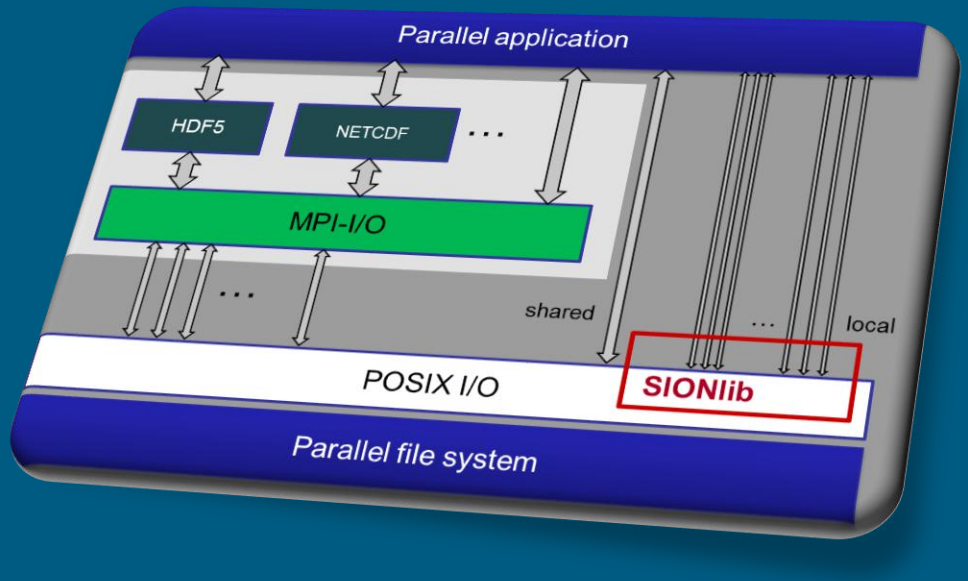
JÜLICH
FORSCHUNGSZENTRUM

Optimisation and Benchmarking

Part 4 – Parallel I/O

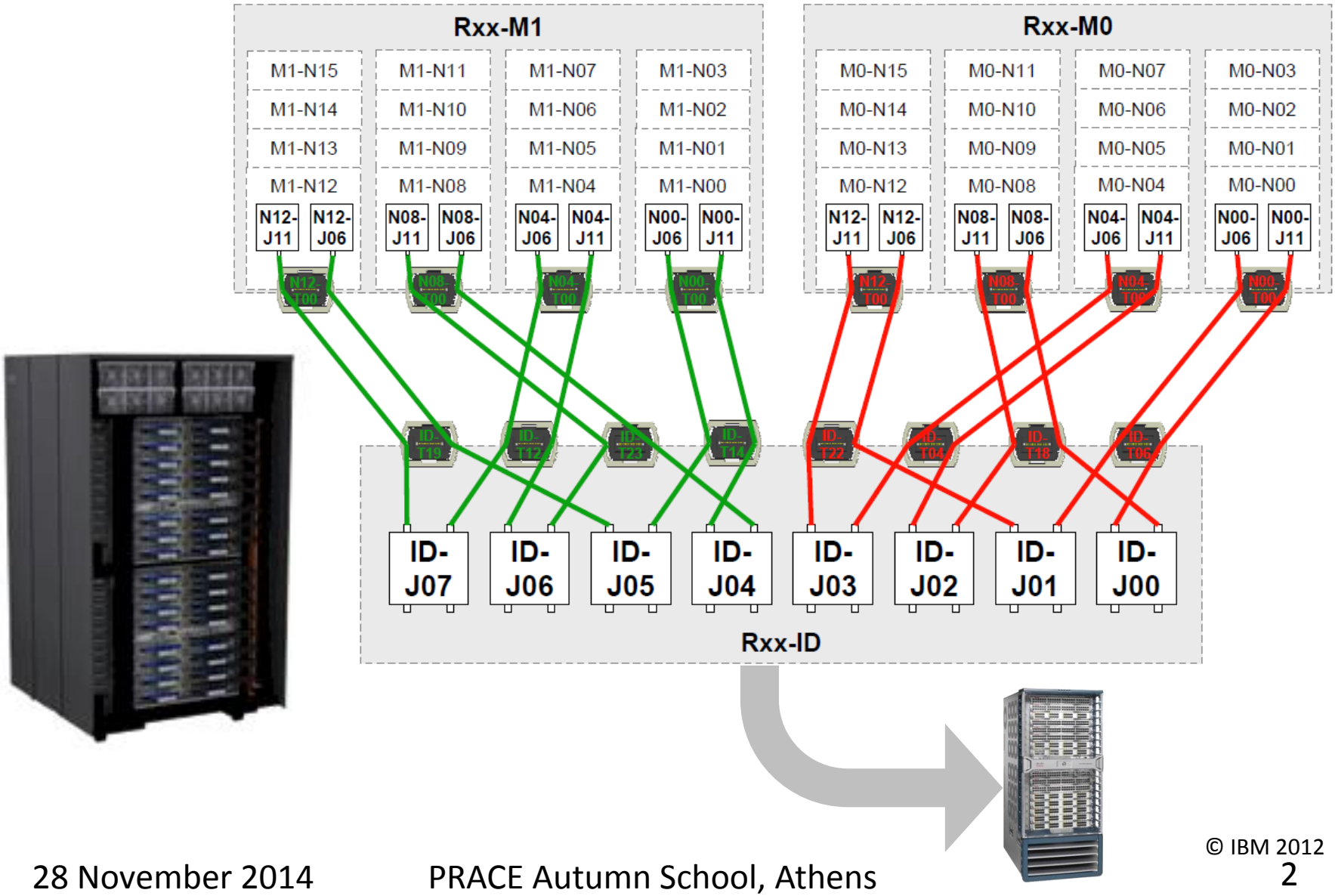


28. Nov 2014|

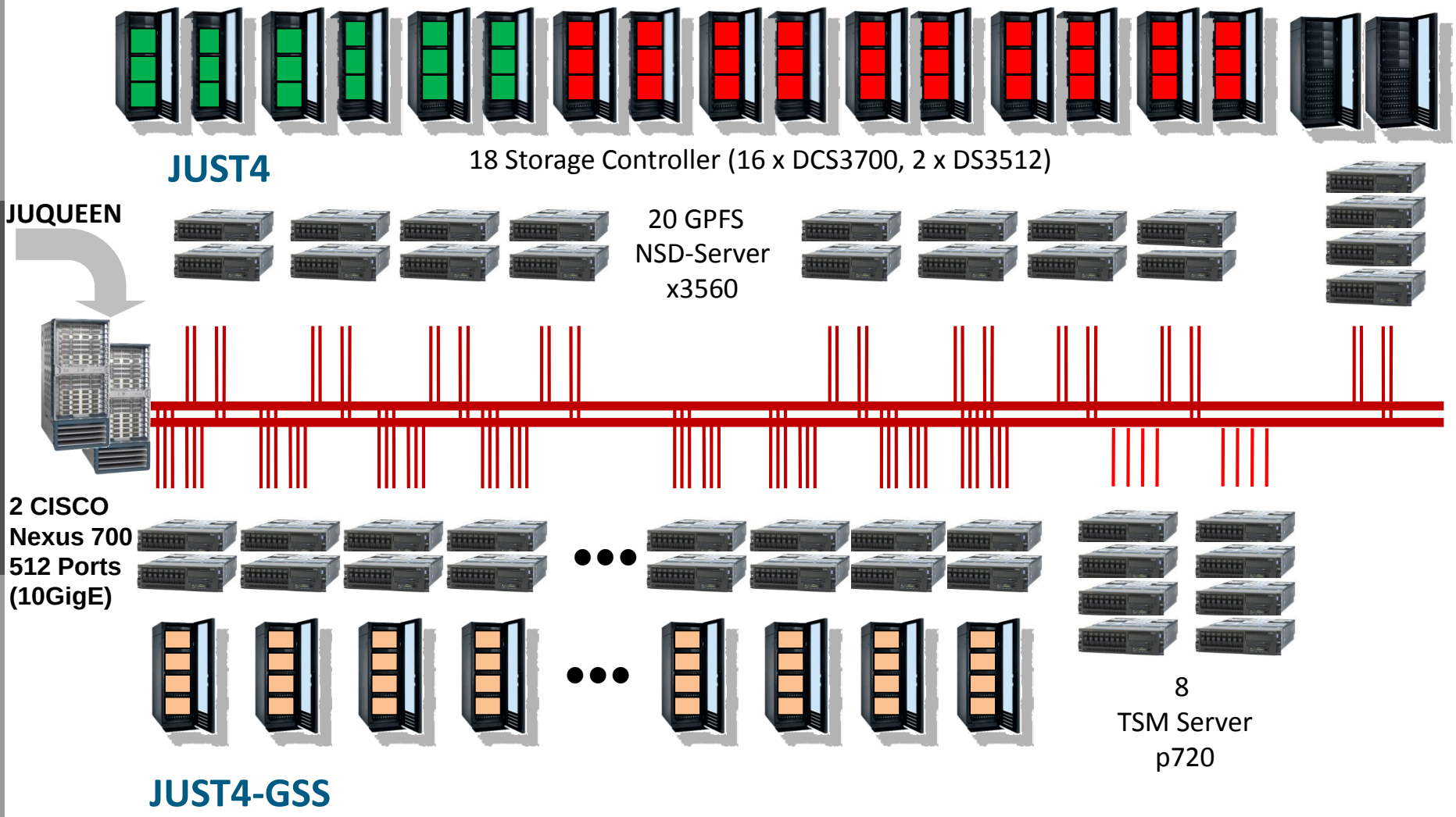
Alan O'Cais
a.ocais@fz-juelich.de

Live notes:
http://supercomputing.cyi.ac.cy/index.php/live

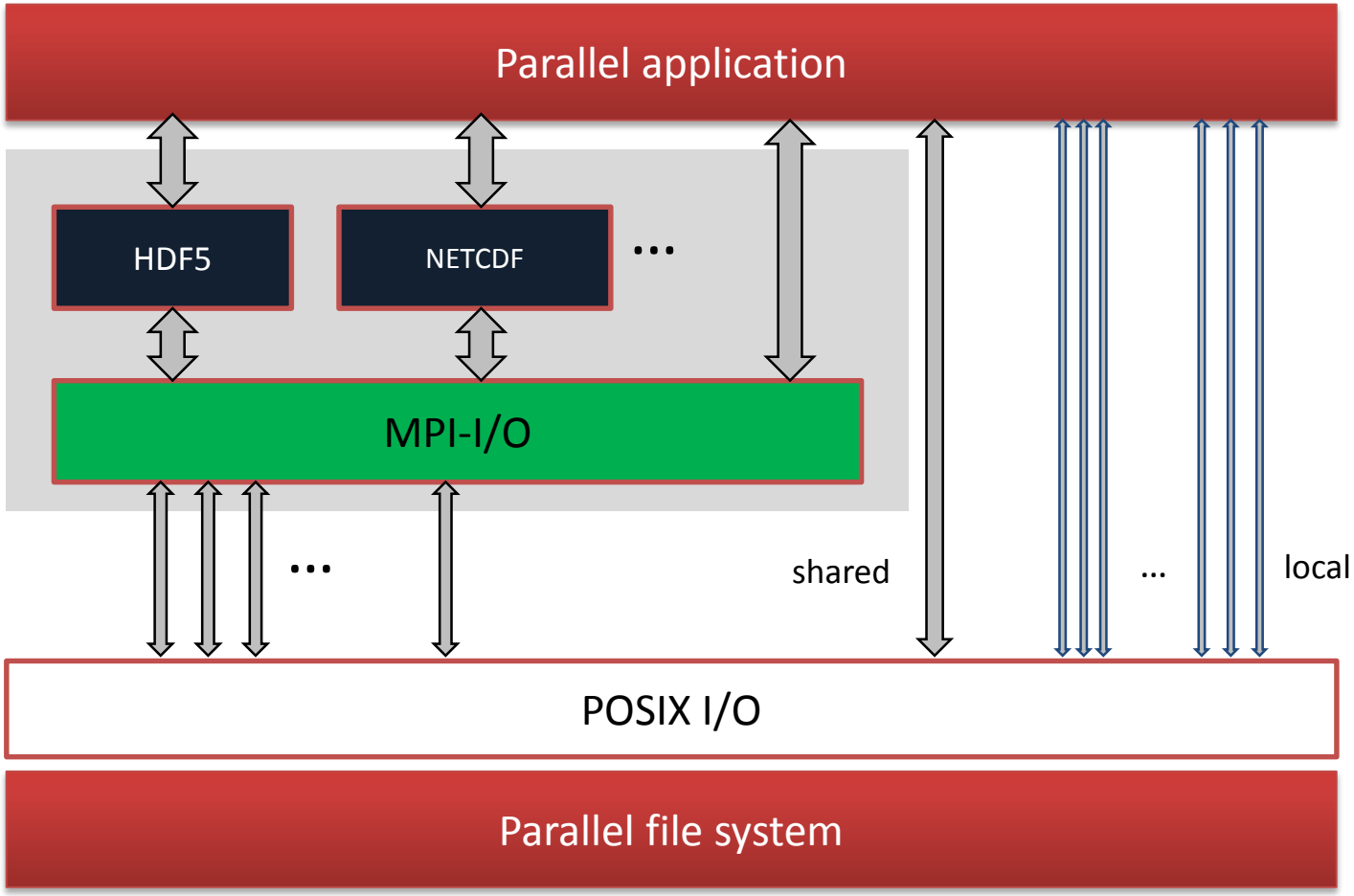
Blue Gene/Q: I/O-node cabling (8 ION/Rack)



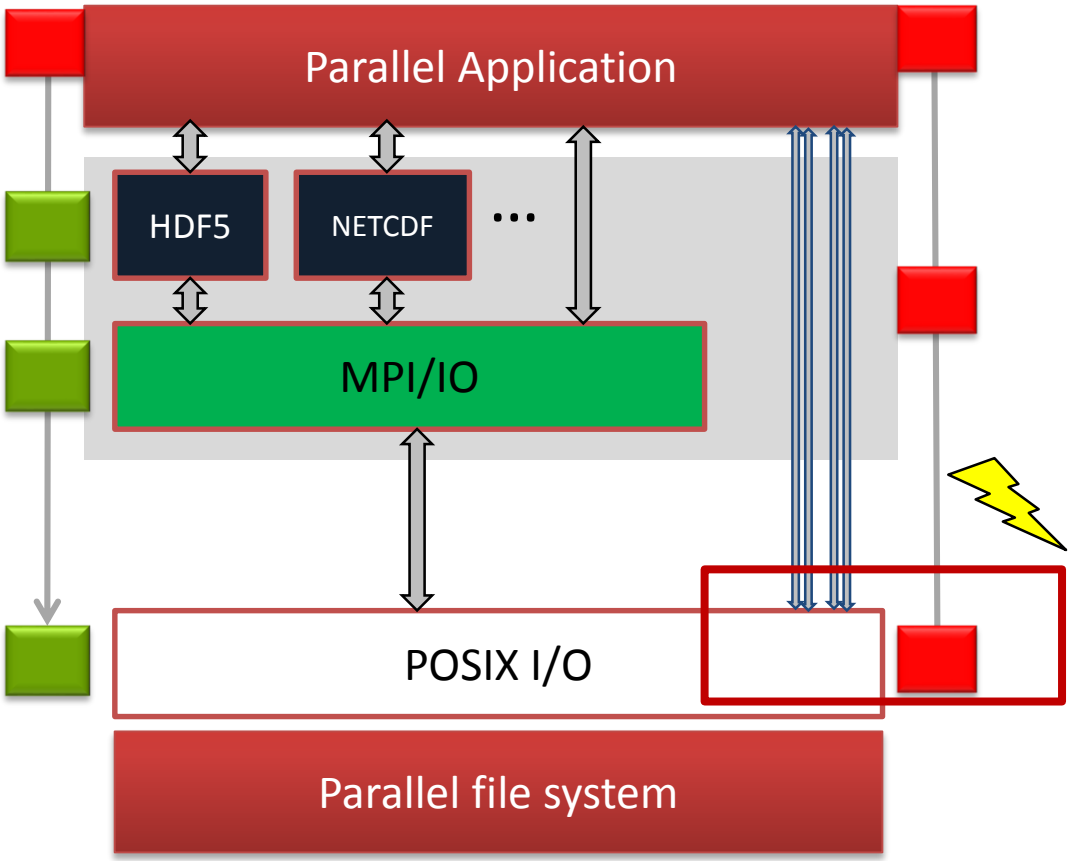
JUQUEEN and JUST I/O-Network



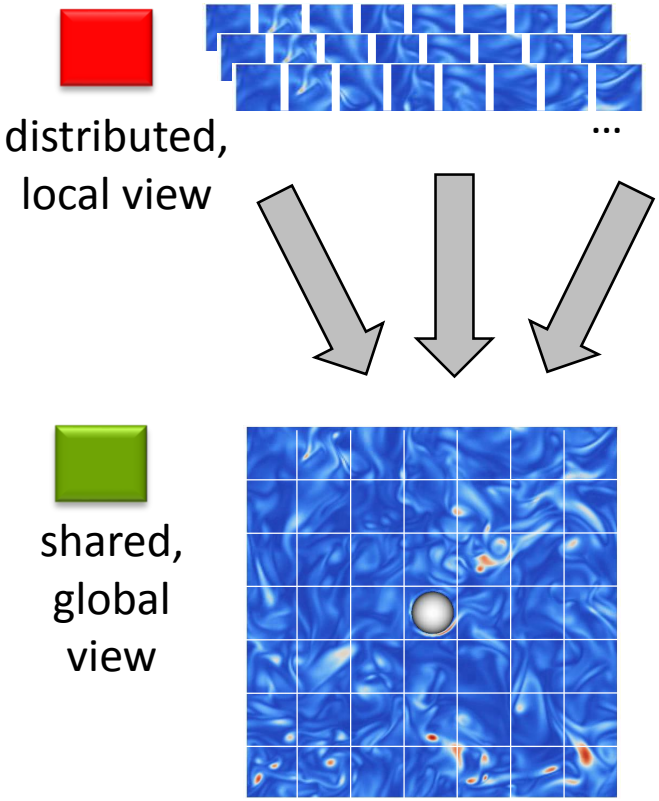
Application View to Parallel I/O



Application View: Data Distribution



Software-view



Data-view

Parallel Task-local I/O at Large Scale

Usage Fields:

- Check-point files, restart files
- Result files, post-processing
- Parallel Performance-Tools

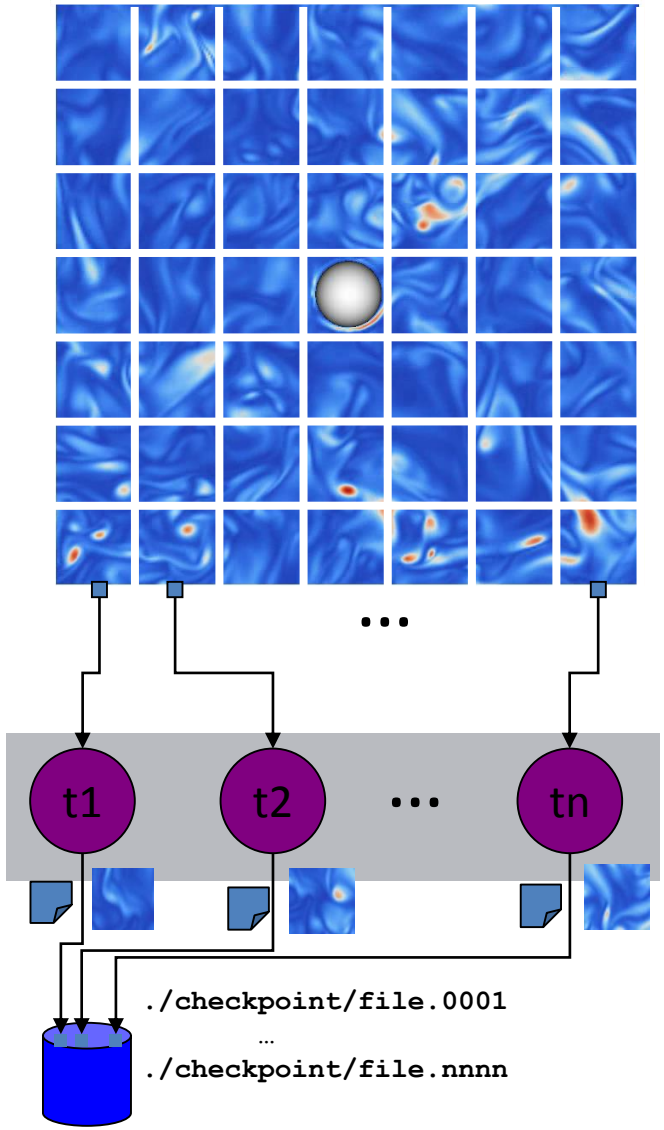
Data types:

- Simulation data (domain-decomposition)
- Trace data (parallel performance tools)

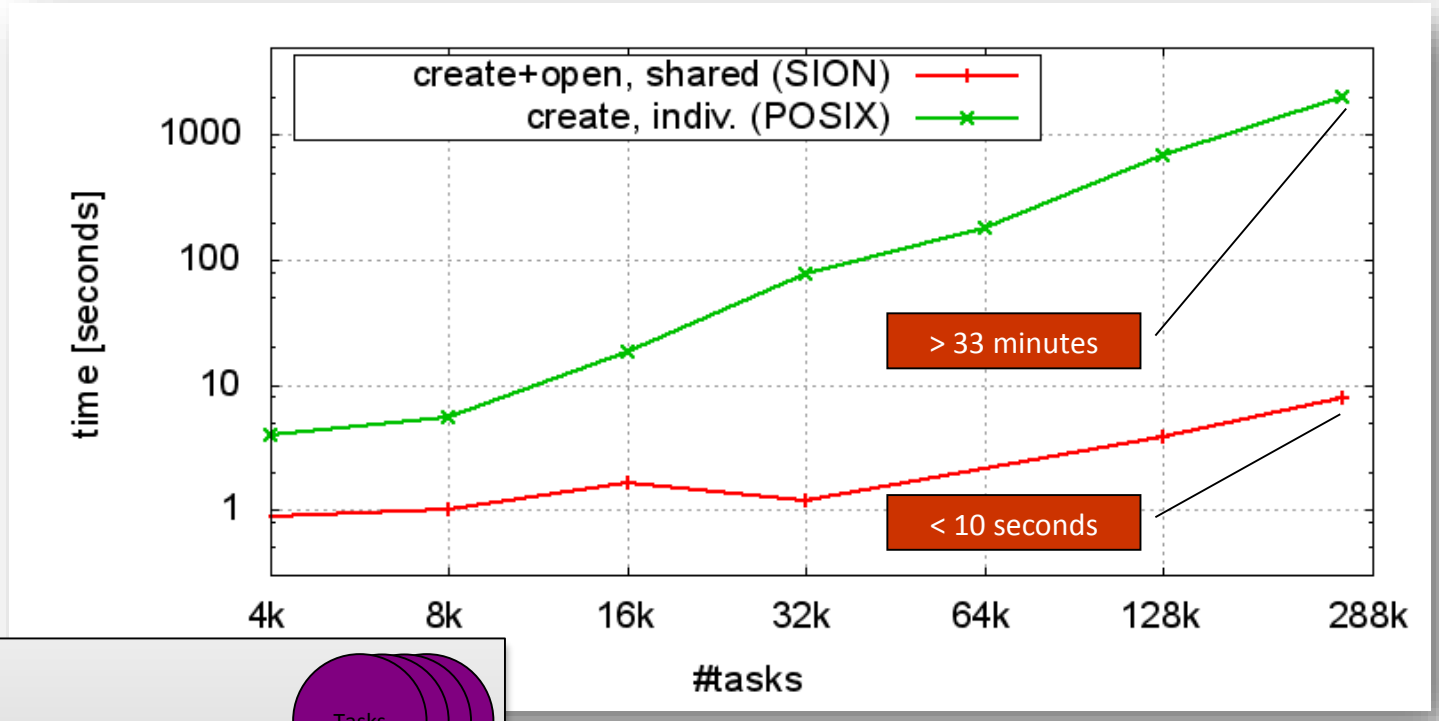
Bottlenecks:

- File creation
- File management

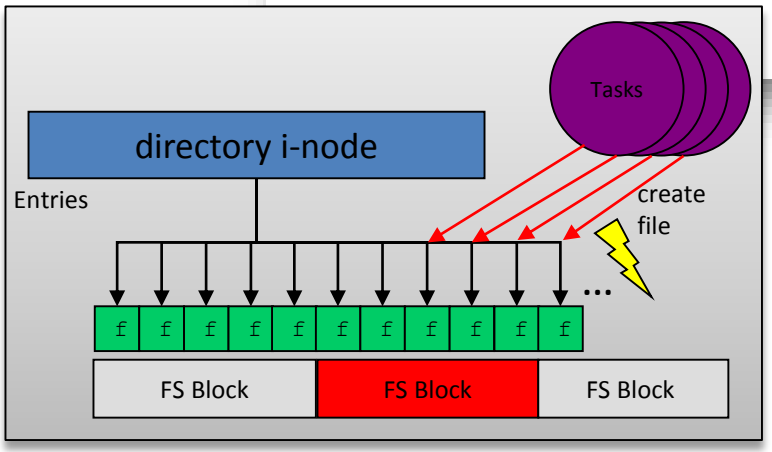
→ #files: $O(10^5)$



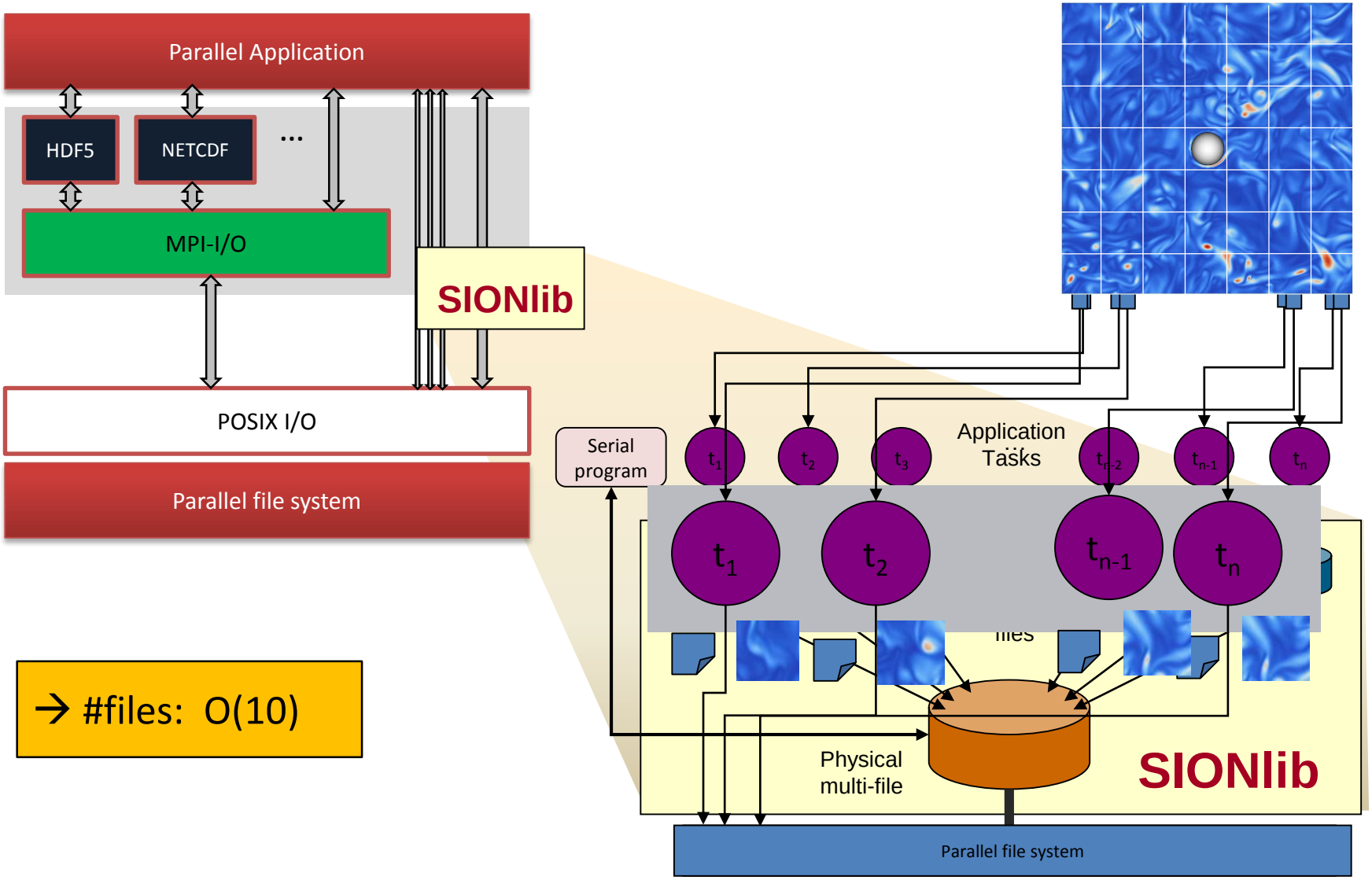
The Showstopper for Task-local I/O: … Parallel Creation of Individual Files



Jugene + GPFS: file create+open, one file per task versus one file per I/O-node

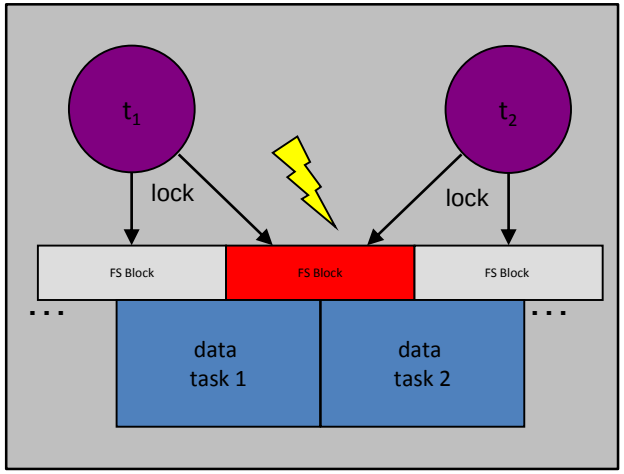


SIONlib: Shared Files for Task-local Data



→ #files: O(10)

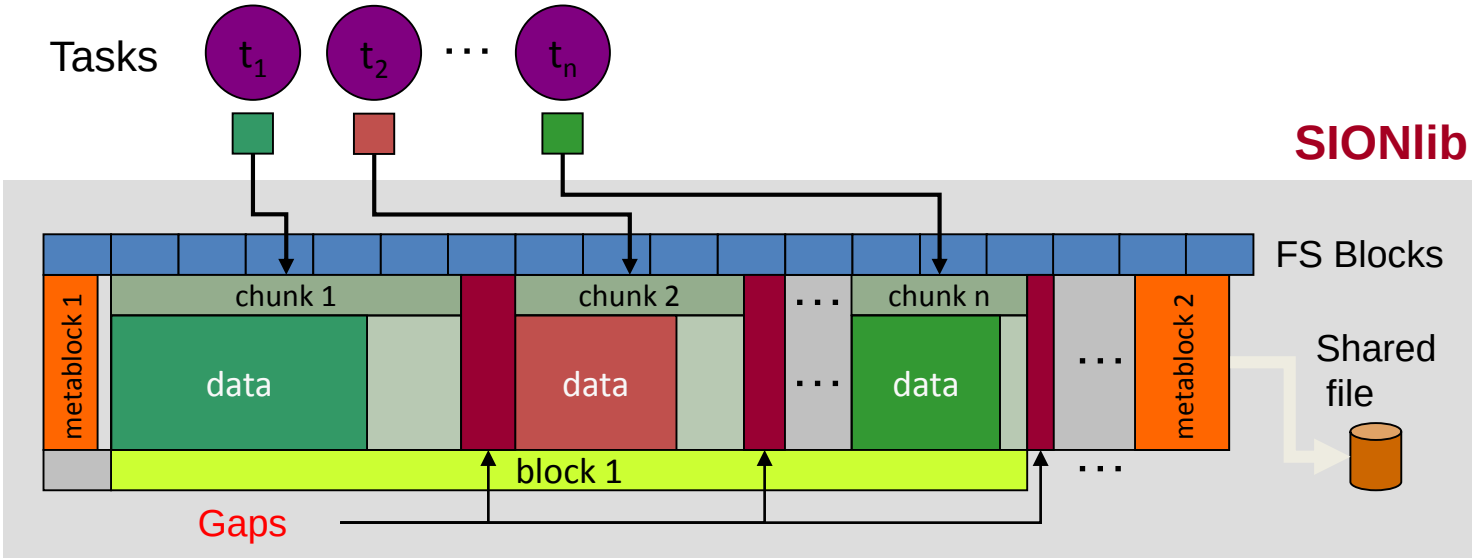
The Showstopper for Shared File I/O: … Concurrent Access & Contention



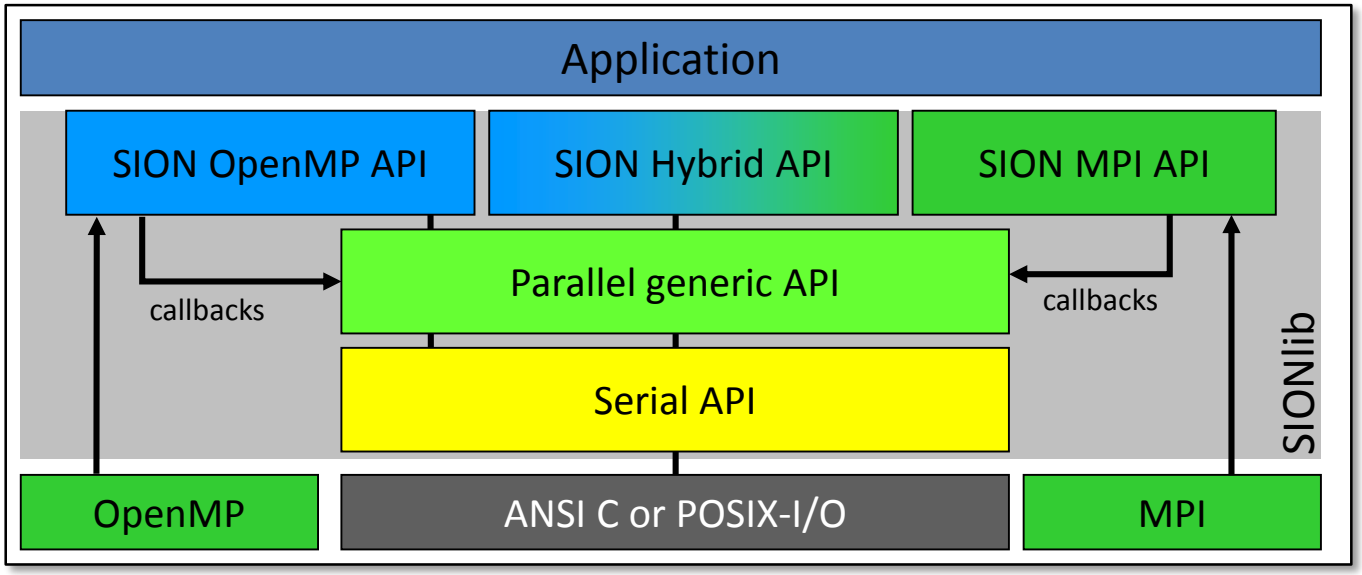
File System Block Locking → Serialization

SIONlib: Logical partitioning of Shared File:

- Dedicated data chunks per task
- Alignment to boundaries of file system blocks → no contention



SIONlib: Architecture & Example



- Extension of I/O-API (ANSI C or POSIX)
- C and Fortran bindings, implementation language C
- Current versions: 1.3p7, 1.4b2
- Open source license: http://www.fz-juelich.de/jsc/sionlib

```
/* fopen() → */
sid=sion_paropen_mpi( filename , "bw",
                      &numfiles, &chunksize,
                      gcom, &lcom, &fileptr, ...);

/* fwrite(bindata,1,nbytes, fileptr) → */
sion_fwrite(bindata,1,nbytes, sid);

/* fclose() → */
sion_parclose_mpi(sid)
```

SIONlib in a NutShell: … Task local I/O

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

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

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

- Original ANSI C version
- no collective operation, no shared files
- data: stream of bytes

SIONlib in a NutShell: … Wrapper function

```
/* Collective Open */
nfiles=1;chunksize=nbytes;
sid=sion_paropen_mpi( filename, "bw", &nfiles ,
                      MPI_COMM_WORLD, &lcomm,
                      &chunksize, &fsblksize,
                      &globalrank, &fileptr ,
                      &newfname);
...
/* Write */
sion_fwrite(bindata,1,nbytes,sid);
...
/* Collective Close */
sion_parclose_mpi(sid);
```

- Includes check for space in current chunk
- Parameter of fwrite: fileptr → sid

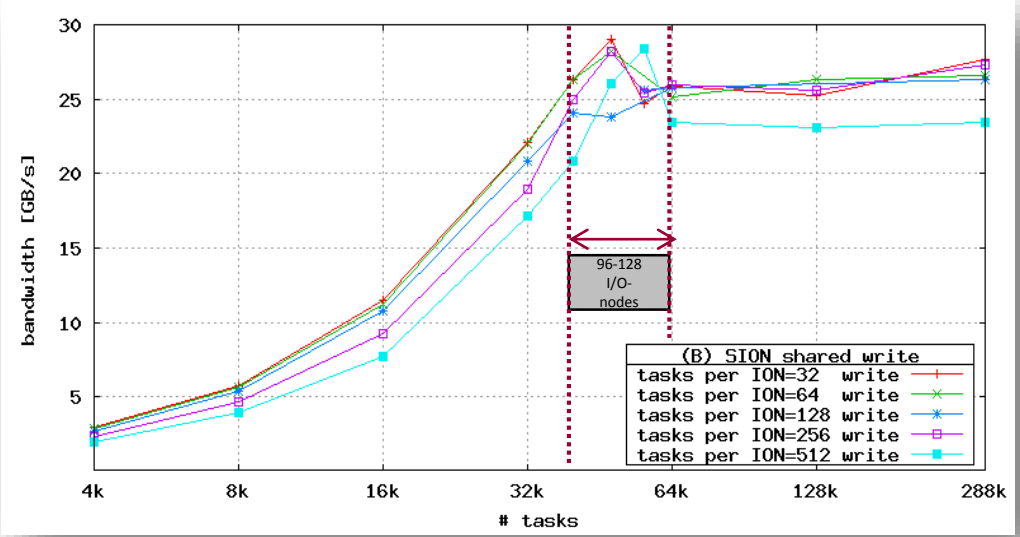
Hands-on: SIONLib, Pt. 1

- Go to the `IO/SIONlib_simple` directory
- Read over `base.c`
- Load the module `SIONlib/1.5.2-gompi-1.6.10`
- Compile the code with `make base`
- Run it with `128` as the argument
- Take a look at `base_with_variables_write.c`
  - Add a SIONlib file open for write using the arguments provided
  - Write the matrix to disk
  - Make it with `make base_with_variables_write`
  - Run it and analyse output

Hands-on: SIONLib, Pt. 2

- Next we are going to read the file
- Take a look at `base_with_variables_read.c`
  - Add a SIONlib file open for read using the arguments provided
  - Read the matrix from disk
  - Make it with `make base_with_variables_read`
  - Run it and analyse output
  - Compare the answers from the write output

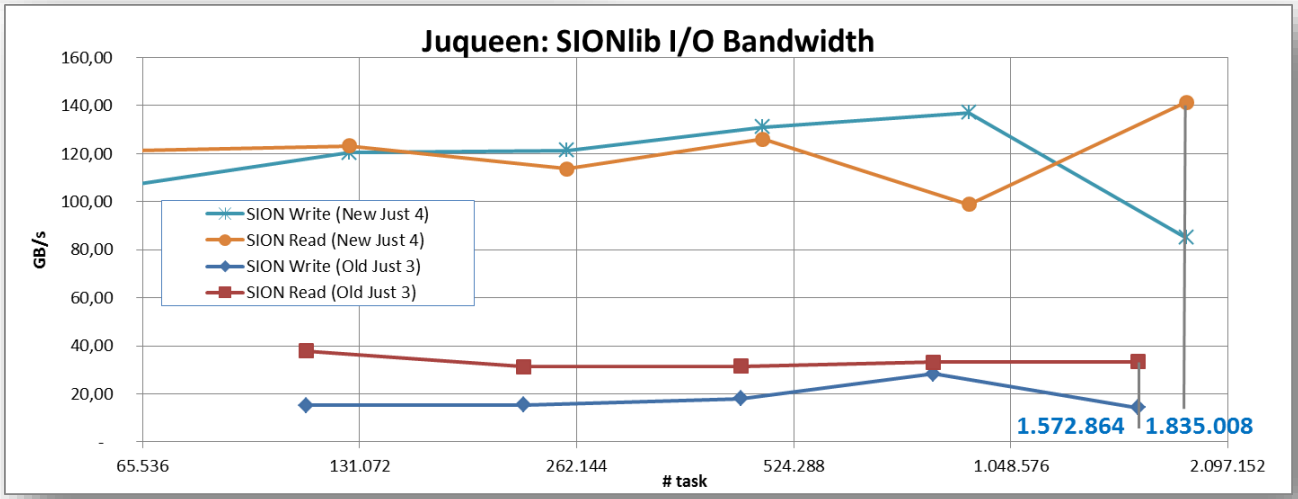
SIONlib: Scaling to Large # of Tasks



***JUGENE**: Total bandwidth (write), one file per I/O-node (ION), varying the number of tasks doing the I/O*

Preliminary Tests
on JUQUEEN
up to **1.8 Mio Tasks**

***JUQUEEN**: Total bandwidth (write/read) , one file per I/O-bridge (IOB)*
Old (Just3) vs. New (Just4) GPFS file system



(One) Pitfall: Portability

- Endianess (byte order) of binary data
- Example (32 bit): 2.712.847.316

=

10100001 10110010 11000011 11010100

Address	Little Endian	Big Endian
1000	11010100	10100001
1001	11000011	10110010
1002	10110010	11000011
1003	10100001	11010100

- Conversion of files might be necessary and expensive
- Solution: Choosing a portable data format (HDF5, NetCDF)

How to choose an I/O strategy?

- Performance considerations
  - Amount of data
  - Frequency of reading/writing
  - Scalability

- Portability
  - Different HPC architectures
  - Data exchange with others
  - Long-term storage

- E.g. use two formats and converters:
  - Internal: Write/read data “as-is”
    → Restart/checkpoint files
  - External: Write/read data in non-decomposed format (portable, system-independent, self-describing)
    → Workflows, Pre-, Postprocessing, Data exchange, …

Special thanks to Wolfgang Frings (JSC)

- These slides are a shortened version of his presentation at:
  - http://www.vi-hps.org/upload/material/tw12/Parallel_IO_VIHPS_13_wfrings.pdf

- Provided base for examples used in tutorials