HDF5 for USQCD

Thorsten Kurth (LBL)

Andrew Pochinsky (MIT)

Abhinav Sarje (LBL)

André Walker-Loud (WM)

Massachusetts
Institute of
Technology

Storage Needs

Storage Needs

- Large machines without full control of the environment

Storage Needs

- Large machines without full control of the environment
- Peta scale storage system with complex performance characteristics

Storage Needs

- Large machines without full control of the environment
- Peta scale storage system with complex performance characteristics
- Storing logically connected data on multiple lattices

Storage Needs

- Large machines without full control of the environment
- Peta scale storage system with complex performance characteristics
- Storing logically connected data on multiple lattices
- Random data elements addressing

Storage Needs

- Large machines without full control of the environment
- Peta scale storage system with complex performance characteristics
- Storing logically connected data on multiple lattices
- Random data elements addressing
- Large amount of data (10s TiB) need to be stored per job

QIO limitations

QIO LIMITATIONS

- Lack of scalability

QIO LIMITATIONS

- Lack of scalability
- Linear addressing only

QIO LIMITATIONS

- Lack of scalability
- Linear addressing only
- Single lattice per file

QIO LIMITATIONS

- Lack of scalability
- Linear addressing only
- Single lattice per file
- Maintenance costs

HDF5

HDF5

- Portable format

HDF5

- Portable format
- Hierarchical namespace

HDF5

- Portable format
- Hierarchical namespace
- Typed storage

HDF5

- Portable format
- Hierarchical namespace
- Typed storage
- Widely used in other fields

HDF5

- Portable format
- Hierarchical namespace
- Typed storage
- Widely used in other fields
- Professionally deployed at all computer centers

HDF5

- Portable format
- Hierarchical namespace
- Typed storage
- Widely used in other fields
- Professionally deployed at all computer centers
- Maintained by LBL at zero cost to USQCD

HDF5 for USQCD

Massachusetts
Institute of
Technology

HDF5 for USQCD

File drivers

- `posix` single-node write, serial data only
- `phdf5` multi-node write, serial and parallel data
- `mpiposix` multi-node write, serial and parallel data

High-level control of file organization is provided for optimizing storage throughput (chunking, alignment, GPFS hints, transfer modes)

HDF5 for USQCD

Object attributes

- `kind` standard string describing object's kind
- `time` 64 bit signed int time (μs since UNIX epoch)
- `sha256` SHA-256 checksum of the dataset
- other attributes ignored by readers

HDF5 for USQCD

Serial data types

Storage is compatible with SciPy conventions.

Serial data are written in HDF5 scalar dataspaces. Floating point data can be written in either single or double precision. The following types are currently provided:

```
String, Real, Complex, VectorInt(M), VectorReal(M),
VectorComplex(M), MatrixReal(N,M), MatrixComplex(N,M),
ColorVector(N), ColorMatrix(N), DiracFermion(N),
DiracPropagator(N)
```

HDF5 for USQCD

Lattice data types

Lattice data are written in HDF5 simple dataspaces. Each object has its own lattice geometry. Floating point data can be written in either single or double precision. The following types are currently provided:

```
LatticeInt, LatticeReal, LatticeComplex,
LatticeColorVector(N), LatticeColorMatrix(N),
LatticeDiracFermion(N), LatticeDiracPropagator(N)
```

HDF5 for USQCD

Example

```
hf = qcd.hdf5.Reader(“prop-sample.h5”);
p_forward = hf:read(“/u1750/forward/G24.2/x4y16z7t0/prop.61”)
p_backward = hf:read(“/u1750/backward-61/P/t19/px0py0pz-1/prop”)
hf:close()
```

Status

Status

- Qlua interface fully implemented in version 0.37.03

Status

- Qlua interface fully implemented in version 0.37.03
- Chroma interface in development

Pointers

- https://usqcd.lns.mit.edu/redmine/projects/qlua
- https://usqcd.lns.mit.edu/w/index.php/QLUA_Tutorial:HDF5
- http://www.hdfgroup.org/HDF5/