Distributed System based OLAP

Hao Wang and Wang Xi

Abstract— **As the volume of data soars recent years, OLAP has to handle more massive multidimensional data than before. With the maturity of distributed computing technology, it provides us an effective way to help OLAP tool handle enormous data. This survey focuses on the combination of OLAP and distributed systems. Based on related papers and existing projects, we explore the development status quo of the application of distributed systems on OLAP.**

I. INTRODUCTION

With the mature of distributed system technology, more and more applications can be transplanted to distributed system to speed up themselves. This survey focuses on distributed system based OLAP. We explore the combination of distributed system technology and OLAP technology. So we conduct the survey into three parts: distributed system, OLAP and the combination of these two technologies.

II. DISTRIBUTED SYSTEM

A. Distributed Database

Distributed database can be explained as a database where not all storage devices are attached to common processors[14].

Two sorts of processes will tell us that, distributed databases can be up-to-date and current: replication and duplication.

Replication, includes applying specialized software or technologies which looks for changes in distributive databases. And Once a change has been noticed, all the replication process will make sure all databases look the same, that is, keeping consistent. The replication process are able to be complex and time-consuming, which will be dependent on the size as well as scales of the distributed databases. And this process will also require many a time and many computer resources. **Duplication**, as opposed to Replications, has less complexity. Basically, it can identify one database as a coordinator, and then copy that database. After several hours, duplication processes are normally done at a set time , which aims to ensure that every distributed location has the same data. In duplication processes, users are able to switch only the coordinator database. Finally, this promises local data would not be overwritten by users.[6]

System administrators are able to distribute collections of data spreading upon many and various physical locations. One distributed database could also reside on decentralized independent points inside Internet or organized network servers(e.g. local internets), on corporate intranets or extranets, or on other organization networks. Since they keep data across multiple spots like computers, distributed databases can improve performances for end-user worksites through permitting transactions to be going on many servers like machines, which will substitute limited to one machine.[17]

In current stage, distributed database management system market and industry are marching dramatically with fresh, innovative entrants in support of the increasing and fostering use of unstructured data as well as NoSQL DBMS engines, and XML databases even NewSQL databases. All of these databases are helping supporting distributed database architecture that offers high availability and fault-tolerance through replication and scale out ability. Some examples are Aerospike[15], Cassandra[10], Clusterpoint[13], Riak[9] and OrientDB[11]. This set of block chaining technologies popularised by bitcoin technology has becoming a kind of implementation of distributed database.

Homogeneous Distributed database means a set of database have identical software and hardware executing on all databases instances, and may appear by one single interface as though it were a single database.

While Heterogeneous Distributed database means a set of database share different hardware as well as operating systems, DBMS, and even

data models.

B. Mapreduce

Purpose: MapReduce technology means a kind of programming model. It is illustrated as an associated implementation for processing and generating a great number of datasets which is amenable for broad varieties of true existing tasks.[6] The purpose to develop MapReduce is for calculating different sorts of achieved data such as inverted indices, graph structures of Web documents that have various expressions, abstracts of the number of pages scratched of every host, as well as the set of frequently sending queries within designated days. Many computations like above cases are conceptually straightforward. Whereas, coming input data does not fit in most large. Meanwhile, distributed computations also have to be across thousands of machines in some limited time period to be down. To help analyze this, there is a set of experiments designed by [Dean et. al] about a new abstraction that allows the authors to express and show the simple computations they were trying to display whereas stuck in the messy details of parallelization, fault tolerance, data distribution and load balancing in libraries, inspired by Lisp.[6]

Programming: The calculation and assessment will take a set of input key-value pairs, which helps to produce a set of output key-value pairs. MapReduce library's users will express the computation through map and reduce two functions. Map, written by the user, takes an input pair and produces a set of intermediate key/value pairs. The MapReduce library groups aggregates all intermediate values concerning with same intermediate keys I and transfers each of them to the "reduce" function. The "reduce" function written by users will accept this intermediate key I as well as a set of values attached with that key. It can merge all of relevant values coming from "reduce" function together to build a probably smaller size set of values. Empty or one output value typically will be produced for each reduce activation. And the intermediate values accepted from users are supplied to the reduce function through a kind of iterator. Implementation like this allows users to manipulate many lists of values which are too large to fit in memory.[6]

Implementation: There are many possible and various implementations of the MapReduce interface. The proper selection depends on the circumstance. The "map" invocations work as a distributed mechanism across multiple machines by automatically segmenting the input data to consist of a set of M splits. The input splits then could be processed concurrently by different machines. The "reduce" invocations work as a distributed mechanism with a partitioning function by partitioning the intermediate key space into R pieces. Note that the R partitions and the partitioning functions will be indicated by users.[21]

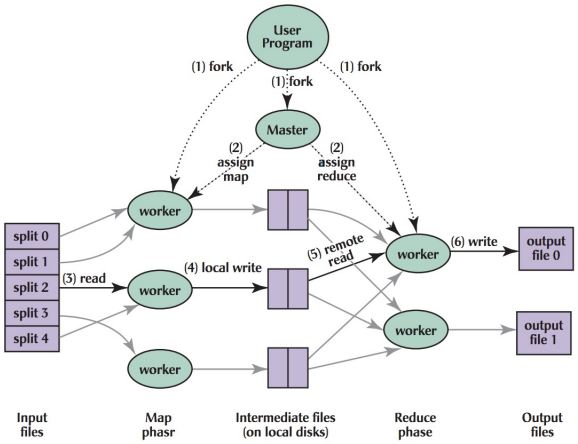

Fig-1. execution overview

Refinement: Besides map and reduce, another useful functions also reach out to users: user-specified partitioning, ordering guarantees, user-specified combiner,custom input and output type,a mode for execution on a single machine.[6]

Fault-tolerance: The MapReduce implementation uses a pull model mapper and a reducer for moving data between the two, rather than a push mode reducer that mapper writes directly. The pull model is able to lead to the establishment of great amount of small files, while many disk looks forward to moving data between mappers and reducers, correctly introduced by Pavlo's researches. Google's MapReduce tricks of implemention includes some tricks like batching, sorting, and grouping of intermediate data as well

as smart scheduling of reads to abate the great costs about fault-tolerance. [7]

Performance: Dean gave a performance computation example in 2008 [6]. [19] One computation searches through approximately one Terabyte of data looking for a particular pattern. The other computation sorts approximately one Terabyte of data.

- Cluster Configuration: All of the programs were executed on a cluster that consisted of 1800 machines approximately. Each machine was distributed two 2GHz Intel Xeon processors with Hyper-Threading enabled, 4GB of memory, two 160GB IDE disks, and a gigabit Ethernet link.[6]
- Grep: The grep program scans through 10^{10} 100 byte records, seeking for a relatively rare three characters pattern. Meantime, the input data will be split into approximately 64MB pieces (M = 15000), and the whole output will be placed into a single one file (R = 1).[6]

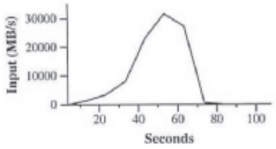

Fig-2. Data transfer rate over time (mr-grep)

- Sort: This experiment set less than 50 user lines to build up the sorting program. And its final sorted output will be written as a set of 2-way replicated GFS files (i.e., 2 terabytes are written as the output of the program). The input data will be split into 64MB pieces (M = 15000) as before. That tests partitioned the sorted output into 4000 files (R = 4000) and set the partitioning function segregating the output data into one of pieces using the initial bytes of the key. This partitioning function for this benchmark has built-in knowledge of the distribution of keys. In a general sorting program, the authors added a prepass MapReduce operation used for collecting a sample of the keys and used the distribution of sampled keys to compute split points for the next sorting pass.[6]

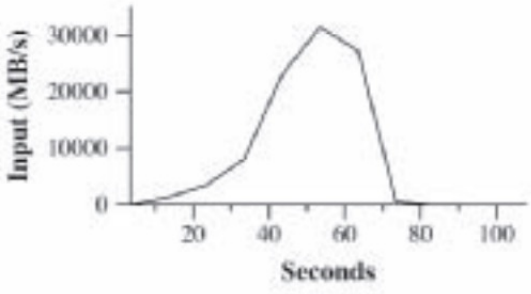

Figure 2: Data transfer rate over time (mr-grep)

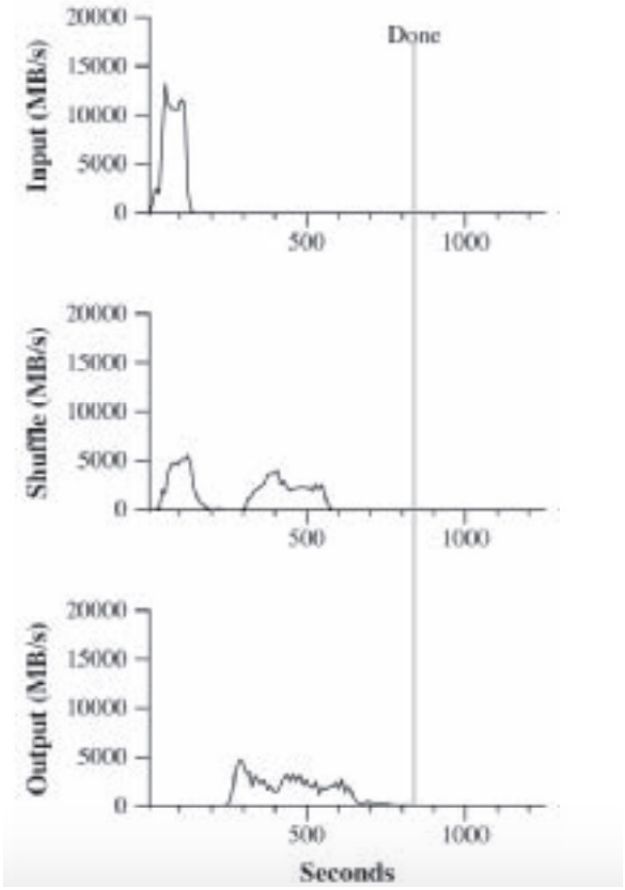

Fig-3. Data transfer rate over time (mr-grep)

Development in database: A great amount of significant advantages over parallel databases is brought by MapReduce. Firstly, it offers fine-grain fault tolerance for large scale works; if failures occur after multi-hours execution will not require a restarting operation from initial job scratch. Secondly, MapReduce grealy helps for dealing with data processing and data loading in a heterogenous system accompanied with many various storage systems. Thirdly, MapReduce also

provides fine framework for the implementation of more complicated functions as opposed to directly in support of SQL. [7]

Cutting-edge researches on MapReduce:

- MapReduce/Hadoop framework has been widely used to process large-scale datasets on computing clusters. Scheduling a map task using data locality considerations is critical to MapReduce's performance. Many works are dedicated to increasing the position of the data to improve efficiency. [22]
- Due to MapRduce's wide range of applications, skyline operators attracts greatly considerable from both research and industry fields. However, calculating a skyline is still a great challenge for handling large scale data. For data-intensive applications, the MapReduce framework also has recently been widely used to help solve it.[18]

C. Hadoop

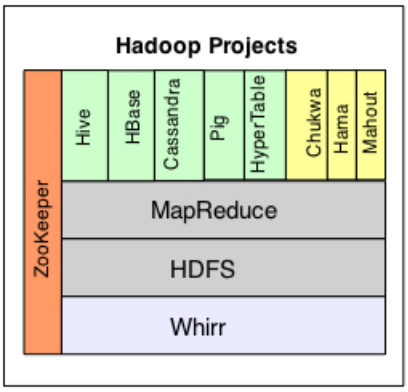

Fig-4. MapReduce architecture

Hadoop is one of open-source implementation of MapReduce developed by Apache, and undoubtedly the most popular MapReduce variant in current industries. Currently there is an increasing number of prominent companies with large amount of customers starting to use Hadoop, including companies such as Yahoo! and Facebook.[8]

Many companies including AOL, Amazon, Facebook, Yahoo and New York Times have successfully used Hadoop for implementing their applications on clusters. For instance, AOL used Hadoop aims to analyze its users' behavioral pattern to offer potential services by running Hadoop to process it. Apache Hadoop is a kind of open source solution of the Googles MapReduce parallel processing framework. Hadoop hides the details of parallel processing, including data distribution to processing nodes, restarting failed subtasks, and consolidation of results after computation. Google's framework permits engineers to set parallel executing programs which concentrate on their computation issues, rather than concurrency problems. Hadoop involves 1) Hadoop Distributed File System (HDFS): a distributed file system that store large amount of data with high throughput access to data on clusters and 2) Hadoop MapReduce: a software framework for distributed processing of data on clusters.[20]

- **HBase** -maps HDFS dataset into a database-like structure, and offers JAVA API to database. Hbase can solve large scale data like million row tables with any column counts. Hbase is broadly known an outgrowth of Bigtable that takes great effort in same functions to against GFS. Compared with Hive, Hbase may not execute above MapReduce rather replace it, but it also can be used as origin or destination of MapReduce operations. Meanwhile, Random access, which is used to support transaction oriented applications, can be brought into Habse by somewhat tuning. Moreover, Hbase can run on HDFS or Amazon S3 infrastructure.[2]
- **Hive** - Apache Hive is a data warehouse infrastructure built on top of Hadoop for providing data summarization, query, and analysis. Hive maps QL on top of a data warehouse established upon Hadoop framework. Some of these queries may take a long time to execute and as the HDFS data is unstructured the map function must extract the data via relational schema. Hive runs above of Hadoops MapReduce function.[2]
- **Hypertable** - Hypertable was an open-source software project to implement a database management system inspired by publications on the design of Google's Bigtable. Factually, any column oriented DFS can be used in Hypertable but only be in support of columns and column families. Based on C plus implementation and only applied HDFS rather than GFS, Hypertable gives a client c plus

and Thrift API. Some considerations occur here that, whether not C++ based Hypertable should be the most optimized Hadoop oriented databases or not. [2]

- **Pig** - is scripting language;a kind of dataflow processing syntax built upon Hadoop DFS, which helps to build a kind of database interpreter combined with interpretive analysis tool. Pig basically applies the scripting language and issues a data flow diagram, which is then used by MapReduce to analyze data in HDFS. And both batch and interactive execution have been supported in Pig by using JAVA API.[2]
- **ZooKeeper** - is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. A sort of cluster configuration tool aiming to seek out a solution to manage distributed serialization.[2]

III. OLAP

On-line analytical processing (OLAP) can be regraded as an essential data warehouse modeling. We introduce OLAP by its motivation and history, OLAP cube, operations and types of OLAP.

A. Motivation and History

With the development and application of database, the amount of data stored in database soars recent decades and the requirements of users for database are not only searching or manipulating from limited records in database but analyzing data and integrating information from even more than hundred million records from database. On-Line Transaction Processing(OLTP) is hard to handle these complicated requirements. To meet the requirements of users, the concept of on-line analytical processing was proposed by Edgar Frank "Ted" Codd. Briefly, on-line analytical processing is an approach to answer multi-dimensional analytical queries swiftly in computing.[4] Since on-line analytical processing is so powerful, OLAP is used in business intelligence to explore data and execute data mining to gather and analyze significance data from large mass data to support human make decisions. With time goes by, data warehousing and on-line analytical processing (OLAP) are essential elements of decision support, which has increasingly become a focus of the database industry.[3]

B. OLAP cube

The key idea of on-line analytical processing is building OLAP cube, which is a multi-dimensional array of data, then executing operations on OLAP cube. By integrating data from database to OLAP cube, users can execute different operations on on-line analytical processing cube to collect and analyze expected data from dimension, level, member and measure of OLAP cube based on different purposes of users swiftly.

1) Dimension: Dimension of OLAP cube is a set of feature such as all data belongs to supermarkets which located in Urbana.

2) Level: The level of dimension is that details of this dimension in different aspects such as day, month or year in time dimension.

3) Member: While, member is what its literal meaning, its member of specific dimension in OLAP cube, such as all day belongs to April 13 in time dimension.

4) Measure: Measure can be regarded as the basic unit of OLAP cube, the value retrieved form multi-dimensions, such as the sale data from Walmart in Urbana on April 13.

C. Operation

After we know the basic structure of OLAP cube, we can execute different operations on OLAP cube to analyze data by different approach. Basic operations can be listed as roll up, drill down, slice and dice, pivot:

1) Roll up: Roll up is also called drill up. Its mainly for summarization. The basic idea is summarizing data from low level to high level. Another way is reducing dimension to collect detail data. More specific, roll up is summarizing sub-dimension detail data of a specific dimension to obtain high level data such as collecting the sale data of all supermarkets in 2016 at Urbana then we summarize all sale data of all supermarket located in Urbana from January to December in 2016. To summarize data based on specific dimension, we can execute roll up operation and the rule of summarization can be decided by users such

as computing totals along a hierarchy or applying a set of formulas such as ”profit = sales - expenses”.[5]

2) Drill down: Drill down is a reverse approach of roll up. This operation to help users focus on detail data by dividing high level data into low level(sub-dimension) or importing new dimensions to generate detail data we interested in from original OLAP cube.

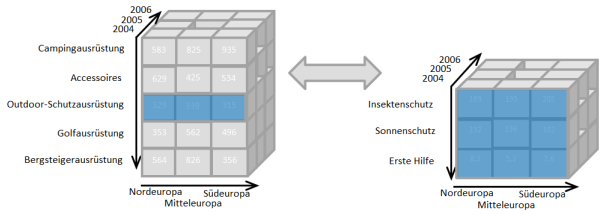

Fig-5 OLAP drill down and up[12]

3) Slice and Dice: Slice and dice can be regarded as select and project on OLAP cube. The main difference of slice and dice is that if the OLAP cube being processed is only two dimensions, the operation is called slice, otherwise, it can be defined as dice, which the result is sub-cube.

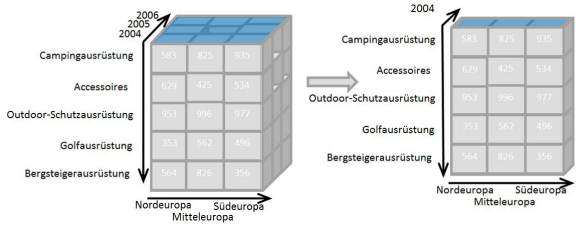

Fig-6 OLAP slicing[12]

4) Pivot: Pivot is the operation that reorients OLAP cube to offer OLAP cube with different dimensions arrangement to help users explore information via different perspectives on the data.[3]

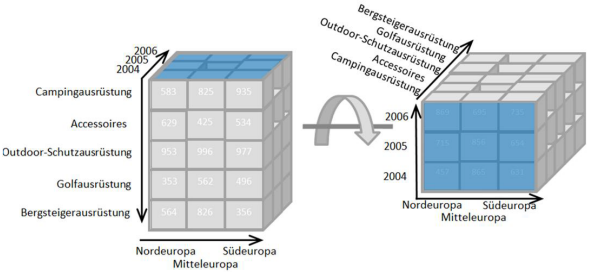

Fig-7 OLAP pivot[12]

D. Types

Depending on different formats of storages of on-line analytical processing, OLAP can mainly be divided into three types: MOLAP (Multidimensional OLAP), ROLAP (Relational OLAP) and HOLAP (Hybrid OLAP).

1) Multidimensional OLAP: MOLAP (Multidimensional OLAP) is based on multi-dimensional array storage. Basic idea is that each dimension data stored in different array to construct OLAP cube structure. Since data stored in array, the index of array can be useful to label value of each dimension. Thus, its fast to execute query and the structure of OLAP cube is easy to interpreted.

2) Relational OLAP: ROLAP(Relational OLAP) is based on relational database. Comparing with MOLAP, ROLAP is more suitable for large data volumes. Since data stored in relational database, approaches of database can be applied to ROLAP.

3) Hybrid OLAP: HOLAP(Hybrid OLAP) is a hybrid of multidimensional OLAP and relational OLAP. Data of hybrid OLAP can be stored in both multi-dimensional array and relational database, depending on the users' decisions. Because HOLAP is an integration of MOLAP and ROLAP, it carries on the advantages of both OLAP types and avoid shortcomings of MOLAP and ROLAP.

IV. MR-CUBE

It is widespread for building OLAP on distributed system in recent years. More and more OLAPs are built based on distributed system. This trend prompts people pay more attentions on the combination of these two technologies. Because there are too many approaches proposed recent years, we introduce the approaches proposed in two most representative and influential papers to help understanding the basic ideas of building OLAP data cube and computation of OLAP cube over mapreduce and comprehending optimized approaches proposed in these papers.

A. Building Cube

The key of OLAP is building OLAP data cube. So the approaches of building data cube become highest priority task. There are three approaches

we have used to build data cube with mapreduce explained in [1]

1) Full Source Scan: Full source scan uses HBase which we mentioned in distributed system part. As its literal meaning, this approach is to scan the whole source. It uses the required attributes labels user indicated to filtering the source and extract data we are interested in. [1]

2) Indexed Random Access: Indexed random access is the approach that builds indexes beforehand to reduce complexity. Since we have indexes we built before extracting, it is easy to extract the tuples we are looking forward to when we uses the indexes.[1] After that, this approach extracts data by random access.

3) Index Filtered Scan: Full source scan and indexed random access has their own advantages and disadvantages. While, we can integrate these two approach to obtain their advantages and avoid their disadvantages. This combination is called index filtered scan. Index filtered scan integrates these two approaches we mentioned above. For example, when we extract specific tuples we are interested in, we can build indexes and an in-memory bitmap to help speed up the extraction. Besides, we can uses the indexes and bitmap which built in memory to filter tuples in map function(map step in distributed system) when we scan the whole database.[1]

B. Cube Computation

After we know general methods to build MR-cube, we should move to understand how to do computation on MR-cube. There are many approaches for on-line analytical processing data cube computation over mapreduce. Overall, basic idea of MR-cube computation is that dividing computation into many pieces(map step) to avoid reduce step(reduce step in distributed system) handles too much computation.[16]

1) Partially Algebraic Measures: Partially algebraic measures proposed in[16] provides us a nice way to compute in parallel. The main idea is that build reducer-friendly and reducer-unfriendly groups, then handle each group respectively. Partially algebraic measures can be used to compute subgroups which are full tuple that are mutually exclusive.[16] Besides, partially algebraic measures can also be used to handle mutually exclusive tuples after projecting on the algebraic attributes. [16]

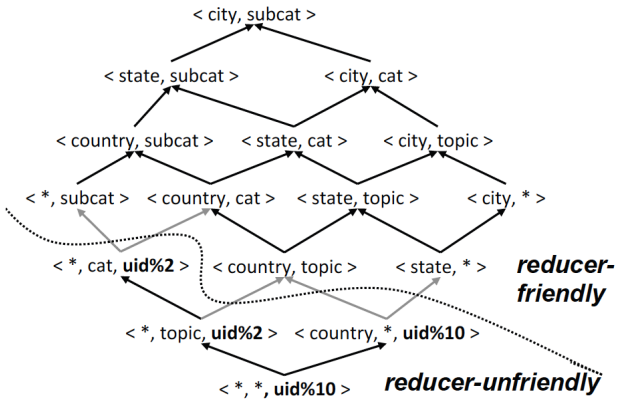

Fig-7 Partially Algebraic Measures[16]

2) Sampling Approach: Sampling approach is the statistics method to estimate the number of reducer-unfriendly groups on each cube region. The basic idea is that we generate a small sample which is extracted randomly from dataset(data cube); then, do cube computation on this sample; after that, analyze the result to predict the reducer-friendly groups and reducer-unfriendly groups we mentioned above. More specific, if the result of computation on sample is more than 0.75rN tuples of cube group G, where N is the size of sample and $r = \frac{c}{|D|}$, we declare group G to be reducer-unfriendly.[16] This sampling approach helps us obtain overview of reducer-friendly and reducer-unfriendly groups on entire data cube. After executing this approach, we can divide data into reducer-friendly and reducer-unfriendly groups to handle them respectively to speed up MR-cube computation.

3) Batch Areas: Batch areas is proposed in[16]. Its aimed to handle the duplicate and incompatibility problem when we pruning with monotonic measures. The framework of batch areas is that for map step, it emits a key-value pair on each batch for each data tuple to reduce the intermediate data we generated during MR-cube computation.[16]; for reduce step, reducer just need to execute traditional cube computation. But the traditional cube computation algorithm works over the set of tuples we generated from map step which is using the batch area as the local cube frame rather than

traditional cube frame.[16]

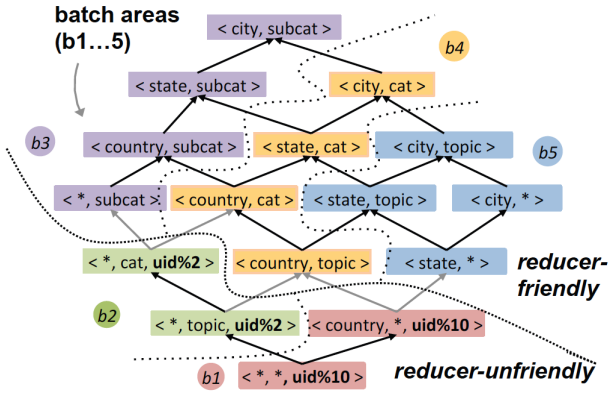

Fig-8 Batch Areas[16]

V. Conclusion

After we analysis distributed system, OLAP and integration of them, we find distributed system and OLAP are powerful and include different approaches to handle specific problems. As a integration of these two technologies, distributed system based OLAP takes advantages of both of them and enhance mutually.

References

[1] Alberto Abelló, Jaume Ferrarons, and Oscar Romero. Building cubes with mapreduce. In *Proceedings of the ACM 14th international workshop on Data Warehousing and OLAP*, pages 17–24. ACM, 2011.

[2] RayonStorage blog. Hadoop introduction. 2011.

[3] Surajit Chaudhuri and Umeshwar Dayal. An overview of data warehousing and olap technology. *ACM Sigmod record*, 26(1):65–74, 1997.

[4] Edgar F Codd, Sharon B Codd, and Clynch T Salley. Providing olap (on-line analytical processing) to user-analysts: An it mandate. *Codd and Date*, 32, 1993.

[5] OLAP Council. Olap and olap server definitions, 1997.

[6] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[7] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: a flexible data processing tool. *Communications of the ACM*, 53(1):72–77, 2010.

[8] Christos Doulkeridis and Kjetil Nørvåg. A survey of large-scale analytical query processing in mapreduce. *The VLDB Journal*, 23(3):355–380, 2014.

[9] http://basho.com/. Basho riak distributed database. 2017.

[10] http://cassandra.apache.org/. Apache cassandra database menagement system. 2017.

[11] http://orientdb.com/. Orientdb database. 2017.

[12] https://commons.wikimedia.org/w/index.php?curid=15420041. Infopedian. 20111.

[13] https://www.clusterpoint.com/. Clusterpoint xml distributed database. 2017.

[14] https:www.its.bldrdoc.govfs1037dir 012_1750.htm. distributed database. 1997.

[15] http://www.aerospike.com/. Aerospike distributed database. 2017.

[16] Arnab Nandi, Cong Yu, Philip Bohannon, and Raghu Ramakrishnan. Data cube materialization and mining over mapreduce. *IEEE transactions on knowledge and data engineering*, 24(10):1747–1759, 2012.

[17] James A O'Brien and George M Marakas. *Management information systems*. McGraw-Hill Irwin, 2006.

[18] Yoonjae Park, Jun-Ki Min, and Kyuseok Shim. Efficient processing of skyline queries using mapreduce. *IEEE Transactions on Knowledge and Data Engineering*, 2017.

[19] Michael O Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM (JACM)*, 36(2):335–348, 1989.

[20] B Thirumala Rao and LSS Reddy. Survey on improved scheduling in hadoop mapreduce in cloud environments. *arXiv preprint arXiv:1207.0780*, 2012.

[21] Kyuseok Shim. Mapreduce algorithms for big data analysis. *Proceedings of the VLDB Endowment*, 5(12):2016–2017, 2012.

[22] Weina Wang, Kai Zhu, Lei Ying, Jian Tan, and Li Zhang. Maptask scheduling in mapreduce with data locality: Throughput and heavy-traffic optimality. *IEEE/ACM Transactions on Networking*, 24(1):190–203, 2016.