Data Movement in Hybrid Analytic Systems: A Case for Automation

Patrick Leyshock, David Maier, Kristin Tufte
Department of Computer Science
Portland State University
Portland, Oregon, U.S.A.
+1.503.725.4036
leyshock, maier, tufte@pdx.edu

ABSTRACT
Hybrid data analysis systems integrate an analytic tool and a data management tool. While hybrid systems have benefits, in order to be effective data movement between the two hybrid components must be minimized. Through experimental results we demonstrate that under workloads whose inputs vary in size, shape, and location, automation is the only practical way to manage data movement in hybrid systems.

Categories and Subject Descriptors
H.2.8 [**Database Management**]: Database Applications

General Terms
Performance, Design, Experimentation

Keywords
Big data analytics; hybrid analytic systems; query optimization; R; SciDB

1. INTRODUCTION
Hybrid data analysis systems integrate high-level analytic tools with powerful server-based "Big Data" management systems. This approach couples the complementary strengths of each component: the analytic tool provides rich analytic functionality and a familiar interface to data scientists, while the data management system operates effectively on large disk-resident data, performing simpler, lower-level analytic tasks.

In hybrid systems, data is stored at both components and analytic work is performed at both components. Thus, data and intermediate results must move between components. Data movement between hybrid components comes with costs: it takes time and energy [1-2]. One way to reduce data movement costs is to *minimize the amount of data moved between hybrid components.* We believe that data scientists should not be responsible for managing and minimizing data movement, and argue that data movement minimization should be automated.

The primary goal of this paper is to demonstrate the need for automatic reduction of data movement, by illustrating that the alternative – viz., manually minimizing data movement – is not viable. We investigate data movement costs using a hybrid system of our own design, named Agrios. Agrios is middleware integrating R and SciDB, and it automatically minimizes data movement between the two components. R is a common analytic platform, and SciDB is an array-based database [3-4]. Agrios operates exclusively on array-modeled data. For additional details on Agrios we point readers to our earlier publication [5].

SSDBM '14, June 30 - July 02 2014, Aalborg, Denmark
 ACM 978-1-4503-2722-0/14/06□$15.00.
http://dx.doi.org/10.1145/2618243.2618273.

2. THE PROBLEM
Data scientists input analytic work to Agrios as *queries*. Each query instance is made up of input data objects and array operators. Hybrid systems can store data at both hybrid components, so the input data objects have a particular initial *placement*. A *placement* is an assignment of *locations* to all input data objects in a query. Placements may vary between query instances. In Agrios, input data objects are located at either R or SciDB. Data objects have other properties in addition to a location; since Agrios uses an array data model, an array data object also has a *size* and *shape* determined by the number and extent of its dimensions.

Hybrid systems can perform analytic work at both components of the hybrid, so for every query instance, the execution locations for each of the query's operations must be specified. A *staging* is a complete assignment of execution locations to a query's operators; it specifies where each operation in the query instance should be performed. Operations in Agrios are performed at R or SciDB.

A placement and a staging determine what data needs to be moved where for query execution. For a particular placement, some stagings move more data than other stagings. A staging S that moves less data than another staging S′ has a lower *cost* than staging S′. *Minimizing data movement in hybrid analytic systems amounts to identifying the staging that moves the minimal amount of data for a given placement.* This staging is the optimal staging – the lowest-cost staging – for that placement.

We argue that finding optimal stagings is challenging, and a task best left to a tool that automatically identifies them. Automatically identifying optimal stagings is not a luxury for hybrid systems with diverse input properties, but a necessity. Manually identifying the optimal staging is an unacceptable alternative. We support this conclusion by demonstrating three claims:

A. *Good stagings for a placement are rare.* The odds of choosing an optimal or acceptable staging are low, as only a small fraction of stagings are good for a given placement.
B. *Good stagings are not good for much.* A staging that has an acceptable cost for one placement will likely not be good for another, arbitrary placement, as a staging is

generally good for at most a small number of placements.

C. *Worst-case costs are unacceptable.* The cost of the worst stagings for a placement are much greater than the cost of the best stagings. Ignoring the choice of staging is a poor strategy.

3. EXAMINATION OF STAGING SPACE

3.1 Methodology

We performed a number of experiments and analyses to argue for Claims A, B, and C above. In the experiments we used a number of synthetic test queries containing operators typically encountered in analytic workflows. For purposes of exposition, in this paper we focus on two representative queries. Query 1, written in R, is:

```
((( A %*% B ) %*% ( C + ( D + E )))
      + (( F + (G %*% (H %*% I ))) [1,1] )) [1:30,1:40]
```

and Query 2 is:

```
(((sum(B) + C) %*% A[1:50, 1:50]  %*% ((( D %*% E )
      + F[1:100,1] )[1:50])
```

The queries are graphically depicted in Figure 1.

Four different operators are used in the two queries: matrix multiplication: '%*%', elementwise addition: '+', subscript: '[ ]', and aggregate sum: 'sum( )'. These four operations can be performed by either R or SciDB. We selected these operations in part for that reason, and also because they: i) are common building blocks of complex analyses, ii) have different algebraic properties, and iii) modify the shape of their inputs in important ways. Though not examined here, Agrios easily handles operations that can be performed at only one of the two components.

Possible staging costs are best represented as a matrix: a *staging-space matrix*. At its simplest, a staging-space matrix is two-dimensional: one dimension's length is determined by the number of possible placements, the other by the number of possible stagings. Each cell of the matrix shows the plan cost for a particular placement and a particular staging.

Staging-space matrices are valuable to our investigation because they show the costs of all possible stagings and placements. The matrix captures not only the optimal stagings for all placements, but also all suboptimal stagings. Staging-space matrices show us both how expensive – and how inexpensive – stagings can be.

We created staging-space matrices using a test harness attached to Agrios. The harness populates a matrix by iterating through all of a query's possible placements and stagings. Transformation-based rewrites in Agrios were disabled to keep the investigation simple. (See [5] for additional information about transformations.)

3.2 Claim A: Good Stagings for a Placement Are Rare

Examining staging space we see that there are very few optimal stagings for each of a query's placements. Query 2 has 64 placements and 512 stagings. For each placement, there are between one and 512 stagings with minimal – i.e. optimal – cost. If Agrios is used in this hybrid system, identification of the optimal staging is guaranteed. If Agrios is not used, the data scientist is responsible for finding the optimal staging from among the alternatives.

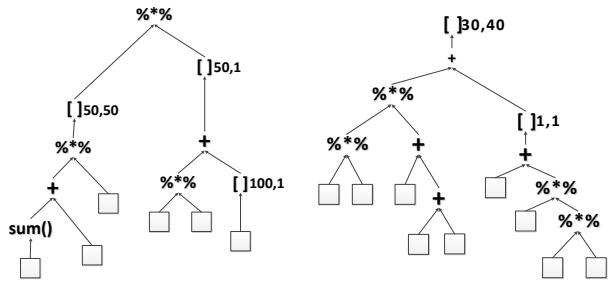

Figure 1. Test Query 1 (left) and Query 2 (right).

Figure 2 presents an analysis of Query 2's staging space. It shows that generally, for a given placement, the percentage of optimal stagings among possible stagings is low. For example, for 40 of the query's 64 placements less than 0.2% of the 512 possible stagings have optimal cost. In fact, for each these forty placements, only one of the 512 possible stagings are optimal. The likelihood of finding an optimal plan does not improve much when we look over all 64 placements, even though there are multiple optimal stagings for some placements. If we look across all pairs of a placement and a staging, the staging will be optimal for the placement only 0.07% of the time.

It might be argued that optimality is too lofty an objective, that a staging is acceptable if it is "close enough" to the optimal cost. Removing the requirement of optimality does not much improve matters. While there are more "acceptable" stagings than optimal stagings, even acceptable stagings are by no means common. Figure 2 also shows how often a given percentage of stagings for a placement fall within 150% and 200% of the optimal cost. Acceptable stagings are nearly as rare as optimal stagings.

In practice, a data scientist likely would not randomly select a staging, but rather reason about what staging is optimal. (Note that this assumes the query volume is small enough that the data scientist has enough time to perform this work. Under some reasonable workloads – e.g., when query instances are performed many times per minute – this assumption is shaky at best.) Moreover, while "in the wild" there might be variation in placements, more often than not we would expect large inputs to be placed at SciDB, and small inputs to be placed at R.

Suppose that a query instance had this sort of "typical" placement – most large objects at SciDB, most small objects at R. This sort of placement suggests that an intuitive, "naive" staging approach such as "Do all operations at SciDB" would be effective. Experiments show, however, that "naive" staging strategies may result in stagings up to ten times more expensive than optimal [5]. Intuition is not a reliable guide for identifying the optimal staging.

These facts support Claim A above: optimal stagings and even acceptable suboptimal stagings, for a given placement, are few and far between. Data scientists hoping to identify optimal or acceptable stagings by hand have a tall order. Agrios, by contrast, automates identification of the optimal staging, relieving data scientists from that burden.

3.3 Claim B: Good Stagings are Not Good for Much

We demonstrated in the section above that acceptable stagings are uncommon. Let us now assume a good staging – even an optimal one – for a particular placement has been identified. Just how useful is this staging for other placements? Alternatively: How often is a good staging a good staging? Ideally, a good staging would perform well over a large percentage of the possible place-

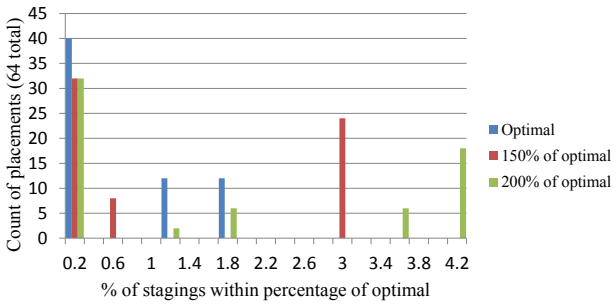

Figure 2. Histogram of placements for Query 2. The horizontal axis shows what percentage of stagings fall within a given percent of optimal. The vertical axis is the count of placements in each range. For forty of the 64 placements, fewer than 0.2% of stagings (i.e. only one staging) are optimal. For over a third of the placements, fewer than 16 of the 512 stagings (3%) are within 150% of optimal cost. The upshot is that there are very few optimal stagings, regardless of placement.

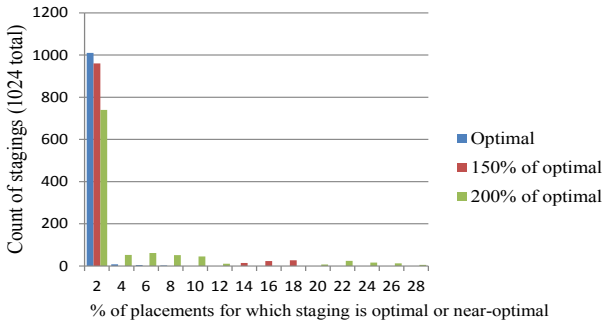

Figure 3. Histogram of stagings for Query 1. The horizontal axis shows the percentage of placements for which the stagings are optimal or near-optimal. Over 1000 stagings (out of 1024) are optimal for fewer than 2% of placements. Given an optimal staging for a particular placement, odds are it is suboptimal for most other placements.

ments: such a staging is *robust*. If, typically, good stagings are robust, then there is little need for Agrios; once an optimal staging for a given placement is identified, the staging can simply be reused for other placements.

Staging space shows that most stagings, however, are not robust. A staging whose cost is optimal for one particular placement is suboptimal for many placements, and wholly unacceptable for others. Figure 3 shows results for Query 1; these results are typical. Nearly all of the query's 1024 stagings are optimal for less than 2% of its 512 placements. Reframing this result in terms of likelihoods, odds are nearly certain that if an optimal staging is identified, it is optimal only for ten or fewer placements out of many hundreds of possible placements. Figure 3 also shows results for experiments when the optimality requirement is relaxed – counts of stagings when costs are within 150% and 200% of optimal for some placement. We see that even suboptimal but acceptable stagings are good for only a limited number of placements.

Examining a particular example explains why some stagings are acceptable for only a limited number of placements. Figure 4 shows two instances of Query 2, differing only in the initial placement of data objects E and F. In the first instance of the query, shown at left in Figure 4, E is placed at R and F is placed at

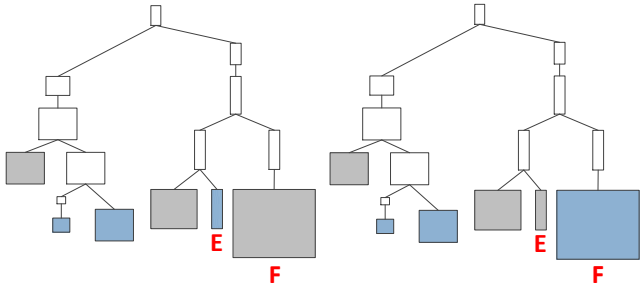

Figure 4. Two instances of Query 2. The instances differ only in their placements; each instance places leaf-level data objects E and F at different locations.

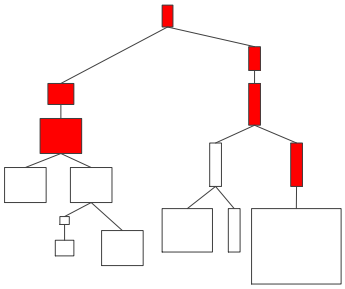

Figure 5. A comparison of the optimal stagings for the two instances of Query 2 depicted in Figure 4. Operations whose execution locations differ between optimal stagings are shaded red. Note that both main branches of the tree contain shaded operators, though the initial placement difference between the two instances is isolated to one branch.

SciDB. In the second instance, shown at right in Figure 4, E is placed at SciDB and F is placed at R. The input placements of these two query instances differ only slightly, but their optimal stagings differ significantly. Figure 5 shows the difference between stagings for both instances; execution locations differing between instances are shaded red. Though the differences in inputs are isolated to the right-hand branch of the tree's root operation, optimal execution locations differ in both the right-hand *and* left-hand branches. Variations in input placements affect the optimal execution locations of "nonlocal" operations, hence local adjustment of a staging for a modified placement might not give optimality.

The fact that two similar placements might have substantially different optimal stagings is significant because intuition suggests otherwise. Suppose a data scientist decided to bank on this intuition, for a workflow exhibiting only small variations in initial data placements. She might dismiss the need for Agrios, believing that once an optimal staging had been identified for a given placement, that it could be reused for the similar placements in the workflow. This example shows that at least in some cases, her intuition would be incorrect.

We showed earlier that optimal stagings are rare for a given placement, and now see that optimal stagings are also typically acceptable for only a small number of placements. Efforts spent on hand-staging may very well be in vain, and the fruit borne from the work likely has utility only for a small number of placements. Automatically identifying the optimal staging is a more sensible solution for managing data movement.

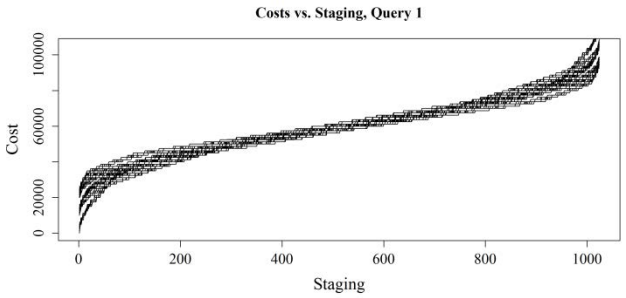

Figure 6. Another perspective on the plan space for Query 1. Costs for all stagings are sorted and plotted, for all 512 placements; each line represents a particular placement. The left end of a line shows the optimal staging cost for that placement.

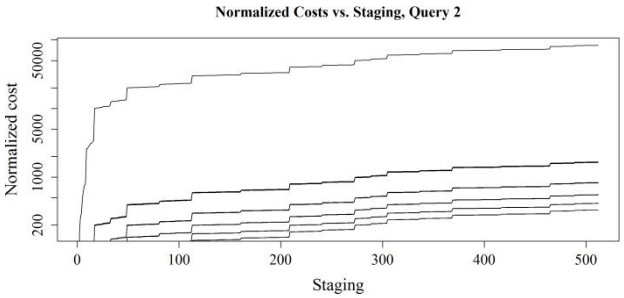

Figure 7. Normalized costs for all stagings of Query 2; one line for each placement. Note both that the vertical axis has a logarithmic scale, and that some lines overlay one another; a total of 64 placements are represented here. This plot illustrates how expensive suboptimal plans can be. Though the worst-case for other placements are not this extreme, all worst-case stagings still move over 200 times more data elements than the optimal staging.

3.4 Claim C: Worst-case costs are unacceptable

Above we motivated the use of Agrios by arguing that hand-identifying optimal stagings was difficult. The difficulty of hand-staging is moot, however, if the price for misidentifying the optimal staging is low. Additional examination of staging space shows that the cost of failure is *not* low, as illustrated by Figures 6 and 7. These plots show that the price of failure is high, arguing for the utility of a system such as Agrios that guarantees identification of the optimal staging.

In Figures 6 and 7, each line represents an individual placement. The line's shape depicts costs for all possible stagings, sorted in increasing order. Thus, the lower-left end of each line shows the optimal cost for that placement. Figure 6 shows actual costs on the vertical axis, while Figure 7 shows normalized costs, i.e. cost as a ratio to optimal costs. For one placement, the most expensive staging moves over *50,000 times* more data elements than the optimal staging. A strategy of ignoring the optimal staging can yield a 50,000-fold cost penalty, at least in some cases. Even in cases where the penalty is not so high, the cost of suboptimal stagings is often substantially greater than the cost of optimal stagings.

4. RELATED WORK

R and several data management tools have already been integrated into hybrid systems. RIOT-DB – "R with I/O Transparency" – integrates R and a MySQL database, though researchers have plans to replace the system's RDBMS back-end with a dedicated array-based storage system [6]. RICE's R-Op integrates R with SAP's HANA, an in-memory parallel database [7]. Ricardo integrates R with the Hadoop stack, an open-source implementation of Google's MapReduce system [8]. While the systems mentioned above are functioning hybrid systems, unlike Agrios none of them automates the reduction of data movement between the hybrid components.

5. CONCLUSION

Hybrid systems, integrating a sophisticated analytic tool with a dedicated "Big Data" management system, provide many benefits to data scientists. With their utility comes the burden of managing data movement between the components of the hybrid.

Data movement is managed by providing a *staging* for each query instance: specifying execution locations for each of the query's operations, given a placement of the input data objects. Stagings may be chosen either by hand, or with a tool that automatically identifies optimal stagings, such as Agrios. We argued that hand-staging is impractical for an important class of workloads, as is relying upon a single staging for all query instances. Our analysis showed that optimal stagings are difficult to find and typically optimal for only a small number of placements. We also showed that the worst-case costs are so large they cannot be ignored.

6. ACKNOWLEDGMENTS

This work is supported by National Science Foundation Grant Number 1110917, and funded in part by the Intel Science and Technology Center for Big Data. Thanks to Jason Nelson, Portland State University's Datalab, the SciDB team, and Paradigm4.

7. REFERENCES

[1] Park, J., Bikshandi, G., Vaidyanathan, K., Tang, P., Dubey, P., and Kim, D. 2013. Tera-Scale 1D FFT with Low-Communication Algorithm and Intel Xeon Phi Coprocessors. *Proceedings of SC13.*

[2] Tiwari, D., Vazhkudai, S., Kim, Y., Ma, X., Boboila, S. and Desnoyers, P. 2012. Reducing data movement costs using energy-efficient, active computation on SSD. *USENIX 2012*, 1-5.

[3] Ihaka, R., and Gentelman, R. 1996. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 299-314.

[4] Stonebraker, M., Brown, P., Poliakov, A., Raman, S. 2011. The Architecture of SciDB. *Proceedings of SSDBM 2011,*

[5] Leyshock, P., Maier, D., and Tufte, K. 2013. Agrios: A hybrid approach to big array analytics. *IEEE International Conference on Big Data*, 85–93.

[6] Yi, Z., Herodotou, H., and Yang, J. 2009. RIOT: I/O-efficient numerical computing without SQL. *CIDR 2009*, 1-11.

[7] Grosse, P., Lehner, W., Weichert, T., Farber, F., and Li, W.S. 2011. Bridging two worlds with RICE. *Proceedings of the VLDB Endowment*, 1307-1317.

[8] Das, S., Simanis, Y., Beyer, K.S., Gemulla, R., Haas, P.J., and McPherson, J. 2011. Ricardo: Integrating R and Hadoop. *Proceedings of the 2010 International Conference on Management of Data,* 987-998