Compilers and Tools for Software Scaling Challenges

Patrick M^{c}Cormick

Los Alamos National Laboratory

What is the Major Challenge to Reaching Productive Exascale Computing?

- **New processor architectures + "old" algorithms and code**
  - Memory wall, heterogeneity, lack of parallelism, etc.
  - Impact will be broad – tools, computation, data analysis, file systems, etc.

How Can Tools & Compilers Help Address this Issue?

- **Compilers (short term):**
  - Multi-/many-core extensions (e.g. OpenMP)
  - GPU-aware extensions (e.g. PGI Accelerator)
  - Versus CUDA, OpenCL?
- **Tools:**
  - GPUs: We’re finally starting to see classic tools (debuggers, profilers, etc)
  - Open questions: At scale? Details of more complex memory hierarchies?
- **Fundamentally we need more/better/new tools…**

What Won’t Compilers and Tools be Able to Help With?

- **They won’t “Save the day”…**
  - There will be no “auto-magic” way to rewrite/port/parallelize/scale your code…
- **But they could make our lives easier…**
  - But the community is going to have to be willing to invest in the efforts to make this happen…

Tools and compilers for petascale were incremental changes, why is this not the case for exascale?

- Well, we’re managing to get by on petascale but…
- Exascale, especially heterogeneous, is a substantial change

What is the one piece of current software that you would scrap or replace? Why?

- **What a loaded question…** 🙂
  - Vi? Emacs?
  - FORTRAN? C++? C?
  - OpenCL? CUDA?
  - MPI?
- **Overall we should be thinking more abstractly to manage complexity…**
  - For example, see Stanford's DSL work at the PPL.