ISSN 2186-7437

NII Shonan Meeting Report

No. 2014-7

Staging and High-Performance Computing Theory and Practice

Oleg Kiselyov
Yukiyoshi Kameyama
Jeremy Siek

May 27–30, 2014

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-Ku, Tokyo, Japan

Staging and High-Performance Computing Theory and Practice

Organizers:
Oleg Kiselyov (University of Tsukuba, Japan)
Yukiyoshi Kameyama (University of Tsukuba, Japan)
Jeremy Siek (Indiana University, USA)

May 27–30, 2014

We have conducted the follow-up to the meeting "Bridging the theory of staged programming languages and the practice of high-performance computing" (Shonan Seminar 19), which took place in May 2012. In the first meeting, researchers in staged programming languages were learning of the problems in high-performance computing (HPC) from HPC experts. Since then, thanks in part to Shonan Challenges put forward at that meeting, the theory and tools of staged programming languages have progressed to the point where they could already be useful in HPC practice. The present follow-up meeting has gauged this readiness and set further milestones. It served as a forum for staged programming researchers to present their progress and for HPC practitioners to evaluate it, fostering the collaboration on real-life applications.

Generative programming, in particular, in the form of *staging*, is widely recognized in HPC as the leading approach to resolve the conflict between high-performance on one hand, and portability and maintainability on the other hand. In its general form, staging is an implementation technique for efficient domain-specific languages (DSL), letting HPC experts conveniently express their domain-specific knowledge and optimization heuristics. However, the results and tools of the current staging research are little used in the HPC community. Partly this is because HPC practitioners are not aware of the progress in staging; staging researchers are likewise unaware of what HPC practitioners really need. The first Shonan meeting brought together HPC practitioners and programming language (PL) researchers to break this awareness barrier. The meeting aimed to solicit and discuss real-world applications of assured code generation in HPC that would drive PL research in meta-programming.

The first Shonan meeting succeeded in its aim. It developed a set of benchmarks, representative HPC examples, where staging could be of help in producing more maintainable code and letting domain experts perform modifications at a higher-level of abstraction. This set was dubbed 'Shonan Challenge'.

Shonan Challenge has greatly stimulated research and development of staging, resulting in extensible compilers based on staging (Rompf et al., POPL 2013) and the revival of MetaOCaml (ML 2013). The answers to Shonan Challenges have been presented in the overview paper (Aktemur et el., PEPM 2013), in (Rompf et al., POPL 2013) and in a poster at APLAS 2012. Shonan Challenge problems (specifically, the Hidden-Markov Model benchmark) were discussed at

the staging tutorials given in 2013 at the premier PL conferences PLDI, ECOOP, and ICFP/CUFP.

The present follow-up meeting was intended as the place to report the progress in staging back to the HPC practitioners who posed the challenges, to evaluate how well the developed tools meet the needs of the HPC practice already, and what is yet to be done.

As before, the workshop participants consisted of three groups of people: PL theorists, HPC researchers, and PL-HPC intermediaries (that is, people who are working with HPC professionals, translating insights from PL theory to HPC practice). To promote the mutual understanding, we have planned for the workshop to have lots of time for discussion. We emphasized tutorial, brainstorming and working-group sessions rather than mere conference-like presentations.

The following people have participated in the seminar, beside the organizers.

1. Baris Aktemur, Ozyegin University (Turkey)
2. Nada Amin, EPFL (Switzerland)
3. Kenichi Asai, Ochanomizu University (Japan)
4. Sven Bodo-Scholz, Heriot-Watt University (Scotland)
5. Zach DeVito, Stanford University (USA)
6. Lindsay Errington, Galois, Inc. (USA)
7. Robert Glück, University of Copenhagen (Denmark)
8. Tobias Grosser, INRIA & ENS (France)
9. Jun Inoue, ENS, Paris (France)
10. Frédéric Loulergue, University of Orléans (LIFO)
11. Takayuki Muranushi, Kyoto University (Japan)
12. Ryan RhodesNewton, Indiana University (USA)
13. Georg Ofenbeck, ETH Zürich (Switzerland)
14. Jonathan Ragan-Kelley, Computer Graphics Group, CSAIL, MIT (USA)
15. Tiark Rompf, Oracle Labs & EPFL (Switzerland)
16. Reiji Suda, University of Tokyo (Japan)
17. Daisuke Takahashi, University of Tsukuba (Japan)
18. Jeremy Yallop, OCaml Labs, Cambridge (UK)

Main Questions

The following questions were raised repeatedly during the seminar:

- What exactly is a DSL and its domain? Can we view staging as a DSL for an optimizing compilation?
- How to debug DSLs? When a DSL program generates code that produces an unexpected answer (e.g., because the user misunderstood the problem or the DSL), what does one do? Run-time system bugs (esp. subtle synchronization bugs) are especially challenging.
- How to spread the good news about metaprogramming throughout industry and education? We need to distill and teach the design patterns and document success stories.

Tangible Outcomes

The participants agreed to continue the development of Shonan challenges, adding more challenges and solving them. We need a set of small problems culminating in a grand challenge. Google Docs, associated with the Google group created after the previous meeting, is a good place for the collaborative development of Shonan Challenges.

We should make Shonan Challenges more well-known, especially in the HPC community. A good first step is to write an article for ACM Queue.

Meeting Schedule

May 27 (Tuesday)

Theme: Introductions, background

- Self-introductions (until lunch)
- Daisuke Takahashi *Performance Tuning for High Performance Computing Applications (Overview of HPC)*
- Baris Aktemur *Shonan Challenges*
- Yukiyoshi Kameyama *An Innocent Solution to Shonan Challenge 2012*
- Nada Amin *Scala/LMS/Delight*
- Tiark Romp *Scala answer to Shonan challenges*
- Final discussion

May 28 (Wednesday)

Theme: Control, interactivity, feedback

- Remaining three self-introductions
- Reiji Suda *More challenges for HPC program generation*

- Georg Ofenbeck *Staging for Spiral: What is the gain?*
- Sven Bodo-Scholz *Embedding application knowledge for improved dynamic adaptation*
- Jeremy Siek (leading discussion)
  - Jonathan Ragan-Kelley *Halide: A language and compiler for image processing pipelines*
  - Ryan Newton *DSL embedding how do we know if its worth it?*
- Oleg Kiselyov *You can do it: Solving Shonan Challenge 1 in 7 easy steps*

May 29 (Thursday)

Theme: more challenges

- Zach DeVito *Terra*
- Jonathan Ragan-Kelley *2D Stencil challenges*
- Baris Aktemur *Specializing Sparse Matrix-Vector Multiplication*
- Takayuki Muranushi *Code Generation Challenge for Staggered Mesh*
- Discussion about Shonan Challenges
- Excursion and Banquet

May 30 (Friday)

Theme: Conclusions

- Robert Glück *On the Mechanics of Program-Generator Generators*
- Group discussion: New Shonan challenges. How to continue

Overview of Talks

Specializing Sparse Matrix-Vector Multiplication

Baris Aktemur, Ozyegin University, Turkey

Sparse matrix-vector multiplication (SpMV) is a kernel operation used in many scientific computation domains. SpMV is amenable to optimization by using generative programming to specialize the function based on the structure and values of the sparse matrix. In this work we evaluate the performance gains obtainable from specialized SpMV by generating code using several different methods. We then discuss how to predict which code generation method will give the best performance. We finally share our experiences regarding generating high-performant SpMV code quickly.

On the Mechanics of a Program-Generator Generator

Robert Glück, University of Copenhagen, Denmark

We present principles and implementation behind a light-weight program-generator generator based on partial evaluation techniques for a recursive flowchart language.

Code Generation Challenge for Staggered Mesh

Takayuki Muranushi, Kyoto University, Japan

In ordered-mesh based solvers of partial differential equations, every discretized physical values lie on integer coordinates. Staggered mesh is where rational-number coordinates such as half-integers are allowed. For example in hydrodynamics simulations, gas density variables may lie in the center of the cubic cells while gas flux variables may sit on the center of the cell surfaces. Staggered meshes are sometimes consequences of higher-order differentiation, and sometimes important for numerical stability or conservation laws. On the other hand, staggered mesh algorithms require slightly different treatment for different variables or different vector components. This poses challenge to abstraction, and the failure of abstraction often results in Fortran or C code where x-component logic is copied and slightly modified for y-component and then z-component, and so on.

In my talk I'll illustrate this staggered mesh problem code by taking elastic wave equations as an example.

DSL embedding - how do we know if it's worth it?

Ryan Newton, Indiana University, USA

The embedded DSL approach has established benefits but also suffers from problems that do not necessarily show up in code figures of academic papers: e.g. terrible error messages and other limitations of syntax overloading. In this talk I will argue against the the traditional cost-benefit analysis ascribed to embedded/standalone DSL design decision. To illustrate, I will discuss three hypothetical alternative front-ends to the Accelerate GPU DSL: Haskell-embedded (the original), standalone with language subsetting, and Racket-embedded with its own type system.

An Innocent Solution to Shonan Challenge 2012

Yukiyoshi Kameyama, University of Tsukuba, Japan

The first Shonan Challenge collects a set of problems for generating optimized programs in the domain of high performance computing. Even though several solutions have been already posted by programming-language experts, it was not clear how a non-expert can solve the problems. In this talk I will report a student's solution to the Hidden Markov Model (HMM) problem, and how he developed his solution to accommodate several optimizations. I also briefly mention the evaluation of his solution and the lessons learned. This talk is based on the work by Haruki Shimizu.

Embedding Application Knowledge for Improved Dynamic Adaptation

Sven-Bodo Scholz, Heriot-Watt University, Scotland

Generating efficient code for a range of heterogeneous hardware from DSLs is hard. Trying to achieve the same for Not-So-Domain-Specific-Languages is even harder. This talk presents some of our latest ideas on how to provide the compiler/ code generator with domain knowledge without modifying the code generator at all. Instead, we use programmer-provided alternatives in conjunction with dynamic adaptation to achieve overall efficiency. This is very much work in progress; besides presenting the motivation and basics of the approach it provides several open questions.

More challenges for HPC program generation

Reiji Suda, University of Tokyo, Japan

I introduce several optimizations (program transformations) that are frequently used by HPC programmers. I present several questions for staging.

Performance Tuning for High Performance Computing Applications

Daisuke Takahashi, University of Tsukuba, Japan

- Performance development of supercomputers
- HPC Challenge (HPCC) Benchmark Suite
- It's all bandwidth
- Performance tuning and program optimization methods

Staging for Spiral: What is the gain?

Georg Ofenbeck, ETH Zürich, Switzerland

Can we utilize modern program language features for the translation between DSLs, DSL rewrites? Can we also perform abstraction over low level transformations and data layouts?