제18회 한국정보처리학회 추계학술발표대회 논문집 제9권 제2호 (2002.11)

광역 객체 컴퓨팅 환경에서 부하를 고려한 선정된 객체의 통합 바인딩 서비스의 구축

강명석*, 정창원*, 주수종*
*원광대학교 컴퓨터공학과
e-mail:gnb@wonkwang.ac.kr

A Construction of Integrated Binding Service of The Selected Objects Considering Loads in Wide-Area Object Computing Environments

Myung-Suk Kang, Chang-Won Jeong*, Su-Chong Joo*
*Dept of Computer Engineering, Won-Kwang University

요 약

최근 분산 컴퓨팅 환경은 급진적으로 광역화되고, 이질적이며, 연합형태의 광역 시스템 구조로 변화하고 있다. 이러한 환경은 네트워크상에 광범위한 서비스를 제공하는 통신 네트워크 기반에서 구현된 수많은 객체로 구성된다. 더욱, 지구상에 존재하는 모든 객체들은 이름이나 속성에 의해 중복된 특성을 갖는다. 이러한 같은 특성을 갖는 객체들은 중복 객체로 정의된다. 그러나 기존의 네이밍이나 트레이딩 메커니즘은 독립적인 위치 투명성이 결여로 중복된 객체들의 바인딩 서비스 지원이 불가능하다. 서로 다른 시스템 상에 존재하는 중복된 객체들이 동일한 서비스를 제공한다면, 각 시스템의 부하를 고려하여 클라이언트의 요청을 분산시킬 수 있다. 이러한 이유로 본 논문에서는 광역 컴퓨팅 환경에서 중복된 객체들의 위치 관리뿐만 아니라 시스템들간의 부하 균형화를 유지하기 위해서 최소부하를 갖는 시스템에 위치한 객체의 선정하여 동적 바인딩 서비스를 제공할 수 있는 새로운 모델을 설계하고 구현하였다. 이 모델은 네이밍 및 트레이딩 기능을 통합한 서비스에 의해 중복된 객체들에 대한 단일 객체 핸들을 얻는 부분과, 얻어진 객체핸들을 사용하여 위치 서비스에 의해 하나 이상의 컨택 주소를 얻는 부분으로 구성하였다. 주어진 모델로부터, 우리는 Naming/Trading 서비스와 위치 서비스에 의한 전체 바인딩 메커니즘의 처리과정을 나타내고, 통합 바인딩 서비스의 구성요소들에 대한 구조를 상세하게 기술하였다. 끝으로 우리의 모델을 구현하기 위해, 윈도우 운영체제와 Solaris 2.5/2.7에서 사용되는 CORBA 사양을 따르는 VisiBroker 4.1과 자바 언어, SQL Server 2000 그리고 LSF를 이용하였다. 그리고 구현 환경과 구성요소에 대한 수행 화면을 보였다.

1. 서론

분산 컴퓨팅의 환경은 인터넷 기술의 발전으로 인하여 점차 광역의 컴퓨팅 환경으로 변화되고 있다. 이러한 환경에서 분산 투명성[1]을 제공하기 위한 연구가 활발하게 진행되고 있다. 대표적인 연구로는 OMG의 CORBA, Microsoft DCOM등이 있다. 이들 연구들은 공통적으로 분산 시스템을 구성하는 객체들의 효과적인 관리와 검색 서비스(discovery system)를 제공하기 위한 디렉토리, 이름 서비스 그리고 트레이딩 서비스[1,2]를 제공한다. 이러한 서비스는 전통적으로 객체 대 주소 매핑(object-to-address)을 제공하고 있다. 이와 같은 방법은 객체의 위치가 변경하였을 때, 객체의 식별자와 위치 정보의 갱신을 피할 수 없게 한다. 그리고 객체의 이름을 사용자가 변경하였을 때에도 마찬가지로 갱신해야만 한다[1,4,6].

또한 이러한 환경에서 동일한 서비스를 제공하는 수많은 객체들이 네트워크 상에 분산되어 존재하게 된다. 이로 인하여 클라이언트들이 임의의 검색 서비스를 이용하여 객체를 찾을 경우, 해당 검색 서비스에서 관리하는 객체들은 이름과 속성으로 중복되어 클라이언트의 요구사항에 대응하는 객체를 찾기란 어렵다. 여기에는 클라이언트가 요청한 객체를 선정하기 위한 전략이 필요하다[2,3].

따라서, 본 논문에서는 광역 컴퓨팅 환경에서 중복된 객체들의 위치 관리뿐만 아니라 중복된 객체들 중에 부하를 고려하여 객체를 선정 할 수 있는 광역 통합 바인딩 서비스 모델을 제시한다. 이 모델은 서비스를 제공하는 객체에 대해 서비스 제공자가 정의한 이름과 속성을 서비스 오퍼 단위로 저장하고, 이를 근거로 클라이언트가 검색할 수 있는 Naming/Trading 서비스와 객체들의 위치 정보를 컨택 레코드로 관리하는 위치 서비스로 구성하였다. 객체들의 이동을 지원하기 위해 Naming/Trading 서비스와 위치 서비스를 각각 독립적으로 운영하며, 이들 서비스가 관리하는 정보의 매핑은 객체를 식별할 수 있는 객체 핸들에 의해 이루어진다. 이로 인하여 객체의 위치가 변경되더라도 위치 서비스가 관리하는 컨택 레코드의 위치 정보만 변경하도록 하였다. 또한, 중복된 객체들 중에 하나의 객체를 선정하기 위해 객체가 위치한 시스템의 부하정보를 이용하여, 부하가 가장 적은 객체를 선정하도록 하였다[2]. 광역 객체 컴퓨팅을 위한 하부 구조는 CORBA 사양을 따

본 연구는 한국과학재단 목적기초연구 (2000-1-30300-010-2) 지원으로 수행되었음.

르는 Borland의 VisiBroker 4.1을 사용하였으며, 통합 바인딩 서비스의 구성요소인 Naming/Trading 서비스와 위치 서비스는 자바(Java2 SDK 1.3.1)로 개발하였다. 서비스 오퍼와 컨택 레코드 정보를 관리하기 위해 Microsoft의 관계형 데이터베이스 시스템인 SQL Server 2000을 이용하였으며, 각 시스템의 부하정보를 획득하기 위해 LSF 4.1을 이용하였다.

본 논문의 구성을 보면, 2장은 우리가 제안한 통합 바인딩 서비스 모델에 대해 기술하였다. 3장에서는 이에 대한 개발환경과 서비스 수행 결과화면을 보인다. 4장에서는 결론 및 향후 연구 내용에 대해 다룬다.

2. 광역 통합 바인딩 서비스 모델

대부분의 분산 시스템에서 네이밍 시스템은 이름과 네트워크 주소의 매핑을 관리한다. 이 방식은 2가지 문제점을 갖는다. 첫 번째는 광역 네이밍 시스템에서 이름 대 주소 바인딩 방법은 변경하기가 어렵다. 두 번째는 대부분의 네이밍 시스템들은 서로 다른 범위에 이름 공간이 분산되어 있으며, 위치에 종속된 이름을 사용한다. 이러한 위치에 종속된 이름들은 이주와 복사 투명성 처리를 매우 어렵게 한다[1]. 분산 객체들은 매번 위치가 변경되거나 복사될 때 추가되고 삭제된다. 이는 트레이딩 서비스도 마찬가지로 적용되는 문제점이다. 또한 이러한 문제점을 해결하더라도 동일한 서비스를 제공하는 객체가 중복되어 있을 경우, 하나의 객체에 집중하는 문제점을 갖게 된다. 본 장에서는 이러한 문제점을 해결하는 모델에 대해 처리 과정과 각 구성요소의 기능에 대해 기술한다.

2.1 광역 통합 바인딩 과정

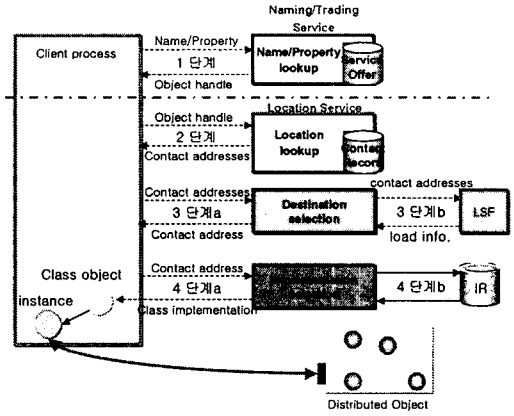
광역 통합 바인딩 과정은 크게 두 단계로 이루어지며, 첫 번째 단계는 Naming/Trading Service에 의해 객체들의 유일한 식별자인 객체 핸들을 얻어오는 단계이며, 두 번째 단계는 위치 서비스에서 Naming/Trading 서비스로부터 얻어진 객체 핸들에 대한 컨택 주소를 얻는 과정이다. 이는 세부적으로 분류하면 4단계로 나뉜다. 다음 (그림 1)은 이에 대한 처리 과정을 보인다.

1 단계 : Name/Property lookup은 사용자가 정의한 이름 또는 속성을 매개변수로 lookup 메소드 호출에 의해 객체 핸들로 반환된다.

2 단계 : Location lookup은 1단계에서 얻어진 객체 핸들은 객체에 액세스하기 위한 컨택 주소의 집합으로 변환된다. 중복되었을 경우는 객체 핸들당 여러 개의 컨택 주소가 반환된다.

3 단계 : Destination selection은 객체 핸들과 매핑 되는 컨택 주소의 집합 중에 하나의 컨택 주소를 선택하는 과정으로 해당 객체가 위치한 시스템에 대한 부하정보를 비교하여 최소 부하를 갖는 시스템에 존재하는 객체의 컨택 주소를 반환한다.

4 단계 : Implementation selection은 해당 시스템의 구현 객체를 선택하는 과정으로 실제 구현 객체와 바인딩하기 위해 클라이언트 프로세스는 새로운 지역 객체를 인스턴스화하고 분산 객체를 초기화한다.

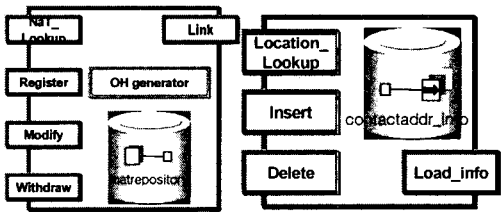


(그림 1) 통합 바인딩 서비스 처리 과정

다음은 통합 바인딩 서비스 모델을 이루는 구성요소들에 대해 세부적으로 설명한다.

2.2 광역 통합 바인딩 서비스의 구성요소

광역 객체 컴퓨팅 환경기반에서 기존의 단일 객체의 바인딩은 물론, 중복객체의 바인딩 서비스까지 지원할 수 있는 Naming/Trading 서비스는 이름/속성 별 객체 핸들과 매핑을 제공한다. 위치 서비스는 객체 핸들을 컨택 주소들의 집합으로 매핑을 제공한다. 다음 (그림 2)는 구성요소를 보인다.



(a) Naming / Trading (b) 위치 서비스

(그림 2) 광역통합 바인딩 서비스의 구성요소

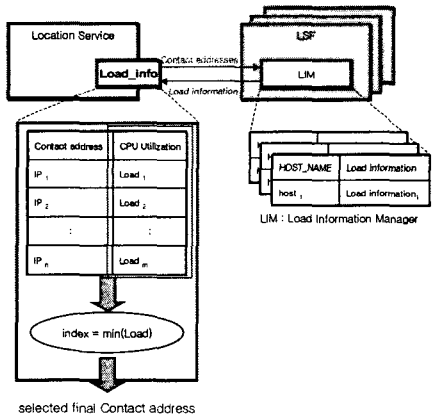
Naming/Trading 서비스는 서비스를 제공하는 객체를 등록 할 때 유일한 객체 핸들을 부여한다. 객체 핸들은 해당 객체가 생명주기가 끝날 때까지 존재하며, 변경되지 않는다. 하나의 객체가 복사되거나, 이동을 하더라도 동일한 객체 핸들을 갖는다. 그리고 이러한 객체의 세부적인 정보는 natrepository의 offer_info 테이블과 property_info 테이블에 저장된다. 제공하는 인터페이스에서 NaT_Lookup()은 클라이언트의 요구사항을 만족하는 객체의 객체 핸들을 찾기 위한 기능이며, Register()는 등록할 객체의 서비스를 알리도록 Naming/Trading 서비스가 갖는 저장소에 저장한다. 그리고 이를 수정하거나 삭제는 Modify()와 Withdraw()이다. 그리고 Link()는 다른 Naming/Trading 서비스와 연합 할 수 있도록 연결시켜주는 오퍼레이션이며, 내부적인 오퍼레이션으로 Object Handle(OH) generator()은 등록되는 객체의 유일한 객체

핸들을 부여한다.

위치 서비스는 Naming/Trading 서비스에 등록된 객체들의 실질적인 주소를 저장하는 기능을 갖는다. 인터페이스는 Location_Lookup()과 시스템의 부하정보를 얻기 위한 Load_info(), 그리고 컨택 주소들의 집합을 삽입하거나 삭제하기 위한 Insert(), Delete() 오퍼레이션을 갖는다. 객체에 대한 위치정보를 관리하기 위한 저장소는 contact addr_info라는 테이블에서 관리한다.

2.3 부하를 고려한 객체 선정 알고리즘

다음 (그림 3)은 각 시스템의 부하정보를 획득하여 최소 부하를 갖는 시스템에 존재하는 객체를 선정하기 위한 알고리즘을 보인다. Naming/Trading 서비스로부터 얻어진 객체 핸들과 매핑된 컨택 주소들을 각 시스템의 LSF에게 전달하면 LSF의 LIM에 의해 시스템의 모니터링된 부하정보를 위치 서비스에게 반환하게 된다. Load_info 오퍼레이션은 시스템의 상태를 먼저 확인하고, 부하 정보 중에 CPU 이용율만 추출하여 이들 값들을 비교한 후 가장 적은 부하를 갖는 컨택 주소를 선정하여 반환한다.



(그림 3) 위치 서비스에서 부하를 고려한 객체 선정을 위한 알고리즘

3. 광역 통합 바인딩 서비스의 설계 및 구현

본 장에서는 앞서 제시한 통합 바인딩 서비스에 대한 각 구성요소에 대한 IDL 설계에 대해 기술하고, 설계 내용과 이에 대해 검증을 위해 윈도우 화면으로 Naming/Trading 서비스와 위치 서비스의 수행 결과화면을 특정 웹 상의 객체를 이름/속성 검색을 통한 바인딩 결과를 보인다.

3.1 각 구성요소의 IDL 설계

앞서 언급한 광역 통합 바인딩 서비스의 구성요소의 오퍼레이션에 따라 각 구성요소의 IDL은 다음과 같다.

다음 (그림 4)는 Naming/Trading 서비스의 IDL을 나타낸다.

```
// Naming/Trading 서비스 오퍼레이션 인터페이스
  interface NaT_Operation
  {
    // 객체 핸들 검색
   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 삭제
    void Withdraw(in ObjectHandle objhandle);
    // 링크
    void Link(in string subpoint);
  };
```

(그림 4) Naming/Trading 서비스의 IDL

다음 (그림 5)는 위치 서비스의 IDL을 나타낸다.

```
// 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 정보를 참조하여 적절한 컨택주소의 탐색
    ContactAddr Load_Info(in conaddrSeq conaddrs);
  };
```

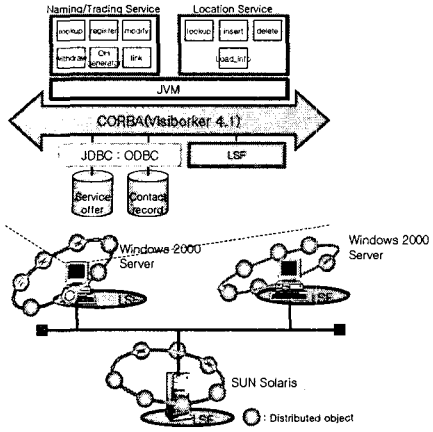
(그림 5) 위치 서비스의 IDL

3.2 구현

구현 환경은 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 컴퓨팅 환경에는 Borland의 VisiBroker 4.1을 사용하였으며, 통합 바인딩 서비스의 구성요소인 Naming/Trading 서비스와 위치 서비스는 자바로 개발하였다. 서비스 오퍼와 컨택 레코드 정보를 관리하기 위해 Microsoft의 관계형 데이터베이스 시스템인 SQL Server 2000을 이용하였으며, 각 시스템의 부하정보를 획득하기 위해 LSF 4.1을 이용한다.

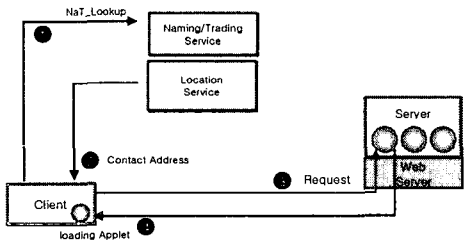
통합 바인딩 서비스의 구성요소인 Naming/Trading 서비스와 위치 서비스는 하나의 특정 시스템에 위치시키고, Naming/Trading 서비스가 분산 객체를 관리하는 영역으로 하나의 도메인을 구성하였다. 그리고 분산 객체는 도메인 내에 각 시스템에 배치한다. 또한 시스템의 부하정보를 얻기 위한 LSF 4.1은 각 시스템에 위치하여 부하정보를 모니터링 한다. 각 구성요소의 인터페이스는 IDL로 작성하였으며, Visiborker4.1 IDL 컴파일로 컴파일 하였다. 또한, 검증을 위한 그래픽 사용자 인터페이스(GUI)는 자바의 경량 컴포넌트인 JFC/Swing을 이용하여 구현하였다. 다음 (그림 6)은 본 연구의 검증을 위한 구현 환경을 나타낸다.



(그림 6) 구현 환경

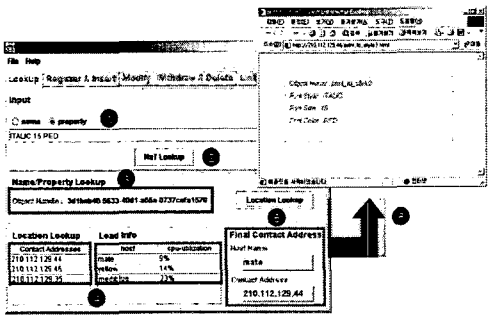
3.3 결과 화면

광역 통합 바인딩 서비스 수행을 검증하기 위해 클라이언트가 통합 바인딩 서비스를 통하여 최소 부하를 갖는 시스템에 존재하는 객체의 컨택 주소를 얻어 서버측에 바인딩 하여 서버로부터 분산 객체인 애플릿을 다운로드하여 클라이언트의 브라우저에 결과를 디스플레이 한다. 이에 대한 과정은 다음 (그림 7)과 같다.



(그림 7) 광역 통합 바인딩 서비스 수행 과정

다음 (그림 8)은 검증을 위한 그래픽 사용자 인터페이스인 윈도우에서 NaT_Lookup 탭에서 속성으로 검색하기 위한 화면과 서버의 서비스결과 화면을 보인다.



(그림 8) 속성으로 검색한 결과 화면

먼저 속성으로 검색하기 위해 속성 값을 기입하고 NaT_Lookup 버튼을 클릭한다. 그 결과 이에 해당하는 객체 핸들은 *"3d1bab30-5533-40d1-a65a-0737cefa1579"*이며 Location_Lookup 버튼을 클릭한 결과 3개의 컨택 주소가 나타났다. 이는 중복된 객체를 의미한다. 따라서 이들 컨택 주소 중에 부하 정보를 고려하여 부하가 최소인 선정된 최종 컨택 주소로 바인딩하여 웹 브라우저로 결과를 보이고 있다.

4. 결론

현재 인터넷 기반의 웹서비스와 전자메일과 같은 광역 분산 서비스들이 클라이언트에게 제공되고 있지만, 국부적인 분산 투명성만을 제공하고 있다. 이러한 분산 투명성을 제공하는 솔루션으로 이름 서비스가 대부분을 차지하고 있다. 그러나 이름만으로는 광역 환경에서는 클라이언트의 요구에 맞는 서비스를 제공하기엔 한계가 있다. 이를 해결하기 위해 속성 기반의 트레이딩 서비스가 출현하게 되었다. 이러한 서비스들이 기반이 된 광역 객체 컴퓨팅 환경에서는 수많은 객체들이 이름이나 속성에 의해 중복됨을 예측하게 한다. 그러나 이들은 객체 대 주소 매핑에 의한 바인딩 서비스를 제공하고 있다. 이는 객체의 이동과 중복성을 해결하지 못하는 단점을 갖는다. 따라서, 단일 객체뿐만 아니라 이름 또는 속성 기반의 중복된 객체들의 효율적인 관리와 최소 부하를 갖는 시스템에 존재하는 객체에 대한 광역 통합 바인딩 서비스 모델을 제안하고, 이에 대해 구현하고, 서비스 수행 결과를 GUI로 보였다.

향후 연구로는 논문에서 제시한 내용을 근거로 서로 다른 도메인상에서 독립적으로 운용하는 광역 통합 바인딩 서비스들간의 연합 모델로 확장하고, 서비스 수행 시간에 대한 성능 평가와 부하에 따르는 성능 평가가 필요하다. 또한, 부하를 측정하기 위한 요소를 추출하여 규칙을 만들고 이를 기반으로 부하분배에 관한 연구가 필요하다.

참고문헌

[1] M. van Steen, F.J. Hauck, G. Ballintijn, A.S. Tanenbaum. "Algorithmic Design of the Globe Wide-Area Location Service." The Computer Journal 41(5):297-310, 1998.

[2] Elarbi Badidi, Rudolf K. Keller, Peter G. Kropf, Vincent V. Dongen, "The Design of a Trading-based COBRA Load Sharing Service" In Proceeding of the Twelfth International Conference on Parallel and Distributed Computing Systems(PDCS' 99), p75~80. 1999.

[3] M. van Steen, F.J. Hauck, G. Ballintijn, A.S. Tanenbaum. "Algorithmic Design of the Globe Wide-Area Location Service." The Computer Journal 41(5):297-310, 1998.

[4] 전병택, 정창원, 주수종, "광역 분산 객체들의 바인딩 지원을 위한 연합 네이밍/트레이딩 모델", 2001년 춘계 학술발표 논문집, 2001.4.28. 제 28권 1호, 정보과학회. p427~429.

[5] 전병택, 정창원, 주수종, "광역 객체 컴퓨팅 환경에서 분산 객체의 관리를 위한 서비스 모델의 설계, 추계 학술발표 논문집, 2001. 10.13. 제 8권 2호, 정보처리학회. p 309~312.

[6] 전병택, 정창원, 주수종, " 광역 객체 컴퓨팅 환경에서 분산 객체의 통합 바인딩 서비스의 최적 객체 선정", 춘계 학술발표 논문집, 2002. 4.12. 제 9권 제 1호, 정보처리학회. p 1499~1502.

[7] 정창원, 오성귀, 주수종, "광역 객체 컴퓨팅 환경에서 이름/속성기반의 통합 바인딩 서비스 방안", 2002. 6. 제 9권-A권 제 2호. 정보처리학회 논문지. p241~248.