Improving Cache Locality with Coalesced Map-Reduce Operations

Michael Choquette, Rokhini Prabhu
mchoquet, rokhinip
March 18, 2015

Project Web Page

---

http://www.andrew.cmu.edu/user/mchoquet/15745/index.html

Project Description

`C0` is a safe subset of the C programming language augmented with contracts which has been developed here at CMU by Frank Pfenning and his team. In our project, we will extend the `C0` language with additional language constructs for `map`, `reduce` and `filter` constructs on immutable `sequences`. We chose `C0` since we are both comfortable with the language, and have access to a working `C0-C` compiler to build off of.

Our goal is to investigate the speedups that can be gained from statically combining sequential `map`, `reduce`, and `filter` operations into compound operations by composing the kernels. This reduces the number of iterations through the data, improving locality, and removes intermediate allocations.

We will evaluate the impact of our optimizations by timing the performance of benchmark tests. We expect our project to consist of 4 phases:

1. Integrating `map, reduce` and `filter` constructs into `C0`
2. Coalesce multiple `map` operations into a single `map` operation
3. Coalesce `map` and `reduce` operations together
4. Coalesce `map` and `filter` operations together

As such, we would have achieved our goal to a 100% if we are able to accomplish all 4 stages of optimization. We also have a 75% mark of having accomplished only 3 out of the 4 stages. Our current optimization pass act on each function individually and given extra time, we would also like to explore how such optimizations would work on an inter-procedural level.

Literature Search

Our main reference for literature is the documentation for the *210 library on Sequences* for we plan to implement similar functionalities into `C0`. We will also be using the *Language guide for C0* to ensure that we still adhere to the safety features of the language.

Plan of Attack and Schedule

Week	Work to be done	Who
1	Get C0 starter code, design grammar and typing rules for `map`, `reduce` and `filter` constructs.	Rokhini and Michael
2	Implement `map`, `reduce` and `filter` in `C0`. Determine what dependency analysis needs to be done for coalescing map functions, and how to do it.	Rokhini and Michael
3	Implement rules for phase (2) of project, design rules for phase (3) and (4).	Rokhini and Michael
4 and 5	Implement rules for phase (3) and (4) of project.	Rokhini and Michael
6	Finish up any leftover work from week 4 and 5 and prepare for presentation.	Rokhini and Michael

Milestone

At the end of week 3 we would like to have finished implementing composition of map operations, and have a strong understanding of how to combine reduce and filter operations.

Resources Needed

We will use a `C0`-to-`C` compiler, but no other special resources. We will run our code on the campus unix machines to test performance.

Getting started

We have not done much apart from writing the proposal so far. We do not foresee any trouble with starting our project immediately.