Weekly Report – 46

Xiufeng Liu

University of Waterloo, CA
toxiufeng@gmail.com

1 This Week

On Monday and Tuesday of this week I attended the training of IBM cloud computing platform, Aether (see Section 3), and the postdoc seminar of the SOSCIP project in IBM research center, Toronto. Stephen also gave me the orientation training about the company of IBM, the time-sheet recording, the project and organization of SOSCIP project. From Wednesday to Friday, I attended the ISS4E group meeting, took the course, met with my supervisor, Lukasz, and two others members of big data analytics team, Xiang and Zahid. In the small group meeting, we have envisioned the system that we will build in the next two years, and discussed some challenges of the work. With regards to the real-time integration and analytics for big data, in recent I read a book, *Big Data: Principles and Best Practices of Scalable Realtime Data Systems* [1] authored by Marz. He proposed a so-called lambda architecture for big data integration and analytics. I think it is very useful to building our system. Based on his idea, I have written something about the two-layered lambda architecture in Section 4.

The impediments of this week: I could not access the Internet in my office until Thursday since my university ID had not been ready.

2 Next Week

I expect I will get the accounts of our cluster, including typhoon and snowball servers, and the account of the database. I will make appointment with Xiang and Zahid on discussing the existing setup in the cluster. On Monday I will meet with Xiang and Zahid. They will go over with me about the existing setup of the clusters, and their course project. In next week I will be familiar myself with the cluster, and the software packages installed in the cluster. I also expect I will get the water and electricity data sets, and make the trail runs of data analytics. On Wedenesday Lukasz will give a presentation about the system architecture of the big data analytics in the group meeting. After that, I will draft the system architecture for the time series data analytics, investigate the technologies, make a plan for the cluster setup and development. In addition, continue the reading of smart energy papers, and go over the knowledge of statistics, data mining and machine learning.

3 IBM Cloud Computing Platform – Aether [2]

Aether is the cloud computing platform to support the SOSCIP project. The hardware architecture of Aether is called *x86 Agile Computing Platform*, which consists of a set of 42 IBM iDataPlex dx360 M3 nodes. Each node has the following specifications:

– Intel Xeon E5603 Quad-Core 1.6 GHz CPU
– 8 GB DDR3 RAM @ 1333 MHz
– 230 GB Hard Drive
– Red Hat Enterprise Linux 6.2 64-bit

Each node is connected to the main network via 1 Gigabit Ethernet. Interconnects between the nodes are Infiniband QDR 4X 20/40G.

To use Aether cloud, users should first make a reservation of the virtual machines based on the selection from a list of available images. During the reservation, users should sepcify the time range of using the cloud. Currently, there are a number of types of the VMs for selections, including the VM for numerical computing (installed with Matlab, Octave, and R), VM DB2 servers, and VM for IBM inforSphere Stream, etc. For data persistance, any files stored the local file system will lost at the end of the reservation. Howerver, if users want to keep the data, two persistent data stores are available on the VMs in Aether cloud. One is called *Sharcnet* and the other is called *Aether*.

For integration time-series data, the worthy noted technology is *IBM InfoShpere Streams* and *Stream Studio*. Users can reserve the virtual machine based on the InfoSphere Streams environment. To my understanding from the presentation, the functionalities of InfoSphere Streams are quite much like the open source streaming framework, *Storm* [3], which offers the on-the-fly data transformations, the readers and sinks for different data storage systems. The Stream Studio can facilitate the development of stream applications.

Harry Liu is the administrator of the Aether Cloud. For any questions about using the cloud, we could contact with him at *email: harryl@ca.ibm.com and phone: 519-661-2111 ext 81229*.

4 Real-time Integration and Analytics

We are witnessing a paradigm shift from batch based data processing to real-time data processing using the Hadoop framework. Despite this progress it is still a challenge to process web-scale data in real-time. However, the requirement of today's business users are demanding to view both historic and real-time data simultaneously. Many organizations find the two challenges of extracting real-time data and analyzing immense volumes of data converge with time. Real-time data accumulates and the inevitable demand for an aggregated historic view requires batch processing. And the batch processing solution is slow, which eventually leads to business users or customers asking to get immediate or near real-time insight, such as the most recent data updates to react faster to market changes.

A lot of technologies can be used to create such a complete data processing system - but to choose the right tools, to incorporate and orchestrate them is complex and daunting. One solution to this problem is the two-layered Lambda Architecture proposed by Marz [1]. He defines the most general data system that runs arbitrary functions on arbitrary data using the following equation "query = function(all data)". The Lambda Architecture defines a clear set of architectural principles for building robust and scalable data systems that obey this equation.

Figure 1 depicts the Lambda Architecture. The idea is simple and two-fold. First, a batch layer computes views on the collected data such as in HDFS, and repeats

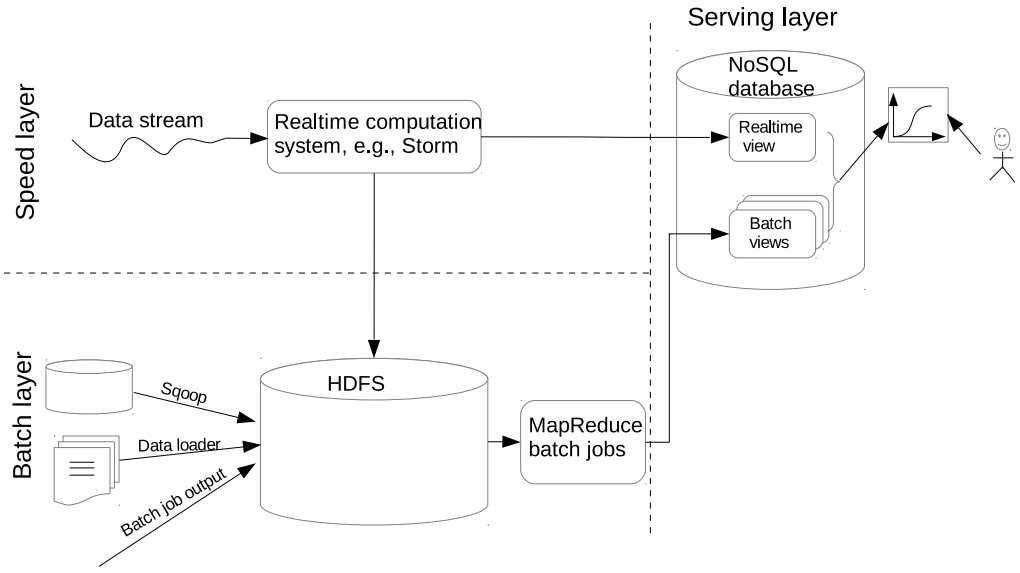

Fig. 1. Two-layered lambda architecture

the process when it is done to infinity. Its output is always outdated by the time it is available since new data has been received in the meantime. Second, a parallel speed processing layer closes this gap by constantly processing the most recent data in near real-time fashion. Any query against the data is answered through the serving layer, , by querying both the speed and the batch layers' serving stores in the serving layer, and the results are merged in response to the query near real-time.

The batch and speed layers both are forgiving and can recover from errors by being recomputed, rolled back (batch layer), or simply flushed (speed layer). A core concept of the Lambda Architecture is that the (historical) input data is stored unprocessed. This concept provides two advantages. Future algorithms, analytics, or business cases can retroactively be applied to all the data by simply adding another view on the whole data. Human errors like bugs can be corrected by re-computing the whole output after a bug fix. Many systems are built initially to store only processed or aggregated data. In these systems errors are often irreversible. The Lambda Architecture avoids this by storing the raw data as the starting point for the batch processing.

The Lambda Architecture itself provides only a paradigm. The technologies with which the different layers of a Lambda Architecture are implemented are independent from the general idea. The speed layer deals only with new data and compensates for the high latency updates of the batch layer. It can typically leverage stream processing systems, such as Storm [3], S4 [4], and Spark [5], etc. The batch serving layer can be very simple in contrast. It needs to be horizontally scalable and supports random reads but is only updated by batch processes. The technologies like Hadoop with Cascading [6], Scalding [7], Python streaming, Pig [8], and Hive [9] are suitable for the batch layer. The serving layer requires a system that has the ability to perform fast random reads and writes. The data in the serving layer is replaced by the fresh real-time data, and the views computed by the batch layer. Therefore, the size of data is relatively small, and the system in serving layer does not need complex locking mechanisms since the views are replaced in one operation. The in-memory storage solutions like, Redis [10] or Memchache [11], are sufficient for this

layer. If users require high availability, low latency and storing large volume of data in the serving store, the systems like HBase [12], Cassandra [13], ElephantDB [14], MongoDB [15], or DynamoDB [16] can be the potential options.

5 Reading

- Read the papers in smart energy course reading lists, and the papers recommended by Lukasz
- Go over the statistics and machine learning

References

1. N. Marz and J. Warren. "Big Data: Principles and Best Practices of Scalable Realtime Data Systems". Manning, 2013
2. Aether Cloud `https://aether.sharcnet.ca` as of 2013-11-17.
3. Storm, distributed and fault-tolerant realtime computation. `http://storm-project.net` as of 2013-11-17.
4. S4: Distributed Stream Computing Platform, `http://incubator.apache.org/s4/` as of 2013-11-17.
5. Spark Streaming, `https://www.cs.duke.edu/~kmoses/cps516/dstream.html` as of 2013-11-17.
6. Cascading, `http://www.cascading.org/` as of 2013-11-17.
7. Scalding, `https://github.com/twitter/scalding/wiki` as of 2013-11-17.
8. Pig, `http://pig.apache.org/` as of 2013-11-17.
9. Hive, `http://hive.apache.org/` as of 2013-11-17.
10. Redis, `http://redis.io/` as of 2013-11-17.
11. Memcache, `http://memcached.org/` as of 2013-11-17.
12. HBase, `http://hbase.apache.org/` as of 2013-11-17.
13. Cassandra, `http://cassandra.apache.org/` as of 2013-11-17.
14. ElephantDB, `https://github.com/nathanmarz/elephantdb` as of 2013-11-17.
15. MongoDB, `http://www.mongodb.org/` as of 2013-11-17.
16. DynamoDB, `http://aws.amazon.com/dynamodb/` as of 2013-11-17.