

## Tema 8 - Transacciones

Felipe Ortega, Gorka Guardiola

GSyC, ETSIT. URJC.

Sistemas Distribuidos (SD)

15 de diciembre, 2020





(cc) 2008- Grupo de Sistemas y Comunicaciones.,  
Algunos derechos reservados. Este trabajo se entrega bajo la licencia  
Creative Commons Reconocimiento - NoComercial - SinObraDerivada  
(by-nc-nd). Para obtener la licencia completa, véase  
<https://creativecommons.org/licenses/by-nc-nd/3.0/es/>.

# Contenidos

8.1 Concepto

8.2 Propiedades básicas

8.3 Two phase commit

8.4 Teorema CAP

Referencias

## 8.1 Concepto

# Transacción

- ▶ Conjunto de operaciones que se realiza de forma conjunta.
- ▶ Transacción **atómica**: se ejecuta todo o nada.
- ▶ Ejemplo: Muevo 20 euros de una cuenta a otra.
  - ▶ O sucede o no, pero no puedo crear dinero.
- ▶ En una máquina es fácil, serializable, sincronizable.

# Transacciones distribuidas

- ▶ Realización de transacciones.
- ▶ En entornos que involucran a varias (muchas) máquinas.
- ▶ Fallos complejos distribuidos, mucho más difíciles de trazar.

## Ejemplos de primitivas para transacciones

- ▶ Requieren primitivas de soporte, proporcionadas por el sistema operativo subyacente, o bien por el lenguaje de ejecución.
- ▶ La siguiente tabla muestra algunos ejemplos de este tipo de primitivas [1].

Primitiva	Descripción
BEGIN_TRANSACTION	Indica el inicio de una transacción.
END_TRANSACTION	Finaliza la transacción e intenta un <i>commit</i>
ABORT_TRANSACTION	Aborta la transacción y restaura los antiguos valores
READ	Lee datos de un archivo, tabla u otro soporte
WRITE	Escribe datos en un archivo, tabla u otro soporte

# Invariantes

- ▶ Propiedades que se deben cumplir.
- ▶ Por ejemplo: la cantidad de dinero permanece constante.
- ▶ Si el estado es correcto, satisface invariantes.
- ▶ Una transacción va de un estado correcto a otro.
- ▶ Las invariantes se deben violar temporalmente (si no, no es factible implementar transacciones en entornos distribuidos).



## 8.2 Propiedades básicas

# Definiciones, propiedades básicas

- ▶ ACID.
- ▶ Atomicidad.
- ▶ Consistencia/Integridad.
- ▶ Aislamiento (implica atomicidad).
- ▶ Durabilidad.

# Atomicidad y aislamiento

- ▶ O sucede o no sucede, pero no sucede a medias.
- ▶ Aislamiento (implica atomicidad) → Unas transacciones no afectan a otras.
- ▶ Atomicidad y aislamiento están muy relacionados.

# Consistencia y durabilidad

- ▶ Consistencia/Integridad → Se va de un estado correcto a otro estado correcto.
- ▶ Durabilidad → El resultado de la transacción es permanente.

## 8.3 Two phase commit

## Ejemplo: Two phase commit (2PC)

- ▶ Un coordinador.
- ▶ N cohortes o trabajadores.
- ▶ Dos rondas.

# Commit Request

- ▶ Commit Request a los trabajadores.
- ▶ Contestan Sí/No.
- ▶ Cogen todos los cierres/recursos locales que necesiten.

# Commit

- ▶ Todos contestan sí → se manda commit.
- ▶ Alguno contesta no → se manda rollback (deshacer).



## Durabilidad en 2PC

- ▶ En cada fase cada participante.
- ▶ Debe ir a disco y apuntar todo en un log.
- ▶ Si se reinicia la máquina tras el commit, debe persistir.

# Fallos en 2PC

- ▶ Timeout, depende de la fase.
- ▶ Si estamos ya en commit, hay que esperar a que se recuperen.

## Problemas del 2PC, rendimiento

- ▶ Tengo todo bloqueado (R/W locks ayudan).
- ▶ Con la posibilidad de que otra transacción lo deshaga.
- ▶ Optimización (optimista), supongo commit o abort.
- ▶ Si mi suposición era mala, deshago (sólo hago el trabajo en ese caso).

## 8.4 Teorema CAP

# Teorema CAP

- ▶ Integridad/Consistencia, Disponibilidad, Tolerancia a particiones.
- ▶ Consistencia: todo el mundo ve los mismos datos.
- ▶ Disponibilidad: siempre se contesta con fallo o datos.
- ▶ Tolerancia a particiones: pueden fallar partes o unas no ver a otras.
- ▶ Elegir 2 de tres (**no se pueden conseguir simultáneamente** los tres).

# Teorema CAP

- ▶ <http://dl.acm.org/citation.cfm?doid=564585.564601>.
- ▶ <https://www.cl.cam.ac.uk/research/dtg/www/files/publications/public/mk428/cap-critique.pdf>.
- ▶ También llamado conjetura de Brewer.
- ▶ Fue muy polémico, en ciertos sentidos, no es un teorema (no muy preciso).
- ▶ [http://markburgess.org/blog\\_cap.html](http://markburgess.org/blog_cap.html).
- ▶ Es, de todas formas muy útil para entender los compromisos.
- ▶ Beating the CAP Theorem Checklist.

# Teorema CAP

- ▶ Establecer compromisos, latencia R/W.
- ▶ Tiempo para ver valores nuevos, Raft, etc.

# Bibliografía I



[van Steen & Tanenbaum, 2017] van Steen, M., Tanenbaum, A. S.  
*Distributed Systems.*

Third Edition, version 01. 2017.



[Birman, 2015] Birman, Kenneth P.

*Reliable distributed systems: technologies, web services, and applications.*

Springer, 2005.