arXiv:1708.04567v1 [cs.PF] 15 Aug 2017

GARDENIA: A Domain-specific Benchmark Suite for Next-generation Accelerators

Zhen Xu, Xuhao Chen, Jie Shen, Yang Zhang, Cheng Chen, Canqun Yang

College of Computer, National University of Defense Technology, Changsha 410073, China

Abstract—This paper presents the Graph Analytics Repository for Designing Next-generation Accelerators (GARDENIA), a benchmark suite for studying irregular algorithms on massively parallel accelerators. Existing generic benchmarks for accelerators have mainly focused on high performance computing (HPC) applications with limited control and data irregularity, while available graph analytics benchmarks do not apply state-of-the-art algorithms and/or optimization techniques. GARDENIA includes emerging irregular applications in big-data and machine learning domains which mimic massively multithreaded commercial programs running on modern large-scale datacenters. Our characterization shows that GARDENIA exhibits irregular microarchitectural behavior which is quite different from structured workloads and straightforward-implemented graph benchmarks.

Index Terms—benchmark suite, graph analytics, massive multithreading, irregular workloads

1 Introduction

With the rise of big-data analysis and machine learning, graph analytics applications have become increasingly important and pervasive. For these frequently executed algorithms, dedicated hardware accelerators [1], [2], [3] are an energy-efficient avenue to high performance. Meanwhile, general purpose accelerators (e.g., GPUs and MICs) which have been widely deployed in many HPC systems and datacenters, are also applied to speedup applications in these areas, such as graph analytics [4], machine learning [5] and sparse linear algebra [6]. However, these applications exhibit irregular runtime behavior which is quite different from that of conventional well-studied HPC applications. Therefore, it is important to set up a standard benchmark suite for architecture researchers to get deep understanding on these applications.

The goal of GARDENIA[1] is to create a suite of *emerging irregular* workloads that can drive accelerator architecture research, including both general-purpose and domain-specific accelerator design. For general-purpose accelerators, GADENIA is an important complement for generic benchmark suites (e.g.,Rodinia and Parboil) which have limited irregularity, and architectural support for irregular algorithms can be exploited by running the benchmarks and locating the performance bottlenecks. For domain-specific accelerators, GARDENIA can be used as a workload representative of real-world big-data analytics applications, and the hardware can be specialized for these algorithms. Besides, GARDENIA benchmarks can be also used by algorithm, library and compiler designers as reference implementations for comparison.

To the best of our knowledge, GARDENIA is the first graph analytics benchmark suite specifically targeting accelerator architecture reserach. It is different from previous GPU graph analytics benchmark suites in many ways: 1) The benchmark implementations incorporate *state-of-the-art* optimization techniques for massively parallel accelerators; 2) The input graphs are selected appropriately for our target platforms, in terms of graph size, desity and topology; 3) The workloads are representatives of popular graph analytics, machine learning and sparse linear algebra applications running on modern datacenters.

1. The source code can be found at github.com/chenxuhao/gardinia

2 Motivation

This work aims to define a benchmark suite that can be used to design next-generation accelerators capable of accelerating real-world irregular (not only HPC) applications on modern datacenters. In this section, we first discuss what are the requirements for such a suite. We then point out the limitations of existing benchmark suites and illustrate why they cannot meet these requirements.

2.1 Benchmark Requirements

A benchmark suite of real-world irregular workloads on accelerators needs to meet the following requirements:

Massive Parallelism Massively parallel accelerators, such as GPUs and MICs (Xeon Phi coprocessors) have been already widely deployed in today's high performance servers and datacenters. Essential big data analysis and machine learning engines that drive many important applications are highly dependent on accelerators to achieve required performance. The trend for future accelerators is to deliver tremendous performance improvements through extreme throughput and memory bandwidth. Consequently, applications that require additional processing power need to be massively parallel to take advantage of the hardware.

Workload Irregularity Regular applications have been intensively investigated on large-scale parallel systems in the past decade. Many scientific and commercial applications, have been successfully mapped onto accelerators. Despite this progress, data-parallel accelerators (DPAs) continue to be confined to structured parallelism. While structured parallelism maps directly to DPAs, much more unstructured applications in real-world cannot simply take advantage of them. Future accelerators will be designed to meet the demands of these irregular applications, and our benchmark suite should represent them.

State-of-the-Art Optimization Techniques Plenty of efforts have been made to map general purpose applications onto GPUs during the past decade, including parallel algorithm innovations and optimization techniques. A benchmark suite should not only represent emerging applications but also use state-of-the-art optimization techniques. This is important because straightforward-implemented workloads may exhibit quite different microarchitectural behaviors compared to optimized ones, which are unqualified to represent real-world applications and may mislead architecture design.

Workload and Dataset Diversity With the rise of heterogeneous computing, real-world workloads are increasingly diverse. They are written in different parallelization models, run on a variety of computing platforms, and cover a wide range of application areas. Besides, the behaviors of these irregular applications change dramatically with different types of input datasets. To cover different program characteristics and behaviors, a diverse collection of both workloads and input datasets is required, so that architects can thoroughly understand the application behaviors and make reasonable trade-offs when designing an accelerator.

2.2 Existing Benchmark Suites and their Limitations

Next we analyze how existing benchmark suites fall short of the presented requirements.

Generic Benchmark Suites for Accelerators. *Rodinia* [7] and *Parboil* [8] are widely used GPU benchmark suites. They provide CUDA, OpenCL and OpenMP implementations. Despite their popularity, most of their benchmarks are structured kernels from HPC applications with limited control and memory irregularity. They are not good candidates for studying irregular applications on accelerators.

Graph Benchmarks for CPUs. *GAPBS* [9] is a graph processing benchmark suite on CPUs. It is a collection of high-performance implementations written in OpenMP. Their implementations are representatives of state-of-the-art performance on multicore CPUs. However, GAPBS does not include any implementation for accelerators.

Graph Benchmarks for GPUs. *Pannotia* [10] assembles a set of graph algorithms implemented in OpenCL. The irregularities of these kernels are characterized on GPUs. With limited knowledge on how to optimize graph algorithms on GPUs at that time, no specific optimization is applied. We will show that workloads which lack of state-of-the-art optimization techniques behaves quite differently from optimized ones in Section 4. *GraphBIG* [11] includes both OpenMP implementations for multicore CPUs and CUDA implementations for GPUs. Mainly focused on CPU versions, most of the GPU workloads are also straightforward implementations, although some of the GPU workloads achieve substantial speedup. *Lonestargpu* [12] includes a couple of irregular CUDA benchmarks with advanced optimizations, but it does not focus on graph analytics, and thus it is not an appropriate candidate for designing domain-specific accelerators. Besides, the datasets provided are not sufficient for deeply understanding graph algorithms.

3 The GARDENIA Benchmark Suite

One of the goals of the GARDENIA benchmark suite is to assemble a program collection that represents important irregular applications running on the accelerators of modern datacenters for big-data analysis and machine learning. GARDENIA meets all the requirements outlined in Section 2:

- Each benchmark has been parallelized using OpenMP for multicore CPUs, CUDA for GPUs and OpenMP target for MICs to exploit massive parallelism.
- GARDENIA is not designed for regular HPC programs which have been well studied on accelerators in the last decade. It focuses on emerging irregular workloads that mimic real-world big-data applications.
- All the benchmarks apply state-of-the-art optimization techniques, i.e., the benchmarks are reasonably optimized to get substantial speedups on GPUs and MICs.
- The workloads as well as their datasets are diverse and are chosen from many different application areas.

3.1 Optimization Techniques

We apply the state-of-the-art algorithm innovations and optimization techniques that have been widely employed in academic and industry libraries [13], [6], [14]. Benchmarks are neither over-optimized nor unoptimized (i.e. straightforward-implemented) for the target accelerator. Techniques bound to specific architectures are not included in our suite.

Mapping Strategies. Basically there are two parallelism strategies for graph algorithms: *quadratic* mapping and *linear* mapping [15]. Quadratic mapping checks every vertex of the graph in each iteration, and a thread may stay idle or get to process its vertex depending on whether the vertex needs to be processed or not [16]. Quadratic mapping is intuitive and used for sequential-movement benchmarks. By contrast, linear mapping maintains a frontier queue holding the vertices to be processed in the current iteration. Threads are created in proportion to the size of the frontier. Each thread is responsible for processing some of the vertices in the frontier, and no thread is idle [17]. Therefore, linear mapping is generally more work-efficient than the quadratic mapping, but it needs extra overhead to maintain the frontier. We use linear mapping for traversal-based benchmarks (see section 3.3 for details).

Load Balancing. Load imbalance problem is one of the key issues for irregular algorithms, and is particularly worse for scale-free (power-law) graph datasets. Hong *et al.* [18] proposed a warp-execution method for BFS to map warps rather than threads to vertices. This scheme is applied in most of the GARDENIA benchmarks. Based on Hong's scheme, Merrill *et al.* [17] proposed a hierarchical load balancing strategy which maps the workload of a single vertex to a thread, a warp, or a thread block, according to the size of its neighbor list. This multi-level scheme turns out to be quite efficient on GPUs, and is employed in traversal-based benchmarks in GAREDNIA.

Push vs. Pull etc. Beamer *et al.* [19] proposed direction-optimizing BFS which uses two different methods to drive BFS: `push` and `pull`. The `push` method examines vertices in the current frontier and scatters status to its outgoing neighbors to create the new frontier. The `push` method is intuitive and commonly used. Conversely, the `pull` method checks unvisited vertices whose parents are in the current frontier and gathers status from the incoming neighbors. The `pull` method is beneficial when the frontier size gets larger than the number of unvisited vertices [13].

In addition, we applied many other optimization techniques, such as kernel fusion, reordering queue [13], read-only data caching, bit operations, vectorization for MIC, and so on.

3.2 Input Datasets

A major characteristic of irregular applications is the input dependency. Research has shown that graph properties substantially affect performance [9]. Therefore, we should characterize graph workloads not only by the algorithms, but also by the properties of graph instances used.

Graph Size Due to the rapid increase of data size in modern datacenters, it has become problematic to manage and manipulate the large volume of data with relatively limited hardware resources. Our selected datasets should include large enough graph instances for big-data analysis. On the other hand, to support architecture research, small but representative datasets are also needed to guarantee that the simulation finishes in reasonable time.

Graph Density/Sparsity The density of a graph is usually represented by the average degree of all vertices. This parameter affects the computation intensity, irregularity, and the amount of locality exists in graph algorithms. A denser graph is likely to have more data reuse since vertices may be visited or updated more times. High density also means more regular memory access behavior. However, denser graphs have more edges, which implies larger working set. Note that real-world graphs are usually sparse, making graph analytics irregular.

Graph Topology In [9], graphs are divided into two categories in terms of topology: meshes and social networks. Meshes usually come from physically spatial sources (such as road maps), and tend to have a high diameter and a low degree distribution. By contrast, social networks come from non-spatial sources, and have a low diameter (small-world) and a power-law degree distribution (scale-free). Since there can be orders of magnitude difference between the degrees (number of neighbors) of different vertices in a social network, load balancing is a major issue for parallel execution [9].

In summary, we select datasets from the University of Florida Sparse Matrix Collection [20], the SNAP database [21], and the Koblenz Network Collection [22] which are all publicly available. These graphs vary widely in size, density, topology and application domain. For example, `twitter` is a large "social network" graph that can be used to evaluate real machines, while `road` is a relatively small-sized graph with "mesh" topology that can be used in simulation.

3.3 Workloads

The GARDENIA suite includes the following workloads:

Breadth-First Search (BFS) is a key graph primitive which traverses a graph in breadth-first order [17]. BFS is a fundamental `traversal-based` graph algorithm. We implement the linear mapping strategy with Merrill's load balancing technique.

Single-Source Shortest Paths (SSSP) computes the shortest distance to all reachable vertices from a given source vertex [23]. We implement delta-stepping algorithm for CPUs and Bellman-Ford algorithm for GPUs. SSSP is a traversal-based algorithm and we apply similar techniques as in BFS.

Betweenness Centrality (BC) is commonly used in social network analysis to measure the influence of a vertex in a graph [24]. It computes each vertex's betweenness centrality score which is the fraction of shortest paths between all vertices that pass through the vertex. BC is also traversal-based.

PageRank (PR) ranks a website based on the score of its neighboring sites (i.e. the sites that link to it) [25]. Each vertex is given a initial score at the beginning. At each iteration, it scans each vertex and updates its score using the its neighbors' scores and degrees. It iterates until the score change is small enough. Unlike above traversal-based benchmarks, PR has `sequential-movement` access pattern and so do all the following benchmarks.

Connected Components (CC) assigns the same and unique label to all vertices in the same connected component [26]. In each iteration, the label is propagated from a smaller-id vertex to its larger-id neighbors. The algorithm iterates until convergence (no propogation happens).

Triangle Counting (TC) counts the number of triangles in an undirected graph [27]. It is essential in graph statistics applications (e.g. clustering coefficients). To find triangles, it computes the intersection of each vertex's neighbor list and its neighbor's neighbor lists.

Stochastic Gradient Descent (SGD) is an approach to solve the matrix factorization problem [28] in recommender systems. It decomposes the $ratings$ matrix into two feature vectors ($users$ and $items$), and makes a rating prediction using the dot-product of $users$ and $items$. It iterates until the error between prediction and the actual rating is small enough.

Sparse Matrix-Vector Multiplication (SpMV) is the most important sparse linear algebra primitive [27]. It examines the non-zero elements in the matrix row by row and multiplies the corresponding elements in the vector.

Symmetric Gauss-Seidel smoother (SymGS) is a key operation in the High Performance Conjugate Gradients (HPCG) [14]. SymGS performs a forward and backward triangular solve whose operation is similar to SpMV but in a data-dependent order. We use vertex coloring [16] to find parallelism and arrange the order of tasks.

4 Evaluation

In this section, we evaluate GARDENIA in terms of workload irregularity, optimization effects, and workload/dataset diversity, showing the necessity of developing this irregular workload benchmark suite. Our evaluation is performed on the NVIDIA K40m GPU with CUDA Toolkit v8.0.

4.1 Workload Irregularity

Branch divergence and memory divergence are two critical factors that reflect workload irregularity. We use two metrics, BDR and MDS, to measure them. BDR (Branch Divergence Rate) is the average ratio of inactive threads per warp, which reflects the amount of branch divergence. MDS (Memory Divergence Scale) is the average number of replays for each instruction executed. The replay will be repeated until all requested data are ready, so MDS may be greater than 1.

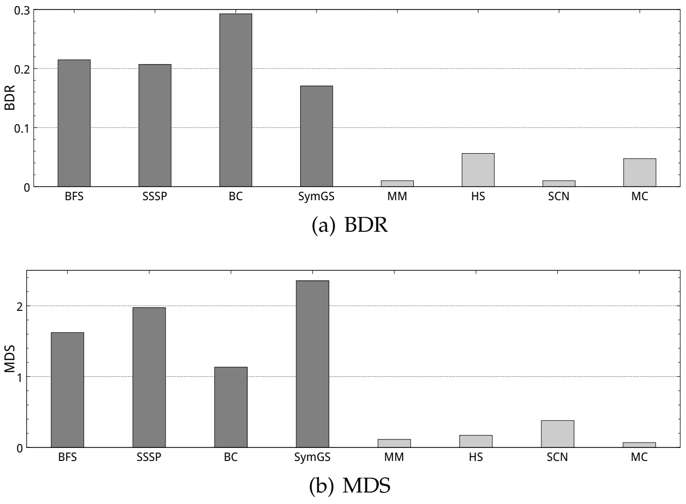

Fig. 1. The difference between irregular and regular workloads.

We select 4 workloads from GARDENIA (irregular) and 4 workloads from NVIDIA SDK and Rodinia (regular) for comparison. The regular workloads covers a wide range of application domains: Matrix Multiplication (MM) is from the dense liner algebra area; Hotspot (HS) does physical simulation; Scan (SCN) is widely used parallel computing primitives; Monte Carlo (MC) is a financial application. We ran all the workloads with the same input dataset. Figure 1 shows that GARDENIA workloads have much higher BDR and MDS compared to the regular ones, which demonstrates that GARDENIA effectively selects irregular workloads in real-world big-data applications. The high irregularity makes GARDENIA workloads behave significantly different from those structured benchmarks found in previous generic suites, which implies the necessity of constructing a irregular benchmark suite like GARDENIA.

4.2 Optimization Effects

We employ state-of-the-art optimization techniques for each GARDENIA workload. This is essential because straightforward implementations may show largely different microarchitectural behaviors and mislead architecture design.

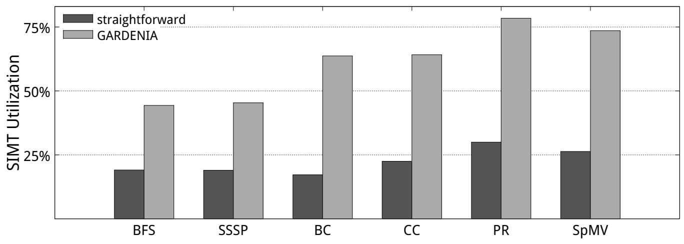

Fig. 2. Optimization effects measured by the SIMT utilization.

Figure 2 compares the SIMT utilization of the *straightforward* implementation (similar to those provided in Pannotia and GraphBIG) and the *GARDENIA* implementation with optimization applied. We observe that the SIMT utilization of the GPU is largely increased by applying optimization. This huge difference indicates that optimization techniques can substantially change program behavior, and it is necessary to include state-of-the-art techniques in order to closely mimic the behavior of real-world applications which are most likely built using these optimizations.

4.3 Workload and Dataset Diversity

To represent emerging irregular workloads for next-generation accelerators, GARDENIA selects representative workloads from various application domains. Furthermore, as graph processing performance depends not only on the workload but also on the dataset to run, a careful collection of graph datasets is of necessity.

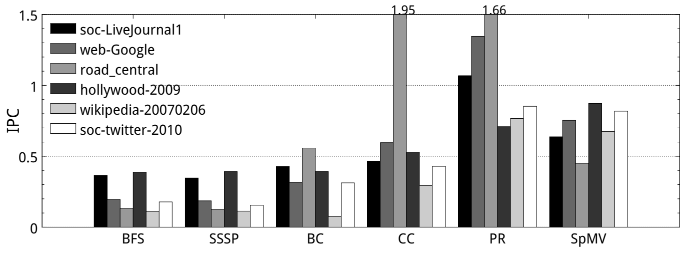

Fig. 3. The program IPC for different workloads and datasets.

Figure 3 shows the program IPC for different workloads and datasets. In each workload cluster, IPC varies widely with the datasets. For the same dataset, different workloads may lead to entirely different IPCs . For example, for `road_central`, BFS and SSSP have low IPCs which are below 0.2, while CC and PR have much higher IPCs (above 1.6). Thus, architecture research needs a diverse set of workloads (in terms of application domains, program behaviors) and input datasets (in terms of graph size, density, and topology) to make microarchitecture characteristics fully manifest, and GARDENIA is designed to meet this requirement.

5 Conclusion

In this paper, we present GARDENIA, a domain-specific benchmark suite for studying irregular algorithms on massively parallel accelerators. With workloads and datasets carefully selected as well as state-of-the-art techniques included, GARDENIA is designed to represent emerging real-world applications on modern datacenters, and to facilitate accelerator architecture research. Our characterization illustrates that GARDENIA benchmarks exhibit quite different microarchitectural behavior from structured workloads and straightforward-implemented graph benchmarks, and thus demonstrating the necessity of constructing such a suite.

References

[1] J. Ahn *et al.*, "A scalable processing-in-memory accelerator for parallel graph processing," in *ISCA-42*, 2015, pp. 105–117.
[2] M. M. Ozdal *et al.*, "Energy efficient architecture for graph analytics accelerators," in *ISCA-43*, 2016, pp. 166–177.
[3] T. J. Ham *et al.*, "Graphicionado: A high-performance and energy-efficient accelerator for graph analytics," in *MICRO-49*, pp. 1–13.
[4] NVIDIA, "nvGRAPH Library," 2016. [Online]. Available: http://docs.nvidia.com/cuda/nvgraph/index.html
[5] NVIDIA, "cuDNN Library," 2016. [Online]. Available: https://developer.nvidia.com/cudnn/
[6] NVIDIA, "CUSPARSE Library," 2016. [Online]. Available: http://docs.nvidia.com/cuda/cusparse/
[7] S. Che *et al.*, "Rodinia: A benchmark suite for heterogeneous computing," in *IISWC '09*, 2009, pp. 44–54.
[8] J. A. Stratton *et al.*, "Parboil: A revised benchmark suite for scientific and commercial throughput computing," UIUC, Tech. Rep. IMPACT-12-01, 2012.
[9] S. Beamer *et al.*, "Locality exists in graph processing: Workload characterization on an ivy bridge server," in *IISWC'15*, pp. 56–65.
[10] S. Che *et al.*, "Pannotia: Understanding irregular gpgpu graph applications," in *IISWC'13*, 2013, pp. 185–195.
[11] L. Nai *et al.*, "Graphbig: Understanding graph computing in the context of industrial solutions," in *SC '15*, 2015, pp. 69:1–69:12.
[12] M. Burtscher *et al.*, "A quantitative study of irregular programs on gpus," in *IISWC '12*, 2012, pp. 141–151.
[13] Y. Wang *et al.*, "Gunrock: A high-performance graph processing library on the GPU," in *PPoPP'16*, 2016.
[14] S. Dalton *et al.*, "Cusp: Generic parallel algorithms for sparse matrix and graph computations," 2014, version 0.5.0. [Online]. Available: http://cusplibrary.github.io/
[15] R. Nasre *et al.*, "Data-driven versus topology-driven irregular computations on gpus," in *IPDPS '13*, 2013, pp. 463–474.
[16] X. Chen *et al.*, "Efficient and high-quality sparse graph coloring on the gpu," *Concurrency and Computation: Practice and Experience*, vol. 01, no. 01, pp. 0–22, jun 2016.
[17] D. Merrill *et al.*, "Scalable gpu graph traversal," in *PPoPP '12*, 2012, pp. 117–128.
[18] S. Hong *et al.*, "Accelerating cuda graph algorithms at maximum warp," in *PPoPP '11*, 2011, pp. 267–276.
[19] S. Beamer *et al.*, "Direction-optimizing breadth-first search," in *SC '12*, 2012, pp. 12:1–12:10.
[20] "The University of Florida Sparse Matrix Collection," 2011. [Online]. Available: http://www.cise.ufl.edu/research/sparse/matrices/
[21] "Snap: Stanford network analysis platform," 2013. [Online]. Available: http://snap.stanford.edu/data/index.html
[22] "Koblenz network collection," 2013. [Online]. Available: http://konect.uni-koblenz.de
[23] A. Davidson *et al.*, "Work-efficient parallel gpu methods for single-source shortest paths," in *IPDPS'14*, 2014, pp. 349–359.
[24] A. McLaughlin and D. A. Bader, "Scalable and high performance betweenness centrality on the gpu," in *SC'14*, 2014, pp. 572–583.
[25] L. Page *et al.*, "The pagerank citation ranking: Bringing order to the web," 1998.
[26] S. Beamer *et al.*, "The GAP benchmark suite," *CoRR*, vol. abs/1508.03619, 2015.
[27] X. Yu *et al.*, "Imp: Indirect memory prefetcher," in *MICRO-48*, 2015, pp. 178–190.
[28] Y. Koren *et al.*, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009.