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

28.01.2020 I BENEDIKT STEINBUSCH
JÜLICH SUPERCOMPUTING CENTRE

SIONLIB FACT SHEET

Data model: *n* sequences of bytes (untyped data)

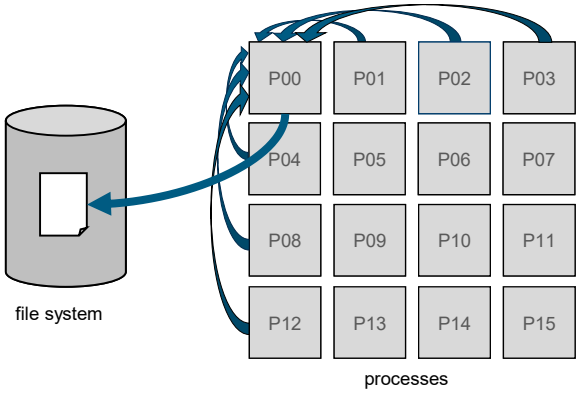
Self describing: no

Full control of file content: no

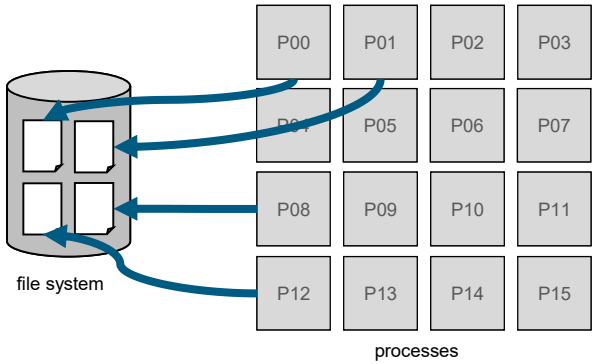
Use cases:

- Data that is by its nature per-task (performance data, logs, …)
- Data for internal use (checkpoints for restarting)

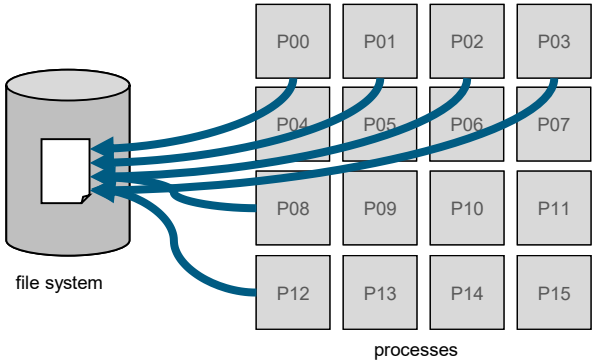
PARALLEL I/O STRATEGIES



One process performs I/O

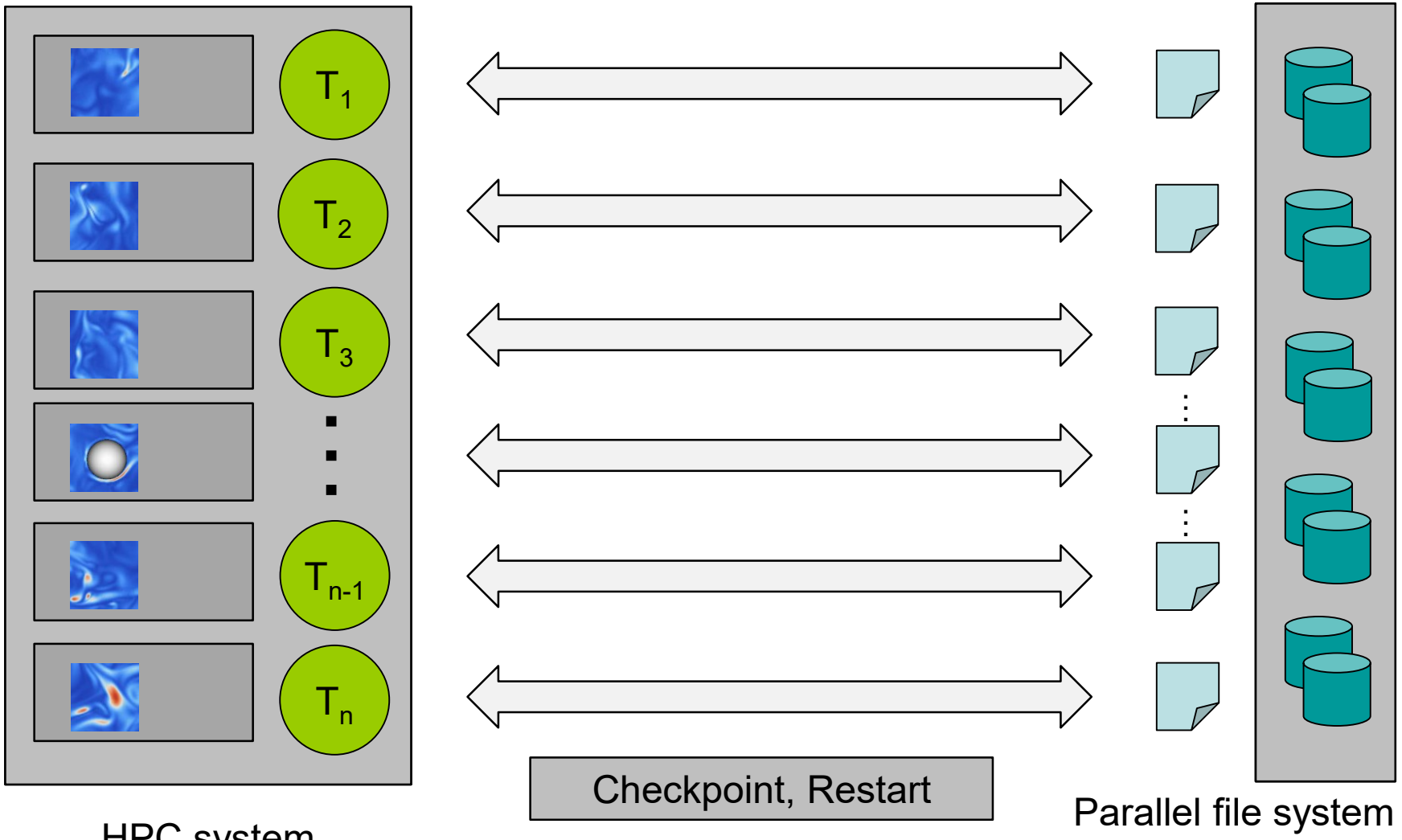


Task-local files

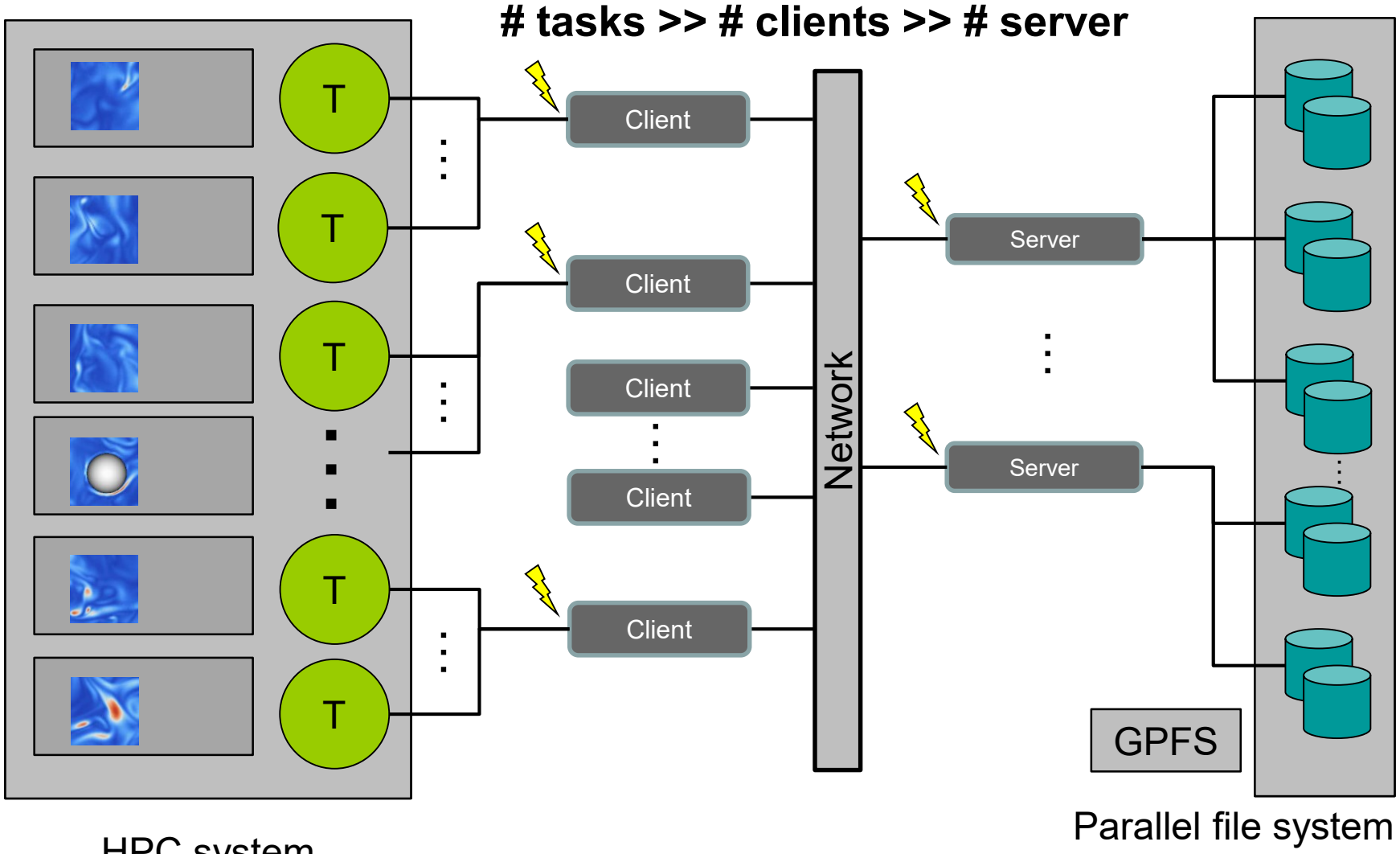


Shared files

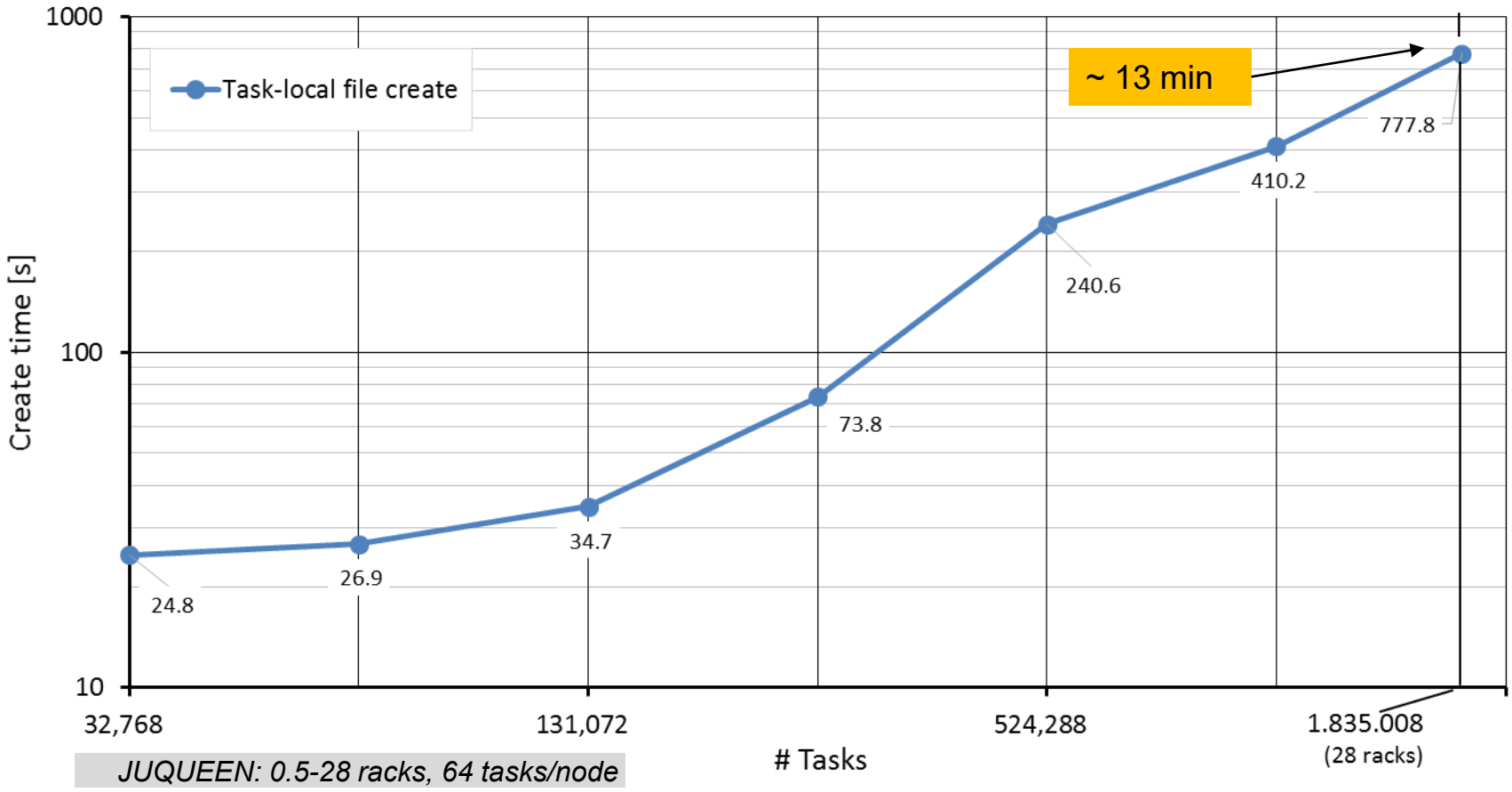
TASK-LOCAL I/O



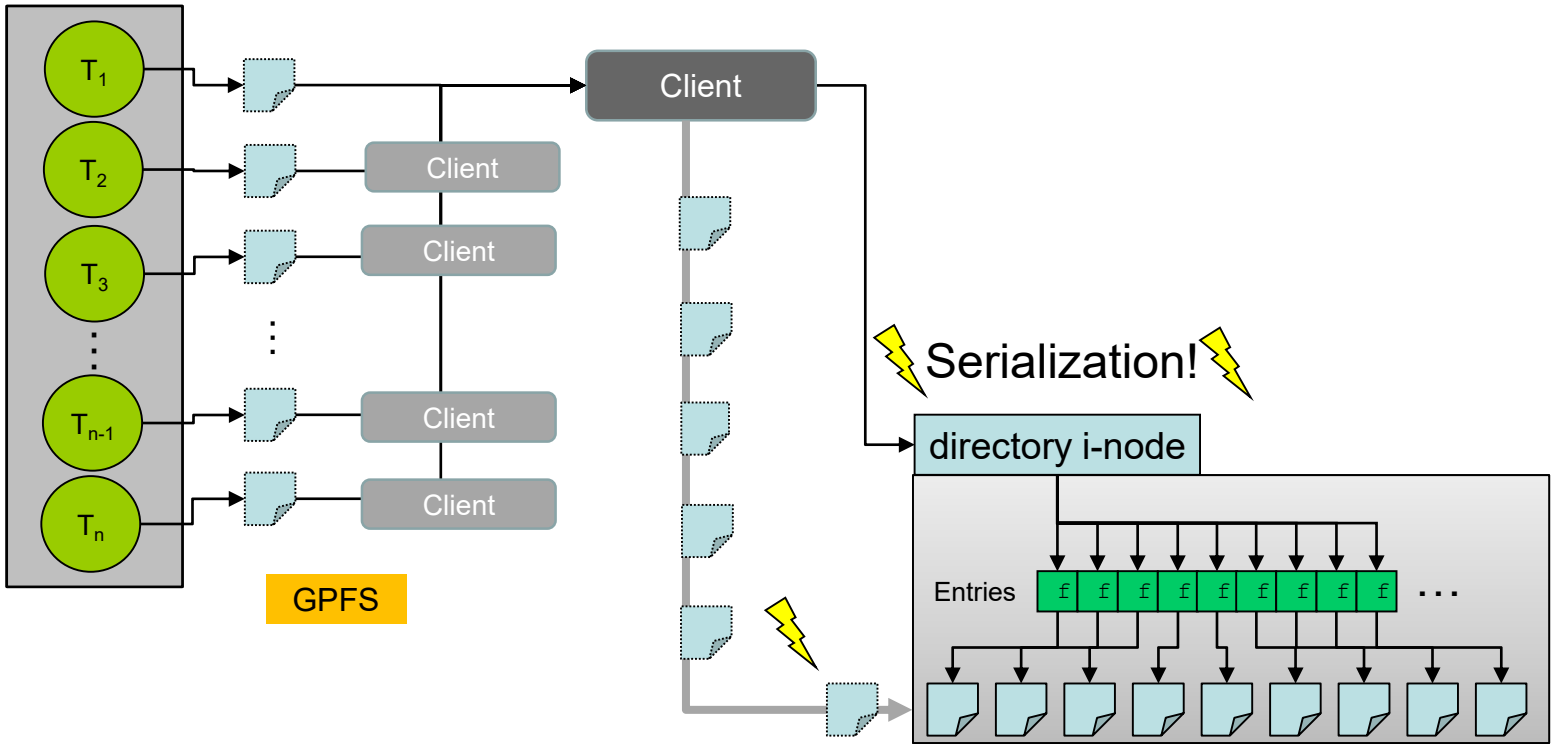
TASK-LOCAL I/O



I/O BOTTLENECK: PARALLEL FILE CREATION

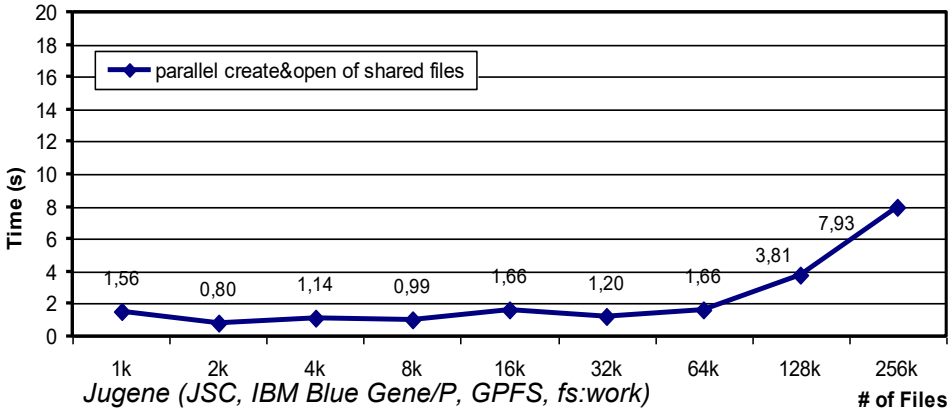


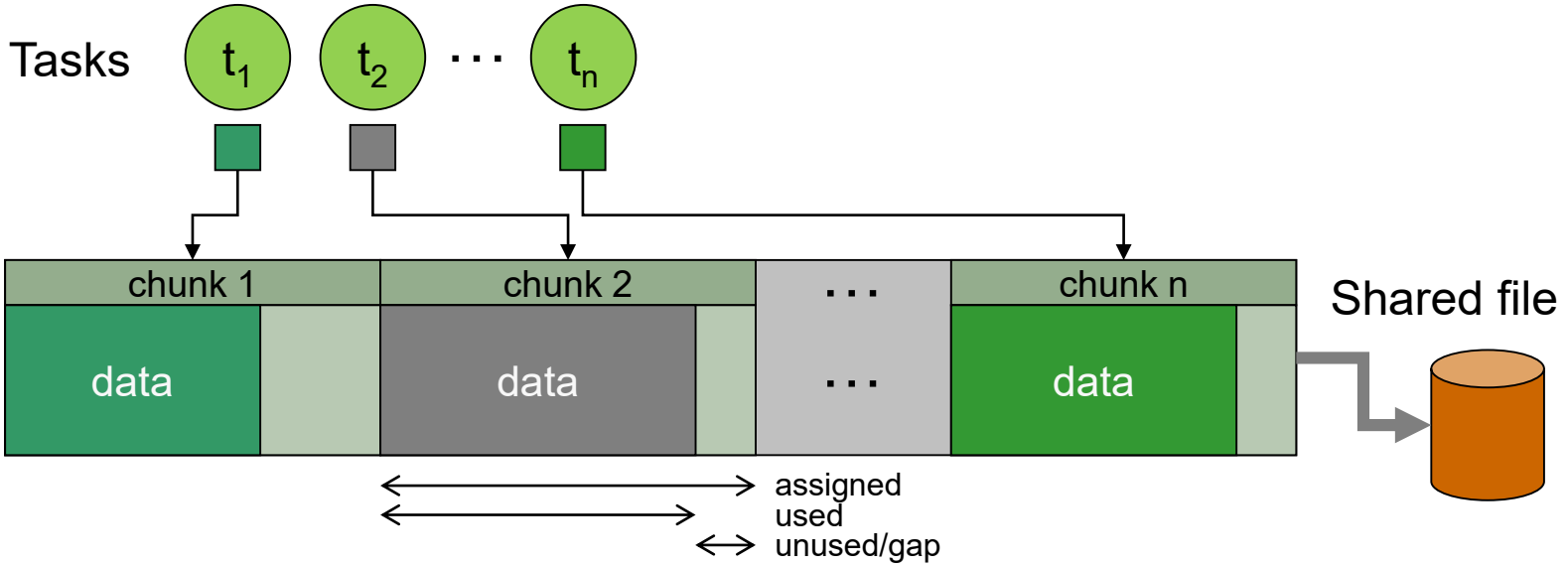
I/O BOTTLENECK: PARALLEL FILE CREATION



FILE FORMAT (1): A SINGLE SHARED FILE

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

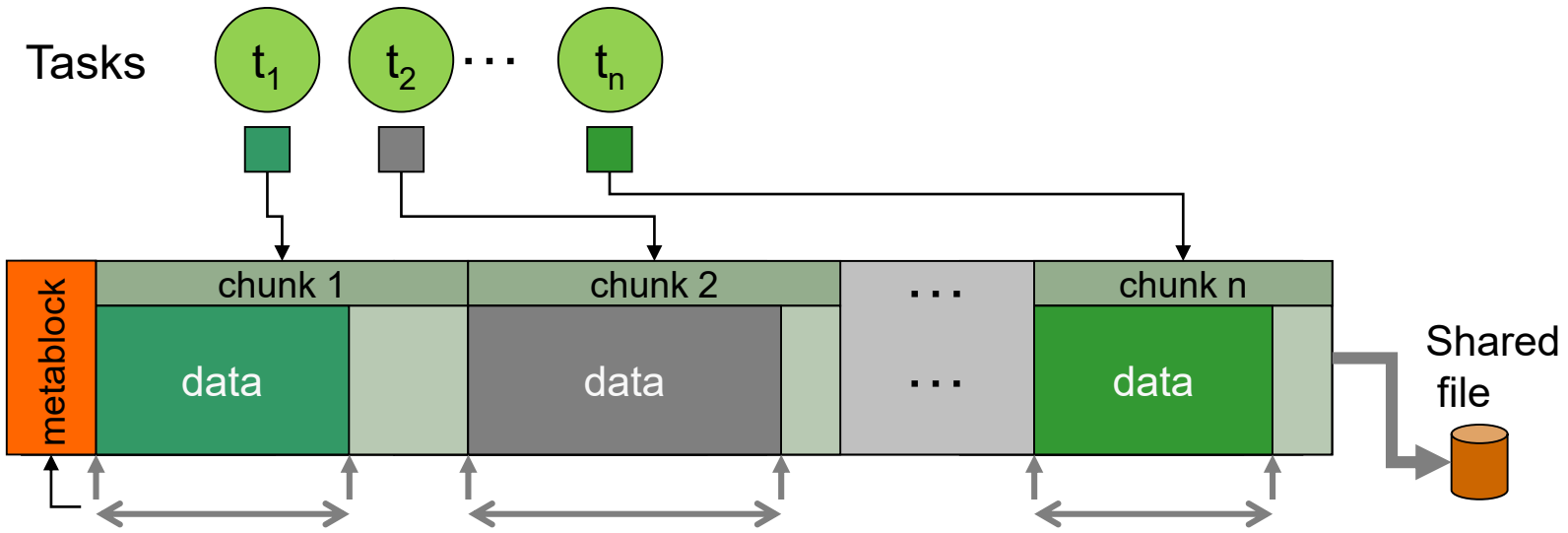




JÜLICH Forschungszentrum

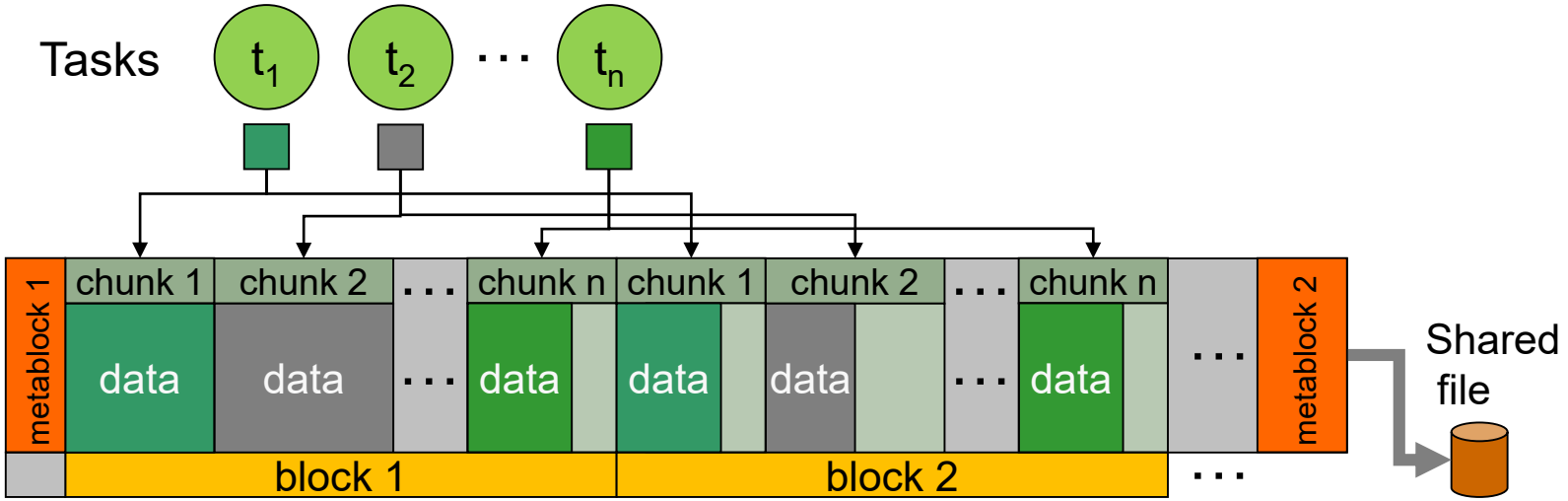
FILE FORMAT (2): METADATA

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



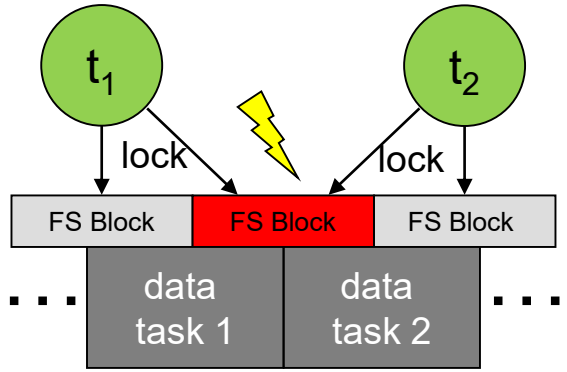
FILE FORMAT (3): MULTIPLE BLOCKS OF CHUNKS

- 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



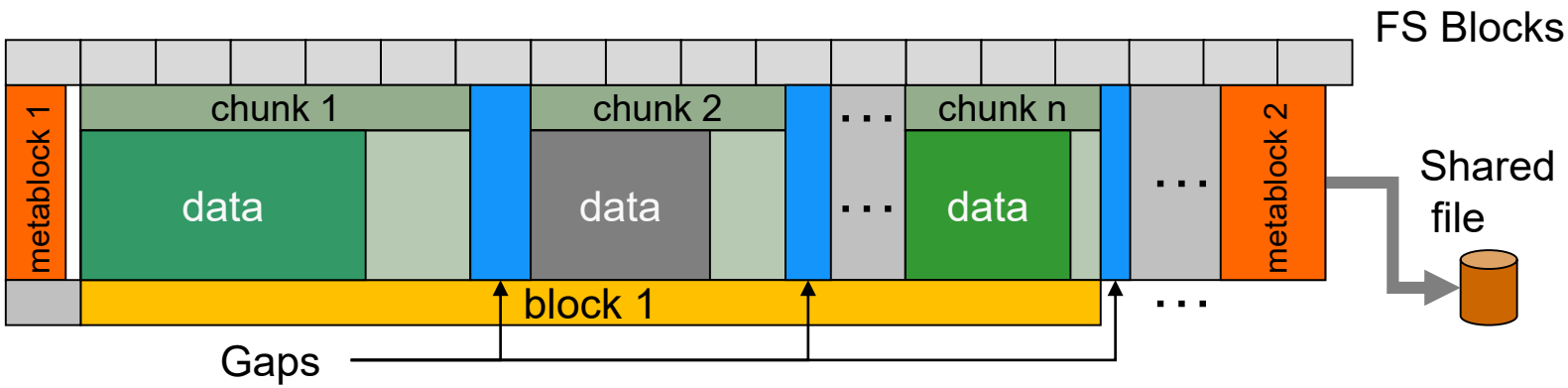
FILE FORMAT (4): ALIGNMENT TO BLOCK BOUNDARIES

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



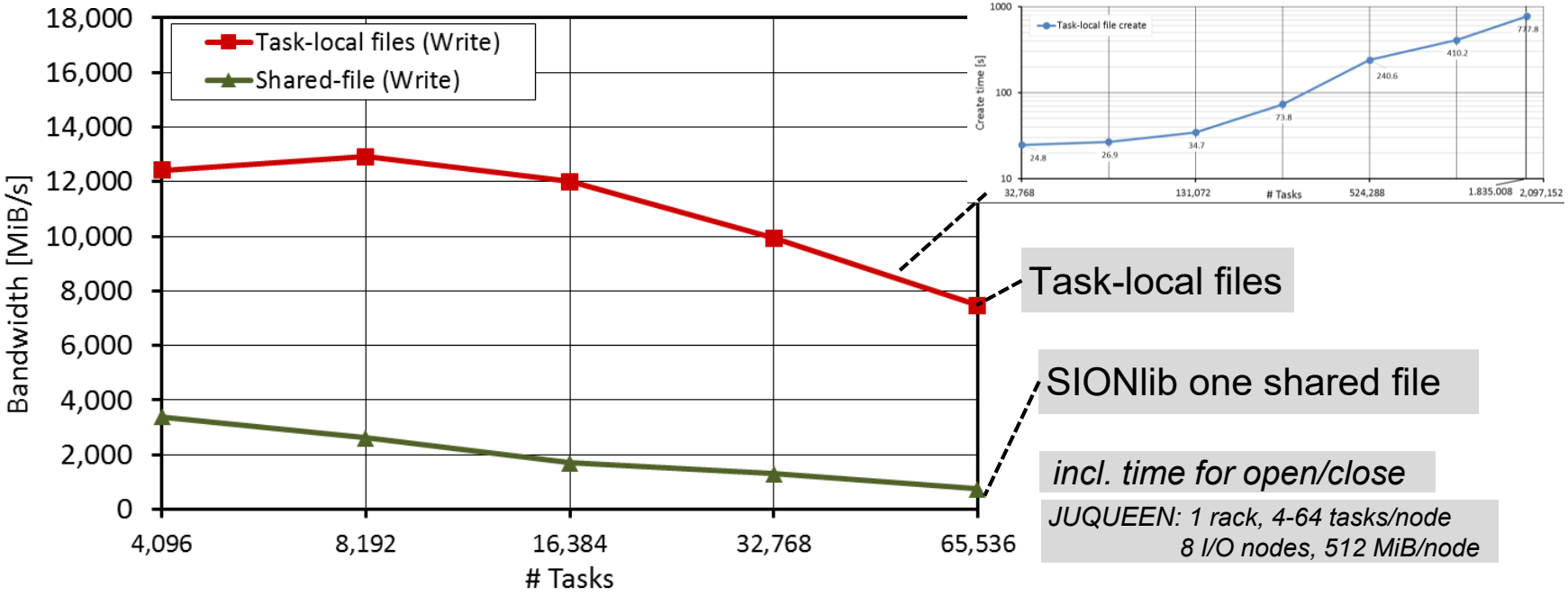
#tasks	data size	blksize	write bandwidth
32768	256 GB	aligned	**3650.2** MB/s
32768	256 GB	not aligned	**1863.8** MB/s

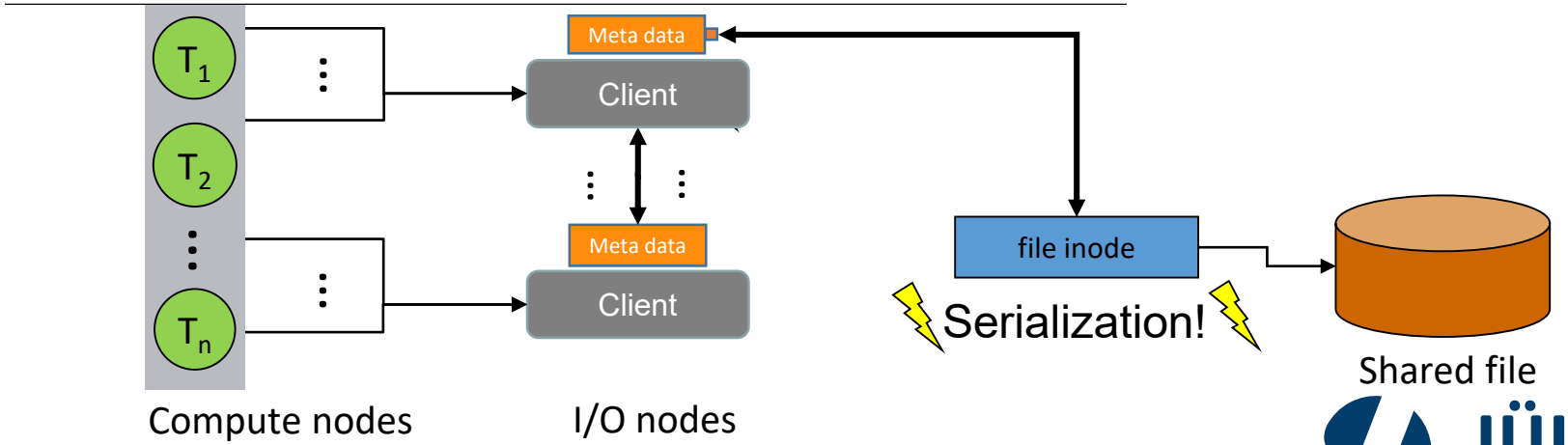
Jugene (JSC, IBM Blue Gene/P, GPFS, fs:work)



JÜLICH Forschungszentrum

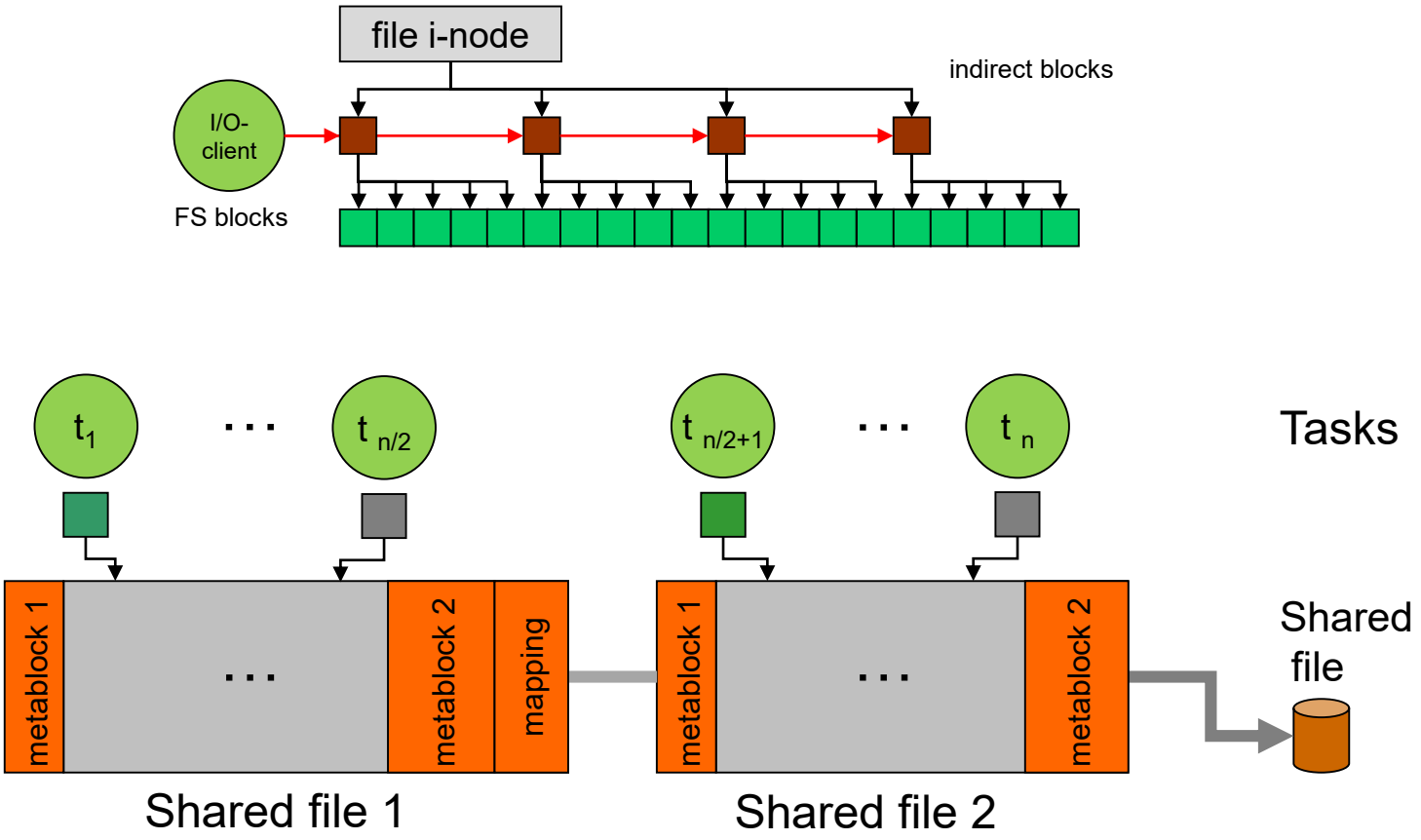
I/O BOTTLENECK: LARGE #TASKS & SHARED FILE



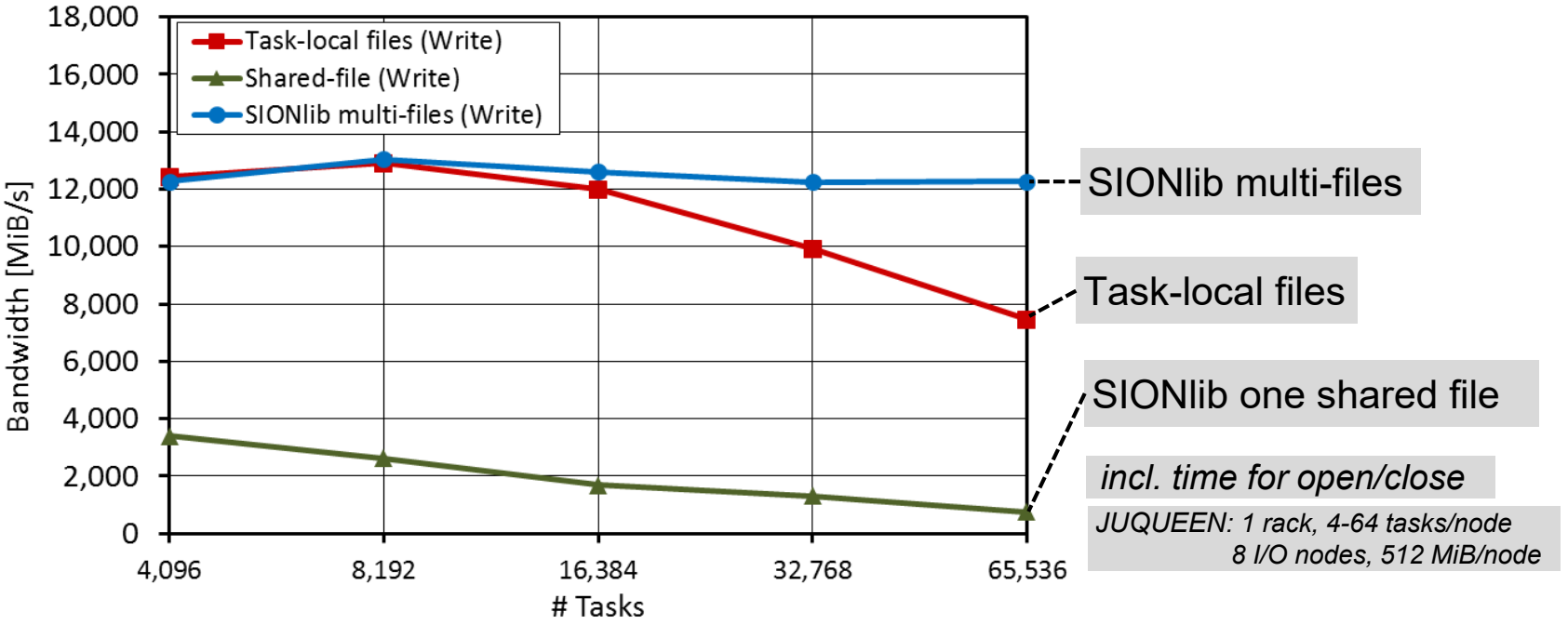


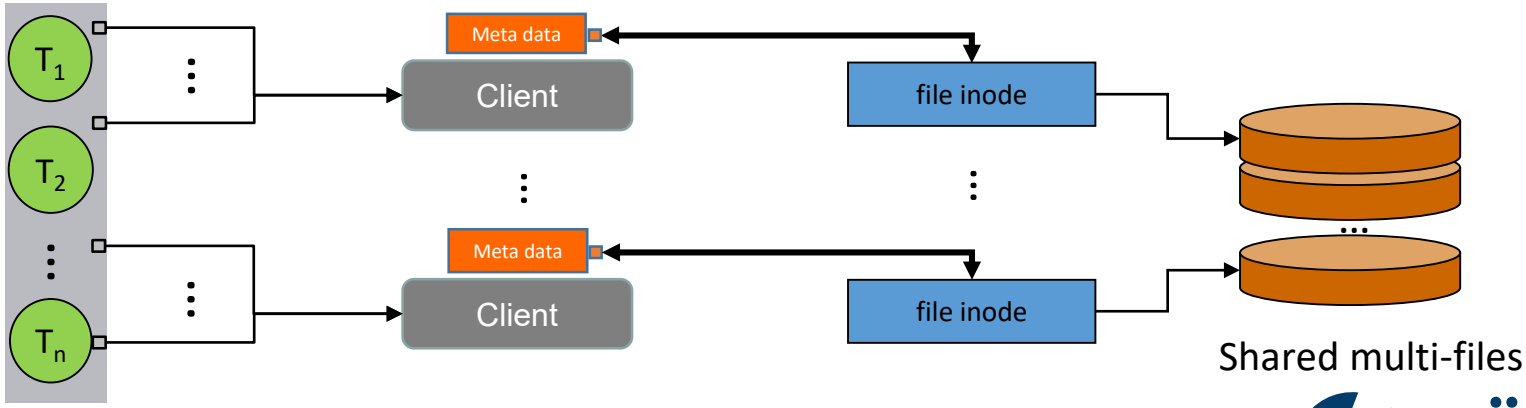
FILE FORMAT (5): MULTIPLE PHYSICAL FILES

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

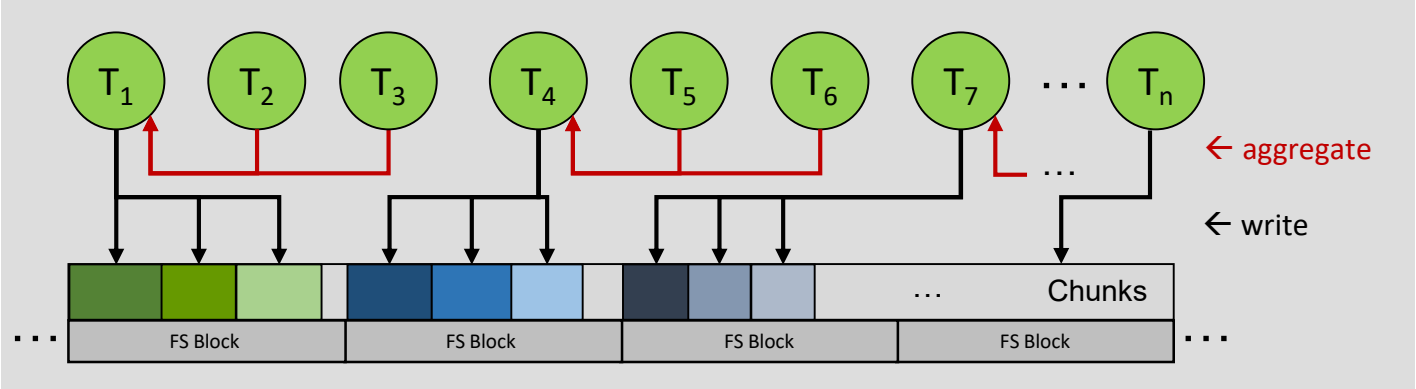


I/O BOTTLENECK: LARGE #TASKS & SHARED FILES



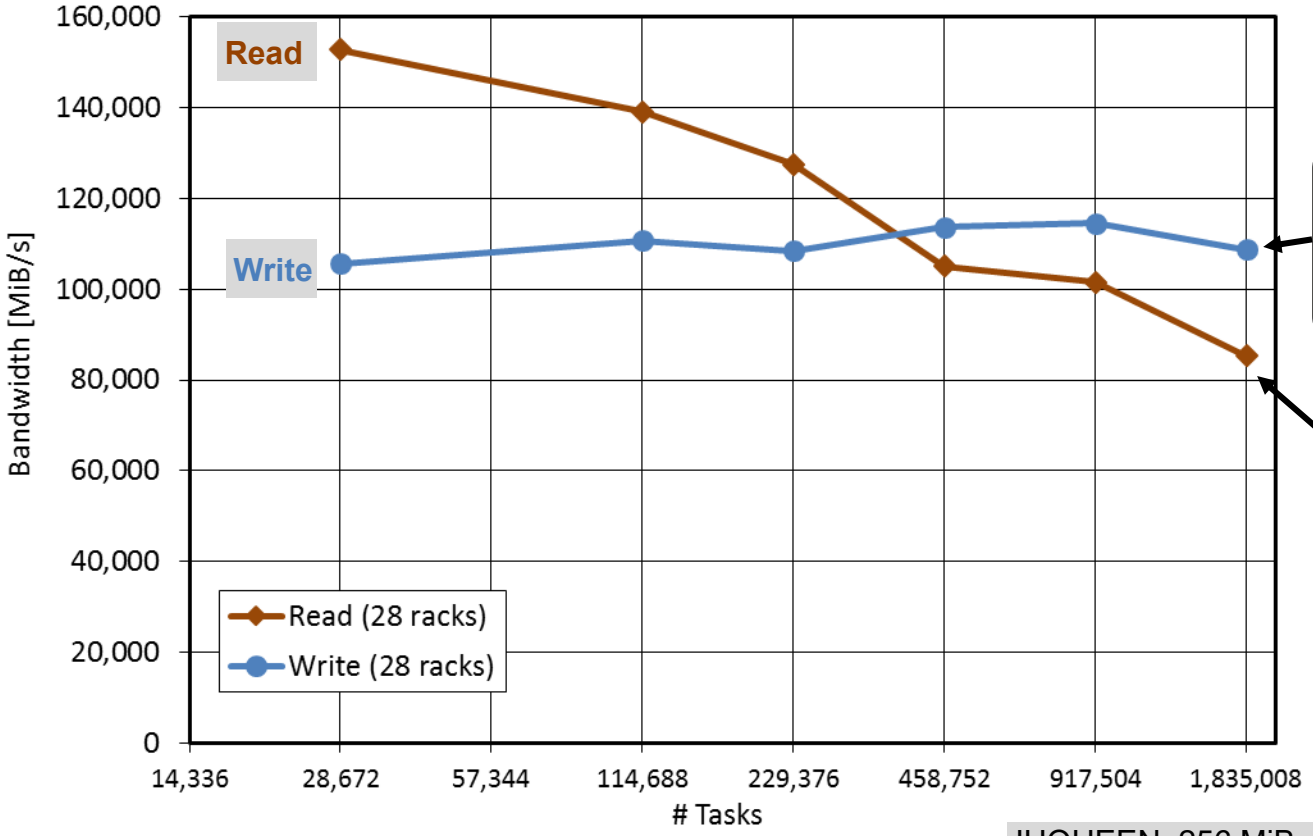


FILE FORMAT (6): COALESCING I/O



- 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)

JUQUEEN SIONLIB SCALING, BANDWIDTH



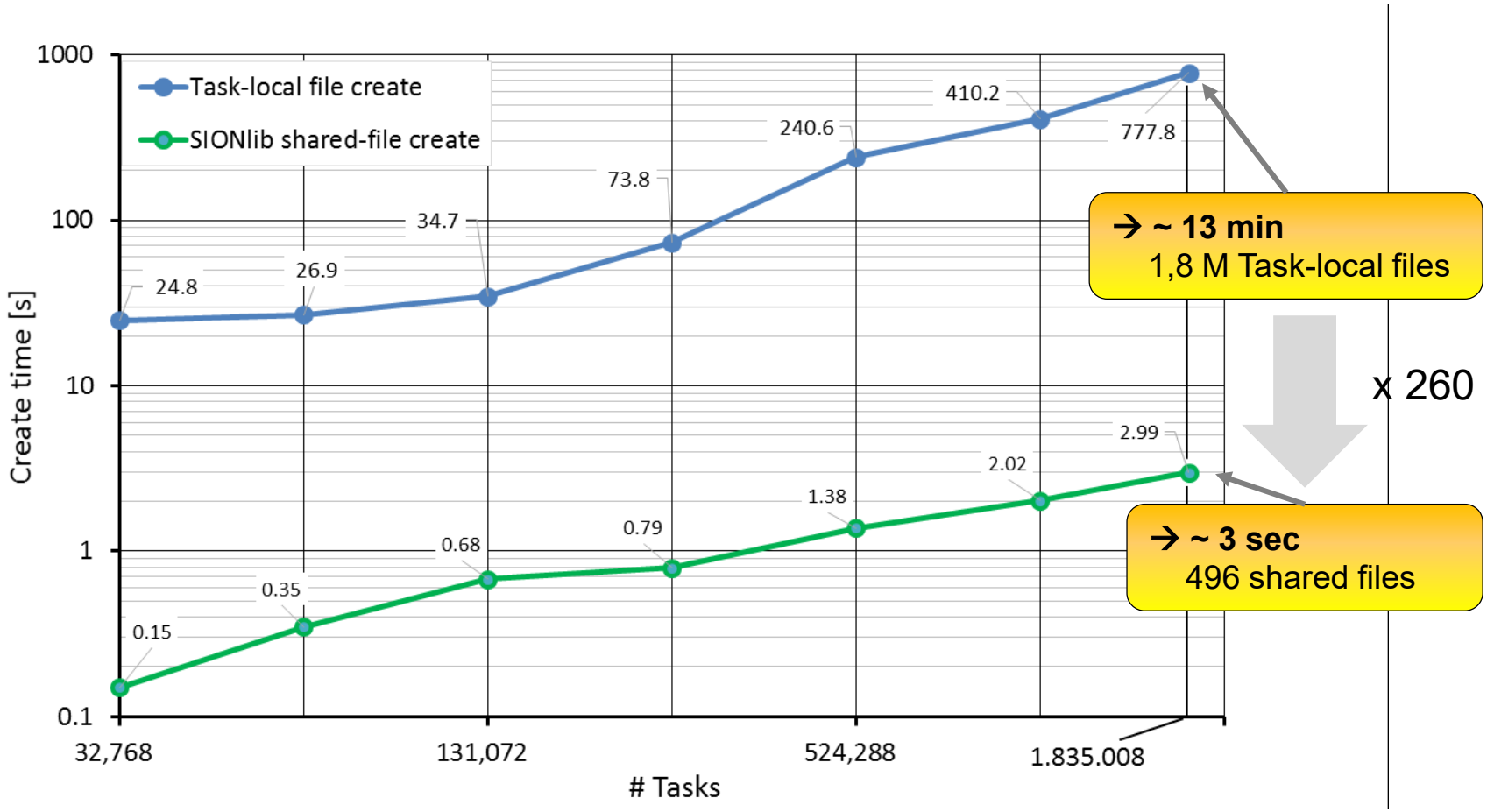
Write: 1,8 M tasks
→ 108 GiB/s
67% of peak I/O bandwidth

Read: 90-150 GiB/s

Read bandwidth degradation:
Characteristic of I/O shared read operation on BG/Q

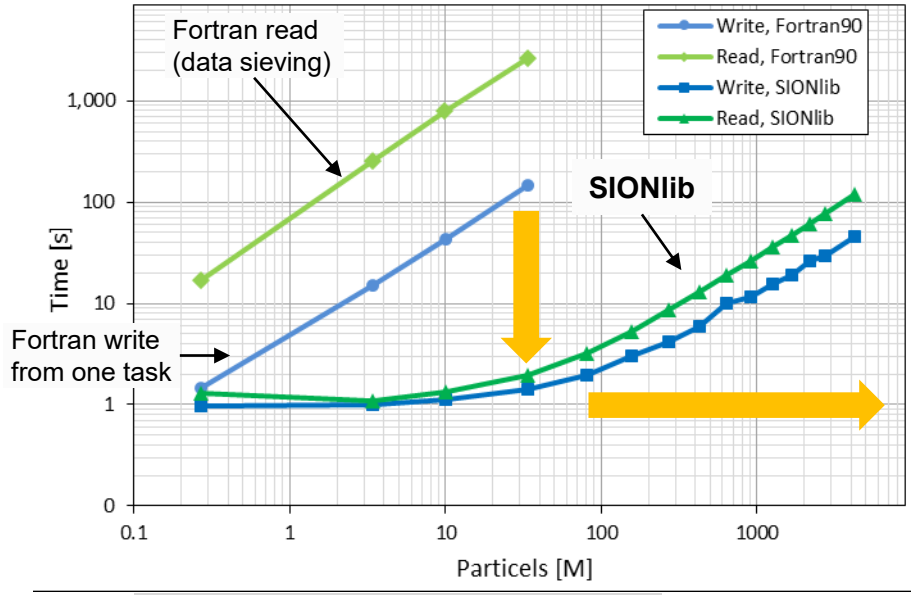
JUQUEEN: 256 MiB per node, 1-64 tasks/node, up to 7 TiB data, one file per I/O-bridge

JUQUEEN SIONLIB SCALING, FILE CREATE

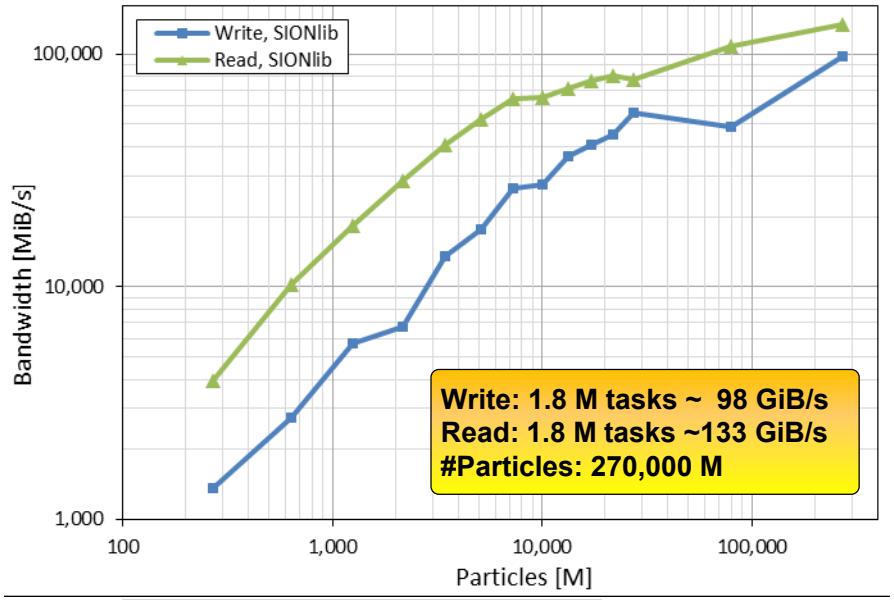


APPLICATION MP2C

MP2C: Massively Parallel Multi-Particle Collision Dynamics
Mesoscopic hydrodynamics + Molecular dynamics



JUQUEEN: 512 nodes, 16 tasks per node



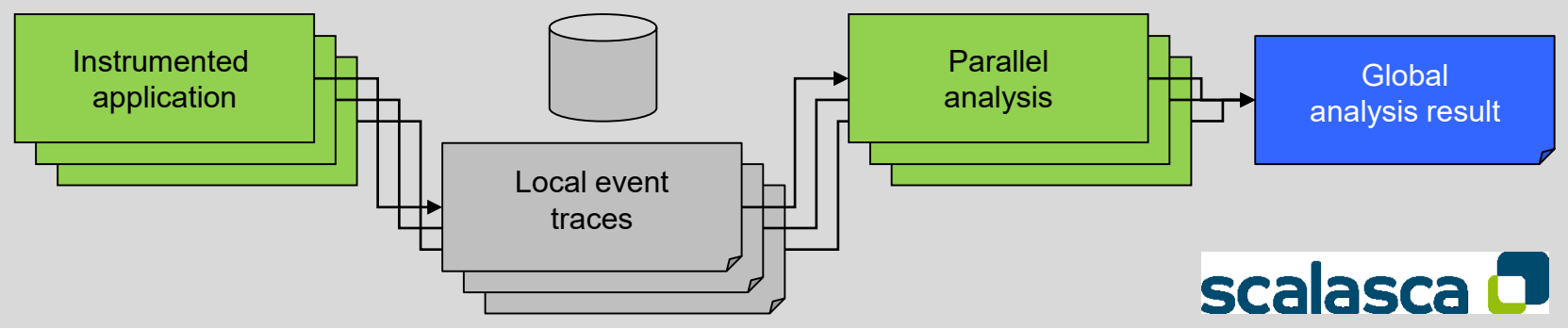
JUQUEEN: 28 racks, 64 tasks per node

WWW: http://www.fz-juelich.de/ias/jsc/EN/Expertise/High-Q-Club/MP2C

PARALLEL PERFORMANCE TOOLS

Scalasca Automatic parallel performance analysis

Score-P Scalable performance measurement

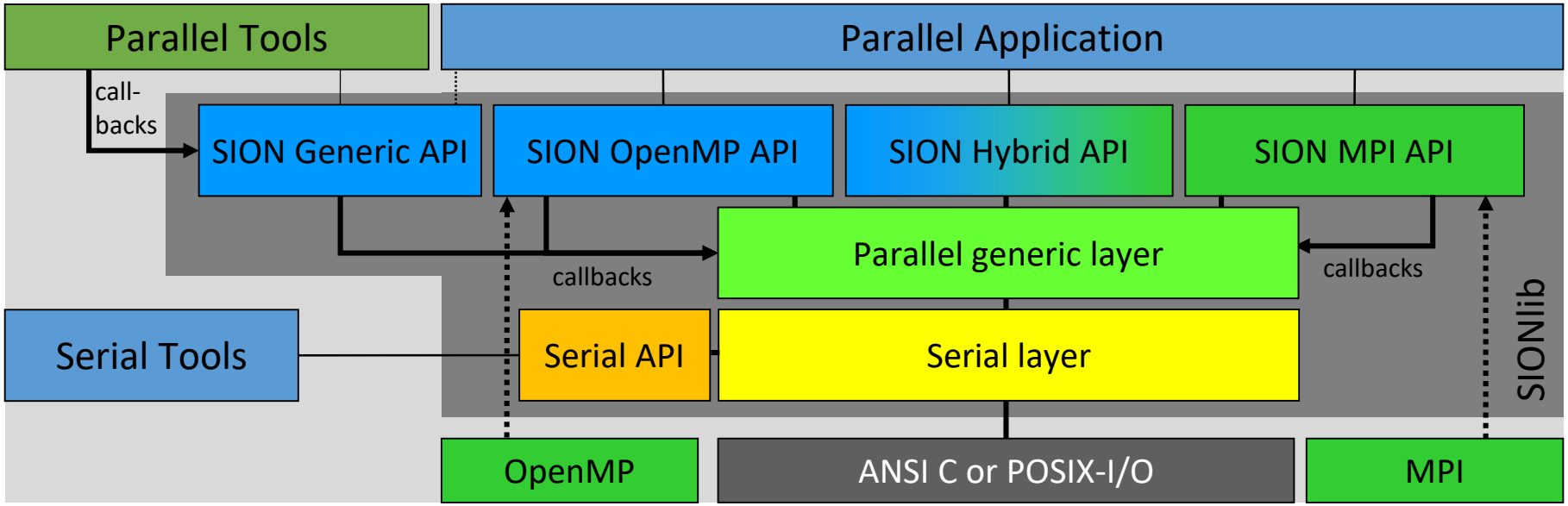


Results:

- Reduced number of event trace files
- Faster output of event trace files

Scalasca/ctest	# Tasks	I/O time / Write
Task-local I/O	131,072	**151.0 s**
SIONlib	131,072	**89.6 s**
JUQUEEN: 2048 nodes		x **1.69** faster

SOFTWARE LAYERS AND APIS



- Implementation in C,
- Language bindings for Fortran, C++, Python
- Designed as extension of ANSI C file I/O API
- Lightweight and easy-to-use library

VERSION, DOWNLOAD, INSTALLATION

- Version: 1.7.6 (November 2019)
- Version: file format: 5
- Open-source license

  https://www.fz-juelich.de/jsc/sionlib

- Installation (or `module spider SIONlib`)

Shell
```
./configure –prefix=<INSTPATH> [--enable-debug] …
make
make test
make install
```

- Query version of SIONlib:

Shell
```
sionversion
SIONlib Version 1.7.2 (svn_rev 2256), fileformat
version 5 (sionversion)
```

HEADER FILE & RETURN CODES

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

Fortran
```fortran
use sion_f90
use sion_f90_mpi
```

- **Important**: include MPI first

- C, C++: C-preprocessor directives, defined in `sion.h`

C
```c
#define SION_SUCCESS 1
#define SION_NOT_SUCCESS 0
```

- Fortran: `1` → success, `0` → not success

TASK-LOCAL I/O

C

```c
/* Open */
sprintf(tmpfn, "%s.%06d", filename, my_nr);
fileptr = fopen(tmpfn, "bw", ...);
...

/* Write */
fwrite(bindata, 1, nbytes, fileptr);
...

/* Close */
fclose(fileptr);
```

- No collective operation, no shared files
- Data: stream of bytes

SIONLIB: PARALLEL OPEN/CLOSE

- Parallel interface, using MPI

```c
sion_paropen_mpi(), sion_parclose_mpi()
```

- Parallel interface, using OpenMP

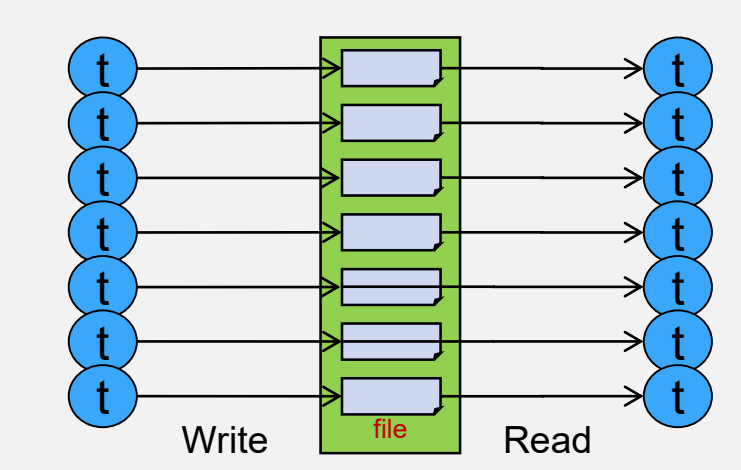
```c
sion_paropen_omp(), sion_parclose_omp()
```

- Parallel interface, using MPI+OpenMPI

```c
sion_paropen_ompi(), sion_parclose_ompi()
```

- SIONlib data format does not distinguish data chunks of MPI tasks and OpenMP threads

  → *SIONlib task*



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

fmode: file mode

- Must specify at least one of the following

`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

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

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

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/CLOSE

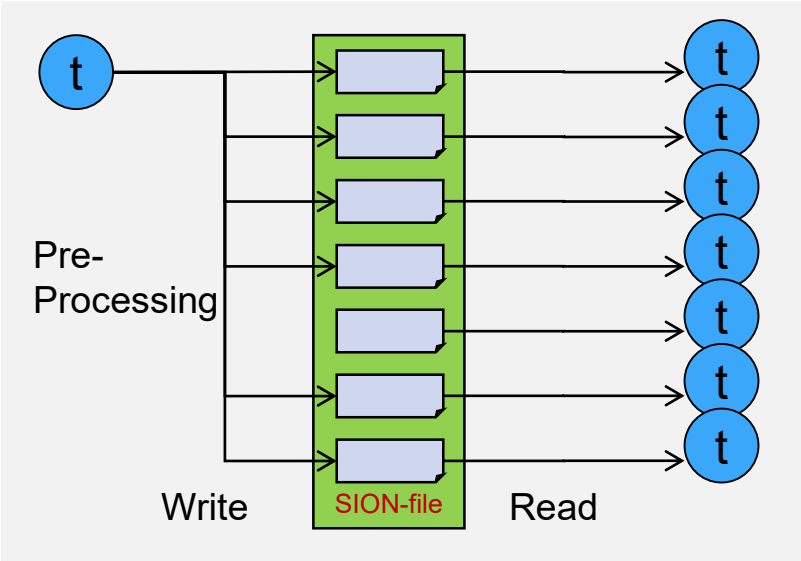
- Serial interface, accessing data of all SIONlib tasks in container

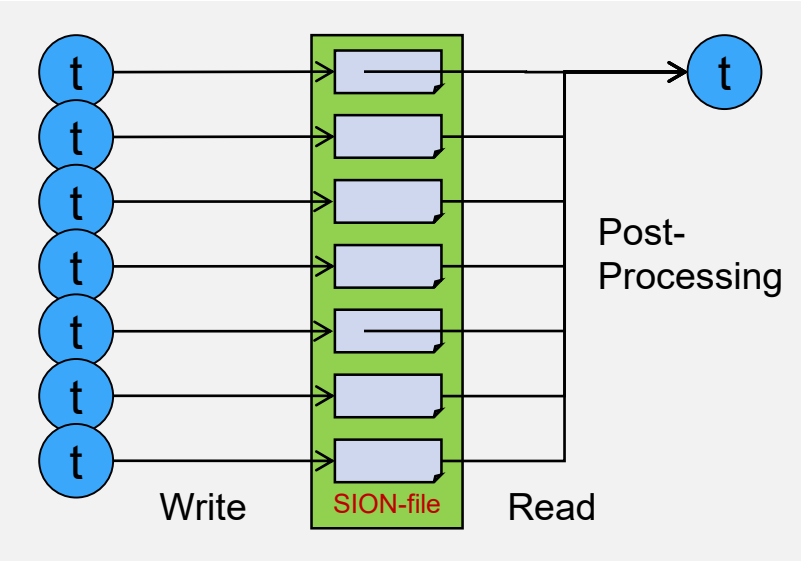
```
C  sion_open(), sion_close()
```

- Serial interface, accessing data of one SIONlib task

```
C  sion_open_rank(), sion_close()
```

Use case: pre- and post-processing tools, converters





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)
```

- Open a SION file in serial mode
- All chunks of all tasks can be selected, via `sion_seek`
- Multiple physical files can be handled
- Designed for serial pre- and post-processing tools
- 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
- Currently no fault tolerant handling of the metadata, call has to executed successfully

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
- Currently no fault tolerant handling of the metadata, call has to be executed successfully

EXERCISE

Exercise 1 – SIONlib hello world

- Write a parallel program `array_1d_write` in C or Fortran which creates and closes an empty SIONlib file
- Use a chunksize to store 1000 Integer

Check the resulting file using:

```
siondump
```

Copy the exercise files

```
$ $PROJECT_training2000/sionlib/copy.sh
```

Load the necessary modules

```
$ module load intel-para SIONlib
```

Use the included Makefile

```
$ make
```

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` chunk, beginning from current position
  - This size must not exceed the chunksize defined in open call
- Internally this function uses **`sion_ensure_free`** space to check whether there is enough space is available
- Returns number of elements written

ENSURE FREE SPACE IN CHUNK

C

```
int sion_ensure_free_space (int         sid,
                            sion_int64 bytes);
```

Fortran

```
FSION_ENSURE_FREE_SPACE (SID, BYTES, IERR)
INTEGER   SID, IERR
INTEGER*8 BYTES
```

- Ensures that there is enough space available for writing
- A new chunk will be allocated if not enough space is left in the current chunk
- The function is a task-local function, which can be called independently from other MPI tasks
- The function moves the file pointer to a new position in some cases and flushes also the local file pointer
- Returns 1 if space could be ensured, there is currently no indicator if a new chunk was allocated

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

Check the resulting file using:

```
siondump
```

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
- Internally this function reads in a while loop until all data is read from file
- Reading more data than stored in one chunk is possible with this wrapper
- Returns number of elements read
- `sion_bytes_avail_in_block` can be used to get all available data

END OF FILE

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

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

- Equivalent to POSIX feof which cannot be used for shared SION files
- 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
- 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
- In parallel write mode seeking is currently not supported

EXERCISE

Exercise 3 – SIONlib read

- Write a parallel program `array_1d_read` in C or Fortran
- Read all elements of your created files and print them to the screen

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

TOOLS: SIONSPLIT

Command

```
sionsplit [options] sionfile prefix

Parameters & Options:
  sionfile    Input file in SIONlib format
  prefix      directory and/or filename-prefix for
              task-local files
  [-v]        verbose mode
  [-g]        use global rank for numbering files
  [-d <num>] number of digits for filename
              generation (default 5)
```

- Generates task-local files from SION-file

TOOLS: SIONCAT

Command

```
sioncat [options] sionfile

Parameters & Options:
  sionfile          Input file in SIONlib format
  [-v]              verbose mode
  [-t <tasknum>]    write only data of task <tasknum>
  [-b <chnknum>]    write only data of chunk <chnknum>
  [-o <outfile>]    file data will be written to, if
                    not specified stdout will used
```

- Extract all or selected data from a SION file

TOOLS: SIONDUMP

Command

```
siondump [options] sionfile

Parameters & Options:
  sionfile   Input file in SIONlib format
  [-a]       print all information about all blocks
  [-m]       print all mapping information
  [-l]       print all sizes in number of bytes
  [-k]       print key-value statistic for each task
  [-K]       print key-value list for each task
  [-S]       print size of internal data structure
             in memory
  [-V]       show version number of SIONlib
  [-v]       verbose mode
```

- Print metadata information of a SION file

TOOLS: SIONDEFRAG

Command

```
siondefrag [options] sionfile new_sionfile

Parameters & Options:
 sionfile            Input file in SIONlib format
 new_sionfile        Output file in SIONlib format
 [-Q <fsblksize>] filessystem blocksize for new
                     sion file in MB (default: input file)
 [-q <fsblksize>] filessystem blocksize for new
                     sion file in bytes
 [-V]                show version number of SIONlib
 [-v]                verbose mode
```

- Generates new SION file from existing one
- Changes the underlying file system block size
- Can be used to remove empty gaps, caused by
  - Not completely filled chunks
  - Alignment to file system blocks

TOOLS: SIONCONFIG

Command

```
sionconfig [options]

Options:
 (--com|--ser|--omp|
  --mpi|--ompi|--gen)       select SIONlib API
 (--cflags|--libs|--path)   select output of sionconfig
 [--32|--64]                Precision
 [--be] [--fe] [--mic]      Cross compile
 [--gcc]                    for GCC Compiler
 [--c|--f77|--f90|--cxx]    Language selection
 [-V]                       show version number
 [-v]                       verbose mode
```

- Returns flags and options for compiler and cinker on stdout

IMPACT

SIONlib application integration

DUNE-ISTL (Multigrid solver)	**NEST** (Human Brain Simulation)
ITM (Fusion-community)	**OSIRIS** (Fully-explicit particle-in-cell code)
LBM (Fluid flow/mass transport)	**PEPC** (Pretty Efficient Parallel C. Solver)
KKRnano (Korringa-Kohn-Rostoker Green Fct.)	**Profasi** (Protein folding and aggr. simulator)
MAXW-DGTD (Human exposure to em. fields)	**PSC** (Particle-in-cell code)
MP2C (Mesoscopic hydrodynamics + MD)	**TurboRVB** (High temp. superconductivity)
muPhi (Flow and Transport in Porous Media)	**MPAS** (Model for prediction across scales)

SIONlib in parallel performance tools

Scalasca	Automatic parallel performance analysis
Score-P	Scalable performance measurement
	Infrastructure for parallel codes (BMBF-project Score-E)

SIONlib in EU funded projects

DEEP-ER	Adaption to new platform, resiliency (buddy checkpointing)
DEEP-EST	Adaption to new platform
EoCoE	Integration into applications

AVAILABILITY

SIONlib

- *https://www.fz-juelich.de/jsc/sionlib*
- Open source license
- *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