Improving the Open64 Backend for GPUs

Northeastern

Rodrigo Dominguez, David R. Kaeli
Dept. of Electrical and Computer Engineering
Northeastern University, Boston, MA
{rdomingu,kaeli}@ece.neu.edu

John Cavazos
Dept. of Computer and Information Science
University of Delaware, Newark, DE
cavazos@cis.udel.edu

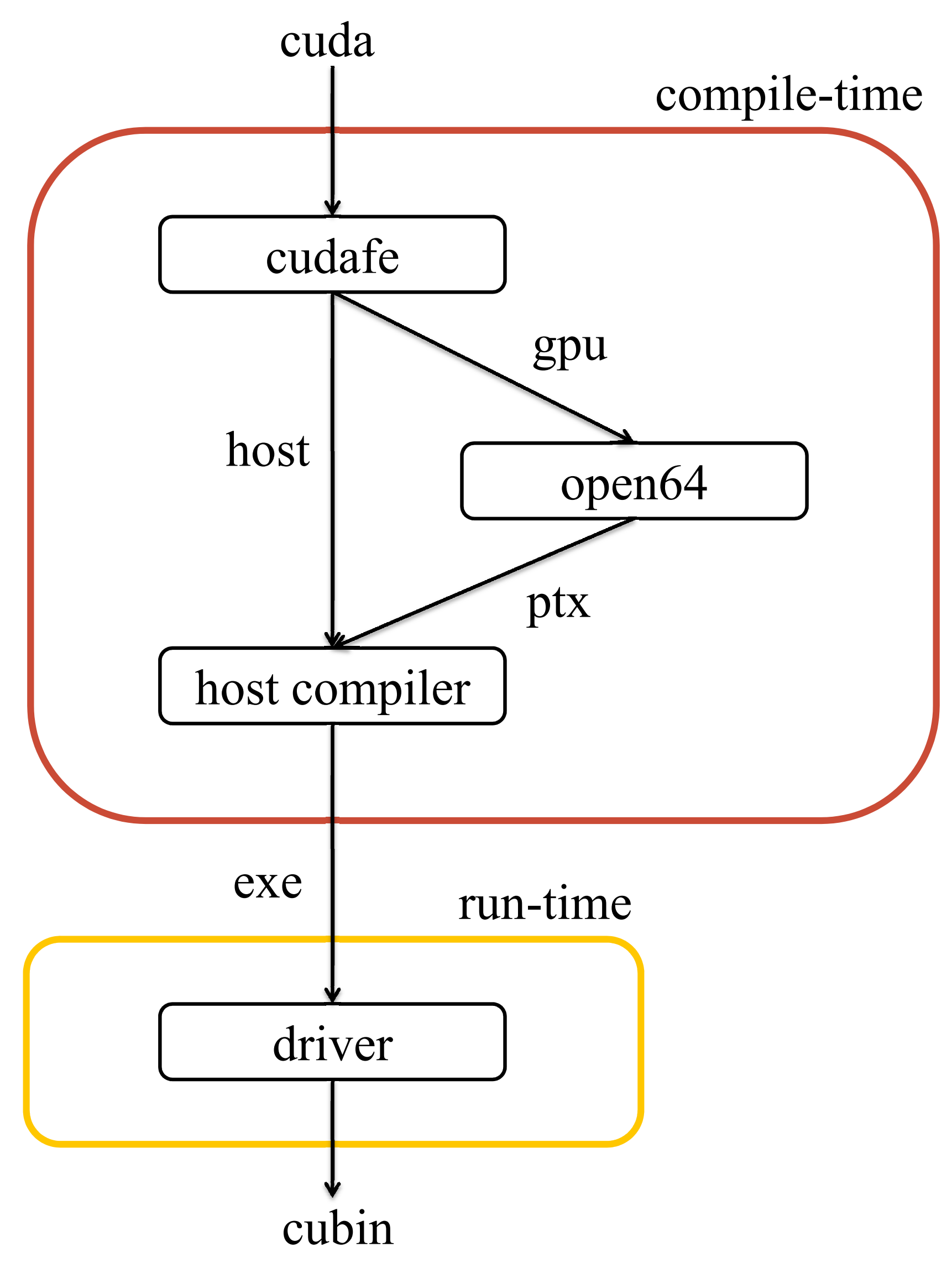
Mike Murphy
NVIDIA Corporation
mmurphy@nvidia.com

Google SUMMER OF CODE 2009
code.google.com/soc

Contribution

- We explore the correlation between CUBIN register pressure and PTX register pressure.
- We evaluate two different optimizations that attempt to reduce register pressure and increase hardware occupancy.

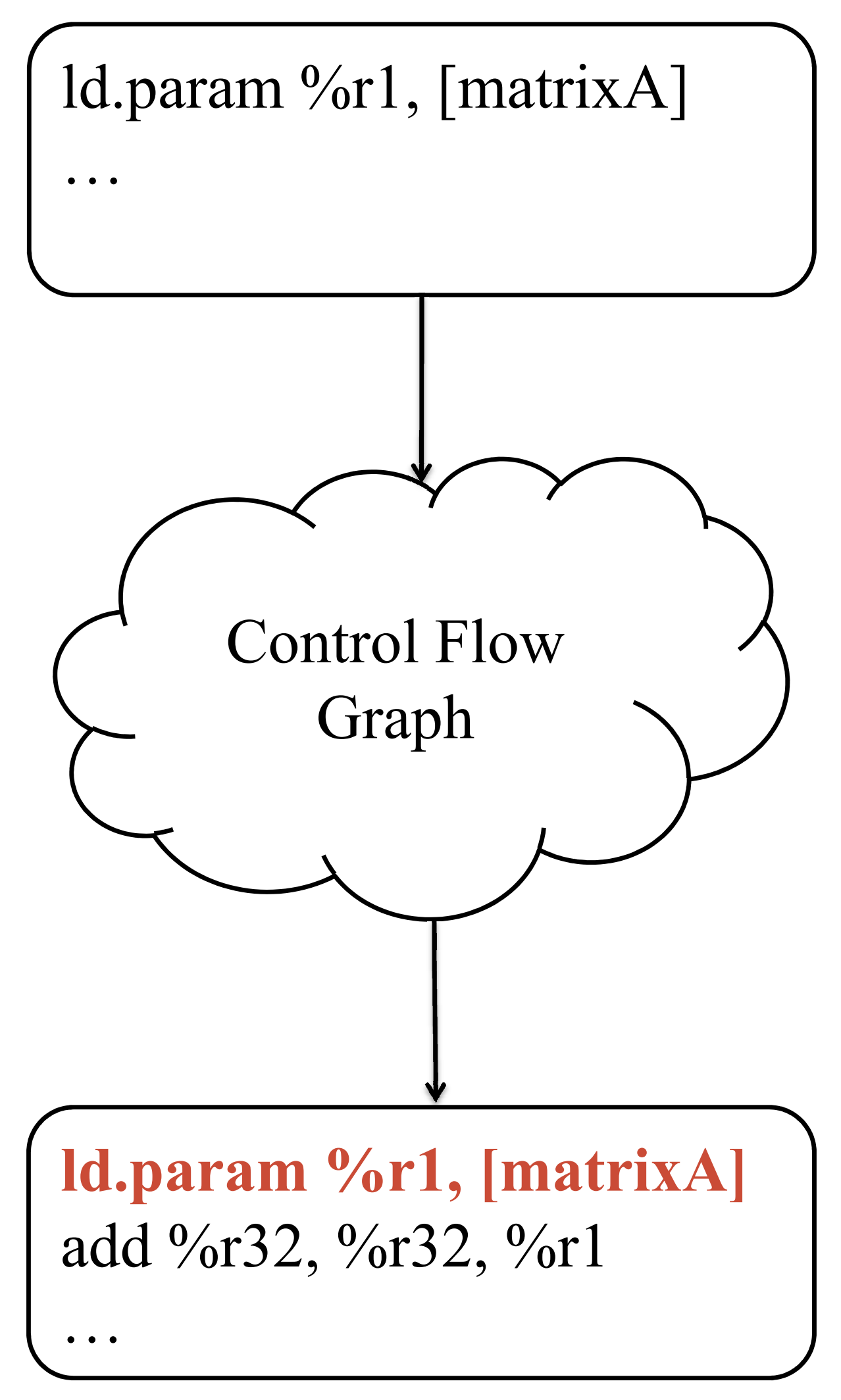
Building Process



The process of compiling a CUDA program into an executable is the following:

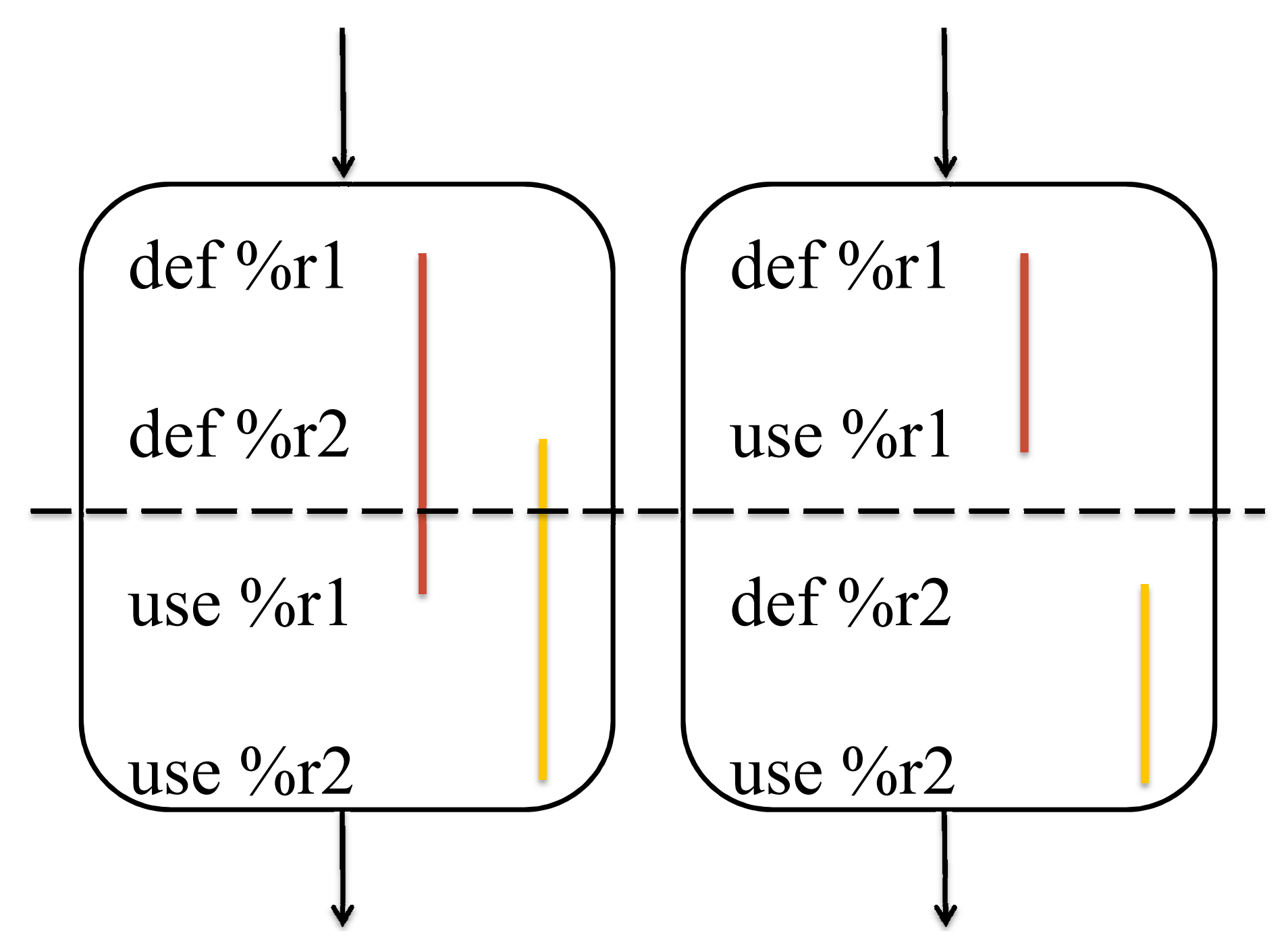
- At compile-time, the CUDA front-end (cudafe) splits the program into the host code and the GPU code.
- The GPU code is compiled by Open64 into an intermediate language called Parallel Thread Execution (PTX).
- The PTX code is included by the host code as a device descriptor.
- The host code (including the descriptor) is compiled and linked with the CUDA libraries into an executable file.
- At run-time, the executable calls the driver and passes it the device descriptor. The driver compiles the PTX code in the descriptor into a CUDA binary (CUBIN).

Register Rematerialization



- We rematerialize memory operations for loading parameters and special registers (e.g. threadId, blockId).

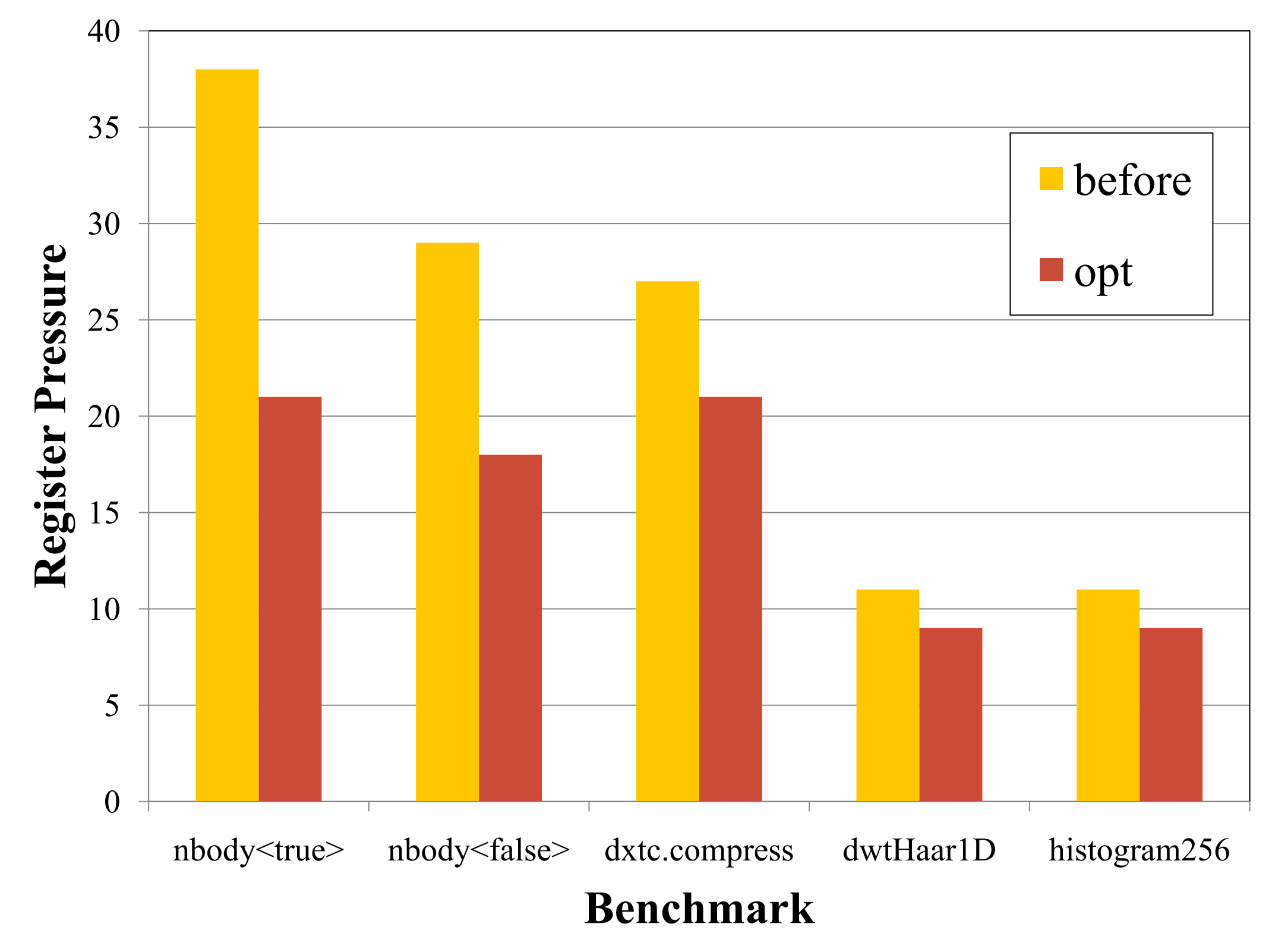
Instruction Scheduling

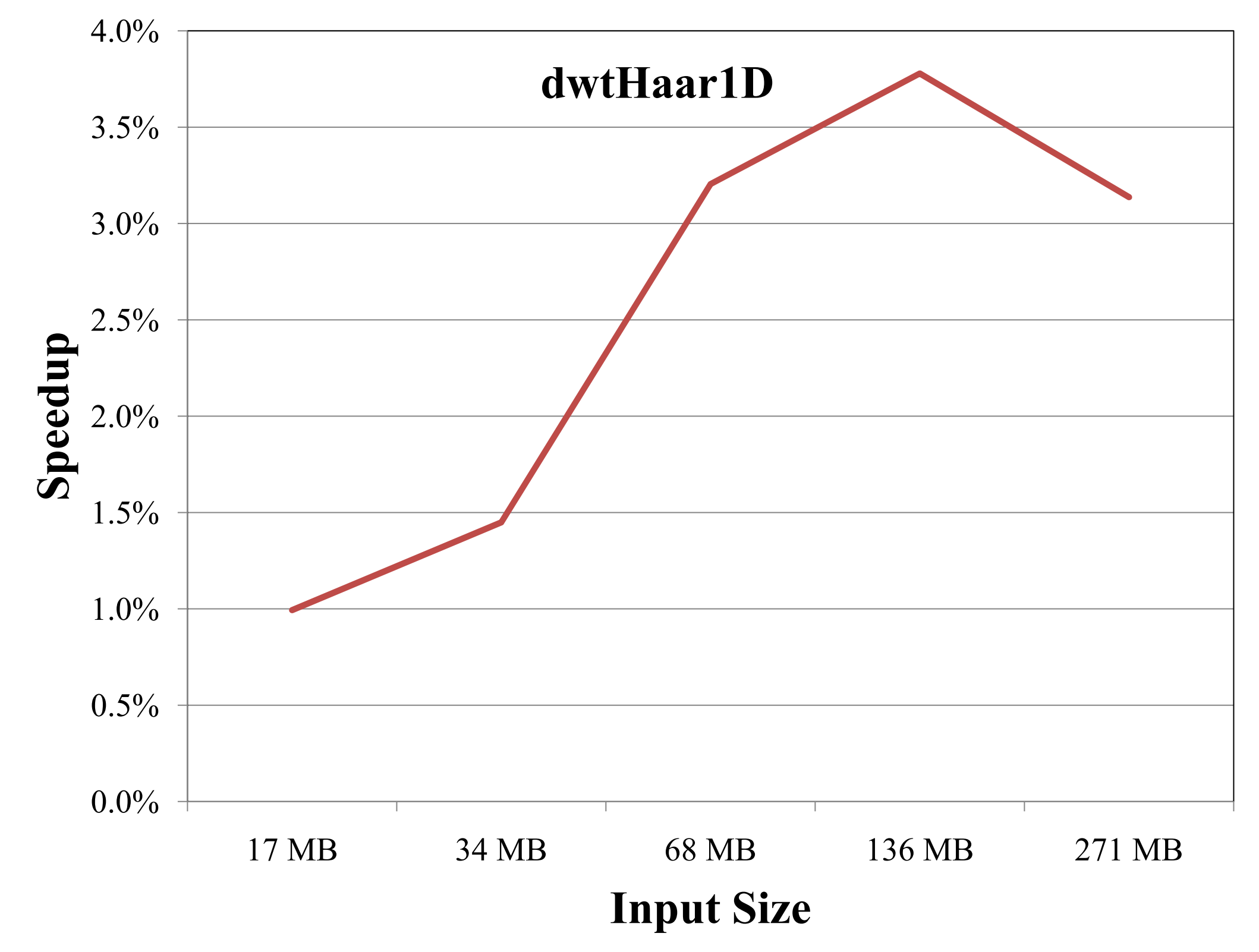


- We implemented a backward list scheduler that reorders instructions within a basic block.
- We start by doing a liveness analysis and building a data dependence graph.
- The scheduler iterates through the ready list, evaluating a cost function that consider the number of live registers and the use-defs associated with each instruction.

Results

We evaluated the optimizations on an NVIDIA Tesla C870 GPU and a 64-bit Intel Xeon @ 1.6GHz CPU with 4MB of L2 cache running Linux 2.6 (Fedora Core 10). We used the applications included in the NVIDIA SDK 2.2. The speedup results were collected using the CUDA Profiler. We compiled 52 benchmarks, containing a total of 748 kernels. We found that the correlation coefficient between PTX register pressure and CUBIN register pressure is 0.94 for the selected applications. In our experiments, 294 (39%) kernels showed less register pressure, 364 (49%) experienced no change, and only 90 (12%) had more register pressure due to our optimizations. For the Haar Wavelet Decomposition benchmark we achieved a 4% speedup.





Future Work

- Evaluate a selective rematerialization algorithm that targets the basic blocks with higher register pressure.
- Consider a bigger set of applications from different fields and with different execution times and kernel sizes.
- Measure the performance impact of the optimizations on the benchmarks that showed negative improvements (higher register pressure).
- Quantify side effects like additional number of dynamic instructions, uncoalesced memory operations, etc.

References

[1] M. Murphy. NVIDIA's Experience with Open64. In *Open64 Workshop at CGO*, April 2008.

[2] K. Wilken, J. Liu, and M. Heffernan. Optimal Instruction Scheduling Using Integer Programming. In *PLDI*, May 2000.

[3] I. Baev, R. Hank, and D. Gross. Prematerialization: Reducing Register Pressure for Free. In *PACT*, 2006.

[4] C. M. Chang, C. M. Chen, and C. T. King. Using Integer Linear Programming for Instruction Scheduling and Register Allocation in Multi-issue Processors. In *Computer and Mathematics with Applications*, 1997.

[5] A. Aleta, J. M. Codina, A. Gonzalez, and D. Kaeli. Demystifying on-the-fly spill code. In *PLDI*, June 2005.

Acknowledgments

This work was supported by Google through the Google Summer of Code program. We would also like to thank NVIDIA for their generous hardware donations to Northeastern University that included the Tesla GPU systems used in this work.