Hybrid Approach for Data-flow Analysis of MPI Programs

Sriram Aananthakrishnan
University of Utah
sriram@cs.utah.edu

Greg Bronevetsky
Lawrence Livermore National Laboratory
greg@bronevetsky.com

Ganesh Gopalakrishnan
University of Utah
ganesh@cs.utah.edu

ABSTRACT

With the increasing cost of developing robust HPC software, precise data-flow analysis for MPI programs – the mainstay of HPC programming – are essential. The knowledge of communication is essential for precise data-flow analysis and the difficulty of statically determining it makes the conventional techniques insufficient. Hybrid methods combining static and dynamic techniques are needed and in this work we demonstrate one such approach in building the parallel control-flow graph which can then be used to leverage the precision of data-flow analyses for MPI programs.

Categories and Subject Descriptors

D.3.4 [**Programming Languages**]: Processors—*compilers*; F.3.2 [**Logics and Meaning of Programs**]: Semantics of Programming Languages—*Program analysis*

Keywords

Data-flow analysis; MPI; parallel control-flow graph

1. INTRODUCTION

Sequential data-flow analyses are insufficient to analyze MPI applications that are, by nature, concurrent. Existing data-flow techniques for MPI [1] [2] suffer from imprecision as they over approximate the communication behavior. The framework based on [3]is communication sensitive, however to determine a send-receive match statically, it employs a complex matching algorithm that requires non-standard inferences such as function composition on its abstractions. To address the matching problem, for a MPI program with N processes, we create N instances of the control-flow graph (CFG) and connect a send with its receive by concretely executing the respective MPI operations. The matchings are then handled by the MPI runtime.

The edges that a connect a send with its receive are called communication edges, i.e pairs (s, r) where s is a node of CFG_i with send semantics and r is a node of CFG_j with receive semantics. The parallel control-flow graph CFG^N consists of N replicated sequential CFG with such communication edges. The CFG^N construction requires composition of multiple analyses, which is the main focus of our work.

ICS'13, June 10–14, 2013, Eugene, Oregon, USA.
ACM 978-1-4503-2130-3/13/06.

2. HYBRID ANALYSIS

Our key insight is that in majority of MPI applications the communication is determined by the values of pid (rank) and size of the MPI communicator. We treat these variables as concrete values instead of symbolic abstractions, enabling us identify the value of the target expressions in a send/receive function. We associate an analysis instance (also a MPI process) for each process of the MPI application. Each instance is a composition of following analyses (i) slicing (ii) constant propagation and folding, (iii) unreachable code elimination (iv) communication invariance (v) dynamic send-receive matching.

The control-flow graph is sliced to contain only statements affecting the communication. Constant propagation and folding allow us to determine the values of target expressions such as $rank\%size - 1$ in MPI functions as each analysis instance gets its copy of rank, size. Unreachable code elimination prunes the path not reachable by an analysis instance. Communication invariance determines that the communication is not input dependent and does not involve ambiguous paths, where execution is dependent on input or the message. The dynamic send-receive matching on reaching the control-flow node of a MPI function concretely executes it. It sends information about the MPI function and the corresponding receiver interprets this information to build the communication edges. Our approach employs hybrid method to build the CFG^N for MPI programs. Data-flow analyses to improve precision can exchange state information along the communication edges using MPI.

The framework in this paper is being implemented based on [4] and is integrated with the ROSE[5] compiler framework. Preliminary evaluations are in progress.

3. REFERENCES

[1] Strout. M, Kreaseck. B, Hovland. P D., *Data-Flow Analysis for MPI Programs*, ICPP '06

[2] Shires. D, Pollock. L,Sprenkle. S, *Program Flow Graph Construction For Static Analysis of MPI Programs*, PDPTA 1999

[3] Bronevetsky. G, *Communication-Sensitive Static data-flow for Parallel Message Passing Applications*, CGO '09

[4] Bronevetsky. G, Burke. M, Aananthakrishnan. S, Zhao. J, Sarkar. V, *Compositional data-flow via Abstract Transition Systems*, (TR)

[5] Quinlan. D, *ROSE Compiler*,http://www.rosecompiler.org