광역 객체 컴퓨팅에서 통합(이름/속성) 기반의 동적 바인딩 서비스 모델의 실험분석

(The Experimental Analysis of Integrated(Name/Property) Dynamic Binding Service Model for Wide-Area Objects Computing)

정 창 원 † 주 수 종 ††
(Chang Won Jeong) (Su Chong Joo)

요 약 광역 환경에서 존재하는 수많은 서버객체들은 이름이나 속성에 의해 다양한 중복된 성질을 갖는다. 이러한 같은 성질을 갖는 서버객체들인 중복객체들에게 서비스를 요청할 때, 기존의 네이밍이나 트레이딩 서비스는 중복된 서버객체들의 바인딩 서비스가 불가능하다. 따라서 우리는 광역 컴퓨팅 환경에서 중복객체의 선정 및 동적 바인딩 서비스를 위한 통합모델을 제시하였다. 본 모델은 중복된 객체들의 위치 관리 기능뿐만 아니라 시스템들간의 부하 균형화를 유지하기 위해서 최소부하를 갖는 시스템상의 객체를 선정하는 동적 바인딩 서비스 기능을 제공한다. 이러한 목적에서 우리는 광역 컴퓨팅 환경에서 중복특성을 가진 서버객체들의 바인딩을 지원하기 위한 서비스 방안과 모델을 구축해 왔다. 본 논문에서는 구축된 모델에 대해 실험환경을 보이고, 연합 모델에서 클라이언트와 서버와의 바인딩 과정을 성능 평가하였고, 부하균형이 우리의 모델에 적용될 수 있는지 확인하기 위하여 주어진 조건을 이용하여 우리의 모델을 검증하였다. 또한 우리는 광역환경을 위한 도메인들간의 연합을 고려한 모델의 수행결과도 분석하였다. 이들 수행 결과를 통해 기존 네트워크의 물리적인 트리 구조상에서 검색 비용이 적음을 보였다.

키워드 : 동적 바인딩 서비스, 통합 모델, 연합 모델, 중복 객체 관리

Abstract Many objects existing on wide area environments have the replication characteristics according to how to categorize using their own names or properties. From the clients' requests, the existing naming and trading services have not supported with the binding service for replicated server object with the same service type. For this reason, we present an integrated model that can support the selection of replicated object and dynamic binding services on wide-area computing environments. This model suggests provides not only location management of replicated objects but also active binding service which enables to select a least-loaded object on the system to keep the balance of load between systems. In this purpose, constructing both the service plan and model for support server object's binding with replication property on wide area computing environments has been researched. In this paper, we showed the test environment and analyzed the performance evaluation of client/server binding procedures via integrated binding service in federation model and verified our model under the condition to see whether load balance can be applied to our model. For the performance evaluation of suggested wide area integrated binding service federation model, evaluated the integrated binding service of each domain and analyzed the performance evaluation of process for non-replication object's under federation model environment. Also, we analyzed the performance evaluation of the federation model between domains for wide area environment. From the execution results, we showed the federation model provides lowers search-cost on the physical tree structure of network.

Key words : dynamic binding service, Integrated Model, Federation Model, Replicated Object Management

· 본 연구는 한국 과학기술부의 특정기초 연구 사업의 지원으로 수행되었음 (R01-2006-000-10147-0)

† 정 회 원 : 전북대학교 차세대 LBS응용연구센터 학술연구교수
mediblue@chonbuk.ac.kr
†† 종신회원 : 원광대학교 전기전자및정보공학부 교수
scjoo@wonkwang.ac.kr

논문접수 : 2004년 11월 15일
심사완료 : 2006년 5월 25일

1. 서 론

광역환경에서 분산시스템의 연구는 객체지향기술, 인터넷기술 및 중앙제어방식에서 분산제어방식의 컴퓨팅 환경으로 급변하고 있다. 또한 그 서비스 영역도 광역화되고 플랫폼들은 이질적이며, 서비스 범위 또한 기존의 범위를 확장시키기 위해 연합하는 형태로 바뀌고 있다[1,2,3]. 분산 서비스 영역이 넓혀지고, 광역 객체 컴퓨팅 환경이 되면서 지구상에 존재하는 서버객체들이 모든 서비스 대상이 될 수 있다. 그러나 이러한 서버객체들을 편리하게 찾고 요청하기 위해서 서버객체의 이름이나 속성을 이용하여 체계적으로 분류해야 한다. 이때 이들 서버객체들로부터 중복된 특성들이 발견된다. 이와 같이 같은 이름이나 특성을 갖는 서버객체들을 중복객체(replicated objects)라 말한다[4-6]. 광역환경에서 클라이언트 객체로부터 원격의 한 서비스를 제공하는 서버객체를 요청할 때, 해당 서버객체가 중복객체 일 경우에 중복객체 중에서 적절한 하나의 객체를 선정하는 데 부하균형화와 같은 지식적인 전략이 추가되어야 한다[7]. 이러한 관점에서 본다면, 기존의 네이밍이나 트레이딩 메커니즘은 독립적인 위치 종속적인 바인딩만을 지원하고 있다. 그러나 광역환경에서는 중복객체의 최적 선정 전략과 위치 투명성의 제공 및 동적 바인딩 서비스가 요구된다[8,9]. 이러한 이유로 본 논문에서는 광역 컴퓨팅 환경에서 중복된 서버객체들의 위치 관리뿐만 아니라 시스템들간의 부하 균형화를 유지하기 위해서 최소 부하를 갖는 시스템에 위치한 서버객체를 선정하여 동적 바인딩 서비스를 제공할 수 있는 새로운 모델로서, 통합(이름/속성)기반의 동적 바인딩 서비스 모델(IDBSM: Integrated(Name/Property) Dynamic Binding Service Model)을 제안하고 구축하였다. 본 모델에 대한 서비스 방안과 모델의 구축에 대해는 논문[10-13]에 자세하게 기술되어 있다.

본 논문에서는 IDBSM의 전반구조, 구현내용 및 실험환경을 보이고 실험분석에 중점을 두었다. 이를 위해 구축된 모델로 이용하여 실험환경을 보이고 주어진 조건과 바인딩 시나리오를 통해 네이밍이나 트레이딩 서비스를 통해 서버객체의 레퍼런스를 얻고, 광역환경에서 클라이언트 객체와의 바인딩 과정을 단계별로 실험하고 분석하였다. 또한 우리는 광역환경을 위한 도메인들간의 연합을 고려한 모델의 수행결과도 분석하였다.

2. 관련 연구

2.1 기존의 분산 객체 모델

분산 시스템의 가장 중요한 목표는 분산 위치 투명성을 제공하는데 있다. 즉, 인터넷으로 연결된 시스템들간에 분산 서비스를 위해 투명성을 제공함으로 사용자들에게 단일 이미지 시스템 환경을 지원하도록 하는데 있다. 이를 위해 기본이 되는 분야가 분산 객체 모델에 관한 연구이다. 네트워크를 기반으로 한 분산 처리와 객체지향 기술을 이용하여 분산 객체 환경을 제공할 수 있는 관련 표준안들이 연구되고 있다. 대표적인 연구로는 OMG의 CORBA, Microsoft의 DCOM, INRIA의 SOS (The SOMIW object-oriented Operating System)그룹에서 제안한 단편화 객체(Fragmented Object) 그리고 Globe의 분산 공유 객체(Distributed Shared Object)와 썬 마이크로시스템스 프로젝트인 스프링 객체(Spring Object)가 있다.

대부분의 분산 객체 모델에서 바인딩 서비스로 네이밍 서비스를 모두 제공하고 있다. 그러나 속성에 의한 검색 메커니즘인 트레이딩 서비스는 CORBA를 제외한 나머지 객체 모델에서는 제공하지 않고 있다. 또한, 객체의 위치 서비스는 Globe[14,15]를 제외한 나머지 객체 모델에서 제공하지 않고 있다. 본 논문에서는 이러한 제한 사항을 극복하고자 한다.

2.2 연합 모델

연합은 적어도 두 개 이상의 네트워크 상에 구성된 분산 시스템 환경들이 서로 상호운용을 위해 투명성을 지원하는 상태를 말한다. 이러한 요구사항을 만족하는 연합 모델은 상호운용성을 지원하기 위한 미들웨어를 포함한 독립적인 시스템들로 구성된다[16-19]. 즉, 각 영역별 자치적인 관리를 갖는 시스템들이 상호운용을 통하여 클라이언트에게 필요한 서비스를 제공하는 집합체로 각각의 서비스를 서로 이용 가능하도록 한다. 상호운용은 시스템의 능력을 공유하기 위해 교섭하는 메커니즘과 시스템 실행 시간에 상호 운용 관계를 확립하는 메커니즘을 포함한다. 이러한 메커니즘은 상호 작용하고 있는 소프트웨어 컴포넌트들이 상호운용을 하기 위해 필수적이다. 다양한 표준들이 제시하는 연합 메커니즘에 대한 비교는 표 2와 같다.

대부분의 연합 모델은 연합할 객체간의 협약 프로세스는 엄격한 규약을 갖는다. 따라서, TINA, ANSA 그리고 CORBA 모델은 각각 전형적인 분산 시스템 특징의 요구사항을 지원하기 때문에 연합 시스템의 요구사항을 완벽하게 만족하지 못하고 있는 실정이다[20]. 이들 플랫폼 구조는 시스템에 의존적이며, 서비스 인터페이스가 단일화되어 있다. 또한 객체간의 관계에 대한 상호작용은 공학 인터페이스 레퍼런스(engineering interface reference)를 기반으로 확립된다. 하지만 연산(computational) 인터페이스간의 서비스 관계에 대한 기능을 제공하지 않고 있다. 연합 시스템에서는 연산 레벨상의 관계를 추가적으로 공학 인터페이스와 분류한다.

표 1 분산 객체 모델의 특성 비교

Issue	CORBA	DCOM	Fragmented Objec	Globe	Spring Model
Design goals	Interoperability	Functionality	Distributed object support OS	Scalability	Extensibility
Object Model	Remote objects	Remote objects	Fragmented objects	Distributed objects	Spring objects
Service	Many of its Own	From environment	Few	Few	Few
Interface	IDL based	Binary	Binary	Binary	IDL based
Sync. communication	Yes	Yes	Yes	Yes	Yes
Async. communication	Yes	Yes	Yes	No	Yes
Callbacks	Yes	Yes	No	No	No
Events	Yes	Yes	No	No	No
Messaging	Yes	Yes	Yes	No	Yes
Object Server	Flexible(POA)	Hard-coded	Hard-coded	Object dependent	Hard-coded
Directory Service	Yes	Yes	No	No	Yes
Trading Service	Yes	No	No	No	No
Naming Service	Yes	Yes	Yes	Yes	Yes
Location Service	No	No	No	Yes	No
Object reference	Object's Location	Interface pointer	Object's location	True identifier	Interface pointer
Replication support	Separate Server	None	None	Separate subobject	By subcontract
Transactions	Yes	Yes	No	No	No
Fault tolerance	By replication	By transactions	No	By replication	No
Recovery support	Yes	By transactions	No	No	No
Security	Various mechanisms	Various mechanisms	No	More work needed	Various mechanisms

표 2 연합 메커니즘 비교

Feature	Federated System	Distributed System	ODP	TINA	ANSA	CORBA
Federation Establishment	dynamic/ static	static	not decided yet	static	static	static
Separation of liaison from channel	YES	NO	YES	NO	NO	NO
Platform Interface	unified/ personalized	unified	unified	unified	unified	unified
Liaison establishment at Computational (C) or engineering (E) level	C, E	E	E, C not yet decided	E	E	E

이는 공학적인 레벨의 바인딩은 연산 관계에 의해 운영되기 때문이다. TINA DPE의 경우는 이질적인 정보통신 네트워크에서 동일한 미들웨어의 집합으로 제공하기 때문에 연합 시스템 대신에 전형적인 분산 시스템으로 본다. 그리고 ANSA와 CORBA 또한 전형적인 분산 시스템에 기반을 두고 있지만, 특히 ANSA에서는 연합과 자치성의 개념을 강조한다. 그러나 이들 모두 정적이거나 고정된 연합방법을 채택하고, 바인딩 모델은 대부분 근거리 통신망에서 유효한 메커니즘으로 제공하며, 이름으로 식별하는 이름 대 주소의 쌍으로 이루어진 매핑 기법을 이용한다. 또한 기존 벤더들이 제품으로 출시한 ORB에서 생성하는 객체 레퍼런스의 구조가 모두 상이

하기 때문에 이에 대한 일관성 있는 작업의 어려움 때문에 표준화되지 않았다.

따라서 본 논문에서는 바인딩 서비스를 제공하기 위해 네이밍 서비스와 트레이딩 서비스를 제공하는 서버들을 개별적으로 두고 이용하는 방법을 통합한다. 이러한 통합된 바인딩 서비스들간의 연합은 연합 시스템과 같이 동적인 연합 메커니즘과 하부구조와 인터페이스는 CORBA의 공통적인 IDL을 이용한다.

3. 통합(이름/속성) 기반의 동적 바인딩 서비스 모델(IDBSM)

본 장에서는 구축된 IDBSM에 대한 관련연구와 이론적 배경 및 구축을 위한 자세한 설계 및 구현내용은 우리가 발표한 논문[11-13]에서 기술되었으므로 재 고찰하는 의미에서 전반구조와 구현내용을 기술하고, IDBSM의 실험환경과 실험 및 결과 분석에 초점을 두었다.

3.1 IDBSM의 전반 구조

본 IDBSM의 구조는 4단계로 이루어진다. 첫 번째 과정은 네이밍/트레이딩 서비스에 의해 서버객체들의 유일한 식별자인 객체 핸들을 얻어오는 과정으로 1 단계이며, 두 번째 과정은 위치 서비스에서 네이밍/트레이딩 서비스로부터 얻어진 객체 핸들에 대한 컨택 주소를 얻는 과정으로 2와 3 단계를 나타낸다. 세 번째 과정은 요청한 클라이언트와 위치 서비스로부터 얻은 서버객체의 컨택주소를 이용하여 바인딩하는 과정은 마지막 4단계에서 이루어진다. 그림 1은 통합 바인딩 서비스의 전반구조 및 수행과정 보인다.

3.2 IDBSM들간의 연합(Federation)

본 논문에서 제안하는 연합구조는 기존 DNS의 네트워크 트리 구조와 유사하나 영역 별 통합 바인딩 서비스와 Cooperator가 한 도메인 내에 위치하며, 계층적으로 Cooperator들간의 정보 교환이 이루어지도록 하였다. 그림 2는 IDBSM간의 Cooperator를 이용한 연합구조를 보이고 있다. 본 모델의 연합과정을 지원하기 위해 사용된 Cooperator의 기능과 역할에 대한 세부내용은 [20] 논문을 참조한다. 광역환경은 통합 트리 기반으로 도메인 A, B, C, D로 구분하였다. 그리고 각 도메인은 IDBSM이 서버객체들을 관리하고 있는 논리적인 도메인과 같다. 각 도메인 상에 Cooperator들은 각각의 영역에 통합 바인딩 서비스와 같이 위치하여, 각 도메인 내의 IDBSM들간의 연동을 위한 상호접속을 지원한다. 연동과정을 살펴보면, 임의의 영역 A의 클라이언트 객체가 서버객체를 탐색하기 위해 Cooperator에게 임포트(NaT_Lookup)요청을 한다. 해당 도메인의 광역 통합 바인딩 서비스의 구성요소인 통합 바인딩 서비스를 통해 임포트 명세와 일치하는 서버객체를 탐색 범위를 넓혀 관련 정보를 교환함으로써 클라이언트 객체에게 투명하게 임포트 요청으로부터 가장 적합한 서버객체에

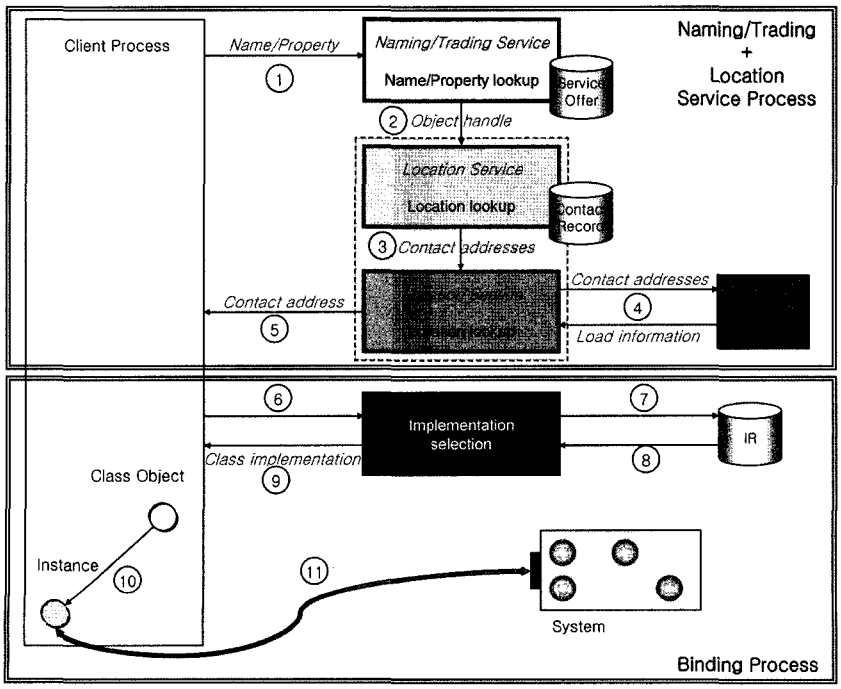


그림 1 통합 바인딩 서비스의 전반구조 및 수행과정

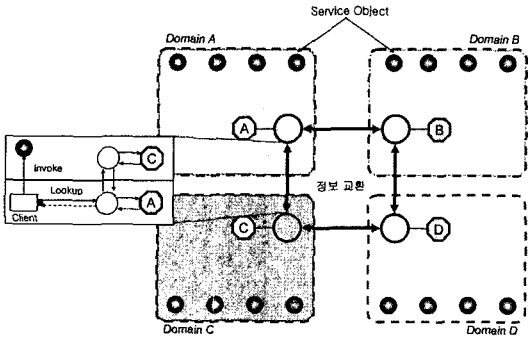


그림 2 Cooperator를 이용한 IDBSM의 연합구조

대한 컨택 주소를 반환한다. 우리가 제안하는 연합방법은 이질적인 환경의 연합 서비스를 제공하기 위해 Cooperator를 통하여 통합 바인딩 서비스가 관리하는 영역내의 서버객체에 대한 정보를 얻을 수 있도록 한다.

기존의 연합방법으로 네이밍 서버들간의 연합 그리고 트레이더간의 연합에 대한 연구가 있으나 대부분이 네이밍 프로토콜 또는 트레이더 프로토콜을 기반으로 연합하는 형태이다. 또한, 고정된 단방향 트리 구조를 구성한다. 이러한 계층적인 트리 구조에서 동일 레벨 상의 노드임에도 불구하고, 근 노드를 경유하여 포워딩 포인터를 따라 검색하는 문제점을 갖는다. 이러한 검색에 대표적인 방법인 반복(iterative) 검색방법이나 재귀(recursive) 검색방법에 비해 검색 비용을 줄일 수 있다. 광역 통합 바인딩 서비스의 연합 모델은 최소한 동일 레벨의 모든 노드들은 Cooperator에 의해 연결됨으로 검색시간을 줄일 수 있다. 따라서 제안한 방법을 이용하여 광역환경에서 서비스 선택의 폭을 넓힐 수 있으며, 시스템의 확장과 가용성을 더욱 높일 수 있다.

4. IDBSM의 구현

앞서 제시한 통합 바인딩 서비스를 구성하는 Naming/Trading 서비스와 위치 서비스 그리고 이들간의 통합 바인딩 서비스의 연합을 지원하는 Cooperator에 대한 기능을 정립하고, 해당 서버객체의 세부 속성 및 메소드 설계를 통해 서버객체의 모듈을 구현하였다.

4.1 IDBSM의 구성요소 구현

광역환경에서 연합을 지원할 수 있는 IDBSM의 전체적인 구성 요소들은 Cooperator, NaT Server, Location Server 그리고 각 시스템에 위치하여 부하정보를 제공하는 LIM(Load Information Manager)으로 구성된다. 여기에서 LIM은 LSF Tool에 의해 제공되므로 정의된 인터페이스를 통해 지체기능들을 그대로 사용하였다. IDBSM에서 각 구성요소에 대한 서비스의 오퍼레이션들과 그들간 서비스를 위한 상호관계성을 기술한 클래스 다이어그램은 그림 3과 같다.

4.1.1 Naming/Trading 서비스

Naming/Trading 서비스가 제공하는 인터페이스들은 객체 핸들을 검색하기 위한 NaT_ Lookup 오퍼레이션과 서비스 오퍼의 등록, 수정, 삭제를 위한 Register(), Modify(), Withdraw() 오퍼레이션들로 구현하였다. 또한 다른 Naming/Trading 서비스와 연결하기 위한 컨택 주소를 관리하기 위한 Link() 오퍼레이션을 갖는다. 먼저 서비스 오퍼에 서버객체를 등록하기 위해서 OH-Generator() 오퍼레이션을 통해 객체핸들을 생성시킨다.

```
// 네이밍/트레이딩 서비스 오퍼레이션 인터페이스
    interface NaT_Operation
{ // 객체 핸들 검색
```

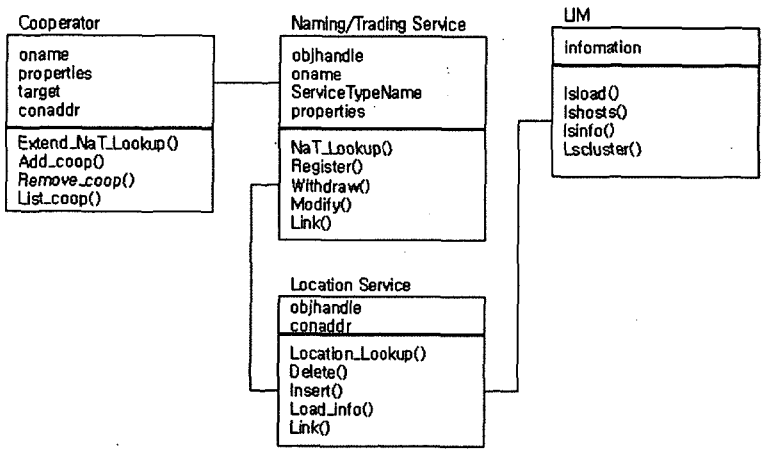


그림 3 IDBSM의 전체 클래스 다이어그램 및 관계성

```
    ObjectHandle NaT_Lookup(in ObjectName oname, in
    PropertyValueSeq valueSeq);
// 객체 등록
    ObjectHandle Register(in ObjectName oname, in
    ServiceTypeName type, in PropertySeq properties);
// 객체 수정
    void Modify(in ObjectHandle objhandle, in
    ObjectName oname, in PropertySeq properties);
// 객체 철회
    void Withdraw(in ObjectHandle objhandle);
// 링크
    void Link(in string subpoint); };
```

4.1.2 위치 서비스

위치 서비스에서 제공하는 인터페이스는 Naming/Trading 서비스로부터 얻어진 객체 핸들과 서버객체의 주소를 매핑하여 검색하는 Location_Lookup() 오퍼레이션과 하나 이상의 컨택 주소를 삽입하고, 삭제하기 위한 Insert(), Delete() 오퍼레이션들로 구현하였다. 또한 각 위치에 대한 LSF의 LIM으로부터 시스템 부하 정보를 요청하고 받기 위한 Load_Info() 오퍼레이션을 사용한다.

```
// 위치 서비스 오퍼레이션 인터페이스
    interface Location_Operation
{ // 컨택 레코드 검색
    conaddrSeq Location_Lookup(in ObjectHandle
    objhandle);
// 컨택 레코드 등록
    void Insert(in ObjectHandle objhandle, in
    ContactAddr conaddr);
// 컨택 레코드 삭제
    void Delete(in ObjectHandle objhandle, in
    ContactAddr conaddr);
// 부하 정보 및 최적 서버 탐색
    ContactAddr Load_Info(in conaddrSeq conaddrs); };
```

4.1.3 연동서비스

Cooperator가 제공하는 인터페이스는 클라이언트 요청에 따라 해당 영역에 위치한 Naming/Trading 서비스에게 이름 또는 속성으로 검색을 위해 도메인간의 연동을 지원한다. 이를 위해 extend_NaT_Lookup() 오퍼레이션과 다른 도메인에 위치한 Cooperator의 연합을 위한 add_coop(), 그리고 삭제하기 위한 remove_coop() 오퍼레이션을 갖으며, 연합된 Cooperator의 리스트를 보기 위한 list_coop() 오퍼레이션 그리고 연합을 통해 얻어진 컨택 주소와 부하정보에 대한 비교하여 선정하는 ds_selection() 오퍼레이션을 갖는다.

```
interface Cooperation {
 // 연합된 Cooperator Lookup
string extend_NaT_Lookup(in string oname, in string
PropertyValue);
 // 연결시킬 Cooperation 등록
string add_coop(in string target, in string conaddr);
 // 연결시킨 Cooperation 삭제
string remove_coop(in string target, in string conaddr);
 // 연결된 Cooperation 리스트
string list_coop(in string target);
 // 검색 결과
string ds_selection(in conaddr, in string
load_information); };
```

4.2 IDBSM의 구현환경

IDBMS의 구현환경은 Sun UltraSparc 2i로 333Mhz의 CPU속도에 운영체제로 Sun Solaris 2.7인 시스템 1대와 X86-based PC 2대로 CPU속도는 각각 500Mhz, 400Mhz이며, 운영체제는 Windows 2000 서버로 구성하였다. 광역 객체 컴퓨팅의 미들웨어로는 CORBA 사양을 따르는 상용화된 미들웨어로 Borland의 VisiBroker 4.1을 사용하였으며, 통합 바인딩 서비스의 구성요소들인 Naming/Trading 서비스와 위치 서비스는 자바(Java2 SDK 1.3.1)로 개발하였다. 서비스 오퍼와 컨택 레코드 정보를 관리하기 위해 Microsoft의 관계형 데이타베이스 시스템인(RDBMS)인 SQL Server 2000을 이용하였으며, 각 시스템의 부하정보를 획득하기 위해 LSF 4.1 툴을 사용하였다. IDBMS의 구성요소들인 Naming/Trading 서비스와 위치 서비스는 하나의 특정 시스템에 위치시키고, Naming/Trading 서비스가 분산 객체를 관리하는 영역으로 하나의 도메인을 구성하였다. 그리고 분산 객체는 도메인 내에 각 시스템에 배치하였다. 각 서비스를 지원하는 서버객체들에 대한 인터페이스는 IDL로 작성하였으며, Visiborker4.1 IDL 컴파일러로 컴파일링을 하였다.

5. IDBSM의 실험 및 분석

본 장에서는 제안한 IDBSM의 실험환경 및 조건 그리고 실험 결과에 대해 기술한다.

5.1 실험환경 및 조건

본 IDBSM의 실험환경은 다음 그림 4와 같이 서로 다른 도메인(D_1, D_2, D_3)에서 각각 통합 바인딩 서비스를 위한 서버(system A, B, C)를 하나씩 두었다. 이들은 각각 관리하는 영역상의 서버객체들에 대한 정보(서비스 오퍼, 위치 정보)를 관리한다. 실험환경으로부터 얻어진 각 도메인간의 평균 네트워크 지연시간은 D_1과 D_2간에는 15ms, D_1과 D_3간에는 35ms 그리고 D_2와 D_3간에는 40ms 측정되었으며, 이를 도메인들간에 네트워크 지연시간의 실험데이타로 활용한다. 실험식을 세우기 위해 현재 클라이언트 객체가 요청하고 있는 도메인을 도메인 D_1이라고 하고, 나머지 두 도메인을 외부 도메인

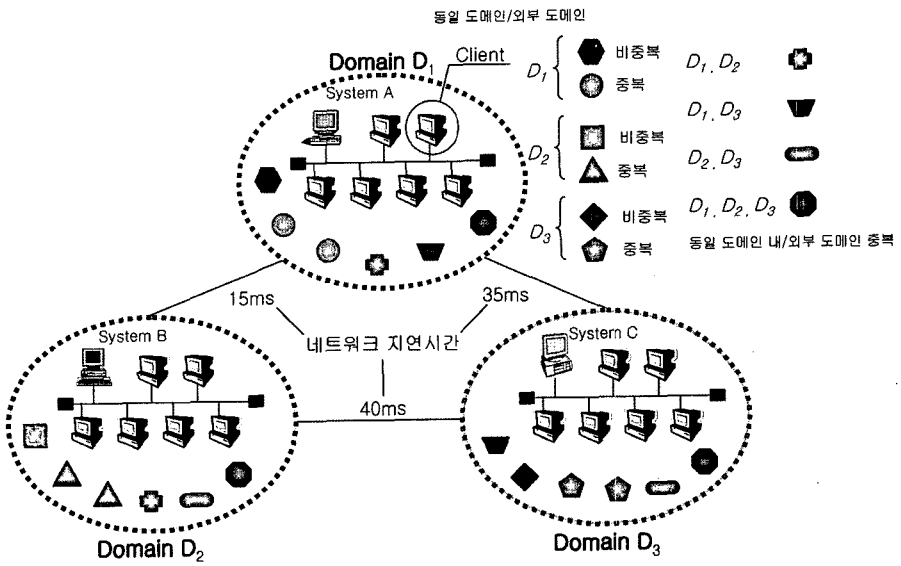


그림 4 IDBMS의 실험 환경

D_2과 D_3이라 부른다.

본 실험환경에서 비중복 객체뿐만 아니라 중복객체를 존재를 고려하기 위해 중복객체들은 동일 도메인 또는 서로 다른 도메인 내에 존재할 수 있다고 가정한다. 본 실험은 D_1내의 클라이언트가 동일 도메인 상에 위치한 통합 바인딩 서비스를 요청할 때, 해당 서버객체가 비중복 또는 중복으로 존재할 경우와 각각 다른 도메인 D_2와 D_3상에 컨택 주소가 존재할 경우를 비중복, 중복으로 구분하여 Lookup 시간 및 중복객체인 경우, 적절한 하나의 서버객체를 선정 시간에 대해 성능을 실험하였다. 그리고 클라이언트 객체와 서버측의 서버객체간의 바인딩 처리 시간을 합하여 전체 서비스 처리 시간을 분석하였다. 이를 위해 Lookup 시간 및 서버객체 선정 시간을 얻는 3가지 경우로 구분하고, 이에 대한 전체 서비스 처리 시간을 얻기 위해 주어진 조건에 대한 실험식들을 제시하고, 5.3절에서 실험결과를 보인다. 마지막으로 도메인간의 연동방법으로 기존 검색방법인 반복 검색과 재귀 검색 방법과 제안한 Cooperator를 이용한 연합방법과의 비교 분석하였다.

5.2 실험식

제안한 통합 바인딩 서비스 모델의 실험환경으로부터 클라이언트의 요청으로부터 서버 객체와 바인딩을 통해 반환결과를 얻을 때까지의 시간을 전체 서비스 처리 시간(Ttotal)은 통합 바인딩 서비스가 최소 부하를 갖는 객체의 컨택 주소를 반환하는데 까지 걸리는 시간인 Lookup 시간 및 선정 처리 시간(TS)에 클라이언트에서 객체에 바인딩하여 서비스 수행 후 결과값을 반환하는 바인딩 처리 시간(TB)의 합으로 구성된다.

먼저, Lookup 시간 및 선정 처리 시간(TS)은 그림 5)와 같이 $T_{n/a} + T_I + T_{ds}$ 의 합으로 구할 수 있다. 그림 5에서 나타난 바와 같이 Lookup 시간 및 선정 처리 시간은 클라이언트에서 통합 바인딩 서비스의 구성요소인 네이밍/트레이딩 서비스가 클라이언트가 요청한 객체의 유일한 객체 핸들을 추출하는 시간($T_{n/a}$) 그리고 위치 서비스에서 객체 핸들에 따르는 컨택 주소를 추출하는 시간(T_I) 그리고 비중복일 경우를 제외하고 여러 개

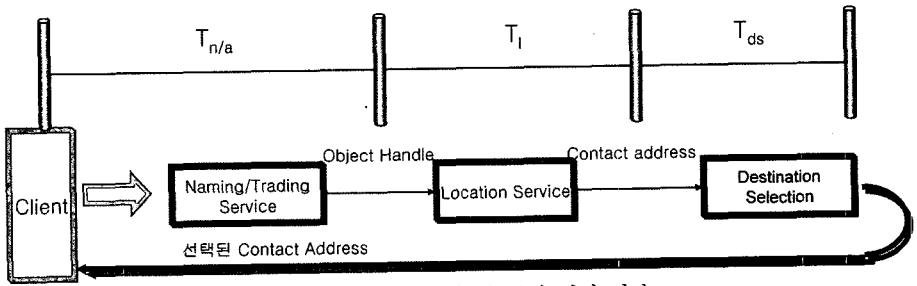


그림 5 Lookup 시간 및 선정 처리 시간

의 컨택 주소가 추출되었을 경우 이들 중 부하가 적은 객체를 선택하여 하나의 컨택 주소를 얻어내는 시간(T_{ds})의 합으로 구할 수 있다. 여기에서 중복된 객체가 비중복된 객체보다 소요 시간이 T_{ds}의 시간에 의해 더 소요된다.

클라이언트와 객체의 바인딩 처리 시간(TB)은 그림 6과 같이 클라이언트에서 객체에 요청하여 객체가 서비스를 하기 위해 대기하는 시간(T_{WT})과 서비스를 수행한 시간(T_{ST}) 그리고 요청시간과 서버로부터 결과 값을 클라이언트 측에서 받는 네트워크 지연 시간($T_{N/W}$)을 합하여 얻을 수 있다.

그림 6에서 나타난 바와 같이 바인딩 처리 시간은 동일한 서비스를 수행하는 객체에 바인딩 하도록 한다. 여기에서 서비스 수행 시간(T_{ST})은 같다. 이때, 클라이언트에서 객체에 접속하기 위한 시간이 네트워크의 지연 시간($T_{N/W}$)과 객체가 위치한 시스템의 부하에 따르는 대기시간(T_{WT})에 따라 변화되는 성능을 분석한다. 각 평균 소요 시간(T_{avg}) 계산하기 위해 다음 수식을 사용한다.

$$T_{avg} = \frac{\sum_{i=1}^{N} FinishTime(T_i) - StartTime(T_i)}{N}$$

N은 전체 서비스의 수이고, StartTime(Ti)와 FinishTime(Ti)는 각 소요 시간의 시작과 종료 시간을 각각 나타낸다.

Case 1 : 서버객체들이 동일 도메인 상에 존재하는 경우 ; 각 동일 도메인 상에서 통합 바인딩 서비스로부터 객체의 컨택 주소를 얻기 위한 Lookup 시간 및 선정 시간(TS)은 다음 수식에 의해 얻어진다.

- 비중복일 경우 $T_S = T_{n/a} + T_l + 2(T_{N/W})$------L1
- 중복인 경우 $T_S = T_{n/a} + T_l + T_{ds} + 2(T_{N/W})$--L2

Case 2 : 서버객체들이 다른 도메인 상에 존재하는 경우 ; 다른 도메인 D2 또는 D3에 클라이언트가 요청한 객체의 컨택 주소가 존재할 경우에는 Cooperator의 상호 작용하는 시간(TC)이 추가되며, Lookup 시간 및 선정 시간은 다음 수식에 의해 얻는다. 먼저 D2 또는 D3의 도메인상에 각각 클라이언트가 요청한 객체에 대한 컨택 주소가 비중복과 중복되어 있을 경우에 Look-up 시간 및 선정 시간에 대한 수식은 다음과 같다. D2에 컨택 주소가 있을 경우는 다음과 같다.

- 비중복일 경우 $TS = TC + D_1(T_{n/a}) + D_3[T_{n/a} + 2(T_{N/W})] + D_2[T_{n/a} + T_l + 2(T_{N/W})]$----------L3
- 중복일 경우 $TS = TC + D_1(T_{n/a}) + D_3[(T_{n/a}) + 2(T_{N/W})] + D_2[T_{n/a} + T_l + T_{ds} + 2(T_{N/W})]$-----L4

D_3에 컨택 주소가 있는 경우는 다음과 같다.

- 비중복일 경우 $TS = TC + D_1(T_{n/a}) + D_2[(T_{n/a}) + 2(T_{N/W})] + D_3[T_{n/a} + T_l + 2(T_{N/W})]$---------L5
- 중복일 경우 $TS = TC + D_1(T_{n/a}) + D_2[(T_{n/a}) + 2(T_{N/W})] + D_3[T_{n/a} + T_l + T_{ds} + 2(T_{N/W})]$----L6

Case 3 : 컨택 주소가 각 도메인 상에 컨택 주소가 중복되어 존재하는 경우 ; 각 도메인(D_1, D_2), (D_1, D_3), (D_2, D_3), (D_1, D_2, D_3)에 클라이언트가 요청한 객체의 컨택 주소가 존재할 경우, Lookup 시간 및 선정 시간은 다음 수식에 의해 얻는다.

(D_1, D_2)에 동일한 객체 핸들을 갖는 중복된 컨택 주소가 있을 경우는

- $TS = TC + D_1[T_{n/a} + T_l] + D_2[T_{n/a} + T_l] + T_{ds} + 2(T_{N/W2}) + D_3(T_{n/a}) + 2(T_{N/W3})$----------L7

(D_1, D_3)에 중복된 컨택 주소가 있을 경우는

- $TS = TC + D_1[T_{n/a} + T_l] + D_3[T_{n/a} + T_l] + T_{ds} + 2(T_{N/W3}) + D_2(T_{n/a}) + 2(T_{N/W2})$----------L8

(D_2, D_3)에 중복된 컨택 주소가 있을 경우

- $TS = TC + D_1(T_{n/a}) + D_2[T_{n/a} + T_l] + 2(T_{N/W2}) + D_3[T_{n/a} + T_l] + T_{ds} + 2(T_{N/W3})$----------L9

(D_1, D_2, D_3)에 중복된 컨택 주소가 있을 경우

- $TS = TC + D_1[T_{n/a} + T_l] + D_2[T_{n/a} + T_l] + D_3[T_{n/a} + T_l] + T_{ds} + 2(T_{N/W2}) + 2(T_{N/W3})$----L10

5.3 실험결과 및 분석

5.3.1 각 도메인 별 통합 바인딩 서비스 Lookup 시간 및 선정 처리 시간

각 도메인 별 위치한 광역 통합 바인딩 서비스의 Lookup 시간 및 선정 처리 시간 결과는 그림 7과 같다.

Lookup 시간 및 선정 시간이 D_2에 있는 시스템 상에 있는 통합 바인딩 서비스의 성능이 가장 좋다. 그러나 클라이언트 측면에서는 동일 도메인 상의 통합 바인딩 서비스를 사용하여 서비스를 제공받는 것이 다른 도메

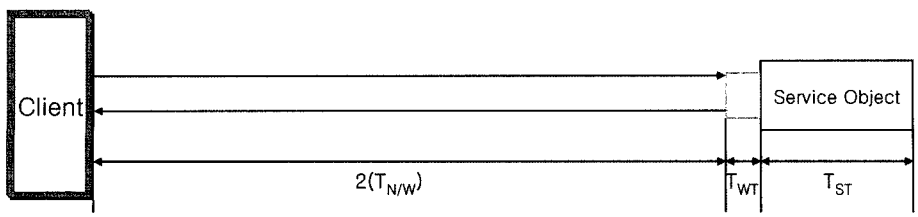


그림 6 바인딩 처리 시간

도메인	형태	T_{lookup}	T_{ds}	T_S
D_1A	비중복	30	0	**30**
	중복	35	323.6	**358.6**
D_2B	비중복	28	0	**28**
	중복	34	322.6	**356.6**
D_3C	비중복	32	0	**32**
	중복	37	325.3	**362.3**

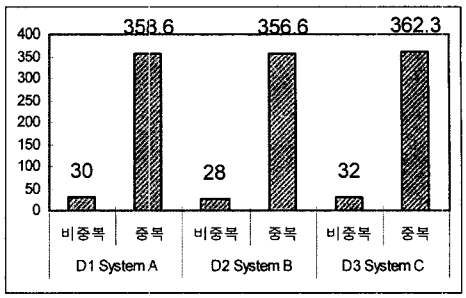


그림 7 각 영역별 통합 바인딩 서비스의 Lookup 시간 및 객체 선정 시간

인에 있는 통합 바인딩 서비스를 사용하는 것보다 Lookup 시간 및 선정 시간이 적게 소요된다. 즉, D_1이나 D_3 관점에서는 외부 도메인에 위치함으로 네트워크 지연시간에 의해 다른 도메인에 위치함으로 인하여 Lookup 시간 및 선정 시간은 높아진다.

5.3.2 외부 도메인상의 통합 바인딩 서비스의 Lookup 시간 및 선정 시간

표 3에서 나타난 바와 같이 클라이언트가 위치한 동일 도메인(D_1)상이 아닌 외부 도메인 (D_2, D_3)상에 서로 다른 도메인상에 컨택 주소가 중복된 경우에 위치한 통합 바인딩 서비스의 Lookup 시간 및 선정 시간은 네트워크 지연시간의 영향으로 지연됨을 알 수 있다.

5.3.3 바인딩 처리 시간

표 4는 클라이언트가 각 도메인상의 특정 시스템 상에 있는 서비스를 제공하는 객체와 바인딩하여 처리되는 시간을 측정한 결과 값이다.

표 4 바인딩 처리 시간

분류	바인딩 처리 시간(T_B)	객체가 위치한 시스템
D_1에 위치한 특정 시스템	36	210.112.129.39
D_2에 위치한 특정 시스템	66.5	203.246.100.34
D_3에 위치한 특정 시스템	107.5	192.203.139.168

클라이언트가 도메인 D_1에 위치하기 때문에 동일 도메인상의 서비스를 제공하는 객체와의 바인딩 처리 시간이 외부 도메인 D_2, D_3상에 위치한 객체와 바인딩하여 처리 시간보다 적게 소요된다.

5.3.4 전체 서비스 처리 시간

전체 서비스 처리 시간은 각 도메인 별 중복, 비중복으로 구분하여 Lookup 시간 및 선정 처리 시간(TS)과 바인딩 처리 시간(TB)을 합하여 소요되는 시간을 측정하였다.

(1) 각 도메인 별 비중복, 중복인 경우

표 5에서 나타난 바와 같이 전체 서비스 처리 시간도 Lookup 시간 및 선정 처리시간과 동일하게 동일 도메인 D_1 상에 위치한 시스템에 서비스 객체와 바인딩하여 서비스를 제공받는 시간이 다른 도메인 D_2나 D_3의 시스템 상에 위치한 서비스 객체와 바인딩 시간이 적게 소요됨을 알 수 있다.

표 5 각 도메인 별 비중복/중복인 경우의 전체 서비스 시간

도메인	형태	적용 수식	T_S
D_1	비중복	L1 +T_{B1}	66
	중복	L2 +T_{B1}	394.6
D_2	비중복	L3 +T_{B2}	229.5
	중복	L5 +T_{B2}	552.1
D_3	비중복	L4 +T_{B3}	273.5
	중복	L6 +T_{B3}	635.8

(2) 서로 다른 도메인에 중복되어 있는 경우

표 6은 서로 다른 도메인 상에 서비스 객체들이 중복되어 있는 경우 전체 서비스 시간을 나타낸다.

서로 다른 도메인에 각각 중복되어 있는 경우에도 동일 도메인 즉 클라이언트와 서비스 객체가 같은 도메인에 있는 경우에 전체 서비스 시간이 적게 소요됨을 알

표 3 외부 도메인 상에 위치한 통합 바인딩 서비스의 Lookup 시간 및 선정 시간

도메인	형태	적용 수식	T_S
D_2B	비중복	L3	**163**
	중복	L4	**485.6**
D_3C	비중복	L5	**166**
	중복	L6	**528.3**

도메인	적용 수식	T_S
D_1A, D_2B	L7	**515**
D_1A, D_3C	L8	**517**
D_2B, D_3C	L9	**516**
D_1A, D_2B, D_3C	L10	**561**

표 6 서로 다른 도메인 상에 중복되어 있는 경우의 전체 서비스 시간

도메인	적용 수식	T_S
D_1, D_2	$L7 + T_{B1}(D_1)$	551
	$L7 + T_{B1}(D_2)$	581.5
D_1, D_3	$L8 + T_{B1}(D_1)$	553
	$L8 + T_{B3}(D_3)$	624.5
D_2, D_3	$L9 + T_{B2}(D_2)$	582.5
	$L9 + T_{B3}(D_3)$	623.5
D_1, D_2, D_3	$L10 + T_{B1}(D_1)$	597
	$L10 + T_{B2}(D_2)$	627.5
	$L10 + T_{B3}(D_3)$	668.5

수 있다.

5.3.5 부하를 고려한 전체 서비스 처리 시간

앞에 보인 실험 결과는 부하를 고려하지 않았을 경우의 전체 서비스 처리 시간이 동일 도메인 상에 있는 서비스 객체와 바인딩하여 서비스를 제공받는 시간이 적게 소요됨을 알 수 있었다. 그러나 실제 해당 시스템의 부하에 따르는 전체 서비스 처리 시간과 네트워크 지연시간에 영향을 받아 전체 서비스 소요 시간이 달라진다. 따라서 시스템 부하와 네트워크 지연시간을 고려하여 다음과 같이 실험하였다. 먼저 각각 서비스 객체가 위치한 시스템에 부하를 10%, 20%, 40%, 60%, 80% 각각 주고, 실제 네트워크 지연시간을 측정하고, 이를 기반으로 서로 다른 도메인 상에 각각 중복되어 있는 경우를 들어 전체 서비스 처리 시간을 측정하였다.

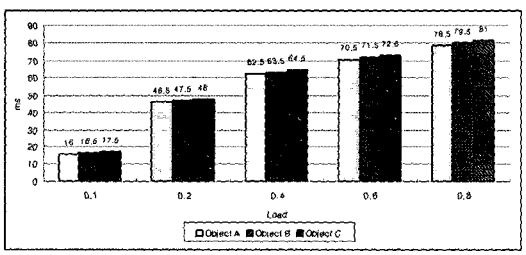


그림 8 각 시스템의 부하를 고려한 서비스 객체의 대기 시간

그림 8은 객체가 위치한 시스템에 부하의 변화를 주었을 때, 서비스 객체의 대기 시간을 측정한 결과를 나타낸다. Object A는 클라이언트와 동일한 도메인 D_1에 위치한 서비스 객체이며, Object B, C는 각각 외부 도메인인 D_2와 D_3에 위치한 서비스 객체이다. 측정 결과 부하가 증가할수록 서비스 객체가 클라이언트에게 서비스를 제공하기 위한 대기 시간이 증가됨을 알 수 있다.

그림 9는 서로 다른 도메인에 중복되어 있는 경우 시스템 부하에 따르는 실제 네트워크 지연시간을 고려한 전체 서비스 시간에 대한 결과를 보였다. 먼저 동일 도메인 D_1에 그리고 서로 다른 도메인 D_2, D_3에 각각 위치한 객체에 대한 전체 서비스 처리 시간 결과를 보인다.

X 좌표는 시간 단위로 2시간 단위로 측정하였다. Y 좌표는 부하부분으로 S1 : 10%, S2 : 20%, S3 : 40%, S4 : 60%, S5 : 80%로 각각 객체가 위치한 시스템에 부하를 주었다. Z 좌표는 전체 서비스 처리 시간을 ms 단위로 측정하였다. (a)는 클라이언트와 같은 도메인 상에 위치한 서비스 객체에 대한 전체 서비스 처리 시간으로 시스템 부하와 실제 네트워크 지연 시간을 고려한 결과 577ms~643ms가 소요된다. (b)는 외부 도메인 D_2에 위치한 객체에 대한 전체 서비스 처리 시간의 결과를 나타낸다. 클라이언트와 서비스 객체가 서로 다른 도메인에 각각 위치한 경우, 전체 서비스 처리 시간은 591ms~704ms가 소요된다. (c)는 클라이언트와 다른 도메인 D_3상에 위치한 객체에 대한 서비스 처리 시간의 결과로 전체 서비스 처리 시간은 646ms~758ms가 소요된다. 즉, 부하에 따라 전체 서비스 처리 시간이 증대됨을 결과로 알 수 있었으며, 동일한 부하일 경우에 클라이언트와 서비스를 제공하는 객체가 같은 도메인에 위치할 경우 전체 서비스 처리 시간에 드는 소요시간이 적게 든다. 그러나 동일 도메인 D_1상의 시스템의 부하가 80%일 경우에는 641ms의 전체 서비스 처리 시간이 소요되며, 같은 서비스를 제공하는 객체가 외부 도메인 D_2와 D_3에 위치할 경우 각각 부하가 D_2에는 10%로 D3에는 20% 일 경우 소요되는 전체 서비스 처리 시간은 612ms와 686ms가 소요된다. 여기에서 특정 시간에 전

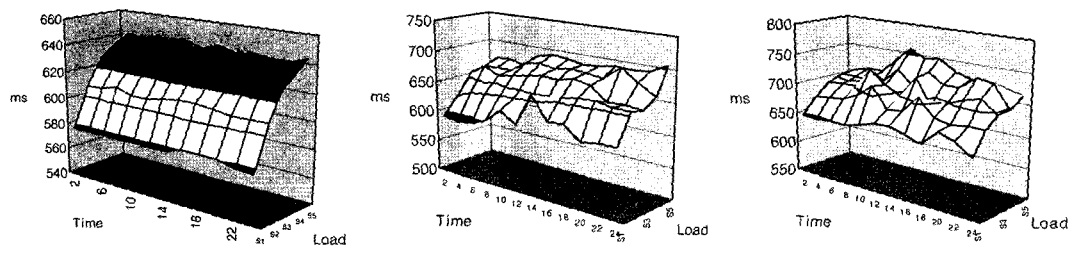


그림 9 서로 다른 도메인에 위치한 객체에 대한 전체 서비스 처리 시간

체 서비스 처리 시간을 측정할 경우, 시스템 부하와 네트워크 지연시간에 따라 동일 도메인 D_1에 위치한 객체에 바인딩하여 서비스 받는 시간보다 외부 도메인 D_2에 위치한 시스템상의 서비스 객체에 서비스 받는 시간이 적게 소요됨을 검증하였다.

5.3.6 기존 검색방법과 제안한 모델의 전체 서비스 처리 시간

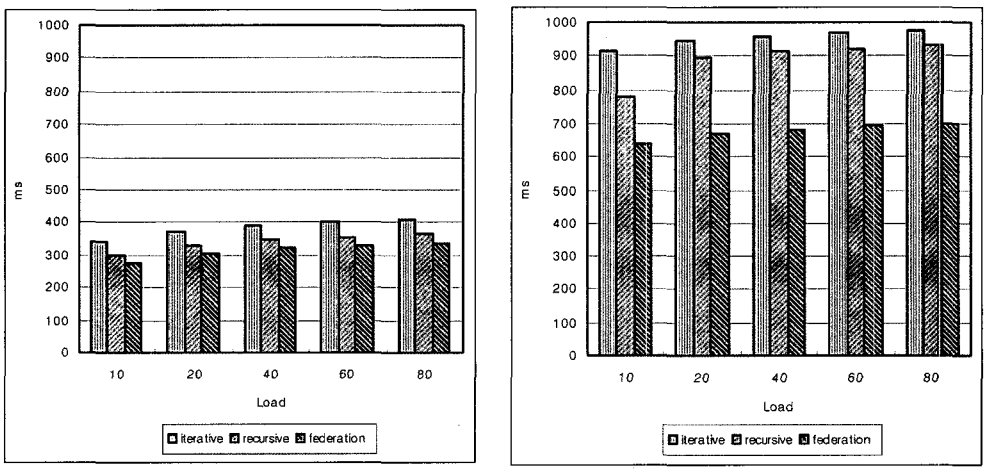
앞선 실험결과를 기반으로 기존의 반복 검색과 재귀 검색방법 그리고 제안한 연합 모델의 전체 서비스 처리시간을 비교한 결과로 반복 검색과 재귀 검색방법은 광역 통합 트리 구조상에서 다른 도메인상의 정보를 얻기 위해 중간 노드 또는 근 노드를 경유하기 때문에 노드의 수가 많아질수록 각 노드별 통신비용과 서비스 처리비용이 추가가 됨으로 전체 서비스 처리 시간은 증가된다.

(a)의 결과는 외부 도메인 D_3에 위치한 비중복 서비스 객체와 바인딩하는 경우로 광역 통합 트리의 논리적 구조에 적용하였다. 도메인 D_1, D_2, D_3는 서로 다른 도메인으로 상위 노드를 갖는다. 여기에서 검색 방법 전체 서비스 처리 시간을 측정하여 비교한 결과 반복 검색 방법과는 71ms의 차이가 있으나, 재귀 검색 방법과는 26ms로 별 차이를 보이지 않음을 알 수 있다. 이와 마찬가지로 그림 10의 (b)는 각 검색 방법 별 중복된 객체일 경우에 전체 서비스 처리 시간을 비교한 결과를 보인다. 비중복 객체일 경우와는 달리 현저하게 제안한 모델의 전체 서비스 처리 시간이 반복 검색과는 276ms의 차이를 보이며, 재귀 검색 방법에 213ms의 차이를 보인다. 결과에 따라 비중복일 경우에는 재귀 검색 방법과 제안한 모델과의 성능의 차이가 적지만, 중복일 경우에는 기존 검색방법인 반복 검색과 재귀 검색 방법에 비해서 성능이 우수함을 알 수 있다.

6. 결 론

인터넷 기술의 발전에 따라 시스템 규모가 점차 커지고, 중앙 집중형 시스템에서 분산시스템들의 협력 형태인 광역 연합환경의 분산시스템으로 변화되고 있다. 현재 인터넷 기반의 웹서비스와 전자메일과 같은 광역 분산 서비스들이 클라이언트에게 제공되고 있지만, 국부적인 분산 투명성만을 제공하고 있다. 특히, 이러한 분산 투명성을 제공하기 위한 연구로 OMG의 CORBA를 비롯하여, 마이크로소프트의 DCOM 그리고 SOS의 단편화 객체 모델, 썬 마이크로시스템스의 스프링 객체 모델, Globe의 분산 공유 객체 모델과 같은 분산 객체 컴퓨팅 모델을 제시하였다. 이들 연구 대부분에서 서비스 객체에 대한 위치 서비스를 네이밍 기반의 시스템을 사용하고 있다. 그러나 객체를 식별하는데 이름만으로는 광역 환경에서는 클라이언트의 요구에 맞는 서비스를 제공하기엔 한계가 있다. 따라서 트레이더의 기능이 필수적이다. 그러나 문제점으로는 위치에 종속된 식별자를 사용하는데 있다. 이러한 위치에 종속된 식별자들은 이주와 복사 투명성 처리를 매우 어렵게 한다. 분산 객체들은 매번 위치가 변경되거나 복사될 때 추가되고 삭제된다. 특히, 광역 컴퓨팅 환경에서는 수많은 객체들이 이름이나 속성에 의해 중복됨을 예측하게 한다.

따라서, 단일 객체뿐만 아니라 이름 또는 속성으로 중복된 객체들의 효율적인 관리와 시스템간의 부하균형화를 유지하며, 객체가 위치한 시스템의 부하가 적은 객체를 선정한 바인딩 서비스를 제공하고자 한다. 이름과 속성 모두를 지원하는 네이밍과 트레이더 기능이 통합된



(a) 비중복인 경우 (b) 중복인 경우

그림 10 검색방법 별 비중복일 경우의 전체 서비스 처리 시간

서비스와 실제 서비스를 제공하는 객체의 컨택 주소를 관리하는 위치 서비스를 각각 독립적으로 수행하여 위치에 종속적인 식별자의 문제점을 해결하였다. 또한, 관리적인 영역 개념을 도입하여 Cooperator 기반의 서로 다른 도메인에 위치하는 객체들의 광역 바인딩을 제공하는 광역 통합 바인딩 서비스의 연합 모델을 제시하였다.

본 논문에서 제시한 통합 바인딩 서비스 연합 모델에 대한 실험을 위해 Cooperator와 네이밍/트레이딩 서비스와 위치 서비스를 특정 시스템에 각각 위치시키고, 통합 바인딩 서비스가 관리하는 영역으로 3개의 도메인을 구성하였다. 실험 환경은 워크스테이션 1대와 PC 2대 그리고 부하 정보를 추출하기 위해 LSF 도구를 사용하였다.

본 논문에서 제시한 연합 모델에 대한 실험은 각 도메인 별 통합 바인딩 서비스의 처리 성능을 평가하고, 연합 모델 환경 하에서 비중복, 중복 객체의 전체 바인딩 서비스 처리의 성능을 평가하였다. 특히, 분산 객체가 위치한 시스템 부하에 따라 전체 바인딩 서비스의 처리 시간을 분석하였다. 그리고 광역 통합 트리 구조상에서 반복 검색과 재귀 검색방법과 제안한 연합 모델과 비교하여 성능이 우수함을 보였다.

향후 연구로는 논문에서 제시한 내용을 근거로 분산 중복 객체들의 최적통합 바인딩 서비스를 위한 지능형 모델 기반의 부하분배에 관한 규칙을 만들고, 연합 모델에 타입 메커니즘을 적용하는 연구가 필요하다. 그리고 이를 기반으로 다양한 위치기반 서비스 애플리케이션에 적용하고자 한다.

참 고 문 헌

[1] Robert Orfali, Dan Harkey and Jeri Edwards, The Essential Client/Server Survival Guide, Van Nostrand Reinhold, 1994.

[2] M. van Steen, P. Homburg, L. van Doorn, A.S. Tanenbaum, and W. de Jonge, "Towards Object-based Wide Area Distributed Systems," Proc. Fourth Int'l Workshop on Object Orientation in Operating Systems, IEEE, Lund, Sweden, Aug. 1995, pp. 224-227.

[3] Deschrevel, J. P, The ANSA Model for Trading and Federation, Architectural report 005, Architecture Projects Management Ltd., Poseidon House, Castle Park, Cambridge CB30RD, UK(1993).

[4] 전병택, 정창원, 주수종, "광역 분산 객체들의 바인딩 지원을 위한 연합 네이밍/트레이딩 모델", 2001년 춘계 학술발표 논문집, 제28권, 1호, 정보과학회, pp. 427-429, 2001.

[5] 전병택, 정창원, 주수종, "광역 객체 컴퓨팅 환경에서 분산 객체의 관리를 위한 서비스 모델의 설계", 추계 학술발표 논문집, 제8권, 2호, 정보처리학회, pp. 309-312, 2001.

[6] 전병택, 정창원, 주수종, "광역 객체 컴퓨팅 환경에서 분산 객체의 통합 바인딩 서비스의 최적 객체 선정", 춘계 학술발표 논문집, 제9권, 1호, 정보처리학회. pp. 1499-1502, 2002.

[7] Dirk Thißen, Helmut Neukirchen. "Integrating Trading and Load Balancing for Efficient Management of Services in Distributed Systems," Proceedings of the 3rd IFIP/GI International Conference on Trends towards a Universal Service Market (USM 2000), Munich, 2000.

[8] Neukirchen, H, "Optimizing the Set of Selected Service in a CORBA Trader by Integrating Dynamic Load Balancing," Diploma thesis at the Department of Computer Science, Infomatik Ⅳ, Aachen University of Technology, 1999.

[9] Elarbi Badidi, Rudolf K. Keller, Peter G. Kropf, Vincent V. Dongen, "The Design of a Trader-based CORBA Load Sharing Service," Proc. Twelfth International Conference on Parallel and Distributed Computing Systems (PDCS'99), pp. 75-80.

[10] 강명석, 정창원, 주수종 "광역 객체 컴퓨팅 환경에서 부하를 고려한 선정된 객체의 통합 바인딩 서비스의 구축", 추계 학술발표 논문집, 제9권, 2호, 정보처리학회, pp. 1487-1490, 2002.

[11] 정창원, 오성권, 주수종, "광역 객체 컴퓨팅 환경에서 이름/속성기반의 통합 바이딩 서비스 방안", 정보처리학회 논문지A, 제9-A권, 2호, pp. 241-248, 2002. 6.

[12] "광역 객체 컴퓨팅 환경에서 부하를 고려한 통합 바인딩 서비스의 설계 및 구현," 한국정보과학회 논문지, 제30권 3호, pp. 293-306, 2003.6.

[13] CW Jeong, SC JOO, SK Han, "Integrated Binding Service Model for Supporting Both Naming/Trading and Location Services in Inter/Intra-Net Environments," Lecture Notes in Computer Science, Vol. 3032, pp. 754-761, 2003. 12.

[14] Wide_Area systems research, http://ringer.cs.utsa.edu/faculty/jon/wa.html.

[15] P. Homburg, M. van Steen, and A.S. Tanenbaum. "An Architecture for a Wide Area Distributed System," Proc. Seventh ACM SIGOPS European Workshop, Connemara, Ireland, Sep. 1996, pp. 75-82.

[16] Axel Kupper, "Locating TINA User Agents : Strategies of a Broker Federation and their Comparison," Proceedings of 6th International Conference on Intelligence in Services and Networks, IS&N 99. Barcelona, Spain, 27~29 April, 1999.

[17] L. Augusto and E. Madira, "A Model for a Federated Trader," Proc. International Conference on Open Distributed Processing, Brisbane, Australia, Feb. 1995.

[18] Y.C. Hu, D.A. Rodney, and P. Druschel, Design and scalability of NLS, A scalable naming and location service, tech. report TR01-381, Rice

University, June, 2001.
[19] Michael Christoffel, "Cooperations and Federations of Traders in An Information market," Proc. 2nd International Conference on Electronic Commerce, York, March, 2001.
[20] 정창원, 주수종, "객체그룹간의 상호접속을 지원하는 연합 트레이더 모델", 한국정보과학회 논문지, 제26권, 9호, pp. 1126-1134, 1999.

정 창 원

1993년 원광대학교 컴퓨터공학과 졸업 1998년 원광대학교 컴퓨터공학과 졸업(석사). 2003년 원광대학교 컴퓨터공학과 졸업(공학박사). 2004년~2006년 전북대학교 차세대 LBS 응용 연구센터 연구교수. 2006년~현재 원광대학교 전기전자 및 정보공학부 Post Doc. 관심분야는 분산객체 컴퓨팅, 멀티미디어 데이타베이스, LBS, 텔레매틱스

주 수 종

1986년 원광대학교 전자계산공학과 졸업 1988년 중앙대학교 컴퓨터공학과 졸업(공학석사). 1992년 중앙대학교 컴퓨터공학과 졸업(공학박사). 1993년 미국 University of Massachusetts at Amherst, Post-Doc. 2003년 미국 University of California at Irvine, Visiting Professor. 1990년~현재 원광대학교 전기전자 및 정보 공학부 교수. 관심분야는 분산실시간 컴퓨팅, 분산객체모델, 시스템 최적화, 멀티미디어 데이타베이스