Analyzing the High Performance Parallel I/O on LRZ HPC systems

Sandra Méndez.
HPC Group, LRZ.
June 23, 2016

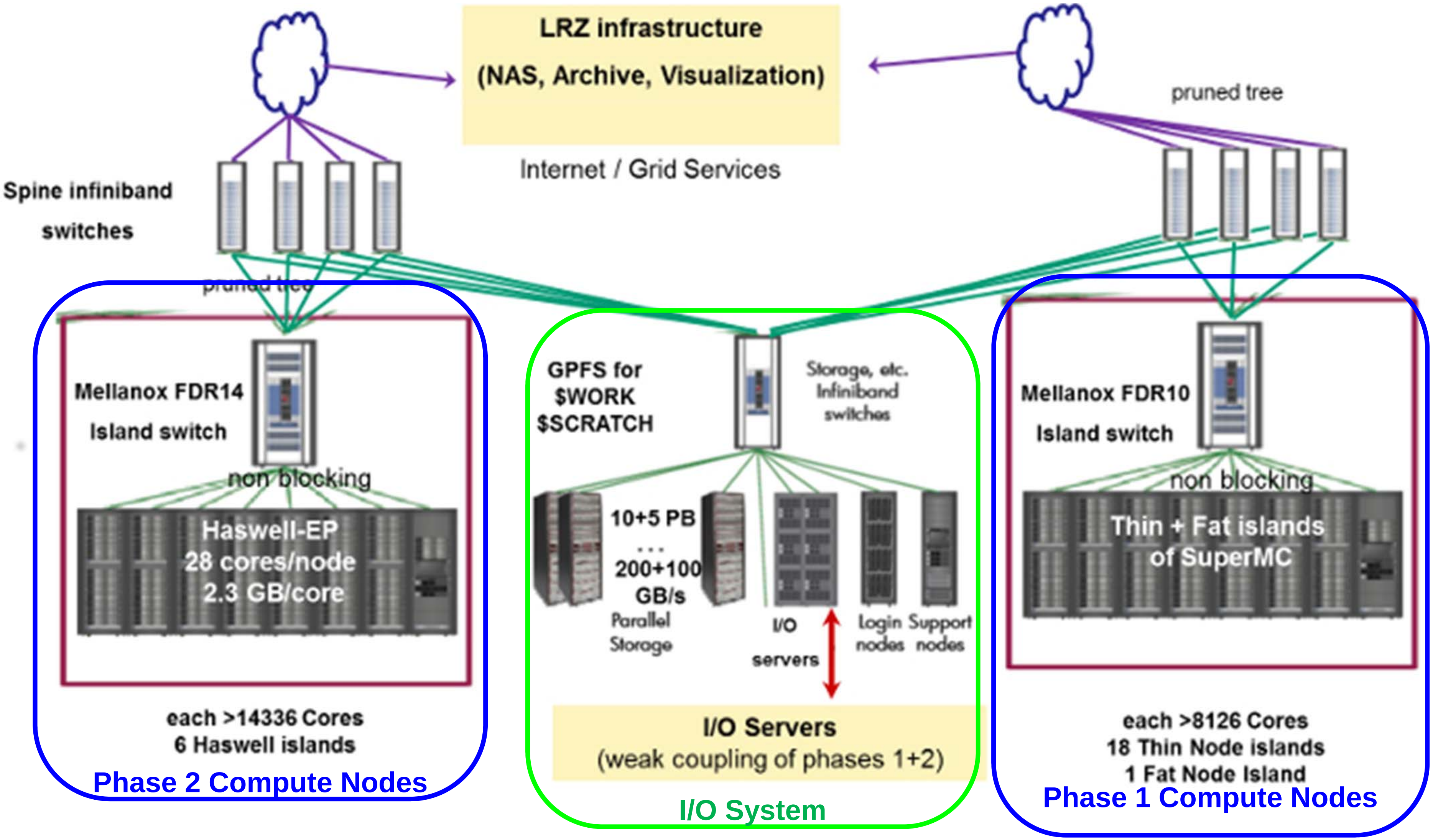
Outline

- SuperMUC supercomputer
- User Projects
- Monitoring Tool
- I/O Software Stack
- I/O Analysis Tool
- Analyzing I/O Problems
- Conclusions

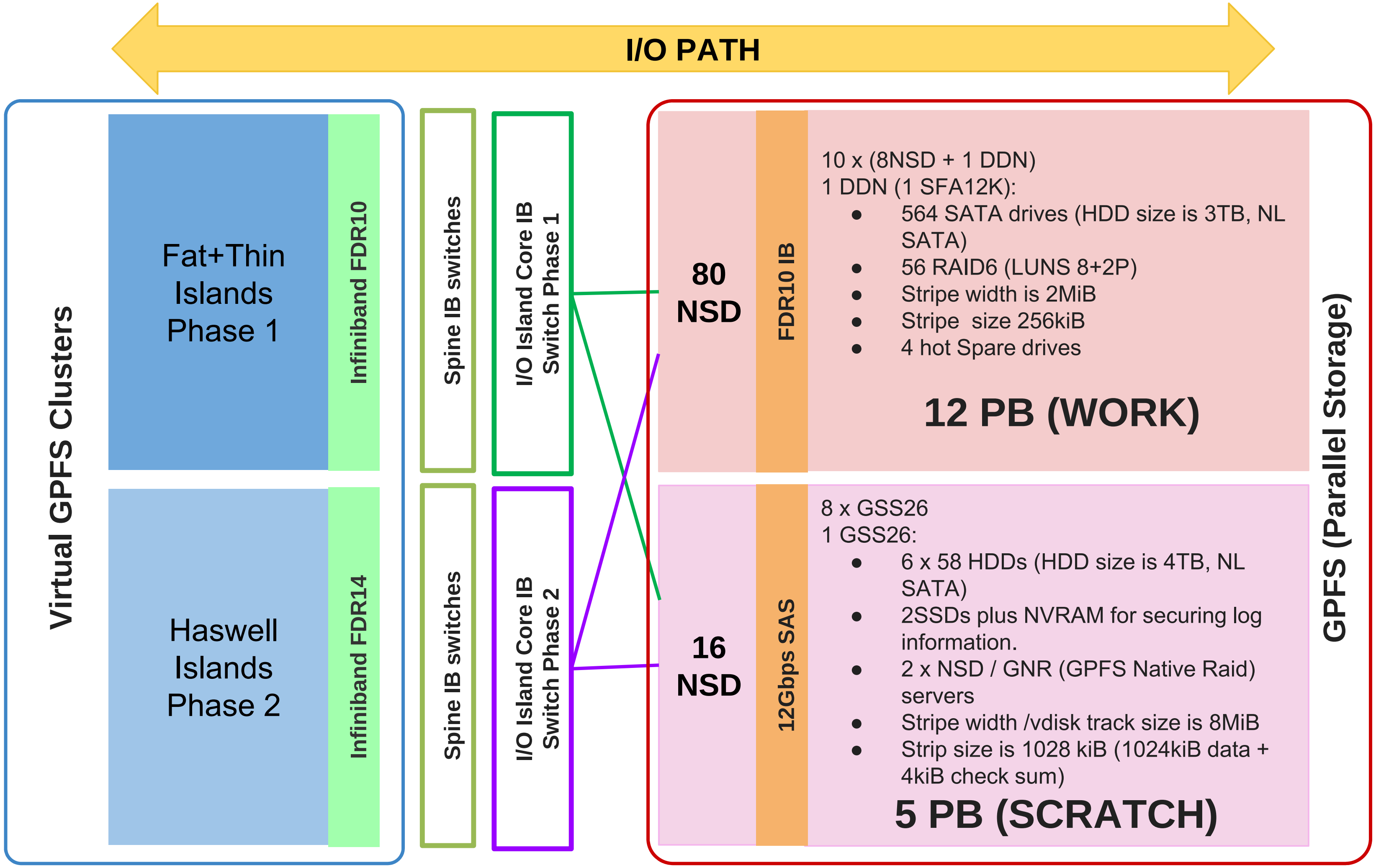
SuperMUC supercomputer - LRZ

- Member of the Gauss Centre for Supercomputing (GCS). Tier-0 centre for PRACE, the Partnership for Advanced Computing in Europe.
- 2012 SuperMUC Phase 1 and 2015 SuperMUC Phase 2. Total Peak Performance 6.4 PFlop/s.

SuperMUC Supercomputer



The I/O PATH on SuperMUC - Parallel Storage (WORK and SCRATCH filespace)

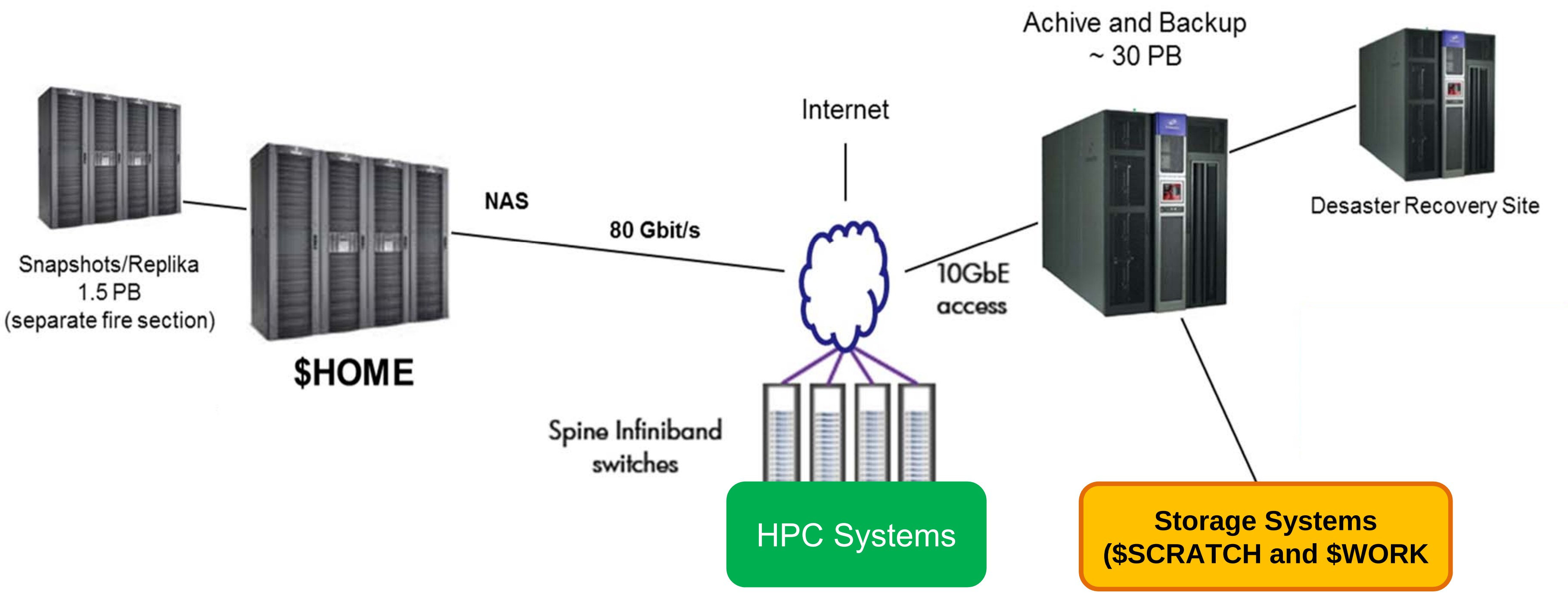


The Global Filesystem HOME (NFS)

- **Available on all HPC cluster systems (environment variable `$HOME`)**
- **Shared area for all user accounts in a project**

Very reliable

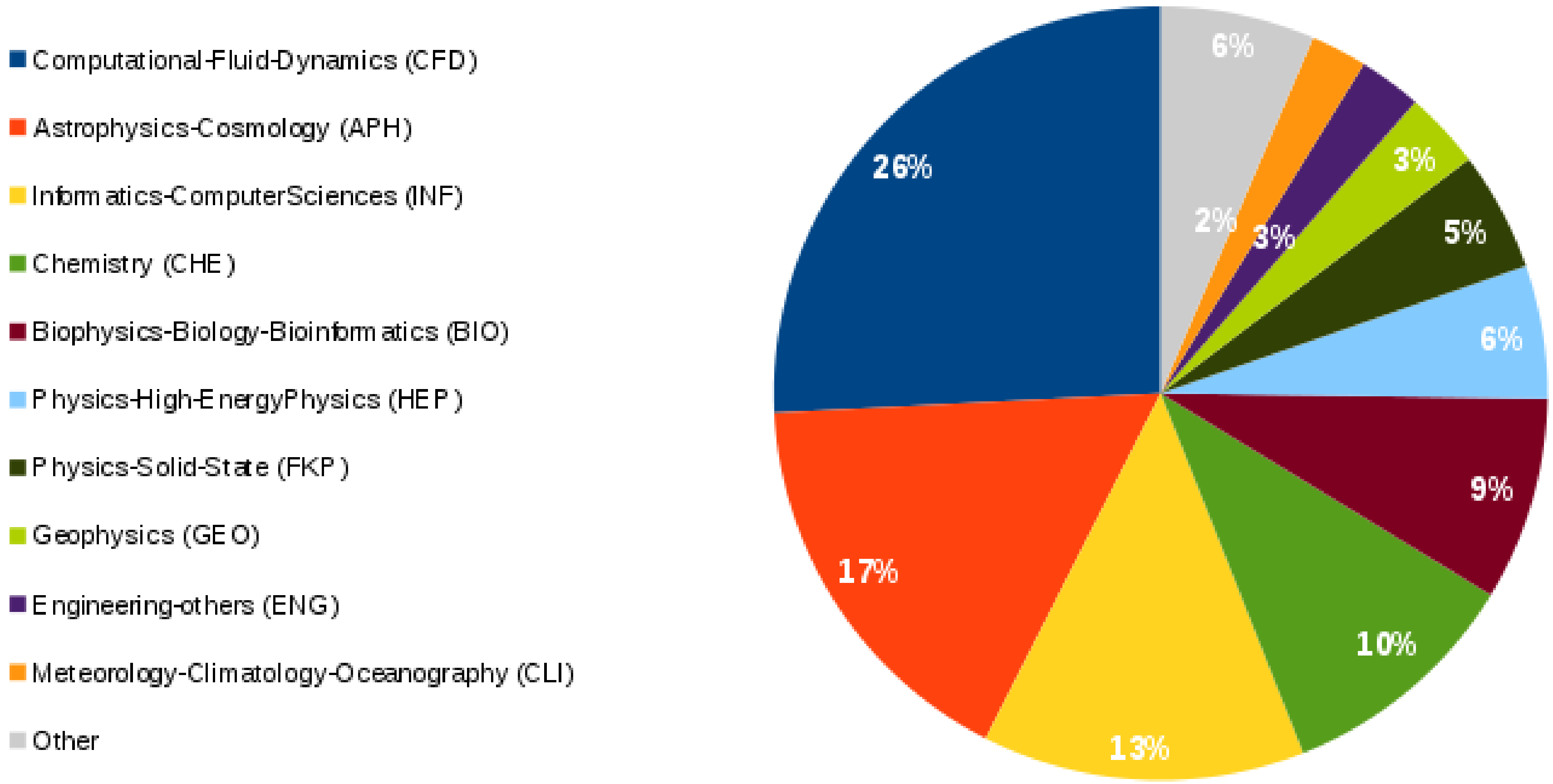
- user-restorable snapshots (last 10 days)
- automatic data protection by LRZ



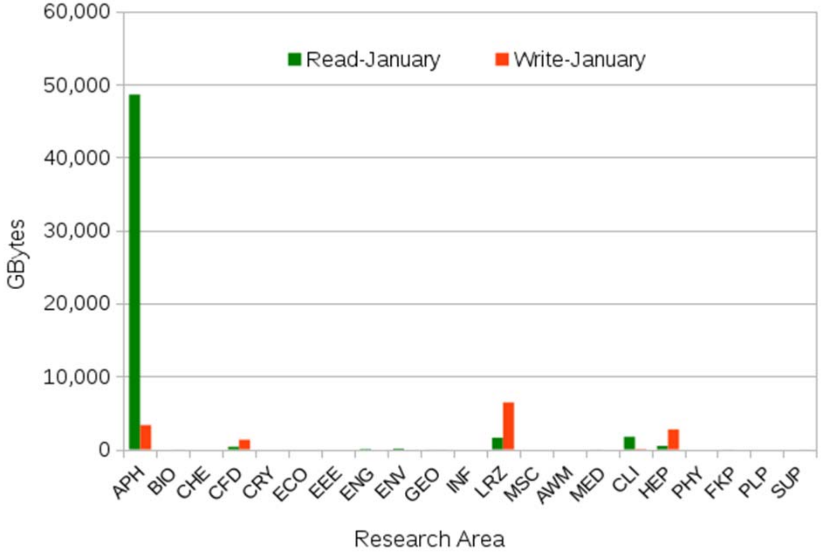
Outline

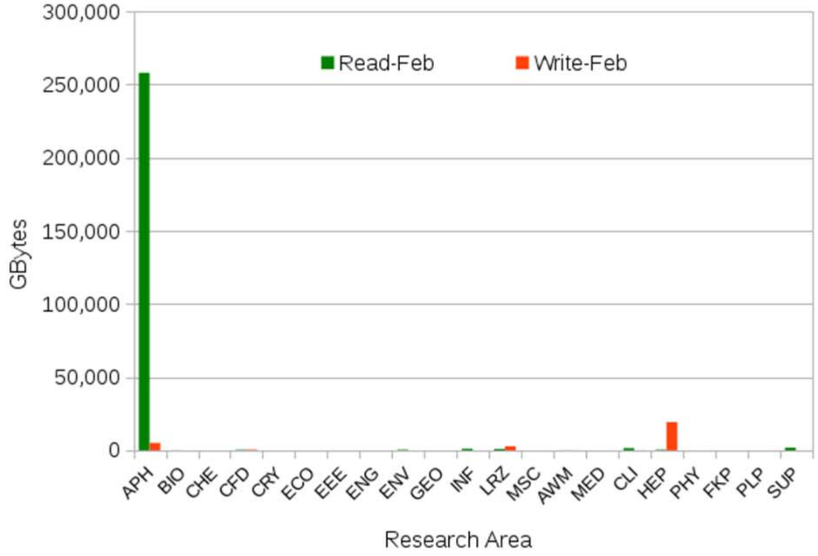
- SuperMUC supercomputer
- **User Projects**
- Monitoring Tool
- I/O Software Stack
- I/O Analysis Tool
- Analyzing I/O Problems
- Conclusions

Projects by Research Area

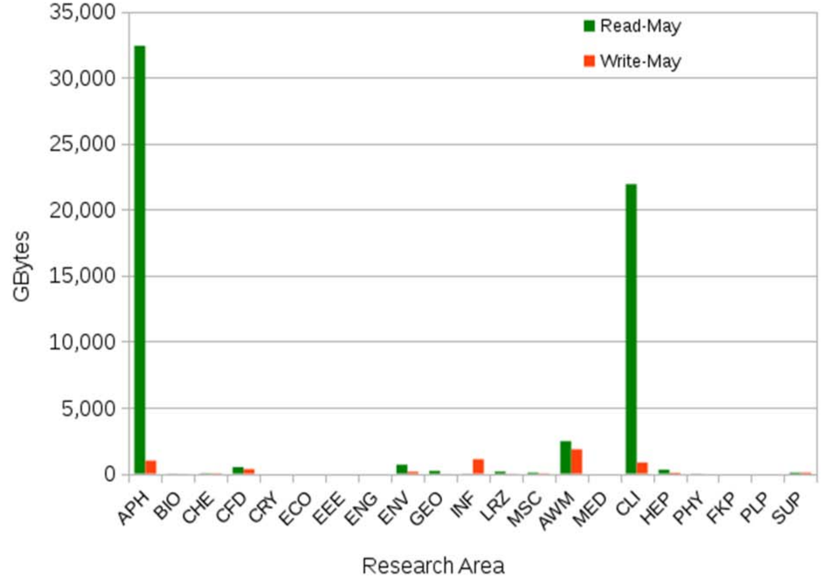


Data Transferred by Research Area



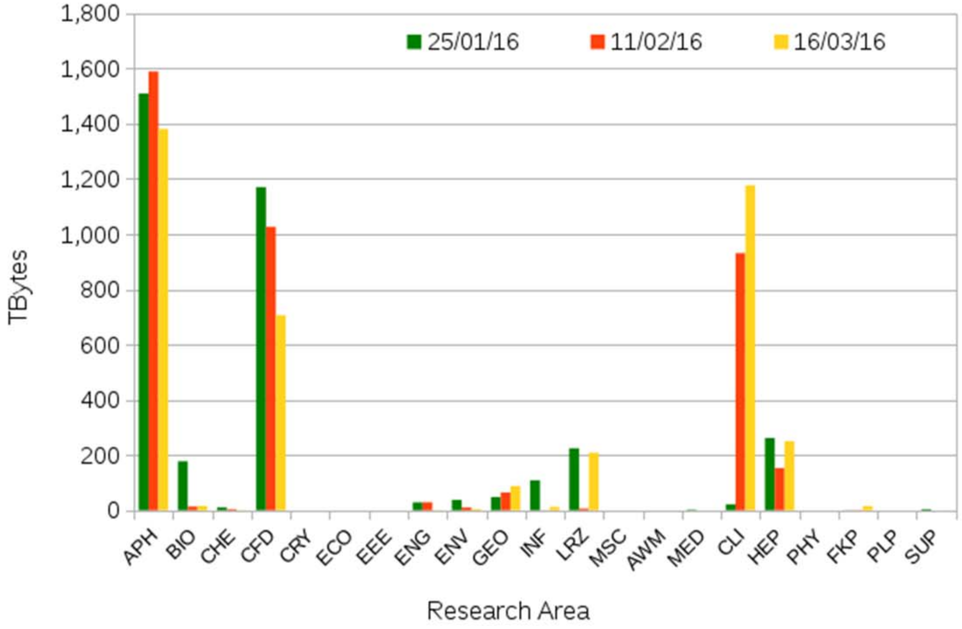


APH	Astrophysics-Cosmology
BIO	Biophysics-Biology-Bioinformatics
CHE	Chemistry
CFD	Computational-Fluid-Dynamics
CRY	Crystallography
ECO	Economics
EEE	Engineering-ElectricalEngineering
ENG	Engineering-others
ENV	Enviromental-Sciences
GEO	Geophysics
INF	Informatics-ComputerSciences
LRZ	LRZ
MSC	Material-Science
AWM	Mathematics
MED	Medicine
CLI	Meteorology-Climatology-Oceanography
HEP	Physics-High-EnergyPhysics
PHY	Physics-others
FKP	Physics-Solid-State
PLP	Plasma-Physics
SUP	Support-Benchmarking

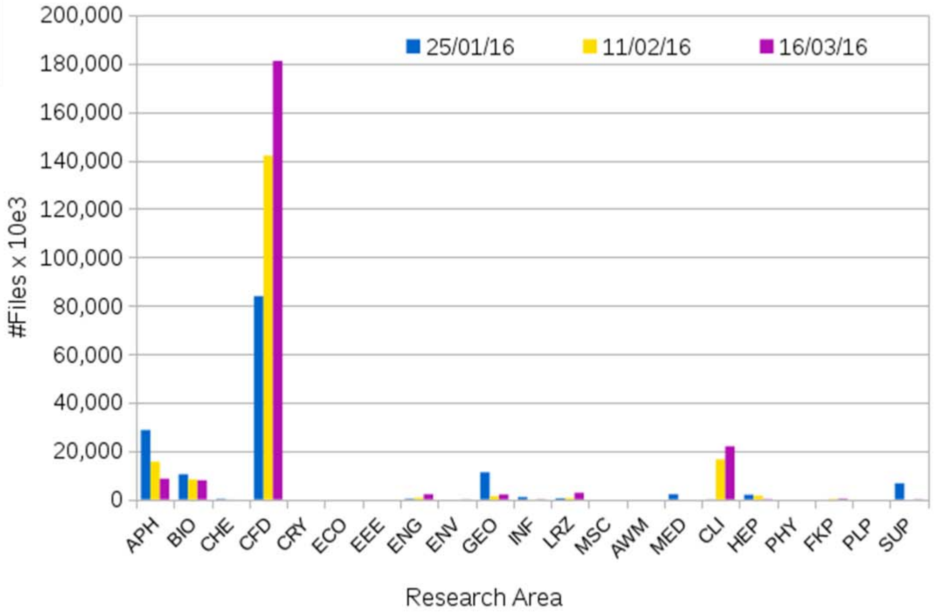


Storage and Files by Research Area – SCRATCH filespace

Storage



Count of Files



I/O Requirements

I/O Libraries

- HDF5 15%, NetCDF or PnetCDF 10%; POSIX, MPI-IO, or an I/O library locally installed 75%.

Storage Parallel

- WORK (70% Capacity) -> 5 fold increase
- SCRATCH (80% Capacity) -> 8 fold increase

Checkpointing and large scale output with a connection to a visualization cluster.
Checkpointing (for the Large-Scale Projects):
Periods: 5 min to 8 hours
Size: 100 GB -> 38%
1TB -> 10%
5TB -> 7%
10TB -> 1%
35TB -> 2%
70TB -> 1%
< 100GB -> 41%

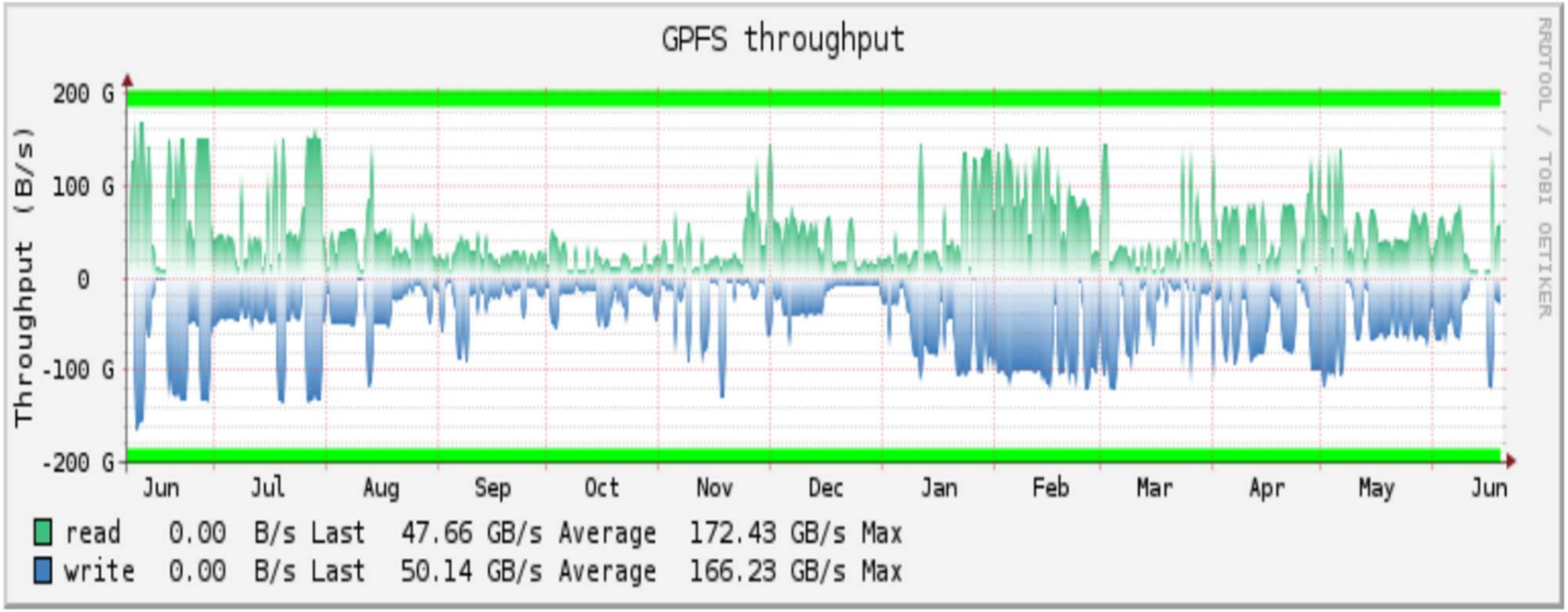
Outline

- SuperMUC supercomputer
- User Projects
- **Monitoring Tool**
- I/O Software Stack
- I/O Analysis Tool
- Analyzing I/O Problems
- Conclusions

GPFS Monitor on SuperMUC: SCRATCH

One Year (06.06.15 19:01 - 20.06.16 19:01)

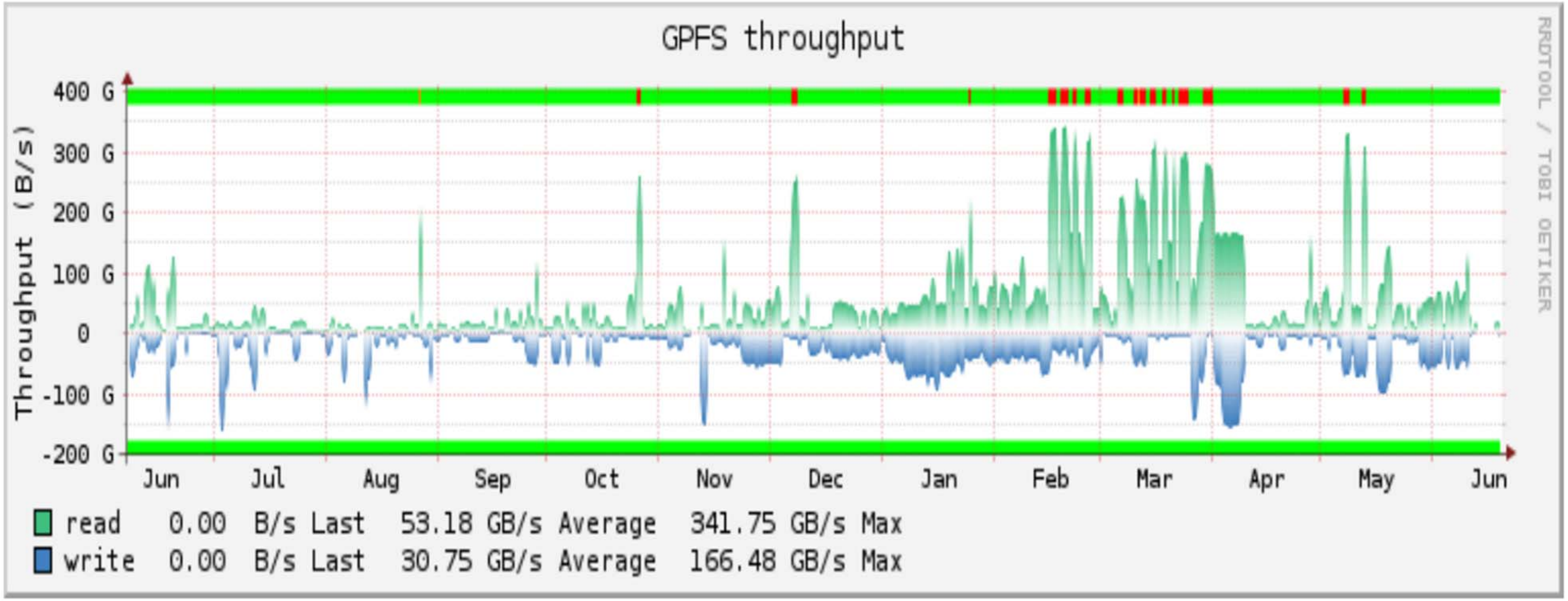
Datasource Throughput



GPFS Monitor on SuperMUC: WORK

One Year (06.06.15 19:00 - 20.06.16 19:00)

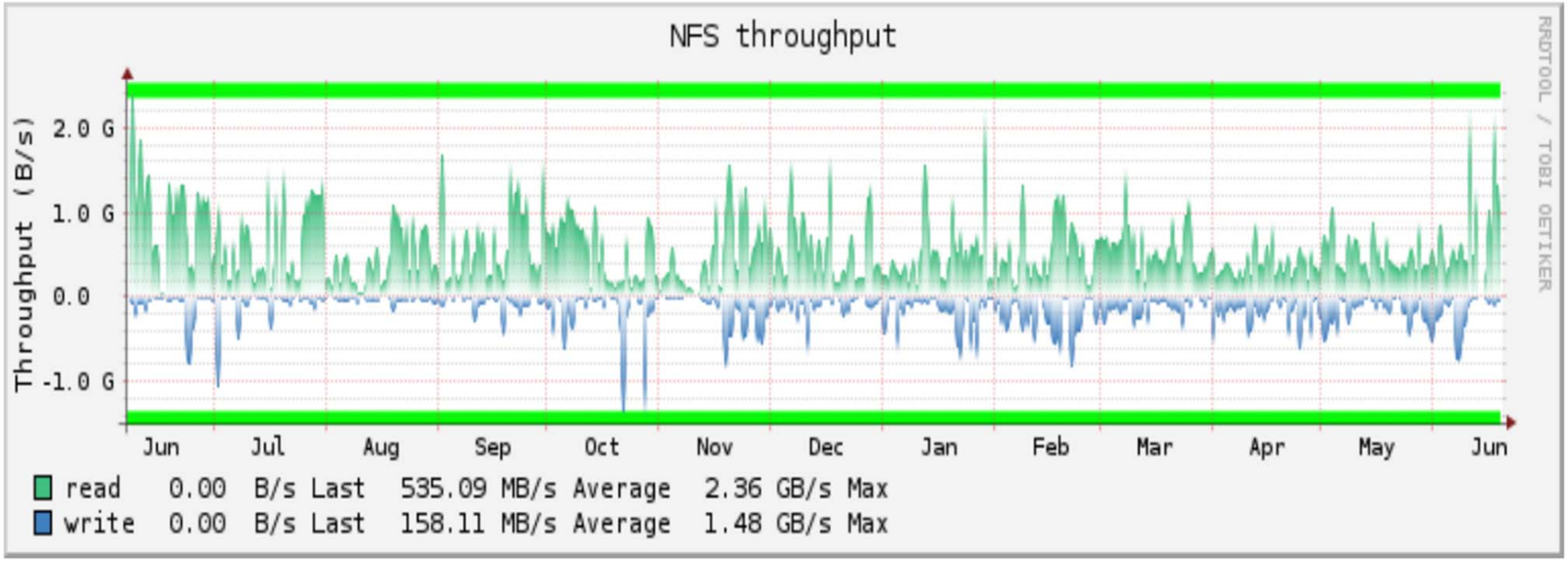
Datasource Throughput



NFS Monitor on SuperMUC: HOME

One Year (06.06.15 18:59 - 20.06.16 18:59)

Datasource Throughput

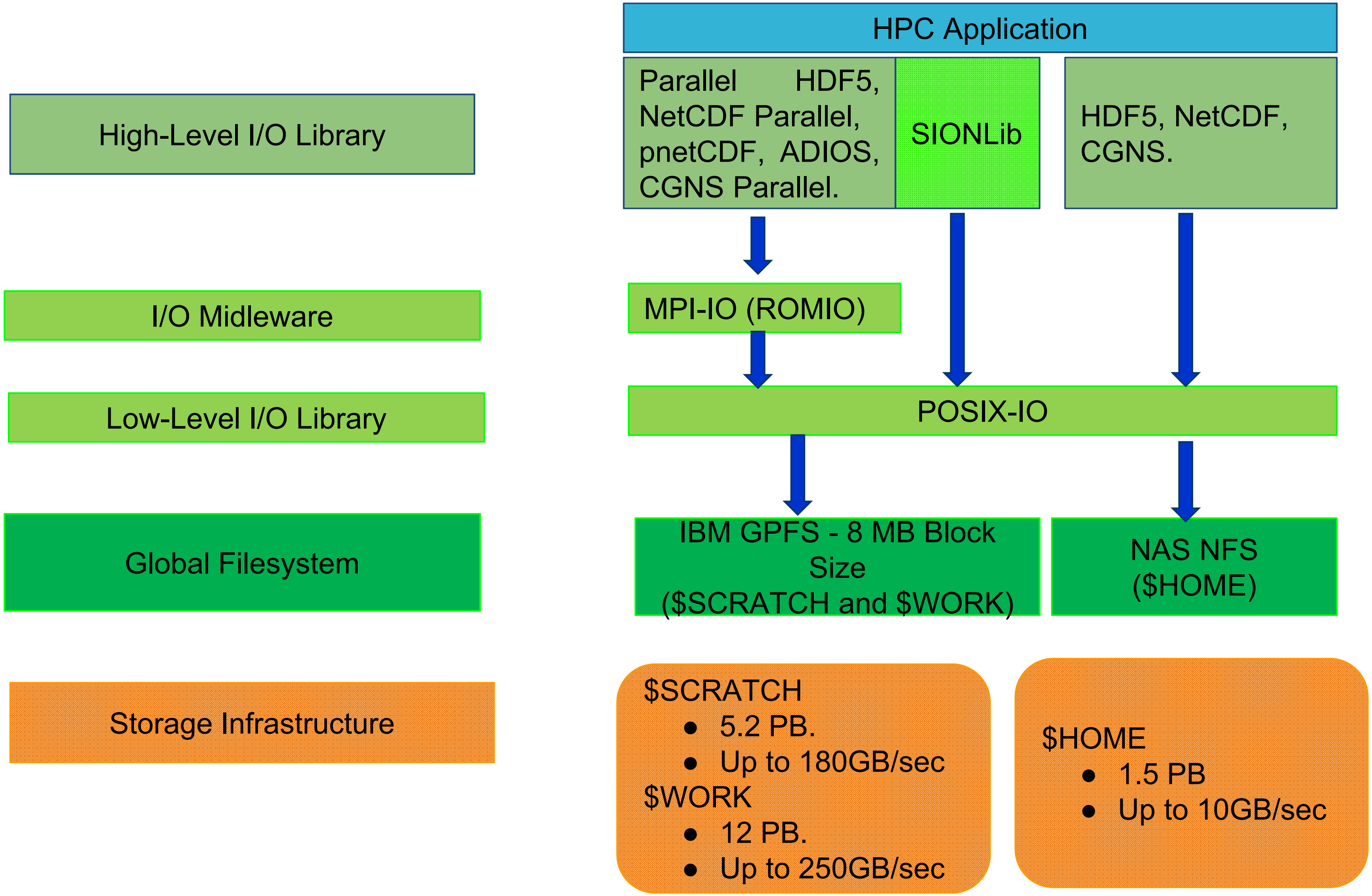


Persyst Tool

Outline

- SuperMUC supercomputer
- User Projects
- Monitoring Tool
- **I/O Software Stack**
- I/O Analysis Tool
- Analyzing I/O Problems
- Conclusions

The I/O Software Stack on SuperMUC



MPI-IO Libraries at LRZ

- **IBM Parallel Environment** (MPI-IO implementation a ROMIO version)
- **Intel MPI** (MPI-IO implementation a ROMIO version)
  - ➢Set the `I_MPI_EXTRA_FILESYSTEM` environment variable to on to enable parallel file system support.
  - ➢Set the `I_MPI_EXTRA_FILESYSTEM_LIST` environment variable to request native support for the specific file system.

ROMIO Hints

Data Sieving:

- `ind_rd_buffer_size`
- `ind_wr_buffer_size`
- `romio_ds_read`
- `romio_ds_write`

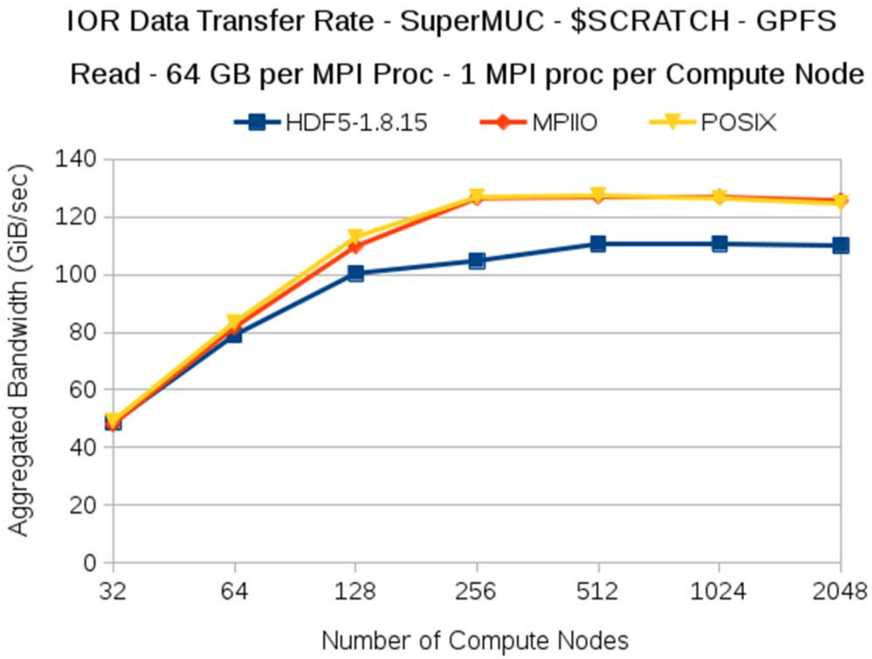
Collective buffering (Two-Phase I/O)

- `cb_buffer_size`
- `romio_cb_read`
- `romio_cb_write`

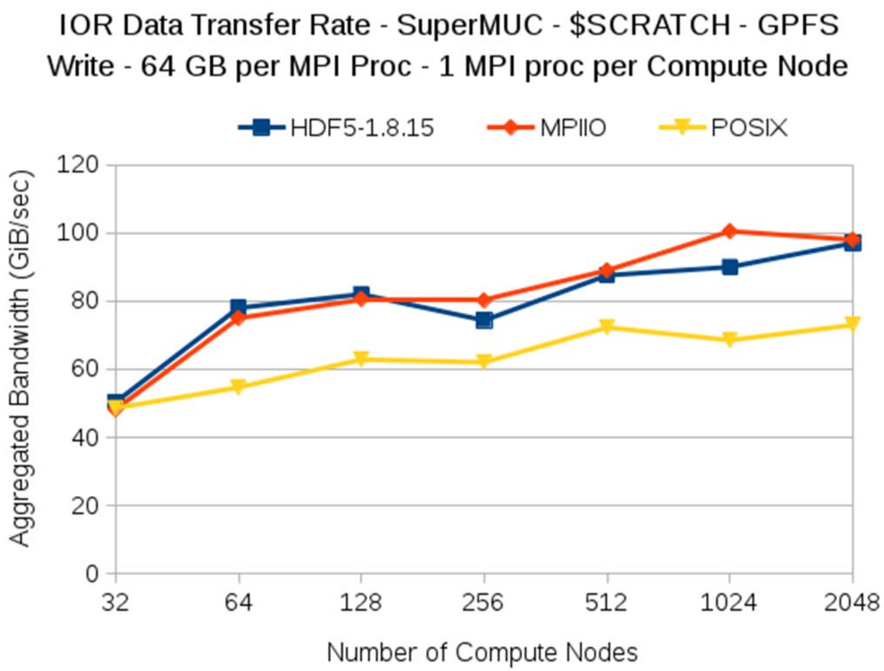
Evaluation of MPI-IO Hints in SuperMUC

Exp.	MPI Processes	Compute Nodes	Access Pattern	Request Size	Hints	Transfer Rate(GiB/sec)
1	512	32	Sequential	1 MiB	romio_cb_read = automatic romio_cb_write = automatic	write = 25.92 read = 23.80
					romio_cb_read = disable romio_cb_write = disable	write = 75.34 read = 67.58
					independent I/O	**write = 80.39** **read = 69.62**
2	512	32	Strided	1 MiB	romio_cb_read = automatic romio_cb_write = automatic	write = 1.63 read = 17.74
					romio_cb_read = enable romio_cb_write = enable	**write = 25.49** **read = 26.10**
					independent I/O	write = 5.15 read = 12.60
3	512	32	Sequential	256 MiB	romio_cb_read = automatic romio_cb_write = automatic	write = 74.48 read = 46.67
					romio_cb_read = disable romio_cb_write = disable	**write = 83.37** **read = 65.88**
					independent I/O	write = 82.29 read = 64.22
4	512	32	Strided	256 MiB	romio_cb_read = automatic romio_cb_write = automatic	write = 71.12 read = 41.80
					romio_cb_read = disable romio_cb_write = disable	**write = 77.22** **read = 70.21**
					romio_cb_read = enable romio_cb_write = enable	write = 24.81 read = 24.90
					independent I/O	write = 71.25 read = 67.71

GPFS Scalability on SuperMUC: Weak scaling up to 4 Islands on SCRATCH



Islands of Thin Nodes.
Peak Performance 130 GB/sec
(From Phase 1 to SCRATCH).



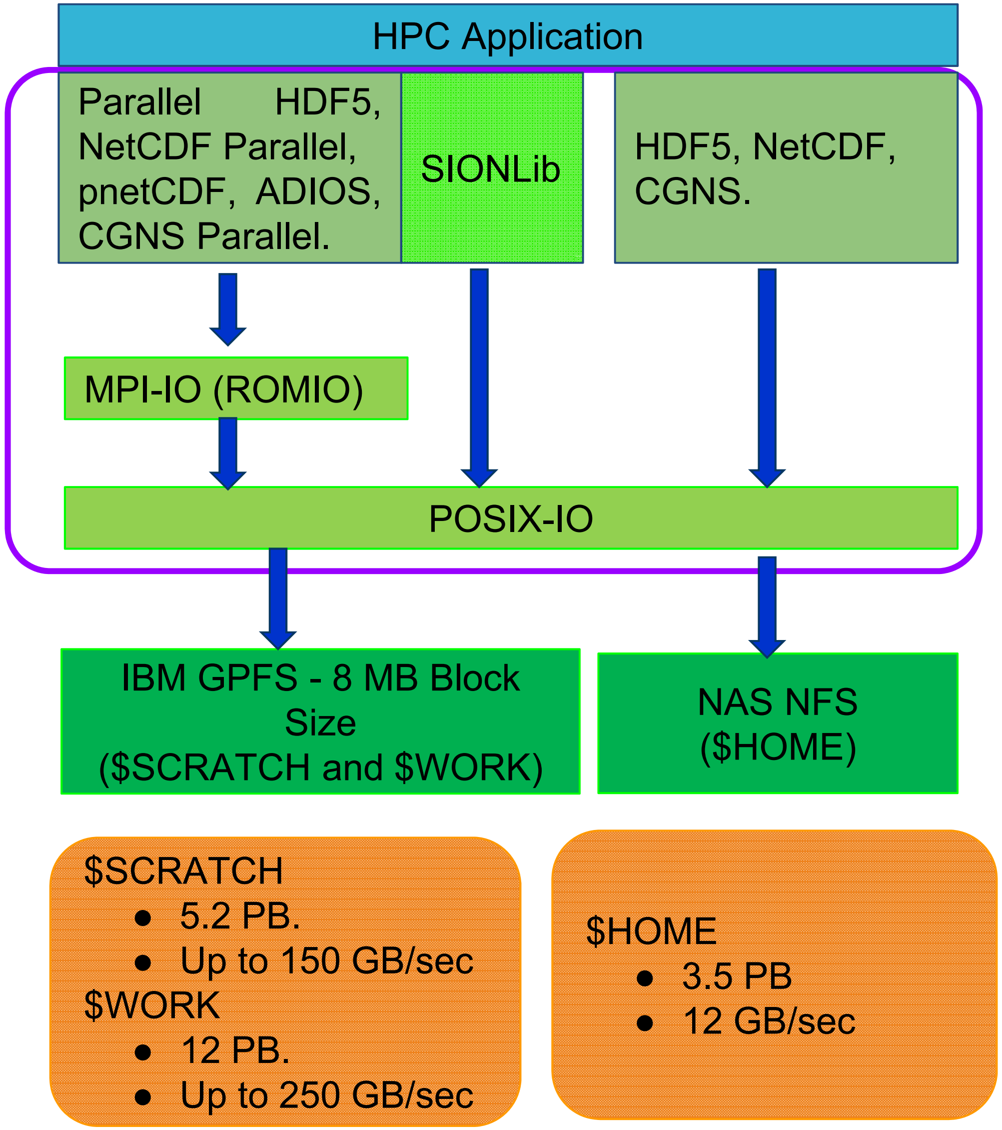
Outline

- SuperMUC supercomputer
- User Projects
- Monitoring Tool
- I/O Software Stack
- **I/O Analysis Tool**
- Analyzing I/O Problems
- Conclusions

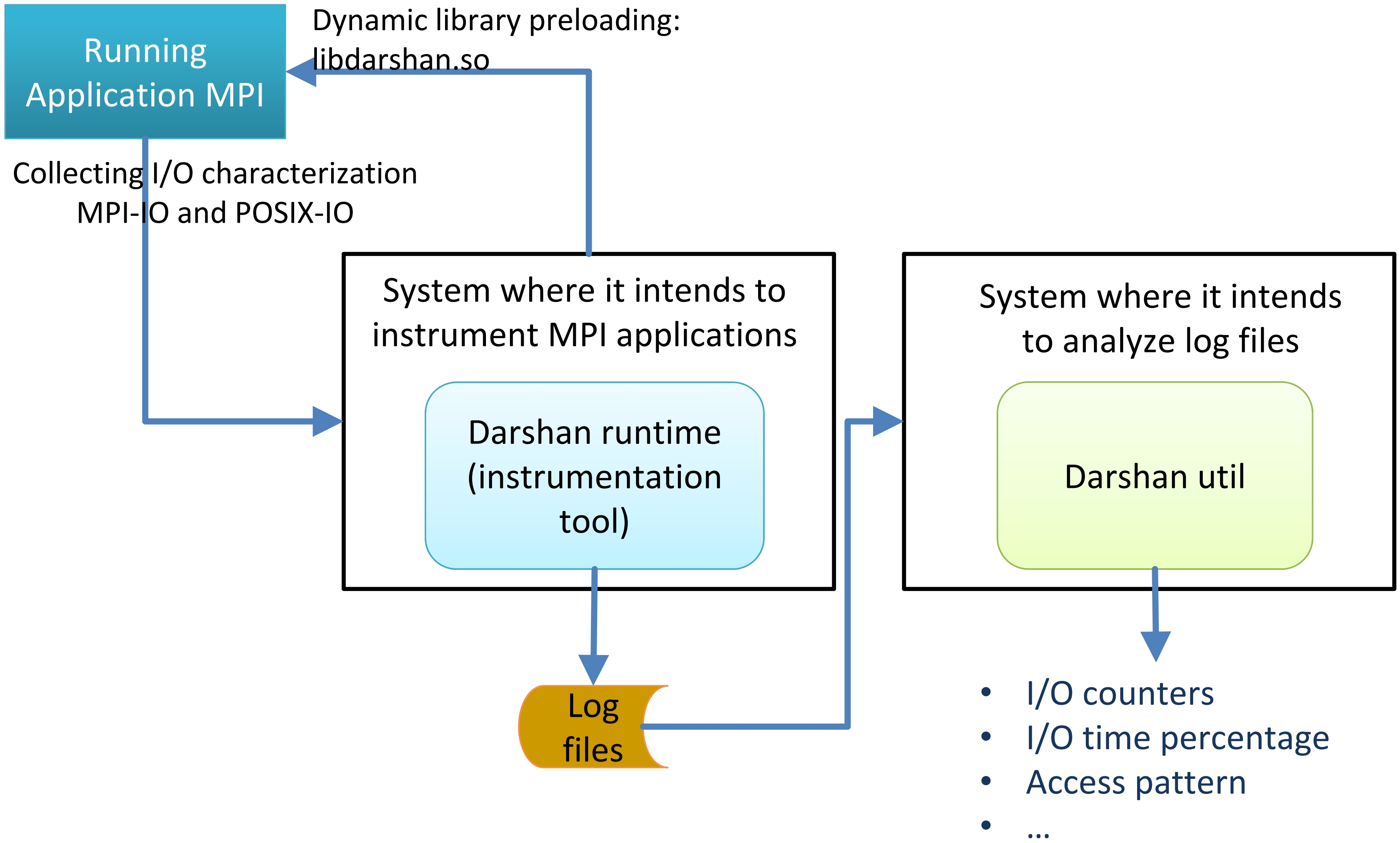
I/O Profiling

I/O Profiling:

- **DARSHAN (POSIX-IO, MPI-IO)**



Darshan Components



How to use Darshan in SuperMUC?

- **To make use of Darshan in its version 2.3 and 3.0, the module appropriate must be loaded.**

```
module load darshan
```

- **Set up the variable FORTRAN_PROG in “true” if the program is a Fortran program and false if it's not.**

```
FORTRAN_PROG=true
```

- **Load the appropriate library.**

```
export LD_PRELOAD=`darshan-user.sh $FORTRAN_PROG`
```

- **Set up Darshan job identifier with loadleveler job identifier.**

```
export JOBID_LL=`darshan-JOBID.sh $LOADL_STEP_ID`
```

- **Set up environment variable DARSHAN_JOBID to environment variable name that contain the job identifier of loadleveler.**

```
export DARSHAN_JOBID=JOBID_LL
```

- **Set up Darshan log path**

```
export LOGPATH_DARSHAN_LRZ=`darshan-logpath.sh`
```

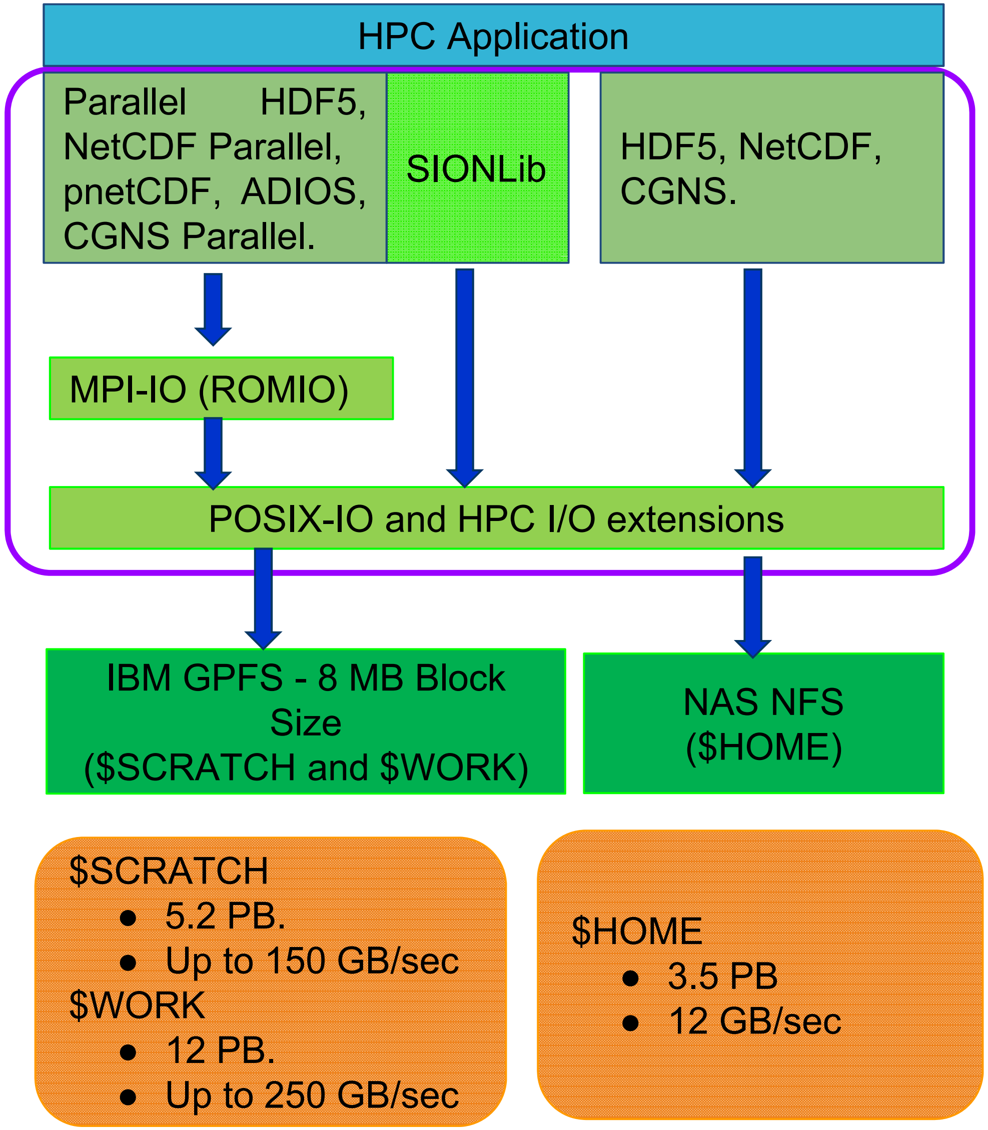
- **Darshan - Splunk format**

```
darshan-splunk.sh $JOBID_LL $LOGPATH_DARSHAN_LRZ $LOADL_STEP_ID
```

Performance Analysis Tools that support I/O analysis

Performance Analysis Tools:

- **VAMPIRTrace (POSIX-IO, MPI-IO)**
- **Scalasca (MPI-IO)**



Outline

- SuperMUC supercomputer
- User Projects
- Monitoring Tool
- I/O Software Stack
- I/O Analysis Tool
- **Analyzing I/O Problems**
- Conclusions

Common I/O Problems on SuperMUC

1. I/O Patterns
   - Time-out error
   - Slow I/O performance
2. MPI-IO Hints configuration
   - Slow I/O performance for Collective operations
3. I/O Scalability
   - Requirement for large scale parallel applications

Methodology of Analysis

Parallel Application

1 or 2

Identification of the Problems source

3

I/O Analysis

Providing a solution or recommendation

- I/O Time percentage (Darshan I/O Tool)
- Data transfer rate (Darshan I/O Tool)

Not related with the I/O

Upper than 10%

- I/O Pattern (Darshan and PAS2P-IO)
- I/O Time per Function (Tracing tool: Scalasca, Vampir, etc.)
- Review of source code of the I/O critical regions.

- Modify the I/O Pattern
- Change I/O Library parameters
- Providing the bounds of the I/O scalability.
- Provide a number of I/O processes convenient to avoid I/O degradation for scalability test

Example: I/O Pattern

Experimental Environment:

- **An Island of thin nodes (512 compute nodes)**
- **2 processors per compute node and 8 cores per processor.**
- **Size of shared memory per node is 32 GBytes (2 GBytes per core).**
- **$WORK filespace (GPFS version 3.5). Up to 180 GB/s.**
- **Block size equal to 8,388,608 bytes (8MiB) and a minimum fragment size of 262,144 bytes.**
- **IBM MPI 1.3, NetCDF 4.3.3, Scalasca 2.2.2 and Darshan 2.3.1.**

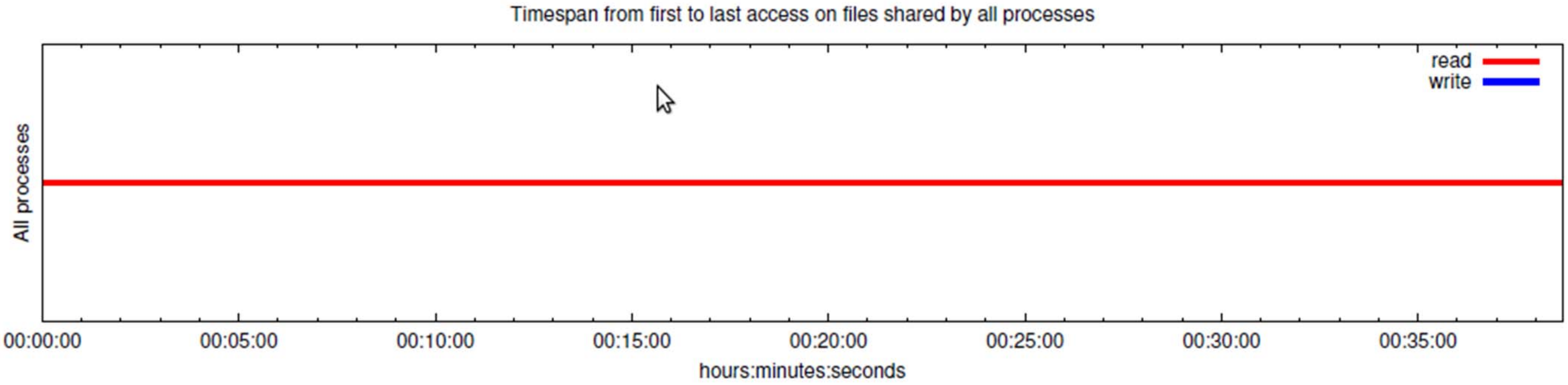
Example: I/O Pattern (1)

- Error Report: Time-out or Slow performance.
- Master-Worker Application (Workers perform the I/O).
- NetCDF Serial (POSIX-IO).
- Darshan report the I/O time of 90%

File with more impact (Read Only)	Size
1. AK135f_5s_fwd_8MB/MZZ/Data/ordered_output.nc4	70GB
2. AK135f_5s_fwd_8MB/MXX_P_MYY/Data/ordered_output.nc4	70GB
3. AK135f_5s_fwd_8MB/MXZ_MYZ/Data/ordered_output.nc4	104GB
4. AK135f_5s_fwd_8MB/MXY_MXX_M_MYY/Data/ordered_output.nc4	104GB
5. AK135f_5s_bwd_8MB/PZ/Data/ordered_output.nc4	70GB
6. AK135f_5s_bwd_8MB/PX/Data/ordered_output.nc4	104GB
Total	522 GB

- Scalasca profile shows that the problem is related with the function load_strain_point_interp (I/O time of 70%) in the call nc_getvar() for FWD in the readfields routine.

Example: I/O Pattern (2)

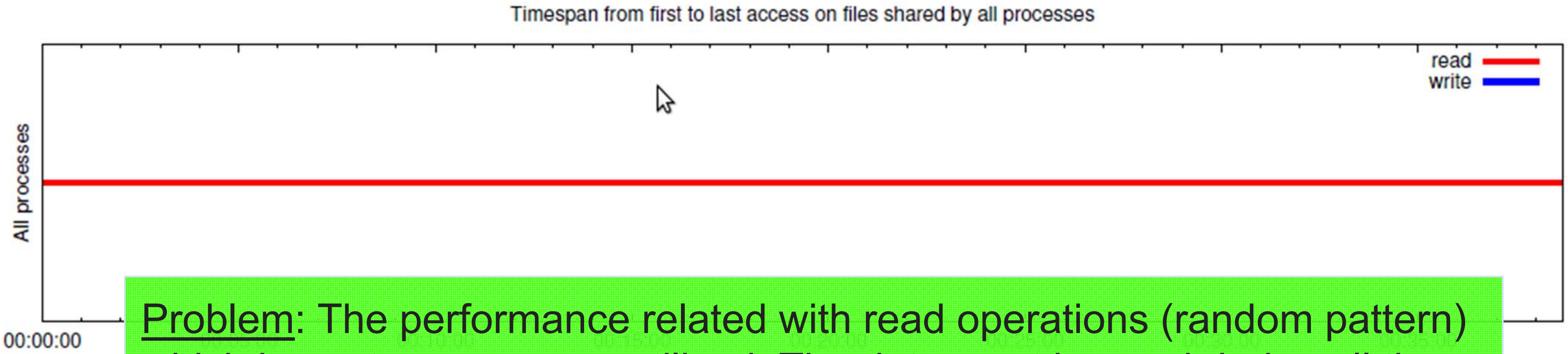


Data Transfer Per Filesystem

File System	Write		Read	
	MiB	Ratio	MiB	Ratio
/gss/scratch	0.00000	0.00000	1400386.82951	1.00000
/home	18.55555	1.00000	2.07610	0.00000

Darshan reports that the application is moving 1.5 TB or 2 TB, usually each process is moving 6GB per file for file sizes of 104 GB and 4GB for the file sizes of 70GB.

Example: I/O Pattern (3)



Problem: The performance related with read operations (random pattern) which impacts on memory utilized. The data must be read during all the execution. Each worker loads the files in memory because each worker is an I/O process.

Recommendation: The user must be avoid the random pattern (that is possible) and reduce the number of I/O processes to avoid loads several times the same file.

Darshan reports that the application is moving 1.5 TB or 2 TB, usually each process is moving 6GB per file for file sizes of 104 GB and 4GB for the file sizes of 70GB.

Example: I/O Scalability

Experimental Environment:

- An Island of thin nodes (512 compute nodes)
- 2 processors per compute node and 8 cores per processor.
- Size of shared memory per node is 32 GBytes (2 GBytes per core).
- $WORK filespace (GPFS version 3.5). Up to 180 GB/s.
- Block size equal to 8,388,608 bytes (8MiB) and a minimum fragment size of 262,144 bytes.
- IBM MPI 1.3, Parallel HDF5 1.8.14, and Darshan 2.3.1.

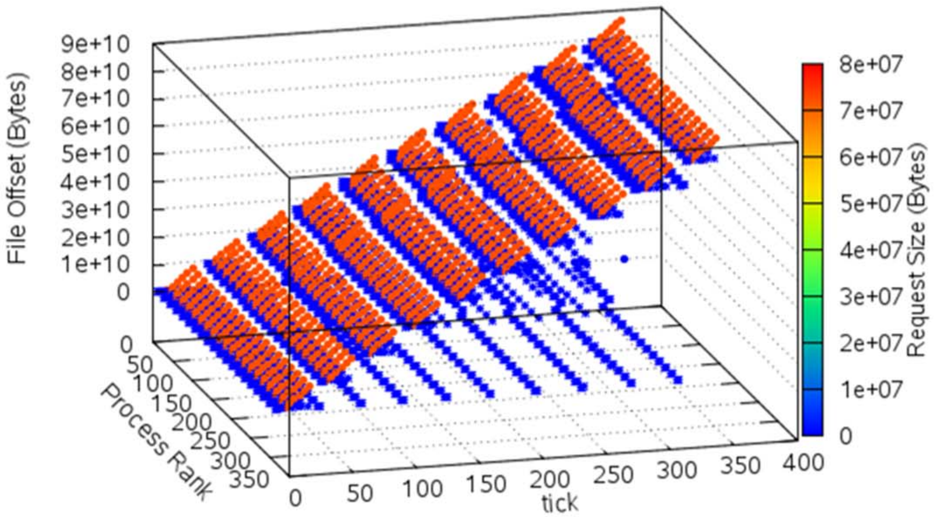
Example: I/O Scalability (1)

- A Particle-in-Cell Application.
- I/O Problem: I/O Scalability Analysis
- Parallel HDF5.

I/O Call tree

Order	MPI-I/O Operation	Data Access Aspect
1	MPI_File_open	
2	Once only by rank 0. MPI_File_get_size	
3	From seven to twelve times MPI_File_read_at	blocking, noncollective, explicit offset
4	Six times (once for each field) MPI_File_set_view MPI_File_write_at_all MPI_File_set_view MPI_File_read_at	 blocking, collective, explicit offset blocking, noncollective, explicit offset
5	Only for the first seven I/O processes MPI_File_write_at	blocking, noncollective, explicit offset
6	MPI_File_set_size	
7	MPI_File_close	

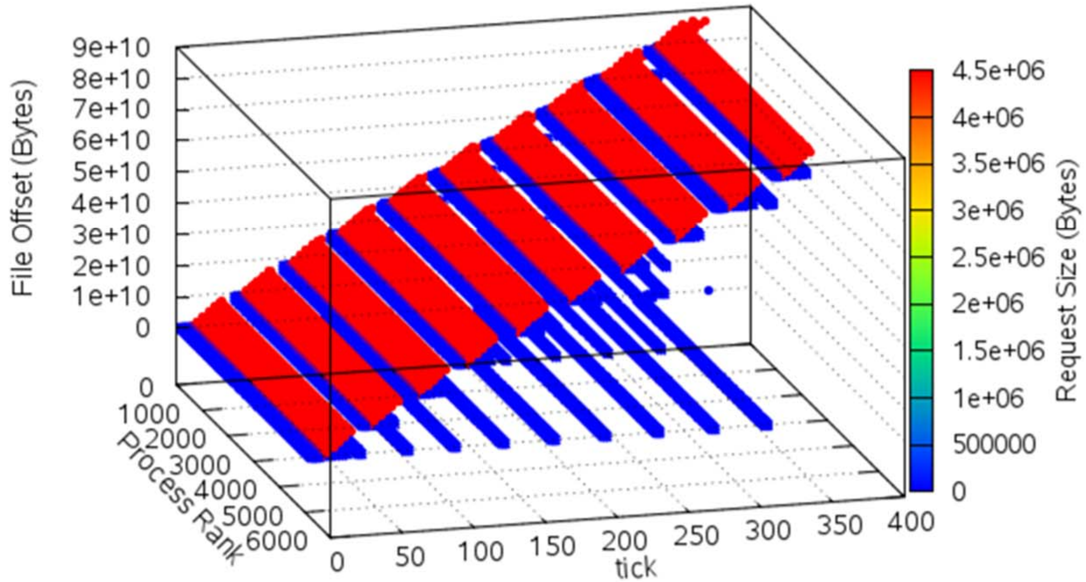
Example: I/O Scalability (2)



Blue = read operations

Red = write operations

I/O Phases Identified (red with more data to transfer)



Example: I/O Scalability (3)

Application I/O Parameters

I/O Parameter	Values
Global Simulation Size	(x, y, z)
Local Simulation Size	$(x\_loc = x, y\_loc = y, z\_loc = \frac{z}{np})$
Compute Nodes	cn
Simulation step	st
fields	fi
writer processes	$wp = cn$
Data Size (Bytes)	ds
RequestSize(Bytes)	$rs = x\_loc \times y\_loc \times z\_loc \times ds$
FileSize(Bytes)	$fz = cn \times rs \times st \times fi$
Data per st (Bytes)	$D_{st} = cn \times rs \times fi$
Data per 1 cn per st (Bytes)	$D_{cnxst} = rs \times fi$

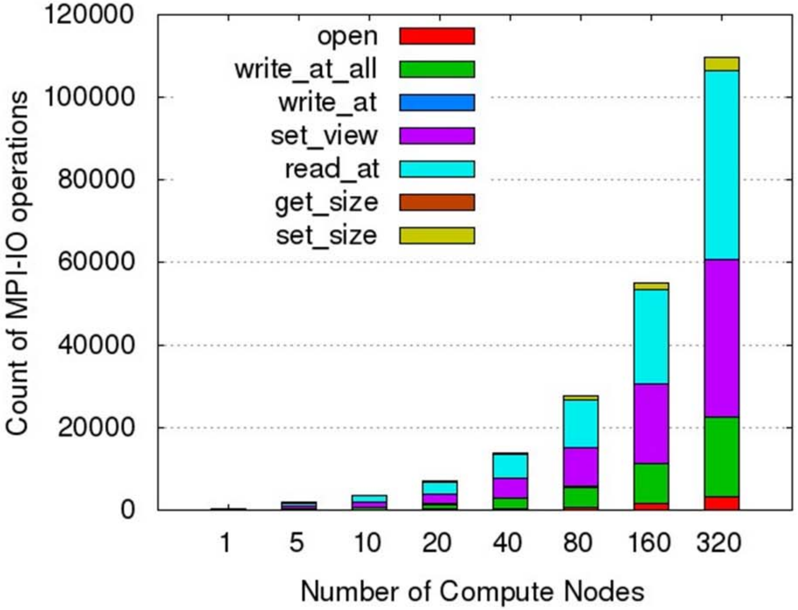
MPI-IO operations considering the I/O Parameters

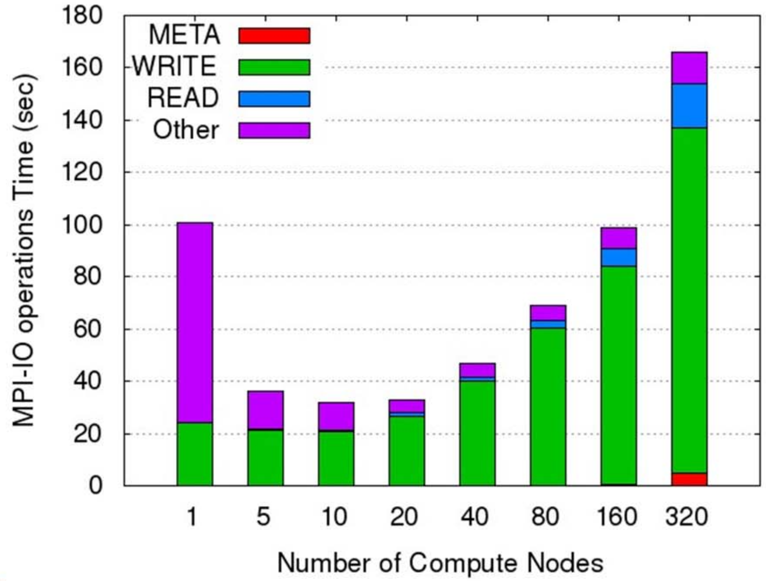
I/O Operation	Count
open	$st \times cn$
write_at_all	$st \times fi \times cn$
write_at	$7 \times st$
set_view	$st \times fi \times cn \times 2$
read_at	$2 \times fi \times st \times$ $\times cn + 23 \times cn$
get_size	st
set_size	$st \times cn$
close	$st \times cn$

Example: I/O Scalability (4)

Global Simulation Size is (52,52,66560), File size = 82 GiB, 16 processes per compute node, 8.05 GiB per Simulation Step, 10 step simulation, 6 fields, 128 data size.

Compute Nodes (cn)	Number of Processes np	Local Simulation Size	Request Size rs (MiB)	Data per 1 cn per st D_{cnxst} (MiB)
1	16	(52,52,4160)	1373.13	8238.75
5	80	(52,52,832)	274.63	1647.75
10	160	(52,52,416)	137.31	823.88
20	320	(52,52,208)	68.66	411.94
40	640	(52,52,104)	34.33	205.97
80	1280	(52,52,52)	17.16	102.98
160	2560	(52,52,26)	8.58	51.49
320	5120	(52,52,13)	4.29	25.75

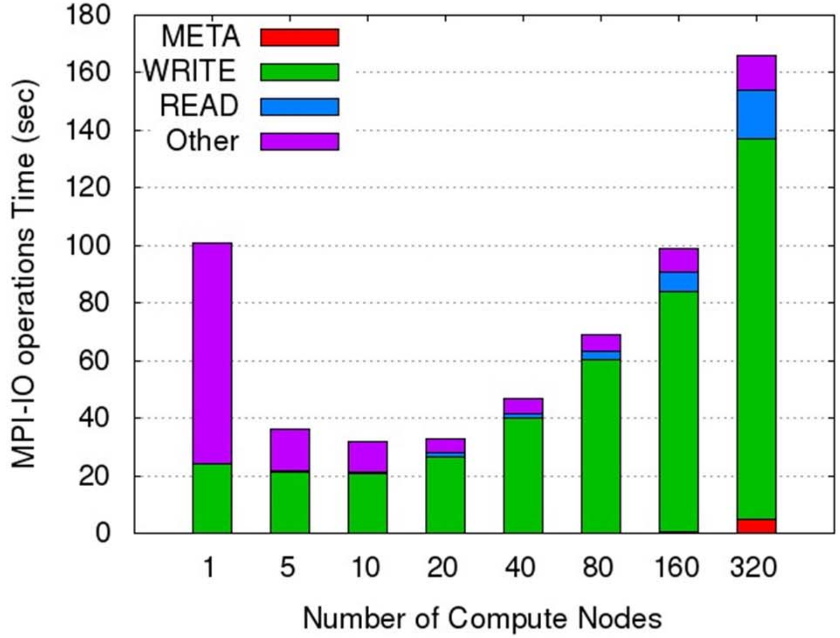


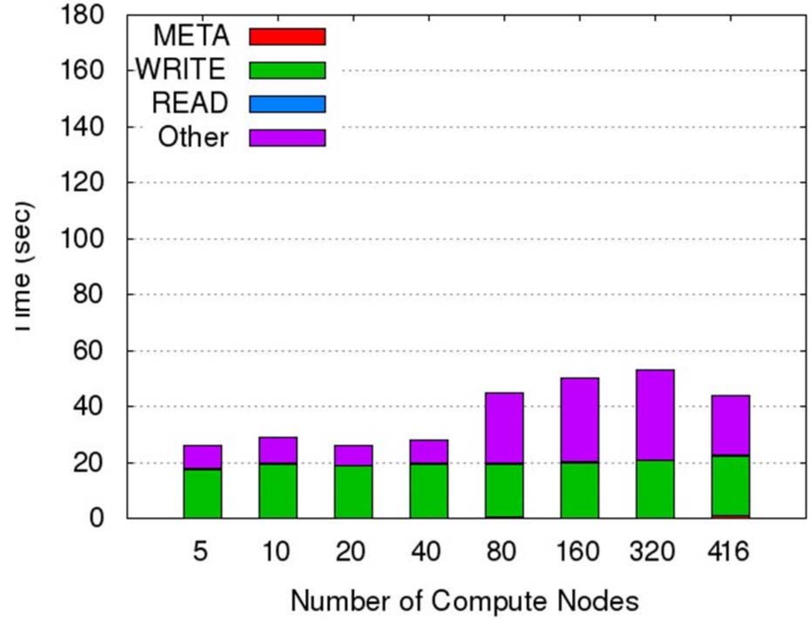


Example: I/O Scalability (5)

Problem: A scalability problem is produced for the strong scaling. The user writes the same amount of data and only increases the compute workload. If the number of I/O processes grows as increases the number of compute nodes then the I/O will impact in the run time.

Recommendation: reduce the number of I/O processes. As consequence the I/O Time remains constant (Right Figure)





Outline

- SuperMUC supercomputer
- User Projects
- Monitoring Tool
- I/O Software Stack
- I/O Analysis Tool
- Analyzing I/O Problems
- **Conclusions**

Conclusions and Future Work

Conclusions:

- The I/O Pattern is usually the source of slow performance.
- POSIX-IO is the library more used in SuperMUC for small and medium jobs (1 Island)
- Parallel I/O is being including at large scale (more than 2 Islands).
- I/O Aggregation has more impact on the scalability (Number of I/O processes per compute nodes).

Future Work:

- I/O Pattern and Performance Analysis at compute node level with Persyst Tool.
- Automatic I/O profiling with Darshan Tool on SuperMUC.
- I/O Scalability analysis using a formal method for detection of I/O phases and I/O operations counters.
- Integration of Darshan logs into Splunk monitoring for identifying data patterns and diagnosing problems at system level.

Thank you for your attention!