Achieving strong scaling in many-GPU calculations in Lattice QCD

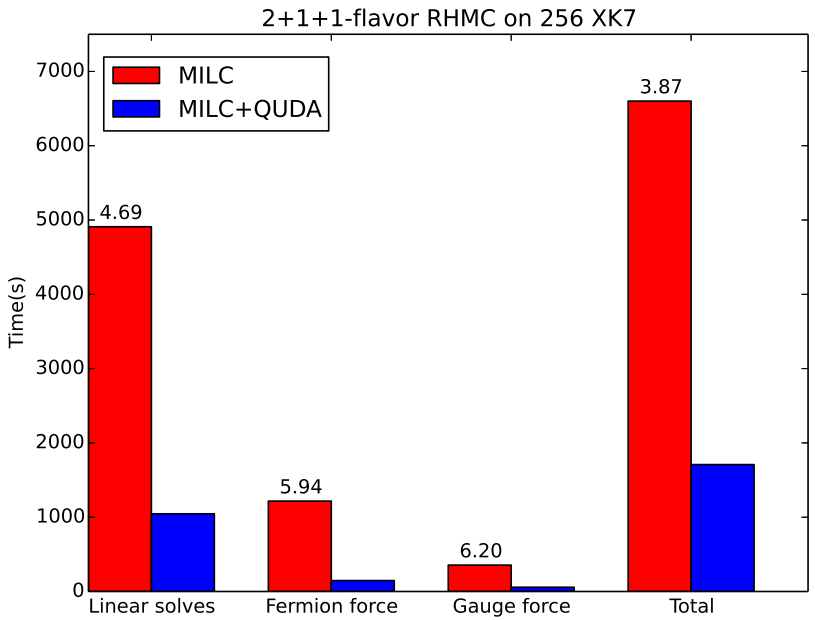
Justin Foley, M. Clark

Microway, NVIDIA

Lattice 2014

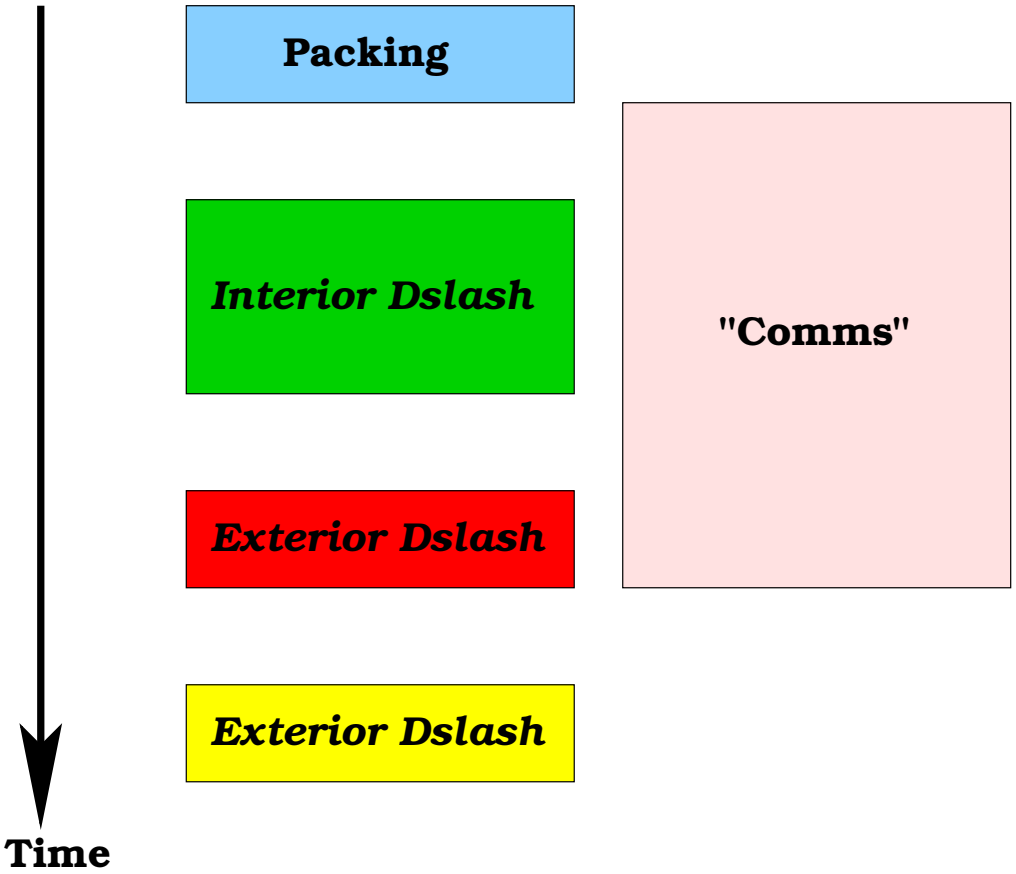
Introduction

- Majority of MILC routines now supported by QUDA
- Chroma utilizes QUDA + QDP-jit
- Amdahl's law no longer the dominant performance bottleneck
- Gauge generation requires scaling of linear solvers to 100s of GPUs and beyond



- Recent concerted efforts in software optimization and algorithmic development

Sample Multi-GPU Dslash profile from previous QUDA releases



- A representative Dslash profile for 2D lattice partitioning

Key features

- Packing kernel copies halo data into a (device) buffer
- "Comms" can include host-device transfers, host-network-buffer transfers, message passing
- Cuda streams enable concurrent kernel execution and host-device copies
- Packing and Dslash kernels execute sequentially - packing has priority
- Separate exterior Dslash kernels for each paritioned dimension
- Significant launch overheads
- Host launches work sequentially

New Dslash software optimizations

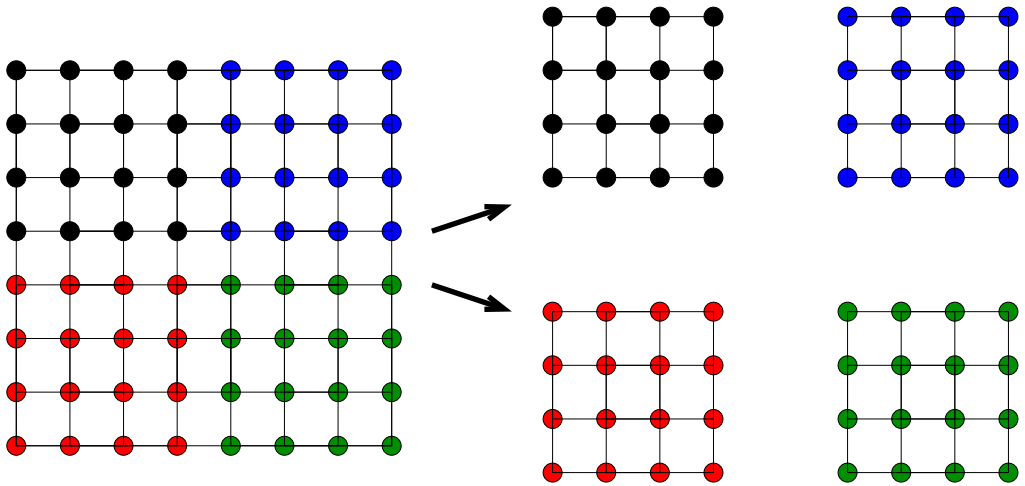
- GPUDirect RDMA now supported in QUDA - communication over IB can completely bypass host memory
- Refactoring of kernel autotuning feature $\Rightarrow$ signficant reduction in launch overhead
- Exploring latency reduction by fusing exterior Dslash kernels
- Host threading using Pthreads - launch kernels and issue messages concurrently
- Stream priorities on compute 3.5 devices and above - concurrent packing and interior Dslash kernel execution

Domain-decomposition preconditioning

- Solve the preconditioned linear system

$$MA\phi = M\eta,$$

  where $M \approx A^{-1}$, but involves less or no inter-processor communication.
- Non-overlapping Additive Schwarz preconditioning
- Evaluate $A^{-1}\rho$ on each lattice subdomain ignoring interprocessor communication
  $M\rho = \sum_{d=1}^{N_D} A_d^{-1}\rho_d$

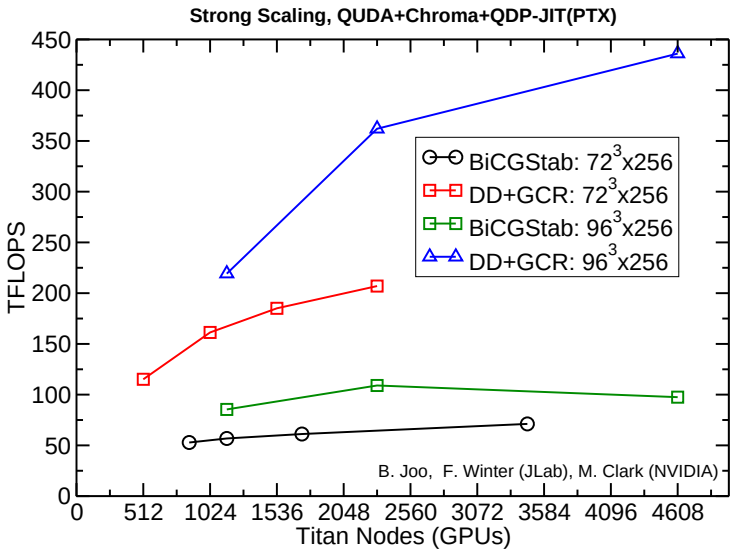


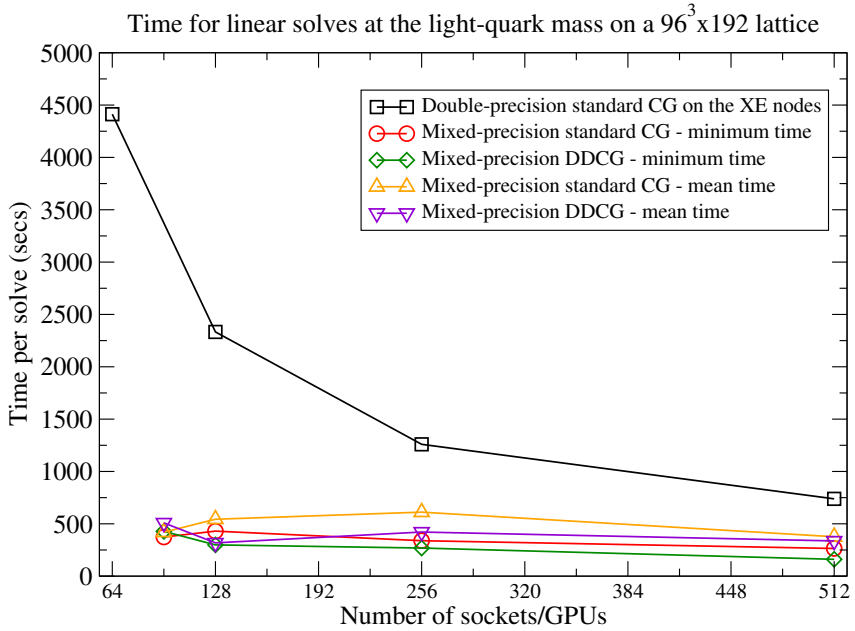
Additive Schwarz preconditioning

- M is block-diagonal in space-time indices (Block Jacobi)
- Zero Dirichlet boundary conditions on each subdomain $\Rightarrow$ $\kappa(A_d) < \kappa(A)$
- Since M is a preconditioner, implement approximately (half-precision data types, small number of inner solver iterations)
- Use MR or steepest-descent algorithm in the preconditioning step

Additive Schwarz preconditioning

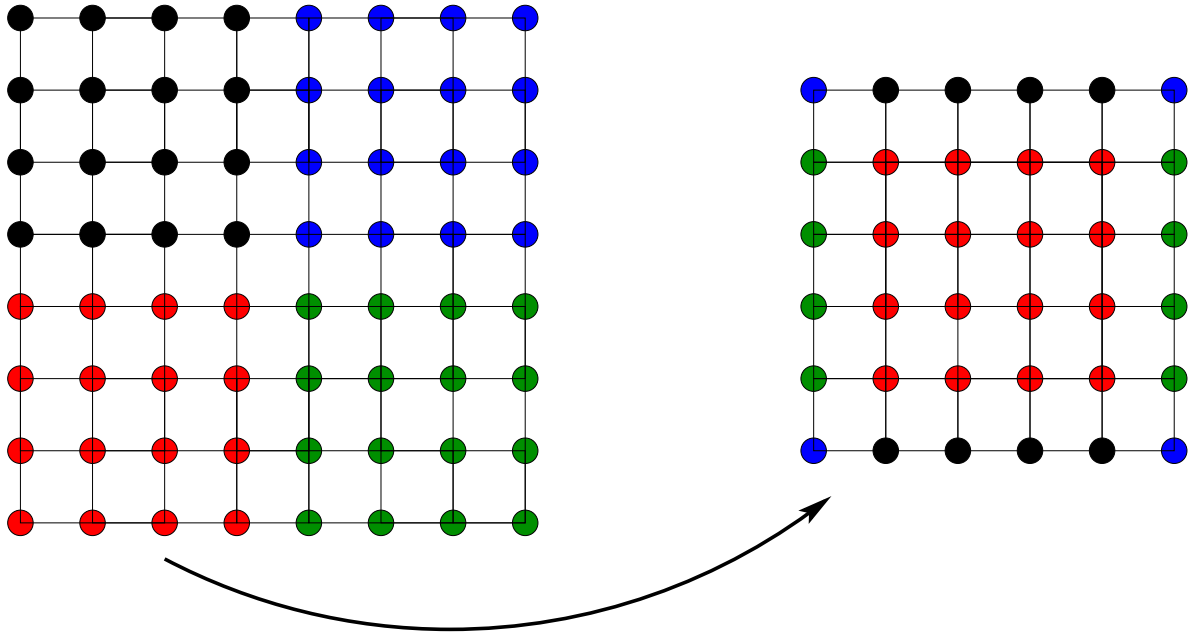
- Impressive results with Clover and GCR
- No observed benefit in HISQ CG





Restricted Additive Schwarz preconditioning

- In the preconditioner, overlap domains to mitigate boundary



effects

- Preconditioning involves restriction to the interior of each domain (red points) which violates Hermiticity
- RAS-preconditioned HISQ GCR breaks even. What about Clover?
- Hermiticity-preserving overlapping Additive Schwarz - C. DeTar, H. Na, C. Winterowd

S-step Krylov solvers

- In linear solvers, communication occurs in the Dslash operation and the calculation of vector inner products
- Inner products involve global synchronizations - GPUs empty out
- Standard solver formulations involve short recurrence relations
- Each iteration extends the Krylov subspace by a single vector
- Tight coupling between Dslash and inner products

S-step Krylov solvers

- S-step solvers separate the generation of Krylov basis vectors and inner product computations
- Opportunities for communication coalescing
- Each iteration extends the Krylov basis by s ($s > 1$) vectors
- Basis vectors are generated up front
- Need to evaluate the same number of inner products as in the single-step formulation, but now (at least) s inner-product calculations can be coalesced into a single computation
- Reduces the number of global synchronizations by a factor s

Sketch of s-step CG

$i = 0$ // labels outer iterations
$r_0 = b - Ax_0$
$r_{i<0} = 0$
while(!converged) do:

Starting with r_si, generate a set of s+1 linearly-independent vectors, V_i, in Span($r_{si}, Ar_{si}, \ldots, A\hat{}sr_{si}$)

for $j = 1, 2, \ldots s$:
Compute r_{si+j}, x_{si+j}
(Requires 2s+1 inner products involving residuals from previous outer iteration and V_i, which can be coalesced into a single reduction)
end for

$i = i + 1$
end do

- By extending lattice subdomains, communications for the Dslash operations can also be coalesced
- Straightforward to implement in QUDA (recycle RAS routines)
- Stability issues:
  - In finite-precision, s-step methods become unstable for large s
  - Hoemenn (2010) - Problem is particularly severe for the monomial basis: $V_i = [r_{si}, Ar_{si}, \ldots, A^s r_{si}]$
  - Chronopoulos and Gear (1989): CG convergence degrades when $s > 5$
    Later extended to $s = 16$ by orthogonalizing V_i (introduces additional computation and communication)
  - Hoemenn: use alternative bases

S-step status

- Prototype implementation of Hoemenn's CA-CG solver in QUDA
- To begin with, test stability and convergence properties using staggered quarks
- Unfortunately, still in debugging phase - (hindered by typos in original description)
- For orthogonalization, utilize highly-optimized, parallel TSQR (Demmel et al. 2008)
- Magma implementation in the works

Summary

- Many-pronged approach to achieving strong-scaling in QUDA applications
- Software optimizations
  - Autotuning optimizations
  - Kernel fusion
  - Host threading
- Utilizing new hardware features
  - GPUDirect RDMA
- Algorithmic Developments
  - Additive Schwarz Preconditioning
  - Restricted Additive Schwarz
  - S-step solvers
- All under active development, but expect features to filter through to release code over next six months