Data-Centric Parallelisation

Magnus Morton, Zhibo Li, Björn Franke

3rd December 2020

Overview

- "Traditional" auto-parallelisation tools are code-centric
- Ignore context of whole program
- Struggles on pointer-chasing or control-flow heavy programs

Overview

- Can we use whole program context?
- Can we replicate a human expert?

Data-Centric Parallelisation

- Data-Centric Parallelisation (DCP) project aims to do so
- We want to develop a data-first paradigm for parallelisation
- Raise the level of abstraction of existing data structures

Motivating Example

- Consider a program that traverses and increments each element of a linked list

```
struct node *curr = list;
while (curr != NULL) {
        curr->value++;
        curr = curr->next;
}
```

- This is not parallelisable by traditional parallelising compiler
- We can see that it trivially parallelisable

Motivating Example

- Our linked list could easily be swapped for an array, or even a set if we know ordering and uniqueness constraints

```
std::set<int> list;
for (auto&& v: list) {
        v++;
}
```

Motivating Example

- Program is now trivially parallelisable
  - Either by automatic tool, or by using parallel library
- Program is also modernised
- DCP seeks to do this automatically

Preliminary Work

- Early work suggests that this sort of approach can give 3.5x - 980x speedups
- This is just by lifting to `stl` data structures

DCP Overview

- DCP is formed of 3 components
- Data structure detection
- Property-based data structure library
- A code generation tool

Data Structure Detection

- We need to detect exisiting use of data structures
- Not just the exact data structure used, but the most abstract
- What complexity do they want for access/insert/delete etc.?
- Does ordering matter?
- What the programmer wrote may not be what they actually want or need

Shape Analysis

- Dynamic analyis
- Pointer-based analysis
- Looking at "shape" of data structure on heap

Shape Analysis

- DDT and MemPick best known approaches
- Require instrumentation
- Assumptions about when and where data structure operations happen
- No data on stack

Shape Analysis

- Probably enough to detect most data structures
  - But maybe not reliably
- Does it tell us everything about the data structure?
  - Doesn't tell us about certain semantics and properties
  - Ordering?
  - Uniqueness?
- Does it tell us if abstraction could be raised?

Pattern Matching

- Analyse static structures
  - e.g. LLVM IR
- Express patterns or constraints in a DSL or LLVM pass
  - e.g. CAnDL idioms
- We can use this to match patterns corresponding to some data structures

Other Static Analyses

- Can we detect data structures at syntactic level?
- MLIR?
- GIMPLE?

Hybrid

- Can we combine static and dynamic approach?
- Narrow search with static matching
- Use information from static analysis to inform/improve dynamic analysis

Current Status

- CAnDL idiom to detect pointer chasing loops

```
for ( node = head ; node != NULL ; node = node->next )
```

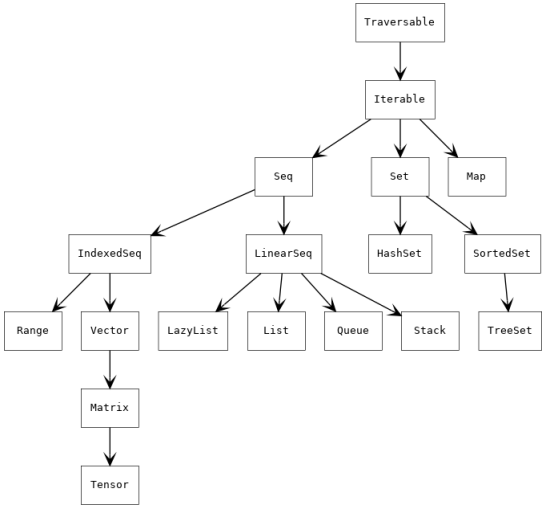
- Successfully detects 3 out of the 5 such loops in data set of 106 programs
- No false positives

Property-Based Data Structure Library

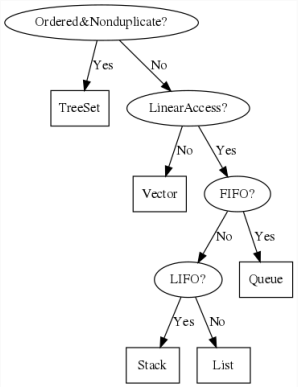
- Inspired by Scala collections library, we have Scale, a C++ property-based data structure library
- Expresses data structures in terms of properties, rather than implementation detail.
- e.g. ordering, access, append complexity etc.

```
scale::Collection<Property1, Property2, ...> collection{size};
```

Property-Based Data Structure Library



Property-Based Data Structure Library



Property-Based Data Structure Library

- Operations on Scale use functional paradigm
- `map`, `reduce`, `zip` etc.
- These will also act as parallel skeleton library

Graph analytics

- Looking to extend Scale to support graph analytics
- There are many competing graph libraries
- Significant activity in the field
- We want a solution which will detect graphs in many existing libraries

Graph analytics

- Graph Scale is currently looking like the Boost Graph Library
- BGL allows specifying graphs based on properties or traits
- Leaves a lot to be desired

Code Generation

- Nothing really exciting planned here yet
- Should be fairly straightforward mapping from detected data type to Scale

Code Generation. . . .?

- Maybe program synthesis?

Summary

- Data-Centric Parallelisation aims to parellelise by focussing on data structures and raising the level of abstraction.
- Developing static/dynamic data structure detection
- We have started developing Scale, a property-based data structure library.
- Vision is for legacy code to be automatically lifted into Scale.