APPLICATION OF GPGPUs AND MULTICORE CPUs IN OPTIMIZATION OF SOME OF THE MPDROOT CODES*

A. Fatkina#, O. Iakushkin , N. Tikhonov, St. Petersburg State University, St. Petersburg, Russia

Abstract

We analyzed the ways to optimize MPDRoot algorithms using existing solutions from external libraries. We also examined the libraries designed to work with graphics accelerators and multi-core CPUs, such as cuRAND, cuFFT and OpenCL FFT.

The paper describes the ways to expedite a portion of Kalman filter by transferring it to GPUs or multi-core CPUs using the implementation included into the MPDRoot package.

INTRODUCTION

MPD (Multi Purpose Detector) is a part of NICA (Nuclotron-based Ion Collider fAcility) [1]. MPDRoot is a framework based on ROOT and FairRoot technologies. It is designed to simulate experiments conducted on MPD and to analyze the resulting data. According to the MPD documentation, collected data can be huge. Fast data processing is necessary to cope with extreme data sets. Thus it is a task of crucial importance that algorithms work well in parallel and distributed environments [2-6]

In this paper, we look into how Graphic Processing Units (GPUs) and multicore CPUs may be applied in MPDRoot project optimization.

PERFORMANCE ANALYSIS OF MPDROOT FRAMEWORK TESTS

The following methods were used to analyze MPDRoot framework:

- Doxygen utility was used to generate classes and functions dependency graphs. It was also used to code navigation.
- ValgrindCallgrind tool was used to profile the package. By using this we got callgraphs for MPDRoot tests.

We selected some algorithms that can be ported on coprocessor architectures and seems to be optimizable.

PROPOSED OPTIMIZATIONS

The following algorithms were considered:

- Fast Fourier Transform;
- Random number generation;
- Kalman Filter.

Figure 1 shows calls of the Rndm() function in the ROOT framework. It is called over 32 million times in the runMC test. The function employs MT (Mersenne Twister) algorithm described in [7].

Random Number Generation

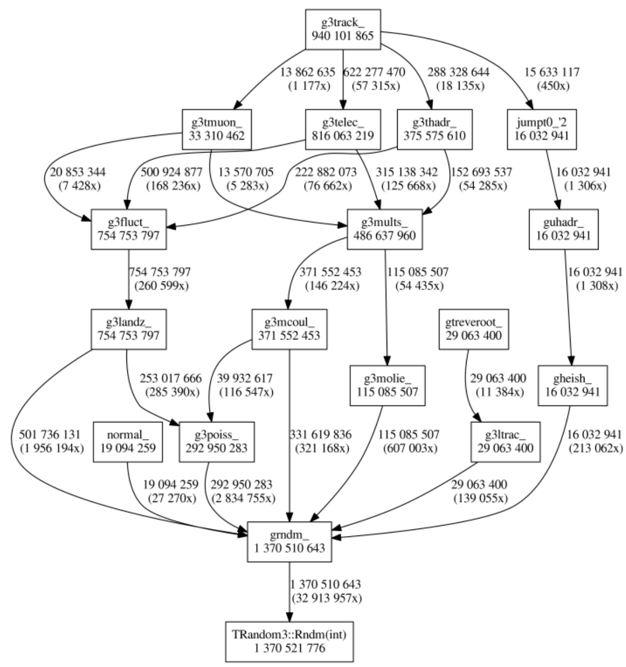

Figure 1: Profiling data - Random number generation.

There is a modified version of this algorithm — SFMT (SIMD-oriented Fast Mersenne Twister) [8], which is twice as fast owing to the SIMD (Single Instruction Multiple Data) principle. It should be noted that the version of MT described in that paper can only be executed on a CPU with a vector processing unit (VPU).

CuRAND library may be used as an alternative to ROOT-based MT. It makes it possible to generate random numbers on GPGPU (general-purpose computing for graphics processing units) using CUDA architecture. The library provides a wide range of generators including MT and MTGP (Mersenne Twister for Graphic Processors) [9].

The use of the cuRAND library to generate random numbers in the GEANT4 framework was proposed at the Annual Science Meeting in 2013 [10]. This approach can either be integrated into the MPD Root project directly with the cuRAND library or indirectly by using GEANT4.

Kalman Filter

MPD Root reconstructs the particles' tracks using Kalman filter. This algorithm is sequential and uses matrixes.

* supported by SPbU (Saint Petersburg State University) grants 9.37.157.2014, 0.37.155.2014 and Russian Foundation for Basic Research grant (project no. 16-07-01113 and no. 16-07-01111).
# st035020@student.spbu.ru

ISBN 978-3-95450-181-6

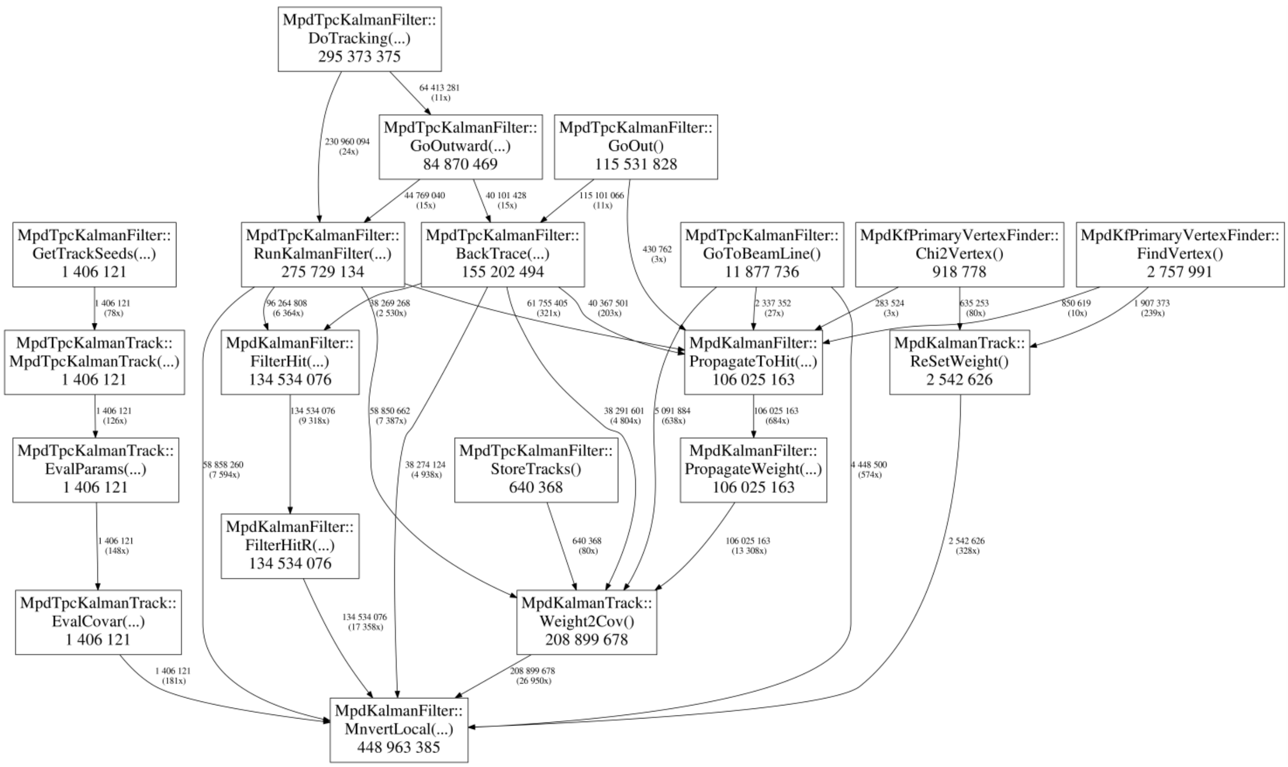

Figure 2: Profiling data - Kalman Filter: MnvertLocal function.

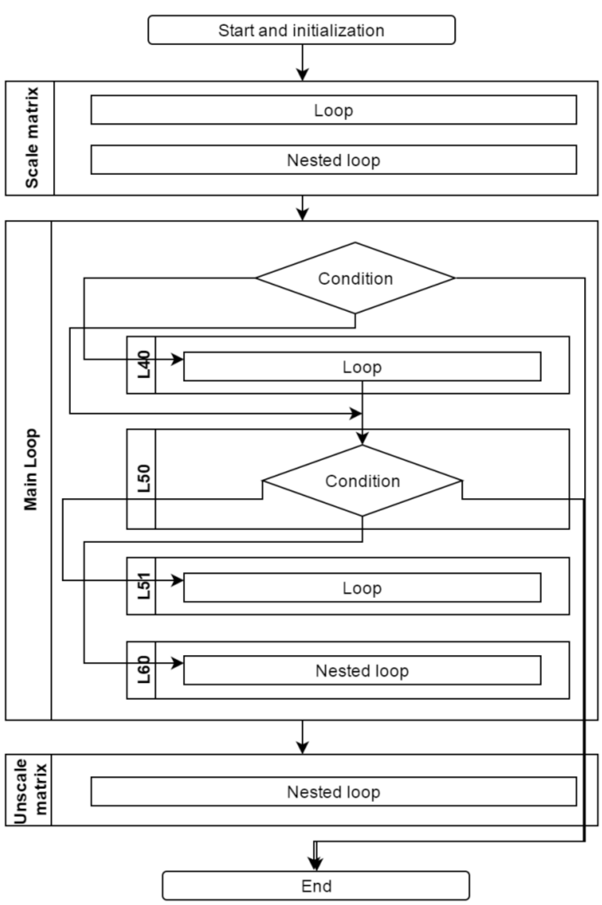

Figure 3: MnvertLocal function scheme.

Figure 2 shows the call graph of MpdKalmanFilter::MnvertLocal() function, which implements the Kalman filter algorithm. It utilizes nested loops to invert matrixes. Figure 3 shows a schematic representation of MnvertLocal function algorithm. Each loop executes similar instructions independent of the values obtained during the previous iterations. They can, therefore, be distributed to multiple cores of the CPU and vectorized, or transferred to the GPGPU.

An alternative way to expedite Kalman filter track reconstruction algorithm is described in [11]. In the paper, the authors proposed using SIMD instructions to optimize the Kalman filter in the CBM@FAIR (Compressed Baryonic Metter) experiment. They tested the algorithm before and after optimization and saw a major improvement: the authors managed to increase the algorithm execution speed by a factor of 120.

Fast Fourier Transform

FFTW library is used in the digitizing algorithm of the TPC detector in TVirtualFFT class, a Fourier transformation shell in the ROOT framework. It provides interfaces to work with OpenMP and MPI technologies. However, it does not support graphic co-processors. There are multi-core-oriented libraries, such as cuFFT and clFFT, which implement algorithms included into FFTW.

However, the use of multiple cores does not necessarily influence the execution time of code segments. In some cases, it can even increase their total processing time. For

ISBN 978-3-95450-181-6

instance, according to the available tests, Fourier transform algorithm represents a minor part of MPDRoot project compared to the other package components (less than 0.01%). This allows to conclude that running these code segments on multiple cores will not reduce the test processing time.

CONCLUSION

In this paper, we consider some of the possible optimizations for MPDRoot project with coprocessor technologies. We proposed some recommendations based on both external libraries and algorithms based on MPDRoot codes.

ACKNOWLEDGMENT

This research was partially supported by SPbU (Saint Petersburg State University) grants 9.37.157.2014, 0.37.155.2014 and Russian Foundation for Basic Research grant (project no. 16-07-01113 and no. 16-07-01111).

REFERENCES

[1] MPD Collaboration, MPD Concept Design Repot; http://mpd.jinr.ru

[2] A.Zinchenko, Y.Murin, V.Kondrat'ev, N.Prokof'ev, "Modeling of the internal tracking system of the NICA/MPD detector." Physics of Particles and Nuclei Letters, 13(4), pp.483-491 (2016).

[3] O. Iakushkin, "Intellectual scaling in a distributed cloud application architecture: A message classification algorithm", Stability and Control Processes in Memory of VI Zubov (SCP), 2015 International Conference, IEEE, p. 634-637. (2015).

[4] S. Abrahamyan, S. Balyan, A. Muradov, V. Korkhov, A. Moskvicheva, and O. Iakushkin. "Development of M-Health Software for People with Disabilities." In International Conference on Computational Science and Its Applications, Springer International Publishing, p. 468-479(2016).

[5] O. Iakushkin, O. Sedova, G. Valery, "Application Control and Horizontal Scaling in Modern Cloud Middleware.", Transactions on Computational Science XXVII, Springer Berlin Heidelberg, p. 81-96.(2016).

[6] Y. Shichkina, A. Degtyarev, D. Gushchanskiy, and O. Iakushkin. "Application of Optimization of Parallel Algorithms to Queries in Relational Databases.", International Conference on Computational Science and Its Applications, Springer International Publishing, p. 366-378 (2016).

[7] M. Matsumoto, N. Takuji, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator", ACM Transactions on Modeling and Computer Simulation, Vol. 8 No. 1 p. 3-30 (1998).

[8] M. Saito, M. Matsumoto, "SIMD-oriented Fast Mersenne Twister: a 128-bit Pseudorandom Number Generator", Monte Carlo and Quasi-Monte Carlo Methods 2006, p. 607 — 622 (2008).

[9] M. Saito, M. Matsumoto, "Variants of Mersenne Twister Suitable for Graphic Processors" ACM Transactions on Mathematical Software, Vol. 39 No. 2 (2013).

[10] Soon Yung Jun, J. Apostolakis, "GPUs in GEANT4", Annual Concurrency Meeting, February 2013.

[11] S. Gorbunov, U.Kebschull, I. Kisel, W. F. J. Müller, V. Lindenstruth, "Fast SIMDized Kalman filter based track fit", Computer Physics Communications, Vol. 178, No. 5, p. 374-383 (2008).

ISBN 978-3-95450-181-6