Parallel I/O

Parallel Programming

2024-01-11

Prof. Dr. Michael Kuhn
michael.kuhn@ovgu.de

Parallel Computing and I/O
Institute for Intelligent Cooperating Systems
Faculty of Computer Science
Otto von Guericke University Magdeburg
https://parcio.ovgu.de

Outline

Parallel I/O

- Review
- Introduction and Motivation
- Storage Devices and Arrays
- File Systems
- Parallel Distributed File Systems
- Libraries
- Future Developments
- Summary

- How does one-sided communication work?
  1. One-sided communication works with messages
  2. Every process can access every other address space
  3. Addresses first have to be exposed via windows
  4. System looks like a shared memory system to the processes

- What is the difference between active and passive target communication?
    1. Both origin and target are involved in active target communication
    2. Both origin and target are involved in passive target communication
    3. Only target process is involved in active target communication
    4. Only target process is involved in passive target communication

- What is the purpose of `MPI_Accumulate`?
    1. Can be used to sum multiple values
    2. Can be used to perform specific reductions on values
    3. Can be used to merge multiple windows
    4. Can be used to collect information about processes

- How can deadlocks and race conditions be detected?
  1. The compiler warns about them
  2. Static analysis can detect some errors
  3. Errors can only be detected at runtime

Outline

Parallel I/O

- Review
- Introduction and Motivation
- Storage Devices and Arrays
- File Systems
- Parallel Distributed File Systems
- Libraries
- Future Developments
- Summary

Computation and I/O

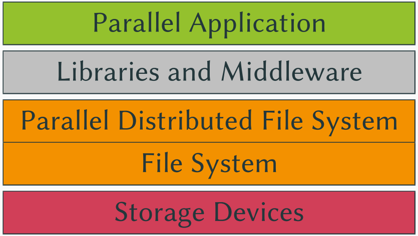
- Parallel applications run on multiple nodes
  - Communication via MPI
- Computation is only one part of applications
  - Input data has to be read
  - Output data has to be written
  - Example: checkpoints
- Processors require data fast
  - Caches should be used optimally
  - Additional latency due to I/O and network

Level	Latency
L1 cache	≈ 1 ns
L2 cache	≈ 5 ns
L3 cache	≈ 10 ns
RAM	≈ 100 ns
InfiniBand	≈ 500 ns
Ethernet	≈ 100,000 ns
SSD	≈ 100,000 ns
HDD	≈ 10,000,000 ns

[Bonér, 2012] [Huang et al., 2014]

Computation and I/O…

- I/O is often responsible for performance problems
  - High latency causes idle processors
  - I/O is often still serial, limiting throughput
- Storage stack is layered
  - Many different components are involved
  - Performance problems influence all layers



Outline

Parallel I/O

- Review
- Introduction and Motivation
- Storage Devices and Arrays
- File Systems
- Parallel Distributed File Systems
- Libraries
- Future Developments
- Summary

Hard Disk Drives

- The first hard disk drive in 1956
  - IBM 350 RAMAC
  - Capacity: 3.75 MB
  - Throughput: 8.8 KB/s
  - Rotational speed: 1,200 RPM
- HDD development is rather slow
  - Capacity: 100× every 10 years
  - Throughput: 10× every 10 years

Improvement of HDD characteristics over time

Parameter	Started with (1957)	Developed to (2019)	Improvement
Capacity (formatted)	3.75 megabytes[17]	18 terabytes (as of 2020)[18]	4.8-million-to-one[19]
Physical volume	68 cubic feet (1.9 m^3)[c][6]	2.1 cubic inches (34 cm^3)[20][d]	56,000-to-one[21]
Weight	2,000 pounds (910 kg)[6]	2.2 ounces (62 g)[20]	15,000-to-one[22]
Average access time	approx. 600 milliseconds[6]	2.5 ms to 10 ms; RW RAM dependent	about 200-to-one[23]
Price	US$9,200 per megabyte (1961)[24]	US$0.024 per gigabyte by 2020[25][26][27]	383-million-to-one[28]
Data density	2,000 bits per square inch[29]	1.3 terabits per square inch in 2015[30]	650-million-to-one[31]
Average lifespan	c. 2000 hrs MTBF[*citation needed*]	c. 2,500,000 hrs (~285 years) MTBF[32]	1250-to-one[33]

[Wikipedia, 2021]

Solid-State Drives

- Advantages
  - Read throughput: Higher by a factor of 15 (150–250 MB/s vs. 0.5–3.5 GB/s)
  - Write throughput: Higher by a factor of 10
  - Latency: Lower by a factor of 100 (75–100 IOPS vs. 90,000–600,000 IOPS)
  - Energy consumption: Lower by a factor of 1–10
- Disadvantages
  - Price: Higher by a factor of 5
  - Endurance: Only allow 10,000–100,000 write cycles
  - Complexity
    - Optimal access size differs for read and write accesses
    - Address translations is more complicated
    - Fast drives can overheat easily

RAID

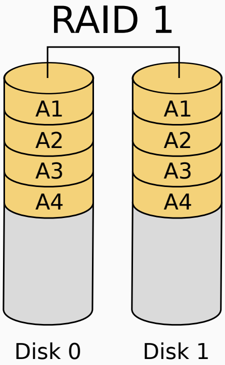
- Storage arrays for higher capacity, throughput and reliability
  - Proposed in 1988 at the University of California, Berkeley
    - Originally: Redundant Array of Inexpensive Disks
    - Today: Redundant Array of Independent Disks
- Capacity
  - Storage array can be addressed like a single, large device
- Throughput
  - All storage devices can contribute to the overall throughput
- Reliability
  - Data can be stored redundantly to survive hardware failures
  - Devices usually have same age, fabrication defects within same batch

RAID...

- Five different variants initially
    - RAID 1: mirroring
    - RAID 2/3: bit/byte striping
    - RAID 4: block striping
    - RAID 5: block striping with distributed parity
- New variants have been added
    - RAID 0: striping
    - RAID 6: block striping with double parity

RAID 1

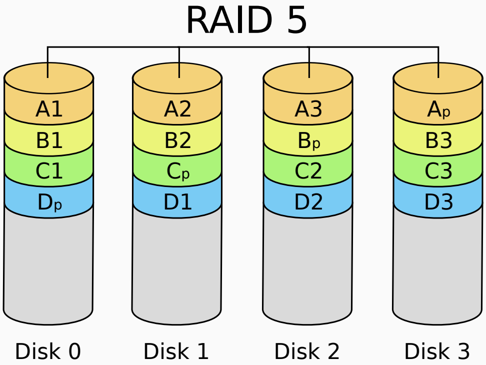
- Improved reliability via mirroring
- Advantages
  - One device can fail without losing data
  - Read performance can be improved
- Disadvantages
  - Capacity requirements and costs are doubled
  - Write performance equals that of a single device



[Cburnett, 2006a]

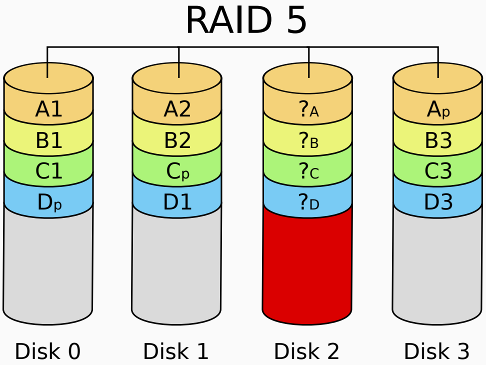
RAID 5

- Improved reliability via parity
    - Typically simple XOR
- Advantages
    - Performance can be improved
    - Requests can be processed in parallel
    - Load is distributed across all devices



[Cburnett, 2006b]

- Data can be reconstructed easily due to XOR
  - $?_A = A1 \oplus A2 \oplus A_p$, $?_B = B1 \oplus B2 \oplus B3, \dots$
- Problems
  - Read errors on other devices
  - Duration (30 min in 2004, 19–20 h in 2021 for HDDs)
  - New approaches like declustered RAID



[Cburnett, 2006b]

- Which RAID level would you choose for a server with 10 HDDs?
    1. RAID 0 (striping)
    2. RAID 1 (mirroring)
    3. RAID 5 (block striping with distributed parity)
    4. RAID 6 (block striping with distributed double parity)

Outline

Parallel I/O

- Review
- Introduction and Motivation
- Storage Devices and Arrays
- File Systems
- Parallel Distributed File Systems
- Libraries
- Future Developments
- Summary

Overview

- File systems provide structure
  - Files and directories are the most common file system objects
  - Nesting directories results in hierarchical organization
    - Other approaches: tagging
- Management of data and metadata
  - Block allocation is important for performance
  - Access permissions, timestamps etc.
- File systems use underlying storage devices or arrays

File System Objects

- User vs. system view
  - Users see files and directories
  - System manages inodes
    - Relevant for `stat` etc.
- Files
  - Contain data as byte arrays
  - Can be read and written (explicitly)
  - Can be mapped to memory (implicit)
- Directories
  - Contain files and directories
  - Structure the namespace

I/O Interfaces

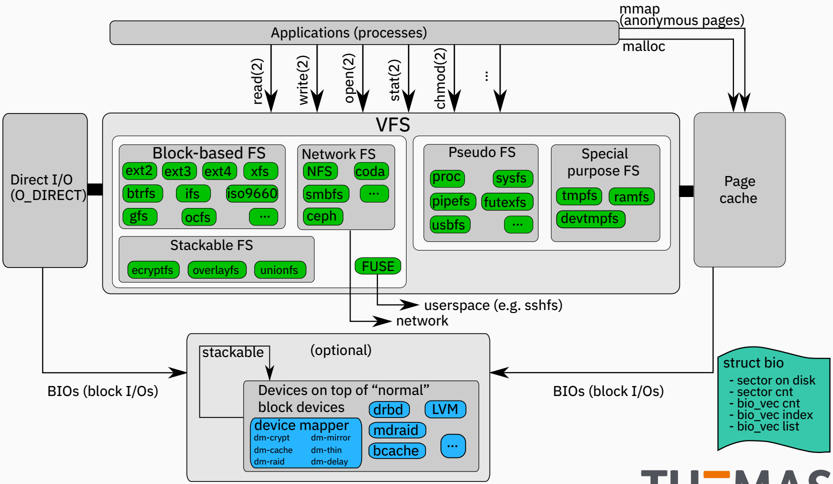
- Requests are realized through I/O interfaces
  - Forwarded to the file system
- Different abstraction levels
  - Low-level functionality: POSIX etc.
  - High-level functionality: NetCDF etc.
- Initial access via path
  - Afterwards access via file descriptor (few exceptions)
- Functions are located in `libc`
  - Library executes system calls

```
fd = open("/path/to/file", ...);
nb = write(fd, data,
           sizeof(data));
rv = close(fd);
rv = unlink("/path/to/file");
```

Virtual File System (Switch)

- Central file system component in the kernel
  - Sets file system structure and interface
- Forwards applications' requests based on path
- Enables supporting multiple different file systems
  - Applications are still portable due to POSIX
- POSIX: standardized interface for all file systems
  - Syntax defines available operations and their parameters
    - open, close, creat, read, write, lseek, chmod, chown, stat etc.
  - Semantics defines operations' behavior
    - `write`: *"POSIX requires that a read(2) which can be proved to occur after a write() has returned returns the new data. Note that not all filesystems are POSIX conforming."*

VFS… [Werner Fischer and Georg Schönberger, 2017]



The Linux Storage Stack Diagram
http://www.thomas-krenn.com/en/wiki/Linux_Storage_Stack_Diagram
Created by Werner Fischer and Georg Schönberger

THOMAS KRENN®

Modern File Systems

- File system demands are growing
  - Data integrity, storage management, convenience functionality
- Error rate for SATA HDDs: 1 in 10^{14} to 10^{15} bits [Seagate, 2016]
  - That is, one bit error per 12.5–125 TB
  - Additional bit errors in RAM, controller, cable, driver etc.
- Error rate can be problematic
  - Amount can be reached in daily use
  - Bit errors can occur in the superblock
- File system does not have knowledge about storage array
  - Knowledge is important for performance
  - For example, special options for ext4

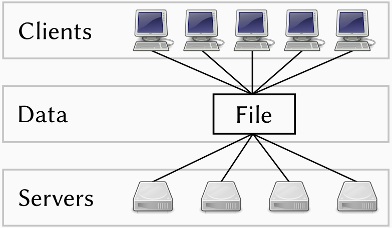
Outline

Parallel I/O

- Review
- Introduction and Motivation
- Storage Devices and Arrays
- File Systems
- Parallel Distributed File Systems
- Libraries
- Future Developments
- Summary

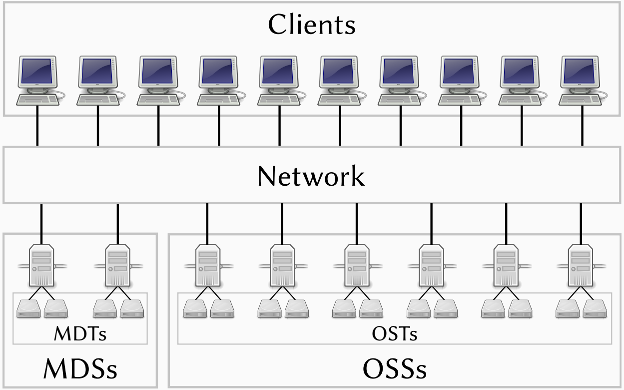
Overview

- Parallel file systems
  - Allow parallel access to shared resources
  - Access should be as efficient as possible
- Distributed file systems
  - Data and metadata is distributed across multiple servers
  - Single servers do not have a complete view
- Naming is inconsistent
  - Often just “parallel file system” or “cluster file system”



Architecture

- Access via I/O interface
  - Typically standardized, frequently POSIX
- Interface consists of syntax and semantics
  - Syntax defines operations, semantics defines behavior
- Data and metadata servers
  - Different access patterns
  - Data vs. request throughput

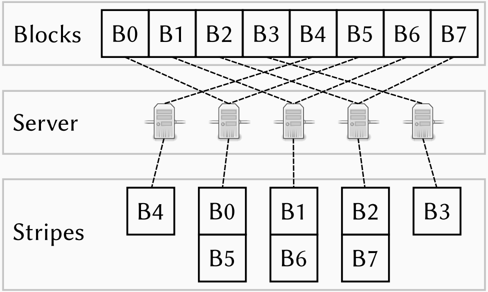


Semantics

- POSIX has strong consistency/coherence requirements
  - Changes have to be visible globally after `write`
  - I/O should be atomic to avoid inconsistencies
- POSIX for local file systems
  - Requirements easy to support due to VFS
- Contrast: Network File System (NFS)
  - Same syntax, different semantics
- Session semantics in NFS
  - Changes only visible to other clients after session ends
  - `close` writes changes and returns potential errors

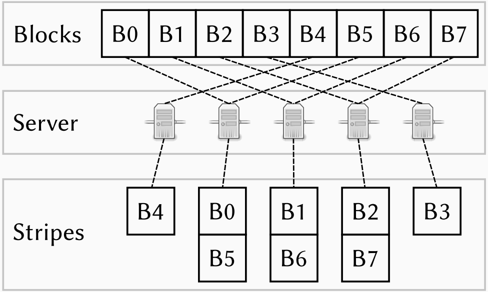
Data Distribution

- File is split up into blocks
  - Blocks are distributed across servers
  - Here, eight blocks across five servers
  - Blocks typically have static size
- Round-robin distribution often used
  - Restart at first server after last
- Does not have to start at first server
  - Typically randomly chosen server



Quiz

- Why is the starting server chosen randomly?
  - Easy implementation
  - Even load distribution
  - Fault tolerance



Performance

- 2009: Blizzard (GPFS)
  - Computation: 158 TFLOPS
  - Capacity: 7 PB
  - Throughput: 30 GB/s
- 2015: Mistral (Lustre)
  - Computation: 3.6 PFLOPS
  - Capacity: 60 PB
  - Throughput: 450 GB/s (5.9 GB/s per node)
  - IOPS: 400,000 operations/s
- 2022: Levante (Lustre)
  - Computation: 14 PFLOPS
  - Capacity: 130 PB
- 2012: Titan (ORNL, Lustre)
  - Computation: 17.6 PFLOPS
  - Capacity: 40 PB
  - Throughput: 1.4 TB/s
- 2019: Summit (ORNL, Spectrum Scale)
  - Computation: 148.6 PFLOPS
  - Capacity: 250 PB
  - Throughput: 2.5 TB/s

Outline

Parallel I/O

- Review
- Introduction and Motivation
- Storage Devices and Arrays
- File Systems
- Parallel Distributed File Systems
- Libraries
- Future Developments
- Summary

Overview

- Low-level interfaces can be used for parallel I/O
    - They are typically not very convenient for developers
- Parallel applications require support for efficient parallel I/O
    - Synchronous and serial I/O are bottlenecks
    - Reading input data and writing output data
- Additional problems
    - Exchangeability of data, complex programming, performance
- Libraries offer additional functionality
    - SIONlib (performance optimizations)
    - NetCDF, HDF (self-describing data and exchangeability)
    - ADIOS (abstract I/O)

MPI-IO

- MPI-IO is a part of MPI specifying a portable I/O interface
  - Has been introduced with MPI-2.0 in 1997
- One of the most popular implementations is called ROMIO
  - ROMIO is developed and distributed as part of MPICH
  - It is used in OpenMPI and MPICH derivatives
  - Supports many file systems via Abstract-Device Interface for I/O (ADIO)
- MPI-IO provides element-based access to data
  - The interface is very similar to MPI's communication interface
  - Supports collective and non-blocking operations as well as derived datatypes

- MPI-IO defines the syntax and semantics of I/O operations
  - Changes are only visible in the current process immediately
  - Non-overlapping or non-concurrent operations are handled correctly
- POSIX I/O has strong coherence and consistency semantics
  - Changes have to visible globally after a `write` and should be atomic
  - Makes it hard to cache data and often requires locks
- Relaxed semantics have less overhead in distributed environments
  - Improved scalability and less need for locking
- MPI-IO semantics is usually enough for scientific applications
  - For example, non-overlapping access to a shared matrix

Positioning	Blocking	Individual	Collective
Explicit Offset	Blocking	read_at write_at	read_at_all write_at_all
	Non-Blocking & Split Collective	iread_at	read_at_all_begin read_at_all_end
		iwrite_at	write_at_all_begin write_at_all_end
Individual File Pointers	Blocking	read write	read_all write_all
	Non-Blocking & Split Collective	iread	read_all_begin read_all_end
		iwrite	write_all_begin write_all_end
Shared File Pointer	Blocking	read_shared write_shared	read_ordered write_ordered
	Non-Blocking & Split Collective	iread_shared	read_ordered_begin read_ordered_end
		iwrite_shared	write_ordered_begin write_ordered_end

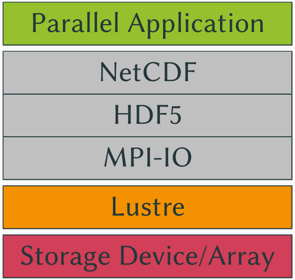
NetCDF

- Developed by Unidata Program Center
    - University Corporation for Atmospheric Research
- Mainly used for scientific applications
    - Especially in climate science, meteorology and oceanography
- Consists of libraries and data formats
    1. Classic format (CDF-1)
    2. Classic format with 64 bit offsets (CDF-2)
    3. Classic format with full 64 bit support (CDF-5)
    4. NetCDF-4 format
- Data formats are open standards
    - CDF-1 and CDF-2 are international standards of the Open Geospatial Consortium

NetCDF...

- NetCDF supports groups and variables
  - Groups contain variables, variables contain data
  - Attributes can be attached to variables
- Supports multi-dimensional arrays
  - `char`, `byte`, `short`, `int`, `float` and `double`
  - NetCDF-4: `ubyte`, `ushort`, `uint`, `int64`, `uint64` and `string`
- Dimensions can be sized arbitrarily
  - Only one unlimited dimension with CDF-1, CDF-2 and CDF-5
  - Multiple unlimited dimensions with NetCDF-4

Interaction



- Data transformation
  - Data is transported through all layers
  - Loss of high-level information
- Complex interactions
  - Optimizations and workarounds on all layers
  - Information about other layers required
- Convenience vs. performance
  - Structured data in application
  - Byte stream in POSIX

Outline

Parallel I/O

- Review
- Introduction and Motivation
- Storage Devices and Arrays
- File Systems
- Parallel Distributed File Systems
- Libraries
- Future Developments
- Summary

Storage Hierarchy

- Current state
  - L1, L2, L3 cache, RAM, SSD, HDD, tape
- Latency gap from RAM to SSD
  - Performance loss if data is not in RAM
- Performance gap is worse on supercomputers
  - RAM is node-local, data is in parallel distributed file system

Level	Latency
L1 cache	≈ 1 ns
L2 cache	≈ 5 ns
L3 cache	≈ 10 ns
RAM	≈ 100 ns
SSD	≈ 100,000 ns
HDD	≈ 10,000,000 ns
Tape	≈ 50,000,000,000 ns

[Bonér, 2012] [Huang et al., 2014]

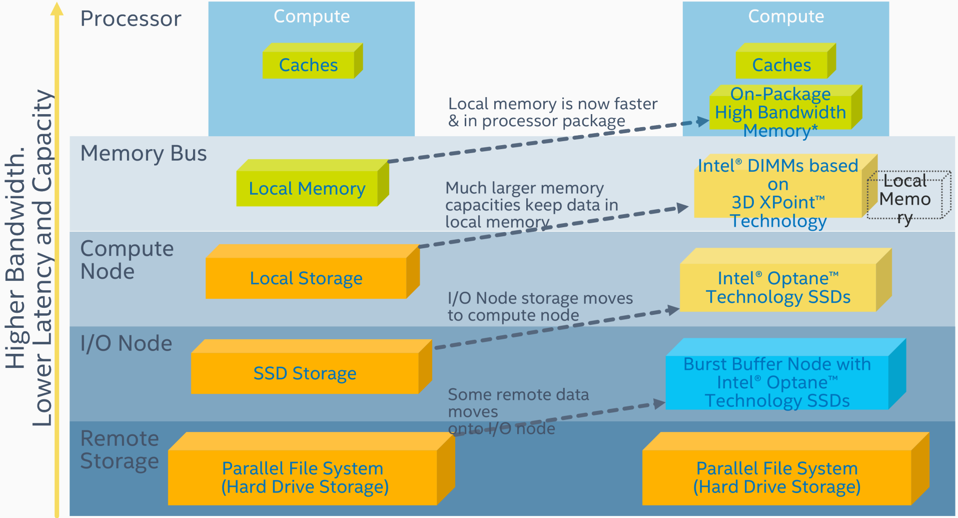
Storage Hierarchy

- Current state
  - L1, L2, L3 cache, RAM, SSD, HDD, tape
- Latency gap from RAM to SSD
  - Performance loss if data is not in RAM
- Performance gap is worse on supercomputers
  - RAM is node-local, data is in parallel distributed file system
- New technologies to close gap
  - Non-volatile RAM (NVRAM), NVM Express (NVMe) etc.

Level	Latency
L1 cache	≈ 1 ns
L2 cache	≈ 5 ns
L3 cache	≈ 10 ns
RAM	≈ 100 ns
NVRAM	≈ 1,000 ns
NVMe	≈ 10,000 ns
SSD	≈ 100,000 ns
HDD	≈ 10,000,000 ns
Tape	≈ 50,000,000,000 ns

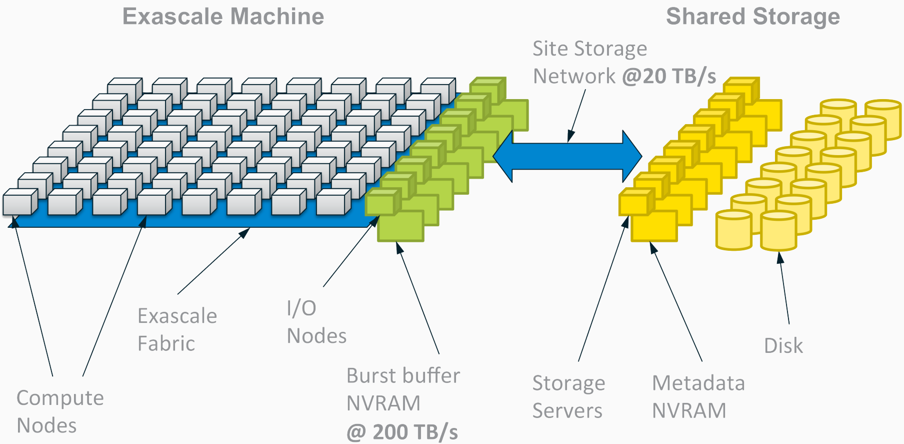
[Bonér, 2012] [Huang et al., 2014]

Storage Hierarchy… [Brent Gorda, 2016]



Storage Hierarchy... [Brent Gorda, 2013]

- I/O nodes with burst buffers close to compute nodes
- Slower storage network to file system servers



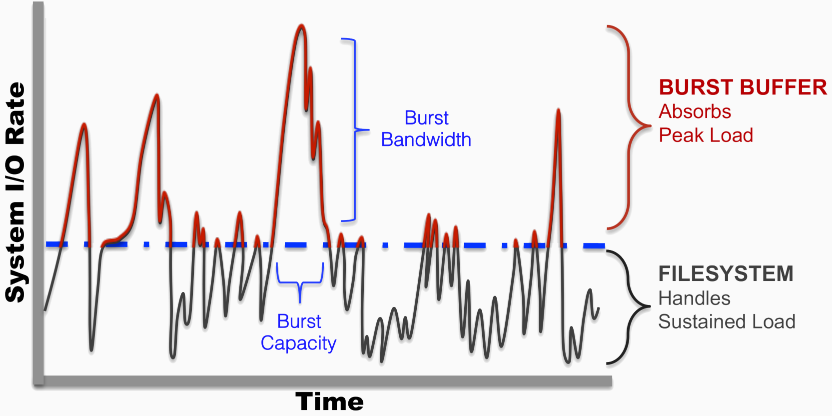
- NVRAM only readily available as Intel Optane Persistent Memory
  - Supports different modes of operation
  - Memory Mode: NVRAM extends RAM transparently
  - Application Direct Mode: Access via device, file or memory API
- Intel announced discontinuation of Optane in 2022
  - No real alternatives are available at the moment
  - Samsung and KIOXIA offer faster SSD solutions
  - Supposedly much faster than NVMe SSDs

Quiz

- How much storage bandwidth is used on average?
  1. 99 %
  2. 50 %
  3. 33 %
  4. 5 %

Analysis of a major HPC production storage system

- 99% of the time, storage BW utilization < 33% of max
- 70% of the time, storage BW utilization < 5% of max

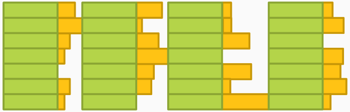


DAOS

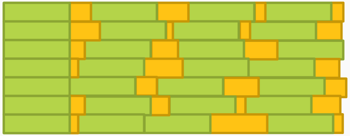
- New holistic approach for I/O
  - Distributed Application Object Storage (DAOS)
- Supports multiple storage models
  - Arrays and records are base objects
  - Objects contain arrays and records (key-array)
  - Containers consist of objects, storage pools consist of containers
- Support for versioning
  - Operations are executed in transactions
  - Transactions are persisted as epochs
- Make use of modern storage technologies

DAOS… [Brent Gorda, 2013]

- I/O is typically performed synchronously
  - Applications have to wait for slowest process, variations are normal
  - File is only consistent after all processes have finished writing



- I/O should be completely asynchronous
  - Eliminates waiting times, makes better use of resources
  - Difficult to define consistency, transactions and snapshots can be used



Outline

Parallel I/O

- Review
- Introduction and Motivation
- Storage Devices and Arrays
- File Systems
- Parallel Distributed File Systems
- Libraries
- Future Developments
- Summary

Summary

- Achieving high performance I/O is a complex task
  - Many layers: storage devices, file systems, libraries etc.
- File systems organize data and metadata
  - Modern file systems provide additional functionality
- Parallel distributed file systems allow efficient access
  - Data is distributed across multiple servers
- I/O libraries facilitate ease of use
  - Exchangeability of data is an important factor
- New technologies will make the storage stack more complex
  - Future systems will offer novel I/O approaches

References

[Bonér, 2012] Bonér, J. (2012). **Latency Numbers Every Programmer Should Know.**
https://gist.github.com/jboner/2841832.

[Brent Gorda, 2013] Brent Gorda (2013). **HPC Technologies for Big Data.**
http://www.hpcadvisorycouncil.com/events/2013/Switzerland-Workshop/Presentations/Day_2/3_Intel.pdf.

[Brent Gorda, 2016] Brent Gorda (2016). **HPC Storage Futures – A 5-Year Outlook.**
http://lustre.ornl.gov/ecosystem-2016/documents/keynotes/Gorda-Intel-keynote.pdf.

[Cburnett, 2006a] Cburnett (2006a). **RAID 1 with two disks.**
https://en.wikipedia.org/wiki/File:RAID_1.svg.

[Cburnett, 2006b] Cburnett (2006b). **RAID 5 with four disks.**
https://en.wikipedia.org/wiki/File:RAID_5.svg.

References ...

[Huang et al., 2014] Huang, J., Schwan, K., and Qureshi, M. K. (2014). **NVRAM-aware Logging in Transaction Systems.** *Proc. VLDB Endow.*, 8(4):389–400.

[Mike Vildibill, 2015] Mike Vildibill (2015). **Advanced IO Architectures.** http://storageconference.us/2015/Presentations/Vildibill.pdf.

[Seagate, 2016] Seagate (2016). **Desktop HDD.** http://www.seagate.com/www-content/datasheets/pdfs/desktop-hdd-8tbDS1770-9-1603DE-de_DE.pdf.

[Werner Fischer and Georg Schönberger, 2017] Werner Fischer and Georg Schönberger (2017). **Linux Storage Stack Diagramm.** https://www.thomas-krenn.com/de/wiki/Linux_Storage_Stack_Diagramm.

[Wikipedia, 2021] Wikipedia (2021). **Hard disk drive.** https://en.wikipedia.org/wiki/Hard_disk_drive.