Explorando Mapeamentos de *Threads* e Dados para Melhoria de Desempenho de Algoritmos de Aprendizado de Máquina*

Matheus W. Camargo[1], **Matheus S. Serpa**[1], **Danilo Carastan-Santos**[1], **Alexandre Carissimi**[1], **Philippe O. A. Navaux**[1]

[1] Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970, Porto Alegre – RS – Brasil

{mwcamargo, msserpa, danilo.csantos, asc, navaux}@inf.ufrgs.br

Resumo. *Algoritmos de Aprendizado de Máquina (ML) são cada vez mais utilizados em diversos problemas científicos e industriais, tendo como uma preocupação importante o tempo de execução destes algoritmos. Neste trabalho exploramos mapeamentos de threads e dados e o impacto em novos algoritmos de ML. Resultados experimentos mostraram que os algoritmos SRCNN e MobileNet apresentaram tempos de execução até 15.1% menores e mais uniformes.*

1. Introdução

Devido ao crescimento da quantidade de dados e poder de processamento hoje disponível, aprendizado de máquina (do Inglês *Machine Learning, ML*) é uma área de pesquisa que encontra-se em constante progresso. Tal progresso é fomentado pelo crescente emprego de ML em diversas aplicações científicas e industriais, tais como em sistemas de detecção de fraudes, mecanismos de recomendação, carros autônomos, previsão de demanda e até mesmo serviços de diagnóstico médico automatizado [Culkin and Das 2017, Stavens et al. 2011, Perols 2011].

No entanto, o surgimento de algoritmos de ML mais complexos, aliado ao aumento da quantidade de dados disponível, acarreta em uma demanda cada vez maior por poder computacional. Estudar formas de melhorar a execução destes algoritmos torna-se, portanto, uma tarefa imprescindível. Neste contexto, o mapeamento de *threads* e dados apresenta-se como um bom recurso, visto que tais técnicas podem fornecer ganhos de desempenho com baixíssimo custo de implantação [Serpa et al. 2018].

O presente trabalho visa portanto explorar como algoritmos recentes de aprendizado de máquina reagem a diferentes mapeamentos. Para isto foi utilizado um conjunto de *benchmarks* de ML, denominado AI-Benchmark [Ignatov et al. 2019], sendo executado com diversos mapeamentos de *threads* e dados. Resultados experimentais mostraram que grande parte dos algoritmos do AI-Benchmark são insensíveis à tais mapeamentos. Contudo, um pequeno conjunto de algoritmos apresentaram não somente ganhos de desempenho, mas também tempos de execução mais previsíveis, quando comparados a execução desses mesmos algoritmos sem mapeamento.

*Este trabalho foi parcialmente financiado pelo projeto Petrobras 2016/00133-9, pelo projeto Petrobras 5900.0111175.19.9 e pelo projeto "GREEN-CLOUD: Computação em Cloud com Computação Sustentável" (#16/2551-0000 488-9), da FAPERGS e do CNPq, programa PRONEX 12/2014.

2. Trabalhos Relacionados

A relevância das políticas de mapeamento de *threads* e dados no desempenho de algoritmos de ML já havia sido apontada em trabalhos anteriores [Diener et al. 2016, Serpa et al. 2018, Serpa et al. 2019]. Foram analisados notadamente os algoritmos de ML K-Nearest Neighbors (KNN), Kmeans e Backpropagation. Os resultados experimentais mostraram reduções no tempo de execução de até 25.2%, quando comparado ao tempo de execução dos algoritmos sem mapeamento.

No entanto, os trabalhos supracitados também observaram que os ganhos de desempenho dependem muito dos padrões de utilização de memória e compartilhamento de dados entre as *threads* dos algoritmos, tornando desafiador caracterizar quais tipos de algoritmos podem se beneficiar dos mapeamentos de *threads* e dados sem executá-los previamente. Nesse sentido, o presente trabalho visa estender o estudo iniciado pelos trabalhos acima mencionados, levando em conta um conjunto mais amplo e recente de algoritmos de ML, que constituem o conjunto *benchmark* de algoritmos AI-Benchmark.

3. Método

Nesta seção apresentamos como avaliamos o impacto dos diferentes mapeamentos nos algoritmos do AI-Benchmark. Primeiramente apresentamos uma descrição dos diferentes mapeamentos utilizados, posteriormente apresentamos sucintamente o AI-Benchmark e, por último, tratamos da arquitetura e do ambiente de execução.

Os seguintes mapeamentos de *threads* foram utilizados:

- **Baseline:** Mapeamento padrão empregado pelo Linux, que foca em balanceamento de carga nos nós disponíveis para execução;
- **Round Robin:** Mapeamento na qual as *threads* são mapeadas de maneira cíclica entre os nós disponíveis para execução;
- **Compact:** Mapeamento na qual *threads* com *ids* próximos são mapeadas a nós próximos, tentando minimizar a distância entre *threads* vizinhas;
- **Scatter:** Mapeamento em que as *threads* são dispostas da maneira mais homogênea possível entre os nós disponíveis para execução. Desta maneira tentando minimizar a distância média entre as *threads*.

Abaixo estão apresentados os diferentes mapeamentos de dados que foram utilizados neste trabalho. O mapeamento de dados é efetuado considerando nós NUMA (do Inglês *Non-uniform memory access*), que aqui são representados pelos processadores presentes nos nós computacionais utilizados.

- **Baseline:** Mapeamento padrão empregado pelo Linux. Emprega uma política *first-touch*, onde o primeiro nó NUMA a acessar uma determinada página de memória passa a aloca-la durante toda a execução;
- **Interleave:** Mapeamento em que páginas consecutivas de memória são distribuídas a nós NUMA consecutivos;
- **NUMA Balancing:** Diferente dos outros mapeamentos, neste as páginas de memória migram durante a execução, sendo alocadas ao último no NUMA que a acessou.

O AI-Benchmark possui 21 algoritmos de aprendizado de máquina, com aplicações em visão computacional, processamento digital de imagens e inteligencia artificial. Cada algoritmo possui duas fases: (i) a de treinamento, onde os modelos de ML

são otimizados pelos respectivos algoritmos e (ii) a de inferência, onde o modelo treinado é utilizado para o seu propósito final (predições, processamento de imagem, etc.). Nesse trabalho foi considerada unicamente a fase de treinamento dos algoritmos do AI-Benchmark, visto que grande parte do custo computacional dos algoritmos de ML vem do treinamento de seus modelos.

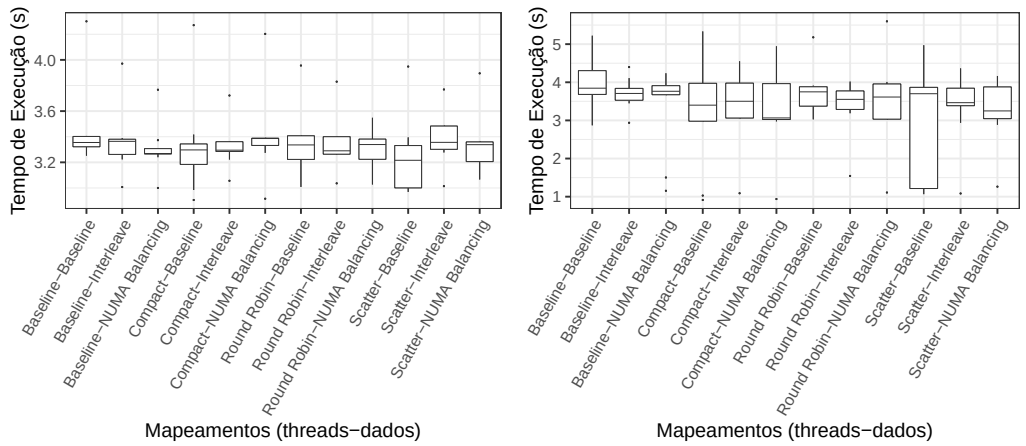
Os experimentos foram particionados em iterações, na qual cada iteração é constituída por diversas execuções do *benchmark*, considerando todas as combinações possíveis de mapeamentos de *threads* e dados supracitados. Ao final de cada iteração os tempos de execução de todos os algoritmos do AI-Benchmark – que são medidos diretamente pelo *benchmark* – são coletados.

A execução dos experimentos foi realizada em um nó computacional de *NUMA factor 2*, contendo dois processadores Intel Xeon E5-2650 v3 (Q3'14) Haswell, 2,3 GHz, totalizando 20 núcleos e 40 *threads* para cada nó.

4. Resultados e Discussão

Nesta seção apresentamos os principais resultados obtidos pelo nosso trabalho. A Figura 1 mostra os resultados dos tempos de execução de dois algoritmos de ML do AI-Benchmark, notadamente o SRCNN 9-5-5 [Dong et al. 2015] e MobileNet-V2 [Sandler et al. 2018]. O eixo x denota um determinado mapeamento de *threads* e dados e o eixo y denota os tempos de execução, em segundos, de 9 iterações (ver Seção 3) dos algoritmos na fase de treinamento. Verificou-se que esses dois algoritmos obtiveram médias de tempo de execução de 4.13% e 15.11% menores, respectivamente, quando comparadas as médias dos tempos de execução sem mapeamento. Além disso, foi observado que o mapeamento de dados NUMA Balancing, quando utilizado isoladamente, resulta em menor variabilidade nos tempos de execução dos dois algoritmos supracitados.

Um caso interessante dos resultados obtidos é o efeito do mapeamento de *threads* Compact, que quando utilizado em conjunto com algum mapeamento de dados na aplicação SRCNN-9-5-5 resulta em menor variabilidade, porém isso não acontece quando



(a) Algoritmo SRCNN-9-5-5 (b) Algoritmo MobileNet-V2

Figura 1: Tempos de execução de dois algoritmos de ML do AI-Benchmark, com diversos mapeamentos de *threads* e dados.

o mesmo mapeamento é utilizado na aplicação MobileNet-V2, reforçando a hipótese de que as características da aplicação exercem influência no resultado da aplicação dos mapeamentos.

5. Conclusão e Trabalhos Futuros

Nesse trabalho foi verificado quais algoritmos de aprendizado de máquina (ML) presentes no *benchmark* AI-Benchmark podem se beneficiar de técnicas de mapeamento de *threads* e dados para melhoria de desempenho. Apresentou-se um primeiro passo no sentido de explorar mapeamento de *threads* e dados para melhorar o desempenho de algoritmos de aprendizado de ML e conseguiu-se efetivamente encontrar alguns algoritmos do AI-Benchmark capazes de se beneficiar de tais mapeamentos. Os próximos passos consistem em descobrir quais são as propriedades de hardware e/ou software que justificariam as características de tempo de execução que encontramos nos experimentos.

Referências

Culkin, R. and Das, S. R. (2017). Machine learning in finance: the case of deep learning for option pricing. *Journal of Investment Management*, 15(4):92–100.

Diener, M., Cruz, E. H., Alves, M. A., Navaux, P. O., and Koren, I. (2016). Affinity-based thread and data mapping in shared memory systems. *ACM Computing Surveys (CSUR)*, 49(4):1–38.

Dong, C., Loy, C. C., He, K., and Tang, X. (2015). Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307.

Ignatov, A., Timofte, R., Kulik, A., Yang, S., Wang, K., Baum, F., Wu, M., Xu, L., and Van Gool, L. (2019). Ai benchmark: All about deep learning on smartphones in 2019. *arXiv preprint arXiv:1910.06663*.

Perols, J. (2011). Financial statement fraud detection: An analysis of statistical and machine learning algorithms. *Auditing: A Journal of Practice & Theory*, 30(2):19–50.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520.

Serpa, M. S., Cruz, E. H., Diener, M., Krause, A. M., Navaux, P. O., Panetta, J., Farrés, A., Rosas, C., and Hanzich, M. (2019). Optimization strategies for geophysics models on manycore systems. *The International Journal of High Performance Computing Applications*, 33(3):473–486.

Serpa, M. S., Krause, A. M., Cruz, E. H., Navaux, P. O. A., Pasin, M., and Felber, P. (2018). Optimizing machine learning algorithms on multi-core and many-core architectures using thread and data mapping. In *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pages 329–333. IEEE.

Stavens, D. M. et al. (2011). *Learning to drive: Perception for autonomous cars*. PhD thesis, Citeseer.