Club de Investigación Tecnológica

Estándares de Sistemas Abiertos de Software

Ing. José Ardón Castro

Abril 1995

Resumen Ejecutivo

El propósito de este trabajo es presentar información relevante sobre el estado de los sistemas abiertos de software en la industria de la tecnología de información, para asistir al lector en la toma de decisiones relacionadas con el diseño de sistemas y la adquisición de componentes de software de sistemas abiertos.

Los objetivos de la investigación son: presentar las principales piezas de software de los sistemas modernos como componentes de sistemas abiertos, caracterizar los sistemas abiertos de software, y mencionar algunos casos de la situación actual del mercado. Un cuarto objetivo del trabajo se enfoca hacia una discusión detallada de los principales estándares que aplican a los sistemas operativos, al "middleware" de cliente servidor y a las bases de datos, como principales componentes de software de sistemas abiertos. La organización del trabajo responde a los cuatro objetivos anteriores.

El informe consta de seis capítulos. El primer capítulo, Introducción, es un resumen del concepto de sistemas abiertos, que sirve como justificación de la tendencia y explica el impacto que ha causado en la industria, conluyendo que los sistemas abiertos son y seguirán siendo la dirección prevaleciente en la industria.

El segundo capítulo, Generalidades de Sistemas Abiertos, caracteriza los sistemas abiertos presentando la historia general de esta tendencia, las promesas para la industria, el impacto en los usuarios y en los proveedores, y concluye con recomendaciones de pasos básicos para hacer uso de esta tendencia. El tercer capítulo, Conceptos de Sistemas Abiertos, presenta los conceptos básicos que deben ser entendidos antes de discutir los estándares de la industria. Se presentan aspectos como compatibilidad, especificaciones de interfaces, estándares, pruebas de cumplimiento, y acceso a datos con SQL.

El cuarto capítulo, Estándares de Sistemas Operativos, presenta en detalle las principales organizaciones y grupos de estándares que se aplican a los sistemas operativos, como principal componente de software de los sistemas abiertos. El capítulo concluye con una tabla que resume la posición de los principales proveedores de sistemas abiertos de software. El quinto capítulo, Cliente-Servidor y Sistemas Abiertos, discute en detalle el modelo de aplicaciones cliente-servidor, el middleware y los "APIs", incluyendo ejemplos de productos y de estándares aplicables. El sexto capítulo, Conectividad de Bases de Datos, esboza los principales estándares y productos relacionados con las bases de datos abiertas y su uso en sistemas abiertos de software.

Conclusión general

El software de los sistemas de cómputo actuales es efectivamente una colección de componentes individuales que, gracias a la tendencia de sistemas abiertos, se pueden adquirir de distintos proveedores, según las necesidades del cliente. La clave para que estos componentes operen en forma conjunta y armoniosa está en su cumplimiento con los estándares que publican las organizaciones independientes. Actualmente existen suficientes estándares para definir parámetros generales, pero quedan detalles pendientes. Los principales componentes de software de sistemas abiertos son: el sistema operativo, el middleware de aplicaciones cliente-servidor, y la base de datos.

Sobre el autor

José Ardón es Director Regional de Consultoría de Price Waterhouse Interamerica Consulting Group, grupo interdisciplinario de profesionales que brindan consultoría y asistencia técnica a empresas e instituciones públicas en Centro América. Este trabajo se preparó con material del Price Waterhouse Technology Forecast: 1995.

Contenido Página

Introducción

Los sistemas abiertos han cambiado totalmente las industrias de la computación y las comunicaciones de datos, moviendo la atención de los compradores hacia productos basados en estandares de múltiples fabricantes, en vez de productos basados en estandares propietarios de un solo proveedor.

Los sistemas abiertos son independientes del fabricante y están diseñados para interconectarse con una amplia variedad de productos de diferentes proveedores. Las raíces históricas de los sistemas abiertos vienen de esfuerzos que resultaron en el modelo de la Organización Inter-nacional de Estándares (ISO, del inglés "Inter-national Standards Organization") para las redes y para estandarizar el sistema operativo UNIX. Sin embargo, los conceptos de sistemas abiertos se han extendido a todos los rincones de las indus-trias de la computación y las comunicaciones.

Los usuarios están migrando desde los macrocomputadores y minicomputadores propietarios hacia los sistemas abiertos a paso relativamente importante. En parte, esta migración se debe a las mejores condiciones de precio-rendimiento que se encuentran en los sistemas abiertos, pero a menudo esta ventaja se ve compensada por los altos costos de migrar las aplicaciones existentes en ambientes propietarios hacia los nuevos ambientes abiertos. Las ventajas se esperan a largo plazo. UNIX ha surgido como la plataforma preferida para efectos pragmáticos, impulsando una tendencia general hacia sistemas abiertos aun en áreas donde UNIX no ha sido tradicionalmente popular, como es el ámbito de las aplicaciones administrativas.

El concepto actual de sistemas abiertos tiene sus raíces en muchos esfuerzos por facilitar los procesos de migración y conversión de aplicaciones entre sistemas diferentes, asi como de interconexión de sistemas y redes disímiles.

El trabajo en el modelo de Interconexión de Sistemas Abiertos (OSI, del inglés "Open Systems Interconnect") para redes, que se inició en la década de 1970, hasta principios de la década de 1980, y el trabajo para estandarizar las múltiples versiones de UNIX existentes en ese momento, son dos de los principales esfuerzos.

Más recientemente, la red mundial Internet y su comunidad de usuarios han desempeñado también un papel creciente en importancia. El mercado de las bases de datos relacionales ha contribuido con estándares para acceso y manipulación de datos. Aun el computador personal (PC) y las redes de microcomputadores de área local han ayudado a introducir conceptos y a fijar ejemplos, que han movido a las industrias a aceptar el requerimiento de que los productos deben soportar estándares comunes.

La cooperación y la colaboración entre provee-dores son ahora prácticas de negocios acep-tadas. Se espera que cada proveedor esté en capacidad de apoyar los productos de sus com-petidores. La habilidad de que productos de diferentes proveedores operen entre sí se ve como una capacidad básica del sistema, y no como el resultado de las complejas estructuras de puentes, compuertas y traductores que se usaba antes. Los usuarios esperan de los proveedores que sean capaces de brindar verdadera *interoperabilidad*.

Los sistemas abiertos han movido el balance del poder entre los proveedores de equipos ("hard-ware") y programas ("software"), y actualmente la atención está sobre los proveedores de software, para que sus productos satisfagan las demandas de los usuarios. Esto ha generado cambios importantes en la industria de software.

La industria seguirá siendo poblada por numerosos proveedores de tecnología, que pondrá sus desarrollos al servicio de fabricantes de sistemas y software, quienes integrarán la tecnología en sus productos comerciales. El resultado será que aparecerán productos cada vez más innovadores, que llegarán al mercado más rápido y a menor costo.

Generalidades de Sistemas Abiertos

Historia

Las raíces de los sistemas abiertos se pueden seguir hasta llegar a dos fuentes primarias. Una fueron los esfuerzos para desarrollar estándares de redes independientes de los proveedores que permitieran a los sistemas comunicarse con un conjunto común de protocolos. El modelo OSI de la ISO fue la fundación de otros esfuerzos subsecuentes de desarrollar productos y perfiles que los fabricantes pudieran usar en sus productos para cumplir los requerimientos de los usuarios.

Uno de estos esfuerzos fue una iniciativa de la industria automovilística que resultó en las especificaciones del Protocolo de Automatizacion de Manufactura/ Protocolo Técnico de Oficina (MAP/TOP, del inglés "Manufacturing Automation Protocol / Technical Office Protocol")

Sin embargo, los esfuerzos por difundir el desarrollo de productos basados en el protocolo OSI han encontrado poco éxito, a pesar del reconocimiento de OSI como el modelo internacional estándar. En cambio, los protocolos TCP/IP, que son independientes del proveedor, han continuado creciendo en aceptación, a pesar de no estar basados en un estándar internacional.

Para el final de la década de los 80, era evidente que alcanzar los sistemas abiertos requería un enfoque consistente tanto para las redes como para los sistemas, por lo que los esfuerzos subsiguientes tomaron ambos en consideración.

Durante muchos años, sistemas abiertos eran un sinónimo del sistema operativo UNIX. Esta asociación fue el resultado de que la Corporación AT&T, propietaria original de UNIX a través de su subsidiaria UNIX Systems Laboratories (USL), haya ofrecido las licencias de UNIX tan abiertamente que éste parecía ser del dominio público, y por lo tanto, independiente de un proveedor. Este UNIX "público" recibió aportes tecnológicos de múltiples proveedores y universidades. No fue sino hacia mitades de 1980 cuando AT&T empezó a cambiar su estrategia de licencias de UNIX, que el mercado comprendió que UNIX es en realidad una tecnología propietaria de un solo proveedor.

Adicionalmente, a pesar de que muchos proveedores opinaron que era necesario unificar los distintos "sabores" de UNIX, el método de unificación que empleó AT&T resultó ser impopular.

En ese momento, la mayoría de los productos UNIX estaban basados o en la versión de AT&T System V o en la versión de la Universidad de California de Berkeley conocida como BSD. AT&T trabajó junto con Sun Microsystems Inc. para unificar las dos versiones, más la versión conocida como XENIX de Santa Cruz Operation Inc. (SCO), en una nueva versión del System V, conocida como System V Release 4 (SVR4). Se esperaba que esta unificación redujera las diferencias entre las implementaciones de UNIX de diferentes proveedores y resolviera problemas de interoperabilidad y portabilidad. Sin embargo, esta iniciativa de AT&T y Sun hizo que otros proveedores reaccionaran y formaran la fundación OSF (del inglés "Open Software Foundation"), por lo que los proveedores de UNIX rápidamente se alinearon en dos campos opuestos.

El concepto de los estándares independientes del proveedor volvió al frente a mitad de los años 1980 cuando Digital Equipment Corporation apeló una compra de las Fuerzas Armadas de los Estados Unidos que especificó UNIX (en particular la versión System V Inferface Definition o SVID) como el sistema operativo requerido. La base de la apelación fue que al especificar UNIX, el gobierno estaba especificando al proveedor particular AT&T.

Como resultado, los criterios de la compra debieron ser modificados para solicitar un sistema operativo que cumpliera con las interfaces de UNIX en vez de ser UNIX per se.

De ese punto en adelante, fue evidente que el gobierno debía especificar productos de acuerdo a estándares, y los proveedores de sistemas operativos propietarios empezaron a buscar formas de que los suyos cumplieran estos estándares.
Al inicio de los años 1990, cuando el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE, del inglés "Institute of Electrical and Electronic Engineers") estaba a punto de completar los estándares POSIX que se mencionan más adelante, varios proveedores como Digital, Hewlett Packard Co., Unisys Corp., e International Business Machines Corp. (IBM) empezaron a ofrecer sus sistemas operativos no UNIX con la habilidad de soportar los estándares de la interfaz POSIX. Como resultado, las aplicaciones que fueran escritas para usar la interfaz POSIX no deberían requerir modificaciones para correr en sistemas propietarios. Esto apartó la definición de sistemas abiertos de la igualdad con UNIX, y aun cuando actualmente todavía hay cierta tendencia a igualar sistemas abiertos con UNIX, esta definición resulta muy restrictiva si la idea de sistemas abiertos es cumplimiento de estándares públicos.

Más importante todavía es el hecho que los sistemas abiertos requieren muchos más componentes de software que un sistema operativo. Hay lenguajes de programación; sistemas de archivos; componentes de administración del sistema, de la red, y de los datos;

interfaz de usuario; y otros elementos que son requeridos para lograr un ambiente de sistemas completamente abierto. La definición de este ambiente ha sido enfocada por la organización X/Open en los perfiles que presentan sus documentos XPG4, que se discuten más adelante. Como resultado, hoy en día el requerimiento principal de un sistema abierto es cumplimiento con los perfiles XPG, no solo UNIX.

La adquisición de USL por Novell en 1993 marcó un cambio importante en las políticas de comercialización de UNIX, que han resultado en una mayor apertura del mercado, principalmente porque Novell accedió a transferir la propiedad y la administración de la marca UNIX a la organización X/Open, que se explica más adelante.

Promesas de los sistemas abiertos

Hay muchas definiciones de sistemas abiertos. Algunas enfatizan la interconectividad de redes y otras enfatizan la interoperabilidad del software. Algunas definiciones enfatizan estándares formales y otras aceptan estándares de facto. Subyacentes a estas definiciones están las promesas de los sistemas abiertos, que según la Open Software Foundation, (OSF) son:

Portabilidad: la capacidad de usar sistemas de aplicación en computadores de múltiples fabricantes.

Interoperabilidad: la capacidad de conectar exitosamente computadores de múltiples vendedores.

Escalabilidad: la capacidad de usar el mismo ambiente de "software" en computadores de diversos tamaños y capacidades.

Otros objetivos importantes de los sistemas abiertos son los siguientes:

Reemplazabilidad: la capacidad para reemplazar los componentes de una arquitectura modular.

Administración: la capacidad de usar un único juego de herramientas para administrar el sistema y sus componentes, con una única interfaz para el administrador.

Impacto en los usuarios

Las fuerzas combinadas de las arquitecturas de aplicaciones distribuidas y los sistemas abiertos han cambiado la industria de la computación en los últimos cinco años.

Los usuarios perciben los sistemas propietarios como caros y difíciles de integrar a los ambientes heterogéneos de hoy en día. Frecuentemente se les iguala con arquitecturas centralizadas basadas en macrocomputadores, aplicaciones caras y monolíticas, configuraciones inflexibles, acceso de datos difícil, y falta de interoperabilidad.

Conforme los productos de sistemas abiertos han mejorado sus capacidades, se han definido más estándares, y se han introducido más productos que cumplen con los estándares. La mayoría de los usuarios están empezando a solicitar productos con estándares abiertos en vez de productos propietarios.

Impacto en los proveedores

Los usuarios han mostrado una tendencia de moverse hacia una definición más pragmática de sistemas abiertos en vez de una definición idealizada. La pureza ha resultado menos importante que la funcionalidad. Sin embargo, los proveedores tienen como requisito básico de sus productos el apoyo a los estándares abiertos.
Incluso Microsoft Corp., uno de los proveedores ajenos a sistemas abiertos más visible, apoya muchos estándares claves de sistemas abiertos y reconoce la necesidad de interoperar con el resto del mundo de los sistemas abiertos.

Los productos con interfaces propietarias frecuentemente no son considerados en los casos en que existen estándares abiertos bien definidos. Este cambio se debe a que los usuarios han reconocido que, en muchos casos, las interfaces propietarias los obligan a comprar de un solo proveedor, resultando en menos opciones y mayores costos.

Pero los sistemas abiertos también han representado ventajas para los proveedores. Actualmente los fabricantes no están obligados a desarrollar todas las partes de tecnología que requieren internamente, por lo que cuando desean introducir nuevos productos, hay una mayor disposición de buscar la tecnología necesaria en fuentes externas de la industria.

Como resultado, la industria se está organizando alrededor de proveedores de tecnología que desarrollan sus ideas y las hacen disponibles a los fabricantes de sistemas y software. Estos fabricantes agregan valor a la tecnología al integrarla en productos finales. El resultado esperado es mayor innovación a menor costo, menor tiempo en llegar al mercado, y ciclos de vida de productos más cortos.

Los fabricantes también pueden ofrecer una mayor variedad de productos que si se vieran obligados a desarrollar la tecnología por sí mismos. Las concesiones de tecnología se han convertido en una práctica aceptada por la industria, que genera ingresos considerables, y aumenta la influencia del desarrollador de la tecnología sobre la industria.

Pasos para llegar a los sistemas abiertos

Para alcanzar las promesas y objetivos de los sistema abiertos, los sistemas de información deben ser diseñados y construidos para maximizar el potencial de portabilidad, escalabilidad, reemplazabilidad y administración. A continuación se presentan algunos enfoques para construir sistemas que logren esos objetivos.

Selección de arquitecturas modulares. Las arquitecturas modulares son los elementos básicos de los sistemas abiertos. Los esfuerzos para definir completamente una amplia arquitectura común aun no hay dado frutos debido a la complejidad de la tarea. Sin embargo, hay consenso sobre los elementos principales, que se muestran en la Figura 1. Estos elementos se instrumentan en la forma de productos que vienen de múltiples fuentes en la industria y que se integran en sistemas completos, ya sea por el proveedor, un consultor en integración de sistemas, o el cliente.

Un requisito clave es que la tecnología de cada componente dentro de la arquitectura se pueda reemplazar con poco impacto sobre los demás componentes.

Diseño basado en estándares. Los estándares son especificaciones para definir las interfaces que han sido aceptados por una organización formal de estándares (llamado estándares de jure), o que han sido adoptados ampliamente por múltiples fabricantes y usuarios, y que se consideran estándares de facto. El diseño de los sistemas debe estar basado en estándares.

Selección de productos de amplia disponibilidad. Los productos que cumplen con los estándares normalmente están

ampliamente disponibles, lo cual es una de las principales características y beneficios de los sistemas abiertos. Los usuarios pueden seleccionar entre diferentes vendedores basándose en precio, rendimiento, soporte, calidad, y otras características, con la confianza de que los productos son funcionalmente equivalentes.

Esta equivalencia es todavía un ideal no alcanzado, pero mediante el estudio de especificaciones detalladas y pruebas prácticas se puede determinar la compatibilidad de los productos entre sí.

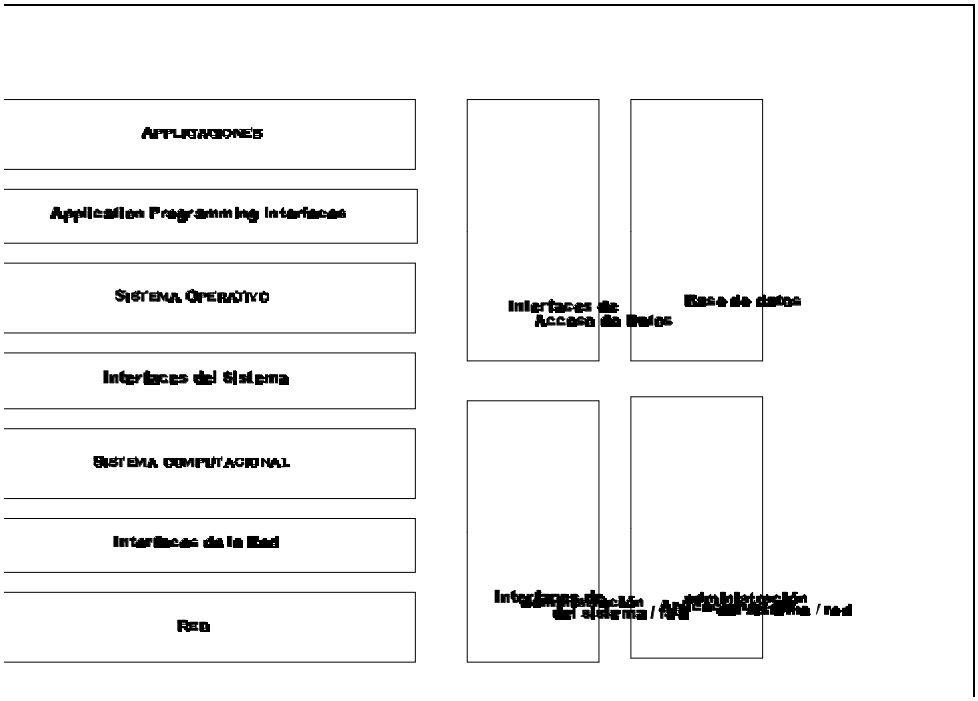


Figura 1

Principales Elementos de una Arquitectura de Sistemas Abiertos de Software

Conceptos de Sistemas Abiertos

Hay varios conceptos básicos presentes en la mayoría de las discusiones de sistemas abiertos. Estos conceptos generales deben ser bien entendidos porque ellos permiten hacer distinciones importantes cuando se planifica un enfoque de sistemas abiertos.

Compatibilidad

Los sistemas de diferentes proveedores que han sido diseñados con el mismo microprocesador, teóricamente pueden proveer diferentes niveles de compatibilidad.

Es posible encontrar diferentes sistemas basados en un procesador SPARC, por ejemplo, que no sean compatibles, que sean compatibles a nivel de código fuente, o que sean compatibles a nivel binario.

Dos sistemas son compatibles a nivel de código fuente si el mismo programa fuente, escrito en un lenguaje de alto nivel como COBOL o C, puede ser compilado y ejecutado en ambos sistemas sin requerir modificaciones. Muchos niveles de estandarización como POSIX y SPEC 1170, que se mencionan más adelante, están orientados a crear compatibilidad de código fuente entre sistemas. A diferencia de la compatibilidad a nivel binario, la compatibilidad de código fuente no requiere hardware idéntico.

Dos sistemas son compatibles a nivel binario si el mismo código objeto ejecutable corre en forma nativa en ambos, sin modificaciones, adaptaciones o emulaciones. El beneficio del la compatibilidad binaria es que produce un mercado de software empacado ("shrink-wrapped") que expande las opciones del usuario y el mercado del desarrollador de paquetes.

Contar con hardware idéntico es necesario pero no condición suficiente para asegurar compatibilidad binaria. Aunque los sistemas usen el mismo procesador, las interfaces que soporta el sistema operativo determinan si un programa compilado puede ejecutar en sistemas de vendedores diferentes.
El PC demostró que la compatibilidad binaria era posible, e hizo evidente las ventajas del mercado de paquetes de software. Para lograr la compatibilidad binaria, los sistemas deben verse exactamente iguales desde la perspectiva de la aplicación. El estándar binario del PC es un resultado de facto de la utilización del sistema operativo MS-DOS en el PC de la IBM.

La compatibilidad binaria en otras combinaciones de hardware y sistemas operativos ha requerido la utilización de un proceso formal de producir una especificación de una Interfaz Binaria de Aplicación (ABI, del inglés "Application Binary Interface").

Para determinar si un sistema cumple totalmente con una especificación binaria se utilizan pruebas de conformidad, que muestran que el código binario corre en todos los sistemas aplicables. Las ABIs son específicas del sistema operativo, lo que implica que se requiere una ABI diferente para cada procesador bajo SVR3, SVR4, o cualquier otro sistema operativo.

Definiciones y especificaciones de interfaces

Las definiciones de interfaces especifican cómo calzan entre sí los componentes de una arquitectura, y describen cómo opera la misma. Cada componente en la arquitectura modular debe tener definiciones de interfaces limpias que describan cuáles servicios provee al resto de la arquitectura y cómo los otros componentes de la arquitectura obtienen estos servicios. Estas definiciones conforman un juego de especifi-caciones que puede ser implantado por un número ilimitado de productos específicos. Lo importante no son los detalles internos de cómo un producto dado provee esos servicios, pero sí que los servicios sean obtenidos externamente de una manera consistente.

Estándares formales

Los estándares formales son el resultado de esfuerzos de organizaciones especializadas oficialmente reconocidas nacional e internacionalmente. Típicamente pasan por varias revisiones y votaciones antes de ser aceptados finalmente, por lo tanto, el proceso de definir estándares formales y obtener consenso es un proceso largo.

Debido a que es necesario que todos los vendedores potenciales se encuentren en igualdad de condiciones, los estándares formales normalmente no se basan en tecnologías existentes, y toman tiempo para convertirse en productos que los soportan. También es cierto que los estándares formales son difíciles de mantener y de extender para cubrir necesidades inmediatas del mercado.

Una vez que un estándar ha sido adoptado, cualquiera puede desarrollar tecnología para implementar esa especificación. Por ejemplo, POSIX 1003.1 es una especificación de interfaz de servicios de sistema, que es un estándar formal.

Estándares de facto

En contraste con estándares formales, los estándares de hecho o de facto, son definidos por el mercado, no por comités ni consorcios.

Un fabricante puede acelerar la adopción de un estándar de facto si desarrolla una tecnología y la hace disponible a otros para la fabricación de productos. Ese es el caso del lenguaje de consulta estructurado SQL (del inglés "Structured Query Language"), originalmente definido por IBM y colocado en el dominio público.

Posteriormente SQL se convirtió en un estándar de facto, y eventualmente, alrededor de 10 años despúes, SQL fue incorporado como un estándar formal del Instituto Americano de Estándares Nacionales (ANSI, del inglés "American National Standards Institute") y la ISO.

Pruebas de cumplimiento

Tener estándares y especificaciones tiene relativamente poco sentido a menos que exista un proceso para verificar que los productos cumplan con esas especificaciones. Las especificaciones por sí solas no miden cumplimiento, de manera que es necesario desarrollar pruebas para medir el grado en que un producto específico cumple con una especificación o estándar. Estas pruebas, llamadas pruebas de cumplimiento, ofrecen la seguridad de que un producto realmente cumple con el estándar que dice su fabricante.

Si un estándar no está totalmente especificado o permite muchas opciones, los productos que pasaron en forma individual una prueba de cumplimiento pueden no trabajar juntos en forma compatible. Para resolver este problema, existen pruebas de interoperabilidad que miden el grado de éxito con que dos implementaciones de la misma especificación trabajan juntas.

Algunas organizaciones proporcionan certificaciones de los resultados de las pruebas de cumplimiento. X/Open ha asumido un papel creciente en la certificación de productos de sistemas abiertos.

SQL y acceso de datos

La administración y el acceso a los datos es una consideración importante en los sistemas abiertos. La evolución de estándares hacia un lenguaje común de manipulación de datos como el SQL ha sido clave. Los estándares para la administración de transacciones distribuidas han evolucionado gracias a los esfuerzos de X/Open.

Estándares de Sistemas Operativos

Introducción

Los estándares de sistemas operativos han nacido alrededor del proceso de definición de estándares de UNIX. Este capítulo describe los principales pasos del proceso y las organiza-ciones independientes han participado en éste.

Definición de la interfaz del System V

Las interfaces para el UNIX System V de Novell Inc. (antes de AT&T) está contenidas en un documento llamado "System V Interface Definition" SVID.

SVID fue inicialmente publicado en 1985 para describir las interfaces de System V Release 2 (SVR2). Fue actualizado en 1989 a la versión actual, Tercera Edición, que describe SVR4. En 1991 se adicionó un quinto volumen para definir modificaciones menores a los cuatro volumenes anteriores, y para añadir secciones nuevas sobre seguridad e interfaces de administración de sistemas remotos. SVID es un precursor del SPEC 1170 que se describe más adelante.

Para que un fabricante pueda decir que su producto es compatible con SVR4, éste debe cumplir con SVID. El cumplimiento se mide con una serie de pruebas llamadas "System V Verification Suite" (SVVS). SVID no es un ABI, tanto porque es independiente de la arquitectura del procesador, como porque no especifica los detalles de cómo se implementa System V en un procesador específico.

POSIX

"Portable Operating System Interface" POSIX es el nombre colectivo de los estándares que han resultado del trabajo de un grupo de comités del IEEE, que define especificaciones de interfaces de sistemas abiertos de computación.
Actualmente hay más de 20 comités POSIX importantes trabajando. La Tabla 1 muestra cada comité, el área en la que está trabajando y el estado de sus esfuerzos.
El grupo de trabajo POSIX 1003.1 inició el proce-so de POSIX al convertir lo que empezó como el estándar */usr/group* en el "IEEE Trial Use Stan-dard" de 1986, que fue adoptado como el están-dar IEEE 1003.1 en 1988 y modificado en 1990.

Una medida clave del progreso de los estándares POSIX es cuántas especificaciones han sido aprobadas y cuántas están en su versión final esperando ser votadas.

La importancia de las especifiaciones POSIX viene de que el ANSI ha certificado al IEEE como el organismo reconocido en este área. El Gobierno de los Estados Unidos utiliza los estándares POSIX para basar sus propios estándares llamados FIPS (del inglés "Federal Information Processing Standards"). Evidente-mente los proveedores tienen mucho interés en hacer sus productos compatibles con POSIX, debido al gran potencial de mercado que representa ese gobierno.

X/Open

X/Open es una organización compuesta por más de 130 fabricantes, usuarios y proveedores de software independiente (ISVs, del inglés "Independent Software Vendors").

Fundada en 1984, su misión es integrar estándares formales y de facto en un grupo de especificaciones para sistemas abiertos llamado el Ambiente Común de Aplicaciones (CAE, del inglés "Common Applications Environment"). Las especificaciones de CAE se publican en un grupo de manuales llamados "X/Open Portability Guide" (XPG), que se mencionan en detalle más adelante.

Tabla 1

Comités de Estándares POSIX

Proyecto Número Promberuecto	Estado	Título original en inglés
P1003.0		Open Systems Environment Guide
P1003.1	IEEE Std 1003.1 1990	System API
P1003.1a		System API extensions
P1003.1b	IEEE Std 1003.1b 1993	Real-time extensions
P1003.1c	IEEE Std 1003.1c 1993	Threads extensions

P1003.1d		Further real-time extensions
1003.1e		Security API extensions TransparentP
P1003.1f		Transparent file access
P1003.1g		Protocol independent interfaces
P1003.2	IEEE Std 1003.2 1992	Shell and utilities
P1003.2a	IEEE Std 1003.2 1992	User portability
P1003.2b		Shell and utilities extensions:interpretations,clarifications,etc. extensions,including symbolic link support by the standard utilities and a new file archive and interchange format
P1003.2c		Security utility extensions TransparentP
P1003.2d		Batch extensions
P1003.3	IEEE Std 1003.3 1992	Test Methods
P1003.5	IEEE Std 1003.5 1992	Ada binding
P1003.5a		Ada binding amendment
P1003.5b		Ada binding to real time
P1003.9	IEEE Std 1003.9 1992	Fortran-77 binding
P1003.10 P1003.13 P1003.14		Supercomputing profile
P1003.13 P1003.13 P1003.14		Real-time profiles
P1003.14 P1003.13 P1003.14		Supercomputing profile
P1003.16		C binding to language-independent API
P1003.18		POSIX Platform Profile
P1003.21		Real-time distributed systems
P1003.22		Distributed Security Framework Guide
P1224.2		X.500 API
P1372		Language-independent system API
P1387.1		Framework for system administration
P1387.2		Software management
P1387.3		User management
P1387.4		Print administration
P2003.1	IEEE Std 2003.1 1992	Test methods for 1003.1
P2003.2		Test methods for 1003.2

Los comités técnicos de X/Open consideran las solicitudes dentro de un proceso de definición de requerimientos de sistemas abiertos. Estos requerimientos se convierten en áreas para las que desarrollan especificaciones, pruebas de cumplimiento y validación, y certificaciones.

El proceso de certificación de X/Open está diseñado para garantizar que un producto cumple con las especificaciones de interfaz de XPG. X/Open calcula que en 1993 el mercado compró sistemas basándose en especificaciones y certificaciones XPG por un monto aproximado a los 7 mil millones de dólares americanos.

X/Open ha surgido como el principal vehículo para un proceso neutral de definir y adoptar especificaciones de sistemas abiertos. Ha evolucionado respondiendo a la industria, creando nuevas estructuras y procesos para cumplir con las demandas del mercado.

X/Open es el único consorcio con un proceso formal, llamado Xtra, para identificar y procesar requerimientos de usuarios de sistemas abiertos. Este proceso, que combina encuestas con conferencias, ha permitido a la organización identificar áreas de trabajo importantes y fijar prioridades que responden a los requerimientos de los usuarios.

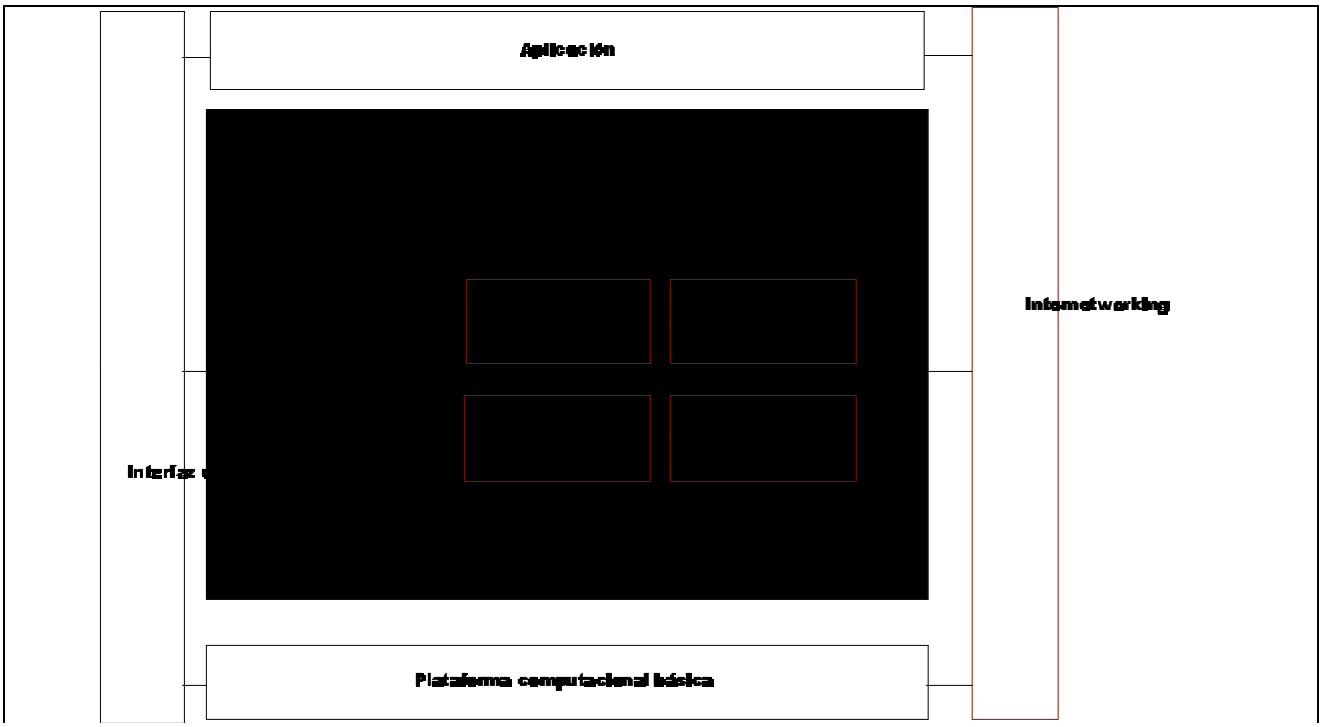
XPG

XPG es el juego de documentos en los que se han publicado las especificaciones de CAE, y son la guía conceptual de hacia donde se dirige X/Open con XPG4 y más adelante. XPG proporciona especificaciones detalladas e implementables para los componentes de XPG en cada una de las áreas tecnológicas de CAE que se muestran en la Figura 2.

La primera publicación de XPG, llamada XPG1, se realizó en 1985, antes de que los sistemas abiertos se convirtieran en una fuerza real en el mercado. XPG1 de 1985, XPG2 de 1987, y XPG3 de 1989, estaban enfocadas a aspectos de portabilidad de aplicaciones relacionados con interfaces básicas de los sistemas operativos, comandos, utilitarios, lenguajes de tercera generación (3GL), administración de datos y de archivos, y redes rudimentarias.
XPG está basado en estándares de jure como POSIX y estándares de facto ampliamente aceptados. Desde su enfoque original hacia portabilidad de aplicaciones, ahora se ha orientado a interoperabilidad de ambientes heterogéneos. XPG incluye gran cantidad de áreas, desde 3GLs, hasta SQL y OLTP (del inglés "On Line Transacction Processing"). XPG está en su cuarta versión, y su amplitud y complejidad han llegado a un punto que se ha hecho muy importante la definición de perfiles, o subconjuntos de XPG, que puedan ser utilizados por las organizaciones para especificar adquisiciones (licitar).

XPG4

XPG4 fue liberado el 1992, ofreciendo muchos componentes actualizados y nuevos. XPG4 extendió y mejoró a sus predecesores en áreas claves del CAE, particularmente en relación con procesamiento distribuido, administración de objetos, y OLTP. XPG4 avanza más allá de portabilidad e interoperabilidad como enfoque principal, y toca aspectos como acceso de datos en arquitecturas tradicionales, y computación en ambientes heterogéneos. XPG4 agrupa los componentes en perfiles que ofrecen algo cercano a conjuntos (bibliotecas) de funciones que un usuario realmente puede comprar. Los perfiles son certificados cuando sus componentes están certificados. Se han enunciado seis perfiles en XPG4, cinco de los cuales están totalmente definidos, y el sexto consiste de los componentes de computación distribuida por definir. Los perfiles certificables actualmente son: el Perfil Base ("Base Profile"), Perfil de Compuerta de Comunicaciones OSI ("OSI Communications Gateway Profile"), Perfil de Servidor Básico ("Base Server Profile"), Perfil de Estación de Trabajo ("Workstation Profile"), y Perfil de Plataforma de Base de Datos ("Database Platform Profile"). Las certificaciones podrán ser emitidas cuando se terminen las pruebas de cumplimiento.



OSF

La "Open Software Foundation" OSF fue fundada en 1988 por un grupo de importantes proveedores de sistemas, incluyendo IBM, DEC, HP, Siemens AG, y Nixdorf Computer AG, para promover las concesiones justas y colaborativas de software de sistemas abiertos.

La formación de OSF se debió a la percepción de estos proveedores de que, según su criterio, AT&T estaba concediendo licencias de UNIX en forma injusta, y de que el acuerdo entre AT&T y Sun para trabajar conjuntamente en System V y el UNIX de Berkeley le daban a Sun una ventaja injusta sobre otros proveedores de UNIX.

OSF pronto se convirtió en una casa desarrolladora de software, bajo el concepto de utilizar a toda la industria como su equipo de desarrollo, por medio de un proceso de emitir públicamente solicitudes de tecnología (RFT, del inglés "Request for Technology").

La primera RFT que emitió OSF resultó en la especificación, diseño e implementación de la interfaz gráfica de usuario (GUI, del inglés "Graphical User Interface") OSF/Motif.

La segunda RFT, denominada ANDF (del inglés "Architecture Neutral Distribution Format"), cuyo resultado final es aún incierto, propuso un método para distribuir software de aplicación en forma independiente de la plataforma de hardware.

El concepto básico de solución propone que el código fuente de una aplicación se convierta a ANDF para distribución, por medio de un "productor" de ANDF, para luego ser transformado en código ejecutable en la máquina de destino, por medio de un "instalador" de ANDF. El proyecto ha sido calificado como *técnicamente difícil, de alto riesgo y de largo plazo.*

En 1990 OSF publicó una lista de organizaciones preseleccionadas para trabajar en ANDF: Hewlett-Packard, Peritus International, Siemens AG, National Semiconductor Corp. y la Universidad de Virginia.

La tercera RFT resultó en la especificación, diseño e implementación del ambiente de computación distribuida DCE ("Distributed Computing Environment"), que se menciona en detalle más adelante.

La cuarta RFT, denominada DME (del inglés "Distributed Management Environment"), no muestra resultados positivos a la fecha.

Sin embargo, la relativa independencia de sus patrocinadores que disfrutaba OSF, la llevó en algunos casos a seleccionar proyectos que no estaban igualmente apoyados por todos sus miembros, aunque sí representaban una carga financiera para todos. Además, algunos de los procesos de selección y desarrollo de tecnología de OSF no fueron conducentes a las prioridades de tiempos de llegada al mercado.

En marzo de 1993 OSF anunció un nuevo esquema de operación: todo el trabajo de desarrollo sería realizado por contratistas externos, todos los proyectos de desarrollo debían contar con soporte de miembros interesados, y todos los proyectos de desarrollo deben ser financiados por los miembros interesados.

OSF asumirá más una labor de administración de proyectos, y menos de fijación de políticas y direcciones, respondiendo a las solicitudes de los miembros. Se espera que los trabajos en DCE y Motif continúen con el apoyo de los miembros, pero el futuro de otros proyectos como ANDF, el sistema operativo OSF/1 y DME es mucho menos claro.

OSF presentará las especificaciones de sus tecnologías, junto con sus pruebas de validación a X/Open, para que sean certificadas por ésta. Como muestra de que el nuevo esquema de OSF tiene apoyo de la industria, tanto Sun como Novell (el nuevo propietario de UNIX) se han unido recientemente.

DCE

El DCE ha sido uno de los logros más importantes de OSF. La tecnología definida por DCE es la fundación para el desarrollo de aplicaciones distribuidas que pueden interoperar entre sistemas de proveedores diferentes.

DCE consiste de un juego de interfaces de programación de aplicaciones (API, del inglés "Application Programming Interface"), que proporcionan acceso a un conjunto de servicios distribuidos en la red.

El código fuente de la versión inicial de los servicios centrales de DCE ha estado disponible para proveedores de sistemas, ISVs e instituciones de investigación desde hace dos años, y los productos de servicios centrales de DCE han estado disponibles en el mercado desde hace un año.

Los componentes de DCE está diseñados para habilitar el procesamiento distribuido por medio de los siguientes servicios:

Threads: soporta la creación , administración y sincronización de múltiples hilos ("threads") de control dentro de un proceso único.

Remote Procedure Call (RPC): Permite que un programa que corre en un sistema invoque la ejecución de otro programa en otro sistema. El RPC de OSF incluye una herramienta de desarrollo y un servicio de apoyo en tiempo de ejecución ("runtime").

La herramienta de desarrollo incluye el lenguaje IDL, que se menciona más adelante, y su compilador, que se utiliza para crear aplicaciones en el modelo cliente servidor. El "runtime" implementa los protocolos de red que usan las aplicaciones del cliente y del servidor para comunicarse entre sí.

Directory Service: Proporciona una base de datos de recursos de DCE, como archivos, servidores, discos o colas de impresión.

Time Service: Proporciona sincronización del tiempo del sistema para todos los nodos que corren en el ambiente DCE.

Security Service: Autentica las identidades de los clientes y los servidores, verifica los niveles de autorización, verifica la integridad de datos, y asegura la privacidad de los datos. Este servicio provee la infraestructura para un solo "logon" de usuario en un ambiente distribuido, y puede ser razón suficiente para adoptar DCE en organizaciones grandes.

Distributed File System (DFS): Provee acceso transparente a los archivos distribuidos del sistema. El DFS está basado en el "Andrew File System" (AFS), desarrollado originalmente por el Proyecto Andrew de la Universidad de Carnegie-Mellon, en vez del más ampliamente conocido "Network File System" (NFS) originalmente desarrollado por Sun. OSF seleccionó AFS como base para DFS por la cantidad de ventajas técnicas que tiene sobre NFS. Por ejemplo, DFS permite que un sistema mantenga en un cache local una copia de parte del sistema de archivos, actualizando el cache solo cuando cambia la

copia original, lo cual mejora el rendimiento y disminuye el tráfico en la red.

DCE incluye el lenguaje de desarrollo IDL (del inglés "Interface Definition Language") y su compilador. Algunos usuarios pioneros han optado por construir clases de C++ sobre IDL para manejar código repetitivo, otros han trabajado con herramientas de terceros como PC-DCE o Visual DCE de Gradient Technologies Inc., y otros han trabajado directamente a nivel de IDL.

El trabajo en DCE continuará a pesar de los cambios en OSF mencionados, dado que tiene una prioridad alta para los patrocinadores de OSF. Se están evaluando extensiones para permitir a DCE interoperar, entre otros, con sistemas de bases de datos relacionales (RDBMS), con sistemas de proceso de transacciones, y con la arquitectura de objetos comunes CORBA (del inglés "Common Object Request Broker Architecture") del Object Management Group (OMG), que se menciona en detalle en el capítulo Cliente-Servidor y Sistemas Abiertos de este informe. Por ejemplo, HP ha propuesto extensiones para el modelo de seguridad de DCE para ambientes CORBA de proceso de transacciones en línea, que posiblemente sean adoptadas por OSF.

La versión 1.1 de DCE, que se espera para fines de 1994, tendrá mejoras sustanciales en confiabilidad, disponibilidad, y mantenimiento, incluyendo funciones de auditoría y contabilidad. La versión tendrá código más compacto y más rápido, un ambiente de "runtime" mejorado, incluirá soporte para caracteres internacionales en los servicios de directorio y nombres, y otras mejoras más.

A pesar que hasta ahora DCE solo ha sido implementado en plataformas UNIX, a partir de 1995 estará disponible en plataformas propietarias como IBM MVS y AS/400, entre otras.
De hecho, una de las mayores atracciones de DCE es que estará soportado en plataformas UNIX y no-UNIX, permitiendo la creación de aplicaciones que cubran todo tipo de sistemas. Desafortunadamente, el soporte de DCE a los sistemas operativos de microcomputadores ha sido bastante débil, debido principalmente a que las tecnologías y productos sobre los que se basa DCE tradicionalmente no han estado disponibles en estos sistemas. Como excepción está el PC-DCE de Gradient antes mencionado.

COSE

En marzo de 1993, una alianza de IBM, HP, Sun, Univel Inc., USL, y SCO, llamada COSE (del inglés "Common Open Software Environment") y dedicada a estandarizar UNIX, anunció que logró un acuerdo sobre un juego de interfaces y servicios comunes para el usuario. Este acuerdo acabó con las disputas de años entre los provee-dores de las interfaces gráficas Open Look y Motif, que causaba múltiples problemas a usuarios y desarrolladores, y prometió una única interfaz de usuario para UNIX.

La especificación de COSE, llamada CDE (del inglés "Common Desktop Environment") ha sido presentada a X/Open para ser considerada y adoptada como especificación X/Open.

Puntos importantes a favor del CDE de COSE es que Sun ha decidido apoyar a Motif y apartarse de su adherencia a Open Look, y que incluye elementos del Workplace Shell de IBM, del Visual User Environment de HP, y del X.desktop de SCO. Novell contribuyó con la especificación del cliente NetWare, y se incluyeron elementos de los protocolos OpenLook y ToolTalk.

Los comités técnicos de X/Open han decidido extender los trabajos de CDE para que incluya integración de aplicaciones, administración y multimedios, entre otros, para que el ambiente llegue a los niveles de funcionalidad de Microsoft Windows y Apple Macintosh.

SPEC 1170

A pesar de que UNIX ha provisto la fundación para las especificaciones de sistemas abiertos y estándares para software de sistemas, los productos UNIX de diferentes proveedores difieren en los APIs del sistema. Aun cuando todos soportan el estándar POSIX.1, este estándar representa un subconjunto de una interfaz de sistema operativo completa. Los productos UNIX difieren en las APIs que contienen, y muchos tiene diferencias en sintaxis y en semántica de un API particular.

En 1993, los principales proveedores de UNIX se unieron y acordaron un juego común de 1,170 APIs, que representan la mayoría de los llamados al sistema que necesitan los programas de aplicaciones. Estos APIs se convirtieron en SPEC 1170. SPEC 1170 no implica un acuerdo de estandarización en un solo UNIX o un solo "kernel" de UNIX, variantes de los cuales seguirán existiendo aun despúes de que la especificación sea aceptada. Sin embargo, pronto el nivel de estandarización de un producto UNIX en particular dependerá de su cumplimiento completo con SPEC 1170.

La base para la especificación SPEC 1170 Common API, es la especificación básica XPG4, que a su vez contiene muchos API estándares existentes, como POSIX 1003.1. SPEC 1170 es un superconjunto de la especificación XPG4, y la extiende a áreas adonde existen especificaciones ya aprobadas, e incluye el SVID 3 nivel 1. SPEC 1170 cubre un promedio del 98% de las APIs utilizadas, porcentaje que representa una mejora significativa sobre el promedio del 60% que cubre XPG4. Cuando SPEC 1170 se combine con CDE de COSE, los desarrolladores que escriban aplicaciones para UNIX tendrán un juego mucho más completo de APIs para aplicaciones portables que el que han tenido en el pasado.

Cuando SPEC 1170 se apruebe, esta especifica-ción determinará qué sistemas operativos se podrán llamar UNIX, independientemente de cuál sea su tecnología base o el origen de su código fuente.

Posición de los proveedores

Hoy en día, una estrategia de sistemas abiertos es crítica para competitividad de todo proveedor. Algunos proveedores han abrazado los sistemas abiertos en la forma de UNIX directamente, otros han elevado los sistemas abiertos y UNIX al mismo nivel de sus sistemas propietarios. En la Tabla 2 se muestra una visión general de la posición de los principales proveedores de sistemas abiertos de software.

En 1992, Sun continuó como el principal provee-dor de UNIX en términos de cantidad de entre-gas, cobertura de mercado y tasa de crecimiento. HP ha estado avanzando en los últimos años, al igual que IBM. Sun ha sentido la presencia de estos dos competidores, y ha reaccionado con un nuevo impulso a sus productos UNIX.

Tabla 2

Estado de los Proveedores de Sistemas Abiertos de Software

Proveedor	Sistema Operativo UNIX	Estado de los proveedores de Sistemas Abiertos	Software para Sistemas Abiertos	Comentarios
HP	HP-UX	MPE/IX	SoftBench, Openview	Agresivamente mercadeando UNIX y paquetes de software comerciales
IBM	AIX	MVS Open Edition AS/400 ("statement of direction")	Netview/6000, DB2/6000, CICS/6000	Balanceando ofrecimientos deUNIX y productos propietarios
Sun	Solaris		Solaris,ONC+, SunNet Manager	Nueva iniciativa de UNIX comercial; pocas alianzas en la industria
Digital	OSF/1	OpenVMS, Windows NT	Pathworks, Accessworks, Motif	Alpha and OSF/1 son importantes para su futuro
Novell	UnixWare	NetWare ("statement of direction")	UNIX, Tuxedo	Enfrenta un reto para mantener la posición de mercado de NetWare, e introducir AppWare, y UnixWare
AT&T-GIS (NCR)	UNIX SVR4		Top End	Principal proponente de SMP; luchando con su estrategia de software
Unisys	UNIX SVR4			Ayudando a su base instalada a migrar hacia sistemas abiertos
Data General	DG-UX			Agresivamente buscando una estrategia de disponibilidad de software
Sequent	Dynix/ptx			Servidores corporativos basados en los chips Intel SMP y UNIX
Pyramid	DC/OSx			Servidores corporativos y departamentales basados MIPS y UNIX
SCO	Open Desktop, Enterprise Server			Un distribuidor líder de sistemas abiertos para Intel; solo software, compite con Sun para el liderazgo de volumen de UNIX
Microsoft		Windows NT, LAN manager for UNIX		Oponiéndose a la tendencia; plataforma propietaria, abierta a otros desarrolladores para conectarse; va a interoperar con sistemas abiertos, pero seguirá su propia dirección

Cliente-Servidor y Sistemas Abiertos

Introducción

El modelo de aplicaciones cliente-servidor es tal vez la más visible manifestación de los sistemas abiertos de software. Cliente-servidor es una forma de procesamiento distribuido que está empezando a cambiar la manera en que muchas aplicaciones se diseñan y construyen. Los sistemas que se construyen con este modelo son inherentemente modulares, algunas de sus funciones corren en los equipos de los clientes y otras en los equipos servidores. Algún tipo de red de datos une los clientes con los servidores. El modelo cliente-servidor descansa en las promesas de sistema abiertos de portabilidad, interoperabilidad y escalabilidad, antes discutidos.

La mayoría de las organizaciones que fueron pioneras con sistemas cliente servidor al final de los años 1980, esperaban que esta tecnología redujera los costos de diseño e implantación de sistemas, y permitiera mover los sistemas desde mini y macro computadores hacia microcom-putadores y servidores de bajo costo. En la práctica, sin embargo los ahorros en equipos se vieron cancelados por los altos costos de desarrollo, entrenamiento, nuevo software requerido por las aplicaciones, y soporte a la operación.

Como resultado, la tendencia actual hacia cliente-servidor ahora se justifica por medio de la mejoras en la flexibilidad y productividad para el usuario que brinda la nueva tecnología. La arquitecturas cliente-servidor se apoyan en ambientes como Microsoft Windows, OS/2 de IBM, y otras interfaces gráficas de usuario que están disponibles para sistemas abiertos de microcomputadores PC, Macintosh, y estaciones de trabajo UNIX. La opinión generalizada es que estas interfaces aumentan la productividad del usuario, principalmente en las aplicaciones de apoyo a la toma de decisiones.

Un segundo catalizador para las arquitecturas cliente-servidor han sido las estrategias de muchas corporaciones de dotar a las unidades de negocios con acceso a los datos requeridos para cumplir con sus objetivos y misión de negocios. Para alcanzar esto, las empresas han dado a los usuarios la responsabilidad de administrar sus propios datos. En muchos de estos casos, la empresa depende de arquitecturas cliente-servidor en sistemas abiertos para coordinar estas bases de datos locales con las bases de datos corporativas centralizadas.

La naturaleza modular de las arquitecturas cliente-servidor favorece su implantación en sistemas abiertos de hardware y software. Muy frecuentemente los componentes de la plataforma tecnológica cliente-servidor (el servidor, la red, las estaciones de usuarios, la base de datos, la herramienta de desarrollo, etc.) vienen de proveedores independientes entre sí.

Finalmente, los modelos cliente-servidor ofrecen la posibilidad de descomponer las aplicaciones en módulos que a largo plazo pueden resultar más fáciles de mantener y de adaptar a los cambios que las aplicaciones monolíticas basadas en macrocomputadores.

Muchas organizaciones están realizando inversiones considerables en tecnologías cliente-servidor, pero, para la mayoría, la computación distribuida sigue siendo "de avanzada", y no están comprometidas a utilizarla para aplicaciones de automatización operativa y de proceso de transacciones. En estas organizaciones, cliente-servidor se utiliza para aplicaciones estratégicas y de apoyo a la toma de decisiones, y aplicaciones de comunicaciones en redes que no necesariamente involucran la administración de los datos vitales de la empresa.

Hay cuatro barreras principales para el uso de la tecnología cliente-servidor en las aplicaciones operativas.

- La primera es que las herramientas de desarrollo cliente-servidor actuales limitan el tamaño y el alcance de las aplicaciones.

- La segunda es que los ambientes y las herramientas cliente-servidor demuestran falta de robustez y confiabilidad cuando se les compara con los ambientes multiusuario tradicionales.
- La tercera es que la mayoría de las herramientas cliente-servidor obligan a los usuarios a aprender esquemas de desarrollo de aplicaciones totalmente nuevos para muchos (bases de datos relacionales distribuidas, programación de objetos, etc.) y sin proporcionar ninguna ayuda de migración.
- El cuarto problema es la complejidad y el esfuerzo requerido para administrar y soportar un ambiente cliente-servidor grande basado en sistema abiertos. Hay pocas herramientas bien reconocidas que soportan manejo de configuraciones, monitoreo del rendimiento, y distribución de nuevas versiones de software.

Cliente-servidor simplifica el procesamiento distribuido, pero todavía es difícil diseñar aplicaciones cliente-servidor que tengan buen desempeño. Hasta que estas barreras no desaparezcan, cliente-servidor deberá coexistir con las arquitecturas centralizadas tradicionales. Se espera que en los próximos años la tecnología cliente-servidor madure lo suficiente como para convertirse en la plataforma primaria para la construcción de las nuevas aplicaciones empresariales.

El aumento en la oferta de paquetes de software en modelos cliente-servidor es una indicación de esta tendencia. Todo parece indicar que en el futuro es inevitable que el modelo cliente-servidor reemplace las arquitecturas centralizadas en la mayoría de las organizaciones.

Antecedentes

Las arquitecturas cliente-servidor son la nueva realidad en los sistemas de información corporativos. Las organizaciones pioneras ya han completado sus primeros proyectos y están utilizando lo que han aprendido a su segunda generación de proyectos cliente-servidor.

La mayoría de las organizaciones está pensando en iniciar proyectos cliente-servidor. La tendencia es gradual, no revolucionaria, pero está claro que cliente-servidor se está convirtiendo en la forma aceptada de construir nuevas aplicaciones, principalmente en el ambiente de sistemas abiertos.

Como toda nueva tecnología, cliente-servidor tiene un nivel de riesgo asociado. La tecnología actual no es apta para todo tipo de aplicaciones. Para tener éxito, los desarrolladores deben dominar nuevas herramientas y técnicas. No todos los usuarios han tenido éxito con esta nueva tecnología.

A pesar de los riesgos, hay cuatro razones principales que impulsan la tendencia a aplicaciones cliente-servidor:

1. Las arquitecturas cliente-servidor están muy asociadas con las interfaces gráficas de usuario como Microsoft Windows, OS/2 de IBM, y Motif de OSF. Para aplicar estas interfaces a las aplicaciones corporativas, es necesario mover éstas a modelos cliente-servidor.

2. Muchos usuarios esperan que la arquitectura cliente-servidor reduzca sus costos de sistemas de información a largo plazo. Al construir una aplicación sobre una red de PCs de sistemas abiertos, que emplea servidores con un costo menor a los $10,000, es posible evitar los altos costos de mini y macrocomputadores propietarios.

   En términos generales, el poder de cómputo por unidad de medida es mucho más barato en los microcomputadores y servidores de sistemas abiertos que en mini y macrocomputadores propietarios. Sin embargo, en muchos casos los costos de re-entrenamiento del personal técnico y de compra de piezas de software, superan los ahorros obtenidos en la compra de equipos.

   La discusión alrededor de los ahorros potenciales depende del punto de vista financiero que emplee la organización para los costos: algunas organizaciones ven los costos de entrenamiento como "costo de inicio", o "costos de una sola vez", y piensan que los costos de los equipos son recurrentes en el tiempo. En este caso, obtener ahorros en compra de equipos es siempre favorable. Pero si los costos de reentrenamiento empiezan a ser recurrentes, por fuga de personal o por crecimiento mayor al esperado, los ahorros en equipo pueden perder su atractivo.

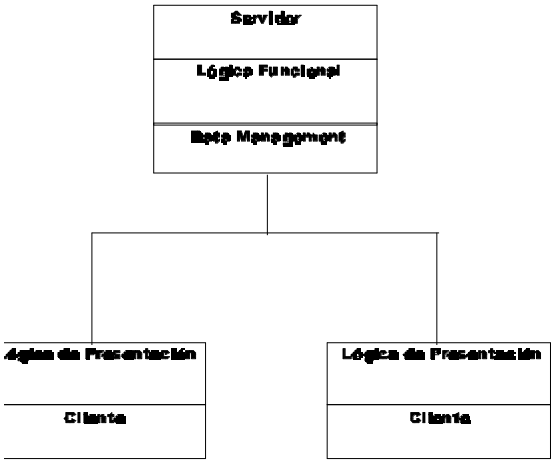
3. Muchos usuarios de la tecnología cliente-servidor mencionan que la principal razón para emplear esta tecnología es que el desarrollo de estos sistemas es más rápido y flexible. La velocidad de desarrollo resulta principalmente del uso de herramientas de sistemas abiertos de software, con interfaz gráfica, que facilitan la preparación y pruebas de prototipos de aplicaciones, permitiendo a los desarrolladores responder más rápidamente a los cambios solicitados por los usuarios. Las arquitecturas cliente-servidor fuerzan a utilizar enfoques modulares para el diseño de aplicaciones, lo cual ayuda en el mantenimiento posterior de las mismas.

4. Finalmente, un número de organizaciones pequeño pero creciente está adoptando aplicaciones cliente-servidor para tomar ventaja de nuevos y poderosos paquetes de aplicaciones que han surgido con la tendencia hacia sistemas abiertos. Prácticamente todo proveedor de software está desarrollando nuevas versiones cliente-servidor de sus productos, y muchos están empezando a instalarlas en sus clientes. Los proveedores están aprovechando la transición a la nueva arquitectura para agregar atractivas nuevas funciones a sus aplicaciones, incluyendo interfaces gráficas, interfaces para bases de datos abiertas, y herramientas para adaptar el núcleo aplicativo. Entre los ejemplos se cuentan Lotus Development Corp. con Lotus Notes, y SAP, con su paquete comercial R/3.

Aplicaciones cliente-servidor

Hace cinco años, cliente-servidor era una técnica para estructurar aplicaciones distribuidas de manera que los programas de despliegue de pantallas se ejecutaran en una PC, mientras las funciones y los datos se ejecutan y almacenan en un servidor. Hoy cliente-servidor se refiere a cualquier tecnología que soporta aplicaciones formadas por componentes distribuidos a lo largo de una red computacional, esquema frecuentemente llamado *computación distribuida*. Estas tecnologías frecuentemente pertenecen a sistemas abiertos de software.

Las aplicaciones cliente-servidor se organizan en componentes, que se instalan en diferentes procesadores a lo largo de la red. Los componentes



Componentes de la Aplicación Cliente-servidor

interactúan por medio de la red para registrar transacciones, generar facturas, entregar mensajes, o realizar otros tipos de funciones. Los principales elementos de una aplicación cliente-servidor se muestran en la Figura 3.

Una aplicación cliente-servidor típica tiene los siguientes tres componentes:

- *Lógica de presentación*: consiste de la lógica de despliegue y de interacción con el usuario.
- *Lógica funcional*: consiste de los algoritmos y las rutinas de la aplicación.
- *Administración de datos (data management)*: consiste del almacenamiento de los datos que procesa la aplicación.

A pesar que los componentes de la aplicación cliente-servidor se diseñan como componentes separados, no siempre se instalan en máquinas separadas. Por ejemplo, el analista puede optar

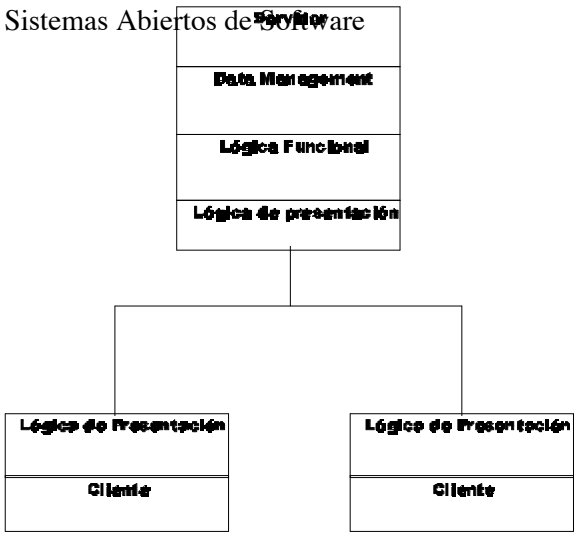


Figura 4
Modelo de Presentación Distribuida

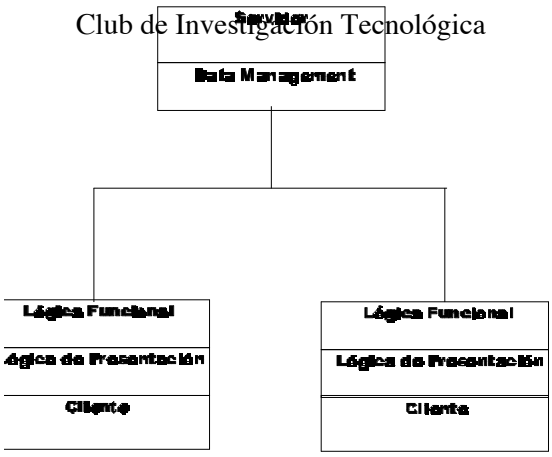
por unir los componentes de lógica funcional y administración de datos en una sola máquina, principalmente por razones de rendimiento. Este tipo de decisiones son frecuentes y dependen de los objetivos de funcionamiento de la aplicación cliente-servidor.

Los componentes de la aplicación ejecutan en los equipos de los clientes o en los equipos servidores. Los clientes hacen solicitudes y los servidores responden con servicios. Un cliente puede tener una colección de rutinas para formular consultas, y el servidor puede consistir de una base de datos. El cliente envía solicitudes de datos y el servidor responde con los grupos de datos solicitados. En algunos diseños, el cliente y el servidor son entidades físicas diferentes, en otros, son solo distinciones lógicas.

Modelos de cliente-servidor

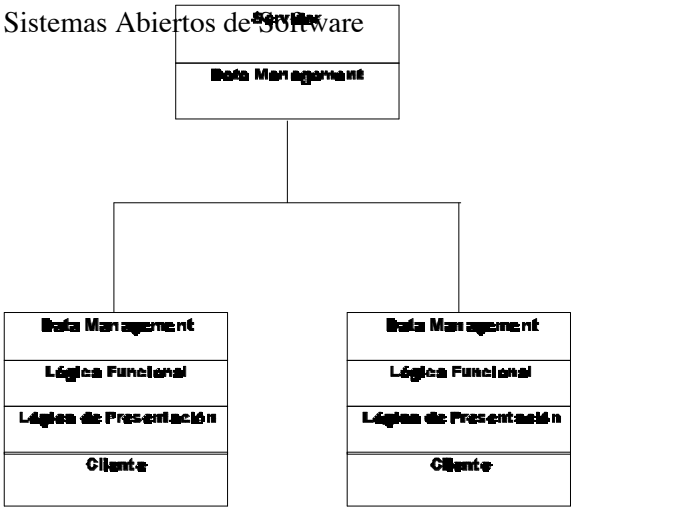
La firma de consultoría Gartner Group ha publicado un informe sobre aplicaciones cliente-servidor, en el cual identifica los siguientes cinco modelos de arquitecturas cliente-servidor:

- Presentación remota
- Presentación distribuida
- Administración de datos (data management)remota
- Lógica distribuida
- Administración de datos distribuida.



I

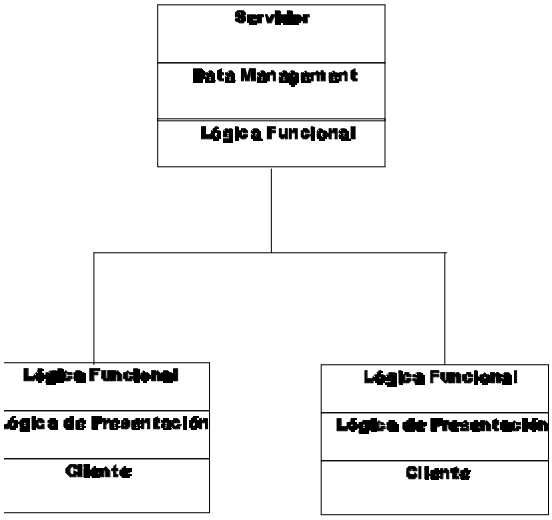
odelo de Data Management Remoto



I

Modelo de Data Management Distribuido

La Figura 3 corresponde al modelo de presentación remota del Gartner Group. Las Figura 4 a 7 muestran los modelos de Presentación distribuida, Administración de datos remota, Lógica distribuida, y



i g u r a 6
odelo de Lógica Distribuida

Administración de datos distribuida, respectivamente.

Otros modelos propuestos incluyen el modelo de la firma Patricia Seybold Group, el modelo de Servicios compartidos ("Shared services"), y el modelo de compañeros distribuidos ("Distributed peers"), los cuales no se discuten por no ser relevantes para los propósitos de este informe.

"Middleware" cliente-servidor

"Middleware" cliente-servidor es un término general que se utiliza para describir software de conectividad que apoya aplicaciones cliente-servidor.

El "middleware" se localiza "en medio" de clientes y servidores, y como tal, es una tecnología facilitadora clave para las aplicaciones cliente-servidor. Es también el componente típico de los sistemas abiertos de software para aplicaciones cliente-servidor.

Los productos de middleware comparten componentes comunes. En primera instancia, proporcionan una interfaz con la red.

En las aplicaciones cliente-servidor, la red está definida por dos protocolos. El primero de ellos es un protocolo de transporte que maneja los canales de comunicación entre componentes. Los componentes usan estos canales para intercambiar estructuras de control y de datos. Algunos ejemplos de protocolos de transporte incluyen el TCP/IP, el IPX/SPX de Novell Inc., el APPN de IBM, y el AppleTalk de Apple Computer Inc.

Algunos productos de middleware solo pueden utilizar uno de estos transportes, otros como el SQL*Net de Oracle Corp., el OpenINGRES/Net de ASK, y el Informix-Net de Informix Software, Inc., pueden usar múltiples protocolos dentro de una misma aplicación.

El segundo protocolo provee una facilidad de comunicaciones a nivel de aplicaciones para el cliente y el servidor. Esta facilidad, llamada en inglés "backplane protocol", provee convenciones para que los componentes cliente-servidor interactúen entre sí. El protocolo de comunicaciones de backplane opera por encima del protocolo de transporte.

Por ejemplo, en una aplicación cliente-servidor de bases de datos, el protocolo de backplane en el cliente empaca comandos de SQL en un formato particular, para que sean transmitidos por medio de un protocolo de transporte estándar hacia el servidor. El servidor utiliza el protocolo de transporte para recibir los formatos a través de la red, y utiliza el protocolo de backplane para desempacar los mandatos SQL. El

proceso luego se invierte para enviar la respuesta del servidor hacia el cliente.

Otra función del middleware es proporcionar las siguientes facilidades a los desarrolladores de aplicaciones:

- *Definición de elementos*: esta es una herramienta para definir nombres y características de elementos que se usan en el ambiente cliente-servidor. Por ejemplo, se utiliza para nombrar los procedimientos almacenados que se llaman en forma remota o RPC (del inglés "Remote Procedure Calls"), y los parámetros y variables que se pasan.

- *Directorio*: La función de directorio administra los nombres y direcciones de red de los componentes, y permite a un componente localizar otros en la red sin conocer su dirección física exacta.

- *Seguridad*: Controla los niveles de acceso a los componentes de las aplicaciones cliente-servidor en el ambiente middleware.

- *Servicios de aplicaciones*: un ambiente de middleware típicamente incluye un servicio que apoya las funciones de las aplicaciones. Por ejemplo, las aplicaciones de automatización de flujos, incluyen un motor que maneja el ruteo de información entre aplicaciones. El middleware de OLTP distribuidas incluye un motor para administrar transacciones.

- *Herramientas de administración*: los ambientes de middleware incluyen utilitarios para administrar y controlar los componentes.

Categorías de middleware

Hay seis tipos principales de middleware, y docenas de productos, todos los cuales cumplen hasta cierto punto con las descripciones de la sección anterior. Cada uno de los productos del mercado ha sido diseñado para satisfacer un conjunto de requerimientos específicos, por lo que es importante tener en cuenta cuál es ese requerimiento al evaluar el producto. Los productos se pueden agrupar en las siguientes categorías.

Funciones distribuidas

El middleware para funciones distribuidas permite a la aplicación cliente-servidor abarcar funciones que se ejecutan tanto en la máquina del cliente como en uno o varios servidores. Frecuentemente, el cliente y las funciones computacionales que requiere están en equipos diferentes, separadas por una red. El middleware para funciones distribuidas permite al cliente solicitar a servidores remotos compartidos que ejecuten una función computacional a su nombre, creando enlaces entre el cliente y las funciones que necesita, independientemente de la localización de las mismas en la red.

Típicamente en esta categoría de middleware, los clientes y servidores se estructuran en parejas de llamado-respuesta. Estas relaciones son predeterminadas por el programador e incorporadas en programas de acople (“stub programs”) que luego se ensamblan junto con la aplicación. En este caso, para efectuar cambios en las relaciones cliente-servidor se requiere recompilar las aplicaciones.

El middleware de funciones distribuidas incorpora ambientes de RPC, incluyendo productos basados en el DCE de OSF y el ambiente "Open Network Computing" de Sun Microsystems Inc.

Existe una variedad de herramientas para generar y administrar RPCs, entre otros: EZ-RPC de NobleNet Inc., y RPC Tool de NetWise Inc.

Esta categoría también incluye el middleware orientado a mensajes, que puede ser utilizado por una aplicación de cliente para invocar una función remota. Estos productos incluyen el MQ Series de IBM, el DECmessageQ de Digital, el Message Express de Momentum Software, y el Communications Integrator de Covia Partnership.

Bases de datos y OLTP distribuidas

El middleware de bases de datos y OLTP distribuidas permite a usuarios y/o aplicaciones leer y actualizar bases de datos que corren en máquinas remotas. El middleware enlaza clientes, que típicamente hacen solicitudes de lectura o actualización, a servidores, que típicamente ejecutan el software de base de datos que satisface estas solicitudes. El middleware está optimizado para complementar las funciones de acceso de datos (en lectura) y de integridad de datos (en actualización) de la base de datos requeridas por la aplicación. Sin embargo, el grado en que este middleware complementa la base de datos varía ampliamente.

Hay tres tipos principales de middleware de bases de datos y OLTP distribuidas, todas con los mismos objetivos de habilitar el acceso a los datos y preservar la integridad de los mismos, pero que realizan sus funciones en forma distinta:

- Los productos de acceso a bases de datos dan a los clientes acceso a una variedad de bases de datos diferentes. Todos estos productos soportan la producción de consultas e informes, algunos también permiten al cliente enviar actualizaciones diferidas en lotes. Entre otros están EDA/SQL de Information Builders Inc., OmniSQL de Sybase Inc., Information Warehouse de Red Brick Systems, y HyperStar de VMark Software Inc., así como el middleware de acceso a bases de datos de Oracle, Sybase, Informix Software Inc., Ingres, y Progress Software Corp.

- Middleware de bases de datos distribui-das permite al cliente leer y escribir simultáneamente en dos o más bases de datos físicas, como si fueran una sola base de datos uniforme. Estos productos soportan la ejecución de consultas e informes, y las operaciones de actualización. El middleware de bases de datos distri-buidas soporta dos modos de operación: "two-phase commit" y replicación. "Two-phase commit" soporta actualizaciones a múltiples bases de datos dentro de una sola transacción. Replicación sincroniza dos o más bases de datos separadas. Sybase e Ingres ofrecen servicios de replicación en sus productos.

- El middleware de OLTP distribuido soporta actualizaciones diferidas de múltiples bases de datos dentro de una sola transacción. Estos productos son independientes de los proveedores específicos de la base de datos. Como ejemplos están: el CICS/6000 de IBM, Encina de Transarc Corp., Tuxedo de Novell, y Top End de AT&T-Global Information Solutions.

  El middleware de bases de datos distribuidas ha estado disponible en el mercado por años, pero su uso no ha sido muy difundido debido a la complejidad del desarrollo de aplicaciones con estos servicios, a las dificultades de diseño de bases de datos distribuidas, y a los problemas de mantenimiento de las aplicaciones resultantes, que son más difíciles de administrar que las arquitecturas centralizadas. Los servicios de replicación son más nuevos, y están empezando a utilizarse con más frecuencia conforme la tecnología madura, debido a que ofrecen la habilidad de distribuir datos sin comprometer la robustez o el rendimiento del sistema. Replicación es una buena respuesta para las situaciones que no tienen las exigencias de sincronización inmediata que ofrece el "two- phase commit".

Archivos distribuidos

El middleware para archivos distribuidos permite al cliente el acceso a archivos que residen en otros computadores ajenos al suyo. En el caso más común, el middleware soporta el acceso del cliente al servidor de archivos, que es un repositorio de archivos que comparten muchos clientes. En este caso, el middleware intercepta las llamadas del sistema operativo al manejador de archivos local que corresponden a archivos residentes en el servidor, y los dirige al servidor.

Esta categoría incluye productos de redes de PCs como el File Service de Novell NetWare, el File Service de Windows NT Advanced Server de Microsoft, y el VINES File Service de Banyan Systems Inc. También incluye servicios para UNIX, como NFS y el DCE Distributed File Service de OSF.

Mensajería y coordinación

El propósito principal del middleware de mensajería electrónica y coordinación es apoyar el intercambio de información y de instrucciones entre usuarios y aplicaciones. Se incluyen dos tipos de middleware en esta categoría: el middleware de mensajería se dedica exclusivamente a apoyar comunicaciones efectivas y confiables. El middleware de coordinación agrega facilidades para automatizar la transferencia de información e instrucciones requeridas para ejecutar una función aplicativa. Estos dos tipos de middleware normalmente se usan en conjunto.

La categoría de middleware de mensajería y coordinación incluye productos de correo electrónico que soportan la interacción cliente-servidor, tales como: Mail Exchange de Soft Switch Enterprise, OpenMail de Hewlett-Packard Co., y Enterprise Mail Server de Microsoft.

Esta categoría también incluye productos de automatización de flujo de trabajo como FlowMark de IBM, Action Workflow de BuilderAction Technologies Inc., WorkMan de Reach, ProcessIT de AT&T-GIS, XSoft's In Concert de Xerox Corp., y Regatta de Fujitsu.

Despliegue distribuido

El middleware de despliegue distribuido permite que las funciones de presentación, o despliegue de usuario, de una aplicación sean manipuladas desde un computador remoto. Típicamente, este middleware permite al

usuario de un PC ingresar datos y mandatos en una aplicación que está ejecutando en un servidor o computador central en la red, y ver los resultados de sus acciones en su ambiente nativo. Esta categoría incluye el X Window System, que se incluye en la mayoría de los productos UNIX, para usar en PCs. Incluye además productos como Easel de Easel Corp., InfoConnect de Unisys Corp. and Digital Communications Associates Inc., Rumba de Wall Data Inc., y Extra! de Attachmate Corp. Otros productos para PCs también pertenecen a esta categoría: Carbon Copy for Windows de Microcom, Inc., y PC/Anywhere de Symantec Corp.

Computación de objetos distribuida

El middleware de computación de objetos distribuida complementa los lenguajes de programación orientados a objetos. Con este tipo de lenguajes, las aplicaciones se construyen usando paquetes auto-definidos de datos y de procedimientos, llamados objetos. Un objeto realiza operaciones sobre sus datos en respuesta a las solicitudes de otro objeto. Estas interacciones se dan entre interfaces funcionales de alto nivel: un objeto envía un mensaje solicitando ejecución a la interfaz del otro objeto. El middleware de computación de objetos distribuida permite a los objetos enviar mensajes a través de la red, sin estar limitados a un solo computador.

En la mayoría de los casos, los objetos se redireccionan a un servicio de corredor de solicitudes de objetos ("object request broker") que localiza el destino del mensaje y establece la comunicación requerida para enviar el mismo. Sin embargo, en algunos sistemas los objetos pueden enviar mensajes no a un objeto específico, sino al objeto mejor capacitado para completar la operación deseada. En este caso, el corredor de objetos debe seleccionar el tipo de objeto apropiado y localizar una instancia del mismo, para enviar el mensaje.

La categoría de middleware de objetos distribuidos incluye corredores de objetos de varias descripciones. En este grupo están incluidos el System Object Model/Distributed System Object Model de IBM, el ObjectBroker de Digital, el Distributed Object Management Facility y el Distributed SmallTalk de HP, el Project DOE de Sun, y las extensiones que Microsoft tiene planeadas para Windows NT, llamadas "Cairo". La categoría también incluye ambientes de desarrollo de objetos como XShell de Expersoft, SuiteDOME de Dome Software Corp., y Cooperative Frameworks de AT&T-GIS. Muchos de estos productos soportan la especificación de interfaz de CORBA del OMG, y los estándares relacionados.

APIs para cliente-servidor

Los productos de middleware cliente-servidor usualmente incorporan una interfaz de programación de aplicaciones (API) que da al desarrollador acceso a sus servicios. El API ofrece una forma común y bien definida para utilizar los servicios del middleware desde el código de su aplicación. Usando los llamados definidos en el API, el desarrollador puede invocar un servicio para su aplicación sin tener que preocuparse de cómo está implementado el mismo. Los APIs se pueden disponer en capas, uno sobre el otro, para aislar las aplicaciones de cambios que puedan darse en las capas inferiores. Por ejemplo, en muchos casos es posible cambiar el protocolo de transporte de la red (de TCP/IP a SPX/IPX) sin afectar el software de aplicación. Los APIs son vitales para apoyar la creación de paquetes de software, o aplicaciones a la medida, que utilizan middleware. La mayoría de estos APIs son propietarios del proveedor del middleware específico.

Muchos proveedores documentan sus APIs y los ponen a disposición de casas de software y clientes. Por ejemplo, el middleware de mensajería MQ Series de IBM se vende con un API llamado MQI. Oracle 7 se vende con un API para SQL en redes llamado el Oracle Call Interface. El DCE de OSF se entrega con un API para sus servicios de RPC, que está expresado en el IDL de DCE RPC.

Muchos de estos APIs para cliente-servidor están diseñados para satisfacer los requerimientos de un tipo específico de aplicación. Por ejemplo MQI permite ligar módulos funcionales pero no soporta funciones de correo electrónico u objetos distribuidos, sin necesidad de esfuerzos considerables. Esta limitación también está presente en la mayoría de los productos de middleware que hay hoy en el mercado: normalmente tienen un solo API para apoyar un juego de funciones específicas de computación distribuida.

Para las categorías de middleware antes mencionadas, hay al menos una organización trabajando en la definición de estándares de APIs que trascienden las interfaces propietarias de los proveedores. Un API estándar permite al desarrollador construir aplicaciones portables para diversos productos del mismo tipo de middleware. A continuación se presentan los principales estándares para APIs que están en proceso de ser definidos.

Funciones distribuidas

Los estándares principales en middleware de funciones distribuidas están asociados con

RPCs. El estándar de facto en RPCs es el XDR (del inglés "External Data Representation") de Sun, que utiliza NFS y que se incluye en prácticamente toda copia de UNIX, además de estar disponible para otras plataformas. Este API está expresado en un lenguaje de definición de interfaz.

El DCE RPC de OSF es otro estándar potencial, como API para servicios de RPC y como una implementación estándar.

El software de RPC de Microsoft para Windows NT está basado en el DCE RPC. Sin embargo, a la fecha ni Microsoft ni ningún otro implementador principal de DCE ha despachado suficientes copias de DCE para considerarlo un estándar de mercado. En este momento, DCE está disponible principalmente para UNIX.

No hay estándares para middleware de mensajes. Los proveedores en esta categoría han decidido específicamente no crear un API común para sus productos.

Bases de datos y OLTP distribuidas

Los APIs más utilizados para middleware de bases de datos y OLTP distribuidas permiten a los clientes tener acceso a servidores de bases de datos, principalmente para leer datos. Otros APIs permiten escribir datos o iniciar transacciones que resultan en actualización de la base de datos.

Todos los principales proveedores de bases de datos suministran su propio API para permitir el acceso de clientes a sus datos por medio de la red. Además, la mayoría de los proveedores tienen planes para soportar el acceso de clientes con el API ODBC (del inglés "Open Database Connectivity") de Microsoft. ODBC es una versión de un API estándar definido por un consorcio de la industria llamado el SAG (del inglés "SQL Access Group"). El ODBC de Microsoft está basado en el CLI (del inglés "Call Level Interface") de SAG. Sin embargo, ODBC ha eclipsado en importancia al CLI de SAG, y se ha convertido en un estándar tan popular, que ya está disponible para ambientes de Macintosh y OS/2, además de Windows.

El principal competidor de ODBC es Sybase con sus APIs Open Client y Open Server. Sybase tiene una base instalada importante de sus APIs, y se ha posicionado en el extremo alto del mercado. ODBC es mejor para consultas e informes desde el PC, pero en su versión actual es muy lento para soportar aplicaciones OLTP. Por el contrario, los APIs Open Client y Open Server de Sybase fueron diseñados desde su inicio para soportar aplicaciones OLTP. Sybase sostiene que hay 750 desarrolladores de paquetes de aplicaciones que utilizan su Open Server API. Sybase ha vendido 300,000 licencias de Open Client y 5,000 de Open Server.

Frecuentemente se confunde el DRDA (del inglés "Distributed Relational Database Access") de IBM como un competidor de ODBC. DRDA es un protocolo, mientras que ODBC es un API, por lo que realmente DRDA es complementario a los APIs como ODBC u Open Client.

Hay dos conjuntos de estándares surgiendo en cuanto a OLTP cliente-servidor: el Distributed Transaction Processing de X/Open, y el CICS/6000 de IBM. En 1993 este último fue liberado para varias versiones de UNIX.

Archivos distribuidos

En middleware de archivos distribuidos los APIs más importantes son el NFS de Sun y el File Service de Novell NetWare. NFS es un estándar de facto en sistemas UNIX, pero también está disponible para PCs, Macintosh, y sistemas propietarios.

El Netware File Service continúa siendo la opción dominante en redes de PCs, que incluye APIs para las principales plataformas. Windows NT Advanced Server de Microsoft es una potencial alternativa a Novell.

Mensajería y coordinación

Tanto Microsoft como Lotus ven el correo electrónico (E-mail) como la clave para penetrar el mercado del software de aplicación para grupos de trabajo, y ambas compañías están tratando de establecer su API para mensajería como el estándar para los desarrolladores independientes. Ni el Microsoft Mail API (MAPI) ni el Vendor Independent Messaging (VIM) de Lotus han obtenido todavía una posición dominante en el mercado.

Despliegue distribuido

El middleware de despliegue distribuido está sobrecargado de estándares de API. Las interfaces a los computadores centrales, como el HLLAPI (del inglés "High Level Language Application Program Interface") de IBM, están entre los estándares más rigurosos y estables de la industria. En el mundo UNIX, el API de X Windows y las bibliotecas de gráficos son estándares estables y ampliamente difundidos.

Computación de objetos distribuida

El principal estándar de API para middleware de computación de objetos distribuida es CORBA del OMG. CORBA es una especificación de interfaz que define cómo los objetos en la máquina del cliente solicitan los servicios de un corredor de solicitudes de objetos , y cómo los objetos en el servidor se relacionan con el corredor para satisfacer la solicitud del cliente. IBM, Digital, HP, Sun, AT&T-GIS, y otros proveedores ofrecen software que cumple con CORBA, para ambientes Microsoft Windows, variedades de UNIX, Macintosh, OS/2, y VMS.

CORBA no define todos los servicios requeridos para completar el ambiente. Para subsanar esto, IBM, Apple, y otros están integrando otros tecnologías con CORBA. Los casos más importantes en este momento son OpenDOC, Taligent, y OpenStep.

El OLE 2.0 (del inglés "Object Linking and Embedding") de Microsoft puede surgir como un estándar de facto en middleware de computación de objetos distribuida. Hoy OLE 2.0 es un conjunto de APIs que permiten al desarrollador construir una aplicación como un conjunto de objetos. Microsoft usa estas interfaces para soportar aplicaciones de documentos compuestos.

Sin embargo, Microsoft ha anunciado planes de producir un nuevo OLE 2.0 que soporta documentos compuestos en diferentes equipos a través de la red. La nueva versión de Windows NT estará equipada con servicios especiales para computación distribuida. Microsoft no soporta las interfaces CORBA en OLE 2.0, y no tiene planes de hacerlo. A cambio, Microsoft ha empezado a dar licencias de OLE 2.0 para que otros proveedores la conviertan a otras plataformas, con la esperanza que OLE 2.0 se convierta en un estándar multi-plataforma.

Organizaciones de estándares

Los consorcios de proveedores y las organizaciones de estándares han sido fuerzas importantes en la evolución de las tecnologías cliente-servidor de sistemas abiertos comerciales.

La OSF fue uno de los primeros al definir un ambiente multiplataforma para computación distribuida. El OMG ha sido líder en la comercia-lización de tecnología para el manejo de objetos distribuidos. X/Open tomó la iniciativa de definir estándares para OLTP distribuida. SAG identificó una interfaz estándar para acceso remoto a bases SQL. Además, la ISO ha asignado alta prioridad a la defínción de estándares de computación distribuida para los próximos años. Sin embargo, durante 1993 estas organizaciones perdieron algo de la importancia que tenían, debido a que SAG perdió relevancia ante Microsoft, y que OSF hizo cambios radicales en su misión y estructura. El Object Management Group continúa siendo una organización de vital importancia. Sin embargo, algunos proveedores miembros que están despachando productos basados en CORBA está empezando a tomar la iniciativa. Antes de esperar la aprobación del OMG para sacar una nueva tecnología, IBM, Sun, y otros, lo hacen por su propia cuenta.

Conectividad de Bases de Datos

Introducción

Los sistemas de administración de bases de datos DBMS (del inglés "Database Management Systems") permiten el almacenamiento, manipulación y administración de datos relacionados, residentes en medios magnéticos computacionales.

Los sistemas de administración de bases de datos relacionales, RDBMS, (del inglés "Relational Database Management Systems") han demostrado ser una excelente herramienta para el desarrollo de aplicaciones flexibles que responden en forma dinámica a los nuevos requerimientos de las organizaciones.

Las nuevas tendencias de sistemas abiertos y aplicaciones cliente-servidor están imponiendo exigencias cada vez mayores en los RDBMS actuales, para apoyar el diseño y construcción de aplicaciones a través de redes de computadores de diferentes marcas y tipos, bases de datos de distintos proveedores, y con lenguajes de programación y herramientas de desarrollo independientes de la base de datos. El modelo cliente-servidor y los sistemas abiertos de computación implican opciones de complejidad creciente como bases de datos distribuidas, manejo de imágenes, y computación de objetos distribuidos.

Frecuentemente una organización requiere aplicaciones que utilizan información alojada en diferentes bases de datos, en computadores disímiles. Las nuevas técnicas de redes de computación permiten la interconexión física de muchos tipos de equipos, que comparten un solo medio de comunicaciones de datos, en el cual participan tanto los clientes como los servidores. Sin embargo, la conexión física de los equipos no es suficiente para lograr que la aplicación manipule datos desde y hacia varias bases de datos en la forma transparente que se desea normalmente.

Estos requerimientos generan la necesidad de contar con herramientas que faciliten la conectividad de aplicaciones basadas en PCs con bases de datos disímiles, o geográficamente dispersas. En muchos casos, esta conectividad debe incluir sistemas de bases de datos tradicionales alojados en equipos centrales de tecnologías anteriores, que todavía almacenan gran parte de los datos corporativos.

Bases de datos relacionales y SQL

Los sistemas de bases de datos tradicionales estaban diseñados para acceder registros de datos cuyas relaciones estaban previamente definidas en términos de localización física o jerarquía. Por lo tanto, era difícil adaptar estos sistemas a los cambios de los requerimientos de las organizaciones.

Los sistemas de bases de datos relacionales, en cambio, usando referencias lógicas como nombre de departamento y número de empleado, pueden proporcionar gran flexibilidad en cuanto a opciones de acceso a los datos.

SQL (Structured Query Language) es el lenguaje estándar de manipulación de bases de datos relacionales, para acceder, leer, y actualizar los datos. La mayoría de los RDBMS soportan el dialecto SQL de ANSI básico, pero prácticamente todos los proveedores tiene sus propias extensiones de SQL para añadir opciones y hacer sus productos más funcionales y competitivos.

Estas extensiones de SQL hacen difícil desarrollar aplicaciones que sean portables entre diferentes RDBMSs. Las principales áreas de divergencia son los tipos de datos, las funciones internas, soporte a unión de tablas en bases de datos diferentes ("outer-join"), manejo de valores nulos, y vistas de tablas.

Hay dos formas principales para utilizar comandos SQL desde programas de aplicación:

1. La interfaz de llamados o CLI (del inglés "call-level interface"), que es un conjunto de funciones que realizan servicios tales como conexión a la bases de datos, ejecución de comandos SQL, recuperación de datos, cierre de la conexión a la base de datos. Cada RDBMS tiene tiene su CLI único, debido a que tiene su propio conjunto de funciones y capacidades.

2. La mayoría de los RDBMS también soportan interfaces de alto nivel o HLI (del inglés "high-level interfaces"), usando lenguajes de tercera generación como COBOL, FORTRAN, y otros. Los HLIs normalmente cumplen con el estándar de ANSI Embedded SQL. Los HLIs son más fáciles de usar porque son estandarizados y requieren menos programación, sin embargo no siempre ofrecen el mismo nivel de rendimiento que un CLI, y a menudo no soportan todas las funciones del RDBMS.

El SQL Access Group

Un grupo de proveedores de RDBMS, software, y hardware, interesados en estandarizar la API de SQL, han formado una organización llamada SQL Access

Group o SAG. Sin embargo, a la fecha ningún proveedor ha implementado completamente el estándar de SAG, pero muchos han adoptado partes del estándar y los han extendido.

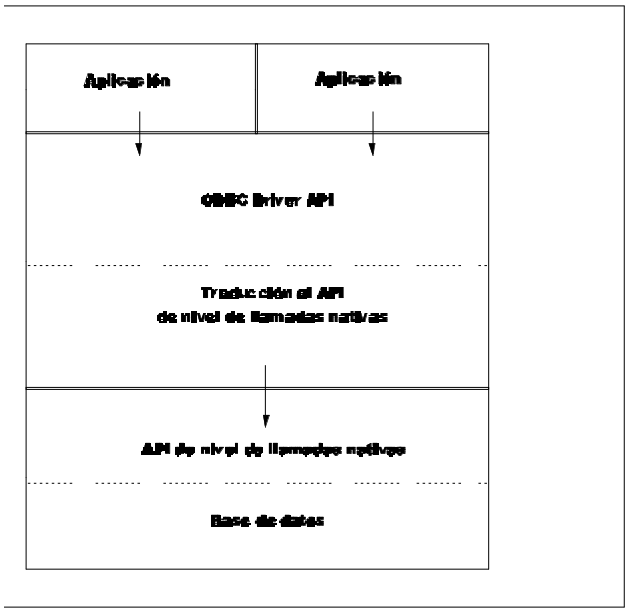
Conectividad de bases de datos

El Open Database Connectivity (ODBC) de Microsoft está basado en el CLI de SAG, pero incluye muchas extensiones. La versión más reciente, ODBC 2.0, se desvían aún más del estandar de SAG que la versión anterior, agregando funciones que permiten navegar bases de datos no relacionales sin usar SQL. Cada versión tiene tres niveles:

- El nivel central soporta un juego mínimo de funciones de CLI, tales como conexión y lectura a la base de datos.
- El nivel 1 soporta funciones adicionales como recuperar nombres de columnas de la tabla, o funciones especiales de un dispositivo específico.
- El nivel 2 soporta funciones como cursores (lectura de varios registros uno a la vez), y "browsing" (lectura de registros hacia adelante y hacia atrás).

Los "drivers" de ODBC traducen los comandos de ODBC en llamadas en la interfaz nativa del RDBMS objetivo (ver Figura 8). Muchos proveedores independientes han desarrollado traductores de ODBC: Intersolv Inc. y PageAhead Software Corp. suministran dos de los conjuntos más populares. Varios proveedores de RDBMS incluyen drivers ODBC con sus productos.

Uno de los problemas con ODBC es que cada uno de estos drivers soporta diferentes funciones de SQL CLI y se comporta diferente. Debido a que ODBC es un mínimo común denominador, los usuarios de estos drivers frecuentemente están imposibilitados de usar todas las funciones de su base de datos y experimentan degradación en el rendimiento. Dada la poca madurez actual de ODBC, la interfaz CLI nativa de la base de datos específica sigue siendo la forma más popular de construir aplicaciones.



i g u r a 8
quema de Operación de ODBC

Referencias

Price Waterhouse Technology Center. 1994. *Technology Forecast: 1995*.
Price Waterhouse World Firm Services BV. Palo Alto, California

Price Waterhouse. 1994. *International Information Technology Review 1994*.
Price Waterhouse World Firm. Londres, Inglaterra

Price Waterhouse. 1993. *International Information Technology Review 1993*.
Price Waterhouse World Firm. Londres, Inglaterra

Price Waterhouse. 1994. *State of the Application Software Market*.
Price Waterhouse LLP. New York, New York.

Datapro. 1993. *Datapro Reports on Distributed Databases 1993*.
McGraw-Hill Incorporated. Delran N.J.