Hands-on tutorial on PASTIX with GPU
11/02/2021

Tony Delarue — Mathieu Faverge — Pierre Ramet
Introduction
Problem and context

Problem: Solve $Ax = b$

- Factorize $A = LU$, where $A$ is a sparse matrix
- Solve $Ly = b$
- Solve $Ux = y$
Problem and context

Problem: Solve $Ax = b$

- Factorize $A = LU$, where $A$ is a sparse matrix
- Solve $Ly = b$
- Solve $Ux = y$

Main steps of the sparse solver

1. Reorganize the unknowns to reduce the fill-in.
2. Create the symbolic matrix $L$.
3. Factorize the matrix.
4. Solve the linear system.
PAStiX factorization principle

Algorithm for a column-block

1. **Factorize** the diagonal block (POTRF/GETRF).
2. **Solve** extra-diagonal blocks (TRSM).
3. **Update** the other column-blocks (GEMM).
## PAStiX functionalities

### Functionalities by scheduler

<table>
<thead>
<tr>
<th></th>
<th>Seq/Static/Dynamic</th>
<th>PARSEC/STARPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>POTRF (Cholesky)</td>
<td>SHM/MPI/LR</td>
<td>SHM/MPI/LR/GPU</td>
</tr>
<tr>
<td>PXTRF ($LL^t$ for complex)</td>
<td>SHM/MPI/LR</td>
<td>SHM/MPI/LR/GPU</td>
</tr>
<tr>
<td>HETRF ($LDL^h$)</td>
<td>SHM/MPI/LR</td>
<td>SHM/MPI/LR/GPU</td>
</tr>
<tr>
<td>SYTRF ($LDL^t$)</td>
<td>SHM/MPI/LR</td>
<td>SHM/MPI/LR/GPU</td>
</tr>
<tr>
<td>GETRF ($LU$)</td>
<td>SHM/MPI/LR</td>
<td>SHM/MPI/LR/GPU</td>
</tr>
</tbody>
</table>
Create a task diagram. It allows us to anticipate dependencies between tasks.

Share the data on the different computation devices.

Take care of computer heterogeneity.
Obtaining better performances with PAStiX-GPU
Define your performance model

- **PASTiX** allows you to define your performance model.
- **POTRF/GETRF** time kernel estimation:
  \[ a_3 \times N^3 + a_2 \times N^2 + a_1 \times N + a_0 \]
- **TRSM** time kernel estimation:
  \[ a_5 \times M \times N^2 + a_4 \times M \times N + a_3 \times N^2 + a_2 \times M + a_1 \times N + a_0 \]
- **GEMM** time kernel estimation:
  \[ a_7 \times M \times N \times K + a_6 \times M \times K + a_5 \times K \times N + a_4 \times M \times N + a_3 \times M + a_2 \times N + a_1 \times K + a_0 \]
- Need to be coherent with your hardware.
- Default value: \( a_7 = 2.1.2e12 \)
Obtaining better performances - granularity

1D or 2D task

We can play with the granularity of the computation tasks.
- 1D if we consider a block-column.
- 2D if we consider only blocks.
Obtaining better performances - granularity

PaStiX compared performance with StarPU on sirocco17 depending on the blocksize
In your home directory, you can create a $HOME/parsec/mca-params.conf file to better configure PARSEC.

- `device.show_capabilities = 1`
- `device.show_statistics = 1`
- `device.cuda.max_streams = 10 # ≥ 3 #`
- `device.cuda.max_events_per_stream = 4`
- `runtime.comm.short_limit = 0`
Obtaining better performances - STARPU

STARPU contains a set of environment variables to define its comportement with GPUs:

- `STARPU_CUDA_PIPELINE=4`
- `STARPU_NWORKER_PER_CUDA=8`
- `STARPU_CUDA_THREAD_PER_WORKER= [0||1]`

You can either export them or call them at the beginning of your command line.
Obtaining better performances - Experiments

PaStiX compared GPU performance on sirocco17 depending of the number of streams per GPU

ParSEC-GPU performances on Sirocco17

StarPU-GPU performances on Sirocco17
Experiments with EoCoE matrix

StarPU-GPU factorization performances and time on Sirocco17 for the Alya matrix.

2 GPUs with 2D tasks and 8 streams per GPU.
3
Conclusion
## Conclusion

- Creation of user tutorials for PASTiX-GPU use.
- Highlight the parameters to look at according to your GPU.
- Give performance results of EoCoE matrices.

## Future works

- Make PASTiX-GPU scale with PARSEC.
- Understand the gap of performances between PARSEC and STARPUS.
- Improve PASTiX-MPI implementation with runtimes to be efficient with GPUs.
Merci pour votre attention !