Reimagining the Science of Databases

July 26th, 2015 by Thomas Hazel

In my previous blog, "Understanding the Science of Databases," I discussed the State of the Art of information storage, all of which is still based on 1970s computer science. This blog goes on to describe the theoretical minimum "cost" of storage manipulation to obtain maximum resource utilization, particularly with respects to the ever increasing capabilities of modern hardware. So why hasn't this 30+ year old science changed with the times?

To be fair, this science has been the workhorse all these years, in everything from File Systems to SQL, NoSQL, and NewSQL databases. Though the science has not changed, its application has. In other words, due to its mathematical limitations, features have been changed/dropped to wring out the last bits of database performance and scale. This change of behavior has resulted in the splintering of solutions and their associated architecture:

- Row-Oriented (transactions)
- Graph-Oriented (associations)
- Column-Oriented (size, analytics)
- Document-Oriented (semi/unstructured)
- File-Oriented (via Hadoop) (distributed, scale)

Each of the above architectures meet some application need, with Row-Oriented solutions being the largest stakeholder and the bedrock of business since the 1980s. The remaining are part of a collection of solutions that were started around the turn of the century to address today's proverbial Big Data demands. Yet, while Row-Oriented still runs the world's systems, the others handle niche use-cases at scale. These newcomers (e.g. Vertica, Mongo, Cloudera) are getting the headlines, while the stalwarts (e.g. Oracle, IBM, Microsoft) are making a business.

But why? These new solutions are storing more and more information and surely more than all Row-Oriented combined, so what gives? It's simple - capabilities of RDBMS systems underpin business applications and business applications are where the

money is made. Often these new solutions provide temporary Data Lakes, using additional hardware, to be processed via Extract, Transform, Load (ETL) into business oriented RDBMS systems. Though this ETL technique is “managing” today’s scale, it’s limiting business agility.

This is where my “reimagining” moment occurred in 2006. Not the actual invention underpinning the science of Deep, but the idea that if one could remove these limitations, one could reverse the Information Silo proliferation due to behavioral fragmentation. In other words, is there a data structure and algorithm that could handle critical RDBMS capabilities at today’s “Infinite-Oriented” performance and scale?

Being an avid inventor I typically start with Thought Experiments to search for unique insights, which hopefully lead to some conceptual breakthrough. This time I started with “theoretical minimums.” In other words, what is the most efficient way to manipulate storage, what is the 1970 science doing wrong and right, and what are these solutions doing with this science? I put all three aspects into my mental blender and let it stir for a few years. What I began to realize is the problem was being looked at in the wrong direction. The problem needed to be inverted.

The goal should not be to manipulate logical structure for optimal mathematics, it should be to optimally manipulate physical hardware against its constraints. The algorithm should not focus on structure behavior, but on resource behavior. There are three major components in modern hardware: Processor, Memory, and Storage. Each component has its optimal utilization, but achieving it could be at the detriment of the other components. Algorithms that do not take this into consideration will ultimately “hit a wall”. In the 1970s, the State of the Art was based on fixed structure and static behavior and is the reason solutions and architectures are fragmenting today's information.

Here is where the breakthrough began to crystalize. Use hardware component modeling (via Machine Learning) to dynamically and continually morph both the data structure and its algorithmic behavior to obtain maximum resource utilization. Schedule such resources like an Operating System Kernel to meet their theoretical minimum “cost”. Not just for Storage, but for Processor and Memory too. Now all that remained was the reimagining of... Information.