Design and Implementation of a Cluster-based Approach for Software Verification

Bachelor's Thesis

Alexander Ried

January 29, 2020

Outline

Motivation

Software Analysis is Resource Hungry

Different Parallelization Techniques

Previous Work

Cluster Support for ParallelBAM

Initial Approach: In-Memory Computing Platform

Second Approach: Actor Model

Large Problems Require Many Resources

- Analyses are often CPU-bound.
- Solution: Parallelization
- Physical limit of cores per computer.
- Solution: Use multiple computers.
- May also help with memory-bound problems.

Combine Different Analyses

- Different analyses solve different problems efficiently.
- Choosing the best one in advance is hard.
- Simply try different analyses.

⇒ Portfolio analysis

Bug Hunting

- Doesn't aim at verification.
- Maximize probability of finding a bug.
- Explore different parts of state space in parallel.

Examples: PredatorHP or Spin/Swarm

Program Verification

How to verify complex software?

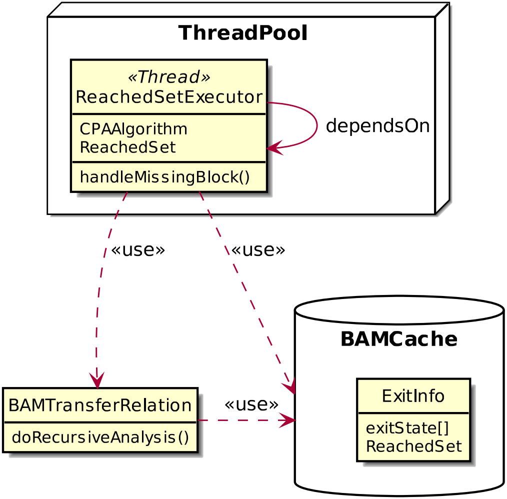
Solution: divide and conquer

- Split input up front (Facebook Infer)
- Cooperative analysis (ParallelBAM in CPAchecker)

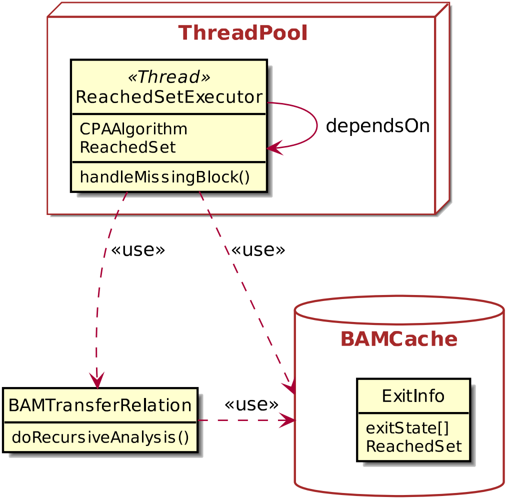
Block Abstraction Memoization

- Cache and reuse result of block analyses (e. g. functions and loops).
- Additional reduce and expand operators for increased cache-hit rate
- Independent of underlying analysis

ParallelBAMAlgorithm



Domain-independent Component Substitution



Minimal changes to existing implementation.

Replace by distributed versions:

- Executor
- Map

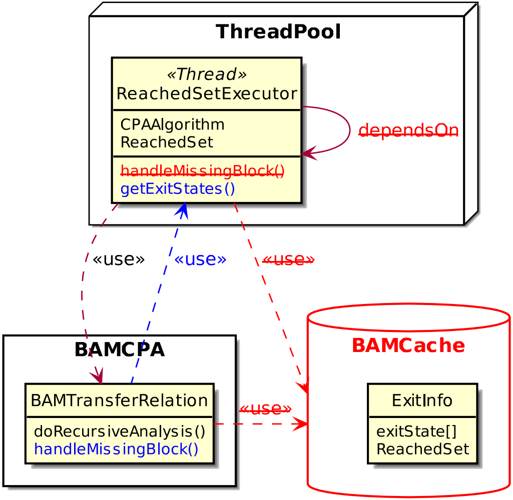
Apache Ignite

- Speeds at petabyte scale
- Key-Value Data Grid
- Compute Grid with Affinity Collocation
- Apache License 2.0

Not Suited!

- Data grid stores serialized values.
- Normalization is required.
- Serialization still too much overhead.
- Possible solution: Work on binary objects? A whole set of new problems.
- Easily used a few hundred MBit/s in CIP pool.
- Moderately complex analyses did not finish.

Reevaluate Requirements



- Keep initialized executors around.
- Replace cache by requests to executors.
- Adjust TransferRelation to handle missing blocks.

Actor Model

- Everything is an actor.
- Actors are containers for private state and behavior.
- Actors exchange messages.
- Actors are location transparent.

Good fit for distributed, parallel systems.

Integration of Actor Model into CPAchecker

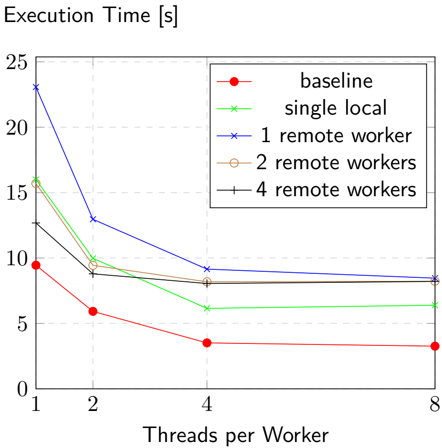
- Up to 50 million msg/sec on a single machine.
- Small memory footprint; 2.5 million actors per GB of heap.
- Apache License 2.0

- Messages may be lost.
- Includes distributed data primitives.

No changes to core CPA components (e. g. reached sets, cpa algorithm)

Evaluation

Runtime Using Multiple Machines

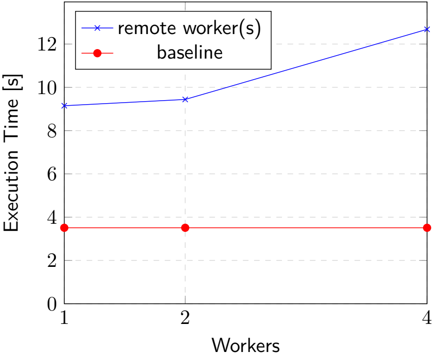


- RERS Problem 4
- Slower than baseline. ☹
- Additional nodes do improve performance. ☺
- Problem not ideally suited for distributed analysis.

Summary

- Actor model may be reasonable for single machine.
- Cluster suitability depends heavily on program structure and size.
- Only tested with value analysis.
- Outlook
  - Setup CI to compare performance impact of each change.
  - Identify why actor version is slower than original implementation.
  - Find better strategy for worker selection.
  - Convert more components to actors for finer-grained parallelization (?)

Evaluation with 4 Total Threads


remote worker(s)
baseline
Execution Time [s]
Workers