A CUDA IMPLEMENTATION OF THE HPCG BENCHMARK

Everett Phillips

Massimiliano Fatica

OUTLINE

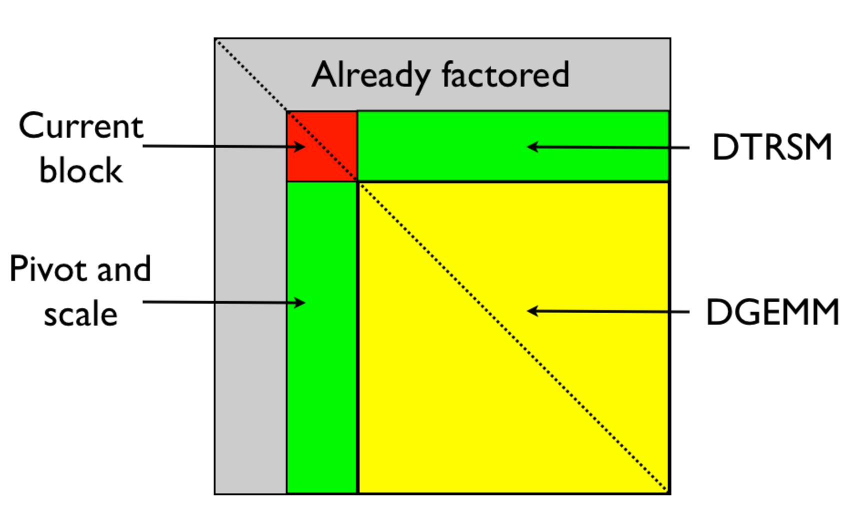
High Performance Conjugate Gradient Benchmark

- Motivation
- Overview
- Optimization
- Performance Results
  - Single GPU
  - GPU Supercomputers
- Conclusion

WHY HPCG ?

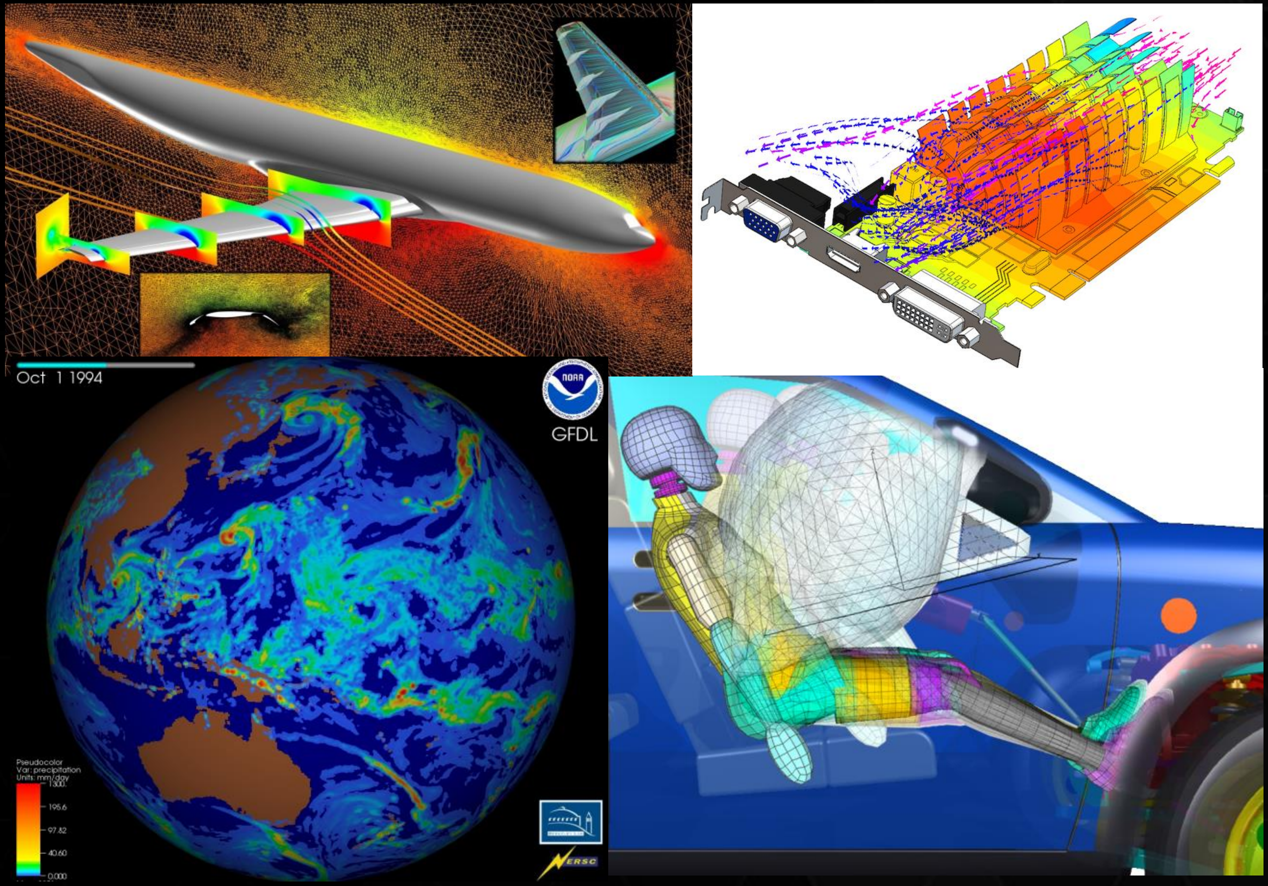
HPL (Linpack) Top500 benchmark

- Supercomputer Ranking / Evaluation
- Dense Linear Algebra (Ax = b)
- Compute intensive
    - DGEMM (Matrix-Matrix Multiply)
    - O(N3)FLOPS / O(N2) Data
    - 10-100 Flop/Byte
- Workload does not correlate with many modern applications



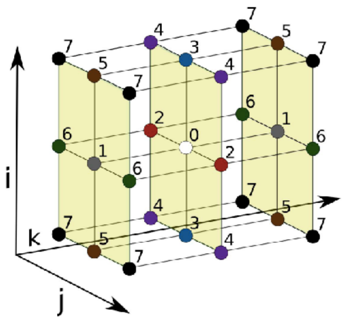
WHY HPCG?

- New Benchmark to Supplement HPL
- Common Computation Patterns not addressed by HPL
- Numerical Solution of PDEs
- Memory Intensive
- Network

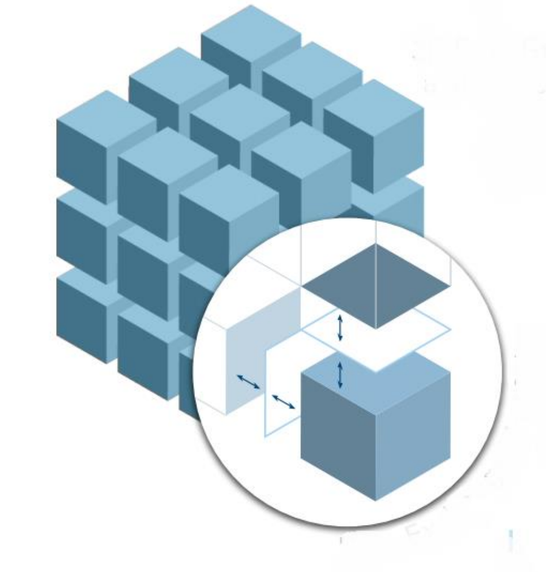


HPCG BENCHMARK

- Preconditioned Conjugate Gradient Algorithm
  - Sparse Linear Algebra (Ax = b), Iterative solver
  - Bandwidth Intensive: **1/6 Flop/Byte**
- Simple Problem (sparsity pattern of Matrix A)
  - Simplifies matrix generation/solution validation
  - Regular 3D grid, 27-point stencil
  - Nx x Ny x Nz local domain / Px x Py x Pz Processors
  - Communications: boundary + global reduction

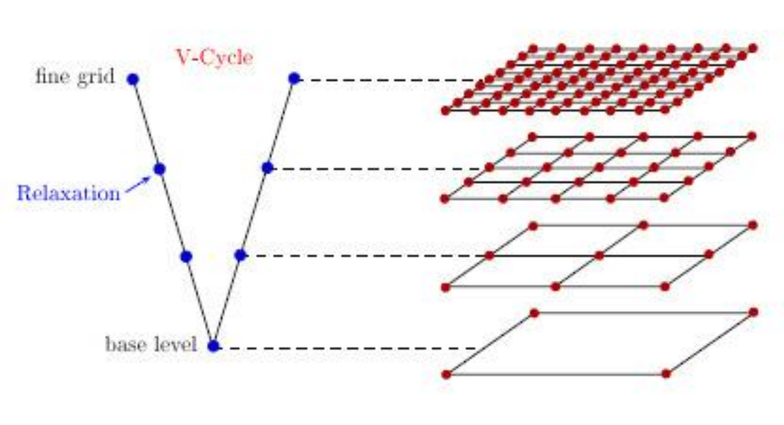


27-point stencil operator



HPCG ALGORITHM

- Multi-Grid Preconditioner
  - Symmetric-Gauss-Seidel Smoother (SYMGS)



- Sparse Matrix Vector Multiply (SPMV)
- Dot Product – MPI_Allreduce()

Algorithm 1 Preconditioned Conjugate Gradient

$k = 0$
Compute the residual $r_0 = b - Ax_0$
while $(\boxed{||r_k||} < \epsilon)$ **do**
    $\boxed{z_k = M^{-1} r_k}$
    $k = k + 1$
    if $k = 1$ **then**
        $p_1 = z_0$
    else
        $\beta_k = \boxed{r_{k-1}^T z_{k-1}} / r_{k-2}^T z_{k-2}$
        $p_k = z_{k-1} + \beta_k p_{k-1}$
    end if
    $\alpha_k = r_{k-1}^T z_{k-1} / \boxed{p_k^T \boxed{Ap_k}}$
    $x_k = x_{k-1} + \alpha_k p_k$
    $r_k = r_{k-1} - \alpha_k Ap_k$
end while
$x = x_k$

HPCG BENCHMARK

- Problem Setup – initialize data structures
- Optimization (required to expose parallelism in SYMGS smoother)
  - Matrix analysis / reordering / data layout
  - Time counted against final performance result
- Reference Run – 50 iterations with reference code – Record Residual
- Optimized Run – converge to Reference Residual
  - Matrix re-ordering slows convergence (55-60 iterations)
  - Additional iterations counted against final performance result
  - Repeat to fill target execution time (few minutes typical, 1 hour for official run)

HPCG

SPMV (y = Ax)

```
Exchange_Halo(x)  //neighbor communications
for row = 0 to nrows
    sum ← 0
    for j = 0 to nonzeros_in_row[ row ]
        col ← A_col[ j ]
        val ← A_val[ j ]
        sum ← sum + val * x[ col ]
    y[ row ] ← sum
```

No dependencies between rows, safe to process rows in parallel

HPCG

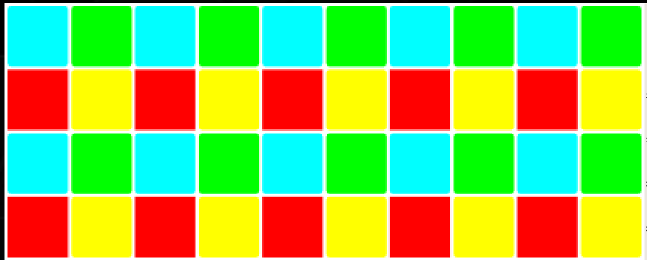
SYMGS (Ax = y, smooth x)

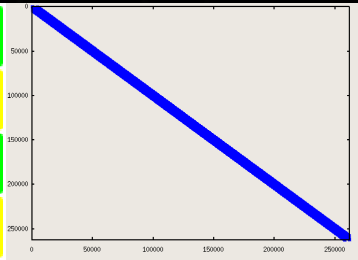
```
Exchange_Halo(x)  //neighbor communications
for row = 0 to nrows  (Fwd Sweep, then Backward Sweep for row = nrows to 0)
    sum ← b[ row ]
    for j = 0 to nonzeros_in_row[ row ]
        col ← A_col[ j ]
        val ← A_val[ j ]
        if( col != row )      sum ← sum – val * x[ col ]
    x[ row ] ← sum / A_diag[ row ]
```

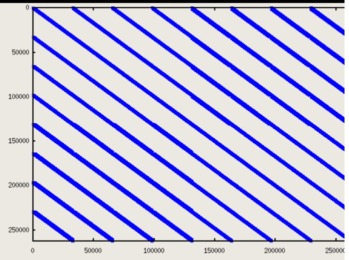
if col < row, must wait for x[col] to be updated

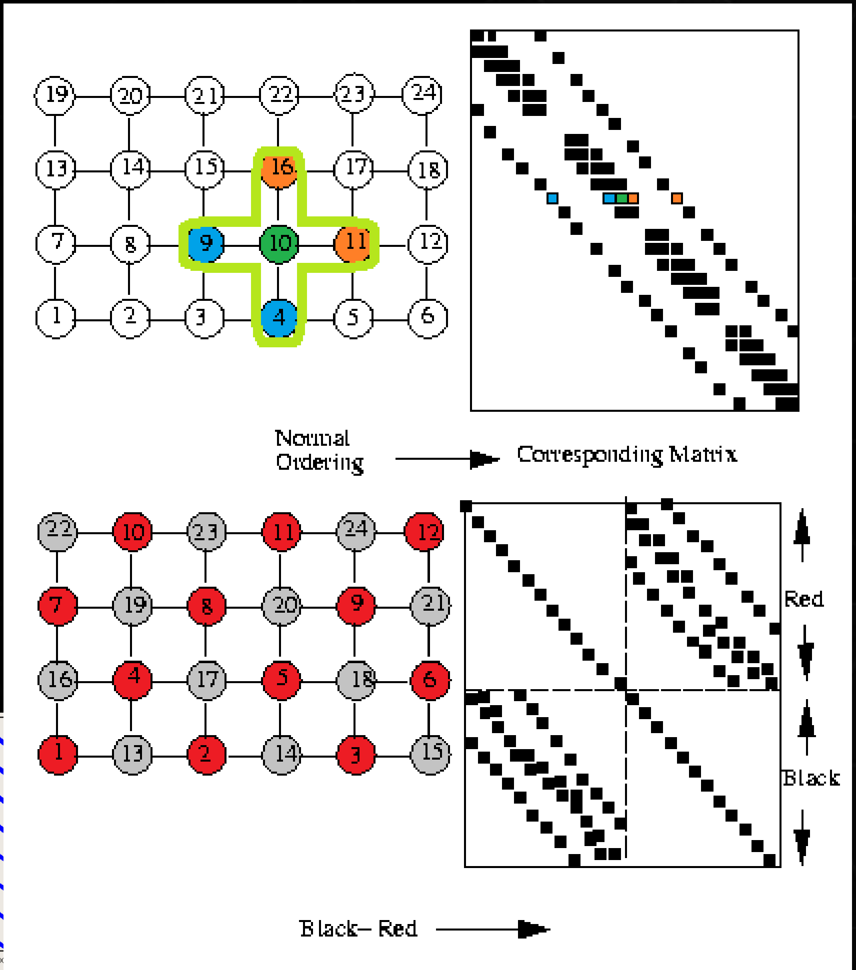
MATRIX REORDERING (COLORING)

- SYMGS - order requirement
  - Previous rows must have new value
  - reorder by color (independent rows)
  - 2D example: 5-point stencil -> red-black
  - 3D 27-point stencil = 8 colors



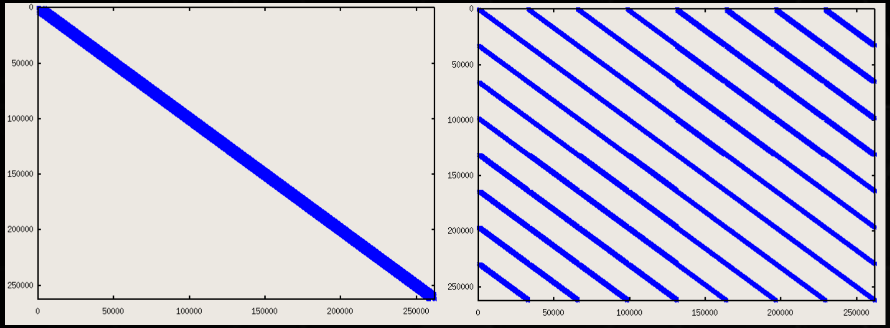






MATRIX REORDERING (COLORING)

- Coloring to extract parallelism
- Assignment of “color” (integer) to vertices (rows), with no two adjacent vertices the same color
- “Efficient Graph Matching and Coloring on the GPU” – (Jon Cohen)
    - Luby / Jones-Plassman based algorithm
    - Compare hash of row index with neighbors
    - Assign color if local extrema
    - Optional: recolor to reduce # of colors



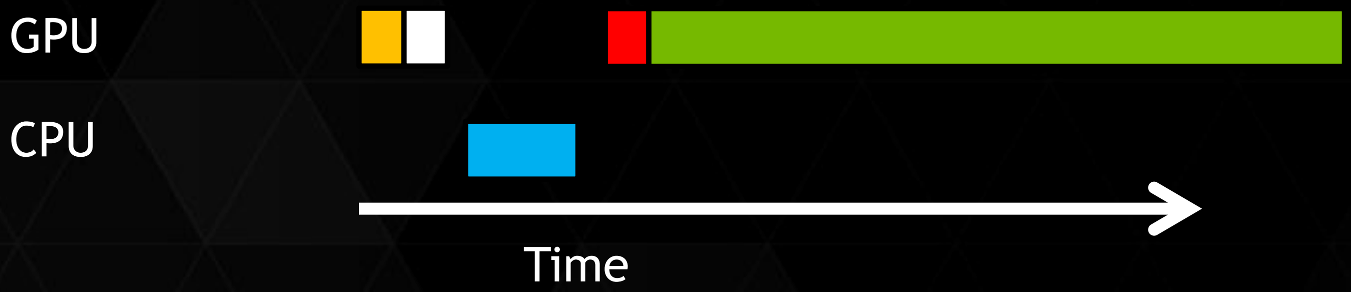
MORE OPTIMIZATIONS

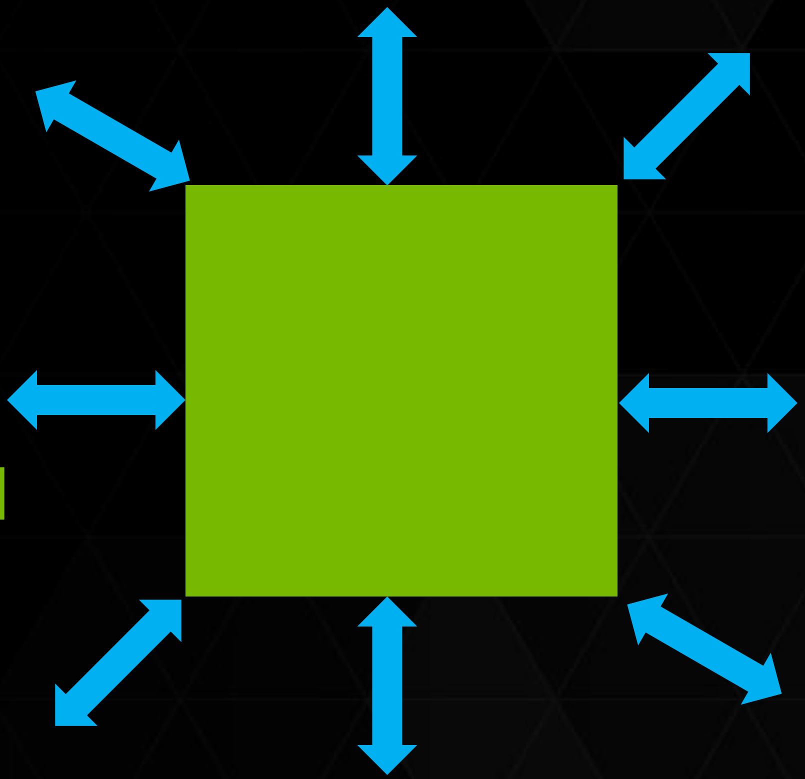
- Overlap Computation with neighbor communication
- Overlap 1/3 MPI_Allreduce with Computation
- __LDG loads for irregular access patterns (SPMV + SYMGS)

OPTIMIZATIONS

- SPMV Overlap Computation with communications

- Gather to GPU send_buffer
  Copy send_buffer to CPU
  MPI_send / MPI_recv
  Copy recv_buffer to GPU
  Launch SPMV Kernel

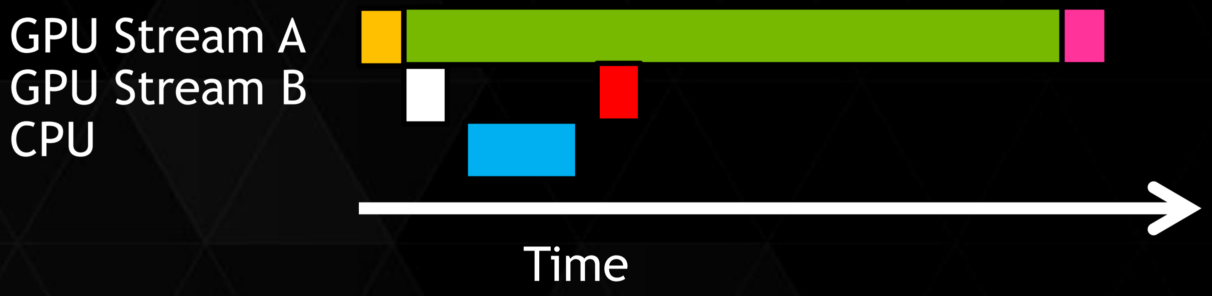


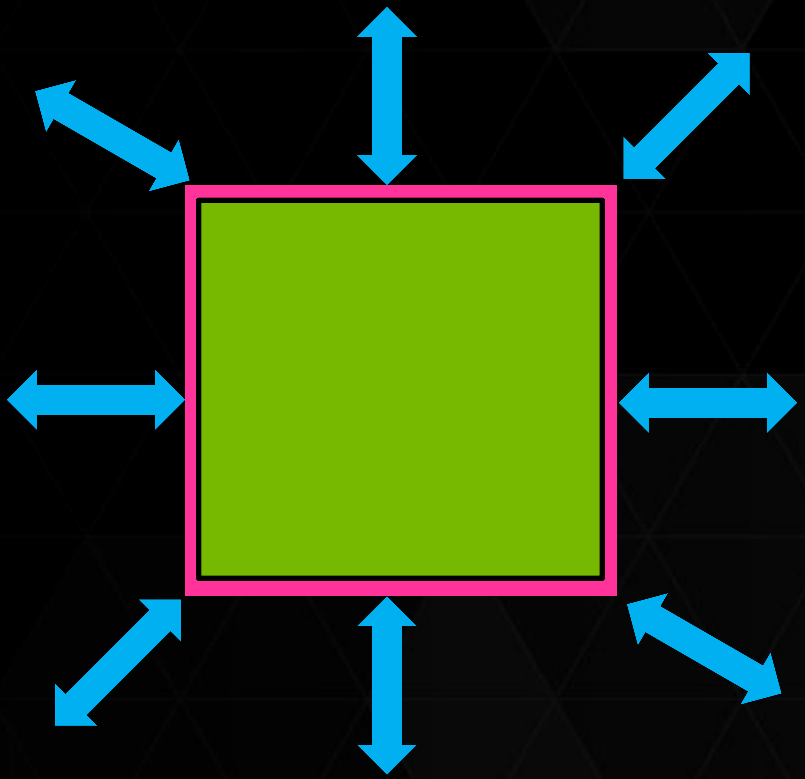


OPTIMIZATIONS

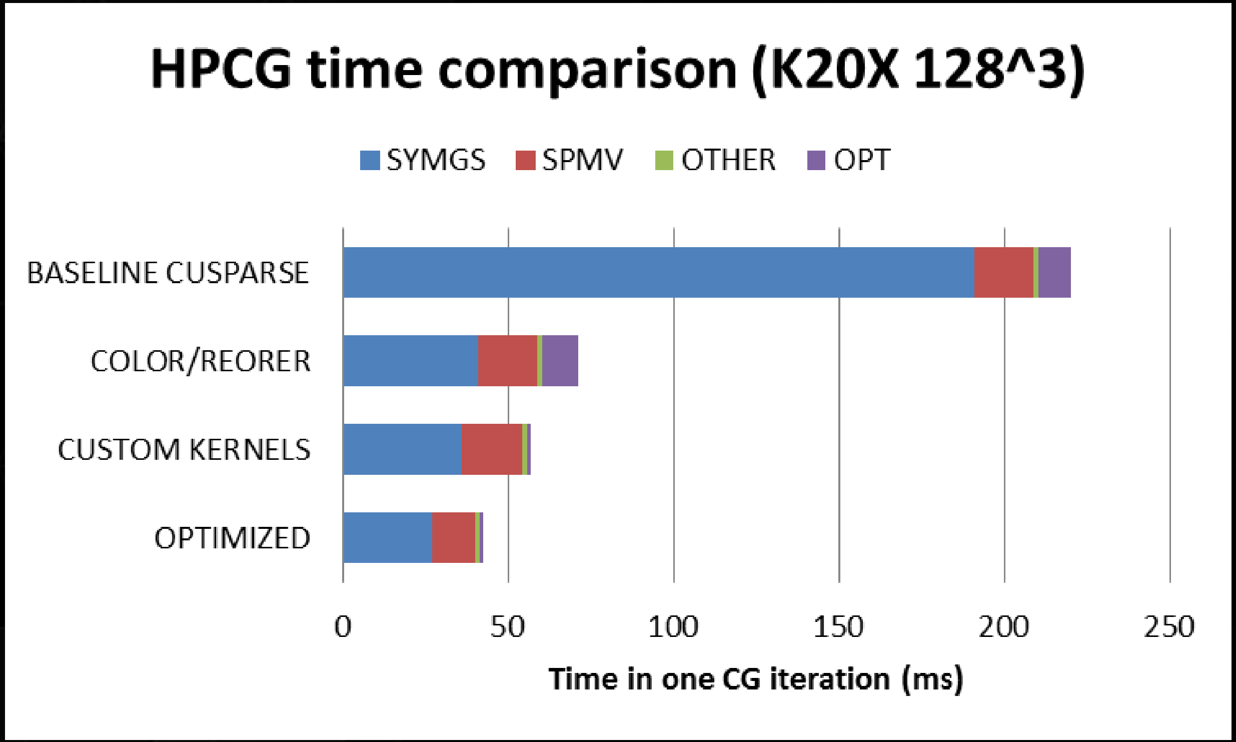
- SPMV Overlap Computation with communications

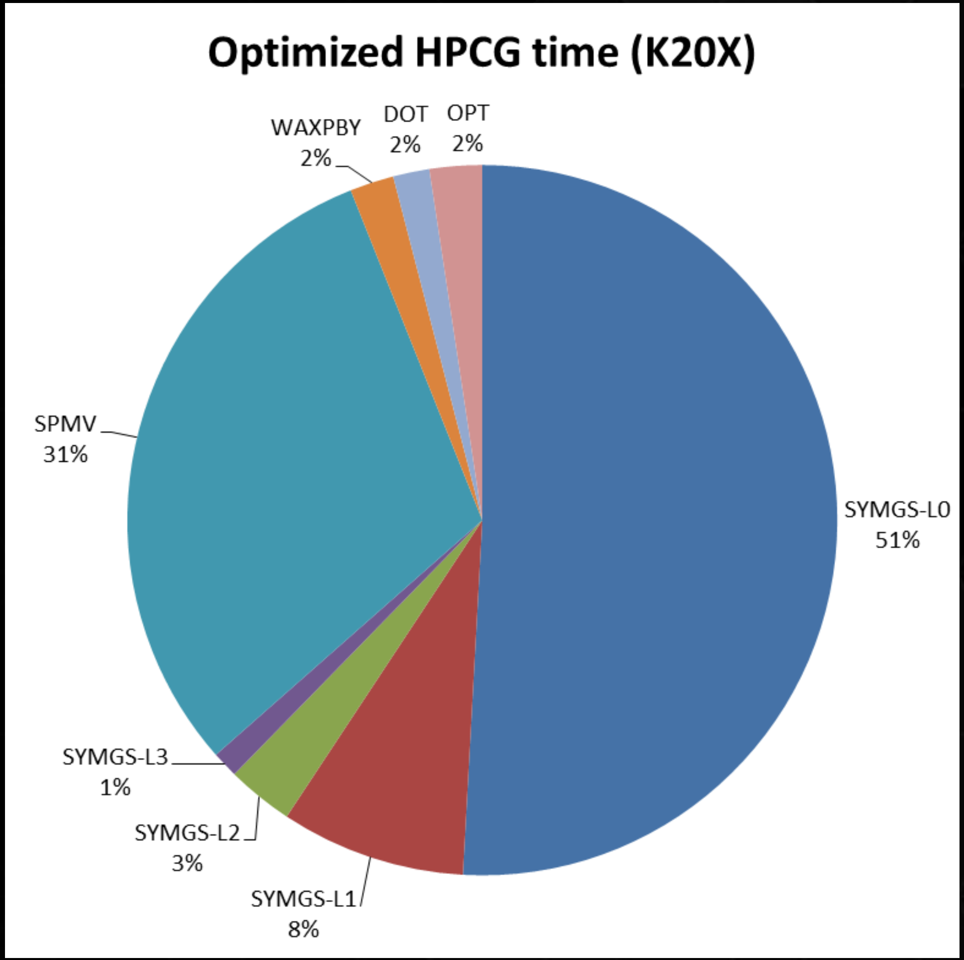
- Gather to GPU send_buffer
  Copy send_buffer to CPU
  Launch SPMV interior Kernel
  MPI_send / MPI_recv
  Copy recv_buffer to GPU
  Launch SPMV boundary Kernel



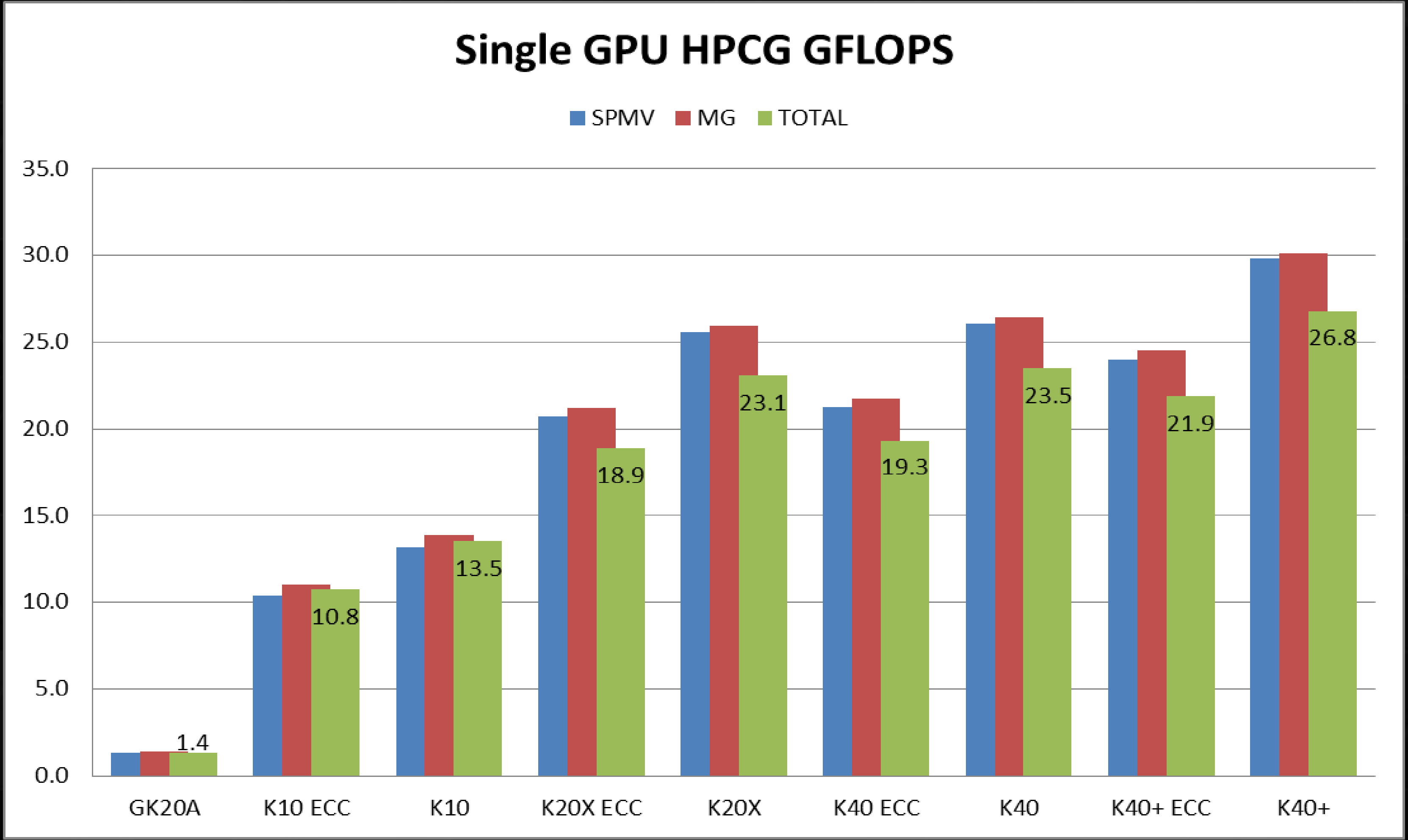


RESULTS – SINGLE GPU

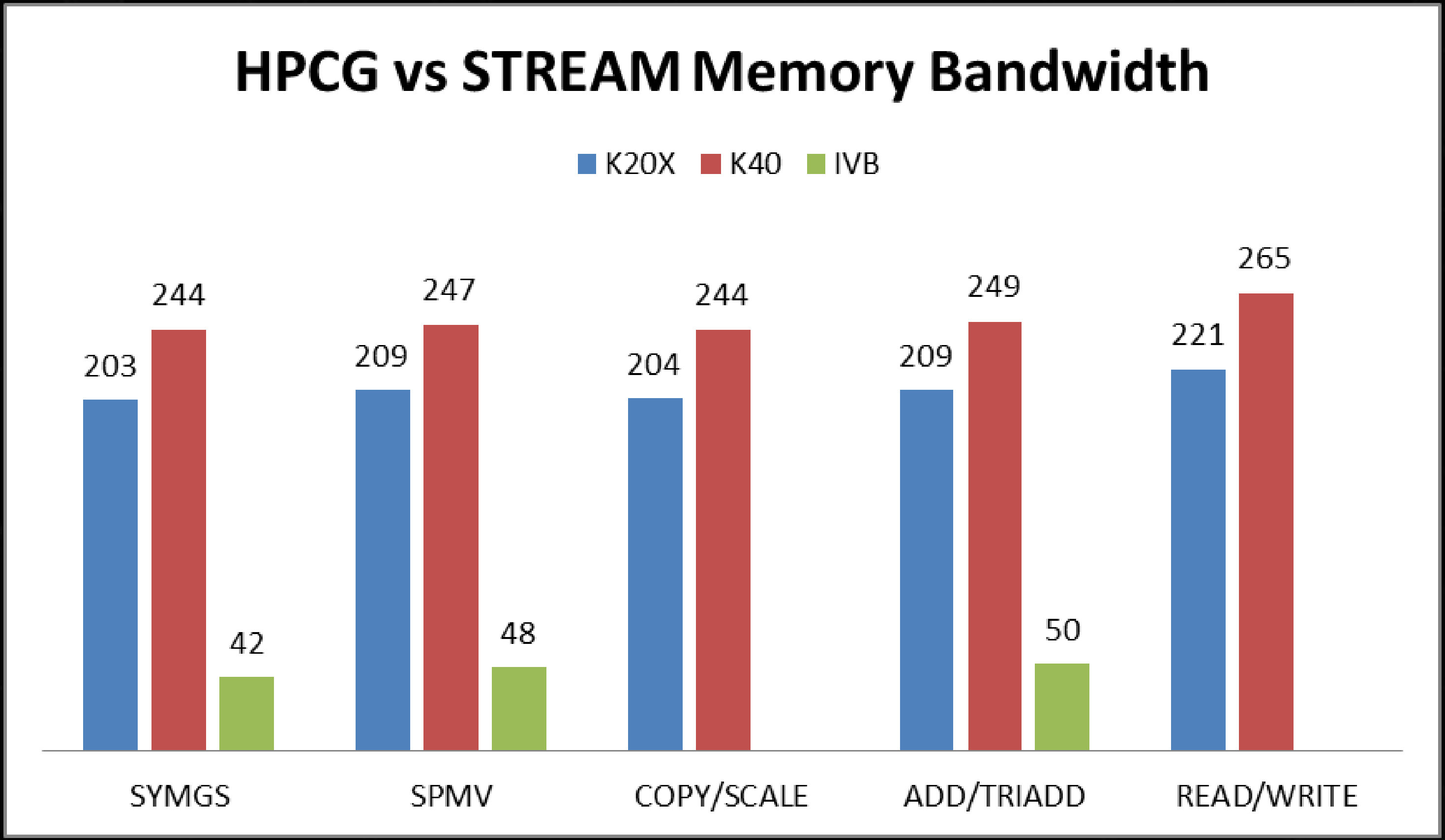




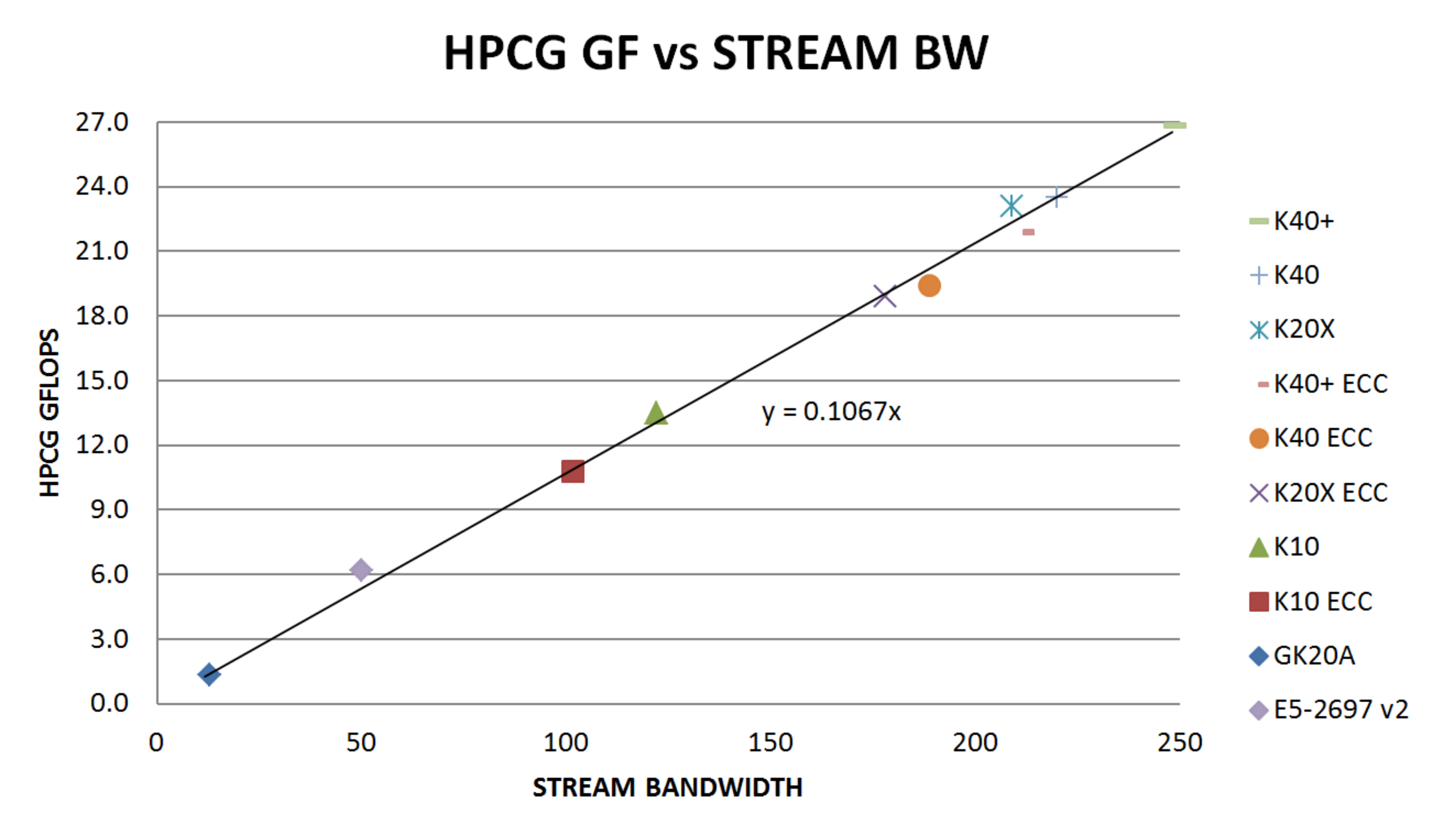
RESULTS – SINGLE GPU



RESULTS - SINGLE GPU

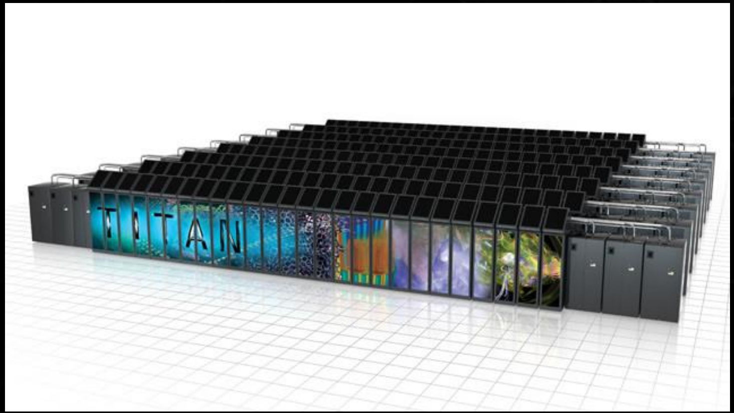


RESULTS – SINGLE GPU



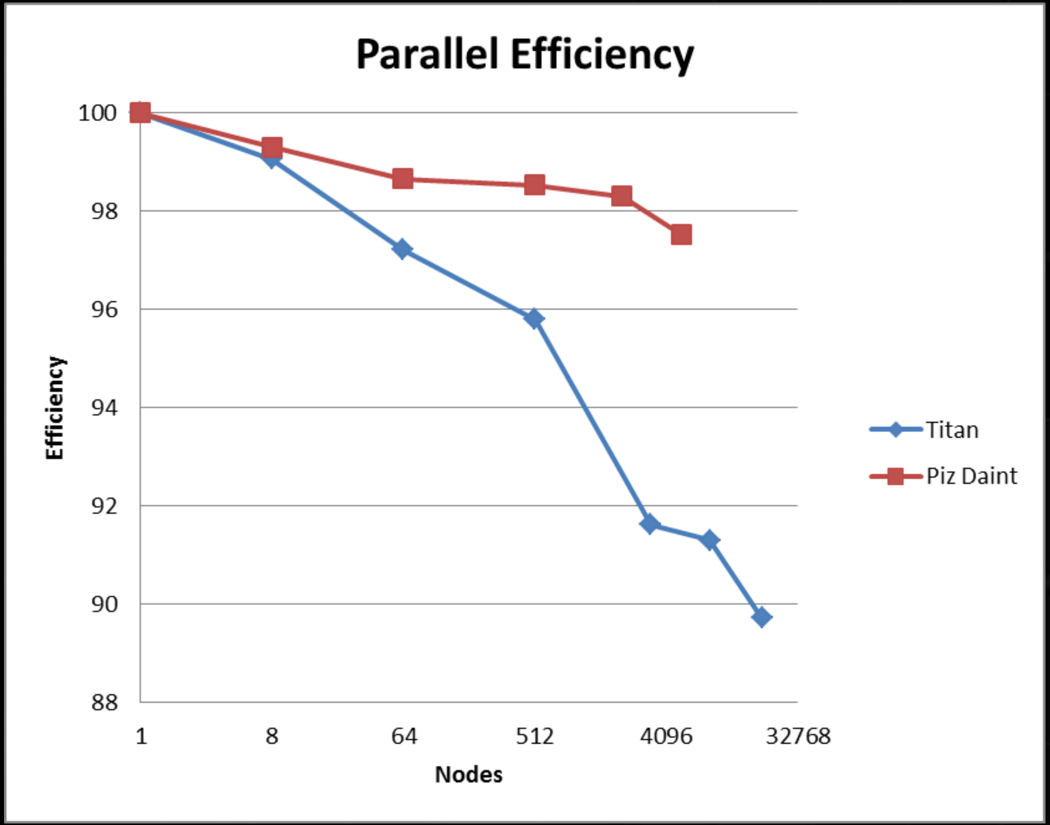
RESULTS – GPU SUPERCOMPUTERS

- Titan @ ORNL
  - Cray XK7, 18688 Nodes
  - 16-core AMD Interlagos + K20X
  - Gemini Network - 3D Torus Topology
- Piz Daint @ CSCS
  - Cray XC30, 5272 Nodes
  - 8-core Xeon E5 + K20X
  - Aries Network – Dragonfly Topology



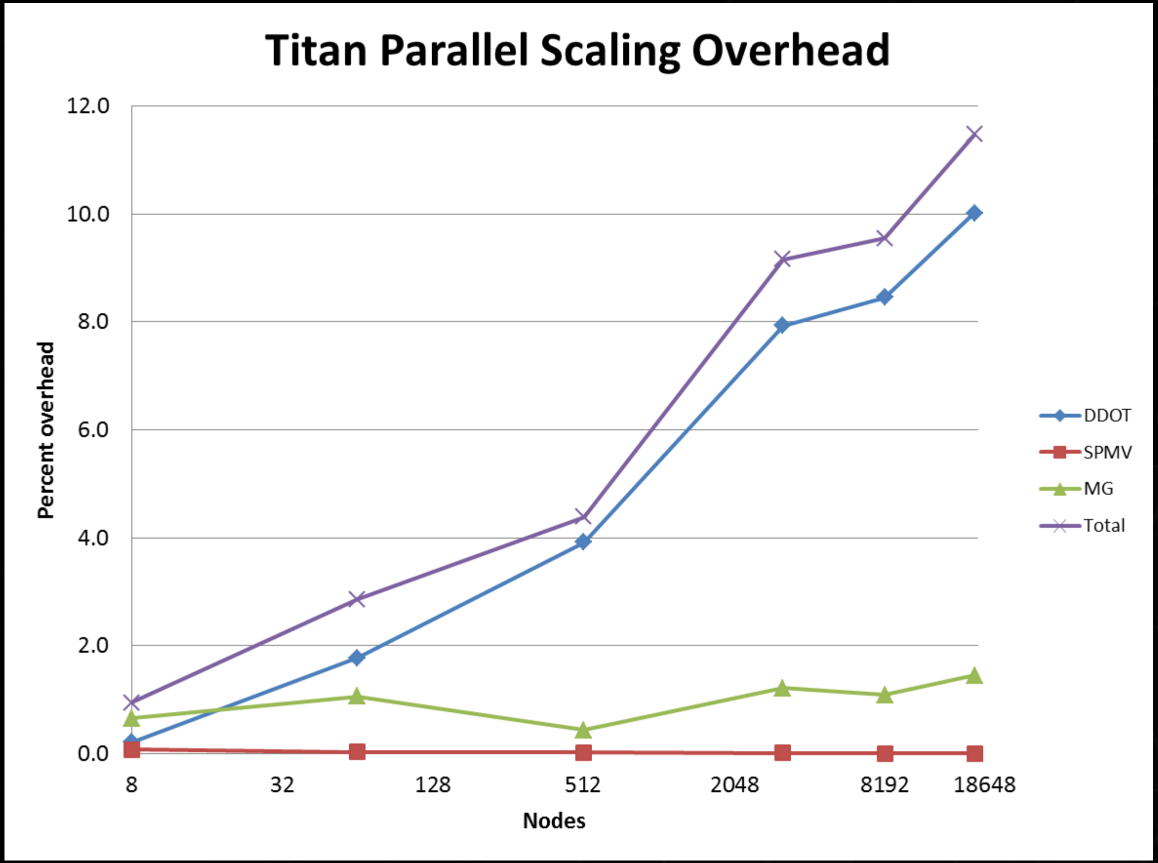
RESULTS – GPU SUPERCOMPUTERS

- 1 GPU = 20.8 GFLOPS (ECC ON)
- ~7% iteration overhead at scale
- Titan @ ORNL
  - 322 TFLOPS (18648 K20X)
  - 89% efficiency (17.3 GF per GPU)
- Piz Daint @ CSCS
  - 97 TFLOPS (5265 K20X)
  - 97% efficiency (19.0 GF per GPU)



RESULTS – GPU SUPERCOMPUTERS

- DDOT (-10%)
  - MPI_Allreduce()
  - Scales as Log(#nodes)
- MG (-2%)
  - Exchange Halo (neighbor)
- SPMV (-0%)
  - Overlapped w/Compute



SUPERCOMPUTER COMPARISON

HPCG Rank	System	HPCG GFLOPS	Itera-tions	#Procs	Processor Type	HPCG Per Proc	Bandwidth Per Proc	Efficiency FLOP/BYTE
1	Tianhe-2	580,109	57	46,080	Xeon-Phi-31S1P	12.59 GF	320 GB/s	0.039
2	K	426,972	51	82,944	Sparc64-viiifx	5.15 GF	64 GB/s	0.080
3	Titan	322,321	55	18,648	Tesla-K20X+ECC	17.28 GF	250 GB/s	0.069
5	Piz-Daint	98,979	55	5,208	Tesla-K20X+ECC	19.01 GF	250 GB/s	0.076
8	HPC2	49,145	54	2,610	Tesla-K20X+ECC	18.83 GF	250 GB/s	0.075
	HPC2	60,642	54	2,600	Tesla-K20X	23.32 GF	250 GB/s	0.093

CONCLUSIONS

- GPUs proven effective for HPL, especially for power efficiency
  - High flop rate
- GPUs also very effective for HPCG
  - High memory bandwidth
  - Stacked memory will give a huge boost
- Future work will add CPU + GPU

ACKNOWLEDGMENTS

- Oak Ridge Leadership Computing Facility (ORNL)
  - Buddy Bland, Jack Wells and Don Maxwell
- Swiss National Supercomputing Center (CSCS)
  - Gilles Fourestey and Thomas Schulthess
- NVIDIA
  - Lung Scheng Chien and Jonathan Cohen