

Uso de métricas



Curso de escalabilidad v2, día 3

Qué veremos hoy

Cuatro señales de oro

Optimización de rendimiento

Métricas comerciales: SLAs, uptime

Objetivo de uptime. Presupuesto de incidencias

Incógnitas desconocidas (unknown unknowns)

¿Qué es una métrica?

Una **magnitud** + un **sistema de medición**

¿Qué es, y cómo se obtiene?

¿De dónde sale?

¿Qué **significa**?

Métricas internas y externas

Internas: se miden en la propia máquina
Afectan a la medida

Externas: se miden desde fuera
Pueden no ser tan precisas

Toda métrica externa tiene su réplica interna

El problema de la medida

La propia medición afecta al resultado

Ocurre a todos los niveles

Si la medición es externa, entonces puede no ser tan precisa

Algunos problemas se resuelven al observarlos 🤔

Cuatro señales de oro

Tráfico: peticiones por segundo

Errores: tasa de fallos en el sistema

Latencia: tiempo en responder

Saturación: si el sistema está cerca del límite

Fuente: [SRE Book](#)

Peticiones

Una petición es parte de un intercambio de información

Petición + respuesta
(request + response/reply)

Orientado a **conexiones de red**

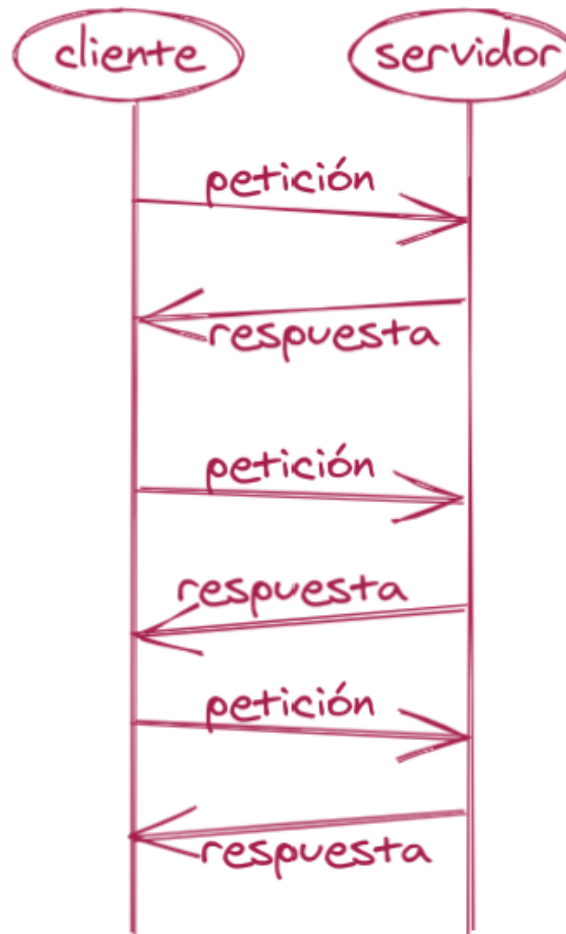
Sockets

HTTP

APIs

Sería equivalente a una consulta (query) a base de datos

Petición + respuesta



Medición de tráfico

Las peticiones se cuentan **externamente**

También se puede contar **internamente** mediante eventos de log

Durante un intervalo de tiempo (típicamente segundo o día)

Se suele registrar el tipo de resultado

- Status HTTP
- Éxito o **error**

Ejercicio: ¿Cómo medir el tráfico?

Discute en grupo cómo medir el tráfico

¿Cómo lo has medido en el pasado?

¿Cómo se podría medir mejor?



¡Chachi piruli!



Tasa de error

En un sistema distribuido no siempre hay "arriba" y "abajo"

Los sistemas pueden fallar intermitentemente

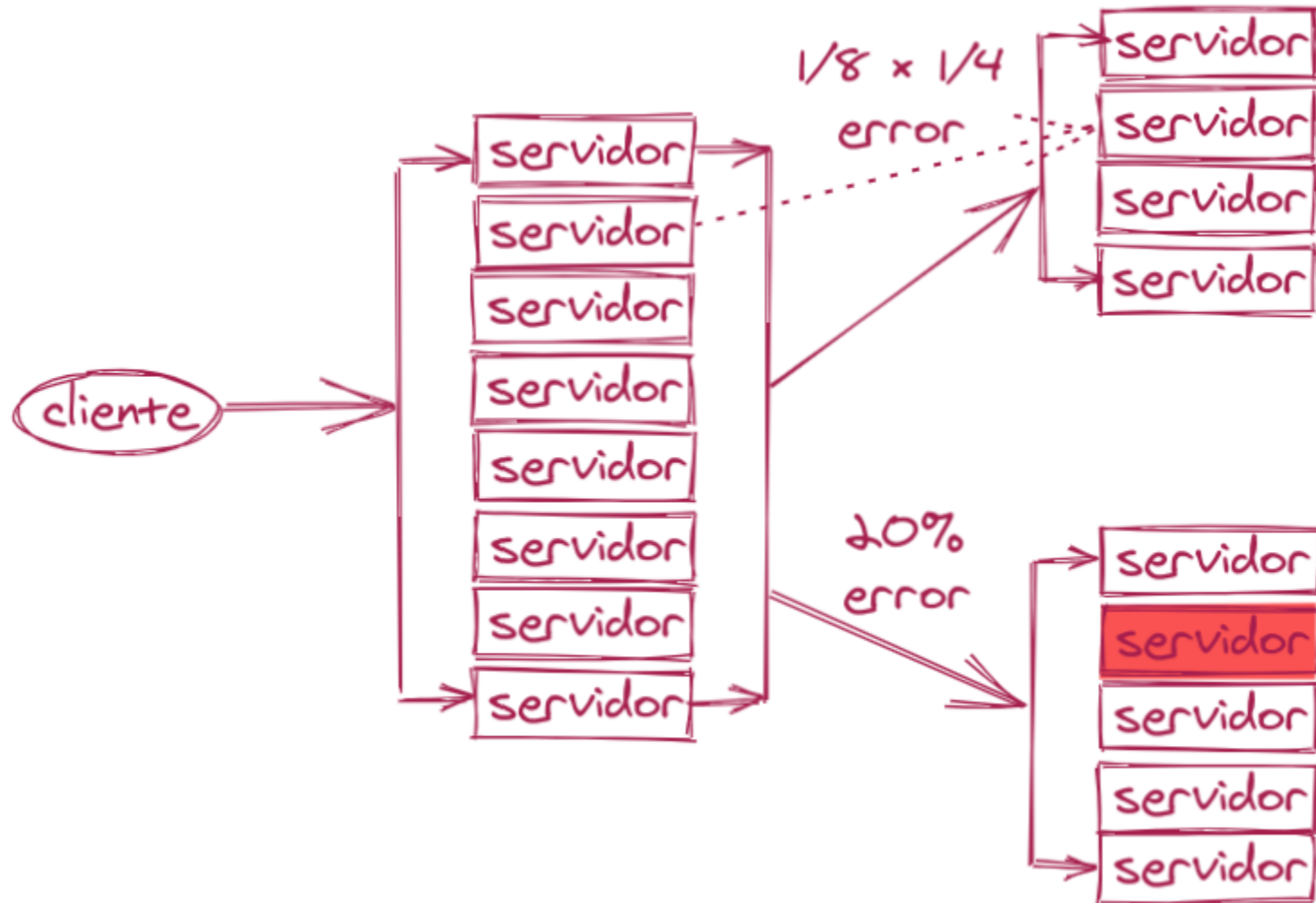
La disponibilidad se calcula como:

$$\frac{\text{peticiones exitosas}}{\text{total peticiones}}$$

La tasa de errores = 1 - disponibilidad

Complicado de comprobar como cliente 😅

Cuando no hay sistema arriba/abajo



Latencia

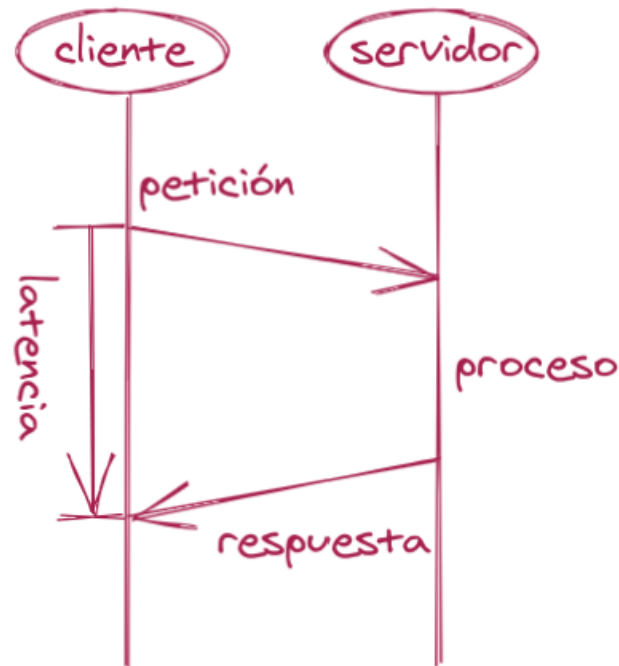
El tiempo de espera entre el inicio de una petición y la recepción de la respuesta

Medido **externamente** en el cliente
normalmente en milisegundos

Incluye tiempos de red

Cuanto más cerca, más preciso

Medición de latencia



Normalmente basta con medir **milisegundos**

Ejercicio: Midiendo latencias

Lanza una petición contra nuestro adorado servicio
<http://service.pinchito.es:3000/a>

Mide la latencia en ms

$\text{Latencia} = \text{tiempo}(\text{fin}) - \text{tiempo}(\text{inicio})$



Ejercicio +

Código:

```
const request = require('basic-request')

measure().catch(console.error)

async function measure() {
  const start = Date.now()

  await request.get('http://service.pinchito.es:3000/a')

  const elapsed = Date.now() - start
  console.log(`Elapsed: ${elapsed} ms`)
}
```

Hacer npm install en el directorio [latency](#)



Ejercicio +

Ahora resolvemos un problema de rendimiento

(En la práctica pasamos a medir `http://service.pinchito.es:3000/d`).

Vuelve a medir la latencia

¿Qué te sale ahora?



Awesome!



Evento

Mensaje enviado de forma **asíncrona**

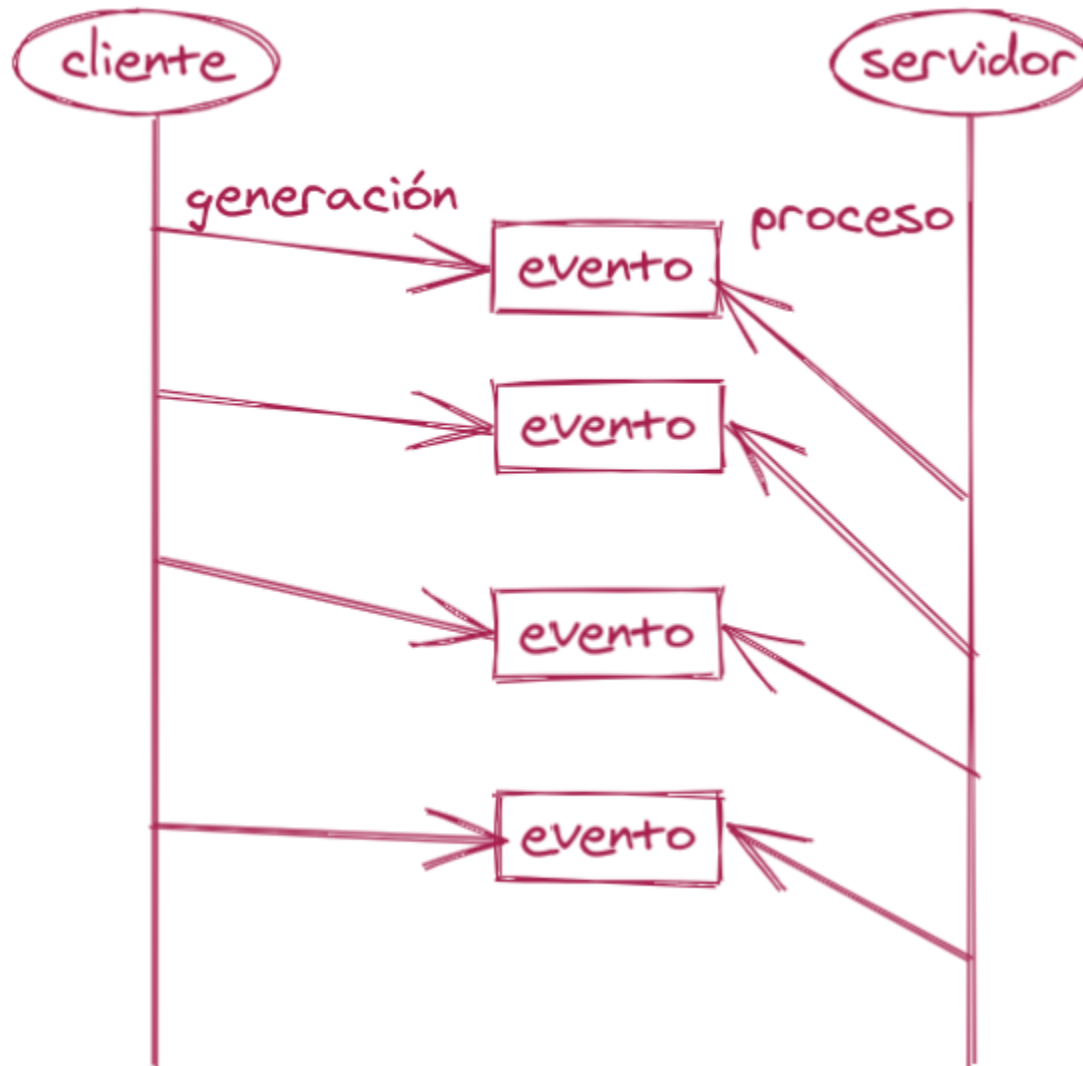
Medido **internamente** en la máquina que procesa

Puede generar una acción o no

Indica procesamiento retardado

- *Fire & forget*
- *Polling*

Proceso de eventos



Saturación

Uso de **recursos finitos**: CPU, red...

En unidades absolutas o porcentaje

Medición de red: 16 Mbit/s

El porcentaje indica mejor la **saturación**

Medición de uso de procesador

Interna (por CPUs): top puede medir 400%

Externa (por máquina): EC2 nunca pasa del 100%

¿Por qué las 4 señales de oro?

Tráfico: importante para la estabilidad y el coste

Errores: crucial para el cliente

Latencia: crucial para el cliente

Saturación: importante para la estabilidad y el coste

La tiranía de las medias

Twitter is a daily reminder that 50 percent of the population are of below average intelligence.

Richard Dawkins

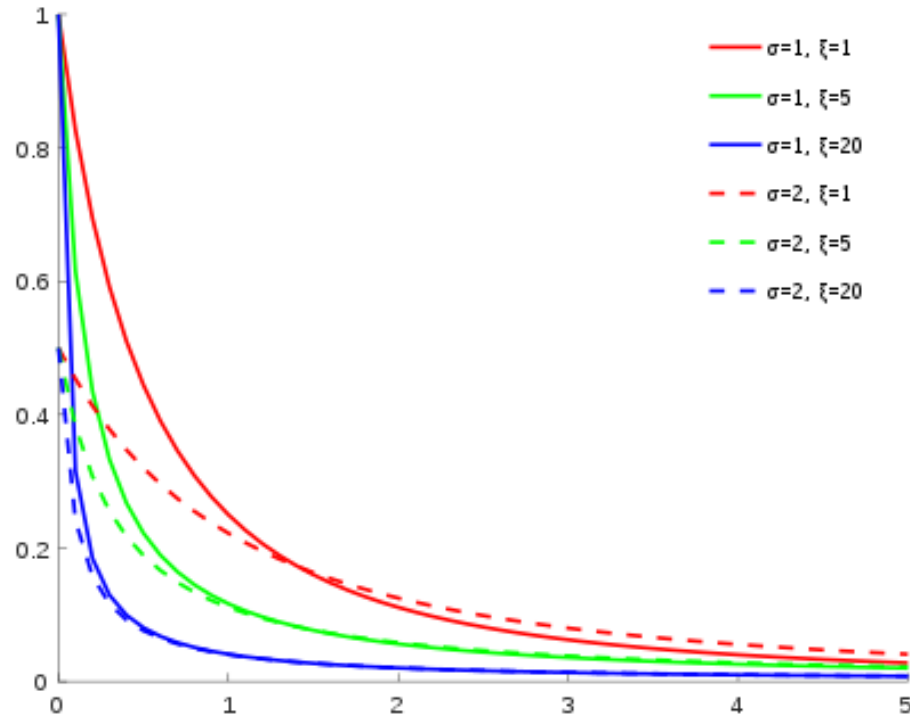
La inteligencia no es lo mismo que el cociente intelectual

La métrica **nunca reemplaza a la magnitud** que se quiere medir

¡Falso!

O no necesariamente cierto

Distribución de Pareto generalizada



Optimización de rendimiento



¿Por qué no escala un sistema?

Cuellos de botella

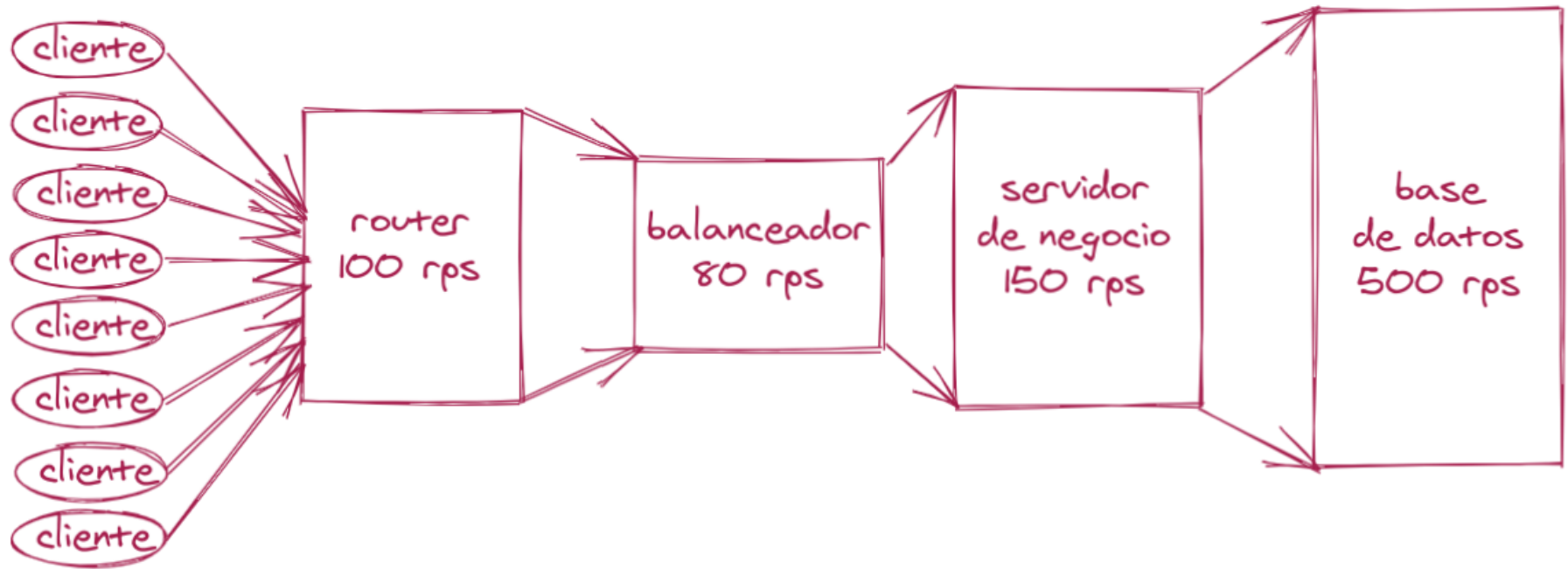
Recursos que se agotan

Aprovechamiento de recursos

Bloqueos

Cuellos de botella

Un sistema no puede rendir más que el componente menos capaz



Corolario: un componente con demasiado exceso de capacidad está **sobredimensionado**

¿Qué recursos se saturan?

Respuesta corta: todos

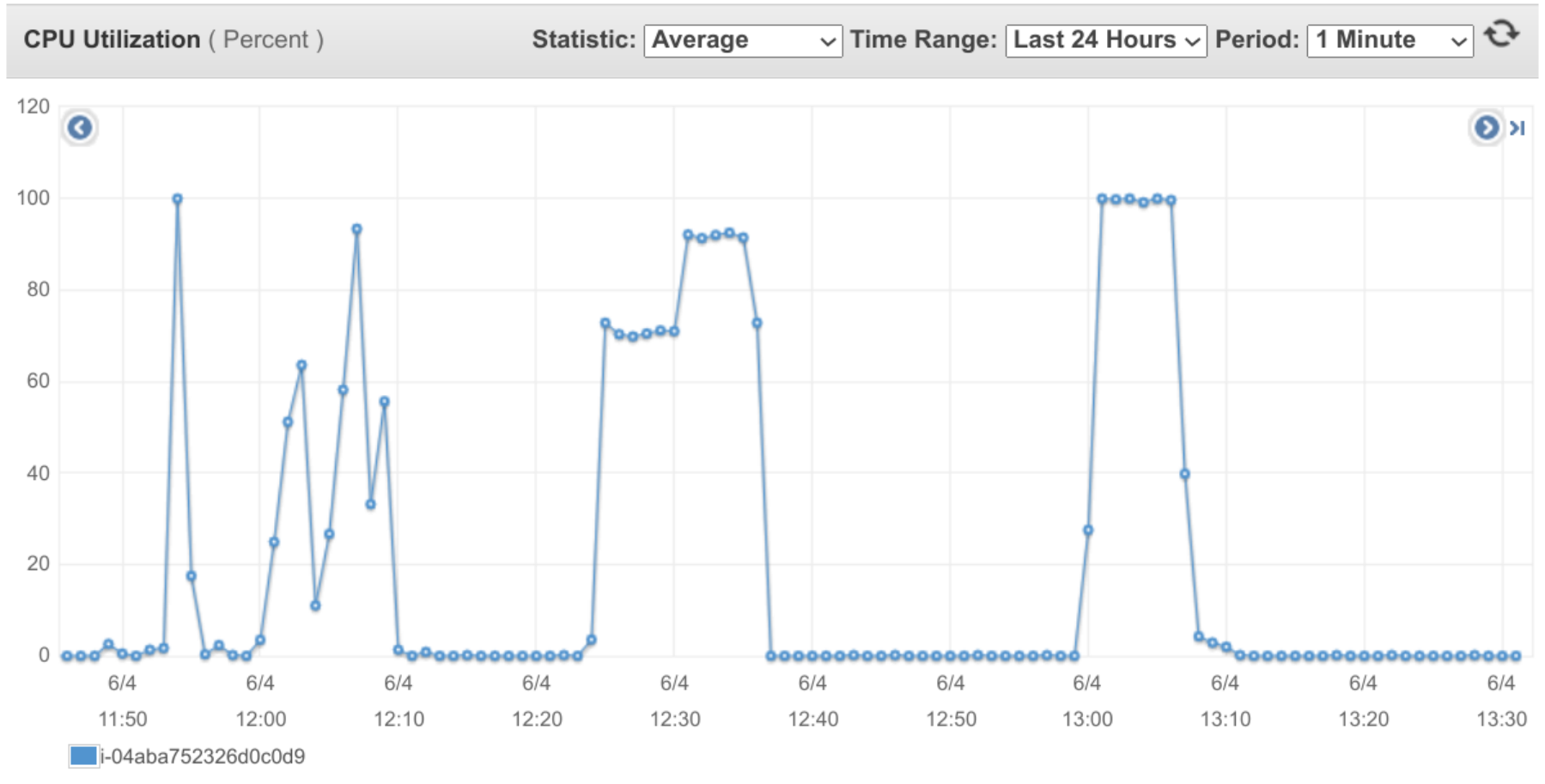
- Memoria *
- CPU
- Ancho de banda
- Descriptores de fichero
- Buffers de entrada/salida *
- ...

Saturación no catastrófica: el sistema se bloquea

*Saturación **catastrófica**: el sistema se rompe

¿Nos enteramos siquiera de que se satura un recurso?

Los clásicos: CPU



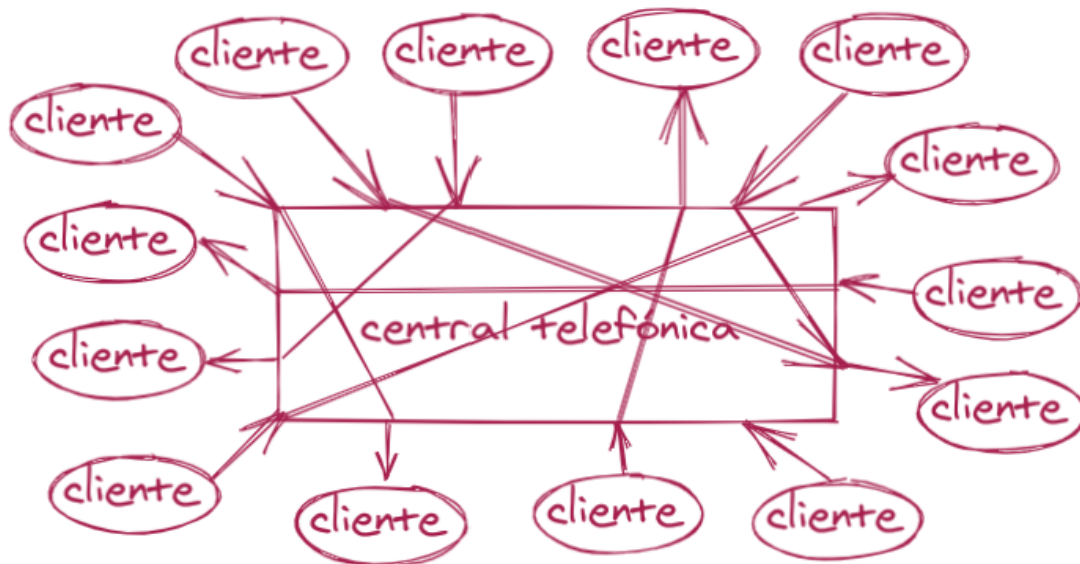
Independencia

No requiere bloqueos

Inmutabilidad

Programación funcional

Ejemplo: Erlang



Ejercicio: Programación funcional

Resuelve el siguiente [challenge](#):

Write a function `solve(limit, array)` that will return the number of elements of the array that are larger than the limit.

Usa challengerco.de

O mide el resultado en local con un array de 10k elementos

Escribe una solución funcional pura



Ejercicio +

Ahora escribe una solución con `forEach()`

Y otra estructurada (`for` clásico)

¿Cuál va más rápido?

¿Puedes acelerarlas?



Ejercicio +

Código funcional:

```
function solve(limit, array) {  
  return array.filter(e => e > limit).length  
}
```

Código forEach():

```
let t = 0  
array.forEach(element => {  
  if (element > limit) t++  
})  
return t
```

Código for():

```
let t = 0  
for (let i = 0; i < array.length; i++) {  
  if (array[i] > limit) t++  
}  
return t
```



¡Tecleando!



Bloqueos

Protegen un recurso

Ejemplo: global kernel lock

Cuando no se pueden procesar peticiones, se acumulan

Buffers de entrada y salida
(que también se saturan)

¿Cuándo usar bloqueos?

Modelado de **dependencias** (e.g. juegos)

Programación mono-proceso (Node.js)

Operaciones atómicas

Separación de lectura / escritura

Mutexes

Eventos

Canales

...

Ejercicio: contador de visitas

Somos un servidor de vídeos en 2012

Queremos contar las visitas a cada vídeo

Teníamos un servidor monoproceso con Node.js
escribiendo a múltiples ficheros

Se nos ha quedado pequeño

¿Puedes identificar posibles cuellos de botella?



Ejercicio +

Diseña un contador de visitas que pueda escalar horizontalmente

¿Qué tipo de operaciones son necesarias?

¿Qué llamadas tendrías en la API?

¿Qué estrategia(s) de escalado horizontal usarías?



Ejercicio +

Es 2013, estamos en una loca expansión internacional

Tenemos visitas de todo el mundo

¿Cómo ampliar el contador?

¿Es posible tener un contador preciso y sincronizado?

¿Se te ocurren alternativas?



Ejercicio + (en común)

Diseña ahora un servidor que asigne identificadores únicos
UUIDs

Tenemos un servidor centralizado
Mantiene un contador único

El servidor se colapsa por tráfico
Además abrimos nuevas regiones
¿Se te ocurren otras opciones?



Beep beep



Métricas comerciales



Disponibilidad

El porcentaje de tiempo que está el **sistema levantado**

También conocido como: *uptime, availability*

Valores típicos:

- 99%: caído 14 minutos al día
- 99.9%: caído 10 minutos a la semana
- 99.999%: caído 24 segundos al mes
- 99.99999%: caído 3 segundos al año
- 99.9999999%: caído **3 segundos al siglo**

Ejercicio: Caídas periódicas

Calcula la disponibilidad de estos sistemas:

Banco nacional: caído 1 hora al día

Subastas de marketing: caído 1 hora al mes

Compra online: caído 5 horas al año



Ejercicio +

Tenemos un compromiso de disponibilidad del 99.9%

Queremos establecer una ventana de mantenimiento mensual
¿De cuántos minutos disponemos?

Una intervención nos cuesta dos horas
Tenemos que hacer una intervención
¿Cómo podemos mantener el compromiso?



Ejercicio + (en común)

¡Extra, extra!

Un comercial ha vendido a un cliente un 99.99% de *uptime* 🤖

Tenemos diez servidores que actualizar

Cada uno requiere 30 minutos de *downtime*

¿Cómo mantenemos el compromiso?



Not bad



SLA

SLI: *Service Level **Indicator***

SLO: *Service Level **Objective***

SLA: *Service Level **Agreement***

Chris Jones *et al*, SRE book

SLI

An SLI is a service level indicator—a carefully defined quantitative measure of some aspect of the level of service that is provided.

Indicador de nivel de servicio
(Vamos, una métrica de toda la vida)

Ejemplos:

- Latencia
- Disponibilidad: 99.999%
- Tasa de errores
- Rendimiento (*throughput*, peticiones por segundo)

SLO

An SLO is a service level objective: a target value or range of values for a service level that is measured by an SLI.

Objetivo de nivel de servicio

Ejemplos:

- Latencia media en una búsqueda < 100 ms
- Tasa de errores < 0.01%

Los objetivos sirven para **fijar expectativas**

SLA, ahora sí

Finally, SLAs are service level agreements: an explicit or implicit contract with your users that includes consequences of meeting (or missing) the SLOs they contain.

Acuerdo de nivel de servicio

Una **entidad de negocio**

Suelen tener **truco**

Ejercicio: Amazon SLA

Amazon Compute SLA

Gastamos €1000/mes en EC2

Hay un terremoto y la disponibilidad cae al 99.01%

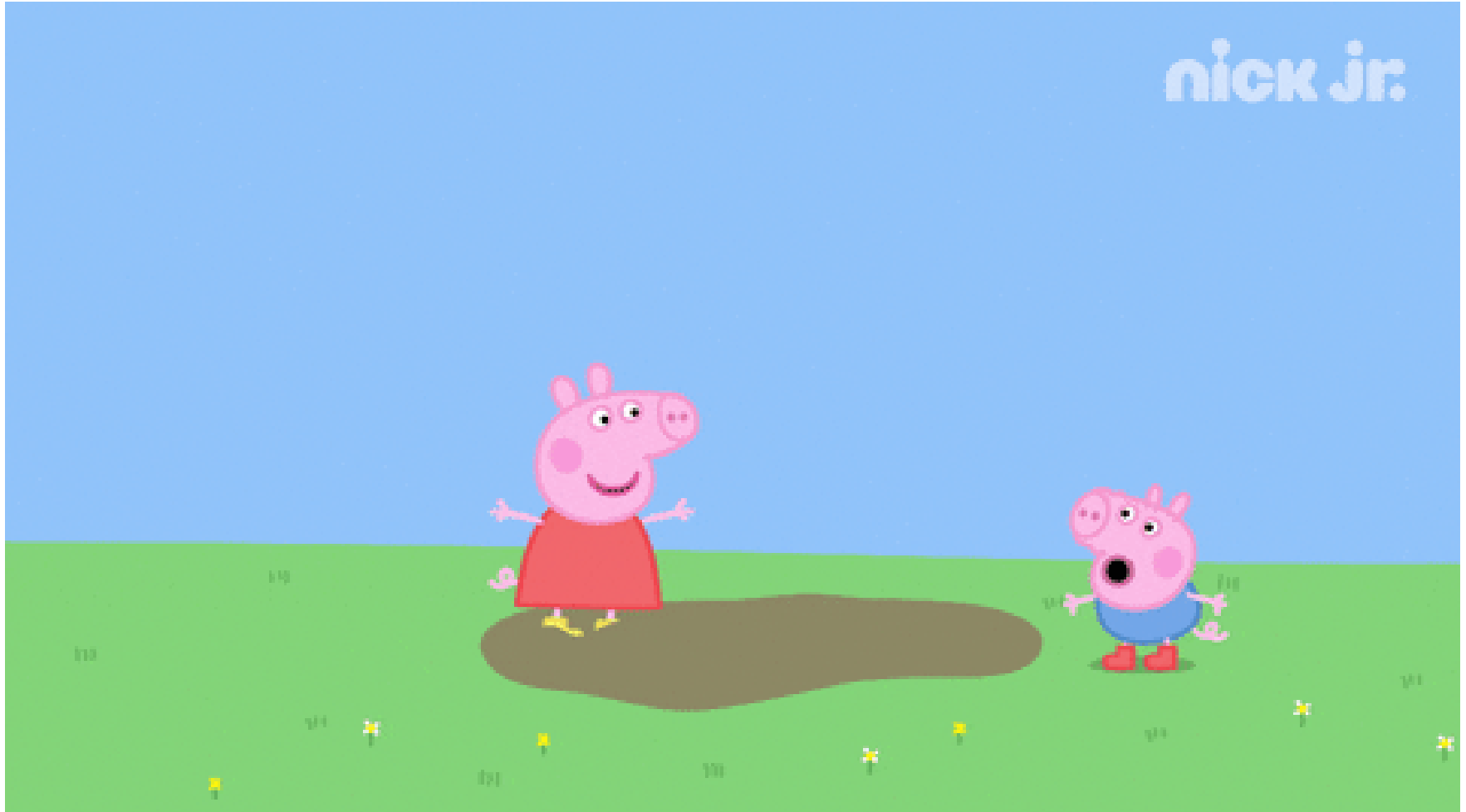
¿Cuánta pasta nos soltará Amazon?

¿Cuánto tiempo tiene que estar caído EC2 en un mes para obtener:

- un 10% de la factura?
- un 100% de la factura?



That was easy!



Degradación de servicio

Muchas veces el SLA sólo atiende a la disponibilidad

A veces el servicio responde pero **se degrada**

Latencia alta, errores esporádicos...

Un SLA puede diseñarse para combinar varios SLOs

- Tasa de error $< 0.1\%$
- Latencia $> 1000\text{ ms} \Rightarrow$ error

Objetivo de uptime



Presupuesto de incidencias

Introducido por Google: *error budget*

La mayoría de las incidencias son por **cambios**
~70%

¿Queremos cumplir el SLO? Limitamos los cambios

Demasiada disponibilidad: **mal**
¡Podíamos haber hecho más cambios!

Ejercicio: Completa el presupuesto

Nuestro objetivo de uptime es del 99.9%

Hemos calculado que el 66% del downtime es por despliegues

Llevamos en el año **siete caídas de una hora** aprox.

¿Cuándo podremos hacer un cambio?



Ejercicio + (en común)

Mirando en detalle las caídas de los últimos años:

- Percentil 50: 1 hora
- Percentil 90: 2 horas

¿Qué podemos hacer para empezar a trabajar antes?



Alley oop!



Incógnitas desconocidas

Lo que no sabemos	incógnitas conocidas	incógnitas desconocidas
	conocimientos conocidos	conocimientos desconocidos
	Conocido	Desconocido

Cuadrante de **Rumsfeld**

¿Qué no sabemos?

Hay **conocimientos conocidos:**

Lo que sabemos que sabemos

Hay **incógnitas conocidas:**

Lo que sabemos que no sabemos

Hay **conocimientos desconocidos:**

Lo que no sabemos que sabemos

Hay **incógnitas desconocidas:**

Lo que no sabemos que no sabemos

Ejemplos de incógnitas desconocidas

Lo que no sabemos

incógnitas
conocidas

"no sabemos qué
pasa a las 5k rps"

incógnitas
desconocidas

"cuando se agotan
los file descriptors
el sistema se cae"

Lo que sabemos

conocimientos
conocidos

"recibimos 1000 rps
de media"

conocimientos
desconocidos

"la carga se
desmadra los lunes"

Conocido

Desconocido

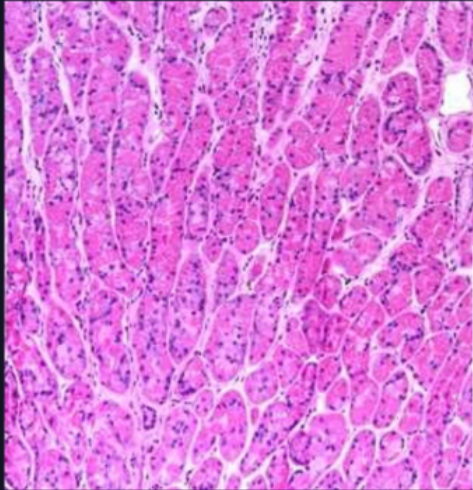
Testing in Production




Fuente

All the World's a Staging Server

Deployment



 LaunchDarkly

- Kill staging
- Test in production
- Launch darkly
- Go faster, be safer

@wiredferret

Heidi Waterhouse

I really hate metrics.

Charity Majors

Bibliografía

SRE Book: [Service Level Objectives](#)

SRE Book: [Embracing Risk](#)

pinchito.es: [Continuous Deployment on the Cheap](#)

Charity Majors: [I test in prod](#)

Charity Majors: [Yes, I Test in Production \(And So Do You\)](#)