DEVELOPING, CONFIGURING, BUILDING, AND DEPLOYING HPC SOFTWARE

MAY 18, 2016
BARRY SMITH
MATHEMATICS AND COMPUTER SCIENCE DIVISION
ARGONNE NATIONAL LABORATORY

WEBINAR SERIES: COLLABORATION AMONG THE IDEAS SCIENTIFIC SOFTWARE PRODUCTIVITY PROJECT, ALCF, OLCF, NERSC

Argonne NATIONAL LABORATORY | Leadership Computing Facility | IDEAS productivity | NERSC | OAK RIDGE National Laboratory | OAK RIDGE LEADERSHIP COMPUTING FACILITY

OBJECTIVES OF THE SERIES

- To bring knowledge of **useful** software engineering practices to HPC scientific code developers
  - Not to prescribe any set of practices as **must use**
    - Be informative about practices that have worked for some projects
    - Emphasis on adoption of practices that help productivity rather than put unsustainable burden
    - Customization as needed – based on information made available
- We will do it through examples and case studies
  - References for available resources
  - Suggestions for further reading

CODE DEVELOPMENT TOPICS

- Emacs/Vim (or IDE such as Eclipse, Visual Studio, Xcode)
- Make/gnumake
- Configure (GNU autotools) / CMake
- Tar ☺
- Git/Mecurial/SVN
  - source code repositories and control
  - next presentation

See on-line demos.

TOPICS COVERED

- Editing tools to search within source code
  - Emacs/Vim – etags and tags
  - Compiling from Emacs: finding compiler errors…
  - IDE – code completion, compiling, syntax checking
    - Very powerful
    - More difficult to use with diverse development team who are not using the same IDE

TOPICS COVERED

- Make/gnumake
  - Rules for compiling code
  - Handling dependencies
  - Automatically computing dependencies
  - Providing help messages
  - Creating libraries
  - Make is slightly more portable (and much clearer) than gnumake. Use gnumake only when needed.

TOPICS COVERED

- Autotools
  - Generating system dependent information for compiling software
- Tar
  - Creating tarball for distribution
  - Providing rule in the makefile