# Flying KIWI: Design of SQL-on-Hadoop System for Interactive Big Data Analytics

Sung-Soo Kim
Electronics and Telecommunications Research Institute (ETRI)
Daejeon, South Korea
*sungsoo@etri.re.kr*

***Abstract*—Business Intelligence (BI) and data warehouses workloads that they run are an important and growing field of the database industry. A large fraction of BI workloads are long-running batch query workloads that are run repeatedly. For instance, enterprises run report-generation workloads on a frequent basis to analyze customer and sales activity. Such batch query workloads are critical for operational and strategic planning, so they have to run and managed efficiently.**

***Keywords*-Database, Data Warehouse, SQL-on-Hadoop, Approximate Query Processing, Graphics Processing Unit, Kernel Estimation**

## I. Introduction

**Big Data Analytics Workload:** Transactional processing is characterized by a large number of short, discrete, atomic transactions. The emphasis of *online transaction processing* (OLTP) systems is (a) *high throughput* (transactions per second), and (b) maintaining *data integrity* in multi-user environments. Analytics processing is characterized by fewer users (business analysts rather than customers) submitting fewer requests, but queries can be very complex and resource-intensive. Response time is frequently measured in tens to hundreds of seconds.

Big data is in many ways an evolution of data warehousing [1]. Business Intelligence (BI) and data warehouses workloads that they run are an important and growing field of the database industry. A large fraction of BI workloads are long-running batch query workloads that are run repeatedly. For instance, enterprises run report-generation workloads on a frequent basis to analyze customer and sales activity. Such batch query workloads are critical for operational and strategic planning, so they have to run and managed efficiently.

On the other hand,

Parallel and distributed database systems, such as Hadoop and Spark, are essential components of todays infrastructure for Big Data analytics. These systems process multiple concurrent workloads consisting of complex user requests, where each request is associated with an (explicit or implicit) service level objective. For example, the workload of a particular user or application may have a higher priority than other workloads. Or, a particular workload may have strict deadlines for the completion of its requests.

To realistically address workload management issues, we should understand the workload behavior of clusters. Workload characterization studies are useful for helping Hadoop operators identify system bottleneck and figure out solutions for optimizing performance. To facilitate the understanding of MapReduce workload analysis in this paper, this section first gives a brief overview of the MapReduce framework and its implementation - Hadoop, with an illustration about how a MapReduce job executes in Hadoop.

**MapReduce Framework:** MapReduce clusters offer a distributed computing platform suitable for *data-intensive* applications. MapReduce was originally proposed by Google and its most widely deployed implementation, Hadoop, is used by many companies including Facebook, Yahoo and Twitter. MapReduce uses a *divide-and-conquer* approach in which input data are divided into fixed size units processed independently and in parallel by *map* tasks, which are executed distributedly across the nodes in the cluster. After the map tasks are executed, their output is shuffled, sorted and then processed in parallel by one or more *reduce* tasks. MapReduce is a parallel and distributed framework proposed by Google for processing large datasets. The fundamental concept of MapReduce is to distribute data among many nodes and process the data in a parallel manner. A MapReduce job consists of two phases: map and reduce. In the map phase, input data is divided into independent chunks and each chunk is assigned to one of the compute nodes. Each chunk is processed by a map task in a completely parallel manner, yielding intermediate key-value pairs data. This intermediate data is the output of the map-phase, as well as the input for the reduce phase. In the reduce phase, the input data is aggregated, summarized, filtered, or combined in some way, forming the final output. Typically, both the input and the output of the job are stored in a distributed file system. The MapReduce processing pattern remains the same, while the specific functions defined for the map and reduce phase change to fit specific problems.

**Approximate Query Processing:** Over the past two decades a large number of approximation techniques have been proposed, which allow for fast processing of large amounts of data by trading result accuracy for response time and space. These techniques include sampling, sketches, and on-line aggregation.

The rest of the paper is organized as follows.

## II. Related Work

## III. System Architecture

## IV. Implementation Details

## V. Experimental Results

## VI. Conclusion

## Acknowledgments

This work was supported by ETRI R&D program ("*Development of Big Data Platform for Dual Mode Batch Query Analysis*, 15ZS1400") funded by the government of South Korea.

## References

[1] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster Computing with Working Sets. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, pages 10–10. USENIX Association, 2010.