Collective I/O Tuning Using Analytical and Machine Learning Models

Florin Isaila

ANL & University Carlos III

Prasanna Balaprakash, Paul Hoveland, Dries Kimpe,

Rob Latham, Rob Ross, Stefan Wild

ANL

Parallel I/O tuning

- Huge parameter space of the storage I/O software stack
- Domain knowledge is increasingly harder: software and hardware complexity

Application

High-Level I/O Library — Block alignment, chunking, etc.

I/O Middleware — Collective/independent I/O, data sieving, collective buffer size, number of aggregators, etc.

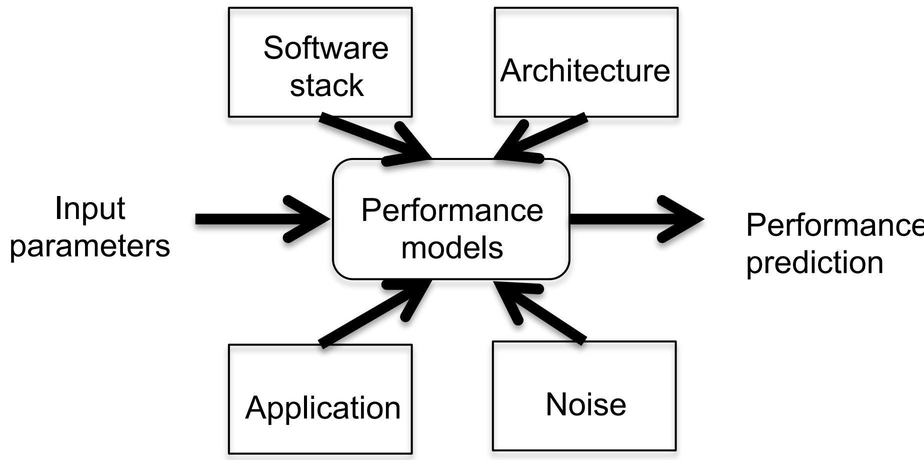
I/O Forwarding

Parallel File Systems — Striping unit, striping factor, striping layout, caching, prefetching, etc.

Storage drivers

Parallel I/O autotuning approaches

- Model based tuning
  - Analytical: Extensive domain knowledge required: software stack, architectural characteristics
  - Machine learning [Kumar2013, Yu2012]
- Search-based tuning
  - Genetic algorithms [Bezhad2013]
  - Simulated annealing [Chen2000]
- Hybrid [Bezhad2014, Bezhad2015]

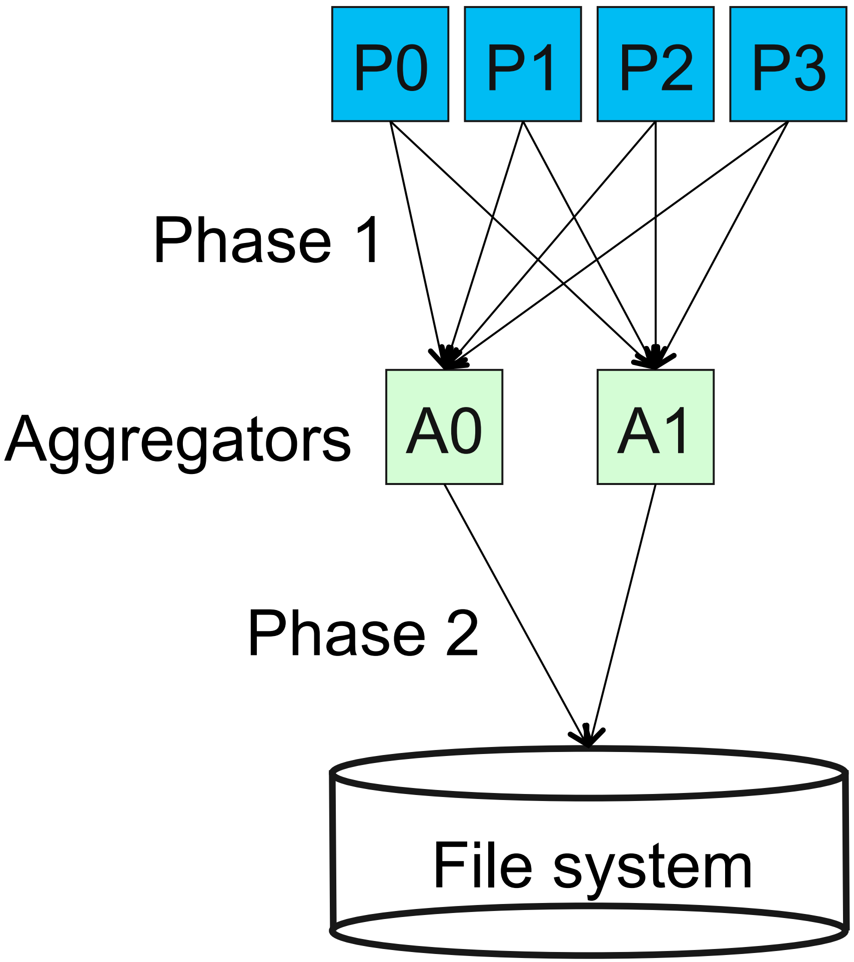


This work

- Model-based tuning of two-phase-I/O, the most popular collective I/O implementation from ROMIO
- Combination of analytical and machine learning models

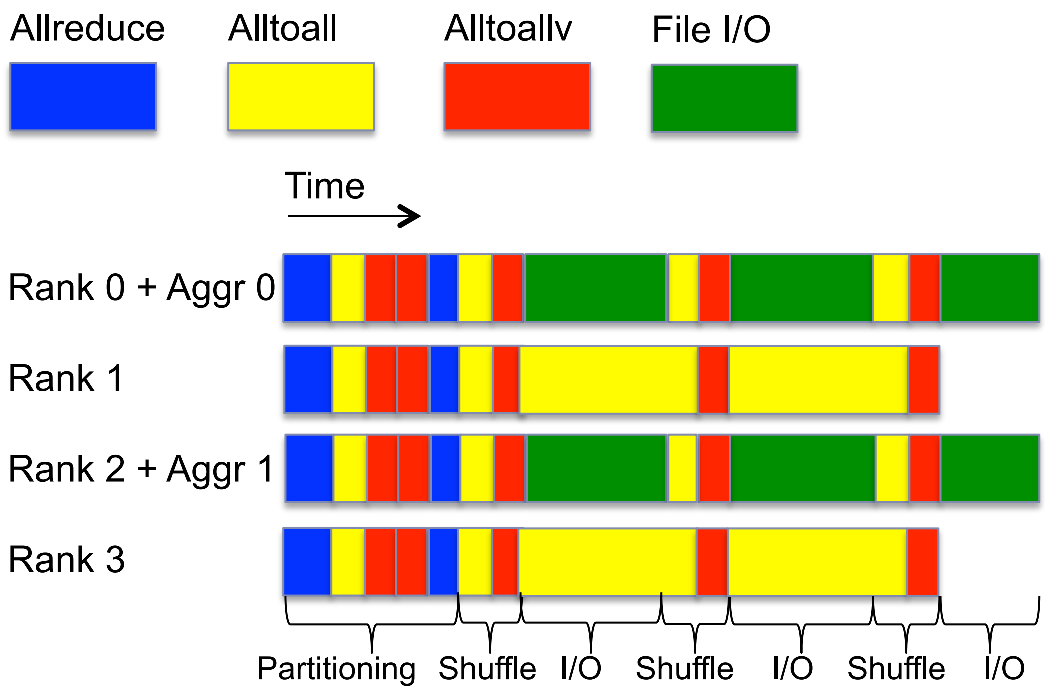
Collective I/O

- Addresses the poor FS performance and scalability
- N processes collectively write or read to a file
- Two-phase I/O write
  - Computation and communication for mapping writes to the file domain
  - Communication for sending data to aggregators
  - Storage I/O for storing the data to the file system



Anatomy of collective I/O

- Several phases if aggregate buffer size < aggregate access size
- A subset of application processes are aggregators

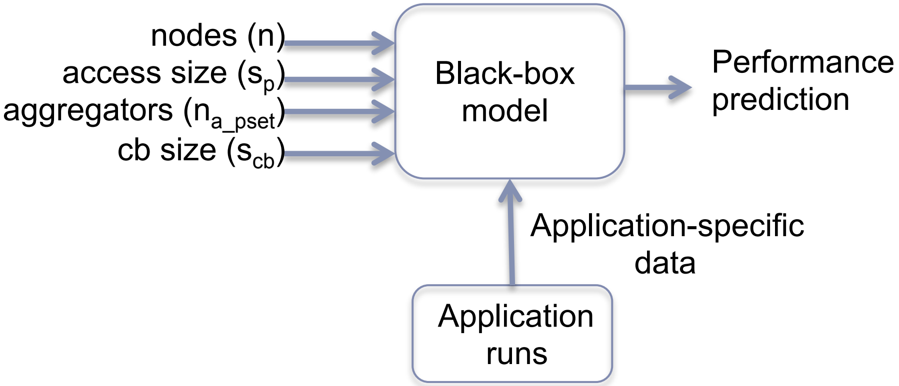


Black box model

4 parameters

- n: number of nodes
- s: access size
- n_a: number of aggregators
- s_{cb}: collective buffer size

Goal: tune n_a and s_{cb}

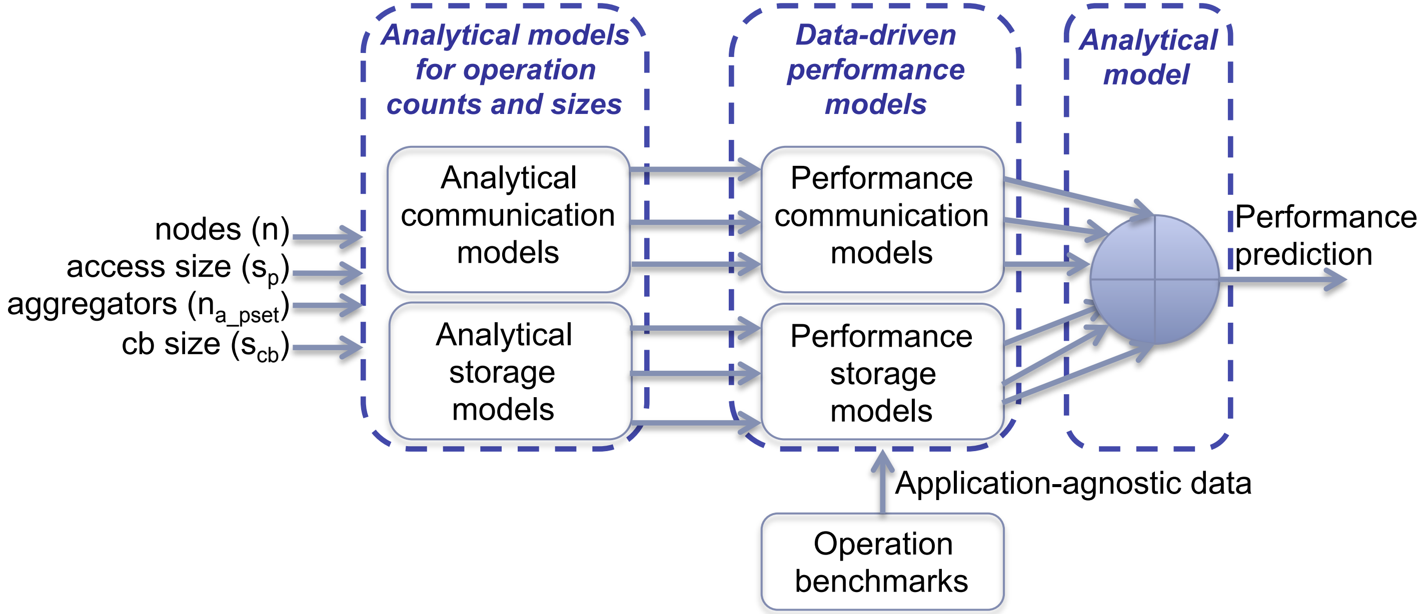


Hybrid model

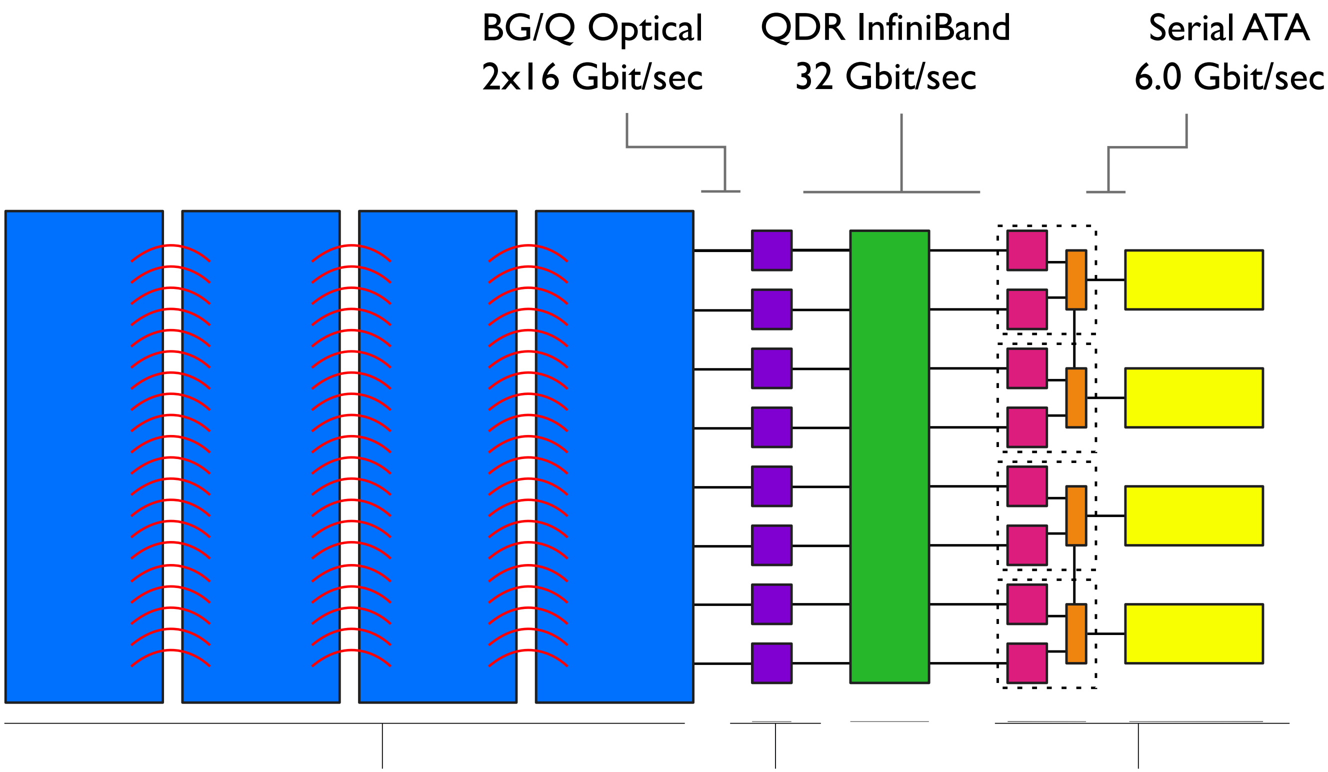
4 parameters

- n: number of nodes
- s: access size
- n_a: number of aggregators
- s_{cb}: collective buffer size

Goal: tune n_a and s_{cb}



Vesta Blue Gene/Q system at ANL



2048 compute nodes 1.6 GHz 16 cores 16GB RAM

5D torus network interconnecting compute nodes

1 I/O node per 32 compute nodes

Client (I/O node) cache: 4GB

GPFS 3.5

- Data: 40 NSD SATA drives
- Max disk throughput: 250 MB/s (cache 1000 MB/s)
- Block size: 8MB

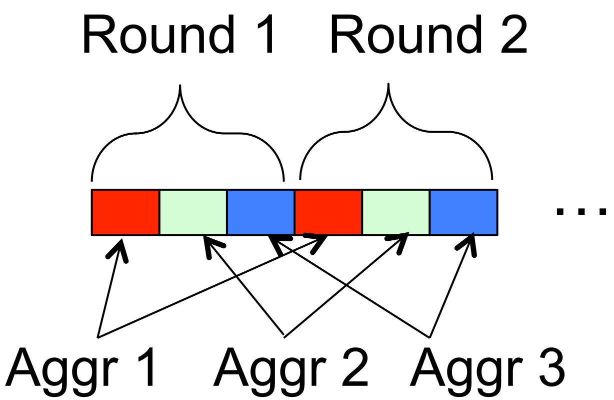
Experimental evaluation

- IOR benchmark: N processes concurrently write and non-overlapping region to the file system through MPI-IO
- MPICH 3.1
- Vesta Blue Gene/Q
  - 2048 compute nodes 1.6 GHz 16 cores 16GB RAM
  - 5D torus network interconnecting compute nodes
  - 1 I/O node per 32 compute nodes
  - Client (I/O node) cache: 4GB
  - GPFS 3.5: Block size: 8MB, 40 NSD SATA data drives (Max throughput: 250 MB/s)
- Benchmark for performance models: ALCF MPI benchmark

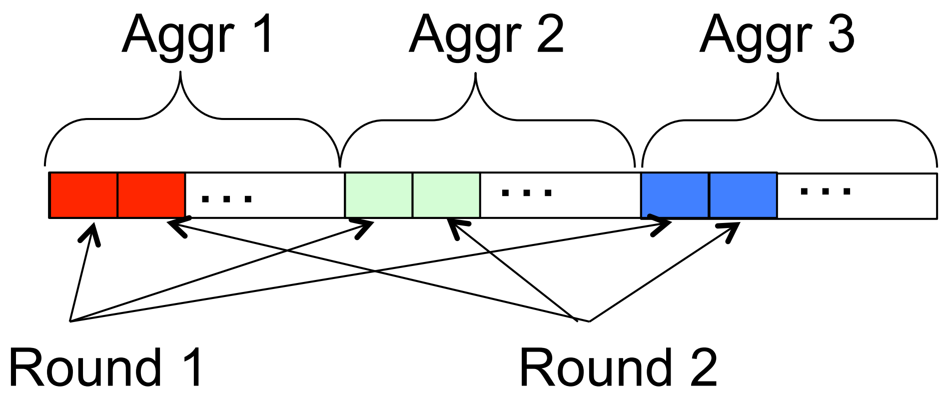
Machine learning model

- Black box models and performance models
  - linear regression, neural networks, support vector machines, random forests, and cubist
  - Selected the model with best RMSE and R^2
- Data set:
  - Black box model: 297 points
    - Processes: 2048 (128 nodes on 16 cores), 4196, and 8392
    - Transfer sizes/core (MB): 1, 2, 4, 8,16, 32, 64, 96, 128, 192, 256
    - Collective buffer size (MB): 8, 16 (default), 32
    - Number of aggregators per 128 nodes: 40, 136, 520 (default)
  - Performance models
    - Alltoall: : 51 points for 2,048, 4,096, and 8,192 ranks and for message sizes between 1 byte and 256KB.
    - Alltoallv: 1,044 points for distributing message sizes between 1 byte and 64 MB (in powers of 2) for subsets of 2,048, 4,096, and 8,192 ranks.
    - Allreduce: 57 points for 2,048, 4,096, and 8,192 ranks and for message sizes between 4 bytes and 1 MB.
    - POSIX: 567 points for various sizes and various subsets of 2,048, 4,096, and 8,192 ranks.

Challenge 1: file locking

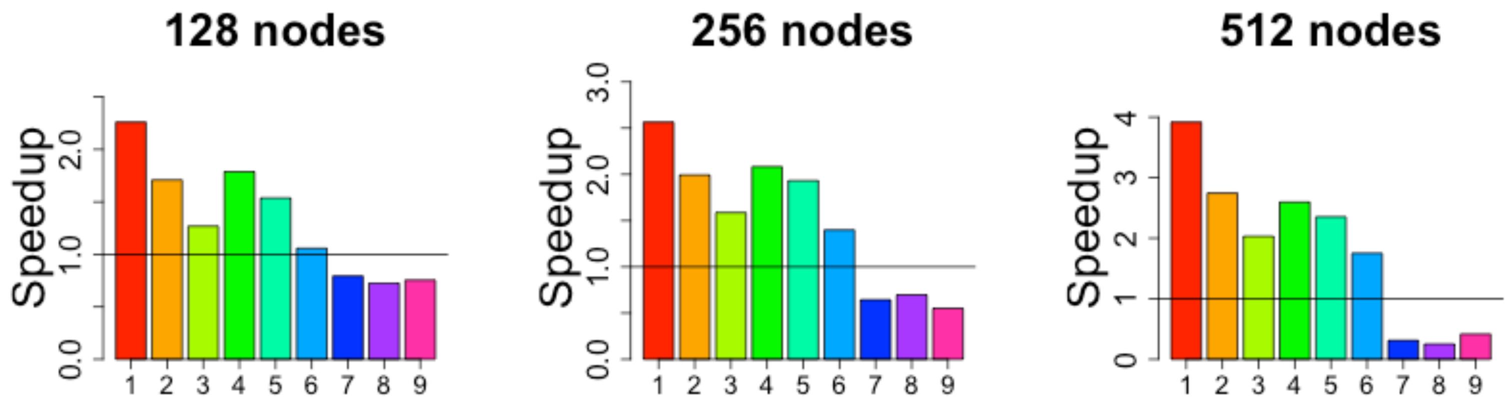


File locking domain repartitioned each round at block granularity



File locking domain partitioned only in the first round at multiple of blocks granularity

Challenge 1: file locking

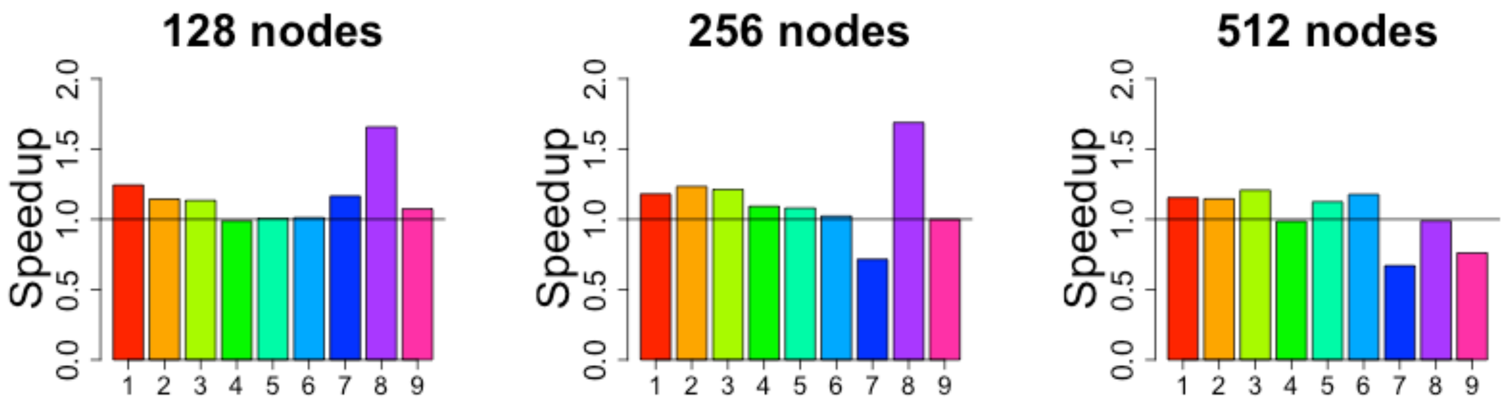


Write size (MB): 8, 16 (default), 32
Number of writing processes per pset: 4, 16, 64 (default)

Depicted: ratio between write throughput without locking and write throughput including locking.

Conclusion: different performance models required

Challenge 2: Topology-aware aggregator placement



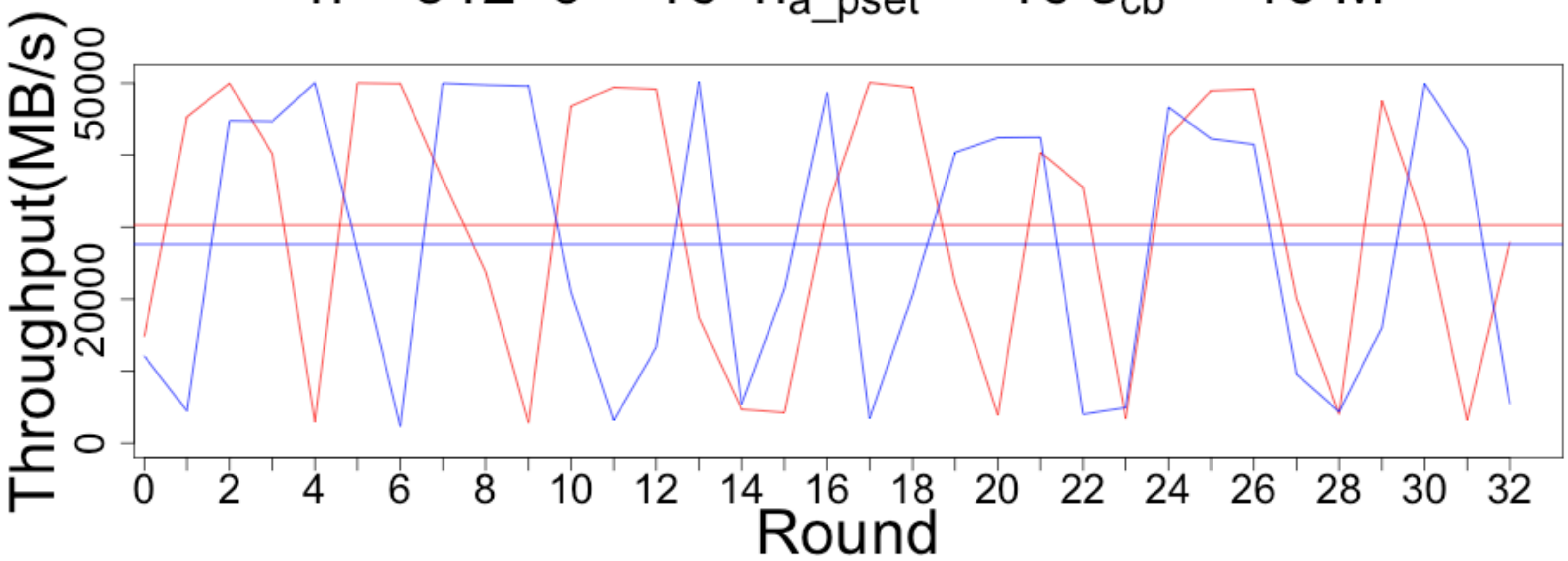
Write size (MB): 8, 16 (default), 32
Number of writing processes per pset: 4, 16, 64 (default)

Depicted: ratio between topology-aware write throughput and random placement write throughput.

Conclusion: test data has to include topology-aware results.

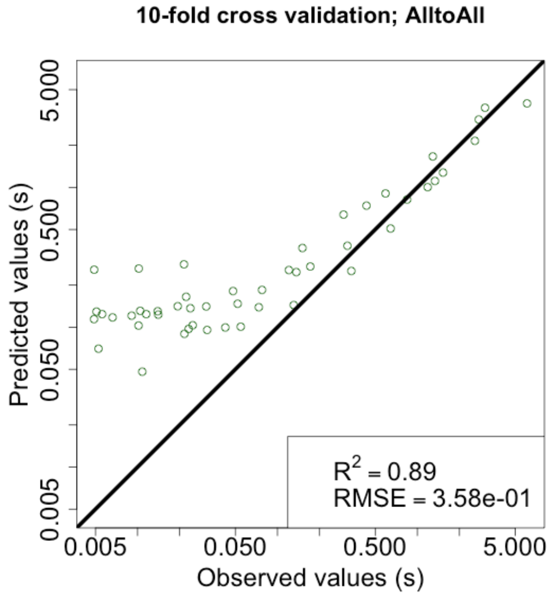
Challenge 3: file system performance variation

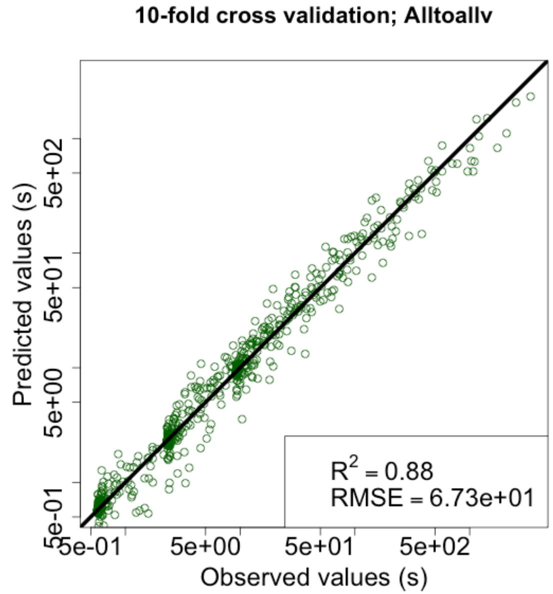
$n = 512 \quad c = 16 \quad n_{a\_pset} = 16 \quad s_{cb} = 16$ M

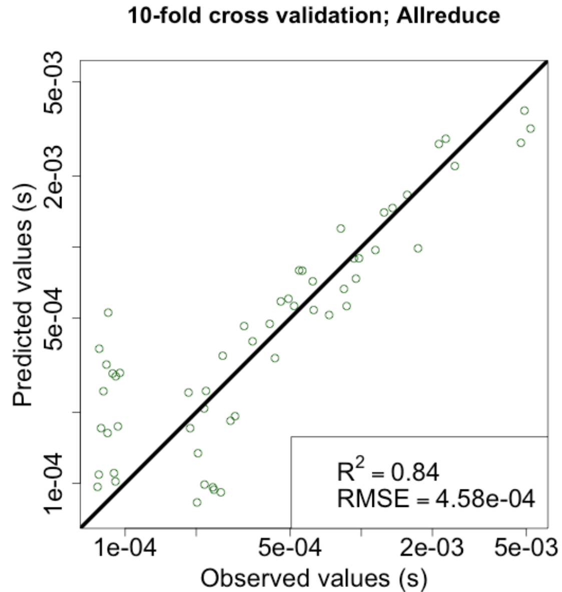


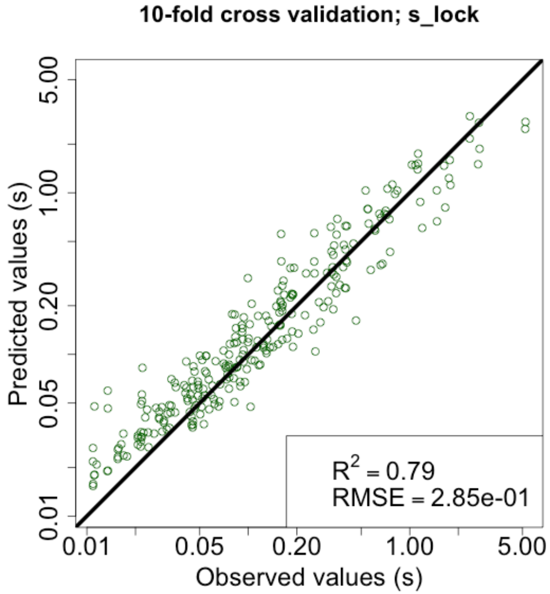
- Main reasons:
  - State of the cache
  - Write back
  - Interference with other applications
- Simple analytical models do not work
- Conclusion: average a large number of rounds to amortize effects

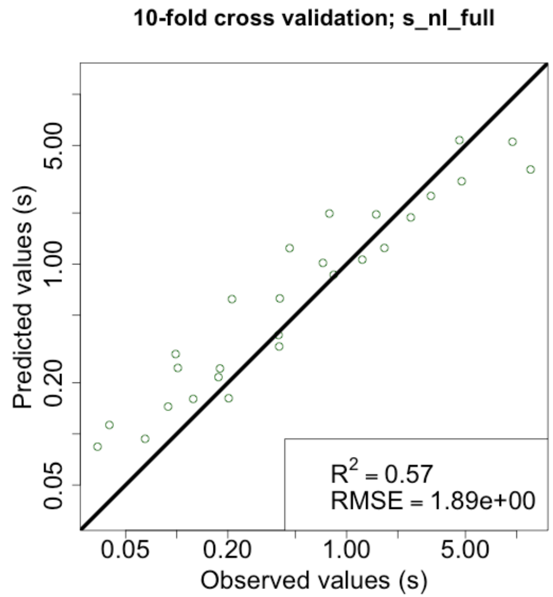
Cross-validation results for model components

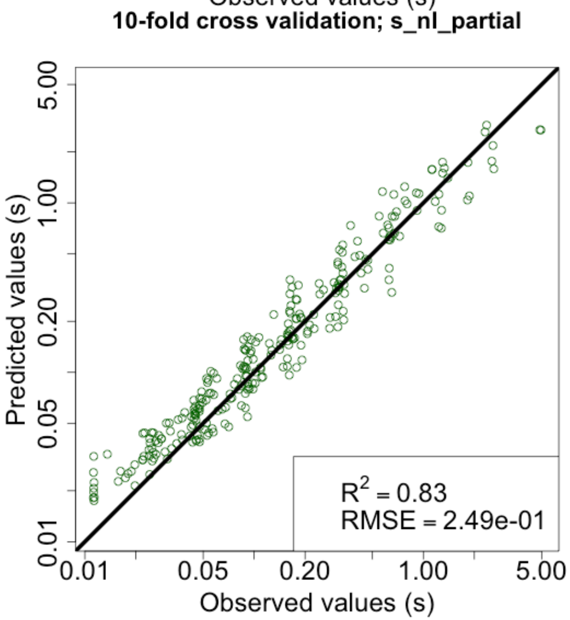




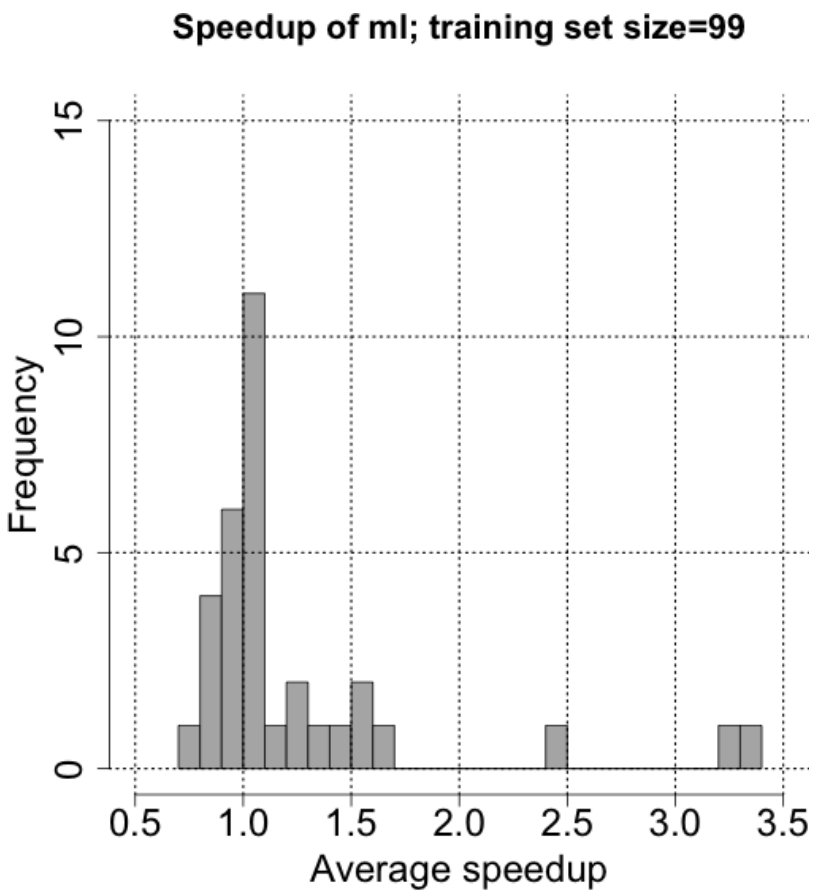


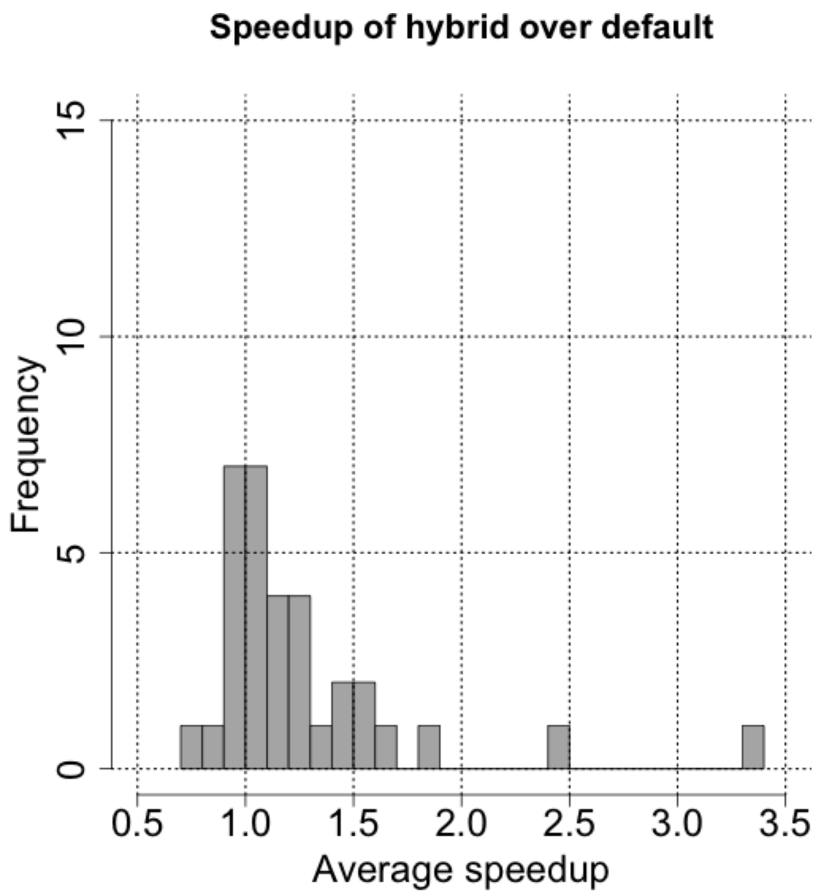




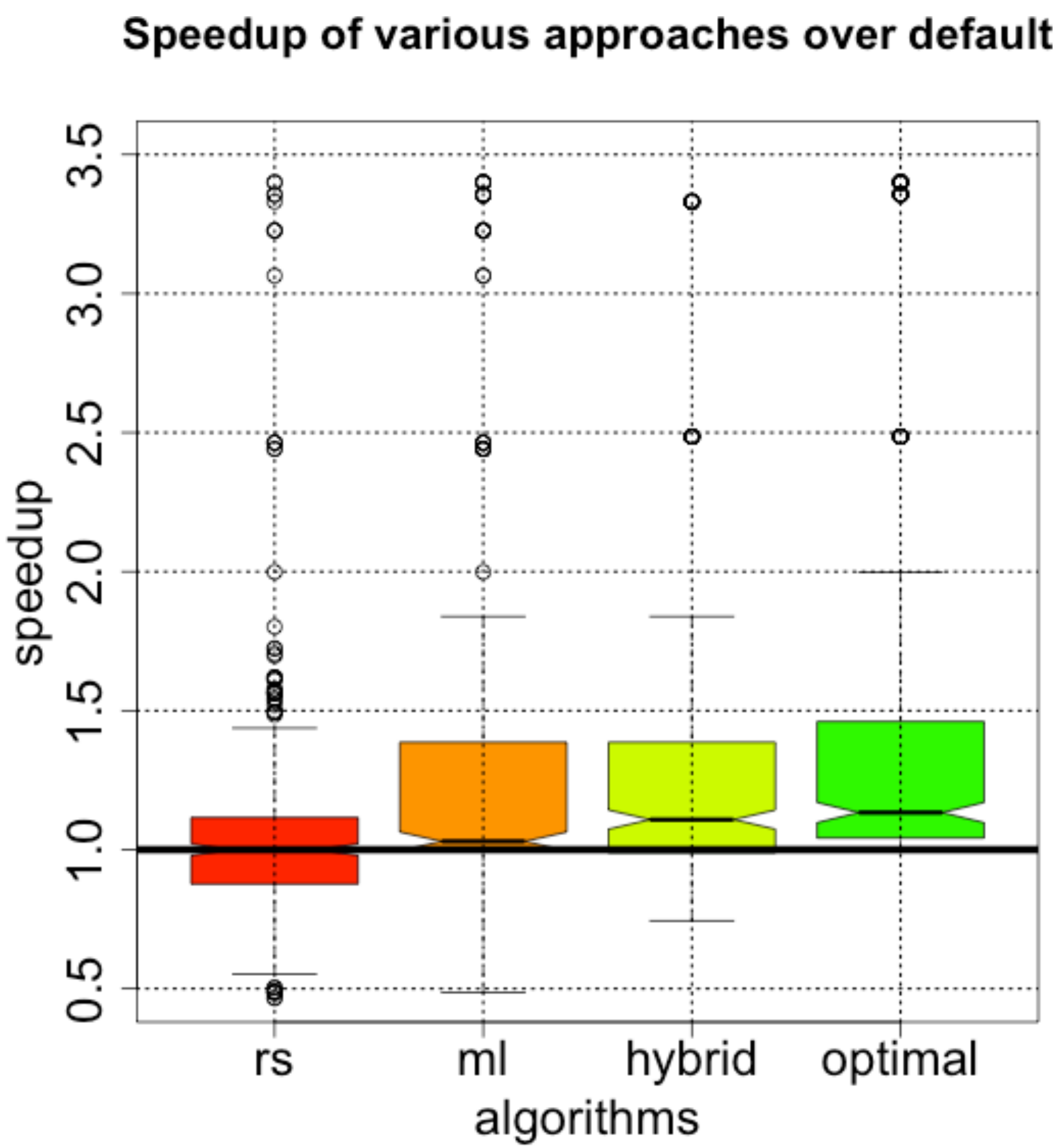


Comparison between black-box and hybrid models

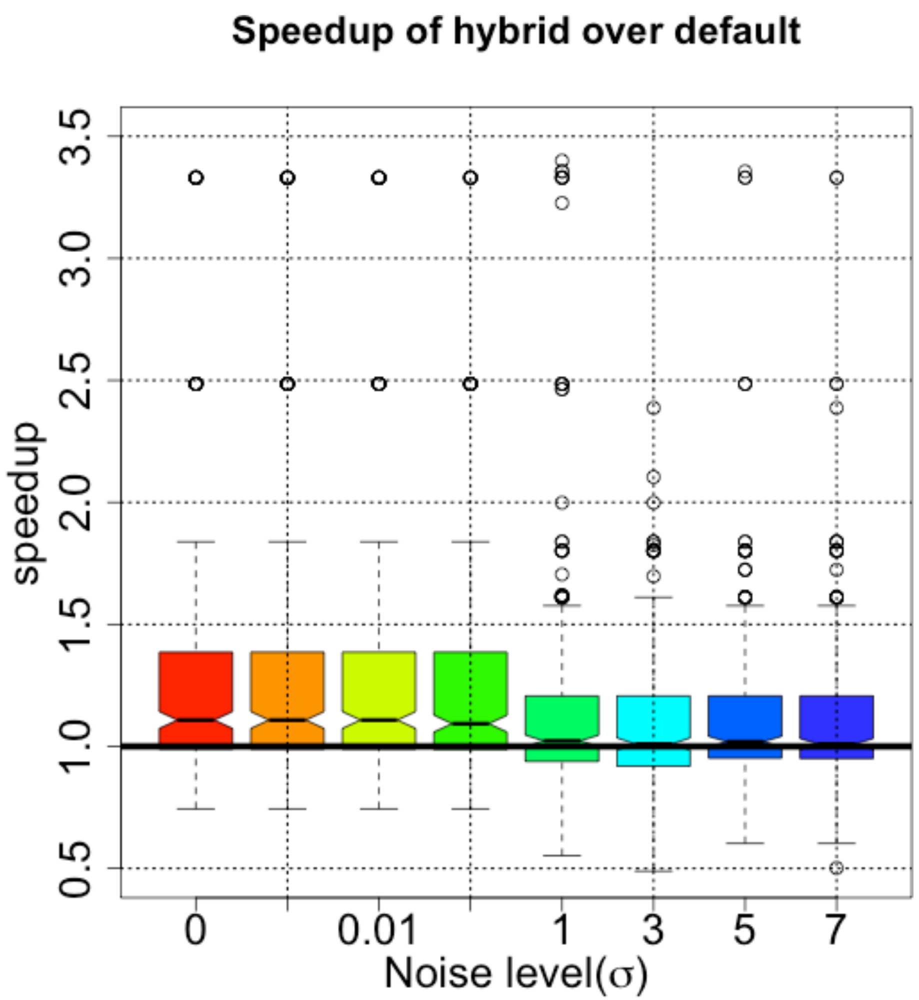


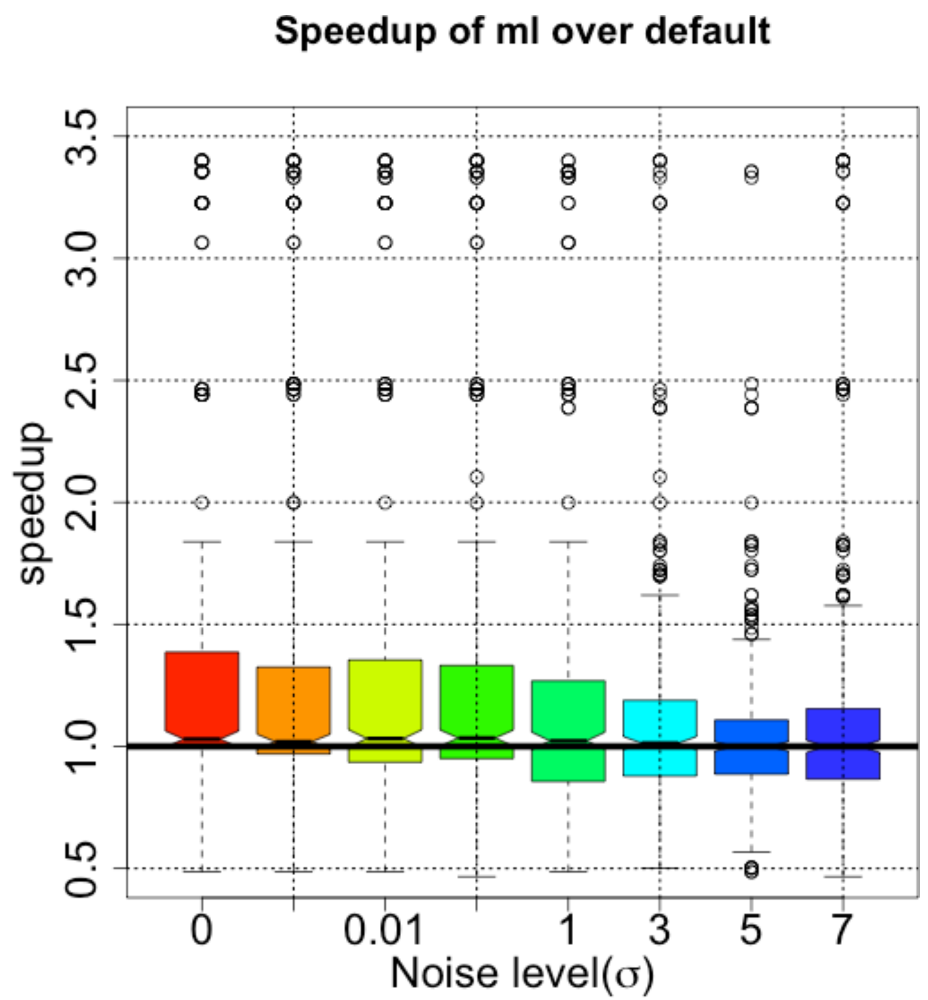


Comparison among various approaches



Impact of noise





Conclusions

- Automatic parameter configuration
  - Machine learning and hybrid models approaches outperform the default values in most cases
  - Hybrid models higher robustness to noise than pure machine learning
  - Hybrid model does not require application reruns
- Factors that limit efficiency of the I/O stack optimization
  - POSIX consistency semantics: File locking
  - File system noise
  - The lack of information about the state of storage hierarchy (e.g. cached versus non-cached)
  - Performance predictability needs to improve

Thank you