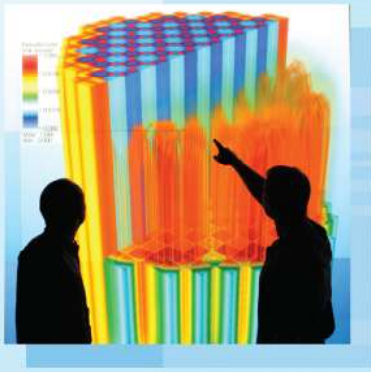
DISCLAIMER

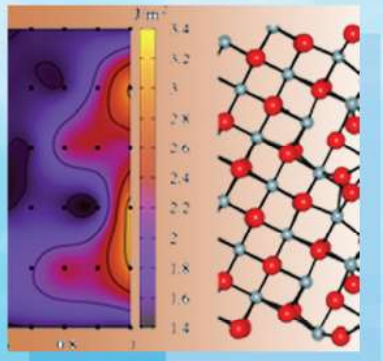
This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof. Reference herein to any social initiative (including but not limited to Diversity, Equity, and Inclusion (DEI); Community Benefits Plans (CBP); Justice 40; etc.) is made by the Author independent of any current requirement by the United States Government and does not constitute or imply endorsement, recommendation, or support by the United States Government or any agency thereof.

Power uprates
and plant life extension



Engineering design
and analysis

Science-enabling
high performance
computing



Fundamental science

Plant operational data

CASL-U-2013-0165-000

Consortium for Advanced Simulation of LWRs

Initial Demonstration of Peregrine in VERA-CS

R. Pawlowski
Sandia National Laboratory

J. Turner
Oak Ridge National Laboratory

S. Palmtag
Core Physics

Robert Montgomery
Pacific Northwest National Laboratory

July 31, 2013

U.S. DEPARTMENT OF ENERGY | Nuclear Energy

Oak Ridge National Laboratory

in partnership with

Electric Power Research Institute
Idaho National Laboratory
Los Alamos National Laboratory
Massachusetts Institute of Technology
North Carolina State University
Sandia National Laboratories
Tennessee Valley Authority
University of Michigan
Westinghouse Electric Company

and individual contributions from

Anatech Corporation
ASCOMP GmbH
CD-adapco, Inc
Core Physics, Inc.
City University of New York
Florida State University
Notre Dame University
Imperial College London
Pacific Northwest National Laboratory
Pennsylvania State University
Rensselaer Polytechnic Institute
Southern States Energy Board
Texas A&M University
University of Florida
University of Tennessee
University of Wisconsin

REVISION LOG

Revision	Date	Affected Sections	Revision Description
Draft	07/27/2013	All	Initial version (draft)
5	07/30/2013	Many	Typos, formatting, incorporated feedback from Rob Montgomery and Kevin Clarno, updated Fig. 3-1, additional references
7	07/31/2013	All	Added Rob Montgomery as author, added DRAFT watermark, changed PEREGRINE back to Peregrine everywhere, added further discussion of results and future work
9	08/01/2013	Summary	additional “future work” items, plus formatting and content of headers and footers

EXECUTIVE SUMMARY

This milestone demonstrates integration of the MOOSE-based Peregrine fuel performance code into the CASL Virtual Environment for Reactor Analysis (VERA). Specifically targeted at VERA's core simulator functionality (VERA-CS), the existing VERA subchannel-neutronics capability based on COBRA-TF (CTF) and Insilico was extended to use the axisymmetric 2D R-Z fuel rod modeling functionality of PEREGINE to replace the simple fuel rod model in CTF. This three-way coupling involved defining and implementing data interfaces, software infrastructure development to support the requirements of each code, and numerical methods to implement the coupled physics. The result is a new VERA capability known as Tiamat.

Development of Tiamat required extensions to the VERA build system to support a "meta build" that uses the MOOSE/libMesh/Peregrine native build system, development of a new driver layer designed to support long-term CASL code integration requirements, and implementation of data transfers for the coupled applications using the DataTransferKit (DTK).

Peregrine is now compiled and tested under the VERA continuous integration server. Tiamat, a new advanced simulation tool consisting of three coupled codes - CTF for multiphase subchannel flow, Insilico for neutronics and Peregrine for fuels performance - has been developed. Tiamat was used to simulate a 17x17 assembly model for Watts Bar Unit 1 Cycle 1, which exceeds the goals of this milestone.

CONTENTS

ACRONYMS

AMA	Advanced Modeling Applications
CASL	Consortium for Advanced Simulation of Light Water Reactors
CTF	COBRA-TF
DOE	U.S. Department of Energy
DOE-NE	U.S. Department of Energy Office of Nuclear Energy
DTK	Data Transfer Kit
FA	Focus Area
HFP	Hot Full Power
HPC	high-performance computing
HZP	Hot Zero Power
INL	Idaho National Laboratory
LANL	Los Alamos National Laboratory
LWR	light water reactor
MPO	Materials Performance and Optimization
ORNL	Oak Ridge National Laboratory
PCI	pellet-cladding interaction
PCM	percent mille (10^{-5})
PoR	plan of record
PPM	parts per million (usually Boron)
PWR	pressurized water reactor
RTM	Radiation Transport Methods
SNL	Sandia National Laboratories
T-H	thermal-hydraulics
TPL	third-party library
V&V	verification and validation
VERA	Virtual Environment for Reactor Applications
VRI	Virtual Reactor Integration

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

Table **Page**

1. Introduction

This report documents the completion of Milestones L2:VRI.P7.02 "Initial Demonstration of Peregrine Integration in VERA-CS" and L3:VRI.PSS.P7.04 "Initial CTF + Insilico + Peregrine Capability". The L2 was formerly listed as L2:MPO.P7.01, but as the milestone progressed, it was reassigned to VRI since the majority of work consisted of software development performed by VRI staff. The milestone L3:VRI.PSS.P7.04 was the VRI milestone to support the MPO L2, but since taking over ownership of the L2, the L3 milestone has been subsumed as part of that. Therefore this report covers both milestones.

The purpose of this milestone is to provide an initial capability demonstration to run a coupled calculation with CTF + Insilico + Peregrine. Completion criteria of the Level 2 Milestone required the capability to run a coupled CTF+Insilico+Peregrine calculation for a 17x17 PWR single-assembly problem. Problem #6 of the challenge problem progression [1] was used for this calculation.

Since the goal was a demonstration, there are many open questions that are identified for follow on PoRs to transform this demonstration into a "production quality" tool.

In Section 2 of this report, a description of the overall code design is described. This section covers a description of the applications, the solution algorithms for each code and a description of how code is transferred between applications. Section 3 provides information on the software integration effort required in the development of the coupled code system. Section 4 contains a description of the test problem used in this Milestone. Section 5 presents the results and discussion of the test problem. Section 6 provides a summary.

This Milestone was a large project and involved the hard work of many people, including (in alphabetical order):

- Roscoe Bartlett, ORNL
- Kenneth (Noel) Belcourt, SNL
- Kevin Clarno, ORNL
- Greg Davidson, ORNL
- Tom Evans, ORNL
- Derek Gaston, INL
- Jason Hales, INL
- Russell Hooper, SNL
- Wenfeng Liu, ANATECH
- Robert Montgomery, PNNL
- Scott Palmtag, Core Physics Inc
- Roger Pawlowski, SNL (Milestone Lead)
- Robert Salko, Penn State University
- Rod Schmidt, SNL
- Dion Sunderland, ANATECH
- John Turner, ORNL

2. Coupled Physics Design

This section describes the overall design of the coupled code in VERA and the individual applications used in the coupling. The coupled code is called "Tiamat", named for a multi-headed dragon tracing back to Babylonian mythos. The mulitheaded nature was to reflect that this coupled driver will allow for simple extension to additional physics and couplings. Section 2.1 describes the individual physics components. Section 2.2 describes the solution procedure and data transfers for the coupled system.

2.1 Participating Physics Components

For this part of the Milestone, three physics application codes are coupled together. All neutronic aspects of the problem (cross-section calculation, neutron transport, power generation) are solved using the Insilico code suite [2,7]. The thermal hydraulics duties are split between solving for energy conservation in the fuel rods by Peregrine [11,12] and energy conservation of the coolant fluid is performed by COBRA-TF [3]. Furthermore, coupling with Peregrine extends the capability to include fuel rod structural mechanics with irradiation effects of materials. The coupling of these codes to create a single-executable multiphysics coupled-code application is done using the VERA infrastructure tool called the Data Transfer Kit (DTK) [5].

2.1.1 COBRA-TF (CTF)

COBRA-TF (CTF) is a thermal-hydraulic simulation code designed for Light Water Reactor (LWR) analysis [3]. CTF has a long lineage that goes back to the original COBRA developed in 1980 by Pacific Northwest Laboratory under sponsorship of the Nuclear Regulatory Commission (NRC). The original COBRA began as a thermal-hydraulic rod-bundle analysis code, but subsequent versions of the code have been continually updated and expanded over the past several decades to cover almost all of the steady-state and transient analysis of a both PWR's and BWR's. CTF is being developed and maintained by the Reactor Dynamics and Fuel Management Group (RDFMG) at the Pennsylvania State University (PSU).

CTF includes a wide range of thermal-hydraulic models important to LWR safety analysis including flow regime dependent two-phase wall heat transfer, inter-phase heat transfer and drag, droplet breakup, and quench-front tracking. CTF also includes several internal models to help facilitate the simulation of actual fuel assemblies. These models include spacer grid models, a pin conduction model, and built-in material properties.

CTF uses a two-fluid, three-field representation of the two-phase flow. The equations and fields solved are:

- Continuous vapor (mass, momentum and energy)
- Continuous liquid (mass, momentum and energy)
- Entrained liquid drops (mass and momentum)
- Non-condensable gas mixture (mass)

Some of the reasons for selecting CTF as the primary T/H solver in the VERA core simulator is the reasonable run-times compared to CFD (although CFD will be available as an option), the fact that it is being actively developed and supported by PSU, and for the ability to support future applications such as transient safety analysis and BWR and SMR applications.

CTF is a control volume code that enforces conservation between volumes. The application code is a steady state solution. Internally it uses a pseudo transient solver to achieve convergence. The interface to the code allows the Tiamat driver to request a solve for a specific steady-state. Therefore each call by the driver to solve for a steady state internally runs a pseudo transient solve.

2.1.2 Insilico (Denovo/XSProc)

Insilico is one of the neutronics solvers in the VERA Core Simulator and is part of the Exnihilo transport suite being developed by ORNL. Insilico is the reactor toolkit package of Exnihilo and includes the reactor toolkit modules used for meshing of PWR geometry, and the cross section generation package based on XSProc. Insilico uses the Denovo deterministic transport code [2,7] to solve for the flux and eigenvalue solutions for the 3D problem using either the discrete ordinates (S_N) solver or the Simplified Legendre (SP_N) solver. Exnihilo also includes the SHIFT Monte Carlo transport package, but SHIFT is not used in this study.

Multigroup cross sections are generated in Insilico using the XSProc, a capability available in the SCALE system [10]. XSProc performs resonance self-shielding with full range Bondarenko factors using either the narrow resonance approximation or the intermediate resonance approximation. The fine energy group structure of the resonance self-shielding calculation can optionally be collapsed to a coarse group structure through a one-dimensional (1D) discrete ordinates transport calculation internal to XSProc. For all of the calculations in this study, the fine energy group structure was collapsed to a 8-group coarse group structure to be used in the Denovo transport solver.

The cross section library used in this study was the SCALE 6.2 252 group ENDF/B-VII.0 neutron cross section data library. This library contains data for 417 nuclides and 19 thermal-scattering moderators.

For coupled calculations, both the S_N and SP_N solvers have been used and tested. However, all of the results in this report were generated with the SP_N solver.

The Insilico code solves for the leading eigenvalue at a single steady state. The interface allows the driver code to request a solve for a specific steady-state.

2.1.3 Peregrine

The Peregrine fuel performance code is being developed by CASL to provide a single rod 3-dimensional fuel performance modeling capability to assess safety margins, and the impact of plant operation and fuel rod design on the thermo-mechanical behavior, including Pellet-Cladding Interaction (PCI) failures in PWRs [11,12]. PCI is controlled by the complex interplay of the mechanical, thermal and chemical behavior of a fuel rod during operation, and therefore modeling PCI requires an integral fuel performance code to simulate the fundamental processes of this behavior.

The focus of the Peregrine is to establish a modern computational framework based on the finite element method to represent the geometric domains of a single nuclear fuel rod composed of UO_2 ceramic pellets contained within a Zircaloy tube. The Peregrine framework consists of a numerical representation of the heat conduction and the equilibrium mechanics equations, which are coupled via the temperature and displacement variables. A material property and constitutive model library has been incorporated into the framework that allows for thermal, mechanical, and chemical property models and irradiation effects models, such as fission

product-induced swelling or irradiation creep to be utilized and modified easily. Finally, the framework allows for versatile time marching algorithms to capture the different temporal regimes associated with fuel performance, such as burnup accumulation over several years followed by a rapid power ramp over several minutes.

Peregrine is built upon INL's MOOSE/ELK/FOX structure/architecture, which is also common to the BISON code [8]. This code architecture uses the finite element method for geometric representation and MOOSE uses a Jacobian-free, Newton-Krylov (JFNK) scheme to solve systems of partial differential equations [9]. While the MOOSE framework can solve either steady state or transient problems, the Peregrine application is a transient code since the material models contain time dependent quantities.

For this milestone, the fuel rod geometric representation in Peregrine was a 2D R-Z axially-symmetric smeared pellet model. The mesh is shown in Figure 2-1.

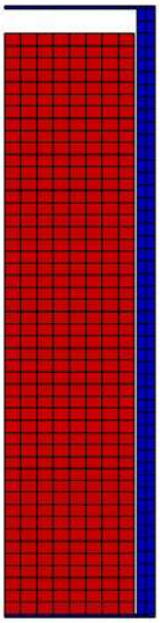


Figure 2-1. 2-D Axisymmetric Finite Element Mesh (FEM) Representation of a Fuel Rod. The red region is fuel and the blue region is cladding. The radial dimension is scaled by a factor of 200.

Peregrine is accessed using the MOOSE MultiApp capability. A separate instance of Peregrine is built for each fuel rod and the applications are solved uncoupled from each other via the MultiApp interface (but coupled to the otheapplication components – CTF and Insilico). Currently the interface supports a "take step" command where the MultiApp is given a time step and attempts to solve each individual fuel rod application separately. Peregrine is a thermo-mechanical code that commutes the thermal expansion, elastic deformation, and cracking of the cladding and pellet in response to the energy generation (heat) and mechanical forces. The interaction of the temperature solution and the mechanical solution is non-linear due to the complex dependency of the material properties on temperature, stress, and strain. As a result, Peregrine must model the heatup of a fuel rod from zero power to full power to appropriately account for these non-linearities. For the coupled solve, Peregrine was ramped to hot full power (HFP) and then was solved coupled to the other codes using a small

time step that allowed the code to equilibrate under iteration with the other codes. As the MultiApp is under development we expect the interface to be extended to allow for a more consistent fully implicit approach.

2.2 Solution Algorithm and Code Coupling

2.2.1 Data Transfers

To couple the codes, data must be passed between each code. Figure 2-2 shows the data transfers between each code.

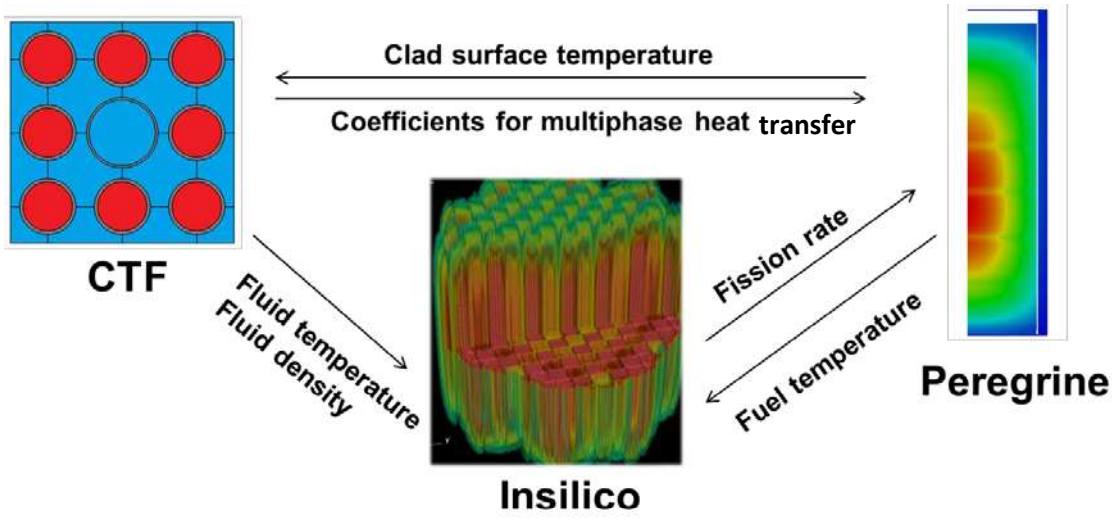


Figure 2-2. Graphical depiction of Tiamat data transfers.

There are five data transfer objects implemented in Tiamat for this capability. All data transfers had to account for parallel communication patterns, unit conversions, pin axis orientation and active fuel offset height. The Data Transfer Kit (DTK) was used for determining the parallel communication mappings and for moving all data between codes. As part of this procedure, DTK performs an intersection search for target coordinate points in source geometry representations. A very valuable benefit was that if a target point was not found in the source geometry, it was recorded and reported. This was critical in tracking down misaligned meshes and coordinate transformation issues between codes. At runtime, if a user makes a mistake in specifying the geometry/mesh information, DTK will help to catch this. All three codes used different unit systems that needed to be converted during data transfers. CTF used English units, Insilico used a mix of CGS and MKS units and Peregrine used MKS units. Pin axis alignment deals with with direction that the fuel rod axis was oriented. The CTF and Insilico codes align the pin axis in the z direction in three dimensional Cartesian coordiantes, but Peregrine used a 2D axially symmetric R-Z model internally and therefore aligned the pin axis on the second coordinate index. In the MOOSE MultiApp DTK interface, an internal translation from 2D cylindrical to 3D Cartesian coordinates is provided, but it keeps the pin aligned in the second coordinate direction (y direction in catesian). Therefore when implementing data transfers involving Peregrine, the y and z coordinates of the target coordinate point vector had to be swapped before passing to the DTK detection algorithm. Each code additionally had a different height at which the active fuel region starts. Each code solves for certain physics above and below the active fuel

region. Therefore each transfer had to be aware of and adjust coordinates for DTK so that the active fuel regions were aligned between codes during the source/target intersection search.

The two data transfers between CTK and Peregrine are essentially surface transfers at the cladding interface. CTF solves for energy conservation in the fluid and Peregrine solves for energy conservation in the fuel pins. CTF provides heat transfer conditions (coolant temperature and heat transfer coefficients) at the fuel pin cladding outer surface and Peregrine provides a temperature at the same surface (in the future, this should include heat flux for transient simulations). Both transfers use the DTK source volume maps where the source code declares a geometric volume where it can evaluate points and the target code declares the coordinates of where it requires field values. In the Peregrine to CTF transfer, a single field, temperature, was transferred. MOOSE post processing was used to evaluate average clad surface temperatures over axially divided regions. Users can specify the number of regions in the input file. In the CTF to Peregrine transfer, four fields were transferred – the coolant liquid average temperature, the coolant vapor average temperature, the coolant liquid heat transfer coefficient and the coolant vapor heat transfer coefficient. The values were average values computed over the four quarter channel control volumes that surround a particular pin. As part of this milestone, a new Neumann-type boundary condition was written for Peregrine that used the four transferred quantities to compute and apply a heat flux on the clad surface boundary of the finite element mesh.

The two data transfers between Insilico and Peregrine are volume to volume transfers. Both transfers use the source volume map from DTK. The data transfer from Peregrine to Insilico moves temperature values into Insilico. MOOSE post processing was used to evaluate average volumetric fuel pin temperatures over axially divided regions. Users can specify the number of regions in the Peregrine input file. The Insilico to Peregrine data transfer maps the fission rate to quadrature points in the Peregrine finite element mesh.

When transferring data to Peregrine, the MultiApp interface was used to connect to DTK. Any data pushed into the MOOSE framework was stored in MOOSE Auxiliary Variables. Therefore, all Peregrine kernels needed to be able to access auxiliary variables. This worked for everything but the fast neutron flux. Therefore this quantity used a user supplied linear profile. In a future PoR, the MOOSE kernels will need to be modified to account for this and Insilico will need to be modified to provide it.

The final data transfer was a single transfer from CTF to Insilico. Since a coupled driver for coupled CTF/Insilico exists [13], this structure was reused for the three code coupling in this milestone. The transfer maps the coolant temperature and density from CTF to Insilico. During the initialization phase, it additionally maps an initial guess for the fuel pin temperatures to Insilico.

Note that the use of DTK means that all data transfers are done directly in memory between MPI processes. There is no file I/O or writing to disks for data transfers.

2.2.2 Solution Procedure

A challenging aspect of this problem is that the different physics associated with these codes are strongly coupled and nonlinear. By strongly-coupled we mean that the quantities calculated in each physics component and passed to the other have a significant impact on the physical quantities computed in other physics components. To solve the coupled system, a simple block-Jacobi fixed-point (FP) iteration used. In this method,

each code is solved concurrently in its own MPI process space using data from the other coupled codes at the previous iterate. The procedure follows:

```
While not converged
        Transfer data between codes
        Apply relaxation
        Solve each code concurrently
        Check for convergence
End while loop
```

An advantage of this scheme is that if the codes are load balanced, each can run in its own process space at the same time, maximizing the process utilization. Separating each component into its own MPI process space also helps to mitigate potential memory issues. Currently, CTF is a serial code and if a full core is run on one process, the memory footprint will be large. If a block Gauss-Seidel were used in this case and all applications exist on every process, there is a concern that memory requirements could exceed available memory on some platforms as we scale to full core. The design of Tiamat allows switching between both Jacobi and Gauss-Seidel algorithms. In addition, a parallel implementation of CTF is in development.

The overall execution is to perform an initialization phase to bring all codes to hot full power (HFP) conditions and then perform the block Jacobi FP iteration between the codes until convergence. The initialization phase starts with a solve of CTF. On this first solve, we use a CTF internal model for the fuel rod energy conservation and neutronics so that is can be solved stand-alone for the starting conditions. Once this model is solved, CTF transfers the coolant temperatures and density and the fuel temperatures to Insilico. Then an Insilico steady state solve is performed using the CTF data. Once Insilico is converged, the data required for Peregrine is transferred from both CTF and Insilico to Peregrine. These HFP conditions are approximate and sometimes cause startup problems for Peregrine. Delaying this transfer did not ease the difficulty. Given the time constraints, startup pathways were not explored in detail and requires further investigation in future PoRs. Most likely a scaled ramping of the Insilico power when transferred to Peregrine will fix this issue. For this demonstration, Peregrine is ramped over 2 days from zero to HFP conditions. Once this is achieved, the initialization phase ends.

The solve phase iterates using block Jacobi FP until convergence. A limitation here is that the control over the MOOSE MultiApp does not allow for multiple iterates or Peregrine “solves” to be performed at each time step. Thus we cannot perform a fully implicit solve of the system at each time step. Given that the transients are very long, we make an assumption of pseudo-steady state and use a very small time step in the MOOSE solve call and iterate over multiple small time steps at HFP to achieve a “steady state” solution at HFP. In future PoRs, an extension of the MOOSE MultiApp interface to allow for this will be required. Once the solve for a fully implicit time step exits, a study can assess any error in the pseudo steady state assumption.

3. Software Integration

Tiamat, which is loosely-based on the existing VERA CTF+Insilico capability, serves as the driver for the coupled CTF+Insilico+Peregrine capability. It extends the CTF+Insilico coupling to allow separation of Insilico processes from the (currently serial) CTF process.

Figure 3-1 shows the VERA package architecture. The components involved in this milestone are highlighted in yellow. In order to integrate Peregrine into VERA, Peregrine, MOOSE and libMesh all had to be incorporated into the build system. Then data transfers had to be implemented between each component in the coupling.

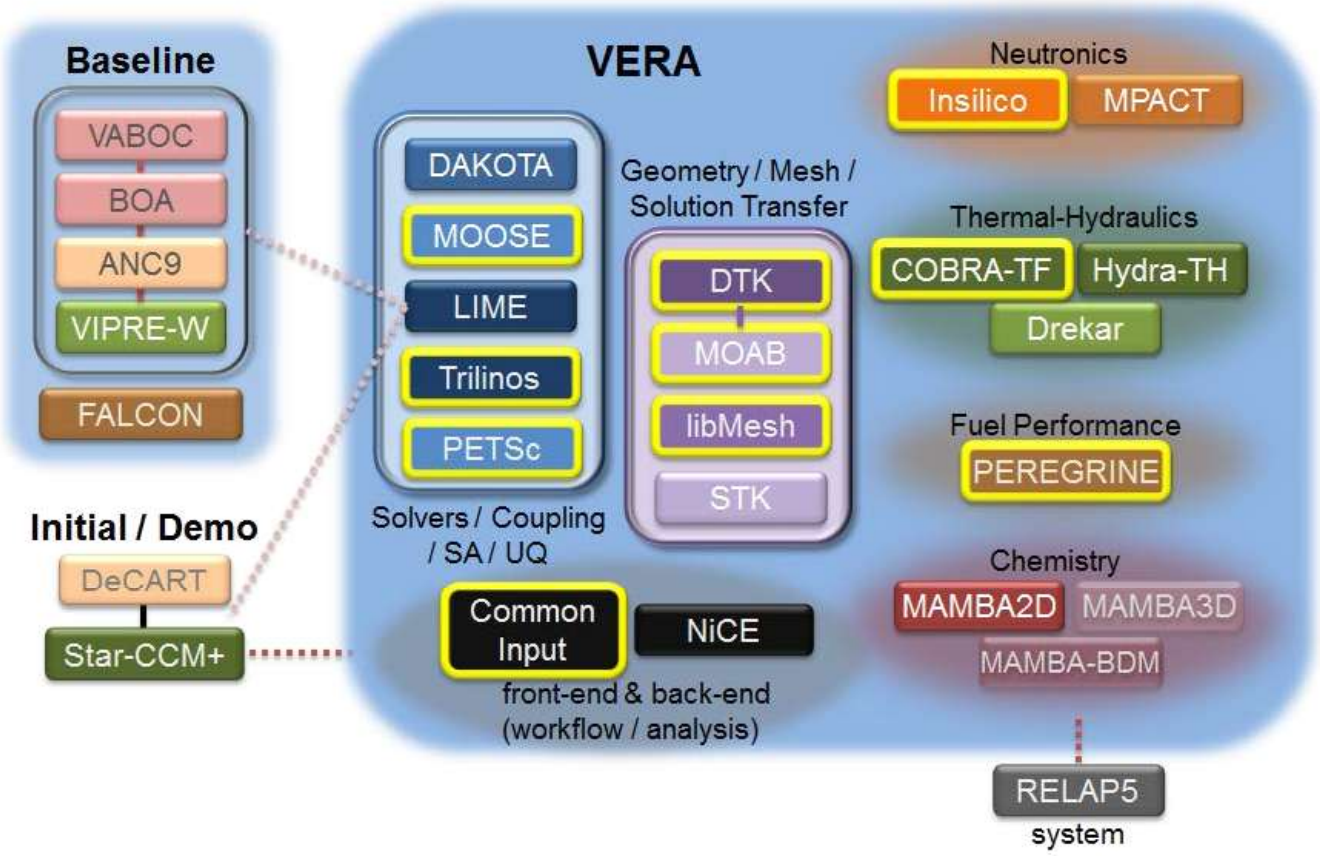


Figure 3-1. VERA package architecture. Yellow outlines indicate components used for this milestone. The MOOSE, libMesh and Peregrine components were integrated into VERA and data transfers between Peregrine, CTF and Insilico were implemented.

3.1 Build System Integration

The precedence in CASL up to this point was to have all integrated codes support the TriBITS build system. If MOOSE/libMesh were made full TriBITS packages, then this would have been a simple integration. However libMesh and MOOSE are both big projects and have many existing users with various funding sources. CASL is just one of many users and can't force them to change (even if it would make our life easier). Therefore we needed to use their native build system. To support the native build systems, we had to develop new capability in TriBITS.

Significant effort went into supporting the native MOOSE/LibMesh build systems. Getting MOOSE/Peregrine to build under TriBITS was not an easy task. MOOSE/Peregrine is built on top of the libMesh library and leverages its build system. libMesh uses Autotools and Moose adds raw makefiles on top of this. libMesh has optional dependencies on Trilinos and if we couple to Moose, it also has a required dependency on Data Transfer Kit. Therefore, we couldn't just build Moose and LibMesh as static TPLs. The configure requirements for libMesh forced us to build and configure Data Transfer Kit and Trilinos (which are configured and built as part of VERA) before configuring libMesh. This is creates a **circular dependence** between build systems! To break this cycle, we had to add the capability to TriBITS to call out to external configure/build systems during the VERA configure/build. This required significant work to add a new capability into TriBITS. The result is that TriBITS is now truly a "meta build system" and this new capability will be leveraged for future work. The DAKOTA team is planning to use this capability and we expect that for Hydra couplings it may be needed for handling the Trilinos/ML/PETSc dependencies.

Here is a list of work performed in developing the build system:

- Developed a new VERA package called CASL_MOOSE that wraps the Moose/libMesh configure/build system and performs the configure/build inline as part of a VERA build. This required writing new commands to get dependency logic and external calls to perform correctly. This is a new capability for TriBITS and can provide a blueprint for any future couplings for packages that don't support a native TriBITS build.
- Changed the export makefile system to write the export makefiles inline during the configure process. The Moose/libMesh configure uses the TriBITS/Trilinos export makefiles for its Trilinos and DTK dependencies. This required changes to core TriBITS functionality and had to be tested thoroughly so as to not disrupt all of CASL and Trilinos users and developers.
- Moose is usually the terminal application or top level so there is no "install" target. Therefore, to treat Moose as a library, we needed to figure out the include header paths, the library paths, the libraries to link against and the correct order to list these libraries on the link line. TriBITS provides this information in the form of cmake configure files and raw "export" makefiles. Since Moose does not, we had to do manual introspection of the Peregrine build and link commands and automate using python scripts where possible. At this point if the libraries required are changed, this will have to be fixed manually. A follow-up PoR items would be for the Moose library to provide the equivalent of a TriBITS export makefile for MOOSE. Then the whole update process could be automated. To date we have updated CASL's version of MOOSE twice (pushing over 1000 new commits each update) and had no issues in updating library dependencies.
- Patched MOOSE to disable c++11 support. This is hard coded in the libMesh configure files to automatically be enabled if the compilers are gnu and the version is 4.4 or above. VERA does not allow c++11 yet and this caused inconsistent builds when using c++11 aware headers. Adding a configure flag to libMesh to disable this feature has been requested.
- Edit the generated export makefiles in the CASL_MOOSE package so that they would compile with Moose. Some of this was due to the FindQT module injecting bad parameters into the link line for the export makefiles. We know how to work around this via configure options or string replaces on the export makefiles during the TriBITS make process. But it would be great to remove Qt dependence in

VERA. This is brought in by SCALE. Elimination of QT requirement would also be very helpful in porting VERA to new platforms.

- There were some other minor issues such as having to set environment variables for libMesh to configure correctly and adding links between directories so that the libMesh autotools configure could detect the correct headers but that was easily handled by TriBITS.
- We encountered significant issues in global variable collisions between packages. This is a common problem due to with packages that do not protect their classes and functions within unique "namespaces" . A significant amount of time had to be devoted to identification of the collisions and coordination with the corresponding application teams to address the issues.

Currently, Tiamat is compiling and is part of the VERA Continuous Integration (CI) testing. Long term, a number of issues will need to be addressed. These are documented in VRI Kanban ticket #2966 and will require effort on part of both the VRI and MOOSE teams.

3.2 Development Process

As part of this milestone, a new moose application was constructed to allow for unit testing of the data transfers to/from MOOSE. The application allows for injecting new MOOSE kernels into any pre-existing MOOSE application. These kernels formed the basis of fast running unit tests so that developers could quickly get turn around during development. This additionally allowed us to simplify the physics and rule out complicating issues when dealing with the real system. For example, in the first tests of transferring data to MOOSE, we used a mock Peregrine input file for MOOSE that solved a simplified heat transfer in a uniform pin/clad with no thermo-mechanical interaction. Without the ability to isolate specific code capabilities and work on manageable chunks, it would have been impossible to couple the codes and have any confidence in results.

The integration process proceeded in small steps that successively added capability. To be able to develop and test code quickly, a series of small problems were added for system and unit level testing. The system level started with the small single pin 8 group problem in the VERA Input driver for CTF and Insilico physics, modified to be extremely fast for development turn around. The MOOSE input file was the simple heat transfer kernel with no complicating physics. Once all transfers were implemented and tested on the single pin, testing progressed to the 3x3 case. This again was taken from the VERA Input driver for CTF and Insilico and used the simple mock driver for Peregrine. Once the data transfers were verified in this manner, the next set of single and 3x3 tests were rerun using the true Peregrine input file. The transition from mock to real Peregrine input allowed for the identification of issues that were physics/discretization specific vs. software implementation. We stress again that had the mock test not been done, debugging of true issues would have been difficult.

Once the 3x3 was running reliably the actual 17x17 Benchmark Problem 6 run was performed.

3.3 Multiphysics Distributor

As mentioned earlier, the Tiamat work has developed tools to assist in splitting applications into unique MPI process spaces. A utility object in Tiamat called a multiphysics distributor was written to handle building the various communicators. Five levels of MPI communicators were needed to create the coupled driver for block Jacobi. Figure 3-2 shows the MPI distribution.

The code builds the standard MPI_COMM_WORLD global communicator. The next level splits MPI_COMM_WORLD into three separate process spaces, one for each application. In the final demonstration run of the 17x17 assembly, CTF is given one process, Insilico is given 36 processes and Peregrine is given 17 processes. The third level of comunicators are the data transfer communicators that are formed from the union of the applications involved in the data transfer. This creates 3 additional communicators. As discussed previously, MOOSE creates multiple instances of fuel rods using the MultiApp capability. Each MultiApp is assigned a subset of the processes contained in the MOOSE communicator. The fourth level (not shown in figure 3-2 are the MultiApp comms. These are internal to the moose MultiApp and not handled of created by Tiamat. The fifth level of communicators are the data transfer between the MultiApp and either CTF or Peregrine. This is formed from the union of the particular MultiApp communicator and either the CTF or Insilico communicator.

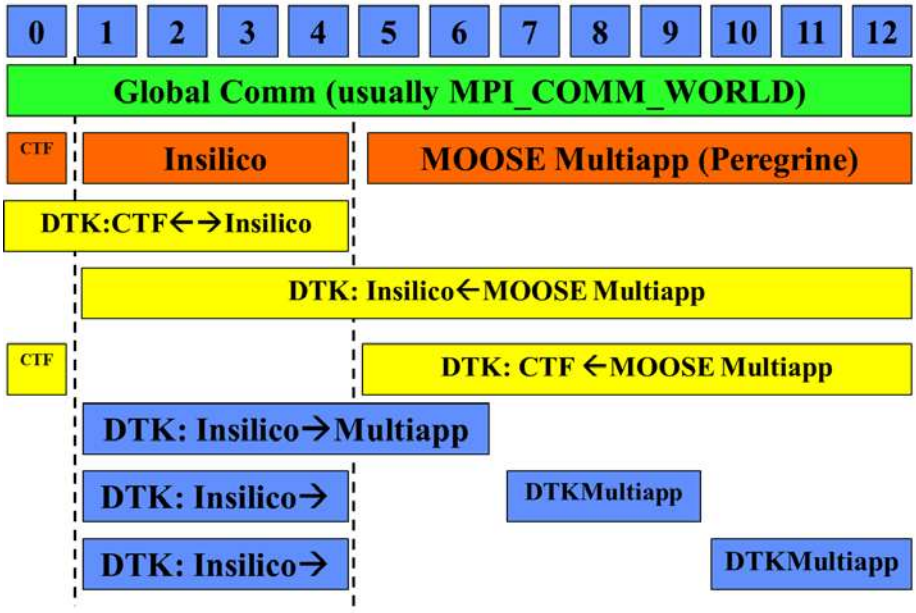


Figure 3-2. Depiction of the MPI communication layers in Tiamat. This example shows 13 communicators in five layers: (1) global communicator (MPI_COMM_WORLD), (2) Application communicators, (3) DTK communicators between CTF, Insilico and Peregrine formed by union of application comms, (4) Peregrine MultiApp communicators for each MultiApp fuel rod formed by subset of Peregrine communicator, (5) DTK transfer comms between Peregrine MultiApp instance and external coupled code (CTF or Insilico) formed by union of MultiApp comm and the external code comm.

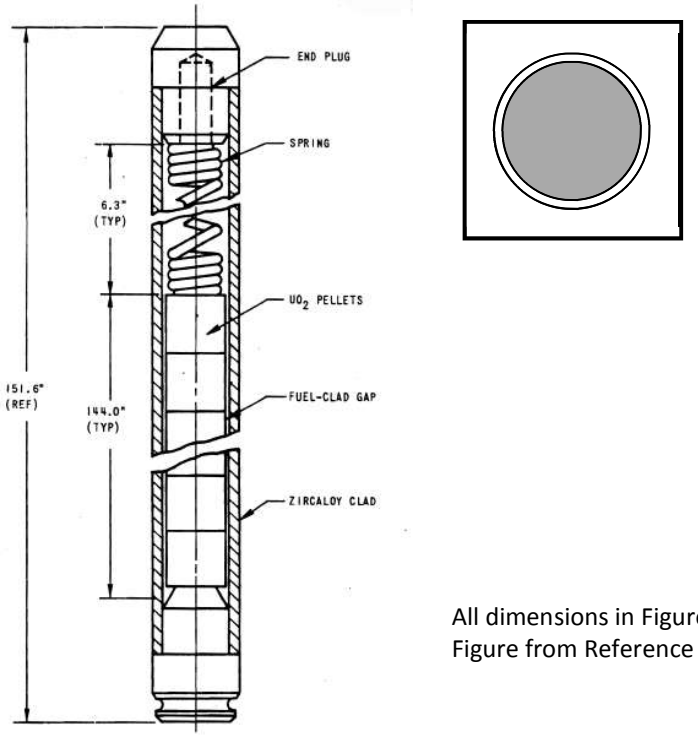
Application distribution and communicator construction is handled via the MultiphysicsDistributor object in Tiamat. Note that this object is not tied to any communicator and can be used to create multiple recursions to build nested hierarchical systems.

4. Test Problem Description

The example problem used in this Milestone is a PWR single assembly based on the dimensions and state conditions of Watts Bar Unit 1 Cycle 1. The dimensions for this problem are identical by AMA Progressive Benchmarks "Problem 3" [1] and "Problem 6". Problem 6 is at Hot Full Power (HFP) and includes T/H feedback. See the milestone report for L3.AMA.VDT.P6.03, "Coupled Single Assembly Solution with VERA (Problem 6)" for more details [13].

The assembly is a standard 17x17 Westinghouse fuel design with uniform fuel enrichment. There are no axial blankets or enrichment zones. The assembly has 264 fuel rods, 24 guide tubes, and a single instrument tube in the center. There are no control rods or removable burnable absorber assemblies in this problem.

The primary geometry specifications of the fuel rod and guide tube materials are given in Figure 4-1 and Table 4-1. The geometry specification for the assembly is given in Figure 4-2 and Table 4-2. For a complete description of the geometry, including spacer grid and nozzle specifications, refer to Reference [1]. The thermal-hydraulic specifications for this problem are shown in Table 4-3.



All dimensions in Figure are in inches
Figure from Reference [14], Figure 4.2-3

Figure 4-1. Fuel Rod Diagram

Table 4-1. Fuel Rod and Guide Tube Descriptions

Parameter	Value	Units
Fuel Pellet Radius	0.4096	cm
Fuel Rod Clad Inner Radius	0.418	cm
Fuel Rod Clad Outer Radius	0.475	cm
Guide Tube Inner Radius	0.561	cm
Guide Tube Outer Radius	0.602	cm
Instrument Tube Inner Radius	0.559	cm
Instrument Tube Outer Radius	0.605	cm
Outside Rod Height	385.10	cm
Fuel Stack Height (active fuel)	365.76	cm
Plenum Height	16.00	cm
End Plug Heights (x2)	1.67	cm
Pellet Material	UO_2	
Clad / Caps / Guide Tube Material	Zircaloy-4	

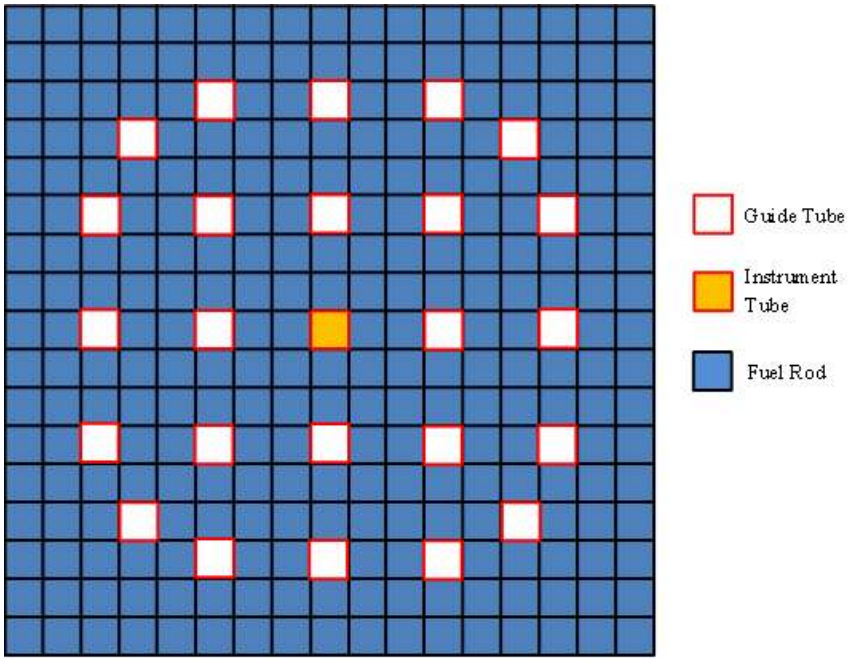


Figure 4-2. Assembly Layout Showing Guide Tubes (GT) and Instrument Tube (IT) placement.

Table 4-2. Assembly Specification

Parameter	Value	Units
Rod Pitch	1.26	cm
Assembly Pitch	21.5	cm
Inter-Assembly Half Gaps	0.04	cm
Geometry	17x17	
Number of Fuel Rods	264	
Number of Guide Tubes (GT)	24	
Number of Instrument Tubes (IT)	1	

Table 4-3. Nominal Thermal-Hydraulic Conditions

Parameter	Value	Units
Inlet Temperature	559	degrees F
System Pressure	2250	psia
Rated Flow (100% flow)	0.6824	Mlb/hr
Rated Power (100% power)	17.67	MWt

5. Results

A simulation of the full assembly 17x17 Benchmark problem 6 was successfully completed. The run was performed using 54 cores of the boris machine at ORNL. The multiphysics distributor assigned one core to CTF, thirty six cores Insilico and the remaining 17 cores to Peregrine.

After ramping to HFP, the Jacobi fixed-point solve took 32 iterations to converge. The entire simulation took 11 hours to complete. This is a fairly high number of iterations and is most likely attributed to the convergence criteria currently being used. A follow on in future PoRs would be a sensitivity analysis of individual code convergence criteria metrics. Both accuracy and efficiency should be explored.

Figures 5-1 and 5-2 show the solution profiles output by the individual application codes.

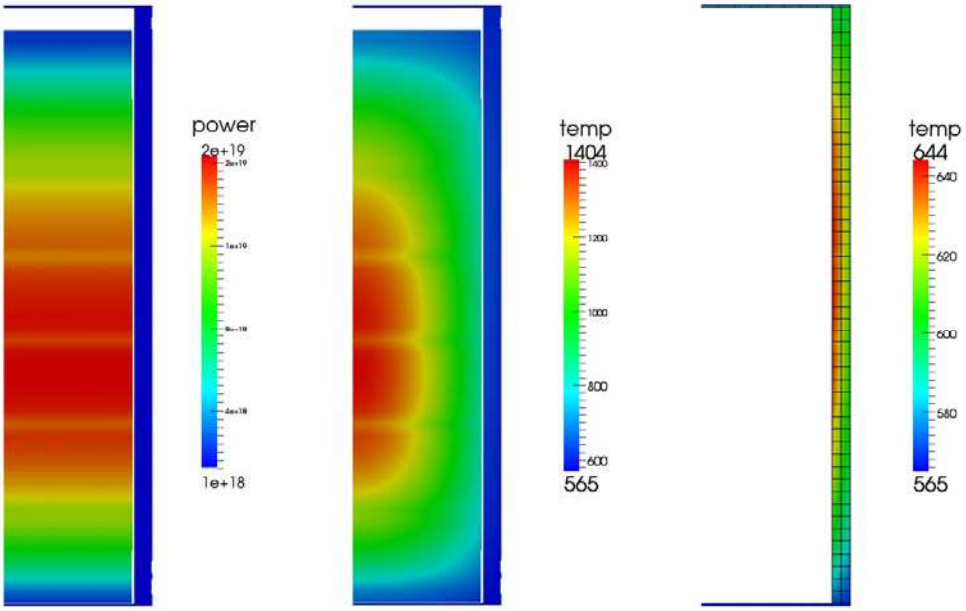


Figure 5-1. Surface plots of fission rate (from Insilico) and temperature in Peregrine for a selected fuel rod in the assembly. The plot on the right is scaled to show clad temperatures.

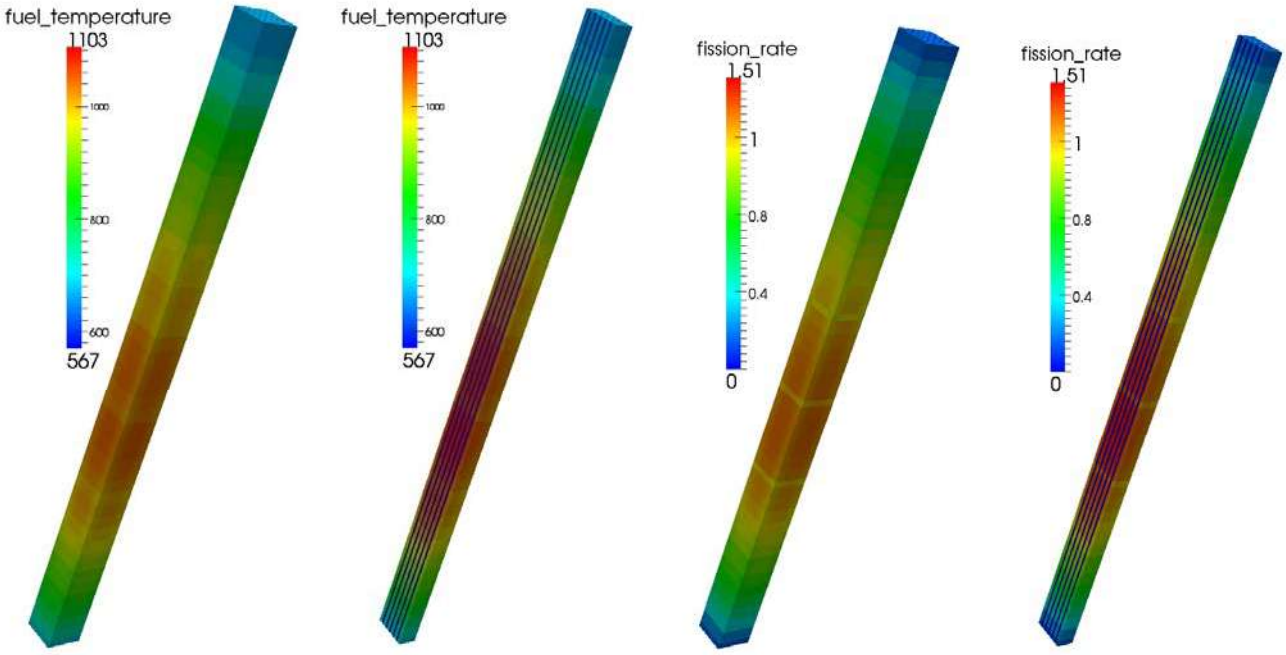


Figure 5-2. Insilico averaged fuel temperature and fission rate.

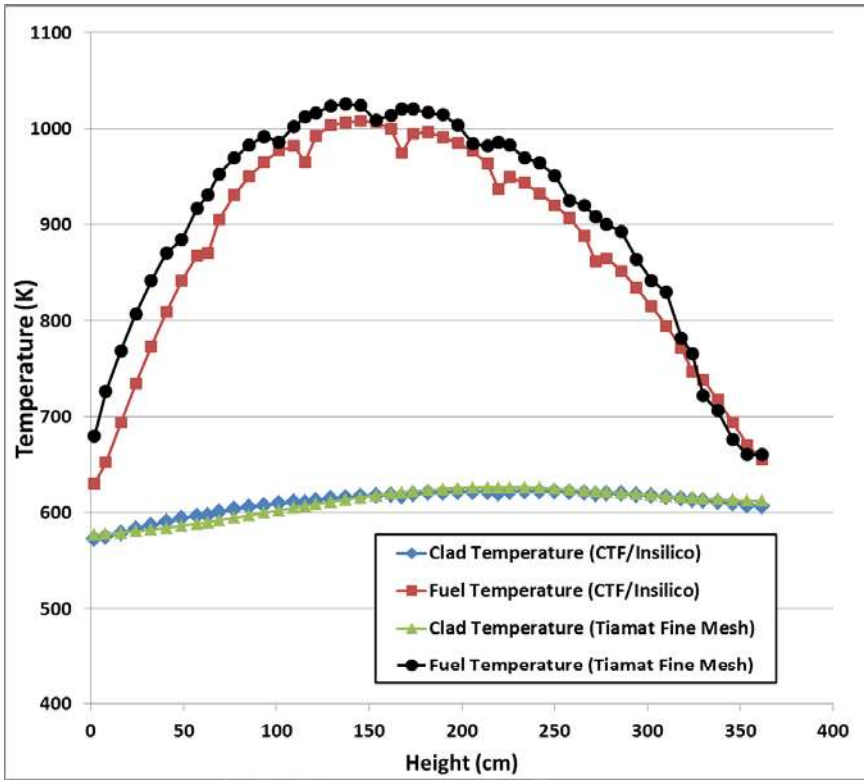


Figure 5-3. Comparison of Tiamat vs. CTF+Insilico coupled capability.

A comparison of the average pin temperature values for the Tiamat simulation compared to CTF+Insilico coupled code is shown in Figure 5-3. Two aspects of this comparison warrant further investigation. One is the fact that fuel temperatures in Tiamat are consistently higher than those generated by the CTF rod model. This could be due to the substantial differences in the rod models themselves, boundary condition treatments, or some combination of effects. A simple test will be to compare simulations without spacer grids, and this will be performed.

In addition, the apparent location of the spacer grids, corresponding to dips in the fuel temperatures, appears to be shifted between the two models. This could be due to indexing differences between the two models, or possibly due to differences in averaging over axial zones, but certainly warrants further investigation.

One point of interest is in the performance solution of the problem. On a coarse Peregrine mesh, the simulation converged quite well. However, when we doubled the number of mesh cells in the axial direction from 24 elements to 48 elements, the simulation encountered difficulty in converging the nonlinear system and many individual pins reported step failures in the MultiApp. To achieve convergence, the Krylov solver paraemeters had to be tightened. Table 5-1 below shows the changes to the Peregrine input file that were made.

Table 5-1. Changes to the Peregrine input file to achieve convergence on the fine mesh.

Solve Parameters	Coarse	Fine
Peregrine Num Axial Elements	24	48
GMRES Tolerance	8.00E-03	1.00E-06
Max Num Krylov Iterations	100	400
Temperature Damping	50	25

Note that this was not an exhaustive study to fine tune the application for performance, rather it allowed for the successful convergence of the demonstration problem. As mentioned earlier, in the next PoR a complete accuracy and sensitivity study on the coupled code convergence should be conducted. Timing statistics were collected for the coarse and fine mesh simulations and are summarized in table 5-2.

Table 5-2. Timing statistics for the coupled 17x17 assembly simulation.

Timings	Coarse	Fine
Total Time (hrs)	3.43	11.03
Initialization time (s)	5534	11360
Solve Time (s)	6344	27920
Num Fixed-point Iterations	15	32
CTF Solve Time (s)	293	579
Insilico Solve Time (s)	5567	20160
Peregrine Solve Time (s)	4068	20880
CTF Time per FP Iteration (s)	20	18
Insilico Time per FP Iteration (s)	371	630
Peregrine Time per FP Iteration (s)	271	653

An important point is that the initialization time was significant compared to the fixed point iteration solve time. Improvements to the Peregrine ramp are being conducted and we expect this time to improve. In moving from the coarse to the fine mesh, the number of fixed point iterations doubled.

Figure 5-4 compares the average temperature of all pins in the assembly for the coarse and fine Peregrine mesh, and indicates that, as expected, the fine mesh captures the effect of the spacer grids with less smearing than the coarse mesh. Further refinement studies are recommended.

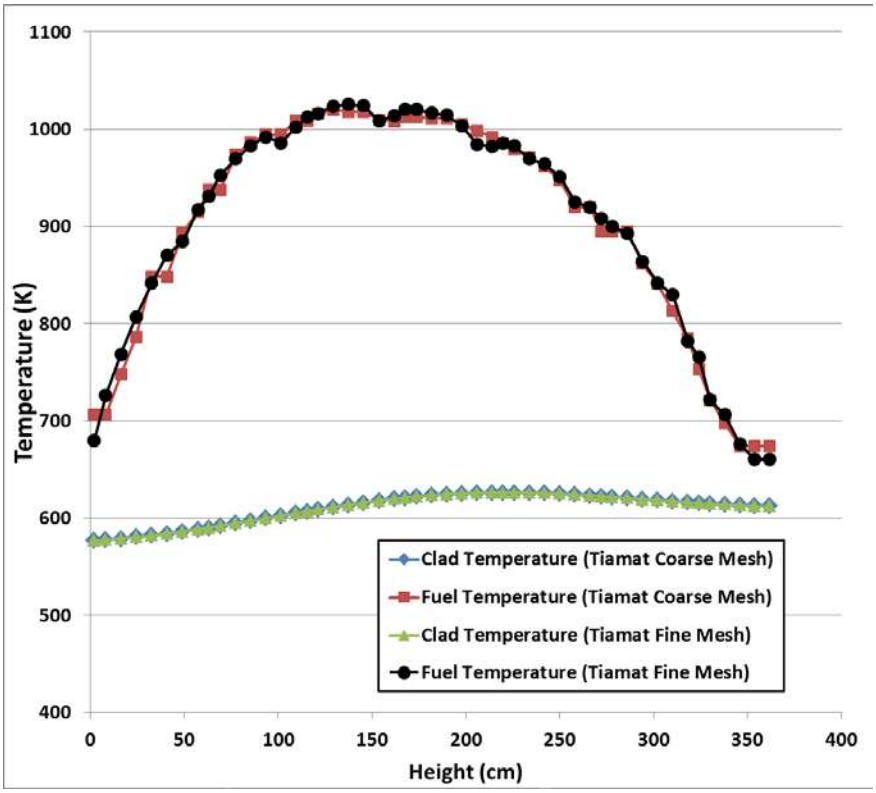


Figure 5-4. Comparison of pin averaged temperatures in assembly for coarse and fine Peregrine mesh.

6. Summary

The milestone successfully integrated Peregrine into VERA-CS and performed an initial demonstration simulation of a 17x17 assembly from Watts Bar cycle 1. Improvements to the TriBITS build system were implemented to support integration, and a new driver (Tiamat) was developed for this capability, and will play a significant role in the evolution of the VERA coupling strategy.

Recommendations for future work include the following. These are not in any particular order of prioritization, and include both physics and numerics tasks as well as software/user-oriented features.

- Investigation of the differences in results between the CTF rod model and Peregrine.
- Further study of the effects of refinement of axial zoning.
- Discussion of path forward for coupling strategy.
- Investigation of numerical behavior and optimal settings for relaxation parameters, convergence criteria, etc.
- Investigate enhancement of MOOSE MultiApp functionality to allow fully implicit solves at each time step (ability to iterate within a time step and back up if another physics component rejects the current time step).
- Investigation of load balancing, i.e. the optimal allocation of cores between components.
- Modifications to MOOSE, Insilico, and model evaluators required to communicate fast neutron flux from Insilico to Peregrine.
- Investigate conservation of data transfers (some of the above modifications are needed prior to this investigation).
- Investigate and improve the current ramping / startup strategy.
- Implementation of a user-controllable ability to select sets of fuel rods to use the CTF rod model or Peregrine.
- Integration of the 3D unstructured-mesh version of Peregrine, also for selected rods or regions.

7. References

[1] A. T. Godfrey, "VERA Core Physics Benchmark Progression Problem Specifications", CASL-U-2012-0131-002, Rev. 2, March 29, 2013.

[2] G.G. Davidson, T.M. Evans, R.N. Slaybaugh, and C.G. Baker, "Massively Parallel Solutions to the k-Eigenvalue Problem," Trans. Am. Nucl. Soc., 103 (2010)

[3] M.N. Avramova. CTF: A Thermal Hydraulic Sub-Channel Code for LWR Transient Analyses, User's Manual, February 2009

[4] R. Schmidt, N. Belcourt, R. Hooper, R. Pawlowski, An Introduction to LIME 1.0 and it's Use in Coupling Codes for Multiphysics Simulation, SAND2011-8524, November, 2011.

[5] S.R. Slattery, P.P.H. Wilson and R.P. Pawlowski, "The Data Transfer Kit: A Geometric Rendezvous-Based Tool for MultiPhysics Data Transfer," M&C 2013 International Conference on Mathematics & Computational Methods Applied to Nuclear Science & Engineering, Sun Valley Idaho, May 5-9, 2013

[6] R. Pawlowski, R. Bartlett, N. Belcourt, R. Hooper, R. Schmidt, A Theory Manual for Multiphysics Code Coupling in LIME Version 1.0, SAND2011-2195, March 2011

[7] T. Evans, A. Stafford, R. Slaybaugh, and K. Clarno, "DENOVO: A New Three-Dimensional Parallel Discrete Ordinates Code in SCALE," Nuclear Technology, 171, 171–200 (2010).

[8] R. Williamson, J. Hales, S. Novascone, M. Tonks, D. Gaston, C. Permann, D. Andrs, and R. Martineau, "Multidimensional multiphysics simulation of nuclear fuel behavior," Journal of Nuclear Materials 423(2012) 149-163.

[9] D. Gaston, C. Newman, G. Hansen, and D. Lebrun-Grandie´. MOOSE: A parallel computational framework for coupled systems of nonlinear equations. Nucl. Eng. Design, 239, p. 1768–1778, 2009.

[10] SCALE: A Comprehensive Modeling and Simulation Suite for Nuclear Safety Analysis and Design, ORNL/TM-2005/39, Version 6.1, Oak Ridge National Laboratory, Oak Ridge, Tennessee, June 2011. Available from Radiation Safety Information Computational Center at Oak Ridge National Laboratory as CCC-785.

[11] R. Montgomery, D. Sunderland, and R. Dunham, "Structural Mechanics Framework for VRI PCI Challenge Problem," L2.MPO.Y1.03, ANA-R-11-0844 Rev. 1, July 2011.

[12] R. Montgomery, et.al., "Peregrine: Validation and Benchmark Evaluation of Integrated Fuel Performance Modeling Using Test Reactor Data and Falcon," L1.CASL.P7.02, July 2013.

[13] S. Palmtag, "Coupled Single Assembly Solution with VERA (Problem 6)," L3.AMA.VDT.P6.03, CASL-U-2013-0150-000, July 25, 2013.

[14] Watts Bar Unit 2 Final Safety Analysis Report (FSAR), Amendment 93, Section 4, ML091400651, April 30, 2009. http://adamswebsearch2.nrc.gov/idmws/ViewDocByAccession.asp?AccessionNumber=ML091400651