Multifidelity DRAM Simulation in SST

Patrick Lavin
Sandia National Labs

Sudhanshu Agarwal
UIUC

Ryan Lynch, Jeffrey Young, Richard Vuduc
Georgia Institute of Technology

Problem: Simulation is slow

We will always want to speed up simulation. While architects are able to pick between levels of detail when designing simulation, we will find further speed-up if we can adjust the level of detail during simulation, depending on the behavior of the simulated components.

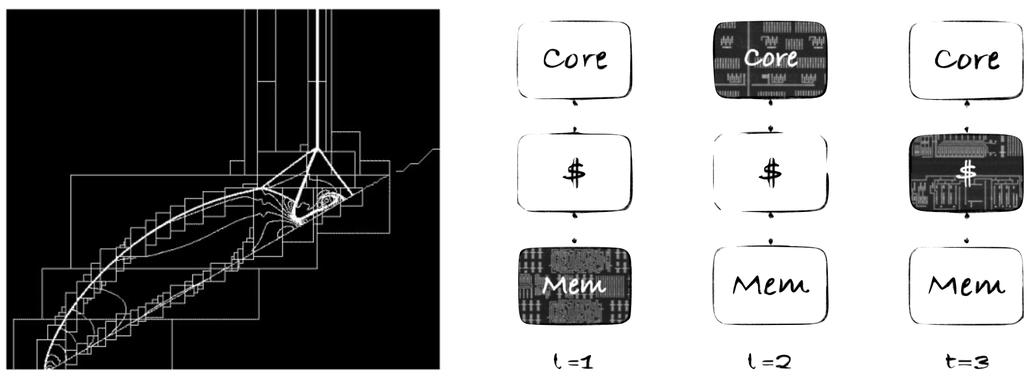

Figure 1: Adaptive mesh-refinement (left, Source Rtfisher, CC BY-SA 3.0). A depiction of complexity moving through difference components as a simulation executes (right).

What is Multifidelity Simulation?

Multifidelity Simulation consists of

1. Monitoring the behavior of simulation components
2. Training surrogate models based on the behavior of those components
3. Swapping out the components with the surrogates during simulation

We want to do this so that we can potentially accelerate simulations for which we don't need full detail for some or all of the components.

1. Phase Detection

Phase detection breaks up a running program into phases. Every 10000 instructions is given a phase id.

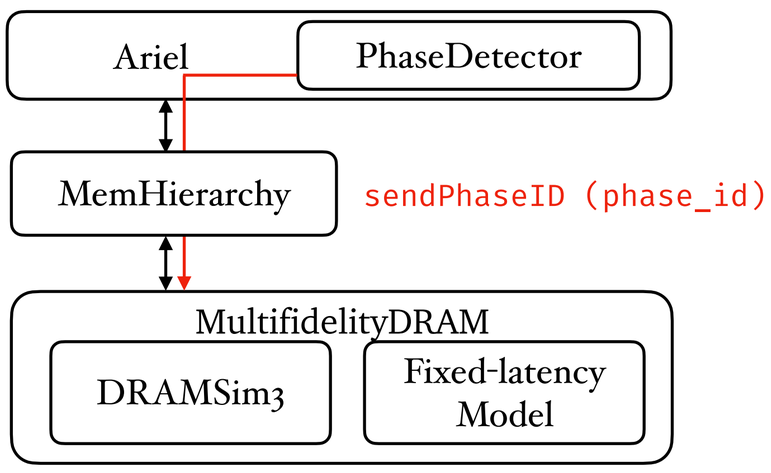

Figure 2: Example SST configuration. The phase detector sends phase messages through the memoryHierarchy to the DRAM simulator.

2. Model Training

We create a simple fixed-latency model for each phase. On the first execution of a phase, we record the average latency of accesses. On subsequent executions, we skip DRAMSim3 and send the response back using the average latency.

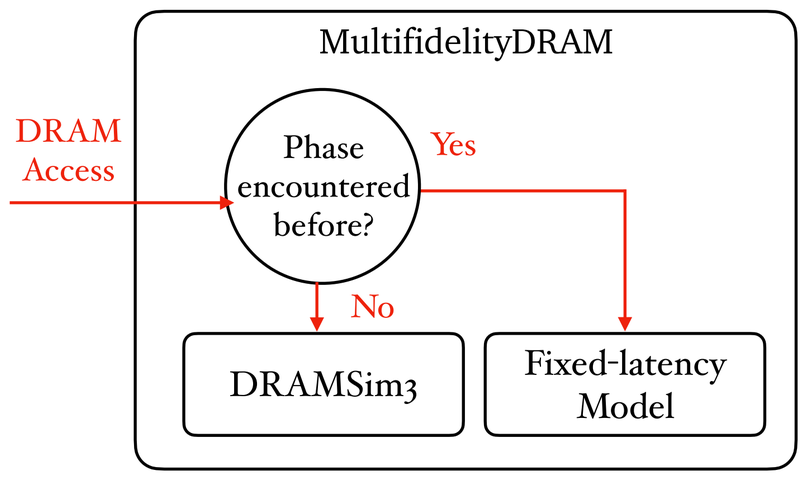

Figure 3: Memory accesses are routed through a model created for the phase

Results

1. This work represents the first implementation of a multifidelity model in a mature simulation framework, the Structural Simulation Toolkit. Prior work had implemented multifidelity simulation in a simple, in-order python cache simulator [Lavin et al., Computing Frontiers '21].
2. We have shown that we are able to achieve some speedup on a simple program, although more work needs to be done to tune the accuracy..

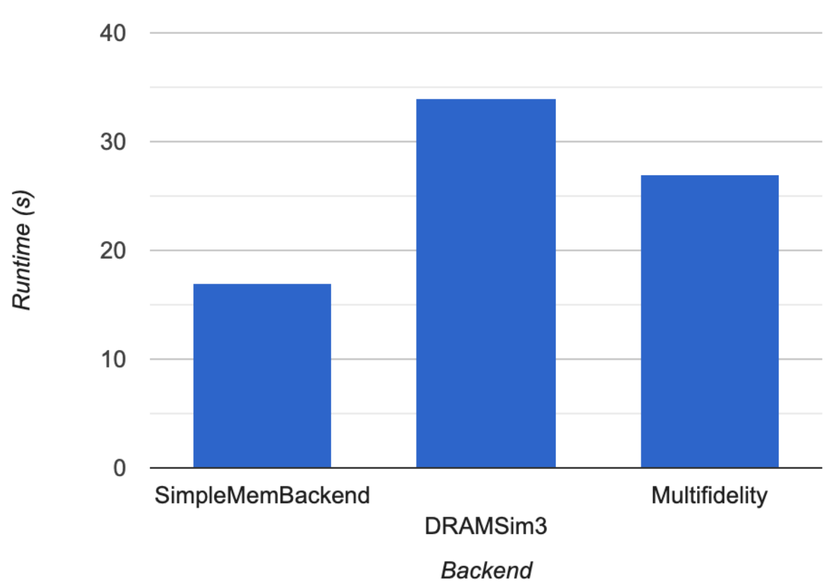

Figure 4: We simulated a simple Hello World program on an architecture with a small cache, to approximate a program with a low cache hit rate. The standard fixed-latency backend, SimpleMemBackend, is about twice as fast as a detailed model, DRAMSim3. Our phase-aware, Multifidelity backend achieved some speedup, but the accuracy of this approach needs to be improved, through (1) different phase detector parameters or (2) different phase detection algorithms.

Current and Future Work

SST interface for phase detection component
The phase detection functionality is currently tightly integrated with SST's Ariel component. We want to make this into its own component that can be easily used with other core models, and create an interface suitable for other phase detection algorithms.

Optimized implementation
The overhead associated with collecting instruction pointers can be significant. Techniques such as sampling can be used to reduce this overhead.

Phase parameter tuning
Choosing phase parameters can be more of an art than a science. A larger study should be conducted on more inputs to choose parameters such as phase length that provide useful phases.

Error bounds for the surrogate models
Statistical simulation is able to provide error bounds on their simulations, which we currently lack. Techniques such as the bootstrap method can give us bounds on how closely our models represented the observed behavior.

Related Techniques

Online Model Swapping for Architectural Simulation
⇒ Create a Markov model for each phase representing the behavior of the LiD, and swap during simulation [Lavin et al., Computing Frontiers '21]

B²Sim: Basic-block centric multifidelity methods
⇒ Simulate a BB only when it is first encountered [Lee et al. CODES+ISSS '06]

SimPoint: Sampled Simulation
⇒ Use offline clustering to find representative regions [Perelman et al., SIGMETRICS '03]

Statistical Sampling
⇒ Use sampling theory to estimate the performance from a sample of the program [Conte et al., ICCD '96]

Also related: Tools for design-space exploration, Higher-level models such as Sniper (interval simulation)