

# I/O

Chris Daley

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

# 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

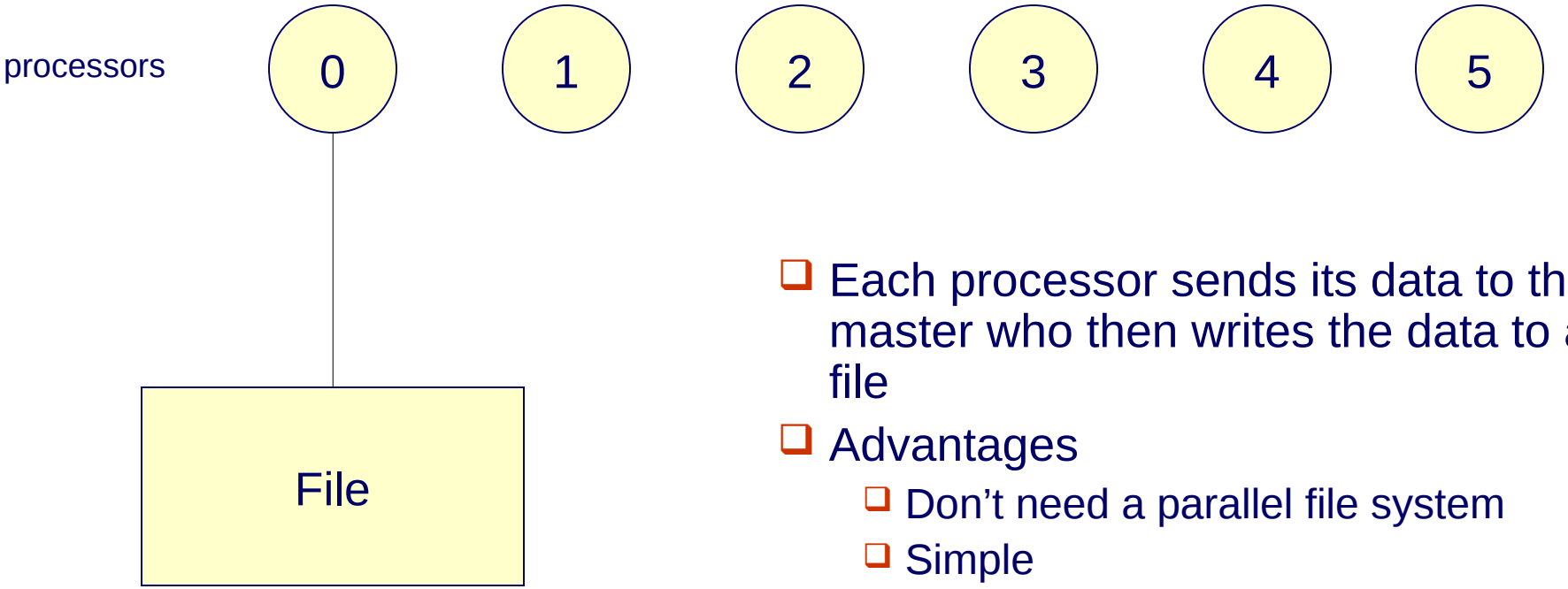
# File Types - Diagnostic Files

- 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

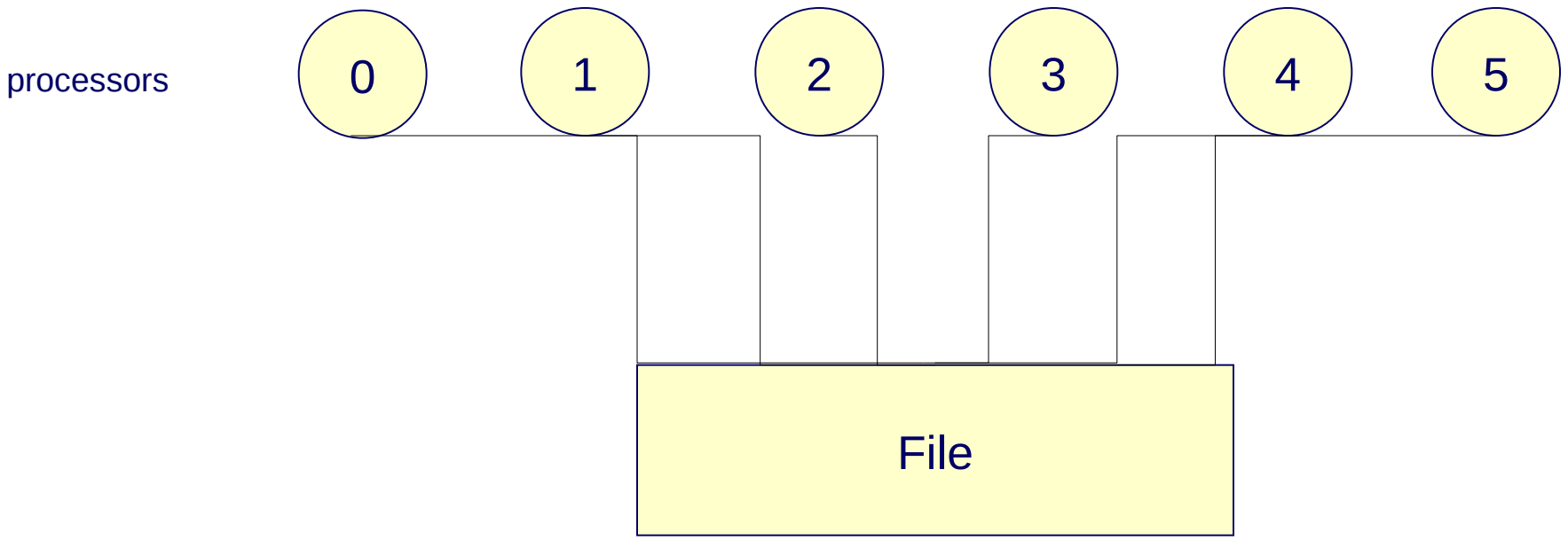



- Each processor sends its data to the master who then writes the data to a file
- Advantages
  - Don’t need a parallel file system
  - Simple
- Disadvantages
  - Not scalable
  - Not Efficient

# Selecting serial I/O

- Simulation/SimulationMain/Config contains “REQUIRES IO”
  - This means all applications will include the default I/O implementation (unless explicitly removed using +noio)
  - The default I/O implementation will be compatible with the chosen mesh package, but may not be what you want
    - HDF5 vs pnetcdf (HDF5 selected)
    - Serial vs parallel (Serial selected)
- No extra options required for serial HDF5 I/O implementation
  - ./setup Sedov -auto +pm4dev
  - ./setup Sedov -auto +ug
- There is no serial I/O implementation for NOFBS uniform grid or pnetCDF library.

# Parallel I/O




- 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

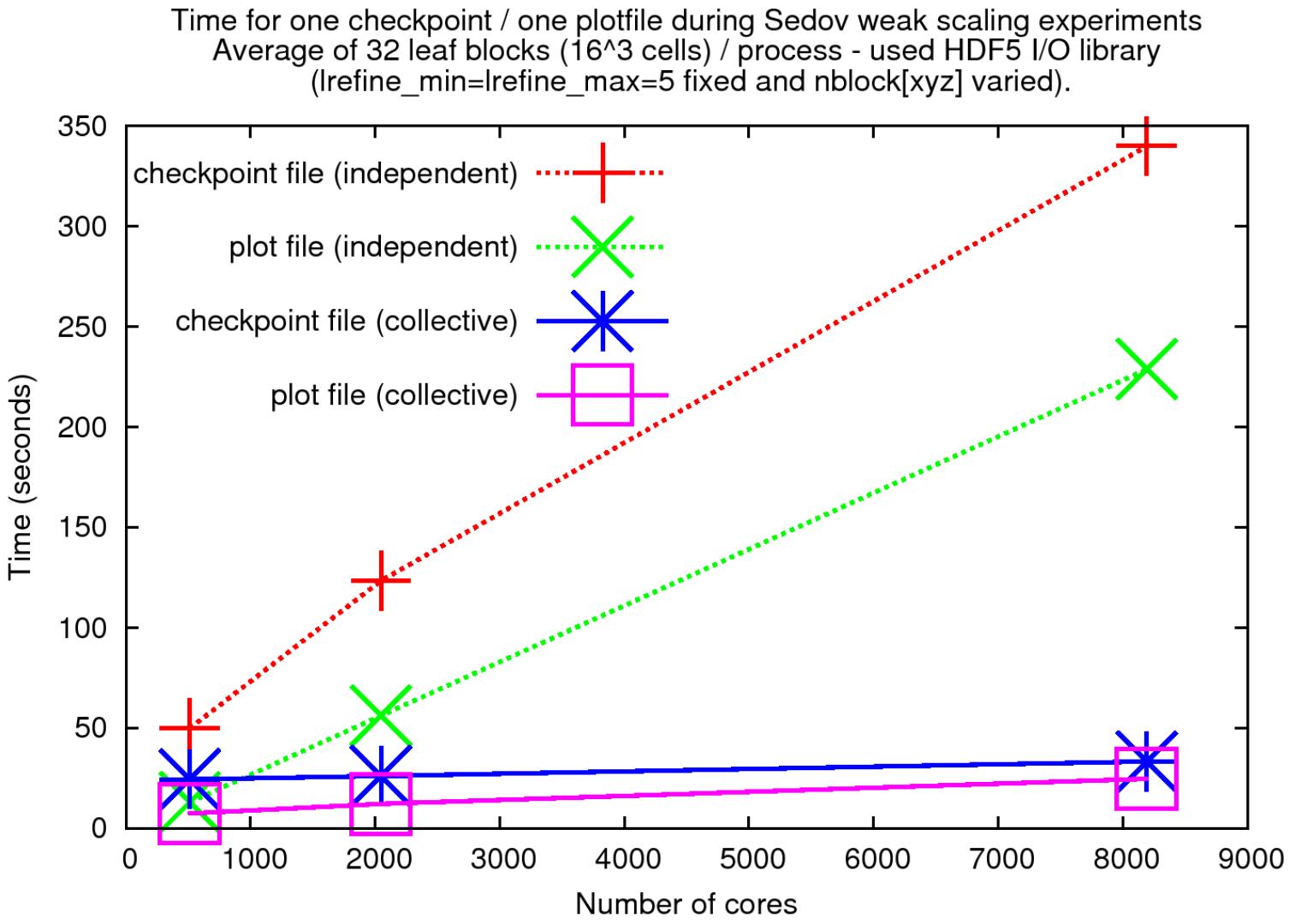
# Selecting parallel I/O

- The explicit setup option for parallel I/O is +parallelIO
  - Only needed when there is an alternative serial I/O implementation (e.g. in PARAMESH HDF5 case)
- PARAMESH
  - ./setup Sedov -auto +pm4dev +parallelIO
  - ./setup Sedov -auto +pm4dev +pnetcdf
- NOFBS
  - ./setup Sedov -auto +nofbs (HDF5 only)
- UG
  - ./setup Sedov -auto +ug +parallelIO
  - ./setup Sedov -auto +ug +pnetcdf

# 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 transferred in fewer disk operations
- Switch on/off with *useCollectiveHDF5* parameter in flash.par
- This can lead to dramatic increases in speed....

# ...e.g., weak scaling test on Intrepid



# Common issues with FLASH I/O

io_h5file_interface.c:(.text+0xc6): undefined reference to `H5Pset_fapl_mpio'

- This means HDF5 library is not built with parallel I/O support
  - 1). Rebuild HDF5 with --enable-parallel or
  - 2). Setup FLASH application with serialIO

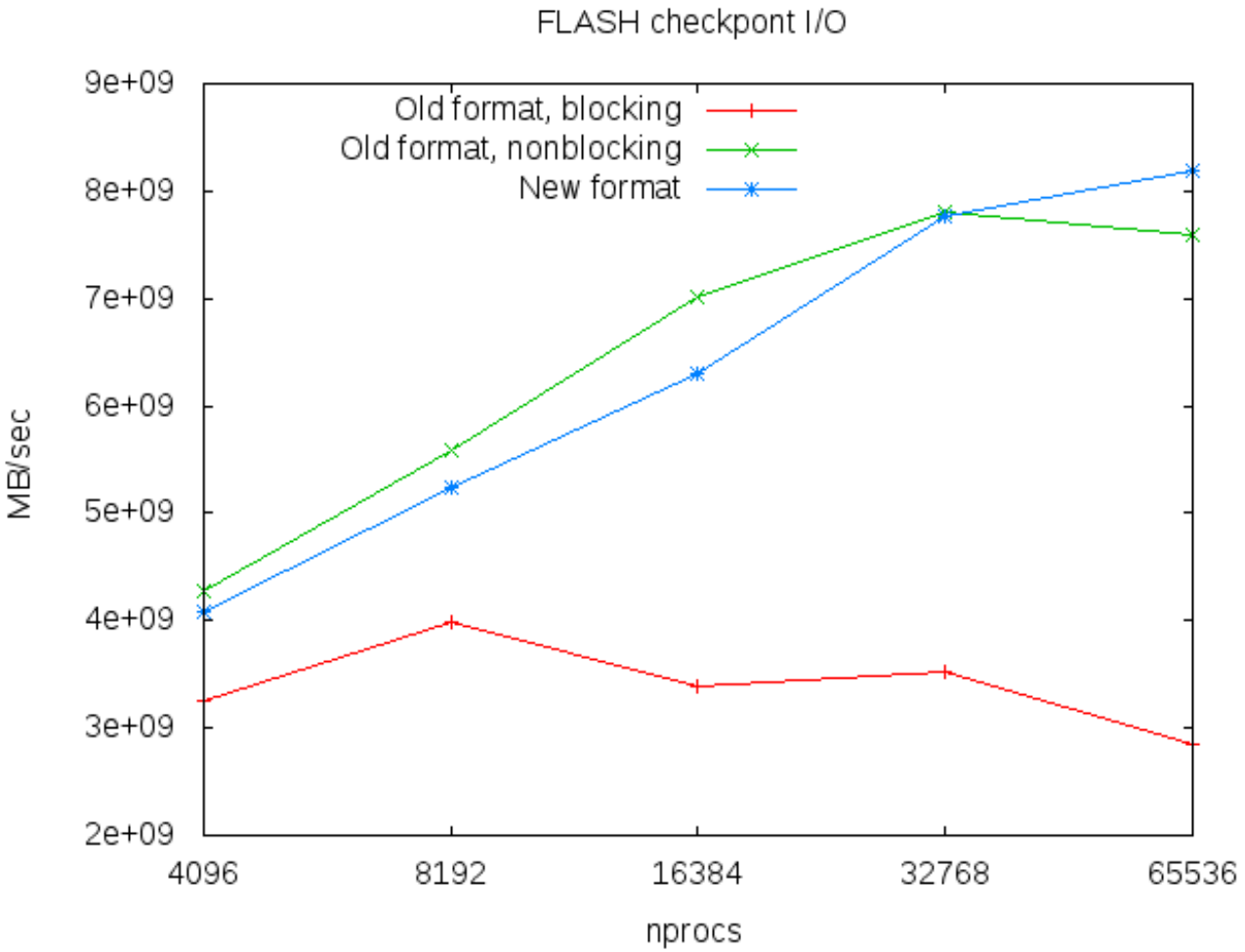
io_h5_attribute.c:48: error: too few arguments to function ‘H5Dopen2’

- This means HDF5 library is version >= 1.8.0, but FLASH only supports v16 API bindings.
  - 1). Add -DH5_USE_16_API to CFLAGS_HDF5 in Makefile.h or
  - 2). Rebuild HDF5 >= 1.8.0 with --with-default-api-version=v16

# Experimental I/O units

- Collaboration with Rob Latham from Argonne National Laboratory
- The key idea is selecting data in memory with MPI derived datatypes (pnetcdf) and HDF5 hyperslabs (HDF5).
  - Default implementation creates identical files to standard FLASH I/O
  - Experimental options for performance
    - A different file layout
    - Non blocking pnetcdf mode
- Explicitly add experimental I/O units at setup time.
  - -unit=IO/IOMain/hdf5/parallel/PM_argonne
  - -unit=IO/IOMain/pnetcdf/PM_argonne
- A bit too raw to be the default I/O implementation, but...

# ...strong scaling results on Intrepid



# Key points

- The default I/O implementation is serial!
  - Generally you should select parallel I/O (with collective optimizations).
- When building your own setup, make sure right units get included
  - Particle I/O is a separate subunit
- The tools quickflash and visit only support FLASH HDF5 files
- Small output files may be compared with FLASH's sfocu tool
  - Used in Flash test suite to identify and quantify regressions

# I/O

Questions?