03 - Procesos en un sistema distribuido

IIC2523 - Sistemas Distribuidos

Cristian Ruz – cruz@ing.puc.cl

Departamento de Ciencia de la Computación
Pontificia Universidad Católica de Chile

Semestre 2-2020

Contenidos

Contenidos

Virtualización

¿Para qué?

Virtualización

¿Para qué?

Importancia de virtualizar

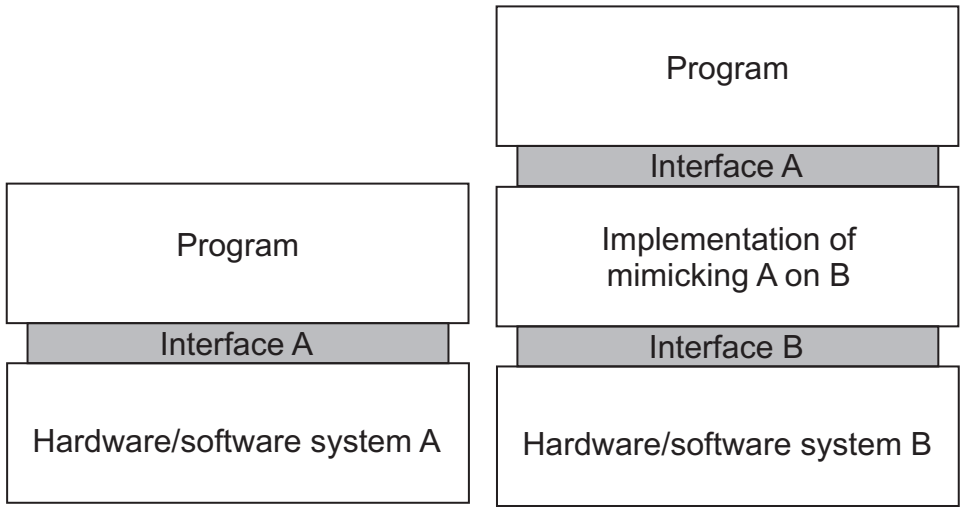
- El *hardware* evoluciona rápidamente
- Necesidad de migrar código, y portabilidad
- Aislamiento de componentes con fallas o bajo ataques

Virtualización

¿Para qué?

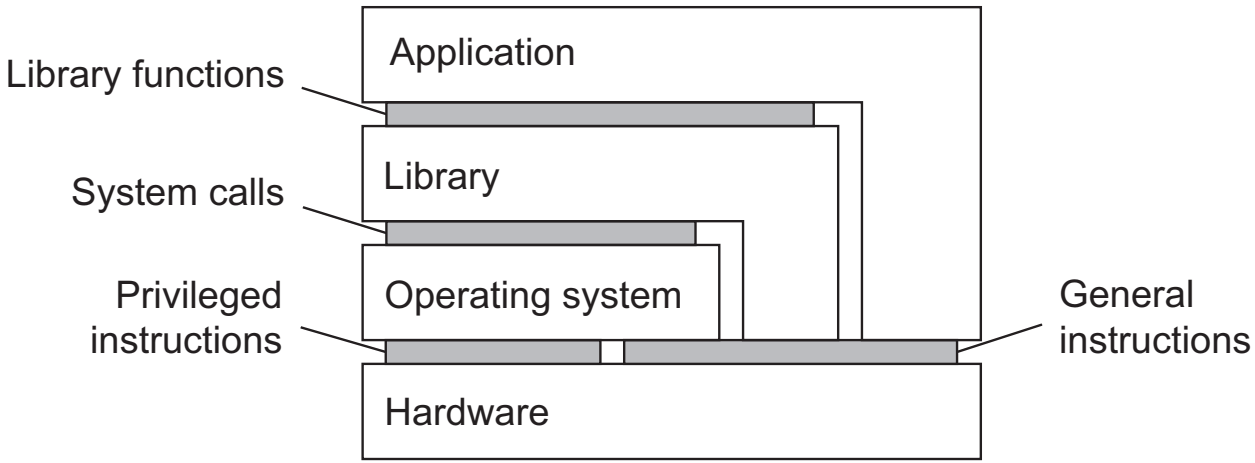
Importancia de virtualizar

- El *hardware* evoluciona rápidamente
- Necesidad de migrar código, y portabilidad
- Aislamiento de componentes con fallas o bajo ataques



Virtualización

Podemos virtualizar a distintos niveles

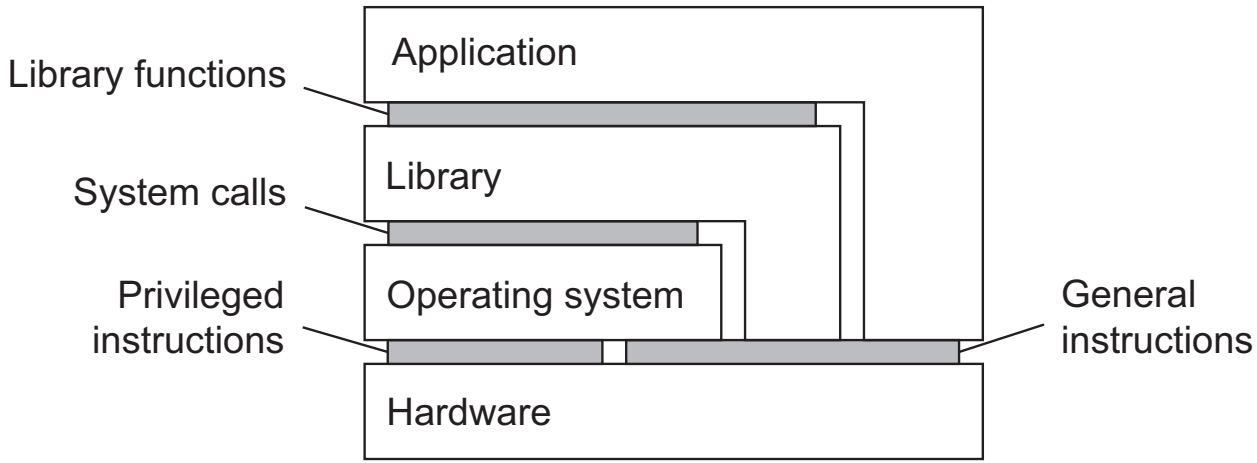


Virtualización

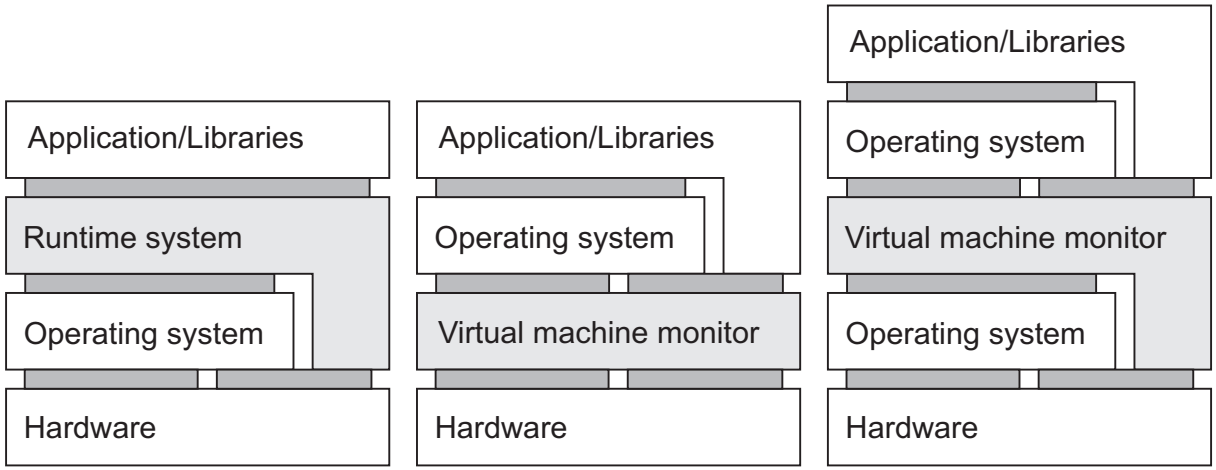
Podemos virtualizar a distintos niveles

¿Qué virtualizar?

- **ISA**: *Instruction Set Architecture*. El conjunto de instrucciones de la máquina.
  - Instrucciones privilegiadas (las que solo puede ejecutar el sistema operativo)
  - Instrucciones no-privilegiadas (las que puede ejecutar cualquier programa)
- **Llamadas al sistema**. Interfaz provista por el sistema operativo.
- **Funciones de biblioteca**. Conjunto conocido como API.



Virtualización



- **Process VM**. Emulación para un proceso.
- **Native VMM**. Instrucciones de bajo nivel. Sistema operativo mínimo (interfaz).
  - Implementado directamente sobre el *hardware*.
  - Diferentes SO *guest* en la misma plataforma.
- **Hosted VMM**. Instrucciones de bajo nivel, delegando llamadas a sistema operativo completo.
  - VMM corriendo sobre SO *host*, con nivel especial de privilegios

¿Siempre se puede virtualizar?

Condición necesaria para virtualizar

For any conventional computer, a virtual machine monitor may be constructed if the set of sensitive instructions for that computer is a subset of the set of privileged instructions.

Problema: Puede haber instrucciones "sensibles" que se ejecutan en *user mode*, y que no generan *traps*.

Instrucciones "sensibles" pueden afectar el comportamiento del sistema operativo.

¿Siempre se puede virtualizar?

Condición necesaria para virtualizar

For any conventional computer, a virtual machine monitor may be constructed if the set of sensitive instructions for that computer is a subset of the set of privileged instructions.

Problema: Puede haber instrucciones "sensibles" que se ejecutan en *user mode*, y que no generan *traps*.

Soluciones

- Emular todas las instrucciones. *Full Virtualization*
- *Wrap* instrucciones sensible no-privilegiadas para que sean manejado por el VMM (solución de VMWare)
- *Paravirtualización*. Modificar el sistema *guest* para garantizar la semántica de todas las operaciones (solución de Xen)

Aplicaciones de virtualización

No podríamos tener *Cloud Computing* sin virtualización

IaaS: *Infrastructure-as-a-Service*

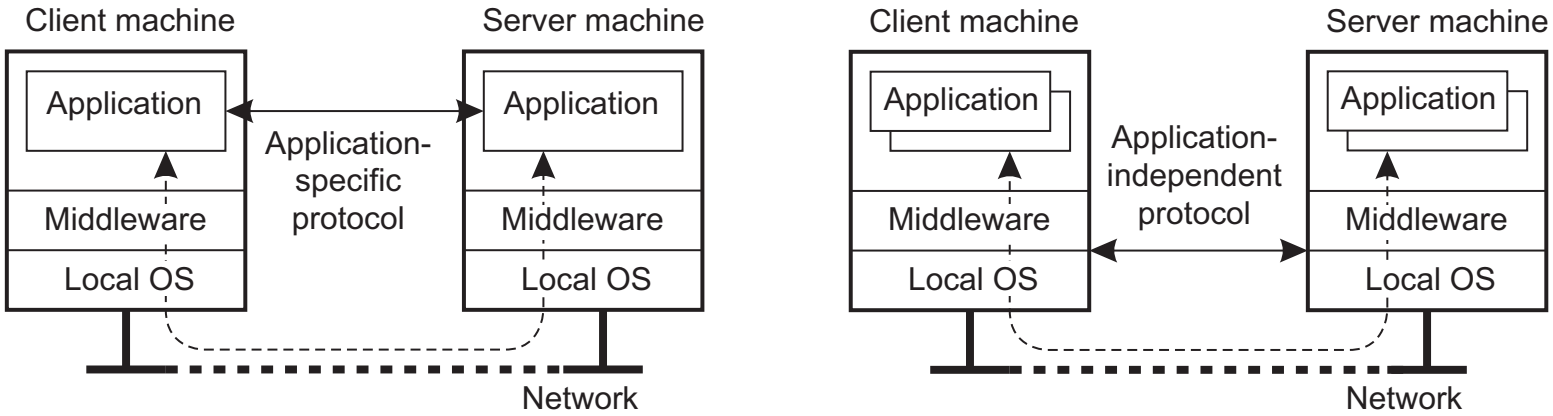
"Arriendo" de máquinas virtuales que podrían esta usando la misma máquina física.

- Aislamiento (*isolation*) casi completo entre clientes.
- Rendimiento menor a aislamiento real

Contenidos

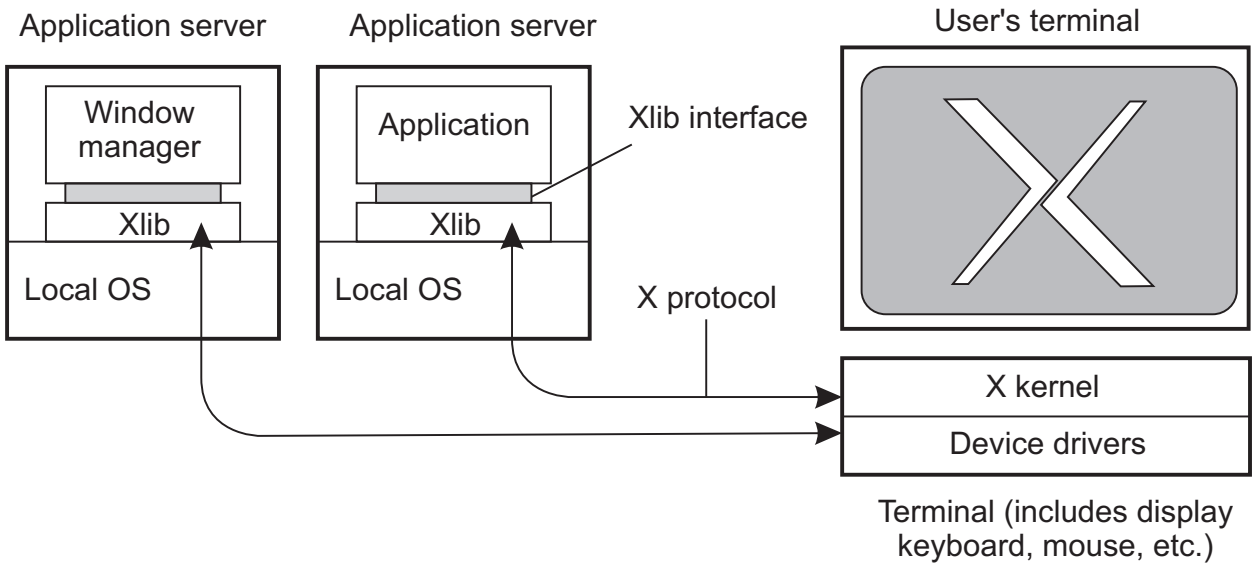
Clientes

Conexión a interfaces remotas



- Aplicaciones pueden conectarse a una contraparte (par) remota mediante protocolos específicos
- Clientes que sólo utilizan un interfaz: **thin clients**

Ejemplo: X Window System

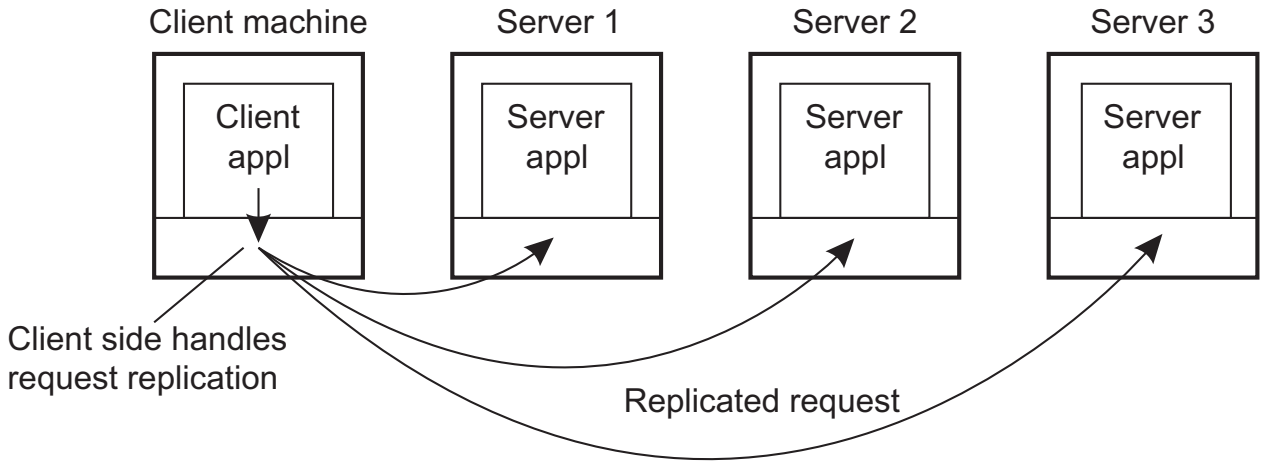


- Aplicaciones son **clientes**, que se comunican con el **X-kernel** (servidor)
- Pueden estar en máquinas distintas
- Aplicaciones suelen esperar comportamiento síncrono
- Delegar el dibujado completamente al cliente: **Virtual Network Computing** (VNC)

Transparencia en clientes

Aspectos de transparencia se delegan a clientes

- **Transparencia de acceso**: mediante *stubs*/*proxys*
- **Transparencia de ubicación/migración**: mantener registro de ubicación en cliente
- **Transparencia de replicación**: *stub* transforma una invocación en múltiples invocaciones
- **Transparencia a fallas**: ocultan fallas en comunicación mediante caches y *retries*



Contenidos

Servidores

¿Qué hace un servidor?

Servidores

¿Qué hace un servidor?

Modelo básico

Proceso que implementa un servicio específico para un conjunto de clientes.
Un servidor **espera** una solicitud de un cliente (*request*), se encarga de que esta solicitud sea atendida, y luego espera por la siguiente.

Servidores

¿Cómo maneja las solicitudes?

Servidores

¿Cómo maneja las solicitudes?

Dos maneras básicas

- **Servidor iterativo**: atiende una solicitud completa antes de atender la siguiente
- **Servidor concurrente**: utiliza un *dispatcher* que delega la atención a un *thread* ó proceso distinto.

Servidores

¿Cómo maneja las solicitudes?

Dos maneras básicas

- **Servidor iterativo**: atiende una solicitud completa antes de atender la siguiente
- **Servidor concurrente**: utiliza un *dispatcher* que delega la atención a un *thread* ó proceso distinto.

¿Cuál es más común?

Servidores

¿Cómo encuentro a un servidor?

Servidores

¿Cómo encuentro a un servidor?

Mediante un puerto específico

ftp-data	20	File Transfer [Default Data]
ftp	21	File Transfer [Control]
telnet	23	Telnet
smtp	25	Simple Mail Transfer
www	80	Web (HTTP)

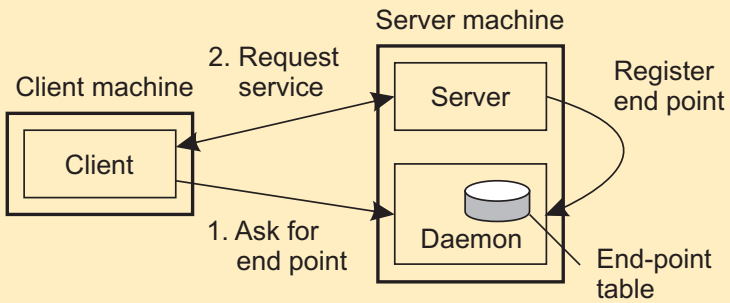
Servidores

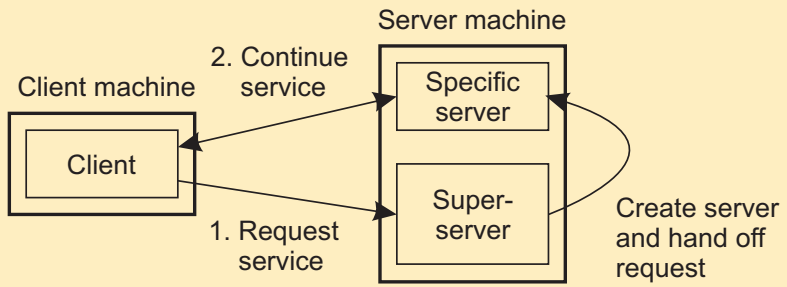
¿Cómo encuentro a un servidor?

Mediante un puerto específico

ftp-data	20	File Transfer [Default Data]
ftp	21	File Transfer [Control]
telnet	23	Telnet
smtp	25	Simple Mail Transfer
www	80	Web (HTTP)

Mediante *endpoints* dinámicos





Servidores

¿Cómo interrumpo a un servidor que está procesando una solicitud?

Servidores

¿Cómo interrumpo a un servidor que está procesando una solicitud?

Mediante un puerto específico para datos urgentes

- Servidor crear un *thread*/proceso para mensajes urgentes
- Solicitudes actuales quedan *en pausa* mientras se atiende la urgente
- Requiere *scheduling* con prioridad en el sistema operativo

Servidores

¿Cómo interrumpo a un servidor que está procesando una solicitud?

Mediante un puerto específico para datos urgentes

- Servidor crear un *thread*/proceso para mensajes urgentes
- Solicitudes actuales quedan *en pausa* mientras se atiende la urgente
- Requiere *scheduling* con prioridad en el sistema operativo

Usando la capa de transporte

- TCP permite mensaje *Out-of-band* (OOB)
- Implementación de TCP en *kernel* puede capturar estos mensajes, y delegar

Servidores

¿Qué deben saber los servidores de los clientes?

Servidores

¿Qué deben saber los servidores de los clientes?

Servidor *stateless*

No almacenan información del cliente luego de atender la solicitud.

- Abren y cierra archivos.
- No se preocupan de actualizaciones de datos en caché de cliente
- Cada conexión es un cliente nuevo

Servidores

¿Qué deben saber los servidores de los clientes?

Servidor *stateless*

No almacenan información del cliente luego de atender la solicitud.

- Abren y cierra archivos.
- No se preocupan de actualizaciones de datos en caché de cliente
- Cada conexión es un cliente nuevo

Consecuencias

- Cliente y servidores independientes
- Menor riesgo de inconsistencias de estado
- Rendimiento: servidor no puede anticipar solicitudes

Servidores

¿Qué deben saber los servidores de los clientes?

Servidor *stateless*

No almacenan información del cliente luego de atender la solicitud.

- Abren y cierra archivos.
- No se preocupan de actualizaciones de datos en caché de cliente
- Cada conexión es un cliente nuevo

Consecuencias

- Cliente y servidores independientes
- Menor riesgo de inconsistencias de estado
- Rendimiento: servidor no puede anticipar solicitudes

También existe el **soft-state**

Servidores

¿Qué deben saber los servidores de los clientes?

Servidores

¿Qué deben saber los servidores de los clientes?

Servidor *stateful*

Mantiene estado de sus clientes

- Sabe qué archivos han sido utilizado. Puede hacer prefetching
- Sabe qué datos han sido *cacheados* por el cliente.

Servidores

¿Qué deben saber los servidores de los clientes?

Servidor *stateful*

Mantiene estado de sus clientes

- Sabe qué archivos han sido utilizado. Puede hacer prefetching
- Sabe qué datos han sido *cacheados* por el cliente.

Consecuencias

- Rendimiento: puede ser alta si se organizan bien las copias locales.
- Escalabilidad: almacenar estado de todos los clientes permanentemente puede ser costoso.
- Ante una falla, el servidor debe recuperar su estado. El problema de *state recovery* no es fácil.

Servidores

¿Qué deben saber los servidores de los clientes?

Servidor *stateful*

Mantiene estado de sus clientes

- Sabe qué archivos han sido utilizado. Puede hacer prefetching
- Sabe qué datos han sido *cacheados* por el cliente.

Consecuencias

- Rendimiento: puede ser alta si se organizan bien las copias locales.
- Escalabilidad: almacenar estado de todos los clientes permanentemente puede ser costoso.
- Ante una falla, el servidor debe recuperar su estado. El problema de *state recovery* no es fácil.

¿**Session state** ó permanent state?

Servidores

¿Qué deben saber los servidores de los clientes?

Servidor *stateful*

Mantiene estado de sus clientes

- Sabe qué archivos han sido utilizado. Puede hacer prefetching
- Sabe qué datos han sido *cacheados* por el cliente.

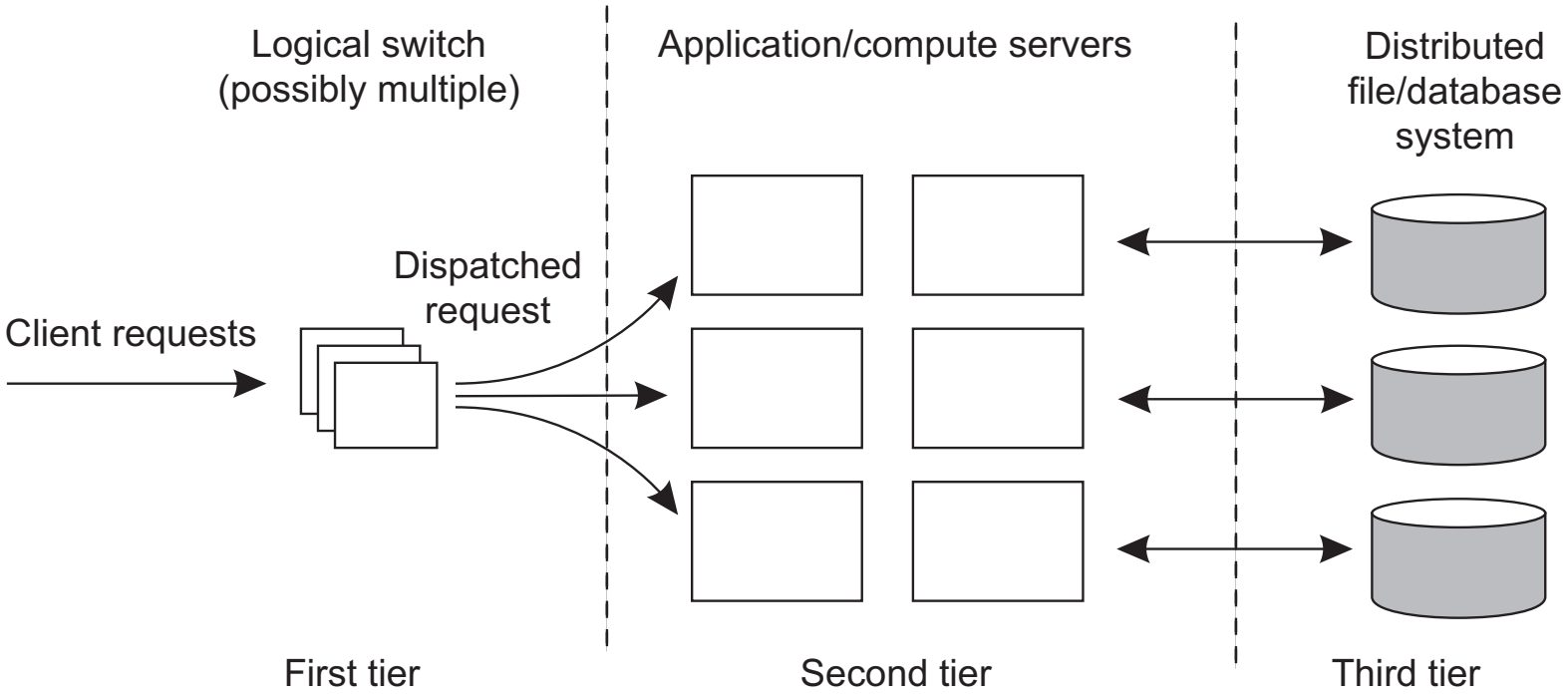
Consecuencias

- Rendimiento: puede ser alta si se organizan bien las copias locales.
- Escalabilidad: almacenar estado de todos los clientes permanentemente puede ser costoso.
- Ante una falla, el servidor debe recuperar su estado. El problema de *state recovery* no es fácil.

¿**Session state** ó permanent state?
Clientes con identificación: **cookies**

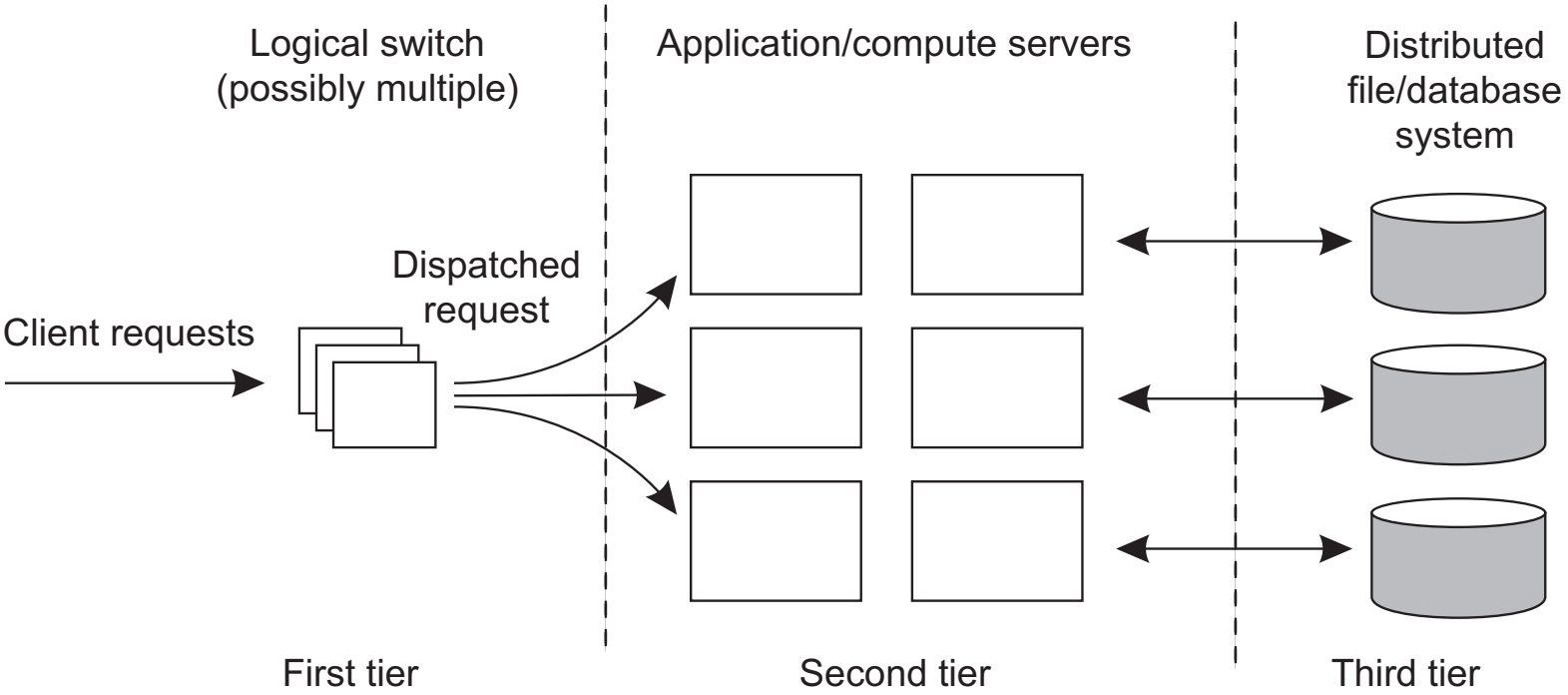
Clusters de servidores

Arquitectura típica de *cluster*



Clusters de servidores

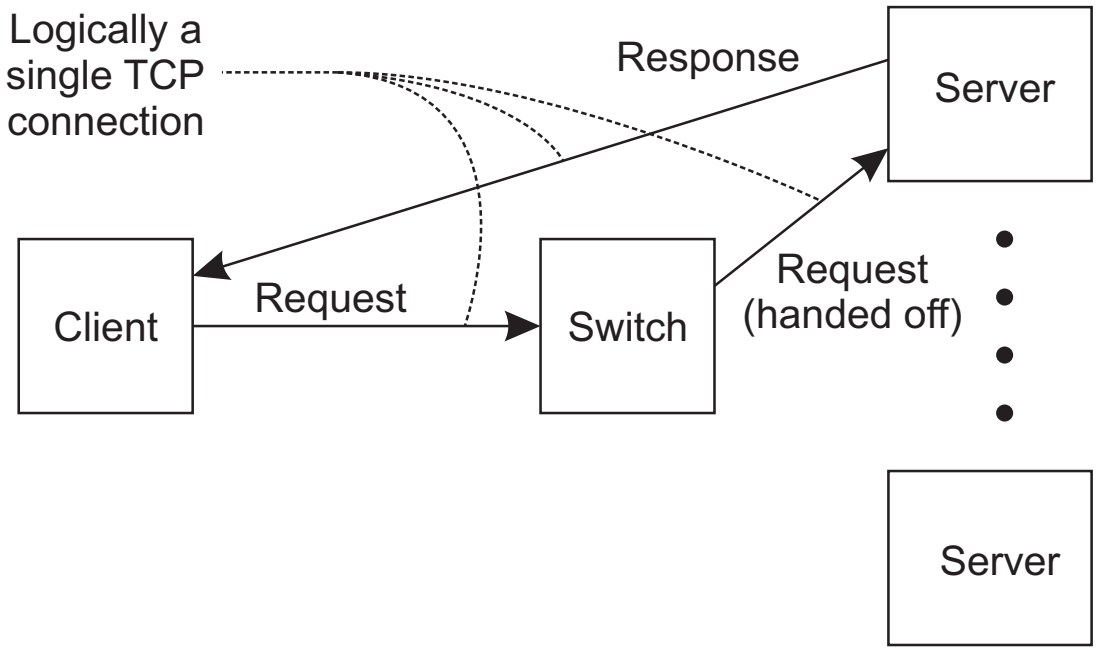
Arquitectura típica de *cluster*



Request dispatching debe ser eficiente

Clusters de servidores

Front-end puede transformarse en un cuello de botella

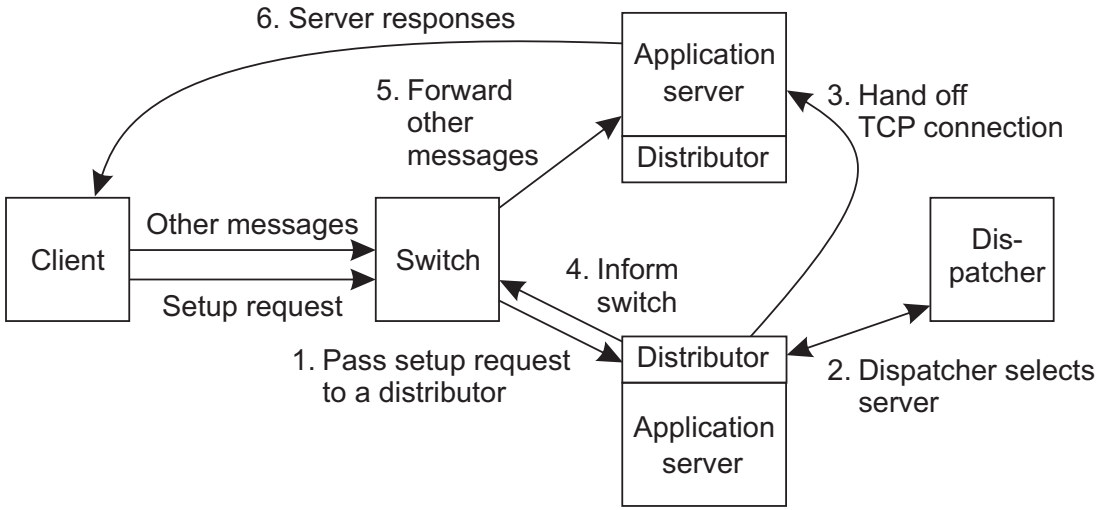


TCP *handoff*. Implementado por **transport-layer switch**

Clusters de servidores

Para alivianar el trabajo del *front-end*

- **Transport-layer switching**: utiliza TCP entre distintos servidores de acuerdo a política de rendimiento
- **Content-aware distribution**: lee contenido de la solicitud y elige el mejor servidor



Clusters de servidores

¿Podemos manejar servidores distribuidos remotamente (Internet)?

Clusters de servidores

¿Podemos manejar servidores distribuidos remotamente (Internet)?

Dispatching

Usando DNS para proveer transparencia de ubicación

1. Cliente busca servicio vía DNS
2. Servidores DNS mantiene registro de **réplicas** para el servicio, y entrega la más cercano (o con algún otro criterio)