HybridPerfopticon: Query Visualization for Hybrid Distributed Databases

Brandon Haynes **Shrainik Jain**
University of Washington
{bhaynes, shrainik}@cs.washington.edu

ABSTRACT
Hybrid databases have recently gained a lot of attention because of the promise they show in several prominent use cases in scientific data management. A hybrid distributed database system usually consists of multiple underlying databases with differing data models. In order to better understand such systems we present a query profiling and visualization tool for queries authored over these hybrid systems. Our approach is to extend an existing visualization tool for distributed databases by adding support for multiple underlying database systems.

Author Keywords
Hybrid Databases, Query Visualization

ACM Classification Keywords
H.5.2 Graphical user interfaces (GUI): Miscellaneous

INTRODUCTION
Stonebraker *et al.* [9] recently observed that the "one size fits all" era is over, and that the demands of big data management warrant a more federated approach to database design. Accordingly, "hybrid databases" – generally dissimilar database management systems (DBMSs) combined in various ways – are quickly emerging as an active area in database research.

There exist many nascent examples of these hybrid DBMSs. For example, Lim *et al.* in Cyclops [7] offers a top-down approach that combines compatible systems into a monolithic whole. Polybase [5], on the other hand, elected a bottom-up approach where disparate systems were more loosely coupled.

To aid in understanding, development and debugging of these new systems, it would be advantageous to have visualization tools targeted at this class of DBMS. Unfortunately, most current profiling systems target queries in only a single database system. For example, SQL Server, Postgres, and MySQL all have a basic 'explain' functionality, which displays a query plan in a human-readable (but often arcane) format. Most 'explain'-type tools lack deeper analysis features such as the ability to view performance over the duration of the query. Further compounding this problem is the limited availability of tools available in a distributed database context, where operator-worker assignments, data skew, and similar problems are much more important.

Some tools, however, have been recently introduced that attempt to fill this need in a distributed database environment. Gprof [2] is a basic visualization tool for system development. However, it tends to be oriented toward developers. Dapper [1] is Google's production tracing tool designed to aid in distributed system development, but it lacks support for distributed query visualization. Finally, Ambrose [4] is a framework designed to visualize map-reduce jobs. While it offers many of the features that one would like to see in a robust distributed database visualization tool it currently targets only the map-reduce space and is not (in its current form) generalizable to a more broad set of DBMSs.

Compounding this is the fact that each of these tools lacks a complete visualization story where the tool is broadly useful across both users and system developers. This has resulted in a scarcity of debugging and visualization in the distributed database space (and as a consequence the hybrid database space).

To date, the Perfopticon [8] system is the only system that is broadly-usable by various audiences and also addresses the profiling requirements of distributed database systems. Perfopticon currently profiles queries on the Myria DBMS [6], and can be used to profile similar relational systems. However, while Perfopticon is able to be extended to other systems, it is not designed to *simultaneously* display the results of queries that cross DBMS boundaries.

Accordingly, in this paper we present an extension to Perfopticon (aptly named "HybridPerfopticon") designed to expose support for the emerging hybrid use-case, where users require robust profiling of plans that co-occur across multiple database systems. This class of queries present a number of unique challenges, such as how to display non-relational features (e.g., array chunks) in a framework that has heretofore targeted only tuple-based relational systems. Additionally, a number of corollary challenges arise including coordinating the display of multiple sub-plans, visualizing data transfer between systems, and gathering profiling data from multiple sources.

For the purpose of demonstration, we created a hybrid distributed database system using SciDB [3] and Myria [**?**] as its component systems. SciDB's array-oriented data model is quite dissimilar from Myria's, making it critical to identify equivalent semantics for communication between these two systems. HybridPerfopticon highlights this communication by helping to visualize the overall hybrid query plan, tuple (and array element) flow between operators, skew in distribution of workloads among the constituent systems, and internal query time statistics. It exposes these additional functions while maintaining complete compatibility with the existing underlying design.

In the next section, we present the architecture of the HybridPerfopticon system, which details the steps necessary to orchestrate hybrid plan execution and create Perfopticon-compatible hybrid plans for visualization purposes. This is followed by some results from the use case analysis. We conclude by looking at some future extensions to HybridPerfopticon.

HYBRIDPERFOPTICON ARCHITECTURE

Our approach to extending Perfopticon to support hybrid queries involved three major additions:

- Generating HybridPerfopticon-compatible SciDB plans
- Creating HybridPerfopticon by extending Perfopticon to recognize components from varying underlying DBMSs
- Creating hybrid plans and exposing them to HybridPerfopticon

In this section we describe each of these changes in detail and highlight the trade-offs in our design decisions.

Generating SciDB Plans & Metrics

Perfopticon was initially designed to profile plans in the Myria DBMS [8]. In the Myria DBMS, each operator is instrumented to store profiling data. When Perfopticon needs this profiling data, a HTTP request is generated, a query is performed over these data, and the aggregated results are returned.

Our first task was to expose similar functionality targeted at the SciDB DBMS. While we could take a similar route in SciDB by instrumenting each operator and responding to aggregation requests, this would be a time-consuming task and would decrease the performance of SciDB itself. Instead, we elected an alternative approach: since the relevant profiling metadata is already present in the SciDB logs, we generated our performance metrics by scraping these log files. While this resulted in decreased performance on startup[1], our approach has the added benefit of requiring no changes to the SciDB source code. We expect that this approach would allow for a low-barrier introduction of similar systems, assuming that the salient metrics already exist in the log files.

The architecture of our log-scraping process is illustrated in Figure 2. The entry point to this process is the `SciDBParser` class, which is responsible (via the `SciDBLog` class) for converting raw log entries into sequences of statements. Statements associated with a given query are aggregated into a physical plan, and the operators for that plan are identified and hierarchically assembled within that plan. During the log interpretation process we also identify performance metrics associated with each executed query and associate those results with the plan.

An illustration of the log-scraping process is listed in Figure 1. In this log fragment (abbreviated for readability), the `SciDBLogParser` identifies the beginning of the query execution process, identifies (and interprets) the associated physical plan, associates a scatter/gather operation with an underlying operator, and notes the time at which the query is committed. Each of these entries are converted into an equivalent statement, and the useful metadata for each statement is embedded therein.

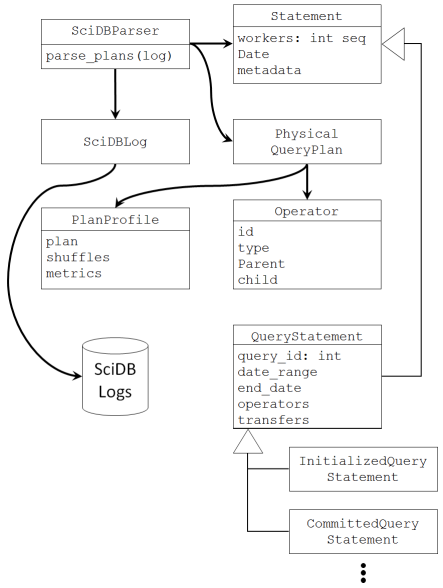

Figure 2. Architecture of the SciDB log parsing components.

Extending Perfopticon to Support Additional DBMSs

In order to extend Perfopticon to support queries across multiple DBMSs, we first explored the use of existing extension points. Since Perfopticon is not designed to be directly extendable (e.g., via a plugin-style architecture), extensive modification to the framework would be required to support the kinds of hybrid plans we intended to visualize. Instead, we elected an alternative approach where a substantial portion of the necessary logic was hidden behind the HTTP server (our "hybrid HTTP server", described in detail below) to which Perfopticon connects for plan and profiling metadata.

While this approach greatly reduced the modifications required to realize our desired functionality, some changes were nonetheless required to Perfopticon itself. For example, it is necessary to be able to identify the system of origin for each plan fragment; this is important not only for visualization purposes (e.g., to color fragments by system type) but

[1]Approximately ten seconds per 100MB of logging data.

SciDB Log Entries

```
2015-05-17 03:13:37,310: Initialized query (123)

2015-05-17 03:13:37,310: Parsing query(123): save_myria(redimension(...))

2015-05-17 03:13:37,365: The physical plan is detected as DML [pPlan]:
  >[pNode] impl_save agg 0 ddl 0 tile 0 children 1
    schema transform_1_1<value:int64 NULL DEFAULT null>
                        [id=0:599,256,0,bin=0:9,*,0]
    output full chunks: yes
    changes dstribution: no
    props sgm 1 sgo 1
    diout hash
    bound start {0, 0} end {599, 9} cells 6000 chunks 3 est_bytes 294006
  ...

2015-05-17 03:13:38,057: requestNextChunk: attId=0, stream=1,
                                           numSent=172, numRecvd=173


2015-05-17 03:13:37,836: Query (123) is being committed
```

HybridPerfopticon Internal Representation

```
InitializedQueryStatement(query_id=123)

ParsedQueryStatement(
        statement="save_myriad(redimension(...)")
PhysicalQueryPlanStatement(
        operators={
                Operator(
                        type="impl_save",
                        array="transform_1_1",
                        ...),
                ...})


ScatterGatherChunkStatement(
        array="transform_1_1",
        sent=172,
        received=173)

CommittedQueryStatement(123)
```

Figure 1. SciDB log fragment (abbreviated for readability) and mapped equivalents in the HybridPerfopticon HTTP server

also functionally. For example, the hybrid HTTP server must know from which system a fragment originated in order to retrieve correct profiling metrics.

A number of similar minor modifications were required throughout the Perfopticon framework. By way of example, SciDB fragments required a SciDB-specific query identifier be attached to each fragment; we envision that additional metadata would be required in order to support other systems. For example, in a graph-oriented DBMS it might be useful to associate a range of vertex identifiers operated on by a given operator.

Hybrid HTTP server

Given that HybridPerfopticon continues to expect HTTP responses in the same form as Perfopticon, our hybrid HTTP server needed to do a substantial amount of manipulation to compose hybrid plans without violating this requirement. A high-level architectural overview of our approach is illustrated in Figure 3.

For each incoming request, our server first distinguished between three cases:

- **Hybrid requests**. When a request for a hybrid plans is received, copies of the associated Myria and SciDB plans are retrieved (the former via the Myria REST endpoint). These plans are then combined into a Perfopticon-compatible format using the algorithm described in simplified form in Algorithm 1. Note that this process (i) attaches a system of origin to each fragment, (ii) concatenates all fragments into a flattened list, and (iii) ensures that parent/child relationships are established between fragment pairs having different origin systems.
- **Myria requests**. When we receive a request for Myria-specific, non-hybrid resources (e.g., metadata about a relation that exists only on the Myria DBMS), we issue a 301 redirect to the Myria REST server. This enables the hybrid HTTP server to be the sole communication endpoint for HybridPerfopticon without our needing to choose between two URLs on the client side.
- **SciDB requests**. Requests for SciDB profiling data (e.g., the execution time offsets for a SciDB subquery) directly leveraged the plan and profiling logic described above; responses for these requests utilized a schema that conformed precisely to their Myria counterparts.

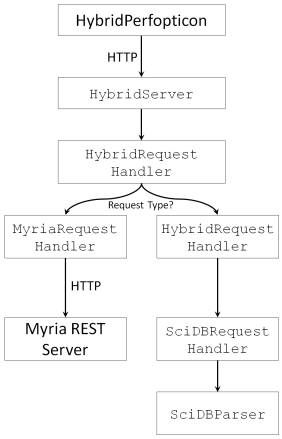

Figure 3. Architecture of the HTTP server that aggregates plans from each database system and returns a (combined) hybrid plan.

RESULTS

We now present an overview of our visual encoding choices and the visualization extensions present in HybridPerfopticon by introducing a sample hybrid use-case. In our example, a user first inputs the hybrid query identified in figure 5. This query is first performs a redimension in SciDB and then transmits the resulting dataset to Myria. The Myria DBMS then performs a join operation and stores a projection of the result.

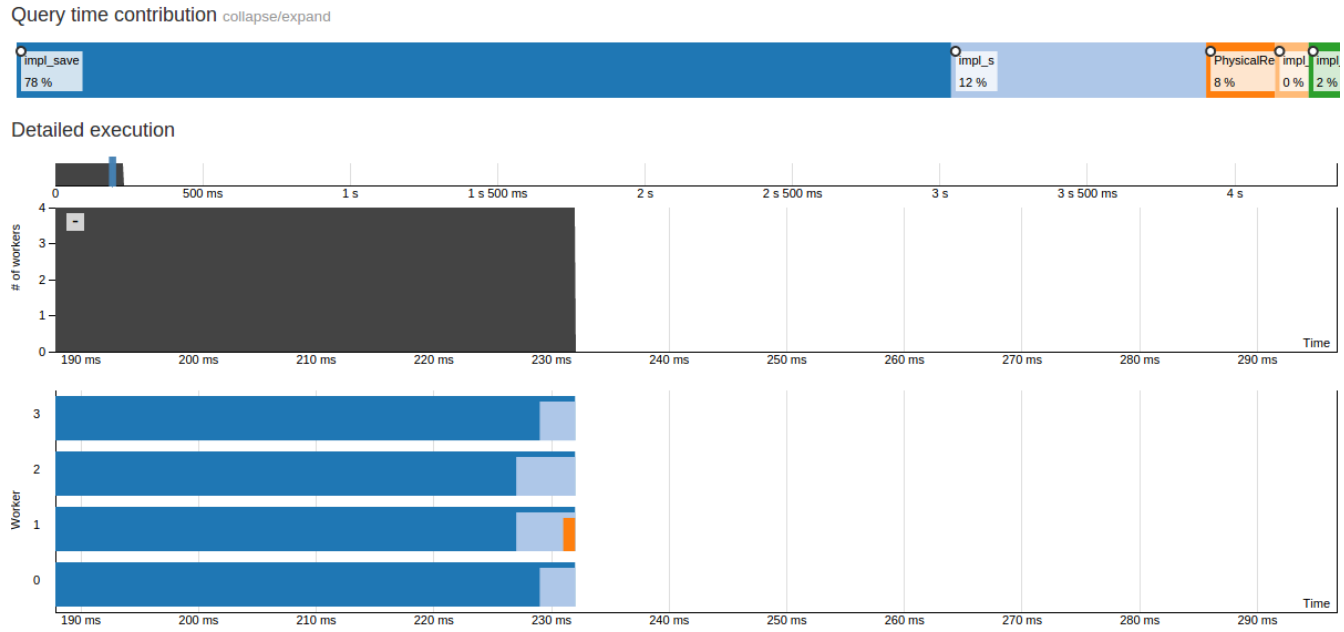

Figure 4. Detailed profiling results for the SciDB fragment in Figure 6.

Algorithm 1 Combining Myria and SciDB plans

function CREATE-HYBRID-PLAN(id, url_{MYRIA})
1: plan = HTTP−GET(id, url_{MYRIA})
2: **for each** $\text{fragment}_{\text{MYRIA}} \in$ plan **do**
3: $\text{fragment}_{\text{MYRIA}}$.origin = ”Myria”
4: **for each** $\text{op}_{\text{MYRIA}} \in \text{fragment}_{\text{MYRIA}}$ **do**
5: **if** op_{MYRIA}.type is `SciDBScan` **then**
6: $\text{plan}_{\text{SCIDB}}$ = load(op_{MYRIA}.parent.id)
7: **for each** $\text{fragment}_{\text{SCIDB}} \in \text{plan}_{\text{SCIDB}}$ **do**
8: $\text{fragment}_{\text{SCIDB}}$.origin = ”SciDB”
9: plan.fragments += $\text{fragment}_{\text{SCIDB}}$
10: **for each** $\text{op}_{\text{SCIDB}} \in \text{fragment}_{\text{SCIDB}}$ **do**
11: **if** op_{SCIDB}.type is `MyriaSave` **then**
12: op_{SCIDB}.child.id = op_{MYRIA}.id
13: return plan

SciDB

```
save(
  redimension(
    load('auto-mpg.csv', -2, 'csv'),
    <mpg: double, type:string>[id=0:*, 1, 0]),
  'intermediate', -1, 'csv+');
```

Myria

```
mpgs = load("intermediate",
            csv(schema(id:int,
                       mpg:float,
                       type:string)));
prices = load("auto-prices.csv",
              csv(schema(type:string,
                         price:int)));
dollars_per_mpg = [from mpgs, prices
                   where mpgs.type = prices.type
                   emit id, type, price / mpg];

store(dollars_per_mpg, result);
```

Figure 5. A hybrid query used as input to HybridPerfopticon. See Figure 6 for generated hybrid plan.

Figure 6 shows the hybrid visualization for this query. Note that we distinguish the origin of each fragment by color.

Since SciDB treats each query as being fully independent, each part of the hybrid query executed on SciDB is represented as a single fragment. Should the input query have had multiple, potentially-interleaved SciDB queries, we would have seen additional SciDB fragments in the resulting hybrid plan visualization.

As with Perfopticon, a user can select individual fragments and get detailed profiling views. This is illustrated in Figure 4. The hybrid HTTP server which serves the plans also returns profiling API requests and automatically reconciles differences between SciDB and Myria fragments.

The next feature added to HybridPerfopticon involves the animation of the query plan by showing tuple- and array element-flow across different operators at various points in query execution process. This is shown to the user by animating the size of the connecting arrows, and adjusting those sizes during the playback process. The idea here is to give a global perspective of how elements are being processed in the hybrid query, with the intention that a user will quickly identify areas of interest for subsequent drill-down.

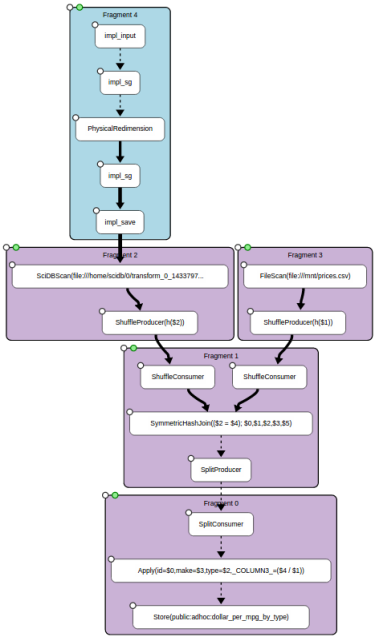

Figure 6. A hybrid plan as visualized by HybridPerfopticon. This hybrid query begins with a SciDB scan/redimension, after which the resulting array is transferred to Myria for joining on a second dataset.

Our final extension in HybridPerfopticon involved illustrating inter-database transfer skew. In our display, skew is binned, color-encoded and displayed as a (normalized) mean over the previous two seconds. This running average is important to minimize rapid fluctuation which are both distracting and serve to conceal the far more important global patterns of skew that occur in this class of queries.

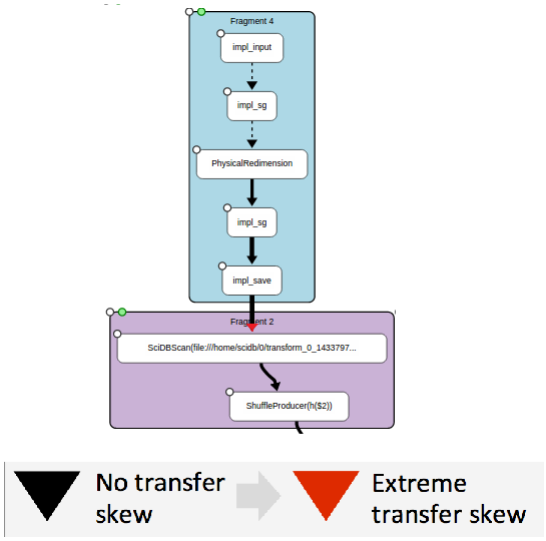

Figure 7. Visualizing skew in inter-database transfers

CONCLUSIONS & FUTURE WORK

In this work we described HybridPerfopticon, a visualization tool for hybrid queries that cross DBMS boundaries. A tool that does so is critically needed, especially given that the performance profiles of this class of queries are not well-understood. Accordingly, we expect that this tool will serve as a useful means by which the hybrid DBMS space may be explored.

While our approach of using existing log files to generate performance metrics is certainly not novel, the use of log file parsing to quickly add a second DBMS to a query visualization framework represents an opportunity to extend the reach of HybridPerfopticon to a wide-ranging set of systems. Future work should evaluate the applicability of this approach, and if generalizable, additional attention should be paid to identifying the optimal interface by which logging metadata may be exposed to HybridPerfopticon.

However, while our log-parsing approach allowed us to quickly expose profiling data for a second DBMS inside HybridPerfopticon, more work remains to be done. In particular, the hybrid HTTP server needs a far more robust caching subsystem; in particular, increasingly large logging files will quickly lead to an intractable startup time. This would be easily remedied by ensuring that the results of the parsing operation were themselves logged to a DBMS for quick subsequent retrieval.

Additionally, future work should evaluate how to best degrade in the face of differing levels of profiling granularity. For example, we were able to produce SciDB profiling metrics at the millisecond level, whereas Myria profiles at nanosecond granularity. A hybrid visualization framework should be able to gracefully reconcile both; similarly, we expect that some systems will not have support for some profiling areas and that the framework should evolve to gracefully handle these cases.

Finally, given that we hypothesize that hybrid queries will tend to be larger than their single-database counterparts, it may be the case that the resulting plans with numerous fragments are not easily navigated in the HybridPerfopticon system. Follow-on work should explore whether these fragments can be combined into larger groupings, either by database system type or (better yet) by grouping fragments by higher-level function.

REFERENCES

1. Google dapper. `http://research.google.com/pubs/pub36356.html`.
2. Gprof. `https://sourceware.org/binutils/docs/gprof/`.
3. Scidb by paradigm4. `http://www.paradigm4.com/`.
4. Twitter ambrose. `https://github.com/twitter/ambrose`.
5. D. J. DeWitt, A. Halverson, R. Nehme, S. Shankar, J. Aguilar-Saborit, A. Avanes, M. Flasza, and J. Gramling. Split query processing in polybase. In *Proceedings of the 2013 ACM SIGMOD International*

Conference on Management of Data, pages 1255–1266. ACM, 2013.

6. D. Halperin, V. Teixeira de Almeida, L. L. Choo, S. Chu, P. Koutris, D. Moritz, J. Ortiz, V. Ruamviboonsuk, J. Wang, A. Whitaker, et al. Demonstration of the myria big data management service. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 881–884. ACM, 2014.

7. H. Lim, Y. Han, and S. Babu. How to fit when no one size fits. In *CIDR*, volume 4, page 35, 2013.

8. D. Moritz, D. Halperin, B. Howe, and J. Heer. Perfopticon: Visual query analysis for distributed databases. In *Computer Graphics Forum (EuroVis), Cagliari, Italy*, volume 34, 2015.

9. M. Stonebraker, S. Madden, D. J. Abadi, S. Harizopoulos, N. Hachem, and P. Helland. The end of an architectural era:(it's time for a complete rewrite). In *Proceedings of the 33rd international conference on Very large data bases*, pages 1150–1160. VLDB Endowment, 2007.