Concurrencia & Paralelismo

Tandil 2020

Agenda de Hoy

- Conceptos de Concurrencia (1h aprox)
- LTs (30’ aprox)
- Presentación parte 2 del trabajo

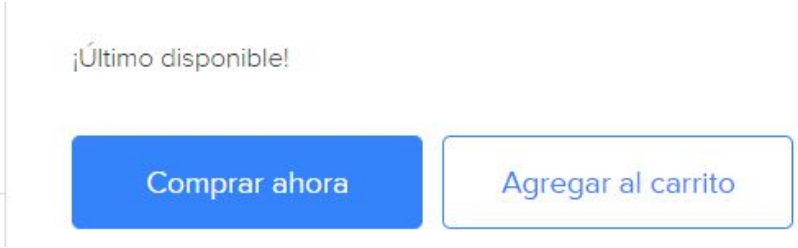
PARCIAL

29 - OCT - 18 hs

Problema 1

Tengo en mi sistema productos y quiero poder interactuar con muchos usuarios

Qué sucede cuando diferentes usuarios quieren modificar el mismo dato?



Problema 1

Puedo llegar a un estado de inconsistencia:

- Datos que se pierden o erroneos (stock negativo)
- Sistema inestable (porque no hay chequeos correctos)
- Respuesta incorrecta al usuario

Problema reducido

- Cuando tengo 2 o más usuarios intentando acceder y actualizar un mismo datos es un problema de **concurrencia** que puede afectar la seguridad de la información

Multiusuario

Los sitios WEB son por definición *multi-usuario*:

- Cargan la misma página
- Acceden y modifican a los mismos datos
- Leen los mismos recursos

Evolución de sistemas concurrentes

Los usuarios en general somos “multi-tarea”

Definición

Concurrencia es la propiedad de los sistemas en la cual los procesos de un cómputo se hacen **simultáneamente**, y pueden **interactuar** entre ellos y el resultado será el mismo. Las operaciones pueden ser ejecutados en múltiples **procesadores**, o ejecutados virtualmente en distintos **threads** de ejecución.

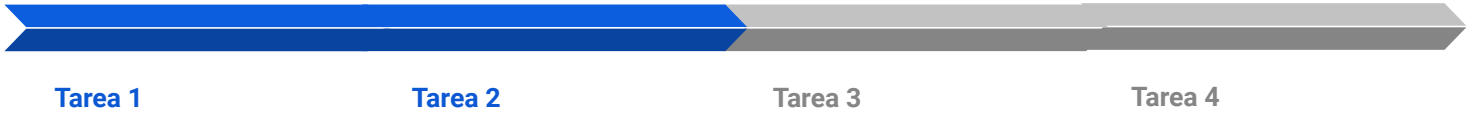
Problema de “integridad” la información

Debemos asegurar que se llegue a un estado correcto de los datos, independientemente del modo que se ejecutaron las acciones.

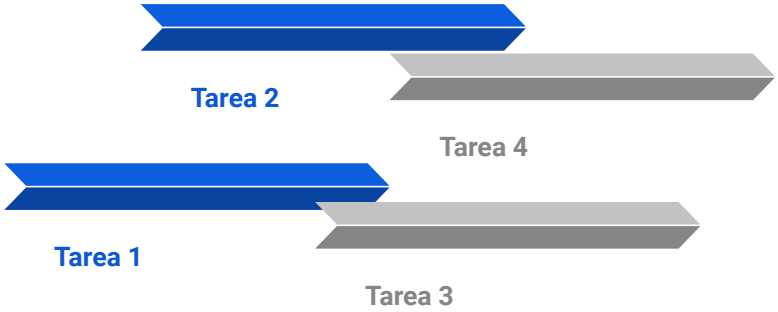
- 2 usuarios , 2 tareas (X,Y)

Qué diferencia tiene con un sistema secuencial?

- En un sistema “secuencial” existe una única invocación que se realiza en orden



- En un sistema “concurrente” existen múltiples tareas que se activan en el mismo tiempo



Origen de concurrencia

La concurrencia surge de resolver un **gran problema** en pequeñas partes

Queremos “reducir el tiempo” de procesamiento/espera

Tengo **múltiples unidades de procesamiento** disponibles

Origen de concurrencia

- **Descomposición por tareas**: procesos/servicios diferentes accediendo a recursos comunes
- <- sistema distribuido

- **Descomposición por datos**. Procesos iguales que resuelven un problema mayor por partes.

  <- sistema paralelo .

Descomposición por tareas

- Distribuir la tarea entre los procesadores
- Cada tarea es diferente
- Por ej: lista de la casa

✓ LAVAR EL AUTO

✓ RETIRAR DINERO

✓ IR AL SUPER

✓ PAGAR EL GAS Y LA LUZ

✓ COMPRAR MATERIALES

Cómo divido?

Descomposición por datos

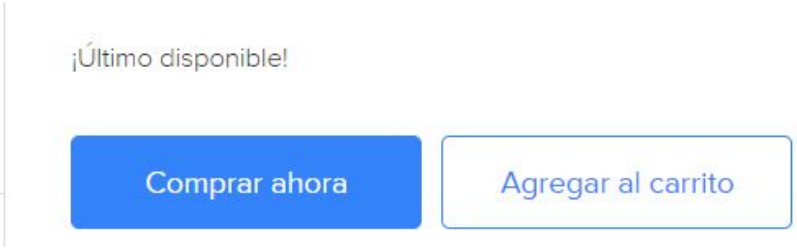
- La tarea es idéntica
- Los datos son compartidos
- Por ej :
  - Ordenar por fecha muchos archivos

Cómo divido?

Sistemas distribuidos y concurrentes

Problema 1

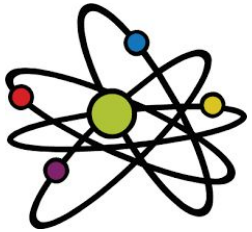
Qué sucede cuando diferentes usuarios quieren modificar el mismo dato?



ideas???

Ideas

- Chequear si está disponible al momento de “Comprar”
  - Avisarle al usuario que el producto está por ser comprado
  - Asegurar que la operación es “atómica”



Problema 2

Qué sucede cuando múltiples usuarios quieren modificar o borrar el mismo dato?

ideas???

Una solución al problema

Podemos “bloquear” el recurso hasta que se libere.

- Esto es un **semáforo**.
  - Puede ser una variable **booleana** o una condición previa a realizar la operación,
  - El usuario que llega primero “cambia el estado”

Implementar mi sitio para múltiples usuario

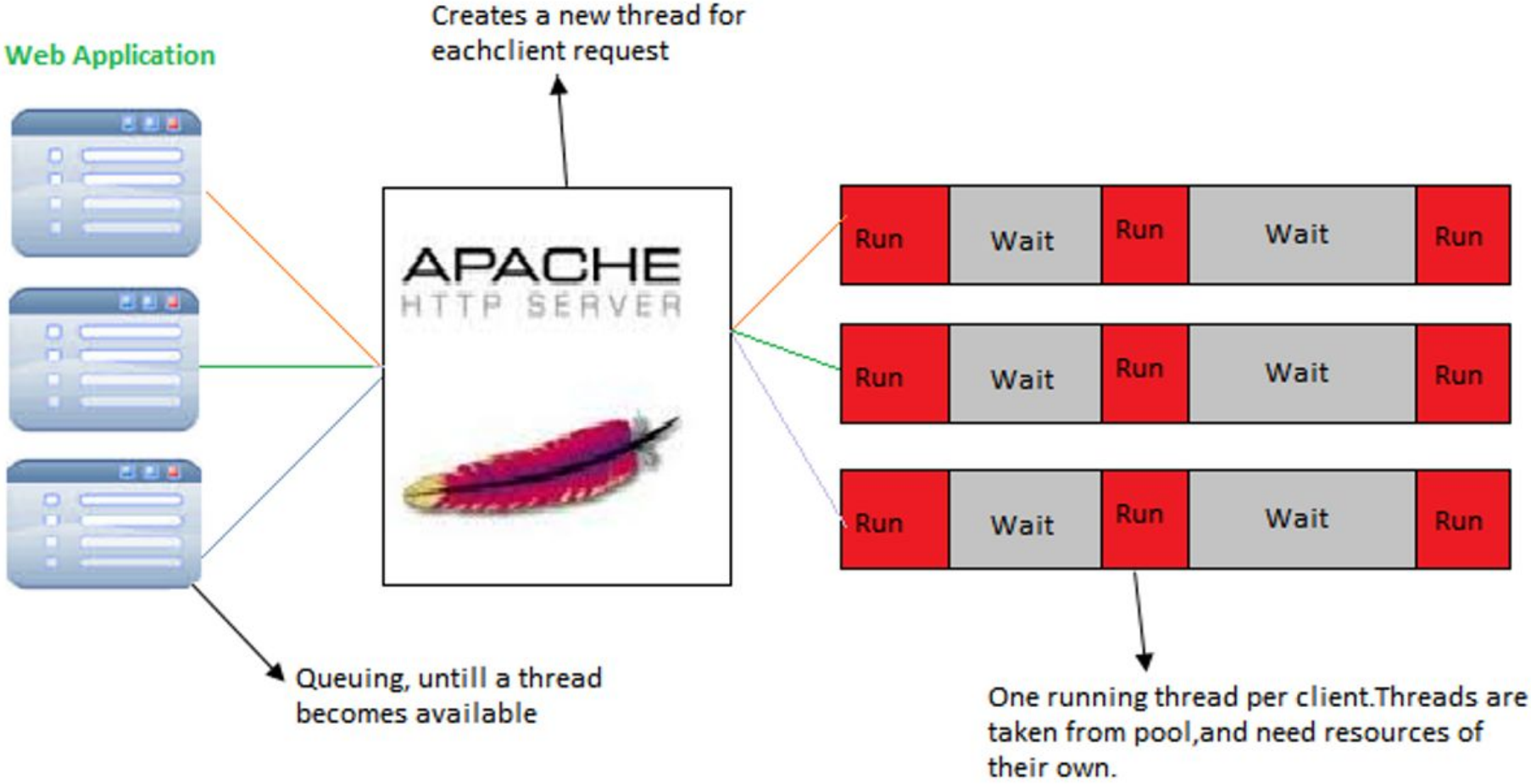
1. Cómo manejo los accesos al servidor?

2. Como resuelvo los accesos a la BBDD?

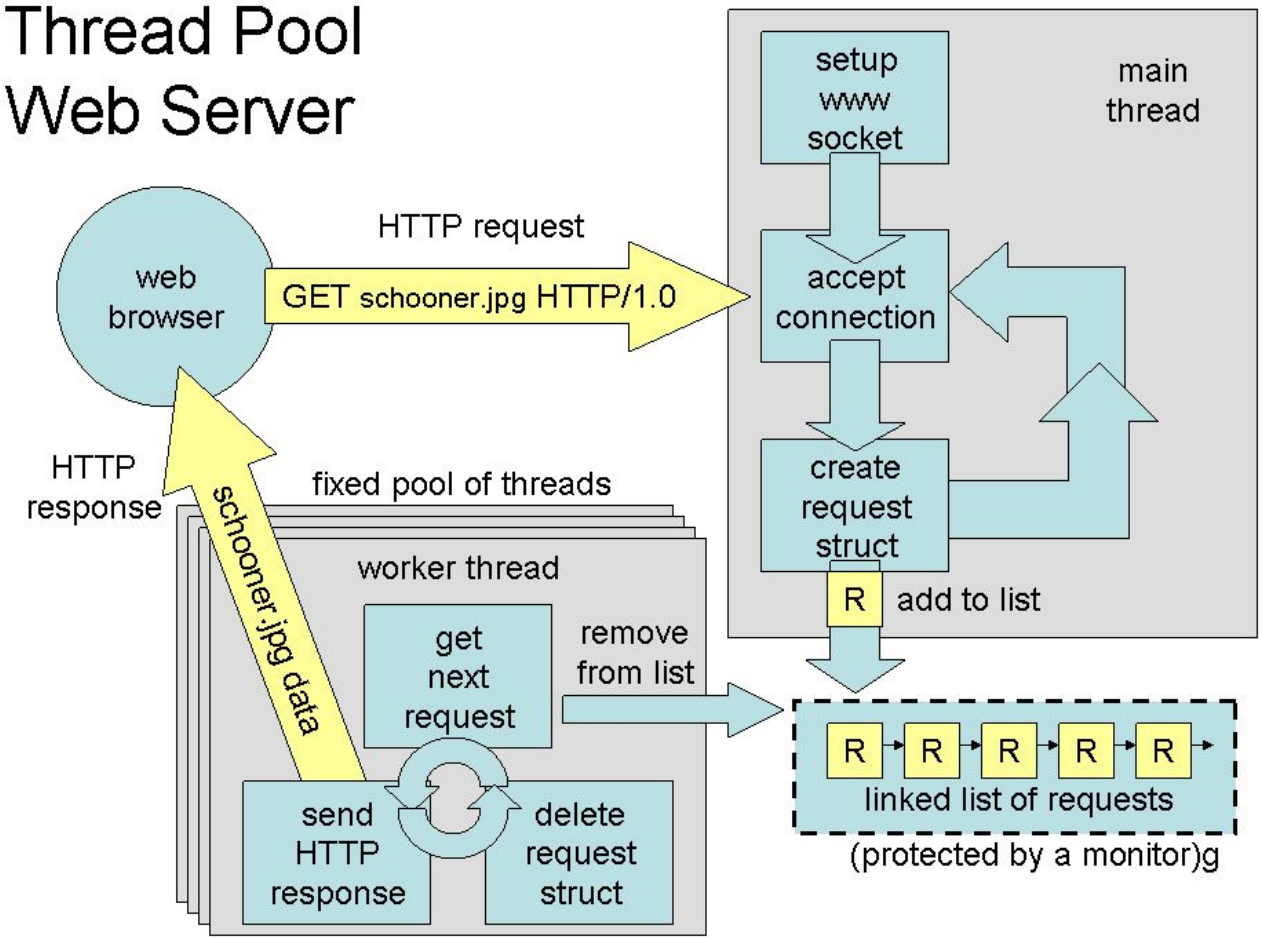
3. Como hago para que múltiples usuarios quieren modificar o borrar el mismo dato? Hay que implementar los métodos de control en las clases?

4. Cómo lo testeo?

1. Como maneja un servidor múltiples conexiones?



1. Como maneja un servidor múltiples conexiones?



2. BBDD concurrente

Las BBDD ya tienen implementados muchos controles y administra las consultas concurrentes.

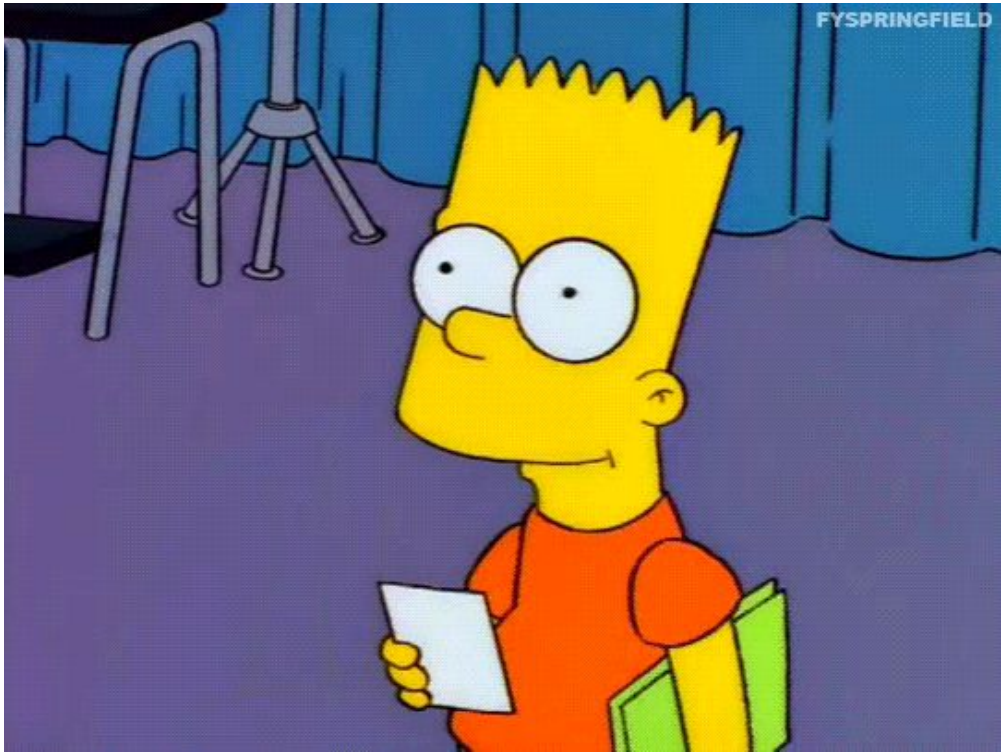
3. Implementamos el control de modificación

1. Agrego una columna para controlar datos “lock”
2. El primer usuario, al comenzar a editar setea “lock = true”
3. El primer usuario, al finalizar setea “lock = false”

Si otro usuario quiere modificar/borrar el mismo registro, chequea que la variable este en “false” sino el sistema lo rechaza.

4. Cómo lo testeo?

Ok, NO es fácil de testear



4. Cómo lo testeo?

1. Con usuarios reales accediendo en simultáneo (o usando varias sesiones)
2. Implementando métodos de testing
3. Con herramientas automáticas

Ejemplos de concurrencia

El objetivo de

Uno de los objetivos de trabajar en una arquitectura distribuida es lograr:

Escalabilidad

Volvamos a las arquitecturas...

MVC es fácilmente distribuible y soporta múltiples usuarios

pero..

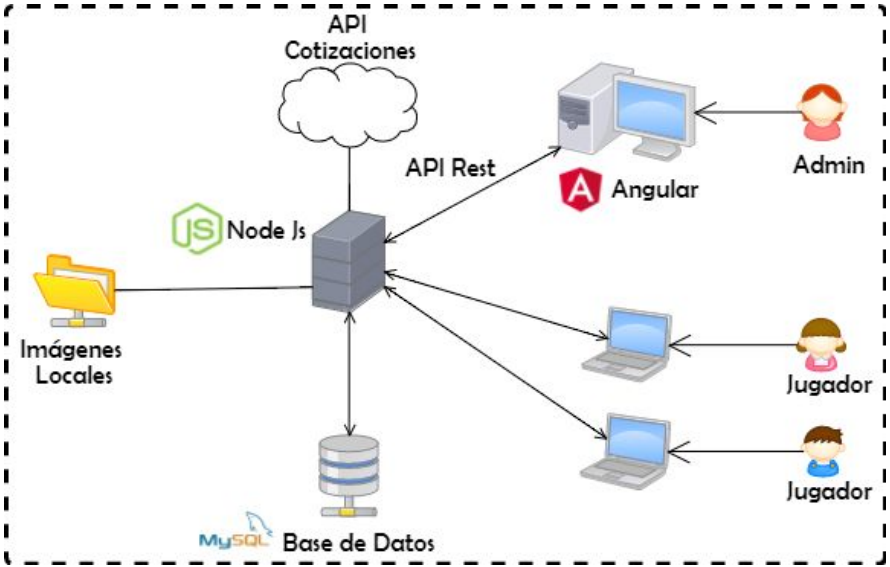
Concurrencia en sistemas distribuidos

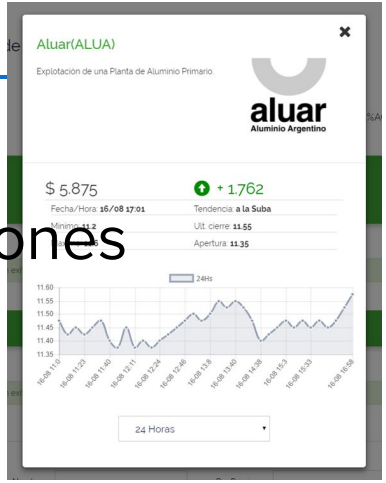
Los sistemas no solo tienen múltiples usuarios, también **interactúan con otros sistemas**

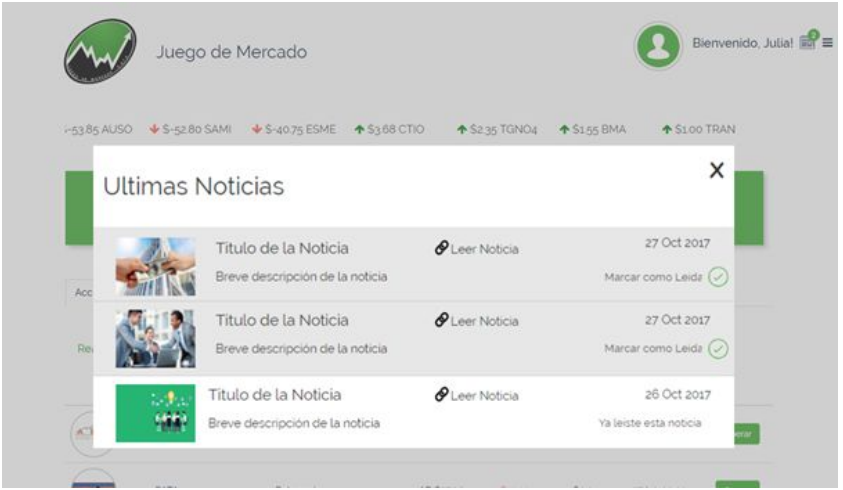
Interacción con otros sistemas

Caso “Juego de Mercado”

- Consume “micro-servicio” del valor de las Acciones







Interacción con otros sistemas

- Surgen potenciales problemas:
  - Robustez del sistema externo
    - No siempre está disponible
  - Capacidad de comunicación
    - La velocidad que necesitamos nosotros no es la disponible

Interacción con otros sistemas

- Para reducir estas problemáticas, debemos tener políticas propias de manejo de los datos (por ej: redundancia de datos)

Concurrencia en sistemas masivos

Cómo puede un servidor manejar miles de jugadores en línea?

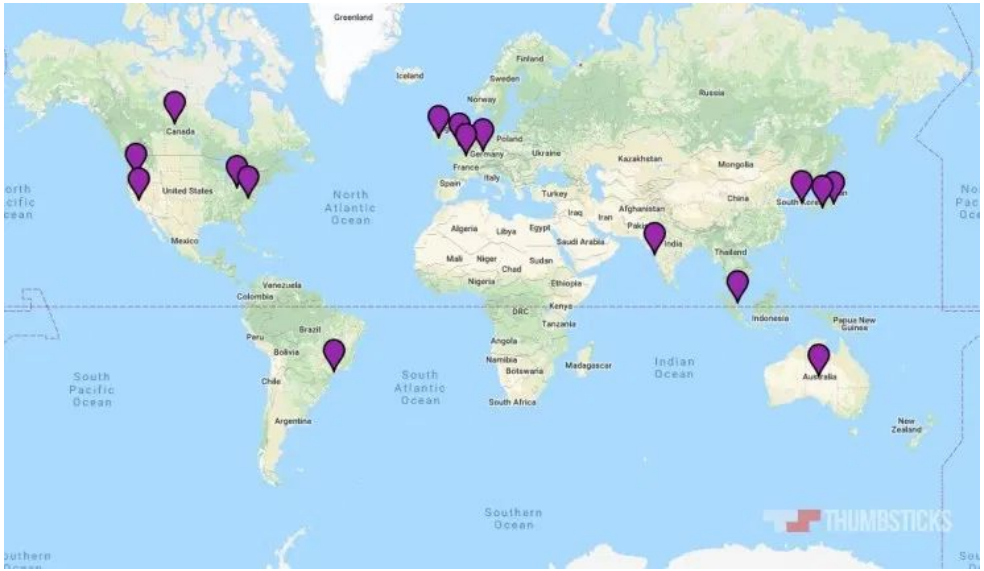
Cómo maneja UBER millones de usuarios?

Concurrencia en sistemas masivos

- HECHO : No todos los usuarios están interactuando entre ellos
  - UBER -> interactúan sólo los usuarios cercanos
  - Fortnite -> acceder a una zona de juego por grupo

Concurrencia en sistemas masivos

- Existen servidores redundantes que manejan diferentes grupos de usuarios, con las mismas tareas
  - Aplican PARALELISMO DE DATOS!
  - Por ejemplo : UBER agrupa usuarios por ciudad



Desafios

Como se trabaja en GoogleDOCS : Ver : link

Resumen

- Los sistemas actuales son “concurrentes”
- Gran parte de la concurrencia es de acceso a datos y se resuelve al centralizar en una BBDD
- Existen patrones de solución ya conocidos al problema basados en “semáforos”

http://web.mit.edu/6.005/www/fa16/classes/19-concurrency/

https://stackoverflow.com/questions/26887644/how-to-simulate-1000-concurrent-user-using-jmeter

Concurrencia y SO

- **Asíncrono** = evento que no tiene lugar en total correspondencia temporal con otro evento

- **Paralelismo** = es un mecanismo computacional en la cual varias acciones pueden realizarse simultáneamente

- **Semáforo** = Es una variable especial que emplean para permitir o denegar el acceso a partes del programas donde se manipulan recursos que deben ser accedidos de forma especial (llamados **secciones críticas**)

Algunos terminos

Productor - consumidor

Lock

Semaforos

Asincronismo

Atomicidad