FERMILAB-CONF-23-025

PROCEEDINGS OF SCIENCE

Two-link Staggered Quark Smearing in QUDA

Steven Gottlieb,[a] **Hwancheol Jeong**[a,*] **and Alexei Strelchenko**[b]

[a] *Department of Physics, Indiana University, Bloomington, Indiana 47405, USA*

[b] *Scientific Computing Division, Fermi National Accelerator Laboratory, Batavia, Illinois, 60510, USA*

E-mail: sg@indiana.edu, sonchac@gmail.com, astrel@fnal.gov

Gauge covariant smearing based on the 3D lattice Laplacian can be used to create extended operators that have better overlap with hadronic ground states. For staggered quarks, we make use of two-link parallel transport to preserve taste properties. We have implemented the procedure in QUDA. We present the performance of this code on the NVIDIA A100 GPUs in Indiana University's Big Red 200 supercomputer and on the AMD MI250X GPUs in Oak Ridge Leadership Computer Facility's (OLCF's) Crusher and discuss its scalability. We also study the performance improvement from using NVSHMEM on OLCF's Summit. Reusing precomputed two-link products for all sources and sinks, it reduces the total smearing time for a baryon correlator measurement by a factor of 100–120 as compared with the original MILC code and reduces the overall time by 60–70%.

The 39th International Symposium on Lattice Field Theory (Lattice2022),
8-13 August, 2022
Bonn, Germany

*Speaker

1. Introduction

Lattice QCD calculations require operators that have a strong overlap with particular hadronic states. For example, lattice QCD calculations that study the low energy hadron spectrum benefit from extended operators that have better overlap with the ground state than local operators at a single lattice site. Decomposing a correlation function in terms of energy eigenstates, a two-point correlation function $C(t) = \sum_{\mathbf{x}} \left\langle O(t, \mathbf{x})\, O^\dagger(0, \mathbf{0}) \right\rangle$ can be expressed as

$$C(t) = \sum_n |\, \langle 0|O|n\rangle|^2\, e^{-E_n t}\,, \tag{1}$$

where E_n is the energy of the n-th energy eigenstate. We can extract some low energy properties from it, including the mass from the ground state contribution. If the configuration is gauge-fixed, one can use extended operators without concern about parallel transport; however, by using gauge-covariant smearing of the source one can avoid having to fix the gauge.

There are two popular kinds of gauge covariant smearing: Jacobi smearing and Gaussian smearing. Jacobi smearing is an iterative version of Wuppertal smearing which takes the three-dimensional scalar propagator as a smeared source [1–4]. The smeared source follows the exponential distribution on a free gauge configuration. Alternatively, Gaussian smearing applies a hopping operator iteratively to the given source. The resulting smeared source follows the Gaussian distribution on a free gauge configuration [2, 3, 5].

In this paper, we are interested in a variant of Gaussian smearing, which replaces the hopping operator with the three-dimensional lattice Laplacian operator for staggered quarks [6]. To preserve the taste symmetry of staggered quarks, the Laplacian should extend to the next-to-nearest-neighbor sites. We define the two-link products joining the next-to-nearest-neighbor sites and call this smearing method two-link staggered quark smearing.

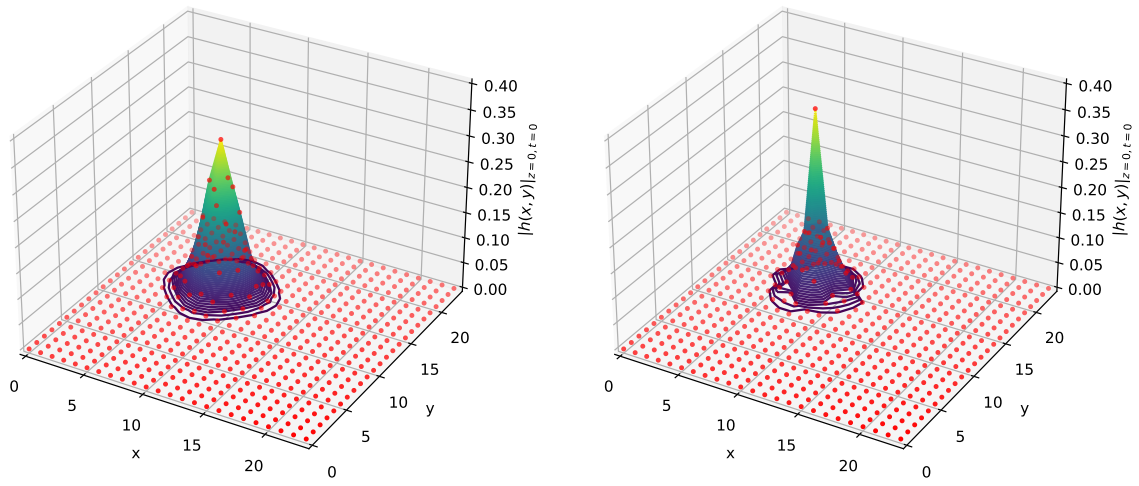
The MILC code with which we started was doing two-site parallel transport by applying single-site parallel transport twice in the same direction. This requires two communications and two matrix-vector multiplies per direction. We found that this smearing was taking an inordinate amount of time when done on the CPU. The situation is even worse when other parts of the calculation run on the GPU, because one allocates only one MPI rank per GPU, requiring multi-threading using OpenMP to use more than one CPU core per rank. Hence, we have implemented the procedure in QUDA [7–9]. The exascale computers in the U.S. all make use of GPUs, so more and more of our projects will make use of this timely addition to our codes.

Section 2 briefly describes the two-link staggered quark smearing. Section 3 describes our GPU implementation and algorithmic improvement. In Secs. 4 and 5, we present benchmark results on some (recent or latest) NVIDIA and AMD GPUs, respectively. We also apply our QUDA smearing routine to a baryon correlator measurement in Sec. 6 to show how our code performs in a production job. We summarize our conclusions in Sec. 7.

2. Two-link staggered quark smearing

Let us consider a hopping operator H defined by

$$H(x, y) = \sum_{\mu=1}^{3} U_\mu(x)\, \delta_{x+\hat{\mu},\, y} + U_\mu^\dagger(x - \hat{\mu})\, \delta_{x-\hat{\mu},\, y}\,, \tag{2}$$



(a) Free gauge ($U = 1$) **(b)** HISQ gauge

Figure 1: Example distributions of a point source smeared by the two-link staggered quark smearing on a free gauge configuration (left) and a HISQ gauge configuration (right). Red data points represent norms of smeared quark field at $(x, y, z, t) = (x, y, 0, 0)$. Contours are drawn by connecting these points.

where x, y are space-time coordinates and $U_\mu(x)$ is the gauge field. An iterative Gaussian smearing operation to a quark field $\psi(y)$ can be written as

$$\widetilde{\psi} = C(1 + \alpha H)^n \psi \,, \tag{3}$$

where $C, \alpha \in \mathbb{R}$ and $n \in \mathbb{Z}$ are tuning parameters. For the unit gauge configuration $U_\mu(x) = 1$, the smeared field $\widetilde{\psi}(x)$ approaches the Gaussian distribution as the iteration count n grows [2, 3, 5].

The 3D lattice Laplacian ∇^2 is defined by

$$\begin{aligned}\nabla^2(x,y) &= \sum_{\mu=1}^{3} \left[U_\mu(x)\,\delta_{x+\hat{\mu},\,y} + U_\mu^\dagger(x-\hat{\mu})\,\delta_{x-\hat{\mu},\,y}\right] - 6\,\delta_{x,y} \\ &= H(x,y) - 6\,\delta_{x,y}\,. \end{aligned} \tag{4}$$

Replacing $U_\mu(x)$ with the *two-link product* $V_\mu(x) \equiv U_\mu(x)U_\mu(x+\hat{\mu})$ and adjusting the coordinate, we define (ignoring factors of a) the two-link 3D lattice Laplacian ∇^2_{two}:

$$4\nabla^2_{\text{two}}(x,y) \equiv \sum_{\mu=1}^{3} \left[V_\mu(x)\,\delta_{x+2\hat{\mu},\,y} + V_\mu^\dagger(x-2\hat{\mu})\,\delta_{x-2\hat{\mu},\,y}\right] - 6\,\delta_{x,y} \tag{5}$$

$$= \sum_{\mu=1}^{3} \left[U_\mu(x)\,U_\mu(x+\hat{\mu})\,\delta_{x+2\hat{\mu},\,y} + U_\mu^\dagger(x-\hat{\mu})\,U_\mu^\dagger(x-2\hat{\mu})\,\delta_{x-2\hat{\mu},\,y}\right] - 6\,\delta_{x,y}\,. \tag{6}$$

Note that ∇^2_{two} preserves taste properties for staggered quarks.

If we rewrite Eq. (3) in terms of the Laplacian ∇^2,

$$\widetilde{\psi} = C(1 + \alpha(\nabla^2 + 6))^n \psi \equiv C'(1 + \alpha'\nabla^2)^n \psi \,, \tag{7}$$

where $C' \equiv C(1+6\alpha)^n$ and $\alpha' \equiv \dfrac{\alpha}{1+6\alpha}$. Now, we define the two-link staggered quark smearing as

$$\widetilde{\psi} = \left(1 + \frac{\sigma}{n}\nabla^2_{\text{two}}\right)^n \psi\,, \tag{8}$$

where σ and n are tuning parameters. This is a taste-preserving gauge covariant smearing for staggered quarks. Figure 1 shows the distributions of a point source after this smearing is applied on the free gauge configuration (Fig. 1a) and a HISQ gauge configuration (Fig. 1b). Although the latter is distorted by the existence of the gauge field, these results imply we can get a better overlap with hadronic ground states with a suitable choice of tuning parameters σ and n [2, 4, 10]. Using a gauge-smeared link in the place of U_μ can relax the distortion as well as increase the overlap with the ground state [11]. There are also other approaches that enhance the overlap with some good properties [5, 12].

3. GPU implementation

Before this project began, the MILC code library [13] provided a CPU based routine for the two-link staggered quark smearing. In need of the GPU equivalent, we have implemented it in QUDA. We have also added an interface for this QUDA smearing in the MILC code, which runs all QUDA benchmarks in this paper.[1]

We also improved upon the CPU algorithm in the MILC code. Instead of carrying out two consecutive parallel transports (as in Eq. (6)) at each smearing iteration, we precompute the two-link product V_μ in advance of smearing, and every iteration just loads it from the memory and uses Eq. (5). This two-link product can even be reused for different sources or sinks. In the GPU algorithm, we perform the smearing with significantly less memory traffic, fewer floating point operations, and less communication between MPI ranks.

In Fig. 2, we compare the smearing time required by the MILC code to that of our QUDA code using all CPUs and GPUs, respectively, on Big Red 200 at Indiana University. Big Red 200 is similar to Perlmutter at NERSC in that it has a CPU partition in which each node contains two AMD 64-core EPYC 7742 processors and a GPU partition in which each node contains an AMD processor and four NVIDIA A100 GPUs. Big Red 200 has a Slingshot 10 network, whereas Perlmutter was upgraded from Slingshot 10 to 11. We ran benchmarks for four different lattice volumes on either two CPU or two GPU nodes. For production jobs, we like to run with a local volume of at least 24^4 per GPU and 32^4 or larger is preferred. Thus, these tests with a fixed number of nodes were not designed for maximum efficiency. In the case of QUDA runs, we also measure the second source smearing that reuses the precomputed two-link product. The first source smearing includes the computation of the two-link product and n iterations of smearing, while the second source smearing does only n iterations of smearing. Typical computations make use of multiple sources and sinks on the same gauge configuration, so the second source smearing result is quite pertinent. The results show that the first source smearing by QUDA on the GPU takes from 0.79 to 0.15 times as long as the MILC code smearing on the CPU, and the second source smearing by QUDA takes from 0.16

[1] We are working on merging the QUDA code and the MILC code interface into the main (develop) branches of QUDA and the MILC code's GitHub repositories, respectively[7, 13].

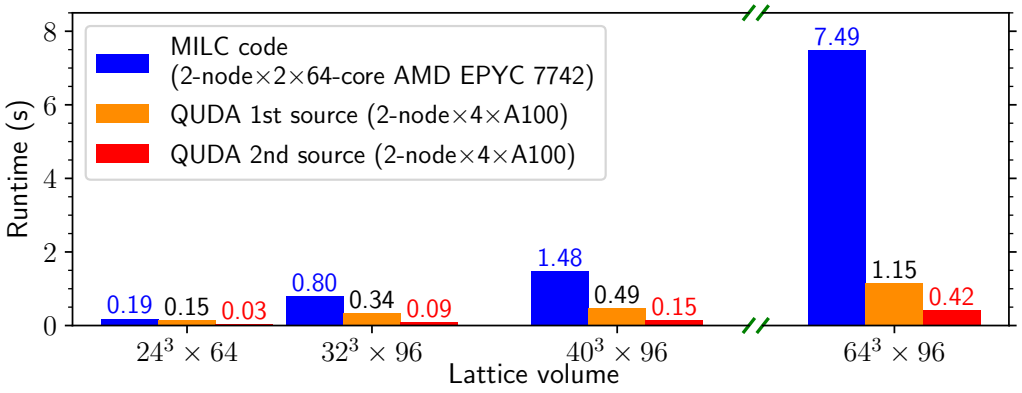


Figure 2: Time taken by smearing a source quark field with $n = 50$ two-link staggered quark smearing by the MILC code and QUDA. They are measured on two nodes of Big Red 200. The MILC code is run with one MPI rank per CPU core (total peak double precision FLOPS $\approx$ 14 TFLOPS), and QUDA is run with one MPI rank per GPU (total peak double precision FLOPS $\approx$ 80 TFLOPS). Lattices are divided by $(x, y, z, t) = (8, 8, 4, 1)$ for MILC code runs and $(x, y, z, t) = (2, 2, 2, 1)$ for QUDA runs. Note that splitting in t-dimension does not improve the performance of 3D Gaussian smearing applied to fixed time sources such as point sources or wall sources.

to 0.06 times as long as the MILC code. The improvement increases for larger lattice volumes, as the smaller cases are too small for the GPU code to run efficiently.

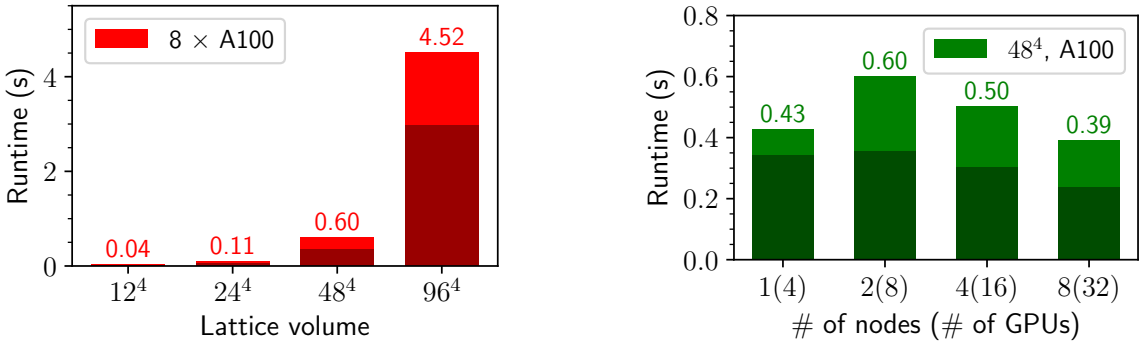
In this section, we have compared the maximum — from the point of view of using all available CPU/GPU resources — performances of the MILC code smearing and the QUDA smearing on two nodes of a computer with both CPU and GPU nodes. In practice, however, our primary goal is to implement the QUDA smearing in measurements running on GPU nodes. In this situation, smearing running on the CPU can become a severe bottleneck. Section 6 discusses an example.

4. Performance on NVDIA GPU

In this section, we report the performance of the two-link staggered quark smearing in QUDA (QUDA two-link smearing) on two NVIDIA GPU based systems. We smear three different color wall sources in succession. Only the first color source smearing computes the two-link product, while the others reuse it.

In Fig. 3, we measure the performance and scalability of the QUDA two-link smearing on NVIDIA A100 GPUs in Big Red 200. The total runtime includes one run of two-link computation (unshaded) and three $n = 50$ smearing iterations (shaded). We find that the single two-link computation takes around 10–40% of the time. This indicates the advantage of reusing the two-link product.

Figure 3a presents the smearing time by varying the lattice volume. While the lattice volume increases geometrically by a factor of 16, the total smearing time increases by factors of 2.75, 5.45, and 7.53, respectively. This indicates the computation has not been saturated yet up to the largest lattice volume, so its performance (FLOPS) would be better for a bigger lattice. Still, for small lattices, it would be a fraction of time compared to typical lattice simulation scales. Figure 3b presents the smearing time by varying the number of nodes and GPUs. Note that the two-node run



(a) Volume scalability

(b) Strong scalability

Figure 3: Time taken by smearing three color sources with $n = 50$ QUDA two-link smearing on NVIDIA A100 GPUs in Big Red 200. The unshaded region represents the time taken by the two-link computation, and the shaded region represents the time taken by $3 \times n$ iterations of smearing. Lattices are divided by $(x, y, z, t) = (2, 2, 1, 1)$ for 4 GPUs, $(2, 2, 2, 1)$ for 8 GPUs, $(4, 2, 2, 1)$ for 16 GPUs, and $(4, 4, 2, 1)$ for 32 GPUs.

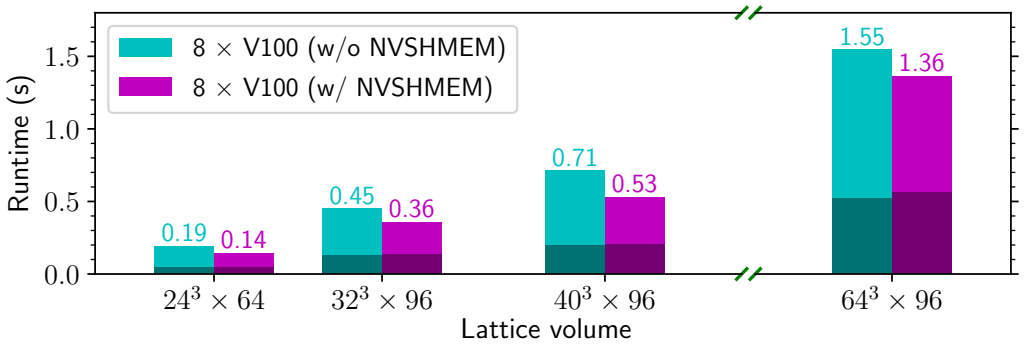
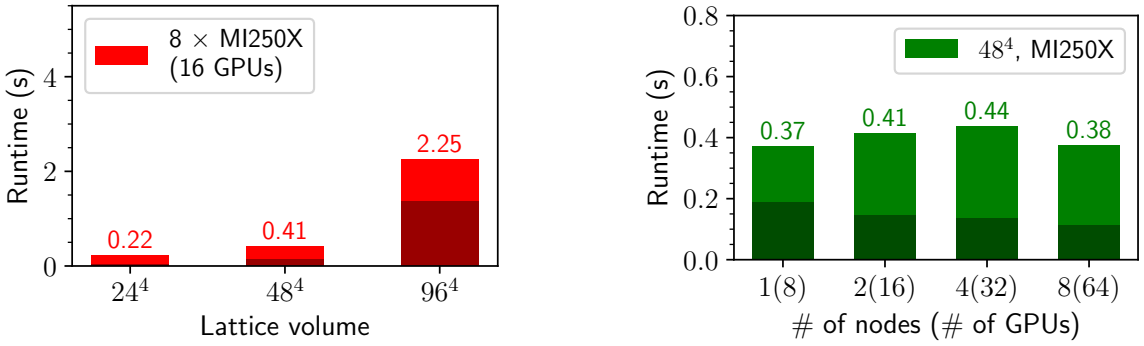


Figure 4: Time taken by smearing three color sources with $n = 50$ QUDA two-link smearing on NVIDIA V100 GPUs in Summit with/without NVSHMEM support enabled in QUDA. The unshaded (shaded) region represents the time taken by the two-link computation (smearing iterations). Here, we use two Summit nodes, but only four V100 GPUs out of six per node, because six is not suitable for dividing the spatial dimension of some representative lattices. All lattices are divided by $(x, y, z, t) = (2, 2, 2, 1)$.

is slower than the one-node run. Even the two-link computation taking three times longer. This implies the communication between off-nodes affects the performance significantly, and it is more prominent for the two-link computation. Excluding the one-node result, the performance improves as the number of nodes increases, but it scales very poorly. Figure 3a and 3b imply that this routine is a communication-intensive calculation.

The observation above suggests that the QUDA two-link smearing may perform better with a faster communication environment. Summit at OLCF supports NVIDIA NVSHMEM technology. NVSHMEM improves strong scaling of GPU operations by enabling direct communication between GPUs with shared memory space [14]. QUDA supports NVSHMEM [15]. Figure 4 shows the performance improvement of the QUDA two-link smearing on Summit by enabling NVSHMEM. Here, the two-link computation takes more than half of the total runtime. We find that NVSHMEM reduces the two-link computation time by around 30–50%.



(a) Volume scalability

(b) Strong scalability

Figure 5: Time taken by smearing three color sources with $n = 50$ QUDA two-link smearing on AMD MI250X GPUs on Crusher. Note that each MI250X contains two GCDs, so the actual number of GPUs used is twice of the number of MI250Xs. The unshaded (shaded) region represents the time taken by the two-link computation (smearing iterations). Lattices are divided by $(x, y, z, t) = (2, 2, 2, 1)$ for 8 GPUs, $(4, 2, 2, 1)$ for 16 GPUs, $(4, 4, 2, 1)$ for 32 GPUs, and $(4, 4, 4, 1)$ for 64 GPUs.

	Big Red 200	Crusher
GPU	4 × NVIDIA A100 (≈ 40 TFLOPS)	4 × AMD MI250X (=8 GPUs) (≈ 210 TFLOPS)
GPU Mem. BW	1555 GB/s	3277 GB/s
NIC	2 HPE × Slingshot-10 (200 Gbps)	4 × HPE Slingshot-11 (800 Gbps)

Table 1: GPU and NIC specification of Big Red 200 and Crusher [17–20]. FLOPS numbers represent the double precision peak performance.

5. Performance on AMD GPU

QUDA also supports HIP on AMD ROCm platform [16], allowing it to run on AMD GPUs. In this section, we report the performance of the QUDA two-link smearing on Crusher at OLCF, an AMD GPU-based system containing hardware identical to that on Frontier. As in Sec. 4, we smear three different color wall sources in order, where the two-link product is computed only at the first smearing and reused for others.

In Fig. 5, we measure the performance and scalability of the QUDA two-link smearing on the AMD MI250X GPUs in Crusher in the same manner as in Fig. 3.[2] Figure 5a and 5b plot the volume and strong scalabilities, respectively. The y-axis ranges are the same as in Fig. 3a and 3b for easy comparison. The smearing iteration performs approximately twice faster here with MI250X compared to that with A100. This result agrees with our expectation that the bottleneck of this

[2]These plots are updated from the ones we presented in the poster, where we observed an abnormal slowdown in the two-link computation on AMD GPUs. It turned out the culprit was a HIP API which was not directly related to the two-link computation. We found a way to avoid this problem and reran the benchmark. We are also working on resolving the issue.

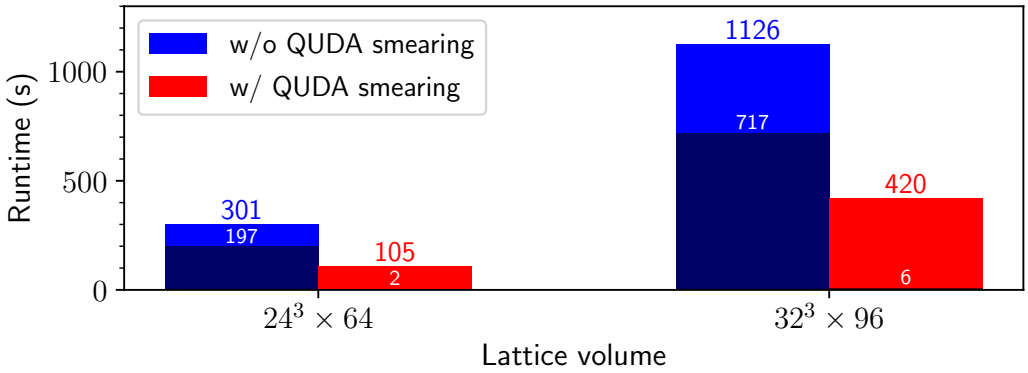


Figure 6: Total time taken by a baryon correlator measurement employing 72 source/sinks which are smeared by the $n = 30$ QUDA two-link smearing. The measurement is carried out on two nodes of Big Red 200's GPU partition using four CPU cores and GPUs per node. With or without the QUDA two-link smearing enabled, all other QUDA-supported calculations are performed on the GPU. The shaded region represents the total smearing time.

calculation is the GPU memory bandwidth because its compute-to-communication ratio is similar to a typical dslash routine (see Eq. (5)), and a MI250X has twice faster memory bandwidth than A100 (see Table 1). On the other hand, the two-link computation performs similarly or slower on the MI250X compared to the A100. The only exception is the 96^4 lattice result, where we observe the expected twice faster performance. Regarding the scalability, we observe a poor scaling as we observed from the A100 GPU.

6. Application: Baryon correlator measurement

Our baryon correlator calculations include many sources and sinks. Figure 6 shows the total runtime taken by an example baryon correlator measurement. Most parts of the calculation were already implemented in QUDA. With the QUDA smearing disabled, the smearing is carried out on the CPU by the MILC code smearing routine while other major calculations are carried out on the GPU by QUDA. Note that here we use only the same number of CPUs as of GPUs. The result shows that the source/sink smearing takes up around 60–70% of the total measurement time when it is run on the CPU. However, enabling the QUDA smearing reduces the smearing time by a factor of 100–120, requiring only around 1–2% of the total measurement time. Thus, the total time for the job is reduced to about 30–40% of the original time.

7. Conclusion

We have implemented two-link staggered quark smearing in QUDA. When run on eight NVIDIA A100 GPUs, it performs up to six times faster than the MILC code run on total 256 cores of four AMD EPYC 7742 CPUs. Reusing the precomputed two-link product for another source on the same gauge field increases the difference up to 18 times.

The scaling behavior of this routine on both NVIDIA A100 and AMD MI250X is rather poor, especially for the two-link computation part. NVSHMEM can improve the performance of the

two-link computation by 30–50%. For a baryon correlator measurement, reusing the two-link product for all 72 sources/sinks reduces the total smearing time from 60–70% to 1–2% of the total measurement time. Thus, even without further optimization, this code can be useful for GPU jobs that require many smearings for sources or sinks.

Acknowledgments

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. We gratefully acknowledge support by the U.S. Department of Energy, Office of Science under award DE-SC0010120. This research was supported in part by Lilly Endowment, Inc., through its support for the Indiana University Pervasive Technology Institute. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. We thank the QUDA [8, 9] developers whose names can be found at their website[7] .

References

[1] S. Gusken, U. Low, K.H. Mutter, R. Sommer, A. Patel and K. Schilling, *Phys. Lett. B* **227** (1989) 266.

[2] S. Gusken, *Nucl. Phys. B Proc. Suppl.* **17** (1990) 361.

[3] C. Alexandrou, F. Jegerlehner, S. Gusken, K. Schilling and R. Sommer, *Phys. Lett. B* **256** (1991) 60.

[4] UKQCD collaboration, *Phys. Rev. D* **47** (1993) 5128 [hep-lat/9303009].

[5] G.M. von Hippel, B. Jäger, T.D. Rae and H. Wittig, *JHEP* **09** (2013) 014 [1306.1440].

[6] Hadron Spectrum collaboration, *Phys. Rev. D* **80** (2009) 054506 [0905.2160].

[7] https://github.com/lattice/quda.

[8] M.A. Clark, R. Babich, K. Barros, R.C. Brower and C. Rebbi, *Comput. Phys. Commun.* **181** (2010) 1517 [0911.3191].

[9] R. Babich, M.A. Clark, B. Joo, G. Shi, R.C. Brower and S. Gottlieb, 9, 2011, DOI [1109.2935].

[10] T.A. DeGrand and R.D. Loft, *Comput. Phys. Commun.* **65** (1991) 84.

[11] S.N. Syritsyn et al., *Phys. Rev. D* **81** (2010) 034507 [0907.4194].

[12] G.S. Bali, B. Lang, B.U. Musch and A. Schäfer, *Phys. Rev. D* **93** (2016) 094515 [1602.05525].

[13] https://github.com/milc-qcd/milc_qcd/tree/develop.

[14] https://developer.nvidia.com/nvshmem.

[15] https://github.com/lattice/quda/wiki/Multi-GPU-with-NVSHMEM.

[16] https://github.com/lattice/quda/wiki/Building-QUDA-With-HIP.

[17] https://kb.iu.edu/d/brcc.

[18] https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet-us-nvidia-1758950-r4-web.pdf.

[19] https://docs.olcf.ornl.gov/systems/crusher_quick_start_guide.html.

[20] https://www.amd.com/en/products/server-accelerators/instinct-mi250x.