The Center for Astrophysical Thermonuclear Flashes

Infrastructure III: I/O

Paul Rich

An Advanced Simulation & Computing (ASC)
Academic Strategic Alliances Program (ASAP) Center
at The University of Chicago

Why worry about I/O?

- Good place to pick up performance
- Easy place to lose performance
- Faster I/O means better sampling of data
- Lots of options on systems for making I/O faster
- Can make subsequent steps easier

Key Flash I/O Feature Overview

- Multiple I/O Modes
  - Serial, Parallel, Hybrid
- Multiple I/O Libraries supported
  - HDF5
  - PnetCDF
  - Direct
  - More can be brought in under FLASH's architecture
- Transparent Restarting
- Arbitrary I/O File Splitting
- Multiple File Types
- Integral Quantities

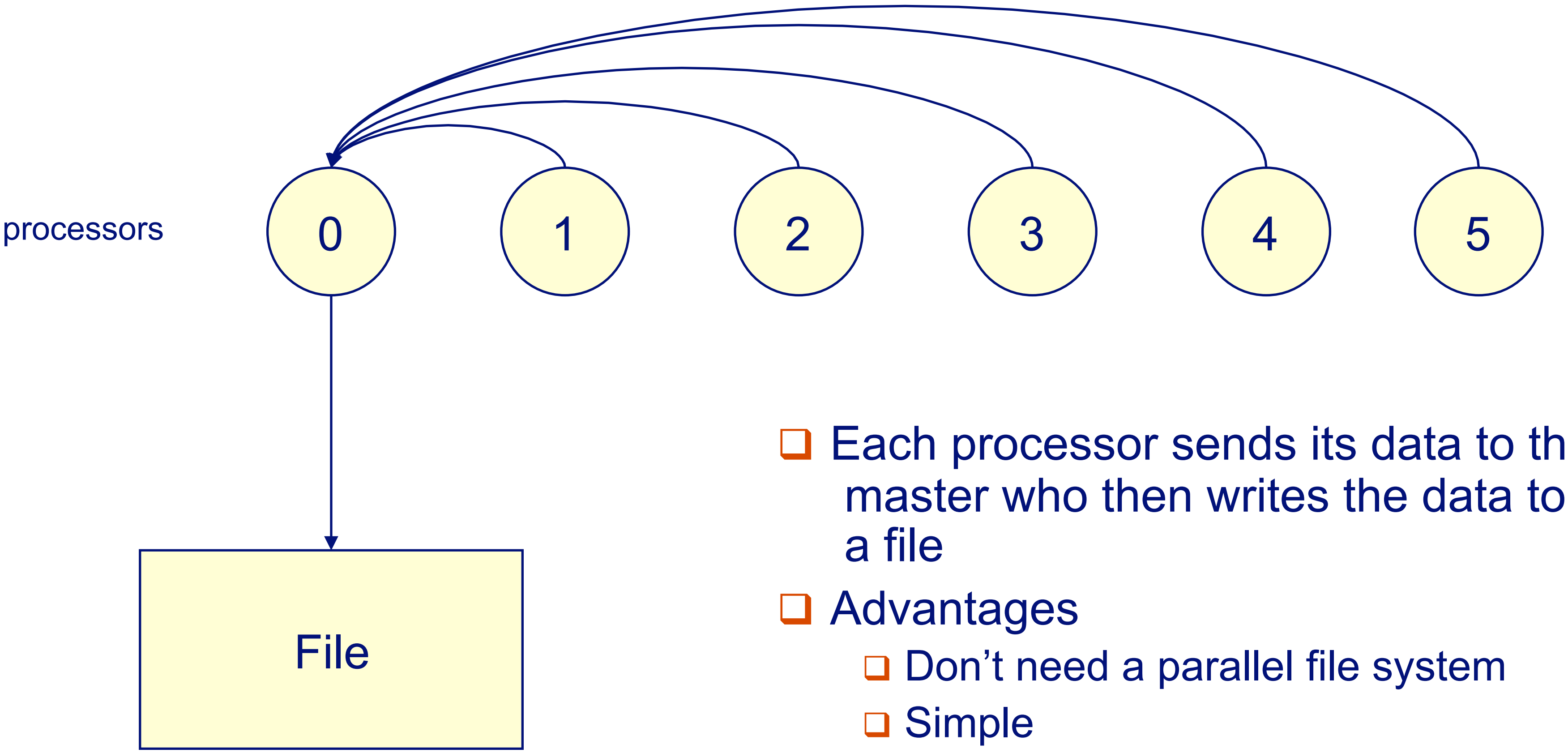
File Types - Diagnostic Files

- Log File: *flash*.log
  - Generated by the Logfile module
  - Collects events during a run, and often provides more data than stdout/stderr
  - Can also put out individual process logfiles -- good for debugging
- Dat File: *flash*.dat
  - Collection of quantities generated per time step
  - Usually integrated over the physical domain
- amr.log -- Paramesh only!
  - Generated by Paramesh in the event of an error
- Timer summaries: *timer_summary_xxxxx*
  - Allows for the collection of individual processor timing data from FLASH’s timers, each processor writes out a file
  - Can be turned off by setting *eachProcWritesSummary* to false

File Types -- Large Files

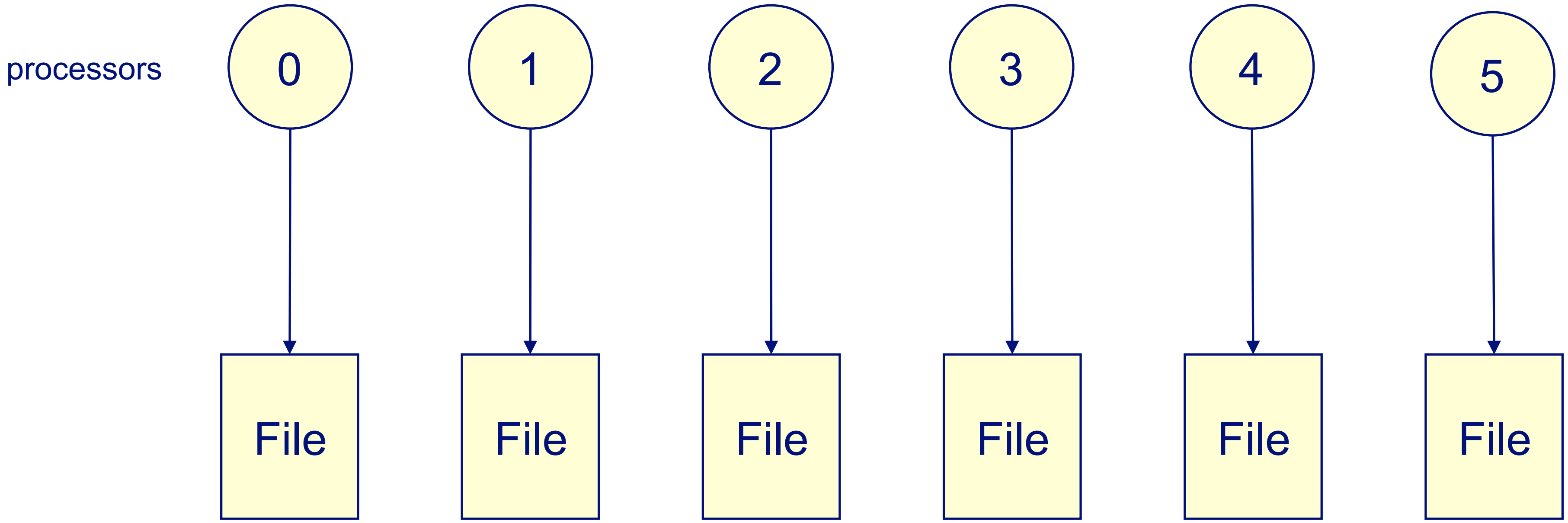
- Checkpoint files: *basename_filetype*_chk_*xxxx*
  - Contain everything you need to restart outside of a parfile
  - Large, but can save a lot of time and CPU hours
  - Can be set to “roll” via the rollingCheckpoint parameter
- Plot Files: *basename_filetype*_plt_cnt_*xxxx*
  - Contains specific Eulerian quantities specified in your parfile
  - Much smaller and faster to output than a checkpoint
  - By default double-sized floating point data is output in single precision
- Particle files: *basename_filetype*_part_*xxxx*
  - Contains header information, particle metadata and particle data
  - Typically very small and fast to output

Serial I/O



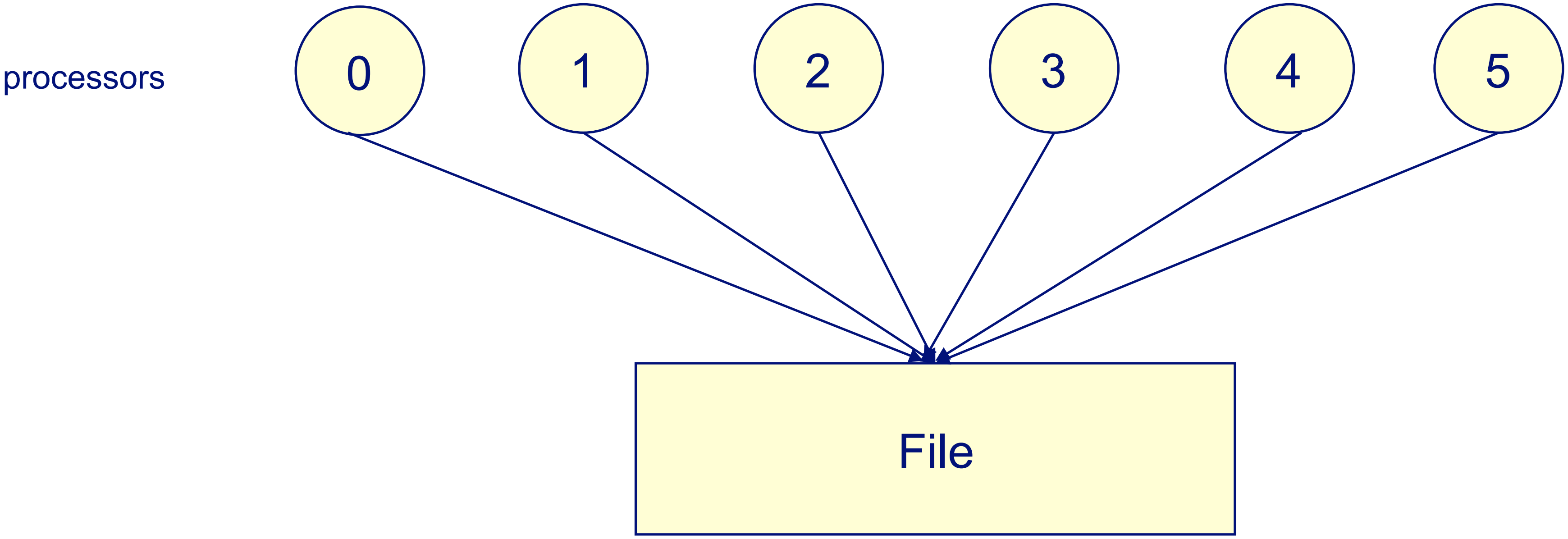
- ❑ Each processor sends its data to the master who then writes the data to a file
- ❑ Advantages
  - ❑ Don’t need a parallel file system
  - ❑ Simple
- ❑ Disadvantages
  - ❑ Not scalable
  - ❑ Not Efficient

Parallel I/O: Separate Files



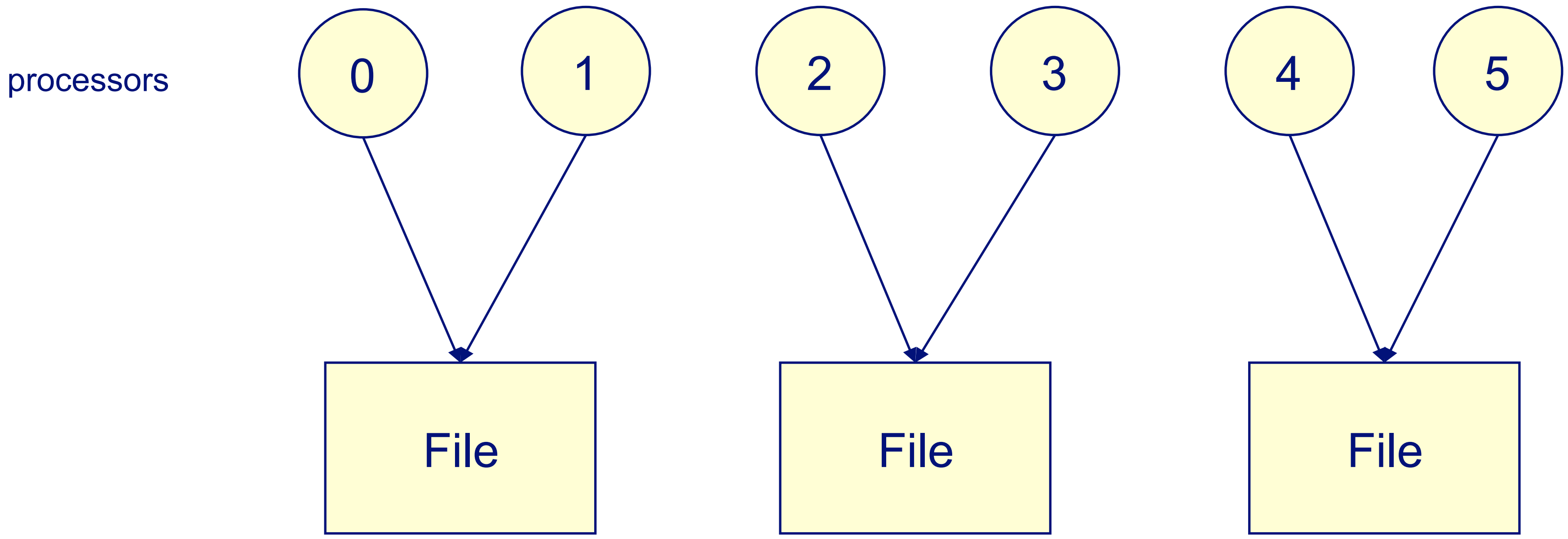
- Each processor writes its own data to a separate file
- Advantages
  - Fast!
- Disadvantages
  - can quickly accumulate many files
  - hard to manage
  - requires post processing

Parallel I/O: Single-file



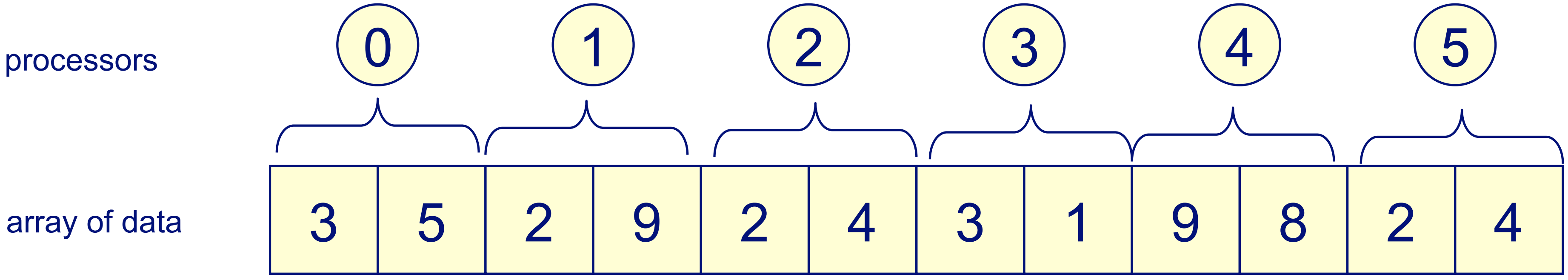
- Each processor writes its own data to the same file using MPI-IO mapping
- Advantages
  - single file
  - scalable
- Disadvantages
  - requires MPI-IO mapping or other higher level libraries

Parallel I/O Split File



- Hybridized model: parallel output to multiple files
- Advantages
  - Potentially more scalable than single file
  - Can take advantage of architecture
- Disadvantages
  - Requires MPI-IO mapping or other higher level libraries
  - Still have multiple files to deal with

Parallel IO single file



processors

0 1 2 3 4 5

array of data

3	5	2	9	2	4	3	1	9	8	2	4

Each processor writes to a section of a data array. Each must know its offset from the beginning of the array and the number of elements to write

HDF5

- Library maintained by the HDF group
- Allows for serial and parallel operations
- Primary IO format for FLASH
- Pros:
  - Data is stored with metadata that increases portability
  - Very flexible data format
  - Handles large volumes of data well
  - Most tools for working with FLASH files are written for this format
- Cons:
  - Can be slower than other IO libraries
  - Lots of settings, can be confusing

HDF5: Notes on Parallel Mode

- ❑ Parallel HDF5 can be run using an independent access pattern or a collective access pattern

- ❑ Collective operations can aggregate reads and writes from multiple processes so that the data can be written in one disk operation

- ❑ This can lead to dramatic increases in speed.

- ❑ Collective mode may not play nice with other HDF5 features

PnetCDF

- Library maintained by Argonne National Laboratory
- Allows for parallel operations, a CDF library can be used for serial tools.
- Every operation is run in collective mode
- Pros:
  - Very fast if collective operations are enabled, can be faster than HDF5
  - Newest version can handle large datasets
  - Interface to files is simpler than HDF5
- Cons:
  - Not as flexible
  - Support for large datasets still experimental
  - Some tools for FLASH do not support PnetCDF files

Direct IO

- Each processor performs a binary write to disk.
- Data split up into *n* files where *n* is the number of processors.
- Pros:
  - Always available.
  - One of the fastest methods available.
- Cons:
  - No automated reader
  - Files will be non-portable
  - Can generate too many files
- Warning:
  - Method of Last Resort!
  - Implementation within FLASH3 is only an example should this mode be necessary.

Flash Center IO Nightmare…

- Large 32,000 processor run on LLNL BG/L
- Parallel IO libraries not yet available
- Intensive I/O application
  - checkpoint files .7 TB, dumped every 4 hours, 200 dumps
    - used for restarting the run
    - full resolution snapshots of entire grid
  - plotfiles - 20GB each, 700 dumps
    - coarsened by a factor of two averaging
    - single precision
    - subset of grid variables
  - particle files 1400 particle files 470MB each
- 154 TB of disk capacity
- 74 million files!
- Unix tool problems
- 2 Years Later still trying to sift though data, sew files together

Integral Quantities

- ❑ Individual file output by the master PE
- ❑ Collects quantities integrated by volume over the grid
  - ❑ Cartesian geometries are supportes along with 2D cylindrical
- ❑ Frequently overrode in individual simulations for additional functionality
- ❑ If modified, the user is responsible for all MPI needed to marshal data
  - ❑ Recommended that you use Flash_mpi.h and FLASH_REAL for MPI calls.
- ❑ Also a good place for step-by-step statistics for debugging

Tips and Tricks for I/O

- Examine the system documentation!
  - Often there are individual file system flags to improve performance
- Experiment with different settings
  - Every system can be a bit different.
  - Data is data to I/O
- When building your own setup, make sure right units get included
  - Particle I/O is a separate subunit
- Restarts do interact with the environment
  - Parameter file changes for this run are used

Questions?

Questions?