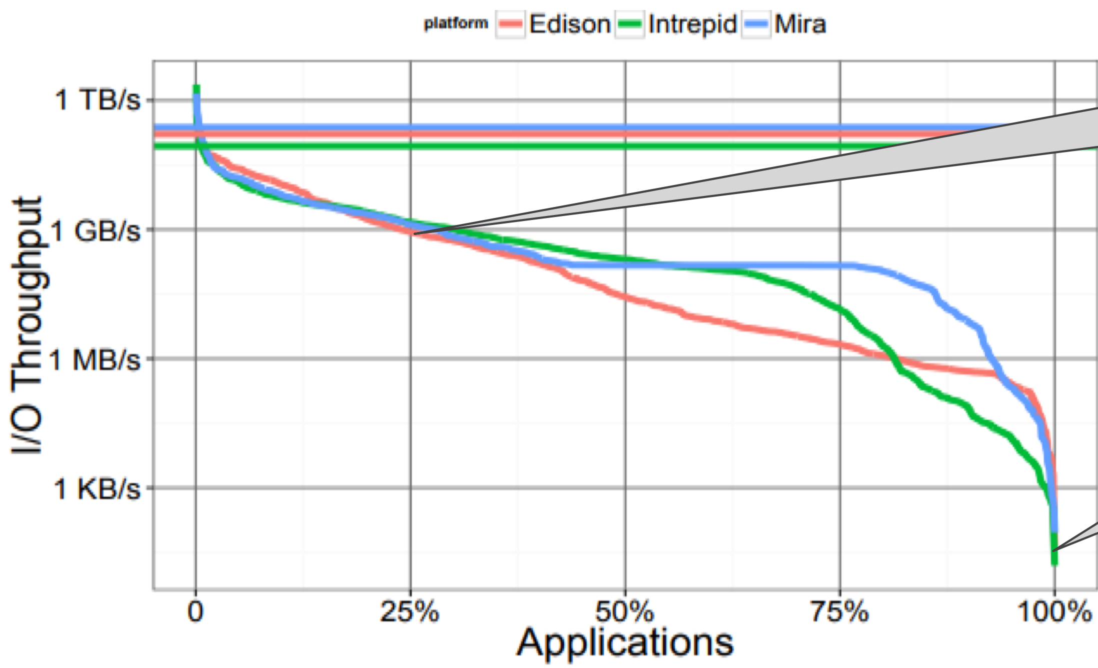
Active Learning-based Automatic Tuning and Prediction of Parallel I/O Performance

Megha Agarwal
Divyansh Singhvi
Preeti Malakar
Suren Byna

Indian Institute of Technology Kanpur, India
Lawrence Berkeley Laboratory, USA

PDSW @ SC'19
November 18, 2019

I/O Performance Statistics



75% of applications achieve less than 1GB/s I/O throughput

A few applications achieve less than 1% of I/O throughput capacity of file systems

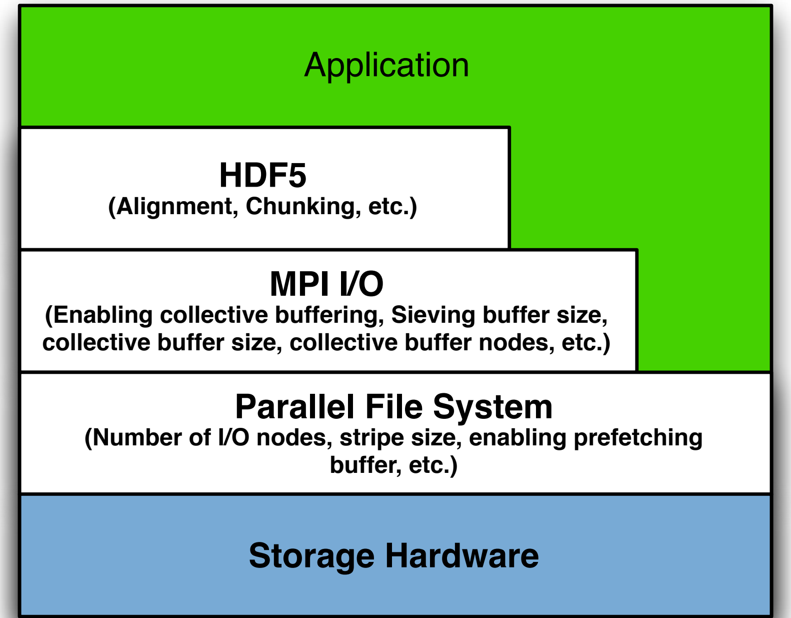
Source: Huong Luu, et al., “A Multiplatform Study of I/O Behavior on Peta-scale Supercomputers”. HPDC '15

Parallel I/O – Challenges

- Exponential growth in compute rates as compared to I/O bandwidths
- Depends on interaction of multiple layers of parallel I/O stack (I/O libraries, MPI-IO middleware, and file system)
- Each layer of I/O stack has many tunable parameters
- I/O parameters are application-dependent

A typical HPC application developer (expert in their scientific domain) resorts to default parameters ☹

Parallel I/O stack – Complexity

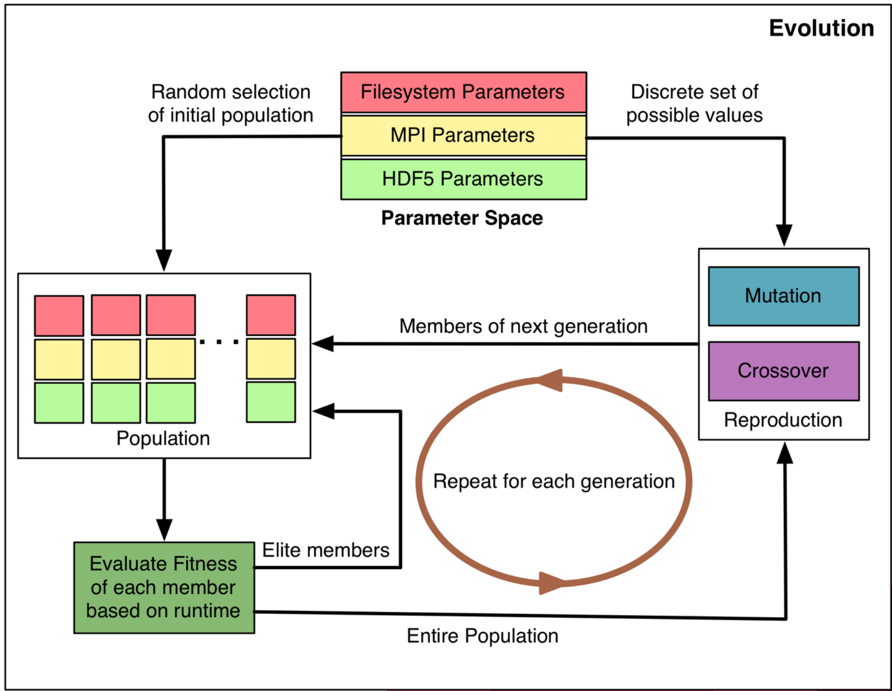


Tunable parameters: cb_nodes, cb_buffer_size, …

Tunable parameters: stripe size, stripe count, …

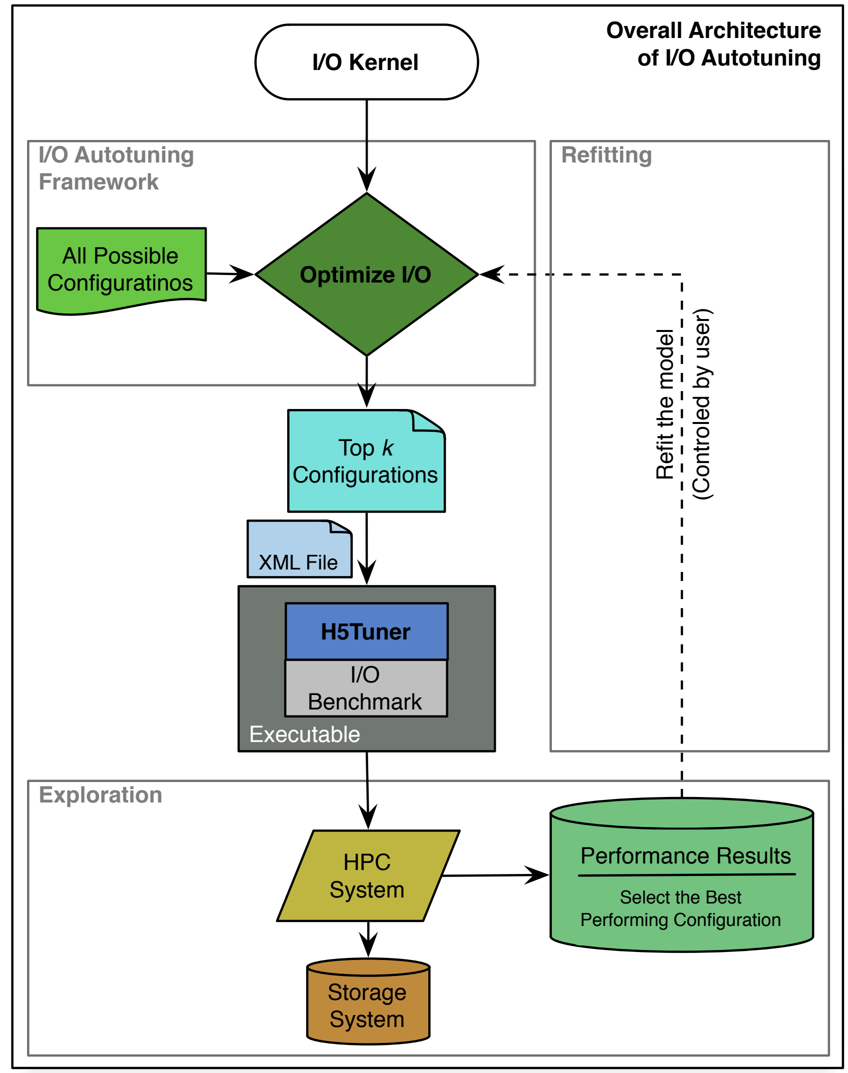
Prior Work

- Heuristic-based search with a genetic algorithm to tune I/O performance
- Analytical models
  - Disk arrays to approximate their utilization, response time, and throughput
  - Application-specific models
- Herbein et al. use a statistical model, called surrogate-based modeling, to predict the performance of the I/O operations



Prior Work

- Heuristic-based search with a genetic algorithm to tune I/O performance
- Analytical models
  - Disk arrays to approximate their utilization, response time, and throughput
  - Application-specific models
- Herbein et al. use a statistical model, called surrogate-based modeling, to predict the performance of the I/O operations



Parameter Tuning – Challenges

- Large number of I/O parameters inter-dependent on each other.
- Real valued parameters do not allow brute forcing the parameter space to find optimal parameters.
- Application-specific models are limited to specific I/O patterns

Our Contributions

An auto-tuning approach based on active learning for improving both read and write performance

1. ExAct: An **execution-based auto-tuner** for I/O parameters (achieves up to 11x speedup over default).

2. PrAct: A fast **prediction-based auto-tuner** for I/O parameters (can tune I/O parameters in 0.5 minutes).

Bayesian Optimization

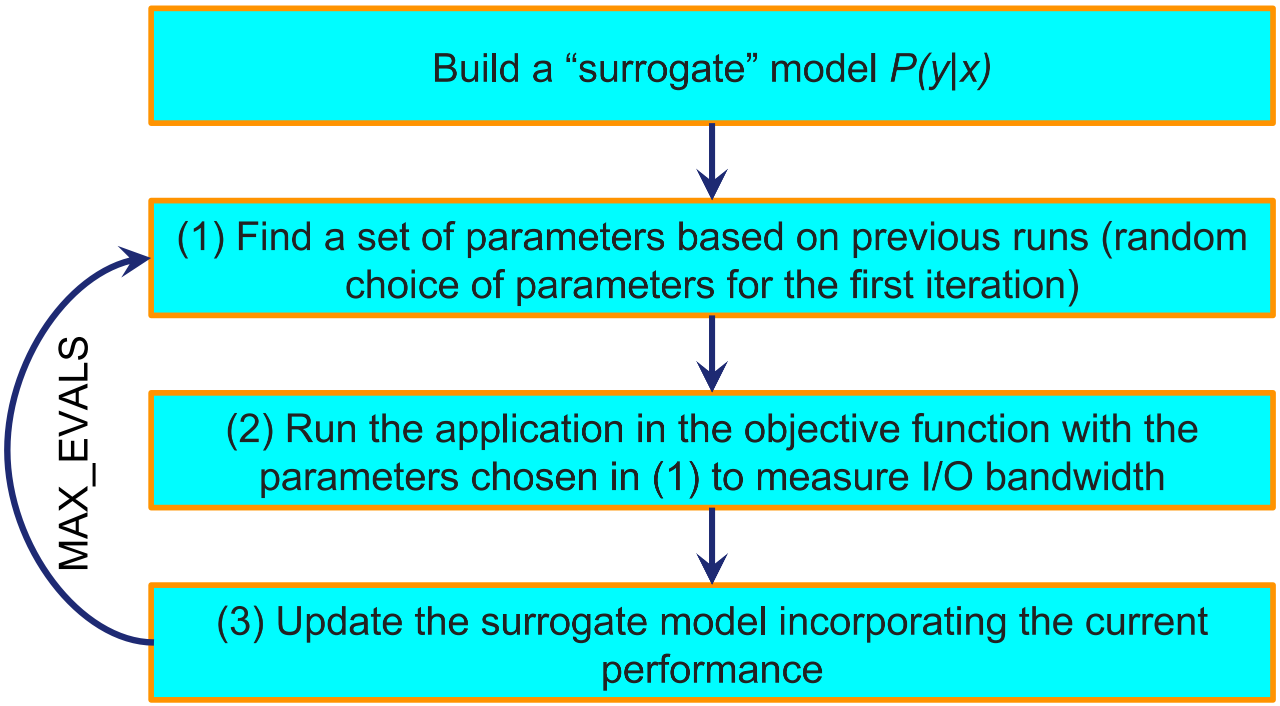
Limit expensive evaluations of the objective function by choosing the next input values based on those that have done well in the past

Mathematically, we can represent our problem as :

$$x^* = argmax_{x \in X} f(x)$$

- f(x) represents our objective function to minimize which in our case is run time of an application or an I/O kernel
- x is the value of parameters
- x* is best value found for each of parameters in sample space X.

Execution-based Auto-tuning (ExAct) Model



Prediction-based Auto-tuning (PrAct) Model

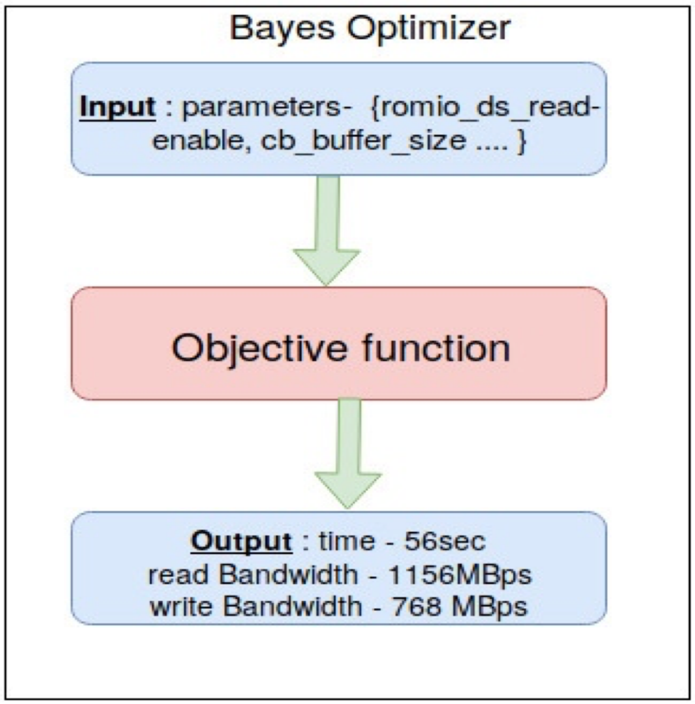
- Developed a performance prediction model using Extreme Gradient Boosting (XGB).

- PrAct uses predicted runtimes in the objective function in Bayesian Optimization model.

(2) Predict I/O bandwidth with the parameters chosen in (1)

- This reduces the time to obtain better performing I/O parameters.

Summary of Approaches



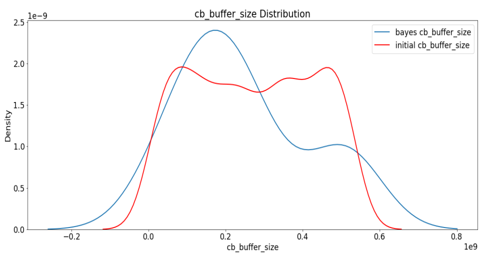
ExAct - Objective function obtains output by running the application on input parameters

Predict is an offline model trained on dataset that predicts I/O bandwidth for a given set of input parameters.

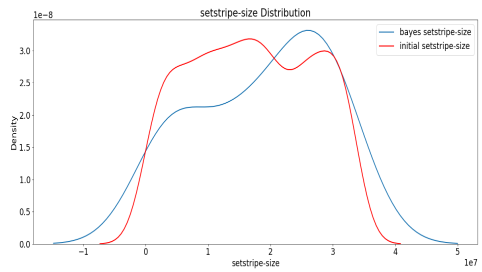
PrAct- Objective function obtains output by running *Predict* on input parameters

Bias and Learning Plots in ExAct

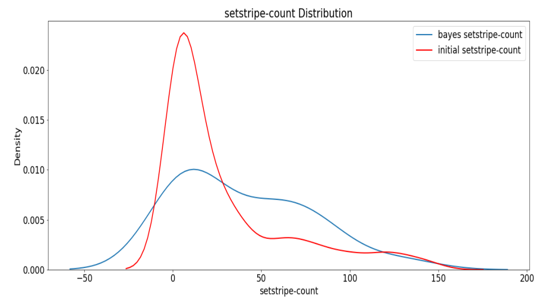
Cb-buffer size distribution



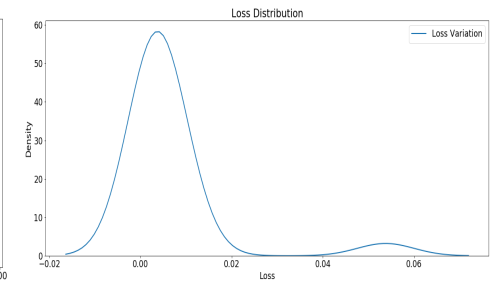
Stripe size distribution

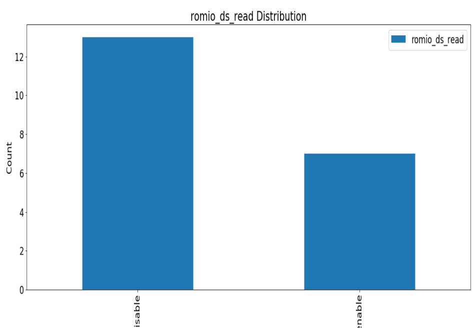


Stripe count distribution

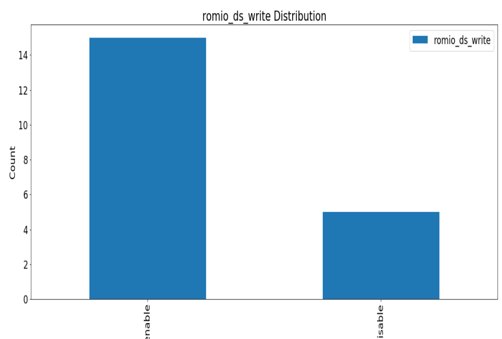


Loss distribution

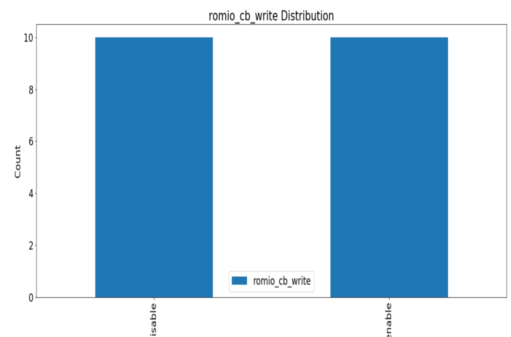




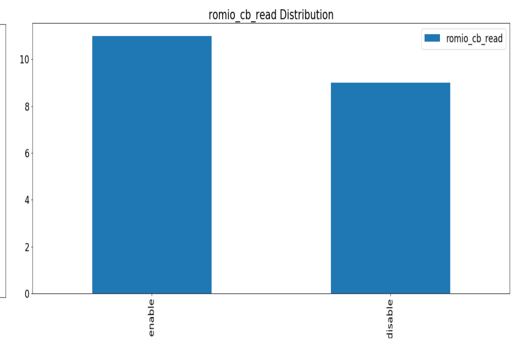
Romio ds_read



Romio ds_write



Romio cb_write



Romio cb_read

Configuration: 200X400X400 on 4X4X8 processes S3DIO

Red - Initial probability distribution

Blue - Post training prob. distribution

Application I/O Kernels for benchmarking

- S3D-IO: I/O kernel of S3D combustion simulation code
  - 40 input configurations
- BT-IO: I/O Benchmark Using NASA's NAS BTIO Pattern
  - 19 input configurations
- IOR: A commonly used file system benchmark
  - 13 input configurations
- Generic I/O: A write-optimized library for writing self-describing scientific data files
  - 45 input configurations

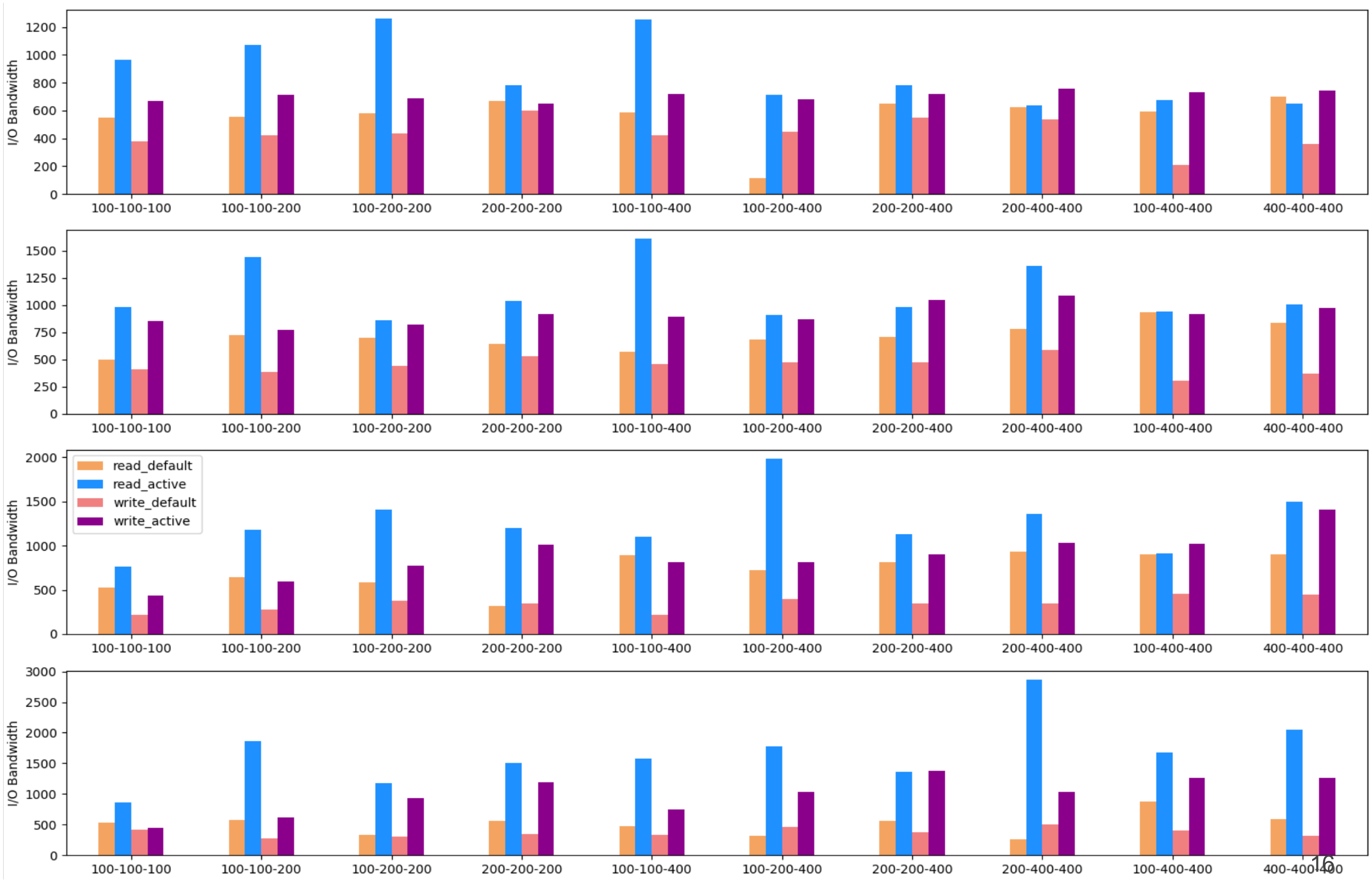
System Configurations

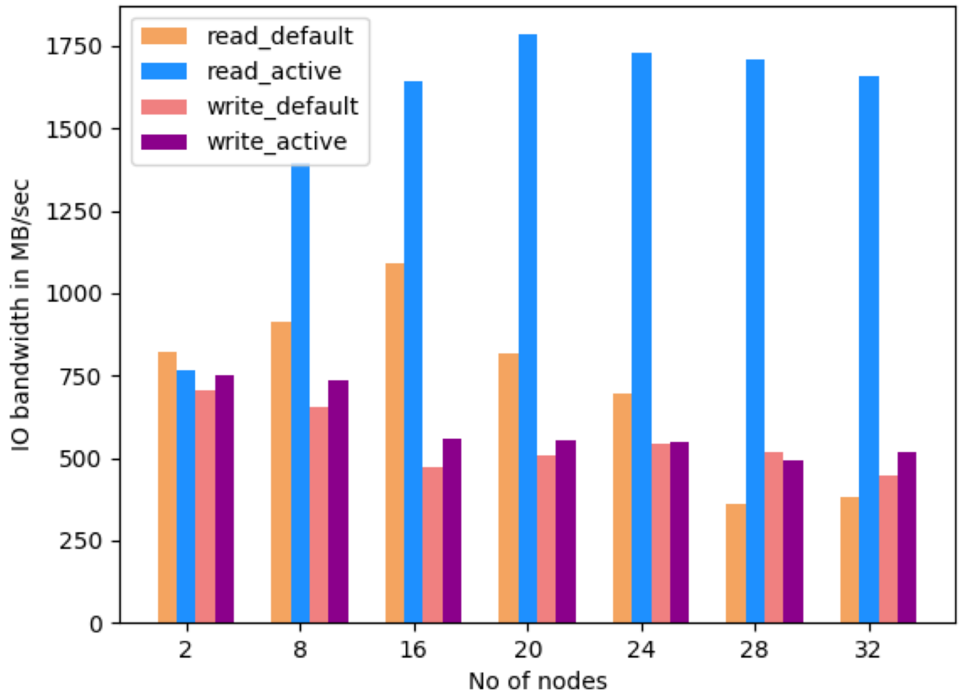
- HPC2010 (464-node supercomputer) at Indian Institute of Technology (IIT), Kanpur
  - Used a maximum of 128 processes.
- Cori, a CrayXC40 system at NERSC, LBNL
  - Used a maximum of 512 processes.

S3D-IO default vs. ExAct on HPC2010 (16 – 128 processes, 8 ppn)

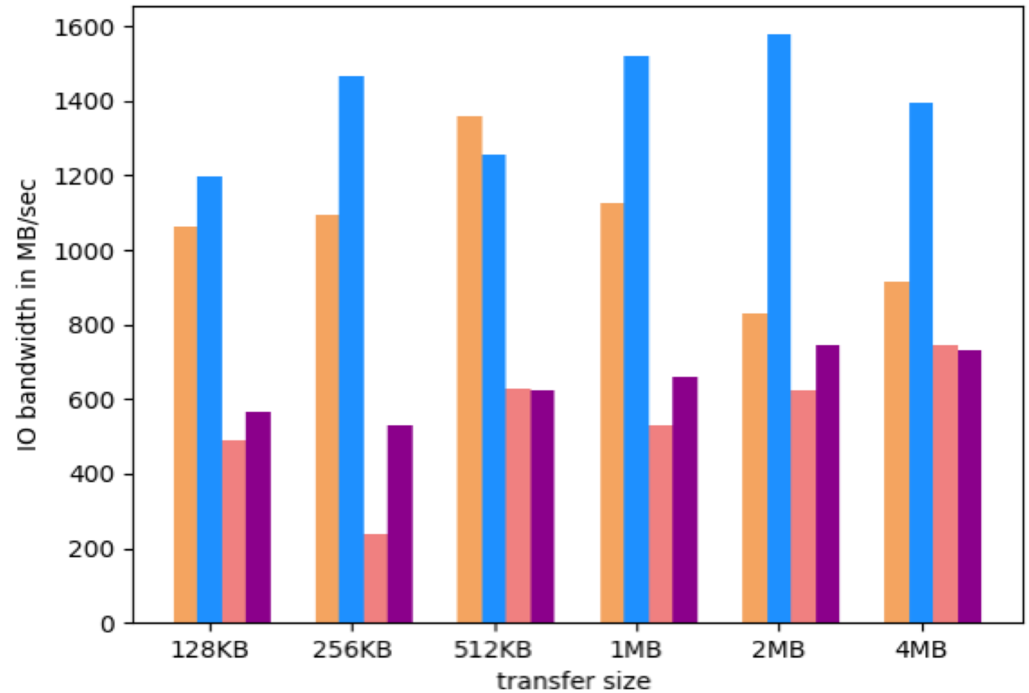
X-axis: Increasing data sizes

Y-axis: I/O bandwidths in MBps





IOR I/O bandwidths for varying node counts.
Strong scaling on 16 – 256 processes.



IOR I/O bandwidths for varying transfer sizes.
Data scaling on 64 cores with 100 MB block size.

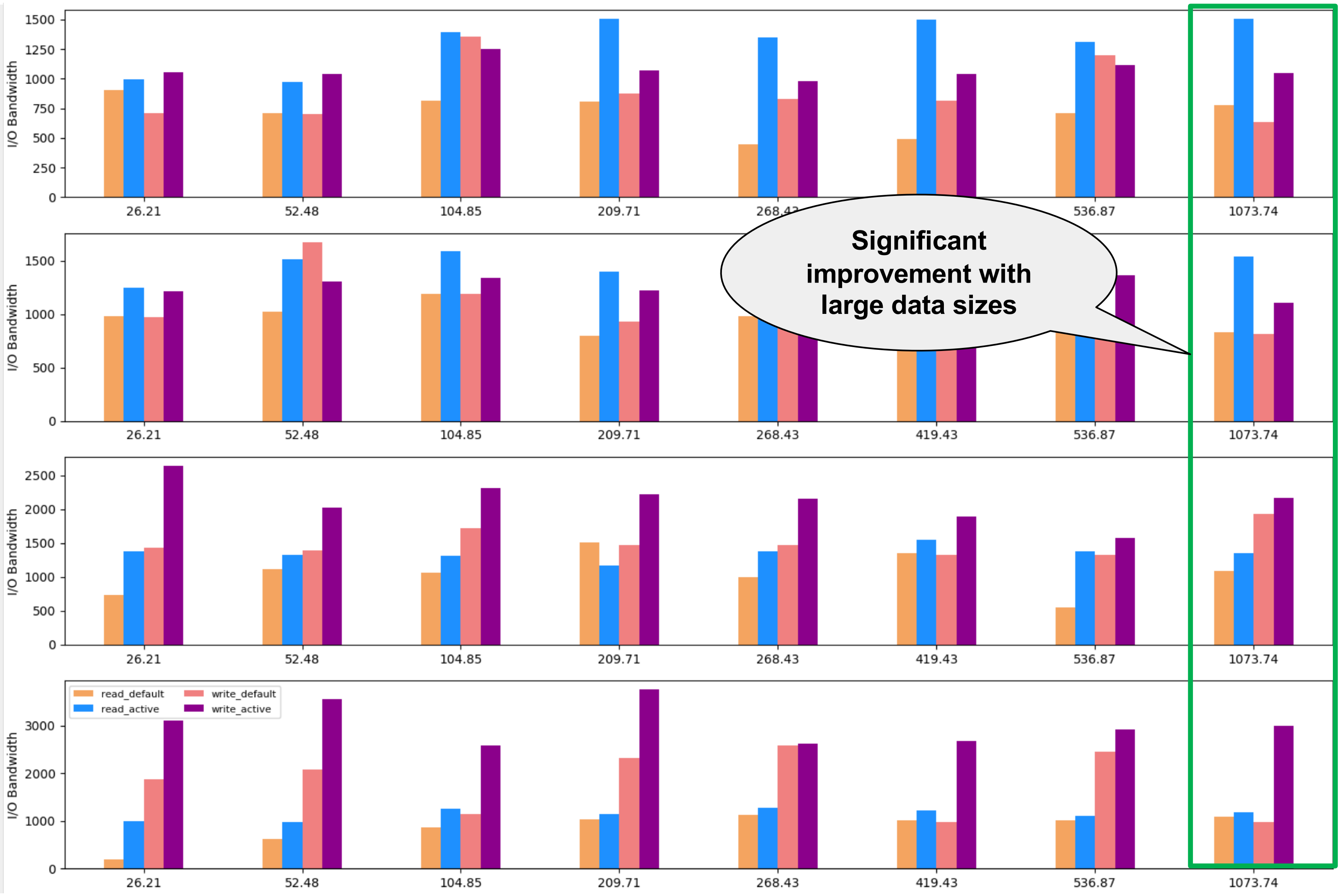
87% read and 20% write improvements (on average)

Default vs. ExAct I/O bandwidths using IOR on HPC2010

Generic-IO default vs. ExAct on HPC2010 (2, 4, 16, 28 nodes)

X-axis: number of particles (in millions)

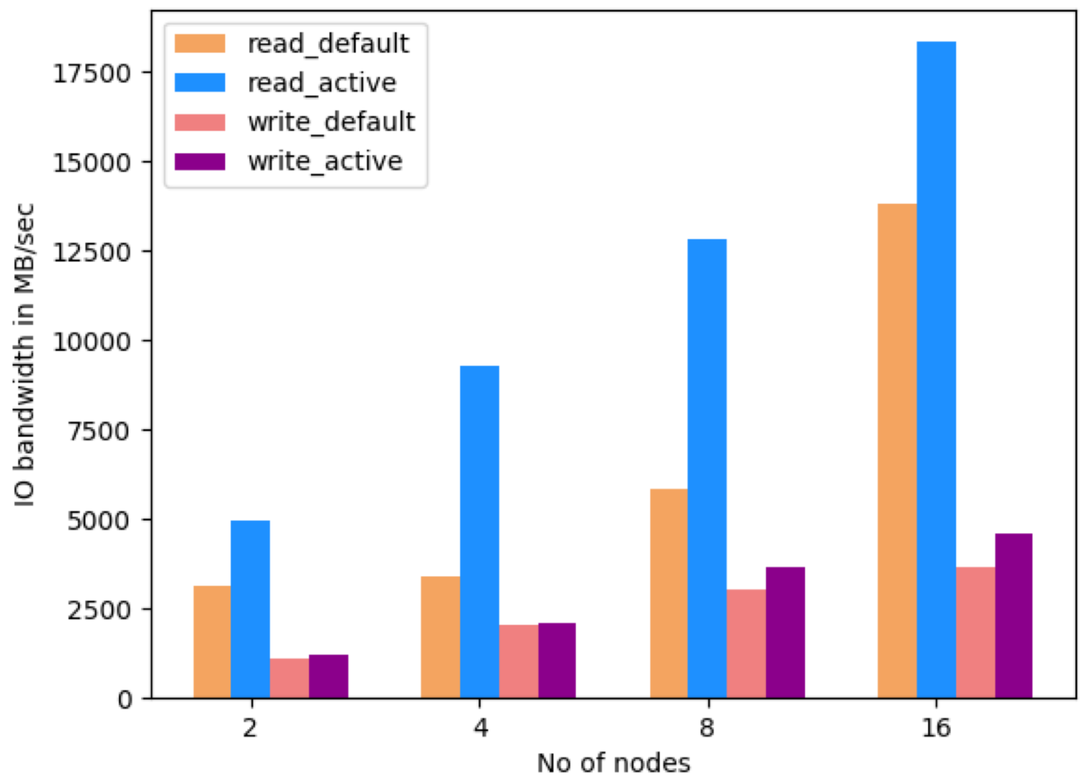
Y-axis: I/O bandwidths in MBps



S3D-IO default vs. ExAct on Cori (2 – 16 nodes, 32 processes per node)

X-axis: Number of nodes

Y-axis: I/O bandwidths in MBps



Weak scaling results for S3D-IO

ExAct Result Summary

Benchmark	Read(Avg)	Write(Avg)	Read(Max)	Write(Max)
S3D-IO	1.97X	2.21X	11.14X	4.03X
IOR	2.1X	1.0X	4.73X	2.23X
BT-IO	1.07X	1.76X	2.93X	4.86X
GenericIO	1.44X	1.51X	3.04X	3.06X

Analysis of tunable parameters

Benchmark	S3D-IO (200 x 200 x 400) on 4 x 4 x 8 processors (16 nodes) on HPC2010
Default parameters	stripe_size = 1 MB, stripe_count = 1, cb read/write = enable, ds read/write = disable, cb_buffer_size = 16 MB, cb_nodes = 16
Default Read/write	3002 /1680 MBps
ExAct parameters	stripe_size= 4 MB, stripe_count = 21, cb read/write = disable/disable, ds read/write = enable/disable, cb_buffer_size = 512 MB, cb_nodes = 13
ExAct Read/write	1198 / 293 MBps
Tuning Time	12.65 minutes

Performance Prediction Model (Predict) Accuracy

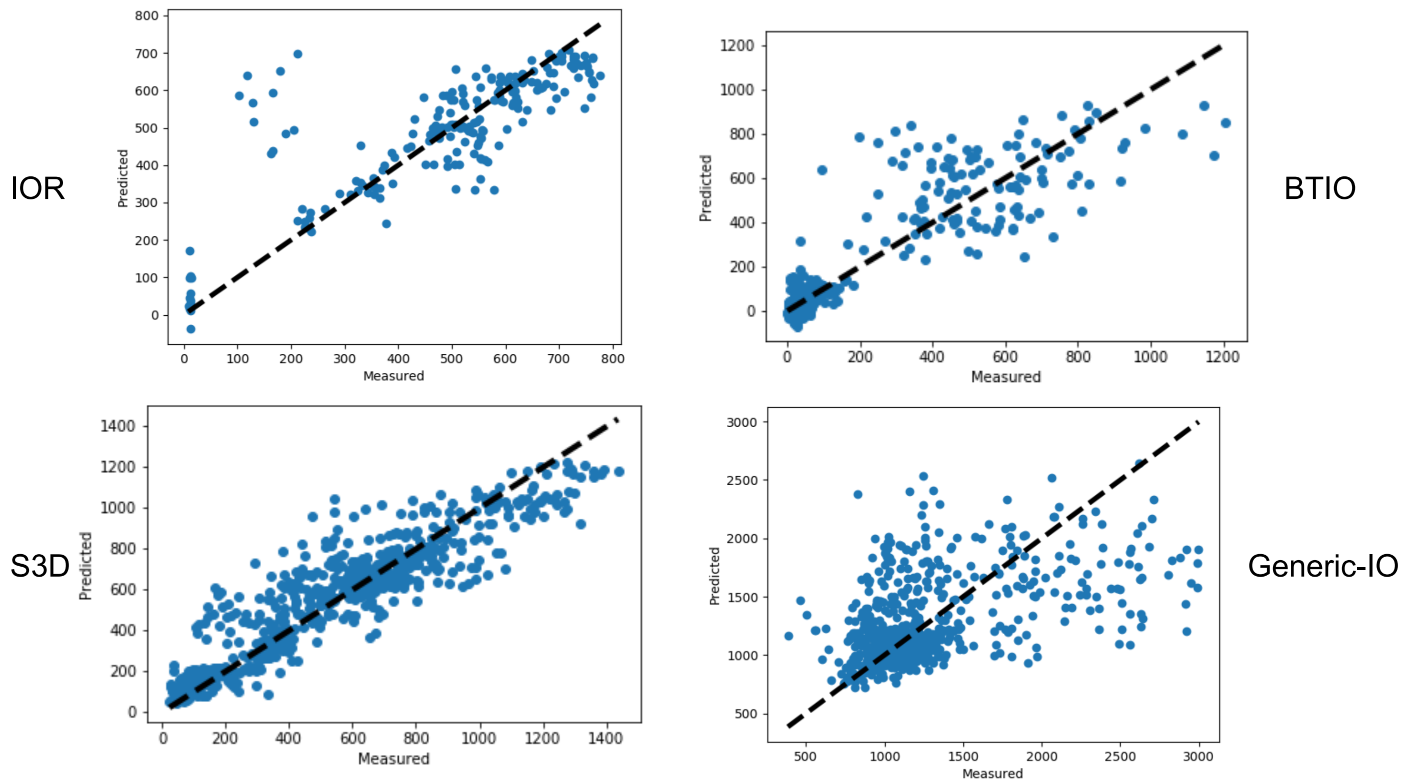
Benchmark	MdAPE (%)		R^2	
	Read	Write	Read	Write
S3D-IO	23.72	10.13	0.50	0.88
BT-IO	21.24	19.23	0.45	0.79
IOR	30.58	10.25	-0.20	0.47
GenericIO	12.22	13.42	0.42	0.24
S3D-IO	8.01	7.25	0.90	0.91

Median absolute percentage error and R^2 measure for various benchmarks on HPC2010 (rows 1 – 4) and Cori (last row) using XGB model-based prediction

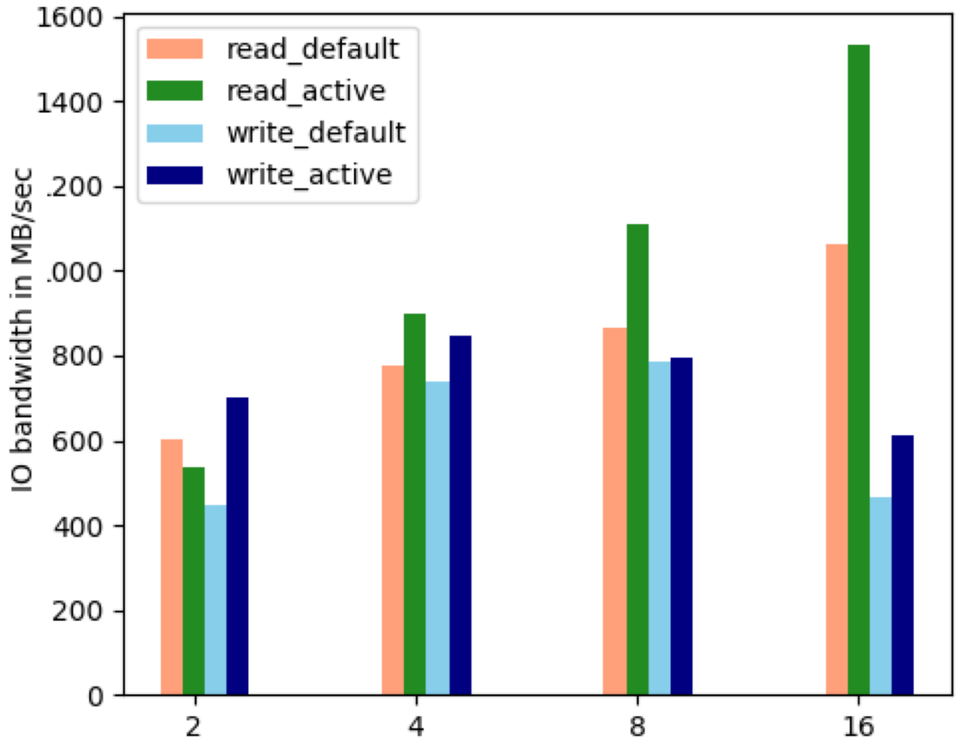
XGB-based Prediction Model Accuracy

Scatter plots of XGB-predicted values vs. measured values of write bandwidths for all benchmarks on HPC2010

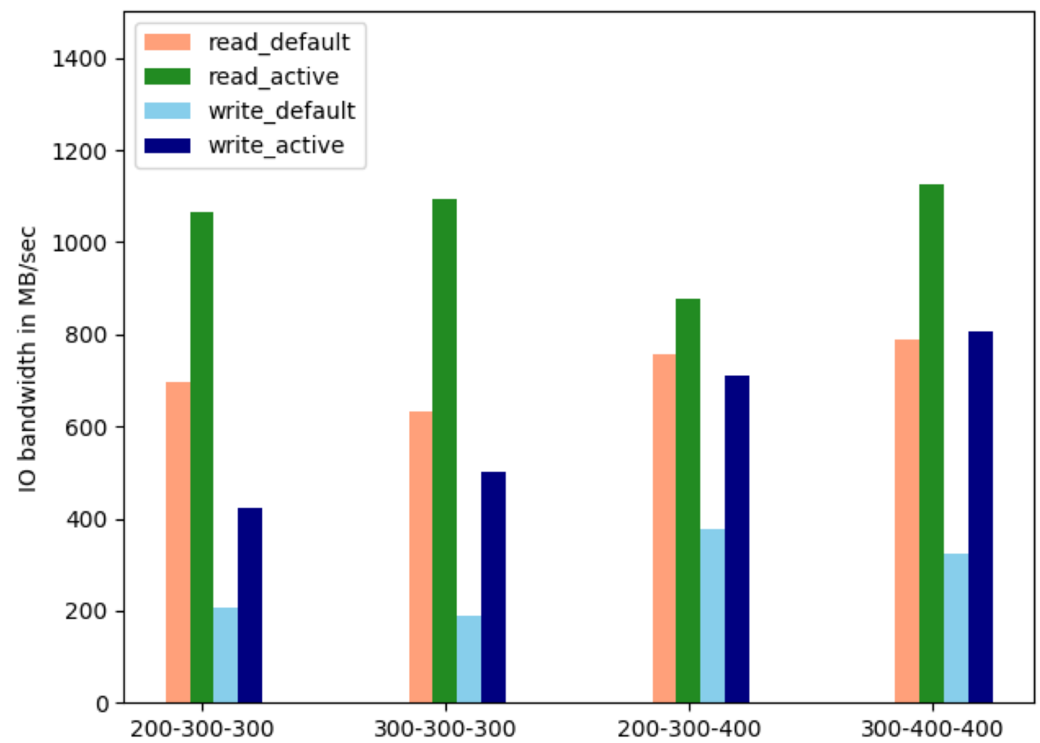
(30/70 split of train/test data)



Results – PrAct



S3D-IO weak scaling on unseen configurations



BT-IO with unseen configurations.

Results – PrAct

- PrAct was also evaluated for configurations that were not present in the training data
- Maximum of 1.6x and 1.2x performance improvement in reads and writes in S3D-IO
- Maximum of 1.7x and 2.5x performance improvement in reads and writes in BT-IO
- Observed degradation in read bandwidths in case of IOR, especially at high node counts. This is expected as the R^2 scores were low

ExAct vs. PrAct – Time vs. Performance Trade-off

- Average training time of PrAct is 18 seconds whereas that of ExAct is 13 minutes (varies with the run time of application)

- PrAct achieves a maximum performance improvement of 2.5x whereas ExAct achieves 11x improvement

Conclusions

- Developed execution-based (ExAct) and prediction-based (PrAct) auto-tuners for selecting MPI-IO and Lustre parameters
- ExAct runs the application and learns, whereas PrAct uses predicted values from analytical model to learn
- The only system-specific input to the model is the range of stripe counts
- Observed a maximum of 11x improvement in read and write bandwidths
- ExAct is able to improve write performance of large data sizes (e.g., 1 billion particles in GenericIO) by 3x
- Predict model uses XGBoost, and obtains less than 20% median prediction errors for most cases, even with 30/70 train/test split

https://github.com/meghaagr13/Autotuning-PIO