Reducción de costos de infraestructura en la nube

Alexander Baquiax
Universidad Galileo de Guatemala
Departamento de Investigación de Operaciones
{alexbqb}@galileo.edu

Abstract

Hoy en día, es un hecho que un gran porcentaje de sistemas informáticos han migrado de despliegues *on prem* a despliegues en la nube. El impacto de este cambio es evidente en la rapidez de construcción de nuevos sistemas, pero, también trae una gran responsabilidad: el control de costos.

A continuación encontraremos un caso real, de cómo usando métodos de programación lineal, podríamos optimizar recursos computacionales de una forma determinista que pueda servir de ayuda para optimizar el uso de recursos y por consiguiente reducir el costo de mantener estos sistemas corriendo.

1 Introducción

Olay es una compañía enfocada en comercio conversacional, que diariamente procesa más de *500M* de mensajes. Para ser capaces de manejar todo este tráfico, la compañía creó varias docenas de servicios que funcionan en conjunto para proveer los servicios que Olay ofrece al mundo.

Cada servicio debe asegurar alta disponibilidad por lo cuál cada uno posee varias replicas. Esto da como resultado que Olay tenga que manejar cientos de aplicaciones y alojarlos sabiamente en la infraestructura disponible.

Actualmente, esta *elección sabia*, es manual. Un operador decide qué infraestructura es suficiente para alojar todas las aplicaciones, en un inicio era factible pero debido al crecimiento de la compañía esto se ha vuelto más complejo de tratar.

Es por ello que decidí proponer una posible forma de calcular esto de forma determinista, buscando suplir la infraestructura necesaria para alojar todas las aplicaciones y al mismo tiempo reducir el costo que esto conlleva.

2 Contexto

El equipo de ingeniería de *Olay* sigue el patrón bien conocido de *contenedores* y usa *Kubernetes*(K8[1]) como un orquestador de todos estos componentes. K8 es un sistema construido por Google que se encarga de alojar correctamente estos contenedores en máquinas virtuales. Lo que *K8* no puede hacer con totalidad es decidir cuántas máquinas virtuales están disponibles. Y es justo esta tarea la que un operador realiza, indicándole al orquestador las máquinas disponibles y los recursos que esta puede usar.

2.1 Contenedores

Para darle más contexto al lector, describiremos superficialmente que es un contenedor en el contexto de software.

Un contenedor es análogamente como un paquete aislado y autosuficiente. Este paquete contiene dependencias y configuraciones aisladas que hacen que una aplicación específica pueda funcionar dentro de ella.

La tecnología detrás de los contenedores, hace que sea posible simular completo aislamiento de las aplicaciones haciéndolas parecer que están corriendo en máquinas completamente diferentes, a pesar de que en realidad estén corriendo en una misma máquina.

La belleza de esto radica en que, cada aplicación puede estar corriendo en sistemas operativos diferentes, con dependencias diferentes, sin afectar al resto de contendedores corriendo en la misma máquina.

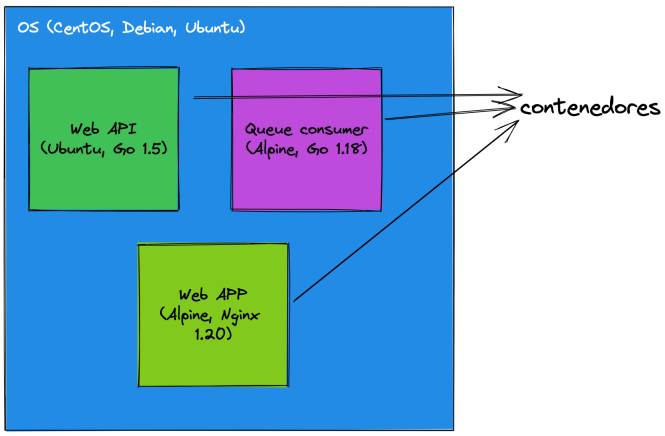

Figure 1: Containers

2.2 K8

Kubernetes nació luego de la aparición de los contenedores, con el fin de proveer a los operadores una herramienta declarativa para automatizar la organización de los contenedores

dentro de no una, sino muchas máquinas disponibles para correr aplicaciones.

Esta automatización representa un ahorro sustancial de trabajo al operador, confiando en la eficiencia de K8 para alojar correctamente aplicaciones en las máquinas con suficiente espacio.

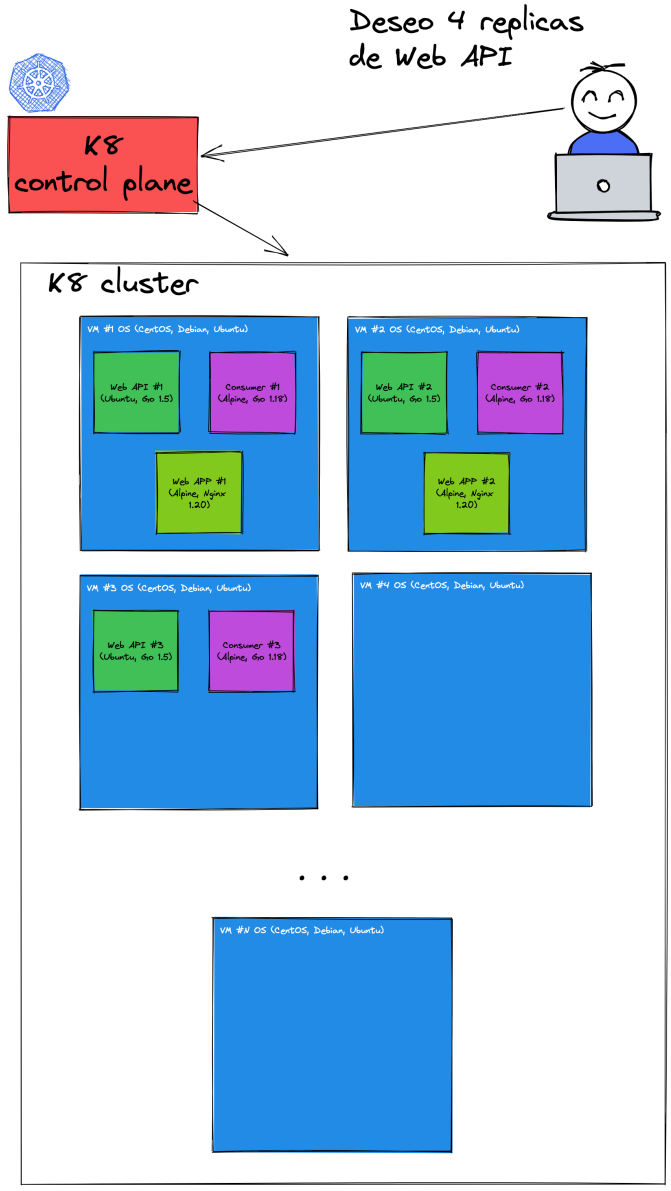

Figure 2: Kubernetes

Máquinas virtuales

K8 no fue diseñado para funcionar en Cloud providers como GCP, AWS o Azur, este también puede funcionar en una versión on prem. Pero para este contexto seguiremos en *el mundo de las nubes*. Es por eso que hablamos de máquinas virtuales (VMs).

Las VMs son ofreciedas por los Cloud providers como de uso general. Es decir uno puede hacer casi cualquier cosa dentro de ellas. K8, necesita tener acceso a todas las VMs disponibles para poder malabarear y alojar dentro de ellas las aplicaciones según sus requisitos.

2.3 Aplicaciones

Las aplicaciones pueden ser casi cualquier pieza de software, como por ejemplo un servidor web que despache web apps, o un HTTP API que se encargue de recibir solicitudes de clientes desde un web app o un mobile app, o un *daemon* app que funcione de forma indefinida tras bambalinas.

Cada aplicación hará uso, casi siempre de los siguientes recursos: disco, memoria y cpu.

Disco

Algunas aplicaciones, en su diseño, son instruidas a escribir archivos a los sistemas operativos. Es por eso de los sistemas operativos usualmente necesitan tener un nivel de persistencia que puede ser satisfecho por HDD o SSDs. El costo de espacio en disco, hoy en día no es tan alto como hace algunos años.

Memoria

La memoria es un recurso muy preciado para las aplicaciones, acá se alojan datos efímeros que son usados en cada transacción dentro de las aplicaciones. Es por eso que su uso es linealmente proporcional al uso de las aplicaciones. El precio de este recurso es bastante superior al del precio de los discos.

CPU

Los procesadores han sido siempre el componente más costoso cuando hablamos de computadoras. De este depende la velocidad con la que las operaciones dentro de las aplicaciones pueden ser ejecutadas. Ya sea con un alto nivel de concurrencia y paralelismo o una alta frecuencia de ejecución.

El costo de este recurso es inclusive, en algunos casos, mucho mayor que el de la memoria.

2.4 VMs, aplicaciones y costos

Recapitulando un poco, hablamos de contenedores, vms y aplicaciones.

Los contenedores son entornos aislados que corren aplicaciones, aislados de más contenedores y que se alojan dentro de VMs. Teniendo las VMs recursos finitos como disco, memoria y cpu.

La forma en la que abastecemos de disco, memoria y cpu es rentando VMs mensualmente. Eso quiere decir que el proceso de aprovisionamiento no se hace de forma aislada, es decir no puedo alquilar tantos GB de RAM y tantos CPUs de forma individual. La forma es rentando tantas VMs sean necesarias que en conjunto me provean tantos CPUs y GB de RAM que yo necesito.

Y justo en este punto, es donde el humano pierde control, y empieza a desperdiciar recursos.

Por ejemplo si yo le dijera a una ingeniera de infraestructura 100GB de RAM y 34 CPUs disponibles. Ella tendría que elegir la combinación más adecuada de VMs que me provean esos requisitos. Podrían ser 10 VMs que posean al menso 10GB de RAM y al menos 3.4 CPUs. La desventaja acá es

que no existen ese tipo de VMs en el mercado, uno puede elegir entre tipos definidos de VMs, como aquellas que tiene 10 VMs pero solo 2 CPUs.

3 Planteamiento del problema

Ya con el contexto en lugar, podrá inferir que lo que tratamos de optimizar es el costo de alquilar VMs pero al mismo tiempo suplir los requisitos de cada aplicación en todo el ecosistema de Olay.

3.1 Recopilación de datos

Olay posee varios cientos de contenedores corriendo en la nube, estos son cerca de *1000*, en conjunto se estima que la cantidad total de memoria requerida es de **2122 GB** y la cantidad de CPUs necesarios es **520**.

Costos

Las máquinas disponibles a considerar son las siguientes:

Nombre	Memory (GB)	CPU	Costo mes ($)
e2-highmem-2	16	2	65.99
e2-standard-4	8	4	97.84
e2-highcpu-8	8	8	144.45

Índices

Tipos de máquinas disponibles

$$j = 1, 2, 3 \tag{1}$$

Tipos de recursos disponibles, en este caso sólo consideramos memoria y CPU

$$i = 1, 2 \tag{2}$$

Parámetros

Parámetro	Representación
b	Cantidad de recurso necesario
c_j	costo por máquina j
a_{ij}	cantidad de recurso i en maquina j

Variables

x_j cantidad de maquinas de tipo j a ser utilizadas

Función objetivo

$$\min c_1 * x_1 + c_2 * x_2 + c_3 * x_3 \tag{3}$$

Restricciones

$$\begin{aligned} a_{i1}x1 + a_{i2}x2 + ... + a_{in} * xn >= b \\ x_j >= 0 \end{aligned} \tag{4}$$

3.2 Resultados

Dado los datos descritos anteriormente, podemos proceder a definir el siguiente problema de minimización

$$\min 65.99 * x_1 + 97.84 * x_2 + 144.45 * x_3$$

Sujeto a:

$$\begin{aligned} 16 * x_1 + 8 * x_2 + 8 * x_3 >= 2122 \\ 2 * x_1 + 4 * x_2 + 5 * x_3 >= 520 \\ x_1, x_2, x_3 >= 0 \end{aligned}$$

Lo cuál nos da un resultado de: $z_* = 14258.345$, con un valor de $x_1 = 90.17$, $x_2 = 84.92$ y $x_3 = 0$

4 Conclusiones

Con este simple proceso de recolección y análisis, para los operadores de Olay será ahora más determinista elegir qué cantidad de qué tipo de máquinas habilitar para que Kubernetes pueda alojar contenedores de una forma eficiente.

Este es un ejemplo del beneficio de la programación lineal en ámbitos que tal vez parezca inusual usarlo.

References

[1] Google. *Kubernetes*. URL: https://kubernetes.io/.