

***DataStax in a Modern Data Architecture***

Cliff Gilmore

SWAT Solutions Architect

cgilmore@datastax.com

# Executive Summary

The DataStax Enterprise platform brings advantages in simplification and overall functionality of a modern data architecture. In this white paper you will learn about the modern Lambda Data Architecture and how DataStax Enterprise fits within that architecture to solve the complex and varied needs of data retrieval, processing and analytics. In addition the different steps of the data pipeline are outlined and the query languages and tools specified. The overall architecture is simple with very few moving parts but as seen in the illustrations contained within, there are many options of accessing the data and the platform supports several significantly different consumers of the information.

**Table of Contents**

[**Executive Summary**](#h.gjdgxs)

[**Introduction**](#h.30j0zll)

[**Enter the Lambda Architecture**](#h.1fob9te)

[**Why do we need a new Architecture?**](#h.3znysh7)

[**Basics of Lambda**](#h.2et92p0)

[**Components of Lambda**](#h.tyjcwt)

[Speed layer](#h.3dy6vkm)

[Batch Layer](#h.1t3h5sf)

[**The DataStax Lambda Architecture**](#h.4d34og8)

[**Components of the DataStax Lambda Architecture**](#h.2s8eyo1)

[Batch layer](#h.17dp8vu)

[Serving and Speed Layers](#h.3rdcrjn)

[**Usage of DataStax Lambda Architecture**](#h.26in1rg)

[Entire Architecture](#h.lnxbz9)

[**Conclusion**](#h.35nkun2)

[**About DataStax**](#h.1ksv4uv)

# Introduction

2016 brings us new consensus as the design principles of a modern data architecture have graduated from the bleeding edge realm of startups and technology giants to becoming the foundation of the modern Enterprise data ecosystem. The days of a massive network of slow and fragile ETL jobs moving data around an enterprise are coming to an end. So too are the extreme difficulties of bringing data from different business units and groups together not just in analytics environments but actually exposing these data sets across the enterprise via services and micro services. This is a result of a modernization in the thinking behind how data should flow through the enterprise and the technologies where it will reside for different capabilities. Furthermore during the last decade the capability of a single machine to grow vertically (more power in one physical machine) has stagnated and the cost of these machines results in significant diminishing returns as they scale up.

This change has led to the enterprise model shifting from large monolithic hardware towards more commoditized “pizza box” type machines that give the best value to dollar ratios. While existing “legacy” applications could not take full advantage of these commoditized servers, a new generation of processing frameworks and databases were required to drive the change in the modern physical topology. Thus arrived the distributed databases and file systems that are becoming more and more the standard today, technologies such as HDFS and Cassandra.

This whitepaper will discuss the basics of the Lambda Architecture and compare how a proposed DataStax architecture would exceed the original design and provide for more operational simplicity and enhanced capabilities.

# Enter the Lambda Architecture

## Why do we need a new Architecture?

One of the largest challenges to building a functioning data architecture has been the reliance on a transactional databases as a system of record that then was responsible for replicating data downstream to data warehouses, operational data stores, raw data extracts, and other dependent systems.

Why is this a challenge? The system of record is often unable to serve either real time or batch analytics, thus resulting in multiple data movements via ETL or database replication. These complex movements result in both delays (minutes, hours, even days!) and fragility of the architecture. At this point the ACID guarantees of the source database become meaningless due to the eventually consistent downstream databases being the likely entry point for reads by Users and Customers. The inability for these databases to scale up also led to fragmentation of data ownership and and a lack of design simplicity.

## Basics of Lambda

How do we solve for both low latency and highly analytical queries and operations? While battling these problems at Twitter, Nathan Marz formalized the concept of a split data flow architecture he coined as the Lambda Architecture.[1] The critical insight was to split data at ingest time so that the different systems optimized for specific capabilities would be able to either serve or start working on the data immediately without having to add an extensive network of ETL functionalities which would add delays to the processing and fragility points as it would rely on change capture from a multitude of systems that would need reconciliation.

## Components of Lambda

The Lambda architecture is logically divided into three layers which each have a different job in the overall data ecosystem. The layers are the Batch Layer, the Speed Layer, and the Serving layer with different combinations of technologies in each. The following is a description of the original Lambda architecture.

### Speed layer

The SPEED layer is designed to process data in near real-time and make it available immediately. This would be accomplished through storing a smaller overall segment of the data in a higher performance environment than the serving layer. It would also involve a stream-processing component that could do basic aggregations and pattern matching on the live stream of data as it flows in off the queuing system.

This architecture does not account for more ad-hoc real-time queries through the lack of a search layer and requires a lot of data movement steps as opposed to integrated processing and result movement.

*[1] How to Beat the CAP Theorem – Nathan Marz 2011 [http://nathanmarz.com/blog/how-to-beat-the-cap-theorem.html]*

The goal if a preformat architecture is to ingest the data as fast as possible and only move the data that is a result of a query or a processing stage. With the goal of no intersystem bulk transfer of raw data.

### Batch Layer

The BATCH layer has the job of storing and processing all data across all applications. It will often be a large file store such as HDFS or a RDBMS based Data Warehouse. Generally this environment will store the full history of a multitude of different data sets. The goal of this layer is to process large analytical tasks that scan over significant subsets of the data that are unrealistic to access in real-time or even near real time. An example would be deep analysis of 100s of TB or even PB of data in a single “full scan” type of run for a specific algorithm or process. These huge data sets will be processed using algorithms written to run on parallel processing frameworks such as MapReduce or Spark and produce results that can be productionalized to the user and Data Scientist in the other Layers.

### Serving Layer

The SERVING layer has the job of making data available to applications for interactive access. This layer would often be a RDBMS or a NoSQL database that directly connections to users and applications. Information from the Batch layer would be processed and loaded into the serving layer for access.

# The DataStax Lambda Architecture

## Components of the DataStax Lambda Architecture

The DataStax lambda architecture has a different look entirely. It is based on Apache Cassandra, with additional capabilities built on top of it. Together, these components make up the DataStax Enterprise platform. This platform provides for multiple processing paradigms on top of the semi-structured data model of Cassandra.

### Batch layer

In the DataStax lambda architecture we can take advantage of the replication capabilities of Cassandra along with the added capabilities of DataStax Enterprise to expose different degrees of ad-hoc data and choose what kind of processing is needed (real-time, near real-time, batch) with a minimum of data movement.

The BATCH layer changes significantly in that while HDFS or a file storage system is still in the picture it is not the sole location of batch processing. Only the largest and deepest retrospective analytical jobs will need to be processed in HDFS. These data sets would be either directly delivered to the Data Scientist from HDFS or stored in Cassandra for application access, it would also be available for search via automated and integrated Solr Indexing. Batch processing via the Spark framework could act directly on the ingested data within Cassandra for more timely narrowly targeted analytics on subsets of the data that would then result in results stored in-place within the Cassandra data model

### Serving and Speed Layers

The Serving layer and the Speed layer are both contained within the DataStax Enterprise platform entirely. With Spark Streaming processing real-time data and storing the results in Cassandra for both direct and search queries, there is no need to forward that information to a serving layer. Roll-ups that have been calculated locally and in the BATCH layer will be served out of Cassandra alongside the raw ingested data. This allows for one environment and significant architectural simplicity over having separate physical layers. While the jobs of both serving and speed are handled with separate data tables within Cassandra, the management of this data is simplified by a common access language (CQL) and operations are unified in a single platform.

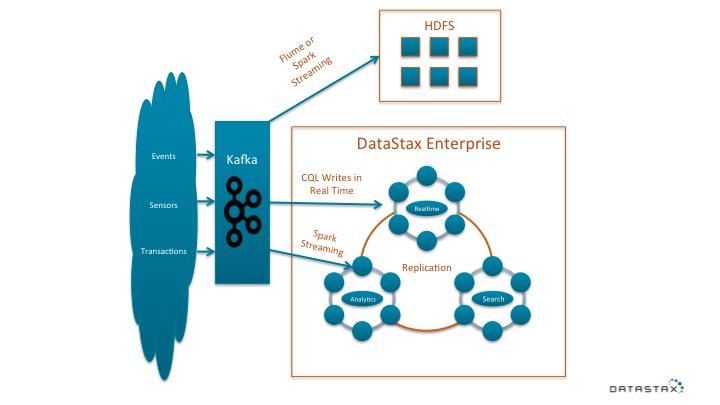
## Usage of DataStax Lambda Architecture

As you can see in this architecture there are no direct ETL jobs required. There are a few key stages to processing and access that we can delineate with different technologies and access points.

#### Event Capture

Events are captured from a multitude of sources and pushed into a message queue, which in this example we implement with Kafka. As events flow in they are made available via topics for multiple subscribers to read in parallel. Kafka provides a scalable and fault tolerant master source of data.

#### Ingest



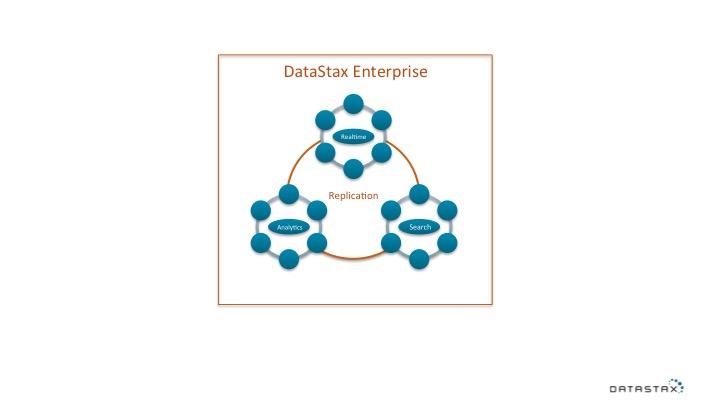
Ingest occurs when different components of the architecture pull data from Kafka or another queue technology.

The first leg is ingest into the distributed file system in an append manner or via a stream framework which would store results in the file system.

The second leg is a direct ingest into Cassandra for serving and speed access via CQL. This is the fastest path to making the data available to serve.

The third leg is spark streaming running on top of Cassandra as part of DataStax Enterprise. This leg will process the aggregations and store those in Cassandra.

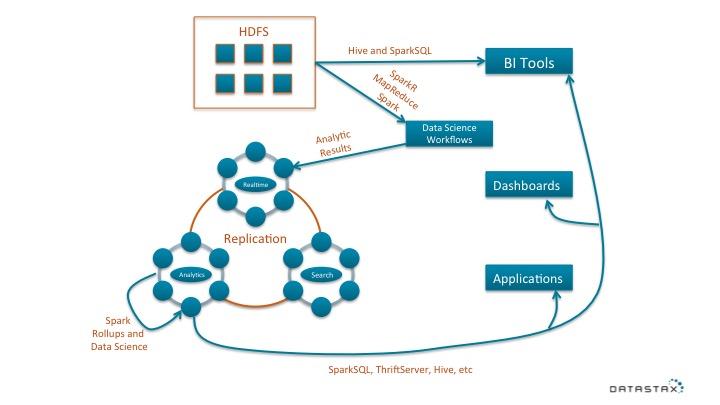
#### Replication



Data within the different datacenters of DataStax Enterprise will be replicated automatically with millisecond latency to serve different workloads in isolation as well as making the data available for indexing of search. This can span one or more physical data centers out of the box. Keyspace settings within Cassandra allow for control over what data goes to what logical or physical data center.

A special case for replication is that when data replicates into a Search configured (Solr) datacenter within the DataStax cluster it will be automatically indexed for search queries.

#### Analytic Workflows

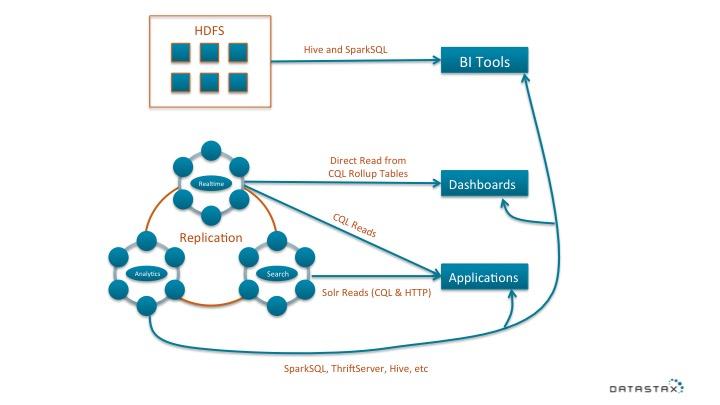


Programmatic or user driven analytics can be run in a few different parts of the architecture. Processing of recent real-time data jobs executed via Spark or SparkSQL can roll-up information to store back into Cassandra. Processing on the entire data universe would occur in the Batch layer or HDFS. This example could be accomplished with MapReduce, SparkSQL, Spark or other operations with the results stored in Cassandra.

Interactive workflows can be accomplished using modern notebook tools such as Apache Zepplin or Jupyter Python notebooks. These workflows are written in Spark (Python, Scala or Java), SparkSQL, or even CQL directly against the data.

Traditional BI tools that speak SQL can access either the batch layer or Cassandra via SparkSQL to push down workloads and deliver a result set into memory of the too.

#### Application Access



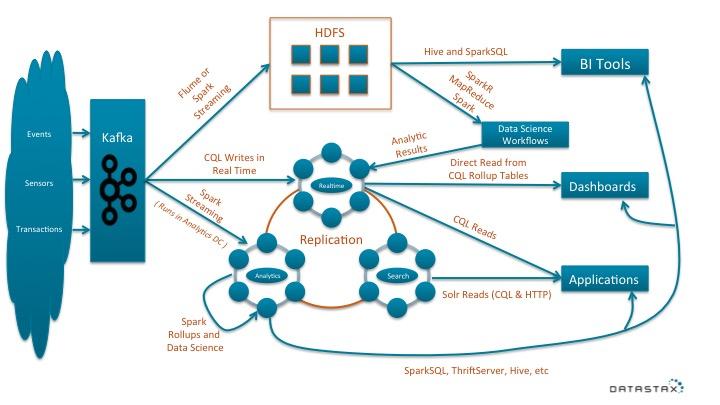
Applications will have a few core access paths.

For the lowest latency direct access of data, CQL can be used to do highly concurrent and high volume of random reads across semi-structured data in Cassandra. These queries will follow known paths to either data stored in “query tables” or rollup tables, which have been processed via rollups or stream micro batches.

For low (~1s or less) latency queries that have some aspect of unknown or ad-hoc qualities the best access path is through the CQL Solr API or HTTP REST API which allows for complex and performant search queries on the data stored in Cassandra within the DataStax Enterprise platform. As soon as data is replicated to the Solr Datacenter it will be indexed and made available for search in this manner.

For totally ad-hoc and complex queries there are a few options for access. These will not be real-time queries but the type that can wait seconds or minutes for completion. Either through the CQL access path combined with client side processing or through an SQL processing path via SparkSQL.

### Entire Architecture



# Conclusion

This DataStax Lambda architecture is optimized for maximum performance and the most operational simplicity possible with production ready technologies of today. Stay tuned for a follow up edition of this whitepaper as DataStax has some exciting technologies in the pipeline to provide an overall solution for data management of the modern enterprise.

# 

# About DataStax

DataStax delivers Apache Cassandra in a database platform purpose built for the performance and availability demands of Web, Mobile, and IOT applications, giving enterprises a secure always-on database that remains operationally simple when scaled in a single datacenter or across multiple datacenters and clouds.

DataStax has more than 500 customers in 38 countries including leaders such as Netflix, Rackspace, Pearson Education, and Constant Contact, and spans verticals including web, financial services, telecommunications, logistics, and government. Based in San Mateo, Calif., DataStax is backed by industry-leading investors including Lightspeed Venture Partners, Meritech Capital, and Crosslink Capital.