ATLAS Offline Software
Performance Monitoring and Optimization

Neelima Chauhan, Gunjan Kabra, Thomas Kittelmann, Robert Langenberg, Rocco Mandrysch, Andreas Salzburger, Rolf Seuster, Elmar Ritsch, Graeme Stewart, Niels van Eldik, Roberto Vitillo

CHEP 2013
14.10 - 18.10.2013

ATLAS

THE UNIVERSITY OF IOWA

Contents

Why do we need software performance improvements?

- LHC will run again in 2015 with:
  - ATLAS trigger rate: $\sim 1\,\mathrm{kHz}$ (2012: $\lesssim 400\,\mathrm{Hz}$)
  - 14 TeV and 25 ns bunch spacing
  - average 25 to 40 interactions per bunch crossing
- For reconstruction software:
  - processing time per event needs to be improved substantially
  - memory consumption needs to be decreased
- Several projects for speed improvement are in progress
- Performance studies of two of these projects will be discussed here

Performance measurements

- Profiling shows highest CPU/event consumers are algorithms for track reconstruction
- Several mathematical operations are executed in these algorithms:
  - vector/matrix operations via CLHEP framework
  - trigonometric functions via standard GNU libm mathematical library
- The operations were monitored by:
  - Intel's Pin tool
  - PAPI
- CPU performance comparisons studies between different mathematical libraries were made in a simple test framework

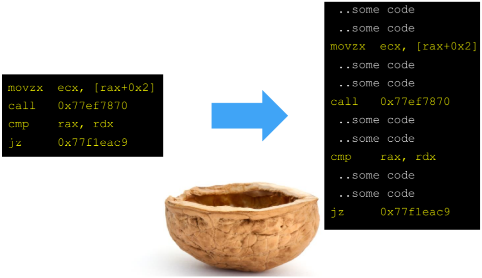
Linear algebra libraries

CLHEP	Eigen	SMatrix	Intel Math Kernel Library
• C++ utility classes for HEP	• C++ templates (headers only) • supports SIMD vectorization	• Implemented in ROOT as expression templates	• BLAS and LAPACK interface • highly optimized
	• expression templates allow removal of temporaries and lazy evaluation		

- All libraries support matrices and vectors with all sizes
- CLHEP is not maintained anymore and not perform well

Pin tool

- Dynamic binary instrumentation framework
  - includes API for abstracting underlying instruction set idiosyncrasies
    $\rightarrow$ no recompilation needed
  - can inject code at the level of functions or instructions



- It is the underlying tool used by Intel Parallel Inspector and Amplifier
- http://www.pintool.org/

Results from monitoring CLHEP functions with Pin

- Monitor calls of CLHEP functions:
    - during reconstruction job
    - with 2012 data sample
      (events passed any Jet, Tau or Missing ET trigger chain)

Five CLHEP functions with highest number of calls:

Function	Calls/Evt
`HepVector::~HepVector()`	3691535
`HepSymMatrix::HepSymMatrix(HepSymMatrix const &)`	1702193
`HepVector::HepVector(int, int)`	1593544
`operator*(HepMatrix const&, HepSymMatrix const&)`	93120
`operator*(HepMatrix const&, HepVector const&)`	42918

Results from monitoring CLHEP functions with Pin

'HepMatrix*HepSymMatrix' arguments with highest number of calls:

1st Argument	2nd Argument	Calls/Evt
3×3	3×3	29333
3×2	2×2	28139
3×5	5×5	13003

'HepSymMatrix*HepVector' arguments with highest number of calls:

1st Argument	2nd Argument	Calls/Evt
5×3	3	23676
3×5	3	11802
1×5	5	4718

- Thanks to pintool,

  now we know how we use CLHEP inside our code,

  with this knowledge we can setup and analyse a realistic test bed
- The results from this testbed are presented in the following slides

PAPI (Performance API)

- Platform-independent interface for hardware performance counters such as: floating point operations, level 1 cache misses, single/double precision vector/SIMD instructions
- Contains low- and high-level sets of routines for accessing counters:

  low level: controls and provides access to all counters

  high level: easily allows one to start, stop and read the counters
- http://icl.cs.utk.edu/papi/

Results from monitoring matrix/vector operations with PAPI

- Monitor floating point operations of several matrix/vector calls with PAPI in a simple test framework
- Compare CLHEP with other classes: Eigen and SMatrix

Floating point operations of 3-dimensional vector/matrix calls:

Operations	CLHEP	Eigen	SMatrix
Matrix allocation	9	9	9
Vector allocation	3	3	3
Vector $+$ Vector	3	3	3
Matrix $\times$ Vector	18	15	n./a.
Matrix $\times$ Matrix	54	47	46

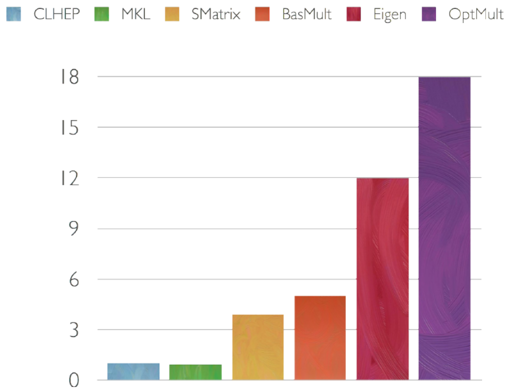
- Matrix $\times$ Vector is not direct available in SMatrix (ROOT v5.34.04)

Further studies: speed comparison measurements

- Evaluated in a small test framework:
  - CPU time of different matrix multiplications
  - comparison studies between: CLHEP, Eigen, SMatrix, MKL and two hand coded C++ routines: BasMult (non-vectorised) and OptMult (vectorised)
- Compiler setup : gcc 4.7.2 and '-O3' for vectorization
- Implemented matrix multiplications:
  - 4×4 with square matrices (including only here BasMult and OptMult)
  - rectangular matrices: $A_{5\times 3} \times B_{3\times 5}$
  - template expression: $C_{5\times 5} = \alpha A_5 B_{3\times 5} + \beta C_{5\times 5}$

Speed comparison with 4×4 square matrices

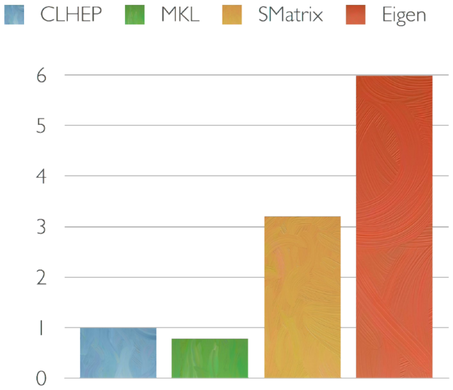
Speedup factor w.r.t. CLHEP:



- Hand vectorized operation is the fastest
- MKL is going though function calls → overhead

Speed comparison with rectangular matrices: $A_{5\times 3} \times B_{3\times 5}$

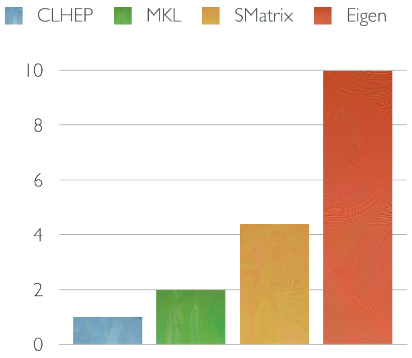
Speedup factor w.r.t. CLHEP:



- Without vectorization
- MKL is going though function calls $\rightarrow$ overhead

Speed comparison with expression templates: $C_{5\times5} = \alpha A_5 B_{3\times5} + \beta C_{5\times5}$

Speedup factor w.r.t. CLHEP:



- Without vectorization
- MKL performs better and overhead is diminished

Conclusion of speed comparison studies: matrix operations

- Hand vectorized operation is the fastest,
  but hand written code is less maintainable and error prone
- Eigen is the fastest linear algebra library
- ATLAS decided to replace CLHEP with Eigen for linear algebra operation for track reconstruction

Trigonometric functions

- GNU libm used as default for trigonometric functions in ATLAS software
- Monitored calls and instructions with Pin during reconstruction job with 2012 data sample

Results with Pin and test framework:

Function	M Call/Evt	Time/Calls [ns]	Time/Evt [s]
exp	3.4	146	0.496
cos	2.5	149	0.373
sin	2.2	149	0.328
atanf	2.1	22	0.0462
sincosf	2.1	24	0.050

- Total times of all trigonometric functions per event: 2.037 s of 14.41 s

CPU time comparison study with alternative math libraries

- VDT
  - developed by CMS
  - designed for auto-vectorization with fast calculations using Padé approximations
  - can be inlined with different API calls, or built into a non-inlined 'drop-in' library (used in this study)
  - further detailed information in Danilo Piparo's talk
- libimf
  - performance optimized library from Intel (Version 2013)
  - can be used as 'drop in' replacement with LD_PRELOAD (use multiple code path for SSE and AVX instructions)
- CPU time comparison study: running reconstruction job with 2012 data sample with GNU libm, VDT and libimf.

CPU time comparison study with alternative math libraries

Results of CPU time comparison study:

Math library	Relative to GNU libm
GNU libm	1.000
VDT	0.923
libimf	0.919

- Reconstruction jobs were running on Sandy Bridge processor with AVX extensions

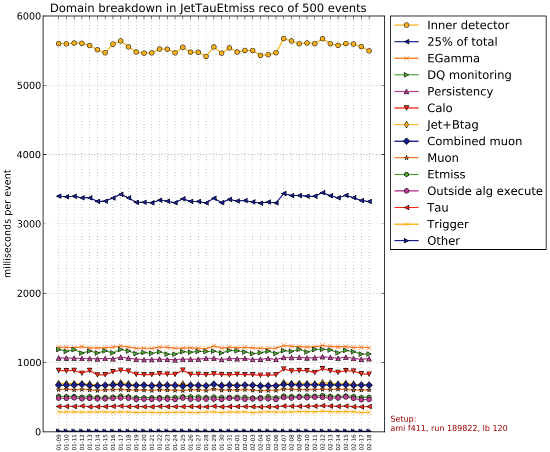
Conclusion:

- libimf provides the fastest trigonometric functions
- ATLAS decided to replace GNU libm with libimf, but keep VDT available in ATLAS software

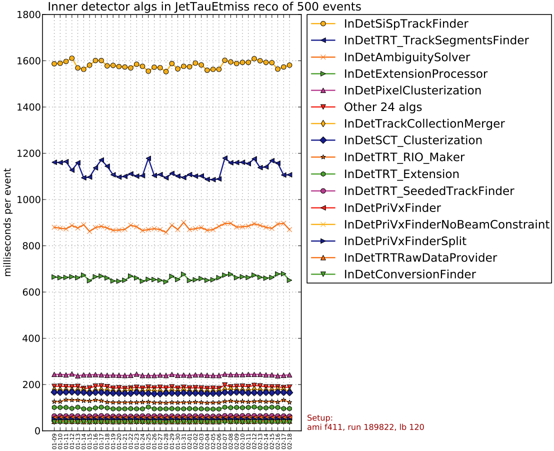
Summary and Conclusion

- Pin provides detailed information about how ATLAS software uses CLHEP and trigonometric functions
- PAPI is an analysis API for hardware performance counters
- Use of these tools has already helped ATLAS achieve significant speed ups in our offline software
- Comparison studies showed:
  - Eigen is the fastest library for matrix and vector operations
  - libimf is the fastest library for trigonometric functions
- ATLAS decided to replace:
  - CLHEP with Eigen for linear algebra operations for track reconstruction
  - GNU libm with libimf for trigonometric functions
- More details about upgrades in tracking algorithms: Talk by Robert Langenberg

Backup slides



- CPU time breakdown per domain depending on the day after the release build during the night
- Measure while processing a data sample from 2011



- CPU time breakdown per domain depending on the day after the release build during the night
- Measure while processing a data sample from 2011

Results with PAPI of matrix/vector operations

- Monitor in a simple test framework floating point operation of several matrix/vector calls with PAPI
- Compare CLHEP with other classes: Eigen and SMatrix

Floating operations of 4-dimensional vector/matrix

Operations	CLHEP	Eigen	SMatrix/SVector
Matrix allocation	16	16	16
Vector allocation	4	4	4
Vector + Vector	4	2	4
Matrix × Vector	32	15	-
Matrix × Matrix	128	58	112

- Matrix × Vector is not availble in SMatrix

Speed comparison with 4×4 square matrices

- Additionally: setup of matrix multiplication with 'std::vectors'
  - basic setup (not vectorized)
  - optimized setup: vectorized without horizontal sums

Basic Multiplication (BasMult):

```
for(int i = 0; i < 16; i+=4){
  for(int j = 0; j < 4; j++){
    z[i+j] = x[i] * y[j] + x[i+1] * y[4 + j] \
    + x[i+2] * y[8 + j] + x[i+3] * y[12 + j];
  }
}
```

Optimized Multiplication (OptMult):

```
for(int i = 0; i < 16; i+=4){
    Vec4d r1 = Vec4d(x[i]) * Vec4d(y);
    for(int j = 1; j < 4; j++){ r1 += Vec4d(x[i+j]) * Vec4d(&y[j*4]); }
    r1.store(&z[i]);
  }
```

CLHEP

- CLHEP - A Class Library for High Energy Physics
- http://proj-clhep.web.cern.ch/proj-clhep/
- A set of HEP-specific utility classes such as random generators, physics vectors, geometry and linear algebra
- CLHEP provides a generic interface for any-dimension matrix/vector
- Problem:
  - Not maintained anymore
  - Not well performed (especially matrix operations)

Eigen

- http://eigen.tuxfamily.org/
- Pure C^{++} template library
  - header only → no binary to compile/install
  - Opensource: MPL2
- It supports:
  - all matrix sizes
  - SIMD vectorization
  - compilers (gcc, icc, clang, ...)
- It is optimized for
  - small fixed-size matrices
  - arbitrarily large dynamic size matrices

SMatrix

- ROOT C^{++} package for high performance vector and matrix computations
- http://root.cern.ch/root/html/MATH_SMATRIX_Index.html
- Implemented as expression templates
- Provide matrix/vector classes of arbitrary dimensions and type
- Classes are templated on the dimension of the matrix/vector and on the scalar type
- Problem:
  - Supports only symmetric matrices
  - Not complete linear algebra package such as Intel MKL or Eigen

Intel Math Kernel Library (MKL)

- http://software.intel.com/en-us/intel-mkl
- Includes:
  - Basic Linear Algebra Subprograms (BLAS)
  - LAPACK routines for solving systems of linear equations
- Optimized:
  - on modern Intel processors
  - for large matrices and BLAS operations: $C = \alpha AB + \beta C$