A Study on Redis Parameter Tuning Based on Non-linear Machine Learning

연세대학교 컴퓨터과학과 서주연

2021년 07월

SW STAR LAB

Software Technology Advanced Research

과제명: IoT 환경을 위한 고성능 플래시 메모리 스토리지 기반 인메모리 분산 DBMS 연구개발

과제번호: 2017-0-00477

과학기술정보통신부
Ministry of Science and ICT

연세대학교
YONSEI UNIVERSITY

IITP 정보통신기술진흥센터
Institute for Information & communications Technology Promotion

목차

01 Introduction

02 Model

03 Experiment and Analysis

04 Conclusion

목차

01 Introduction

02 Model

03 Experiment and Analysis

04 Conclusion

01 Introduction

- High performance services to handle real-time data

- **In-Memory Database**
  - » Use main memory as data storage
  - » Respond faster than disk-based database

- **Redis**
  - » Low delay for data access
  - » Save data as Key-Value pairs
  - » Persistence Methods
    - – To preserve data from DRAM volatility

01 Introduction

- **Redis Persistence method**
  - » RDB (Redis Database)
    - – Take snapshots at regular intervals
  - » AOF (Append-Only File)
    - – Generate log records for commands that change the dataset
    - – Append them to the log file
  - ➔ Performance Degradation
    - – Delaying data processing
    - – Additional memory usage

- **Background Operation**
  - » Single thread program
  - » E.g., Closing connections of clients in timeout, purging expired keys that are never requested, and so forth
  - » Data processing performance degradation still occurs

01 Introduction

- **Redis Parameter Tuning**
  - » Find optimal parameter values for different workloads
  - » The wide range of parameters and values

- **Utilize Machine Learning Methods**
  - » OtterTune
    - – Use Supervised, Unsupervised Learning
    - – Find the optimal parameter values for a particular workload through the results obtained from different workloads
    - – Consider only linear relationships of extracted data
  - » **RS-OtterTune (Redis Simplified OtterTune)**
    - – Applying non-linear machine learning methods to Redis - RandomForest, XGBoost
    - – Up to 45.9% performance improvement over default parameter setting

목차

01 Introduction

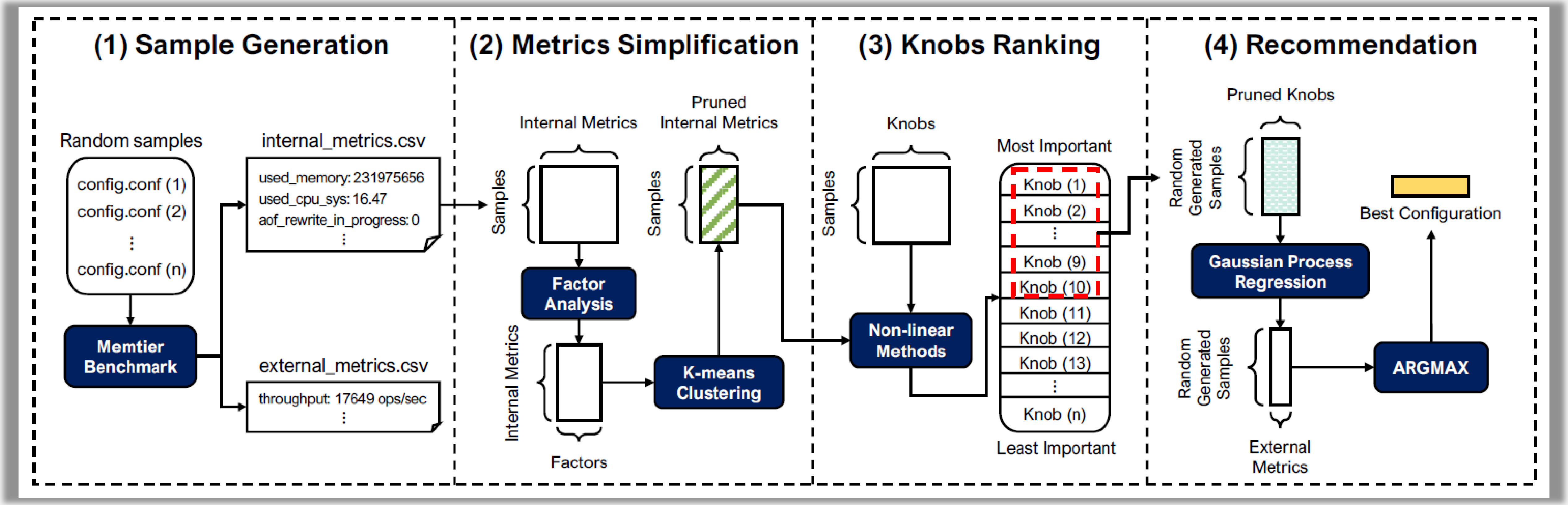
02 Model

- Sample Generation
- Metrics Simplification
- Knobs Ranking
- Recommendation

03 Experiment and Analysis

04 Conclusion

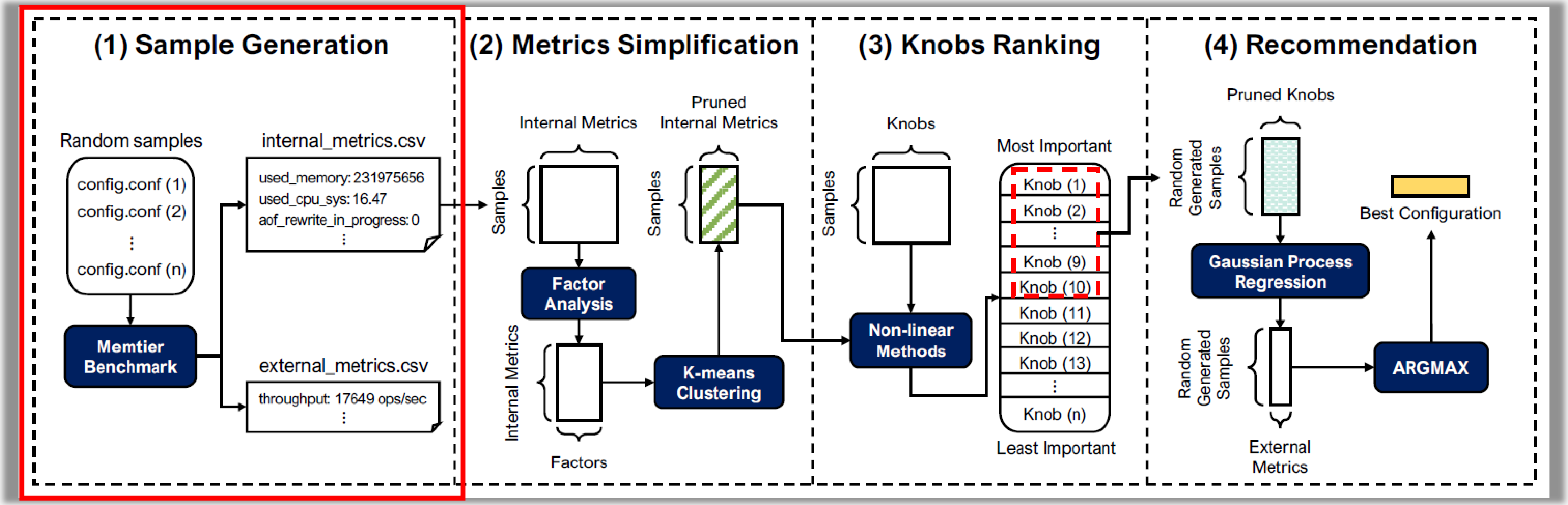
02 Model



RS-OtterTune Model Architecture

02 Model

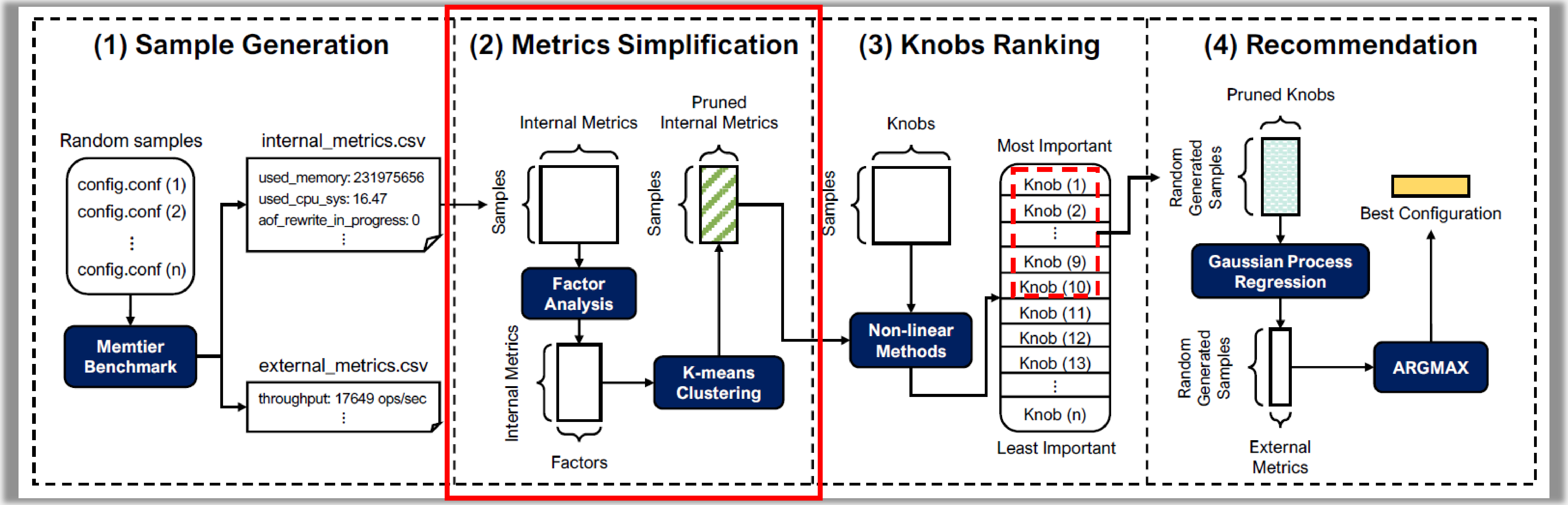
└ Sample Generation



- Generate Redis configuration files with random values for each parameter
- Measure Internal / External metrics with Memtier-benchmark and save them to each file

02 Model

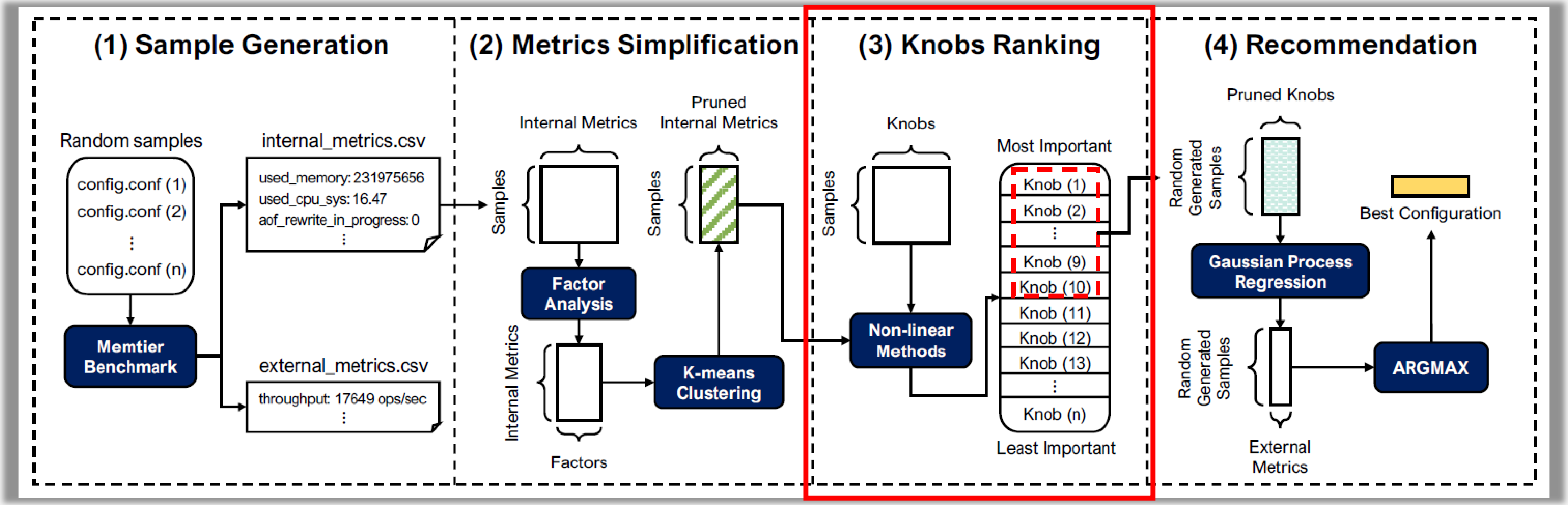
└ Metrics Simplification



- Select internal metrics with similar characteristics and simplify them
- Find variance for correlations between internal metrics through Factor Analysis
- Using the extracted factors, obtain *k* clusters through K-means Clustering

02 Model

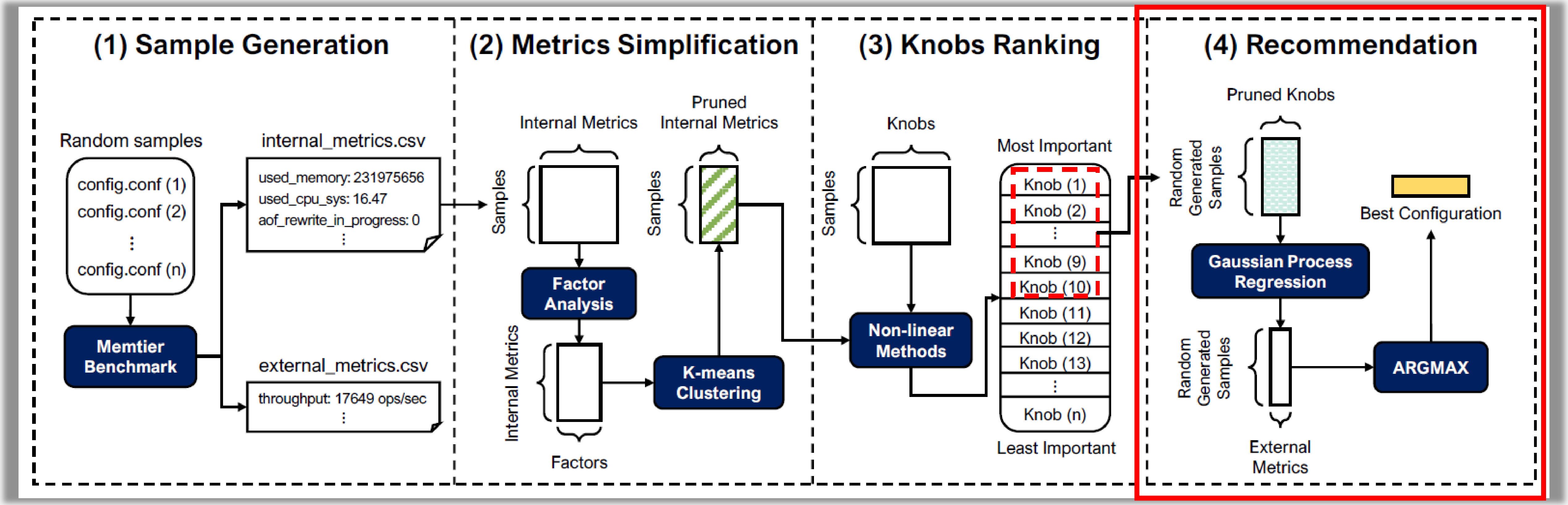
└ Knobs Ranking



- Select the influential knobs using the pruned internal metrics
- The degree of influence of knobs is measured and sorted - Top 10 knobs are utilized
- Non-linear machine learning methods : RandomForest, XGBoost

02 Model

└ Recommendation



- Recommend the optimal configuration for a specific workload through Gaussian Process Regression
- x is the pruned knobs obtained from Knobs Ranking, and y is learned with external metrics
- Generate a number of random configuration files and predict the performance through the learned GP model

목차

01 Introduction

02 Model

03 Experiment and Analysis

- Experimental Setup
- Results Analysis

04 Conclusion

03 Experiment and Analysis

└ Environmental Setup

- **Google Cloud Platform**
  - » Data sample generation
  - » Parameter tuning

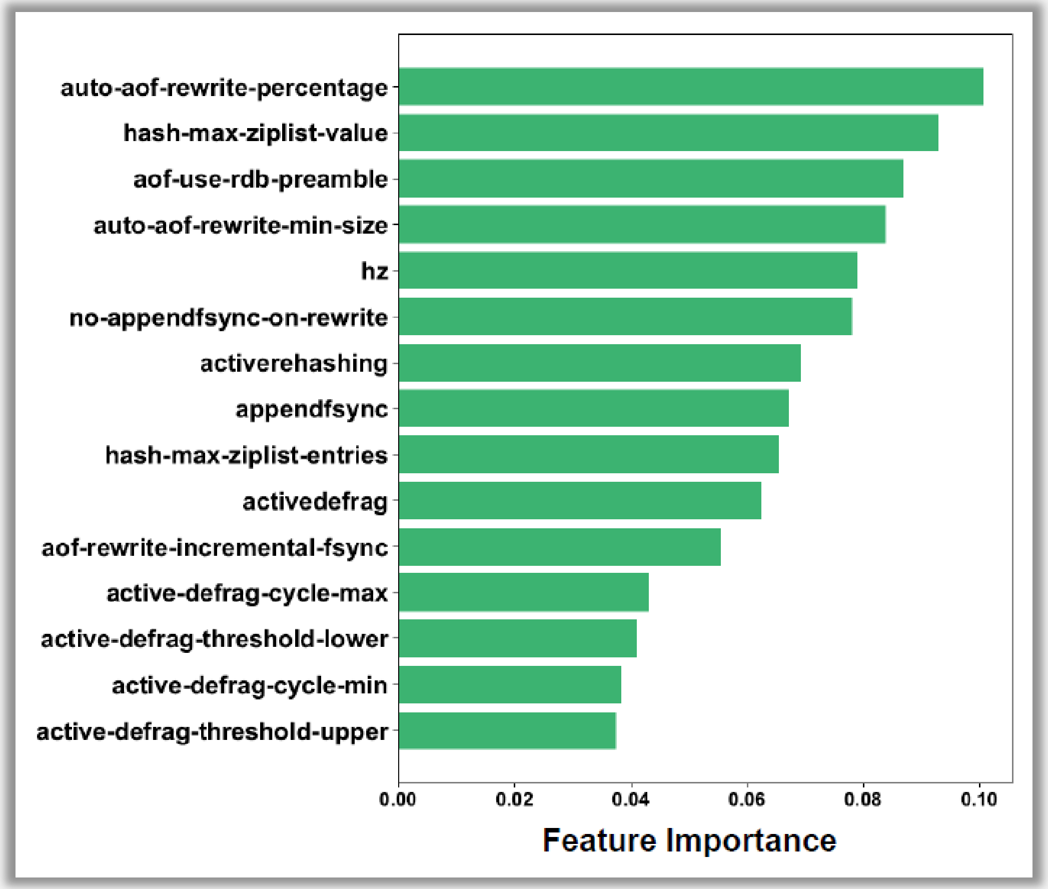
- **Memtier_Benchmark**
  - » Key size: 16 B
  - » Value size: 128 B
  - » # Requests: 1,000,000
  - » Workloads
    - – Write-Only
    - – Read-Write(1:1)

- **Redis Persistence Methods**
  - » RDB
  - » AOF

03 Experiment and Analysis

└ Results Analysis

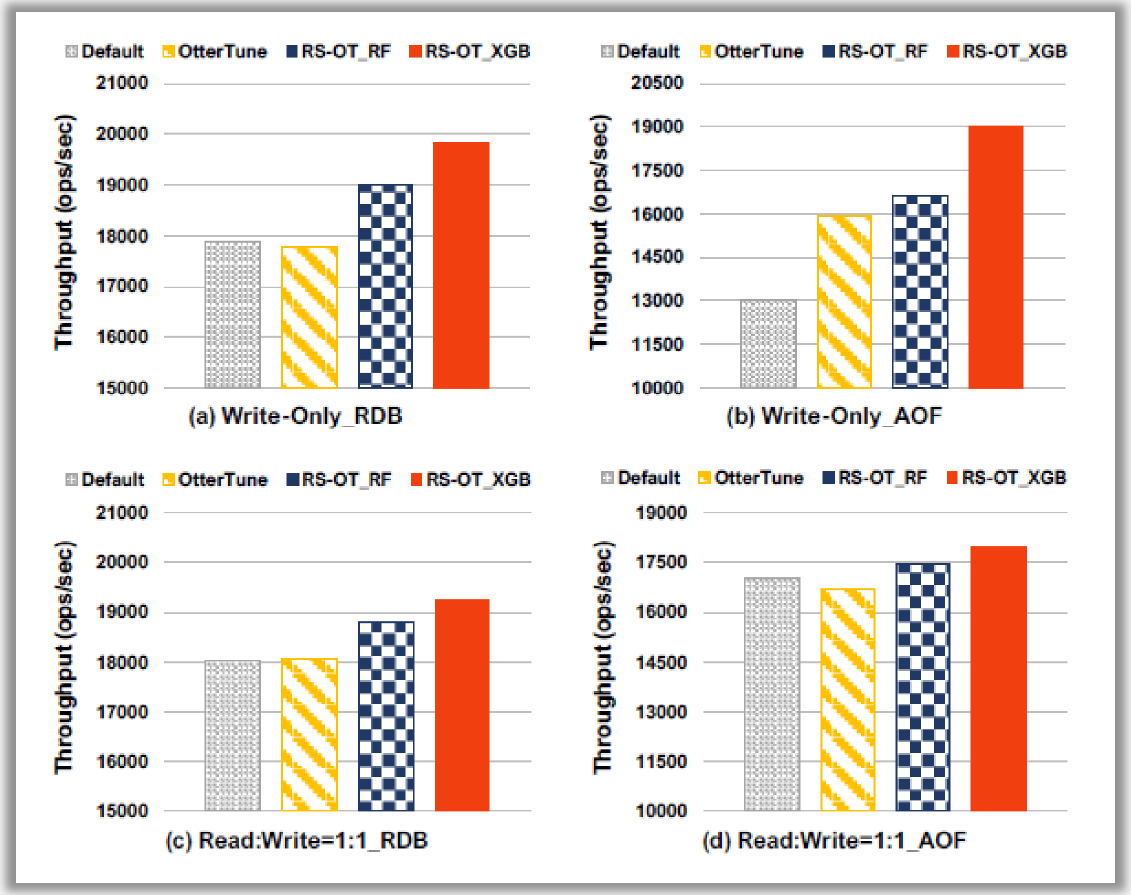
- **Parameter Contribution Assessment**
  - » Workload
    - – Write-Only
    - – Persistence method: AOF
    - – Knobs Ranking: XGBoost
  - » Sort the analyzed parameters in order of importance
  - » The top 10 parameters were optimized
  - » Reflect them in the configuration file



03 Experiment and Analysis

└ Results Analysis

- **Comparative Experiment**
  - » Default vs OtterTune vs RS-OT_RF vs RS-OT_XGB
  - » Throughput (ops/sec)
    - – Optimizing parameters using ML > Default
    - – Non-linear methods (RF, XGB) > Linear method (Lasso)
  - » Write-Only_AOF
    - – Excessive Disk I/O & Memory Usage
    - – About 45.9% improvement with XGBoost



(a) Write-Only_RDB

(b) Write-Only_AOF

(c) Read:Write=1:1_RDB

(d) Read:Write=1:1_AOF

목차

01 Introduction

02 Model

03 Experiment and Analysis

04 Conclusion

04 Conclusion

- **To alleviate performance degradation in Redis**
  - » Optimize the parameters using machine learning
  - » Utilize Non-linear methods rather than linear method
  - » Confirm that Redis performance degradation can be improved by parameter tuning
  - » Significant improvements in Redis performance when non-linear methods are used

Thank you

Q/A