MPP SQL Engines: Architectural Choices and Their Implications on Benchmarking

Ravi Chandran[1](✉), K.T. Sridhar[2], and M.A. Sakkeer[2]

[1] XtremeData, Inc., Schaumburg, IL, USA
ravi@xtremedata.com
[2] XtremeData Technologies Ltd., Bangalore, India
{sridhar, sakkeer}@xtremedata.com

Abstract. One approach to big data benchmarking is to study the implications of the underlying computing platform, both hardware and software. The software stack for big data analytics comprises a suite of technology solutions including MPP SQL Engines. This paper provides an overview of the architectural choices in the underlying hardware infrastructure, the architectural choices in the design of parallel SQL Engines, and the implications of both on benchmarking. A benchmark suite developed at XtremeData for internal engineering use is also described.

Keywords: MPP SQL · Cloud object-store · Benchmark

1 Introduction

The inflection point in computing solutions, initiated by the convergence towards virtualized-x86-Linux hardware and public cloud offerings, has been accelerated by big data [1]. The rapid growth in big data volumes is orders-of-magnitude larger than the growth in the traditional enterprise operational data. This has created tremendous challenges, since no enterprise can afford to ignore the value of big data analysis. Big data has also been democratized: where it was once the domain of global enterprises, now businesses of all sizes have easy access to and can benefit from big data.

Data volumes are growing much faster than traditional IT budgets can possibly increase, so incremental evolution of legacy technologies cannot meet the demands. Market pressures have spawned a wide range of new technologies in recent years. Many of these technologies are narrowly focused on specific applications. The focus of this paper is on structured datasets, essentially the traditional ecosystem of Data Warehouse-Data Marts (DW-DM) extended to include new big data sources. Unstructured or loosely-structured data [2], such as text files, documents [3, 4], audio and video, have their own specialized solutions. The challenge of structured big data analytics is to deploy solutions that scale affordably in the context of loading, joining, aggregating, analyzing and reporting on billions of records quickly enough to meet business demands.

Over the past few years, the market has seen the rise of many "NoSQL" [5] solutions that are still evolving today; but the general consensus emerging is that Hadoop-like solutions need to be augmented by a full-featured SQL engine that can operate at large scale. SQL skills are ubiquitous in the market and a MPP (Massively Parallel Processing) SQL engine that hides the complexity of parallel execution is an attractive solution that can be immediately deployed.

In the context of benchmarking big data systems, it might be useful to study the architecture of the underlying infrastructure: both hardware and software. The focus of this paper is restricted to only one particular component of the big data software stack: a MPP SQL engine. In the following sections, the hardware infrastructure choices are first described, followed by the architectural choices for the MPP SQL engine. This is followed by a discussion of the implications on benchmarking.

2 Hardware Infrastructure

All components of a big data solution need to meet one common criterion; the ability to be deployed on today's converged, sharable hardware infrastructure. This converged architecture applies equally to public clouds, like Amazon Web Services (AWS) [6] and Google Compute Engine (GCE) [7], and to private clouds within the enterprise data center. Hardware infrastructure in today's converged data center comprises three layers: server-network-storage.

2.1 Server

By definition the server layer is a virtualized-x86-Linux machine; other CPUs and operating systems are not significant contenders. Given this, there is very little differentiation in terms of hardware performance or functionality between vendors of x86 machines: all are more or less equal.

2.2 Network

The network layer offers much more interesting choices. Unlike the server layer, the performance of the network tends to be more of a step function rather than continuous: 1 or 10 gigE, DDR[1] or QDR[2] InfiniBand. Network switches also have a range of capabilities that can significantly impact data-intensive processing: bi-section bandwidth, lane throttling/bandwidth guarantees [8]. Today the most prevalent network infrastructure is based on 10 gigE.

2.3 Storage

The storage configuration for data-intensive processing has diverged away from the traditional SAN[3]/NAS[4] and has converged to two choices: "object store" and

[1] Double Data Rate.

[2] Quad Data Rate.

[3] Storage Area Network.

[4] Network Attached Storage.

"distributed filesystem". This shift was driven by the need for scalability in MPP systems, which shared solutions like SAN/NAS cannot provide. The new offerings in storage are exemplified by OpenStack [9]: Cinder (block store, filesystem) and Swift (object store). These offerings are very similar to those offered by public cloud providers, like AWS and GCE. Underlying both these solutions is the same hardware infrastructure: a cluster of machines with locally-attached storage. This cluster is typically built with the same x86-Linux machines user in the server layer, but other choices may be also be employed in the future, such as low-power ARM CPUs. The storage medium itself uses rotating magnetic disks, flash memory or some combination.

The block store could be local storage attached to each Linux server or a distributed solution spread across multiple servers. The object store is a scalable redundant storage system that offers the features of a traditional SAN/NAS system, but in a much more scalable fashion, leveraging commodity Linux clusters.

This combination of object and block store enables a re-definition of the traditional Data Warehouse architecture. Figure 1 shows a simplified block diagram of such a traditional DW, comprising two major components: a reliable storage layer and an SQL querying layer. In today's cloud architecture, storage can be decoupled from the SQL querying engine, as shown in Fig. 2. The reliable storage layer is provided by the cloud object-store. The SQL querying layer is configured as a set of computing resources in the cloud, populated with data from the object store. The populated data can reside either in a cloud block store or locally in storage associated with the computing resources, during the lifetime of the SQL querying engine. When the engine is shutdown, changed data can be uploaded back to object store, and re-populated on a re-start.

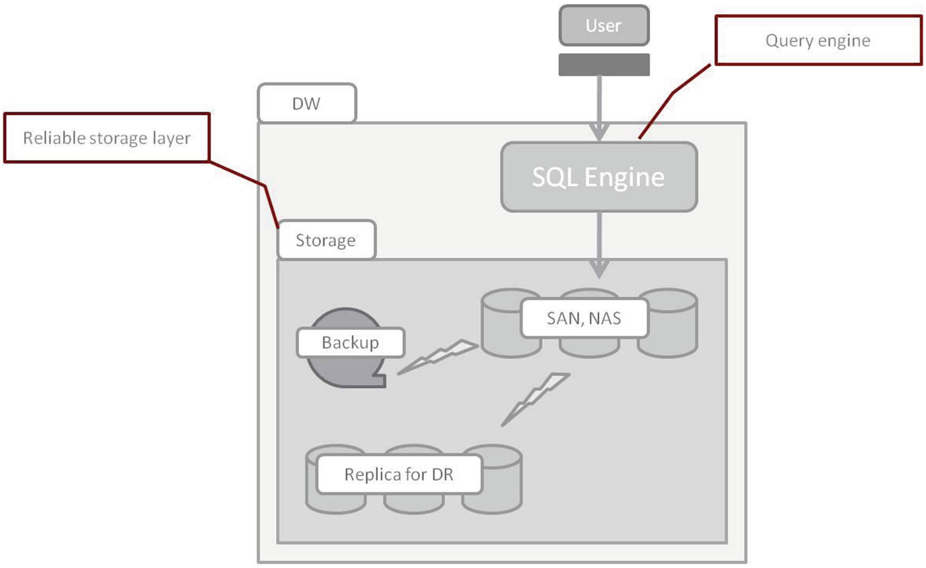

Fig. 1. Architecture of traditional Data Warehouse

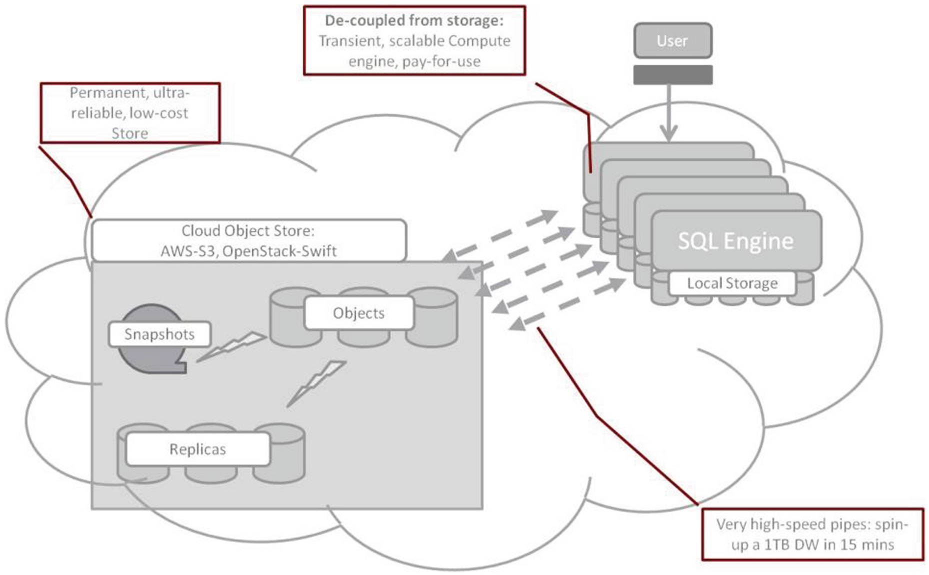
De-coupled from storage: Transient, scalable Compute engine, pay-for-use
User
Permanent, ultra-reliable, low-cost Store
Cloud Object Store: AWS-S3, OpenStack-Swift
SQL Engine
Local Storage
Objects
Snapshots
Replicas
Very high-speed pipes: spin-up a 1TB DW in 15 mins

Fig. 2. New Data Warehouse architecture enabled by the Cloud

This decoupling enables the storage and querying layers to scale independent of each other. This is especially so, if the SQL querying engine is designed to be parallel, shared-nothing in its architecture (MPP), as discussed in the next section.

As illustrated, the storage offerings in the cloud (public or private) enable a new generation of scalable big data platforms, such as XtremeData's "elan" solution (**el**astic **an**alytics).

3 MPP SQL Engines

The advent of big data spurred a drive towards scalable, parallel SQL database systems. As illustrated in Fig. 3, the major components of any database system are: Data Dictionary (catalogs and all meta-data), SQL Execution Engine and the Storage Engine. The challenges to increasing the scalability and performance of an open-source database system like postgreSQL [10], can be addressed at several different levels,

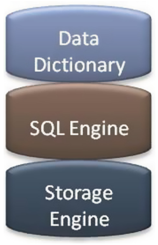

Fig. 3. Components of a DB Engine like postgreSQL: starting point for MPP SQL

leading to implementations that may be characterized as "Sharded", "Federated" or "True-MPP", as described below.

Early implementations were "Sharded": tables were broken up into shards and each shard was stored in a separate SQL database instance, typically an open-source engine like MySQL and postgreSQL. Later, commercial vendors incorporated the sharding process in their offering, leading to "Federated" solutions, where the end-user did not have to directly handle the sharding, as shown in Fig. 4.

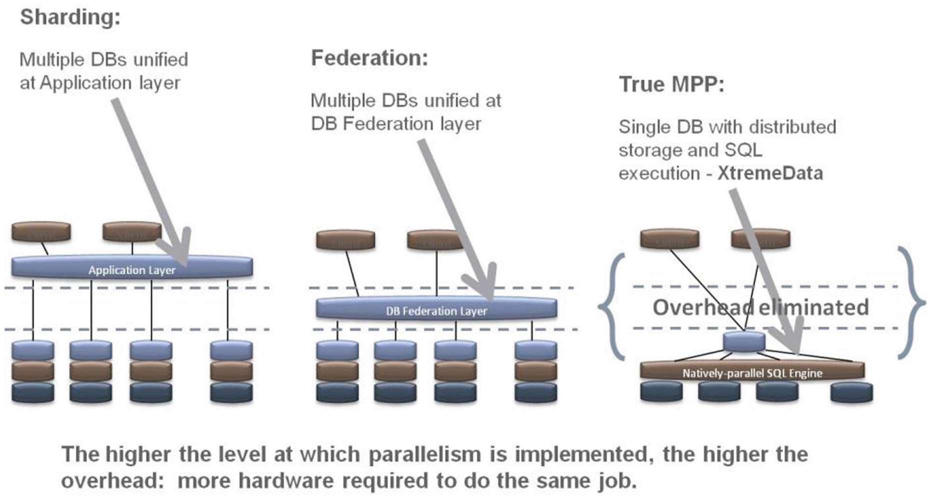
Sharding:
Multiple DBs unified at Application layer
Application Layer
Federation:
Multiple DBs unified at DB Federation layer
DB Federation Layer
True MPP:
Single DB with distributed storage and SQL execution - XtremeData
Overhead eliminated
Natively-parallel SQL Engine
The higher the level at which parallelism is implemented, the higher the overhead: more hardware required to do the same job.

Fig. 4. Architectural approaches to MPP SQL

Almost all commercial offerings in the market today are federated systems, with multiple complete instances of postgreSQL running under the hood. From a benchmarking viewpoint, this is worth examining a little more closely, since the implications of multiple complete instances are very significant.

Within postgreSQL, the core SQL execution engine (although evolving continuously) is single-threaded. This has a direct impact on the usage efficiency of the underlying hardware, since the x86 CPU's path is in the direction of many cores. A single-thread can only use one core. Therefore to leverage the many-core CPUs (2x8-core CPUs per server is available today, more in the future), vendors of federated solutions recommend one instance of postgreSQL per core. This recommendation was reasonable even a few years ago, when there were only 2 or 4 cores per server. But with 16 cores today and probably 32 or 64 within a year or two, this recommendation rapidly degrades into unmanageable complexity.

A standard data center rack (42U) can hold 16 servers: this implies 256/512/1024 cores per rack: 256/512/1024 complete independent instances of postgreSQL in each rack is not a reasonable solution. This is especially so in the context of big data analytics, where joins and group-aggregates across multiple large tables are routine operations. These operations necessarily (in the general case, where data subsets cannot be guaranteed to be co-located in each node) require N:N exchange of data between all N nodes in the cluster. 'N' in this case refers to the number of logical nodes or database instances (256/512/1024) rather than physical servers (16 per rack). This is a serious limitation of federated designs, and any big data benchmark should be capable of quantifying the impact on performance and scalability.

A more manageable architecture is to limit the number of logical nodes, and create a natively-parallel, multi-threaded core SQL execution engine. A few vendors, including XtremeData, have taken this approach.

4 Implications for Benchmarking

Big data benchmarking of MPP SQL engines has to take in consideration the implications of both the hardware infrastructure and the architecture of the SQL engine.

4.1 Hardware Infrastructure

A benchmark should produce quantified results that can be used to assess the capability of the underlying hardware:

- Servers:
  - CPU capability (# of Cores)
  - Memory size and bandwidth
  - Network bandwidth
- Storage:
  - # of tiers (Memory, Flash, Disk)
  - size and bandwidth for each tier
- Network:
  - Topology: point-to-point latency
  - Switch bisectional bandwidth

Many tests are available in the industry to assess each of these hardware parameters independent of big data workloads. The results of these tests should also be reflected in a big data benchmark. For example, if CPU capability is poor (low clock frequency, small number of cores), the big data benchmark suite should include tests that clearly highlight this fact. In the benchmark we developed (described in later section), there are a series of queries that perform a full table scan of a single table, but with increasing complexity in the associated computation. This sets a baseline time for full table scan (typically limited by storage bandwidth) and assesses CPU capability.

4.2 SQL Engine

Assessing the SQL engine's capabilities for big data analytics requires tests that span these operations:

- Functionality:
  - SQL language support
  - Partitions, Indexes, Cursors, Window Functions
- Performance:
  - Parallel load from external source
  - Single table tests:
    - Scan-Filter-Complex compute
    - Group-Aggregate
    - Window Functions
  - Multi-table tests:
    - Joins
    - Joins + all of the above
  - Table creation within DB (Create Table As): for data-intensive, iterative processing
- Scalability:
  - Scale DB size while holding system size constant
  - Scale system size holding DB size constant

Assuming that a solution satisfies the functionality criteria, the keys metrics for big data analytics are Performance and Scalability.

Performance: This is obviously a key factor in the context of big data, when dealing with 100's of TBs deployed on clusters of 100's of x86 nodes. Physical footprint translates directly to costs of operation: every cubic-foot represents real-estate, power, cooling and labor costs. Minimizing physical footprint by extracting maximum performance from the underlying infrastructure is a fundamental requirement.

Scalability: An ideal system will support linear scalability across two dimensions: size of database and size of physical system.

A benchmark suite that can quantify metrics related to performance and scalability of SQL databases will be a useful component of any broader big data benchmark. The Transaction Processing Performance Council publishes a suite of benchmarks covering workloads that include On-Line Transaction Processing (OLTP) and Decision Support systems [11]. These benchmarks are domain-specific and try to simulate real-life workloads in order to compare systems from different vendors. Others have approached benchmarking from different perspectives [12, 13].

The next section describes a benchmark created specifically to simulate big data analytic workloads, from an engineering viewpoint rather than a business application viewpoint.

5 Benchmark

The design goals for the benchmark suite were: ease of use, portability across SQL platforms, and measurements across a range of metrics that would quantify performance and scalability – of both the SQL software layer and the underlying hardware layer.

In order to meet the goals of ease of use and portability, the benchmark is written entirely in SQL. This includes the code for table definition, data generation and table population, and a set of approximately 40 queries, organized into 8 groups. The entire SQL script is designed to be run with a simple command on the system under test, with the only parameters defining the range of scale factors to be tested. The benchmark creates tables that are large relative to system memory size, to ensure that storage systems are exercised. The benchmark also includes multiple large tables, to ensure that in a parallel system data is distributed across nodes, and the underlying network used for data movement is also exercised.

The benchmark script creates tables with columns that include integer, string and floating-point data types of various sizes (Table 1):

Table 1. Benchmark: column data types and quantity

Datum	Data type	Size (bytes)	Quantity
64 bits integer	bigint	8	7
32 bits integer	integer	4	15
String	varchar	8, 10, 12, 20, 34	17
float 8	double precision	8	5

Tables are created starting with the smallest at 32 K rows, with increasing sizes going up to 16 B rows. Each new table created is twice as big as the previous table.

Typically the benchmark is run at scale factors of 0 to 5, where the database size doubles for each scale factor increase. At scale factor 0, the database size is 0.33 TB and the largest table has 0.5 B rows, at scale factor 1, the database size is 0.66 TB and the largest table has 1 B rows. At scale factor 5, the database size is 10 TB and the largest table has 16 B rows. The data is generated in such a manner that the number of rows at each stage of the processing pipeline (Scan-Filter-Join-Group-Aggregate-…), approximately doubles with each increase in scale factor. Therefore it is a reasonable expectation that query execution times approximately double with each increase in scale factor. The queries are organized into 8 groups: Scan-Filter, Equi-joins, Non equi-joins, Group/Aggregation, Sub-Queries, Set Operations, Top N, Table Creation (Tables 2, 3, 4):

Table 2. Benchmark: scan-filter queries

Class	# of Tables	Output
Scan-Filter	1	Scan with filter of 0 % selectivity
	1	Scan with filter of 1 % selectivity
	1	Scan with filter of 10 % selectivity
	1	Scan with complex filer using bit-shift, double multiplication and division, 0 %
	1	Scan with complex CASE filter, 0 %
	1	Scan with CASE projection, sort with top ten rows

Table 3. Benchmark: join queries

Class	# of Tables	Output
Equi-Join	2	2-Table inner equi-join, 0 rows result
	3	3-Table inner equi-join, 0 rows result
	4	4-Table inner equi-join, 0 rows result
	2	2-Table inner equi-join with expressions, 0 rows result
	3	3-Table inner equi-join with expressions, 0 rows result
	4	4-Table inner equi-join with expressions, 0 rows result
	2	2-Table left outer equi-join, 0 rows result
	2	2-Table right outer equi-join, 0 rows result
	2	2-Table full outer equi-join, 0 rows result
	3	Inner equi-join on 3 sub-query selects
	5	5-Table inner equi-join; big/small tables; grouping; aggregation; sort; 10 rows result
	5	5-Table full outer equi-join; big/small tables; grouping; aggregation; sort; 10 rows result
Non Equi-Join	2	Nested loop: inner non equi-join; 0 rows result
	2	Nested loop: full outer non equi-join; 0 rows result
	2	Cartesian product; grouping; sort; limited to 10 rows result

6 Benchmark Results

The benchmark described above has been used to assess many different hardware platforms and several different SQL database engines, including XtremeData's dbX and a few leading competitors. In this section, some generalized results are presented, characterizing hardware platform layers and SQL software layers.

6.1 Hardware Platforms

The benchmark has been characterized on a variety of different hardware platforms, using dbX as the SQL engine. The key factors of the hardware platform that determine performance are: computing power, storage bandwidth and network bandwidth.

Table 4. Benchmark: group-aggregates, set operation and table creation queries

Class	# of Tables	Output
Group/Aggregate	1	Group by 4 cols; sort; limited to 10 rows
	1	Count on distinct rows sub-query; 1 row
	1	Count on distinct rows sub-query with expression; 1 row
	1	Group with aggregation; sort; limited to 10 rows
	2	Count on join sub-query producing distinct rows; 1 row
Sub-query	2	Count with filter using sub-query; 1 row
	4	Sub-queries in projection and filter; 1 row
	2	Union; limited 10 10 rows
	2	Intersection; 0 rows
	2	Except: limited to 10 rows
Top Few	1	10 unique rows of a full table scan
	1	10 unique rows of a group sub-query
	2	10 unique rows of a join sub-query
Table Creation	2	CREATE TABLE AS: join sub-query result
	2	INSERT INTO: join sub-query result (duplicate rows)
	2	SELECT INTO: join sub-query result

As a rough measure of computing power, we use "Ghz" as the metric. For example, a 4-core 3.0 Ghz Xeon will be rated as 12 Ghz. This is not a perfect metric by any means, but is reasonable for these purposes. Storage and network bandwidth are directly measured in GB/s.

	XD-1	XD-2	VH_SSD	VH_SAN	XD-8	VC-1	VC-2
CPU	Nehalem	Xeon	Xeon	Xeon	Opteron	Nehalem	Nehalem
Storage	Direct	Direct	SSD	SAN	Direct	Direct	Direct
Network	IB	IB	Fiber	FC	IB	10GigE	10GigE

Where:

- XD-1/2/8: Refers to XtremData configured platforms of 1/2/8 nodes. Each node is a dual-socket Opteron server with direct-attached disks and a DDR 4xInfiniBand (IB) network.
- VH-SSD/SAN: VendorH, a Tier-1 enterprise-grade system with an SSD fiber-attached array (custom link) or a FiberChannel (FC) attached SAN.
- VC-1/2: VendorC, a Tier-2 enterprise-grade system with dual-socket Nehalem servers, direct-attached disks and a 10 gigE network fabric.

The relative strengths of CPU power across platforms (Table 5):

The relative strengths of Disk bandwidth across platforms (Table 6):

Since these hardware platforms cover a broad spectrum of physical footprint and costs, it is difficult to make direct performance comparisons. But it is instructive to isolate queries that are known to be CPU-limited (heavy computation) and queries that

Table 5. Hardware platforms: CPU capability

	XD-1	XD-2	VH_SSD	VH_SAN	XD-8	VC-1	VC-2
CPU type	Nehalem	Xeon	Xeon	Xeon	Opteron	Nehalem	Nehalem
Clock, Ghz	3.3	2.4	2.26	2.26	2.4	2.9	2.93
# Cores	6	6	8	8	4	8	6
# Sockets	2	4	8	8	16	8	16
Total, Ghz	39.60	57.60	144.64	144.64	153.60	185.60	281.28

Table 6. Hardware platforms: Disk bandwidth

	XD-1	XD-2	VH_SSD	VH_SAN	XD-8	VC-1	VC-2
Disk, GB/s	1.60	2.0	4.00	5.00	5.20	8.36	10.64

are known to be I/O limited (heavy data movement), and compare relative performance.

CPU-Limited Queries: Two queries were selected (c.2 and c.5), both performing DISTINCT operations with Joins and sub-queries. As shown, the performance of the CPU-limited queries correlates well with the CPU strengths of the hardware platforms (Fig. 5).

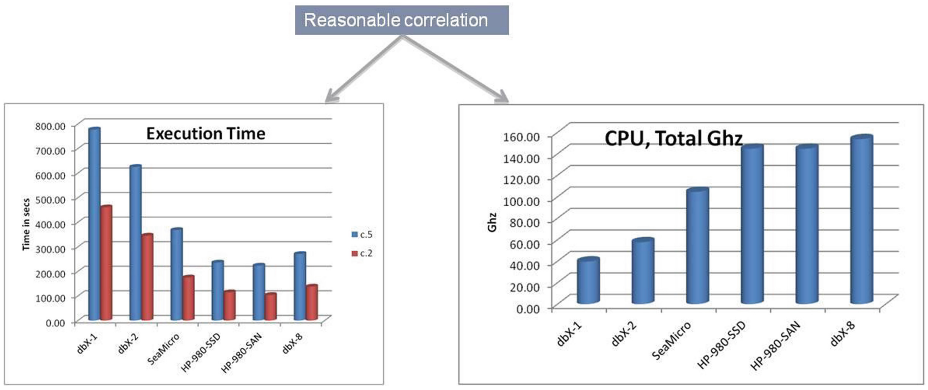

Fig. 5. Benchmark Results: CPU-limited queries

I/O-Limited Queries: Three queries were selected (b.1, b.2 and b.3), performing 2, 3 and 4 table Joins respectively. Again, as shown, the performance of the I/O-limited queries correlates well with the Disk bandwidths of the hardware platforms (Fig. 6).

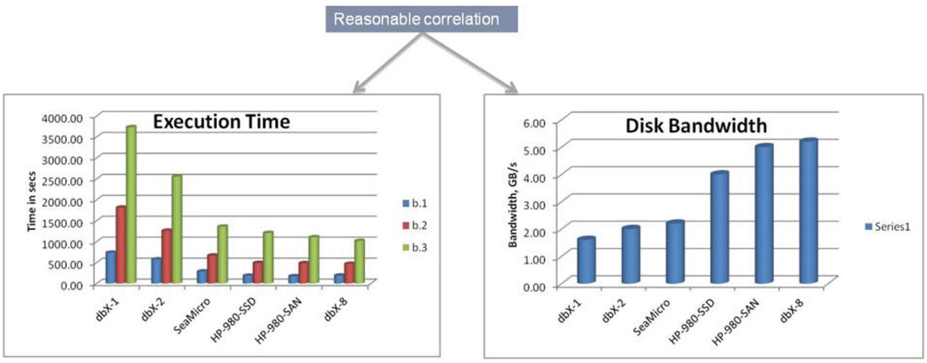

Fig. 6. Benchmark Results: I/O-limited queries

6.2 SQL Engines

Another point of reference on the usefulness of the benchmark is to characterize the performance of different SQL database engines running on the same underlying hardware infrastructure. This exercise equalizes the hardware and highlights the strengths and weaknesses of the SQL software layer.

The results below were performed on a single physical node: two-socket server with 6-Core Nehalem 3.33 Ghz CPUs, 72 GB of memory and 6 × 15000 rpm disks as RAID5, direct-attached.

The benchmark was run on 3 scale factors: 1, 2 and 3, with database sizes of 0.66, 1.32, and 2.64 TB respectively. The SQL database engines tested were XtremeData's dbX and a leading competitor, referred to as CompetitorY(or CY) below. From the benchmark results, three basic metric are shown below: raw performance at scale factor 3, linearity of scaling with database size, and consistency of query performance with query complexity and database size (Fig. 7).

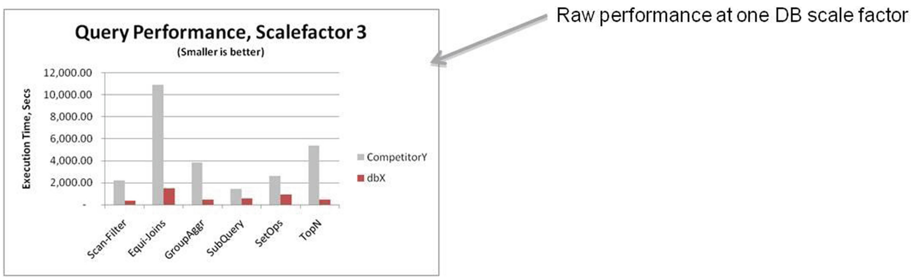

Fig. 7. Results: SQL database engines: raw performance

Raw query performance at the fixed scale factor of 3 (2.64 TB), shows that dbX was consistently faster across all query groups, an average of 7–10x faster.

Linearity of scaling and consistency of performance is illustrated next. In both charts, the vertical axis is of the same scale, and denotes normalized query execution times, with scale factor 1 set to 1. Therefore normalized execution times for scale factors 2 and 3 should ideally be 2 and 4 respectively – shown above as the "Ideal Linear" line in blue. Large divergence from the Ideal Linear line indicates non-linear scaling with database size. The charts show that dbX performance closely matches the Ideal Linear line, and also shows consistent performance across all query groups. CompetitorY results reveal significant sub-linear scaling with database size and also a large variance in performance across query groups (Fig. 8).

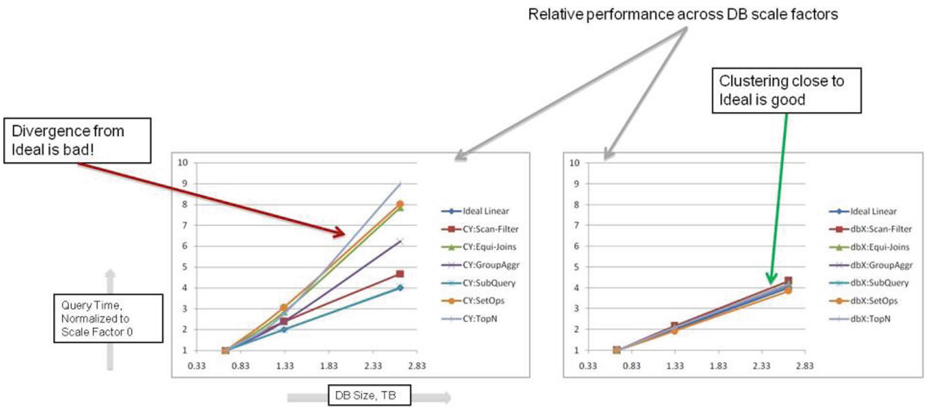

Fig. 8. Results: SQL database engines: scaling

7 Conclusions

Analysis platforms for (structured) big data need to include a full-featured MPP SQL database engine. The SQL engine should be capable of deployment in the shared, virtualized hardware infrastructure that today's data centers are converging towards, whether public or private Clouds. The SQL engine should offer high performance (maximize hardware efficiency) and offer near linear scalability with increases in database size and system size. These features determine physical footprint of the underlying hardware, which directly determines the total cost of operation. A benchmark has been developed to characterize a big data analysis platform: hardware resource layers and the SQL database layer. This benchmark is merely a starting point, and has several deficiencies:

- Synthetic data with known, fixed distribution
- All tables have same column schema
- Each new table is 2x previous table
- SQL data generation (INSERT) – can be slow!

Results have been presented using the benchmark across a variety of hardware platforms and on two competing SQL database engines. The benchmark is being continually updated to reflect newer features and application demands. Currently work is in progress on adding the following tests to the suite: ANI SQL Window functions, Insert/Update/Delete queries, continuous ingest during querying.

References

1. Big data: The next frontier for innovation, competition, and productivity, McKinsey& Company. http://www.mckinsey.com/insights/business_technology/big_data_the_next_frontier_for_innovation
2. Cloudera-Hadoop. http://www.cloudera.com/content/cloudera/en/products-and-services/cdh.html
3. Apache CouchDB. http://couchdb.apache.org
4. MongoDB: A Document Oriented Database. http://www.mongodb.org/about/
5. NoSQL Distilled. http://martinfowler.com/books/nosql.html
6. Amazon Web Services. https://aws.amazon.com/products/
7. Google Compute Engine:Quickstart: Creating an instance and launching Apache - Google Developers. https://developers.google.com/compute/docs/quickstart
8. VMware 10GE QoS Design Deep Dive with Cisco UCS, Nexus. http://bradhedlund.com/2010/09/15/vmware-10ge-qos-designs-cisco-ucs-nexus/
9. OpenStack Open Source Cloud Computing Software. https://www.openstack.org/
10. PostgreSQL: The world's most advanced open source database. http://www.postgresql.org/
11. TPC – Homepage. http://www.tpc.org/default.asp
12. Seng, J.-L., Yao, S.B., Hevner, A.R.: Requirements-driven database systems benchmark method. Decis. Support Syst. **38**(4), 629–648 (2005). doi:10.1016/j.dss.2003.06.002, http://dx.doi.org/10.1016/j.dss.2003.06.002
13. Darmont, J., Bentayeb, F., Boussaid, O.: Benchmarking data warehouses. Int. J. Bus. Intell. Data Min. **2**(1), 79–104 (2007). doi:10.1504/IJBIDM.2007.012947, http://dx.doi.org/10.1504/IJBIDM.2007.012947