PARALLEL I/O

HPC.NRW Competence Network

MPI IN SMALL BITES

PREREQUISITES

HPC.NRW

To get the most out of this part of the course you should know and understand

- point-to-point communication
- collective communication
- blocking vs. nonblocking operations
- derived datatypes
- info objects

LEARNING OBJECTIVES

HPC.NRW

At the end of this lesson, you will be able to

- Motivate the use of parallel I/O
- Discuss the advantages and disadvantages of MPI I/O
- Understand common terms of MPI I/O
- Create, open, and close files using MPI I/O
- Access specific parts of a file
- Read and write from and to files using MPI I/O

Introduction

HPC.NRW Competence Network

PARALLEL I/O

MOTIVATION

HPC.NRW

- Scientific datasets become larger with growing simulation detail
- Scientific workflows require data to be read and written from and to persistent storage
- I/O strategies are becoming more important
  - Serialized I/O can easily become a scalability bottleneck
  - Many process-local files may become a meta-data bottleneck

- Efficient parallel I/O strategies are needed

ADVANTAGES OF MPI I/O

HPC.NRW

- Portability
  - Part of MPI standard since 1997
  - Widespread support across MPI library implementations
- Ease of use
  - Abstract interface blends into MPI look & feel
- Efficiency
  - MPI libraries can adapt I/O strategies based on specific platform
  - MPI libraries can implicitly use MPI communication facilities
- High-level interface
  - Coordinated access to files for multiple processes
  - Distributed I/O through collective operations.

DISADVANTAGES OF MPI I/O

HPC.NRW

- Data handling in heterogeneous environments
  - MPI libraries often only support native data representations
    - Big-/Little-endian conversion has to be done by the user
- Focus on binary I/O
  - Text output to shared files using MPI is cumbersome

REQUIREMENTS FOR MPI I/O

- Application needs to use MPI (multi-processing)
  - Serial and multi-threading-only applications don’t use MPI
- Understanding of collective communication
  - File handles work like communicators for file I/O
  - Coordinated file access may be collective
- Handling of non-blocking Operations
  - Overlap computation and I/O (if supported by the MPI library and system)
- Derived Datatypes
  - Non-contiguous file access is specified using MPI derived datatypes

File

An ordered collection of typed data items

Displacement

An absolute byte position relative to the beginning of a file

Offset

A position in the file relative to the current view expressed as a count of elementary types

File view

Process-specific definition of which data of a file is visible to an MPI process, defined by a displacement, elementary type and file type

Elementary type

Smallest addressable data item in a file

File type

Defines which elementary data items are visible to a specific process

File pointer

An offset into the current view (can be process-local and shared)

File handle

An opaque object created by MPI_File_open and closed by MPI_File_close representing a file. All file operations reference an open file through the file handle.

Basic file operations

HPC.NRW Competence Network

PARALLEL I/O

OPENING A FILE

HPC.NRW

MPI_File_open(comm, filename, amode, info, fh**)** [MPI 4.0, p.645]

IN	comm	communicator handle
IN	filename	name of file to open
IN	amode	file access mode
IN	info	info object
OUT	fh	file handle

- File namespace is implementation dependent
- Collective over `comm`
- All processes in `comm` open the same, shared file
  - For process-local files use `MPI_COMM_SELF`
  - All processes must have access to the file (i.e., shared file system)
- Additional information can be passed to MPI library via `MPI_Info` object

ACCESS MODES I

HPC.NRW

- Access mode (amode) is a bit-vector, which is modified using
  - → `|` (bit-wise OR operator) in C
  - → `IOR` (bit-wise OR operator) in Fortran90 and beyond
  - → `+` (Add operator) in Fortran77
- Exactly one of the following modes must be set
  - `MPI_MODE_RDONLY` — read-only access
  - `MPI_MODE_RDWR` — read and write access
  - `MPI_MODE_WRONLY` — write-only access

- Further (optional) modes
  - `MPI_MODE_CREATE` — create file if it does not exist
  - `MPI_MODE_EXCL` — return error if file already exists
  - `MPI_MODE_DELETE_ON_CLOSE` — automatically delete file after closing
  - `MPI_MODE_UNIQUE_OPEN` — Indicate that the file is not opened elsewhere
  - `MPI_MODE_SEQUENTIAL` — Indicate sequential file access (e.g., tapes)
  - `MPI_MODE_APPEND` — automatically set file pointers to end of file

QUERYING THE ACCESS MODE

HPC.NRW

MPI_File_get_amode(fh, amode**)** [MPI 4.0, p.652]

IN	fh	file handle
OUT	amode	access mode used to open the file

- Retrieve access mode from file handle
- Enables libraries to query data even if open call was not intercepted

CLOSING A FILE

HPC.NRW

MPI_File_close(fh**)**[MPI 4.0, p.648]

INOUT fh file handle

- Collective operation on `comm` that was used to open the file
- File is deleted when `MPI_MODE_DELETE_ON_CLOSE` was specified in access mode

- Default error handler is `MPI_ERRORS_RETURN`
  - Return values should be evaluated by the user application
  - I/O errors are usually less catastrophic
    - e.g., file or path does not exist
- Default error handler (for future file handles) can be set using `MPI_FILE_NULL`

DELETING A FILE

HPC.NRW

MPI_File_delete(filename, info**)**[MPI 4.0, p.648]

IN	filename	name of file to delete
IN	info	info object

- Deletes file identified by **filename**
- Returns `MPI_ERR_NO_SUCH_FILE` if file does not exist
- Additional info can be given in **info** object (e.g., file-system specifics)

RESIZE A FILE

HPC.NRW

MPI_File_set_size(fh, size**)**[MPI 4.0, p.649]

INOUT	fh	file handle
IN	size	size to truncate or expand file

- Collective call
- Sets a new size for the file ($\rightarrow$ meta-data operation)
  - If now smaller, file is truncated
  - If now larger, old data is preserved and new data is undefined
- Implementation-defined whether new space is preallocated
- Does not affect individual file pointers
- Erroneous for sequential files (i.e., opened with `MPI_MODE_SEQUENTIAL`)
- Erroneous when non-blocking operations are pending for file

PREALLOCATE A FILE

HPC.NRW

MPI_File_preallocate(fh, size**)**[MPI 4.0, p.650]

INOUT	fh	file handle
IN	size	size to preallocate file

- Collective call
- Allocate storage space for file ($\rightarrow$ meta-data operation)
  - Previously written data is unaffected
  - Potentially new space is undefined
  - File size unchanged if given size is smaller than current file size
- Erroneous for sequential files (i.e., opened with `MPI_MODE_SEQUENTIAL`)
- Erroneous when non-blocking operations are pending for file

QUERYING THE FILE SIZE

HPC.NRW

MPI_File_get_size(fh, size**)** [MPI 4.0, p.651]

IN	fh	file handle
OUT	size	size of the file in bytes

- Query the current size in bytes
  - E.g., for data partitioning
- Consistency-wise a data access operation

QUERYING THE PROCESS GROUP OF A FILE HANDLE

HPC.NRW

MPI_File_get_group(fh, group**)**[MPI 4.0, p.651]

IN	fh	file handle
OUT	group	group of processes which share this file handle

- Returns a duplicate of the group of the communicator used to open the file
- User is responsible for freeing the group (using `MPI_Group_free`)
- Collective I/O operations are collective over the process group represented by `group`

SUMMARY

HPC.NRW

- Errors in MPI I/O are non-fatal by default
- Files are referenced in file operations a file handle
- Operations to open, close, resize files are available
- Files can be deleted via filename

File views

HPC.NRW Competence Network

PARALLEL I/O

LEARNING OBJECTIVES

HPC.NRW

At the end of this lesson, you will be able to

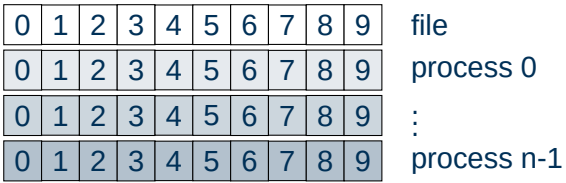
- Understand how view provide access to a file
- Create distinct views on a file in MPI I/O
- Use consecutive views for multi-part files

FILE VIEWS

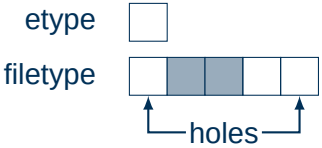
HPC.NRW

- Each process has view connected to each file handle
- A view determines the visible parts of a file to a process
- A view is defined by a displacement, an elementary type, and a file type
- Can be set by `MPI_File_set_view`
- Can be queried by `MPI_File_get_view`
- A filetype must be a collection of elementary-typed items or elementary-type sized gaps

- Each handle always has a view attached to it
- The default view comprises
  - A displacement of `0` (start of the file)
  - An elementary type of `MPI_BYTE` (no typed structure)
  - A file type of `MPI_BYTE` (full access to all bytes of a file)



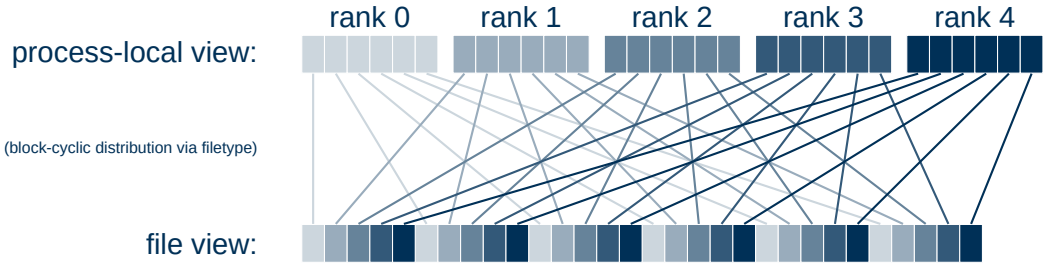
– A filetype needs to be constructed from one of more elementary types



– The resulting file layout is an infinite concatenation of filetype starting a s displacement

THE COMPLEXITIES OF FILE ACCESS ABSTRACTION

HPC.NRW



THE COMPLEXITIES OF FILE ACCESS ABSTRACTION

HPC.NRW

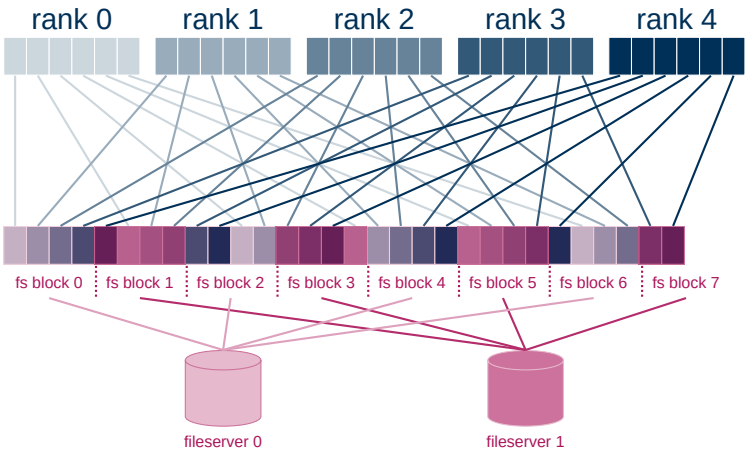
process-local view:

(block-cyclic distribution via filetype)

file view:

filesystem view:

(filesystem striping across multiple fileservers)



SET A FILE VIEW I

HPC.NRW

MPI_File_set_view(fh, disp, etype, filetype, datarep, info**)** [MPI 4.0, p.656]

INOUT	fh	file handle
IN	disp	displacement
IN	etype	elementary datatype
IN	filetype	filetype
IN	datarep	data representation
IN	info	info object

- The displacement specifies the byte position of the first elementary type of the view
- The elementary type specifies the core structure of the view
  - Use distinct subsequent views for differently structured parts of a file
- The filetype specifies the visible elementary types of the view
  - Holes in the filetype are inaccessible and invisible to the current process
- Data representation provides information on file interoperability

GET THE CURRENT FILE VIEW

HPC.NRW

MPI_File_get_view(fh, disp, etype, filetype, datarep, info)**)** [MPI 4.0, p.656]

IN	fh	file handle
OUT	disp	displacement
OUT	etype	elementary datatype
OUT	filetype	filetype
OUT	datarep	data representation

- Get logical file structure (displacement, elementary type, filetype) of current view
- Get current data representation set for view

native Data is stored in the file as it is in memory. Best precision and performance. No guarantee of portability.

internal Data is stored in the file using an implementation-specific format. Data conversion may affect precision and performance. No portability guarantees across MPI implementations.

external32 Data is stored in the file using a standardized format. Data conversions may affect precision and performance. Portable across any platform and implementation.

SUMMARY

HPC.NRW

- File access in MPI I/O is determined by a view
- Default view provides access to the full file for all processes on byte level (i.e., MPI_BYTE)
- Views can be used to write data to non-contiguous regions of the file
- MPI provides a logical view on the file, physical view may differ
- Data representations can provide portability if needed.

File access

HPC.NRW Competence Network

PARALLEL I/O

LEARNING OBJECTIVES

HPC.NRW

At the end of this lesson, you will be able to

- Identify which positioning options exist with MPI I/O
- Know the advantages and disadvantages of different positioning options

TAXONOMY OF DATA ACCESS ROUTINES

HPC.NRW

- Positioning
  - Individual file pointers do no use a special suffix
  - Explicit offsets use suffix `_AT`
  - Shared file pointers use suffixes `_SHARED` and `_ORDERED`
- Synchronism
  - Blocking calls have no special prefix
  - Non-blocking calls use `i` prefix
  - Split-collective calls use `(_begin|_end)` suffix
- Coordination
  - Individual calls have no special modifier
  - Collective calls use the `_ALL` suffix

- Individual file pointers
  - Each process has its own file pointer that is only modified by accesses of that specific process
- Shared file pointers
  - Single file pointer for all processes that share the file handle
  - Modifies by any shared file pointer access
  - Can only be used when all processes have access to the same parts of a file
- Explicit offset access
  - No file pointer used or modified
  - Explicit offset is given with each access to provide position
  - Cannot be used with `MPI_MODE_SEQUENTIAL`

COORDINATION WITH DATA ACCESS

- Individual access (like POSIX)
  - Each process accesses the file without coordination with others
  - Good when I/O phases are skewed among processes
- Coordinated access
  - Collective calls enable transparent collective communication
    - Collect data to avoid file system contention
    - Distribute collected data to maximize I/O bandwidth

- Blocking and Nonblocking interfaces available
  - Individual file access
    - Similar look & feel to nonblocking point-to-point communication
  - Collective file access
    - Similar to collective communication for **explicit offsets** and **individual file pointers**
    - Split-collective (begin/end, **no test!**) for **shared file pointers**
- Potentially overlap I/O transfers with computation or communication

OVERVIEW OF DATA ACCESS ROUTINES

HPC.NRW

		coordination	
positioning	**synchronism**	**noncollective**	**collective**
explicit offsets	blocking	MPI_File_(read|write)_at	MPI_File_(read|write)_at_all
	nonblocking	MPI_File_i(read|write)_at	MPI_File_i(read|write)_at_all
	split collective	N/A	MPI_File_(read|write)_at_all_begin MPI_File_(read|write)_at_all_end
individual file pointers	blocking	MPI_File_(read|write)	MPI_File_(read|write)_all
	nonblocking	MPI_File_i(read|write)	MPI_File_i(read|write)_all
	split collective	N/A	MPI_File_(read|write)_all_begin MPI_File_(read|write)_all_end
shared file pointers	blocking	MPI_File_(read|write)_shared	MPI_File_(read|write)_ordered
	nonblocking	MPI_File_i(read|write)_shared	N/A
	split collective	N/A	MPI_File_(read|write)_ordered_begin MPI_File_(read|write)_ordered_end

Consistency and Semantics

HPC.NRW Competence Network

PARALLEL I/O

- Operations provide sequential consistency semantics
  - Operations outcome will be as if performed in some serial order consistent with program order.
  - Each access appear atomic (if atomic mode is set)
  - Exact ordering of accesses is unspecified
- User-imposed consistency may be obtained using program order and call to MPI_File_sync
- A matched pair of split-collective accesses compose a single data access operation
- A completed non-blocking data access (initiation and completion) compose a single data access operation

MPI_File_set_atomicity(fh, flag**)**[MPI 4.0, p.711]

INOUT	fh	file handle
IN	flag	true to set atomic mode, false to set nonatomic mode

- Collective call
- All processes must pass the same value
- Changing consistency for an open file handle affects new data access

MPI_File_get_atomicity(fh, flag**)**[MPI 4.0, p.711]

IN	fh	file handle
OUT	flag	true if atomic mode, false if nonatomic mode

- Noncollective call

ENSURING FULL COMPLETING OF PENDING I/O OPERATIONS

HPC.NRW

MPI_file_sync(fh**)**[MPI 4.0, p.712]

INOUT fh file handle

- Ensure all previous writes to file handle by the calling process are transferred to the storage device
- Updates by other processes to the storage device become visible to the local process reads

Using Info hints

HPC.NRW Competence Network

PARALLEL I/O

MPI INFO OBJECT RECAP[MPI 4.0, ch.10]

HPC.NRW

- Container for string-based key-value pairs
- Key-value pairs are hints to the MPI library
- MPI library may ignore the hint
- User must comply with the hints they provide
- Some hints can be set only once (e.g., at handle creation)
- Other hints may be changed several times at runtime

EXAMPLE OF INFO OBJECT CREATION

C/C++

HPC.NRW

```
MPI_Info info;
MPI_Info_create(&info);
MPI_Info_set(info, "my_key", "my_value");
...
// use info object
...
MPI_Info_free(&info);
```

ADVANTAGES AND DISADVANTAGES OF INFO HINTS

HPC.NRW

- Advantages
  - + Application can communicate detail about behavior to MPI library
    - Significant potential for performance improvements (when default does not perform well)
  - + Application can react to settings communicated by MPI library
- Disadvantages
  - No guarantee that MPI library uses a hint
  - Not all MPI libraries support all hints specified in MPI standard
  - Some MPI library support special hints for individual filesystems

MPI_File_set_info(fh, info**)**[MPI 4.0, p.653]

INOUT	fh	file handle
IN	info	info object

- Sets keys provided in info.
- No effect on default or previously set keys not present in info

MPI_File_set_info(fh, info**)**[MPI 4.0, p.653]

IN	fh	file handle
OUT	info	new info object

- Returns a new info object
- Get keys currently set for file handle
- MPI implementation must return all supported values
  - Query default values

LIST OF SELECTED MPI I/O INFO KEYS

(see full list at [MPI 4.0, pp.655])

HPC.NRW

Key	Type	Description
collective_buffering	boolean	Enable/disable collective buffering
cb_block_size	integer	Block size used for collective buffering
cb_buffer_size	interger	Total buffer space per target node
cb_nodes	integer	Number of target nodes used for collective buffering
striping_factor	integer	Number of fileservers to use per file
striping_unit	integer	Size of each filesystem stripe

LIST OF SELECTED ROMIO INFO KEYS

HPC.NRW

Key	Description
romio_cb_read	Enable/disable collective buffering for reads
romio_cb_write	Enable/disable collective buffering for writes
romio_no_indep_rw	Indicate whether independent reads and writes will occur
romio_cb_ds_threshold	Threshold in bytes on when to enable data sieving
romio_cb_alltoall	Enable/disable collective buffering for alltoall
ind_rd_buffer_size	Buffer size for data sieving on reads
ind_wr_buffer_size	Buffer size for data sieving on writes
romio_ds_read	Enable/disable data sieving on reads
romio_ds_write	Enable/disable data sieving on writes
romio_filesystem_type	Filesystem a file is created on
romio_aggregator_list	List of target nodes (processes) to choose as aggregators