# Accelerating Flash-X Simulations with Asynchronous I/O

Presenter: Rajeev Jain

Rajeev Jain*, Houjun Tang♦, Akash Dhruv*, Austin Harris[Φ], Suren Byna♦

* Argonne National Lab
♦ Lawrence Livermore National Lab
Φ Oak Ridge National Lab

# Contents

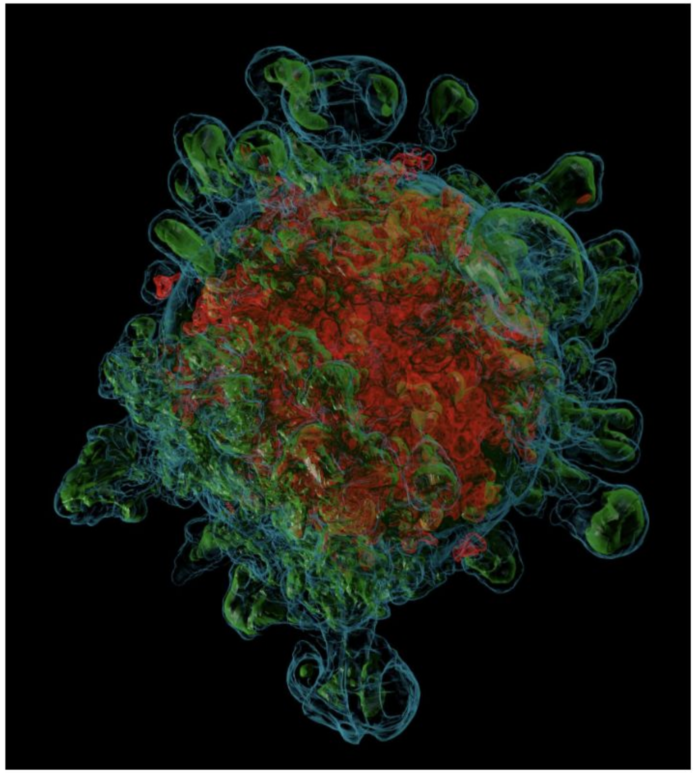
# Flash-X

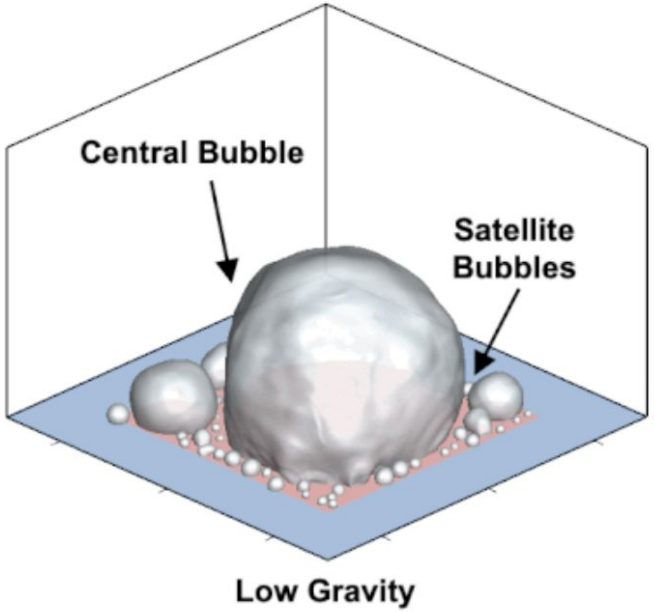
- Highly scalable multiphysics simulation code for heterogeneous compute architecture
- Supports “uniform” and “adaptive” mesh
- Primarily written in Fortran
- Component based code
- Eulerian base discretization
- AMR is used to:
  - Reduce memory footprint
  - Reduce computation
- Used for various simulations:
  - Galaxy clusters to
  - Turbulent Nuclear Burning

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

## Simulations using Flash-X

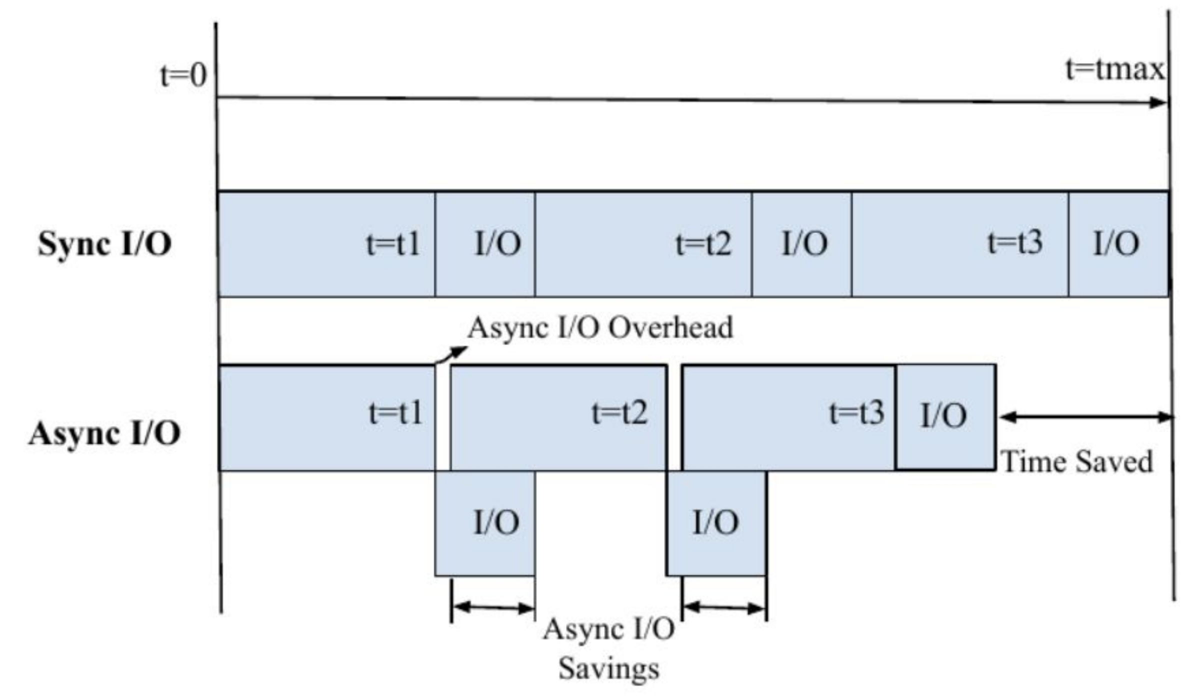
Core-collapse Supernova and Gravity Effects on Pool Boiling.





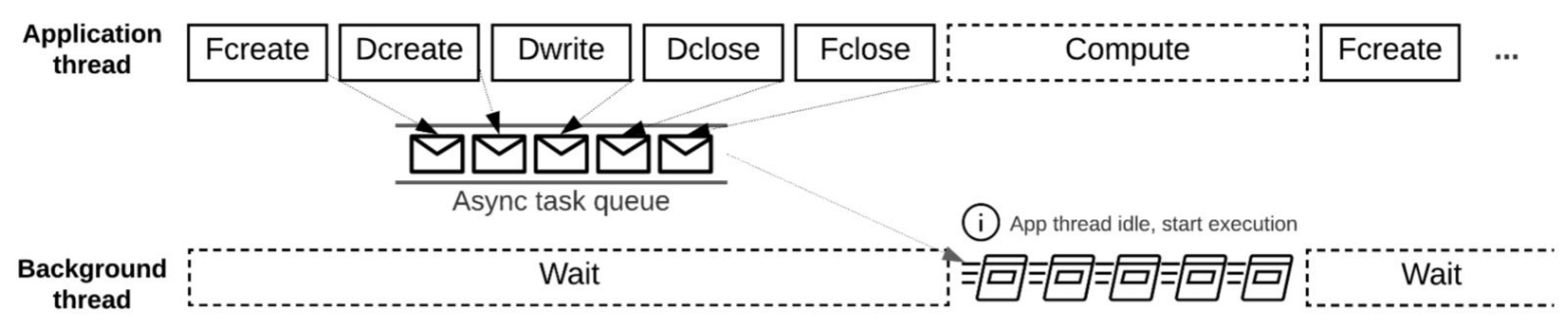
# Accelerating Flash-X with Asynchronous I/O

- Previous Flash-X versions only supports synchronous HDF5 I/O
  - I/O cost can be high with large scale simulation runs
  - Asynchronous I/O that can overlap computation with I/O can reduce the runtime



# HDF5 Asynchronous IO

- 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



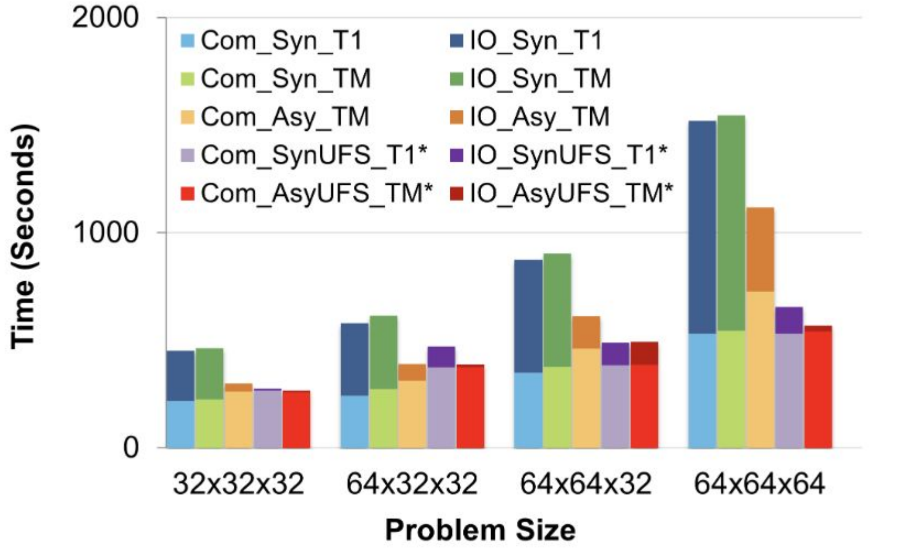
# Async IO implementation in Flash-X

- 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

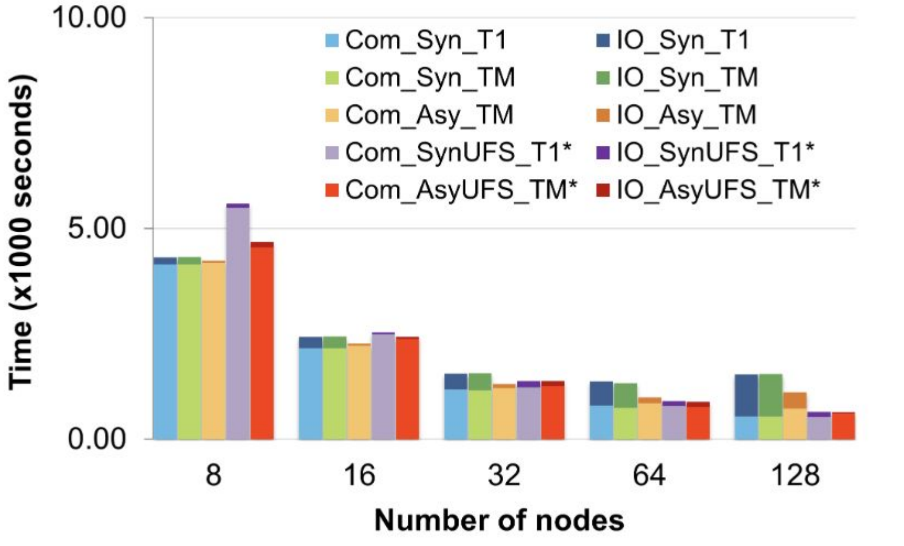
```
...
    /* create a parallel hdf5 dataset */
#ifdef FLASH_IO_ASYNC_HDF5
    dataset = H5Dcreate_async(*file_identifier, record_label_new,
                H5T_NATIVE_DOUBLE, dataspace, H5P_DEFAULT, H5P_DEFAULT, H5P_DEFAULT, io_es_id);
#else
    dataset = H5Dcreate(*file_identifier, record_label_new,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

# Results: 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

# Results: Streaming Sine Wave

- The streaming sine wave test problem is a test problem for verifying the correctness of the streaming advection operator in thornado as well as the Flash-X interface to thornado.
- This problem uses GPU  and CPU (threading).
- One GPU per MPI rank, and the data is copied from GPU to CPU memory automatically by FLASH-X before being written out
- At a higher number of nodes the interference between COM_ time and IO_ is higher as the I/O time as a whole increases: it is 27.1% for the 256-node synchronous case.

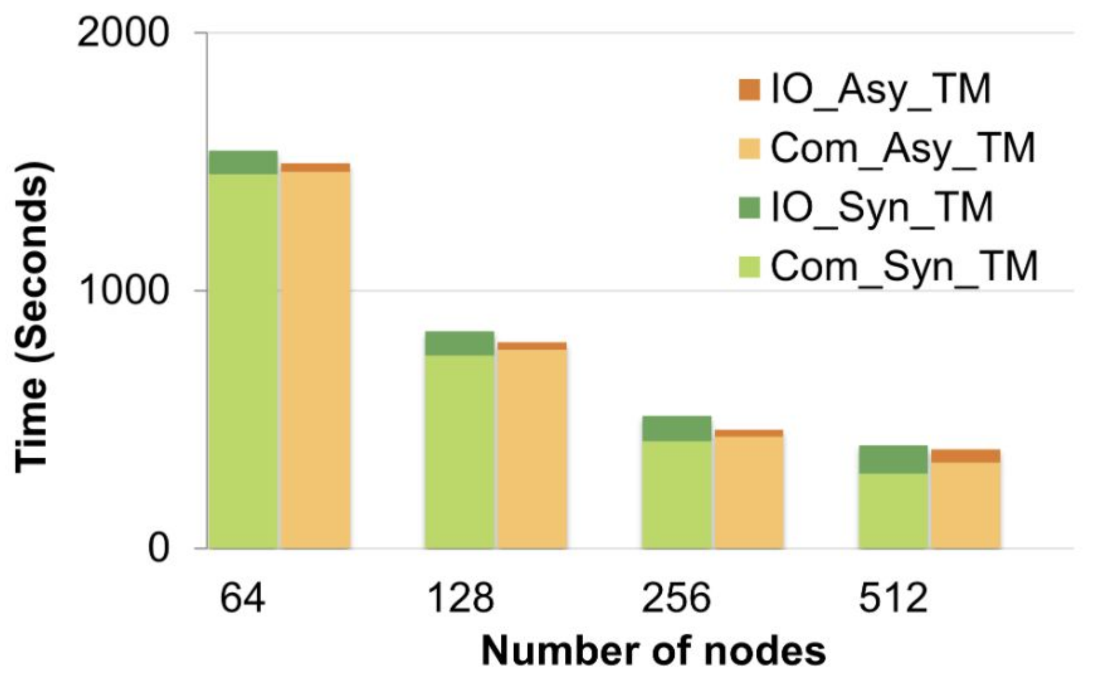



**Fig. 7:** Streaming sine wave - strong scaling

The total time required by synchronous I/O increases with increasing number of nodes. This is due to the fact that communication is time-consuming and the GPFS file-system write operation does not scale well.

# Results: Deforming Bubble Problem

- This is a benchmark problem for multiphase CFD applications in Flash-X. The deformation is computed by level-set advection and redistancing algorithm.
- For results shown in Fig. 6, the number of bubbles per MPI process is varied. Fig. 1 shows bubble undergo deformation under a velocity field.
- For the 64-node case the I/O time as a percentage of the total simulation time goes down from 22.3% to 4.7%.
- For the 256-node case, the I/O time is significantly higher for the synchronous case; this is due to the fact that a lot of communication is required to write the file to disk from 256 nodes (or 5,376 MPI ranks) and the GPFS file system on Summit does not scale well.
- The asynchronous I/O time for 256 nodes remains the same as for other cases, but the Com_ time has increased because a greater percentage of Com_ time overlaps with IO_ time.

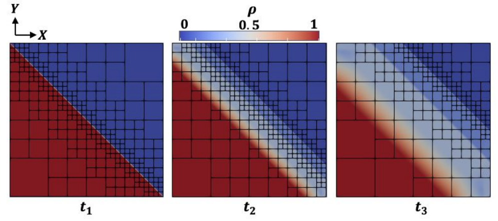



**Fig. 1:** Contours of energy (E) for time $t_3 > t_2 > t_1$, and an example of block structured AMR grids.

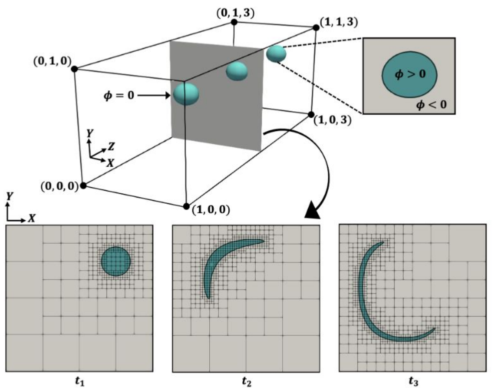



**Fig. 2:** Schematic of the deforming bubble problem: The bubbles are defined by using a signed distance function, $\phi$, that undergoes deformation under a prescribed velocity field.

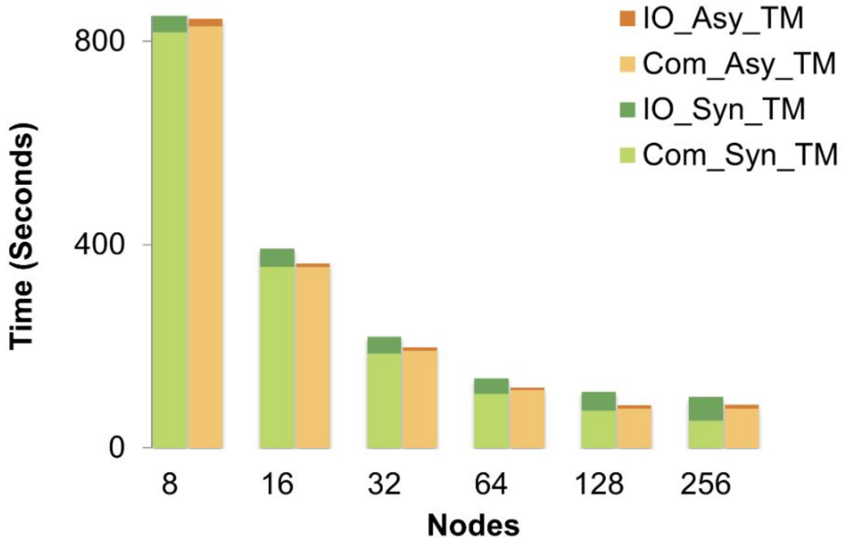



**Fig. 6:** Deforming bubble - strong scaling

# Conclusions and Future Work

- This work presents the performance evaluation of various problems from Flash-X that show significant performance gains by enabling asynchronous I/O.

- Heterogeneous applications utilizing MPI threads and GPUs are carefully chosen and set up to understand the limitations and advantages of the proposed method.

- The Flash-X code main branch already supports this feature, and it can be invoked by simply adding the *+hdf5AsyncIO* setup option in the setup command.

- We study three  problems: Sod uses AMReX for mesh refinement and communication, deforming bubble uses Paramesh and only MPI (no threads), and streaming sine wave uses also GPUs for computations.

- In the future, we want to add compression to the checkpoint files written asynchronously and study the performance.

## 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
```