DejaVu: Recycling Tuning Setups in Hive Query Compilation

Edson Ramiro Lucas Filho[1], Eduardo Cunha de Almeida[1], and Stefanie Scherzinger[2]

[1] Universidade Federal do Paraná, Brazil
{erlfilho,eduardo}@inf.ufpr.br
[2] OTH Regensburg
stefanie.scherzinger@oth-regensburg.de

Abstract. SQL-on-Hadoop processing engines have become state-of-the art, yet the skills required to tune these systems are rare in the job market. Automated tuning advisers can profile the low-level MapReduce jobs and propose appropriate tuning setups, but up-front tuning is time consuming and costly. In this demo, we present *DejaVu*. *DejaVu* integrates with Hive and effectively reduces the tuning costs by caching tuning setups for *partial query plans*: When the SQL-on-Hadoop engine Hive compiles SQL queries into physical query plans, single MapReduce jobs tend to be similar between query plans. By recycling the tuning setups for *similar* low-level MapReduce jobs, *DejaVu* can effectively cut down the time spent profiling the TPC-H query workload in half, achieving similar impact on the performance of the jobs. While we employ Starfish in this demo, *DejaVu* can leverage any third-party MapReduce tuning adviser.

1 Introduction

More than a decade after the publication of the seminal MapReduce paper [1], we can observe a clear preference among Hadoop or Spark users for higher-level query languages [5]. SQL-on-Hadoop systems compile SQL queries into query plans of MapReduce jobs. Naturally, this greatly improves the productivity of data scientists. Yet compiling queries to query plans, and then scheduling its jobs in a cluster is only half the battle: The underlying MapReduce framework needs to be tuned for performance.

The expertise required for allocating the right mix of physical resources and for twiddling with the tuning knobs is rare. Also, Hive [7] currently implements almost a 1000 tuning parameters, so manual tuning is quite out of the question.

Automated tuning advisers for MapReduce frameworks rely on profiling MapReduce jobs [6, 2], at the cost of imposing a runtime overhead. For instance, the Starfish tuning adviser can cause an overhead of up to 50% [6] during profiling. Thus, when the query workload is highly dynamic, re-tuning is costly in cloud-based environments.

In this demo, we present *DejaVu*, a tool that integrates with the SQL-on-Hadoop engine Hive. Rather than contributing yet another tuning adviser, *DejaVu* employs third-party tuning advisers like Starfish, but caches tuning setups for partial query plans to avoid unnecessary job profiling. Our approach relies on two observations regarding the query plans compiled from SQL queries: (1) The individual MapReduce jobs within a query plan often have different resource requirements. (2) Since the jobs are generated, MapReduce jobs tend to be similar *across* query plans.

Regarding observation (1), we conclude that each job benefits from having a custom-tailored *tuning setup*, i.e., the configuration of its tuning parameters. In contrast, uniformly applying a tuning setup to all jobs requires maximum settings, and can be wasteful. Consequently, we pursue *non-uniform* tuning and assign one tuning setup per job.

Let us now consider observation (2). We regard two MapReduce jobs as similar from the perspective of tuning, if they have the same *code signature*. Intuitively, the code signature of a MapReduce job captures the SQL operators implemented by this job, as well as the expected size of the input. This information is available through the Hive query compiler. Our hypothesis (which we can confirm in our experiments) is that jobs that share the same code signature benefit from the same tuning setups. We therefore may *recycle* tuning setups for similar jobs, to reduce profiling time.

Let us consider the TPC-H queries compiled on Hive-0.13, which yields 106 MapReduce jobs[3]. For 71% of these jobs, there is at least one other job with the same code signature. Fewer than a third of the jobs have a unique code signature.

This inspires the idea of recycling tuning setups for similar jobs. Over time, we may even be able to assign tuning setups for ad-hoc queries, which we have not encountered yet. In fact, ad-hoc queries are prevalent in many query workloads [8], yet tuning advisers for MapReduce frameworks rely on profiling the complete workload up front. Thus, our approach can be used in environments where traditionally, tuning advisers fail.

As we show shortly, by non-uniform tuning and by recycling tuning setups, we can effectively reduce profiling costs. Note that earlier, we have reported on a predecessor tool called *Chameleon* [4]. Like *DejaVu*, *Chameleon* performs non-uniform tuning. Yet in *Chameleon*, tuning setups are only assigned manually to MapReduce jobs. In contrast, *DejaVu* delegates job tuning to third-party tuning advisers and is fully automated. In [3], our full paper presents in greater detail the caching mechanism of DejaVu and the experimental results.

2 Caching Code Signatures

We next introduce code signatures, and the code signature cache as the datastructure at the heart of *DejaVu*.

The SQL-on-Hadoop engine Hive compiles SQL queries into query plans. The low-level MapReduce jobs are annotated by Hive, and annotations are accessible via the Hive Java API. Each job carries a list of the physical query operators that are implemented by this job. Further, for each operator, an estimated input *cardinality* is given.

Example 1. TCP-H query 1 is compiled by Hive 0.13 to a sequence of two MapReduce jobs. The first job is annotated with the operators Filter, Select, and GroupBy, each with cardinality 2. The job is further annotated with the operators TableScan, ReduceSink, and FileSink, each with a cardinality of 1. In our setting (c.f. Section 3), the estimated input size is 7.24GB. □

We hypothesize that jobs with the same annotations have similar resource requirements. Therefore, they may be executed with the same tuning setups, even though their Java code differs. In our experiments, as discussed shortly, we are able to confirm this.

[3] Here, we ignore approx. 50 auxiliary jobs that are executed only locally, rather than as a MapReduce job, so they do not require a MapReduce tuning setup.

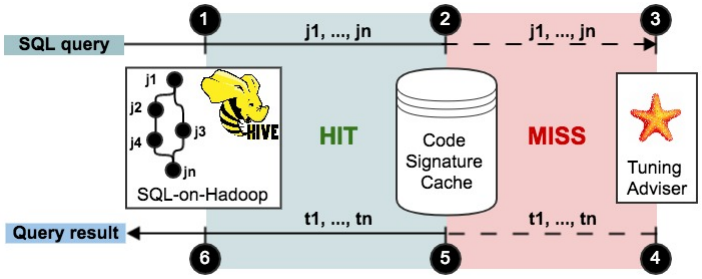


Fig. 1: The *DejaVu* code signature cache. Jobs j_i are mapped to tuning setups t_i, using the code signature as cache key.

We capture the annotations by the *code signature*, a tuple stating the input size (given in order of magnitude), and the annotated query operators with their cardinalities.

Example 2. We continue with TPC-H query 1. The code signature of the first job is $\langle 9, \textit{FileSink} : 1, \textit{Filter} : 1, \textit{GroupBy} : 2, \textit{ReduceSink} : 1, \textit{Select} : 2, \textit{TableScan} : 1 \rangle$ □

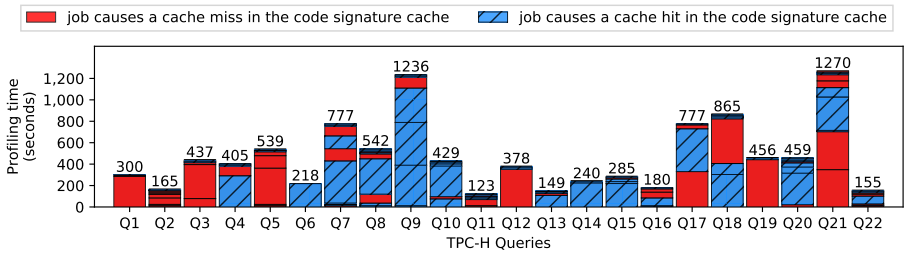
Figure 1 visualizes the code signature cache in *DejaVu*. Initially, the cache is empty. A SQL query is compiled by Hive into the query plan (Step 1). For each of the jobs $j_1, \ldots, j_n$ in this query plan, we look up the tuning setup in the code signature cache (Step 2), using the code signature as key. For each cache miss, we employ the Starfish tuning adviser for profiling the job and generating a tuning setup (Step 3). The tuning setups, denoted $t_1, \ldots, t_n$, are stored in the code signature cache, with the code signatures of the jobs as look-up keys (Step 4). As the cache becomes populated, we observe more cache hits (Steps 5 and 6). In the best case, we have cache hits for all jobs in the query plan. Then, we can simply recycle the tuning setups of similar jobs, and need not turn to Starfish for profiling at all.

3 Profiling TPC-H Queries

We have implemented the code signature cache in Java, and integrated it with Apache Hive 0.13. We leverage tuning setups generated by the Starfish tuning adviser [6]. We run Starfish with sampling turned off (i.e., profiling all MapReduce jobs completely), to obtain high-quality tuning profiles. Using Starfish 0.3.0[4], we are tied to Hadoop 0.20.2. We evaluate the TPC-H queries rewritten in HiveQL[5]. The data has been generated with a scale factor of 10, which amounts to 10.46GB of data stored on disk.

Our experiments were executed in a cluster with three physical machines. We isolate the master node on one machine, so that it does not influence the profiling of worker jobs. In particular, each machine has a Intel(R) Core(TM) i3-3240 CPU @ 3.40GHz, 4GB of RAM, 1TB of disk. All our profiling runs are configured with the out-of-the-box tuning setup that we refer to as "Hadoop Standard".

[4] https://www.cs.duke.edu/starfish/release.html, last accessed September 2019.
[5] https://issues.apache.org/jira/browse/HIVE-600



(a) Total time Starfish spends profiling (no sampling): 10,396.97 seconds.

Fig. 2: The code signature cache reduces the profiling time by over 50% for TPC-H.

Recurring code signatures. We have compiled the TPC-H queries on Hive 0.13. There is a considerable share of recurring code signatures, with 24 code signatures shared by 70% of the 106 MapReduce jobs.

We profile the 22 TPC-H queries in the order of the TPC-H benchmark specification. Figure 2 shows the profiling time per query. In total, over ten thousand seconds are spent on profiling. In Figure 2 We visually distinguish two groups of jobs:

1. Jobs which cause a cache miss in the code signature cache,
2. and jobs which cause a cache hit in the code signature cache.

We can observe that for the first TPC-H query, all jobs would cause a cache miss. Yet already for the second and third queries, we'd have cache hits, even though the savings are minor. With the code signature cache becoming more populated, we get more cache hits, and in some cases, some substantial savings in the profiling time. For instance, for queries Q6 and Q14, we can recycle all tuning setups from the cache. Thus, they require no profiling at all.

You may find more details about the caching mechanism and other results in our full paper [3].

4 Demo Outline

To make the internals of *DejaVu* transparent in our demo, we have built a web interface, as shown in Figure 3:

- In the upper left, we can choose among the TCP-H queries and inspect their code.
- Upon the push of a button, we have Hive compile a query into a query plan and submit it to be executed by Hadoop, as shown in the left.
- The queries will have their jobs profiled in the case of a cache miss. In the case of a cache hit, the job will receive new tuning setup (generated by Starfish).
- To the right, our interface visualizes the contents of the code signature cache. We can see how the cache is getting populated over time.
- Also, to the right we can see the rank of run times and observe the benefits of the tuning cache.

The web-based GUI is for demonstration purposes only. At this point, *DejaVu* is fully operational and integrated with Hive. Visitors will be able to interact with *DejaVu*.

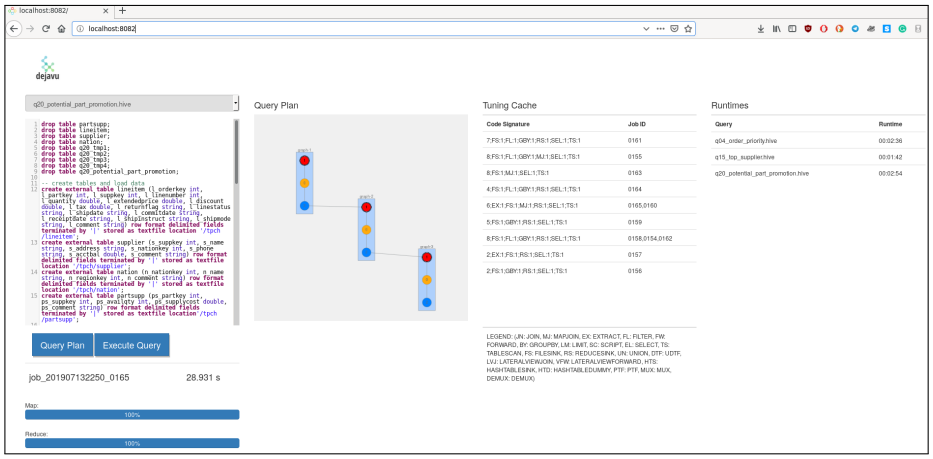


Fig. 3: The interactive *DejaVu* demo shows TPC-H queries, their query plans compiled by Hive, and the code signature cache getting populated, as TPC-H queries are compiled and profiled over time.

Acknowledgments We thank Herodotos Herodotou for all the support with Starfish. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

References

1. Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. In: OSDI (2004)
2. Duan, S., Thummala, V., Babu, S.: Tuning Database Configuration Parameters with iTuned. ReCALL **2**(1), 1246–1257 (aug 2009)
3. Filho, E.R.L., de Almeida, E.C., Scherzinger, S.: Don't Tune Twice: Reusing Tuning Setups for SQL-on-Hadoop Queries. In: ER 2019 – 38th International Conference on Conceptual Modeling (2019)
4. Filho, E.R.L., Picoli, I.L., de Almeida, E.C., Le Traon, Y.: Chameleon: The Performance Tuning Tool for MapReduce Query Processing Systems. In: 29th SBBD – Demos and Applications Session – ISSN 2316-5170 October 6-9, 2014 – Curitiba, PR, Brazil (2014)
5. Floratou, A., Minhas, U.F., Özcan, F.: SQL-on-Hadoop: full circle back to shared-nothing database architectures. Proceedings of the VLDB Endowment **7**(12), 1295–1306 (2014)
6. Herodotou, H., Lim, H., Luo, G., Borisov, N., Dong, L., Cetin, F.B., Babu, S.: Starfish: A Self-Tuning System for Big Data Analytics. In: CIDR (2011)
7. Thusoo, A., Sarma, J.S., Jain, N., Shao, Z., Chakka, P., Zhang, N., Antony, S., Liu, H., Murthy, R.: Hive - A petabyte scale data warehouse using hadoop. In: Proceedings - International Conference on Data Engineering. pp. 996–1005 (2010)
8. Yanpei Chen, S.A., Katz, R.H., Chen, Y., Alspaugh, S., Katz, R.: Interactive Query Processing in Big Data Systems: A Cross Industry Study of MapReduce Workloads. Tech. Rep. 12, University of California, Berkeley (aug 2012)