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

CS6235

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:
  Paper link: http://portal.acm.org/citation.cfm?id=1413403
  Talk link: http://www.eecs.berkeley.edu/~volkov/volkovOB-sc08talk.pdf

HiPEAC Slides

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???
Recall MM Code (from text, p. 87)

```c
__global__ void MatrixMulKernel(float *Md, float *Nd, float *Pd, int Width) {
  __shared__ float Mds[TILE_WIDTH][TILE_WIDTH];
  __shared__ float Nds[TILE_WIDTH][TILE_WIDTH];
  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
  int Row = by * TILE_WIDTH + ty;
  int Col = bx * TILE_WIDTH + tx;
  float Pvalue = 0;
  // Loop over the Md and Nd tiles required to compute the Pd element
  for (int m = 0; m < Width / TILE_WIDTH; ++m) {
    // Collaborative (parallel) loading of Md and Nd tiles into shared memory
    Mds[ty][tx] = Md[Row*Width + (m*TILE_WIDTH + tx)];
    Nds[ty][tx] = Nd[(m*TILE_WIDTH + ty)*Width + Col];
    __syncthreads();
    // Make sure all threads have completed copy before calculation
    for (int k = 0; k < TILE_WIDTH; ++k)
      Pvalue += Mds[ty][k] * Nds[k][tx];
    __syncthreads();
    // Make sure calculation complete before copying next tile
  }
  // m loop
  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

<table>
<thead>
<tr>
<th>GPU Rule of Thumb</th>
<th>Lecture (for GTX 280)</th>
<th>Lecture (for Fermi C2050)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threading</td>
<td>Generate lots of threads (up to 512/block) to hide memory latency</td>
<td>Only 64 threads/block provides 2 warps, sufficient to hide latency plus conserves registers</td>
</tr>
<tr>
<td></td>
<td>For GPU Threading, no need to manage significant reuse within threads</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For Fermi, more cores/block, fewer registers/thread, so use 96 threads/block</td>
<td></td>
</tr>
<tr>
<td>Shared memory</td>
<td>Use to exploit reuse across threads</td>
<td>Communicate shared data across threads and coalesce global data</td>
</tr>
<tr>
<td></td>
<td>For GPU, use shared memory to coalesce global data</td>
<td></td>
</tr>
<tr>
<td>Registers</td>
<td>Use for temporary per-thread data</td>
<td>Exploit significant reuse within a thread</td>
</tr>
<tr>
<td></td>
<td>For Fermi, increase bandwidth for global memory through parallel accesses</td>
<td></td>
</tr>
<tr>
<td>Texture memory</td>
<td>Not used</td>
<td>Not used</td>
</tr>
</tbody>
</table>

Comparison with MKL (Intel)

- MKL does not provide a GPU version of SGEMM
- MKL uses OpenMP for multi-threading on CPUs
- MKL uses AVX instructions for optimization

Volkov Slides 5-17, 24

Comparison with MKL (Intel)

- MKL does not provide a GPU version of SGEMM
- MKL uses OpenMP for multi-threading on CPUs
- MKL uses AVX instructions for optimization

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

Source Code and Representative Input

Library/Application Developer → Compiler Decision Algorithm

Auto-tuning Framework

* Select from optimized implementations

Optimized code variants

CUDA-CHiLL and CHiLL

Recipe describes how to optimize code for application context

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