Habanero-Java: Multicore Programming for the Masses

PPoPP 2014 Tutorial

Instructors:

Shams Imam, Vivek Sarkar
Department of Computer Science
Rice University
Houston, TX, USA

Abstract:

Habanero-Java (HJ) is a pedagogic parallel programming model being developed at Rice University. It includes a powerful set of task-parallel programming constructs that can be added as simple extensions to standard Java programs to take advantage of current and future multicore and heterogeneous architectures. Past implementations of HJ required language extensions to Java with special compiler support. In this tutorial, we will focus on the new library implementation of HJ called HJ-lib that can be used with any standard Java 8 implementation. HJ integrates a wide range of parallel programming constructs (e.g., async tasks, futures, data-driven tasks, forall, barriers, phasers, transactions, actors) in a single programming model that enables unique combinations of these constructs (e.g., nested combinations of task and actor parallelism). The orthogonal classes of parallel constructs enables programmers with a basic knowledge of Java to get started quickly with expressing a wide range of parallel patterns. HJ is capable of expressing many different forms of parallel patterns including data parallelism, pipeline parallelism, stream parallelism, loop parallelism, and divide-and-conquer parallelism.

HJ-lib puts a particular emphasis on the usability and safety of parallel constructs. For example, no HJ program using async, finish, isolated, and phaser constructs can create a logical deadlock cycle. Further, the future and data-driven task extensions to the async construct facilitate a functional approach to parallel programming. Finally, any HJ program written with async, finish, and phaser constructs that is data-race free is guaranteed to also be internally and externally deterministic.

HJ-lib is built using lambda expressions and can run on any Java 8 JVM. Older JVMs can be targeted by relying on external bytecode transformations tools for compatibility. The HJ runtime is responsible for orchestrating the creation, execution, and termination of HJ tasks, and features both work-sharing and work-stealing schedulers. HJ is used at Rice University as an introductory parallel programming language for second-year undergraduate students. A wide variety of benchmarks have been ported to HJ, including a full application that was originally written in Fortran 90.

The specific topics to be covered in the tutorial include:

- Fork-Join style computations using async-finish constructs.
- Futures and Data-Driven Futures for asynchronous computations.
- Weak atomic constructs using isolated.
- Actors for event-driven style computations.

This tutorial should be relevant to practitioners and researchers interested in enabling software to execute efficiently on parallel computers using higher-level parallel extensions to Java than threads and locks. All attendees will leave the tutorial with sufficient information to get started on writing HJ programs on their own. In addition, the semantic foundations and implementation techniques described in the tutorial will be useful for language designers and implementers (who may be interested in implementing similar parallel extensions in other serial languages or on other target architectures), as well as to application and library programmers (who may be interested in building robust parallel software frameworks on top of HJ-lib). Being a pedagogic programming model, HJ-lib is also an attractive tool for educators with numerous educational resources available from the sophomore-level COMP 322 course offered at Rice University.

The prerequisite knowledge assumed is familiarity with the foundations of sequential programming and with the basics of object-oriented programming as embodied in Java or C++. No parallel programming experience is assumed as a prerequisite.

Installation Instructions:

We welcome tutorial attendees to pre-install HJ-lib, so that they can try out hands-on examples during the tutorial. Installation instructions are available in the Download and Set Up page. API Documentation and code examples are also available online.

Software dependencies: Please ensure that your system has Java 1.8 (or higher) installed for full functionality. The Download and Set Up page includes download instructions for Java 8 JDK and the IntelliJ IDE which supports Java 8. Any future Eclipse version that supports the full Java 8 language can also be used.

Online Resources:

1. PPoPP Tutorial Slides
2. Sample Project with Examples from PPoPP Tutorial
3. API Documentation and code examples
4. Rice University's COMP 322 course materials.
5. Videos in edX site for COMP 322 (send email to Shams Imam and Vivek Sarkar if you have access problems).

Biography:

Shams Imam is a fifth-year graduate student in the Department of Computer Science at Rice University working under Prof. Vivek Sarkar in the Habanero Multicore Software Research Project. His research interests mostly include Parallel Programming Models and Runtime Systems with the aim to make writing task parallel programs on multicore machines easier for programmers. His currently work involves building a generic framework that efficiently supports all synchronization patterns, not only those available in actors or the fork-join model, in task parallel programs. His previous research includes work on developing an efficient implementation of a programming model integrating both Actors and Fork-Join parallel models. At Rice, he is currently involved in building a cooperative runtime for Habanero-Java. Previously, he has been involved in projects such as Habanero-Scala, CnC-Scala, CnC-Matlab, and CnC-Python.

Vivek Sarkar is the E.D. Butcher Professor of Computer Science at Rice University. He conducts research in programming languages, program analysis, compiler optimizations and virtual machines for parallel and high performance computer systems, and currently leads the Habanero Multicore Software Research project at Rice (www.habanero.rice.edu). Prior to joining Rice, he was Senior Manager of Programming Technologies at IBM Research. His past projects at IBM include the X10 programming language, the Jikes Research Virtual Machine for the Java language, the ASTI optimizer used in IBM's XL Fortran product compilers, the PTRAN automatic parallelization system, and profile-directed partitioning and scheduling of Sisal programs. In 1997, he was on sabbatical as a visiting associate professor at MIT, where he was a founding member of the MIT RAW project. Vivek holds a B.Tech. degree from the Indian Institute of Technology, Kanpur, an M.S. degree from University of Wisconsin-Madison, and a Ph.D. from Stanford University. Vivek was elected to the IBM Academy of Technology in 1995, and inducted as an ACM Fellow in 2008. He has given tutorials at several past conferences including PLDI 1993, POPL 1996, ASPLOS 1996, PLDI 2000, OOPSLA 2003, ECOOP 2004, OOPSLA 2006, PPoPP 2007, PLDI 2007, PLDI 2008, and has also taught many short courses and full-length courses.