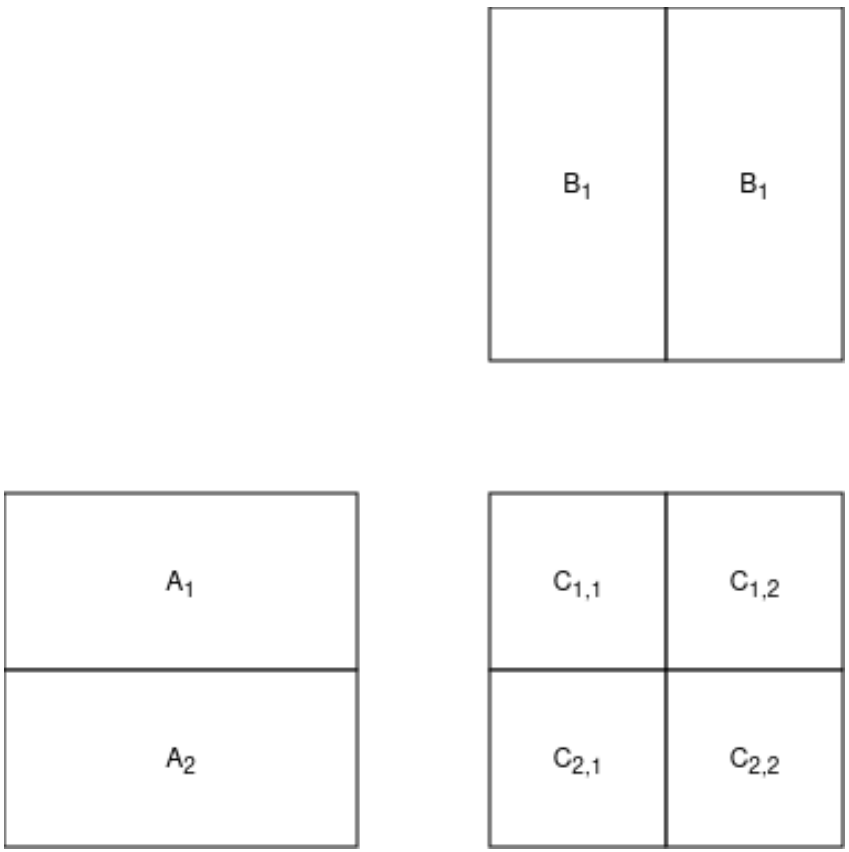
TER Project - StarPU reductions

Atte Torri - atte.torri@universite-paris-saclay.fr

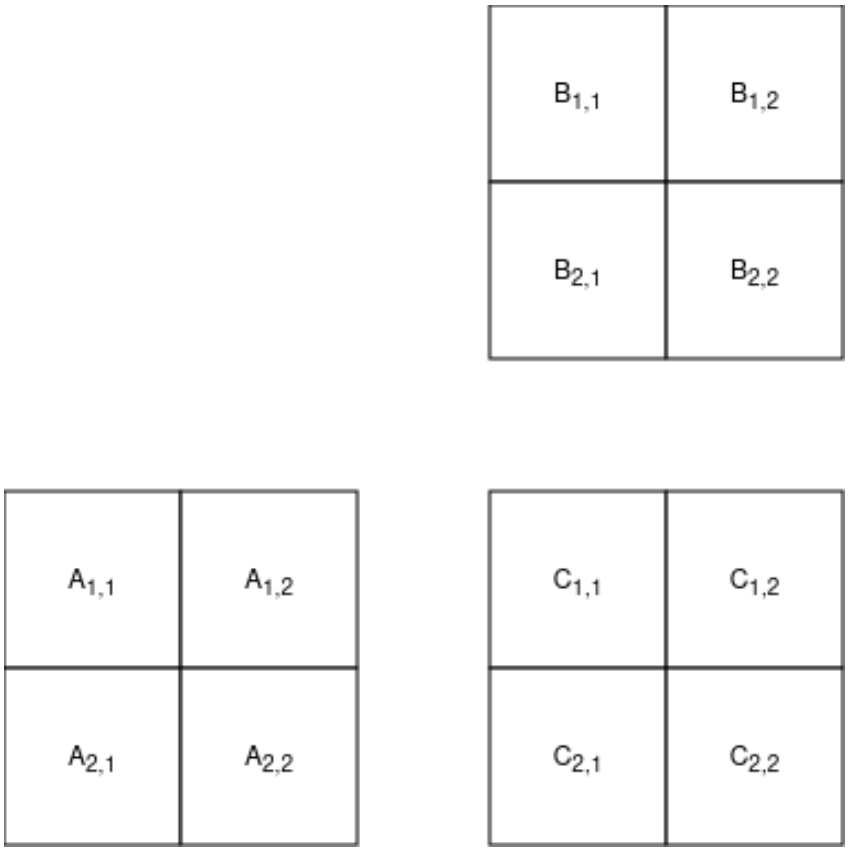
1D GEMM



- Large copies
- No pipelining

Figure: 1D GEMM

2D GEMM



- Pipelining
- Possibility for commutative operations
- Dependencies limit parallelism

Figure: 2D GEMM

Reduction operation

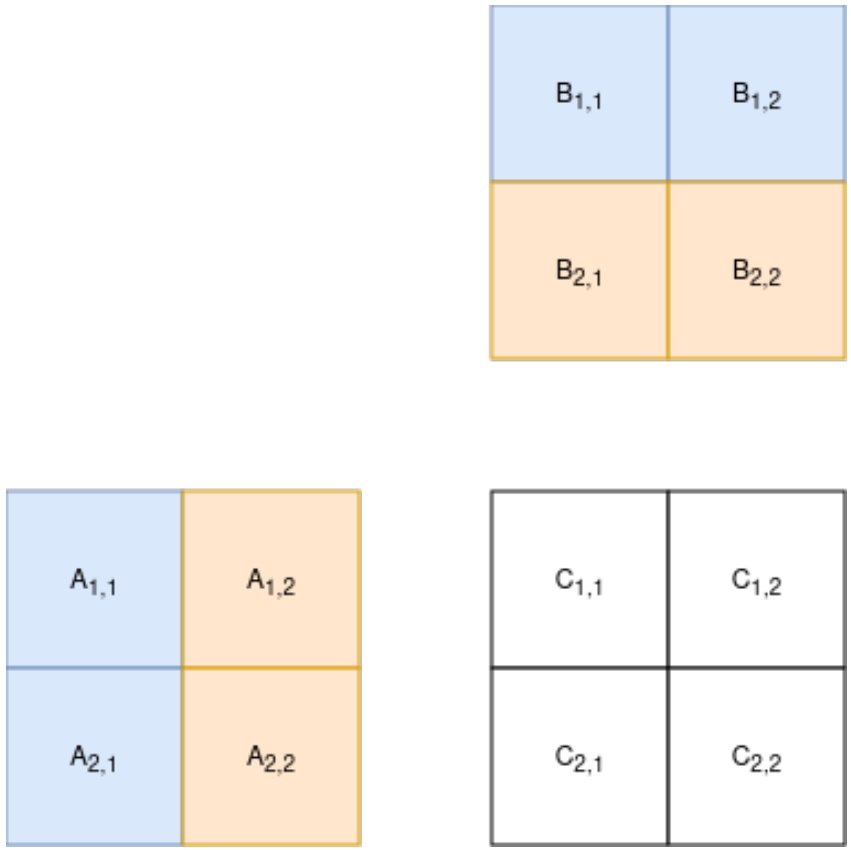


Figure: The GEMM contains an associative and commutative operation

Splitting into parallel computations

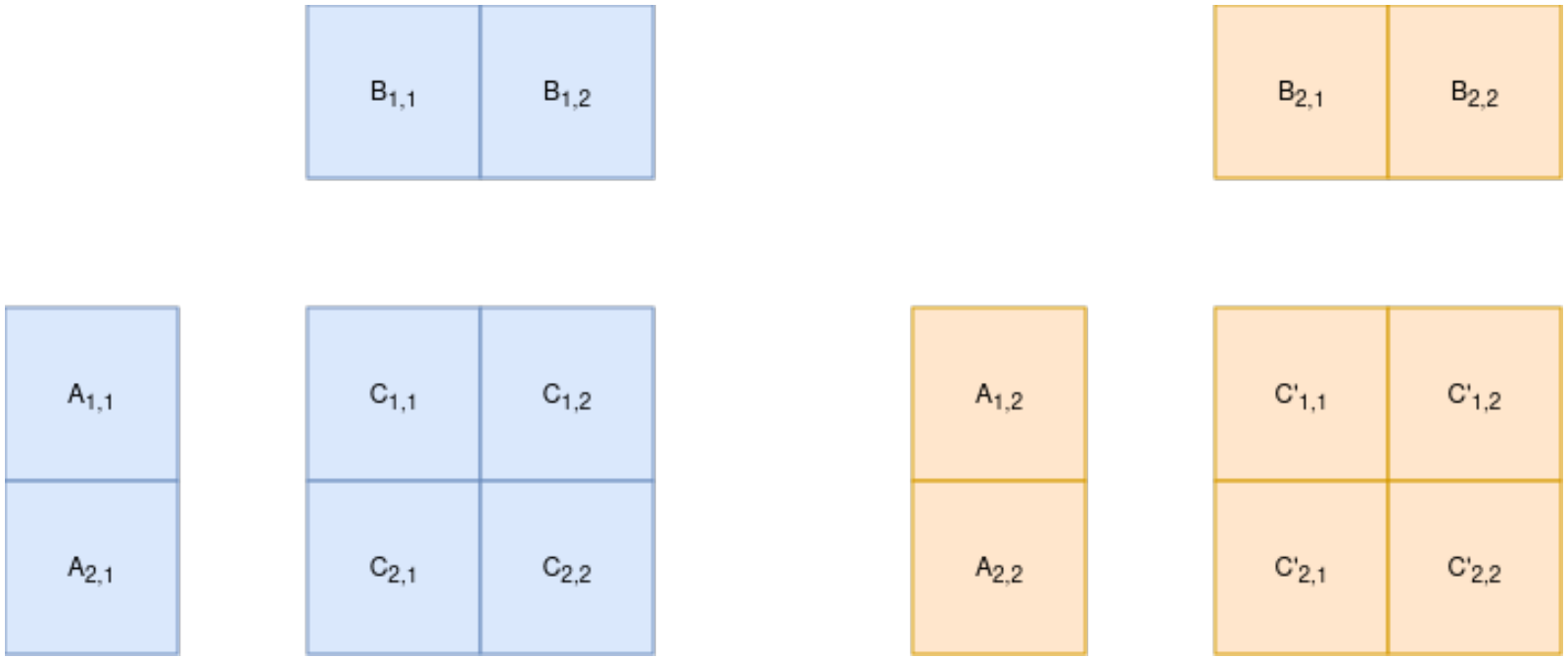


Figure: The GEMM operation can be split into multiple operation to be computed independently

Performing the reduction

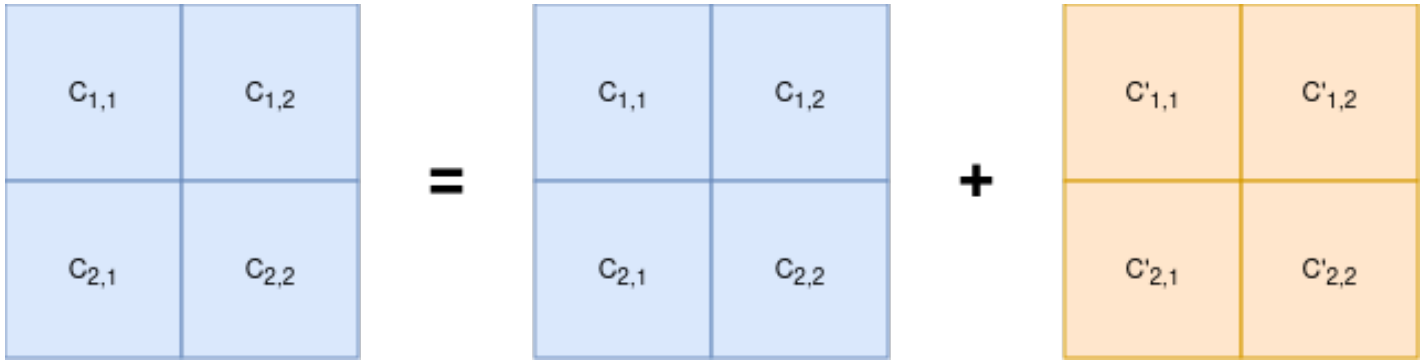


Figure: The independent results must be combined in a reduction operation

Reduction basics

- If the computation contains an associative operation multiple times in a row, we can split the computation into multiple independent sub-computations.
- The data buffer C in $C = \alpha \times A \times B + \beta \times C$ in a tiled computation is one such example.
- Note the tiled form for tile $i, j \in I, J$ can be written as $C_{i,j} = \sum_{k=0}^{K} \alpha \times A_{i,k} \times B_{k,j} + \beta \times C_{i,j}$.
- Reductions allow StarPU to create temporary buffers on each worker/machine, transferring data only when local computations are done.
- If the reduction operator is commutative, any reduction tree should work.

Using reductions in StarPU

- Creating initialisation and reduction codelets. These instruct StarPU how to create local data buffers and how to assemble them when the computation is complete.
- Bind the codelets to a data handle $\mathcal{H}$ with the function `starpu_data_set_reduction_methods`.
- Use the data handle $\mathcal{H}$ to perform some computations with mode flag `STARPU_REDUX` or `STARPU_MPI_REDUX`.
- Reduction wrapped up automaticaly by StarPU or by using the `starpu_mpi_redux_data` to benefit from priority setting or redction tree selection.
- `STARPU_REDUX` spawns temporary buffers on each worker on each contributor MPI node
- `STARPU_MPI_REDUX` spawns one temporary buffer per contributing MPI node

To do

- Implement reduction codelets for C data handle
- Implement reductions for MPI and non MPI cases.
- Try out different execution nodes for the GEMM tasks, either owner of A data, B data or C data.

Next

Continue working on the TER project.

```
# Login to cluster
ssh qdcster_XX@chome.metz.supelec.fr
# Allocate a machine to work on
salloc --partition cpu_tp_resa --time 4:00:00
       --reservation M1QDCS_TERSTARPU17 --exclusive
# Allocate multiple machines to run code interactively
salloc --partition cpu_tp_resa --qos 8nodespu
       --reservation M1QDCS_TERSTARPU17 --nodes 4
       --exclusive --time 4:00:00
# Run code with sbatch non-interactively
sbatch --partition cpu_tp_resa --qos 8nodespu
       --reservation M1QDCS_TERSTARPU17 --nodes 4
       --exclusive --time 4:00:00
       --export=ALL batch.sl
```