LINUX
PLUMBERS
CONFERENCE
August 24-28, 2020

Project Ranger Update

Andrew MacLeod
Aldy Hernandez

- Introduced GNU tools cauldron 09/19

- Quick summary
- Changes since last year
- Current stage 1 plans
- Future plans

Quick summary

- Extensible infrastructure (range-ops)
- Multiple sub-range support – precision!
- On-demand range query in GCC

Range Operations

- Range equation solver for instructions
- LHS = OP1 + OP2
- Can solve for LHS, OP1 or OP2 with the other 2
- Allows general calculations in reverse

    if (x < 5)
    is really [1,1] = (x < 5) on the true edge,
    which solves x as [MIN, 4]

- This has been in trunk since Oct 2019

Multiple sub-ranges

- Multiple non-overlapping, integral sub-ranges
- Templated
  - Choose your desired precision
  - “Widest_irange” for maximal precision
- Replaces value_range: a single pair plus “anti-range”
- Live in trunk, available now

Ranger API

On demand, cached calculations & queries
- no other infrastructure requirements

- range_of_expr (tree expr, gimple *s)
- range_of_stmt (gimple *s)
- range_on_edge (edge e, tree name)
- range_on_entry (basic_block bb, tree name)
- range_on_exit (basic_block bb, tree name)

- Preparing for trunk now

Significant changes

- Class irange/value_range merge
- Numerous range-op refinements
- Internal restructuring of ranger
- Consolidation with existing VRPs
- Relational query prototype

Irange changes

- Uses trees internally, not wide_int
- Multi-range API mostly unchanged
- Compatibility layer with legacy
  - int_range <1> is now a value_range
- Deprecated legacy API
- Porting guidelines document available

Relational object

- class value_relation
- Tracks ==, !=, <, <=,>, >=, no relation
- Can be combined union/intersect/not
- (a_2 < b_6) union (a_2 == b_6)
  - Results in a_2 < = b_6
- If (b_6 == a_2) will resolve to true if the intersection of the condition and the known relation is... the condition.

  Ie, the condition is a subset of the known relation.

Registering Relations

- Follows range-ops model, but not integrated there yet
  - Query/find relation between 2 of LHS, OP1, OP2
  - Augmented with any known ranges
  - Simple for if (x_1 < b_2).
  - x_2 = b_3 + 6
    - Registers (x_2 > b_3) for signed values
    - For unsigned, if ranges are provided
      - (x_2 < b_3) for x_2 == [0, 5]
      - (x_2 > b_3) for x_2 == [6, MAX]
      - (x_2 != b_3) for x_2 == [0, MAX] // effectively no range available
    - Range-ops or query can use/calc b_3 ranges instead

Simple Example

- (x_2 < b_3) for x_2 == [0, 5]
- (x_2 > b_3) for x_2 == [6, MAX]

```
X_2 = b_3 + 6
if (x_2 <= 5)          //  x_2 =  [0,5] on true edge, b_3 = [0, MAX-6]
     overflowed()      //  (x_2 < b_3) will be true here
else                   //  x_2 = [6, MAX]  false edge, b_3 = [MAX-5, MAX]
     something()       //  (x_2 > b_3) will be true here
// afterwards,  (x_2 != b_3) will be true
```

Relational Queries

- Operates as an oracle
- Tracks “equivalency sets” and “other relations”
- Equiv sets solves first, then relations
- Register relations as statements are seen
- Currently requires dominators for efficiency
- API WIP… currently
  - bitmap query_equiv (tree name, gimple *s = NULL);
  - bool relation_oracle::apply_relations (irange &r, gimple...

Current Stage 1

- Range-ops integrated in GCC 10
- Class irange now in trunk (July)
- Ranger going thru final performance
  - Should be checked in by early Sept.
  - Includes 3 or 4 pass conversions for speedup
    - (walloca, wrestrict, and wprintf)
  - Hybrid EVRP : old and new coexist for now
- Relation oracle due Early October.

The future

- Ranger includes iterative updating
  - Replace EVRP and VRP with just common pass
- “push” range queries to appropriate passes
- Enhanced range-ops for multi-range
- Block “outgoing range” refinements
- Tighter integration with dominators/relations
- Non integral ranges.
- Bit-mask tracking