SAND2013-9999P

Exceptional Science in the National Interest

The Path to Exascale

Experiences porting and debugging for Intel Xeon Phi

Simon D. Hammond, FNITS/SMTS

Scalable Computer Architectures (01422)
Sandia National Laboratories, NM, USA
sdhammo@sandia.gov

Overview of Sandia

Sandia National Laboratories

- Two main sites - Albuquerque, NM and Livermore, CA
- Varied portfolio of workloads
  - Department of Energy (ASCR, NNSA/ASC)
  - Collaborations with industry
  - Applied engineering, mathematics, physics, computer science
- Heavy demands for high performance computing
  - Growth in traditional physics as well as big data and efficient analytics

Do More in Parallel

Sandia National Laboratories

- Growing demands not just for *more* computing but also *more* capable computing

  Pressure to deliver computationally fast algorithms for increasingly large data sets, reduced turn around times and high resolution or greater accuracy

- Leads to a greater focus on how to scale our applications

  Scaling not just to more nodes but also how to scale within the node, handling potentially many memory domains (NUMA, GPUs *etc*)

Where We Are...

Sandia National Laboratories

- Highly Varied Application Pool

  Fortran, C and C++, some >1M lines of code, sophisticated models, evolved over long periods, in many cases well validated with physical experiments

- Mainly based on explicit message passing (MPI)

- Little use of threading or vectorization

  Developers realize there is value but question is how to add these to the applications with a complete rewriting

- Willingness to change code but not to keep rewriting

Our Path Forward

Sandia National Laboratories

- Widespread use of *mini-apps* as a mechanism to capture and explore fundamental areas of interest

  Mantevo Project: http://www.mantevo.org

- Department of Energy NNSA/ASC Test Bed Program

  Make representative computing architectures available for experimentation including Xeon Phi, NVIDIA GPUs, AMD Fusion APUs, IBM POWER, Cray XC30 and soon ARM

  http://www.sandia.gov/asc/computational_systems/HAAPS.html

- Architectural simulation capabilities being developed

  Allows us to explore issues further out and assess whether our changes are robust enough to last many hardware generations

Experiences with Xeon Phi

Sandia National Laboratories

- Initial ports of an application are actually very easy

  MPI, OpenMP, Cilk Plus, TBB, MKL *etc* will all work "out of the box" on Xeon Phi

- Initial performance will probably be poor unless you treat Xeon Phi as an MPI everywhere model (expect slower than Sandy Bridge)

- MPI everywhere probably means your code will have same errors as it would on Xeon

- Motivates the addition of OpenMP pragmas and code changes to improve vectorization

Experiences with Xeon Phi

Sandia National Laboratories

- Adding loop-level OpenMP pragmas and fixing vectorization will get you a reasonable way but we are pushing for more

- Use affinity and environment control (e.g. MPI, OpenMP)

  Up to a 2x performance difference if you get this right

- Challenge: more complex loop structures with function calls

  Keeping a "global" view of what you are parallelizing is very hard even if you the algorithm you have to know *how* it is implemented across the function stack

- This is where the bugs begin to set in - expectations of how code is implemented don't *always* hold true

  Parallelism exposes the strain often non-deterministically or with very subtle errors sometimes for specific combinations of inputs

Experiences with Xeon Phi

Sandia National Laboratories

- Traditional path has been to use some debugging tools and a fair amount of printf (written to screen)

- Serious problems using this approach for Xeon Phi
  - Using print statements in threads creates a *lot* of output - too much to really process
  - The printing often changes the performance (and sometimes removes the error)
  - May need to run many times to see the effect even though it is present

- These necessarily aren't *new* problems but the scale of them is harder to handle and the focus on threading/vectorization is causing them to appear

Experiences with Debugging

Sandia National Laboratories

- Often subtle data races (made more likely by high thread count)

  Find them with a debugger, employ use of OpenMP atomics or criticals

- Logic errors when you aren't using "parallel for"

  OpenMP Parallel regions need to carefully thought out, careful alignment for data structures improves performance but increases risk of arithmetic being wrong

- The more heroic the coding, the more you may need a debugger

- Being able to see what threads are doing (in a debugger visual) is a huge help, especially to compare parameters

Experiences with the Software Stack

Sandia National Laboratories

- New compiler and new architecture exposes bugs in the code

  Compiler implements a new strategy for compilation, developers say this is a “compiler bug”

- Be careful of being tempted to use SIMD and vector dependency pragmas

  You **will** spend time debugging these statements unless you are sure they are valid

- Software stack was originally more unstable but is a great improvement today

  More to come from Intel and there are compiler and MPI updates frequently to help

Worth the Strain?

Sandia National Laboratories

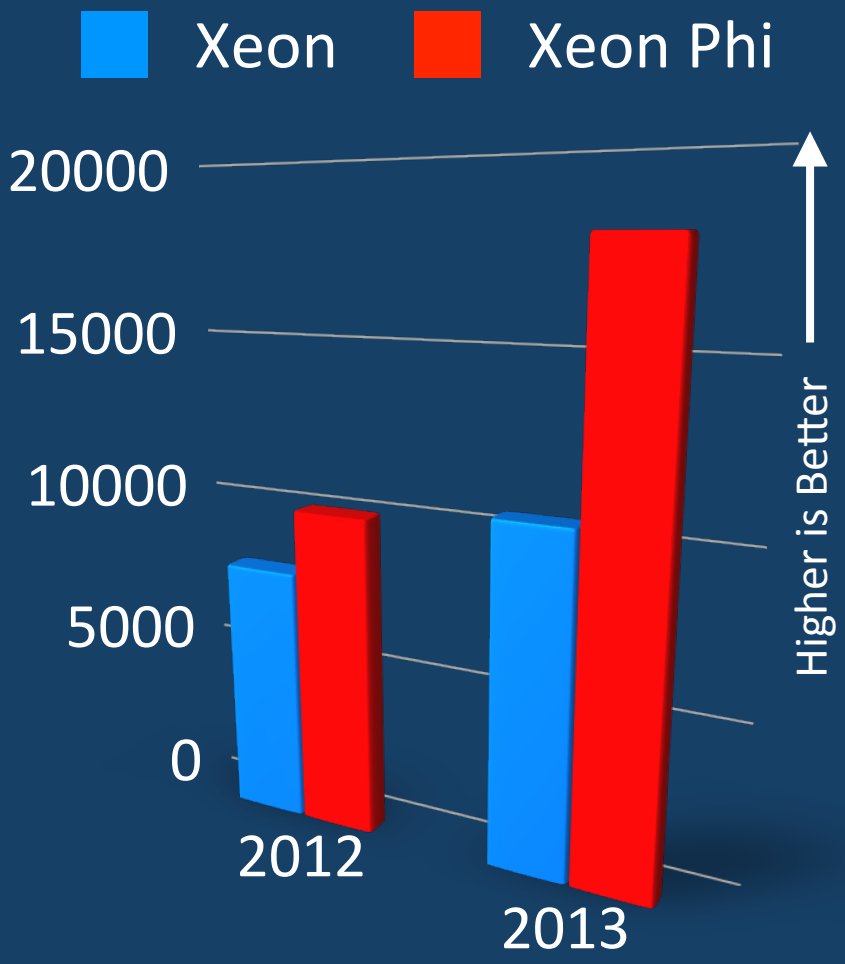
- First time through can be tough

  Lots to consider, not easy to find solutions, vectorization can be very challenging, threading creates bugs

- Gain experience with mini-apps or kernels

- Finding other things are also important

  Selectively using large pages, careful placement of threads, many options to MPI



MiniFE Performance on Pre-Production Intel Xeon Phi co-processor (codenamed Knights Corner). Data from Sandia Nat. Labs

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.