Flash-X and HDF5

Presenter: Rajeev Jain, Argonne National Lab

HDF5 Team Contact: Houjun Tang, Berkeley National Lab

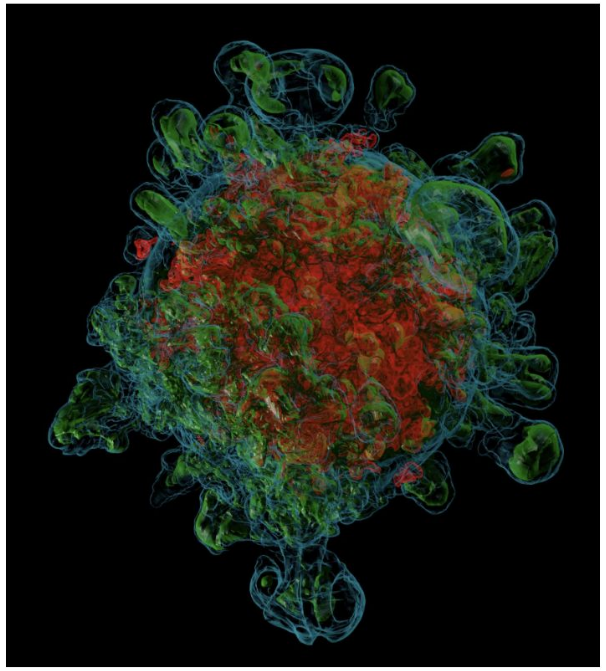
Flash-X

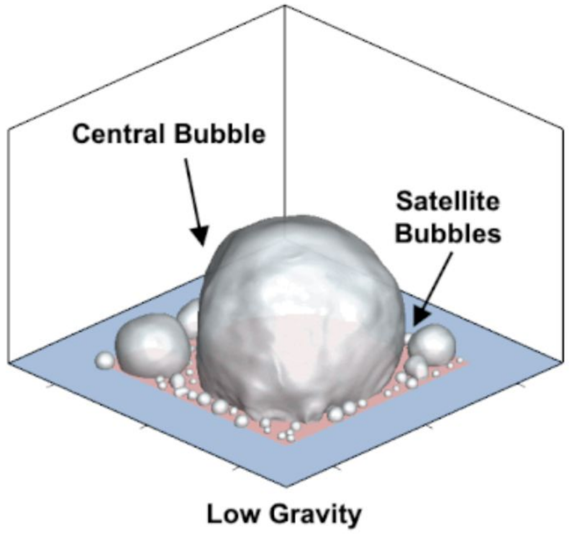
- Large global user base
- Composable multiphysics software
- Supports platform heterogeneity
- Derived from FLASH - 20+ years
- Apache-2.0 license
- https://flash-x.org/

Flash-X is an R&D 100 award winner for 2022

Simulations using Flash-X

Core-collapse Supernova and Gravity Effects on Pool Boiling.

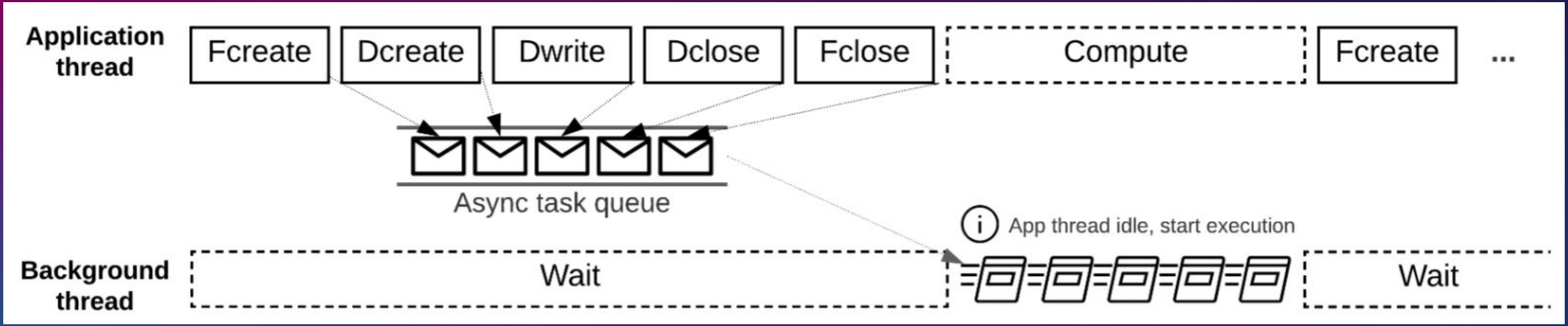




https://www.rdworldonline.com/rd-100-2022-winner/flash-x-a-multiphysics-simulation-software/

Flash-X and HDF5

- For over 20 years FLASH code has relied primarily on HDF5 for I/O
- New async API introduced by HDF5 1.13
- Asynchronous I/O VOL connector (github.com/hpc-io/vol-async) enables:
  - Transparent background thread execution of HDF5 I/O operations
  - Overlaps I/O with computation to reduce the total application execution time



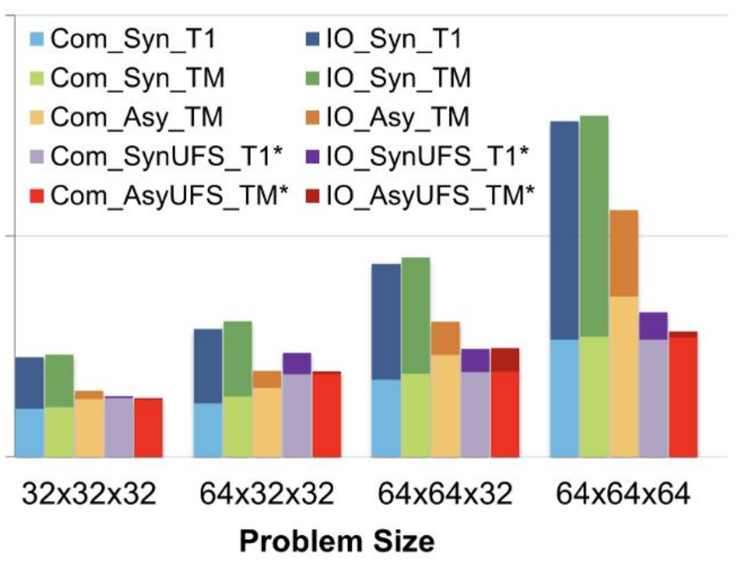
Async IO implementation in Flash-X

- Requires HDF5 1.13+, vol-async, and Argobots to be installed
- Flash-X async I/O can be turned on by add **`+hdf5async`** to the setup command

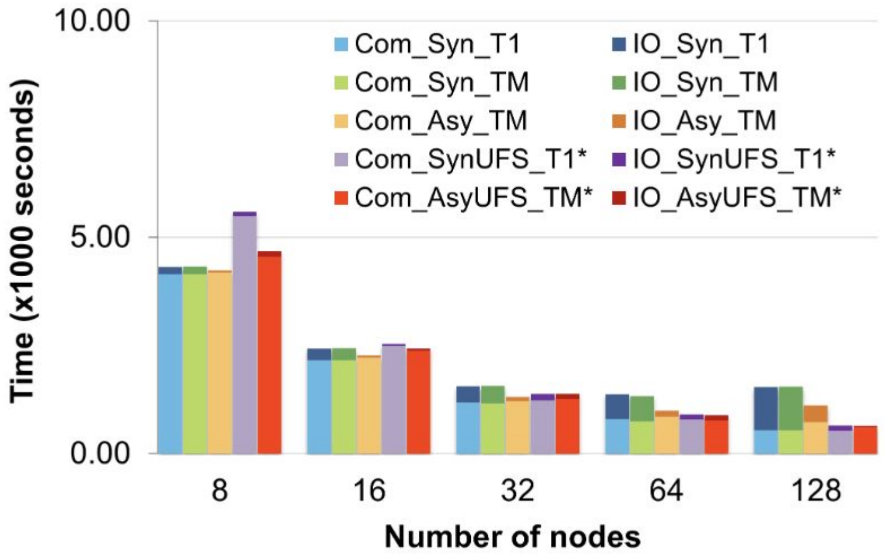
```
...
    /* create a parallel hdf5 dataset */
#ifdef FLASH_IO_ASYNC_HDF5
    dataset = H5Dcreate_async(*file_identifier, record_label_new,
                  H5T_NATIVE_DOUBLE, dataspace, H5P_DEFAULT, H5P_DEFAULT, H5P_DEFAULT, io_es_id);
#else
    dataset = H5Dcreate(*file_identifier, record_label_new,
                  H5T_NATIVE_DOUBLE, dataspace, H5P_DEFAULT,H5P_DEFAULT, H5P_DEFAULT);
#endif
...
```

Async IO Results with Sod

- Sod is a compressible flow explosion problem widely used for verification of shock-capturing simulation codes.
- We used a 3D Sod problem with tracer particles.
- Each runs for 109 steps, writes a checkpoint file every 33 steps, a plot file every 10 steps, and compared the total execution time with 5 different configurations that uses Synchronous and Asynchronous I/O, with and without MPI_THREAD_MULTIPLE, and using GPFS and UnifyFS.
- For cases with async, the majority of the write operations are overlapping with Flash-X's computation. Exceptions include the initial data writes and the last step as there is no computation to overlap with.



(a) Sod - weak scaling, 16 to 128 nodes



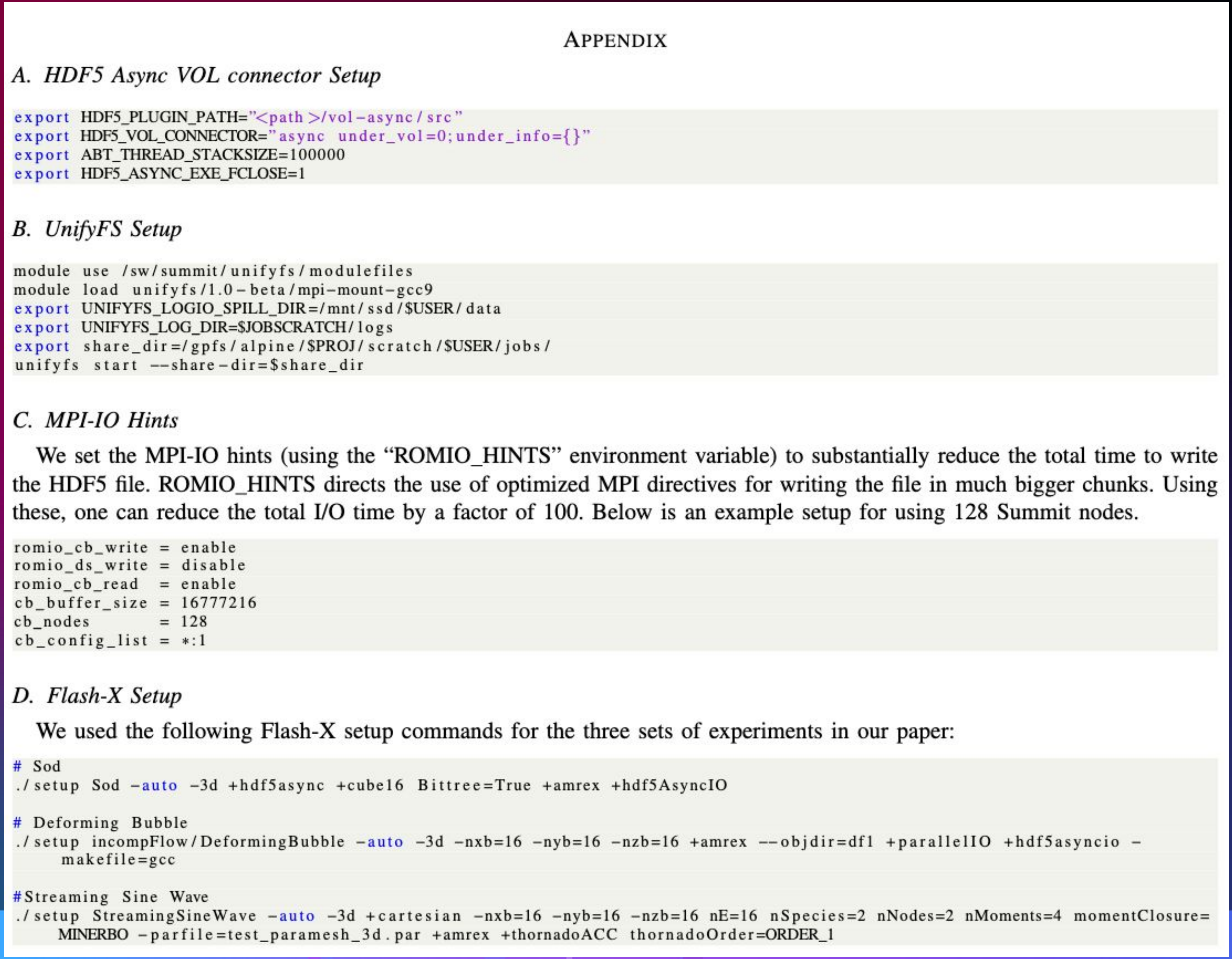
(b) Sod - strong scaling, problem size 64x64x64

Ongoing and Future work

- Enable lossy compression with:
    - SZ + HDF5 Filter
    - SZ3 + HDF5 Filter
    - ZFP +HDF5 Filter
    - Implemented via easy setup options and parameters
    - Initial results on summit show saving of 90% in IO space occupied with minimal loss of accuracy
    - Analyzing performance

- Apply asyncIO on more large production problems

- Simultaneous compression and asyncIO and understand limitations



Appendix

A. *HDF5 Async VOL connector Setup*

```
export HDF5_PLUGIN_PATH="<path>/vol-async/src"
export HDF5_VOL_CONNECTOR="async under_vol=0;under_info={}"
export ABT_THREAD_STACKSIZE=100000
export HDF5_ASYNC_EXE_FCLOSE=1
```

B. *UnifyFS Setup*

```
module use /sw/summit/unifyfs/modulefiles
module load unifyfs/1.0-beta/mpi-mount-gcc9
export UNIFYFS_LOGIO_SPILL_DIR=/mnt/ssd/$USER/data
export UNIFYFS_LOG_DIR=$JOBSCRATCH/logs
export share_dir=/gpfs/alpine/$PROJ/scratch/$USER/jobs/
unifyfs start --share-dir=$share_dir
```

C. *MPI-IO Hints*

We set the MPI-IO hints (using the "ROMIO_HINTS" environment variable) to substantially reduce the total time to write the HDF5 file. ROMIO_HINTS directs the use of optimized MPI directives for writing the file in much bigger chunks. Using these, one can reduce the total I/O time by a factor of 100. Below is an example setup for using 128 Summit nodes.

```
romio_cb_write  = enable
romio_ds_write  = disable
romio_cb_read   = enable
cb_buffer_size  = 16777216
cb_nodes        = 128
cb_config_list  = *:1
```

D. *Flash-X Setup*

We used the following Flash-X setup commands for the three sets of experiments in our paper:

```
# Sod
./setup Sod -auto -3d +hdf5async +cube16 Bittree=True +amrex +hdf5AsyncIO

# Deforming Bubble
./setup incompFlow/DeformingBubble -auto -3d -nxb=16 -nyb=16 -nzb=16 +amrex --objdir=df1 +parallelIO +hdf5asyncio -
    makefile=gcc

#Streaming Sine Wave
./setup StreamingSineWave -auto -3d +cartesian -nxb=16 -nyb=16 -nzb=16 nE=16 nSpecies=2 nNodes=2 nMoments=4 momentClosure=
    MINERBO -parfile=test_paramesh_3d.par +amrex +thornadoACC thornadoOrder=ORDER_1
```