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

Phase 2 Compute Nodes
- Mellanox FDR14 Island switch
- Haswell-EP 28 cores/node 2.3 GB/core
- Each >14336 Cores 6 Haswell islands

I/O System
- GPFS for SWORK $SCRATCH
- 10+5 PB 200+100 GB/s Parallel Storage

Phase 1 Compute Nodes
- Mellanox FDR10 Island switch
- Thin + Fat islands of SuperMC
- Each >8126 Cores 18 Thin Node islands 1 Fat Node Island

LRZ infrastructure (NAS, Archive, Visualization)
Internet / Grid Services

© 2016 LRZ
The I/O PATH on SuperMUC - Parallel Storage (WORK and SCRATCH file space)

**Fat+Thin Islands Phase 1**
- 10 x (8NSD + 1 DDN)
- 1 DDN (1 SFA12K):
  - 564 SATA drives (HDD size is 3TB, NL SATA)
  - 56 RAID6 (LUNS 8+2P)
  - Stripe width is 2MiB
  - Stripe size 256kiB
  - 4 hot Spare drives

**Haswell Islands Phase 2**
- 12 PB (WORK)
- 8 x GSS26
- 1 GSS26:
  - 6 x 58 HDDs (HDD size is 4TB, NL SATA)
  - 2SSDs plus NVRAM for securing log information.
  - 2 x NSD / GNR (GPFS Native Raid) servers
  - Stripe width /vdisk track size is 8MiB
  - Strip size is 1028 kiB (1024kiB data + 4kiB check sum)

**Virtual GPFS Clusters**
- I/O Switch Phase 1
- I/O Switch Phase 2
- Infiniband FDR10
- Infiniband FDR14

**I/O Island Core IB Switch Phase 1**
**12Gbps SAS**
**10 x (8NSD + 1 DDN)**
**1 DDN (1 SFA12K):**
- 564 SATA drives (HDD size is 3TB, NL SATA)
- 56 RAID6 (LUNS 8+2P)
- Stripe width is 2MiB
- Stripe size 256kiB
- 4 hot Spare drives

**GPFS (Parallel Storage)**
- FDR10 IB
- 80 NSD
- 12 PB (WORK)
- 16 NSD
- 5 PB (SCRATCH)
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

- Computational-Fluid-Dynamics (CFD)
- Astrophysics-Cosmology (APH)
- Informatics-ComputerSciences (INF)
- Chemistry (CHE)
- Biophysics-Biology-Bioinformatics (BIO)
- Physics-High-EnergyPhysics (HEP)
- Physics-Solid-State (FKP)
- Geophysics (GEO)
- Engineering-others (ENG)
- Meteorology-Climatology-Oceanography (CLI)
- Other
Data Transferred by Research Area

![Graph showing data transferred by research area for January, February, and May.]

- **Read-January**
- **Write-January**
- **Read-Feb**
- **Write-Feb**
- **Read-May**
- **Write-May**

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

© 2016 LRZ

High Performance Parallel Input/Output on LRZ HPC Systems

*HPC I/O in the Data Center Workshop*
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
One Year (06.06.15 19:01 - 20.06.16 19:01)

Datasource Throughput

GPFS Monitor on SuperMUC: SCRATCH

Throughput (B/s)

Jun Jul Aug Sep Oct Nov Dec Jan Feb Mar Apr May Jun

-200 G -100 G 0 100 G 200 G

read 0.00 B/s Last 47.66 GB/s Average 172.43 GB/s Max
write 0.00 B/s Last 50.14 GB/s Average 166.23 GB/s Max
One Year (06.06.15 19:00 - 20.06.16 19:00)

Datasource Throughput

- **read**: 0.00 B/s Last 53.18 GB/s Average 341.75 GB/s Max
- **write**: 0.00 B/s Last 30.75 GB/s Average 166.48 GB/s Max
One Year (06.06.15 18:59 - 20.06.16 18:59)

Datasource Throughput

Throughput (B/s)

-1.0 G  0.0  1.0 G  2.0 G

Jun  Jul  Aug  Sep  Oct  Nov  Dec  Jan  Feb  Mar  Apr  May  Jun

NFS throughput

- read 0.00 B/s Last 535.09 MB/s Average 2.36 GB/s Max
- write 0.00 B/s Last 158.11 MB/s Average 1.48 GB/s Max
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

- High-Level I/O Library
- I/O Middleware
- Low-Level I/O Library
- Global Filesystem
- Storage Infrastructure

HPC Application
- Parallel HDF5, NetCDF Parallel, pnetCDF, ADIOS, CGNS Parallel.
- SIONLib
- HDF5, NetCDF, CGNS.

I/O Infrastructure
- MPI-IO (ROMIO)
- POSIX-IO
- IBM GPFS - 8 MB Block Size ($SCRATCH and $WORK)
- NAS NFS ($HOME)

- $SCRATCH
  - 5.2 PB
  - Up to 180GB/sec
- $WORK
  - 12 PB
  - Up to 250GB/sec
- $HOME
  - 1.5 PB
  - Up to 10GB/sec
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

<table>
<thead>
<tr>
<th>Exp.</th>
<th>MPI Processes</th>
<th>Compute Nodes</th>
<th>Access Pattern</th>
<th>Request Size</th>
<th>Hints</th>
<th>Transfer Rate(GiB/sec)</th>
</tr>
</thead>
</table>
| 1    | 512           | 32            | Sequential     | 1 MiB        | romio.cb_read = automatic  
             romio.cb_write = automatic  
             romio.cb_read = disable  
             romio.cb_write = disable | write = 25.92 
                                read = 23.80 | **write = 80.39** 
                                **read = 69.62** |
| 2    | 512           | 32            | Strided        | 1 MiB        | romio.cb_read = automatic  
             romio.cb_write = automatic  
             romio.cb_read = enable  
             romio.cb_write = enable | write = 1.63  
                                read = 17.74 | **write = 25.49**  
                                **read = 26.10** |
| 3    | 512           | 32            | Sequential     | 256 MiB      | romio.cb_read = automatic  
             romio.cb_write = automatic  
             romio.cb_read = disable  
             romio.cb_write = disable | write = 74.48 
                                read = 46.67 | **write = 83.37** 
                                **read = 65.88** |
| 4    | 512           | 32            | Strided        | 256 MiB      | romio.cb_read = automatic  
             romio.cb_write = automatic  
             romio.cb_read = disable  
             romio.cb_write = disable  
             romio.cb_read = enable  
             romio.cb_write = enable | write = 71.12 
                                read = 41.80 | **write = 77.22** 
                                **read = 70.21** |
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:

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

```
HPC Application

Parallel HDF5, NetCDF Parallel, pnetCDF, ADIOS, CGNS Parallel.

SIONLib

HDF5, NetCDF, CGNS.

MPI-IO (ROMIO)

POSIX-IO

IBM GPFS - 8 MB Block Size
($SCRATCH$ and $WORK$)

NAS NFS ($HOME$)

$SCRATCH$
- 5.2 PB.
- Up to 150 GB/sec

$WORK$
- 12 PB.
- Up to 250 GB/sec

$HOME$
- 3.5 PB
- 12 GB/sec
```
Darshan Components

Running Application MPI

Dynamic library preloading: libdarshan.so

Collecting I/O characterization
MPI-I/O and POSIX-I/O

System where it intends to instrument MPI applications

Darshan runtime (instrumentation tool)

Log files

System where it intends to analyze log files

Darshan util

• I/O counters
• I/O time percentage
• Access pattern
• ...

© 2016 LRZ

High Performance Parallel Input/Output on LRZ HPC Systems

HPC I/O in the Data Center Workshop
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.
  ```bash
  module load darshan
  ```
- Set up the variable FORTRAN_PROG in “true” if the program is a Fortran program and false if it’s not.
  ```bash
  FORTRAN_PROG=true
  ```
- Load the appropriate library.
  ```bash
  export LD_PRELOAD=`darshan-user.sh $FORTRAN_PROG`
  ```
- Set up Darshan job identifier with loadleveler job identifier.
  ```bash
  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.
  ```bash
  export DARSHAN_JOBID=JOBID_LL
  ```
- Set up Darshan log path
  ```bash
  export LOGPATH_DARSHAN_LRZ=`darshan-logpath.sh`
  ```
- **Darshan - Splunk format**
  ```bash
  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

I/O Analysis

Not related with the I/O

3

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

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.

Providing a solution or recommendation

- 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
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 ($8$MiB) 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)

- 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)

<table>
<thead>
<tr>
<th>File Path</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. AK135f_5s_fwd_8MB/MZZ/Data/ordered_output.nc4</td>
<td>70GB</td>
</tr>
<tr>
<td>2. AK135f_5s_fwd_8MB/MXX_P_MYY/Data/ordered_output.nc4</td>
<td>70GB</td>
</tr>
<tr>
<td>3. AK135f_5s_fwd_8MB/MXZ_MYZ/Data/ordered_output.nc4</td>
<td>104GB</td>
</tr>
<tr>
<td>4. AK135f_5s_fwd_8MB/MXY_MXX_M_MYY/Data/ordered_output.nc4</td>
<td>104GB</td>
</tr>
<tr>
<td>5. AK135f_5s_bwd_8MB/PZ/Data/ordered_output.nc4</td>
<td>70GB</td>
</tr>
<tr>
<td>6. AK135f_5s_bwd_8MB/PIX/Data/ordered_output.nc4</td>
<td>104GB</td>
</tr>
</tbody>
</table>

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.
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.
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

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

<table>
<thead>
<tr>
<th>I/O Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Simulation Size</td>
<td>((x, y, z))</td>
</tr>
<tr>
<td>Local Simulation Size</td>
<td>((x_{loc} = x, y_{loc} = y, z_{loc} = \frac{z}{np}))</td>
</tr>
<tr>
<td>Compute Nodes</td>
<td>(cn)</td>
</tr>
<tr>
<td>Simulation step</td>
<td>(st)</td>
</tr>
<tr>
<td>fields</td>
<td>(fi)</td>
</tr>
<tr>
<td>writer processes</td>
<td>(wp = cn)</td>
</tr>
<tr>
<td>Data Size (Bytes)</td>
<td>(ds)</td>
</tr>
<tr>
<td>RequestSize (Bytes)</td>
<td>(rs = x_{loc} \times y_{loc} \times z_{loc} \times ds)</td>
</tr>
<tr>
<td>FileSize (Bytes)</td>
<td>(fz = cn \times rs \times st \times fi)</td>
</tr>
<tr>
<td>Data per (st) (Bytes)</td>
<td>(D_{st} = cn \times rs \times fi)</td>
</tr>
<tr>
<td>Data per 1 (cn) per (st) (Bytes)</td>
<td>(D_{cn \times st} = rs \times fi)</td>
</tr>
</tbody>
</table>

#### MPI-IO operations considering the I/O Parameters

<table>
<thead>
<tr>
<th>I/O Operation</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>open</td>
<td>(st \times cn)</td>
</tr>
<tr>
<td>write_at_all</td>
<td>(st \times fi \times cn)</td>
</tr>
<tr>
<td>write_at</td>
<td>(7 \times st)</td>
</tr>
<tr>
<td>set_view</td>
<td>(st \times fi \times cn \times 2)</td>
</tr>
<tr>
<td>read_at</td>
<td>(2 \times fi \times st \times cn + 23 \times cn)</td>
</tr>
<tr>
<td>get_size</td>
<td>(st)</td>
</tr>
<tr>
<td>set_size</td>
<td>(st \times cn)</td>
</tr>
<tr>
<td>close</td>
<td>(st \times cn)</td>
</tr>
</tbody>
</table>
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.

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

© 2016 LRZ
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 included 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!