# Data Virtualization: Unlocking Data for AI and Machine Learning

For reliability, accuracy and performance, both AI and machine learning heavily rely on large sets. Because larger the pool of data, the better you can train the models.

That’s why it’s critical for big data platforms to efficiently work with different data streams and systems, regardless of the structure of data (or lack of it), data velocity and volume.

However, this can be easier said than done.

Today every data platform faces these systemic big data challenges:

1. **Compute and Storage overlap:** Traditionally compute and storage was never de-lineated. As data volumes grew, you had to invest in compute *as well as* storage.
2. **Non-uniform access of data:** Over the years, too much dependency on business operations and applications have ledcompanies to acquire, ingest and store data in different physical systems like file systems, databases and data warehouses (e.g. SQL Server and Oracle), big data systems (e.g. Hadoop). This results in disparate systems each with its own method to access data.
3. **Hardware bound compute:** You have your data in nice storage schema (e.g. an SQL server), but you are hardware constrained to execute your query, as it takes several hours to complete!
4. **Data is remote:** Data is either dispersed across geo-locations, or uses different underlying technology stack (e.g. SQL, Oracle, Hadoop etc.), stored on premise or in the cloud. This requires raw data to be physically moved to get processed, and thus increased Network I/O cost.

With the advent of machine learning and AI, beating these challenges has become a business imperative. The genesis of data virtualization is rooted on this premise.

**What is Data Virtualization?**

Data virtualization offers techniques to abstract the way we handle and access data. It allows you to manage and work with data across heterogenous streams and systems, regardless of their physical location or format.

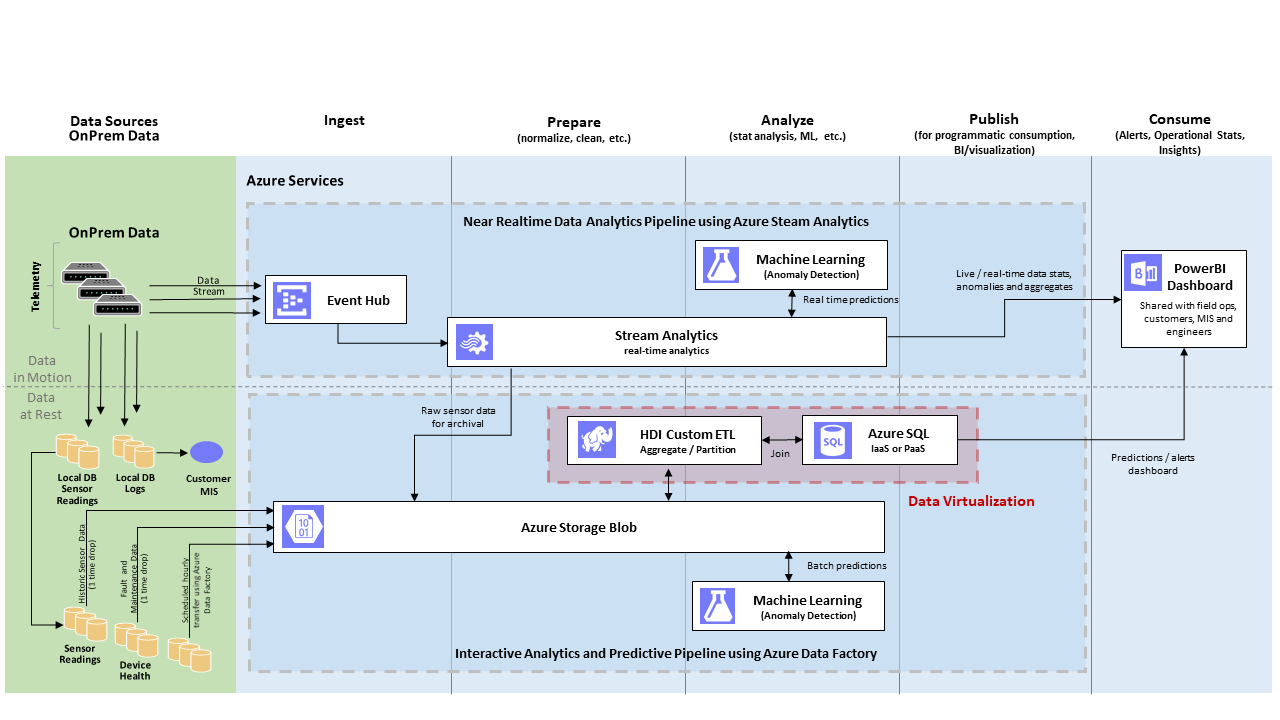
As such, data virtualization can be defined as a set of tools, techniques, and methods to access and interact with data without worrying about its physical location and what compute is done on it.

For example, say you have a pile of data spread across disparate systems and want to query all this data in a unified manner without moving the data around. That’s when you want to leverage data virtualization techniques.

In this blog, we will go over a few data virtualization techniques and illustrate how they make big data handling both efficient and easy.

**Data Virtualization Architectures**

Data virtualization can be illustrated using lambda architecture implementation of advanced analytics stack on Azure cloud:



**Figure 1: Lambda Architecture Implementation using Azure Platform Services**

In big data processing platforms, a lot of data gets ingested per second, which includes both data at rest and in motion. This big data is then collected in canonical data stores (e.g. Azure storage blob) and subsequently cleaned, partitioned, aggregated and prepared for downstream processing.

Examples of downstream processing are advanced analytics machine learning, visualization, dashboard report generation etc.

This downstream processing is backed by SQL Servers, and based on the number of users it can get overloaded when many queries are executed in parallel by competing services.

To address such overload scenarios, data virtualization provides **Query Scale-out** where part of the compute is offloaded to more powerful systems like Hadoop clusters.

Another scenario shown in figure 1 involves ETL processes running in the HDInsight clusters. ETL transform may need access to referential data stored in the SQL servers.

Data virtualization provides **Hybrid Execution** which allows you to query referential data from remote stores, such as SQL servers.

**Query Scale-out**

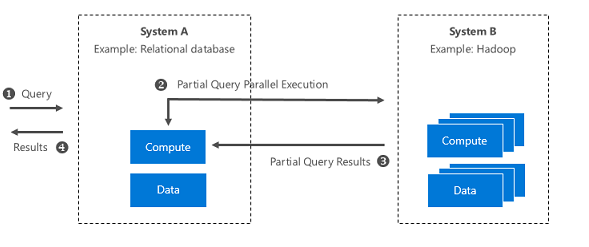
**What it is:**

Say you have a multi-tenant SQL Server running on a hardware constrained environment. You want to offload some of the compute to speed up the queries. You also want to access the big data that won't fit in the SQL Server.

In such cases, Query Scale-out can be used.

Query Scale-out uses PolyBase technology (introduced in SQL Server 2016). PolyBase allows you to execute part of the query remotely on a faster, higher capacity big data system, such as Hadoop clusters.

The architecture for Query Scale-out can be illustrated as:



**Figure 2: System-level illustration of Query Scale-out**

**What it solves:**

* **Compute and storage overlap:** Delineates compute from storage by running queries in external clusters. Extends SQL server storage by enabling access of data in HDFS.
* **Hardware bound compute:** You can run parallel compute leveraging faster systems.
* **Data is remote:** Keep the data where it is, only return the processed result set.

You can deploy and further explore Query Scale-out using the [one-click automated demo at the solution gallery](https://gallery.cortanaintelligence.com/Tutorial/Data-Virtualization-Techniques-Using-SQL-Server-2016-and-HDInsight).

**Hybrid Execution**

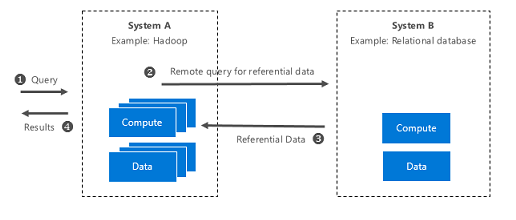
**What it is:**

Say you have ETL processes which run on your unstructured data and then store the data in blob. You need to join this blob data with referential data stored in a relational database. How would you uniformly access data across these distinct data sources?

In such cases, Hybrid Execution can be used.

Hybrid Execution allows you to “push” query to a remote system, such as a SQL Server, to access the referential data.

The architecture for Hybrid Execution can be illustrated as:



**Figure 3: System-level illustration of Hybrid Execution**

**What it solves:**

* **Uniform access of data:** No longer constrained by where and how data is stored.
* **Data is remote:** Access reference data from external systems for use in downstream apps.

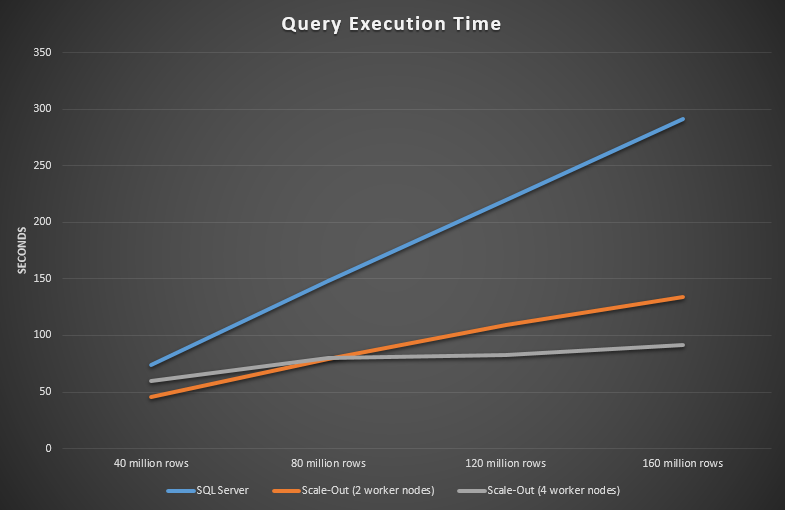
You can deploy and further explore Hybrid Execution using the [one-click automated demo at the solution gallery](https://gallery.cortanaintelligence.com/Tutorial/Data-Virtualization-Techniques-Using-SQL-Server-2016-and-HDInsight).

**Performance Benchmarks: What optimization gains can you expect?**

You may ask yourself if it is worth to use these techniques.

Query Scale-out makes sense when the data already exists on Hadoop. Referring to figure 1, you may not want to push all the data to HDInsight just to see the performance gain.

However, one can imagine a use case where lots of ETL processing happens in HDInsight clusters and the structured results are published to a SQL Server for downstream consumption for example, by reporting tools. To give you an idea of the performance gains you can expect using these techniques, here are some benchmark numbers based on the datasets used in the solution demo. These benchmarks were produced by varying the size of datasets and the size of HDInsight clusters:



**Figure 4: Query execution time with and without scaling**

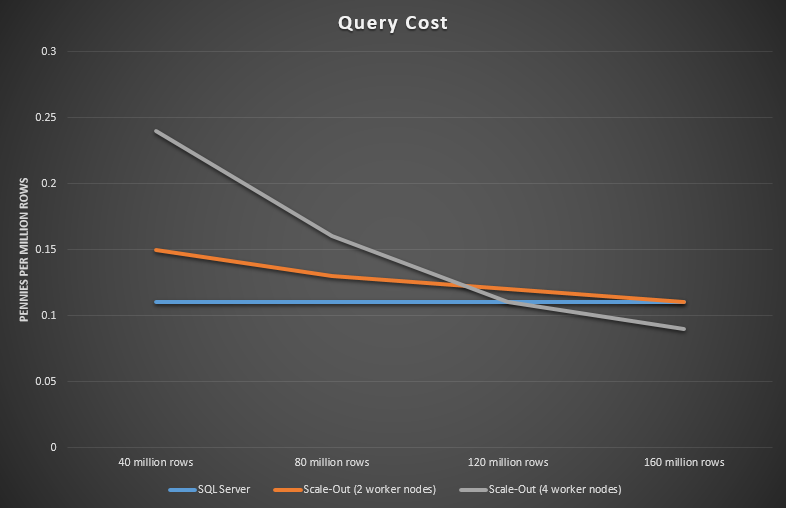
The x axis shows the number of rows in the table used for benchmarking. The y axis shows the number of seconds the query took to execute.

Note the linear increase in execution time with SQL Server only (blue line) versus when HDInsight is used with SQL server to scale out the query execution (orange and grey lines).

Another interesting observation is the flattening out of execution time of a four versus a two-worker node HDInsight cluster (compare grey line with orange).

Of course, these results are specific to the simplified dataset and schema we are providing with the solution demo. With much larger real world datasets in SQL Server, which typically runs multiple queries competing for resources, more dramatic performance gains should be expected.

The next question to ask is *when it becomes cost effective to switch over to using Query Scale-out.* The below chart incorporates the pricing of resources used in this experiment. You can see the detailed [pricing calculation here](https://github.com/Azure/cortana-intelligence-data-virtualization-techniques-using-sql-server-2016-and-hdinsight/blob/master/Benchmark-and-Costing.md).



**Figure 5: Query execution time with and without scaling (with pricing)**

You can see that with 40 million rows it is cheapest to execute this query on SQL Server only, and by the time you have 160 million rows scaling out becomes cheaper. This shows that as the number of rows increases, it would become cheaper to run with scaling out.

You can use these types of benchmarks and calculations to help you deploy resources with the proper balance of performance and cost.

**Try the solution yourself, with one-click deployment.**

To try out the data virtualization techniques discussed in this blog, deploy the solution demo in your Azure subscription using the [automated one-click deployment solution](https://gallery.cortanaintelligence.com/Tutorial/Data-Virtualization-Techniques-Using-SQL-Server-2016-and-HDInsight).

For a deeper understanding on how to implement the data virtualization techniques, read the [technical guide](https://github.com/Azure/cortana-intelligence-data-virtualization-techniques-using-sql-server-2016-and-hdinsight/blob/master/README.md).