I/O at the German Climate Computing Center (DKRZ)

Julian M. Kunkel, Carsten Beyer

kunkel@dkrz.de

German Climate Computing Center (DKRZ)

16-07-2015
Outline

1. Introduction
2. Workload
3. System View
4. Obstacles
5. R&D
6. Summary
About DKRZ

German Climate Computing Center

DKRZ – Partner for Climate Research
Maximum Compute Performance.
Sophisticated Data Management.
Competent Service.
Scientific Computing

- Research Group of Prof. Ludwig at the University of Hamburg
- Embedded into DKRZ

Research

- Analysis of parallel I/O
- I/O & energy tracing tools
- Middleware optimization
- Alternative I/O interfaces
- Data reduction techniques
- Cost & energy efficiency
Scientific Workflow

A typical workflow

- HPC simulation
  - ES-Models
  - Initial conditions or checkpoint
  - Checkpoint
  - at termination (<= 8 h)
- Post-processing
  - Tool A
  - Tool B
  - Scientific Variables
- Federated data archive
  - Tool C
  - visualized
  - External usage of data products
  - Tool ?

Technical background

- Application/domain-specific I/O servers for HPC-IO
- Different post-processing tools
- Involved libraries/formats: NetCDF4 (HDF5), NetCDF3, GRIB, ...
HPC-IO with Application-specific I/O Servers

Since parallel I/O is slow and not offering the right features, users are developing their own I/O middleware

I/O servers
- Subset of processes dedicated for I/O
- Act as burst buffers and fix file system issues
- May asynchronously pull data from the model
- May perform additional data conversion (grid, reductions...)
- Example tools: XIOS, CDI-PIO (> 4 in the climate community!)

Challenges
- Adds another complex layer (not) easy to understand
- Performance portability
- Coupling of models with different styles of I/O servers
- Process mapping and parameterization
Job Mix

One year on Blizzard
- Typically small (analysis) jobs
- A few large (model) runs
- ca. 4% peak
Last Supercomputer: The Blizzard Supercomputer

- **Computation:** 249 nodes
  - Microprocessors: 16 Power6 dual-core (total: 7968 cores)
  - Memory: 64 or 128 GByte per node (2 or 4 GB per core)
  - Interconnect: 2 DDR-Infiniband quad-port adapters ⇒ max 5 GB/s

- **File systems:** GPFS
  - Servers: 12 I/O nodes (same hardware as compute nodes)
  - Capacity: 7 Petabyte
  - Storage hardware
    - 6480x 1TB HD Sata (RAID 6, 4+2P)
    - 1440x 2TB HD Sata (RAID 6, 8+2P)
    - HDDs are connected using FC via 24x IBM DS5300 Controller
  - Metadata hardware
    - 56x 146GB 15K SCSI FC HDDs
    - Connected by 3x IBM DS4700 and 1x DS5300 with expansion
  - Max. throughput: 30 GByte/s
Tape Library with HPSS

- 6 Oracle/StorageTek SL8500 libraries (+ a smaller one)
  - More than 67,000 slots
- One SL8500 library at Garching for backups/disaster recovery
- Variety of tape cartridges/drives
- On Blizzard: 500 TB disk cache
- Update on Mistral: 3 PB disk cache
Performance in Production on Blizzard

- Average FC-throughput (for all I/O servers)
  - 3 GB/s read
  - 1 GB/s write

- Metadata statistics across login and interactive nodes
  - Captured using mmpmon
  - Average 1000 open/close per s
  - Average 150 readdir per s
  - Compute nodes require much less
Understanding the Data Stored

<table>
<thead>
<tr>
<th>Mount</th>
<th># of Files</th>
<th>Total Size</th>
<th>Avg. file size</th>
</tr>
</thead>
<tbody>
<tr>
<td>home</td>
<td>23 M</td>
<td>90 TByte</td>
<td>0.2 MiB</td>
</tr>
<tr>
<td>work</td>
<td>117 M</td>
<td>5273 TByte</td>
<td>38.1 MiB</td>
</tr>
<tr>
<td>scratch</td>
<td>28 M</td>
<td>420 TByte</td>
<td>15.5 MiB</td>
</tr>
</tbody>
</table>

![Size frequency distribution for home/project folders](image1)

![Size frequency distribution for work/scratch](image2)
File Formats

Motivation
- Gear optimization effort towards mostly used I/O libraries
- Understand the requirements for the procurement

Accuracy of the approach
- Many users use numerical extensions for created files
- 40% of small files have the extension "data" or "meta"

Results
- NetCDF: 21 Million files (17% of files, 34% of capacity)
- Grib: 9 M files
- HDF5: 200 K files
- Tar: 12% capacity!
File Formats

- Problem: File extensions do not match the content
  ⇒ Sample of files analyzed with file and cdo
- 25% from home
- 20% from work/scratch: 1 PB, 26 M files

Scientific file formats for work/scratch

- No Scientific Format
- txt
- NetCDF
- GRIB
- NetCDF2
- Others

No Scientific Format
- 21.0%
- NetCDF
- 32.0%
- GRIB
- 15.0%
- NetCDF2
- 22.0%
- EXTRA
- 2.0%
- IEG
- 2.0%
- netCDF4 SZIP
- 2.0%
- GRIB SZIP
- 2.0%
- Others
- 2.0%
Insights from File Analysis

Home:
- Not much insight
- Mostly code/objects
- Many empty directories, broken links ...

Work/Scratch:
- Many old/inefficient file formats around
- Many small files + TXT
- A small fraction of data volume is compressed:
  - 2% NetCDF and 2% GRIB SZIP, 3% GZIP compressed
  - A small fraction (3% of volume) of NetCDF4/HDF5
Mistral Supercomputer

- Phase 1 system, installed Q2/15
- Vendor: Atos (Bull)
- Nodes: 1500 with 2 Intel E5-2680 Haswell@2.5 GHz
  - 24 cores/node
  - 2 Nodes/blade, 9 blades/Chassis, 4 Chassis/Rack
- HPL-performance: 1.1 Petaflop/s
- Storage capacity: 20 Petabyte
- Network: FatTree with FDR-14 Infiniband
  - 3 Mellanox SX6536 core 648-port switches
  - 1:2:2 blocking factor
  - 1:1 within chassis (18 nodes)
  - 1:2 9 uplinks per chassis, to 3 linecards on each core switch
  - 1:2 between linecards and spinecards
- Power consumption (HPL): 700 kW
ClusterStor Servers
Phase 1: I/O Architecture

- Lustre 2.5 (+ Seagate patches: some back ports)
- 29 ClusterStor 9000 with 29 Extensions (JBODs)
  - 58 OSS with 116 OST
- ClusterStor 9000 SSUs
  - GridRaid: 41 HDDs, PD-RAID with 8+2(+2 spare blocks)/RAID6, 1 SSD for Log
  - 6 TByte disks
  - SSU: Active/Active failover server pair
  - ClusterStor Manager
  - 1 FDR uplink/server
- Peak performance
  - Infiniband FDR-14: 6 GiB/s ⇒ 348 GiB/s
  - CPU/6 GBit SAS: 5.4 GiB/s ⇒ 313 GiB/s
- Multiple metadata servers
  - Root MDS + 4 DNE MDS
  - Active/Active failover (DNEs, Root MDS with Mgmt)
  - DNE phase 1: Assign responsible MDS per directory
Performance Results

- Throughput measured with IOR
  - Buffer size 2000000 (unaligned)
  - 84 OSTs (Peak: 227 GiB/s)
  - 168 client nodes, 6 procs per node

<table>
<thead>
<tr>
<th>Type</th>
<th>Read</th>
<th>Write</th>
<th>Write rel. to peak</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX, independent¹</td>
<td>160 GB/s</td>
<td>157 GB/s</td>
<td>70%</td>
</tr>
<tr>
<td>MPI-IO, shared²</td>
<td>52 GB/s</td>
<td>41 GB/s</td>
<td>18%</td>
</tr>
<tr>
<td>PNetCDF, shared</td>
<td>81 GB/s</td>
<td>38 GB/s</td>
<td>17%</td>
</tr>
<tr>
<td>HDF5, shared</td>
<td>23 GB/s</td>
<td>24 GB/s</td>
<td>10%</td>
</tr>
<tr>
<td>POSIX, single stream</td>
<td>1.1 GB/s</td>
<td>1.05 GB/s</td>
<td>0.5%</td>
</tr>
</tbody>
</table>

- A few slow servers significantly reduce IOR performance
  - Also: Congestion on IB routes degrade performance

- Metadata measured with a load using Parabench: 80 kOPs/s

¹ 1 stripe per file
² 84 stripes per file on 21 SSUs
Monitoring Tools

On Mistral

- For compute
  - Nagios (status & performance)
  - Planned: XDMoD (for utilization)
  - Slurm statistics (accounting)

- Seagate’s Data Collection System (DCS)
  - Metadata and data rates
  - CPU and MEM utilization
  - Node Health

- ltop
- cscli lustre_perf
- ClusterStor Manager

On Blizzard

- Nagios
- llview (for Load-Leveler)
- Ganglia
- ibview
Monitoring I/O Performance with ClusterStor
Obstacles

Lack of knowledge

- Usage of file formats and middleware libraries is limited
  - Analysis of file extensions does not suffice
  - Library usage could theoretically be monitored, but ...

- The workflows of users is sometimes diffuse

- The cause of inefficient operations is unknown

Shared nature of storage

- With 1/60th of nodes one can drain 1/7th of I/O performance
  \[ \Rightarrow \] 10% of nodes drain all performance
  - Applications may use 10% I/O over time, this seems fine

- But: interaction of ill-formed I/O degrades performance
  - I/O intense benchmark increased application runtime by 100%

- Metadata workloads are worse, problematic with broken scripts
Obstacles

Difficulties in the analysis

- Performance is sensitive to I/O patterns, concurrent activity
- Infiniband oversubscription
- Application-specific I/O servers increase complexity
- Capturing a run’s actual I/O costs
- Lustre’s (performance) behavior

Others

- Outdated (and inefficient) file formats are still dominant
- Performance of RobinHood may be too slow (2000 ops/s)
- Capability increase from Blizzard to Mistral
  - Compute performance by 20x
  - Storage performance by 20x
  - Storage capacity by 7x ⇒ Data compression is an option
Consequences

There is a need for

- Guaranteed performance for large-scale simulation
- An automatic and systematic analysis of users’ workflow
- Interfaces and middleware to avoid domain-specific I/O servers
- (Lossy) compression to improve TCO
- Methods to understand I/O performance
Dealing with Storage in ESiWACE

H2020 project: ESiWACE Center of Excellence

Work package 4

Partners: DKRZ, STFC, ECMWF, CMCC, Seagate

1. Modelling costs for storage methods and understanding these
2. Modelling tape archives and costs
3. Focus: Flexible disk storage layouts for earth system data
   - Reduce penalties of „shared“ file access
   - Site-specific data mapping but simplify import/export
   - Allow access to the same data from multiple high-level APIs
Scalable I/O for Extreme Performance (SIOX)

Started as collaborative project between UHH, ZIH and HLRS

SIOX aims to
- collect and analyse
  - activity patterns and
  - performance metrics
- system-wide

In order to
- assess system performance
- locate and diagnose problem
- learn optimizations
SIOX Ongoing Work

**Automatic assessing the quality of the I/O**

Your Read I/O consisted of:
- 200 calls/100 MiB
- 10 calls/10 MiB were cached in the system’s page cache
- 10 calls/20 MiB were cached on the server’s cache
- 100 calls/40 MiB were dominated by average disk seek time (0.4s time loss)
- ...
- 5 calls/100 KiB were unexpected slow (1.5s time loss)

**Follow up Project**

- Together with our partners we submitted a follow up project
- To increase scalability and assessment capability
Virtual Laboratory for I/O Investigation

Virtual Lab: Conduct what if analysis

- Design new optimizations
- Apply optimization to application w/o changing them
- Compute best-cases and estimate if changes pay off

Methodology

- Extract application I/O captured in traces
  1. Allow manipulation of operations and replay them in a tool
  2. Allow on-line manipulation

So far: Flexible Event Imitation Engine for Parallel Workloads (feign)

- Helper functions: to pre-create environment, to analyze, ...
- A handful of mutators to alter behavior
Planned R&D

Accounting of I/O

- Account jobs based on their demand for I/O in Slurm
- Simple approach use statistics from `/proc/self/io`
- Use system-wide statistics or via application instrumentation?

Reduce interference of concurrent I/O

- Evaluate methods to ensure performance for large-scale runs
- Fence inefficient I/O using storage pools/Network Request Scheduler?
- System wide burst-buffers vs. application-specific servers?
- Consider interference of small file accesses to parallel I/O
  - 400 GByte SSD-tier could host all files < 8 KiB (30% of files)
- In-situ visualization
Summary

- Climate research is data intensive science
- The lack of knowledge of user activity is costly
  - A focus on R&D on most beneficial optimizations is not possible
  - Users may use suboptimal tools and I/O methods
- Understanding system behavior and performance is painful
- Maybe we could increase our TCO with e.g. by
  - data compression (and providing less capacity)
  - providing less storage bandwidth
- R&D in our research group fosters
  - understanding performance and costs
  - aims for optimization (with little change from user perspective)