LLNL's Chapel Experiences

Chapel Lightning Talks (BOF)
SC12, Salt Lake City, UT
Nov 15, 2012

Rob Neely
Jeff Keasler
Bert Still
Evi Dube

LLNL-PRES-599372

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Chapel and the long view from some app developers

- Our interest in adopting Chapel for production apps is for many years out
  - Large ASC multi-physics codes take 6-9+ years to (re)write
  - Languages take a generation to develop and mature
  - We need to invest now in languages we plan on using in 10-15+ years
    - Else, we'll still have no viable alternatives to using C++, Fortran, python, MPI+X in 2025!
- Our language priorities:
  - Primary: Performance portable, robust, flexible/adaptable
  - Secondary (but desirable!): Elegance and user productivity

Chapel seems to have struck a chord with LLNL developers. Chapel is:

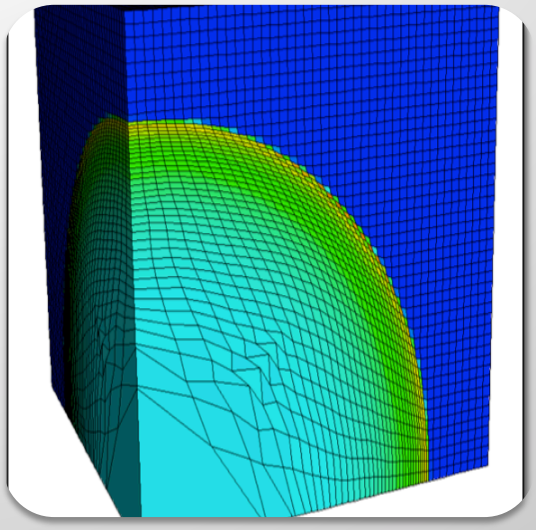
- The most appealing alternative to status quo + incremental improvements we've been exposed to
- Easy to learn – not "exotic" to non computer-scientists
- A nice serial language, even without parallel constructs
- Performance portable – "baked-in" to the language
- Gets the abstractions "just right". E.g.
  - Iterators, locales, distributed domains
  - Very easy to move between different data layouts
- Well thought-out: Follows *principal of least astonishment*

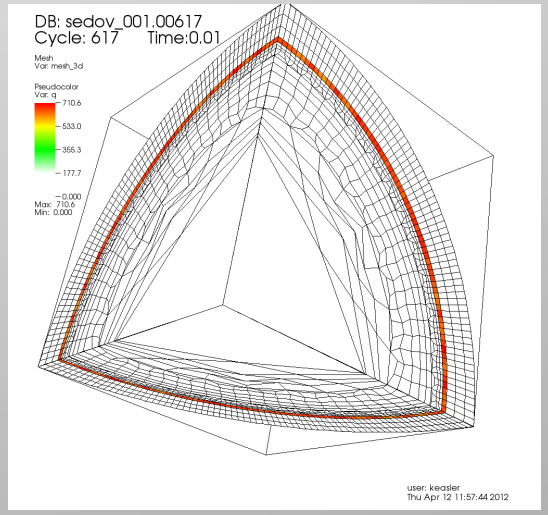
One can almost imagine asking a physicist to write in Chapel!

Timeline: early experiences with Chapel at LLNL

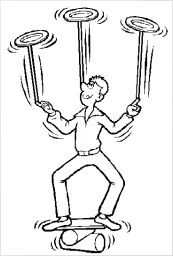
April 2011 ..... Sept 2012

- Initial Development of LULESH Proxy App at Cray
- LLNL Programming Model Survey
- Invitation for Brad C to visit LLNL
  - Tutorial on Chapel basics
  - Result: General enthusiasm from developers
- LLNL gains basic familiarity, learns from initial LULESH port
- Reciprocated visit to Seattle
  - **Block Coding -> Unstructured Coding ~ 6 hours**
  - **25 extra lines of code!**
- Continued work by Chapel team on LULESH
- Paper submitted to IPDPS (pending)





Chapel must balance the needs of the research community with those of potential production users (note: we're the latter!)



- Existing Chapel features feel rich enough to support mesh-based physics apps
  - Need to verify via ports of more/all of our proxy apps
  - Especially need to investigate non-mesh based codes – e.g. particle transport
- Effort needed to (im)prove scalability and performance
  - Lots of known low-hanging fruit
- Hierarchical locales needed to support likely future architectures
  - Accelerators/co-processors, processor-in-memory, NVRAM, etc.
- This must be a community effort
  - No one organization can support alone, must leverage
  - Need vendor support/commitment for platform-specific optimizations
  - But how?


Lawrence Livermore
National Laboratory