Status of the BmnRoot optimization

S.NEMNYUGIN

SAINT-PETERSBURG STATE UNIVERSITY

Summary of previous BmnRoot optimizations

- Implementation of OpenMP multithread parallelization in some simulation modules of BmnRoot.
- PROOF (Parallel ROOT Facility) integration into the event reconstruction part of the BmnRoot framework.
- Geant4 multithreading in simulation part of the BmnRoot.
- Vectorization of the ADC Strip Decoder module with vector intrinsics.
- Comparative study of various compilers (GCC vs Intel).

Current performance and optimization issues

Timing for two tracking methods

- L1 (CellAuto) tracking ~0.5 sec / event.
- Vector Finder (VF) ~2.8 sec / event.

Test bench

ACER Nitro 5 AN515-52-75S2,CPU Intel Core i7 8750H (6 cores, 2x Hyperthreading, AVX2 vector extension), 32 Gb RAM.

Focus of optimization

- BmnKalmanFilter.cxx / BmnKalmanFilter.h
- BmnFieldMap.cxx / BmnFieldMap.h
- BmnNewFieldMap.cxx / BmnNewFieldMap.h

Optimization methods under consideration

1. “Small” code improvements.
2. Vectorization of Kalman Filter and Field Map modules by vector intrinsics.
3. Evaluation of performance efficiency of computations offload on hybrid architectures.
4. Algorithmic optimizations.

“Small” code improvements

```
#include <vector>
…
vector<Double_t> xIn;
xIn[0] = par->GetX();
xIn[1] = par->GetY();
xIn[2] = par->GetTx();
xIn[3] = par->GetTy();
xIn[4] = par->GetQp();
vector<Double_t> xOut(5, 0.);
vector<Double_t> F1(25, 0.);
…
```

“Vectors are sequence containers representing arrays that can change in size.”

But convenience of manipulating with dynamic arrays must be paid. And cost is performance!

What may be done:

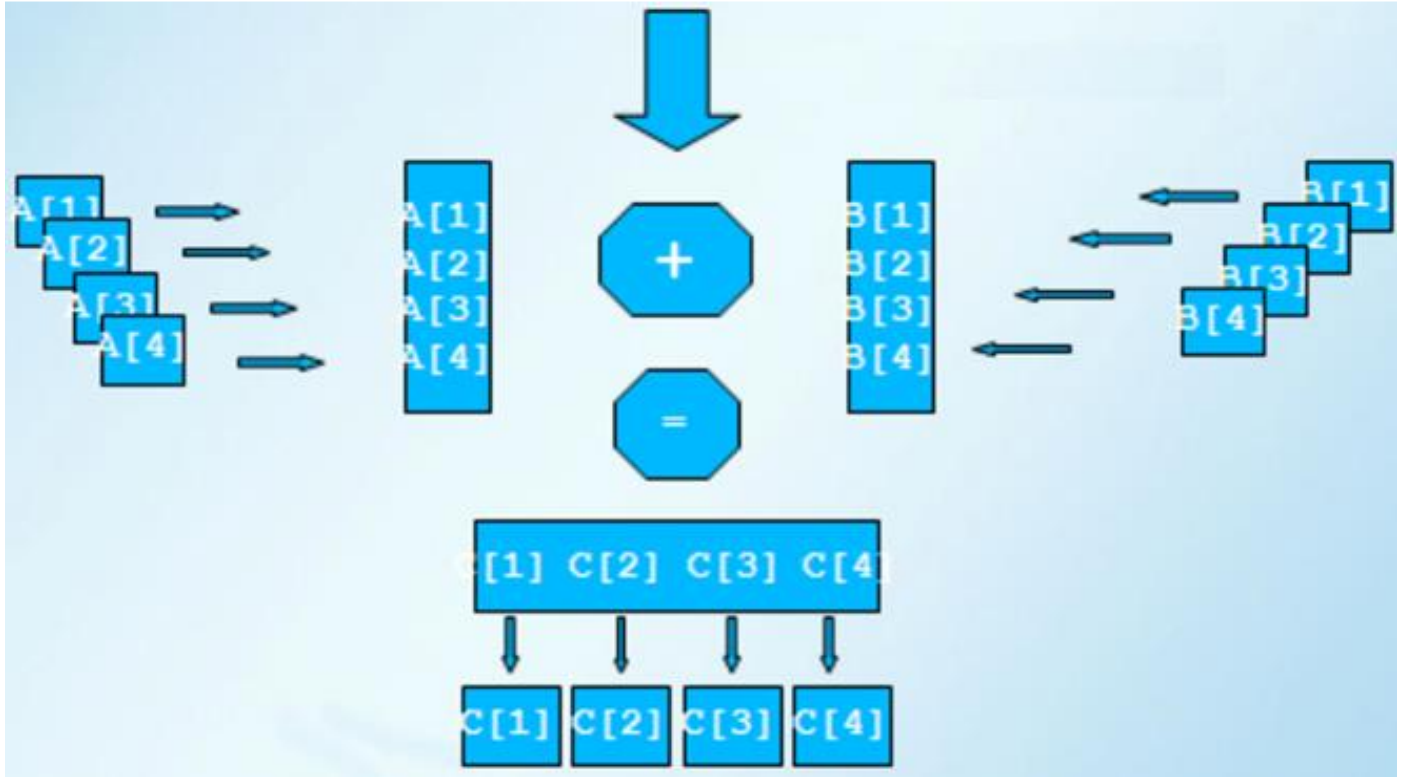
1. Go back to conventional static arrays if possible.
2. If using vector templates:
   - use more efficient methods for elements substitution (less memory transactions);
   - *a priory* reservation of estimated number of elements.

```
vector<Double_t> prevPredX;
prevPredX.push_back(prevNode->GetPredictedParam()->GetX());
…
```

```
vector<Double_t> prevPredX;
prevPredX.reserve(5);
prevPredX.emplace_back(prevNode->GetPredictedParam()->GetX());
…
```

Estimated improvement in time ~10 % but it may be more significant

Vectorization of Kalman Filter and Field Map modules by vector intrinsics



The data is packed into vectors, which are then processed in parallel => Loops iterations are reduced.

Vectorization for conventional arrays is partly implemented. Work is in progress.

Vectorization by `xmmintrin`-vector intrinsics for `vector` templates is more laborious. Work is in progress.

Intrinsics

```
…
#pragma GCC target("avx2")
#pragma GCC optimize("O3")
#include <x86intrin.h>
…
__m256d  s;
__m256d *cInxx, *cIn_tmpxx;
cInxx = (__m256d*) cIn;
cIn_tmpxx0 = (__m256d*) cIn_tmp;
…
s = _mm256_broadcast_pd(0);
…
```

SIMD extensions are assembly functions, and programming languages with any higher level of abstraction cannot process them directly.

However, there are built-in wrappers for their use, which are called *intrinsics*.

Algorithmic optimizations. Field map.

Hotspots

Analysis Configuration | Collection Log | Summary | Bottom-up | Caller/Callee | Top-down Tree | Flame Graph | Platform

Grouping: Function / Call Stack

Function / Call Stack	CPU Time ▼	Module	Function (Full)
BmnNewFieldMap::FieldInterpolate	22.106s	libBmnField.so.0.0.0	BmnNewFieldMap::FieldInterpolate(TArrayF*, double
TArrayF::At	20.888s	libBmnField.so.0.0.0	TArrayF::At(int) const
TArray::BoundsOk	17.334s	libBmnBase.so.0.0.0	TArray::BoundsOk(char const*, int) const
CbmKFMath::multQtSQ	17.272s	libKF.so.0.0.0	CbmKFMath::multQtSQ(int, double const*, double cc
BmnFieldMap::Interpolate	16.762s	libBmnField.so.0.0.0	BmnFieldMap::Interpolate(double, double, double)
BmnNewFieldMap::IsInside	15.514s	libBmnField.so.0.0.0	BmnNewFieldMap::IsInside(double, double, double, ir
std::unordered_map<int, BmnStsVectorFinder::hit	13.312s	libBmnTracking.so.0.0.0	std::unordered_map<int, BmnStsVectorFinder::hitinfc
std::set<int, std::less<int>, std::allocator<int>>::in	13.156s	libMpdGen.so.0.0.0	std::set<int, std::less<int>, std::allocator<int>>::insert
CbmKFFieldMath::ExtrapolateRK4	9.484s	libKF.so.0.0.0	CbmKFFieldMath::ExtrapolateRK4(double const*, do
operator new	6.330s	libstdc++.so.6	operator new(unsigned long)

Too much addresses to the Field Map?

574			
575	Double_t BmnFieldMap::Interpolate(Double_t dx, Double_t dy, Double_t dz) {	0.2%	0.656s
576	// Interpolate in x coordinate		
577	fHb[0][0] = fHa[0][0][0] + (fHa[1][0][0] - fHa[0][0][0]) * dx;	0.5%	1.392s
578	fHb[1][0] = fHa[0][1][0] + (fHa[1][1][0] - fHa[0][1][0]) * dx;	0.4%	1.112s
579	fHb[0][1] = fHa[0][0][1] + (fHa[1][0][1] - fHa[0][0][1]) * dx;	0.4%	1.140s
580	fHb[1][1] = fHa[0][1][1] + (fHa[1][1][1] - fHa[0][1][1]) * dx;	0.4%	1.258s
581			
582	// Interpolate in y coordinate		
583	fHc[0] = fHb[0][0] + (fHb[1][0] - fHb[0][0]) * dy;	0.9%	2.672s
584	fHc[1] = fHb[0][1] + (fHb[1][1] - fHb[0][1]) * dy;	0.7%	2.228s
585			
586	// Interpolate in z coordinate		
587	return fHc[0] + (fHc[1] - fHc[0]) * dz;	1.8%	5.296s

List of first hotspots and source code of one of most important hotspot from dynamic analysis

Intrinsics

```
…
#pragma GCC target("avx2")
#pragma GCC optimize("O3")
#include <x86intrin.h>
…
__m256d  s;
__m256d *cInxx, *cIn_tmpxx;
cInxx = (__m256d*) cIn;
cIn_tmpxx0 = (__m256d*) cIn_tmp;
…
s = _mm256_broadcast_pd(0);
…
```

SIMD extensions are assembly functions, and programming languages with any higher level of abstraction cannot process them directly.

However, there are built-in wrappers for their use, which are called *intrinsics*.

Tuning of the BmnRoot for hybrid architectures

Dilemma - choice of the programming technology – something new (Intel OneAPI Data Parallel C++ etc.) or traditional (CUDA or OpenCL)?

CUDA – hybrid architectures with General Purpose GPU. Implementation of CUDA into simulation module of the BmnRoot is in progress and its efficiency is under evaluation.

Summary

- Intel® VTune™ Profiler was used to analyze the code of the BmnRoot software package.
- “Small” code improvements are considered.
- Vectorization of the tracking is in progress.
- Hybridization of the BmnRoot is under evaluation.
- Need for the Field Map usage in the BmnRoot become more and more obvious.

Thank you for attention