Paralelização do algoritmo k-medoids para agrupamento de sinais fracos

Vivian Renata Nunes Toito, André Leon S. Gradvohl

Faculdade de Tecnologia – Universidade Estadual de Campinas (UNICAMP)
Caixa Postal 456 – 13.484-370 – Limeira – SP – Brasil

v118949@dac.unicamp.br, gradvohl@ft.unicamp.br

Abstract. *This paper presents some results about the project which aims parallelizing k-medoids algorithm for weak signals clustering. Weak signals clustering is a process that supports the decision making. For fast results, the program is implemented using OpenMP, which uses multicore architecture and shared memory. Initially, tests were performed with the serial version and, after, with the parallels version, that registered a low runtime in most of them.*

Resumo. *Este artigo apresenta resultados prévios sobre o projeto que visa a paralelização do algoritmo k-medoids para agrupamento de sinais fracos. O agrupamento de sinais fracos é um processo que auxilia a tomada de decisão. Para obter respostas mais rápidas, o programa é implementado usando o OpenMP, que utiliza arquitetura de múltiplos núcleos e memória compartilhada. Inicialmente, foram realizados testes com a versão serial e, em seguida, com versões paralelizadas, registrando na maioria deles uma baixa significativa de tempo de execução.*

1. Introdução

O algoritmo k-medoids surgiu como uma solução para o problema na área de mineração de dados em que grandes agrupamentos de dados tornavam o próprio processo muito complexo. Uma definição formal sobre este algoritmo, segundo Barioni (2006), é que, de maneira geral, o objetivo dos algoritmos de detecção de agrupamentos baseados no algoritmo k-medoids consiste em encontrar um conjunto de agrupamentos não sobrepostos de modo que cada agrupamento possua um objeto representante - denominado *medoid* - sendo este o objeto mais próximo possível do centro do agrupamento. O *medoid* é, portanto, o elemento que possui a menor distância entre os outros elementos do grupo e o mais centralmente localizado.

O primeiro algoritmo baseado em k-medoids denominado *Partitioning Around Medoids* (PAM), ilustrado na Figura 1, tem como estratégia a troca de um objeto *medoid* por outro não-*medoid* até serem encontrados os melhores agrupamentos possíveis a fim de manter a qualidade dos grupos encontrados (QIAO et al 2011). Essa troca acontece a partir do cálculo de uma variável denominada "custo" que é determinada, neste projeto em específico, pela equação da distância Euclidiana e segue a premissa de que quanto menor for o valor calculado quando comparado ao anterior, maior será a qualidade do grupo que será formado. No caso da Figura 1, pelo fato do

candidato à *medoid*, representado pela cor vermelha, ter um maior custo – 26 – quando comparado ao custo do *medoid* que é representado pela cor azul escuro – 20 –, não é feita a troca e o *medoid* atual é mantido.

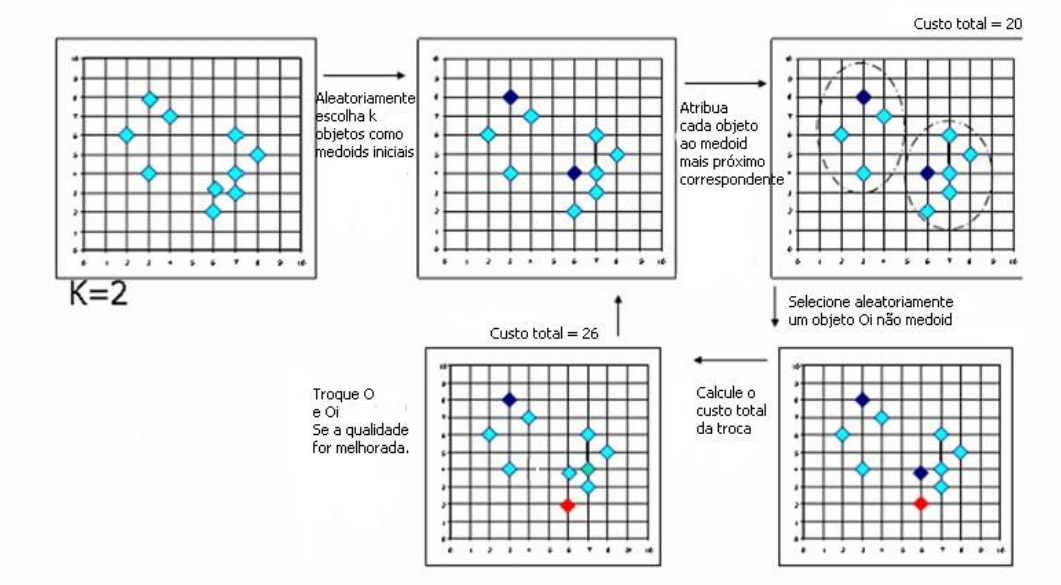

Figura 1. Funcionamento do algoritmo k-medoids

Quando se trata de uma grande base de dados, o algoritmo k-medoids demanda um grande tempo de processamento para mostrar resultados. Sendo assim, se faz útil a paralelização deste a fim de tentar diminuir seu tempo de execução. Sinais fracos podem ser definidos como informações imprecisas que juntas podem apontar tendências futuras sendo essas, segundo Coffman (1997), oportunidades ou ameaças para uma organização. Dessa forma, é essencial um algoritmo eficiente de classificação de sinais fracos, como o k-medoids que tem como característica o reconhecimento de padrões.

2. Resultados Prévios

Neste projeto, a versão serial é comparada à versão paralelizada do algoritmo desenvolvido por Todd Gamblin (2010) na linguagem de programação C++, em relação ao tempo de execução e eficiência. Na Figura 2, é possível observar o comportamento do algoritmo k-medoids em sua versão serial e utilizada nesse projeto. Nesse caso, os elementos pertencentes ao mesmo grupo são destacados pela mesma cor, sendo os medoids aqueles representados pelo caractere “o”.

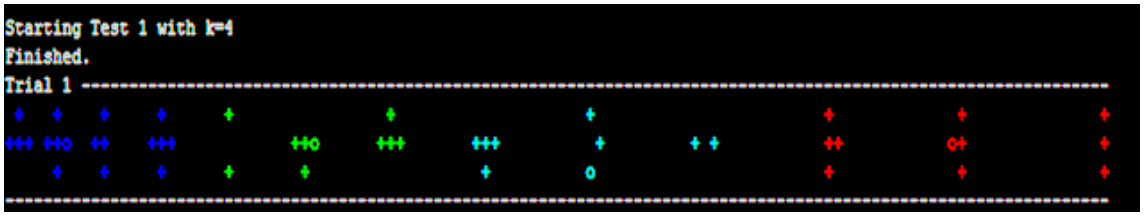

Figura 2. Versão serial do algoritmo k-medoids

A interface de programação de aplicação (API) OpenMP é a biblioteca utilizada para paralelização do algoritmo k-medoids, pois se baseia em uma arquitetura de múltiplos núcleos e memória compartilhada. Os trechos escolhidos para serem paralelizados foram aqueles que continham, principalmente, iterações encadeadas, pois estas demandam um alto processamento e levam mais tempo para serem executadas.

Os resultados dos testes mostram que ao paralelizar os três módulos centrais do algoritmo k-medoids: `kmedoids.h`, `kmedoids.cpp` – que apresentam os cálculos de custo e trocas dos medoids – e `dissimilarity.h` – que apresenta os cálculos de similaridade – em relação ao número de núcleos, há uma queda significativa do tempo de execução. A Figura 3 ilustra graficamente o comportamento dos resultados com os tempos médios que são derivados da média de três iterações do algoritmo em cada caso e a Figura 4 apresenta o *speedup* dos tempos registrados. É importante destacar que a versão serial apresentou tempo médio de 115,585 segundos. A partir desses testes, é possível determinar se a paralelização do k-medoids será tão eficiente quanto necessário para ser utilizado nos objetivos propostos.

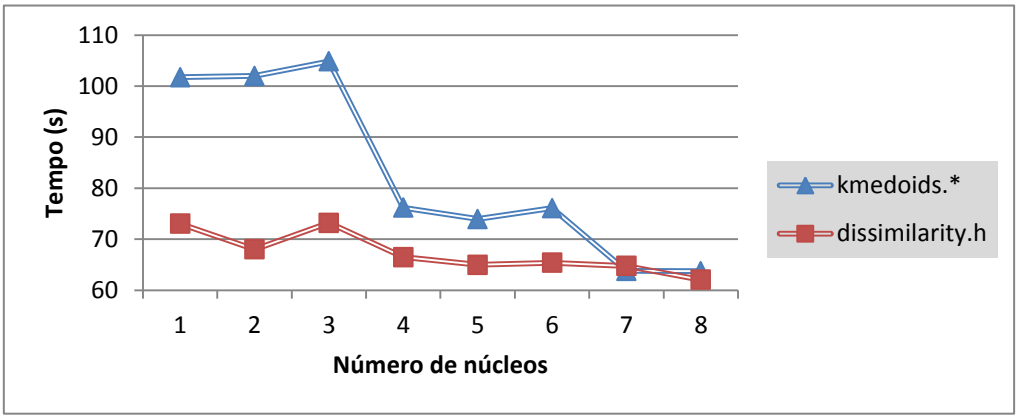

Figura 3. Gráfico da comparação entre os tempos registrados na paralelização de módulos do algoritmo em função do número de núcleos utilizados.

A hipótese que pode ser levantada para tal comportamento se dá pelo fato do módulo ser responsável por gerar as matrizes de similaridade do algoritmo que darão origem a k agrupamentos, sendo esta uma tarefa com alto custo de processamento. Dessa forma, quando paralelizada, o trabalho dos processadores é amenizado e, consequentemente, o tempo de execução é reduzido gradativamente conforme o número de núcleos aumenta. No que se refere aos módulos `kmedoids.cpp` e `kmedoids.h` houve uma queda brusca de tempo nos testes com 4 núcleos e 8 núcleos justamente por serem o máximo de núcleos físicos e lógicos disponíveis, respectivamente.

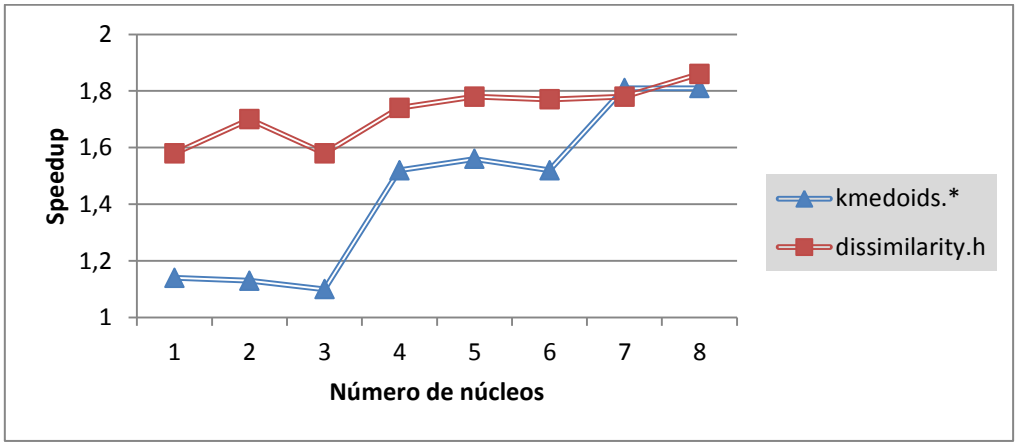

Figura 4. Gráfico do *speedup* da paralelização de módulos do algoritmo em função do número de núcleos utilizados.

Os testes foram feitos em ambiente Linux através de acesso remoto utilizando uma máquina de 32 Gigabytes de memória RAM com processador AMD FX, com 8 núcleos lógicos (4 físicos), 4,2 GHz. Os dados de entrada utilizados consistem em 1000 pares ordenados com o valor de k = 6. Ainda estão sendo realizados testes com diferentes valores de k para análise de resultados com mais precisão.

Conclusão

Com os dados apresentados, é possível observar que a paralelização dos módulos principais do algoritmo apresentaram resultados significativos quando comparado com o tempo obtido a partir da versão serial, em particular o módulo `dissimilarity.h`. Desta forma, concluímos que a partir dos testes já realizados a paralelização do algoritmo se mostra eficaz e apresenta resultados positivos quando o conjunto é formado por números inteiros. Em trabalhos futuros, faremos uma análise mais concentrada em informações textuais que geralmente compõem os sinais fracos.

3. Referências Bibliográficas

Barioni, M. C. N. Operações de consulta por similaridade em grandes bases de dados complexos. São Carlos, 2006. 145 p. Tese de Doutorado - Instituto de Ciências Matemáticas e de Computação - ICMC, USP.

Coffman, B. Weak Signal Research, Part I: Introduction, Journal of Transition Management, MG Taylor Corporation, 1997. Disponível on-line em: http://www.mgtaylor.com/mgtaylor/jotm/winter97/wsrintro.htm.

Gamblin, T. “Muster Documentation”, Disponível on-line em: http://tgamblin.github.com/muster/main.html.

Shaoyu Qiao; Xinyu Geng; Min Wu, "An Improved Method for K_Medoids Algorithm," Business Computing and Global Informatization (BCGIN), 2011 International Conference on , vol., no., pp.440,444, 29-31 July 2011