Optimización de la carga de trabajo en computación en la nube usando el algoritmo Grey Wolf Optimizer

Santiago Sotelo, Natalia Rincón, Juan Muñoz

October 2024

1 Introducción

La optimización de recursos en infraestructuras de computación en la nube es crucial para maximizar el rendimiento y minimizar costos operativos. Este proyecto aborda el desafío de asignación de recursos en clústeres de servidores, donde una gestión ineficiente puede llevar a saturación o subutilización de los mismos, afectando la calidad del servicio. Para esto, se ha seleccionado el dataset *Alibaba Cluster Trace 2017*, en particular los apartados Batch_Tasks y Server_Usage, los cuales proporcionan información detallada sobre el consumo de CPU y memoria de cada tarea y la utilización de los servidores en tiempo real.

El objetivo de este estudio es optimizar la asignación de recursos mediante el Grey Wolf Optimizer (GWO) , un algoritmo inspirado en el comportamiento jerárquico de los lobos grises, que ha demostrado eficacia en la exploración de espacios de búsqueda complejos. En este caso se busca balancear la carga y maximizar la utilización de recursos, distribuyéndolos de manera equilibrada y eficiente entre las tareas del clúster.

El algoritmo de optimización de lobos grises es una metaheurística inspirada en los lobos grises (Canis lupus). El algoritmo GWO imita la jerarquía de liderazgo y el mecanismo de caza de los lobos grises en la naturaleza. Para simular la jerarquía de liderazgo se emplean cuatro tipos de lobos grises, como alfa, beta, delta y omega. Además, se implementan los tres pasos principales de la Caza: buscar presas, rodear presas y atacar presas. [1]

1.1 ¿Cómo funciona el algoritmo de lobos?

Jerarquia

- La solución más apta como lobo Alfa
- Segunda mejor solución como lobo Beta

- Tercera mejor solución como lobo Delta
- Resto de soluciones candidatas como lobos Omega

1.2 Fases de caza

1. Búsqueda: Los lobos alfa, beta y delta guían la búsqueda de la presa, explorando el espacio de soluciones para identificar las mejores opciones.
2. Cercamiento: Los lobos se acercan a la solución, ajustando sus posiciones basándose en la distancia a los mejores lobos (alfa, beta, delta).
3. Ataque: Una vez la solución está cercana, el algoritmo refina la posición de los lobos (soluciones), convergiendo a un óptimo. [1]

2 Planteamiento del problema

2.1 Contexto

En los sistemas de computación en la nube, la asignación de tareas de procesamiento y el uso eficiente de los recursos de servidor son aspectos fundamentales que afectan directamente el rendimiento, la disponibilidad y los costos operativos. A medida que los servicios en la nube se expanden, la administración efectiva de los recursos se vuelve cada vez más compleja, pues requiere ajustarse dinámicamente a las variaciones en la demanda y las condiciones de los recursos. Cuando la carga de trabajo no se distribuye adecuadamente, los servidores pueden experimentar problemas como la sobrecarga de CPU, falta de memoria, saturación de red y, en casos críticos, fallos de hardware, los cuales comprometen la continuidad del servicio y aumentan los gastos operativos [2].

Uno de los principales desafíos es la fluctuación constante en la demanda de recursos, que puede hacer que algunos servidores operen a su capacidad máxima mientras otros permanecen subutilizados. Esto no solo genera cuellos de botella y aumenta los tiempos de respuesta, sino que también encarece las operaciones debido a una administración ineficaz de los recursos. Asimismo, la variabilidad en los tipos de carga —con tareas que presentan diferentes requerimientos de CPU, memoria y almacenamiento— exige soluciones de asignación de recursos que sean capaces de adaptarse en tiempo real para evitar problemas como la congestión de red y la saturación de entrada/salida de disco [3].

Además, la gestión de servidores en entornos en la nube enfrenta desafíos de seguridad significativos, como el riesgo de ataques DDoS, intrusiones y la necesidad de mantener actualizado el software para protegerlo de vulnerabilidades conocidas. El mantenimiento y las actualizaciones regulares son esenciales, pues sin ellos los servidores se vuelven vulnerables a fallos que pueden comprometer tanto el rendimiento como la integridad de los datos [4].

El uso de técnicas de optimización como el Grey Wolf Optimizer (GWO) permite enfrentar estos desafíos mediante la asignación dinámica de tareas, adaptándose a los cambios de carga y mejorando la eficiencia en el uso de los recursos. En este proyecto, el conjunto de datos Alibaba Cluster Trace 2017 proporciona información detallada sobre tareas y uso de recursos en un entorno de clúster real, permitiendo modelar un sistema de optimización que responde a la capacidad y estado actual de los servidores en tiempo real. Así, el GWO puede distribuir las tareas de forma eficiente, priorizando los servidores menos costosos y evitando picos de saturación, con lo cual se reducen los costos operativos y se maximiza la disponibilidad del sistema [5].

3 Descripción del dataset

El conjunto de datos Alibaba Cluster Trace 2017 ofrece un panorama detallado sobre la operación de un clúster de servidores, registrando métricas esenciales para entender y optimizar la asignación de recursos en entornos de computación en la nube. Los datos se dividen en dos secciones principales, Batch_Tasks y Server_Usage, que contienen la siguiente información:

1. Batch_Tasks

Propósito: Este apartado documenta las tareas de procesamiento por lotes, que suelen ser intensivas en recursos y varían en términos de duración y carga. Esta información permite analizar la demanda de cada tarea y ajustar su asignación en función de los recursos disponibles.

Atributos Principales:

task_id (int): Identificador único para cada tarea. Es un número entero que permite rastrear la ejecución y el estado de cada tarea a lo largo del tiempo.

instance_type (string): Especifica el tipo de instancia utilizada para ejecutar la tarea. Este valor categórico describe las configuraciones de recursos asignadas, como la cantidad de CPU y memoria, y es clave para adaptar la asignación de tareas a servidores con distintas capacidades.

start_time y end_time (timestamp): Marcas de tiempo que indican el inicio y finalización de cada tarea. Estos valores en formato de fecha y hora son esenciales para calcular la duración de cada tarea y evaluar cómo se distribuye la carga de trabajo a lo largo del tiempo.

cpu_usage (float): Indica el porcentaje de CPU utilizado por cada tarea durante su ejecución. Este dato numérico es crucial para identificar las tareas de alta demanda y ajustar su distribución de acuerdo con la capacidad de los servidores.

memory_usage (float): Porcentaje de uso de memoria requerido por cada tarea, en relación con la capacidad de la instancia asignada. Este valor ayuda a balancear la asignación de recursos y evitar que los servidores alcancen su capacidad máxima.

priority_level (int): Nivel de prioridad de cada tarea, expresado como un valor entero, que indica la urgencia o importancia de la tarea en el clúster. Las tareas de mayor prioridad pueden recibir asignaciones de servidores más potentes para cumplir con plazos específicos o condiciones críticas de procesamiento.

2. Server_Usage

Propósito: Este apartado documenta el uso de recursos de cada servidor en tiempo real, proporcionando información vital sobre el estado y rendimiento de los servidores en distintos momentos. Esta sección es clave para monitorizar la capacidad de cada servidor y optimizar la asignación de tareas en función de su estado actual.

Atributos Principales:

server_id (int): Identificador único para cada servidor en el clúster. Este número entero permite realizar un seguimiento del rendimiento de cada servidor de forma individual.

cpu_utilization (float): Porcentaje de uso de CPU en tiempo real para cada servidor. Este valor es fundamental para entender la capacidad de procesamiento actual y evitar la sobrecarga de CPU en servidores que ya están cercanos a su límite.

memory_utilization (float): Porcentaje de uso de memoria en cada servidor, que permite monitorizar la disponibilidad de memoria y gestionar las asignaciones para evitar sobrecargas.

network_io (float): Mide la tasa de entrada/salida de red (en Mbps) para cada servidor, un valor esencial para evaluar el tráfico de red y evitar problemas de congestión que puedan afectar el rendimiento de las tareas.

disk_io (float): Mide la tasa de entrada/salida de disco (en Mbps), un indicador crucial para observar cómo las actividades de lectura/escritura en el disco afectan el rendimiento de las tareas.

timestamp (timestamp): Marca temporal para cada registro de uso del servidor, en formato de fecha y hora. Este valor permite construir un patrón histórico del uso de recursos y anticipar picos de demanda, mejorando la toma de decisiones en la asignación de tareas.

3.1 Relación de los datos con el modelo de optimización

Los datos de Batch_Tasks y Server_Usage son fundamentales para desarrollar un modelo de optimización eficaz, pues ofrecen un panorama detallado de cómo las tareas consumen recursos específicos y del estado actual de cada servidor en el clúster. Al integrar estas métricas en el Grey Wolf Optimizer (GWO), el modelo puede asignar tareas de manera que se minimicen los costos operativos, se maximice el uso de recursos y se evite la sobrecarga. La combinación de los datos históricos de tareas y el monitoreo en tiempo real del uso de servidores permite que el GWO responda dinámicamente a cambios en la carga de trabajo, ajustando las asignaciones de acuerdo con las capacidades y demandas de los servidores.

4 Actualización del Contexto

La asignación de recursos en la computación en la nube es un área crítica y en constante evolución, especialmente debido a la naturaleza dinámica de los entornos en la nube, donde la demanda de recursos puede variar significativamente en periodos cortos de tiempo. Tradicionalmente, se han empleado métodos de asignación sencillos como Round-Robin y First-Come, First-Served, que distribuyen las tareas de manera equitativa o en orden de llegada. Sin embargo, estos métodos presentan limitaciones, ya que no consideran las características específicas de las tareas ni las variaciones en la capacidad de los servidores, lo que puede llevar a asignaciones subóptimas y sobrecarga en ciertos nodos [5].

En los últimos años, las técnicas de optimización basadas en metaheurísticas han ganado popularidad para abordar los desafíos de asignación de recursos de manera más eficiente. Algoritmos como Particle Swarm Optimization (PSO) y Ant Colony Optimization (ACO) son ampliamente utilizados debido a su capacidad para encontrar soluciones óptimas en problemas de gran complejidad. PSO, inspirado en el comportamiento social de las aves, busca el equilibrio entre la exploración y explotación en el espacio de búsqueda para asignar tareas de manera eficiente [6]. Por su parte, ACO, que emula el comportamiento de búsqueda de rutas de las hormigas, es efectivo en la optimización de rutas y en la asignación de recursos en redes de comunicación [7]. Sin embargo, ambos métodos pueden enfrentar problemas de convergencia prematura y de ajuste dinámico en entornos donde la demanda cambia rápidamente [8].

En este contexto, el Grey Wolf Optimizer (GWO) ha emergido como una metaheurística prometedora para la asignación de recursos en la nube. El GWO, inspirado en la jerarquía y el comportamiento de caza de los lobos grises, utiliza una estructura de liderazgo en la que las soluciones candidatas (lobos) se dividen en roles jerárquicos (alfa, beta, delta y omega), lo que permite guiar la búsqueda hacia soluciones óptimas mediante un proceso de acercamiento a la presa (solución óptima). Esta estructura permite un balance efectivo entre

exploración y explotación, adaptándose a cambios en la carga de trabajo de manera ágil y eficiente [9]. A diferencia de otros métodos, el GWO es especialmente útil en entornos donde se requiere una reasignación constante de recursos para evitar sobrecargas, pues su dinámica permite ajustar las asignaciones en tiempo real [10].

5 Propuesta de solución

5.1 Objetivo

El objetivo principal de este proyecto es desarrollar un modelo de optimización para la asignación de tareas en un entorno de computación en la nube, minimizando los costos operativos y maximizando la utilización de recursos de manera equilibrada. Para lograrlo, el modelo debe asignar tareas a servidores específicos dentro del clúster, considerando la capacidad de cada servidor (uso de CPU, memoria, almacenamiento, y entrada/salida de red) y el costo de operación de cada uno de estos recursos, evitando la sobrecarga y subutilización.

De esta forma, la asignación de recursos debe permitir una distribución eficiente de las tareas, donde los servidores que operan a menor costo sean preferidos, y se evite la saturación de los servidores con alta carga de trabajo. El modelo debe ser capaz de ajustar las asignaciones dinámicamente, en respuesta a cambios en la demanda y disponibilidad de recursos.

5.2 Función objetivo

La función objetivo de este modelo se orienta a minimizar el costo total de uso de los servidores, representado como la suma del costo de CPU, memoria, red y almacenamiento utilizados por cada tarea en función de la capacidad de cada servidor. Este enfoque permite ajustar dinámicamente las asignaciones de recursos, priorizando los servidores que operan a menor costo y evitando la saturación de aquellos con alta carga de trabajo [11].

5.3 Definición Formal de la Función Objetivo

Dado un conjunto de tareas $T = \{t_1, t_2, \ldots, t_n\}$ y un conjunto de servidores $S = \{s_1, s_2, \ldots, s_m\}$, definimos la función objetivo como costo total de operación (CTO) en la que queremos minimizar el costo total de operación de los servidores, el cual depende del costo por uso de CPU, memoria, red y disco. La función objetivo se puede expresar como:

$$\text{CTO} = \sum_{i=1}^{m} (C_{\text{cpu},i} \cdot U_{\text{cpu},i} + C_{\text{mem},i} \cdot U_{\text{mem},i} + C_{\text{net},i} \cdot U_{\text{net},i} + C_{\text{disk},i} \cdot U_{\text{disk},i})$$

Donde:

- $C_{\text{cpu},i}$, $C_{\text{mem},i}$, $C_{\text{net},i}$, $C_{\text{disk},i}$ representan los costos unitarios de uso de CPU, memoria, red y disco para el servidor s_i [12].
- $U_{\text{cpu},i}$, $U_{\text{mem},i}$, $U_{\text{net},i}$, $U_{\text{disk},i}$ indican el uso de cada recurso en el servidor s_i, calculado como la suma del uso de cada tarea asignada al servidor [10].

5.4 Restricciones

Capacidad de los recursos: Cada servidor tiene límites en los recursos que puede manejar, lo cual se representa mediante las siguientes restricciones:

$$U_{\text{cpu},i} \leq \text{Cap}_{\text{cpu},i}, \quad U_{\text{mem},i} \leq \text{Cap}_{\text{mem},i}, \quad U_{\text{net},i} \leq \text{Cap}_{\text{net},i}, \quad U_{\text{disk},i} \leq \text{Cap}_{\text{disk},i}$$

Donde $\text{Cap}_{\text{cpu},i}$, $\text{Cap}_{\text{mem},i}$, $\text{Cap}_{\text{net},i}$, y $\text{Cap}_{\text{disk},i}$ son las capacidades máximas de CPU, memoria, red y disco para cada servidor s_i [11]. Y cada tarea debe asignarse a un único servidor, garantizando que la carga se distribuya sin duplicación:

$$\sum_{j=1}^{m} x_{i,j} = 1 \quad \forall i \in T$$

Donde $x_{i,j}$ es una variable binaria que toma el valor de 1 si la tarea t_i se asigna al servidor s_j, y 0 en caso contrario. Y algunas tareas pueden tener un nivel de prioridad mayor, lo que requiere asignarlas a servidores con suficiente capacidad o menor costo. Esta restricción asegura que las tareas críticas se atiendan conforme a su urgencia y requisitos, evitando que queden relegadas en la asignación [1].

5.5 Solución

En la solución desarrollada, los lobos alfa representan las soluciones óptimas actuales, mientras que los betas y deltas son las siguientes mejores soluciones que guían a los lobos omega hacia mejores regiones en el espacio de búsqueda [1].

- **Lobo Alfa (α)**: La mejor solución actual, que guía el movimiento de toda la manada hacia el óptimo. [1]
- **Lobo Beta (β)**: Segunda mejor solución, que apoya la dirección del alfa. [1]
- **Lobo Delta (δ)**: Tercera mejor solución, que también influencia la dirección de la manada. [1]
- **Lobos Omega (ω)**: Representan el resto de las soluciones, que siguen a los líderes y exploran el espacio de búsqueda. [1]

5.5.1 Etapas de Caza en el GWO

- **Búsqueda (Search)**: Los lobos alfa, beta y delta exploran el espacio de búsqueda para identificar las mejores soluciones posibles. En esta etapa, se fomenta la exploración para descubrir regiones prometedoras. [1]
- **Cercamiento (Encircle)**: Los lobos se aproximan gradualmente a las mejores soluciones ajustando sus posiciones en función de la distancia a los lobos líderes. [1]
- **Ataque (Attack)**: Conforme las soluciones convergen, los lobos se acercan a las posiciones de los líderes alfa, beta y delta, refinando la búsqueda y explotando las mejores áreas del espacio. [1]

La actualización de la posición de cada lobo X se define por:

$$X(t+1) = X(t) - A \cdot |C \cdot X_\alpha - X(t)|$$

donde:

- X_α, X_β, y X_δ son las posiciones de los lobos líderes. [1]
- A y C son coeficientes que controlan la convergencia y el acercamiento de las soluciones hacia los líderes [1]

5.6 Implementación del GWO con el Conjunto de Datos Alibaba Cluster Trace 2017

La solución desarrollada utiliza los datos de *Batch_Tasks* y *Server_Usage* del conjunto de datos *Alibaba Cluster Trace 2017* para alimentar el GWO y optimizar la asignación de tareas de la siguiente manera:

1. **Preparación de los Datos**:
   - **Batch_Tasks**: Se utilizan los campos task_id, cpu_usage, memory_usage, y priority_level para representar las tareas que necesitan asignación. Las métricas de uso de CPU y memoria de cada tarea se emplean como entradas para definir la carga de trabajo que cada tarea impone al servidor seleccionado.
   - **Server_Usage**: Se toman los valores de cpu_utilization, memory_utilization, network_io, y disk_io de cada servidor, así como sus capacidades máximas, para modelar el estado actual del clúster. Esto permite al algoritmo seleccionar los servidores que minimicen los costos operativos sin superar sus capacidades.
2. **Definición de la Función Objetivo**: La función objetivo del GWO se orienta a minimizar el costo total de operación (CTO), que depende del uso y costo unitario de CPU, memoria, red y disco en cada servidor. La función objetivo se expresa como:

$$\text{CTO} = \sum_{i=1}^{m} (C_{\text{cpu},i} \cdot U_{\text{cpu},i} + C_{\text{mem},i} \cdot U_{\text{mem},i} + C_{\text{net},i} \cdot U_{\text{net},i} + C_{\text{disk},i} \cdot U_{\text{disk},i})$$

donde $C_{\text{cpu},i}$, $C_{\text{mem},i}$, $C_{\text{net},i}$, y $C_{\text{disk},i}$ representan los costos unitarios por servidor y $U_{\text{cpu},i}$, $U_{\text{mem},i}$, $U_{\text{net},i}$, $U_{\text{disk},i}$ el uso de cada recurso en el servidor s_i.

3. **Proceso de Optimización**:
   - **Inicialización**: Se inicializan los lobos en posiciones aleatorias en el espacio de búsqueda, que representan diferentes asignaciones de recursos.
   - **Iteración**: En cada iteración, los lobos se mueven hacia las posiciones de los lobos alfa, beta y delta. Las tareas se reasignan a servidores que optimizan la función objetivo, respetando las restricciones de capacidad y prioridades de cada tarea.
   - **Actualización de Jerarquía**: Al final de cada iteración, los lobos alfa, beta y delta se actualizan para reflejar las mejores soluciones encontradas hasta ese momento.
   - **Convergencia**: El algoritmo continúa hasta alcanzar un criterio de parada, que en este caso es el número máximo de iteraciones.
4. **Asignación de Recursos y Validación**: El modelo final de GWO proporciona una asignación óptima de tareas a servidores. Para validar los resultados, se compara el costo total de operación (CTO) alcanzado con métodos tradicionales y se evalúa la distribución de la carga de trabajo para asegurar que los recursos se utilicen eficientemente, evitando sobrecargas y subutilización.

6 Referencias

1 S. Mirjalili, "Grey Wolf Optimizer (GWO)" [Online]. Available: https://seyedalimirjalili.com/gwo

2 K. Gai, M. Qiu, and H. Zhao, "Cost-aware multimedia data allocation for heterogeneous memory using genetic algorithm in cloud computing," IEEE Transactions on Cloud Computing, vol. 4, no. 1, pp. 34-45, 2016. doi:10.1109/TCC.2015.2459727.

3 B. P. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," 2010 Fifth International Joint Conference on INC, IMS and IDC, 2011, pp. 44-51. doi:10.1109/NCM.2010.5606969.

4 K. Hashizume, D. G. Rosado, E. Fernández-Medina, and E. B. Fernandez, "An analysis of security issues for cloud computing," Journal of Internet Services and Applications, vol. 4, no. 1, pp. 1-13, 2013. doi:10.1186/1869-0238-4-5.

5 Alibaba, "Alibaba cluster trace 2017," Alibaba Group, 2017. [Online]. Available: https://github.com/alibaba/clusterdata.

6 R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities," in *Proceedings of the 2009 International Conference on High Performance Computing & Simulation*, 2009, pp. 1–11. doi:10.1109/HPCSIM.2009.5192685.

7 J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, Perth, WA, Australia, 1995, pp. 1942–1948. doi:10.1109/ICNN.1995.488968.

8 M. Dorigo and L. M. Gambardella, "Ant colonies for the traveling salesman problem," *Biosystems*, vol. 43, no. 2, pp. 73–81, 1997. doi:10.1016/S0303-2647(97)01708-5.

9 R. S. Parpinelli and H. S. Lopes, "New inspirations in swarm intelligence: a survey," International Journal of Bio-Inspired Computation, vol. 3, no. 1, pp. 1–16, 2011. doi:10.1504/IJBIC.2011.038398.

10 Y. Cao, Q. Ding, and B. Hu, "A multi-objective optimization approach based on grey wolf optimizer for task scheduling in cloud computing," Applied Soft Computing, vol. 93, 2020. doi:10.1016/j.asoc.2020.106311.

11 B. Addis, D. Ardagna, B. Panicucci, and L. Zhang, "Autonomic management of cloud service centers with availability guarantees," IEEE Transactions on Network and Service Management, vol. 9, no. 2, pp. 202–213, 2012. doi:10.1109/TNSM.2012.032312.110017.

12 S. J. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 3rd ed., Upper Saddle River, NJ, USA: Prentice-Hall, 2009.