ResTune: Resource Oriented Tuning Boosted by Meta-Learning for Cloud Databases

연세대학교 컴퓨터과학과 안성현

2024년 5월

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

정보통신기술진흥센터
Institute for Information & communications Technology Promotion

Contents

01 Introduction

Background

1. 자원 활용 최적화 필요성
2. SLA 만족
3. 응용프로그램 튜닝 시간 만족

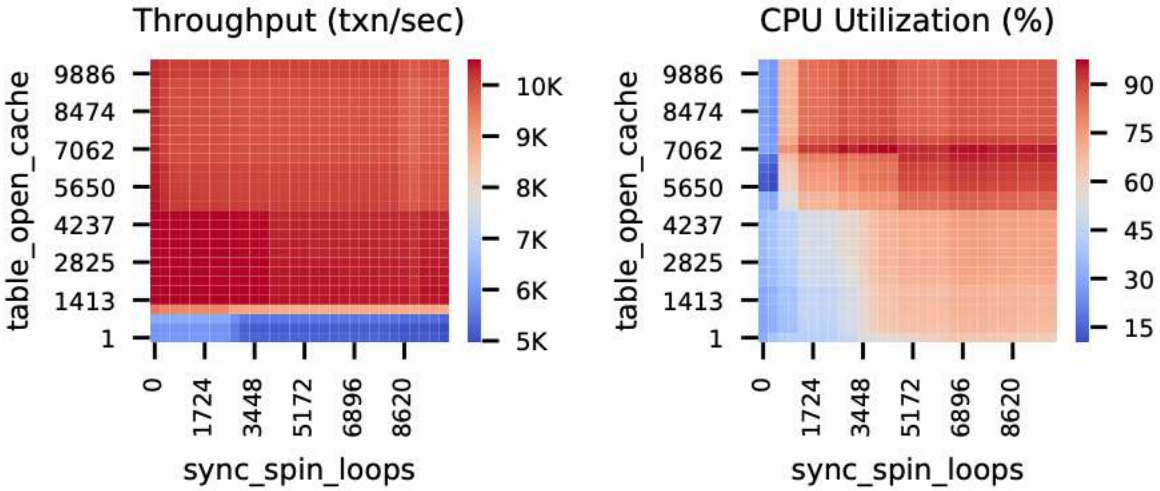


Figure 1: TPS and CPU Usage for Real Workload with 2 Knobs

Earlier research

방법	전략	설명	한계
BestConfig	탐색 기반	여러 휴리스틱을 사용하여 좋은 구성을 찾는 탐색 기반 방법입니다. 새로운 튜닝 요청이 있을 때마다 전체 탐색 과정을 다시 시작하며 과거 경험을 활용하지 않습니다.	과거 경험을 활용하지 않습니다.
OtterTune	베이지안 최적화 기반	튜닝을 블랙박스 최적화 문제로 모델링하는 BO 기반 방법을 사용합니다. Ottertune은 또한 작업량 매핑 전략을 사용하여 과거 경험을 고려합니다.	하드웨어 변경에 적응할 수 없으며, 이는 클라우드 환경에서 광범위한 데이터를 사용하는 것을 제한합니다.
iTuned		튜닝을 블랙박스 최적화 문제로 모델링하는 BO 기반 방법을 사용합니다.	
CDBTune	강화 학습 기반	내부 메트릭과 구성 노브 사이의 신경망을 학습하여 DBMS 성능을 튜닝합니다.	학습 과부하가 상대적으로 높아서 모델을 학습하는 데 수천 번의 반복이 필요합니다.
QTune			

What is ResTune?

- ResTune
  - Resource Oriented Tuning Boosted by Meta-Learning for Cloud Databases

- Goal
  - 자원 지향 구성 튜닝문제 해결
    - SLA 제약 조건이 있는 최적화 문제

$$\Theta = \Theta_1 \times \Theta_2 \times \ldots \times \Theta_m \text{ with } \Theta_i \in [0, 1].$$

$$\begin{aligned} &\arg\min_{\theta} f_{res}(\theta), \\ \text{s.t.} \quad &f_{tps}(\theta) \geq \lambda_{tps} \\ &f_{lat}(\theta) \leq \lambda_{lat} \end{aligned} \tag{1}$$

Let $f$res , $f$tps , $f$lat denote the resource utilization (e.g., $f$cpu , $f$memory or $f$io)

02 Design

Design Overview

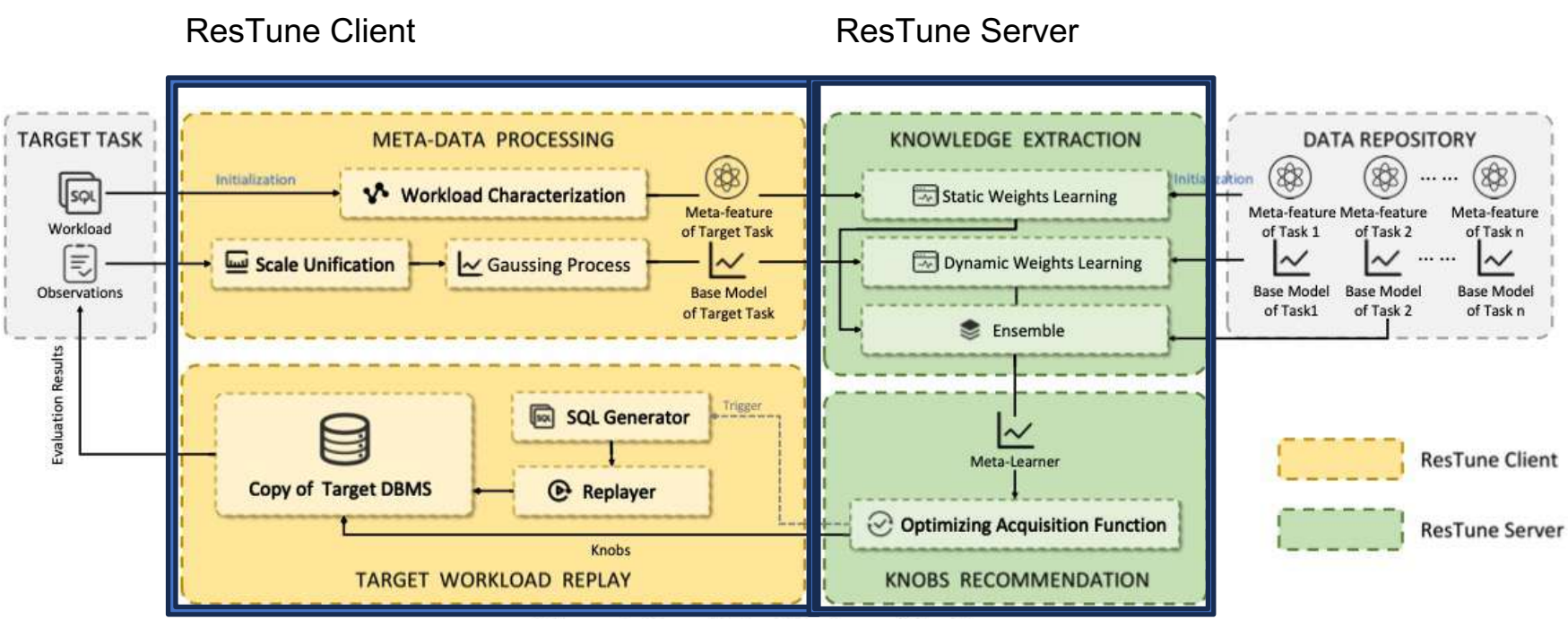


Figure 2: Overall Architecture of ResTune

Design Overview

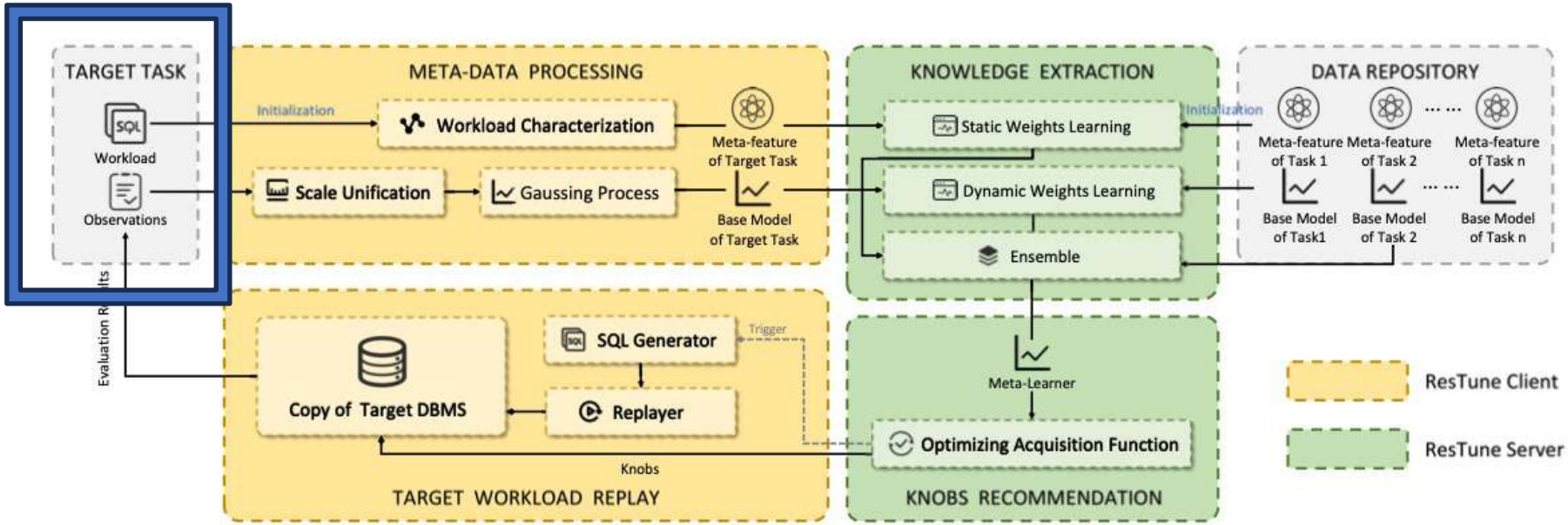


Figure 2: Overall Architecture of ResTune

Design Overview

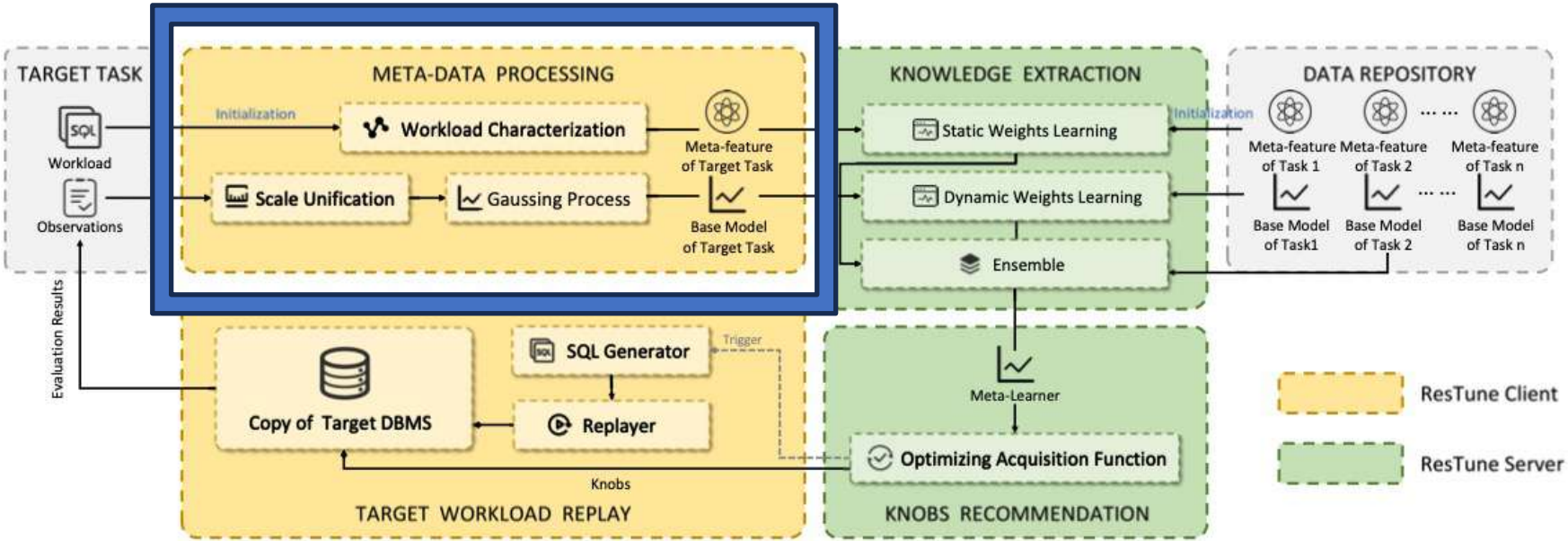


Figure 2: Overall Architecture of ResTune

Design Overview

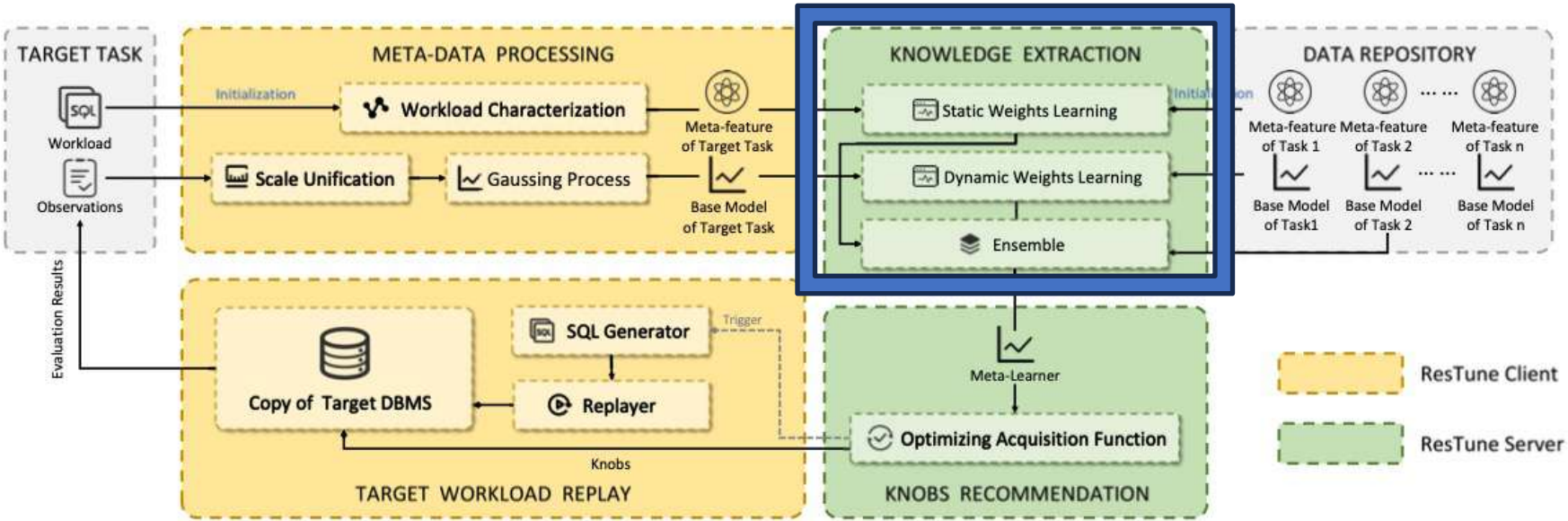


Figure 2: Overall Architecture of ResTune

Design Overview

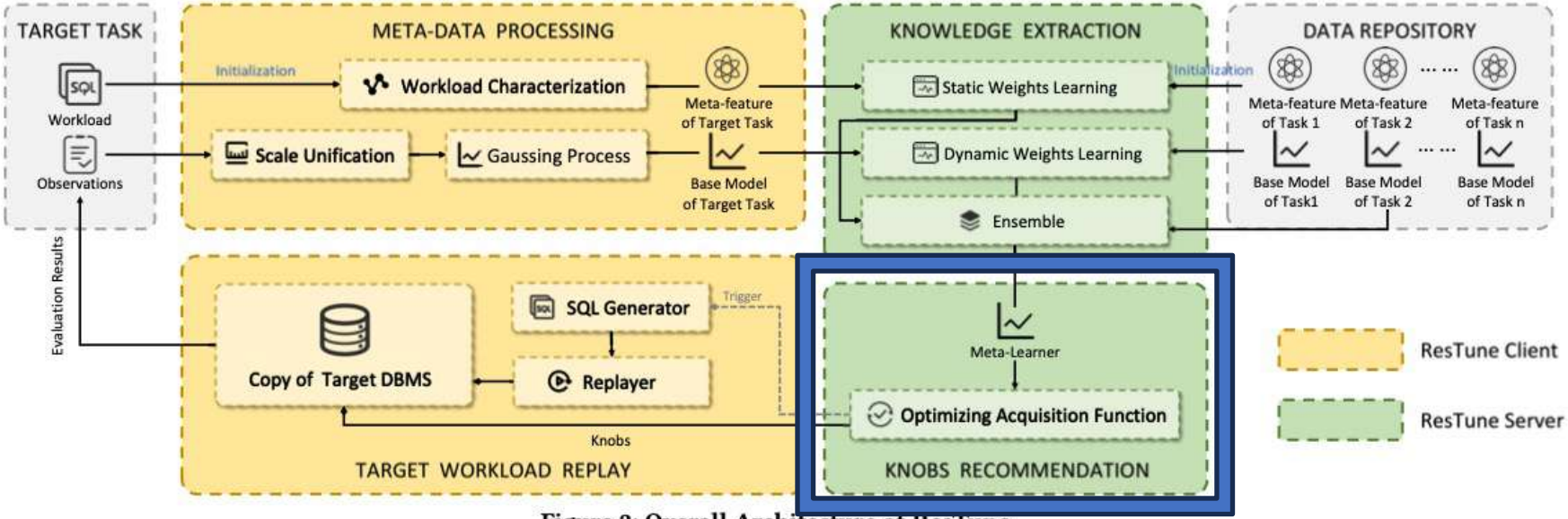


Figure 2: Overall Architecture of ResTune

Design Overview

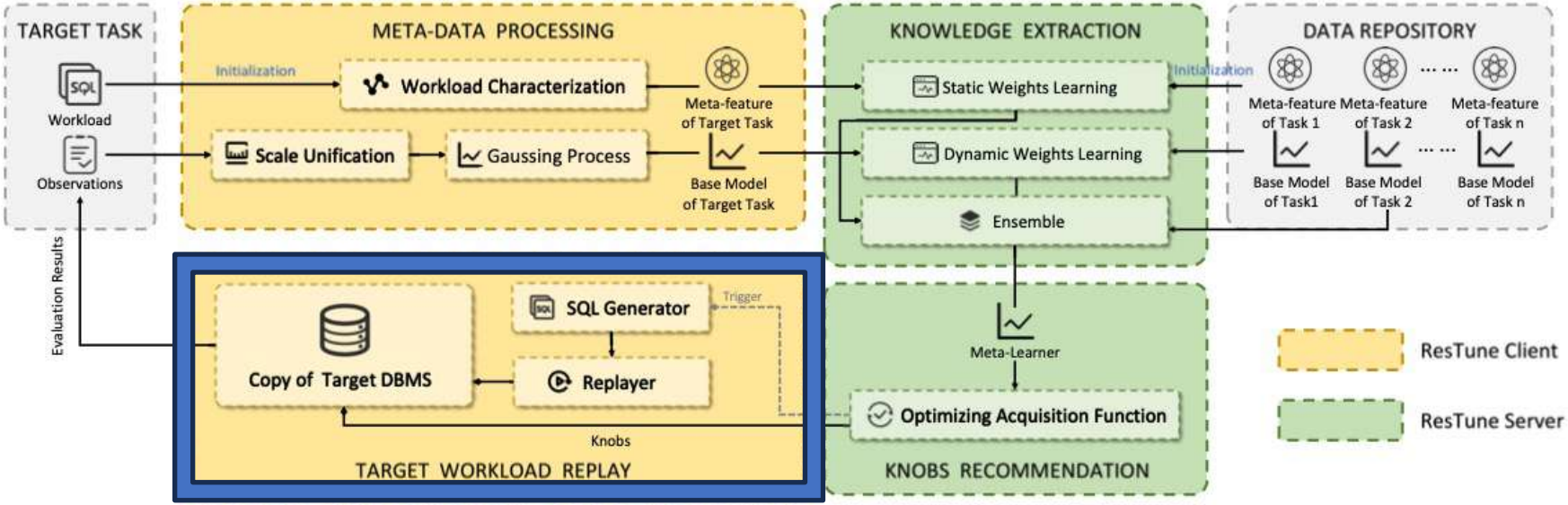
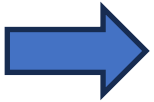


Figure 2: Overall Architecture of ResTune

SOLVING CONSTRAINED OPTIMIZATION

Modeling Constrained Functions using CBO

$$\begin{aligned} &\arg\min_{\theta} f_{res}(\theta), \\ \text{s.t.} \quad & f_{tps}(\theta) \geq \lambda_{tps} \\ & f_{lat}(\theta) \leq \lambda_{lat} \end{aligned} \qquad (1)$$



$$\tilde{f}_{tps}(\theta) \text{ and } \tilde{f}_{lat}(\theta) \qquad \tilde{f}_{res}(\theta)$$

$$H = \left\{ \left(\theta_i, f_{res}(\theta_i), f_{tps}(\theta_i), f_{lat}(\theta_i)\right) \right\}_{i=1}^{n}$$

- Feasible configurations
- Infeasible configurations

$$\mu_u(\theta) \quad \sigma_u^2(\theta),\ u \in \{res, tps, lat\}$$

SOLVING CONSTRAINED OPTIMIZATION

Guiding Search in Feasible Region

$$\alpha_{EI}(\theta) = \mathbb{E}\left[max(0, f_{res}(\theta_{best}) - f_{res}(\theta)\right] \quad (2)$$

$$I_C(\theta) = \Delta(\theta)\, max(0, f_{res}(\theta_{best}) - f_{res}(\theta)) \quad (3)$$

$$\tilde{I}_C(\theta) = \tilde{\Delta}(\theta) max(0, f_{res}(\theta_{best}) - \tilde{f}_{res}(\theta)) = \tilde{\Delta}(\theta)\tilde{I}(\theta) \quad (4)$$

SOLVING CONSTRAINED OPTIMIZATION

Guiding Search in Feasible Region

$$\tilde{I}_C(\theta) = \tilde{\Delta}(\theta) max(0, f_{res}(\theta_{best}) - \tilde{f}_{res}(\theta)) = \tilde{\Delta}(\theta)\tilde{I}(\theta) \quad (4)$$

$$\mathbb{E}[\tilde{\Delta}(\theta)] = Pr[\tilde{f}_{tps}(\theta) \geq \lambda_{tps}, \tilde{f}_{lat}(\theta) \leq \lambda_{lat}]$$

$$Pr[f_{tps}(\theta) \geq \lambda_{tps}, f_{lat}(\theta) \leq \lambda_{lat}]$$

$$Pr[f_{tps}(\theta) \geq \lambda_{tps}] \cdot Pr[f_{lat}(\theta) \leq \lambda_{lat}]$$

$$\begin{aligned} \alpha_{CEI}(\theta) &= \mathbb{E}\left[\tilde{I}_C(\theta)|\theta\right] = \mathbb{E}\left[\tilde{\Delta}(\theta)\tilde{I}(\theta)|\theta\right] = \mathbb{E}\left[\tilde{\Delta}(\theta)|\theta\right]\mathbb{E}\left[\tilde{I}(\theta)|\theta\right] \\ &= Pr[\tilde{f}_{tps}(\theta) \geq \lambda_{tps}] \cdot Pr[\tilde{f}_{lat}(\theta) \leq \lambda_{lat}] \cdot \alpha_{EI}(\theta) \end{aligned} \quad (5)$$

BOOSTING TUNING PROCESS

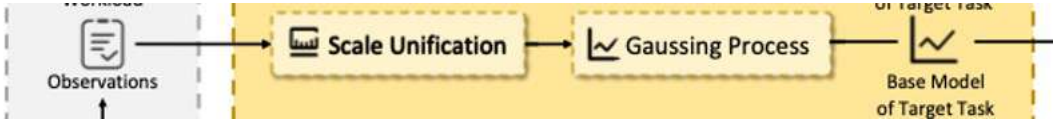
Solving resource-oriented tuning problem

correlation among them. Intuitively, the same workloads running on different hardware share information for tuning knobs. Even for different tasks, the relationship between hidden features can lead to knowledge sharing. As cloud providers, we can collect abundant

learner L_M

BOOSTING TUNING PROCESS

Scale Unification



$$\mathcal{H}_i = \{\theta_j^i, f_u(\theta_j^i, w_i)\}_{j=0}^{n_i}, i = 1, ..., T$$

$$L_M(\theta) \quad f_u(\theta) \leq \lambda_u$$

$$L_u^M(\theta) \leq \lambda_u'$$

$\lambda_u^{'}$ can be set as $L_M^u(\theta_d)$.

PROOF. If meta-learner predicts the relative performance value of θ to be a smaller value than that of θ_d .i.e. $L_M^u(\theta) \leq L_M^u(\theta_d)$, then it's predicted that $f_u(\theta, w_t) \leq f_u(\theta_d, w_t)$ .i.e $f_u(\theta, w_t) \leq \lambda_u$. If meta-learner predicts the the relative performance value of θ to be a larger value than that of θ_d .i.e. $L_M^u(\theta) \geq L_M^u(\theta_d)$, then it's predicted that $f_u(\theta, w_t) \geq f_u(\theta_d, w_t)$ .i.e $f_u(\theta, w_t) \geq \lambda_u$. There the re-scaled constraint $\lambda_u^{'}$ can be set as $L_M^u(\theta_d)$.

$$\lambda_u = f_u(\theta_d, w_{target}). \qquad \lambda_u' = L_u^M(\theta_d).$$

BOOSTING TUNING PROCESS

Workload Characterization

Feature Extraction

- calculate the TF-IDF feature vector for each query

Classification Model

- random forest model to classify each query

Workload Embedding Procedure

input workload. The averaged probability distribution represents the meta-feature for the input workload by characterizing the appearance frequencies of the queries.

BOOSTING TUNING PROCESS

Knowledge Extraction

Problem

1. 과거 실행 수가 많아지면 확장성 좋지 않음
2. 모든 인스턴스에 대해서 GP를 학습하는 것은 가정에 의존하는 것임

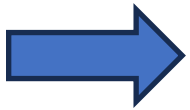
Base-learners

$l_i, i = 1, ..., T$

Meta-learner

$$\mu_M(\theta) = \frac{\sum_{i=1}^{T+1} g_i \mu_i(\theta)}{\sum_{i=1}^{T+1} g_i} \quad (6)$$

$$\sigma_M^2(\theta) = \sum_{i=1}^{T+1} v_i \sigma_i^2(\theta), \qquad v_i = \begin{cases} 1 & \text{i=T+1} \\ 0 & \text{otherwise} \end{cases} \quad (7)$$



$O(n^3)$

BOOSTING TUNING PROCESS

Base-Learner Evaluation - Learning from Meta-feature

6.4.1 Learning from meta-feature. At initialization, we measure the similarity for tuning tasks based on the meta-feature of workload. If a workload is more similar to the target workload, a larger weight is assigned to its base-learner. Therefore, the weight g_i of base-learner

Epanechnikov quadratic kernel

$$g_i = \gamma\left(\frac{\|m_i - m_{T+1}\|_2}{\rho}\right), \qquad \gamma = \begin{cases} \frac{3}{4}(1-t^2) & t \le 1 \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

BOOSTING TUNING PROCESS

Base-Learner Evaluation - Learning from Model Predictions

$$R_u(l_i) = \sum_{j=1}^{n_t} \sum_{k=1}^{n_t} \mathbb{1}\left(l_u^i(\theta_j) \leq l_u^i(\theta_k)\right) \oplus \mathbb{1}\left(f_u(\theta_j, w_{T+1}) \leq f_u(\theta_k, w_{T+1})\right)$$

BOOSTING TUNING PROCESS

Base-Learner Evaluation - Adaptive Weight Schema

- 초기화 단계에서는 메타특징 추출하고 이를 부트스트랩 최적화를 한다.
- 더 많은 관측치가 수집되면 모델 예측의 순위에 따라 가중치를 할당한다.

03 Implementation & Evaluation

Implementation

1.Setting.

- implement ResTune using BoTorch
- use version 5.7 of MySQL RDS

Table 1: Hardware Configurations for Database Instances

	A	B	C	D	E	F
CPU	48 cores	8 cores	4 cores	16 cores	32 cores	64 cores
RAM	12GB	12GB	8GB	32GB	64GB	128GB

2. Workload.

Table 2: Workloads

Name	SYSBENCH	TPC-C	Twitter	Hotel	Sales
Size(G)	10,30,100	13,100	29	14	10
#Thread	64	56	512	256	256
R/W Ratio	7:2	19:10	116:1	19:1	154:1
Request Rate(txn/s)	21K	2K	30K	/	/

Implementation

3. Data Repository.

Category	Details
메타 데이터	34개의 과거 튜닝 작업에서 수집된 워크로드 특징 및 관찰 이력
실험 설정	기본 설정: 모든 34개의 과거 기본 모델 사용
하드웨어 설정 변경	32개의 과거 기본 모델 사용 (목표 작업의 메타 데이터 제외)
워크로드 설정 변경	17개의 과거 기본 모델 사용 (인스턴스 A와 B 사이에서 전환)
과거 모델의 일반화 가능성 평가	다른 인스턴스에서의 튜닝 작업에 대해 17개의 과거 기본 모델 (인스턴스 A와 B 사이의 전환) 및 34개의 과거 기본 모델 사용

Evaluation - Efficiency Comparison

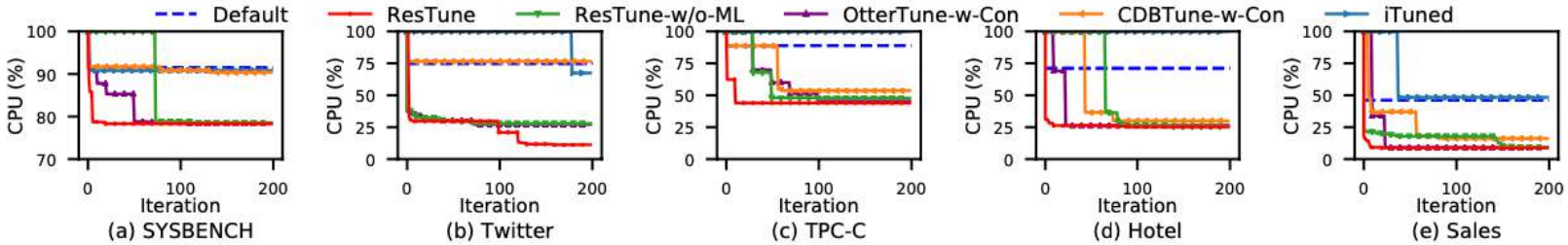


Figure 3: Efficiency Comparison

Evaluation - Evaluation on Generalization

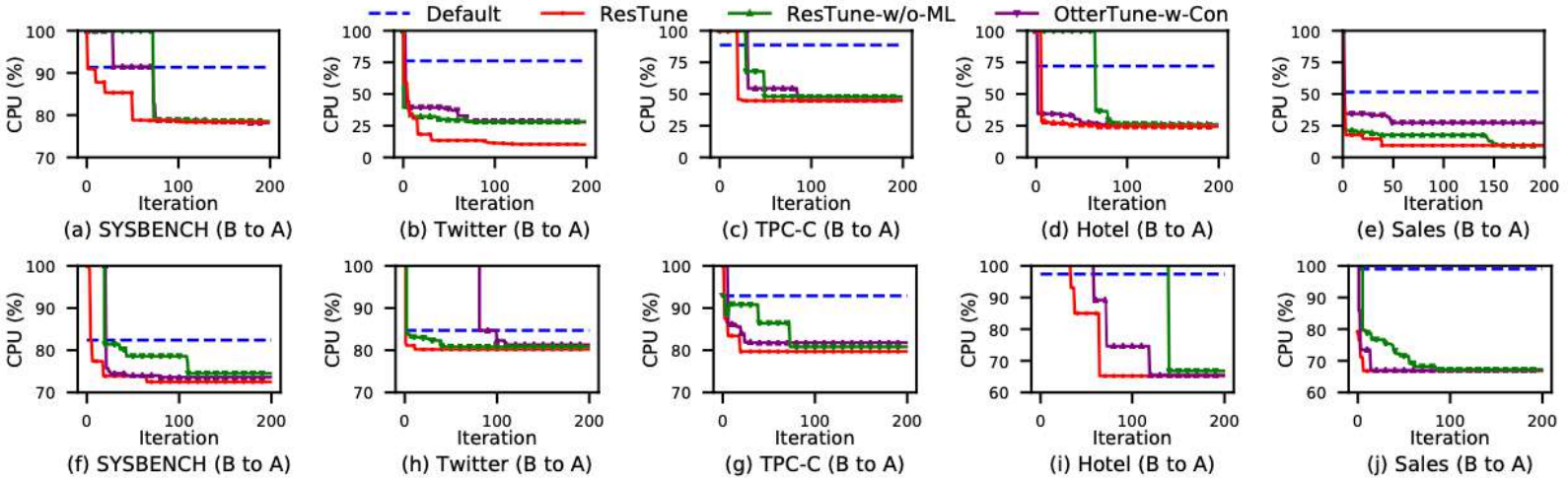
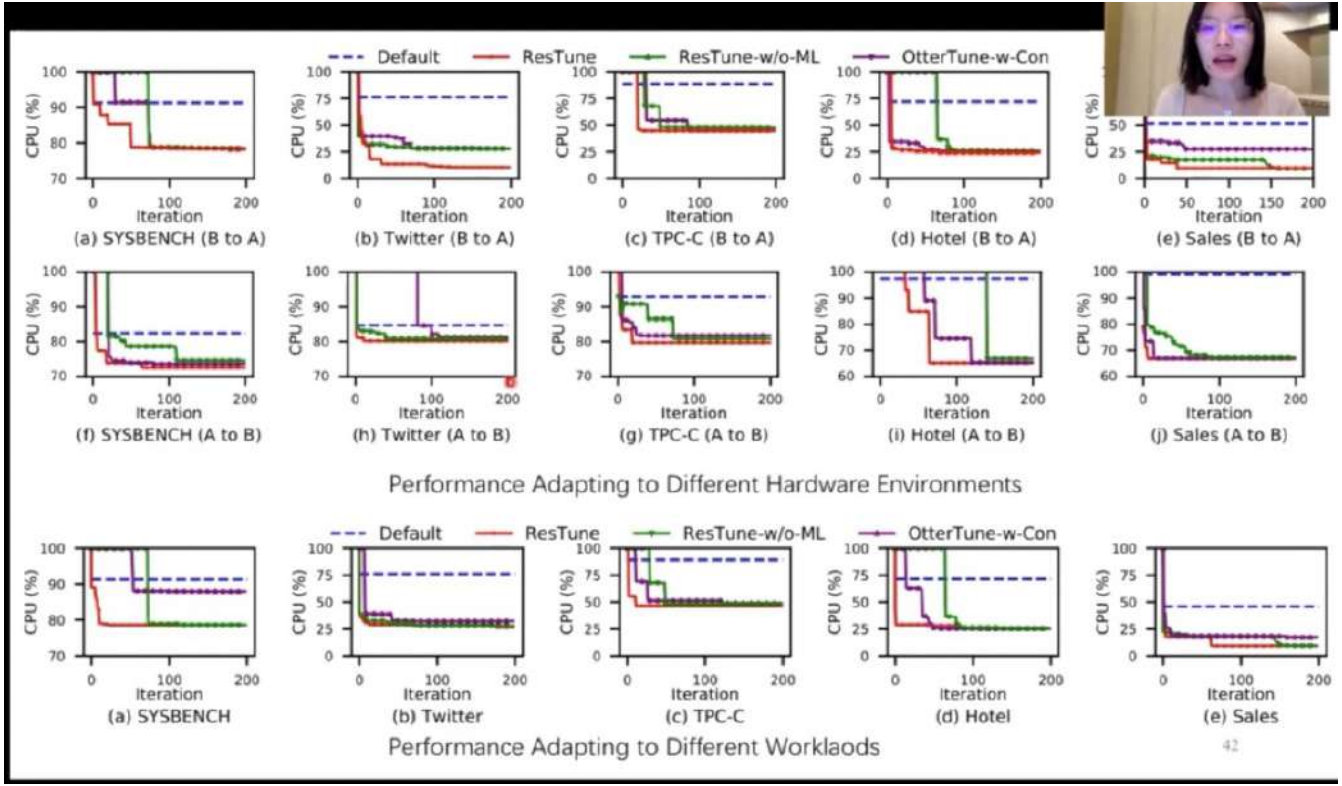


Figure 4: Performance Adapting to Different Hardware Environments

Evaluation - Evaluation on Generalization



https://dl.acm.org/doi/10.1145/3448016.3457291

Evaluation - Evaluation on Generalization

Table 4: Workload Adaptation on More Instances

Instance			C	D	E	F
SYSBENCH	Improvement	Restune	5.02%	8.13%	17.16%	20.38%
		Restune-w/o-ML	3.34%	7.58%	16.76%	19.96%
	Iteration	Restune	37	64	100	35
		Restune-w/o-ML	57	80	115	53
		Speed Up	35%	20%	14%	34%
TPC-C	Improvement	Restune	4.96%	19.22%	33.26%	47.60%
		Restune-w/o-ML	2.78%	18.28%	33.09%	42.62%
	Iteration	Restune	12	25	45	18
		Restune-w/o-ML	99	47	79	25
		Speed Up	87.87%	46.80%	43.03%	28%

Evaluation - Evaluation on Generalization

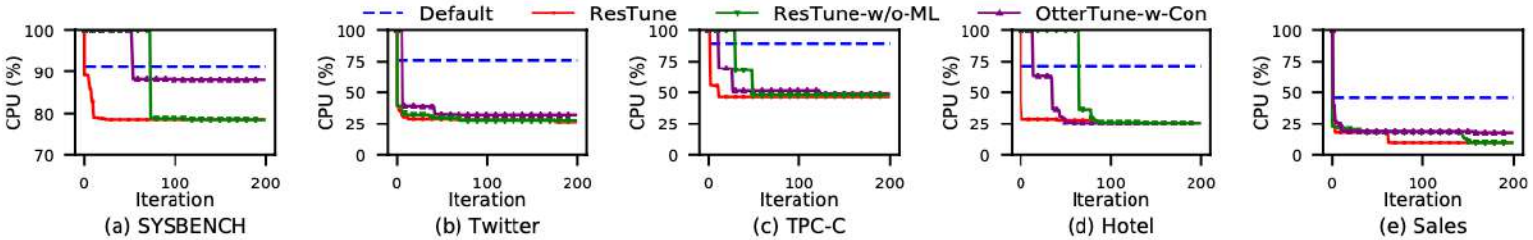
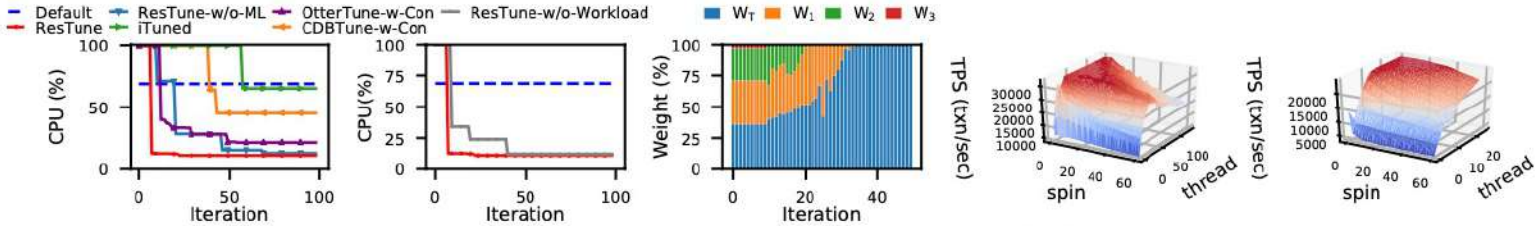


Figure 5: Performance Adapting to Different Workloads

Evaluation - Case Study



(a)Tuning Evaluation of Different Methods

(b)Ablation Study of Workload Charaterization

(c)ResTune's Weight Assignment

(d)W_T's Response Surface for TPS

(e)W_1's Response Surface for TPS

Figure 6: Case Study on Twitter Workload with 3 Tuning Knobs

Table 5: Statistics about Workload Variations

Workload	WT	W1	W2	W3	W4	W5
R/W Ratio	116:1	32:1	19:1	14:1	11:1	9:1
Distance to Wt	0	0.075	0.156	0.191	0.278	0.342
Static Weight	53.57%	46.00%	20.98%	4.80%	0%	0%
Ranking Loss	/	17.93%	22.71%	27.75%	34.04%	60.91%

Evaluation - Case Study

Table 6: Best Configurations Found by Different Methods

	thread_concurrency	spin_wait_delay	lru_scan_depth	CPU
Default	0	6	1024	75%
Grid Search	17	0	100	14.43%
ResTune	**13**	**0**	**356**	**11.22%**
ResTune-w/o-ML	14	1	100	12.97%
OtterTune-w-Con	30	2	2244	20.59%
CDBTune-w-Con	122	3	180	45.03%
iTuned	43	21	100	65.10%

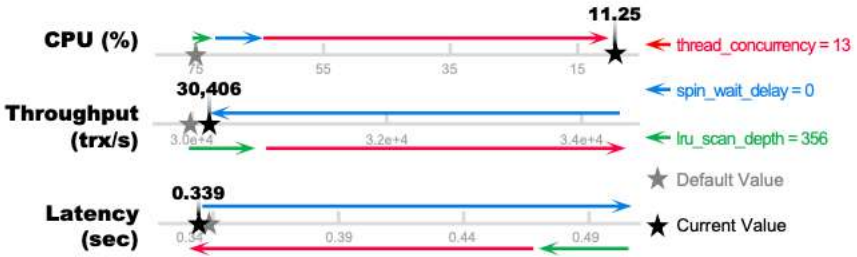


Figure 7: SHAP Path: Features Contributions from Default Knobs

Evaluation - Sensitivity Analysis

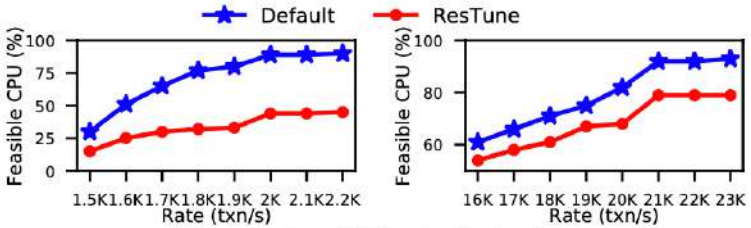


Figure 8: Sensitivity Analysis of Request Rate

Table 7: Sensitivity Analysis of Data Size

#Warehouses	Size(GB)	Hit Ratio	Default CPU	best CPU	Improvement
100	7.29	0.996	90.21	58.51	35.13%
200	16.26	0.995	87.78	43.95	49.93%
500	35.26	0.991	88.60	40.48	50.77%
800	56.59	0.984	78.59	34.53	58.52%
1000	117.06	0.946	46.00	36.62	44.80%

Evaluation - Tuning other types of Resources

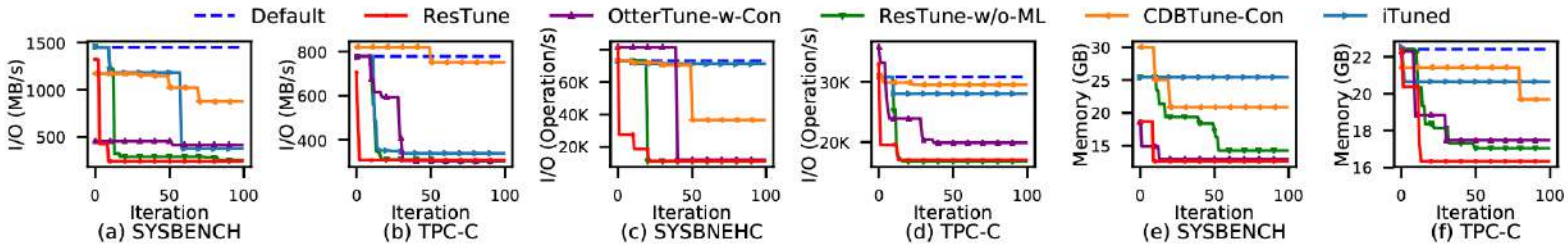


Figure 9: Tuning Other Types of Resources

Evaluation - TCO Analysis

Table 8: 1-year-TCO Reduction Optimizing CPU Usage

Workload		InstanceA	InstanceB	InstanceC	InstanceD	InstanceE	InstanceF
SYSBENCH	Original CPU	43 Cores	7 Cores	4 Cores	16 Cores	29 Cores	58 Cores
	Optimized CPU	21 Cores	6 Cores	4 Cores	15 Cores	24 Cores	46 Cores
	Avg TCO↓	$8,749	$398	$0	$398	$1,988	$4,772
TPCC	Original CPU	44 Cores	8 Cores	4 Cores	16 Cores	30 Cores	52 Cores
	Optimized CPU	38 Cores	7 Cores	4 Cores	13 Cores	20 Cores	27 Cores
	Avg TCO↓	$2,386	$398	$0	$1,193	$3,977	$9,942

Table 9: 1-year-TCO Reduction Optimizing Memory on Instance E

	Original MEM	Optimized MEM	TCO↓(AWS)	TCO↓(Azure)	TCO↓(Aliyun)
SYSBENCH	25.4GB	12.64GB	$983	$855	$2144
TPCC	22.5GB	16.34GB	$475	$412	$1035

04 Discussion

Discussion

- Pros
  - **Efficient Resource Optimization**
  - **Adaptability to Different Hardware and Workloads**
  - **Dynamic Weight Adjustment Mechanism**

- Cons
  - **클라우드 환경(AWS RDS)으로 인한 제한사항 ex)** innodb_flush_method
  - **ResTune은 현재 성능 예측 및 최적화를 위해 다중 출력 가우시안 프로세스 모델에 의존하고 있습니다. 효과적이기는 하지만, 다중 목표 베이지안 최적화(MOBO)를 탐구함으로써 이 접근 방식을 더욱 향상시킬 수 있습니다.**
  - **데이터의 양이 증가함에 따라 가우시안 프로세스 모델의 계산 복잡성이 병목 현상이 될 수 있습니다**

Appendix

TF - IDF (Term Frequency-Inverse Document Frequency)

주로 문서의 유사도를 구하는 작업에서 사용
문서 : d , 단어 : t, 문서의 총 개수 n

tf(d,t) : 특정 문서 d에서의 특정 단어 t의 등장 횟수
df(t) : 특정 단어 t가 등장한 문서의 수
idf(t) : df(t)에 반비례하는 수

$$TF(t,d) = \frac{number\ of\ times\ t\ appears\ in\ d}{total\ number\ of\ terms\ in\ d}$$

$$IDF(t) = log\frac{N}{1+df}$$

$$TF-IDF(t,d) = TF(t,d) * IDF(t)$$

https://wikidocs.net/31698

https://kinder-chen.medium.com/introduction-to-natural-language-processing-tf-idf-1507e907c19

Thank you.