

SISTEMAS DISTRIBUIDOS

RAID CONTROLLERS DON'T MAKE SENSE AT OUR SCALE; EVERYTHING IS REDUNDANT AT HIGHER LEVELS. WHEN A DRIVE FAILS, WE JUST THROW AWAY THE WHOLE MACHINE.



MACHINE? WE THROW AWAY WHOLE RACKS AT A TIME.

YEAH, WHO REPLACES ONE SERVER?



WE JUST REPLACE WHOLE ROOMS AT ONCE. AT OUR SCALE, MESSING WITH RACKS ISN'T ECONOMICAL.

WOW.
LIKE GOOGLE!



WE DON'T HAVE SPRINKLERS OR INERT GAS SYSTEMS. WHEN A DATACENTER CATCHES FIRE, WE JUST ROPE IT OFF AND REBUILD ONE TOWN OVER.

MAKES SENSE.

I WONDER IF THE ROPE IS REALLY NECESSARY.



CURSO DE ESCALABILIDAD V2, DÍA 1

VUESTROS ANFITRIONES



Alex Fernández (pinchito)
Ingeniero senior
freelancer



Alfredo López Moltó
Ingeniero de software
en hdivsecurity.com

PROGRAMA

¿Qué es escalabilidad?

Escalado horizontal y vertical

Estrategias de escalado

Sistemas distribuidos

Replicación de servidores

QUE ES ESCALABILIDAD



Fuente

DEFINICION DE ESCALABILIDAD

The capacity to be changed in size or scale.

The ability of a computing process to be used or produced in a range of capabilities.

Lexico (Oxford Dictionary)

ESCALAR ARRIBA ¡Y ABAJO!



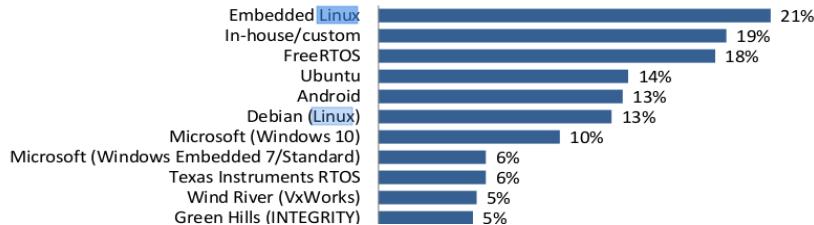
USO EN LITERATURA



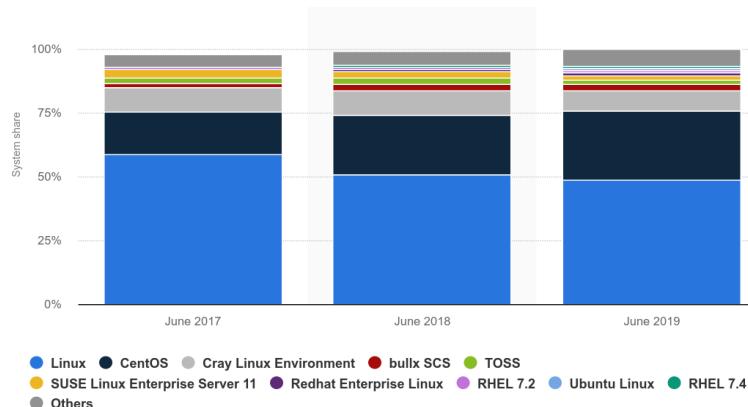
EJEMPLO: LINUX

De embedded

Please select ALL of the operating systems you are currently using.



a supercomputers



¿QUE HACE QUE ALGO SEA ESCALABLE?

La pregunta correcta es: ¿por qué no escala algo?

- Escasez de algún recurso
- Tiempo de espera creciente
- Imposibilidad de responder
- Bloqueo de algún recurso
- ...

EJERCICIO: SERVICIO POCO ESCALABLE

Instala Node.js

Instala loadtest

```
$ npm install -g loadtest
```

Ejecuta el comando

```
$ loadtest http://service.pinchito.es:3000/a -n 2000 -c 100
```

Anota la media de peticiones por segundo (rps),
latencia media y número de errores



EJERCICIO +

Crea una instancia en AWS EC2
Región eu-west-3, Europe (Paris)
Tipo t2-small (o superior)

Imagen **aps-loadtest-2020-06-12** (ami-07893127df0865a01)

Accede por SSH y ejecuta el comando anterior:

```
# ssh ubuntu@[direcciónIP]  
$ loadtest http://service.pinchito.es:3000/a -n 2000 -c 100
```

Anota todo de nuevo: rps, latencia, errores



EJERCICIO +

Ahora ajustar las rps de loadtest de 100 ± 100

```
$ loadtest http://service.pinchito.es:3000/a -n 2000 -c 100 --rps 300
```

```
$ loadtest http://service.pinchito.es:3000/a -n 2000 -c 100 --rps 400
```

...

Anotar rps, latencia y errores resultantes

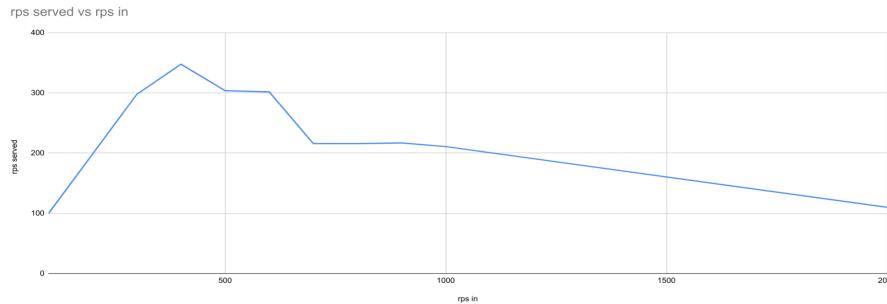
Ir subiendo hasta 1000, luego hacer 2000

```
$ loadtest http://service.pinchito.es:3000/a -n 2000 -c 100 --rps 2000
```

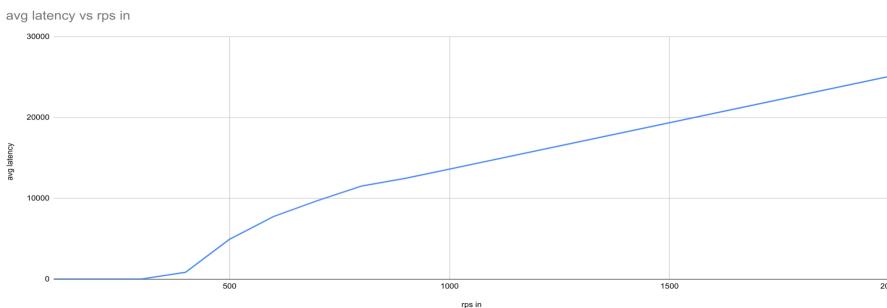


EJERCICIO +

Dibuja una gráfica con rps configuradas
y rps resultantes



Otra con rps vs latencia



EJERCICIO +

Ahora prueba contra:

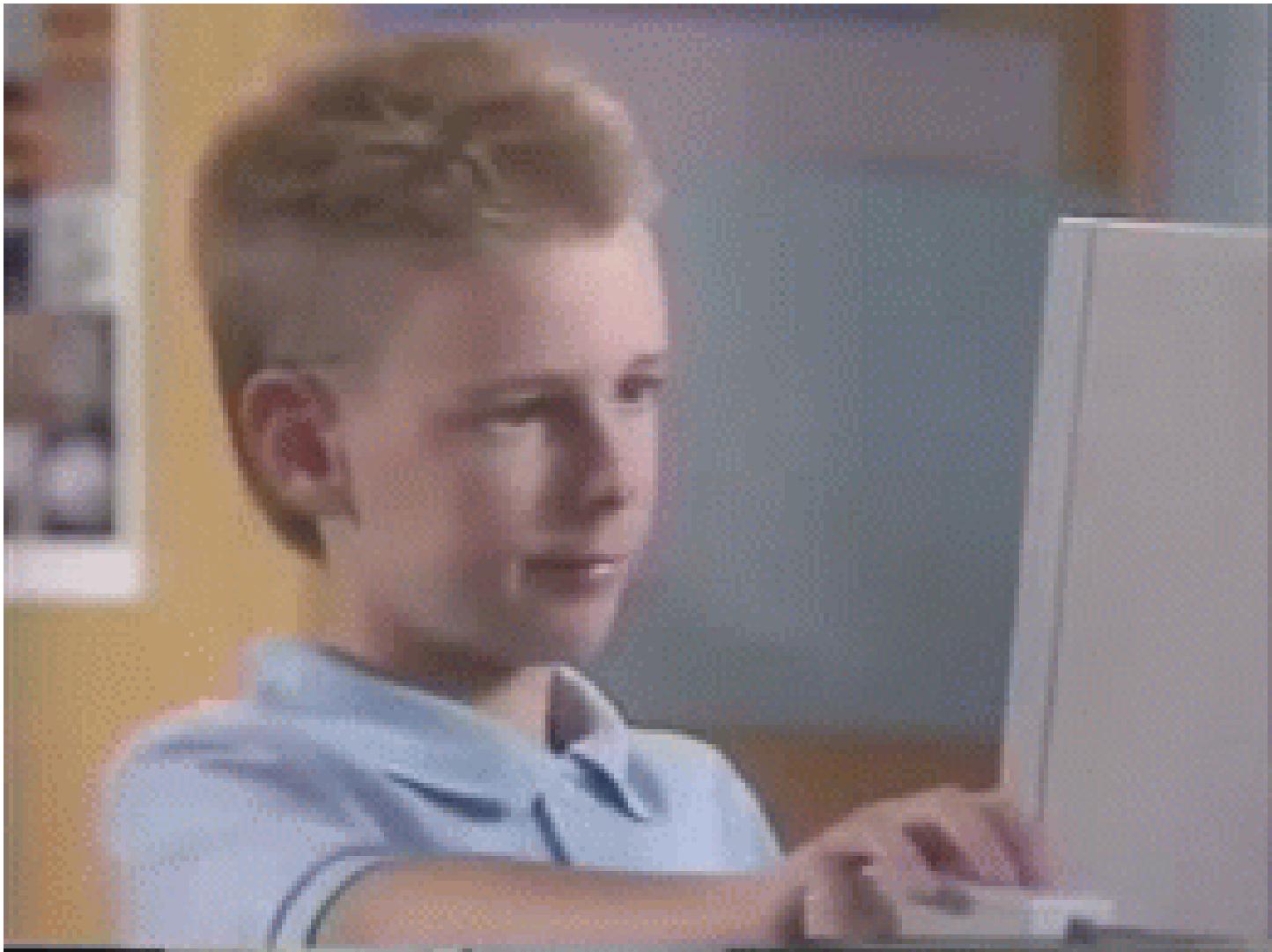
```
loadtest http://service.pinchito.es:3000/d -n 2000 -c 100
```

¿En qué se diferencian?

¿Cómo se portan la latencia y la respuesta de rps?

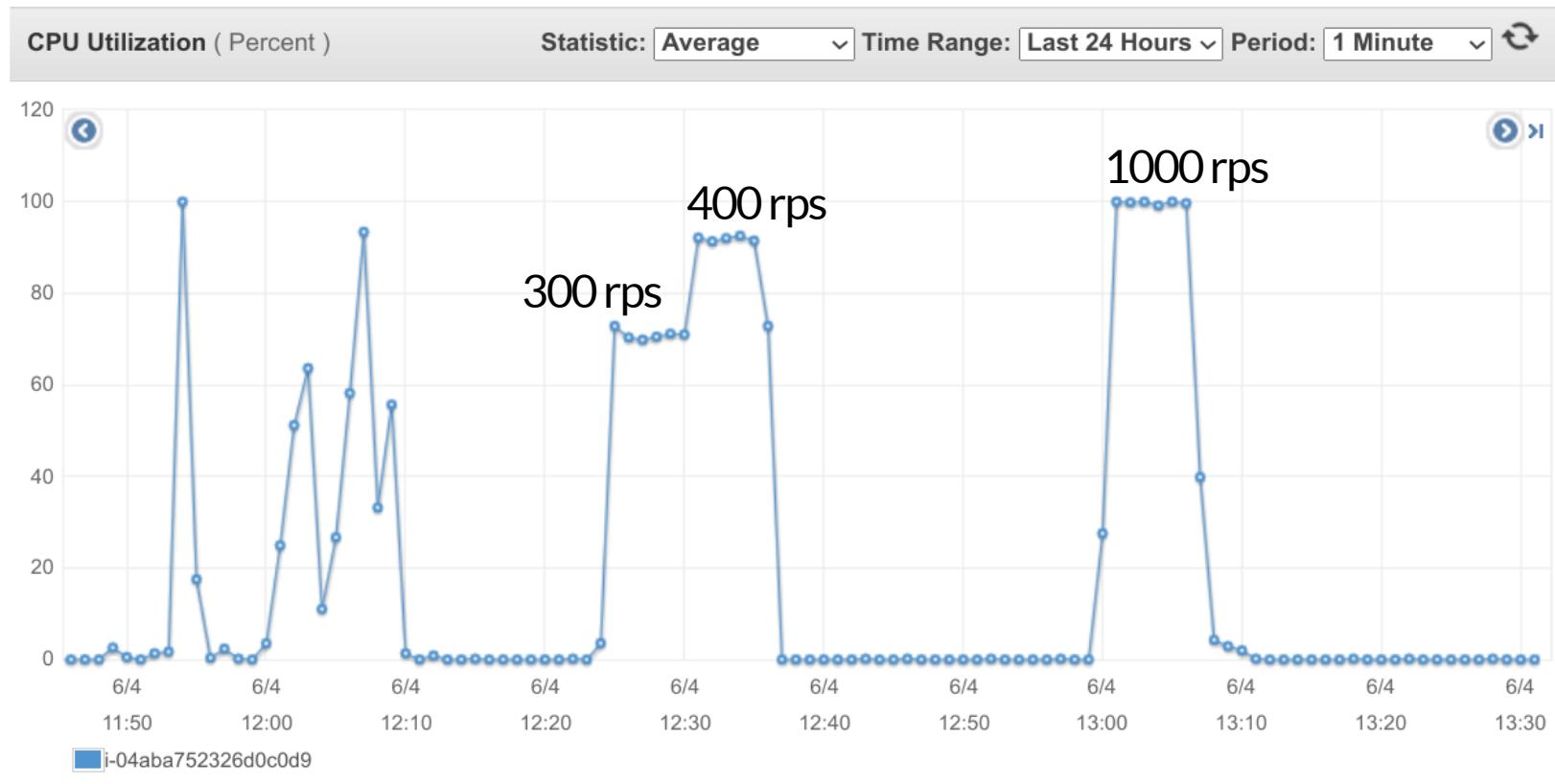


SUCCESS!



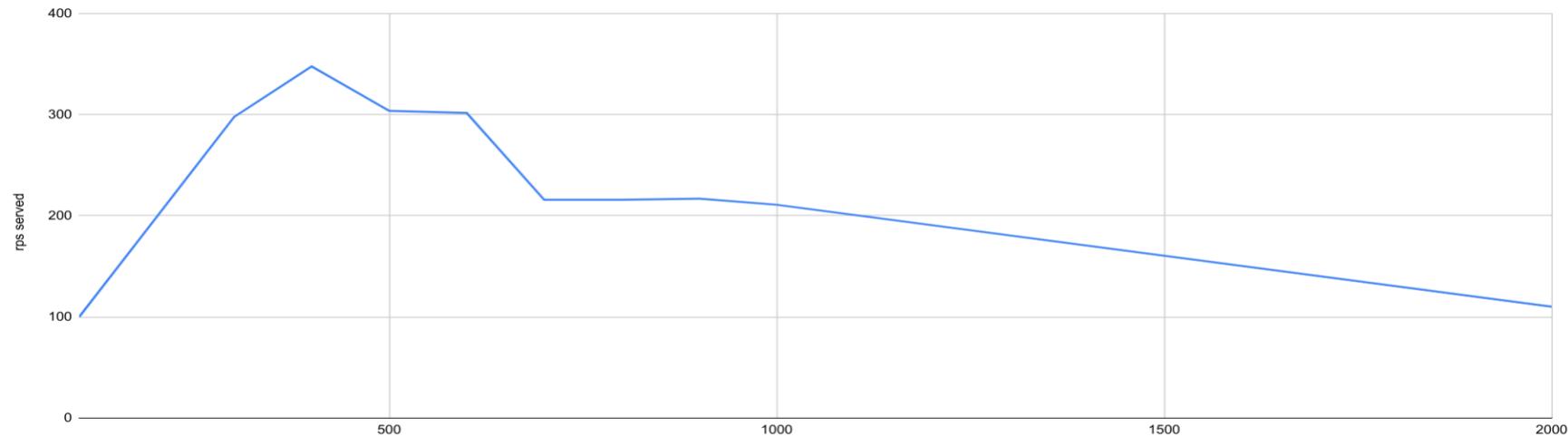
¿QUE RECURSO SE HA AGOTADO?

Gráfica de CPU de AWS:



RPS VS THROUGHPUT

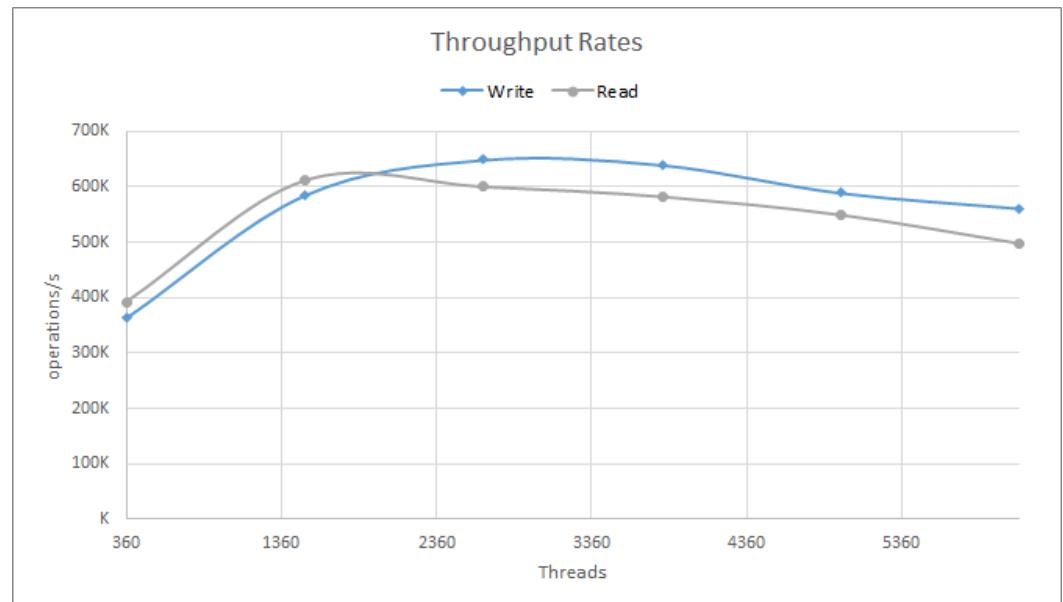
rps served vs rps in



Fuente

Fuente

Throughput Rates



PERFILES DE ESCALABILIDAD

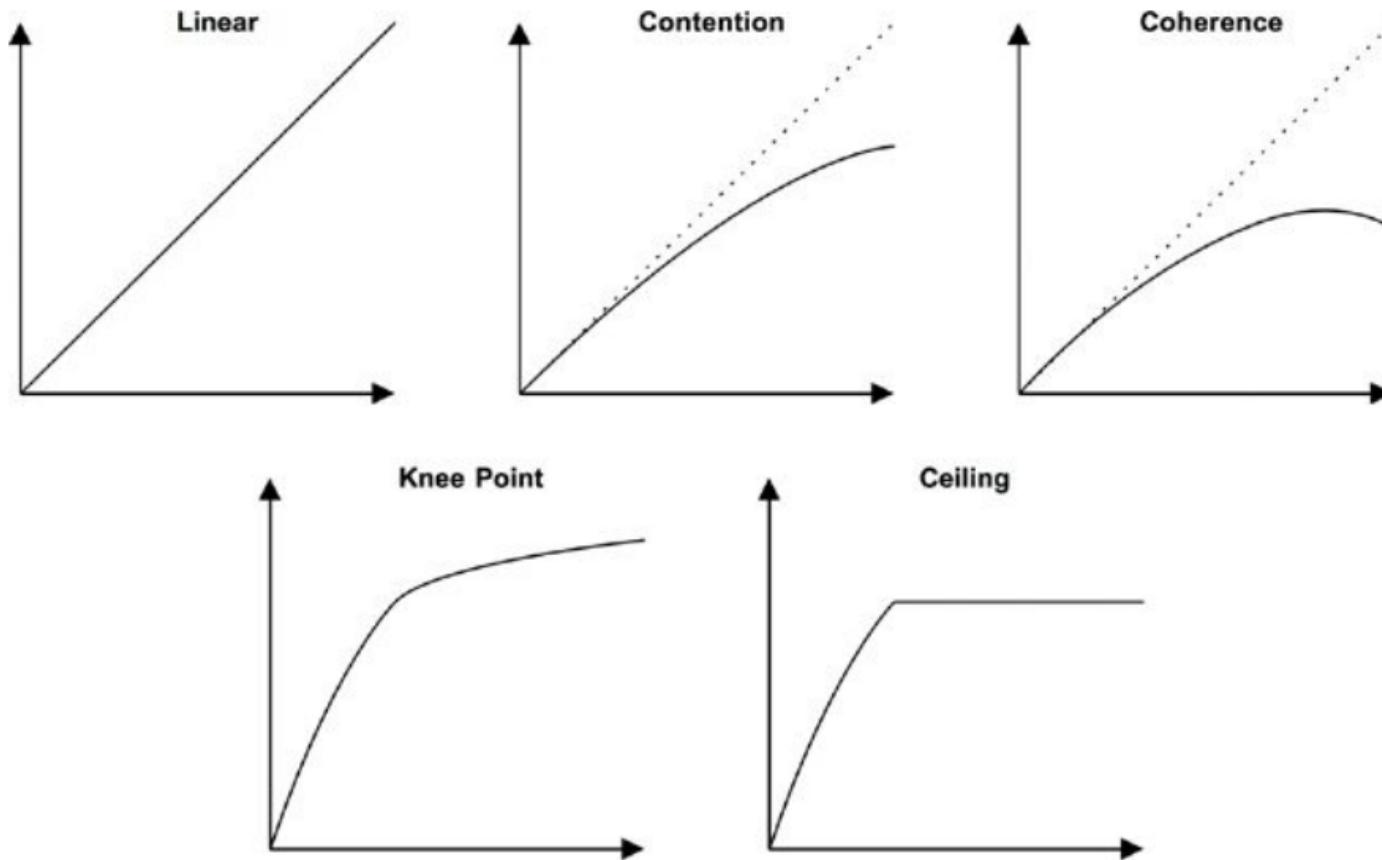
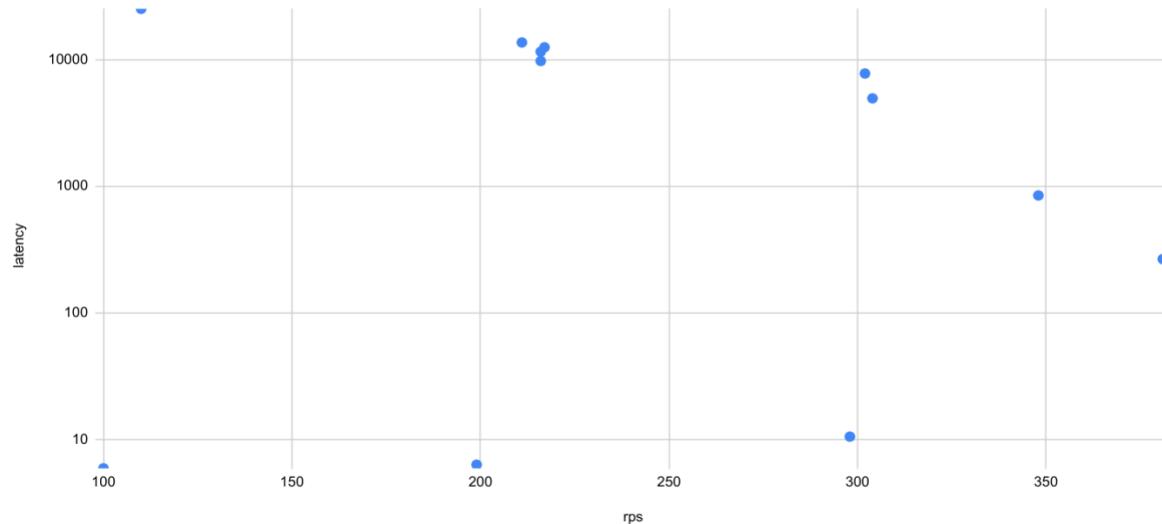


Figure 2.16 Scalability profiles

Brendan Gregg: Systems Performance

LATENCIA VS RPS

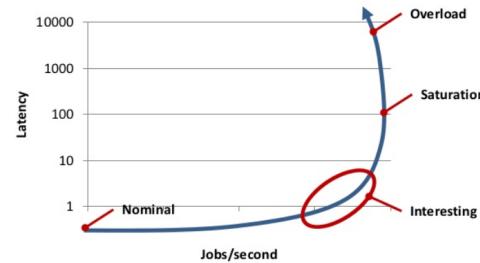
latency vs rps



Fuente

Fuente

The latency curve



LEY DE LITTLE

Little, 1952 - 1960

El número medio de peticiones en vuelo L es igual a:
la tasa de peticiones por segundo λ
por el tiempo medio por petición W .

$$L = \lambda W.$$

[Wikipedia](#)

Si incrementamos la concurrencia L ,
el tiempo medio por petición W crece en igual medida.

ESCALADO V ↔ Y H ⇔

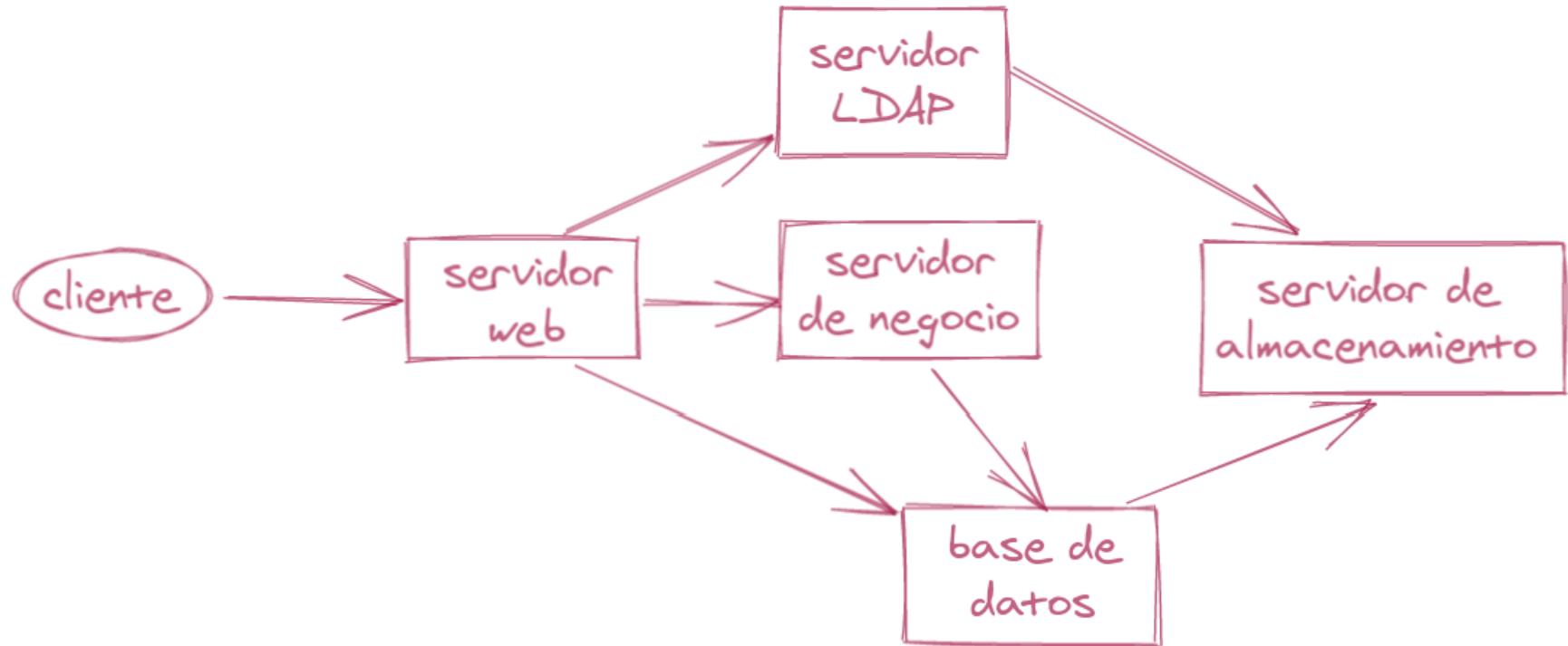


COMIENZOS DIFÍCILES



IBM mainframe

SERVIDORES ESPECIALIZADOS



LAS TIPICAS CABINAS



Y ENTONCES LLEGO GOOGLE



Cabina de almacenamiento, 1996

ESCALADO VERTICAL ⇕

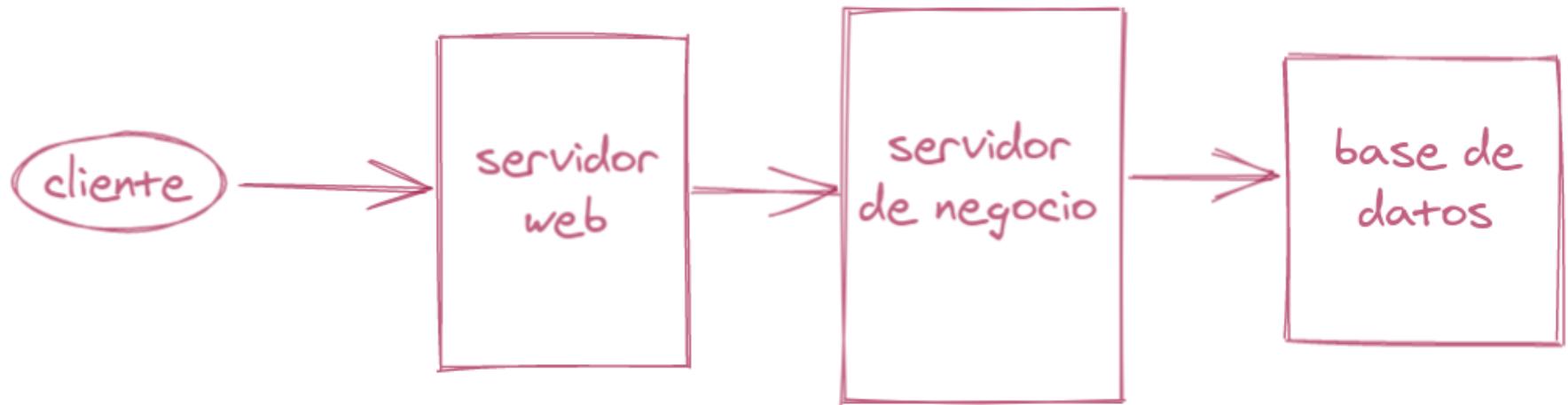
Comprar una máquina más gorda

Y otra más gorda

Así hasta que se acaban las máquinas

Es difícil volver a una máquina más pequeña 😅

ESCALADO VERTICAL ⇕



SSHHH...

Hace muchas décadas que los superordenadores son...
clusters (racimos) de máquinas pequeñas



IBM Blue Gene/P: 164k cores, 2007

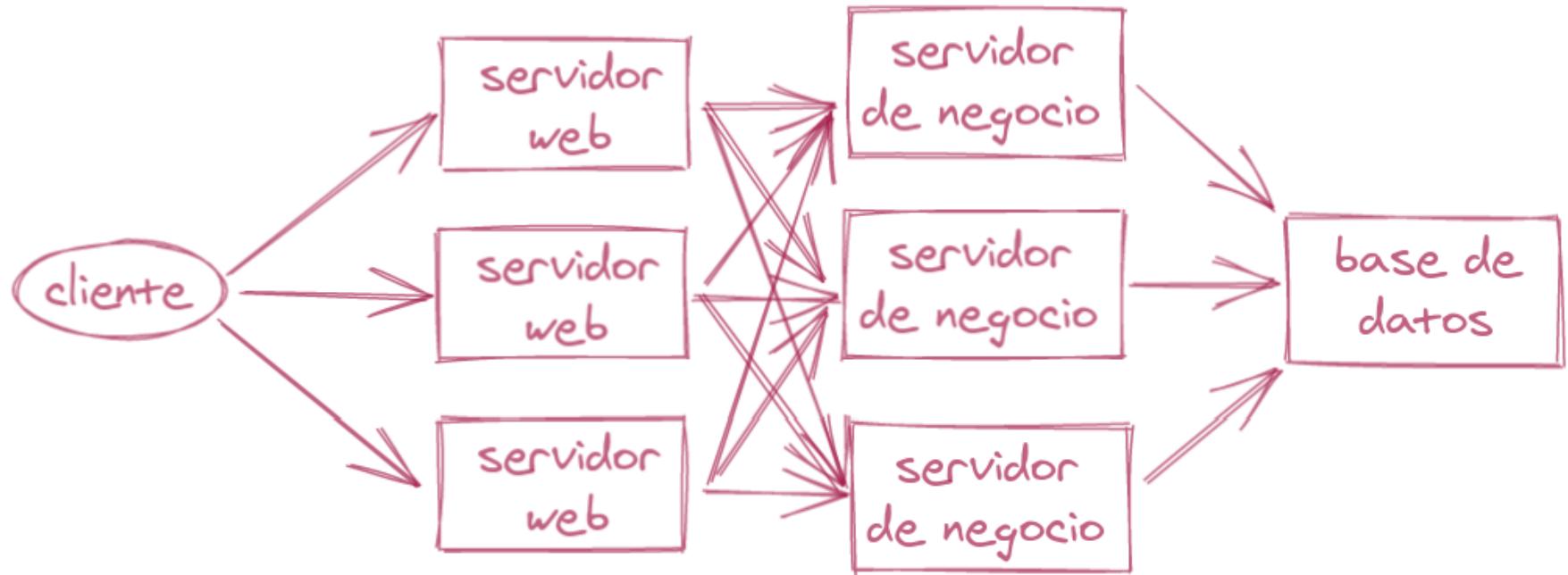
ESCALADO HORIZONTAL ⇔

Colocar muchas máquinas para hacer una función
("provisionamiento")

Añadir y quitar máquinas para escalar

Si falla una máquina se quita del servicio

ESCALADO HORIZONTAL ⇔



EJERCICIO: ALMACENAMIENTO

Diseña un almacenamiento corporativo de 15 TB

Opción 1 ⇔: storage area network (SAN)

Mejor opción de diciembre 2008

Opción 2 ⇔: discos en bruto

Mejor opción de julio 2009



EJERCICIO +

Añade controladores

Opciones RAID

Mide la diferencia económica entre opción 1↔ y opción 2↔

¿Precio final?



EJERCICIO +

Considera estrategias de redundancia

Tolerancia a fallos

Opciones de redundancia: 2x, 3x, ?

Considera estrategias de escalado

¿Cómo afectan al precio?



SUCCESS!



Chuck Norris

ESTRATEGIAS H ↔

Balanceo (servidor)

Balanceo (cliente)

Afinidad

Independencia

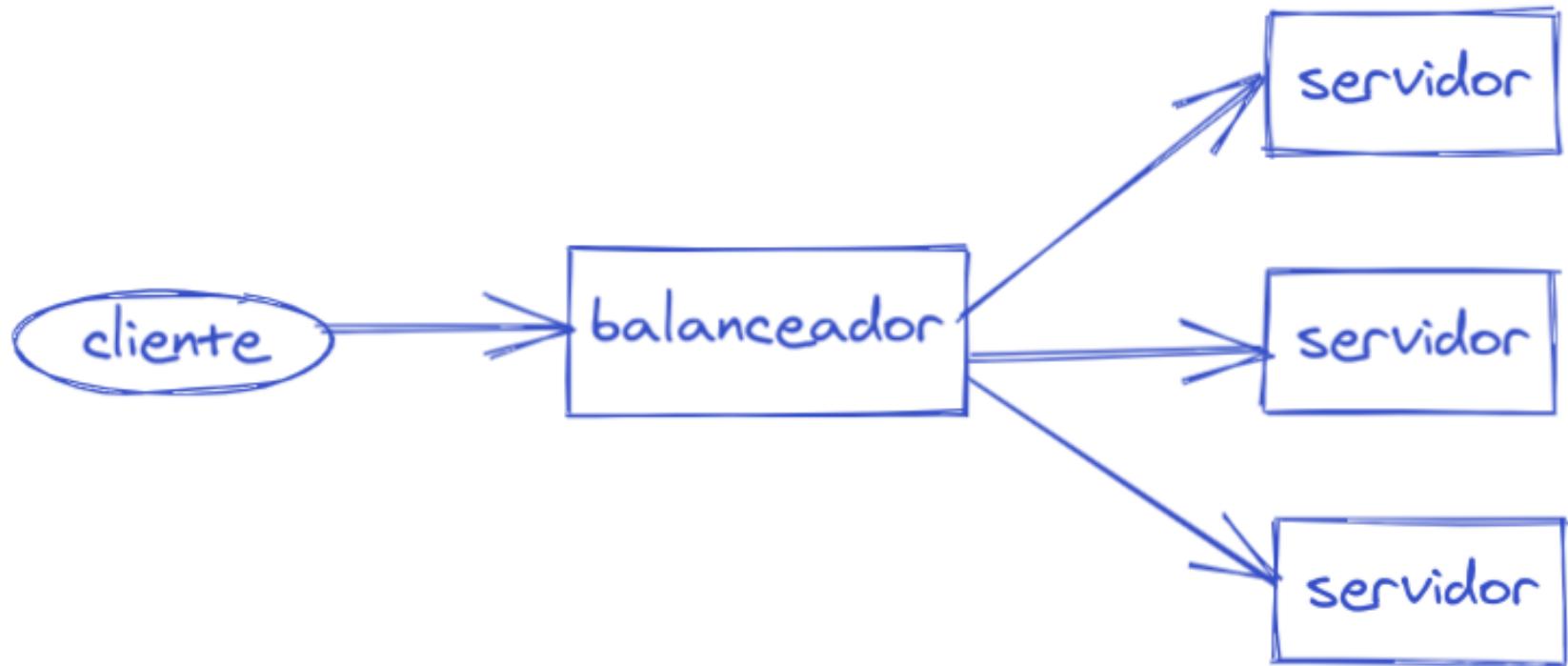
Sharding

Clustering

Replicación

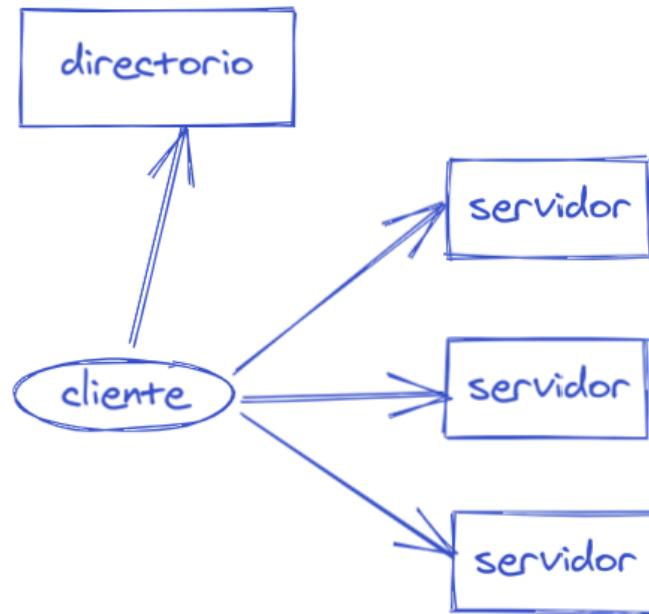
Colas

BALANCEO EN SERVIDOR



Ejemplo: ELB de AWS, Cloud Load Balancing de Google

BALANCEO EN CLIENTE



Ejemplo: cliente de Facebook

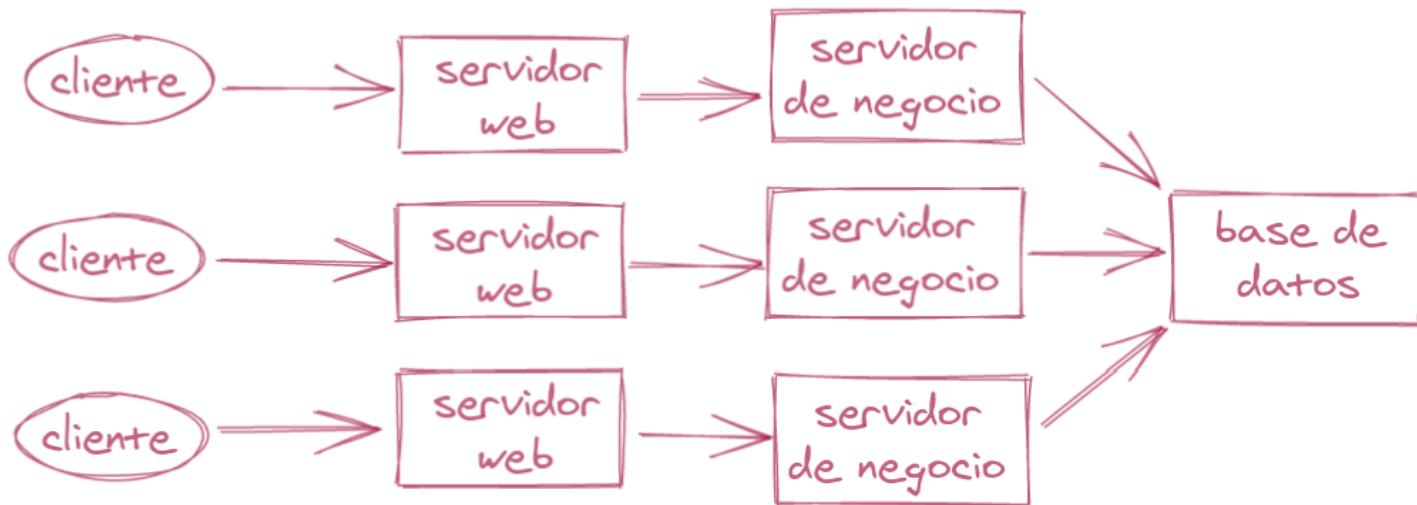
Elige el endpoint para la API en el propio navegador

Ejemplo: balanceo por DNS

AFINIDAD ⇔

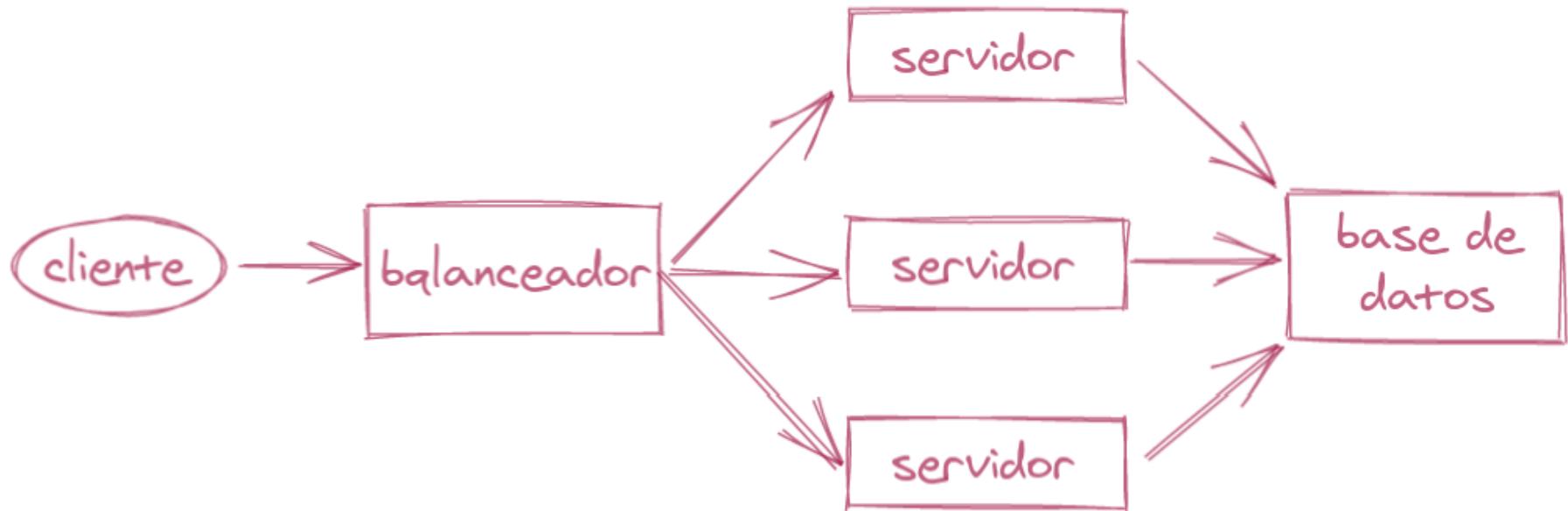
Por cookie o geográfica

Necesita un router sofisticado
En cliente o en servidor



INDEPENDENCIA \leftrightarrow

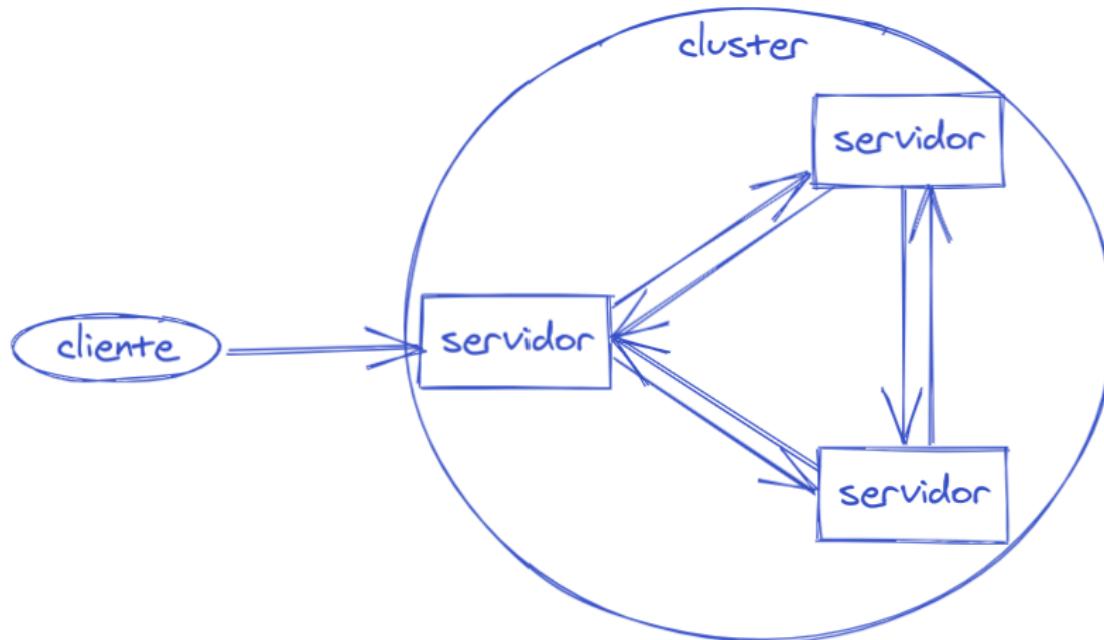
Balanceo neutro (o ciego)



CLUSTERING ⇔

Término genérico (racimo):
hacer una máquina de muchas

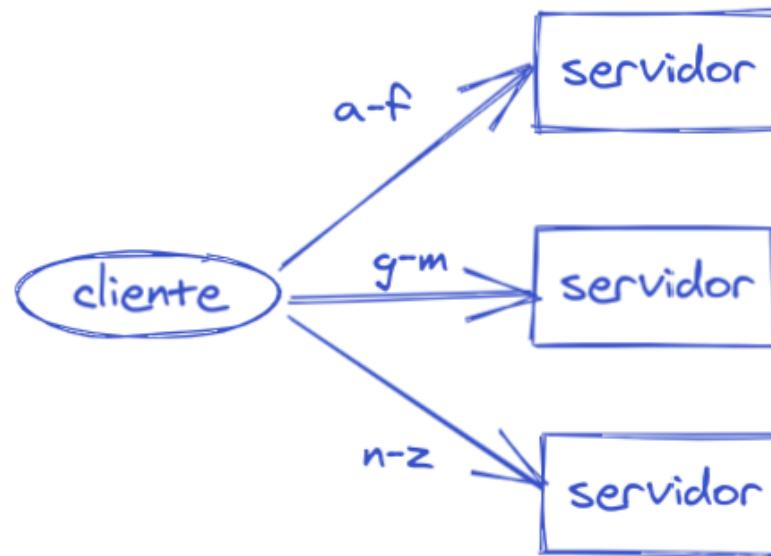
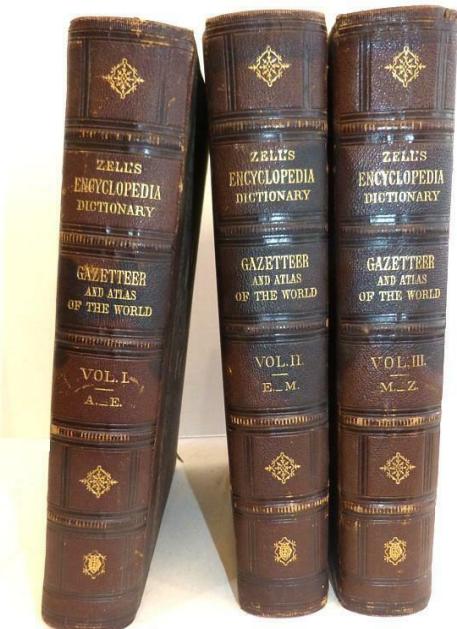
En bases de datos suele significar:
se puede acceder a cualquier máquina del grupo



SHARDING \leftrightarrow

Balanceo por clave

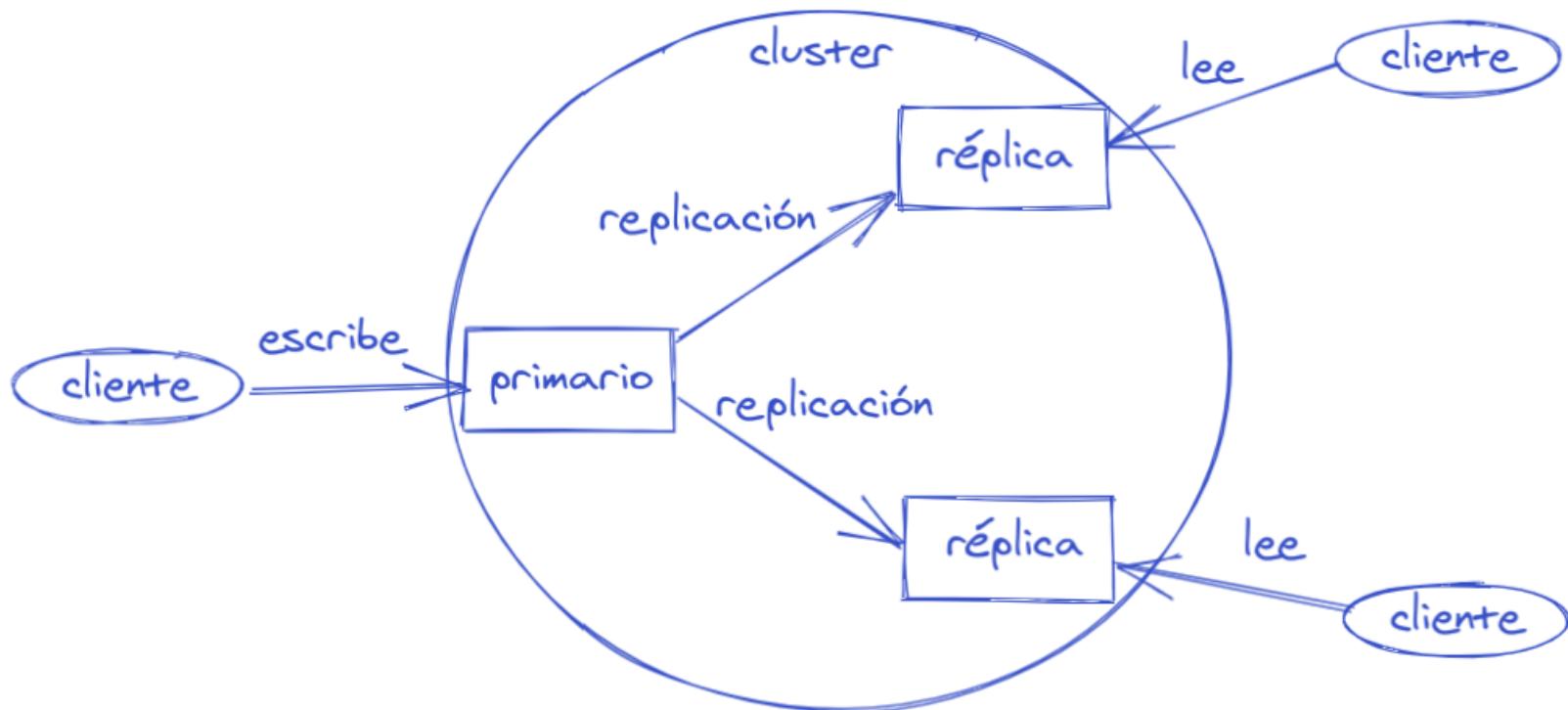
Necesita un algoritmo de reparto (hashing)



REPLICACION ⇔

Un servidor primario (escribir) y varias réplicas (leer)

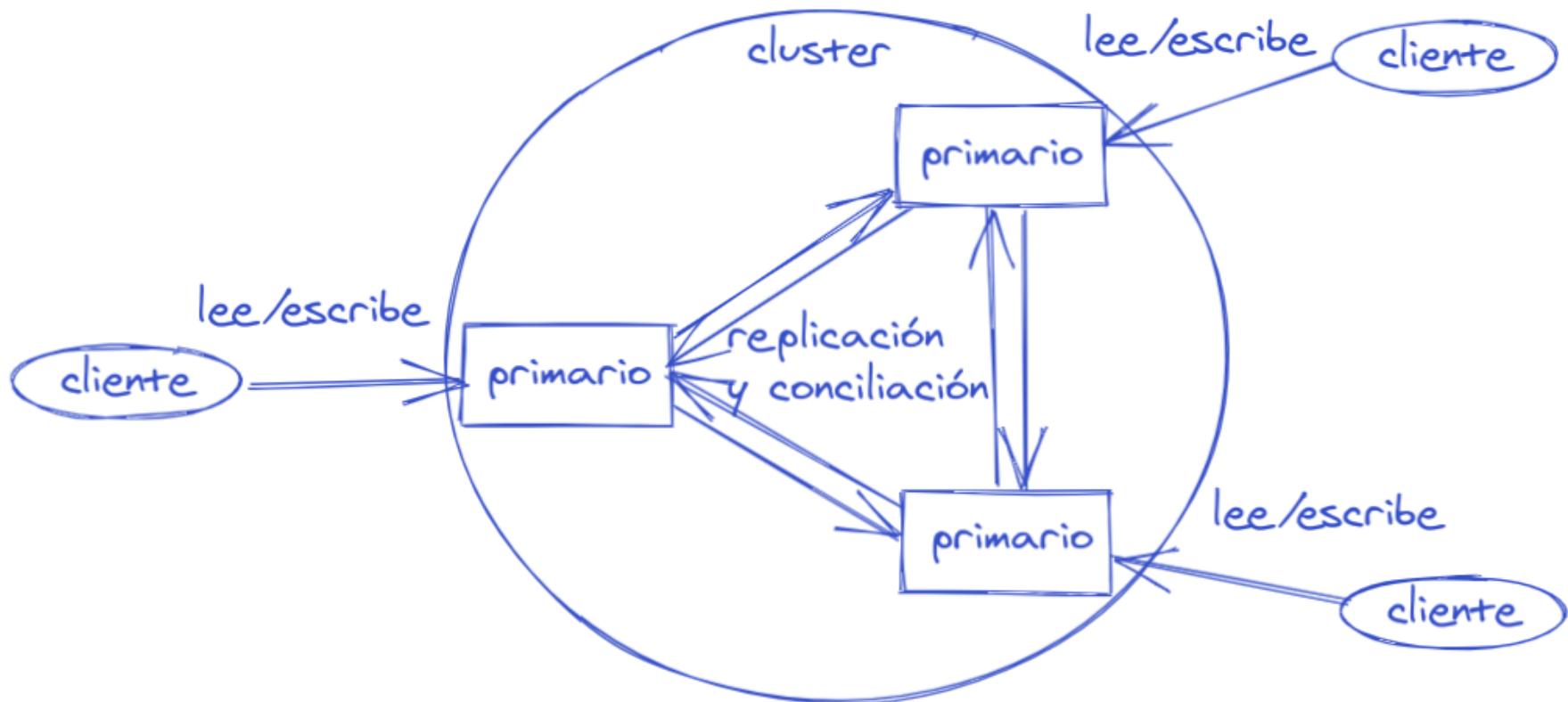
Útil cuando lees > escribes



REPLICACION ACTIVA ⇔

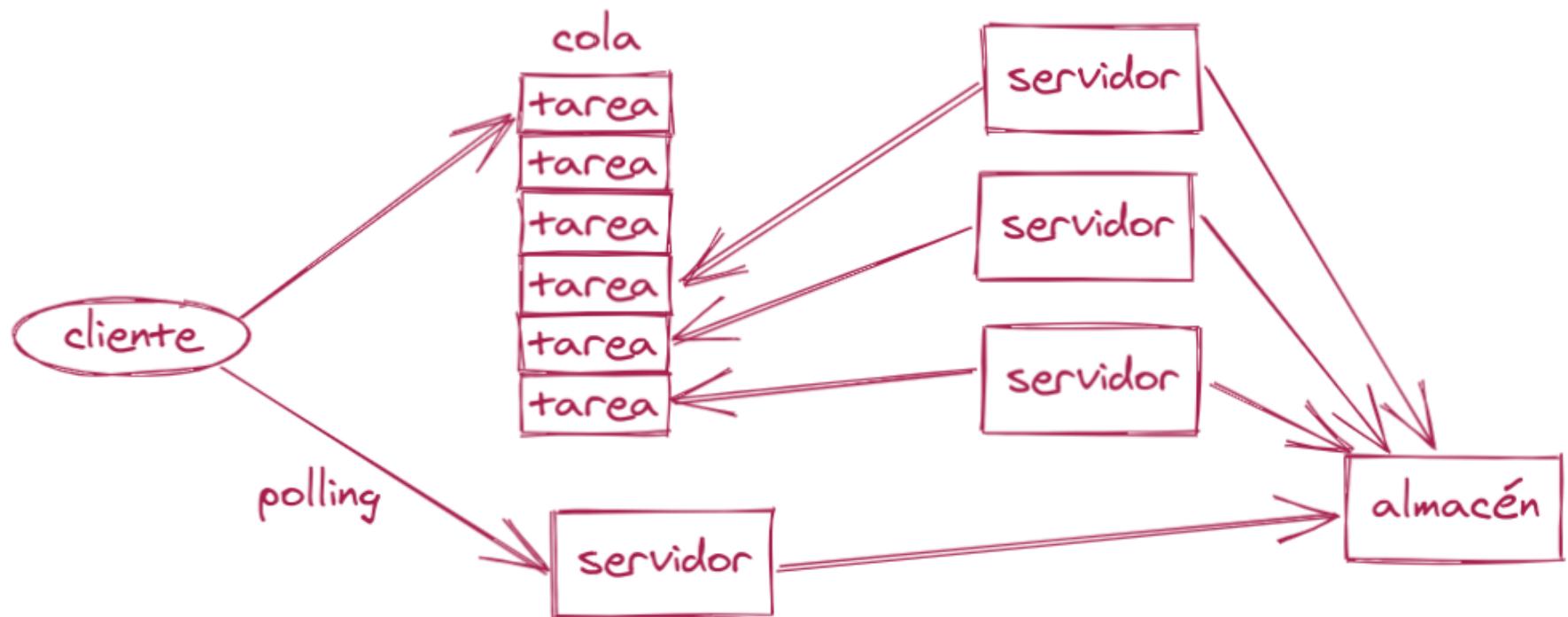
Activo-activo, múltiples primarios...

Necesita conciliación



COLAS ⇔

Permite independizar la producción del consumo de tareas



Mecanismo de *polling*, *poll* = encuesta
(NO *pooling*, *pool* = piscina)

EJERCICIO: ALMACENAMIENTO ESCALABLE

Trabajas para un buscador en enero de 2000

Tienes que almacenar el índice

Diseña una estrategia de escalado

Asume índice = tamaño de las páginas

Usa discos de la época



EJERCICIO +

10 KB por página

50 millones de páginas

4 millones de búsquedas al día

10 términos de búsqueda máximo

Tiempo objetivo de 0.1 segundos por búsqueda



EJERCICIO +

$$50M \text{ páginas} \times 10 \text{ KB} = 500 \text{ GB}$$

Disco más barato: Seagate ST317242A, 17.2 GB, \$152

$$32 \text{ discos} \times 16 \text{ GB} = 512 \text{ GB}, \$4864$$

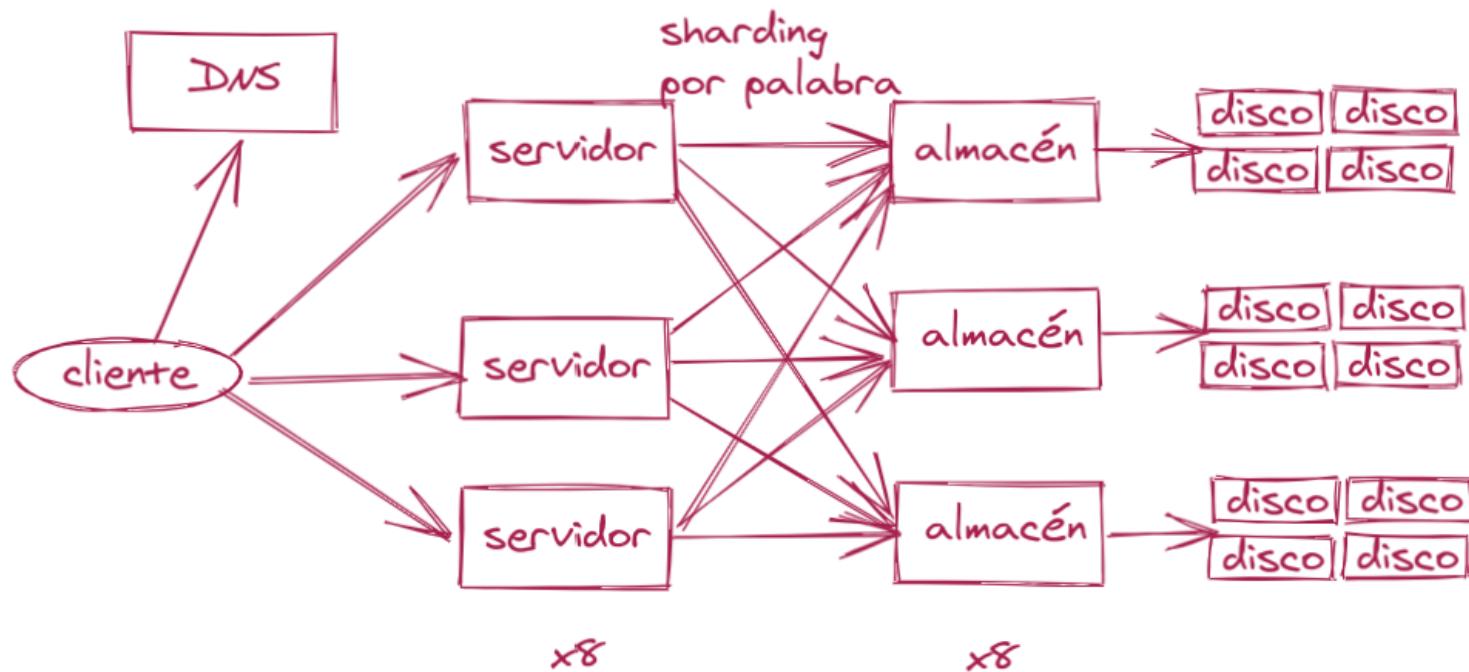
$$8 \text{ servidores} \times 4 \text{ discos} = 32 \text{ discos}$$

$$4M \text{ búsquedas} \times 100 \text{ ms} = 400k \text{ segundos} = 4.6 \text{ servidores}$$

Contando hora punta: al menos 8 servidores



EJERCICIO +



100 ms para ~5 términos de búsqueda

Tiempo medio de consulta al almacenamiento < 20 ms



EJERCICIO +

Tiempo de consulta: *seek time* + 1/2 vuelta + formateo

Seek time: unos 8 ms

Disco de 7200 rpm: 8 ms por vuelta

Total consulta: >12 ms

Parece factible; mejor agregar una capa de caché



WELL DONE!



SISTEMAS DISTRIBUIDOS



TEOREMA CAP

Eric Brewer, 1999 (demostrado por Gilbert&Lynch, 2002).

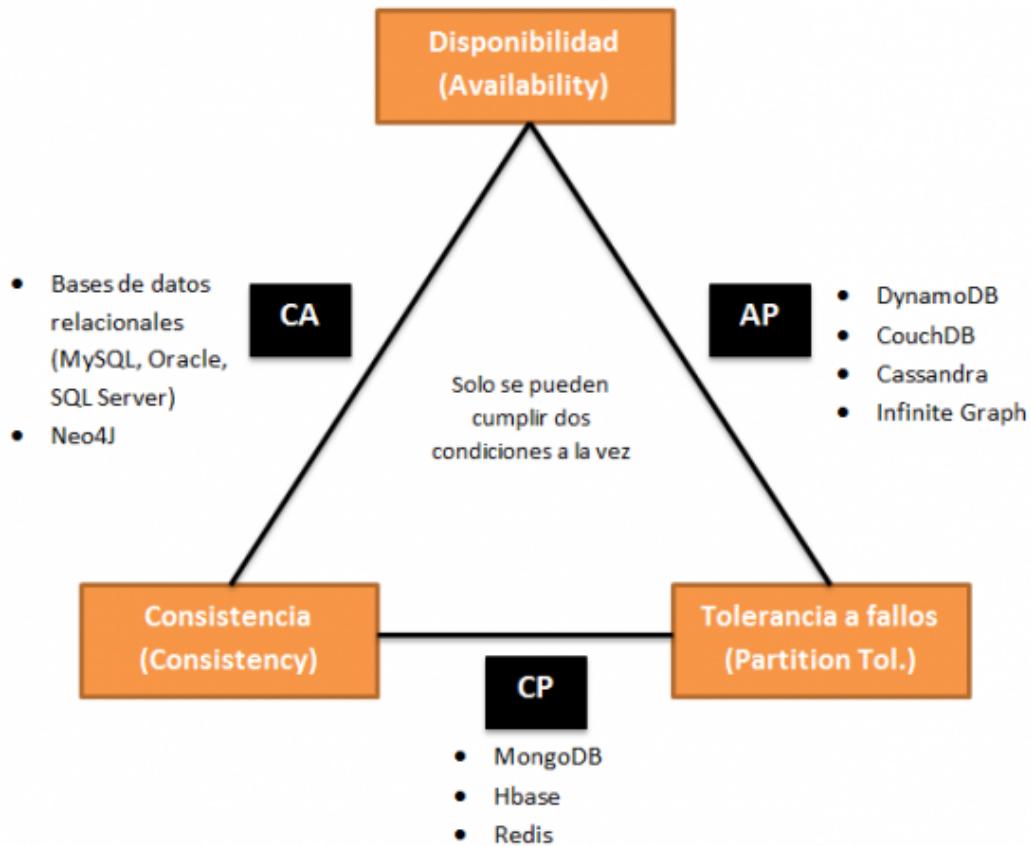
C: consistency, consistencia

A: availability, disponibilidad

P: network partition, partición de red

Elige dos.

¿COMO AFECTA A LAS BASES DE DATOS?



Clasificación según el teorema CAP (mal)

CAP 2012

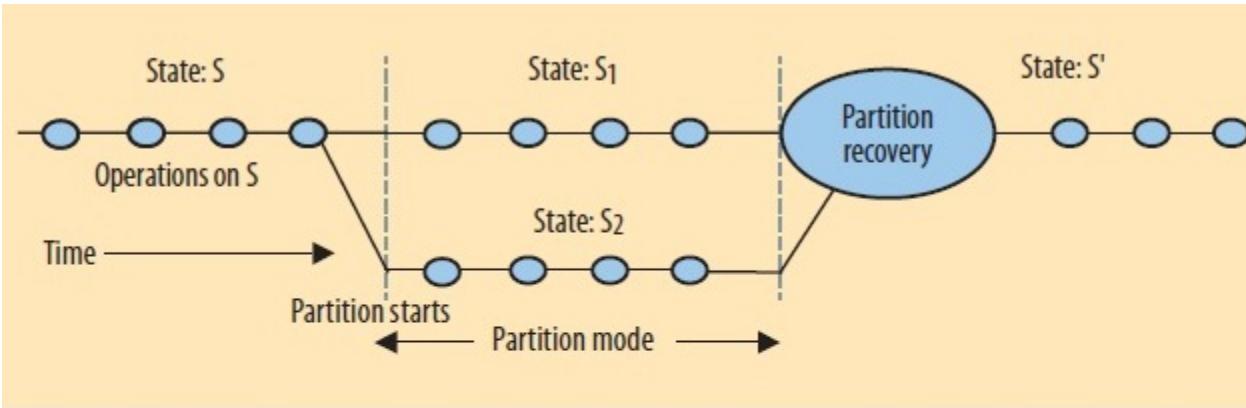


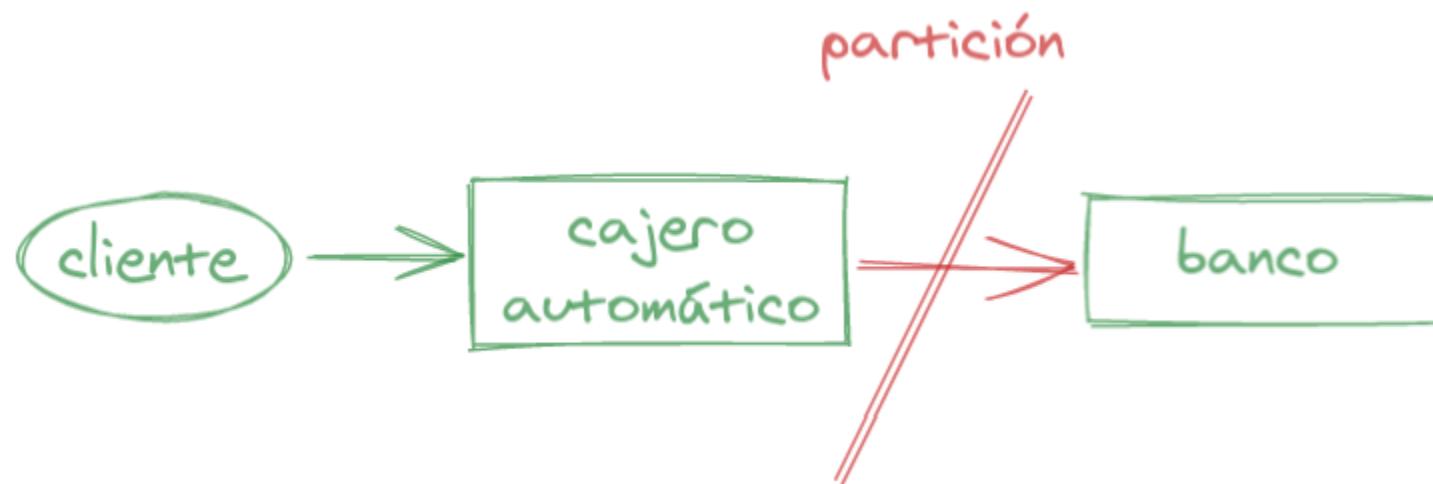
Figure 1. The state starts out consistent and remains so until a partition starts. To stay available, both sides enter partition mode and continue to execute operations, creating concurrent states S_1 and S_2 , which are inconsistent. When the partition ends, the truth becomes clear and partition recovery starts. During recovery, the system merges S_1 and S_2 into a consistent state S' and also compensates for any mistakes made during the partition.

Eric Brewer, 2012

EJERCICIO: DISEÑA UN CAJERO AUTOMÁTICO

No hay conexión con el banco (hay partición de red)

¿Le damos el dinero al cliente?



EJERCICIO +

Ante una P, hay que decidir entre A y C

Availability (disponibilidad): soltamos la pasta

El cliente puede no tener saldo

Consistency: no soltamos la pasta

Al fin y al cabo somos una entidad financiera seria



EJERCICIO +

Solución: opción oculta A-

Para no sacrificar la disponibilidad, damos con un límite

Mantenemos un registro de dinero sacado en la tarjeta

Al volver la conexión, reintegramos la consistencia



EJERCICIO +

Diseña un algoritmo de conciliación

Vale también para corregir transacciones duplicadas
y todo tipo de errores

Pista: los bancos llevan trabajando en ello siglos

Pista: prueba con operaciones atómicas



¡A TOPE!



PUNTO UNICO DE FALLO

SPoF, *Single Point of Failure*

Según algunos es malísimo

Es habitual y rutinario y útil y práctico

EFECTIVIDAD EN COSTE

Esperamos no sólo que el coste no suba al escalar

¡Esperamos que el coste por petición baje!

EJERCICIO: CONTROL DE COSTES

Amazon ofrece ElastiCache for Redis

Un Redis "completamente gestionado"

En realidad cobra por instancia

¿Cuál es el sobrecoste asociado?



EJERCICIO +

Precio de ElastiCache for Redis comparado con instancias
Instancias cache.r5.large, cache.r5.12xlarge

Bajo demanda, zona Europa (París)

¿Mejora con nodos reservados vs instancias reservadas?

¿Vale la pena el sobrecoste?



EJERCICIO +

Incluye el coste de mantener Redis

Supón un coste fijo de 3h/mes + 1h/instancia
a €50/hora

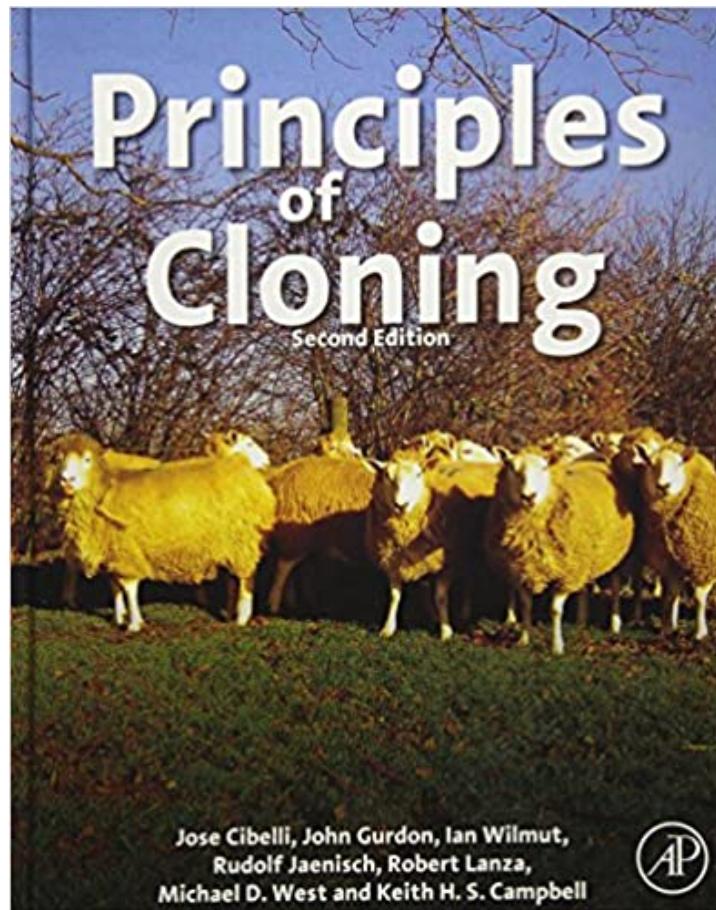
El coste de mantener ElastiCache for Redis es la mitad



GOOD JOB!



REPLICACIÓN DE SERVIDORES



REPOSITORIOS

Todo debe estar "repositado"

Incluyendo la configuración

SCRIPTS

Replicar el sistema de forma automática

También todo en repo!

Para más sofisticación: [Ansible](#)

IMAGENES

AMIs: lock-in de Amazon AWS

Docker: formato estándar

Imágenes portables

Usa el kernel local (en Linux)

SERVERLESS

Una alternativa atractiva a los servidores alquilados

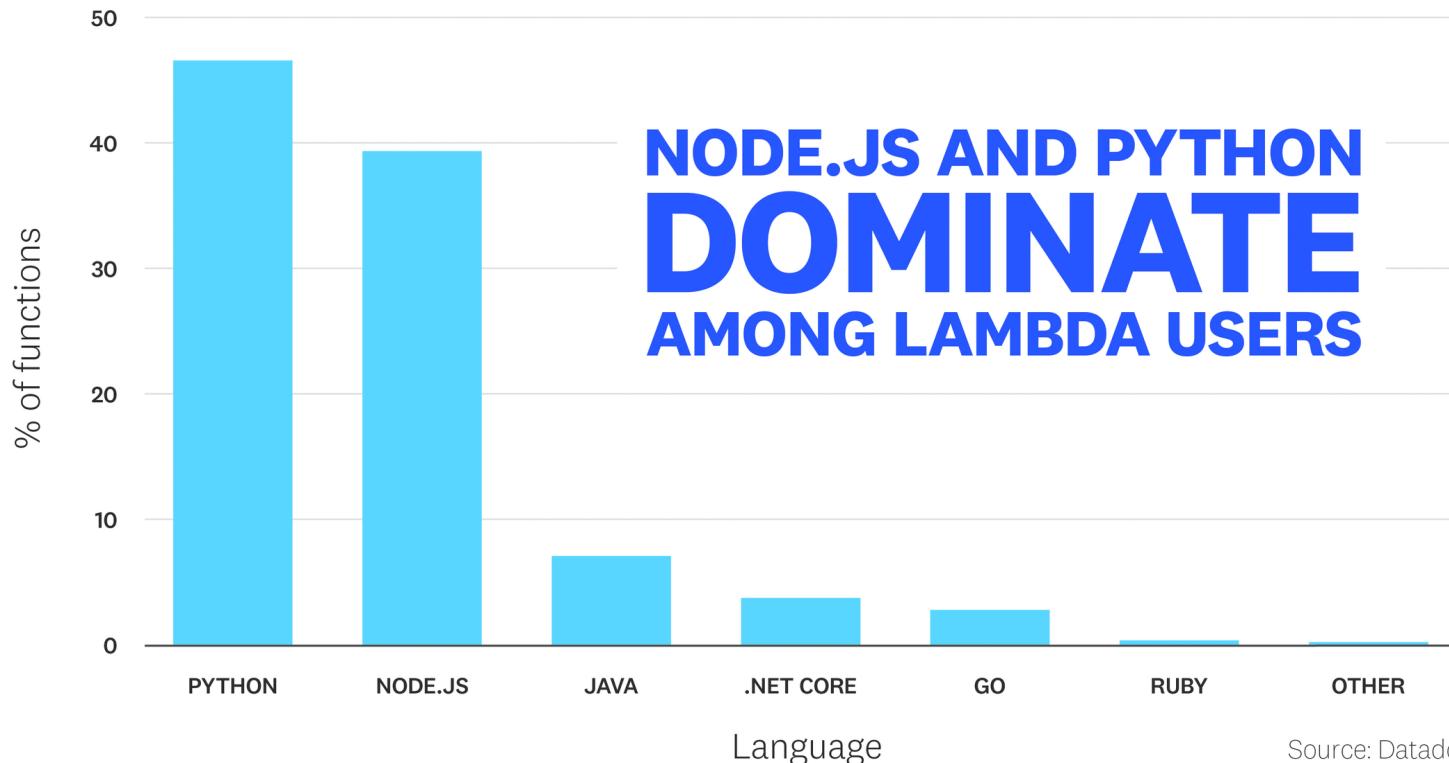
El paquete se sube a un servidor y se replica según haga falta

Escalado fácil (a veces trivial)

You want efficient application scaling? Go serverless!

LENGUAJES EN AWS LAMBDA

Most Popular Languages by Distinct Functions



Source: Datadog

Fuente

EJERCICIO: CONTROL DE COSTES (II)

Un servicio tarda 100 ms por petición

El servicio, en Node.js, consume hasta 500 MB de memoria

Recibimos de media 100 req/s, con picos de hasta 300

Calcula el coste mensual de correrlo en [AWS Lambda](#)



EJERCICIO +

Ahora calcula el coste en servidores EC2 equivalentes

Podemos crear varios procesos para servir peticiones

El consumo de CPU típico del proceso es del 25%

Asume un carga del 50% de CPU



EJERCICIO +

Suma 3h/mes de mantenimiento por servidor a €50/h

Calcula el ahorro de costes en servidores de EC2

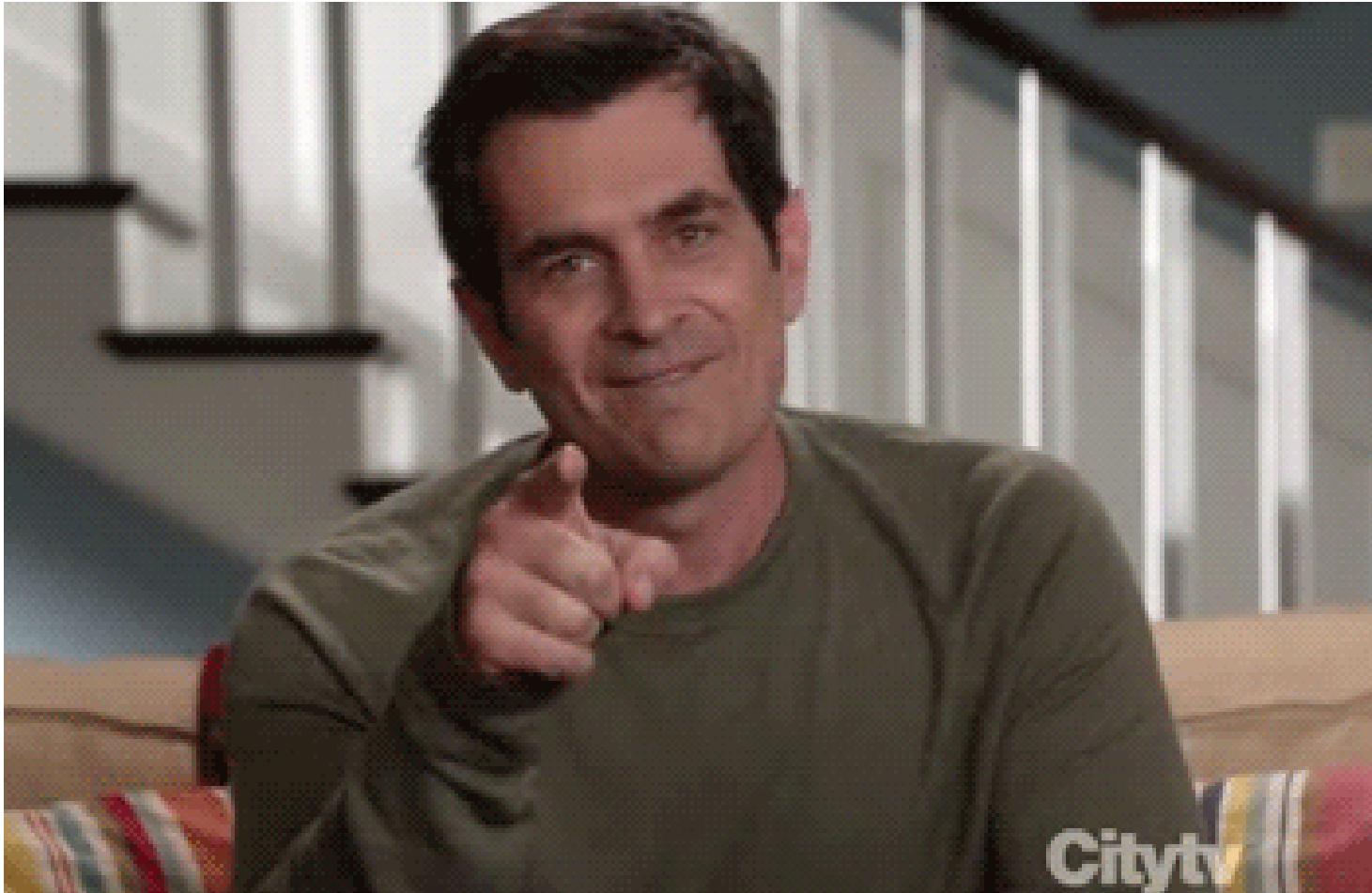
Calcula el ahorro si implementamos un balanceo
con la carga al 80%

¿A partir de qué nivel de carga sale rentable EC2?

¿Cómo nos aseguramos de ser rentables en el tiempo?



YOU DID IT!



AWS LAMBDA: CONTRAS

Coste

Versiones anticuadas

Despliegue primitivo

Falta de control

EJERCICIO: DISPOSITIVO DE PRESENCIA

Antiguamente: "dispositivo de hombre muerto"

Usar AWS Lambda para crear un servicio



Debe escribir un valor en Redis

El valor caduca a los cinco minutos



EJERCICIO +

Provisionar un Redis (el más pequeño)

Provisionar un servicio en Lambda

Escribe un valor testigo

Duración: cinco minutos



EJERCICIO +

Configurar para que corra cada tres minutos



AWESOME!



You got it, dude.

BIBLIOGRAFIA

Eric Brewer: CAP Twelve Years Later: How the "Rules" Have Changed

Brendan Gregg: Systems Performance: Enterprise and the Cloud

John Allspaw: Web Operations: Keeping the Data On Time

HighScalability.com: Favorite posts on HighScalability