A Demonstration of Interactive Analysis of Performance Measurements with Viska

Helga Gudmundsdottir, Babak Salimi, Magdalena Balazinska, Dan R. K. Ports, Dan Suciu
University of Washington
helgag, bsalimi, magda, drkp, suciu@cs.washington.edu

ABSTRACT

The ultimate goal of system performance analysis is to identify the *underlying causes* for performance differences between different systems and different workloads. We make this goal easier to achieve with Viska, a new tool for generating and interpreting performance measurement results. Viska leverages cutting-edge techniques from big data analytics and data visualization to aid and automate this analysis, and helps users derive meaningful and statistically sound conclusions using state-of-the-art causal inference and hypothesis testing techniques.

Keywords

performance debugging, causal inference, data analytics

1. INTRODUCTION

Much of systems research consists of performance analysis – to learn when one system outperforms another, to identify architectural choices responsible for the difference, and to identify performance anomalies in particular workloads. Ultimately, researchers and developers want to quickly uncover the underlying *causes* that contribute to these factors. Common types of questions include comparing performance across systems (Why is one system faster than another for a particular workload?), understanding the performance of distributed systems (With multiple components and frequent releases, what change caused a sudden increase in latency?), and performance tuning (Why is a particular database system configuration affecting runtime?).

These questions are challenging because systems evolve over time and have hundreds of configuration knobs for tuning. Performance is also often closely tied to the underlying platform and available resources. As a result, performance analysis today requires deep expertise in each system and its codebase, and detailed experimentation, instrumentation, and profiling.

The Viska project aims to make performance analysis easier with a new tool for generating and interpreting performance measurement results, helping users derive meaningful and statistically sound conclusions. Viska collects a variety of features that describe the execution of a query, including query execution time, attributes of the input data (e.g. input data sizes and other statistics), attributes of the executed queries (e.g., operators in the queries), and system configuration parameters (e.g., available CPU, memory, and disk IO resources). Each query execution profile takes the form of a feature vector. Viska allows users to explore these feature vectors to identify correlations and isolate potential causes.

SIGMOD'17, May 14-19, 2017, Chicago, IL, USA

DOI: http://dx.doi.org/10.1145/3035918.3056448

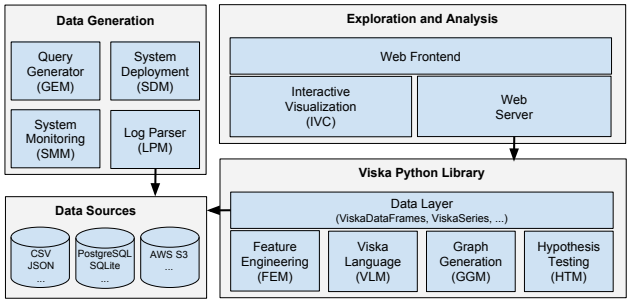

Figure 1: Viska System Architecture

Viska is specifically designed to address the unique challenges of performance data, namely:

- High dimensionality. The data consists of a large number of runs of experiments with different workloads, configuration settings, and deployments. Viska focuses investigative efforts by automatically identifying important features from this large set of variables.
- Unclear relationships between variables. Variables in this dataset, are rarely conditionally independent. In fact, many are highly correlated, leading users to draw spurious conclusions about cause and effect. Viska is able to verify that conclusions are statistically sound.
- Evolving research questions. Viska supports interactive data exploration, allowing the user to re-evaluate and re-iterate different hypotheses easily and effectively.

At the heart of the Viska system is a Hypothesis Testing Module (HTM) that enables users to ask questions about the *causal relationship* between selected features and system performance. Viska's HTM is based on causal inference following the Neyman-Rubin model [13]. The HTM answers causality questions based on the set of query execution profiles and a causal graph produced by Viska's Graph Generation Module (GGM), that captures conditional dependence and independence information between variables.

While performance analysis with Viska culminates in hypothesis testing with the HTM, the user must first be able to identify and refine a hypothesis. Viska supports the end-to-end task of performance debugging, including data generation and visualization. The complete system architecture is shown in Figure 1. As the figure shows, Viska provides an initial module that facilitates the generation, collection, and management of performance data for analysis. Viska further includes a rich and configurable set of visualizations that enable users to visually explore the data and identify potential root causes to investigate. Third, Viska enables users to engage in feature engineering and produce new features from their initial set of features. Finally, the data is ready for analysis with the HTM.

In this demonstration, the attendees will go through the experience of debugging specific performance questions in two recent versions of PostgreSQL. Attendees will go through the three key steps of the debugging process: (1) data visualization and identification of potential root causes of a suspicious performance behavior; (2) causal inference to determine if a potential cause is an actual cause; and (3) feature engineering to refine the analysis process.

Overall, this demonstration makes three contributions:

- It presents a new approach for performance debugging in database systems based on causal inference. It shows both the theoretical aspects of the approach and its implementation in the Viska system.
- It demonstrates the components necessary for end-to-end performance debugging including data visualization, feature engineering, data collection and generation, and ultimately hypothesis testing. It shows how these components interact during the performance debugging process.
- It enables conference attendees to experience the challenges of performance debugging on real data from recent versions of PostgreSQL and how the Viska system effectively supports that task.

2. THE VISKA SYSTEM

The Viska system automates and guides the three core tasks of performance analysis: generating, exploring and analyzing performance data. In this section, we present an overview of Viska, and how its various components (shown in Figure 1) address each of these tasks. As a running example, we will compare the performance of two recent versions of PostgreSQL and investigate the causes of the difference. While this is an intentionally simple example to shorten exposition, it already presents a number of challenges that are not easy to address without Viska's key features.

2.1 Generating performance data

Viska analyzes performance data obtained by benchmarking systems. This data consists of a set of profiles each representing a single execution of an experiment. A profile has a set of *input* variables that vary between experiments, such as the types of queries run, data set used, or configuration parameters. Viska records these along with a set of *observed* variables: properties of the execution (e.g., number of rows processed), system-level metrics (e.g., CPU and disk utilization), and outcome variables (e.g., query execution time). Table 1 gives a simple example of what a dataset might look like, showing the first four records. We do not describe the modules for workload generation, system deployment, and metrics gathering here.

Version	*TableSize*	*OutputRows*	*CPU*	*DiskIO*	*Runtime*
9.4.5	2 GB	800665	77.44	890.00	2.4
9.4.5	8 GB	1	88.89	650.20	16.3
9.6.0	8 GB	47989007	91.21	1420.67	17.0
9.6.0	8 GB	39838415	89.72	1337.10	16.1

Table 1: Example dataset

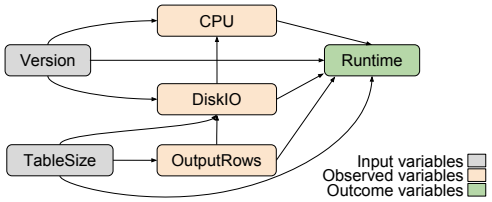

Figure 2: Example causal graph generated from the dataset described in Table 1 using the GGM.

2.2 Exploring performance data

Recent innovations in visual data exploration tools are able to help users make sense of high dimensional datasets [17, 5, 16]. Drawing inspiration from this line of work, we incorporate various visualization techniques into Viska's *Interactive Visualization Component* (IVC), which provides an engaging way for users to explore performance results. In contrast to general-purposes visualization tools like Tableau, our choices of visual encodings and charts shown are customized to support a performance analysis. This workflow includes selecting subpopulations (e.g. filtering long running queries); creating new features derived from others (e.g. binning or merging variables); creating and interpreting a causal model (described in Section 2.3); and generating and exploring a causal dependency graph (see example in Figure 2).

The IVC shows coordinated multi-views with distributions of important variables, which the user can interact with to discover outliers, trends and correlations. Figure 3 displays a simplified version of our example dataset, showing the distribution of input and outcome variables. The user is able to brush the group of longer running queries and see immediately the distribution of input variables for this selection. Here, most long-running queries executed on a large input table, but a few outliers ran on a smaller table. By further brushing to select those outliers in the `TableSize` chart, the user can see that these queries were all executed on the new version of the database. The user can quickly investigate and form hypotheses in this manner.

While visual exploration tools are useful in generating hypotheses, they have been criticized for encouraging users to draw conclusions that may not be statistically significant [3]. Viska emphasizes measures of bias and statistical significance at each step of the workflow to address this problem.

2.3 Analyzing Performance Data

The Hypothesis Testing Module (HTM) enables Viska to generate statistically sound explanations for observed relationships between various features of a system, such as workload parameters, system metrics, and performance results. It is built upon a long tradition of work in discovering cause-effect dependencies. In particular, the HTM is based on the Neyman-Rubin Causal Model [13]. In this model, the objective is to quantify the causal effect of a binary *treatment* on an *outcome* given a vector of potentially confounding effects called *covariates*.

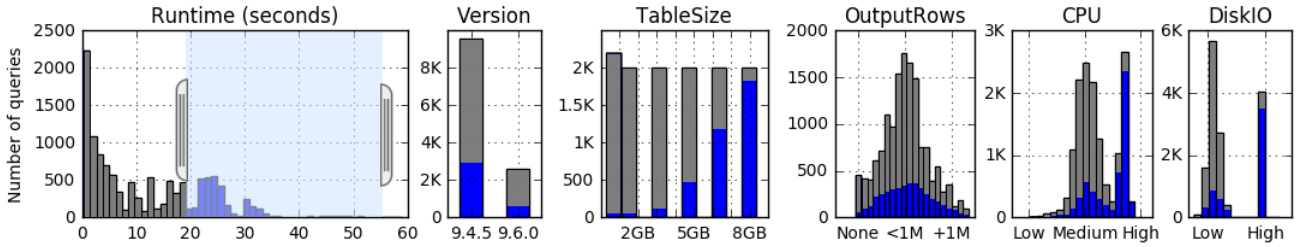

Figure 3: Example interactive visualization of important variables. Blue indicates a filtered sub-population of queries.

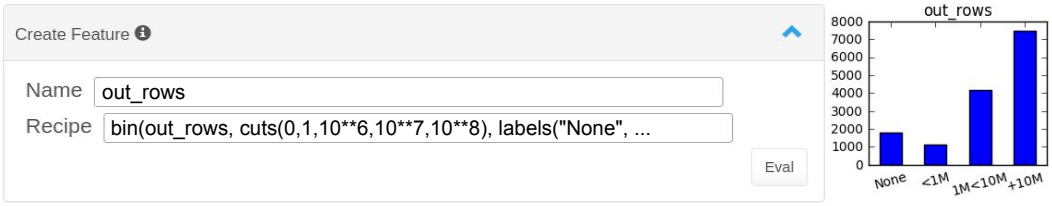

Figure 4: Screenshot of a user creating a new feature by custom binning, using Viska's expression language.

To investigate the performance difference between two versions of PostgreSQL, we might construct our model as follows: We select `Version` as our treatment (after making it binary). We refer to samples from the new version as *treated*, and others as the *control*. We choose `Runtime` as the outcome variable, and select the four other potentially confounding variables in the table as covariates. The goal of the analysis is to compute the *average treatment effect*:

$$\text{ATE} = \mathbb{E}[\texttt{Runtime}|\texttt{Version} = \text{new}] - \mathbb{E}[\texttt{Runtime}|\texttt{Version} \neq \text{new}]$$

over the possible values of the covariates. The key challenge in this computation is that there are unlikely to be samples representing both the treated and control groups for every unique set of covariate values. For example, the new version of PostgreSQL may achieve higher disk throughput, leading to values of `DiskIO` not obtainable for the older version.

To draw statistically valid conclusions in spite of this challenge, Viska's HTM uses techniques such as *matching* and *subclassification* which aim to prune data in such a way that the treated and control groups have similar covariate values [9, 12]. Intuitively, the result of this pruning simulates a controlled experiment. To obtain statistically efficient estimators for ATE, HTM follows the suggestions of [7], and uses causal graphs for covariate selection. The *Graph Generation Module* (GGM) of Viska is used to infer causal graphs, such as the one seen in Figure 2, by checking conditional independence between variables. We do not describe the techniques used for graph discovery here.

Finally, the HTM uses one of several statistical tests, such as the Z-test, to compute the statistical significance of ATE. To avoid the challenge of multiple hypothesis testing [15], where excessive analysis of a single dataset risks finding artificial patterns, Viska takes two approaches. First, it keeps track of the number of hypotheses that a user has tested and adjusts statistical significance accordingly. Second, it enables a user to run new sets of experiments to generate fresh data and continue an investigation.

3. DEMONSTRATION DETAILS

Our demonstration will allow attendees to explore several performance questions using Viska. Here, we focus on one question: *Is the newest version of PostgreSQL faster at read queries than the previous one, and, if so, why?* We run a workload of simple generated queries on a large table, recording the runtime for each query along with the workload parameters, database configuration, query plans, and system-level metrics.

Defining the Population: During the first phase of the demonstration, the user is shown an overview of the dataset. Along with basic statistics, the first screen presents the user with coordinated views of the distribution of both outcome and input variables, similar to Figure 3. The user can use brushing to define the population of interest. The tool immediately reflects changes in the outcome variables as different subpopulations are selected. This allows the user to quickly gain intuition into which variables are important and form initial hypotheses as to potential causes on performance.

Feature Engineering: It is often the case that the set of initial features is not sufficient for analysis. The user can compute new features as a function of existing ones by writing *recipes* in Viska's expression language. One reason to do this is that a causal hypothesis needs to be specified as a binary variable. Thus, users will write a recipe to convert the feature representing the system version into a new binary variable (i.e. new vs. old). As another example, recipes can serve to transform continuous variables into discrete ones using binning, as shown in Figure 4. Viska also includes techniques to merge features and thereby reducing dimensionality.

Causal Analysis: Now, the user is presented with the causal analysis screen. The user is prompted to select the treatment variable – the possible cause – from the list of all binary features. In this case, the user selects the newly created `new_version` feature. By default, the outcome variable is the query runtime (though the user can also investigate the causal effect of database version on other variables). Initially, the user selects no covariates.

Next, the user is presented with charts showing the distribution of the outcome variable and any selected covariate, color coded to clearly distinguish between the treated and control groups. The resulting ATE statistics are displayed, along with several other statistical metrics (such as imbalance measures). The user will see a negligible average treatment effect (0.08 seconds), indicating that the runtime is largely unaffected by the version.

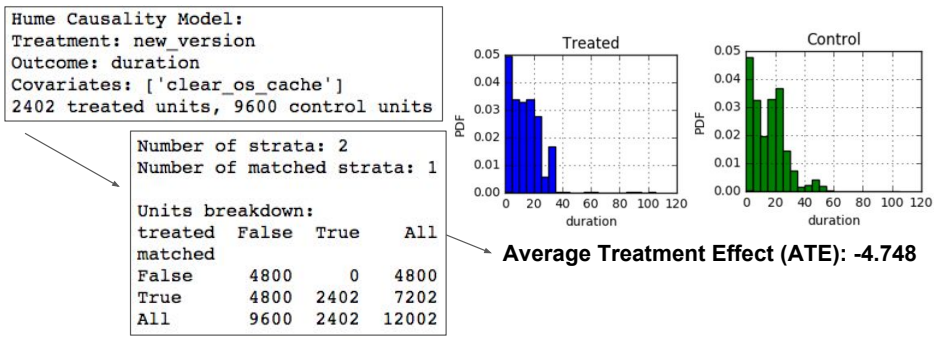

Figure 5: Example workflow for causal inference, starting with defining the model parameters, then running coarsened exact matching, and finally interpreting results.

Using Viska's visualization component, the user will discover that the variable `clear_os_cache` is imbalanced across the treated and control groups, i.e. few measurements were taken for the new version with a warm OS cache. Suspecting that this could have confounding effects, the user adds this variable as a covariate, and re-runs the causal analysis. The results show that the new version is in fact significantly faster after properly adjusting for `clear_os_cache`. This workflow is shown in Figure 5.

Having seen that the new version of PostgreSQL is faster than the old, the user can now explore different hypotheses as to *why* that may be, using different treatments or covariates. One hypothesis may be that disk read throughput is higher and the user can indeed verify a 25% increase in the average treatment effect when that is considered. Considering CPU utilization shows an increase of 250%, reflecting the multi-core support introduced in the new version.

4. RELATED WORK

Work on causality in database systems [11] uses provenance to explain query answers and non-answers. Provenance has also successfully been used for root-cause analysis in networking applications [4]. These techniques, however, assume that variables can be connected to their causes through logical expressions (i.e., query operations). In system performance debugging, this assumption does not hold.

Taint tracking can help identify configuration variables responsible for performance regressions [1]. Such an approach could be integrated with Viska as an additional feature.

In prior work [10], we developed a system to answer comparative performance questions for MapReduce jobs. That approach, however, did not perform causal inference when generating explanations. Other prior systems that compare measurements to diagnose performance regressions [14] require only one variable to be changed at a time because they do not perform causal analysis. DBSherlock [18], unlike Viska, requires that users specify domain-knowledge rules or indicate the true causes of anomalous events to build a causal model. Additionally, DBSherlock focuses on explaining anomalous events while Viska enables more general reasoning about the effect of different variables on performance.

Many techniques exist for predicting query performance and automatically tuning database systems [8, 6, 2]. These approaches build models of query runtime in order to find configurations that yield high performance, rather than pinpoint root causes for specific performance behaviors.

5. CONCLUSION

We demonstrate Viska, a system for automating and guiding the generation, analysis, and exploration of performance data. Viska builds on techniques from big data analytics, data visualization, and causal inference.

Acknowledgements: We thank Helgi Sigurbjarnarson and the anonymous reviewers for their feedback. This work is supported in part by the National Science Foundation through NSF grants IIS-1247469, AITF 1535565, IIS-1524535, IIS-1614738, and CNS-1615102, and gifts from the Big Data ISTC and Facebook. Helga Gudmundsdottir is also supported by a Microsoft Research Women's Fellowship.

6. REFERENCES

[1] M. Attariyan et al. X-ray: Automating root-cause diagnosis of performance anomalies in production software. In *OSDI '12*, 2012.

[2] P. Barham et al. Using Magpie for request extraction and workload modelling. In *OSDI '04*, 2004.

[3] C. Binning et al. Towards sustainable insights: or why polygamy is bad for you. In *CIDR'17*, 2017.

[4] A. Chen et al. Data provenance at internet scale: Architecture, experiences, and the road ahead. In *CIDR'17*, 2017.

[5] F. Chirigati et al. Data polygamy: the many-many relationships among urban spatio-temporal data sets. In *SIGMOD'16*, pages 1–15, 2016.

[6] I. Cohen et al. Correlating instrumentation data to system states: A building block for automated diagnosis and control. In *OSDI '04*, 2004.

[7] X. De Luna et al. Covariate selection for the nonparametric estimation of an average treatment effect. *Biometrika*, 2011.

[8] H. Herodotou and S. Babu. A what-if engine for cost-based MapReduce optimization. *IEEE Data Eng. Bull.*, 36(1):5–14, 2013.

[9] S. M. Iacus et al. Cem: software for coarsened exact matching. *Journal of Statistical Software*, 30(9):1–27, 2009.

[10] N. Khoussainova, M. Balazinska, and D. Suciu. Perfxplain: Debugging mapreduce job performance. *PVLDB*, 5(7):598–609, 2012.

[11] A. Meliou et al. WHY so? or WHY no? functional causality for explaining query answers. In *MUD in conjunction with VLDB*, pages 3–17, 2010.

[12] P. R. Rosenbaum and D. B. Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):pp. 41–55, 1983.

[13] D. B. Rubin. *The Use of Matched Sampling and Regression Adjustment in Observational Studies*. Ph.D. Thesis, Department of Statistics, Harvard University, 1970.

[14] R. R. Sambasivan et al. Diagnosing performance changes by comparing request flows. In *NSDI'11*, 2011.

[15] J. P. Shaffer. Multiple hypothesis testing. *Review of psychology*, 46:561, 1995.

[16] Tableau. http://www.tableau.com, 2003.

[17] M. Vartak et al. Seedb: Automatically generating query visualizations. *PVLDB*, 7(13):1581–1584, 2014.

[18] D. Y. Yoon, N. Niu, and B. Mozafari. Dbsherlock: A performance diagnostic tool for transactional databases. In *SIGMOD'16*, pages 1599–1614, 2016.