

A STATISTICS-BASED PERFORMANCE TESTING METHODOLOGY FOR CLOUD APPLICATIONS

SEN HE, GLENNA MANNS, JOHN SAUNDERS,
WEI WANG, LORI POLLOCK, MARY LOU SOFFA

背景

- ▶ クラウドに展開しているアプリケーションのパフォーマンステストを正確に行うことは困難
 - テストは複数回行い、パフォーマンスの平均をアプリケーションのパフォーマンスとみなす
 - 同じテストを実行しても同じパフォーマンスが得られない
 - 分布が異なり、平均スループットも大きく異なることがある

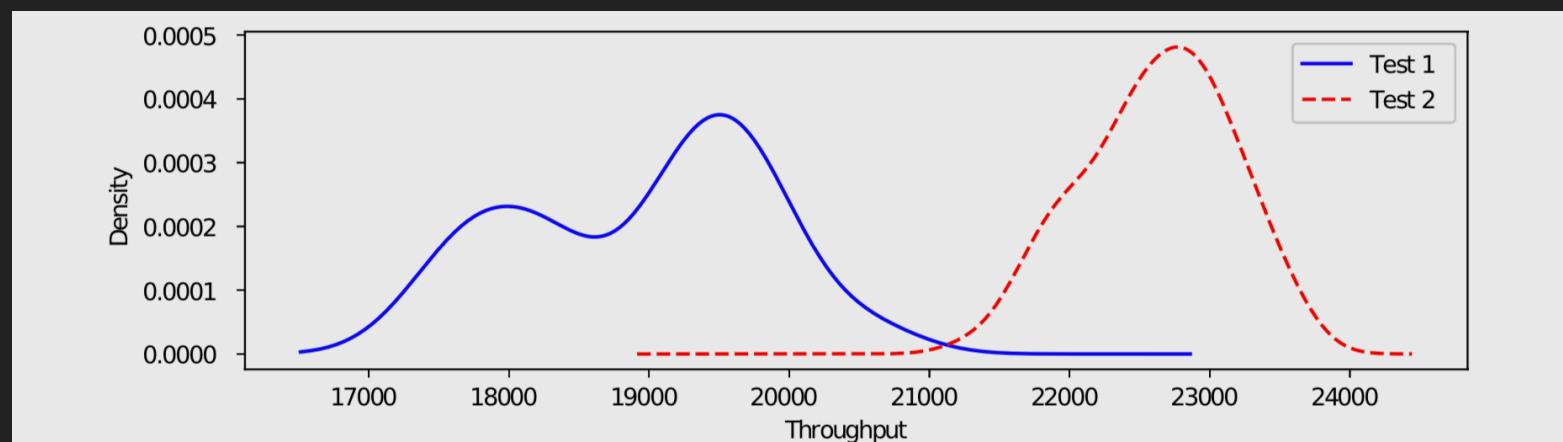


Figure 1: Performance distributions of YCSB from two tests.

Figure 1: Performance distributions of YCSB from two tests.



クラウドパフォーマンステストの課題（1）

- ▶ 仮想マシンによるハードウェアリソースの競合やプロセススケジューリングがランダム
 - パフォーマンスの良し悪しが判断しづらい
- ▶ クラウドサービスがブラックボックス
 - ユーザが実行環境を制御できないため、良いパフォーマンス悪いパフォーマンス両方のケースを網羅できない

クラウドパフォーマンステストの課題（2）

▶ パフォーマンステストで発生するクラウドの使用コスト

- 理論上、数ヶ月にわたりテストを実行すれば、ほぼ正確な結果が得られる
- ただし、現実的にはあり得ない

研究の目的と貢献

▶ 目的

- 新しいパフォーマンステスト手法であるPT4Cloudの提案

▶ 貢献

- 信頼性の高い停止条件を提供し、正確なパフォーマンス結果の取得
- 高い精度を維持しながらテスト実行回数を削減
- 提案手法を多角的に評価

PT4Cloud

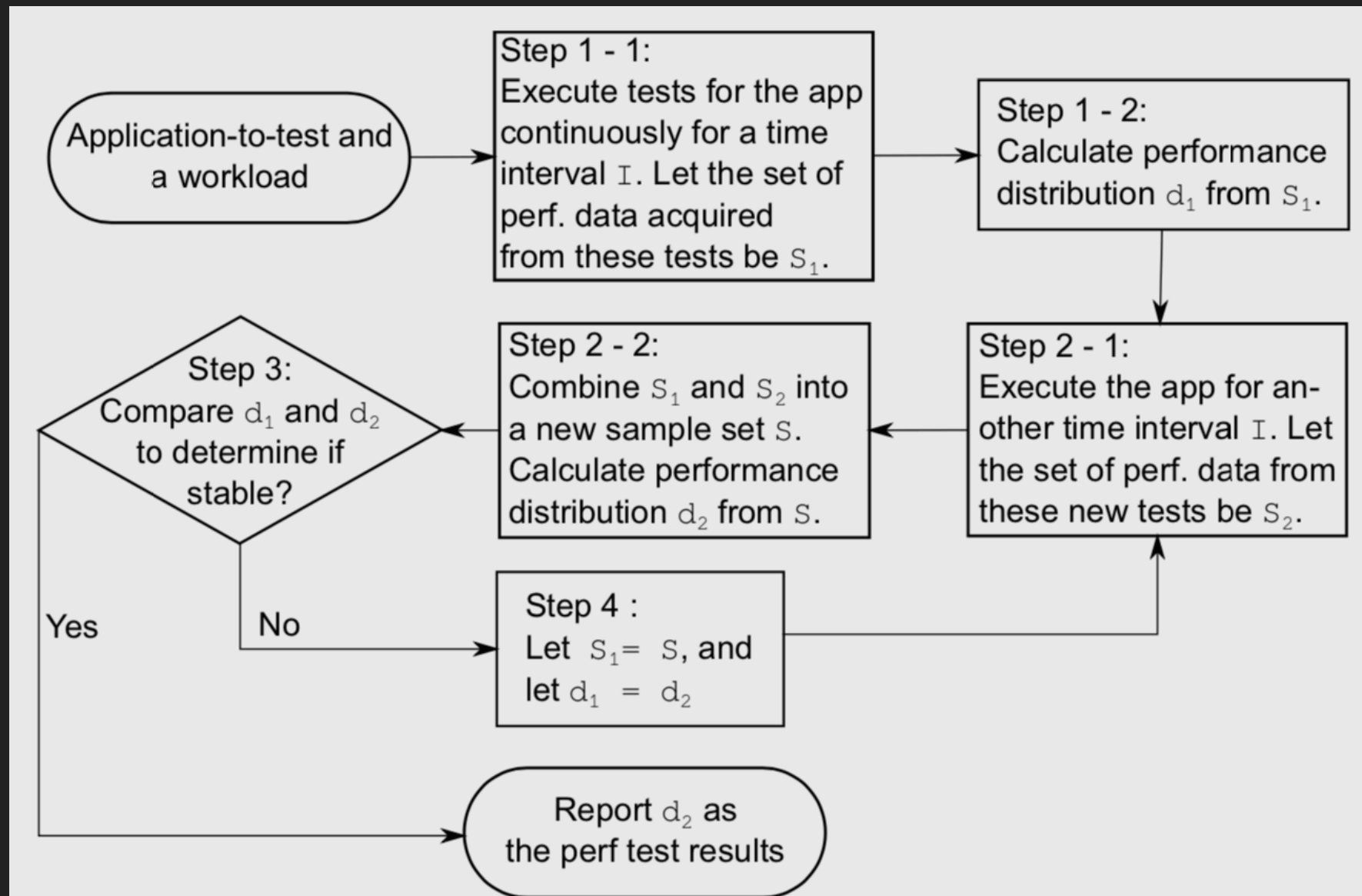
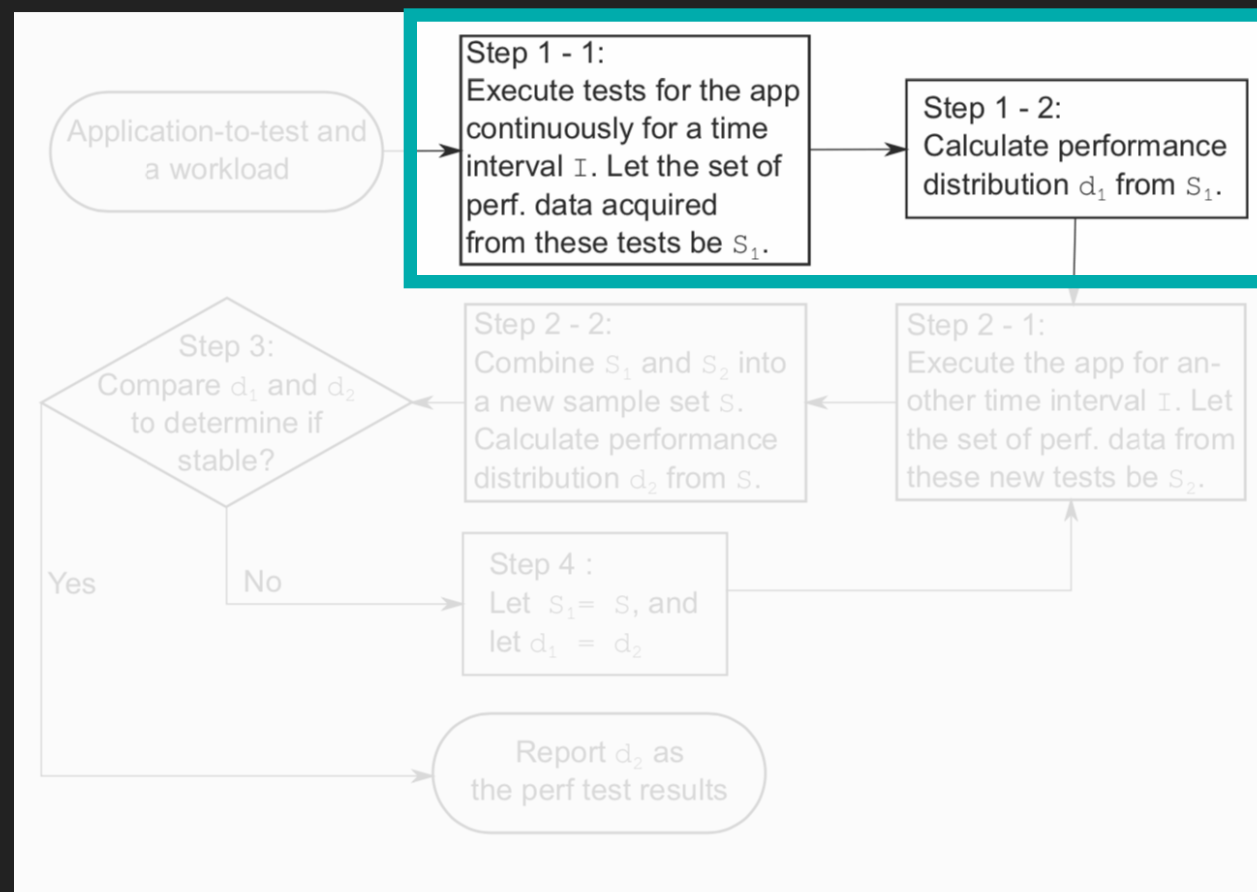


Figure 2: The workflow of *PT4Cloud*.

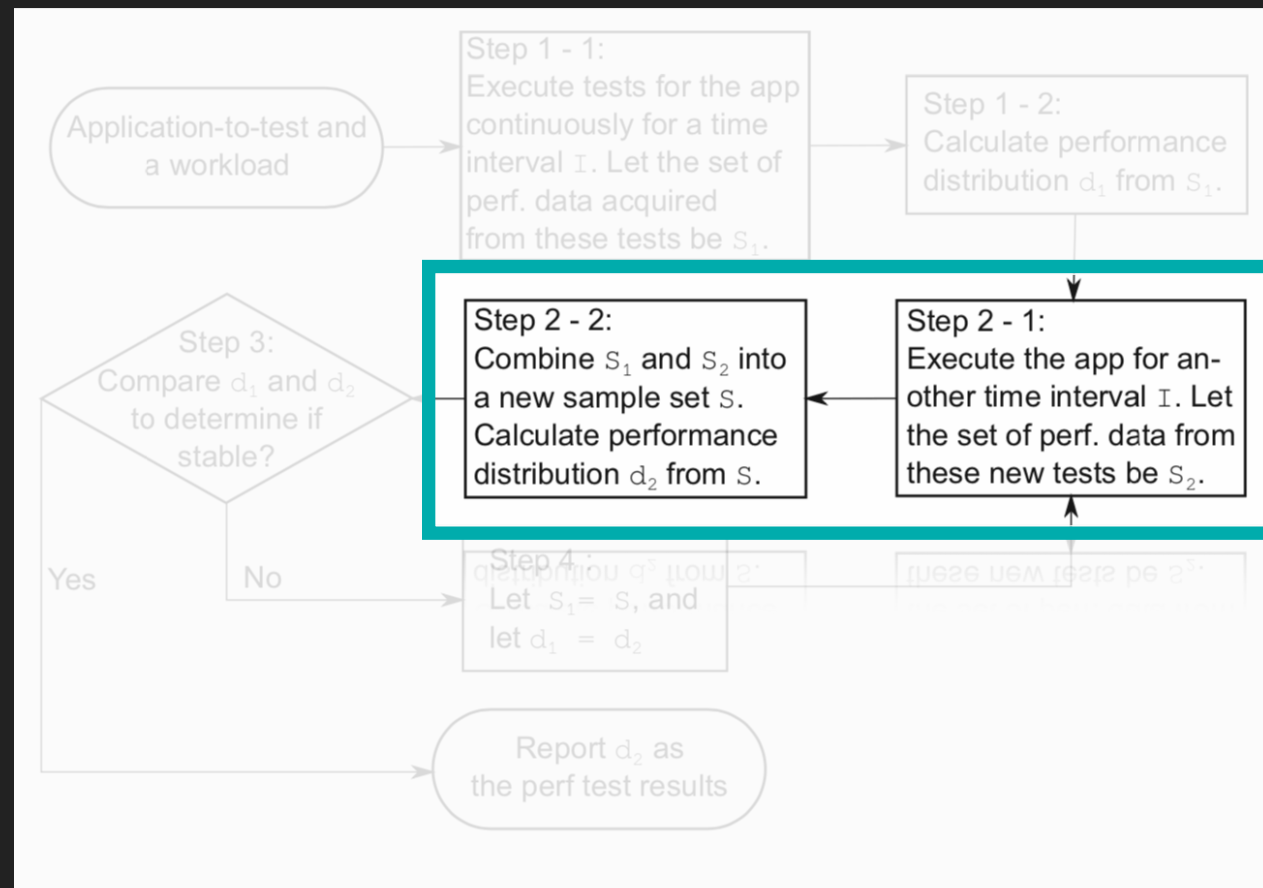
Figure 2: The workflow of *PT4Cloud*.

PT4Cloud ~ STEP 1 ~



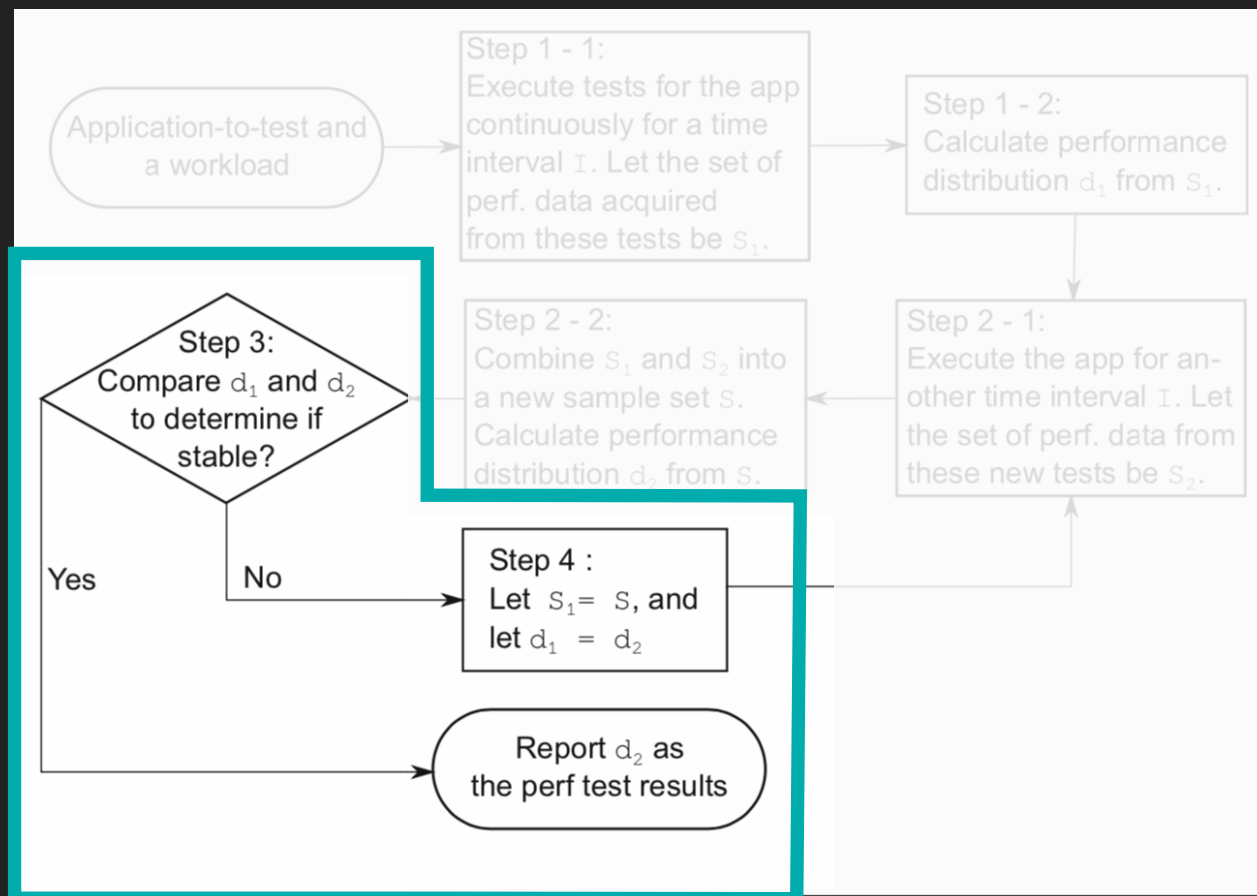
- 一定の間隔でテストを実行し、パフォーマンスデータ S_1 を取得
- 取得したデータを標本化 d_1
 - ▶ カーネル密度推定より、ノンパラメトリック確率密度関数を算出
 - ▶ ブートストラップ法より分布の信頼帯の生成

PT4Cloud ~ STEP 2 ~



- STEP 1と同様に，一定の間隔でテストを実行し，パフォーマンスデータ S_2 を取得
- S_1 と S_2 を合わせて標本化 d_2

PT4Cloud ~ STEP 3・4 ~



- ノンパラメトリック統計手法を用いて d_1 と d_2 を比較
- 分布の変化が少なかった場合
→ テスト終了
- 分布に変化が見られる場合
→ $d_1 \leftarrow d_2, S_1 \leftarrow S_2$ に入れ替え
STEP 2から繰り返し実行

PT4Cloudの停止条件

▶ カルバック・ライブラー情報量

(Kullback–Leibler divergence)

- 2つの確率分布の違いを定量的に測る尺度

$$D_{KL}(P||Q) = \int P(x) \log \frac{P(x)}{Q(x)} dx.$$

- 値が0 : 2つの確率分布は一致している
- 値が ∞ : 2つの確率分布は全く一致していない

PT4Cloudの停止条件

▶ カルバック・ライブラー情報量

(Kullback–Leibler divergence)

- 値が1のとき一致, 値が0のとき不一致となるように変換

$$L(P||Q) = 2^{-D_{KL}(P||Q)}$$

- 確率分布の一致確率を算出 (なぜこの式になるかは不明)

$$p = L(d_1||d_2) \times L(d_2||d_1)$$

ベンチマークと実験環境

▶ ベンチマーク構成

- ft, ep, IMAはCHM-1では
実行できなかった
→ メモリ不足

- 合計33のベンチマークで評価

Table 1: Benchmarks and their application domains.

Benchmark	Domain	Origin
<i>ft.C</i>	HPC	NPB
<i>ep.C</i>	HPC	NPB
<i>JPetStore (JPS)</i>	Web	J2EE
<i>YCSB</i>	DB	YCSB
<i>TPC-C</i>	DB	OLTPBench
<i>In-Memory Analytics (IMA)</i>	ML	CloudSuite

Table 2: VM configurations used for evaluation.

Config.	VM Cnt x VM Type	Core-Cnt	Mem Size
CHM-1	4 x Small (Sm)	1/VM	2GB/VM
CHM-2	2 x Medium (Md)	2/VM	4GB/VM
CHM-3	1 x Large (Lg)	4/VM	8GB/VM
AWS-1	4 x m5.large (M5Lg)	2/VM	8GB/VM
AWS-2	2 x m5.xlarge (M5XLg)	4/VM	16GB/VM
AWS-3	1 x m5.2xlarge (M52XLg)	8/VM	32GB/VM

VM2-3	1 x m2.xlarge (M2XLg)	8/VM	32GB/VM
VM2-5	5 x m2.xlarge (M2XLg)	4/VM	16GB/VM
VM2-1	1 x m2.xlarge (M2XLg)	8/VM	32GB/VM

実験の評価

▶ 目標基準値

- 確率分布の一致確率 ($\geq 90\%$)
- 信頼水準 ($\geq 99\%$)

▶ テストの実行間隔：1 週間

▶ 合計 6 週間のパフォーマンスデータより評価

分布が安定するのにかかった期間

ほとんどの組み合わせにおいて
2週間で分布は安定する
(分布の一致確率 $\geq 90\%$)

	“Test run Length” / “Accuracy (%)”					
	CHM-1	CHM-2	CHM-3	AWS-1	AWS-2	AWS-3
<i>ft.C</i>	N/A	2W/91	2W/99	2W/98	2W/99	2W/98
<i>ep.C</i>	N/A	2W/94	2W/92	2W/96	2W/99	2W/99
<i>JPS</i>	2W/98	2W/99	3W/96	2W/99	3W/96	2W/96
<i>YCSB</i>	3W/94	3W/93	2W/90	3W/93	2W/94	2W/93
<i>TPC-C</i>	2W/94	2W/92	2W/90	2W/97	2W/96	2W/95
<i>IMA</i>	N/A	2W/95	2W/95	2W/94	2W/96	2W/95

IMA	N/A	5M\82	5M\82	5M\84	5M\80	5M\82
TPC-C	5M\84	5M\85	5M\80	5M\81	5M\80	5M\82

分布が安定するのにかかった期間

ほとんどの組み合わせにおいて
2週間で分布は安定する
(分布の一致確率 $\geq 90\%$)

	“Test run Length” / “Accuracy (%)”					
	CHM-1	CHM-2	CHM-3	AWS-1	AWS-2	AWS-3
<i>ft.C</i>	N/A	2W/91	2W/99	2W/98	2W/99	2W/98
<i>ep.C</i>	N/A	2W/94	2W/92	2W/96	2W/99	2W/99
<i>JPS</i>	2W/98	2W/99	3W/96	2W/99	3W/96	2W/96
<i>YCSB</i>	3W/94	3W/93	2W/90	3W/93	2W/94	2W/93
<i>TPC-C</i>	2W/94	2W/92	2W/90	2W/97	2W/96	2W/95
<i>IMA</i>	N/A	2W/95	2W/95	2W/94	2W/96	2W/95

全てのデータを用いた分布は
実際の分布と
平均して約95%
の確率で一致

IMV	N/A	5M\82	5M\82	5M\84	5M\89	5M\82
TPC-C	5M\84	5M\85	5M\80	5M\81	5M\89	5M\82

分布が安定するのにかかった期間

3週間かかったものはWeb, DB系
ネットワーク・ディスク I/Oは
CPUやメモリよりパフォーマンス変動が大きい

	“Test run Length” / “Accuracy (%)”					
	CHM-1	CHM-2	CHM-3	AWS-1	AWS-2	AWS-3
<i>ft.C</i>	N/A	2W/91	2W/99	2W/98	2W/99	2W/98
<i>ep.C</i>	N/A	2W/94	2W/92	2W/96	2W/99	2W/99
<i>JPS</i>	2W/98	2W/99	3W/96	2W/99	3W/96	2W/96
<i>YCSB</i>	3W/94	3W/93	2W/90	3W/93	2W/94	2W/93
<i>TPC-C</i>	2W/94	2W/92	2W/90	2W/97	2W/96	2W/95
<i>IMA</i>	N/A	2W/95	2W/95	2W/94	2W/96	2W/95

IMA	N/A	5M\92	5M\92	5M\94	5M\96	5M\95
TPC-C	5M\94	5M\95	5M\90	5M\91	5M\96	5M\92

既存のテスト手法との比較

2つの既存手法に比べて
平均精度を大きく上回っている
提案手法PT4Cloudの平均精度：95.4%
(既存手法の平均精度：65.2%, 66.2%)

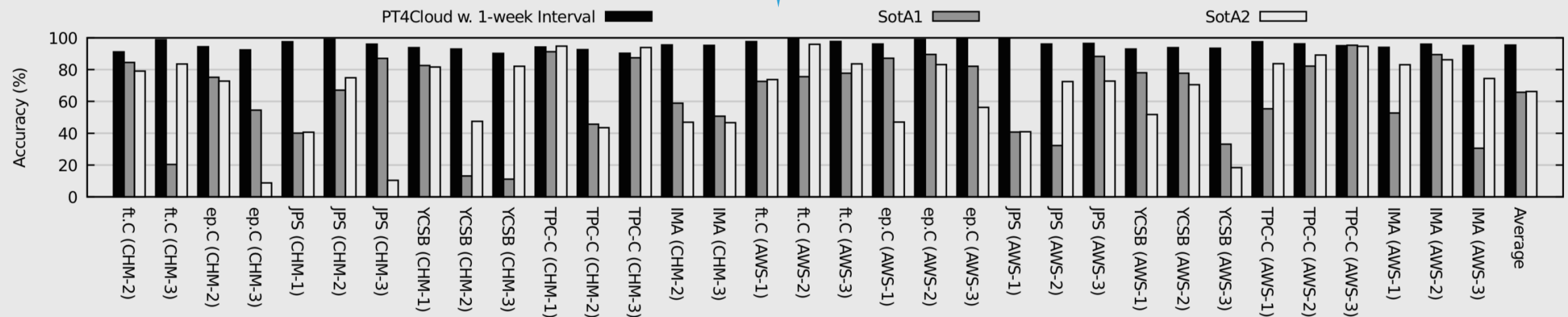


Figure 6: Accuracy of *PT4Cloud* when compared with two other state-of-the-art performance testing methodologies.

Figure 6: Accuracy of *PT4Cloud* when compared with two other state-of-the-art performance testing methodologies.

既存のテスト手法との比較

2つの既存手法に比べて

信頼性の高い分布を取得するために
要した期間は1ヶ月を下回る結果
(テスト実行回数の約62%を短縮)

かなり効率が良くなった

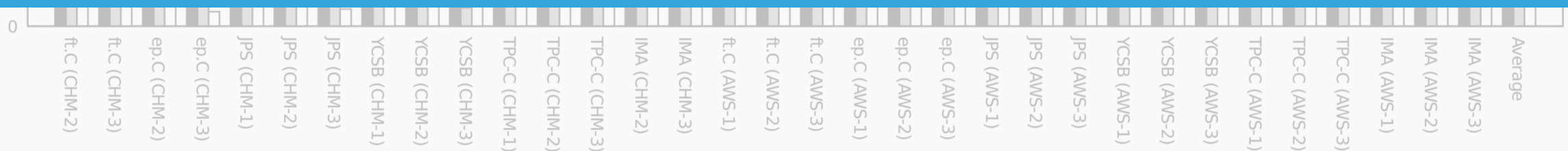


Figure 6: Accuracy of *PT4Cloud* when compared with two other state-of-the-art performance testing methodologies.

Figure 6: Accuracy of *PT4Cloud* when compared with two other state-of-the-art performance testing methodologies.

テスト効率を向上させるために

1,2週間のデータのみで分布がかなり近似している

||

テスト実行間隔を短くできる可能性がある

テスト効率を向上させるために

1,2週間のデータのみで分布がかなり近似している

||

テスト実行間隔を短くできる可能性がある



PT4Cloudの設計に
改善の余地あり？

PT4Cloud using short interval

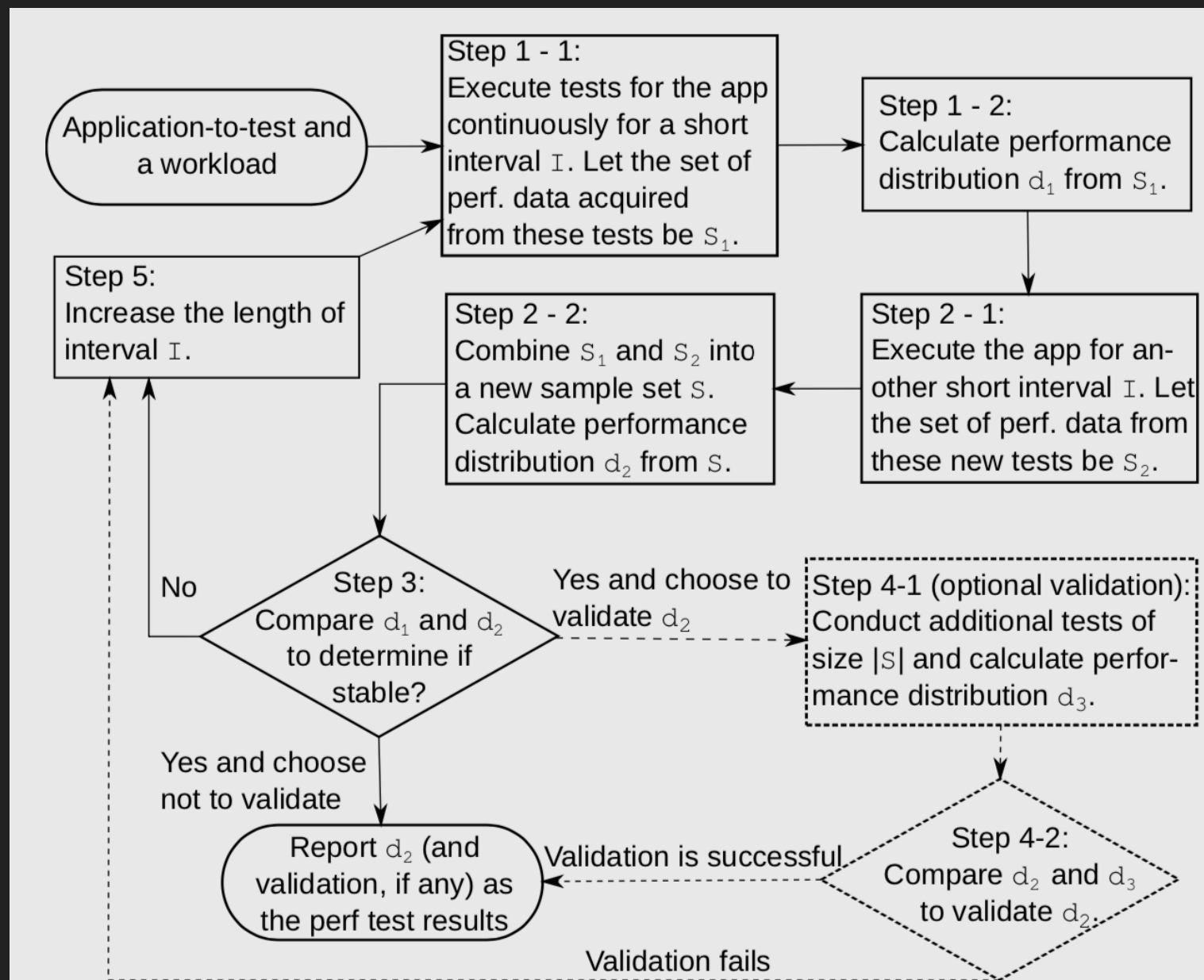


Figure 7: The workflow of *PT4Cloud* when using intervals shorter than one week.

PT4Cloud ~ additional STEP 5 ~

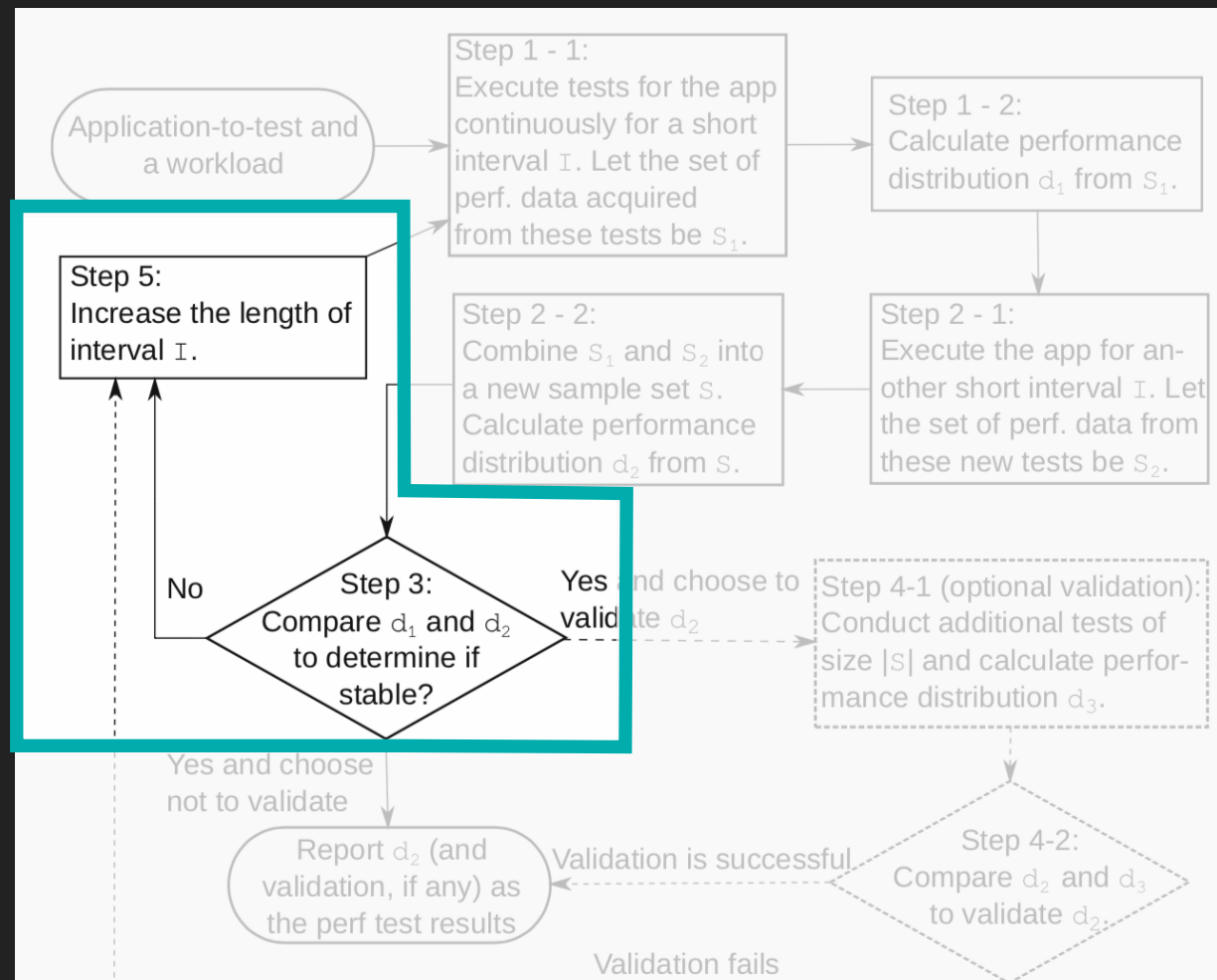


Figure 7: The workflow of *PT4Cloud* when using intervals shorter than one week.

- 分布に変化が見られる場合
→ テスト実行間隔を増やして
もう一度実行

shorter than one week.

Figure 7: The workflow of *PT4Cloud* when using intervals



PT4Cloud ~ optional STEP 4 ~

- 変化が少ない場合, 任意で検証を実施できる
- 間隔を増やして追加でデータ取得
- 追加データの分布と比較
- 成功すれば終了, 失敗すれば繰り返し実行

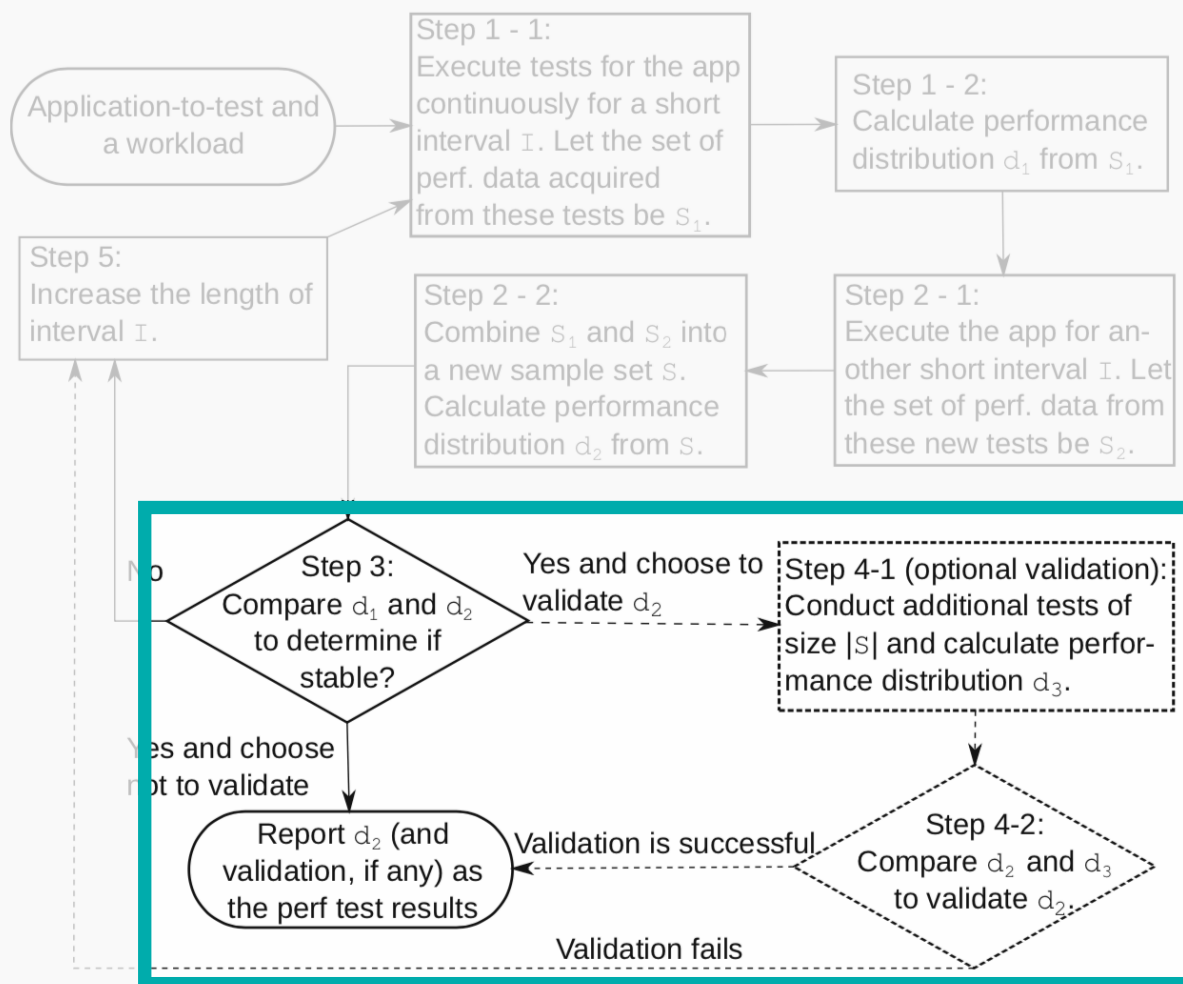


Figure 7: The workflow of *PT4Cloud* when using intervals shorter than one week.

shorter than one week.

Figure 7: The workflow of *PT4Cloud* when using intervals

評価実験

▶ PT4Cloud

- 最初の提案手法

▶ PT4Cloud (Reduced Interval)

- 提案手法のテスト実行間隔を縮小

▶ PT4Cloud (Reduced Interval + Reduced Sampling)

- テスト実行間隔の縮小に加え、データ取得の期間も縮小

評価実験（テスト削減効率）

最初の提案手法：62%削減
間隔のみ縮小：81.5%削減
両方とも縮小：90.1%削減

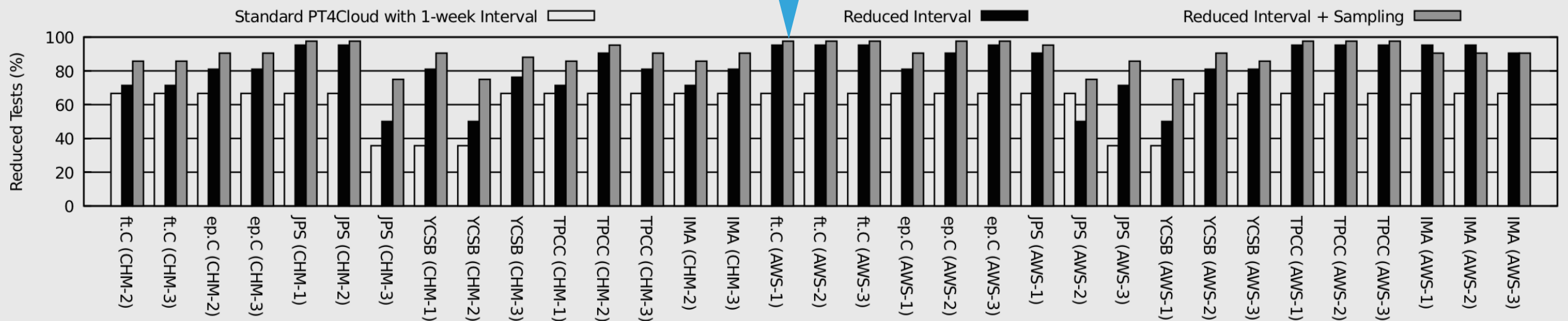


Figure 8: Number of tests reduced by *PT4Cloud* and our test reduction techniques, compared to the ground truth tests.

Figure 8: Number of tests reduced by *PT4Cloud* and our test reduction techniques, compared to the ground truth tests.

評価実験（分布の精度）

全てにおいて
90%以上の精度を記録

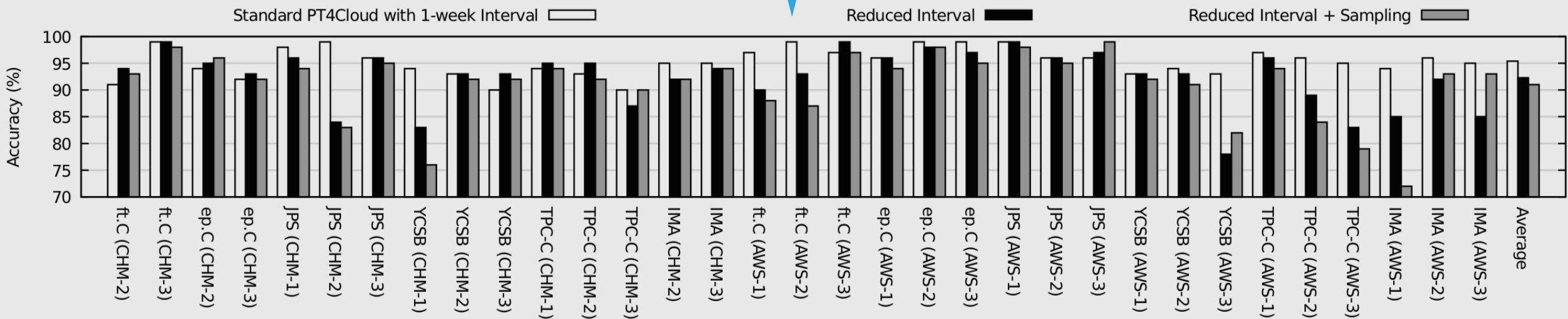


Figure 9: Accuracy of test reduction techniques.

Figure 9: Accuracy of test reduction techniques.

まとめ・所感

- ▶ パフォーマンステストの効率化を図った新たな手法の提案
- ▶ テストを効率的に行い、かつ、信頼性の高い結果が得られた
- ▶ やっていることが直感的に理解しやすい
- ▶ お金がないとこんな研究できないなと感じた

補足資料

PT4Cloudが取得した分布

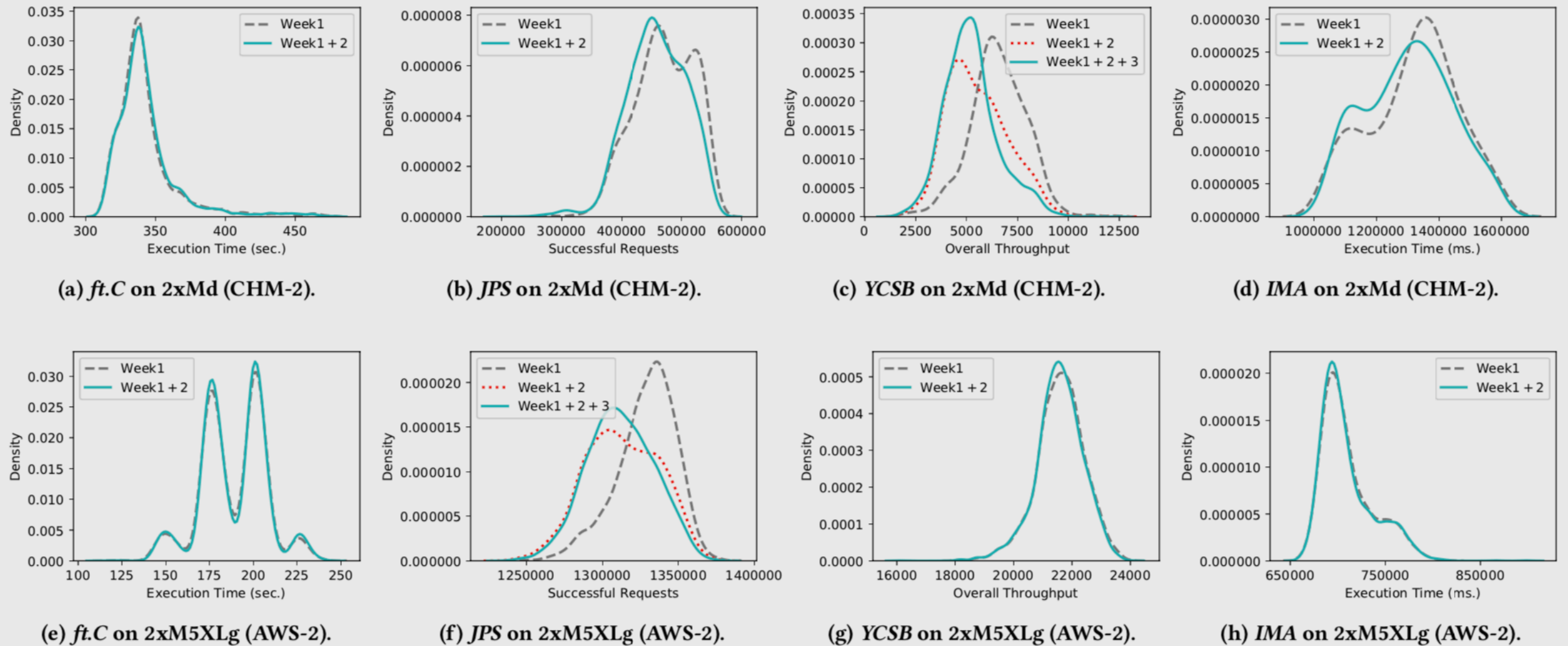


Figure 4: Performance Distributions acquired with *PT4Cloud* for four benchmarks on configurations of CHM-2 and AWS-2.

Figure 4: Performance Distributions acquired with *PT4Cloud* for four benchmarks on configurations of CHM-2 and AWS-2.

PT4Cloudと経験分布の一致

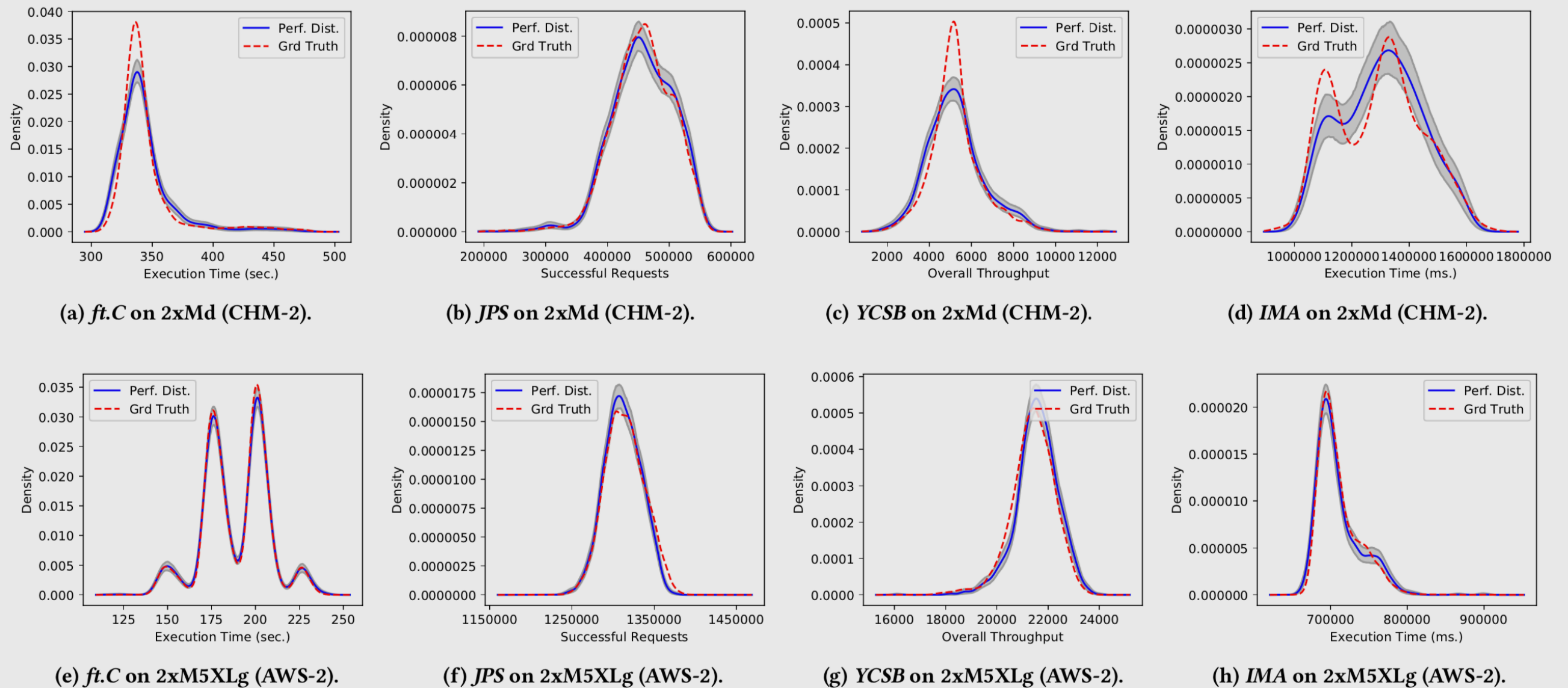


Figure 5: Performance distributions obtained with *PT4Cloud* and ground truth performance distributions for four benchmarks on two VM configurations on CHM and AWS. Shaded areas are confidence bands.

on two VM configurations on CHM and AWS. Shaded areas are confidence bands.

Figure 5: Performance distributions obtained with *PT4Cloud* and ground truth performance distributions for four benchmarks

既存手法

ソフトウェアエンジニアリング研究

- ▶ テスト実行から得られたパフォーマンスデータの反復性を検出することにより、パフォーマンステストを停止
 - パフォーマンステストの結果全体から短い期間のデータを選択し、選択したデータと他の期間と比較して、この期間のデータが繰り返されるかどうかを判断
- ▶ 上記のプロセスを1000回繰り返して、繰り返されるパフォーマンス結果の割合（反復性）を推定

既存手法

ソフトウェアエンジニアリング研究の問題点

- ▶ クラウドパフォーマンスの不確実性要因などの外部要因ではなく、アプリケーションの内部要因によるパフォーマンスの変化を検出するためだけに設計されているため

既存手法

コンピュータシステムの研究

- 特定の統計の信頼区間（CI）が希望の幅に狭まる場合、パフォーマンステストが停止