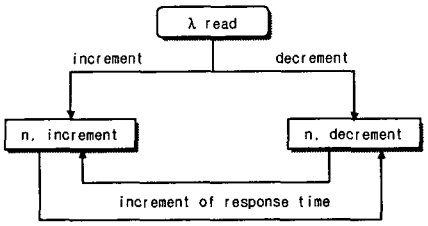

그림 3. 읽기 비율에 의한 n값의 동적 결정
Fig. 3 Dynamic decision of n-value on read ratio

이때 통신비용을 최소화하기 위하여 각 노드에 존재하는 객체는 자신의 읽기 연산 비율과 응답 시간을 알고 있으므로 이 정보를 하나의 노드로 집중시켜서 전체의 읽기 연산 비율과 응답 시간을 구할 수 있다. 이를 위해서 중재자 노드가 선택되면 각 노드의 객체들은 갱신 모드 연산이 발생하여 중재자로 호출 정보를 전달할 필요가 생기면 각각의 읽기 연산 비율과 응답 시간에 대한 정보를 중재자 노드에서는 객체 관리자가 이를 전달받아 n을 결정하는 기준으로 삼는다.

Ⅳ. 시뮬레이션 결과 및 검토

본 논문에서 제안된 알고리즘을 평가하기 위해 사용된 시스템으로는 인텔 팬티엄-Ⅳ 2.53㎓ 이며, redhat LINUX 7.2 버젼 사용 환경에서 C++ 언어를 사용하여 하였다. 시뮬레이션을 위한 객체의 사용에 있어서 동적으로 갱신 비용을 변화시키면서 지역적인 집중의 성질을 갖는 작업을 발생시키기 위해 각 노드에 객체를 자주 호출하는 상태와 객체의 사용이 적은 두 가지 상태를 가지는 간단한 마코프 모델을 [그림 4]와 같이 설정하였다.

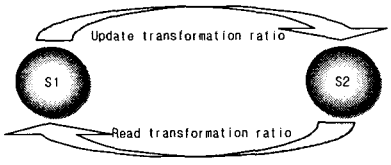

그림 4. 상태 변환을 위한 마코프 모델
Fig. 4 Markov model for state change

각각의 노드는 다수 읽기 상태(S1)와 다수 갱신 상태(S2)의 두 가지를 가지고 각각 S1 과 S2의 상태를 교대로 유지하며, 갱신 변환 비율과 읽기 변환 비율은 각 상태가 서로 바뀔 때의 변환 비율이다. 시뮬레이션에서는 20개의 노드가 주어진 마코프 모델에 따라 객체를 호출하는 것으로 가정하였으며, 각 노드는 동일한 성능을 가진다고 가정하였다.

시뮬레이션에 사용한 객체의 사용 빈도수 패턴은 시간의 흐름을 일정 구간으로 나누었을 때 임의의 구간 상에서 객체를 호출하는 횟수로 정의하여 호출 횟수가 높을수록 객체에 대하여 자주 호출이 발생하고, 해당 노드는 복제본을 설정해 줄 것을 중재자에게 요구한다. 시뮬레이션의 대상이 된 노드들은 상태 전이도에 따라 변화하며, 각각 자신의 노드에 객체를 소유할 수 있는 서버 노드 및 객체를 호출하는 작업이 발생하는 클라이언트 노드도 될 수 있다. 객체의 상태 전이도는 [그림 5]와 같다.

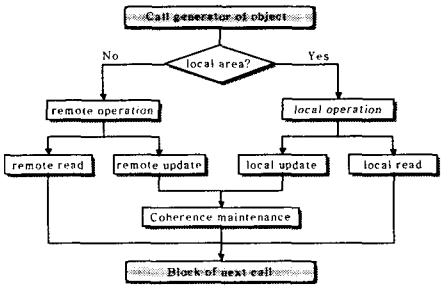

그림 5. 객체의 상태 전이도
Fig. 5 State change of object

각 노드는 일반적인 분산 객체 시스템 구성도가 객체 관리 모듈을 포함하고 있어 객체에 대한 호출이 발생할 때마다 데이터의 갱신 작업도 동시에 발생한다. 갱신되는 객체의 관리 데이터는 해당 타임 슬롯 내에 발생한 각 노드의 객체 읽기 연산 호출과 갱신 연산 호출의 횟수이다. 이를 근거로 각 노드는 적응적 중복을 수행하게 된다. 본 논문에서 사용한 공유 메모리 상의 시뮬레이션 모델은 [그림 6]과 같다.

여기서 중재자 노드는 객체의 호출 정보와 적절한 복제본의 수를 유지하고 각각의 노드들은 자신들의 객체 사용 정보를 유지한다. 작업의 수행 노드에서는 노드 내부의 객체 호출 데이터를 근거로 일정 기간동안 임계치 이상의 호출을 했는가를 판단하여 만일 그렇다면 해당 노드에서 객체 호출이 자주 발생한 것으로 한다.

호출이 발생하는 노드의 부하가 미리 정한 임계치 이

하인가를 검토하여 작업이 자주 발생하고 부하가 임계치 이하인 경우라면 해당 노드는 객체의 복제본을 유지하는 것이 효율적이다.

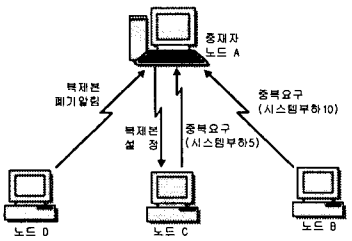

그림 6. 시뮬레이션 모델
Fig. 6 Simulation model

해당 노드는 중재자의 노드에게 복제본 설정 요구 메시지와 자신의 부하에 대한 정보를 전송하며 중재자 노드는 현 상태에서 적절하다고 판단되는 수 이상의 복제본 유지는 허용하지 않는다. [그림 6]에서 노드 D에 객체의 복제본이 존재하는 한 노드 B와 C의 복제본 설정 요구는 모두 무시된다.

노드 D는 자신이 가지고 있는 객체에 대한 호출을 관리하여 일정 시간동안 임계치 이하로 호출 횟수가 떨어지면, 복제본을 가지고 있을 조건에 위배되는 것으로 간주한다. 중재자가 아닌 노드 D에 존재하는 객체는 그 복제본 설정 사항이 해당 노드 D 자신에 의해 취소되고 이 사실을 중재자 노드에게 알리며, 기억 장소로부터 객체를 삭제한다.

작업이 반복되면서 시스템 내에는 복제본의 위치와 복제본 수가 항상 변하며, 해당 상태에게 적절하다고 판단되는 수의 객체 복제본이 존재 및 위치가 결정되는 것이다. 만약 이 때 적절한 복제본의 수가 1이라고 판단되면 중복 기법이 아닌 이주 기법이 적용되어야 하며, 기존의 중복 기법을 그대로 적용하면서 중재자 노드가 이동하는 형태를 갖게 한다.

객체 복제 관리 시스템에서 적정한 복제본의 수를 결정하는데는 영향을 주는 요인을 정형적으로 측정되기보다는 시스템 상태에 따라 변화하는 값들이기 때문에 복제본의 수가 분산 시스템의 성능에 어떠한 영향을 미치는가를 확인하기 위해서 복제본 수를 1~6개까지 변화시켜가면서 전체 작업의 수행시간이 어떻게 변화하는가를 살펴보았다. 각 노드는 1,000번의 객체 호출 작업을 수행하는데 이 때 각 호출 작업의 특성은 마코프 모델에 따르고, 각 노드에 발생한 객체의 사용 빈도수 패턴에 의한 적정한 복제본 수를 결정하기 위한 시뮬레이션 결과는 [그림 7]과 같다. 읽기 연산의 비율이 높을 때는 복제본의 수가 많을수록 좋은 성능을 보이다가 읽기 연산 비율이 낮을 때에는 급격한 성능의 저하를 나타낸다.

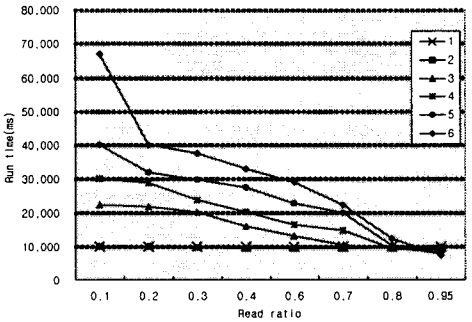

그림 7. 복제본 수에 의한 수행 시간
Fig. 7 Run time of replication number

예를 들면 복제본의 수가 2개일 때를 살펴보면, 읽기 비율이 높아짐에 따라 성능의 향상을 보이다가 읽기 비율이 60%대를 지나면 성능의 급격한 저하를 보이고 있다. 반면 복제본의 수가 1인 경우 즉, 중복 기법을 적용하지 않은 경우 읽기 연산의 비중이 변하여도 큰 차이를 나타내지 않고 있다. 이주 작업에 있어서도 읽기 작업보다는 갱신 작업이 많은 비용을 소모하는 것으로 나타났는데 이는 이주 작업에 의한 비용 증가보다는 갱신 작업 자체에 의한 비용 증가로 해석된다. 또한 이주 기법과 중복 기법을 비교하기 위하여 동일한 조건에서 각각의 수행 시간의 결과는 [그림 8]과 같다.

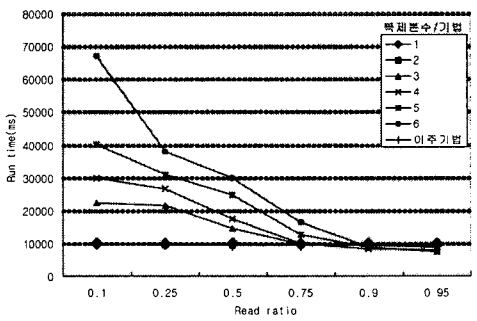

그림 8. 중복과 이주 기법 적용한 수행 시간
Fig. 8 Run time using duplication and migration mechanism

이와 같은 결과에서 객체의 수행 성능이 몇 구간으로 나누어 다르게 나타나는데, 읽기 연산 비율 λread가 25%~75%일 때는 일관성 유지비용이 크지만, 병렬 수행에 따른 성능의 향상 효과가 커서 2개의 복제본을 유지하여 중복 수행하는 것이 효율이 좋았으며, λread가 75%~90%, 90%~95%인 구간에서는 각각 3~4개의 복제본을 유지할 때 가장 좋은 성능을 보여 일관성 유지에 필요한 비용보다는 병렬 수행 효과가 훨씬 크다는 것을 알 수 있다

중복 기법과 이주 기법을 적용한 시뮬레이션 결과를 통하여 얻은 데이터를 토대로 읽기 연산 비율 λread가 실시간으로 계속해서 변화하는 상태에서 각 중복 기법 따른 전체 수행 시간의 결과는 [그림 9]와 같다.

각 기법들에 대하여 작업 수행의 결과를 비교 평가해 보면, 복제본의 수가 점차적으로 증가함에 따라 전체적으로 성능이 약간씩 개선되는 것을 알 수 있으며, 복제본의 수가 4개 이상이 되면, 전체 수행 시간이 급격하게 증가하여 성능 저하를 초래하였으며, 이는 일관성 유지비용이 복제본의 수에 기하급수적으로 비례하기 때문이다.

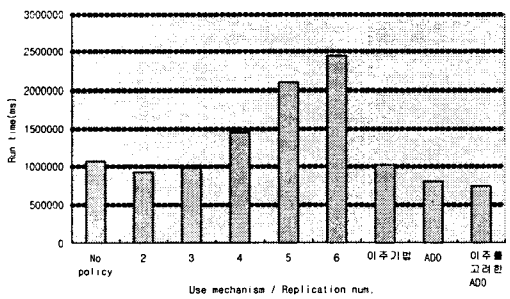

그림 9. 각 중복 기법의 수행 시간
Fig. 9 Run time of each duplication mechanism

[그림 9]의 각 중복의 수행 시간에서는 복제본의 수가 3개일 때보다는 2개일 때가 오히려 성능이 더 좋게 나타난 사실이 특징적이다. ADO 알고리즘 적용은 시스템 환경에 따라 각각의 상황에서 가장 좋은 복제본의 수를 선택하게 되어 만족할만한 성능의 개선을 보여주고 있으며, ADO 기법을 적용한 상태에서 이주 기법을 고려하여 적용한 결과 적응적 중복을 실시해서 얻은 성능 향상 비율이 개선됨을 알 수 있다.

V. 결 론

본 논문에서는 ADO 알고리즘을 통하여 분산 객체 시스템을 구성하였을 때, 기존의 전통적 중복 기법에서 사용하는 일관성 유지비용 외에 별도의 부가 비용이 없이도 제한적으로 병렬 수행 효과가 있었다.

중복 기법에 있어서 가장 큰 오버헤드로 알려진 일관성 유지비용을 최소화시키기 위하여 이 비용을 결정하는 가장 핵심적인 요소인 객체의 복제본의 수를 동적으로 변화시켜 관리함으로써 전체 수행 시간의 개선이 있었고, 이주 작업은 이미 중복 관리를 위하여 사용하고 있는 시스템 데이터를 이용하기 때문에 별도의 부가 메세지를 사용하지 않고서도 성능 향상을 보았다. 본 논문에서는 모델을 단순화하기 위하여 한 종류의 객체만이 존재하는 상태를 가정하였으나, 일반적으로 서로 연관된 객체들은 시간적 연관성을 가지고 있기 때문에 이들을 집단화하면 전체 수행 시간 중 대기 시간을 줄일 수 있었다. 그리고 향후 일정 기간동안 적정 복제본의 수가 변하지 않는 상태에서도 복제본의 위치를 변화시키거나 다른 중재자를 선택하는 등의 변화를 통한 성능의 개선 연구가 필요하다.

참고문헌

[1] Kai Hwang, Advanced Computer Architecture : Parallelism, Scalability, Programmability, McGrawHill, 1993.

[2] Daniel E. Lenoski, Wolf-Dietrich Wber Scalable Shared-Memory Multiprocessing, Morgan Kaufmann Publishers, 1995.

[3] J. Protic, M. Tomasevic & V. Milutinovic, "Distributed shared memory : concepts and systems", IEEE parallel & distributed

technology, vol. 4, no. 2, pp.63-79, Summer, 1996.

[4] Y. Amir, "Replication over a Partitioned Network", The Hebrew Univ. Jerusalem, Feb. 1995.]

[5] J. Achibald & J. L. Baer, "Cache Coherence Protocol : Evaluation Using a Multiprocessor Simulation Model", ACM Trans. on Comp. System, vol.4, no.4, pp273-298, Nov. 1996.

[6] G. D. Parring and S. K. Shrivastava, "Implementing Concurrency Control in Distributed Object-Oriented Programming, ECOOP88, Oslo, Auguest, 1998.

[7] M. C. Little and D. L. McCUE, "The Replica Management System : a Scheme for Flexible and Dynamic Replication," Proc. of the 2nd Workshop on Configurable Distributed Systems. Pittsburgh, March, 1994.

[8] T. Loukopoulos and I. Ahmad, "Static and Adaptive Replication Algorithms for Fast Information Access in Large Distributed Systems," IEEEP roceedings of the 20th International Conference on Distributed Computing Systems , pp. 385-392, 2000.

[9] H. Stockinger, A. Samar, B. Allcock, I. Foster, K. Holtman, B. Tierney, "File and Object Replication in Data Grids," IEEE Proceedings of the Tenth IEEE International Symposium on High Performance Distributed Computing, pp. 76-86, 2001.

저 자 소 개

박 종 선

목포과학대학 컴퓨터정보과 조교수

오 수 열

목포과학대학 컴퓨터정보과 조교수

장 용 철

1988년 조선대학교 컴퓨터공학과(공학사)

1990년 조선대학교 대학원 컴퓨터공학 과(공학석사)

1997년 조선대학교 대학원 컴퓨터공학과(공학박사)

1990년~ 현재 목포과학대학 컴퓨터정보과 부교수

관심분야 : 멀티미디어 시스템, 분산 객체 시스템