UnifyFS: A File System for Burst Buffers

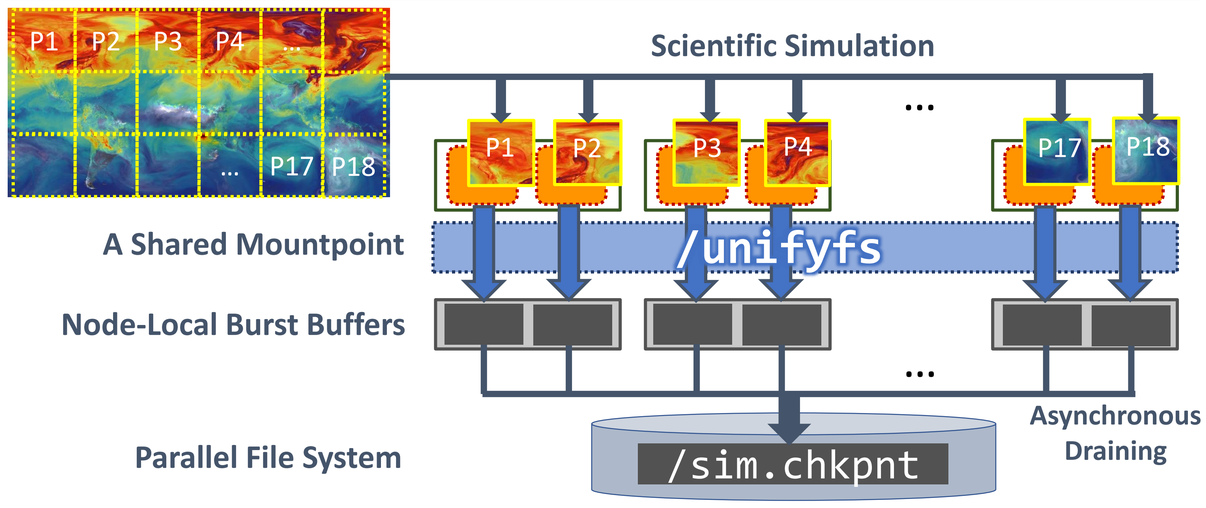
Kathryn Mohror (PI), Adam Moody, Cameron Stanavige, Tony Hutter (**Lawrence Livermore National Laboratory**)
Sarp Oral (Co-PI), Feiyi Wang, Hyogi Sim, Mike Brim, Swen Boehm (**Oak Ridge National Laboratory**)
Craig Steffen, Celso Mendes (**National Center for Supercomputing Applications**)
Contacts: kathryn@llnl.gov, oralhs@ornl.gov

UnifyFS: Project Description and Scope

UnifyFS Goals

- A user-level file system, highly-specialized for shared file access on HPC systems with distributed, node-local burst buffers
- Fast and scalable I/O by fully exploiting the node-local NVM devices
- Integration with resource managers to instantiate UnifyFS in user jobs
- Integration with I/O and checkpoint/restart libraries for transparent use by applications
- Usable directly by applications or through I/O or checkpoint/libraries
- LLNL, ORNL, and NCSA collaboration

UnifyFS Creates a Shared Namespace On Node-Local Storage



Using UnifyFS in a Job Is Easy

```
#!/bin/bash -l

#SBATCH -N 1024
#SBATCH -t 06:00:00
#SBATCH -J huge_job
#SBATCH -L SCRATCH

export UNIFYFS_SPILLOVER_DATA_DIR=/mnt/bb/$USER/data
export UNIFYFS_SPILLOVER_META_DIR=/mnt/bb/$USER/meta

unifyfs start --mount=/unifyfs --stage-in=/pfs/my/data/in &

srun -n 4096 ./simulation

unifyfs terminate --cleanup --stage-out=/pfs/my/data/out
```

Launch UnifyFS daemon and mount at `/unifyfs`, stage-in the input data from `/pfs/my/data/in`

Terminate the UnifyFS daemon and store the data to `/pfs/my/data/out`

`unifyfs` command line tool

- start-up and termination
- Independent from resource managers
- Provides advanced configuration options
  - *`mount`*: mount point name to be used
  - *`stage_in`*: path to data to be staged in at the beginning of the job
  - *`stage_out`*: path to data to be staged out at the end of the job
  - *`cleanup`*: specifies if UnifyFS storage should be cleaned up at the end of the job
  - *`consistency_model`*: desired consistency model
  - *`script`*: custom launch/termination script

Will Support Common HPC I/O Use Cases

- Checkpoint/Restart
- Scientific applications generating periodic output data
- Ensemble applications sharing data through files
- I/O with HPC I/O libraries including HDF5, ADIOS, MPI-I/O, PnetCDF

Simply Change the File Path to Use UnifyFS

```
int main(int argc, char **argv) {
 MPI_Init(argc, argv);

 for (t = 0; t < TIMESTEPS; t++) {

  /* do work ... */

  checkpoint();
 }

 MPI_Finalize();

 return 0;
}
```

```
void checkpoint(void) {
 int rank;

 MPI_Comm_rank(MPI_COMM_WORLD, &rank);

 // file = "/pfs/shared.chpt";
 file = "/unifyfs/shared.ckpt";

 File *fs = fopen(file, "w");

 if (rank == 0)
  fwrite(header, ..., fs);

 long offset = header_size +
               rank*state_size;
 fseek(fs, offset, SEEK_SET);
 fwrite(state, ..., fs);
 fclose(fs);
}
```

The only required change is to use `/unifyfs` instead of `/pfs`

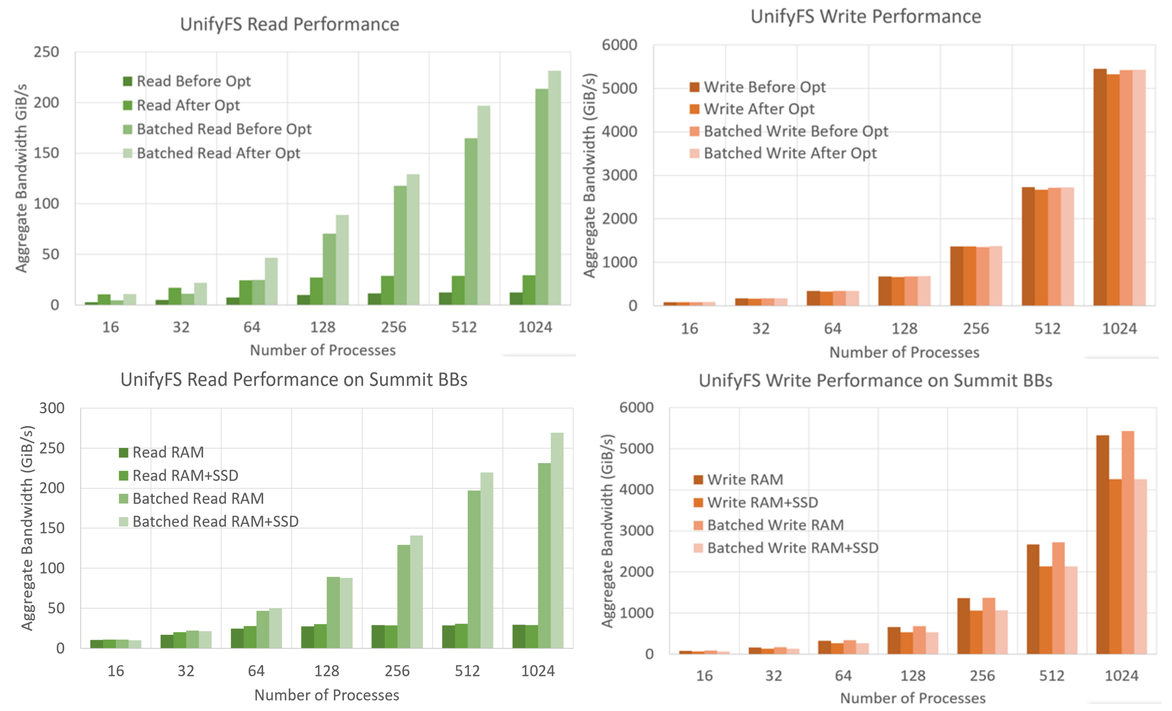
Transparent support of a shared checkpoint file via POSIX I/O

- UnifyFS transparently intercepts application I/O calls via the mountpoint
- `/unifyfs/shared.chpt` becomes visible to all processes in a job

Laminated consistency model

- Disjointed write and read phases, ideal for most checkpoint workload

UnifyFS Performance on Summit



- On Summit@ORNL
  - Conventional `write(2)` and `read(2)`
  - Batched write/read using `lio_listio(3)`
  - 8 processes per node
- Improved read performance 2x within the last year
- UnifyFS can exploit both system RAM and SSD

Work in Progress to Improve UnifyFS

- Improving metadata handling performance
- `libunifyfs`, UnifyFS programming interface
- Supporting IO libraries including HDF5 and VeloC

Try UnifyFS!

UnifyFS is currently available *(MIT License)*

- Officially in spack: `spack install unifyfs`
- Github project repository at https://github.com/LLNL/UnifyFS
- User support via a mailing list, ecp-unifycr@exascaleproject.org

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. LLNL-POST-801859.
This research was supported by the Exascale Computing Project (ECP), Project Number: 17-SC-20-SC, a collaborative effort of two DOE organizations - the Office of Science and the National Nuclear Security Administration, responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering and early testbed platforms, to support the nation's exascale computing imperative.