An Automated Framework for Characterizing and Subsetting GPGPU Workloads

Vignesh Adhinarayanan and Wu-chun Feng
Department of Computer Science, Virginia Tech
Blacksburg, VA 24061, U.S.A.
{avignesh, wfeng}@vt.edu

***Abstract*—Graphics processing units (GPUs) are becoming increasingly common in today's computing systems due to their superior performance and energy efficiency relative to their cost. To further improve these desired characteristics, researchers have proposed several software and hardware techniques. Evaluation of these proposed techniques could be tricky due to the ad-hoc nature in which applications are selected for evaluation. Sometimes researchers spend unnecessary time evaluating redundant workloads, which is particularly problematic for time-consuming studies involving simulation. Other times, they fail to expose the shortcomings of their proposed techniques when too few workloads are chosen for evaluation.**

To overcome these problems, we propose an *automated framework that characterizes and subsets GPGPU workloads*, depending on a user-chosen set of performance metrics/counters. This framework internally uses principal component analysis (PCA) to reduce the dimensionality of the chosen metrics and then uses hierarchical clustering to identify similarity among the workloads. In this study, we use our framework to identify redundancy in the recently released SPEC ACCEL OpenCL benchmark suite using a few architecture-dependent metrics. Our analysis shows that a subset of *eight* applications provides most of the diversity in the 19-application benchmark suite. We also subset the Parboil, Rodinia, and SHOC benchmark suites and then compare them against each another to identify "gaps" in these suites. As an example, we show that SHOC has many applications that are similar to each other and could benefit from adding four applications from Parboil to improve its diversity.

I. Introduction

Accelerators such as graphics processing units (GPUs) are becoming increasingly common in today's high-performance computing (HPC) systems due to their superior performance and energy efficiency relative to their cost. This can be seen from the increasing share of accelerators in the TOP500 list which ranks supercomputers in terms of performance. In the latest June 2015 list, a total of 90 systems use accelerators up from 75 systems six months ago [1].

To meet the computational and energy efficiency demands put forth by HPC applications, researchers have proposed several software and hardware solutions for GPUs. However, the current ad-hoc approach to evaluate such research techniques is fraught with danger. Typically, a random set of applications relevant to the HPC community is put together and used for such studies. Alternatively, a pre-existing benchmark suite such as Parboil [2], Rodinia [3], [4], or SHOC [5] is used for evaluation, without much thought into the nature of the applications included in these suites. A rigorous evaluation requires a diverse set of applications that is representative of the targeted domain. However, simply increasing the number of applications for evaluation is generally a bad idea for time-consuming studies that involve simulation. Furthermore, this approach may end up (unintentionally) over-emphasizing certain types of workloads.

To combat the above problem, at least to some extent, the SPEC committee put together the SPEC ACCEL benchmark suite [6]. This suite consists of 19 OpenCL applications and 12 OpenACC applications, which are supposed to be representative of the HPC domain and to stress the various components of an HPC accelerator, i.e., GPU. However, it is not clear whether this suite is well balanced and avoids redundancy in its coverage of the application space.

In this study, we study the 19 applications from the SPEC ACCEL OpenCL suite, identify redundancy among these applications, and subset the suite using the well-known principal component analysis (PCA) and clustering analysis techniques. We also perform this subsetting for other prominent and well-established GPGPU workloads from Parboil, SHOC, and Rodinia. Then, we compare applications across suites to identify areas where these suites could be improved. In all, we provide a way to systematically identify relevant and well-balanced applications for evaluating various techniques targeted at the GPU. Our major contributions are the following:

- A set of architecture-dependent metrics that are most important for characterizing high-performance computing (HPC) GPGPU workloads for the purpose of evaluating modern accelerators.
- A methodology that captures best approaches from previous studies in order to systematically study GPGPU workloads.

- A concrete illustation of redundant workloads in the production-oriented SPEC ACCEL benchmark suite and the academia-oriented Rodinia, Parboil, and SHOC benchmark suites as well as a proposed subsetting of these benchmark suites to eliminate such redundancy.
- Recommendations for expanding the existing benchmark suites if they lack specific application coverage.

Our major findings are noted as follows. First, **it is possible to successfully subset GPGPU workloads with architecture-dependent metrics.** We validate this with the publicly available speedup results for SPEC ACCEL for *18* different hardware platforms by comparing speedup results calculated with the original SPEC ACCEL suite against the subsetted suite. Second, **SPEC ACCEL and Parboil exhibit the highest diversity, while Rodinia and SHOC not only show lower diversity, but also more redundant workloads.**

The rest of the paper is organized as follows. We discuss related work in Section II and distinguish our work from others in this field. We describe our hardware and the workloads used in Section III and explain our methodology in Section IV. The characterization, subsetting, and comparison results are presented in Section V and we conclude in Section VI.

II. Related Work

While there exists a significant body of work in the area of characterizing and subsetting workloads, they differ from one another in the specific task they seek to accomplish, the hardware platforms they are targetted at, the domain they focus on, the metrics, and the techniques they use. In this section, we discuss how these works differ from one another and how our work differs from these.

First, we classify the related work in terms of the task they seek to accomplish. *Characterization* focuses on studying each application within a benchmark suite based on a certain metric of interest. This may be followed up with a *diversity analysis*, where applications within a benchmark suite are compared against one another to find the ones that are similar or dissimilar to each other. *Subsetting* goes one step further in composing a well-balanced and well-represented suite by employing a formal methodology to analyze redundancy and remove applications that do not provide any additional information (or value). This is also usually followed by a validation step, which would show that we indeed have a representative suite after subsetting. *Input selection* is closely related to the above, where the emphasis is on selecting representative input instead of selecting representative applications. Finally, the *comparison and expansion* task looks at several different benchmark suites to identify gaps in existing or emerging workloads. Table I chronicles the effort made in this field. The emphasis in our work is on the subsetting and expansion tasks for less-explored devices such as the graphics processing unit (GPU) and for the associated high-performance computing (HPC) domain, which has not been done previously.

While previous work has explored dimensionality-reduction techniques such as principal component analysis (PCA) [7], correlation elimination [11], genetic algorithm [11], and Plackett and Burmann (P&B) design technique [10] in combination k-means and hierarchical clustering techniques [7], the combination of PCA and hierarchical clustering has proven to be the most successful [10]. We adopt and apply these best practices in our work. While micro-architecture independent metrics have proven to be more successful in CPUs for the subsetting task, the limited support for GPU application profiling does not allow such detailed level of profiling. Therefore, we use architecture-dependent metrics in our work. We validate our subsetting approach using the SPEC ACCEL results reported to and publicly available from SPEC. This is similar to the technique proposed by Phansalkar et al. [13] where the SPEC rating calculated from the original benchmark suite is compared against the subsetted benchmark suite. By adopting the best approaches used in previous studies and applying them to GPGPU workloads, we expand on existing literature by focusing on a new domain.

Next, we distinguish our work from previous work on workload characterization in the GPGPU space. Kerr et al. characterized CUDA workloads from the NVIDIA CUDA SDK and Parboil benchmark suite for the purposes of optimizing these applications [18]. Goswami et al. went further by performing diversity analysis on CUDA SDK, Parboil, and Rodinia on the GPGPU-Sim simulator [19] rather than on real hardware. Che et al. performed a diversity analysis on the Rodinia benchmark suite and compared the breadth of their benchmark suite against the Parsec workloads. Our work goes beyond the above in that we perform the *subsetting* task, which involves removing redundant workloads in a suite to make it well balanced, and a *validation* task in our methodology. To illustrate the efficacy of our automated frameowrk, we work on the recently released *production-oriented* SPEC ACCEL in addition to the academic benchmarks noted above. Finally, we note that our work is performed on a *real* and *modern* hardware systems.

III. Experimental Setup

In this section, we describe the hardware platform and the workloads used in this study.

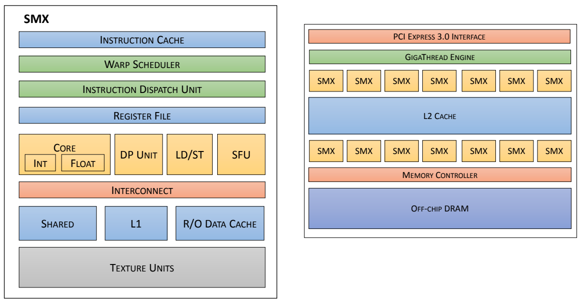
A. Hardware Platform

We characterize several CUDA workloads on a NVIDIA Kepler GK110 GPU [20]. The block diagram for this GPU

TABLE I: Summary of Related Work

Paper	Task	H/W	Benchmarks	Metrics	Method	Validation
Eeckhout et al. [7]	Diversity Analysis & Input Selection	Alpha (CPU)	SPECint95, TPC-D	μ-arch dependent	PCA + Hierarchical Clustering	Same clusters formed for different μ-arch configuration
Phansalkar et al. [8] and Joshi et al. [9]	Subsetting	Alpha AXP-2116 (CPU)	SPEC CPU 2000, MiBench, MediaBench	μ-arch independent	PCA + K-Means	Predict IPC and cache miss rate for entire suite
Yi et al. [10]	Subsetting	CPU Simulation	SPEC CPU 2000	μ-arch dependent	PCA, P&B, 5 non-statistical methods	Mean speedup on different architectures with and without subsetted suite
Hoste et al. [11], [12]	Comparison	Alpha 21164A (CPU)	BioInfoMark, BioMetrics workload, CommBench, MediaBench, MiBench, SPEC CPU 2000	μ-arch independent	PCA/Genetic Algorithm + K-Means with BIC	N/A
Phansalkar et al. [13]	Subsetting	Sun UltraSPARC, x86, Itanium, IBM Power (CPUs)	SPEC CPU 2006, SPEC CPU 2000	μ-arch dependent	PCA + Hierarchical clustering/K-Means	SPEC score without subsetting vs SPEC score with subsetting
Hoste et al. [14]	Comparison (Phase-level)	Intel Pentium 4 (CPU)	BioInfoMark, BioMetrics, MediaBench, SPEC CPU 2000	μ-arch independent	PCA/Genetic Algorithm + K-Means with BIC	N/A
Isen et al. [15]	Comparison	IBM J9 VM (Embedded)	MIDPmark, MorphMark, Caffeine, EEMBC Java, Real mobile applications	VM-level metrics	PCA/Genetic Algorithm + Hierarchical clustering	N/A
Jia et al. [16]	Subsetting	Intel Westmere (CPU)	BigDataBench	μ-arch dependent	PCA + Hierarchical clustering/K-Means	No validation
Panda et al. [17]	Diversity Analysis	Intel Xeon E5345 (CPU)	TPC-H, SPEC CPU 2006, SPECjbb2013	μ-arch independent	PCA + Hierarchical clustering/K-Means	No validation
Kerr et al. [18]	Characterization	Ocelot GPU simulator	CUDA SDK, Parboil	μ-arch dependent	N/A	N/A
Goswami et al. [19]	Characterization and Diversity Analysis	GPGPU-Sim	CUDA SDK, Parboil, Rodinia	μ-arch agnostic	PCA + Hierarchical clustering	N/A
Che et al. [4]	Diversity Analysis & Comparison	NVIDIA GTX480 (GPU)	Parsec, Rodinia	μ-arch dependent	PCA + Hierarchical clustering	N/A
This paper	**Subsetting** & Comparison	**NVIDIA Kepler GTX Titan (GPU)**	**SPEC ACCEL, SHOC** Parboil, Rodinia	μ-arch dependent	PCA + Hierarchical clustering	**SPEC score without subsetting vs SPEC score with subsetting**

is shown in Fig. 1.



(a) Streaming Multiprocessor (b) Full chip

Fig. 1: Block diagram of NVIDIA Kepler

This GPU consists of 15 streaming multiprocessor (SMX) units. Each SMX consists of an instruction cache, four warp schedulers which are responsible for scheduling warps (a group of threads) to the SMX, and eight instruction dispatch units which decides on the instruction to be scheduled in a given clock cycle. There are 192 CUDA cores in each SMX and each has its own integer and single-precision floating point arithmetic logic units (ALUs). Each ALU can perform an add, multiply, or a fused-multiply operation. Each SMX also has double-precision (DP) units, special function units (SFU), and load/store (LD/ST) units for executing corresponding instructions. Apart from these instructions, the GPU is also capable of executing branch instructions, atomic instructions, and shuffle instructions. Each SMX also has a 65,536 x 32-bit register file, 6 KB configurable shared memory and L1 cache, a 48 KB read-only data cache, and several texture units. Common to all the SMXs, there is a 1563 KB L2 cache, six memory controllers, and a 6 GB off-chip DRAM.

B. Workloads

In this study, we subset four different GPGPU benchmark suites, namely SPEC ACCEL, SHOC, Parboil, and Rodinia. For the SPEC ACCEL benchmark suite, instead of using the OpenCL version available via SPEC, we use CUDA-equivalent version from the original sources. This is because of the rich set of performance counters and metrics available for CUDA programs on NVIDIA architecture via *nvprof* interface. Considering the importance of choosing the right set of metrics in performing the analysis, we chose the CUDA version over the OpenCL version for SPEC ACCEL. For the other 3 benchmark suites, several implementations of available and we choose the CUDA implementation for reasons mentioned above.

The description of each application in SHOC, Parboil, and Rodinia is presented in in Table II, Table III, and Table IV respectively. SPEC ACCEL benchmarks are a subset of these applications, so we do not describe it separately.

TABLE II: Summary of SHOC benchmarks

Benchmark	**Description**	**Size**
BFS [21]	Breadth-first search	S4
FFT [22]	512-pt 2-D fast Fourier transforms	S4
MD	Molecular dynamics application that calculates Lennard-Jones potential	S3
MD5-Hash	Hashing function	S4
Reduction	Sum of elements in an array	S4
Scan [23]	Performs prefix sum calculations	S4
GEMM	General matrix-matrix multiplication	S4
Sort [24]	Performs a fast radix sort on several key-value pairs	S4
Spmv [25]	Sparse matrix-vector multiplication (CSR scalar, CSR vector, and ELLPACKR)	S4
Stencil2D	2-D 9-pt stencil computation	S3
Triad [26]	Performs streaming dot-product multiplication	S4
QTC [27]	Quality threshold clustering	S4
S3D	Computes the rate of a chemical reaction	S4

TABLE III: Summary of Parboil benchmarks [2]

Benchmark	**Description**	**Size**
BFS	Breadth-first search	1M nodes
CUTCP	Distance-cutoff Coulombic potential	Default
HISTO	Saturating Histogram	Medium
LBM	Lattice-Boltzmann method fluid dynamics	Default
MM	Dense matrix-matrix multiply	Default
MRI-G	Magnteic resonance imaging on a regular grid	Default
MRI-Q	Magnetic resonance imaging in non-cartesian space	Small
SAD	Sum of absolute differences	Default
SPMV	Sparse-matrix dense-vector multiplication	Medium
STENCIL	3-D stencil operation	Small
TPACF	Two-point angular correlation F=function	Default

TABLE IV: Summary of Rodinia benchmarks

Benchmark	**Description**
Backprop	Train weights in neural network using backward propagation technique
BFS [21]	Breadth-first search
B+ Trees [28]	Traverses a B+ tree
CFD [29]	Computational fluid dynamics for unstructured grids
Gaussian	Gaussian elimination method for solving equations
Heart Wall [30]	Track changing shape of a mouse's heart
HotSpot [31]	Solves differential equation to generate processor's heatmap
K-Means	Clustering of data points
LavaMD [32]	N-body algorithm
Leukocyte [33]	Computing maximal gradient inverse coefficient of variation to track leukocyte
LUD	LU decomposition for solving a system of linear equations
MummerGPU	Local sequence alignment
Mycocyte [34]	Structured grid application to simulate mycocyte cells
NN	Nearest neighbor algorithm
NW	Needleman-Wunsch algorithm for DNA sequence alignment
Particle Filter [35]	Estimates the location of an object from noisy data
Path Finder	Find the shortest path between two points in a 2-D grid
SRAD [34]	Speckle reducing anisotropic diffusion for removing speckles in image
Stream Cluster	Online clustering algorithm

IV. Methodology

We use the following methodology to subset GPGPU workloads. First, we collect a set of suitable micro-architectural events or metrics to characterize an application. We use principal component analysis (PCA) to remove redundancy in the collected metrics. The we group similar metrics together using a clustering technique. Each of these steps and the rationale behind the choices we made are described next.

A. Metrics

The next two steps, i.e., dimensionality reduction and clustering analysis have been thoroughly studied in the past and making the right choice of techniques for these steps is easier. Choosing the right metric to characterize an application, on the other hand is specific to the end goal of the study and the target architecture.

Ideally, one would want a benchmark suite such as SPEC ACCEL to stress the different components of an architecture. Therefore, we look at the various components available in our target architecture shown in Figure 1, and pick the most relevant metric for each component. The chosen metrics are summarized in Table V. We group these metrics into four major categories as described below:

TABLE V: Relevant metrics for understanding the impact of workload on the microarchitecture

Category	Hardware unit	Abbr	Metrics
Front End	Instruction Cache	IPC	Number of instructions executed per cycle (ipc)
	Instruction Dispatch Unit	ISS	Number of instructions issued per cycle (issued_ipc)
Execution Units	Core (Int)	INT	Number of integer instructions executed per cycle (inst_integer)
	Core (Float)	FP_SP	Number of single-precision instructions executed per cycle (inst_fp_32)
	DP Unit	FP_DP	Number of single-precision instructions executed per cycle (inst_fp_64)
	LD/ST Unit	LD_ST	Number of compute load/store instructions executed per cycle (inst_compute_ld_st)
	SFU	SFU	Number of single-precision floating-point special operations per cycle (flop_count_sp_special)
	Control	CTRL	Number of control instructions such as jump, branch, etc. per cycle (inst_control)
	Other instructions	MISC	Number of miscellaneous instructions executed per cycle (inst_misc)
On-chip Data Transfer	L1 Cache and Shared Memory	L1_SH	Utilization level of L1 cache and shared memory combined relative to the peak utilization (l1_shared_utilization)
	Texture Cache	TEX	Utilization level of texture cache relative to the peak utilization (tex_utilization)
	L2 Cache	L2	Utilization level of L2 cache relative to the peak utilization (l2_utilization)
Off-chip Data Transfer	Memory controller and DRAM	DRAM	Utilization level of DRAM relative to the peak utilization (dram_utilization)

Front End: This category includes all metrics associated with the scheduling of instructions. The relevant metrics are the number of instructions that were issued and the number of instructions that were executed. The warp scheduler, while an important component, does not have any key relevant metric that can be measured in our GPU.

Instruction Mix: This includes such metrics as integer instructions, floating point instructions (single precision and double precision), control instructions, special purpose instructions, and miscellaneous instructions such as shuffle and atomic operations. All metrics are measured on a per-cycle basis.

On-chip Data Transfer: This is related to the portion of the memory hierarchy present on the chip. The utilization of shared memory, L1 cache, L2 cache, and texture memory is measured here.

Off-chip Data Transfer: This is related to the portion of the memory hierarchy present off the chip. The utilization of DRAM is measured.

To collect the above metrics, we use NVIDIA's *nvprof* interface. The metrics are collected for each GPU kernel within an application. If an application spans multiple kernels, we pick the longest running one to represent that application.

While some of the metrics chosen in this step are dependent on each other, we leave the task of removing correlated metrics to the next step.

B. Principal Component Analysis

We use the principal component analysis (PCA) technique for dimensionality reduction. The advantages of using a PCA are two fold: (i) they remove redundancy in the collected metrics and (ii) they help in reducing the number of variables to be used in further steps as most of the useful information regarding an application can be obtained from a few principal components.

At a higher level, the goal of this technique is to rotate the m axes associated with a raw data set so that the projection on n axes shows a high variation. These n transformed axes can then be used to represent the original information with fewer variables at the cost of a minimal loss in information.

A key decision to make in this step is the amount of principal components to retain for the subsequent stage. To retain most of the information available in the raw variables, we choose to limit the loss of variance to utmost 10%. We therefore retain 6 principal components to achieve this target.

To avoid giving overdue importance to raw variables with high numerical value, we normalize all variables to have a mean of *zero* and a standard deviation of *one* before we begin this step.

C. Hierarchical Clustering

Two clustering schemes are commonly used - hierarchical clustering and K-means. The disadvantage of using k-means is that one must decide on the value of k apriori. Therefore, we use the hierarchical clustering scheme to group similar applications together.

In this technique, we identify the data points that are close to each other in each iteration. At the end of the iteration the closest pair is grouped together. To calculate the distance between two clusters, we use the single linkage distance technique, which is the distance between the closest points in the two clusters. This technique has been successfully employed in the past for workload characterization and we base our decision on past successful usage.

Once the clusters are formed, we present the resultant clusters in the form of a dendogram. Applications that are similar to each other are connected by shorter line segments while dissimilar applications are connected by longer line segments in this diagram.

D. Automated Application Subsetting

Using the techniques described above we automatically subset the benchmark applications as follows:

- First, the user selects the applications and problem size for each application and writes a execution script to run all these applications.
- The user also selects metrics for his study from a list of metrics provided by *nvprof* tool (`nvprof --query-events`).
- We run the applications with *nvprof* and collect the selected metrics.
- The kernel with the highest execution time within an application is selected to represent each application.
- A Python script normalizes the collected metrics and performs a principal component analysis on the data.
- Top 'n' PCs (n=6 here) are used to perform a hierarchical clustering.
- The clustered workloads are presented in the form of a dendogram using Python's data analytics packages.

V. Results and Discussion

In this section, we characterize and subset the SPEC ACCEL benchmark suite. We validate our subsetting methodology and apply the method to SHOC, Rodinia, and Parboil benchmark suites. We compare the diversity of each of these suites and identify "gaps" in them.

A. SPEC ACCEL

We present a high-level characterization of the SPEC ACCEL based on compute- and memory-centered metrics separately. Then we analyze redundancy in SPEC ACCEL and identify a smaller subset of applications that provide sufficient diversity while keeping the suite well balanced. We also formally verify our subsetting methodology.

Characterizing SPEC ACCEL: Fig. 2 shows the instruction mix for the applications belonging to this suite. We see a diverse mix of applications such as: (i) *LBM*, which has high percentage of single-precision floating-point operations (ii) *lavaMD*, which predominantly performs double-precision floating-point operations (iii) *histogram*, which is mostly based on integer operations (iv) *BFS*, which uses many control-flow operations and (v) *Needleman-Wunsch* with plenty of load/store operations. Applications such as *BFS* and *Gaussian elimination* also perform many miscellaneous operations that do not fit in the above categories.

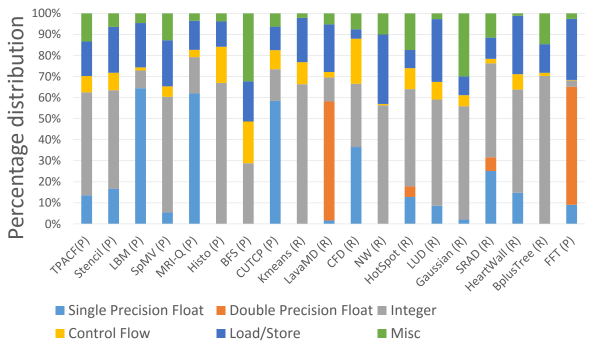


Fig. 2: Instruction mix for SPEC ACCEL benchmark suite

Applications such *TPACF* and *stencil* have very similar instruction mix, but they vary significantly in their memory behavior which is shown in Fig. 3. Our diversity analysis and subsetting procedure should consider them dissimilar. Applications such as *LUD* and *Gaussian elimination*, which have similar instruction mix as well as memory behavior, should be considered similar.

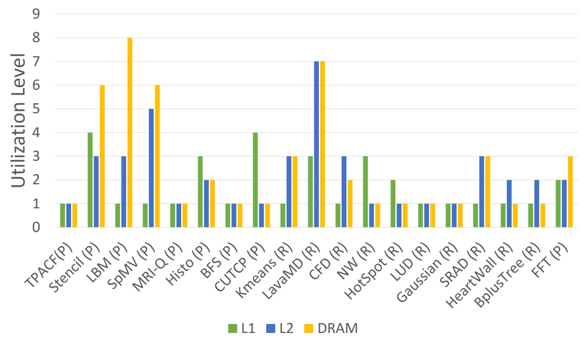


Fig. 3: Memory Utilization for SPEC ACCEL benchmark suite

Subsetting SPEC ACCEL: The diversity of these applications is presented in the form of a dendogram in Fig. 4. The x-axis of the dendogram is the linkage distance which is a measure of similarity and the y-axis is the applications. Applications that are similar to each other are connected by lines with a shorter distance. For example, *NW*, *K-Means*, *LUD*, *BFS*, and *Gaussian* are all linked by short lines and they form a dense cluster. These five applications may be replaced by a single application to make the suite uniformly balanced. *Stencil*, on the other hand, is unique and not connected by a short line with any other application. We have formed eight clusters and color coded them in Fig. 4. Choosing one application from each of these clusters will

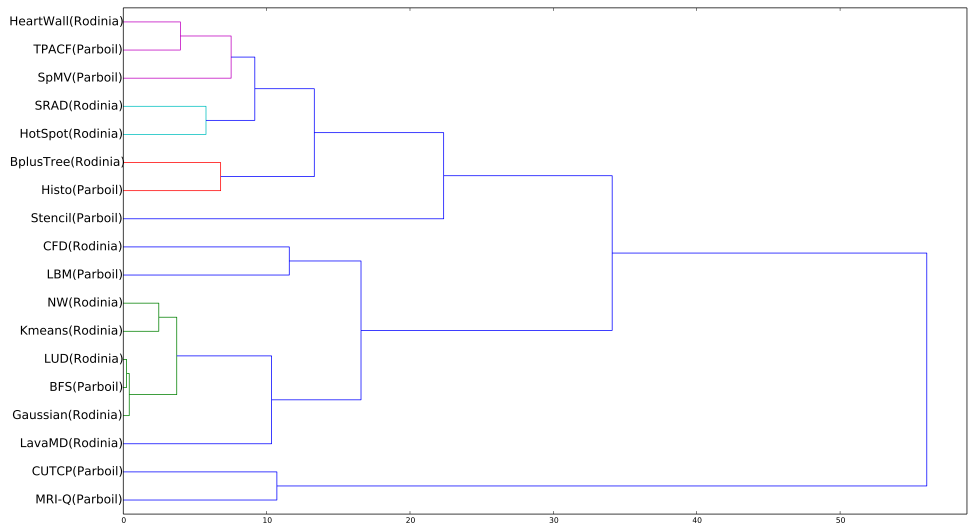


Fig. 4: Dendogram for SPEC ACCEL benchmark suite. The x-axis represents linkage distance.

help form a balanced suite. Representative subsets with four, six, and eight applications are shown in Table VI.

TABLE VI: Representative subset of SPEC ACCEL

Four apps	TPACF, Stencil, LUD, CUTCP
Six apps	TPACF, Histo, Stencil, LBM, LUD, CUTCP
Eight apps	TPACF, Histo, Stencil, CFD, LBM, LUD, CUTCP, MRI-Q

Validating the subsetting approach: We show that the chosen metrics and the methodology is appropriate for the given task using the following validation technique. We obtain the SPEC rating, which is the speed up obtained relative to a reference machine, for eighteen machines populated with different accelerators. Then, we calculate a new speed up value from only the subset of eight applications obtained via the subsetting procedure. If our selected subset is truly representative of the entire suite, then the new speed up score will be nearly the same as the score calculated from the entire benchmark suite. The original and the new speed up values for the eighteen machines whose results have been submitted to SPEC are presented in Table VII.

TABLE VII: Speedup results with and without subsetting

Accelerator	Original Speedup	New Speedup	Error (%)
NVIDIA Tesla C2070	0.98	0.98	0.45
NVIDIA Tesla K20 #1	1.52	1.50	0.69
NVIDIA Tesla K20 #2	1.44	1.43	1.04
Intel Xeon E5620	0.25	0.25	1.97
NVIDIA GTX 680	1.15	1.11	3.37
NVIDIA Tesla K40m	1.92	1.99	3.93
NVIDIA GTX TITAN #2	2.17	2.28	4.89
NVIDIA Tesla K40c #3	1.87	1.96	5.05
NVIDIA Tesla K40c #1	1.98	2.09	5.53
NVIDIA Tesla K20c	1.68	1.77	5.65
NVIDIA Tesla K20Xm	1.72	1.84	6.69
NVIDIA Tesla K20 #3	1.29	1.20	6.69
NVIDIA GTX TITAN #1	2.41	2.58	7.16
NVIDIA Tesla K40c #2	1.90	2.05	7.76
Intel Xeon E5-2697 v3	2.09	1.90	9.05
AMD Radeon HD 7970	1.71	1.95	13.67
AMD Radeon R9 290	1.41	1.61	13.87
Intel Xeon Phi 5110P	0.44	0.32	27.36
Average error			6.94

We achieve an overall error rate of 6.94% when we use subsetting. This indicates that **our chosen subset of benchmarks can reasonably predict the performance of the entire suite**. Forming such smaller subsets will help reduce evaluation time for newer architectures. We did expect a small difference in the obtained speedup. This is because our chosen subset is bias free. meaning it does not favor a specific kind of application. But the original suite was biased to certain types of applications as seen from the presence of dense clusters in our dendogram.

The prediction accuracy is relatively low for nearly all platforms whose vendor is different from our experimental platform (2 out of 2 AMD and 2 out of 3 Intel accelerators). In fact, the bottom four platforms, in terms of prediction error, are all from a different vendor. This is understandable because certain metrics such as *special floating point operations* may not even be meaningful for these architectures. This stresses the importance of using architecture-independent metrics for characterization and subsetting. We plan to evaluate architecture-independent metrics once binary profiling tools such as PIN becomes available for GPUs in the future. However, we also note that architecture-dependent metrics work for the various NVIDIA GPUs spanning even two microarchitecture generations.

B. SHOC, Rodinia, and Parboil Benchmark Suites

In this section, we present the subsetting results for SHOC, Rodinia, and Parboil benchmark suites. We omit the high-level characterization of instruction mix and memory utilization for these benchmarks due to space constraints.

Subsetting SHOC benchmark suite: Figure 5 shows the dendogram representation of the clusters formed from the top six principal components for the SHOC benchmark suite. SHOC has many applications that are very similar to each other with linkage distance lower than five for many pairs of applications (Lower value for linkage distance is indicative of similarity between applications) . Representative subsets of size four, six, and eight applications are shown in Table VIII.

TABLE VIII: Representative subset of SHOC

Four apps	Sort, BFS, GEMM, SpMV (vector)
Six apps	Sort, BFS, SpMV (scalar), Scan, GEMM, SpMV (vector)
Eight apps	Sort, BFS, SpMV (scalar), Triad, Scan, GEMM, SpMV (vector), Stencil2D

The following are some of the observations we make from the dendogram.

Observation 1: GEMM, FFT, and MD, while being fundamentally different algorithms, all exhibit similar execution behavior. One among the three is sufficient in the *level 1* primitives (i.e., basic parallel algorithms) of the SHOC benchmark suite if the end goal is architecture evaluation.

Observation 2: CSR scalar representation of the SpMV benchmark is similar to the Triad application and the CSR vector representation of SpMV is similar to Reduction and Stencil applications. Thus by including SpMV in a study, three other *level 1* applications can be removed.

Observation 3: The two "real-world" *level 2* applications S3D and QTC already belong to different clusters (i.e, they show widely differing behaviors). Other real-life applications exhibiting characteristics of Sort, BFS, Scan, and Stencil2D is currently lacking.

Subsetting Rodinia benchmark suite: Fig. 6 shows the dendogram representation of the clusters formed for the Rodinia benchmark suite. Based on this dendogram, we arrive at representative subsets of size four, six, and eight applications which is shown in Table IX.

TABLE IX: Representative subset of Rodinia

Four apps	Hotspot, CFD, LUD, Stream Cluster
Six apps	Hotspot, Backprop, CFD, LUD, LavaMD, Stream Cluster
Eight apps	Hotspot, Backprop, Leukocyte, CFD, LUD, LavaMD, Stream Cluster, B+ Trees

We make the followin observations regarding the Rodinia benchmark suite.

Observation 4: One of SRAD and HotSpot, one of heartwall and Backprop, one of leukocyte and CFD, One among NW, BFS, KMeans, Particle Filter, Gaussian, LUD, and NN, and either StreamCluster or Bplus tree provides sufficient diversity for the benchmark suite.

Observation 5: Applications belonging to the same "dwarf" category may differ widely in their behavior, where dwarf is a fundamental computation and communication idiom. Similarly, applications belonging to different dwarf categories (ex. BFS, NW) exert the microarchitecture in a similar fashion.

Subsetting Parboil benchmark suite: Figure 7 shows the dendogram representation of the clusters formed from the top six principal components for the Parboil benchmark suite. Based on this dendogram, we arrive at representative subsets of size 4, 6, and 8 applications which is shown in Table X.

TABLE X: Representative subset of Parboil

Four apps	TPACF, Stencil, SGEMM, CUTCP
Six apps	BFS, TPACF, Stencil, SGEMM, MRI-Gridding, CUTCP
Eight apps	LBM, BFS, TPACF, Stencil, SAD, SGEMM, MRI-Gridding, CUTCP

The following obervations are made regarding the parboil benchmark suite.

Observation 6: The linkage distances of all the clusters are ten or more. Compared with the other benchmark suites, this is significantly higher indicating that a diverse set of applications are covered by this suite.

Expanding the existing benchmark suites: We put together all the benchmark suites we have examined so far together and perform a diversity analysis of the ensemble. This will help in identifying gaps in existing benchmark suites. Fig. 8 shows the dendogram of the ensemble with ten clusters formed and color coded.

Table XI shows the coverage of each of the three benchmark suites separately. Parboil has applications represented in nine out of the ten clusters formed whereas SHOC had

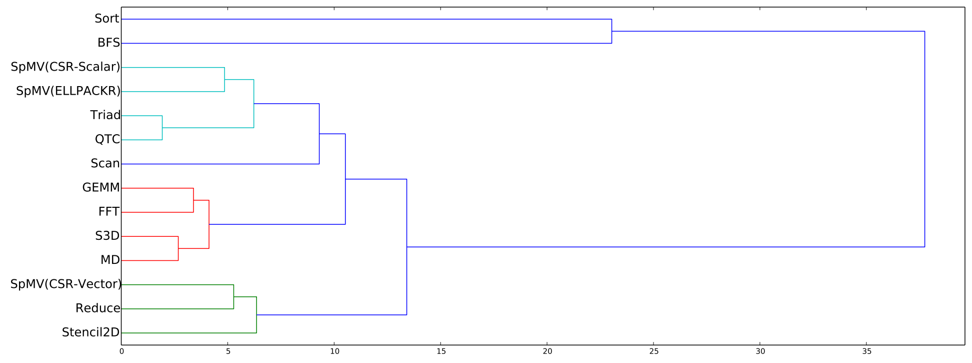


Fig. 5: Dendogram for SHOC benchmark suite. The x-axis represents the linkage distance.

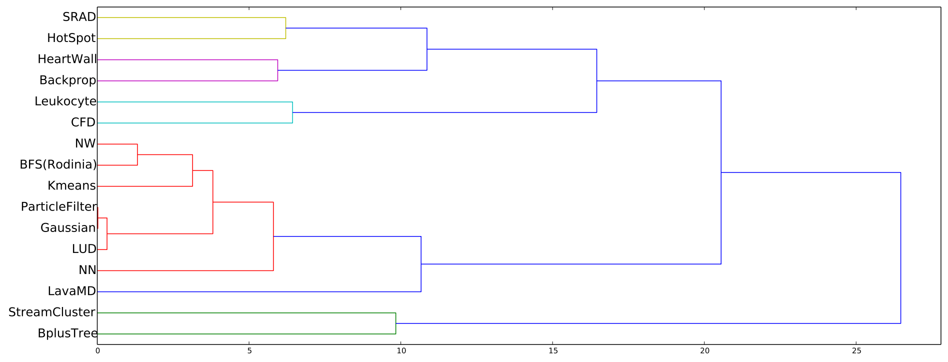


Fig. 6: Dendogram for Rodinia benchmark suite. The x-axis represents the linkage distance.

representation in only four cluster. We show what happens to the coverage of each benchmark suite when we change the number of clusters formed. In general, Parboil shows the most variety.

TABLE XI: Coverage of existing benchmark suites

Six clusters	SHOC - Three out of six Rodinia - Four out of six Parboil - Six out of six
Eight clusters	SHOC - Three out of eight Rodinia - Four out of eight Parboil - Eight out of eight
Ten clusters	SHOC - Four out of ten Rodinia - Five out of ten Parboil - Nine out of ten

Observation 7: Parboil shows the most coverage among Parboil, Rodinia, and SHOC.

Based on the above study, we make recommendations for filling in the "gaps" in the the three benchmark suites, especially SHOC and Rodinia. These recommendations are summarized in the Table. XII. The boldfaced applications are the ones that need to brought in from other sources.

TABLE XII: Expanding existing benchmark suites

SHOC	Sort, BFS, GEMM, SpMV (vector), **Stencil, LBM, TPACF, SGEMM, MRI-Gridding, CUTCP (all from parboil)**
Rodinia	Hotspot, Backprop, CFD, LUD, Stream Cluster, **SAD, Stencil, SGEMM, MRI-Gridding, CUTCP (all from parboil)**
Parboil	LBM, BFS, TPACF, Stencil, SAD, SGEMM, MRI-Gridding, CUTCP, Histo, **backprop (Rodinia)**

VI. Conclusion

We identified metrics for subsetting GPGPU workloads. Using PCA and clustering, we subset the SPEC ACCEL benchmark suite. Making use of results reported to SPEC, we validated our metric and methodology. After this validation, we performed subsetting on other popular GPU benchmark suites such as SHOC, Rodinia, and Parboil using the same set of metrics and methodology. We compared the different

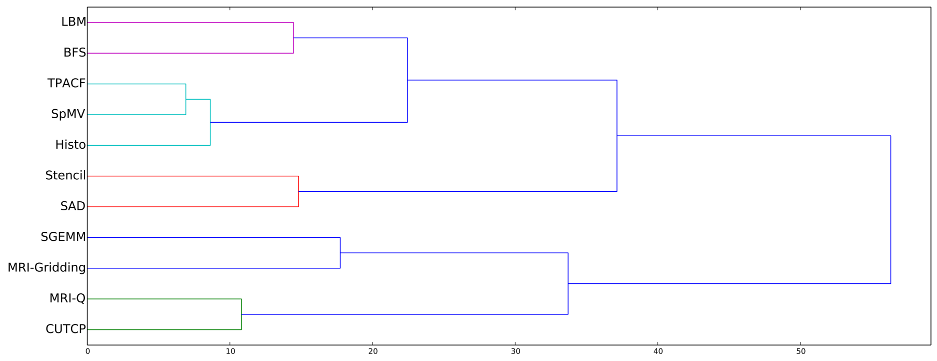


Fig. 7: Dendogram for Parboil benchmark suite. The x-axis represents the linkage distance.

suites, identified gaps in them, and offered solution to fill in the missing application types. We hope that our results will help in choosing the appropriate set of applications for evaluating newly proposed techniques for GPUs in the high-performance computing domain.

Acknowledgment

This work was supported in part by NSF I/UCRC IIP-0804155 and IIP-1266245 via the NSF Center for High-Performance Reconfigurable Computing.

References

[1] "TOP500 Supercomputer Site." http://www.top500.org.

[2] J. A. Stratton, C. Rodrigues, I.-J. Sung, N. Obeid, L.-W. Chang, N. Anssari, G. D. Liu, and W.-M. Hwu, "Parboil: A Revised Benchmark Suite for Scientific and Commercial Throughput Computing," *Center for Reliable and High-Performance Computing*, 2012.

[3] S. Che, M. Boyer, J. Meng, D. Tarjan, J. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A Benchmark Suite for Heterogeneous Computing," in *Proceedings of the 2009 IEEE International Symposium on Workload Characterization (IISWC)*, pp. 44–54, Oct 2009.

[4] S. Che, J. W. Sheaffer, M. Boyer, L. G. Szafaryn, L. Wang, and K. Skadron, "A Characterization of the Rodinia Benchmark Suite with Comparison to Contemporary CMP Workloads," in *Proceedings of the 2010 IEEE International Symposium on Workload Characterization (IISWC)*, pp. 1–11, IEEE Computer Society, 2010.

[5] A. Danalis, G. Marin, C. McCurdy, J. S. Meredith, P. C. Roth, K. Spafford, V. Tipparaju, and J. S. Vetter, "The Scalable Heterogeneous Computing (SHOC) Benchmark Suite," in *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units (GPGPU)*, pp. 63–74, ACM, 2010.

[6] G. Juckeland, W. C. Brantley, S. Chandrasekaran, B. M. Chapman, S. Che, M. E. Colgrove, H. Feng, A. Grund, R. Henschel, W. mei W. Hwu, H. Li, M. S. Mller, W. E. Nagel, M. Perminov, P. Shelepugin, K. Skadron, J. A. Stratton, A. Titov, K. Wang, G. M. van Waveren, B. Whitney, S. Wienke, R. Xu, and K. Kumaran, "SPEC ACCEL: A Standard Application Suite for Measuring Hardware Accelerator Performance," in *PMBS@SC*, vol. 8966 of *Lecture Notes in Computer Science*, pp. 46–67, Springer, 2014.

[7] L. Eeckhout, H. Vandierendonck, and K. De Bosschere, "Workload Design: Selecting Representative Program-Input Pairs," in *Proceedings of the 2002 International Conference on Parallel Architectures and Compilation Techniques*, pp. 83–94, 2002.

[8] A. Phansalkar, A. Joshi, L. Eeckhout, and L. John, "Measuring Program Similarity: Experiments with SPEC CPU Benchmark Suites," in *Proceedings of the 2005 International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 10–20, March 2005.

[9] A. Joshi, A. Phansalkar, L. Eeckhout, and L. K. John, "Measuring Benchmark Similarity Using Inherent Program Characteristics," *IEEE Transactions on Computers*, vol. 55, p. 782, 2006.

[10] J. Yi, R. Sendag, L. Eeckhout, A. Joshi, D. Lilja, and L. John, "Evaluating benchmark subsetting approaches," in *Workload Characterization, 2006 IEEE International Symposium on*, pp. 93–104, Oct 2006.

[11] K. Hoste and L. Eeckhout, "Comparing benchmarks using key microarchitecture-independent characteristics," in *Workload Characterization, 2006 IEEE International Symposium on*, pp. 83–92, Oct 2006.

[12] K. Hoste and L. Eeckhout, "Microarchitecture-Independent Workload Characterization," *Micro, IEEE*, vol. 27, pp. 63–72, May 2007.

[13] A. Phansalkar, A. Joshi, and L. K. John, "Analysis of Redundancy and Application Balance in the SPEC CPU2006 Benchmark Suite," in *Proceedings of the 34th Annual International Symposium on Computer Architecture (ISCA)*, pp. 412–423, ACM, 2007.

[14] K. Hoste and L. Eeckhout, "Characterizing the Unique and Diverse Behaviors in Existing and Emerging General-Purpose and Domain-Specific Benchmark Suites," in *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and software (ISPASS)*, pp. 157–168, April 2008.

[15] C. Isen, L. John, J. P. Choi, and H. J. Song, "On the representativeness of embedded java benchmarks," in *Proceedings of the 2008 IEEE International Symposium on Workload Characterization (IISWC)*, pp. 153–162, Sept 2008.

[16] Z. Jia, J. Zhan, L. Wang, R. Han, S. Mckee, Q. Yang, C. Luo, and J. Li, "Characterizing and Subsetting Big Data Workloads," in *Proceedings of the 2014 IEEE International Symposium on Workload Characterization (IISWC)*, pp. 191–201, Oct 2014.

[17] R. Panda and L. John, "Data analytics workloads: Characterization and similarity analysis," in *Proceedings of the 2014 IEEE International Performance Computing and Communications Conference (IPCCC)*, pp. 1–9, Dec 2014.

[18] A. Kerr, G. Diamos, and S. Yalamanchili, "A Characterization and Analysis of PTX Kernels," in *Proceedings of the IEEE International Symposium on Workload Characterization (IISWC)*, pp. 3–12, Oct 2009.

[19] N. Goswami, R. Shankar, M. Joshi, and T. Li, "Exploring GPGPU Workloads: Characterization Methodology, Analysis and Microarchitecture Evaluation Implications," in *Proceedings of the 2010 IEEE International Symposium on Workload Characterization (IISWC)*, pp. 1–10, Dec 2010.

[20] Nvidia Corporation, *Tesla K20 GPU Active Accelerator: Board Specification*, Jan. 2013.

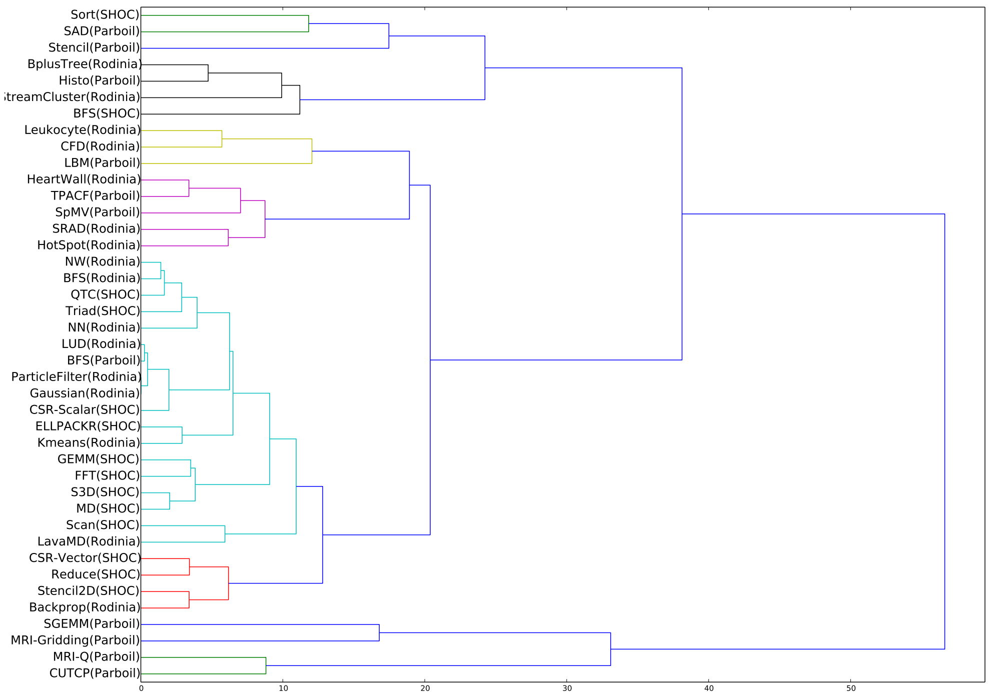


Fig. 8: Diversity analysis for all benchmark suites put together to identify opportunities for expansion.

[21] L. Luo, M. Wong, and W.-m. Hwu, "An effective gpu implementation of breadth-first search," in *Proceedings of the 47th Design Automation Conference*, DAC '10, (New York, NY, USA), pp. 52–55, ACM, 2010.

[22] V. Volkov and B. Kazian, "Fitting fft onto the g80 architecture, 2008," *E63*.

[23] S. Sengupta, M. Harris, Y. Zhang, and J. D. Owens, "Scan primitives for gpu computing," in *Proceedings of the 22Nd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware*, GH '07, (Aire-la-Ville, Switzerland, Switzerland), pp. 97–106, Eurographics Association, 2007.

[24] N. Satish, M. Harris, and M. Garland, "Designing efficient sorting algorithms for manycore gpus," in *Parallel Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pp. 1–10, May 2009.

[25] N. Bell and M. Garland, "Efficient sparse matrix-vector multiplication on cuda," tech. rep., Nvidia Technical Report NVR-2008-004, Nvidia Corporation, 2008.

[26] P. R. Luszczek, D. H. Bailey, J. J. Dongarra, J. Kepner, R. F. Lucas, R. Rabenseifner, and D. Takahashi, "The hpc challenge (hpcc) benchmark suite," in *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, SC '06, (New York, NY, USA), ACM, 2006.

[27] E. R. Hawkes, R. Sankaran, J. C. Sutherland, and J. H. Chen, "Direct numerical simulation of turbulent combustion: fundamental insights towards predictive models," *Journal of Physics: Conference Series*, vol. 16, no. 1, p. 65, 2005.

[28] J. Fix, A. Wilkes, and K. Skadron, "Accelerating braided b+ tree searches on a gpu with cuda," in *2nd Workshop on Applications for Multi and Many Core Processors: Analysis, Implementation, and Performance (A4MMC), in conjunction with ISCA*, 2011.

[29] A. Corrigan, F. F. Camelli, R. Lhner, and J. Wallin, "Running unstructured grid-based cfd solvers on modern graphics hardware," *International Journal for Numerical Methods in Fluids*, vol. 66, no. 2, pp. 221–229, 2011.

[30] L. G. Szafaryn, K. Skadron, and J. J. Saucerman, "Experiences accelerating matlab systems biology applications," in *Proceedings of the Workshop on Biomedicine in Computing: Systems, Architectures, and Circuits*, pp. 1–4, Citeseer, 2009.

[31] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. Stan, "Hotspot: a compact thermal modeling methodology for early-stage vlsi design," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 14, pp. 501–513, May 2006.

[32] L. G. Szafaryn, T. Gamblin, B. R. De Supinski, and K. Skadron, "Experiences with achieving portability across heterogeneous architectures," *Proceedings of WOLFHPC, in Conjunction with ICS, Tucson*, 2011.

[33] M. Boyer, D. Tarjan, S. T. Acton, and K. Skadron, "Accelerating leukocyte tracking using cuda: A case study in leveraging manycore coprocessors," in *Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing*, IPDPS '09, (Washington, DC, USA), pp. 1–12, IEEE Computer Society, 2009.

[34] L. G. Szafaryn, K. Skadron, and J. J. Saucerman, "Experiences accelerating matlab systems biology applications," in *Proceedings of the Workshop on Biomedicine in Computing: Systems, Architectures, and Circuits*, pp. 1–4, Citeseer, 2009.

[35] M. A. Goodrum, M. J. Trotter, A. Aksel, S. T. Acton, and K. Skadron, "Parallelization of particle filter algorithms," in *Proceedings of the 2010 International Conference on Computer Architecture*, ISCA'10, (Berlin, Heidelberg), pp. 139–149, Springer-Verlag, 2012.