Causal inference for data-driven debugging and decision making in cloud computing

Philipp Geiger
Max Planck Institute for
Intelligent Systems,
Tübingen, Germany
pgeiger@tuebingen.mpg.de

Lucian Carata
Univeristy of Cambridge,
Cambridge, United Kingdom
lc525@cam.ac.uk

Bernhard Schölkopf
Max Planck Institute for
Intelligent Systems,
Tübingen, Germany
bs@tuebingen.mpg.de

Abstract

Cloud computing involves complex technical and economical systems and interactions. This brings about various challenges, two of which are: (1) debugging and control to optimize the performance of computing systems, with the help of sandbox experiments, and (2) prediction of the cost of "spot" resources for decision making of cloud clients.

In this paper, we formalize debugging by counterfactual probabilities and control by post-(soft-)interventional probabilities. We prove that counterfactuals can approximately be calculated from a "stochastic" graphical causal model (while they are originally defined only for "deterministic" functional causal models), and based on this sketch a data-driven approach to address problem (1). To address problem (2), we formalize bidding by post-(soft-)interventional probabilities and present a simple mathematical result on approximate integration of "incomplete" conditional probability distributions. We show how this can be used by cloud clients to trade off privacy against predictability of the outcome of their bidding actions in a toy scenario. We report experiments on simulated and real data.

1 Introduction

In recent years, the paradigm and business model of cloud computing [2] has become increasingly popular. It allows to rent computing resources on-demand, and to use them efficiently by sharing them in a smart way, in particular using auctions to sell unused resources.

Several new challenges arise from the paradigm of cloud computing. On a technical level, it is a problem to understand, control and debug the involved computing systems up to the size of several data centers, with as much automation as possible, to optimize their performance. We will go into more detail on this in Section 4.1. On an economical level, while auctions for "spot" resources help providers to use resources more efficiently, the unpredictability of their prices and performance complicates bidding and buying decisions for clients. We will go into more detail on this in Section 5.1.

In the absence of exact models, it is natural to try to address such problems using data-driven methods [13, 12, 19, 8, 21]. However, standard machine learning usually applies in settings where the underlying system is invariant, often based on the assumption that samples are i.i.d., and does not make predictions about the effect of interventions, which is important though for debugging, decision making and integration of heterogeneous data.

1.1 Contributions

The present paper takes first steps towards addressing challenges of cloud computing using *causal* models. Inferring causal models from (observational) data is notoriously hard, and convincing applications of causal modeling to real world problems are scarce. The present paper is no exception in that the main focus is conceptual rather than empirical. Our main contributions are:

- We present two theoretical results for approximations in causal modeling, Propositions 1 and 2, which are of relevance for the subsequent cloud problems and possibly beyond, in Section 3. It needs to be emphasized that the practicability of these theoretical results remains to be proved.

- In Section 4, we suggest first steps towards causal models and approximate counterfactuals as a principled, data-driven approach for addressing cloud control and performance debugging problems, integrating sandbox experiments.

- In Section 5, we use approximate integration of causal knowledge to enable cloud clients to better predict performance and costs, while preserving privacy, in a toy setting.

1.2 Structure

The remainder of this paper is structured as follows:

- in Section 3, we give brief introductions to causal models and cloud computing;

- Section 3 contains the definition of counterfactuals (in addition to our two theoretical results);

- Section 6 contains simplistic real-world and simulated experiments for our two approaches, as well as a preliminary causal model of a more realistic cloud system;

- in Section 7, we discuss related work;

- and we conclude the paper with Section 8.

2 Background

2.1 Causal models

Here we only give succinct definitions. For a more detailed introduction to causal models we refer the reader to [14, 15, 20, 17].

In this paper, we generally assume variables to be discrete, although some results may also hold for the continuous case. Let V be a set of variables. A *graphical causal model (GCM)* [14, 20] over V consists of

- a directed acyclic graph (DAG) G with V as node set, called *causal diagram* or *causal DAG*,

- a conditional probability density $p_{X|PA_X=pa_X}$ (for all pa_X in the domain of PA_X) for each $X \in V$, where PA_X are the parents of X in G.

A *functional causal model (FCM) M* is a special GCM that includes, for each observed variable X,

- a hidden root (i.e., parentless) *background variable U_X* with an arrow only to X,

- such that $X = f_X(PA_X, U_X)$ for some function f_X, i.e., X is fully determined by PA_X, U_X.

Each FCM *induces* a CGM by dropping the background variables. By *causal models* we refer to FCMs as well as CGMs.

Given a causal model M and a tuple of variables Z of M, the *post-interventional causal model $M_{\mathrm{do}\,Z=z}$* is defined as follows: drop the variables in Z and all incoming arrows from

the causal diagram, and fix the value of variables in Z to the corresponding entry of z in all remaining conditional densities. Based on this, we define the *post-interventional density of Y after setting Z to z (relative to M)*, denoted by $p_{Y|\text{do }Z=z}$, by the the density of Y in $M_{\text{do }Z=z}$. Note that we will use expressions like $p(x|y)$ as shorthand for $p_{X|Y}(x|y)$.

On a non-mathematical level, we consider M to be a correct causal model of some part of reality, if it correctly predicts the outcomes of interventions in that part of reality (clearly there are other reasonable definitions of causation).

2.2 Cloud computing

Traditionally, both businesses and individuals have used dedicated local computers, or computer networks, for storing, managing and processing data. However, this can be inefficient in several ways: the overhead of maintaining such an infrastructure is high, and one needs to buy enough computers to handle peak loads, while during normal operation most will remain unutilized

Cloud computing significantly changes this, by allowing computing resources to be rented on demand. A company, the *cloud provider*, is now responsible for running all the hardware, keeping it upgraded and sharing it amongst multiple clients. Such an infrastructure can be run in a highly efficient manner: tens or hundreds of *virtual machines (VMs)*, i.e., emulations of computer systems, chartered by different clients, run on a single *physical server* and share its resources such as central processing units (CPUs), memory and network. Note that we refer to a system as being *in production* if this system does actual work for clients and visitors and if contracts have to be met w.r.t. this system (in contrast, e.g., to an experimental system).

3 Two approximations in causal modeling

3.1 Structural counterfactuals and an approximation

Let M_0 be an FCM over a set V of variables, and let U denote the set of independent background variables in M_0. Let E, X, Y be (sets of) variables in V. The *structural counterfactual probability of Y being y, had X been x, given evidence $E = e$*, can be defined [14] based on M_0 as[1]

$$p(Y_{\text{do }X=x} = y|e) := \sum_u p(y|\text{do }(x), u)p(u|e). \tag{1}$$

Even though computer systems are "more deterministic" than many other systems, due to interactions with the environment and missing information, usually one can only infer a GCM, and not an FCM, of a computer system. Without an FCM though, counterfactual probabilities (Equation (1)) are generally not uniquely determined, i.e., they cannot be derived from a GCM. Let us give an example.

Example 1 (GCMs do not determine counterfactual probabilities). *Let $V = \{X, Y\}$ for binary X, Y, and consider the GCM M with DAG $X \to Y$ and conditionals $p_X(0) = \frac{1}{2}$ and $p_{Y|X}(0|x) = p_Y(0) = \frac{1}{2}$. M is induced by two very different FCMs. On the one hand, the FCM M_0 with structural equations*

$$X := U_X,$$
$$Y := U_Y,$$

and $U_X \sim U_Y \sim Uniform(\{0, 1\})$, where $Uniform(\{0, 1\})$ denotes the uniform distribution on $\{0, 1\}$, induces M. On the other hand, the FCM M_0' with structural equations

$$X := U_X,$$
$$Y := X \,\text{XOR}\, U_Y,$$

---

[1]Note that [14] in his definition uses functions instead of (deterministic) conditionals.

and $U_X \sim U_Y \sim Uniform(\{0, 1\})$ induces M. But in M_0 we have

$$p(Y_{\text{do }X=1} = 0|X = 0, Y = 0)$$
$$= \sum_{u_Y} p(Y = 0|\text{do }X = 1, u_Y)p(u_Y|X = 0, Y = 0)$$
$$= \sum_{u_Y} p(Y = 0|u_Y)p(u_Y|X = 0, Y = 0)$$
$$= 1 \cdot 1 + 0 \cdot 0 = 1,$$

while in M_0' we have

$$p(Y_{\text{do }X=1} = 0|X = 0, Y = 0)$$
$$= \sum_{u_Y} p(Y = 0|\text{do }X = 1, u_Y)p(u_Y|X = 0, Y = 0)$$
$$= \sum_{u_Y} p(Y = 0|X = 1, u_Y)p(u_Y|X = 0, Y = 0)$$
$$= 0 \cdot 1 + 1 \cdot 0 = 0.$$

This gives an extreme example of counterfactual probabilities not being determined by a GCM.

For a more detailed discussion of this phenomenon we refer the reader to [17].

Now we show that nonetheless counterfactual probabilities can be calculated *approximately*, and one can *know*, from only the GCM, how wrong the approximation is at most – on average. This will be important for our approach to debugging in Section 4, and, as we belief, for other areas as well.

Let M be a GCM and let Z be the set of its root variables (variables with no parents in the causal DAG). For any (sets of) variables X, Y, E in M we define the *approximate structural counterfactual* or *approximate counterfactual* as[2]

$$\tilde{p}(Y_{\text{do }X=x} = y|e) := \sum_w p(y|\text{do }(x), w)p(w|e), \tag{2}$$

where $W := Z \setminus X$.

Proposition 1. *Let M_0 be an FCM that induces a GCM M, and let Z denote the root variables in M. For all (sets of) variables E, X, Y we have*

$$\text{D}(p(Y_{\text{do }X=x}|E)\|\tilde{p}(Y_{\text{do }X=x}|E)) \leq \text{H}(E|Z), \tag{3}$$

where $p(Y_{\text{do }X=x}|e)$ is defined w.r.t. M_0 and $\tilde{p}(Y_{\text{do }X=x}|e)$ w.r.t. M.

We prove (using monotonicity of the KL divergence and properties of entropy) a generalization of Proposition 1 – Proposition 3 – in Section A.1.[3]

Example 2. *To give some intuition about the approximate counterfactual and the proposition, let us first consider the following two special cases: If M is already an "FCM" in the sense that all its variables are completely determined by the root nodes, then we have $H(E|Z) = 0$, and thus, based on Equation (3), both quantities coincide, which seems natural. If the evidence comprises the root nodes, $Z \subset E$, then the approximation amounts to the simple conditional $p(y|\text{do }(x), w)$ (where w is the part of e the corresponds to W), similar as if we had evidence on all background variables in an FCM.*

Note that for the M in Example 1, the approximate counterfactual does not help much. It can be calculated as

$$\tilde{p}(Y_{\text{do }X=1} = 0|X = 0, Y = 0) = p(Y = 0|\text{do }(x)) = p(Y = 0) = \frac{1}{2}$$

---

[2]The idea of a counterfactual definition based on only the GCM has been mentioned in [14, Section 7.2.2], but not been further investigated. Depending on the specific setting and the available information, there may be more suitable approximations to encode counterfactual-like probabilities.

[3]Note that, if we chose the set Z in Proposition 3 such that it is as "close" (in the causal diagram) to Y as possible, this could yield better approximations than simply letting Z be the root nodes, as done in $\bar{p}(Y_{\text{do }X=x} = y|e)$. We leave this as a question for future work.

As is easy to see, this implies the KL divergence between $\tilde{p}(Y_{\mathrm{do}\,X=1}|X=0,Y=0)$ and the true $p(Y_{\mathrm{do}\,X=1}|X=0,Y=0)$ under both, M_0 and M_0' of Example 1, to be 1. This KL divergence coincides with the upper bound to the KL divergence in Proposition 1, since $H(X,Y|Y)=1$ in Example 1.

The practical meaningfulness of the approximate counterfactual probability, in particular for decision making, remains subject to debate. We will briefly comment on it in Remark 1 below.

3.2 Approximate integration of causal knowledge

The following result will be important for Section 5 since it can be used to preserve some amount of *privacy*. Consider random variables $C, X_0, \ldots, X_K, Z$. A typical causal structure which satisfies the assumptions we make below is depicted in Figure 4 on page 13. Here we introduce what can be seen as an approximation to "transportability", as introduced by Bareinboim and Pearl [4], in the following simple case: we would like to know $p(z)$, we do know the mechanism $p(z|x_0, \ldots, x_K)$ plus, from a different source, $p(x_k, c)$ for all k, but we do *not* know $p(x_0, \ldots, x_K)$. Define the approximation

$$\bar{p}(z) := \sum_{x_0,\ldots,x_K,c} p(z|x_0,\ldots,x_K) \prod_k p(x_k|c)p(c). \tag{4}$$

Proposition 2. *If* $Z \perp\!\!\!\perp C|X_0, \ldots, X_K$, *then* $\mathrm{D}(p(Z)\|\bar{p}(Z)) \le \sum_k \mathrm{H}(X_k|C)$.

Note that based on the proposition, again, we can *know* how wrong the approximation is at most, using only the available information $p(x_k|c), p(c)$. A proof (again using monotonicity of the KL divergence and properties of entropy), can be found in Section A.2.

Example 3. *To get an intuition, consider the case that all X_k are fully determined by C: then $\bar{p}(z)$ and $p(z)$ coincide, which is reflected by $\sum_k \mathrm{H}(X_k|C)$ being 0. As already mentioned, an example of a causal model which implies the condition of the proposition is depicted in Figure 4 on page 13.*

While here we apply the proposition for a predictability-privacy problem in Section 5, it is more generally applicable where joint distributions are not available. In particular, while in Section 5 we will focus on approximate integration for *privacy* reasons, an even more frequent reason may be that only (insufficient) marginals are *known*. Keep in mind that stronger statements on the set of possible $p(z)$ under the available information may exist, e.g., based on ideas in [3].

4 Problem 1 – models for control and debugging – and our approach

We start with the problem statement (Section 4.1), followed by our approach (Section 4.2). Then we illustrate our approach in detail based on several toy scenarios and discuss advantages over previous work (Section 4.3).

4.1 Problem statement

Cloud computing involves technical systems of the highest complexity, which have to be controlled and debugged, ideally in a (semi-)automatic way. More specifically, the *control problem* can be stated as follows: During the operation of a cloud server many decisions automatically have to be made regarding how resources, such as complete computers, or parts, such as CPU time, are allocated among the various applications or virtual machines (VMs) of clients. The goal is to optimize this automatic decision making, based on some given utility function, encoding, e.g., energy consumption, guarantees given to customers, or simply profit.

The *(performance) debugging problem* (closely related to "performance attribution") can be formulated as follows: the general goal is to understand which component of a system contributes to what extent to the measured performance. Based on this, it can be decided

which components have to be modified to perform a "gradient step" towards the optimal performance. To give an example, a cloud computing client may wonder whether the high latency of his web server is caused from concurrent programs within his VM (which he could directly intervene on), or by other, concurrent VMs on the same physical cloud sever. We will come back to this example in Section 4.3, where we address a toy scenario, as well as Section 6.2, where we give an example of a preliminary but realistic causal model that can help in such a situation. Note that we presently focus on debugging for *individual observations*, i.e., on the unit-level.

Usually, plenty of heterogeneous knowledge and data is available about the involved systems: expert knowledge, formal program code and system specifications (often containing non-causal associational knowledge), data from the very system or similar ones, and data from sandbox experiments.

4.2 Outline of our approach

We now sketch several steps of a unified approach based on causal models, which can potentially help to address the control and the debugging problem. In what follows, we will refer to the cloud system "in production", i.e., the fully configured system with a specific set of applications, as the *"target system"*. Note that, depending on the specific setup, some steps may be canceled.

4.2.1 Step 1A: inference of causal diagram and some mechanisms

Given: the various information sources described below.

Procedure: Keep in mind that the inference procedure we describe here is usually not based on the target system itself, since some details of it (such as the specific VMs running on it) are varying quickly, but instead on *past* experience with other systems of equal or similar configuration. In particular, usually not all details of the target system are known during this step, so that some mechanisms stay underdetermined, but can be inferred later during Step 1B. As usual, the main sources for causal inference are randomized interventional experiments, observational data, and expert knowledge. A necessary condition to harness the first two sources is the decision about - and performance of - *measurements* of the system, for which we propose to use tools discussed by Carata et al. [7], Snee et al. [19].

Note the important fact that many aspects of computer systems (hardware and software) are - *by design* - modular, i.e., separable into individually manipulable input-output mechanisms, which is a central assumption in causal models [14, 17]. To give a simple example: to see if erroneous behavior is caused by the network, one can unplug the network cable and check if the error occurs nonetheless – a procedure which generally would not change any other mechanism, such as the CPU or keyboard. Furthermore, the same (or similar) mechanisms occur in different systems, which is very helpful for extrapolation from experiments. Note that there is an additional source of information which is specific to computer systems: a lot of knowledge about non-causal associations, such as which program calls which other program during execution, is available, often in a well-formatted way (e.g. program code or system architecture specifications). Such information could be translated into hypotheses on causal association (or be used for measurement selection), in a (semi-)automatic way.

The output of this procedure is a causal diagram G of the target system, together with those mechanisms, i.e., conditionals in the causal model M of the target system, which can be inferred based on past experience. For those mechanisms which cannot be known based on past experience, but only when the target system is revealed (e.g., the specific VMs running on it), but which *cannot be explored directly on the target system* either (since tentative configurations may violate contracts with clients [8, 21]), we discuss the integration of sandbox experiments in Step 1B below, which should then complete the causal model M.

4.2.2 Step 1B: design and integration of sandbox experiments

Given: an additional cloud system, the *"experimental system"*, equivalent in hardware to the target system, the causal diagram G of the target system, some variable X (e.g. performance

of some VM) in G, and the identity (e.g., VM) but not all properties of the mechanism that produces X, and whose unknown properties should be inferred during the experiment.

Procedure: The knowledge of G allows to integrate sandbox experiments in a principled way:

1. Derive all direct influences of X from G, i.e., the parents PA_X (which could include resources such as CPU time or size of requests received from the internet).

2. *Design* the sandbox experiment on the experimental system such that (1) the experimental system has the same conditional $p(x|\text{pa}_X)$ as mechanism for X (e.g., by simply running the same VM on the experimental system as is planned to run on the target system) and (2) all variables in PA_X are randomly varied.

3. Based on the gathered data, regress X on PA_X and plug the inferred conditional $p(x|pa_X) = p(x|\text{do} \, pa_X)$ as mechanism for X into M. This is possible since all parents of X were "intervened" and regressed upon.

Without going further into detail, it needs to be mentioned that the transfer of the conditional between experimental and target system can be seen as a simple example of "transportation" of causal relations as defined by Pearl and Bareinboim [16].

4.2.3 Step 1C: control

Given: causal model M of the target system, some utility U, which is variable in M or a function of one or several variables in M, and some variable X (e.g. concurrent workload, CPU time, network bandwidth) in M, which should be controlled such as to optimize u (or $p(u)$).

Procedure: As M predicts the effect on u of modifying any of its mechanisms, it can be used to find the mechanism, or "policy", $p(x|\text{pa}_X) = \pi(x|pa_X)$, which maximizes u.

4.2.4 Step 1D: observation-level performance debugging

Given: causal model M of the target system, a variable Y in M that measures the performance which we want to optimize, a performance debugging query Q, and an (individual) observation $Y = y, F = f$, where F contains all observables besides Y. (Since we move on the level of individual observations instead of populations, we term this step "observation-level performance debugging".)

Procedure: For the performance debugging query Q, we assume the following form: "In the current situation, would it improve performance Y from the current y to y', if we would set X to x', given side information $F = f$?" The side information f may contain an observation x of X. Stated this way, it seems natural to translate this query into a query for the structural counterfactual probability $p(Y_{\text{do} \, X=x'} = y'|y, f)$.[4] Then, based on Section 3.1 and in particular Proposition 1, we can calculate the approximate answer $\tilde{p}(Y_{\text{do} \, X=x'} = y'|y, f)$ from the GCM M, if $H(E|Z)$ is small, where Z is a set of root nodes.

Remark 1 (The value of (approximate) counterfactuals for performance debugging). *A remark is due regarding the notion of a counterfactual and its application to performance debugging. In the narrow sense, a counterfactual statement is always a statement about the past and so it is neither falsifiable, nor can it help for any (falsifiable recommendations regarding) future decision.*

In contrast, here we have in mind a broader notion of a counterfactual: a situation where one observes a system with a poor performance and asks how the performance could be debugged when the system remains in the "same" *state, or visits the same or similar states again. (In the language of causal models, "state" means the tuple of background variables.) This question is relevant in situations where the debugging action can be performed quickly after the observation of poor performance, and where ones assumes that the* state changes *comparably slowly, i.e., the state varies smoothly with time.*[5] *Alternatively, the question*

---

[4]Clearly, there are other ways to formalize attribution and debugging.

[5]It seems that a more thorough analysis of this argument might be fruitful, as it could theoretically justify the frequent usage of counterfactual reasoning in everyday life. We leave this to future work.

can be relevant if one has a good "subjective" judgement about the similarity of the state between two points in time – if the judgement is based on pbjective observables though, a non-counterfactual form of reasoning may be more appropriate.

Situations where counterfactual reasoning may be useful arise, in particular, whenever one does not assume to "know" the population-level distribution of the state well enough (but one beliefs in the structural equations), for instance, because it varies with time, and instead one wants to reason on the observation-level, i.e., unit-level. Because on the population-level, there are better ways for decision making than counterfactual reasoning, see Step 1C.

We propose one way to formalize performance debugging questions, and to answer them, based on one possible formalization of counterfactual probabilities proposed by Pearl [14]. It remains an open question whether there are better formalizations than ours for the debugging questions we consider, and whether the general notion of a counterfactual probability, as well as its formalization by Pearl [14], are sensible. For a discussion, see also [17].

Note that an additional issue, which we are not able to settle here, is how close our approximation of a counterfactual comes to the true counterfactual in practice.

4.3 Application to toy scenarios and discussion of potential advantages over previous approaches

For researchers familiar with causal inference, some of the steps described above may seem trivial. However, all current approaches to the described problems we are aware of are lacking a *principled* (formal) language, with concepts such as causal sufficiency, for such things as integration of sandbox experiments and performance debugging.

We will now give toy examples to make the approach outlined in Step 1B through Step 1D more concrete, and simultaneously show the advantages of our approach based on causal models over some previous approaches. (For examples of applications of Step 1A, see Sections 6.1 and 6.2.) Keep in mind that, clearly, the approach we outlined does not completely solve the problem: the inference of knowledge it relies on remains a challenge as with all other approaches. However, our approach may be less prone to errors and more data-efficient.

- **Step 1B:** Integrating sandbox experiments without a principled approach [8, 21], can lead to errors: e.g., if not all parents (direct causes) of a variable X are varied during the experiment and regressed upon afterwards or, say, X is regressed on its causal children. Any methodology that does not include reasoning about concepts such as causal effect, causal sufficiency or randomization is prone to such mistakes. Let us give a toy example of how our approach works for sandbox experiments, and how other approaches can go wrong in terms of variation and regression.

  Example 4 (Design and integration of sandbox experiments, and possible mistakes)**.** *Imagine we are the cloud provider and we want to decide whether we can put some VM A on some cloud server, where already other concurrent VMs are running. Let $L \in \{0, 1\}$ denote the performance of (the main application running inside) A, with $L = 0$ denoting good, and $L = 1$ bad performance. For instance, L could denote some latency. Assume that Figure 1 depicts the correct causal DAG of the target system, i.e., when A would be running on the mentioned cloud sever. In particular, the performance depends on two factors, say amount of requests $R \in \{0, 1\}$ coming into A from the internet, on the one hand, and usage $S \in \{0, 1\}$ of the CPU of the cloud sever by the concurrent VMs, on the other, where 0 stands for "low" and 1 for "high". And in turn, R, S depend on H which may denote the state of the internet users, which send requests to A but potentially also to concurrent VMs and therefore also influence S. (Alternatively, H could denote a* parameter *for the behaviour of the internet users, i.e., for the distribution of their states.)*

  Assume the true mechanism underlying L to be

  $$L := R \operatorname{AND} S,$$

  where AND *denotes the logical "AND". I.e., the performance is bad iff A has to serve many requests ($R = 1$) and at the same time CPU usage by concurrent VMs is high ($S = 1$). Furthermore, assume that on the target system, we have $R \approx S$.*
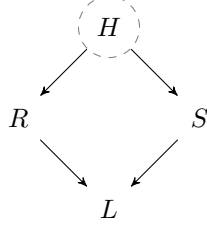
8

Figure 1: Causal diagram when running VM A on the target system. Not varying S or not regressing on S during the sandbox experiment can lead to wrong predictions of performance L on the target system, especially when some hidden source H (say internet users) introduces strong correlations between R and S.

For instance, this could be due to the fact that A and concurrent VMs serve internet users in the same time zone. Additionally, assume $p(R = 0) = p(S = 0) = \frac{1}{2}$.

Suppose we have inferred the causal DAG in Figure 1 based on Step 1A. We now want to infer the mechanism underlying the performance L, so, following Step 1B (taking L as X), we would perform a sandbox experiment where we would vary both, R and S, and afterwards regress on both, R and S. We would correctly infer the mechanism $L := R \text{ AND } S$. Additionally knowing $p(r, s)$ (say from previous experience, or from reports by the cloud clients) we would correctly predict the probability of bad performance of A on the target system, $p(L = 1)$, to be

$$\sum_{r,s} p(L = 1 | r, s) p(r, s) = 0 \cdot \frac{1}{2} + 1 \cdot \frac{1}{2} = \frac{1}{2}$$

In contrast, without such a principled approach, two things can happen.

If in the sandbox experiment, only R is varied and regressed upon, while S is kept to a constant 0 (because it was not properly inferred or communicated as an influence factor, or simply because on the experimental system no concurrent VMs are emulated), then $p(L = 1)$ would be wrongly predicted as

$$\sum_{r} p(L = 1 | r, 0) p(r) = 0 \cdot \frac{1}{2} + 0 \cdot \frac{1}{2} = 0.$$

And even if in the sandbox experiment, S would be varied according to the correct $p(s)$ on the target system (e.g., because the concurrent VMs of the target system would be emulated well on the experimental system), but if one would forget about regressing on S, then still one would wrongly predict $p(L = 1)$ to be

$$\sum_{r} p(L = 1 | r, s) p(r) p(s) = 0 \cdot \frac{1}{4} + 0 \cdot \frac{1}{4} + 0 \cdot \frac{1}{4} + 1 \cdot \frac{1}{4} = \frac{1}{4}.$$

Clearly, this was only a simplistic toy example, but to the best of the knowledge of the author, such problems have not been thematized in the literature [8, 21] yet.

- **Step 1C:** Causal models provide a principled tool for control of cloud systems that allows to integrate various forms of information, such as results of sandbox experiments obtained in 4.2.2. Furthermore, compared to, e.g., [13], which is based on adaptive control, an advantage of using causal models is that they allow to encode and integrate knowledge about which mechanisms vary and which stay invariant.

  Example 5 (Control based on causal models). *Consider A, the same VM as in Example 4, with performance L. Recall that there we inferred the mechanism for L to be $L := R \text{ AND } S$.*

  Now assume that we consider a different target system than in Example 4, namely, a system that involves a policy $\pi(r|s)$ that controls the amount S of CPU that is occupied by VMs other than A. We depict the causal DAG in Figure 2.
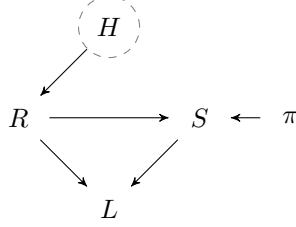
Figure 2: Causal diagram when running A on a system controlled by policy π. It is similar to the system in Figure 1, except that now S is influenced by the choice of the policy (as well as the current R which serves as an input to the policy), and therefore we add π to the diagram and draw an arrow to S. Note that handling the policy, which is rather a parameter than a variable, in such a way is similar to the use of so-called "selection diagrams" in [16], where the mechanisms that vary are marked by special nodes with arrows to them.

Suppose the goal is as follows: keep the probability of poor performance below $\frac{1}{2}$, i.e., $p(L = 1|\pi) \leq \frac{1}{2}$, while allocating as little CPU as possible to A, i.e., minimizing $p(S = 0|\pi)$ (so that more CPU can be used by other VMs). Furthermore, assume $p(R = 0) = \frac{1}{2}$, as in Example 4.

Using the causal DAG and "plugging in" our knowledge of the mechanisms, it is easy to see that the optimal policy is $\pi(S = 1|r) = 1$, i.e., always occupy the CPU by other VMs. Because then

$$p(L = 1|\pi) = \sum_{r,s} p(L = 1|r, s)\pi(s|r)p(r) \tag{5}$$

$$= p(L = 1|0, 1)\frac{1}{2} + p(L = 1|1, 1)\frac{1}{2} \tag{6}$$

$$= 0 + \frac{1}{2} = \frac{1}{2} \tag{7}$$

so the goal w.r.t. performance L is still met. This shows how causal models provide a principled tool to integrate sandbox experiments, based on Step 1B, to perform control, as proposed in Step 1C (the S here corresponds to the X there).

Let us mention a potential advantage of control based on causal models in case cloud systems are time-varying. Assume H denotes a parameter for the behavior of the internet users (we indicated this meaning in Example 4.2.2). Suppose H varies for some reason, say due to an ad campaign, in an unpredictable way. We know that the behavior of the internet users influences L only via R, since the rest of the cloud system is not affected by the internet. This knowledge is encoded in the causal DAG in Figure 2. Based on this, we have

$$p(l|r, s, h) = p(l|r, s).$$

So we have formally reasoned that even if H varies, the mechanism $p(l|r,s)$ stays the same. Hence, to derive the new optimal policy π, all one has to do is to infer the new $p(r)$ and plug it into Equation 5 (and optimize for π). Furthermore, we can be certain that we identified the new system and the new optimal policy (given our assumptions are correct). This sort of reasoning has been analyzed, on a more general level, by Pearl and Bareinboim [16] (but they do not apply it to control settings).

In contrast, approaches to (adaptive) control for cloud computing which are not based on modularity and such reasoning [13] may try to infer the complete information, $p(r)$ as well as $p(l|r,s)$, from scratch upon a variation of H, assuming it to be a completely new "environment". And even if such approaches utilize the invariant $p(l|r,s)$ after a variation of H in one way or another, they are usually missing the language to reason about the identifiability of the new system (after the variation in H), as we did above based on causal models.
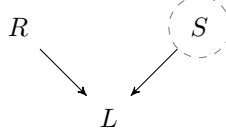
Figure 3: Causal diagram for observation-level performance debugging in a toy setting. S is unobserved, but nonetheless we assume $p(l|r,s)$ to be known, may it be that the provider publishes it, or the client knows it from own (sandbox) experiments.

> *It needs to be emphasized that here we considered an overly simplistic scenario. In more complex and realistic scenarios, there are much more mechanisms involved that could potentially vary or stay invariant, respectively. See Section 6.2 for an example of a causal DAG of a more realistic but still simple cloud system.*

- **Step 1D:** We now give en example for how observation-level debugging can be performed based on Step 1D. This approach can be seen as complementary to other methods for this problem [12], where errors may arise from confusing causation with correlation, or where it is more difficult to integrate heterogeneous knowledge such as sandbox experiments.

Example 6 (Observation-level performance debugging)**.** *Note that, while this is a toy scenario, the assumptions we make in this example regarding what is known/observed and what not are close to realistic [19].*

Similar as in Example 4, consider a VM with performance (latency) L running on a cloud system, with $R \in \{0,1\}$ denoting the amount of incoming requests, and $S \in \{0,1\}$ the amount of, say, CPU time allocated to concurrent VMs (0 stands for "low" and 1 for "high"). Here, denote the VM by B. In contrast to Example 4, assume the causal DAG depicted in Figure 3. Furthermore, let $L \in \{0,1,2,3\}$ and the structural equation for L be given by

$$L := R + S + U_L, \tag{8}$$

with $p(U_L = 0) = \frac{1}{2}$. Suppose $p(S = R) = p(S = 0) = \frac{1}{2}$, where $p(S = 0) = \frac{1}{2}$ may be seen as encoding some prior belief.

Now assume that the client whom B belongs to wonders, whether it would improve the latency L to a desired 0 in the current situation where she observes $L = 2, R = 1$, if she decreased the amount of incoming requests to a lower level, i.e., if she set R to 0. (Note that "current situation" can include the nearby future, if the unobserved variables vary comparably slowly, see Remark 1.) She does not *observe S due to neither the cloud provider nor other clients publishing this information. This is a realistic assumption in cloud computing. Based on Step 1D, she translates this question into a query for the counterfactual probability $p(L_{\mathrm{do}\,R=0} = 0 | R = 1, L = 2)$.*

Suppose that while S is not published, $p(l|r,s)$ is known, may it be that the provider publishes it, or the client knows it from own (sandbox) experiments. That is, $p(r), p(s)$ and $p(l|r,s)$ are give, but not the structural Equation 8 itself. Now, although the structural Equation 8 would be needed to calculate the counterfactual $p(L_{\mathrm{do}\,R=0} = 0 | R = 1, L = 2)$ exactly (see Example 1) she can calculate the approxi- mate *counterfactual probability defined in Equation 2 as*

$$\tilde{p}(L_{\mathrm{do}\,R=0} = 0 | R = 1, L = 2)$$
$$= \sum_s p(L = 0 | \mathrm{do}\,R = 0, s) p(s | R = 1, L = 2)$$
$$= \sum_s p(L = 0 | \mathrm{do}\,R = 0, s) p(L = 2 | R = 1, s) p(s | R = 1) \frac{1}{p(L = 2 | R = 1)}$$
$$= \frac{1}{4},$$

where we plugged in R, S for the set of root variables Z, which yields S as W, and as evidence E we took (R, L) with value $(1, 2)$. Based on this, she concludes that the probability that setting R to 0 helps for decreasing latency L to 0 is rather small (in the current situation).

Note that the true counterfactual probability (Equation 1 in Section 3.1) in this specific case is given by

$$
\begin{aligned}
&p(L_{\mathrm{do}\,R=0} = 0 | R = 1, L = 2) \\
&= \sum_{u_R, u_S, u_L} p(L = 0 | \mathrm{do}\, R = 0, u_R, u_S, u_L) p(u_R, u_S, u_L | R = 1, L = 2) \\
&= \sum_{u_L, s} p(L = 0 | \mathrm{do}\, R = 0, s, u_L) p(s, u_L | R = 1, L = 2) \\
&= 0 + p(L = 0 | \mathrm{do}\, R = 0, S = 0, U_L = 1) p(S = 0, U_L = 1 | R = 1, L = 2) \\
&\quad + p(L = 0 | \mathrm{do}\, R = 0, S = 1, U_L = 0) p(S = 1, U_L = 0 | R = 1, L = 2) + 0 \\
&= 0,
\end{aligned}
$$

which would lead to the even stronger conclusion that setting R to 0 for decreasing L to 0 would not work at all.

Note that the upper bound of Proposition 1 here takes the value

$$
\begin{aligned}
\mathrm{H}(R, L | R, S) &= \mathrm{H}(L | R, S) \\
&= \sum_{r,s} p(r, s) \mathrm{H}(L | r, s) \\
&= 1.
\end{aligned}
$$

Recall that we picked $p(U_L = 0) = \frac{1}{2}$, i.e., rather strong noise. For less noise, the approximation would be even better and the bound smaller.

Note that generally, one could try to *learn* (in the sense of machine learning) things such as how to perform and integrate the experiment [19], but one would always have to rely on prior assumptions, which may then be more difficult to encode.

5 Problem 2 – cost predictability versus privacy – and our approach

We start with the problem statement (Section 5.1), followed by our approach (Section 5.2). Then we present a toy example (Section 5.3), and some additional remarks (Section 5.4).

5.1 Problem statement

Here we consider an economical aspect of cloud computing. Currently, one common way for clients to purchase cloud resources from a provider is via an auction mechanism for "spot" (i.e., short-term) resources, which can be described in a simplified way as follows: The customer enters a bid, e.g., for an hour of usage. Once the price determined by the provider (based on supply, demand, and other private factors) drops below the bid, the customer gets the resource, usually as long as her bid exceeds the price (within the hour). This approach has several advantages, in particular for the provider: he can sell resources which are unused but which fluctuate a lot (due to guarantees given to "dedicated" or "on-demand" customers). But clients can profit as well: the spot resources are usually significantly cheaper than the long-term dedicated resources.

An obvious drawback of spot resources is that this kind of mechanism comes with a high *uncertainty* for the clients: it is hard to tell how the prices will evolve in the future, and, in particular, purchased resources can be terminated in an unforeseeable way, which is, to some extent, due to the *unpredictability of the other clients*. Therefore, if the client does not want to take these risks which can significantly harm his/her business, they often avoid this mechanism.
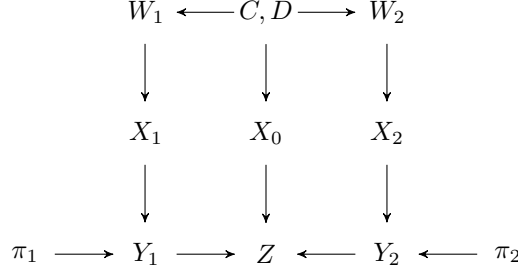
$$W_1 \longleftarrow C, D \longrightarrow W_2$$

$$\downarrow \qquad\qquad \downarrow \qquad\qquad \downarrow$$

$$X_1 \qquad\qquad X_0 \qquad\qquad X_2$$

$$\downarrow \qquad\qquad \downarrow \qquad\qquad \downarrow$$

$$\pi_1 \longrightarrow Y_1 \longrightarrow Z \longleftarrow Y_2 \longleftarrow \pi_2$$

Figure 4: Causal diagram G_2. D is hidden.

5.2 Sketch of our approach

In what follows, we present a very first step towards addressing the problem based on available observational data and causal models. We assume that there is one provider, and clients $1, \ldots, K$. By "stakeholders" we refer to provider and clients together. For each point in time (say, the beginning of an hour), let X_k denote client k's demand for the next hour, Y_k the cloud product that the client buys from the provider, W_k the information based on which the client decides her demand (e.g., hour of the day), which may not always be fully known though, and π_k her policy determining which cloud product Y_k to buy, given her demand X_k. Let X_0 denote the provider's pricing parameter at that time point (which may depend, e.g., on energy costs), and let Z denote the outcome of the provider's mechanism applied to the Y_k. (Generally, Z can include the price as well as say termination of spot resources; for simplicity, let it only denote the cost/price for the moment, which can comprise the indirect costs resulting from loss of visitors through termination.)

We assume the following simple mechanism (which is a simplified version of the auction described above): all clients k always get the product they want, but the subsequent price vector Z varies and is not known in advance. The causal diagram G_2 for the complete causal structure, for the case $K = 2$, is depicted in Figure 4. The role of C will be explained below, while D denotes the hidden part of the confounder (C, D).

Our approach to the uncertainty problem, towards more predictable prices and subsequent reduced costs, is based on the idea that clients may not want to share all, but are willing to share some of their information between each other. More specifically, we propose the following two-step procedure which allows the clients to *trade off privacy versus predictability interests*, by jointly picking a variable C such that $p(X_k|C)$ allows an approximate prediction of Z which still preserves some privacy.[6]

5.2.1 Step 1A: jointly picking C

First, all stakeholders k pick their candidates for C (possibly based on a given list and some "privacy budget"), balancing their privacy interests against minimizing $\mathrm{H}(X_k|C)$. If the intersection of their candidates is non-empty, they reveal $\mathrm{H}(X_k|C)$ for all k and joint candidates C.[7] They pick the C that minimizes $\sum_k \mathrm{H}(X_k|C)$ to optimize the predictability, based on Proposition 2.

5.2.2 Step 2B: prediction and individual decision

Now all clients k reveal their $p(x_k|c)$. $p(c)$ is assumed to be common knowledge. Furthermore, all $p(y_k|x_k, \pi_k)$ are either known a priori (based on the possible products the provider offers)

---

[6]An extreme approach would be to directly infer a joint model for all clients from their joint data (i.e., considering all clients as a "single client"). Here we assume that this is not possible, due to heterogeneous data, privacy interests, etc.

[7]If the intersection is empty, the procedure is canceled without result, and the stakeholders proceed in the classical, non-collaborative way.

or revealed now. The provider reveals $p(z|x_0, y_0, \ldots, y_K)$ and $p(x_0|c)$. Now $\bar{p}(z|\pi_1, \ldots, \pi_K)$ can be calculated, based on Equation 4. More specifically, we have

$$\bar{p}(z|\pi_1, \ldots, \pi_K) = \sum_{x_0, \ldots, x_K} p(z|x_0, \ldots, x_K; \pi_1, \ldots, \pi_K) \prod_{k=0}^{K} p(x_k|c)p(c)$$

$$= \sum_{x_0, \ldots, x_K} \left(\sum_{y_1, \ldots, y_K} p(z|x_0, y_1, \ldots, y_K) \prod_{k=1}^{K} p(y_k|x_k; \pi_k) \right) \prod_{k=0}^{K} p(x_k|c)p(c)$$

Then, based on Proposition 2, the clients narrow down the set of possible $p(z|\pi_1, \ldots, \pi_K)$ to those for which

$$\mathrm{D}(p(Z|\pi_1, \ldots, \pi_K) \| \bar{p}(Z|\pi_1, \ldots, \pi_K)) \leq \sum_k \mathrm{H}(X_k|C).$$

Based on this constraint on $p(z|\pi_1, \ldots, \pi_K)$, each client k decides on their strategy π_k, e.g., based on game-theoretic considerations.

5.3 Application to toy scenario

To illustrate the approach, let us give an example.

Example 7. *A cloud provider, Clark, offers to his clients, Alice ($k = 1$) and Bob ($k = 2$), monthly (dedicated) large resources ($Y_k = 2$), rather expensive, or hourly spot small ($Y_k = 0$) and large ($Y_k = 1$) resources, which are usually cheaper. However, if Alice and Bob happen to both order large spot resource for the same hour, the cost for both of them ($[Z]_1, [Z]_2$) is significantly higher than the hourly rate for the monthly large resource, since Clark may have to buy a new resource, or he may have to cancel one of his client's applications, causing the loss of web site visitors. Now assume Alice and Bob, during Step 1A, pick the hourly weather forecast, which is 0 for sunny and 1 for cloudy, for C, since it is public information anyway that both their web sites are* weather related*: Alice runs a website for* outdoor *activities, Bob one for* indoor *activities, both in the same region. And the remaining uncertainty w.r.t. their demand (X_k being 0 for "small" or 1 for "high"), i.e., $H(X_k|C)$, is small. The causal diagram for this scenario is G_2 depicted in Figure 4. Based on this, Alice and Bob can conclude that they will rarely require a large resource at the same time, and they can go for spot resources as their respective (dominant) strategies π_k.*

5.4 Discussion

In some cases, the provider could infer the joint distribution of all X_k, based on past data, which would contain all relevant information. However, the complete system is so complex that it is unlikely to be *stationary*. Note that during each step, already some information is revealed, but this is transparent to the stakeholders. Limitations of our approach are that (1) the clients may not even be willing to reveal their $p(x_k)$, or (2) X_k may not be predictable or the model may be wrong (although humans and organizations usually do plan ahead).

It needs to be emphasized, that here we ignore strategic aspects, which can lead to problems in certain scenarios. As a potential next step, such aspects could be analyzed based on game theory [18].

6 Experiments

6.1 Control and debugging problem on simple but real cloud system

Here we test small parts of our approach in Section 4.2 on a very simple, but real cloud system: a physical server running a specific application (a web server) together with some concurrent workload (another web server). The system we consider has the same causal DAG as two of the examples in Section 4.3. And while the scenarios are generally similar, the system we consider here is simpler, for experimental purposes.
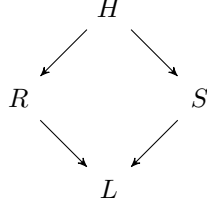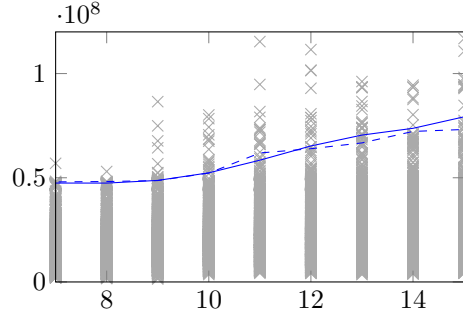
Figure 5: Causal diagram G_1.

Figure 6: X-axis: Number of simultaneous requests $S = s$. Y-axis: 99th percentile of prediction $\hat{p}(l|\text{do } s)$ (dashed blue) is close to 99th percentile (solid blue) of ground truth test data from $p(l|\text{do } s)$ (subsample in gray).

A source H keeps sending simultaneous request to application and concurrent workload (drawn from a multivariate correlated Poisson distribution), of which R are received by the application and S by the concurrent workload. Then, for each request, the latency (performance) of the application is measured in nanoseconds by L.

We examine how well Step 1A works. First, we infer the causal diagram G_1 depicted in Figure 5, as well as an estimate of $p(r, s, l)$ from observational samples of the system, based on Step 1A, and together denote them by (incomplete) M_1. Then, from M_1, using back-door adjustment [14], we derive a prediction $\hat{p}(l|\text{do } s)$ for $p(l|\text{do } s)$. Besides Step 1A, this tests the applicability of Step 1C, when thinking of a simple controller that outputs a constant for S (e.g. by putting the application on another machine with such a concurrent workload), as well as Step 1D which relies on post-interventional distributions (of an updated model, though). The outcome is depicted in Figure 6, where we use the 99th percentile as statistic, which is common in cloud computing. It shows that the prediction is close to the ground truth test data, both in magnitude and in trend.

6.2 Example of a more realistic cloud system

The experiments in the previous Section 6.1 were performed on an overly simplistic system. Here we want to give an example of a preliminary, partial causal model (causal DAG plus some knowledge on the mechanisms, e.g., additivity) of a more *realistic* system to which our approach in Section 4.2, in particular the performance debugging in Step 1D, is meant to be applied. Note that this is merely for illustration purposes, we do *not test* any hypothesis here.[8]

We consider a cloud sever running serveral VMs. We focus on one specific VM, call it A for the moment. Inside the VM A, a web sever B (more specifically: "lighttpd") runs. We

---

[8]The inference of the causal model of – and the application of our approach to – such a complex system turned out to be more difficult than expected. Therefore, no evaluation of our approach applied to this system can be reported at this stage.
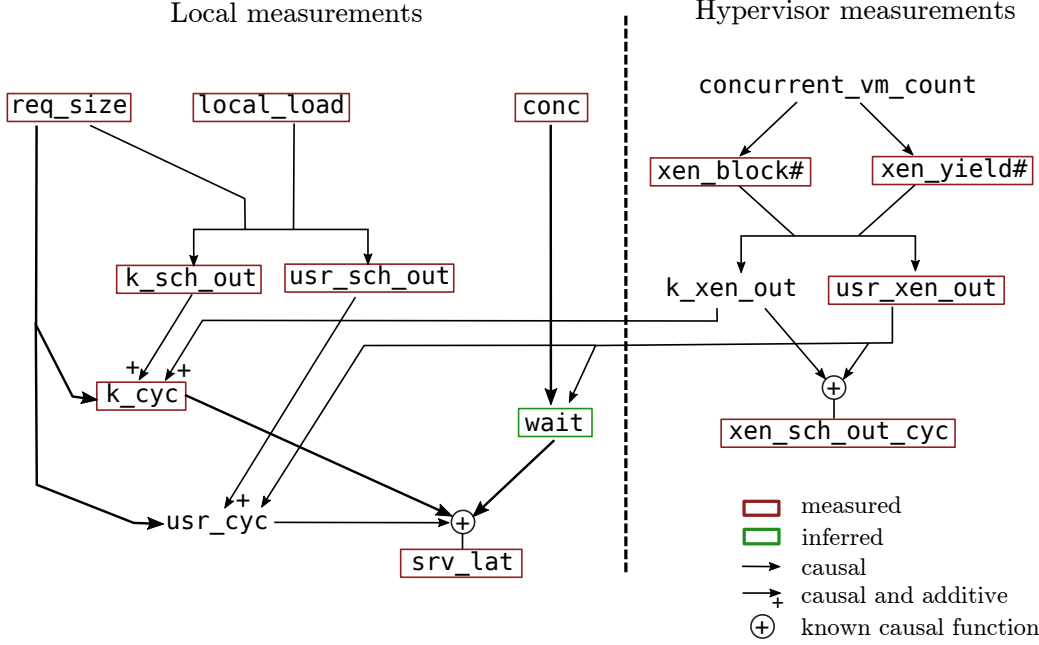
Figure 7: Example of a preliminary causal DAG of a cloud system. Variables on the left side are measured within the VM A, that runs together with other VMs on the cloud server. The right side contains measurements outside the VM (the "hypervisor" is the program that is responsible for allocating the cloud server's resources among the VMs). The objective is to minimize the latency of some web server B running in A, denoted by "srv_lat", while keeping utilization by other VMs, denoted by "concurrent_vm_count", as high as possible. Possible *manipulations* include reducing the workload within the VM A, denoted by "local_load", versus changing the number of concurrent VMs. If the causal model is good, it can help to pick the optimal manipulations. The figure is taken from [6] which also gives descriptions of the remaining variables not described here.

consider the following observed and hidden variables, among others, measured inside and outside A[9]:

- "req_size": size of the file requested by an internet user from the web server B;

- "local_load": resource-consuming activity of other applications in A, besides B;

- "concurrent_vm_count": number of VMs running concurrently with A on the physical sever (outside A);

- "srv_lat": latency of the web server B, which can be seen as part of the *objective* which needs to be minimized.

We depict the partial causal model in Figure 7. It is taken from Carata [6], who also gives descriptions for all other variables in the figure not discussed here. Note that this is a model of an *experimental* system, while on a system in production, some variables, such as "local_load" and "concurrent_vm_count", could be influences by a (hidden) common cause, similar to H in the previous experiment in Section 6.1.

This model was inferred as described in Step 1A, in an iterative and sequential way, based on non-causal associational knowledge about the program execution structure (known from the program code) as well as the general system architecture, further expert knowledge, and independence tests on sampled data. As can be seen, often the integrated knowledge allows

---

[9]In cloud computing, it is important to distinguish between inside and outside of A, since, for privacy reasons, often only things inside A can be known to the client that A belongs to.
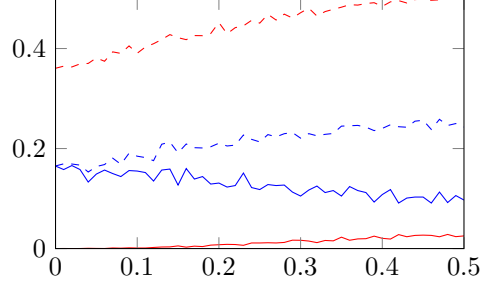
Figure 8: X-axis: Parameter r (the higher, the more influence from D). Y-axis: $\bar{p}(z)$ (dashed blue) is close to $p(z)$ (solid blue) even when D gets strong and C weakens; $\frac{1}{4}\mathrm{D}(p(Z)\|\bar{p}(Z))$ (solid red), $\frac{1}{4}\sum_k \mathrm{H}(X_x|C)$ (dashed red).

to draw conclusions on the *additivity* of mechanisms, which can be based on the fact the runtime of one program essentially is the sum of the runtimes of its subroutines.

Such a model could help for decision making in various ways, for instance for the performance debugging problem mentioned in Sections 4.1 and 4.3: A cloud client, the owner of A, may observe a high latency ("srv_lat") of his web server B, together with some other variables. He wonders if, in this situation, the high latency is caused by other programs within his VM A ("local_load"), by other, concurrent VMs ("concurrent_vm_count") running on the same physical cloud sever, or simply by large requests ("req_size") coming in at that moment. Based on this, he could conclude whether he should intervene on "local_load", which may be the simplest, or rather intervene on "concurrent_vm_count" say by changing to another cloud product, such as a dedicated server, which may be more expensive.

6.3 Predictability-privacy problem on simulated data

For our approach in Section 5.2 to work, $\bar{p}(z)$ has to approximate $p(z)$ reasonably well. Here we examine to what extent this is the case in a simulated version of the toy example in Section 5.3, additionally testing how tight the bound in Proposition 2 is. Compared to the toy example, we restrict ourselves to spot resources, i.e., $\{0,1\}$ for Y_k, and assume the following specific mechanisms: The policy π_k is for both to simply purchase their demand ($Y_k := X_k$), Clark's pricing is "cheap" ($Z = 0$) versus "very expensive for one of them since both want large" ($Z = 1$), in particular $Z := Y_1 \text{ AND } Y_2$. Furthermore,

$$X_k := C \text{ XOR } D \text{ XOR } N_{X_k},$$

where D is some confounder which Alice and Bob do not want to reveal.

Now for "each" $0 \leq r \leq 0.5$, we draw 1000 samples of $C \sim Bernoulli(0.5 - r)$, $D \sim Bernoulli(r)$ to find out how wrong $\bar{p}(z)$ gets when increasing the confounder D that is not revealed or adjusted for, and $N_{X_k} \sim Bernoulli(0.2 - 0.2r)$ (to also examine a little variation in the noise strength). The outcome is depicted in Figure 8. It shows that $\bar{p}(z)$ is a good estimate in this simple setting (which is also due to the fact that already $p(x_1), p(x_2)$ alone reveal something about $p(x_1, x_2)$). It also shows that (in this setting), the bound from Proposition 2 may be improvable, as the dashed red line is far away from the solid red line.

7 Related work

Regarding Section 3, approximations to non-identifiable quantities in causal models were examined by Balke and Pearl [3]. While their technique does not seem directly applicable to the setup of Proposition 1, it may allow to derive stronger statements, i.e., further narrowing down the set of possible $p(z)$, than Proposition 2, which could be examined in future work. We discussed some related work for Section 4, i.e., the control and debugging problem, in Section 4.3. Additionally, maybe the work closest to our investigation in that section is by Lemeire et al. [10], which suggests to use causal models for performance modeling of programs, but does not consider counterfactuals, or more complex computing systems.

Generally, the utilization of modularity based on causal models in that section is strongly inspired by the theory of "transportability" of causal relations developed by Pearl and Bareinboim [16], however, that theory has not been applied to (cloud) computing problems so far. The relation between causality and control is also considered by Bottou et al. [5]. Regarding Section 5, the work by Angel et al. [1] can be seen as related in that they allow the provider to hide their exact costs while still making some information of the costs available to others. The work by McSherry and Talwar [11] investigates privacy-preserving mechanisms, but does not consider the integration of the revealed information to an (estimate) of a causal model.

8   Conclusions

This paper assayed how causal inference can, in principle, help with technological and economical problems in cloud computing. Guided by these problems, we presented two theoretical results for approximate causal inference, and reported initial experimental results. The application of causal inference in this domain is, to the best of our knowledge, the first of its kind. We believe the potential in this area is very significant, both for applications and for methodological work. Problems in computing systems, which often require sophisticated interventions to bring the system closer to its "optimum", rarely fit the classical settings that machine learning excels at.

In particular, for issues such as integration of sandbox experiments, (formally) reasoning about concepts such as causation, causal sufficiency and randomization seems crucial, and methodology which neglects this, such as classical machine learning, may be prone to errors. Another concept which plays an important role in causal modeling (but, of course, also in some other areas) is that of identifiability, which helps to "critically" reason about what can and what cannot be inferred based on the given. We used it for the control problem for cases that only some "modules" of the system vary.

A next step would be to extend the experimtens on real cloud systems, such as the system for which a preliminary model was derived in Section 6.2, and based on this, advance the approach we sketched in Section 4. Another future step would be to use aspects of game theory and mechanism design, to extend our approach for the predictability-privacy trade-off in Section 5.

A   Appendix

Here we present proofs for Section 3.

A.1   Generalized version and proof of Proposition 1

We start by stating and proving a generalization of Proposition 1.

Proposition 3 (Generalization of Proposition 1). *Let M_0 be a FCM over discrete variables that induces a GCM M. Let the triple (X, Y, Z) of (sets of) variables in M be such that $(Y \perp\!\!\!\perp An(Z)|Z)_M$ (i.e., are d-separated [14]) and X does not influence $W := Z \setminus X$. Let E be an arbitrary set of variables in M. Let*

$$p^Z(Y_{\mathrm{do}\, X=x} = y|e) := \sum_w p(y|\mathrm{do}\, X = x, w)p(w|e). \tag{9}$$

Then

$$\mathrm{D}(p(Y_{\mathrm{do}\, X=x}|E)\|p^Z(Y_{\mathrm{do}\, X=x}|E)) \leq \mathrm{H}(E|Z) \tag{10}$$

(where $p(Y_{\mathrm{do}\, X=x}|E)$ is defined w.r.t. M_0 and $p^Z(Y_{\mathrm{do}\, X=x}|E)$ w.r.t. M).

This is a generalization of Proposition 1. To see this, let Z denote the set of root nodes of M. This implies

$$p^Z(Y_{\mathrm{do}\, X=x} = y|e) = \tilde{p}(Y_{\mathrm{do}\, X=x} = y|e)$$

for $p^Z(Y_{\mathrm{do}\, X=x} = y|e)$ as defined above and $\tilde{p}(Y_{\mathrm{do}\, X=x} = y|e)$ as defined as in Section 3.1. But Proposition 3 above applied to this $p^Z(Y_{\mathrm{do}\, X=x} = y|e)$ coincides with Proposition 1.

Proof. Let U_1 be the set (tuple) of background variables that influence Z and $U_0 = U \setminus U_1$. Then

$$p^Z(Y_{\text{do } X=x} = y|e) \tag{11}$$

$$= \sum_w p(y|\text{do } X = x, w)p(w|e) \tag{12}$$

$$= \sum_{w,u_0} p(y|\text{do } X = x, w, u_0)p(w|e)p(u_0|\text{do } X = x, w) \tag{13}$$

$$= \sum_{w,u_0} p(y|\text{do } X = x, w, u_0)p(w|e)p(u_0|w) \tag{14}$$

$$= \sum_{w,u_0} p(y|\text{do } X = x, w, u_0)p(w|e)p(u_0), \tag{15}$$

where Equation (14) is due to the fact that the distribution of U_0 is invariant and X does not influence W, so W can be written as the same function of U_0 in M_0 and $(M_0)_{\text{do } X=x}$; and Equation (15) is due to the fact that $W \subset Z$ and $Z \perp\!\!\!\perp U_0$ by definition of U_0.

On the other hand, we have

$$p(Y_{\text{do } X=x} = y|e) \tag{16}$$

$$= \sum_{u:p(u,e)>0} p(y|\text{do } X = x, u)p(u|e) \tag{17}$$

$$= \sum_{u_0,u_1:p(u_0,u_1,e)>0} p(y|\text{do } X = x, u_0, u_1)p(u_0, u_1|e) \tag{18}$$

$$= \sum_{u_0,u_1,w:p(u_0,u_1,e)>0} p(y, w|\text{do } X = x, u_0, u_1)p(u_0, u_1|e) \tag{19}$$

$$= \sum_{u_0,u_1,w:p(u_0,u_1,e),p(u_0,u_1,w)>0} p(y, w|\text{do } X = x, u_0, u_1)p(u_0, u_1|e) \tag{20}$$

$$= \sum_{u_0,u_1,w:p(u_0,u_1,e),p(u_0,u_1,w)>0} p(y|\text{do } X = x, u_0, u_1, w)p(w|\text{do } X = x, u_1, u_0)p(u_0, u_1|e) \tag{21}$$

$$= \sum_{u_0,u_1,w:p(u_0,u_1,e),p(u_0,u_1,w)>0} p(y|\text{do } X = x, u_0, w)p(w|\text{do } X = x, u_1, u_0)p(u_0, u_1|e) \tag{22}$$

$$= \sum_{u_0,u_1,w:p(u_0,u_1,e),p(u_0,u_1,w)>0} p(y|\text{do } X = x, u_0, w)p(w|u_1, u_0)p(u_0, u_1|e) \tag{23}$$

$$= \sum_{u_0,u_1,w:p(u_0,u_1,e),p(u_0,u_1,w)>0} p(y|\text{do } X = x, u_0, w)p(w|u_0, u_1, e)p(u_0, u_1|e) \tag{24}$$

$$= \sum_{u_0,u_1,w:p(u_0,u_1,e),p(u_0,u_1,w)>0} p(y|\text{do } X = x, u_0, w)p(w, u_0, u_1|e) \tag{25}$$

$$= \sum_{u_0,u_1,w:p(u_0,u_1,e,w)>0} p(y|\text{do } X = x, u_0, w)p(w, u_0, u_1|e) \tag{26}$$

$$= \sum_{u_0,w:p(u_0,e,w)>0} p(y|\text{do } X = x, u_0, w) \sum_{u_1:p(u_0,u_1,e,w)>0} p(w, u_0, u_1|e) \tag{27}$$

$$= \sum_{u_0,w:p(u_0,e,w)>0} p(y|\text{do } X = x, u_0, w)p(w, u_0|e) \tag{28}$$

$$= \sum_{u_0,w} p(y|\text{do } X = x, u_0, w)p(w, u_0|e), \tag{29}$$

where Equation (22) is due to Markovianity and $(Y \perp\!\!\!\perp An(Z)|Z)_M$, which implies $(Y \perp\!\!\!\perp U_1|Z)_M$, and thus $(Y \perp\!\!\!\perp U_1|W)_{M_{\text{do } X=x}}$, Equation (23) follows from the fact that X does not influence W, Equation (24) follows from the fact that U_1 already determines W.

Note that $p(w|e)p(u_0) = 0$ implies $p(w, u_0|e) = 0$ and therefore

$$\mathrm{D}[p(Y_{\mathrm{do}\,X=x}|E)\|p^W(Y_{\mathrm{do}\,X=x}|D)]$$

is defined.

Now we can calculate

$$\mathrm{D}[p(Y_{\mathrm{do}\,X=x}|E)\|p^Z(Y_{\mathrm{do}\,X=x}|E)] \tag{30}$$

$$= \sum_e p(e)\mathrm{D}[p(Y_{\mathrm{do}\,X=x}|e)\|p^Z(Y_{\mathrm{do}\,X=x}|e)] \tag{31}$$

$$\leq \sum_e p(e)\mathrm{D}[p(W, U_0|e)\|p(W|e)p(U_0)] \tag{32}$$

$$= \sum_{e,w,u_0} p(w, u_0, e) \log \frac{p(w, u_0|e)}{p(w|e)p(u_0)} \tag{33}$$

$$= \sum_{e,w,u_0} p(w, u_0, e) \log \frac{p(w, u_0, e)}{p(w, e)p(u_0)} \tag{34}$$

$$= \mathrm{I}(W, E : U_0) \tag{35}$$
$$\leq \mathrm{I}(Z, E : U_0) \tag{36}$$
$$= \mathrm{I}(Z : U_0) + \mathrm{I}(E : U_0|Z) \tag{37}$$
$$= 0 + \mathrm{H}(E|Z) - \mathrm{H}(E|U_0, Z), \tag{38}$$

where inequality (32) follows from the monotonicity (which follows from the chain rule) of the Kullback-Leibler divergence [9] together with equations (29) and (15), Equation (37) is the chain rule for mutual information, and $\mathrm{I}(W : U_0) = 0$ is due to U_0 not influencing W and Markovianity.

$\square$

Note that, if we chose the set Z in the above proposition such that it is as "close" (in the causal diagram) to Y as possible, this could yield better approximations $p^Z(Y_{\mathrm{do}\,X=x} = y|e)$ than simply letting Z be the root nodes, as done in $\bar{p}(Y_{\mathrm{do}\,X=x} = y|e)$. We leave this as a question for future work.

A.2 Proof of Proposition 2

Here we give a proof for Proposition 2.

Proof. We calculate

$$\mathrm{D}(p(Z)\|\bar{p}(Z)) \tag{39}$$
$$\leq \mathrm{D}(p(X_0, \ldots, X_K, C)\|p(C)\Pi_k p(X_k|C))$$
$$= \mathrm{D}(p(C)p(X_0|C)p(X_1|X_0, C)\cdots p(X_K|X_0, \ldots, X_{K-1}, C)\|p(X)\Pi_k p(X_k|C))$$
$$= \sum_{x_0,\ldots,x_K,c} p(x_0, \ldots, x_K, c) \log \frac{p(c)}{p(c)} \frac{p(x_0|c)}{p(x_0|c)} \frac{p(x_1|x_0, c)}{p(x_1|c)} \frac{p(x_2|x_0, x_1, c)}{p(x_2|c)} \cdots$$
$$\cdot \frac{p(x_K|x_0, \ldots, x_{K-1}, c)}{p(x_K|c)}$$
$$= \sum_{x_0,\ldots,x_K,c} p(x_0, \ldots, x_K, c) \log \frac{p(c)}{p(c)} \frac{p(x_0|c)}{p(x_0|c)} \frac{p(x_1, x_0|c)}{p(x_1|c)p(x_0|c)} \frac{p(x_2, x_0, x_1|c)}{p(x_2|c)p(x_0, x_1|c)} \cdots$$
$$\cdot \frac{p(x_K, x_0, \ldots, x_{K-1}|c)}{p(x_K|c)p(x_0, \ldots, x_{K-1}|c)}$$
$$= \mathrm{I}(X_1 : X_0|C) + \mathrm{I}(X_2 : X_0, X_1|C) + \ldots + \mathrm{I}(X_K : X_0, \ldots, X_{K-1}|C)$$
$$\leq \mathrm{H}(X_1|C) + \mathrm{H}(X_2|C) + \ldots + \mathrm{H}(X_K|C),$$

$$\tag{40}$$

where inequality (39) follows from the monotonicity (which follows from the chain rule) of the Kullback-Leibler divergence [9]. □

References

[1] S. Angel, H. Ballani, T. Karagiannis, G. O'Shea, and E. Thereska. End-to-end performance isolation through virtual datacenters. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pages 233–248, 2014.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[3] A. Balke and J. Pearl. Counterfactual probabilities: Computational methods, bounds and applications. In *Proceedings of the Tenth international conference on Uncertainty in artificial intelligence*, pages 46–54. Morgan Kaufmann Publishers Inc., 1994.

[4] E. Bareinboim and J. Pearl. Transportability of causal effects: Completeness results. In *Proceedings of the 26th National Conference on Artificial Intelligence (AAAI)*, pages 698–704. AAAI Press, Menlo Park, CA., 2012.

[5] L. Bottou, J. Peters, J. Quinonero-Candela, D. X. Charles, D. M. Chickering, E. Portugaly, D. Ray, P. Simard, and E. Snelson. Counterfactual reasoning and learning systems: The example of computational advertising. *The Journal of Machine Learning Research*, 14(1):3207–3260, 2013.

[6] L. Carata. *Provenance-based computing.* PhD dissertation, University of Cambridge, 2016.

[7] L. Carata, O. Chick, J. Snee, R. Sohan, A. Rice, and A. Hopper. Resourceful: fine-grained resource accounting for explaining service variability. Technical report, University of Cambridge, Computer Laboratory, 2014.

[8] R. C. Chiang, J. Hwang, H. H. Huang, and T. Wood. Matrix: Achieving predictable virtual machine performance in the clouds. In *11th International Conference on Autonomic Computing (ICAC 14)*, pages 45–56, 2014.

[9] T. Cover and J. Thomas. *Elements of Information Theory.* Wileys Series in Telecommunications, New York, 1991.

[10] J. Lemeire, E. Dirkx, and F. Verbist. Causal analysis for performance modeling of computer programs. *Scientific Programming*, 15(3):121–136, 2007.

[11] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*, pages 94–103. IEEE, 2007.

[12] K. Ostrowski, G. Mann, and M. Sandler. Diagnosing latency in multi-tier black-box services. In *5th Workshop on Large Scale Distributed Systems and Middleware (LADIS 2011)*, volume 3, page 14, 2011.

[13] P. Padala, K.-Y. Hou, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, and A. Merchant. Automated control of multiple virtualized resources. In *Proceedings of the 4th ACM European conference on Computer systems*, pages 13–26. ACM, 2009.

[14] J. Pearl. *Causality.* Cambridge University Press, 2000.

[15] J. Pearl. Causal inference in statistics: An overview. *Statistics Surveys*, 3:96–146, 2009.

[16] J. Pearl and E. Bareinboim. Transportability of causal and statistical relations: A formal approach. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pages 540–547. IEEE, 2011.

[17] J. Peters, D. Janzing, and B. Schölkopf. *Elements of causal inference: foundations and learning algorithms.* MIT Press, 2017.

[18] Y. Shoham and K. Leyton-Brown. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations.* Cambridge University Press, 2008.

[19] J. Snee, L. Carata, O. R. Chick, R. Sohan, R. M. Faragher, A. Rice, and A. Hopper. Soroban: attributing latency in virtualized environments. In *Proceedings of the 7th USENIX Conference on Hot Topics in Cloud Computing*, pages 11–11. USENIX Association, 2015.

[20] P. Spirtes, C. Glymour, and R. Scheines. *Causation, prediction, and search.* MIT, Cambridge, MA, 2nd edition, 2000.

[21] W. Zheng, R. Bianchini, G. J. Janakiraman, J. R. Santos, and Y. Turner. Justrunit: Experiment-based management of virtualized data centers. In *Proc. USENIX Annual technical conference*, 2009.