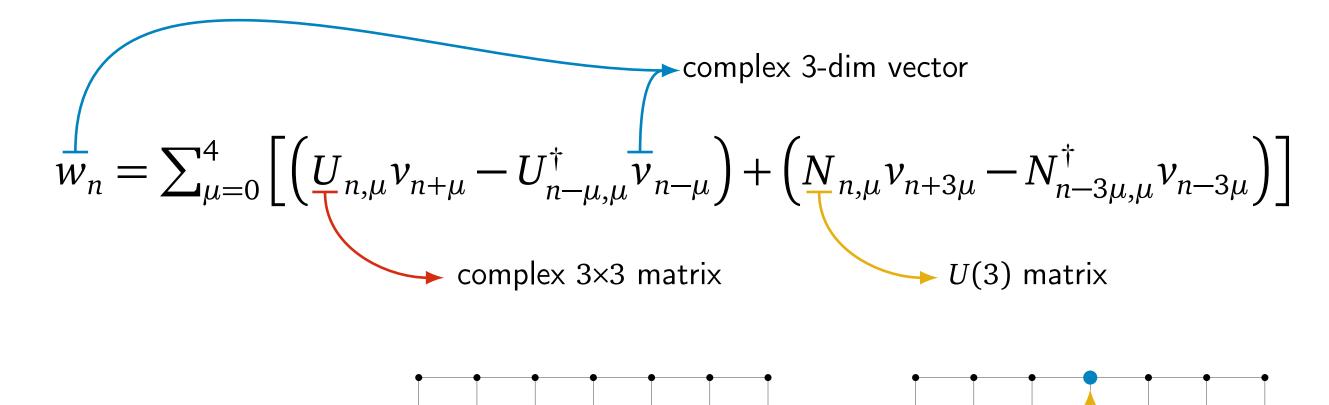


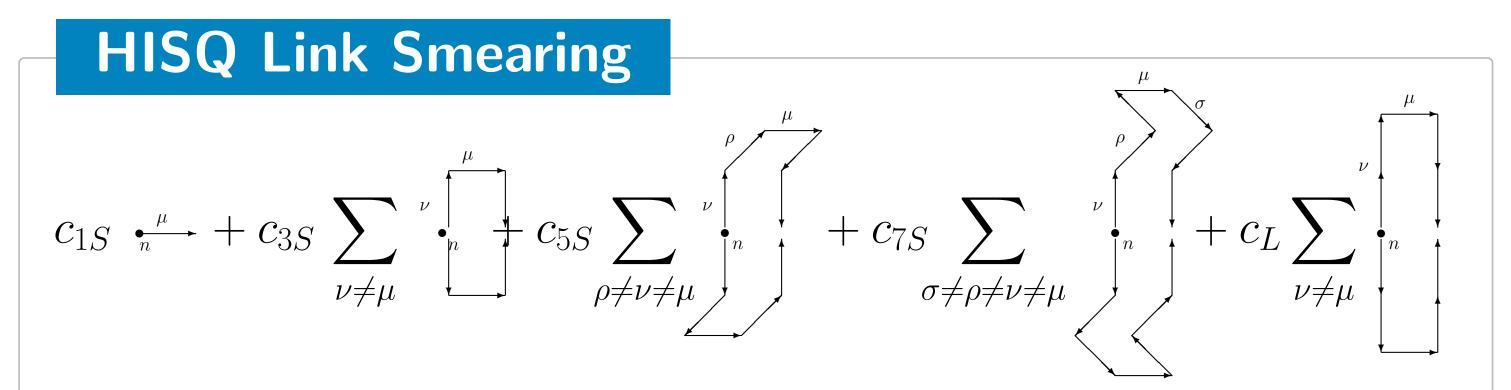
Performance of Staggered Fermion Kernels using Grid

Patrick Steinbrecher¹, Swagato Mukherjee¹ ¹Brookhaven National Laboratory - USA

HISQ Dslash Operator

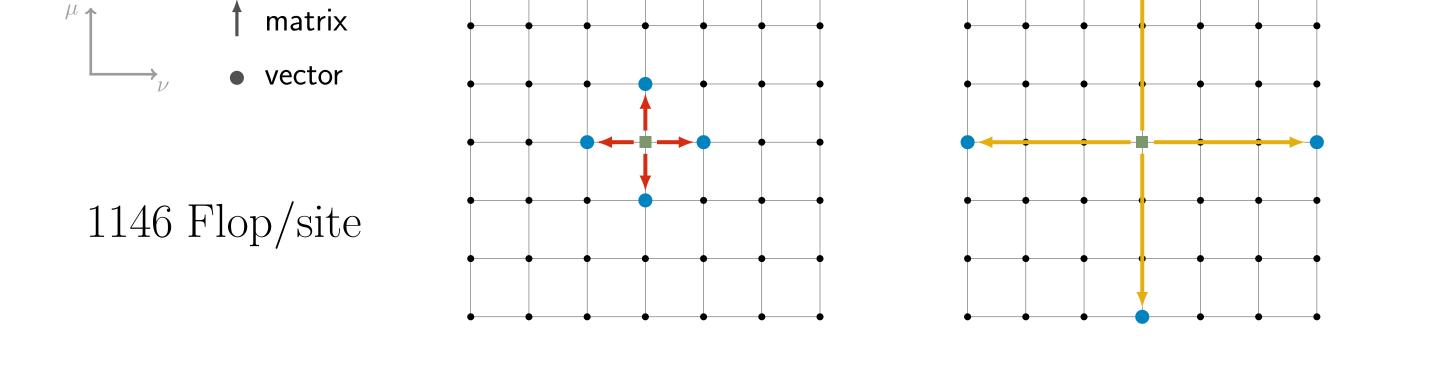
The key operation in many Lattice QCD simulations is the inversion of the fermion matrix. It requires a 4-dimensional stencil which calculates the product of a vector ν by a sparse matrix known as the Dslash operator and stems from a discretized 4-dimensional derivative.

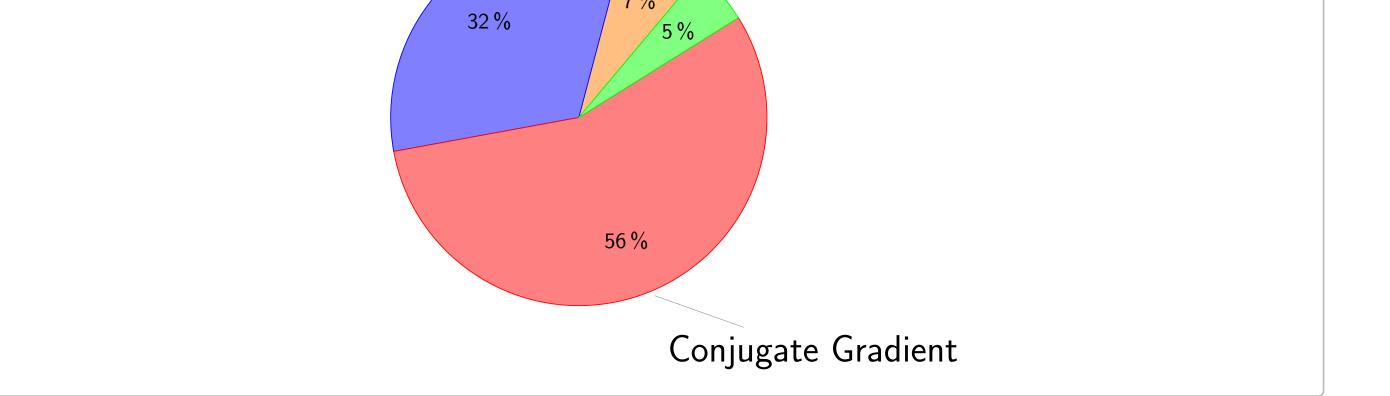


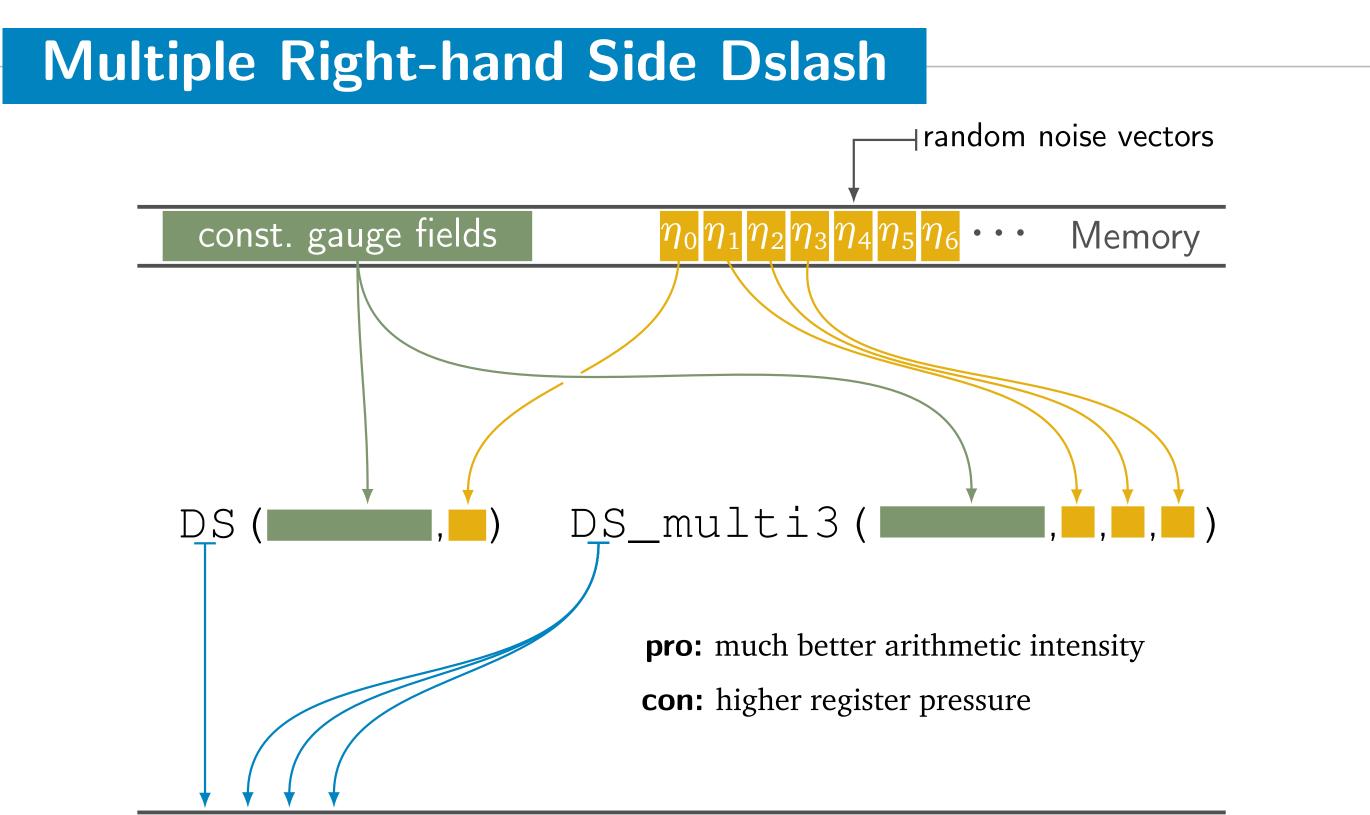


The HISQ action suppresses unphysical interactions of quarks by smoothing each link with a weighted sum of neighboring links referred to as smearing. It is mainly used in the force calculation for the Hybrid Monte Carlo algorithm and can take up to 40% of the total runtime. Gauge Force

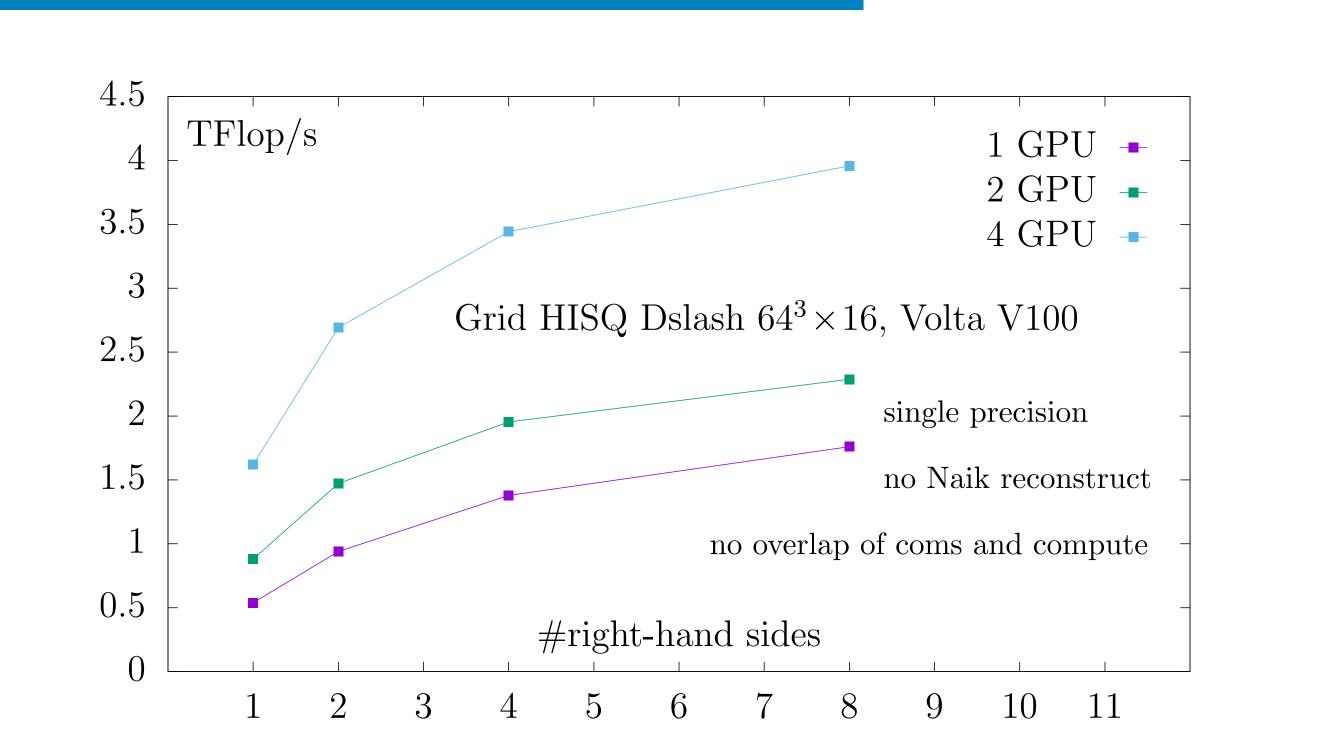
Fermion Force







Multi-GPU Dslash Performance

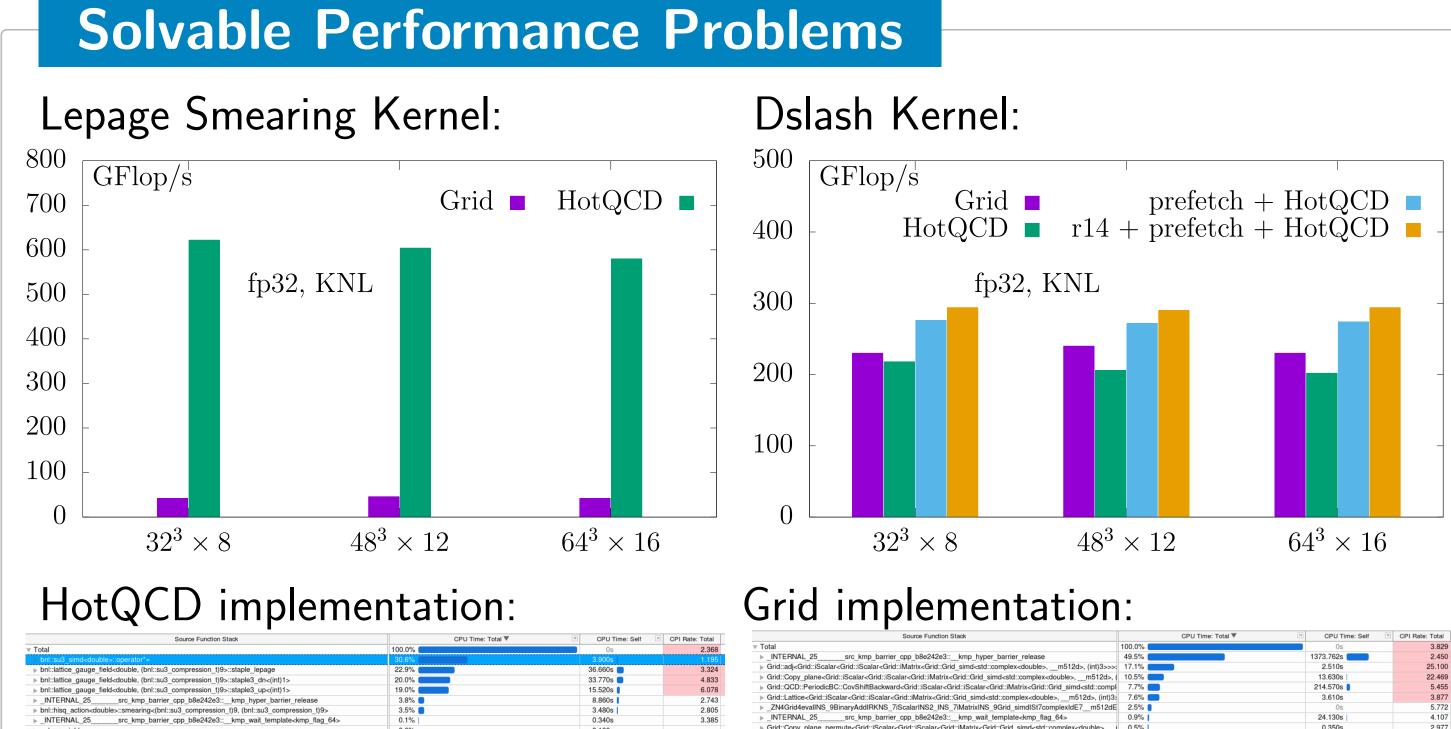




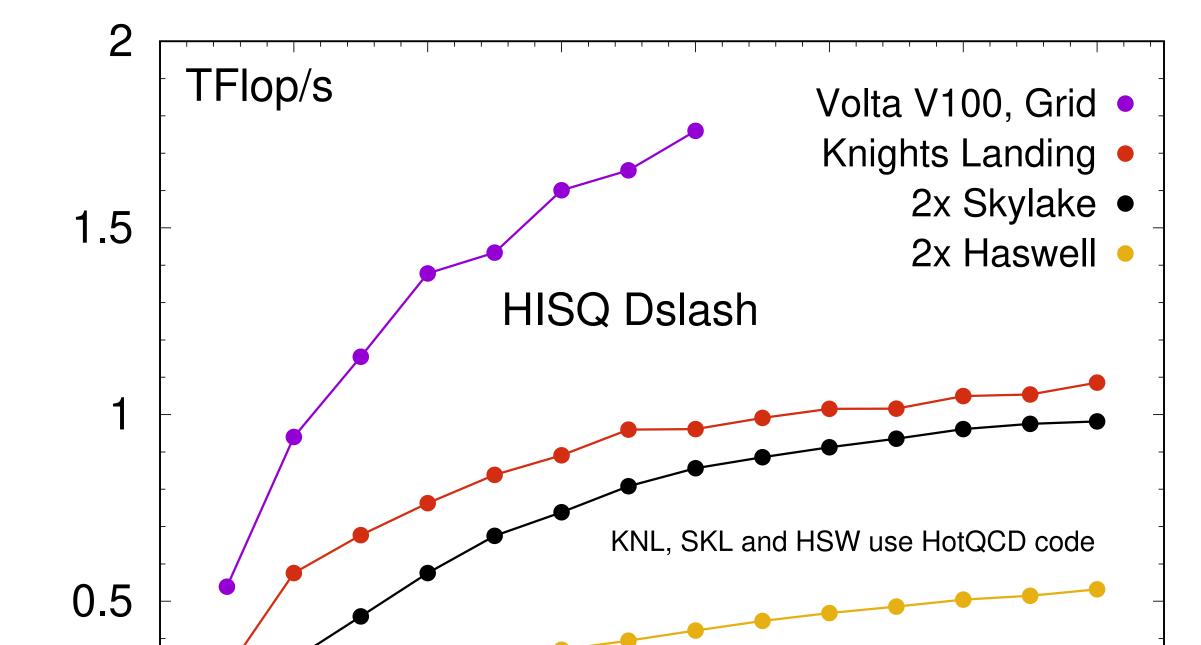
For many Lattice QCD applications, a large number of fermion matrix inversions are performed on a single gauge field. In order to exploit reuse of these gauge fields, we can apply the Dslash operation for multiple right-hand sides (rhs) at once. Increasing the number of rhs from one to four more than doubles the arithmetic intensity (Flop/byte) of the Dslash operation.

Grid [1] available at: github.com/paboyle/Grid

- architecture independent code
- high and low level interface written in C++
- support for multiple Lattice actions



Single-GPU Dslash Performance Comparison



kmp yield				0.0%				0.100s		Grid::Copy_plan
sched_yield				0.0%				0.060s		kmp_yield
kmp_barrier				0.0%				0s	2.000	std::vector <grid< p=""></grid<>
										OneLink_Lepag
										▶kmp_join_bar
										kmpc_for_sta
										sched_yield
										Grid::alignedAlle
										kmp_init_imp
						_				
0: + - r 🖻	3s 3.5s	4s 4.5	5s 5s		7s	7.5s 8s	8.5s	9s	✓ H/W Context ▼	0: •
cpu_108									Running	0_uqp
cpu_71									🗹 📥 CPU Time	S cpu_14
cpu_68									Spin and Overhead	<pre></pre>
									□ ▼CPU_CLK_UNHAL	≩ cpu_40 cpu_79
cpu_68 cpu_85									■ ♥ CPU_CLK_UNHAL ✓ CPU Time	T
cpu_68 cpu_85 cpu_17									CPU_CLK_UNHAL CPU Time CPU Time	cpu_79
cpu_68 cpu_85 cpu_17 cpu_92									■ ♥ CPU_CLK_UNHAL ✓ CPU Time	cpu_79 cpu_35
cpu_68 cpu_85 cpu_17									CPU_CLK_UNHAL CPU Time CPU Time	cpu_79 cpu_35 cpu_83

Grid::Copy_plane_permute <gri< p=""></gri<>	d::iScalar <grid::iscalar<grid::imatrix<grid::grid_simd<std::complex<double>,</grid::iscalar<grid::imatrix<grid::grid_simd<std::complex<double>	0.5%	0.350s	2.977
▶kmp_yield		0.5%	13.750s	12.770
std::vector <grid::lattice<grid::i< p=""></grid::lattice<grid::i<>	Scalar <grid::iscalar<grid::imatrix<grid::grid_simd<std::complex<double>,m5</grid::iscalar<grid::imatrix<grid::grid_simd<std::complex<double>	0.5%	13.050s	27.286
OneLink_LepageStaple\$omp\$p	arallel@146	0.4%	11.140s	5.880
kmp_join_barrier		0.3%	7.490s	10.346
kmpc_for_static_init_4		0.3%	7.240s	12.721
sched_yield		0.2%	6.650s	76.700
Grid::alignedAllocator <grid::isc< p=""></grid::isc<>	alar <grid::iscalar<grid::imatrix<grid::grid_simd<std::complex<double>,m512</grid::iscalar<grid::imatrix<grid::grid_simd<std::complex<double>	0.2%	5.800s	36.353
kmp_init_implicit_task		0.2%	4.440s	13.469
Q: + - ⊭ ⊭ 0	s			/ Context 👻
		205		Running CPU Time
0_uqp		20s 25s An Alexandro La Constantina de la Const Constantina de esta constantina de la C		Running CPU Time Spin and Overhead
ti cpu_0 cpu_14		205 255 The second state of the second state		Running CPU Time Spin and Overhead CPU_CLK_UNHAL
ta cpu_0 ta cpu_14 ta cpu_40		205 255 an Hill Ben constant with a Ben System 1997 and 1997 and Fill Ben Constant and the standard of the standard and the standard and 1997 and 1997 and 1997 and 1997 and 19 Fill Ben Constant and 1997 and And Adalah para Day 1997 and 1 Fill Standard and 1997 and		Running CPU Time Spin and Overhead CPU_CLK_UNHAL Time
cpu_0 g cpu_14 cpu_40 cpu_79				Running SPU Time Spin and Overhead SPU_CLK_UNHAL Time SPU Time
te cpu_0 cpu_14 ≩ cpu_40 cpu_79 cpu_35		205 255 and the loss of the devices of the loss of the and the loss of the devices of the loss of the devices and the loss of the loss of the devices of the devices and the loss of the loss of the devices of the devices and the loss of the loss of the devices of the devices and the loss of the loss of the devices of the devices and the loss of the loss of the devices of the devices and the loss of the loss of the devices of the devices and the loss of the loss of the devices of the devices and the devices of the loss of the devices of the devices and the devices of the loss of the devices of the devices and the devices of the loss of the devices of the devices and the devices of the loss of the devices of the devices of the devices and the devices of the loss of the devices of		Running SPU Time Spin and Overhead SPU_CLK_UNHAL Time SPU Time
cpu_0 cpu_14 ♀ cpu_40 cpu_79 cpu_35 cpu_83				Running CPU Time Spin and Overhead CPU_CLK_UNHAL Time

- Grid Dslash achieves optimal performance
- Grid Lepage smearing performance is limited by shortcomings of parallel transport approach
 - one parallel region for each link multiply in a staple
 - too many synchronization points
 - unnecessary loads and stores of intermediate results

-					fp32, sing	le node	
		# I	right-ha	and sic	les		
0	2	4	6	8	10	12	14
#	rhs	1	2	3	4	5	6
Flop	/byte	0.73	1.16	1.45	1.65	1.80	1.91

[1] P. Boyle et al., Grid: A next generation data parallel C++ QCD library. arXiv:1512.03487





www.usqcd.org

psteinbrecher@bnl.gov

Computing Properties of Hadrons, Nuclei and Nuclear Matter from Quantum Chromodynamics