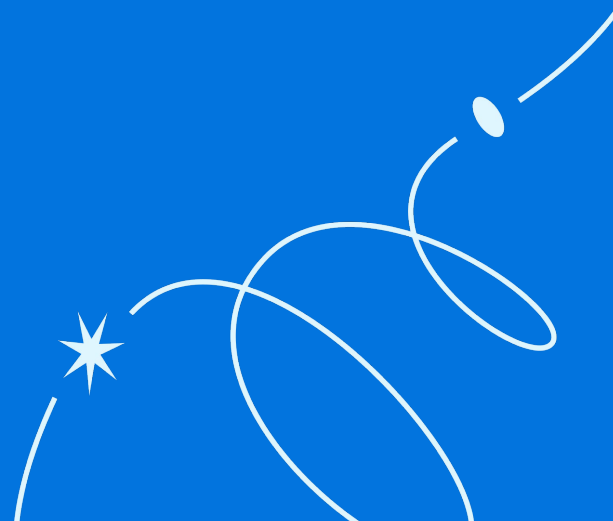
Remind

Escalado dinámico con recursos compartidos

Percona University
Montevideo, 24/10/23



Hoy veremos

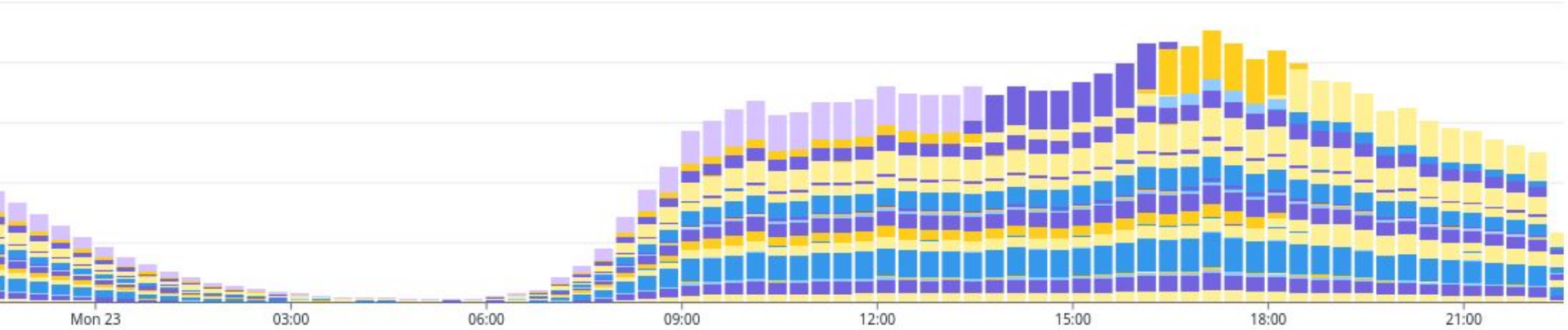
1. Introducción
2. Formulación del problema
3. La solución propuesta
4. Ejemplos de implementación

Autoescalado

- Capacidad de adaptar la capacidad a una demanda dinámica de forma automática
- Variables y sistemas de control
- Optimización de recursos

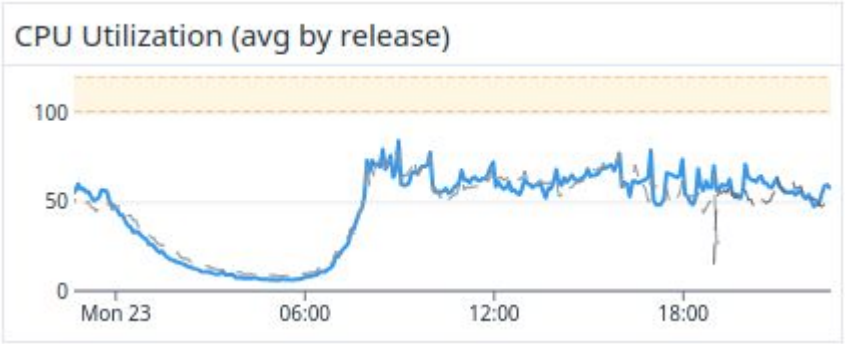
Autoescalado

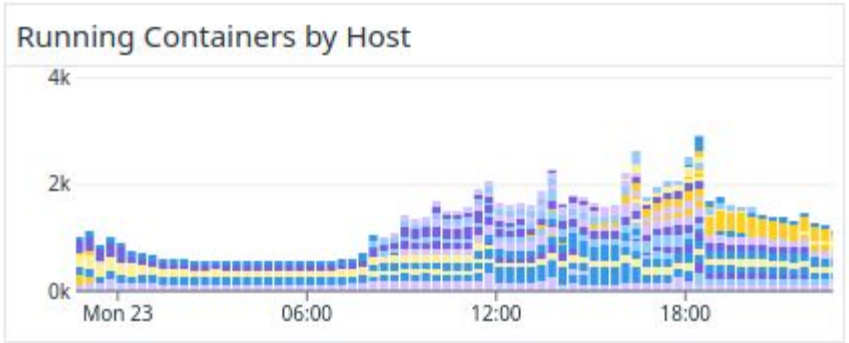
Peticiones en balanceadores de carga, diario



Autoescalado

- Apps 12factor
- Diversos sistemas
  - ECS (docker)
  - EC2 (instancias)
  - RDS (bases de datos)
- Variable de control
  - CPU
  - req/s



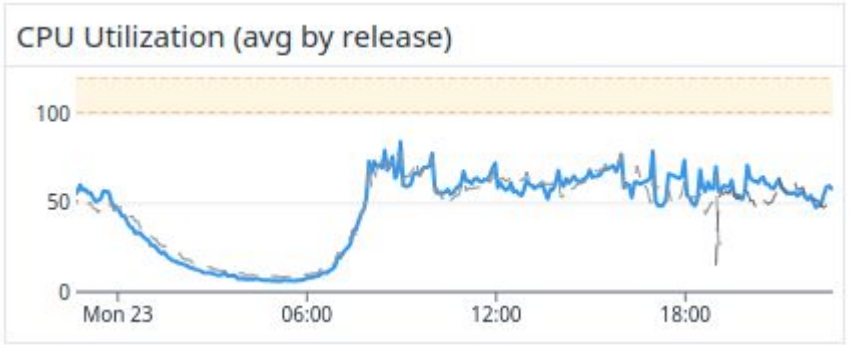


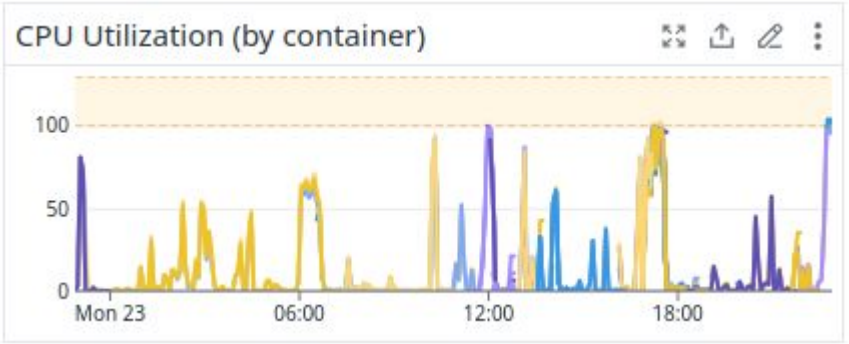
Procesos por lotes

- Naturaleza asíncrona (vs. tráfico de usuario)
- Programados o de activación manual
- Tráfico por rachas

Lotes vs. tráfico

- Tráfico irregular sobre los recursos compartidos
- ¿Variable de control?



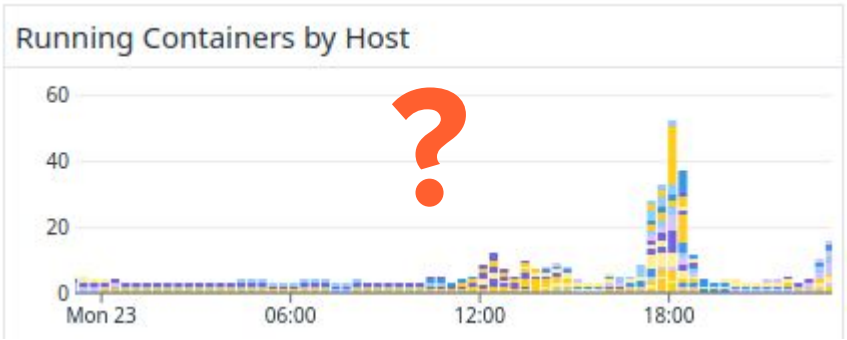


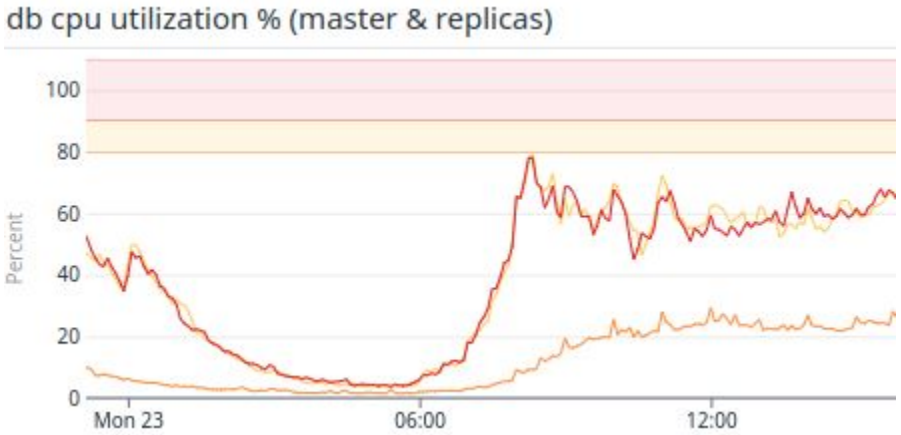
El problema

- Procesos por lotes que comparten recursos con tráfico síncrono
- Autoescalado óptimo
    - sin afectar negativamente el recurso

El problema bajado a tierra

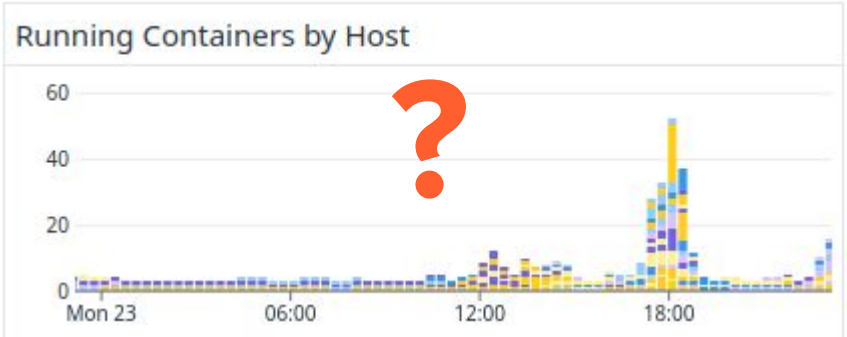
- DB tiene capacidad disponible limitada
- Proceso asíncrono con *p* copias, cada una aumentando el uso de la DB
- ¿Hasta dónde puedo escalar el proceso por lotes sin exceder la capacidad?

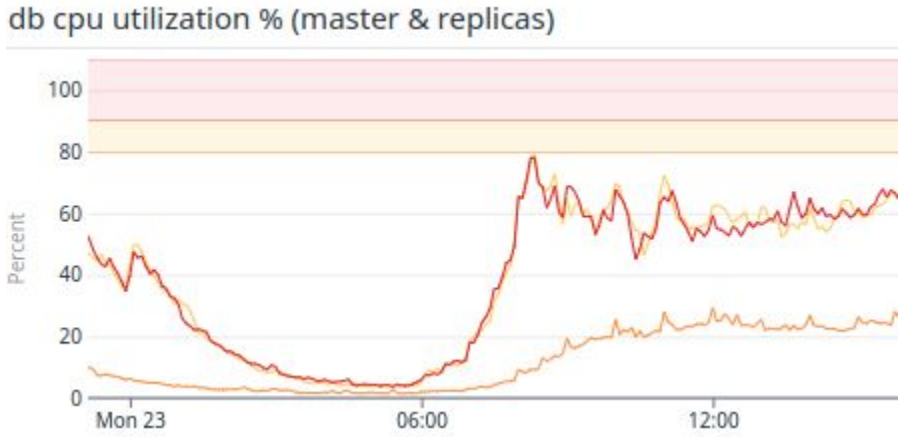




El problema bajado a tierra

- Variable de control lógica: CPU de la DB (*r* ≤ 70%)
- Ajustar la cantidad de copias *p* (*p = p(r)*)





Una solución lineal

Supongamos una medida

- *p* copias del proceso
- Utilización *R%* de la BD, de la cual *r%* corresponde al proceso (*R = r + s*)
  - Esto es posible en RDS y PG (*)
- Queremos alcanzar una utilización del 70%

k es utilización por copia : ***k = r / p***

Busco *p* para que R = 70% = s + k*p

Una solución lineal

$$p = (70\% - s) / k$$

En general, si tengo un *setpoint S*, y hago una medida de *p, r,* y *R*,

$$p \leftarrow [\ (S - R) / r + 1\ ] * p$$

- Si $S > R$, p será incrementado
- Si $S < R$, p decrementa
- Si $S = R$, p es estable

Una solución naïve

Dos situaciones que rompen el esquema

- El proceso no genera la suficiente carga: $R < S$
- El algoritmo colocará más y más copias, queriendo incrementar R

Una solución naïve

Dos situaciones que rompen el esquema

- El proceso deja de procesar datos
- Caso extremo de la situación anterior
    - Por maś copias que el algoritmo agregue, la carga no incrementa

Solución

Pérdida de linealidad

- Cuando se detecta una disminución de *k*, la carga por copia, se pasa a **modo no lineal**
- Cambio de variable de control
    - Se intenta mantener la carga

Pérdida de linealidad

- *Setpoint* en BD de 90%
- Con *p = 2* copias, tengo utilización del 50%, 20% propia y 30% externa
  - Cada copia me da 10%
- Hago *p = 6*, para obtener 90% = 60% + 30% externa
  - Modo lineal: espero 90%
- **Obtengo 70%**
  - Paso a modo no lineal. *Setpoint* no lineal: 70%.
- Sé que para 70% = 40% propio + 30% externo, necesito *p = 4*
  - Cambio a *p = 4* y controlo mi nuevo *setpoint* no lineal

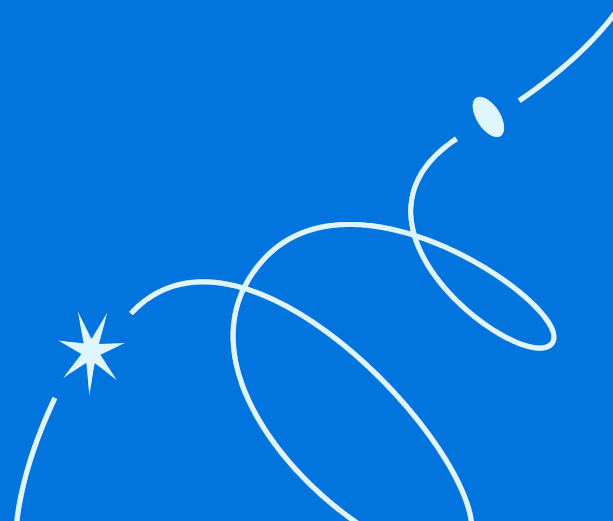
¿Cómo vuelvo a la zona lineal?

- Como controlo un nuevo *setpoint*, puedo no darme cuenta de que el sistema es nuevamente lineal
- Pruebas estocásticas cuando el sistema está estable
  - Pruebo al azar volver a modo lineal, y aumentar $p = p + 1$
  - Si logro un aumento, estoy en zona lineal

Implementaciones

- De pruebas
- Postgres
- RDS PG Aurora + ECS

Remind

Preguntas

ignacio@remind101.com