I/O Benchmarking

Parallel I/O NHR Workshop

Anna Fuchs, Jannek Squar

7 May 2024

Scientific Computing
Universität Hamburg

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Who are we

- Postdoc and PhD in flight
- **Universität** Hamburg
- Parallelization on application level
- Tooling, Monitoring, Tracing
- Middleware and data formats
- Lustre
- Compression
- System benchmarking
- *Started collaboration with DKRZ this spring*

Our tasks

- Conducting I/O benchmarks for the new system
- Determining user requirements regarding I/O
- Does current system match current needs?
- What is missing now and will be needed tomorrow?
- Exploring the I/O needs of climate applications and workflows
- Today: insights into
  - methodology
  - selective benchmarks
  - topics open for discussion

I/O Requirements

- 2 different perspectives
  - User:

I/O Requirements

- 2 different perspectives
  - User: get my shit done fast

I/O Requirements

- 2 different perspectives
  - User: get my shit **done** fast
    - done - reliable, predictable, accessible..

I/O Requirements

- 2 different perspectives
  - User: get my shit done **fast** get **my shit** done fast
    - done - reliable, predictable, accessible..
    - fast - technically fast

I/O Requirements

- 2 different perspectives
  - User: get **my shit** done fast
    - done - reliable, predictable, accessible..
    - fast - technically fast
    - my shit - in a way I can work with

I/O Requirements

- 2 different perspectives
  - User: get **my shit** done fast
    - done - reliable, predictable, accessible..
    - fast - technically fast
    - my shit - in a way I can work with
    - new metric: time-to-solution

I/O Requirements

- 2 different perspectives
  - User: get **my shit** done fast
    - done - reliable, predictable, accessible..
    - fast - technically fast
    - my shit - in a way I can work with
    - new metric: time-to-solution
    - expression tool: application

I/O Requirements

- 2 different perspectives
  - User: get **my shit** done fast
    - done - reliable, predictable, accessible..
    - fast - technically fast
    - my shit - in a way I can work with
    - new metric: time-to-solution
    - expression tool: application
  - System: *what* resources do you need *when* and *how long*?
    - user mostly unable to answer
    - shouldn't be tasked with doing so

I/O Requirements

- 2 different perspectives
  - User: get **my shit** done fast
    - done - reliable, predictable, accessible..
    - fast - technically fast
    - my shit - in a way I can work with
    - new metric: time-to-solution
    - expression tool: application
  - System: *what* resources do you need *when* and *how long*?
    - user mostly unable to answer
    - shouldn't be tasked with doing so
- Needs a third perspective to mediate
- Core factors to be identified from both sides
- Alignment verification for user and system
- We combine expertise from both sides and are able to collaborate

System

- Determine the technical path from application to persistent storage
- Capture all hard- and software capabilities and their interactions

System

- Determine the technical path from application to persistent storage
- Capture all hard- and software capabilities and their interactions
- Simple specs
  - static hardware in number, data rates, IOPS, volume, ...
  - clients, servers, disks, networks

System

- Determine the technical path from application to persistent storage
- Capture all hard- and software capabilities and their interactions
- Simple specs
  - static hardware in number, data rates, IOPS, volume, ...
  - clients, servers, disks, networks
- Advanced specs
  - system software configuration
  - Lustre threading, RPCs, cache buffers, aggregations, etc.
  - Linux limitations with Lustre $\rightarrow$ single stream <1-3 GiB/s
  - $\rightarrow$ determine static bottlenecks

System

- Determine the technical path from application to persistent storage
- Capture all hard- and software capabilities and their interactions
- Simple specs
  - static hardware in number, data rates, IOPS, volume, ...
  - clients, servers, disks, networks
- Advanced specs
  - system software configuration
  - Lustre threading, RPCs, cache buffers, aggregations, etc.
  - Linux limitations with Lustre $\rightarrow$ single stream <1-3 GiB/s
  - $\rightarrow$ determine static bottlenecks
- Superior specs
  - interaction of asynchronous layers
  - $\rightarrow$ determine dynamic bottlenecks

System

- Determine the technical path from application to persistent storage
- Capture all hard- and software capabilities and their interactions
- Simple specs
  - static hardware in number, data rates, IOPS, volume, ...
  - clients, servers, disks, networks
- Advanced specs
  - system software configuration
  - Lustre threading, RPCs, cache buffers, aggregations, etc.
  - Linux limitations with Lustre $\rightarrow$ single stream <1-3 GiB/s
  - $\rightarrow$ determine static bottlenecks
- Superior specs
  - interaction of asynchronous layers
  - $\rightarrow$ determine dynamic bottlenecks
- Real world
  - Dynamic load $\rightarrow$?

The Complexity of Compute vs. I/O

- Compute is complex, but manageable with expertise
- Compute shares asynchronous layers
- Performance is relatively reproducible, until it's not

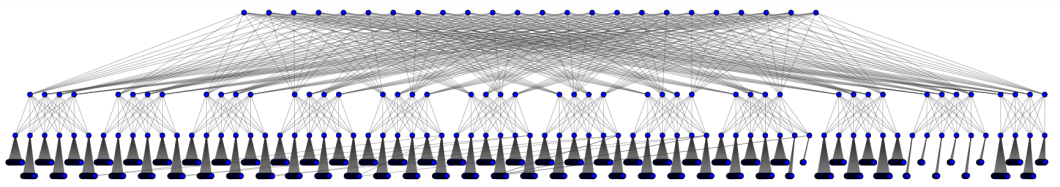
The Complexity of Compute vs. I/O

- Compute is complex, but manageable with expertise
- Compute shares asynchronous layers
- Performance is relatively reproducible, until it's not
- Network $\rightarrow$ A shared resource
- Unpredictable load and beyond direct control
- Network issues directly notable and cause immediate frustration

I/O Complexity

- I/O predictability begins with frustration
- Increased complexity, unpredictability, and lack of control
- Every component is shared
- Client: competing for resources within the application (less CPU, more network)
- Network: May or may not be shared with other jobs or partly
- Server: May or may not be shared with other jobs pr party
- Disk: May or may not be shared with other jobs or party
- Combinatorics: Exploring the vast possibilities each with unpredictable load

Transparent Network



I/O is unsexy

- Millions of paths with unforeseen loads
  - same solutions may or may not work intermittently
  - spoiler: isolated benchmark runs are useless
- What do we do about it?
  1. Take it or leave it
     - it's too complex, just accept it and make the best of it
     - 1 GB/s per 700 compute nodes
  2. University committed to research
     - we aim to figure it out precisely

Application View

- System view $\rightarrow$ application view (its requirements)
- Non-exclusive strategies:

Application View

- System view $\rightarrow$ application view (its requirements)
- Non-exclusive strategies:
  1. User Stories
     - Similar to the system, seek and understand paths: user idea $\rightarrow$ implementation
     - Individual approach: talk with user
     - Requires significant effort and communication!
     - But: previous knowledge already available

     $\rightarrow$ Accurate knowledge

Application View

- System view $\rightarrow$ application view (its requirements)
- Non-exclusive strategies:
  1. User Stories
     - Similar to the system, seek and understand paths: user idea $\rightarrow$ implementation
     - Individual approach: talk with user
     - Requires significant effort and communication!
     - But: previous knowledge already available

     $\rightarrow$ Accurate knowledge
  2. Monitoring
     - Technical approach: user $\equiv$ black box
     - Focus on isolated local individual applications or global system behaviour
     - Static analysis (SLURM statistics)
     - Sampling/Tracing

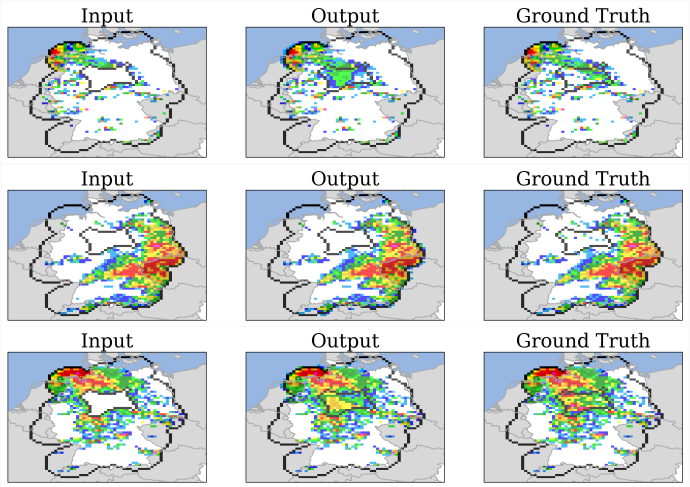
Gathering Data: Sampling

- Advantages: easy, low-impact on performance
- Disadvantages: precision, time-related associations (timeline)

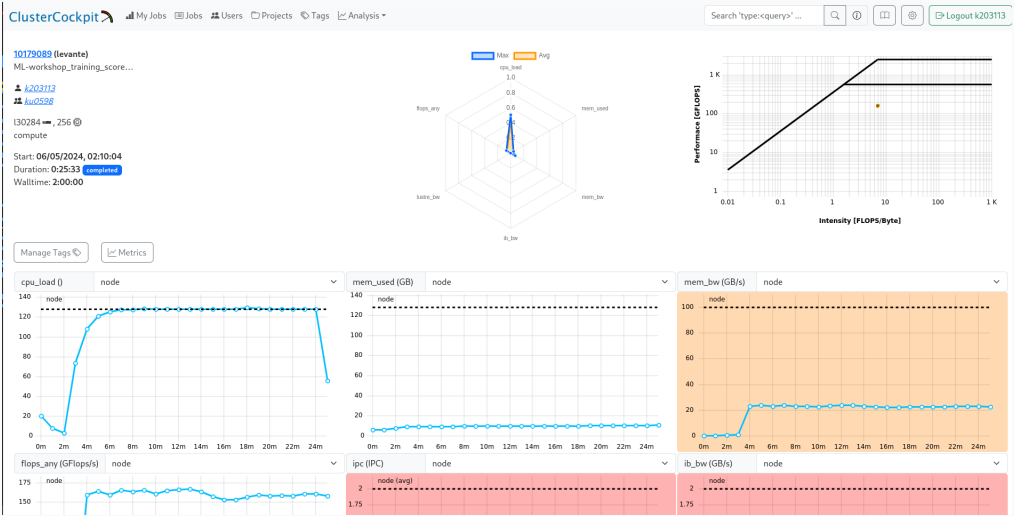
Gathering Data: Sampling

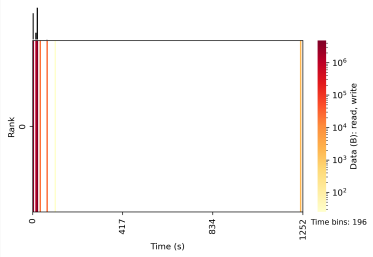
- Advantages: easy, low-impact on performance
- Disadvantages: precision, time-related associations (timeline)
- Some tools
  - ClusterCockpit
  - Darshan
  - Scorep Profile (+ Cube)

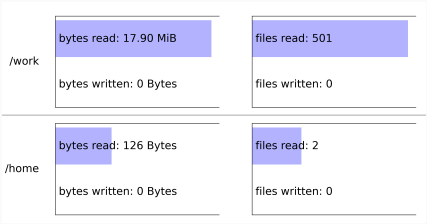
ML example: crai RADOLAN[1]

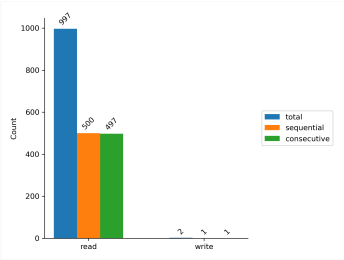


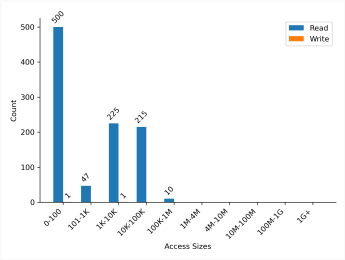
[1] Example provided by Johannes Meuer and Christopher Kadow (DA)


ClusterCockpit
My Jobs
Jobs
Users
Projects
Tags
Analysis
Search 'type:<query>' ...
Logout k203113
10179089 (levante)
ML-workshop_training_score...
k203113
ku0598
I30284 , 256
compute
Start: 06/05/2024, 02:10:04
Duration: 0:25:33 completed
Walltime: 2:00:00
Max
Avg
cpu_load
mem_used
mem_bw
ib_bw
flops_any
Performance [GFLOPS]
Intensity [FLOPS/Byte]
Manage Tags
Metrics
cpu_load ()
node
mem_used (GB)
mem_bw (GB/s)
flops_any (GFlops/s)
ipc (IPC)
ib_bw (GB/s)
node (avg)

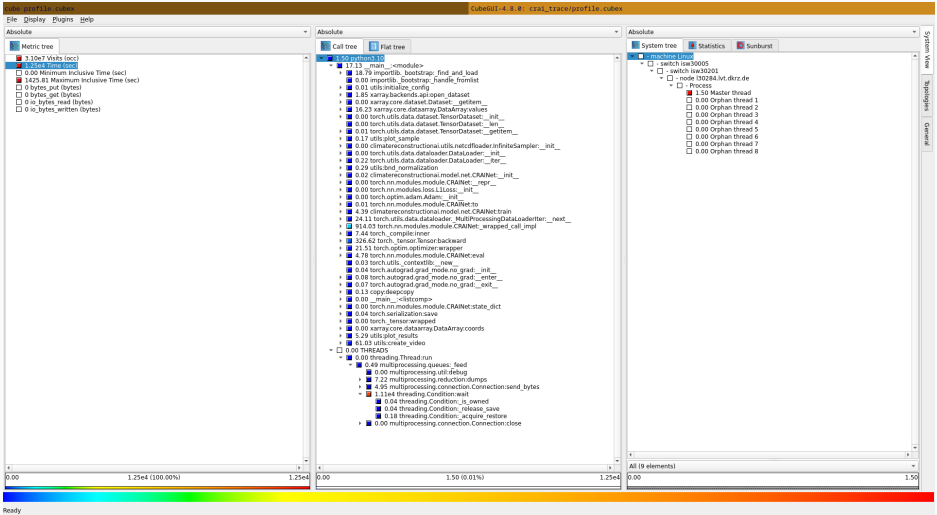

Rank
0
0
417
834
1252
Time (s)
Time bins: 196
Data (B): read, write
10^6
10^5
10^4
10^3
10^2


/work
bytes read: 17.90 MiB
bytes written: 0 Bytes
files read: 501
files written: 0
/home
bytes read: 126 Bytes
bytes written: 0 Bytes
files read: 2
files written: 0


Count
1000
800
600
400
200
0
997
500
497
2
1
1
read
write
total
sequential
consecutive


Count
500
400
300
200
100
0
500
1
47
225
1
215
10
0-100
101-1K
1K-10K
10K-100K
100K-1M
1M-4M
4M-10M
10M-100M
100M-1G
1G+
Access Sizes
Read
Write

Scorep Profile



Gathering Data: Tracing

- Advantage: Precision, timeline
- Disadvantage: Heavy impact, recompilation(?) $\rightarrow$ only on demand

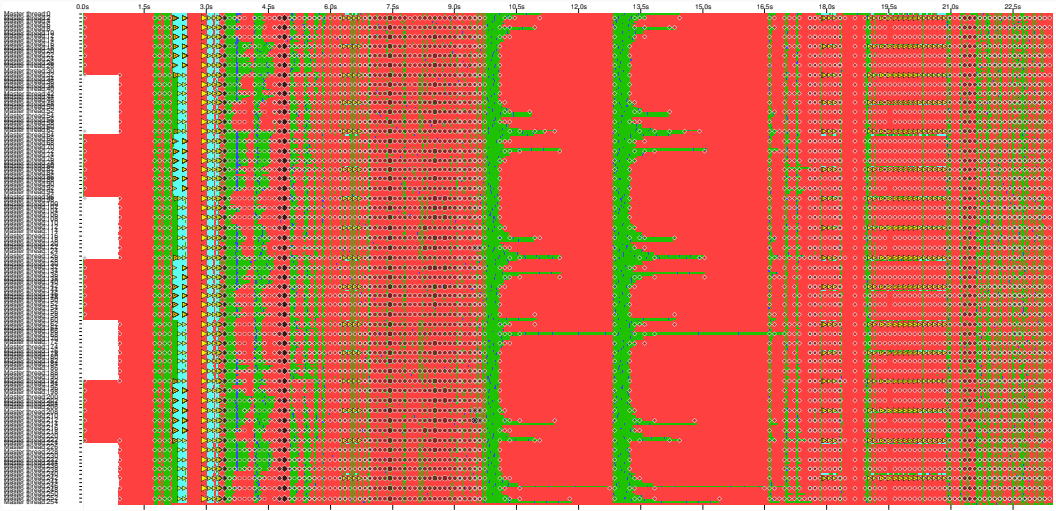
Gathering Data: Tracing

- Advantage: Precision, timeline
- Disadvantage: Heavy impact, recompilation(?) $\rightarrow$ only on demand
- Some tools
  - Tau
  - Scorep Trace + Vampir

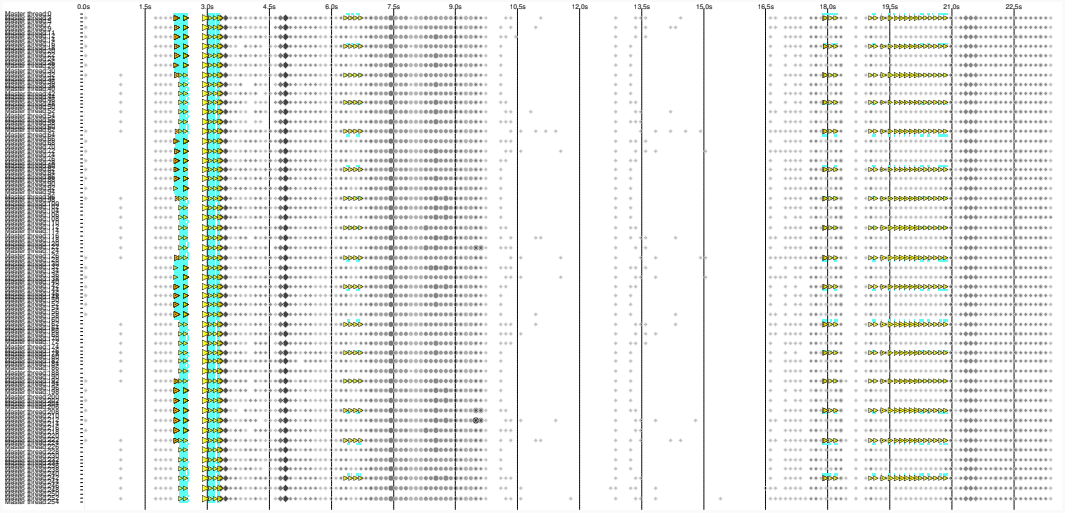
Gathering Data: Tracing

- Advantage: Precision, timeline
- Disadvantage: Heavy impact, recompilation(?) $\rightarrow$ only on demand
- Some tools
  - Tau
  - Scorep Trace + Vampir
- Example: ICON test `exp.test_nwp_R02B04_R02B05_nest`
  - Runtime $\approx$ 1 minute
  - 256 processes
  - Trace size without filter: 37GB (!)

ICON example



ICON example: `exp.test_nwp_R02B04_R02B05_nest`



I/O pattern

- Rarely just I/O (except copying or backups)
- Need further characterization of application behavior
- Helps to know CPU bound/portion, network bound/potion, ...
- With this knowledge, optimize for primary bottlenecks
  - one bottleneck may shift burden to another path
  - example: compressing large data to reduce volume
  - by means of creating in billions of files, overwhelming MDS
  - data becomes unmanageable

Network congestion – toxic I/O on Levante

Reference job (baseline): numio benchmark (PDE solver+); 128 procs 2 Nodes, 1 per **cell 2** and **9**

- computation + halo-exchange (pairwise 190 kiB) , 35k iterations
- **Coll**: + MPI collectives 265 KiB Allgrather every 70 iterations
- **Sync/async**: MPI-IO 1 GiB every x iterations

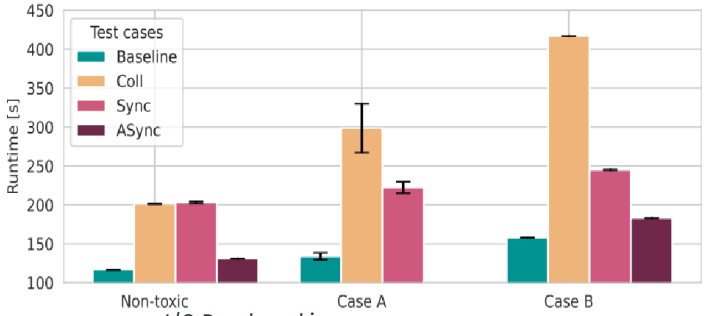
Background job: toxic IO (**A** and **B**), read file per process, md5sum per file, 1 node, **cell 9**

A:

- 3000 files
- 100 MB - 7 GB each
- stripe count 160

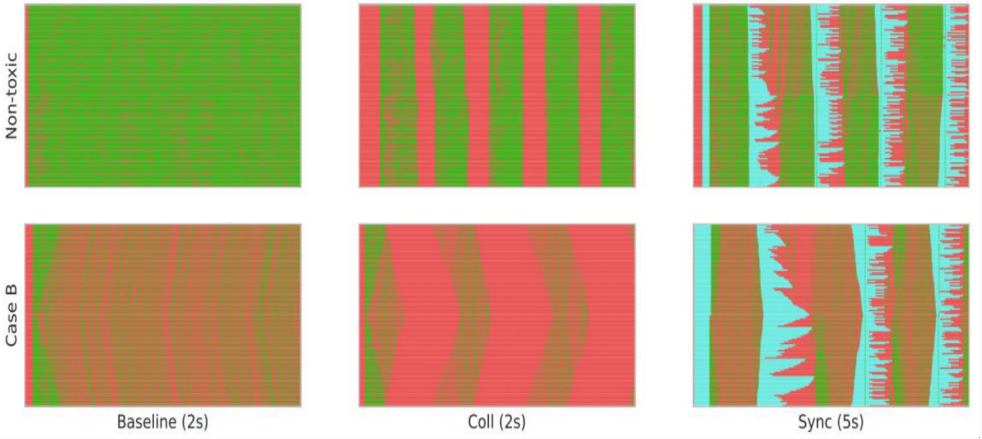
B:

- 365 files
- 167 GB each
- stripe count 16



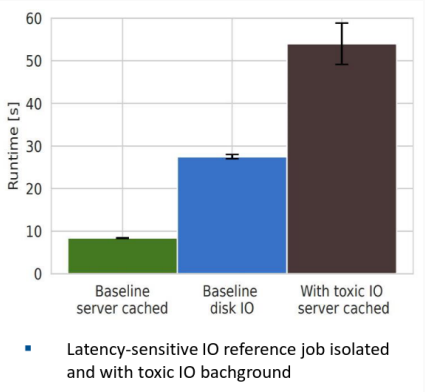
Network congestion – toxic I/O traces

- Latency-sensitive jobs susceptible to delays -> idle waves
- Application's IO not latency-sensitive, therefore no substantial delay



Network congestion – small I/O

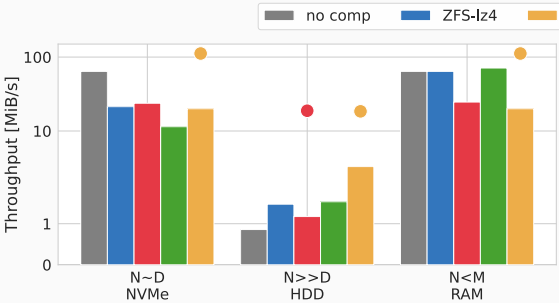
- Virtual lanes: can't kill MPI with I/O anymore
- What about kill I/O with I/O?



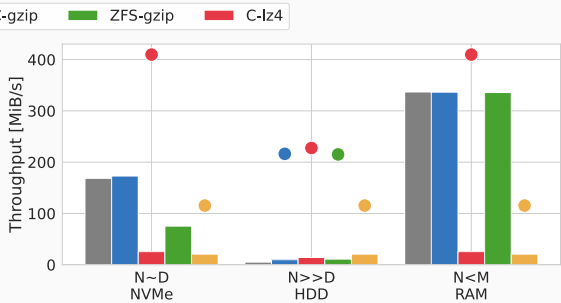
Read and checksum 4 byte in every 1 MiB block - LATENCY!

I/O suicide

- Sparse read SINGLE stream, Lustre+ZFS, 1 OST (+compression)
- IPoIB 2.6 GB/s, NVMe 2-5 GB/s, HDD 235 MB/s
- Prefetching
  - Absend for 1 MiB pattern
  - Overwhelming for HDD for 128 KiB pattern (ZFS record size)

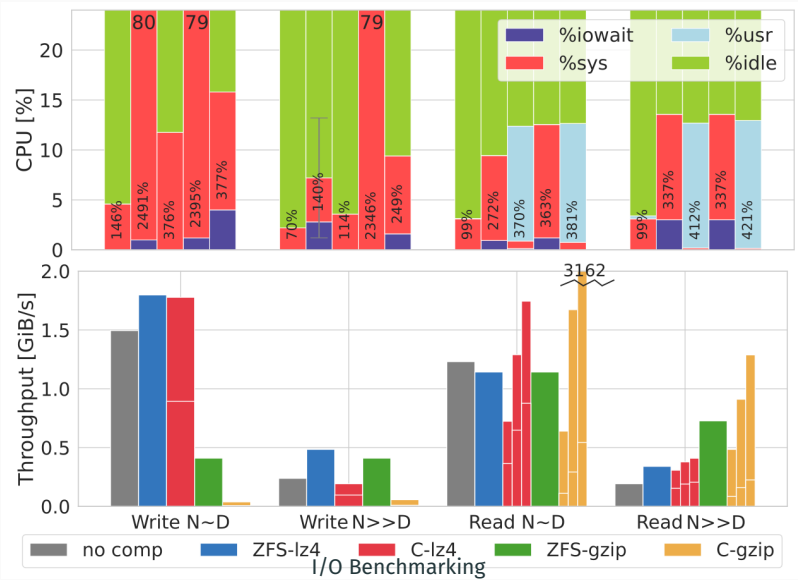


read 4 KiB blocks every 1MiB

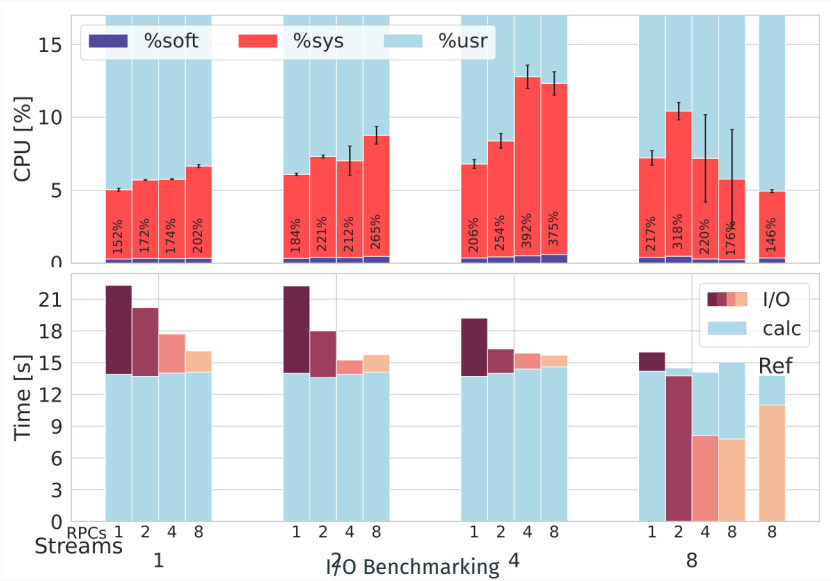


every 128 KiB

Bulk I/O with compression - SINGLE stream, L+ZFS, IPoIB



Async compute and parallel write with compression (lz4), L+ZFS



Summary

- Tracking user/application behavior is fundamental
- I/O is hard to do correctly
- Asynchrony (asynchronocity?): blessing and curse
- Hard to predict, hard to tune to bleeding edge
- Not bleeding edge should not be that hard
- Application's pattern has the absolute power to kill any performance

I/O read pattern