Title: The Power to enable Big SQL on Power

This document can be found on the web, www.ibm.com/support/techdocs

Under the category of “White Papers.

Tecdoc Number:

Version Date: In process

Draft

Systems Group

Chuck Gray Architect

"Analytics is a set of tools that enable solutions for the enterprise to simplify and align their information assets, technologies, and architecture in a way that support the analytics framework and delivery of knowledge about the enterprise for decision support. This allows the enterprise to improve and apply analytics across their business to drive revenue and margin increases, and reduce both customer churn and information management costs. IBM supports the information mining with one of the largest integrated offerings of services, hardware, and software. It enables access to social media, unstructured data, and structured content. We blend, manipulate, and glean from the enterprise Information Assets, intelligence and insight to direct our enterprise decisions. We can enhance this by blending Traditional and Cognitive computing paradigms." Chuck Gray

1

BigSQL value enabled with Power

1.1 Architecture

IBM's Big SQLis an SQL solution that moves away from traditional MR frameworks. It introduces a shared-nothing parallel database architecture that leverages state-of-the-art relational database technology from IBM. Big SQL is not like other SQL like systems over Hadoop which pulls data into the edge service. Big SQL uses the data as it remains on the HDFS cluster.

IBM's Big SQLdoes not impose any relational database management system (RDBMS) structure or restrictions on the layout and organization of the data, which helps maintain the flexibility of Hadoop. The database infrastructure provides a logical view of the data by allowing storage and management of metadata and a view of the query compilation, plus the optimization and runtime environment for optimal SQL processing.

IBM's Big SQLexploits sophisticated query rewrite transformations targeted at complex nested decision support queries.

1.1.1 Parallel design

The general architecture of Big SQL 3.0

A user configures the server so that user applications connect on a specific node. This is where SQL statements will be routed.

Commonly referred to as the coordinating node, this is where SQL statements will be compiled and optimized to generate a parallel execution query plan. A runtime engine then distributes the parallel plan to worker nodes and orchestrates the consumption and return of the result set.

IBM's Big SQLdoes not require a worker node on every HDFS data node. It can operate just as efficiently in a configuration where it is deployed on a subset of the Hadoop cluster. The flexibility is out standing.

1.2 Execution

Once a worker node receives a query plan, it dispatches special processes that know how to read and write HDFS data natively.

IBM's Big SQLutilizes native and Java open source based readers and writers that are able to ingest different file formats. To improve performance, the Big SQL engine

pushes predicates down to these processes so that they can, in turn, apply projection and selection closer to the data.
Another function performed by these processes is the transformation of input data into an appropriate format for use inside Big SQL.

This transformation allows for intermediate data to be loaded into the Big SQL engine memory for additional processing. IBM's Big SQLutilizes a variety of in-memory buffer pools to help optimize query and utility-driven processing.

In addition to distributed MPP processing, each worker node further parallelizes execution both inside the Big SQL engine and on the HDFS side. Big SQL automatically determines the best degree of intra-query parallelism dynamically, per query, based on available resources.

2

Big SQL Value to the enterprise in using information as an asset

When considering SQL, the most fundamental question is.

What is the right tool for the job?

As enterprises start using Hadoop as a central data repository for all data originating from sources as varied as operational systems, sensors, smart devices, metadata and internal applications SQL processing becomes a very interesting choice, as it supports interactive queries that require a few seconds (or even milliseconds) of response time.

Today most enterprise data management and analytical tools rely on SQLfor interactive query execution. SQL as an access engine provides processing benefits from decades of research, usage experience and optimizations. This tied to the SQL skills pool far exceeds that of the newer open source tools. The general-purpose processing framework and tools of Hadoop may still be appropriate for ad hoc analytics.

Right now, users have an excellent opportunity to jump into the world of big data and Hadoop with the introduction of Big SQL 3.0. In terms of ease of adoption and transition for existing analyticworkloads, it delivers uniquely powerful capabilities.

2.1 Big SQL security

This is done with out sacrificing security; it introduces capabilities that are found in traditional relational data warehouses. In addition, it can access and join with data originating from heterogeneous federated sources.

2.2 BigSQL scaling

IBM's Big SQLfrom IBM represents an important step for Hadoop being an enterprise enabled platform.

It is an alternative to MapReduce with a massively parallel processing (MPP) SQL engine. The MPP engine deploys directly on the physical Hadoop Distributed File System (HDFS) cluster. A key difference from other SQL offerings on Hadoop is that this engine actually pushes processing down to the same nodes that hold the data. Because it natively operates in a shared-nothing environment, it does not suffer from limitations common to shared-disk architectures, caused by the need to move "shared" data around).

IBM's Big SQL introduces an evolutionary step in low-latency parallel execution infrastructure that is able to access Hadoop data natively for reading and writing. It extends SQL 2011 language support with broad relational data type support, including support for stored procedures. Its focus on comprehensive SQL support translates into industry-leading application transparency and portability. It is designed for concurrency with automatic memory management and comes equipped with a rich set of workload management tools. Additional extensions include scale out parallelism to dozens of data processing nodes and scale up parallelism to hundreds of cores.

2.3

Example of Value add to the Big Data business enabled by IBM

IBM's Big SQL utilizes the Apache Hive database catalog for table definitions, location, storage format and encoding of input files. This means it is not restricted to tables created and/or loaded via Big SQL. As long as the data is defined in the Hive Metastore and accessible in the Hadoop cluster, Big SQL can get to it.

Some of the metadata from the Hive catalog is stored locally by Big SQL for ease of access and to facilitate query execution. The Big SQL catalog resides on the headnode. This allows for local access during query compilation and optimization. Query plans for stored procedures are stored as packages in the catalog for future re-execution or execution by other client application which helps save the recompile cost. These packages are managed automatically and invalidated, for example, whenever a topology change is detected in the Big SQL cluster.

Scheduler

The fundamental component in IBM's Big SQL is the scheduler service. This scheduler acts as the Big SQL liaison between the worlds of SQL and Hadoop. It provides a number of important services.

Hive Metastore interface so when queries are compiled and optimized, the Big SQL catalog is queried for basic information, such as column names and data types. Further optimizations can be achieved by understanding the physical layout of the data and associated statistics. The Big SQL engine will contact the scheduler at various stages of execution to push down and retrieve critical metadata.

Partition elimination

IBM's Big SQLsupports Hive partitioned tables, so the compiler will push predicates down to the scheduler to eliminate partitions that are not relevant for a given query.

Scheduling of work

The scheduler maintains information about where and how data is stored on Hadoop. Big SQL uses that information to help ensure that work is processed efficiently, as close to the data as possible. In traditional shared-nothing architectures, one of the tasks of the compiler-optimizer is deciding which nodes to involve in a query. This is usually achieved through control of data partitioning. Big SQL, however, does not impose any database-style partitioning. The system instead supports the concept of scattered (or dynamic) partitioning, in which no assumptions are made on what nodes the data should be accessed from. This allows Big SQL not only to assign work where the data resides locally but also to take into account worker node load and other hints for query plan distribution.

Concurrency and workload management

Enterprise solutions over Hadoop are limited to experimental propositions unless they can handle real-life concurrent business intelligence (BI) and analytics workloads. This is where IBM's Big SQLoffers a novel architecture by coupling the flexibility of HDFS with mature, multi-tenant and proven data warehouse technology from IBM.

The problem of memory management has been pervasive in Hadoop ecosystems in part because it is less mature than other frameworks.

Much of the tuning exercise in Hadoop becomes an intensive trial and error process: reacting to out-of-memory exceptions, adjusting service properties, rerunning queries. It is a very workload-specific exercise, one that requires adjustment as the workload changes. Such as one workload might need more mapper memory while another might have higher reducer memory requirements.

IBM's Big SQL introduces automatic memory tuning. At installation, the user specifies how much memory is available for the system to use, and Big SQL optimizes execution within these limits.

There are two key aspects to this fully autonomic behavior. The first is it use memory efficiently and secondly prevent out-of-memory exceptions. This is particularly important when the workload involves multiple concurrently running applications.

The addition of the Sub queries to Big SQL

In SQL, subqueries are a way of life. They allow for much more flexibility in expressing the query, improved readability and, in some cases, improved performance. Yet many SQL-on Hadoop solutions have significant limitations for when and where sub queries may be used (if they can be used at all) and how deeply they may be nested.

The Big SQL query rewrite engine aggressively decorrelates sub queries whenever possible, taking into account additional table metadata, such as null ability and constraint information, to help ensure the most high-performing query execution possible.

Table expressions

Beyond the ability to simply query tables, Big SQL provides a number of table expressions that may be used as sources of data for queries.

These restrictions exist because such join operations are difficult to implement efficiently on such a system as Hadoop, in which data is randomly scattered across the nodes of the cluster and true indexes are generally not present. However, there are certainly classes of queries in which one of the two tables being joined is small or the filtering criteria on the table results in a small data set, in which case these queries can be effectively executed.

Big SQL provides the ability to perform all valid SQL standard join operations.

Join strategies

Beyond the ability to simply express joins using all of the available SQL standard join constructs, Big SQL provides for a number of strategies for implementing joins, automatically selecting the most appropriate join for a query based on available table statistics.

These strategies include hash join, merge join and nested loop join. Big SQL implements a parallel hybrid hash join algorithm that is designed to be optimized for clusters of highly symmetric multiprocessor (SMP) systems and can adapt its behavior to the resources available in the system. Its memory-conscious algorithm dynamically balances the workload when processing buckets of data stored in a disk. Buckets are scheduled depending on the amount of memory available, and the processes assigned to each bucket are dynamically chosen. These results in an efficient use of the shared memory pool, which saves I/O by avoiding hash loops, and helps minimize memory contention with a smart balancing of processes across buckets in the memory pool.

Windowing and analytic functions

Hadoop is a system designed for large-scale analytics. One of the key SQL features created to address analytic problems is SQL OLAP functions. Big SQL supports all of the standard OLAP specifiers, including the following:

•• LEAD
•• LAG
•• RANK
•• DENSE_RANK
•• ROW_NUMBER
•• FIRST_VALUE
•• LAST_VALUE
•• RATIO_TO_REPORT
•• Standard aggregate functions

Additional query features

The following is a list of additional significant features provided by Big SQL 3.0.

Grouping sets, cube and rollup

A grouping set effectively allows for multiple GROUP BY expressions to be performed on a query, with CUBE and ROLLUP operations indicating how to aggregate data within subgroups.

Routines

Routines are application logic that is managed and executed within the Big SQL database manager, allowing for encapsulation and execution of application logic at the database rather than within the application. Beyond simply centralizing the application logic within the database, routines also allow for fine grained access control. Database users can be granted permission to execute routines that access data sources that the user would not otherwise be capable of accessing. Additional application specific security checks may be implemented within the routines in addition to the normal SQL security enforcement.

•• CORRELATION
•• COVARIANCE
•• STDDEV
•• VARIANCE
•• REGR_AVGX
•• REGR_AVGY
•• REGR_COUNT

•• REGR_INTERCEPT
•• REGR_ICPT
•• REGR_R2
•• REGR_SLOPE
•• REGR_SXX
•• REGR_SXY
•• REGR_SYY
•• WIDTH_BUCKET
•• VAR_SAMP
•• VAR_POP
•• STDDEV_POP
•• STDDEV_SAMP
•• COVAR_SAMP
•• COVAR_POP

Big SQL supports the following types of routines:

•• Scalar functions – A function that takes zero or more arguments and returns a single value. This value may be of any SQL data type.

•• Table functions – A function that takes zero or more arguments and returns rows of values Table functions may be used in the FROM clause of SELECT statements.

•• Procedures – A procedure encapsulates application logic that may be invoked like a programming subroutine, including both input and output parameters. Procedures may include flow-of-control logic and error handling and may invoke static or dynamic SQL statements.

Development of a routine

Routines may be developed in a number of popular programming languages:

•• SQL – Routines may be developed and managed entirely in SQL PL. Being a high-level language, SQL PL allows for rapid application development because the routines are executed entirely within the Big SQL runtime engine.

•• Java – Java routines provide all of the flexibility of the Java programming language, along with access to the enormous wealth of available Java libraries. Java routines are ideal for interacting with external data sources or accessing complex functionality that would not otherwise be accessible via SQL routines. In addition, the Java virtual machine (JVM) sandbox provides increased reliability and security over routines implemented in C/C++.

•• C/C++ – C/C++ routines allow for very low-level, efficient execution and direct access to native development libraries.

•• Mixture – A routine developed in one language may invoke a routine in another language. A SQL-bodied stored procedure may execute a SELECT statement in which a C++ function is applied to a column in the result set of the query.

Routines and security

Beyond the basic ability to grant individual users, groups or roles the ability to manage or execute routines, the privileges of the routine itself may be finely tuned. Specifically:

•• Static SQL within the routines execute with the privileges of the user that defined the routine.

•• Dynamic SQL within the routines is invoked under the privileges of the user invoking the routine, by default, restricting the risk of accidental privilege escalation, such as SQL injection attacks.

If desired, this behavior may be modified to allow for execution of dynamic SQL under the rights of the package or routine owner.

Interoperability of routines

For all languages in which routines may be developed SQL PL, Java or C/C++, Big SQL shares most common language features and functionality with IBM DB2 database software, meaning that the same routine developed for Big SQL can be easily migrated between systems with a few (or no) modifications.

In addition, Big SQL provides standards-compliant APIs. By adhering to the standard APIs, such as SQLJ routines for Java and PARAMETER STYLE SQL or PARAMETER STYLE GENERAL routines for C/C++, routines that are developed for Big SQL can be shared across any other database platform implementing the standard.

Data ingestion

In addition to the existing Hadoop mechanisms for moving data, Big SQL provides a LOAD statement that is more familiar to the SQL user. Users can load flat-file data into Big SQL, and the SQL LOAD HADOOP statement allows a table to be populated from external data sources via standard JDBC drivers. The data loading process is coordinated in parallel, with multiple nodes of the cluster processing ranges of values from the source system. This allows for parallelism within the Hadoop cluster as well as the ability to take advantage of any partitioning capabilities of the source from which data is being extracted.

Last thought in the section

This is intended to highlight some of the features of analytic enablers on Big Insights supported by Power that uses existing skills that may exist within your enterprise.

3

Summary

3.1

Value of Power as it enables Hadoop

Analytics is often thought of as answering questions using data, but it involves more than simple data queries. Analytics involves the use of mathematical methods to develop insight from data to information and then wisdom.

The face of analytics is rapidly changing based on how to drive value from data assets. It is moving from a write weighted model to a balance CPU and I/O. The way Power is design to support the Big Data work load is spot on. The large number of threads and core speeds with Hadoop design is like having a developed for purpose platform. This IBM design is being developed into a support design that will leverage the hardware today into the enhancement in design going forward. The commodity model where you take off-the-shelf components to build your deployment worked yesterday for non-business critical deployments. Today the decision support is impacting day to day business decisions. If a drive dies in the commodity design after 300 seconds, (tunable value) the Hadoop scheduler will kick off a rebuild of blocks that were lost on that node. These blocks will now be building over your data network. This will impact your delivery of user requested analytics. To minimize this storm, we define storage and compute to maximize availability RAS features of all the components in the Power platform.

The following information is about types of analytics and road to the future. Today

Predictive analytics encompasses a variety of statistical techniques from modeling, machine learning, and data mining that analyze current and historical facts to make predictions about the future,or otherwise unknown, events.
Decision models describe the relationship between all the elements of a decision — the known data (including results of predictive models), the decision, and the forecast results of the decision — in order to predict the results of decisions involving many variables. These models can be used in optimization, maximizing certain outcomes while minimizing others. Decision models are generally used to develop decision logic or a set of business rules that will produce the desired action for every customer or circumstance.

Now and into the near future (High Compute and I/O)

Prescriptive analytics automatically synthesizes big data, multiple disciplines of mathematical sciences and computational sciences, and business rules, to make predictions and then suggests decision options to take advantage of the predictions. The first stage of business analytics is descriptive analytics, which still accounts for the majority of all business analytics today.

Descriptive analytics answers the questions what happened and why did it happen. Descriptive analytics looks at past performance and understands that performance by mining historical data to look for the reasons behind past success or failure. Most management reporting - such as sales, marketing, operations, and finance - uses this type of post-mortem analysis.

Social Media Analytics – thisis outside data sourcing that allows the business to listen to public opinion or sentiment and would relay heavily on entity methods.

Visual analytics – this is the process of visually representing the results of your mining to the consumer by filtering the results using with KPIs = Key Performance Indicators and KRI = Key Result Indicators. This is expanding to where complex graphing is executed and results homed in Big Data repository.

The future of analytics is now on Power and the direction for the future of analytics around Big Data.

Entity Analytics – thisfocuses on sorting through data while grouping together data that relates to the same entity. This information under normal views would not have the association one to another.

Cognitive model is one of the most common distinctions in the literature on cognitive style is between analytic and holistic styles. Analytic thinking involves understanding a system by thinking about its parts and how they work together to produce larger-scale effects. Holistic thinking involves understanding a system by sensing its large-scale patterns and reacting to them.