Brookhaven
National Laboratory

Multiscale Lattice Gauge Theory Algorithms and Software for Exascale hardware

Peter Boyle

Brookhaven National Laboratory

Software: https://www.github.com/paboyle/Grid

- Lattice QCD and muon g-2
- Grid code for structured Lattice Gauge theory calculations, developed under ECP
  - Parallelization & portability: *covariant programming*
  - Performance

- Exascale algorithms and SciDAC-5
  - Multiple right-hand-side multigrid and GPU tensor units

- Why?

Fermilab 2023 experimental update

arXiv:2308.06230
Phys. Rev. Lett. 131, 161802 (2023)

Muon g-2 doubles down with latest measurement, explores uncharted territory in search of new physics

August 10, 2023

Share Tweet Email

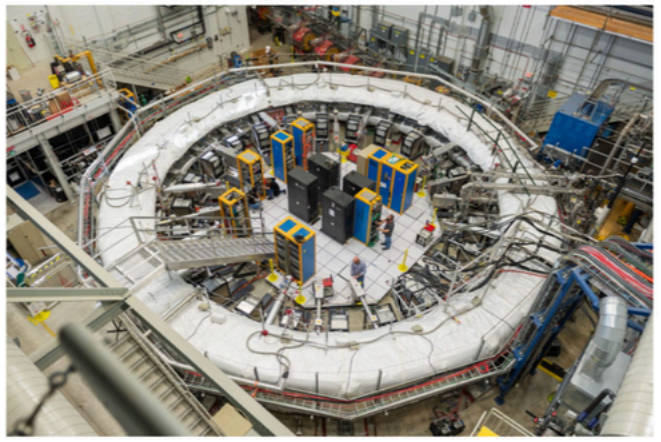
Media contact

- Tracy Marc, Fermilab, media@fnal.gov, 224-290-7803

Physicists now have a brand-new measurement of a property of the muon called the anomalous magnetic moment that improves the precision of their previous result by a factor of 2.

An international collaboration of scientists working on the Muon g-2 experiment at the U.S. Department of Energy's Fermi National Accelerator Laboratory announced the much-anticipated updated measurement on Aug. 10. This new value bolsters the first result they announced in April 2021 and sets up a showdown between theory and experiment over 20 years in the making.

"We're really probing new territory. We're determining the muon magnetic moment at a better precision than it has ever been seen before," said Brendan Casey, a senior scientist at Fermilab who has worked on the Muon g-2 experiment since 2008.



The announcement on Aug. 10, 2023, is the second result from the experiment at Fermilab, which is twice as precise than the first result announced on April 7, 2021. Photo: Ryan Postel, Fermilab

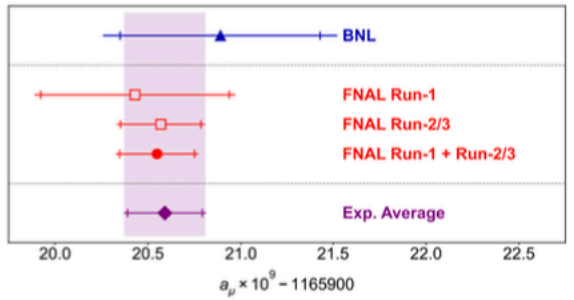


FIG. 3. Experimental values of a_μ from BNL E821 [8], our Run-1 result [1], this measurement, the combined Fermilab result, and the new experimental average. The inner tick marks indicate the statistical contribution to the total uncertainties.

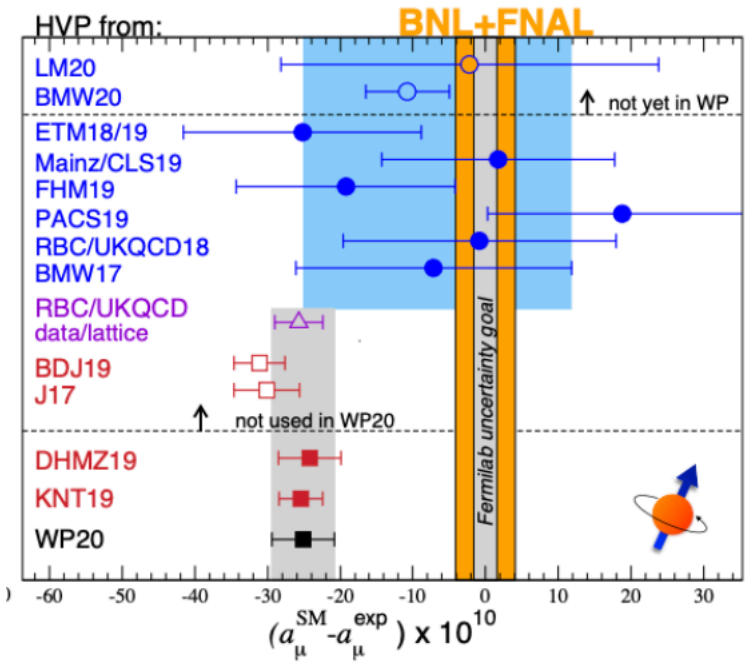
The experiment was "really firing on all cylinders" for the final three years of data-taking, which came to an end on July 9, 2023. That's when the collaboration shut off the muon beam, concluding the experiment after six years of data collection. They reached the goal of collecting a data set that is more than 21 times the size of Brookhaven's data set.

Physicists can calculate the effects of the known Standard Model "dance partners" on muon g-2 to incredible precision. The calculations consider the electromagnetic, weak nuclear and strong nuclear forces, including photons, electrons, quarks, gluons, neutrinos, W and Z bosons, and the Higgs boson. If the Standard Model is correct, this ultra-precise prediction should match the experimental measurement.

Calculating the Standard Model prediction for muon g-2 is very challenging. In 2020, the Muon g-2 Theory Initiative announced the best Standard Model prediction for muon g-2 available at that time. But a new experimental measurement of the data that feeds into the prediction and a new calculation based on a different theoretical approach — lattice gauge theory — are in tension with the 2020 calculation. Scientists of the Muon g-2 Theory Initiative aim to have a new, improved prediction available in the next couple of years that considers both theoretical approaches.

Muon g-2 has displayed a persistent 3-4 sigma tension with standard model 'predictions'

- But the prediction has made use of experimental e+ e- cross-section measurements and is not ab-initio
- More recent lattice results indicate reduced tension with SM



New ’20 BMW lattice result in tension with r-ratio, reduces tension to muon g-2

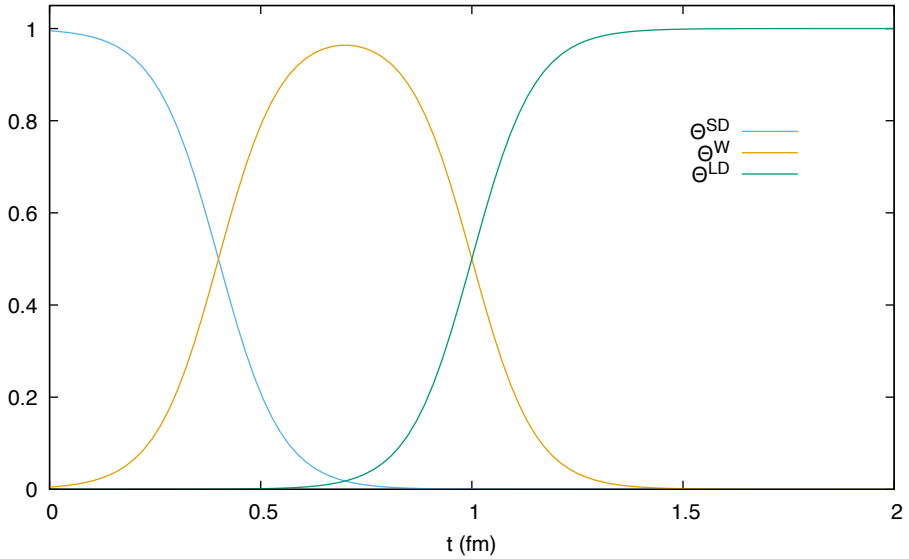
New ‘23 CMD-3 $e^+e^- \rightarrow \pi\pi$ cross section is in tension with Babar, KLOE
reduces tension with SM for g-2

Big need to address theoretical error to maximise impact of Fermilab Muon G-2 experiment

Lattice groups have broken the HVP calculation into short, middle and long-distance windows in Euclidean time

Makes it easier to compare and check results between collaborations and confront experiment with robust consensus results.

In future: compare to corresponding R-ratio energy-windows

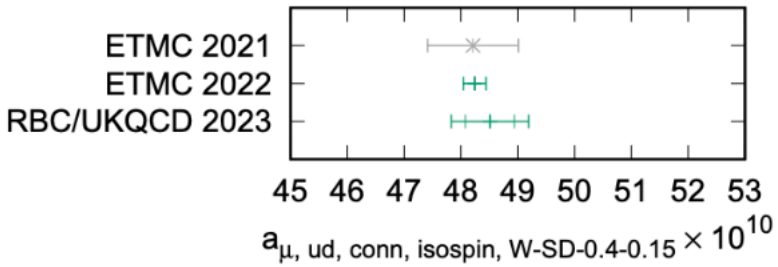


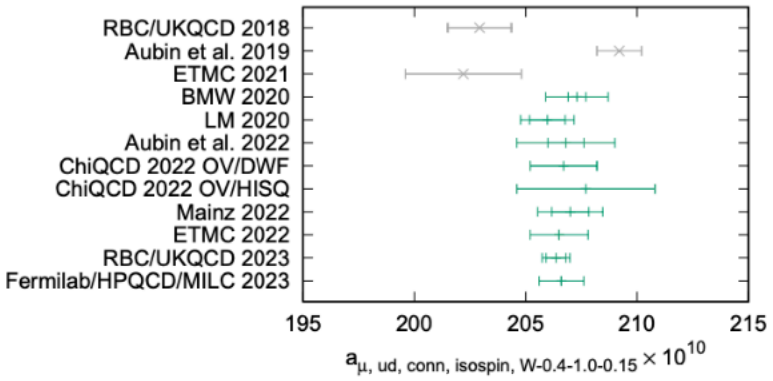
HVP short-distance and intermediate-distance window update (2301.08696)

An update of Euclidean windows of the hadronic vacuum polarization

T. Blum,[1] P. A. Boyle,[2,3] M. Bruno,[4,5] D. Giusti,[6] V. Gülpers,[3] R. C. Hill,[3] T. Izubuchi,[2,7] Y.-C. Jang,[8,9] L. Jin,[1,7] C. Jung,[2] A. Jüttner,[10,11] C. Kelly,[12] C. Lehner,[6,*] N. Matsumoto,[7] R. D. Mawhinney,[9] A. S. Meyer,[13,14] and J. T. Tsang[10,15]
(RBC and UKQCD Collaborations)

We compute the standard Euclidean window of the hadronic vacuum polarization using multiple independent blinded analyses. We improve the continuum and infinite-volume extrapolations of the dominant quark-connected light-quark isospin-symmetric contribution and address additional sub-leading systematic effects from sea-charm quarks and residual chiral-symmetry breaking from first principles. We find $a_\mu^{\rm W} = 235.56(65)(50) \times 10^{-10}$, which is in 3.8σ tension with the recently published dispersive result of $a_\mu^{\rm W} = 229.4(1.4) \times 10^{-10}$ [1] and in agreement with other recent lattice determinations. We also provide a result for the standard short-distance window. The results reported here are unchanged compared to our presentation at the Edinburgh workshop of the g-2 Theory Initiative in 2022 [2].





Intermediate window: confirming BMW
RBC-UKQCD presently has the smallest error.
Multiple groups confirming each other

RBC-UKQCD collaboration has published short and middle-distance windows contributions to HVP
Emerging consensus behind BMW on middle-distance part of HVP with meaningful precision

Working on long distance windows, complete HVP determination and reducing finite volume effects

- How?

Lattice QCD involves numerical evaluation of the Feynman path integral

$$\int d\pi \int d\phi \int dU \quad e^{-\frac{\pi^2}{2}} e^{-S_G[U]} e^{-\phi^*(M^\dagger M)^{-1}\phi}$$

- Outer Metropolis Monte Carlo algorithm
  - Draw gaussian momenta and pseudofermion as gaussian $\eta = M^{-1}\phi$
  - Metropolis acceptance step
  - Proposal includes inner molecular dynamics at constant Hamiltonian:

$$H = \frac{\pi^2}{2} + S_G[U] + \phi^*(M^\dagger M)^{-1}\phi$$

- Multiple petaflops years calculations, integrate 10^{10} degrees of freedom
- Matrix “M” represents the Dirac equation on a background quantum fluctuation of the gluon field
- Inversions of “spinor” fields typically use Conjugate Gradients (or multi-level CG).
  - Performed at each step of MCMC
- Strong scaled 4D Dirac PDE solver performance on structured grid is critical

Grid QCD code

Design considerations

- Performance portable across multi and many core CPU's

$$\mathrm{SIMD} \otimes \mathrm{OpenMP} \otimes \mathrm{MPI}$$

- Performance portable to GPU's

$$\mathrm{SIMT} \otimes \mathrm{offload} \otimes \mathrm{MPI}$$

- N-dimensional cartesian arrays
- Multiple grids
- Data parallel C++ layer : Connection Machine inspired

Accelerator.h: Lean internal API to offload: similar ideas to RAJA and Kokkos

- Device lambda capture
- O(1) overhead true LRU data cache on device

Data Layout changes with vector length of architecture: covariant programming

Native interfaces in: "Grid Python Toolkit" (Lehner), "Hadrons" (Portelli)
Used as library by MILC, CPS, Qlat

Internal interface to parallelism gives cross platform portability
(high level code does not see this – use the data parallel API)

```
// HIP specific
accelerator_inline int acceleratorSIMTlane(int Nsimd) {
 return hipThreadIdx_z;
}
#define accelerator_for2d( iter1, num1, iter2, num2, nsimd, ... )     \
  {                                                                     \
    typedef uint64_t Iterator;                                          \
    auto lambda = [=] accelerator                                       \
      (Iterator iter1,Iterator iter2,Iterator lane ) mutable {          \
      { __VA_ARGS__;}                                                   \
    };                                                                  \
    int nt=acceleratorThreads();                                        \
    dim3 hip_threads(nt,1,nsimd);                                       \
    dim3 hip_blocks ((num1+nt-1)/nt,num2,1);                            \
    hipLaunchKernelGGL(LambdaApply,hip_blocks,hip_threads,              \
                       0,0,                                             \
                       num1,num2,nsimd,lambda);                         \
  }
```

```
// OpenMP specific
#define accelerator
#define accelerator_inline strong_inline

#define accelerator_for(iterator,num,nsimd, ... )  \
 thread_for(iterator, num, { __VA_ARGS__ });

#define accelerator_for(iterator,num,nsimd, ... ) \
 thread_for(iterator, num, { __VA_ARGS__ });

#define accelerator_barrier(dummy)

#define accelerator_for2d(iter1, num1, iter2, num2, nsimd, ... )\
 thread_for2d(iter1,num1,iter2,num2,{ __VA_ARGS__ });
```

Also: CUDA and SYCL implementations of internal interface

Future: OpenMP target device, and C++ std parallelism

Covariant programming : capturing the variation between SIMD and SIMT in a single code

The struct-of-array (SoA) portability problem:

- Scalar code: CPU needs struct memory accesses struct calculation
- SIMD vectorisation: CPU needs SoA memory accesses and SoA calculation
- SIMT coalesced reading: GPU needs SoA memory accesses struct calculation
- GPU data structures in memory and data structures in thread local calculations *differ*

Model	Memory	Thread
Scalar	Complex Spinor[4][3]	Complex Spinor[4][3]
SIMD	Complex Spinor[4][3][N]	Complex Spinor[4][3][N]
SIMT	Complex Spinor[4][3][N]	Complex Spinor[4][3]
Hybrid?	Complex Spinor[4][3][Nm][Nt]	Complex Spinor[4][3][Nt]

How to program portably?

- Use operator() to transform memory layout to per-thread layout.
- Two ways to access for read
- operator[] returns whole vector
  - operator() returns SIMD lane threadIdx.y in GPU code
  - operator() is a trivial identity map in CPU code
- Use coalescedWrite to insert thread data in lane threadIdx.y of memory layout.

- Where?

- GPU optimization for HIP, SYCL, CUDA and OpenMP

- CPU vector optimization for SSE, AVX, AVX2, AVX512; ARM NEON, ARM SVE

- Significant usage on systems with Nvidia, AMD, and Intel GPUs
  - USA: Frontier, Summit, Perlmutter, Polaris, Aurora
  - Europe & UK: Booster, Lumi-G, Leonardo, Tursa
  - Japan: Fugaku

- Important that GPU systems have at least 200Gbit/s network card for *each* GPU currently to give scalability

- Good performance cross platform:

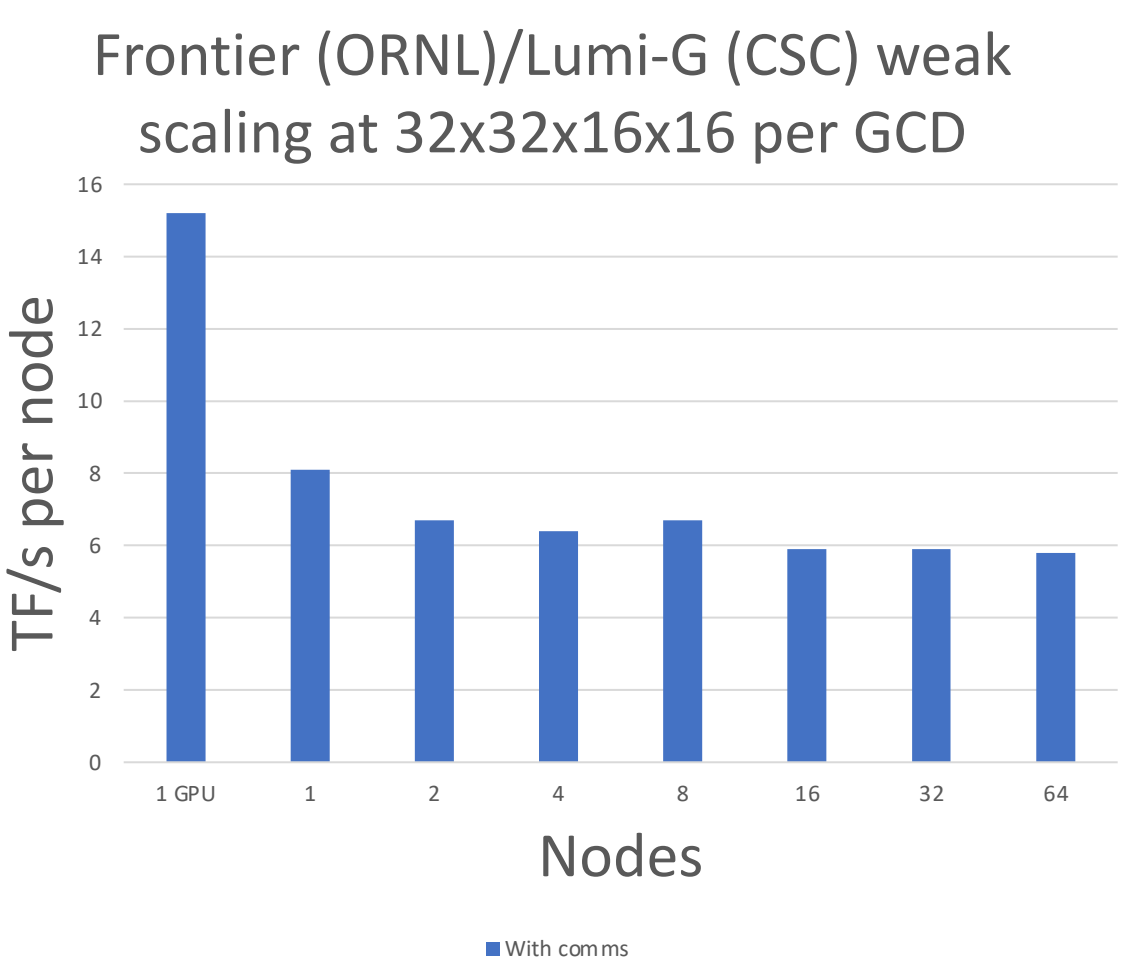
  - 10+TF/s per node on quad A100, quad MI250, and four/six PVC nodes

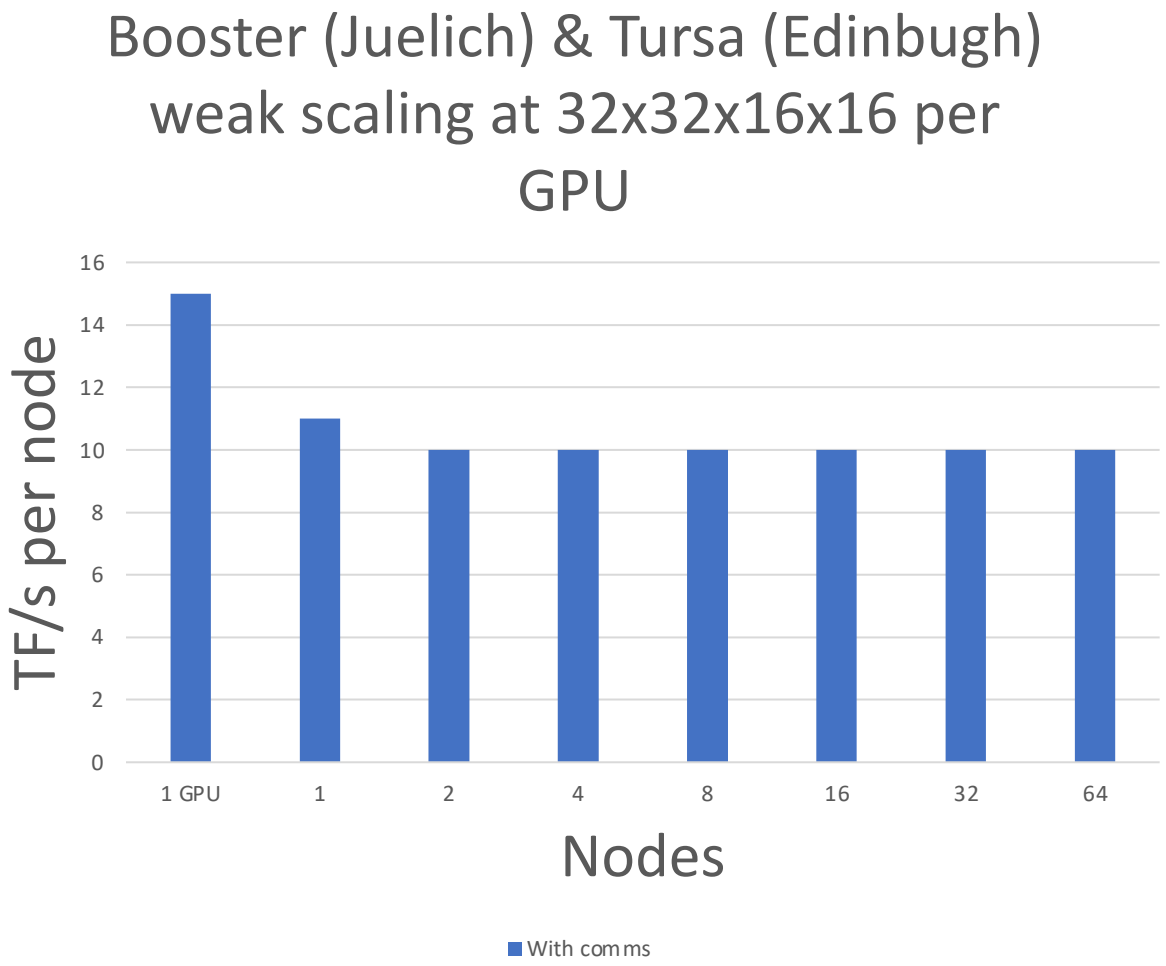
  - 1 – 1.3 TF/s per node on Intel Sapphire Rapids and AMD Genoa two socket CPU nodes

  - Runs well on Fugaku / ARM SVE

- How fast?

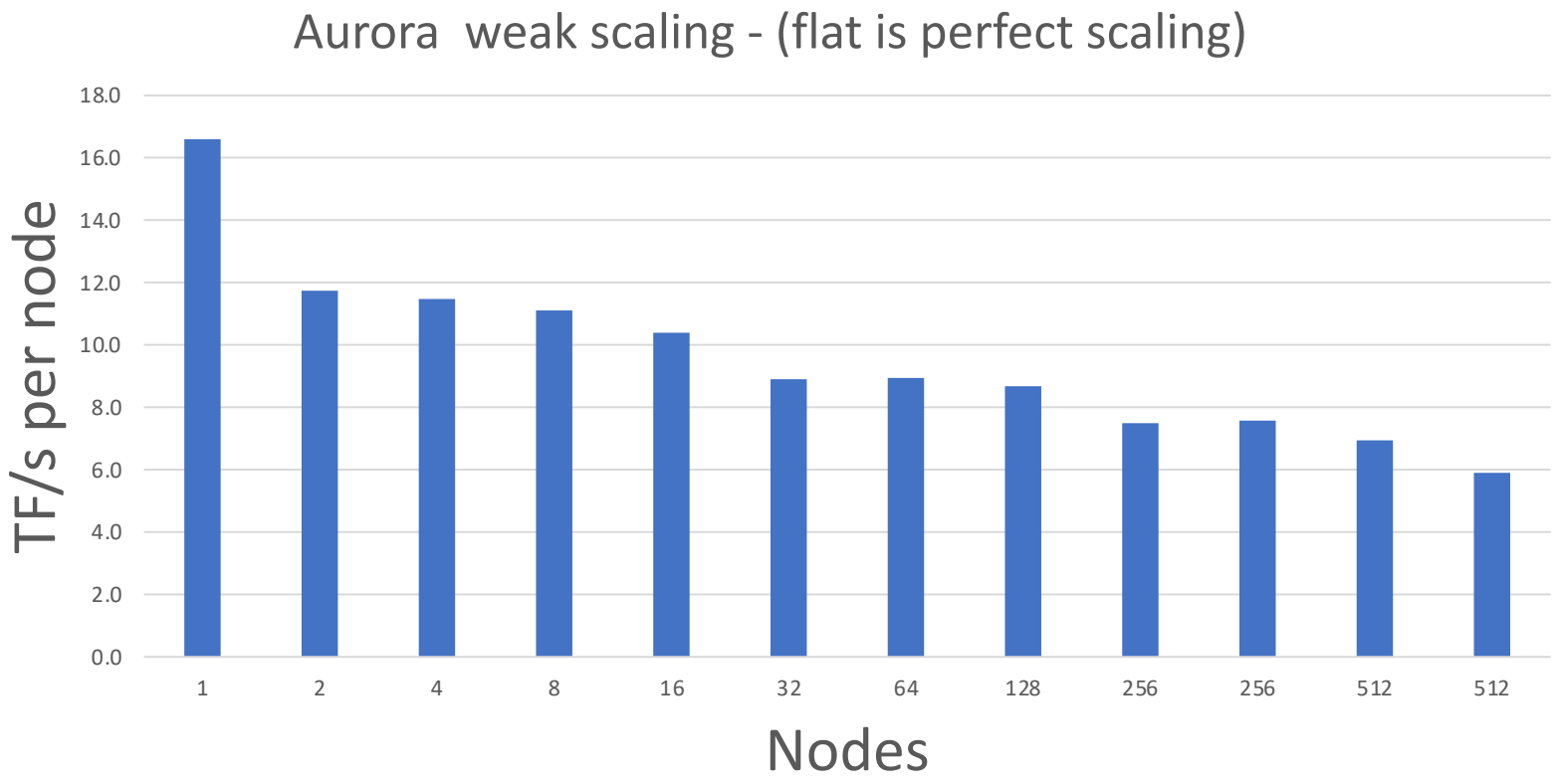
Weak scaling : NB perfect line scaling is displayed as flat as plot performance **PER** node





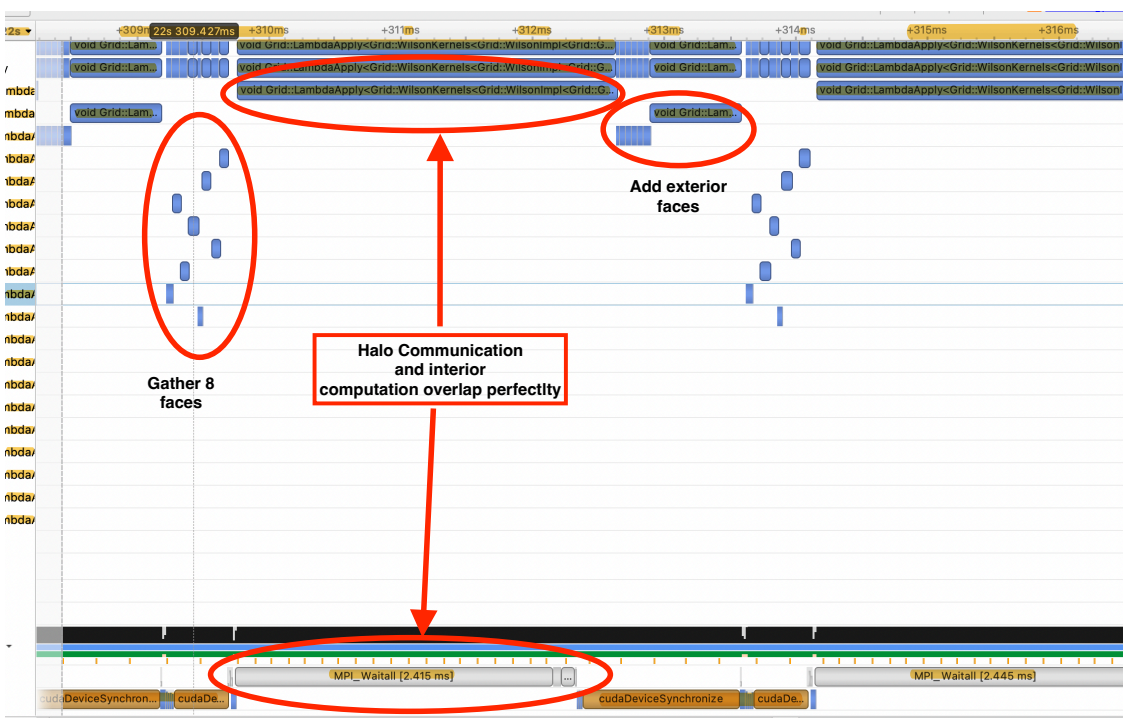
HPE Cray EX
AMD MI250X x 4 GPUs nodes
4x Slingshot: 1600Gbit/s per node

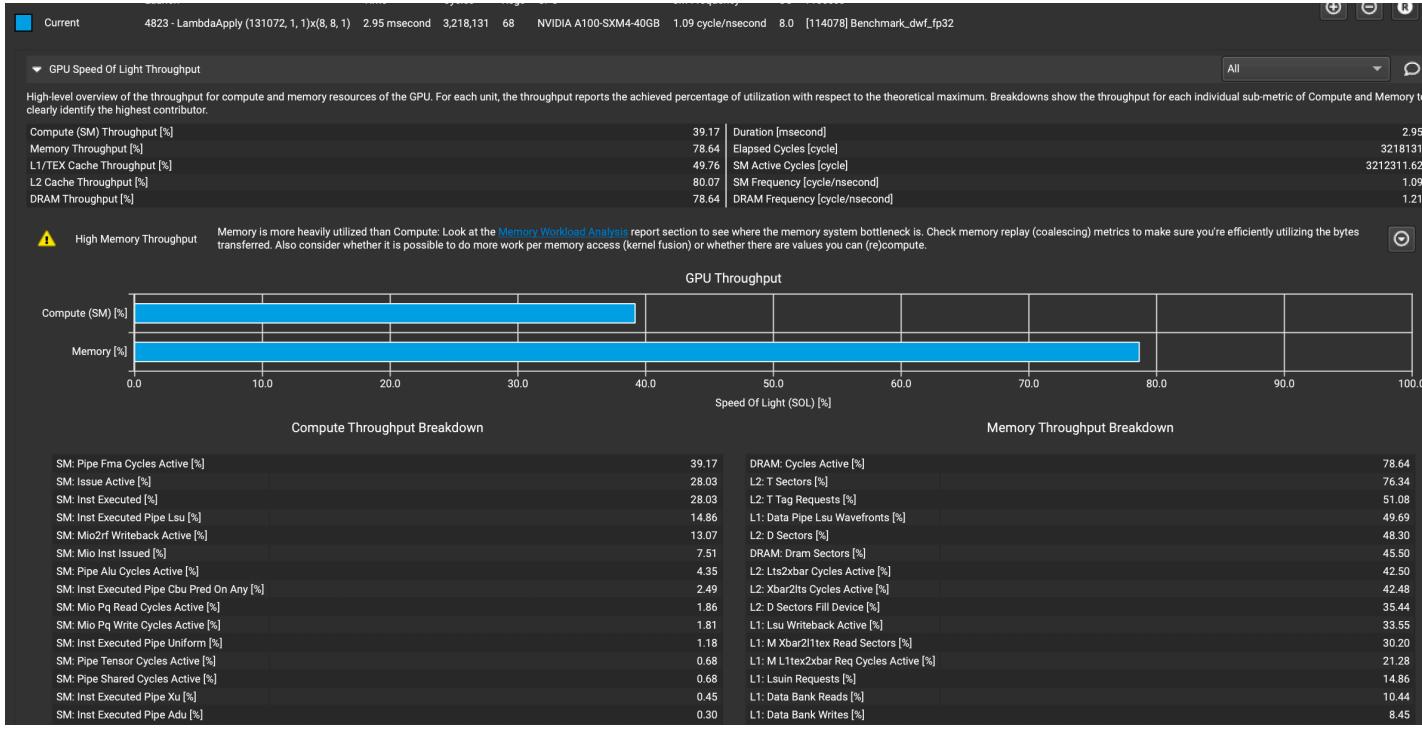
ATOS Sequana
Nvidia A100 x 4 GPU nodes
4x HDR IB: 1600Gbit/s per node



Intel PVC GPUs on Aurora at ANL, 3200Gbit/s per node

(ANL caveat: this is based on early software at ANL and subject to further improvement)





Code has been profiled in detail: kernels execute back-to-back.

Multi-rail infiniband & slingshot exceeds 180GB/s and 90% of bidirectional four rail IB wirespeed *concurrent* with computation
i.e. 1.6 Tbit/s per node

On Booster, Tursa, Lumi-G, Leonardo, Frontier the communications and computation overlap perfectly

Each of 26 kernels in Dirac matrix are reported by Nvidia at 80% of peak memory speed

- What next?

SciDAC-5 : Multiscale acceleration

Adapt our algorithms to the new problems enabled by the Exascale:

more length scales => critical slowing down

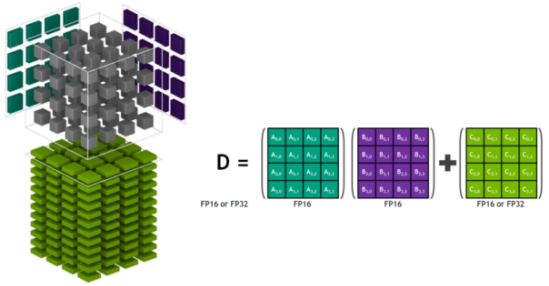
Multigrid Dirac solvers:

- Learn near null space of Dirac matrix: non-trivial in gauge theory
  - Break these vectors into local wavelet basis chunks
  - Determine a representation of Dirac operator within this critical subspace
- Use as a near-null space multigrid preconditioner

New idea:

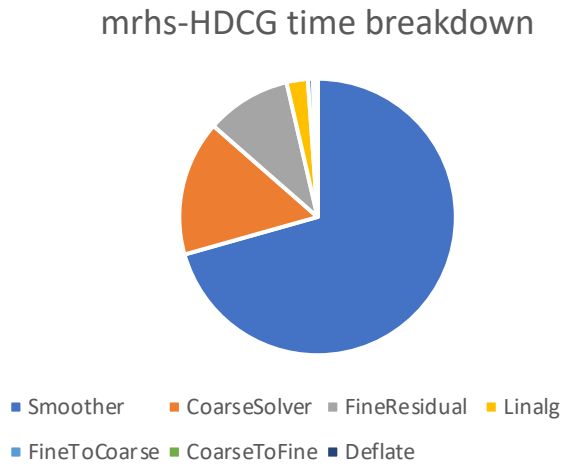
solve multiple right-hand sides simultaneously and use GPU Tensor hardware.

3-5 TF/s per GPU in double precision ZGEMM and 30x faster !

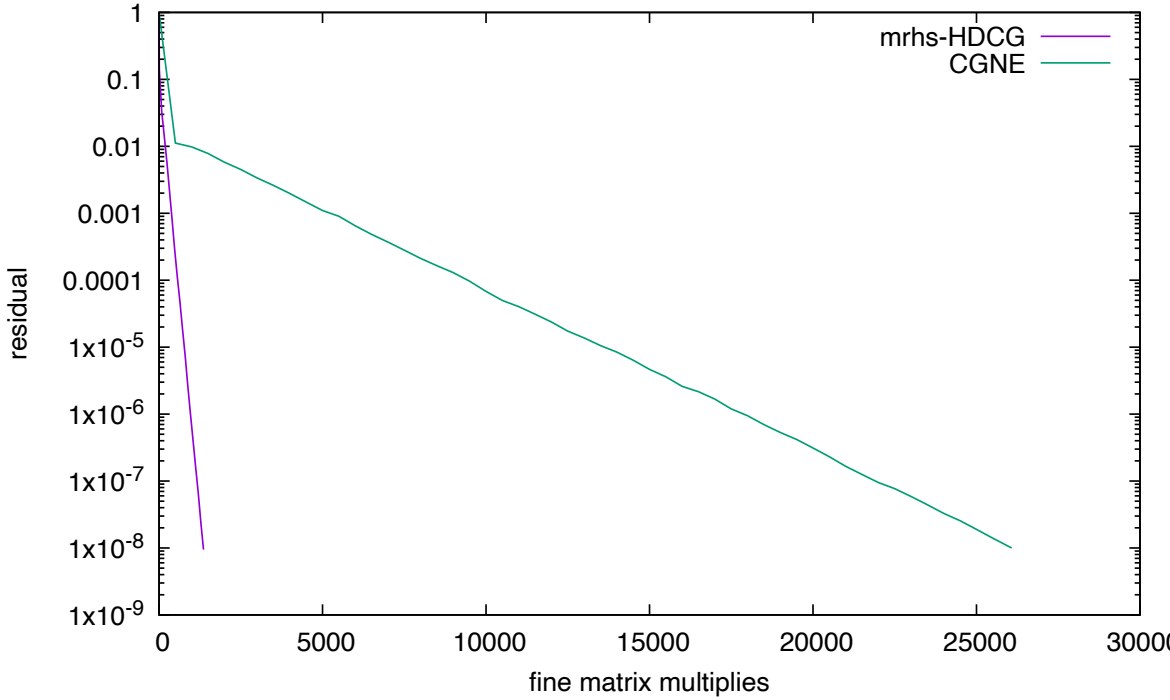


SciDAC-5 multigrid status: physical quark masses, 18 nodes, Frontier

- Red-black preconditioned conjugate gradient solves ***single*** RHS in 770s
- mrhs-HDCG solves ***twelve*** RHS in 1017s
  - 9.1x speed up wall clock
  - batched BLAS ZGEMM on GPU on red named routines: 30x speedup!
  - 17x reduction in fine matrix multiplies (26000 vs. 1500)
- Algorithm required for RBC-UKQCD large volume muon g-2
  - Scheduled innovation – and more gain anticipated!



Total	1017s
FineSmoother	710s
CoarseSolver	159s
FineResidual	100s
FineLinalg	25s
FineToCoarse	6s
CoarseToFine	5s
Deflate	0.3s



Summary

- After significant effort, Grid software is portable AND performance portable
- Problem sizes enabled by Exascale require new algorithms

- Order of magnitude gain with bespoke algorithm aimed at muon g-2

- Multigrid: learned compression of QCD Nc=3 into learned wavelet basis
  - Use of GPU tensor cores to operate on that basis
  - Accelerates convergence as a preconditioner; multiRHS turns problem into fast GEMM operations
  - Batched BLAS routines VERY helpful
  - Use of ML hardware can be a blend of old and new approaches:
    - retains the underlying mathematics of physics that took centuries to understand

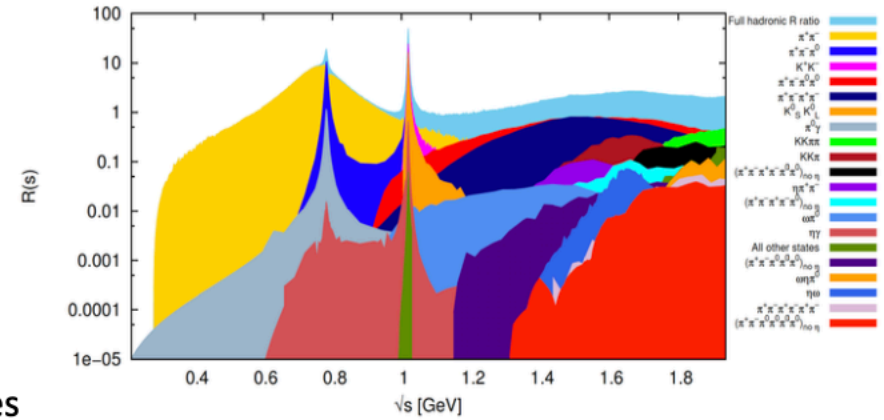
BACKUP

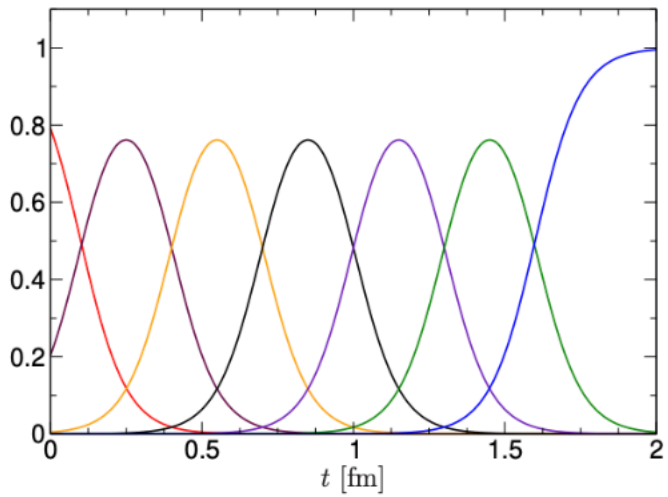
- Is there a theoretical way forward?

  - Personal opinion:
    - Difficult to resolve new, discrepant experimental inputs without growth in R-ratio error.

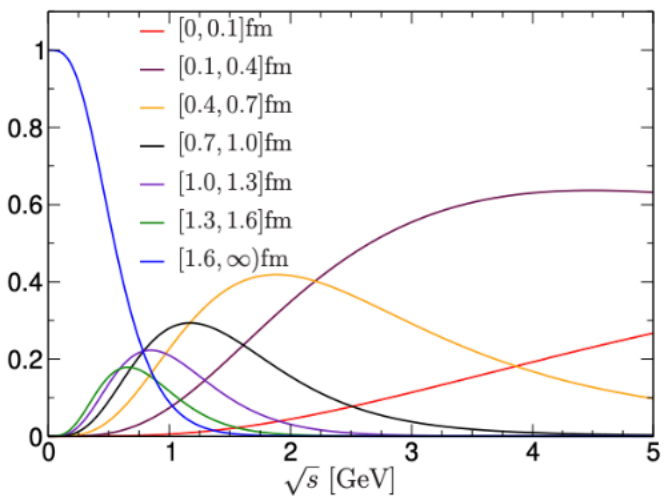
  - Lattice results are growing increasingly precise and may soon become the reference SM prediction

  - Comparing windows, and reconstructing by intermediate states may identify the discrepant modes in more detail





Euclidean space window
For vector correlation function



Minkowski CofM energy weighting

Physics Letters B,
Volume 833,
2022,
137313,

Preprints

https://arxiv.org/abs/2401.16620

https://arxiv.org/abs/2203.17119

https://arxiv.org/abs/2203.06777

Software:

https://github.com/paboyle/Grid

https://github.com/aportelli/Hadrons

https://github.com/lehner/gpt