FERMILAB-CONF-12-120-CD

Fermilab multicore and GPU-accelerated clusters for lattice QCD

D Holmgren, N Seenu, J Simone, A Singh

Fermi National Accelerator Laboratory, Batavia, Illinois, USA

E-mail: `djholm@fnal.gov, nirmal@fnal.gov, simone@fnal.gov, amitoj@fnal.gov`

Abstract. As part of the DOE LQCD-ext project, Fermilab designs, deploys, and operates dedicated high performance clusters for lattice QCD (LQCD) computations. We describe the design of these clusters, as well as their performance and the benchmarking processes that were used to select the hardware and the techniques used to handle their NUMA architecture. We discuss the design and performance of a GPU-accelerated cluster that Fermilab deployed in January 2012. On these clusters, the use of multicore processors with increasing numbers of cores has contributed to the steady decrease in price/performance for these calculations over the last decade. In the last several years, GPU acceleration has led to further decreases in price/performance for ported applications.

1. Introduction

Lattice QCD (LQCD) uses large scale numerical simulations to study the strong interactions between quarks mediated by gluons (quantum chromodynamics, or QCD). In the United States, most lattice QCD theorists are members of the USQCD collaboration. Members of USQCD study Standard Model QCD problems in high energy and nuclear physics, as well as various theories beyond the standard model.

The Office of Science of the U.S. Department of Energy, through the Office of High Energy Physics and the Office of Nuclear Physics, has strongly supported LQCD through the SC LQCD (FY06-FY09), SC LQCD-ext (FY10-FY14) and LQCD ARRA (FY09-FY13) hardware projects. These projects have funded the purchase and operations of clusters dedicated to LQCD computations. Time on these cluster resources is allocated by USQCD. Together with the Office of Advanced Scientific Computing Research, the high energy and nuclear physics program offices have also supported LQCD software development through the SciDAC (Scientific Discovery through Advanced Computing) [1] and SciDAC-2 [2] programs.

Fermilab currently operates four LQCD clusters. Three of these are conventional Infiniband clusters based on multicore processors, and the fourth is a GPU-accelerated cluster. Both multicore and GPU architectures have improved the price/performance of LQCD clusters.

2. Computational Requirements

Inversion of the Dirac operator (*Dslash*) is the most computationally intensive task of LQCD codes.[3] Using the improved staggered action (*asqtad*) as an example, during each iteration of the *asqtad Dslash* inverter, eight sets of SU(3) matrix-vector multiplies occur using nearest and next-next-nearest neighbor spinors. The 3x3 matrices describe gluons, and the 3x1 spinors

Operated by Fermi Research Alliance, LLC under Contract No. De-AC02-07CH11359 with the United States Department of Energy.

describe quarks. For a calculation using multiple computers, ideally these floating point operations overlap with the communications of the hyper-surfaces of the sub-lattices held on neighboring nodes. The low ratio of floating point operations to bytes of memory read or written during *Dslash* execution (in single precision, an SU(3) matrix-vector multiply requires 66 floating point operations and has input and output operands totaling 120 bytes) results in a strong demand for memory bandwidth. Effective parallel machine designs for LQCD calculations must balance floating point performance, memory and network bandwidths, and network latency. On any cluster, one of these will be the limiting factor determining performance for a given problem size.

Inversion of the *Dslash* operator is computationally equivalent to inverting a very large, very sparse matrix. For example, a $48^3 \times 144$ problem using the improved staggered action, typical of the size of current simulations, results in a complex matrix with 47.8 million $\times$ 47.8 million elements, of which 1.15 billion are non-zero (about one in every 2 million). Iterative techniques such as "conjugate gradient" are used to perform these inversions. Specific LQCD simulations, such as the computation of a decay constant, require many TFlop/sec-years of calculations using large-scale parallel machines. LQCD codes employ MPI or other message passing libraries for communications. Networks such as Infiniband provide the required high bandwidth and low latency.

Figure 1 below represents a typical LQCD cluster. A cascaded Infiniband network is used, with a large "spine" switch connected to "leaf" switches. All of the Fermilab LQCD clusters described here implement this design. On accelerated clusters, each of the nodes shown contains one to four GPUs. As the Infiniband fabric contributes greatly to the cost of these clusters, when tolerated, over-subscription is used to reduce the total number of Infiniband switch ports and cables. The over-subscription of such a fabric equals the total number of network links connecting leaf to spine switches, divided into the number of worker nodes. Currently available leaf switches have 36 switch ports. On a fabric with 2:1 over-subscription, such a switch will have 8 uplinks to the spine switch, and 24 connections to work nodes. Since many simulations may require 24 or fewer worker nodes, a substantial fraction of jobs run on such a cluster will have all Infiniband communications "local" to one of the leaf switches.

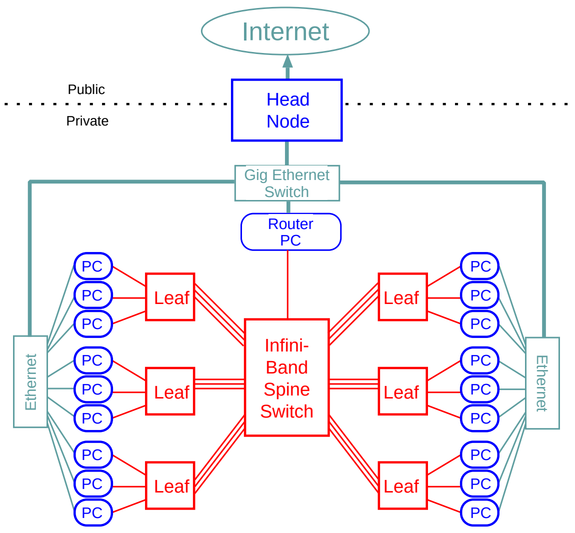

Figure 1. Typical LQCD Infiniband cluster schematic. Infiniband spine and leaf switches are shown in red. LQCD parallel codes execute on the worker nodes (labeled "PC"), under the direction of a batch scheduler running on the head node.

3. LQCD Clusters at Fermilab

Table 1 lists the currently operational Fermilab LQCD clusters. The Kaon, JPsi, Ds, and Dsg clusters were installed, respectively, in 2006, 2008, 2010, and 2012. The performance values listed are sustained GFlops per worker node on LQCD parallel applications using 128 cores, where the figure gives the performance average of *improved staggered* and *domain wall fermion* actions. For the Dsg cluster, in addition to the performance of the worker nodes when using only the Intel E5630 conventional processors, the performance of GPU-accelerated code is given in italics. In general, GPU performance varies widely across LQCD applications, with observed acceleration compared to conventional nodes of two to fifteen on LQCD codes; the value shown for the Dsg cluster corresponds to *isotropic clover* code that utilizes four GPUs in parallel using two Dsg worker nodes (each Dsg worker node contains two GPUs).

Table 1. Fermilab LQCD clusters

Name	**CPU**	**Nodes**	**Cores**	**Performance**
Kaon	Dual 2.0 GHz Dual-Core Opteron 240	300	1200	4.3 GF/node
JPsi	Dual 2.1 GHz Quad-Core Opteron 2352	856	6848	9.8 GF/node
Ds	Quad 2.0 GHz 8-Core Opteron 6128	421	13472	50.9 GF/node
Dsg	Dual 2.53 GHz Quad-Core Intel E5630	76	608	23.1 GF/node
	Dual NVIDIA M2050	*152*	*68096*	*93.4 GF/GPU*

4. Multicore Processors and LQCD

LQCD codes scale very well on multicore processor clusters. Most implementations rely on assigning an MPI rank to each core of a worker node, rather than using a hybrid approach such as a mixture of OpenMP threading and MPI message passing. All currently available multi-socket multicore commodity computers have NUMA architectures. Since memory bandwidth limits LQCD application performance, binaries should either be built using NUMA-aware MPI libraries, or invoked with NUMA controls. Figure 2 shows the performance degradation that occurs on LQCD codes that use non-local memory. In this example, the performance of the conjugate gradient inverter of both scalar (one process) and parallel (four process) versions of LQCD code is shown as a function of the local lattice size. In both cases the red curve corresponds to the execution of the code using non-local memory, and the blue curve corresponds to the correct use of local memory. Fermilab provides all users with *numactl*-based shims that force non-NUMA-aware MPI launchers to bind MPI ranks to cores and to set strict local memory allocation policies.

As core counts per node have increased, LQCD code efficiencies have risen as well. This is a consequence of having multiple MPI ranks per worker node. Any communications among these local ranks has higher bandwidth and lower latency than similar communications between MPI ranks running on different nodes. "Weak scaling" describes the relative performance of simulations as the number of MPI ranks are increased for problems of proportionally greater lattice sizes, keeping the local volume per rank constant. Figure 3 shows the ratio of the performance per rank on a 128-rank problem to the performance per rank of the problem that uses exactly the number of cores contained in a single node. These data were measured on a sequence of LQCD clusters from 2005 until 2011 at Fermilab and at Jefferson Lab. The single and dual core per node clusters were Intel-based, and the quad core, eight core, and thirty-two cores per node clusters were AMD-based. An additional bonus from the increase in core counts per node has been that, for fixed problem sizes, newer clusters with higher core counts per node require fewer Infiniband switch ports and network interfaces.

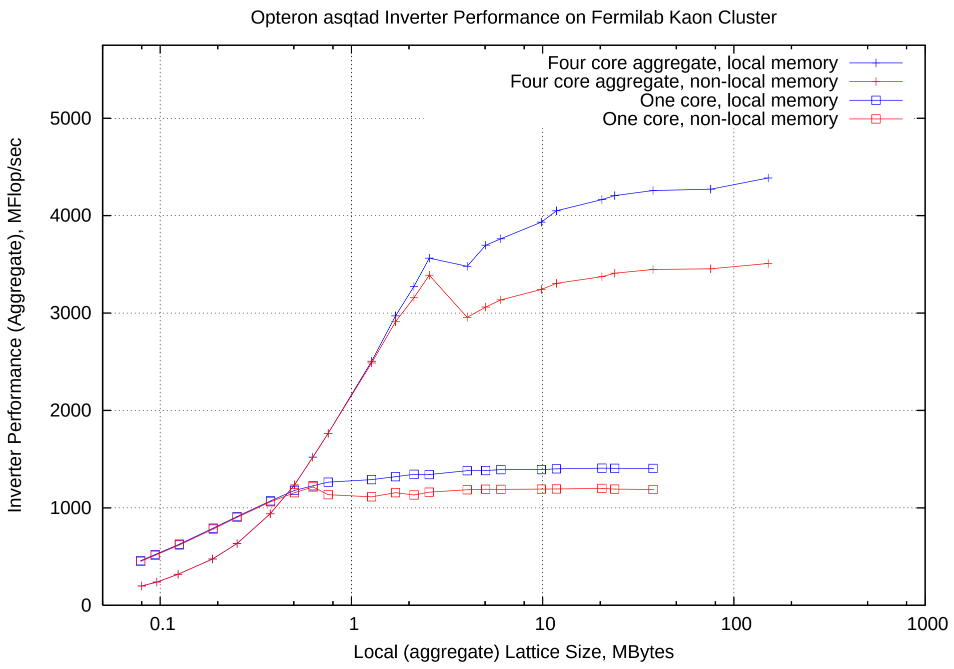

Figure 2. Performance of NUMA-aware (“local memory”) and non-aware (“non-local memory”) binary invocations

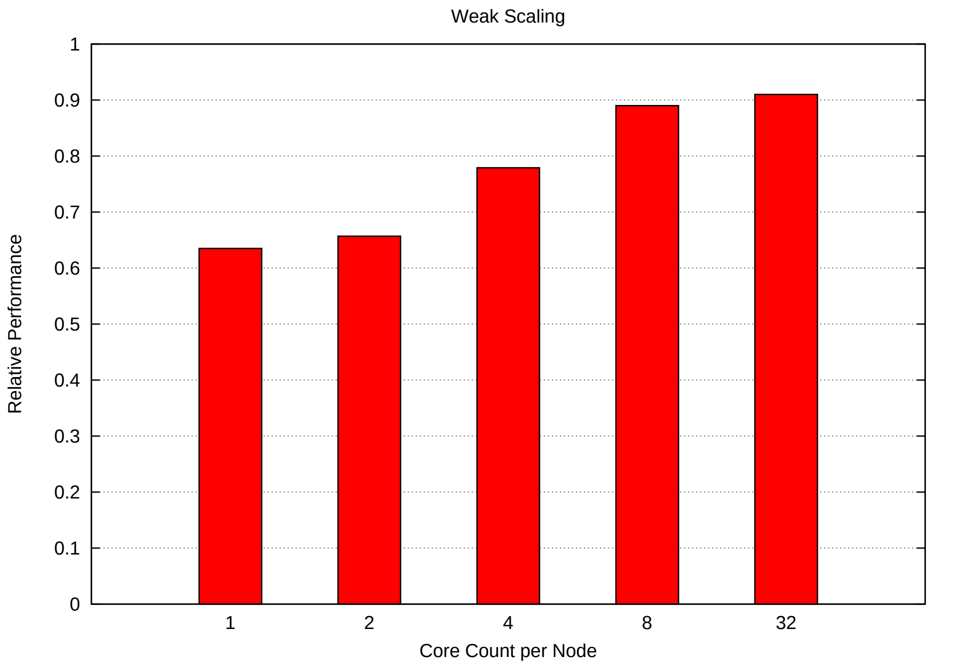

Figure 3. Weak scaling as a function of the number of cores per worker node

5. Benchmarking

Fermilab awards purchase contracts to computer vendors according to a best value process. Price/performance, in dollars per sustained MFlop, is a key component in determining the value of a vendor's proposed system. For each of the JPsi[4], Ds[5], and Dsg[6] purchases, vendors were provided with benchmark packages that reported aggregate single-node performance using all available cores of LQCD applications and micro-benchmarks. The benchmark package for the Dsg cluster included both GPU-accelerated and non-accelerated applications. The benchmarks can be run on systems with a wide range of processor sockets per node and cores per socket. The included MPI launcher automatically tailors execution to match the NUMA design.

6. GPU-Accelerated LQCD

An early use of GPU acceleration on an LQCD application was published in 2005[7], in which the authors described coding in OpenGL with Cg. More recently, Barros *et al.* at Boston University implemented a Wilson-Dirac operator in 2007 using NVIDIA's CUDA programming model[8], and reported[9] about a ten-fold speedup over contemporary conventional x86 processors. The work at Boston University evolved under the SciDAC-2 program into the *QUDA* framework. Initially[10] this framework included Wilson-Dirac code that ran on a single GPU, with performance optimizations that included mixed-precision algorithms and reduced representations of $SU(3)$ matrices. Reduced representations exploit symmetry in these data structures such that fewer operands are stored and moved, thus lowering the required memory bandwidth during parts of the calculation.

Further refinements to *QUDA* have included support for the clover, twisted-mass, asqtad, HISQ, and domain wall fermion actions[11], and support for running codes across many GPUs in parallel[12]. Parallel GPU capabilities are required to support large problems and to exploit strong scaling to minimize wall clock times for gauge configuration generation.

7. The Dsg GPU-Accelerated Cluster

In 2011, Fermilab designed and purchased a GPU-accelerated cluster with optimal strong-scaling performance on calculations that require many GPUs operating in parallel. To meet this requirement, vendor-proposed host systems were restricted to those that provided sufficient PCI Express bandwidth to support the installed GPUs (sixteen PCIe gen2 lanes per GPU) and quad-data-rate Infiniband (eight PCIe gen2 lanes per Infiniband host channel adapter, with one host channel adapter per par of GPUs).

The purchase of the Dsg cluster was awarded to Hewlett-Packard. The GPU host machines are model SL390s G7 blade servers; each server has two conventional processor sockets, populated with Intel "Westmere" E5630 4-core processors. The hosts contain 48 GBytes of system memory. Each host houses a pair of NVIDIA Tesla M2050 GPUs; each M2050 has 448 cores, and has 1 TFlop/sec peak single precision performance and supports hardware double precision. The M2050 GPUs each have 3 GBytes of ECC-capable memory. Dsg consists of a total of 76 hosts, with a total of 152 GPUs. A full bisection bandwidth quad data rate Infiniband fabric connects all hosts.

The performance of Dsg GPUs varies with the specific LQCD application. In production running, users have reported application-dependent speed-ups of between 2.1 and 13.3, comparing Ds node-hours to Dsg node-hours for performing equivalent calculations. Large-scale ($96^3 \times 192$ and $64^3 \times 96$) gauge configuration generation has been demonstrated on this cluster using parallel runs on 128 GPUs. The Dsg cluster has been in production since March 1, 2012.

8. Decreases in Price/Performance for LQCD Clusters

Fermilab has tracked LQCD price/performance data since 1995. Figure 4 shows the price/performance observed on production Infiniband clusters deployed since 2005. The fit uses data points between 2005 and 2010 and has a halving time of 1.613 years. Between 1995 and 2004, clusters based on Myrinet switched networks, on gigabit Ethernet toroidal mesh networks,

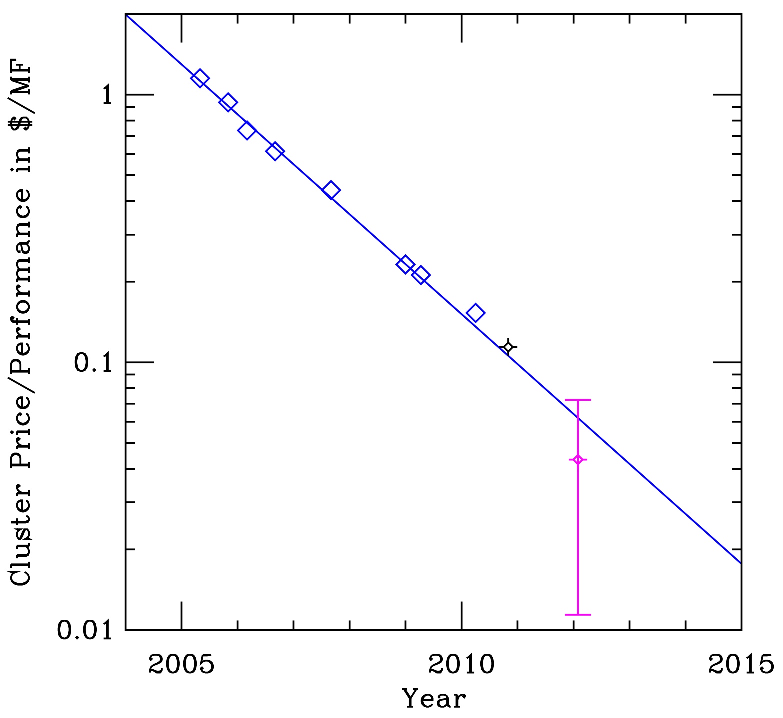

Figure 4. Price/performance, in dollars per sustained LQCD TFlop/sec. The blue diamonds are clusters purchased at Fermilab or Jefferson Lab; the left two are clusters based on single core computers, and the others are clusters based, respectively, on 2, 4, 8, 8, 8, and 8 cores per node. The black star is the Ds cluster with 32 cores per node. The pink range shows the cluster equivalent price/performance for codes running on Dsg that achieve speed-ups of between 2.1 and 13.3.

and on switched fast Ethernet networks exhibited a faster halving time of 1.25 years; these were much smaller clusters with consequently less expensive network fabrics. Higher core counts per node have compensated for the cessation of increasing processor clock speeds that occurred in the middle of the last decade. Based on data from Dsg, the use of GPUs results in price/performance that significantly surpasses the long term trend.

9. Summary

Fermilab operates a number of conventional and GPU-accelerated clusters based on Infiniband interconnects dedicated to lattice QCD computations. Historically, the emergence of multicore platforms with increasing core counts per node has benefited LQCD codes in terms of cost per unit of performance. Since the performance of parallel computations depends on the slowest processing element, control of NUMA-related run time behaviors, such as locking processes to cores and enforcing local memory allocation policies, is critical to achieving both consistent performance as well as the highest possible performance. For those LQCD applications that have been ported to use GPUs, significant improvement in cost effectiveness has been observed on the latest accelerated cluster at Fermilab.

The Fermi National Accelerator Laboratory is operated by Fermi Research Alliance, LLC under Contract No. De-AC02-07CH11359 with the United States Department of Energy. This contribution is FERMILAB-CONF-12-120-CD.

[1] National infrastructure for lattice gauge computing URL `http://www.scidac.gov/HENP/HENP_QCD.html`

[2] Studying the theory of quarks and gluons formulated on a space-time lattice using lattice quantum chromodynamics URL `http://www.scidac.gov/physics/quarks.html`

[3] Holmgren D J 2005 *Nuclear Physics B (Proc. Suppl.)* **140** 183 (*Preprint* `arXiv:hep-lat/0410049`)

[4] Benchmark package for the jpsi cluster URL `http://www.usqcd.org/fnal/fermi_bench.tar.bz2`

[5] Benchmark package for the ds cluster URL `http://www.usqcd.org/fnal/fermi_bench_fy10.tar.bz2`

[6] Benchmark package for the dsg cluster URL `http://lqcd.fnal.gov/fermi_bench_fy11.tar.bz2`

[7] Egri G I, Fodor Z, Hoelbling C, Katz S D, Nogradi D and Szabo K K 2007 *Comput. Phys. Commun.* **177** 631 (*Preprint* `arXiv:hep-lat/0611022`)

[8] NVIDIA CUDA web page URL `http://www.nvidia.com/object/cuda_home_new.html`

[9] Barros K, Babich R, Brower R, Clark M A and Rebbi C 2008 *PoS (Lattice2008)* 045 (*Preprint* `arXiv:hep-lat/0810.5365`)

[10] Clark M A, Babich R, Barros K, Brower R C and Rebbi C 2010 *Comput. Phys. Commun.* **181** 1517 (*Preprint* `arXiv:hep-lat/0911.3191`)

[11] Babich R, Clark M A and Joo B 2010 *Proceedings of Supercomputing'10* (*Preprint* `arXiv:hep-lat/1011.0024`)

[12] Babich R, Clark M A, Joo B, Shi G, Brower R C and Gottlieb S 2011 *Proceedings of Supercomputing'11* (*Preprint* `arXiv:hep-lat/1109.2935`)