Accelerating Irregular Algorithms on GPGPUs Using Fine-Grain Hardware Worklists

1. **Abstract**

In this paper, we propose a novel fine-grain hardware worklist for GPGPUs that addresses the classic weaknesses of data-driven implementations of irregular algorithms. A set of distributed hardware worklists banks are tightly integrated with the GPGPU lane to reduce memory contention and software overheads. We also detail multiple work redistribution schemes of varying complexity that can be employed to improve load balancing. Furthermore, using a virtualization mechanism to support seamless work spilling and inter-core redistribution, we evaluate challenging irregular algorithms from the LonestarGPU benchmark suite on a cycle-level simulator. We find that using hardware worklists on a GPGPU yields speedups ranging from 1.2–2.4 over highly optimized software baselines on a nominal GPGPU.

2. **Motivation**

GPGPUs excel at exploiting conventional data parallelism to achieve high performance and energy efficiency. However, it is much more challenging to map more irregular and memory bound applications to GPGPUs which allows tasks to have conflicting accesses, to be generated dynamically, and to modify the work structure. Even aggressive software optimizations do not fully mitigate issues with memory contention, suboptimal load balancing, and software overheads.

3. **Mapping Irregular Algorithms to GPGPUs**

Irregular algorithms frequently apply a set of operations to a subset of elements in the data structure which are referred to as active nodes. The check operator determines whether or not the element assigned to the thread is an active node or not. The compute operator performs the actual work required for the algorithm to progress and can generate more work by activating inactive nodes. There are two standard approaches to mapping irregular algorithms to GPGPUs:

- **Topology-Driven Approach**
  - Work is determined based on thread index
  - All elements are visited whether or not they are active
  - Number of threads spawned is equal to the number of elements

- **Data-Driven Approach**
  - Work is determined by accessing a shared software worklist
  - Only active nodes are visited
  - Number of threads spawned is equal to the number of hardware threads

4. **Fine-Grain Hardware Worklists**

Fine-grain hardware worklists (HWWL) are implemented as distributed banks tightly integrated with the GPGPU lanes in order to reduce memory operations when interacting with the worklist. A work redistribution unit facilitates dynamic load balancing between banks within a core as well as across cores via a special redistribution network. A virtualization unit allows work that does not fit in the banks to seamlessly spill to an overflow buffer in memory and refill empty banks as necessary.

5. **Evaluation**

We used GPGPU-Sim 3.0 with four cores (16 lanes each) and a FIFO-based DRAM model. We compared the performance of highly optimized topology- and data-driven implementations of irregular algorithms from the LonestarGPU benchmark suite running on a nominal GPGPU to double-buffered data-driven implementations using fine-grain hardware worklists. All results are normalized to the best of the two software baselines.

6. **Acknowledgments**

This work was supported in part by NSF CAREER Award #1149444, an NDSEG Fellowship, and donations from Intel Corporation and NVIDIA Corporation. The authors wish to thank the Lonestar Center’s Compute Team for their insightful advice, particularly Martin Burchard, Rajesh Narasimha and Sreenath Pail for generously providing additional code and their extensive help in understanding the various implementations of LMG benchmarks.


Contact Author: Ji Kim, 364 Upson Hall, Ithaca, NY 14853; jikim@cs.cornell.edu