Automated GPU Design Space Exploration

Reference: Stargazer: Automated Regression-Based GPU Design Space Exploration, Wenhao Jia, Kelly A. Shaw, and Margaret Martonosi, ISPASS 2012

Observation: Designing GPUs involves discerning relative importance and potential interactions of many design options, which is difficult even just for small design spaces.

Our Work: Build GPU performance models from random design samples using automated parameter selection that can account for parameter pair interactions.

Method: A stepwise algorithm automatically selects key design parameters and significant parameter interactions to build performance models with only relevant terms.

Results: Our method automatically and efficiently reveals the relative importance of different design options for a 1 million-point design space in GPGPU-Sim.

Conclusions:
- Regression methods can automatically and accurately model complex trade-offs in GPU design spaces.
- 4 orders of magnitude reduction in design space evaluation time with less than 1.1% average error.

Optimizing GPU Cache Utility

Reference: Characterizing and Improving the Use of Demand-Fetched Caches in GPUs, Wenhao Jia, Kelly A. Shaw, and Margaret Martonosi, Intl. Conf. Supercomputing 2012

Observation: GPU caches have unpredictable and even detrimental performance impact.

Our Work: Automate GPU cache payoff analysis.

Analysis 1: Cache hit rate is a poor predictor of performance payoff.

Analysis 2: Cache-induced memory traffic reduction is a better performance predictor.

Result 2: Based on the taxonomy, a compile-time algorithm can intelligently enable or disable caching on a per-instruction basis to improve performance.

- Step 1: Compute load addresses of load instructions.
- Step 2: Estimate cache-on and cache-off traffic.
- Step 3: Decide whether to cache for each instruction based on cache-on traffic vs. cache-off traffic.

In real-system evaluation with an NVIDIA Tesla C2070, this algorithm improves the average benefit of caching from 5.8% to 18%.

Conclusions:
- Conserving memory bandwidth instead of hiding latency is GPU caches’ main purpose.
- A locality-based taxonomy helps programmers and tools predict GPU cache utility.
- A compile-time caching control algorithm improves the benefit of caching by 3X.