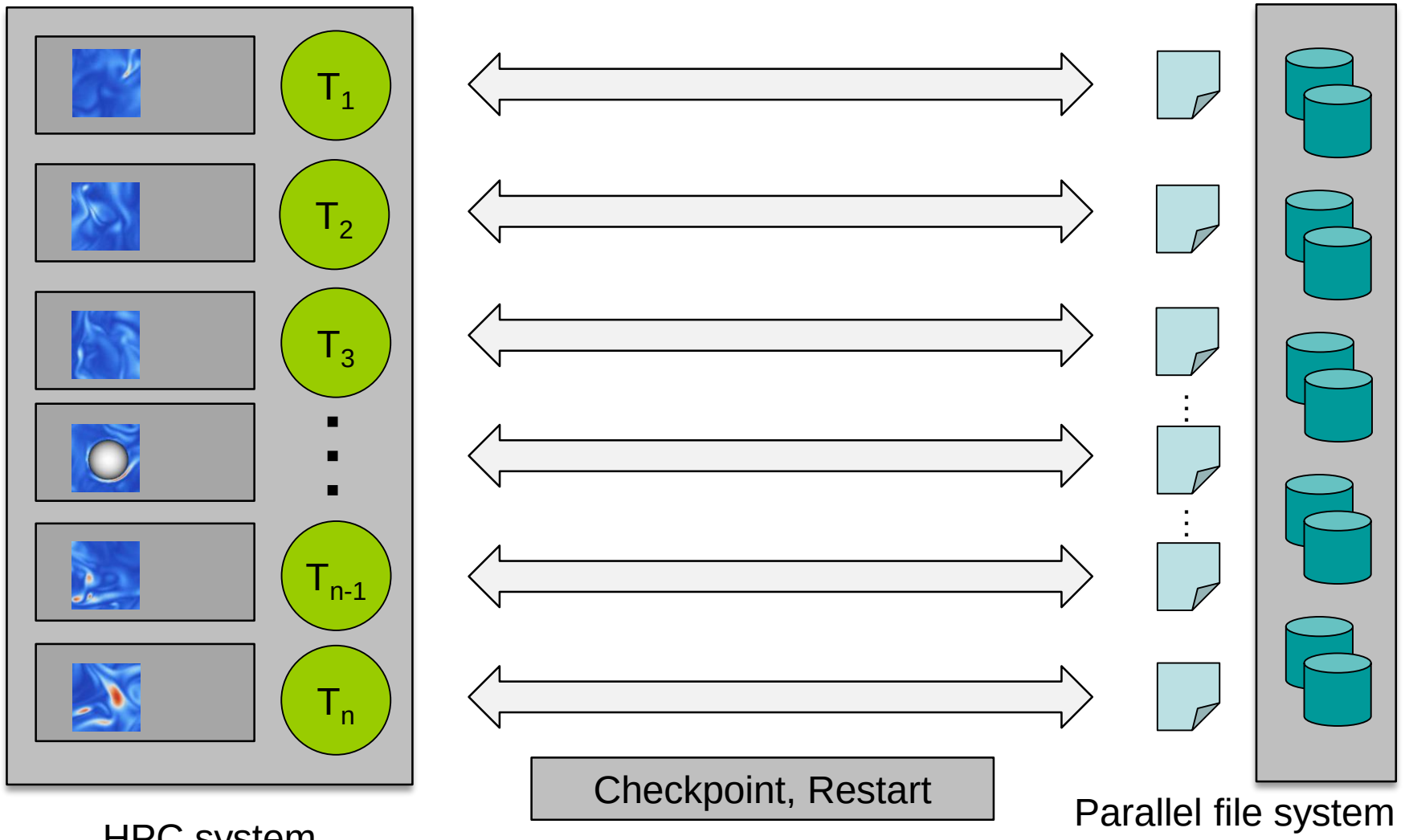
PARALLEL I/O AND PORTABLE DATA FORMATS

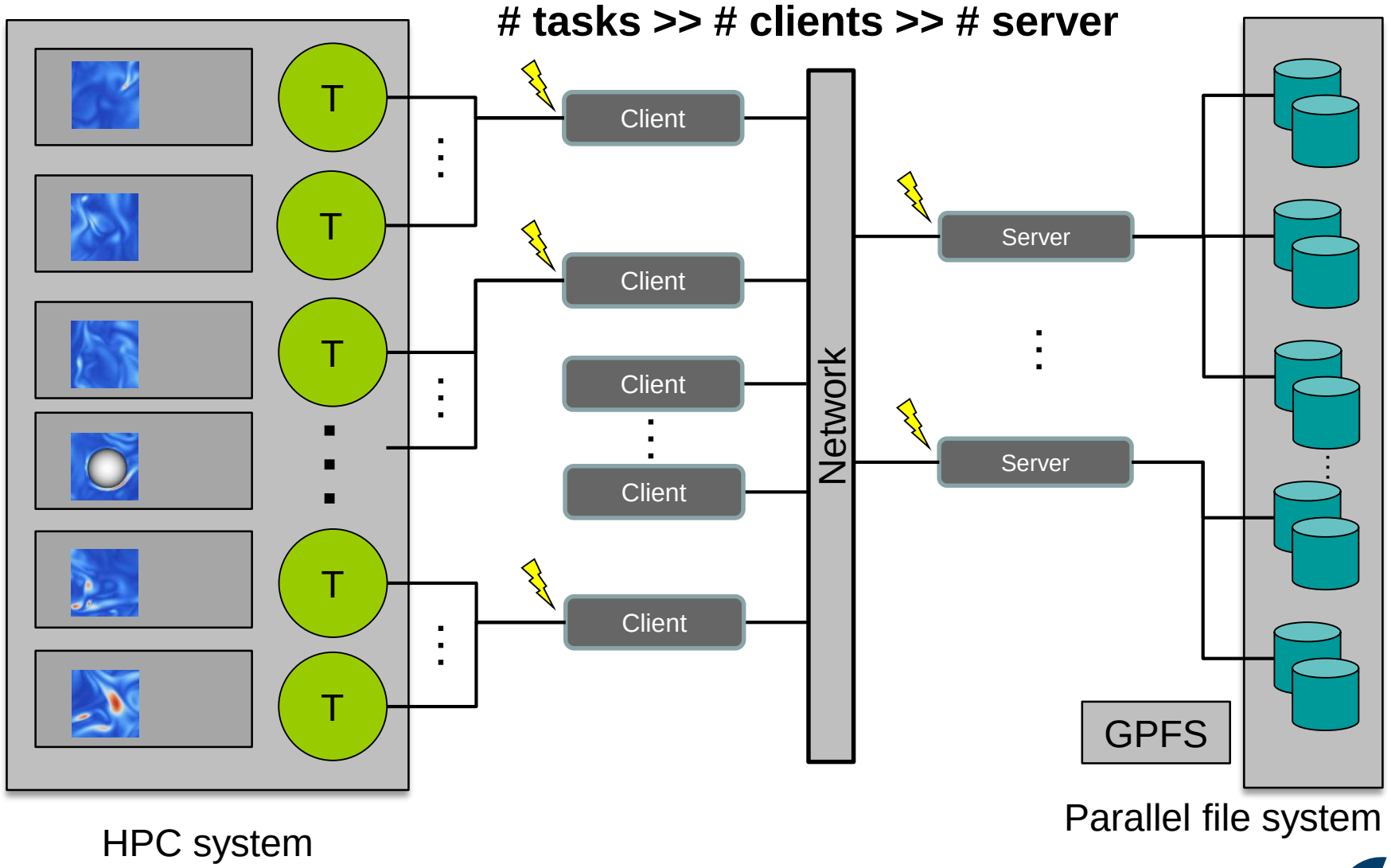
SCALABLE I/O FOR PARALLEL ACCESS TO TASK-LOCAL FILES WITH SIONLIB

23.02.2022 I SEBASTIAN LÜHRS (S.LUEHRS@FZ-JUELICH.DE)

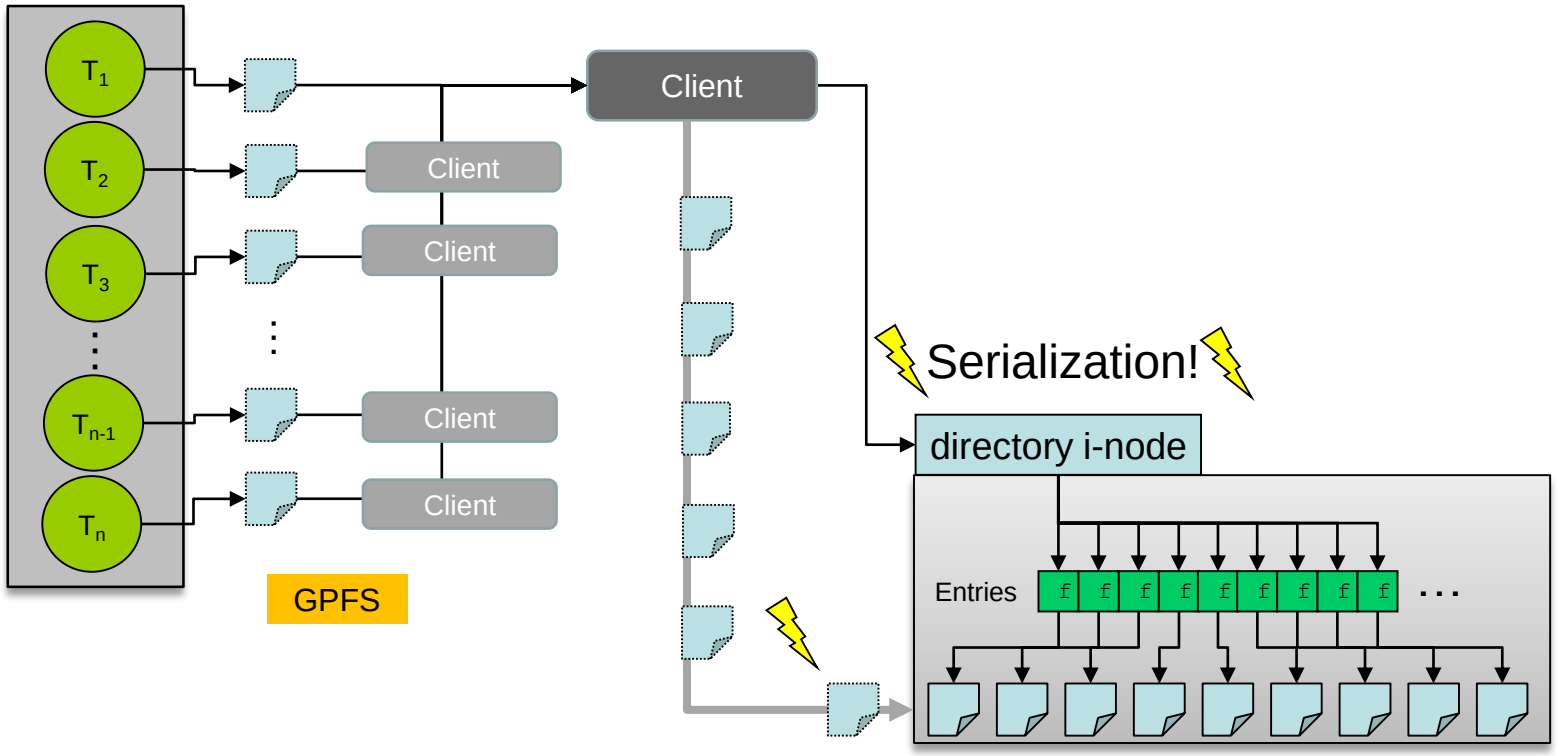
Task-Local I/O



Task-Local I/O

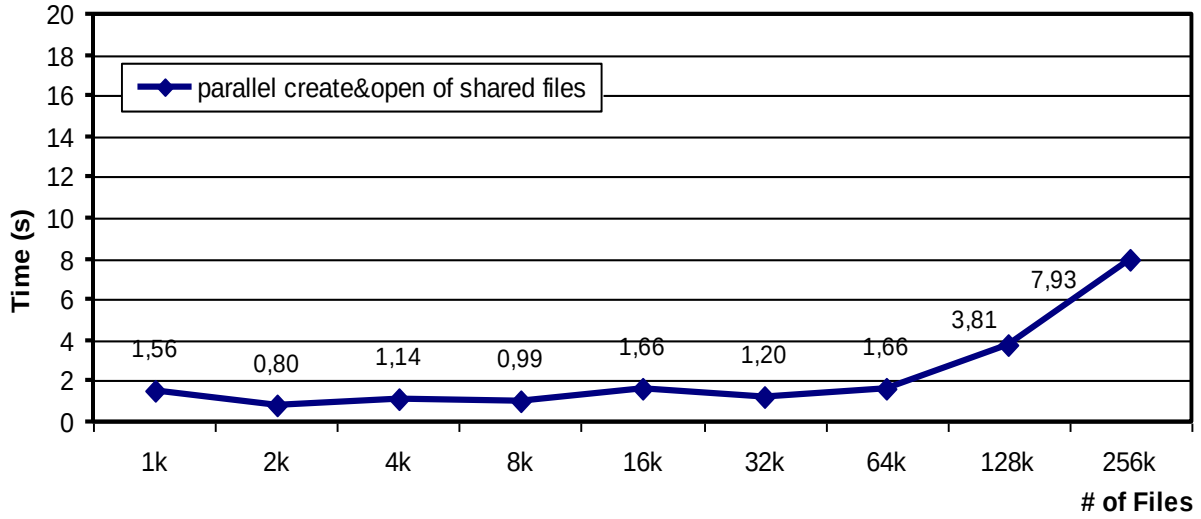


I/O Bottleneck: Parallel File Creation

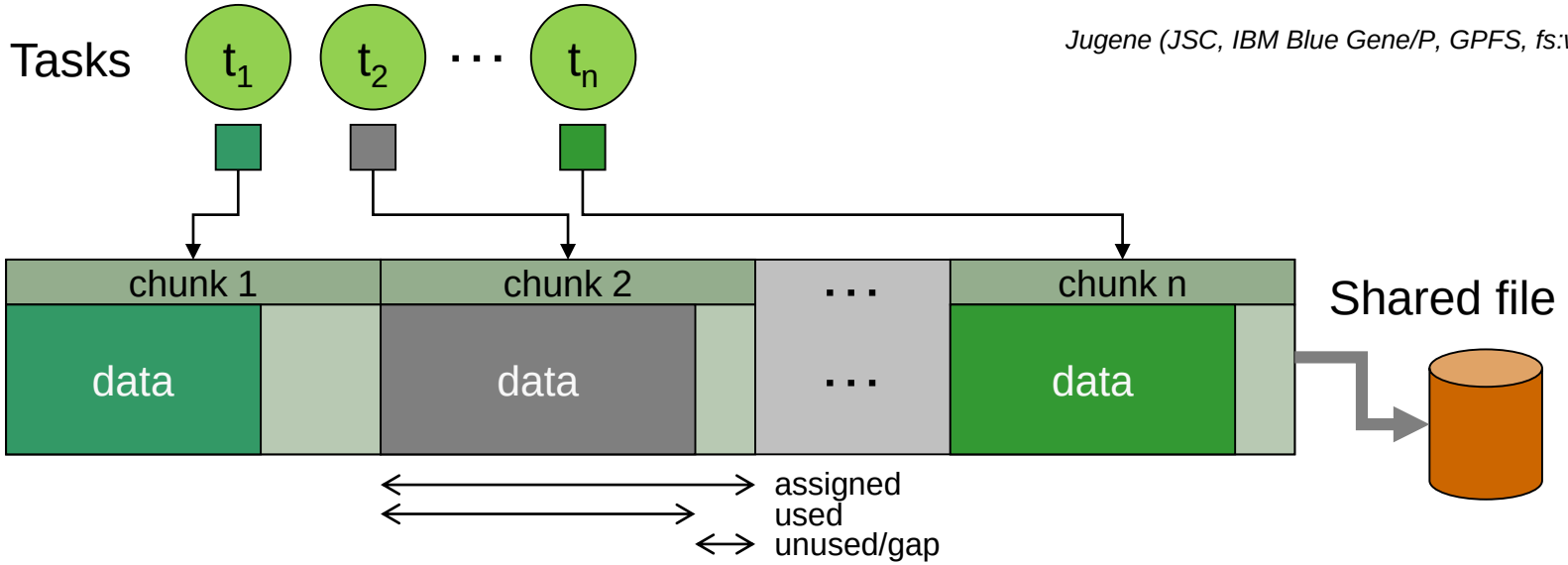


File Format (1): a Single Shared File

- Create and open is fast
- Simplified file handling
- Only one big file
- Logical partitioning required

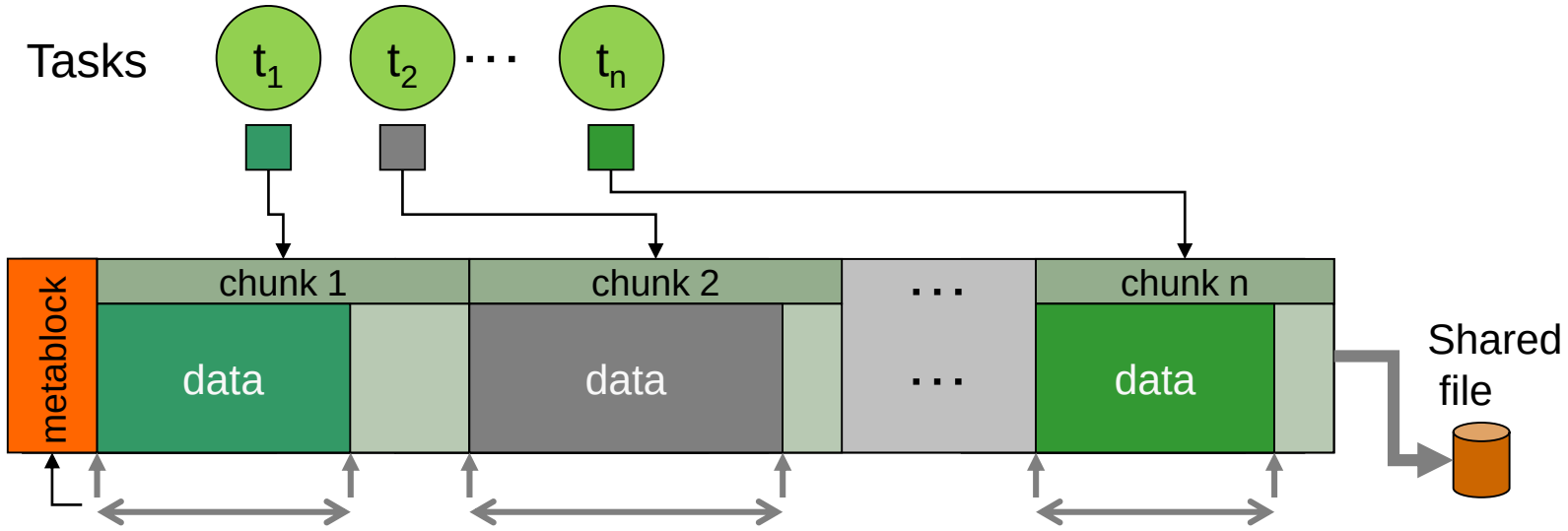


Jugene (JSC, IBM Blue Gene/P, GPFS, fs:work)



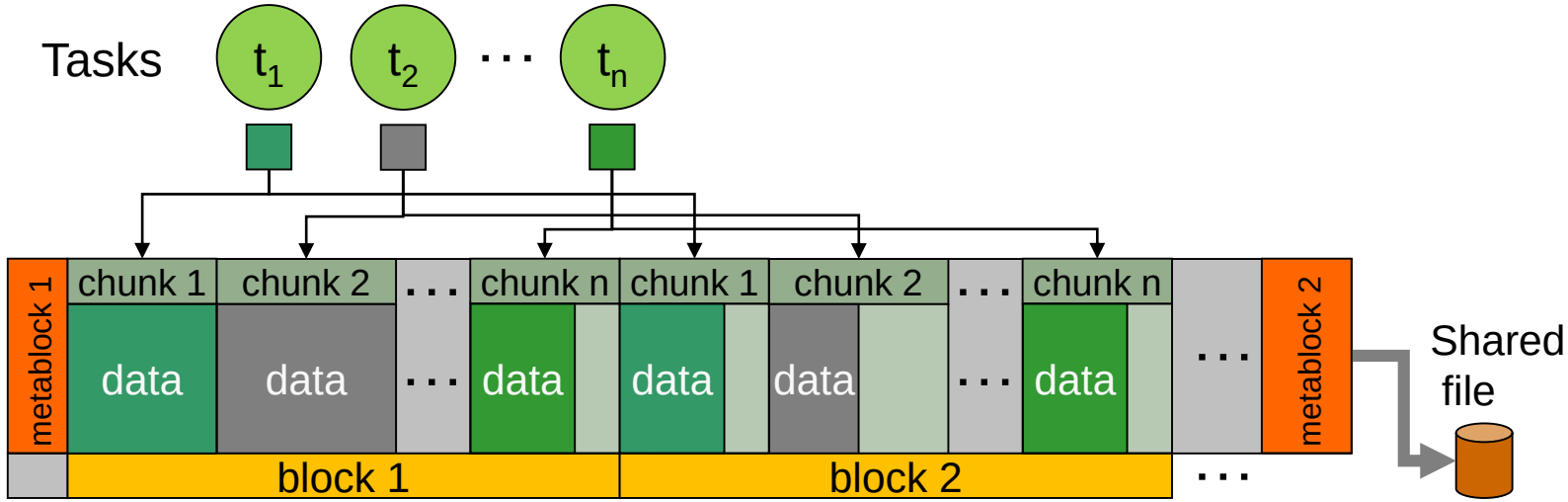
File Format (2): Metadata

- Offset and data size per task
- Tasks have to specify chunk size in advance
- Data must not exceed chunk size



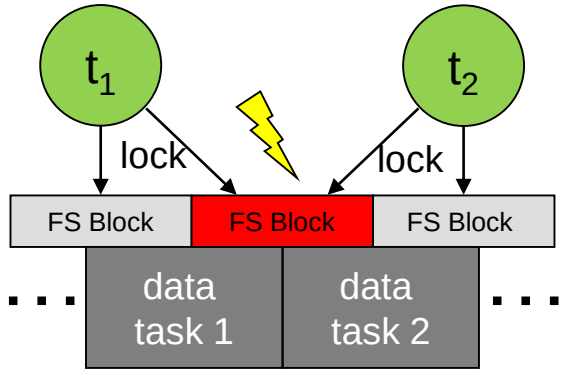
File Format (3): Multiple Blocks of Chunks

- Enhancement: define blocks of chunks
- Metadata now with variable length (#tasks * #blocks)
- Second metadata block at the end
- Data of one block does not exceed chunk size



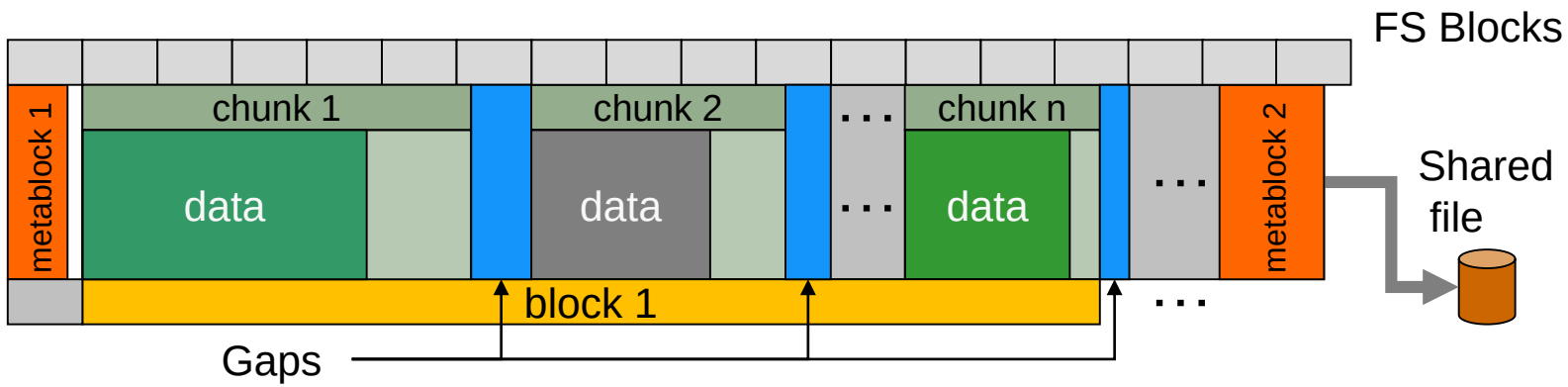
File Format (4): Alignment to Block Boundaries

- Contention when writing to same file-system block in parallel



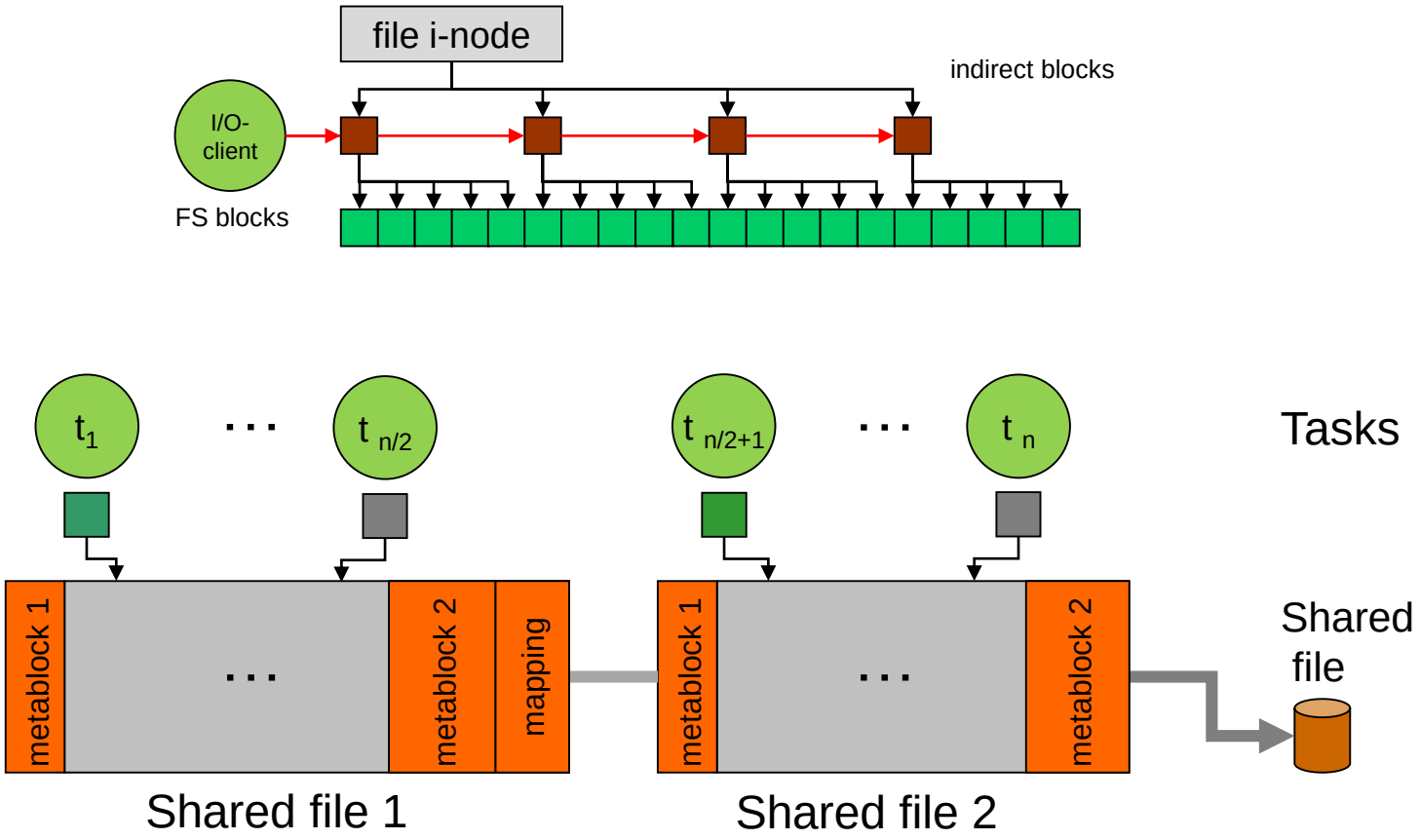
#tasks	data size	blksize	write bandwidth
32768	256 GB	aligned	**3650.2** MB/s
32768	256 GB	not aligned	**1863.8** MB/s

Jugene (JSC, IBM Blue Gene/P, GPFS, fs:work)

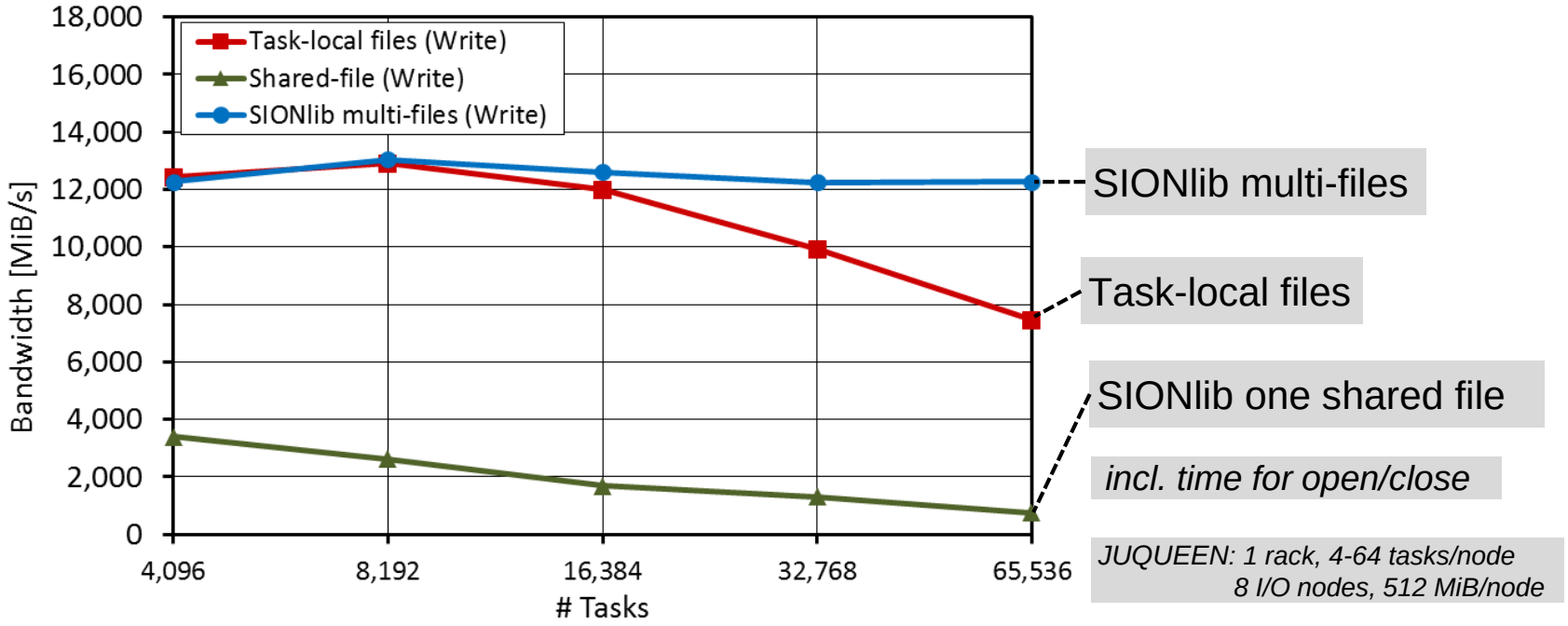


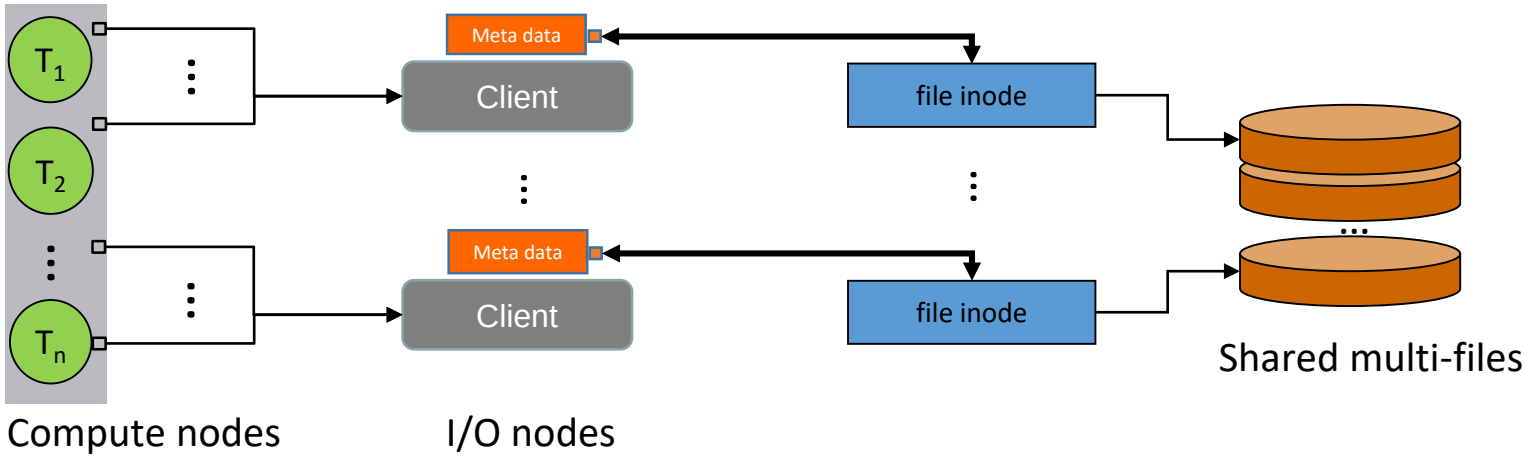
File Format (5): Multiple Physical Files

- Variable number of underlying physical files
- Bandwidth degradation on GPFS by using a single shared file

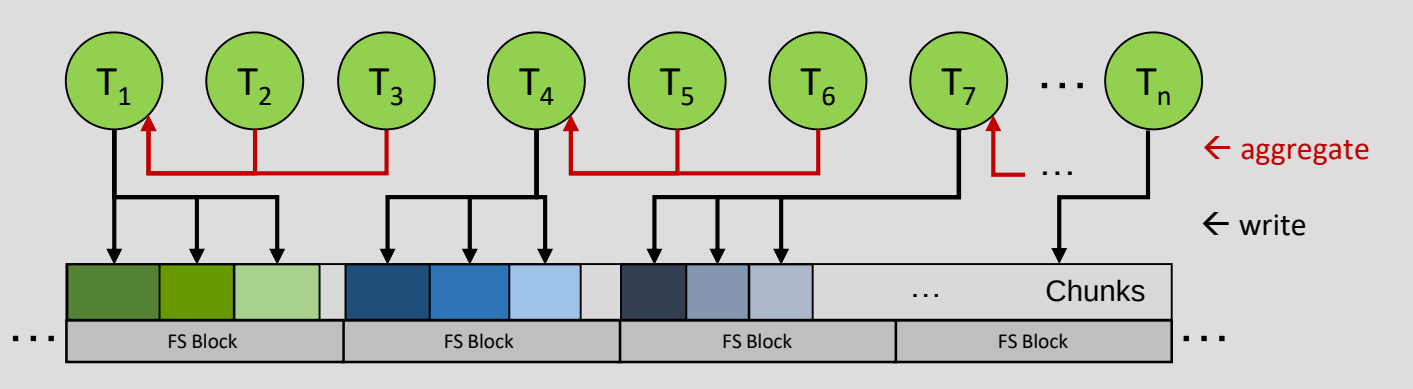


I/O Bottleneck: Large #tasks & Shared fileS





File Format (6): Coalescing I/O



- data/task very small → sparse file container
- Coalescing I/O: Aggregation of data among tasks
- Variable number of senders/collectors
- Collective write/read operations required
- Advantages: No alignment between chunks of the same collector; less gaps, fewer data streams, less congestion in I/O infrastructure
- Reduced buffering on collector task (one file system block)

Version, Download, Installation

- Version: 1.7.7 (July 2021), Version 2.0.0-rc.3 (February 2021)
- Open-source license
- https://www.fz-juelich.de/jsc/sionlib
- *sionlib_jsc@fz-juelich.de*

References:
- Wolfgang Frings, Felix Wolf, Ventsislav Petkov, **Scalable Massively Parallel I/O to Task-Local File**, Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, Portland, Oregon, November 14-20, 2009, SC'09, New York, ACM, ISBN 978-1-60558-744-8.
- Wolfgang Frings, **Efficient Task-Local I/O Operations of Massively Parallel Applications**, Schriften des Forschungszentrums Jülich, IAS Series, 30, 2016, ISBN 978-3-95806-152-1

Header Files & Datatypes

C
```
#include <sion.h>
```

Fortran
```
use sion_f90
use sion_f90_mpi
```

Special datatypes are typically used for all parameters that are used to describe or to compute file positions

C
```
sion_int32
```

Fortran
```
INTEGER(kind=4)
```

C
```
sion_int64
```

Fortran
```
INTEGER(kind=8)
```

Collective Open (MPI)

C

```
int sion_paropen_mpi (char  *fname, const char *file_mode,
                      int *numFiles,
                   MPI_Comm gComm,   MPI_Comm *lComm,
                 sion_int64 *chunksize,
                 sion_int32 *fsblksize,
                      int *globalrank,
                     FILE **fileptr, char **newfname);
```

Fortran

```
FSION_PAROPEN_MPI (FNAME, FILE_MODE, NUMFILES,
                   GCOMM, LCOMM, CHUNKSIZE, FSBLKSIZE,
                   GLOBALRANK, NEWFNAME, SID)
CHARACTER*(*) FNAME, FILE_MODE, NEWFNAME
INTEGER       NUMFILES, FSBLKSIZE, GLOBALRANK, SID
INTEGER       GCOMM, LCOMM
INTEGER*8     CHUNKSIZE
```

- Open a SION file in parallel for reading or writing data
- Collective call, called by each task at the same time
- Accesses one or more physical files of a logical SION file
- Parameters are passed “by reference” to pass back information in read open mode

Parameters of Open Calls I

fname: file name

- Character string describing path and file name
- Will not be extended by SION-specific suffix
- In general multiple physical files are generated
- First file: **filename**
- All other files: **filename** + “.” + 6-digit-number (000001 ...)
- All commands and function calls use the base name

file_mode: file mode

- r,rb,br (read block), open existing SION file for reading
- w,wb,bw (write block), create a new SION file, open for write; overwrite if existing
- posix use internally POSIX interface for file access, otherwise ANSI-C

Parameters of Open Calls II

sid: SIONlib file descriptor

- Unique integer value, referring internally to data structure
- Associated to SION file (internal file handle)
- Allows multiple simultaneously opened files
- C: return code, Fortran: last parameter of open call
- Integer file handle for Fortran necessary

chunksize: size of data per task

- Pointer of type: sion_int64* (C), Integer*8 (Fortran)
- Size of data in bytes written by this tasks (maximum size of single sion_fwrite call)
- May be different for each task and must be set if open for writing
- Will be increased internally to the next multiple of the file system block size

Parameters of Open Calls III

fsblksize: file system block size

- Size of file system block in bytes
- Read-mode: file system block size at write time
- Write-mode: automatically detected by SIONlib if set to -1 *(recommended)*

fileptr: ANSI-C file pointer

- Can be replaced by NULL pointer if not needed *(recommended)*
- Will be removed in future versions
- Only needed if wrapper functions for writing and reading are not used
- Not available in Fortran

newfname:

- File name of physical file assigned to this task, NULL can be used

Parameters of sion_paropen_mpi

gComm: MPI Communicator

- Call is collective over all tasks of this communicator
- Each task gets assigned one chunk of SION file
- Read: number of tasks must be equivalent to number tasks written to SION file

lComm: MPI Communicator

- Tasks of the same communicator are writing to the same physical file
- MPI_Comm_null if not specified, otherwise Union of lComm must equal gComm

numfiles: Number of physical files

- If using lComm: set numfiles=-1
- Read-mode: parameters will be set by open call

globalrank:

- Rank of task in global communicator gComm

Serial Open

C

```
int sion_open (char         *fname,
               const char   *file_mode,
               int          *ntasks,   int *nfiles,
               sion_int64 **chunksizes,
               sion_int32   *fsblksize,
               int **globalranks,  FILE **fileptr);
```

Fortran

```
FSION_OPEN (FNAME, FILE_MODE, NTASKS, NUMFILES,
            CHUNKSIZES, FSBLKSIZE, GLOBALRANKS, SID)
CHARACTER*(*) FNAME, FILE_MODE
INTEGER       NUMFILES, NTASKS, FSBLKSIZE, SID
INTEGER       GLOBALRANKS(NTASKS)
INTEGER*8     CHUNKSIZES(NTASKS)
```

- All chunks of all tasks can be selected, via `sion_seek` (does not work in writing mode)
- Multiple physical files can be handled
- Reads all metadata of all tasks into memory

Collective Close

C
```
int sion_parclose_mpi(int sid)
```

Fortran
```
FSION_PARCLOSE_MPI (SID, IERR)
INTEGER SID, IERR
```

- Closes a SION file in parallel on all tasks/threads
- Collective call, called by each task/thread at the same time
- Metadata will be collected from each tasks
- Metadata blocks of SION file will be written in this call

Serial Close

C
```
int sion_close(int sid)
```

Fortran
```
FSION_CLOSE(SID, IERR)
INTEGER SID, IERR
```

- Closes a SION file in serial mode
- Metadata blocks of SION file will be written in this call

Exercise

Exercise 1 – SIONlib hello world

- Directory preparation:
  - `/p/project/training2202/sionlib/copy.sh`
  - `cd /p/project/training2202/<username>/sionlib`

- Write a parallel program in C or Fortran which creates and closes an empty SIONlib file
- Use a chunksize to store 1000 Integer
- Use the template file `exercise_1.c` or `exercise_1.f90`

```
module load Intel ParaStationMPI # Load compiler and MPI
module load SIONlib # Load SIONlib libs

mpicc exercise_1.c `sionconfig --libs --mpi`
mpif90 `sionconfig --cflags --mpi --f90` exercise_1.f90 `sionconfig --libs
--mpi --f90`
```

- To start a job on the compute node:

```
srun -N 1 --ntasks-per-node=48 --reservation=pario-2022-02-23
--account=training2202 --time=00:02:00 ./a.out
```

Check details of the resulting file using:

```
siondump <sionlib_output_file>
```

Write Data

C

```
size_t sion_fwrite (void *data,
                    size_t size,
                    size_t nmemb,
                    int    sid);
```

Fortran

```
FSION_WRITE (DATA, SIZE, NMEMB, SID, RC)
INTEGER SID
INTEGER*8 SIZE,NMEMB, RC
```

- Write `size*nmemb` bytes of data, beginning from current position
  - This size must not exceed the chunksize defined in open call!
- Returns number of elements written

Exercise

Exercise 2 – SIONlib write

- Extend your parallel program in C or Fortran
- Each task should fill a array with 1000 elements using its unique rank
- Each task should write the array into the prepared SIONlib file
- Use the template file `exercise_2.c` or `exercise_2.f90`

Check the resulting file using:

```
siondump <sionlib_output_file>
```

Read Data

C

```
size_t sion_fread (void  *data,
                   size_t size,
                   size_t nmemb,
                   int    sid);
```

Fortran

```
FSION_READ (DATA, SIZE, NMEMB, SID, IERR)
INTEGER SIZE, NMEMB, SID, IERR
```

- Read `size*nmemb` bytes from current position in chunk
- Cannot read more than a chunk
  - `sion_bytes_avail_in_block` must be used to get all available data
- Returns number of elements read

End of File

C
```
int sion_feof(int sid);
```

F
```
FSION_FEOF (SID, IERR)
INTEGER SID, IERR
```

- Internally this function flushes all buffer and checks current positions against chunk boundaries
- Moves file pointer to next chunk if end of current chunk is reached
- The function is a task-local function, which can be called independently from other MPI tasks.
- Returns 1 if pointer is behind last byte of data for this task

Seek: Change File Position

C

```
int sion_seek (int         sid,
               int         rank,
               int         chunknr,
               sion_int64 posinchunk );
```

Fortran

```
FSION_SEEK (SID, RANK, CHUNKNUM, POSINCHUNK, IERR)
INTEGER     SID, RANK, CHUNKNUM, IERR
INTEGER*8   POSINCHUNK
```

- Sets the file pointer to a new position, only available in reading mode
- Seek parameters:
  - `rank`: rank number (0,...), or SION_CURRENT_RANK
  - `chunknum`: chunk number (0,...), or SION_CURRENT_BLK
  - `posinchunk`: position (0,...), or SION_CURRENT_POS

Command line tools

- siondump: Show metadata of file
- sionsplit: Split SIONlib file into task-local files
- sioncat: Extract data from file
- siondefrag: Contracting all of the chunks of a task, which are spread in the file over multiple blocks, into a single chunk
- sionconfig: Get compile and link options

SIONlib 2.0.0

- Currently `2.0.0-rc.3`
- Continuous read and write:
  - `sion_fread` can read without limitations, even if the amount of data requested spans multiple chunks
  - `sion_fwrite` can write without limitations, even if the amount of data specified spans multiple chunks
- POSIX file pointer access is removed
- Simplified open functions:

```
int sion_open(const char *name, sion_open_mode mode, int n, const sion_options *options);
```

- New seek functions
- Removal of deprecated items
- New build system
- See also porting guide: https://apps.fz-juelich.de/jsc/sionlib/docu/2.0.0-rc.3/porting-2.html

Exercise

Exercise 3 – Mandelbrot set

- Run the example for the master-worker decomposition (`type = 2`) of the Mandelbrot example using SIONlib (`mandelsion.c` or `mandelsion.f90`)

Hints

Options for running the program:

`-t 2` (master-worker decomposition)

`-f 0` (use SIONlib)