Arquitecturas Distribuidas Cliente/Servidor

Dos fuerzas opuestas

- Dos fuerzas opuestas guían el desarrollo de las aplicaciones distribuidas:
  - La modularización de las aplicaciones tiende a llevar más módulos cada vez más cerca del usuario.
    - Las PC son cada vez más potentes.
    - Los usuarios quieren autonomía local y funciones específicas.
  - Los usuarios quieren acceder *toda* la información corporativa; la información es un valor de la organización: cuantos más la usen mejor se amortiza.

Dos fuerzas opuestas (2)

- Estos requisitos son casi imposibles de alcanzar en una arquitectura (de hardware *y* software) monolítica.
- La solución yace en el desarrollo de una aplicación dividida en componentes más o menos autónomos que ejecutan en unidades de hardware interconectadas por redes de alta velocidad.

Dos fuerzas opuestas (3)

- En términos más generales se busca un estilo arquitectónico que habilite tres cualidades:
  - Integrabilidad: de datos y aplicaciones.
  - Modificabilidad: de aplicaciones, de representación de los datos, de ubicación física de los componentes.
  - Escalabilidad: si la organización crece el sistema debe acompañar y permitir el crecimiento de forma transparente para las unidades que ya están en producción.
  - Todo esto en un contexto de grandes cantidades de datos y transacciones.

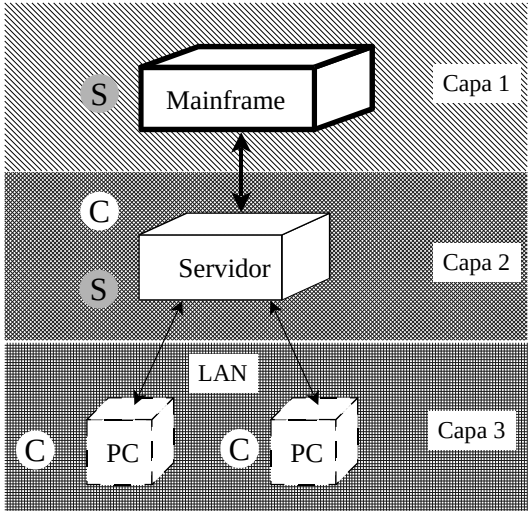
Arquitectura de 3 capas

- En la implementación más trivial de este estilo los recursos físicos están distribuidos verticalmente en tres niveles.
  - El nivel superior contiene los equipos más poderosos que administran los datos corporativos más importantes
  - El segundo nivel contiene servidores LAN que median el acceso entre los dos niveles extremos
  - El tercer nivel está formado por PCs utilizadas por los usuarios

Estructura de hardware (2)

- La estructura puede expandirse horizontalmente en cualquiera de las capas.
- Esta posibilidad implica que es posible satisfacer la cualidad de *escalabilidad*.
- También es posible expandirla verticalmente generalizando el estilo a *n*−capas.

Estructura de hardware



Mainframe: información corporativa, almacén de datos.

Servidores: uno por LAN, mediatizan todas las comunicaciones de las PCs de la LAN con la Capa 1.

La mayor parte del procesamiento propio de la aplicación se efectúa en clientes; y las tareas compartidas, en mainframes o servidores de LAN.

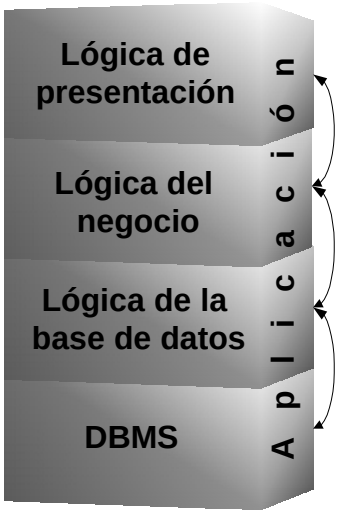
Arquitectura de *n*−capas

- Con la tecnología disponible hoy día, en principio, es simple implementar un sistema basado en este estilo.
- Sin embargo, el problema no es tecnológico sino de ingeniería (diseño):
  - ¿En qué capas se deben poner los datos?
  - ¿En qué capas se deben poner los programas?
  - ¿En qué capas se deben poner las interfaces con el usuario?

Integrabilidad y modificabilidad

- Para alcanzar estas dos cualidades se requiere:
  - Una adecuada estructuración de la aplicación
  - Mecanismos de comunicación estándar (conectores) entre los componentes de esa estructura
- Para ver la estructura general de una aplicación basada en este estilo usaremos la estructura lógica.

Estructura lógica



Lógica de presentación: formateo de pantalla, administración de ventanas, etc.

Lógica del negocio: toma datos suministrados por el usuario y/o la base de datos y efectúa los cálculos necesarios según los requerimientos del usuario.

Lógica de la base de datos: manipula grandes cantidades de información (SQL), es independiente del estrato superior pero depende del tipo de DBMS usado.

DBMS: es donde tiene lugar el verdadero procesamiento y almacenamiento de la información.

Estructura lógica (2)

- Con esta división se logra:
  - Utilizar mejor cada equipo pues se pueden correr sobre ellos programas que los utilizan más eficientemente.
  - Utilizar distintos lenguajes para programar características notablemente diferentes de la aplicación.
  - Modificabilidad de la aplicación
- Últimamente la estructura lógica se aumenta con una capa más para ruteo de transacciones.

Estructura lógica (3)

- Claramente puede usarse cualquier lenguaje para programar cualquier capa pero hay lenguajes específicos para cada capa.
  - Presentación: HTML+Javascript, Tcl/Tk, Motif
  - Negocio: Java, C, Cobol, 4gl
  - Base de datos: SQL, Cobol
  - DBMS: C
- Si cada capa se diseña con ocultación de información entonces tenemos modificabilidad.

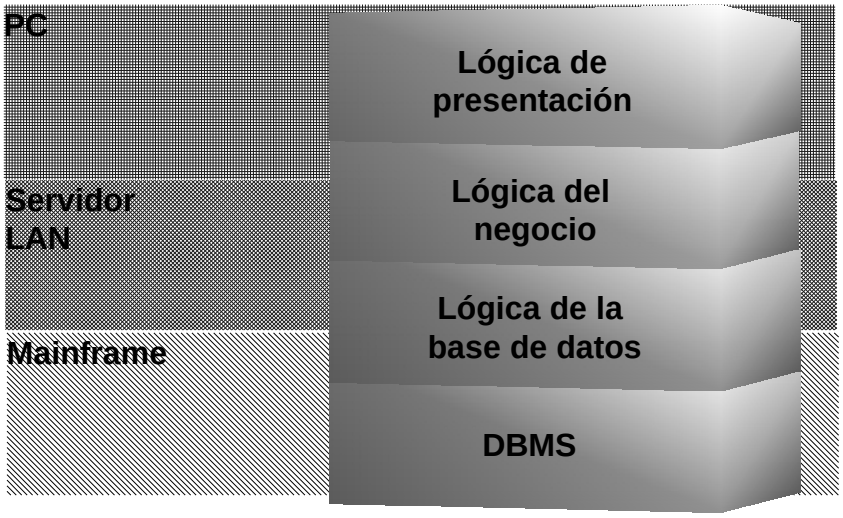
¿Qué y cómo distribuir?

- Entonces tenemos el hardware y el software distribuido en varias capas. Esto genera algunas preguntas:
  - ¿Qué capas del software deben distribuirse en las distintas capas de hardware?
  - ¿Qué ventajas y desventajas tiene una distribución sobre la otra?
  - ¿Cómo deben comunicarse los componentes en cada capa?

¿Cómo distribuir?

- En general la lógica de presentación con las facilidades para E/S se ubica en los clientes los cuales yacen en la capa 3.
- Si las PC clientes tienen el poder de cómputo suficiente, algunas porciones de la lógica de negocio también pueden residir en estos equipos. Por ejemplo, las validaciones simples de los datos de entrada. También aquellas porciones de la lógica de negocio que son específicas de los usuarios que usan cada PC.

Estructura física



¿Cómo distribuir? (2)

- Incluso parte de la lógica de la base de datos puede ubicarse en un cliente si los datos son cuasi−estáticos, o de baja interacción y están fuertemente relacionados con la actividad de los usuarios de esos clientes.
- Si hay un único servidor en la capa 2, los fragmentos comunes a todos los usuarios de la lógica de negocio deben ubicarse en él. La lógica de la base de datos también debe ponerse en este servidor (si el DBMS lo

¿Cómo distribuir? (3)

- Si el DBMS o la lógica de la base no son únicos, en capa 2 podría ponerse un *middle−ware* para ruteo de transacciones.
- Si hay más de un servidor en capa 2, y si estos sirven a grupos disjuntos de PCs, entonces la lógica de negocio debe dividirse entre estos servidores.
  - Más aun puede darse el caso que cada división de la lógica de negocio use distintos DBMS en cuyo caso la lógica de la base también se divide.

¿Cómo distribuir? (4)

- Una combinación de los dos casos anteriores se resuelve convenientemente agregando una nueva capa entre la 1 y la 2 con uno o varios servidores que corran *middle−ware* para ruteo de transacciones.
- Por cuestiones de eficiencia puede ser posible o necesario poner parte de la lógica de negocio en el DBMS (procedimientos almacenados).

¿Cómo distribuir? (5)

- EL DBMS usualmente se ubica en capa 1. Este puede estar constituido por un RDBMS y archivos planos a nivel del SO.
- El RDBMS puede ser nativo en el SO (DB2) o una aplicación (Oracle).
- EL DBMS puede estar centralizado o distribuido, puede hacer replicaciones, soportar varios protocolos de comunicación, hacer balanceo de carga, etc.

¿Cómo distribuir? (6)

- El criterio general para distribuir debe basarse en lo siguiente:
  - La cantidad de datos relevantes a cada aplicación
  - El número de usuarios activos que ejecutan aplicaciones contra esos datos
  - La cantidad de interacciones entre los diferentes componentes de la aplicación
  - Las características técnicas de las plataformas (hardware, SO, red, etc.)

Presentación distribuida

- Separa la presentación del resto del sistema tiene numerosas ventajas:
  - El procesamiento de gráficos de alta resolución requiere mucho poder de cómputo. Si el servidor hiciera esto para muchos clientes se estaría haciendo un uso ineficiente de las PC y del servidor
  - Hardware específico en las PC puede posibilitar aplicaciones impensadas
  - Se utilizan lenguajes específicos para programar
  - Se puede cambiar el hardware cliente sin tener que tocar la lógica de negocio.

Procesamiento distribuido

- Procesamiento de aplicación (negocio+base de datos) reside en el cliente.
  - Precio/perfomance de PCs sobre mainframes, no es necesaria sincronización entre fragmentos de la aplicación, menor tráfico entre presentación y procesamiento.
  - Múltiples copias del código, desaprovecha los mainframes, sobrecarga las PC.

Procesamiento distribuido (2)

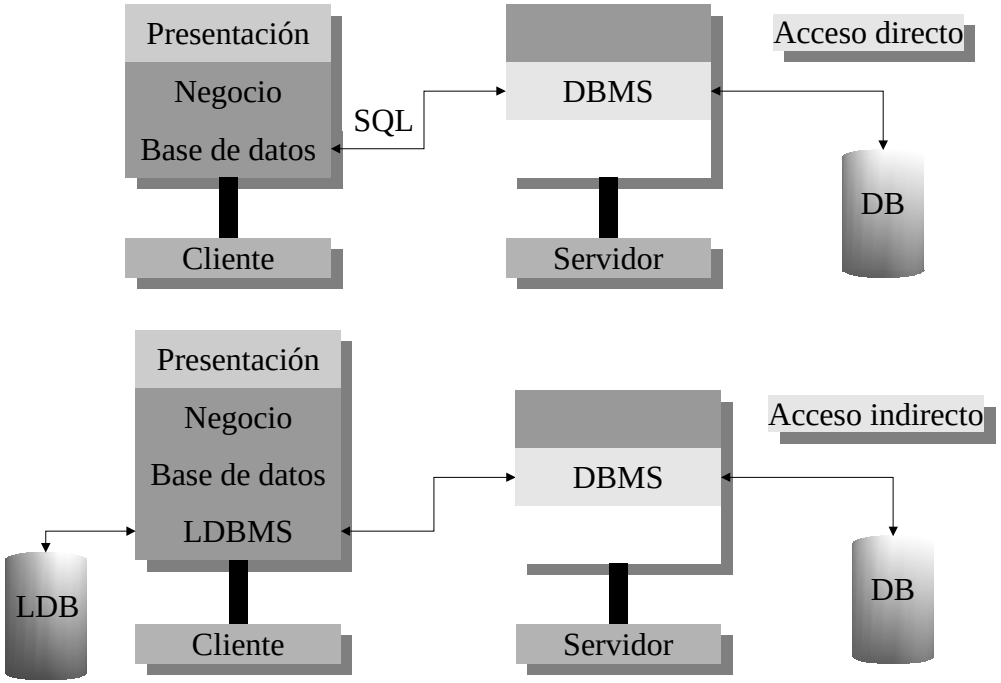
- Procesamiento de aplicación reside en el servidor
  - Sincronización ídem anterior, simplifica mantenimiento del código, reduce tráfico negocio−base de datos.
  - Desaprovecha PCs, incrementa tráfico presentación−negocio, sobrecarga los servidores

Procesamiento distribuido (3)

- El procesamiento de la aplicación está dividido entre clientes y servidor.
  - Los fragmentos comunes se instalan cerca de la base, los relacionados con la presentación cerca de los clientes. En teoría es posible eliminar las desventajas de los dos modelos anteriores y combinar las ventajas
  - Sin embargo, no es simple desde el punto de vista de la ingeniería (diseño) y requiere una cuidadosa planificación de interfaces, módulos, etc.

Datos distribuidos

- ¿En cuál o cuáles capas físicas deben ubicarse los datos?
- ¿Qué datos deben ubicarse en cada capa?
- Los datos compartidos deben almacenarse en servidores de bases de datos corporativos, los datos que con específicos de una porción de la aplicación o de un cliente deben almacenarse en bases de datos locales.
- Estos criterios dan lugar a dos estructuras.
  - Integridad (administración distribuida de transac.), heterogeneidad(ruteadores de transacciones).



Conectores

- HTTP y otros protocolos estándar que permitan enviar y recibir datos entre procesos remotos.
  - XML sobre HTTP
- Tubos: mecanismo tipo *connection−oriented* que provee facilidades sólo para determinar los límites de los paquetes, la identidad del remitente, y para verificar la llegada del paquete.

Conectores (2)

- Interacción SQL tipo cliente/servidor: mecanismo tipo *connection−oriented* que permite pasar un pedido SQL y sus datos asociados desde un proceso a otro.
  - Impone severas restricciones a los protocolos y formatos utilizados por los clientes.
    - Sintaxis, funcionalidad y formatos de datos soportados
  - Sólo sirve contra RDBMS.
- RPC y *sockets*