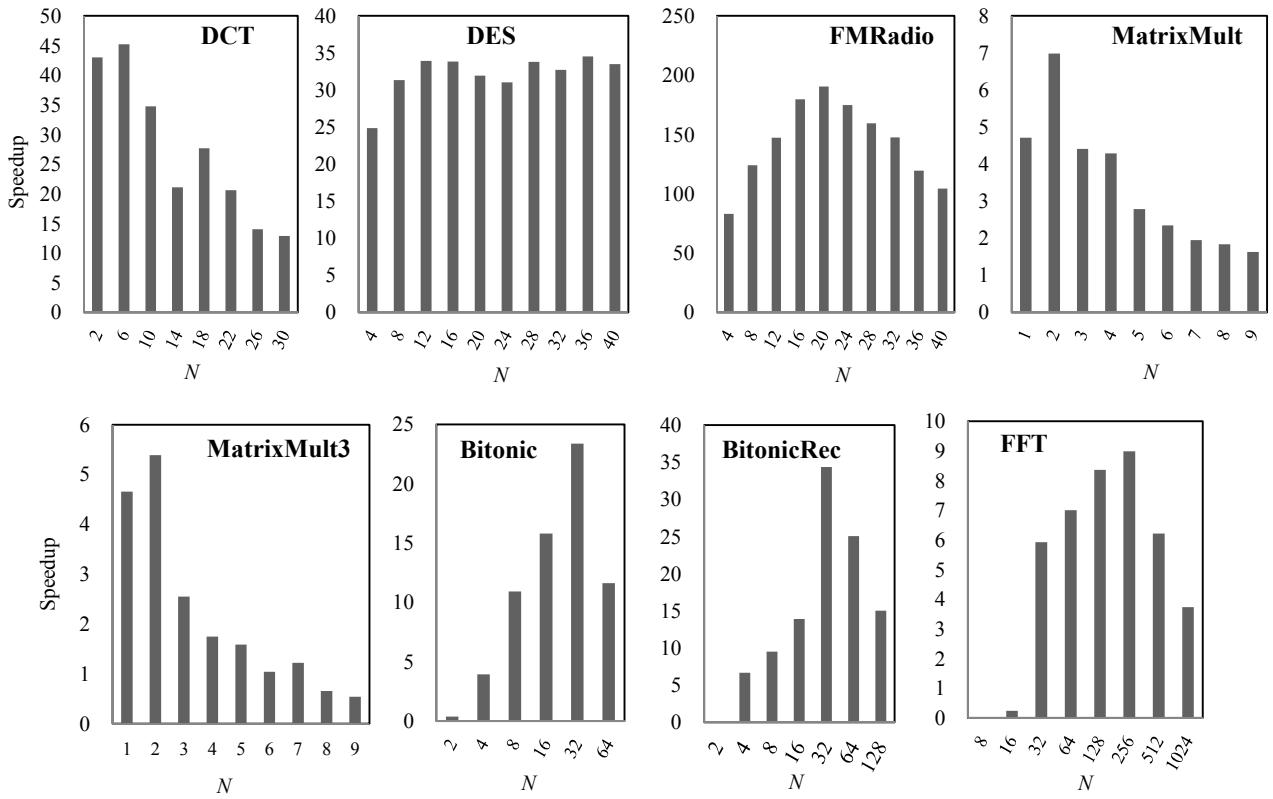

Figure 6. Mapping to a single GPU. The speedup is reported relative to a CPU implementation.

8. Conclusion

We proposed a scalable framework that automatically maps most stream processing applications onto GPUs. We developed an efficient graph partitioning algorithm that splits the complex application into several partitions, each of which can utilize the small on-chip SM memory effectively, and hence achieve good performance. Our proposed strategy obtained performance that augments that of a previous single partition solution. In addition, our proposed strategy is able to scale the performance to up to four GPUs. We also support stateful filters by running them on the CPU cores. The code generation scheme proposed is able to orchestrate the exchange of data within the individual GPUs, between the multiple GPUs, as well as the GPUs and the CPU cores. The results indicate the scalability and improvement when several GPUs are targeted.

It is conceivable that certain embarrassingly parallel applications can be mapped successfully to large scale multi-node, multi-GPU systems. However, on a single node, it is unlikely that the number of GPUs per node will increase significantly beyond the current four due to power, interconnect, and form-factor issues. Our work is the first to show that complex and often tightly coupled streaming applications can be successfully partitioned and mapped automatically onto multiple GPUs. As future work, we would like to investigate how even larger and more complex applications can be specified in StreamIt (or its derivative) so as to run on a large cluster of multi-GPU nodes.

Benchmark	Description	Original N [11]
Bitonic	Sorting algorithm for N float elements applying the bitonic algorithm	8
BitonicRec	Same as above, recursive method	8
DCT	Discrete Cosine Transform for a matrix of $N \times N$ floats	8
DES	DES encryption algorithm with N rounds, input 8 bytes, output as 16 hex digits	16
FFT	Fine grained FFT transform on N elements	32
FMRadio	$(N+3)$-band equalizer radio	8
MatrixMult	Blocked matrix multiplication algorithm for $2N \times 2N$ matrices, split into blocks of 2×2	2
MatrixMult3	Same as above for $(3N+3) \times (3N+3)) \times ((3N+3) \times 3N)$ matrices, with blocks of 3×3	–

Table 1. Benchmark characterization.

References

[1] NVIDIA CUDA 4.0. http://developer.nvidia.com/cuda-toolkit-40.

[2] Streamit benchmarks. http://groups.csail.mit.edu/cag/streamit/shtml/benchmarks.shtml.

[3] S. S. Baghsorkhi, M. Delahaye, S. J. Patel, W. D. Gropp, and W.-m. W. Hwu. An adaptive performance modeling tool for GPU architectures. In *The 15th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP '10)*, 2010.

[4] I. Buck, T. Foley, D. Horn, J. Sugerman, K. Fatahalian, M. Houston, and P. Hanrahan. Brook for GPUs: stream computing on graphics hardware. In *ACM SIGGRAPH '04*, 2004.

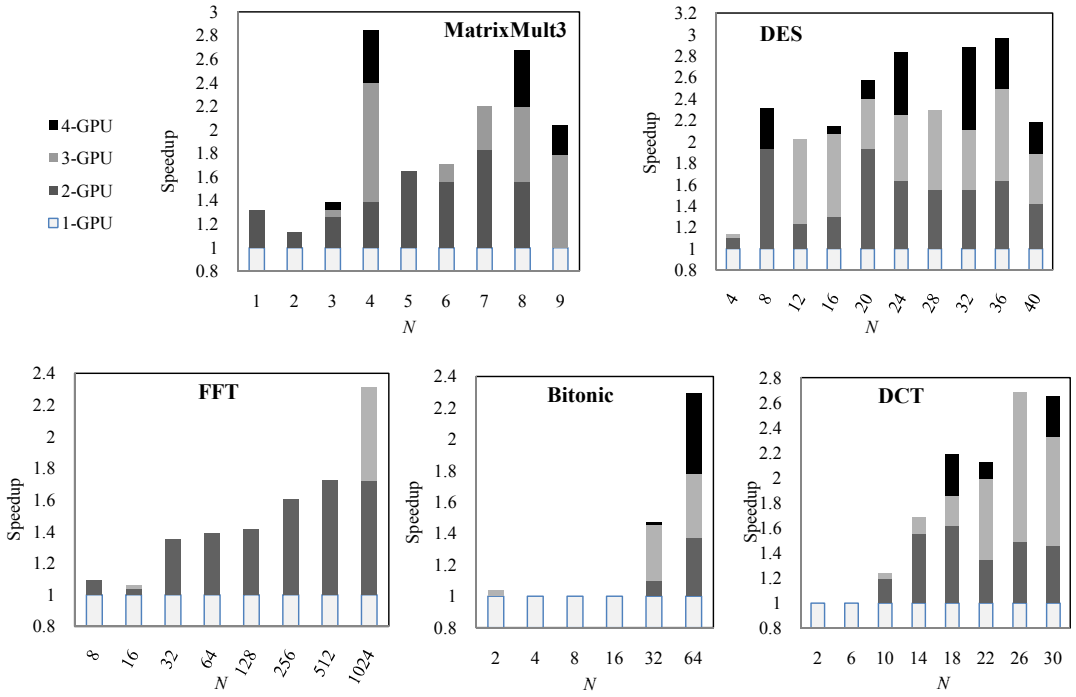

Figure 7. Additional speedup resulted from the mapping to multiple GPUs compared to a single GPU.

[5] L. Chen, O. Villa, S. Krishnamoorthy, and G. R. Gao. Dynamic load balancing on single- and multi-GPU systems. In *2010 IEEE International Parallel and Distributed Processing Symposium (IPDPS'10)*, 2010.

[6] G. Diamos and S. Yalamanchili. Speculative execution on multi-GPU systems. In *2010 IEEE International Parallel and Distributed Processing Symposium (IPDPS '10)*, 2010.

[7] C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *The 19th Design Automation Conference (DAC '82)*, 1982.

[8] M. I. Gordon, W. Thies, M. Karczmarek, J. Lin, A. S. Meli, A. A. Lamb, C. Leger, J. Wong, H. Hoffmann, D. Maze, and S. Amarasinghe. A stream compiler for communication-exposed architectures. In *The 10th international conference on Architectural support for programming languages and operating systems (ASPLOS '02)*, Oct 2002.

[9] M. I. Gordon, W. Thies, and S. Amarasinghe. Exploiting coarse-grained task, data, and pipeline parallelism in stream programs. In *The 12th international conference on Architectural support for programming languages and operating systems (ASPLOS '06)*, 2006.

[10] A. Hagiescu, W.-F. Wong, D. F. Bacon, and R. Rabbah. A computing origami: folding streams in FPGAs. In *The 46th Annual Design Automation Conference (DAC '09)*, 2009.

[11] A. Hagiescu, H. P. Huynh, W. F. Wong, and R. S. M. Goh. Automated architecture-aware mapping of streaming applications onto GPUs. In *2011 IEEE International Parallel and Distributed Processing Symposium (IPDPS '11)*, 2011.

[12] A. H. Hormati, M. Samadi, M. Woh, T. Mudge, and S. Mahlke. Sponge: portable stream programming on graphics engines. In *The 16th international conference on Architectural support for programming languages and operating systems (ASPLOS '11)*, 2011.

[13] H. P. Huynh, Y. Liang, and T. Mitra. Efficient custom instructions generation for system-level design. In *2010 International Conference on Field-Programmable Technology (FPT '10)*, 2010.

[14] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 1998.

[15] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 1970.

[16] Khronos OpenCL Working Group. *The OpenCL Specification, version 1.0.29*, 8 December 2008.

[17] M. Kudlur and S. Mahlke. Orchestrating the execution of stream programs on multicore platforms. In *The 2008 ACM SIGPLAN conference on Programming language design and implementation (PLDI '08)*, 2008.

[18] E. A. Lee and D. G. Messerschmitt. Static scheduling of synchronous data flow programs for digital signal processing. *IEEE Transactions on Computers*, 36(1), 1987.

[19] J. Nickolls and W. J. Dally. The GPU computing era. *IEEE Micro*, 30, 2010. ISSN 0272-1732.

[20] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krger, A. E. Lefohn, and T. J. Purcell. A survey of general-purpose computation on graphics hardware. *Computer Graphics Forum*, 26(1), 2007.

[21] D. Schaa and D. Kaeli. Exploring the multiple-GPU design space. In *2011 IEEE International Parallel and Distributed Processing Symposium (IPDPS'11)*, 2009.

[22] J. A. Stuart and J. D. Owens. Multi-GPU MapReduce on GPU clusters. In *2011 IEEE International Parallel and Distributed Processing Symposium (IPDPS '11)*, 2011.

[23] A. Udupa, R. Govindarajan, and M. J. Thazhuthaveetil. Software pipelined execution of stream programs on GPUs. In *The 7th annual IEEE/ACM International Symposium on Code Generation and Optimization (CGO '09)*, 2009.

[24] H. Wong, M.-M. Papadopoulou, M. Sadooghi-Alvandi, and A. Moshovos. Demystifying GPU microarchitecture through microbenchmarking. In *2010 IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS '10)*, 2010.

[25] Y. Zhang and J. D. Owens. A quantitative performance analysis model for GPU architectures. In *(The 17th International Symposium on High Performance Computer Architecture (HPCA '11))*, 2011.