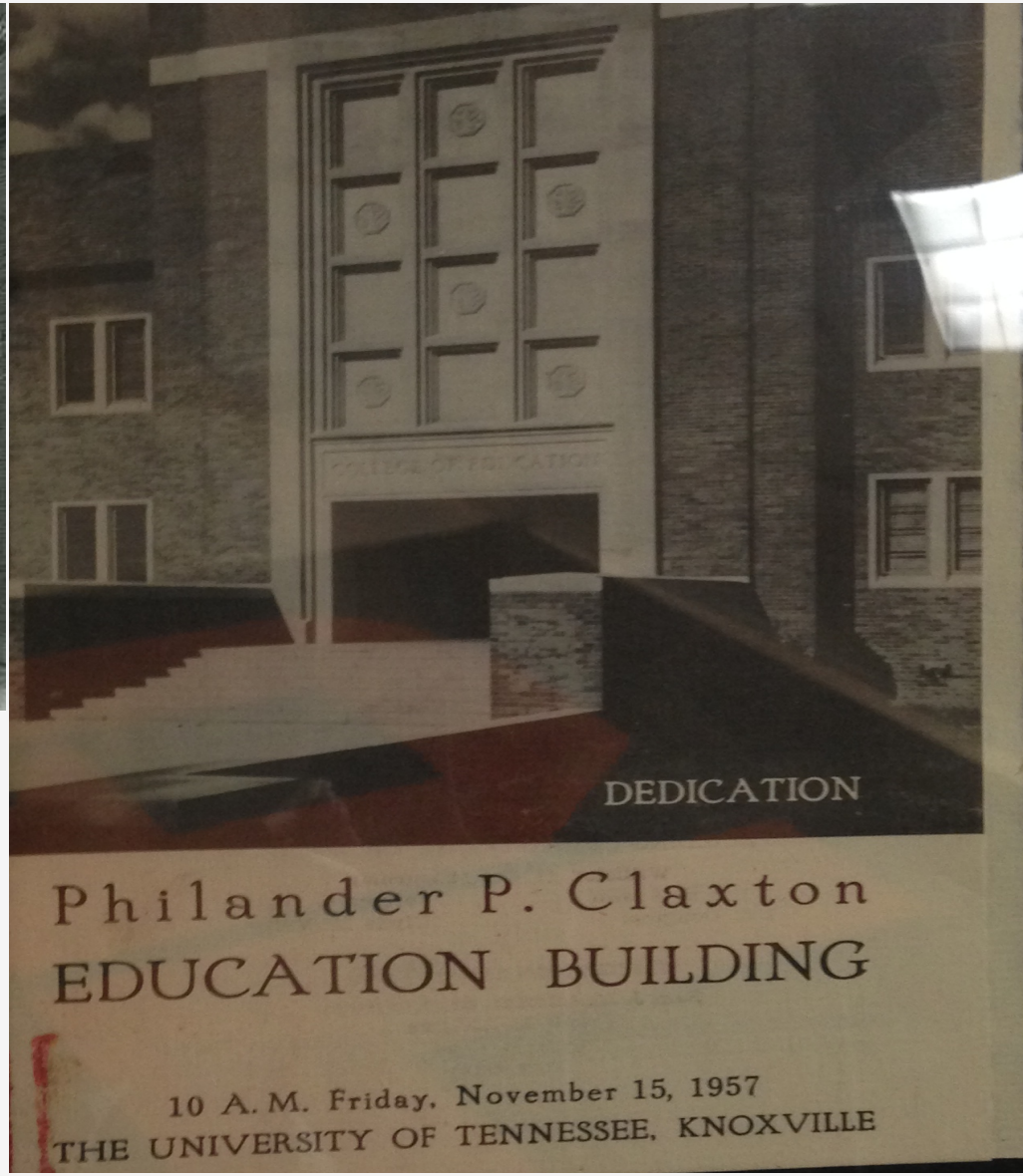
MAGMA: LU factorization for small matrices

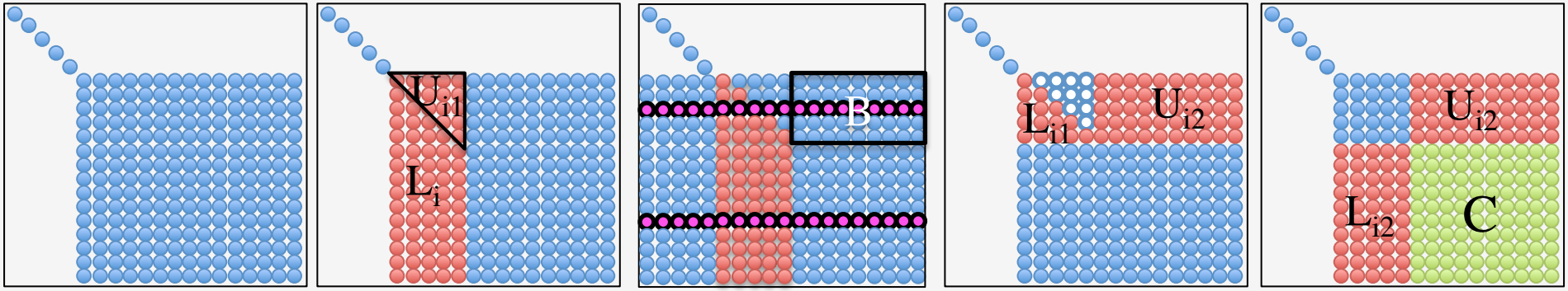


The PLASMA meeting

consistency over 57 years

same day and same time 10 A.M.

The LU Factorization xGETRF



panel facto
$PA_i = L_iU_{i1}$
dgetf2

swapping rows

dlaswp

Triangular
update $U_{i2}=L_{i1}^{-1}B$
dtrsm

schur update
$C = C - L_{i2}\,U_{i2}$
dgemm

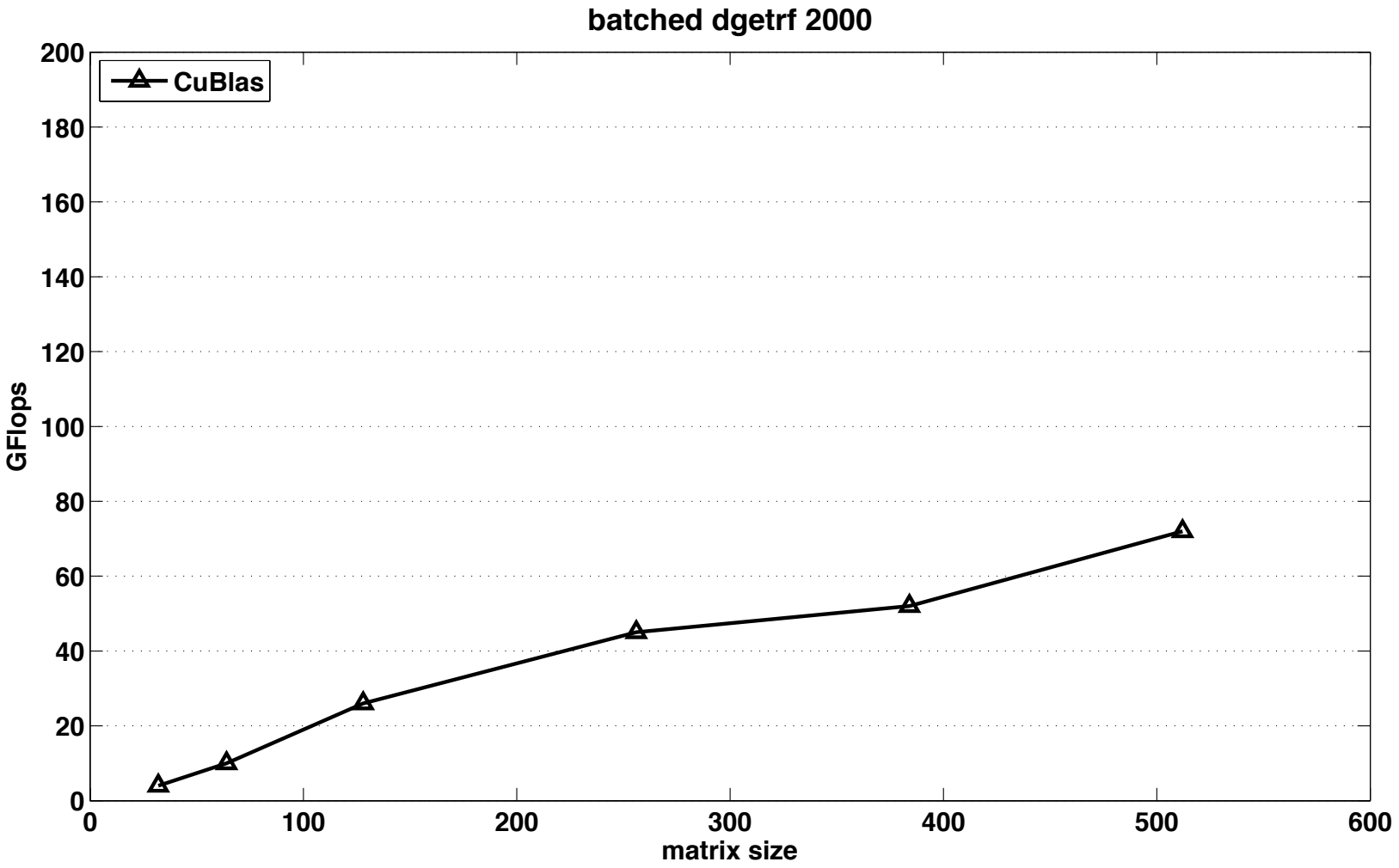
Main points

- Panel factorization mostly sequential due to memory bottleneck
- Triangular solve has little parallelism
- Schur complement update is the only easy parallelize task
- Partial pivoting complicates things even further
- Bulk synchronous parallelism (fork-join)

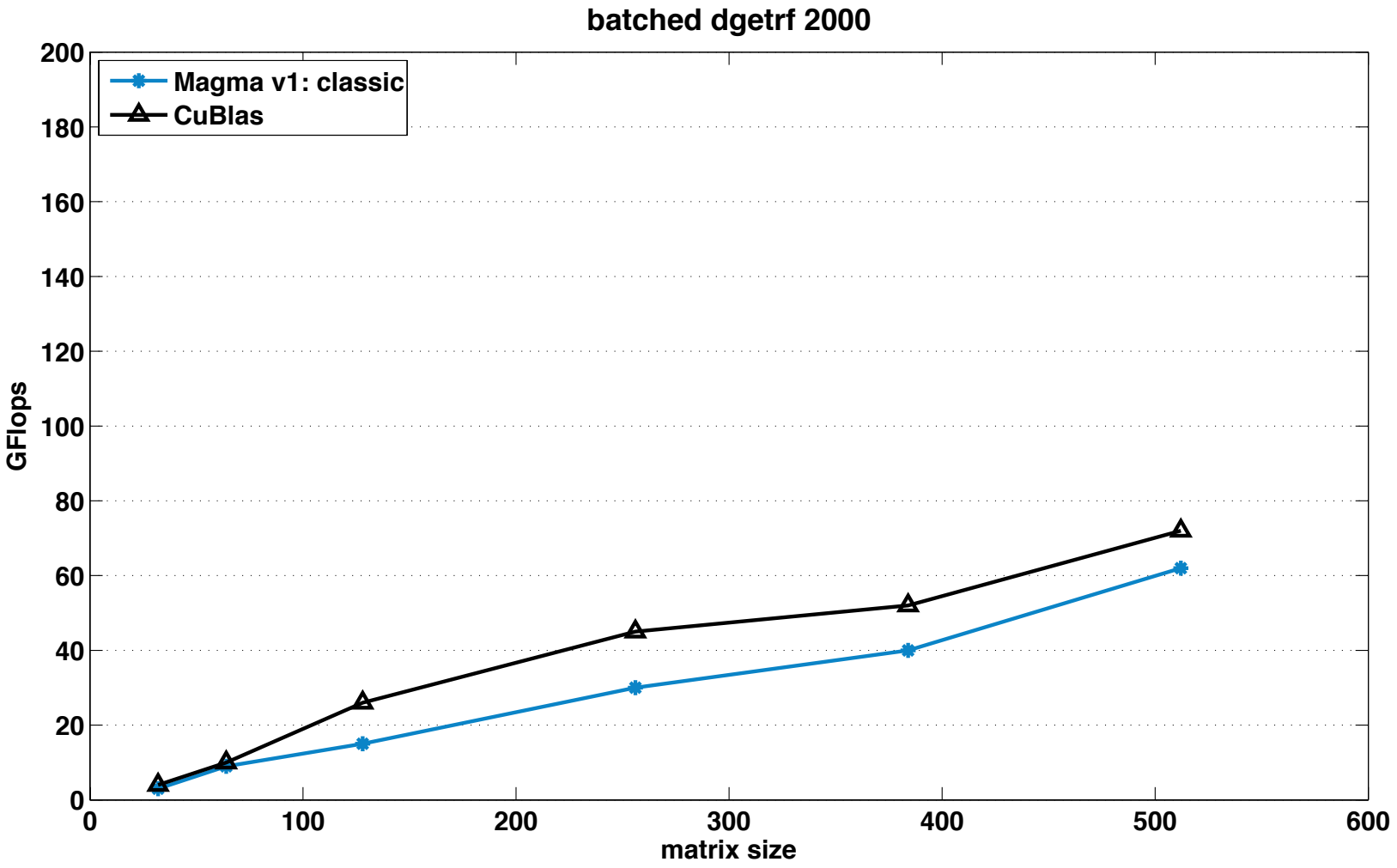
Total Cost of the Algorithm

- For each step it's the cost of the panel factorization
  - Each panel is of size NB and each column of the panel requires
    - 1 ***idmax*** for the column to find the max absolute value below the diagonal
    - 1 ***swap*** for two rows: current I and the row of the *idamax* output
    - 1 ***dscal*** for the column below the diagonal
    - 1 ***dger*** to update the remaining part of the panel
    - Total number of operations for the panel is $(M-{}^{1}/_{3}NB)NB^2 + O(NB^2) = O(M)$
  - The ***trsm*** is simply a gemm with the inverse of L: $B = L^{-1}B$
    - cost is NB*NB*N
  - The update is simply
    - 1 ***dgemm*** that updates the trailing matrix, cost is (M-NB)(N-NB)NB
- Total cost of the algorithm is $mn^2-1/3n^3$.
  for m=n it is $2/3n^3$

Performance

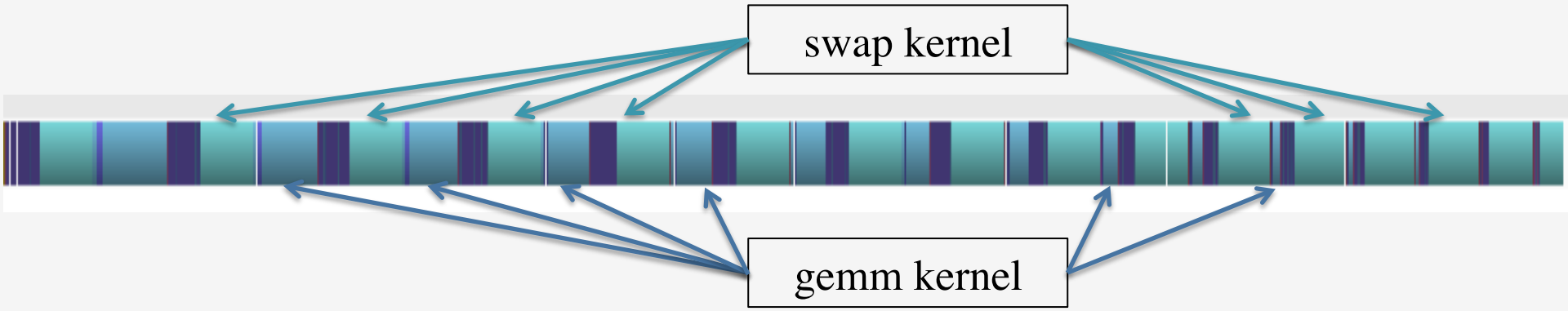


Performance



Profiler tracing

swap kernel *dlaswp*:



Main points

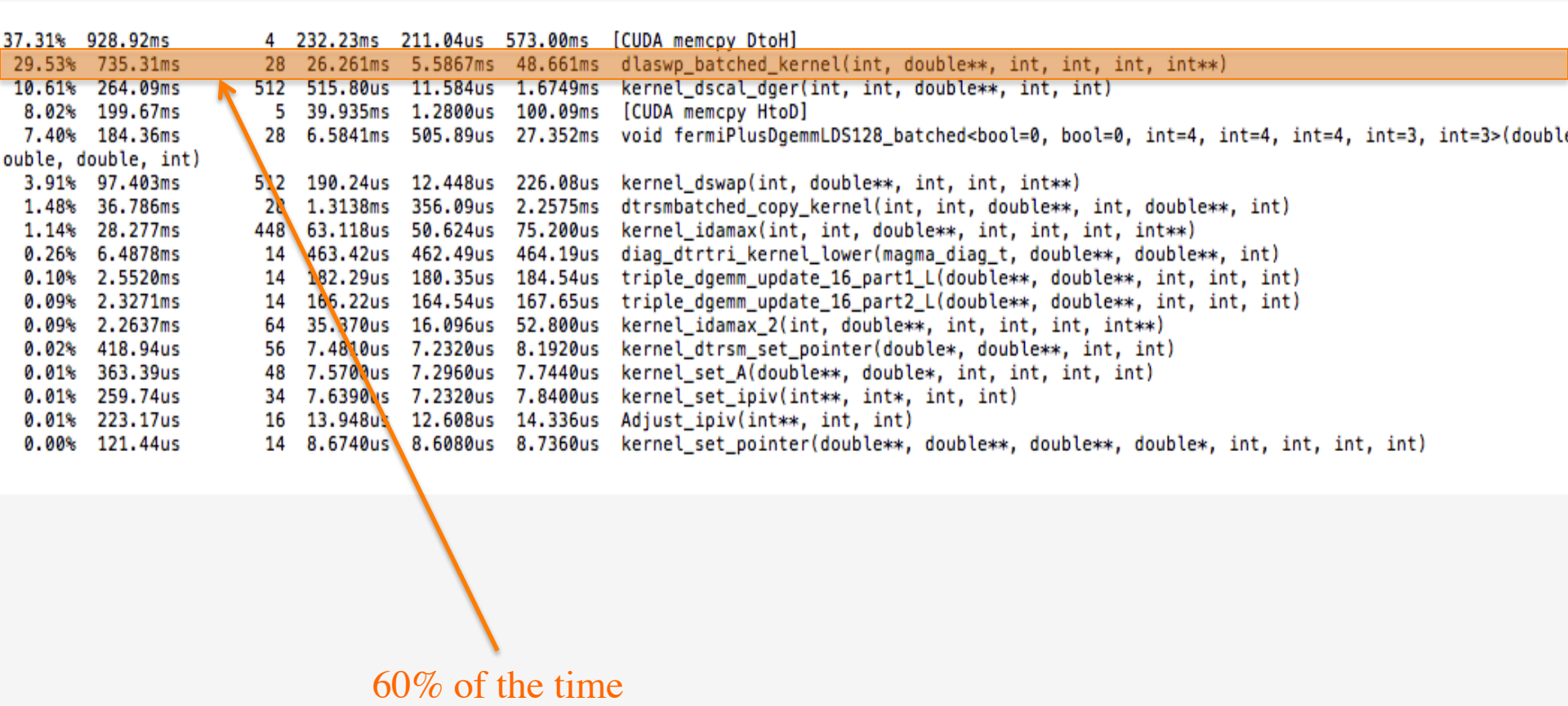
- Swap is expensive, it cost around 60% of the total time

Profiler details

```
37.31%  928.92ms         4  232.23ms  211.04us  573.00ms  [CUDA memcpy DtoH]
 29.53%  735.31ms        28  26.261ms  5.5867ms  48.661ms  dlaswp_batched_kernel(int, double**, int, int, int, int**)
 10.61%  264.09ms       512  515.80us  11.584us  1.6749ms  kernel_dscal_dger(int, int, double**, int, int)
  8.02%  199.67ms         5  39.935ms  1.2800us  100.09ms  [CUDA memcpy HtoD]
  7.40%  184.36ms        28  6.5841ms  505.89us  27.352ms  void fermiPlusDgemmLDS128_batched<bool=0, bool=0, int=4, int=4, int=4, int=3, int=3>(double
ouble, double, int)
  3.91%  97.403ms       512  190.24us  12.448us  226.08us  kernel_dswap(int, double**, int, int, int**)
  1.48%  36.786ms        28  1.3138ms  356.09us  2.2575ms  dtrsmbatched_copy_kernel(int, int, double**, int, double**, int)
  1.14%  28.277ms       448  63.118us  50.624us  75.200us  kernel_idamax(int, int, double**, int, int, int, int**)
  0.26%  6.4878ms        14  463.42us  462.49us  464.19us  diag_dtrtri_kernel_lower(magma_diag_t, double**, double**, int)
  0.10%  2.5520ms        14  182.29us  180.35us  184.54us  triple_dgemm_update_16_part1_L(double**, double**, int, int, int)
  0.09%  2.3271ms        14  166.22us  164.54us  167.65us  triple_dgemm_update_16_part2_L(double**, double**, int, int, int)
  0.09%  2.2637ms        64  35.370us  16.096us  52.800us  kernel_idamax_2(int, double**, int, int, int, int**)
  0.02%  418.94us        56  7.4810us  7.2320us  8.1920us  kernel_dtrsm_set_pointer(double*, double**, int, int)
  0.01%  363.39us        48  7.5700us  7.2960us  7.7440us  kernel_set_A(double**, double*, int, int, int, int)
  0.01%  259.74us        34  7.6390us  7.2320us  7.8400us  kernel_set_ipiv(int**, int*, int, int)
  0.01%  223.17us        16  13.948us  12.608us  14.336us  Adjust_ipiv(int**, int, int)
  0.00%  121.44us        14  8.6740us  8.6080us  8.7360us  kernel_set_pointer(double**, double**, double**, double*, int, int, int, int)
```

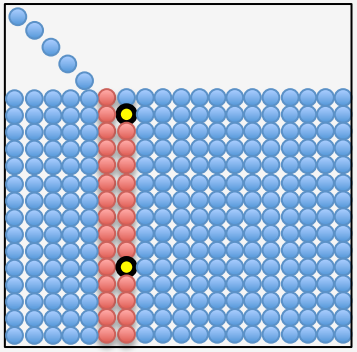
Profiler details

```
37.31%  928.92ms         4  232.23ms  211.04us  573.00ms  [CUDA memcpy DtoH]
29.53%  735.31ms        28  26.261ms  5.5867ms  48.661ms  dlaswp_batched_kernel(int, double**, int, int, int, int**)
10.61%  264.09ms       512  515.80us  11.584us  1.6749ms  kernel_dscal_dger(int, int, double**, int, int)
 8.02%  199.67ms         5  39.935ms  1.2800us  100.09ms  [CUDA memcpy HtoD]
 7.40%  184.36ms        28  6.5841ms  505.89us  27.352ms  void fermiPlusDgemmLDS128_batched<bool=0, bool=0, int=4, int=4, int=4, int=3, int=3>(double
ouble, double, int)
 3.91%  97.403ms       512  190.24us  12.448us  226.08us  kernel_dswap(int, double**, int, int, int**)
 1.48%  36.786ms        28  1.3138ms  356.09us  2.2575ms  dtrsmbatched_copy_kernel(int, int, double**, int, double**, int)
 1.14%  28.277ms       448  63.118us  50.624us  75.200us  kernel_idamax(int, int, double**, int, int, int, int**)
 0.26%  6.4878ms        14  463.42us  462.49us  464.19us  diag_dtrtri_kernel_lower(magma_diag_t, double**, double**, int)
 0.10%  2.5520ms        14  182.29us  180.35us  184.54us  triple_dgemm_update_16_part1_L(double**, double**, int, int, int)
 0.09%  2.3271ms        14  166.22us  164.54us  167.65us  triple_dgemm_update_16_part2_L(double**, double**, int, int, int)
 0.09%  2.2637ms        64  35.370us  16.096us  52.800us  kernel_idamax_2(int, double**, int, int, int, int**)
 0.02%  418.94us        56  7.4810us  7.2320us  8.1920us  kernel_dtrsm_set_pointer(double*, double**, int, int)
 0.01%  363.39us        48  7.5700us  7.2960us  7.7440us  kernel_set_A(double**, double*, int, int, int, int)
 0.01%  259.74us        34  7.6390us  7.2320us  7.8400us  kernel_set_ipiv(int**, int*, int, int)
 0.01%  223.17us        16  13.948us  12.608us  14.336us  Adjust_ipiv(int**, int, int)
 0.00%  121.44us        14  8.6740us  8.6080us  8.7360us  kernel_set_pointer(double**, double**, double**, double*, int, int, int, int)
```



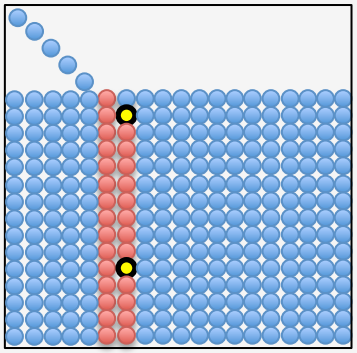
LU details

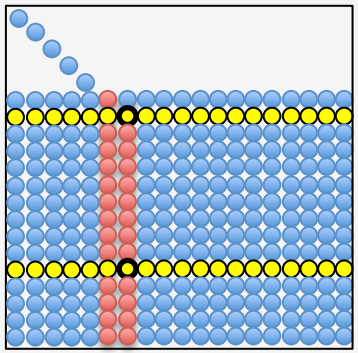
How the swap work?



LU details

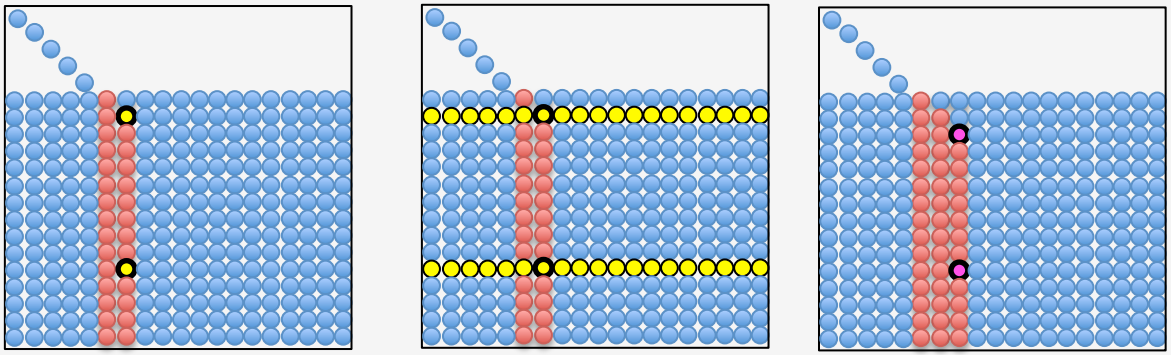
How the swap work?





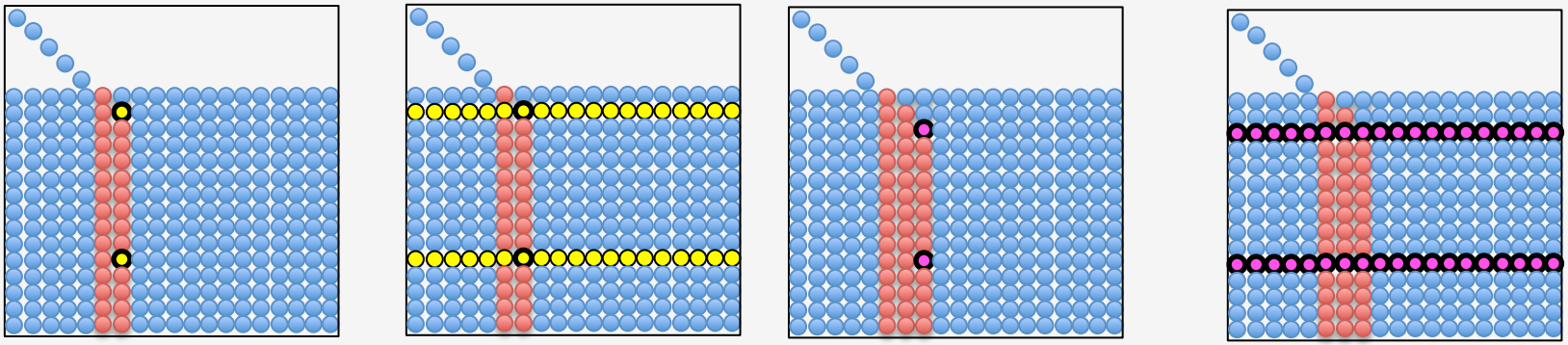
LU details

How the swap work?



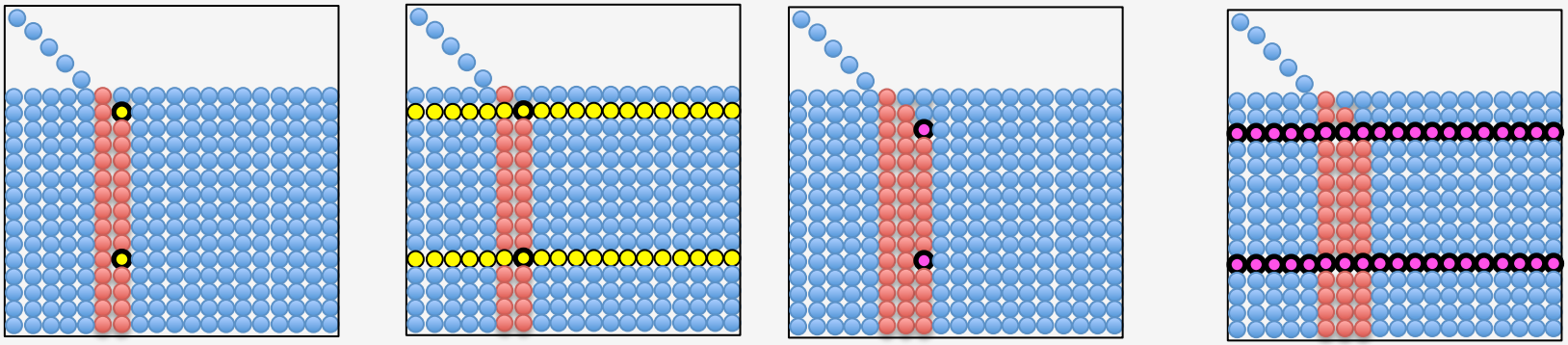
LU details

How the swap work?



LU details

How the swap work?



Bottlenecks:

- Swap is sequential and data dependent.

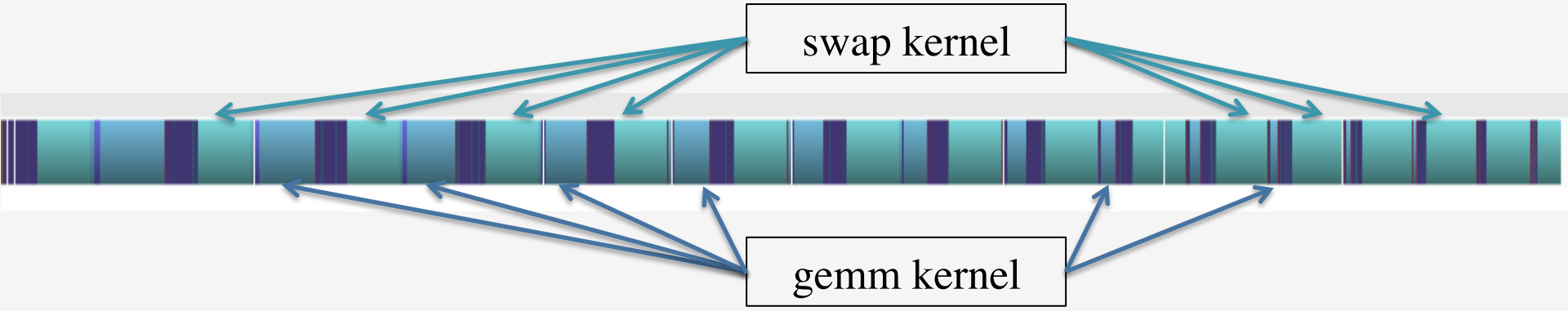
- Data is not coalescent: a GPU warp cannot read 32 value a the same time unless matrix is stored in transpose. However if matrix is stored in transpose the swap is fast BUT the other components becomes very slow.

Proposition:

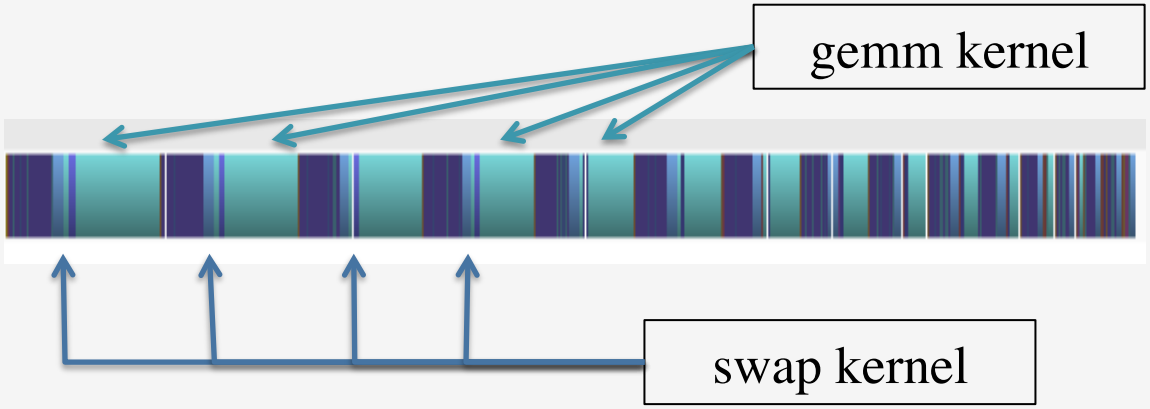
- Develop a parallel version of swap.
- Also it improves the write back of the swapped rows as the data is now coalescent.

Profiler tracing

sequential swap:

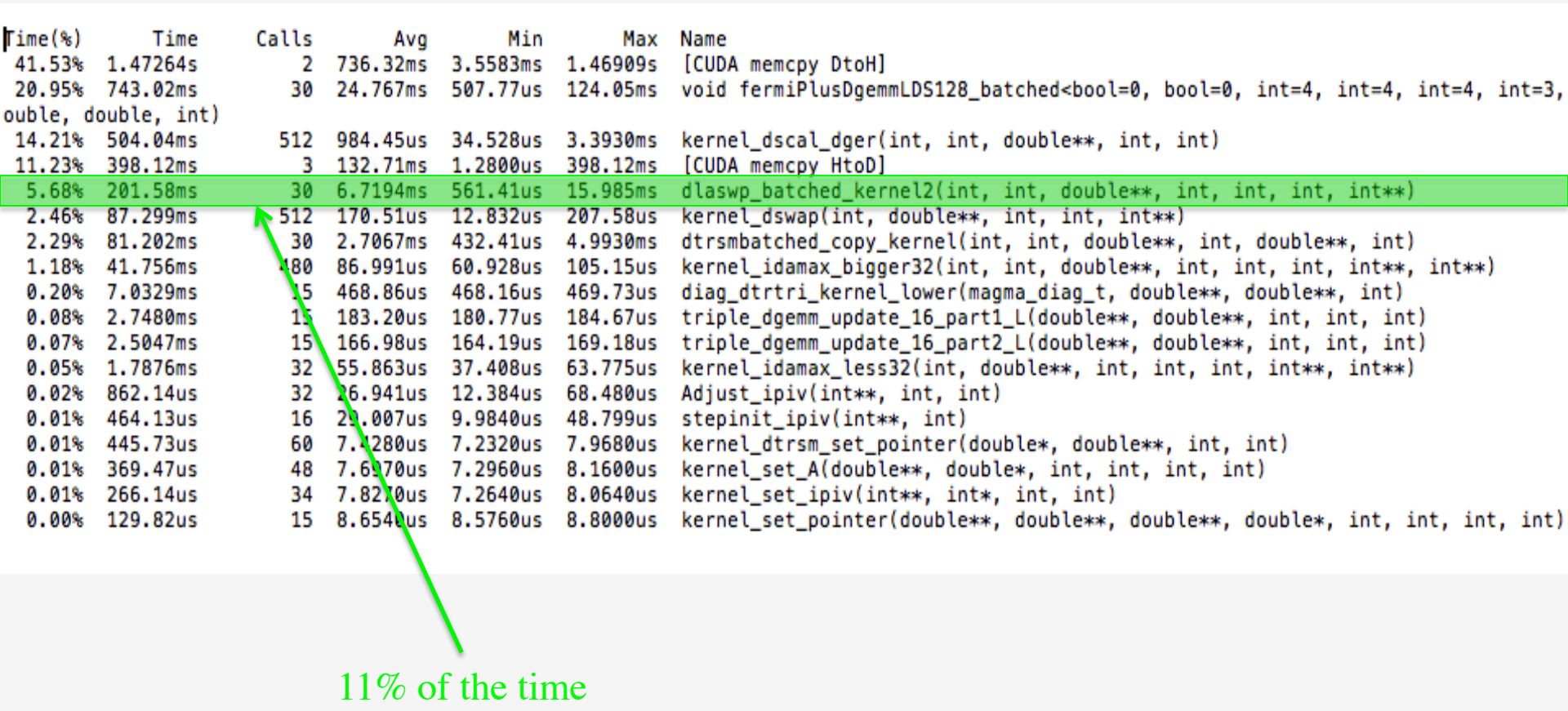


parallel swap:

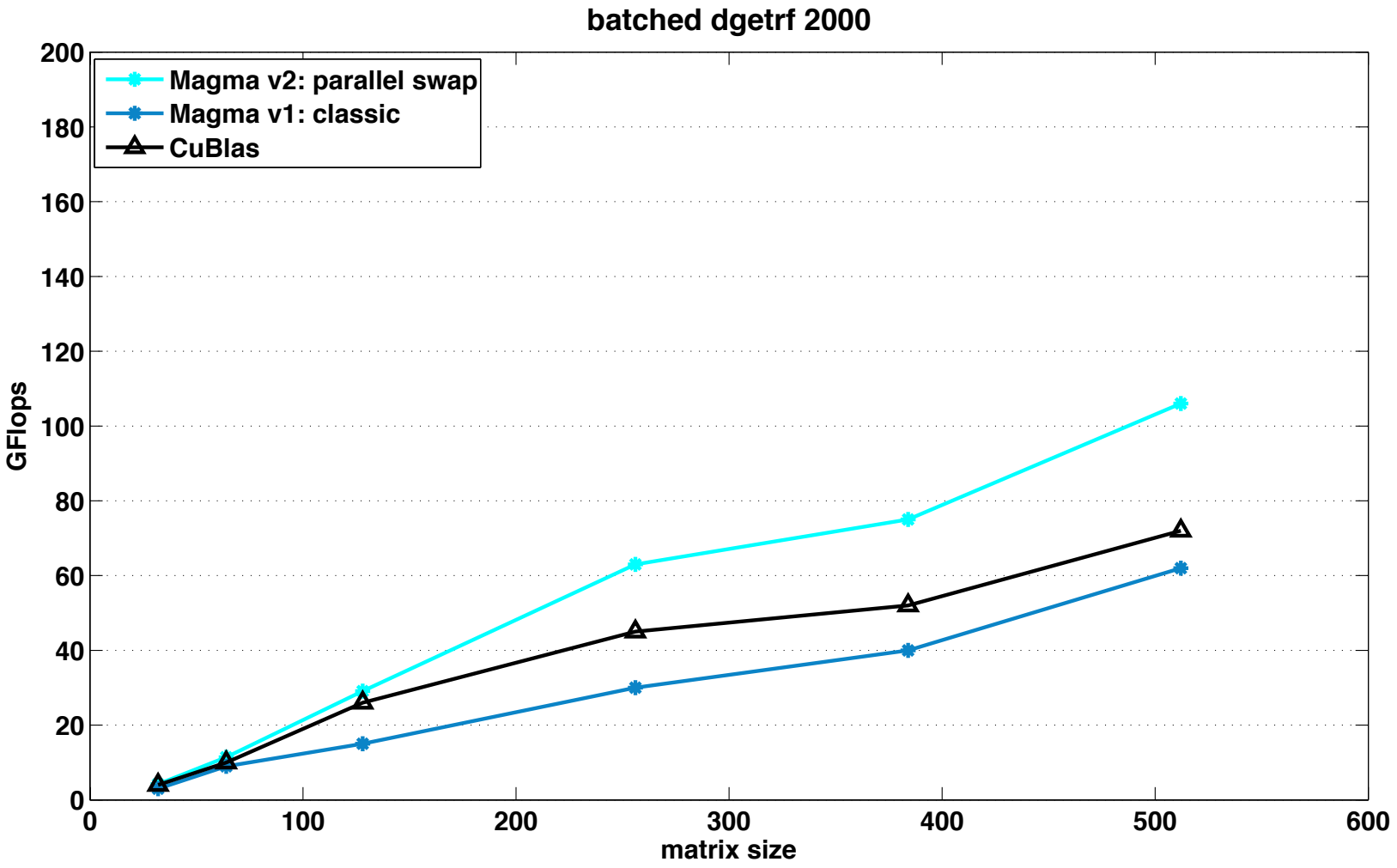


Profiler details

Time(%)	Time	Calls	Avg	Min	Max	Name
41.53%	1.47264s	2	736.32ms	3.5583ms	1.46909s	[CUDA memcpy DtoH]
20.95%	743.02ms	30	24.767ms	507.77us	124.05ms	void fermiPlusDgemmLDS128_batched<bool=0, bool=0, int=4, int=4, int=4, int=3,
ouble, double, int)						
14.21%	504.04ms	512	984.45us	34.528us	3.3930ms	kernel_dscal_dger(int, int, double**, int, int)
11.23%	398.12ms	3	132.71ms	1.2800us	398.12ms	[CUDA memcpy HtoD]
5.68%	201.58ms	30	6.7194ms	561.41us	15.985ms	dlaswp_batched_kernel2(int, int, double**, int, int, int, int**)
2.46%	87.299ms	512	170.51us	12.832us	207.58us	kernel_dswap(int, double**, int, int, int**)
2.29%	81.202ms	30	2.7067ms	432.41us	4.9930ms	dtrsmbatched_copy_kernel(int, int, double**, int, double**, int)
1.18%	41.756ms	480	86.991us	60.928us	105.15us	kernel_idamax_bigger32(int, int, double**, int, int, int, int**, int**)
0.20%	7.0329ms	15	468.86us	468.16us	469.73us	diag_dtrtri_kernel_lower(magma_diag_t, double**, double**, int)
0.08%	2.7480ms	15	183.20us	180.77us	184.67us	triple_dgemm_update_16_part1_L(double**, double**, int, int, int)
0.07%	2.5047ms	15	166.98us	164.19us	169.18us	triple_dgemm_update_16_part2_L(double**, double**, int, int, int)
0.05%	1.7876ms	32	55.863us	37.408us	63.775us	kernel_idamax_less32(int, double**, int, int, int, int**, int**)
0.02%	862.14us	32	26.941us	12.384us	68.480us	Adjust_ipiv(int**, int, int)
0.01%	464.13us	16	29.007us	9.9840us	48.799us	stepinit_ipiv(int**, int)
0.01%	445.73us	60	7.4280us	7.2320us	7.9680us	kernel_dtrsm_set_pointer(double*, double**, int, int)
0.01%	369.47us	48	7.6970us	7.2960us	8.1600us	kernel_set_A(double**, double*, int, int, int, int)
0.01%	266.14us	34	7.8270us	7.2640us	8.0640us	kernel_set_ipiv(int**, int*, int, int)
0.00%	129.82us	15	8.6540us	8.5760us	8.8000us	kernel_set_pointer(double**, double**, double**, double*, int, int, int, int)

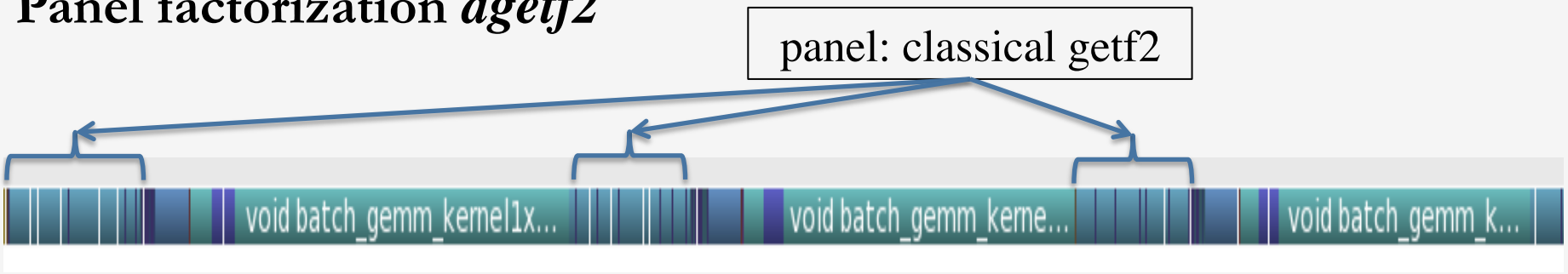


Performance



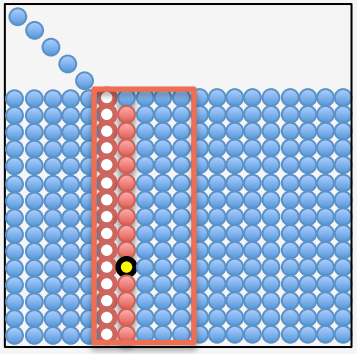
Profiler tracing

Panel factorization *dgetf2*



LU details

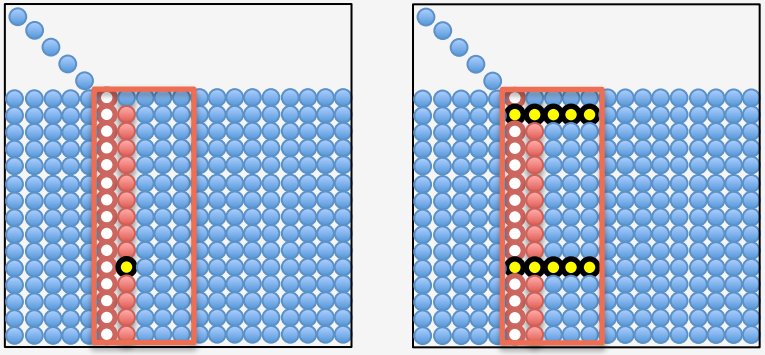
How the dgetf2 work?



1. find the max absolute value of a column below the diag “pivots”

LU details

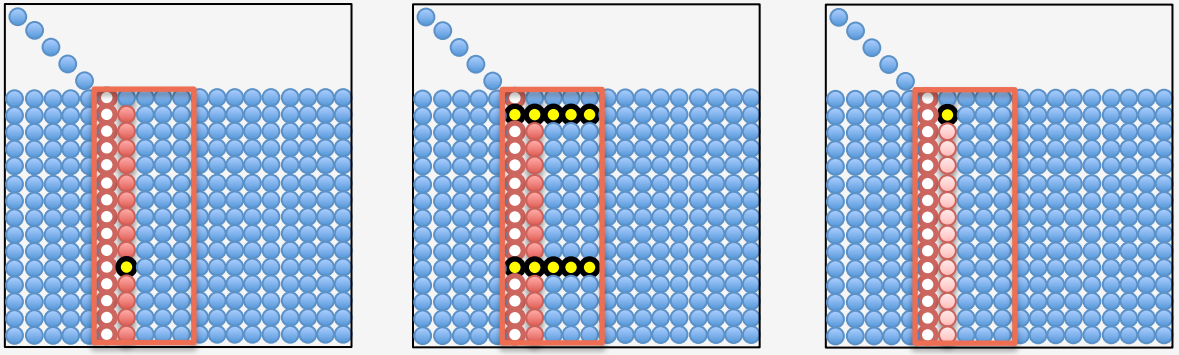
How the dgetf2 work?



1. find the max absolute value of a column below the diag "pivots"
2. swap the row of size *nb*

LU details

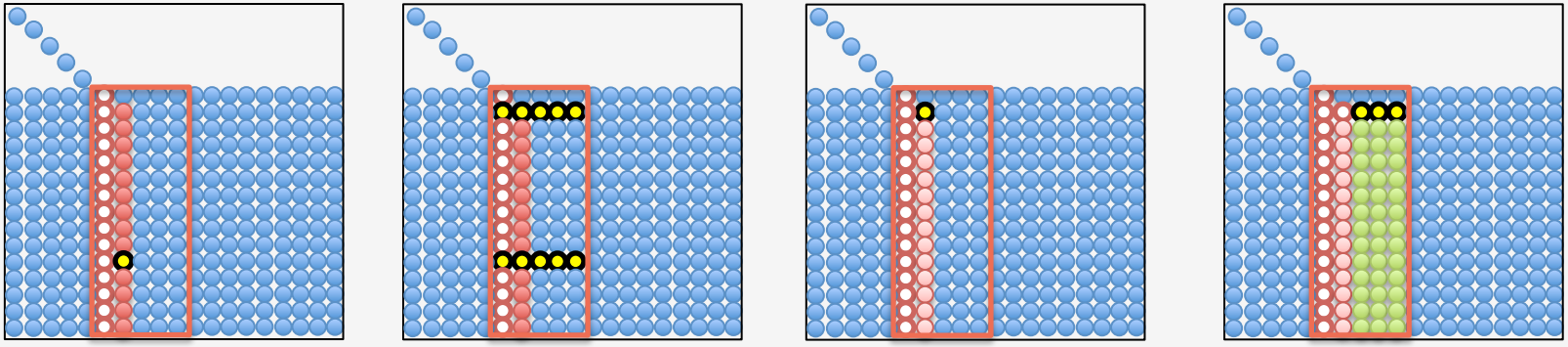
How the dgetf2 work?



1. find the max absolute value of a column below the diag "pivots"
2. swap the row of size *nb*
3. scal the column below the diag by the inverse of the pivots

LU details

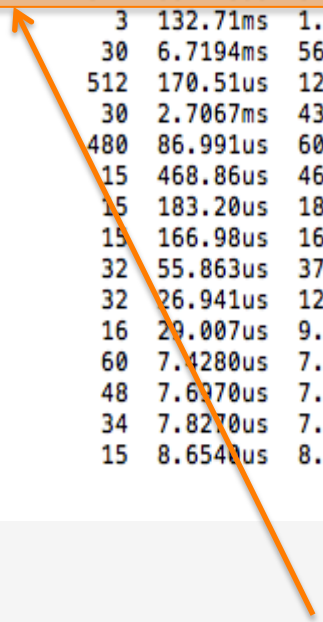
How the dgetf2 work?



1. find the max absolute value of a column below the diag "pivots"
2. swap the row of size *nb*
3. scal the column below the diag by the inverse of the pivots
4. update the right of the column dger

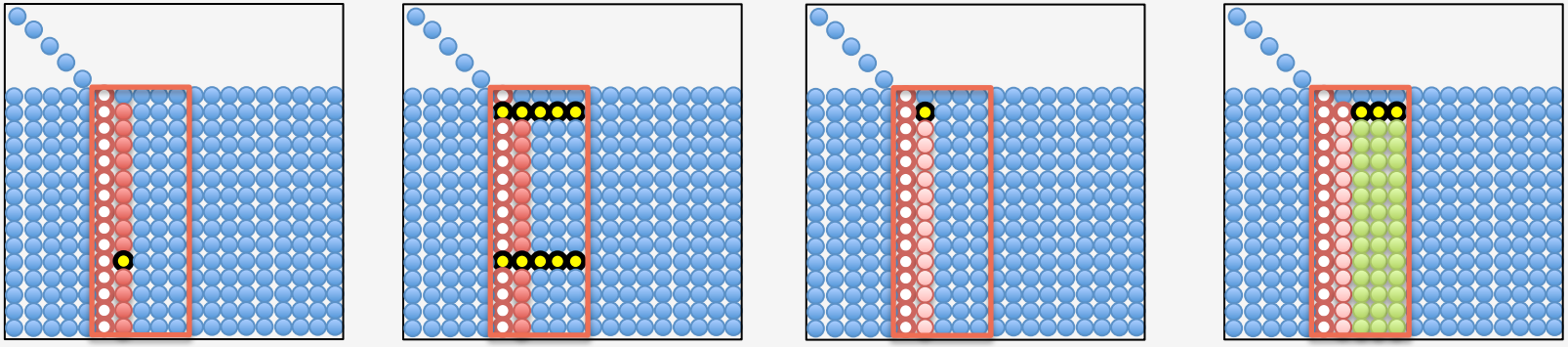
Profiler details

Time(%)	Time	Calls	Avg	Min	Max	Name
41.53%	1.47264s	2	736.32ms	3.5583ms	1.46909s	[CUDA memcpy DtoH]
20.95%	743.02ms	30	24.767ms	507.77us	124.05ms	void fermiPlusDgemmLDS128_batched<bool=0, bool=0, int=4, int=4, int=4, int=3,
ouble, double, int)						
14.21%	504.04ms	512	984.45us	34.528us	3.3930ms	kernel_dscal_dger(int, int, double**, int, int)
11.23%	398.12ms	3	132.71ms	1.2800us	398.12ms	[CUDA memcpy HtoD]
5.68%	201.58ms	30	6.7194ms	561.41us	15.985ms	dlaswp_batched_kernel2(int, int, double**, int, int, int, int**)
2.46%	87.299ms	512	170.51us	12.832us	207.58us	kernel_dswap(int, double**, int, int, int**)
2.29%	81.202ms	30	2.7067ms	432.41us	4.9930ms	dtrsmbatched_copy_kernel(int, int, double**, int, double**, int)
1.18%	41.756ms	480	86.991us	60.928us	105.15us	kernel_idamax_bigger32(int, int, double**, int, int, int, int**, int**)
0.20%	7.0329ms	15	468.86us	468.16us	469.73us	diag_dtrtri_kernel_lower(magma_diag_t, double**, double**, int)
0.08%	2.7480ms	15	183.20us	180.77us	184.67us	triple_dgemm_update_16_part1_L(double**, double**, int, int, int)
0.07%	2.5047ms	15	166.98us	164.19us	169.18us	triple_dgemm_update_16_part2_L(double**, double**, int, int, int)
0.05%	1.7876ms	32	55.863us	37.408us	63.775us	kernel_idamax_less32(int, double**, int, int, int, int**, int**)
0.02%	862.14us	32	26.941us	12.384us	68.480us	Adjust_ipiv(int**, int, int)
0.01%	464.13us	16	29.007us	9.9840us	48.799us	stepinit_ipiv(int**, int)
0.01%	445.73us	60	7.4280us	7.2320us	7.9680us	kernel_dtrsm_set_pointer(double*, double**, int, int)
0.01%	369.47us	48	7.6970us	7.2960us	8.1600us	kernel_set_A(double**, double*, int, int, int, int)
0.01%	266.14us	34	7.8270us	7.2640us	8.0640us	kernel_set_ipiv(int**, int*, int, int)
0.00%	129.82us	15	8.6540us	8.5760us	8.8000us	kernel_set_pointer(double**, double**, double**, double*, int, int, int, int)



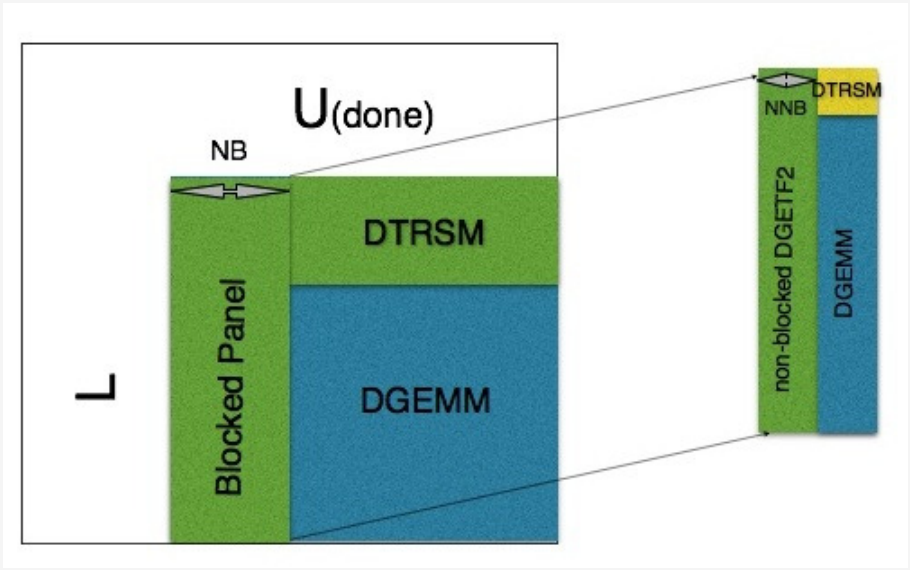
LU details

How the dgetf2 work?



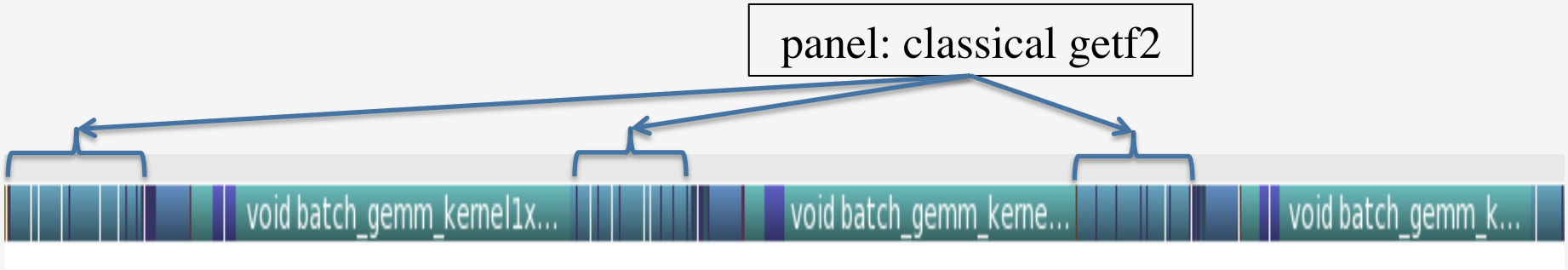
Proposition:

- Nested blocking fashion:
  Develop a nested blocking technique that block also the panel in order to replace the dger kernel by dgemm kernel.

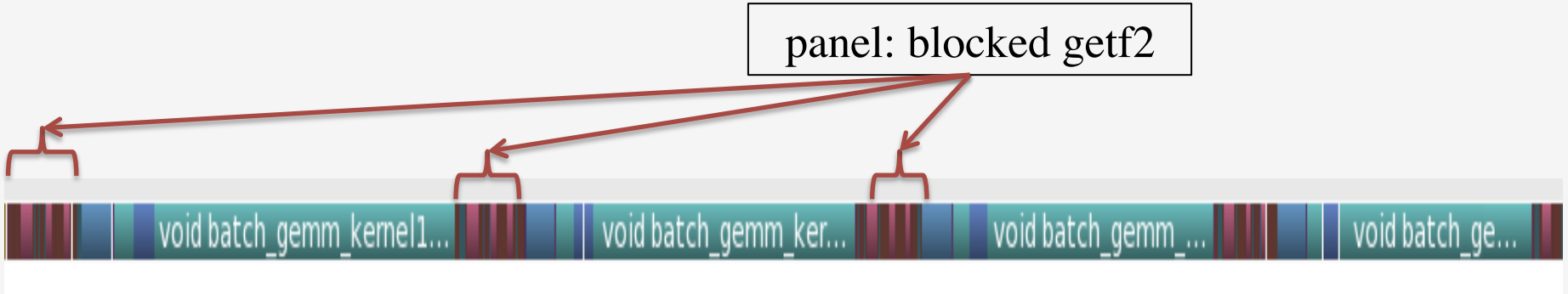


Profiler tracing

Classical dgetf2:

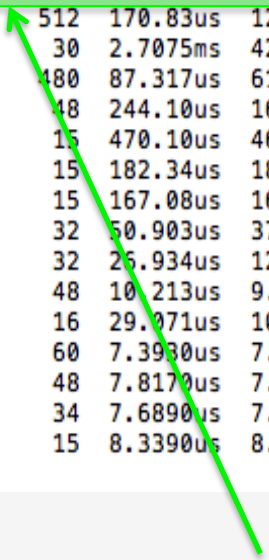


Blocked dgetf2:



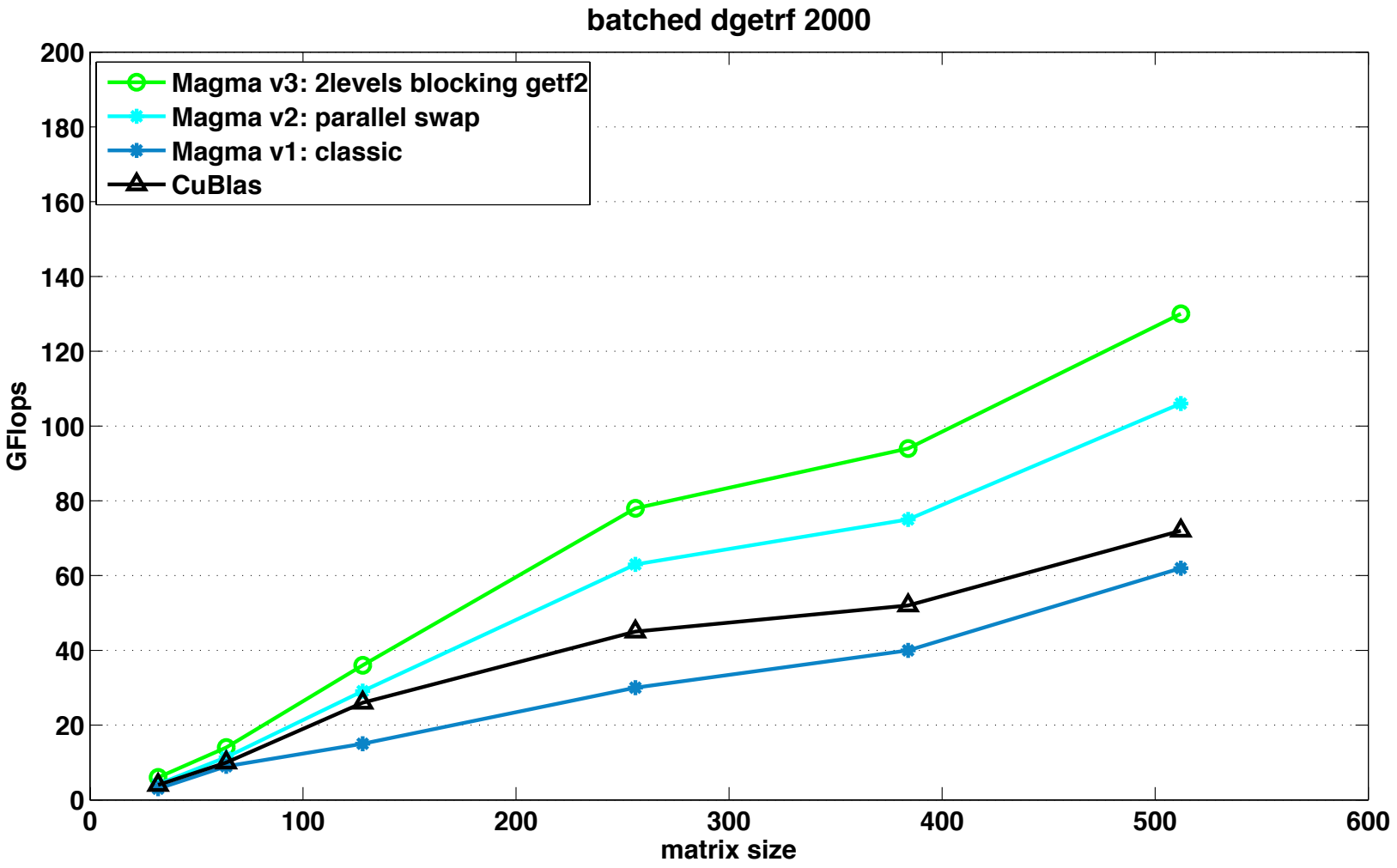
Profiler details

Time(%)	Time	Calls	Avg	Min	Max	Name
44.36%	1.45537s	2	727.69ms	1.7322ms	1.45364s	[CUDA memcpy DtoH]
25.98%	852.33ms	78	10.927ms	95.840us	124.36ms	void fermiPlusDgemmLDS128_batched<bool=0, bool=0, int=4, int=4, int=4, int=3,
12.13%	397.99ms	3	132.67ms	1.3120us	397.99ms	[CUDA memcpy HtoD]
6.15%	201.67ms	30	6.7224ms	560.09us	15.989ms	dlaswp_batched_kernel2(int, int, double**, int, int, int, int**)
4.10%	134.46ms	512	262.62us	34.303us	903.87us	kernel_dscal_dger(int, int, double**, int, int)
2.67%	87.466ms	512	170.83us	12.864us	218.30us	kernel_dswap(int, double**, int, int, int**)
2.48%	81.224ms	30	2.7075ms	427.87us	4.9912ms	dtrsmbatched_copy_kernel(int, int, double**, int, double**, int)
1.28%	41.912ms	480	87.317us	61.248us	105.73us	kernel_idamax_bigger32(int, int, double**, int, int, int, int**, int**)
0.36%	11.717ms	48	244.10us	167.07us	321.09us	kernel_dgetf2trsm(int, int, double**, int, int)
0.21%	7.0515ms	15	470.10us	469.50us	470.78us	diag_dtrtri_kernel_lower(magma_diag_t, double**, double**, int)
0.08%	2.7350ms	15	182.34us	180.38us	183.87us	triple_dgemm_update_16_part1_L(double**, double**, int, int, int)
0.08%	2.5062ms	15	167.08us	164.77us	169.28us	triple_dgemm_update_16_part2_L(double**, double**, int, int, int)
0.05%	1.6289ms	32	50.903us	37.472us	63.104us	kernel_idamax_less32(int, double**, int, int, int, int**, int**)
0.03%	861.92us	32	26.934us	12.320us	68.287us	Adjust_ipiv(int**, int, int)
0.01%	490.24us	48	10.213us	9.9520us	10.464us	kernel_set_zgerpointer(double**, double**, double**, double**, int, int, int
0.01%	465.15us	16	29.071us	10.240us	48.992us	stepinit_ipiv(int**, int)
0.01%	443.61us	60	7.3930us	7.2320us	7.8400us	kernel_dtrsm_set_pointer(double*, double**, int, int)
0.01%	375.23us	48	7.8170us	7.2960us	8.0640us	kernel_set_A(double**, double*, int, int, int, int)
0.01%	261.44us	34	7.6890us	7.2640us	7.8400us	kernel_set_ipiv(int**, int*, int, int)
0.00%	125.09us	15	8.3390us	8.3200us	8.4160us	kernel_set_pointer(double**, double**, double**, double*, int, int, int, int



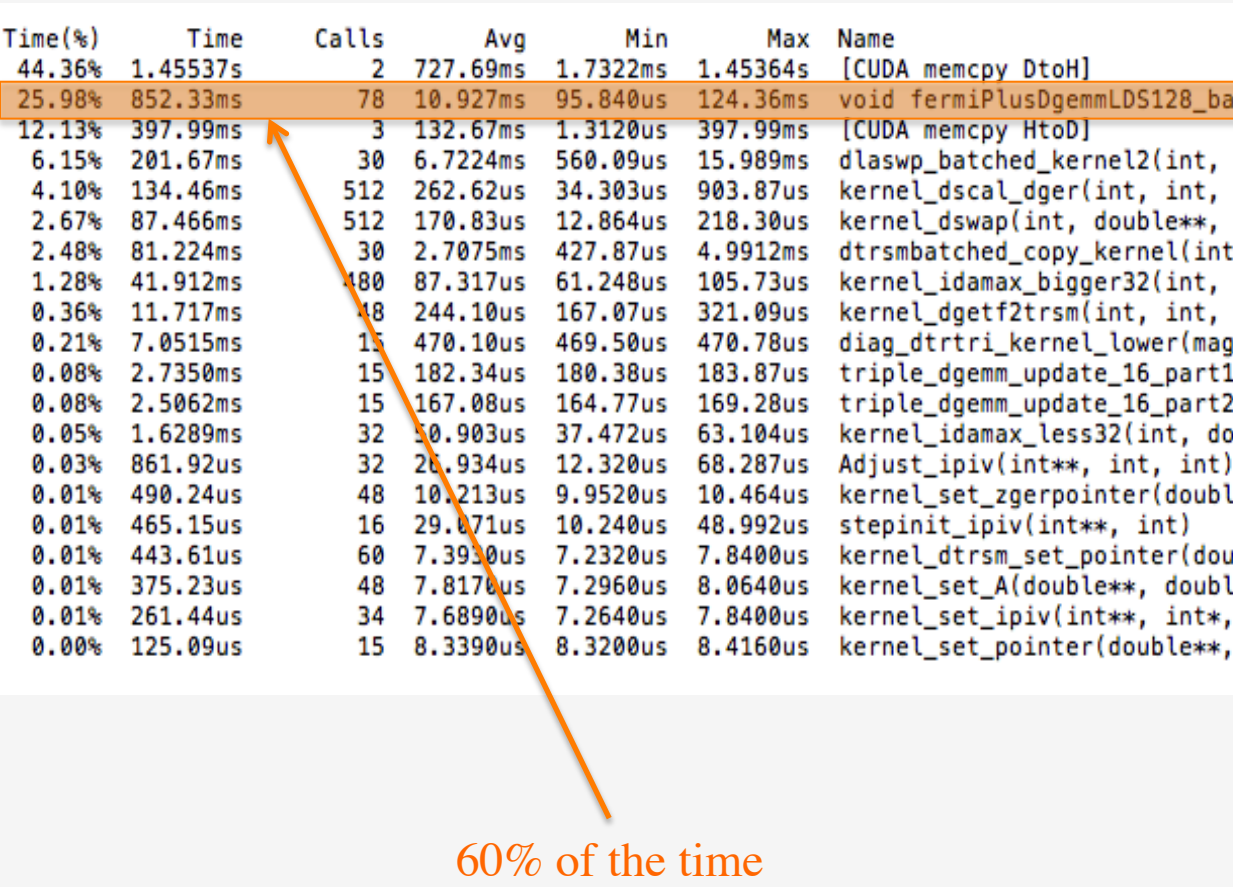
8% of the time

Performance

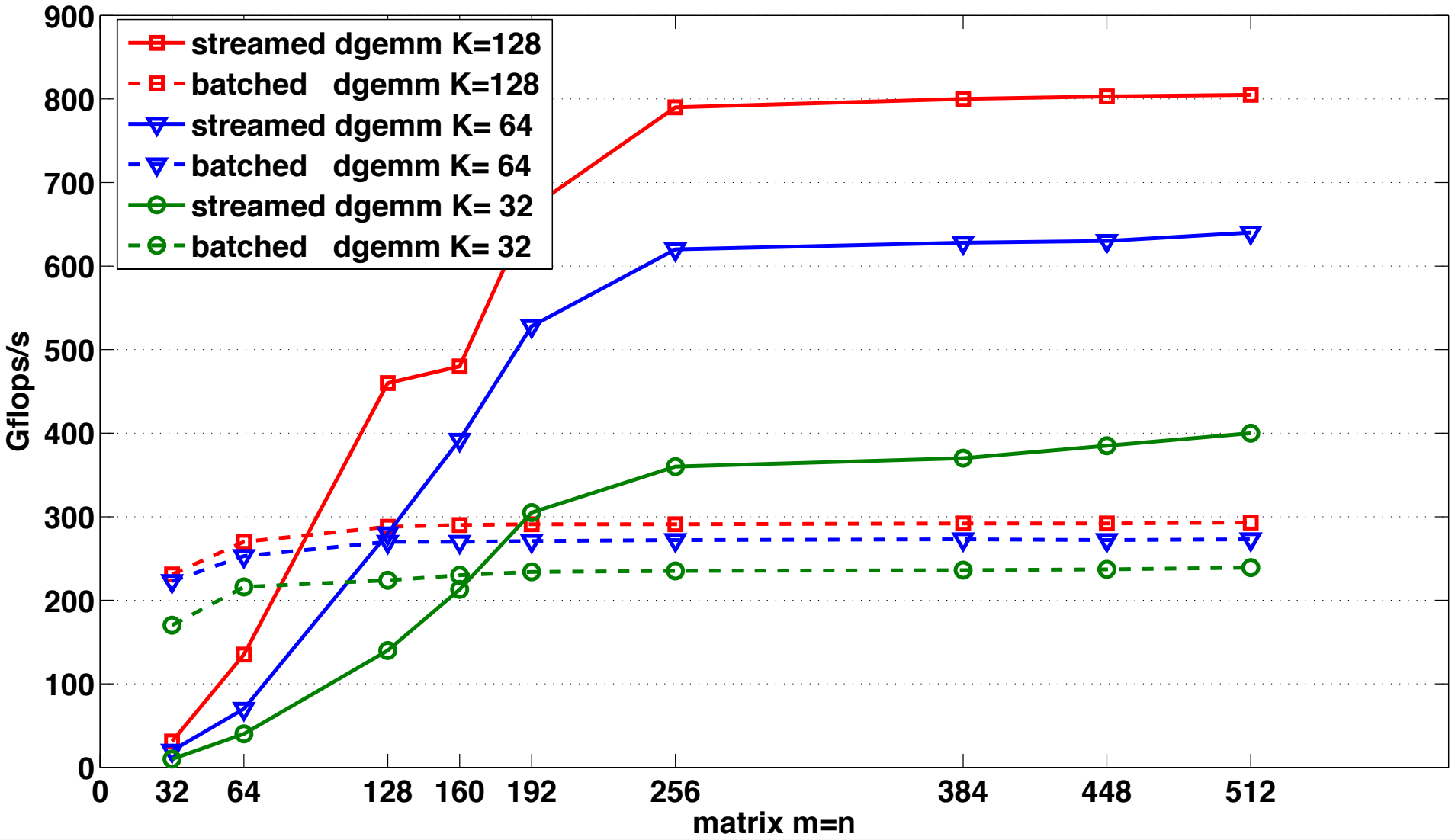


Profiler details

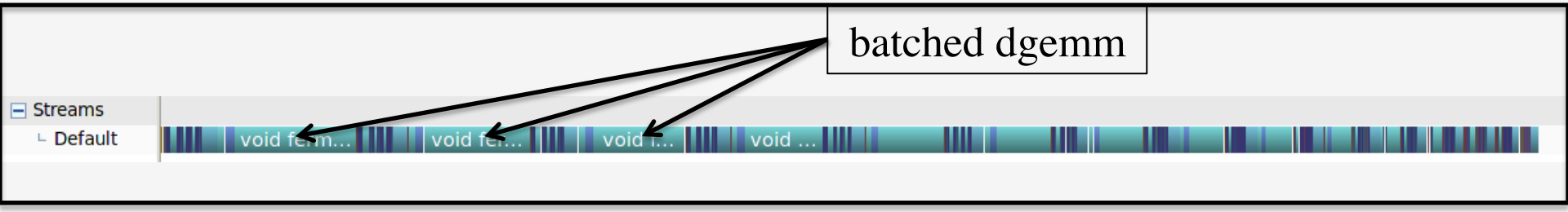
Time(%)	Time	Calls	Avg	Min	Max	Name
44.36%	1.45537s	2	727.69ms	1.7322ms	1.45364s	[CUDA memcpy DtoH]
25.98%	852.33ms	78	10.927ms	95.840us	124.36ms	void fermiPlusDgemmLDS128_batched<bool=0, bool=0, int=4, int=4, int=4, int=3
12.13%	397.99ms	3	132.67ms	1.3120us	397.99ms	[CUDA memcpy HtoD]
6.15%	201.67ms	30	6.7224ms	560.09us	15.989ms	dlaswp_batched_kernel2(int, int, double**, int, int, int, int**)
4.10%	134.46ms	512	262.62us	34.303us	903.87us	kernel_dscal_dger(int, int, double**, int, int)
2.67%	87.466ms	512	170.83us	12.864us	218.30us	kernel_dswap(int, double**, int, int, int**)
2.48%	81.224ms	30	2.7075ms	427.87us	4.9912ms	dtrsmbatched_copy_kernel(int, int, double**, int, double**, int)
1.28%	41.912ms	480	87.317us	61.248us	105.73us	kernel_idamax_bigger32(int, int, double**, int, int, int, int**, int**)
0.36%	11.717ms	48	244.10us	167.07us	321.09us	kernel_dgetf2trsm(int, int, double**, int, int)
0.21%	7.0515ms	15	470.10us	469.50us	470.78us	diag_dtrtri_kernel_lower(magma_diag_t, double**, double**, int)
0.08%	2.7350ms	15	182.34us	180.38us	183.87us	triple_dgemm_update_16_part1_L(double**, double**, int, int, int)
0.08%	2.5062ms	15	167.08us	164.77us	169.28us	triple_dgemm_update_16_part2_L(double**, double**, int, int, int)
0.05%	1.6289ms	32	50.903us	37.472us	63.104us	kernel_idamax_less32(int, double**, int, int, int, int**, int**)
0.03%	861.92us	32	26.934us	12.320us	68.287us	Adjust_ipiv(int**, int, int)
0.01%	490.24us	48	10.213us	9.9520us	10.464us	kernel_set_zgerpointer(double**, double**, double**, double**, int, int, int
0.01%	465.15us	16	29.071us	10.240us	48.992us	stepinit_ipiv(int**, int)
0.01%	443.61us	60	7.3930us	7.2320us	7.8400us	kernel_dtrsm_set_pointer(double*, double**, int, int)
0.01%	375.23us	48	7.8170us	7.2960us	8.0640us	kernel_set_A(double**, double*, int, int, int, int)
0.01%	261.44us	34	7.6890us	7.2640us	7.8400us	kernel_set_ipiv(int**, int*, int, int)
0.00%	125.09us	15	8.3390us	8.3200us	8.4160us	kernel_set_pointer(double**, double**, double**, double*, int, int, int, int

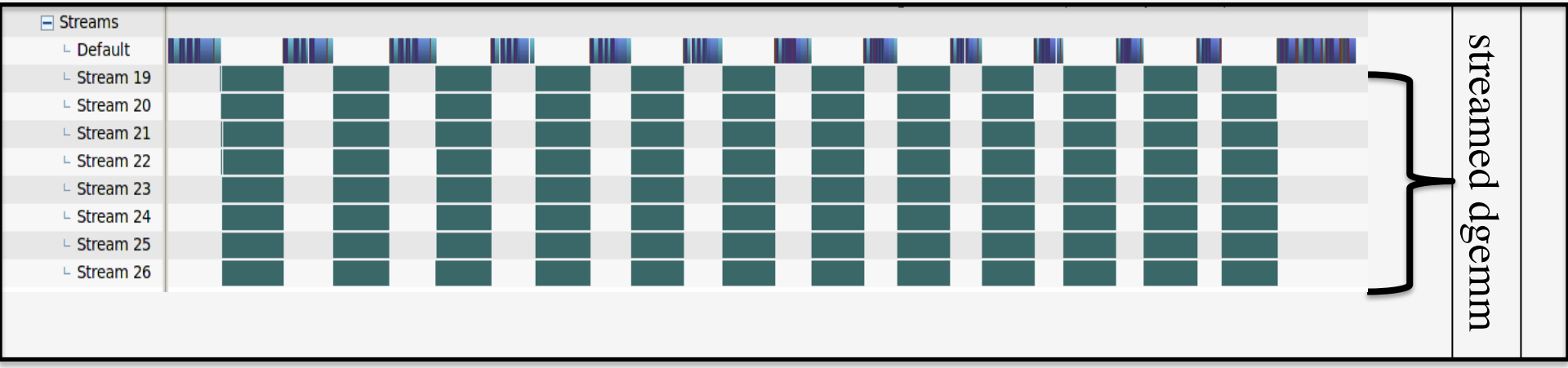


Performance

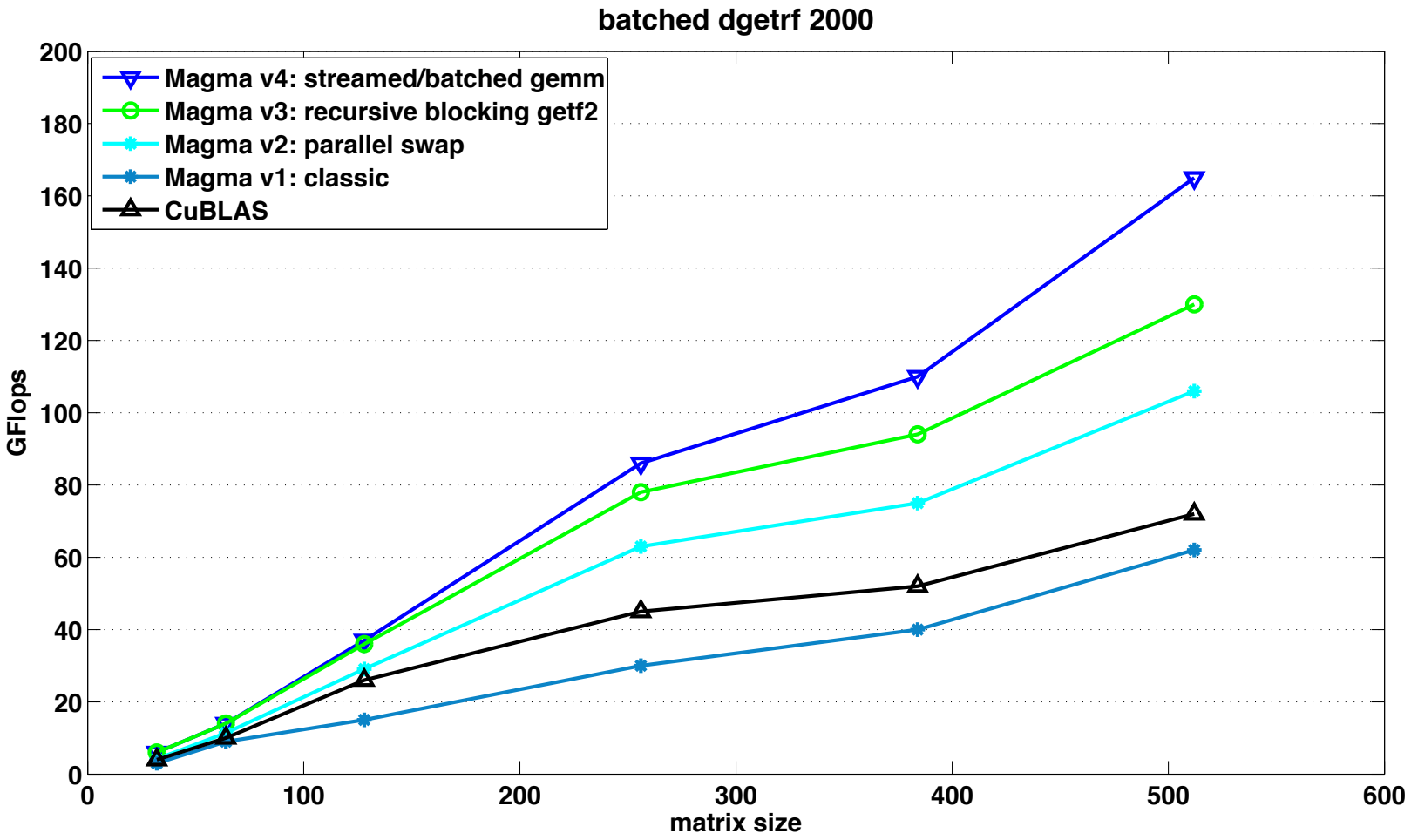


Profiler tracing

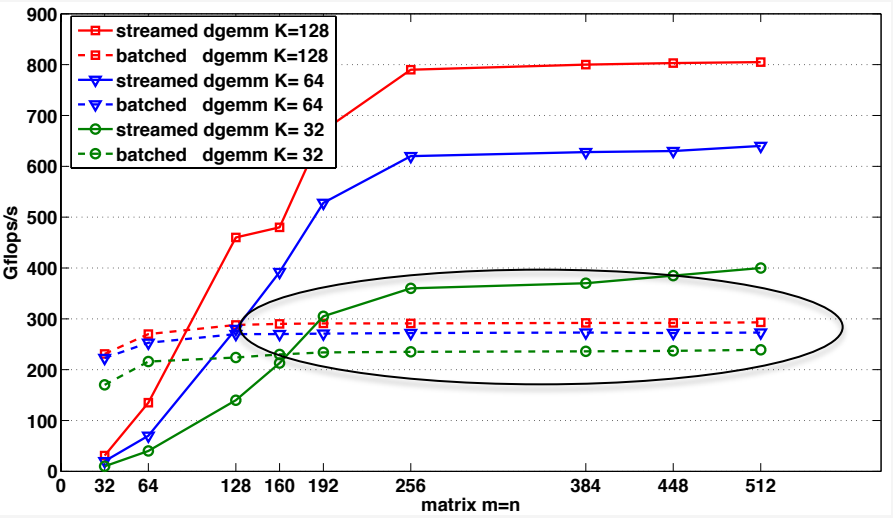




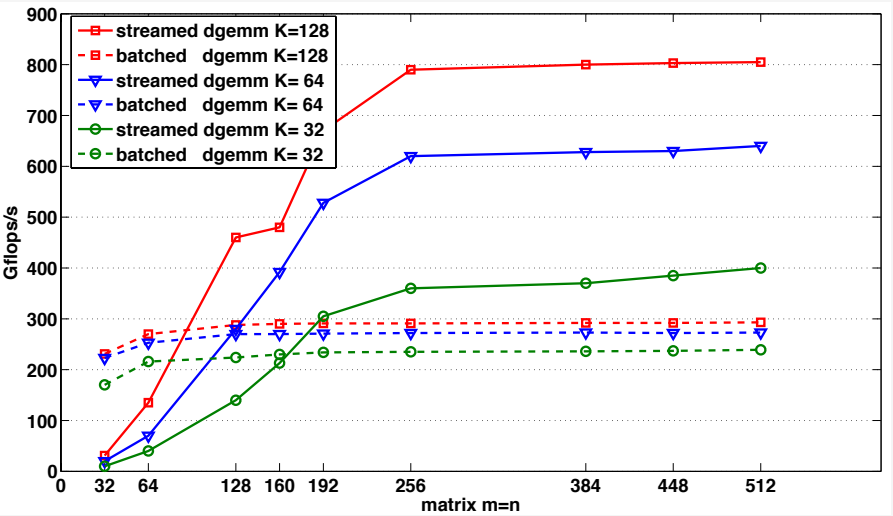
Performance



LU details

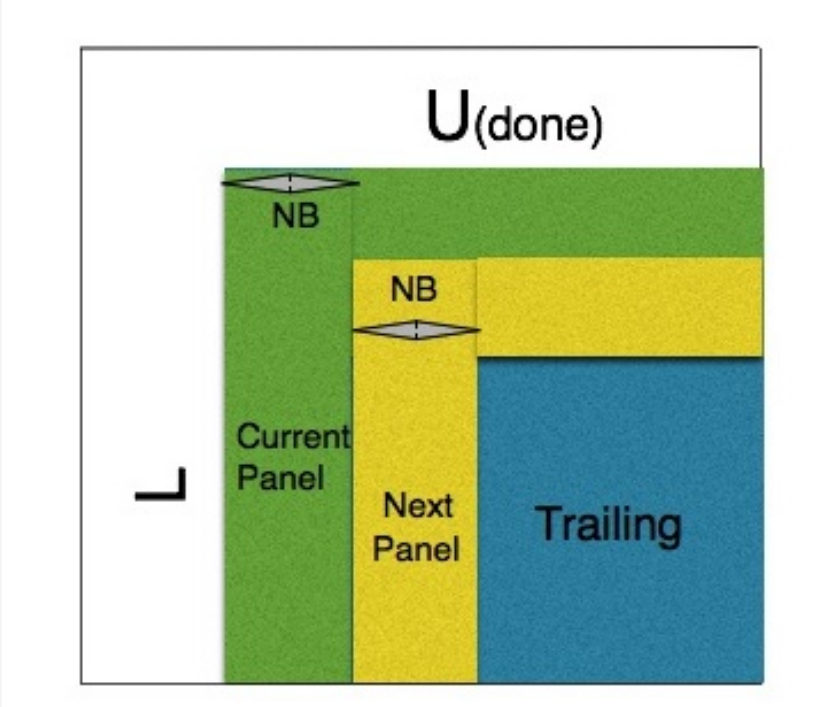

streamed dgemm K=128
batched dgemm K=128
streamed dgemm K= 64
batched dgemm K= 64
streamed dgemm K= 32
batched dgemm K= 32
Gflops/s
matrix m=n
0
100
200
300
400
500
600
700
800
900
0
32
64
128
160
192
256
384
448
512

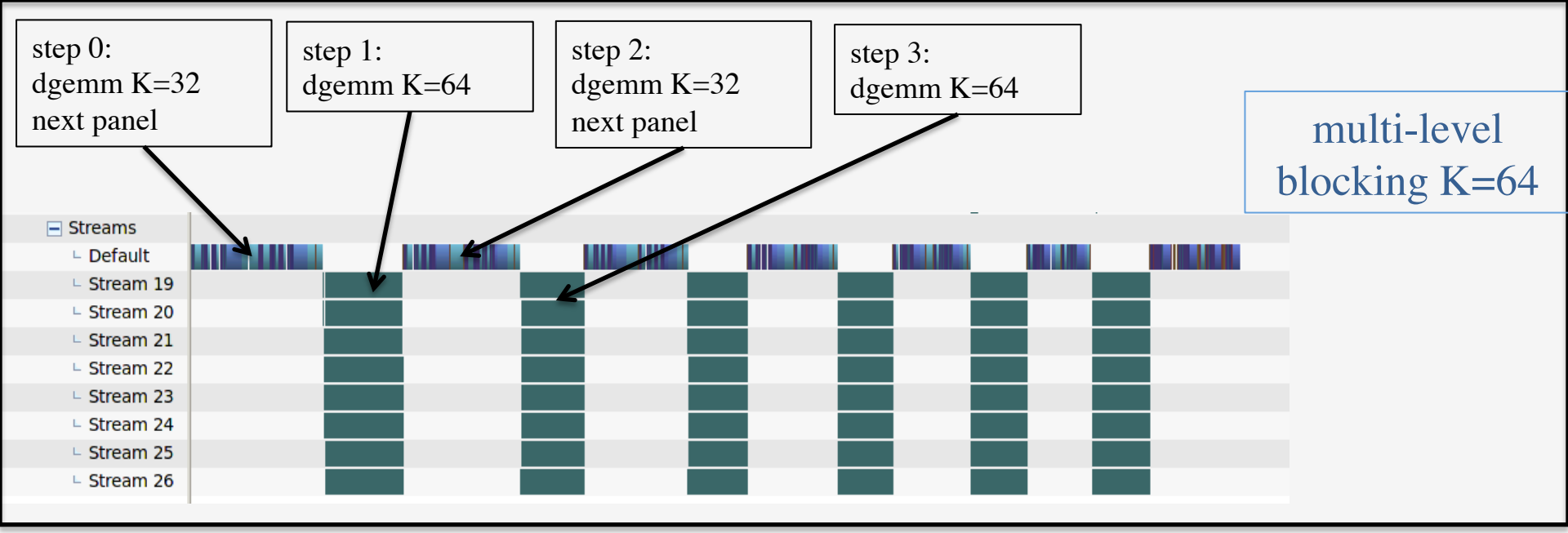
LU details

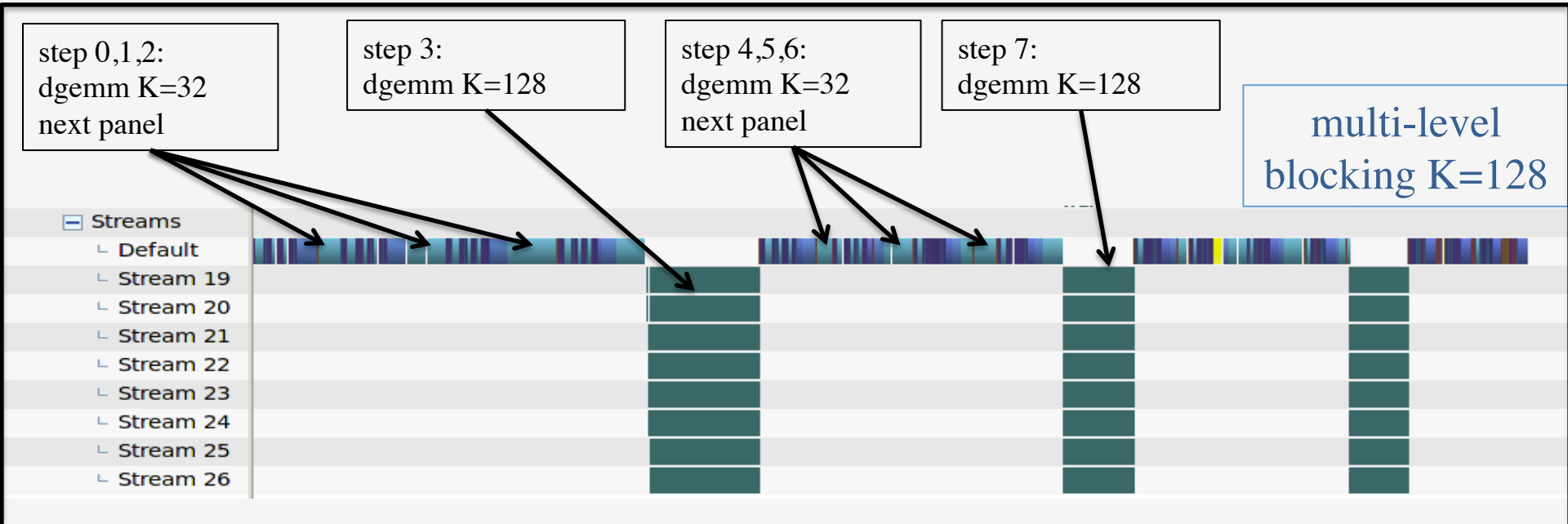


Proposition:

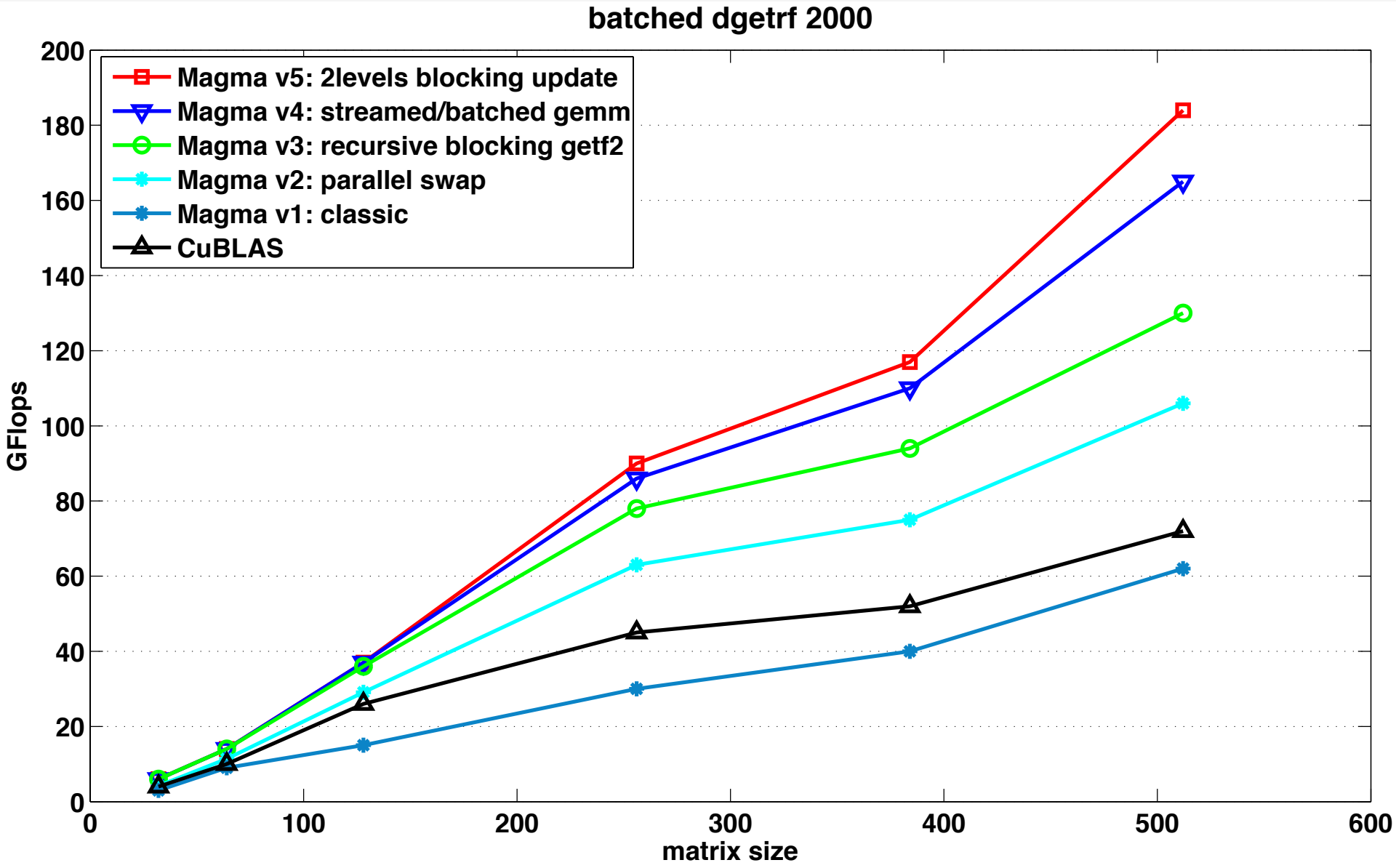
- **Multi-levels blocking of the update***:*
  Take advantage of the dgemm with larger K




step 0:
dgemm K=32
next panel
step 1:
dgemm K=64
step 2:
dgemm K=32
next panel
step 3:
dgemm K=64
multi-level
blocking K=64
Streams
Default
Stream 19
Stream 20
Stream 21
Stream 22
Stream 23
Stream 24
Stream 25
Stream 26


step 0,1,2:
dgemm K=32
next panel
step 3:
dgemm K=128
step 4,5,6:
dgemm K=32
next panel
step 7:
dgemm K=128
multi-level
blocking K=128
Streams
Default
Stream 19
Stream 20
Stream 21
Stream 22
Stream 23
Stream 24
Stream 25
Stream 26

Performance



Summary and future work

Batched GPU one sided factorization:

- Batched xpotrf is done, toward the batched xgeqrf
- Use the batched technique to factorize tall-skinny matrices for both dense and sparse
- Develop a version of the panel that factorize only 1 large panel:
  - to be integrated in MAGMA standalone-GPU one-sided factorization
  - also might be interesting to port it to the Nvidia Jetson (arms+gpu)

Ongoing stuff:

- Multi-GPU SVD code (1-stage and 2-stages) also using the dynamic version of MAGMA.
- Distributed Eigenvalue/SVD for PARSEC
- Distributed Eigenvalue on GPU using two-stages (paper submitted to SC14)
- Tasks based eigensolver routines for multicore/distributed/GPU/MIC