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 other 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 November 2015 list, a total of 104 systems use accelerators, up from 75 systems an year 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 illustration 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 targeted 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 with 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 framework, 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 conduct our experiments on a NVIDIA Kepler GK110 GPU [20], the block diagram of which is presented in Fig. 1. This GPU consists of 15 streaming multiprocessors

TABLE I: Summary of Related Work

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

(SMs). Each SM consists of an instruction cache, four warp schedulers which are responsible for scheduling warps (a collection of threads) to the SMs, and eight instruction dispatch units which determines the instruction scheduled in a given clock cycle. There are 192 CUDA cores within each SM where each core 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. The SMs are also provided with double-precision (DP) units, special function units (SFU), and load/store (LD/ST) units for executing the corresponding instructions. Apart from these instructions, the GPU is also capable of executing branch instructions, atomic instructions, and shuffle instructions. Each SM also has a 65,536 x 32-bit register file, 64 KB configurable shared memory and L1 cache, a 48 KB read-only data cache, and several texture units. Common to all the SMs, there is 1536 KB of 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 three benchmark suites, implementations in several languages are available, but we choose the CUDA implementation for reasons noted above.

A short description of the applications in SHOC, Parboil, and Rodinia is presented in Table II, Table III, and Table IV. SPEC ACCEL benchmarks are a subset of these applications,

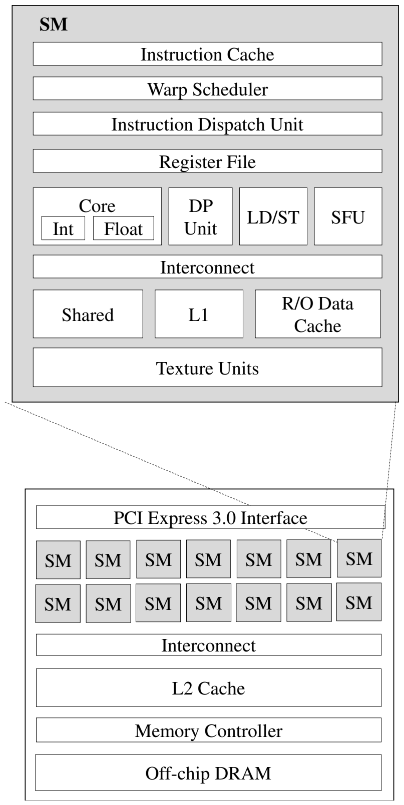

Fig. 1: Block diagram of NVIDIA Kepler

so we do not describe them separately.

TABLE II: Summary of SHOC benchmarks

Benchmark	Description	Size [1]
BFS	Breadth-first search	S4
FFT	512-pt 2-D fast Fourier transforms	S4
MD	Molecular dynamics application that calculates Lennard-Jones potential	S3
Reduction	Sum of elements in an array	S4
Scan	Performs prefix sum calculations	S4
GEMM	General matrix-matrix multiplication	S4
Sort	Performs a fast radix sort on several key-value pairs	S4
SpMV	Sparse matrix-vector multiplication (CSR scalar, CSR vector, and ELLPACKR)	S4
Stencil2D	2-D 9-pt stencil computation	S3
Triad	Performs streaming dot-product multiplication	S4
QTC	Quality threshold clustering	S4
S3D	Computes the rate of a chemical reaction	S4

IV. Methodology

In this section, we describe the methodology used to subset GPGPU workloads. The general approach can be described as follows: (i) collect a set of suitable micro-architectural events/metrics during an application's execution (ii) apply principal component analysis (PCA) to remove redundancy in the collected metrics and (iii) group applications showing similar metrics together using a clustering technique. Each of these steps and the rationale behind the choices we made are described next.

[1]S3 refers to medium-sized problems and S4 refers to large-sized problems in SHOC terminology.

TABLE III: Summary of Parboil benchmarks

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
MatMul	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
Stencil3D	3-D stencil operation	Small
TPACF	Two-point angular correlation function	Default

TABLE IV: Summary of Rodinia benchmarks

Benchmark	Description
Backprop	Train weights in neural network using backward propagation technique
BFS	Breadth-first search
BPlusTree	Traverses a B+ tree
CFD	Computational fluid dynamics for unstructured grids
Gaussian	Gaussian elimination method for solving equations
HeartWall	Track changing shape of a mouse's heart
HotSpot	Solves differential equation to generate processor's heatmap
K-Means	Clustering of data points
LavaMD	N-body algorithm
Leukocyte	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	Structured grid application to simulate mycocyte cells
kNN	Nearest neighbor algorithm
NW	Needleman-Wunsch algorithm for DNA sequence alignment
ParticleFilter	Estimates the location of an object from noisy data
PathFinder	Find the shortest path between two points in a 2-D grid
SRAD	Speckle reducing anisotropic diffusion for removing speckles in image
StreamCluster	Online clustering algorithm

A. Metrics

The subsequent two steps, i.e., dimensionality reduction and clustering analysis have been thoroughly studied in the past [7]–[19] and making the right choice of techniques for these steps is easier. Choosing the right metric, on the other hand, is highly dependent on the end goal of the study and the target architecture.

The expectation for a benchmark suite such as SPEC ACCEL is to stress the various components of an architecture. Therefore, we look at all the major components in our target architecture (shown in Figure 1) and pick the most relevant metric for each component to perform this study. Qualitatively, we reason that any component could be a potential bottleneck for performance, and therefore picking one metric per component is justified. We also quantitatively justify our choice of metrics by validating the metrics and the approach in Section V.

The chosen metrics are summarized in Table V. We group these metrics into four major categories as described below:

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

Instruction Mix: This includes metrics such 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 within the chip. Utilization of shared memory, L1 cache, L2 cache, and texture memory is measured.

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

While some of the metrics chosen in this step are correlated to 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 dataset and (ii) they help in reducing the number of variables used in subsequent steps which can be useful for visually presenting the relevant information.

At a higher level, the goal of this technique is to rotate the m axes associated with raw dataset in order to increase the variance of the data projected on to n fewer axes. 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 is the number of principal components to retain for the subsequent stages. We choose to limit the loss of variance to utmost 10%, which is consistent with past studies [19], in order to retain most of the information available in the raw variables. For our collected dataset, using six principal components is sufficient to meet this target.

C. Hierarchical Clustering

Two clustering schemes, namely, hierarchical and k-means clustering, could potentially be used to group similar applications. The advantage of hierarchical clustering is that the decision on the number of clusters can be made *after* the clustering process which makes it easier to perform a number of what-if analyses. Therefore, we use hierarchical clustering in our framework.

In this technique all the data points are individual clusters initially. The clusters with the shortest single linkage distance, which is the distance between the closest points in the two clusters, are grouped together iteratively. This process continues until all the clusters are grouped into one cluster.

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

D. Automated Application Subsetting Framework

We implement a framework (Fig. 2) in Python that uses the techniques described above to automatically subset benchmark applications. The framework works as follows:

- The end-user selects the applications and problem size for each application and writes a execution script to run all these applications.
- The user also provides a list of metrics to perform this study from a list of metrics provided by the *nvprof* tool (`nvprof --query-metrics`).
- **Profiling:** The framework runs the applications with *nvprof* and collects the selected metrics for all the GPU kernels in the chosen applications. Some of the metrics used in this study are non-aggregatable as they are pre-normalized by *nvprof*. Therefore, for each application, the longest running kernel is picked as the representative kernel and the corresponding metrics are gathered together.
- **Data preprocessing:** A Python script normalizes the collected metrics as the next step (i.e., PCA) is susceptible to dissimilar ranges of values.

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)

- **Dimensionality reduction:** Principal component analysis is performed on the normalized data using the `sklearn` package.
- **Clustering:** The top six principal components are used to perform a hierarchical clustering using the `scipy` package.
- **Graphing:** The clustered workloads are graphed and presented in the form of a dendrogram using the `matplotlib` package. This graph is the used by the end-user for manual randomized subsetting.

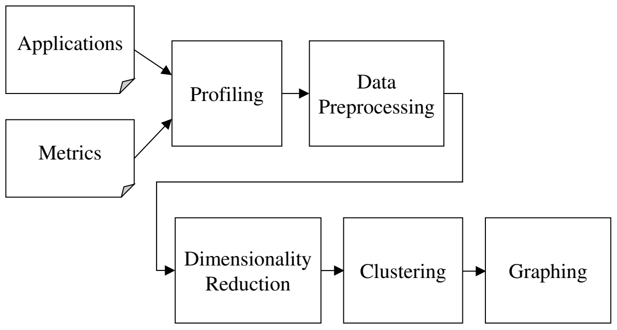

Fig. 2: Automated application subsetting framework

V. Results and Discussion

This section presents the results of our characterization and subsetting experiments with SPEC ACCEL, which is followed by a validation of the metrics and methodology. Similar sets of results are presented for the academia-oriented SHOC, Rodinia, and Parboil. A diversity analysis is performed to identify "gaps" in these benchmark suites.

A. *Results for SPEC ACCEL*

In this section, we present a high-level characterization of SPEC ACCEL based on compute- and memory-centered metrics. Next, we analyze redundancy in this suite and identify a smaller subset of applications that provide sufficient diversity while keeping the suite well balanced. Finally, we *empirically* validate our approach.

Characterization Results: Fig. 3 shows the instruction mix for the applications in SPEC ACCEL. In this suite, we observe a diverse mix of applications. We find applications that are predominantly composed of (i) single-precision floating point instructions (e.g., *LBM*) (ii) double-precision floating point instructions (e.g., *LavaMD*) and (iii) integer instructions (e.g., *BPlusTree*). We also notice applications with a relatively high percentage of control-flow instructions (e.g., *BFS*) and load/store instructions (e.g., *NW*). Applications such as *BFS* and *Gaussian* also perform many miscellaneous operations that do not fit in the above categories. Based on the above observations, we could infer that SPEC ACCEL is diverse in terms of instruction mix.

Fig. 4 presents the utilization of the various levels of the memory hierarchy for SPEC ACCEL applications. The values presented in this graph are normalized against the peak bandwidth offered by the corresponding level of the memory hierarchy. We observe that only *4 out of the 19* applications show a high DRAM utilization above 50%. This indicates that most of the applications in this benchmark suite are compute-bound. We also notice that nearly all applications utilize only a fraction of the bandwidth offered by the L1 and L2 caches. Based on the above observations, we could gather that SPEC ACCEL is less diverse in terms of memory utilization.

Our expectation is that applications such as *TPACF* and

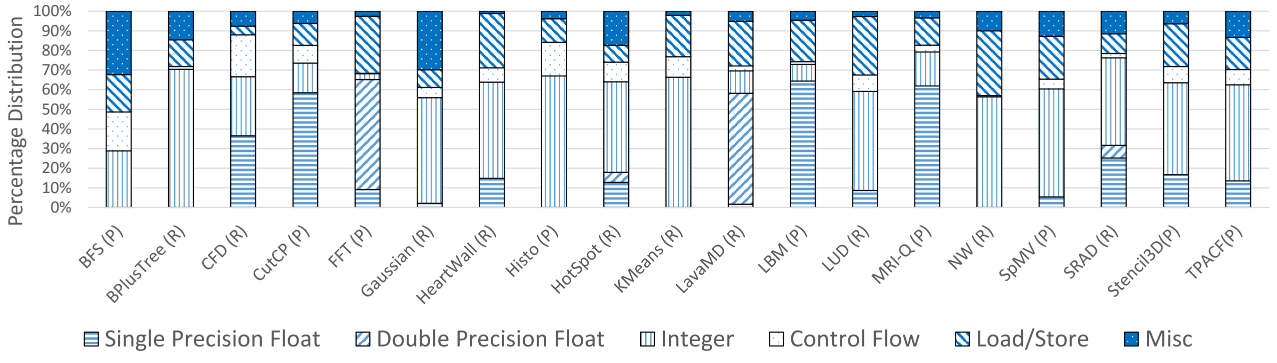

Fig. 3: Instruction mix for SPEC ACCEL benchmark suite. Applications derived from Rodinia are denoted by (R) and those derived from Parboil are denoted by (P).

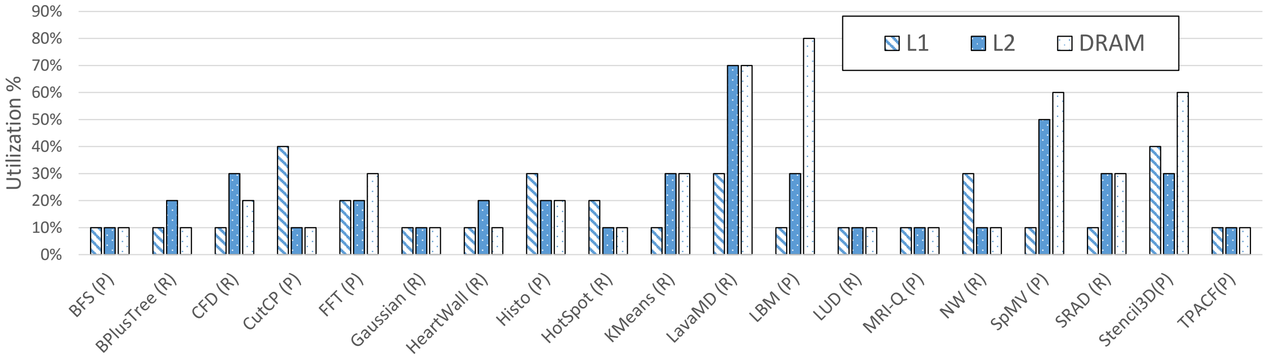

Fig. 4: Memory Utilization for SPEC ACCEL benchmark suite. Applications derived from Rodinia are denoted by (R) and those derived from Parboil are denoted by (P).

Stencil3D which have very similar instruction mix, but different memory behavior, will be categorized as dissimilar by our framework.

Subsetting Results: The diversity of the applications in SPEC ACCEL is presented in the form of a dendrogram in Fig. 5. The x-axis of the dendrogram represents linkage distance which is a measure of similarity. Applications that are similar to each other are connected by lines with a shorter distance. For example, *NW*, *KMeans*, *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. *Stencil3D*, on the other hand, is unique and not connected by a short line with any other application. As expected, this application is slotted in a cluster different from *TPACF* due to the differences seen in memory utilization in Fig. 4.

In Fig. 5, we have formed eight clusters which are color coded. Choosing one application *randomly* from each cluster will help form a balanced suite. Representative subsets with four, six, and eight applications are shown in Table VI.

TABLE VI: Representative subset for SPEC ACCEL

Four apps	TPACF, Stencil3D, LUD, CutCP
Six apps	TPACF, Histo, Stencil3D, LBM, LUD, CutCP
Eight apps	TPACF, Histo, Stencil3D, CFD, LBM, LUD, CutCP, MRI-Q

Validation: We show that the chosen metrics and the methodology is appropriate for the given task using the following validation technique. We gather the SPEC rating, which is the speedup obtained relative to a reference machine, for eighteen machines populated with different accelerators. Then, we calculate a new speedup value from the eight *subsetted* applications obtained through our methodology. If our selected subset is truly representative of the entire suite, then the new speedup score will be nearly the same as the score calculated from the entire benchmark suite.

The original and new speedup values for the eighteen machines whose results have been submitted to SPEC are presented in Table VII. The overall error rate for our subsetting methodology is 6.94% and the worst-case error rate

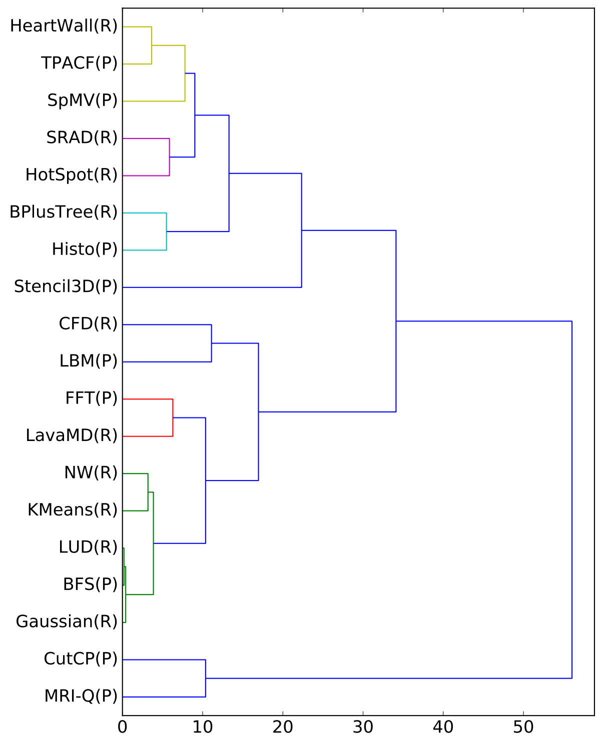

Fig. 5: Dendrogram for SPEC ACCEL benchmark suite. The x-axis represents linkage distance. Applications derived from Rodinia are denoted by (R) and those derived from Parboil are denoted by (P).

is 27.36%. By comparison, inadvertently selecting all the applications from a dense cluster in the dendrogram (e.g., *NW, KMeans, LUD, BFS, and Gaussian*) would result in a worst-case error rate of 145%. This shows that our chosen subset of benchmarks can reasonably predict the performance of the entire suite in a variety of platforms. Forming such smaller subsets will help reduce evaluation time for newer architectures.

Next, we analyze the data in Table VII to identify cases where our methodology shows a relatively poor accuracy. The error rate is high 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 machines are all from a different vendor. However, the metrics and the methodology works well for the various NVIDIA GPUs spanning multiple microarchitectural generations.

The above observation is understandable because certain metrics such as *special floating point operations* may not be meaningful for architectures from other vendors. This stresses the importance of using architecture-independent metrics for characterization and subsetting. With the recent introduction of binary profiling tools for GPUs such as SASSI [21], it may be possible to expand this study with architecture-independent metrics in the near future.

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

B. Results for SHOC, Rodinia, and Parboil

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 6 shows the dendrogram for 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 (a lower value for linkage distance is indicative of similarity between applications). Representative subsets of four, six, and eight applications are shown in Table VIII.

TABLE VIII: Representative subset for 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

We make the following observations from the dendrogram.

Observation 1: *GEMM*, *FFT*, and *MD*, while being fundamentally different algorithms, all exhibit similar execution behavior. If the end goal is to compare relative speedup on different architectures, it is sufficient to pick one among these three from the *level 1* primitives (i.e., basic parallel algorithms) of SHOC.

Observation 2: CSR scalar representation of *SpMV* is similar to *Triad* and CSR vector is similar to *Reduction* and

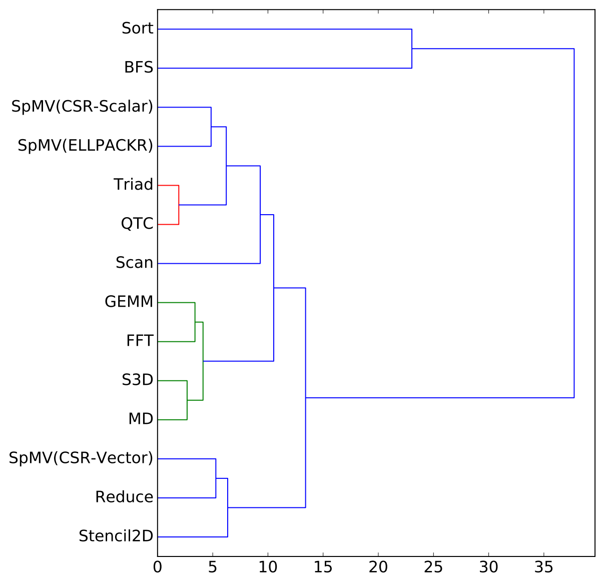

Fig. 6: Dendrogram for SHOC benchmark suite. The x-axis represents the linkage distance.

Stencil2D. Thus, by including *SpMV* in a study, three other *level 1* applications may be avoided.

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

Subsetting Rodinia benchmark suite: Fig. 7 shows the dendrogram representation of the clusters formed for Rodinia. Based on this dendrogram, we arrive at representative subsets of four, six, and eight applications which is shown in Table IX.

TABLE IX: Representative subset for Rodinia

Four apps	Hotspot, CFD, LUD, StreamCluster
Six apps	Hotspot, Backprop, CFD, LUD, LavaMD, StreamCluster
Eight apps	Hotspot, Backprop, Leukocyte, CFD, LUD, LavaMD, StreamCluster, BPlusTrees

We make the following observations regarding the Rodinia benchmark suite.

Observation 4: One of *SRAD* and *HotSpot*, one of *Heart-Wall* and *Backprop*, one of *Leukocyte* and *CFD*, one among *NW, BFS, KMeans, ParticleFilter, Gaussian, LUD, and kNN*, and either *StreamCluster* or *BPlusTree* is sufficient to capture the majority of the diversity of Rodinia.

Observation 5: Applications belonging to the same “dwarf” category may differ widely in their behavior, where a dwarf is a fundamental computation and communication

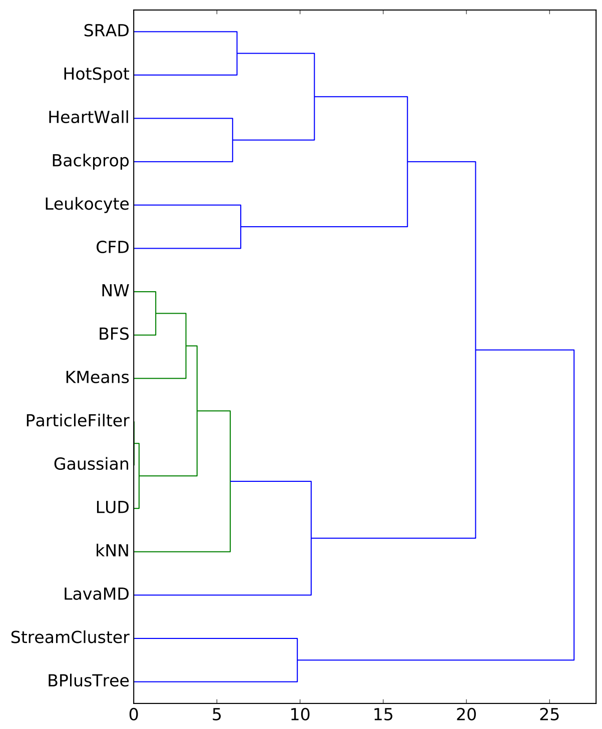

Fig. 7: Dendrogram for Rodinia benchmark suite. The x-axis represents the linkage distance.

idiom. Similarly, applications belonging to different dwarf categories (ex. *BFS, NW*) can exert the microarchitecture in similar fashion.

Subsetting Parboil benchmark suite: Figure 8 shows the dendrogram representation of the clusters formed for Parboil. Based on this dendrogram, we arrive at representative subsets of four, six, and eight applications which is shown in Table X.

TABLE X: Representative subset for Parboil

Four apps	TPACF, Stencil3D, MatMul, CutCP
Six apps	BFS, TPACF, Stencil3D, MatMul, MRI-G, CutCP
Eight apps	LBM, BFS, TPACF, Stencil3D, SAD, MatMul, MRI-G, CutCP

The following observations are made regarding the Parboil benchmark suite.

Observation 6: The linkage distances of all the clusters are ten or more. When compared to 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 all the benchmark suites we have examined so far together and perform a diversity analysis on the ensemble. This will help

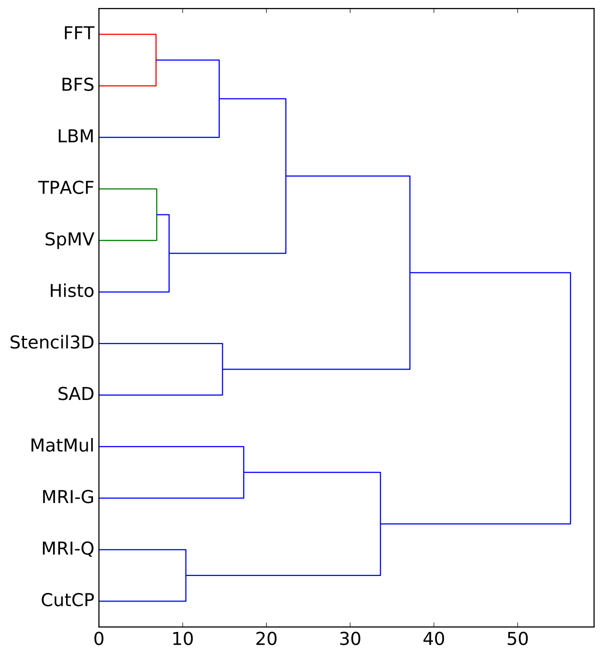

Fig. 8: Dendrogram for Parboil benchmark suite. The x-axis represents the linkage distance.

in identifying gaps in the existing benchmark suites. Fig. 9 shows the dendrogram of the ensemble with ten clusters formed and color coded. Parboil has applications represented in nine out of the ten clusters in Fig. 9 whereas SHOC has representation in only four clusters and Rodinia in five. This shows that SHOC and Rodinia has more gaps when compared to Parboil. Table XI also shows the impact of changing the number of clusters on the representativeness of the various benchmark suites. In general, Parboil shows the most diversity, followed by Rodinia, and then SHOC.

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" found in the three benchmark suites. These recommendations are summarized in the Table. XII. Boldfaced applications in the table are the recommended applications to be included from other sources in order to improve the corresponding suite's diversity.

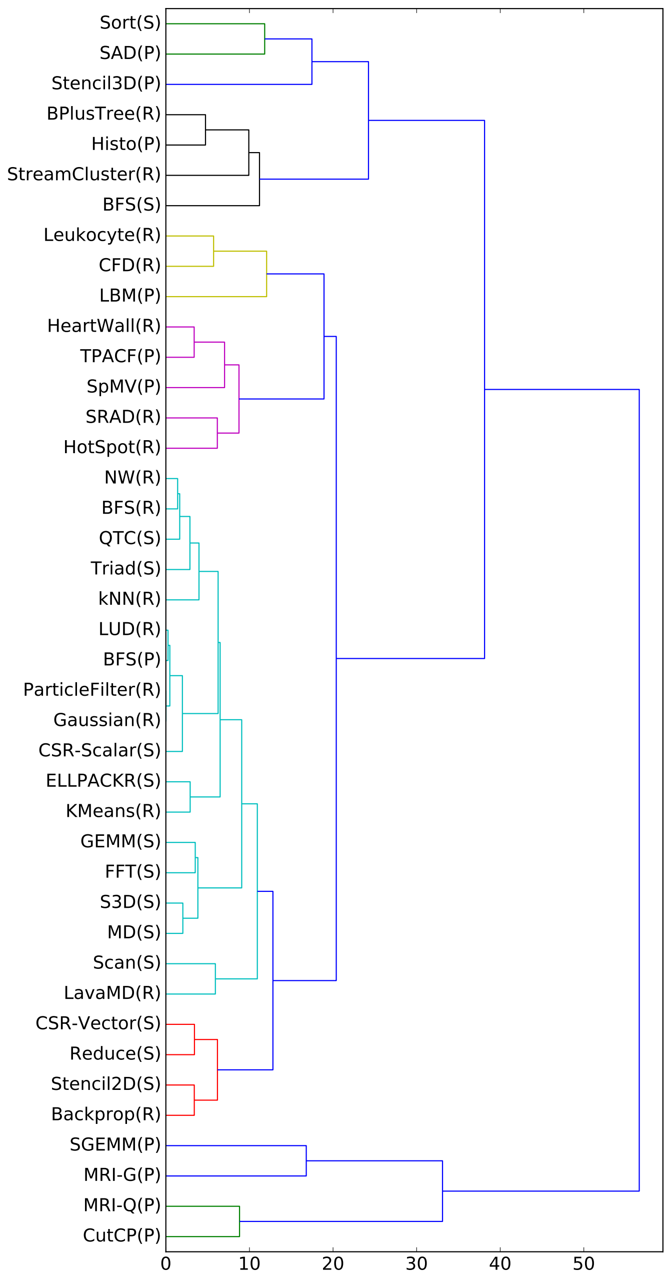

Fig. 9: Diversity analysis for all benchmark suites put together to identify opportunities for expansion. SHOC applications are denoted by (S), Parboil by (P), and Rodinia by (R).

VI. Conclusion

In this paper, we developed a framework for subsetting GPGPU workloads using conventional techniques such as PCA and hierarchical clustering. We identified a set of metrics for GPGPU workloads that could be used for the above task. We validated our framework and choice of metrics using

TABLE XII: Expanding existing benchmark suites

SHOC	Sort, BFS, GEMM, SpMV (vector), **Stencil3D, LBM, TPACF, MatMul, MRI-G, CutCP (all from Parboil)**
Rodinia	Hotspot, Backprop, CFD, LUD, StreamCluster, **SAD, Stencil3D, MatMul, MRI-G, CutCP (all from Parboil)**
Parboil	LBM, BFS, TPACF, Stencil3D, SAD, SGEMM, MRI-G, CutCP, Histo, **Backprop (Rodinia)**

speedup results that were independently reported to SPEC. The results showed that our methodology worked better than the random subsetting approach. The cross-platform results highlighted the need for architecture-independent metrics in such a study which is left for future exploration. We showed that all benchmark suites had something unique to offer in the evaluation space. Similarly, no benchmark suite covered the entire spectrum of benchmark applications. Keeping the above in mind, researchers should adopt a methodological approach to choose an appropriate set of applications for evaluating their proposed techniques.

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.

[21] M. Stephenson, S. K. Sastry Hari, Y. Lee, E. Ebrahimi, D. R. Johnson, D. Nellans, M. O'Connor, and S. W. Keckler, "Flexible Software Profiling of GPU Architectures," in *Proceedings of the 42nd Annual International Symposium on Computer Architecture (ISCA)*, pp. 185–197, ACM, 2015.