

Mega Servidores

Integrantes:

- Mario Benavente**
- Felipe Gómez**
- Leonardo Rikhardsson**

Sección 2 - Grupo 2

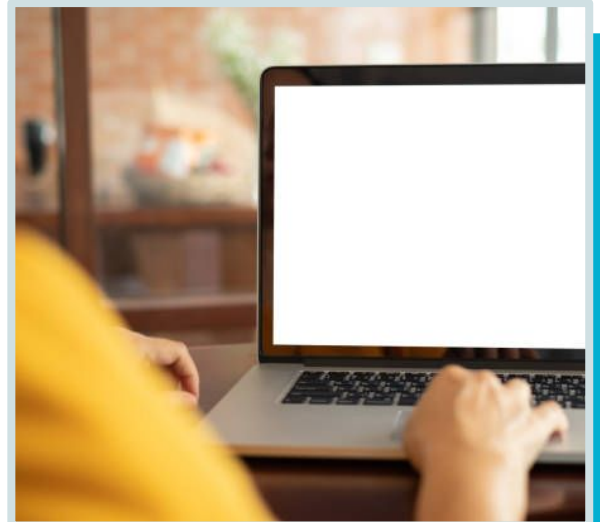
Sobre nuestro Proyecto

Tenemos 100 computadores y K mega-servidores, donde deberemos asignar un servidor a cada computador. Se nos comenta que, por un problema aún no identificado, cada computador es incapaz de compartir servidor con 3 de los 99 computadores restantes, los cuales se encuentran en una lista.

“Usted trabaja en una empresa y es el encargado de asignar K distintos mega-servidores a los 100 computadores de la empresa. La única limitación corresponde a que cada computador puede estar conectado a un solo mega-servidor [...] asumimos que [sus recursos] son ilimitados. Sin embargo, por un error aún no detectado, cada computador presenta una incompatibilidad con otros 3 computadores, que le impide conectarse a un servidor que se encuentre conectado a alguno de estos 3 computadores. Luego, para cada computador existe una lista con los números de los computadores con los que tiene este problema. [...] A usted le asignan la importante tarea de que entregue una asignación posible para que todos los computadores puedan mantener una conexión estable con su mega-servidor.”

"El Reto: Asignación de Mega-Servidores a Computadoras"

Nuestro proyecto consiste en modelar una solución al problema planteado, en función de lógica de predicados. Luego habrá que traducirlo a CNF (Forma Normal Conjuntiva) para poder trabajarlo programáticamente mediante SAT Solvers, y finalmente encontrar el mínimo valor de K para el cual el problema es siempre satisfactible.



Objetivo

El objetivo de este proyecto es abordar el problema presentado, respetando las siguientes restricciones:

- Cada computador debe estar conectado al menos a un mega-servidor
- Cada mega-servidor debe tener al menos un computador conectado a éste
- Cada computador tiene una lista negra: 3 computadores con los que no puede compartir servidor debido a un error aún no identificado

Solución planteada

Modele la situación como un problema de satisfacibilidad:

Plantearemos el problema como una función de lógica proposicional (ϕ) que se haga verdadera si se cumplen las restricciones planteadas por el enunciado.

Para que el problema tenga sentido, necesitamos definir los conjuntos y enunciados con los cuales trabajaremos:

Conjunto de los computadores: $C = \{1, \dots, 100\}$

Conjunto de los mega-servidores : $S = \{1, \dots, K\}$

Lista de conexiones: $N = \{X_{i,j} \mid i \in C, j \in S\}$

$$\sigma(X_{i,j}) = \begin{cases} 1 & \text{si } i \text{ está conectado a } j \\ 0 & \text{si no} \end{cases}$$

Lista de incompatibilidad : $L = \{Y_{i,r} \mid i, r \in C\}$

$$\sigma(Y_{i,r}) = \begin{cases} 1 & \text{si } r \text{ está en la lista negra de } i \\ 0 & \text{si no} \end{cases}$$

Solución planteada

Con esto listo, partiremos identificando las restricciones de nuestro problema:

- i) Cada computador debe estar conectado a un único mega-servidor
- ii) Cada mega-servidor tiene al menos un computador conectado a éste
- iii) Una “lista negra” es válida si y sólo si posee únicamente 3 computadores
- iv) Un computador no puede compartir mega-servidor con ninguno de los 3 computadores en su lista

Solución planteada

i) Cada computador debe estar conectado a un único mega-servidor:

Notamos que, para cada computador i en C , buscamos que, para todo par de mega-servidores j_1 y j_2 en S , con j_1 distinto de j_2 , si el computador i está conectado a j_1 , no puede estar conectado a j_2 .

$$\bigwedge_{i=1}^{100} \left(\bigwedge_{1 \leq j_1, j_2 \leq K} (X_{i,j_1} \Rightarrow \neg (X_{i,j_2})) \right)$$

Solución planteada

ii) Cada mega-servidor tiene al menos un computador conectado a éste:

Ésta se resuelve rápido, pues solamente necesitamos que X_{ij} sea verdadero al menos una vez para todo mega-servidor j en S , y para al menos un computador i en C con tal de cumplir con la restricción.

$$\bigwedge_{j=1}^K \left(\bigvee_{i=1}^{100} X_{i,j} \right)$$

Solución planteada

iii) Una “lista negra” es válida si y sólo si posee únicamente 3 computadores:

Usando esta vez el enunciado Y_{ir} , necesitamos que, para todo computador i en C , tengamos solamente 3 computadores r_1 , r_2 y r_3 , que formen parte de su “lista negra”. Es decir, para todo trío de computadores r_1 , r_2 , r_3 , en C , distintos entre sí, solamente una combinación de computadores pertenece a la lista, esto para cada computador i en C .

$$\bigwedge_{1 \leq i \leq 100} \left(\bigvee_{1 \leq r_1, r_2, r_3 \leq 100} (Y_{i,r_1} \wedge Y_{i,r_2} \wedge Y_{i,r_3}) \right)$$

Solución planteada

iv) Un computador no puede compartir mega-servidor con ninguno de los 3 computadores en su lista:

Podemos visualizar esto como el siguiente escenario: para todo conjunto de computadores i, r_1, r_2, r_3 en C , distintos entre sí, donde tenemos que r_1, r_2 y r_3 están en la lista negra de i ($Y_{ir_1}, Y_{ir_2}, Y_{ir_3}$), si también se cumple que i está conectado a un mega-servidor j , entonces tanto r_1 como r_2 y r_3 no pueden conectarse a j , esto para todo servidor en S .

$$\bigwedge_{1 \leq i, r_1, r_2, r_3 \leq 100} \left(\bigwedge_{j=1}^K ((Y_{i,r_1} \wedge Y_{i,r_2} \wedge Y_{i,r_3} \wedge X_{i,j}) \Rightarrow \neg (X_{r_1,j} \vee X_{r_2,j} \vee X_{r_3,j})) \right)$$

Solución planteada

Finalmente, nuestra función de lógica proposicional (ϕ) equivale a aplicar conjunciones a todas las fórmulas recién planteadas. Como podrán apreciar a continuación, es muy larga!

$$\begin{aligned} \phi = & \left(\bigwedge_{i=1}^{100} \left(\bigwedge_{1 \leq j_1, j_2 \leq K} (X_{i,j_1} \Rightarrow \neg (X_{i,j_2})) \right) \right) \wedge \left(\bigwedge_{j=1}^K \left(\bigvee_{i=1}^{100} X_{i,j} \right) \right) \wedge \\ & \left(\bigwedge_{1 \leq i \leq 100} \left(\bigvee_{1 \leq r_1, r_2, r_3 \leq 100} (Y_{i,r_1} \wedge Y_{i,r_2} \wedge Y_{i,r_3}) \right) \right) \wedge \left(\bigwedge_{1 \leq i, r_1, r_2, r_3 \leq 100} \left(\bigwedge_{j=1}^K ((Y_{i,r_1} \wedge Y_{i,r_2} \wedge Y_{i,r_3} \wedge X_{i,j}) \Rightarrow \neg (X_{r_1,j} \vee X_{r_2,j} \vee X_{r_3,j})) \right) \right) \end{aligned}$$

Solución planteada

Transforme la fórmula booleana anterior a CNF:

Hay dos cosas que podemos hacer para transformar la fórmula ya planteada a CNF:

1) Definición de Implicancia

$$p \rightarrow q \equiv \neg p \vee q$$

2) Leyes de De Morgan

$$\neg(p \wedge q) \equiv \neg p \vee \neg q$$

$$\neg(p \vee q) \equiv \neg p \wedge \neg q$$

Solución planteada

1) Definición de Implicancia:

$$\begin{aligned} \phi = & \left(\bigwedge_{i=1}^{100} \left(\bigwedge_{1 \leq j_1, j_2 \leq K} (\neg X_{i,j_1} \vee \neg X_{i,j_2}) \right) \right) \wedge \\ & \left(\bigwedge_{j=1}^K \left(\bigvee_{i=1}^{100} X_{i,j} \right) \right) \wedge \\ & \left(\bigwedge_{1 \leq i \leq 100} \left(\bigvee_{1 \leq r_1, r_2, r_3 \leq 100} (Y_{i,r_1} \wedge Y_{i,r_2} \wedge Y_{i,r_3}) \right) \right) \wedge \\ & \left(\bigwedge_{1 \leq i, r_1, r_2, r_3 \leq 100} \left(\bigwedge_{j=1}^K (\neg (Y_{i,r_1} \wedge Y_{i,r_2} \wedge Y_{i,r_3} \wedge X_{i,j}) \vee \neg (X_{r_1,j} \vee X_{r_2,j} \vee X_{r_3,j})) \right) \right) \end{aligned}$$

Solución planteada

2a) Ley de De Morgan para conjunción:

$$\begin{aligned}\phi = & \left(\bigwedge_{i=1}^{100} \left(\bigwedge_{1 \leq j_1, j_2 \leq K} (\neg X_{i,j_1} \vee \neg X_{i,j_2}) \right) \right) \wedge \\ & \left(\bigwedge_{j=1}^K \left(\bigvee_{i=1}^{100} X_{i,j} \right) \right) \wedge \\ & \left(\bigwedge_{1 \leq i \leq 100} \left(\bigvee_{1 \leq r_1, r_2, r_3 \leq 100} (Y_{i,r_1} \wedge Y_{i,r_2} \wedge Y_{i,r_3}) \right) \right) \wedge \\ & \left(\bigwedge_{1 \leq i, r_1, r_2, r_3 \leq 100} \left(\bigwedge_{j=1}^K ((\neg Y_{i,r_1} \vee \neg Y_{i,r_2} \vee \neg Y_{i,r_3} \vee \neg X_{i,j}) \vee \neg (X_{r_1,j} \vee X_{r_2,j} \vee X_{r_3,j})) \right) \right)\end{aligned}$$

Solución planteada

2b) Ley de De Morgan para disyunción:

$$\begin{aligned}\phi = & \left(\bigwedge_{i=1}^{100} \left(\bigwedge_{1 \leq j_1, j_2 \leq K} (\neg X_{i,j_1} \vee \neg X_{i,j_2}) \right) \right) \wedge \\ & \left(\bigwedge_{j=1}^K \left(\bigvee_{i=1}^{100} X_{i,j} \right) \right) \wedge \\ & \left(\bigwedge_{1 \leq i \leq 100} \left(\bigvee_{1 \leq r_1, r_2, r_3 \leq 100} (Y_{i,r_1} \wedge Y_{i,r_2} \wedge Y_{i,r_3}) \right) \right) \wedge \\ & \left(\bigwedge_{1 \leq i, r_1, r_2, r_3 \leq 100} \left(\bigwedge_{j=1}^K ((\neg Y_{i,r_1} \vee \neg Y_{i,r_2} \vee \neg Y_{i,r_3} \vee \neg X_{i,j}) \vee (\neg X_{r_1,j} \wedge \neg X_{r_2,j} \wedge \neg X_{r_3,j})) \right) \right)\end{aligned}$$

Convertir CNF a SAT Solver | PySat

Por sugerencia de enunciado, para resolver el problema mediante un SAT Solver, utilizamos PySat (un SAT Solver para Python). Nos plantean que el problema será un archivo .txt de 101 líneas, donde la primera estipula la cantidad de servidores, y las otras 100 líneas son de la forma: [Índice], [PC Restringido 1], [PC Restringido 2], [PC Restringido 3]

Esto se puede visualizar en la siguiente imagen de ejemplo, y seguido de esto mostraremos resultados al tratar de resolver los archivos de prueba que se nos entregaron:

```
1 10
2 1,2,19,23
3 2,5,8,13
4 3,100,2,5
5 ...
6 ...
7 100,99,2,4
8
```


Convertir CNF a SAT Solver (Soluciones) | PySat

test-false1.txt

No tiene solución porque hay dos computadoras (41 y 47) que están incompatibles y deben asignarse al mismo servidor.

test-false1.txt

No tiene solución porque los computadores del 4 al 100, tienen incompatibilidad con el 1, y el 2 y 3 tendrían que estar solos.

Convertir CNF a SAT Solver (Soluciones) | PySat

test-true1.txt

Servidor 1: [1, 12, 23, 34, 45, 56, 67, 78, 89, 100]

Servidor 2: [2, 13, 24, 35, 46, 57, 68, 79, 90, 1]

Servidor 3: [3, 14, 25, 36, 47, 58, 69, 80, 91, 2]

Servidor 4: [4, 15, 26, 37, 48, 59, 70, 81, 92, 3]

Servidor 5: [5, 16, 27, 38, 49, 60, 71, 82, 93, 4]

Servidor 6: [6, 17, 28, 39, 50, 61, 72, 83, 94, 5]

Servidor 7: [7, 18, 29, 40, 51, 62, 73, 84, 95, 6]

Servidor 8: [8, 19, 30, 41, 52, 63, 74, 85, 96, 7]

Servidor 9: [9, 20, 31, 42, 53, 64, 75, 86, 97, 8]

Servidor 10: [10, 21, 32, 43, 54, 65, 76, 87, 98, 9]

Convertir CNF a SAT Solver (Soluciones) | PySat

test-true2.txt

Servidor 1: [1, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100]

Servidor 2: [2, 3]

Mínimo valor de K para que el problema sea siempre satisfactible

Tanto por los resultados anteriores como por intuición, podemos buscar una cota superior sobre la cual encasillar K .

Primero, notamos que K necesariamente debe ser al menos 4 porque cada computador tiene una lista con 3 computadores distintos entre sí.

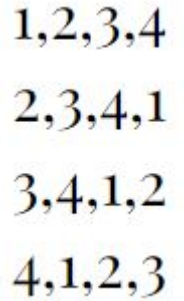
Segundo, para facilitar la búsqueda de esta cota, asumimos que las listas son de forma cíclica. Esto es, para un computador i , su lista consiste de los computadores $i+1$, $i+2$ e $i+3$, y da la vuelta a los índices si es que $i+1$, $i+2$ y/o $i+3$ es/son mayor(es) que 100.

Mínimo valor de K para que el problema sea siempre satisfactible

Siguiendo el razonamiento anterior, tendríamos que, para 4 computadores, el .txt se vería como el esquema a la derecha:

Es directo notar que, para cada computador, necesitamos un mega-servidor.

Pero, ¿qué pasaría para 5 o más computadores?



1,2,3,4
2,3,4,1
3,4,1,2
4,1,2,3

Mínimo valor de K para que el problema sea siempre satisfactible

En este caso, vemos que, dentro de la “lista negra” del computador 1 no se encuentra el computador 5, pero el 1ro aparece en la lista del 5to.

Siendo este el caso, necesitaríamos 5 mega-servidores.

Siguiendo esta línea de pensamiento, los casos de 6 y 7 computadores son similares, los cuales necesitan 6 y 7 servidores respectivamente.

Pero, veamos que, para 8 computadores, solo necesitamos 4, pues tenemos las líneas “1,2,3,4” y “5,6,7,8”, lo que significa que podemos alocar los computadores 1 y 5 en el mismo servidor y, similarmente, emparejar el 2do y el 6to, 3 y 7, 4 y 8.

1,2,3,4

2,3,4,5

3,4,5,1

4,5,1,2

5,1,2,3

Mínimo valor de K para que el problema sea siempre satisfactible

En base a lo anterior, probaremos por inducción que, para cualquier cantidad $n \geq 4$ de computadores, podemos satisfacer el problema con 7 servidores:

- 1) Tomamos como casos iniciales el razonamiento que utilizamos para 4, 5, 6 y 7 computadores, sabiendo que necesitamos, respectivamente, 4, 5, 6 y 7 servidores, siendo trivial que cada uno puede resolverse con a lo más 7 servidores.
- 2) Hipótesis Inductiva: Suponemos que, para una cantidad $n > 7$ de computadores, el problema es satisfactible.
- 3) Procedemos con el Paso Inductivo, donde buscaremos demostrar que esto sirve para un $i = n + 1$:

Mínimo valor de K para que el problema sea siempre satisfactible

Notamos que, por la naturaleza del problema, es posible visualizar la situación como añadir un i -ésimo computador a un problema ya satisfecho para n computadores.

Por Principio del Palomar, si removemos este i -ésimo computador, sabemos que el problema está satisfecho, donde tendríamos al menos un servidor que contiene al menos $\lceil n/7 \rceil$ computadores. Seguido de esto, tenemos dos casos:

i) $\lceil n/7 \rceil = \lceil i/7 \rceil$: si añadir el i -ésimo elemento al problema no altera el Principio del Palomar, entonces el problema sigue siendo satisfactible, pues tendríamos algún servidor donde ubicar el computador i sin romper restricciones.

Mínimo valor de K para que el problema sea siempre satisfactible

ii) $\lceil i/7 \rceil = \lceil n/7 \rceil + 1$: contrario a lo anterior, añadir este i -ésimo elemento al problema SI altera el Principio del Palomar, diciéndonos que ahora hay al menos un mega-servidor donde están conectados $\lceil n/7 \rceil + 1$ servidores. Siendo ese el caso, ahora simplemente bastaría con alocar este i -ésimo elemento en cualquiera de los servidores que cumpla con nuestras restricciones, pues todos, antes de la adición del último elemento, tienen la misma cantidad $\lceil n/7 \rceil$ de elementos.

Por lo tanto, hemos llegado a que el mínimo valor de K para que el problema sea siempre satisfactible es 7.

Conclusiones

Hemos demostrado que, si el problema se traduce adecuadamente a la lógica proposicional, es posible aprovechar un SAT Solver para resolver casos específicos para este problema, todo de forma programática: o el código nos dice que no es posible resolverlo para los datos entregados, o nos entrega una forma de solucionarlo.

Sumado a eso, se llegó a un mínimo valor K para el cual el problema siempre puede resolverse, de donde nuestra intuición nos dice que para cualquier cantidad de servidores mayor o igual que 7, el problema puede resolverse siempre.

¿Dudas o Preguntas?
