Motivation

Efforts to achieve communication-optimality:
Motivation

Efforts to achieve communication-optimality:

Figure 2: Illustratory evolution of MMM algorithms reaching the I/O lower bound.
Portability and Usability

Frontend
- **COSMA LAYOUT**
- **CUSTOM LAYOUT**
- **SCALAPACK LAYOUT**
- **C/C++ INTERFACE**
- **FORTRAN INTERFACE**

Backend
- **CPU**
- **GPU**
- **CUDA**
- **ROCm**
Follow the 30 seconds tutorial: https://github.com/eth-cscs/COSMA#using-cosma-in-30-seconds
Follow the 30 seconds tutorial: https://github.com/eth-cscs/COSMA#using-cosma-in-30-seconds

**COMPILE COSMA**

- `git clone --recursive https://github.com/eth-cscs/COSMA cosma` &
  `cd cosma`
- `mkdir build && cd build`
- `cmake -DCOSMA_BLAS=CUDA -DCOSMA_SCALAPACK=MKL -DCMAKE_INSTALL_PREFIX=<install-dir>`
- `make -j 8`
- `make install`
Follow the 30 seconds tutorial: https://github.com/eth-cscs/COSMA#using-cosma-in-30-seconds

### COMPILE COSMA

- git clone --recursive https://github.com/eth-cscs/COSMA cosma
- && cd cosma
- mkdir build && cd build
- cmake -DCOSMA_BLAS=CUDA -DCOSMA_SCALAPACK=MKL -DCMAKE_INSTALL_PREFIX=<install-dir>..
- make -j 8
- make install

### LINK TO COSMA

- # link to COSMA, before any SCALAPACK
- LIBS += -L<install-dir>/lib64 -lcosma_pxgemm -lcosma -lgrid2grid -lTiled-MM -lcublas -lcudart -lrt
- # include headers
- INCS += -I<install-dir>/include
**SCALAPACK LAYOUT**

**used in CP2K**

**COMPILE COSMA**

- git clone --recursive
  https://github.com/eth-cscs/COSMA cosma
  && cd cosma
- mkdir build && cd build
- cmake -DCOSMA_BLAS=CUDA
  -DCOSMA_SCALAPACK=MKL
  -DCMAKE_INSTALL_PREFIX=<install-dir>
  ..
- make -j 8
- make install

**LINK TO COSMA**

# link to COSMA, before any SCALAPACK

- LIBS += -L<install-dir>/lib64
  -lcosma_pxigemm
  -lcosma -lgrid2grid
  -lTiled-MM
  -lcublas -lcudart -lrt

# include headers

- INCS += -I<install-dir>/include
**COMPILATION OF COSMA**

- `git clone --recursive https://github.com/eth-cscs/COSMA cosma` 
- `&& cd cosma`
- `mkdir build && cd build`
- `cmake -DCOSMA_BLAS=CUDA -DCOSMA_SCALAPACK=MKL -DCMAKE_INSTALL_PREFIX=<install-dir> ..`
- `make -j 8`
- `make install`

**LINK TO COSMA**

- `# link to COSMA, before any SCALAPACK`
- `LIBS += -L<install-dir>/lib64 -lcublas -lcudart -lrt`
- `# include headers`
- `INCS += -I<install-dir>/include`

---

**SCALAPACK LAYOUT**

- **used in CP2K**
- **user code untouched!**
COSMA
SCALAPACK LAYOUT

VS

SCALAPACK

- Create communicators
- Allocate memory
- Solve Perfect Matching
- Transform Data Layout
- Transpose
- Multiply
- Free communicators
- Free memory
256 nodes, including all the overheads

{$P = 16 \times 16$}

$\uparrow$ higher = better

$\sim 3 \times$ faster on GPU

$\sim 2 \times$ faster on CPU

Matrix Dimensions (square)
CP2K: RPA 128 water molecules
### SCALAPACK LAYOUT

**CP2K: RPA 128 water molecules**

<table>
<thead>
<tr>
<th>PDGEMM INFO</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DIMENSIONS</strong></td>
</tr>
<tr>
<td>M</td>
</tr>
<tr>
<td>17408</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>BLOCK DIMENSIONS</strong></th>
<th><strong>TRANSPOSE FLAGS</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOCK M</td>
<td>BLOCK N</td>
</tr>
<tr>
<td>8704</td>
<td>8704</td>
</tr>
</tbody>
</table>
### CP2K: RPA 128 water molecules

#### PDGEMM INFO

<table>
<thead>
<tr>
<th>M</th>
<th>N</th>
<th>K</th>
<th>ROWS</th>
<th>COLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>17408</td>
<td>17408</td>
<td>3473408</td>
<td>128</td>
<td>1</td>
</tr>
</tbody>
</table>

#### BLOCK DIMENSIONS

<table>
<thead>
<tr>
<th>BLOCK M</th>
<th>BLOCK N</th>
<th>BLOCK K</th>
<th>MATRIX A</th>
<th>MATRIX B</th>
</tr>
</thead>
<tbody>
<tr>
<td>8704</td>
<td>8704</td>
<td>13568</td>
<td>transposed (T)</td>
<td>non-transposed (N)</td>
</tr>
</tbody>
</table>

#### DIAGRAM

46 x

~3.5 million x ~17.5 k = ~17.5 k
RPA BENCHMARK, PIZ DAINT, 128 GPU NODES

- ~7 × faster PDGEMM
- ~3 × faster TOTAL TIME
- ~25% faster PDGEMM

GPU-accelerated

- COSMA
- CRAY-LIBSCI_ACC
- MKL
- CRAY-LIBSCI

OTHERS

PDGEMM

LOWER = BETTER
RPA BENCHMARK, PIZ DAINT, 1024 GPU NODES

- COSMA: 99.86 s
- CRAY-LIBSCI_ACC: 177.86 s

- PDGEMM: ~2 × faster
- OTHERS: also improved

COSMA also helps in other routines like Cholesky!

\[\downarrow \text{lower = better} \]
COSMA API

- substantial change
- minimal change
- user-code unchanged

- integration effort
- performance gain

- COSMA LAYOUT: max performance
- CUSTOM LAYOUT: moderate performance
- SCALAPACK LAYOUT: moderate performance

user-code unchanged
COSMA can deal with matrices that are very irregularly distributed as well.
CUSTOM LAYOUT

row-slices

col-slices

local view

global view

pointers

stride

P1

P0

P2

P3
COSMA API

- **substantial change**
  - COSMA LAYOUT
  - max performance
- **minimal change**
  - CUSTOM LAYOUT
  - moderate performance
- **user-code unchanged**
  - SCALAPACK LAYOUT
  - moderate performance

integration effort vs. performance gain
<table>
<thead>
<tr>
<th>shape</th>
<th>benchmark</th>
<th>ScaLAPACK total comm. volume per rank [MB]</th>
<th>speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>min</td>
<td>mean</td>
</tr>
<tr>
<td>AC</td>
<td>strong scaling</td>
<td>203</td>
<td>222</td>
</tr>
<tr>
<td></td>
<td>limited memory</td>
<td>816</td>
<td>986</td>
</tr>
<tr>
<td></td>
<td>extra memory</td>
<td>303</td>
<td>350</td>
</tr>
<tr>
<td>BC</td>
<td>strong scaling</td>
<td>2636</td>
<td>2278</td>
</tr>
<tr>
<td></td>
<td>limited memory</td>
<td>368</td>
<td>541</td>
</tr>
<tr>
<td></td>
<td>extra memory</td>
<td>133</td>
<td>152</td>
</tr>
<tr>
<td>AB</td>
<td>strong scaling</td>
<td>3507</td>
<td>2024</td>
</tr>
<tr>
<td></td>
<td>limited memory</td>
<td>989</td>
<td>672</td>
</tr>
<tr>
<td></td>
<td>extra memory</td>
<td>122</td>
<td>77</td>
</tr>
<tr>
<td>A</td>
<td>strong scaling</td>
<td>134</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>limited memory</td>
<td>47</td>
<td>101</td>
</tr>
<tr>
<td></td>
<td>extra memory</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>overall</td>
<td>1.07</td>
<td>2.17</td>
</tr>
</tbody>
</table>
• Best Student Paper Award at SC19 supercomputing conference in Denver, US

• In collaboration with Prof. Torsten Hoefler (ETH Zurich)

• Code: https://github.com/eth-cscs/COSMA
• ACM Digital Library: https://dl.acm.org/doi/10.1145/3295500.3356181
• Arxiv: https://arxiv.org/abs/1908.09606
• YouTube Presentation: https://www.youtube.com/watch?v=5wiZWw5ItR0

Questions?
marko.kabic@cscs.ch
## SCALAPACK-wraper

### 128 nodes: Piz Daint Supercomputer (Cray XC50)

<table>
<thead>
<tr>
<th>ALGORITHM</th>
<th>CPU-ONLY</th>
<th>CPU-ONLY</th>
<th>CPU-ONLY</th>
<th>GPU ACCELERATED</th>
<th>GPU ACCELERATED</th>
<th>GPU ACCELERATED</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CRAY-LIBSCI</td>
<td>MKL</td>
<td>COSMA-CPU</td>
<td>CRAY-LIBSCI_ACC</td>
<td>COSMA-GPU</td>
<td>COSMA-GPU</td>
</tr>
<tr>
<td><strong>CONFIGURATION</strong></td>
<td>1MPI x 12T</td>
<td>1MPI x 12T</td>
<td>1MPI x 12T</td>
<td>1MPI x 12T</td>
<td>1MPI x 12T</td>
<td>1MPI x 12T</td>
</tr>
<tr>
<td>CP2K RPA-RI 128-H2O [s]</td>
<td>6379.14</td>
<td>2305.41</td>
<td>2238.94</td>
<td>865.73</td>
<td>781.60</td>
<td></td>
</tr>
<tr>
<td>46 x PDGEMM [s]</td>
<td>5896.45</td>
<td>1836.85</td>
<td>1723.62</td>
<td>338.47</td>
<td>257.99</td>
<td></td>
</tr>
<tr>
<td>NODE GFLOP/s</td>
<td>128.30</td>
<td>411.87</td>
<td>438.92</td>
<td>2235.19</td>
<td>2932.44</td>
<td></td>
</tr>
<tr>
<td>% PEAK PERF.</td>
<td>25.70%</td>
<td>82.51%</td>
<td>87.92%</td>
<td>49.67%</td>
<td>65.17%</td>
<td></td>
</tr>
<tr>
<td><strong>NODE TYPE (128 nodes)</strong></td>
<td>Intel® Xeon® E5-2690 v3 @ 2.60GHz (12 cores, 64GB RAM)</td>
<td>NVIDIA® Tesla® P100 16GB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>NODE PEAK PERF[GFLOP/s]</strong></td>
<td>499.2</td>
<td>4500</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This is only using CPU nodes on the GPU partition of Piz Daint. However, CPU node peak perf is much higher on the CPU partition. Max peak assumes the data is already on GPU, which explains why it is not fully achieved.

- ~10% faster
- ~25% faster
1024 nodes: Piz Daint Supercomputer (Cray XC50)

<table>
<thead>
<tr>
<th>ALGORITHM</th>
<th>CRAY-LIBSCI_ACC</th>
<th>COSMA-GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIGURATION</td>
<td>1MPI x 12T</td>
<td>1MPI x 12T</td>
</tr>
<tr>
<td>CP2K RPA-RI 128-H2O [s]</td>
<td>317.68</td>
<td>175.12</td>
</tr>
<tr>
<td>46 x PDGEMM [s]</td>
<td>139.82</td>
<td>75.26</td>
</tr>
<tr>
<td>NODE GFLOP/s</td>
<td>676.37</td>
<td>1256.49</td>
</tr>
<tr>
<td>% PEAK PERF.</td>
<td>15.03%</td>
<td>27.92%</td>
</tr>
<tr>
<td>NODE TYPE (1024 nodes)</td>
<td>NVIDIA® Tesla® P100 16GB</td>
<td></td>
</tr>
<tr>
<td>NODE PEAK PERF[GFLOP/s]</td>
<td>4500</td>
<td></td>
</tr>
</tbody>
</table>

~2× faster
COSMA can deal with matrices that are very irregularly distributed as well.
CUSTOM LAYOUT

col-slices

row-slices

global view

local view

pointers

stride
CUSTOM LAYOUT

![Diagram of custom layout with row-slices, col-slices, global view, and local view.]
IDEA: relabel the ranks to minimize the communication cost
**IDEA:** relabel the ranks to minimize the communication cost
IDEA: relabel the ranks to minimize the communication cost
IDEA: relabel the ranks to minimize the communication cost
IDEA: relabel the ranks to minimize the communication cost
Optimal relabeling = maximum weighted perfect matching.