SIMD computing for multivariate polynomials

Supervisors:

François Boulier	Francois.Boulier@univ-lille1.fr	http://cristal.univ-lille.fr/~boulier
Pierre Fortin	Pierre.Fortin@univ-lille1.fr	http://lip6.fr/Pierre.Fortin
François Lemaire	Francois.Lemaire@univ-lille1.fr	http://cristal.univ-lille.fr/~lemaire

Hosting laboratory:
CRIStAL (http://www.cristal.univ-lille.fr/CFHP/), Université de Lille

Hosting team:
CFHP team (*Calcul Formel et Haute Performance* - Computer Algebra and High Performance Computing, http://www.cristal.univ-lille.fr/CFHP/)

Duration: 5 or 6 months, starting from February or March 2018

Funding: standard pay of Université de Lille

Keywords: SIMD, multivariate polynomial, modular arithmetic, OpenMP, SPMD-on-SIMD

Scientific context. Nowadays SIMD (*single instruction, multiple data*) parallelism is present on all current HPC (*High Performance Computing*) architectures (multi-core CPUs, GPUs ...). Morever its share in the overall compute power of multi-core CPUs is constantly increasing: from former 128-bit SSE or AltiVec, to current 256-bit AVX[2] and now to the new 512-bit AVX-512 from 2017. Therefore one must now investigate such parallelism in order to exploit at best HPC architectures: not exploiting SIMD can indeed limit the sustained performance down to 1/16 of the CPU peak performance.
In computer algebra, numerous high level algorithms rely on low-level operations on polynomials using modular arithmetic (i.e. integer computations with modulus). Since these polynomials may not be large enough to fully exploit the multiple cores and caches of modern high-end CPUs, SIMD parallelism on one single core is a primary target for such computations [1].

Objectives. This internship aims at deploying elementary polynomial operations (mainly evaluations and multiplications) on the SIMD units of modern CPUs. As far as SIMD hardware is concerned, we will focus on AVX2 and AVX-512 which are the most recent SIMD instructions.
We will investigate here high level SIMD programming in order to (try to) ensure both portability and high performance. We will hence consider here two SIMD parallel paradigms.

- The first one is based on OpenMP programming which relies on compiler directives (pragmas) to indicate which loops the compiler will have to vectorize, and to guide the vectorization process. This simple approach can however be limited by the directive expressiveness and by the compiler capabilities.

- The second paradigm is based on the SPMD-on-SIMD (*single program, multiple data*) model, where all computations are written as scalar ones and it is up to the compiler to merge such scalar computations in SIMD instructions. Compared to OpenMP, more control is given to the programmer who may be able to hence obtain more efficient SIMD code [2, 3]. Compared to lower-level SIMD programming paradigms (intrinsics, assembly), the programmer needs neither to write the specific SIMD intrinsics for each architecture, nor to know the vector width, nor to implement data padding with zeroes according to this vector width. This paradigm is available in OpenCL (OpenCL implicit vectorization) and in the Intel SPMD Program Compiler[1] (`ispc`) [4]: we plan here to rely on `ispc`.

We thus plan to compare these two approaches, while adapting the algorithms in order to obtain the best SIMD performance. We will focus here on multivariate polynomials with low degrees, which are used by the CFHP team (in differential algebra mainly) and thus consider the relevant algorithms in this context. Special care will have to be taken with divergence issues due to modular arithmetic and to the fixed SIMD width.

The larger polynomials could also benefit from multi-core parallelization: we will rely here on OpenMP task parallelism which will nicely fit the higher level parallelization of our algorithms. In practice, we will rely on two HPC servers: a first one with two 18-core CPUs (two Intel Xeon E5-2695 v4 CPUs, with 72 hardware threads and AVX2 SIMD units), and a second one based on the new Intel Xeon Phi architecture (Knights Landing 7290 with 72 CPU cores (288 hardware threads) and with the newest AVX-512 SIMD units).

This work will enable to substantially accelerate numerous high-level computer algebra applications (GCD computation, differential algebra ...). Depending on the internship progress, such higher level algorithm (like simple GCD algorithm) could also be investigated for efficient SIMD processing.

Required skills: Interest in scientific and high performance computing, strong motivation and reasoning are mainly required, as well as C programming. No skill in computer algebra is required. HPC skills are welcome but not mandatory.

This internship may be followed by a Ph.D. thesis in the same research area.

References:

[1] J. van der Hoeven, G. Lecerf, and G. Quintin. *Modular SIMD arithmetic in MATHEMAGIX*. ACM Trans. Math. Softw. 43, 1, Article 5, 2016.

[2] P. Eberhart, J. Brajard, P. Fortin and F. Jézéquel. High Performance Numerical Validation using Stochastic Arithmetic. Reliable Computing, Vol. 21, pp 35-52, 2015.

[3] B. Lange and P. Fortin. Parallel dual tree traversal on multi-core and many-core architectures for astrophysical N-body simulations. Euro-Pare 2014.

[4] M. Pharr and W.R. Mark. ispc: A SPMD compiler for high-performance CPU programming. Innovative Parallel Computing (InPar), pp. 1-13, 2012.

[1] See: `https://ispc.github.io/`