PAMUKKALE ÜNİVERSİTESİ MÜHENDİSLİK FAKÜLTESİ
PAMUKKALE UNIVERSITY ENGINEERING COLLEGE
MÜHENDİSLİK BİLİMLERİ DERGİSİ
JOURNAL OF ENGINEERING SCIENCES
YIL : 2001
CİLT : 7
SAYI : 1
SAYFA : 63-69

FARKLI BAKIŞ AÇILARINDAN JAVA RMI VE CORBA'NIN KARŞILAŞTIRILMASI

Mustafa TÜRKSEVER, R. Cenk ERDUR
Ege Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Bornova/İzmir

Geliş Tarihi : 23.06.2000

ÖZET

Bu çalışmada, günümüzde dağıtık sistemlerin nesneye dayalı olarak modellenmesinde en çok kullanılan standartlardan olan CORBA ve Java RMI farklı bakış açılarından karşılaştırılmaktadır. Bu bakış açıları; performans, dağıtık programlama ve sistem düzeyinde sunulan servislerdir.

Anahtar Kelimeler : Java RMI, CORBA, Arayüz tanımlama dili, Dağıtık nesne işlem

COMPARISION OF JAVA RMI AND CORBA FROM DIFFERENT PERSPECTIVES

ABSTRACT

In this paper, two of the widely used distributed object system standards which are CORBA and Java RMI has been compared from different perspectives. These perspectives are performance, distributed programming and system level services.

Key Words : Java RMI, CORBA, Interface definition language, Distributed object computing (DOC)

1. GİRİŞ

Dağıtık nesne işlem (Distributed Object Computing - DOC) paradigması, nesneye dayalı yöntemlerin dağıtık bilgi işlem alanına uyarlanması sonucu ortaya çıkmıştır. Nesneye dayalı yöntemler, yeniden kullanıma izin vermekte, esnek, modüler ve bakımı kolay yazılım üretimini desteklemektedir. Bu nedenle, nesneye dayalı yöntemler, yazılımların kalitesini artıran, geliştirme zamanı ve maliyeti azaltan bir teknoloji olarak günümüz yazılım endüstrisindeki en önemli eğilimlerden birisidir. Dağıtık nesne işlem, nesneye dayalı teknolojinin yararlarını dağıtık sistemlere taşımaktadır. Dağıtık nesne işlem paradigması, açık, çok-katmanlı (multi-tiered) ve birbiriyle işbirliği içinde olan dağıtık nesnelerden oluşan dağıtık sistemlerin tasarımını mümkün kılmaktadır. Böylece, daha güvenilir ve bakımı kolay dağıtık sistemler tasarlanabilecek ve günümüzde telekomünikasyondan tele tıp uygulamalarına kadar birçok alanın önemli bir gereksinimi karşılanmış olacaktır (Saleh et all., 1999).

CORBA (Common Object Request Broker Architecture) mimarisi (Orfali et all., 1997; Orfali and Harkey, 1998; Pedrick et all., 1998). (http://www.omg.org) ve Java RMI (Remote Method Invocation) (Anon., 1998) günümüzde nesneye dayalı dağıtık sistemler geliştirilmesinde en çok kullanılan standartlardır. Bunun dışında, Microsoft kuruluşunun kendine özgü dağıtık nesne standardı DCOM (Distributed Component Object Model) (Anon., 1997) bulunmaktadır.

Bu çalışmada ilk önce CORBA ile Java RMI standartları tanıtılacak, daha sonra bu iki standart farklı bakış açılarından karşılaştırılacaktır. Bu bakış açıları; performans, dağıtık programlama ve verilen servisler olacaktır. İkinci bölümde CORBA mimarisi, üçüncü bölümde Java RMI mimarisi kısaca tanıtılacaktır. Dördüncü bölüm, her iki dağıtık sistem standardının belirtilen bakış açılarından karşılaştırılması ile ilgilidir. Sonuçlar ve tartışma beşinci bölümde, yararlanılan kaynaklar altıncı bölümde yer almaktadır.

2. CORBA

Bu bölümde CORBA standardı ana hatları ile tanıtılacaktır.

2. 1. CORBA Standardına Genel Bakış

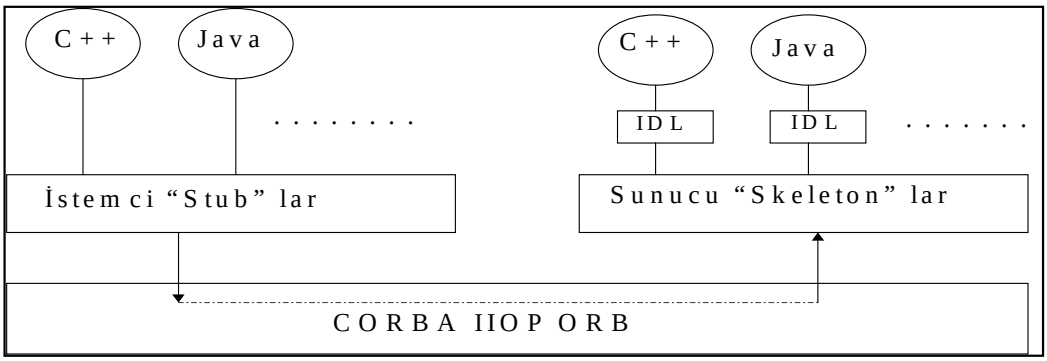
CORBA, OMG (Object Management Group-Nesne Yönetim Grubu) tarafından geliştirilmiş dağıtık nesne mimarisi standardıdır. OMG, bilgisayar endüstrisinde yer alan 800'den fazla kuruluş tarafından desteklenen bir çalışma grubudur. Microsoft sözkonusu destekçi kuruluşlar arasında bulunmamaktadır. Bunun nedeni Microsoft'un DCOM (Dağıtık Bileşen Nesne Modeli) adındaki kendine özgü dağıtık nesne mimarisini geliştirmiş olmasıdır.

CORBA nesneleri bilgisayar ağı üzerinde herhangi bir yerde bulunabilmektedir. Uzak istemciler bu nesnelere metod çağrımları yolu ile erişebilmektedir. İstemcilerin, sunucu nesnelerin bilgisayar ağı üzerindeki yerini bilmesine gerek yoktur.

Sunucu nesnelerin hangi dille yazıldığı da istemciler açısından önemli değildir. Örneğin, bir sunucu nesne C++ sınıfları olarak veya uzun bir COBOL programı olarak gerçekleştirilmiş olabilir. İstemci açısından önemli olan sunucu nesnenin dışarıya sunduğu arayüzdür. Sunucu nesne arayüzünde, uzak istemcilere sunulabilecek servisler belirlenmektedir. Bu nedenle sunucu nesne arayüzü uzak istemciler ve sunucular arasında geçerli olan bir kontrata (anlaşma) benzetilebilmektedir. Bir nesnenin başka bir nesneden servis isteyebilmesi için, o nesnenin dışarıya sunduğu arayüzü bilmesi gerekmektedir.

IDL (Interface Definition Language-Arayüz Tanımlama Dili), CORBA nesnelerinin arayüzlerini tanımlamak için kullanılan dildir. IDL ile yazılan arayüzlerde bulunan metod prototiplerinin gerçekleştirimleri CORBA'nın desteklediği (CORBA ile bağlanabilen-binding) C, C++, Ada, Smalltalk, COBOL, Java dillerinde yapılabilmektedir. Diğer dillerin de desteklenmesi için çalışmalar sürmektedir. IDL, Şekil 1'den de görüldüğü gibi işletim sistemi ve programlama dilinden bağımsız arayüzler tanımlanmasını sağlayarak farklı dillerde yazılmış istemci ve sunucuların içten-işletilebilirliğine (interoperability) izin vermektedir.

IDL dili, C++ dilinin bir alt kümesi olarak tasarlanmıştır. Bu alt küme, kalıtım (inheritance), aykırı durum yönetimi (exception handling) gibi özellikleri içermektedir.

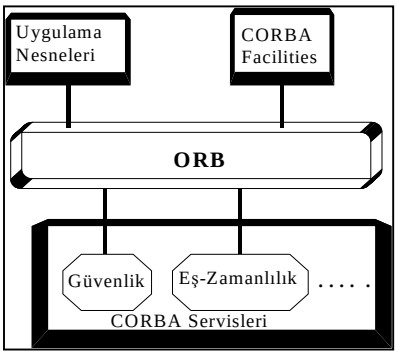


Şekil 1. Farklı dillerde yazılmış CORBA bileşenleri birbirinden servis alabilmektedir

2. 2. CORBA Mimarisi

CORBA mimarisi Şekil 2'den de görüldüğü gibi dört temel elemandan oluşmaktadır. Bu elemanlar şunlardır:

1. ORB (Object Request Broker).
2. CORBA servisleri (CORBA Services).
3. CORBA'nın uygulama seviyesinde hizmetleri (CORBA Facilities).
4. Uygulama nesneleri.



Şekil 2. CORBA Mimarisinin Temel Elemanları

ORB :

ORB; nesnelerin birbirlerinin yerini, gerçekleştirim dillerini, alt düzey iletişim mekanizmalarını bilmeden saydam (transparent) olarak birbirlerinden istekte bulunabilmelerini ve yanıtlar alabilmelerini sağlamaktadır.

CORBA ORB'nin sunduğu özelliklerden bazıları aşağıda listelenmiştir.

Statik ve dinamik metod çağrımı.

Arayüz ve nesne gerçekleştirimlerini birbirinden ayırması ve dilden bağımsız veri tipleri sunması nedeniyle farklı dillerde yazılmış veya farklı işletim sistemi platformlarında bulunan nesneleri çağırabilmek.

Tüm CORBA ORB'lerinde bir "Interface Repository" (arayüz deposu) bulunur. "Interface Repository", sunucuların dışarıya sunacağı metodlar ve parametrelerine ilişkin bilgileri tutar. İstemciler bu bilgileri işletim zamanında (run-time) metod çağırabilmek. için kullanmaktadır. Geç bağlama (late binding) esneklik kazandıran bir özelliktir.

ORB tek bir bilgisayarda tekil olarak işletilebileceği gibi, ağ üzerinde bulunan diğer ORB'ler ile de bağlanabilir. Bu bağlantı, CORBA 2.0 sürümünde gelen Internet Inter ORB Protocol (IIOP) ile sağlanmaktadır.

Gömülü (built-in) güvenlik mekanizmaları bulunmaktadır.

ORB, RPC'de olduğu gibi uzaktaki fonksiyonu doğrudan çağırmaz, karşı tarafta bulunan bir nesne üzerinde çağırır. Bunun anlamı, aynı fonksiyonun onu alan nesneye göre farklı işlemler yapabilmesidir. Örneğin, "Configure-yourself" isminde bir metod, uzaktaki veri tabanı nesnesi üzerinde çağrılırsa farklı, yazıcı nesnesi üzerinde çağrılırsa farklı işlemler yapacaktır. Bu özellik, çok-biçimli iletiler (polymorphic messages) olarak adlandırılmaktadır.

CORBA'nın bir nesnenin arayüzü ve gerçekleştirimini ayırması nedeni ile daha önceden varolan bilişim sistemlerinin CORBA ortamlarına tümleştirilmesi olasıdır. Bunun için varolan kodlara (örneğin CICS, COBOL, vb.) ilişkin IDL tanımları yazılarak bunların ORB üzerindeki nesneler olarak görülmesi sağlanmalıdır.

ORB, istemcilerden gelen çağrımları alır, bu çağrımı yerine getirebilecek bir nesne arar ve bulur, çağrımı bu nesneye iletir ve sonuçları döndürür. İstemci, servisi veren nesnenin nerede olduğu veya hangi dille yazıldığı gibi ayrıntıları bilmek zorunda değildir.

CORBA Servisleri :

CORBA servisleri, IDL ile yazılmış arayüzler biçiminde paketlenmiş sistem düzeyinde servisler topluluğudur. Bu servislerden bazıları aşağıda tanıtılmaktadır.

Kalıcılık (Persistance) servisi, CORBA bileşenlerini nesneye dayalı veri tabanları, ilişkisel veri tabanları veya dosya sistemleri gibi ortamlarda saklamak için bir arayüz sunmaktadır.

İsimlendirme (Naming) servisi, ağ üzerindeki CORBA bileşenlerinin diğerlerini isme göre bulabilmesini sağlamaktadır. Bu servis, ISO X500, OSF-DCE, Sun NIS+ gibi standartları desteklemektedir.

Eş-zaman kontrol (Concurrency Control) servisi, farklı hareketler (transactions) veya iş-parçaları (threads) üzerinde kilitler (lock) tanımlama olanağı veren kilit yöneticisini (lock manager) kullanıma sunmaktadır.

Sorgu (Query) servisi, nesneler için sorgu olanakları sağlamaktadır. SQL3 ve Object Database Management Group (ODMG) Object Query Language (OQL) standartlarına dayanmaktadır.

Lisans (Licence) servisi, CORBA bileşenlerinin kullanımlarını ölçmekte ve gerektiğinde ücretlendirmeyi belirlemektedir.

Özellikler (Properties) servisi, herhangi bir CORBA bileşeninin kullanıcı tarafından verilen bir özellik ile ilişkilendirilmesini sağlamaktadır. Örneğin, bir kullanıcı belirli bir CORBA bileşenini bir isim ve tarih değeri ile ilişkilendirebilmektedir.

Güvenlik (Security) servisi, nesne güveniği için bir çerçeve (framework) sunmaktadır. Yetkilendirme (authentication), erişim kontrol listeleri, gizlilik gibi konuları içermektedir.

CORBA Hizmetleri (Facilities) :

"CORBA Facilities", uygulama nesnelerinin doğrudan kullanabileceği ve IDL ile tanımlanmış olan uygulama düzeyi çerçeveler (frameworks) topluluğudur. Gezgin etmenler (mobile agents), güvenlik duvarları (firewalls), uluslar arası yapma (internationalization) "CORBA facilities" arasında sayılabilmektedir.

Uygulama Nesneleri :

Uygulama nesneleri kullanıcıların geliştirdiği uygulamaya ilişkin nesnelerdir.

2. 3. ORB Seçimi

Şu anda pazarda bir çok firma tarafından üretilen çok sayıda ORB bulunmaktadır. Bu ORB'lerden Java destekli olanların bazıları Tablo 1'de görülmektedir.

Tablo 1. Pazarda Bulunan Bazı Java ORB İsimleri ve Üretici Firmalar

Firma İsmi	ORB İsmi
IBM	Component Broker
Iona Technologies	Orbix
Inprise	Visibroker
Fujitsu	ObjectDirector
Netscape	Netscape Internet Service Broker
JavaSoft	JavaIDL
ObjectEra	JBroker
XeroxParc	ILU
Distributed Objects Group	JavaORB

Yukarıda listelenen ORB'ler içerisinde Iona'nın "OrbixWeb 3.0", Inprise'ın "Visibroker for Java 3.3" ve JavaSoft'un "JavaIDL" ürünleri yaygın kullanım alanları bulmuşlardır.

Java IDL, CORBA/Java ORB'dir ve JDK1.2. sürümü ile ücretsiz olarak gelmektedir. Java IDL, IIOP protokolünü destekler, bir isimlendirme servisi bulunur. IDL-To-Java derleyicisi ile IDL'ler Java "stub ve skeleton" kodlarına dönüşür. Java IDL'nin avantajları arasında, ücretsiz olarak İnternet'ten indirilme olanakları, geniş on-line belgelemesi ile CORBA 'yı yeni öğrenenler için başlangıç noktası oluşturma özelliği sayılabilir. Java IDL'nin ORB'si, ORB'lere ilişkin özelliklerden tümünü içermez, minimalist bir yaklaşımla tasarlanmıştır. CORBA Arayüz Deposu (Interface Repository) desteklemez bu nedenle dinamik çağrımlar yapılamaz. Diğer önemli bir eksikliği, isimlendirme servisi işletimi herhangi bir nedenle bittiğinde mevcut bilgiler kaybolmaktadır, kalıcılık özelliği (persistance) yoktur. Bunun dışında, güvenlik duvarları, dinamik sunucu aktivasyonu, iş dengeleme (load balancing) gibi ek özellikleri de desteklememektedir.

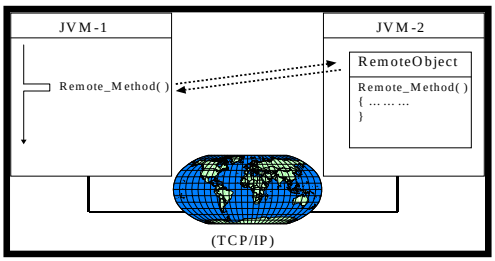
Iona, CORBA teknolojileri konusunda önderlik yapan kuruluşlar arasındadır. C++ Orbix ORB ürünü, Unix, OS/2, Windows 95, NT, Macintosh System 7.5, OpenVMS and MVS gibi 20'den fazla işletim sistemi tarafından desteklenmektedir. 1997 sonunda çıkan OrbixWeb 3.0 ürünü, Java dilini desteklemekle birlikte, JavaSoft Java IDL ürününde bulunmayan özellikleri de desteklemektedir.

Visibroker for Java 3.3 1997 yılı sonlarında çıkmıştır ve OSAgent adı verilen hata-toleranslı (fault-tolerant) bir isimlendirme servisi içermektedir. Statik ve dinamik çağrımlar, arayüz deposu, IIOP Gatekeeper güvenlik duvarı, otomatik iş-parçası yönetimi gibi JavaSoft Java IDL'de olmayan ileri özellikleri içermektedir.

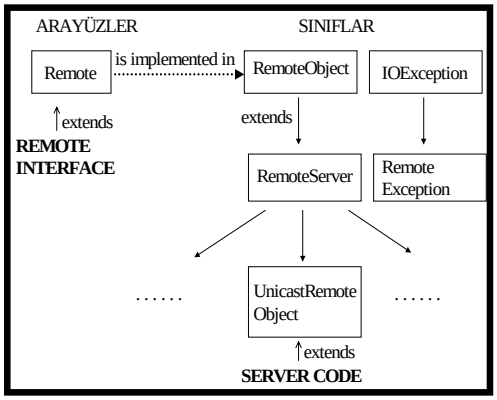
3. JAVA RMI

Java RMI, dünyanın farklı yerlerinde bulunan ve TCP/IP ile bağlantısı sağlanmış sanal Java makinaları üzerindeki Java nesnelerinin normal metod çağrımları kullanarak iletişimde bulunabilmesi için gerekli olan katmanları içermektedir. Şekil 3'te Java sanal makinası-1 (JVM-1) üzerinde bulunan bir Java uygulamasının (veya Java applet'inin) JVM-2 üzerinde bulunan RemoteObject isimli bir nesnenin Remote-Method ( ) isimli bir metodunu çağrımı görülmektedir.

RMI sisteminin kullandığı arayüzler ve sınıflar java.rmi paketinde bulunmaktadır. Bu arayüz ve sınıflar arasındaki ilişki Şekil 4'te görülmektedir.



Şekil 3. RMI ile uzak metod çağrımı



Şekil 4. RMI arayüz ve sınıfları

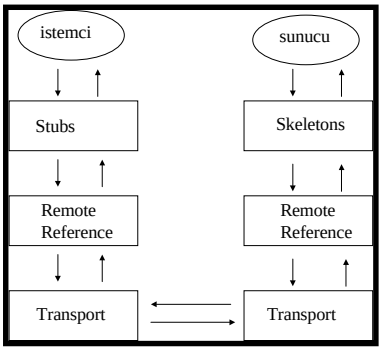
3. 1. Java RMI Mimarisi

Java RMI mimarisi üç katmandan oluşmaktadır.

Bunlar :

- Stub ve Skeleton Katmanı (Koçan ve iskelet katmanı)
- Remote Reference Katmanı (Uzak referans katmanı)
- Transport Katmanı (Taşıma katmanı)

Şekil 5'ten de görüldüğü gibi, uzakta bulunan bir Java nesnesinin metodlarını çağıran bir istemci ilk önce kendi tarafında bulunan "stub" kod ile iletişime geçer. "Stub" kod, isteği "remote reference" katmanına, bu katman da "transport" katmanına geçirir. Daha sonra, karşı taraftaki bilgisayarda istek önce "remote reference" katmanına oradan da "skeleton" koda iletilerek istenen servise ulaşılır. Servisin sonuçları da aynı biçimde istemciye iletilmektedir.



Şekil 5. Java RMI mimarisi katmanları

3. 1. 1. Stub ve Skeleton Katmanı

İstemci, uzakta bulunan bir nesnenin metodlarını çağırmaya başladığı anda ilk olarak "stub" kod ile iletişime geçecektir. İstemcinin uzaktaki nesne ile ilgili olarak elinde bulunan referans aslında yerel "stub" koda olan referanstır. Karşı tarafta bulunan "skeleton" kod ise uzak nesnenin istenen metodunun gerçekleştiriminin çağrılması ve sonuçların elde edilmesinden sorumludur.

Bir RMI çağrımında gerçekleşen olayların adımları ve bu adımlar içerisinde stub ve skeleton kodların görevleri aşağıda verilmiştir.

a) Stub kodu, object serialization (nesnelerin byte stream'leri biçimine dönüştürülüp aktarılması) tekniği ile çağrımı parametreleri ile birlikte paketler.
b) Stub kodu, isteğin uzak nesneye gönderimini yapar. Bu istek "remote reference" ve "transport" katmanları yolu ile karşı tarafa aktarılır. Karşı taraftaki "transport" ve "remote reference" katmanları ile de skeleton koda ulaşır.
c) Karşı taraftaki skeleton kodu gelen paketi açar, parametreleri çıkarır ve uzak nesnenin istenen metodunun gerçekleştirimini çağırır.
d) Skeleton kod sunucudaki nesneden gerçekleştirimin sonucunu alır.
e) Skeleton kod sonucu paketleyerek, isteğin geldiği yoldan benzer biçimde karşı tarafa gönderir.
f) Sonuçlar, stub koda gelir. Stub kodu, sonuçların paketini açar ve sonuçları istemciye geçirir.

"Stub ve Skeleton" katmanının diğer bir özelliği de, yazılan programların platform bağımsız olmasını sağlamasıdır. Bu katman kullanıcıyı uzak nesnenin gerçekleştirim ayrıntılarını veya "transport" katmanı ayrıntılarını bilme zorunluluğunu ortadan kaldıran bir soyutlama sağlamaktadır.

3. 1. 2. Uzak Referans (Remote Reference) Katmanı

Bu katman temel olarak programcının yazdığı kodlar ve bilgisayar ağı iletişimi arasında bir köprü görevi yapmaktadır.

Bu katmanın görevlerinden birisi stub kodundan gelen çağrımları "transport" katmanı biçemlerine dönüştürmek veya "transport" katmanından gelen istekleri skeleton kodunun anlayacağı biçime dönüştürmektir.

Bu katmanın diğer bir görevi uzak referans protokollerini (remote reference protocols) gerçekleştirmektir. Bu protokoller, uçtan uca (point to point) veya tekrarlı (replicated) nesnelere çağrım protokolleri olabilir. Tekrarlı bir nesne aynı anda birden fazla "instance"ı işletimde olan nesnelerdir. Örneğin, bir Java uygulaması farklı iş-parçaları (thread) içinde Java String sınıfının birden fazla "instance"ını işletiyor olabilir. RMI sistemi, tekrarlı bir nesne çağrıldığında, tüm "instance"ların aynı iletiyi algılamalarını sağlamaktadır. Multi-cast protokol olarak da adlandırılan tekrarlı nesnelere çağrım protokolü JDK1.1 sürümünde desteklenmemektedir. JDK1.1 sürümünde, sunucu sınıf, java.rmi.server.UnicastRemoteObject sınıfından türetilerek uçtan uca protokolün gerçekleştirimi sağlanmaktadır.

3. 1. 3. Taşıma (Transport) Katmanı

Bu katman bağlantının sağlanması, bakımı, kapatılması gibi işlerden sorumludur. İletişimin sağlanmasında Java soketleri kullanılmaktadır. JDK1.1 sürümünde, aktarım protokolü olarak TCP (Transmission Control Protocol) kullanılmaktadır. RMI mimarisinde her katman birbirinden bağımsız olduğu için aktarım protokolü değiştirmek, örneğin

UDP (User Datagram Protocol) protokolünü programcıların kullanımına sunmak olasıdır.

3. 2. Java RMI İsimlendirme Servisi

İstemcilerin uzaktaki nesnelerin metodlarını RMI ile çağrılabilmeleri için, sözkonusu uzak nesnelere ilişkin referansları elde etmesi gerekmektedir. RMI modelinde bu amaca yönelik olarak bir isim sunucu (name server) kullanılmaktadır. Uzak nesneler java.rmi.Naming isimli bir sınıfta bulunan bind ( ) metodunu kullanarak kendilerini isim kayıtları (Naming Registry) servisine kaydettirmelidirler.

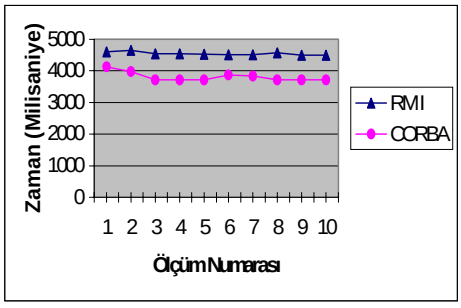
4. CORBA İLE JAVA RMI KARŞILAŞTIRMASI

Bu bölümde, Java RMI ile CORBA, performans, dağıtık programlama ve sunulan servisler açısından karşılaştırılacaktır.

4. 1. Performans

Performans karşılaştırılması için bir dosyanın içeriğini istemciye aktaran bir istemci/sunucu uygulama hem CORBA hem de Java RMI ile gerçekleştirilmiştir. Bunun için dosya sunucu CORBA ve Java RMI nesneleri ile istemci Java sınıfları kullanılmıştır. RMI uygulaması için "JDK" aracı, CORBA uygulaması için "Visibroker for Java 3.3" kullanılmıştır. "Visibroker for Java 3.3" barındırdığı ileri özellikleri ve yeterli dokümantasyonundan dolayı kullanım için seçilmiştir.

Performans ölçümü için yaklaşık 49000KB uzunluğunda bir dosyanın transferi için on kez zaman ölçümü yapılmıştır. Elde edilen sonuçlar Şekil 6'da görülmektedir.



Şekil 6. CORBA ve Java RMI ile dosya sunucudan içerik aktarım zamanları karşılaştırılması

Yukarıdaki grafik incelendiğinde tipik bir dosya transferi işlemi için CORBA'nın ortalaması 3808.2 milisaniye iken, RMI'ın ortalamasının 4544.6 milisaniye olduğu görülmektedir. Aradaki yaklaşık 700 milisaniyelik fark çok büyük bir fark değildir. O nedenle, bir dağıtık sistem tasarımında CORBA veya Java RMI arasında seçim yapmak için diğer kriterlerin incelenmesi daha fazla önem arzetmektedir.

4. 2. Dağıtık Programlama

Java RMI, ağ işlem (networking) için kullanıcıya soket veya "stream" lere göre daha üst düzeyde bir arayüz sunmaktadır. Bu nedenle, RMI ile dağıtık programlama yapmak soket ve "stream" kullanımına göre daha az karmaşıktır. Programcı açısından bakıldığında, RMI kullanıldığında istemci/sunucu uygulamaların geliştirilmesi sırasında ağ işlem (networking) ayrıntıları ile uğraşmak gerekmemektedir. CORBA da aynı şekilde soketlere göre daha üst düzeyde bir arayüz sunmaktadır.

Java RMI kullanıldığında istemci ve sunucu da Java ile yazılacağı için dağıtık programlama tümüyle Java dilinde gerçekleşmektedir, böylelikle Java dilinin özellikleri ve avantajları dağıtık programlamada da kullanılabilmiş olmaktadır.

Dağıtık nesne işleme dayanan sistemlerde bulunan sunucu nesneler servislerini bir arayüz ile istemcilere tanıtmaktadır. Java RMI kullanıldığında bu arayüzler Java dili arayüzleri (Java interfaces) olarak yazılabilmektedir. Böylece, Java RMI kullanıldığında sunucular, istemciler ve arayüzler Java dili ile yazılmaktadır. Bu durum dağıtık programlamada saf Java yaklaşımı olarak da adlandırılmaktadır.

CORBA kullanıldığında sunucu nesnelere ilişkin arayüzler tamamen ayrı bir arayüz tanımlama dili (interface definition language– idl) ile yazılmaktadır. CORBA ile dağıtık sistem geliştirmek arayüzler IDL dili ile yazılmak zorundadır. Bu da geliştiriciler için fazladan bir eğitim gerektirebilmektedir.

CORBA'nın, Java RMI'a karşı en büyük üstünlüğü farklı programlama dillerinde yazılmış istemci ve sunucuların bağlantısını sağlamasıdır. Örneğin, Smalltalk dilinde gerçekleştirilmiş bir istemci, C++ dilinde gerçekleştirilmiş olan bir sunucu nesneden servis alabilmektedir. Bu özellik, IDL gibi ayrı bir arayüz tanımlama dili olması nedeni ile sağlanmaktadır. IDL ile yazılmış tanımlar IDLtoSmalltalk, IDLtoC++, vb. derleyicilerden geçirilerek istemci "stub" ve sunucu "skeleton" oluşturulmaktadır. Bu "stub" ve "skeleton" kodlar daha sonra istemci ve sunucu arasındaki bağlantıyı sağlamaktadır.
Sonuç olarak, geliştirilecek olan dağıtık sistemdeki tüm istemci ve sunucular Java gibi platform

bağımsız tek bir dil ile gerçekleştirilecek ise saf Java yaklaşımı ve dolayısı ile Java RMI kullanımı gerçekleştirimi kolaylaştıracaktır. Eğer, Java desteği olmayan platformlarda bulunan veya farklı dillerdeki gerçekleştirimler arasında bağlantı kurulacak ise CORBA kullanımı uygun olacaktır. Bu durumda fazladan IDL öğrenilmesi gerekecektir fakat getirdiği esneklik IDL öğrenilmesi için kaybedilmesi olası olan zamanı karşılayacaktır.

4. 3. Servisler

CORBA, bölüm 2. 2.'de sözedildiği gibi kalıcılık, isimlendirme, lisans gibi sistem seviyesi servisleri sunmaktadır. Java RMI bu tür servisler açısından zengin değildir.

Kalıcılık servisi ile nesnelerin belli bir formatta veri tabanı veya kütük sistemlerinde saklanması sağlanmaktadır. Java RMI ile geliştirilen sistemlerde nesnelerin saklanması programcının sorumluluğundadır ve gerekli modüllerin tüm ayrıntıları ile yazılması gerekmektedir.

CORBA isimlendirme servisi ISO X500, OSF-DCE, Sun NIS+ gibi standartları destekleyen gelişmiş bir servistir. Java RMI'da bir isimlendirme servisi (registry) bulunmaktadır fakat CORBA isimlendirme servisi kadar gelişmiş değildir.

CORBA'nın sunduğu diğer önemli bir servis lisans servisidir. Günümüzde, İnternet ve ağ teknolojilerindeki gelişmeler dağıtık sistem yazılımı geliştirme yaklaşımlarını da etkilemiştir. Dağıtık sistem geliştiricileri, bazı ilgili bileşenleri yeniden kullanım yolu ile İnternet üzerinden elde ederek sistemlerine entegre edebilmektedir. Bu yöntemle yazılım geliştiren kuruluşlar daha kolay, hızlı, kaliteli ve güvenilir yazılımlar geliştirebilecektir. Örneğin, herhangi bir şirket e-ticaret uygulaması geliştirmek istediğinde, kredi kartı işlemlerini yürütmek üzere bir bankanın sunduğu CORBA sunucusuna bağlantı sağlamak isteyebilecektir. Sözkonusu bankaya birçok e-ticaret uygulaması geliştiren şirketin bağlandığını düşünürsek, bankanın bu servisten bir gelir beklemesi doğal bir yaklaşım olacaktır. CORBA bileşenlerinin kullanımlarını ölçen ve gerektiğinde ücretlendiren lisans servisi bu gibi durumlarda gereksinim duyulan bir servis olacaktır.

CORBA, yukarıda sözedilenler dışında sorgu, eşzaman kontrol, güvenlik gibi servisler de sunmaktadır. Bu servisler de Java RMI'a göre daha gelişmiş servislerdir.

5. SONUÇLAR VE TARTIŞMA

Bu çalışmada, dağıtık sistemlerin nesneye dayalı olarak modellenmesinde kullanılan CORBA ve Java RMI standartları kısaca tanıtılarak çeşitli açılardan karşılaştırılmıştır.

Değişik açılardan yapılan karşılaştırmalar sonucunda aşağıdaki sonuçlara varılmıştır:

- CORBA ve Java RMI tabanlı sistemler arasında belirgin bir performans farkı yoktur.
- Çok karmaşık olmayan ve Java destekli platformlardan oluşan dağıtık sistemler Java RMI kullanılarak gerçekleştirilebilir.
- Farklı dillerde gerçekleştirilmiş olan ve farklı işletim platformlarında bulunan istemci ve sunucuların bağlantısının sağlanmak istendiği karmaşık sistemlerde CORBA kullanılmalıdır.
- Geliştirilen dağıtık sistemde kalıcılık, servis kullanımlarının lisansı ve ücretlendirmesi gibi konular önemli ise bu tür servisleri sunan CORBA kullanılmalıdır.

6. KAYNAKLAR

Anonymous, 1997. Microsoft Visual Basic Component Tools Guide. Microsoft Corporation, 796p.

Anonymous, 1998. Sun Microsystems. "Java Development Kit Documentation JDK1.2". Available at http://www.javasoft.com.

Orfali, R., Harkey, D. and Edwards, J. 1997. Instant CORBA, John Wiley and Sons.

Orfali, R. and Harkey, D. 1998. Client/Server Programming With Java and CORBA, John Wiley and Sons.

Pedrick, D., Weedon, J., Goldberg, J. and Bleifield E. 1998. Programming with Visibroker, Wiley Computer Publishing.

Saleh, K., Probert, R. and Khanafer, H. 1999. "The Distributed Object Computing Paradigm: Concepts and Applications", The Journal of Systems and Software, 47 (1999) 125-131, Elsevier Science Inc.