# ACIC: Automatic Cloud I/O Configurator for Parallel Applications

Mingliang Liu
Tsinghua University
liuml07@gmail.com

Ye Jin
NCSU
yjin6@ncsu.edu

Jidong Zhai
Tsinghua University
zhaijidong@tsinghua.edu.cn

Yan Zhai
UW-Madison
zhaiyan920@gmail.com

Qianqian Shi
Tsinghua University
shiqiezi@gmail.com

Xiaosong Ma
NCSU and ORNL
ma@ncsu.edu

Wenguang Chen
Tsinghua University
cwg@tsinghua.edu.cn

## ABSTRACT

To tackle the highlighted I/O bottleneck problem in cloud, we propose ACIC, a system which automatically searches for optimized I/O system configurations from many candidates for each individual application running on a given cloud platform. The top ACIC-recommended configuration improves the applications' performance by a factor of up to 10.5 (3.0 on average), and cost saving of up to 89% (53% on average), compared with a commonly used baseline.

## Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Modeling techniques

## Keywords

Storage, Modeling, Performance, Cloud Computing

## 1. INTRODUCTION

More and more HPC users today are running their programs in the cloud [1]. Unfortunately, cloud platforms amplify the growing performance gap between I/O systems and other components [6]. The cloud enables users to setup optimized I/O configurations for **individual** applications upon their execution, instead of configuring I/O system for **all** applications that may run in the system [2]. However, taking advantage of this configurability and deriving optimized per-application I/O configuration are very challenging manually. Moreover, the complexity of both the underlying system and the parallel applications, and the obscureness of I/O system hardware/software details due to virtualization, make white-box modeling and analysis unrealistic.

We propose ACIC (**A**utomatic **C**loud **I**/O **C**onfigurator), which automatically searches for optimized I/O system configurations from many candidates for each individual application running on a given cloud platform. It takes advantage of a black-box machine learning model to train between representative I/O system exploration variables and optimal configuration target (cost and performance). After training the model on the target cloud platform, given the I/O characteristics of a certain application, ACIC evaluates candidate I/O configurations and recommends an optimized configuration for performance or monetary cost. The originality of our work lies in the cost-saving mechanisms that make such approaches affordable on clouds: *1*) we enable *reusable training* by adopting a generic synthetic I/O benchmark and systematically sample the parameter space; *2*) we propose the potential of building a shared, public I/O performance/cost training database.

## 2. APPROACH

Figure 1 illustrates the ACIC architecture. The central component of ACIC is a black-box prediction model, which can be bootstrapped with a certain amount of initial training. ACIC takes both the cloud system I/O configuration parameters and application I/O characteristics. ACIC chooses the synthetic benchmark [5] to collect training data because it's generic, highly configurable and open-source. whose parameters are generated from the reduced I/O characteristic space. ACIC collects the performance (cost) metric on the target cloud system configured with the candidate I/O configurations as training data. It trains the black-box prediction model with classification and regression trees (CART) [3], a popular machine learning method. Users provide the I/O characteristics of one individual application as input to the prediction model. The output is the top predicted I/O configurations from many candidates.

| **Name** | **Value** | **Rank** |
|---|---|---|
| Disk device | {EBS, ephemeral} | 10 |
| File system | {NFS, PVFS2} | 5 |
| Instance type | {cc1.4xlarge, cc2.8xlarge} | 12 |
| I/O server number | {1, 2, 4} | 3 |
| Placement | {parttime, dedicated} | 7 |
| Stripe size | {64KB, 4MB} | 6 |
| Num. of all processes | {32, 64, 128, 256} | 14 |
| Num. of I/O processes | {32, 64, 128, 256} | 4 |
| I/O interface | {POSIX, MPIIO} | 9 |
| I/O iteration count | {1, 10, 100} | 13 |
| Data size | {1, 4, 16, 32, 128, 512 (MB)} | 1 |
| Request size | {256KB, 4MB, 16MB, 128MB} | 8 |
| Read and/or write | {read, write} | 2 |
| Collective | {yes, no} | 11 |
| File sharing | {share, individual} | 15 |

Table 1: The variables affecting performance and cost.

To summarize, we list the I/O configurations and application I/O characteristics parameters considered in this ACIC prototype in Table 1. To define the range of the values, we studied over 12 HPC applications in several scientific areas with different scale (32 to 256). Concatenated together,

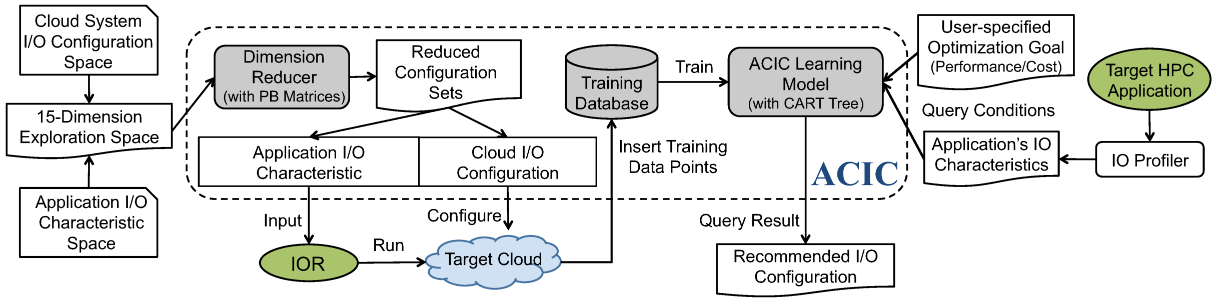

Cloud System I/O Configuration Space
15-Dimension Exploration Space
Application I/O Characteristic Space
Dimension Reducer (with PB Matrices)
Reduced Configuration Sets
Application I/O Characteristic
Cloud I/O Configuration
Training Database
Train
ACIC Learning Model (with CART Tree)
User-specified Optimization Goal (Performance/Cost)
Target HPC Application
Query Conditions
Application's IO Characteristics
IO Profiler
Insert Training Data Points
ACIC
Input
Configure
Run
IOR
Target Cloud
Query Result
Recommended I/O Configuration

Figure 1: Predicting model architecture

these parameters make a 15-D exploration space as listed in Table 1. The top 6 variables are I/O system options in cloud, while the other ones are workload characteristics. For each parameter, we sample its value range during our bootstrap training. Considering the number of model parameters and their value range, the training space is so huge that tuning both the benchmark input size and I/O system configurations to cover all kinds of HPC applications' behavior is impossible. ACIC employs a dimension reducer using Plackett-Burman (PB) matrices [4] to rank all the I/O parameters and only trains the top 10 of them.

## 3. EVALUATION

All our experiments are performed on Amazon EC2 Cluster Computing Instances (CCIs) (*cc*2.8*xlarge*) [1]. We use the Amazon Linux OS 201202, Intel compiler 11.1 and Intel MPI 4.0. The baseline configuration is the popular but simple dedicated NFS server with one EBS disk attached. The performance and cost are calculated as:

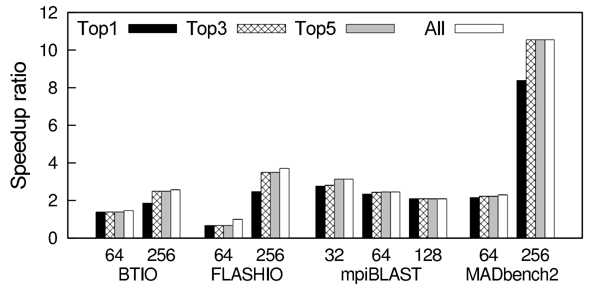
$$speed\_up = \frac{time_{baseline/median}}{time_{ACIC}} \quad (1)$$

$$cost\_saving = \frac{cost_{baseline/median} - cost_{ACIC}}{cost_{baseline/median}} \times 100\% \quad (2)$$

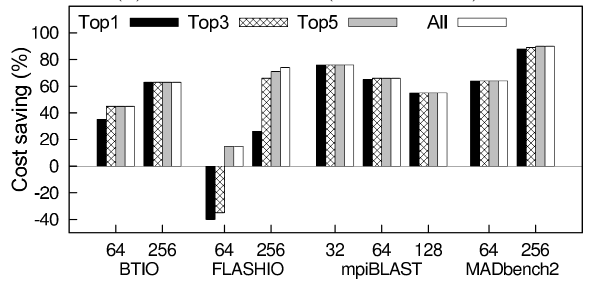
We examine the potential difference made by verifying top-$k$ recommendation set from ACIC. This is considering that the parameter filtering done by PB design and the rather sparse parameter sampling we perform may lead to co-champions. Figure 2 shows the execution time and cost improvement achieved by the best configuration among the top 1, 3, and 5 ACIC recommendations and eventually all I/O configurations (the true optimal configuration). The results reveal that the top recommendation (median if ACIC has co-champions) works fairly well, though considering more top candidates does help sometimes (256-process FLASHIO, 32-process mpiBLAST, and 256-process MADbench2). Particularly, we see that in almost all cases, little further gain can be achieved by checking beyond the top 3 ones.

## 4. CONCLUSION

We propose ACIC, an automatic cloud I/O system configuration tool for HPC applications encompassing several statistical and machine learning techniques to enable application-dependent, incremental model training and black-box performance/cost prediction. We released it to HPC community for free at `http://hpc.cs.tsinghua.edu.cn/ACIC`



(a) Execution time (over baseline)



(b) Total cost (under baseline)

Figure 2: Effectiveness of top-$k$ ACIC recommendations

## 5. ACKNOWLEDGEMENTS

Thanks go to our reviewers. This work was supported by the National High Technology Research and Development Program of China (863 Program) No. 2010AA012403.

## 6. REFERENCES

[1] Amazon. aws.amazon.com/ec2/hpc-applications/, 2011.

[2] M. Liu, J. Zhai, Y. Zhai, X. Ma, and W. Chen. One optimized i/o configuration per hpc application: leveraging the configurability of cloud. In *APSys*, 2011.

[3] L. Olshen and C. Stone. Classification and regression trees. *Wadsworth International Group*, 1984.

[4] R. Plackett and J. Burman. The design of optimum multifactorial experiments. *Biometrika*, 1946.

[5] H. Shan and et. al. Characterizing and Predicting the I/O Performance of HPC Applications Using a Parameterized Synthetic Benchmark. In *SC*, 2008.

[6] Y. Zhai, M. Liu, J. Zhai, X. Ma, and W. Chen. Cloud versus in-house cluster: evaluating amazon cluster compute instances for running mpi applications. In *SC*. ACM, 2011.