Nikhef
Jan Just Keijser
Nikhef/PDP
Amsterdam

CHEP2015
OKINAWA, japan

ROOT/RooFIT optimizations for Intel Xeon Phi

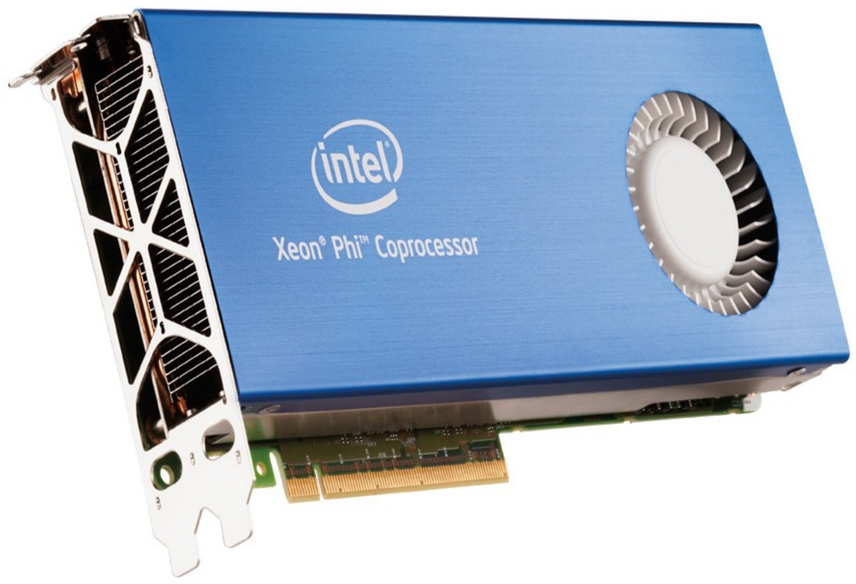
Step-like approach to optimizing ROOT/RooFIT

1. Port ROOT and RooFIT to Intel Xeon Phi (-mmic) architecture
2. Run stresstest benchmarks on regular Xeon E5 and Xeon Phi
3. Ensure all tests pass and all results match
4. Per source file: use Intel compiler flags to generate optimization and vectorization reports

   -O3 -vec-report=7 -qopt-report=5

5, Tune code by hand, add '#pragma omp' directives
6. Recompile
7. Rerun stresstest benchmarks
8. Go to step 3 and repeat.

This is a slow process!

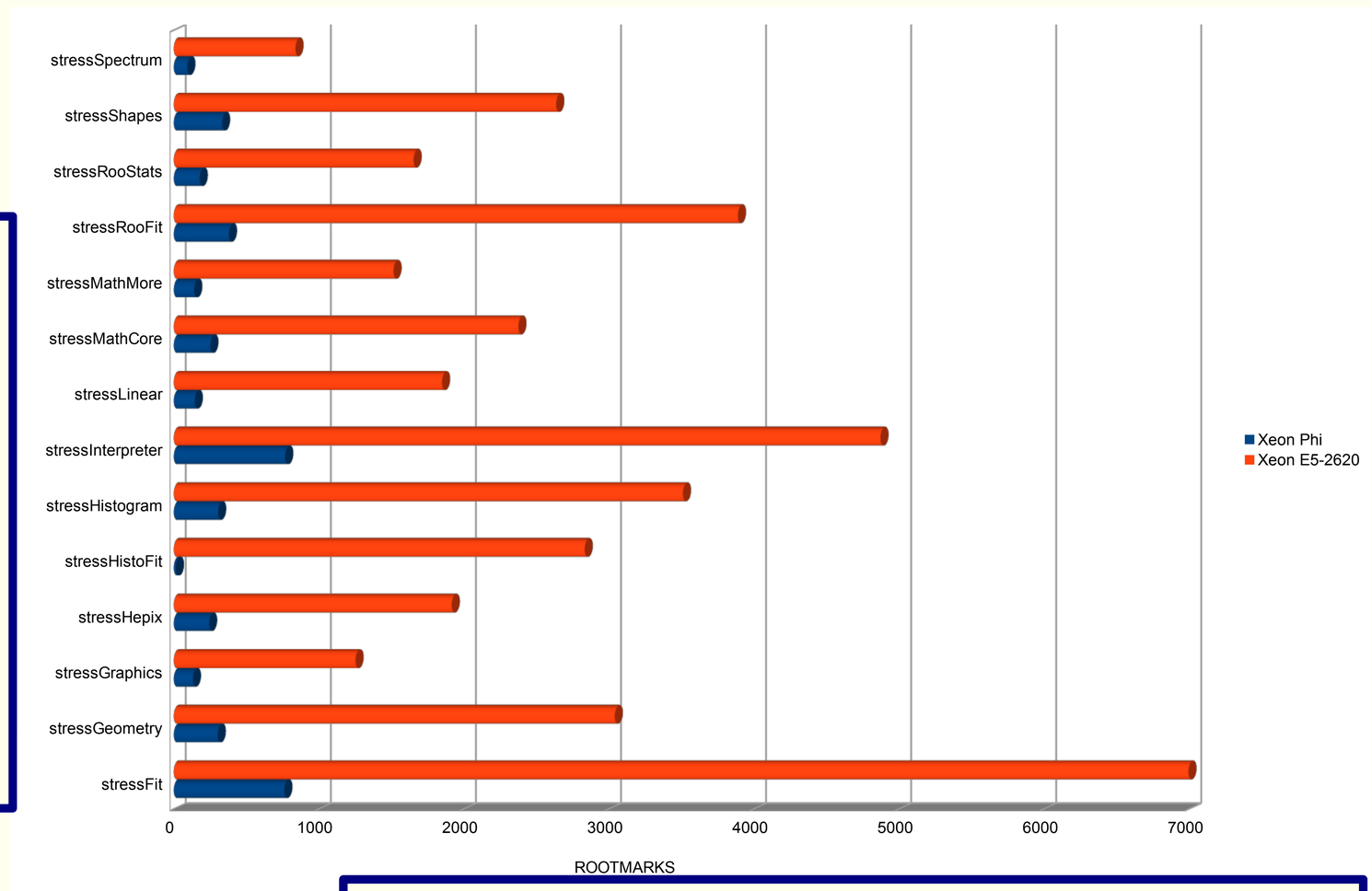


House of ~~Cards~~Versions

Versions of all software components are important:

- Scientific Linux 6.6
- Intel MPSS stack 3.4
- Intel C/C++ Composer XE compiler v15.0.0.090
- Root 5.34.19 (with fixes applied)

ROOT/RooFIT stress benchmark results - unoptimized



Results

- Building ROOT for Xeon Phi is actually cross-compiling to a different platform – all underlying libraries needed to be ported as well
- All stresstests pass except one (compiler optimization error)
- Current Xeon Phi performance is bad (single threaded!)
- Several Intel C/C++ compiler bugs found (& reported)

Next steps:

- Wrinkle out remaining compiler mistakes/bugs
- Use Intel's VTune tools to find optimization hotspots
- Fix the hotspots, go back to step 3

Keep this in mind while porting software:

“When teaching a pig how to dance, it is not important how well the pig dances. It's amazing that the pig dances at all.”
(author unknown)