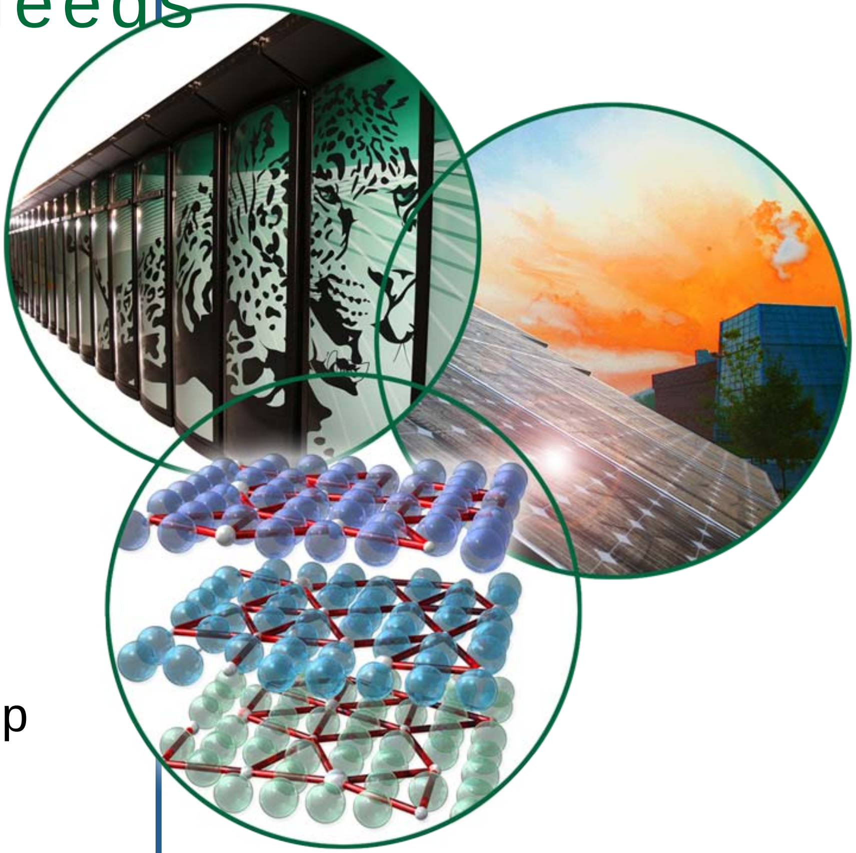
Oak Ridge Leadership Computing Facility Tool Needs Perspective



Richard Graham
OLCF Programming Environment Lead
CSM – Application Performance Tools Group
Leader

U.S. DEPARTMENT OF ENERGY

OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE DEPARTMENT OF ENERGY

Operational Characteristics

- Production use of facility (stability, sustainable support, long-term reproducibility) ➔ Considerable inertia to change
- Users typically run on a variety of platforms and demand portable performance, requiring standard (or, at least, ubiquitous) solutions
- Well defined (long term) support models
- Do want to address future needs
- Tool usage is low (but is rising)
  - Involvement of consumers in determining what is produced is critical

Programming Environment Requirements

- Portable programming model
  - Cross platform
  - General purpose (not aimed at a small number of problem domains)
- Full fledged programming environment
  - Generate executables
    - Correct
    - Efficient
    - Supports the rest of the tool chain
  - Analyze run-time characteristics of executables
    - Correctness
    - Performance
  - Accelerate source-code changes
    - Local
    - Global

Programming Environment Requirements – cont'd

- Mathematical library support
- Well defined support model
  - Fix tool defects
  - Training
- Long-term support model
- Cross-platform tools
- Multiple compilers, debuggers, performance analysis tools
  - Redundancy in critical tool-chain
  - Broad range of analysis capabilities

OLCF Titan Project – Programming Environment strategy

- New system architecture
  - GPU based
  - Virtually non-existent HPC Programming Environment
- Leverage existing commercial efforts
  - Compilers
  - Debuggers
  - Performance analysis tools
  - Math Libraries
- Impact the second round of applications being ported to the system
- Start to do research on the longer-term technical challenges
  - Institutional funding
  - Collaborations

Compilers

Compiler Support - Approach

- Enhance existing compiler capabilities
  - CAPS HMPP compiler – started Dec 2009
    - Cross platform full support for dynamic host/accelerator computing
    - Coordinate with VampirNG/VampirTrace and DDT
  - Cray Integrated Open PE
    - Packaging, testing, and supporting 3rd party compilers (PGI)
- Develop new Cray Compilation Environment (CCE)
  - Incremental releases every 4-6 months
- Enhance the OpenMP standard for acceleration support
  - ORNL, Cray, CAPS, PGI, and others members on the sub-committee

Cray and CAPS NDA

Accomplishments to date

- HMPP (2.4)
  - Supports C++ and directives
  - Works with MADNESS application (C++)
  - Supports directives to reuse data resident in the GPU
  - Inlining support
  - User defined data type support
  - Control data layout/coordination (CPU vs. GPU)
  - Asynchronous data transfer between CPU and GPU
  - Shared memory direct copy
  - Inter-procedural Fortran 90 module support
  - Support for C++ HMPP Runtime API (extending to C and Fortran)
  - HMPP Wizard Tool (help insert directives)
  - Define compiler and performance analysis tools support

- Cray Compilation Env
  - Not public at this stage

- Cray Integrated Open PE Accelerator Enhancements
  - Compilers: PGI Accelerator
  - Library/Tools: NVIDIA toolchain/SDK
  - Performance analysis: VAMPIR
  - Debugging: TotalView, DDT

Cray and CAPS NDA

Debuggers

Goals

- Debug entire application
  - Host processor
  - Accelerator
- Debug at scale
- Debug in the context of the user source code

Approach

- Based on Allinea Software, Inc's DDT debugger
  - Complete support for Heterogeneous Multi-Core support
  - Rely on NVIDIA cuda-gdb for GPU debugging
- Leverage ongoing scalability work for OLCF
  - 3 year project which began mid 2009
  - Scalable infrastructure developed
    - Startup on 220K processors
    - Routine debugging at 100,000+ processes, with full applications
  - Moving focus to scalable analysis

Accelerator debugging enhancements

- Phase I – Q4 calendar year 2010
  - Improved thread support
  - GPU scalability
- Phase 2 – Q2 calendar year 2011
  - HMPP and PGI heterogeneous compiler support
  - Improved thread support

Allinea NDA

Performance Analysis Tools

Goals

- Analyze performance at scale
- Analyze full application performance
  - Host processor
  - Accelerator
- Analyze the performance in the context of user source code

Approach

- Enhance Current Tools
  - Vampir
  - CrayPAT, Apprentice
  - Use NVIDIA performance counter interface
- New Capabilities
  - Cray Optimization Explorer
  - HMPP wizard
- Tighter compiler integration

New Capabilities

- VampirNG/VampirTrace
- Improve GPU support
- Improved Scalability: Improve scalability

- Cray Performance Tools
  - Cray Optimization Explorer (COE)
    - Scoping tool to help users port and optimize applications
    - Performance measurement and analysis tools for porting and optimization
  - CrayPAT/Apprentice^2
    - Integrated with COE

Cray and TU-Dresden NDA

OLCF: Long-Term Functionality Requirements

- Tools must work on full system scale
- Strong support for large scale source-code transformations
  - Porting and optimizing codes
  - Support production code bases: order 1,000,000+ lines of source-code, multi-language
  - Support for major architecture changes
  - Semi-automated: need to speed up the porting process an order of magnitude
- Detailed memory performance analysis
  - Local (full memory hierarchy) and remote
- Usable tools
  - Analysis in a user friendly context
- Interoperable Tool Chain
- “Traditional” tool functionality still needed