How Fast Direct Solvers Can Benefit from GPU-acceleration

Mirko Myllykoski
University of Jyväskylä, Finland — Umeå University, Sweden
mirko.myllykoski@jyu.fi, umu.se

Introduction

This poster highlights the primary findings of my PhD thesis [2] that was published and publicly defended at University of Jyväskylä, Finland in 2015. The three main themes of the thesis are: block cyclic reduction type fast direct solvers, GPU computing and image processing. This poster focuses on the first two themes.

The fast direct solvers are a group of specialized numerical methods that usually have arithmetical complexities of the order \( O(N \log N) \). The thesis focuses on symmetric block tridiagonal linear systems that can be presented in a separable form using Kronecker matrix tensor products. In that case, the so-called PSCR method [1, 5] can be used. Suitable linear systems arise, for example, when a Poisson or a Helmholtz equation is discretized using linear or bilinear finite-elements.

PSCR method

Let \( K \) be either \( \mathbb{R} \) or \( \mathbb{C} \). The Kronecker matrix tensor product is defined for matrices \( B \in \mathbb{K}^{m \times n} \) and \( C \in \mathbb{K}^{p \times q} \) as:

\[
B \otimes C = \begin{bmatrix}
  b_{11}C & b_{12}C & \cdots & b_{1n}C \\
  b_{21}C & b_{22}C & \cdots & b_{2n}C \\
  \vdots & \vdots & \ddots & \vdots \\
  b_{m1}C & b_{m2}C & \cdots & b_{mn}C
\end{bmatrix} \in \mathbb{K}^{mn \times pq}.
\]

Let us focus on a separable linear system \( Au = f \) with:

\[
A = A_1 \otimes M_1 + M_1 \otimes A_2,
\]

where the factor matrices \( A_1, M_1 \in \mathbb{K}^{n \times n} \) and \( A_2, M_2 \in \mathbb{K}^{m \times m} \) are symmetric and tridiagonal. The PSCR method can be described using a set of orthogonal projection matrices:

\[
P^{(i)} = P^{(i)} \otimes I, \quad i = 1, \ldots, k,
\]

with the property \( \text{Im}(P^{(i)}) \subset \cdots \subset \text{Im}(P^{(k)}) \). The solution process is recursive and each recursion step involves the solution of a projected system of the form:

\[
P^{(i)} A P^{(i)} v = P^{(i)} g,
\]

where only a very sparse set of components of \( v \) is actually required and \( g \) has only a few non-zero elements. Each projected system decouples to a set of independent sub-systems and each sub-system can be block diagonalized. That is, the inverse of the coefficient matrix in each sub-system can be decomposed as:

\[
(RW \otimes I)(A \otimes M_1 + M_1 \otimes A_2)^{-1}(W^T Q \otimes I).
\]

The orthogonal projection matrices \( R \in \mathbb{K}^{m \times m} \) and \( Q \in \mathbb{K}^{n \times n} \) define the required solution components and the non-zero components of the right-hand side vector, respectively.

The diagonal matrix \( A \) and the matrix \( W \) fulfill the conditions:

\[
W^T A_i W = A_i \quad \text{and} \quad W^T M_i W = M_i,
\]

where \( A_i \in \mathbb{K}^{n \times n} \) and \( M_i \in \mathbb{K}^{m \times m} \) are non-zero diagonal blocks from projected factor matrices \( P^{(i)} A P^{(i)} \) and \( P^{(i)} M P^{(i)} \), respectively.

Speedup

Initial experiments with simplified PSCR solvers indicated up to sixfold speedups when a desktop- and a GPU-level Nvidia GPU was compared against a quad-core desktop-level Intel CPU [3]. The implementation was later generalized in [4]. The generalized implementation can be applied to real and complex valued systems with arbitrary \( n_1 \) and \( n_2 \). The performance of the generalized implementation was evaluated by solving a set of Poisson and Helmholtz equations on a Nvidia K40c GPU and comparing the run times against a single-threaded CPU implementation:

The horizontal axes show the operational intensity [Flop/Byte] and the vertical axes show the obtained floating-point performance [GFlop/s]. The models show that the implementation is inherently memory-bound on contemporary GPUs. However, there is still room for improvement.

Conclusions

- GPUs can provide significant performance improvements in the context of the block cyclic reduction methods.
- However, the generalized implementation did not perform quite as well as expected on newer GPUs.
- In particular, the tridiagonal solver did not perform well on newer Nvidia GPUs and replacing it with a more state-of-the-art approach would probably improve the performance.

References


Acknowledgements

The research was supported by the Academy of Finland (grant #252549), the Jyväskylä Doctoral Program in Computing and Mathematical Sciences (COMAS), and the Foundation of Nokia Corporation. Partial support has also been received from the EU Horizon 2020 project NLA4PET under grant agreement No. 671633. The author would like to extend his gratitude to his supervisors Prof. Timo Rossi and Prof. Jari Toivanen. The author would also like to thank his other collaborators, Prof. Roland Goseïinski and Prof. Tommi Karkkolainen.