Karlsruhe Institute of Technology

Parallel I/O

Introduction to Parallel I/O

Begatim Bytyqi | 14. June 2023

Outline

Introduction

What's I/O?

- I/O stands for Input/Output.
- Migration of data from memory to storage and vice versa.
- Why is it a challenge?

What’s I/O?

- I/O stands for Input/Output.
- Migration of data from memory to storage and vice versa.
- Why is it a challenge?
  1. Computation is prioritized.
  2. Complex stack.
  3. Scientists think in terms of their science problem.
  4. Data is stored as bytes.

I/O Stack in HPC clusters

- Application
- High Level I/O Library
- MPI-I/O Layer
- Parallel File Systems
- Storage Device

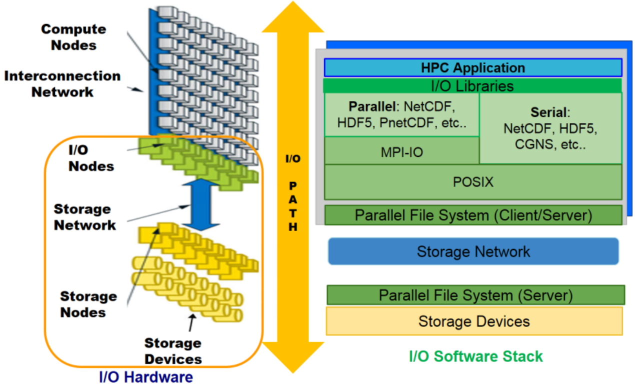


Figure: Typical High Performance I/O System[1]

Storage hierarchy

- Higher
  - Latency decreases
  - Cost increases.
  - Capacity decreases.
- Lower
  - Latency increases.
  - Cost decreases.
  - Capacity increases.

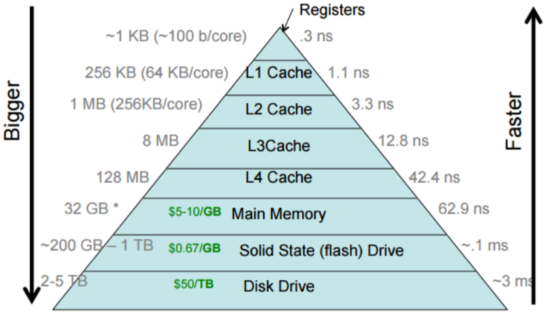


Figure: Storage hierarchy [2]

Hard disks

- Sequential
  - Read time, 0.1ms
- Random
  - Read time 7.1ms (Why??)

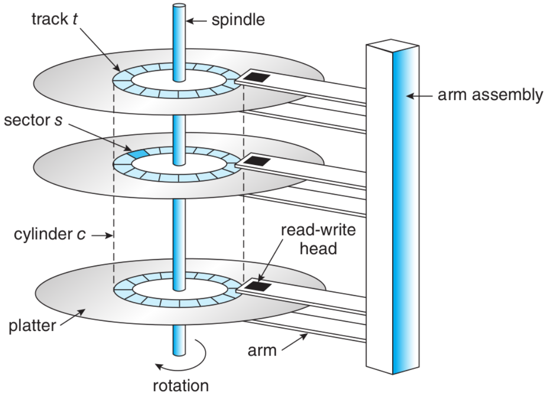


Figure: HDD moving-head disk mechanism [4]

Hard disks

- Sequential
  - Read time, 0.1ms
- Random
  - Read time 7.1ms (Why??)
  - Seek time, 4ms
  - Rotation time, 3ms
  - Read time, 0.1ms

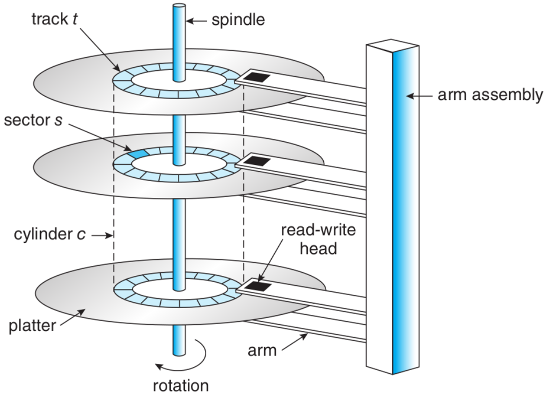


Figure: HDD moving-head disk mechanism [4]

Why not just use SSDs then?

Advantages

1. 10x higher throughput compared to HDDs
2. 100x lower latency compared HDDs
3. Similar performance for both random and sequential accesses.

Disadvantages

1. 10x higher cost
2. Limited endurance
3. Complexity

Usage

- Metadata management, node local storage and small sized parallel file systems.

RAID

Redundant Arrays of Independent (Inexpensive) Disks

Storage arrays for higher capacity, throughput and reliability.

- Capacity
  - Storage array can be addressed like a single, large device.
- Throughput
  - All storage devices contribute to the overall throughput
- Reliability
  - In case of failure data isn't lost.
- Possible configurations
  - RAID0: block-level striping (no redundancy)
  - RAID1: mirroring
  - RAID4: block-interleaved parity
  - RAID5: block-interleaved distributed parity
  - RAID6: P+Q redundancy

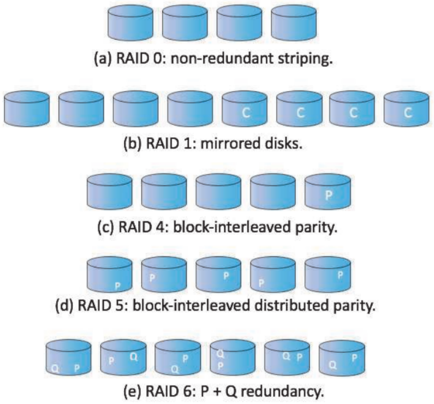


Figure: RAID levels [4]

File Systems

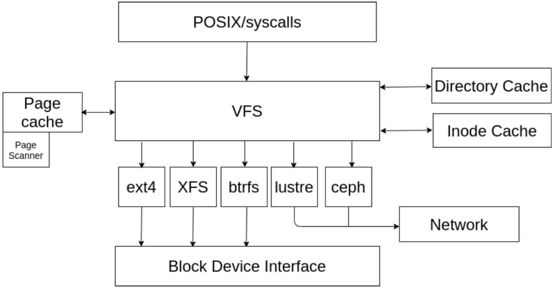
Overview

Karlsruhe Institute of Technology

- File system provides structure
  - Tree structure whose nodes are *directories* and leafs are *files*.
  - Hierarchical organization of data.
- Seamless data and metadata management.
  - This is provided by a special structure called *inode*.
  - Block allocation is important.
  - File type, permission bytes, timestamps, size etc.
- Map logical I/O requests to physical I/O requests.

Virtual File System

- An abstraction layer that allows different file systems to be accessed using a common interface.
- Forwards application requests based on path.
- Key features:
  - Interoperability
  - Transparency
  - Extensibility
  - Security
  - Caching
- Caches:
  - Inode cache
  - Data cache
  - Directory cache



[1] Gregg, Brendan. Systems performance: enterprise and the cloud. Pearson Education, 2014.

Files

What is a file?

Files

What is a file?

- POSIX is the IEEE Portable Operating Standard Interface for Computing Environments. It defines a standard way for an application program to obtain basic services from the operating system.
- POSIX: An object that can be written to, or read from or both.
  It has certain attributes, including access permissions and type.
- An object consisting of data and metadata.
  - Data is basically just byte arrays.
  - Metadata is the description of the actual file data.
- Directories contain files and directories.

Parallel File Systems

Overview

- Network file system.
- Provides access to shared resources.
- Parallel access to thousands of clients.
- All the complexities abstracted.
- POSIX compliant.
- Access via I/O interface.
- Most used parallel file systems:
  - Lustre
  - IBM Spectrum Scale (GPFS)
  - BeeGFS - On Demand BeeGFS

Striping

- Striping is a technique used to improve the performance and scalability of file systems.
- It involves splitting a single large file into smaller chunks, known as data blocks.
- These data blocks are then distributed or "striped" across multiple storage units.
- Enhances throughput performance.
- Increases the file size limit,
- Some file systems, such as Lustre and BeeOND, provide flexibility by allowing users to adjust the striping policy to optimize performance.
- Possible contention issues.
- Performance improvement only visible when I/O is done in parallel.

Lustre
Overview

- Object-based Parallel Distributed Kernel Space FS
- Servers
  - MGS, MDS, OSS
- Targets
  - MGT, MDT, OST
- Clients
  - MGC, MDC, OSC
- Clients are using a POSIX layer and do not have any direct access to the underlying storage.
- 2-writes before data lands into the disk sub-system.

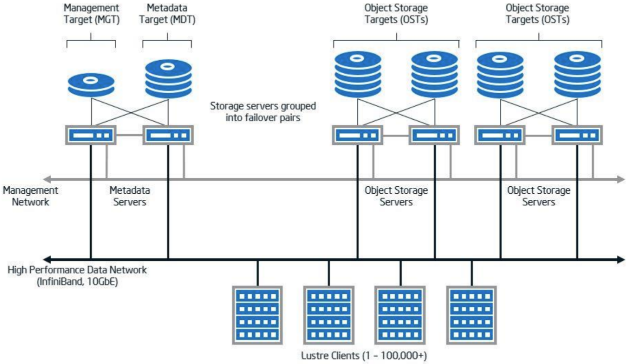


Figure: Lustre architecture Copyright@ 2014 Intel Corporation

IBM Spectrum Scale (GPFS)
Overview

Karlsruhe Institute of Technology

- Block-based Parallel Distributed Kernel space FS.
- Models
  - SAN Model
  - NSD Server Model
  - SNC Model
- Performance scales with the Network Storage Device (NSD) servers.
- Distributed locking mechanism for data and metadata synchronization
- Metadata can be delegated to Clients.
- 1-write needed to the underlying Disk sub-system.
- Features like Replication, QoS are available for quite some time.

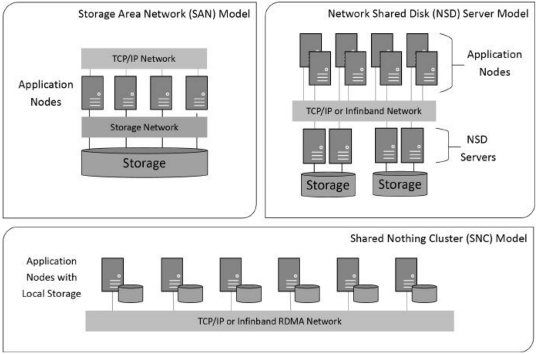


Figure: GPFS architecture [3]

BeeOND

- Removes the I/O load from the shared file system.
- Doesn't use spining disks as storage hardware.
- A shared namespace.
- Only available during job lifetime. Requires data staging.

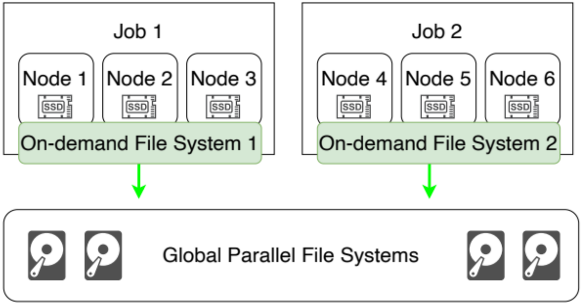


Figure: BeeOND architecture

$Home vs $Workspace

Data is classified into two main groups:

1. Hot data (Data used frequently)
   - Data written and read by applications.
   - Not backed up.
   - To be stored for long term storage.
   - Requires high performant storage.
2. Cold data (Data rarely used)
   - Data stored for long term usage.
   - Backed up.
   - Requires more capacity.
   - Shouldn’t be accessed excessively by HPC applications.

Property	$HOME	workspace	$TMPDIR	BeeOND
Visibility	global	global	local	job local
Lifetime	permanent	limited	job walltime	job walltime
Disk space	2.5 PB	13.5 PB	800 GB	n * 750 GB
Quotas	yes	yes	no	no
Snapshot	yes	yes	no	no
Backup	yes	no	no	no
Total read perf	25 GB/s	110 GB/s	750 MB/s	n * 700 MB/s
Total write perf	25 GB/s	110 GB/s	750 MB/s	n * 700 MB/s
Read perf/node	10 GB/s	10 GB/s	750 MB/s	10 GB/s
Write perf/node	10 GB/s	10 GB/s	750 MB/s	10 GB/s

Figure: File systems at HoreKA

Performance tuning

- General suggestions:
  - Fewer I/O requests with large blocks of data.
  - Use local node storage for task local files, to reduce the load from the shared file system.
  - Use several clients to increase the throughput performance.
  - Avoid competitive file accesses, to reduce contention between clients.
  - Don’t use HOME directory!!!. Use a workspace instead.
- Lustre
  - Choosing a stripe size.
    - For shared files, maximal striping should be used.
    - The stripe size should be a multiple of the page size.
    - The smallest recommended stripe size is 512KB.
    - For large file counts, a smaller stripe size gives better performance.
    - A good stripe size for sequential I/O using high-speed networks is between 1 MB and 4 MB. Maximum stripe size is 4 GB
- BeeOND
  - Choosing the number of metadata servers.
  - Choosing the stripe size (similar suggestions as for Lustre).

Parallel I/O

Collated I/O

n-processes, 1-worker

Advantages

- Easy to implement.
- Single file accessed.

Disadvantages

- Memory consumption
- Scalability
- Idle workers.
- Limited throughput

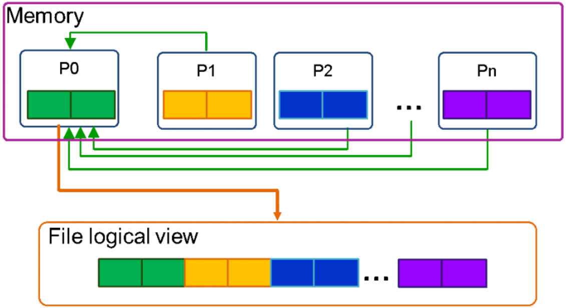


Figure: Serial I/O [1]

Independent I/O

n-processes, n-workers

Advantages

- Easy to implement.
- High throughput rates.
- No communication needed.

Disadvantages

- Large number of files.
- Metadata performance issues.

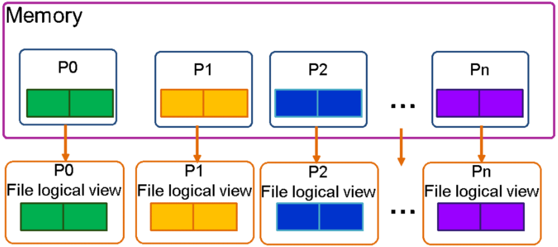


Figure: File per process model [1]

Collective I/O

n-processes, n-workers

Advantages

- Single file used.
- Good performance.

Disadvantages

- Contention issues.
- Synchronization required.

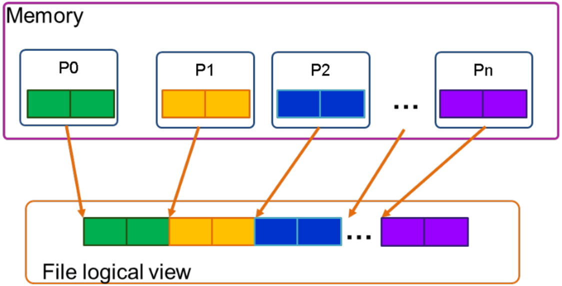


Figure: Single file, multiple writers model.[1]

Collective I/O

n-processes, k-workers (n>k)

Advantages

- Single file used.
- Scalability.
- Tuning capabilities.
- Less I/O requests.

Disadvantages

- Complexity.
- Communication.

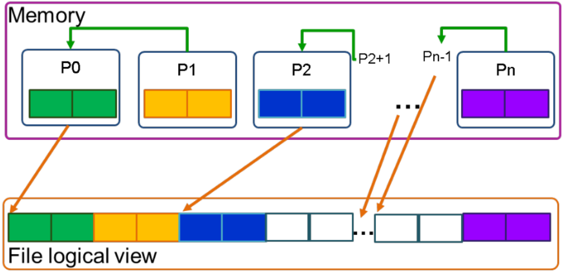


Figure: Shared file, collective writers [1]

Managing Concurrent Access

- Files are treated as random global shared memory regions,allowing multiple clients to access them simultaneously.
- Clients are required to acquire locks on the units they intend to access before performing any I/O operations
- To ensure orderly access, byte range locks are utilized, which allow fine-grained control over concurrent accesses to different parts of the file.
- Unit boundaries are dictated by the storage system regardless of access pattern.
- Enables caching on clients as well- as long as client has a lock, it knows its cached data is valid.
- Locks are reclaimed from clients when other desire access.

Libraries

MPI I/O

- Provides low-level interface to carrying out parallel I/O.
- Facilitate concurrent access by groups of processes.
- Two approaches;
  - **Independent**
    - Each MPI rank is handling the I/O independently using non-collective calls like MPI_File_write() and MPI_File_read().
    - Similar to POSIX-I/O, but supports derived datatypes and thus noncontiguous data and nonuniform strides and can take advantage of MPI_Hints.
  - **Collective**
    - When doing collective I/O all MPI tasks participating in I/O have to call the same routines.
    - This allows the MPI library to do I/O optimizations, like data sieving and two-phase I/O.

pNetCDF
Parallel NetCDF

Karlsruhe Institute of Technology

- NetCDF stands for Network Common Data format.
- pNetCDF is a parallel I/O library for accessing NetCDF files in parallel.
- It is built on top of MPI I/O and provides a simple interface for parallel I/O operations on NetCDF files.
- pNetCDF supports non-blocking I/O operations and allows concurrent accesses to the same file.
- It is optimized for accessing data in a row-major format and supports both C and Fortran APIs.
- pNetCDF is used in many scientific applications, including weather forecasting, climate modeling, and data assimilation.

HDF5

- HDF stands for *Hierarchical Data Format*.
- Organizing file contents into a structure similar to Unix-like file system
- HDF5 is a versatile parallel I/O library for managing large and complex datasets.
- It provides a high-level abstraction for storing and retrieving data in hierarchical, structured, or unstructured formats.
- HDF5 supports a wide range of data types, including scalar, array, compound, and variable-length.
- It is designed to work efficiently with parallel file systems and can handle very large files and datasets.
- A feature-rich library with a relatively steep learning curve. Has bindings for various programming languages.

ADIOS2

- ADIOS2 (Adaptive Input/Output System) is a modern parallel I/O library that provides high-performance data streaming and staging capabilities.
- It is designed to handle large-scale scientific data generated by HPC simulations and experiments.
- Build around the *engine*, which abstracts the data transport method. It can be changed by simply modifying a configuration file.
- ADIOS2 supports several transport methods, including MPI, TCP, and RDMA, and provides flexible configuration options.
- It supports various data formats, BP, HDF5.
- ADIOS2 is widely used in HPC applications, including astrophysics, climate modeling, and fusion research.

SIONlib

- Uses a custom SIONlib binary file format.
- Tries to solve the metadata problem of the independent I/O approach.
- Maps logical files into a single physical file.
- Each task maintains the task-local file view.
- Collective: open/close and independent: write/read.
- Replacement for standard I/O APIs

Performance Measurements

I/O Performance Engineering workflow

Karlsruhe Institute of Technology

Pathological workloads are detected by administrators using system monitoring tools.

The typical iterative optimization process is the *closed loop of performance tuning.*

1. Measurements (Collecting metrics)
   - Profiling
   - Tracing
2. Analysis (Measurement visualizations)
   - Offline Analysis
   - Online Analysis
3. Generation of Alternatives
   - Modeling and estimating
4. Implementation
   - Modifying the source code
   - Testing

I/O Performance Engineering workflow

Pathological workloads are detected by administrators using system monitoring tools.

The typical iterative optimization process is the *closed loop of performance tuning.*

1. Measurements (Collecting metrics)
   - Profiling
   - Tracing
2. Analysis (Measurement visualizations)
   - Offline Analysis
   - Online Analysis
3. Generation of Alternatives
   - Modeling and estimating
4. Implementation
   - Modifying the source code
   - Testing

The loop ends when the desired performance is achieved or further tuning is not possible.

Measurements

Karlsruhe Institute of Technology

Types of measurements

- Application-oriented (Application instrumentation)
  - Source code modification
  - Re-link object files with patched functions
  - LD_PRELOAD
  - Modify machine code directly
- System-oriented
  - From /proc file
  - jobstats (Lustre only)
  - mmperfmon
  - beegfs-ctl
  - DDN (Lustre only)

Darshan

Karlsruhe Institute of Technology

- Lightweight, scalable I/O characterization tool.
- Developed by Argonne National Laboratory.
- Profiles HPC applications that use one of the following libraries:
  - POSIX-I/O, MPI-I/O, pNetCDF, and parallel HDF5.
- Hybrid MPI+OpenMP is also supported.
- Instruments I/O (doesn't sample) by intercepting calls.
- Darshan can be setup both at compile time or execution time.
  - Using LD_PRELOAD is recomended.
- After logs are collected:
  - A quick pdf report can be generated.
  - A detailed report can be generated.
  - All information can be dumped into a single ASCII file.

Best practices

Best practices

- In the case of large number of files, structure them into subdirectories.
- Avoid opening and closing files frequently.
- Consider I/O middleware such as ADIOS2, HDF5, pNetCDF or SIONlib.
- Work with less files.
- Large chunk I/O requests.
- Avoid debugging output.
- Perform I/O only when needed.

Questions?

Thank you!!

References I

[1] Mendez et al. *Best Practice Guide - Parallel I/O*. 2019. URL: https://prace-ri.eu/wp-content/uploads/Best-Practice-Guide_Parallel-IO.pdf.

[2] Harward. 2019. URL: https://cs.brown.edu/courses/csci1310/2020/notes/l10.html.

[3] Quincey Koziol et al. *High performance parallel I/O*. CRC Press, 2014.

[4] Abraham Silberschatz, Peter B Galvin, and Greg Gagne. *Operating system concepts essentials*. Wiley Publishing, 2013.