Parallel IO Tuning

Prof Preeti Malakar

Submitted By- Divyansh Singhvi
Megha Agarwal

Problem Statement

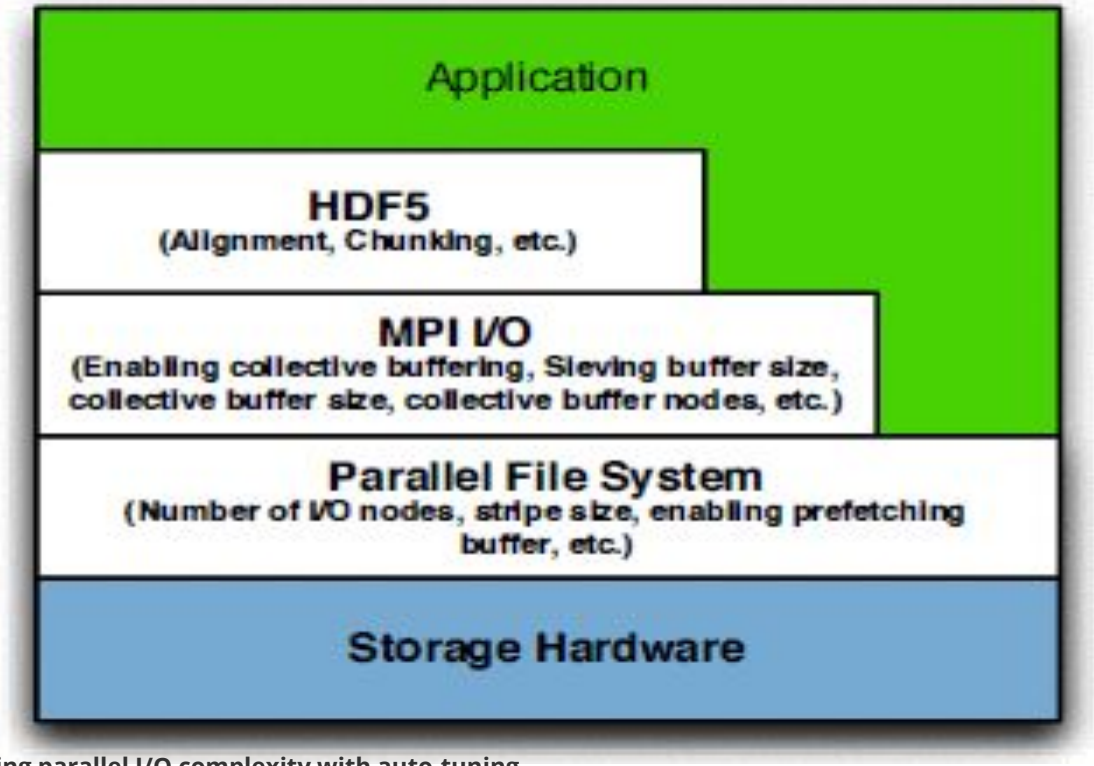
Automating Tuning of parameters for optimizing Parallel IO

Parameters : Lustre (file system), MPI-IO(MPI-IO middleware), PNetCDF/HDF5(high-level I/O libraries)

Introduction

- Parallel I/O performance depends on interaction of multiple layers of parallel I/O Stack(high-level I/O libraries, MPI-IO middleware, and file system)
- Each layer has several tunable parameters
- Since the layers are interdependent finding best parameters for a given stack is challenging for a specific application's I/O pattern
- A normal HPC application developer(expert in their scientific domain) resorts to default parameters resulting in poor performance.

Introduction

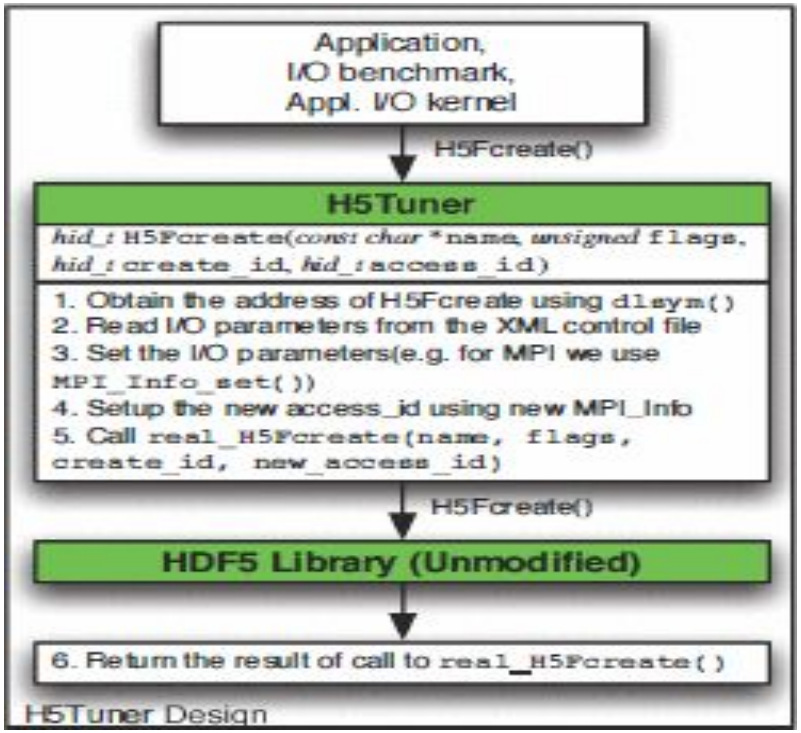


Src - **Taming parallel I/O complexity with auto-tuning**

Parallel Performance dependencies

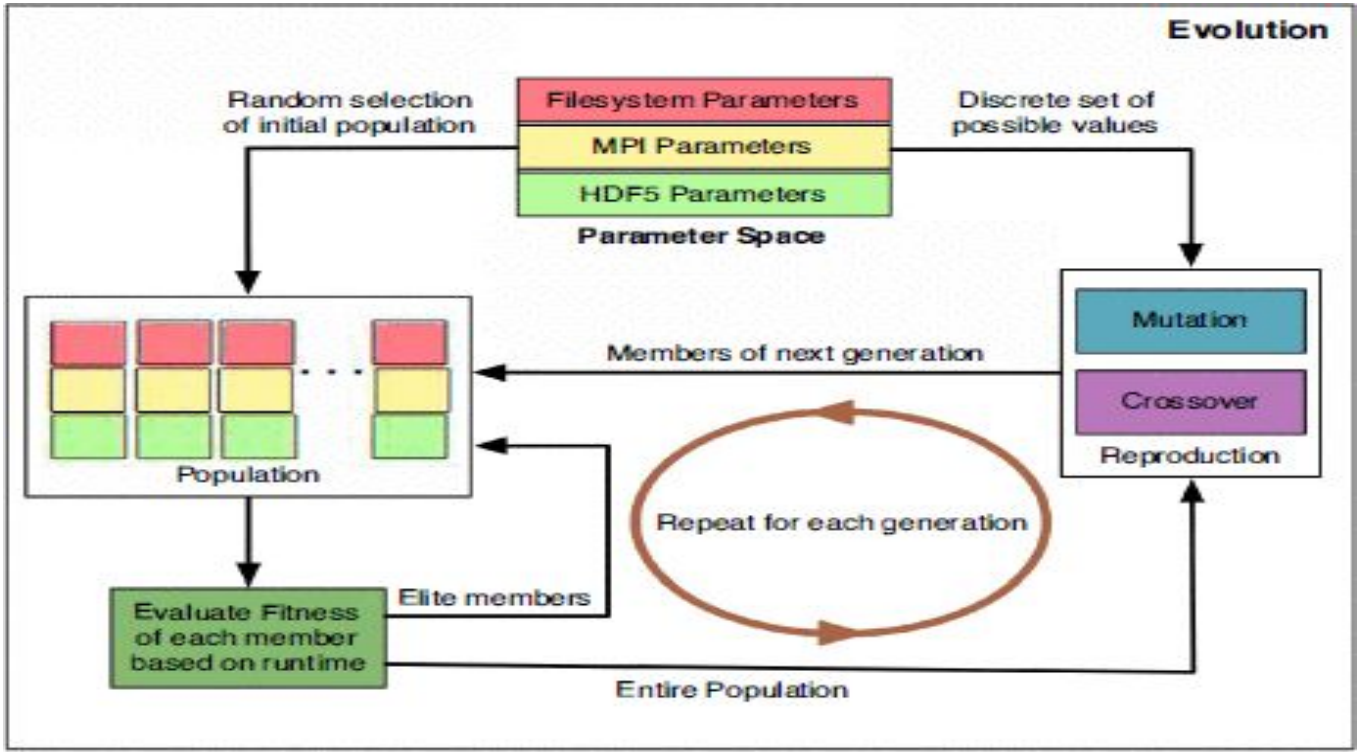
1. The I/O architecture;
2. The speed and amount of I/O hardware (disks, etc.);
3. How well the file system can handle concurrent reads and writes; and
4. How well the file system’s caching policies (read-ahead, write-behind) work for the given application.

Previous Work



Src - **Taming parallel I/O complexity with auto-tuning**

Previous Work



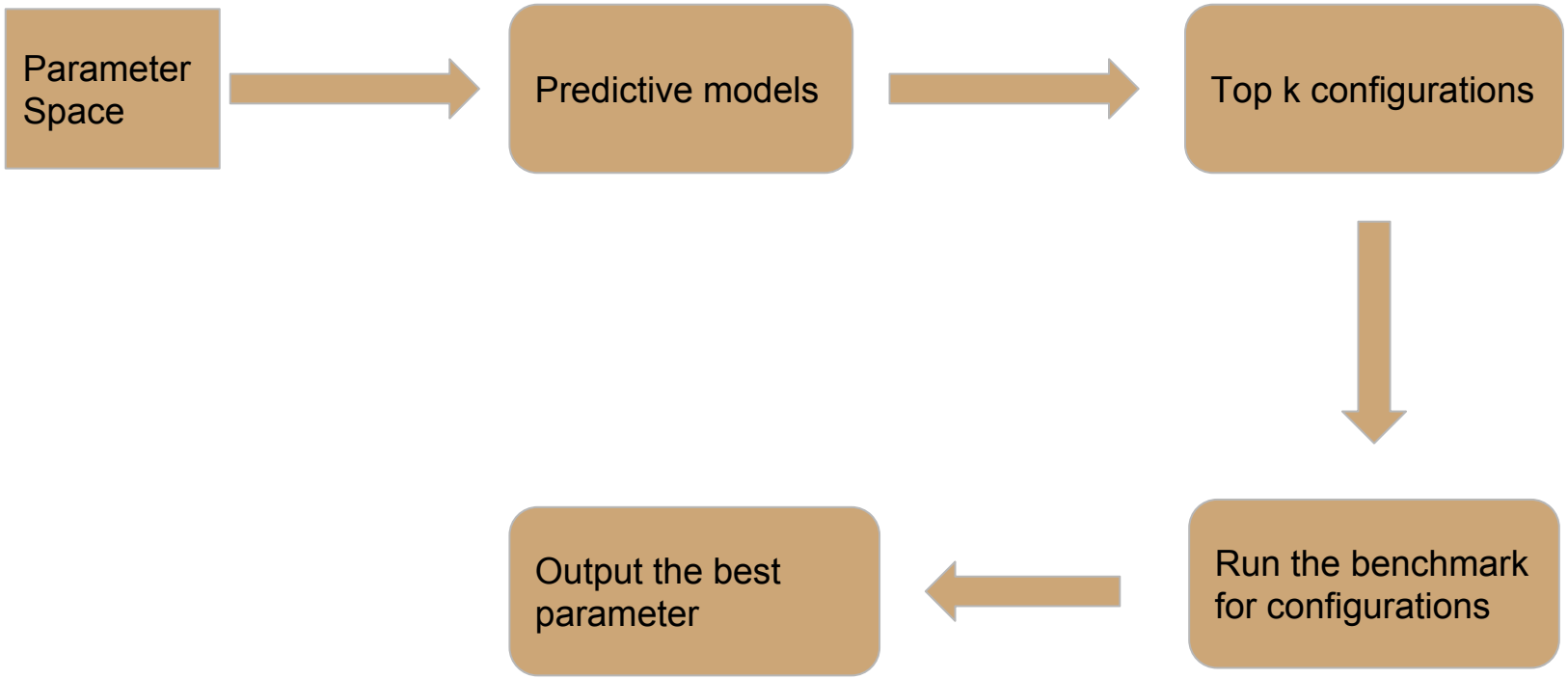
Src - **Taming parallel I/O complexity with auto-tuning**

Challenges

- Large number of parameters interdependent on each other.
- Some of the parameters are real valued hence doesn’t allow brute forcing the parameter space to find optimal parameters.
- Even if the parameters are constrained in a range, the number of possible combinations are several which makes the process of auto tuning very time consuming.

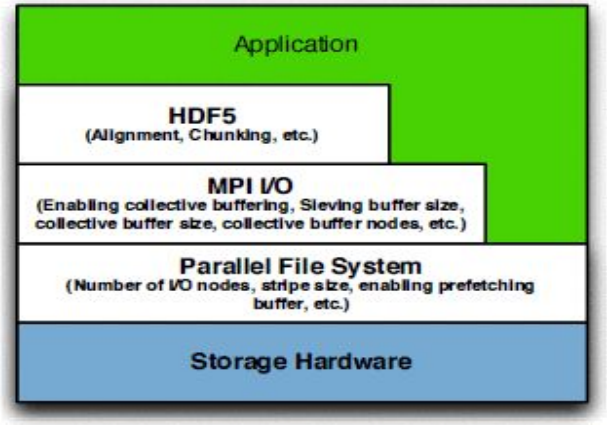
Our Approach

Design


Parameter Space
Predictive models
Top k configurations
Run the benchmark for configurations
Output the best parameter

Step 1 - Parameter Space

- It will consists of various important parameters for all the three levels of parallel I/O stack (file system, MPI-IO, high level libraries)
- Example : cb_nodes, stripe count , stripe size, romio_ds_read, romio_ds_write, romio_cb_write, romio_cb_read



Src - **Taming parallel I/O complexity with auto-tuning**

Step 2 - Creating predictive models

- Training set comprise of only few parameter values among the infinite possible values of real valued parameters.
- Groups of various parameters may be constructed which have higher correlation to obtain multiple models predicting I/O time based on subset of parameters.
- The model will be a SVM classifier.

Step 3 - Obtain k best configuration

- With our predictive models we will be picking k best performing parameter values.

Step 4 : Output the best parameter

Approach 1

- Run each of the best k configuration over the benchmark application.
- Print out the one giving the best performance.

Approach 2

- Randomly select one of the k parameter and give it as an output.

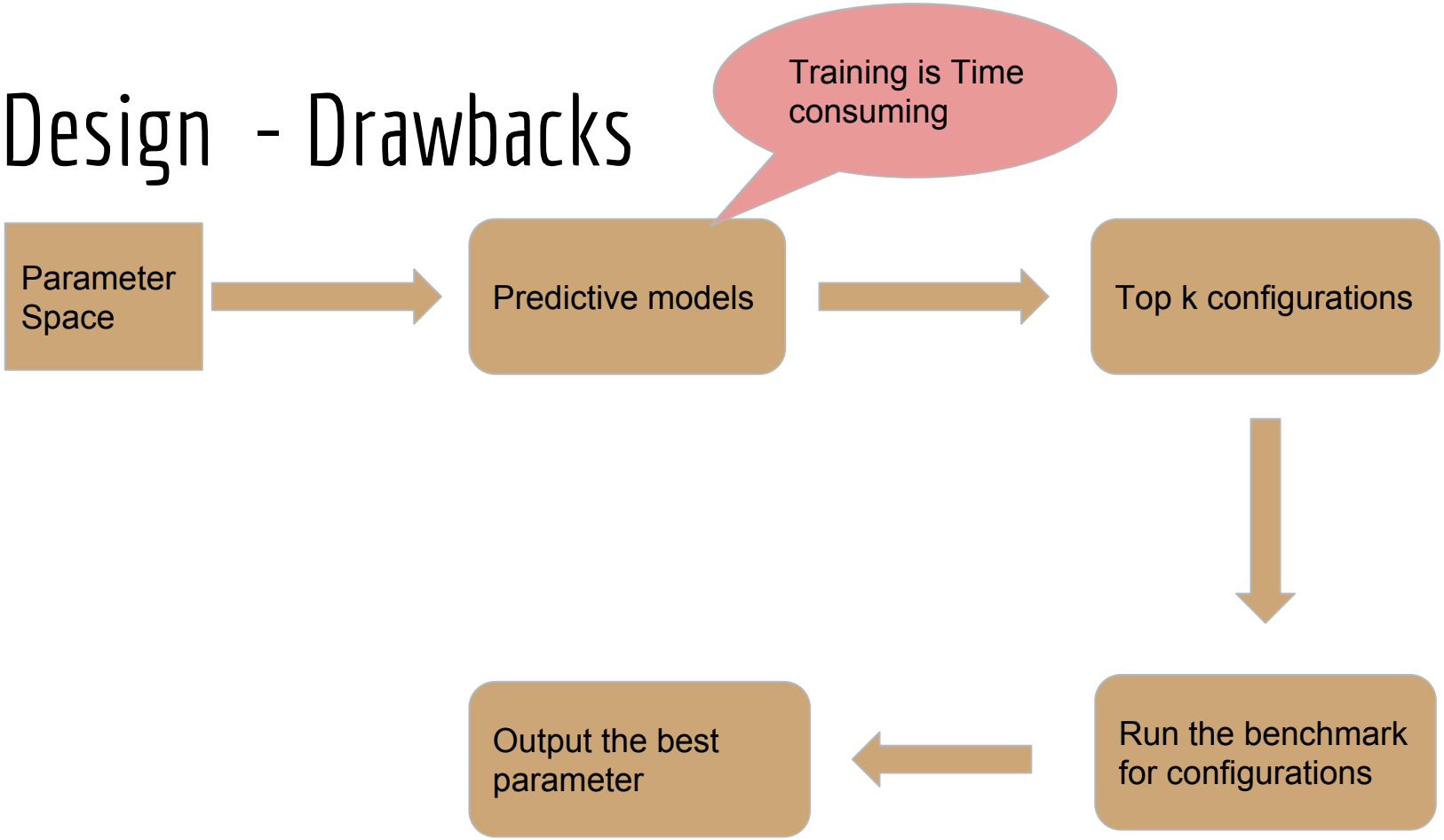
Approach 3

- Using machine learning algorithms like genetic algorithms to find the best performance which will introduce a bit of randomness and can hope to improve result even if predictive model is flawed.

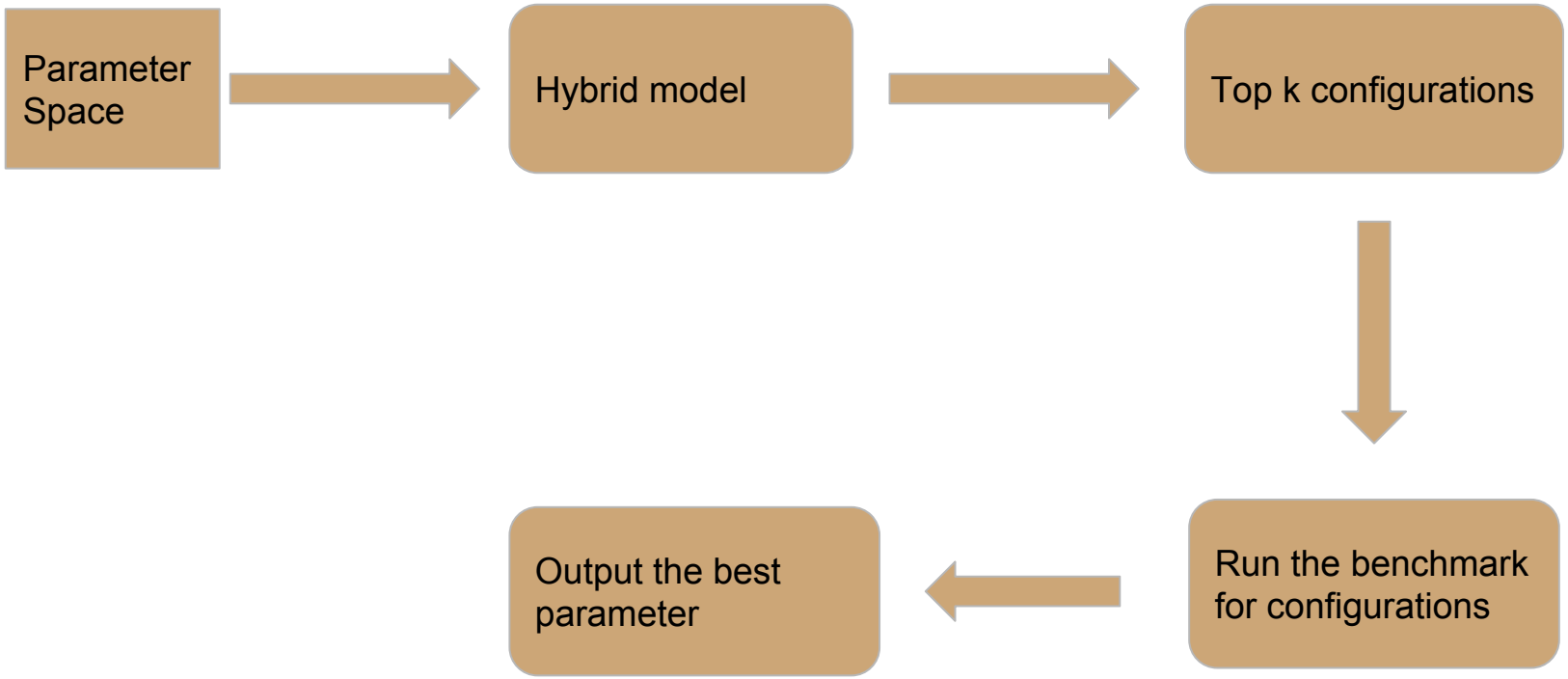
Current Progress

1) Figured out the approach for auto tuning
2) Installation of PNetCDF on HPC
3) Plug and play with S3D-IO by changing cb and ds options on HPC
4) Trying to pass hints for MPIIO in environment variables but are not working.
5) So now we are trying to make wrapper for MPI_info calls to tune parameters independent of source code.

Design - Drawbacks



Optimization that may be done



Further work may be possible - Hybrid Model

- Some parameters may be set analytically based on heuristics
- Other parameters will be modeled by SVM
-

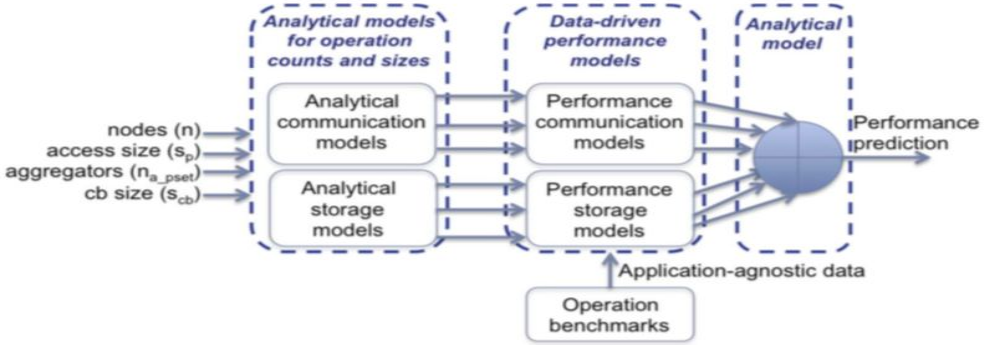


Figure 2. Hybrid model of two-phase I/O.

Src - [Collective I/O Tuning Using Analytical and Machine Learning Models]