Delft University of Technology
Parallel and Distributed Systems Report Series

A Study of Application Kernel Structure for Data Parallel Applications

Jie Shen, Ana Lucia Varbanescu, Xavier Martorell, Henk Sips
{j.shen,h.j.sips}@tudelft.nl, a.l.varbanescu@tudelft.nl, xavim@ac.upc.edu

Report number PDS-2015-001

ISSN 1387-2109

Published and produced by:
Parallel and Distributed Systems Group
Department of Software and Computer Technology
Faculty of Electrical Engineering, Mathematics, and Computer Science
Delft University of Technology
Mekelweg 4
2628 CD Delft
The Netherlands

Information about Parallel and Distributed Systems Report Series:
reports@pds.ewi.tudelft.nl

Information about Parallel and Distributed Systems Group:
http://www.pds.ewi.tudelft.nl/

J. Shen, A.L. Varbanescu, X. Martorell, H. Sips

A Study of Application Kernel Structure

In this paper, we study the application kernel structure for data parallel applications. The application kernel structure includes two aspects, the number of kernels in the application and their execution flow. Based on the analysis of application kernel structure, we classify data parallel applications into five classes. To examine the coverage of the classification, we check five benchmark suites with a total of 86 applications, and show that the five classes cover all 86 applications. The classification makes it possible to design efficient partitioning strategies for data parallel applications, and to propose an application-driven method that selects the best performing partitioning strategy for a given workload.

Contents

List of Figures

List of Tables

1 Application Kernel Structure

Our work focuses on data parallel applications, where there is massive parallelism to exploit. Each parallel section of code in the program is called a *kernel*, and a data parallel application can have one or multiple kernels executed in certain sequences.

We want to propose a right classification of data parallel applications, as a right classification makes it possible to propose efficient partitioning strategies. However, simply using the number of kernels to classify the applications is not sufficient, so we define *application kernel structure*, and use it to guide our classification.

The application kernel structure includes two aspects: the number of kernels and the type of kernel execution flow. The kernel execution flow can be a sequence (one kernel after another), a loop, or a full DAG (kernel execution forming a directed acyclic graph).

2 Application Classification

Based on the application kernel structure, we propose five application classes. Figure 1 shows the classification results. In the figure, each rectangle represents a kernel, and its length represents the amount of computation the kernel needs to perform. The arrows show the kernel execution flow.

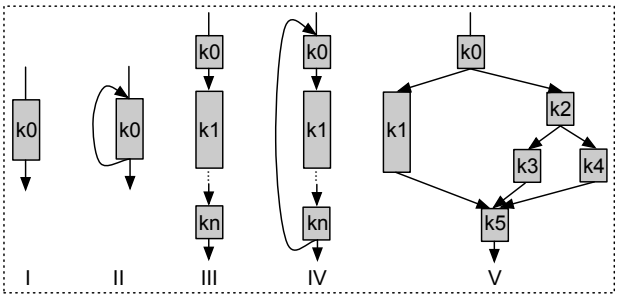

Figure 1: Application classification results and the typical kernel structure of each application class.

The five application classes are listed as follows:

- **SK-One** (Class I) only has a single kernel.
- **SK-Loop** (Class II) also has a single kernel, but the kernel is iterated in a loop.
- **MK-Seq** (Class III) has multiple kernels of different kinds, and the kernel execution follows a sequence.
- **MK-Loop** (Class IV) has multiple kernels of different kinds. The kernels are executed in a sequence, and the sequence is further iterated in a loop.
- **MK-DAG** (Class V) has multiple kernels of different kinds. The kernel execution forms a DAG, and therefore the execution is expected to be very dynamic.

3 Examining the Classification

To determine the coverage of applications by these five classes, we examine five benchmark suites, which are

- The **Parboil** Benchmark Suite [1]: 12 applications.

- The **SHOC** Benchmark Suite [2]: 14 applications.
- The **Rodinia** Benchmark Suite [3]: 17 applications.
- The **Nvidia** OpenCL SDK Code Samples [4]: 30 applications.
- The **Mont-Blanc** Benchmark Suite (obtained from Barcelona Supercomputing Center): 13 applications.

There are 86 applications in total. We analyze the kernel structure per application, and identify the class the application belongs to. Table 1 lists the examination results. We also count the number of applications that fall in each class, and calculate the percentage.

Table 1: Classification examination.

Origin	Application	SK-One	SK-Loop	MK-Seq	MK-Loop	MK-DAG
Parboil	bfs-base		•			
	bfs-nvidia					•
	cutcp		•			
	histogram			•		
	lbm		•			
	mri-gridding			•		
	mri-q			•		
	sad					•
	sgemm	•				
	spmv-jds	•				
	stencil		•			
	tpacf	•				
SHOC	bfs-1		•			
	bfs-2					•
	fft			•		
	gemm	•				
	md	•				
	md5hash	•				
	reduction	•				
	scan			•		
	sort			•		
	spmv-ellpack	•				
	spmv-csr	•				
	stencil2d			•		
	triad		•			
	s3d					•
Rodinia	backprop			•		
	bfs				•	
	cfd					•
	gaussian				•	
	heartwall		•			
	hotspot		•			
	kmeans		•			
	lavaMD	•				
	leukocyte			•		
	lud			•		
	nn	•				
	nw			•		
	particlefilter-naive		•			
	particlefilter-float				•	
	pathfinder		•			
	srad			•		
	streamcluster				•	

Nvidia	oclBlackScholes	•				
	oclBoxFilter			•		
	oclConvolutionSeparable			•		
	oclDCT8x8	•				
	oclDotProduct	•				
	oclDXTCompression		•			
	oclFDTD3d		•			
	oclHiddenMarkovModel			•		
	oclHistogram			•		
	oclMarchingCubes			•		
	oclMatrixMul	•				
	oclMatVecMul	•				
	oclMedianFilter	•				
	oclMersenneTwister			•		
	oclNbody		•			
	oclParticles				•	
	oclPostprocessGL	•				
	oclQuasirandomGenerator			•		
	oclRadixSort				•	
	oclRecursiveGaussian			•		
	oclReduction			•		
	oclScan			•		
	oclSimpleGL	•				
	oclSimpleTexture3D	•				
	oclSoberFilter	•				
	oclSortingNetworks				•	
	oclTranspose	•				
	oclTridiagonal			•		
	oclVectorAdd	•				
	oclVolumeRender	•				
Mont-Blanc	2d-convolution	•				
	3d-stencil	•				
	atomic-monte-carlo-dynamics	•				
	dense-matrix-multiplication	•				
	fft					•
	histogram	•				
	merge-sort			•		
	n-body		•			
	reduction			•		
	sparse-matrix-vector-multiplication	•				
	vector-operation	•				
	cholesky					•
	stream*				•	
Count		31	15	25	8	7
%		36%	17%	29%	9%	8%

*STREAM originates from the STREAM benchmark [5]

In Classes III–V, there are some applications in which one or some of kernels are iterated in a loop. The actual execution of that loop can be unfolded/unrolled. Therefore, the loop structure does not affect the application's main kernel structure.

Summary: The study shows that the five application classes cover all 86 applications. The application distribution is not even. SK-One is the largest application class, which has 31 applications (36% of all applications). MK-Seq, SK-Loop, and MK-Loop are the second, third, and fourth largest classes, having 25, 15, and 8 applications, respectively. MK-DAG has the least number of applications, with 7 applications (8%) falling in this class.

References

[1] John A Stratton, Christopher Rodrigues, I-Jui Sung, Nady Obeid, Li-Wen Chang, Nasser Anssari, Geng Daniel Liu, and W-M Hwu. Parboil: A Revised Benchmark Suite for Scientific and Commercial Throughput Computing. Technical Report IMPACT-12-01, University of Illinois at Urbana-Champaign, 2012. 4

[2] Anthony Danalis, Gabriel Marin, Collin McCurdy, Jeremy S. Meredith, Philip C. Roth, Kyle Spafford, Vinod Tipparaju, and Jeffrey S. Vetter. The Scalable Heterogeneous Computing (SHOC) Benchmark Suite. In *GPGPU 2010*, pages 63–74, 2010. 5

[3] Shuai Che, Michael Boyer, Jiayuan Meng, David Tarjan, Jeremy W. Sheaffer, Sang-Ha Lee, and Kevin Skadron. Rodinia: A Benchmark Suite for Heterogeneous Computing. In *IISWC 2009*, pages 44–54, 2009. 5

[4] Nvidia. https://developer.nvidia.com/opencl. 5

[5] John D. McCalpin. STREAM: Sustainable Memory Bandwidth in High Performance Computers. http://www.cs.virginia.edu/stream/. 6