Eur. Phys. J. A (2019) **55**: 199
DOI 10.1140/epja/i2019-12919-7

THE EUROPEAN PHYSICAL JOURNAL A

Regular Article – Theoretical Physics

Status and future perspectives for lattice gauge theory calculations to the exascale and beyond*

Bálint Joó[1,a], Chulwoo Jung[2,b], Norman H. Christ[3], William Detmold[4], Robert G. Edwards[1], Martin Savage[5], and Phiala Shanahan[4]

[1] Theory Center, Thomas Jefferson National Accelerator Facility, Newport News, VA 23606, USA
[2] Physics Department, Brookhaven National Laboratory, Upton, NY 11973, USA
[3] Department of Physics, Columbia University, New York, NY 10027, USA
[4] Department of Physics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
[5] Institute for Nuclear Theory, University of Washington, Seattle, WA 98195-1550, USA

Received: 20 August 2019 / Revised: 8 October 2019
Published online: 14 November 2019

Communicated by U.-G. Meißner

Abstract. In this and a set of companion white papers, the USQCD Collaboration lays out a program of science and computing for lattice gauge theory. These white papers describe how calculation using Lattice QCD (and other gauge theories) can aid the interpretation of ongoing and upcoming experiments in particle and nuclear physics, as well as inspire new ones.

Executive summary

In 2018, the USQCD Collaboration Executive Committee organized several subcommittees to recognize future opportunities and formulate possible goals for lattice field theory calculations in several physics areas. The conclusions of these studies, along with community input, are presented in seven white papers [1–7].

Numerical studies of lattice gauge theories in general —and of lattice quantum chromodynamics (Lattice QCD) in particular— have been a driver of high performance computing (HPC) for nearly 40 years. Lattice QCD practitioners have innovated in the algorithmic, hardware, and performance space with an impact that has reached substantially beyond the field. Examples include influence on supercomputer architectures such as the IBM Blue Gene line of systems, development of algorithms that are used in other domains, such as hybrid Monte Carlo, and early adoption of new technologies such as graphics processing units.

The power of computers continues to increase. At the same time, the adoption of novel algorithmic approaches such as better preconditioners, improved linear and eigensolvers, more efficient molecular dynamics time integrators, and more powerful boost the available statistics. Thus, the scientific output of Lattice QCD calculations has far exceeded the growth one would expect purely from hardware speed-up alone.

These advances have been supported in two main ways, under the umbrella of the USQCD Collaboration. One consists of software development through successive generations of the United States Department of Energy (DOE) Scientific Discovery through Advanced Computing (SciDAC) program, and now also by the DOE Exascale Computing Project (ECP). The other is a series of infrastructure projects, known as LQCD, supported by the DOE Office of High Energy Physics and the Office of Nuclear Physics. These efforts have led to large-scale cooperation with colleagues at large-scale and leadership-class computing facilities (LCFs), as well as industrial partners. U.S. Lattice QCD researchers has invested in and fostered a community with strong HPC expertise, which allows them to be energetic participants in the drive towards exascale computing, with the USQCD Collaboration forming a very active component within the ECP.

Lattice QCD has been an early adopter of many disruptive technologies and its community expertise in the use of heterogeneous, multi-core and accelerated architectures at leadership computing facilities may be of benefit to experimental colleagues.

The community is also working actively to develop approaches which can successfully bring artificial intelligence and machine learning approaches to our computational

* Contribution to the Topical Issue "Opportunities for Lattice Gauge Theory in the Era of Exascale Computing" edited by William Detmold, Andreas Kronfeld, Ulf-G. Meißner.

[a] e-mail: bjoo@jlab.org

[b] e-mail: chulwoo@bnl.gov (corresponding author)

toolkit where appropriate, and efforts are beginning to develop quantum computing methods for Lattice QCD.

After an introduction to Lattice QCD in sect. 1, a description of Lattice QCD workflow is given in sect. 2. Sections 3–5 give more detailed descriptions of algorithms, techniques and challenges in each major part of Lattice QCD worflow: gauge generation, propagator generation and correlation function construction. We discuss the hardware landscape as we see it currently going forward to the exascale in sect. 6, and discuss the software technologies and strategy that underlie our successful exploitation of leadership as well as smaller scale departmental sized computer systems in sect. 7. We conclude the white paper with a section on the potential use of machine learning techniques (sect. 8) and an outline of the state of the field in its approach to research in quantum computing (sect. 9).

1 Lattice QCD and leadership computing

1.1 Introduction

Lattice quantum chromodynamics is the only known, model-independent, non-perturbative method available for the evaluation of path integrals in quantum chromodynamics (QCD), and as such plays a vital role in modern theoretical high energy and nuclear physics. Lattice QCD calculations underpin vital research, for example testing the limits of the Standard Model of particle physics, aiding in the understanding of the production and properties of hybrid meson resonances and many other areas. The focus of this white paper is a description of the state of the art in computational technology underpinning such calculations, as well as further research opportunities in the computational arena to benefit future calculations on forthcoming pre-exascale and exascale systems. We also consider rapidly developing new areas of computation such as machine learning (ML) and Big Data approaches and quantum computing. We refer the reader to the companion white papers for details of research opportunities in various areas of nuclear and high energy physics [1–6].

In this initial section, we give a very high level overview of the interactions of Lattice QCD and computing and the main topics of this white paper, which are then expanded upon in (occasionally technical) detail in the subsequent sections. We will detail the mechanics of Lattice QCD workflows in sect. 2 outlining the main stages of the computation and commenting on the primary computational features of the stages. In sect. 3 we will discuss current Lattice QCD generation algorithms and advances including topics of current research, which we will follow in sect. 4 with a discussion of linear solvers and eigensolvers. Thereafter we will discuss *correlation function construction* in sect. 5, present our view of the current and upcoming hardware systems in sect. 6 and our software efforts to exploit them in sect. 7. We will round out our overview with details of nascent research into machine learning and quantum computing in sect. 8 and sect. 9 respectively.

1.2 Evaluating Lattice QCD path integrals

Through the process of discretizing four-dimensional spacetime onto a regular lattice, path integrals are turned into a high-dimensional but countable set of integrals. Due to analytical continuation to Euclidean time, the path integral weight of a configuration of fields can take on an interpretation as a Boltzmann-like probabilistic weight. Mathematically, the system to be solved becomes similar in nature to a crystalline system or spin-glass such as one may encounter in condensed-matter physics, and the path integrals themselves can be evaluated numerically through Monte Carlo methods. The probabilistic weight due to dynamical quarks corresponds to the determinant of the respective fermion kernels which are prohibitively expensive to evaluate directly. As such, these determinants are universally simulated by expressing the determinant as auxiliary integral over so called pseudofermion fields, which can be evaluated in the same process as the main path integral.

1.3 Exploiting leadership computers and new architectures

The cost of such calculations is formidable and in the pursuit of ever more realistic calculations, Lattice QCD practitioners have been working at the forefront of high performance computing (HPC) for over 40 years contributing in aspects as diverse as *building custom supercomputers*, *carrying out architecture specific code optimizations*, *inventing and contributing to new simulation algorithms*, adopting *software engineering best practices* and most recently applying other practices such as opportunistic running through workflow systems using similar tools as colleagues in experiment. Indeed, Lattice QCD has turned into a mature science with complex simulation campaigns carried out in an industrial fashion. Current sophisticated calculations have been carried out over a long period of time at large-scale facilities such as Oak Ridge Leadership Computing Facility (OLCF), Argonne Leadership Computing Facility (ALCF) and the National Energy Research Scientific Computing Center (NERSC), as well as using dedicated cluster facilities at Thomas Jefferson National Accelerator Facility (Jefferson Lab), Fermi National Accelerator Laboratory (Fermilab) and Brookhaven National Laboratory (BNL) in a coordinated fashion. An important aspect when using so many resources is the automation of the analysis portion of calculations where Lattice QCD researchers have formed a partnership with the ATLAS PanDA [8] team at BNL to bring workflow technologies such as the ones used in experimental big-data analysis into regular use for Lattice QCD calculations.

1.4 Lattice QCD spin-offs into other science and technology domains

The innovations in Lattice QCD research have had several spin-offs in other areas of science. The hybrid Monte

Carlo algorithm [9] used to generate lattice gauge configurations with dynamical fermions has found many other applications, including machine learning [10], the study of protein structures, *e.g.*, [11,12], and in the analysis of financial time series *e.g.*, [13,14]. Two of the same authors also developed a precursor of Riemann manifold HMC [15] more than 20 years before it was fully developed in [16].

1.5 Designing and utilizing novel hardware in partnership with industry

Lattice QCD practitioners have built or were involved in building several custom computers designed to carry out Lattice QCD simulations efficiently including early systems built at Columbia University, the GF11 supercomputer built at IBM [17], the APE series of computers in Europe *e.g.* [18] and relatively recently the QCDSP systems built at Columbia University (CU) and Brookhaven National Laboratory (BNL), QPACE systems at University of Regensburg [19], and the QCDOC [20] systems built as collaborative projects between CU, BNL, the RIKEN-BNL Research Center (RBRC) and Edinburgh University. The QCDOC system in particular was a sister project of the IBM Blue Gene/L [21] supercomputer, with the two systems having overlapping and collaborating design teams and having shared several pieces of co-designed hardware elements. Work under contract continued with IBM to develop the subsequent Blue Gene/P and Blue Gene/Q systems [22]. Lattice QCD researchers now work closely with companies such as Intel, Nvidia, HPE, and SGI in order to maximally exploit up and coming hardware offerings.

Lattice QCD researchers in Europe started to exploit graphics processing units (GPUs) as early as 2007 writing code over computer graphics interfaces [23] and in the U.S. since 2008–2009. The then newly released CUDA programming API was used to produce the QUDA library [24–28]. QUDA has become the basis for the exploitation of GPUs by USQCD, as discussed below in sect. 7. Jefferson Lab fielded the first large-scale, GPU-accelerated cluster designed specifically for Lattice QCD in 2009. Thus, the Lattice QCD community was ready when large-scale GPU resources appeared in 2012 on the Titan and BlueWaters systems at the OLCF and at the National Center for Supercomputing Applications (NCSA). Since then, the Lattice QCD community has formed strong partnerships with Nvidia (with several Lattice QCD researchers having found careers there), and with Intel to produce highly optimized codes to run on the emerging Intel Xeon Phi Knights series of computers working successively with Knights Ferry, Knights Corner and most recently Knights Landing (KNL), which powers several large-scale systems including the Cori system at NERSC and the Theta system at the ALCF. Several Lattice QCD researchers received the prestigious Gordon Bell Prize for High Performance Computing in 1998 (price/performance) and in 2006 (Special Achievement Award). Lattice QCD software was a Gordon Bell finalist in 2018.

1.6 Lattice QCD software

Lattice QCD in the U.S. has greatly benefited from continuous funding through successive iterations of the *Scientific Discovery through Advanced Computing* (SciDAC) program of the US Department of Energy (DOE), Office of Science. Through this funding an extremely capable set of codes and libraries have been developed that allow Lattice QCD practitioners to exploit the architectural diversity of currently available computing hardware. Most Lattice QCD codes are written in a mixture of C and C++, making use of on-node threading (typically with OpenMP on multi-core CPUs and CUDA on Nvidia GPU accelerators) and internode communications with message passing between nodes (and sometimes within them). Lattice QCD codes attempt to leverage development best practices including the use of distributed version control systems, and regular testing. Indeed, some pieces of software support full continuous integration testing.

1.7 Towards the exascale

As we stand at the dawn of the exascale era, the USQCD Collaboration is participating in the U.S. Exascale Computing Project (ECP), to ensure readiness of our codes for the forthcoming generation of computers. The USQCD contribution encompasses many areas including the improvement of gauge field generation, research in the areas of linear solver and related algorithms (*e.g.*, eigensolvers, and trace estimation), a re-tooling of the software infrastructure paying special attention to upcoming architectures, modularization and layering of software components and their interoperability, novel computing languages, and new programming models to provide productivity and performance portability.

1.8 Machine learning and quantum computing

Recent trends in high performance computing have raised to prominence the rapidly developing areas of machine learning, big data analysis, and quantum computing. Deep learning has major commercial applications that are influencing both hardware (such as the tensor cores on the Nvidia Volta V100 GPU, or the increase in support for low precision arithmetic on the Intel Xeon Phi Knights Mill architecture) as well as how some areas of the scientific enterprise are carried out. While machine learning has not yet been applied at a large scale to Lattice QCD simulation, it offers several areas where it can assist in optimizing simulation campaigns.

Finally, although the use of quantum computing for production Lattice QCD calculations is likely still several years away, the community is already investing resources to be ready for its exploitation, working with early quantum computing systems.

1.9 Summary

In summary, Lattice QCD calculations are, and have historically been, at the very forefront of high performance computing. Research topics range from areas as diverse as hardware architecture design, applied mathematics, software and workflow technologies, to machine learning and quantum computing. Lattice QCD has had several algorithmic spin-offs into other disciplines and maintains close links with industrial partners for co-design and early adoption and exploitation of new hardware technologies. Computational Lattice QCD is well poised to utilize the very first exascale and pre-exascale systems and is already undertaking research into exploiting the newly emerging area of quantum computing.

2 Lattice QCD calculations need speed and throughput

Lattice QCD calculations are generally comprised of three main parts. The first step is *the generation of ensembles of gauge field configurations*, which is a Monte Carlo sampling of the strong force fields in the vacuum, including all the effects of gluonic self interactions and the interactions of gluons and sea quarks. The ensembles are parameterized by the *strong coupling* and the *masses of the sea quarks*. These parameters implicitly set the lattice spacing. The second step of the process is the computation of quantities of interest (known as *observables*) which for quantities involving quarks generally includes the computation of *quark propagators*. The computation of an observable is also referred to as "measurement". We discuss all these stages in detail in subsequent sections, and will restrict our focus here to their computational properties.

2.1 Gauge generation

Generally several ensembles need to be generated, with a range of sea quark masses, lattice spacings and volumes, in order to allow controlled extrapolations between ensembles to the physical quark masses, the continuum limit and infinite volume. While calculations directly at the physical quark masses have recently become feasible, heavier mass ensembles may be required, for example, for parameter tuning purposes or to map the quark mass dependence of quantities of interest, depending on the calculation

Within each ensemble successive configurations are generated from preceding ones via a Markov process, so that the probability of a configuration occurring is proportional to its probabilistic weight in the path integral (*importance sampling*). In this case, measurements can simply be averaged over the configurations in the ensemble to form the estimators for the path integrals. The statistical uncertainty of these *ensemble averages* depends on the number of independent configurations, N_{cfg}, in the ensemble, and decreases as $1/\sqrt{N_{\mathrm{cfg}}}$. As such the statistical precision can be controlled by increasing the number of configurations in the ensemble, so that given sufficient computer time high-precision calculations are possible, for example, as are needed to test the Standard Model.

The primary computational cost of gauge generation is the cost of a single Markov update step during which time the lattice Dirac equation may need to be solved hundreds of times —as detailed in sect. 3— and at various quark masses. Due to the sequential nature of the Markov process, the only parallelism available is the data parallelism arising between the lattice sites, making gauge generation a *strong scaling* challenge. In current advanced simulations, a single Markov step may take between 10 minutes and 1 hour on on hundreds or thousands of GPU devices on a leadership system such as OLCF Titan. One generally saves the state of the fields every couple of steps, resulting in a data access pattern of writing one configuration of size a few tens of GB every hour or so.

2.2 Propagator computation

The second step in the process is the computation of quark propagators. This is akin to releasing a test charge in an electric field and following its progress along the field lines, but a color charge is used instead of an electric one in the background of the strong force (gauge) field, and rather than "following the movement" of a quark, an object known as a quark propagator is generated. The quark propagator is labeled at each end by indices corresponding to the position, color, and other quantum numbers. These indices later need to be contracted into "colorless" *correlation functions* in a step known as *correlation function construction*. The latter two steps are deeply related, depending on the nature of the observables at hand, since that dictates the number and nature of propagators which need to be computed. From a workflow point of view however, it is worth separating them, as the propagator calculation and contraction steps have somewhat different computational characteristics.

The propagator calculation step involves solving the lattice Dirac equation for a variety of color sources. Depending on the observables, O(100k–1M) propagator solves may need to be carried out on each configuration in a given ensemble. Here one can use the data parallelism available from the lattice, but also since each gauge configuration in an ensemble, and each quark source are treated as being independent, one has access to a large degree of comfortable parallelism. This step is very much in the vein of *industrial computing*. Rather than being a strong-scaling challenge *this step is gated by throughput*. Typically O(10–100) GPUs are used per solve instead of O(1000), and O(10–100) separate propagator computations can be run in a single job, on a leadership class system potentially using up to the whole system in ensemble mode. Since smaller partitions are needed for each solve, *mid-range* cluster resources built out of commodity white box components and having potentially less capable, *but substantially cheaper* communication fabrics than the leadership systems have proved extraordinarily cost effective for this step of calculations.

Due to their size, leadership systems can provide higher instantaneous throughput at any given time, potentially shortening time to solution. Ultimately, however, the overall progress on either type of system are gated also by the sizes of computer time allocations. Typical propagator runs in the area of quark propagators been able to regularly use O(2000–8000) GPU nodes of a system such as Titan at OLCF, or indeed several 1k–4k partitions of systems like the Intrepid IBM Blue Gene/P system (and its predecessors) at ALCF.

This phase of the calculation is entirely dominated by a variety of linear and eigensolvers, which will be discussed further in sect. 4. Further, this stage has a very different I/O pattern from gauge generation. In the case of Wilson fermions, a propagator component is about one third the size of a gauge field. Hence in a several hour job, with perhaps O(10k) solves, there is the potential to generate of O(10–100TB) of data. This is generally not feasible to save outright and typically some amount of *in-situ data reduction* is carried out. Hence this step is characterized by an ensemble style, with a high degree of output I/O that is generally written in a temporally structured pattern, and database technologies may well be used to store the output data. Due to the high amount of I/O this stage can also benefit from burst buffer technologies and potentially I/O staging and in-memory coupling with subsequent analysis steps.

2.3 Correlation function construction

The final contraction step generally uses the outputs of the propagator stage, to generate correlation functions, which can then be subjected to statistical analysis (fitting, error estimation, etc.). Typically, this stage is also carried out in an ensemble style. However, it is distinct from the propagator calculation, since the contractions typically need to be performed only over 3-dimensional space for a given value of Euclidean time. Hence, one can in principle often exploit parallelism among the Euclidean time-slices and codes also need to access 3-dimensional subsets of the original data. Since one needs to work generally on one time-slice at a time, the contraction codes are typically single node using at most thread level parallelism on multi-core processors. These codes have generally not yet been transferred to GPUs to such an extent as the preceding stages and exploiting more recent architectures for this work is a critical target of current development projects. These codes have yet again different I/O patterns, requiring essentially random read access to the products of the propagator calculation step, and can also benefit from database technologies, burst buffers and potential in-memory coupling to the preceding propagator calculation steps.

While gauge generation campaigns are typically carried out by simulating only a handful of ensembles concurrently over long periods of time, propagator and correlation function calculations employ a much higher level of ensemble parallelism and campaigns need to manage thousands of jobs. Workflow technologiues such as PANDA [8] or home-grown ones such as [29] present opportunities to have better control over the analysis campaigns, for example to automatically schedule and distribute jobs, restart failed ones and to exploit backfill cycles where available.

2.4 Simulations of QCD at finite temperature and density

An important exception to the pattern discussed above are calculations carried out at finite temperature. In these calculations, finite temperature is achieved by having shorter Euclidean time extents, so lattices are typically smaller than in the zero-temperature calculations, discussed above. In these situations, one can in many cases carry out both the gauge generation and the analysis part of the calculation using single nodes, and the key to progress comes from carrying out ensemble jobs to perform parameter sweeps, for example, to find phase transitions, critical points or to determine quantities as a function of temperature or density. In turn, finite-temperature codes have made perhaps best use of the hardware available in hybrid, accelerated nodes, by being able to separately use the GPUs for one part of their work (*e.g.*, propagator calculation) while using the CPUs concurrently to make progress on other aspects (*e.g.*, gauge generation). Due to their ensemble nature, and the requirements of using only a single node (or at most a few nodes) per task, these calculations can also derive a large benefit from workflow engines, and especially if these can exploit the backfill cycles of large systems such as those at leadership facilities. As finite-temperature calculations advance, the spatial lattices are expected to grow in size, and it will be necessary to move beyond large ensembles of single node jobs, to ensembles of multi-node jobs. Preparing code for this future is a focus of the DOE Office of Nuclear Physics SciDAC-4 project.

2.5 Impacts of hardware architecture trends

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. Simply put, a single stream of gauge generation may not be able to strong scale as effectively, to as many nodes as before. As a result, one can anticipate that gauge generation will also undergo a transformation to a more ensemble style calculation where (ideally) several independent Markov chains run in parallel. Each would use a larger partition than is used for propagator calculations, but with a single partition not yet taking up a very large fraction of a leadership system. Parallelism between Markov chains will be required to use the leadership hardware most efficiently. In this situation, gauge generation will become a much more complex task to manage in terms of human time, and it therefore will benefit from suitable workflow systems.

Finally, we note that Lattice QCD calculations are voracious, since the statistical precision is limited principally by the number of configurations in an ensemble and by the number of propagators one can compute on these configurations. As such, it is highly unlikely that the full needs of very high precision lattice calculations can be met by a single computer system even in the exascale era. It is highly likely that just as now, work will proceed through coordinated use of several sites, including ASCR facilities as well as local, institutional, or collaboration wide resources. To facilitate this, high speed networks and efficient data transfer tools (such as Globus [30]) are essential and are comparable in need to those of high-energy or nuclear physics experiments.

3 The computational casino: Gauge generation

3.1 Overview

In this section, we consider details of the gauge generation. As mentioned previously, the overall gauge generation process is a Markov chain, which generates each successive configuration from the previous one. As a large ensemble of independent configurations is required at each set of physical parameters, it is typical to run a few chains simultaneously. Successive configurations are typically correlated, and the number of Markov steps that must be taken before two configurations can be considered independent is characterized by some form of *autocorrelation time*, of which we typically consider two kinds. First is the *integrated autocorrelation time* for an observable, which is used to inflate the statistical error for the observable to take into account the autocorrelations between the configurations. The second is the *exponential autocorrelation time*, which is the maximum of all integrated autocorrelation times in the Markov process. They provide a guide how many Markov steps one needs to take, starting from a given position, before one can consider that one is sampling the desired equilibrium probability distribution properly.

The process of equilibration therefore is a modest upfront cost to generating an ensemble (typically a few hundred to a thousand updates) which needs to be paid at least once. While the ideal approach is to generate one single very long stream to minimize the cost of equilibration and the effects from rapidly increasing integrated autocorrelation times in simulations with ever finer lattice spacings (see sect. 3.6), the serial nature of ensemble generation can make the effective utilization of very large partitions on current large scale computing resources challenging due strong scaling constraints and as a result it is not uncommon to generate up to 4–5 streams per set of physical parameters in parallel. After an initial chain is deemed to have been equilibrated, other streams are generally split off from it. Each of these streams must also decorrelate from the main stream before its configurations can be considered independent.

As an example, in the USQCD Wilson-clover program, typically one aims for 6000–10000 Markov updates comprised of configurations in 4–5 ensembles, with an initial equilibration of approximately 1000 updates in the first ensemble and around 200–300 in each branching ensemble. This typically yields around 400–500 usable configurations which are still potentially correlated, so that binning and other error estimation techniques are used to estimate the true statistical uncertainties.

Since the parallelism of Markov chains is limited, one must rely on data parallelism provided by the lattice sites and the fact that the lattice gauge theories are local. Within a given chain, it is not possible to change the lattice volume (which would change the integral the Monte Carlo is evaluating). Further, while the typical cost of a simulation scales only mildly with the number of lattice sites (the volume), it scales in a much worse fashion with other physical parameters such as reducing lattice spacing or the light-sea-quark masses.

In order to focus the power of the computing elements onto these more challenging factors, rather than on increasing the volume, the primary scaling metric for gauge generation is *strong scaling*. In other words, as one scales to an increasing number of compute nodes, the global problem size remains unchanged and correspondingly, the problem size on each individual compute node decreases. Since the predominant communication patterns are nearest-neighbor (stencil-like) boundary exchanges and global sums, decreasing local problem size will engender a worse surface-to-volume ratio for each compute node. This results in decreased opportunity to *overlap local computation with communication* and a greater exposure to bottlenecks arising from communication latencies and bandwidths. Further, the evolution of hardware has favored increased single-node floating-point capability and memory bandwidth, while internode fabric capabilities have not advanced at a comparable rate, making the problem of strong scaling even worse on recent architectures.

In order to address the strong-scaling concerns, a great deal of current research is focused on communication reduction and avoidance in one form or another. Typical approaches include using domain-decomposition oriented preconditioners for linear solvers, communicating nearest-neighbor boundary data in reduced precision (requiring less bandwidth), and using communication-reduction oriented ("s-step") solvers which reduce the number of reduction points and, hence latency effects.

One of the recent advances in Lattice QCD has been the development of adaptive multi-grid methods for linear solvers for some fermion actions. Typically such solvers provide a near order of magnitude improvement over the best available implementation of regular Krylov solvers such as conjugate gradients [31] or BiCGStab [32] for light quark masses. However, for these solvers to be most effective they require a setup phase that can be computationally expensive, and they are more constrained by strong scaling on their coarser grids. In order to exploit multigrid solvers for gauge generation, both these issues will need to be confronted as we shall describe in sect. 3.4.

Finally, autocorrelation times are known to increase as the lattice spacing is reduced, leading to a phenomenon of *critical slowing down*. This is important as one performs calculations on ever finer lattices in order to give a better lever-arm for continuum extrapolations. Thus, in addition to combating the effects of strong scaling, a large component of research is focused on new Monte Carlo techniques that reduce this critical slowing down, as discussed in sect. 3.6.

In order to put these statements into context, in the following subsections we will describe today's workhorse algorithm of hybrid Monte Carlo. We will discuss our state of the art implementation for so called Wilson clover fermions (although the general principles are the same for any fermion action) including algorithmic improvements and the use of the multi-grid algorithm and will show the benefit of these improvements on the Summit and Titan systems at OLCF.

3.2 Hybrid Molecular Dynamics Monte Carlo

Hybrid Monte Carlo (HMC) [9], often known as Hamiltonian Monte Carlo, belongs to a class of algorithms referred to as hybrid molecular dynamics Monte Carlo (MDMC). Treating the $SU(3)$ link matrices of a gauge field as canonical coordinates, the methods proceed by ascribing *canonically conjugate momenta* and generating a Hamiltonian system, with

$$H(\pi, U) = T(\pi) + S(U), \tag{1}$$

where $T(\pi)$ is a kinetic energy term depending on the momenta, and the potential term $S(U)$ is the action to simulate. By drawing π on each lattice link from a Gaussian heat-bath, the kinetic energy term maintains its familiar form $T(\pi) = \frac{1}{2}\|\pi\|^2$. One can generate new configurations from old ones by performing *Hamiltonian molecular dynamics* (MD) time integration of Hamilton's equations in a fictitious simulation time. Starting from some state with momenta π' and gauge field (coordinate) U', a new state (π, U) is generated by MD and is then either accepted or rejected with a Metropolis [33] acceptance probability. If the trial state is rejected the original (π', U') becomes the next state in the chain.

In order for the process to work, it must be both *ergodic* and have the required equilibrium probability as its *fixed point*. Since the MD is energy conserving, the energy change along a trajectory and the resulting acceptance probability are determined solely by the truncation error in the numerical MD integration algorithm, which can be controlled by changing the integration step-size. However in this case the system is integrated on a single hypersurface of (nearly constant) energy. In order to ensure ergodicity, the momenta are refreshed regularly from a heat-bath which has the effect of moving the system to a different energy hypersurface. In the regular HMC algorithm, momentum refreshment is done before every trajectory but more elaborate approaches to momentum refreshment have been proposed (such as generalized hybrid Monte Carlo [34]).

In order for the desired equilibrium probability to be the *fixed point* of the Markov chain, it is sufficient (but not necessary) to ensure that the *detailed balance* is maintained by the algorithm. This can be ensured by using *reversible time-integration* methods which also preserve the integration measure. The requirements of reversibility and area preservation limit us in our choice of integrators and in practice they are satisfied by utilizing a reversible combination of *symplectic integrator steps*. Commonly used integrators are the second order leapfrog, second order minimum norm (2MN) [35], fourth order minimum norm (4MN) [35, 36], and more recently force-gradient [37–39] integration schemes, which over a trajectory of unit length, have truncation errors of $O(\delta\tau^2)$ and $O(\delta\tau^4)$ respectively where $\delta\tau$ is the integrator step-size. As the lattice volume grows, generally one needs to take smaller steps to maintain a constant acceptance rate. It can be shown that for an integrator that scales as $O(\delta\tau^n)$ for some integer power n, the numerical cost of the algorithm scales as $O(V^{1+1/2n})$.

In order to include quarks, one employs the method of pseudofermions. In this case, the Grassman integrals over quark fields are carried out explicitly and each fermion flavor adds a determinant weight into the equilibrium probability of a given configuration. Evaluating the full matrix determinant is computationally prohibitive, and instead the determinant terms are expressed as an integral over bosonic fields, making use of the identity for two degenerate flavors of quark, with fermion kernel M

$$\det(M^\dagger M) \propto \int \mathrm{d}\phi^\dagger \mathrm{d}\phi \, e^{-\phi^\dagger (M^\dagger M)^{-1}\phi}, \tag{2}$$

where the ϕ fields are known as pseudofermion fields. These integrals are folded into the main HMC process, by refreshing the pseudofermion fields from a heatbath at the start of every update and carrying out the integrals stochastically over the whole simulation. Typically the quarks are considered in degenerate pairs since the combination of $M^\dagger M$ in the integrals is manifestly Hermitian positive definite (HPD) and the resulting determinant is manifestly real. One exception is domain-wall fermion formalism where the determinant for a single flavor can be rewritten to be manifestly HPD [40, 41], without the need for the squared operator.

To simulate flavor combinations that cannot be expressed as $M^\dagger M$ or in other HPD ways, one can take roots of the squared term which are computed using an approximation such as the *optimal rational approximation* expressed in partial fraction form, *e.g.*, for the square root, as

$$\phi^\dagger \left(M^\dagger M\right)^{-1/2} \phi = A \sum_{i=1}^{N} p_i \phi^\dagger \left(M^\dagger M + q_i\right)^{-1} \phi, \tag{3}$$

where N is the order of the approximation and A, p_i and q_i are approximation coefficients. This approach is referred to as rational hybrid Monte Carlo (RHMC) [42]. Other approximation schemes are possible for example by using Chebyshev polynomial approximations in the polynomial hybrid Monte Carlo algorithm (PHMC) [43].

The molecular dynamics algorithms are composed of updates to the momenta and gauge field combined in a reversible manner. The momentum update needs to evaluate the force term resulting from the action, which for fermionic terms ultimately results in the need to solve the Dirac equation in several forms (discussed in sect. 4). These computations require the use of linear solvers, and the vast majority of the time spent in the MD updating is spent in force computations and in linear solvers.

3.3 State of the art algorithms

Current state of the art gauge generation codes employ an impressive battery of algorithmic tricks. As quark masses approach their physical values the linear system to be solved for the light quarks becomes increasingly more ill conditioned, and the resulting forces grow in size requiring finer and finer time-steps [44]. To overcome this, simulations employ *mass preconditioning* [45–47] by breaking up the light quark determinant, into a chain of auxiliary determinants as

$$\det(M^\dagger M) = \det\left[\frac{M^\dagger M}{M_0^\dagger M_0}\right] \det\left[\frac{M_0^\dagger M_0}{M_1^\dagger M_1}\right] \ldots \\ \det\left[\frac{M_{n-1}^\dagger M_{n-1}}{M_n^\dagger M_n}\right] \det\left[M_n M_n\right], \qquad (4)$$

where we use the shorthand notation

$$\left[\frac{M_i^\dagger M_i}{M_{i+1}^\dagger M_{i+1}}\right] = M_{i+1}^{-1}\left[M_i^\dagger M_i\right](M_{i+1}^\dagger)^{-1} \qquad (5)$$

and M_i and M_{i+1} are slightly different in some way, for example by having a slightly different quark mass. Since matrices in each term are identical except up to a small perturbation, the ratio terms will be close to the identity matrix with a small perturbation. As such, the force terms which result from the small perturbation will also be small. The final term, which cancels off the effects of the chain is a regular two flavor term, but is no longer at a light mass and can be integrated with reasonable cost. In order to take advantage of mass preconditioning one needs to use *a multiple time-scale integrator* [48] which allows each term in the action to be integrated on a separate time-scale. Terms with small forces can be integrated with fewer long steps (and fewer force evaluations), while terms with large forces (such as the cancellation two flavor piece or the gauge forces) need many fine time-steps, but their evaluation is considerably less expensive numerically. Heuristically time step sizes can be selected using the infinity norms of the associated MD forces [47], and can be more formally understood and tuned using the technology of *Poisson brackets* [37, 38].

A recent advance in this area has been the introduction of force-gradient integrators [37–39] which can give a fourth order accurate integrator with fewer linear system solutions per time step needed than the previously discussed fourth order minimum norm integrator. Force-gradient integrators can be understood using the framework of shadow Hamiltonian methods and Poisson brackets, however a particularly elegant implementation trick has been discovered [39], which makes their implementation surprisingly straightforward. Extending them to multiple time scales is also possible.

Finally to reduce the cost of linear systems even further one can utilize so called *chronological predictors*. These components attempt to provide a good initial guess to the current system to be solved based on the solutions from previous steps along the MD trajectory. While technically their use violates reversibility, this is done in a soft way which can be controlled by performing an accurate enough solution.

3.4 The use of multi-grid solvers in HMC

The use of multi-grid solvers in HMC presents numerous opportunities but also has some basic challenges. The biggest challenge is due to the need to set-up and refresh the near-null space basis on which the multi-grid method relies. Second, strong scaling is a more difficult challenge when using multi-grid solvers than for typical single grid solvers, since multi-grid strong scaling is dictated by the coarsest grid with the smallest number of lattice points. We discuss multi-grid solvers in detail in sect. 4 and here consider our working code which uses the implementation in the QUDA [24, 25, 27, 28] library. Our work builds on previous efforts, using the QOP-MG library [49] on CPUs, which has been reported in [50] in HMC and previously in [51] in the context of a DD-HMC simulation [52].

The QUDA multi-grid implementation constructs the null space by solving for the *null vectors* v_i by running an iterative solver on the system $Mv_i = 0$ with a random initial guesses for the vectors v_i, until some absolute precision, or maximum number of iterations is reached for each one. Current calculations use 24–32 null-space vectors per level, and so potentially, the setup cost is equivalent to nearly 24–32 solves which would pose a substantial overhead if it were performed before every solve. One way to reduce the overhead is to use the same subspace for several MD steps and refresh the subspace occasionally as done in [51].

Previous work has shown that for heavy quark masses a preconditioner may remain effective over a whole trajectory [50] although for lighter quarks the preconditioner deteriorates [51] as the MD proceeds, and the gauge fields and the resulting Dirac operator evolve. The deterioration is evidenced by an increase in the iteration count in the multi-grid solver. A simple strategy is to set an iteration threshold, and to refresh the subspace vectors when this threshold is reached. A further reduction in the refreshment cost can be achieved by not recomputing the subspaces using completely new random initial guesses, but by taking the existing vectors and iterating the null space solve on them for a fixed number of iterations to "polish" them. This brings two new parameters into play: the iteration threshold for refreshment, and the number of refresh iterations, both of which need to be tuned for optimal performance.

Several interesting research avenues remain open in the use of multi-grid solvers in gauge generation. One can attempt to reduce the cost of subspace creation, for example by using an adaptive process such as described [53, 54]. The strong scaling challenges can also be tackled for example using domain-decomposed preconditioners [54] and communication avoiding solvers. Higher raw performance may potentially be obtained by working with multi-grid algorithms in a block-solver mode solving several systems at once. Fitting the latter into an HMC algorithm still requires additional research.

3.5 A case study: Chroma on Summit

As a case study we describe briefly the implementation of gauge generation in the *Chroma* code [55] for use with GPUs in general and on the OLCF Summit supercomputer in particular. Chroma is built on a data parallel framework known as QDP++, and on GPU systems there is an implementation of this known as QDP-JIT [56] which generates the necessary GPU kernels out of the QDP++ expression templates "just-in-time" using the NVPTX back end of the LLVM compiler framework. Thus using the QDP-JIT framework all the lattice wide operations utilized by Chroma are automatically GPU accelerated. Chroma in turn implements a variety of HMC integration schemes, including leapfrog, second and fourth order minimum norm integrators, and most recently a force-gradient integrator [37–39].

For the linear solvers, Chroma calls out to the implementations in the QUDA library [24, 25, 27, 28], which provides a variety of solvers for the lattice Dirac equation as discussed earlier and in sect. 4. In particular for two flavor and determinant ratio solves we have used its implementation of the aggregation multi-grid solver, where multi-grid is used as a preconditioner to a generalized conjugate residual (GCR) iteration.

The multi-grid preconditioner features smoothers using the minimal residual (MR) [57] algorithm whereas the bottom solver is a recursively multi-grid preconditioned GCR [57] except for the lowest level where it is unpreconditioned. Recent improvements have also yielded an implementation of communication-avoiding GCR (CA-GCR) which can be used instead of MR in the smoothers, and instead of GCR at the coarsest level of the multigrid hierarchy, although comparative numbers from Summit are not yet available for this development. Subspace creation and operator coarsening can be done entirely on the GPU. The chronological predictor from QUDA has also been interfaced with Chroma which will give QUDA access to Chroma's chronological vectors for potential future algorithmic optimizations. Subspace refreshment is implemented as described earlier by setting an iteration threshold which can trigger refreshment if exceeded, and the number of refreshment iterations can also be chosen by the user.

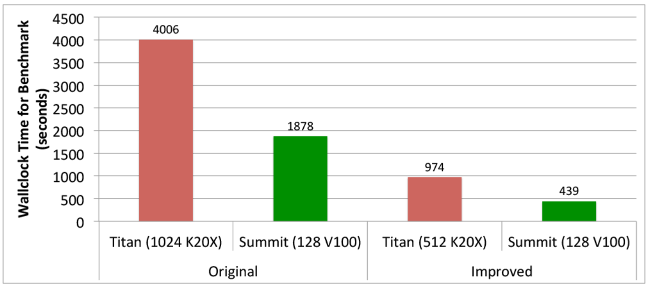

Fig. 1. Wallclock times for running a Wilson-clover gauge generation benchmark, on Titan (red bars) and Summit (green bars) prior to (left) and after (right) algorithmic and code optimizations.

In fig. 1, we show the wallclock times of running a gauge generation benchmark, which is a single trajectory on a lattice of size $64^3 \times 128$ sites with light quarks corresponding to a pion mass of 172 MeV. We show the performance of the original setup on Titan, which used three determinant ratio terms and three single-flavor (rational approximation) terms, two of which acted as the cancellation terms for the chain of determinant ratios. The evolution used the fourth order minimum norm time-stepper with five force evaluations (MN5FV). The two-flavor term used a GCR solver with a domain-decomposed preconditioner (DD+GCR) [26].

The optimized algorithm uses instead four determinant ratio terms, with the final term in the chain being canceled by a very heavy two-flavor term, and only one single-flavor rational term for the strange quark. The determinant ratio terms had the choice of masses re-optimized. All two flavor and determinant ratio solves on Summit used the multi-grid solver, with the subspace generated for the lightest quark flavor. Re-optimizing the determinant ratio mass choices was possible because the multi-grid solver really tamed the cost of the ratio terms featuring light masses. The optimized setup also used the newly developed force-gradient integrator [37, 38], the chronological predictor from QUDA and a host of QUDA improvements including pipelining of the GCR solver.

One can see that simply moving from 1024 nodes of Titan to 32 nodes of Summit (using 4 out of the 6 GPUs per node) without any algorithmic improvement resulted in a reduction in wallclock time from 4006 to 1878 seconds, or a 2.13× wall-clock speedup. However taking into account the reduction in the number of GPU devices (8×) the integrated speedup is 17×. Some of this is due to relief from strong scaling effects faced by the 1024 GPU Titan run, and the rest is from hardware improvements going from Titan to Summit. When the algorithmic improvements are also folded in the Summit time drops to 439 seconds, an overall wall-clock speedup of 9.13× compared to the original Titan run and taking into account the reduction in devices an overall improvement of 73×. The algorithmic improvements fed back to 512 Titan nodes (the maximum number of devices onto which the coarsest grid of the multi-grid could scale for this problem size) resulted in a run-time of 974 seconds, a wall-clock speedup of 4.11× and an integrated improvement of about 8.2× taking into account the reduction in the number of devices.

3.6 Critical slowing down and future prospects

As the lattice spacing is reduced in current and future simulations, the range of length scales in the problem (stretching from the pion Compton wavelength down to the lattice spacing) grows and the current evolution algorithms suffer critical slowing down as noted earlier: the stiff modes which evolve quickly must be integrated with a small step size while the soft, long-distance modes will change very little in such a step and require many steps to evolve significantly.

While this problem of critical slowing down has been recognized from the beginning and interesting solutions proposed in the 1980s, *e.g.*, ref. [58], current calculations may now involve a sufficiently large range of scales that substantial benefit may result from such methods. The study and development of methods to reduce critical slowing down is a current focus of the Lattice QCD application project within the DOE Exascale Computing Project. The most promising approaches are based on Fourier acceleration. Here the canonical momenta in the HMC evolution are made to depend on the gauge fields in such a way that the stiff, short-distance modes are given a larger, fictitious mass in the kinetic energy term, so that energy equipartition requires those modes to move with a small velocity while the soft, long-distance modes are given a small mass and hence a larger velocity.

Introducing such a gauge-field dependent mass faces two difficulties. First, the resulting kinetic energy depends on both the canonical momenta and coordinates, making the problem non-separable and requiring an implicit integration scheme [16]. Second, the usual relation between wavelength and frequency is spoiled by gauge symmetry, requiring either a gauge-invariant operator such as the lattice Laplacian be used in the mass term or that a gauge-fixed evolution be performed. At present a number of methods are being developed including the Riemann manifold hybrid Monte Carlo (RMHMC) algorithm [16] and the look-ahead HMC (LAHMC) algorithm [59]. While there are encouraging signs, a significant amount of statistics is still needed to estabilsh the effect of these algorithms on known observables with longest autocorrelation times.

3.7 Multi-grid inspired Monte Carlo methods

Finally, we note that there has been research into using multi-grid like approaches (distinct from multi-grid linear solvers) to speed up the decorrelation of HMC simulations. Recent work [60] has shown that one can reduce overall equilibration time for a given fine system by projecting a fine lattice onto a coarse lattice (restriction), equilibrating that at a faster rate using a suitably matched coarse action, then prolongating back to the fine level and re-equilibrating on the fine level. This approach can have several practical applications, from reducing the equilibration time of a single Markov chain, to producing independent seed configurations for several Markov chains which can then be simulated on their respective fine grids in parallel. To date this approach has been applied to pure Yang-Mills theories but there are no in principle issues with applying it to simulations with dynamical fermions.

4 Linear systems and eigensystems in Lattice QCD

4.1 Solvers in Lattice QCD

As we have discussed above, the solution of various forms of Dirac equations constitute a major part of Lattice QCD calculations. While colloquially referred to as *inversions*, what is required is the application of a matrix inverse onto a source, in other words, the solution of a linear system of equations. The solution of linear systems is a very rich field of numerical linear algebra and Lattice QCD calculations can, on the one hand drive research in this area of applied mathematics and on the other be a successful application of existing techniques.

The linear systems under considerations typically have dimensions proportional to the lattice volume, which for current calculations can be $O(10^8–10^9)$ sites, depending on the fermion formulation used (Wilson-like and staggered fermion formulations result in four-dimensional systems, while domain-wall like fermions possess an additional fifth dimension). In all cases, the linear operators in these systems are *complex valued* and *sparse* following either a nearest neighbor (Wilson-like or domain-wall–like) or next-to-nearest neighbor stencil pattern (improved staggered fermions). An exception is the so called *overlap formulation* where the linear operator itself is not nearest neighbor but evaluating it relies on applying a nearest-neighbor Wilson kernel. The linear operators themselves are not Hermitian, but typically have a J Hermitian form (some form of γ_5 Hermiticity, depending on the action) which is generally maximally indefinite. The methods of choice solving these systems are typically iterative solvers for sparse linear systems. As the quark masses in the linear operators approach the physical light quark masses from above, the resulting linear systems become ill conditioned resulting in poor convergence behavior for the more conventional Krylov subspace solvers, which is another form of *critical slowing down*.

It is standard practice to use a so called *even-odd (or red-black) checkerboard preconditioning* where iterative solvers need to be applied only to half the lattice sites (one checkerboard) using the Schur complement operator. The system on the other half of the sites is then usually trivial to solve. In HMC simulations, pseudofermions need only be kept for the checkerboard on which the Schur-complement operator acts, with the determinant on the other checkerboard being handled explicitly if it is not trivial. This gives a speedup of 2–3× in solves and also reduces the forces in the gauge generation compared to the unpreconditioned case.

Due to the different spectral properties of the different fermion formulations, the methods of choice for solving the various resulting Dirac equations differ. There are several systems of equations to solve: a) $Mx = b$ needed for propagators; b) $M^\dagger Mx = b$ needed for two flavor and ratio terms in gauge generation; c) the shifted system $(M^\dagger M + \sigma_i I)x_i = b$ with solutions x_i and shifts σ_i for systems arising in, for example, rational approximations; and finally d) $(M + \delta m_i I)x_i = b$ is useful, where δm_i is a shift in the quark mass (staggered and overlap fermions).

4.2 Commonly used Krylov subspace methods

The method of choice for forms c) and d) to date have been the use of multi-shift conjugate gradients (M-CG) [61,62]. This method generates all the solutions x_i for the cost of converging the system with the smallest shift. One downside of the method is that it relies on all initial guesses for the x_i to be parallel (typically the zero vector is used for all x_i), which limits several tricks to aid performance such as restarting methods and residual replacement strategies and hence also the use of reduced precision. Communication reduction approaches that rely on using reduced communication preconditioners also cannot be used. To combat these challenges, typically the solves are done to single precision initially, and then, if double precision is required, the individual solutions are polished with subsequent non-multishift, potentially mixed precision iterations. Additionally chronological approaches may be used so that information built up during polishing one solution can be applied to the next.

We note that in case b) one has a manifestly Hermitian, positive definite (HPD) combination of $M^\dagger M$ meaning that classical conjugate gradients [31] should always be able to solve the system, although in ill conditioned cases convergence may be very slow. In the case of form a) one can turn to conjugate gradients on the normal equations (CGNE) or on the normal residuals (CGNR). However, form b) can also be solved for some fermion actions using a two-step process by first solving $M^\dagger y = b$ for an auxiliary vector y and then solving $Mx = y$. The convergence of each step is thus now gated by the condition number of M or $M^\dagger$, which is the square root of that of $M^\dagger M$, which can result in an overall gain. However, since M and $M^\dagger$ are no longer HPD, a solver is needed which can deal with non-HPD matrices.

The Wilson and Wilson-clover formulations have perhaps lent themselves to the richest exploration of algorithms in the area of solvers for Lattice QCD. Early on it has been discovered that for heavy to medium light quark masses BiCGStab [32] is an effective solver for forms a) and b). In order to save effort at light quark masses, especially in the analysis part of calculations a variety of methods were developed including deflated solvers such as Eig-CG [63] and Eig-BiCGStab [64] which find the basis for the low modes of the operator during the initial first few solves and then deflate them in subsequent ones. Another approach to deflation was proposed in ref. [65], using FGMRES-DR [66] where a subspace developed during an FGMRES [67] Arnoldi cycle is reused in subsequent *augmented* Arnoldi cycles. Yet another approach to deflation was found in ref. [68], where a GCR solver was deflated based on a subspace generated by near zero modes of the operator M supported over blocks of the lattice, an idea that is in essence a two grid variant of an additive multigrid algorithm. Explicit deflation by a pre-computed basis of low-lying eigenvectors is also possible, and may be desirable if the eigenvectors computed can be used elsewhere, such as in the low mode averaging (LMA) and all mode averaging (AMA) approaches to computing correlation functions (see sect. 5). Recently *adaptive multi-grid* algorithms have also been successfully applied to Wilson-clover systems [54,69] which we discuss in sect. 4.3.

Other formulations have had varied success in employing newer solver technologies to date. BiCGStab, for example, simply fails with domain wall fermions. Multigrid approaches such as HDCG [70] and HDCR [71] have been developed for domain wall fermions but presently use the squared $M^\dagger M$ operator. Recently, a similar approach to [68], based on multispliting, has been shown to be effective for domain wall fermions [72]. In the area of staggered fermions, there has been an exploration of block solvers to make more efficient use of available memory bandwidth [73], with current development also exploring extended Krylov subspace methods. In parallel, research is underway as part of ECP to bring the benefits of multigrid solvers to staggered fermions [74].

4.3 Adaptive aggregation multi-grid methods

Multi-grid and related algorithms, based on local coherence [68], have perhaps been the biggest breakthrough technology in Lattice QCD calculations of Wilson-clover fermions of recent years [27,51,54,69,75]. The multi-grid cycles are typically implemented as a preconditioner to a flexible outer solver such as GCR or FGMRES. The preconditioning step computes an approximate inverse applied to a vector and so is itself a solver. Multi-grid preconditioners work, colloquially speaking, by separating high and low frequency modes of the linear operator. The error to the system from the high frequency modes is reduced by a step known as *smoothing*, and can be affected by for example iterating a simple solver like MR [57], or a Schwarz-alternating process [54]. To deal with the components of the error from low-modes of the operator, the system is projected onto a *coarse lattice* resulting in a coarse linear system using a coarse linear operator and vectors in a step known as *restriction.* A correction to reduce errors from the low modes is then computed on the coarse grid and the result is then moved back to the fine grid in a step called *prolongation.* Since this step may bring in some unwanted high frequency modes, the system may undergo *smoothing* again to result in a final multi-grid correction. The steps

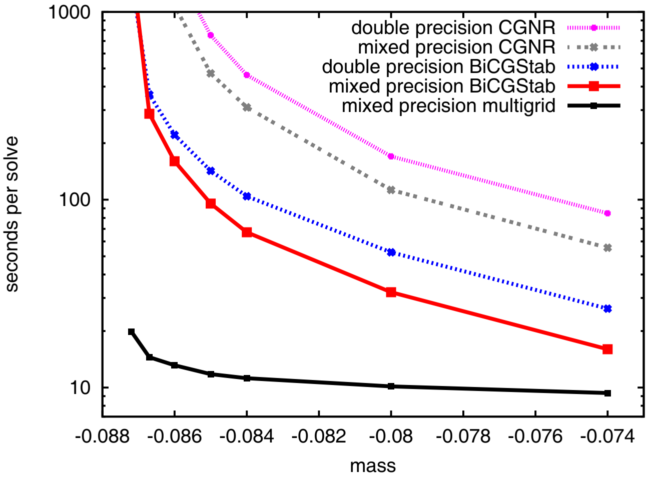

Fig. 2. Scaling of solvers with bare quark mass parameter. One can see that CGNR and BiCGStab solvers diverge in terms of solve time as the quark mass is reduced, while the scaling of multi-gird with decreasing quark mass is comparatively flat. From ref. [76].

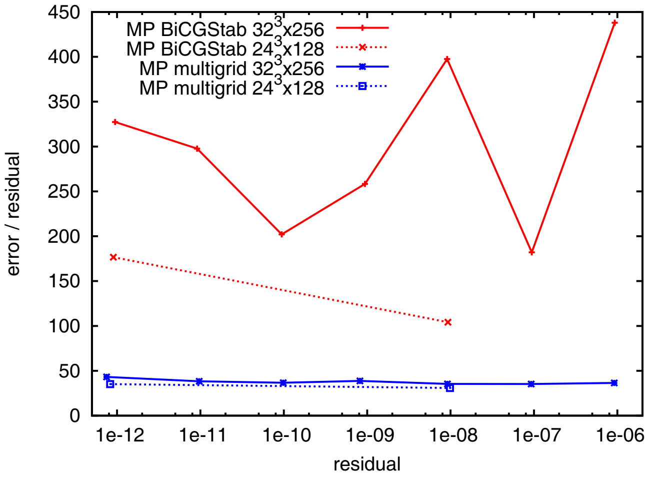

Fig. 3. A comparison of the ratio of the error to the residuum between BiCGStab and multi-grid solvers. The multi-grid result is low and does not fluctuate with the solver target residuum, indicating better solution quality. From ref. [75].

of restriction, coarse solve, prolongation and smoothing can be combined into various multi-grid cycles, including recursively such as the so called V cycle, K cycle, and F cycles.

The multi-grid methods used successfully to date in Lattice QCD have been variants of *adaptive smoothed aggregation* [77]. To define the coarse operator and the restriction and prolongation operators, the lattice is split into blocks and fine degrees of freedom on these blocks are aggregated to form the coarse degrees of freedom. This relies on the phenomenon of *local coherence* [68], which is a physical re-statement of the mathematical *weak approximation property*. Loosely speaking, local coherence means that the long wavelength modes on blocks of the lattice are good representations of long wavelength modes on the whole lattice, allowing the aggregation over blocks to produce a good coarse operator, which captures faithfully the low modes of the original operator.

Multi-grid algorithms have several very attractive features: They are *optimal* in the sense that their convergence properties (when properly tuned) should depend only on the volume of the system to be solved and they *eliminate the critical slowing down in terms of quark mass* as can be seen in fig. 2. As a result a well optimized implementation such as the one in the QUDA library for GPUs [24,27] can provide close to an order of magnitude improvement over the best optimized BiCGStab implementation for Wilson-clover fermions. This also results in highly increased energy efficiency of the computations. Since multi-grid works on minimizing the error rather than the residuum of a given system, the solutions produced tend to be of *higher quality* than, say, BiCGStab, in the sense that the error of the solution tends to be smooth and small over all the lattice sites whereas residuum based methods have error components that can fluctuate a great deal, as shown in fig. 3. Due to being used primarily as a preconditioner, multi-grid cycles can make use of the reduced precision capabilities of recent hardware architectures. However, as a result of working on a succession of coarser grids, the strong scaling of the method is gated by the volume of the coarsest level. This tends to be less of a problem for propagator analysis where one can scale up in ensemble mode running several solves simultaneously in its own small partition, but can be limiting in strong scaling situations like gauge generation. There are several ways to reduce strong scaling effects, for example by employing domain decomposition in the smoothers [54] or turning to *communications avoiding algorithms*. If the coarse system is sufficiently small, it can be replicated and solved (redundantly but faster) by all the nodes of a parallel calculation. This is a very active area of research as we head towards the exascale.

4.4 Reduced precision and communication reduction

Hardware architectural developments have had a strong influence on the development of Lattice QCD linear solvers. The linear operators for the most frequently used fermion formulations have low arithmetic intensities (*e.g.*, less than 1 FLOP/byte in single and less than 0.5 FLOP/byte in double precision) and are memory bandwidth bound on current architectures. In a parallel system, depending on the size of the halo region the computation may be impacted by both network bandwidth or latency. Multi-grid approaches on the coarser grid can typically have high surface to volume ratios and may also be latency bound. As a result a great deal of effort has gone into optimizing memory bandwidth use and reducing communications needs in implementations. Further, modern hardware often has a greater capability to deliver single precision FLOPs than double precision. As such, it is desirable to perform as much computation in reduced precision as possible.

Communication latencies can be reduced through the use of pipelining and *communication avoiding* Krylov solvers such as *s-step methods* that reduce the number of potentially latency sensitive *global reductions*. Bandwidth use can be reduced in cases by applying domain specific knowledge such as by employing gauge compression as pioneered in the QUDA library, which uses the properties of $SU(3)$ matrices to represent them either as two rows (reconstructing the third via vector-product) or by representing them through the coefficients of the 8 Gell-Mann generator matrices. These approaches reduce the amount of data transferred through the memory system, trading bandwidth for the freely available FLOPs required to reconstruct the original data before use.

Mixed precision can be exploited in solvers through using varieties of iterative refinement such as pioneered by the QUDA library in the form of *reliable updates*. Using a flexible outer solver process, which allows for nonstationary preconditioners, permits multiple optimizations to be employed in the preconditioner. First, one can use reduced precision with all the benefits that brings and second one can employ communication reduction/avoidance techniques such as domain-decomposition, or passing halo boundaries in reduced precision to save on network bandwidth.

Flexible outer solvers can also aid in fault-tolerance, since they can embed the desired regular solver as their preconditioner. Should the embedded solver complete without issue the surrounding outer flexible solver would not need more than one iteration. Should soft-faults affect the embedded solve, since it acts as a variable preconditioner, the outer process which should still converge eventually [78].

Another way to improve performance of these solvers is to turn to block-solver approaches. One feature of the block solver approach is that one can improve data reuse by applying the same linear operator to several vectors at once [73, 79] thus increasing the arithmetic intensity and enabling higher compute efficiencies. In some cases, this approach can also benefit vectorization (where one can imagine potentially vectorizing over multiple systems). Since in the analysis step many hundreds of thousands of systems need to be solved with the same gauge field, this approach can find easy application.

We should note that many of the above techniques can be successfully combined. For example gauge compression is straightforward to combine with all solvers for all fermion formulations. Residual replacement techniques such as reliable updates can also be broadly applied, except in special situations like the shifted conjugate gradients solver. Variants of these techniques are available to all fermion formulations in the QUDA library, for example.

4.5 Eigensolvers and deflation

While typical Lattice QCD problems have an enormous number of degrees of freedom, the distribution of eigenvalues, or singular values for non-Hermitian matrices, offers a path towards a significant reduction in computational cost. The spectrum of lattice Dirac operators is typically dense except for a relatively small number of low-lying (small in absolute value) eigenvalues, where the density is low. As shown in fig. 4, there are around ~ 2000 such low-lying eigenpairs for a physical box of size $\sim 5\,\mathrm{fm}$, while the total number of degrees of freedom is on the order of 10^9. Not only does this allow various exact and approximate eigenvector-based deflated linear solvers to be effective, but these eigenvectors can also be used to construct effective approximations for various quantities, further reducing the number of Dirac operator applications necessary to extract a given level of signal from each lattice configuration by as much as *two orders of magnitude*. Some of these techniques such as all mode averaging (AMA) [80, 81] or all-to-all [82] methods, will be described in sect. 5.

Naturally, *eigensolvers* or *singular value decomposition (SVD) solvers* are indispensable tools in such approaches. Fortunately, the distribution of eigenvalues, except for the smallest ones, tends to be rather stable between different lattice configurations which makes it possible to use polynomials with predetermined coefficients (typically Chebyshev polynomials) to create an effective filters for unwanted eigenvalues. Using such filters, thousands of eigenpairs can be converged by implicitly restarted Lanczos (IRL) [83] with the total number of applications of Dirac operators being typically less than twice the number of desired eigenvectors, minimizing the number of linear algebra operations. This allows the generation the necessary eigenvectors, often on the order of thousands, with an amount of Dirac operator applications comparable to those for 2–30 undeflated inversions, not dissimilar to the amount of work typically used to find the near-nullspace vectors for multi-grid solvers described in the previous section.

While the techniques developed so far have made the computational cost for eigenvector generation comparable with other (in)exact techniques, the need for storing these eigenvectors (temporarily or for the long term) poses an additional challenge. The mixed precision techniques for solvers are also helpful here. Also, the local coherence property [68] or "smoothness" of the eigenvectors, which underlies the success of the multi-grid algorithms in Lattice QCD, offers an opportunity for a significant reduction of storage space as well. Local coherence implies that the number of effective degrees of freedom is significantly smaller than the nominal degrees of freedom for these eigenvectors. This feature has been exploited to yield a compression algorithm on domain-decomposed eigenvectors [84] resulting in an order of magnitude or more of data reduction in existing eigenvectors. Local coherence also leads to a new and more efficient method for generating eigenpairs: by calculating eigenvectors directly on the subspace defined by domain decomposed lowest eigenvectors, one can generate eigenvectors that are nearly as accurate as the ones generated in the original vector space for a variety of measurements, as shown in fig. 5 for deflated CG, while decreasing the memory footprint at the same time. Lattice QCD simulations on exascale machines will use finer lattice spacing to control the discretization error.

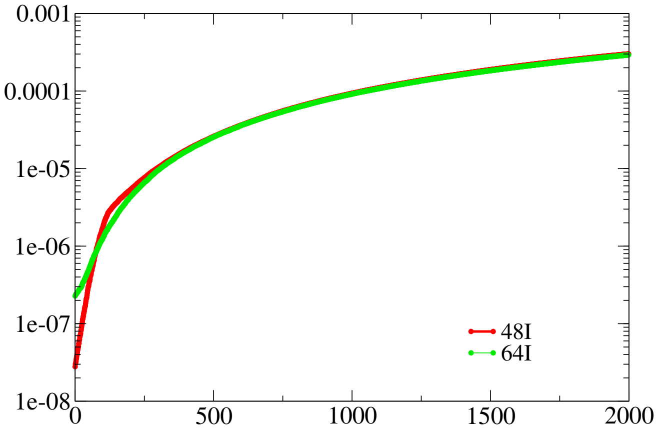

Fig. 4. Eigenvalues of the preconditioned Möbius domain-wall Dirac operator on 2+1 flavor Möbius ensemble (48I, 64I). See ref. [85] for details.

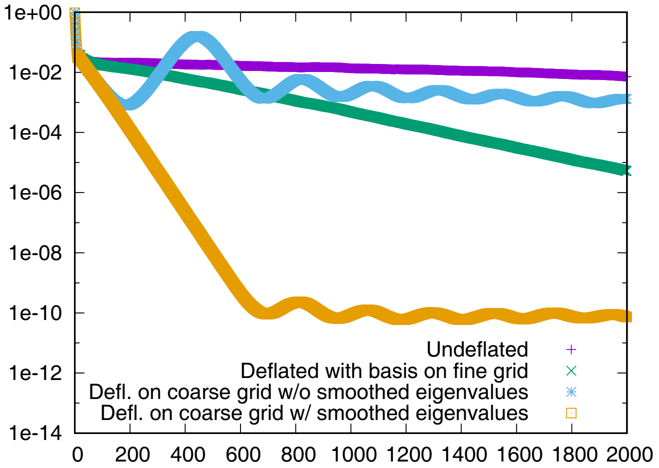

Fig. 5. Comparison of residual evolution for undeflated and deflated conjugate gradients, with eigenvectors generated on fine and coarse grids built with domain decomposed eigenvectors [84].

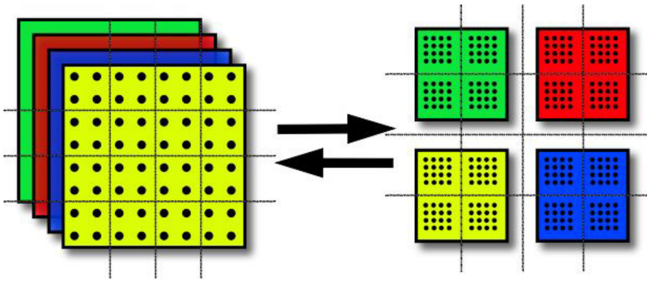

Fig. 6. Illustration of split grid.

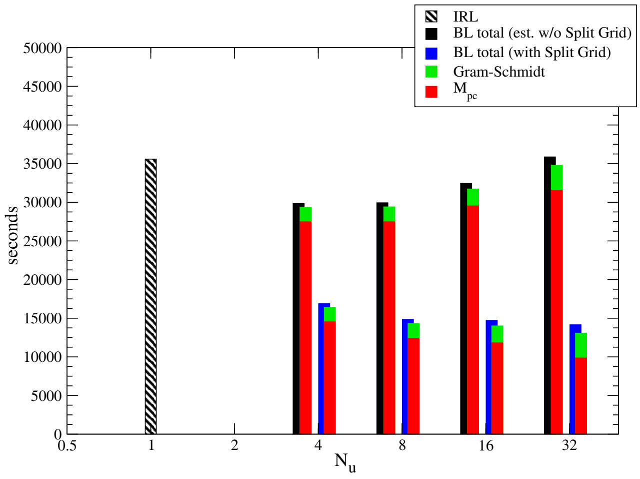

Fig. 7. Timings of block Lanczos with split grid on 512 nodes of ALCF Theta. From ref. [87].

This means the local coherence-based algorithms will result in even larger relative savings in memory, as the size of the blocks will increase in units of lattice spacing.

Many Lattice QCD applications must cope with the need to run on sufficiently many nodes to have enough memory for all necessary data, even though running on so many nodes would result in inefficiencies from strong scaling. While in some instances, one can split the workflow up into an ensemble approach, running ensemble members in essentially separate MPI jobs, this is not always desirable or possible. Inspired by software frameworks developed for big data such as Hadoop [86], the split-grid algorithm allows a single application to switch between one large domain and multiple smaller domains within a single application, so that the performance of the Dirac operator application within a sub-domain is not in the strong-scaling limit and can benefit from a better surface-to-volume ratio and network-bandwidth use, while routines without significant internode communications can be run on the larger partition making use of the aggregate memory to increasing data reuse and eliminate the need for a large amount of disk I/O. Figure 6 is an illustration of split-grid approach, while fig. 7 is an example of split-grid technique applied to block Lanczos, which shows a substantial overall speedup despite the fact that the number of Dirac operator applications is the same as, or larger than, the case for the single-vector IRL.

5 Correlation function construction

The penultimate[1] stage of the Lattice QCD workflow is the construction of Euclidean correlation functions. There are a variety of methods in use for this phase, developed to optimize the calculations of particular observables. Calculations of two-point correlation functions typically rely

[1] The final stage of the Lattice QCD workflow is statistical analysis, combining information from (ideally) several ensembles to produce a physics results with statistical and systematic uncertainty estimates. As this step poses no special computation problem, we do not cover it in this white paper.

on *quark smearing* to smooth unphysical ultraviolet fluctuations that lead to statistical noise. However, the same methods that can be used to extract the energies of eigenstates are not necessarily applicable to matrix elements, which also suffer from statistical noise. Hence one is concerned with two sources of statistical error in correlation functions: a) noise coming from the *gauge field sampling* (gauge noise) which can be controlled by the number of gauge configurations generated, and b) noise coming from measurements probing a given gauge configuration (measurement noise). This latter noise can be reduced by carrying out many independent measurements per configuration. Additionally, many quark operator constructions are needed to study nuclear systems; however, the number of permutations to contract the quark lines grows factorially. This challenge is discussed in more detail in sect. 5.4 of the companion white paper *Hadrons and Nuclei* [4].

5.1 Distillation

In a Monte Carlo calculation in lattice gauge theory, the physically relevant signal in a correlation function falls exponentially as a function of Euclidean time and is rapidly overwhelmed by statistical fluctuations. Operators that create low-lying energy eigenstates at early values of Euclidean time are therefore invaluable and improve the quality of data extracted exponentially. The generalized eigenvector variational procedure [88–90] provides a robust method to project onto the finite-volume eigenstates in a system, and helps to ameliorate the rapid fall-off of the signal with the statistical noise. The method relies on the construction of a large but diverse basis of operators that have varying overlaps onto the ground states as well as the excited energy levels in a system. A matrix of correlation functions is constructed with this basis, and the generalized eigenvalue problem is solved. The time-dependence of the eigenvalues allows the determination of the finite-volume energy levels in the system.

An efficacious approach to constructing a suitable basis of correlation functions is provided by the *distillation* method [91]. In this method, a low-energy basis of vectors is used to construct a low-rank definition of a quark-smearing operator. The factorization of the quark smearing leads to the construction of propagators within the low-rank basis, as well as the construction of multi-quark hadron operators that can be projected onto definite momentum. A major advantage of the approach is that, *a posteriori*, Euclidean correlation functions can be constructed. In particular, the method is well suited for investigations of multi-hadron correlation functions. The hadron operator constructions can have relative momentum and projected into an overall definite momentum transforming under suitable representations of the lattice cubic group of rotations [92].

5.2 Low- and all-mode averaging

As noted previously, the statistical errors of a given Lattice QCD calculation are limited by the number of measurements, which involve generating quark propagators from sources placed on different points of the lattice. The number of propagator inversions on a single configuration can reach into the tens-to-hundreds of thousands or more. Deflation techniques that exploit the relative sparseness of the lowest eigenvalues of the Dirac operator M, as well as multi-grid approaches, both of which lower the number of iterations in linear solvers thus bring a substantial reduction in computational cost. However, there are other properties of M which allow for even further cost reduction. One such property is that the accuracy of high and intermediate modes in the solutions does not affect the overall error of the measurement for most of the quantities measured in Lattice QCD. In other words, many "sloppy" measurements (with looser solver convergence criteria) with sources placed in different locations, or measured on different lattice configurations, produce a much smaller error than using fewer, but more accurate solutions (with tight solver convergence criteria). All-mode-averaging (AMA) [80] exploits this property by defining what is allowed for "sloppy" measurement without introducing bias.

For an observable $\mathcal{O}$ defined in terms of propagators, we can define an approximation $\mathcal{O}^{(appx)}$ which is both numerically cheaper than calculating $\mathcal{O}$ and remains covariant under lattice symmetry transformations, *i.e.*, for $g \in G$, $\langle \mathcal{O}^{(appx)}[U^g] \rangle = \langle \mathcal{O}^{(appx),g}[U] \rangle$, where g is a transformation from a larger set G, U^g is the transformed gauge field and $\langle\rangle$ denotes an ensemble average. With these definitions, an improved and unbiased estimate $\mathcal{O}^{(\mathrm{imp})}$ can be constructed as

$$\left\langle \mathcal{O}^{(\mathrm{imp})} \right\rangle = \left\langle \left(\mathcal{O} - \mathcal{O}^{(appx)} \right) \right\rangle + \frac{1}{N_G} \left\langle \sum_{g \in G} \mathcal{O}^{(appx),g} \right\rangle . \tag{6}$$

For low-mode averaging (LMA) the propagators in $\mathcal{O}^{(appx)}$ are constructed purely from the lowest eigenmodes, while in the case of AMA, one uses the "sloppy" propagators obtained either via a fixed iteration count or with a stopping condition that is relaxed compared to the more accurate propagators used to compute $\mathcal{O}$.

This further reduction of necessary M applications (from sloppy solves, and/or efficient eigensolvers to find low modes), on top of the reduction already obtained from the (in)exact deflation makes it possible for many measurements to achieve statistical errors comparable to those from fluctuation of gauge configurations (*e.g.*, ref. [80])

5.3 All-to-all propagators

The all-to-all (A2A) technique is used frequently when the quantities to be measured formally require a number of propagators from sources on each of the lattice sites, which typically number 10^6 or more. A well known example is a class of operators called "disconnected diagrams", in which there are one or more quark loops not connected to the external legs of the operators.

A2A approximates an arbitrary component of the M^{-1} as a sum of the outer product of pre-calculated vectors:

$$M^{-1} \simeq \sum_{i}^{N_l} |\lambda_i\rangle \frac{1}{\lambda_i} \langle\lambda_i| \\ + \sum_{i'}^{N_h} \left(M^{-1} - \sum_{i}^{N_l} |\lambda_i\rangle \frac{1}{\lambda_i} \langle\lambda_i| \right) |\eta_{i'}\rangle\langle\eta_{i'}| = \sum_{i=1}^{N_l+N_h} |v_i\rangle\langle w_i|,$$

where $|\lambda_i\rangle$ is the eigenvector with the eigenvalue λ_i and $\{|\eta_i\rangle\}$ form an orthonormal basis of stochastic sources. The idea being that propagators from the stochastic sources include the necessary high frequency modes, while the eigenvector basis captures the low lying part of the space.

The A2A technique turns the constructions of correlation functions from arbitrary lattice sites into a series of linear algebra operations without the need for additional Dirac operator inversions or inter-node communication except for the global sums. A sufficient number of eigenvectors for the low modes with efficiently constructed (stochastic) high modes allows for a reduction of the measurement error from the A2A procedure to smaller than the gauge noise. Further, there have been detailed studies in how to choose the pattern for selecting the $|\eta_i\rangle$ which allows for much better suppression of statistical error than the naive $1/\sqrt{N_h}$ factor, by judiciously choosing the grouping of degree of freedom which eliminates the most significant source of errors from random noise (*dilution*), or which allow for progressive winnowing of random points by utilizing Hadamard vectors (*hierarchical probing* [93]).

5.4 Multi-particle contractions

The evaluation of many-body correlation functions is a challenging aspect of the workflow of Lattice QCD calculations. One area where these calculations arise is the extraction of a few excited energy levels in many-hadron systems. The intrinsic finite-volume nature of the calculations is actually an advantage. The finite-volume energies, and matrix elements, can be related to their Minkowski space infinite volume scattering amplitudes via the Lüscher relations [94–96]. The analytic behavior of these S-matrix amplitudes can be used to extract resonant properties of states including branching fraction for decays that can then be directly confronted with experiment [97]. The low-energy behavior of scattering amplitudes in nuclear systems provides information on nuclear many-body forces [98].

These calculations must necessarily include many-quark operator constructs that provide overlap onto the finite-volume energy levels. With a large basis of quark fields, a particularly challenging aspect is the large combinatoric connections between the operators leading to large numbers of quark-line graphs that must be evaluated. A consequence of variational calculations is that there are common intermediate contractions that can be cached. Improved algorithms can find optimal orderings for constructing the temporaries and the graph evaluations, leading to significant cost savings. Research along these directions has been a focus of the Exascale Computing Project for Lattice QCD. The large combinatorics can be ameliorated by recasting the fermionic integration over the quark fields into the determinant of systems of equations [99,100]. It is clear that more research is needed in these areas.

6 Exploiting leadership computing systems

6.1 Historical perspective and machine building

Lattice QCD practitioners have been at the forefront of computing, to the extent of frequently designing special purpose systems that were tuned to the needs of QCD calculations. Notable examples of custom Lattice QCD systems in the U.S. include the ACPMAPS [101], QCDSP [102], and QCDOC [20] supercomputers. ACPMAPS was built from Weitek XL and Intel i860 processors and a custom-built communications backplane such that processing speed, latency, and data bandwith were all well balanced. QCDSP was built from Texas Instruments DSP processors with a network fabric made of point-to-point serial links. QCDOC combined IBM's System-on-a-chip intellectual property with a PowerPC-440 embedded CPU, a Hummer floating point unit, a communications network built on high-speed serial sinks (HSSL), and a custom memory controller. This project was a sister project of IBM's Blue Gene/L system with close collaboration between the design teams and with some of the co-designed components (such as the memory controller) being shared between the two systems. The communications fabrics of QCDOC and QCDSP, were designed to provide sufficient performance to allow the domain-wall and staggered fermion formulations of quarks to sustain a specific minimum level of performance even in the strong scaling limit. Indeed both QCDOC and the Blue Gene line were noted for their excellent strong scaling properties. USQCD researchers maintained their close links with IBM working as subcontractors in some cases on design elements of successor Blue Gene/P and Blue Gene/Q architectures. This work has also developed close ties with ALCF, where high performance Lattice QCD codes were used, for example, to troubleshoot the Blue Gene racks on delivery. With such a deep technical background, partnership with vendors, and ready software USQCD made excellent use of the line of Blue Gene systems at ALCF and to a certain degree at Lawrence Livermore National Laboratory (LLNL). QCD codes running on QCDSP and on a Blue Gene/L system at LLNL both won Gordon Bell Prizes.

In the space of computational clusters, innovations were made at Fermilab and Jefferson Lab to provide highly cost-effective systems for analysis, including meshing cluster nodes in a three-dimensional torus using gigabit ethernet links, Infiniband for adequate latency, and introducing

GPU accelerators. The first GPU accelerator-based cluster was contructed in Europe [23] with gaming GPUs, followed by a larger-scale installation at Jefferson Lab and systems with scientific-computing GPUs at Fermilab. Later, GPUs were incorporated into leadership-class systems such as Titan at OLCF or BlueWaters at NCSA (2012).

6.1.1 Utilizing GPUs

The first GPUs used for Lattice QCD were programmed using the OpenGL graphics programming interface [23]. The appearance of the more useable CUDA programming system spurred the USQCD development of the QUDA QCD library for GPUs [24, 25, 28] and its successful interfacing with the Chroma code [55, 103]. The original developers of QUDA found successful careers at Nvidia and maintained their links with the QCD community. This collaboration led to improvements of QUDA especially in terms of strong scaling, using the Edge analysis cluster at LLNL to strong scale Lattice QCD to over 100 GPUs for the first time [26]. In 2012, the era of GPUs in large petascale resources in the U.S. began with OLCF deploying Titan and NCSA deploying the BlueWaters systems, respectively. Lattice QCD codes were, thus, ready to exploit these new resources as soon as they became available. More recently USQCD researchers have been early users on the Summit system at OLCF, in particular providing test cases and benchmarks to OLCF. Some of the successful outcomes of this work are described in sect. 3.5. USQCD researchers also started utilizing tensor cores, first for mixed precision inverters studied in ref. [72].

6.1.2 Utilizing Intel Xeon Phi$^{\mathrm{TM}}$ architecture

Lattice QCD researchers also developed links with Intel Corporation to develop efficient kernels for the Many Integrated Core (MIC) architecture devices, later code-named Xeon Phi$^{\mathrm{TM}}$. The work began on the Intel Knights Ferry system and proceeded to Intel Knights Corner, in partnership with engineers at Intel, specifically Intel Parallel Computing Labs at Intel Santa Clara and Bangalore, India. High-performance code for the Wilson formulation of QCD available in 2013 [104, 105]. USQCD also collaborated with European colleagues on Xeon Phi software [106] and code for staggered and domain-wall fermions was being developed contemporaneously and reported shortly after, *e.g.* in ref. [79].

With these experiences, USQCD was well positioned to partner with NERSC during the procurement of the Cori system with sizeable Xeon Phi Knights Landing (KNL) partition of approximately 9600 nodes. Overall three USQCD codes (Chroma, MILC, CPS) askwere chosen to be Tier-1 NERSC Exascale Application Partnership (NESAP) codes, with another (QLua) becoming a Tier-2 code. As part of this partnership, these codes were further developed and optimized through hackathons and dungeon sessions with NERSC and Intel [107, 108]. In the context of USQCD, Jefferson Lab deployed a KNL cluster in 2016 and enlarged it in 2018. Brookhaven deployed one of the first production KNL cluster with dual rail Omni-Path network in 2017, which lead to the discovery and the development of the solution for the Omni-Path performance issue with KNL-based systems [109]. The KNL developments enable USQCD researchers to be productive on Cori (at NERSC) and on Theta (at ALCF). A recent summary of KNL code performance worldwide can be found in ref. [110].

6.2 Current DOE leadership computational landscape

The current computational landscape in the U.S. is in transition. ALCF, while awaiting the forthcoming exascale Aurora system, still operates the Mira Blue Gene/Q machine, having supplemented it with Theta, a system based on Intel KNL chips. NERSC operates Cori, which is a combination of dual socket Haswell servers (data partition) and KNL systems (Cori KNL partition).

Recently, OLCF deployed Summit, which, at the time of writing, is the fastest computer system in the world according to the Top 500 rankings [111]. Summit is a GPU-based system featuring the latest Nvidia Tesla V100 (Volta) GPUs. Each node has six GPU devices, connected with high-speed NVLink connections. The nodes themselves are connected with Infiniband. At the same time, OLCF continues to operate Titan, a Cray XT7 system featuring Tesla K20X GPUs and the Cray Gemini interconnect. USQCD codes can readily exploit Summit by treating it as a regular (albeit large) GPU cluster, however its Infiniband network can pose challenges to strong scaling. USQCD has been running on Titan since its arrival. Communications difficulties in GPU systems have driven software development activity to incorporate communication reduction and avoidance techniques to alleviate the situation.

6.3 Exascale and pre-exascale systems

The ASCR roadmap anticipates three large systems: i) Perlmutter at NERSC [112], ii) Aurora at ALCF [113], and iii) Frontier at OLCF [114]. The Perlmutter system will be delivered by Cray and will feature both a CPU (powered by Advanced Micro Devices CPUs) and a GPU-enabled partition, utilizing Cray's next-generation ethernet-compatible Slingshot fabric. From the point of view of existing GPU software, his architecture is an evolutionary step, so USQCD researchers should be well placed to be able to use the system.

At this point in time, very few public details are available about the Aurora system. The Early Science Program call for proposals [115] lists a few items of guidance, such as the suggestion that the architecture will be optimized to support codes with sections of fine grained concurrency, and that OpenMP 5 will likely contain the constructs necessary to guide the compiler to get optimal performance.

Similarly, not much information is yet available regarding the forthcoming Frontier system from OLCF, except that delivery is expected to begin in 2021 and that the system will support advanced simulation capabilities, high performance data analytics and artificial intelligence applications. The architecture might be expected to differ from that of Aurora.

As part of the Exascale Computing Project (ECP), some USQCD researchers are involved in co-design activities with ECP industrial partners. It is hoped that the combination of co-design partnerships, participation in early science projects (NESAP2, Aurora ESP etc.), and direct collaboration with ALCF, OLCF, and NERSC will enable the USQCD community to continue to be ready these new pre-exascale and exascale systems as soon as they become available.

6.4 General hardware trends

6.4.1 On-node parallelism

The general trend of new and future hardware is towards systems with increasingly high compute densities. This manifests itself primarily in ever more powerful nodes, with the main power of the node often coming from on-node parallelism of some nature, either via many cores on the node or via accelerator devices such as GPUs. The primary forms of parallelism on node have typically been multi-threading on CPU cores, single-instruction multiple-thread (SIMT) parallelism on GPUs and single-instruction multiple-data (SIMD) parallelism on CPUs (also known as vector parallelism).

6.4.2 Mixed precision

Graphics and machine-learning applications typically tolerate reduced precision compared with traditional scientific computing applications. Therefore, recent hardware provide higher 32-bit and 16-bit processing rates at the expense of double precision. For example, the tensor cores on the Nvidia Volta architecture enable OLCF's Summit to already breach the *exa-ops* barrier [116]. Intel, in turn, has announced the Knights Mill architecture which is an Intel Xeon Phi chip with improved single precision performance compared to the previous Knights Landing architecture [117] designed specifically to serve the artificial intelligence market. This trend can be expected to persist into the future, requiring developers of high performance libraries and frameworks to invest in codes that allow mixing of precisions and to follow developments in multi-precision algorithms.

6.4.3 Memory bandwidth

Nodes typically host a variety of memory speeds, often including high-bandwidth memory. On recent GPU systems, this has been on-device HBM/HBM2, whereas in KNL systems it has been on-package MCDRAM. For applications whose data fit into the fast memory completely, no real management of the fast memory was required. For others, a variety of solutions became available over time. Initially, one had to manage the memory explicitly and, in some cases, software cache systems were developed [56]. KNL could be run in a *cache mode* where the fast MCDRAM acted as a direct mapped level-3 cache that required no user intervention, at the cost of sacrificing some memory bandwidth. More recently, unified virtual memory and managed memory were developed on the Nvidia GPUs to reduce the burden of explicit management of transfers between host and device memory spaces.

6.4.4 Inter-processing-element bandwidth

While compute nodes have been getting denser and more powerful, the balance of memory bandwidth within the compute devices and between them has generally deteriorated. For example, the Blue Gene/Q system featured 42.6 GB/sec bandwidth to DRAM and 10 inter-processor links each running at 2.0 GB/sec giving a potential maximum off-chip transfer rate of 20 GB/sec which is approximately half the DRAM bandwidth. On the other hand, Summit contains 6 Nvidia Volta GPUs each having 900 GB/sec bandwidth to stacked high bandwidth HBM2 memory. Thus, the node provides a total of about 5400 GB/sec of memory bandwidth (without even considering the Power9 CPU sockets). Each GPU has three NVLink 2.0 bidirectional links running at 50 GB/sec per direction. Hence, the total off chip bandwidth of a GPU is 300 GB/sec (bidirectional) or 150 GB/sec in one direction (into or out of the GPU). The 300 GB/sec NVLink bandwidth is one-third of the HBM2 memory bandwidth. In turn, the maximum bidirectional bandwidth from the network interface card into the dual-rail Infiniband fabric is about 50 GB/sec. This is now less than 1% the bandwidth available within the node from the HBM2 memories on the GPUs.

The expectation of future systems is that this trend will continue, and the memory hierarchies will deepen (cache, fast memory like HBM2, DRAM, NVRAM, remote-node memory via fabric, disk). The compute elements will have access to limited memories with very high bandwidth, but the deeper one gets into the hierarchy the lower the bandwidth to the compute node will become. One can also anticipate islands of relatively high bandwidth connectivity within a node, *e.g.*, GPUs connected with NVLink. This skewing of the balance of inter-processing-element bandwidth to the fastest available memory bandwidth, can become a serious bottleneck for strong scaling, where using more compute nodes results in a smaller per-node problem, thus placing a heavier demand on communications between processing elements, for example for halo exchange. As such, research into algorithms that can reduce or avoid communications continues to be of the essence.

7 Software

Having a dependable, portable, and efficient software base is key to exploiting both current leadership class systems and future exascale systems. The software needs to address a variety of needs. First and foremost, it needs to be able to exploit the current generation of systems, in order to deliver the science goals of today. Second, it should provide a good basis for development to be able to exploit future systems. It should be modular and extensible to allow the addition or integration of high performance components addressing future architectures, and it should allow the exploration of new algorithmic directions.

7.1 Portable and efficient software from SciDAC

Through funding under multiple iterations of the DOE SciDAC program, USQCD has developed a layered approach to software. The four primary layers considered are: i) a communications layer called QCD Message Passing (QMP) ii) a data parallel productivity layer called QCD Data Parallel (QDP) iii) a layer for optimized components, primarily of Dirac operators and linear solvers known as "Level 3", and iv) an application layer which through judicious use of the layers below could encode the physics algorithms and be used for production. This last layer features the well-known Lattice QCD application codes MILC, Chroma, the Columbia Physics System (CPS), as well as newer code frameworks such as QLua and FUEL.

The QMP layer was defined to allow portable communications to systems that may lack a full-blown implementation of MPI, such as custom QCD systems like the QCDOC with its custom serial communications, and the Jefferson Lab gigabit ethernet clusters, on which QMP was implemented using M-VIA. The QDP layer had several implementations, including one over C++ known as QDP++ and one over C called QDP/C. QDP++ uses C++ expression template mechanisms to provide operator overloading, allowing the majority of code to follow the mathematics in a straightforward way.

Level 3 introduced a variety of high performance code libraries over the years, including SSE implementations of Wilson-Dirac operator, custom high performance implementation of the Wilson-Dirac and domain-wall fermion operators via the BAGEL library [118], optimized solvers for Möbius domain-wall fermions in the MDWF library [119], the first publicly available implementation of multi-grid for Wilson-clover fermions (QOP-MG) [49,75], linear solvers and QCD software components for GPUs in the QUDA library [24] and for Xeon Phi and Xeon architectures in the QPhiX library [104], and others.

These high-performance libraries also serve as crucial laboratories for developing new techniques. A prime example is the QUDA library, which, apart from its advanced solver algorithms, features a host of performance optimizations for GPUs, including per-kernel performance autotuning. Reduced precision is used where possible in the Krylov iterative steps and extensively in preconditioners. QUDA pioneered the use of gauge-link compression for performance (hitherto it has been used primarily to reduce file sizes on I/O) and the use of 16-bit precision in Lattice QCD. QUDA supports a variety of inter GPU communications including regular MPI (for between nodes), as well peer-to-peer and GDR for GPUs within a node or connected by NVLink. The choice of communications strategy is also tuned automatically. QUDA was also pioneering in the use of a domain decomposition preconditioner with the aim of improving scalability and continues to innovate in the area of communications-reduction and avoidance. Many developments in the investigation of multi-grid algorithms (*e.g.*, multi-grid for staggered fermions, communications avoidance in smoothers and the bottom solver, and reduced precision on GPUs) take place within the QUDA library. The library continues to be a testing ground for work in block solvers and extended Krylov-subspace solvers, as discussed in sect. 4.

In turn there have been innovations in other libraries. BAGEL, for example, pioneered the utilization of reduced precision on halo-swaps, some investigations of features of programming models (nested parallelism in OpenMP, use of the Kokkos [120] programming model) has taken place in the mg_proto [121] library, while SIMD aware layouts have been pioneered in BAGEL/BFM and are now in use in the Grid [122] software. Recursive-descent, cache-oblivious techniques are used in the MDWF library [119].

7.1.1 Adapting to disruption

Occasionally, new hardware technologies can disrupt existing programming paradigms. The arrival of GPUs was a prime example. One issue with GPUs was that due to the degree of acceleration they could provide to highly optimized code, other parts of an application which were not accelerated, could become a bottleneck due to Amdahl's law. Examples of this are various analysis tasks such as the creation of quark sources, which were insignificant before solvers were accelerated by GPUs and by algorithmic improvements, but have now become bottlenecks. In the gauge generation phase, code outside linear solvers in the computation of molecular dynamics forces suddenly came to dominate. One approach was to re-code the various routines one by one and optimize them for the GPU.

Another way was adopted in QDP++. In this instance, the expression templates of QDP++, which had hitherto generated code for the Lattice QCD expressions within the C++ compiler, were rewritten to generate code at run-time in a dynamic manner using a just-in-time (JIT) approach. This implementation of QDP++ is known as QDP-JIT [56]. The current QDP-JIT implementation generates code at runtime using the LLVM compiler framework, which targets GPUs using the NVPTX back-end and also has targets for most currently available mass market CPUs such as Power9, Intel x86, and compatible chips including Knights Landing and AVX512 support. By accelerating the entire QDP++ framework, all application code on top of it, including Chroma, is likewise automatically accelerated, thereby reducing the effects of Amdahl's law.

7.2 Developments under the Exascale Computing Project

USQCD software development currently is being undertaken under a SciDAC-4 project supported by the Office of Science, Office of Nuclear Physics and ASCR and through the Exascale Computing Project funded by ASCR. Broadly speaking, the purpose of SciDAC funding is to enable partnerships to make the best effective use of the current DOE leadership facilities, while the Exascale Computing Project (ECP) aims to ensure that high quality scientific software is available for the forthcoming pre-exascale and exascale systems. The majority of the software discussion above focuses on developments that were funded primarily by SciDAC. Here, other activities in the scope of USQCD's ECP participation are discussed.

As noted above, USQCD has several strands of work under ECP to carry out research in algorithms for improved linear solvers, to research gauge generation algorithms addressing critical slowing down, and to improve methods of post analysis. The ECP project also features software development, including the development of production systems, as well as experimentation with new hardware, and programming models.

Two primary data parallel layers being developed are Grid [122] and QEX [123]. Grid is a data-parallel framework similar to and inspired by QDP++. Unlike QDP-JIT, it utilizes GPUs through static compilation and features a data layout that is highly beneficial for SIMD processing on CPUs. QEX uses a new language called Nim which is a higher-level language similar to Python or Julia but featuring multiple dispatch, a module system similar to Python, incremental multi-stage compilation and an extremely rich approach to metaprogramming. Nim itself compiles to C directly with an extremely friendly C foreign-function interface to existing libraries. Both of these frameworks serve to remedy missing features of the previous QDP++ and QDP software layers (such as the ability to have multiple grids in scope in QDP++ for use in multi-grid codes) and to provide features the earlier software layers perhaps didn't anticipate such as the ability to carry out the split-grid approach discussed earlier.

Apart from the re-architecting of frameworks and applications, the software work under ECP also includes experimentation with novel hardware in partnership with several Path Forward projects with vendors, experimentation with new programming models such as Kokkos, or by combining and improving existing programming models with C++ features (*e.g.*, examining the interplay of directive based models like OpenACC and OpenMP with expression template methods). These exploratory projects typically select a set of kernels and attempt to implement them in the new programming model, to observe the resulting performance and to note any programming pain points. In this way, USQCD software work is synergistic with and beneficial to other ECP software technology projects.

8 Opportunities using machine learning

Machine learning (ML) is emerging as a powerful and versatile computational tool for scientific applications [124, 125]. For Lattice QCD, ML methods promise to accelerate typical computational workflows and also to enable new directions of exploration within lattice field theory. To be effective in the Lattice QCD context, ML methods must be adapted significantly from their use in other fields. In particular, ML implementations that incorporate the complex exact and approximate symmetries of Lattice QCD datasets must be developed, and rigorous methods of uncertainty quantification and error propagation must be investigated. While some promising early explorations of ML for Lattice QCD have been undertaken recently, these applications are not straightforward and to successfully apply them at the scale of state-of-the-art Lattice QCD calculations will need continued targeted research and development.

Gauge field generation. Several ML approaches have been proposed to accelerate the generation of Lattice QCD gauge-field configurations. Particularly promising are efforts to overcome critical slowing down in HMC as the lattice spacing is decreased [126, 127] (fig. 8), and ML approaches aimed at reducing autocorrelation within HMC [128]. While these studies for Lattice QCD are in early stages, for several of these algorithms it has been proven that they maintain the balance and ergodicity properties of HMC and are guaranteed to produce the correct probability distribution in the infinite statistics limit. There are also considerable parallel efforts to accelerate Monte Carlo methods in related systems [129–131]; successes in these simpler systems are promising for QCD.

Parameter optimization. Algorithms at all stages of the Lattice QCD workflow involve a significant number of tunable continuous and discrete parameters, whose optimal values are difficult to determine. For example, optimization of the many parameters defining HMC gauge generation (such as the concrete composition of the action to simulate the fermion determinant, and various tunable parameters in the molecular dynamics such as step sizes) is a complex task with strong correlations between parameters choices. Similarly, significant tuning is involved in the construction of approximate nullspaces in multi-grid solvers on large lattices, where the multi-grid scheme can involve a large number of levels and solver parameters at each level. In cases such as these, the parameter space is too large to feasibly explore using simple grid searches. ML tools such as Bayesian optimization using Gaussian processes are natural candidates as tools to optimize the parameters.

Sparse matrix inversions. As discussed in sect. 4, sparse matrix inversions are a significant cost in Lattice QCD

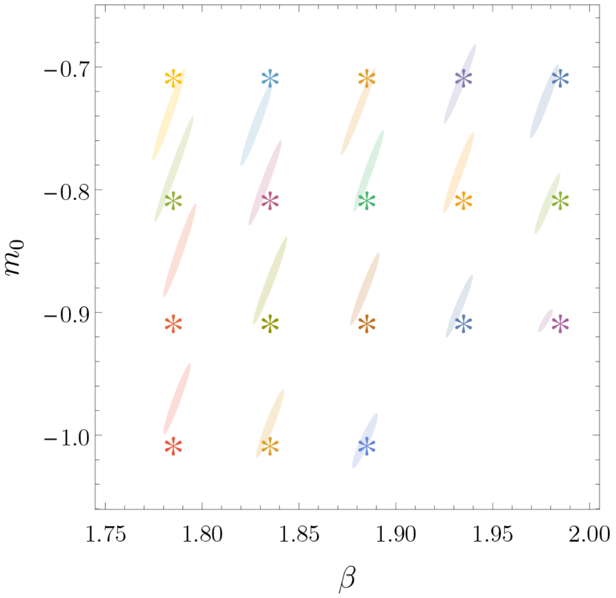

Fig. 8. Machine learning applied to parameter regression in ref. [127].

calculations. Preliminary explorations in applying ML to this task have recently been reported [132] with promising results. Showing that the approach works at the scale of state-of-the-art Lattice QCD simulations remains challenging.

Correlator optimization. ML techniques may provide a method to optimize Lattice QCD interpolating operators to minimize overlap onto unwanted states. This would reduce the computation required to determined properties of states which can be accessed with current techniques, and allow states which have previously been inaccessible to be studied. This is particularly relevant for nuclei, where exponentially more complex operator constructions are required to effectively suppress unwanted contamination of excited states. A number of ML approaches well suited to this task have been applied successfully to similar problems in molecular design and drug discovery. Recently, an ML approach to determining estimators for Lattice QCD has also been explored [133].

Phase transitions. ML has been successfully applied to detection of phase transitions and discovery of order parameters in a number of condensed matter systems [134–137]. A first application of similar methods in the context of quantum field theory has recently been used to study the deconfinement transition in $SU(2)$ Yang-Mills theory at finite temperature [138]. Further refinements may enable new insights into the nature of the QCD deconfinement and chiral symmetry restoration transitions.

Lefschetz thimbles and learnifolds. In recent studies of theories with sign problems, which are difficult to study using standard Monte Carlo methods (such as Lattice QCD at nonzero baryon density), a new approach has emerged based on complexification of the necessary integrations. Recently, variants of this approach based on optimizing the integration manifold using machine learning have been developed, with successful applications to the $1+1$-dimensional Thirring model [139] and scalar ϕ^4 theory in $1+1$ dimensions [140, 141]. While promising, extending these approaches to finite-density QCD is an extremely challenging long-term research problem.

9 Quantum computing and quantum information science

USQCD has carefully studied its HPC and workforce requirements during previous self-studies and through DOE initiated review processes, such as the Exascale Requirements Review that was completed in 2016. We have determined that, with access to exascale computing resources, we can determine many of the strong-interaction physics quantities that are important to the DOE missions in nuclear physics and in high-energy physics with the required precision. Indeed, that is the focus of this and the companion white papers [1–6]. That said, we have also determined that there are other important physics quantities that will not be able to be determined with sufficient precision, even with access to exascale computing resources or beyond. Precision computations directly from Lattice QCD of modest- and high-density systems, the real-time evolution of systems, transport in systems, out of equilibrium systems, and high-energy inelastic processes, such as fragmentation, are simply out of reach.

As USQCD is an organization that has been at the forefront of HPC architectures and their development. Examples include ACPMAPS, QCDSP, and QCDOC, which were designed by lattice-gauge theory teams, as well as early adoption of IBM Blue Gene, GPUs, Intel Xeon Phis. Through its long history of close collaboration with technology industry and vendors, it is natural for us to seek forward-looking solutions to our current challenges. The rapid developments in quantum information science (QIS) and quantum computing (QC) make it time to explore how quantum devices can complement classical computing resources. At present, cloud-accessible quantum devices at IBM (fig. 9) and Rigetti are available, as is private access to trapped ion systems. Further, there us the expectation that a range of such devices will become readily available, along with simulators and technical support. Finally, technology companies such as Google, IBM, Intel, and Microsoft expect to have universal quantum computers with about 50-qubits (without error correction) available now. IBM announced an operational 50-qubit QC near the end of 2017, and currently has a 20-qubit QC, a 16-qubit QC and two 5-qubit QCs available to users through the IBM Q Experience web-interface. The architectures of these QCs encompass superconducting qubits (IBM, Google and Intel) and topological qubits (Microsoft). D-Waves quantum devices use quantum annealing to address minimization problems. A growing number of other technology companies are also building programmable QCs,

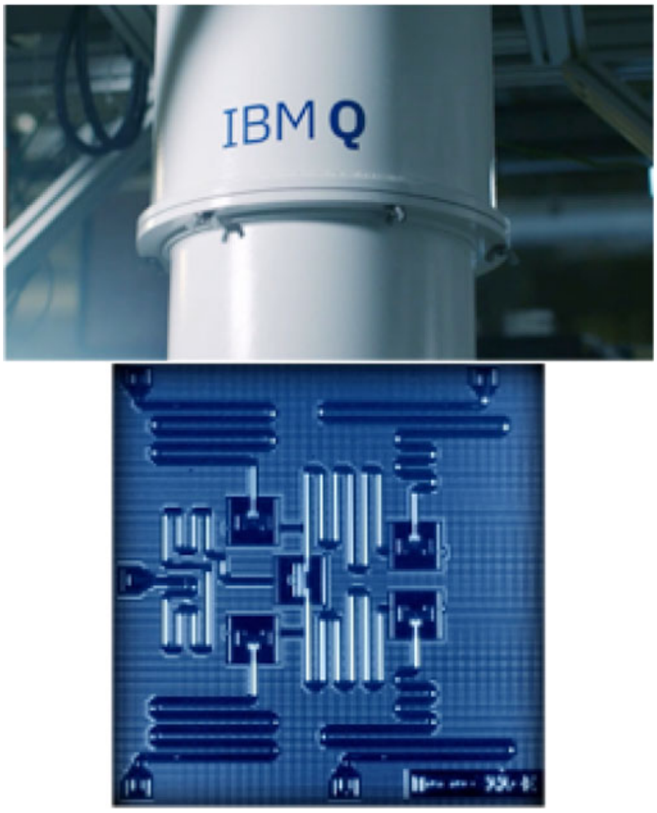

Fig. 9. Dilution refrigerator housing the superconducting quantum hardware (left) and the IBM QX2 quantum chip comprised of five qubits and radio-frequency couplings (right).

such as Rigetti and IonQ. In addition, some university- and national-laboratory-based groups are making significant progress in quantum simulation with recent results announced in quantum many-body systems obtained with programmable quantum simulators using more than 50 cold trapped ions as qubits.

Current hardware specifications, such as number of qubits, coherence times, measurement errors, are presently very restrictive, a circumstance that has been dubbed the noisy intermediate scale quantum (NISQ) era by John Preskill [142]. It is therefore unlikely that realistic quantum field theories will be simulated on such devices in the foreseeable future. In many ways, the present situation is reminiscent of the situation Lattice QCD found itself in during the 1970s with classical computing. Then, it was clear that certain classes of problems would be solvable with sufficient classical computing resources and with complementary developments in algorithms for linear algebra and ones specific to quantum field theory. With a quantum advantage for some problems in atomic and molecular systems expected to be gained with greater than about 50 qubits supporting a modest size gate depth, a quantum advantage for QCD is expected to require many orders of magnitude more and with error correction. However, with such capabilities, we expect to be able to perform reliable calculations of non-equilibrium systems, of the fragmentation of hadrons, and high-density strongly interacting systems.

It is conceivable that the first quantum devices to be deployed in a meaningful way to tackle QCD will be embedded in exascale, or beyond, classical computing hardware, and will consist of a few-qubit systems distributed within classical compute nodes, in analogy with the deployment of GPUs in present and future classical hardware. This scenario suggests that Lattice QCD could benefit from developing hybrid classical-quantum algorithms to perform some of the tasks currently performed solely on classical hardware or for new tasks for systems currently out of reach.

Given that this is only one possible scenario, it seems wise for a fraction of the USQCD Collaboration to embark on an effort to understand the potential of QC and QIS for QCD calculations important to high-energy and nuclear physics of the future. It makes sense to learn how to use available hardware to develop an understanding of this new computing technology, which can then be deployed to address the USQCD mission and begin to develop relevant algorithms. We anticipate the next several years to involve some retooling for some of the collaboration, a period to develop ties with technology companies, and to learn from other communities. USQCD has established a committee to perform ongoing evaluations of QIS and QC as it relates to quantum field theory and quantum chromodynamics. This committee is expected to provide regular updates to the USQCD Executive Committee. Finally, we expect that a funding framework similar in spirit to the SciDAC program would serve us well, as it continues to do so for classical computing developments.

As outlined in a white paper prepared for the US DOE, *Quantum Computing for Theoretical Nuclear Physics* [143], one can imagine that a plausible approach to begin to understand how to compute properties of quantum field theories with quantum computers is to begin by analyzing low-dimensional theories, starting with scalar field theory and the Schwinger model. Then increasing the dimensionality of the systems, moving to non-Abelian theories, and then finally to 3+1 dimensional QCD. This would include optimizing the layout on qubits, optimizing quantum circuits and then running codes on available quantum devices. It is conceivable that the need for efficient Trotterization of the QCD evolution operator could drive the design of quantum hardware and communication fabrics in the same way it did in classical computing. These are obvious and conceptually simple directions to consider.

Data Availability Statement This manuscript has no associated data or the data will not be deposited. [Authors' comment: All data have been given proper reference in the text.]

Publisher's Note The EPJ Publishers remain neutral with regard to jurisdictional claims in published maps and institutional affiliations.

References

1. USQCD Collaboration (Alexei Bazavov, Frithjof Karsch, Swagato Mukherjee, Peter Petreczky), *Hot-dense Lattice QCD*, `https://doi.org/10.1140/epja/i2019-12922-0` (2019).
2. USQCD Collaboration (Richard Brower, Anna Hasenfratz, Ethan T. Neil *et al.*), *Lattice gauge theory for physics beyond the Standard Model*, `https://doi.org/10.1140/epja/i2019-12901-5` (2019).
3. USQCD Collaboration (Vincenzo Cirigliano, Zohreh Davoudi *et al.*), *The role of Lattice QCD in searches for violations of fundamental symmetries and signals for new physics*, `https://doi.org/10.1140/epja/i2019-12889-8` (2019).
4. USQCD Collaboration (William Detmold, Robert G. Edwards *et al.*), *Hadrons and nuclei*, `https://doi.org/10.1140/epja/i2019-12902-4` (2019).
5. USQCD Collaboration (Andreas S. Kronfeld, David G. Richards *et al.*), *Lattice QCD and neutrino-nucleus scattering*, `https://doi.org/10.1140/epja/i2019-12916-x` (2019).
6. USQCD Collaboration (Christoph Lehner, Stefan Meinel *et al.*), *Opportunities for lattice QCD in quark and lepton flavor physics*, `https://doi.org/10.1140/epja/i2019-12891-2` (2019).
7. USQCD Collaboration (Bálint Joó, Chulwoo Jung *et al.*) this paper.
8. PANDA, `http://news.pandawms.org/panda.html`.
9. S. Duane, A.D. Kennedy, B.J. Pendleton, D. Roweth, Phys. Lett. B **195**, 216 (1987).
10. Christophe Andrieu, Nando De Freitas, Arnaud Doucet, Michael I. Jordan, Mach. Learn. **50**, 5 (2003).
11. C. Schütte, A. Fischer, W. Huisinga, P. Deuflhard, J. Comput. Phys. **151**, 146 (1999).
12. A. Brass, B.J. Pendleton, Y. Chen, B. Robson, Biopolymers **33**, 1307 (1993).
13. D.S. Dias, R.S. Ehlers, *Stochastic volatily models using Hamiltonian Monte Carlo methods and Stan*, arXiv:1712.02326 [stat.AP] (2017).
14. T. Takaishi, *Financial time series analysis of SV model by hybrid Monte Carlo*, arXiv:0807.4394 [q-fin.ST] (2008).
15. Simon Duane, Brian J. Pendleton, Phys. Lett. B **206**, 101 (1988).
16. Mark Girolami, Ben Calderhead, J. R. Stat. Soc. B **73**, 123 (2011).
17. John Beetem, Monty Denneau, Don Weingarten, SIGARCH Comput. Archit. News **13**, 108 (1985).
18. P. Vicini *et al.*, Comput. Phys. Commun. **110**, 216 (1998).
19. Paul Arts *et al.*, PoS **LATTICE2014**, 021 (2015) arXiv:1502.04025 [cs.DC].
20. P.A. Boyle, D. Chen, N.H. Christ, M.A. Clark, S.D. Cohen, C. Cristian, Z. Dong, A. Gara, B. Jo, C. Jung, C. Kim, L.A. Levkova, X. Liao, G. Liu, R.D. Mawhinney, S. Ohta, K. Petrov, T. Wettig, A. Yamaguchi, IBM J. Res. Dev. **49**, 351 (2005).
21. N.R. Adiga, G. Almasi, G.S. Almasi, Yariv Aridor, Rajkishore Barik, D. Beece, Ralph Bellofatto, Gyan Bhanot, Randy Bickford, M. Blumrich, A.A. Bright, Jos Brunheroto, Calin Cascaval, J. Castanos, W. Chan, Luis Ceze, Paul Coteus, S. Chatterjee, D. Chen, K. Yates, *An overview of the Blue Gene/L supercomputer*, in *SC '02: Proceedings of the 2002 ACM/IEEE Conference on Supercomputing* (IEEE, 2002) pp. 1–22.
22. P.A. Boyle, PoS **LATTICE2012**, 020 (2012).
23. Gyozo I. Egri, Zoltan Fodor, Christian Hoelbling, Sandor D. Katz, Daniel Nogradi, Kalman K. Szabo, Comput. Phys. Commun. **177**, 631 (2007) arXiv:hep-lat/0611022 [hep-lat].
24. M.A. Clark *et al.*, Comput. Phys. Commun. **181**, 1517 (2010) arXiv:0911.3191 [hep-lat].
25. Ronald Babich, Michael A. Clark, Bálint Joó, *Parallelizing the QUDA library for multi-GPU calculations in lattice quantum chromodynamics*, in *ACM/IEEE International Conference High Performance Computing, Networking, Storage and Analysis, New Orleans* (IEEE, 2010) `https://doi.org/10.1109/SC.2010.40`, arXiv:1011.0024 [hep-lat].
26. R. Babich, M.A. Clark, B. Jo, G. Shi, R.C. Brower, S. Gottlieb, *Scaling Lattice QCD beyond 100 GPUs*, in *SC11 International Conference for High Performance Computing, Networking, Storage and Analysis Seattle, Washington, November 12-18, 2011* (ACM, 2011) arXiv:1109.2935 [hep-lat].
27. M.A. Clark, Bálint Joó, Alexei Strelchenko, Michael Cheng, Arjun Gambhir, Richard Brower, *Accelerating Lattice QCD Multigrid on GPUs Using Fine-Grained Parallelization*, in *ACM/IEEE International Conference High Performance Computing, Networking, Storage and Analysis, Salt Lake City, Utah* (2016) `https://doi.org/10.1109/SC.2010.40`, arXiv:1612.07873 [hep-lat].
28. M.A. Clark, R. Babich, *QUDA: A library for QCD on GPUs*, `http://lattice.github.io/quda/`.
29. Venkitesh Ayyar, Daniel C. Hackett, William I. Jay, Ethan T. Neil, EPJ Web of Conferences **175**, 09009 (2018) arXiv:1802.00851 [hep-lat].
30. Ian Foster, IEEE Internet Comput. **15**, 70 (2011).
31. Magnus R. Hestenes, Eduard Stiefel, J. Res. Natl. Bur. Stand. **49**, 409 (1952).
32. H.A. van der Vorst, SIAM J. Sci. Stat. Comput. **13**, 631 (1992).
33. N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, J. Chem. Phys. **21**, 1087 (1953).
34. Alan M. Horowitz, Phys. Lett. B **268**, 247 (1991).
35. Tetsuya Takaishi, Philippe de Forcrand, Phys. Rev. E **73**, 036706 (2006) arXiv:hep-lat/0505020 [hep-lat].
36. I.P. Omelyan, I.M. Mryglod, R. Folk, Comput. Phys. Commun. **146**, 188 (2002) arXiv:cond-mat/0110585.
37. A.D. Kennedy, M.A. Clark, P.J. Silva, PoS **LAT2009**, 021 (2009) arXiv:0910.2950 [hep-lat].
38. M.A. Clark, Bálint Joó, A.D. Kennedy, P.J. Silva, Phys. Rev. D **84**, 071502 (2011) arXiv:1108.1828 [hep-lat].
39. Hantao Yin, Robert D. Mawhinney, PoS **LATTICE2011**, 051 (2011) arXiv:1111.5059 [hep-lat].
40. TWQCD Collaboration (Yu-Chih Chen, Ting-Wai Chiu), Phys. Lett. B **738**, 55 (2014) arXiv:1403.1683 [hep-lat].
41. C. Jung, C. Kelly, R.D. Mawhinney, D.J. Murphy, Phys. Rev. D **97**, 054503 (2018) arXiv:1706.05843 [hep-lat].
42. M.A. Clark, A.D. Kennedy, Phys. Rev. Lett. **98**, 051601 (2007).
43. Roberto Frezzotti, Karl Jansen, Phys. Lett. B **402**, 328 (1997) arXiv:hep-lat/9702016 [hep-lat].
44. CP-PACS, JLQCD Collaborations (A. Ukawa), Nucl. Phys. Proc. Suppl. **106**, 195 (2002).
45. Martin Hasenbusch, Phys. Lett. B **519**, 177 (2001) arXiv:hep-lat/0107019.

46. M. Hasenbusch, K. Jansen, Nucl. Phys. B **659**, 299 (2003) arXiv:hep-lat/0211042 [hep-lat].
47. C. Urbach, K. Jansen, A. Shindler, U. Wenger, Comput. Phys. Commun. **174**, 87 (2006) arXiv:hep-lat/0506011 [hep-lat].
48. J.C. Sexton, D.H. Weingarten, Nucl. Phys. B **380**, 665 (1992).
49. J.C. Osborn, *QOPQDP software library*, `http://usqcd-software.github.io/qopqdp/`.
50. Meifeng Lin, *Multigrid in HMC*, `https://indico.fnal.gov/event/7435/session/1/contribution/9/material/slides/0.pdf` (2013).
51. Martin Lüscher, JHEP **12**, 011 (2007) arXiv:0710.5417 [hep-lat].
52. Martin Lüscher, JHEP **05**, 052 (2003) arXiv:hep-lat/0304007 [hep-lat].
53. Martin Lüscher, Comput. Phys. Commun. **156**, 209 (2004) arXiv:hep-lat/0310048 [hep-lat].
54. Andreas Frommer, Karsten Kahl, Stefan Krieg, Björn Leder, Matthias Rottmann, SIAM J. Sci. Comput. **36**, A1581 (2014) arXiv:1303.1377 [hep-lat].
55. Robert G. Edwards, Bálint Joó, Nucl. Phys. B. Proc. Suppl. **140**, 832 (2005).
56. F.T. Winter, M.A. Clark, R.G. Edwards, B. Joó, *A framework for lattice QCD calculations on GPUs*, in *Proceedings of the 2014 IEEE 28th International Parallel and Distributed Processing Symposium*, IPDPS '14 (IEEE Computer Society, Washington, DC, USA, 2014) pp. 1073–1082.
57. Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd edition (Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2003).
58. G.G. Batrouni, G.R. Katz, Andreas S. Kronfeld, G.P. Lepage, B. Svetitsky, K.G. Wilson, Phys. Rev. D **32**, 2736 (1985).
59. Jascha Sohl-Dickstein, Mayur Mudigonda, Michael DeWeese, *Hamiltonian Monte Carlo without detailed balance*, in *Proceedings of the 31st International Conference on Machine Learning, Proceedings of Machine Learning Research*, Vol. **32**, edited by Eric P. Xing, Tony Jebara (PMLR, Bejing, China, 2014) pp. 719–726.
60. Michael G. Endres, Richard C. Brower, William Detmold, Kostas Orginos, Andrew V. Pochinsky, Phys. Rev. D **92**, 114516 (2015) arXiv:1510.04675 [hep-lat].
61. U. Glassner, S. Gusken, T. Lippert, G. Ritzenhofer, K. Schilling, A. Frommer, Int. J. Mod. Phys. C **7**, 635 (1996) arXiv:hep-lat/9605008 [hep-lat].
62. Beat Jegerlehner, *Krylov space solvers for shifted linear systems*, arXiv:hep-lat/9612014 (1996).
63. Andreas Stathopoulos, Kostas Orginos, *Computing and deflating eigenvalues while solving multiple right hand side linear systems in Quantum Chromodynamics* (2007) arXiv:0707.0131 [hep-lat].
64. Abdou Abdel-Rehim, Kostas Orginos, Andreas Stathopoulos, PoS **LAT2009**, 036 (2009) arXiv:0911.2285 [hep-lat].
65. Andreas Frommer, Andrea Nobile, Paul Zingler, *Deflation and flexible SAP-preconditioning of GMRES in lattice QCD simulation*, arXiv:1204.5463 [hep-lat] (2012).
66. Ronald Morgan, SIAM J. Sci. Comput. **24**, 20 (2002).
67. Youcef Saad, SIAM J. Sci. Comput. **14**, 461 (1993).
68. Martin Lüscher, JHEP **07**, 081 (2007) arXiv:0706.2298 [hep-lat].
69. R. Babich, J. Brannick, R.C. Brower, M.A. Clark, T.A. Manteuffel *et al.*, Phys. Rev. Lett. **105**, 201602 (2010) arXiv:1005.3043 [hep-lat].
70. P.A. Boyle, *Hierarchically deflated conjugate gradient* arXiv:1402.2585 [hep-lat] (2014).
71. Azusa Yamaguchi, Peter Boyle, PoS **LATTICE2016**, 374 (2016) arXiv:1611.06944 [hep-lat].
72. Jiqun Tu, *Solving DWF Dirac Equation Using Multisplitting Preconditioned Conjugate Gradient* arXiv:1811.08488 [hep-lat] (2018).
73. M.A. Clark, Alexei Strelchenko, Alejandro Vaquero, Mathias Wagner, Evan Weinberg, Comput. Phys. Commun. **233**, 29 (2018) arXiv:1710.09745 [hep-lat].
74. Richard C. Brower, M.A. Clark, Alexei Strelchenko, Evan Weinberg, Phys. Rev. D **97**, 114513 (2018) arXiv:1801.07823 [hep-lat].
75. J.C. Osborn, R. Babich, J. Brannick, R.C. Brower, M.A. Clark *et al.*, PoS **LATTICE2010**, 037 (2010) arXiv:1011.2775 [hep-lat].
76. J. Brannick, R.C. Brower, M.A. Clark, J.C. Osborn, C. Rebbi, Phys. Rev. Lett. **100**, 041601 (2008) arXiv:0707.4018 [hep-lat].
77. M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, J. Ruge, SIAM J. Sci. Comput. **25**, 2004 (2004).
78. Michael A. Heroux, *Toward resilient algorithms and applications*, in *Proceedings of the 3rd Workshop on Fault-tolerance for HPC at Extreme Scale*, FTXS '13 (ACM, New York, NY, USA, 2013) pp. 1–2.
79. O. Kaczmarek, C. Schmidt, P. Steinbrecher, M. Wagner, *Conjugate gradient solvers on Intel Xeon Phi and NVIDIA GPUs*, in *Proceedings, GPU Computing in High-Energy Physics (GPUHEP2014): Pisa, Italy, September 10-12, 2014* (Verlag Deutsches Elektronen-Synchrotron, Hamburg, 2015) pp. 157–162, arXiv:1411.4439 [physics.comp-ph].
80. Thomas Blum, Taku Izubuchi, Eigo Shintani, Phys. Rev. D **88**, 094503 (2013) arXiv:1208.4349 [hep-lat].
81. Eigo Shintani, Rudy Arthur, Thomas Blum, Taku Izubuchi, Chulwoo Jung, Christoph Lehner, Phys. Rev. D **91**, 114511 (2015) arXiv:1402.0244 [hep-lat].
82. Justin Foley *et al.*, Comput. Phys. Commun. **172**, 145 (2005) arXiv:hep-lat/0505023.
83. Daniela Calvetti, Lothar Reichel, Danny C. Sorensen, Electron. Trans. Numer. Anal. **2**, 1 (1994).
84. M.A. Clark, Chulwoo Jung, Christoph Lehner, EPJ Web of Conferences **175**, 14023 (2018) arXiv:1710.06884 [hep-lat].
85. RBC, UKQCD Collaborations (T. Blum *et al.*), Phys. Rev. D **93**, 074505 (2016) arXiv:1411.7017 [hep-lat].
86. Mike Cafarella, Doug Cutting, *Apache hadoop software library*, `https://hadoop.apache.org/`.
87. Yong-Chull Jang, Chulwoo Jung, PoS **LATTICE2018**, 309 (2019).
88. Christopher Michael, Nucl. Phys. B **259**, 58 (1985).
89. Martin Lüscher, Ulli Wolff, Nucl. Phys. B **339**, 222 (1990).
90. Benoit Blossier, Michele Della Morte, Georg von Hippel, Tereza Mendes, Rainer Sommer, JHEP **04**, 094 (2009) arXiv:0902.1265 [hep-lat].
91. Michael Peardon *et al.*, Phys. Rev. D **80**, 054506 (2009) arXiv:0905.2160 [hep-lat].

92. Jozef J. Dudek, Robert G. Edwards, Christopher E. Thomas, Phys. Rev. D **86**, 034031 (2012) arXiv:1203.6041 [hep-ph].
93. Andreas Stathopoulos, Jesse Laeuchli, Kostas Orginos, *Hierarchical probing for estimating the trace of the matrix inverse on toroidal lattices* arXiv:1302.4018 [hep-lat] (2013).
94. Martin Lüscher, Nucl. Phys. B **354**, 531 (1991).
95. Martin Lüscher, Nucl. Phys. B **364**, 237 (1991).
96. K. Rummukainen, Steven A. Gottlieb, Nucl. Phys. B **450**, 397 (1995) arXiv:hep-lat/9503028 [hep-lat].
97. Raul A. Briceno, Jozef J. Dudek, Robert G. Edwards, David J. Wilson, Phys. Rev. Lett. **118**, 022002 (2017) arXiv:1607.05900 [hep-ph].
98. Paulo F. Bedaque, Phys. Lett. B **593**, 82 (2004) arXiv:nucl-th/0402051 [nucl-th].
99. William Detmold, Kostas Orginos, Phys. Rev. D **87**, 114512 (2013) arXiv:1207.1452 [hep-lat].
100. Pranjal Vachaspati, William Detmold, PoS **LATTICE2014**, 041 (2014) arXiv:1411.3691 [hep-lat].
101. Paul B. Mackenzie, E. Eichten, G. Hockney, H.B. Thacker, R. Atac, A. Cook, M. Fischler, I. Gaines, D. Husby, T. Nash, Nucl. Phys. Proc. Suppl. **4**, 580 (1988).
102. Dong Chen *et al.*, Nucl. Phys. Proc. Suppl. **73**, 898 (1999) arXiv:hep-lat/9810004 [hep-lat].
103. R.G. Edwards, B. Joó, F. Winter, *The Chroma Code Web Page*, `http://jeffersonlab.github.io/chroma/`.
104. Bálint Joó, Dhiraj D. Kalamkar, Karthikeyan Vaidyanathan, Mikhail Smelyanskiy, Kiran Pamnany, Victor W. Lee, Pradeep Dubey, William Watson, *Lattice QCD on Intel® Xeon Phi^{TM} coprocessors*, in *Supercomputing, Lecture Notes in Computer Science*, Vol. **7905**, edited by Julian Martin Kunkel, Thomas Ludwig, Hans Werner Meuer (Springer Berlin Heidelberg, 2013) pp. 40–54.
105. Jefferson Lab GitHub Projects, *QPhiX Library*, `https://github.com/jeffersonlab/qphix.git`.
106. Simon Heybrock, Bálint Joó, Dhiraj D. Kalamkar, Mikhail Smelyanskiy, Karthikeyan Vaidyanathan, Tilo Wettig, Pradeep Dubey, *Lattice QCD with domain decomposition on Intel® Xeon Phi^{TM} co-processors*, in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '14 (IEEE Press, Piscataway, NJ, USA, 2014) pp. 69–80.
107. Carleton DeTar, Douglas Doerfler, Steven Gottlieb, Ashish Jha, Dhiraj Kalamkar, Ruizi Li, Doug Toussaint, PoS **LATTICE2016**, 270 (2016) arXiv:1611.00728 [hep-lat].
108. Bálint Joó, Dhiraj D. Kalamkar, Thorsten Kurth, Karthikeyan Vaidyanathan, Aaron Walden, *Optimizing Wilson-Dirac operator and linear solvers for Intel® KNL*, in *High Performance Computing - ISC High Performance 2016 International Workshops, ExaComm, E-MuCoCoS, HPC-IODC, IXPUG, IWOPH, P^3MA, VHPC, WOPSSS, Frankfurt, Germany, June 19-23, 2016, Revised Selected Papers* (Springer International Publishing, 2016) pp. 415–427.
109. Peter Boyle, Michael Chuvelev, Guido Cossu, Christopher Kelly, Christoph Lehner, Lawrence Meadows, *Accelerating HPC codes on Intel(R) Omni-Path architecture networks: From particle physics to machine learning* (2017) arXiv:1711.04883 [cs.DC].
110. Peter A Boyle, PoS **LATTICE2016**, 013 (2017) arXiv:1702.00208 [hep-lat].
111. E. Strohmaier, H. Simon, J. Dongarra, M. Meuer, *Top 500 List, November 2018*, `https://www.top500.org/lists/2018/11` (2018).
112. NERSC, *Perlmutter Web Page*, `http://www.nersc.gov/systems/perlmutter/`.
113. ALCF, *Aurora*, `https://aurora.alcf.anl.gov/`.
114. OLCF, *Frontier: OLCF's Exascale Future*, `https://www.olcf.ornl.gov/2018/02/13/frontier-olcfs-exascale-future`.
115. ALCF, *ALCF Aurora 2021 Early Science Program: Data and Learning Call For Proposals*, `https://www.alcf.anl.gov/alcf-aurora-2021-early-science-program-data-and-learning-call-proposals`.
116. T. Kurth, S. Treichler, J. Romero, M. Mudigonda, N. Luehr, E. Phillips, A. Mahesh, M. Matheson, J. Deslippe, M. Fatica, Prabhat, M. Houston, *Exascale deep learning for climate analytics*, arXiv:1810.01993 [cs.DC] (2018).
117. Intel Corporation, *Intel unveils strategy for state-of-the-art artificial intelligence*, `https://newsroom.intel.com/news-releases/intel-ai-day-news-release/`.
118. Peter A. Boyle, Comput. Phys. Commun. **180**, 2739 (2009).
119. A.V. Pochinsky, *Möbius domain wall fermion inverter*, `http://www.mit.edu/avp/mdwf`.
120. H. Carter Edwards, Christian R. Trott, Daniel Sunderland, J. Parallel Distrib. Comput. **74**, 3202 (2014).
121. B. Joó, *mg_proto: a prototype multi-grid library for QCD*, `https://github.com/jeffersonlab/mg_proto`.
122. Peter Boyle, Azusa Yamaguchi, Guido Cossu, Antonin Portelli, *Grid: A next generation data parallel C++ QCD library* arXiv:1512.03487 [hep-lat] (2015).
123. Xiao-Yong Jin, James C. Osborn, PoS **ICHEP2016**, 187 (2016) arXiv:1612.02750 [hep-lat].
124. Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre G.R. Day, Clint Richardson, Charles K. Fisher, David J. Schwab, *A high-bias, low-variance introduction to machine learning for physicists* arXiv:1803.08823 [physics.comp-ph] (2018).
125. Ian Goodfellow, Yoshua Bengio, Aaron Courville, *Deep Learning* (MIT Press, 2016) `http://www.deeplearningbook.org`.
126. M.S. Albergo, G. Kanwar, P.E. Shanahan, Phys. Rev. D **100**, 034515 (2019) arXiv:1904.12072 [hep-lat].
127. Phiala E. Shanahan, Daniel Trewartha, William Detmold, *Machine learning action parameters in lattice quantum chromodynamics* arXiv:1801.05784 [hep-lat] (2018).
128. Akinori Tanaka, Akio Tomiya, *Towards reduction of autocorrelation in HMC by machine learning* arXiv:1712.03893 [hep-lat] (2017).
129. Lei Wang, Phys. Rev. E **96**, 051301 (2017) arXiv:1702.08586 [physics.comp-ph].
130. Stefan Beyl, Florian Goth, Fakher F. Assaad, Phys. Rev. B **97**, 085144 (2018) arXiv:1708.03661 [cond-mat.str-el].
131. Xiao Yan Xu, Yang Qi, Junwei Liu, Liang Fu, Zi Yang Meng, Phys. Rev. B **96**, 041119 (2017) arXiv:1612.03804 [cond-mat.str-el].
132. Boram Yoon, *Estimation of matrix trace using machine learning* arXiv:1606.05560 [stat.ML] (2016).
133. Boram Yoon, *Machine learning estimators for lattice QCD observables* arXiv:1807.05971 [hep-lat] (2018).

134. J. Carrasquilla, R.G. Melko, Nat. Phys. **13**, 431 (2017) arXiv:1605.01735 [cond-mat.str-el].
135. G. Torlai, R.G. Melko, Phys. Rev. B **94**, 165134 (2016) arXiv:1606.02718 [cond-mat.stat-mech].
136. K. Chng, J. Carrasquilla, R.G. Melko, E. Khatami, Phys. Rev. **X7**, 031038 (2017) arXiv:1609.02552 [cond-mat.str-el].
137. Chian-De Li, Deng-Ruei Tan, Fu-Jiun Jiang, Ann. Phys. **391**, 312 (2018) arXiv:1703.02369 [cond-mat.dis-nn].
138. Sebastian Johann Wetzel, Manuel Scherzer, Phys. Rev. B **96**, 184410 (2017) arXiv:1705.05582 [cond-mat.stat-mech].
139. Andrei Alexandru, Paulo F. Bedaque, Henry Lamm, Scott Lawrence, Phys. Rev. D **96**, 094505 (2017) arXiv:1709.01971 [hep-lat].
140. Yuto Mori, Kouji Kashiwa, Akira Ohnishi, Phys. Rev. D **96**, 111501 (2017) arXiv:1705.05605 [hep-lat].
141. Yuto Mori, Kouji Kashiwa, Akira Ohnishi, Prog. Theor. Exp. Phys. **2018**, 023B04 (2018) arXiv:1709.03208 [hep-lat].
142. John Preskill, Quantum **2**, 79 (2018).
143. Joseph Carlson, David J. Dean, Morten Hjorth-Jensen, David Kaplan, John Preskill, Kenneth Roche, Martin J. Savage, Matthias Troyer, *Quantum computing for theoretical nuclear physics* (2018), Institute for Nuclear Theory report 18-008.