Tema 4:

Coda

Coda: Introducción

- Intenta solucionar distintos requisitos que AFS no cumple
  - Limitación de la replicación a volúmenes de sólo lectura sería un problema para la escalabilidad de tableros de anuncios bases de datos, etc.
  - Los fallos o interrupciones programadas de servidores y componentes de red causan molestias a los usuarios durante minutos u horas.
    - En una implementación grande esto es frecuente que ocurran algún tipo de problema que puede producir estos problemas
  - La aparición de dispositivos portátiles hace que sea necesario poder trabajar sin conexión.

- Coda aborda un diseño cuyo principio fundamental es la ***disponibilidad constante de los datos***

Coda: Arquitectura

- Utiliza la misma terminología que AFS
  - Vice → Servidor
    - `Gestores de réplicas`
  - Venus → Cliente
    - `Frontal: Hacen transparente el sistema distribuido a los usuarios`
    - `Gestores de réplicas: ya que gestionan las copias de la cache`
- El conjunto de servidores que mantiene las réplicas de un volumen de archivos se conoce como ***grupo de almacenamiento del volumen (volume storage group, VSG)***
- Un cliente que quiera acceder a un fichero en dicho volumen accederá al ***grupo de almacenamiento disponible (available VSG, AVSG)***
  - Sus miembros variarán según los servidores estén o no accesibles

Coda: Arquitectura

- Como en AFS a los clientes se les notifica los cambios en los ficheros utilizando el mecanismo de *promesa de devolución de llamada*, pero se añade un servicio de distribución de actualizaciones a cada réplica
  - Al cerrar un cliente un fichero las copias las difunden a todos los servidores AVSG

- Cuando AVSG está vacío se dice que se trabaja sin conexión
  - Fallo de red
  - Caída servidores
  - Desconexión del cliente → portátiles y dispositivos móviles

Coda: Arquitectura

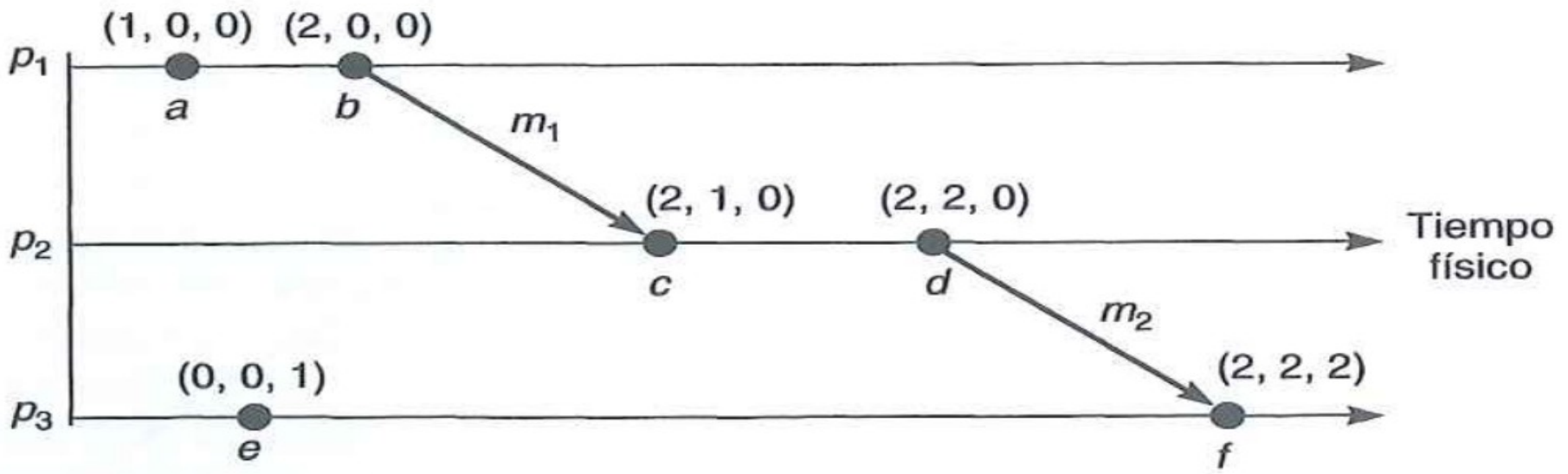
- Las copias del cliente se validan periódicamente cuando hay conexión
- Para trabajar sin conexión se necesita tener en la caché todos los ficheros necesarios para trabajar
  - Se almacena un ***histórico de la lista de archivos utilizados mientras el cliente estaba conectado*** y usa como base para predecir que ficheros se utilizarán
- Cuando la conexión restablece se comprueba la validez y se armonizan las copias de los clientes con las de los servidores
    - `En el peor de los casos esto puede requerir alguna intervención manual de los usuarios para resolver conflictos`
- Se apoya en la inclusión en cada *versión* de un fichero de un ***vector de versiones Coda (Coda version vector, CVV)***.
  - Cada elemento del vector contiene una estimación del número de modificaciones de la versión de ese servidor
  - Las réplicas con números menores deberán ser actualizadas con réplicas más nuevas

Coda: Estrategia de replicación

- Relojes vectoriales
  - Reglas de actualización
    - RV1: Inicialmente, $V_i[j]=0$ para i,j = 1,...,N procesos o servidores
    - RV2: Justo antes de que el servidor pi coloque una marca de tiempo en un fichero, coloca $V_i[i] = V_i[i] + 1$
    - RV3: El servidor p_i incluye el valor $t = V_i$ en los mensajes que mantiene con los otros servidores
    - RV4: Cuando el servidor p_i recibe una marca de tiempo en un mensaje establece Vi[j] = max(Vi[j], t[j]), para j = 1,2,...,N. Esta mezcla toma el valor máximo entre los dos componente a componente

Coda: Estrategia de replicación

- Relojes vectoriales
  - Comparación de vectores
    - `V = V' si y sólo si V[j] = V'[j] para j = 1,2,...,N`
    - `V <= V' si y sólo si V[j] <= V'[j] para j = 1,2,...,N`
    - `V < V' si y sólo si V <= V' ^ V != V'`
  - Dada una versión en el servidor "*e*" y otra en el "*c*"
    - "e" es anterior a "c" si y sólo si $V_e <= V_c$
    - "e" es concurrente con "c" si y sólo si no se cumple que $V_e <= V_c$ y que $V_c <= V_e$



Coda: Estrategia de replicación

- Cuando no se cumple que $V_e <= V_c$ o que $V_c <= V_e$ para dos CVV entonces hay un conflicto
  - Cada réplica refleja una actualización que la otra no refleja
  - Acciones:
    - El fichero se señala como ***inapropiado***
    - Se informa del conflicto al propietario
- Cuando se cierra un fichero modificado
  - Venus envía un mensaje de actualización a cada sitio en el AVSG
    - Contiene CVV nuevo
    - Nuevos contenidos del fichero
  - Cada VICE de AVSG comprueba si el CVV es mayor y si es así
    - Actualiza los datos del fichero
    - Devuelve acuse de recibo positivo
  - Venus actualiza el CVV con los servidores que responden y manda el nuevo CVV a los AVSG

Coda: Estrategia de replicación

- Como el mensaje se envía a AVSG y no al VSG no todos los servidores con replicas recibirán el CVV
- Todo CVV contendrá un contador de modificaciones exacto sólo para el servidor local, el resto tendrá contadores más bajos
- Los archivos en un volumen replicado son accesibles a cualquier cliente que pueda acceder al menos a uno de ellos.
- El rendimiento puede mejorarse compartiendo la carga de peticiones entre diferentes replicas

Coda: Estrategia de replicación

- Ejemplo
  - Considere una secuencia de modificaciones a un archivo F en un volumen que está replicado en tres servidores: S1, S2 y S3. El VSG para F es {S1, S2, S3}. F es modificado, aproximadamente, al mismo tiempo por dos clientes C1 y C2. Debido a un fallo en la red C1 puede acceder sólo a S1 y S2 (C1(AVSG)={S1, S2}) y C2 sólo a S3 (C2(AVSG)={S3})
    1. Inicialmente los CVV para los tres servidores son [1,1,1]
    2. C1 ejecuta un proceso que abre, modifica y cierra F.
       - Venus de C1 difunde el mensaje de actualización a su AVSG, {S1, S2}, con un CVV [2,2,2]
       - S1 y S2 modifican F y devuelven una confirmación
       - Venus modifica el CVV a [2,2,1]
    3. C2 ejecuta dos procesos, cada uno de los cuales abre, modifica y cierra F.
       - Venus de C2 tras cada modificación difunde un mensaje de actualización a su AVSG, {S3}, con un CVV [2,2,2] y [3,3,3]
       - S3 modifica F devuelve sendas confirmaciones
       - Venus modifica el CVV a [1,1,3]

Coda: Estrategia de replicación

- Ejemplo

  4. Algún tiempo después, se restablece la red y Venus C2 realiza un chequeo rutinario para ver si los Servidores no accesibles lo están ya, y ve que S1 y S2 lo están.
     - Modifica su AVSG a {S1, S2, S3} para el volumen que contiene F
     - Solicita los CVV de F que tienen los miembros del nuevo AVSG.
     - Venus de C2 descubre que S1 y S2 tiene un CVV [2,2,1] y S3 un CVV [1,1,3]
     - Marca el fichero F como **inapropiado**
     - Solicita la intervención manual del propietario de F

- Y si el paso 3 no hubiera existido?

Coda: Semántica de actualización

- Siendo
  - T tiempo máximo de validad de una copia de la cache sin notificación
  - lastest(F, AVSG, 0) última copia de F valida en la cache
  - lastest(F, AVSG, T) copia de la cache con T segundos de permanencia sin verificación.
  - lostCallback(AVSG, T) perdida de alguna devolución de llamada de actualización en los últimos T segundos
  - conflict(F, AVSG) los valores de F están en conflicto en los AVSG
- La semántica de actualización de CODA es:
  - tras un `open` con éxito:

(AVSG ≠ Ø y (lastest(F,AVSG,0) o (lastest(F,AVSG,T) y lostCallback(AVSG,T) y inCache(F)))) o (AVSG = Ø y inCaché(F))

  - tras un `open` fallido:

(AVSG ≠ Ø y conflict(F, AVSG)) o (AVSG = Ø y !inCaché(F))

  - tras un `close` con éxito:

(AVSG ≠ Ø y updated(F, AVSG)) o (AVSG = Ø)

  - Tras un `close` fallido:

AVSG ≠ Ø y conflict(F, AVSG)

Coda: Acceso a las réplicas

- open
  - Cuando el Cliente solicita un fichero F que no está en la caché elige un Servidor dentro de AVSG de la siguiente forma
    - Elección aleatoria
    - Basada en criterios de rendimiento basados en proximidad física, carga, etc
  - Se solicita los atributos y el fichero al Servidor elegido
  - Cuando se recibe el fichero se verifica que es la misma copia que hay en todos los AVSG
    - Si no es,
      - se elige el Servidor con la última cópia
      - se traen los datos de el
      - se informa a los AVSG que algunos Servidores poseen réplicas anticuadas

Coda: Acceso a las réplicas

- close
  - Tras una modificación y cierre de un fichero sus atributos y contenidos se transmiten en paralelo a todos los AVSG.
    - Al no incluir a VSG pueden haber servidores que no se actualicen
  - Se deja en manos de los clientes la propagación de los cambios a los servidores.
    - Los servidores sólo se ven implicados cuando se descubre una réplica atrasada en un open
- Promesa de devolución de llamada
  - La promesa de devolución de llamada se mantiene en los servidores
    - Si el servidor S1 elegido por un cliente C1 para coger la copia de un fichero F no está el AVSG de otro cliente C2 que modifique ese fichero, el servidor S1 no será actualizado y no podrá emitir sobre C1 una devolución de llamada.

Coda: Coherencia caché

- Cada Venus debe detectar los siguientes eventos dentro de los T segundos siguientes a que hayan ocurrido:
  - Aumento del tamaño de un AVSG, debido a que un servidor inaccesible se ha vuelto accesible
  - Reducción del tamaño de un AVSG, debido a que un servidor se vuelve inaccesible
  - Pérdida de un evento de devolución de llamada
- Para conseguir esto se hace lo siguiente
  - Cada T segundos Venus envía un mensaje de sondeo a los VSG de los ficheros que tiene en cache
    - Si Venus recibe una respuesta por un servidor previamente inaccesible
      - Aumenta el AVSG
      - Retira las promesas de devolución de llamada de cualquier fichero del volumen en cuestión perteneciente al nuevo AVSG
        - Esto se hace ya que al aumentar el AVSG la copia de la cache puede no coincidir con la última versión disponible en el nuevo AVSG
    - Si Venus no recibe respuesta de un servidor previamente accesible
      - Reduce el AVSG
      - Si ese servidor es el elegido de alguna de las copias en cache se retiran de ese servidor todas las promesas de devolución de llamada
      - Si una de las respuestas indica que un mensaje de devolución de llamada fue enviado pero no atendido, se retira la promesa de devolución de llamada del fichero correspondiente

Coda: Coherencia caché

- Como se trata el caso de actualizaciones perdidas por un servidor por no estar en el AVSG de otro cliente que realiza la actualización
  - En respuesta de los mensajes de sondeo de Venus, los servidores mandan un *vector de versiones del volumen o CVV del volumen.*
  - El CVV del volumen contiene un resumen de los CVV de todos los ficheros del volumen.
  - Si Venus detecta diferencias en los CVV del volumen devueltos quiere decir que el AVSG hay archivos que no está al día.
  - Aunque en la cache puede haber fichero de ese volumen que no hayan sido modificados, Venus elimina la promesa de devolución de llamada a todos los ficheros que tenga en la caché de ese volumen.
- Reflexión:
  - Venus sondea a los servidores en los VSG de los archivos que tiene en la cache.
  - Un simple mensaje sonda sirve para actualizar los AVSG y comprobar las devoluciones de llamada para todos los archivos en el volumen.
  - Teniendo en cuanta que T es relativamente grande (10 minutos en versión experimental), estos mensajes no suponen un problema para el crecimiento en cliente, servidores y redes de área amplia

Coda: Operación sin conexión

- La política de reemplazo de los ficheros de la cache menos recientemente usados puede permitir trabajar desconectado sólo un breve espacio de tiempo
  - Coda permite a los usuarios especificar una lista priorizada de ficheros y directorios que Venus debería esforzarse por mantener en la caché.
    - Los objetos en el nivel más alto se identifican como ***pegadizos*** y deben ser retenido en memoria caché en todo momento
  - Coda también proporciona una herramienta para detectar los archivos que se acceden con más frecuencia y marcarlos con una prioridad determinada
- Cuando se restablece la conexión Coda empieza el proceso de ***reintegración*** para cada fichero modificado.
  - Venus ejecuta una secuencia de operaciones de actualización de las réplicas de los AVSG
  - Si hay conflictos, la copia de la cache del cliente se almacena en el servidor y se informa al usuario que inició la reintegración
  - Las copias temporales se almacenan en un ***covolumen*** asociado a cada volumen del servidor.
    - Para estas copias se requiere poco almacenamiento.

Coda: Rendimiento

- Bajo cargas de referencia diseñadas para simular poblaciones de usuarios en un rango de 5 a 50 usuarios AFS típicos
  - Sin replicación, no hay diferencias significativas entre el rendimiento de AFS y Coda
  - Con triple replicación y
    - 5 usuarios, el tiempo que necesita Coda es sólo un 5% mayor que AFS sin replicación
    - 50 usuarios, el tiempo de Coda aumenta un 70% frente al 16% de AFS sin replicación.
  - Estas diferencias son debidas al mantenimiento de las réplicas

Coda: Reflexiones

- Coda utiliza CVV para detectar conflictos escritura-escritura sin tener en cuenta la semántica de los datos.
  - Los cliente podrían haber modificado diferentes objetos de un fichero de un modo compatible pero sería imposible realizar una fusión automática.
  - Necesita de la intervención humana ya que se supone que el sistema desconoce la semántica de los datos
- La semántica relativamente simple de los directorios hace posible la automatización de resolución de conflictos por Coda
  - Los únicos cambios que pueden hacerse en un directorio son:
    - Insetar una entrada
    - Eliminar una entrada
  - Coda fusiona directamente el estado de los directorios en conflicto