Asignación de Prioridades Fijas en Sistemas Distribuidos de Tiempo Real Utilizando un Algoritmo de Descenso por Gradiente

Juan M. Rivas y J. Javier Gutiérrez

Grupo de Ingeniería Software y Tiempo Real, Universidad de Cantabria, Santander
{rivasjm, gutierjj}@unican.es

Palabras clave: sistemas distribuidos, asignación prioridades fijas, optimización, descenso por gradientenote.

Resumen extendido. En la literatura podemos encontrar una diversa variedad de políticas de planificación, cada una con diferentes capacidades para planificar, complejidad computacional, limitaciones a su aplicabilidad, etc. Sin embargo, la política clásica de prioridades fijas sigue siendo la más utilizada en entornos industriales [1], principalmente por su adecuado rendimiento, relativa baja complejidad, y alto soporte en sistemas operativos de tiempo real [2].

De igual manera, las plataformas *hardware* también han visto una gran evolución debido a los crecientes requisitos que imponen aplicaciones modernas en dominios como el de la automoción o aviónica. Este impulso lleva a que en la actualidad las plataformas más habituales utilizadas en la industria tengan carácter distribuido [1], en el que la carga computacional está compuesta de flujos que se distribuyen en una serie de nodos interconectados por redes de comunicación como el bus CAN.

Por todo ello, los sistemas distribuidos planificados por prioridades fijas siguen representando uno de los principales focos de atención en la industria. En este tipo de sistemas, la computación se divide en flujos de principio a fin (*end-to-end flows*) que se activan periódicamente (o con un tiempo mínimo entre activaciones), formados por una secuencia de *steps*. Cada *step* es planificado en un nodo concreto, y por lo tanto debe tener asignado un valor de prioridad fija, que se establece *offline*. Esta asignación de prioridades fijas a todos los *steps* es un paso crítico, ya que una asignación deficiente puede provocar un desaprovechamiento de los recursos computacionales, e incluso la incapacidad de cumplir los requisitos temporales impuestos.

El proceso de asignación de prioridades fijas en sistemas distribuidos es un problema NP, para el cual se han propuesto numerosos algoritmos [2]. Por un lado, se han propuesto adaptaciones de algoritmos de optimización de propósito general, tales como el templado simulado, algoritmos genéticos o programación lineal con enteros mixta (siglas MILP en inglés) [2]. También se han propuesto algoritmos heurísticos diseñados específicamente para este problema, como HOPA [3]. En general se constata que algoritmos de tipo MILP son capaces de aportar soluciones óptimas, aunque poseen claros problemas de escalabilidad que reducen su aplicabilidad a sistemas relativamente simples, con un número no elevado de *steps*, cargas en los nodos por debajo del 70%, y rango reducido de periodos. Por otro lado, algoritmos específicos como HOPA son computacionalmente más eficientes, pero sin embargo ofrecen soluciones que en general son incapaces de aproximarse a las soluciones óptimas.

En este trabajo buscamos aportar una nueva técnica de asignación de prioridades fijas que se sitúe en un punto intermedio, es decir, que aporte soluciones que se aproximen a la solución óptima, pero manteniendo una escalabilidad adecuada. Con este objetivo proponemos adaptar el algoritmo de optimización de propósito general de Descenso por Gradiente (*Gradient Descent* en inglés) para su aplicación al problema de asignación de prioridades fijas en sistemas distribuidos. Algoritmos de tipo Descenso por Gradiente son actualmente extensamente utilizados en el ámbito del *Machine Learning*, ya que son la base sobre la que se construyen los algoritmos de entrenamiento de redes neuronales. En este tipo de sistemas, los algoritmos basados en el Descenso por Gradiente han demostrado ser capaces de optimizar funciones con millones de parámetros.

El objetivo de los algoritmos de Descenso por Gradiente es el de minimizar una función matemática, también conocida como la función coste. Su funcionamiento básico es el siguiente: partiendo de un punto inicial, el siguiente punto se calcula con una traslación en la dirección opuesta al gradiente de la función coste, que matemáticamente representa la dirección de mayor descenso de la función. Este proceso se repite hasta que se detecta un mínimo. Para maximizar la probabilidad de que el mínimo encontrado sea el mínimo global, se han propuesto modificaciones que añaden un factor adicional de aceleración acumulada durante el descenso [4]. Intuitivamente, estas modificaciones emulan el proceso físico por el que una esfera con masa desciende por una superficie por el efecto de la gravedad, y a medida que desciende puede acumular momento que más adelante puede utilizar para sobrepasar mínimos locales.

Proponemos modificar el algoritmo genérico de Descenso por Gradiente para su aplicación en la asignación de prioridades fijas en sistemas distribuidos de la siguiente manera:

- Planteamos una función coste que representa el inverso del *slack* en el sistema (*invslack)*. De esta manera, minimizar la función coste es equivalente a mejorar la planificabilidad del sistema.

Este trabajo ha sido financiado en parte por MCIN/ AEI /10.13039/501100011033/ FEDER “Una manera de hacer Europa” a través del proyecto PID2021-124502OB-C42 (PRESECREL).

- Relajamos la restricción clásica de que los valores de las prioridades fijas sean números enteros, permitiendo números decimales.
- Cada gradiente está conformado por las derivadas parciales de la función coste con respecto a la prioridad de cada *step*. Para su cálculo proponemos utilizar un incremento no infinitesimal.
- El cálculo del gradiente requiere aplicar un análisis de tiempos de respuesta 2*N veces, siendo N el número de *steps* en el sistema. Para acelerar este proceso proponemos vectorizar este análisis.
- Para maximizar la probabilidad de obtener un mínimo global proponemos utilizar la técnica Adam [2]

Hemos llevado a cabo una evaluación preliminar del algoritmo GDPA (*Gradient Descent Priority Assignment*) sobre sistemas sintéticos. Para ello GDPA se ha implementado en Python utilizando la librería Numpy. Se han generado 50 sistemas distribuidos formados por 4 flujos de principio a fin con 4 *steps* cada uno, distribuidos en 4 nodos. La utilización de todos ellos se varía desde el 50% hasta el 90%. Se compara el rendimiento de una serie de algoritmos de asignación de prioridades fijas: (1) PD que es un algoritmo sencillo y no iterativo; (2) HOPA; (3) *Brute-Force* que explora todas las combinaciones posibles de asignaciones; (4) GDPA-Random, que es el algoritmo GDPA comenzando con una asignación inicial aleatoria; (5) GDPA-PD que es GDPA comenzando con la asignación obtenida con PD; y (6) GDPA-HOPA que es GDPA comenzando con la asignación obtenida con HOPA. Los resultados preliminares obtenidos se resumen en la Ilustración 1.

En la figura (a) se muestra el número total de sistemas que se consiguieron planificar en función de la utilización media de los nodos. Como era de esperar, *Brute-Force* domina ya que siempre es capaz de encontrar una solución si esta existe. Adicionalmente podemos observar que todos los algoritmos GDPA se aproximan al algoritmo *Brute-Force*, mejorando además los resultados obtenidos por HOPA.

En la figura (b) se muestran los tiempos medios de ejecución de cada algoritmo para encontrar una solución planificable. Se observa que los algoritmos GDPA son más lentos que HOPA, aunque la diferencia no es acusada, y siempre alejadas del algoritmo *Brute-Force*. Se observa además que, entre los algoritmos que utilizan el Descenso por Gradiente, GDPA-Random es el que más tiempo requirió. Esto es debido a que en general siempre comienza con una solución peor, por lo que requiere de más iteraciones para encontrar una solución planificable.

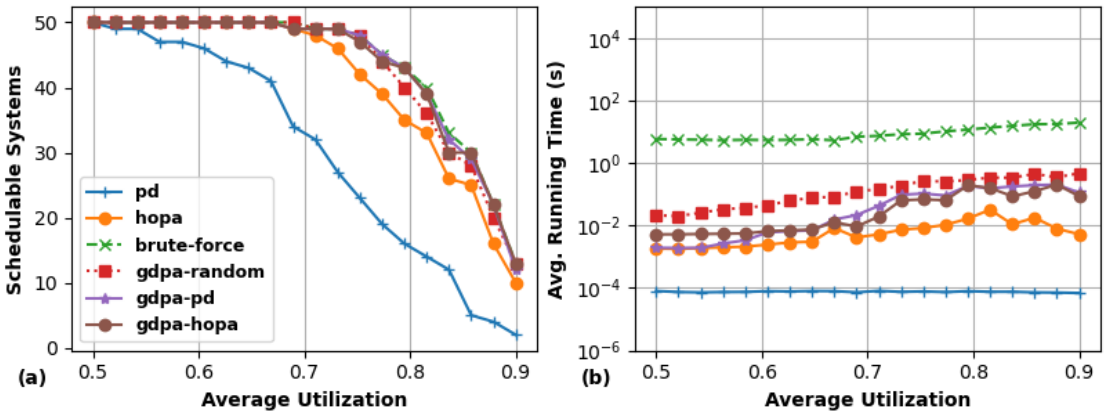

Ilustración 1 Resultados de la evaluación preliminar de GDPA. La figura (a) muestra el número total de sistemas planificados en función de la utilización media del sistema. La figura (b) muestra el tiempo medio de ejecución de cada algoritmo para encontrar una solución planificable en función de la utilización media del sistema

Con este punto de partida, como trabajo futuro nos proponemos extender este estudio para abarcar sistemas más complejos, además de estudiar el efecto de los diferentes hiper-parámetros que configuran el algoritmo de Descenso por Gradiente.

REFERENCIAS

[1] Akesson, B., Nasri, M., Nelissen, G., Altmeyer, S., Davis, R. I. (2022). A comprehensive survey of industry practice in real-time systems. Real-Time Systems, 58(3), 358–398. https://doi.org/10.1007/s11241-021-09376-1

[2] Davis, R. I., Cucu-Grosjean, L., Bertogna, M., Burns, A. (2016). A review of priority assignment in real-time systems. In Journal of Systems Architecture (Vol. 65, pp. 64–82). Elsevier B.V. https://doi.org/10.1016/j.sysarc.2016.04.002

[3] Gutiérrez, J. J., Harbour, M. G. (1995). Optimized priority assignment for tasks and messages in distributed hard real-time systems. Proceedings of the 3rd Workshop on Parallel and Distributed Real-Time Systems, WPDRTS 1995, 124–132. https://doi.org/10.1109/WPDRTS.1995.470498

[4] Ruder, S. (2016). An overview of gradient descent optimization algorithms. http://arxiv.org/abs/1609.04747