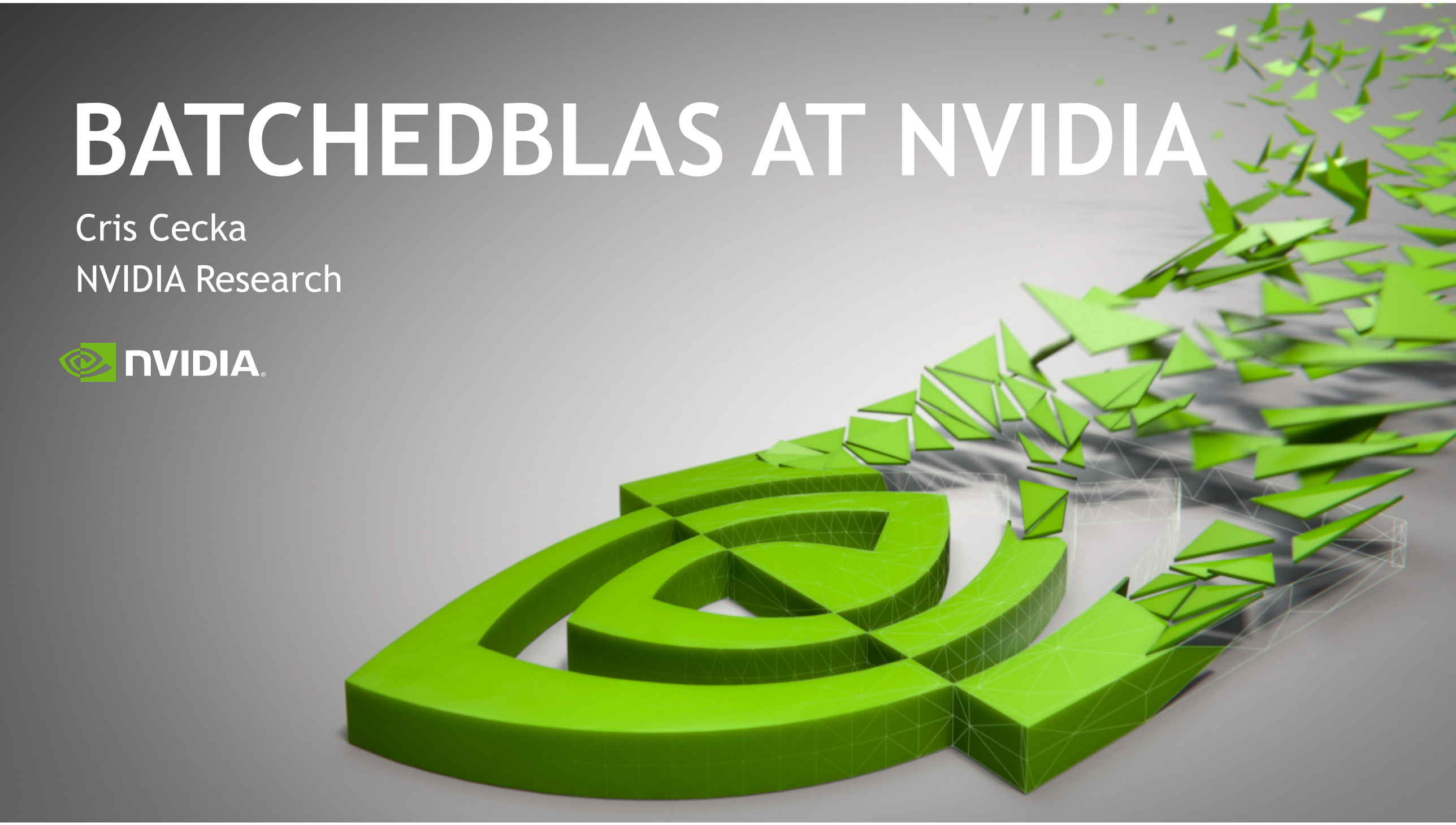
BATCHEDBLAS AT NVIDIA

Cris Cecka
NVIDIA Research



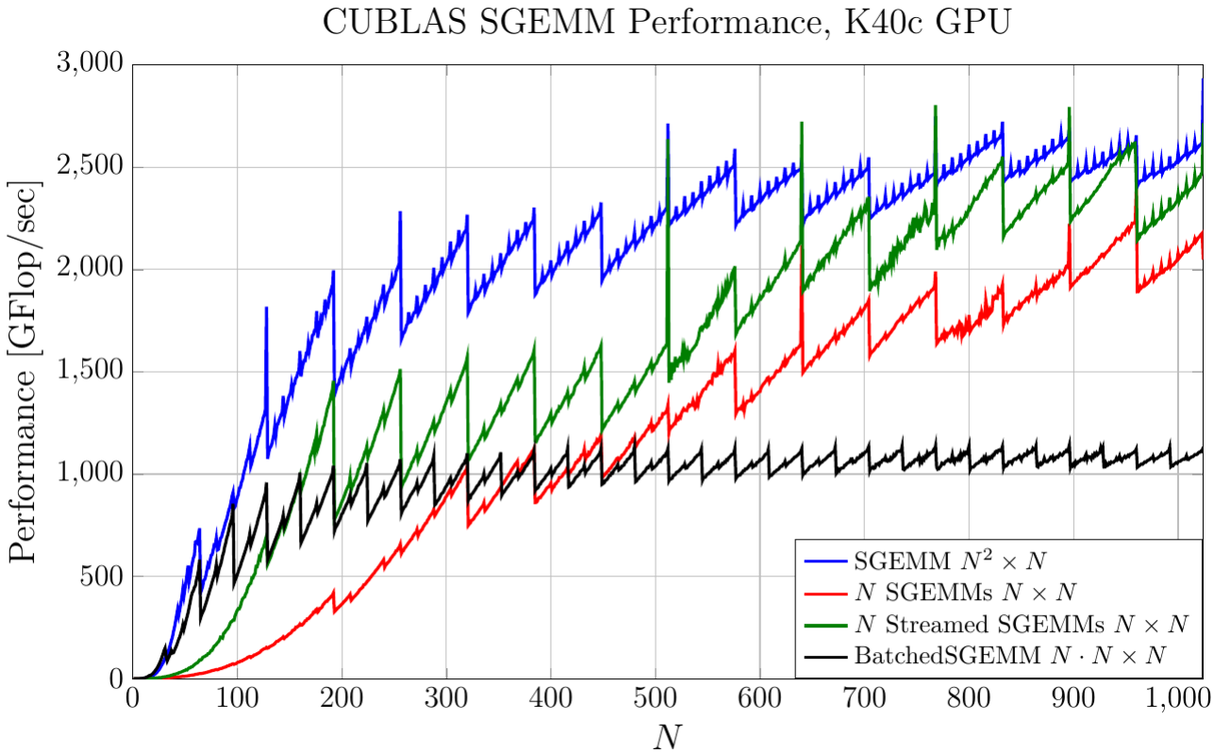
BatchedGEMM

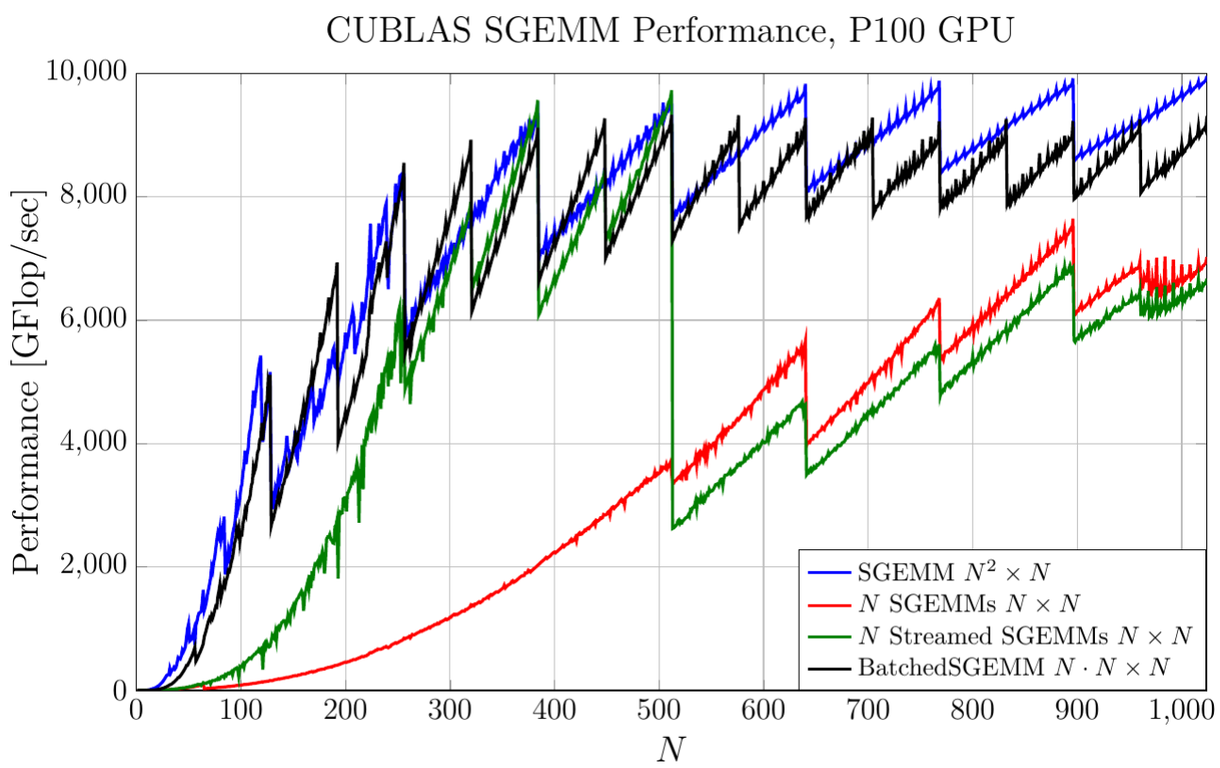
$$C[p] = \alpha\, \mathrm{op}(A[p])\, \mathrm{op}(B[p]) + \beta\, C[p]$$

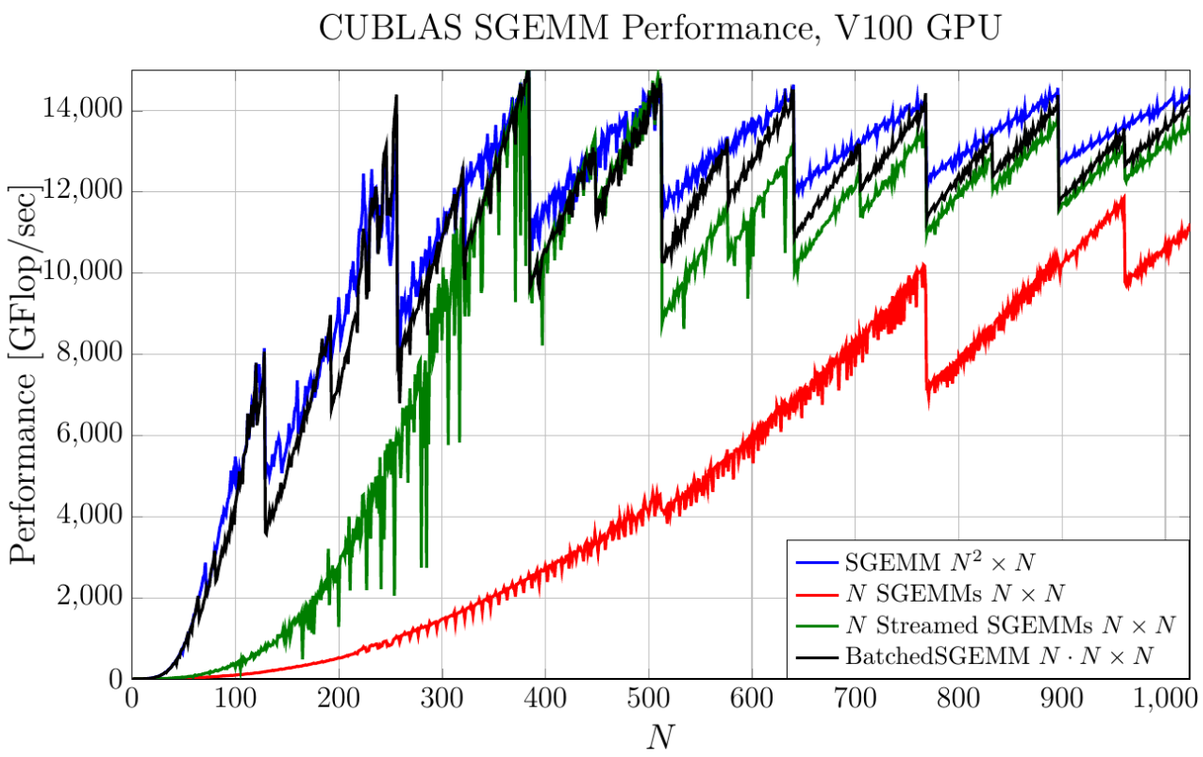
```
cublas<T>gemmBatched(cublasHandle_t handle,
                     cublasOperation_t transA, cublasOperation_t transB,
                     int M, int N, int K,
                     const T* alpha,
                     const T** A, int ldA,
                     const T** B, int ldB,
                     const T* beta,
                     T** C, int ldC,
                     int batchCount)
```

- Fixed size
- Variable stride via pointer-to-pointer

BatchedGEMM Benchmark

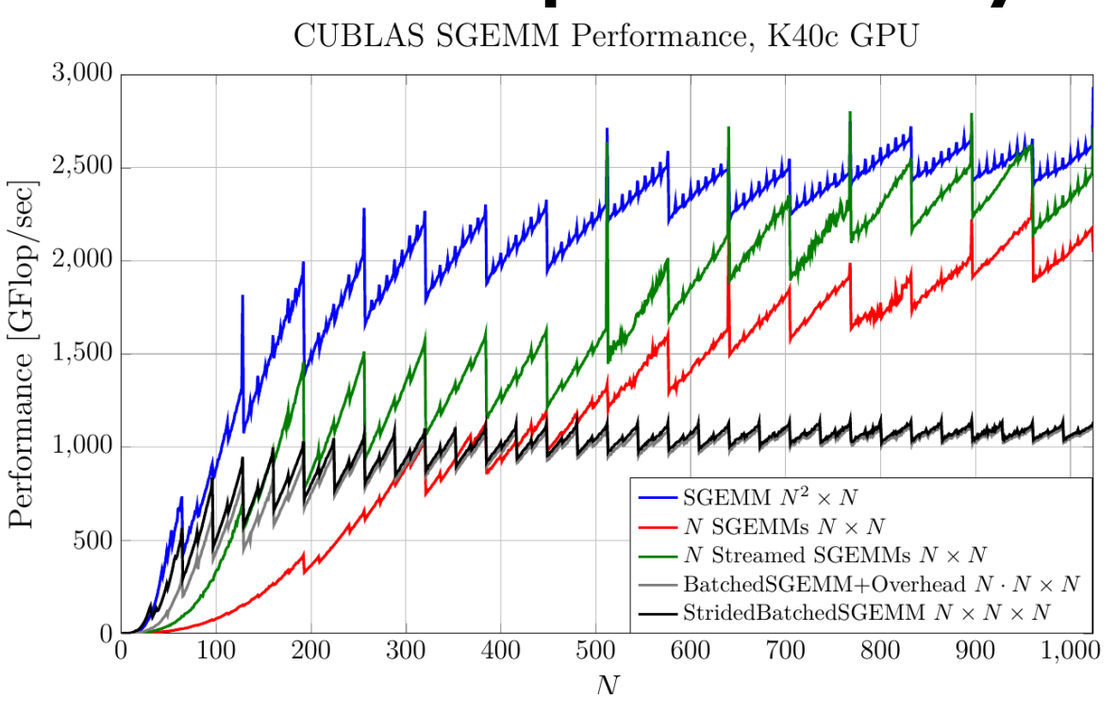


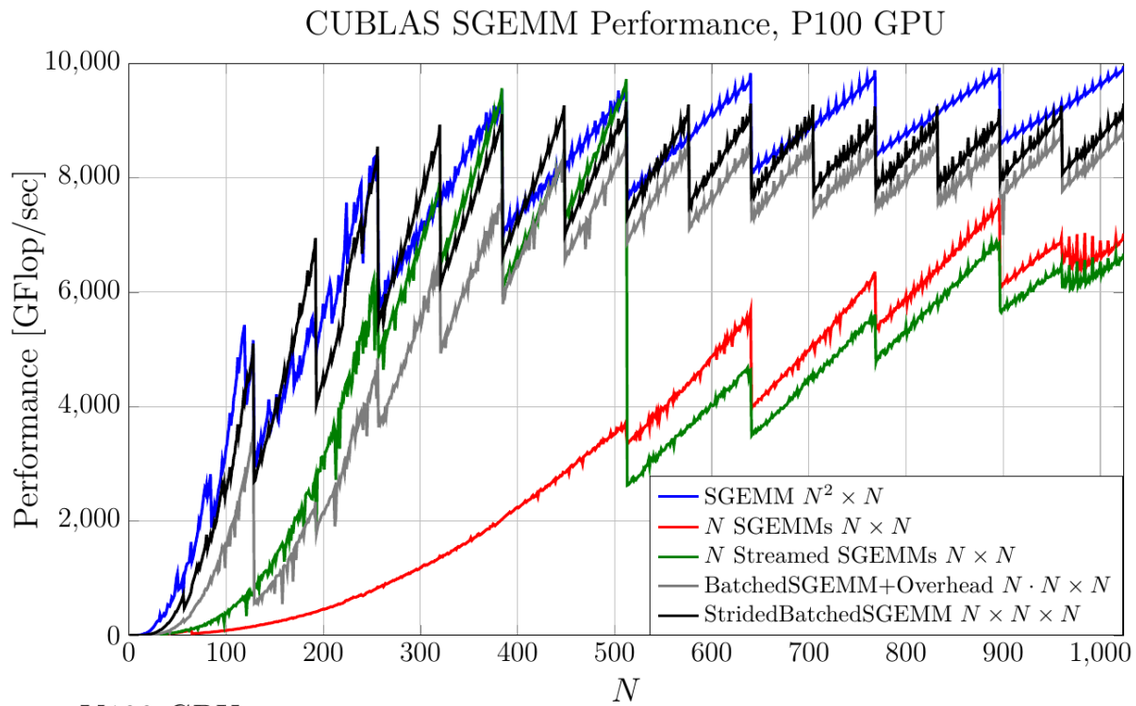


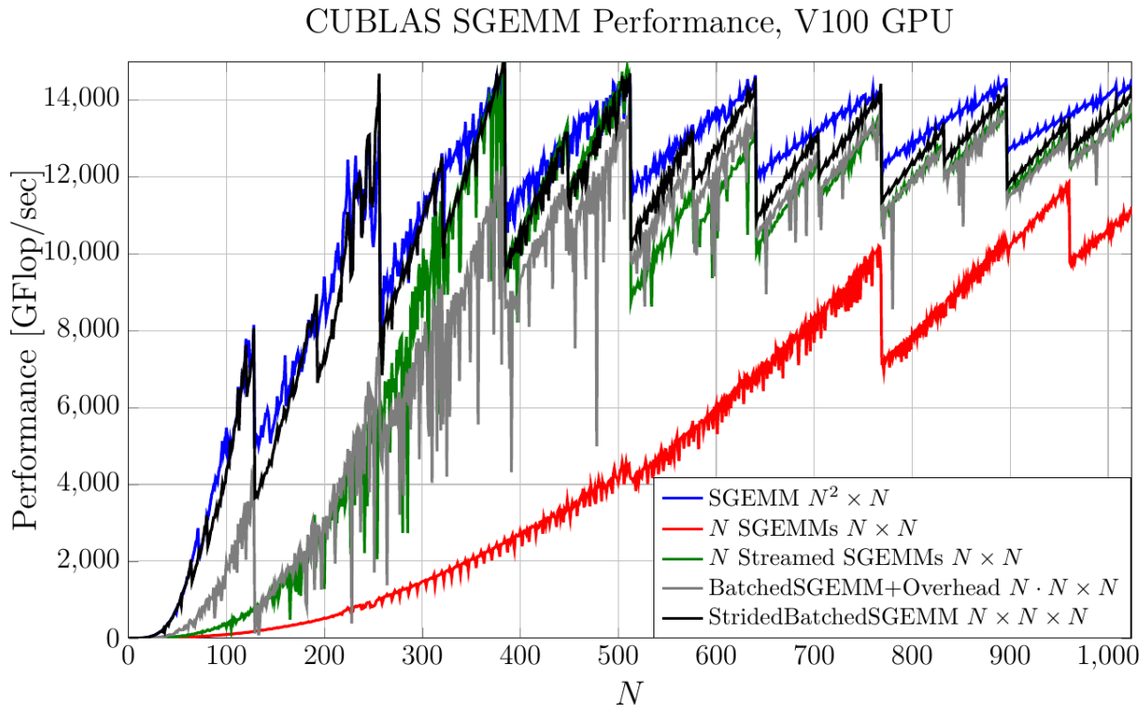


NVIDIA Benchmarks

- Except actually…







Strided BatchedGEMM

```
cublas<T>gemmStridedBatched(cublasHandle_t handle,
                            cublasOperation_t transA, cublasOperation_t transB,
                            int M, int N, int K,
                            const T* alpha,
                            const T* A, int ldA1, int strideA,
                            const T* B, int ldB1, int strideB,
                            const T* beta,
                            T* C, int ldC1, int strideC,
                            int batchCount)
```

- Common use case for Pointer-to-pointer BatchedGEMM.
- No Pointer-to-pointer data structure or overhead.
- Performance on par with pure GEMM (P100 and beyond).

Tensor Contractions

$$C_{mnp} = A_{**} \times B_{***}$$

$$C_{mnp} \equiv C[m + n \cdot \mathrm{ldC1} + p \cdot \mathrm{ldC2}]$$

Case	Contraction	Kernel1	Kernel2	Case	Contraction	Kernel1	Kernel2
1.1	$A_{mk}B_{knp}$	$C_{m(np)} = A_{mk}B_{k(np)}$	$C_{mn[p]} = A_{mk}B_{kn[p]}$	4.1	$A_{kn}B_{kmp}$	$C_{mn[p]} = B^{\top}_{km[p]}A_{kn}$	
1.2	$A_{mk}B_{kpn}$	$C_{mn[p]} = A_{mk}B_{k[p]n}$	$C_{m[n]p} = A_{mk}B_{kp[n]}$	4.2	$A_{kn}B_{kpm}$	$C_{mn[p]} = B^{\top}_{k[p]m}A_{kn}$	
1.3	$A_{mk}B_{nkp}$	$C_{mn[p]} = A_{mk}B^{\top}_{nk[p]}$		4.3	$A_{kn}B_{mkp}$	$C_{mn[p]} = B_{mk[p]}A_{kn}$	
1.4	$A_{mk}B_{pkn}$	$C_{m[n]p} = A_{mk}B^{\top}_{pk[n]}$		4.4	$A_{kn}B_{pkm}$		
1.5	$A_{mk}B_{npk}$	$C_{m(np)} = A_{mk}B^{\top}_{(np)k}$	$C_{mn[p]} = A_{mk}B^{\top}_{n[p]k}$	4.5	$A_{kn}B_{mpk}$	$C_{mn[p]} = B_{m[p]k}A_{kn}$	
1.6	$A_{mk}B_{pnk}$	$C_{m[n]p} = A_{mk}B^{\top}_{p[n]k}$		4.6	$A_{kn}B_{pmk}$		
2.1	$A_{km}B_{knp}$	$C_{m(np)} = A^{\top}_{km}B_{k(np)}$	$C_{mn[p]} = A^{\top}_{km}B_{kn[p]}$	5.1	$A_{pk}B_{kmn}$	$C_{(mn)p} = B^{\top}_{k(mn)}A^{\top}_{pk}$	$C_{m[n]p} = B^{\top}_{km[n]}A^{\top}_{pk}$
2.2	$A_{km}B_{kpn}$	$C_{mn[p]} = A^{\top}_{km}B_{k[p]n}$	$C_{m[n]p} = A^{\top}_{km}B_{kp[n]}$	5.2	$A_{pk}B_{knm}$	$C_{m[n]p} = B^{\top}_{k[n]m}A^{\top}_{pk}$	
2.3	$A_{km}B_{nkp}$	$C_{mn[p]} = A^{\top}_{km}B^{\top}_{nk[p]}$		5.3	$A_{pk}B_{mkn}$	$C_{m[n]p} = B_{mk[n]}A^{\top}_{pk}$	
2.4	$A_{km}B_{pkn}$	$C_{m[n]p} = A^{\top}_{km}B^{\top}_{pk[n]}$		5.4	$A_{pk}B_{nkm}$		
2.5	$A_{km}B_{npk}$	$C_{m(np)} = A^{\top}_{km}B^{\top}_{(np)k}$	$C_{mn[p]} = A^{\top}_{km}B^{\top}_{n[p]k}$	5.5	$A_{pk}B_{mnk}$	$C_{(mn)p} = B_{(mn)k}A^{\top}_{pk}$	$C_{m[n]p} = B_{m[n]k}A^{\top}_{pk}$
2.6	$A_{km}B_{pnk}$	$C_{m[n]p} = A^{\top}_{km}B^{\top}_{p[n]k}$		5.6	$A_{pk}B_{nmk}$		
3.1	$A_{nk}B_{kmp}$	$C_{mn[p]} = B^{\top}_{km[p]}A^{\top}_{nk}$		6.1	$A_{kp}B_{kmn}$	$C_{(mn)p} = B^{\top}_{k(mn)}A_{kp}$	$C_{m[n]p} = B^{\top}_{km[n]}A_{kp}$
3.2	$A_{nk}B_{kpm}$	$C_{mn[p]} = B^{\top}_{k[p]m}A^{\top}_{nk}$		6.2	$A_{kp}B_{knm}$	$C_{m[n]p} = B^{\top}_{k[n]m}A_{kp}$	
3.3	$A_{nk}B_{mkp}$	$C_{mn[p]} = B_{mk[p]}A^{\top}_{nk}$		6.3	$A_{kp}B_{mkn}$	$C_{m[n]p} = B_{mk[n]}A_{kp}$	
3.4	$A_{nk}B_{pkm}$			6.4	$A_{kp}B_{nkm}$		
3.5	$A_{nk}B_{mpk}$	$C_{mn[p]} = B_{m[p]k}A^{\top}_{nk}$		6.5	$A_{kp}B_{mnk}$	$C_{(mn)p} = B_{(mn)k}A_{kp}$	$C_{m[n]p} = B_{m[n]k}A_{kp}$
3.6	$A_{nk}B_{pmk}$			6.6	$A_{kp}B_{nmk}$		

: Single GEMM

(Provided compact layout)

Tensor Contractions

$$C_{mnp} = A_{**} \times B_{***}$$

$$C_{mnp} \equiv C[m + n \cdot \mathrm{ldC1} + p \cdot \mathrm{ldC2}]$$

Case	Contraction	Kernel1	Kernel2
1.1	$A_{mk}B_{knp}$	$C_{m(np)} = A_{mk}B_{k(np)}$	$C_{mn[p]} = A_{mk}B_{kn[p]}$
1.2	$A_{mk}B_{kpn}$	$C_{mn[p]} = A_{mk}B_{k[p]n}$	$C_{m[n]p} = A_{mk}B_{kp[n]}$
1.3	$A_{mk}B_{nkp}$	$C_{mn[p]} = A_{mk}B^{\top}_{nk[p]}$	
1.4	$A_{mk}B_{pkn}$	$C_{m[n]p} = A_{mk}B^{\top}_{pk[n]}$	
1.5	$A_{mk}B_{npk}$	$C_{m(np)} = A_{mk}B^{\top}_{(np)k}$	$C_{mn[p]} = A_{mk}B^{\top}_{n[p]k}$
1.6	$A_{mk}B_{pnk}$	$C_{m[n]p} = A_{mk}B^{\top}_{p[n]k}$	
2.1	$A_{km}B_{knp}$	$C_{m(np)} = A^{\top}_{km}B_{k(np)}$	$C_{mn[p]} = A^{\top}_{km}B_{kn[p]}$
2.2	$A_{km}B_{kpn}$	$C_{mn[p]} = A^{\top}_{km}B_{k[p]n}$	$C_{m[n]p} = A^{\top}_{km}B_{kp[n]}$
2.3	$A_{km}B_{nkp}$	$C_{mn[p]} = A^{\top}_{km}B^{\top}_{nk[p]}$	
2.4	$A_{km}B_{pkn}$	$C_{m[n]p} = A^{\top}_{km}B^{\top}_{pk[n]}$	
2.5	$A_{km}B_{npk}$	$C_{m(np)} = A^{\top}_{km}B^{\top}_{(np)k}$	$C_{mn[p]} = A^{\top}_{km}B^{\top}_{n[p]k}$
2.6	$A_{km}B_{pnk}$	$C_{m[n]p} = A^{\top}_{km}B^{\top}_{p[n]k}$	
3.1	$A_{nk}B_{kmp}$	$C_{mn[p]} = B^{\top}_{km[p]}A^{\top}_{nk}$	
3.2	$A_{nk}B_{kpm}$	$C_{mn[p]} = B^{\top}_{k[p]m}A^{\top}_{nk}$	
3.3	$A_{nk}B_{mkp}$	$C_{mn[p]} = B_{mk[p]}A^{\top}_{nk}$	
3.4	$A_{nk}B_{pkm}$		
3.5	$A_{nk}B_{mpk}$	$C_{mn[p]} = B_{m[p]k}A^{\top}_{nk}$	
3.6	$A_{nk}B_{pmk}$		

Case	Contraction	Kernel1	Kernel2
4.1	$A_{kn}B_{kmp}$	$C_{mn[p]} = B^{\top}_{km[p]}A_{kn}$	
4.2	$A_{kn}B_{kpm}$	$C_{mn[p]} = B^{\top}_{k[p]m}A_{kn}$	
4.3	$A_{kn}B_{mkp}$	$C_{mn[p]} = B_{mk[p]}A_{kn}$	
4.4	$A_{kn}B_{pkm}$		
4.5	$A_{kn}B_{mpk}$	$C_{mn[p]} = B_{m[p]k}A_{kn}$	
4.6	$A_{kn}B_{pmk}$		
5.1	$A_{pk}B_{kmn}$	$C_{(mn)p} = B^{\top}_{k(mn)}A^{\top}_{pk}$	$C_{m[n]p} = B^{\top}_{km[n]}A^{\top}_{pk}$
5.2	$A_{pk}B_{knm}$	$C_{m[n]p} = B^{\top}_{k[n]m}A^{\top}_{pk}$	
5.3	$A_{pk}B_{mkn}$	$C_{m[n]p} = B_{mk[n]}A^{\top}_{pk}$	
5.4	$A_{pk}B_{nkm}$		
5.5	$A_{pk}B_{mnk}$	$C_{(mn)p} = B_{(mn)k}A^{\top}_{pk}$	$C_{m[n]p} = B_{m[n]k}A^{\top}_{pk}$
5.6	$A_{pk}B_{nmk}$		
6.1	$A_{kp}B_{kmn}$	$C_{(mn)p} = B^{\top}_{k(mn)}A_{kp}$	$C_{m[n]p} = B^{\top}_{km[n]}A_{kp}$
6.2	$A_{kp}B_{knm}$	$C_{m[n]p} = B^{\top}_{k[n]m}A_{kp}$	
6.3	$A_{kp}B_{mkn}$	$C_{m[n]p} = B_{mk[n]}A_{kp}$	
6.4	$A_{kp}B_{nkm}$		
6.5	$A_{kp}B_{mnk}$	$C_{(mn)p} = B_{(mn)k}A_{kp}$	$C_{m[n]p} = B_{m[n]k}A_{kp}$
6.6	$A_{kp}B_{nmk}$		

: Single SB-GEMM
(Any layout)

Other Batched

Fixed, usually pointer-to-pointer, batched primitives:

```
cublasGemmStridedBatchedEx          (Mixed precision)
cublasGemmBatchedEx                 (Mixed precision)
cublasDgetrfBatched
cublasDgetriBatched
cublasDgetrfBatched
cublasDgetrsBatched

cusolverDnDpotrsBatched
cusolverDnDpotrfBatched
cusolverDnDgesvdjBatched
cusolverDnDsyevjBatched
cusolverSpDcsrqrsvBatched

cusparseDgtsv2StridedBatch
cusparseDgtsvInterleavedBatch
cusparseDgpsv2StridedBatch
cusparseDgpsvInterleavedBatch
```

Summary

- NVIDIA supporting Batched BLAS/LAPACK
  - All fixed batch, rather than variable batch.
  - Mostly pointer-to-pointer input.
  - Some strided batch input.
    - GEMM, GTSV, GPSV
- Emerging interleaved batch support.
  - GTSV, GPSV