L9: CUDA-CHiLL Research and Introduction to Dense Linear Algebra

Administrative

- Assignment
  - Due Monday, Feb. 11, 5PM
  - Use handin program on CADE machines
    - "handin CS6235 lab2 <probfile>"
- Why the extension?
  - I am going to be providing execution times for my own implementation + using compiler support
  - Can you do better???

Outline

- Plan to build some educational material around research software
- Today's lecture will provide an overview of the compiler, and introduce linear algebra techniques
- References:

Malik Khan, Protonu Basu, Gabe Rudy, Mary Hall, Chun Chen, and Jacqueline Chame. 2013. A script-based autotuning compiler system to generate high-performance CUDA code. *ACM Trans. Archit. Code Optim. 9, 4, Article 31 (January 2013), 25 pages.*

Volkov, V., and Demmel, J. W. 2008. Benchmarking GPUs to tune dense linear algebra, *SC08*, November 2008.

Paper link: http://portal.acm.org/citation.cfm?id=1413402

Talk link: http://www.eecs.berkeley.edu/~volkov/volkov08-sc08talk.pdf

Volkov code: http://forums.nvidia.com/index.php?showtopic=47689&st=40&p=314014&#entry314014

HiPEAC Slides

Recall MM Code (from text, p. 87)

```
__global__ void MatrixMulKernel (float *Md, float *Nd, float *Pd, int Width) {
1.    __shared__ float Mds [TILE_WIDTH] [TILE_WIDTH];
2.    __shared__ float Nds [TILE_WIDTH] [TILE_WIDTH];
3 & 4.    int bx = blockIdx.x; int by = blockIdx.y;  int tx = threadIdx.x; int ty = threadIdx.y;
//Identify the row and column of the Pd element to work on
5 & 6.    int Row = by * TILE_WIDTH + ty;    int Col = bx * TILE_WIDTH + tx;
7.        float Pvalue = 0;
// Loop over the Md and Nd tiles required to compute the Pd element
8.        for (int m=0; m < Width / TILE_WIDTH; ++m) {
// Collaborative (parallel) loading of Md and Nd tiles into shared memory
9.          Mds [ty] [tx] = Md [Row*Width + (m*TILE_WIDTH + tx)];
10.         Nds [ty] [tx] = Nd [(m*TILE_WIDTH + ty)*Width + Col];
11.         __syncthreads();                    // make sure all threads have completed copy before calculation
12.         for (int k = 0; k < TILE_WIDTH; ++k)    // Update Pvalue for TKxTK tiles in Mds and Nds
13.           Pvalue += Mds [ty] [k] * Nds [k] [tx];
14.         __syncthreads();                    // make sure calculation complete before copying next tile
          } // m loop
15.       Pd [Row*Width + Col] = Pvalue;
}
```

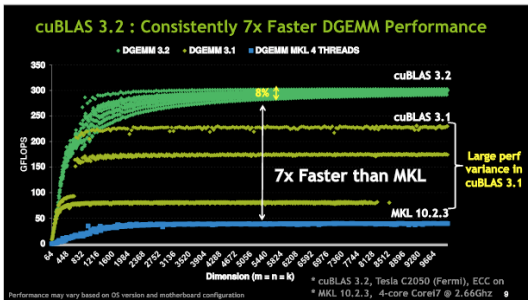

Preview: SGEMM (CUBLAS 2.x/3.x) on GPUs

- SGEMM result is not from algorithm of L5
- Why? Significant reuse can be managed within registers

	GPU Rule-of-Thumb	Lecture (for GTX 280)	Lecture (for Fermi C2050
Threading	Generate lots of threads (up to 512/ block) to hide memory latency	Only 64 threads/block provides 2 warps, sufficient to hide latency plus conserves registers	More cores/block, fewer registers/thread, so use 96 threads/block
Shared memory	Use to exploit reuse across threads	Communicate shared data across threads and coalesce global data	Communicate shared data across threads and coalesce global data
Registers	Use for temporary per-thread data	Exploit significant reuse within a thread	Exploit significant reuse within a thread
Texture memory	Not used	Not used	Increase bandwidth for global memory through parallel accesses

Volkov Slides 5-17, 24

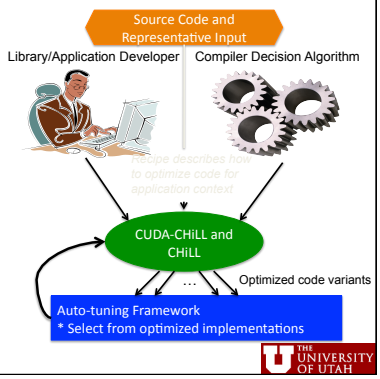
Comparison with MKL (Intel)



Slide source: http://www.scribd.com/doc/47501296/CUDA-3-2-Math-Libraries-Performance

CHiLL and CUDA-CHiLL: Compiler-Based Auto-Tuning Technology

- Increase compiler effectiveness through ***auto-tuning*** and ***specialization***
- Provide high-level interface to code generation (***recipe***) for library or application developer to suggest optimization
- Bridge domain decomposition and single-socket locality and parallelism optimization
- Auto-tuning for different optimization goals: performance, energy, reliability



THE UNIVERSITY OF UTAH