Sistemas Operativos Distribuidos

Fiabilidad y Seguridad

Fallos

Víctor Robles Forcada Fernando Pérez Costoya
Francisco Rosales García José María Peña Sánchez

Conceptos Básicos

- Diversos elementos de un sistema distribuido pueden fallar:
  - Procesadores, red, dispositivos, software, etc.
- Tipos de fallos:
  - Transitorios: Falla una vez y luego funciona correctamente
  - Intermitentes: El fallo aparece de forma intermitente
  - Permanentes: Una vez falla el elemento, ya no se recupera
- Soluciones basadas generalmente en replicación

Víctor Robles Forcada Fernando Pérez Costoya
Francisco Rosales García José María Peña Sánchez

Clasificación de fallos

De acuerdo al funcionamiento del componente con fallo:

- *Fallo de parada*: el componente que falla se para y no interfiere en el resto del sistema
- *Fallos por omisión*: fallo que causa que un componente no responda a parte de su cometido. Por ejemplo, un canal de comunicación puede presentar fallos por omisión de envío y recepción
- *Fallos de temporización (rendimiento)*: no se cumple el rendimiento esperado, el componente responde demasiado tarde
- *Fallos de respuesta*: El elemento responde incorrectamente a las peticiones
- *Fallos bizantinos (arbitrario)*: comportamiento arbitrario y malicioso. El elemento falla de forma descontrolada

Víctor Robles Forcada Fernando Pérez Costoya
Francisco Rosales García José María Peña Sánchez

Ejemplos de fallos

- Procesador:
  - Fallo parada
  - Fallos bizantinos
- Red de comunicación:
  - Puede tener todos los tipos de fallos
- En el reloj:
  - Fallo de respuesta. El reloj se adelanta o se atrasa
- Dispositivos de almacenamiento:
  - Fallo parada (no se puede leer ni escribir)
  - Fallos por omisión: algunos datos son inaccesibles
  - Fallos bizantinos: datos corruptos

Replicación

Replicación

- Disponer de copias de datos, servicios o recursos
- Objetivos
  - Mejorar el rendimiento
  - Mejorar la disponibilidad
  - Tolerancia a fallos
- Requisitos
  - Transparencia
  - Coherencia
  - Rendimiento

Sistema Replicado

- El elemento a replicar se suele denominar *objeto*
- Un objeto está compuesto por una serie de copias físicas denominadas *réplicas*
- Un sistema replicado debe contemplar:
  - Modelo de Sistema: Cómo funcionan cada uno de los *gestores de réplicas*
  - Comunicación del Sistema: Intercambio de información entre los gestores de replicas. Muy útil mecanismos *multicast*

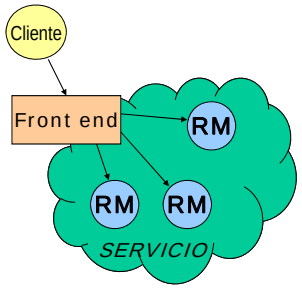
Modelo de Sistema Replicado

Cada gestor de réplicas debe garantizar las propiedades de una transacción dentro de sus datos asociados

Características:
- Funcionamiento determinista
- Recuperabilidad
- Conjunto estático o dinámico

Fases de una petición:
- Recepción (*front-end*)
- Coordinación
- Ejecución
- Acuerdo
- Respuesta

} Todos los RMs



RM: Gestor de réplicas (*Replica Manager*)

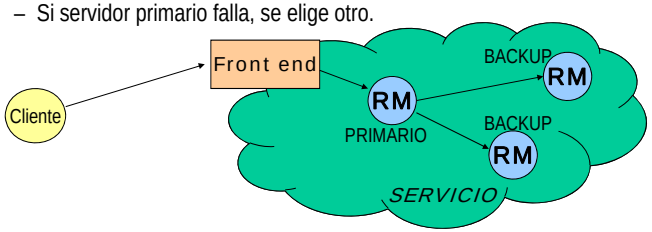
Mantenimiento de las Réplicas

Coherencia de réplicas:
- Débil: operaciones pueden devolver valor obsoleto
  - Ejemplos: DNS y NIS de Sun
- Estricta: operaciones devuelven siempre valor actualizado
  - Menos eficiente (compromiso coherencia-eficiencia)

Posibles esquemas:
- Replicación con copia primaria
- Replicación con copias simétricas

Esquema con Copia Primaria

Existe una copia primaria y copias secundarias:
- Lecturas sobre cualquier copia
- Actualizaciones sobre la primaria que las propaga a secundarias
  - Dependiendo del método de propagación, la coherencia será débil o estricta.
  - Alternativa: avisar a secundarias para que lean nueva versión.
- Si servidor primario falla, se elige otro.



Esquema con Copia Primaria

- Recepción:
  - El *front-end* recibe la petición
- Coordinación:
  - El RM primario recibe la petición, la procesa de forma atómica
  - Si ya la ha ejecutado (ID único ya procesado), re-envía la respuesta
- Ejecución:
  - El RM primario ejecuta y almacena el resultado
- Acuerdo:
  - Si la petición es una actualización se notifica a los RM secundarios
- Respuesta:
  - El RM primario responde al *front-end*

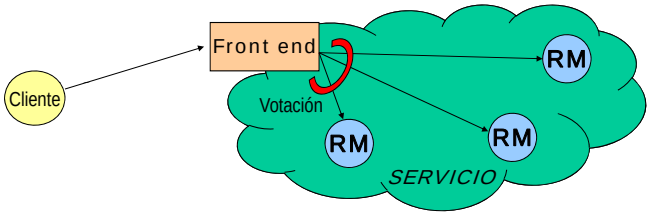
Esquema con Copia Primaria

Discusión:
- Vale para servicios no deterministas
- Si hay N servidores el servicio sobrevive a N-1 fallos
- No soporta fallos *bizantinos* (en el primario)
- El servicio requiere múltiples comunicaciones
- Permite descargar los procesos de sólo-lectura a los *backups*

Ejemplo:
- Servicio de NIS

Esquema con Copias Simétricas

Esquema de votación:
- N réplicas: cada una con un número de versión
- Lectura requiere quórum
- Escritura requiere quórum



Esquema con Copia Simétricas

- Recepción:
  - El *front-end* recibe la petición y re-envía (*multicast*) al grupo de RM
- Coordinación:
  - Del modelo de comunicación se requiere transmisión fiable y en orden
- Ejecución:
  - Se procesa la misma petición en cada RM
- Acuerdo:
  - Debido a la semántica *multicast* no se requiere fase de acuerdo
- Respuesta:
  - Se evalúan todas las respuestas recogidas por el *front-end*

Esquema con Copia Simétricas

Discusión:
- La suposición de un *multicast* fiable y ordenado es clave
- Un sistema de 2N+1 elementos soporta hasta N fallos bizantinos
- Se relajan las necesidades del sistema en relación a la ordenación (operaciones conmutables), pero se compromete la consistencia
- Permite descargar los procesos de solo-lectura a una replica

Seguridad

Conceptos de Protección y Seguridad

- Protección: Evitar que se haga un uso indebido de los recursos que están dentro del ámbito del SO. Mecanismos y políticas que aseguren que los usuarios sólo acceden a sus propios recursos (archivos, zonas de memoria, etc.)
- Seguridad: Es un concepto mucho más amplio y está dirigida a cuatro requisitos básicos:
  - *Confidencialidad*: La información sólo es accesible por las partes autorizadas
  - *Integridad*: Los contenidos sólo podrán modificarse por las partes autorizadas
  - *Disponibilidad*: Los componentes de un sistema informático sólo están disponibles por las partes autorizadas
  - *Autenticación*: Capacidad de verificar la identidad de los usuarios

Tipos de Peligros

- *Interrupción*: Se destruye un componente del sistema o se encuentra no disponible o utilizable (ataque a la disponibilidad)
  - Destrucción de disco duro, eliminación del sistema gestor de ficheros
- *Intercepción*: Ataque contra la confidencialidad
  - Escucha de canal de comunicaciones, copia ilícita de programas
- *Modificación*: Capacidad de modificar un componente (ataque hacia la integridad)
  - Alterar un programa, modificar el contenido de los mensajes
- *Fabricación*: un elemento no autorizado inserta objetos extraños en un sistema (ataque contra la autenticación)
  - Inclusión de un registro en un fichero, inserción de mensajes en una red

Problemas de Seguridad

Elemento	*Confidencialidad*	*Integridad*	*Disponibilidad*
Hardware	Robado Copiado	Destruido Pinchado Falsificado	Sobrecargado Fallido Robado Destruido No disponible
Software	Robado Copiado Pirateado	Caballo de Troya Modificado Falsificado	Borrado Mal instalado Caducado
Datos	Descubiertos Espiados Inferidos	Dañados Error HW Error SW Error usuario	Borrados Mal instalados Destruidos

Problemas de Seguridad en Instalaciones Informáticas

Ataques en líneas de comunicación

- *Ataques pasivos*: espionaje o monitorización de transmisiones
  - Lectura del contenido de los mensajes
  - Análisis de tráfico (cifrado)
- *Ataques activos*: modificaciones o creaciones de flujos de datos
  - Enmascaramiento: Un elemento se hace pasar por otro diferente
  - Reenvío: Captura de una unidad de datos y su posterior retransmisión
  - Modificación de mensajes: Se altera parte de un mensaje válido
  - Denegación de servicio: Previene o imposibilita el uso normal o la gestión de las instalaciones de comunicación

Problemas de Seguridad (I)

- Uso indebido o malicioso de programas
  - *Caballo de Troya (Troyanos)*: Programa que hace cosas no autorizadas
    - Programa de *login* modificado, editor de texto que hace copias a otros directorios
  - *Puerta trasera*: Crear un agujero de seguridad al sistema a través de un programa privilegiado
    - Identificación reservada a un programa, parches a un compilador para que añada código no autorizado
- Usuarios inexpertos o descuidados
- Usuarios no autorizados (intrusos)
- Virus: Es un pequeño programa capaz de reproducirse a sí mismo, infectando cualquier tipo de archivo ejecutable, sin conocimiento del usuario

Problemas de Seguridad (II)

- Gusanos: Es un código maligno cuya principal misión es reenviarse a sí mismo.
  - No afectan a la información de los sitios que contagian
  - Consumen amplios recursos de los sistemas y los usan para infectar a otros equipos
- Rompedores de sistemas de protección: Programas que tratan de romper la seguridad para ejecutar accesos ilegales (Satan)
- Ataques de denegación de servicio: Bombardeo masivo con peticiones de servicio. Los atacantes se enmascaran con identidades de otros usuarios (*spoofing*)
- *Phising*: Es la capacidad de duplicar una página web para hacer creer al visitante que se encuentra en la página original en lugar de en la copiada

Cifrado

Cifrado

- Cifrado de la información con clave K_C:
  - C(Info, K_C) → InfoCif.
  - Notación típica: $\{Info\}K_C$
- Descifrado con clave K_D:
  - D(InfoCif, K_D) → Info. original
- 2 alternativas:
  - sistemas de clave secreta
  - sistemas de clave pública

Tipos de Sistemas de Cifrado

- Sistemas de clave secreta:
  - $K_C = K_D$
  - Emisor y receptor comparten la clave
  - Ejemplo: DES
  - Problema de la distribución de la clave
- Sistemas de clave pública:
  - $K_C \neq K_D$
  - K_D es la clave secreta del servidor
  - K_C es la clave pública del servidor
  - K_D no puede deducirse a partir de K_C
  - Menos eficientes que sistemas de clave secreta
  - Ejemplo: RSA

Kerberos

- Sistemas de autenticación desarrollado en M.I.T. (mediados 80)
  - Basado en sistemas de clave secreta
- Usado en muchos sistemas; AFS, RPC de Sun, Windows 2000
- Autenticación de cliente y servidor basado en tercer componente
  - Reside en máquina "segura"
  - Conoce claves secretas de clientes y servidores

Kerberos

Características:

- Claves de clientes y servidores no se transmiten por la red
- Claves no se almacenan mucho tiempo en clientes
- *Timestamps* para detectar retransmisiones maliciosas
- Claves con plazo de expiración
- Permite autenticación entre dominios
- Puede usarse también para cifrar los datos

Tickets y *Autenticadores*

- Objetos básicos en la autenticación
- *Ticket*:
  - Registro que el cliente incluye en el mensaje que permite a un servidor verificar su identidad
  - Está cifrado con clave del servidor e incluye entre otros:
    - identidad del cliente, clave para la sesión, plazo de expiración
- *Autenticador*:
  - Registro que el cliente incluye en el mensaje que asegura que el mensaje se ha generado recientemente y no se ha modificado
  - Está cifrado con clave de sesión e incluye entre otros:
    - tiempo actual y *checksum*

Componentes

- 2 componentes implementados normalmente como una entidad
- Servidor de autenticación (AS):
  - Proporciona el ticket inicial:
    - Ticket de concesión de tickets (TGT)
  - Normalmente se solicita TGT en el *login* del usuario
  - El TGT se usa para comunicarse con el TGS
- Servidor de concesión de tickets (TGS)
  - Cuando un cliente necesita comunicarse con un nuevo servidor solicita al TGS, usando un TGT, un ticket para identificarse
- AS serviría para obtener tickets para identificarse ante cualquier servidor pero normalmente sólo se usa para comunicarse con TGS

Protocolo con AS

- Secuencia de mensajes (C cliente;S servidor):
  - C→AS:
    - {C,S,...} sin cifrar
  - AS genera una clave se sesión aleatoria K_{SES} y envía mensaje:
    - $\{K_{SES}, S, Texpiración,...\}K_C$
    - {Ticket} K_S que contiene $\{K_{SES}, C, Texpiración,...\}K_S$
  - C→S: envía mensaje con ticket y autenticador:
    - {Ticket} K_S + {marca de tiempo, checksum,...} K_{SES}
    - Servidor comprueba que no se ha modificado mensaje y que la marca de tiempo es reciente
  - S→C:
    - {marca de tiempo} K_{SES}
- Si se precisa integridad de datos pueden cifrarse con K_{SES}

Necesidad del TGS

- AS es suficiente pero genera un problema de seguridad:
  - Clave del cliente (generada normalmente a partir de contraseña) disponible todo el tiempo en nodo cliente
  - O se pide contraseña periódicamente a usuario o se almacena en la máquina
  - Ambas soluciones inadecuadas
- Solución: Uso de TGS
  - En *login* se obtiene de AS el ticket para TGS (TGT)
  - *Tickets* para nuevos servidores se piden a TGS
  - TGT es diferente en cada *login* y tiene una vida corta

Protocolo Completo

- Protocolo en *login* (ya analizado):
  - C solicita a AS ticket para comunicarse con TGS (TGT)
  - AS devuelve $\{TGT\}K_{TGS}$ y clave de sesión con TGS (K_{SESTGS})
- Protocolo con TGS: C quiere autenticación con nuevo servidor S
  - C→TGS: solicita ticket para identificarse ante S
    - $\{TGT\}K_{TGS}$ + {marca de tiempo, checksum,...} K_{SESTGS} +{S}
  - TGS→C envía mensaje con clave de sesión y ticket:
    - $\{K_{SES}, S, Texpiración,...\}K_{SESTGS}$ + {Ticket} K_S
  - C→S: envía mensaje con ticket y autenticador:
    - {Ticket} K_S + {marca de tiempo, checksum,...} K_{SES}
  - S→C:
    - {marca de tiempo} K_{SES}