White Paper

Lambda

Lambda Architecture and HPCC Systems

February 2014

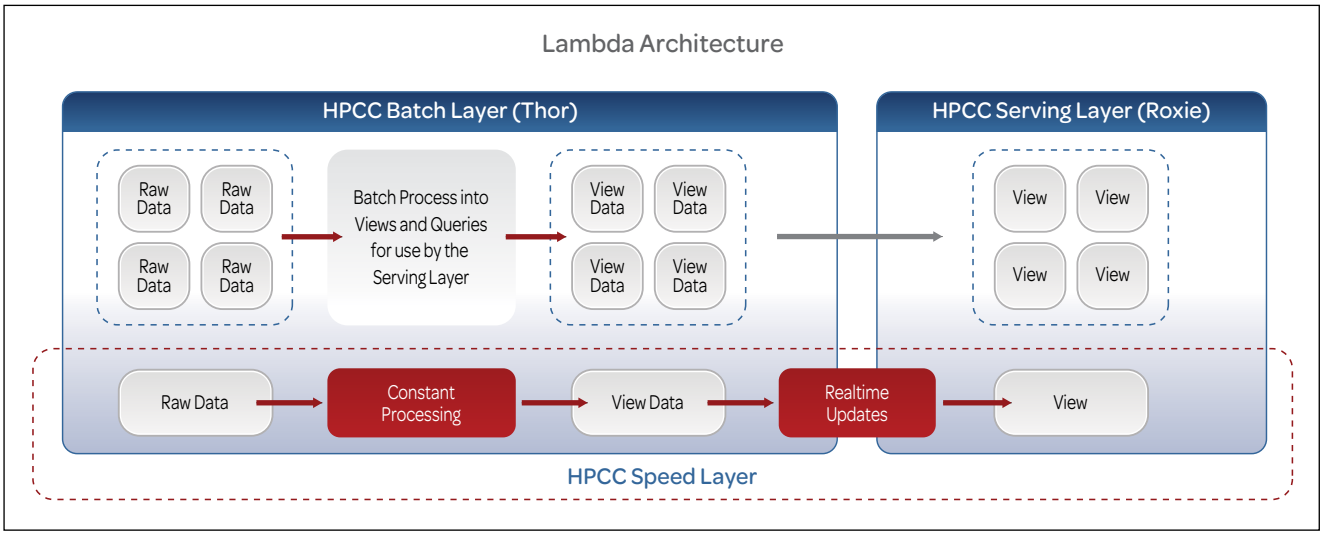
LexisNexis®

The term "Lambda Architecture", coined by Nathan Marz and outlined in his book *Big Data: Principles and best practices of scalable real-time data systems*, describes a solution for something that is in more and more demand: the efficient processing and serving of extremely high volumes of data in an efficient, scalable and fault-tolerant manner.

Lambda Architecture is a set of platform-agnostic principles and best practices for Big Data processing systems. Lambda is made up of three layers: The **Batch Layer**, which is focused on the ingest and storage of large quantities of data and the calculation of views from that data; the **Serving Layer**, focused on the query and retrieval of views loaded by the batch layer; and a **Speed Layer**, which serves the need for cases where data must be ingested and made available for querying in a low-latency, streaming manner.

Many big data solutions have naturally evolved towards this model, shaped by the unrelenting demand for ever-better performance against ever-increasing data. One of these solutions is HPCC Systems (HPCC), a processing platform long used by LexisNexis for hosting and processing their big data resources, and made available to the open source community in 2011.

HPCC Systems is a naturally-evolved example of the Lambda Architecture. Developed over many years of constant demand by LexisNexis for better performance against larger sets of data, it adopted the structures and principles of Lambda Architecture as the most high-performing, scalable and maintainable solution for big data processing.



The Batch Layer

As data processing and serving systems grow exponentially larger, so do the consequences of loss or corruption of the data being processed. A system implementing the Lambda Architecture must be tolerant of faults at a number of levels, from failure of hardware, to failure of the external processes providing the data, to handling failure in the data itself caused by human error. A Batch Layer must be able to:

- Recover from any hardware failures of the servers that make up the batch processing cluster
- Recover from the accidental loading of corrupt or incorrect data by humans or external interfaces.
- Easily expand as the data contained in it grows.
- Recompute all queries and views from the raw data at any time.

In addition, the Lambda batch layer stores an immutable, append-only, constantly expanding master copy of the system's data. From this data it computes and re-computes the queries, views, derivative data and indices for consumption by the serving layer.

HPCC's Batch Layer, Thor, is a data repository composed of a Thor Master node handling a cluster of nodes known as Thor slaves. Analogous to a Hadoop MapReduce cluster, the Thor master node co-ordinates the work done on multiple slave nodes, handling the node-to-node transfer of data needed to perform complex sorts, joins and functions against the data, and creates the final views and queries for delivery to the Serving Layer.

HPCC's Batch Layer is fault-tolerant on a number of levels. An HPCC batch cluster can be configured for either automatic or manual swapping out of bad nodes, while the data stored on each node is backed up on other nodes in the cluster. When a node fails—and, in large clusters of hundreds or more nodes, this is not only inevitable but a regularly expected occurrence—HPCC can seamlessly switch out the failed node with a fresh one, reload the data from backup, and restart the job being processed from the most recently defined persist checkpoint.

The Batch Layer of HPCC allows a file to be read from, or to be written to, but you cannot read from a file and then update that same file. This feature has the effect of encouraging another tenant of the Lambda Architecture: storing a read-only copy of the raw data used by the system, and computing and recomputing views at need. This procedure is an HPCC best practice. The raw data is kept in its original format. Any changes, cleansing, normalization or other transformations of the data are saved to additional files, if desired, during the computation of views.

Raw data is loaded into an HPCC system and distributed across all the nodes of the batch layer via HPCC's web-based administrative front end, or programmatically with HPCC's programming language ECL, (Enterprise Control Language). Once loaded in, the data is accessed in ECL by defining a dataset attribute for the raw data file:

```
dsPerson:=DATASET('~incoming::persondataraw',{UNICODE name, UNICODE Address,
STRING city, STRING phoneNumber,DATE dateofbirth},CSV);
```

An additional View attribute is created, which contains all of the logic needed to prepare and transform the raw data into the view. To minimize processing overhead, various checkpoints/persists can be defined within the View attribute so that, if an error occurs, or if the view is rerun without the original data changing, the View generation process starts at the latest checkpoint at which the data remains unchanged rather than returning to the beginning of the process.

```
dsPersonView:=PersonView(dsPerson);
```

Finally, this view is prepared for exposure by the Serving Layer, in this case by being compiled to an index:

```
personIndex:=INDEX(dsPersonView,{name,address,phone,city,state,zip,RecPtr},'~i
ndexes::personIndex');
BUILDINDEX(personIndex);
```

At this point it can be deployed to the Serving Layer in a variety of ways, either programmatically from within ECL itself, by using the HPCC front-end, or via a scheduled update.

The Serving Layer

This layer of the Lambda Architecture is focused on one thing: serving up views of the data as quickly as possible. It is a read-only layer, serving up queries and views from data compiled on the Batch Layer.

HPCC's Serving Layer, Roxie, is designed specifically with this goal in mind. A Roxie cluster is built to handle thousands of simultaneous calls to access the views and queries loaded onto the cluster. Data goes one way, from the Batch Layer to this Serving Layer. The Serving Layer never writes data back to the Batch Layer, although queries and views on the Serving Layer can be accessed by the HPCC batch layer when processing new data.

Like the Batch Layer, the HPCC Serving Layer is designed for geometric scalability and fault-tolerance. Like the Batch Layer, it allows easy switching out of failed nodes. If one node fails, the HPCC Serving Layer continues to run with minimal degradation in performance.

Together, the HPCC Batch and Serving Layers handle the majority of data consumers' needs. In some cases, however, data needs to be processed and made available for querying on a real-time or streaming basis.

The Speed Layer

This is where the Lambda Architecture's Speed Layer comes into play. It sits atop the hardware and software of the Batch Layer and Serving Layer, using their processes to stream raw data into the Batch Layer, process it into views, and deploy those views on the Serving Layer. It meets the need for low-latency real-time or near real-time data views required by some systems.

This layer can be the most challenging to implement on big data platforms, which are frequently geared towards data warehousing and analysis arenas where a certain level of latency is acceptable.

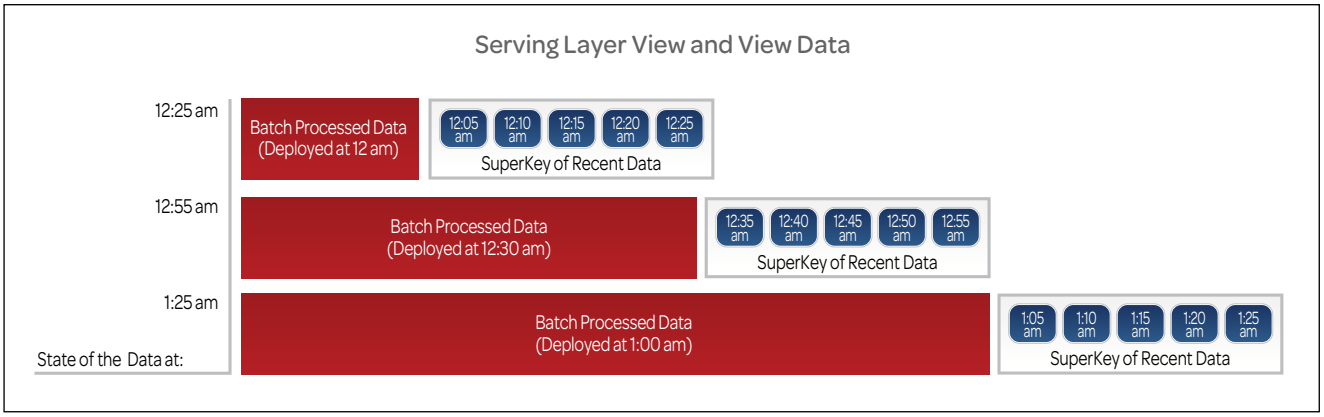
In **HPCC's Speed Layer**, real-time and streaming data processing are achieved by combining a number of different features of the Batch Layer and Serving Layer. HPCC includes an Apache Kafka Consumer plugin that allows it to receive data from Apache Kafka, an open-source, large-scale messaging system with high fault-tolerance, high throughput, and built-in partitioning and replication. Once in the Batch Layer, this data is processed into its final form and continuously deployed to the Serving Layer for consumption.

In many big data batch processing systems, adding data to the file or index underlying a view on the Serving Layer requires a recompilation of the entire index when new data is added. This isn't practical in the case of processing and serving up high volumes of streaming data. HPCC has a built-in solution for this situation, one that allows the data files and indexes underlying Serving Layer views to be incrementally updated: Superfiles and Superkeys.

An HPCC Superfile is a virtual file composed of a large number of sub-files and/or sub-Superfiles. A Superkey is very similar: a virtual index key composed of one-to-many payload indexes and/or additional Superkeys. All sub-files and sub-keys adhere to the same data layout, and sub-files can be added to a Superfile on the Serving Layer without the entire file or view needing to be redeployed.

Needless to say, virtual Superkeys do not perform as quickly as a single compiled file containing the same data would perform. Therefore, in the speed layer, Superkeys are used to supplement batch-processed views. A batch-processed view is generated on a scheduled basis, daily or hourly; and an additional, constantly updated Superkey is used in

conjunction with the batch-processed data to provide up-to-the minute data on the Serving Layer. The recent data in the Superkey is rolled into the next batch processed view, and the Superkey cleared out to be incrementally repopulated until the next scheduled batch processing of the data, and so on. An illustration of this practice is shown below:



Together, the Batch, Serving and Speed Layers of HPCC fulfill the core requirements of the Lambda Architecture:

Scalable Both the Batch and Serving Layers are built on COTS hardware, and are geometrically scalable to thousands of nodes by simply adding new hardware to a cluster.

Debuggable Because the HPCC Batch Layer keeps the raw data for all queries and views, and has the ability to re-compute them from scratch, it is easier to trace errors back to their source. HPCC's middleware layer provides a way to view all versions of the ECL used to calculate a view, including the original query text, and the original result output. Finally, the middleware also has a graphical view of any ECL transformation that is run on the Batch Layer, allowing you to see specifics of performance, data distribution, etc. for any query processed.

Extensible Within the Batch Layer, query attributes can contain and combine an infinite number of other query attributes to generate new views for the Server Layer. The Batch Layer's language, ECL, can reference external Java and C++ libraries, and also obtain information from other locations via Web Services.

Fault Tolerant HPCC has processes in place to handle node failure in both the Batch and Serving Layers with no impact to end users, and no loss of data. Any data corruption or mistakes made during processing of views can be fixed by re-computing the views and queries from the copy of the raw data stored on the Batch Layer.

Ad-hoc Querying The Batch Layer allows for ad-hoc querying against any data. Both the raw data and all views computed from it are available for ad-hoc queries, which can easily be converted into Serving Layer views as needed.

Maintainable The number of different layers of software in the HPCC Batch Layer and Serving Layer are minimal. Hardware failures are easily handled, and the clusters are also easy to expand.

Generic HPCC can handle a wide range of data (CSV, XML, BLOB, and more) and perform any transformation or process upon that data that can be imagined. It has been used for legal, medical, and analytic systems across several industries, handling each domain's data as easily as any other's.

Summary

Hopefully this information will be valuable to those evaluating and building systems for big data analytics, and confirm that an HPCC Systems platform implements the best practices which have come to be expected of big data processing platforms.

For more information on the Lambda Architecture, read the original chapter on the topic in Nathan Marz's book, A new Paradigm for Big Data. More information about HPCC Systems can be found at the HPCC Systems website. If you want to do some hands-on exploring, you can spin up your own HPCC System on the Amazon Cloud, or as a Virtual Machine Image . Finally, information on other platforms' implementation of Lambda is described at http://www.drdobbs.com/database/applying-the-big-data-lambda-architectur/240162604.

For More Information

Call 877.316.9669 or visit http://hpccsystems.com

About LexisNexis® Risk Solutions

LexisNexis® Risk Solutions (www.lexisnexis.com/risk/)is a leader in providing essential information that helps customers across all industries and government predict, assess and manage risk. Combining cutting-edge technology, unique data and advanced scoring analytics, Risk Solutions provides products and services that address evolving client needs in the risk sector while upholding the highest standards of security and privacy. LexisNexis Risk Solutions is part of Reed Elsevier, a leading publisher and information provider that serves customers in more than 100 countries with more than 30,000 employees worldwide.

About HPCC Systems®

HPCC Systems® from LexisNexis® Risk Solutions offers a proven, data-intensive supercomputing platform designed for the enterprise to process and deliver Big Data analytical problems. As an alternative to Hadoop and mainframes, HPCC Systems offers a consistent data-centric programming language, two processing platforms and a single architecture for efficient processing. Customers, such as financial institutions, insurance carriers, insurance companies, law enforcement agencies, federal government and other enterprise-class organizations leverage the HPCC Systems technology through LexisNexis® products and services. For more information, visit http://hpccsystems.com.

LexisNexis®