IIC2523
Sistemas Distribuidos

Hernán F. Valdivieso López
(2025 - 2 / Clase 16)

Teorema PAC y PACELC
¿Cuáles son las limitaciones de un Sist. Distribuido?

Temas de la clase

1. Teorema PAC
    a. ¿Qué significa y qué implicancias tiene?
    b. Ejemplos aplicando el teorema
    c. Críticas
2. Teorema PACELC
    a. ¿Qué significa y su diferencia con el teorema PAC?
    b. Ejemplos aplicando el teorema

Introducción

Imaginen que compran entradas para el cine... pero cuando llegas se dan cuenta que otra persona tiene un *ticket* para el mismo puesto de ustedes.

🤔 ¿Qué creen que pasó en ese sistema? 🤔

Teorema PAC

Teorema CAP

En 2000, Eric Brewer propuso el Teorema CAP, pero fue formalizado Gilbert y Lynch:

In a network subject to communication failures, it is impossible for any web service to implement an atomic read/write shared memory that guarantees a response to every request

Fuente: Perspectives on the CAP Theorem

Teorema CAP

En 2000, Eric Brewer propuso el Teorema CAP, pero fue formalizado Gilbert y Lynch:

> *In a network subject to communication failures, it is impossible for any web service to implement an atomic read/write shared memory that guarantees a response to every request*

Para los mortales, esto indica que:

> *En presencia de una partición de red (P), un sistema distribuido debe elegir entre Consistencia (C) o Disponibilidad (A), pero no puede garantizar ambas.*

Fuente: Perspectives on the CAP Theorem

Teorema CAP

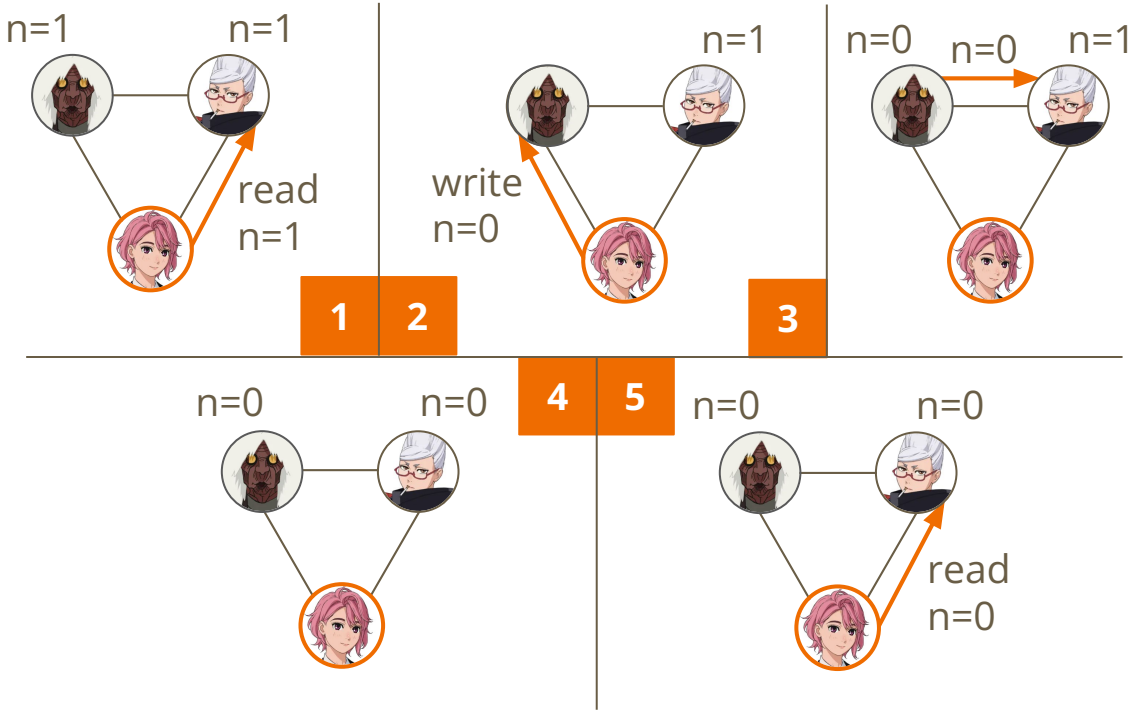
¿Qué es C, A y P? 🤔

- **Consistencia (C de *Consistency*):** Todas las réplicas tienen el mismo valor en un momento dado, y todas las lecturas devuelven el valor más reciente (consistencia fuerte).
- **Disponibilidad (A de *Availability*):** Cada solicitud recibe siempre una respuesta (éxito o falla), sin garantizar que contenga la versión más reciente de la información.
- **Tolerancia a la Partición (P de *Partition tolerance*):** El sistema continúa funcionando a pesar de la pérdida arbitraria de mensajes o la falla de partes del sistema, es decir, una partición de red.

Teorema CAP - Ejemplos

Sistema CA - Sin partición, se logra consistencia y disponibilidad.

La red puede responder en todo momento y se pueden coordinar para lograr tener los datos actualizados.



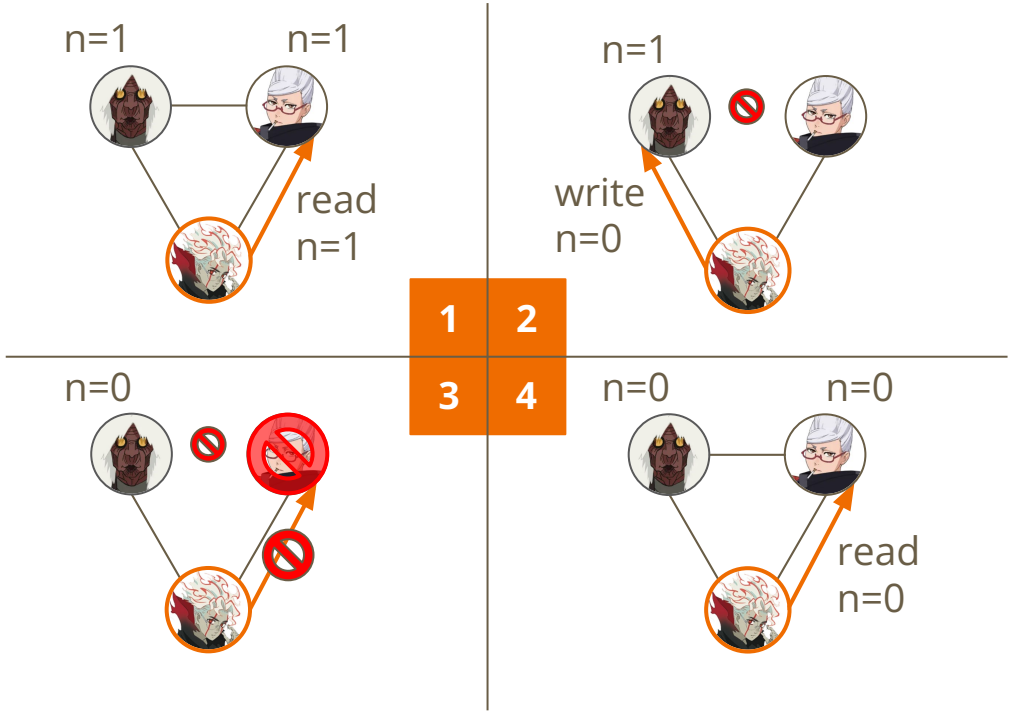
Teorema CAP - Ejemplos

Sistema CP - Tolerante ante partición (sigue funcionando) y se prioriza consistencia.

Si algún nodo se aísla debido a una partición, deja de procesar solicitudes para garantizar que los datos que entrega sean consistentes con el resto del sistema.

Uno o más nodos dejan de verse como disponibles.

Ejemplos: *Quorum based*



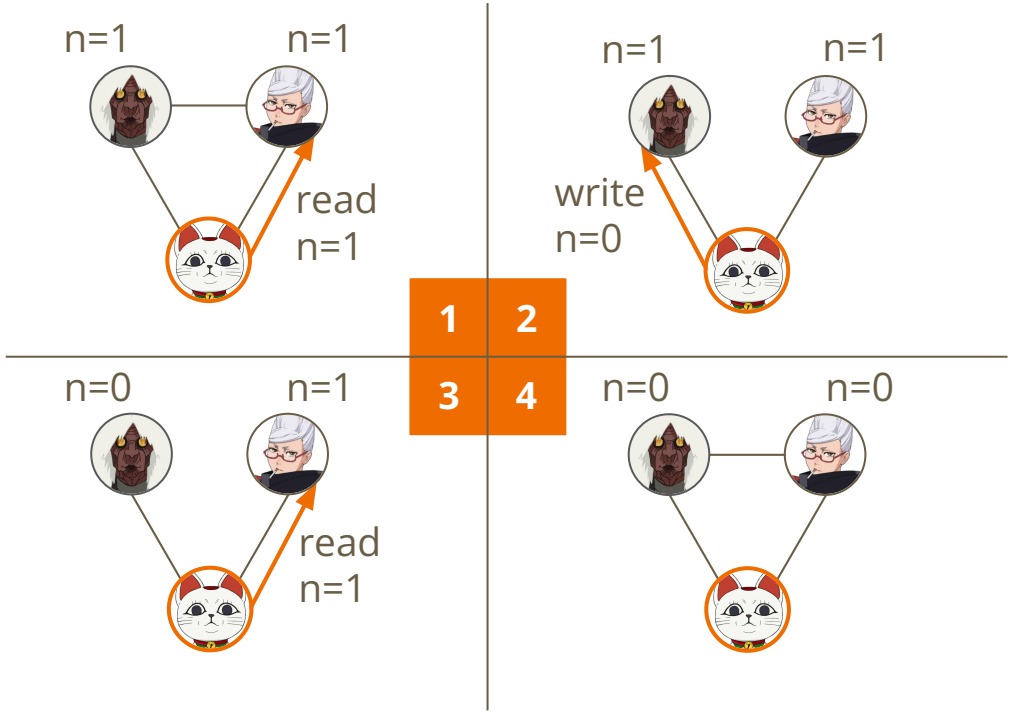
Teorema CAP - Ejemplos

Sistema AP - Tolerante ante partición (sigue funcionando) y se prioriza disponibilidad.

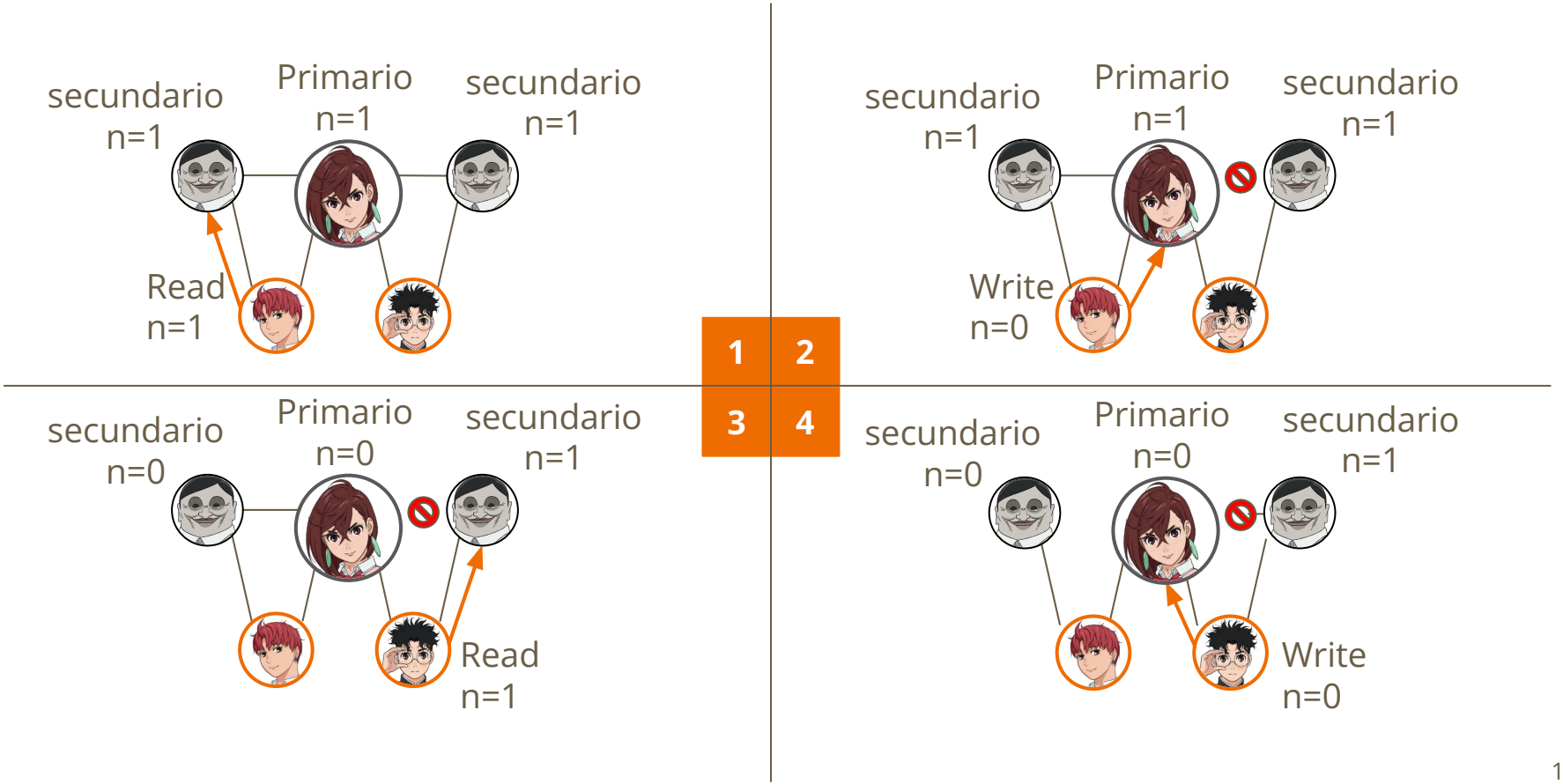
Si un nodo se aísla, continúa sirviendo solicitudes, incluso si no puede comunicarse con otras partes del sistema.

Se pueden entregar datos potencialmente obsoletos o inconsistentes, pero el sistema permanece disponible.

Ejemplos: protocolo *Gossip*



Teorema CAP - Ejemplo inicial



Teorema CAP - Críticas

Crítica 1 - CAP solo se activa cuando hay partición.

- En la práctica, las particiones no ocurren todo el tiempo.
- Si bien van a ocurrir, ¿qué sucede cuando no ocurre?
  - CAP solo dice que se puede lograr consistencia y disponibilidad... ¿pero hay algo más?
  - CAP no ayuda a pensar eso.

Teorema CAP - Críticas

Crítica 2 - El "todo o nada" es irreal

- CAP impone una visión binaria: o hay consistencia o no, o hay disponibilidad o no.
- Pero en la práctica, hay gradientes:
  - Consistencia causal
  - Quorum *reads/writes*
  - Latencia con *timeouts* variables

Teorema CAP - Críticas

Crítica 3 - No distingue entre latencia y disponibilidad

- Para Gilbert & Lynch: *Si un nodo no responde en tiempo finito, no es disponible.*
- En la práctica, uno puede esperar más, aunque tendrá cierto costo para la comunicación.
  - Si un sistema responde en 10s pero otro en 50ms, ¿ambos son disponibles? ¿no hay otro factor que influye en el diseño?

Teorema CAP - Reflexión

- El Teorema CAP muy útil para entender el límite del sistema distribuido, pero no suficiente para diseñar un sistema.
- No contempla compensaciones de requisitos cuando el sistema funcionaba con normalidad.
- Para diseñar, mejor ocupar otro teorema un poco más completo.

Teorema PACELC

Teorema PACELC

En 2010, Daniel J. Abadi propuso el Teorema PACELC:

> *if there is a partition (P), how does the system trade off availability and consistency (A and C); else (E), when the system is running normally in the absence of partitions, how does the system trade off latency (L) and consistency (C)?*

Fuente: Consistency Tradeoffs in Modern Distributed Database System Design

Teorema PACELC

En 2010, Daniel J. Abadi propuso el Teorema PACELC:

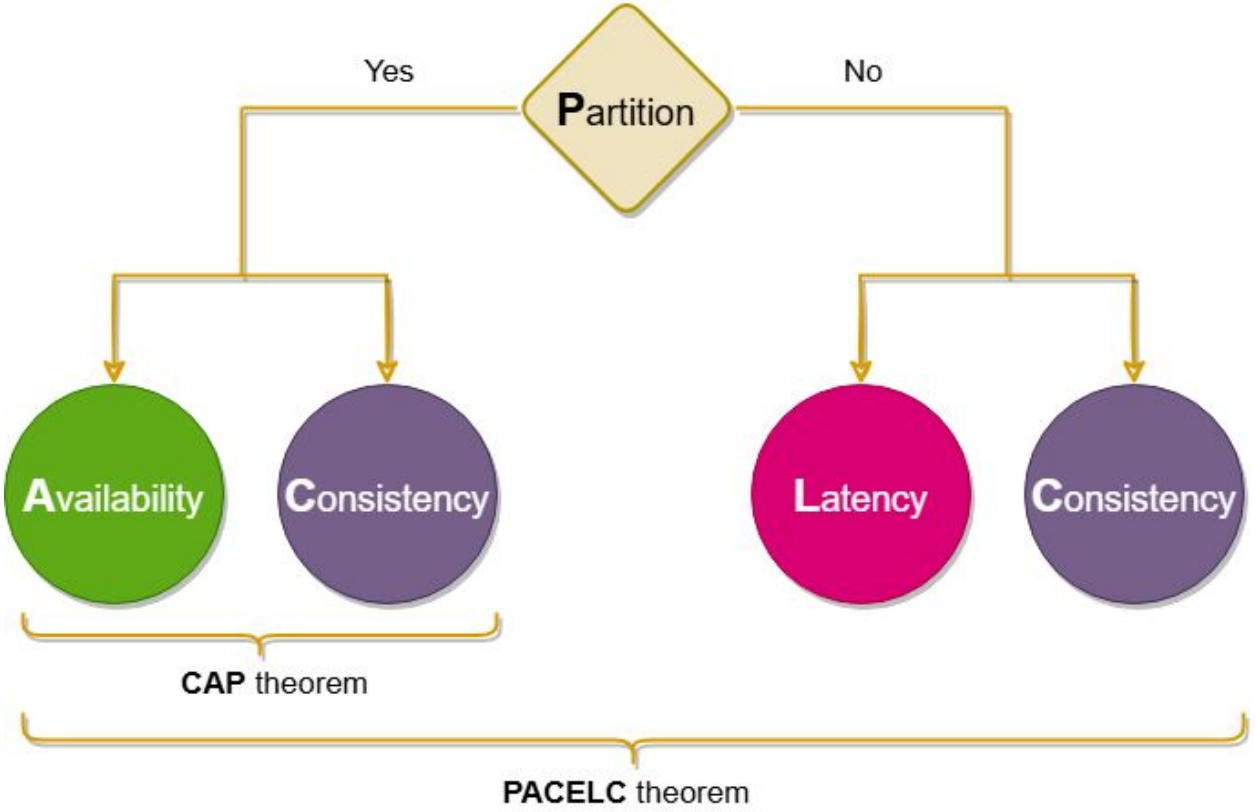
if there is a partition (P), how does the system trade off availability and consistency (A and C); else (E), when the system is running normally in the absence of partitions, how does the system trade off latency (L) and consistency (C)?

Para los mortales, este protocolo extiende CAP con el caso donde no hay partición:

En un sistema que no presenta partición de red (E), el sistema debe elegir entre latencia (L) y consistencia (C).

Fuente: Consistency Tradeoffs in Modern Distributed Database System Design

Teorema PACELC



Teorema PACELC

¿Qué es P, A, C, E, L y C? 🤔

- **(P) → *Partition tolerance***
- **(C) → *Consistency***
- **(A) → *Availability***
- ***ELSE* (E)**
- **Latencia (L de *Latency*):** es el tiempo que tardan los datos en transferirse a través de la red. Alta latencia implica un gran retraso en la comunicación.
- **(C) → *Consistency***

Teorema PACELC

¿Qué es P, A, C, E, L y C? 🤔

- **(P) → *Partition tolerance***
- **(C) → *Consistency***
- **(A) → *Availability***
- ***ELSE* (E)**
- **Latencia (L de *Latency*):** es el tiempo que tardan los datos en transferirse a través de la red. Alta latencia implica un gran retraso en la comunicación.
- **(C) → *Consistency***

Se introduce el concepto de Latencia y que ahora debemos matizar entre esta propiedad y consistencia.

Teorema PACELC

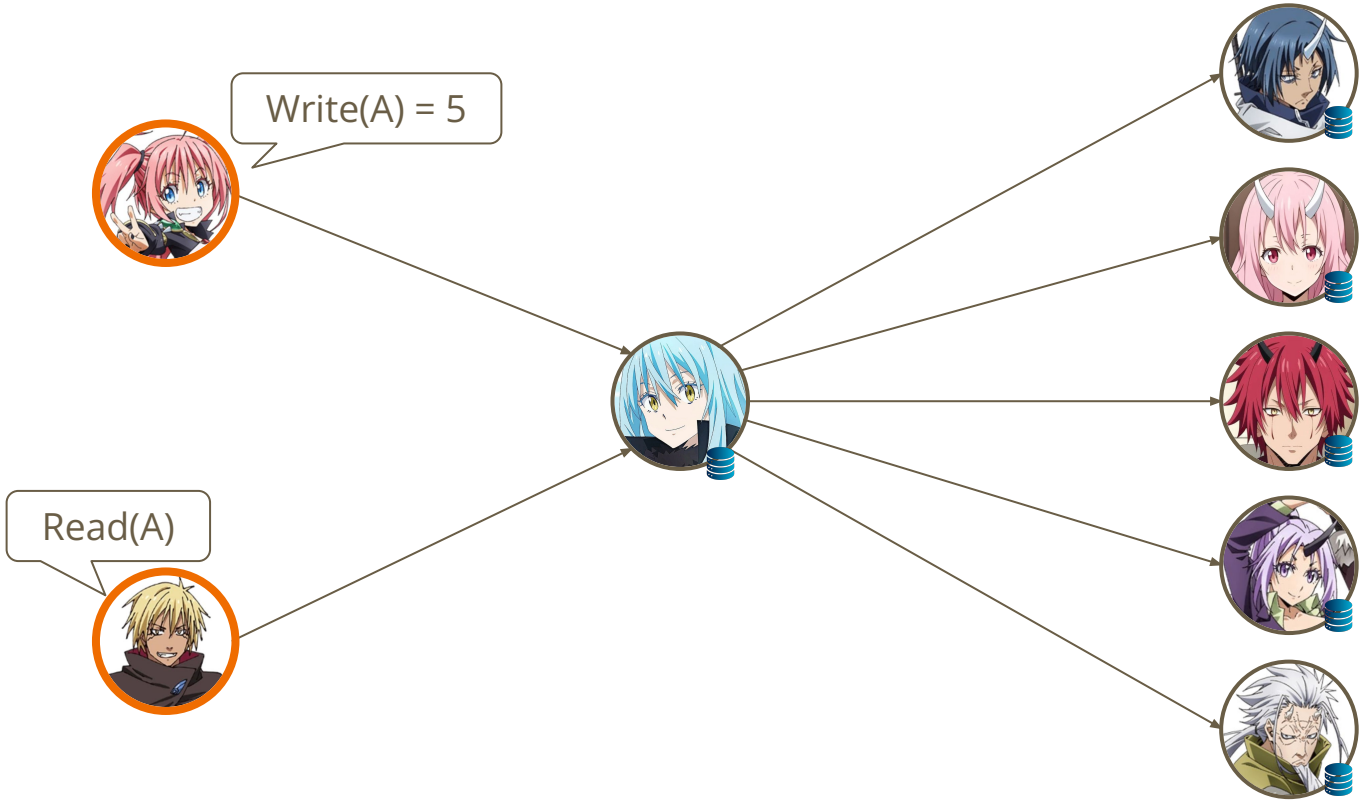
VS PAC

Imaginen el escenario en el que solicita comprar entradas de cine y recibe la confirmación de la compra 4 hora después.

- Según el teorema CAP, este sistema está disponible y consistente. Y fin del análisis.
- Para PACELC, dicha demora es una propiedad relevante del sistema.
  - En este contexto, es una demora inaceptable, es decir, se priorizó la consistencia por sobre la latencia.
  - Si bien sigue estando disponible, es a costa de una alta demora en la comunicación.

Teorema PACELC - Ejemplos

Priorizar Consistencia por sobre Latencia: *Primary-Based Protocols*



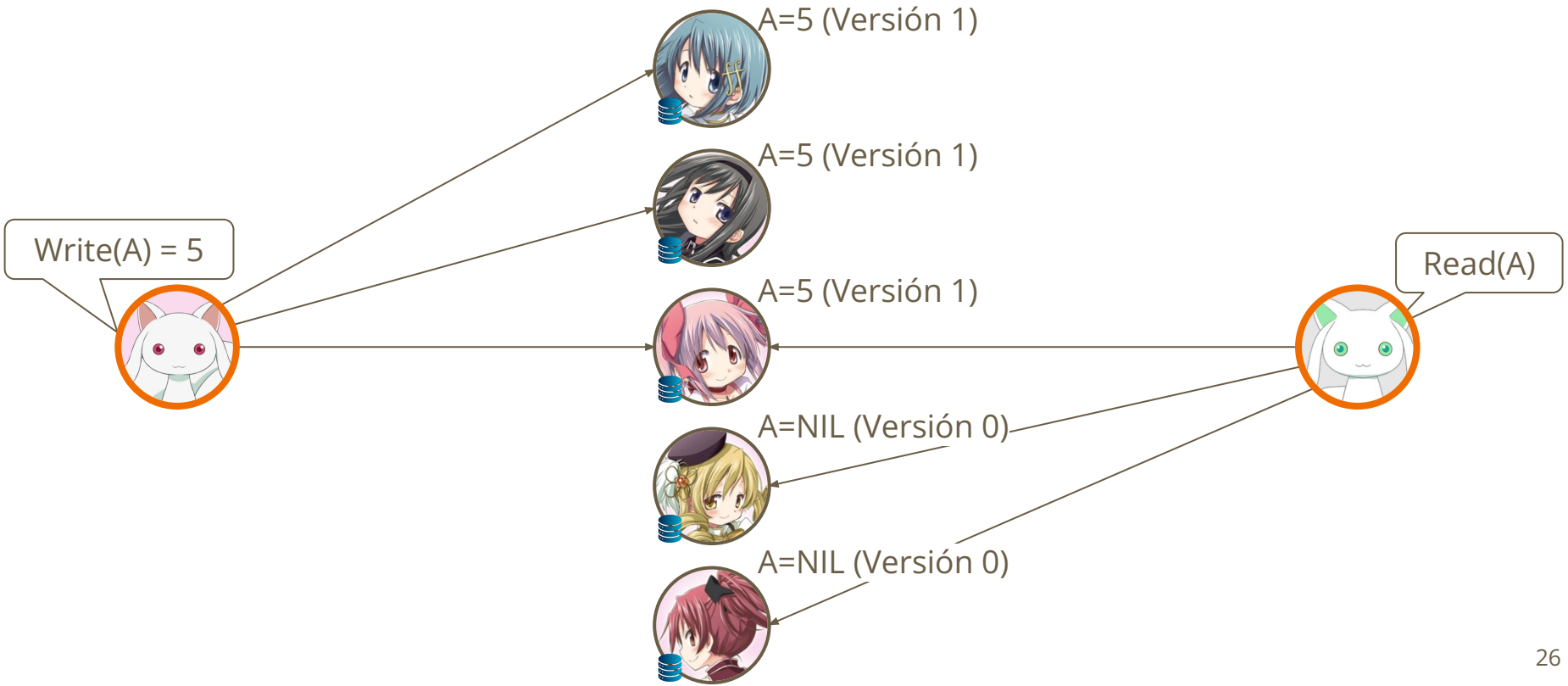
Teorema PACELC - Ejemplos

Priorizar Consistencia por sobre Latencia: *Primary-Based Protocols*

- **Las peticiones pasan por un nodo primario antes de llegar a los nodos réplica.**
- El nodo primario ordena las peticiones y asegura que los nodos réplica apliquen los cambios en el mismo orden.
  - Se mantiene la consistencia.
- Costo de una mayor latencia por...
  - Tiempo de secuenciamiento en el nodo primario.
  - Desacoplamiento entre cliente y nodos de datos.
  - Posible distancia geográfica entre cliente y nodo primario.

Teorema PACELC - Ejemplos

Priorizar Consistencia por sobre Latencia: *Quorum-based*



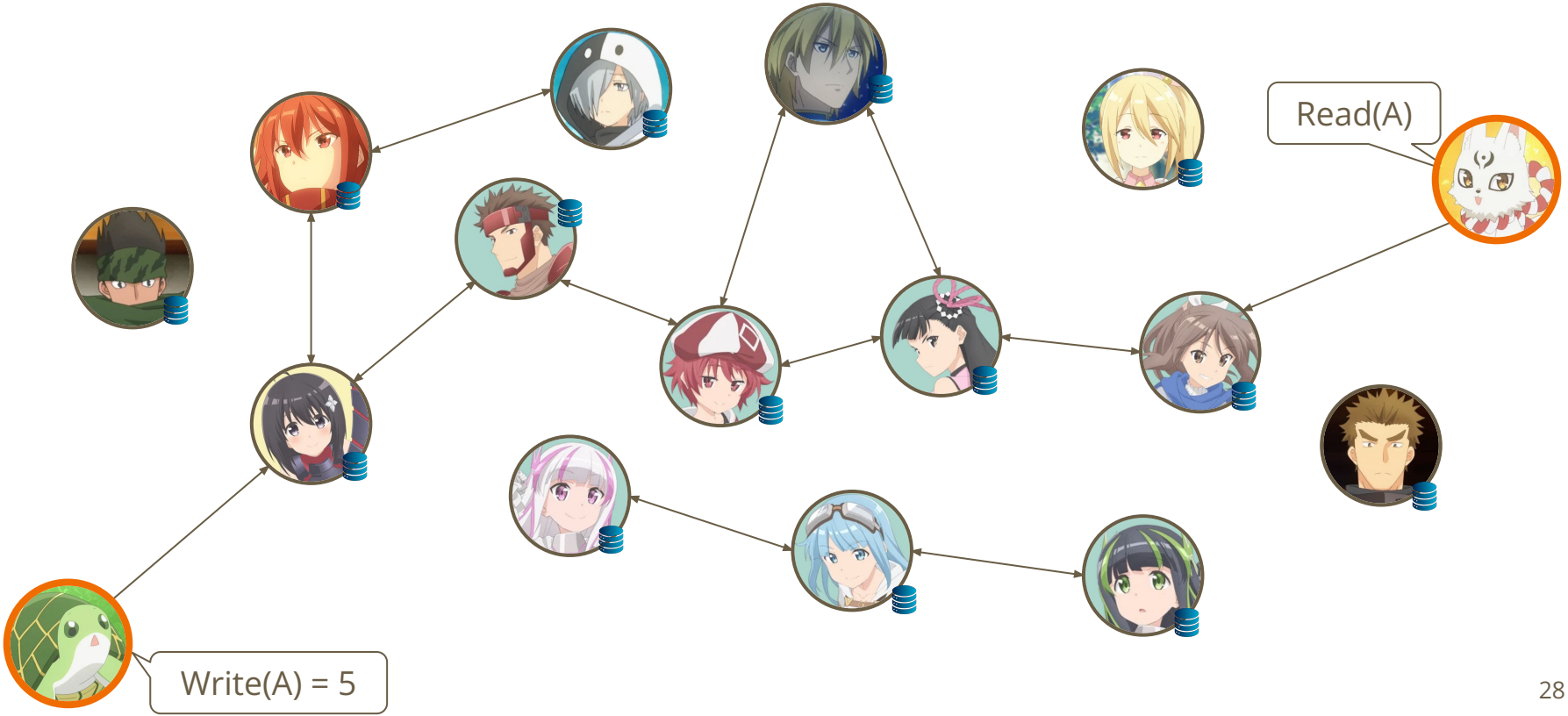
Teorema PACELC - Ejemplos

Priorizar Consistencia por sobre Latencia: *Quorum-based*

- **Las operaciones pueden ir a cualquier nodo, pero debe conseguir el *Quoarum* mínimo.**
- Aumenta la consistencia global del sistema.
- Costo de una mayor latencia por...
  - Espera a todos los nodos (el más lento domina).
  - Latencia de red según ubicación geográfica de los nodos.

Teorema PACELC - Ejemplos

Priorizar Latencia por sobre Consistencia: Protocolo *Gossip*



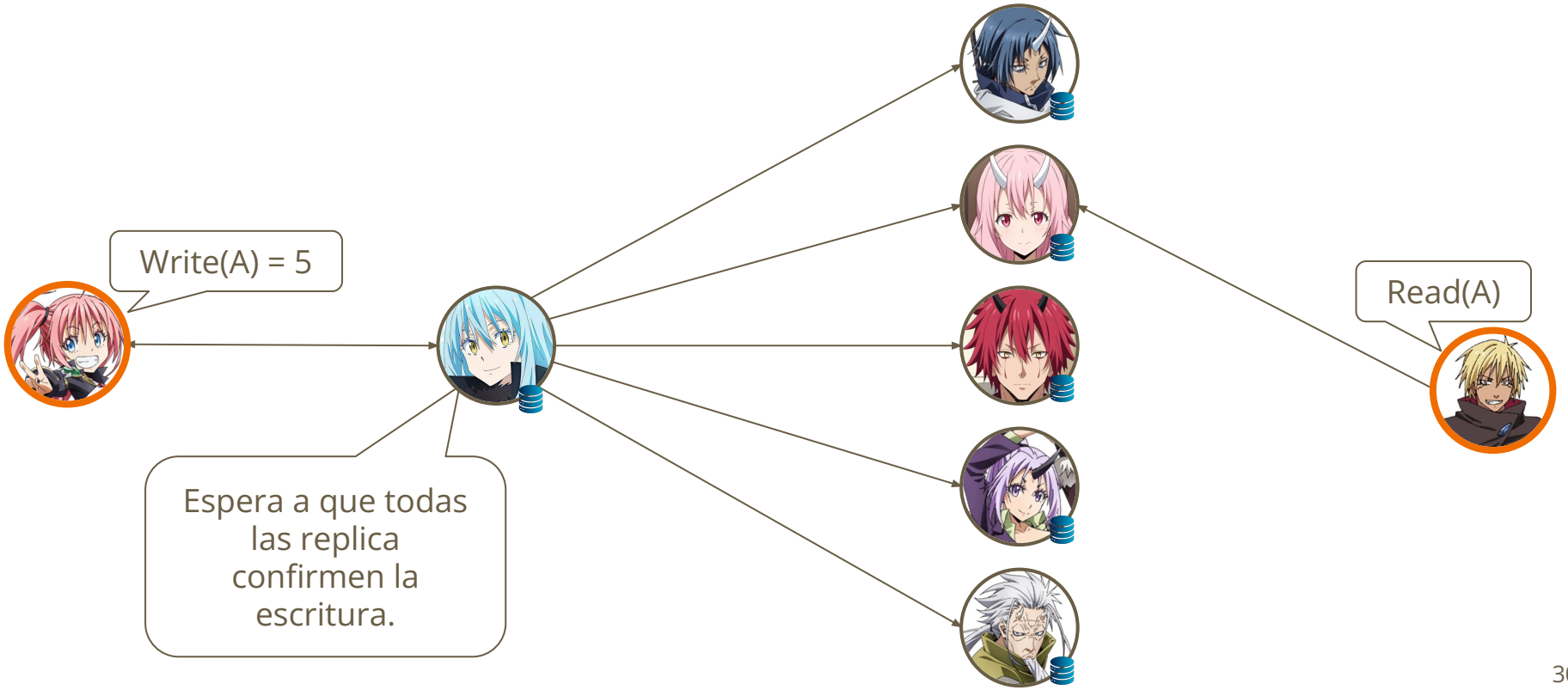
Teorema PACELC - Ejemplos

Priorizar Latencia por sobre Consistencia: Protocolo *Gossip*

- **Las peticiones pasan por cualquier nodo y eventualmente se transmite la información**
- Se obtienen respuestas de forma rápida.
- Costo de una mayor Consistencia por...
  - Los nodos más lejanos al que ejecutó la operación se demora más en ver la información.
  - Los usuarios, según el nodo consultado, pueden ver información diferente.

Teorema PACELC - Ejemplos

Priorizar Consistencia o Latencia según la operación : *Synchronous Data Replication*



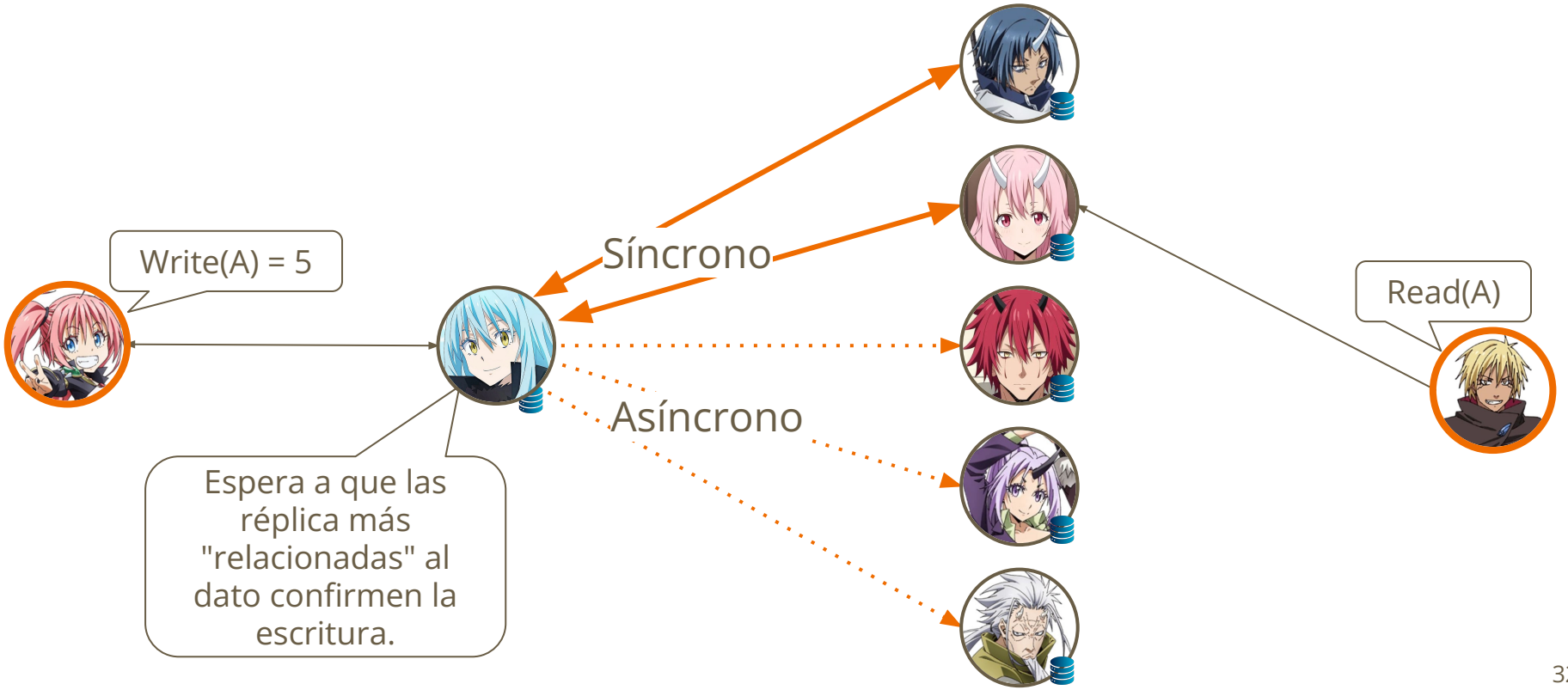
Teorema PACELC - Ejemplos

Priorizar Consistencia o Latencia según la operación : *Synchronous Data Replication*

- **El nodo primario recibe todas las escrituras y nodos secundarios sirven para lecturas.**
- Replicación sincrónica
  - El nodo primario espera a que todos los secundarios se actualicen antes de confirmar la escritura.
- Aumenta la consistencia global del sistema cuando se modifica información a costo de una mayor latencia por esperar a la confirmación.
- Se prioriza la latencia exclusivamente en la lectura, aunque todavía es posible que la lectura vea información desactualizada.

Teorema PACELC - Ejemplos

Punto medio entre latencia y consistencia: *Asynchronous Data Replication*



Teorema PACELC - Ejemplos

Punto medio entre latencia y consistencia: *Asynchronous Data Replication*

- **Modelo híbrido de replicación:**
  - Sincrónica hacia un subconjunto de réplicas.
  - Asíncrona hacia el resto.
- El nodo primario maneja las escrituras.
- Lecturas pueden dirigirse a réplicas o al primario.
- Balancea *tradeoffs*:
  - Mejora consistencia sin el costo total de latencia.
  - Reduce latencia sin perder completamente la sincronía.

Teorema PACELC - Casos reales

Amazon Dynamo

- Base de datos NoSQL de uso interno de Amazon.
- **En partición:** prefiere disponibilidad → PA
- **Sin partición:** elige latencia sobre consistencia fuerte → EL
  - Usa consistencia eventual.
- **Resultado:** PA/EL

Fuente: Dynamo: Amazon's Highly Available Key-value Store

Teorema PACELC - Casos reales

Amazon Dynamo

- Base de datos NoSQL de uso interno de Amazon.
- **En partición:** prefiere disponibilidad → PA
- **Sin partición:** elige latencia sobre consistencia fuerte → EL
  - Usa consistencia eventual.
- **Resultado:** PA/EL

Desafío para la casa... ¿y qué pasa con DynamoDB, el sucesor de Dynamo? Paper de DynamoDB

Fuente: Dynamo: Amazon's Highly Available Key-value Store

Teorema PACELC - Casos reales

Google Spanner

- Base de datos NewSQL distribuida globalmente por Google.
- **En partición:** prefiere consistencia → PC
- **Sin partición:** elige consistencia por sobre latencia → EC
  - Usa reloj *TrueTime* para garantizar orden y serialización global.
  - Ralentiza sus operaciones para reducir la incertidumbre de que se pierde consistencia.
- **Resultado:** PC/EC

Fuente: Spanner: Google's Globally-Distributed Database

Teorema PACELC - Casos reales

Git

- Sistema de control de versiones distribuido.
- **En partición:** prefiere disponibilidad → PA
- **Sin partición:** elige latencia por sobre consistencia → EL
  - Opta por una consistencia eventual cuando el usuario haga *merge* de todo.
  - Prioriza que podamos seguir pusheando (y creando *branch* de ser necesario).
  - Aunque esté todo conectado, cada usuario puede ver información distinta.
- **Resultado:** PA/EL

Poniendo a prueba lo que hemos aprendido 👀

Un sistema bancario global tiene múltiples sucursales con bases de datos replicadas para mejorar el acceso regional. Los clientes pueden consultar su saldo en cualquier sucursal local. Las consultas se atienden desde la réplica local, la cual se sincroniza periódicamente con un servidor central.

En condiciones normales, la réplica local puede mostrar saldos con cierta **demora** respecto al servidor central debido a la sincronización diferida. En caso de que una sucursal pierda conexión con el servidor central, se notifica de un error en la plataforma y solo se permite el acceso cuando se recupere la conexión.

¿Cuál es el comportamiento del sistema respecto al **teorema PAC y PACELC** para las consultas de saldo?

a. PA/EL

b. PC/EC

c. PC/EL

d. PA/EC

Poniendo a prueba lo que hemos aprendido 👀

Un sistema bancario global tiene múltiples sucursales con bases de datos replicadas para mejorar el acceso regional. Los clientes pueden consultar su saldo en cualquier sucursal local. Las consultas se atienden desde la réplica local, la cual se sincroniza periódicamente con un servidor central.

En condiciones normales, la réplica local puede mostrar saldos con cierta **demora** respecto al servidor central debido a la sincronización diferida. En caso de que una sucursal pierda conexión con el servidor central, se notifica de un error en la plataforma y solo se permite el acceso cuando se recupere la conexión.

¿Cuál es el comportamiento del sistema respecto al **teorema PAC y PACELC** para las consultas de saldo?

a. PA/EL

b. PC/EC

c. PC/EL

d. PA/EC

Próximos eventos

Próxima clase

- Consistencia centrada en el cliente, ¿Existen otro tipo de consistencia?
- Hablar de la tarea 3 si es que queda tiempo

Evaluación

- Mañana se publica la última tarea aplicando el contenido de transacciones distribuidas.
- Control 5 se publica el miércoles de la otra semana y se entrega el miércoles antes de la prueba. Tendremos toda la clase del otro miercoles para trabajar en él.

IIC2523
Sistemas Distribuidos

Hernán F. Valdivieso López
(2025 - 2 / Clase 16)

Créditos (animes utilizados)

Itai no wa Iya nano de Bōgyoryoku ni Kyokufuri Shitai to Omoimasu (Bofuri)

Tensei shitara Slime Datta Ken

Dandadan

Puella Magi Madoka Magica