Current status of the BmnRoot performance optimization

S.NEMNYUGIN

SAINT-PETERSBURG STATE UNIVERSITY

Outline

- Previous results
- Decoder optimization
- Summary

Work is supported by Pilot JINR Programme..

Previous results of optimization

Levels of optimization:

1. External optimization on the base of distributed computing (data parallelism). Multithreaded **Geant4**. **PROOF**.
2. Internal optimization – removing of «hotspots». Vectorization (compiler vectorization, intrinsics, libraries), parallelization.

ROOT Data Analysis Framework

PROOF (Parallel ROOT Facility)

Part of **ROOT** (CERN & MIT).

The **PROOF** system allows:

- parallel analysis of trees in a set of files;
- parallel analysis of objects in a set of files;
- parallel execution of scripts

on parallel computing systems.

Geant4 multithreading

Event-level parallelism.
Multicore CPU. Thread-safe.

HiEn physics events are independent => data parallel => multitask parallelism,
may be implemented on computing clusters and multicore CPU. Thread-safety!

Optimization loop

1. Hotspot analysis.
2. Optimization.
3. QA and speedup evaluation.

Hotspots of the BmnRoot decoding modules

Software tool of hotspot analysis - Intel® VTune™ Profiler.
Main focus of analysis – module of decoding signals from the BM@N detectors of the BmnRoot.

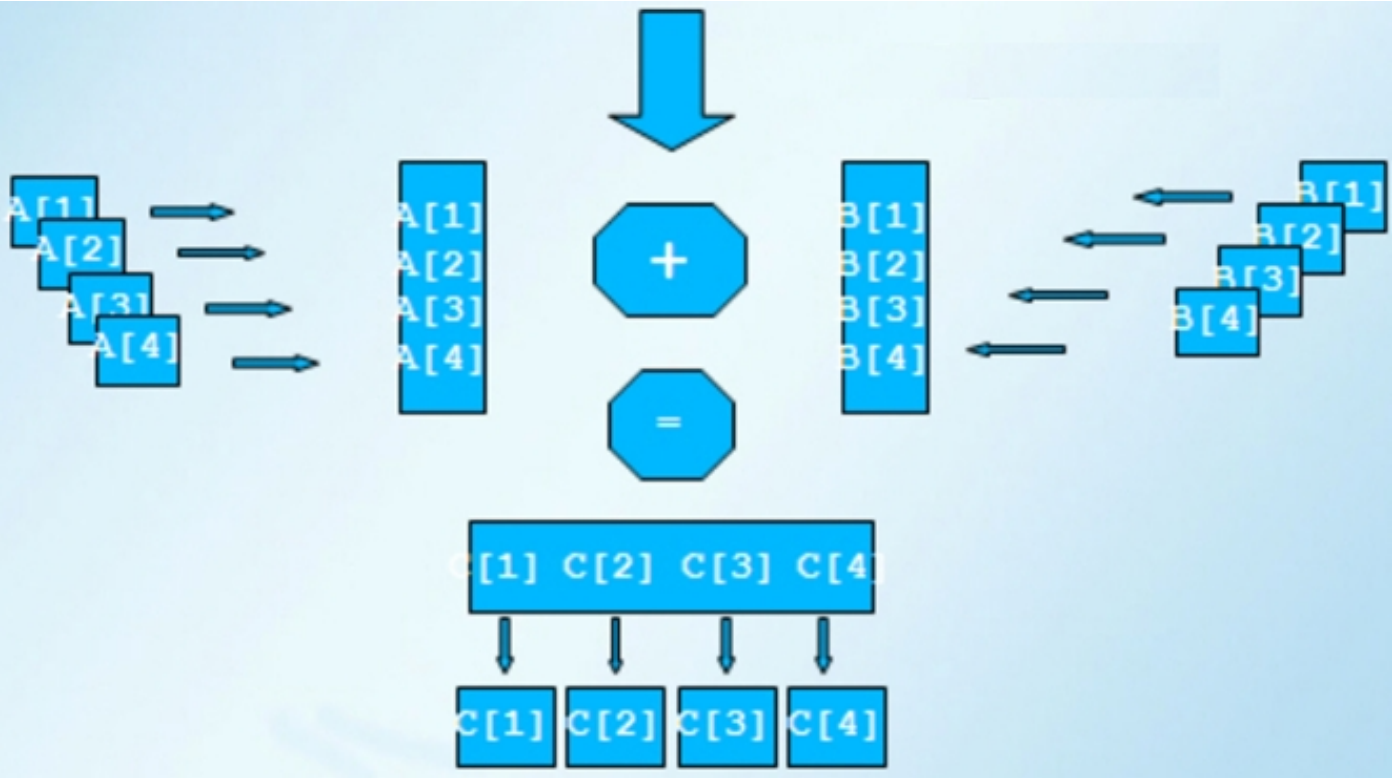
Function / Call Stack	CPU Time ▼	Module
BmnAdcProcessor::CalcEventMods	47.065s	libDecoder.so.0.0.0
BmnAdcProcessor::PrecalcEventMods	12.527s	libDecoder.so.0.0.0
__read	12.094s	libc.so.6
BmnGemRaw2Digit::ProcessAdc	8.779s	libDecoder.so.0.0.0
func@0x18e5d4	5.198s	libc.so.6
__write	4.632s	libc.so.6
func@0x18650	4.622s	libpq.so.5
operator new	4.574s	libstdc++.so.6
deflate	4.000s	libz.so.1
TBufferFile::ReadFastArray	2.896s	libRIO.so
frombuf	2.821s	libRIO.so
BmnSiliconRaw2Digit::ProcessAdc	2.752s	libDecoder.so.0.0.0
func@0x999c0	2.492s	libc.so.6
BmnAdcProcessor::RecalculatePedestalsAugmer	2.248s	libDecoder.so.0.0.0

10000 events.
Experiment of 2018 with Ar beam.

Two most important hotspots:

1. **CalcEventMods** – 29% of total execution time;
2. **PrecalcEventMods** – 8% of total execution time.

Vectorization



The data is packed into vectors, which are then processed in parallel.

Dimension and size of vector depend on version of SIMD-extension.

Intrinsics

```
#include<xmmintrin.h>
   /* Arithmetic Operators */
friend F32vec4 operator +(const F32vec4 &a, const F32vec4 &b) {
      return _mm_add_ps(a,b);
}
  /* Logical */
friend F32vec4 operator&( const F32vec4 &a, const F32vec4 &b ){
      return _mm_and_ps(a, b);  // mask returned
}
  /* Comparison */
friend F32vec4 operator<( const F32vec4 &a, const F32vec4 &b ){
      return _mm_cmplt_ps(a, b); // mask returned
}
```

Example of vector overloading

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*.

Vectorisation of decoding module (SSE)

```
// definition of vector variables
fvec fNvalsMin_vec = 0;
fvec * fNvals_vec, * fSMode_vec, * fCMode_vec;
for (Int_t iCr = 0; iCr < fNSerials; ++iCr) {
    // boxing
    fNvals_vec = (fvec *)fNvals[iCr];
    fSMode_vec = (fvec *) fSMode[iCr];
    fCMode_vec = (fvec *) fCMode[iCr];
    for (Int_t iCh = 0; iCh < fNChannels/4; iCh++) {
        // vectorized if
        fSMode_vec[iCh] = if3(fvec( fNvals_vec[iCh] > fNvalsMin_vec ),
            fSMode_vec[iCh] / fNvals_vec[iCh], 0.0);
        fCMode_vec[iCh] = if3(fvec( fNvals_vec[iCh] > fNvalsMin_vec ),
            fCMode_vec[iCh] / fNvals_vec[iCh], 0.0);
    }
    // unboxing
    fNvals[iCr] = (Float_t*)fNvals_vec;
    fSMode[iCr] = (Float_t*)fSMode_vec;
    fCMode[iCr] = (Float_t*)fCMode_vec;
}
```

Code fragment of the vectorised CalcEventMods function

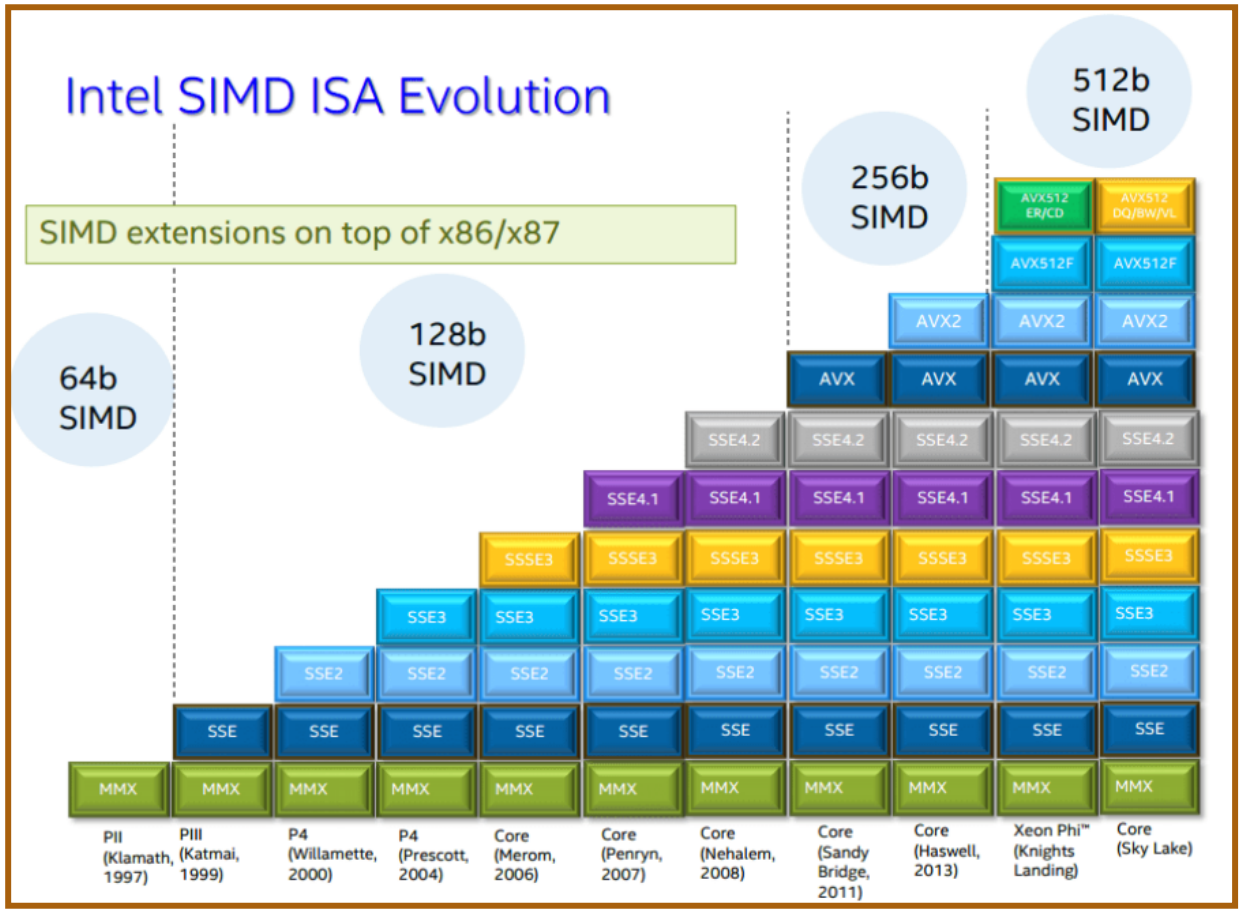
Results of SSE vectorization

Function / Call Stack	CPU Time ▼	Module
BmnAdcProcessor::CalcEventMods_simd	18.422s	libDecoder.so.0.0.0
__read	12.991s	libc.so.6
BmnAdcProcessor::PrecalcEventMods_simd	8.572s	libDecoder.so.0.0.0
BmnGemRaw2Digit::ProcessAdc	8.490s	libDecoder.so.0.0.0
operator new	7.913s	libstdc++.so.6
func@0x18e5d4	5.601s	libc.so.6
func@0x18650	4.876s	libpq.so.5
__write	4.638s	libc.so.6
func@0x999c0	4.386s	libc.so.6
deflate	4.226s	libz.so.1
BmnSiliconRaw2Digit::ProcessAdc	2.938s	libDecoder.so.0.0.0
TBufferFile::ReadFastArray	2.850s	libRIO.so
frombuf	2.784s	libRIO.so
BmnAdcProcessor::RecalculatePedestalsAugr	2.210s	libDecoder.so.0.0.0
fread	1.992s	libc.so.6

List of first hotspots when decoding data after vectorization using SSE instructions

- Speedup of **CalcEventMods** is 2.
- Speedup of **PrecalcEventMods** is 1.25.

AVX (Advanced Vector Extension) vectorization



Results of AVX-vectorization

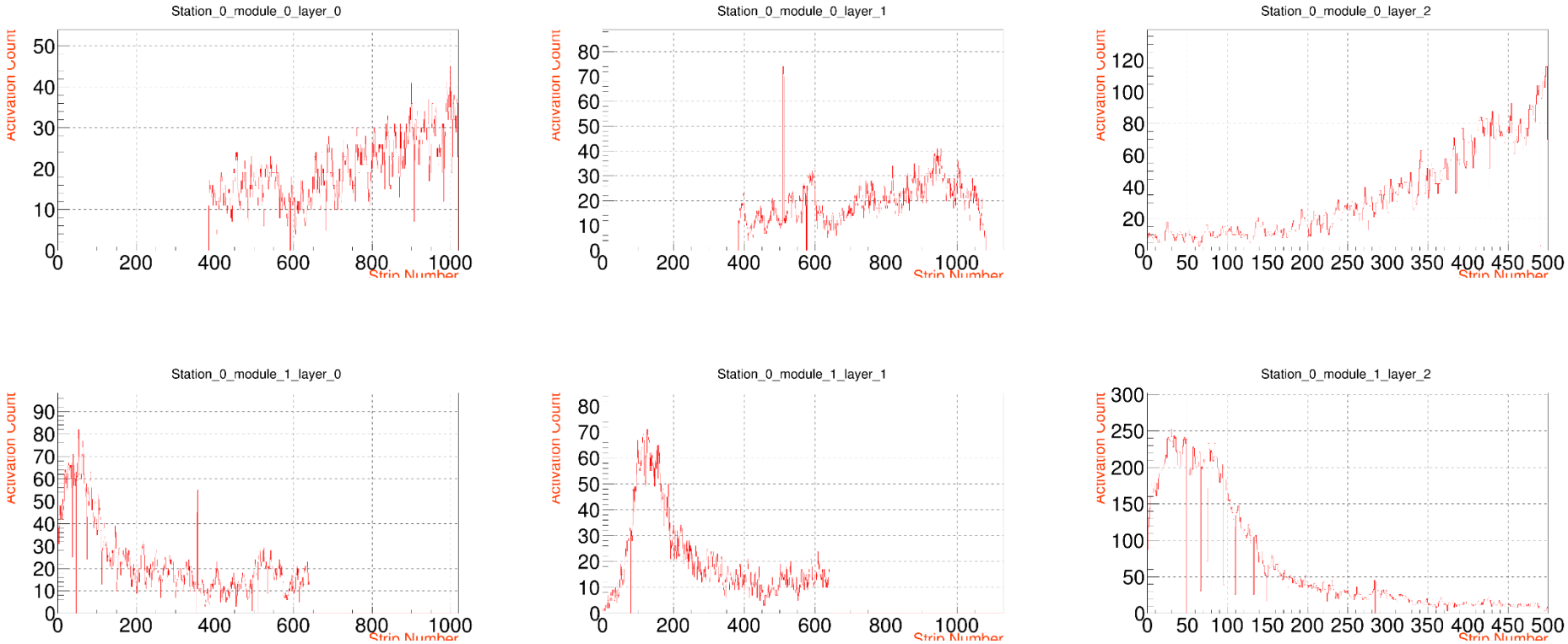
Function / Call Stack	CPU Time ▼	Module
__read	12.588s	libc.so.6
BmnAdcProcessor::CalcEventMods_simd	11.606s	libDecoder.so.0.0.0
BmnAdcProcessor::PrecalcEventMods_simd	8.673s	libDecoder.so.0.0.0
BmnGemRaw2Digit::ProcessAdc	6.361s	libDecoder.so.0.0.0
func@0x18b554	5.236s	libc.so.6
__write	4.444s	libc.so.6
operator new	3.923s	libstdc++.so.6
frombuf	3.142s	libRIO.so
TBufferFile::ReadFastArray	3.020s	libRIO.so
BmnSiliconRaw2Digit::ProcessAdc	2.846s	libDecoder.so.0.0.0
func@0x96870	2.614s	libc.so.6
BmnAdcProcessor::RecalculatePedestalsAugmented	2.560s	libDecoder.so.0.0.0
TStreamerInfo::ReadBuffer<char**>	2.539s	libRIO.so
TBufferFile::WriteFastArray	1.738s	libRIO.so

List of the first hot spots when decoding data after vectorization using AVX instructions

- CalcEventMods function - 8% of total runtime
- PrecalcEventMods function - 6%

The acceleration of the entire decoding program is more than 22%.

QA



Distribution of signals by channels on different elements of the detector chamber

Summary

- Intel® VTune™ Profiler was used to analyze the code of the BmnRoot software package.
- The most time-consuming parts of the BmnRoot software package decoding module have been identified.
- SSE and AVX vectorization of the BmnAdcProcessor module of the BmnRoot package were performed.
- The changed code was checked for efficiency and correctness with a positive result. The execution time of event decoding is reduced by more than 22%.
- The vectorized code was uploaded to the official repository of the BmnRoot software package.

Thank you for attention