Parallel I/O on JUQUEEN

5. Februar 2013, JUQUEEN Porting and Tuning Workshop

Wolfgang Frings

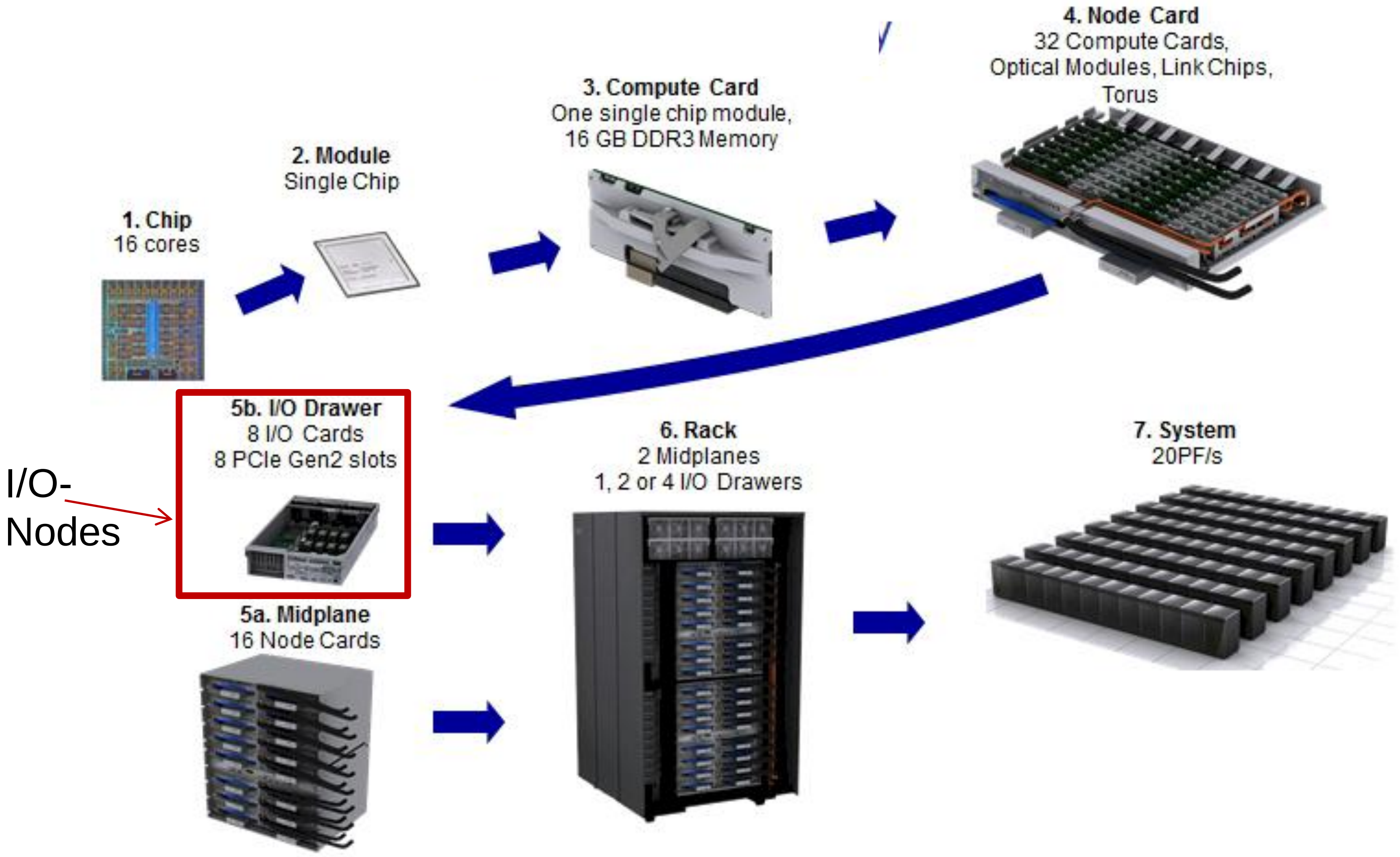
w.frings@fz-juelich.de

Jülich Supercomputing Centre

Overview

- Blue Gene/Q I/O Hardware
  - Overview, I/O-nodes, Cabling, Services
- GPFS and data path: cluster vs. BG/Q
- Pitfalls
  - Small blocks, I/O to individual files, false sharing
  - Tasks per Shared File, portability
- I/O Libraries & Software Stack
- SIONlib Overview
- Task-mapping to I/O-node
- I/O Characterization with Darshan

Blue Gene/Q Packaging Hierarchy



1. Chip
16 cores

2. Module
Single Chip

3. Compute Card
One single chip module,
16 GB DDR3 Memory

4. Node Card
32 Compute Cards,
Optical Modules, Link Chips,
Torus

5b. I/O Drawer
8 I/O Cards
8 PCIe Gen2 slots

I/O-
Nodes

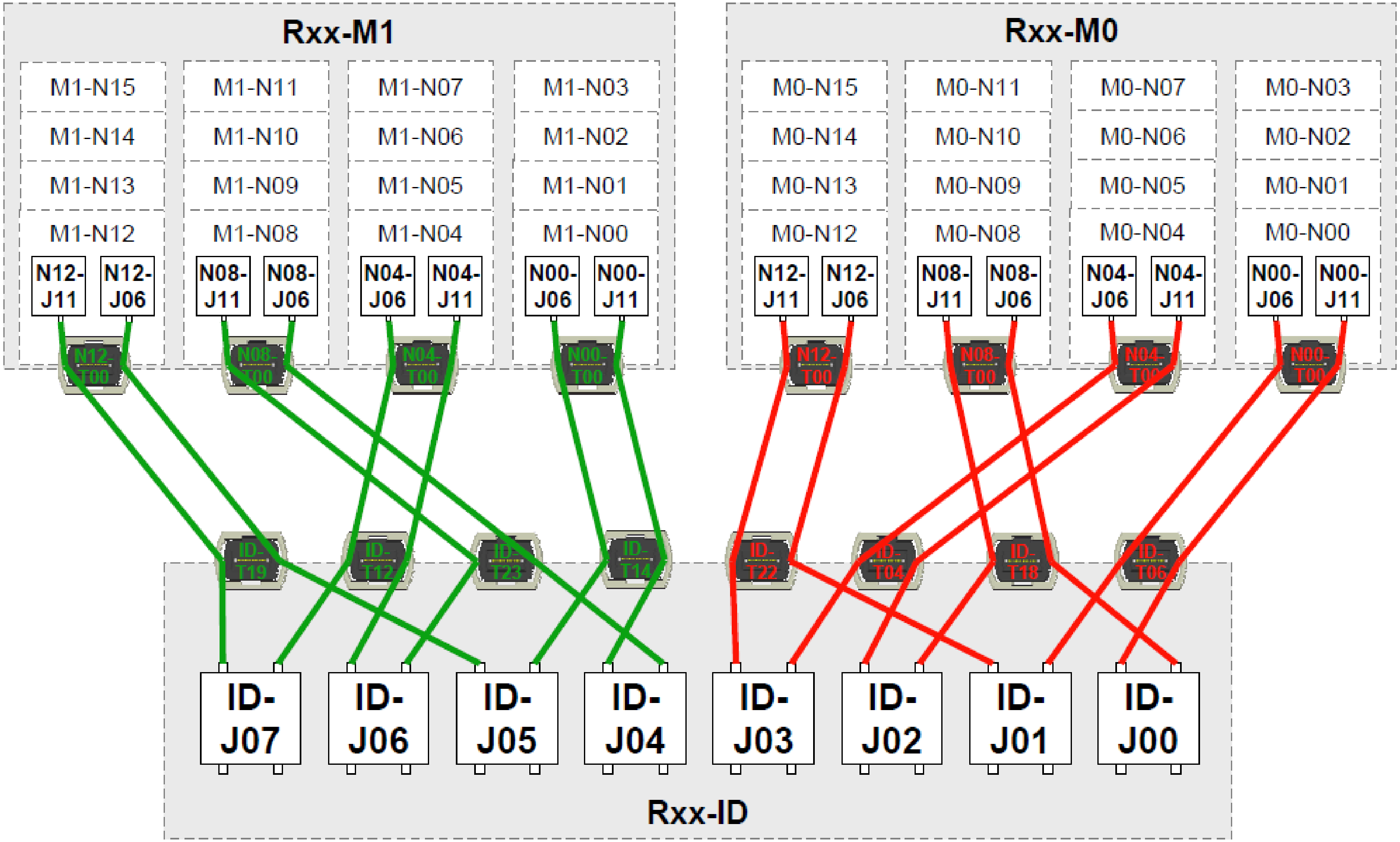
5a. Midplane
16 Node Cards

6. Rack
2 Midplanes
1, 2 or 4 I/O Drawers

7. System
20PF/s

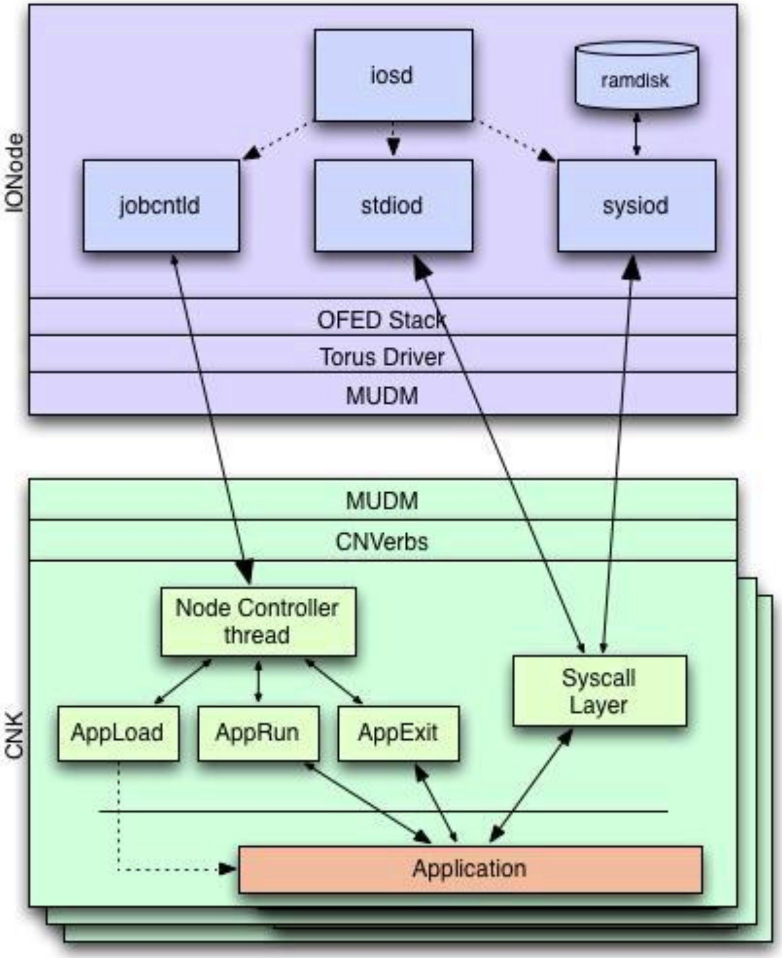
© IBM 2012

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



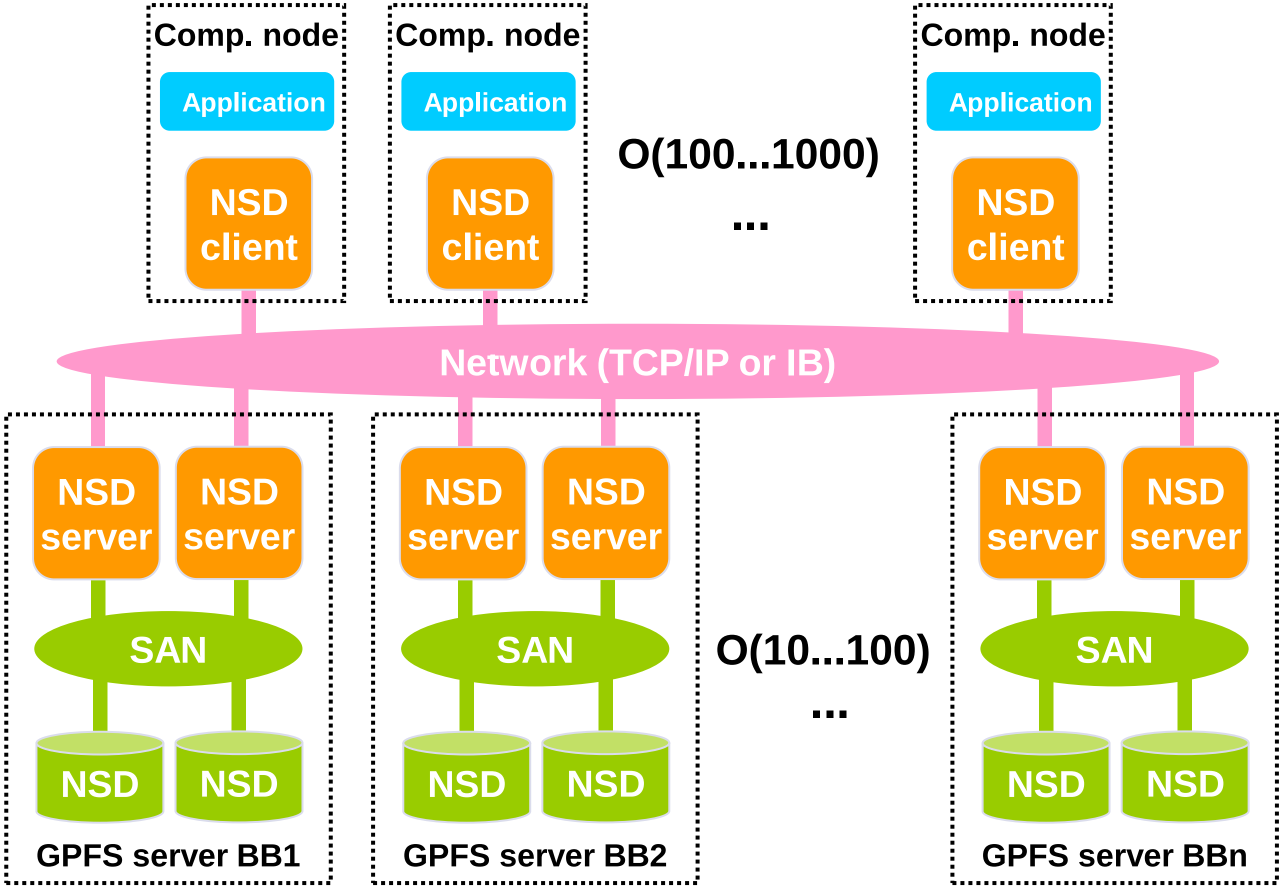
Blue Gene/Q: I/O Services

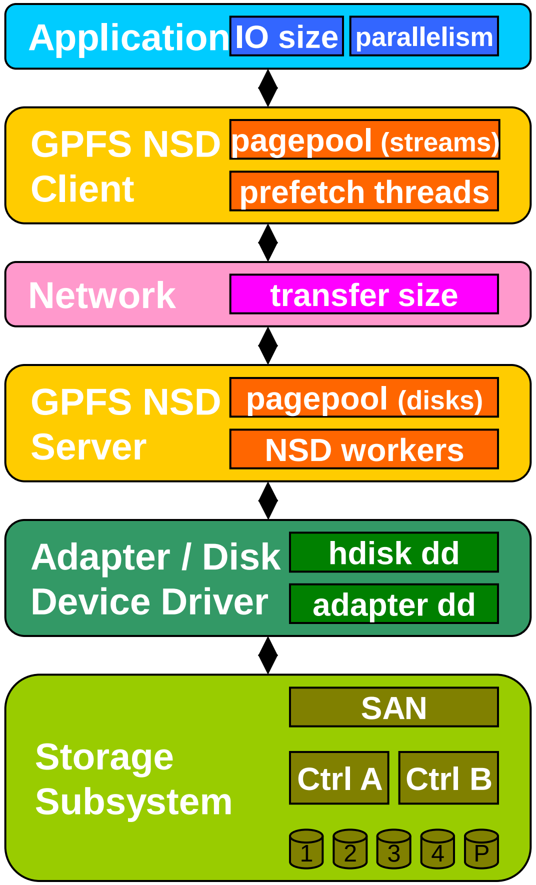
- Function shipping system calls to I/O-node
- Support NFS, GPFS, Lustre and PVFS2 filesystems
- PowerPC64 Linux running on 17 cores
- Supports ratio of **8192:1** compute task to I/O-node
  - Only 1 I/O-Proxy per compute node
  - Significant internal changes from BGP
- Standard communications protocol
  - OFED verbs
  - Using Torus DMA hardware for performance



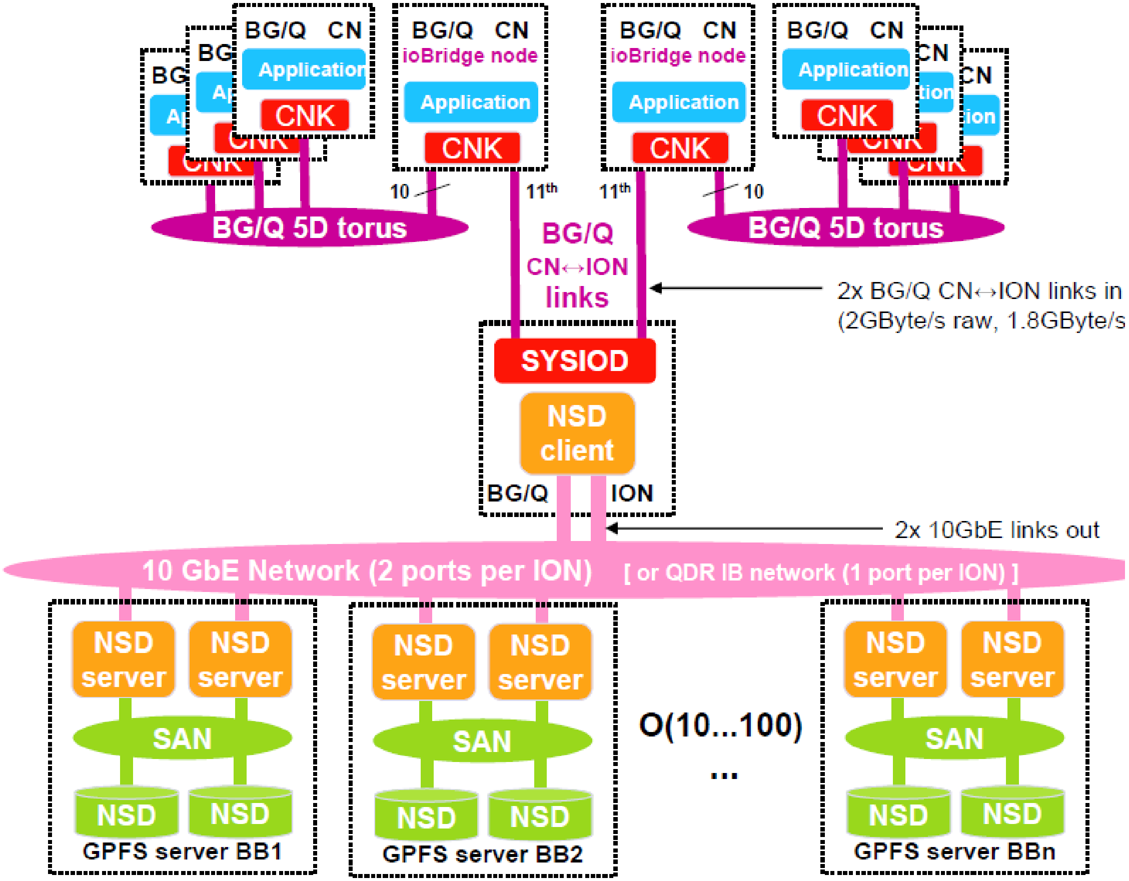
© IBM 2012

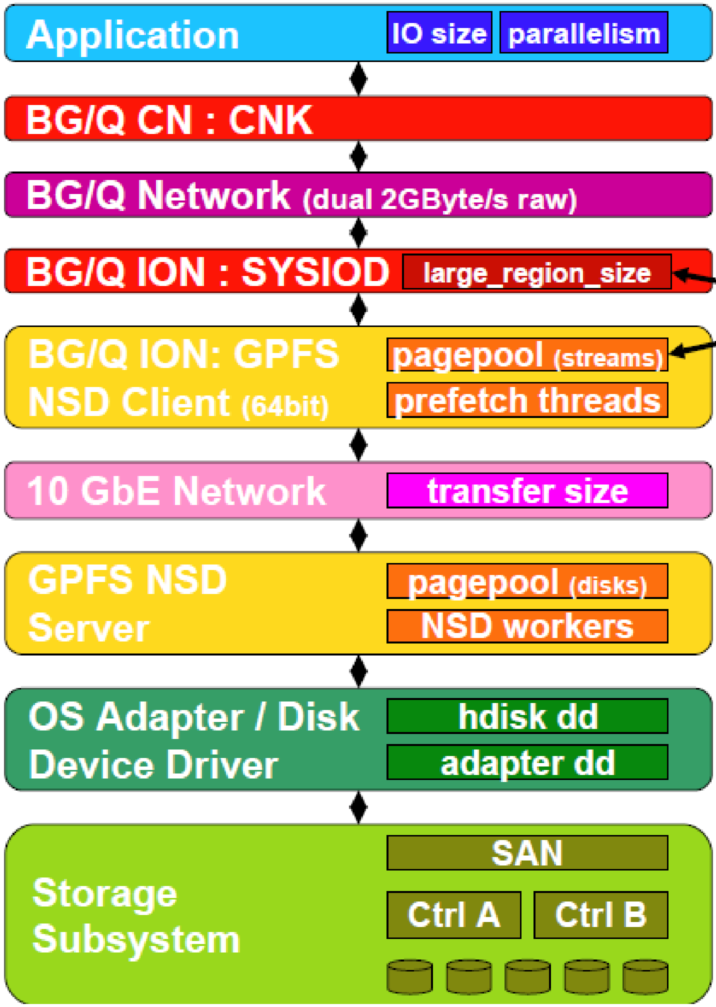
IBM General Parallel File System : Architecture and I/O Data Path





IBM General Parallel File System : Architecture and I/O Data Path on BG/Q

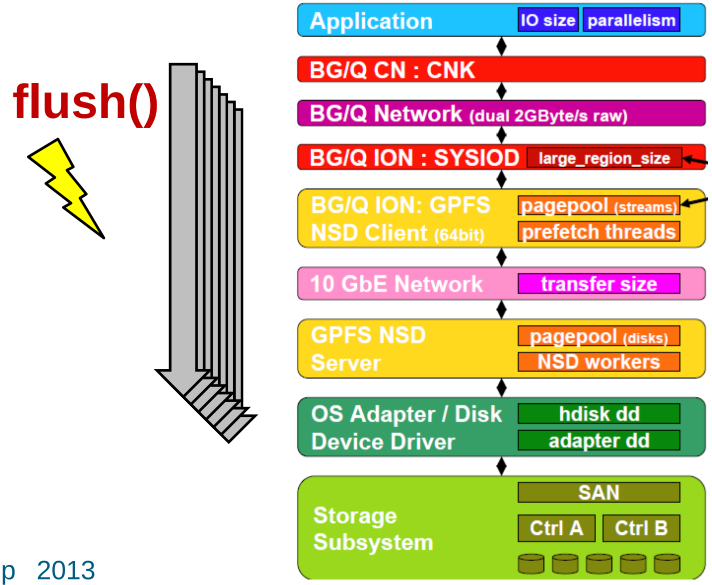




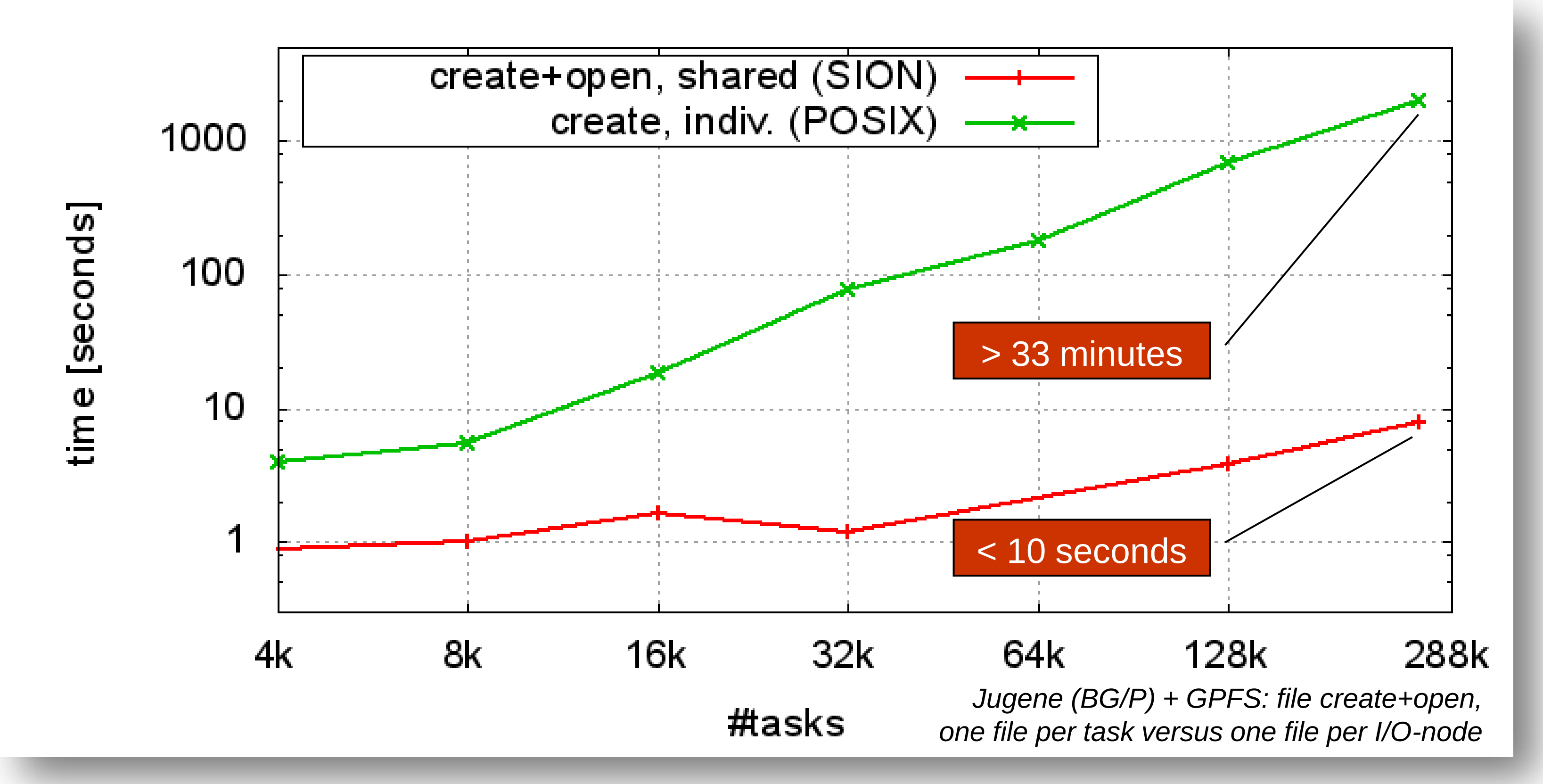
© IBM 2012

Pitfall 1: 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



Pitfall 2: Parallel Creation of Individual Files



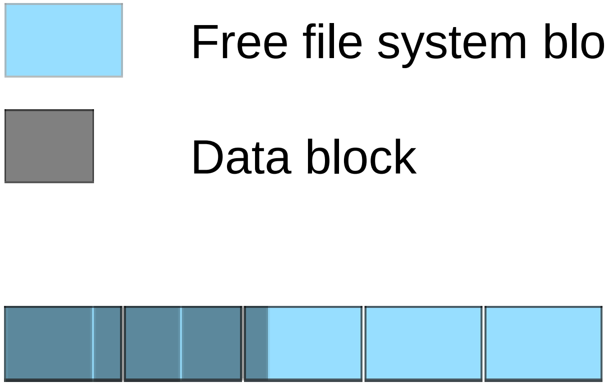
Jugene (BG/P) + GPFS: file create+open, one file per task versus one file per I/O-node

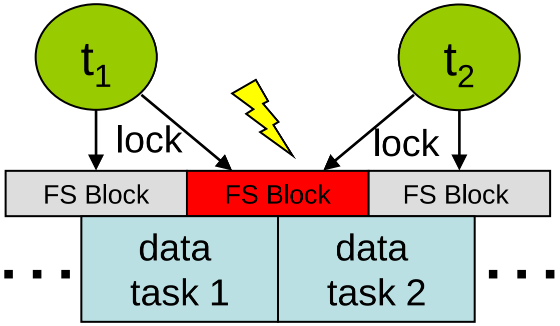
- Contention at node doing directory updates (directory meta-node)
- Pre-created files or own directory per task may help performance, but does not simplify file handling
- Complicates file management (e.g. archive)

→ **shared files are mandatory**

Pitfall 3: False sharing of file system blocks

- Parallel I/O to shared files (POSIX)



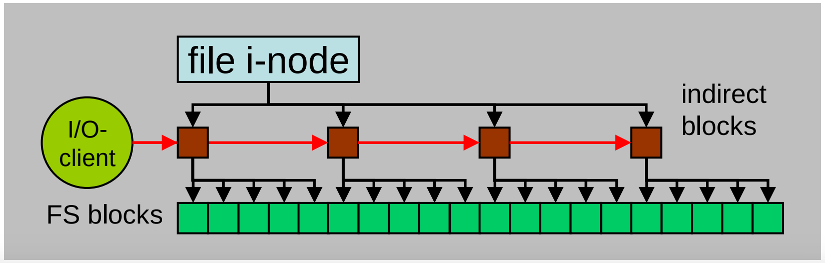


- Data blocks of individual processes do not fill up a complete file system block
- Several processes share a file system block
- Exclusive access (e.g. write) must be serialized
- The more processes have to synchronize the more waiting time will propagate

Pitfall 4: Number of Tasks per Shared File

Meta-data wall on file level

- File meta-data management
- Locking

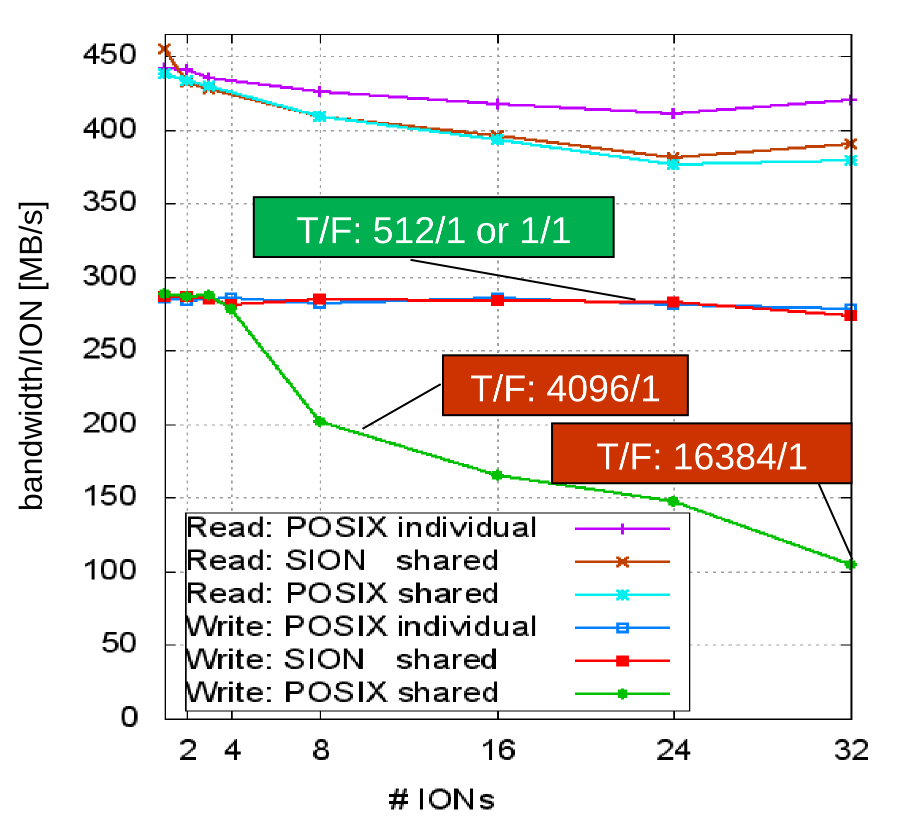


Example Blue Gene/P

- Jugene (72 racks)
- I/O forwarding nodes (ION)
- GPFS client on ION
- Solution:

  → tasks : files ratio ~ const

  → SIONlib:
  one file per ION
  implicit task-to-file mapping



Pitfall 5: Portability

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

2.712.847.316

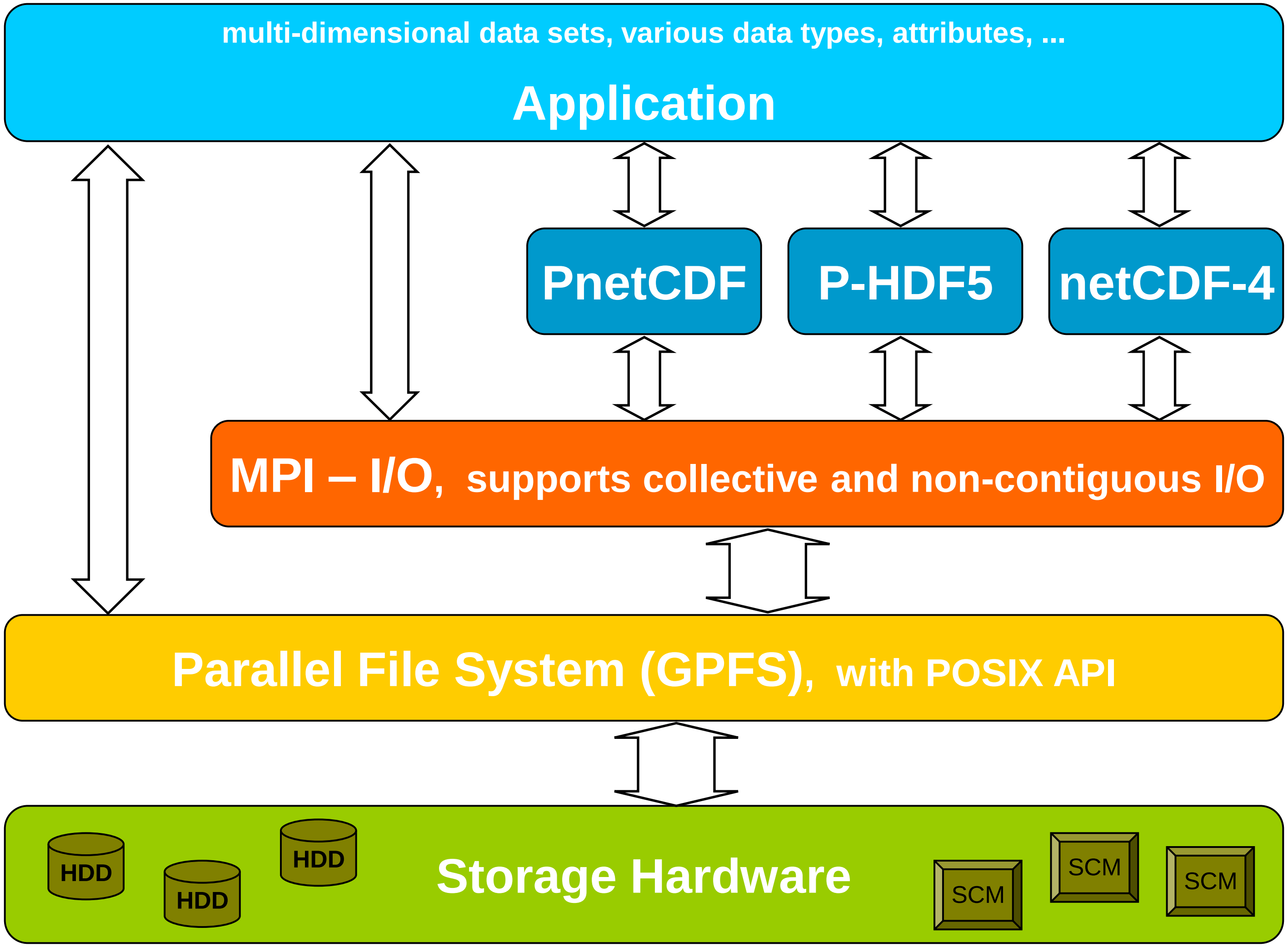
=

10100001 10110010 11000011 11010100

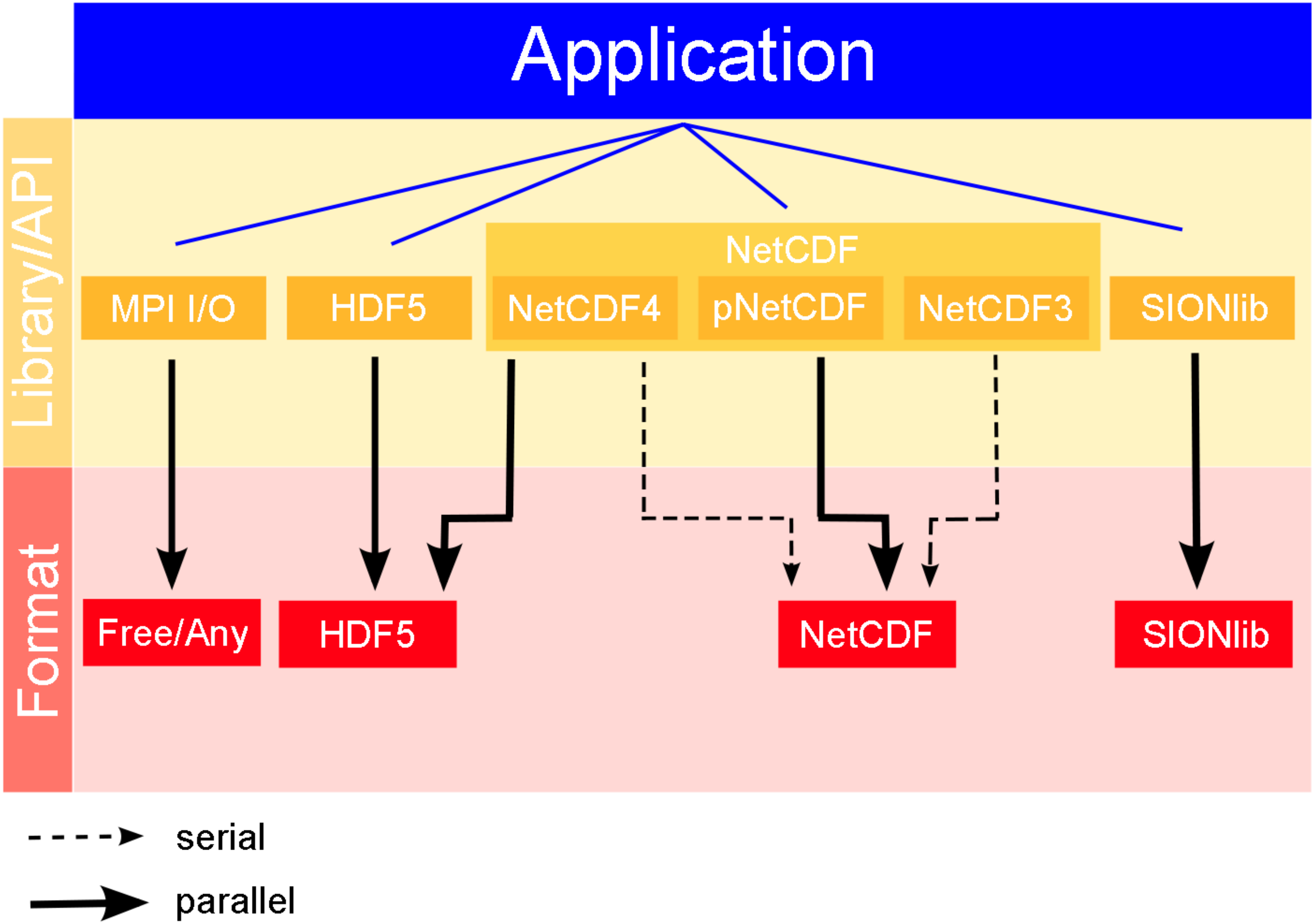
Address	Little Endian	Big Endian
1000	**11010100**	**10100001**
1001	**11000011**	**10110010**
1002	**10110010**	**11000011**
1003	**10100001**	**11010100**

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

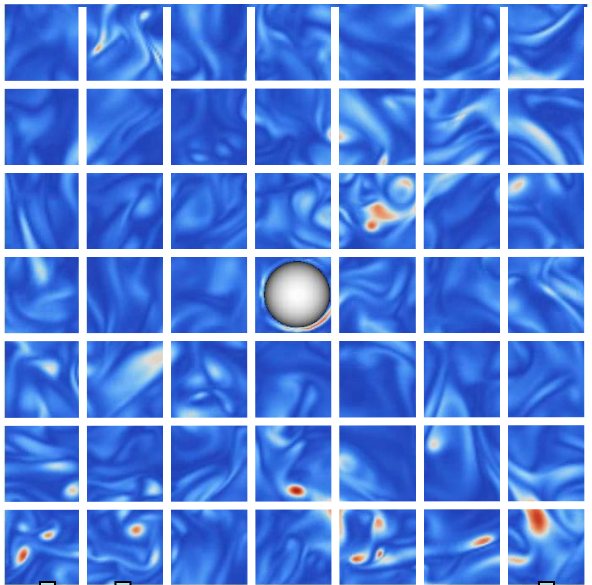
The Parallel I/O Software Stack


multi-dimensional data sets, various data types, attributes, ...
Application
PnetCDF
P-HDF5
netCDF-4
MPI – I/O, supports collective and non-contiguous I/O
Parallel File System (GPFS), with POSIX API
HDD
HDD
HDD
Storage Hardware
SCM
SCM
SCM

Paralel I/O Libraries: APIs + Formats



SIONlib: Parallel Task-local I/O

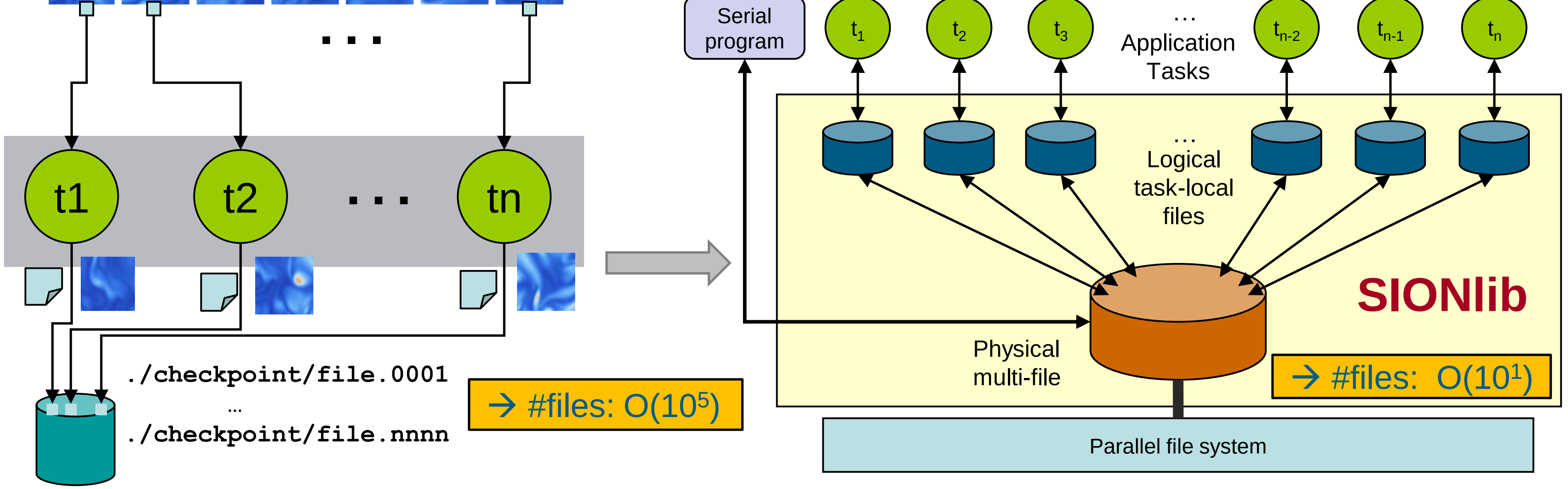


Usage Examples:

- Check-point files, restart files
- Result files, post-processing

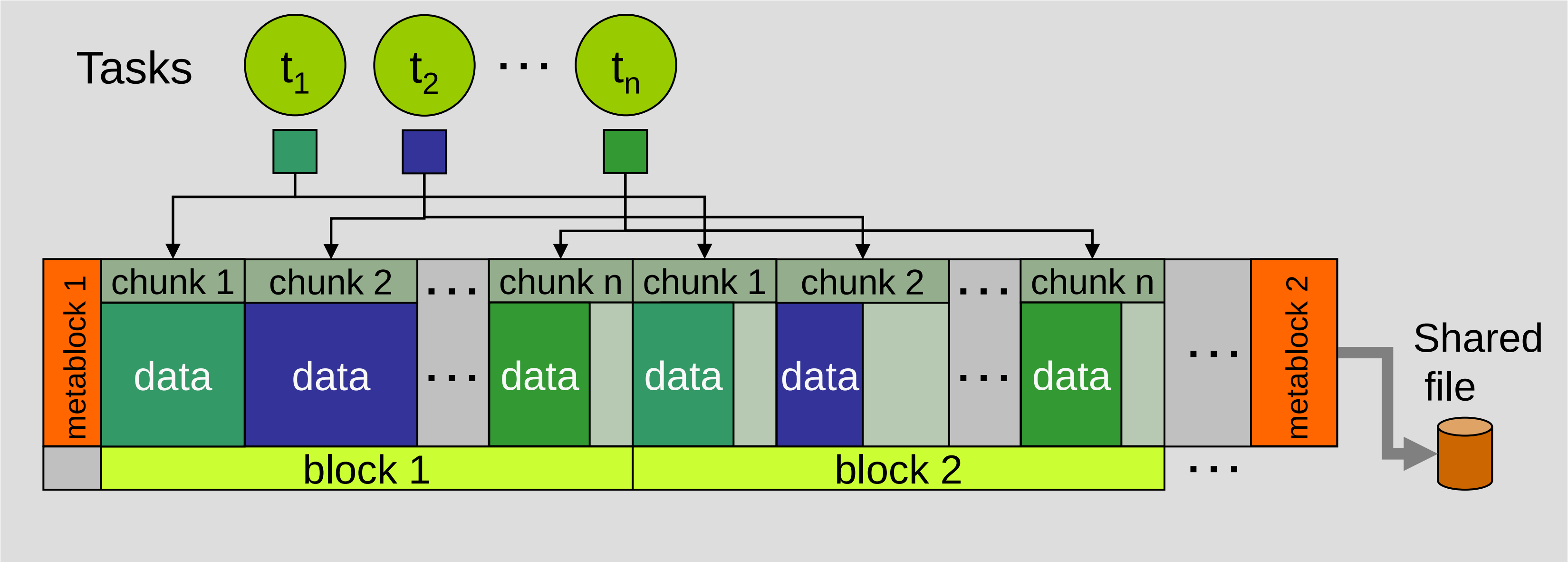
Data types:

- Simulation data (domain-decomposition)
- Performance data (e.g. trace data of parallel performance tools)

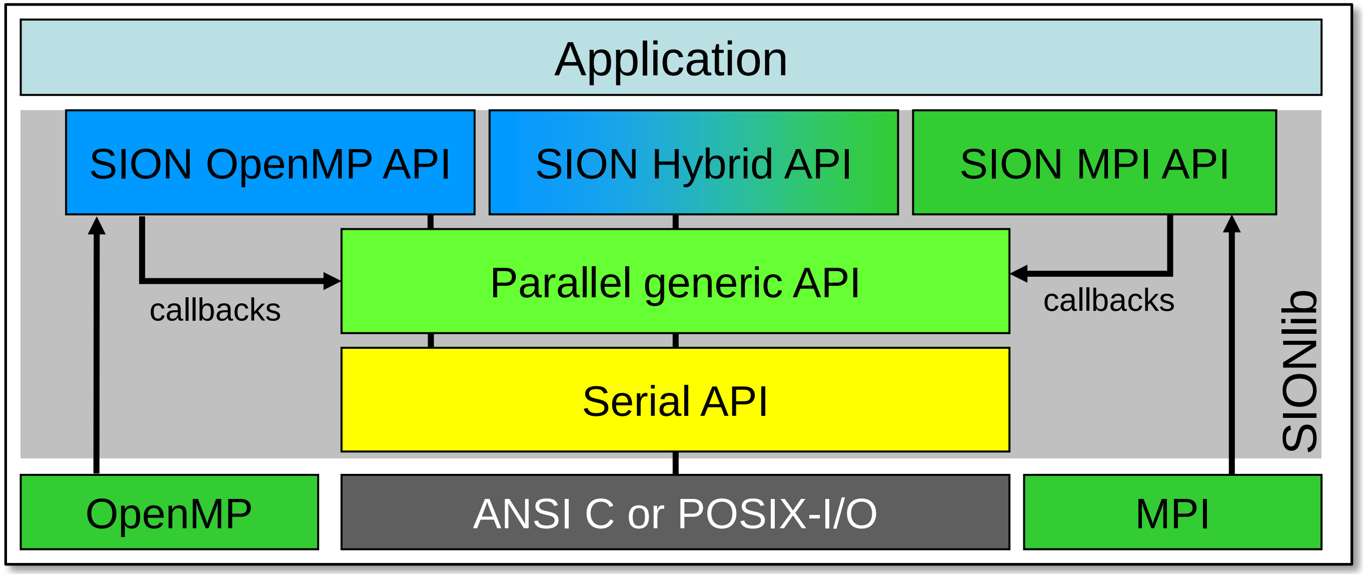


SIONlib: Overview & File Format

- Self-describing container format for task-local binary data
- Meta data handling (offset and data size, big/little endian, …)
- Multiple chunks per task
- Automatic alignment to file system blocks
- Transparent support of **multiple physical files** **(e.g. per ION)**
- File coalescing: support for collective I/O



SIONlib: Architecture & Example



- Extension of ANSI C-API
- C and Fortran bindings, implementation language C
- Current version: 1.3p7
- JUQUEEN: `module load sionlib`
- 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)
```

BGQ: Tasks connected to same I/O-node

- MPIX_Call not available on BG/Q
- Sample implementation using BG-personality and information about I/O-Bridges
- SIONlib utility function: `sion_get_IO_comm_mpi()`

```
include <firmware/include/personality.h>
int myMPIX_Pset_same_comm_create(MPI_Comm *commSame) {

 Personality_t personality;
 int           rc, rank, factor, bridgeid;

 MPI_Comm_rank(MPI_COMM_WORLD, &rank);

 /* get location information */
 Kernel_GetPersonality(&personality, sizeof(Personality_t));

 factor=1;
 bridgeid  = personality.Network_Config.cnBridge_E;        factor*= personality.Network_Config.Enodes;
 bridgeid += personality.Network_Config.cnBridge_D*factor; factor*= personality.Network_Config.Dnodes;
 bridgeid += personality.Network_Config.cnBridge_C*factor; factor*= personality.Network_Config.Cnodes;
 bridgeid += personality.Network_Config.cnBridge_B*factor; factor*= personality.Network_Config.Bnodes;
 bridgeid += personality.Network_Config.cnBridge_A*factor;

    /* communicator consists of all task working with the same I/O-node */
    rc=MPI_Comm_split(MPI_COMM_WORLD, bridgeid, rank, commSame);
    return(rc);
}
```

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, …*

Summary

- Application I/O has to exploit **parallelism** to make use of the full available bandwidth of HPC I/O systems
- Fast internal I/O with special data formats and I/O libraries
- Portable data formats are needed to efficiently process data in heterogeneous environments
- Multiple solutions to portable parallel I/O are available

- Training Course: **Parallel I/O and Portable Data Formats**

  18 March to 20 March 2013

  http://www.fz-juelich.de/SharedDocs/Termine/IAS/JSC/DE/Kurse/parallelio-2013.html