Towards Causal Interpretation on Datacenter-scale Stragglers

Jihye Choi, Apurbaa Bhattacharjee, Ashwin Keshav Tayade
{jihye, apurbaa, tayade}@cs.wisc.edu

1 Introduction

Stragglers are tasks in a job that take extremely long to complete and delay the completion of the job. These stragglers are a common problem in the environment of a datacenters. Stragglers are found to extend the completion time in 20% of jobs by 1.5 times in Google datacenter, and affect 47% and 29% in Facebook and Microsoft's datacenters [21]. Common strategies to mitigate stragglers are re-execution and speculative scheduling. Re-execution finds a machine stalling with tasks, then clone the tasks on different machine. Speculative scheduling predicts machines that under perform and avoid scheduling tasks on them. However, these approaches become inefficient especially when data is distributed unevenly across tasks, the tasks with large work cause the stalling. Moreover, they mitigate the symptoms but don't find the root cause that give rise to stragglers. Commonly used causal diagnosis methods even require the laborious involvement of human experts and fail to scale.

There have been efforts to bring Machine Learning (ML) ideas to efficiently identify the root causes of stragglers from traces of datacenter-scale jobs [21, 9]. In particular, Zheng et al. introduce a statistical ML framework, Hound [21] that provides interpretations on what system conditions contribute to large latency from the job traces in datacenters. Ensembling various statistical ML methods such as predictive modeling, dependence modeling and causal modeling, Hound outputs causal inference that is easy to interpret and robust to false interpretations. This is one of the pioneering work in this literature. However, there still exists a lot of scope for improvement to address the current limitations of Hound. To be specific, extensive evaluation on the reliability of causal inference is required. Since a statistical model cannot perform well on all datasets, Hound pursues the unbiased inference, which implies that ensembling explanations from multiple models can reduce the risk of false explanations even if the minority of models suffer from degraded performance. However, it is still unclear to what extent the models would remain reliable under various circumstances. If the majority of them become degenerated under certain scenarios, simply increase the size of ensembles by incorporating more models would not be a scalable approach to make the majority vote unbiased.

In this paper, focused on the causal modeling of the Hound framework, we advance the robust inference on straggler diagnosis with respect to the limitation as mentioned above. Hound's causal model constructs Rubin Causal Models (RCM) [16] with Inverse Probability Weighting (IPW) [12] as a causal effect estimator. Unfortunately, IPW performs poorly in the presence of dominant irrelevant covariates or highly skewed distribution of covariates [14]. These cases are prone to occur in real-world settings, especially in datacenters where job traces are aggregated from different machines with a wide range of diverse specifications. In need of identifying the appropriate use of causal effect estimators as a part of the Hound framework, we benchmark causal effect estimators including IPW and other advanced estimators such as Causal Forest [6] and Bayesian Network [7] under simulation environments.

Once we identify the right causal effect estimator from the extensive evaluation, we replace IPW of Hound with it and present results from the real-world datasets [13, 4]. Ousterhout et al. provides the dataset for job trace analysis that are generated by Spark driver while running various benchmarks on clusters of Amazon EC2 machines and provide causal diagnosis by human experts as well [13]. We investigate the reliability of our causal interpretation framework compared to the groundtruth expert diagnoses.

Our main contributions are the following:

- We benchmark causal effect estimators under simulated settings where 1) the majority of the covariates are irrelevant to the outcomes, and 2) the distribution of control units are highly skewed.
- We generate explanations on root causes of stragglers from real-world job traces.
- We evaluate the performance with our causal interpretation methods in comparison to human expert diagnoses.

2 Background

2.1 Problem Setting

In this section, we introduce notations we use throughout this paper, and formalize the notion of Average Treatment Effect (ATE) based on Rubin Causal Models (RCM).

Consider a dataframe $D = [X, Y, T]$ where $X = [X_1, X_2, ..., X_m]$, $X_i \in \mathbb{R}^n$ for $\forall i \in [m]$, $Y \in \mathbb{R}^n$ and $T \in \{0,1\}^n$ respectively denote the continuous covariates for all units, the outcome vector and the treatment indicator (1 for treated, 0 for control). Notation x_{ji} indicates the value for ith covariate of jth unit. X is a tuple of system conditions, such as average processor usage, queuing delay and garbage collection frequency), which represents profiles from jobs that exhibit long-tailed latency distributions. These job profiles are collected throughout the datacenter and across time comprise a dataset. Y represents latency (i.e., stragglers have large Y values). Each time, we set ith system condition X_i as treatment factor, and infer model that takes Y as the dependent variable and $X' = [X_1, X_2, ..., X_{i-1}, X_{i+1}, ..., X_n]$ as independent variables. If $x_{ji} > \theta$, the jth unit belongs to treated group, where θ is some threshold. It belongs to control group, otherwise.

Our goal is to estimate the causal effect of X_i on Y while eliminating bias induced by other covariates, and RCM is the right formal mathematical framework for this purpose. For instance, we want to explain how high and low processor usage affect the latency while controlling for all other system conditions such as memory usage, scheduling events, and so forth. RCM quantifies the effect τ, which is the difference in responses with and without treatment as follows: for the jth sample from X,

$$\tau = Y^j(1) - Y^j(0)$$

where $Y^j(0)$ and $Y^j(1)$ are potential outcomes corresponding to the outcome we would have observed if we had assigned control or treatment to the jth sample. The average treatment effect (ATE) is then defines as

$$\tau = \mathbb{E}[Y^j(1) - Y^j(0)]$$

This metric exhibits a methodological challenge that in observational data, we cannot obtain both $Y^j(1)$ and $Y^j(0)$ at the same time. Hence, to estimate τ dealing with this counterfactualness, RCM involves causal effect estimators such as Inverse Probability Weighting (IPW) [12] and Causal Forest [6].

2.2 Causal Effect Estimation

Inverse Probability Weighting with Adaboost estimates the propensity scores more accurately than regression as it deals with problems such as collinearity, over-fitting, and outliers in causal effect estimation.

We also explore other graphical (Bayesian Network) and non-graphical (Causal Forest) causal estimators. Causal forest is an advanced causal effect estimator which estimates the heterogeneous treatment effects. The algorithm proposed Athey and Imbens [6] is composed of causal trees that estimate the effect of the treatment at the leaves of the trees. In Causal Bayesian Networks (CBN) [7], the structure learning learns the structure of the graph, where a path from node X to node Z is defined as a sequence of linked nodes starting at X and ending at Z. Then we perform causal inference on the DAG, such that, X is the cause of Z, if there is path from X to Z.

3 Modified Hound Framework

3.1 Causal Modeling

Base learner outputs concise causal explanations for stragglers in the form of profiled metrics per job. Please note that throughout this project, we focus on improving the performance of CA learner. Throughout the project, our key focus is to generate better causal inference results, for which we modify the Hound pipeline by detaching the PR and DP base learners and also remove the ensemble learning which is not needed with a single base learner. We run the modified Hound framework on top of the Spark framework [19] by installing all the necessary package dependencies.

Figure 1 shows our modified Hound pipeline. AIPW base learner for causal inference is the baseline causal inference method. We compare advanced Causal Forest estimator with this baseline. For benchmarking and comparing different estimators, we use the same maximum depth level (30) for both the `DecisionTreeClassifier` in AIPW and Causal Forest implementations. We discuss in details how we tune this hyper-parameter in Section 4.1. The CA base learner outputs a causality profile for each of the jobs. The causality profiles from the CA base learner are fed into the meta learner which performs the topic modeling using Latent Dirichlet allocation (LDA). The meta learner finds prominent topics from the entire dataset, then identifies a mix of relevant topics for each job. This modified pipeline is the design we run our experiments on in this project.

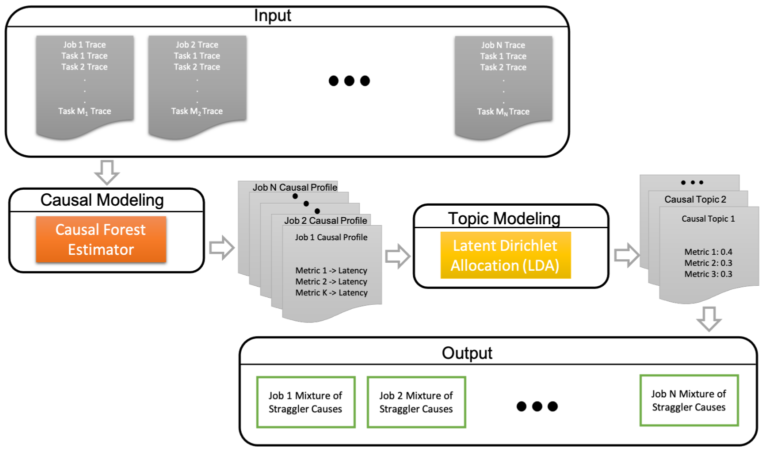


Figure 1: Modified Hound Framework. We use Causal Forest for Causal Modeling and LDA for Topic Modeling.

3.2 Topic Modeling

Given the profiled metrics from the previous stage, we use Latent Dirichlet Allocation (LDA) for topic modeling to extract semantics across the profiles that are easy to interpret. Metrics which occur in multiple profiles can be considered as prominent causes of stragglers in many jobs.

4 Evaluation

4.1 Simulations

We benchmark Athey-Imbens Causal Forest [6] on two simulated settings that commonly arise in the practical data-center scenarios. For detailed implementation, we adopt the design of generating simulation datasets from [11, 1]. For benchmarking the causal forest and causal Bayesian network estimators, we build upon the implementation of [3, 2].

Presence of Irrelevant Covariates Out of 15 covariates in total, there are 5 important covariates and 10 irrelevant covariates. For important covariates $1 \leq i \leq 5$ let $\alpha_i \sim N(10s, 1)$ with $s \sim \text{Uniform}\{1, 1\}, \beta_i \sim N(1.5, 0.15), x_i \sim \text{Bernoulli}(0.5)$. For unimportant covariates $5 < i \leq 15, x_i \sim \text{Bernoulli}(0.1)$ in the control group ($t_i = 0$) and $x_i \sim \text{Bernoulli}(0.9)$ in the treatment group ($t_i = 1$). This simulation generates 30000 samples (15000 control units and 15000 treatment units) for model fitting, and 10000 samples (5000 control units and 5000 treatment units) for evaluation.

Table 1 shows the results obtained on the simulated dataset having irrelevant covariates using three different causal estimators; AIPW, Causal Forest, and Bayesian Network. In Table 1 we have shown the difference in values of the ground truth ATE and the predicted ATE. The values show that AIPW and Causal Forest will be more robust estimators when the dataset has irrelevant covariates.

Table 1: Predicted ATE subtracted from ground-truth ATE for simulation data with irrelevant covariates

AIPW	Causal Forest	Bayesian Network
0.467	-0.398	3.15

Table 2: Predicted ATE subtracted from ground-truth ATE for simulation data with different imbalance ratios

	AIPW	Causal Forest	Bayesian Network
Ratio 1	45.62	1.04	76.98
Ratio 2	41.60	1.44	51.12
Ratio 3	35.10	0.08	35.06
Ratio 4	0.33	-0.27	5.72
Ratio 5	-8.57	-1.95	-8.53
Ratio 6	-33.92	0.82	-33.92

Imbalanced Data The data for this experiment has covariates with decreasing importance. We generate a fixed batch of 2000 treatment and 40000 control units. We sample from the controls to construct different imbalance ratios: 40000 in the most imbalanced case (Ratio 1), then 20000 (Ratio 2), 10000 (Ratio 3), 2000 (Ratio 4, balanced number of control and treatment units), 1000 (Ratio 5), and 100 (Ratio 6).

Table 2 shows the results obtained on the simulated dataset having different levels of imbalance using three different causal estimators; AIPW, Causal Forest, and Bayesian Network. We perform experimental trials

Table 3: Causal topics extracted across the Lenovo traces using AIPW and Causal Forest estimators

Topic	AIPW		CF	
	Topic Keywords	Weights	Topic Keywords	Weights
T0	CTN_MEM_USAGE(-) , CTN_NET_T_BYTES(-) , CTN_CPU_USAGE(-) , CTN_NET_R_BYTES(-)	[0.32, 0.24, 0.24, 0.21]	CTN_NET_T_BYTES(-) , CTN_NET_R_BYTES(-) , MACHINE_MEM_USAGE(-) , MACHINE_CPU_USAGE(-)	[0.39, 0.3, 0.16, 0.15]
T1	CTN_CPU_USAGE(+)	[1.]	MACHINE_NET_T_BYTES(+) , MACHINE_CPU_USAGE(+) , MACHINE_NET_R_BYTES(+) , CTN_NET_R_BYTES(+) , CTN_NET_T_BYTES(+) , CTN_CPU_USAGE(+)	[0.24, 0.2, 0.19, 0.14, 0.12, 0.11]
T2	MACHINE_CPU_USAGE(+) , MACHINE_NET_T_BYTES(+) , MACHINE_NET_R_BYTES(+)	[0.47, 0.27, 0.26]	CTN_MEM_USAGE(-)	[1.]
T3	MACHINE_NET_R_BYTES(-) , MACHINE_NET_T_BYTES(-) , MACHINE_CPU_USAGE(-)	[0.37, 0.32, 0.31]	MACHINE_NET_R_BYTES(-) , MACHINE_NET_T_BYTES(-)	[0.58, 0.42]
T4	CTN_NET_R_BYTES(+) , CTN_NET_T_BYTES(+)	[0.5, 0.5]	MACHINE_CPU_USAGE(+)	[1.]
T5	CTN_MEM_USAGE(+)	[1.]	CTN_MEM_USAGE(+) , MACHINE_MEM_USAGE(-)	[0.64, 0.36]
T6	MACHINE_MEM_USAGE(-)	[1.]	MACHINE_CPU_USAGE(+) , MACHINE_NET_R_BYTES(+) , MACHINE_NET_T_BYTES(+)	[0.4, 0.31, 0.29]
T7	MACHINE_MEM_USAGE(+)	[1.]	MACHINE_MEM_USAGE(+)	[1.]

to tune the maximum depth hyper parameter of the `DecisionTreeClassifier` in the AIPW estimator. We try different depth levels from 1 to 30 for each of imbalance ratios, and report the best result. In Table 2 the depth level used for Ratio 1, 2, and 3 is 30, for Ratio 4 is 2, for Ratio 5 is 20, for Ratio 6 is 10. We can observe here that for the balanced dataset (Ratio 4), APIW gives optimal results with very small depth level of the decision tree classifier. In Causal Forest estimator, we set the maximum depth level value as 30 which gives us close to optimal results. From Table 2 we observe that for the different levels of imbalance in the dataset, Causal Forest performs better than AIPW and Bayesian Network.

4.2 Real-world Datasets

In this section, we present experimental results from two real-world datasets; 1) a trace from Lenovo's containers and 2) a trace from a datacenter running Spark workloads. Table 4 shows the features used for each of datasets. We use AIPW and Causal Forest as causal effect estimators in Causal Modeling, and qualitatively evaluate their performance in comparison to expert diagnosis in Section 4.2.3.

4.2.1 Lenovo trace

Lenovo dataset has 140 jobs and 219,142 tasks. Table 3 summarizes the topics extracted across the Lenovo dataset. Table 5 shows an example of final causal interpretations when AIPW estimator is used.

4.2.2 Spark trace

Ousterhout et al. provides JSON logs from running the Big Data Benchmark and TPC-DS Benchmark on clusters of 5-60 machines [13]. These are collected by the Spark driver (version 1.2.1) while running various benchmarks on clusters of Amazon EC2 machines. The dataset contains 7 different traces, 2 of which are runs which use Big data benchmark (BDBench) using 5 machines and the rest are traces from TPC-DS benchmark. We only use the first trace which has been collected using 6 trials and in each trial, 10 queries were executed in random order. We flatten the JSON trace into a CSV file, drop the null values, and add a small delta value for the zero values to take care of the division by zero errors.

Table 6 and Table 7 summarizes the topics found from the Spark BDBench trace using AIPW and Causal Forest estimators respectively.

Table 4: Task metrics for the Lenovo trace and the Spark trace.

Dataset	Task Metric
Lenovo	CTN_CPU_REQ, CTN_MEM_REQ, CTN_CPU_USAGE, CTN_MEM_USAGE, CTN_NET_R_BYTES,CTN_NET_T_BYTES, MACHINE_CPU_USAGE, MACHINE_MEM_USAGE, MACHINE_NET_R_BYTES, MACHINE_NET_T_BYTES
Spark	JVM_GC_Time, Shuffle_Write_Time, Hadoop_Bytes_Read, Bytes_Received_Per_Second, Bytes_Transmitted_Per_Second, CPU_Utilization_Total_System_Utilization, Cpu_Utilization_Counters_Total_User_Jiffies, Executor_Run_Time, Shuffle_Bytes_Written

Table 5: Causal inference results with AIPW estimator

CTN_ID+	Topics	Confidence
`carts-5874b9657f -b95v6-2018-05-25`	[2]	[1.]
`front-end-79f895cb65 -phhm7-2018-05-24`	[6]	[1.]
`front-end-79f895cb65 -42x5g-2018-05-27`	[2, 3, 6]	[0.53, 0.24, 0.23]

Table 9 shows an example of the final inference results using AIPW and Causal Forest estimators. This causality profile has a list of metrics with their statistical significance. Each topic has several metrics with their confidence scores summing up to 1. We observe that for several jobs there are mixed causes of stragglers.

4.2.3 Comparison with human expert diagnoses

In this section, we analyse the causes behind majority fraction of the stragglers in the Spark BDBench workload. Table 7 and 6 illustrate that causal forest leads to learning more fine-grained topics as compared to AIPW. We define contradicting topics as those having topic keyword such as `X(-)` in a topic, but `X(+)` in another. The topics extracted from causal forest show lesser number of contradicting topics, hence we claim that the quality of topics identified by causal forest estimator is better as compared to AIPW estimator.

Ousterhout et al. provides expert diagnoses for the causes of stragglers for the BDBench (memory) [13]. In the expert analysis paper, Shuffle write (disk) and Garbage collection are the causes for majority fraction of the stragglers in BDBench workload. From Figure 2 (c) and (d), we analyse the top topics contributing to straggler causes. The results using AIPW do not make concrete sense in identifying the straggler causes, since both T5 and T0 have contradicting topics. T5 can be interpreted as Shuffle write (disk) with low confidence score. The results using Causal Forest identify T1, T0, and T3 as causes for most of the stragglers. T1, T3 can be interpreted as Shuffle write (disk), T0 can be interpreted as Garbage collection. From our comparison with the human expert diagnoses of the Spark traces, we can see that the interpretations obtained from causal forest estimator are more consistent with the ground truth than the interpretations from AIPW estimator.

5 Related Work

5.1 Straggler Mitigation

Dean and Ghemawat [8], pointed out several causes for stragglers such as defects in hardware, incorrect hardware configurations, resource competition. Ananthanarayanan *et al.* [15] also pointed out that stragglers consistently slow down the the completion of jobs. stragglers might occur due to some transient node behavior like how many tasks are currently running on a single node, how much is the resource contention in the node, or what is the pattern of workload on the node. Some state-of-the-art straggler mitigation methods [15] include replicating or spawning multiple redundant copies of the tasks on several nodes, and considering the execution time of the task which completes first. These methods primarily use speculative re-execution [8], [20] and scheduling approaches [15], [17]. If there is some prior knowledge about a node being slow for a particular type of task, then the scheduling mechanism will avoid nodes for tasks which are known to perform poorly on them.

5.2 Causal Discovery

As [10] does, we could explore whether stragglers are caused by interference from background daemons, poor scheduling, constrained concurrency models, or power throttling. [10] requires deep expertise in the specific system and architecture.

On the other hand, Zheng et al. proposes a ML-inspired framework, Hound that automates the generation of root cause diagnosis [21]. To output interpretable and robust explanations, Hound constructs three basic statistical estimators Hound framework's Rubin CA model uses basic

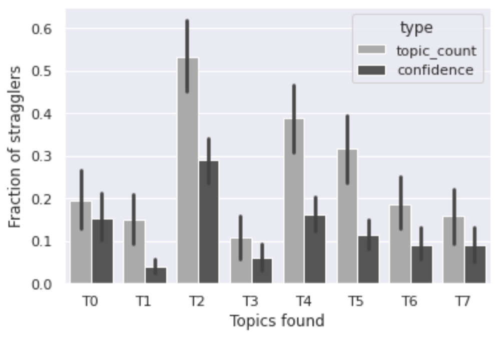
Topic	Topic Keywords	Weights
T0	`Bytes_Transmitted_Per_Second(-)` , `Executor_Run_Time(+)` , `Cpu_Utilization_Counters_Total_User_Jiffies(+)`	[0.35, 0.33, 0.32]
T1	`JVM_GC_Time(-)` , `Shuffle_Bytes_Written(+)` , `Shuffle_Write_Time(+)` , `CPU_Utilization_Total_System_Utilization(-)`	[0.43, 0.25, 0.17, 0.14]
T2	`Bytes_Received_Per_Second(-)` , `Bytes_Transmitted_Per_Second(-)`	[0.68, 0.32]
T3	`Executor_Run_Time(+)` , `Cpu_Utilization_Counters_Total_User_Jiffies(+)` , `CPU_Utilization_Total_System_Utilization(-)`	[0.36, 0.33, 0.31]
T4	`JVM_GC_Time(-)`	[1.]
T5	`JVM_GC_Time(-)` , `Cpu_Utilization_Counters_Total_User_Jiffies(+)` , `Shuffle_Bytes_Written(+)` , `Executor_Run_Time(+)`	[0.37, 0.23, 0.21, 0.19]
T6	`Shuffle_Bytes_Written(+)`	[1.]
T7	`JVM_GC_Time(+)` , `Cpu_Utilization_Counters_Total_User_Jiffies(+)` , `Executor_Run_Time(+)` , `CPU_Utilization_Total_System_Utilization(+)`	[0.37, 0.28, 0.2, 0.14]

Table 6: AIPW

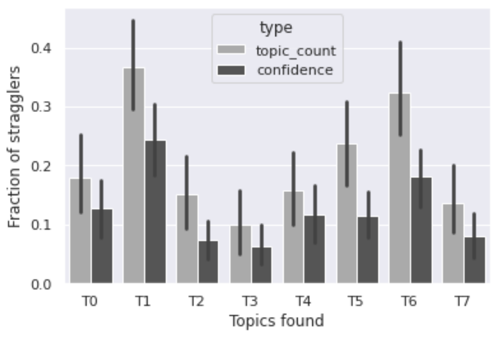
Topic	Topic Keywords	Weights
T0	`JVM_GC_Time(+)` , `Executor_Run_Time(+)`	[0.72, 0.28]
T1	`Shuffle_Bytes_Written(+)` , `Cpu_Utilization_Counters_Total_User_Jiffies(+)` , `Executor_Run_Time(+)`	[0.36, 0.33, 0.31]
T2	`Executor_Run_Time(+)` , `Shuffle_Write_Time(+)` , `Cpu_Utilization_Counters_Total_User_Jiffies(+)` , `CPU_Utilization_Total_System_Utilization(+)`	[0.48, 0.18, 0.17, 0.17]
T3	`Shuffle_Write_Time(+)` , `Bytes_Received_Per_Second(-)` , `Bytes_Transmitted_Per_Second(-)` , `CPU_Utilization_Total_System_Utilization(-)` , `Executor_Run_Time(+)`	[0.25, 0.23, 0.19, 0.18, 0.15]
T4	`Executor_Run_Time(+)` , `CPU_Utilization_Total_System_Utilization(-)`	[0.62, 0.38]
T5	`Executor_Run_Time(+)` , `JVM_GC_Time(-)`	[0.53, 0.47]
T6	`Cpu_Utilization_Counters_Total_User_Jiffies(+)` , `Executor_Run_Time(+)`	[0.55, 0.45]
T7	`JVM_GC_Time(+)`	[1.]

Table 7: Causal Forest

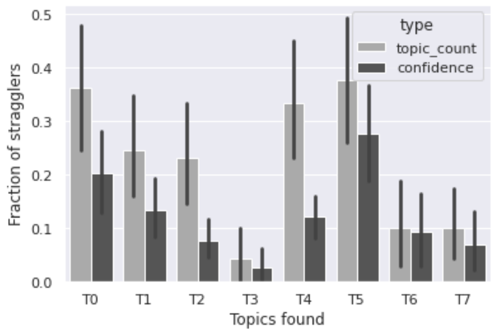
Table 8: Hound's causal topics, topic keywords, and confidence weights from the Spark BDBench trace using AIPW or Causal Forest as causal effect estimator



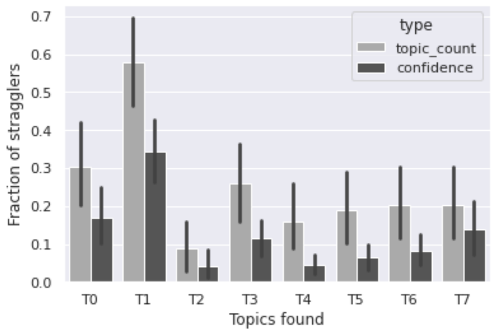
(a) From Lenovo trace, when AIPW is used in causal modeling



(b) From Lenovo trace, when Causal Forest is used in causal modeling



(c) From Spark trace, when AIPW is used in causal modeling



(d) From Spark trace, when Causal Forest is used in causal modeling

Figure 2: The causes identified for the stragglers from Lenovo and Spark traces. These plots show the distribution across all queries of the fraction of the query's stragglers that can be attributed to a particular topic.

Table 9: Causal inference results on Spark BDBench using different causal estimators. The topics are described in Table 6 for AIPW estimator and Table 7 for Causal Forest estimator. From the 86 Spark jobs we have sampled the causal inference for the latency of those jobs.

Job ID	AIPW		CF	
	Topic	Confidence	Topic	Confidence
`job_83`	`T6`	[1.]	`T1`	[1.]
`job_46`	`T5, T4`	[0.59, 0.41]	`T5, T1, T6`	[0.39, 0.39, 0.21]
`job_7`	`T1, T2, T0`	[0.56, 0.24, 0.2]	`T3, T1, T0`	[0.43, 0.39, 0.18]
`job_1`	`T7`	[1.]	`T5, T4, T6`	[0.44, 0.32, 0.24]
`job_33`	`T2`	[1.]	`T4, T3`	[0.57, 0.43]
`job_67`	`T5`	[1.]	`T7`	[1.]

statistical methods such as Predictive Modeling, Dependence Modeling and Causal Modeling. Also, unlike the above, Hound is independent of any specific system or architecture detail.

6 Conclusion

In this work, we used advanced causal estimators instead of the AIPW estimator used in the original hound implementation with the goal to see how advanced estimators can improve causal interpretations. We observed that Causal Forest gives more fine-grained topics with closer interpretations with the expert diagnoses results than AIPW. Causal Forest is quite slow running as compared to AIPW used in the original implementation of Hound. Hence in addition to the quality of results, performance is another factor that can be explored to conclude which estimator is more robust than the other. Maybe a deeper and more extensive study is required to make a decisive conclusion.

Other advanced estimators can be benchmarked with different datasets having ground truth expert diagnoses. Advanced non-graphical estimators such as Synthetic Control [5], Bayesian Rule Lists [18] or graphical estimators like Bayesian network [7] can be explored as a future work. During this work, we have tried out Bayesian networks on the simulated dataset, but could not integrate it in Hound in time. To integrate Bayesian network in Hound, we need to perform clustering on the graph embeddings to obtain the causal interpretations. The causal inference can further be extended to deep learning methods for causal inference [22].

References

[1] Bayesian network implementation. `https://github.com/almostExactMatch/daemr`.

[2] Bayesian network implementation. `https://github.com/quantumblacklabs/causalnex/tree/develop/docs/source`.

[3] Causal forest implementation. `https://github.com/timmens/causal-forest`.

[4] Hound implementation. `https://www.seas.upenn.edu/~leebcc/documents/zheng2018-hound.zip`.

[5] A. Abadie. Using synthetic controls: Feasibility, data requirements, and methodological aspects. *Journal of Economic Literature*, 2019.

[6] S. Athey and S. Wager. Estimating treatment effects with causal forests: An application. *arXiv preprint arXiv:1902.07409*, 2019.

[7] S. Chiappa and W. S. Isaac. A causal bayesian networks viewpoint on fairness. *IFIP Advances in Information and Communication Technology*, page 3–20, 2019.

[8] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[9] H. Du and S. Zhang. Hawkeye: Adaptive straggler identification on heterogeneous spark cluster with reinforcement learning. *IEEE Access*, 8:57822–57832, 2020.

[10] J. Li, N. K. Sharma, D. R. Ports, and S. D. Gribble. Tales of the tail: Hardware, os, and application-level sources of tail latency. In *Proceedings of the ACM Symposium on Cloud Computing*, pages 1–14, 2014.

[11] Y. Liu, A. Dieng, S. Roy, C. Rudin, and A. Volfovsky. Interpretable almost matching exactly for causal inference, 2019.

[12] J. K. Lunceford and M. Davidian. Stratification and weighting via the propensity score in estimation of causal treatment effects: a comparative study. *Statistics in medicine*, 23(19):2937–2960, 2004.

[13] K. Ousterhout, R. Rasti, S. Ratnasamy, S. Shenker, and B.-G. Chun. Making sense of performance in data analytics frameworks. In *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*, pages 293–307, 2015.

[14] D. Pregibon. Resistant fits for some commonly used logistic models with medical applications. *Biometrics*, pages 485–498, 1982.

[15] X. Ren, G. Ananthanarayanan, A. Wierman, and M. Yu. Hopper: Decentralized speculation-aware cluster scheduling at scale. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 379–392, 2015.

[16] D. B. Rubin. Causal inference using potential outcomes: Design, modeling, decisions. *Journal of the American Statistical Association*, 100(469):322–331, 2005.

[17] N. J. Yadwadkar, G. Ananthanarayanan, and R. Katz. Wrangler: Predictable and faster jobs using fewer resources. In *Proceedings of the ACM Symposium on Cloud Computing*, pages 1–14, 2014.

[18] H. Yang, C. Rudin, and M. Seltzer. Scalable bayesian rule lists. In *International Conference on Machine Learning*, pages 3921–3930. PMLR, 2017.

[19] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, page 2, USA, 2012. USENIX Association.

[20] M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, and I. Stoica. Improving mapreduce performance in heterogeneous environments. In *Osdi*, volume 8, page 7, 2008.

[21] P. Zheng and B. C. Lee. Hound: Causal learning for datacenter-scale straggler diagnosis. *Proc. ACM Meas. Anal. Comput. Syst.*, 2(1), Apr. 2018.

[22] Álvaro Parafita and J. Vitrià. Causal inference with deep causal graphs, 2020.