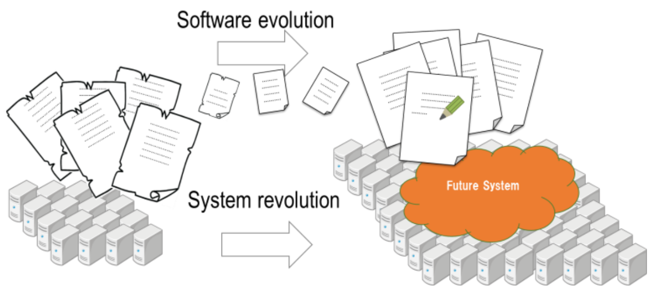
Achieving High Peromance Portabilty across Diverse System Architectures and Generations

Overview

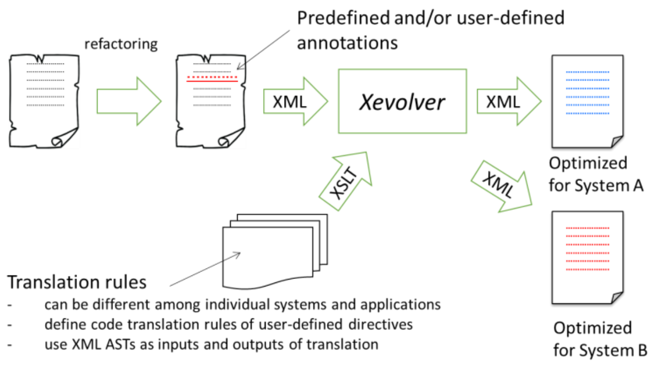
High-performance computing (HPC) system architectures are revolutionarily becoming larger and more heterogeneous. Meanwhile, it is not affordable to rewrite each application for every new system. This Xevolver project takes an evolutionary approach to incremental migration of existing software resources to new systems. The goal of this project is to establish an effective migration path to new algorithms, implementation schemes, and programming environments for massively-parallel and heterogeneous systems in an upcoming extreme scale computing era.



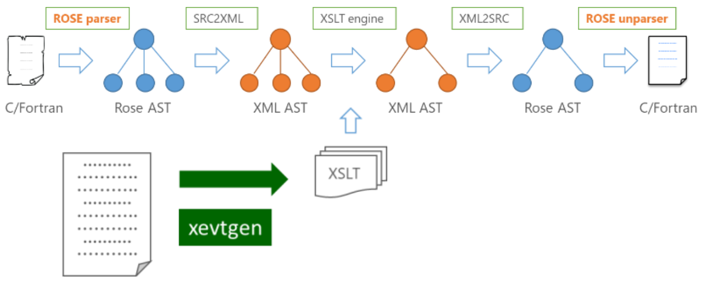
System revolution and software evolution

Separation of System-Awareness

HPC system architectures are getting more complicated and diversified. Due to the complexity, it becomes harder and harder to exploit the full potential of a particular system without performance optimizations specific to the system. That is, an application code must be thoroughly optimized and specialized for a specific platform to achieve high performance. The diversity of system architectures increases the number of system architectures that have to be considered during the life of an application. Accordingly, increases in system complexity and diversity would force programmers to further invest enormous time and effort for HPC application development and maintenance. To overcome this difficulty, we are developing an extensible code transformation framework, Xevolver, so that users can define their own code transformation rules for special demands of individual systems and individual applications. As a result, users can express the information about system-specific performance optimizations separately from application codes and thereby


refactoring
Predefined and/or user-defined annotations
XML
Xevolver
XML
Optimized for System A
XSLT
XML
Optimized for System B
Translation rules
- can be different among individual systems and applications
- define code translation rules of user-defined directives
- use XML ASTs as inputs and outputs of translation

User-defined code transformation



Code transformation rule generation.

facilitate HPC application migration in the future.

Xevolver Framework

In practice, there are repetitive patterns in the code modifications, and hence we can assume that those code modifications could be replaced with a smaller number of code transformations. Under this assumption, we are developing Xevolver to enable users to express their own code optimizations for special demands of individual systems and individual applications. Instead of simply modifying a code by hand, users can easily define custom code transformations to optimize and specialize an application code for a particular system. In Xevolver, such code transformation rules can be defined separately from an application code. Accordingly, Xevolver enables to express system-specific and/or application-specific code optimizations separately from application codes.

To define a user-defined code transformation rule, what users have to do is to simply write two version of a code, or a code pattern; the original version and its transformed version. Then,

Xevolver: XML-based AST transformation framework.
C/Fortran programs are converted to their ASTs in XML, and exposed to programmers for AST transformations. The transformed ASTs are converted back to C/Fortran programs.

Xevparser/Xevtgen: Tools for generating code transformation rules.
A code transformation rule in XSLT is generated from a simple rule template written in Fortran.

autoOMP: Automatic OpenMP insertion tool.
OpenMP directives are automatically inserted based on compiler's optimization messages.

Xev-GMP: Automatic code generation of GMP multiple-precision code from C code.

FFTE: A fast Fourier transform package for GPU clusters.

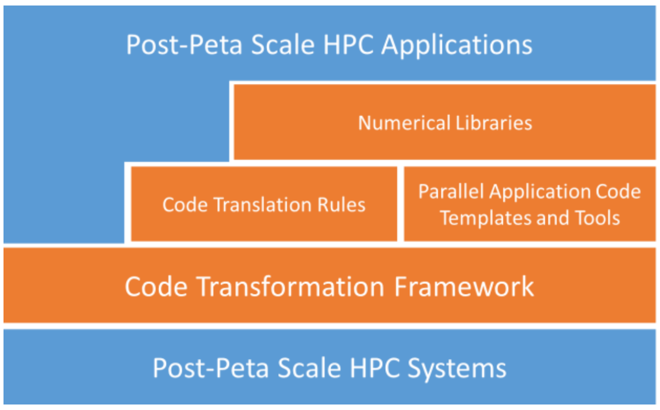
AMGS: An algebraic multigrid library.

FXTPACK: Fast orthogonal function transform library.

PACC: A directive-based programming framework for accelerating large-scale stencil computation.

All the software packages are available at http://xev.sc.cc.tohoku.ac.jp.

one of our tools named Xevtgen generates a machine-usable code transformation rule from such a simple description about the code transformation. Therefore, users do not need to care about any special knowledge internally required for implementing their code transformations.


Post-Peta Scale HPC Applications
Numerical Libraries
Code Translation Rules
Parallel Application Code Templates and Tools
Code Transformation Framework
Post-Peta Scale HPC Systems

Hierarchical abstraction of system configuration.

Hierarchical abstraction of HPC systems

We do not claim that everything for performance optimization should be expressed as user-defined code transformations. Rather, our claim is that appropriate abstractions should be used for appropriate purposes. User-defined code transformations should be used to express code modifications that are unavoidable even if all the abstractions are properly used.

Abstraction technologies such as numerical libraries are strongly required to hide complicated system configurations from application developers. We are developing numerical libraries to hierarchically abstract HPC system configurations. Our numerical libraries are optimized for multiple platforms, such as GPU, MIC, and CPU cluster systems, so that application developers can use different implementations with common interfaces, resulting in high performance portability. The numerical libraries are designed to support as many data structures as possible to cover various use cases while achieving high performance. We also investigate auto-tuning technologies to adapt the optimized implementations of numerical libraries to similar platforms in order to achieve high performance portability.

HPC refactoring catalog

In this project, we are cataloging expert knowledge and experiences about code optimizations as the HPC refactoring catalog. The HPC refactoring catalog is open to the public (https://one.sc.cc.tohoku.ac.jp/hpcref/) so that programmers can share and reuse the knowledge and experiences. Description about the code optimization, code examples, and performance evaluation results are provided in the catalog, and code transformation rules are also provided for some important code optimization techniques.

Project information:
Project leader: Hiroyuki Takizawa (Tohoku Univ.)
Project member:
Reiji Suda (Univ. Tokyo),
Daisuke Takahashi (Univ. Tsukuba),
Ryusuke Egawa (Tohoku Univ.)
Contact: **takizawa@tohoku.ac.jp**