Big Data is Simple

Jonathan Ginter, BMC

There's a lot of noise and confusion over Big Data, as there is with any new technology concept. Remember the Web 2.0 conference where the official t-shirt slogan was "Web 2.0 is _____" and you had to write in your own definition? Par for the course in our field, unfortunately, and Big Data is no exception.

However, I feel compelled to wade into the fray on Big Data in the hope that I can redefine the discussion. I feel as though the problem represented by Big Data has become a monster in the eyes of the general public - something that requires complex technology, data centres full of servers and experts hovering over it all. Consequently, I would like to put forward a different vision that removes some of the challenges and makes the whole topic less scary and seemingly insurmountable.

Do We Have To Make This Complicated?

Let me start by stating something that might ruffle some feathers (and I apologize in advance if it does) - *people in our line of work tend to make a simple problem more complicated than it needs to be*. We don't like to admit it, but I believe we would all have stories about how someone else has been guilty of exactly this kind of thing (of course, we would never do that). This behaviour isn't malicious in any way. Technical people simply expect that any interesting problem should be worthy of our skill. Since we are expecting a complex problem, we find precisely what we expect to find. Simplicity is rarely sought, so it is rarely found.

However, I believe that Big Data is actually a simple problem.

Let's begin by defining what we are trying to accomplish with Big Data. Although there are all kinds of edge cases, the primary goal is to put one or more data models together and find correlations between them. We do this because we want to find the root cause of some particular behaviour or observed outcome in our environment.

- Why is this server performing badly?
- Why does this product appeal to buyers?
- Why did this candidate lose the election?

In all of these cases, we are trying to find the most statistically significant correlation between different aspects of the data available to us. The important thing to realize about this is that you don't have to process all of the data to reach the conclusions you are looking for. In fact, you might not have to process much of it at all.

Elections: A Classic Big Data Problem

Elections are actually a good example of this. This is a classic case of Big Data that the public has been dealing with for decades. In an election, we watch the votes being

counted in real time, getting updates on a constant basis. Yet we routinely call the results of an election long before they full tally is available. Why is that? We are able to do this because we know that once a trend is established and an adequate number of precincts have reported in to support that trend, the game is essentially over. We also rely on a certain amount of historical knowledge about how various precincts have voted in the past. When we get past that magic tipping point, we know that no amount of additional data is likely to reverse the trend. I have seen some races called with only 1% of precincts having reported their results.

Imagine how quickly elections would be called, though, if all precincts were reporting their tallies in real time. We would no longer wait for urban centres to tip the trend or for partisan counties to add their voice - all voices would be heard at the same time. I suspect that, assuming a single time zone, any election whose data was hand-counted but reported in this way would produce a winner within less than an hour after the polls closed. If the counting was even partially automated, a winner would probably be declared within a few seconds.

Time To First Result

My point is this - if you crawl through data in a *statistically meaningful way* and stream back the results as you find them, you can reach conclusions almost immediately. It will no longer matter how long the query took to run. The only important metric will be the **time-to-first-result**.

Google understood this a long time ago when they reinvented searching on the web. Get the most relevant results up quickly.

Time-to-first-result has two significant advantages:

- It goes to the heart of the user's need to make decisions quickly
- It is immune to the overall size of the data being processed

Consider what this means. The efficiency of the data crawling method suddenly becomes less important. Even if the query lasts for days before fully processing all of the data, that won't matter if you can reach *meaningful* conclusions within the first 10 seconds. The fact is that most important correlations are noticeable very quickly in any statistically relevant sample. When that is not the case, *that fact is also meaningful in and of itself.*

A Simple Solution

So how simple could this problem become? A truly simple implementation would be to store data in flat files on a standard large-capacity drive and implement a query agent that uses brute force to process those files during a query. Existing hardware standards would allow that kind of solution to process thousands of records per second or more. Moreover, since you are looking at every data record in detail, every field is an equal

citizen in terms of query criteria. So long as your query agent selects the files in a statistically meaningful way - which will be specific to the problem space - and returns results as it finds them, the system will produce meaningful analysis in < 1 second.

At the risk of over-simplifying this, I am describing a system that could be implemented using a laptop and one or more off-the-shelf high-capacity drives. And since these are files on disk, the system can benefit from all of the standard HA procedures for file systems, such as RAID redundancy, fail-over, etc. Scaling and maintaining such a solution is well within the know-how of your standard Operations team.

NOTE: I include general query status (X minutes of data processed out of Y total, etc) in my definition of "results", since some queries might find nothing at all. The knowledge that a full hour's worth of data has been processed and nothing was found is an important result that the user needs to know, since it might indicate that there is nothing to be found or that the query criteria need to be adjusted.

Statistical Significance

To address the issue of statistical significance, you need to organize your data to make that selection easy for the query agent. This means finding the **major defining metric** for your problem space. Looking at our previous 3 examples:

- **Performance data:** almost every query will have a large time component to it - e.g., last hour, last 24 hours, year-to-date, October 2010, etc
- **Sales data:** again, I believe that time is a major component. Although some would argue in favour of a product ID, I believe that it would be worse to take this year's sales data for a given product and unknowingly mix it with last year's sales data. However, you might not mind if similar products had their data mixed together for the same time period.
- **Election data:** geography is more important than time in this case, since the queries are likely going to ask for analysis of a particular state, county or city as their primary goal.

When time is the defining metric, statistical relevance can be achieved by processing data from different time periods simultaneously - e.g., holidays, non-holidays, weekdays, weekends, high-volume periods and low-volume periods. When the differentiator is geography, the query agent should target different cities and regions.

Moreover, when queries include the defining metric as part of the query, they not only help the query agent process data in a statistically significant way, they also help the query agent ignore data that is clearly outside the scope of the query - a nice side effect.

Conclusion

The race for speed taking place among current solutions is simply a race to make the

overall query short enough - a nice, technically complex problem that I am sure everyone will love working on. However, no matter how fast the solutions become, there will always be larger amounts of data to manage. Chasing those performance goals will be a never-ending story.

My fervent hope is that the BigData community will start driving solutions that reflect simplicity and time-to-first-result instead of chasing after massive data centers and ever-receding performance goals.

Bio:

Jonathan Ginter works as an Architect for BMC Software, responsible for the APM product line. Jonathan has over 20 years of experience as a developer, architect and product manager for a variety of applications, from embedded real-time data processing for performance monitoring to command-and-control systems for water and power. He was a regular contributor to Coradiant's blogs on web performance and now contributes frequent articles to BMC's APM and DevOps blogs. His most recent projects have included designing a BigData system for BMC's APM solution.