Xevolver - User-defined code transformation framework (1)

XML-based AST (Abstract Syntax Tree) Transformation
- Transformation is what compiles internally done for transforming a code.
- Extensible Markup Language (XML) is well-established for describing tree data.
- Extensible Stylesheet Language Transformations (XSLT) is a standard XML data format to express XML data conversion.

With separate transformation rules, users can maintain the original code and transform it in various ways instead of directly modifying it.

XML-based AST (Abstract Syntax Tree) Transformation
- Transformation is what compiles internally done for transforming a code.
- Extensible Markup Language (XML) is well-established for describing tree data.
- Extensible Stylesheet Language Transformations (XSLT) is a standard XML data format to express XML data conversion.

With separate transformation rules, users can maintain the original code and transform it in various ways instead of directly modifying it.

Evaluation and Discussions

Combining Auto-tuning and User-defined Code Transformation

Xevolver with Xevolver is used to transform the Himeho benchmark to its auto-tunable version.

An auto-tuning framework, OpenTuner, is used to determine the parameters of the auto-tunable Himeho benchmark.

The performance quickly improves as the auto-tuning process proceeds. Application developers can enjoy this performance gain while hiding the code complexity behind code transformations defined by performance engineers.

This separation of concerns is very helpful to achieve an appropriate division of labor between those two kinds of programmers.

Xevolver offers an easy way of defining custom transformation rules.

In this case study, Xevolver generates transformation rules from XSLT rules of Fortran code patterns. The most common way to tell people about code transformation is to show the original and transformed versions of a code.

Xevolver is one of Xevolver utility tools to generate XSLT rules from Fortran code patterns that define rules by the target and transformed versions of a code portion.

Code patterns are written in Fortran syntax with several directives only to define a code transformation. It is a kind of dummy code because it does not need to represent a meaningful computation, but a transformation rule.

Some parts of the code patterns could be represented as special "Tagen" variables so that the transformation rules are applied not only to the specific code but also a certain code pattern (see Rule 2).

Conclusions

This poster shows a case study of using user-defined code transformation for making an existing code auto-tunable. For auto-tuning, application-specific coding techniques are often required especially for legacy codes. By using custom code transformation, application developers do not need to directly modify their codes themselves. As a result, they can enjoy the benefit of auto-tuning without degrading the code maintainability.

Acknowledgments
The authors would like to thank Profs. Riji Sato, Daisuke Takahashi, Ryusuke Egawa, and Kazuhiko Komatsu for valuable discussions on this project.

This work was partially supported by JST CREST “An Evolutionary Approach to Construction of a Software Development Environment for Massively Parallel Heterogeneous Systems” and Grant-in-Aid for Scientific Research (B) 16H02822.

The Xevolver framework is online available at http://xevolver.is.tohoku.ac.jp.