UNIVERSIDAD TORCUATO DI TELLA

Tecnología Digital V: Diseño de algoritmos
Licenciatura en Tecnología Digital
Segundo Semestre, 2023

TP1 - Asignación de instancias en la nube

Motivación

La computación en la nube, o Cloud Computing, se ha convertido en un pilar fundamental en el ámbito tecnológico, redefiniendo la forma en que las organizaciones gestionan y acceden a sus recursos informáticos. Esta tecnología permite a las empresas escalar sus operaciones con una flexibilidad sin precedentes, optimizando costos mediante la utilización de modelos de pago por uso y eliminando la necesidad de mantener infraestructuras físicas costosas.
Las empresas tienen la habilidad de solicitar la asignación de instancias de computo(puede ser máquinas virtuales, bases de datos u otros sistemas). La asignación de estas instancias a las máquinas disponibles en el proveedor de la nube es un problema complejo y crítico dentro del ámbito de la computación en la nube. Este proceso, conocido como orquestación de recursos, implica determinar cómo distribuir de manera óptima las cargas de trabajo de los usuarios en los servidores físicos disponibles. Este problema es de particular interés debido a varias razones:

1. Optimización de Recursos: La eficiencia en la asignación de instancias afecta directamente el uso de recursos del proveedor de la nube. Una asignación subóptima puede llevar a un uso ineficiente de la infraestructura, aumentando los costos operativos y reduciendo el retorno de inversión.

2. Rendimiento del Servicio: Una asignación adecuada garantiza que los recursos computacionales, como CPU, memoria y ancho de banda, sean utilizados de manera óptima, asegurando un alto rendimiento de las aplicaciones. Esto es crucial para mantener la satisfacción del cliente y cumplir con los acuerdos de nivel de servicio (SLAs).

3. Minimización de Latencia: La proximidad de las instancias a los usuarios finales puede influir en la latencia de las aplicaciones. Una asignación bien gestionada puede minimizar la latencia al ubicar las instancias en regiones geográficas óptimas, mejorando así la experiencia del usuario.

Resolver este problema de manera efectiva no solo mejora la eficiencia operativa del proveedor de la nube, sino que también proporciona beneficios significativos a los usuarios, como un mejor rendimiento, costos reducidos y una experiencia más confiable y consistente. Dado el crecimiento continuo de la computación en la nube, encontrar soluciones innovadoras y eficientes para la asignación de instancias sigue siendo un área de investigación activa y esencial en la ingeniería de sistemas distribuidos.
Dentro de los recursos con los que cuentan los clúster en la nube se encuentran las GPU, que son ampliamente utilizadas en aplicaciones de aprendizaje automático y procesamiento de datos. La asignación de instancias de GPU en la nube es un problema particularmente desafiante debido a la alta demanda de estos recursos[1]. En este contexto, la asignación de

[1] https://www.theinformation.com/articles/ai-developers-stymied-by-server-shortage-at-aws-micr

instancias de GPU en la nube es un problema crítico que requiere soluciones eficientes y escalables para garantizar la eficiencia operativa.
Cada instancia solicitada por los usuarios tiene un beneficio, una ganancia asociada, que representa el beneficio que el proveedor obtiene al asignar esa instancia. Esto se obtiene de distintos factores que se determinan al momento de solicitar la instancia (como el tipo de instancia, la disponibilidad esperada del usuario, restricciones geográficas, etc).
El objetivo en el cual se va a enfocar este trabajo práctico es el de maximizar el beneficio total obtenido por el proveedor de la nube al asignar las instancias a las máquinas disponibles haciendo foco en la asignación de las GPUs. Para ello, se cuenta con un conjunto de instancias (con las GPU solicitadas y su beneficio) y un conjunto de máquinas disponibles en el proveedor de la nube que tienen distintas cantidades de GPU.

El problema

Sean $(g_1, b_1), \ldots, (g_n, b_n) \in \mathbb{R}^2$ una lista de n instancias donde g_i representa la cantidad de GPUs requeridas por la $i-$ésima instancia y b_i el beneficio asociado a la misma. Además, se cuenta con una lista de m máquinas disponibles en el proveedor de la nube, donde cada máquina j tiene una cantidad G_j de GPUs disponibles. Sea x_{ij} una variable binaria que indica si la instancia i es asignada a la máquina j. El problema que buscamos resolver es:

$$\text{máx} \sum_{i=1}^{n} \sum_{j=1}^{m} b_i \cdot x_{ij} \tag{1}$$

$$\text{sujeto a} \sum_{i=1}^{n} g_i \cdot x_{ij} \leq G_j \qquad \forall j = 1, \ldots, m \tag{2}$$

$$\sum_{j=1}^{m} x_{ij} \leq 1 \qquad \forall i = 1, \ldots, n \tag{3}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i = 1, \ldots, n, j = 1, \ldots, m \tag{4}$$

Donde (1) indica la función objetivo que es la suma de los beneficios por cada instancia asignada a una máquina. En (2) se garantiza que la cantidad de instancias asignadas a la máquina j sea menor o igual a la cantidad de GPUs disponibles en la misma. La desigualdad (3) asegura que cada instancia sea asignada a lo sumo a una máquina. Finalmente, (4) establece que las variables de decisión son binarias.
Este problema es conocido como el Multiple 0-1 Knapsack Problem (MKP) y es un problema NP-completo[2]. Es una variante conocida del problema de la mochila. Postulado de manera general, el problema de la mochila consiste en determinar la cantidad de elementos de un conjunto finito que pueden ser incluidos en una mochila de capacidad limitada de manera que se maximice el valor total de los elementos incluidos. En el caso del MKP, se tienen múltiples mochilas, cada una con una capacidad distinta, y se busca determinar la cantidad de elementos que pueden ser incluidos en cada mochila de manera que se maximice el valor total de los elementos incluidos.

[2]No se conocen algoritmos con complejidad polinomial para resolverlo.

Las instancias

Queremos evaluar dos alternativas para que el proveedor de la nube pueda tomar una decisión con respecto a qué tipo de máquinas comprar. Las alternativas que se están evaluando son: tener dos máquinas con 1000 GPU, o tener cuatro máquinas con 500 GPU.
Entre otros factores, lo que se busca es determinar en cuál de las dos alternativas se pueden encontrar soluciones con mayor facilidad.
Para cada una de las alternativas se consideran escenarios con distinta cantidad de instancias solicitadas variando el promedio de GPU por instancia.

El trabajo

El trabajo práctico debe ser realizado en grupos de exactamente 3 personas. La resolución del trabajo se compone de varias etapas, incluyendo el diseño de 3 algoritmos, programación, experimentación, así también como el reporte detallado de la evaluación de los métodos implementados, los resultados obtenidos y la discusión sobre el caso real. **Todos los algoritmos deben ser implementados en `C++` y además el algoritmo de Programación Dinámica debe implementarse en Python**. Se pide:

1. **Fuerza bruta.** Proponer e implementar un algoritmo de fuerza bruta.

2. **Backtracking.** Partiendo de la implementación anterior, implementar un backtracking incorporando al menos 1 poda por factibilidad y 1 poda por optimalidad.

3. **Programación dinámica.** Implementar un algoritmo de programación dinámica, incluyendo la rutina de reconstrucción de la solución. Para el contexto de este TP, se puede implementar por separado el caso de dos, tres y cuatro máquinas en caso de que resulte conveniente.

4. **Experimentación y discusión.** Realizar una experimentación exhaustiva considerando las alternativas siendo evaluadas. Considerar distinta cantidad de instancias y sus requerimientos, abordar las siguientes preguntas:

   - *Performance*: ¿En qué tipo de escenarios funciona mejor un algoritmo que otro? ¿Cómo impacta la cantidad de GPUs por instancia en el rendimiento de los algoritmos?
   - *Lenguaje*: ¿Cómo afecta la elección del lenguaje de programación y la implementación elegida?
   - *Propuesta*: En base a los análisis previos, ¿cuál sería la sugerencia del grupo respecto al algoritmo, implementación y lenguaje de programación?

5. **Algoritmos, informe, presentación de resultados y entrega del código.** El modelo, la descripción de los algoritmos, los operadores, las decisiones de diseño, la implementación, el testing realizado, la presentación de resultados, instrucciones de compilación y ejecución.

Modalidad de entrega

Se pide presentar el modelo y la experimentación en un informe de máximo 15 páginas que contenga:

- introducción al problema y la decisión,
- descripción de los algoritmos propuestos e implementados, según corresponda,
- consideraciones generales respecto a la implementación del modelo, incluyendo dificultades que hayan encontrado,
- resumen de resultados obtenidos en la experimentación,
- conclusiones, posibles mejoras y observaciones adicionales que consideren pertinentes.

Junto con el informe debe entregarse el código con la implementación del modelo. El mismo debe ser entendible, incluyendo comentarios que faciliten su corrección y ejecución.

Fechas de entrega

Formato Electrónico: **jueves 19 de septiembre de 2024, 23:59 hs**, enviando el trabajo (informe + código) vía el campus virtual.
Importante: El horario es estricto. Las entregas recibidas después de la hora indicada serán consideradas re-entrega.