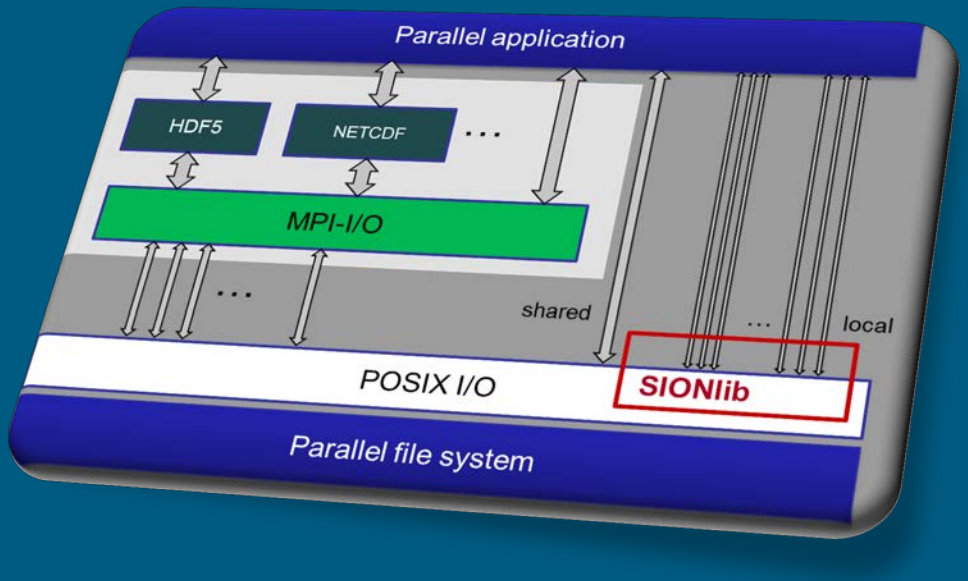
JÜLICH
FORSCHUNGSZENTRUM

Parallel file I/O bottlenecks and solutions

- → Views to Parallel I/O: Hardware, Software, Application
- → Challenges at Large Scale
- → Introduction SIONlib
- → Pitfalls, Darshan, I/O-Strategies



Wolfgang Frings

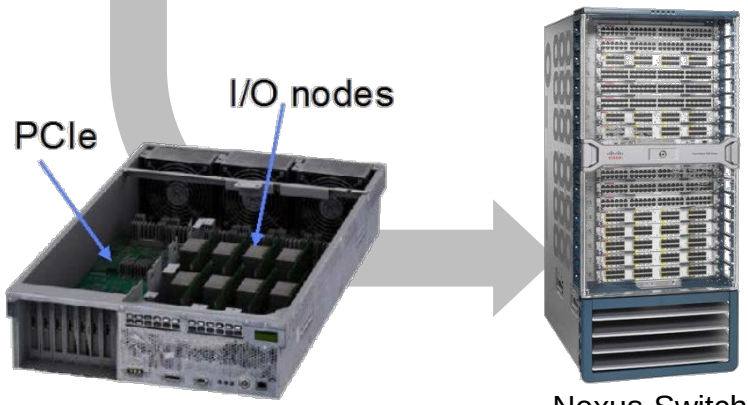
W.Frings@fz-juelich.de

Jülich Supercomputing Centre

12th VI-HPS Tuning Workshop, JSC, Jülich 2013

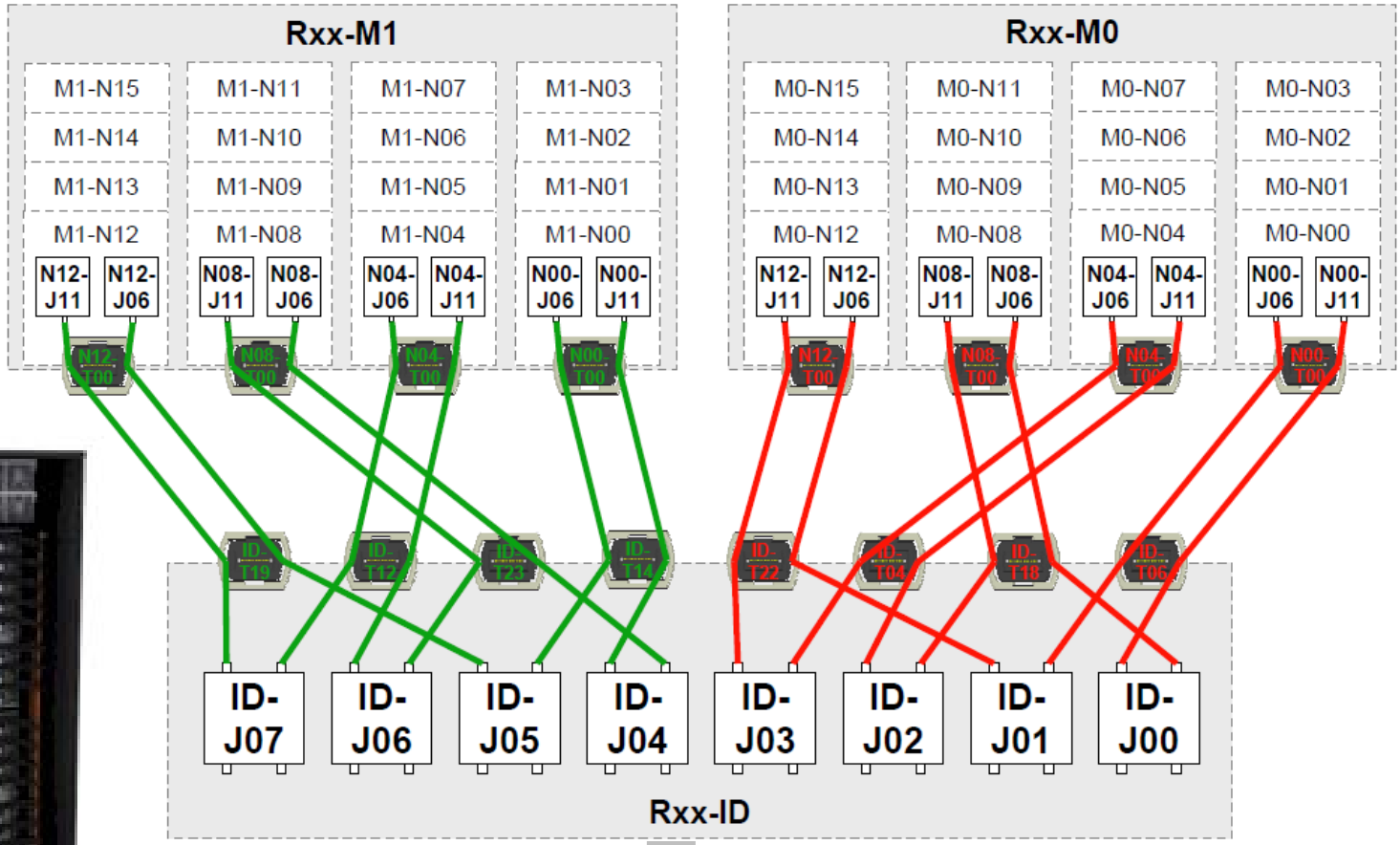
JUQUEEN: Jülich's Scalable Petaflop System

- IBM Blue Gene/Q JUQUEEN
- IBM PowerPC® A2 1.6 GHz,
  16 cores per node
  28 racks (7 rows à 4 racks)
  28,672 nodes (**458,752 cores**)
- 5D torus network
- 5.9 Pflop/s peak
  5.0 Pflop/s Linpack
- Main memory: **448 TB**
- **I/O Nodes**: **248** (27x8 + 1x32)
- **Network**: 2x CISCO Nexus 7018
  Switches (connect I/O-nodes)
  Total ports: **512 10 GigEthernet**

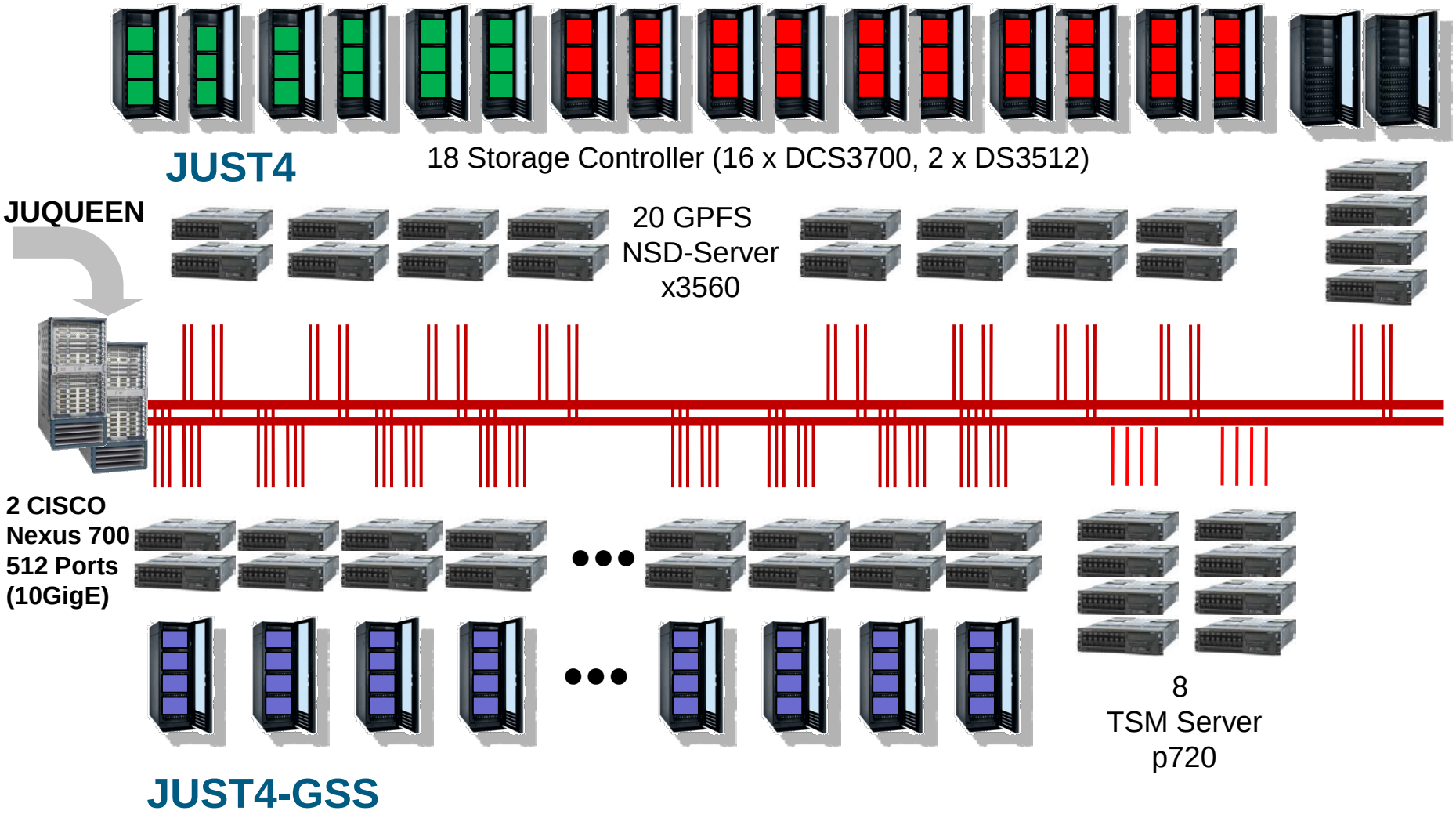


Nexus-Switch

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



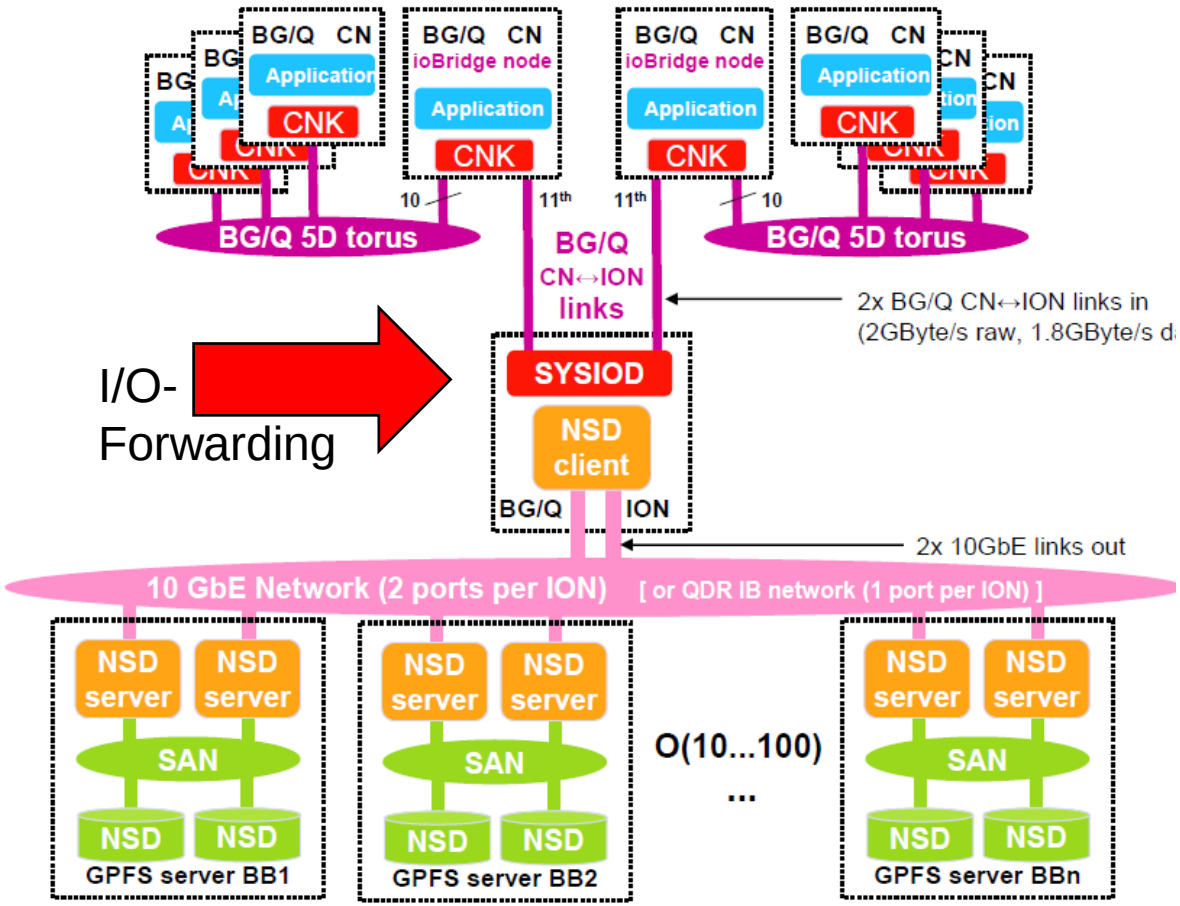
JUQUEEN and JUST I/O-Network

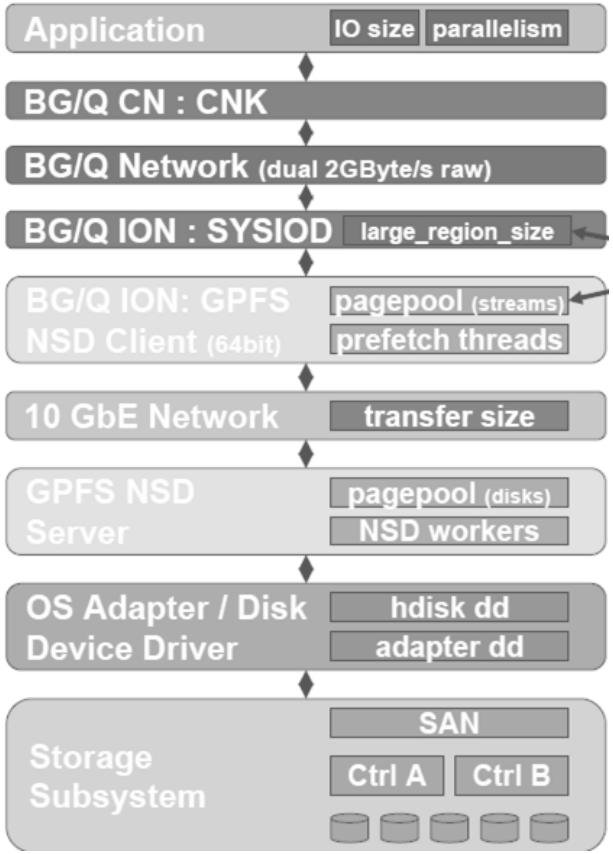


Parallel I/O Hardware at JSC (Just4, GSS)

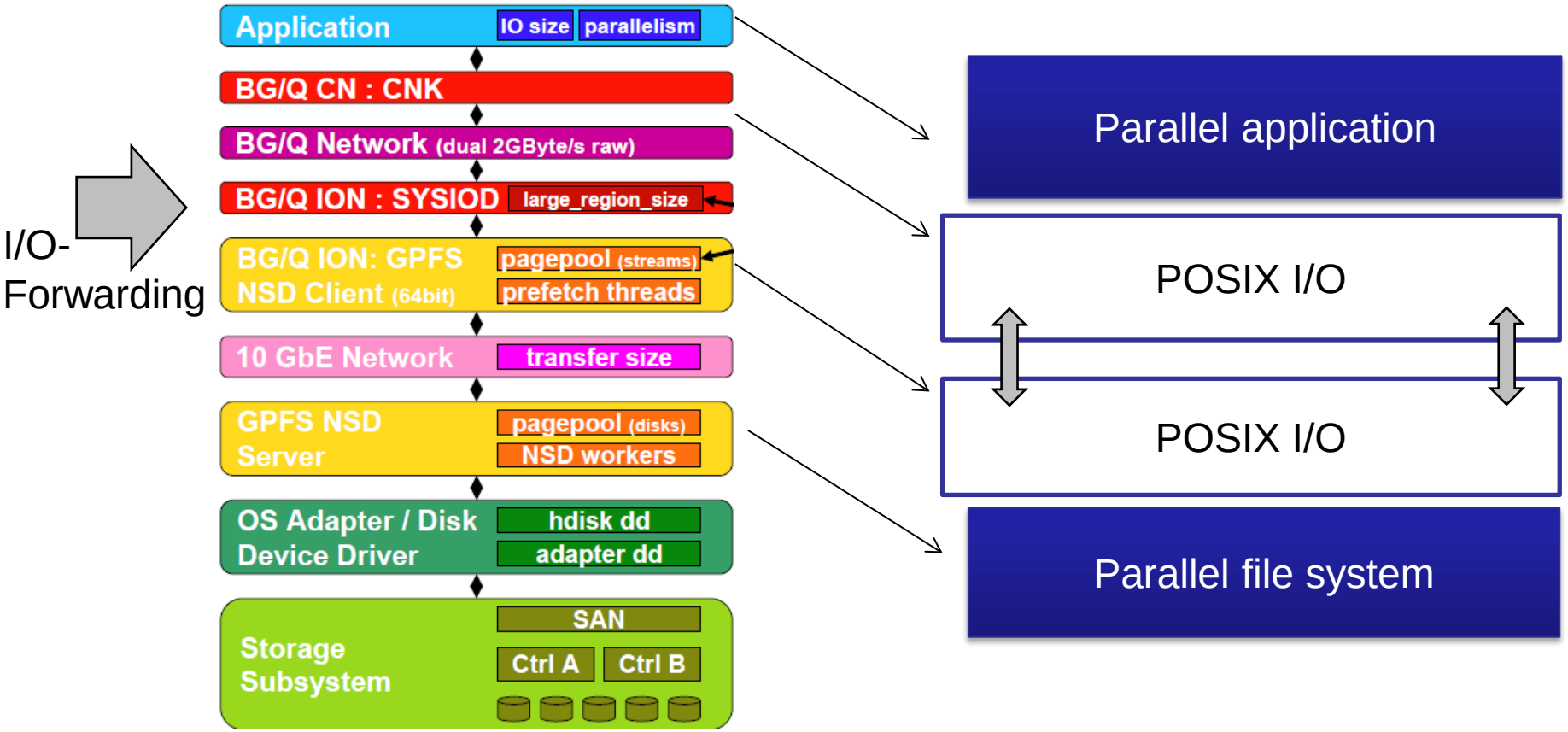
- **Ju**elich **St**orage Cluster (JUST)
- **Just4** (04/2012)
  - GPFS-Filesystems $HOME, $ARCH
  - Capacity: 3.4 Pbyte
  - Hardware: 2x DS3512, 16x DCS3700
- **Just4-GSS** (09/2013)
  - GPFS-Filesystem $WORK
  - Capacity: **7.4 Pbyte**
    I/O Bandwidth: up to **200 GB/sec**
  - Hardware: IBM System x® GPFS™ Storage Server solution, GPFS Native RAID
  - 20 Building blocks: each 2 x X3650 M4 server, 232 NL-SAS disks (2TB), 6 SSD

Software View to Parallel I/O: … GPFS on IBM Blue Gene/Q (I)

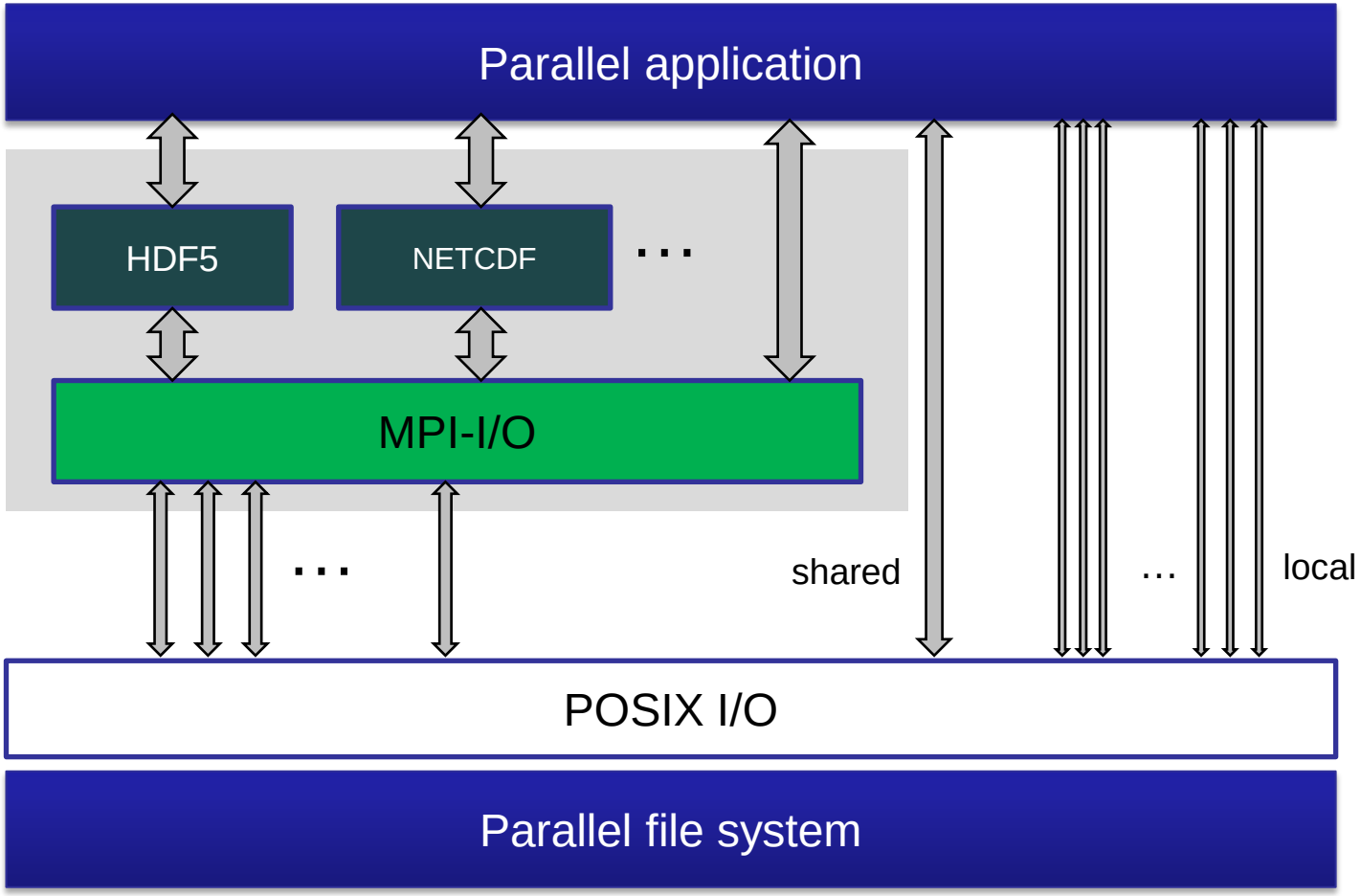




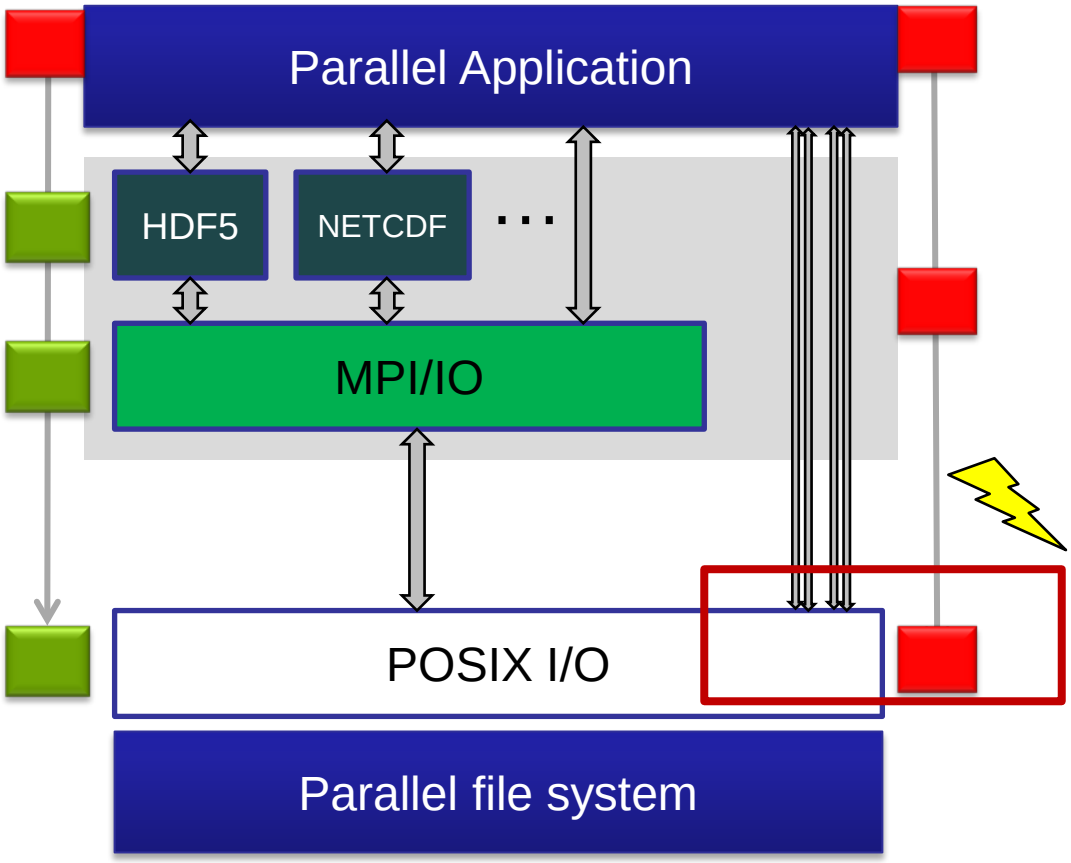
Software View to Parallel I/O: … GPFS on IBM Blue Gene/Q (II)



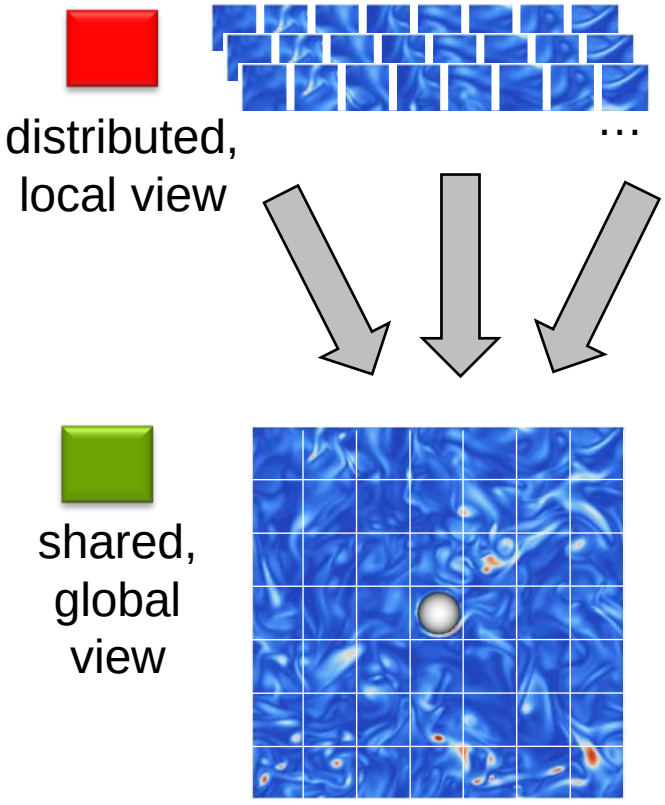
Application View to Parallel I/O



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

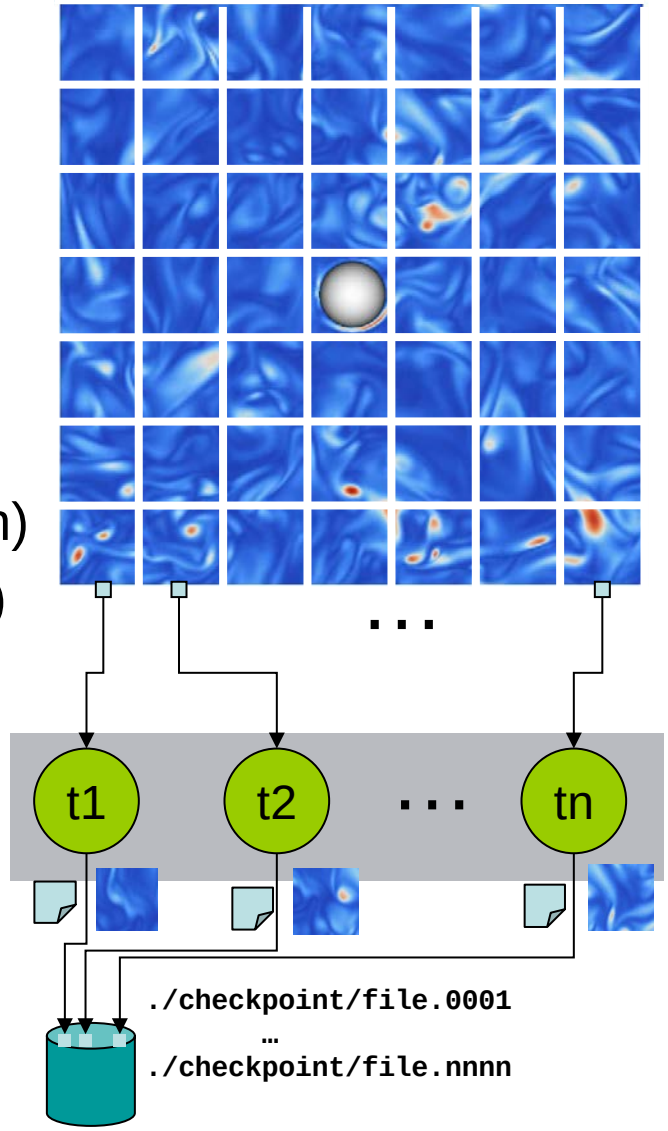
Data types:

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

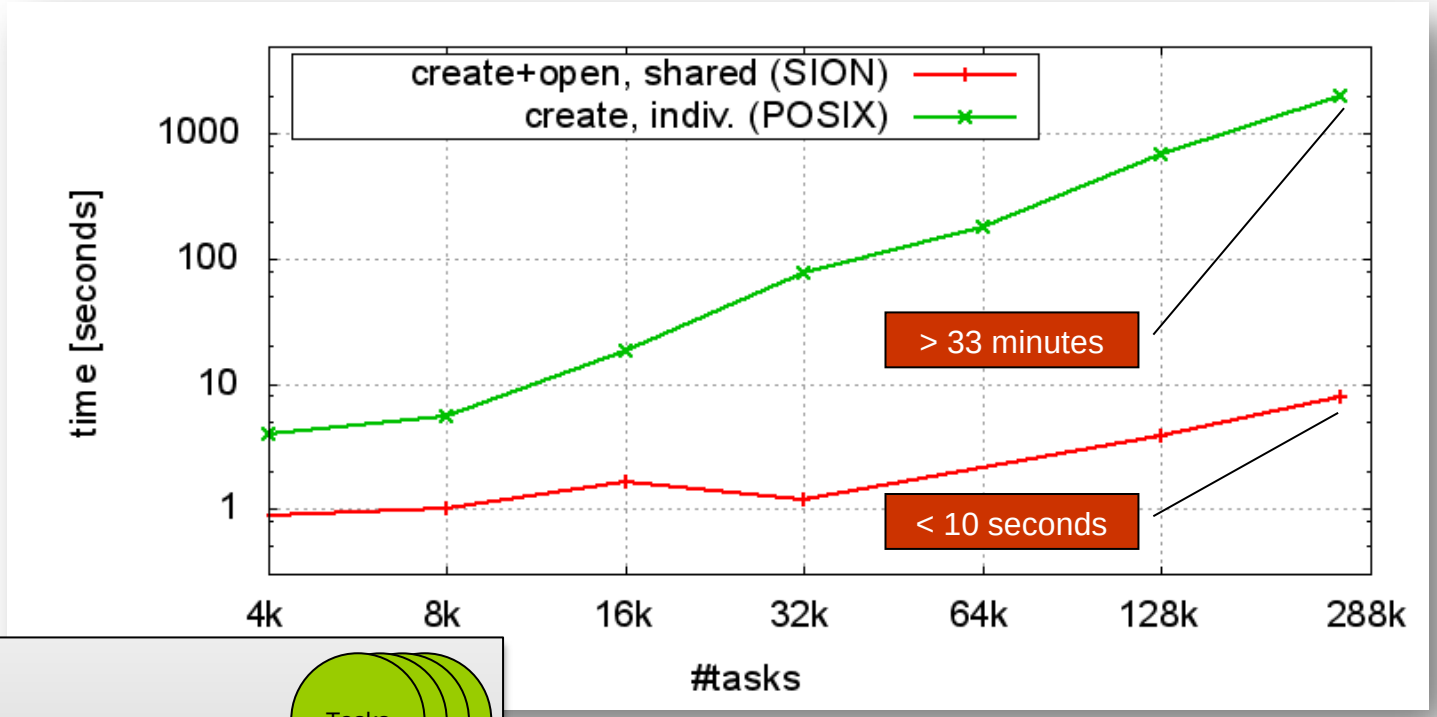
Bottlenecks:

- File creation
- File management

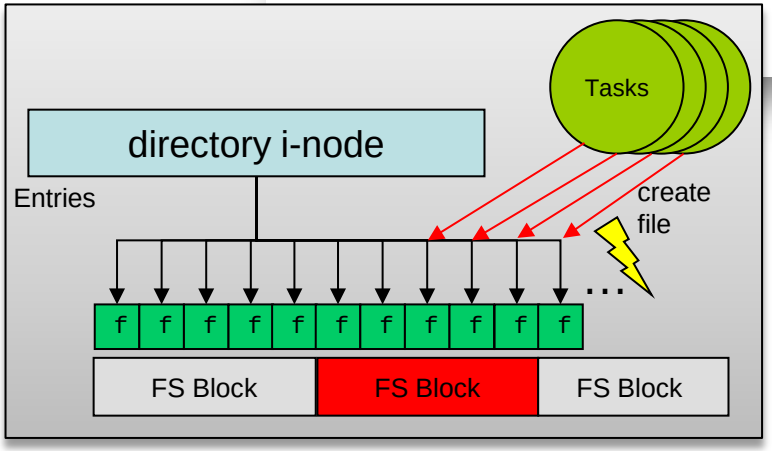
→ #files: $O(10^5)$



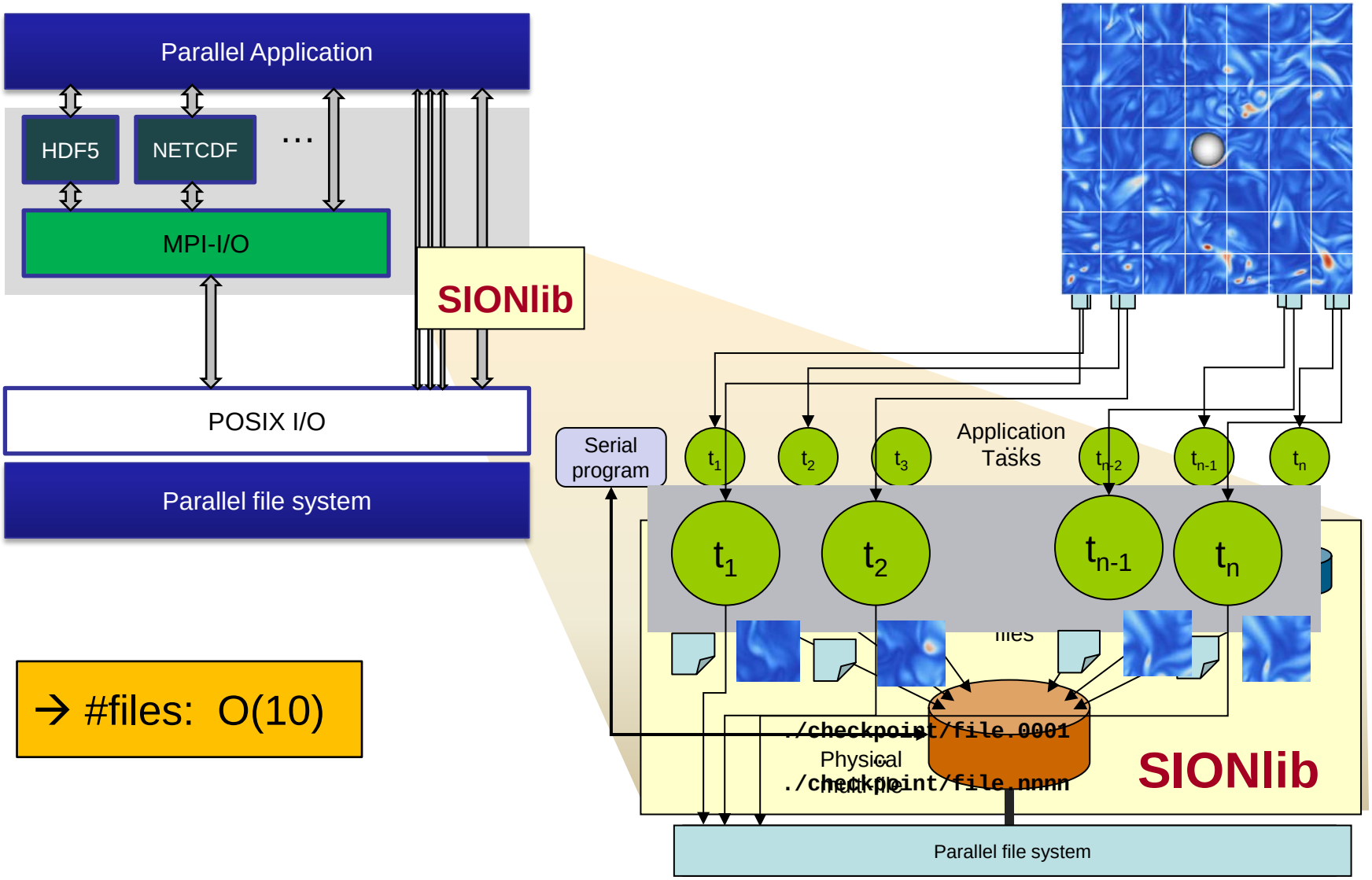
The Showstopper for Task-local I/O: … Parallel Creation of Individual Files



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

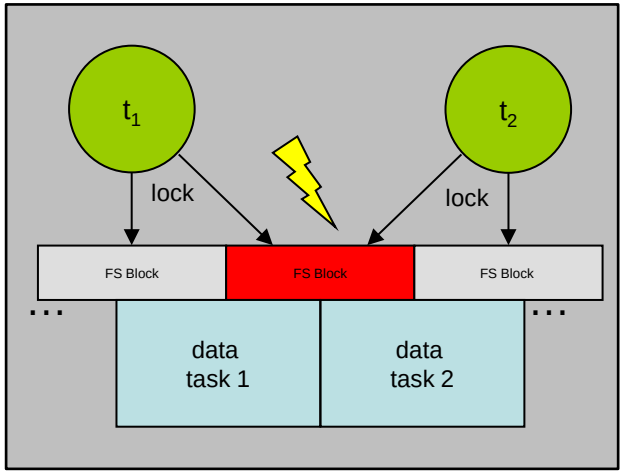


SIONlib: Shared Files for Task-local Data



→ #files: O(10)

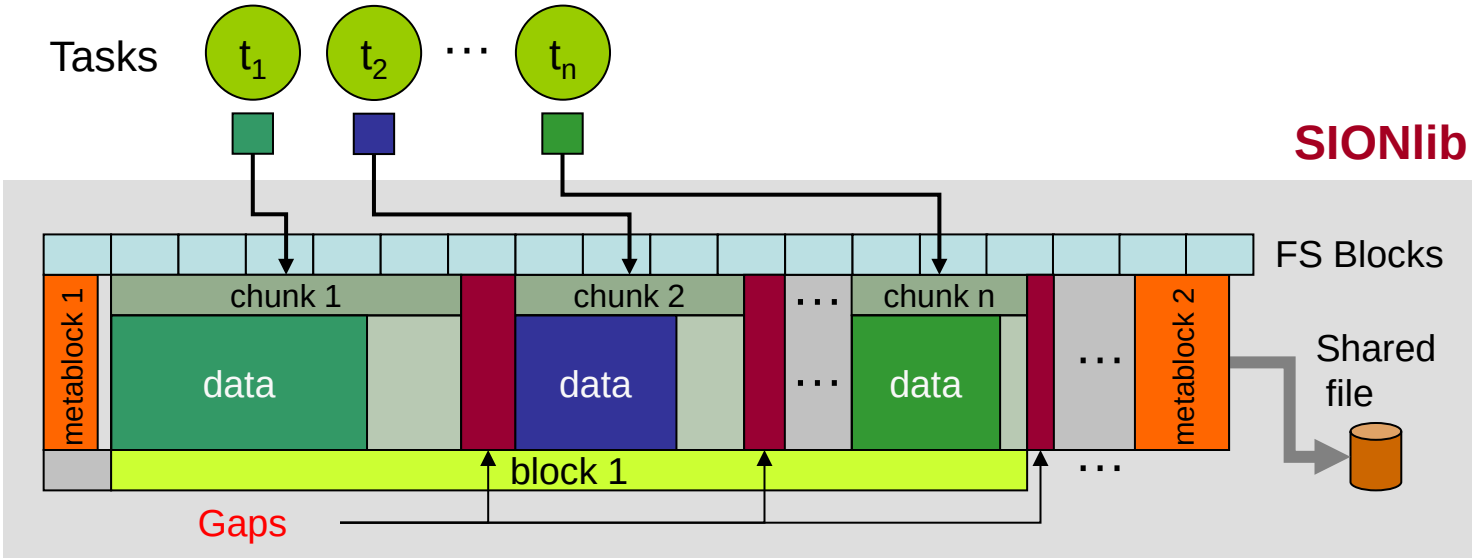
The Showstopper for Shared File I/O: … Concurrent Access & Contention



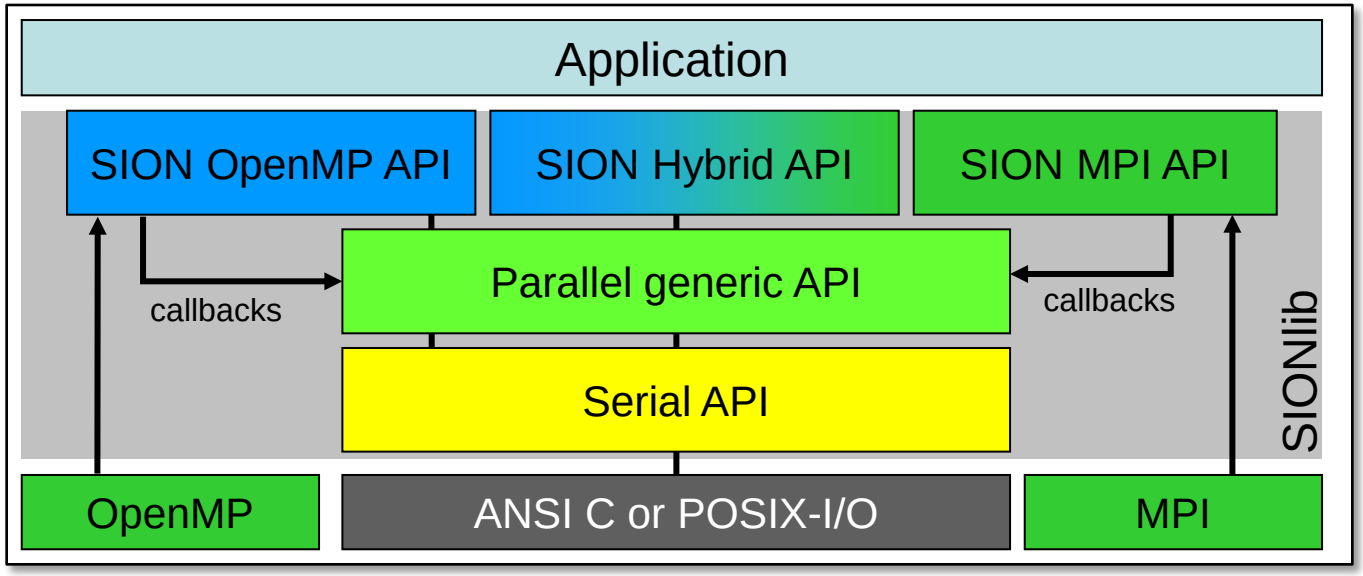
File System Block Locking → Serialization

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: … Add SIONlib calls

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

- Collective (SIONlib) open and close
- Ready to run ...
- Parallel I/O to one shared file

SIONlib in a NutShell: … Variable Data Size

```
/* Collective Open */
nfiles=1;chunksize=nbytes;
sid=sion_paropen_mpi( filename, "bw", &nfiles ,
                      &chunksize , MPI_COMM_WORLD,
                      &lcomm , &fileptr , ...);
...
/* Write */
if(sion_ensure_free_space(sid, nbytes)) {
   fwrite(bindata,1,nbytes,fileptr);
}
...
/* Collective Close */
sion_parclose_mpi(sid);
```

- Writing more data as defined at open call
- SIONlib moves forward to next chunk, if data to large for current block

SIONlib in a NutShell: … Wrapper function

```
/* Collective Open */
nfiles=1;chunksize=nbytes;
sid=sion_paropen_mpi( filename, "bw", &nfiles ,
                      &chunksize , MPI_COMM_WORLD,
                      &lcomm , &fileptr , ...);
...

/* Write */
sion_fwrite(bindata,1,nbytes,sid);

...
/* Collective Close */
sion_parclose_mpi(sid);
```

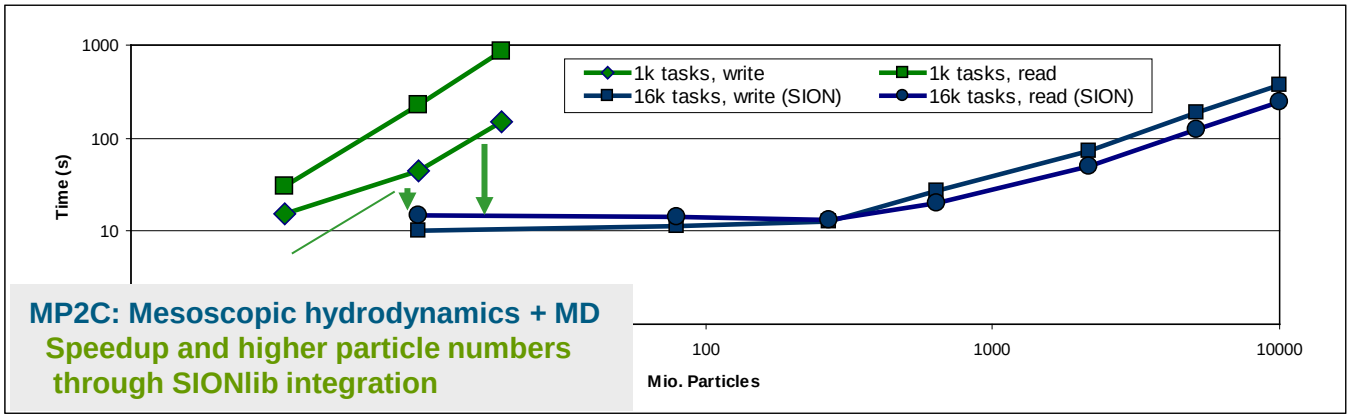
- Includes check for space in current chunk
- Parameter of fwrite: fileptr → sid

SIONlib: Applications

- **Applications**

DUNE-ISTL (Multigrid solver, Univ. Heidelberg) **ITM** (Fusion-community),
LBM (Fluid flow/mass transport, Univ. Marburg), **PSC** (particle-in-cell code),
OSIRIS (Fully-explicit particle-in-cell code), **PEPC** (Pretty Efficient Parallel C. Solver)
Profasi: (Protein folding and aggr. simulator) **NEST** (Human Brain Simulation)

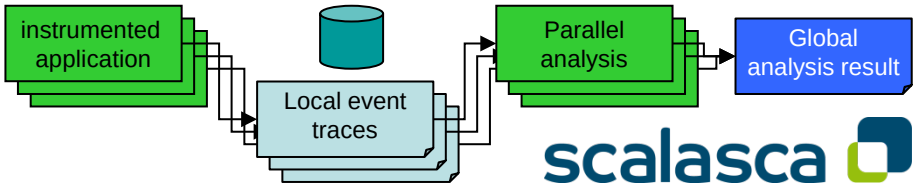
MP2C:



MP2C: Mesoscopic hydrodynamics + MD
Speedup and higher particle numbers through SIONlib integration

- **Tools/Projects**

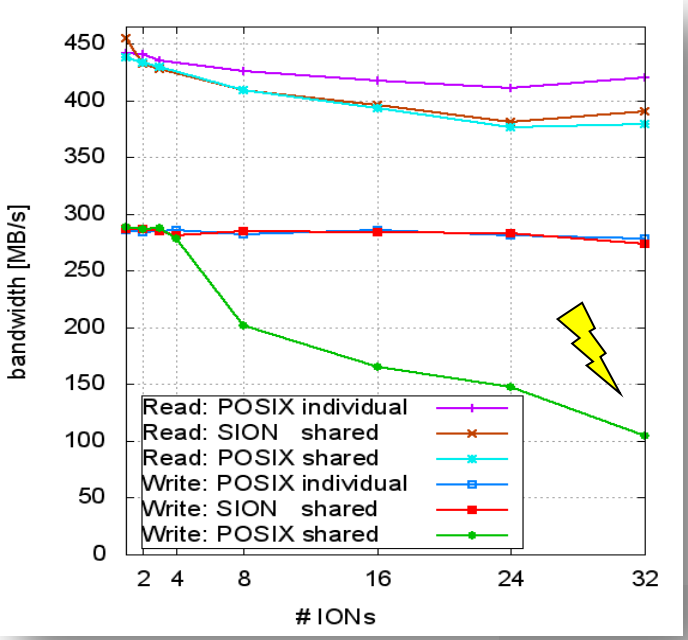
Scalasca: Performance Analysis



Score-P: Scalable Performance Measurement Infrastructure for Parallel Codes

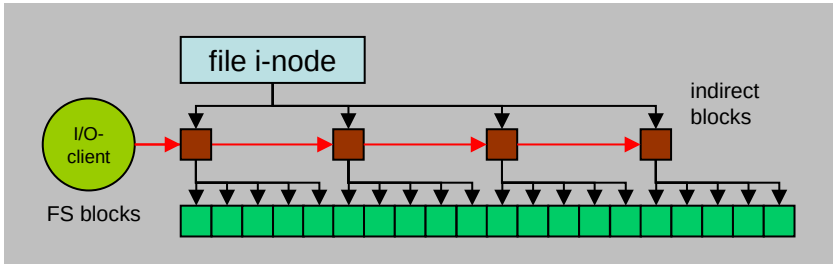
DEEP-ER: Adaption to new platform and parallelization paradigm

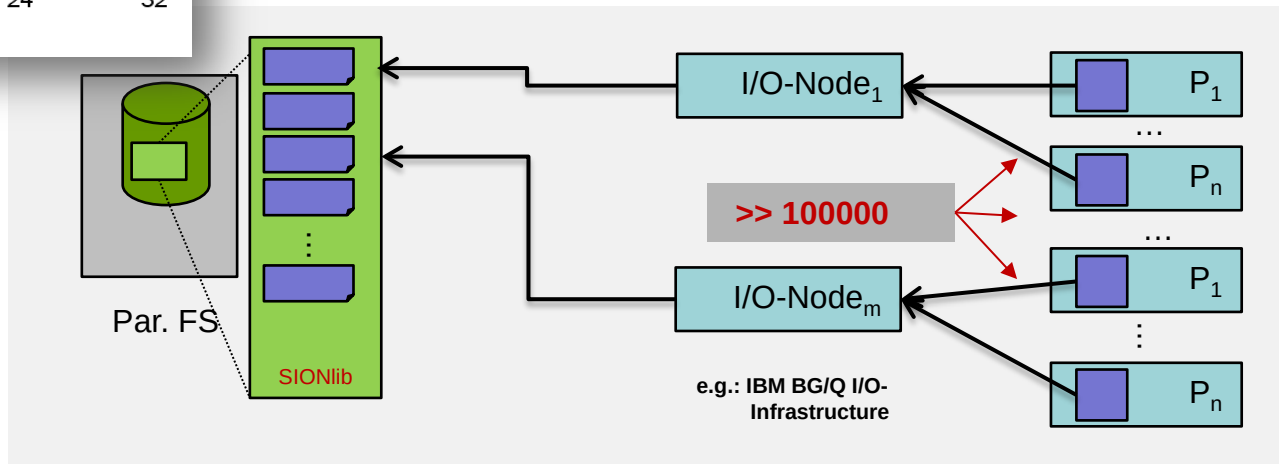
Are there more Bottlenecks?
… Increasing #tasks further …



JUGENE: Bandwidth per ION, comparison individual files (POSIX), one file per ION (SION) and one shared file (POSIX)

- Bottleneck: file meta data management
- by first GPFS client which opened the file



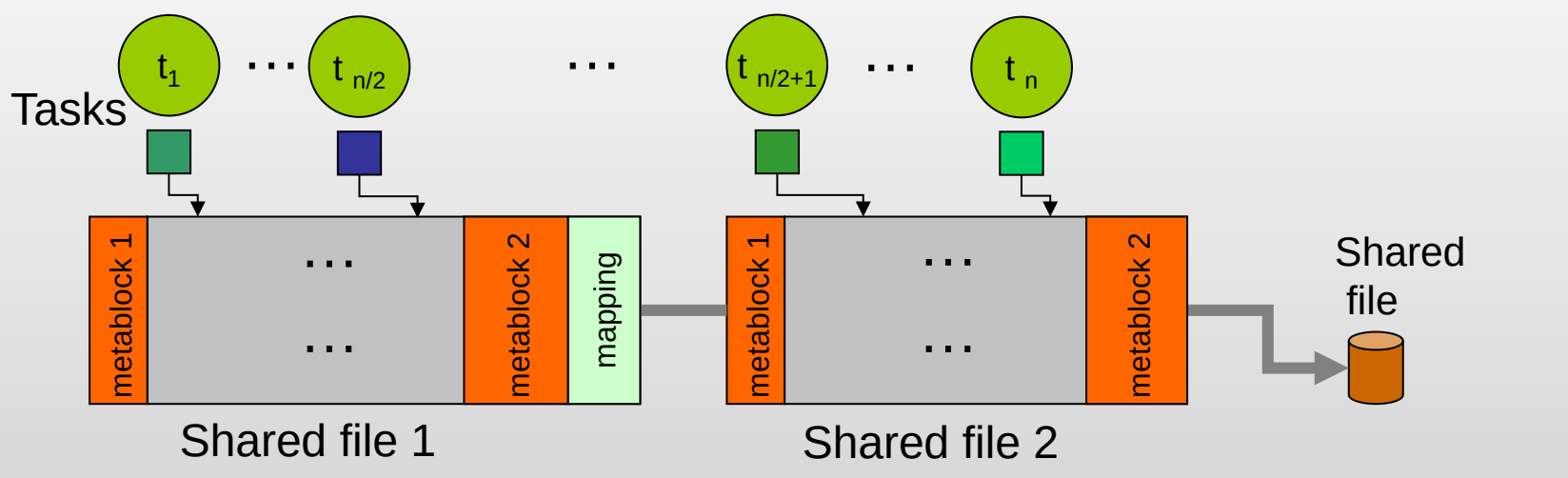


SIONlib: Multiple Underlying Physical Files

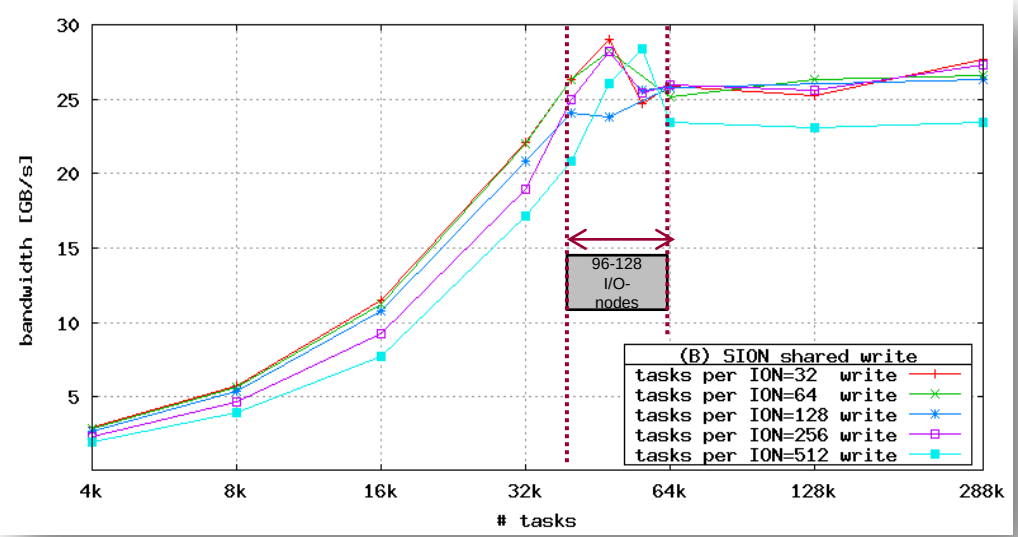
- Parallelization of file meta data handling using multiple physical files
- Mapping: Files : Tasks

1 : n ← **p : n** → n : n

→ IBM Blue Gene: One file per I/O-node (locality)



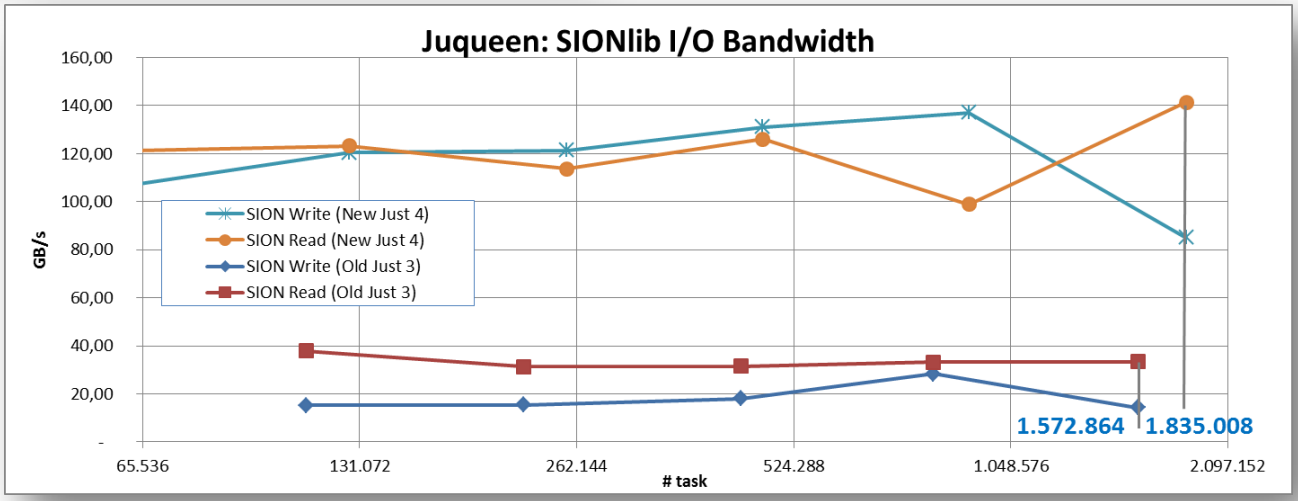
SIONlib: Scaling to Large # of Tasks



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

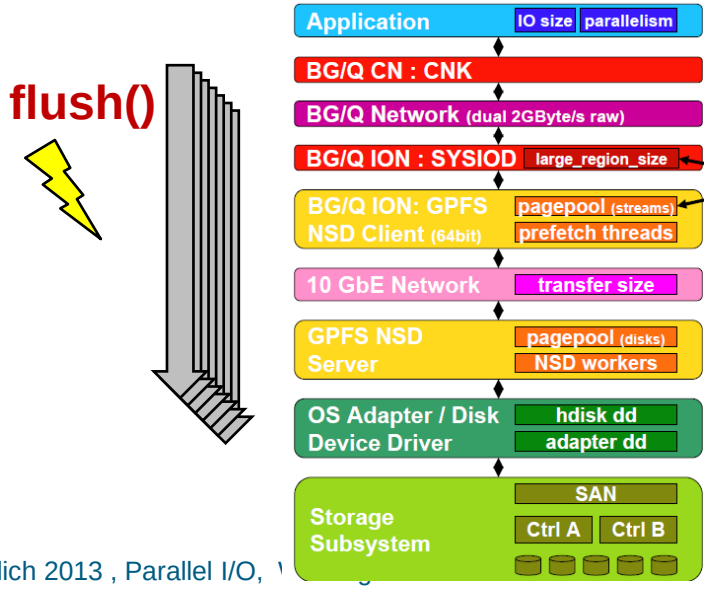
Preliminary Tests
on JUQUEEN
up to **1.8 Mio Tasks**

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



Other Pitfalls: Frequent flushing on small blocks

- Modern file systems in HPC have large file system blocks
- A flush on a file handle forces the file system to perform all pending write operations
- If application writes in small data blocks the same file system block it has to be read and written multiple times
- Performance degradation due to the inability to combine several write calls



Other Pitfalls: 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)

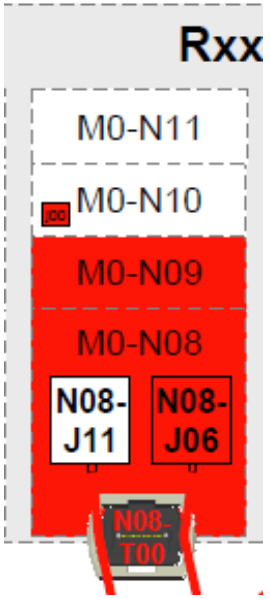
BGQ: Tasks connected to same I/O-node

- MPIX_Calls now available on BG/Q
  (see `http://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JUQUEEN/UserInfo/MPIextensions.html`)
- Communicator: All tasks belonging to same I/O Bridge Node

```
FORTRAN: MPIX_PSET_SAME_COMM_CREATE(INTEGER pset_comm_same,
                                    INTEGER ierr)
C: #include <mpix.h>
   int MPIX_Pset_same_comm_create( MPI_Comm *pset_comm_same )
```

- Usage: implementation of own I/O strategy
  (One file per I/O-bridge)
- Passing new communicator to SIONlib paropen-Call (as local communicator)

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



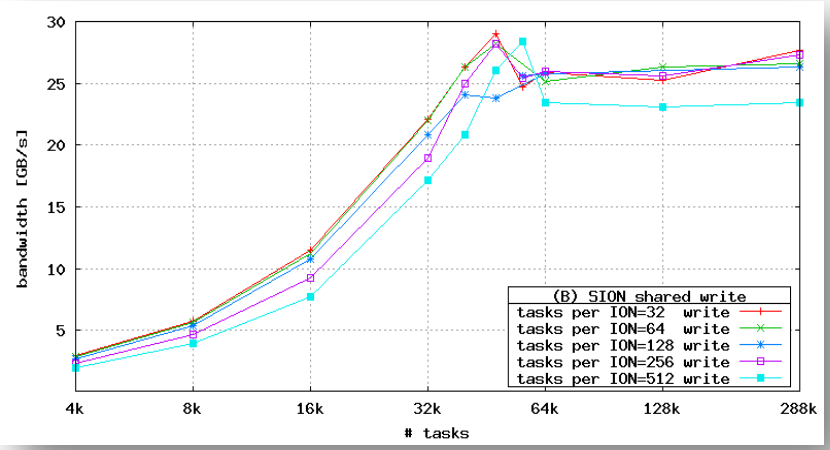
Darshan – I/O Characterization

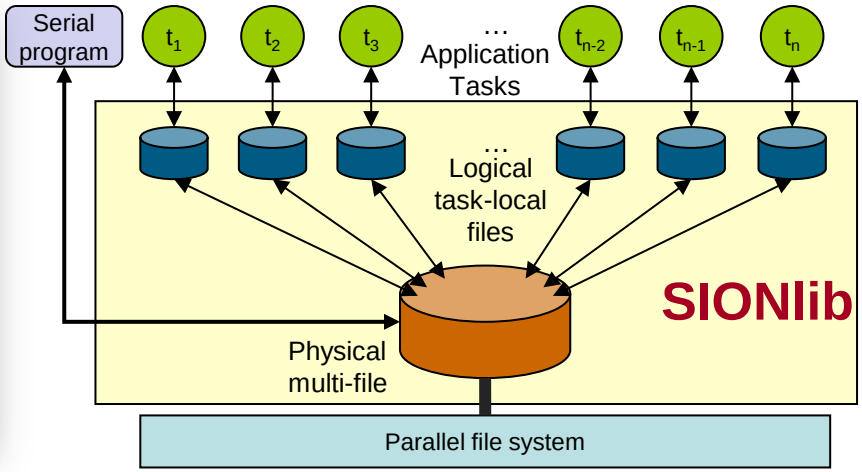
- Darshan: Scalable HPC I/O characterization tool (ANL)
  - http://www.mcs.anl.gov/darshan
- Profiling of I/O-Calls (POSIX, MPI-I/O, HDF5, NetCDF) during runtime
- Replaces Compiler-Calls (mpi*xxx)* by Darshan wrappers:
  - Re-link application, Re-run application → logfile
- Generate report from logfile:
  `darshan-job-summary` **`<logfile>`** → PDF-file
- On JUQUEEN:
  - **`module load darshan`**
    → version 2.2.4p (patched for JUQUEEN)
  - Report Path: ***`/work/darshan/<year>/<month>/<day>`***
  - ***`darshan-show-last-report.sh`***

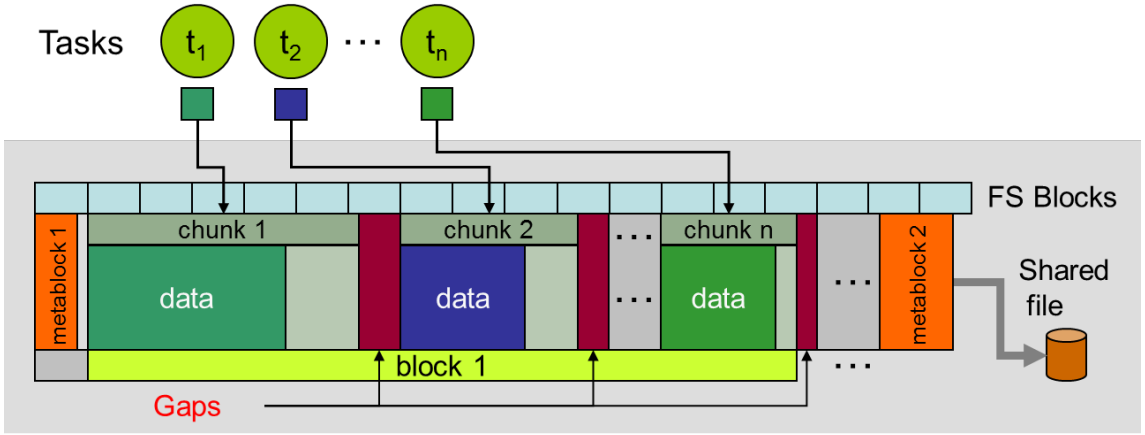
How to choose an I/O strategy?

- 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, …*

Questions ?







Thank You !