Scalable Matrix Architecture (Fundamentals and Vector-Matrix Facility)

José Moreira
IBM Corporation

Abel Bernabeu
Esperanto

Comments on the state-of-the-art for matrix computing

- Most high-performance matrix computations rely on the following kernel:

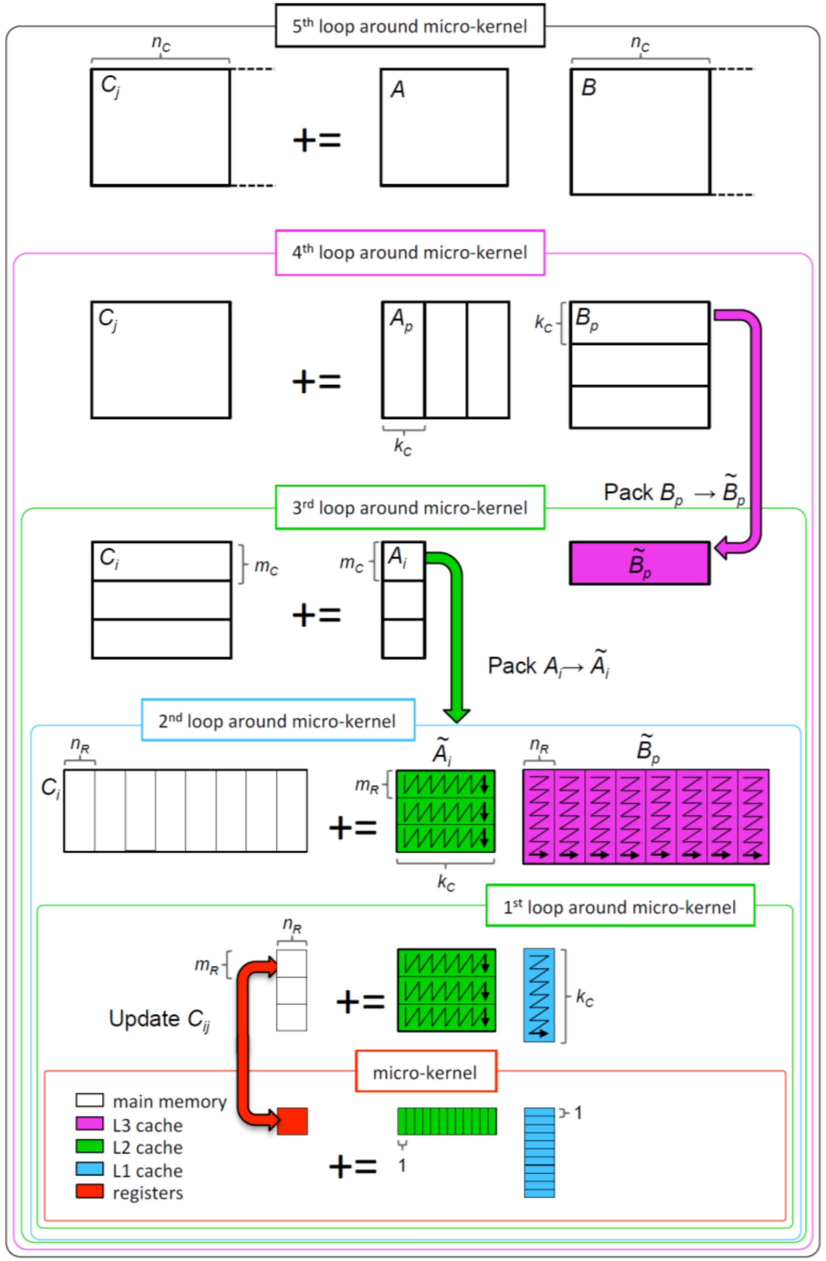
$$\boldsymbol{C}_{m\times n} = \boldsymbol{A}_{m\times K} \times \boldsymbol{B}_{K\times n}$$

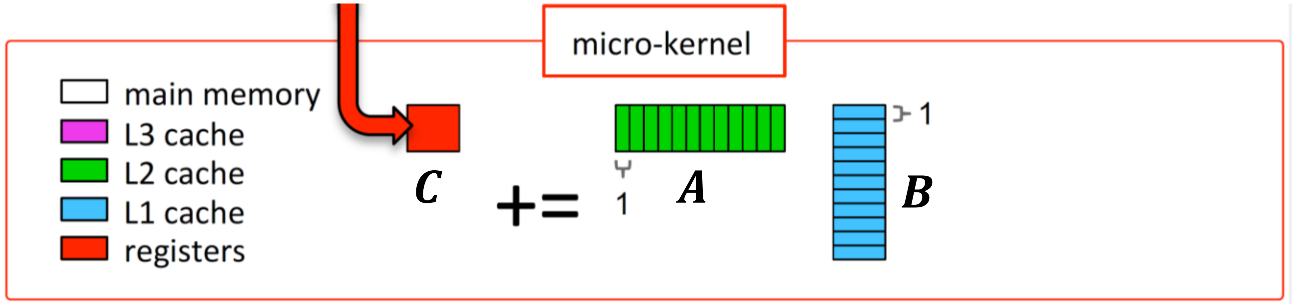
$\boldsymbol{C}_{m\times n}$	is a register-resident matrix, with $m \approx n$, and
$\boldsymbol{A}_{m\times K}$ and $\boldsymbol{B}_{K\times n}$	are streamed from memory, with $m \ll K$ and $n \ll K$

- Matrix $\boldsymbol{A}_{m\times K}$ is organized as a vector of K elements, each element a column vector of m elements of $\boldsymbol{A} = [\boldsymbol{A}^0 \quad \boldsymbol{A}^1 \quad \cdots \quad \boldsymbol{A}^{K-1}]$
- Matrix $\boldsymbol{B}_{K\times n}$ is organized as a vector of K elements, each element a row of n elements of $\boldsymbol{B} = \begin{bmatrix} \boldsymbol{B}_0 \\ \boldsymbol{B}_1 \\ \vdots \\ \boldsymbol{B}_{K-1} \end{bmatrix}$
- We will consider only the case of deterministic values of $\boldsymbol{C}$, as produced by the following algorithm:

$$\boldsymbol{C} \leftarrow 0; \text{for } \big(k = [0, K)\big)\ \boldsymbol{C} \leftarrow \boldsymbol{A}^k \times \boldsymbol{B}_k + \boldsymbol{C}$$

Anatomy of a high-performance matrix multiply





Performance bounds

- Let Δ be the latency (in cycles) of an elemental multiply-add operation
- Computing each element of $\boldsymbol{C}$, as described above, requires evaluating a dependence chain of K multiply-adds, and therefore takes time $K\Delta$

$$\boldsymbol{C} \leftarrow \boldsymbol{A}^k \times \boldsymbol{B}_k + \boldsymbol{C}$$

$$\Delta$$

- The computation of $\boldsymbol{C}$ requires mnK multiply-adds
- The maximum computation rate that can be sustained is

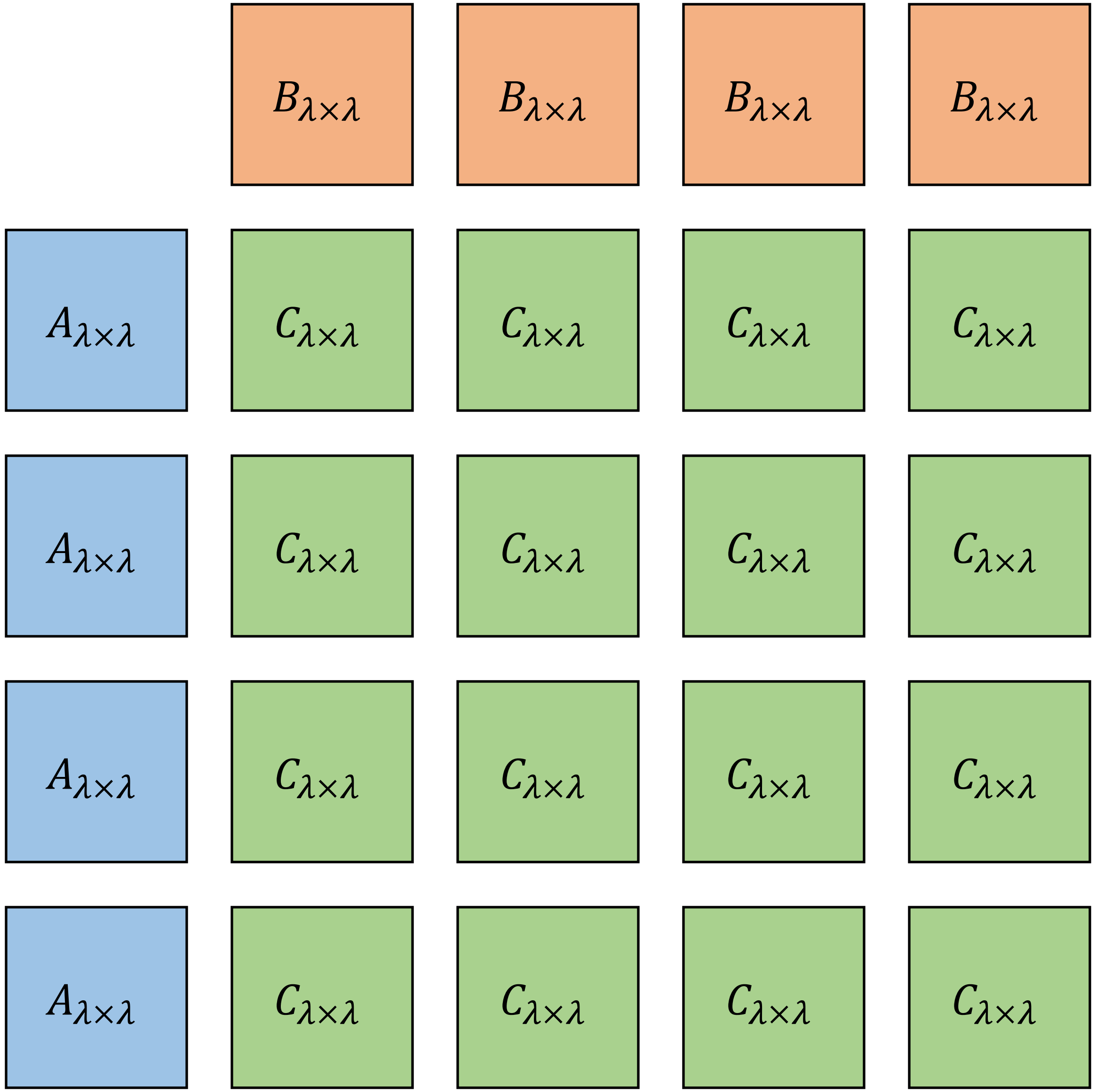
$$R = \frac{mnK}{K\Delta} = {}^{mn}/_{\Delta} \quad \text{madds/cycle}$$

- This sets an upper bound on performance, dictated by the size of the $\boldsymbol{C}$ panel that can be kept in registers (an architectural parameter) and the latency of a multiply-add (a design/technology parameter) – *Note*: integer arithmetic is more forgiving
- For modern server processor and 32-bit floating-point elements, a reasonable value of Δ is 4 cycles – the range $\Delta \in [2,8]$ should cover most cases

Some configurations (just to get a feel)

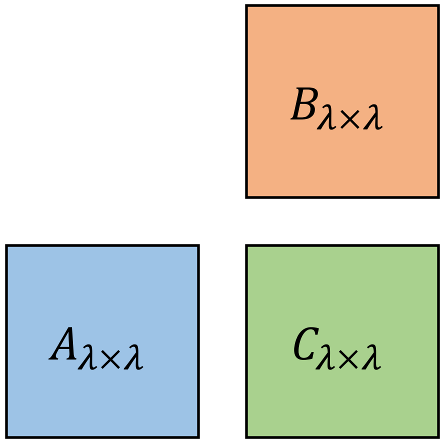
- Let the architected space consist of 32 vector registers of size L words (32-bit)
  - Then, $mn \leq 32L$ and the maximum computation rate is $R = {}^{32L}/_{4} = 8L \text{ madds/cycle}$ (single-precision floating-point)
- The maximum computation rate of a vector-based matrix facility, using vector registers only, scales with L
- For $L = 4, R = 32 \text{ madds/cycle}$
- For $L = 16, R = 128 \text{ madds/cycle}$
- *Note*: In the above, we use all architected space for $\boldsymbol{C}$, but in a load/store architecture we need to reserve space for $\boldsymbol{A}$ and $\boldsymbol{B}$, so cut the bound in half
  - $mn \leq 16L$ and the maximum computation rate is $R = {}^{16L}/_{4} = 4L \text{ madds/cycle}$
  - For $L = 4, R = 16 \text{ madds/cycle}$
  - For $L = 16, R = 64 \text{ madds/cycle}$
- *Note*: $4L$ madds/cycle is the performance equivalent of 4 vector pipes operating concurrently

Option A: 1 Matrix per vector register

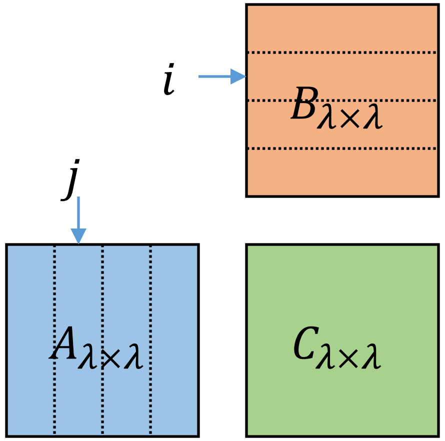


- An L-word vector register holds an $\lambda \times \lambda$ matrix, $\lambda = \sqrt{L}$
- 16 registers hold a $4\lambda \times 4\lambda$ panel of $\boldsymbol{C}$
- 4 registers hold a $4\lambda \times \lambda$ panel of A
- 4 registers hold a $\lambda \times 4\lambda$ panel of B
- We compute $\boldsymbol{C}_{4\lambda\times 4\lambda} \leftarrow \boldsymbol{A}_{4\lambda\times\lambda} \times \boldsymbol{B}_{\lambda\times 4\lambda} + \boldsymbol{C}_{4\lambda\times 4\lambda}$
- Total of $4\lambda \times 4\lambda \times \lambda = 16\lambda^3$ multiply-adds
- Minimum time = $\lambda\Delta$ cycles
- Maximum computation rate $R = \frac{16\lambda^3}{\lambda 4} = 4\lambda^2 = 4L$
- This is the upper bound with 16 registers for $\boldsymbol{C}$
- Total of $8\lambda^2$ elements loaded ($^{8\lambda}/_{\Delta}$ words/cycle)
- Computational intensity $\eta = \frac{16\lambda^3}{8\lambda^2} = 2\lambda$ madds/word
- This works for $L = 4, 16, 64, \ldots$ words
- Single- and double-precision are incompatible

Option A: Compute instructions

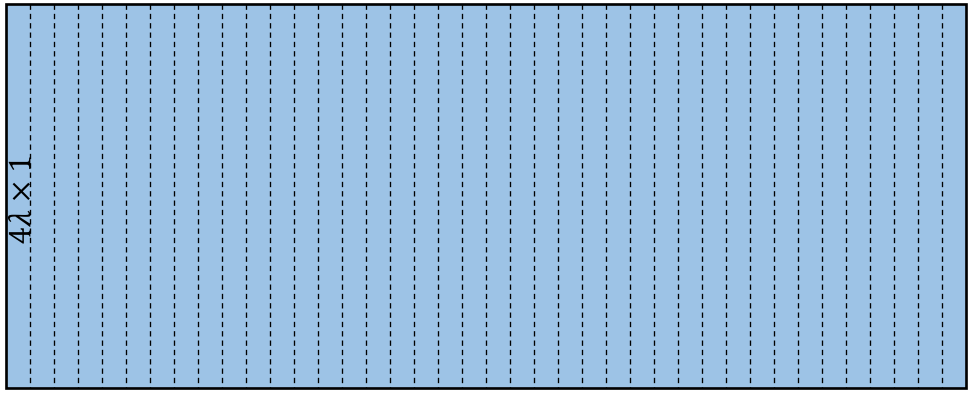


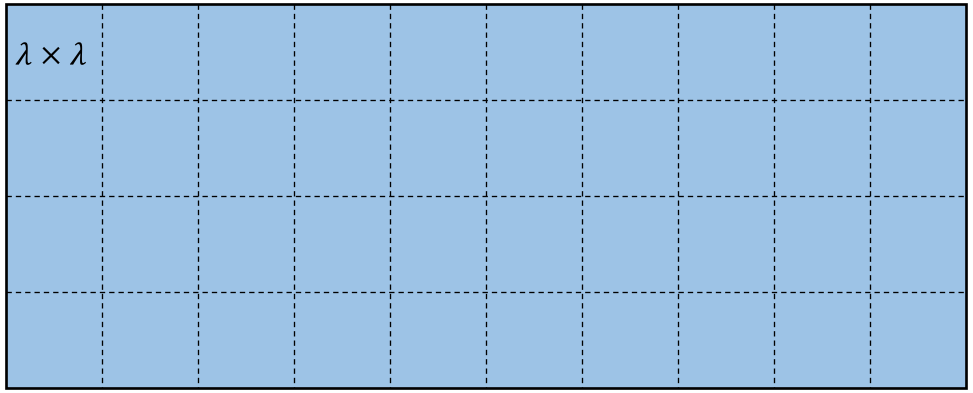
- rank-λ update (matrix multiply)
  - $\boldsymbol{C}_{\lambda\times\lambda} \leftarrow \pm\boldsymbol{A}_{\lambda\times\lambda} \times \boldsymbol{B}_{\lambda\times\lambda} \pm\boldsymbol{C}_{\lambda\times\lambda}$
  - Computations: λ^3 madds/instruction
  - Latency: $\lambda\Delta$
  - Must dispatch/issue 4 computational instructions/operations every λ cycles to achieve maximum computation rate ($4L$ madds/cycle)



- rank-1 update (outer product)
  - $\boldsymbol{C}_{\lambda\times\lambda} \leftarrow \pm\boldsymbol{A}^j \times \boldsymbol{B}_i \pm \boldsymbol{C}_{\lambda\times\lambda}$ (usually $i = j$)
  - Computations: λ^2 madds/instruction
  - Latency: Δ
  - Must dispatch/issue 4 computational instructions/operations every cycle to achieve maximum computation rate ($4L$ madds/cycle)
  - rank-λ update can be cracked into λ rank-1 updates

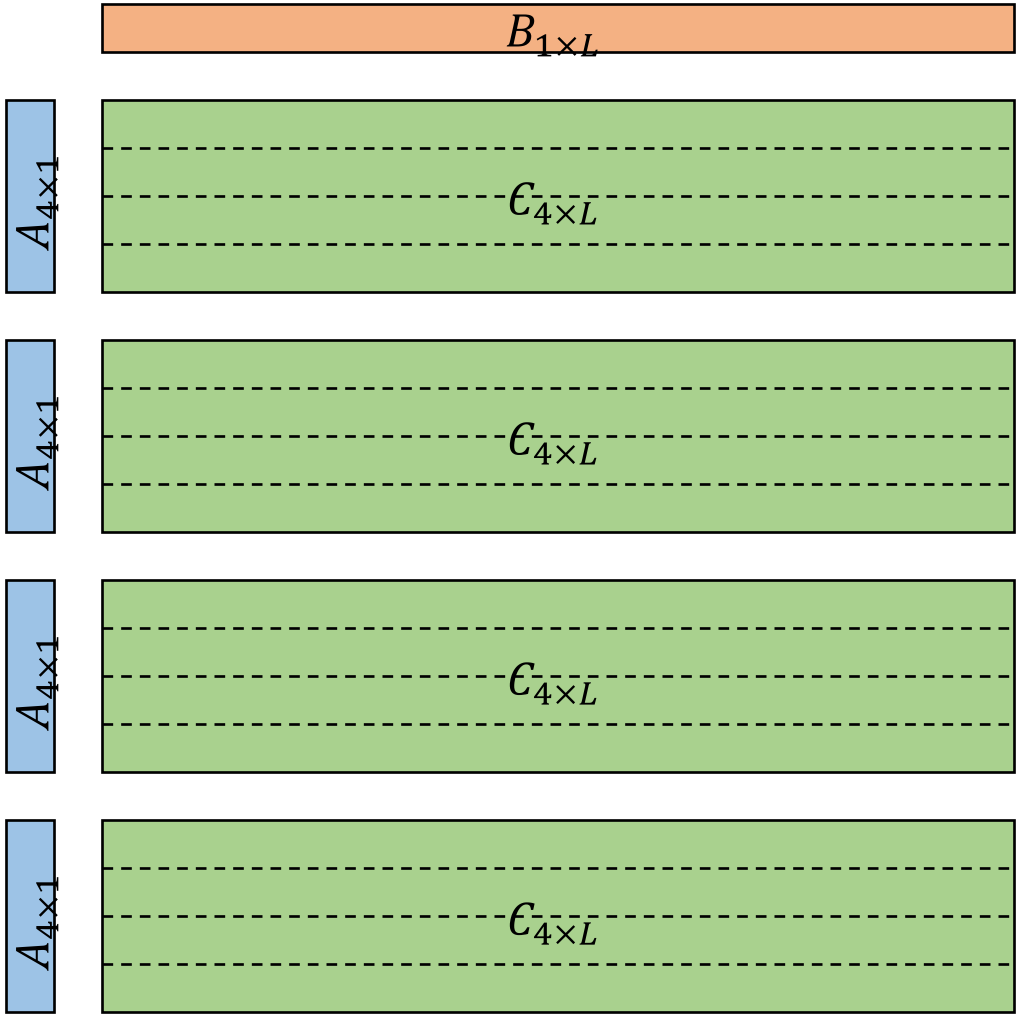
Option A: Software impact





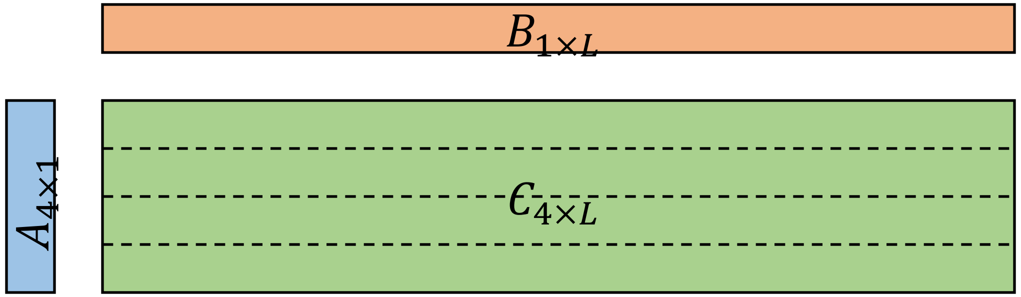
- Before the compute kernel is executed, the input matrices are *packed* to optimize streaming performance
- $\boldsymbol{A}$ panels are typically formatted as column-major matrices of shape $4\lambda \times K$
- $\boldsymbol{B}$ panels are typically formatted as row-major matrices of shape $K \times 4\lambda$
- For Option A to work, both $\boldsymbol{A}$ and $\boldsymbol{B}$ must be packed into $\lambda \times \lambda$ blocks
- Both the compute and packing kernels must be modified to support matrix operations
- $\boldsymbol{C}$ panel must also be reformatted for load/store

Option B: 1 Matrix in 4 vector registers



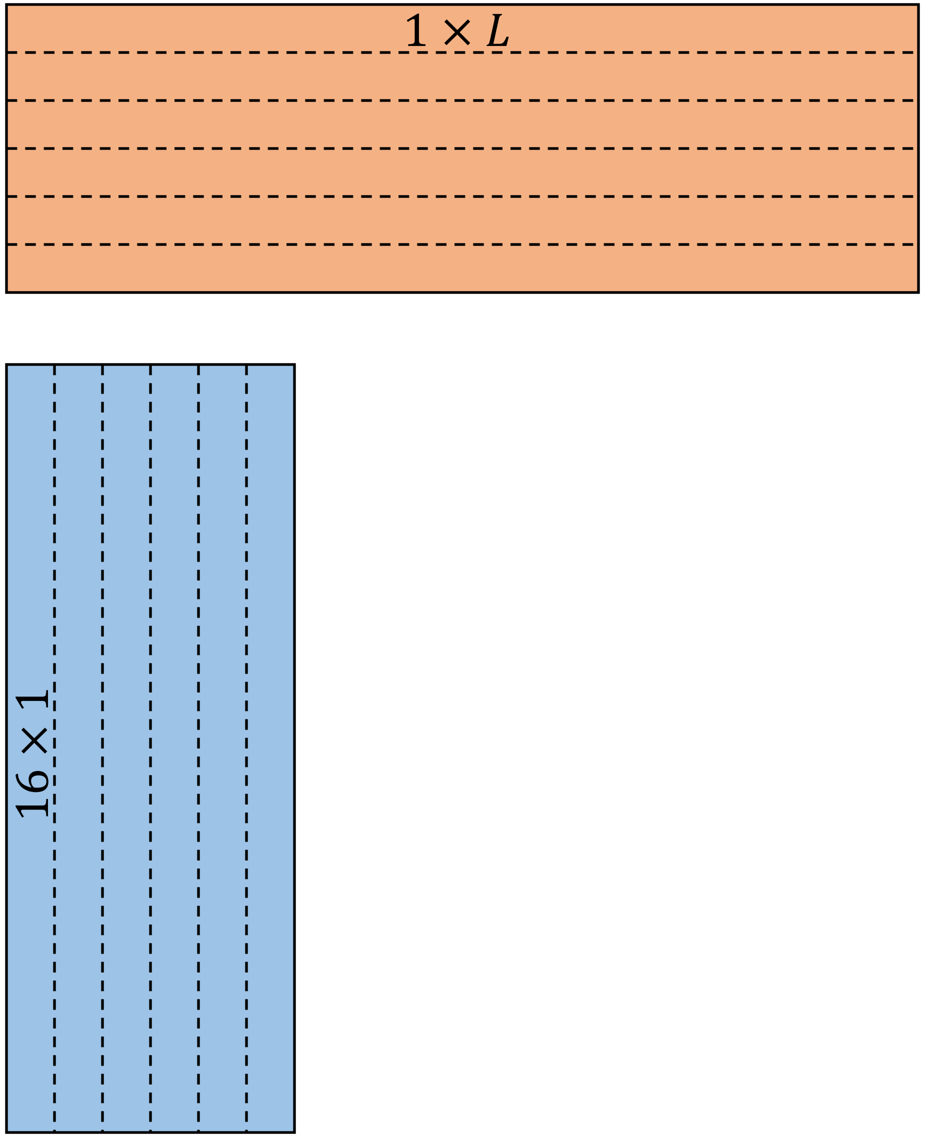
- An L-word vector register holds a row of $\boldsymbol{C}$
- 4 registers hold a $4 \times L$ panel of $\boldsymbol{C}$
- 16 registers hold a $16 \times L$ panel of $\boldsymbol{C}$
- An L-word vector register holds a row of $\boldsymbol{B}$
- "Some" combination of registers holds a column of $\boldsymbol{A}$
- We compute $\boldsymbol{C}_{16\times L} \leftarrow \boldsymbol{A}_{16\times 1} \times \boldsymbol{B}_{1\times L} + \boldsymbol{C}_{16\times L}$
- Total of $16L$ multiply-adds
- Minimum time = Δ cycles
- Maximum computation rate $R = \frac{16L}{\Delta} = 4L$ madds/cycle
- This is the upper bound with 16 registers for $\boldsymbol{C}$
- Total of $L + 16$ words loaded ($^{L+16}/_{\Delta}$ words/cycle)
- $\eta = \frac{16L}{L+16} = [\frac{16}{5}, 16)$ madds/word
- This works $\forall L \geq 4$ elements
- *Single-* and double-precision are compatible $\forall L \geq 8$

Option B: Compute instructions



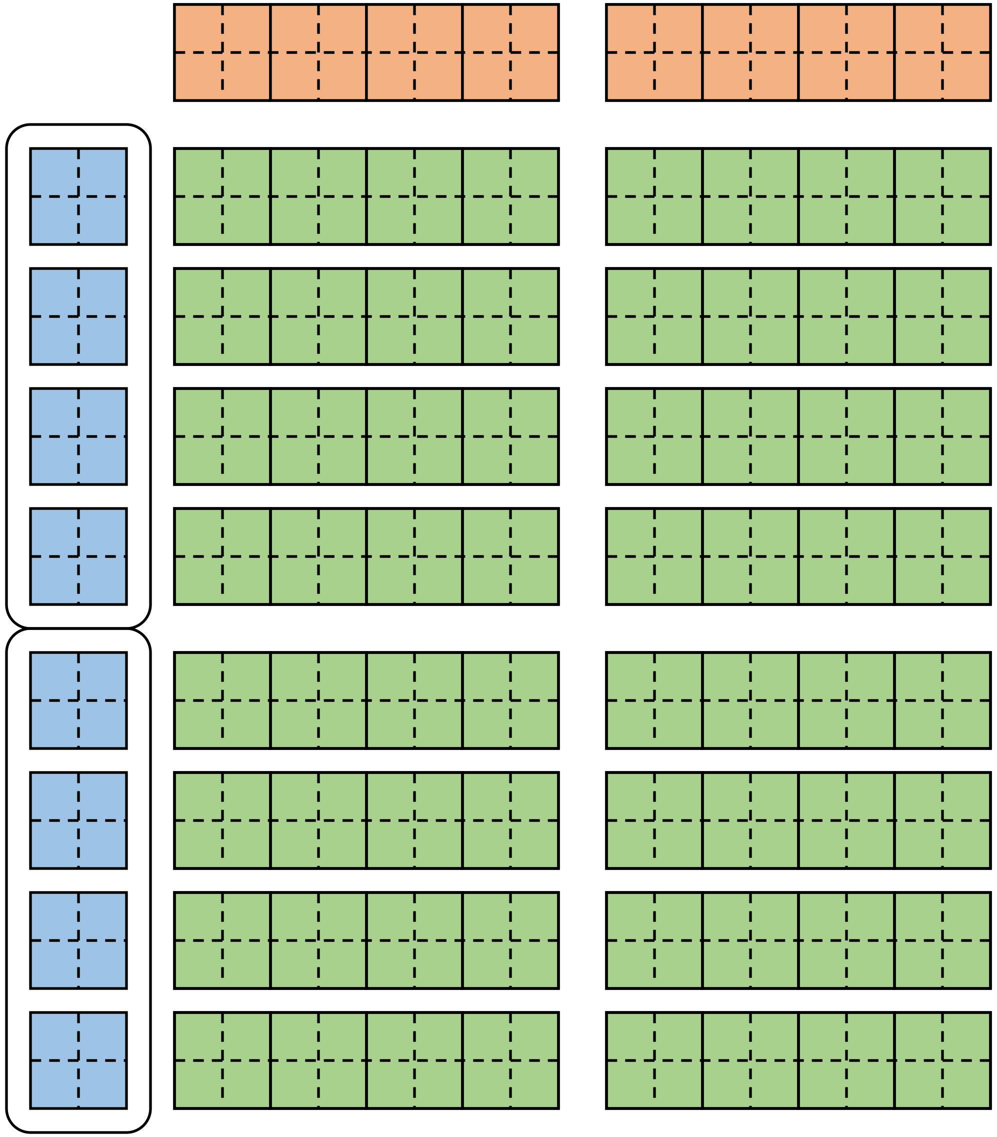
- rank-1 update (outer product)
  - $\boldsymbol{C}_{4\times L} \leftarrow \pm\boldsymbol{A}_{4\times 1} \times \boldsymbol{B}_{1\times L} \pm \boldsymbol{C}_{4\times L}$
  - Computations: $4L$ madds/instruction
  - Latency: Δ
  - Must dispatch/issue 1 computational instruction/operation every cycle to achieve maximum computation rate ($4L$ madds/cycle)

Option B: Software impact



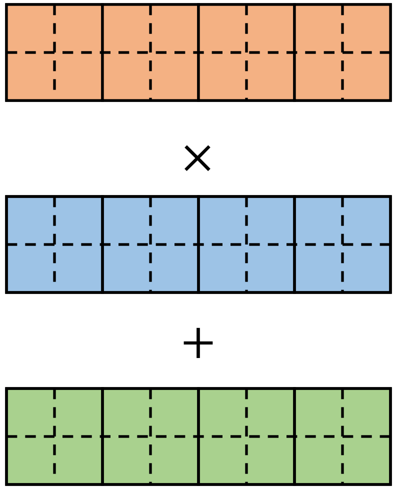
- Conventional BLAS formatting
  - $\boldsymbol{A}$ panel formatted as column-major $16 \times K$ matrix
  - $\boldsymbol{B}$ panel formatted as row-major $K \times L$ matrix
  - $\boldsymbol{C}$ panel does not have to be reformatted
- Easier pre/post-processing of rows/columns of matrices
  - Does not require reformatting data from/to vector format to/from matrix format
  - Although not that critical for matrix multiplication, insert/extract of rows from matrix registers from/to vector registers is useful in other algorithms (e.g., DFT)

Option C: Matrix as an element type



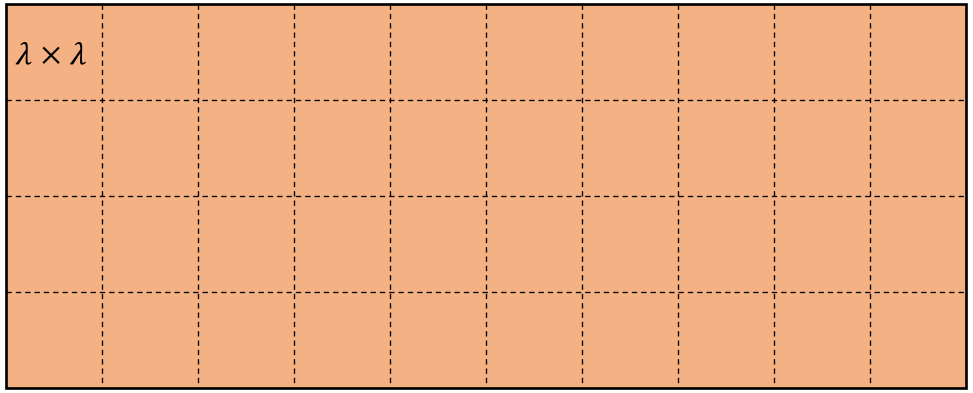
- We fix a λ and define a new “vector element type” – a $\lambda \times \lambda$ matrix of words (e.g., $\lambda = 2$)
- A vector register of length L holds ${}^{L}/_{\lambda^2}$ of these matrices (e.g., $L = 16, {}^{L}/_{\lambda^2} = 4$)
- “Some” number of registers hold an $8\lambda \times \lambda$ panel of $\boldsymbol{A}$ (8 $\lambda \times \lambda$ matrices)
- 2 registers hold a $\lambda \times {}^{2L}/_{\lambda}$ panel of $\boldsymbol{B}$ (${}^{2L}/_{\lambda^2}$ $\lambda \times \lambda$ matrices)
- 16 registers hold an $8\lambda \times {}^{2L}/_{\lambda}$ panel of C ($16L$ words)
- We compute $\boldsymbol{C}_{8\lambda \times {}^{2L}/_{\lambda}} \leftarrow \boldsymbol{A}_{8\lambda \times \lambda} \times \boldsymbol{B}_{\lambda \times {}^{2L}/_{\lambda}} + \boldsymbol{C}_{8\lambda \times {}^{2L}/_{\lambda}}$
- Total of $16L\lambda$ multiply-adds
- Minimum time = $\lambda\Delta$ cycles
- Maximum computation rate $R = \frac{16L}{\Delta} = 4L$ madds/cycle
- This is the upper bound with 16 registers for $\boldsymbol{C}$
- Total of $2L + 8\lambda^2$ words loaded (${}^{2L+8\lambda^2}/_{\lambda\Delta}$ words/cycle)
- $\eta = \frac{16L\lambda}{2L+8\lambda^2} = [\frac{8\lambda}{5}, 8\lambda)$ madds/word
- This works $\forall L \geq \lambda^2$ words
- *S*ingle- and double-precision are compatible $\forall L \geq 2\lambda^2$ words

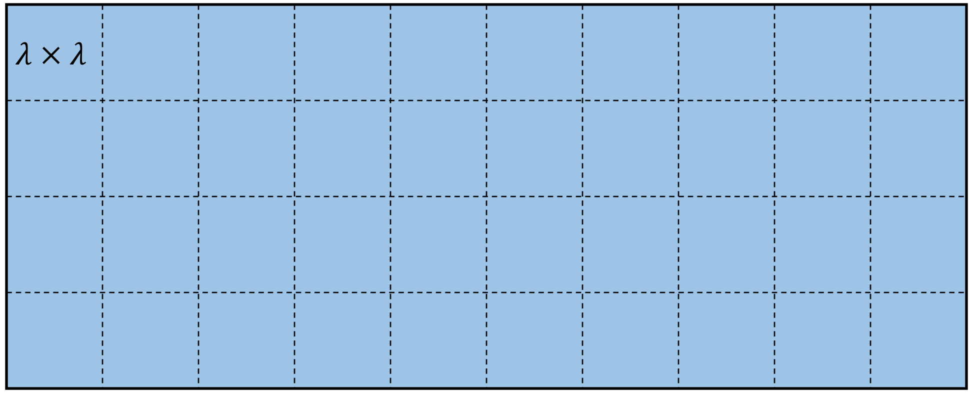
Option C: Compute instructions



- A vector of matrix multiplies
  - $\boldsymbol{C}_{\lambda\times\lambda} \leftarrow \pm\boldsymbol{A}_{\lambda\times\lambda} \times \boldsymbol{B}_{\lambda\times\lambda} \pm \boldsymbol{C}_{\lambda\times\lambda}$ ($^{L}/_{\lambda^2}$ times)
  - Computations: λL madds/instruction
  - Latency: $\lambda\Delta$
  - Must dispatch/issue $^{4}/_{\lambda}$ computational instructions/operations every cycle to achieve maximum computation rate $(4L$ madds/cycle)

Option C: Software impact





- For Option C to work, both $\boldsymbol{A}$ and $\boldsymbol{B}$ must be packed into $\lambda \times \lambda$ blocks – $\boldsymbol{A}$ in column-major, $\boldsymbol{B}$ in row-major
- Both the compute and packing kernels must be modified to support matrix operations
- $\boldsymbol{C}$ panel must also be reformatted for load/store

Conclusions

- The size of the architected register space imposes an upper bound on the performance of deterministic matrix multiply kernels
- If we use the architected scalable vector register space ($32L$ words) to hold the panel of $\boldsymbol{C}$ our performance upper bound, for a reasonable choice of latency parameters, is $8L$ madds/cycle – a more realistic value is $4L$ madds/cycle (single-precision floating-point)
- We advocate to pursue the vector register-based matrix extensions first, starting with three possible options:
    - Option A : 1 matrix/1 vector register ($R = 4L$ madds/cycle with 4 matrix pipes)
    - Option B : 1 matrix/n vector registers ($R = 4L$ madds/cycle with 1 matrix pipe)
    - Option C : n matrices/1 vector register ($R = 4L$ madds/cycle with $^4/_\lambda$ matrix pipes)