Percipient

intelligent data engineering

real-time PLUS™

Powering Comprehensive Real-Time Applications

A Percipient Technology White Paper

Author: Ravi Shankar Nair

Chief Technology Officer

Updated Dec 2016

The demand for real-time analytics and streaming applications has grown over the years. Today, the explosion of cloud, mobile and social media, alongside sensors, devices and drones, all point to a new real-time paradigm. This paradigm offers organisations the ability to implement in-the-moment marketing, revolutionise manufacturing efficiencies, and re-imagine public services.

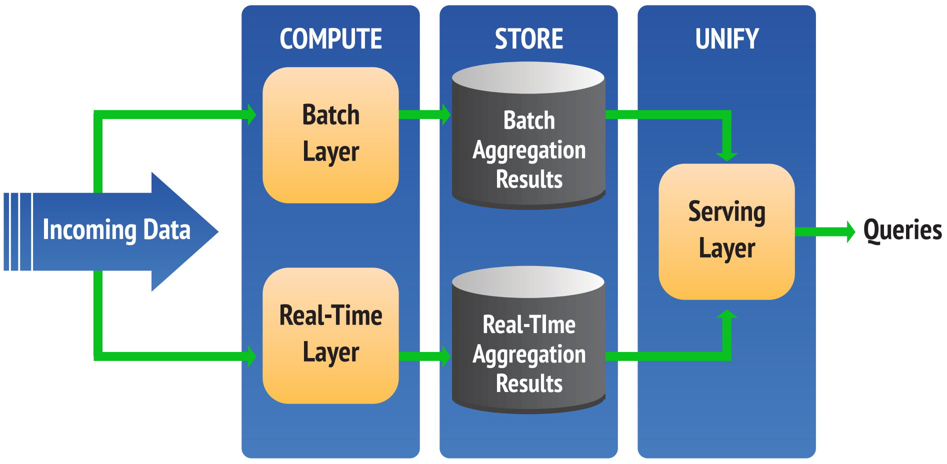
Not surprisingly, there have been many attempts to devise an architecture that ensures real-time data processing is secure, scalable, and extensible. But that is only half the story. To derive maximum value from real-time data, it is important that real-time data can be efficiently combined with incrementally updated batch data, and with data stored in a variety of other databases and warehouses.

Lambda Architecture

A milestone in the journey to seamlessly join real-time and batch data was that proposed by Nathan Marz in 2012, which he termed the Lambda Architecture.

Thanks to the incremental algorithms implemented in the real-time layer, the computation cost is significantly reduced.

Fig 1: Lambda Architecture



As shown in Figure 1, this architecture is composed of three layers: batch, real-time and serving. The batch layer has two major tasks: (a) managing historical data, and (b) receiving arriving data, and then combining the two and re-computing the results by iterating over the entire combined data set. As the batch layer operates on the full data, it allows the system to produce the most accurate results. However, this comes at the cost of high latency due to its prolonged computation time.

The real-time layer, on the other hand, is used to provide results in a low-latency, near real-time fashion. The real-time layer receives the arriving data and performs incremental updates to the batch layer results. Thanks to the incremental algorithms implemented in the real-time layer, the computation cost is significantly reduced. Finally, the serving layer enables various queries of the results sent from the batch and real-time layers.

Marz's Lambda architecture has proven to be highly applicable and has been adopted by online giants like Yahoo and Netflix. For example, Netflix is able to offer movie recommendations by using the batch layer to train an analytical model from scratch. Meanwhile the real-time layer is used to enable incremental updates of the original model. In this way, Netflix is able to meet its twin needs of accuracy and responsiveness.

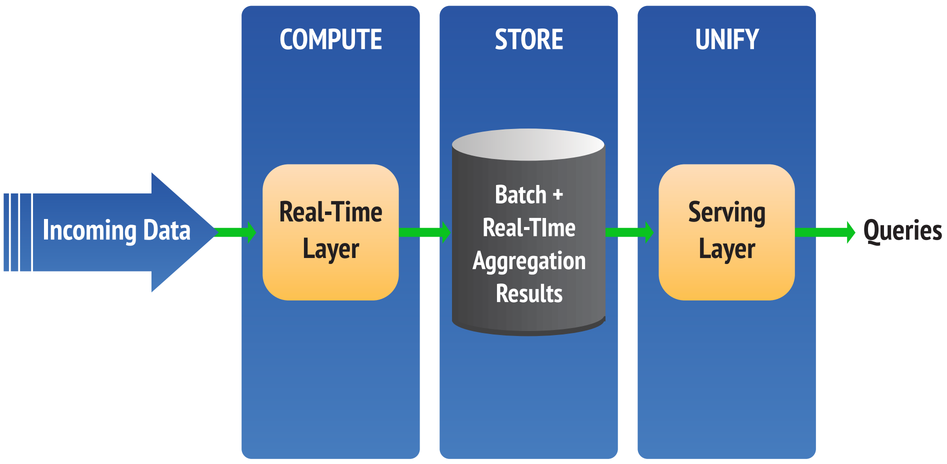
However, Lambda does have its weaknesses, in particular the coding overhead that it can create. This problem prompted LinkedIn's Jay Kreps to post an article in the summer of 2014, describing an alternative architecture that he called Kappa.

This architecture is ideal for running analytics on data that does not require a different algorithm for the batch and real-time layers.

Kappa Architecture

One of the important motivations for the Kappa architecture (Figure 2) was elimination of the need to maintain two separate code bases for the batch and speed layers. The key idea was a design capable of handling both real-time data processing and continuous data reprocessing using a single stream processing engine. Data reprocessing is an important requirement for making visible the effects of code changes on the results.

Fig 2: Kappa Architecture



As a consequence, the Kappa architecture is composed of only two layers: real-time processing and serving. The real-time processing layer runs the stream processing jobs. Normally, a single stream processing job is run to enable real-time data processing. Data reprocessing is only done when codes within the stream processing job need to be modified. This is achieved by running another modified stream processing job and reprocessing all previous data. Finally, similar to the Lambda architecture, the serving layer is used to query the results.

This architecture is ideal for running analytics on data that does not require a different algorithm for the batch and real-time layers. For example, in order to understand the transportation needs across locations, it may be necessary to apply an algorithm on the movement of individuals in these locations during different times of the day. This algorithm is relevant whether the data is streaming in real-time, or is historical data.

This means that Kappa is not a replacement for Lambda, and some use-cases using the Lambda architecture cannot be migrated to Kappa. Hence, it is necessary to accurately evaluate which architecture is best for any given use-case. A misplaced design decision can have serious consequences for the final implementation of the project.

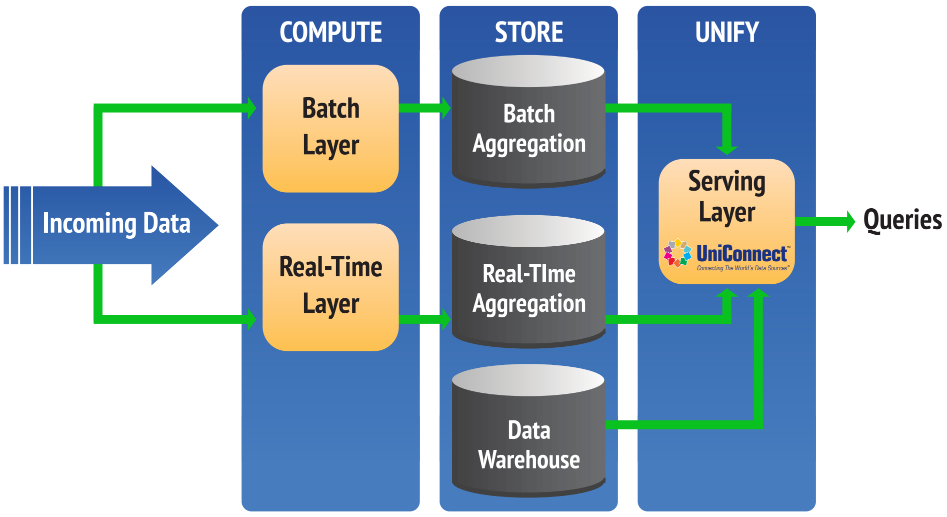
real-time PLUS™ Architecture

While both Lambda and Kappa deal with processing real-time and batch data in different ways, it is likely that a third source of data needs to be queried. This is data that has already been stored in a database, for example, customer information.

In the Netflix example above, movie recommendations would be even more powerful if it was possible to join behavioural data pertaining to the audience's clicks, views, ratings and location, with their demographic and profile data stored in a separate database.

Percipient offers organisations the ability to seamlessly join warehoused, batch and real-time data in order to achieve the most comprehensive analytics model possible.

Fig 3: Real-TimePLUS Architecture



To achieve this, **Percipient has developed an innovative approach called Real-TimePLUS**, as shown in Figure 3. Depending on the use case and data sources, Percipient offers organisations a careful evaluation of whether a Lambda or Kappa architecture is more appropriate. Furthermore, by deploying its UniConnect platform, Percipient offers organisations the ability to seamlessly join warehoused, batch and real-time data in order to achieve the most comprehensive analytics model possible.

First, Percipient's UniConnect platform facilitates the aggregation of data across data sources in memory. By deploying UniConnect within the Serving Layer, UniConnect then enables a further analytical model to be applied to the combined results of the real-time and batch layers, unified with data separately residing in one or more databases, an Enterprise Data Warehouse and/or a big data storage platform like Hive.

Open Source Software

The Lambda and Kappa architectures described above can be implemented by combining various open-source technologies, such as Apache Kafka, Apache HBase, Apache Hadoop (HDFS, MapReduce), Apache Spark, Apache Drill, Spark Streaming, Apache Storm, and Apache Samza.

For example, data can be ingested using a publish-subscribe messaging system such as Apache Kafka. Meanwhile, data and model storage can be implemented using a persistent storage platform like HDFS. A high-latency batch system such as Hadoop MapReduce can be used in Lambda architecture's batch layer to train models while low-latency systems, for instance Apache Storm, Apache Samza, and Spark Streaming can be used to implement updates in the real-time layer.

Percipient is well positioned to ensure that an organisation is advised of the appropriate architecture for its real-time needs.

Alternatively, Apache Spark can be used as a common platform to develop the batch and speed layers in a Lambda architecture. In this way, much of the code can be shared between the batch and speed layers. The serving layer can be implemented using a NoSQL database, such as Apache HBase.

As open source technologies experts, Percipient is well positioned to ensure that an organisation is advised of the appropriate architecture for its real-time needs, and how this architecture can be introduced and embedded via a selection of the most rigorously tested open source software available today, blended with its proprietary code.