Big Data

A pragmatic business problem resolution perspective

The hype cycle of big data has brought a number of single stream suggestions to bear in resolving the world's insatiable appetite for information. There are 3 primary solution focus areas that show intent in solving the big data problem, each with their own specific benefit: -

Technology – Database fragmentation, Multi thread parallelism, Logical and Physical Partitioning;

Infrastructure – Processor Componentisation, Tiered Storage Provisioning, Hardwired Distributed Storage; and

Application – Data De-Normalisation & Aggregation, Read-ahead logic, Pre Process aggregation, Performance re-engineering.

The truth be told, each specific solution in isolation provides of itself an improvement opportunity and each will bring varying degrees of success to an organisations ability to handle the big data problem if, and only if, the right questions are asked. Only through this rigorous analysis and functional de-composition can one select the right method(s) for resolution to provide long term resolution as opposed to short term 'disguising' of the issue.

Let's pause for a moment and consider the statement 'Big Data'…. Does it mean lots of the same stuff or a wide variety of lesser volumes? Does it mean a need to visualise the detail or an aggregated view or do we need to trawl/farm the data continuously with the intent of detecting deviation from the norm or threshold triggers? All are valid questions and of course all have different answers and possible solutions but few people and organisations are recognising the breadth of the problem and here-in lays the rub.

Big data is a business problem and not an IT one yet technology has replied with many solutions some of which are lightly referenced below: -

Storage engines (viz. Amazon) and the cloud and thin provisioning providing potentially limitless (reasonably priced) storage for an organisation to grow therefore available storage is typically no longer a severely limiting factor;

Rapid load and retrieval parallelism (initiated by the Teradata's of the world) and in memory storage (SAP HANA, Hadoop HDFS, Columnar DB's) is not new and allows massive volumes of data to be processed in the blink of an eye therefore data retrieval rate is no longer a limitation;

Improved buffering, dynamic index determination (query access path analysis) and internal optimisation for standard relational database technology allows databases (MS SQL, DB2, Oracle, SYBASE, MySQL and others) to read huge amounts of data seemingly instantaneously meaning that databases are less and less the limiting factor; and

Solid State Disks (SSD) provide a way around the disk IO factor therefore IO is said to be less of a problem than it used to be.

In essence, I believe that one should be asking different questions or combinations thereof focussed at the business user if you truly want to implement the correct solution to the problem. Things like 'Is time relevance of data important?', 'Are your business actions going to be affected by providing an aggregated view of the whole?', 'Is your historical data in fact relevant if the parameters under which they were collected have changed?', 'Is the level of granularity of your data of business relevance and should there be a level of roll-up?' and 'What is your data life-cycle policy and is it working for you?'. These and other data usage related discussions will be allow a data bigot (such as I) to deliver a long term sustainable solution. In the rest of this paper I look to cover the business directed questions in a high level of 'How does this affect the solution to be provided' manner.

Is Time of relevance in your data?

By asking this question we allow determination of time relevance boundaries that could impact data availability – the reality is that while the CIO's may want to boast on the amount of data that they have, it is my experience that data older than a year is in fact seldom used to influence near real-time or tactical decision making. This data could be aggregated (by month/week/year) to reduce volumes and provide the historical trending required without forcing decision engines to trawl billions of rows. Also, older data usually applies to a different set of operational conditions (less tills, fewer branches, less AMS, different product ranges etc.), subsequently, without some intense rationale around ratios for 'levelling the playing field' context is often incorrectly represented and one may be better served by shortening the historical time horizon against which queries run.

Is your data life-cycle working for you?

Similar to the time relevance question, stale data is often the clogging factor for queries that require a rapid response. It pays to have a decent data design/lifecycle and methodology for staged archival (Logical and Physical) as this will allow true analysis of how frequently you need to access the old data since you will be able to track physical access. In determining actual usage, many organisations that I have worked with have achieved results exceeding 2000% improvement in queries executed purely by applying a slightly more informed data life-cycle.

Have you analysed what you are asking the queries to do?

Many say that this is IT 101, however, never in my 30 years in the industry, never have I come across an organisation that consistently or iteratively applies this performance consideration to their solutions. The truth be told, databases are a funny beast and in reality, a single record added to a relational database may result in a query going from seconds to hours or perhaps even deadlock. There is an independent practice, developed by a former colleague called Cost Effective Computing (CEC)[i], which is loosely associated to the Zachman architecture framework and will, very quickly, pin-point areas of 'congestion' and opportunities for improvement in a problematic solution's processing path.

Is your data at the correct level of granularity?

Incorrect granularity is more often than not a legacy of a poor architectural and or business decision made with good intent at the time but little consideration for the future. A point in case is counting the number of wheels crossing a road counter and being counted as a pair. The relevance disappears if one considers that there could be large multi wheel trucks or motor cycles all counted in the same way. These counts without relevant associated context of weight, time between triggers, speed etc. serve little purpose other than to be just that, a count.

Is your data suitably isolated as to allow exploitation of parallelism?

If our solutions attempt to farm information from the transactional system in their raw form, there are some challenges that will be encountered, not the least of which is the contention with the operational requirements. In transactional systems, data is arranges to allow rapid single transaction processing to improve the user experience and seldom has consideration for parallelism or record isolation to allow multi stream retrieval. This situation gave rise to the Data Warehousing of old where data was re-arranged (typically in a monthly cycle) and the Operational Data Store concepts. I strongly urge not to knock this old school approach and consider changing the refresh cycle to allow for more current information. You will in most cases notice that with farming Big Data, there is seldom a requirement for sub-second responses and minutes will normally suffice.

In summary, the above is by no means an exhaustive list of business questions to ask and one should realise that although there may be nuggets of improvement opportunities, on-going smooth operations and information provisioning from needs to be approached differently, it is after all Data so why not start there. A more pragmatic and complete approach requires a passion for performance and data collectively, a rare trait indeed. I would urge all organisations that if you find individuals that possess these traits, hold onto them as they will be your saviours as the volumes of data continue to explode.

Author: Vaughan Nothnagel

[i] Cost Effective Computing is a framework practice developed by John Clark which overlays the Zachman framework and is used to fix poor performing application systems irrespective of the platform.