High-Performance I/O: HDF5 for Lattice QCD

T. Kurth[1], A. Sarje[1], A. Pochinsky[2], S. Syritsyn[3], A. Walker-Loud[4]

[1]Lawrence Berkeley National Laboratory, [2]Massachusetts Institute of Technology,
[3]Brookhaven National Laboratory, [4]The College of William & Mary, Jefferson Lab

Introduction

- increasing demand on I/O performance in Lattice calculations
- fast parallel FS such as Lustre and GPFS available
- Lattice QCD practitioners should focus on physics, not I/O issues
- I/O demand lower than in some other fields, but it will become a major challenge on Exascale computing platforms
- we should gear up for these challenges and alleviate current and anticipated I/O bottlenecks.
- Motivated by this, we have incorporated a I/O software that:
  - is portable, i.e. the leading hpc architectures are supported
  - is standardized, i.e. third-party applications can read/write the files (Mathematica, Python, MATLAB, ...)
  - is stable and supported by IT experts
  - not proprietary
  - supports fast and reliable parallel I/O
  - supports flexible data types
  - simplifies data organization (single file can contain configs, propagators, etc ...)
- the **Hierarchical Data Format v5 (HDF5)** fulfills all these needs
- we added HDF5 support to QDP++ and QLUA

QDP++ and HDF5

- `HDF5Reader` and `HDF5Writer` classes drive HDF5 file access
- usage pattern resembles `XMLReader` and `XMLWriter`
- new commands resemble `cd`, `pwd` etc. known from UNIX/LINUX
- `attachAttribute` command for attaching metadata information to stored objects
- datatype automatically determined by read/write routines and committed into file if necessary
- hidden parallelism via MPIO
- dataset chunking allows to make use of Lustre striping
- data integrity: objects written to file are closed after every access

QLUA and HDF5

- follows narrow interface design approach of QLUA's QIO, i.e. `hf:write(path, object[, options])`
- HDF5 reader/writer objects, separately optimized
- allows user to specify global file options, i.e. file driver, chunk size, etc.
- automatic datatype creation and commitment
- all datasets are stored with checksums and timestamps
- convenience methods such as `ls`, `stat`, `chdir`, etc.

Lattice HDF5 I/O vs QIO

- QIO is the proprietary I/O driver in USQCD software stack
- offers possibility to manually regroup physical nodes into I/O groups using the `-iogeom` flag
- QIO benefits from suitable I/O geometries, but exhibits unstable performance in repeated runs

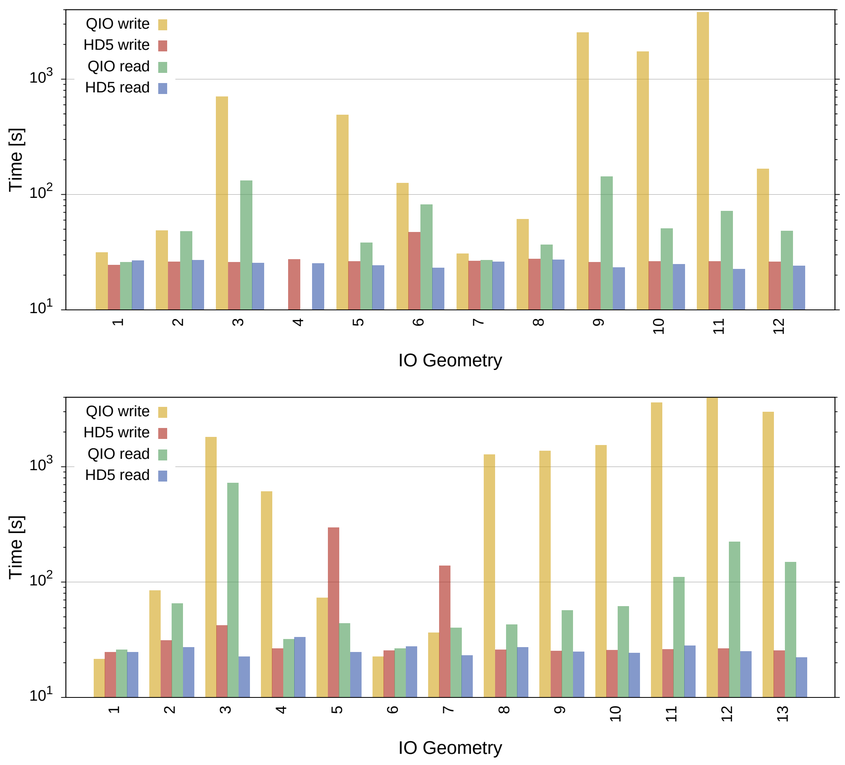

Figure: influence of `-iogeom` on QIO performance compared to HDF5 I/O on $64^3 \times 128$ lattice

Scaling

- Test scaling of HDF5 I/O on two architectures
  - Cray XC30 (Edison): 24 cores/node Intel Ivy Bridge, Aries dragonfly interconnect
  - Cray XK7 (Titan): 16 cores/node AMP Interlagos, Gemini interconnect

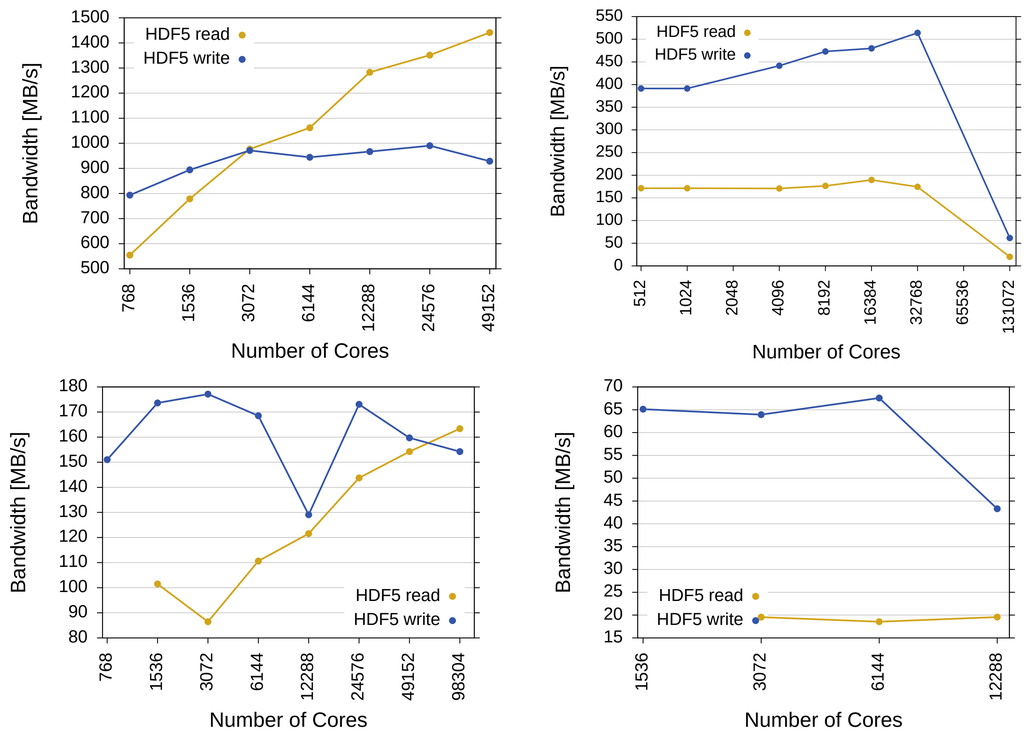

Figure: HDF5 I/O weak- (top) and strong-scaling (bottom) on edison (left) and titan (right) on a $128^3 \times 256$ lattice

Lustre Striping

- striping distributes chunks of large datasets over multiple OST's
- I/O performance depends on size and number of stripes
- I/O performance depends on the chunk size of datasets

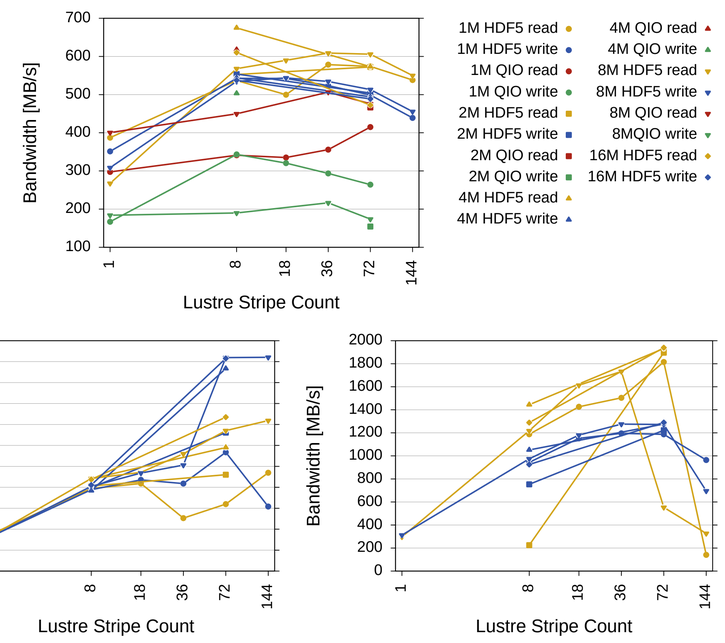

Figure: performance dependence on stripe count and size for 4 (top), 64(left) and 1024 I/O nodes (right) on Edison

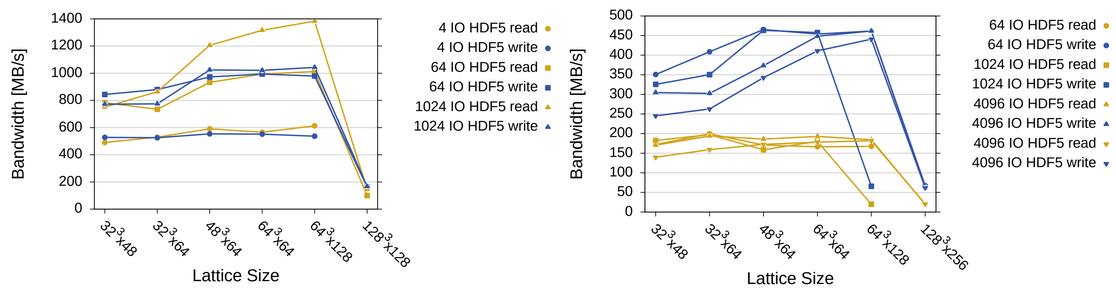

Figure: performance dependence on lattice size for 4 and 64 I/O nodes on Edison (left) and Titan (right)

Conclusion

- we implemented HDF5 into the USQCD software stack
- $\sim$10$-$20% improvement of I/O performance w/o tuning
- factor $\sim$5-8 improved performance w/ dataset chunking
- I/O performance more stable compared to QIO
- simplified data organization and convenience routines

Acknowledgements

- Balint Joo and Samuel Williams.
- DOE SciDAC-3 grants for the CalLat and USQCD Collaboration
- DOE Advanced Scientific Computing Research DE- AC02-05CH11231
- DOE Office for Nuclear Physics DE-FG02-94ER40818