Milestone Report

Updated Webpage

https://github.com/devinqu02/alpaca

Major Changes

Instead of using an embedded emulator, we built a non-volatile memory (NVM) simulator on the HW VM. This gives us more control over the compiler, the runtime, and the environment. Emulators don't necessarily support NVM and sometimes they don't have ways to load elf files. Emulators meeting our requirements were not well documented or easy to use. We designed a simulator to address all the issues above.

What You Have Accomplished So Far

NVM simulator: The simulator takes in as input another program and simulates running the code on an energy-harvesting device with NVM. It comprises the simulator program and a wrapper object file to link your compiled code. The wrapper code is written to ensure an accurate simulation of energy harvesting devices for easy and reliable benchmarking later.

Runtime + Frontend: We completed the runtime and front-end for our task-based intermittent computing framework. The "front-end" (implemented using C macros) allows the programmers to write code in terms of tasks. The runtime keeps track of the execution state of the program. The runtime also "commits" variables to memory at the end of tasks using a 2-phase commit.

LLVM analysis pass (scalars): We have also implemented an LLVM pass to find write-after-read occurrences of (scalar) global variables for each task in our framework. This analysis pass will allow us to update our IR to selectively create private copies of global variables.

Meeting Your Milestone

We did not meet our milestone as we were hoping to have replicated privatization of scalar variables based on the original paper by the milestone deadline which was our 75% goal. We didn't properly anticipate the difficulty of setting up an emulator or the time commitment required to build a simulator (as discussed below). We still intend to hit our 100% goal of replicating the paper in its entirety and attempt parts of our 125% goal such as improved array dependence analysis.

Surprises

When building the NVM simulator, a huge surprise was how difficult it was to work with linker scripts. We needed to assign global variables to a separate section of memory which would be treated as non-volatile. One way of achieving this involved working with linker scripts. Given the

lack of good documentation/examples (as well as issues with PIE/ASLR), this took much longer than anticipated.

Revised Schedule

- **Week 5 (11/25):**
    - Devin - Implement scalar privatization pass
    - Shubham - Implement runtime for array privatization
    - Both - Add microbenchmarks for scalar privatization + array privatization
- **Week 6 (12/2):**
    - Devin - Implement LLVM pass for array privatization
    - Shubham - Implement runtime for improved array analysis (if time permits)
    - Devin - Implement pointer analysis in LLVM (if time permits)
    - Both - Finish implementation, evaluate benchmarks and complete final report

Resources Needed

Environment - We are using HW VM w/ custom NVM simulator
(we might need a separate device to test on to allow only 1 process to run real-time for accurate benchmarking/timely power loss simulation. We can use a cloud VM which we can get anytime)
Embedded Hardware and/or Emulator - No longer needed
Benchmarks - We can reuse the 6 benchmarks from the original paper (+ adding stubs for sensor I/O) (we will benchmark different versions of our own compiler)