Performance of Sparse Matrix-Multiple Vectors Multiplication on Multicore and GPUs

Walid Abu-Sufah, University of Illinois at Urbana-Champaign; Khalid Ahmad, University of Jordan

Introduction

We implemented a heuristics-based auto-tuning framework for sparse matrix-vector multiplication (SpMV) on GPUs. For a given sparse matrix, our auto-tuner delivers the highest performing SpMV kernel which combines the use of the most efficient storage format and tuned parameters of the corresponding CUDA code targeting the underlying GPU architecture.

Different storage schemes/architectures perform best for matrices with different sparsity patterns. So far, our auto-tuner considers the storage formats: Diagonal (DIA), ELLPACK (ELL), Blocked ELLPACK, and our Blocked Transpose Jagged Diagonal Storage format (BTJAD). Other formats are currently being considered including SELL-C-σ and SEL-P.

When nonzero values are restricted to a small number of diagonals, DIA-SpMV is the best performing kernel. For matrices with uniform row lengths, ELL-SpMV is the best performing kernel. Currently, we are integrating sparse matrix-multiple vector multiplication (SpMM) kernels in our auto-tuner. So far, we have implemented DIA-SpMM for structured matrices and ELL-SpMM for uniform row length matrices.

The Platforms and Matrices

We evaluated kernels on nodes of two HPC clusters: (1) one node of NVIDIA benchmarking PSG cluster using a Kepler K40m GPU with a dual socket Intel Xeon E5-2690 v2 @ 3.00GHz CPU, and (2) one node of the Cyprus Institute Cy-Tera cluster using a Fermi P100 GPU with a dual socket Intel Xeon E5-2690 v2 @ 2.61GHz CPU.

We used 1 structured matrices that represent common stencil operations on regular grids (Table 1). They are formed by applying a Laplacian stencil operator to every point in an N-dimensional space. The results in nonzero elements are restricted to a small number of diagonals, where the number of diagonals is the number of points in the stencil. From the University of Florida collection we used 28 structured matrices (Table 2) and 29 uniform row length matrices (Table 3). Performance is evaluated in terms of GFLOP/s, which is computed by dividing the number of arithmetic operations by the average running time of 1000 runs. Our time measurements do not include the time required to transfer data between the host and the GPU. Our experiments test our SpMM implementations for multiplying a sparse matrix by up to 512 vectors.

Structured Matrices

Uniform Row Length Matrices

Acknowledgement

We thank Ahmed Sameh of Purdue University for his comments. We thank NVIDIA Corporation for using the PSG benchmarking cluster. This work was supported in part by the Scientific Research Support Fund of Jordan under grant number IT/2013-2 and the USACISPSP project, funded by the European Commission under the 7th Framework Programme through Capacities Research Infrastructure. PRA-2011-1.2 Virtual Research Communities Combination of Collaborative Project and Coordination and Support Actions (CP-CSI) under grant agreement RI-314140. This work also used the Stampede cluster of the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number OCI-1053575.

K40m GPU Results

Structured Matrices

Our DIA SpMM: 2.4x faster than CUSP DIA-SpMV

Our ELL SpMM: 2.8x faster than CUSP ELL-SpMM

Uniform Row Length Matrices

Our DIA SpMM: 5.2x faster than cuSPARSE CSR-SpMM

Our ELL SpMM: 3.9x faster than cuSPARSE CSR-SpMM

K40m GPU & Intel CPUs Results

Structured Laplace Matrices

Our DIA SpMM: 7.2x faster than the best performing Intel MKL CSR-SpMM on dual socket Intel Ivy Bridge 10-core

Our ELL SpMM: 22.2x faster than the best performing Intel MKL CSR-SpMM on dual socket Intel Westmere hexa-core

Structured Matrices

Uniform Row Length Matrices

Structured Laplace Matrices

LAPLACIAN OPERATORS DISCRETIZED AS K-POINT FINITE DIFFERENCE STENCILS ON REGULAR GRIDS

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Grid</th>
<th>Diagonal</th>
<th>Nonzeros/Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Laplace 3pt</td>
<td>2</td>
<td>3</td>
<td>2,999,997</td>
</tr>
<tr>
<td>Laplace 5pt</td>
<td>2</td>
<td>5</td>
<td>6,626,669</td>
</tr>
<tr>
<td>Laplace 7pt</td>
<td>2</td>
<td>7</td>
<td>8,986,005</td>
</tr>
<tr>
<td>Laplace 9pt</td>
<td>2</td>
<td>9</td>
<td>26,178,647</td>
</tr>
</tbody>
</table>

TABLE I

28 UNIVERSITY OF FLORIDA STRUCTURED MATRICES

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Diagonals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum</td>
<td>10,319,760</td>
</tr>
<tr>
<td>Minimum</td>
<td>19,996</td>
</tr>
<tr>
<td>Average</td>
<td>1,537,409</td>
</tr>
</tbody>
</table>

TABLE II

29 UNIVERSITY OF FLORIDA UNIFORM ROW LENGTH MATRICES

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Average nonzeros per row</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum</td>
<td>7,791,168</td>
</tr>
<tr>
<td>Minimum</td>
<td>20,360</td>
</tr>
<tr>
<td>Average</td>
<td>1,339,107</td>
</tr>
</tbody>
</table>

TABLE III

- Results of single precision matrices
- Our DIA-SpMV, ELL-SpMV, NVIDIA cuSPARSE CSR-SpMM, DIA-SpMM, and ELL-SpMM on NVIDIA K40m GPU; GCC 4.8.1 input and output data on device
- Structured matrices obtained from: https://sites.google.com/site/sparseautotuner/matrices/laplace
- http://www.cise.ufl.edu/research/sparse/matrices/

Performance may vary based on OS version and motherboard configuration.