MIX10: A MATLAB TO X10 COMPILER

by

Vineet Kumar

School of Computer Science
McGill University, Montréal

Thursday, October 31st 2013

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

Abstract

TBD

Résumé

TBD

Acknowledgements

TBD

Table of Contents

Appendices

List of Figures

List of Tables

Chapter 1
Introduction

MATLAB is a popular numeric programming language, used by millions of scientists, engineers as well as students worldwide[Mol]. MATLAB programmers appreciate the high-level matrix operators, the fact that variables and types do not need to be declared, the large number of library and builtin functions available, and the interactive style of program development available through the IDE and the interpreter-style read-eval-print loop. However, even though MATLAB programmers appreciate all of the features that enable rapid prototyping, their applications are often quite compute intensive and time consuming. These applications could perform much more efficiently if they could be easily ported to a high performance computing system.

On one hand, all the aforementioned characteristics of MATLAB make it a very user-friendly and thus popular application to develop software among a non-programmer community. On the other hand, these same characteristics make MATLAB a difficult language to compile statically. Even the de facto standard, Mathworks' implementation of MATLAB is essentially an interpreter with a *JIT accelarator*[The02] which is generally slower than statically compiled languages. GNU Octave, which is a popular open source alternative to MATLAB and is mostly compatible with MATLAB, is also implemented as an interpreter[Oct]. Lack of formal language specification, unconventional semantics and closed source make it even harder to write a compiler for MATLAB. Furthermore, the use of arrays as default data type and the dynamicity of the base types and shapes of arrays also make it harder to add support for concurrency in a static MATLAB compiler. Math-

works' proprietary solution for concurrency is the *Parallel Computing Toolbox*[Mat13], which allows users to use multicore processors, GPUs and clusters. However, this toolbox uses heavyweight worker threads and has limited scalability.

In this thesis, our aim is to provide MATLAB's ease of use, to benefit from the advantages of static compilation, and to expose scalable concurrency. Our solution is MIX10, an open, source-to-source compiler that statically compiles MATLAB for high performance computing by translating MATLAB programs to X10 [**?**], a language specifically designed for high performance computing systems. X10 is an object-oriented and statically-typed language which uses cilk-style arrays indexed by *Point* objects and rail-backed multidimensional arrays, and has been designed with well-defined semantics and high performance computing in mind. The X10 compiler can generate C++ or Java code and supports various communication interfaces including sockets and MPI for communication between nodes on a parallel computing system.

We have concentrated both on providing an efficient translation for the sequential core of X10, as well as providing an effective bridge to the concurrency features of X10. One key way in which we interface with concurrency in MATLAB is by designing and implementing a translation of the MATLAB `parfor` construct to X10. We also introduced concurrency constructs in MATLAB analogous to those provided in X10, thus allowing users to further specify fine-grained concurrency in their programs.

The overall goal of the MIX10 project it to allow scientists and engineers to write programs in MATLAB (or use existing programs already written in MATLAB), and at the same time enjoy the benefits of high performance computing via the X10 system without having to learn a new and unfamiliar language. Also, since the X10 compiler has backends that can produce both C++ and Java, MIX10 can also be used by systems that use MATLAB for prototyping and C++ or Java for production.

1.1 Contributions

The major contributions of this thesis are as follows:

Identifying key challenges: We have identified the key challenges in performing a semantics-

preserving efficient translation of MATLAB to X10.

Overall design of MIX10: We provide the design of a source-to-source translator, building upon the *Mc***LAB** front-end and analysis toolkits[Doh11, DH12].

Techniques for efficient compilation of MATLAB arrays: Arrays are the core of MATLAB. All data, including scalar values are represented as arrays in MATLAB. Efficient compilation of arrays is the key for good performance. X10 provides two types of array representations for multidimensional arrays: (1) Cilk-styled, region-based arrays and (2) rail-backed arrays. We compare and contrast these two array forms for a high performance computing language in context of being used as a target language and provide techniques to compile MATLAB arrays to two different representations of arrays provided by X10.

Analysis to identify variables safe to be declared as `Long` type: In MATLAB all variables are by default of type `Double`. In X10 however, certain variable uses, for example, use of a variable as an array index in an array access operation, require the variable to be of type `Long`. We provide an analysis to automatically identify variables that can be safely declared to be of type `Long` without affecting the correctness of the generated X10 code. This helps to eliminate most of, otherwise necessary, typecast operations which our experiments showed to be a major performance bottleneck in the generated code.

Support for concurrency constructs: `parfor` is a MATLAB construct that allows parallel execution of for loop iterations. We provide technique to effectively compile `parfor` construct to X10. We also discuss our strategy to handle vectorized instructions in a concurrent fashion. Besides providing support for existing MATLAB concurrency features, we have also introduced X10 like concurrency constructs in MATLAB, allowing MATLAB programmers to expose fine-grained concurrency in their programs.

Code generation strategies for key language constructs: There are some very significant differences between the semantics of MATLAB and X10. A key difference is that

MATLAB is dynamically-typed, whereas X10 is statically-typed. Furthermore, the type rules are quite different, which means that the generated X10 code must include the appropriate explicit type conversion rules, so as to match the MATLAB semantics. Other MATLAB features, such as multiple returns from functions, a non-standard semantics for `for` loops, and a very general range operator, must also be handled correctly. We have also designed and implemented a template-based system that allows us to generate specialized X10 code for a collection of important MATLAB builtin operations.

isComplex value analysis: We designed an analysis for identification of complex numerical values in a MATLAB program. This helped us to extend MIX10 compiler to also generate X10 code for MATLAB programs that involve use of complex numerical values.

Working implementation and performance results: We have implemented the MIX10 compiler over various MATLAB compiler tools provided by the *Mc*LAB toolkit. In the process we also implemented some enhancements to these existing tools. We provide performance results for different X10 backends over a set of benchmarks and compare them with results from other MATLAB compilers including Mathworks' MATLAB implementation and Octave.

1.2 Thesis Outline

This thesis is divided into 10 chapters, including this one and is structured as follows.

Chapter 2 provides an introduction to the X10 language and describes how it compares to MATLAB from the point of view of language design. *Chapter 3* gives a description of various existing MATLAB compiler tools upon which MIX10 is implemented, presents a high-level design of MIX10, and explains the design and need of MIX10 IR. In *Chapter 4* we introduce different types of arrays provided by X10, we identify pros and cons of both kinds of arrays in the context of X10 as a target language and describe code generation strategies for them. We also present a detailed description of the IntOk analysis. In *Chapter 5* we describe our code-generation strategies for existing MATLAB concurrency constructs

and addition of new fine-grained concurrency controls. *Chapter 6* gives details of code generation strategies for important MATLAB constructs. In *Chapter 7* we provide a detailed description of our analysis to identify complex numerical values in MATLAB programs. *Chapter 8* provides performance results for code generated using MIX10 for a suite of benchmarks. *Chapter 9* provides an overview of related work and *Chapter 10* concludes and outlines possible future work.

Chapter 2

Introduction to X10 and contrast with MATLAB

TBD

Chapter 3
Background and High level design

TBD

Chapter 4
Compiling for high performance of sequential code

TBD

Chapter 5
Concurrency constructs in MiX10

TBD

Chapter 6
Code generation

TBD

Chapter 7
isComplex value analysis

TBD

Chapter 8
Evaluation

TBD

Chapter 9
Related work

Chapter 10
Conclusions and Future Work

TBD

Appendix A
XML structure for builtin framework

TBD

Appendix B
isComplex analysis Propagation Language

TBD

Appendix C
MIX10 IR Grammar

TBD

Bibliography

[DH12] Anton Dubrau and Laurie Hendren. Taming MATLAB. In *Proceedings of OOPSLA 2012*, 2012, pages 503–522.

[Doh11] Jesse Doherty. McSAF: An Extensible Static Analysis Framework for the MATLAB Language. Master's thesis, McGill University, December 2011.

[Mat13] MathWorks. Parallel computing toolbox, 2013. http://www.mathworks.com/products/parallel-computing/.

[Mol] Cleve Moler. The Growth of MATLAB and The MathWorks over Two Decades. http://www.mathworks.com/company/newsletters/news_notes/clevesco

[Oct] GNU Octave. http://www.gnu.org/software/octave/index.html.

[The02] The Mathworks. Technology Backgrounder: Accelerating MATLAB, September 2002. http://www.mathworks.com/company/newsletters/digest/sept02/a