Out-of-core algorithms for dense matrix factorization on GPGPU

Ed D’Azevedo
Oak Ridge National Laboratory
davezvedo@ornl.gov

Amy Huang
Chinese University of Hong Kong
ah081135@cuhk.edu.hk

Kwai Wong
University of Tennessee
kwong@utk.edu

Watson Wu
Chinese University of Hong Kong
watsonwu@cuhk.edu.hk

Background

• Solution of large dense matrix problems arises from diverse applications such as modelling the response of heating of fusion plasma to radio frequency (RF) wave, modelling radiation heat transfer, boundary element method, and large-scale least squares problem.

• General Purpose Graphics Processing Unit (GPGPU) offers performance several times faster than multi-core CPU. GPGPU has dedicated memory to provide very high memory bandwidth. Even inexpensive (less than $200) consumer-grade video cards contain Nvidia Fermi GPGPU processor with 1 GBytes of device memory. However, the PCI connection can be bottleneck in transferring data between GPGPU device and CPU host.

• The MAGMA Library [2] achieves very high performance on GPGPU for dense matrix computations. However, the largest problem size is limited to the amount of local device memory on GPGPU.

• Idea: Adapt out-of-core algorithms for solving large problems on GPGPU so that a matrix of size say 10 GBytes can still be factored on GPGPU with only 1 GBytes of device memory.

Cholesky factorization

Organise matrix as wide column panels that still fit on GPGPU (see Figure).

Use left-looking out-of-core algorithm to minimise the amount of data transfer [1]. Cholesky factorisation access only lower triangular part

1. Cholesky factorization of diagonal block (magma_dpoftr_gpu)

\[
L_1L_2 = A_{12}
\]

2. Update column panel (cublasDstrm)

\[
L_21 = A_{21}
\]

3. Symmetric update of unfactored submatrix (cublasDseyr)

\[
A_{12} \leftarrow A_{12} - L_1L_2
\]

4. Cholesky factorization of updated submatrix

\[
L_2L_2 = \tilde{A}_{12}
\]

• Use right-looking algorithm MAGMA magma_dpoftr for factorization of diagonal block of panel Y in device memory of GPGPU

• As the factorization proceeds, widths of panels X and Y are increased to use available device memory.

LU factorization

• Similar to Cholesky factorization but access to full matrix using fixed size panels

• If panel Y has width \( [N/K] \), then need extra \( O(KN^2) \) transfers. Therefore panel Y takes up most available memory to be as wide as possible.

• MAGMA magma_dgetrf designed for nearly square system and need max\([M,N] \)\(^2 \) amount of GPGPU memory.

• Hybrid factorization algorithm: LU factorization of narrow column panel performed on multi-core CPU host using LAPACK DGETRF, right-looking update performed by GPGPU using CUBLAS

The out-of-core algorithms were tested using only 1 GB (out of 5 GB) on purpose to highlight the message that substantial performance is achieved even with only a small amount of device memory:

### Results

<table>
<thead>
<tr>
<th>Environment</th>
<th>MKL (12 CPU)</th>
<th>MAGMA 1.0</th>
<th>Out-of-core algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>N=25,000</td>
<td>127 Gflops/s</td>
<td>27 T Gflops/s</td>
<td>200 Gflops/s</td>
</tr>
<tr>
<td>N=35,000</td>
<td>123 Gflops/s</td>
<td>27 T Gflops/s</td>
<td>214 Gflops/s</td>
</tr>
</tbody>
</table>

Table 2: Comparison of MAGMA 1.0 to out-of-core LU factorization (DGETRF) using only 1 GBytes out of 5 GBytes of Nvidia M2070.

<table>
<thead>
<tr>
<th>Environment</th>
<th>MKL (12 CPU)</th>
<th>MAGMA 1.0</th>
<th>Out-of-core algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>N=25,000</td>
<td>122 Gflops/s</td>
<td>266 Gflops/s</td>
<td>246 Gflops/s</td>
</tr>
<tr>
<td>N=35,000</td>
<td>123 Gflops/s</td>
<td>266 Gflops/s</td>
<td>263 Gflops/s</td>
</tr>
</tbody>
</table>

Table 3: Comparison of MAGMA 1.0 to out-of-core Cholesky factorization (DPOTRF) using only 1 GBytes out of 5 GBytes of Nvidia M2070.

References


Acknowledgements

The submitted manuscript has been authored by a contractor of the U.S. Government under Contract No. DE-AC05-00OR22725. Accordingly, the U.S. Government retains a non-exclusive, royalty-free license to publish or re-print the published form of this contribution, or allow others to do so, for U.S. Government purposes. The NSF Kennesaw Project (National Science Foundation award OCI-0915715) provided computing resources for this project. Amy Huang and Watson Wu were supported by a fellowship from the Department of Mathematics, the Chinese University of Hong Kong.