White Paper

# Big Data Analytics with Hadoop and Sybase IQ

A White Paper by Bloor Research
Author : Philip Howard
Publish date : April 2012

Big data is important because it enables you to analyse large amounts of raw data where it was not practical, either for cost or technology reasons, to do so previously.

Philip Howard

## Introduction

Big data is all the rage. While we will address the question of what big data is, the real question is how it differs from the traditional world of analytics and data warehousing that we were familiar with just a couple of years ago. Historically, data warehouses catered for data that was originally transactional. As companies started to want to analyse other forms of data, such as clickstream data, these were converted into relational format (they are not intrinsically relational) so that they could be stored and analysed with a conventional warehouse. However, as the world has become increasingly instrumented with sensors, monitors and other devices, the need to store such data has ballooned.

At the same time, companies have recognised that there is significant value to be obtained from analysing unstructured data (usually text but also video and audio) such as comments on message boards, tweets and the like. The problem with analysing this sort of data is that it is not easy to put it into relational format for analytic purposes: search is about the most you can hope for. Further, there is so much of this data that it would be prohibitively expensive to store it in a conventional manner. As a result of these considerations, companies like Google and Yahoo! developed new methods for storing and analysing this sort of data. As it turns out, these methods are also suitable for storing and analysing the instrumented data discussed in the previous paragraph.

While there are many options, the most popular method for handling big data is Hadoop. However, Hadoop has limitations: for example, it does not support ad hoc enquiries, it is not especially fast and it lacks enterprise-class data management capabilities. It is, on the other hand, much more cost-effective at managing non-relational data than a conventional data warehouse. For all of these reasons enterprises are looking at how they can combine Hadoop with a traditional data warehouse. In this paper we will not only discuss what big data is and why you might want to deploy big data analytics but also what sort of facilities you need when combining Hadoop with a traditional warehouse and, specifically, what facilities Sybase (an SAP company) offers in this respect.

## What is big data?

Big data encompasses all sorts of data, but the key question is how does big data differ from the traditional, mainly transactional, data that we are used to processing for analytic purposes? The best way to think of this difference is that the new types of data represented within big data consist of raw data. The data that we have historically been used to manipulating is transactional data. This is processed by a transactional database and then passed to a data warehouse for analysis. A lot of data used in big data has not been processed in this way—it goes straight from the originating point into the warehouse—hence the reason for calling it 'raw'. Such data can take multiple forms: it may be derived from a sensor or (smart) meter or monitoring device of some sort, it may be a database log, it may be log data from security and/or network devices, it may be images, it may be clickstream data, or it may be textual in nature, tweets or comments on social networking sites or blogs.

One caveat should be placed around this equivalence between raw and big data. In particular, the data may need to filtered and/or aggregated (in effect, cleansed) but the structure of the data has not changed in any way so it remains, in that sense, raw. There are several cases where this may apply, of which the following represent three examples:

- Suppose you are monitoring someone's heartbeat. What you are interested in is if there is a sudden spike or slump in that heartbeat, not if it continues to do what it is supposed to do. Even if you want to store that information for future analysis, what you want to store is that the heart performed on a regular basis and then spiked or slumped. In other words you would store a value together with a time period and then the exception occurred that you are interested in. Another example of this sort would be where you are using active RFID tags for tracking purposes: as long as the product or item being tracked stays in the same place (unsold if you are a retailer) you are not interested. Similar considerations would apply to logistics companies tracking vehicle deliveries: it is only an exception (the truck is in a traffic jam or has broken down) that is interesting.

- Call detail records (CDRs) in telecommunications companies are often recorded multiple times and some of these records typically contain more or less information about the particular call that has been made. This is slightly more complex than the previous examples because not only do you need to filter the data, you also want to store the most complete set of information, which may have to be garnered from multiple versions of the same CDR. A similar example occurs in capital markets where details of the same stock trade may be delivered from different platforms (Bloomberg, Reuters et al).

- A third example would be around, say, tweets. There are two cases here. The first is that if you want to conduct sentiment analysis about your company or products, say, then you are only interested in tweets that mention your company or products: all the other tweets must be filtered out. Secondly, if a tweet is re-tweeted, perhaps many times, it may make sense to treat this as a single tweet with a multiplying factor rather than as multiple individual tweets. In other words you would want to apply the same filtering and aggregation as in the previous examples.

One final point on the definition of big data is that it is often mistakenly thought that big data is only about unstructured data. This is not true. Sensor, log and clickstream data are all structured, though tweets and so forth are certainly unstructured. Sometimes, the former types of data are described as being semi-structured. This isn't true either—it is simply that the data generated from these devices is not intrinsically relational. They can be formatted in that way if that is appropriate but that is not necessary if this information is going to be treated as big data, as opposed to conventional data.

## Why should you care?

Big data is important because it enables you to analyse large amounts of raw data where it was not practical, either for cost or technology reasons, to do so previously. While the specific reasons why you should care about big data are as diverse as the types of data that constitute big data, in general big data analytics revolve around needle in haystack problems. That is, there is a wealth of data and somewhere in there is information that can help your organisation to be more competitive, more efficient and more responsive.

The issue is analogous to that of data mining 15 years ago. At that time, techniques such as market basket analysis were in their infancy and you needed professionals in white lab coats to analyse the data for you in order to establish correlations in product buying patterns; or customer segmentation, to take another example. Today, such applications of data mining are routine: over time we have built models that work to discover this sort of information on an automated basis. This is not yet the case with big data though we can expect that it will be in due course as today's professionals in white coats (data scientists) discover and automate the discovery of relevant patterns within big data.

Some companies have already identified at least some of the needles that they can usefully search for. Examples include sentiment analysis, smart metering, influencer analytics (that is, who influences whom to buy what), sensor-based applications (for example, using in-car telemetry to support motor insurance or monitoring oil rig drill-bits for pre-emptive maintenance), network analytics in telecommunications and so on. While some of the resulting analytic applications are stand-alone it is also often the case that you will wish to combine the results of big data analysis with conventional data about your customers, for example. We will further discuss this later in the paper when we consider the options provided by Sybase.

## What do you need to handle big data?

Big data differs from traditional, transactional data in a number of ways. To begin with, there is a lot more of it, so there is (an even greater) volume/storage issue. Secondly, big data is often not intrinsically relational. Some of the more structured data (for example, data from smart meters or sensors, or log data) can be readily put into relational format but unstructured data (tweets, for example) are less amenable to this approach. Thirdly, in addition to these considerations—volume and variety—big data often needs to be processed rapidly: this is referred to as velocity but it is, essentially, a by-product of the fact that there is so much data.

The volume, variety and velocity of big data also have important consequences in terms of systems that will support big data analytics. For example, traditional data warehouses scale up: that is, you have a single system to which you add more hardware. While this has advantages in terms of performance and fault tolerance, as an example it is relatively costly; and with the sheer volume of information involved with big data it is rarely practical from a cost perspective. It is therefore necessary to be able to scale out: that is, to add more small, inexpensive servers rather than extend a single large system. Ideally, both scale up and scale out should be options. In addition, depending on workload, and this is especially true in cloud-based (public or private) environments, you may need to dynamically scale in and out: the technique known as elasticity.

A further major difference between traditional and big data environments is that setting up the former is heavily dependent on defining the relevant schema, which may be normalised or it may be a star, snowflake or other specific schema. In any case, there is a considerable amount of skill and effort involved in defining this schema. By contrast, the schema required for big data (in, say, Hadoop) is virtually non-existent: you can effectively load the data and go.

In summary, storing big data within a Hadoop or similar environment has significant advantages over traditional approaches to data warehousing: it is less expensive, caters for non-relational data, scales out easily and is simple to establish and implement.

However, there is a downside. In particular, Hadoop does not have the fault tolerant capabilities that most enterprises expect: for example, both the NameNode and JobTracker within Hadoop represent single points of failure. Secondly, Hadoop has been designed to run batch-based queries and is not generally suitable for real-time query processing. Moreover, it will not perform as well as a purpose-built data warehouse. Finally, there are also issues with respect to programming. In the main, queries will be developed using the MapReduce framework. However, this requires skills in languages such as Java, which are in shorter supply and are therefore more costly than equivalent SQL capabilities. Also, of course, SQL is declarative whereas MapReduce is procedural, requiring the developer to define 'how' the query will be answered rather than relying on the database optimiser for this purpose. While there are SQL-like methods for accessing Hadoop, these are relatively primitive and lack the full functionality of ANSI standard SQL.

Because of all these caveats nobody is suggesting that you run all of your analytics (including relational analytics) within Hadoop but, because of the costs involved, it is not practical to put all of this data into your data warehouse either. Therefore we must consider how a traditional data warehouse can work alongside a Hadoop implementation in a hybrid deployment. This raises two issues, one of which is physical and one of which is logical. In the former case there is the question of the link between the two systems. You could, of course, do this using a standard ODBC/JDBC connection. This should be fine if you have only small amounts of data that you do not want to pass between the systems very often. However, where performance is an issue a lower-level API (based on PHP, for example) will be much more efficient.

At the logical level there are various alternatives: you might want to do some processing in Hadoop and then pass the results to the warehouse for further processing; you might want to be processing data in both locations and then combine the results; you might want to simply ship some data from one place to the other; and so on. Which approach will be preferable will vary according to your environment and the queries that you want to run but, ideally, the environment should be flexible enough to cater for a variety of scenarios so that you can select an approach that is most suitable for your particular project.

## How Sybase supports big data

There are two important things to note about Sybase IQ and how it can be used in big data environments. The first is with respect to the product's architecture itself and the second with regard to how it can be used in conjunction with Hadoop. We will briefly discuss each of these points in turn. For a more detailed description see the Bloor Research InDetail report into Sybase 1Q 15.4.

### Physical architecture

The architecture of Sybase IQ is illustrated in Figure 1. It is a shared everything massively parallel architecture, with every node connected to every other node in a full mesh provided by the interconnect. This reduces I/O issues and improves resilience. The only exception to the shared everything approach is that each node can have its own, local, temporary storage. The big advantage of offering shared everything, and shared disks in particular, is that you do not have to distribute your data across the various disks in use; not doing this removes what can be a significant administrative headache. It also means that you effectively have a scale out architecture as well as a scale up architecture.

Each node in the Sybase IQ environment is designated as either a read/write node or a read only node. In the case of the former, each node can be flexibly designated as a read or a write node, as required. Thus, if you are running a large overnight batch update you might want all of your read/write nodes to operate as write nodes, but have them running as read nodes during the day. In addition, you can add new nodes as needed, dynamically, so that you can scale out incrementally. Similarly, you can remove nodes dynamically so that Sybase IQ has the elasticity that is required for cloud-based and similar environments.

Nodes (servers) can be linked into a logical server, as shown in Figure 1. In addition, one logical server can 'loan' nodes to other logical servers on a scheduled basis, for example to support overnight batch loading. This approach to logical servers supports mixed query workloads because you can assign particular queries to a logical server and that query can then only use the resources available to that logical server. How many logical servers, and the number of nodes within each group, is entirely up to you, and you also have the ability to assign multiple logical servers to a single physical server if you have a large enough machine. A graphical administration tool is provided to support the creation of these logical groupings, add or remove nodes, designate read only or read/write nodes and so on.

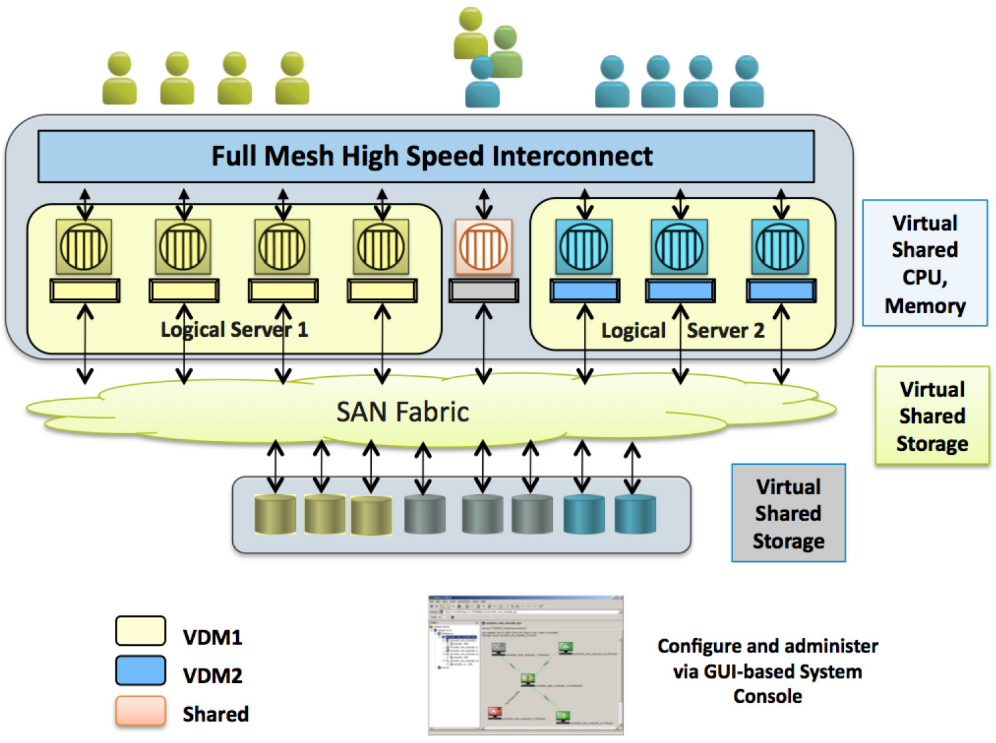
Full Mesh High Speed Interconnect
Logical Server 1
Logical Server 2
Virtual Shared CPU, Memory
SAN Fabric
Virtual Shared Storage
Virtual Shared Storage
VDM1
VDM2
Shared
Configure and administer via GUI-based System Console

**Figure 1:** Sybase IQ PlexQ architecture

## How Sybase supports big data

Each logical server can have its own logins so that specific users, departments or queries can always be directed to a designated logical server. In particular, and given that Sybase IQ supports in-database MapReduce (see below), this means that you can have a logical server dedicated to big data queries while leaving other logical servers to support traditional analytics and business intelligence. However, as we shall see (below), this is not the only way in which Sybase supports big data, it is just one of multiple options.

When a query is received the receiving node is designated as the 'leader' node and, if the optimiser decides that this query can benefit from distribution, then the other nodes to be involved in processing this query are designated as 'worker' nodes. Any node can be a leader or worker but only one node can be a leader for any particular query.

There are a number of functions specifically to support high speed loading. The first is that the product supports pipeline parallelism so that where indexes have separate data structures these can be updated at the same time as data is being loaded. In addition, bushy parallelism (where sub-tasks run in parallel) is supported.

Should any node fail, you can switch users or responsibilities to another node. Hot standby, failover and load-balancing are possible across nodes. These functions are not automated but are under the DBA's control, which allows the DBA to dynamically allocate resources based upon business needs. There are also load-balancing capabilities for Sybase IQ data loading tasks.

There is also a NonStopIQ HA-DR methodology, which typically employs a local SAN and a remote one, with either synchronous or asynchronous communications between them. The big advantage of this is not just that it provides disaster recovery but also that it eliminates the need to take the system down, even for planned outages. Note that as more and more companies adopt operational BI and embed query capability into operational applications then the warehouse increasingly becomes as mission-critical as those applications, for which you need a solution such as this.

One further feature is worth mentioning. This is that Sybase PowerDesigner provides a data warehouse reference configuration architecture whereby you input the warehouse workload and then the software estimates and prints a reference architecture for you, together with a detailed bill-of-materials.

### Big Data

Sybase IQ has in-database support for both data and text analytics; supports Time Analytic indexes (also known as date/time indexes, which are useful for applications involving smart meters, for example); has geospatial capabilities (again, useful for certain big data applications); and provides in-database native MapReduce capabilities (which includes the provision of pre-certified data mining functions). This means that you run relevant analytics against big data directly within a logical server. This operates as an in-process, user-defined function (UDF) running in the database kernel (native MapReduce). A major feature of the native MapReduce implementation is that it effectively makes MapReduce declarative rather than procedural. That is, you specify what has to be done but not how you have to do it, which is determined by the database optimiser. This is achieved because you can embed MapReduce code within a SQL statement. Moreover, you can specify what partitions you would like to work with and then the software will automatically assign processing according to the scale of the task (based on the disjoint sets being used).

For out-of-process functions, the Sybase API can support programmatic tasks that have been written externally (in Hadoop, for example) and yet bring the results into Sybase IQ for deeper processing—on the fly.

However, it will often be the case that you want to deploy Hadoop for your haystacks and use it in conjunction with Sybase IQ for your needles. This is because Hadoop is much less costly. However, it is not suitable (or at least not yet) where compliance or security are issues, or where exceptional performance is required or if you need to perform ad hoc queries. So there will often be a requirement to use Hadoop in conjunction with your data warehouse, for example where you want information from Hadoop to participate in ad hoc queries, even though those are not supported by Hadoop per se. In order to enable this, Sybase offers four different ways of integrating with Hadoop, as illustrated in Figures 2–5.

## How Sybase supports big data

The first method, illustrated in Figure 2, shows client side federation—results from Sybase IQ and Hadoop being collated by the Toad product from Quest Software (a partner of Sybase). For example, in telecoms you might use Sybase IQ to provide aggregated customer loyalty data and Hadoop with aggregated network utilisation data; and then Quest Toad for Cloud can bring the data together from both sources, linking customer loyalty to network utilisation or network faults such as dropped calls. This method is better suited for pre-aggregated data from different data domains.

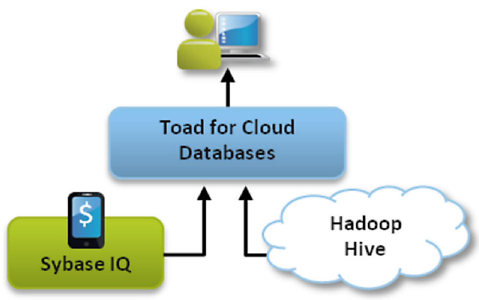

**Figure 2:** Client-side federation between Sybase IQ and Hadoop

The second method involves straightforward ETL processes, moving HDFS (Hadoop distributed file system) subsets into Sybase IQ for analysis. Sqoop is shown here but you could use the data integration tool of your choice, though it may not have been optimised for use with Sybase IQ. An example use case is in ecommerce, where clickstream data from weblogs stored in HDFS and outputs of MapReduce jobs on that data (to study browsing behaviour), are loaded into Sybase IQ via an ETL process. The transactional sales data in Sybase IQ is joined with the clickstream data to understand and predict customer browsing to buying behaviour. The HDFS brought into Sybase IQ is stored in Sybase IQ column store and appropriate schema and the HDFS data is treated as par with other data stored in Sybase IQ.

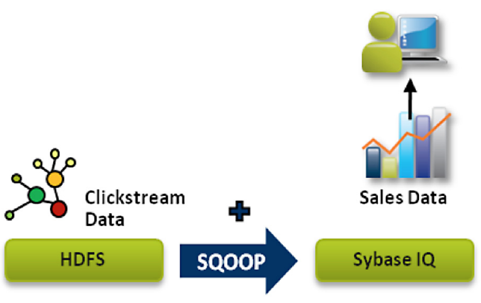

**Figure 3:** ETL from Hadoop to Sybase IQ

Thirdly, you can federate data from Hadoop into Sybase IQ. Here, HDFS subsets are materialised in Sybase IQ as in-memory virtual tables. This process can be triggered on the fly as a part of a query. An example here would be in retail, where point of sale (POS) detailed data is stored in Hadoop. Sybase IQ fetches the POS data at fixed intervals from HDFS from specific hot-selling SKUs, combined with inventory data in Sybase IQ to predict and prevent inventory 'stockouts'. In this scenario, the HDFS data is not stored in the Sybase IQ column store, instead, it is transient: that is, it is available only for the life of the query that triggered the HDFS data fetch.

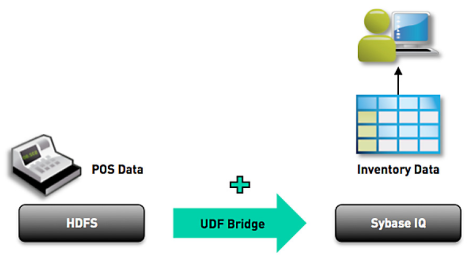

**Figure 4:** Federating data from Hadoop into Sybase IQ

Finally, you can federate Hadoop processes (jobs) into Sybase IQ whereby you trigger MapReduce tasks in both Sybase IQ and Hadoop and then join the results on the fly. You might use this approach in the utility sector, with smart meter and smart grid data combined for load monitoring and demand forecasting. Smart grid transmission quality data (multi-attribute time series data) stored in HDFS can be computed via Hadoop MapReduce jobs triggered from Sybase IQ and combined with smart meter data stored in Sybase IQ to analyse demand and workload. Similar to the previous scenario, the results of MapReduce tasks executed in Hadoop are not stored in the Sybase IQ column store, instead, it is transient: that is, it is available only for the life of the query that triggered the MapReduce job.

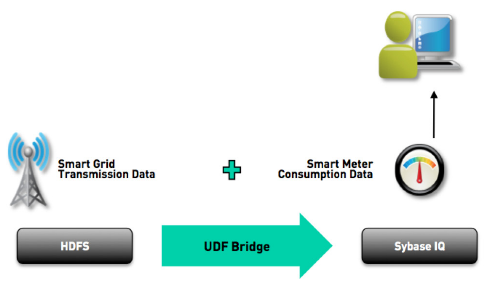

**Figure 5:** Federating Hadoop processes into Sybase IQ

## Conclusion

The Hadoop market is immature and still evolving. Nevertheless, it is clear that Hadoop, or one of its variants, or one of its competitors, has a major role to play in reducing the costs of analysis against instrumented and/or unstructured data. However, as we have noted, Hadoop has significant drawbacks when compared with traditional data warehouses. It therefore makes sense to combine the two and Sybase offers a comprehensive and impressive set of different ways of achieving this. The importance of having multiple integration options should not be underestimated: even within a single company one would expect different queries or query types to be most suitable to different approaches, whether entirely in-warehouse, in-Hadoop or via one or other hybrid approach.

### Further Information

Further information about this subject is available from
http://www.BloorResearch.com/update/2126

## Bloor Research overview

Bloor Research is one of Europe's leading IT research, analysis and consultancy organisations. We explain how to bring greater Agility to corporate IT systems through the effective governance, management and leverage of Information. We have built a reputation for 'telling the right story' with independent, intelligent, well-articulated communications content and publications on all aspects of the ICT industry. We believe the objective of telling the right story is to:

- Describe the technology in context to its business value and the other systems and processes it interacts with.
- Understand how new and innovative technologies fit in with existing ICT investments.
- Look at the whole market and explain all the solutions available and how they can be more effectively evaluated.
- Filter "noise" and make it easier to find the additional information or news that supports both investment and implementation.
- Ensure all our content is available through the most appropriate channel.

Founded in 1989, we have spent over two decades distributing research and analysis to IT user and vendor organisations throughout the world via online subscriptions, tailored research services, events and consultancy projects. We are committed to turning our knowledge into business value for you.

## About the author

Philip Howard
Research Director - Data Management

Philip started in the computer industry way back in 1973 and has variously worked as a systems analyst, programmer and salesperson, as well as in marketing and product management, for a variety of companies including GEC Marconi, GPT, Philips Data Systems, Raytheon and NCR.

After a quarter of a century of not being his own boss Philip set up his own company in 1992 and his first client was Bloor Research (then ButlerBloor), with Philip working for the company as an associate analyst. His relationship with Bloor Research has continued since that time and he is now Research Director focused on Data Management.

Data management refers to the management, movement, governance and storage of data and involves diverse technologies that include (but are not limited to) databases and data warehousing, data integration (including ETL, data migration and data federation), data quality, master data management, metadata management and log and event management. Philip also tracks spreadsheet management and complex event processing.

In addition to the numerous reports Philip has written on behalf of Bloor Research, Philip also contributes regularly to IT-Director.com and IT-Analysis.com and was previously editor of both "Application Development News" and "Operating System News" on behalf of Cambridge Market Intelligence (CMI). He has also contributed to various magazines and written a number of reports published by companies such as CMI and The Financial Times. Philip speaks regularly at conferences and other events throughout Europe and North America.

Away from work, Philip's primary leisure activities are canal boats, skiing, playing Bridge (at which he is a Life Master), dining out and walking Benji the dog.

## Copyright & disclaimer