Distributed General Matrix Multiply and Add for a 2D Mesh Processor Network

Bo Kågström and Mikael Rännar
Department of Computing Science
University of Umeå
S-901 87 Umeå, Sweden

Bo.Kagstrom@cs.umu.se
Mikael.Rannar@cs.umu.se

DISTRIBUTED DGEMM

- Square 2D processor network with torus connectivity (logically).
- Possible to embed subtorus networks.
- Distribute $C(M \times N), A(M \times K)$ and $B(K \times N)$ blockwise
  - Square matrices $\longrightarrow$ Square blocks
  - Rectangular matrices $\longrightarrow$ Rectangular blocks
- Features:
  - User can perform _GEMM operations on *any* subarrays of A, B, C (with allowable sizes of subarrays), only using a square submesh.

    $C = \beta C + \alpha \text{op}(A) \cdot \text{op}(B),$

    where $\text{op}(X) = \{X, X^T\}$
  - Possible to perform different GEMM operations on non-overlapping submeshes simultaneously.

DISTRIBUTED DGEMM ALGORITHM

Distributed algorithm consists of two main parts:

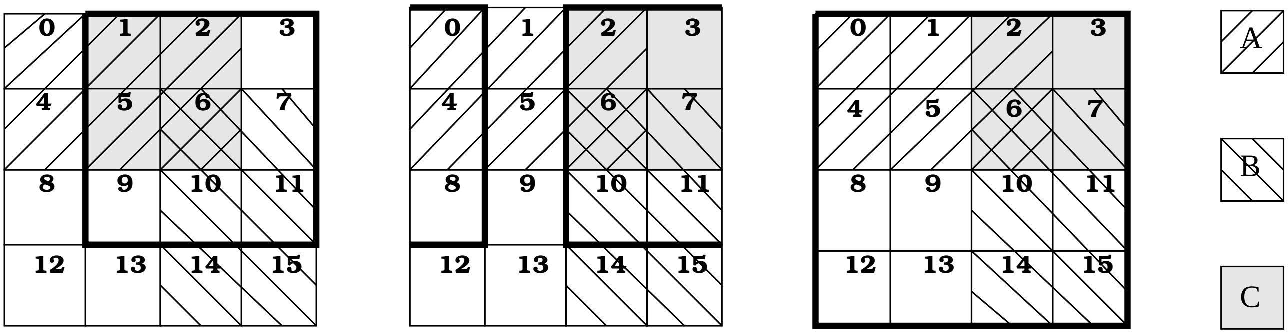
- Alignment and data movements of subarrays involved in the GEMM operation.
(expand, reshape, move)

- Distributed blocked matrix multiplication algorithm on (sub)matrices using only a square submesh.

The matrix C is never moved.

EXPAND

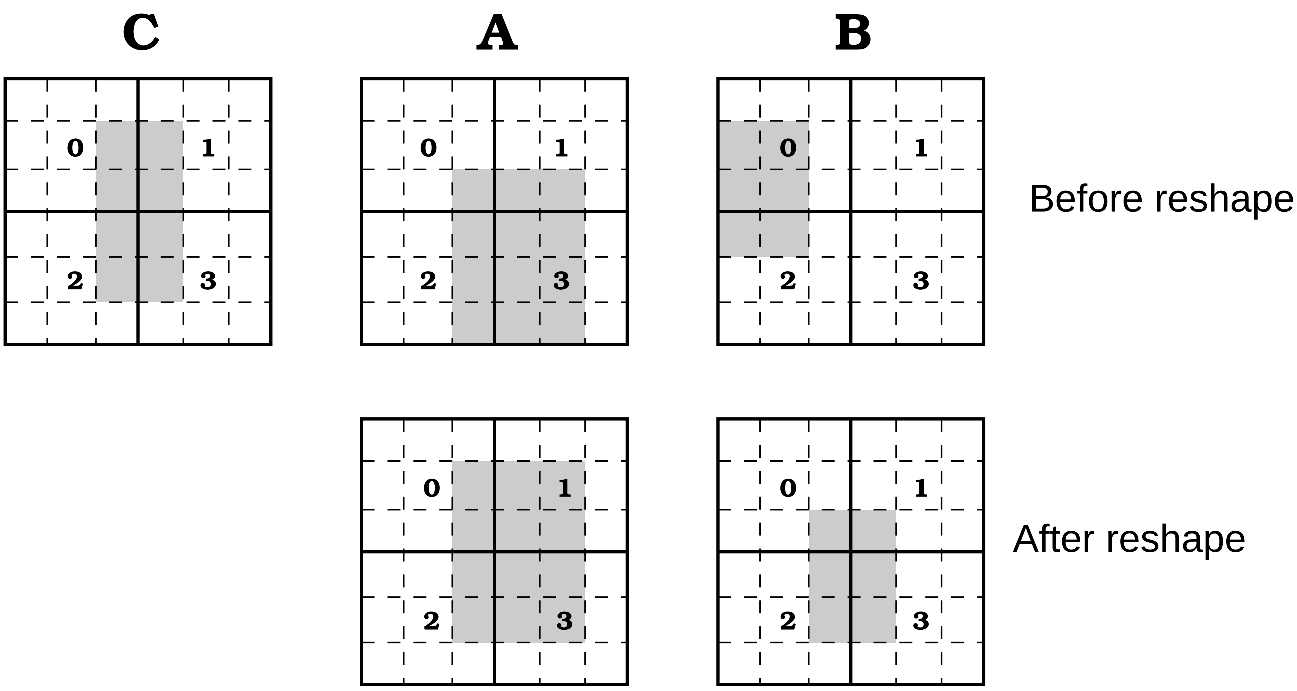
- Determine size of submesh involved in distributed block algorithm.
- `SubDIM` = max # processors that store subarrays involved.
- The submesh starts at processor storing the $(1,1)$–block of subarray C.



a) b) c)

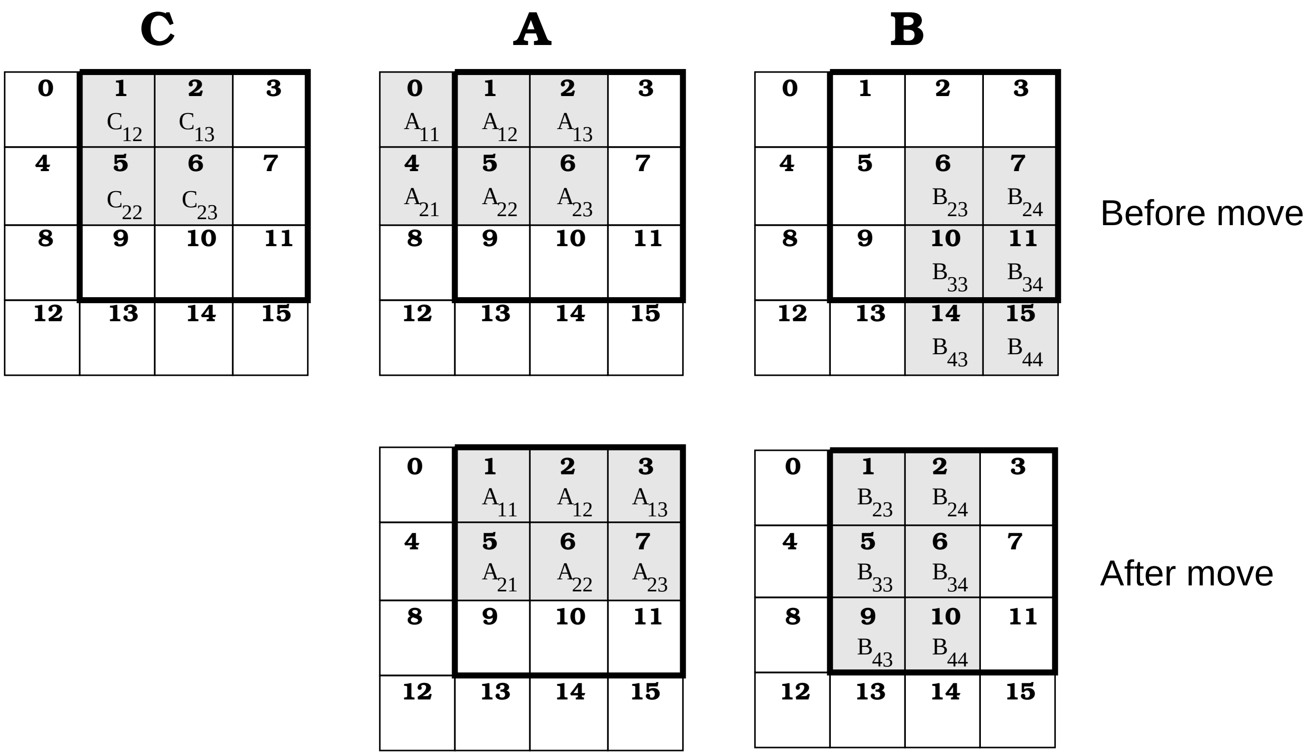
RESHAPE

- Move subarrays A and B such that the *local* matrix multiplications of the distributed algorithm are well-defined.
- Non-overlapping parts of A and B are adjusted in parallel.



MOVE

- Move subarrays A and B to submesh using the complete network.
- Non-overlapping parts of A and B are moved in parallel.



DISTRIBUTED BLOCK ALGORITHM

- Performs the actual GEMM operation.
- With a well defined interface it is possible to use different algorithms.
  - BLOCK–TORUS with nearest neighbour communication on subtorus.
    [Cannon, Cherkassky-Smith]
  - Broadcast–Multiply–Roll
    [Fox et al]
- Uses only a square submesh large enough to hold the subarrays
  $\Longrightarrow$
  can have more than one GEMM operation going on simultaneously.

OTHER APPROACHES

- PUMMA [J. Choi et al, 1993]

- BiMMeR [S. Huss-Lederman et al, 1994]

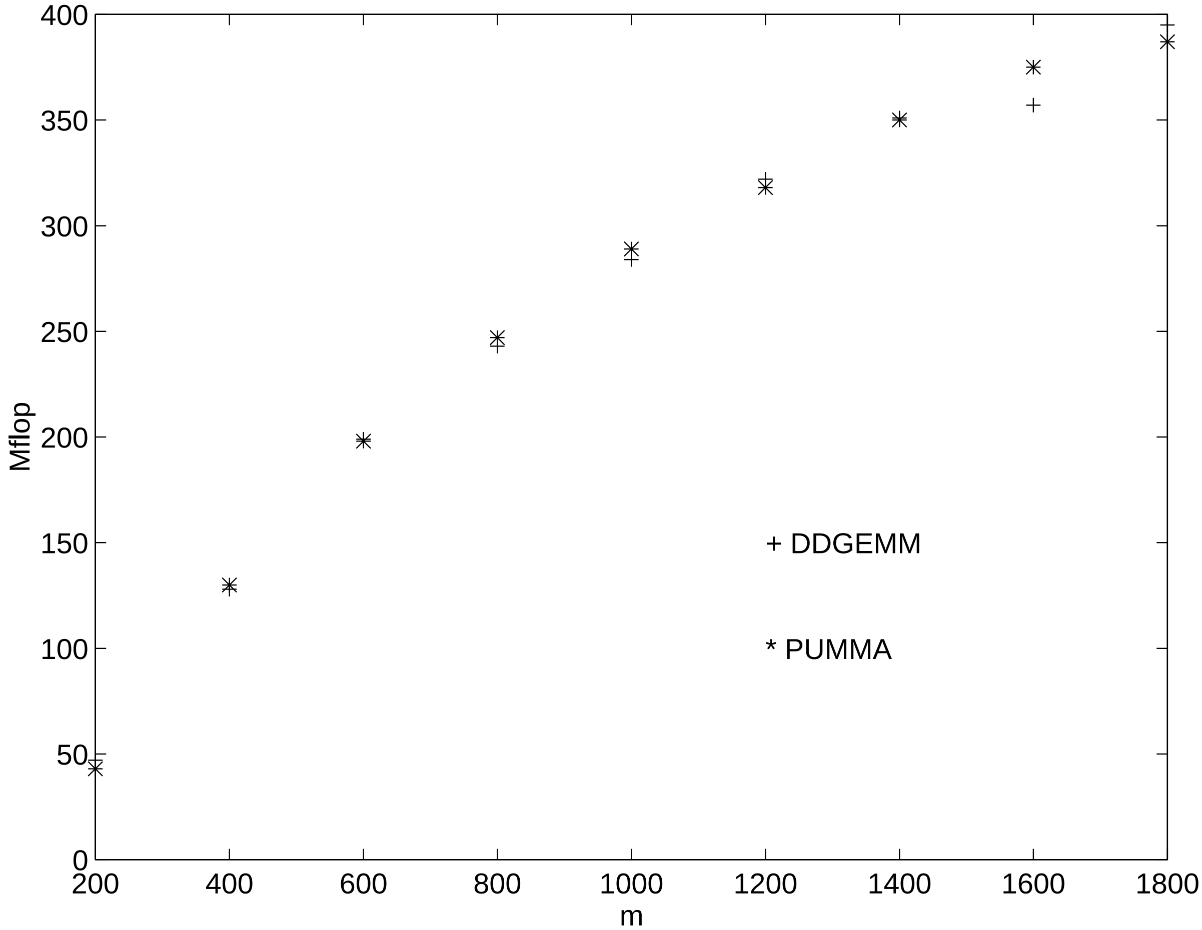
- SUMMA
  [R. van de Geijn and J. Watts, 1995]

- 3D// [R. Agarwal, F. Gustavson et al, 1995]

- Different vendors own products

⋮

DISTRIBUTED GEMM ON FULL MATRICES



ddgemm vs. PUMMA over various matrix sizes (m), using 16 nodes, logically a 4×4 mesh, on an Intel Paragon.

Similar performance.

PARALLEL SPEEDUP

$$S_p = \frac{\text{time of optimized single node } \texttt{dgemm}}{\text{time of } \texttt{ddgemm} \text{ on } p \text{ nodes}}$$

$$E_{p_c} = \frac{S_p}{p_c}$$

$$E_p = \frac{S_p}{p},$$

where p_c is the number of nodes storing a block of subarray C and p is the number of nodes in the submesh.

Maximal parallel speedup $= p_c$

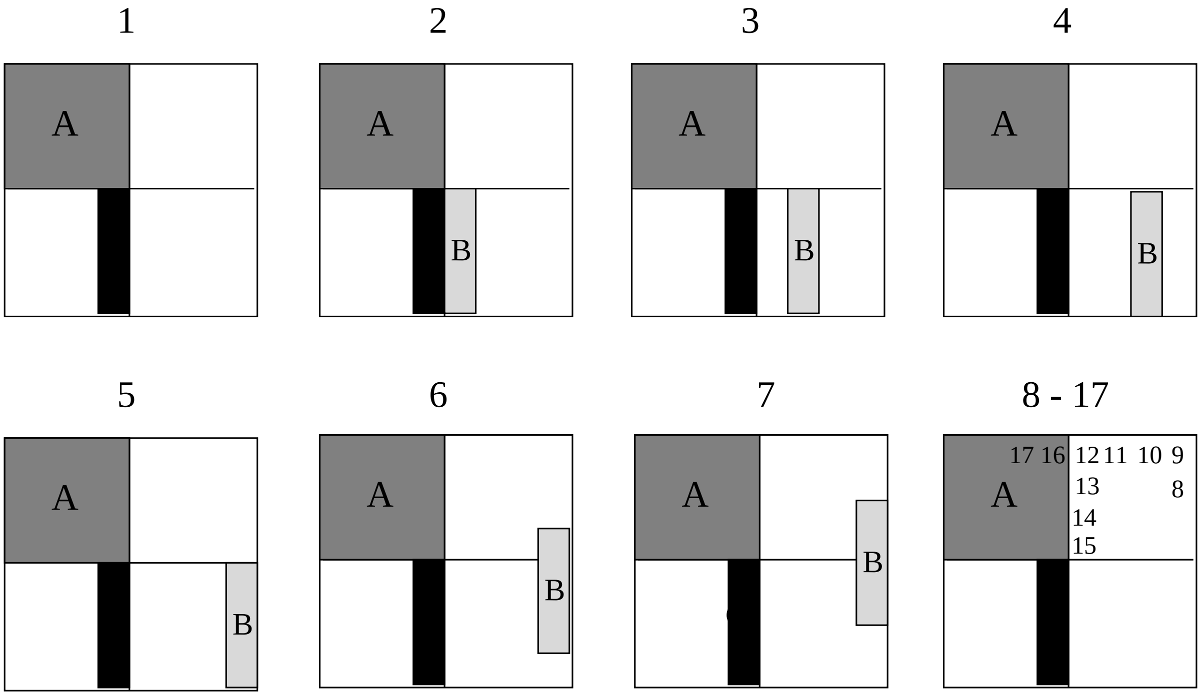
$E_{p_c} = E_p$
$\Longrightarrow$
Distributed GEMM operation on full submesh.

DDGEMM ON DIFFERENT SHAPES OF A, B AND C

Matrix sizes m	n	k	dgemm 1 node	ddgemm 16 nodes	S_p	p_c	E_{p_c} (%)
48	48	192	2833	2847	0.995	1	100
48	192	192	11332	2887	3.93	4	98
48	192	96	5448	1485	3.67	4	92
192	48	48	2688	769	3.50	4	87
192	192	48	10745	786	13.67	16	85
96	192	48	5365	767	6.99	8	87
192	48	192	11238	2880	3.90	4	98
192	192	192	44956	2930	15.34	16	96
96	192	96	10974	1488	7.38	8	92
96	48	192	5632	2864	1.97	2	98
96	96	192	11261	2885	3.90	4	98
48	96	192	5665	2858	1.98	2	99
48	192	48	2723	766	3.55	4	89

- Intel iPSC/2, using 16 nodes (4×4 mesh).
- A, B, and C "in place". (No reshape, no move)
- Mainly test of distributed block algorithm + initial overhead.
- Increasing k (# columns of A) $\Rightarrow$ Better utilization. (more local GEMM, communication constant)
- Similar results on Intel Paragon (Bergen).

TIMING WITH MOVE



No.	Time (ms)
1	20091
2	20250
3	20251
4	20561
5	20386
6	20728

No.	Time (ms)
7	20762
8	20289
9	20260
10	20580
11	20642
12	20979

No.	Time (ms)
13	20812
14	20726
15	20597
16	20842
17	20851

- Intel Paragon, 64 nodes (logically 8×8). $m = k = 1840$, $n = 460$
- No reshape (BLOCK = .true.)
- All subarrays "in place" – 18266 ms
- 10 – 15 % overhead

SUMMARY

- We provide a distributed general matrix multiply and add operation (`ddgemm`)
- Fortran 77
- Portable (PICL, PVM), easy to use
- Operates on subarrays as well as on full matrices
- All communication hidden for the user
- Uses only a square submesh
- Possible to perform several GEMM operations (or other computations) on non-overlapping submeshes simultaneously
- All node programs make the same call to `ddgemm` (SPMD model)
- GEMM-based approach also for distributed (parallel) level 3 BLAS
- Computational experiments show that our approach is effective