Parallel I/O

Philip Blakely

Laboratory for Scientific Computing, University of Cambridge

Input/Output

- All scientific codes need some form of output.
- In many cases these days, this is of the order of Gigabytes; we tend do post-processing rather than run-time reduction of data.
- If codes run in parallel, is the bottleneck for scalability the output? Amdahl's law comes into play once more.
- Most published demonstrations of parallel scaling are with output turned off.

Things you should already have considered

- How much data do you need to output? Is there any way of reducing this?
- Consider what plots/data you actually need for validation or comparison with experiment or for prediction.
- Consider what file-format(s) your visualisation/post-processing packages support. If necessary, consider writing your own post-processing tool or visualisation plugin to help.
- A complete dump of all the code's internal data is overkill in some cases, but that choice is often needed for debugging, full analysis and pretty pictures for publications.

Poor-man's I/O

The easiest method is to copy data to the root process and output there.
This may have the following issues:

- Memory usage on root process.
- Data communication time may adversely affect performance.

Alternatively, output one file per process
This may have the following issues:

- Visualisation/post-processing is harder (have to read/process many files)
- File-system performance is not necessarily better.
- If only one disk is involved, then extra seek-overhead as different processes write may be a problem.

MPI I/O

- The MPI standard has a well-defined set of function calls to handle parallel I/O.
- Processes can read/write from their own offsets within a file:
  - MPI_File_open
  - MPI_File_write
  - MPI_File_write_all
  - MPI_File_write_at
  - ...
- See `http://wgropp.cs.illinois.edu/courses/cs598-s16/lectures/lecture32.pdf` for an introduction.
- This is hard to get right, though, and (probably) even harder to get good performance.
- The easiest file-format to output this way would be a single block of binary data, where each process can trivially determine its offset within the file.

HDF5

- The HDF5 file-format and API is developed by the HDF-Group, which began in 1987.
- HDF5 is strictly speaking a file-format but one that allows a lot of freedom to the user.
- Essentially it's a file-system packaged into a single file.
- Allows for n-dimensional data arrays of general datatypes, groups (like directories), data-attributes (meta-data), and symlinks.
- It is up to you how to define your data structure (and hence your post-processing approach).
- Official APIs exist for C, Fortran, C++ and Java.
- A Python interface `h5py` exists, but not produced by HDFGroup.
- (Note: HDF is now moving to a paid-for Enterprise Support Edition plus a free Community Edition. The latter should be suitable for researchers, I hope...)

Compilation

- HDF5 needs to be compiled with the same compiler and MPI implementation as the rest of your software package.
- On CSD3, need to load appropriate module, use:

  ```
  spack find -dvl hdf5
  ```

  to find an appropriate one (lists against compilers used, MPI used, and configuration flags).

Data types

- Data-types include all C/Fortran types, plus any compound types you create yourself (much like MPI)
- Data is stored in a platform agnostic way, independent of the architecture, i.e. floating point numbers will be correct on porting between different endian-ness architectures.
- For example:
  - `H5T_NATIVE_CHAR`
  - `H5T_NATIVE_FLOAT`
  - `H5T_NATIVE_DOUBLE`
  - `H5T_NATIVE_ULONG`
  - `H5T_TIME`
  - `H5T_ARRAY` (Best for fixed-size vectors)

HDF5 API

To write a simple array of integer dimension `dimn` and length `size[]` in each dimension:

```
int dimn = 1;
hsize_t size[] = {3};
float myArray[] = {4,8,9};

// Open a file and get its handle
hid_t fileId = H5Fcreate("MyData.hdf", H5F_ACC_TRUNC,
  H5P_DEFAULT, H5P_DEFAULT);

// Define a group within the file
hid_t groupId = H5Gcreate(fileId, "MyGroup", H5P_DEFAULT,
  H5P_DEFAULT, H5P_DEFAULT);
```

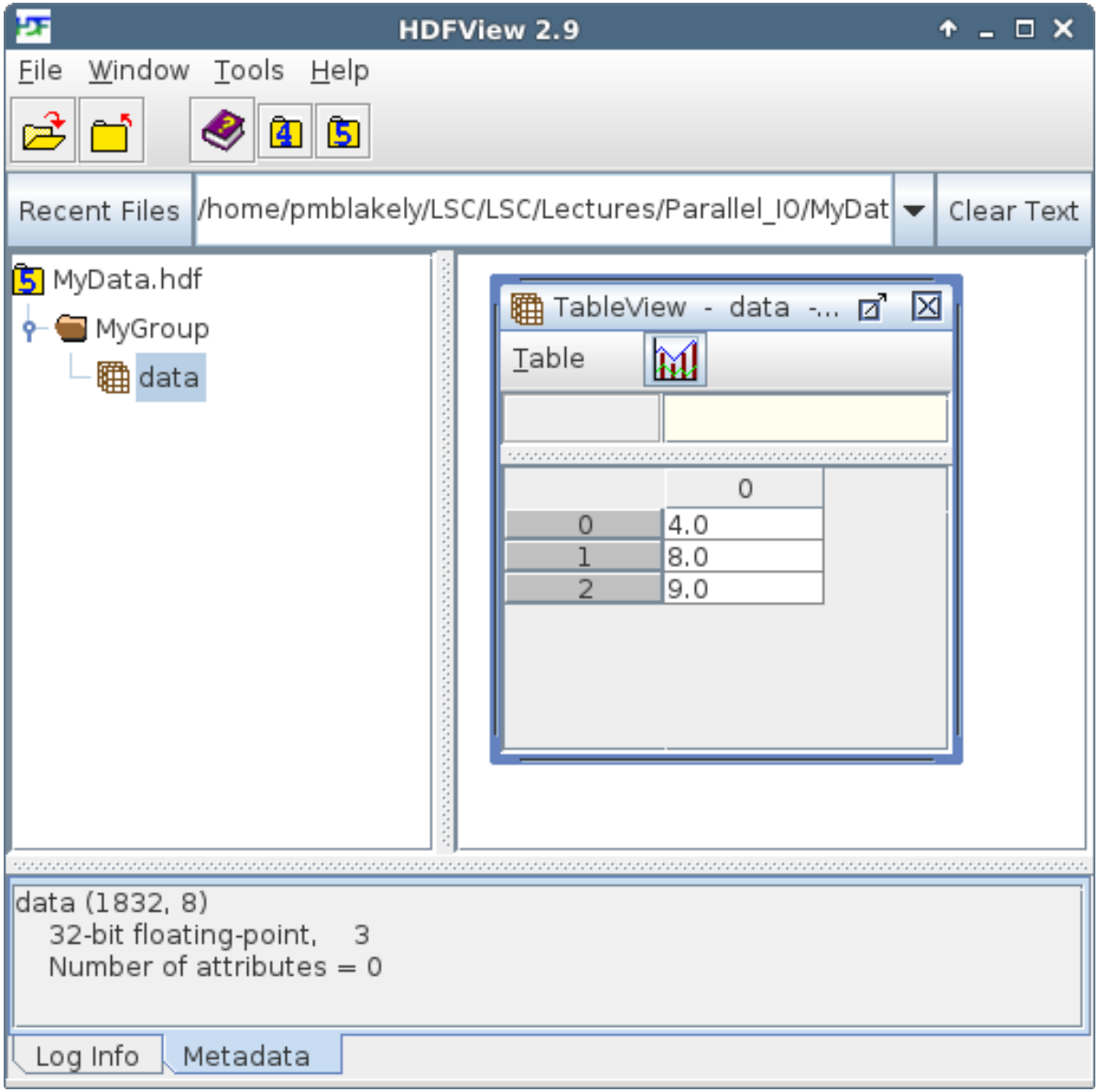
HDF5 API

```
// Create a dataspace which will be put into a data-set
hid_t dataSpace = H5Screate_simple(dimn, size, size);
// Create the data-set
hid_t dataSet = H5Dcreate(groupId, "data", cellDataType,
  dataSpace, H5P_DEFAULT, H5P_DEFAULT, H5P_DEFAULT);
hid_t fileSpace = H5Dget_space(dataSet);

// Write the data itself, assuming it's in single-precision.
H5Dwrite(dataSet, H5T_NATIVE_FLOAT, H5S_ALL, fileSpace,
  H5P_DEFAULT, (void*)myArray);

// Tidy up all data and file handles
H5Sclose(fileSpace);
H5Dclose(dataSet);
H5Sclose(dataSpace);
H5Gclose(groupId);
```

HDF5 API


HDFView 2.9
File Window Tools Help
Recent Files
/home/pmblakely/LSC/LSC/Lectures/Parallel_IO/MyDat
Clear Text
MyData.hdf
MyGroup
data
TableView - data -...
Table
0
0 4.0
1 8.0
2 9.0
data (1832, 8)
32-bit floating-point, 3
Number of attributes = 0
Log Info
Metadata

HDF in parallel

- HDF5 supports different processes outputing different data.
- Typically, a single large data-set is created in a file, and each process outputs a subset of this array:

```
hid_t subarray = H5Sselect_hyperslab(dataSpace,
    H5S_SELECT_SET, start, stride, count, block )
```

  where **stride**, **count**, **block** are arrays of size **dimn**.
- Then:

```
H5Dwrite(dataSet, cellDataType, subarray, fileSpace, plistId
    startAddress);
```

- HDF5 API and MPI implementation will work out how best to schedule outputing of data.
- This works best if your data is a distributed multi-dimensional array.
- Complexities occur if you're doing something complicated such as Adaptive Mesh Refinement...

HDF utilities

`hdfview` is a Java program that allows a simple GUI for viewing an HDF file.

Other command line utilities include:

- h5ls - list contents of an HDF5 file
- h5dump - dump contents of an HDF5 object to stdout/file
- h5diff - Compare two HDF5 files, or two groups within the same HDF5 file.

```
$ h5dump MyData.hdf
HDF5 "./MyData.hdf" {
GROUP "/" {
   GROUP "MyGroup" {
      DATASET "data" {
         DATATYPE  H5T_IEEE_F32LE
         DATASPACE  SIMPLE { ( 3 ) / ( 3 ) }
         DATA { (0): 4, 8, 9 }  } } } }
```

HDF5-based formats

The issue with HDF5 is that you still need to impose a layout on the file itself.
Some research communities have defined HDF5 layouts for their needs:

- CGNS (CFD General Notation System)
  `https://cgns.github.io/`
- NetCDF (Self-describibg array-oriented scientific data)
  `https://www.unidata.ucar.edu/software/netcdf/`
- HDF-EOS5 (Earth Observing System) from NASA
  `https://earthdata.nasa.gov/user-resources/standards-and-references/hdf-eos5`

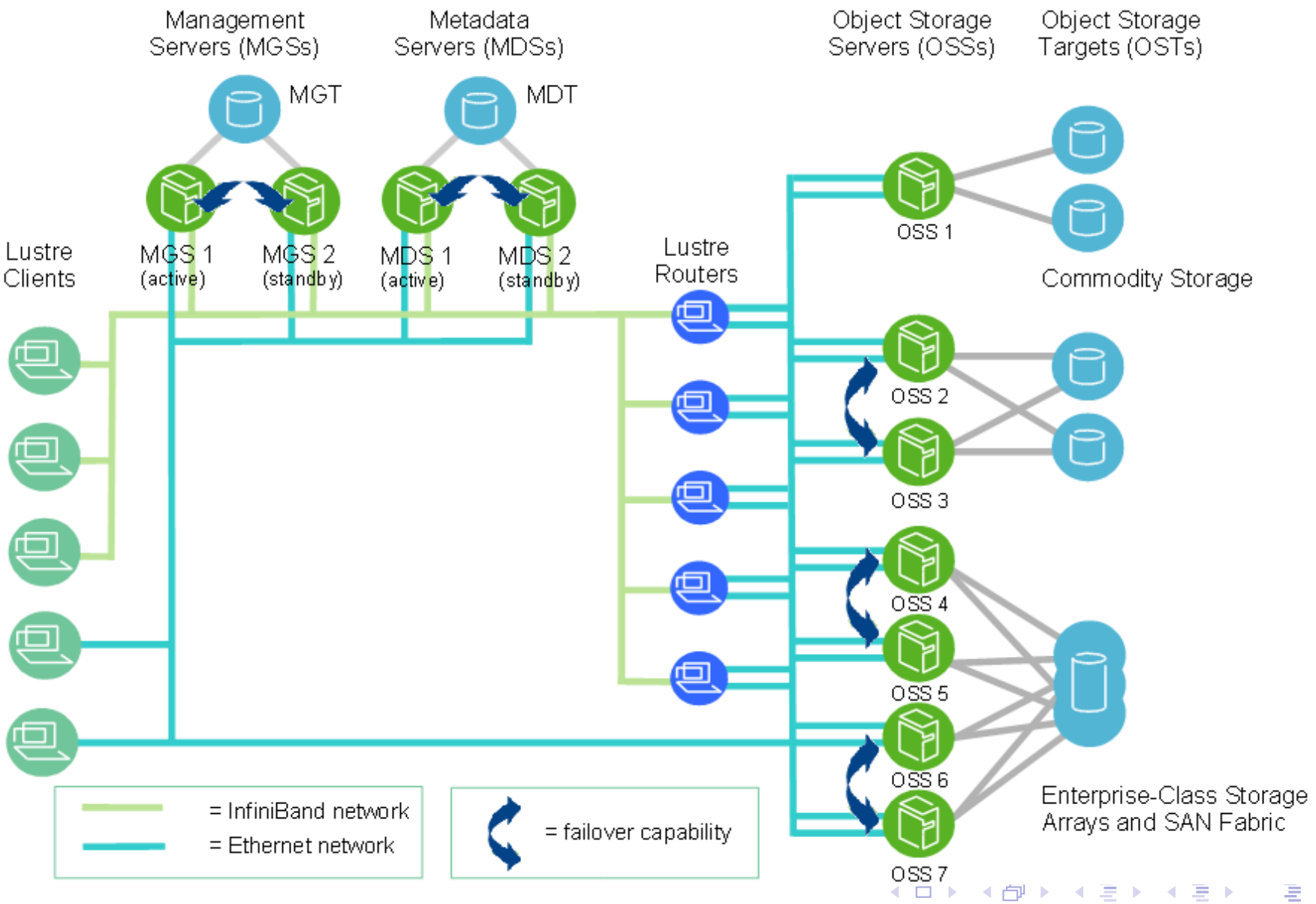
Parallel file-systems

- Ordinary disks can only maintain a serial input/output, so this can become the bottleneck for large numbers of processes.
- Parallel file-systems do exist: Lustre, Ceph, etc.
- These have their own problems (see numerous e-mails from HPCS about file-systems)
- They also require some tuning effort from the application writer to get the best performance out of them.

Striping

- Lustre works by having multiple filestores (or Object Storage Targets) which can be read/written separately.
- Files can be striped across multiple OSTs, so that the available bandwidth for reading/writing a file is increased proportionally (assuming no bottlenecks on network/individual servers).
- This does not happen automatically; you have to specify how a file is striped *ab initio*.
- See `http://wiki.lustre.org/Main_Page`

Lustre overview



Command line utilities

For Lustre (as on CSD3):

```
$ lfs getstripe ./test.out
lmm_stripe_count:  24
lmm_stripe_size:   1048576
lmm_pattern:       1
lmm_layout_gen:    0
lmm_stripe_offset: 2
        obdidx           objid           objid           group
             2         7398897       0x70e5f1               0
            15         7542547       0x731713               0
             4         7585327       0x73be2f               0
...
lfs setstripe -c 32 -S 1m ./newFile.hdf
```

For Ceph:

```
getfattr -n ceph.file.layout.pool file
setfattr -n ceph.file.layout.stripe_unit -v 1048576 file2
```

These only work for the creation of new files/directories, so require manual set-up for good performance.

MPI Implementation support

- Intel 2018 compiler is required for full Lustre I/O support (I think).
- Intel 2017 only supported efficient Lustre reading.
- Still need to set environment variables:

```
export I_MPI_EXTRA_FILESYSTEM=on
export I_MPI_EXTRA_FILESYSTEM_LIST=lustre
```

- OpenMPI and MPICH seem to have supported Lustre for much longer, but I haven't experimented with it.

HDF5 and parallelism

- The HDF5 API needs to ensure that a file-structure is consistently known across all processors.
- Hence various functions require implicit communication:
  - Dataset creation/deletion
  - Group creation/deletion
  - Full set: https://support.hdfgroup.org/HDF5/doc/RM/CollectiveCalls.html
- Therefore, it gives better performance to have a very simple group and dataset layout.
- Datasets can be written to from multiple processors simultaneously using 'hyperslabs' as previously demonstrated.
- May be appropriate to have one set of functions to create file-structure (quick: serialization matters less), and one set to write the actual data (slow).

HDF and striping

MPI (and hence HDF) needs to be explicitly told about Lustre (or similar) striping.

```
MPI_Info fileIOInfo;
MPI_Info_create(&fileIOInfo);
MPI_Info_set(fileIOInfo, "striping_unit", "1048576");
MPI_Info_set(fileIOInfo, "striping_factor", "32");
```

See `https://www.mpi-forum.org/docs//mpi-2.2/mpi22-report/node272.htm` for universal options.
Now, HDF5 can take the MPI_Info object:

```
H5Pset_fapl_mpio(pListId, MPI_COMM_WORLD, fileIOInfo);
hid_t fileId = H5Fcreate("MyData.hdf", H5F_ACC_RDWR,
    H5P_DEFAULT, pListId);
```

to improve its performance.

Testing performance

To test your machine's performance:

- IOR/mdtest: https://github.com/hpc/ior
- EPCC: https://github.com/EPCCed/benchio

For testing your application's performance:

- Darshan: http://www.mcs.anl.gov/research/projects/darshan/
- (Note: Must output Darshan logs to /home not /rds)
- Darshan example output (Darshan_output.pdf).