IPDPS 2020 New Orleans

iWAPT2020

HLRS High-Performance Computing Center | Stuttgart

Improving Collective I/O Performance with Machine Learning Supported Auto-tuning

Ayse Bagbaba

High Performance Computing Center Stuttgart

hpcabagb@hlrs.de

EXPERTISE

MARIE CURIE ACTIONS

Outline

- The Problem and Setup
  - Motivation & Objectives
  - Experiments
  - Performance Factors
  - Modeling
- Methodology
  - I/O Auto-tuning
  - Performance Models
- Evaluation
  - Evaluation 1: Influence of Auto-tuning
  - Evaluation 2: Validation of Performance Model
- Conclusion and Future Work

Problem Statement

- **Collective I/O** is one of the most important I/O access optimizations

- Various layers offer **tunable parameters** for improving collective I/O performance

- **Finding good configurations** of an I/O application is often challenging
  - **complex interdependencies** between the multiple layers
  - **diversity** among applications and HPC platforms
  - **the size of** configuration space
  - the **lack of** time or experience

Need for an automatically tuning solution for collective I/O operations

- compatible with as many engineering applications as possible,
- usable for engineers and scientists with little knowledge of parallel I/O,
- portable across multiple HPC platforms

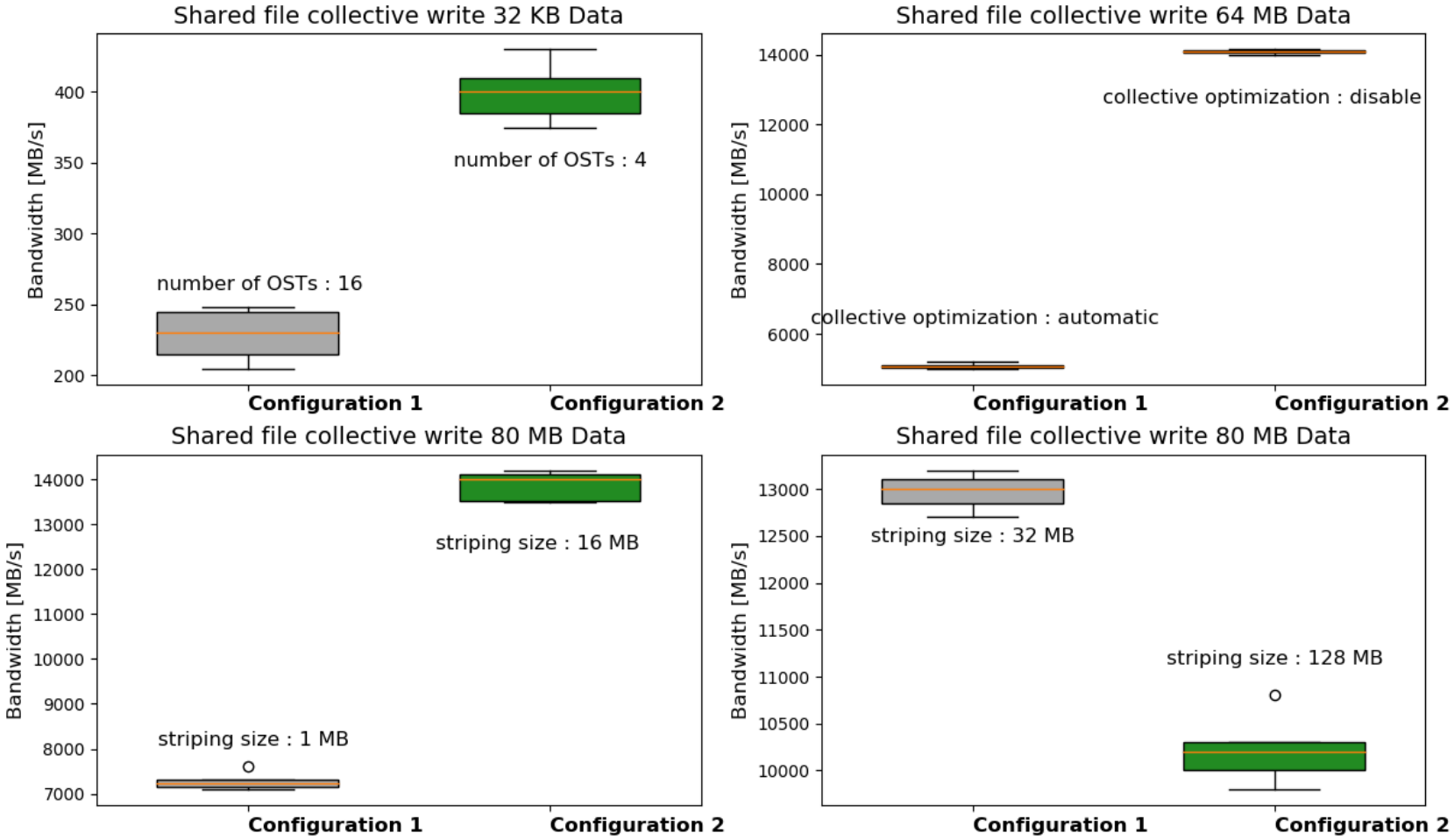
- follow the current MPI standard,
- run transparently to the users,
- produce acceptable little overhead,
- improve I/O performance automatically

System Specifications

- All evaluations were made on Hazel Hen (Cray XC40) with an InfiniBand connected Lustre file system
  - with a peak performance of 7.42 PFLOPS, Hazel Hen is one of the most powerful HPC systems in the world

Architecture	Hardware	File System	Storage	Bandwidth
Cray XC40	Intel Xeon E5-2680 v3 Cray Aries Network 7712 Compute nodes 90 Service nodes	Lustre 7 MDTs 54 OSTs	Cray Sonexion 2000	3.75 GB/s per OST

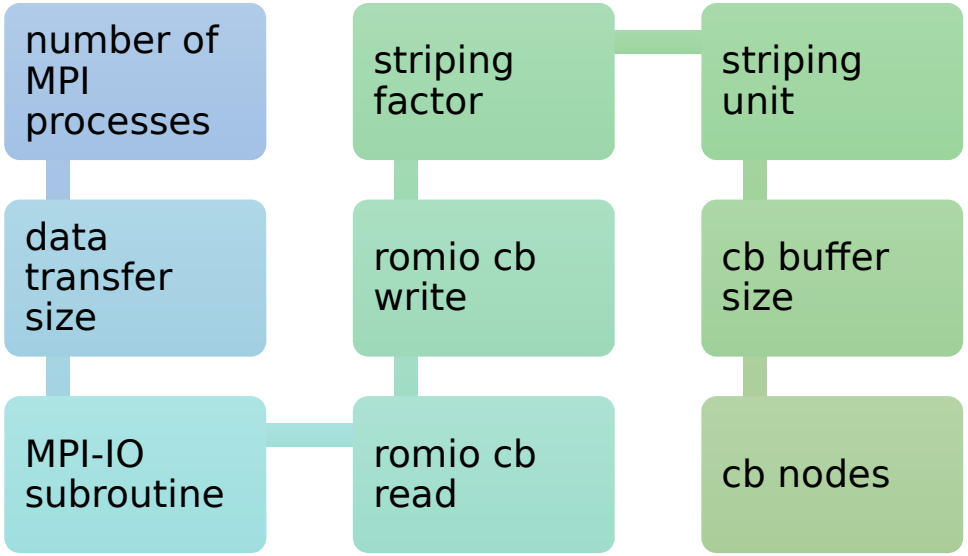
Impact of Wrong Parameters - an example



striping over 4 OSTs was about 71% better than the performance of striping over 16 OSTs

- disabling the collective buffering for collective I/O achieved about 269% improvement for writing performance, for non-small data transfer size

Performance Factors



- **How to set the values of these configuration parameters?**
- **How to create a performance predictor based on them?**

Modeling

- The I/O performance model can be formally defined as follows:

$$\phi = f(\alpha, \zeta, \omega),$$

- The modeling approach represents the similar cases that can be represented by using **a single model**
- **Machine learning** may provide an appropriate method

Methodology

Characterize collective I/O performance	Use IOR with simple I/O patterns to evaluate the impact of the filesystem and application characteristics
Use self-implemented benchmarks and applications	Transfer results to synthetic and real HPC applications which have more complex I/O patterns
Find out the best configuration	Find the best parameters for MPI-IO layer to allow more efficient usage of I/O resources
Support with machine learning	Evaluate algorithms e.g, machine learning techniques to automate the search best parameter set
Validate with real applications	Validate approach with real applications from project partners

I/O Auto-tuning

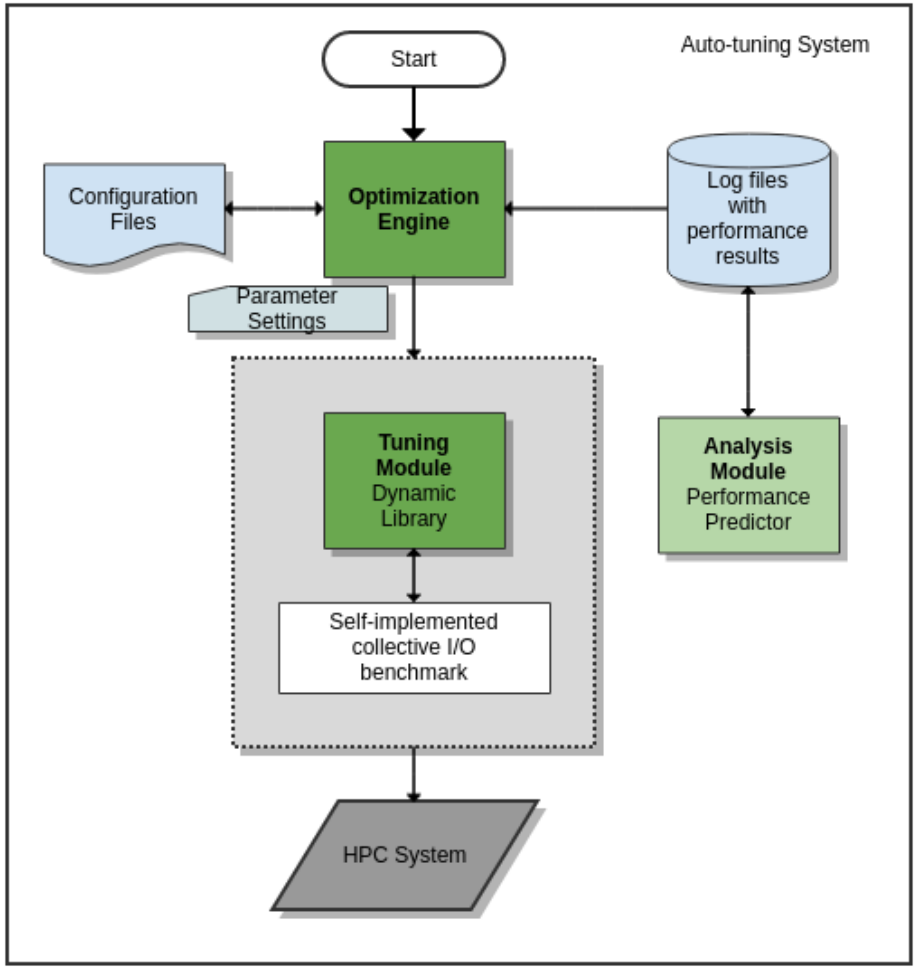
- **I/O auto-tuning** framework **for MPI-IO library**
  - tune I/O configuration parameters between layers transparently
  - use predictive models
    - characterize collective I/O
    - assist engineers/administrators in finding the better configuration,
    - use the knowledge of prior runs to choose the next promising configuration

- **Knowledge bridge** between application users and system administrators

I/O Auto-tuning - Approach

Optimization approach consists of three basic steps:

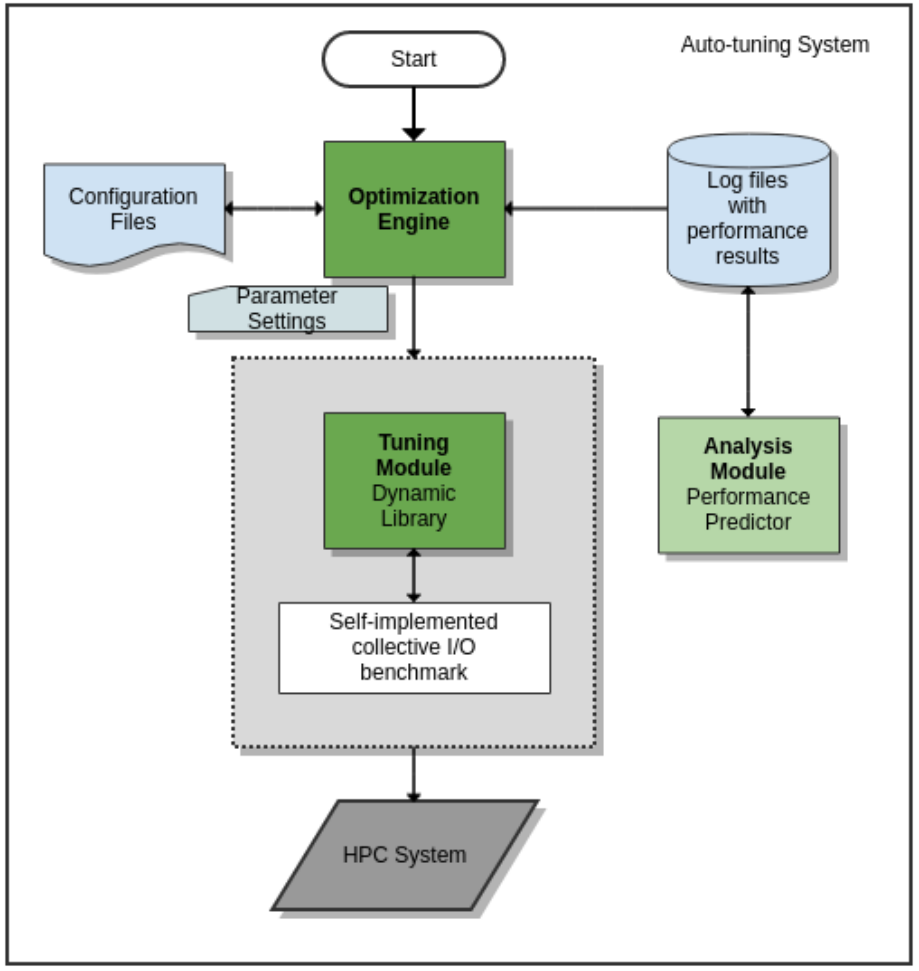
- identifying configurations' searching scope,
- choosing the best configuration parameters,
- suggesting or tuning.



I/O Auto-tuning - Modules

- Monitoring Module
  - surrounds MPI-IO layer
- Tuning Module
- Optimization Engine Module
- Analysis Module

- These modules support each other as a team

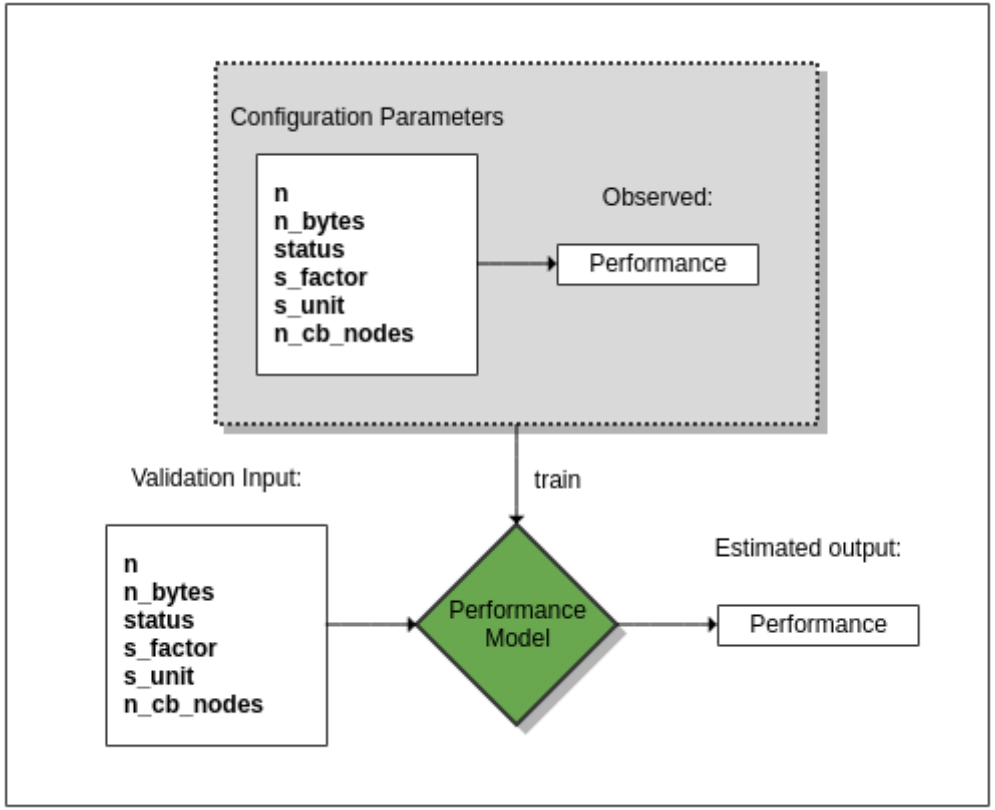


Performance Models

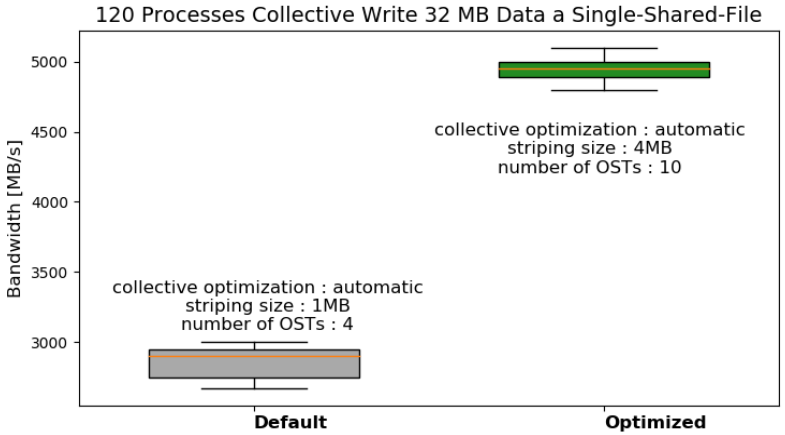
- Provides a performance estimate
  - **create a performance predictor model** to be expected from a given set of fixed and variable parameters
  - this performance model **is trained** on a number of samples
  - performance **is estimated** on the validation set

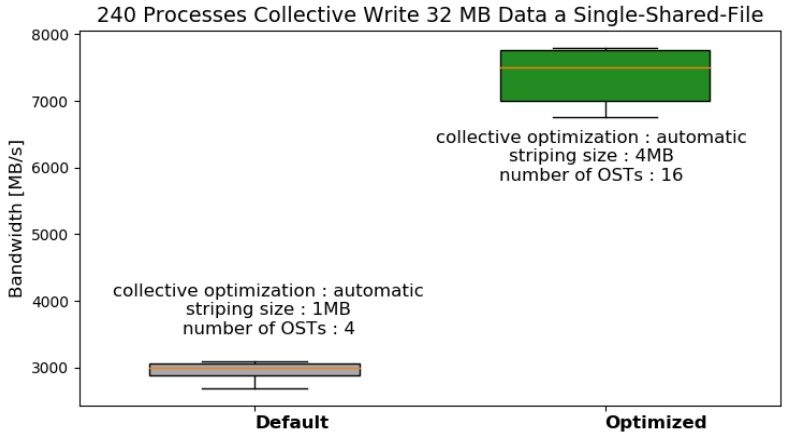
Name	Value
n	24-1200
n_bytes	256 B - 196 MB
n_cb_nodes	1 - 16
s_factor	1 - 16
s_unit	1 MB - 32MB
status	automatic; disable; enable
IO pattern	collective

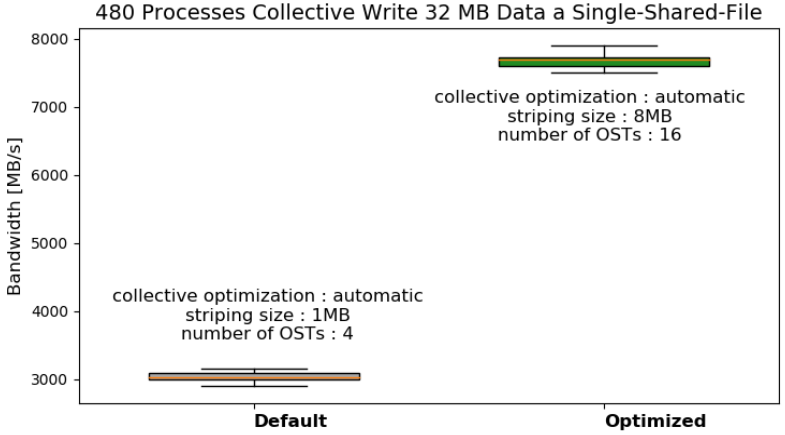
Training Set Configurations' Scope

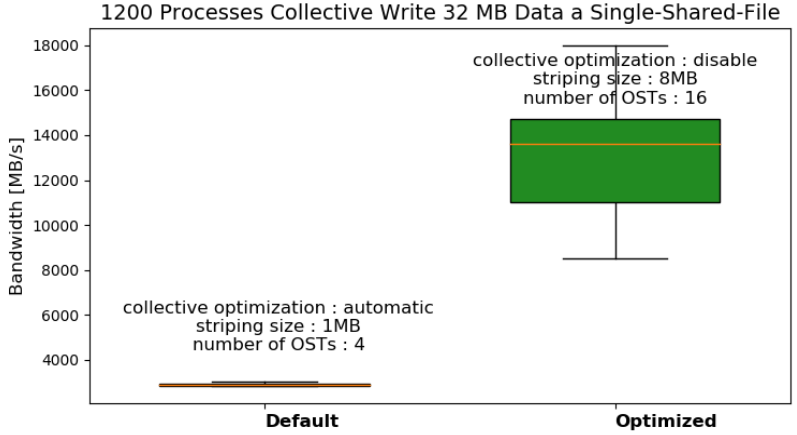


Evaluation - Auto-tuning



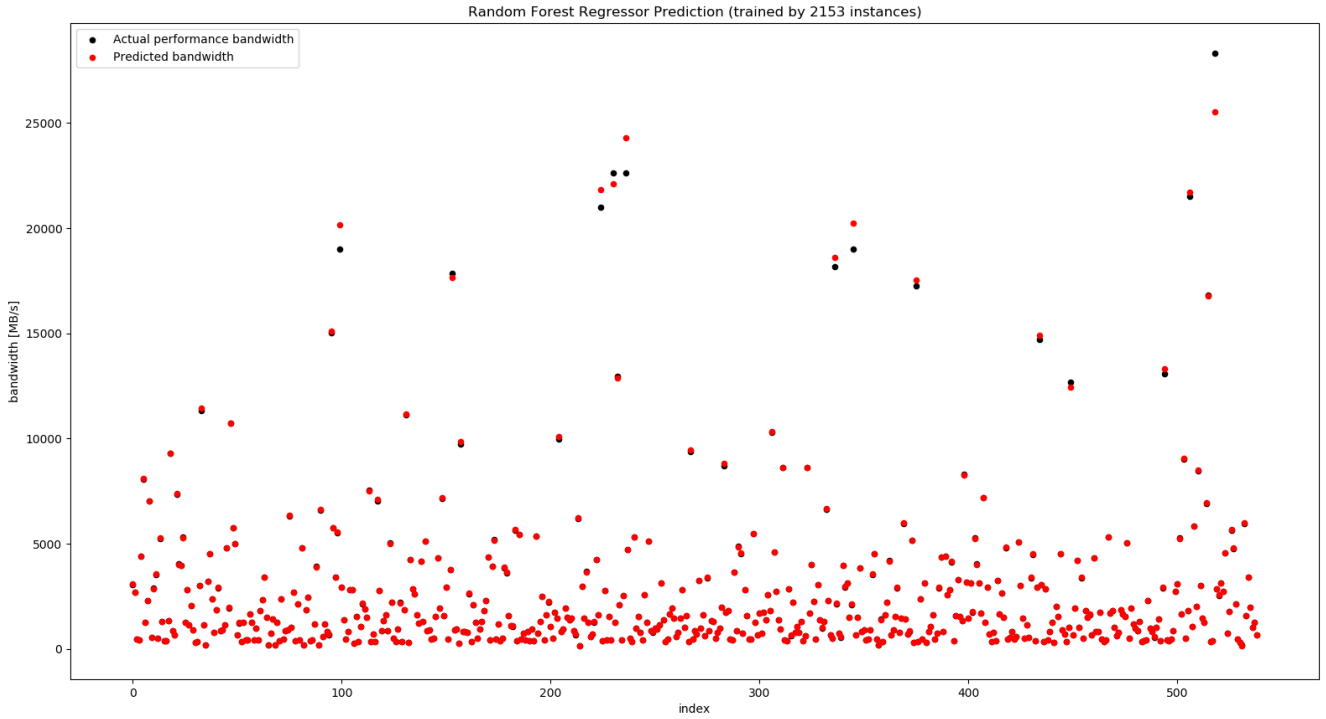






- As I/O benchmark software, IOR was used
- The overhead of the approach on 1 MPI process is measured as 0.02 seconds for MPI open and 0.2 seconds for MPI write.

Evaluation - Performance Model



max_depth	Prediction errors under different depths		
	Accuracy	MAE	RMSE
3	82.16 %	495.86	963.36
4	90.52 %	287.92	576.51
5	95.15 %	147.25	325.94
7	98.87 %	46.27	180.32
10	99.68 %	24.85	167.20

Prediction errors in MB/s for training sets under
different depth of each tree in the forest

Random forest regression performance model

maxdepth = 10, Accuracy: 99.68 %

Conclusion

- **a machine learning supported collective I/O auto-tuning solution for engineering applications**
  - can be understood by engineers or scientists with little knowledge of parallel I/O without any post-processing utility
  - implemented upon the MPI-IO library to be compatible with MPI based engineering applications, and be portable to different HPC platforms as well
- **an accurate indicator of the expected collective I/O performance**
  - can capture parallel I/O behavior as a function of application and file system characteristics
  - can provide insights into the metrics that impact I/O performance significantly

Future Work

- new parameters can be easily integrated to auto-tuning configuration files
- the auto-tuning solution will be tested on engineering applications in different professional areas to show the usability
- searching process can be reduced via performance models to consume less computing resources

Thanks for listening!

hpcabagb@hlrs.de