Summit & Beyond for the Intensity Frontier

Jiqun Tu[1,*]

[1]**Columbia University**

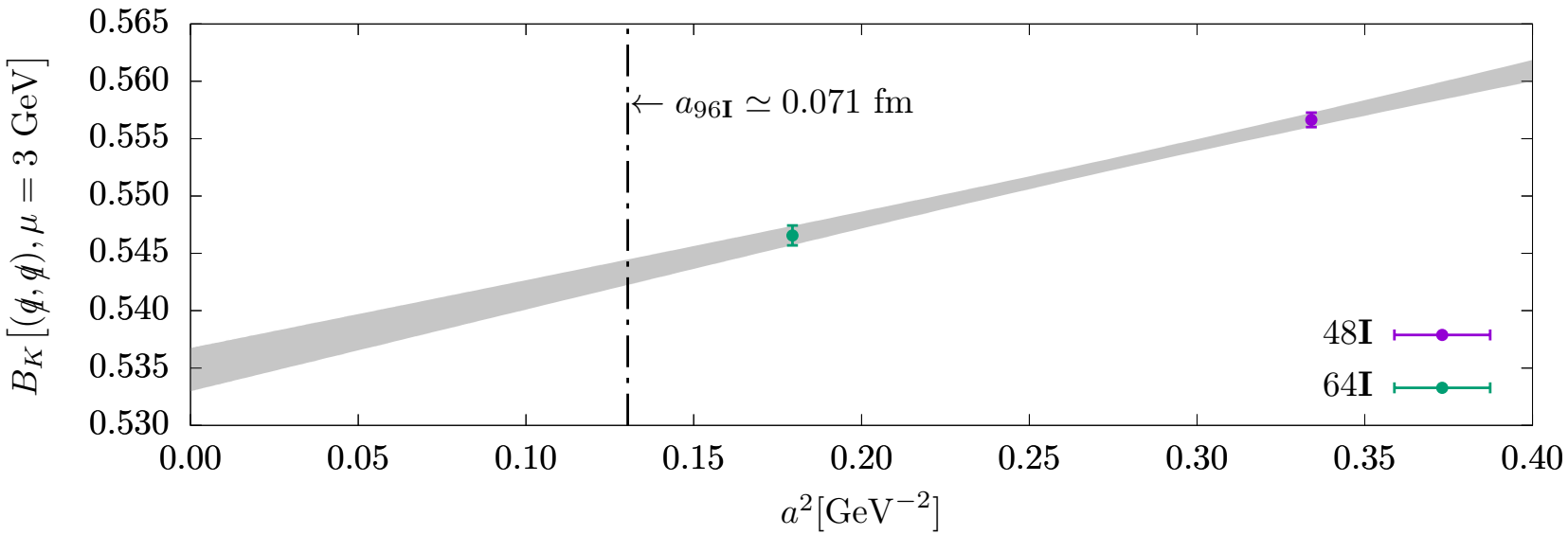
2019 Lattice X Intensity Frontier Workshop, BNL
September 24, 2019

*This is a combined work of the RBC & UKQCD collaborations. Major contributions to this work by Chulwoo Jung, Robert Mawhinney and David Murphy. Substantial hardware and software support for QUDA implementation from Kate Clark (NVIDIA).

Target Ensemble Evolution

RBC/UKQCD ensembles with 2+1 flavor Möbius domain wall fermion and Iwasaki gauge action.

Ens.	β	$L^3 \times T \times L_s$	m_π[MeV]	$m_\pi L$	a^{-1}[GeV]
48I	2.13	$48^3 \times 96 \times 24$	139.1(4)	3.8633(63)	1.7293(36)
64I	2.25	$64^3 \times 128 \times 12$	139.0(5)	3.7778(84)	2.3572(69)
96I	2.31	$96^3 \times 192 \times 12$	$\simeq 141$	$\simeq 4.88$	$\simeq 2.77$



Light Quarks: u and d.

$$\det D \det D = \det(D^\dagger D) = \int \Big[d\phi^\dagger\Big]\Big[d\phi\Big] \exp\Big[- \|D^{-1}\phi\|^2\Big]$$

Fermion Determinants

Evolution is dominated by solution of the Dirac equations $Ax = b$ (under the even-odd preconditioning):

$$\begin{pmatrix} M_{ee} & M_{eo} \\ M_{oe} & M_{oo} \end{pmatrix} \begin{pmatrix} \psi_e \\ \psi_o \end{pmatrix} = \begin{pmatrix} \phi_e \\ \phi_o \end{pmatrix},$$

with the underlying matrix

$$A = D^{\dagger}_{prec} D_{prec} = \left[1 - M^{-1}_{ee} M_{eo} M^{-1}_{oo} M_{oe}\right]^{\dagger} \left[1 - M^{-1}_{ee} M_{eo} M^{-1}_{oo} M_{oe}\right].$$

These equations are solved with the conjugate gradient algorithm.

Conjugate Gradient (CG)

Algorithm 1 Conjugate Gradient $Ax = b$

$r_0 = b - Ax_0$

$p_0 = r_0$

$k = 0$

while have not converged **do**

$\alpha_k = \langle r_k, r_k \rangle / \langle p_k, Ap_k \rangle$ $\leftarrow$ matrix multiplication/dslash

$x_{k+1} = x_k + \alpha_k p_k$

$r_{k+1} = r_k - \alpha_k A p_k$

$\beta_k = \langle r_{k+1}, r_{k+1} \rangle / \langle r_k, r_k \rangle$

$p_{k+1} = r_{k+1} + \beta_k p_k$

$k = k + 1$

end while

Move Over, China: U.S. Is Again Home to World's Speediest Supercomputer

June 8, 2018

Figure 1: Source: New York Times.

Summit at ORNL

- The Summit machine at Oak Ridge National Laboratory (ORNL), with a peak double precision performance of $\simeq 200$ peta-flops, is currently the fastest supercomputer in the world.
- It is one of the pre-exascale machines for DoE's exascale computing project (ECP).
- 6 Tesla V100 GPUs per node with a total of 4608 nodes. Our proposed jobs run on 1024 nodes (6096 GPUs) at a time.
- CPS (Columbia Physics System) with QUDA [**1109.2935**] (https://github.com/lattice/quda) solvers.

CG on Summit

A brief summary: **communication** is the bottleneck.

	supply	demand with plain CG
compute flops (fp32)	15.7 tera-flops	3965 mega-ops
memory bandwidth	900 GB/sec	276.4 MB
network bandwidth	8.3 GB/sec	17.3 MB
compute/memory	17.4 ops/byte	14.3 ops/byte (0.8×supply)
network/memory	0.009	0.062 (7.0×supply)

Table 1: Compute flops, memory and network bandwidth demand versus supply for one iteration of CG on Summit. Note that this is only an illustration that does not take into account the control flow structure containing the computing, memory loading/storing and communication.

Algorithm 2 Preconditioned Conjugate Gradient $Ax = b$

$r_0 = b - Ax_0$

$p_0 = z_0 = M^{-1} r_0$

$k = 0$

while have not converged **do**

$\alpha_k = \langle r_k, z_k \rangle / \langle p_k, Ap_k \rangle$

$x_{k+1} = x_k + \alpha_k p_k$

$r_{k+1} = r_k - \alpha_k A p_k$ $\leftarrow$ matrix multiplication/dslash, comm.

$z_{k+1} = M^{-1} r_{k+1}$ $\leftarrow$ preconditioning solve, better no comm.

$\beta_k = \langle z_{k+1}, r_{k+1} \rangle / \langle z_k, r_k \rangle$

$p_{k+1} = z_{k+1} + \beta_k p_k$

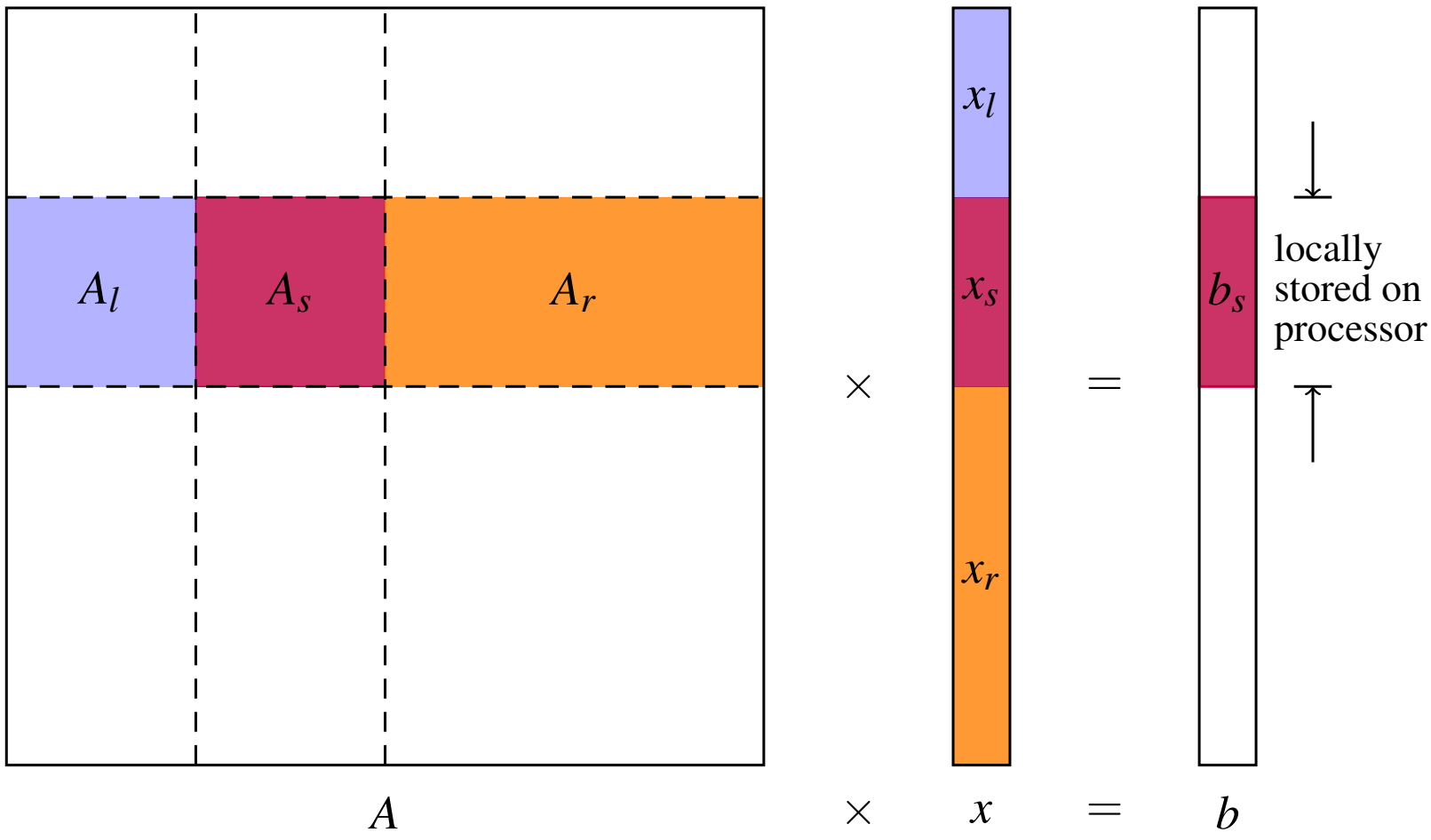
$k = k + 1$

end while

Multisplitting Algorithm

For reference see [**D. O'leary 1985**].

$$Ax = b : A_l x_l + A_s x_s + A_r x_r = b_s$$



Multisplitting Algorithm

Solve

$$A_l x_l + A_s x_s + A_r x_r = b_s,$$

Rearrange into an iterative form

$$\begin{aligned} A_s x_s^{(k+1)} &= b_s - A_l x_l^{(k)} - A_r x_r^{(k)} \\ &= b_s - \left(A x^{(k)} - A_s x_s^{(k)} \right) \\ &= r^{(k)} + A_s x_s^{(k)} \equiv \hat{b}_s^{(k)} \end{aligned}$$

For each cycle,

- use communication to calculate the right-hand-side $\hat{b}_s$.
- solve $A_s x_s^{(k+1)} = \hat{b}_s^{(k)}$ locally.
- the updated solution $x_s^{(k+1)}$ will be used to ready the next cycle.

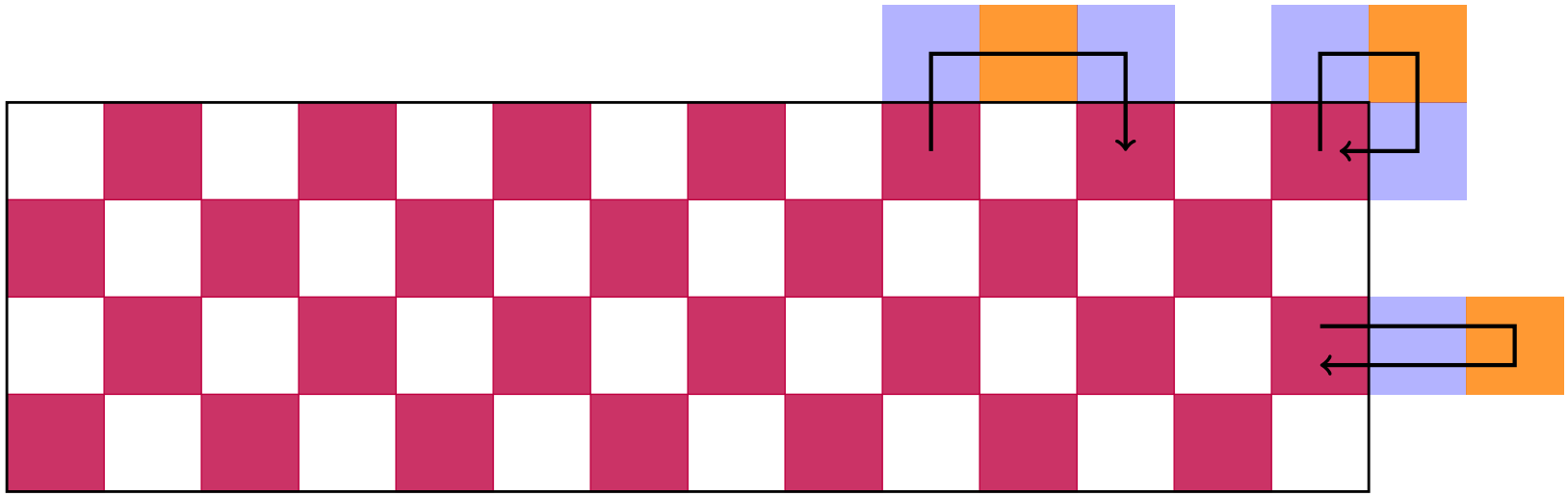
Get A_s for each node by chopping off all off-block-diagonal terms: applying zero Dirichlet boundary condition.

The Normal Operator

- 4 hopping terms in the normal operator:

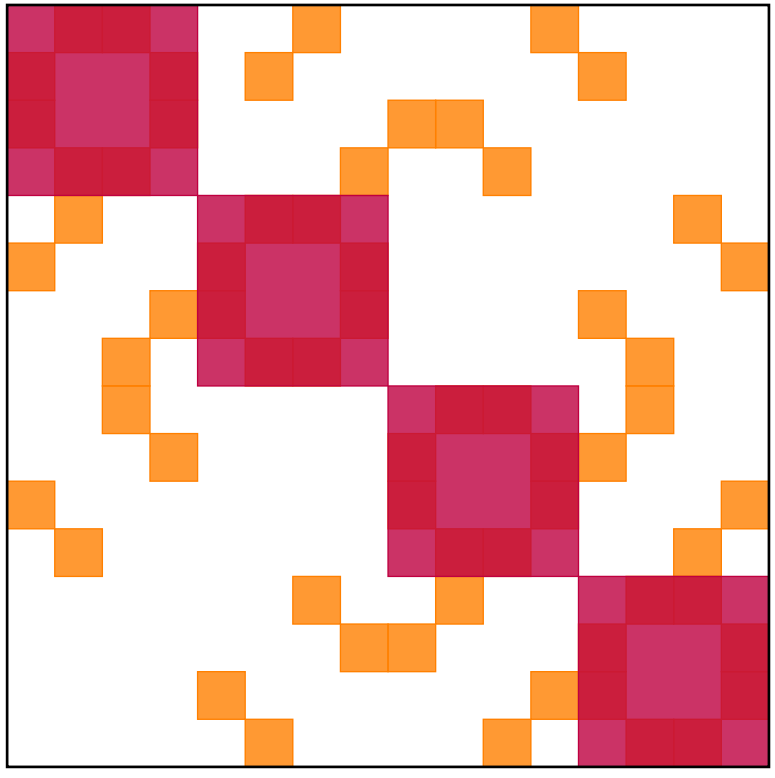
$$A = D^{\dagger}_{prec} D_{prec} = \left[1 - M^{-1}_{ee} M_{eo} M^{-1}_{oo} M_{oe}\right]^{\dagger} \left[1 - M^{-1}_{ee} M_{eo} M^{-1}_{oo} M_{oe}\right].$$

- Need to apply Dirichlet boundary condition on A instead of the individual hopping terms $M_{eo/oe}$: the *snake* terms, terms that hop out of the boundary and hop back.

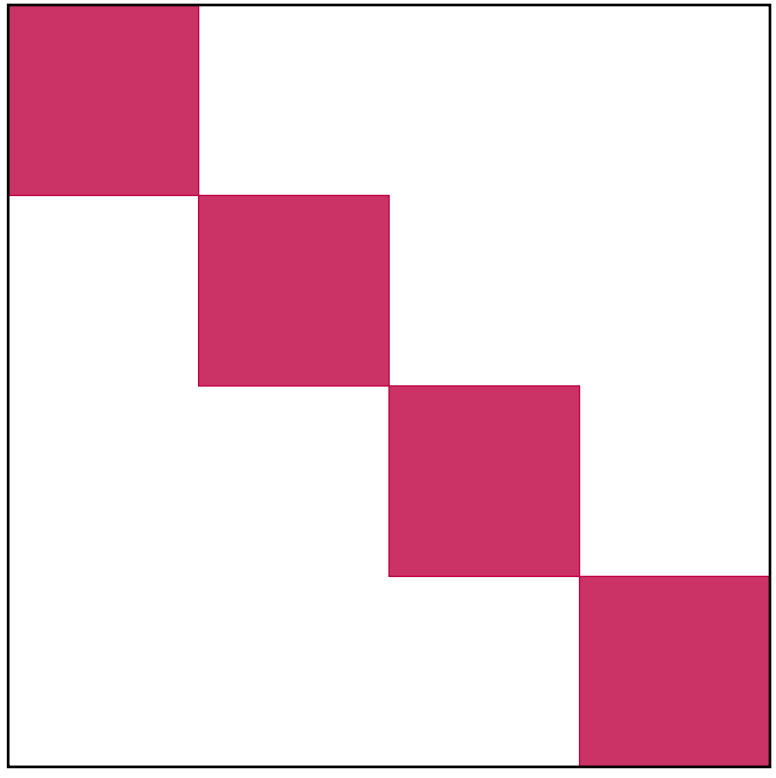


As a Preconditioner

Multisplitting preconditioned CG (MSPCG, effectively the same as additive Schwarz) [**1811.08488**].



A



$$M = \bigoplus_s A_s$$

As a Preconditioner

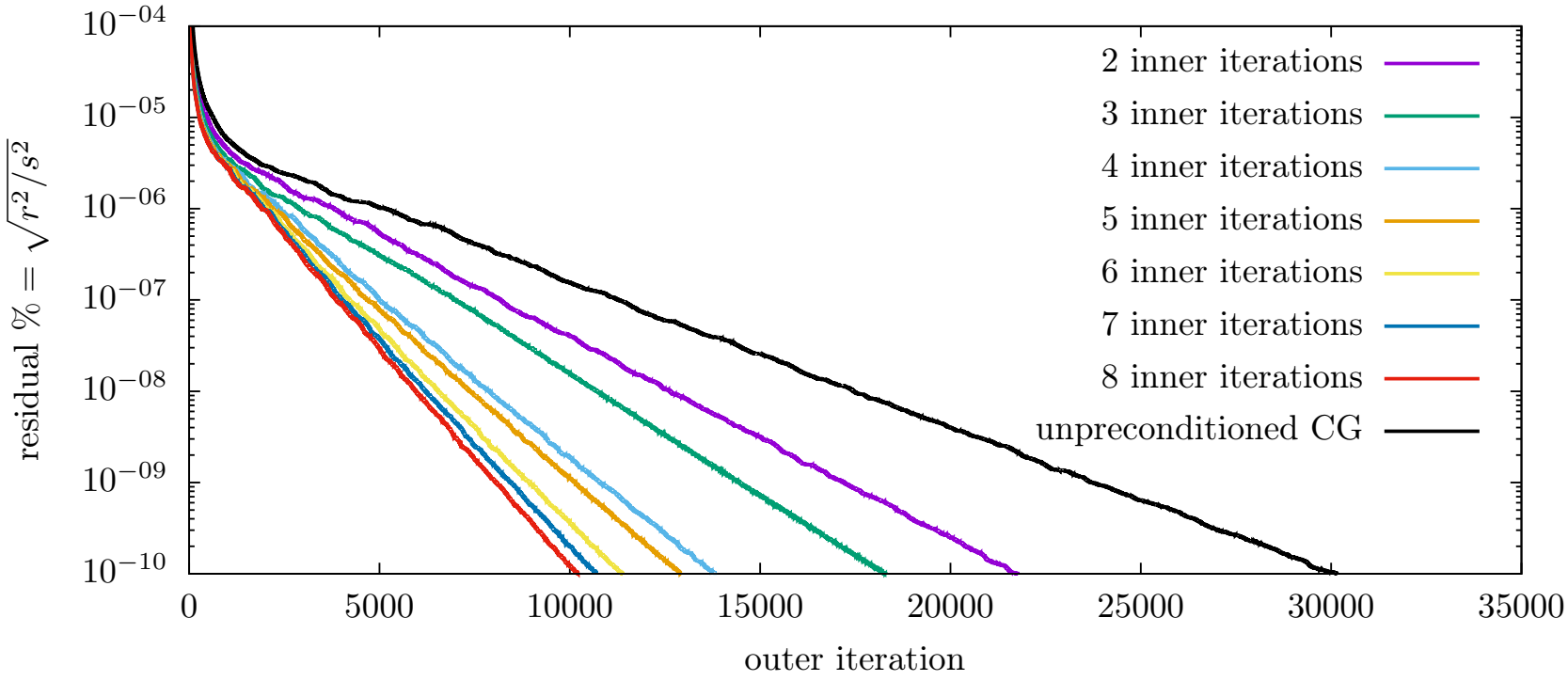


Figure 2: Residual versus (outer) iteration of the MSPCG on Summit solving Dirac equation $M^\dagger M x = M^\dagger y$ to the accuracy of 10^{-10} on the $96^3 \times 192 \times 12$, 2+1 flavor Möbius domain wall fermion, $a^{-1} \simeq 2.77$ GeV lattice with physical pion mass. y is a gaussian random source vector.

Tuning the Solver

- We observe that the number of iterations for outer CG is greatly reduced even if the inner preconditioner is solved in a sloppy way, e.g. iterating only 3-6 times.
- Our observation is supported by several theoretical works, say, [**G. Golub 1999**] and [**V. Simoncini 2003**].
- Thus the number of preconditioner solve is a parameter that can be tuned to achieve maximum speed up.

the **Polak-Ribière** (PR) formula and a non-trivial **reliable update** scheme is needed to insure the smooth convergence of such a mixed precision (double + half) solver with inexact preconditioner.

Fletcher-Reeves vs. Polak-Ribière

Algorithm 3 Preconditioned Conjugate Gradient $Ax = b$

$r_0 = b - Ax_0$

$p_0 = z_0 = M^{-1} r_0$

$k = 0$

while have not converged **do**

$\alpha_k = \langle r_k, z_k \rangle / \langle p_k, Ap_k \rangle$

$x_{k+1} = x_k + \alpha_k p_k$

$r_{k+1} = r_k - \alpha_k A p_k$ $\leftarrow$ matrix multiplication/dslash, comm.

$z_{k+1} = M^{-1} r_{k+1}$ $\leftarrow$ preconditioning solve, better no comm.

$\beta_k = \langle z_{k+1}, r_{k+1} \rangle / \langle z_k, r_k \rangle_{\mathbf{FR}}$ $\leftarrow$ $\beta_k = \langle z_{k+1}, r_{k+1} - r_k \rangle / \langle z_k, r_k \rangle_{\mathbf{PR}}$

$p_{k+1} = z_{k+1} + \beta_k p_k$

$k = k + 1$

end while

A Reliable Update Scheme

- It is well-known that during CG the accumulated residual differs significantly from the true residual.

$$r_{n+1} = r_n - \alpha_n A p_n \neq \hat{r}_n = b - A x_n$$

- With mixed precision solver the difference is even larger. Reliable updates are needed to correct the lower precision accumulated residual calculated with higher precision true residual without destroying the orthogonality of the Krylov space.

A Reliable Update Scheme

- A reliable update scheme with explicit restarts consists of several cycles: at the beginning of each cycle the true residual is calculated with higher precision before CG iterations are performed based on a new Krylov space.
- A *sophisticated* reliable update scheme [**H. Van der Vorst 2000**] is implemented to estimate the deviation with

$$d_k = d_{k-1} + \epsilon_l \left(\|A\| \cdot \|x_k\| + \|r_k\| \right)$$

and an higher precision correlation is made if

$$d_k > \sqrt{\epsilon_l} \|r_k\|$$

while the Krylov space is kept.

A Reliable Update Scheme

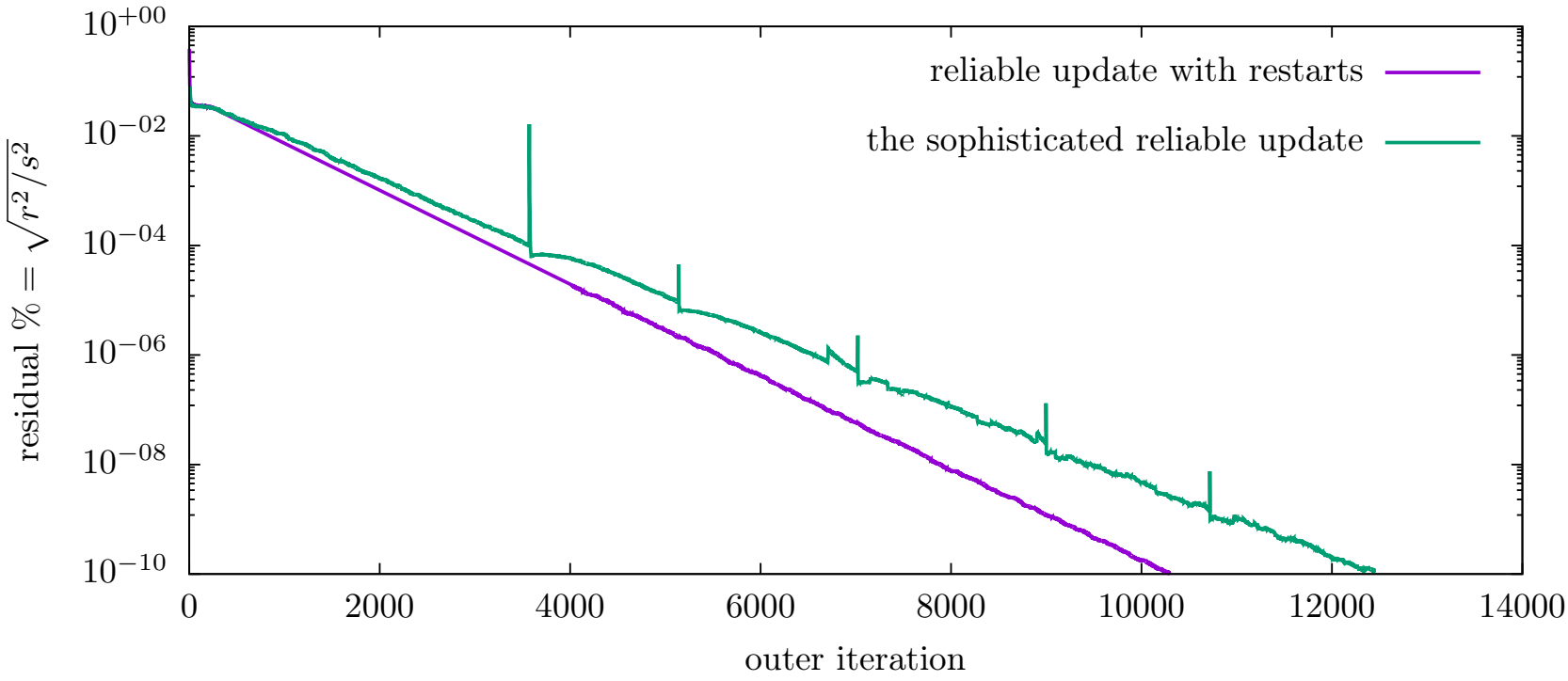


Figure 3: Comparison between reliable update using restarts with the more sophisticated reliable update.

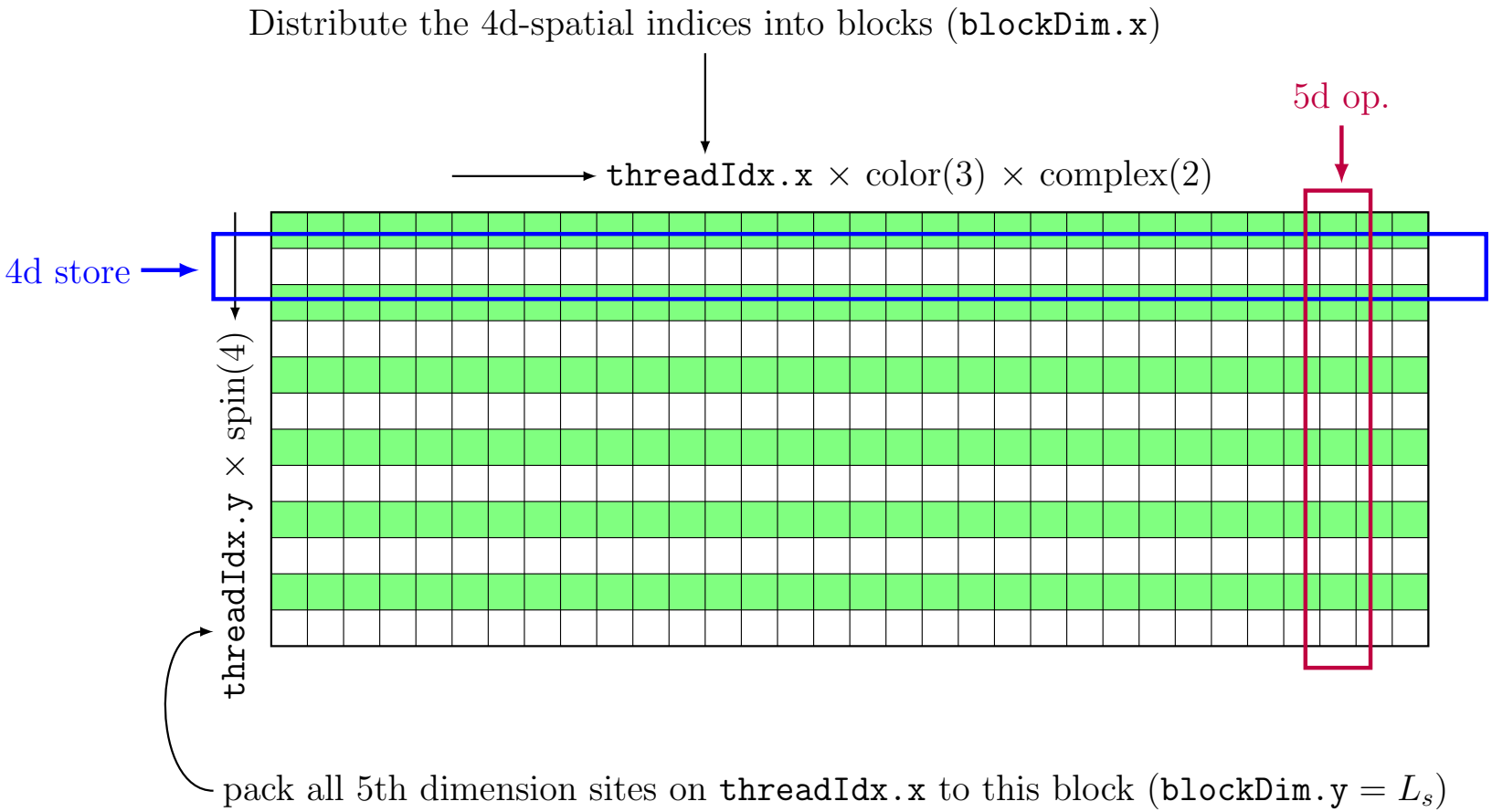
A brief summary: **communication** is **no longer** the sole bottleneck.

	supply	demand with MSPCG
compute flops (fp32)	15.7 tera-flops	3965×12 mega-ops
memory bandwidth	900 GB/sec	276.4×12 MB
network bandwidth	8.3 GB/sec	17.3 MB
compute/memory	17.4 ops/byte	14.3 ops/byte (0.8×supply)
network/memory	0.009	0.062 (7.0/12×supply)

Table 2: Compute flops, memory and network bandwidth demand versus supply for one iteration of MSPCG on Summit. Note that this is only an illustration that does not take into account the control flow structure containing the computing, memory loading/storing and communication.

Maximize GPU Performance on Summit

Exploit the programmable L1 cache (**share memory**) in CUDA to reduce memory traffic.

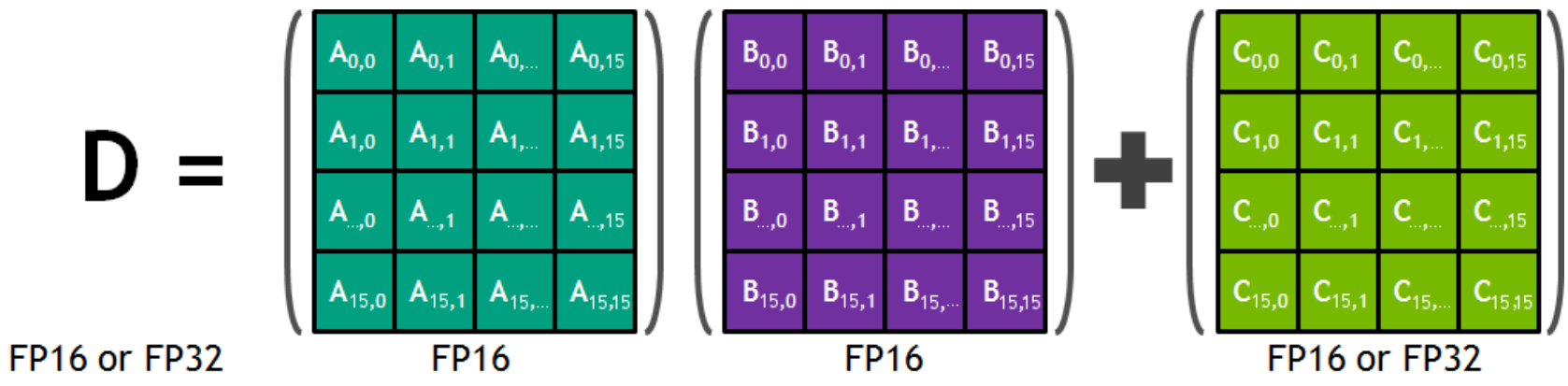


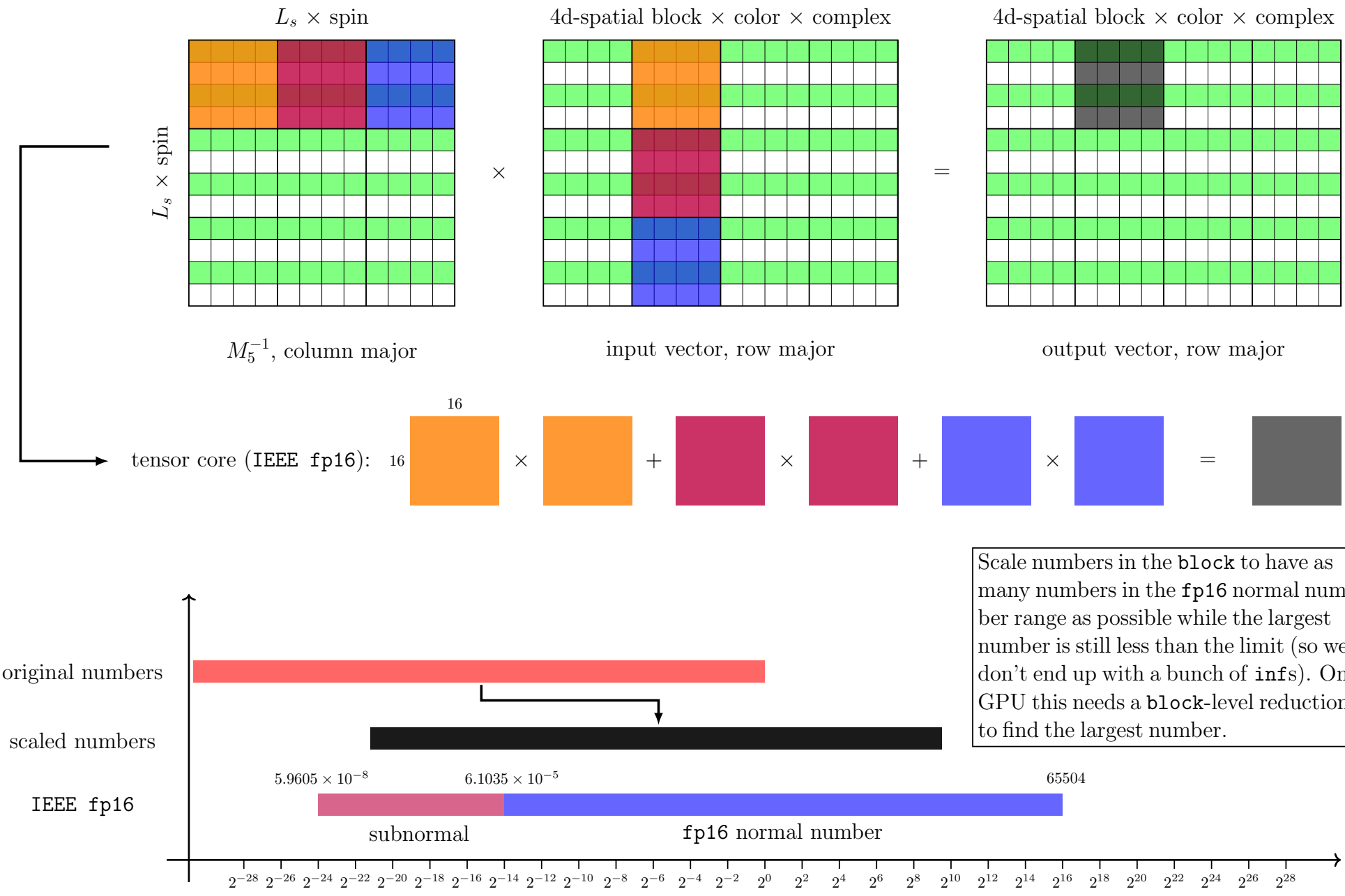
Maximize GPU Performance on Summit

Use **tensor cores** to perform the fifth dimension matrix multiplications, e.g. M_5^{-1}. Motivations:

- 125 tera-flops (compared to 15.7 with fp32) peak fp16 performance for dense matrix multiplication.
- Computation takes a large portion of the total time.
- Data is already loaded in L1 cache so almost zero latency and memory traffic.
- M_5^{-1} is 50% dense.

"Tensor cores provide a huge boost to convolutions and matrix operations."





Scale numbers in the `block` to have as many numbers in the `fp16` normal number range as possible while the largest number is still less than the limit (so we don't end up with a bunch of `infs`). On GPU this needs a `block`-level reduction to find the largest number.

nodes	inner iter.	(outer) iter.	r.u.	performance/node	time	speed up
256	–	42133	471	4.66	486.3	1.10×
	05	16903	195	1.56(01)/5.45(35)/37.29(53)	456.0	
	06	14860	173	1.56(01)/5.51(31)/37.60(58)	442.6	
	07	13787	161	1.56(01)/5.48(28)/37.49(60)	460.2	
	08	12922	151	1.56(01)/5.44(26)/37.55(63)	469.5	
512	–	42427	474	3.85	296.6	1.13×
	05	17625	203	1.26(01)/4.54(37)/36.21(52)	271.0	
	06	15425	179	1.27(01)/4.55(33)/36.26(57)	262.1	
	07	14409	168	1.26(01)/4.57(30)/36.39(60)	268.3	
	08	13597	159	1.27(01)/4.53(28)/36.35(63)	276.0	
1024	–	42482	474	2.93	195.2	1.22×
	05	18250	210	1.00(01)/3.68(34)/34.62(45)	183.3	
	06	15959	185	1.01(01)/3.68(35)/34.79(54)	159.7	
	07	14985	174	1.01(01)/3.68(32)/35.06(58)	163.6	
	08	14287	167	1.00(01)/3.69(29)/34.76(61)	169.1	

Table 3: Strong scaling of the MSPCG on Summit. Times are in seconds. r.u. = reliable updates. Perf. are in tera-flops. For MSPCG perf. is given in format of precise(double)/sloppy(half)/precondition(half) dslashes (percentage).

Wrap up for MSPCG

- 20 peta-flops of sustained solver performance on Summit with 1024 nodes.
- MSPCG provides a way to trade **local memory bandwidth and flops** for inter-node **communication**. Compared to current number it would bring more significant speed up if former becomes even more cheaper than the later, which seems to be the consensus trend.
- "The current trend in leadership computer systems is for individual compute nodes to attain denser and denser floating-point capability, resulting in systems with fewer, more floating point capable nodes. At the same time, interconnect speeds have grown more slowly, which can present challenges for strong-scaling problems such as gauge generation." – [**1904.09725**]

Wrap up for MSPCG

We do not yet know the specifications for the three exascale mathines (Aurora at ANL, Frontier at ORNL and EI Capitan at LLNL) but we can compare Titan at ORNL with Summit.

	Titan (2010)	Summit (2018)	factor
compute flops (fp32) [tera-flops]	4.0	94.2	23
memory bandwidth [TB/sec]	0.24	5.40	22
network bandwidth [GB/sec]	25	50	2

Table 4: For Titan numbers see https://www.olcf.ornl.gov/wp-content/uploads/2013/02/Titan_Architecture_1-JL.pdf. For Summit numbers see https://www.olcf.ornl.gov/for-users/system-user-guides/summit/summit-user-guide/.

Heavy Quark: s

$$\det D = \det(D^\dagger D)^{1/4}(D^\dagger D)^{1/4} = \int \Big[d\phi^\dagger\Big]\Big[d\phi\Big] \exp\Big[- \|(D^\dagger D)^{-1/4}\phi\|^2\Big]$$

Rational Hybrid Monte Carlo (RHMC)

The traditional approach: form rational approximation

$$x^{-p} \simeq \alpha_0 + \sum_k \frac{\alpha_k}{x + \beta_k}$$

and compute the single flavor determinant

$$\det \left[\frac{D(m_1)}{D(m_2)} \right] = \left\{ \det \left[\frac{D^\dagger D(m_1)}{D^\dagger D(m_2)} \right] \right\}^{1/2}$$

with multishift CG allows $D^\dagger D + \beta_k$ to be inverted for all k simultaneously: only one dslash is needed for each iteration for all the shifts.

Problems on Summit:

- Dslash is cheap on GPU: CPU force calculations for the shifts becomes the bottleneck.
- Mixed precision (double+half) multishift CG loses precision for the shifts: the reliable update/defect correction scheme in plain CG does not apply to the shifts. Consequence: takes long time to *refine* the solutions for the shifted inversions.

Exact One Flavor Algorithm (EOFA)

- **Idea**: Use Schur determinant identity applied to spin structure of D to factorize

$$\det\left[\frac{D(m_1)}{D(m_2)}\right] = \frac{1}{\det(H_1)} \cdot \frac{1}{\det(H_2)}$$

  with H_1 and H_2 hermitian and positive-definite [**1403.1683**].
- **Benefits**:
  - Minimal CPU force calculation overhead;
  - No multishift CG needed;
  - Easier to add Hasenbusch masses to tune HMC.

Exact One Flavor Algorithm (EOFA)

Dslash structure is similar to the plain MDWF one [**1706.05843**]:

$$\tilde{M}_{ee}(m_1, m_2, m_3) = M_{ee}(m_1) \pm \beta P_{\pm} |u_{\pm}\rangle\langle v_{\pm}|.$$

From Sherman-Morrison formula, with $M_5^{-1} = M_{5+}^{-1} P_+ + M_{5-}^{-1} P_-$,

$$\left[\tilde{M}_{ee}\right]^{-1} = \left[M_{ee} \pm \beta P_{\pm} |u_{\pm}\rangle\langle v_{\pm}|\right]^{-1} = M_{ee}^{-1} \mp P_{\pm} \frac{|x_{\pm}\rangle\langle y_{\pm}|}{1 \pm \langle v_{\pm}|x_{\pm}\rangle},$$

where

$$|x_{\pm}\rangle = \beta M_{5\pm}^{-1} |u_{\pm}\rangle, \ |y_{\pm}\rangle = M_{5\pm}^{-\dagger} |v_{\pm}\rangle.$$

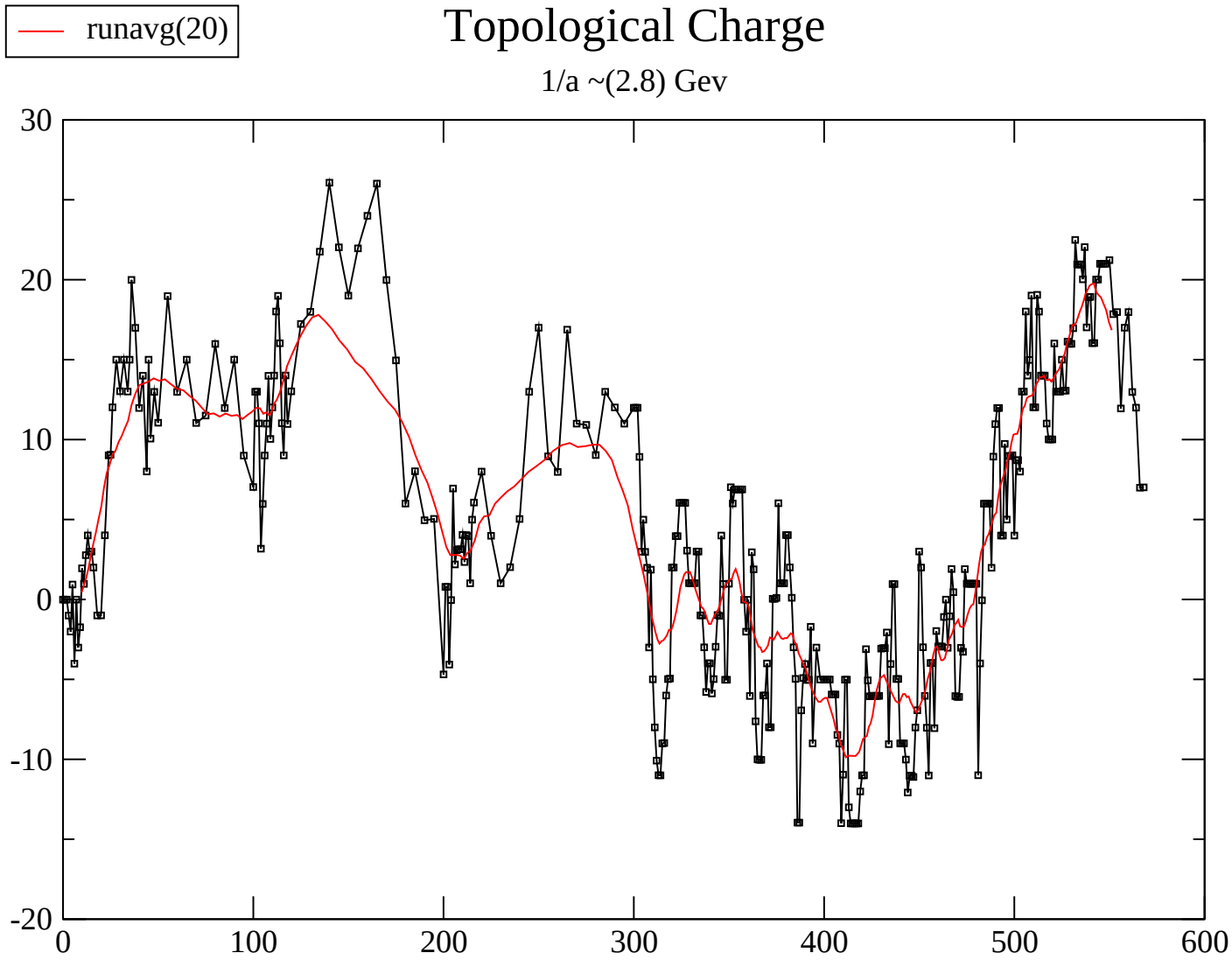
Exact One Flavor Algorithm (EOFA)

- The original dslash framework in QUDA is extended to include the additional term for EOFA.
- The programmable L1 cache (*shared memory*) on the GPU is fully utilized and the additional term adds negligible overhead.
- EOFA speeds up the heavy quark (single flavor) part of the evolution by a factor of 4.

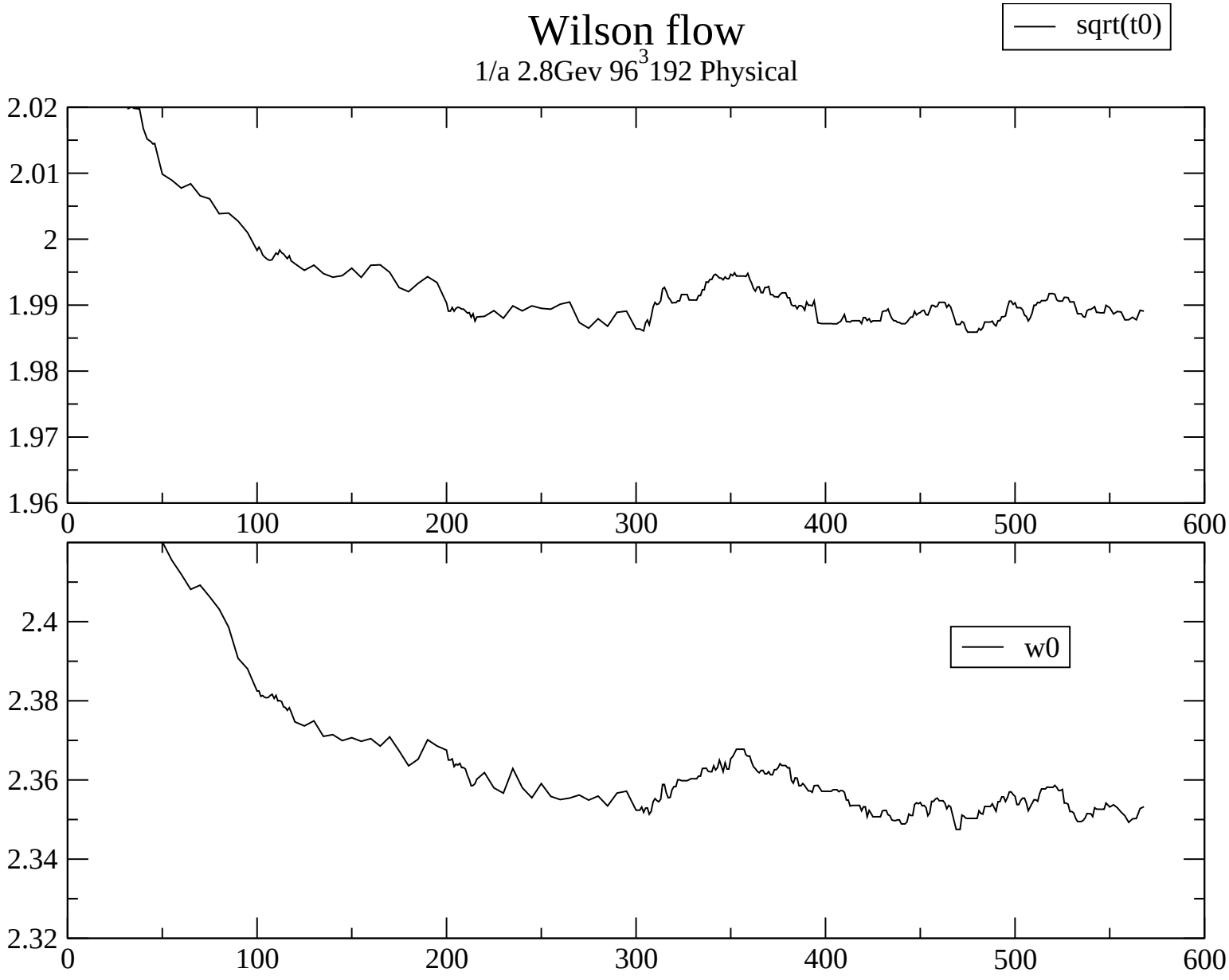
heavy quark (EOFA)	18.2%
light quark (quotient)	73.9%
gauge	7.2%
total time (512 nodes)	6328 seconds, 900 node-hour

Table 5: Achieved timing of one trajectory of the target lattice generation on Summit.

Current Progress



Current Progress



Beyond Summit

- Compared to the 2.77 GeV 96**I** lattice a 3.5 GeV lattice that includes the charm quark increases generation cost by a factor of

$$(3.5/2.77)^{4(\text{volume})+5(\text{c.s.d.})} \simeq 10.$$

- With MSPCG and potential algorithmic improvements in fighting against **critical slowing down**, this factor of 10 will likely be realized on the exascale machines.