Reducción de costos de infraestructura en la nube (PE)

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

Abstract

En un anterior documento de investigación disponible acá, describí un modelo básico para encontrar la combinación correcta de máquinas virtuales que en conjunto suplan una cantidad demandada de recursos.

A continuación podremos encontrar un modelo más robusto y genérico que contempla un listado más amplio de restricciones y variables. También se introduce el uso de la **programación entera** que nos permitirá dar un resultado más preciso de los costos involucrados. Esto último es una corrección a la investigación anterior, ya que en ese punto erróneamente se hizo una aproximación de los resultados obtenidos, sabiendo ahora que esa es una mala práctica ya que probablemente los valores hayan dejado de ser factibles.

1 Introducción

Hoy en día la administración de recursos en la nube es un tema crucial para el éxito de una empresa. Como miembro de una *start-up* creciente, he visto qué tan importante se vuelve ser consciente de de la forma en la que usas tus recursos. En este caso específico, recursos económicos.

Hace unas semanas me vi envuelto en un proyecto de reducción de costos de la compañía para la que laboro y en medio de este, observé un patrón común: **el sobre aprovisionamientos de recursos computacionales**.

Mi idea para poder evitar esto en el futuro fue analizar qué era lo que necesitábamos en realidad y cómo asegurar que estuviésemos usando sólo lo necesario. Y fue acá dónde pensé utilizar la Programación Lineal para poder tener una forma sistemática de evaluar y analizar estos requisitos.

2 Contexto

Mi enfoque está basado completamente en el uso de recursos computacionales en proveedores de servicios tales como: Google Cloud Platform, AWS o Azure.

El problema general se basa en que como usuarios de servicios de la nube, necesitamos *máquinas* que en conjunto nos provean ciertas capacidades computacionales. Este problema no sólo abarca arquitecturas basadas en contenedores, sino cualquier versión que tenga como componente base **máquinas virtuales**. Y el reto es alquilar la menor cantidad de máquinas que suplan completamente nuestras necesidades.

Máquinas virtuales
Las máquinas virtuales no son más que máquinas físicas alojadas por alguien más en alguna parte del mundo, que se nos alquilan con acceso a ellas a través de internet.

Estas máquinas tiene características que hallaríamos en máquinas físicas tales como: memoria, capacidad de disco, capacidad de transferencia de red, capacidad de operaciones I/O, memoria gráfica y más.

3 El problema

El problema tentativo que se puede generar cuando hacemos uso de máquinas virtuales, como mencioné anteriormente, es el sobre aprovisionamiento. En otra palabras abarcar más de lo que realmente necesitamos.

Una causa de este malgasto de dinero es el no tener una forma determinista para calcular cuántas unidades necesitamos. Muchas veces se delega a la experiencia de los operadores humanos, y esto deja de tener precisión.

Tanto en el mundo de contenedores como el de uso general de máquinas virtuales, este tema se vuelve importante a medida que se escala.

El objetivo entonces será **minimizar la cantidad de máquinas a usar**, y dado que la unidad mínima de alquiler es una máquina, necesitamos contemplar números enteros solamente.

3.1 Conjuntos conocidos

Para poder resolver este problema utilizando Programación Lineal, necesitamos definir primero los conjuntos de recursos conocidos.

Tipos de máquinas virtuales
El conjunto de tipos de máquinas virtuales disponibles en el proveedor de servicios. Acá es importante mencionar que existen diversos tipos de máquinas en cada proveedor. La diferencia entre cada máquina radica en la cantidad de recursos que esta posee. Algunos poseen más memoria que otras, y otras poseen más CPUs, y justo acá entra la complejidad de elegir el subset correcto de máquinas.

$$V = v_1, v_2, ..., v_n \tag{1}$$

Tipos de recursos
Luego tenemos el conjunto de recursos que buscamos satisfacer.

$$R = r_1, r_2, ..., r_6 \tag{2}$$

En este caso puntual consideraremos el siguiente listado de recursos:

	Recurso	Rango	Dimensiones
1	Memoria	$\in \mathbb{R}^+$	Gigabyte
2	Disco	$\in \mathbb{R}^+$	Gigabyte
3	GPUs	$\in \mathbb{N}$	Unidades
4	CPUs	$\in \mathbb{N}$	Unidades
5	Network	$\in \mathbb{R}^+$	Gb/s
6	IOPS	$\in \mathbb{N}$	IOPS

3.2 Índices
La forma de identificar cada subelemento dentro de los conjuntos será:

$$v \in V \tag{3}$$

Por cada tipo de máquina virtual conocido.

$$r \in R \tag{4}$$

Por cada tipo de recurso conocido.

3.3 Parámetros
Existe un listado de valores conocidos que usaremos como base de nuestro problema.

Costo
El primer valor conocido es el costo por hora del alquiler de las máquinas. Cada tipo de máquina $v \in V$ posee un costo diferente, a este costo lo conoceremos como:

$$c_v \tag{5}$$

Recursos por tipo de máquina
Otro dato importante es conocer la cantidad de recurso r que la máquina v posea. Esto es un valor conocido, especificado por el proveedor, lo denotaremos como:

$$q_{rv} \tag{6}$$

Cantidad de recursos demandada
Al final del día, un administrador sistemas conocerá la cantidad de recursos por tipo r requerido en su totalidad. Este valor será conocido como:

$$d_r \tag{7}$$

3.4 Variables
Los valores que estamos buscando son las cantidades x_v de máquinas de tipo v que en conjunto suplan los recursos demandados.:

$$x_v \in \mathbb{N} \tag{8}$$

3.5 Función objetivo
El objetivo es determinar la **menor** cantidad de dinero a gastar en máquinas, que en conjunto suplan la cantidad de recursos demandados.

$$\min costo : \sum_{v \in V} c_v * x_v \tag{9}$$

3.6 Restricciones
La restricciones más evidentes son la demanda de recursos, esto podemos expresarlo de la siguiente forma:

$$\sum_{v \in V} x_v * q_{rv} >= d_r; \forall r \in R \tag{10}$$

Y las siempre presentes, restricciones de NO negatividad.

$$\sum_{i=1}^{n} x_i >= 0; x \in \mathbb{N} \tag{11}$$

En este caso nuestras R restricciones son del tipo $>=$ puesto que necesitamos satisfacer un mínimo.

4 La solución
Con todo lo anterior bien definido, podemos proceder a resolver el problema utilizando técnicas de Programación Lineal. Algo que es muy importante denotar es que esta solución se hará utilizando **Programación Entera**, dado que no podemos alquilar una porción solamente de una máquina virtual. Quedando el problema final de la siguiente manera:

$$\begin{aligned} \min costo : & \sum_{v \in V} c_v * x_v \\ S.A & \\ & \sum_{v \in V} x_v * q_{rv} >= d_r; \forall r \in R \\ & \sum_{i=1}^{n} x_i >= 0; x \in \mathbb{N} \end{aligned} \tag{12}$$

4.1 AMPL solver
Para resolver este problema de programación entera, utilizamos AMPL con su modulo **cplex**.

AMPL
AMPL Solver es un programa diseñado para resolución de problemas de programación lineal. Así como Excel Solver, este provee una versión gratuita que permite resolver problemas con un limitado número de restricciones y variables.

Para nuestro caso específico ha sido suficiente para resolver nuestro problema general.

Para resolver este problema utilizamos dos fuentes de información.

1. Un archivo *.mod* que contiene la definición del modelo a corregir 2. Un archivo *.dat* que contiene los datos a usar para resolver el problema.

Ambos archivos están disponibles en el repositorio de GitHub del proyecto.

4.2 La solución

El modelo de AMPL utilizado es el siguiente:

```
# Reducción de costos de infraestructura

option solver cplex; # Programación Entera

# Definiendo conjuntos
set V; # Tipos de máquinas
set R; # Tipos de recursos a utilizar


# Definiendo parámetros
param c{V}; # costo por tipo de máquina
param d{R}; # demanda por tipo de recurso

param q{V, R}; # cantidad de recurso R por tipo de máquina V

# Definiendo variables
# Cantidad X de máquinas a elegir por tipo V,
# + restricción de No negatividad
var x{V} integer >= 0;

minimize costo:
    sum{v in V} c[v]*x[v]
;

subject to demandas {r in R}:
    sum{v in V} x[v] * q[v, r] >= d[r]
;

data "project.dat";

solve;

option omit_zero_rows 1;
display costo, x;
```

Figure 1: AMPL model

Cabe resaltar que utilizar Software diseñado para resolver problemas de Programación Lineal, hace todo el proceso mucho más sencillo.

4.3 Los datos

La ventaja de usar AMPL es que podemos ser capaces de definir un modelo que sea agnóstico a los datos. Aún así para propósitos de este documento, usaremos un set de datos que puedan probar el funcionamiento del modelo.

Estos datos pueden descargados desde este link. Ellos cuenta la siguiente distribución de datos:

Tipos de máquinas	154
Tipos de recursos	6

Resultados

Haciendo une evaluación para poder calcular le mejor combinación de máquinas por hora, de los siguientes recursos.

CPU	60 unidades
Memoria	400 GB
Disco	2 GB
Network	100 Gb/s
GPU	0 unidades
IOPS	1000 IOPS

Obtenemos los siguientes resultados que nos indican que el precio por hora que estaríamos pagando por 18 máquinas es de $2.3019, y según el modelo la mejor combinación es:

- **1** máquina **e2-highmem-2**
- **12** máquinas **e2-highmem-4**
- **5** máquinas **e2-micro**

```
project > ≡ project.dat
~/github.com/baquiax/ppl2/project
   4   param: d :=
   5           "CPU"       60
   6           "Memory"    400
   7           "Disk"      2
   8           "Network"   100
   9           "GPU"       0
  10           "IOPS"      6000;
  11

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  GITLENS

→  project git:(main) ampl project.mod
CPLEX 20.1.0.0: optimal integer solution; objective 2.3019
21 MIP simplex iterations
0 branch-and-bound nodes
costo = 2.3019

x [*] :=
    e2-highmem-2   1
    e2-highmem-4  12
        e2-micro   5
;
```

Figure 2: AMPL result

5 Conclusiones

El resultado de esta investigación de muestra la aplicación de la Programación Entera en situaciones comunes dentro de ámbitos laborales, como el mío. Así también la simplicidad que nos dan aplicaciones como AMPL que nos ayudan a generalizar problemas y a automatizarlos.

El problema que cubrimos acá es un problema real, con costos reales e impacto real. Durante la primera fase de este investigación hallamos una forma determinista de calcular número de máquinas virtuales a alquilar. Mi intención es llevar este modelo a la práctica real y evaluar iterativamente su impacto.