COMPOSE-HPC: Software Composition for Extreme Scale Computational Science and Engineering

X-Stack Software Research Program

Project Team

Institution ***(Lead)***	***Lead PI*** *and Co-PIs*	*Email Address*	*Additional Participants*
Galois, Inc.	Matt Sottile	matt@galois.com	
LLNL	Tom Epperly	epperly2@llnl.gov	Tammy Dahlgren, Adrian Prantl
ORNL	**David Bernholdt**	**bernholdtde@ornl.gov**	Wael Elwasif, Samantha Foley
PNNL	Manoj Krishnan	manoj@pnl.gov	Daniel Chavarria
SNL	Rob Armstrong	rob@sandia.gov	Ben Allan, Geoff Hulette

Project web site: http://compose-hpc.sourceforge.net/

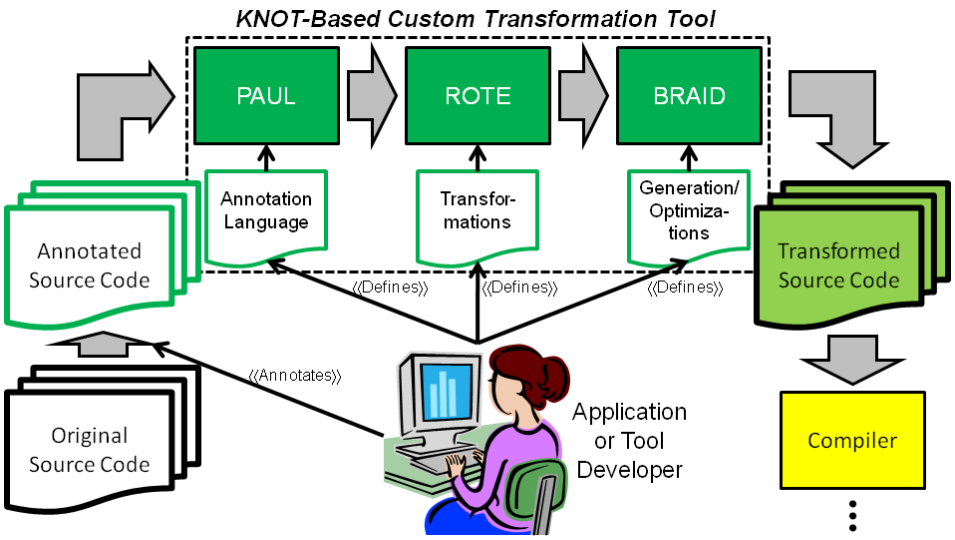
Project Description

The COMPOSE-HPC project seeks to provide tools to help the developers of high-performance computational science and engineering (CSE) software reliably automate code transformations that they need to apply during development. These include transformations needed to compose software in various ways, to evolve it for new hardware architectures, to extend and enhance its scientific capabilities, or simply to improve its quality and performance. Today, such transformations are often performed, quite tediously, by hand. Where tools are used, they are most often simple tools, such as grep, sed, perl or python, which manipulate the code as text, without taking into account the structure and semantics of the programming language being transformed. The transformations of interest are many and varied, but a few examples include:

- Updating calls to a library or utility routine used pervasively throughout a code;
- Changing data structures to improve performance, such as switching between an array of structures (derived types) and a structure containing arrays; or
- Expressing some platform-specific optimizations as transformations of code, applied at build time, as a more sustainable alternative to trying to maintain multiple distinct versions of the code.

The fundamental goal of the COMPOSE-HPC project is to create an environment that allows application developers to easily specify and apply a broad range of transformations to their own codes in a reliable and safe manner. Tools exist today that provide language-aware transformations of source code, but they are either specialized and not extensible, or general but requiring a fairly deep understanding of programming language and compiler concepts. COMPOSE-HPC is developing a tool chain that will provide much of the power of sophisticated language tools in a form that is friendlier to application developers who are domain experts rather than compiler experts. The KNOT toolkit includes three components. PAUL will allow application developers to specify and process "annotation languages" – typically structured comments (directives) that can be included in an application source code to guide the transformation process without effecting how a regular compiler interprets the code. ROTE will allow application developers to specify transformations which respect the language semantics, which can be controlled by source code annotations via PAUL. Finally, BRAID provides code generation and optimization capabilities tailored to the specific needs of transformations.

We also plan to develop a number of specific tools using the KNOT tool chain, both as examples of transformations for others to learn from and as useful tools in their own right. The focus we have chosen



for this level of our work is *software composition*, which we interpret broadly. Scientific application development today involves large teams developing code in a modular fashion, using multiple languages and programming models, and often leveraging third-party libraries. As computer architectures evolve, there is a growing need to compose software modules written for different types of processors (e.g., CPU vs. GPU), adopting different programming models and using various models of parallelism. Developers seeking to utilize leadership-class systems are increasingly pushed toward the composition of multiple parallel tasks as a means to expose the levels of concurrency required for scaling on these machines. All of these forms of software composition pose real, practical challenges for computational scientists, for whom the software is not the end product, but rather a tool for pursuing their scientific research.

Multi-language programming is common in CSE today, and the choices to deal with it have been "hand-rolled" bindings, tools for pair-wise interoperability, and Babel, developed by team members under previous SciDAC projects as a language interoperability tool that treats all supported languages (C, C++, Fortran, Java, and Python) on an equal footing. In the COMPOSE-HPC project, we will use the KNOT tool chain to produce a "Babel 2.0" language interoperability tool which will offer similar functionality to the original Babel along with the ability for much more tailored and highly optimized connections between languages without having to carry along features not needed by the application.

We also plan to study composition problems that reside at a higher level of abstraction. For example, when software from different developers is used together, it is often hard to determine exactly how a routine is meant to be used, how it will behave, and the range of parameters for which it is expected to function correctly. To help remedy this situation, code can be annotated with *contracts* providing "executable documentation" of the expected behavior for both the caller and the callee. These annotations will be transformed automatically into code to verify the contracts.

Finally, we will use KNOT as part of an effort to help applications evolve to more flexible forms of parallelism, which we think of as "composing threads" and "composing concurrency". Today, applications are limited (at execution time) to a particular combination of MPI processes and threads per process. We will develop tools to allow software components with different "threading models" to coexist within an application. We will also develop tools to facilitate multiple-program multiple-data (MPMD) style programming, allowing developers to expose greater levels of concurrency in their applications by allowing independent tasks to be executed simultaneously.

We are very interested in connecting with extreme-scale applications that can make use of the tools and concepts we are developing. If there transformations or related tools not discussed here that would be particularly useful for your project, we are open to discussion.