Automatic Parallelisation of Rust Programs at Compile Time

Michael Oultram

***Abstract*—There is a significant amount of research in automatic translation of sequential source code into a parallelized version. Most of this has been focused on FORTRAN and C which are both unsafe programming languages. This paper explores the literature and applies these ideas to the safe programming language Rust. TODO: Expand, and cleanup**

I. Introduction

II. Literature Review

Most research is for FORTRAN and the DO loops (Banerjee 1993).

TODO: Look at the different models, try to explain the differences

Some people have converted C-to-CUDA (Baskaran, Ramanujam and Sadayappan 2010; Verdoolaege et al. 2013).

III. Problem Details

References

Banerjee, Utpal (1993). *Loop transformations for restructuring compilers: the foundations*. Boston: Kluwer Academic Publishers. ISBN: 079239318X.

Baskaran, Muthu, Jj Ramanujam and P Sadayappan (2010). "Automatic C-to-CUDA code generation for affine programs". In: *Compiler Construction*, pp. 244–263.

Verdoolaege, Sven et al. (2013). "Polyhedral parallel code generation for CUDA". In: *ACM Transactions on Architecture and Code Optimization*. ISSN: 15443566. DOI: 10.1145/2400682.2400713.