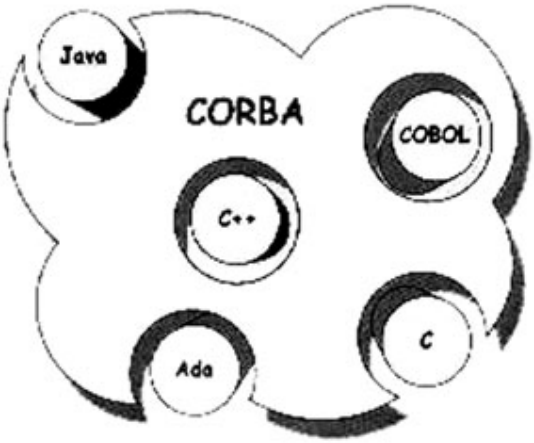
CORBA ؛ استانداردی برای سیستم‌های توزیع‌یافته

امین صفائی

ماهنامه شبکه - آبان ۱۳۸۴ شماره 59



اشاره :

CORBA سرنام واژه‌های Common Object Request Broker Architecture استانداردی برای نرم‌افزارهای ترکیبی (componentry) است که توسط شرکت object OMG (managementgroup)طراحی شده و پشتیبانی می‌گردد. این استاندارد API، پروتکل‌های رابطه‌ای ومدل‌های اطلاعاتی object/service را تعریف می‌کند که می‌تواند نرم‌افزارهای ناهمگون (نوشته شده به زبان‌های متفاوت) را به هم مرتبط‌سازد. بنابراین با استفاده از CORBA می‌توانیم ازشی (object) در پلتفرم‌های توزیع‌یافته (distributed)، به صورت مشترک استفاده کنیم؛ بدون نگرانی از این که شی در چه موقعیت مکانی قرارگرفته یا به چه پلتفرمی متعلق است. CORBA می‌تواند کدهای نوشته شده (در برخی زبان‌های برنامه‌نویسی) را بسته بندی و به آن اطلاعاتی از قبیل توانایی اجرایی کدها و چگونگی اجرای آن‌ها را اضافه نماید. به‌طوری که این کدهای بسته بندی‌شده (یا شی‌ها) بتوانند از برنامه‌های دیگر (یا حتی شی‌های CORBA) که تحت شبکه قرار دارند، اجرا شود. CORBA برای تعیین‌کردن اینترفیس‌هایی که به دیگران ارائه می‌کند، از یک زبان رابطه‌ای به نام IDL استفاده می‌کند و از طریق این زبان می‌تواند کدهای اجرایی (مثلا جاوا یا C++) را بشناسد. امروزه می‌توان زبان‌های معروفی مثل جاوا، Smalltalk، C++،Python، آدا، سی و لیسپ را با استفاده از این زبان (IDL) به هم مرتبط ساخت و از توانایی‌های آن‌ها استفاده نمود. قسمت اول این مقاله در آغاز نرم‌افزارهای توزیع یافته را معرفی کرده و دلایل نیاز به آن را مشخص می‌نماید. سپس این نرم‌افزارها را با سیستم‌های متمرکز یا توزیع یافته مقایسه می‌کند و در ادامه با طرح این سؤال که چرا به CORBA نیاز داریم، CORBA را معرفی و ساختار آن را تشریح می‌کند و بعد چگونگی کار این استاندارد را توضیح می‌دهد. درپایان نیز مزایا و ضعف‌های آن را به اختصار بیان می‌نماید.

نرم افزارهای توزیع‌یافته (Distributed)

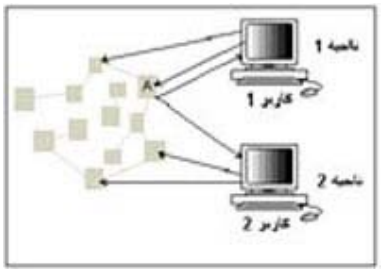
همان‌طور که اشاره شد، کار اصلی CORBA ارائه استانداردی برای ایجاد و اجرای نرم افزارهای توزیع‌یافته می‌باشد. اگرچه توزیع‌یافته بودن نرم‌افزارها می تواند مشکلاتی برای تولید کنندگان و کاربران آن داشته باشد، ولی به‌دلیل غیرمتمرکز بودن اطلاعات نرم‌افزارها، پراکندگی کاربران، و نیاز به اجرای برخی برنامه‌ها با استفاده

از بیش از یک پردازنده ، راه دیگری جز استفاده از این نرم‌افزارها نیست.

برخی از نرم‌افزارها به دلیل این که کاربران آن باید از طریق این نرم افزارها با هم در ارتباط باشند، بر روی چند کامپیوتر اجرا می شوند. بدین ترتیب که هر کاربر قسمتی از این نرم افزار را برروی کامپیوتر خود اجرا می‌کند و شی‌های مشترک بر روی یک چند سرور اجرا می شوند. شکل 1 نشان می دهد که دو کاربر از شی‌های به اشتراک گذاشته شده استفاده می کنند. همان‌طور که در این شکل مشخص شده‌است، دو کاربر یکی در ناحیه 1 و دیگری در ناحیه 2 بدون در نظر گرفتن موقعیت جغرافیایی، به شی‌های گوناگونی دسترسی‌دارند و در برخی موارد، بر حسب نیاز به یک شی (object A) دسترسی خواهند داشت.

همان‌طور که قبلا اشاره شد، یکی دیگر از دلایل نیاز به نرم‌افزارهای توزیع‌یافته، پراکندگی اطلاعات می‌باشد. برخی از نرم‌افزارها، از آنجایی که به اطلاعاتی نیاز دارند که برروی چند کامپیوتر است (به‌دلایل امنیتی و حفاظت اطلاعات شخصی) و صاحب اطلاعات فقط اجازه دسترسی از راه دور را به کاربران می‌دهد، باید درچند کامپیوتر اجرا شوند. از طرف دیگر، نیاز به نرم‌افزارهای توزیع‌یافته در سیستم‌هایی که می‌خواهند از مزایای استفاده از چند پردازنده (به‌صورت موازی) استفاده کنند، محسوس به نظر می رسد.

تا این جا دلایل نیاز دنیای امروز به نرم افزارهای غیرمتمرکز توضیح داده شد. ممکن است از خود سؤال کنید چه فرقی بین سیستم‌های متمرکز و توزیع‌یافته وجوددارد؟ جدول1 برخی از این تفاوت‌ها را نشان می‌دهد.



شکل 1- استفاده مشترک از شی‌ها

)برای مشاهده تصاویر مقاله در ابعاد بزرگتر روی آنها کلیک کنید(

دلایل نیاز به CORBA

دلایل نیاز به CORBA را می‌توان در پنج‌مورد زیر فهرست کرد:

1- گوناگونی سیستم‌عامل‌ها

2- تنوع پلتفرم‌های سخت افزاری

3- گوناگونی پروتکل‌های شبکه (برای مثال ATM،TCP/IP و Ethernet)

4- گوناگونی زبان‌های برنامه‌نویسی

(Perl ,Smalltalk, C ,C++ ,Java ,COBOL ,Basic)

5- نیاز ارتباط یافتن سیستم‌های جدید با سیستم‌های قدیمی

امنیت	سیستم	سیستم
درویژگی‌های توزیع‌یافته‌ای که object های درسیستم‌های مختلفی قرار دارند، توزیع‌یافته بیشتری نسبت به سیستم های محلی نیاز است .	متمرکز	توزیع‌یافته

با استفاده از CORBA، دیگر دلیلی برای نگرانی در مورد این‌که از چه زبان برنامه‌نویسی، سیستم‌عامل یا پلتفرمی استفاده می‌کنید، وجود ندارد. از آن‌جایی که CORBA با مکانیزم ساده‌ای می‌تواند سیستم‌های مختلف از هر نوع و هراندازه را به هم متصل کند، درخیلی از موقعیت‌ها بسیار مفید به نظر می‌رسد. یکی از مصارف CORBA در سرورهایی است که با تعداد زیادی کلاینت و ترافیک سنگین اطلاعاتی باید به خوبی کارکنند. استفاده از CORBA فقط محدود به نرم‌افزارهای بزرگ نمی‌شود و حتی درسیستم‌های بلادرنگ نیز کاربرد دارد.

CORBA و ساختار آن

گروه OMG در سال 1989 با هدف ایجاد و پشتیبانی استاندارد برنامه‌های شی‌گرای غیر متمرکز به‌وجود آمد. البته، این گروه در واقع تولید کننده هیچ نرم‌افزاری نیست، بلکه مشخصات (Specification) این برنامه‌ها را با استفاده از فناوری و نظرات اعضا مشخص می‌کند.

این گروه شامل بیش از 700 شرکت و سازمان تولیدکننده فناوری سیستم‌های توزیع‌یافته، بانک‌های اطلاعاتی، پلتفرم‌ها و سازندگان بزرگ نرم‌افزار می‌باشد. تلاش اصلی این گروه در راستای تعریف امکانات و ساختار لازم برای سیستم‌های شی‌گرای غیرمتمرکزاست و یکی ازمکانیزم‌های اصلی در این رابطه که هسته مرکزی این گروه نیز می‌باشد، Object Request Broker (ORB) نام دارد. سال 1991 را می‌توان سال تولد اولین نسخه CORBA (Request Broker Architecture Common Object) توسط این گروه نامید.

مشکلات و ازکارافتادگی شی‌ها	همه شی ها با هم خراب می‌شوند	شی‌ها جداگانه خراب می‌شوند
اگر پردازنده‌ای که دو object را اجرا می‌کند دچار مشکل شود، هر دو object دچار مشکل خواهند شد.		
دسترسی همزمان	فقط با استفاده از با چند Thread	دارد
برنامه‌ها بر روی سیستم‌های متمرکز، به‌صورت پیش فرض از یک thread استفاده می‌کنند. ولی در سیستم‌های توزیع‌یافته به‌صورت پیش فرض از thread استفاده می‌شود.		
ارتباط بین سیستم ها	سریع	آهسته
از آن‌جایی که ارتباط بین object ها دریک پردازنده سریع تر از ارتباط بین object ها درچند پردازنده می باشد، پیشنهاد می‌شود اگر ارتباط تنگاتنگی بین شی‌ها وجود دارد، از سیستم‌های متمرکز استفاده شود .		

جدول 1- جدول مقایسه‌ای سیستم‌های متمرکز و توزیع‌یافته

CORBA ساختار استانداردی برای سیستم‌های توزیع‌یافته شی‌گراست که می‌تواند سیستم‌های ناهمگون و پراکنده را به هم مرتبط سازد. همان‌طور که گفته شد، CORBA ساختاری برای objectهای توزیع‌یافته مشخص می‌کند. محوراصلی این ساختار بر اساس درخواست سرویس از شی‌های توزیع‌یافته می‌باشد. این شی‌ها سرویس‌ها را از طریق اینترفیس‌هایشان که به زبان Interface Definition Language (IDL)

مشخص شده‌اند، ارائه می‌دهند. Objectهای غیر متمرکز را می توان از object referencesها که توسط اینترفیس‌های IDL تایپ شده اند، شناسایی کرد.

همان‌طورکه در شکل 2 می‌بینید، درخواست‌کننده سرویس (کلاینت) شماره مرجع object شی سرویس‌دهنده را از طریق اینترفیس آن در اختیاردارد (Interface A) و ORB درخواست کلاینت را به شی سرویس‌دهنده و جواب درخواست را به شی درخواست کننده می رساند.

ORB درواقع یک سرویس توزیع‌یافته است که به درخواست شی‌های دوردست (remote object) رسیدگی می‌کند. بدین ترتیب که این شی‌ها را در شبکه قرار می دهد، با شی‌ها ارتباط برقرار کرده، و درخواست سرویس را مطرح می‌کند. سپس برای جواب سرویس درخواستی صبر می‌کند و سرویس موردنظر را به کلاینت درخواست کننده منتقل می‌کند. ORB این کار را بدون توجه به موقعیت مکانی سرویس‌دهنده و زبان برنامه درخواست‌کننده انجام می‌دهد. نیازی نیست که کلاینت درخواست کننده به زبان CORBA تقاضای خود را مطرح کند. ORB زبان برنامه درخواست‌کننده را شناسایی کرده (برای اکثر زبان‌های برنامه نویسی) و این زبان را ترجمه می کند.

کارکرد CORBA

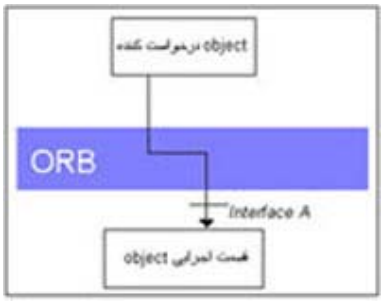
نرم‌افزارهایی که از CORBA استفاده می کنند، حاوی شی‌هایی‌هستند. البته معمولا از یک شی چندین نمونه(Instances) از یک نوع وجود دارد. مثلا در یک سایت تجارت الکترونیک، نمونه سبدهای خرید زیادی وجود دارد که با وجودی که همه آن‌ها کاری یکسان انجام می‌دهند، هر کدام به مشتریان متفاوتی تعلق دارد و اطلاعات خرید آن مشتری را ذخیره می کنند.

برای هر نوع از شی‌ها، مثلا سبد خرید می‌باید یک اینترفیس در IDL OMG تعریف کنیم که نشانگر سرویس‌هایی است که شی سرور در اختیار کلاینت‌های تقاضاکننده قرارمی‌دهد. هر کلاینت که تقاضای اجرای عملیاتی از شی را دارد، باید از این اینترفیس‌ها استفاده و آرگومان ها را کنار هم چیده و ارسال‌نماید. وقتی که این تقاضا به شی سرویس‌دهنده می‌رسد، این شی با استفاده ازاینترفیس مشابه آن، عملیات درخواستی کلاینت را انجام می دهد و آرگومان ها را جدا سازی می کند. پس از انجام عملیات درخواستی، جواب درخواست، بسته بندی شده و با استفاده از تعریف اینترفیس از همان مسیری که آمده بود، بر می گردد. چنان که گفته شد، اینترفیس IDL برای اکثر زبان‌های برنامه‌نویسی از جمله Smalltalk , IDLscript , Python, Lisp, Ada , COBOL, java, C++, C و از طریق استاندارد OMG تعریف شده است.

جداسازی اینترفیس و قسمت اجرایی که توسط OMG IDL مهیا شده است، یکی از مهم‌ترین پایه‌های اصولی CORBA می باشد و ایجاد اینترفیس برای هر شی در CORBA ضروری است. ازطرف دیگر قسمت اجرایی هر شی از دید سیستم مخفی می‌ماند و کلاینت از آن چیزی نمی‌داند.

در نتیجه کلاینت‌ها فقط می‌توانند از طریق اینترفیس‌ها به سرویس‌ها دسترسی داشته و تنها سرویسی را اجرا کنند که توسط اینترفیس‌های IDL معرفی شده اند.

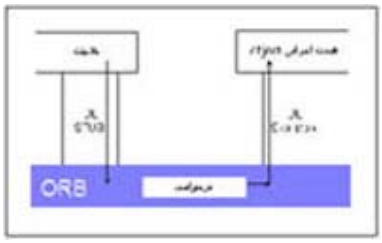
همان‌طور که در شکل 3 نشان داده شده‌است، یک کلاینت توسط اینترفیس IDL تقاضای سرویس می‌کند. برای این کار IDL باید به کلاینت stubs و skeletons object کامپایل شود. stubs و skeletonsها درواقع نقش نمایندگان کلاینت و سرور را بازی می‌کنند و از آن جایی که IDL تعریف مشخصی برای اینترفیس ها دارد، اگر حتی این دو نماینده به دو زبان متفاوت یا حتی برروی دو ORB از دو شرکت مختلف اجرا شوند، می‌توانند بدون مشکل به هم مرتبط شوند.



شکل 2- ساختار ساده CORBA

همان‌طور که اشاره شد، در CORBA هر نمونه ازاشیا دارای یک کد یکتا (unique) به نام object reference هستند. کلاینت‌ها از این اشیا برای هدایت درخواست‌هایشان و شناساندن خود به ORBها استفاده می‌کنند. اگرچه کلاینت‌ها درخواست‌هایشان را به نماینده سرور یا IDL stub منتقل می‌کنند نه به خود سرور، چنین به نظر می‌رسد که کلاینت مستقیما به سرور دسترسی دارد. این درخواست سپس از طریق ORB و skeleton به قسمت اجرایی می‌رسد و این قسمت به این درخواست رسیدگی می‌کند. شکل 4 مکانیزم درخواست سرویس ازراه دور را نشان می دهد.

برای ایجاد ارتباط و درخواست از شی دوردست، در اولین قدم، کلاینت object reference شی موردنظر را از راه‌های گوناگونی مثل Naming Service به‌دست‌میآورد و با استفاده از مکانیزم درخواست محلی، درخواست خود را مطرح می کند. وقتی که ORB متوجه می‌شود object reference موجود، مخصوص یک شی دوردست است، مسیرخود را عوض کرده و از طریق شبکه (پروتکل IIOP) به‌دنبال ORB شی دوردست می‌گردد و درخواست خود را مطرح می کند.



شکل 3- درخواست از کلاینت به object implementationمنتقل می‌شود.

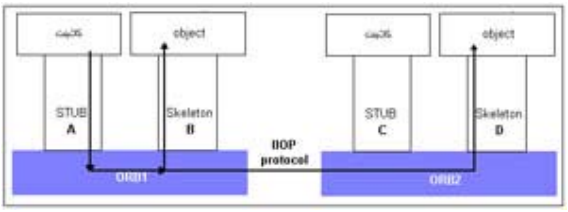
چگونگی این مکانیزم از این قرار است که گروه OMG استانداردی دو مرحله‌ای برای این رویه دارد. در مرحله اول کلاینت از نوع شی‌ای که درخواست می کند، اطلاع دارد (مثلا سبد خرید) و stub کلاینت (A) و skeleton شی توسط IDL یکسان تولید شده است. این بدین معنی است که کلاینت دقیقا می‌داند که چه عملیاتی (وبا چه پارامترهای ورودی) می‌تواند درخواست کند و چه زمانی درخواست به شی مقصد می‌رسد. در

مرحله دوم باید

ORBهای کلاینت و شی مورد نظر، هر دو از یک پروتکل یکسان تبعیت کنند. این پروتکل حاوی تمامی اطلاعات مورد نیاز (ازجمله تمامی پارامترهای ورودی/خروجی) برای دسترسی به شی مقصد می‌باشد. اگرچه ORBها می‌توانند شی‌های محلی را از دوردست تشخیص دهند، از آن جایی که کلاینت حاوی هیچ objec treferenceای در موقع درخواست نیست، قادر به تمایز شی‌های دوردست و محلی نمی‌باشد و همان‌طور که قبلا ذکر شد، این از اصول اصلی CORBA می‌باشد (بدون اهمیت بودن موقعیت مکانی Object).

مزایا و معایب CORBA

با توجه به آنچه گذشت، اکنون‌به‌نظر می‌رسد بحث درباره مزایا و معایب این استاندارد، ضروری باشد. از این‌رو به پاره‌ای از مزایا و معایب CORBA اشاره می‌شود:



شکل 4- ارتباط ORB به) ORB مکانیزم درخواست سرویس از راه دور(

- کلاینت نیازی به دانستن موقعیت مکانی شی ندارد. یک شی می‌تواند هر جا باشد، به کلاینت متصل باشد یا برروی سروری در آن سوی کره زمین، هیچ فرقی نمی‌کند.

- موقعیت مکانی شی می‌تواند بدون بروز اشکال در نرم افزار تغییر یابد.

- کلاینت نیازی به دانستن این‌که سروری برای جواب دادن به درخواست وجود دارد یا نه، ندارد.

- کلاینت و سرور می‌توانند به دو زبان کاملا متفاوت نوشته شوند و این مزیت مهم CORBA اجازه استفاده ازقدرت زبان‌های مختلف را ازطریق (IDL) برای تهیه نرم‌افزارهای بزرگ می‌دهد.

- کلاینت نمی‌داند یک شی چگونه عمل می‌کند. به همین خاطر یک سرور می‌تواند بدون این که کلاینت اطلاع پیدا کند یک بار از فایل ساده و بار دیگر از پایگاه اطلاعاتی شی‌گرا برای ذخیره اطلاعات استفاده نماید.

- سرعت و کارایی سیستم‌هایی که از CORBA استفاده می کنند بسیار بالا است.
- سیستم‌های عامل و پروتکل سرور و کلاینت می تواند متفاوت باشد.

اگرچه همان‌طورکه می‌بینید استفاده از CORBA می‌تواند مزایای زیادی داشته باشد، اما دارای نقاط ضعف زیر نیز هست:

- از انتقال و جابه‌جایی شی (objectها) پشتیبانی نمی کند.

- CORBA بهIDL هایی نیاز دارد که هنوز برای برخی از زبان‌ها تعریف نشده است و وقت زیادی برای یادگیری آن‌ها نیاز است.

- اگر CORBA نتواند نیازهای صنایع امروز را برآورده سازد و رواج نیابد، آینده مبهمی در انتظار خواهدداشت و ممکن است به جمع سیستم‌های قدیمی بپیوندد.

- از آنجایی که خصوصیات CORBA هر چند وقت یک‌بار عوض می‌شود، نیاز به آموزش و بروزآوری بیشتری دارد.
- از آن جایی که شعار CORBA، سرعت و کارایی بیشتر نسبت به سیستم‌های توزیع‌یافته جاوا مثل RMI است و همه نرم‌افزارهای کاربردی نیازی به سرعت بالای آن ندارند، شاید بتوان گفت اگر بخواهیم فقط با جاوا کار کنیم آسانی و کاربرد سیستم‌هایی مثل RMI از CORBA بیشتر است.

جمع‌بندی

البته نمی‌توان ادعا کرد با این مطالب همه مسائل مربوط به CORBA شرح داده‌شد. هرچند سعی شد بسیاری از مطالب مهم مربوط به آن در حد امکان توضیح داده شود. بااین حال پاره‌ای ازاین موارد دربخش دوم این مقاله خواهد آمد که به معرفی سرویس‌ها و محصولات CORBA می‌پردازد و با طرح مثال‌های ساده و مقایسه این فناوری با RMI وCOM، این بحث را پیگیری می‌کند.

منابع :

1- Orfali Harkey-Client/Server Programming with Java and CORBA

2- Vogel, Andreas-Java Programming with CORBA

3- http://www.omg.org

4- Reilly, D-Java RMI CORBA, A comparison of two competing technologies