Compas'2023 : Parallélisme / Architecture/ Système
LISTIC/USMB - Annecy France, 4-7 juillet 2023

Portable vectorization and precision tuning through a search space exploration and machine learning

Lana Scravaglieri

Résumé
Scientific applications with large workloads heavily rely on high-performance computing and hardware evolution to solve new more complex problems. To improve the performance of processors, manufacturers are strongly invested in designing efficient vector units and associated SIMD extensions. However, SIMD instruction sets are difficult to use and not portable across architectures. Deploying applications that use SIMD parallelism on various processors requires a more portable and robust SIMD programming model.
Multiple tools have already been developed with two approaches [7], either explicit with intrinsics wrapper libraries [5, 2, 1] or implicit with hints for the compiler to improve auto-vectorization [6, 9, 4]. These tools have a common limitation : they only focus on generating semantically equivalent instructions for different architectures without taking advantage of their specificities. Wrappers use one-to-one associations to generate hardware-specific instructions, thus missing optimization opportunities. Indeed, small kernels or sequences of intrinsics may benefit from different ordering depending on the target architecture. The use of a vectorizing compiler makes code more maintainable and more portable than intrinsics because architecture specific instructions are generated at compile time. However, the use of vectorization-friendly data layout is critical for efficient auto-vectorization [9, 3] but is often neglected for readability. Finally, vectorization and arithmetic precision are also attractive to study together as they both affect the balance between loading, storing, casting, and computing.
We propose to study the impact of multiple parameters on vectorization performance : store precision, compute precision, data layout, prefetch and frequency. The goal is to predict the best value of these parameters for a new code with a target architecture. To explore this optimization space, we will evaluate multiple benchmarks on all sensible combinations of parameters and identify classes where codes have similar performance improvements with similar parameter values. Then, we will use supervised learning classification algorithms to predict the category of a code [8]. Additional interesting parameters to study are the vector length and stride as they are tightly related to the data layout. We could then compare the performance of a manually vectorized code (using a wrapper library) against automatically optimized code using these hints tuned with the rest of the parameters.

Mots-clés : vectorisation, SIMD, précision, disposition des données

---

Bibliographie

1. Cassagne (A.). – *Optimization and parallelization methods for software-defined radio*. – Theses, Université de Bordeaux, décembre 2020.

2. Estérie (P.), Falcou (J.), Gaunard (M.) et Lapresté (J.-T.). – Boost.simd : Generic programming for portable simdization. – In *Proceedings of the 2014 Workshop on Programming Models for SIMD/Vector Processing*, *WPMVP '14*, WPMVP '14, p. 1–8, New York, NY, USA, 2014. Association for Computing Machinery.
3. Gruber (B. M.), Amadio (G.), Blomer (J.), Matthes (A.), Widera (R.) et Bussmann (M.). – LLAMA : the low level abstraction for memory access. *CoRR*, vol. abs/2106.04284, 2021.
4. Haj-Ali (A.), Ahmed (N. K.), Willke (T.), Shao (Y. S.), Asanovic (K.) et Stoica (I.). – Neurovectorizer : End-to-end vectorization with deep reinforcement learning. – In *Proceedings of the 18th ACM/IEEE International Symposium on Code Generation and Optimization*, *CGO 2020*, CGO 2020, p. 242–255, New York, NY, USA, 2020. Association for Computing Machinery.
5. Kretz (M.) et Lindenstruth (V.). – Vc : A c++ library for explicit vectorization. *Software : Practice and Experience*, vol. 42, n11, 2012, pp. 1409–1430.
6. OpenMP Architecture Review Board. – OpenMP application program interface version 5.1, may 2020.
7. Pohl (A.), Cosenza (B.), Mesa (M. A.), Chi (C. C.) et Juurlink (B.). – An evaluation of current simd programming models for c++. – In *Proceedings of the 3rd Workshop on Programming Models for SIMD/Vector Processing*, *WPMVP '16*, WPMVP '16, New York, NY, USA, 2016. Association for Computing Machinery.
8. Scravaglieri (L.), Popov (M.), Lima Pilla (L.), Guermouche (A.), Aumage (O.) et Saillard (E.). – Optimizing performance and energy across problem sizes through a search space exploration and machine learning. *Available at SSRN 4396668*, 2022.
9. Tian (X.), Saito (H.), Girkar (M.), Preis (S. V.), Kozhukhov (S. S.), Cherkasov (A. G.), Nelson (C.), Panchenko (N.) et Geva (R.). – Compiling c/c++ simd extensions for function and loop vectorizaion on multicore-simd processors. – In *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*, pp. 2349–2358, 2012.