Formulario de estimaciones

Las nueve cuentas de la sección de estimaciones, cada una con su fórmula en variables, qué significa cada símbolo, el ejemplo de C01 paso a paso, una figura y un truco para recordarla. En la versión web, al final hay una calculadora para cambiar los supuestos.

Cómo recordarla · Trampa típica · Valores de C01 en verde

0 Números de bolsillo

Lo que conviene saberse de memoria antes de cualquier cuenta.

TIEMPO

$1 \text{ día} = 86{,}400 \text{ s} \approx 10^5$

$1 \text{ M/día} \approx 11.6 \text{ req/s}$

MES

costos: 730 h

SLO: 30 días $= 720$ h

predicciones al mes = diarias × 30.4

BITS Y BYTES

$\text{bits/s} = 8 \times \text{bytes/s}$

$1 \text{ Gb/s} = 125 \text{ MB/s}$

BYTES POR VALOR

FP32 **4** · FP16/BF16 **2** · INT8 **1** · INT4 **½**

imagen 224 × 224 × 3 = 150,528 valores

UTILIZACIÓN OBJETIVO

GPU online: $\rho \approx 0.5$–0.7

HPA de CPU: $\rho \approx 0.6$

batch offline: cerca de 100 %

COLAS (M/M/1) Y LITTLE

$W = \frac{S}{1-\rho} \cdot p99 \approx 4.6W$

$L = \lambda W$ (requests en vuelo)

INSTANCIAS (APROX., US-EAST-1)

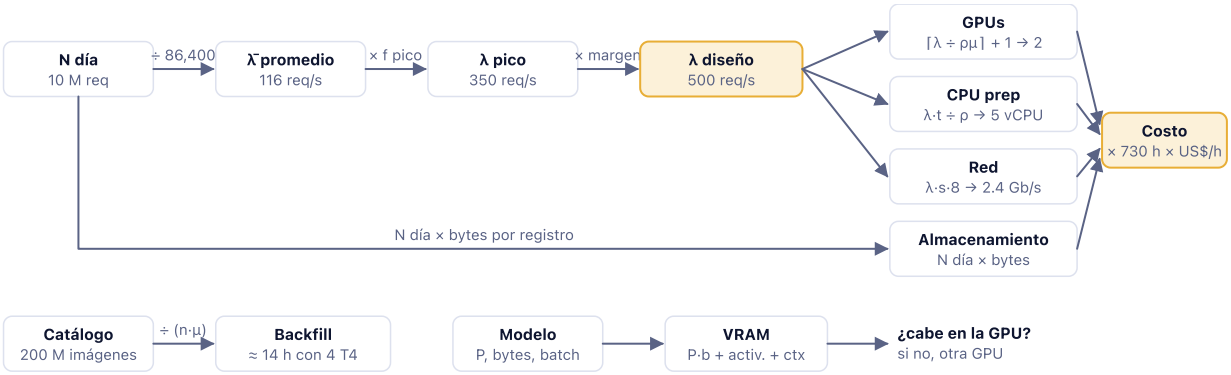
g4dn.xlarge (T4) **US$0.526/h** · g4dn.2xlarge **0.752** · c6i.xlarge (4 vCPU) **0.17** (≈ 0.0425 por vCPU-h)

POR USO (APROX.)

entre AZs **US$0.01/GB** por sentido · S3 GET **US$0.0004** por 1,000 · ALB ≈ **US$0.008** por GB procesado

La cadena

Cada número sale del anterior. Si aprendes el orden, las fórmulas se acomodan solas.



Del volumen diario al pico de diseño; de ahí, GPUs, CPU y red; todo termina en el costo. La VRAM solo pregunta si el modelo cabe, y el backfill es un cálculo aparte.

1 Tráfico

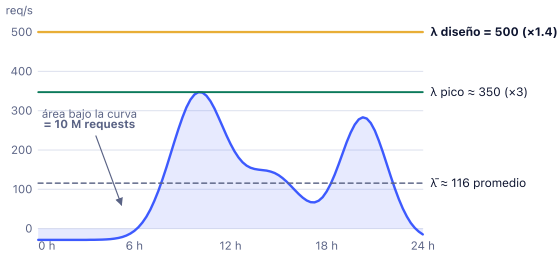
¿Cuántos requests por segundo tienes que aguantar?

$$\bar{\lambda} = \frac{N_{\text{día}}}{86{,}400\ \text{s}}$$

$$\lambda_{\text{pico}} = f_{\text{pico}} \cdot \bar{\lambda} \qquad f_{\text{pico}} \approx 24 \cdot s_{\text{hora pico}}$$

$$\lambda_{\text{diseño}} = m \cdot \lambda_{\text{pico}}$$

SÍMBOLO	QUÉ ES	UNIDAD	EN C01
$N_{\text{día}}$	requests (o imágenes) por día	req/día	**10 M**
$\bar{\lambda}$	tasa promedio	req/s	**≈ 116**
f_{pico}	factor de pico diurno (2–5×)	–	**3**
$s_{\text{hora pico}}$	fracción del día en la hora pico	–	**12.5 % → 3×**
m	margen por spikes y crecimiento (1.3–1.5)	–	**≈ 1.4**
$\lambda_{\text{diseño}}$	tasa para la que dimensionas	req/s	**500**



La curva de un día. El área bajo la curva son los 10 M requests; el promedio la reparte parejo; dimensionas para la línea de diseño.

CON LOS NÚMEROS DE C01

1. $\bar{\lambda} = \dfrac{10{,}000{,}000}{86{,}400\ \text{s}} \approx 116\ \text{req/s}$
2. $\lambda_{\text{pico}} = 3 \times 116 \approx 350\ \text{req/s}$
3. $\lambda_{\text{diseño}} = 1.4 \times 350 \approx 490 \rightarrow 500\ \text{req/s}$ (redondeas hacia arriba)
4. por GPU con 2 GPUs: $\dfrac{500}{2} = 250\ \text{req/s}$

CÓMO RECORDARLA

Divide por el día, multiplica por el pico, súmale margen. Atajo: 1 M/día ≈ 11.6 req/s. Con 10^5 s por día subestimas ≈ 14 %.

TRAMPA

Dimensionar con el promedio. El pico decide cuántas máquinas pones; el promedio decide cuánto pagas, porque la mayor parte del día el sistema trabaja muy por debajo de su capacidad.

2 Presupuesto de latencia

¿Cuántos milisegundos le tocan a cada tramo?

$$\mathrm{SLO}_{p99} \geq \sum_i b_i + \text{margen}$$

$$b_{\text{modelo}} = \mathrm{SLO} - \sum_{i \neq \text{modelo}} b_i - \text{margen}$$

$$p99(X+Y) \neq p99(X) + p99(Y)$$

SÍMBOLO	QUÉ ES	UNIDAD	EN C01
SLO_{p99}	objetivo de latencia server-side	ms	200
b_i	presupuesto p99 del tramo i	ms	ver la barra
margen	lo que no está en la tabla: fotos gigantes, GC, throttling	ms	≈ 48
t_{timeout}	tiempo máximo hacia el tramo siguiente	ms	150 (gateway → Triton)



- ALB + routing **10**
- Leer el body **30**
- Decode + resize + crop **50**
- Normalizar → tensor **2**
- Hop gateway → Triton **10**
- Cola de batching **2**
- Cómputo GPU **15**
- Postproceso + JSON **3**
- Overhead de Python **30**
- margen **≈ 48**

Las cotas altas de p99 de cada tramo, apiladas. Suman ≈ 152 ms y quedan ≈ 48 ms de margen contra el SLO.

CON LOS NÚMEROS DE C01

1. $\sum b_i \approx 10 + 30 + 50 + 2 + 10 + 2 + 15 + 3 + 30 \approx 152$ ms
2. $\text{margen} = 200 - 152 \approx 48$ ms
3. $t_{\text{timeout}}^{\text{gateway}\to\text{Triton}} = 150$ ms (después ya no cumples el SLO: falla rápido)
4. $b_{\text{cola de batching}} \leq \text{max_queue_delay} = 2$ ms

CÓMO RECORDARLA

Reparte el SLO antes de elegir tecnología. Primero la cuenta y después el stack: así sabes dónde puedes gastar (2 ms de cola de batching) y dónde no (decodificar un JPEG de 12 MP sin cuidado).

TRAMPA

Los percentiles no se suman. La suma de p99 suele sobrestimar el p99 total, pero no es una cota garantizada: úsala para repartir y mide el p99 de punta a punta. Y la subida de la foto no entra: el SLO es server-side.

3 GPUs

¿Cuántas GPUs pones y qué tan cargadas van?

$$\mu_{\text{plan}} = \rho_{\text{obj}} \cdot \mu_{\max}$$

$$n_{\text{thr}} = \left\lceil \frac{\lambda_{\text{diseño}}}{\mu_{\text{plan}}} \right\rceil \qquad n = \max\left(n_{\text{thr}} + 1, n_{\text{AZ}}\right)$$

$$u = \frac{\lambda}{n\mu_{\max}}$$

$$W = \frac{S}{1-\rho} \qquad p99 \approx 4.6W \qquad L = \lambda W$$

SÍMBOLO	QUÉ ES	UNIDAD	EN C01
$\mu_{\max}$	máximo medido por GPU (perf_analyzer)	img/s	≈ 1,200 (T4)
ρ_{obj}	utilización objetivo (50–70 %)	–	0.5
n_{thr}	GPUs por throughput	GPU	1
n_{AZ}	una por zona (disponibilidad)	GPU	2
u	carga real sobre el máximo	%	5–42 %
W, S	tiempo en el sistema y de servicio	s	–
L	requests en vuelo (Little)	–	–



Izquierda: cerca de la saturación la espera explota (M/M/1). Derecha: la carga real de cada T4 en C01; el número de GPUs lo fija la disponibilidad, no el throughput.

CON LOS NÚMEROS DE C01

1. $\mu_{\text{plan}} = 0.5 \times 1{,}200 = 600$ img/s
2. $n_{\text{thr}} = \left\lceil \frac{500}{600} \right\rceil = \lceil 0.83 \rceil = 1$
3. $n = \max(1+1,\ 2) = 2$ T4 (una por AZ)
4. $u_{\text{diseño}} = \frac{500}{2 \times 1{,}200} \approx 21\%$ $\quad u_{\text{falla}} = \frac{500}{1{,}200} \approx 42\%$ $\quad u_{\text{prom}} = \frac{116}{2 \times 1{,}200} \approx 5\%$
5. PyTorch eager FP32: $\frac{500}{\approx 100 \text{ img/s}} \approx 5$ GPUs

CÓMO RECORDARLA

Mide, baja a ρ, redondea hacia arriba, suma una. El orden importa: primero el máximo medido, luego la utilización objetivo, luego el techo, al final N+1. Y con Little, $L = \lambda W$: 500 req/s × 50 ms = 25 requests en vuelo.

TRAMPA

Planear al 90 %: con $\rho = 0.9$ el tiempo en el sistema es 10× el de servicio. Y `nv_gpu_utilization` no mide capacidad usada: marca tiempo con kernels corriendo.

4 ¿Y sin GPU?

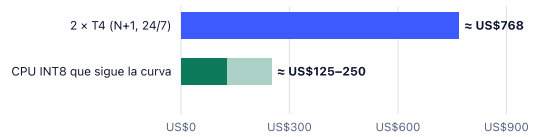
¿Cuántos vCPU harían el mismo trabajo y cuánto cuestan?

$$C_{ocup} = \lambda \cdot t_{CPU}$$

$$C_{prov} = \frac{\lambda t_{CPU}}{\rho_{HPA}}$$

$$\text{Costo}_{mes} = C_{facturado} \cdot p_{vCPU\text{-}h} \cdot 730\ \text{h}$$

SÍMBOLO	QUÉ ES	UNIDAD	EN C01
t_{CPU}	segundos de CPU por imagen por vCPU	s	FP32 0.04–0.08 · INT8 0.008–0.025
C_{ocup}	vCPU ocupados (Little con S = t)	vCPU	–
ρ_{HPA}	objetivo de CPU del autoscaler	–	0.6
$C_{facturado}$	promedio que pagas (sigue la curva + mínimos)	vCPU	≈ 4–8
$p_{vCPU\text{-}h}$	precio por vCPU-hora	US$	≈ 0.0425



Las GPUs con N+1 no bajan de 2 las 24 horas; la CPU puede seguir la curva diaria. Con INT8, CPU-only puede costar igual o menos.

CON LOS NÚMEROS DE C01

1. FP32: $500 \times (0.04\text{–}0.08)\ \text{s} = 20\text{–}40$ vCPU ocupados en el pico
2. INT8, pico: $\frac{500 \times (0.008\text{–}0.025)}{0.6} \approx 7\text{–}21$ vCPU
3. INT8, promedio: $\frac{116 \times (0.008\text{–}0.025)}{0.6} \approx 2\text{–}5$ vCPU $(+$ mínimos $\Rightarrow 4\text{–}8)$
4. $(4\text{–}8) \times 0.0425 \times 730 \approx$ US$125–250/mes vs $2 \times 0.526 \times 730 \approx$ US$768

CÓMO RECORDARLA

vCPU ocupados = tráfico × segundos de CPU por imagen. Es Little otra vez: cada vCPU es un "servidor" y t_{CPU} su tiempo de servicio.

TRAMPA

Comparar pico contra pico. La diferencia está en el promedio: la CPU escala con la curva y la GPU con N+1 se queda encendida. Y lo mides la primera semana: latencia por vCPU y pérdida de INT8 en el golden set.

5 Memoria de GPU (VRAM)

¿Cabe el modelo en la GPU, y cuánto sobra?

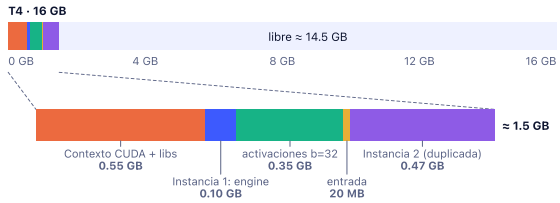
$$M_{\text{pesos}} = P \cdot b$$

$$M_{\text{entrada}} = B \cdot C \cdot H \cdot W \cdot b_{\text{in}}$$

$$M_{\text{total}} \approx M_{\text{CUDA}} + k\left(M_{\text{engine}} + M_{\text{act}}(B) + M_{\text{entrada}}(B)\right)$$

Para LLMs, lo que domina es el KV cache: $M_{\text{KV/token}} = 2 \cdot L_{\text{capas}} \cdot H_{\text{kv}} \cdot d_{\text{head}} \cdot b$ (Llama 3 8B en BF16: $2 \cdot 32 \cdot 8 \cdot 128 \cdot 2 = 128$ KiB por token).

SÍMBOLO	QUÉ ES	UNIDAD	EN C01
P	parámetros del modelo	–	**24.5 M (ResNet-50 de 500 clases)**
b	bytes por parámetro: FP32 4 · FP16 2 · INT8 1	B	**2 (FP16)**
B	tamaño del batch	–	**32**
$C \cdot H \cdot W$	canales × alto × ancho de la entrada	–	**3 × 224 × 224**
k	instancias del modelo en la GPU	–	**2**
M_{CUDA}	contexto CUDA + librerías	GB	**≈ 0.3–0.8**



Arriba, la T4 completa; abajo, el zoom de lo que sí se usa. La VRAM no es la restricción en C01.

CON LOS NÚMEROS DE C01

1. $M_{\text{pesos}} = 24.5\text{ M} \times 2\text{ B} \approx 49\text{ MB}$
2. $M_{\text{entrada}} = 32 \times 3 \times 224 \times 224 \times 4\text{ B} \approx 19\text{ MB}$
3. $M_{\text{act}}(32) \approx 0.2\text{–}0.5\text{ GB}$ $\quad M_{\text{CUDA}} \approx 0.3\text{–}0.8\text{ GB}$
4. $M_{\text{total}} \approx 1\text{–}3\text{ GB} \ll 16\text{ GB} \Rightarrow$ la VRAM no limita

CÓMO RECORDARLA

Parámetros × bytes es el piso, no el total. Súmale el contexto CUDA, las activaciones del batch y una copia por instancia.

TRAMPA

Creer que solo pesan los pesos. En visión casi siempre sobra memoria; en LLMs el KV cache crece con los tokens en vuelo y es lo que decide cuántas conversaciones caben.

6 CPU para preprocesamiento

¿Cuántos vCPU necesita el gateway que decodifica y prepara la imagen?

$$\bar{t}_{\text{prep}} = \sum_k w_k t_k$$

$$C_{\text{ocup}} = \lambda_{\text{diseño}} \cdot \bar{t}_{\text{prep}} \qquad C_{\text{req}} = \frac{C_{\text{ocup}}}{\rho_{\text{HPA}}}$$

$$n_{\text{nodos}} = \max\left(\left\lceil \frac{C_{\text{req}}}{\text{vCPU por nodo}} \right\rceil, n_{\text{AZ}}\right)$$

SÍMBOLO	QUÉ ES	UNIDAD	EN C01
w_k	fracción del tráfico de cada tipo de imagen	–	**85 % móvil · 15 % web**
t_k	CPU por imagen de ese tipo	ms	**≈ 5 · ≈ 60 (originales)**
$\bar{t}_{\text{prep}}$	CPU promedio por imagen (con draft mode)	ms	**≈ 6**
ρ_{HPA}	objetivo de CPU del HPA	–	**0.6**
vCPU por nodo	tamaño del nodo	vCPU	**4 (c6i.xlarge)**



Arriba, lo que hace el gateway con cada foto. Abajo, por qué el promedio engaña: una minoría de fotos grandes se come la mayor parte de la CPU.

CON LOS NÚMEROS DE C01

1. $C_{\text{ocup}} = 500 \times 6 \text{ ms} = 3 \text{ vCPU ocupados}$
2. $C_{\text{req}} = \frac{3}{0.6} = 5 \text{ vCPU}$
3. $n_{\text{nodos}} = \max(\lceil 5/4 \rceil,\ 2) = 2 \times \text{c6i.xlarge} = 8 \text{ vCPU}$
4. mezcla de originales: $0.85 \times 5 + 0.15 \times 60 = 13.25 \text{ ms} \Rightarrow 68\%$ lo pone el 15%

CÓMO RECORDARLA

La CPU se cuenta igual que la GPU: tráfico × tiempo ÷ utilización, y al final redondeas a nodos enteros con uno por zona.

TRAMPA

Quedarte con el promedio: la cola de fotos gigantes manda (por eso el draft mode no es micro-optimización). Y la VM de la GPU solo tiene 4 vCPU: el preprocesamiento va en su propio tier.

7 Red entre tiers

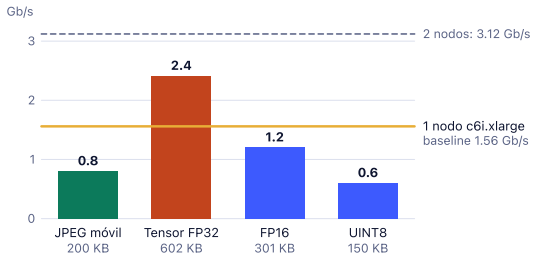
¿Cuántos Gb/s viajan del gateway al model server, y cuánto cuestan?

$$s = C \cdot H \cdot W \cdot b$$

$$BW = \lambda \cdot s \cdot 8 \ \frac{\text{bits}}{\text{byte}} \qquad u_{\text{NIC}} = \frac{BW / n_{\text{nodos}}}{BW_{\text{baseline}}}$$

$$V_{\text{mes}} = s \cdot N_{\text{día}} \cdot 30 \qquad \text{Costo}_{\text{AZ}} = f_{\text{cruce}} \cdot V_{\text{mes}} \cdot p_{\text{GB}}$$

SÍMBOLO	QUÉ ES	UNIDAD	EN C01
s	bytes por request que cruzan la red	B	**602,112 (FP32)**
BW	ancho de banda	Gb/s	**≈ 2.4**
BW_{baseline}	red sostenida de la instancia (no el “hasta”)	Gb/s	**1.56 (c6i.xlarge)**
f_{cruce}	fracción que cruza de zona	–	**0.5 si balanceas al azar**
p_{GB}	precio entre AZs (ida y vuelta)	US$/GB	**0.02**



Lo que viaja a 500 req/s según el formato. El tensor FP32 pesa 3× el JPEG móvil y deja cada nodo del gateway al 77 % de su baseline.

CON LOS NÚMEROS DE C01

1. $s = 3 \times 224 \times 224 \times 4 \text{ B} = 602{,}112 \text{ B} \approx 602 \text{ KB}$
2. $BW = 500 \times 602 \text{ KB} \times 8 \approx 2.4 \text{ Gb/s}$
3. $u_{\text{NIC}} = \dfrac{2.4/2}{1.56} \approx 77\%$ si cae una zona: $\dfrac{2.4}{1.56} \approx 154\%$
4. $V_{\text{mes}} \approx 602 \text{ KB} \times 10 \text{ M} \times 30 \approx 180 \text{ TB} \Rightarrow 0.5 \times 180{,}000 \text{ GB} \times 0.02 \approx \text{US\$1,800/mes}$

CÓMO RECORDARLA

Bytes × requests × 8 = bits por segundo. Y la tabla de bytes por valor: FP32 4, FP16 2, UINT8 1 (÷ 2 y ÷ 4 del tensor).

TRAMPA

Olvidar el × 8, usar el ancho de banda “hasta” en vez del baseline sostenido, y balancear al azar entre zonas: la transferencia entre AZs costaría más que las dos GPUs. Ruteo por zona desde el día 1.

8 Almacenamiento y costo mensual

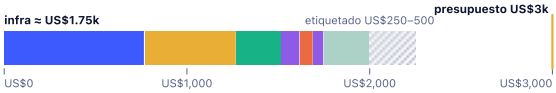
¿Cuánto guardas, cuánto pagas al mes y cuánto cuesta cada predicción?

$$S_{\text{día}} = N_{\text{día}} \cdot s_{\text{registro}} \qquad S_{\text{mes}} \approx \frac{30 \cdot S_{\text{día}}}{r_{\text{compresión}}}$$

$$\text{Costo}_{\text{mes}} = \sum_j n_j p_j \cdot 730\ \text{h} + \sum_k q_k c_k$$

$$c_{1k} = \frac{\text{Costo}_{\text{mes}}}{N_{\text{día}} \cdot 30.4} \cdot 1{,}000$$

SÍMBOLO	QUÉ ES	UNIDAD	EN C01
s_{registro}	bytes por log de predicción	B	≈ 300
$r_{\text{compresión}}$	Parquet comprimido (5–10×)	–	5–10
$n_j,\ p_j$	instancias de cada tipo y su precio por hora	US$/h	T4 0.526 · c6i 0.17
$q_k,\ c_k$	lo que se cobra por uso (GB, requests) y su precio	–	ALB 0.008/GB
c_{1k}	costo por 1,000 predicciones	US$	≈ 0.006



- GPUs (2 × g4dn.xlarge) **≈ US$768** 44 %
- ALB (hora + GB) **≈ US$503** 29 %
- Gateway (2 × c6i.xlarge) **≈ US$248** 14 %
- Logs, métricas, S3 **≈ US$100** 6 %
- EKS **≈ US$73** 4 %
- EBS, NAT, transferencia **≈ US$58** 3 %

La infraestructura de C01 por línea, contra el presupuesto de US$3k. El ALB es la segunda línea de costo porque cobra por cada GB de imagen que lo cruza.

CON LOS NÚMEROS DE C01

1. $S_{\text{día}} = 10\ \text{M} \times 300\ \text{B} = 3\ \text{GB/día} \approx 90\ \text{GB/mes crudo}\ (\div 5\text{–}10\ \text{en Parquet})$
2. $\text{GPUs} = 2 \times 0.526 \times 730 \approx 768 \qquad \text{gateway} = 2 \times 0.17 \times 730 \approx 248 \qquad \text{EKS} = 0.10 \times 730 \approx 73$
3. $\text{ALB} = 0.0225 \times 730 + 2{,}000\ \frac{\text{GB}}{\text{día}} \times 30.4 \times 0.008 \approx \text{US\$500}$
4. $\text{infra} \approx \text{US\$1.75k/mes} \qquad c_{1k} = \frac{1{,}750}{304\ \text{M}} \times 1{,}000 \approx \text{US\$0.006}$

CÓMO RECORDARLA

Mes de costos = 730 h. Precio por hora × horas × instancias, y después lo que se cobra por uso (GB procesados, requests a S3, transferencia). Para SLOs el mes es de 30 días = 720 h.

TRAMPA

Olvidar lo que se cobra por uso: aquí el ALB pesa ≈ 30 % de la infra. El almacenamiento casi nunca es lo caro; lo caro es la privacidad de lo que guardas.

9 Backfill de 200 M imágenes

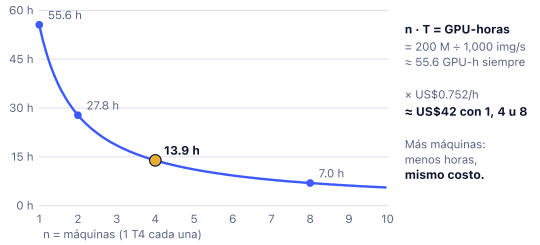
¿Cuánto tarda y cuánto cuesta procesar el catálogo existente?

$$T = \frac{N_{\text{catálogo}}}{n \cdot \mu_{\text{offline}}}$$

$$\text{GPU-h} = n \cdot T = \frac{N_{\text{catálogo}}}{\mu_{\text{offline}}} \qquad \text{Costo} = \text{GPU-h} \cdot p_{\text{GPU-h}}$$

$$C_{\text{decode}} = \mu_{\text{offline}} \cdot t_{\text{decode}} \qquad \text{GET} = N_{\text{catálogo}} \cdot p_{\text{GET}}$$

SÍMBOLO	QUÉ ES	UNIDAD	EN C01
$N_{\text{catálogo}}$	imágenes por procesar	—	**200 M**
μ_{offline}	throughput por GPU en batch (IO + decode)	img/s	**≈ 1,000**
n	máquinas en paralelo	—	**4**
t_{decode}	CPU por thumbnail	s	**≈ 0.002**
$p_{\text{GPU-h}}$	precio de la instancia	US$/h	**0.752 (g4dn.2xlarge)**
p_{GET}	precio por GET de S3	US$	**0.0004 / 1,000**



El tiempo cae como 1/n, pero las GPU-horas no cambian: paralelizar compra tiempo, no ahorra dinero.

CON LOS NÚMEROS DE C01

1. $T = \dfrac{200\text{ M}}{4 \times 1{,}000\text{ img/s}} = 50{,}000\text{ s} \approx 14\text{ h}$
2. $\text{GPU-h} = 4 \times 13.9 \approx 56 \Rightarrow 56 \times 0.752 \approx \text{US\$42 on-demand (menos en spot)}$
3. $C_{\text{decode}} = 1{,}000 \times 2\text{ ms} = 2\text{ cores} \Rightarrow \text{g4dn.2xlarge (8 vCPU)}$
4. $\text{GET} = 200\text{ M} \times \dfrac{0.0004}{1{,}000} = \text{US\$80}$ originales: $200\text{ M} \times 13.25\text{ ms} \approx 736\text{ CPU-h}$

CÓMO RECORDARLA

Más máquinas = menos horas, mismo costo. $n \cdot T$ es constante: el número de máquinas lo decide tu fecha límite, no tu presupuesto.

TRAMPA

Correrlo contra el Triton online. Es un job aparte, reanudable por shards. El cuello de botella es IO + decode, no la GPU, y los thumbnails se recomprimieron: valida en una muestra que las predicciones coinciden con las de los originales.

Números y precios del curso (C01, M1, M3, M4, M13), aproximados para us-east-1: verifica precios en la calculadora de AWS. Las cotas de hardware se **miden**; en la entrevista, dilo así.

© 2026 Erik Palacios · Contenido bajo CC BY-NC-ND 4.0 · ML desde cero · Reportar un error