Solving@Home

Wiley Corning :: CS 739 :: Spring 2021

Motivation

Program synthesis is slow - in fact, it’s NP-Hard.

However, many techniques are well suited to a divide-and-conquer approach.

- If you’re enumerating, just divide up the search space!

Prior Work

Folding@Home has been very successful.

But how can we deal with a malicious or malfunctioning volunteer?

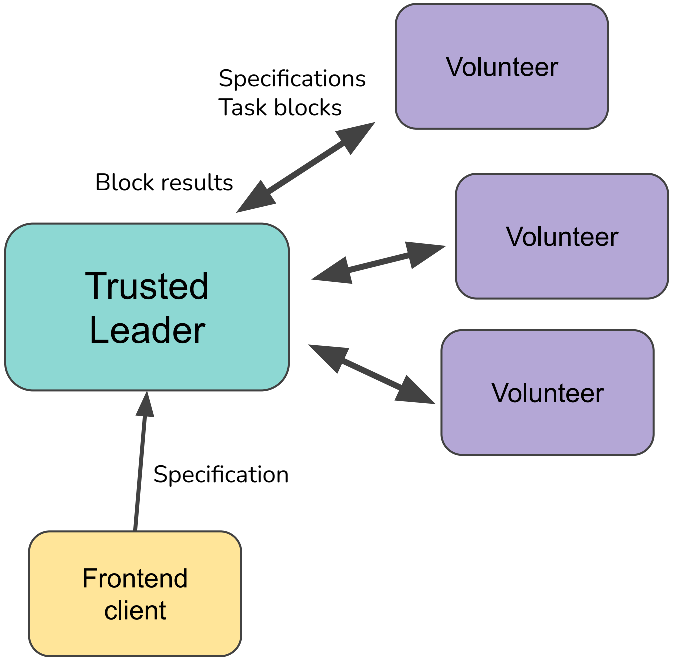
- Confirming a solution is simple, but it’s hard to detect false negatives.

We use a consensus approach modeled on *Castro and Liskov, 1999*.

System architecture

Central leader node:

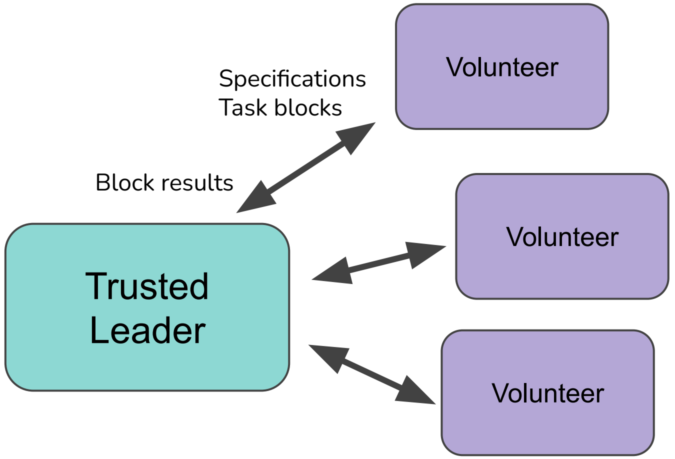
- Handles client requests
- Maintains authoritative progress state
- Dispatches *task blocks* to volunteers



Task delegation

The leader incrementally generates task blocks.

- e.g., “check for a solution in this range.”

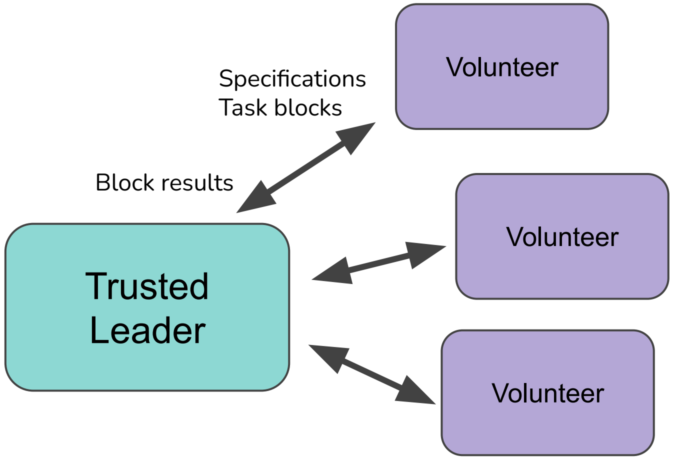


Each task block can be assigned to at most 3F+1 volunteers at a time.

- Consensus is reached when 2F+1 volunteers agree.
- The system can work on multiple problems simultaneously.

Parallelism

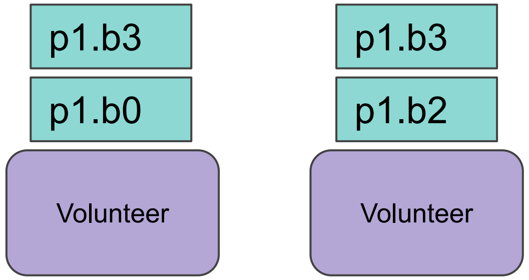
Given enough volunteers, we can work on multiple task blocks simultaneously.



Volunteers and leader communicate over a bidirectional streaming channel.

- Assignment and completion of tasks is decoupled.

Slow or crashing nodes



Each volunteer can have at most *n* pending tasks.

- For reasonably small task blocks, this effectively acts as load balancing.

If a volunteer crashes, its pending tasks will be freed for another to try.

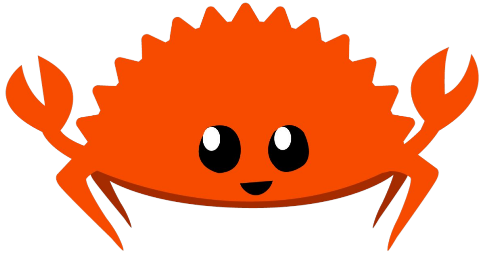
The synthesis task

Demo version: cycle through *every* program in increasing size.

- Frontend client sends example-guided specifications.
- Volunteers are equipped with a hardcoded grammar + interpreter.
- Volunteers check each program for matching output.

Implementation details

Built with Rust, using the Tokio runtime and Tonic gRPC library.



Remaining TODOs

- Cloud deployment
- Performance metrics
  - Demonstrate effect of slow / crashing nodes
- Robustness testing
  - Demonstrate resilience to <= F adversaries

Future Goals

Goal for synthesis architecture: support *reductions*.

- i.e., volunteers report new terms that are distinct modulo some equivalence relation.
- Leader then sends batches of distinct “building block” terms.

Connect to the Semgus solver ecosystem.

Allow frontend clients to disconnect and reconnect.

Allow volunteers to resume after disconnection.

Provide status updates to client while work is ongoing.

Thank you!