Batched Linear Algebra Problems on Hardware Accelerators Based on GPUs

Tingxing(Tim) Dong

Lunch talk August 28th, 2015

Innovative Computing Laboratory
University of Tennessee, Knoxville

Motivations: application driven

- High-order FEMs,

  100x100, 200x200, batched GEMM, GEMV
- Astrophysics, subsurface transportation simulation,

  140x140 batched LU
- Radar signal processing,

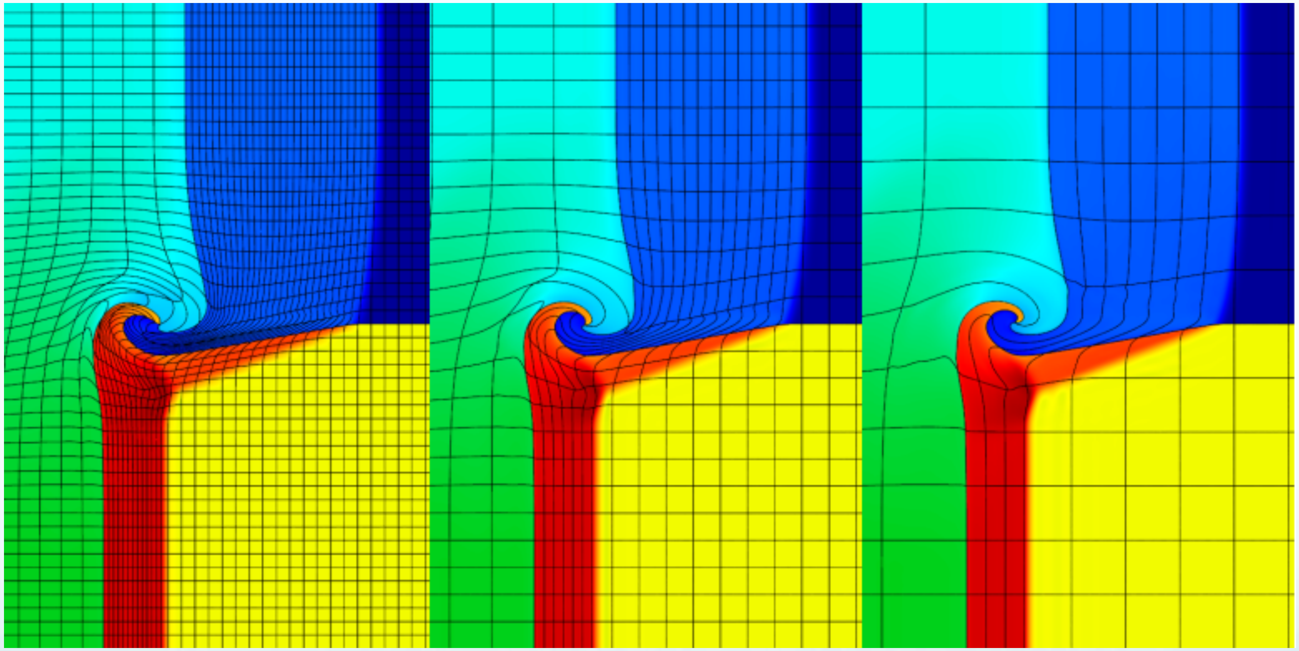
  200x200 batched QR
- Computer vision

  batched Cholesky
- Further accelerating CA sparse iterative solvers

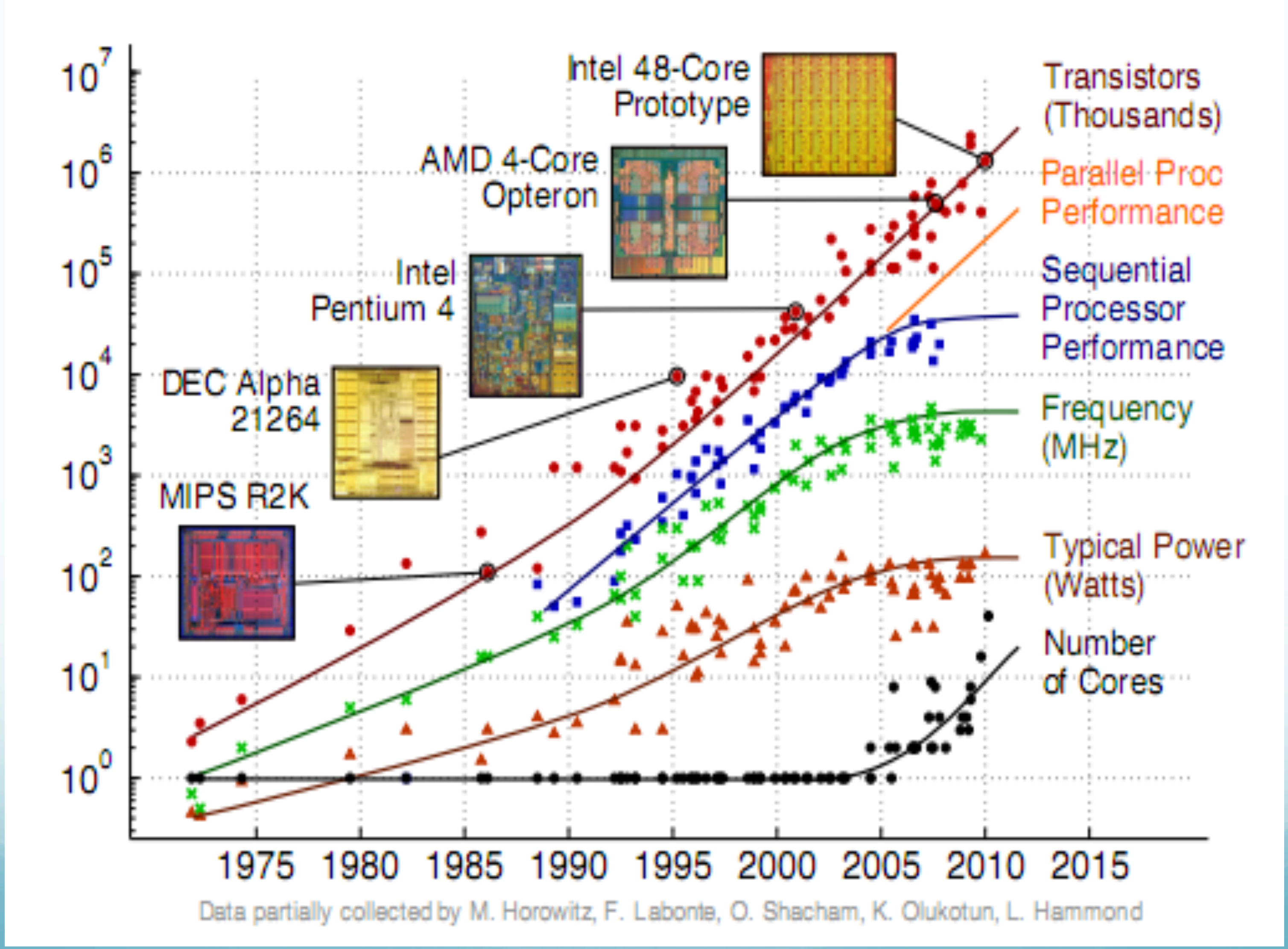
  (with a new mixed-precision orthgonalization technique)

Example: High order method FEM

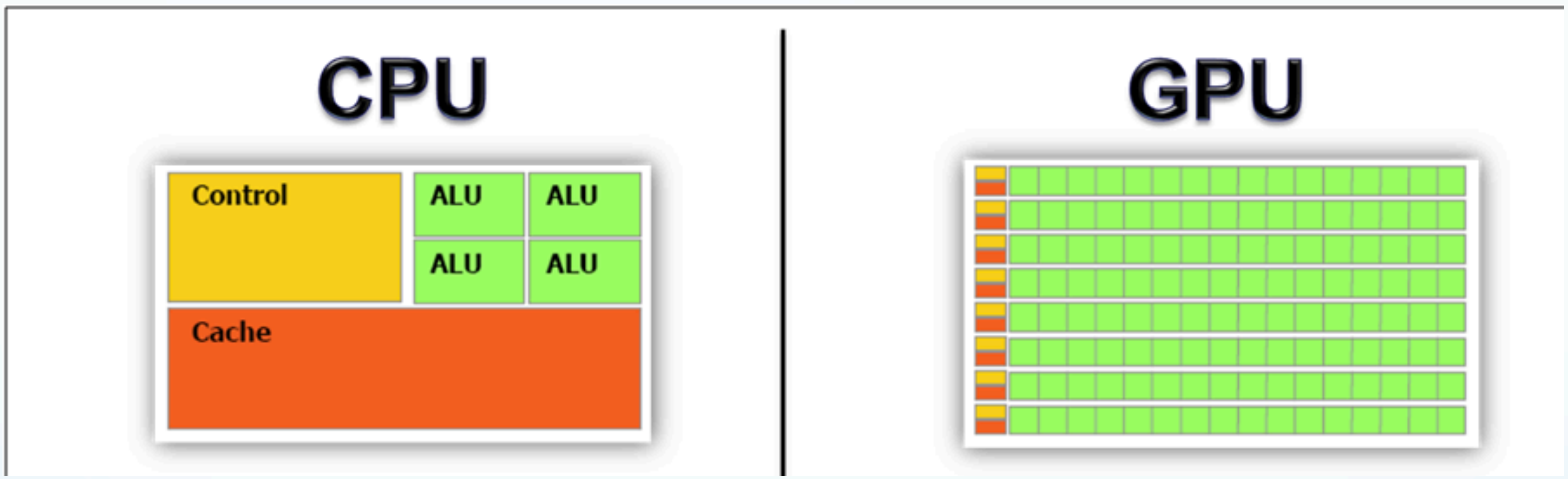
Q8-Q7	Q4-Q3	Q2-Q1



Motivations: living with multi/many cores:



The difference between CPU and GPU

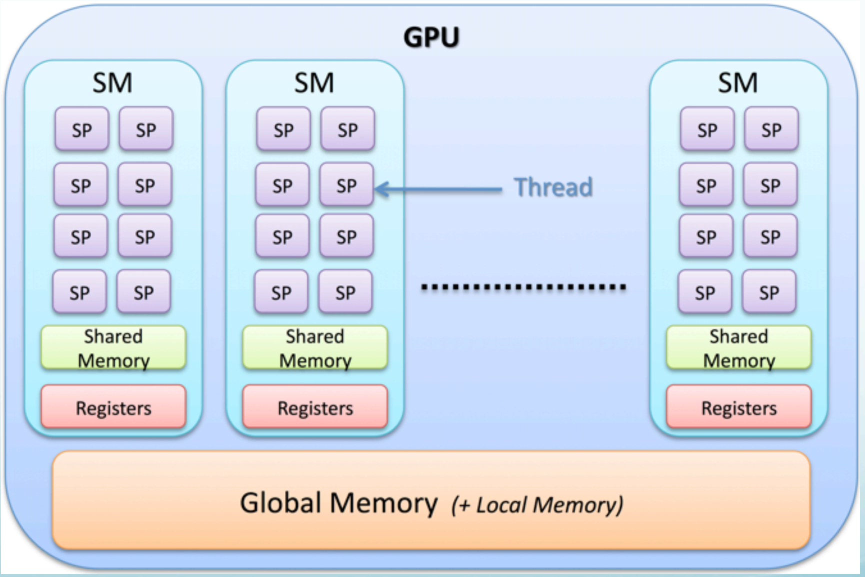


GPU: Many arithmetic units: computing is cheap.

High Latency: memory access is expensive

Programming on Accelerators: SIMT

- Single Instruction Multiple Threads (SIMT)
- Fine grained data parallelism
- Multi-threading to hide latency



Related work

- **CUBLAS 4.1, (January, 2012) MKL 11.3 Beta (April, 2015)**

  Batched GEMM

- **CUBLAS 5.0, October, 2012**

  Batched LU (M=N <= 32 * 32)

  Batched TRSM (<= 32*32)

- **CUBLAS 5.5, July 2013**

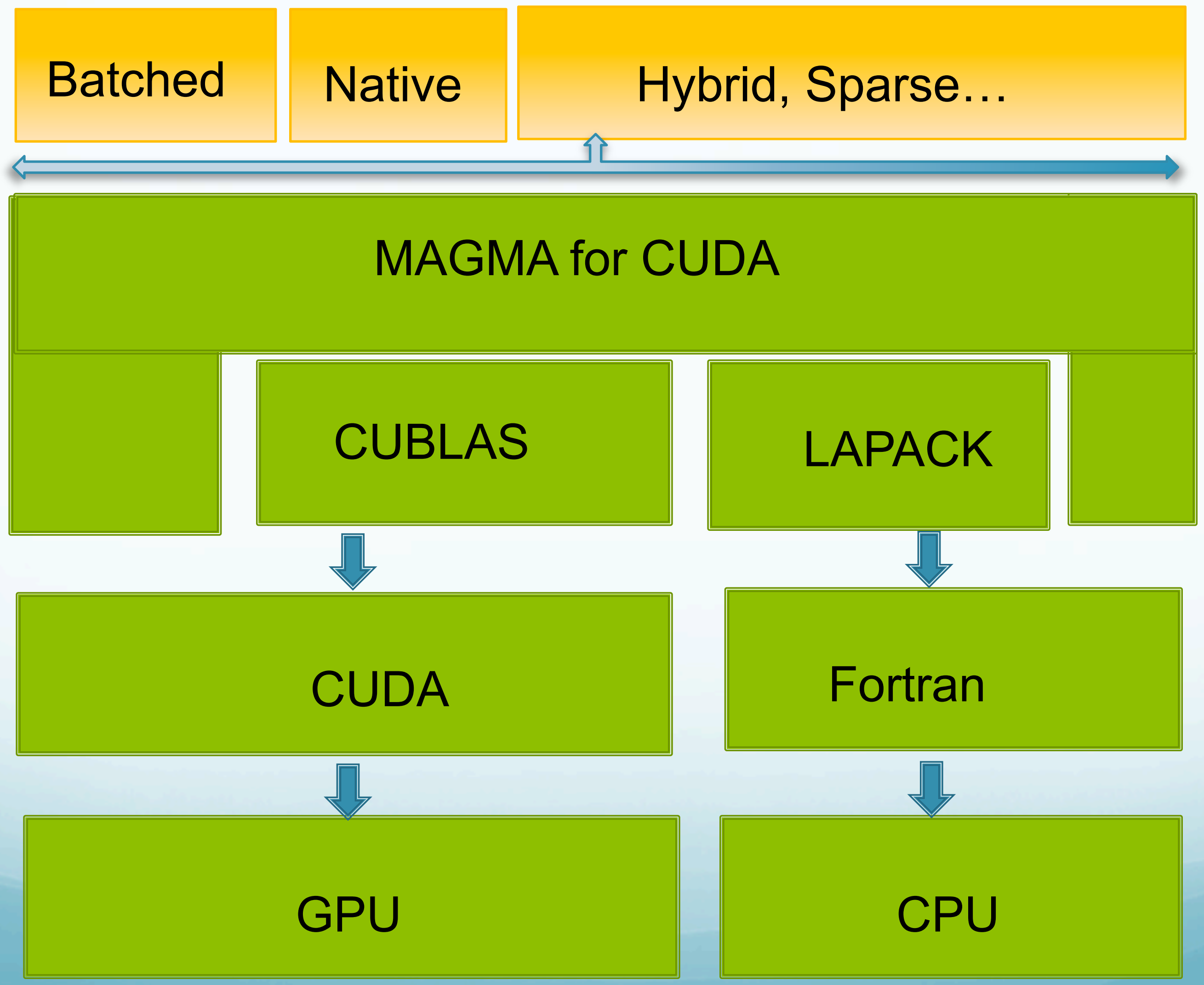
  Batched LU (M=N)

- **CUBLAS 6.5, August, 2014**

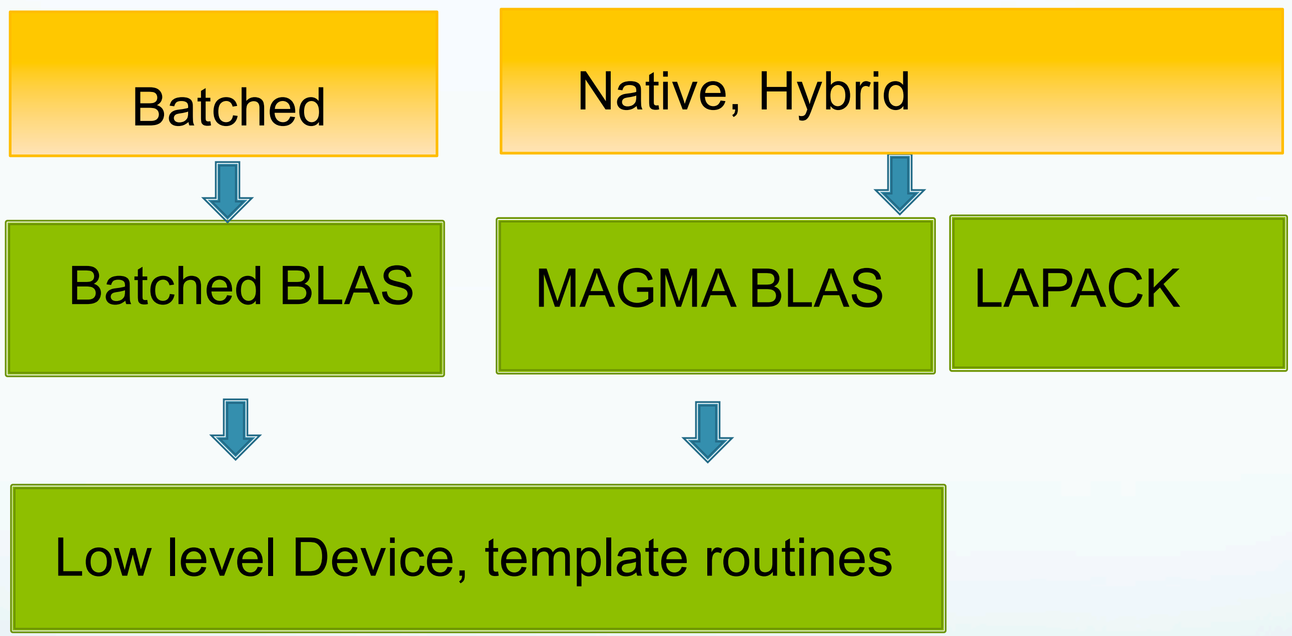
  Batched QR

  Batched TRSM

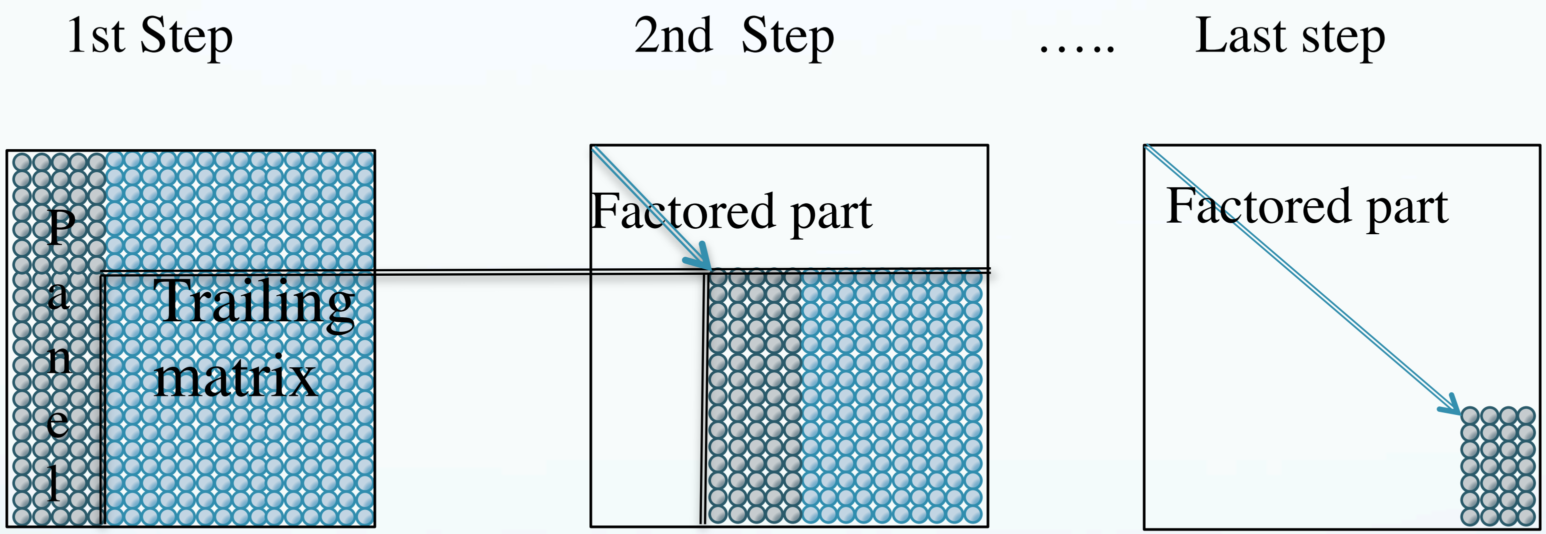
MAGMA Software Stack



Design of Batched in MAGMA



One-sided factorization:



- Panel (skinny) factorization : memory bound
- Trailing matrix update: more data parallel
- Matrix may not be square

Two sided factorization: Bi-diagonal

$$A = UBV^T$$

- U, V are orthogonal

$$U^TU = UU^T = I$$

$$V^TV = VV^T = I$$

$$B = \begin{pmatrix} \gamma_1 & \phi_2 & 0 & \cdot & \cdot & \cdot & \cdot \\ 0 & \gamma_2 & \phi_3 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \gamma_3 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot & \cdot & \gamma_{n-1} & \phi_n \\ 0 & \cdot & \cdot & \cdot & 0 & \cdot & \gamma_n \end{pmatrix}.$$

Householder transformation

$$V = \begin{bmatrix} x \\ x \\ x \\ x \\ x \end{bmatrix} \quad \text{Exiting H, let } H\,V = \begin{bmatrix} \alpha \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \alpha\, e$$

where, H = I - beta uu*

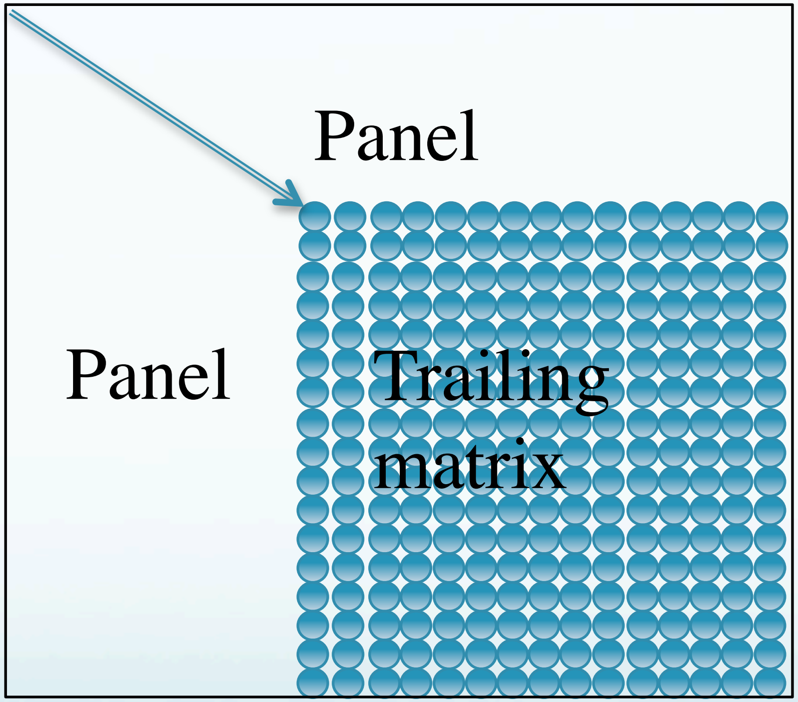
Sequential algorithm

$$U_1^*A = \begin{bmatrix} \mathbf{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ \mathbf{0} & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ \mathbf{0} & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ \mathbf{0} & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ \mathbf{0} & \mathbf{x} & \mathbf{x} & \mathbf{x} \end{bmatrix} \longrightarrow U_1^*AV_1 = \begin{bmatrix} x & \mathbf{x} & \mathbf{0} & \mathbf{0} \\ 0 & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{x} & \mathbf{x} & \mathbf{x} \end{bmatrix}$$

$$\longrightarrow U_2^*U_1^*AV_1 = \begin{bmatrix} x & x & 0 & 0 \\ 0 & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{0} & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{0} & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{0} & \mathbf{x} & \mathbf{x} \end{bmatrix} \longrightarrow U_2^*U_1^*AV_1V_2 = \begin{bmatrix} x & x & 0 & 0 \\ 0 & x & \mathbf{x} & \mathbf{0} \\ 0 & 0 & \mathbf{x} & \mathbf{x} \\ 0 & 0 & \mathbf{x} & \mathbf{x} \\ 0 & 0 & \mathbf{x} & \mathbf{x} \end{bmatrix}$$

$$\longrightarrow U_3^*U_2^*U_1^*AV_1V_2 = \begin{bmatrix} x & x & 0 & 0 \\ 0 & x & x & 0 \\ 0 & 0 & \mathbf{x} & \mathbf{x} \\ 0 & 0 & \mathbf{0} & \mathbf{x} \\ 0 & 0 & \mathbf{0} & \mathbf{x} \end{bmatrix} \longrightarrow U_4^*U_3^*U_2^*U_1^*AV_1V_2 = \begin{bmatrix} x & x & 0 & 0 \\ 0 & x & x & 0 \\ 0 & 0 & x & x \\ 0 & 0 & 0 & \mathbf{x} \\ 0 & 0 & 0 & \mathbf{0} \end{bmatrix} = B.$$

Blocked algorithm



- Panel: memory bound
- Trailing matrix update: GEMM

Analysis: Bi-diagonal factorization

- 50% operations (ops) as Level 2 BLAS
- Level 2 BLAS

  -- only $O(n^2)$ ops with $O(n^2)$ input data

  --memory bounded

  --does not scale with cores

- The other 50% is Level 3 BLAS GEMM

  -- $O(n^3)$ ops with $O(n^2)$ input data

Analysis: Bi-diagonal factorization

- 4/3 n^3 operations (ops) are GEMV

  GEMVN: 2/3 n^3

  GEMVT: 2/3 n^3

- 4/3 n^3 operations (ops) are GEMM

- Total 8/3 n^3 ops

Amdahl's law

- Amdahl's law tells the maximum speedup achieved on multiple processors compared to one processor

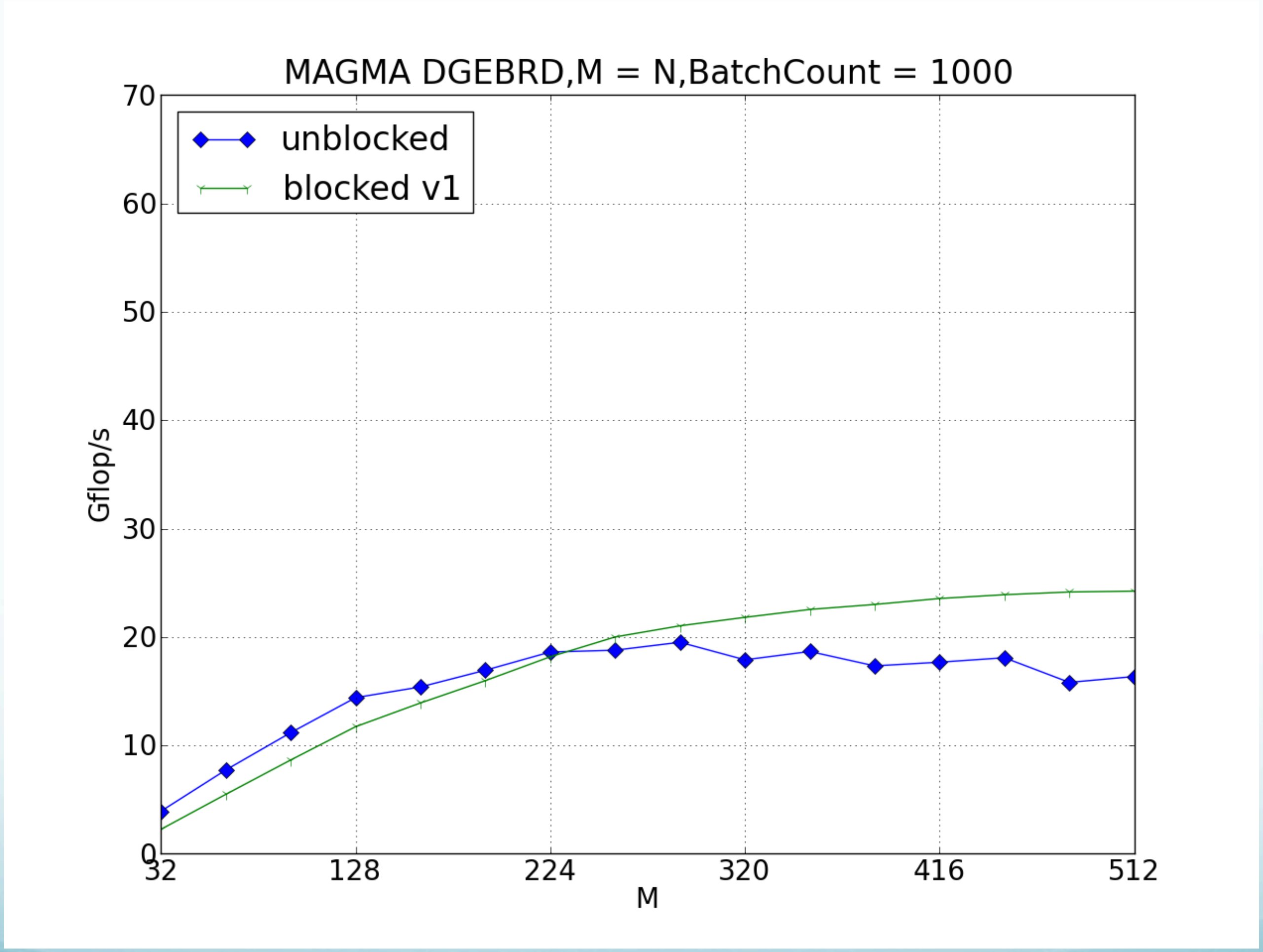
$$SpeedUp = \frac{1}{(1-F)+\frac{F}{N}}$$

- e.g

  if F=0.5, N=∞; then, speedup up to 2

- Indicates Bi-diagonal runs up to 2x speed of level 2 BLAS

Batched GEBRD on K40c



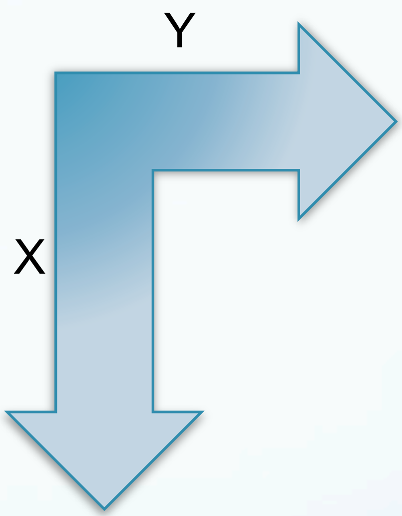
GEMV in GEBRD

- 13 GEMV calls in each step of GEBRD
- 2 Square GEMV: big (deal with trailing matrix)
- 11 Fat, Tall GEMV: small, row/column = NB

Num of calls	Fat matrix	Tall matrix	Square
GEMV Nontranspose	1	6	1(Big)
GEMV transpose	2	2	1(Big)

Redesign: templating GEMV

Name	DIM_X	DIM_Y	TILE_X	TILE_Y
GEMVN	✔	✔	✔	N/A
GEMVT	✔	✔	N/A	✔



Optimization 1: Auto-tuning GEMV

- Source level code tuning

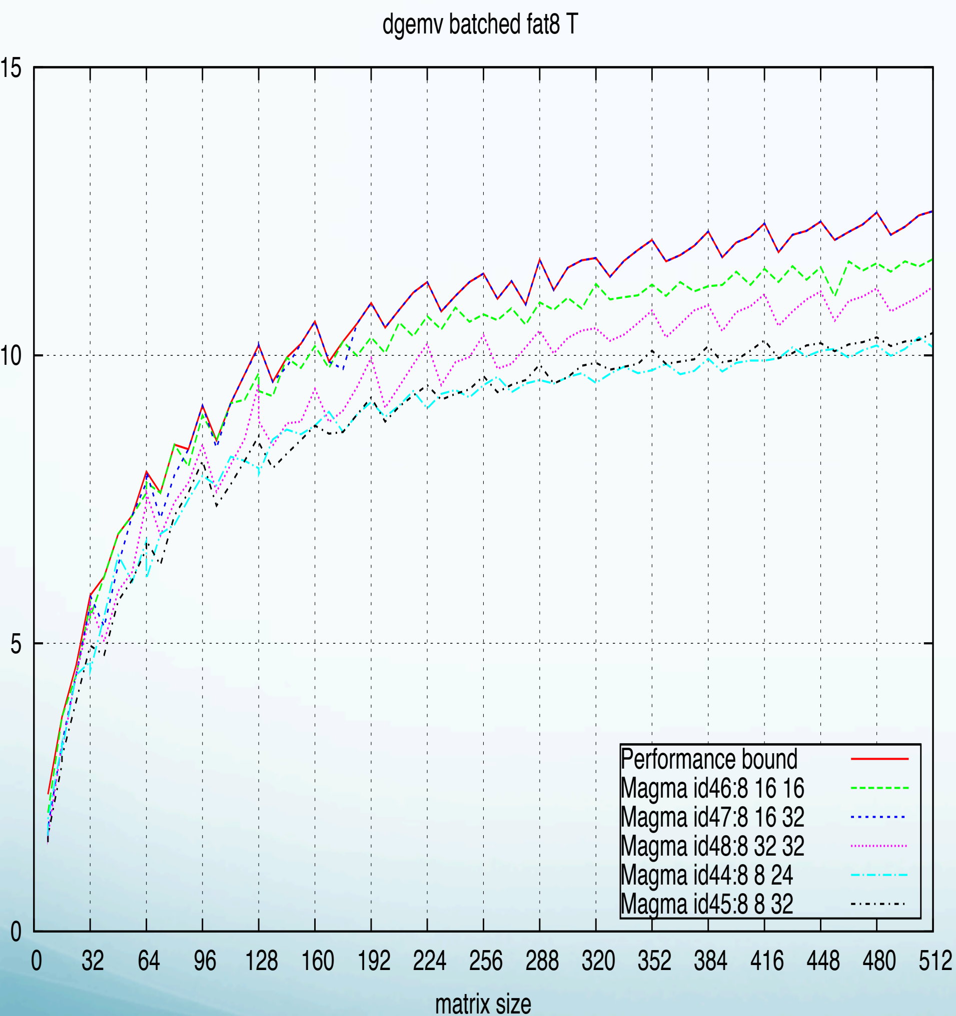
- Template code with tuning parameters

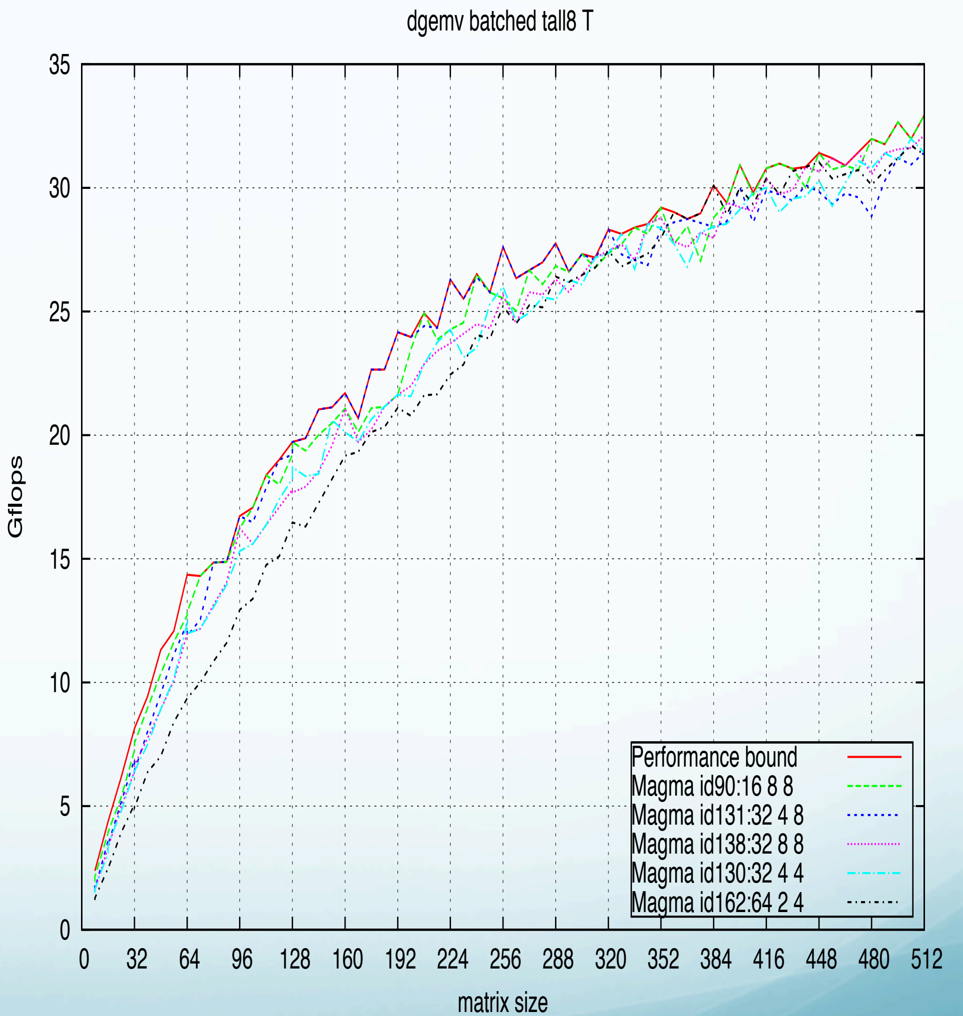
- Prune: Inferior/invalid configurations will be eliminated

  based on hardware information, CUDA limitation

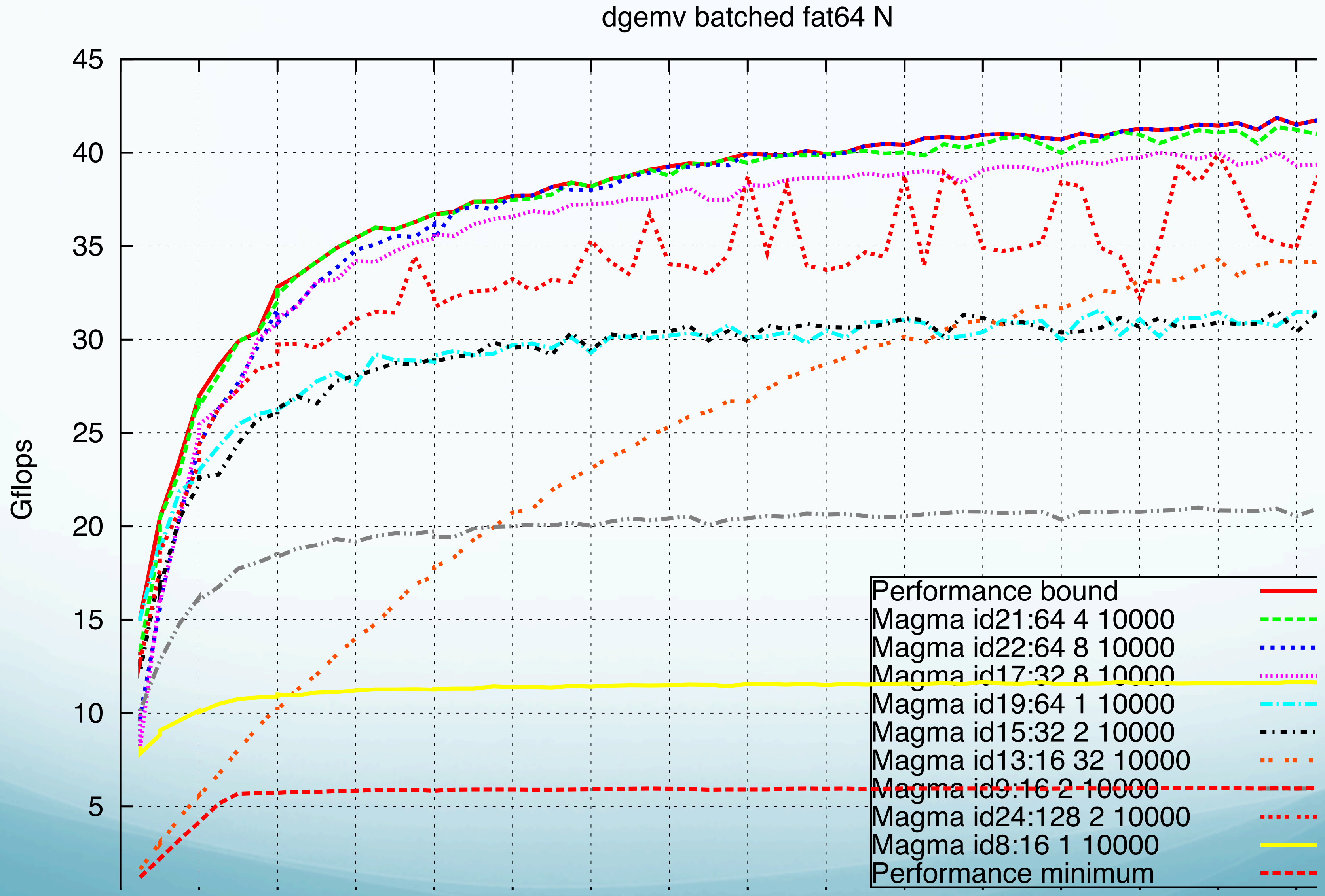
  e.g. **262** valid kernels of S/D/C/Z GEMVN

Fat/Tall GEMVT (M/N=8)

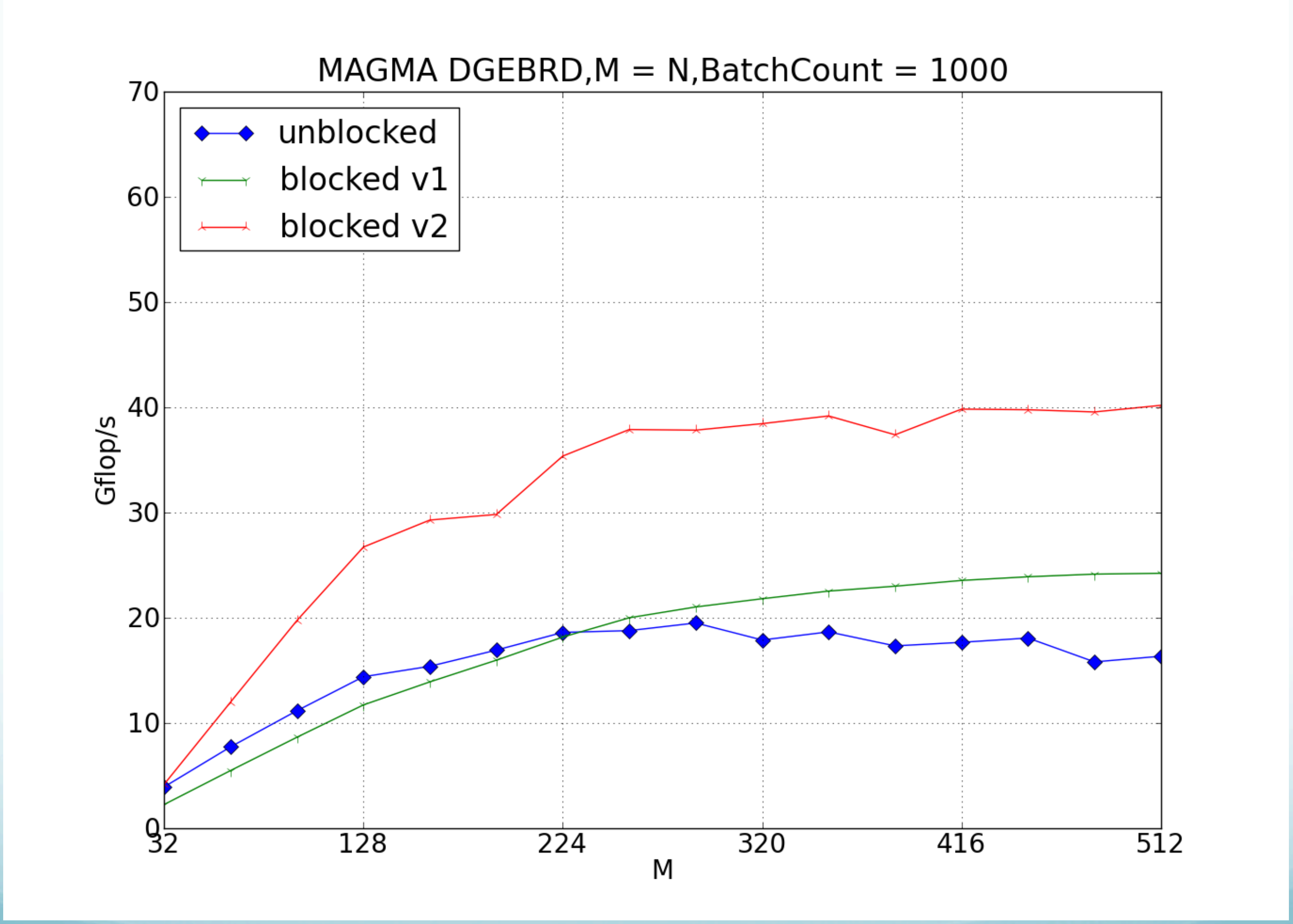




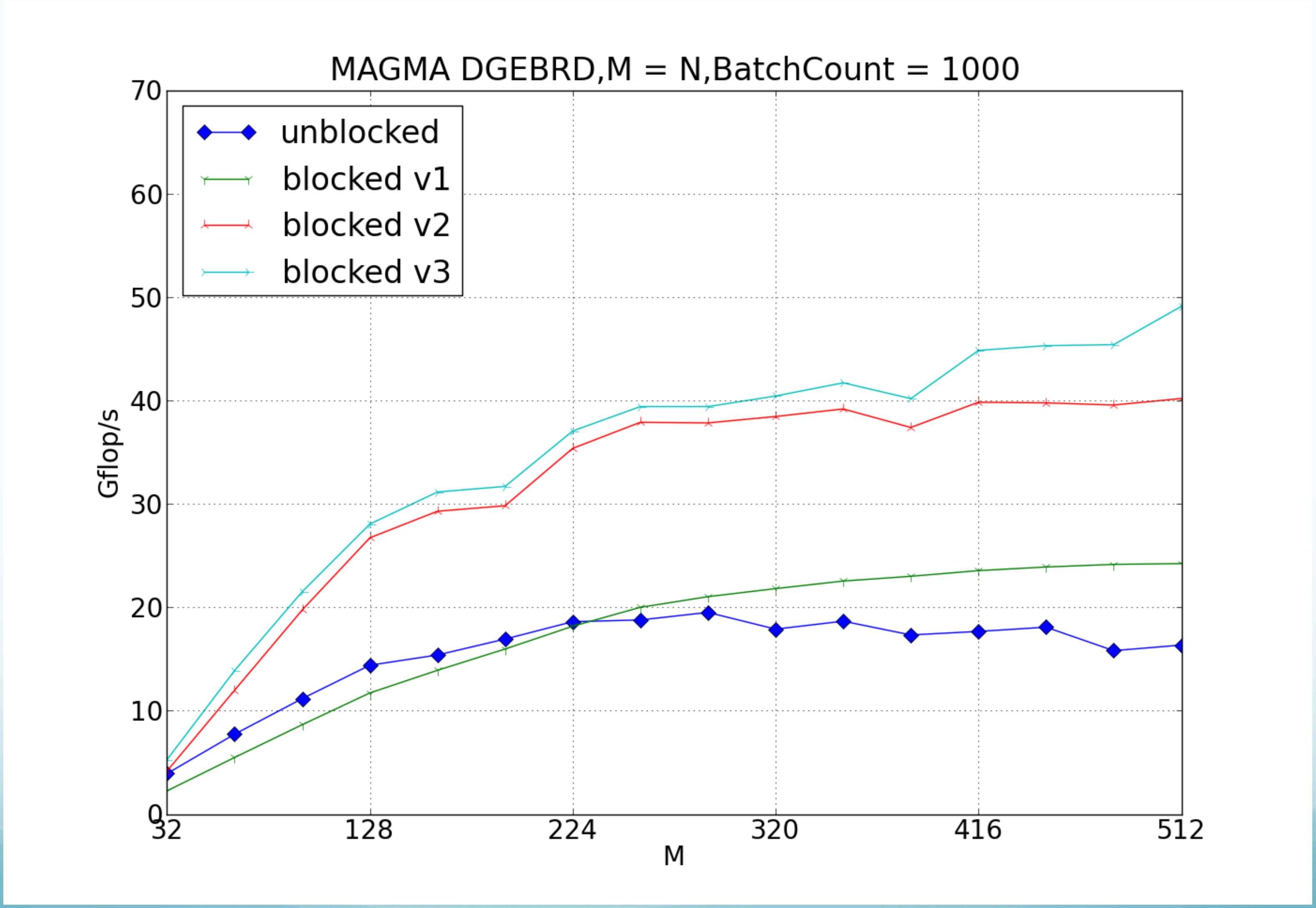
Fat GEMVT (M=64)



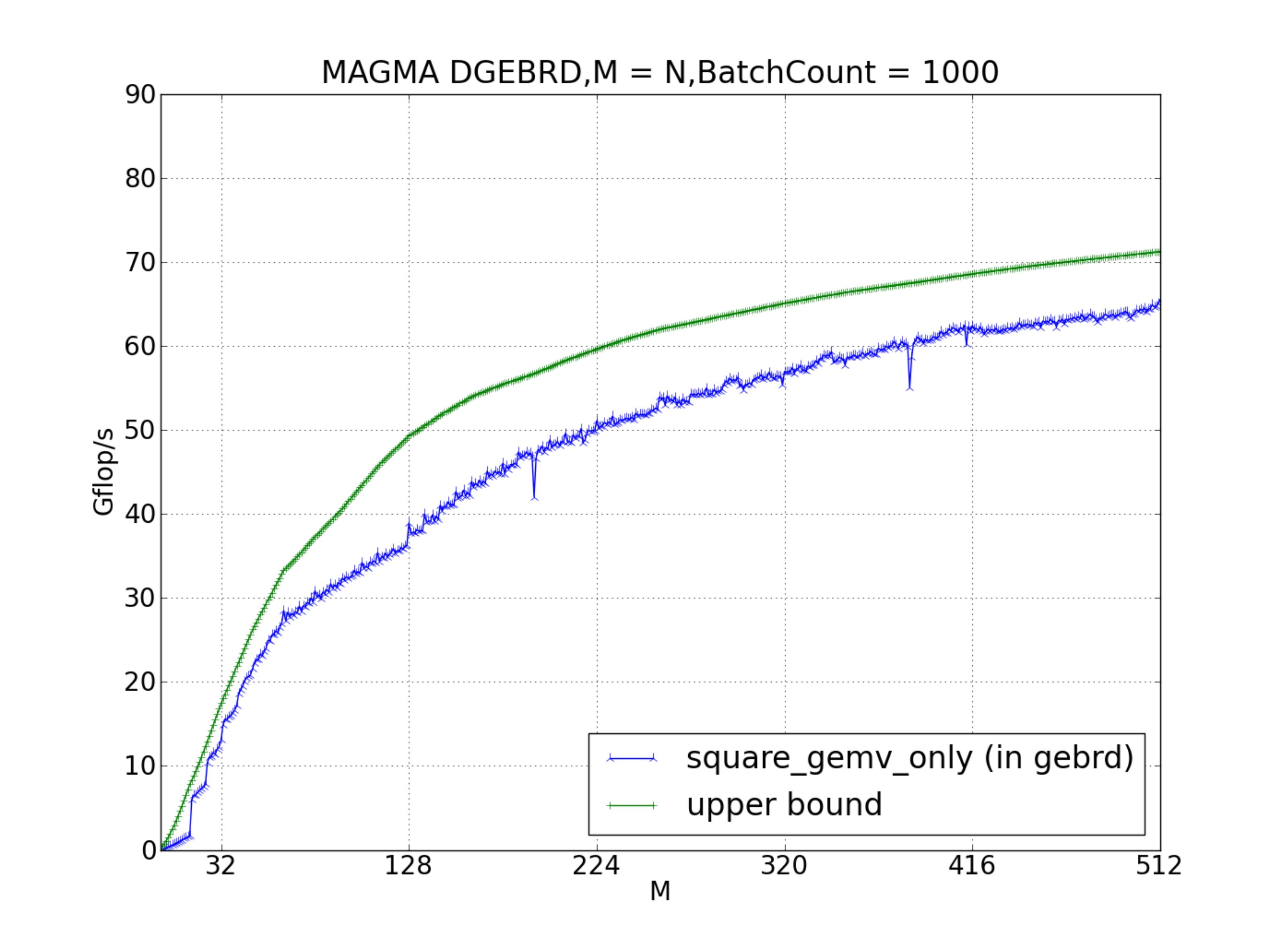
Batched GEBRD on K40c



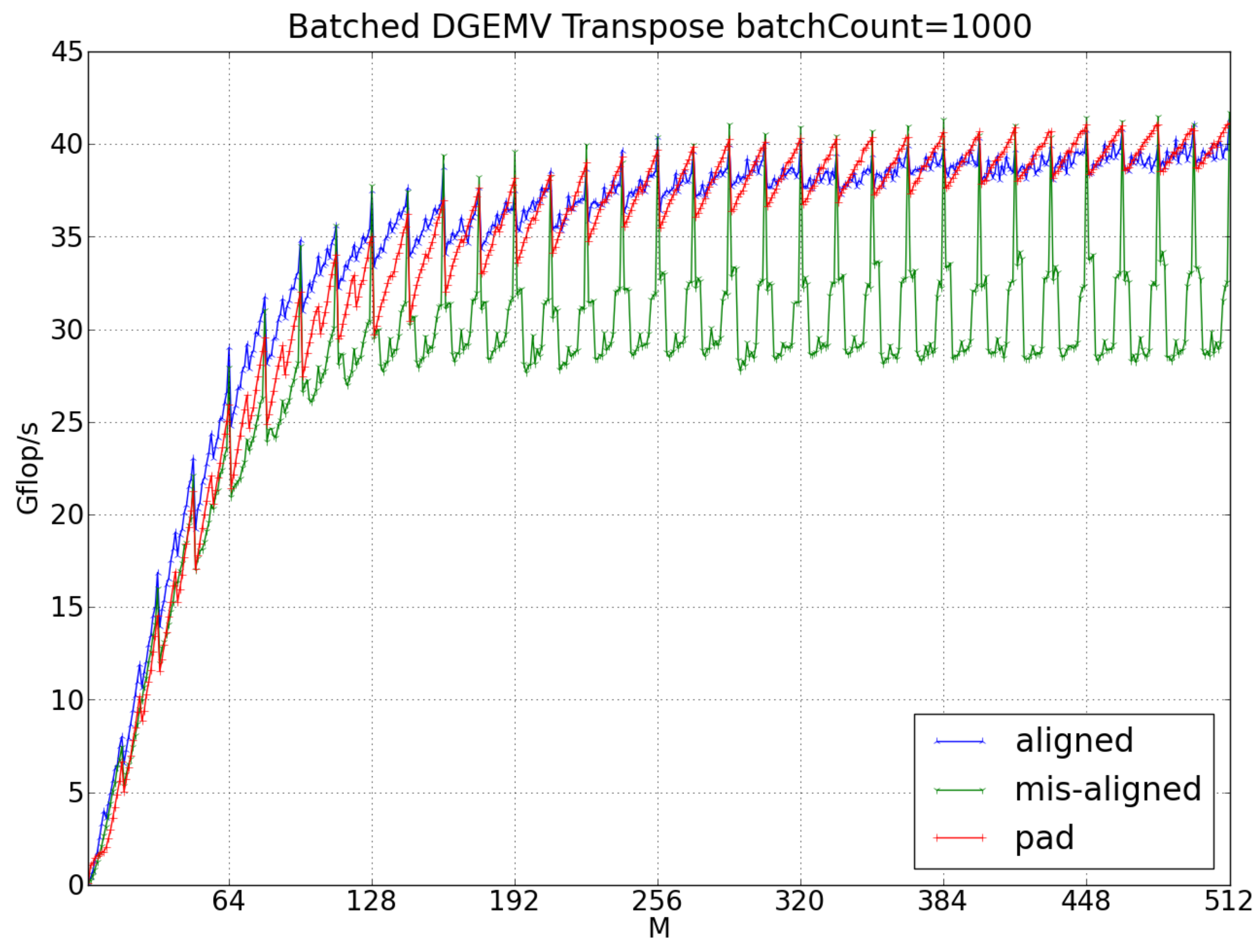
Batched GEBRD on K40c



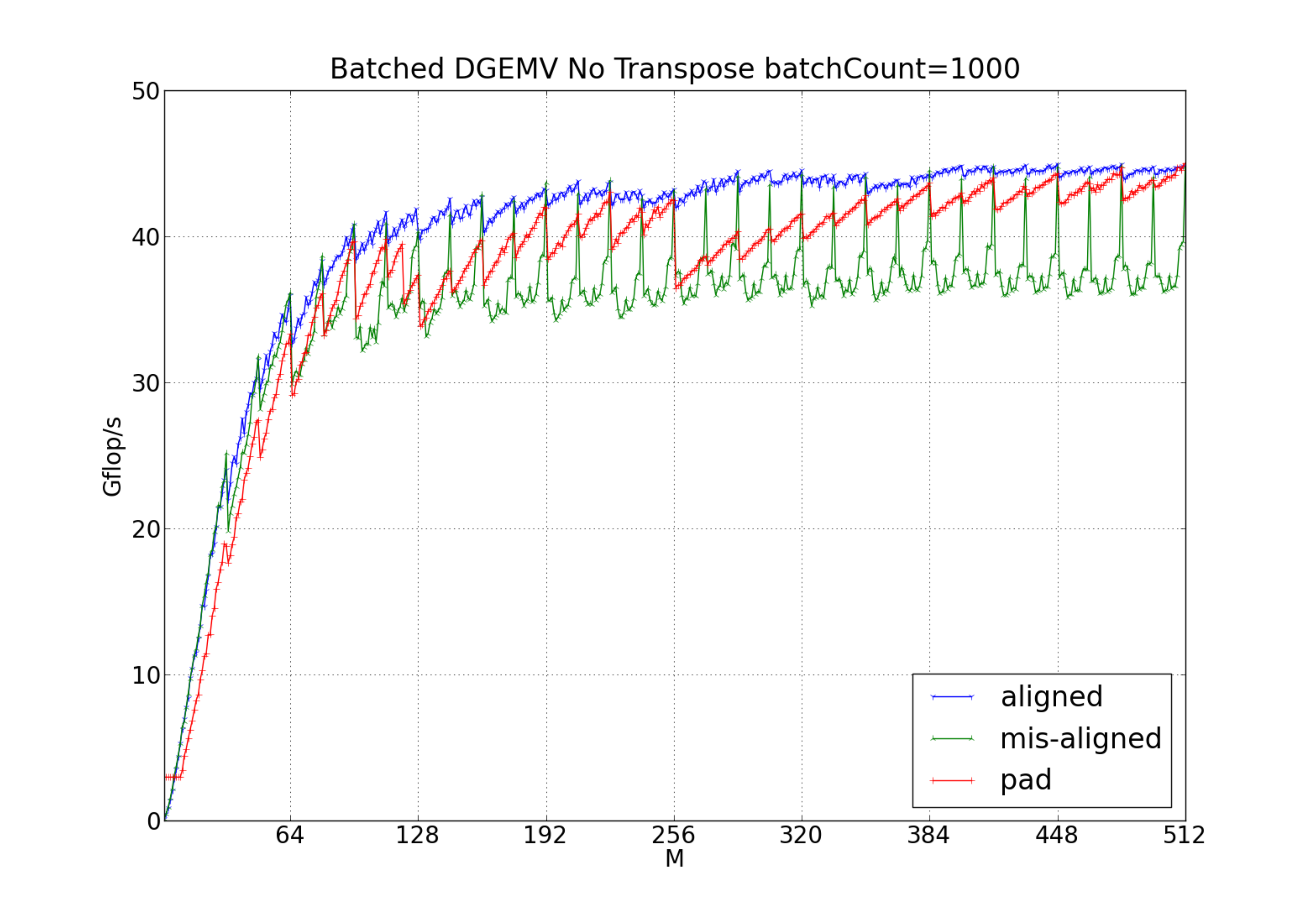
Upper bound: GEMV



Optimization 2: Alignment GEMVT

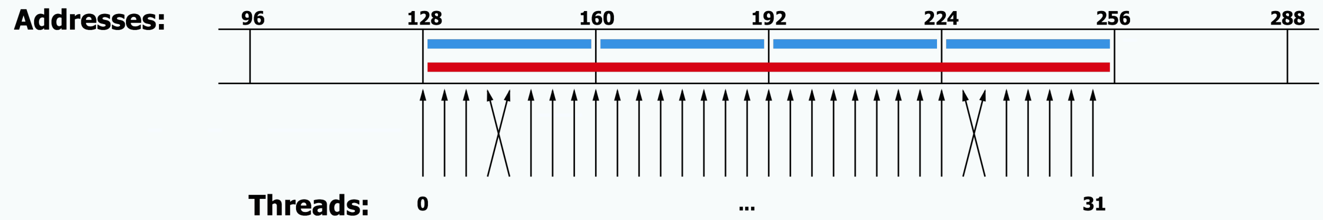


Optimization 2: Alignment GEMVN



Data Alignment on GPU

Aligned accesses (sequential/non-sequential)

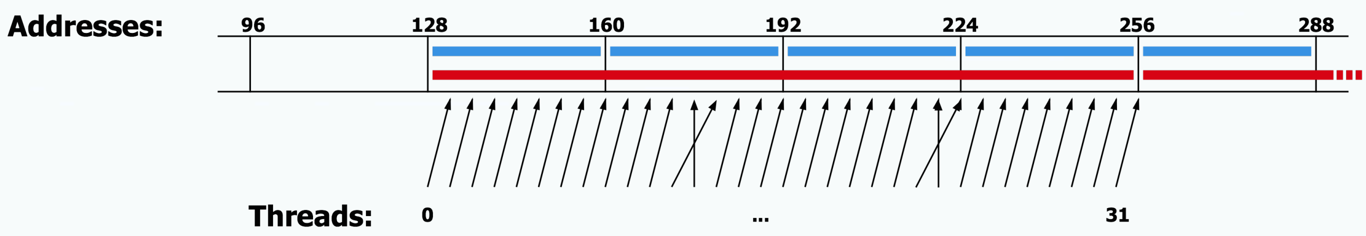


Compute capability:	2.x, 3.x, 5.x	
Memory transactions:	Uncached	Cached
	1x 32B at 128 1x 32B at 160 1x 32B at 192 1x 32B at 224	1x 128B at 128

Figure from CUDA Programming Guide 7.0

Data Misalignment on GPU

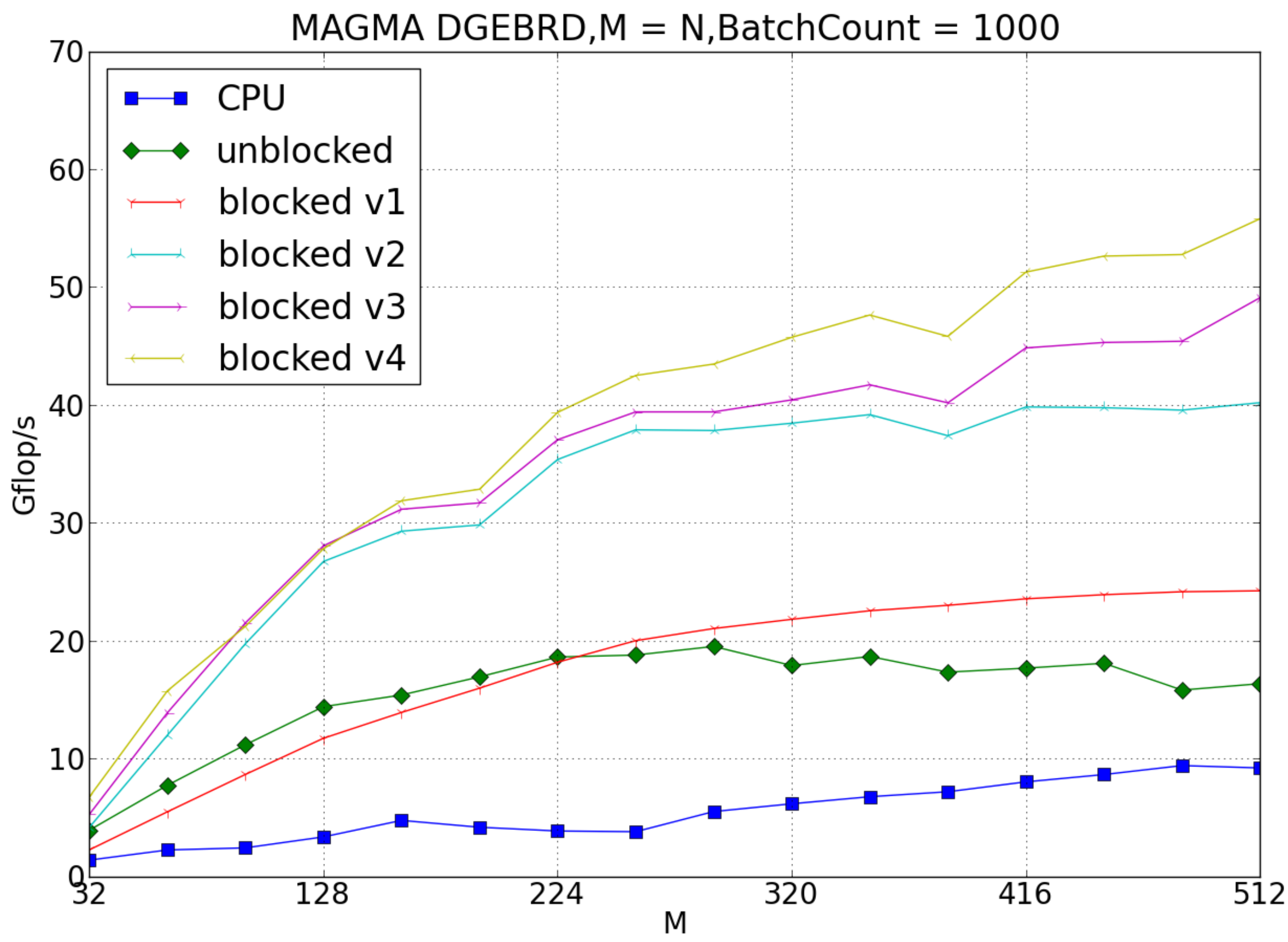
Mis-aligned accesses (sequential/non-sequential)



Compute capability:	2.x, 3.x, 5.x	
Memory transactions:	Uncached	Cached
	1x 32B at 128 1x 32B at 160 1x 32B at 192 1x 32B at 224 1x 32B at 256	1x 128B at 128 1x 128B at 256

Figure from CUDA Programming Guide 7.0

Batched GEBRD on K40c

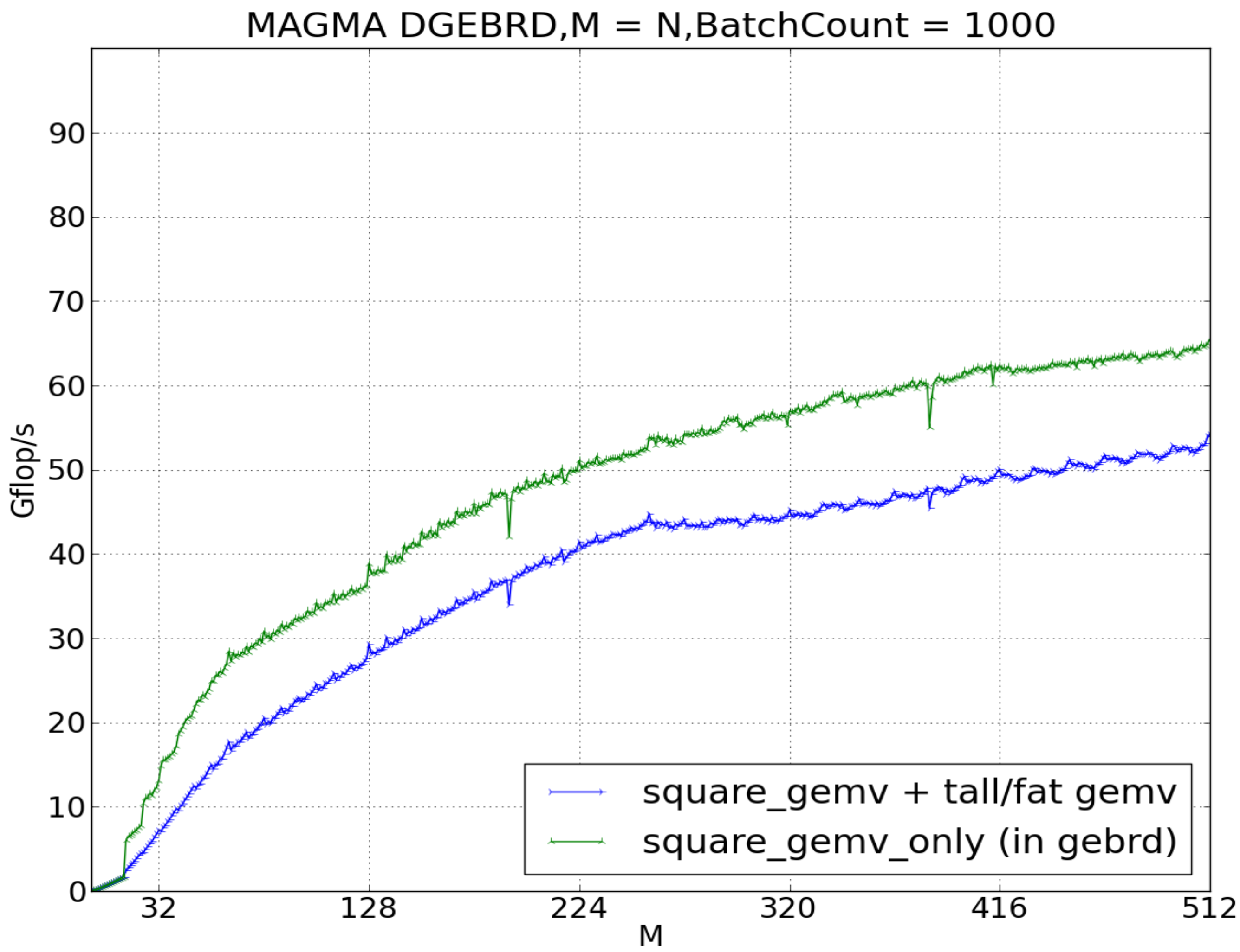


GEMV in GEBRD

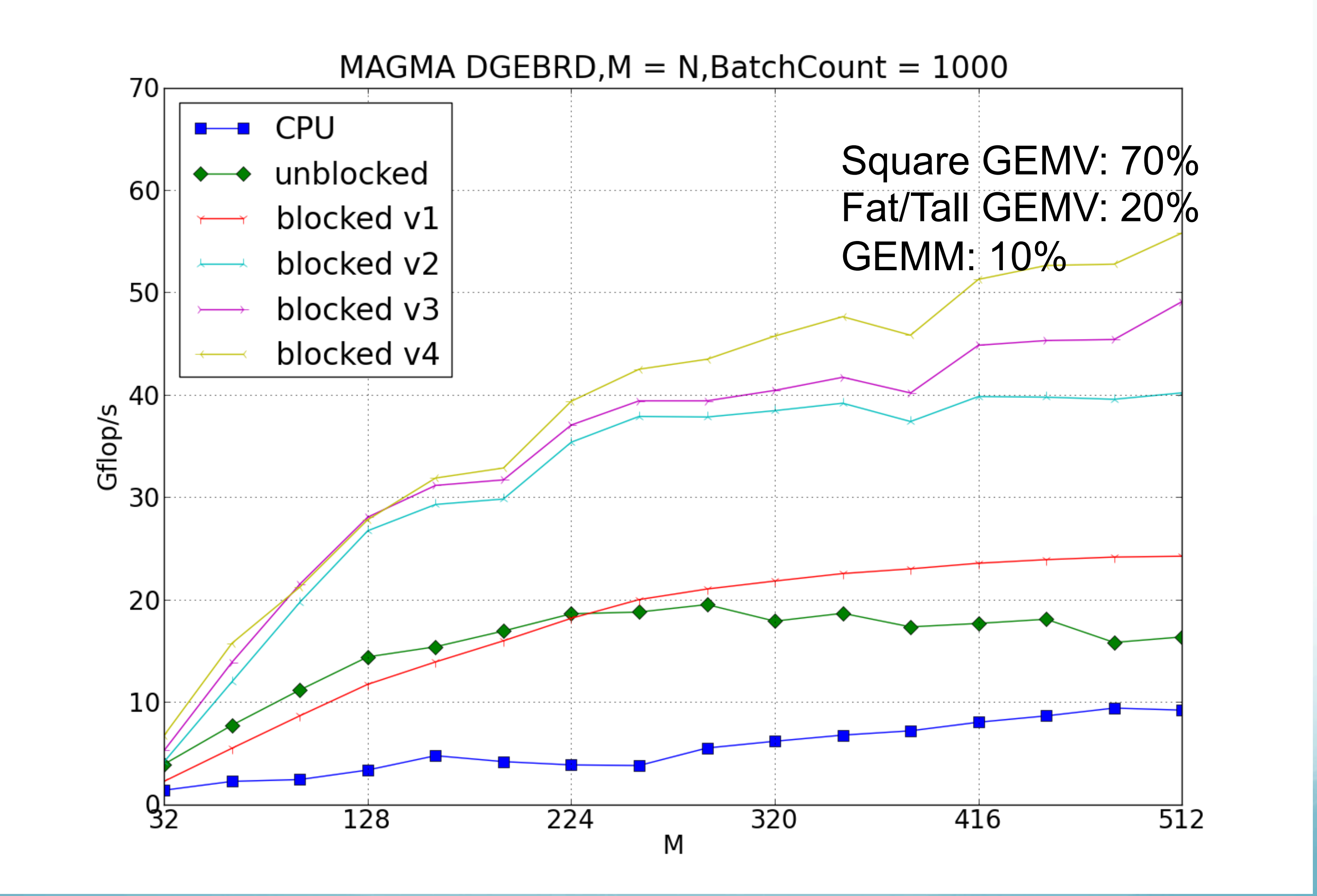
- 13 GEMV calls in each step of GEBRD
- 2 Square GEMV: big
- 11 Fat, Tall GEMV: small, row/column = NB

Num of calls	Fat matrix	Tall matrix	Square
GEMV Nontranspose	1	6	1(Big)
GEMV transpose	2	2	1(Big)

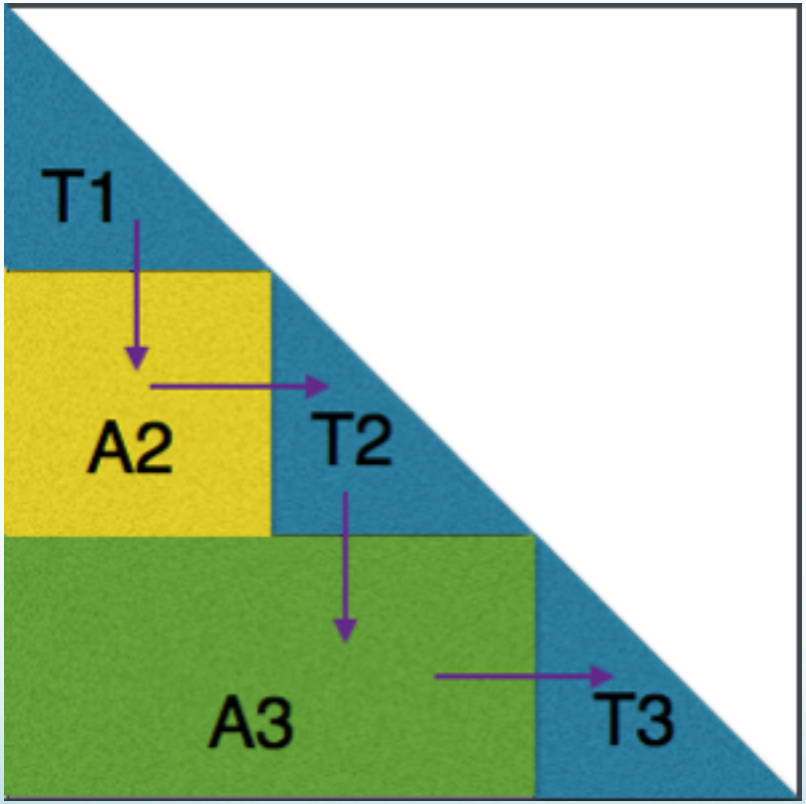
Tall/Fat GEMV Contribution



Batched GEBRD on K40c



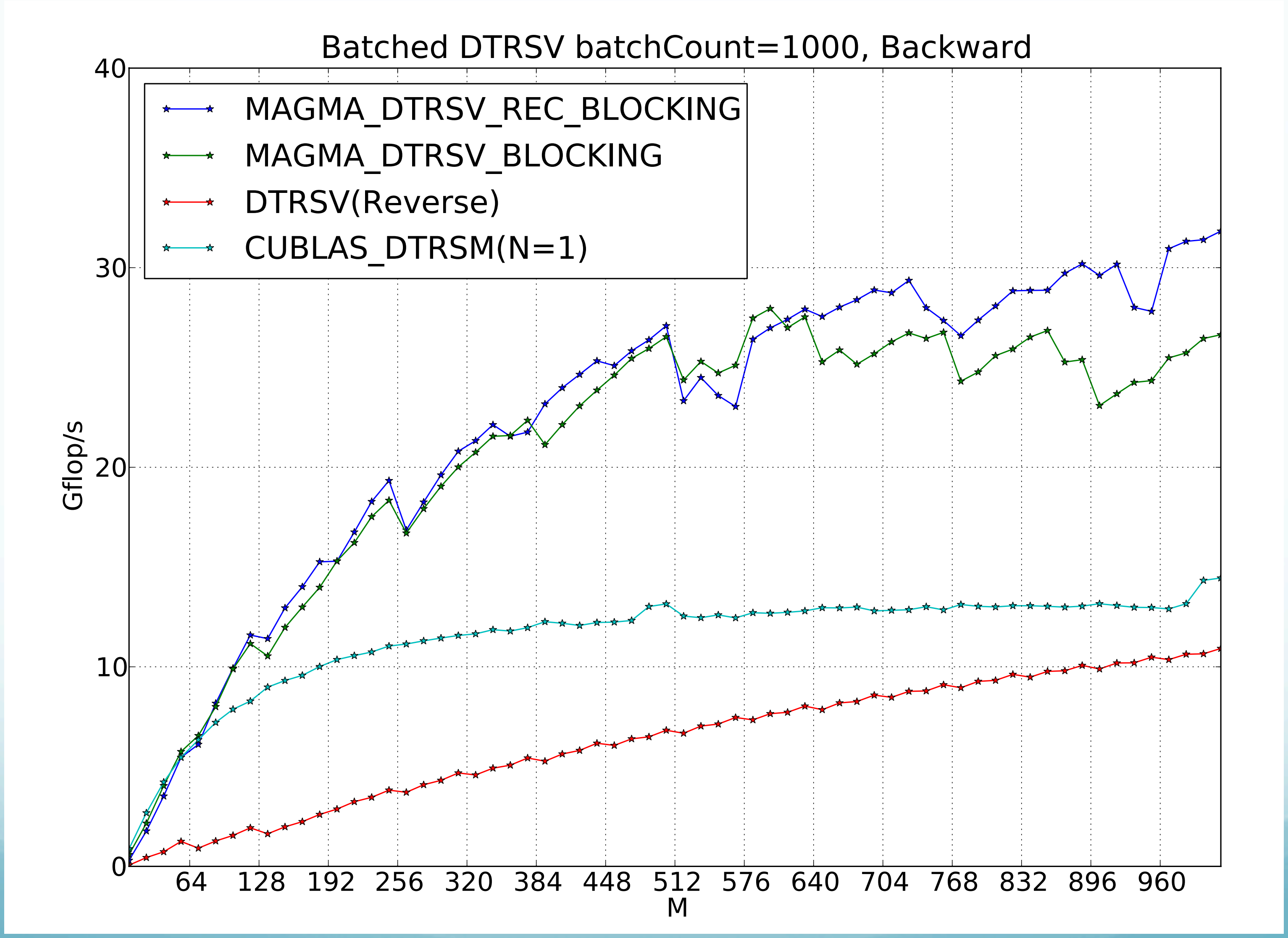
Batched TRSV: has GEMV



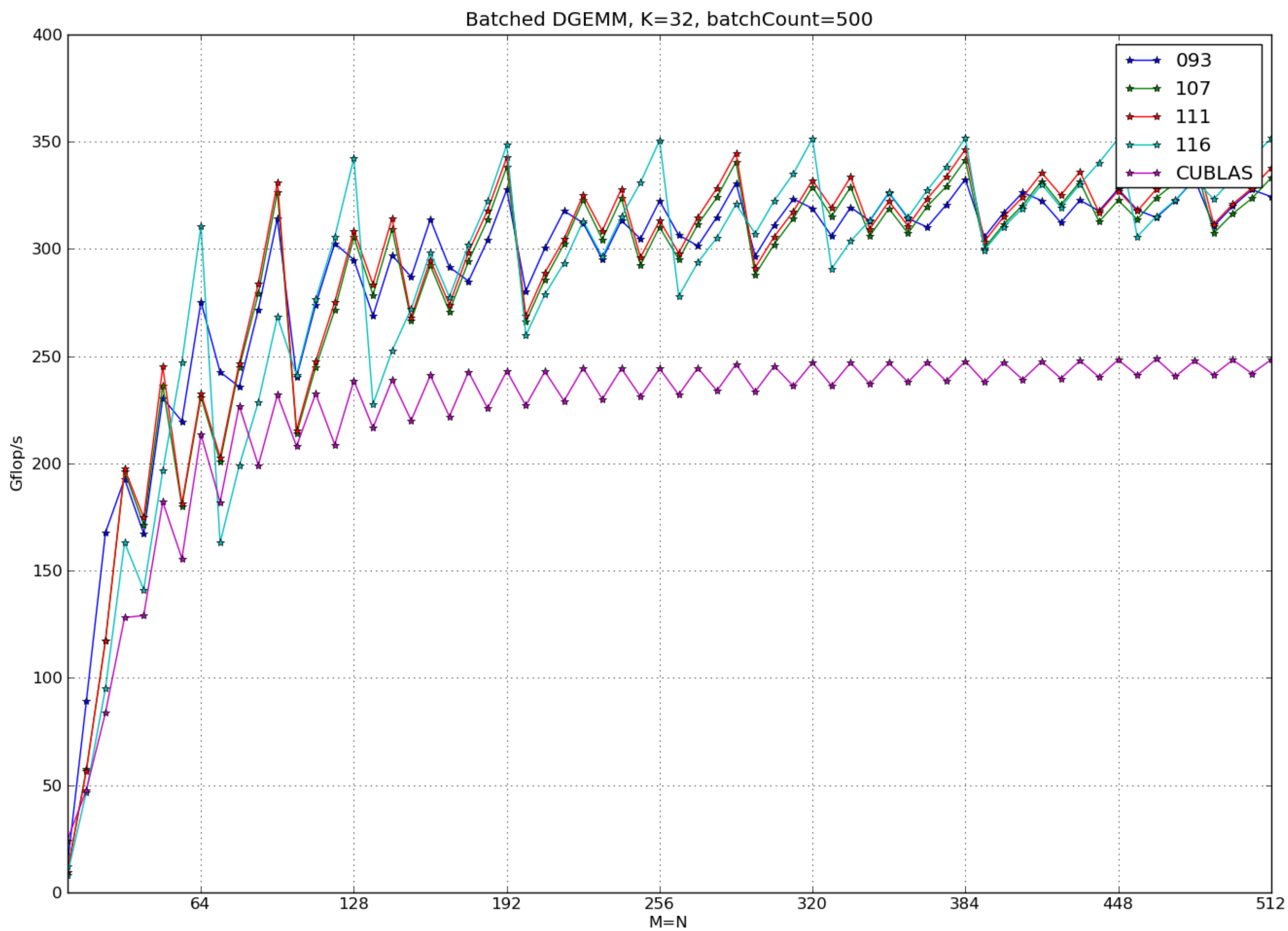
Ti: Triangular solver

Ai: GEMV to update

Batched TRSV



Tuning Batched GEMM:



Future work of GEBRD

- Support different size

- Optimized for size <128

  merge small gemv, gemm together