Automatic Cloud I/O Configurator for Parallel Applications

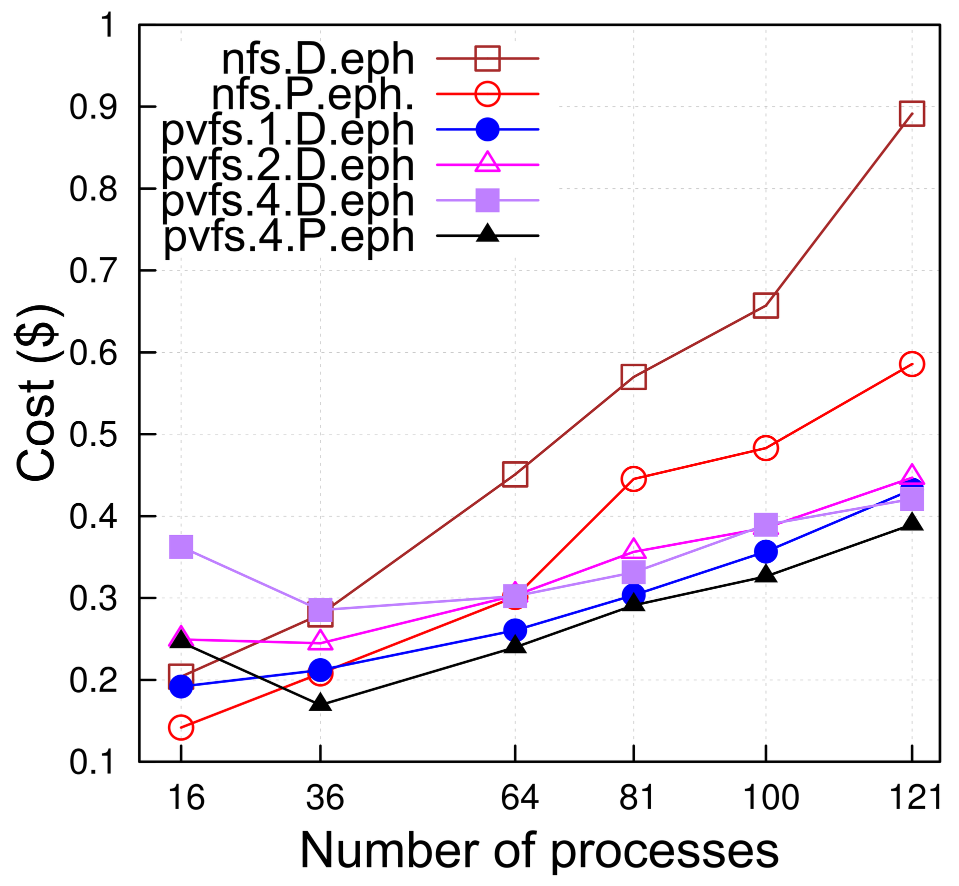
Mingliang Liu←, Ye Jin→, Jidong Zhai←, Yan Zhai←, Qianqian Shi←, Xiaosong Ma→, Wenguang Chen←

✉ liuml07@mails.tsinghua.edu.cn http://hpc.cs.tsinghua.edu.cn/ACIC

1. Problem

There is a trend to run HPC applications in cloud. However, the cloud amplifies the increasing I/O gap. Configuring I/O system dedicatedly is feasible but challenging as:

1. Configuration space is pretty large
2. Virtualization increases the complexity of the I/O system
3. The optimal configurations for performance and cost contradict



(Total cost of running BTIO with typical I/O configurations)

2. Exploration Space

Name	Value	Rank
I/O system options in cloud		
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**
Workload characteristics		
Num. of all processes	32, 64, 128, 256	14
Num. of I/O processes	32, 64, 128, 256	**4**
I/O interface	POSIX, MPI-IO	**9**
I/O iteration count	1, 10, 100	13
Data size	{1,4,16,32,128,512}MB	**1**
Request size	256KB, {4,16,128}MB	**8**
Read and/or write	read, write	**2**
Collective	yes, no	11
File sharing	share, individual	15

We studied 12 applications in several scientific areas with different scale (32 to 256).

- It's impossible to analyze the 15-D manually (over $1M$ combinations).
- Only the top 10 are considered, whose ranks are directed by PB matrix [2]

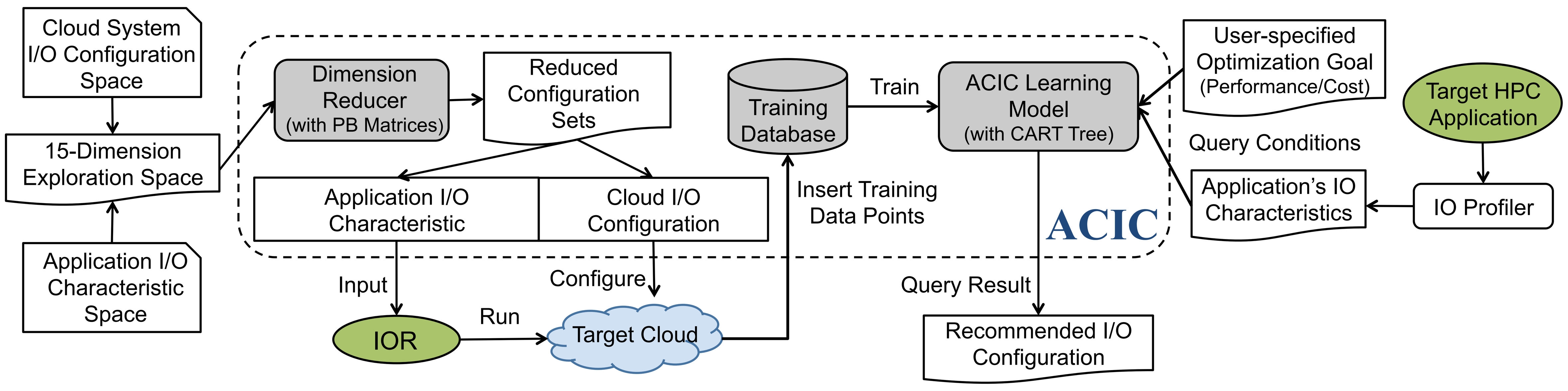
3. Contributions

We propose ACIC (**A**utomatic **C**loud **I**/O **C**onfigurator), which automatically searches optimized I/O system configurations from candidates for each individual application.

We make the black-box approach affordable on clouds by cost-saving mechanisms:

1. Enable *reusable training* by adopting a generic synthetic I/O benchmark
2. *Reduce* the exploration space by determining ranks of the parameters
3. Propose the potential of building a *shared*, public training database

4. Approach



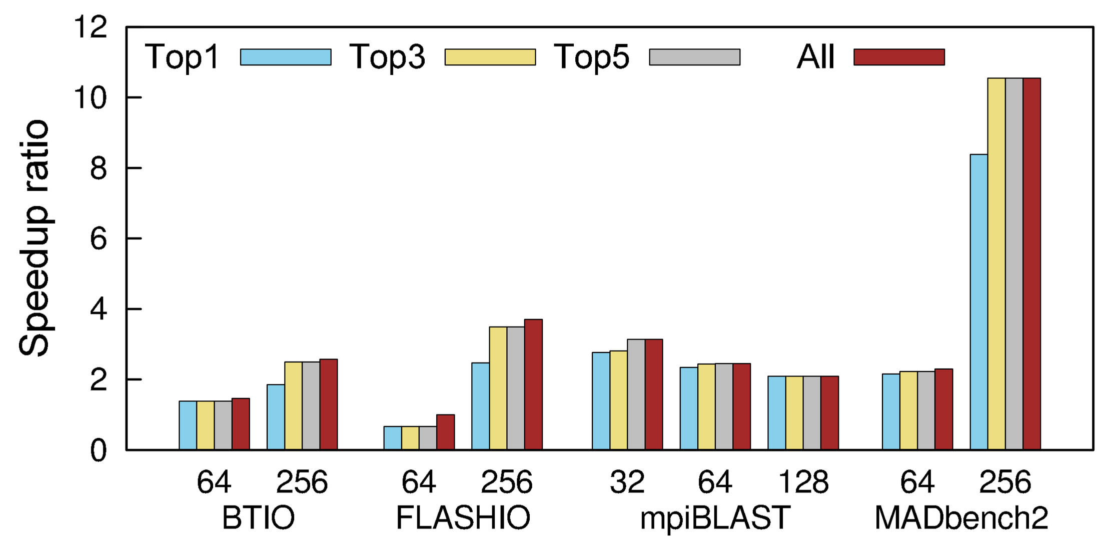
ACIC employs a machine learning model named classification and regression trees (CART [1]). We use the IOR [3] synthetic benchmark to collect training data (performance and cost).

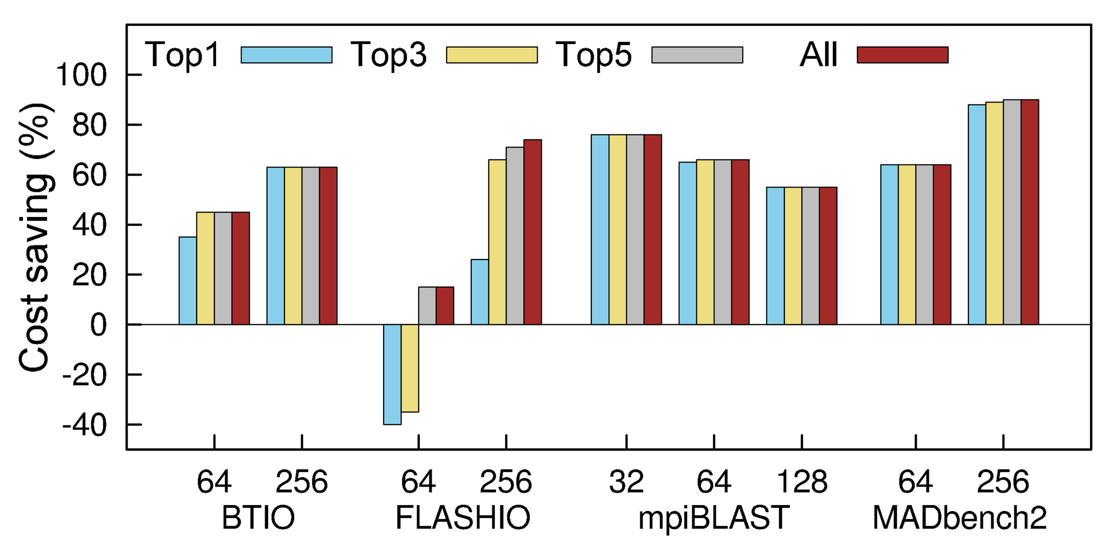
- **Input**: I/O characteristics of one individual application (see the table in Section 2)
- **Output**: the top predicted I/O configurations from all trained candidates

5. Evaluation

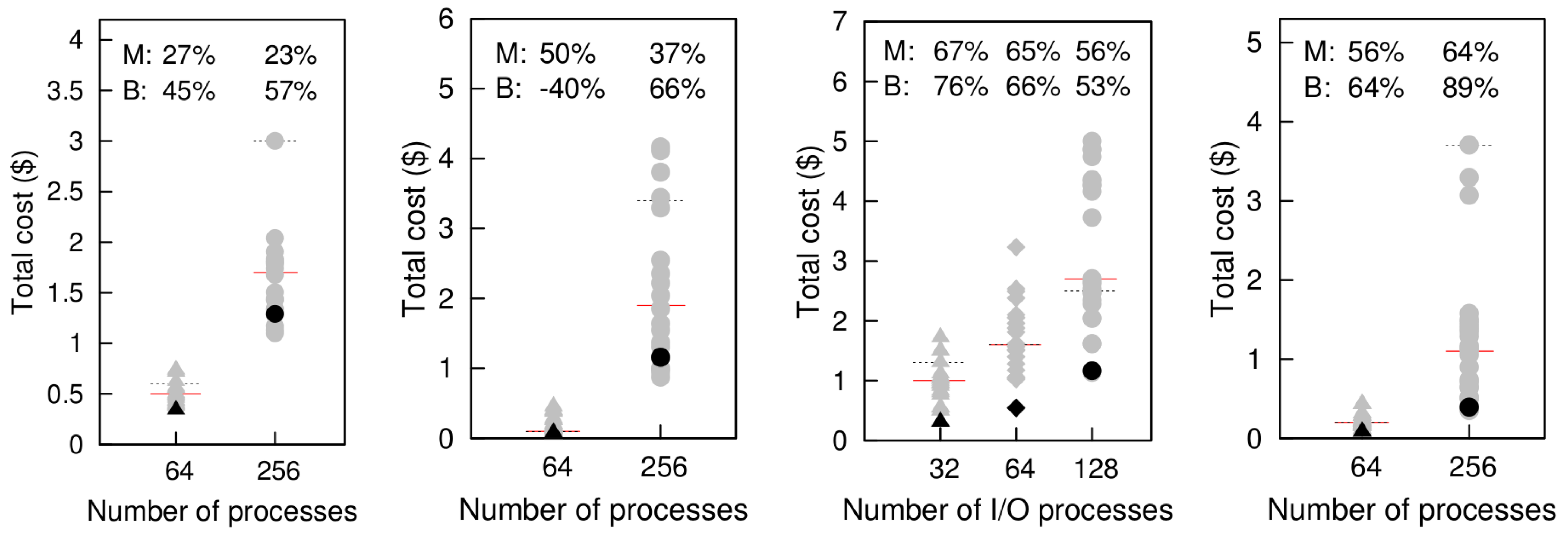
Experiments are performed on Amazon EC2 Cluster Computing Instances. The baseline configuration is the simple but popular dedicated NFS server attaching EBS disks. Following are performance and cost of top 1, 3, and 5 ACIC predicted and all candidate configurations.

$$speed\_up = \frac{Time\_baseline}{Time\_ACIC} \qquad cost\_saving = \frac{Cost\_baseline - Cost\_ACIC}{Cost\_baseline} \times 100\%$$





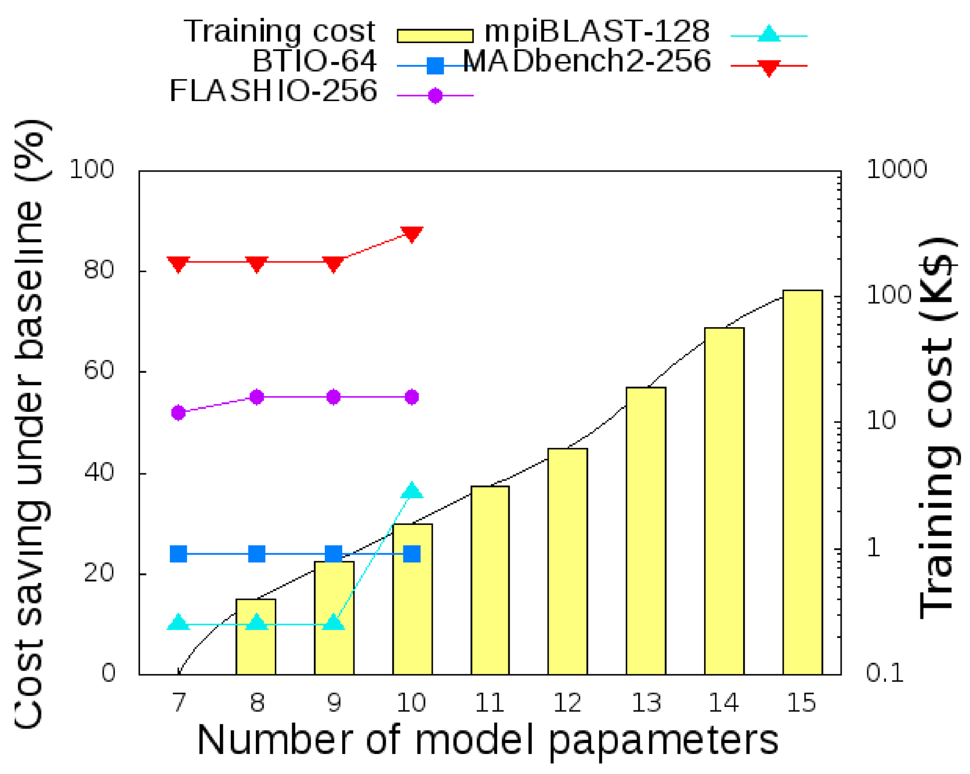
1. The top 1 works fairly well although considering more top candidates helps sometimes
2. Little further gain can be achieved by checking beyond the top 3 ones



(a) BTIO (b) FLASHIO (c) mpiBALST (d) MADbench2

(Black Dot: ACIC predicted; Red Solid Line: Median; Black Dotted Line: Baseline)

6. Overhead



- More parameters → Higher accuracy
- The training cost grows exponentially
- Amortize the cost by sharing database

7. Conclusion

ACIC is the first automatic cloud I/O system configuration tool for HPC applications which enables application-dependent performance/cost optimization.

Scan Barcode!

References

[1] Olshen and C. Stone. Classification and Regression Trees. Wadsworth International Group, 1984.

[2] Plackett and J. Burman. The Design of Optimum Multifactorial Experiments. *Biometrika*, 1946.

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