The HiPE/x86 Erlang Compiler: System Description and Performance Evaluation

Mikael Pettersson, Konstantinos Sagonas, and Erik Johansson
Computing Science Department, Uppsala University, Sweden
{mikpe,kostis,happi}@csd.uu.se

Abstract. Erlang is a concurrent functional language, tailored for large-scale distributed and fault-tolerant control software. Its primary implementation is Ericsson's Erlang/OTP system, which is based on a virtual machine interpreter. HiPE (High-Performance Erlang) adds a native code execution mode to the Erlang/OTP system. This paper describes the x86 version of HiPE, including a detailed account of decisions and principles that guide the design of its compiler and runtime system. We also present a brief performance evaluation which indicates that HiPE/x86 delivers performance improvements on par with the more mature HiPE/SPARC system.

1 Introduction

Erlang is a functional programming language which supports concurrency, communication, distribution, fault-tolerance, automatic memory management, and on-line code updates\(^1\). It was designed for soft real-time control systems which are commonly developed by the telecommunications industry. Judging from commercial applications written in Erlang, the language is quite successful in this domain.

The most widely used implementation of Erlang, Ericsson’s Erlang/OTP system, was until recently exclusively based on the BEAM virtual machine interpreter. This, and the fact that Erlang is a dynamically typed language requiring runtime type tests, made Erlang/OTP quite slow compared to implementations of other functional languages. The HiPE system [8, 9] was developed with the aim of significantly reducing this performance gap. HiPE achieves this by allowing flexible, user-controlled, just-in-time compilation of Erlang functions or modules to native machine code. As reported in [8], HiPE is currently the fastest Erlang implementation and offers performance which is competitive with implementations of other strict functional languages such as Scheme or SML/NJ. One drawback was that, until recently, HiPE only supported SPARC-based machines. To enable more widespread use of HiPE, we have developed an x86 version of the system. The result, HiPE/x86, is presented in this paper. Since October 2001, HiPE/x86 is included in the open source release of Erlang/OTP.\(^1\)

\(^1\) Available at www.erlang.org. See also www.csd.uu.se/projects/hipe.
In this paper, we present a detailed account of the architecture of the HiPE/x86 compiler and runtime system, design decisions we made and their associated tradeoffs. The purpose of doing so is two-fold: First, we document our implementation in a form which is potentially more easy for other implementors to follow than HiPE’s source code. Second, we believe that this information is applicable to other garbage-collected tail-recursive high-level languages and as such we hope that our experience will prove useful to anyone that gets involved in a similar project.

The paper begins with a brief overview of Erlang’s characteristics (Sect. 2). After presenting the architecture of HiPE (Sect. 3), Sections 4–6 describe the HiPE/x86 system in detail. Some of its planned improvements are also discussed (Sect. 6). Section 7 contains a brief performance evaluation, Section 8 contrasts design decisions with those in related systems, and finally Section 9 ends with some concluding remarks.

2 The Erlang Language and Erlang/OTP

Erlang is a dynamically typed, strict, concurrent functional language. The basic data types include atoms, numbers, and process identifiers; compound data types are lists and tuples. There are no assignments or mutable data structures. Functions are defined as sets of guarded clauses, and clause selection is done by pattern matching. Iterations are expressed as tail-recursive function calls, and Erlang consequently requires tailcall optimisation. Erlang also has a catch/throw-style exception mechanism. Erlang processes are created dynamically, and applications tend to use many of them. Processes communicate through asynchronous message passing: each process has a mailbox in which incoming messages are stored, and messages are retrieved from the mailbox by pattern matching. Messages can be arbitrary Erlang values. Erlang implementations must provide automatic memory management, and the soft real-time nature of the language calls for bounded-time garbage collection techniques.

Erlang/OTP is the standard implementation of the language. It combines Erlang with the Open Telecom Platform (OTP) middleware [16], a library with standard components for telecommunications applications. Erlang/OTP is currently used industrially by Ericsson Telecom and other software and telecommunications companies around the world for the development of high-availability servers and networking equipment. Additional information about Erlang can be found at www.erlang.org.

3 HiPE: Brief System Overview

HiPE is included as an optional extension in the Open Source Erlang/OTP system. It consists of a compiler from BEAM virtual machine bytecode to native machine code (UltraSPARC or x86), and extensions to the runtime system to support mixing interpreted and native code execution, at the granularity of individual functions.