中国科学院深圳先进技术研究院
SHENZHEN INSTITUTES OF ADVANCED TECHNOLOGY
CHINESE ACADEMY OF SCIENCES

USENIX ATC’ 25

Swift: Fast Performance Tuning with GAN-Generated Configurations

Chao Chen[1], Shixin Huang[1], Xuehai Qian[2], Zhibin Yu[1,3]

July, 9th, 2025

1

清華大學 Tsinghua University

2

3

Background

- Big data processing requires various frameworks

- Configurations play a vital role in performance tuning
  - Performance can be improved by up to 89x [Yu18]

Framework	Configuration Parameters-Description	Range	Default
Flink	**parallelism.default** - The default parallelism for programs that have no parallelism specified.	1-24	1
	taskmanager.numberOfTaskSlots - # of parallel operator or function instances a TaskManager runs.	1-8	1
	jobmanager.heap.mb - JVM heap size (in megabytes) for the JobManager.	1024-4096	1024
	taskmanager.heap.mb - JVM heap size (in megabytes) for the TaskManager.	1024-10240	1024
	taskmanager.memory.fraction - The relative amount of memory that the task manager reserves.	0.5-0.8	0.7
Spark	**spark.shuffle.file.buffer** - Size of the in-memory buffer for each shuffle file output stream, in KB.	2-128	32
	spark.executor.cores-The number of cores to use on each executor.	4-30	core #
	spark.driver.memory-Amount of memory to use for the driver process, in MB.	1024-36864	1024
	spark.executor.memory-Amount of memory to use per executor process, in MB.	7168-36864	1024
	spark.memory.fraction-Fraction of (heap space - 300MB) used for execution and storage.	0.5-1	0.75

Yu et al. 2018. Datasize-Aware High Dimensional Configurations Auto-Tuning of In-Memory Cluster Computing. ASPLOS' 18. ACM, New York, NY, USA, 14 pages.

Background

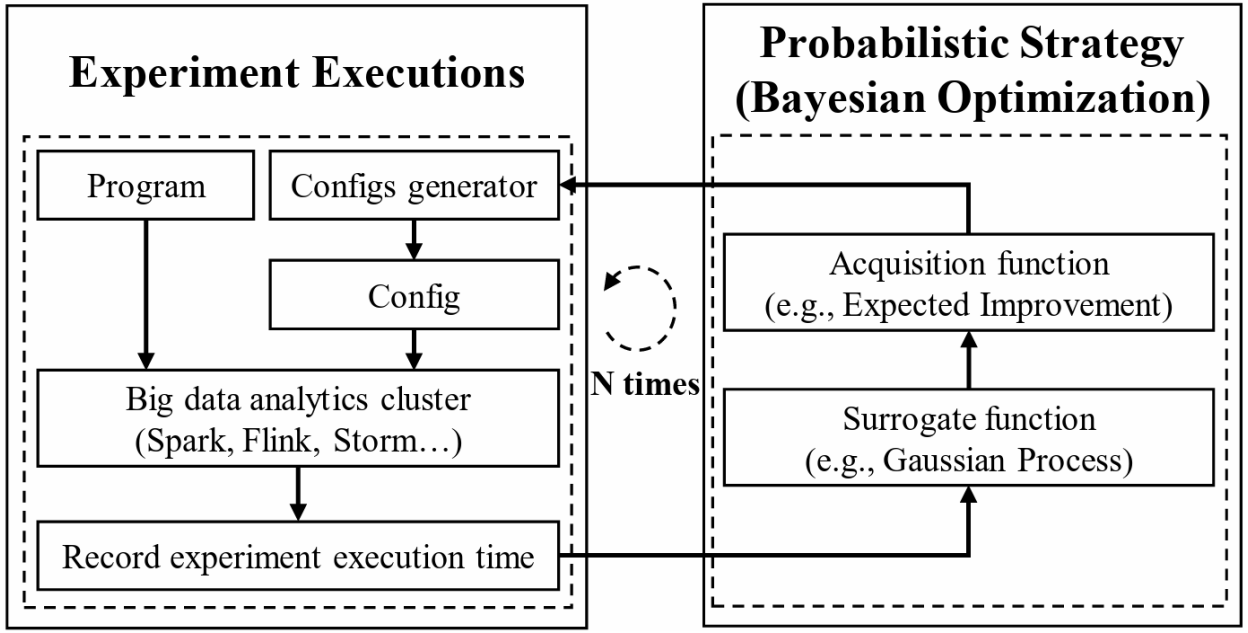
- Rule- and analytical tuning methods
  - Leverage the experiences of human experts
  - Analytical models for precise evaluation
  - Fail for complex systems

- Simulation tuning approaches
  - Probe system internals
  - Hard to cover all factors

- Machine Learning (ML) tuning methods
  - ML performance models for evaluation
  - Samples in large quantities

ML methods become popular for complicated parameter tuning

Background



Workload	Collecting(h)	Modeling(s)	Searching(m)
KMeans	92	10	7
Bayes	60	11	9
PageRank	67	9	10
TeraSort	68	11	8
WordCount	82	12	7
NWeight	53	12	9

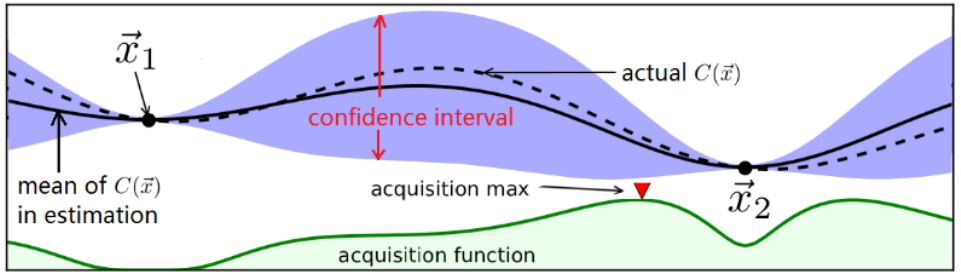
ML models need a large amount of samples

- Accurate modelling
- Sample collection is extremely time-consuming [Chen24]

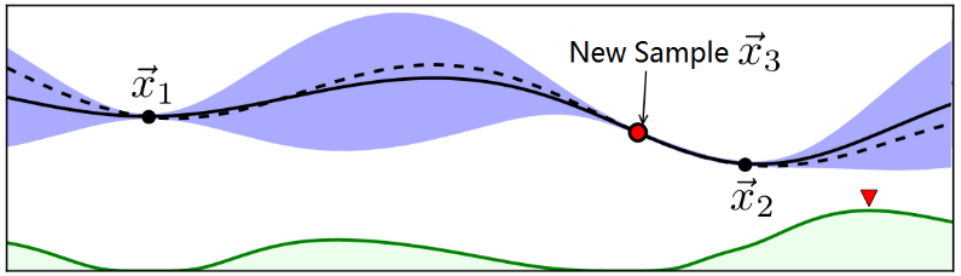
C. Chen, J. Xin and Z. Yu, "TIE: Fast Experiment-Driven ML-Based Configuration Tuning for In-Memory Data Analytics," in IEEE Transactions on Computers, vol. 73, no. 5, pp. 1233-1247, May 2024

Background

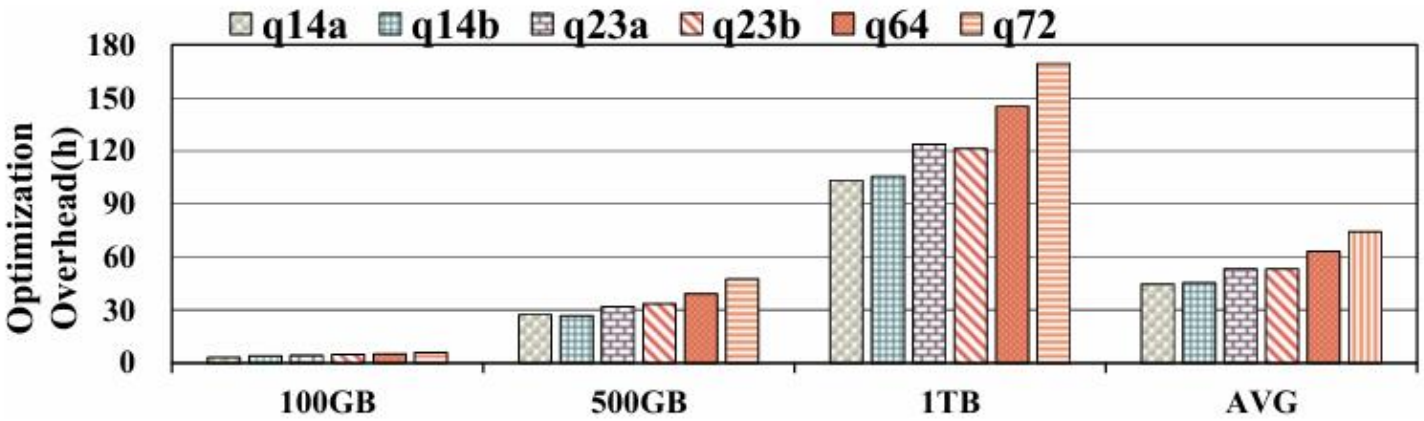
- Bayesian optimization (BO) is adopted to speed up [Alipourfard17]
- BO is slow when data size increases
  - Over 100 hours for 1TB data



(a) t = 2



(b) t = 3



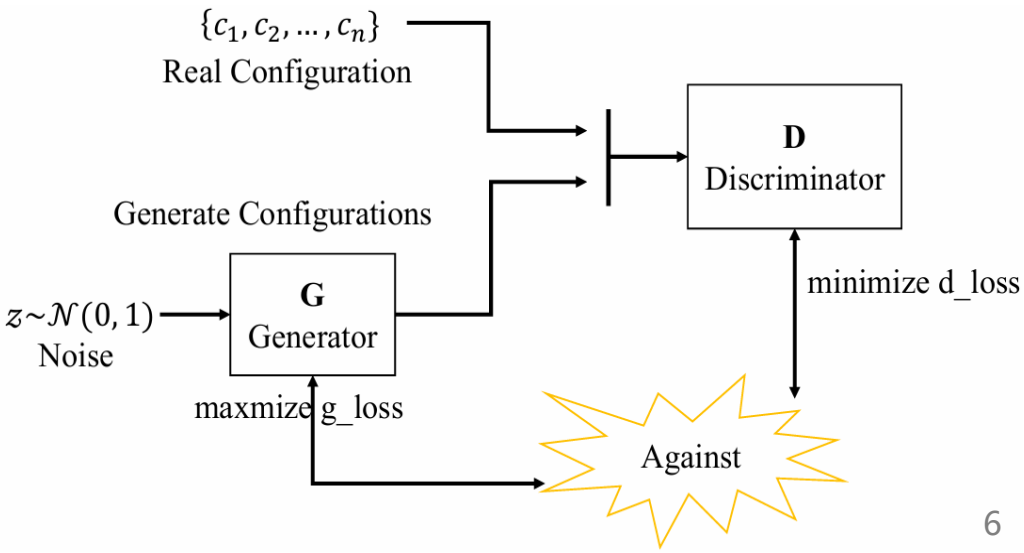
Alipourfard, Omid, Hongqiang Harry Liu, Jianshu Chen, Shivaram Venkataraman, Minlan Yu, and Ming Zhang. "Cherrypick: Adaptively Unearthing the Best Cloud Configurations for Big Data Analytics." In Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation, 469–82. NSDI' 17. USA: USENIX Association, 2017.

Key Idea

- Configuration as a vector: $\{c_1, c_2, \dots, c_n\}$

- Target and generated configurations: "*similar but not too similar*"
  - "similar" to ensure configurations are similar to real applications
  - "not too similar" to allow exploration
  - If the target is a good configuration, it guarantees the next sample selection starts from a good sample

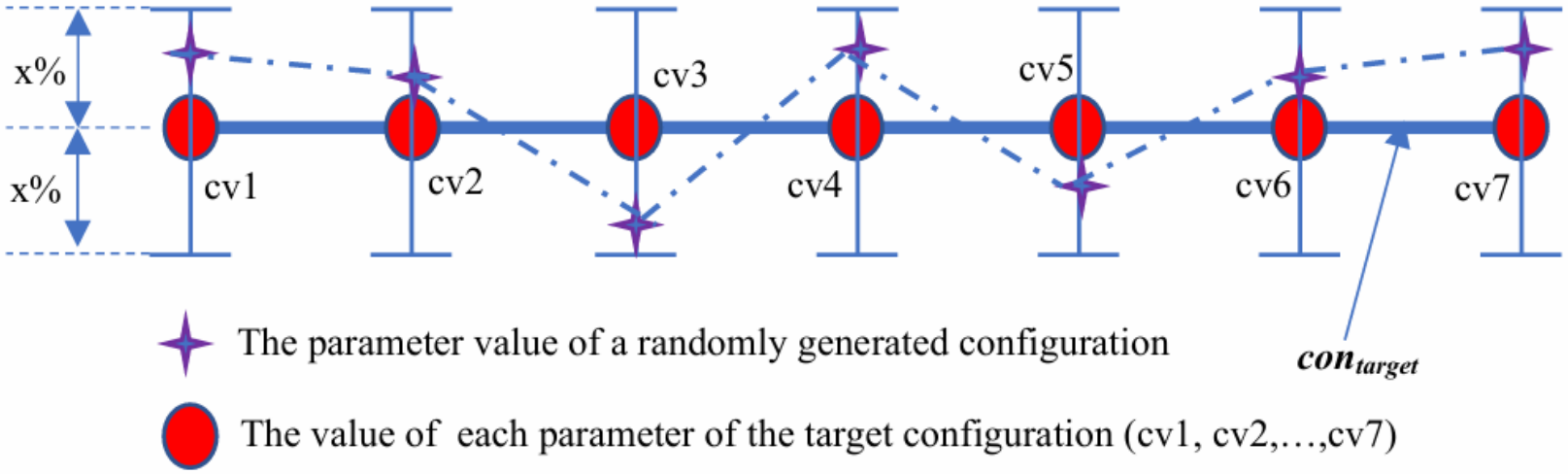
- Generative adversarial network (GAN)
  - Generate distribution similar to the input



Key Idea

- Configuration similarity measurement

- One-neighborhood (ON): Manhattan distance (MD)
  - $MD(conv_1, conv_2) = |c_{11} - c_{21}| + \cdots + |c_{1n} - c_{2n}|$
  - Configuration based on small random perturbations

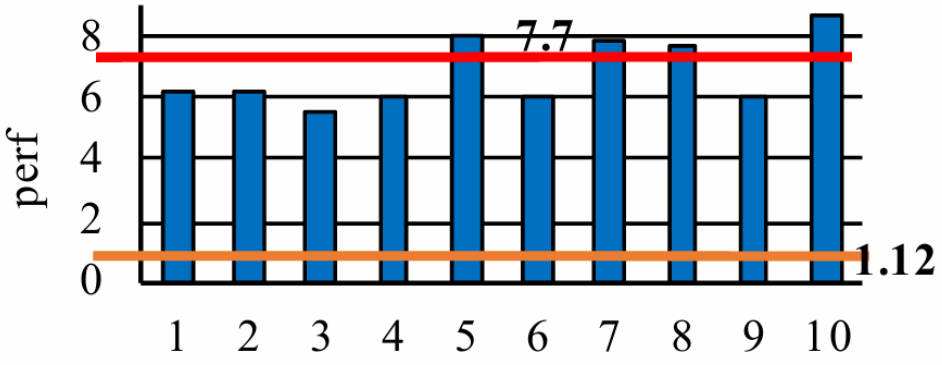


Only vector distances are used.

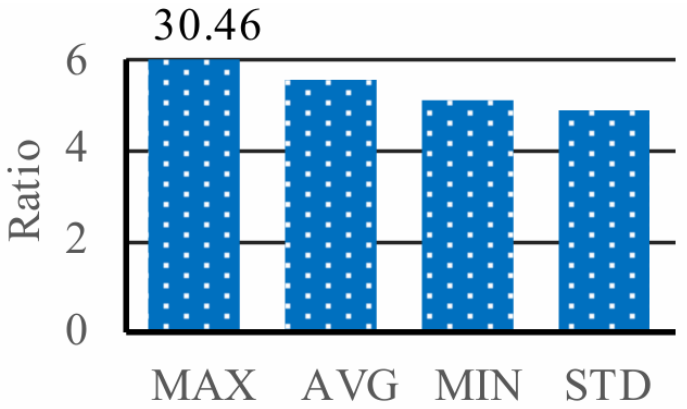
No distributions between configurations!

Key Idea

- Jensen-Shannon (JS) divergence measurement
  - Widely used for quantifying two probability distribution similarities
  - Inherently used by GAN as a loss function to ensure resemblance



(a) perf with GAN generated configurations



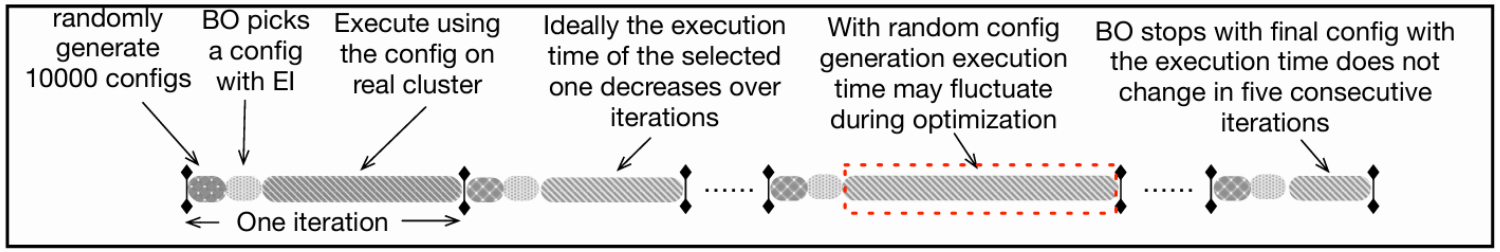
(b) JS comparison with ON

Not only short distance, but only similar distributions of element values

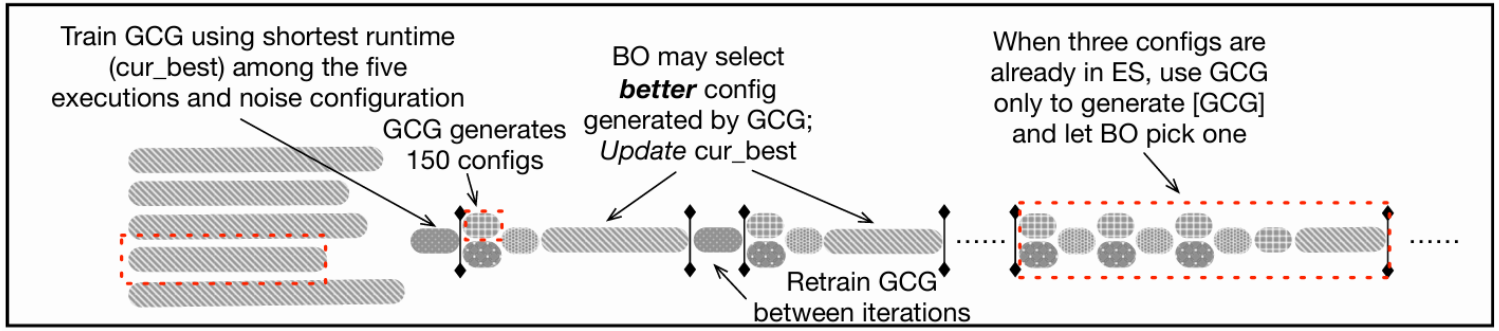
Swift: Fast Performance Tuning with GAN + JS divergence

Key Idea

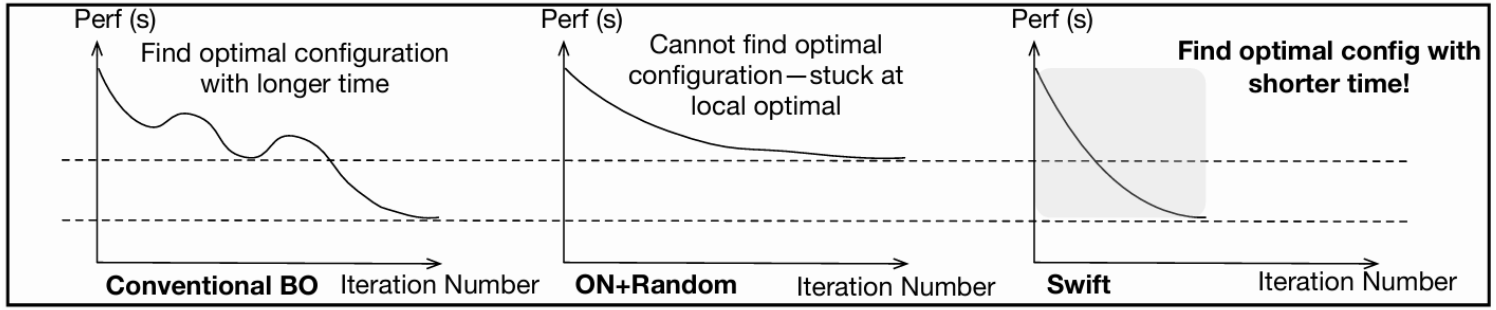
- Swift reduces the BO time
  - Mixing random and GAN-generated configurations that have similar distributions with the "current best"
- Swift outperforms conventional BO and ON+Random methods



(a) Conventional BO: optimization time dominated by # of iterations, i.e., how fast real execution time decreases



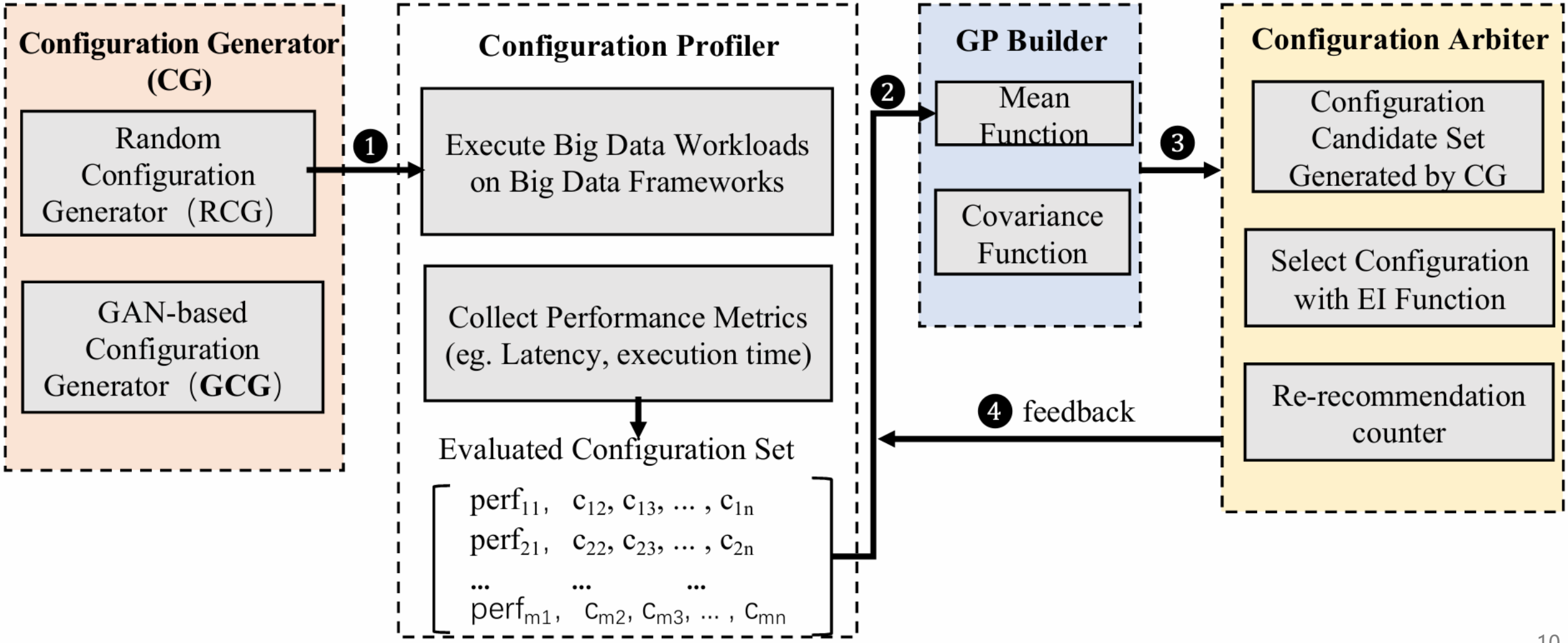
(b) Swift: execution time smoothly decreases due to configs from GCG ➜ less # of iterations ➜ faster convergence



(c) Conceptual demonstration of performance over iteration/time in three methods: i) conventional BO; ii) Use ON instead of GCG; iii) Swift: use a mix of configurations generated by GCG and generated randomly

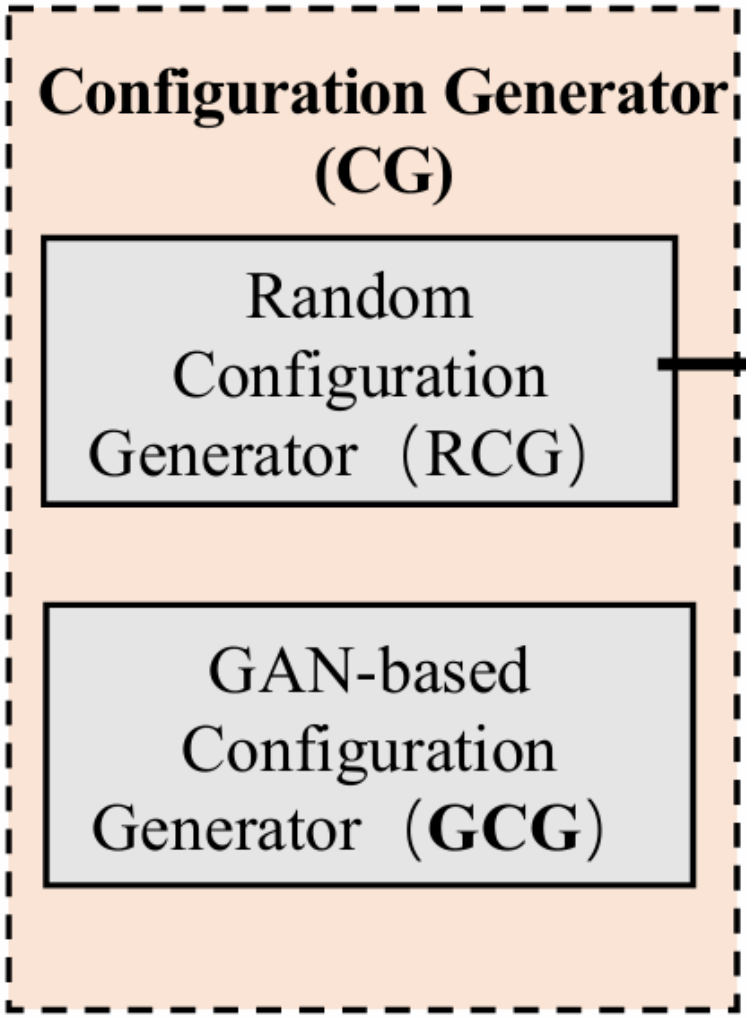
Design

- Four Components



Design

- Configuration Generator (CG)
- Random Configuration Generator (RCG)
- GAN-based Configuration Generator (GCG)



Design

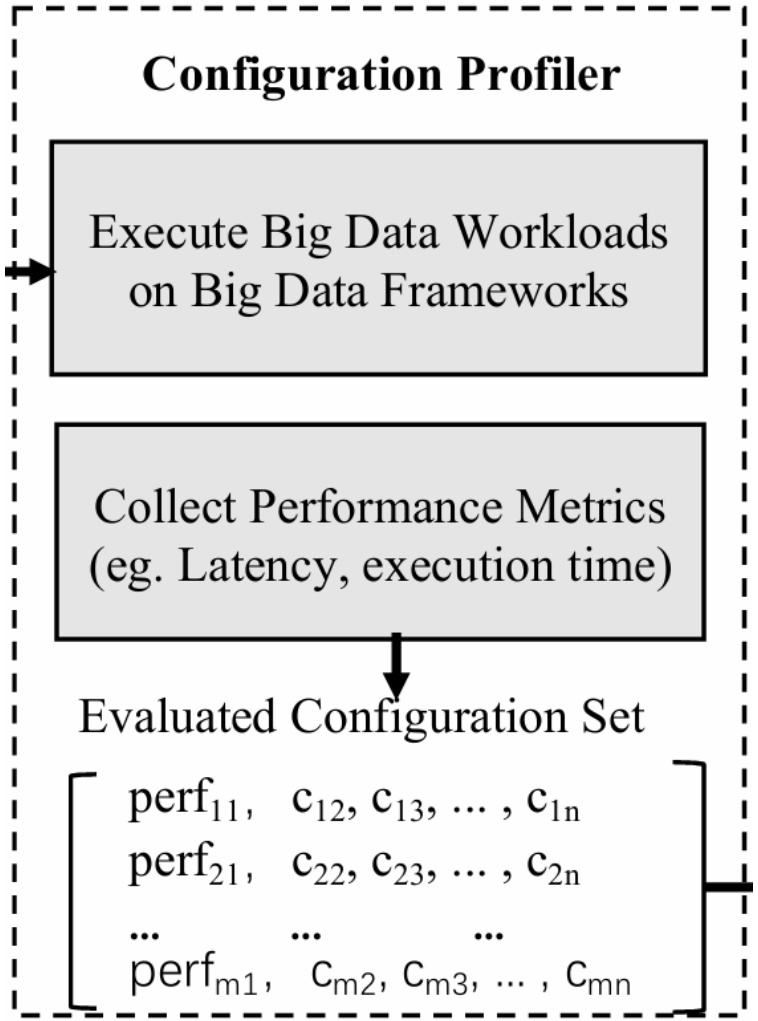
- Configuration Profiler

- Configuration vector

$$conv = \{c_0, c_1, \ldots, c_i, \ldots, c_{n-1}\}$$

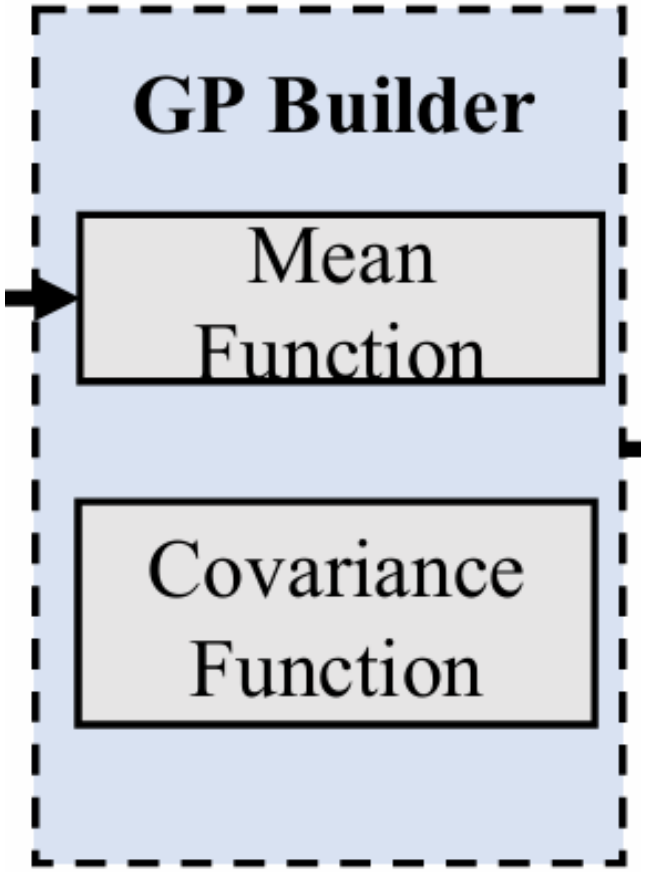
- Sample
  - $spv = \{pf, conv\}$,pf is the performance

- Profiling performance and configurations as Evaluation Set (ES) matrix



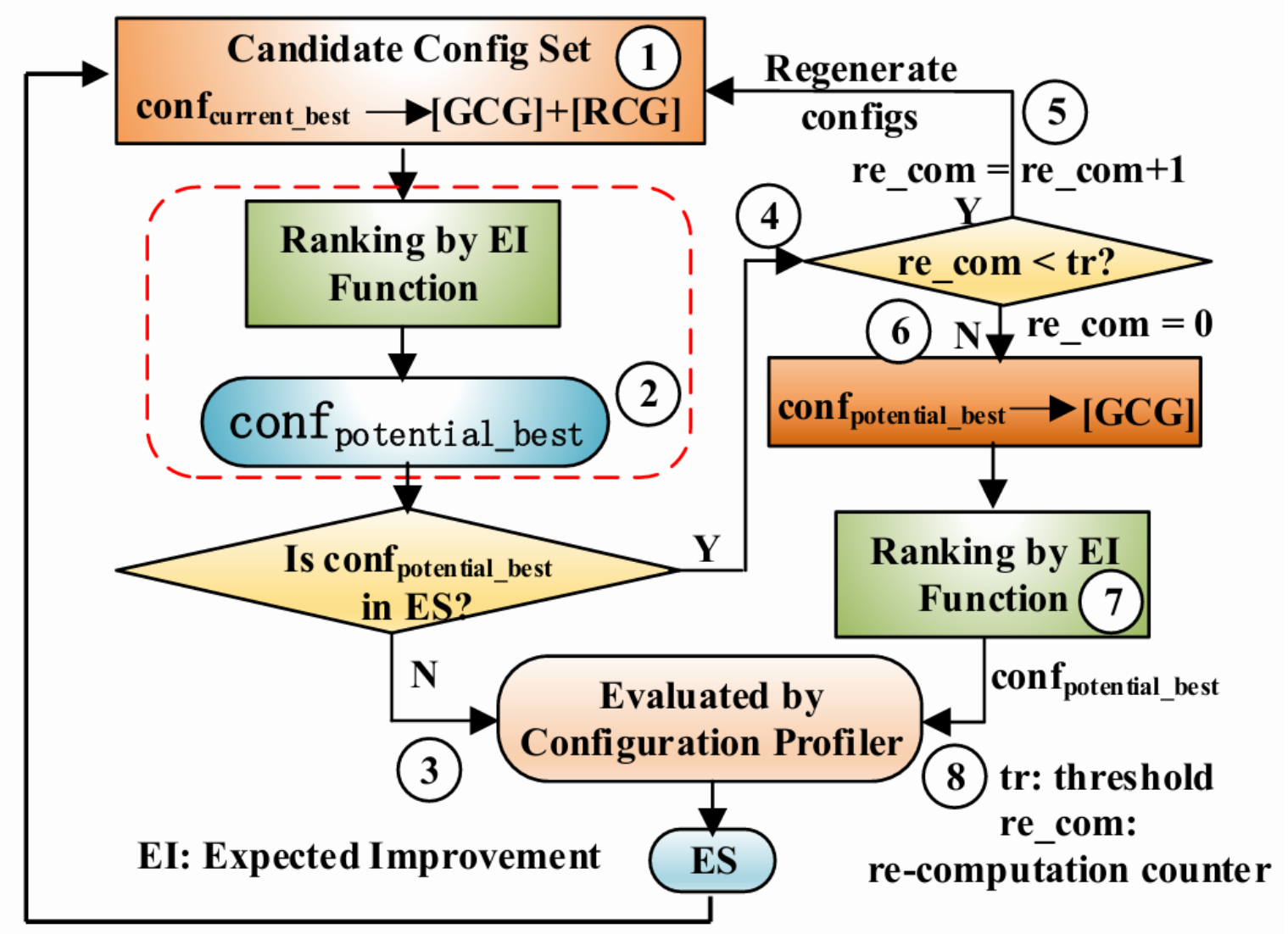
Design

- Gaussian Process (GP) Builder
- Configuration Profiler to generate vectors
- Compute performance mean and covariance
- Smooth kernel: Matern5/2



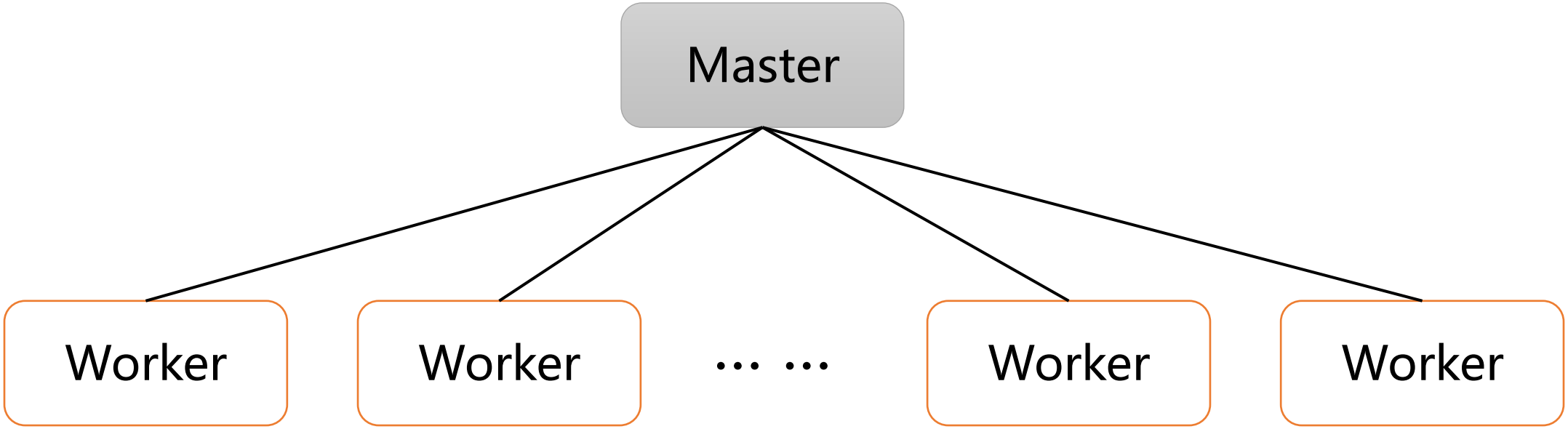
Design

- BO iteration
- Random RCG + GAN GCG, potential perf.
- Regenerate if not in Evaluation Set (ES)
- Evaluate and add to ES



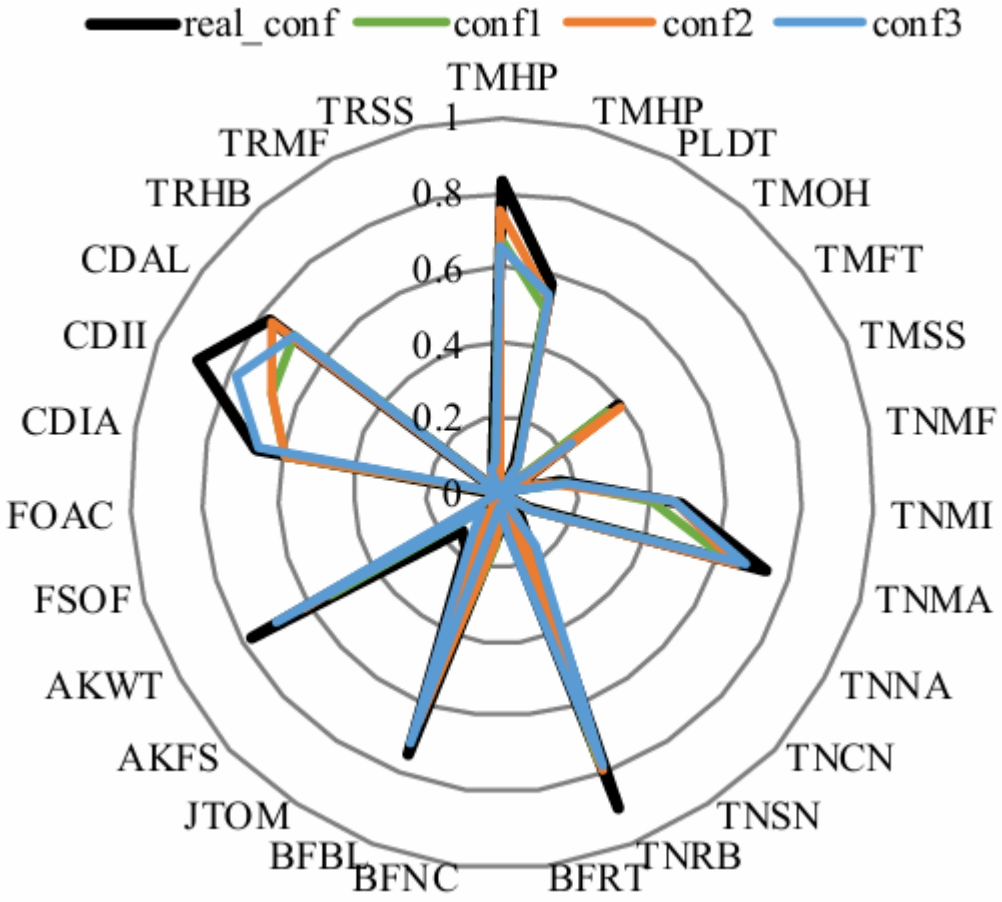
Evaluation

- Eight servers
  - 1 master + 7 workers
  - Intel (R) Xeon (R) CPU E5-2630 v3 @2.40GHz
  - 64 GB memory
  - Spark + Flink applications



Evaluation

- Quality of GAN-generated configurations
- Flink program with 27 configs
- 1 real + 3 GAN configurations
- Similar with small MDs

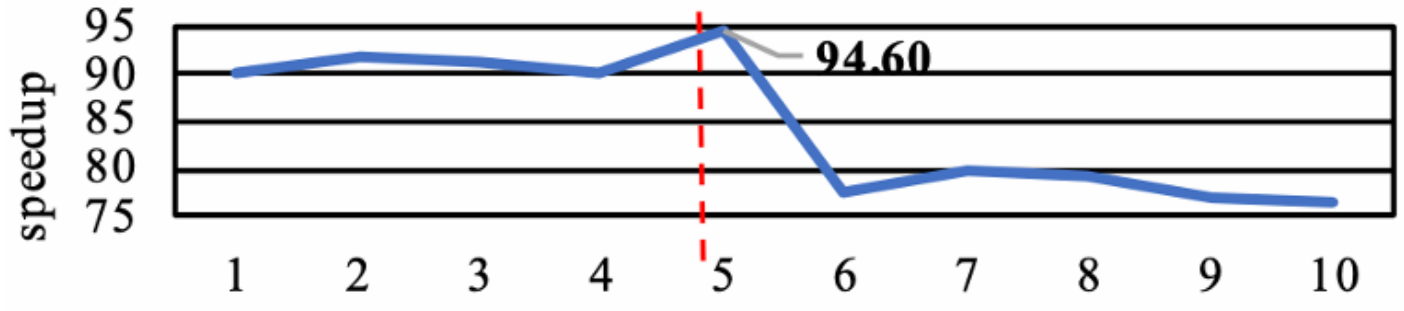


The quality of GCG generated configurations

Evaluation

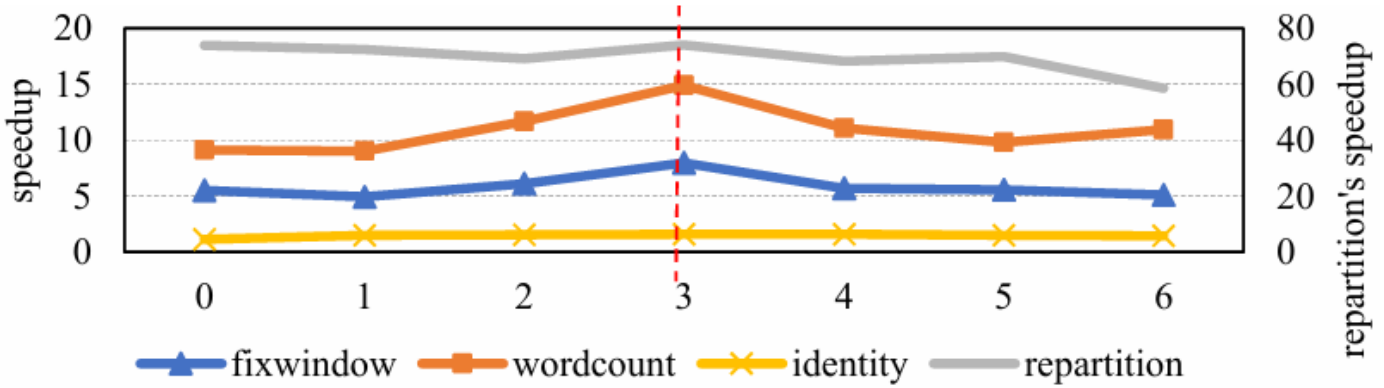
- Hyper-parameter evaluation

- Initial sample number vs. speedup



The speedup with different number of initial configurations for repartition over default configurations.

- Tolerance threshold vs. speedup



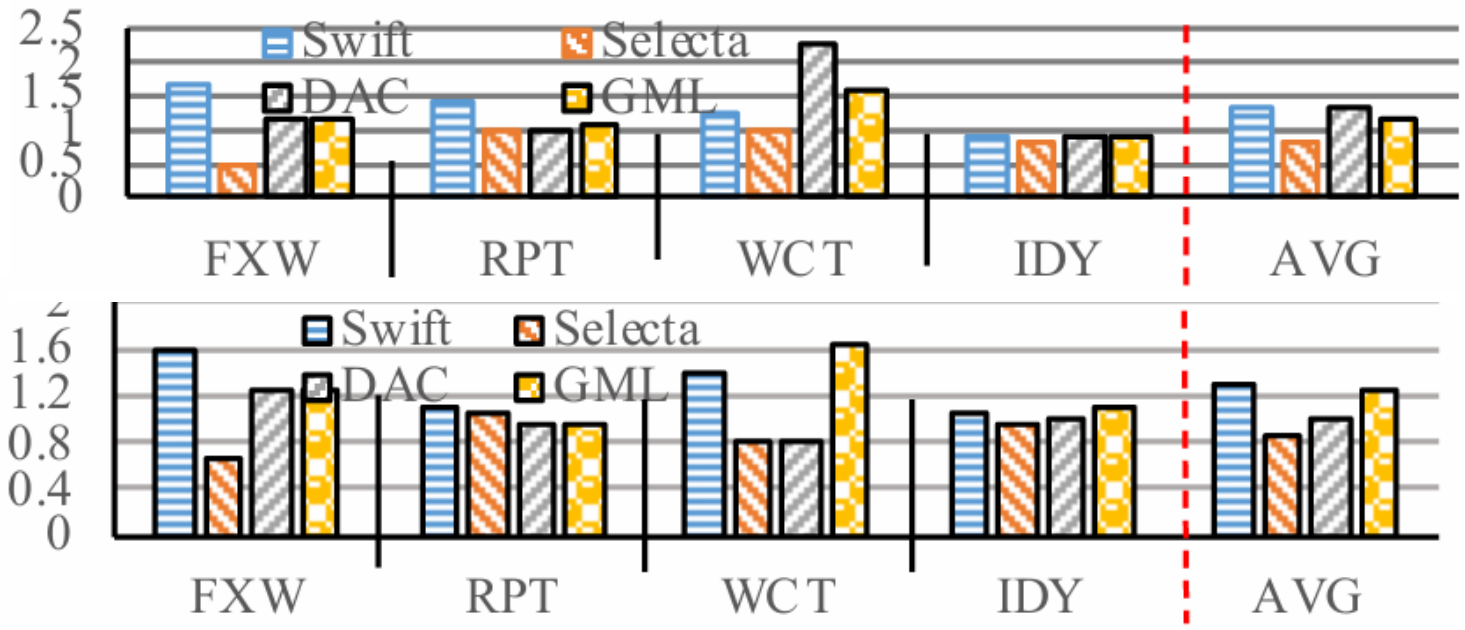
The performance variation with different threshold values. The X axis represents the threshold values.

Evaluation

- Flink optimization time
  - CherryPick,Selecta,DAC,GML
  - 2.2×, 4.3×, 9× and 7.2× that of Swift

Bench	Swift	CherryPick	Selecta	DAC	GML
Fixwindow	25	50+	100	750	500
Repartition	17	50+	100	550	440
Wordcount	23	50+	100	600	460
Identity	24	50+	100	600	420
In Total	89	200+	400	2,500	1820

- Latency and throughput
  - 1.28x improvement on average

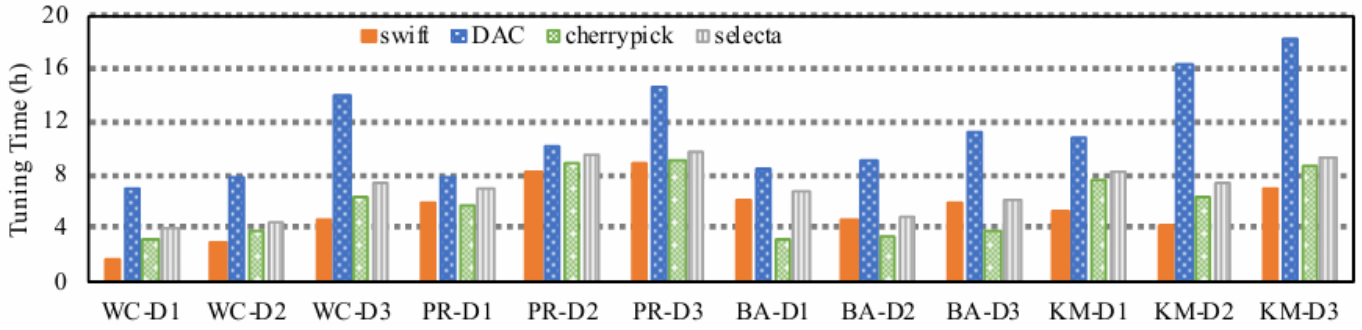


Latency and throughput by Swift, Selecta, DAC, and GML

Better performance with shorter time for Flink

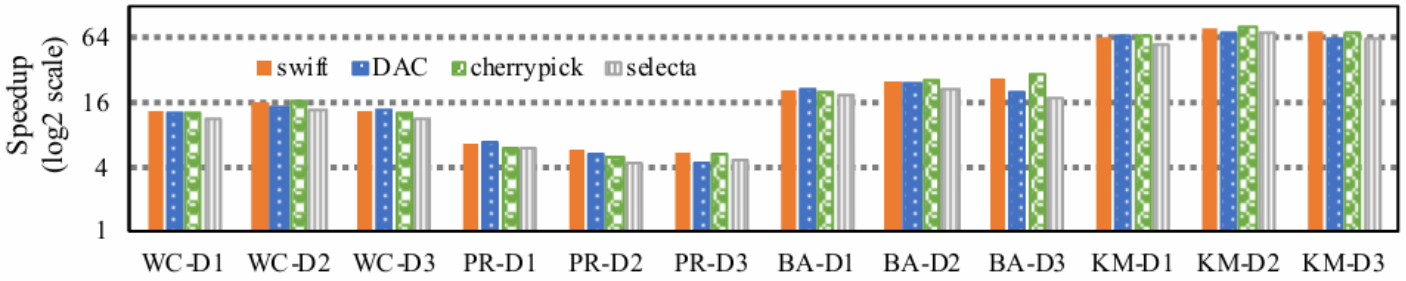
Evaluation

- Spark optimization time
  - DAC,CherryPick,Selecta
  - 2.2×, 1.2×, and 1.4× that of Swift



Optimization time comparison.

- Speedup comparison
  - Swift achieves higher speedups than others



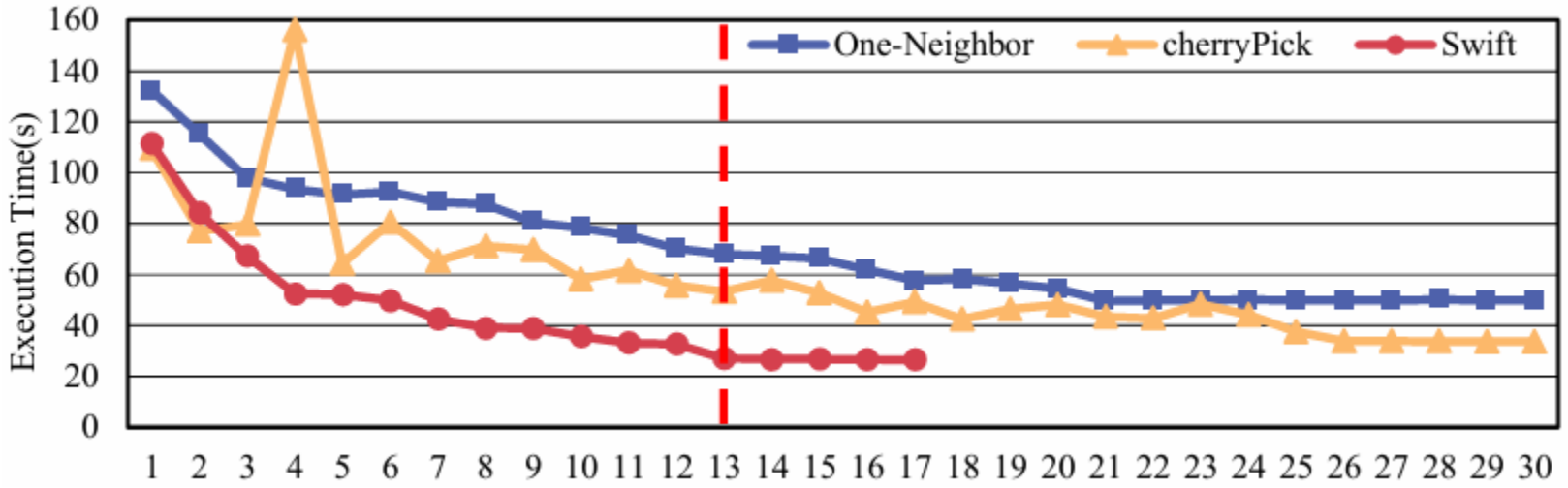
Speedup comparison. The baseline is the execution time of each program with default configuration.

Better performance with shorter time for Spark

Evaluation

- Swift: mixing random and GAN-generated configurations
  - Skew the search space toward the optimal configuration
  - Faster convergence

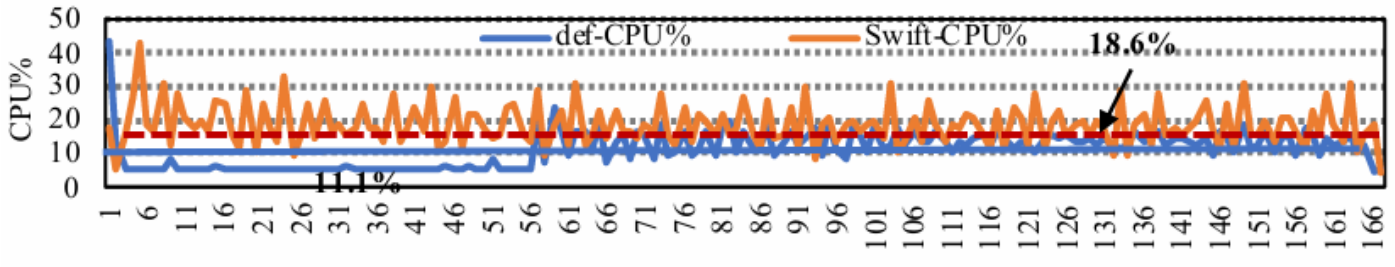
- CherryPick: uniform space search
  - Fluctuations throughout the process



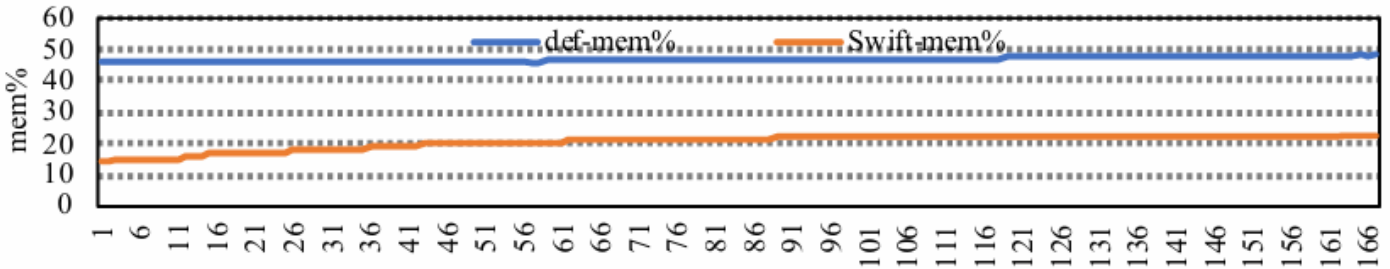
The performance of Spark WordCount tuned by the configurations selected by ON, CherreyPick(BO), and Swift during iterations. The X axis denotes the iteration number.

Evaluation

- Case study on *FixWindow*
- Speedup by 10.7× w.r.t. default configurations
- CPU utilization increases by 7.5%
- Memory utilization drastically reduces from 46.8% to 20.1%.



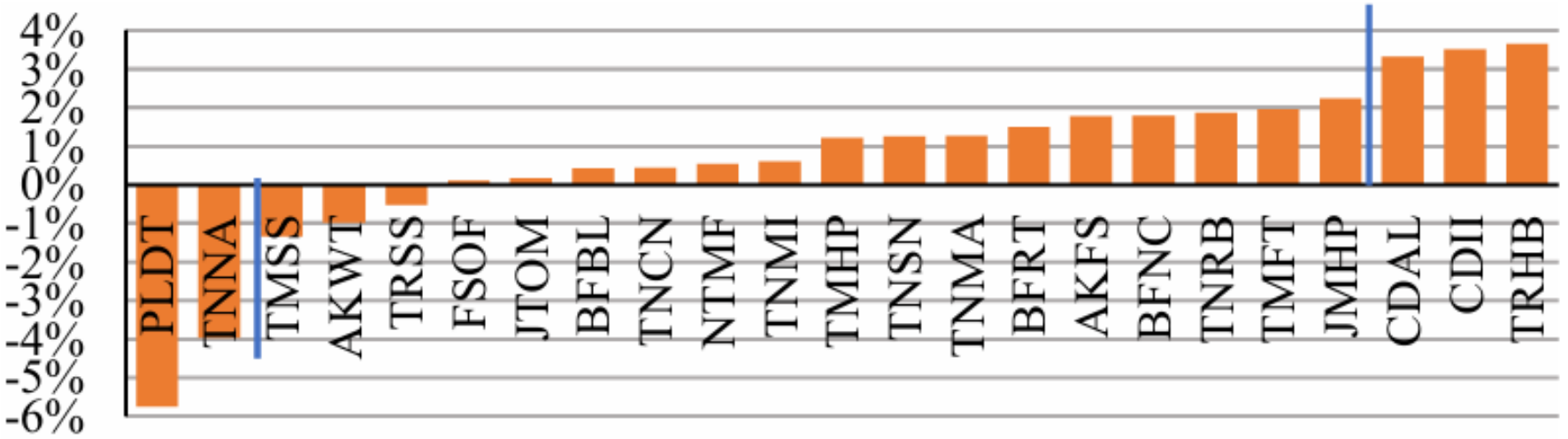
CPU utilization of Fixwindow tuned by Swift and with default configuration.



Memory utilization of Fixwindow tuned by Swift and with default configuration.

Evaluation

- TRHB, CDII, and CDAL improve the performance
  - *taskmanager.runtime.hashjoin-bloom-filters*
  - *compiler.delimited-informat.min-line-samples*
  - *compiler.delimited-informat.max-sample-len*
  - All three parameters are **related to memory**
  - Bloom filter reduces the memory consumption



The importance quantification of configuration parameters for Fixwindow with respect to performance.

Conclusion

Swift

- Conventional Bayesian optimization is slow for configuration optimization with search fluctuations
- Swift adopts GAN network and a novel Jensen-Shannon divergence mechanism to address the challenges
- Swift provides better performance with shorter time for both Spark and Flink data analytics programs

Thank you!