Parallel
Matrix-Matrix Multiplication

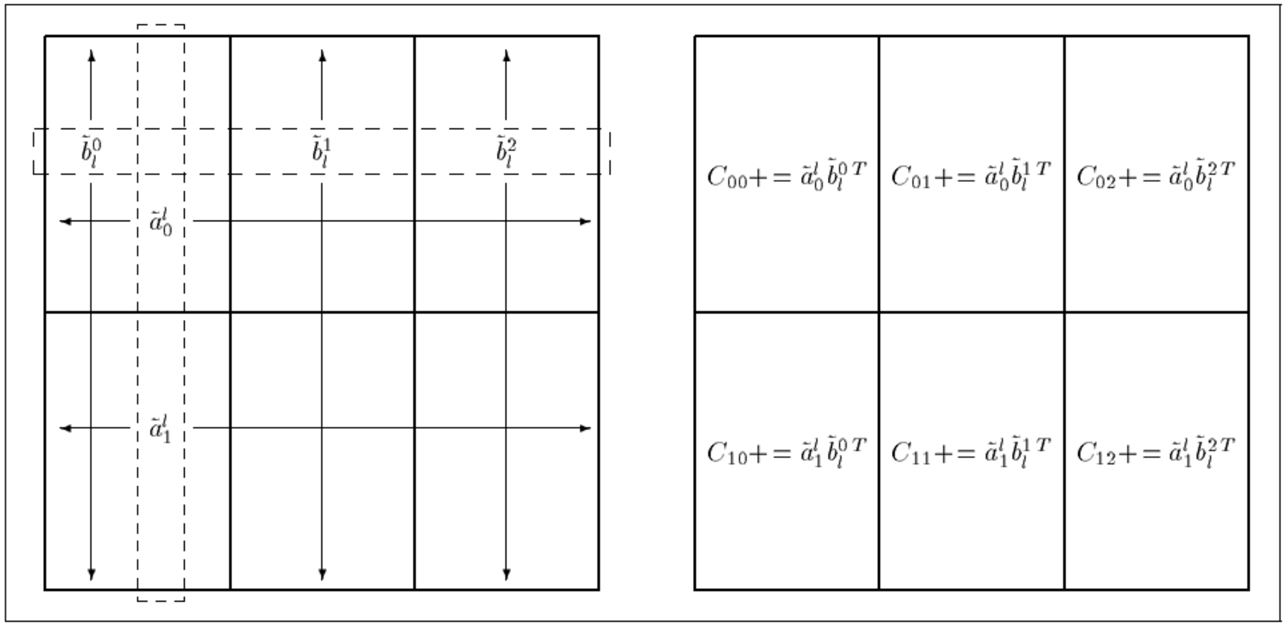
Parallel (Dense) Matrix- (Dense) Matrix Multiplication

- SUMMA by Robert A. van de Geijn and Jerreel Watts, Concurrency: Practice and Experience, 1997.
- SUMMA – Scalable Universal Matrix Multiplication Algorithm
- 2D block decomposition
- Formulated as a sequence of rank-one updates
- Each rank-one update is parallelized

Credit: Figures from the paper.

```
$C_{ij} = 0$
**for** $l = 0, k-1$
    broadcast $\tilde{a}_i^l$ within my row
    broadcast $\tilde{b}_l^j$ within my column
    $C_{ij} = C_{ij} + \tilde{a}_i^l \tilde{b}_l^{j\,T}$
**endfor**
```

Figure 1: Pseudo-code for $C = AB$.



$\tilde{b}_l^0$
$\tilde{b}_l^1$
$\tilde{b}_l^2$
$\tilde{a}_0^l$
$\tilde{a}_1^l$
$C_{00} {+}{=}\ \tilde{a}_0^l \tilde{b}_l^{0\,T}$
$C_{01} {+}{=}\ \tilde{a}_0^l \tilde{b}_l^{1\,T}$
$C_{02} {+}{=}\ \tilde{a}_0^l \tilde{b}_l^{2\,T}$
$C_{10} {+}{=}\ \tilde{a}_1^l \tilde{b}_l^{0\,T}$
$C_{11} {+}{=}\ \tilde{a}_1^l \tilde{b}_l^{1\,T}$
$C_{12} {+}{=}\ \tilde{a}_1^l \tilde{b}_l^{2\,T}$

Figure 2: Operations implementing the inner loop of Fig. 1 of a 2×3 mesh of nodes.

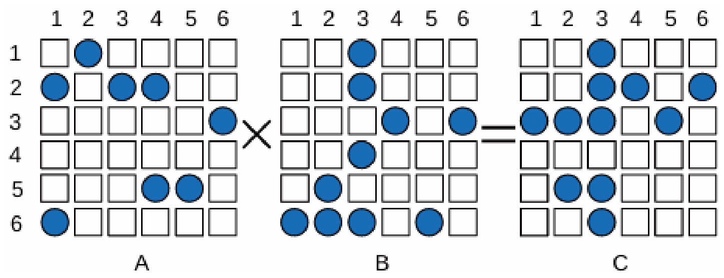
Optimizations

- Broadcast is pipelined with computations
- Instead of processing one column of A and one row of B for rank one update, process a set of columns of A and set of rows of B leading to matrix-matrix multiplication formulations.
- This will lead to blocking and higher cache efficiency.

Sparse Matrix – Sparse Matrix Multiplication: SPGEMM

- Sources:
  - “TileSpGEMM: a tiled algorithm for parallel sparse general matrix matrix multiplication on GPUs” by Niu et al., PPoPP 2022.
  - “Multithreaded sparse matrix-matrix multiplication for many-core and GPU architectures” by Deveci et al., Parco 2018

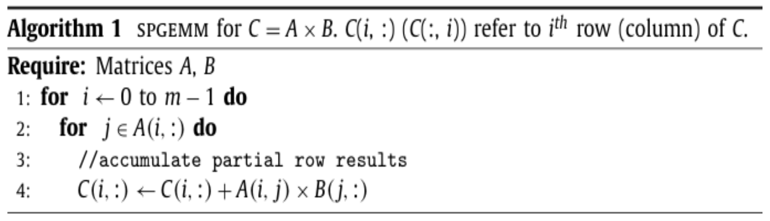
- SpGEMM – one of the fundamental building blocks in sparse linear solvers, graph processing frameworks and machine learning applications



Credit: Figure from the paper by Niu et al.

Gustavson's row-row formulation for parallelism

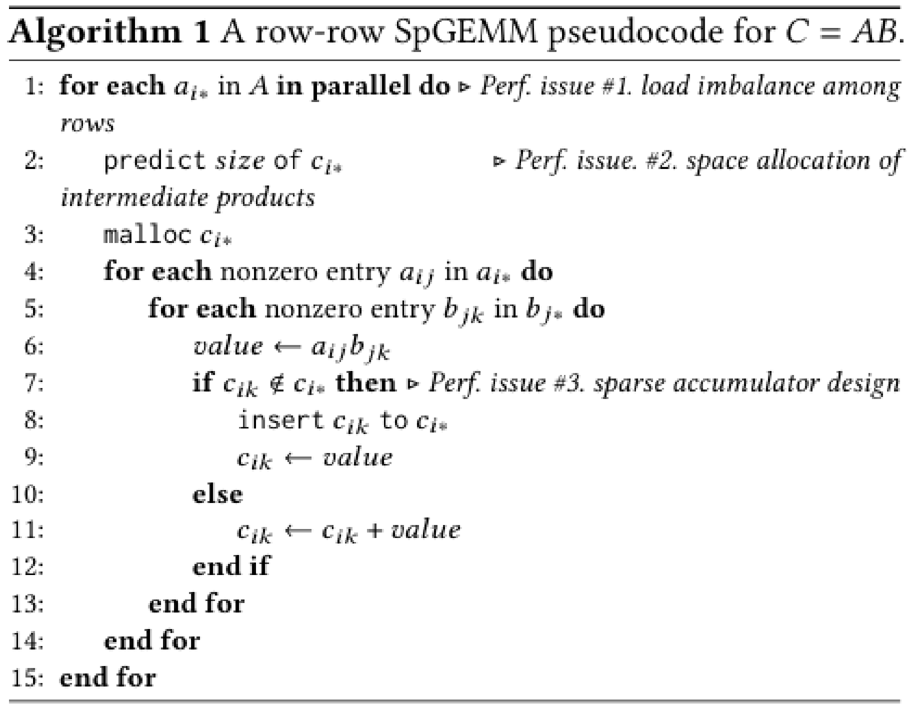
- Most parallel SpGEMM algorithms are based on Gustavson's formulation



Algorithm 1 SPGEMM for $C = A \times B$. $C(i, :)$ $(C(:, i))$ refer to i^{th} row (column) of C.

```
Require: Matrices $A$, $B$
for $i \leftarrow 0$ to $m-1$ do
    for $j \in A(i, :)$ do
        //accumulate partial row results
        $C(i, :) \leftarrow C(i, :) + A(i, j) \times B(j, :)$
```

Credit: By Deveci et al.



Algorithm 1 A row-row SpGEMM pseudocode for $C = AB$.

```
for each $a_{i*}$ in $A$ in parallel do ▷ Perf. issue #1. load imbalance among rows
    predict size of $c_{i*}$ ▷ Perf. issue. #2. space allocation of intermediate products
    malloc $c_{i*}$
    for each nonzero entry $a_{ij}$ in $a_{i*}$ do
        for each nonzero entry $b_{jk}$ in $b_{j*}$ do
            $value \leftarrow a_{ij} b_{jk}$
            if $c_{ik} \notin c_{i*}$ then ▷ Perf. issue #3. sparse accumulator design
                insert $c_{ik}$ to $c_{i*}$
                $c_{ik} \leftarrow value$
            else
                $c_{ik} \leftarrow c_{ik} + value$
            end if
        end for
    end for
end for
```

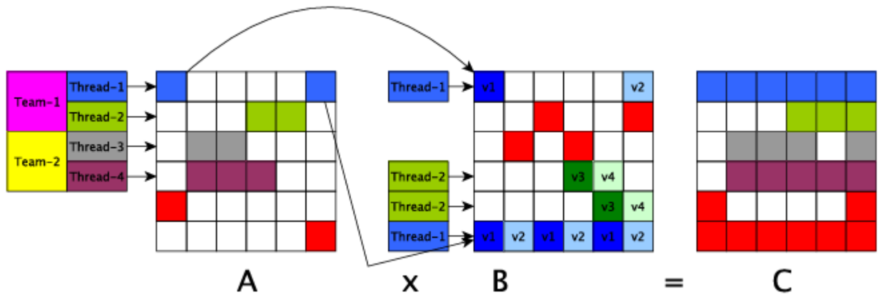
Credit: By Niu et al.

Parallelization

- 1-D row-based partitioning is the preferred choice; the rows of C can be computed independently
- For GPUs, a two-level hierarchy is followed:
  - Rows are assigned to first level of hierarchy (blocks or warps)
  - Calculations within rows done by second level (threads within blocks or warps)

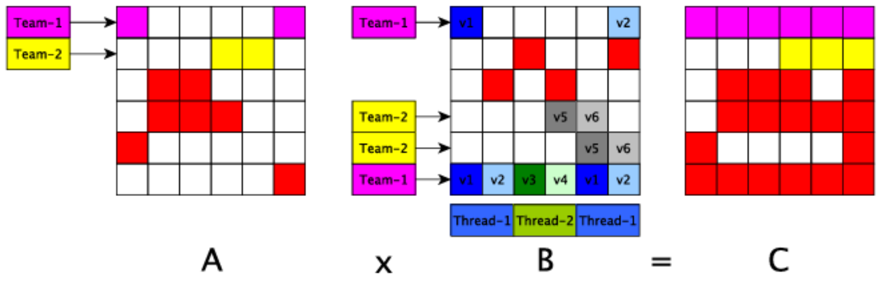
Three partitioning schemes for GPUs by Deveci et al.

- Thread-sequential: Each thread in a team works on a different row



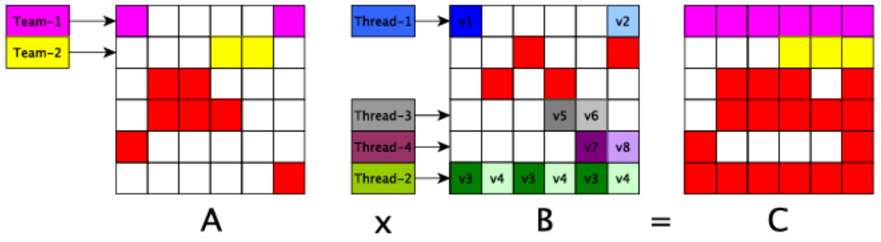
(a) Thread-Sequential: Thread-1 is assigned to a single row of A. It sequentially traverses the corresponding rows of B, one and six. It exploits vector parallelism for rows of B.

- Team-sequential: A whole team works on a single row



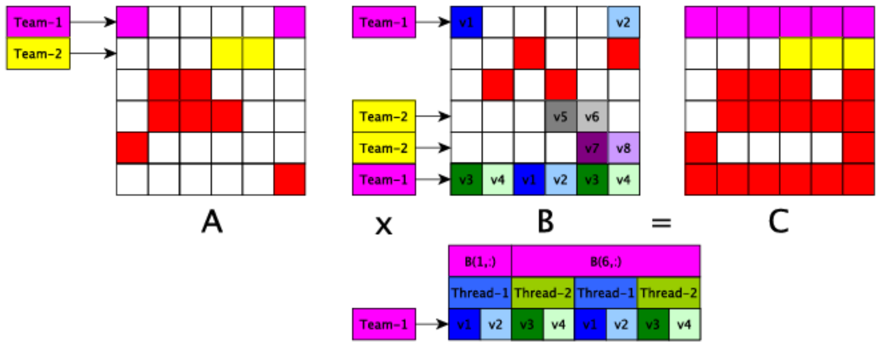
(b) Team-Sequential:Team-1 is assigned to a single row of A. It sequentially traverses the corresponding rows of B, one and six. It exploits both thread and vector parallelism for rows of B.

- Team-parallel: Start with a scheme like Team-sequential. But different threads in a Team work on different rows of B.



(c) Thread-Parallel: Team-1 is assigned to a single row of A. Thread-1 and Thread-2 work on first and sixth rows of B in parallel. They further exploit vector parallelism for rows of B.

- Thread-flat Parallel: A row of A is assigned to a Team, but the method flattens the second and third loops, and assign the computations to all the threads



(d) Thread-Flat-Parallel: Team-1 is assigned to single row of A. The multiplications are flattened as shown in the bottom, and both thread and vector parallelism are exploited in this single dimension. Thread-1 and thread-2 work on different portions of the sixth row of B