GPU TECHNOLOGY CONFERENCE

CUDA Accelerated Linpack on Clusters

E. Phillips and M. Fatica, NVIDIA Corporation | September 21 2010

PRESENTED BY NVIDIA.

Outline

- Linpack benchmark
- Tesla T10
  - DGEMM Performance / Strategy
  - Linpack Results
- Tesla T20
  - DGEMM Performance / Strategy
  - DTRSM
  - Linpack Results
- Conclusions

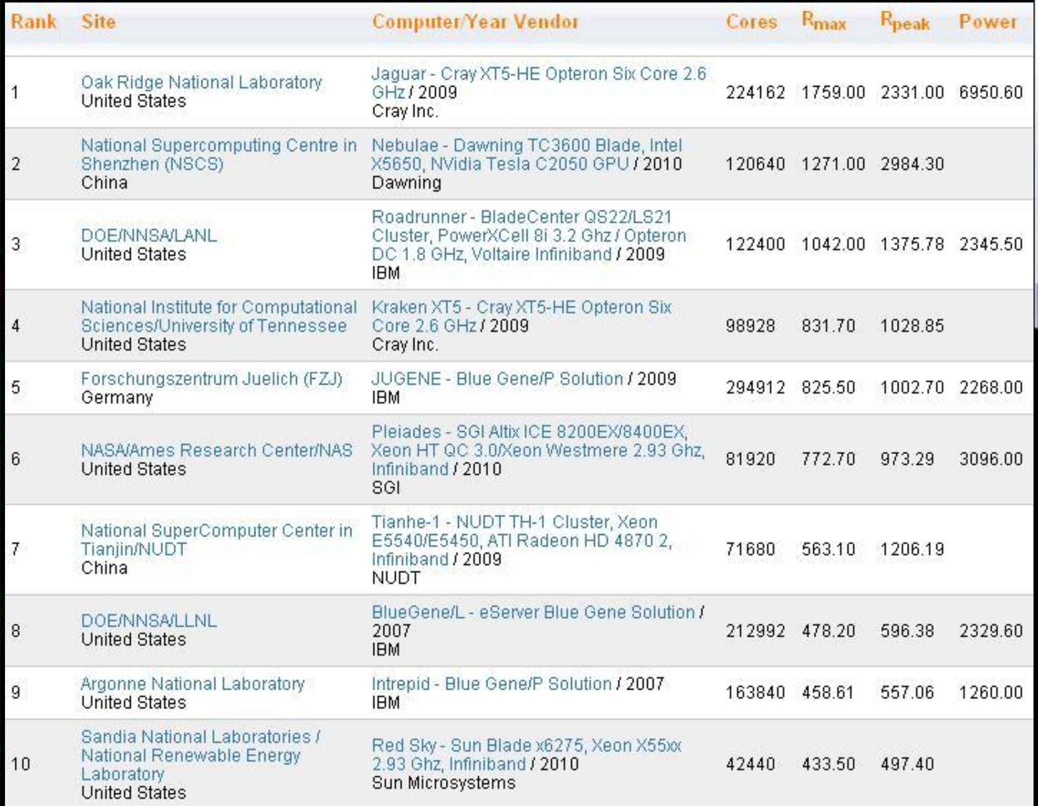
LINPACK Benchmark

The LINPACK benchmark is very popular in the HPC space, because it is used as a performance measure for ranking supercomputers in the TOP500 list.

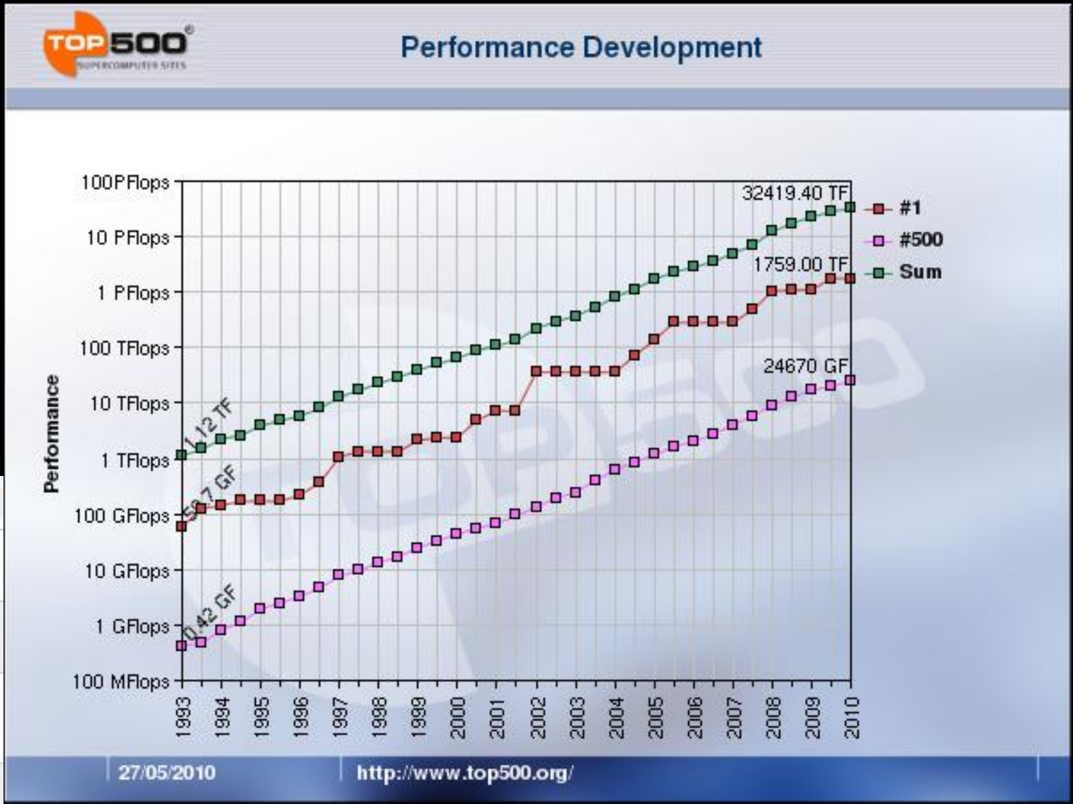
The most widely used implementation is the HPL software package from the Innovative Computing Laboratory at the University of Tennessee:

> it solves a random dense linear system in double precision arithmetic on distributed-memory computers.

Top500 list



Rank	Site	Computer/Year Vendor	Cores	R_{max}	R_{peak}	Power
1	Oak Ridge National Laboratory United States	Jaguar - Cray XT5-HE Opteron Six Core 2.6 GHz / 2009 Cray Inc.	224162	1759.00	2331.00	6950.60
2	National Supercomputing Centre in Shenzhen (NSCS) China	Nebulae - Dawning TC3600 Blade, Intel X5650, NVidia Tesla C2050 GPU / 2010 Dawning	120640	1271.00	2984.30	
3	DOE/NNSA/LANL United States	Roadrunner - BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 GHz, Voltaire Infiniband / 2009 IBM	122400	1042.00	1375.78	2345.50
4	National Institute for Computational Sciences/University of Tennessee United States	Kraken XT5 - Cray XT5-HE Opteron Six Core 2.6 GHz / 2009 Cray Inc.	98928	831.70	1028.85	
5	Forschungszentrum Juelich (FZJ) Germany	JUGENE - Blue Gene/P Solution / 2009 IBM	294912	825.50	1002.70	2268.00
6	NASA/Ames Research Center/NAS United States	Pleiades - SGI Altix ICE 8200EX/8400EX, Xeon HT QC 3.0/Xeon Westmere 2.93 Ghz, Infiniband / 2010 SGI	81920	772.70	973.29	3096.00
7	National SuperComputer Center in Tianjin/NUDT China	Tianhe-1 - NUDT TH-1 Cluster, Xeon E5540/E5450, ATI Radeon HD 4870 2, Infiniband / 2009 NUDT	71680	563.10	1206.19	
8	DOE/NNSA/LLNL United States	BlueGene/L - eServer Blue Gene Solution / 2007 IBM	212992	478.20	596.38	2329.60
9	Argonne National Laboratory United States	Intrepid - Blue Gene/P Solution / 2007 IBM	163840	458.61	557.06	1260.00
10	Sandia National Laboratories / National Renewable Energy Laboratory United States	Red Sky - Sun Blade x6275, Xeon X55xx 2.93 Ghz, Infiniband / 2010 Sun Microsystems	42440	433.50	497.40	



LINPACK Benchmark

Solve a dense NxN linear system:

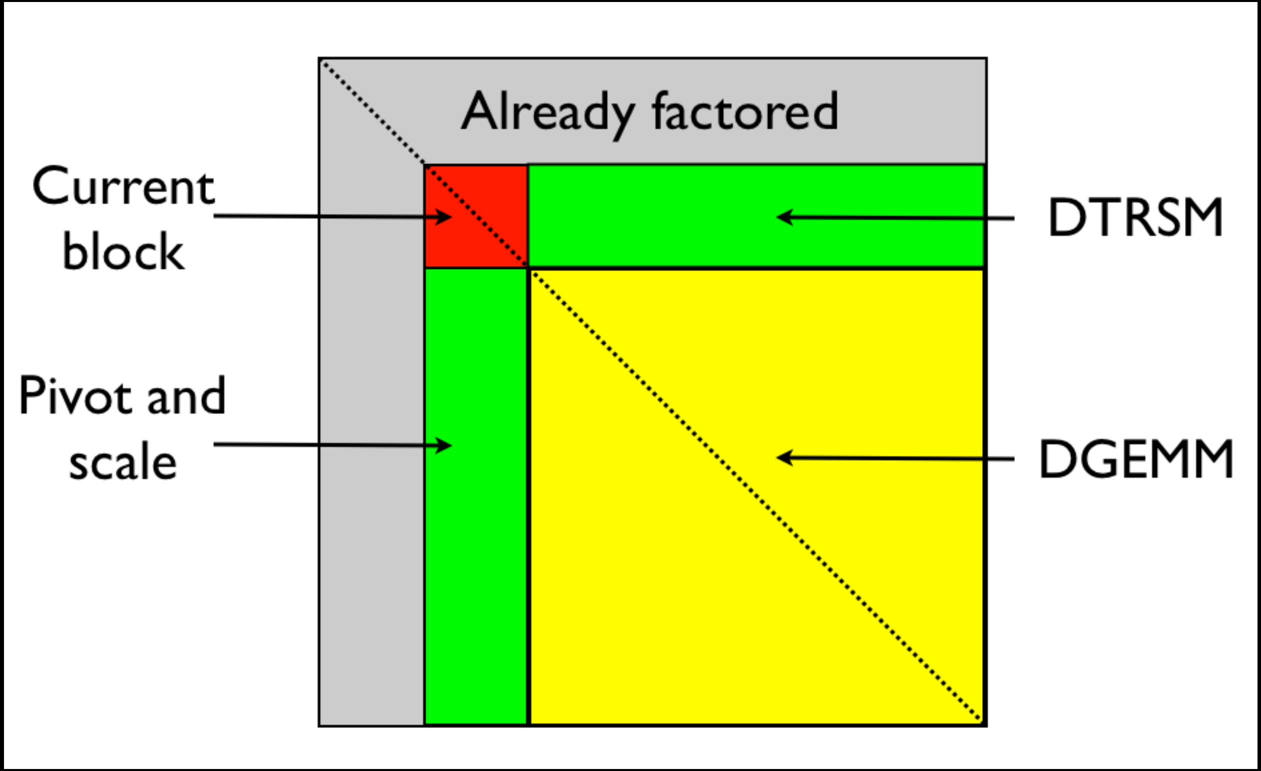
$$Ax=b$$

Solution is obtained by Gaussian elimination with partial pivoting

Floating point workload:

2/3 N^3 + 2 N^2

(LU decomposition) (back solve)



Factorize the current block (red), update the green and yellow parts when done

The bigger the problem size N is, the more time is spent in the update (DGEMM)

CUDA Accelerated LINPACK

Both CPU cores and GPUs are used in synergy with minor or no modifications to the original source code (HPL 2.0):

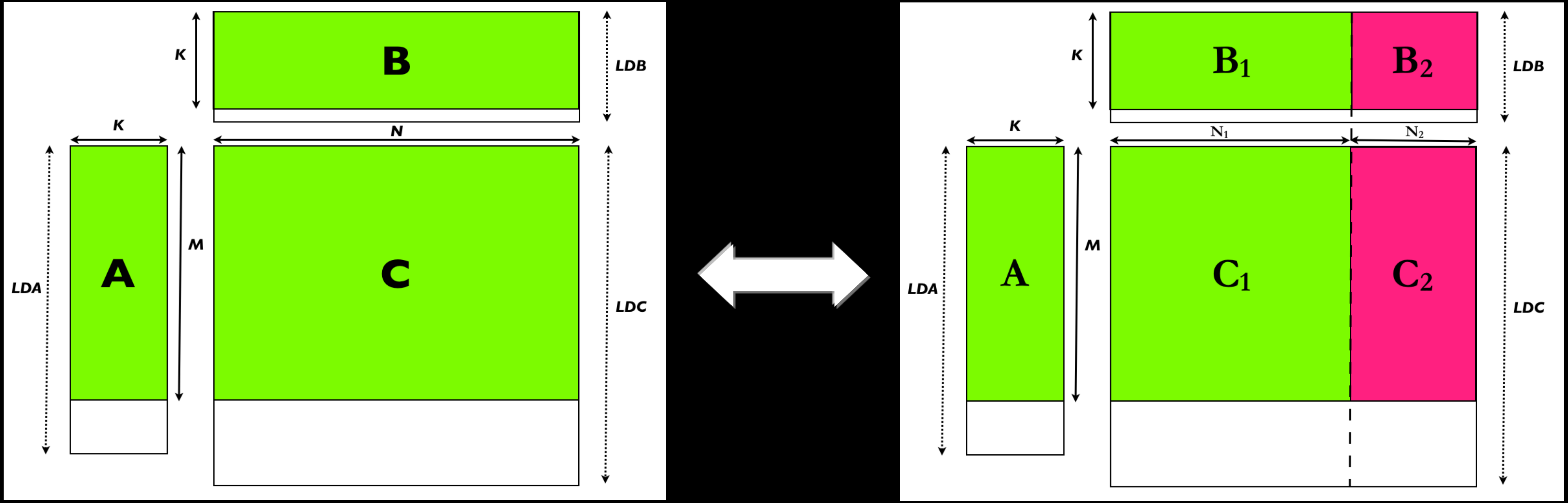
- An host library intercepts the calls to DGEMM and DTRSM and executes them simultaneously on the GPUs and CPU cores.

- Use of pinned memory for fast PCI-e transfers, up to 5.7GB/s on x16 gen2 slots. Only changes to the HPL source

NVIDIA Tesla GPU Computing Products

	Server Module		1U Systems		Workstation Boards	
	Tesla M2070 / Tesla M2050	Tesla M1060	Tesla S2050	Tesla S1070	Tesla C2070 / Tesla C2050	Tesla C1060
GPUs	1 T20 GPU	1 T10 GPU	4 T20 GPUs	4 T10 GPUs	1 T20 GPU	1 T10 GPU
Single Precision	1030 GFlops	933 GFlops	4120 GFlops	4140 GFlops	1030 Gflops	933 GFlops
Double Precision	515 Gflops	78 GFlops	2060 GFlops	346 GFlops	515 Gflops	78 GFlops
Memory	6 GB / 3 GB	4 GB	12 GB (S2050)	16 GB 4 GB / GPU	6 GB / 3 GB	4 GB
Mem BW	148.4 GB/s	102 GB/s	148.4 GB/s	102 GB/s	144 GB/s	102 GB/s

DGEMM: C = alpha A B + beta C



DGEMM(A,B,C) = DGEMM(A,B1,C1) U DGEMM(A,B2,C2)

The idea can be extended to multi-GPU configuration and to handle huge matrices

Find the optimal split, knowing the relative performances of the GPU and CPU cores on DGEMM

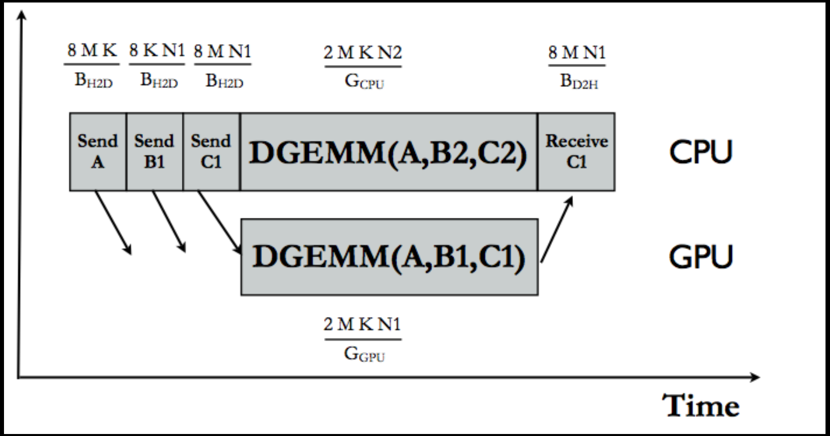
Overlap DGEMM on CPU and GPU

```
// Copy A
cublasSetMatrix (m, k    , sizeof(A[0]), A, lda, devA, m_gpu);
// Copy B1
cublasSetMatrix (k ,n_gpu, sizeof(B[0]), B, ldb, devB, k_gpu);
// Copy C1
cublasSetMatrix (m, n_gpu, sizeof(C[0]), C, ldc, devC, m_gpu);

// DGEMM on GPU
// Control returns immediately to CPU
 cublasDgemm('n', 'n', m, n_gpu, k, alpha, devA, m,devB, k, beta, devC, m);

// DGEMM on CPU
 dgemm('n','n',m,n_cpu,k, alpha, A, lda,B+ldb*n_gpu, ldb, beta,C+ldc*n_gpu, ldc);

// Copy C1
 status = cublasGetMatrix (m, n, sizeof(C[0]), devC, m, C, *ldc);
```



Using CUDA, it is very easy to express the workflow in the diagram

Optimal split

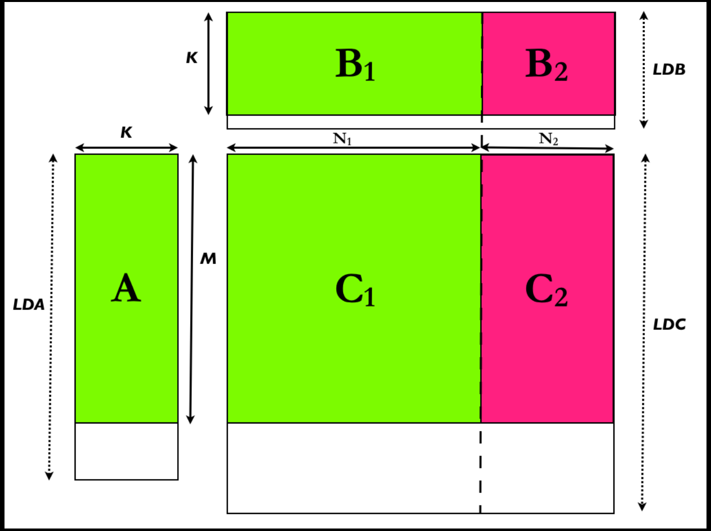
If A(M,K), B(K,N) and C(M,N), a DGEMM call performs 2*M*K*N operations

$$T_{CPU}(M,K,N2) = T_{GPU}(M,k,N1) \qquad N=N1+N2$$

If G_{CPU} denotes the DGEMM performance of the CPU in Gflops and G_{GPU} the one of the GPU,

The optimal split is

$$\eta = G_{GPU} / (G_{CPU}+G_{GPU})$$



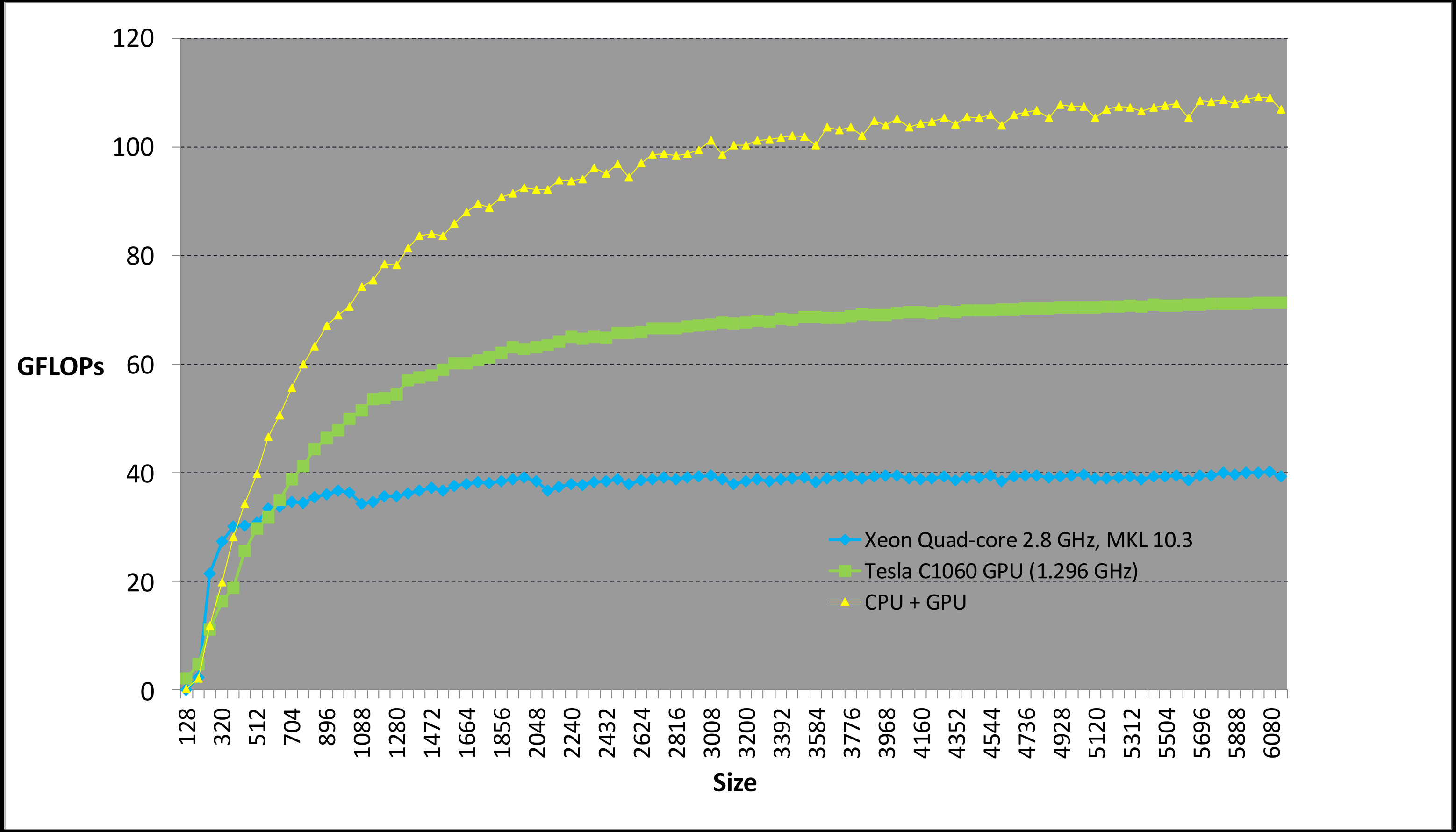
DGEMM performance on GPU (T10)

A DGEMM call in CUBLAS maps to several different kernels depending on the size
With the combined CPU/GPU approach, we can always send optimal work to the GPU.

M	K	N	M%64	K%16	N%16	Gflops
448	400	12320	Y	Y	Y	82.4
12320	400	1600	N	Y	Y	75.2
12320	300	448	N	N	Y	55.9
12320	300	300	N	N	N	55.9

Tesla T10 1.44Ghz, data resident in GPU memory. Optimal kernel achieves 95% of peak

DGEMM Performance



Results on workstation

SUN Ultra 24 workstation with an Intel Core2 Extreme Q6850 (3.0Ghz) CPU, 8GB of memory plus a Tesla C1060 (1.296Ghz) card.

-Peak DP CPU performance of 48 GFlops
-Peak DP GPU performance of 77 Gflops (60*clock)

```
T/V                N      NB     P     Q           Time              Gflops
----------------------------------------------------------------------------
WR00L2L2       23040   960     1     1          97.91            8.328e+01
----------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0048141 ...... PASSED
```

83.2 Gflops sustained on a problem size using less than 4GB of memory: 66% of efficiency

```
T/V                N      NB     P     Q           Time              Gflops
--------------------------------------------------------------------
WR00C2R2       32320  1152     1     1         246.86            9.118e+01
--------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0042150 ...... PASSED
```

91.1 Gflops sustained on a problem size using 8GB of memory: 73% of efficiency

PCI-e transfer speed

	SUN Ultra 24 PCI-e x16 gen2		Supermicro 6016GT PCI-e x16 gen2	
	Pageable Memory	Pinned Memory	Pageable Memory	Pinned Memory
H2D	2132 MB/s	5212 MB/s	4665 MB/s	5745 MB/s
D2H	1882 MB/s	5471 MB/s	4064 MB/s	6059 MB/s

CUDA offers a fast PCI-e transfer when host memory is allocated with cudaMallocHost instead of regular malloc.

Effect of PCI-e bandwidth

Page locked memory

```
T/V                  N      NB     P     Q           Time              Gflops
-------------------------------------------------------------------------------
WR00L2L2        23040   960     1     1            97.91           8.328e+01
-------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=       0.0048141 ...... PASSED
```

Pageable memory

```
T/V                  N      NB     P     Q           Time              Gflops
-------------------------------------------------------------------------------
WR00L2L2        23040   960     1     1           117.06           6.966e+01
-------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=       0.0048141 ...... PASSED
```

Results on cluster

Cluster with 8 nodes, each node connected to half of a Tesla S1070-500 system:

-Each node has 2 Intel Xeon E5462 (2.8Ghz with 1600Mhz FSB) , 16GB of memory and 2 GPUs (1.44Ghz clock).

-The nodes are connected with SDR Infiniband.

```
T/V                 N       NB     P     Q              Time                Gflops
--------------------------------------------------------------------------------
WR11R2L2       118144    960     4     4            874.26             1.258e+03
--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0031157 ...... PASSED
```

1.258 Tflop/s sustained using 8GB of memory per MPI process.
The first system to break the Teraflop barrier was ASCI Red (1.068 Tflop/s) in June 1997 with 7264 Pentium Pro processors. The GPU accelerated nodes occupies 8U (11U including Ethernet and Infiniband switches).

GPU TECHNOLOGY CONFERENCE

FERMI T20 GPU

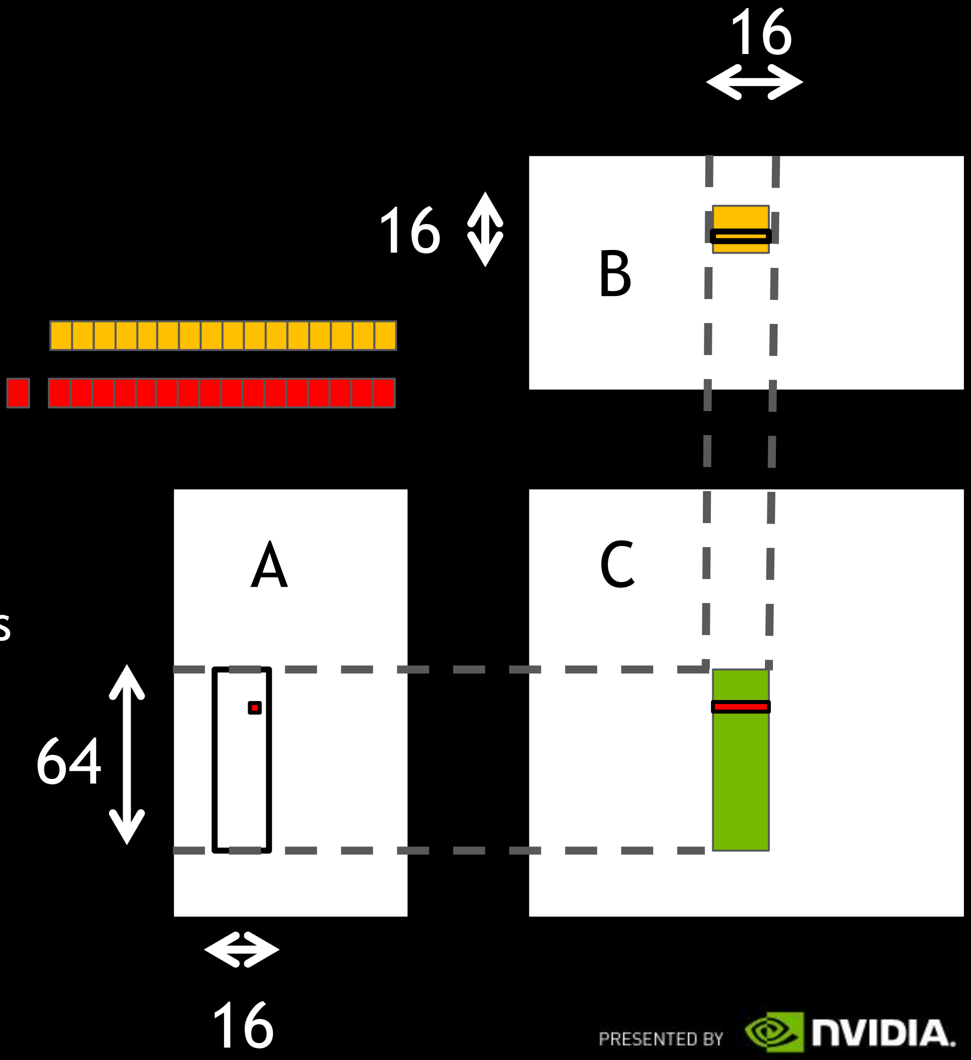
PRESENTED BY NVIDIA.

T20 (Fermi) Architecture

- Increased DP Throughput
  - 515 GFLOPS *PEAK* vs 85 GFLOPS T10
  - Theoretical 75% DGEMM : 515 x .75 = **386 GFLOPS**
- Cache Heirarchy
  - 64KB configurable L1 + shared memory (48/16 or 16/48)
  - 768 KB unified L2
  - Load/Store architecture
- Dual Copy Engines
  - Overlap both download and upload of data with compute
  - Support Asynchronous 2D data transfers

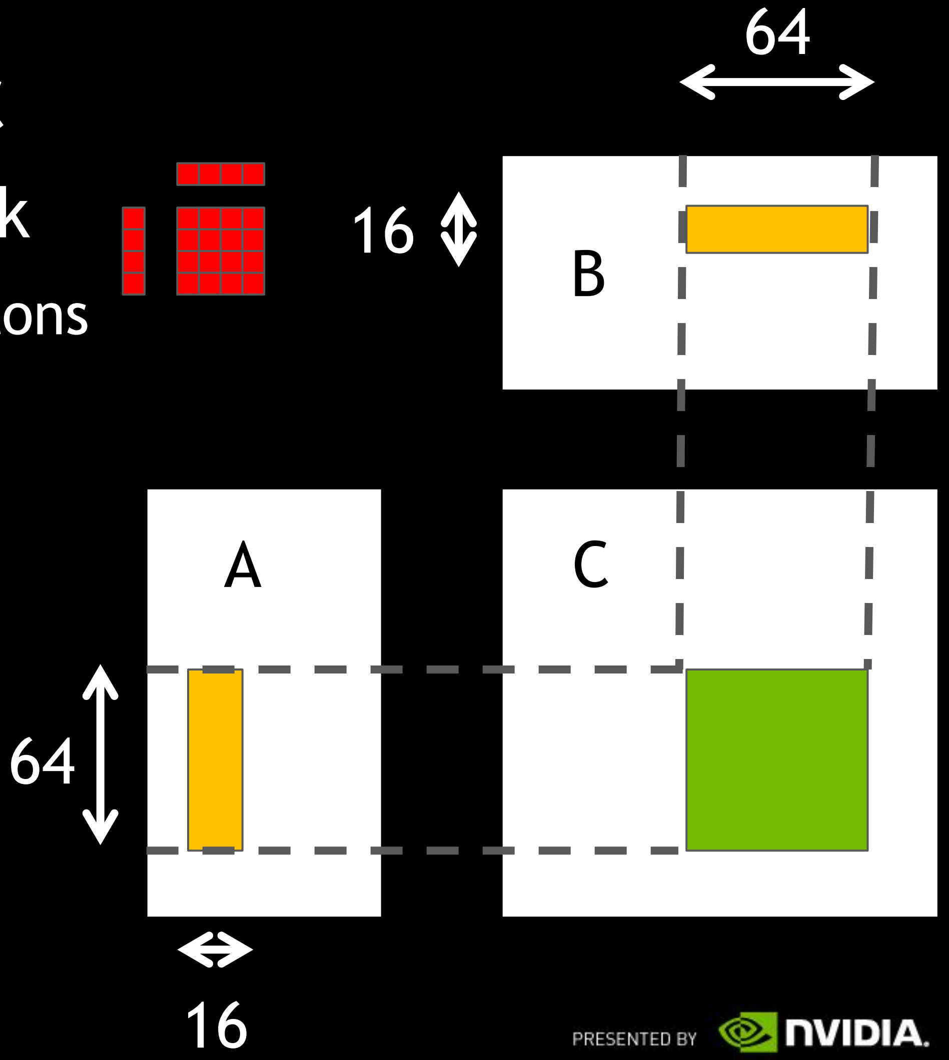
T10 DGEMM

- T10 DGEMM algorithm (Volkov)
  - 64 Threads 64x16 of C
  - B reuse in shared memory
  - A loaded from global memory
  - DP limited on T10
    - DP 8x slower than other instructions
- T20 DP full speed
  - T10 DGEMM runs 175 GFLOPS
  - Different Bottlnecks



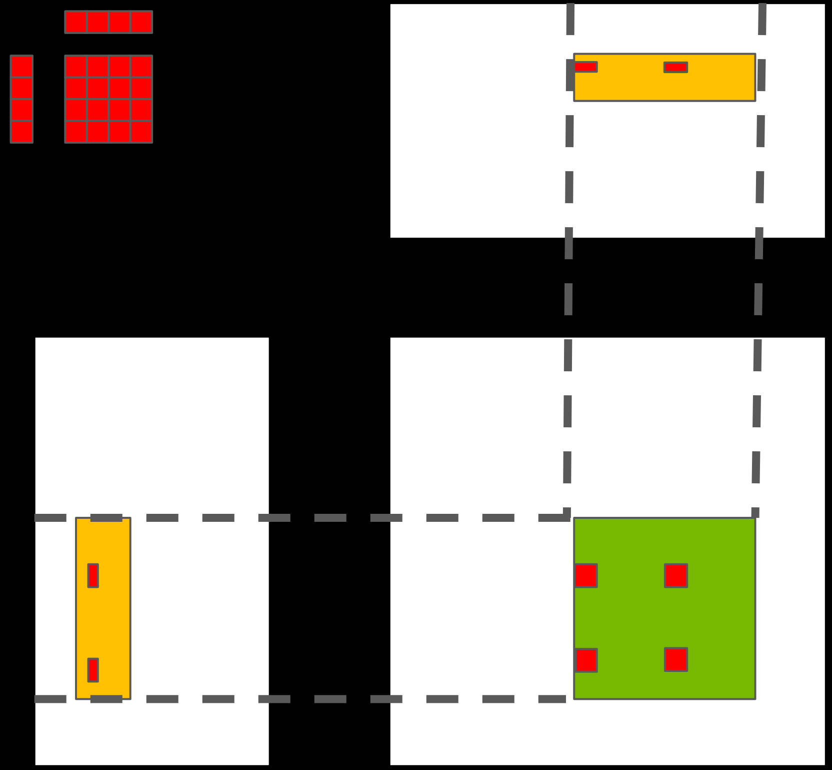
T20 DGEMM

- 16x16 Threads update 64x64 of C
- Instruction throughput bottleneck
  - Maximize Density of DFMA instrucitons
  - Register Blocking
  - Use wide vector Loads (128 bits)
  - DFMA dual issues with Texture
    - Texture fetch A and B
- Hide Latencies
  - Double Buffer Shared Memory



T20 DGEMM Performance

- Inner Loop
  - Multiply 4x4 elements per thread
  - 4 Loads (128-bit), 16 DFMA = 80%
- Outer Loop (16 iterations)
  - 256 DFMA / 341 instruction = 75 %
- CUDA code ~ 301 GFLOPS
  - General – all configs – all sizes
- Linpack version ~360 GFLOPS
  - NT config, M%64, N%64, K%16
  - 360/515 = 70%   360/386 = 93%



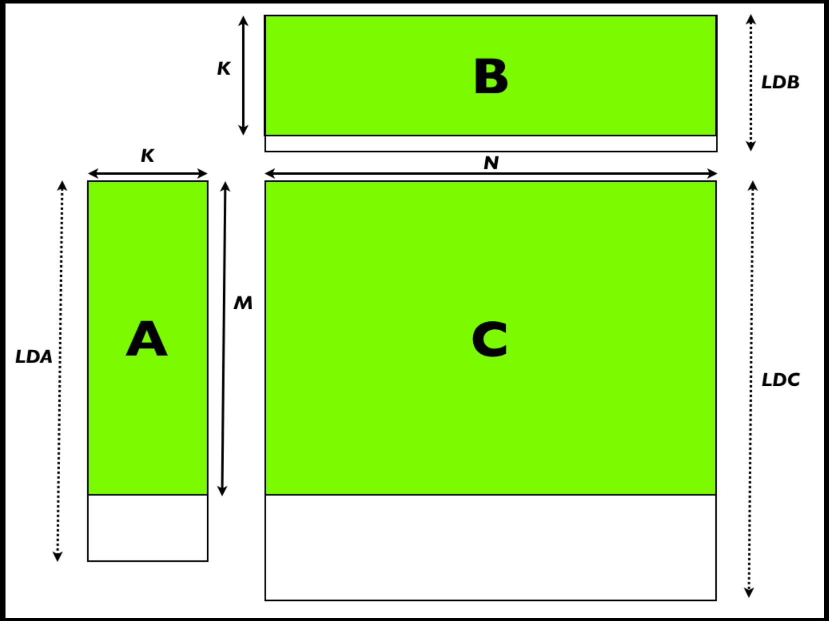
T20 DGEMM Efficiency

- Why T20 DGEMM 70% PEAK < T10 95% PEAK ?
  - Faster compute exposes other bottlenecks

- Compare with T10 SGEMM
  - T10 622 GFLOPS peak MAD
  - 2x Effective resources: bandwidth (102 GB/s *2 = 204 GB/s)
  - SGEMM performance: 400 GFLOPS (Hand Code)
  - 400 / 622 = 64%

DGEMM Compute/COPY Analysis

- Assume N,M>>K
- Copy Time = $\frac{8*(mk+nk+2mn)}{PCIe} \sim \frac{16mn}{PCIe}$
- Compute Time = $\frac{2mnk}{GFLOPS}$
- % Compute = $\frac{1}{1+\frac{8*GFLOPS}{K*PCIe}} = \frac{1}{1+x}, \mathrm{x} = \frac{8*\mathrm{GFLOPS}}{\mathrm{K}*\mathrm{PCIe}}$



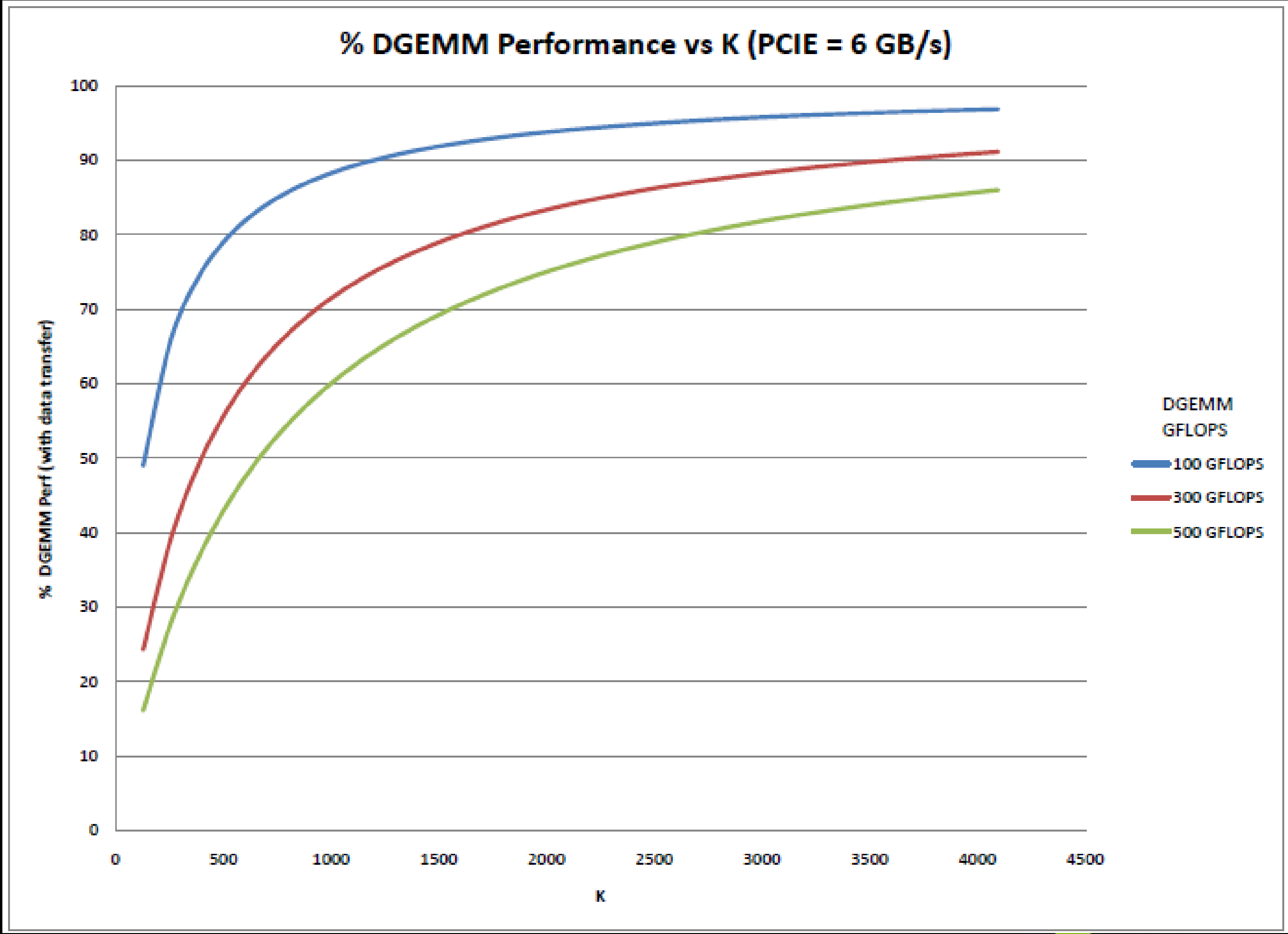
DGEMM Analysis

- K~1000,PCIe=6GB/s

  100 GFLOPS= ~88%

  300 GFLOPS= ~70%

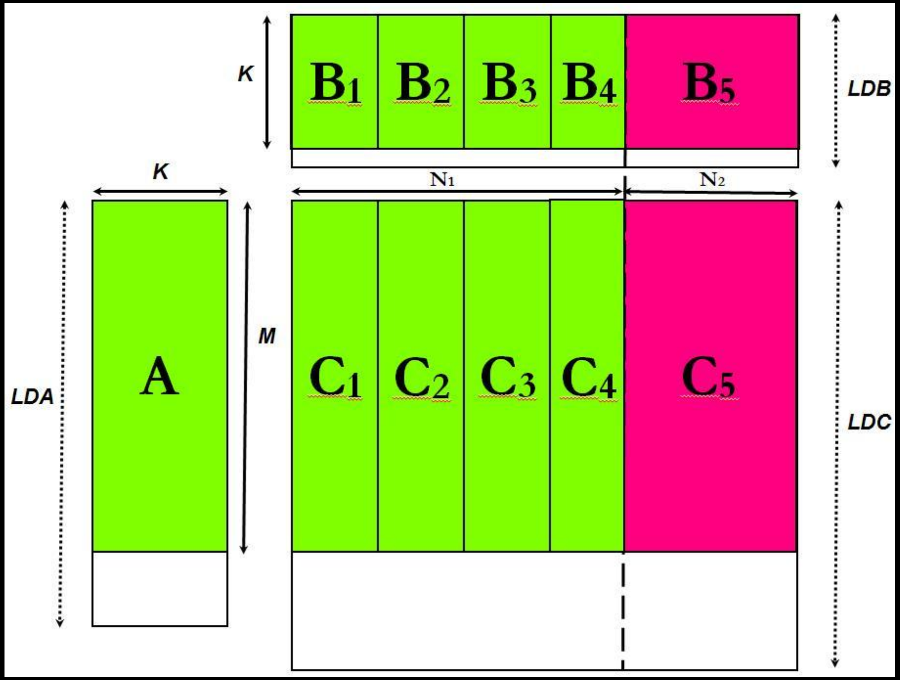
  500 GFLOPS= ~60%
- Larger K increases compute intensity
- Overlapping Copy Benefit increases with GPU Perf



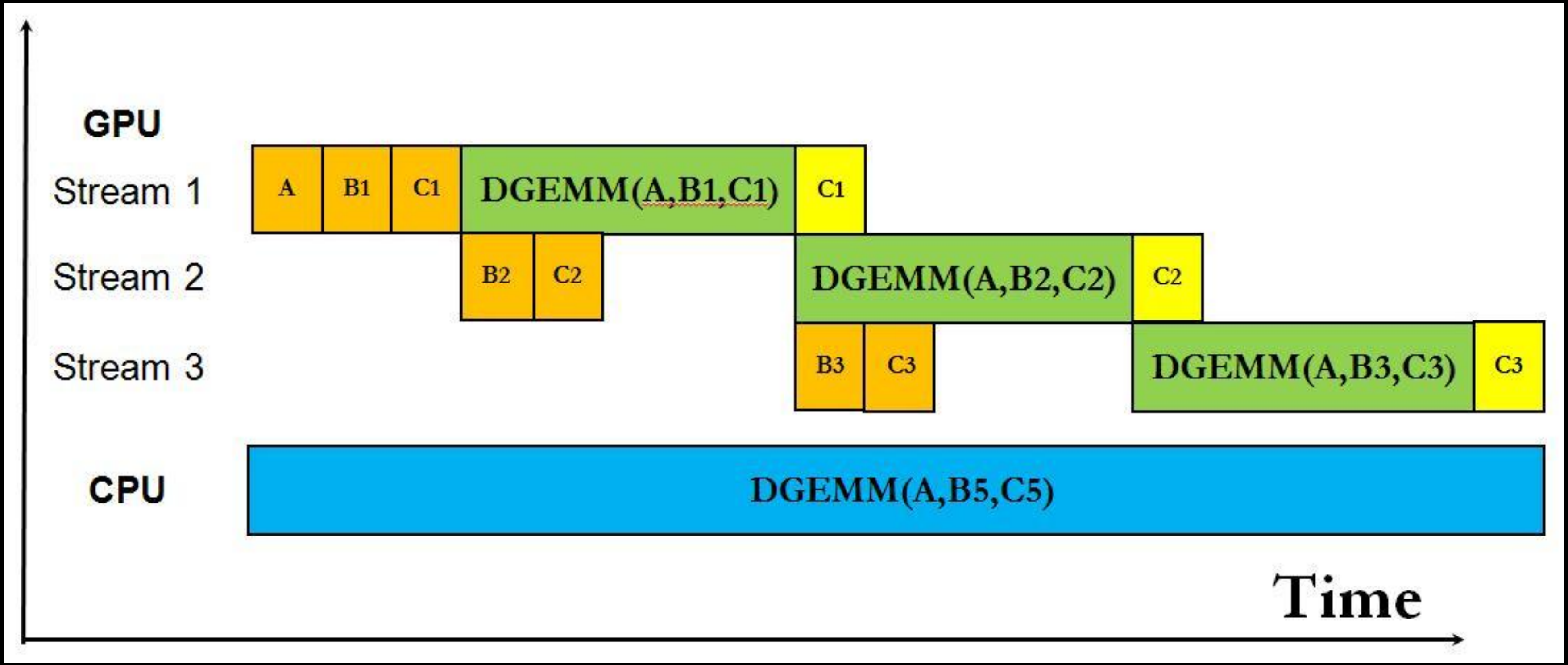
Fermi DGEMM Strategy

- Slice Matrix into several pieces
- Use Stream API
  - Overlap COPY + Compute

```
Copy A H2D
Loop over pieces:
    Copy Bi,Ci H2D
    DGEMM A,Bi,Ci
    Copy Ci D2H
CPU DGEMM A,B_last,C_last
```



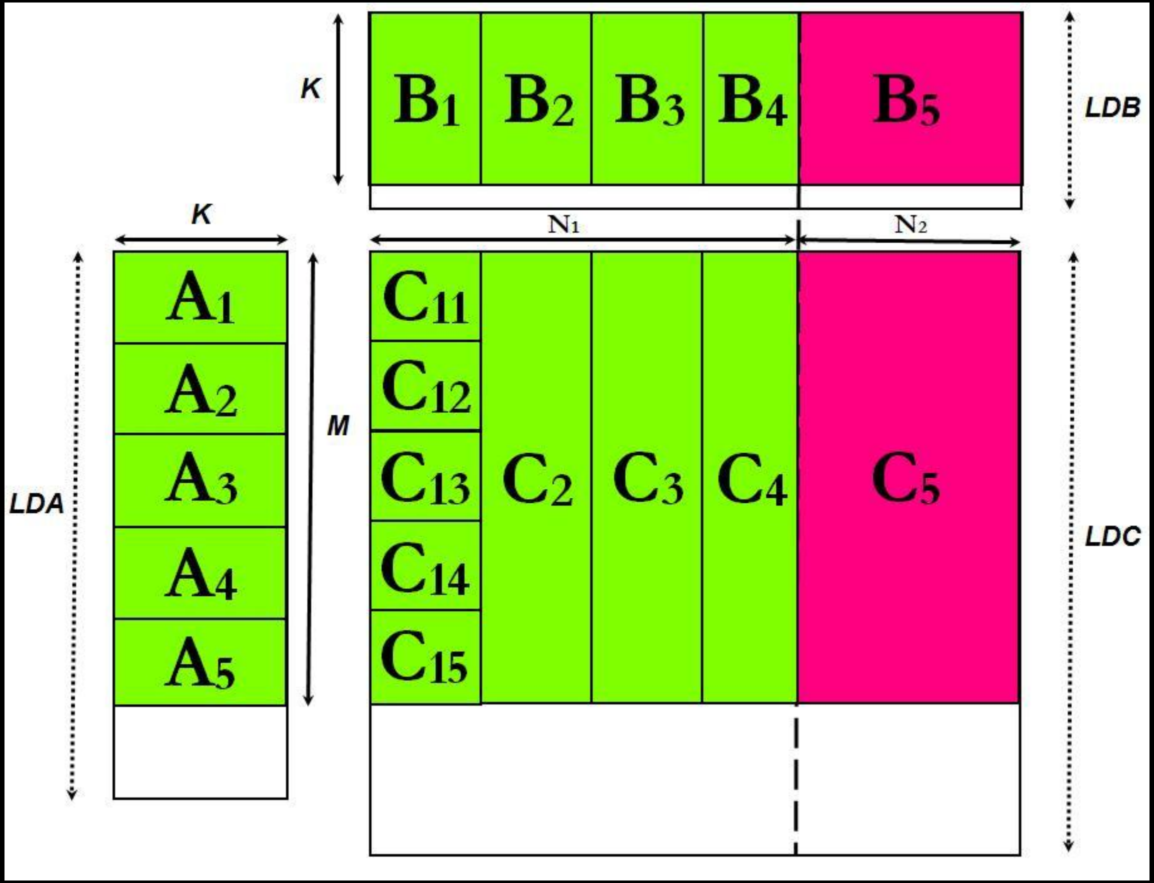
DGEMM Compute/Copy Overlap



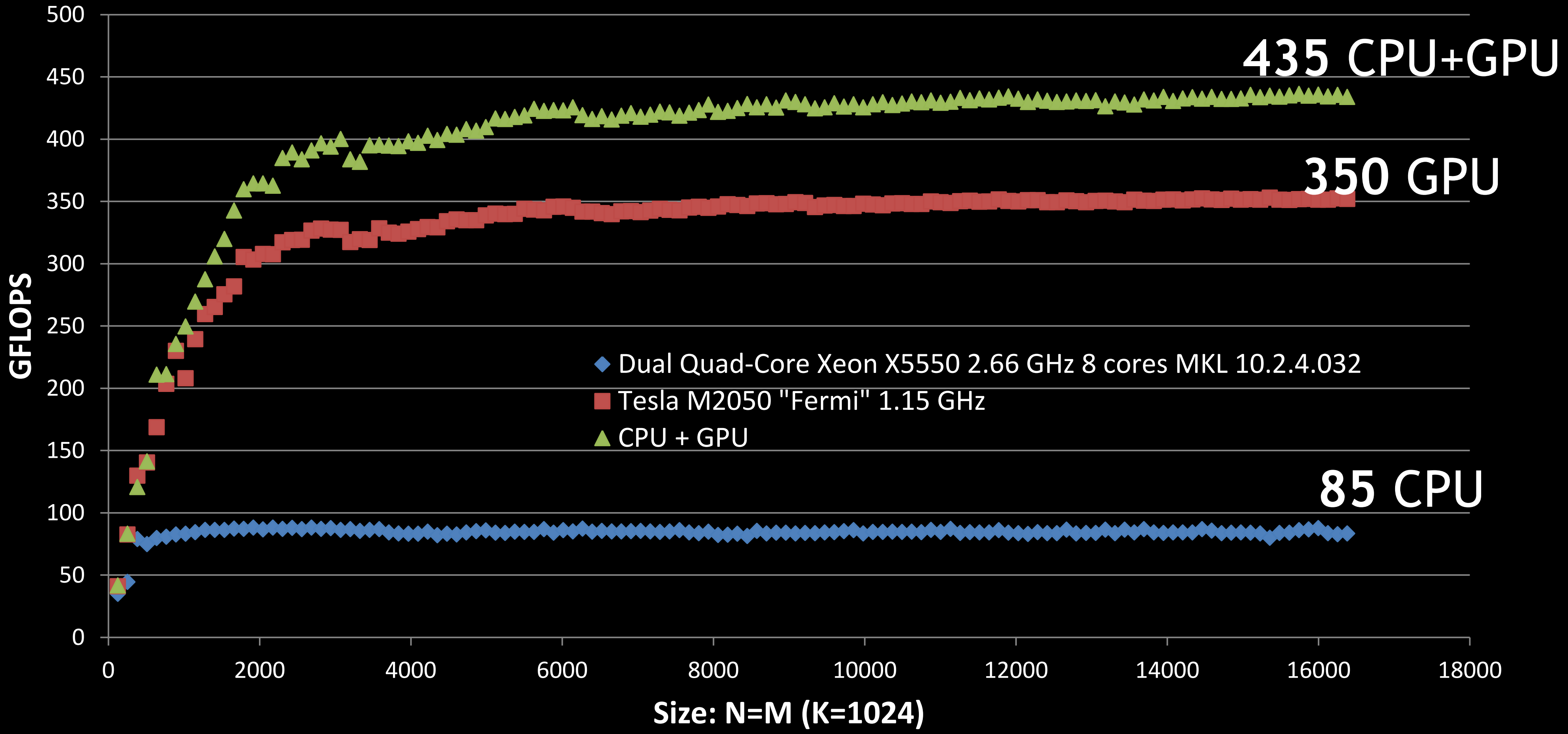
Additional Overlap strategy

- Copy A matrix can be significant for smaller matrix sizes

- Split A for additional Overlap



Fermi DGEMM Performance



Optimizations - Auto Split

- Keep track of CPU and GPU performance and adjust split
  - Wallclock() CPU time
  - Cuda event record GPU time
  - Compute optimal split for next iteration

```
cudaEventRecord(GPU_start,0);
Loop: launch GPU copys + Kernels
cudaEventRecord(GPU_stop,0);
CPU_start = wallclock();
Call CPU_DGEMM
CPU_stop = wallclock();
cudaEventSynchronize(GPU_stop);
```

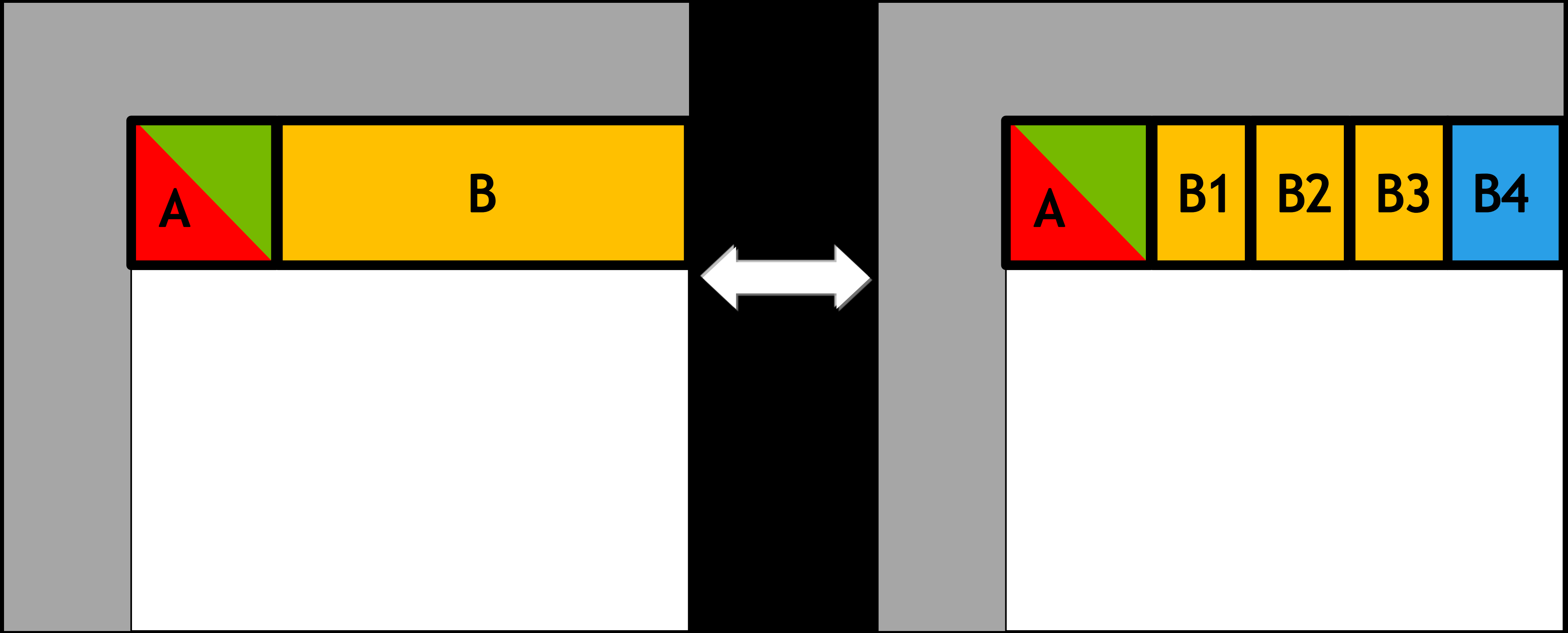
```
GPU_GFLOPS = GPU_FLOPS/GPU_TIME
CPU_GFLOPS = CPU_FLOPS/CPU_TIME
SPLIT = GPU_GFLOPS/(GPU_GFLOPS+CPU_GFLOPS)
```

DTRSM

- Triangular Solve with Multiple RHS
  - AX=B
  - B overwritten with solution X
- Same CPU/GPU work Split strategy as DGEMM
- GPU DTRSM Blocked Algorithm
  - Customized DTRSM Kernel on diagonal 64x64 blocks of A
  - Propagate to next row block of B with DGEMM
  - Take advantage of Hand Code DGEMM Kernel performance

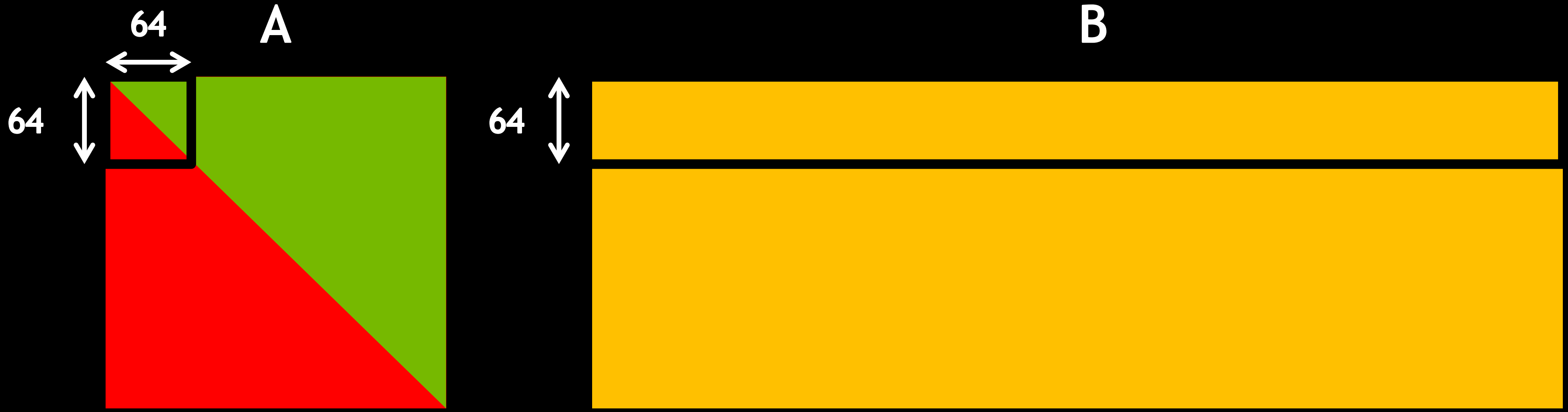
DTRSM

Same basic splitting approach as DGEMM



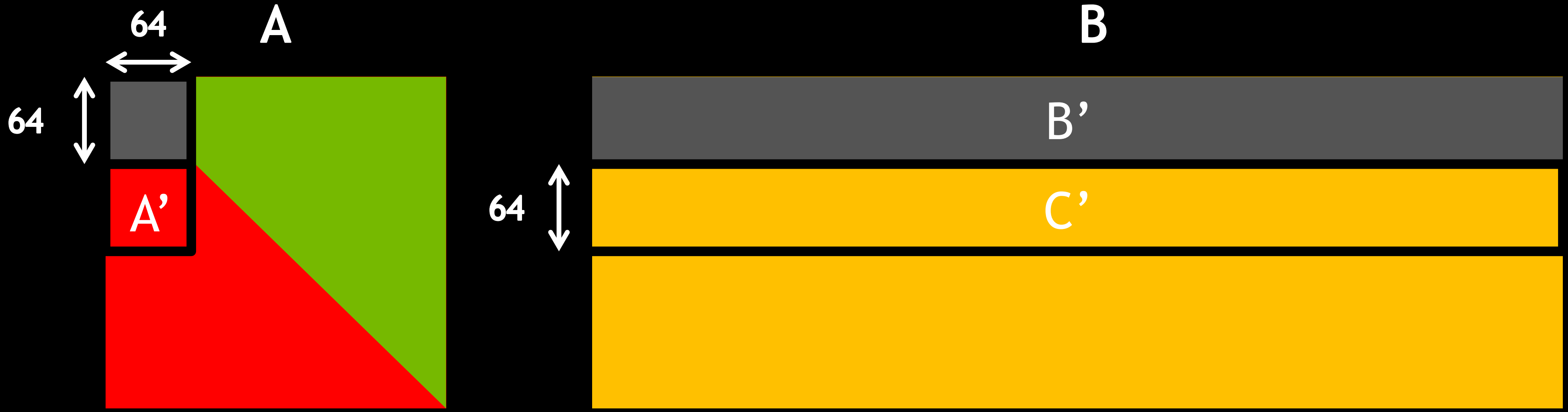
DTRSM

- GPU implimentation based on blocked DGEMM update
- First compute diagonal block of A / Row block of B



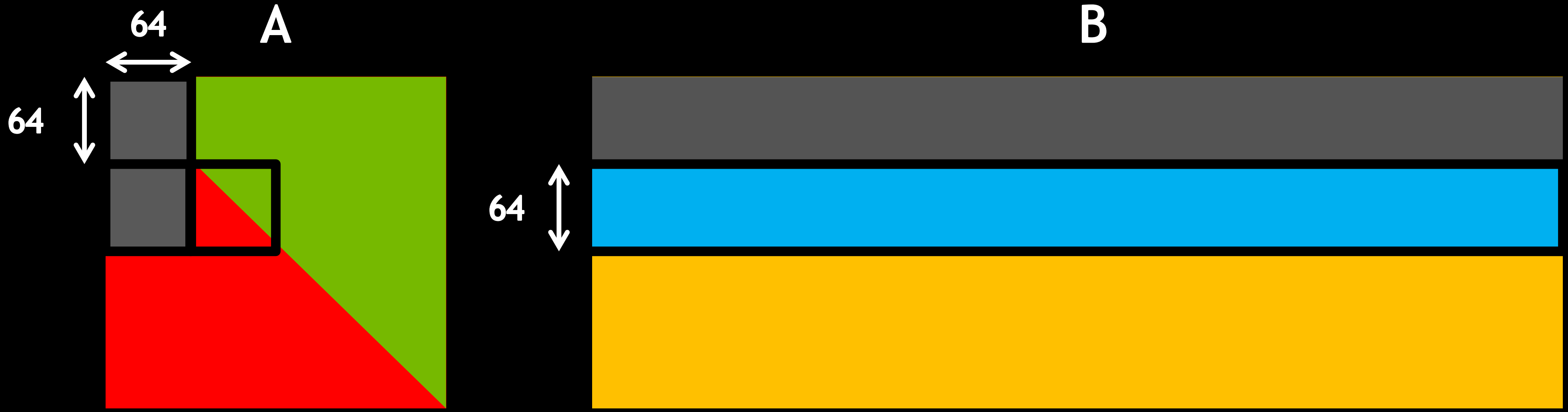
DTRSM

- GPU implimentation based on blocked DGEMM update
- First compute diagonal block of A / Row block of B
- DGEMM update next row B with non-diagonal row of A



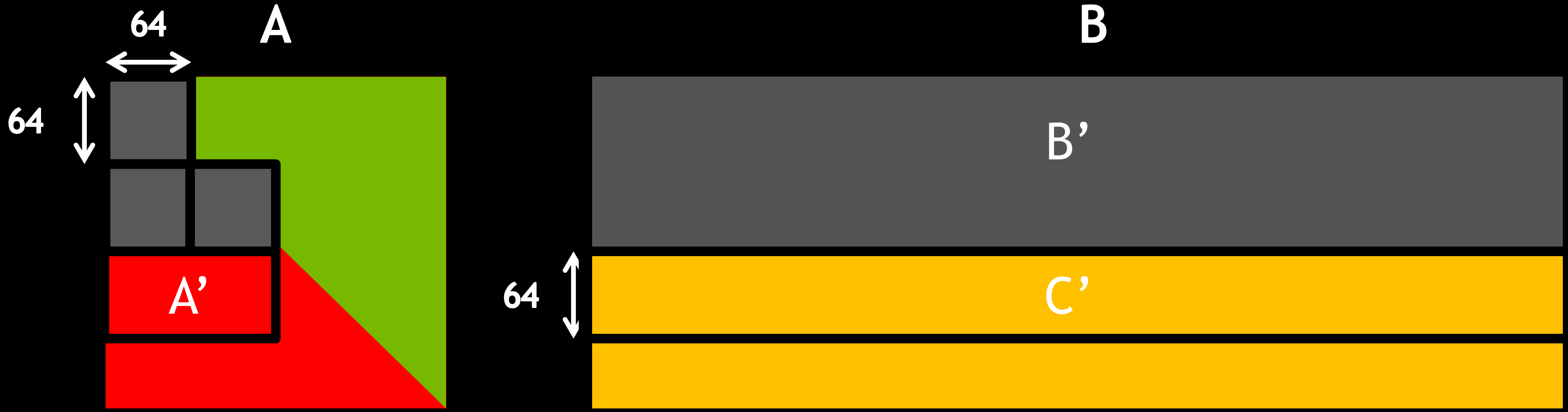
DTRSM

- Repeat:
- DTRSM diagonal block



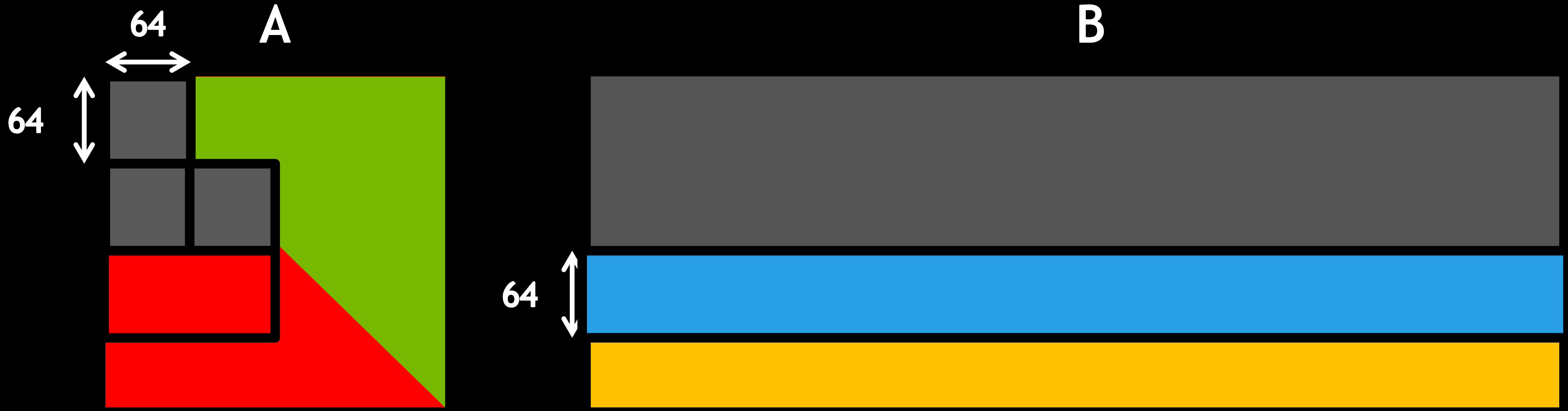
DTRSM

- Repeat:
- DTRSM diagonal block, then DGEMM update next row block



DTRSM

- Continue until all rows have been processed



GPU TECHNOLOGY CONFERENCE

Results on single node

Supermicro 6016GT-TF: Dual Intel Xeon X5560 Nehalem 2.8 GHz
96GB memory, 1 or 2 Tesla M2050 1.15Ghz cards

-Peak DP 89 + 515 = 604 GFLOPS
-DGEMM 89 + 350 = 439 GFLOPS

```
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR10L2L2      108032   768     1     1            2011.42              4.179e+02
--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0039415 ...... PASSED
================================================================================
```

417.9 GFLOPS = 69% of *PEAK* or 95% of *DGEMM*

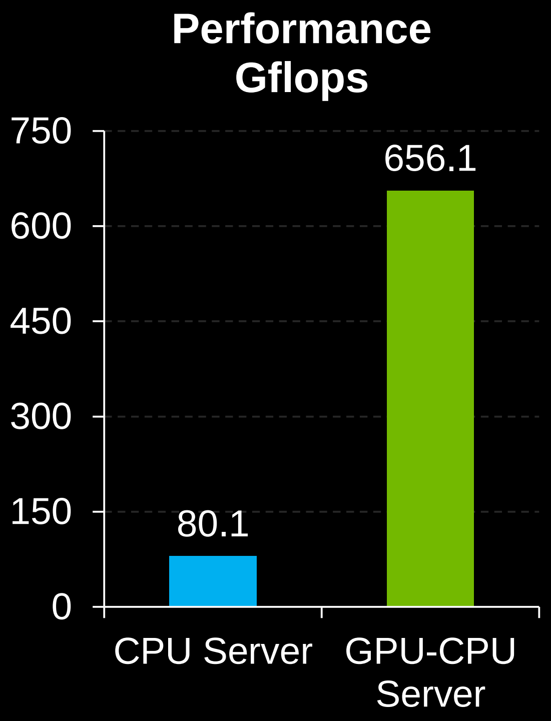
```
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR10L2L2      108032   768     1     2            1192.13              7.051e+02
--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0040532 ...... PASSED
================================================================================
```

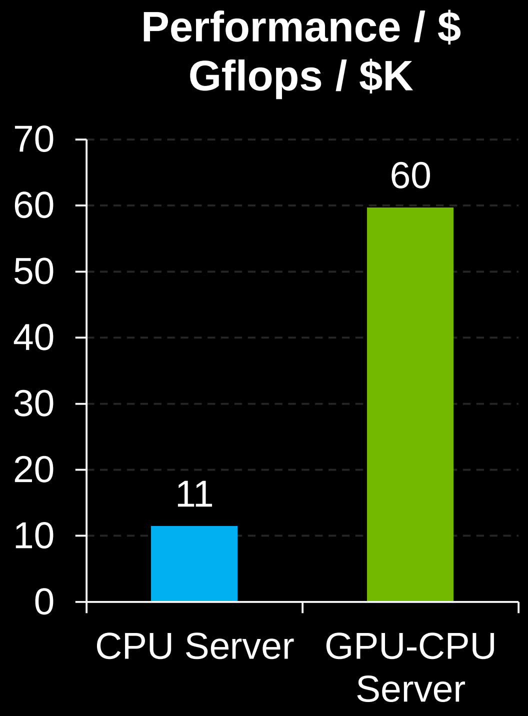
705.1 GFLOPS = 63% of *PEAK* or 89% of *DGEMM*

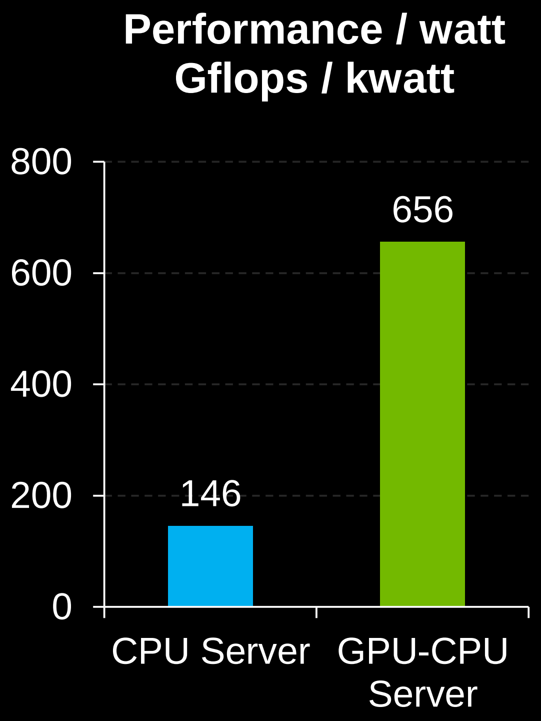
SuperServer 6016GT-TF
2 CPUs + 2 GPUs in 1U

PRESENTED BY NVIDIA.

Single Node Linpack Performance





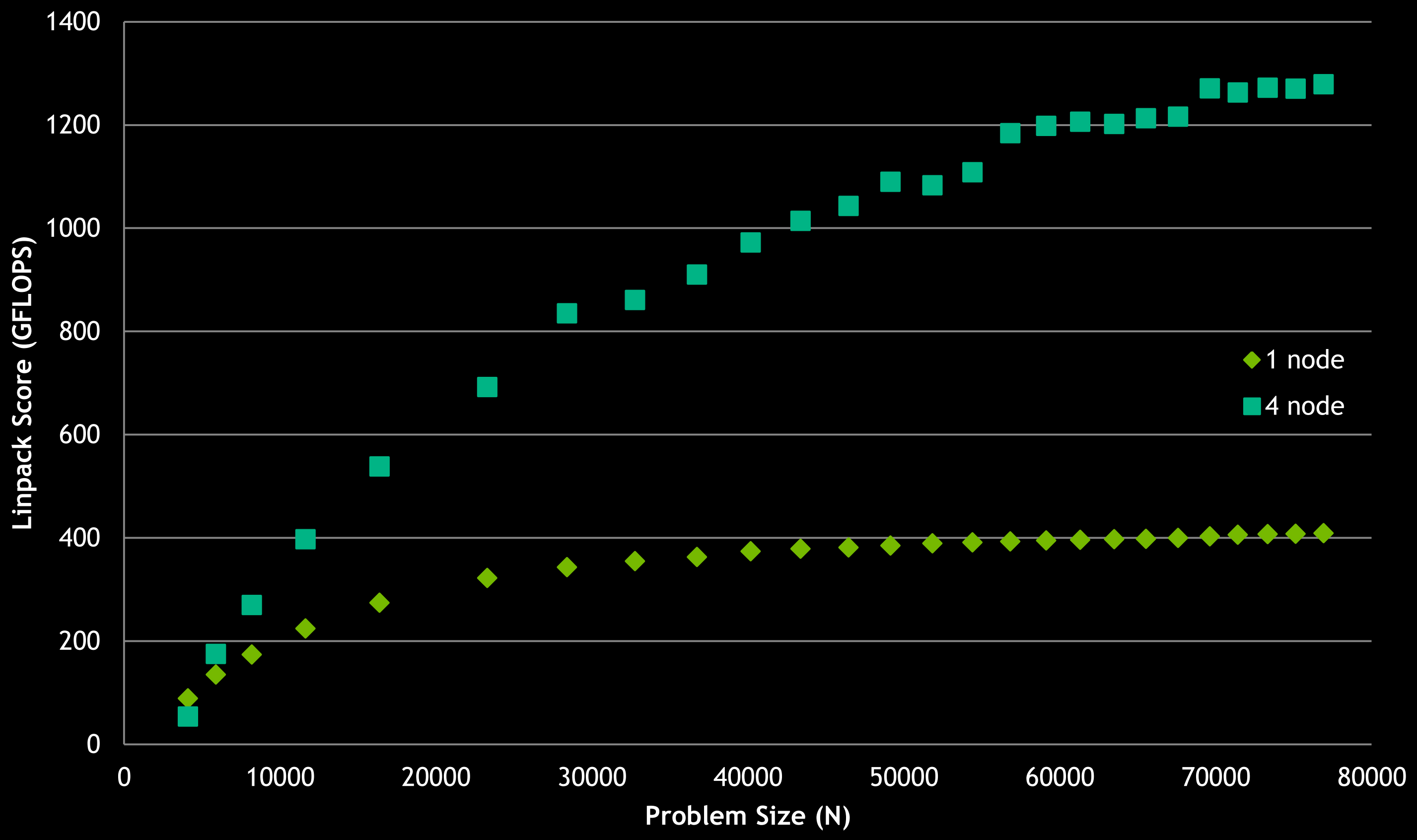


CPU 1U Server: 2x Intel Xeon X5550 (Nehalem) 2.66 GHz, 48 GB memory, $7K, 0.55 kw

GPU-CPU 1U Server: 2x Tesla C2050 + 2x Intel Xeon X5550, 48 GB memory, $11K, 1.0 kw

Effect of Problem Size (memory size)

- Memory use ~ 8 bytes*N*N
- More memory increases performance and efficiency



Cluster Performance

CASPUR Jazz Cluster			
Node	16x HP DL360 G7	**Interconnect**	Infiniband QDR
CPU	2x X5650 @ 2.67 GHz (6 cores)	**Switch**	HP (Voltaire) 36P Managed Switch
RAM	48 GB	**HCA**	HP InfiniBand 4X QDR ConnectX-2 PCIe G2 Dual Port
Peak Perf.	1152 GFLOPS (DP)	**GPU**	NVIDIA S2050 (half x node)
xHPL Perf.	892 GFLOPS (DP)		(2x GPUs C2050 @ 1.15 GHz & 3 GB DDR5 RAM)
Notes	MKL_NUM_THREADS = 6 /CPU		OMPL_NUM_THREADS = 6 /CPU
	Infiniband QDR as OpenIB no RDMA		**GFLOPS: best results per # Procs**
	mpirun -mca tbl openib,self -mca btl_openib_flags 1 -bysocket -bind-to-socket -hostfile ./myhostfile -np #Procs ./run_linpack		

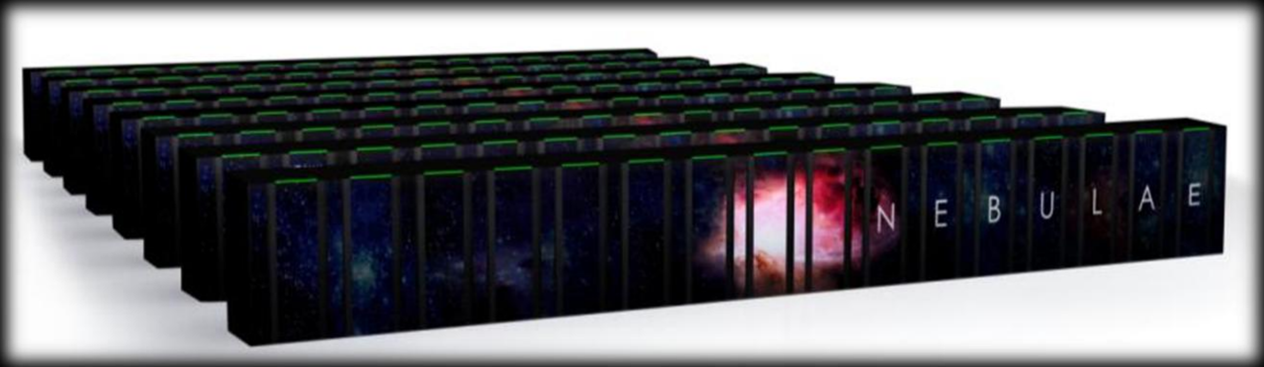
						GFLOPS			
#Nodes	**#Procs**	**PxQ**	**N**	**MemTot(MB)**	**MemNode(MB)**	**Nb=768**	**x Node**	**% xHPL**	**% peak**
1	1	1x1	70000	41870	41870	410	410	80.6%	64.4%
1	2	1x2	70000	41870	41870	640	640	71.7%	55.6%
2	4	2x2	105000	94208	47104	1319	660	73.9%	57.2%
4	8	2x4	150000	192261	48065	2551	638	71.5%	55.4%
6	12	3x4	180000	276855	46143	3814	636	71.3%	55.2%
8	16	4x4	210000	376831	47104	5092	637	71.4%	55.3%
10	20	4x5	235000	471893	47189	6295	630	70.6%	54.6%
12	24	4x6	256000	560000	46667	7498	625	70.0%	54.2%
14	28	4x7	280000	669922	47852	8771	627	70.2%	54.4%
15	30	5x6	290000	718628	47909	9147	610	68.4%	52.9%
16	32	4x8	300000	769043	48065	10050	628	70.4%	54.5%

Nebulae Cluster

#2 on Latest Top500

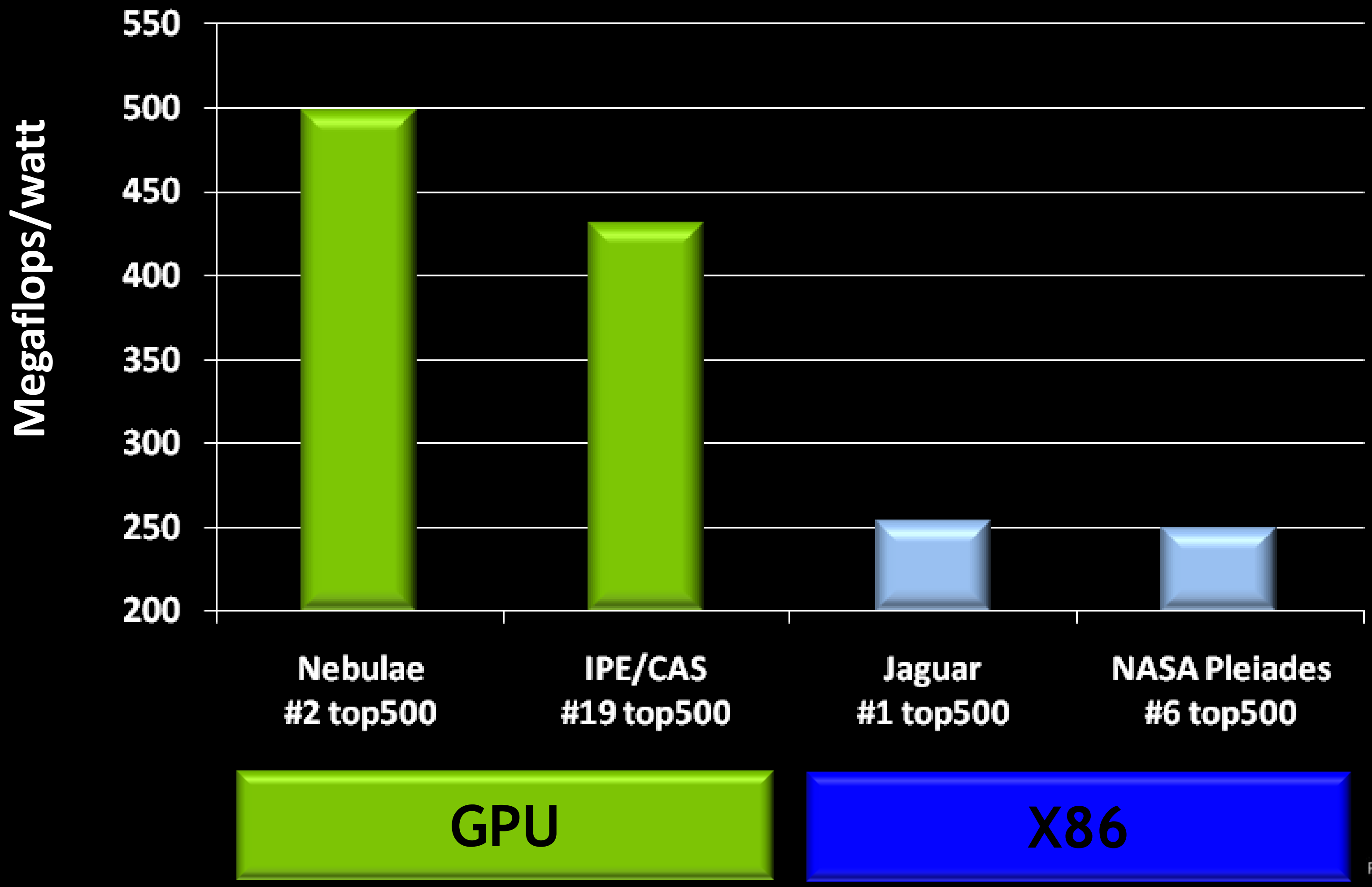
4640 Tesla T20 GPUs

1.271 PetaFlops



```
T/V                  N        NB     P     Q              Time                 Gflops
--------------------------------------------------------------------------------
WR13C2L4    2359296   768    32    145          6886.10         1.271e+06
--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=  0.0033806 ...... PASSED
```

Cluster Performance / Watt



Future Challenges

- Compute growing faster than bandwidths
  - System Memory, PCIe, Network
- Optimize HPL code to reduce GPU idle time
  - Pipeline swap / update
- Dynamic Task Scheduling / Load Balancing
  - PLASMA / MAGMA / StarPU
  - Task Dependency Graph / Performance Models

Conclusions

- Easy to accelerate the Linpack performance of workstations and clusters using Tesla and CUDA

- Increasing the performance per node reduces the cost of high performance interconnects on clusters

- Improved Power Efficiency

- Code is available from NVIDIA