UNIVERSIDADE PRESBITERIANA MACKENZIE

FACULDADE DE COMPUTAÇÃO E INFORMÁTICA

Marcelo Vironda Rozanti

Otimização de cálculos de distância em arquiteturas SIMD

SÃO PAULO
2019

Marcelo Vironda Rozanti

Otimização de cálculos de distância em arquiteturas SIMD

Projeto do Trabalho de Conclusão de Curso apresentado à Faculdade de Computação e Informática da Universidade Presbiteriana, desenvolvida como requisito parcial para aprovação na disciplina de Metodologia da Pesquisa em Computação, ministrada pelo professor Dr. Everton Knihs, do curso de Ciência da Computação.

Orientador: Prof. Dr. Calebe de Paula Bianchini

SÃO PAULO
2019

Otimização de cálculos de distância em arquiteturas SIMD

Marcelo Vironda Rozanti

10 de junho de 2019

Resumo

A combinação de recentes avanços em computadores, o fenômeno Big Data e o uso de algoritmos de Machine Learning, clássicos ou contemporâneos, permitem produzir uma gama de novos dispositivos de impacto significativo na vida humana. Estes algoritmos comumente requerem alto custo computacional e são compostos por sub-algoritmos que realizam operações mais especializadas. Este projeto visa otimizar um pequeno conjunto destes sub-algoritmos destinados ao cálculo de distância euclidiana em diversas arquiteturas com suporte a paralelismo computacional. O código correspondende a este trabalho pode ser acessado no repositório <https://github.com/hpc-fci-mackenzie/edcalc> do Grupo de HPC da Universidade Presbiteriana Mackenzie. O presente trabalho pode ser útil a estudantes, cientistas e profissionais das áreas de Computação de Alta Performance e Inteligência Artificial.

Palavras-chave: SIMD Distância Euclidiana HPC AVX-512 Otimização

Abstract

The combination of recent hardware improvements, the Big Data phenomenon and the usage of classical or contemporary Machine Learning algorithms allow for the development of a range of devices capable of significant human life impact. These algorithms usually require vast computational resources and are composed of sub-algorithms that perform more specialized operations. This project attempts to optimize a small set of these sub-algorithms desgined to perform euclidean distance calculation in a variety of architectures that support parallel computing. The code relating to this project will be available at the HPC Group of Mackenzie Presbyterian Univesrity: <https://github.com/hpc-fci-mackenzie/edcalc>. The present proposal may be of use to students, scientists and professionals in the fields of High Performance Computing and Artificial Intelligence.

Keywords: SIMD Euclidean Distance HPC AVX-512 Optimization

Sumário

Lista de tabelas

Lista de ilustrações

1 INTRODUÇÃO

Este projeto teve como objetivo otimizar algoritmos de cálculo de distância em arquiteturas com suporte a instruções dos conjuntos MMX (*Multimedia Extensions*), SSE (*Streaming SIMD Extensions*) e AVX (*Advanced Vector Extensions*) para a arquitetura de processadores com registradores de 512-bit de largura. Ao utilizar funções nativas do processador alvo, é possível atingir graus maiores de eficiência para uma determinada arquitetura. No entanto, segundo Bramas (2011): "From one CPU generation to the next, improvements and changes are made at various levels. Some modifications are hidden from the programmer and might improve existing codes without any update. This includes the low-level modules (speculation, out-of-order execution, etc.) but also the CPU's clock frequency. On the other hand, some new features and improvements require to be explicitly used". O que é possível interpretar deste trecho é que determinadas modificações não necessitam de atualização no código. No entanto, outras melhorias devem ser necessariamente explicitadas para atingir ganhos mensuráveis.

No caso do uso de supercomputadores, essa explicitação é essencial, tal como esclarece (HOHENAUER et al., 2013, Abstract): "One frequent concern about retargetable compilers, though, is their lack of machine-specific code optimization techniques in order to achieve highest code quality. While this problem is partially inherent to the retargetable compilation approach, it can be circumvented by designing flexible, configurable code optimization techniques that apply to a certain range of target architectures."

1.1 CONTEXTUALIZAÇÃO E RELEVÂNCIA

O âmbito deste trabalho foi o uso de instruções SIMD: múltiplos operandos operados em paralelo em um único item de memória, como notado por Stallings (2010, p. 359). Compiladores atuais não otimizam todos os trechos críticos com o mesmo desempenho que a codificação direta composta por instruções explícitas. No caso de operações vetorizadas, é necessário dar "dicas" ou *hints* ao compilador escolhido para garantir que certas otimizações sejam realizadas, como nos ensinam Jeffers, Reinders e Sodani (2016, p. 192): "Even the most brilliantly conceived compiler cannot auto-vectorize many important loops without hints".

1.2 OBJETO DE PESQUISA

O objeto da pesquisa foi implementações e arquiteturas SIMD, com o suporte da documentação de funções intrínsecas x86 da Intel®, encontradas nos *headers* `immintrin.h` para instruções AVX e `xmmintrin.h` para instruções SSE.

Como é visto em Lomont (2011, p. 8), parte da etimologia das funções a serem empregadas é definida como a seguir:

Tabela 1: Sufixos

Marcadores	**Significado**
`[a/d]`	Ponto flutuante de precisão única ou dupla
`[i/u]nnn`	Inteiro com ou sem sinal de tamanho binário nnn onde nnn é 128, 64, 32, 16, ou 8
`[ps/pd/sd]`	Valor único empacotado, valor duplo empacotado ou escalar duplo
`epi32`	Inteiro assinalado empacotado de 32-bit
`si256`	Inteiro 256-bit escalar

Esta tabela representa parte da lógica empregada na nomenclatura das funções, com o intuito de facilitar a compreensão da implementação e providenciar uma fácil iniciação ao usuário/programador.

1.2.1 PROBLEMA DE PESQUISA

Muitos algoritmos de classificação, no que é pertinente ao campo de Machine Learning, se utilizam de medidas de similaridade tanto para o "treino" destes classificadores, como para o seu uso em produção. Estes algoritmos, por sua vez, requerem alto custo computacional. Pode-se criar uma solução mais eficiente que as atuais em arquiteturas paralelas de 512-bit? Ainda, como adaptar a solução a arquiteturas com maior largura de registradores?

1.2.2 FÓRMULAS UTILIZADAS

Por similaridade, entende-se a distância entre dois pontos em um espaço *n-dimensional*, isto é, vetores de tamanho n. As fórmulas a seguir denotam as medidas de distância utilizadas neste trabalho onde p e q são os pontos cuja distância se quer descobrir:

Distância de Manhattan

$$d_1(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{n} \mid p_i - q_i \mid$$

Distância Euclidiana

$$d_2(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

Distância Cosseno

$$\alpha(\mathbf{p}, \mathbf{q}) = \frac{\sum_{i=1}^{n} p_i q_i}{\sqrt{\sum_{i=1}^{n} p_i^2} \sqrt{\sum_{i=1}^{n} q_i^2}}$$

Visualização:

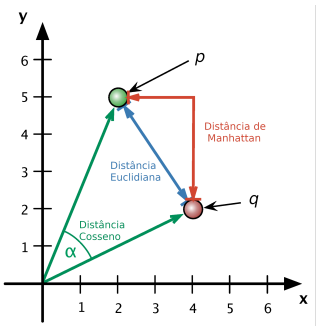

Figura 1: Ilustração das métricas

Estas operações são usadas constante e intensivamente em algoritmos pertinentes ao domínio de Machine Learning como explicitado por Wang e Sun (2012) que ressaltam, ainda, o problema da Maldição da Dimensionalidade, também conhecido pelo anglicismo *Curse of Dimensionality*, sendo possível depreender a importância da eficiência da implementação dos cálculos mencionados.

1.2.3 HIPÓTESE BÁSICA

É possível tirar vantagem da arquitetura de todo processador para resolver um determinado problema computacional, como visto por Tang et al. (2016) e Andre (2016). Testes de performance podem ser criados para verificar a superação de eficiência dos algoritmos em discussão.

1.2.4 VARIÁVEIS

- Arquitetura alvo:
  No contexto desta pesquisa, as implementações foram testadas no processador Xeon Platinum™ com suporte à instruções AVX-512, mas poderiam ser aplicadas em arquiteturas com maior largura de registradores.
- Compilador:
  Os compiladores utilizados foram `icc/icpc` (Intel®) e `gcc/g++` (GNU), em diferentes momentos da pesquisa.
- Entrada ou pontos no espaço:
  Pontos mencionados no item 1.2.2 a serem escolhidos como argumento às funções.

1.3 OBJETIVOS DO ESTUDO

O objetivo desse estudo é ganhar maior compreensão sobre as capacidades e limitações encontradas no design eficiente de implementações em arquiteturas paralelas, bem como ampliar o conhecimento sobre compiladores e tradução código-máquina.

1.3.1 OBJETIVO GERAL

Tornar algoritmos-base de cálculo de distância mais eficientes em determinadas arquiteturas.

1.3.2 OBJETIVOS ESPECÍFICOS

Para alcançar o objetivo geral proposto para a resolução do problema de pesquisa, os seguintes objetivos específicos foram estabelecidos:

- Identificação das teorias envolvidas no estudo de otimização os aspectos que privilegiam o processo de desenvolvimento das otimizações.

- Estudo no âmbito computacional atual dos processos de técnicas e boas práticas de otimização.
- Identificação dos aspectos que qualificam alta-performance que devem ser observados na criação e desenvolvimento destes algoritmos.

1.4 JUSTIFICATIVA

Como já mencionado no item 1.2.1, os algoritmos de classificação podem usar intensamente o cálculo de distância como instrumento intermediário e têm um impacto direto em sua performance. Qualquer ganho de eficiência nestes algoritmos de cálculo de distância em função da arquitetura pode ter uma influência notável na performance geral dos classificadores em fase de treinamento ou em fase prática, ou, de produção, como afirmam Tian et al. (2015) e Martin et al. (2002).

1.5 DELIMITAÇÃO DO ESTUDO

Deve-se restringir a implementação dos algoritmos evidenciados no item 1.2.1, para a arquitetura 512-bit:

- Delimitação Organizacional: família Intel® de processadores
- Delimitação por Indicadores de Desempenho: testes com o intuito de verificar a eficiência do código proposto.

1.6 ORGANIZAÇÃO DO ESTUDO

Descrição dos capítulos:

1. Introdução

   Preâmbulo deste trabalho.

2. Referencial Teórico

   Sustentação argumentativa sobre o tema proposto.

3. Metodologia de Pesquisa

   Sistematização dos instrumentos e processos de estudo empregados no presente trabalho.

4. Cronograma

   Planejamento das tarefas necessárias para a conclusão deste trabalho, bem como suas expectativas de início e conclusão.

5. Descrição da plataforma-alvo.

   Breve narrativa sobre ambiente e ferramentas utilizadas ao longo do estudo.

6. Implementações de cálculo de distância.

   Detalhamento da solução final.

2 REFERENCIAL TEÓRICO

Em relação ao referencial teórico estudado neste TCC, foi de suma importância entender os seguintes tópicos:

2.1 Computação SIMD

O bloco fundamental dos conjuntos de instruções SIMD é composto por operadores booleanos, por exemplo, `AND`, `OR`, `XOR` e suas variantes, bem como deslocamentos lógicos e aritméticos, conversão de tipos e operações de seleção de dados como `max` e `min`, diz Hughes (2015, p. 29).

Extensões de CPU são, muitas vezes, específicas a uma família de processadores e podem ou não existir em mais de um compilador apenas, como é o caso de extensões SIMD, como dizem Franchetti et al. (2005): "The most common language extension supplying such primitives is to provide within the C programming language function-call like intrinsic (or built-in) functions and new data types to mirror the instructions and vector registers. For most SIMD extensions, at least one compiler featuring these language extensions exists".

Isso significa que pode-se usar uma API para acessar estas funções instrínsecas ao processador nativo. A API, por sua vez, é composta por *Programming Language Abstractions*, ainda segundo Franchetti et al. (2005, p. 15): "Recognizing the tedious and difficult nature of assembly coding, most hardware vendors which have introduced multimedia extensions have developed programming-language abstractions. These give an application developer access to the newly introduced hardware without having to actually write assembly language code. Typically, this approach results in a function-call-like abstraction that represents one-to-one mapping between a function call and a multimedia instruction. There are several benefits of this approach".

2.2 Organização e arquitetura de computadores

Arquitetura computacional é um componente chave no desenvolvimento e o programador deve ter um entendimento prático sobre o tópico. “It is concerned with all aspects of the design and organization of the central processing unit and the integration of the CPU into de computer system itself.” (STALLINGS, 2010). Isso mostra a importância da compreensão base de sistemas computacionais para aproveitar o equipamento ao máximo, visando aumento de performance e, consequentemente, uso eficiente de recursos, sejam estes financeiros ou temporais, como em qualquer outro projeto de software de qualidade.

3 METODOLOGIA DA PESQUISA

No que tange à Metodologia empregada neste TCC, o trabalho teve início com uma revisão da literatura específica sobre o tema da pesquisa. Esta pesquisa abrange conceitos fundamentais de computadores de forma genérica (2.2) e específica (2.1), bem como o “estado da arte” em termos de técnicas de otimizacão paralela e *benchmarking* correspondente.

3.1 ETAPAS DA PESQUISA

Para definir as etapas da pesquisa, foi necessário atender às delimitações de estudo (item 1.5), desenvolvendo, para cada tipo de distância especificado no item 1.2.2, suas respectivas implementações para AVX-512 com as APIs NVIDIA® OpenACC e Intel® Intrinsics. Excluindo funções intermediárias, isto é, que funcionam como instrumento interno, soma-se 6 funções implementadas, como descrito no Cronograma apresentado no item 4.

Assim, pode-se dizer que as etapas desenvolvidas neste estudo foram:

1. Estudo da Documentação;
2. Implementação da Distância Manhattan em Intel Intrinsics;
3. Implementação da Distância Euclidiana em Intel Intrinsics;
4. Implementação da Distância Cosseno em Intel Intrinsics;
5. Implementação da Distância Manhattan em OpenACC;
6. Implementação da Distância Euclidiana em OpenACC;
7. Implementação da Distância Cosseno em OpenACC;
8. Análise e testes das implementações;
9. Artigo.

3.1.1 CONCEITOS EMPREGADOS

- Compilador:

  Programa capaz de traduzir instruções de uma linguagem de programação para outra.

- Entrada:

  Pontos num espaço *n-dimensional*.

- Throughput:

  Número médio de tarefas completadas em uma determinada unidade temporal (STALLINGS, 2010, p. 299).

- Dados empacotados (STALLINGS, 2010, p. 379):

  Packed byte: oito bytes empacotados em uma quantidade 64-bit.

  Packed word: quatro palavras de 16-bit empacotadas em 64 bits.

  Packed doubleword: duas palavras duplas de 32-bit empacotadas em 64 bits.

- Extensões de CPU, melhor descritas no item 2.1

  MMX: Multimedia Extensions.

  SSE: Streaming SIMD Extensions.

  AVX: Advanced Vector Extensions.

- API:

  Application Programming Interface.

3.2 CLASSIFICAÇÃO DA PESQUISA

O tempo total previsto para a conclusão desta pesquisa é de 1 ano, como mostrado no capítulo 4.

3.2.1 NATUREZA DA PESQUISA

Esta é uma pesquisa Aplicada, já que sua aplicação prática é imediata.

3.2.2 ABORDAGEM

Esta pesquisa é baseada em cálculos, medidas objetivas e dados verificáveis, através de testes automatizados.

3.2.3 FINS

Esta pesquisa foi voltada para encontrar caminhos, formas, maneiras e procedimentos para atingir um determinado fim, buscando definir um processo ou uma ferramenta que leve à solução do problema proposto (1.2.1).

3.2.4 PESQUISA PROPOSITIVA

Código-fonte, bem como suas instruções de compilação e documentação interna e externa utilizada para gerar o binário que ultrapasse soluções atuais para cálculo de distância nas arquiteturas alvo.

3.2.5 MEIOS

Quanto aos meios, foram utilizados os recursos mencionados na Bibliografia e Dados documentais (item 3.2.7). Testes automatizados servirão para medir *speedup* em relação a outros algoritmos conhecidos.

3.2.6 PESQUISA BIBLIOGRÁFICA

Para o andamento desta pesquisa foi necessário compreender a taxonomia de Flynn e programação paralela discorrida por Stallings (2010), Hennessy e Patterson (2011).

3.2.7 PESQUISA DOCUMENTAL

Documentação de funções da família de instruções x86 Intel® Intrinsics extraído de Intel (2019) e da API OpenACC da NVIDIA (2019).

4 CRONOGRAMA

As atividades desta pesquisa se desenvolveram de acordo com o cronograma apresentado a seguir, no prazo de 12 meses:

Tabela 2: Cronograma de atividades

ATIVIDADE	MÊS											
	1	2	3	4	5	6	7	8	9	10	11	12
Estudo da Documentação												
Implementação da Distância de Manhattan (Intel® Intrinsics)												
Implementação da Distância de Euclidiana (Intel® Intrinsics)												
Implementação da Distância de Cosseno (Intel® Intrinsics)												
Implementação da Distância de Manhattan (OpenACC)												
Implementação da Distância de Euclidiana (OpenACC)												
Implementação da Distância de Cosseno (OpenACC)												
Artigo												

Referências

ANDRE, F. *Exploiting Modern Hardware for High-Dimensional Nearest Neightbor Search*. 2016. Acesso em 3 Mar. 2019. Disponível em: <https://doi.org/10.1007/s10044-007-0065-y>.

BRAMAS, B. *Fast Sorting Algorithms using AVX-512 on Intel Knights Landing*. 2011. Acesso em 3 Mar. 2019. Disponível em: <https://hal.inria.fr/hal-01512970v1>.

FRANCHETTI, F. et al. Efficient utilization of simd extensions. *Proceedings of the IEEE*, IEEE, v. 93, n. 2, p. 409–425, 2005.

HENNESSY, J.; PATTERSON, D. *Computer Architecture: A Quantitative Approach*. 5. ed. [S.l.: s.n.], 2011. ISBN 0123838738, 9780123838735.

HOHENAUER, M. et al. Retargetable code optimization with simd instructions. 2013. Acesso em 3 Mar. 2019. Disponível em: <https://books.google.com.br/books?isbn=1447138627>.

HUGHES, C. *Single-Instruction Multiple-Data Execution*. Morgan & Claypool Publishers, 2015. (Synthesis Lectures on Computer Architecture). ISBN 9781627057646. Disponível em: <https://books.google.com.br/books?id=xz3GCQAAQBAJ>.

INTEL. *Intel Intrinsics Guide*. 2019. Acesso em 3 Mar. 2019. Disponível em: <https://software.intel.com/sites/landingpage/IntrinsicsGuide/>.

JEFFERS, J.; REINDERS, J.; SODANI, A. *Intel Xeon Phi processor high performance programming: Knights Landing Edition*. 2. ed. Cambridge, MA: Elsevier Science, 2016. Disponível em: <http://cds.cern.ch/record/2222648>.

LOMONT, C. *Introduction to Intel® Advanced Vector Extensions*. 2011. Disponível em: <https://computing.llnl.gov/tutorials/linux_clusters/Intro_to_Intel_AVX.pdf>.

MARTIN, B. et al. *Optimization of Euclidean Distance Transforms for Pattern Recognition*. 28660 Boadilla (Madrid), Espanha, 2002. Acesso em 3 Mar. 2019. Disponível em: <http://www.enriquefrias-martinez.info/yahoo_site_admin/assets/docs/364-1221.34740646.pdf>.

NVIDIA. *OpenACC-API*. 2019. Acesso em 3 Mar. 2019. Disponível em: <https://www.openacc.org/sites/default/files/inline-files/OpenACC.2.7.pdf>.

SNYDER, L. *A Taxonomy of Synchronous Parallel Machines*. 1988. Disponível em: <https://apps.dtic.mil/dtic/tr/fulltext/u2/a204256.pdf>.

STALLINGS, W. *COMPUTER ORGANIZATION AND ARCHITECTURE DESIGNING FOR PERFORMANCE*: Deisgning for performance. 8. ed. [S.l.]: PEARSON, 2010.

TANG, B. et al. *Exploit Every Cycle: Vectorized Time Series Algorithms on Modern Commodity CPUs*. 2016. Acesso em 3 Mar. 2019. Disponível em: <http://www.adms-conf.org/2016/ADMS_simd_paper3.pdf>.

TIAN, X. et al. *Effective SIMD Vectorization for Intel Xeon Phi Coprocessors*. 2015. Acesso em 3 Mar. 2019. Disponível em: <downloads.hindawi.com/journals/sp/2015/269764.pdf>.

WANG, F.; SUN, J. Distance metric learning in data mining (part i). 2012.