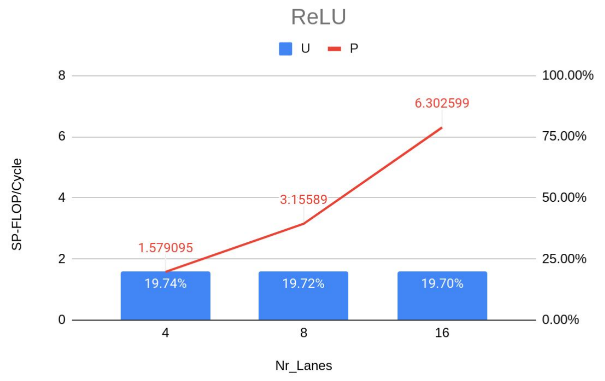
# Week 11

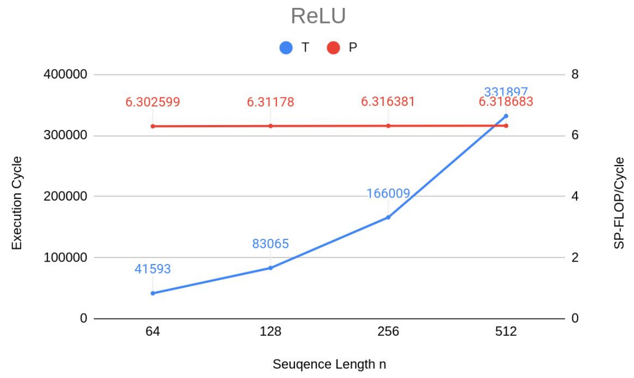
## BERT-base vs BERT-large

- d_model: 768 → 1024
- # heads: 12 → 16
- dk = d_model / # heads: 64 → 64 (self attention parameter size)
- d_ff = 4 * d_model: 3072 → 4096 (feed forward parameter size)

## Kernel Group I (U: utilization, P: SP-FLOP/Cycle, T: execution cycle)

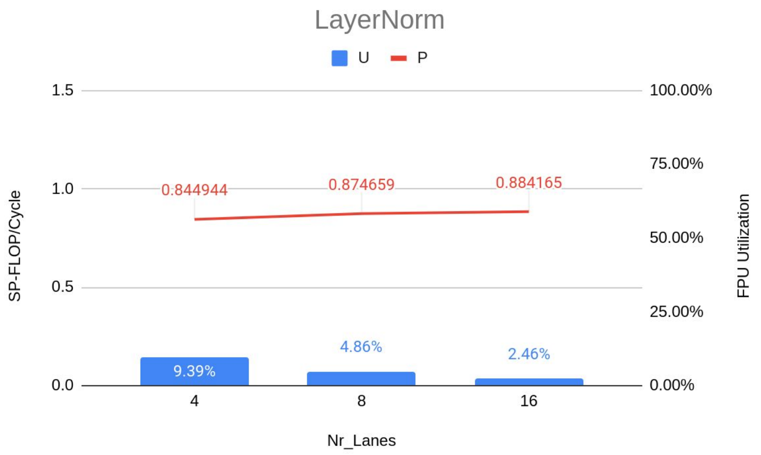
- ReLU, Dropout
- Vectorize columns with unit ld/st
- Input x: n x d_ff
  - Doubling Nr_Lanes:
    - U constant, P doubled, T halved
    - Max vl also doubled, d_ff is large enough, we can always use max vl.
  - Increasing d_ff:
    - U, P constant, T linearly increased
  - Doubling n:
    - U, P constant, T doubled
    - n is not the vector

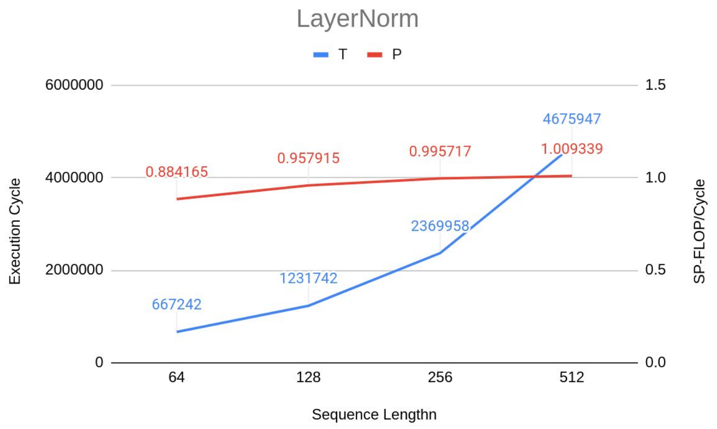




# Kernel Group II

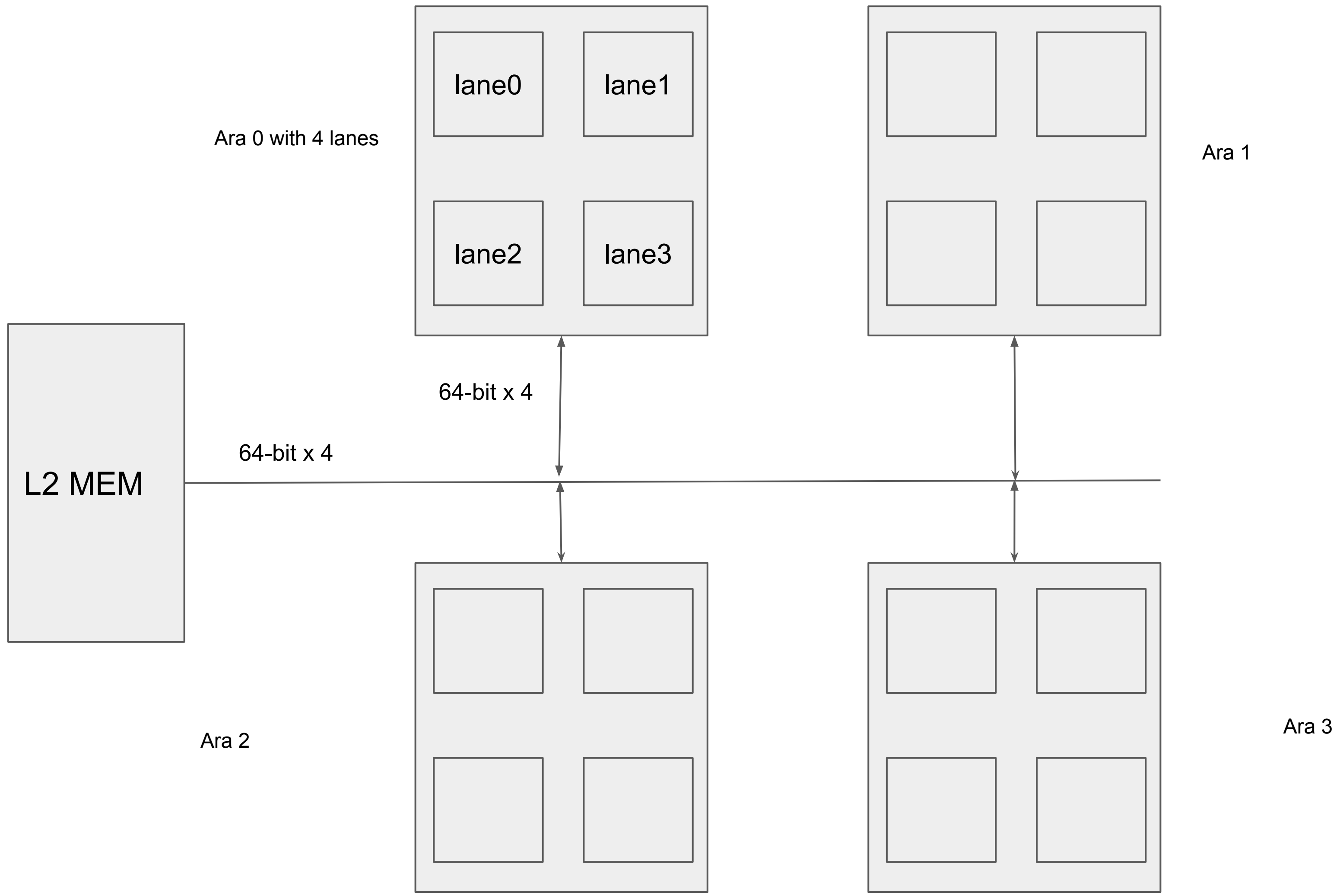
- LayerNorm, Softmax
- Vectorize rows using strid ld/st (stride size = column size)
- Input x: n x dk (Softmax), n x d_model (LayerNorm)
  - Doubling Nr_Lanes:
    - U halved, P, T constant
    - vl = n = 64
  - Increasing d_model:
    - U, P constant, T linearly increased
  - Doubling n:
    - U, P slightly increased, T almost doubled
    - vl doubled, we should expect doubled P, U, constant T
    - reason: strided load performs poorly, still memory bounded

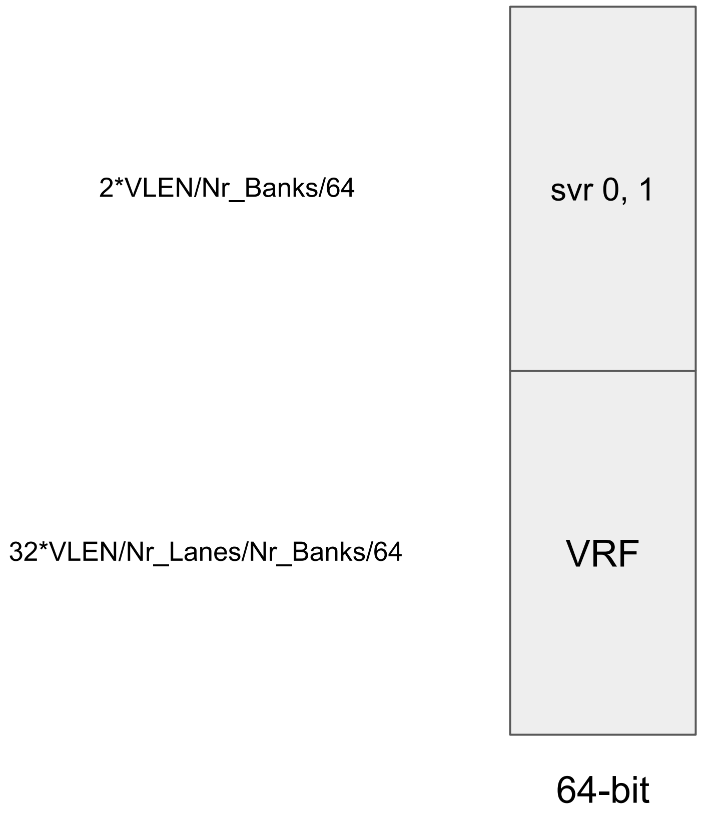




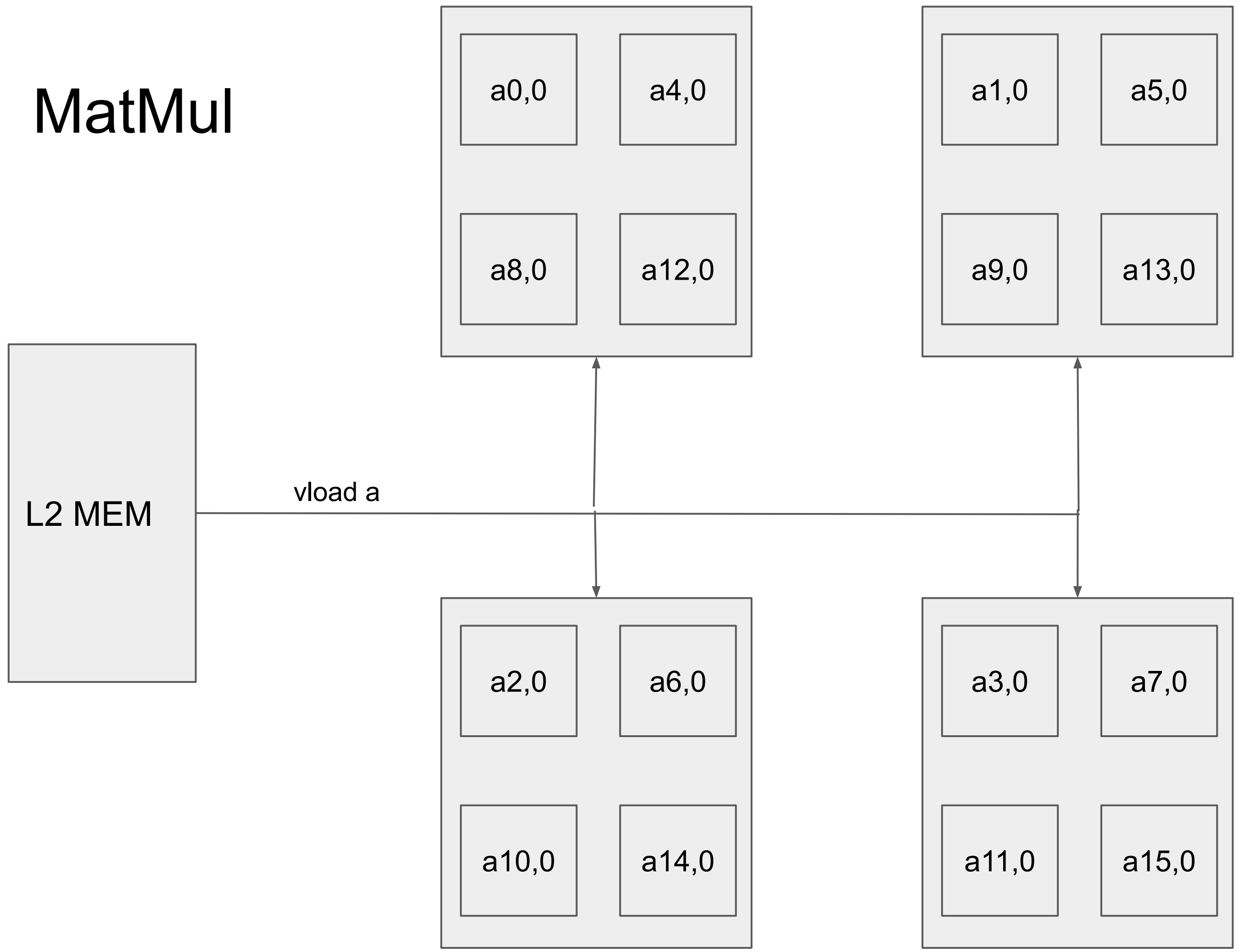
## Kernel Group III

- MatMul
- Feed forward layer (n***x***d_model * d_model***x***d_ff, n***x***d_ff * d_ff***x***d_model)
  - d_ff and d_model large enough
  - Utilization > 90% (4/8/16 lanes)
- Self attention (n***x***d_model * d_model***x***dk)
  - Doubling Nr_lanes:
    - P, T constant, U halved
  - Increasing d_model:
    - P, U decreased

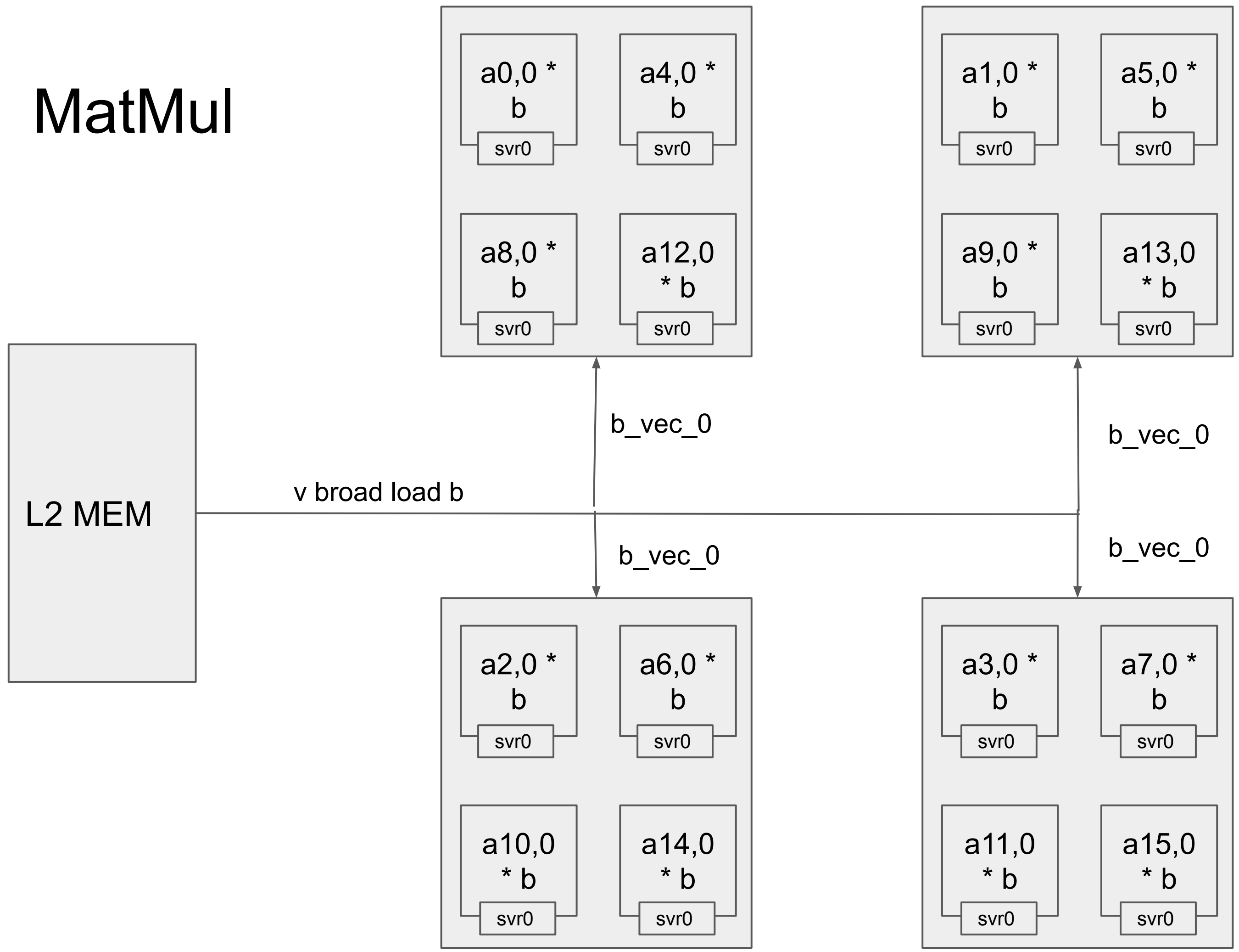

Ara 0 with 4 lanes
lane0
lane1
lane2
lane3
Ara 1
64-bit x 4
64-bit x 4
L2 MEM
Ara 2
Ara 3


2*VLEN/Nr_Banks/64
svr 0, 1
32*VLEN/Nr_Lanes/Nr_Banks/64
VRF
64-bit

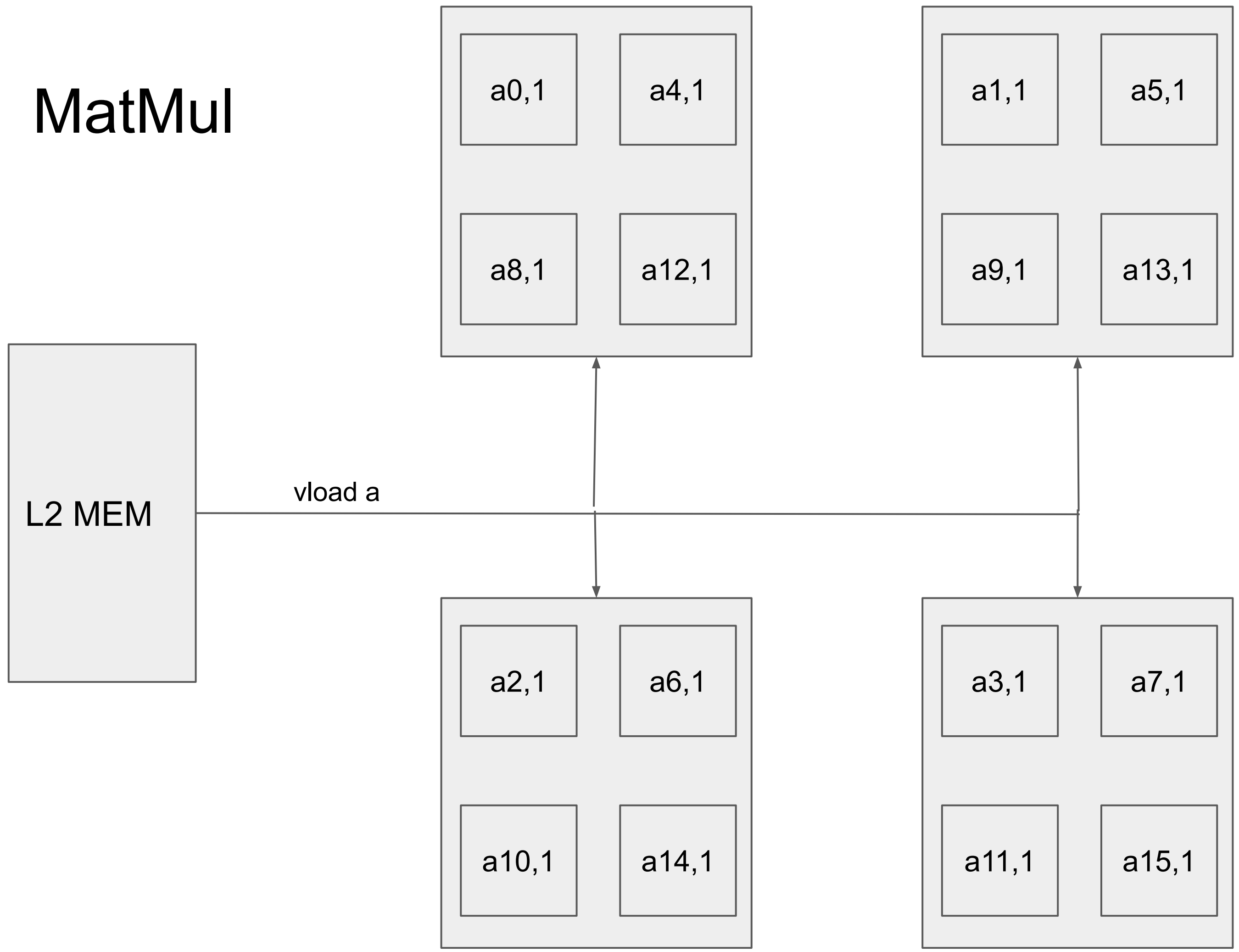
# MatMul


a0,0
a4,0
a8,0
a12,0
a1,0
a5,0
a9,0
a13,0
L2 MEM
vload a
a2,0
a6,0
a10,0
a14,0
a3,0
a7,0
a11,0
a15,0

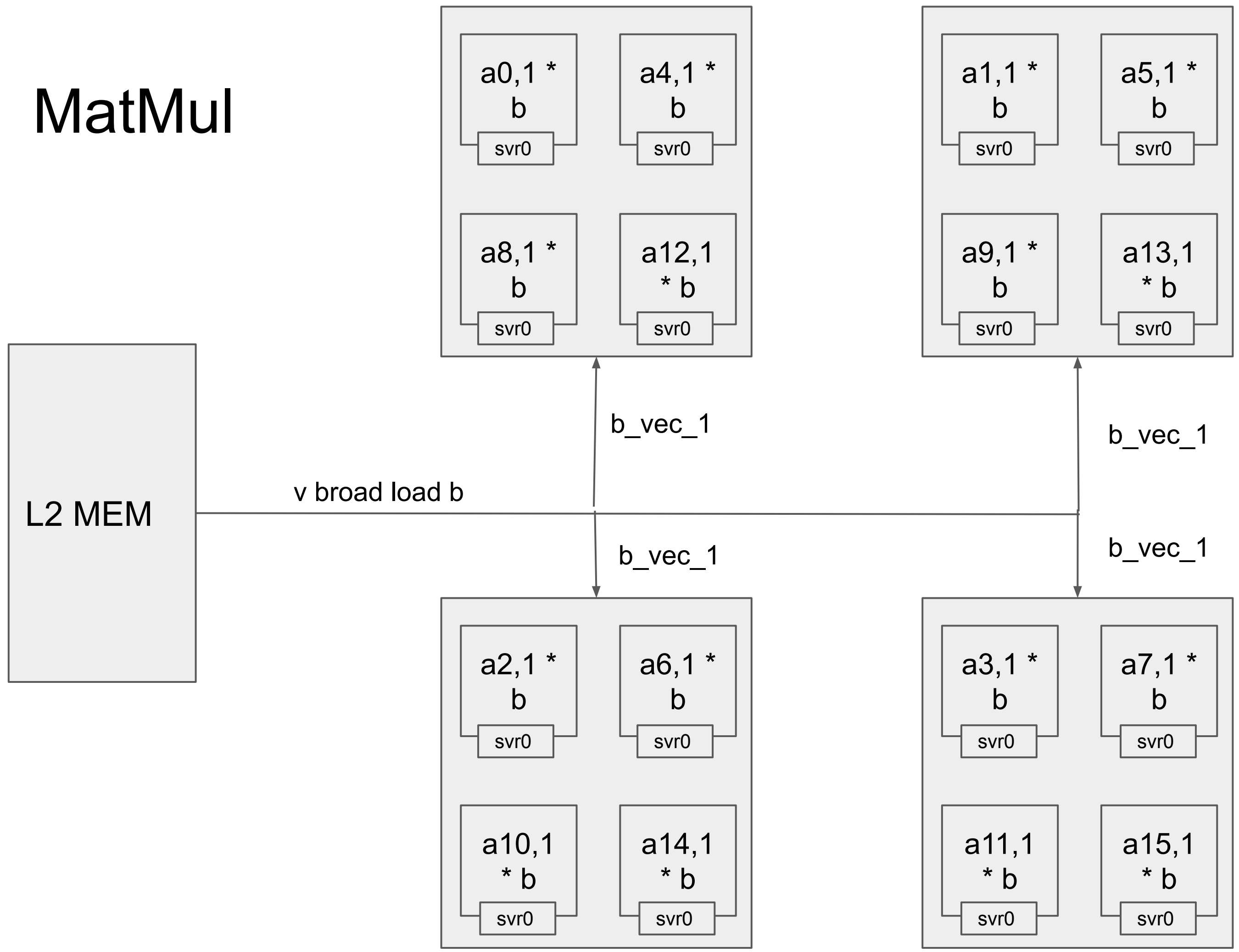
# MatMul


a0,0 * b
svr0
a4,0 * b
svr0
a8,0 * b
svr0
a12,0 * b
svr0
a1,0 * b
svr0
a5,0 * b
svr0
a9,0 * b
svr0
a13,0 * b
svr0
L2 MEM
v broad load b
b_vec_0
b_vec_0
b_vec_0
b_vec_0
a2,0 * b
svr0
a6,0 * b
svr0
a10,0 * b
svr0
a14,0 * b
svr0
a3,0 * b
svr0
a7,0 * b
svr0
a11,0 * b
svr0
a15,0 * b
svr0

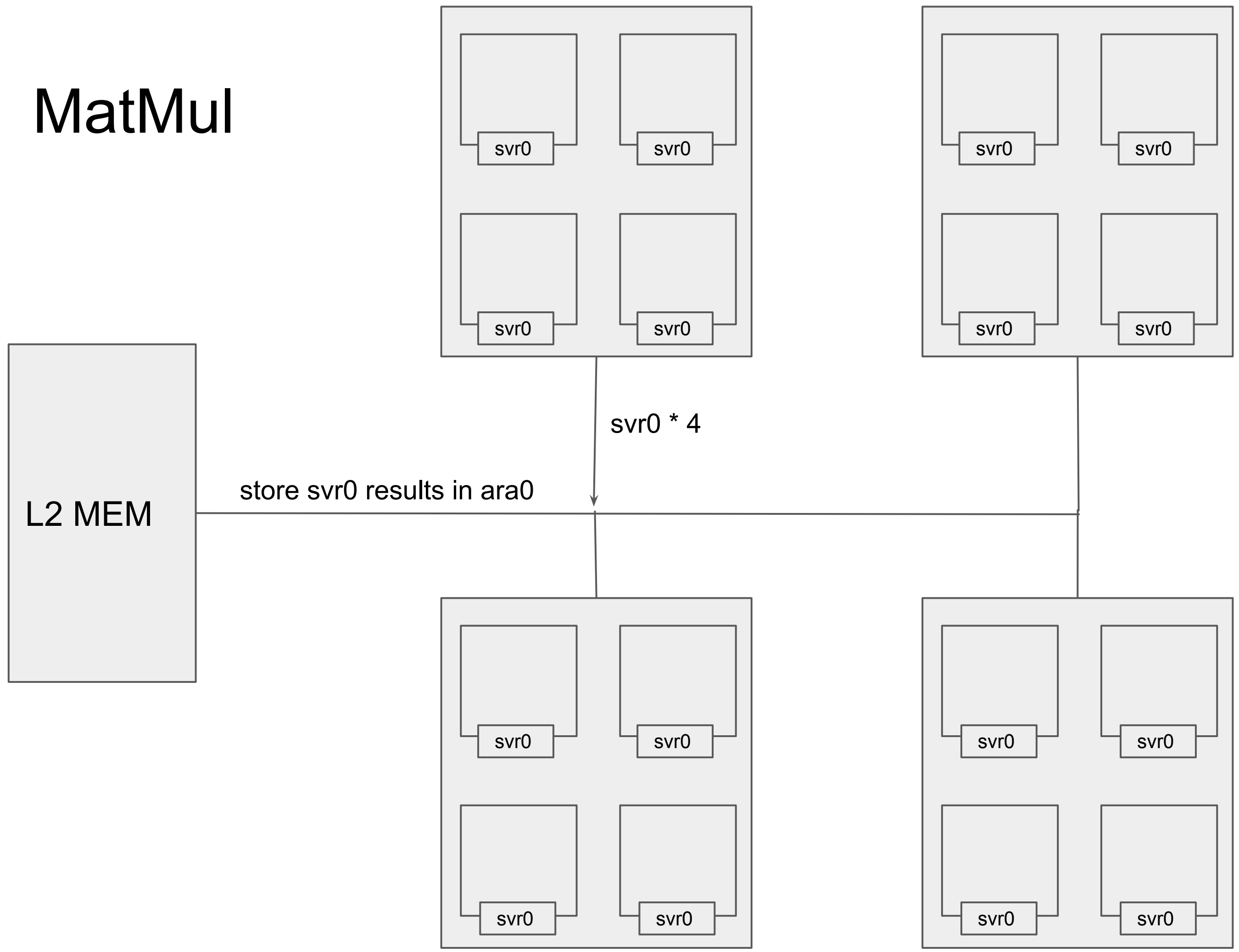
# MatMul


a0,1
a4,1
a8,1
a12,1
a1,1
a5,1
a9,1
a13,1
L2 MEM
vload a
a2,1
a6,1
a10,1
a14,1
a3,1
a7,1
a11,1
a15,1

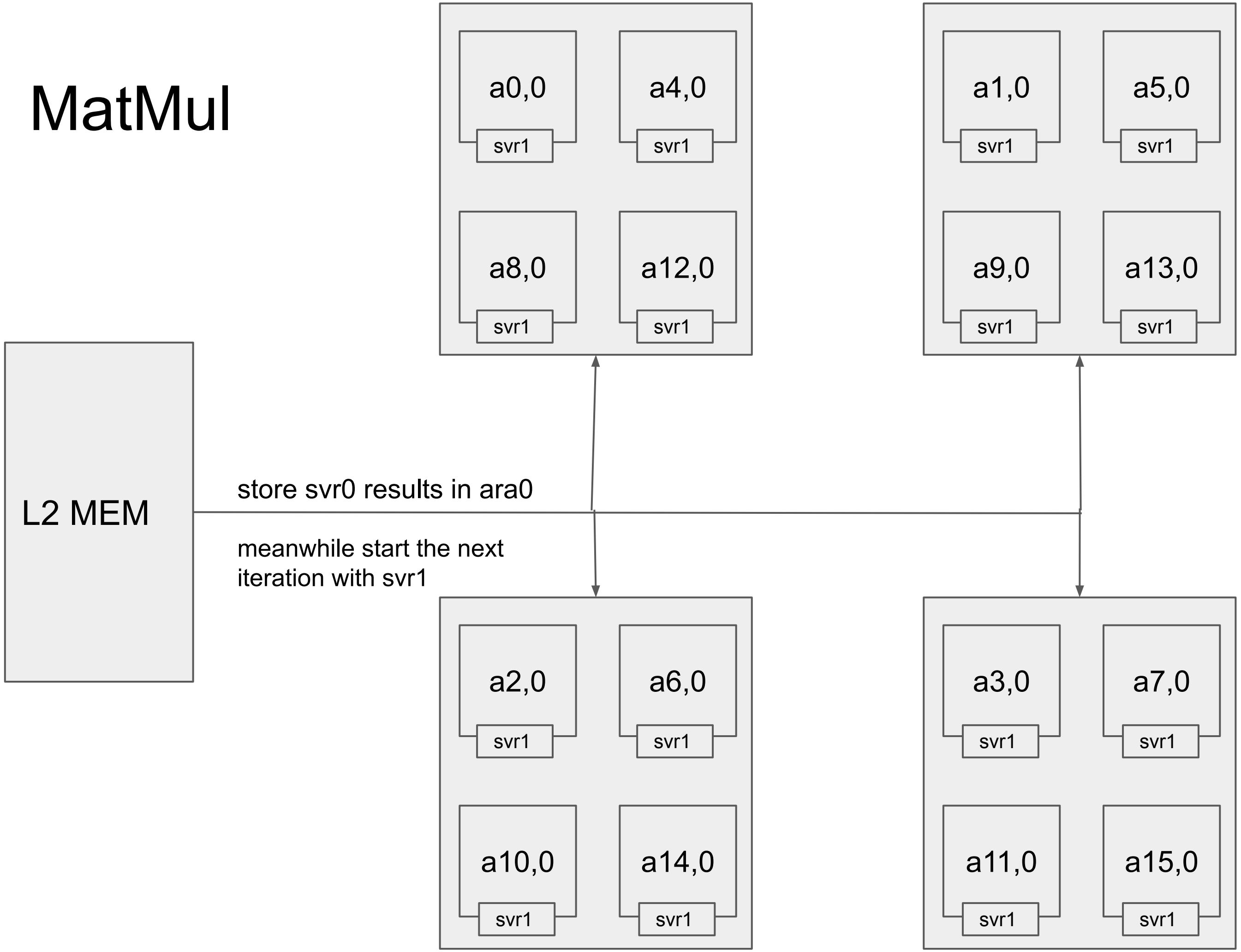
# MatMul


a0,1 * b
svr0
a4,1 * b
svr0
a8,1 * b
svr0
a12,1 * b
svr0
a1,1 * b
svr0
a5,1 * b
svr0
a9,1 * b
svr0
a13,1 * b
svr0
L2 MEM
v broad load b
b_vec_1
b_vec_1
b_vec_1
b_vec_1
a2,1 * b
svr0
a6,1 * b
svr0
a10,1 * b
svr0
a14,1 * b
svr0
a3,1 * b
svr0
a7,1 * b
svr0
a11,1 * b
svr0
a15,1 * b
svr0

# MatMul


svr0
svr0
svr0
svr0
svr0
svr0
svr0
svr0
svr0 * 4
L2 MEM
store svr0 results in ara0
svr0
svr0
svr0
svr0
svr0
svr0
svr0
svr0

# MatMul



a0,0
svr1
a4,0
svr1
a8,0
svr1
a12,0
svr1
a1,0
svr1
a5,0
svr1
a9,0
svr1
a13,0
svr1
L2 MEM
store svr0 results in ara0
meanwhile start the next iteration with svr1
a2,0
svr1
a6,0
svr1
a10,0
svr1
a14,0
svr1
a3,0
svr1
a7,0
svr1
a11,0
svr1
a15,0
svr1