INL/EXT-12-25580

Towards a Supported Common NEAMS Software Stack

Cormac Garvey

April 2012

The INL is a U.S. Department of Energy National Laboratory
operated by Battelle Energy Alliance

INL/EXT-12-25580

Towards a Supported Common NEAMS Software Stack

Cormac Garvey

April 2012

Idaho National Laboratory
Center for Advanced Modeling and Simulation
Idaho Falls, Idaho 83415

http://www.inl.gov

Prepared for
NEAMS
and for the
U.S. Department of Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

Towards a supported common NEAMS software stack

Cormac Garvey
Center for Advanced Modeling and Simulation
Idaho National Laboratory

Introduction

The NEAMS software development efforts benefits from the reuse of existing high quality third party libraries (TPL), which have resulted from numerous years of open-source community development and previous DOE investments. Each NEAMS code is dependent on a specific collection of TPLs, because they often include layered interdependencies, we call such a collection of TPLs a *software stack*. A TPL stack provides key functionality such as access to numerical solvers, geometric modeling, mesh generation and parallel I/O. The following third party software stack related activities are currently undertaken by each NEAMS code group.

- Third party software stack needs to be built, installed and tested before any work on or use of a NEAMS code can begin.
- Each time a NEAMS code is moved to a new HPC resource, the third party software stack needs to be built/installed/tested. Each HPC resource may have a different software environment complicating matters.
- As bugs are fixed and features are added to TPLs (e.g. cutting-edge numerical algorithms for the latest HPC hardware), they need to be updated regularly. Quality assurance tests for NEAMS codes need to be re-run to verify the new TPL did not break anything. The API for some TPL's are not backward compatible and therefore break the NEAMS simulation codes, this can be a time consuming task to trouble-shoot/correct this problem.
- A NEAMS code may uncover a critical bug in a TPL. NEAMS code developers may need to spend a considerable amount of time communicating with the TPL developers to have the software bug fixed or establish a sufficient temporary work-around.
- Multiple versions of TPL stacks need to be maintained corresponding to different compilers, different compiler options and different MPI libraries.

All of the above third party software stack tasks take up considerable time for each NEAMS code team and taking time away from developing the NEAMS IPSC Simulation codes.

In this report we propose a more efficient and cost-effective approach to handling the NEAMS third party software stack requirements. Initially, we focus on the NEAMS FUEL and Reactors codes, but the principles can be applied to other programs like

CASL[1]. A centrally maintained common NEAMS third party software stack has the following advantages.

- The NEAMS code teams can focus all their time on developing their respective simulation code.
- Common TPL requirements across NEAMS code groups can be identified and managed with improved efficiency by eliminating redundancy.
- All leadership class HPC resources will have an identical developer and user environments with all the required TPL's already installed and tested, eliminating the effort needed to port to new HPC systems.
- Third party software stack will be more reliable because the it will be actively maintained and upgraded on a regular basis.
- As the NEAMS IPSC's mature, a detailed performance analysis can be undertaken to tune the software stack to deliver optimal performance.
- TPL stack configurations supporting different qualities of service can be independently installed and maintained. For example, a configuration maximizing debug features can be installed for NEAMS code developers while another maximizing performance can be installed for NEAMS code users.

The NEAMS IPSC simulation codes are still being actively developed and improved. So, in the short-term what is required is a stable, reliable and consistent third party software stack environment, which will allow the NEAMS code developers to efficiently develop and refine their simulation codes.
Based on the needs outlined above, a third party software stack will be provided on all NEAMS HPC resources based on the GNU compiler suite (gcc, g++ and gfortran) (later the tuned third-party software stack may use a different compiler, depending on the performance analysis.). The GNU compiler suite seems like a natural choice, it's a free open source compiler, available on all HPC systems and most TPL's support it.

In this report we will summarize the key NEAMS Fuel and Reactor simulation codes and identify what TPL's they are currently are dependent on. Then we will propose a common NEAMS third party software stack, describe its basic structure and outline its implementation plan.

MOOSE[2] (Gaston, Newman and Hansen)/BISON (Newman, Hansen and Gaston)/ Marmot (Tonks, Hansen and Gaston)

A Key component of several NEAMS Fuel codes is the BISON application, which can simulate the physical properties of individual nuclear fuel pellets of a typical light water reactor over the entire lifespan of those fuel elements. A wide range of interesting physical phenomena from reactor startup through latter stages of fuel burnup can be simulated. Extensive studies of thermomechanical behavior of reactor fuel pellets have been performed with the BISON code, including swelling due to thermal expansion and oxygen diffusion, material cracking due to strain, nonlinear

[1] Consortium for Advanced modeling and Simulation of Light water reactors.
[2] Multiphysics Object Oriented Simulation Environment

mechanics, and lower-length scale coupling. Bison is an application built on the MOOSE common framework approach. Marmot, which likewise is built on MOOSE, provides the lower-length scale mesoscale simulations to compute properties needed by the continuum-scale solution process.
Figure 1 lists the current MOOSE/BISON third party software stack requirements.

Moose/Bison/Marmot Third Party Libraries requirements
MVAPICH2 1.6
PETSc 3.1-p8 **(Balay, Brown and Buschelman)**
hypre 2.7b **(Falgout, Kolev and Schroder)**
BLAS **(Dongarra)**
LAPACK 3.1 **(LAPACK web page)**
TBB **(Threading Building Blocks)**

Figure 1 NEAMS FUEL IPSC (Moose/Bison/Marmot) third party library requirements.

AMP[3] (Clarno)/FUEL

The AMP nuclear Fuel performance code is targeted for the design, evaluation, uncertainty quantification, and optimization of coupled-physics nuclear fuel/assembly simulations. It includes thermo-mechanics, for fuel performance, radiation transport and depletion (from SCALE), and single channel, one-dimensional flow. The AMP/FUEL code requires the third party software outlined in figure 2.

Amp/Fuel/Libmesh Third Party Library requirements
CMAKE 2.8.6 or higher **(CMAKE web page)**
MPICH2 1.3.2 **(MPICH2 web page)**
HDF5 1.8.7 **(HDF5 web page.)**
Hypre 2.4.0b
Silo 4.7.2 **(Silo web page.)**
SUNDIALS 2.4.0 **(Sundials web page)**
Trilinos 10.2.* **(Trilinos web page)**

Figure 2 NEAMS FUEL (AMP/FUEL) Third-Party Library requirements.

[3] Advanced Multi-Physics

NEK5000 (Fischer, Lottes and Kerkemeier), MOAB[4] (Tautges, MOAB) & CGMA[5] (Tautges, CGMA)

Nek5000 is an open-source (GPL) highly parallel incompressible computational fluid dynamics solver based on the spectral element method. It is a key component of Argonne's SHARP project with a current emphasis on modeling coupled neutronics and thermal-hydraulics.
MOAB is a component for representing and evaluating mesh data. MOAB can store structured and unstructured meshes, consisting of elements from the finite element method plus other element shapes such as general polygons and polyhedral.
CGMA is a code library, which provides geometry functionality used for mesh generation and other applications. This functionality includes that commonly found in solid modeling engines, like geometry creation, query and modification; CGMA also includes geometry decomposition tools and support for shared material interfaces.
The NEK5000, MOAB and CGMA third party software requirements are presented in figure 3.

NEK5000/MOAB/CGMA Third Party Library requirements
HDF5 1.8.3
Zoltan **(Zoltan web page)** (optional)
METIS **(Karypis)** (optional)
BLAS
NetCDF 4.1.1 **(netCDF web page)**

Figure 3 NEAMS Reactors IPSC (NEK5000/MOAB/CGMA) third party library requirements.

Unification of NEAMS FUEL and Reactor TPL stack requirements

One of the advantages of centrally maintaining and deploying a NEAMS TPL stack is that common TPL's can be identified and we can start the process of simplifying and unifying NEAMS TPL requirements. From a stability and consistency point of view it is also advisable to include the gcc compiler suite used in the third party software stack (the latest gcc version which does not break any of the NEAMS IPSC builds). This is especially important as the third party software stack is deployed to other HPC resources, eliminating possible errors due to gcc compiler version differences between HPC resources.

The second area in which there appears to be some flexibility in TPL choice is in the MPI library selection. All NEAMS codes require an MPI library, some specify which type and version, but all NEAMS codes will run correctly as long as a stable and

[4] A Mesh-Oriented datABase
[5] Common Geometry Module, Argonne.

compliant (with MPI2 standards (Message Passing Interface Forum. MPI2: Extensions to the Message-Passing Interface.)) MPI library is used. We propose that the latest stable MVAPICH2 (Panda) MPI implementation be used and included in the NEAMS third party software stack.

There is also overlap with PETSc, hypre and HDF5 among the NEAMS code teams. There are version number differences between the NEAMS IPSC's, so some work will be required to standardize these libraries. The current proposed complete NEAMS Fuel and Reactor third party software stack is highlighted in figure 4, some of the library version numbers have not been determined yet, the latest version which works with all NEAMS IPSC's will be chosen.

NEAMS Fuel and Reactor IPSC Third party software stack
GCC 4.6.1 or higher
MVAPICH2 1.7 or higher
CMAKE 2.8.6 or higher
PETSc
hypre
Trilinos 10.6.2 or higher
SUNDIALS 2.4.0 or higher
HDF5
NetCDF 4.1.1 or higher
BLAS
LAPACK 3.1 or higher
Silo 4.7.2 or higher
Zoltan
METIS
TBB

Figure 4 Proposed supported/maintained NEAMS Fuel and Reactor IPSC third party software stack.

Figure 5 Structure of Common NEAMS software stack.

Structure of NEAMS Fuel and Reactor IPSC TPL stack

Implementing the NEAMS Fuel and Reactor IPSC TPL stack on a number of different HPC resources have many challenges.

- Access privileges will vary from one HPC environment to the next, which will limit what can be done on each system.
- Some of the NEAMS IPSC codes have export control restrictions and so these codes will need to have the appropriate protections, limiting access.

One way the NEAMS third party software stack can be implemented consistently on all NEAMS HPC resources is by setting up appropriate NEAMS work areas or project groups and controlling access to certain directories with the appropriate access permissions set. Figure 5 outlines the structure of the proposed NEAMS Fuel and Reactor IPSC third party software stack environment. For example the NEAMS, Fuel, Reactors, Utilities and TPL directories will be locked-down to the NEAMS group only and each IPSC (directory MOOSE, AMP, NEK5000, MOAB and LASSO) will have their own group access privileges. It is also important that each TPL have a debug version which will help track down software bugs.

Linux environment modules (Furlani) will be developed (defining TPL locations and setting appropriate environmental variables) so that each NEAMS code group can easily and conveniently bring the NEAMS TPL into their environment in a consistent manner, no matter what HPC resource they are using.

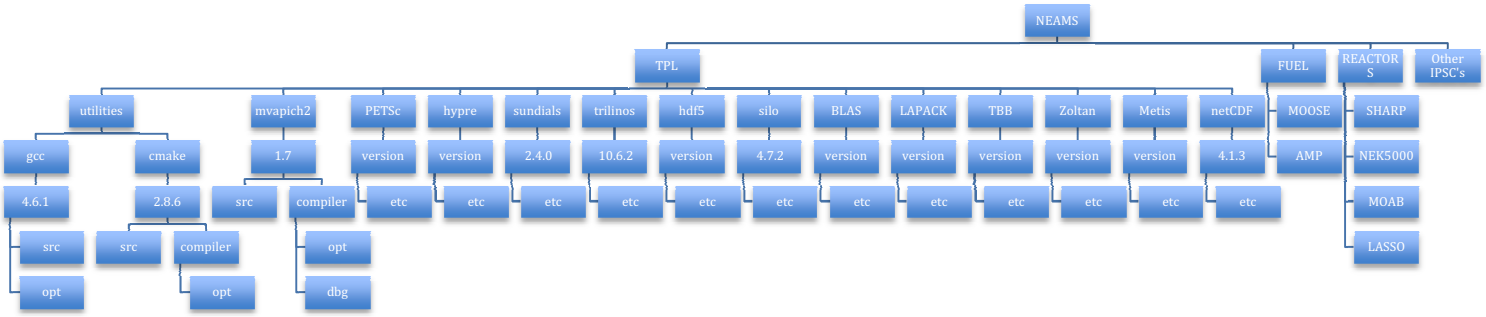

Figure 6 Directory structure of NEAMS supported TPL stack, FUEL and REACTORS IPSC's work areas.

Cost of TPL stack maintenance and deployment service

As already mentioned in this report, maintaining a stable and up to date TPL stack can be quite time consuming for each NEAMS code team and take away a considerable portion of their time from developing the simulation code. There are also considerable inefficiencies in each NEAMS code team maintaining its own TPL stack (often building TPL's which have already been built and vetted by other IPSC's). A more efficient way of dealing with the NEAMS TPL stack requirements is to have them maintained by a single central body (possibly the NEAMS/ECT). By unifying as much as possible the NEAMS TPL stack requirements and automating some of the building/installing/deploying tasks, we estimate that the cost for this ongoing NEAMS common software stack service would be about 1.0 FTE (excluding deploying on other NEAMS HPC resources).

Conclusion

The NEAMS Fuel and Reactor IPSC's code is dependent on a complex third party software stack. Currently, each IPSC spends a considerable amount of time and effort maintaining their own TPL stack. Centralized maintenance of a common NEAMS software stack can improve efficiency and reduce duplicate efforts. We propose that the NEAMS/ECT take responsibility for the NEAMS IPSC TPL stack building, installing, testing, integrating and deploying it on all NEAMS HPC resources. The cost to the NEAMS program for this TPL maintenance service is quite reasonable, only 1.0 FTE, considerably less time than the combined time currently devoted to this task by all NEAMS IPSC's.

The next phase in this effort will be to implement the proposed common third party software stack at the INL HPC resources and then deploy it to all NEAMS HPC resources.

Works Cited

1. Balay, Satish, et al. PETSc Web Page. 2011. <http://www.mcs.anl.gov/petsc>.
2. Clarno, Kevin. AMP_Nuclear_Fuel_Performance. 2010. <http://neptune.ornl.gov/wiki/index.php/AMP_Nuclear_Fuel_Performance>.
3. CMAKE web page. <http://www.cmake.org>.
4. Dongarra, J. "Basic Linear Algebra Subprograms Technical Forum Standard." International Journal of High Performance Applications and Supercomputing 16 (2002): 1-111.

5. Falgout, Rob, et al. hypre web page. 2011. <https://computation.llnl.gov/casc/linear_solvers/sls_hypre.html>.
6. Fischer, Paul F., James W. Lottes and Stefan G. Kerkemeier. nek5000. 2008. <http://nek5000.mcs.anl.gov>.
7. Furlani, John L. "Modules: Providing a Flexible User Environment." Proceedings of the Fifth Large Installation Systems Administration Conference. San Diego, 1991. 141-152.
8. Gaston, D., et al. "MOOSE: A parallel computational framework for coupled systems of nonlinear equations." 2009.
9. HDF5 web page. 2011. <http://www.hdfgroup.org/HDF5>.
10. Karypis, George. METIS web page. 2011. <http://glaros.dtc.umn.edu/gkhome/metis/metis/overview>.
11. LAPACK web page. 2012. <http://www.netlib.org/LAPACK>.
12. Message Passing Interface Forum. MPI2: Extensions to the Message-Passing Interface. 1997. <http://www.mpi-formum.org>.
13. MPICH2 web page. 2012. <http://www.mcs.anl.gov/research/projects/mpich2>.
14. netCDF web page. 2012. <http://www.unidata.ucar.edu/software/netcdf>.
15. Newman, C., G. Hansen and D. Gaston. "Three dimensional coupled simulation of thermomechanics, heat, and oxygen diffusion in UO2 nuclear fuel rods." Journal of Nuclear Materials, 2009.
16. Panda, Dhabaleswar K. MVAPICH2. <http://mvapich2.cse.ohio-state.edu>.
17. Silo web page. 2012. <https://wci.llnl.gov/codes/silo>.
18. Sundials web page. <https://computational.llnl.gov/casc/sundials/main.html>.
19. Tautges, Tim. CGMA. 2010. <http://trac.mcs.anl.gov/projects/ITAPS/wiki/CGM>.
20. —. MOAB. 2010. <http://trac.mcs.anl.gov/projects/ITAPS/wiki/MOAB>.
21. Threading Building Blocks. 2012. <http://threadingbuildingblocks.org>.
22. Tonks, Michael, et al. "Fully-coupled Engineering and Mesoscale Simulations of Thermal Conductivity in UO2 Fuel using and Implicit Multiscale approach." Journal of Physics 180.1 (2009).
23. Trilinos web page. 2012. <http://trilinos.sandia.gov>.
24. Zoltan web page. 2010. <http://www.cs.sandia.gov/Zoltan>.