

## Chapter 11: A Biased Take on a Moving Target: Complex Analytics

by Michael Stonebraker

In the past 5-10 years, new analytic workloads have emerged that are more complex than the typical business intelligence (BI) use case. For example, internet advertisers might want to know “How do women who bought an Apple computer in the last four days differ statistically from women who purchased a Ford pickup truck in the same time period?” The next question might be: “Among all our ads, which one is the most profitable to show to the female Ford buyers based on their click-through likelihood?” These are the questions asked by today’s data scientists, and represent a very different use case from the traditional SQL analytics run by business intelligence specialists. It is widely assumed that data science will completely replace business intelligence over the next decade or two, since it represents a more sophisticated approach to mining data warehouses for new insights. As such, this document focuses on the needs of data scientists.

I will start this section with a description of what I see as the job description of a data scientist. After cleaning and wrangling his data, which currently consumes the vast majority of his time and which is discussed in the section on data integration, he generally performs the following iteration:

```
Until (tired) {
  Data management operation(s);
  Analytic operation(s);
}
```

In other words, he has an iterative discovery process, whereby he isolates a data set of interest and then performs some analytic operation on it. This often suggests either a different data set to try the same analytic on or a different analytic on the same data set. By and large what distinguishes data science from business intelligence is that the analytics are predictive modeling, machine learning, regressions, ... and not SQL analytics.

In general, there is a pipeline of computations that constitutes the analytics. For example, Tamr has a module which performs entity consolidation (deduplication) on a collection of records, say  $N$  of them, at scale. To avoid the  $N^2$  complexity of brute force algorithms, Tamr identifies a collection of “features”, divides them into ranges that are unlikely to co-occur, computes (perhaps multiple) “bins” for each record based on these ranges, reshuffles the data in parallel so it is partitioned by bin number, deduplicates each bin, merges the results, and finally constructs composite records out of the various clusters of duplicates. This pipeline is partly SQL-oriented (partitioning) and partly array-oriented

analytics. Tamr seems to be typical of data science workloads in that it is a pipeline with half a dozen steps.

Some analytic pipelines are “one-shots” which are run once on a batch of records. However, most production applications are incremental in nature. For example, Tamr is run on an initial batch of input records and then periodically a new “delta” must be processed as new or changed input records arrive. There are two approaches to incremental operation. If deltas are processed as “mini batches” at periodic intervals of (say) one day, one can add the next delta to the previously processed batch and rerun the entire pipeline on all the data each time the input changes. Such a strategy will be very wasteful of computing resources. Instead, we will focus on the case where incremental algorithms must be run after an initial batch processing operation. Such incremental algorithms require intermediate states of the analysis to be saved to persistent storage at each iteration. Although the Tamr pipeline is of length 6 or so, each step must be saved to persistent storage to support incremental operation. Since saving state is a data management operation, this makes the analytics pipeline of length one.

The ultimate “real time” solution is to run incremental analytics continuously services by a streaming platform such as discussed in the section on new DBMS technology. Depending on the arrival rate of new records, either solution may be preferred.

Most complex analytics are array-oriented, i.e. they are a collection of linear algebra operations defined on arrays. Some analytics are graph oriented, such as social network analysis. It is clear that arrays can be simulated on table-based systems and that graphs can be simulated on either table systems or array systems. As such, later in this document, we discuss how many different architectures are needed for this used case.

Some problems deal with dense arrays, which are ones where almost all cells have a value. For example, an array of closing stock prices over time for all securities on the NYSE will be dense, since every stock has a closing price for each trading day. On the other hand, some problems are sparse. For example, a social networking use case represented as a matrix would have a cell value for every pair of persons that were associated in some way. Clearly, this matrix will be very sparse. Analytics on sparse arrays are quite different from analytics on dense arrays.

In this section we will discuss such workloads at scale. If one wants to perform such pipelines on “small data” then any solution will work fine.

The goal of a data science platform is to support this iterative discovery process. We begin with a sad truth. Most data science platforms are file-based and have nothing to do with DBMSs. The preponderance of analytic codes are run in R, MatLab, SPSS, SAS and operate on file data. In addition, many Spark users are reading data from files. An exemplar of this state of affairs is the NERSC high performance computing (HPC) system at Lawrence Berkeley Labs. This machine is used essentially exclusively for complex analytics; however, we were unable to get the Vertica DBMS to run at all, because of configuration restrictions. In addition, most “big science” projects build an entire software stack from the bare metal on up. It is plausible that this state of affairs will continue, and DBMSs will not become a player in this market. However, there are some hopeful signs such as the fact that genetic data is starting to be deployed on DBMSs, for example the 1000 Genomes Project [13] is based on SciDB.

In my opinion, file system technology suffers from several big disadvantages. First metadata (calibration, time, etc.) is often not captured or is encoded in the name of the file, and is therefore not searchable. Second, sophisticated data processing to do the data management piece of the data science workload is not available and must be written (somehow). Third, file data is difficult to share data among colleagues. I know of several projects which export their data along with their parsing program. The recipient may be unable to recompile this accessor program or it generates an error. In the rest of this discussion, I will assume that data scientists over time wish to use DBMS technology. Hence, there will be no further discussion of file-based solutions.

With this backdrop, we show in Table 1 a classification of data science platforms. To perform the data management portion, one needs a DBMS, according to our assumption above. This DBMS can have one of two flavors. First, it can be record-oriented as in a relational row store or a NoSQL engine or column-oriented as in most data warehouse systems. In these cases, the DBMS data structure is not focused on the needs of analytics, which are essentially all array-oriented, so a more natural choice would be an array DBMS. The latter case has the advantage that no conversion from a record or column structure is required to perform analytics. Hence, an array structure will have an innate advantage in performance. In addition, an array-oriented storage structure is multi-dimensional in nature, as opposed to table structures which are usually one-dimensional. Again, this is likely to result in higher performance.

The second dimension concerns the coupling between the analytics and the DBMS. On the one hand, they can be independent, and one can run a query, copying the result to a different address space where the analytics are run. At the end of the analytics pipeline (often of length one), the result

can be saved back to persistent storage. This will result in lots of data churn between the DBMS and the analytics. On the other hand, one can run analytics as user-defined functions in the same address space as the DBMS. Obviously the tight coupling alternative will lower data churn and should result in superior performance.

In this light, there are four cells, as noted in Table 1. In the lower left corner, Map-Reduce used to be the exemplar; more recently Spark has eclipsed Map-Reduce as the platform with the most interest. There is no persistence mechanism in Spark, which depends on RedShift or H-Base, or ... for this purpose. Hence, in Spark a user runs a query in some DBMS to generate a data set, which is loaded into Spark, where analytics are performed. The DBMSs supported by Spark are all record or column-oriented, so a conversion to array representation is required for the analytics.

A notable example in the lower right hand corner is MADLIB [8], which is a user-defined function library supported by the RDBMS Greenplum. Other vendors have more recently started supporting other alternatives; for example Vertica supports user-defined functions in R. In the upper right hand corner are array systems with built-in analytics such as SciDB [15], TileDB [6] or Rasdaman [1].

In the rest of this document, we discuss performance implications. First, one would expect performance to improve as one moves from lower left to upper right in Table 1. Second, most complex analytics reduce to a small collection of “inner loop” operations, such as matrix multiply, singular-value decomposition and QR decomposition. All are computationally intensive, typically floating point codes. It is accepted by most that hardware-specific pipelining can make nearly an order of magnitude difference in performance on these sorts of codes. As such, libraries such as BLAS, LAPACK, and ScaLAPACK, which call the hardware-optimized Intel MKL library, will be wildly faster than codes which don’t use hardware optimization. Of course, hardware optimization will make a big difference on dense array calculations, where the majority of the effort is in floating point computation. It will be less significance on sparse arrays, where indexing issues may dominate the computation time.

Third, codes that provide approximate answers are way faster than ones that produce exact answers. If you can deal with an approximate answer, then you will save mountains of time.

Fourth, High Performance Computing (HPC) hardware are generally configured to support large batch jobs. As such, they are often structured as a computation server connected to a storage server by networking, whereby a program must pre-allocate disk space in a computation server cache for its storage needs. This is obviously at odds with a DBMS, which expects to be continuously running as a ser-

	Loosely coupled	Tightly coupled
Array representation		SciDB, TileDB, Rasdaman
Table representation	Spark + HBase	MADLib, Vertica + R

Table 1: A Classification of Data Science Platforms

vice. Hence, be aware that you may have trouble with DBMS systems on HPC environments. An interesting area of exploration is whether HPC machines can deal with both interactive and batch workloads simultaneously without sacrificing performance.

Fifth, scalable data science codes invariably run on multiple nodes in a computer network and are often network-bound [5]. In this case, you must pay careful attention to networking costs and TCP-IP may not be a good choice. In general MPI is a higher performance alternative.

Sixth, most analytics codes that we have tested fail to scale to large data set sizes, either because they run out of main memory or because they generate temporaries that are too large. Make sure you test any platform you would consider running on the data set sizes you expect in production!

Seventh, the structure of your analytics pipeline is crucial. If your pipeline is on length one, then tight coupling is almost certainly a good idea. On the other hand, if the pipeline is on length 10, loose coupling will perform almost as well. In incremental operation, expect pipelines of length one.

In general, all solutions we know of have scalability and performance problems. Moreover, most of the exemplars noted above are rapidly moving targets, so performance and scalability will undoubtedly improve. In summary, it will be interesting to see which cells in Table 1 have legs and which ones don't. The commercial marketplace will be the ultimate arbiter!

In my opinion, complex analytics is current in its "wild west" phase, and we hope that the next edition of the red book can identify a collection of core seminal papers. In the meantime, there is substantial research to be performed. Specifically, we would encourage more benchmarking in this space in order to identify flaws in existing platforms and to spur further research and development, especially benchmarks that look at end-to-end tasks involving both data management tasks and analytics. This space is moving fast, so the benchmark results will likely be transient. That's probably a good thing: we're in a phase where the various projects should be learning from each other.

There is currently a lot of interest in custom parallel algorithms for core analytics tasks like convex optimization; some of it from the database community. It will be inter-

esting to see if these algorithms can be incorporated into analytic DBMSs, since they don't typically follow a traditional dataflow execution style. An exemplar here is Hogwild! [12], which achieves very fast performance by allowing lock-free parallelism in shared memory. Google Downpour [4] and Microsoft's Project Adam [2] both adapt this basic idea to a distributed context for deep learning.

Another area where exploration is warranted is out-of-memory algorithms. For example, Spark requires your data structures to fit into the combined amount of main memory present on the machines in your cluster. Such solutions will be brittle, and will almost certainly have scalability problems.

Furthermore, an interesting topic is the desirable approach to graph analytics. One can either build special purpose graph analytics, such as GraphX [7] or GraphLab [11] and connect them to some sort of DBMS. Alternately, one can simulate such codes with either array analytics, as espoused in D4M [10] or table analytics, as suggested in [9]. Again, may the solution space bloom, and the commercial market place be the arbiter!

Lastly, many analytics codes use MPI for communication, whereas DBMSs invariably use TCP-IP. Also, parallel dense analytic packages, such as ScaLAPACK, organize data into a block-cyclic organization across nodes in a computing cluster [3]. I know of no DBMS that supports block-cyclic partitioning. Removing this impedance mismatch between analytic packages and DBMSs is an interesting research area, one that is targeted by the Intel-sponsored ISTC on Big Data [14].

---

### Commentary: Joe Hellerstein

6 December 2015

I have a rather different take on this area than Mike, both from a business perspective and in terms of research opportunities. At base, I recommend a "big tent" approach to this area. DB folk have much to contribute, but we'll do far better if we play well with others.

Let's look at the industry. First off, advanced analytics of the sort we're discussing here will not replace BI as Mike suggests. The BI industry is healthy and growing. More fundamentally, as noted

<sup>1</sup>Tukey, John. Exploratory Data Analysis. Pearson, 1977.

statistician John Tukey pointed out in his foundational work on Exploratory Data Analysis,<sup>1</sup> a chart is often much more valuable than a complex statistical model. Respect the chart!

That said, the advanced analytics and data science market is indeed growing and poised for change. But unlike the BI market, this is not a category where database technology currently plays a significant role. The incumbent in this space is SAS, a company that makes multiple billions of dollars in revenue each year, and is decidedly not a database company. When VCs look at companies in this space, they're looking for "the next SAS". SAS users are not database users. And the users of open-source alternatives like R are also not database users. If you assume as Mike does that "data scientists will want to use DBMS technology" — particularly a monolithic "analytic DBMS" — you're swimming upstream in a strong current.

For a more natural approach to the advanced analytics market, ask yourself this: what is a serious threat to SAS? Who could take a significant bite out of the cash that enterprises currently spend there? Here are some starting points to consider:

1. **Open source stats programming:** This includes R and the Python data science ecosystem (NumPy, SciKit-Learn, iPython Notebook). These solutions don't currently don't scale well, but efforts are underway aggressively to address those limitations. This ecosystem could evolve more quickly than SAS.
2. **Tight couplings to big data platforms.** When the data is big enough, performance requirements may "drag" users to a new platform — namely a platform that already hosts the big data in their organization. Hence the interest in "DataFrame" interfaces to platforms like Spark/MLlib, PivotalR/MADlib, and Vertica dplyr. Note that the advanced analytics community is highly biased toward open source. The cloud is also an interesting platform here, and not one where SAS has an advantage.
3. **Analytic Services.** By this I mean interactive online services that use analytic methods at their core: recommender systems, real-time fraud detection, predictive equipment maintenance and so on. This space has aggressive system requirements for response times, request scaling, fault tolerance and ongoing evolution that products like SAS don't address. Today, these services are largely built with custom code. This doesn't scale across industries — most companies can't recruit developers that can work at this level. So there is ostensibly an opportunity here in commoditizing this technology for the majority of use cases. But it's early days for this market — it remains to be seen whether analytics service platforms can be made simple enough for commodity deploy-

ment. If the tech evolves, then cloud-based services may have significant opportunities for disruption here as well.

On the research front, I think it's critical to think outside the database box, and collaborate aggressively. To me this almost goes without saying. Nearly every subfield in computing is working on big data analytics in some fashion, and smart people from a variety of areas are quickly learning their own lessons about data and scale. We can have fun playing with these folks, or we can ignore them to our detriment.

So where can database research have a big impact in this space? Some possibilities that look good to me include these:

1. **New approaches to Scalability.** We have successfully shown that parallel dataflow — think MADlib, MLlib or the work of Ordonez at Teradata<sup>2</sup> — can take you a long way toward implementing scalable analytics *without* doing violence at the system architecture level. That's useful to know. Moving forward, can we do something that is usefully faster and more scalable than parallel dataflow over partitioned data? Is that necessary? Hogwild! has generated some of the biggest excitement here; note that it's work that spans the DB and ML communities.
2. **Distributed infrastructure for analytic services.** As I mentioned above, analytic services are an interesting opportunity for innovation. The system infrastructure issues on this front are fairly wide open. What are the main components of architectures for analytics services? How are they stitched together? What kind of data consistency is required across the components? So-called Parameter Servers are a topic of interest right now, but only address a piece of the puzzle.<sup>3</sup> There has been some initial work on online serving, evolution and deployment of models.<sup>4</sup> I hope there will be more.
3. **Analytic lifecycle and metadata management.** This is an area where I agree with Mike. Analytics is often a people-intensive exercise, involving data exploration and transformation in addition to core statistical modeling. Along the way, a good deal of context needs to be managed to understand how models and data products get developed across a range of tools and systems. The database community has perspectives on this area that are highly relevant, including workflow management, data lineage and materialized view maintenance. VisTrails is an example of research in this space that is being used in practice.<sup>5</sup> This is an area of pressing need in industry as well — especially work that takes into account the real-world diversity of analytics tools and systems in the field.

<sup>2</sup>e.g., Ordonez, C. Integrating K-means clustering with a relational DBMS using SQL. TKDE 18(2) 2006. Also Ordonez, C. Statistical Model Computation with UDFs. TKDE 22(12), 2010.

<sup>3</sup>Ho, Q., et al. More effective distributed ML via a stale synchronous parallel parameter server. NIPS 2013.

<sup>4</sup>Crankshaw, D, et al. The missing piece in complex analytics: Low latency, scalable model management and serving with Velox. CIDR 2015. See also Schleier-Smith, J. An Architecture for Agile Machine Learning in Real-Time Applications. KDD 2015.

<sup>5</sup>See <http://www.vistrails.org>.

## References

- [1] P. Baumann, A. Dehmel, P. Furtado, R. Ritsch, and N. Widmann. The multidimensional database system rasdaman. In *SIGMOD*, 1998.
- [2] T. Chilimbi, Y. Suzue, J. Apacible, and K. Kalyanaraman. Project adam: Building an efficient and scalable deep learning training system. In *OSDI*, 2014.
- [3] J. Choi et al. ScaLAPACK: A portable linear algebra library for distributed memory computers—design issues and performance. In *Applied Parallel Computing Computations in Physics, Chemistry and Engineering Science*, pages 95–106. Springer, 1996.
- [4] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le, et al. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems*, pages 1223–1231, 2012.
- [5] J. Duggan and M. Stonebraker. Incremental elasticity for array databases. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 409–420. ACM, 2014.
- [6] A. Elmore, J. Duggan, M. Stonebraker, M. Balazinska, U. Cetintemel, V. Gadepally, J. Heer, B. Howe, J. Kepner, T. Kraska, et al. A demonstration of the BigDAWG polystore system. In *VLDB*, 2015.
- [7] J. E. Gonzales, R. S. Xin, D. Crankshaw, A. Dave, M. J. Franklin, and I. Stoica. Graphx: Unifying data-parallel and graph-parallel analytics. In *OSDI*, 2014.
- [8] J. M. Hellerstein, C. Ré, F. Schoppmann, D. Z. Wang, E. Fratkin, A. Gorajek, K. S. Ng, C. Welton, X. Feng, K. Li, et al. The MADlib analytics library: or MAD skills, the SQL. In *VLDB*, 2012.
- [9] A. Jindal, P. Rawlani, E. Wu, S. Madden, A. Deshpande, and M. Stonebraker. Vertexica: your relational friend for graph analytics! In *VLDB*, 2014.
- [10] J. Kepner et al. Dynamic distributed dimensional data model (D4M) database and computation system. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 5349–5352. IEEE, 2012.
- [11] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein. Distributed graphlab: a framework for machine learning and data mining in the cloud. In *VLDB*, 2012.
- [12] B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 693–701, 2011.
- [13] N. Siva. 1000 genomes project. *Nature biotechnology*, 26(3):256–256, 2008.
- [14] M. Stonebraker, S. Madden, and P. Dubey. Intel big data science and technology center vision and execution plan. *ACM SIGMOD Record*, 42(1):44–49, 2013.
- [15] The SciDB Development Team. Overview of SciDB: large scale array storage, processing and analysis. In *SIGMOD*, 2010.