JFP **24** (6): 612, 2014.
doi:10.1017/S0956796815000039

Special Issue on Run-Time Systems and Target Platforms for Functional Languages Editorial

Compiling functional languages to the existing variety of platforms calls for sophisticated implementations of run-time systems. This special issue focuses on this often-neglected aspect. We volunteered to compile this special issue in 2012 and immediately started soliciting papers. The original call for papers covered native-code platforms as well as run-time systems originally designed for non-functional languages such as the Java Virtual Machine or the .NET Common Language Runtime.

The submissions covered a wide spectrum of languages and implementations, with a focus on native-code run-time systems. This issue contains the first two accepted papers.

In *MultiMLton: A Multicore-Aware Runtime for Standard ML*, Sivaramakrishnan, Ziarek and Jagannathan offer a comprehensive account of a scalable multicore run-time system for an extended version of Standard ML. MultiMLton provides ACML, an asynchronous version of Concurrent ML, which allows programmers to elegantly express concurrency and exploit parallelism. To implement ACML's concurrency efficiently, the MultiMLton run-time system provides extremely cheap concurrency. To that end, MultiMLton provides *parasites*, which run conceptually asynchronous computations on "host threads," without allocating new threads. Moreover, the MultiMLton garbage collector implements thread-local heaps, and the run-time system minimizes coordination between these heaps.

In *A Run-Time Representation of Scheme Record Types*, Keep and Dybvig describes the run-time representation of records in the Chez Scheme system. The R6RS standard for Scheme specifies records that allow run-time creation of generative record types, single inheritance, and modular constructors. Chez Scheme extends the R6RS record system with additional facilities for implementing object-oriented programming and foreign fields. The run-time representation for these record types and records poses many small but non-trivial challenges. Keep and Dybvig's paper provides a complete tour with all the information an implementor might need to replicate or improve upon their design for Chez Scheme.

We thank the authors and referees for their outstanding efforts. We also gratefully acknowledge the support of Matthias Felleisen, David Tranah, and the JFP editorial office. Finally, we hope that you enjoy reading the papers as much as we did.

Michael Sperber
Active Group GmbH
sperber@deinprogramm.de

Lennart Augustsson
Standard Chartered Bank
lennart@augustsson.net