REDwood: A Framework for Accelerating Traverse-REDuce Applications on Heterogeneous Systems

Student (Graduate): Yanwen Xu
University of California, Santa Cruz

Adviser: Tyler Sorensen
University of California, Santa Cruz

1 PROBLEM AND MOTIVATION

The end of Moore's law and Dennard's scaling has led to an explosion of specialized processing units (PUs); e.g., less than 20% of the die area of an iPhone contains the CPU; the rest contains GPU and other acceleration PUs [3]. While many applications are mapped exclusively to one PU, to fully take advantage of these systems, applications must be decomposed and executed across PUs.

In this work, we identify a class of applications, which we call *traverse-reduce* applications, that have a flexible heterogeneous decomposition. Traverse-reduce applications utilize spatial partitioning trees: the tree is repeatedly traversed, and reduction operators are applied to the visited nodes. Spatial partitioning provides the flexibility to tune leaf node sizes to a given heterogeneous system. A large leaf node size will reduce the number of traversed tree nodes but increase the number of elements to compute; thus, the overall application will be bottlenecked by node computations. On the other hand, a small leaf node size will reduce the number of elements to be computed; thus, the overall application will be bottlenecked by the irregular memory accesses from the tree traversal.

To efficiently implement these applications on heterogeneous systems, we propose REDwood: a framework that provides a high-level interface to efficiently implement a wide range of traverse-REDuction algorithms. The REDwood backends will offload the reduction computation to accelerator PUs and manage data communications and synchronization.

2 BACKGROUND

Traverse-reduce applications. Currently, we have implemented two traverse-reduce applications in REDwood. The n-body problem and k-nearest neighbor (KNN) applications are classic problems in scientific computing and statistics, where the naïve implementation requires computing a pairwise interaction of points in a large dataset, giving a runtime of $O(n^2)$. An effective way to accelerate n-body and KNN is to use spatial trees, such as quadtree, octree, and kd-tree [2]. These spatial trees organize items in the dataset into a tree structure based on their spatial coordinates to speed up the comparison process; for instance, Barnes-Hut (BH) algorithm has a complexity of $O(n \log n)$.

GPU SoC Heterogeneous System. While REDwood aims to support many types of accelerator PUs, our implementation currently targets GPU SoC heterogeneous systems. GPUs consist of many simple cores that excel at massively parallel computation, thus are better at reducing dense data at leaf nodes but are less suited to tolerate irregular memory accesses from tree traversals. On the other hand, CPUs have complex hardware components such as load/store queues to tolerate memory latency, thus making them strong candidates for tree traversals. To best utilize the strengths of different PUs, applications can be decomposed across PUs, for example, as done with graph algorithms in [1].

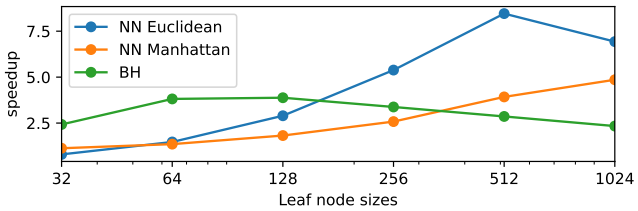
3 TECHNIQUES

REDwood provides a high-level interface and runtime that allows programmers to decompose their traverse-reduce applications into traversal and reduction phases. In the traverse phase, CPU threads can access the spatial tree structure and check traversal conditions. The reduction phase can be efficiently computed on the accelerator PU in parallel. REDwood combines three techniques to for efficient heterogeneous computation:

- **Configurable leaf node size** allows REDwood to adapt to various heterogeneous systems with different relative throughput between the CPU and the accelerator PU.
- **Double buffering** enables REDwood to execute the traverse and reduction phases in parallel with minimum data synchronization overhead.
- **Lightweight coroutines** allow a single CPU thread to execute many traversals concurrently to avoid stalling when a particle's traversal depends on a reduction.

4 PRELIMINARY RESULTS

As our initial benchmarks, we implemented three traverse-reduce applications in REDwood: Barnes-Hut, Nearest Neighbor (NN) with Manhattan distance, and NN with Euclidian distance. We configured REDwood to target a CPU-GPU backend using C++ and CUDA, respectively. Experiments are executed on an Nvidia Jetson Nano with 128-core Maxwell GPU and Quad-core ARM A57 CPU. For inputs, we use 1 million particles uniformly distributed in $\mathbb{R}^3$ for BH and $\mathbb{R}^4$ for NN. Our baselines are CPU sequential implementations of the algorithms without REDwood. The figure below shows how our three applications' speedup varies over leaf node size.



REDwood achieves maximum speedups of 8.46×, 3.88×, and 2.56× at leaf node sizes 512, 1024, 128 for NN Euclidean, NN Manhattan, and BH, respectively. We aim to implement more applications in REDwood, e.g., from [2], and provide backends that target more PUs, e.g., FPGAs.

REFERENCES

[1] Abdullah Gharaibeh, Tahsin Reza, Elizeu Santos-Neto, Lauro Beltrao Costa, Scott Sallinen, and Matei Ripeanu. 2013. Efficient large-scale graph processing on hybrid CPU and GPU systems. *arXiv preprint arXiv:1312.3018* (2013).
[2] Nikhil Hegde, Jianqiao Liu, Kirshanthan Sundararajah, and Milind Kulkarni. 2017. Treelogy: A benchmark suite for tree traversals. In *2017 IEEE ISPASS*. IEEE.
[3] Yakun Sophia Shao, Brandon Reagen, Gu-Yeon Wei, and David Brooks. 2015. The aladdin approach to accelerator design and modeling. *IEEE Micro* 35, 3 (2015).