Towards a Performance-Portable FFT Library for Heterogeneous Processors

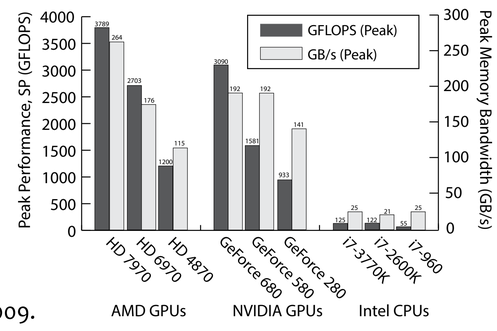
Student: Carlo del Mundo <cdel@vt.edu> (ECE)
Advisor: Dr. Wu-chun Feng <feng@cs.vt.edu> (ECE, CS)

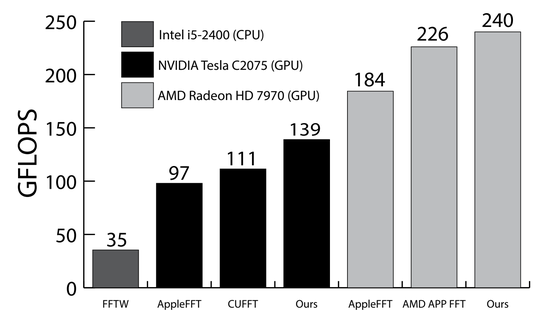
Goal: *Accelerate FFT across a set of heterogeneous processors*

Motivation

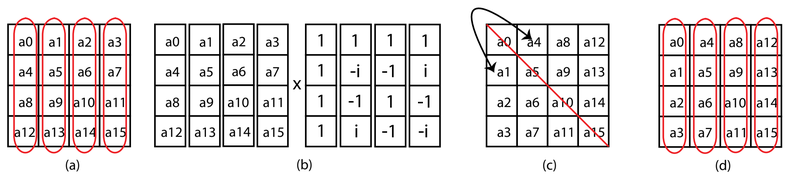
- FFT is identified as a key computational idiom for scientific computing [1]. To date, FFTW is arguably the most popular FFT library, but written only for CPUs.
- Heterogeneous processors such as the graphics processing unit (GPU) provide a rich set of computational resources. The following figures compare performance across CPUs and GPUs in (1) peak compute and peak bandwidth and (2) against popular FFT libraries for N = 16, respectively. GPUs have significantly higher computational throughput than CPUs outperforming FFTW-based codes by factors as high as 6.8!

1. K. Asanovic et al., "A View of the Parallel Computing Landscape," Communications of the ACM, Oct. 2009.





Background



Fast Fourier Transform (FFT)

- A data parallel scientific algorithm belonging to a family of computations known as spectral methods
- The figure above depicts the Cooley-Tukey method applied to a 16-pt FFT. Input is organized in row-major order. In (a), independent sub-FFTs are calculated per column. The twiddle multiplication stage is shown in (b) where a point-wise multiplication is performed on each element of the input array. Matrix transpose occurs in (c) where elements across the diagonals are exchanged. Finally, independent sub-FFTs are calculated again in (d).

Graphics Processors

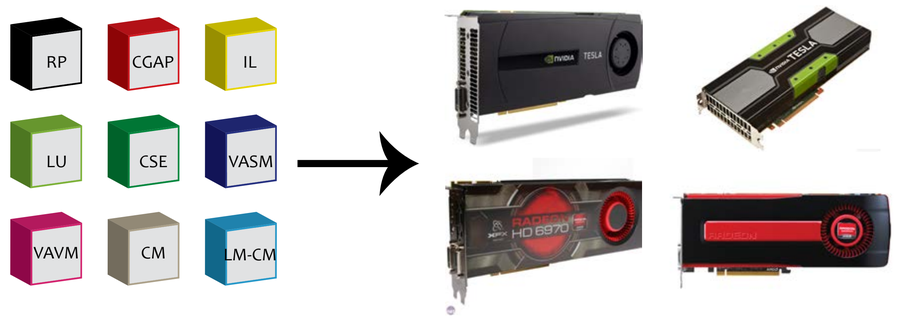
- Massively parallel architectures suitable for data-parallel algorithms
- Achieving high-performance on GPUs requires exploration of an optimization space. The optimization process is complex and rarely behaves in an isolated manner. This problem is further exacerbated by architectural differences across vendors and GPU generations.

Approach

Apply optimizations to graphics processors *in isolation* and *in concert* to characterize machine-level behavior.

Application & Experimental Testbed

- 1D FFT for N = 16, 64, 256
- AMD Radeon HD 6970, AMD Radeon 7970
  NVIDIA Tesla C2075, NVIDIA Tesla K20c



Results

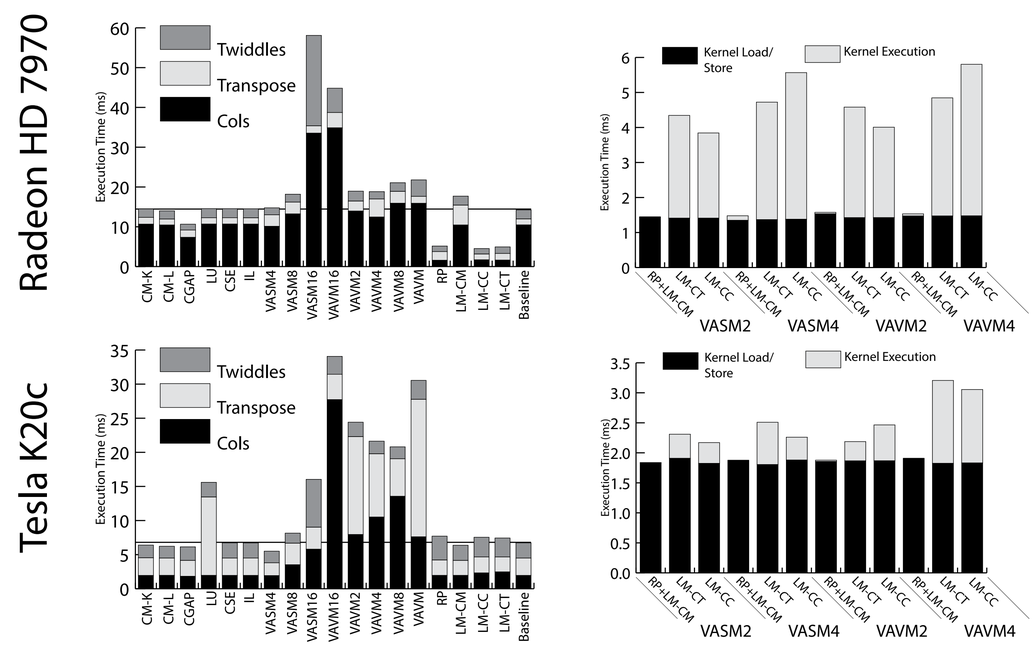
For brevity, we depict our results for N = 16 on AMD Radeon HD 7970 and NVIDIA Tesla K20c only.

In Isolation

- On-chip memory optimizations (RP, LM-CC, LM-CT) highly effective for Radeon HD 7970, but not so much for K20c.

In Concert

- RP + LM-CM combined with 8- or 16-byte vector access and scalar arithmetic (VASM) and constant memory (CM-K) effective for all architectures



Insights & Conclusions

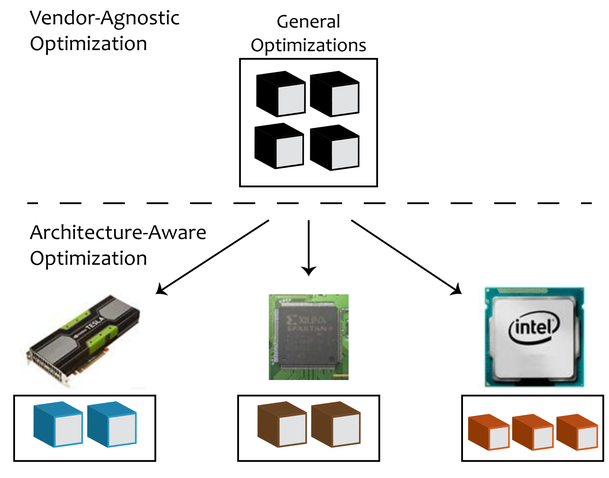
Despite radical architectural differences across GPU vendors and generations, we've identified a set of optimizations applicable for FFT for AMD and NVIDIA GPU architectures. These optimizations are as follows.

- RP (Register Preloading) - All data elements are first preloaded onto the register file of the respective GPU. Computation is facilitated solely on registers.
- CGAP (Coalesced Global Access Pattern) - Threads access memory contiguously (the kth thread accesses memory element k)
- VASM2/4 (Vector Access, Scalar Math, float{2/4}) - Data elements are loaded as the listed vector type. Arithmetic operations are scalar (float x float).
- LM-CM (Local Memory, Communication Only) - Data elements are loaded into local memory only for communication. Threads swap data elements solely in local memory.
- CM-K (Constant Memory - Kernel Argument) - The twiddle multiplication stage of FFT is precomputed on the CPU and stored in GPU constant memory for fast look up.

Furthermore, performance of 1D FFTs on graphics processors is limited by global memory data transfer comprising up to 99% of the execution time on our optimized implementations. Our optimized implementations deliver speed-ups as high as 31.5 over a baseline GPU implementation, and 9.1 over a multi-threaded FFTW CPU implementation with AVX vector extensions.

Future Work

- Adapt work to other parallel processors, e.g. multi-core CPUs, FPGAs, and the Intel Many Integrated Core (MIC). Identify vendor-agnostic and vendor-aware optimizations and apply in the context of an optimized, multi-platform FFT library for heterogeneous computing.



Publications

C. del Mundo and W. Feng, "Towards a Performance-Portable FFT Library for Heterogeneous Computing," IEEE International Symposium on Workload Characterization (IISWC), Portland, Oregon, USA, Sept. 2013. (Submitted April 2013.)

C. del Mundo, V. Adhinarayanan, and W. Feng, "Accelerating Fast Fourier Transform for Wideband Channelization," IEEE International Conference on Communications (ICC), Budapest, Hungary, June 2013. (Accepted for publication in January 2013.)