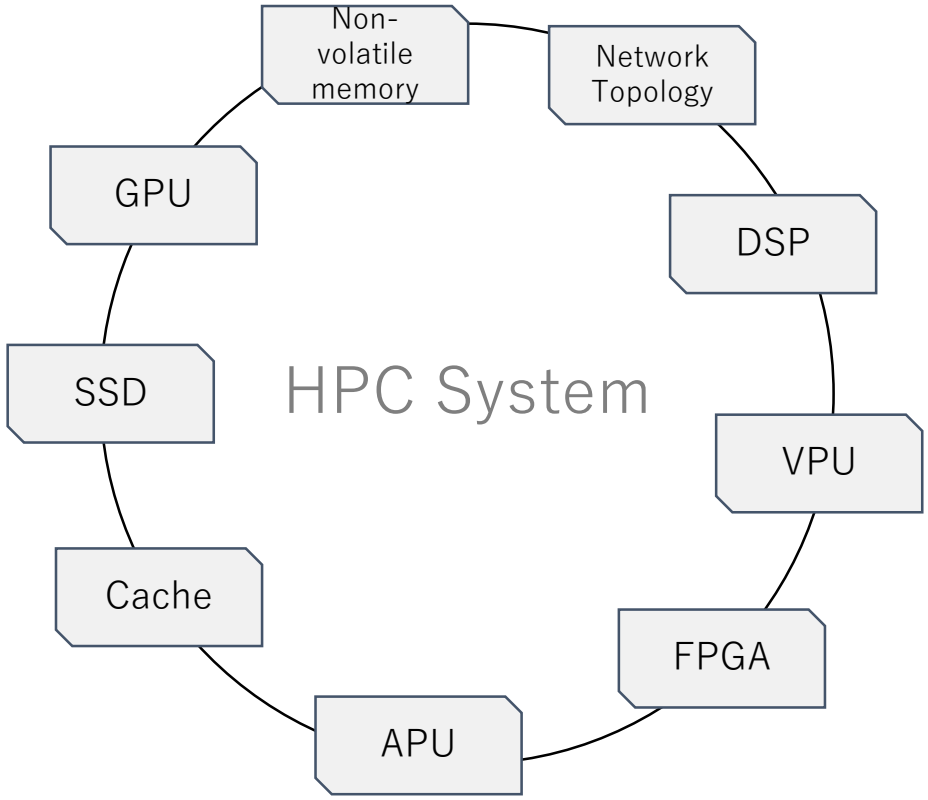
Systematic Generation of Optimized Codes of Stencil Computation for HPC System with a Hierarchical Structure

The University of Tokyo

Osamu Ishimura



Do you like making DSL platforms from scratch?

Problem & Contribution

Problem

- Recent tendency of the complication of the structure of HPC system (GPU, many-core processor, FPGA, DSP, etc.) makes it difficult to build efficient applications.
- Domain-specific language (DSL) platform is a promising approach. However, these platforms are not portable by themselves, and the difficulty of development is just transferred to the lower layers.

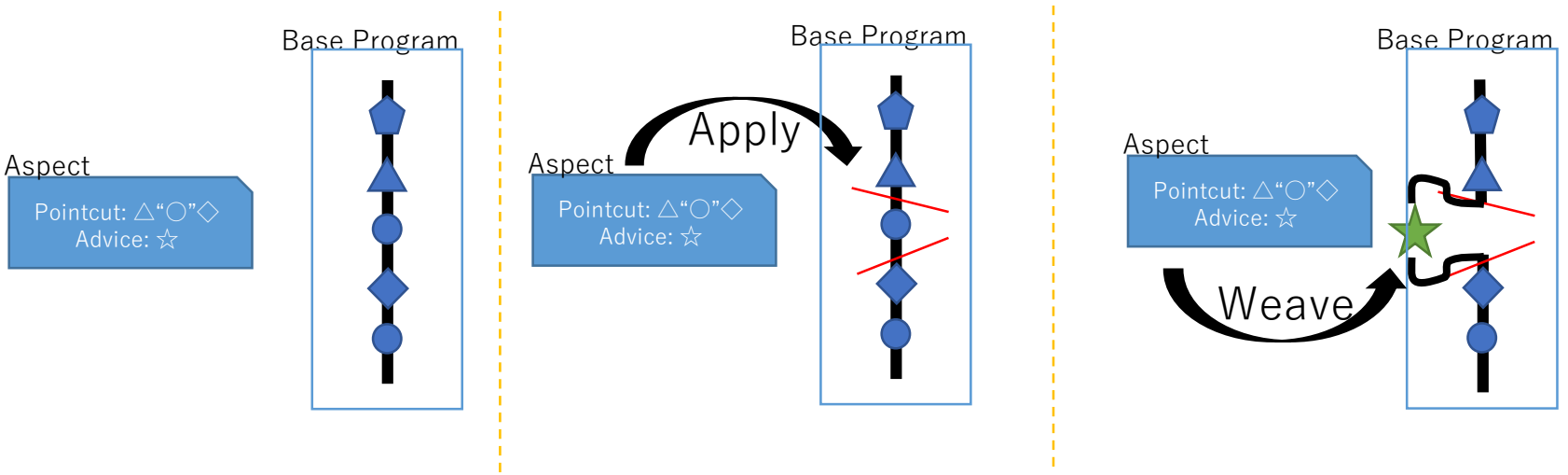
Contribution

- Designed the programming model to separate program adaptations for systems from program logic by corresponding hardware hierarchies and self-similarity of region segmentation of stencil programs.
- Developed the program adapters and optimizers, which are modularized and portable; on Aspect-Oriented Programming.
- It enables to separate platform developments and platform turning for systems.

Aspect Oriented Programming

In AOP, Cross-cutting Concern in Objects in OOP is extracted to Aspects

- In Joint-Point Model, Aspect has Pointcuts and Advices.
- Pointcuts extract Joint-Points by Pattern Matchings from the base program and weave the Advices.



Structure of the Platform

Platform

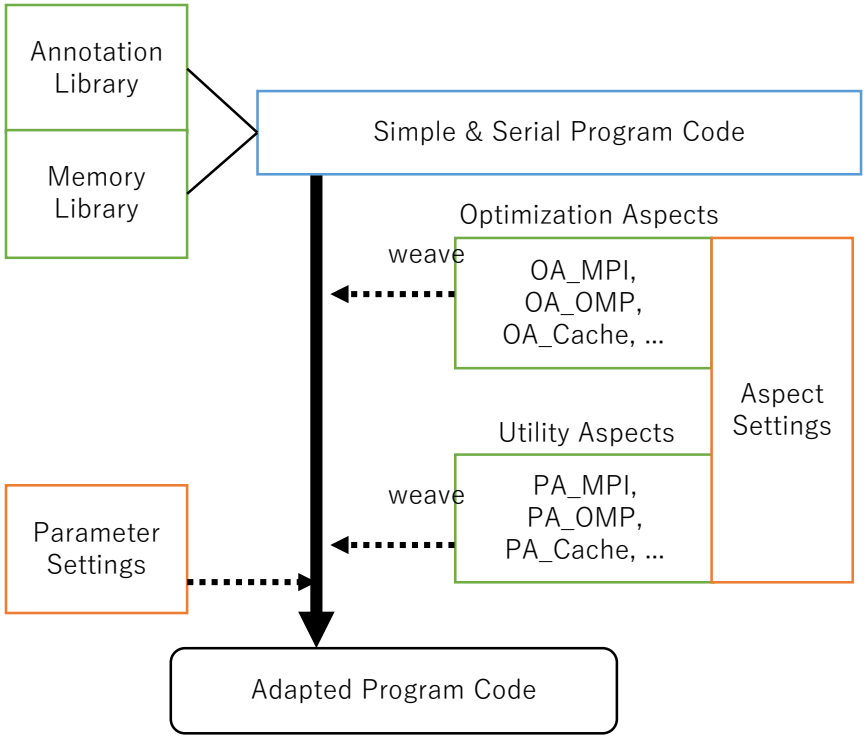
- Annotation Library
- Memory Library
- Optimization Aspects
- Utility Aspects

Providers

- Parameter and Aspects Setting for specific HPC Systems

End-Users

- Serial Program Codes of Their Applications



Evaluation

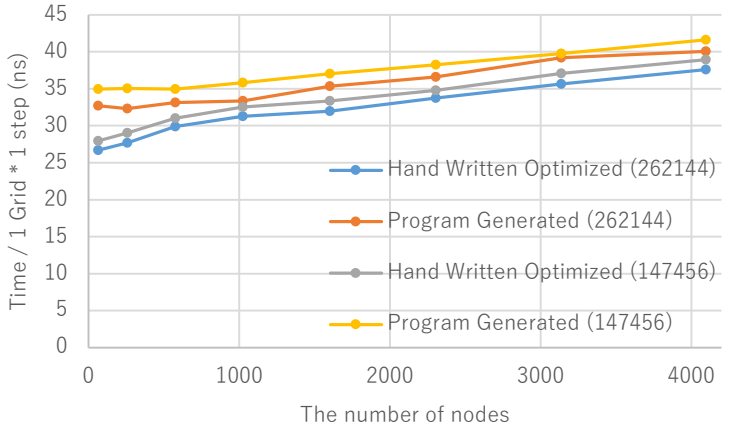


Fig. 1: Comparison between Hand Written and Program Generated on Oakforest-PACS

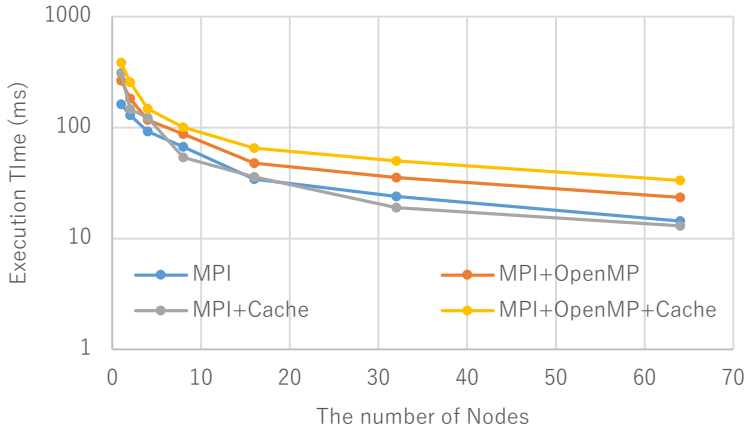


Fig 2: Comparison among Aspect Combinations on Reedbush-U

Results

- Fig. 1 shows the weak scaling performance of the platform. It indicates that our platform has a little size of overhead.
- Fig. 2 shows the strong scaling performance of the platform. It indicates the platform has some performance issue related to the OpenMP Layer.

Future work

- Generalize the programming model to programs with a similarity
- Enhance the AOP compiler or change over to a macro-system based compiler (Branch the process depends on the data type or interface type and provide features to know the logic of a function from aspects. (Naive AOP implementation does not have these feature. It will enable to adapt the system to a no layer-by-layer hierarchy.)
- Change the grain size of kernel to adapt to GPU and SIMD architecture.