Porting an Asteroseismology Code on GPU thanks to MAGMA library

Alejandro Estaña, Daniel Reese, François Ligneres & Jérôme Ballot
Objective:
Accelerate TOP (Two-dimensional Oscillation Program)
Objective:
Accelerate TOP (Two-dimensional Oscillation Program)

\[
\begin{align*}
\lambda \rho & = -\ddot{v} \cdot \nabla \rho + \rho \nabla \cdot \ddot{v}, \\
\lambda \rho \dot{v} & = -\nabla p + \rho \ddot{g}_o - \rho \nabla \Psi - 2\Omega \vec{e}_z \times \rho \dot{v}, \\
\lambda p - \lambda c_o^2 \rho & = \frac{\rho_o N_e^2 c_o^2 \dot{v} \cdot g_o}{\|g_o\|^2}, \\
0 & = \Delta \Psi - \rho.
\end{align*}
\]

Pulsation equations

Differential equations expressed in the form of a generalized eigenvalue problem

\[
Av = \lambda Bv
\]

\[
(A - \sigma B)^{-1} Bw = \mu w
\]

Using \textbf{LAPACK} for linear algebra computations
Actual state of the code

% Execution time

LAPACK by MKL multithreaded

Speedup = \frac{\text{Elapsed time 1 thread}}{\text{Elapsed time N thread}}

2 x (Intel® Skylake 18 coeurs)
Is there a faster alternative to LAPACK?

**ScalAPACK** has been tested and it was slower than LAPACK

Libraries that use GPUs:

- SLEPc
- MUMPS
- cuSOLVER
- **MAGMA**
What is MAGMA?

Task Splitting & Task Scheduling

4 GPUs 1 CPU

call DGETRF(matrix_dimX, matrix_dimY, matrix, d_dim, piv_vector, info_lapack)
call magmfv_dgetrf(matrix_dimX, matrix_dimY, matrix, d_dim, piv_vector, info_lapack)
MAGMA was not installed in CALMIP

Mini-project with Atos

Nicolas Renon
Emmanuel Courcelle

Christophe Berthelot
Paul Karlshoefer

Me
The project start

Planning

1) Installation of MAGMA using intel-MKL and gnu-OpenBlas
2) Create simple MAGMA examples written in Fortran and C
3) Change TOP code to use MAGMA. Compiling TOP with MAGMA

First day

18 / 02 / 2021 09 / 07 / 2021
One month later

Planning

1) Installation of MAGMA using **intel-MKL** and **gnu-OpenBlas**

2) Create simple MAGMA examples written in **Fortran** and **C**

3) Change TOP code to use MAGMA. Compiling TOP with MAGMA

Everything works and speedup tests look good!
One month later

Planning

1) Installation of MAGMA using intel-MKL and gnu-OpenBlas
2) Create simple MAGMA examples written in Fortran and C
3) Change TOP code to use MAGMA. Compiling TOP with MAGMA

Everything works and speedup tests look good!

MAGMA do not work correctly with big matrices (> 64 Go)
Testing big matrices

MAGMA \texttt{int( 4 )}:
4 bytes, \([-2.147.483.648 : +2.147.483.647]\]

MAGMA \texttt{int( 8 )}:
8 bytes, \([-9.223.372.036.854.775.808 : 9.223.372.036.854.775.807]\]

Simple examples \checkmark work as expected with \texttt{int( 8 )} MAGMA version

TOP \xmark is still giving some strange results. WHY?

One month
Why TOP do not work with MAGMA int8?

- Checking differences between the examples and TOP
  Same flags were used in both cases.
  Linked libraries where the same. Checked using “lssd”.

- Contact Marc Gates, MAGMA developer => Change all “int” into “int8” within MAGMA.

- TOP is written in Fortran but it is run from a Python interface thanks to F2PY.
Speed-up results

- 2 x (Intel® Skylake 18 coeurs)
- 4 x (GP-GPU Nvidia V100)

LU Speed-up  MKL_36_threads/ MAGMA

Matrix size (Go)

Speed-up

One month  Four months
Speed-up results

Maximum LU speedup
(1 thread / MAGMA) = 112
Total acceleration
1 hour to 1 minute

One month  Four months
Conclusion

- Small bugs can take time to be solved...
- MAGMA is faster on GPU than multi-threaded LAPACK for the tests we performed
- It is easy to move from LAPACK to MAGMA
- Thanks to MAGMA our code is prepared for future hardware changes
- Other codes in LPT & IRAP are going to take advantage from MAGMA
Thank you for your attention