Analysis and Characterization of GPU Benchmarks for Kernel Concurrency Efficiency

Pablo Carvalho[1], Lúcia M. A. Drummond[1], Cristiana Bentes[2(✉)], Esteban Clua[1], Edson Cataldo[3], and Leandro A. J. Marzulo[4]

[1] Instituto de Computação, Universidade Federal Fluminense, Niterói, Brazil
pablocarvalho@id.uff.br, {lucia,esteban}@ic.uff.br
[2] Eng. de Sistemas e Computação, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, Brazil
cris@eng.uerj.br
[3] Programa de Pós-graduação em Engenharia Elétrica e de Telecomunicações, Universidade Federal Fluminense, Niterói, Brazil
ecataldo@im.uff.br
[4] Ciência da Computação, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, Brazil
leandro@ime.uerj.br

Abstract. Graphical Processing Units (GPUs) became an important platform to general purpose computing, thanks to their high performance and low cost when compared to CPUs. However, programming GPUs requires a different mindset and optimization techniques that take advantage of the peculiarities of the GPU architecture. Moreover, GPUs are rapidly changing, in the sense of including capabilities that can improve performance of general purpose applications, such as support for concurrent execution. Thus, benchmark suites developed to evaluate GPU performance and scalability should take those aspects into account and could be quite different from traditional CPU benchmarks. Nowadays, Rodinia, Parboil and SHOC are the main benchmark suites for evaluating GPUs. This work analyzes these benchmark suites in detail and categorizes their behavior in terms of computation type (integer or float), usage of memory hierarchy, efficiency and hardware occupancy. We also intend to evaluate similarities between the kernels of those suites. This characterization will be useful to disclosure the resource requirements of the kernels of these benchmarks that may affect further concurrent execution.

1 Introduction

Graphics Processing Units (GPUs) have been gaining prominence in general-purpose computing. Their cost/performance ratio combined with their high computational power, have attracted a broad range of users, going from the world fastest supercomputers to the shared virtual infrastructures such as cloud environments.

E. Mocskos and S. Nesmachnow (Eds.): CARLA 2017, CCIS 796, pp. 71–86, 2018.
https://doi.org/10.1007/978-3-319-73353-1_5

Some benchmark packages have been created over the years to evaluate performance of GPUs in a number of real-world applications. The most important benchmark suites for GPUs are: Rodinia [1], Parboil [2], and SHOC [3]. Unlike benchmarks proposed for CPUs, these benchmarks are composed of a series of kernels, where each kernel represents a task submitted for fine-grain parallelism on the GPU and may have different needs for resources during execution.

The benchmark suites have been used over the years to evaluate aspects of the GPU architecture and its fine-grain parallelism, helping in determining the benefits of new hardware features. In this paper, we are particularly interested in analyzing the individual behavior of the kernels of these benchmarks in terms of resource usage. Our main motivation is the impact that the kernel resource requirements has in further concurrent kernel execution.

Concurrent kernel execution is a relatively recent feature in NVIDIA GPUs. The scheduling policy follows a *left-over* strategy, where the hardware assigns as many resources as possible to one kernel and then assigns the remaining resources to other kernels. In other words, the blocks of one kernel are distributed to the SMs for execution, and if there is space left for more blocks to execute, the blocks of other kernels can execute concurrently. The number of blocks which can execute concurrently on an SM is limited by: (i) the maximum number of active warps per SM imposed by the hardware, (ii) the maximum number of active thread blocks per SM imposed by the hardware, (iii) the number of thread blocks that the shared memory can accommodate given the consumption of each thread block, (iv) the number of thread blocks that the registers can accommodate given the consumption of each thread block. Therefore, a resource-hungry kernel could prevent the concurrent execution of other small kernels. According to Pai *et al.* [4], around 50% of the kernels from the Parboil and Rodinia benchmark suites consume too many resources and prevent concurrent execution of other kernels.

In this scenario, little is known about the behavior of the kernels of these benchmarks in terms of the resources requirements that affect concurrent execution. This work aims at presenting a detailed analysis of the execution of kernels from the three main benchmarks suites, considering the following resource utilization: integer, single and double precision floating point operations, SM efficiency, GPU occupancy and memory operations. Through this analysis, we intend to extend the comprehension of the kernels execution in order to guide further decisions on convenient and more efficient concurrent execution. We also propose to group the kernels with similar characteristics in terms of resource usage. For so, we use the Principal Component Analysis (PCA) statistical method for reducing information dimensionality and K-means clustering for creating the proposed groups.

Our results show that Rodinia and Parboil, although less updated than SHOC, have applications with the highest resource usage. We observed four distinct groups of kernels in the three benchmark suites. The first group concentrates kernels with low resource usage. The second group contains kernels that stress resource usage. The third and fourth groups have kernels with medium

resource utilization and relatively low occupancy. We conclude that the kernels from the third and fourth groups are good candidates for concurrent execution, since they are more likely to leave unused resources.

The rest of the paper is organized as follows. Section 2 presents previous work on GPU benchmark characterization. Section 3 discusses the benchmark suites studied. Section 4 describes the methodology used in our experiments. Section 5 discusses and analyzes our experimental results. Finally, Sect. 6 presents our conclusions and directions for future work.

2 Related Work

There are few previous effort in benchmark suite characterization for GPUs. The work by Che *et al.* [5] characterizes Rodinia applications in terms of instructions per cycle (IPC), memory instruction mix, and warp divergence. Their focus, however, is to compare with the CPU benchmark Parsec. Kerr *et al.* [6] analyze GPU applications from NVIDIA SDK and Parboil in terms of control flow, data flow, parallelism and memory behavior. This work, however, performs the study on a GPU emulator that execute the PTX code of the kernels. The work of Goswami *et al.* [7] presents a set of microarchitecture independent GPU workload characteristics that are capable of capturing five important behaviors: kernel stress, kernel characteristics, divergence characteristics, instruction mix, and coalescing characteristics. They studied Rodinia, Parboil and CUDA SDK, but their study is based on a simulation of a generic GPU, that they have heavily instrumented to extract the studied characteristics.

Burtscher *et al.* [8] study a suite of 13 irregular benchmarks in terms of control-flow irregularity and memory-access irregularity, and compare with regular kernels from CUDA SDK. In a later work, O'Neil and Burtscher [9] presents a microarchitectural workload characterization of five irregular GPU applications from the LonestarGPU benchmark suite[1]. Their study is based on simulations and they also focus on the impact of control flow and memory access irregularity. Bakhoda *et al.* [10] characterizes 12 non-graphics kernels on a GPU simulator. They study the performance impact of some microarchitecture design choices: interconnect topology, use of caches, design of memory controller, parallel workload distribution mechanisms, and memory request coalescing hardware.

Our study, on the other hand, is the first to perform the characterization on the kernels of the three main GPU benchmark suites, rather on the applications as a whole. Our analysis is based on actual executions of the benchmarks on recent GPU Maxwell architecture and rises important characteristics for further understanding concurrent kernel executions.

3 Benchmark Suites

Benchmarks are programs designed to evaluate computational systems. By analyzing the execution of these programs through well-defined metrics, it is possible to learn about the operation of a specific architecture, identify bottlenecks

[1] http://iss.ices.utexas.edu/?p=projects/galois/lonestargpu.

in a program execution and compare different architectures [11]. Nevertheless, it is essential to choose benchmark suites that are able to stress the resource usage of the hardware being evaluated. In [12], authors recommend the use of algorithmic methods that capture a pattern of computation and communication (called "Dwarfs") to design and evaluate parallel programming models and architectures. Moreover, they recommend the use of thirteen Dwarfs instead of traditional benchmarks.

The focus of this paper is to evaluate the benchmarks developed for general purpose computing. The analyzed benchmarks, Rodinia [1,5], Parboil [2] and SHOC [3,13], provide implementations for other processors. However, our goal here is to study their behavior in the context of GPUs.

The Rodinia benchmark package, released in 2009 (now in version 3.1), focuses on the analysis of heterogeneous systems. Rodinia offers 23 applications with CUDA, OpenCL and OpenMP implementations, covering nine of the thirteen Dwarfs.

The Parboil package was developed in 2008 to test and demonstrate the capability of the first generation of GPUs with CUDA technology. According to the concept of its development, the package's composition was designed neither to deliver fully optimized and low-level versions for a particular device, nor to deliver full versions of applications that would discourage modifications. Currently, Parboil is composed of 11 applications, covering a subset of the thirteen Dwarfs. Most of these applications are only implemented in CUDA, while some have a basic CPU implementation. Therefore, this benchmark suite does not seem appropriate for comparing CPUs with GPUs nor evaluating hybrid (GPU+CPU) systems.

Scalable Heterogeneous Computing (SHOC) benchmark suite was designed for GPUs and multi-core processors. Moreover, it provides MPI+CUDA versions that allow the execution using multiple GPUs in a cluster. SHOC applications are organized in three levels: (*i*) level 0 has 6 applications that measure low-level hardware characteristics, such as memory bandwidth and peak FLOPS; (*ii*) level 1 provides 10 applications that correspond to a subset of the thirteen Dwarfs; and (*iii*) level 2 consists of 2 real applications.

4 Methodology

Our kernel characterization targets on their behavior on integer and floating-point operations, SM efficiency, GPU occupancy and memory operations. We extracted these metrics from the NVIDIA *nvprof* tool [14] as seen in Table 1.

Although *nvprof* returns the maximum, average and minimum values, for each metric, we used only the average values, since the variance is not high. From the extracted data, the sum of *shared_load_transactions* and *shared_store_transactions* provides the total transactions on shared memory. The sum of *local_load_transations* and *local_store_transactions* provides the total local transations. The sum of *gld_transactions* and *gsd_transactions* provides total transactions on global memory.

Table 1. Selected metrics and characteristics

Metric	Description
sm_efficiency	Time percentage that at least one *warp* is active in a SM in relation to all the GPU SMs
achieved_occupancy	Active *warps* rate in a single SM in relation to the maximum number of active *warps* supported by the SM
shared_load_transactions	Number of loading operations at the shared memory
shared_store_transactions	Number of writing operations at the shared memory
local_load_transactions	Number of loading operations at the local memory
local_store_transactions	Number of writing operations at the local memory
gld_transactions	Number of reading operations at the global memory
gst_transactions	Number of storage operations at the global memory
inst_fp_32	Number of single precision floating point operations
inst_fp_64	Number of double precision floating point operations
inst_integer	Number of integer operations

The analysis of the measured data follows past work in benchmark characterization [5,7,15–17] in using Principal Component Analysis (PCA) and clustering. The data is first mean-centered and normalized to make it comparable. After that, we used the PCA to reduce the data dimensionality and show the characteristics that contribute most to its variance. PCA returns a number of principal components, that are linear combinations of the original features. The first principal component (PC1) exhibits the largest variance, followed by the second principal component (PC2).

With the results of PC1 vs PC2, we group similar kernels using the K-means grouping technique. The values of K used in the K-means clustering were obtained experimentally for each analysis.

5 Experimental Results

5.1 Experimental Environment

Our experiments were conducted on a GPU GTX 980 (Maxwell architecture) with 2048 CUDA cores running at 1126 MHz in 16 SMs, with 4 GB of global memory and 2 MB of L2 cache. Each SM has 96 KB of shared memory and 256 KB of registers. To compile and run the benchmarks we used CUDA version 7.5. The statistical analysis was performed using the R language. All applications were executed with the standard input data sets.

5.2 Individual Analysis

We first analyze kernels belonging to each benchmark suite separately. In order to distinguish the kernels in the presented charts, without compromising visibility, we used a coding scheme where each kernel is identified by a letter that

corresponds to the application it belongs, and a number that distinguishes it from the other kernels of the same application.

Parboil. For the Parboil analysis, we did not use double precision floating-point operations in the metrics. This is the oldest benchmark suite from the three studied, launched before GPUs had double precision support. Parboil has 11 applications with 26 kernels. We first detect what we call *low-expressive* kernels. These kernels use a small amount of resources and run in a very short time. Thus, they were removed from our analysis. Consider that P_K is the percentage of total execution time required to run kernel K in some application and that M is the mean of the execution times for all kernels of that same application. We will remove K from the analysis if P_K is one standard deviation bellow M. Only 3 kernels were removed.

Parboil applications do not make use of concurrent execution. All kernels are executed on the same stream.

Figure 1 shows the biplot chart for the results of PC1 vs PC2 followed by the K-means clustering ($K = 5$). Each point in the chart represents a kernel (named according to our coding scheme). Arrows denote vectors that correspond to the metrics analyzed. Vector lengths are proportional to the magnitude of each metric, and the angle between two vectors represent the correlation between the corresponding metrics. Vectors with a small angle indicate that metrics are highly correlated. Vectors forming a 90° angle are uncorrelated, and vectors in opposite directions have negative correlation. The proximity of points to the vectors shows kernels with the greater values obtained in that metric.

The direction of vectors in the chart indicates a correlation between the number of shared memory transactions with the number of floating-point operations. This means that Parboil kernels normally use shared memory to store floating point data. We have also noticed a correlation between global memory transactions and SM efficiency. This means that the applications usually have a good number of warps to hide global memory latency, when global memory is intensively used.

The five groups identified by K-means are represented in different colors in Fig. 1 and specified in Table 2. Group 1 is composed of 10 kernels. This group contains Parboil's most representative kernels. These are compute-intensive kernels with high efficiency, a large number of operations with integers and single-precision floating-point, and a relatively high occupancy.

Group 2 contains 2 kernels with high efficiency, but low occupancy. In these kernels, the large number of integer and floating-point operations are enough to maintain the SM busy, and there is no need for more warps to hide latency.

Group 3 has 6 kernels. These kernels are characterized by the average resource usage. The position of its kernels in the chart suggests that the metrics present average values when compared to the more resource-consuming groups (groups 1 and 2) and the less resource-consuming groups (groups 4 and 5).

Group 4 has 3 small kernels from the *mri-gridding* application. These kernels are the less representative in terms of execution time.

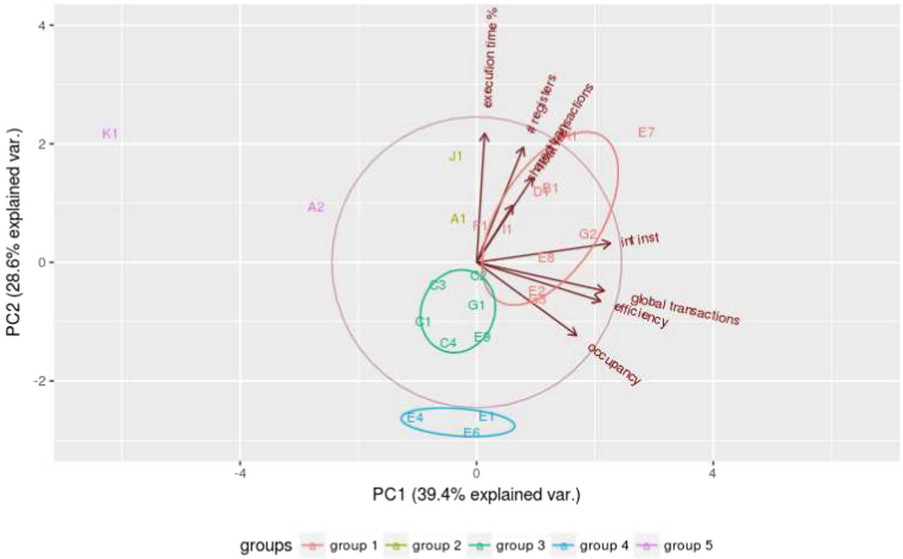

Fig. 1. Parboil results

Table 2. Parboil kernels for each group.

Group	Application	# kernels	Kernels
1	cutcp	1	cuda_cutoff_potential_lattice6overlap(B1)
	lbm	1	performStreamCollide_kernel(D1)
	mri-gridding	3	splitRearrange(E2), splitSort(E8), gridding_GPU(E7)
	mri-q	1	ComputeQ_GPU(F1)
	sad	2	mb_sad_calc(G2), larger_sad_calc_8(G3)
	spmv	1	spmv_jds(I1)
	sgemm	1	mysgemmNT(H1)
2	bfs	1	BFS_kernel_multi_blk_inGPU(A1)
	stencil	1	block2D_hybrid_coarsen_x(J1)
3	histo	4	histo_prescan_kernel(C1), histo_main_kernel(C2), histo_final_kernel(C3), histo_intermediates_kernel(C4)
	mri-gridding	1	scan_L1_kernel(E9)
	sad	1	larger_sad_calc_16(G1)
4	mri-gridding	3	reorder_kernel(E1), uniformAdd(E4), binning_kernel(E6)
5	bfs	1	BFS_in_GPU_kernel(A2)
	tpacf	1	gen_hists(K1)

Group 5 contains only 2 kernels. These are the kernels with the smallest occupancy, which make them good candidates for concurrent execution with other applications kernels. During their execution there is a higher probability of having unused resources that could be allocated to a concurrent application.

Rodinia. In Rodinia, 58 kernels from 22 applications were analyzed[2]. In the detection of *low-expressive* kernels, 14 kernels were removed.

Rodinia applications do not make use of concurrent execution. Except for the Huffman application included in version 3.0 (2015), all other Rodinia applications execute their kernels on the same stream.

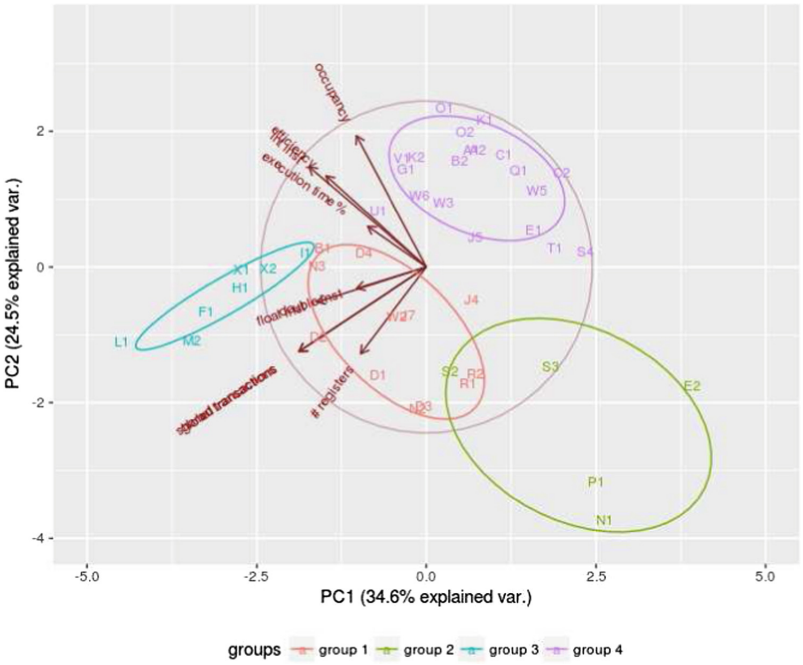

Fig. 2. Rodinia results

Figure 2 shows the biplot chart for the results of PC1 vs PC2 followed by the K-means clustering ($K = 4$). We can observe in this chart that most of the analyzed kernels that make transactions in global memory, also make transactions in shared memory, indicating that Rodinia applications are mostly optimized to take advantage of shared memory. We can also observe a high correlation between single and double precision floating-point operations, although double

[2] We did not analyze the CFD application, since nvprof was not able to correctly extract the corresponding metrics.

precision operations are less pronounced in this benchmark suite. In addition, the correlation between integer operations, SM efficiency, and occupancy indicates that integer-based applications are compute-intensive and have a great number of warps. This maintains the SM busy most of the time.

Four groups were identified and are shown in Table 3. Group 1 has memory-intensive kernels and consists of 14 kernels from 8 applications. Although the most remarkable characteristic of kernels in this group is the amount of memory operations, there are other interesting characteristics: (i) the group does not perform double-precision floating-point operations, (ii) it has variable occupancy while its efficiency is high for the majority of the members, and (iii) most of the kernels perform integer operations.

Group 2 contains 5 kernels. These kernels presented low resource usage and short execution time. All five kernels have little relevance in terms of resource usage, compared to the other groups, and are probably not recommended for assessing the GPU architecture and its parallelism. It is important to notice that all the kernels of the *Myocite* application are in this group.

Group 3 has 6 kernels. This group is characterized by intensive double-precision floating point operations, although it also performs a large number of single precision floating-point operations. These kernels are floating-point based compute intensive.

Group 4 is the one with more kernels (19). The group is characterized by high GPU occupancy and by the large number of integer operations. We can observe in the chart that kernels of this group are plotted in the opposite direction of the vectors corresponding to the memory operations metrics. This means that these kernels are not memory intensive.

SHOC. Compared to the other benchmark suites, SHOC is the suite that received updates more recently. For this reason, the applications S3D and Triad make massive use of concurrent execution, using multiple streams. SHOC contains not only real applications, but also some microbenchmarks. The SHOC level zero applications contains only microbenchmark kernels to test low-level details of the hardware. We did not include these applications in our study. Our analysis focused on 47 kernels of level one and 59 kernels of level two applications. In SHOC, we did not remove *low-expressive* kernels, since level one applications are much smaller than level two applications.

Figure 3 shows the biplot chart for the results of PC1 vs PC2 followed by the K-means clustering ($K = 3$). The chart presents a strong correlation between the percentage of the execution time and the number of operations in shared memory, which means that kernels that take more time to execute are optimized to take advantage of shared memory. Operations in global memory are highly correlated with SM efficiency, which means that kernels that need to access the global memory are able to efficiently hide memory access latency. There is also a certain proximity between operations with integers and with single precision floating point, integers are probably used for controlling loops that contain floating point operations.

Table 3. Rodinia kernels for each group.

Group	Application	# kernels	Kernels
1	backprop	1	bpnn_layerforward_CUDA(B1)
	dwt2d	4	dwt_cuda::fdwt97Kernel<int=128, int=6>(D1), dwt_cuda::fdwt97Kernel<int=192, int=8>(D2), dwt_cuda::fdwt97Kernel<int=64, int=6>(D3), c_CopySrcToComponents<float>(D4)
	huffman	1	vlc_encode_kernel_sm64huff(I1)
	hybridsort	2	bucketsort(J4) bucketcount(J7)
	lud	2	lud_perimeter(N2), lud_internal(N3)
	nw	2	needle_cuda_shared_1(R1), needle_cuda_shared_2(R2)
	pathfinder	1	dynproc_kernel(U1)
	srad-v1	1	reduce(W2)
2	gaussian	1	Fan1(E2)
	lud	1	lud_diagonal(N1)
	myocyte_10	1	kernel(P1)
	particlefilter-float	2	likelihood_kernel(S2), normalize_weights_kernel(S3)
3	heartwall	1	kernel(F1)
	hotspot	1	calculate_temp(H1)
	lavaMD	1	kernel_gpu_cuda(L1)
	leukocyte	1	IMGVF_kernel(M2)
	srad-v2	2	srad_cuda_1(X1), srad_cuda_2(X2)
4	b+tree	2	findRangeK(A1), findK(A2)
	backprop	1	bpnn_adjust_weights_cuda(B2)
	bfs	2	Kernel(C1), Kernel2(C2)
	gaussian	1	Fan2(E1)
	hotspot3d	1	hotspotOpt1(G1)
	hybridsort	1	mergeSortPass(J5)
	kmeans	2	invert_mapping(K1), kmeansPoint(K2)
	mummergpu	2	mummergpuKernel(O1), printKernel(O2)
	nn	1	euclid(Q1)
	particlefilter-float	1	find_index_kernel(S4)
	particlefilter-naive	1	kernel(T1)
	sc_gpu	1	kernel_compute_cost(V1)
	srad-v1	3	srad2(W3), prepare(W5), srad(W6)

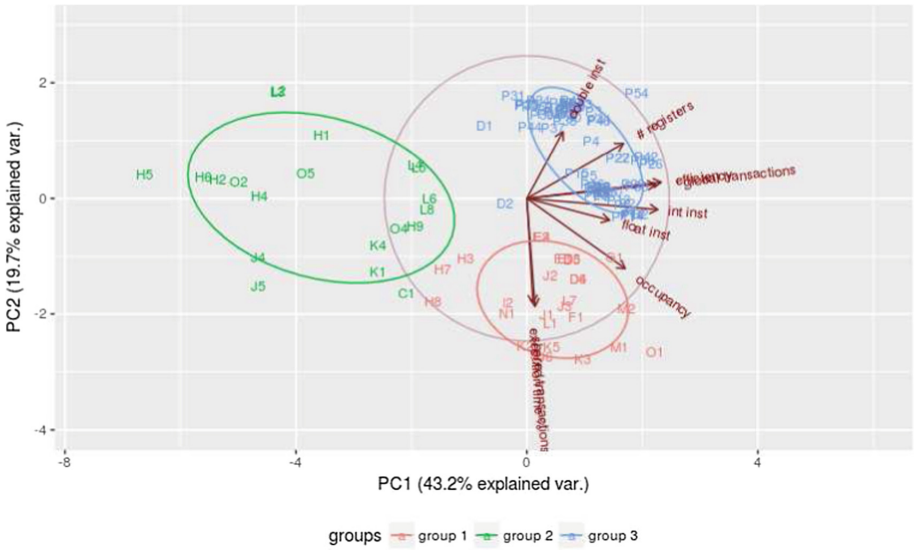

Fig. 3. SHOC results

Three groups of kernels were identified as show in Table 4. Group 1 is composed of 28 memory-intensive kernels that present the higher execution time. This group is mostly composed of level one parallel algorithms, and is the most diverse group. Kernels in this group use more operations with integers than the others.

Group 2 contains 20 kernels that have low significance. The chart shows that kernels of this group are positioned in the opposite direction of the vectors of the metrics, indicating that kernels in this group do not consume much of the analyzed resources.

Group 3 contains 58 kernels mostly from the S3D application. For these kernels, we observed short execution times and low occupancy. These characteristics confirm the capability of S3D to massively exploit concurrent execution. Kernels warps do not occupy the whole SM, and the underutilized resources can be allocated to other kernels. Most of the kernels do not take advantage of shared memory, but provide high efficiency. We can also observe two subgroups among the S3D kernels, one that performs a high number of double-precision floating-point operations, while the other subgroup does not.

5.3 Global Analysis

In this analysis, we assembled all kernels of the benchmark suites, with a total number of 173 kernels. The motivation for this analysis is to show the similarities and differences between applications of the benchmark suites.

Figure 4 shows the biplot chart for the results of PC1 vs PC2 followed by the K-means clustering ($K = 4$). The name of the kernels in this chart is formed

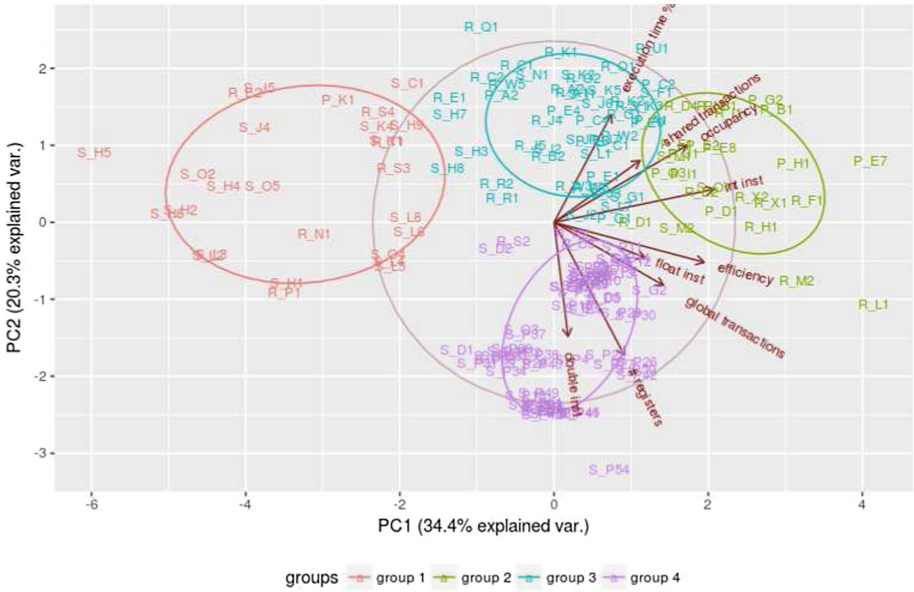

Fig. 4. Results for the kernels of all benchmarks suites.

with a similar coding scheme used in the previous analysis, but each kernel name starts with a letter indicating which benchmark suite it belongs (R, P or S). Comparing the direction of the vectors in this chart with the SHOC chart, we can observe that the large number of kernels in SHOC influenced the position of some vectors such as the percentage of execution time and the number of double-precision floating-point instructions. The position and angle between these two vectors are very similar.

Four groups of kernels were identified and are analyzed separately as follows.

Group 1 - Little Resource Usage. This group contains 27 kernels, and is a result of the combination of the kernels from benchmark suites that present low use of all the resources analyzed (integer and floating-point operations, SM efficiency, GPU occupancy and memory operations). Table 5 shows the number of kernels of each benchmark suite that comprises this group. We can observe in this table that most of the kernels in this group belongs to SHOC. Most of them are level one SHOC parallel algorithms.

Group 2 - High Resource Utilization. Group 2 consists of 26 kernels that have high efficiency, due to a great number of integer and single-precision floating-point operations. These kernels also have an average occupancy of about 70%. This is the group that present the highest resource utilization. Table 5 shows the number of kernels per benchmark suite. We can observe that Rodinia and Parboil have more kernels that intensively exploit the GPU resources than SHOC.

Table 4. SHOC kernels from each group.

Group	Application	# kernels	Kernels
1	FFT	4	FFT512_device(D3), IFFT512_device(D4), IFFT512_device(D5), FFT512_device(D6)
	GEMM	4	maxwell_sgemm_128x64_nn(E1), gemm_kernel2x2_tile_multiple_core(E2), maxwell_sgemm_128x64_nt(E3), gemm_kernel2x2_tile_multiple_core(E4)
	MD5Hash	1	FindKeyWithDigest_Kernel(F1)
	MD	1	compute_lj_force(G1)
	NeuralNet	3	axpy_kernel_val(H3), kernelBackprop1(H7), gemm_kernel1x1_core(H8)
	Reduction	2	reduce(I1), reduce(I2)
	Scan	4	reduce(J1), reduce(J2), bottom_scan(J3), bottom_scan(J6)
	Sort	3	findRadixOffsets(K2), radixSortBlocks(K3), reorderData(K5)
	Spmv	2	spmv_csr_vector_kernel(L1), spmv_csr_vector_kernel(L7)
	Stencil2D	2	StencilKernel(M1), StencilKernel(M2)
	Triad	1	Triad(N1)
	QtClustering	1	QTC_device(O1)
2	BFS	1	BFS_kernel_warp(C1)
	NeuralNet	6	kernelFeedForward3(H1), kernelBackprop3b(H2), kernelBackprop3a(H4), kernelInitNablaB(H5), kernelBackprop2(H6), kernelInitNablaW(H9)
	Scan	2	scan_single_block(J4), scan_single_block(J5)
	Sort	2	scan(K1), vectorAddUniform4(K4)
	Spmv	6	zero(L2), zero(L3), spmv_ellpackr_kernel(L4), spmv_csr_scalar_kernel(L5), spmv_ellpackr_kernel(L6), spmv_csr_scalar_kernel(L8)
	QtClustering	3	reduce_card_device(O2), trim_ungrouped_pnts_indr_array(O4), update_clustered_pnts_mask(O5)
3	FFT	2	void chk512_device(D1), chk512_device(D2)
	MD	1	compute_lj_force(G2)
	QtClustering	1	compute_degrees(O3)
	s3d	54	all kernels (P1 to P54)

Group 3 - Medium Resource Utilization. This group is composed of 51 kernels, and has some similar characteristics to group 2, high number of integer operations and average occupancy around 60%. Kernels in this group are smaller than the ones in group 2, presenting a less significant percentage of execution time. Table 5 shows the number of kernels from each suite in this group. We observe that this group contains a similar number of kernels from the three suites.

Group 4 - Low Occupancy and High Efficiency. This group is composed of 69 kernels, characterized by low occupancy, high efficiency and low percentage of execution time. This group contains mostly S3D kernels from the SHOC suite. From the 69 kernels, 44 are from S3D. This application is a computational chemistry application that solves Navier-Stokes equations for a regular 3D domain [18]. The computation is floating-point intensive, and it was parallelized by assigning each 3D grid point to one thread. The low occupancy of each of its kernels impels their concurrent implementation. Another feature of this group is the smaller number of operations with integers and the highest average use of registers than the other groups. Table 5 shows the distribution of kernels of this group in the suites. Notice that rhere are no Parboil kernels in this group, and there are only three kernels from Rodinia.

Table 5. Number of kernels for each group and benchmark suite.

Group	Patboil	Rodinia	SHOC
1	1	6	20
2	11	12	3
3	11	23	17
4	0	3	66

5.4 Discussion

Our results show that Rodinia and Parboil presented more diversity in their kernels. SHOC, on the other hand, provides less diversity but it is the only suite that exploits kernel concurrency massively. When the three suites are analyzed together, we observed four distinct groups of kernels: (1) Low significance, (2) High resource utilization, (3) Medium resource utilization and (4) Low occupancy and high efficiency. Kernels from group 1 are the ones with short execution time and low resource usage, which means that they are not appropriate for assessing the GPU hardware. Kernels from group 2 present high resource utilization, which indicates that they are not good candidates for concurrent kernel execution. Kernels from groups 3 and 4 have medium resource utilization and relatively low occupancy. These kernels are more likely to leave unused resources and provide space for concurrent execution.

6 Concluding Remarks

This work presented a detailed characterization of the three most important benchmark suites for GPUs, Rodinia, Parboil and SHOC. Our study focused on revealing the behavior of the kernels in terms of integer, single and double precision operations, SM efficiency, GPU occupancy and memory operations. We also proposed to group similar kernels in order to identify the ones with similar behavior.

The analysis and characterization of representative GPU kernels is an essential step to understand the effect of resource requirements in further concurrent execution in modern GPUs. Cross-kernel interference can drastically affect performance of applications executed in GPUs concurrently. The problem is caused by concurrent access of co-located kernels to shared resources. We believe that identifying kernels with complementary access profiles to execute concurrently can reduce interference among them. Thereby, the characterization is a first and fundamental step to be used in future strategies of kernels scheduling in GPUs.

Our results showed that the benchmarks have kernels with good diversity in terms of resource usage. We identified groups of kernels with similar behavior and distinguished the kernels that are more likely to leave unused resources. These kernels are better candidates for efficient concurrent execution.

Concurrent kernel execution is a relatively new feature in GPUs. It would be interesting that future benchmark suites for GPU exploit this feature to the full. As future work, we intend to analyze the kernels behavior in different GPU architectures. We also intend to perform a study on the performance interference of the concurrent execution of different types of kernels, and propose a intelligent strategy to benefit from all the information gathered.

References

1. Che, S., Boyer, M., Meng, J., Tarjan, D., Sheaffer, J.W., Lee, S.-H., Skadron, K.: Rodinia: a benchmark suite for heterogeneous computing. In: Proceedings of the IEEE International Symposium on Workload Characterization (IISWC), pp. 44–54 (2009)
2. Stratton, J.A., Rodrigues, C., Sung, I.-J., Obeid, N., Chang, L.-W., Anssari, N., Liu, G.D., Hwu, W.M.W.: Parboil: a revised benchmark suite for scientific and commercial throughput computing (2012)
3. Danalis, A., Marin, G., McCurdy, C., Meredith, J.S., Roth, P.C., Spafford, K., Tipparaju, V., Vetter, J.S.: The scalable heterogeneous computing (SHOC) benchmark suite. In: Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units, pp. 63–74 (2010)
4. Pai, S., Thazhuthaveetil, M.J., Govindarajan, R.: Improving GPGPU concurrency with elastic kernels. In: ACM SIGPLAN Notices, vol. 48, pp. 407–418. ACM (2013)
5. Che, S., Sheaffer, J.W., Boyer, M., Szafaryn, L.G., Wang, L., Skadron, K.: A characterization of the Rodinia benchmark suite with comparison to contemporary CMP workloads. In: Proceedings of the IEEE International Symposium on Workload Characterization (2010)

6. Kerr, A., Diamos, G., Yalamanchili, S.: A characterization and analysis of PTX kernels. In: IEEE International Symposium on Workload Characterization, IISWC 2009, pp. 3–12. IEEE (2009)
7. Goswami, N., Shankar, R., Joshi, M., Li, T.: Exploring GPGPU workloads: characterization methodology, analysis and microarchitecture evaluation implications. In: 2010 IEEE International Symposium on Workload Characterization (IISWC), pp. 1–10. IEEE (2010)
8. Burtscher, M., Nasre, R., Pingali, K.: A quantitative study of irregular programs on GPUs. In: 2012 IEEE International Symposium on Workload Characterization (IISWC), pp. 141–151. IEEE (2012)
9. O'Neil, M.A., Burtscher, M.: Microarchitectural performance characterization of irregular GPU kernels. In: 2014 IEEE International Symposium on Workload Characterization (IISWC), pp. 130–139. IEEE (2014)
10. Bakhoda, A., Yuan, G.L., Fung, W.W., Wong, H., Aamodt, T.M.: Analyzing CUDA workloads using a detailed GPU simulator. In: IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS 2009, pp. 163–174. IEEE (2009)
11. Bienia, C.: Benchmarking Modern Multiprocessors. Princeton University, Princeton (2011)
12. Asanovic, K.: The landscape of parallel computing research: a view from Berkeley, Technical report UCB/EECS-2006-183, EECS Department, University of California, Berkley, CA, USA (2006)
13. SHOC (2012). https://github.com/vetter/shoc/wiki
14. NVIDIA Corporation: Profiler user's guide (2017). http://docs.nvidia.com/cuda/profiler-users-guide/index.html#nvprof-overview, an optional note
15. Bienia, C.: Benchmarking modern multiprocessors, Ph.D. thesis, Princeton University (2011)
16. Joshi, A., Phansalkar, A., Eeckhout, L., John, L.K.: Measuring benchmark similarity using inherent program characteristics. IEEE Trans. Comput. **55**(6), 769–782 (2006)
17. Che, S., Skadron, K.: Benchfriend: correlating the performance of GPU benchmarks. Int. J. High Perform. Comput. Appl. **28**(2), 238–250 (2014)
18. Spafford, K., Meredith, J., Vetter, J., Chen, J., Grout, R., Sankaran, R.: Accelerating S3D: a GPGPU case study. In: Lin, H.-X., Alexander, M., Forsell, M., Knüpfer, A., Prodan, R., Sousa, L., Streit, A. (eds.) Euro-Par 2009. LNCS, vol. 6043, pp. 122–131. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14122-5_16