Methods of Processing Massive Data

Cao Wei	Hao Jie	Jin Shan	Zhang Cheng
Department of IIIS	Department of CS	Department of CS	Department of CS
Tsinghua University	Tsinghua University	Tsinghua University	Tsinghua University
Stu.ID 2013311397	Stu.ID 2013310635	Stu.ID 2013380060	Stu.ID –

Abstract

Traditional methods are restricted in modern IT since the quantity of the data becomes extremely huge. Instead, some technical methods of processing the massive data are proposed. In this paper, we introduced some typical methods in processing massive data of three difference stages in data processing, storage, executing, and knowledge acquisition. Moreover, we gave a brief introduction of a theoretical model based on BigData which might be a crucial model in future IT technology.

1 Introduction

With the rapid development of internet, many kinds of applications such as social network and online video streaming generate large amount of data. Processing these massive data is called *BigData Processing*. In BigData processing, we mainly focus on the data storage, algorithm executing and knowledge mining.

Storage The distributed storage system to widely used to store the data. In order to keep the data safe, the system should be fairly reliable. Distributed storage system usually introduces redundancy to increase the reliability. In distributed storage, the data file is dispersed across the storage nodes in network. An end-user can retrieve the data by contacting a subset of the nodes. The most commonly used redundancy schemes are data replication (e.g. Google File System) and erasure code (e.g. OceanStore).

The strategy of data replication is very simple and easy to manage, we just need to replicate the data file as many time as we want. For example, In Google File System, each file is replicated for three times. Replication is the most simple redundancy strategy. However, it has the drawback of low storage efficiency. By using the strategy of replication, we need a large number of disks to store those copies of the data file. As for the strategy of erasure code, it has high storage efficiency compared to the strategy of replication. However, it is hard to maintain the reliability. That means that when a storage node fails, the repair cost to repair the failed storage node is very high. The erasure codes used in storage system are usually (n, k) maximum distance separable (MDS) codes.

Executing MapReduce is a programming model for executing the algorithms in large data sets with a parallel, distributed algorithm on a cluster. A MapReduce program is composed of a Map() procedure that performs filtering and sorting (such as sorting students by first name into queues, one queue for each name) and a Reduce() procedure that performs a summary operation (such as counting the number of students in each queue, yielding name frequencies). The MapReduce System orchestrates by marshaling the distributed servers, running the various tasks in parallel, managing all communications and data transfers between the various parts of the system, and providing for redundancy and fault tolerance.

Knowledge Mining Under the Big Data trend, tabular data on the Web has become a rich source of structured data that is useful for ordinary users to explore. According to a recent Google study

[1], there are a total of 14 billion raw HTML tables and, among those, 150 million relational data tables, easily making Web tables one of the richest structured data sources. There are many real-world applications that can benefit from high quality Web tables. First, search engines can utilize Web tables to improve the result quality. Existing search engines usually return relevant Web pages for queries and require users to mine answers from the returned Web pages. Obviously if there are many relevant high-quality Web tables, search engines can utilize these high-quality structured data to directly compute the answers and do not require users to search answers from documents. Query-answer (QA) systems can also utilize Web tables to easily generate answers for queries. Second, many systems want to provide Internet users with OLAP functionality, which is not supported in traditional keyword-based search systems. For example, if a user wants to know the average price of iPhone 5 in Beijing. Search engines cannot answer such queries. If we can extract Web tables about prices of iPhone 5 in Beijing, we can utilize these Web tables to answer such queries. Third, knowledge bases can use Web tables to enrich themselves by adding more concepts, entities and relationships between columns. There are many research challenges in Web table understanding. The first challenge is how to identify large-scale high-quality Web tables. One main approach proposed new machine learning (ML) techniques to annotate table cells with entities that they likely mention, table columns with types from which entities are drawn for cells in the column, and relations that pairs of table columns seek to express. Another classic approach utilized traditional information retrieval (IR) technology, which enabled us to produce column concepts that are as good as those produced by ML based method, but in a much more scalable fashion.

Moreover, besides the huge quantity of data, it is also a serious problem to deal with the rapid and high dimensional data. In such a background, the data streaming model was proposed. It is a widely used model not only on IT but also in astronomic, communicating and economic fields. The first appearance of this model could be traced back to 20th century, but it is one of the hottest topic in recent years since the scale of the data becomes the bottleneck of modern IT development. Based on such model, the data streaming algorithms have the characteristics of high timeliness, low complexity and approximation. The approximation is the core of the data streaming algorithm. The real data in internet is actually occupied by the large quantity of redundancy. In this paper we introduce several classical data streaming algorithms to give a sketch how this model work on the modern data processing.

The rest of the paper is organized as follows. In section 2, we survey two important redundancy scheme, one is hierarchical code, the other is regenerating code. In Section 3 we describe three approach to identify large-scale high-quality Web tables. In Section 4 we describe MapReduce as the programming model for processing large data sets. In Section 5 we deal with the rapid and high dimensional data. In section 6 we give the conclusion.

2 Storage

2.1 Intuition

Since the amount of data is very large, we usually use distributed storage system to store the data. In order to keep the data safe, the system should be fairly reliable. Distributed storage system usually introduces redundancy to increase the reliability. In distributed storage, the data file is dispersed across the storage nodes in network. An end-user can retrieve the data by contacting a subset of the nodes. The most commonly used redundancy schemes are data replication (e.g. Google File System) and erasure code (e.g. OceanStore).

The strategy of data replication is very simple and easy to manage, we just need to replicate the data file as many time as we want. For example, In Google File System, each file is replicated for three times. Replication is the most simple redundancy strategy. However, it has the drawback of low storage efficiency. By using the strategy of replication, we need a large number of disks to store those copies of the data file.

As for the strategy of erasure code, it has high storage efficiency compared to the strategy of replication. However, it is hard to maintain the reliability. That means that when a storage node fails, the repair cost to repair the failed storage node is very high. The erasure codes used in storage system are usually (n, k) maximum distance separable (MDS) codes. With MDS code, the original data file is divided into k fragments, and then they are encoded into n fragments using a kind of MDS code.

After that these n fragments are dispersed into n storage nodes in network. At last, we can recover the original data file by downloading data from any k nodes out of these n nodes. This means that with the same amount of redundancy, the storage system using erasure code is more reliable than those using replication.

As the erasure code is more reliable, it is more suitable for the distributed storage system. However, despite of reliability, the distributed storage should also be efficient. When a storage node failure occurs in the storage system, in order to maintain the same reliability, we should introduce a new node to the system to repair the failed one. In an efficient storage system, the amount of data transferred during this process should be as low as possible. With erasure code, in order to repair the failed node, firstly, there is a need to reconstruct the whole original data file by downloading data from k nodes, then the lost data node is generated from the whole data file which is reconstructed in the first stage. In this repair process, the amount of data transferred is k times of the lost data. When nodes are distributed in the network, this means large cost of repair bandwidth. So the system is not very efficient. To improve the efficiency, it is very important need to reduce the cost of repair bandwidth.

Lots of research focuses on reducing the repair cost of erasure code. Hierarchical code uses less than k nodes in the repair process, each helper node uploading all the data it stores. Regenerating code employs the idea of network coding, and uses more than k nodes in the repair process, but each helper node uploads a linear combination of the data it stores. Both hierarchical code and regenerating code reduce the cost of repair bandwidth compared to erasure code.

2.2 Hierarchical codes

Hierarchical Code(HC) reduces the cost of repair bandwidth by employing less nodes in the repair process. Hierarchical code can be generated iteratively through its code graph, Fig. 1 shows the code graph of $HC(k, H)$ hierarchical code.

fig.1

Let n be the number of storage nodes in the system. The original uncoded data is supposed to be dispersed in k nodes, i.e. the number of systematic node is k. Therefore, the number of parity node is $h = n - k$. In Fig. 1, the data in each node of an $HC(k, H)$ code is generated by Eq. (1):

$$P_i = \begin{cases} S_i & i \leq k \\ \sum_{j=1}^{k} c_{i,j} S_j & k < i < k + h, c_{i,j} \in GF(2^q) \end{cases} \tag{1}$$

Step 1, Choose parameter k_0, h_0, and build $HC(k_0, h_0)$ code using Eq. (1). Generate h_0 parity nodes from $d_0 = k_0$ systematic nodes. $G_{d_0,1}$ denotes the collection of the k_0 systematic nodes and the h_0 parity nodes , we call $G_{d_0,1}$ as the code group of Layer 0. The combination degree of $G_{d_0,1}$ is d_0. In Fig. 1(a), $k_0 = 2, h_0 = 1$.

Step 2, Choose parameter g_1 and h_1. Replicate the group structure in Step 1 for g_1 times to get g_1 groups denoted as $G_{d_0,1}, ..., G_{d_0,g_1}$. Then generate h_1 parity nodes from $g_1 k_0$ systematic nodes. Denote $G_{d_0,1}, ..., G_{d_0,g_1}$ and the newly generated h_1 parity nodes as $G_{d_1,1}$. We call $G_{d_1,1}$ as code group of Layer 1. The combination degree of $G_{d_1,1}$ is $d_1 = g_1 d_0 = g_1 k_0$. The number pof parity nodes in $G_{d_1,1}$ is $H_1 = g_1 h_0 + h_1$. In Fig.1(b), $g_1 = 2, h_1 = 2, k_1 = d_1 = 4, H_1 = 4$.

Step 3, Repeat Step 2 to generate the code graph layer by layer. In Step i, choose parameter g_i and h_i. Replicate the structure of $G_{d_{i-1},1}$ for g_i times, then add h_i parity nodes by encoding $g_i d_{i-1}$ systematic nodes. Denote this code group as $G_{d,1}$. $G_{d,1}$ is code group of Layer i. In $G_{d,1}$, $k_i = d_i = g_i d_{i-1}$, $H_i = g_i H_{i-1} + h_i$. We denote the code as $HC(k_i, H_i)$ code.

In Fig. 1(b), $G_{2,1}, G_{2,2}$ are code groups of Layer 0, $G_{4,1}$ is code group of Layer 1. Duminuco *et al*. established the following rules for data reconstruction and node regeneration.

Reconstruction Condition Consider B^k, a set of k nodes in the code graph of a $HC(k, h)$ code. If the nodes in B^k are chosen fulfilling the following condition:

$$|G_{d,l} \bigcap B^k| \leq d, \forall G_d$$

belonging the code which means that in B^k there can be a maximum of d nodes chosen from any code group G_d , then the nodes in B^k can reconstruct the original data.

Regeneration Condition Denote $G(p)$ as all the code groups which contain the failed node p, $R(p)$ as the set of nodes which helps in the repair process. If $\forall p$, $R(p)$ fulfils the following conditions:

$$|G_{d,i} \bigcap R(p)| \leq d, \forall G_{d,i}$$

and

$$\exists G_{d,i} \in G(p) : R(p) \subseteq G_{d,i}, |R(p)| = d$$

then the nodes in $R(p)$ are valid to repair the failed node p.

Hierarchical code reduces the number of nodes that participate in the repair process, so the cost of repair bandwidth is reduced. And less helper node also mean that the I/O cost in the repair process is lower.

2.3 Regenerating codes

Regenerating code is another kind of redundancy strategy in distributed storage, and it is both reliable and efficient. Regenerating code introduces the idea of network coding by allowing the helper node to transfer a linear combination of the data in it. Let M denotes the total size of original data file in term of symbols over a finite field of size q, each node stores α symbols. Let d be the repair degree of a failed node i, i.e. d nodes help repair node i. In the repair of node i, each helper node uploads β symbols. As a result, the total cost of repair bandwidth is $\gamma = d\beta$. The main idea of regenerating code is to divide the data in the storage node into multiple blocks. During the repair process, the blocks uploaded by every helper node are linear combinations of blocks stored in that node. The amount of uploaded blocks is less than the node capacity, i.e. $\beta < \alpha$. Dimakis et al. consider the repair process as the information flow graph of a multicast network. By analyzing the min-cut of the graph, Dimakis et al. derive the tradeoff between the storage capacity α and repair bandwidth γ according to cut-set bound of network coding. The two important class of regenerating codes are Minimum Storage Regenerating (MSR) code and Minimum Bandwidth Regenerating (MBR) code.

In MSR code:

$$(\alpha_{MSR}, \gamma_{MSR}) = (\frac{M}{k}, \frac{M}{k}\frac{d}{d-k+1}) \tag{2}$$

and in MBR code:

$$(\alpha_{MBR}, \gamma_{MBR}) = (\frac{M}{k}\frac{2d}{k(2d-k+1)}, \frac{M}{k}\frac{2d}{k(2d-k+1)}) \tag{3}$$

Recently, the research works most focus on the design of the MSR code and MBR code which can achieve exact regeneration.

3 Executing

MapReduce is a programming model for processing large data sets with a parallel, distributed algorithm on a cluster. A MapReduce program is composed of a Map() procedure that performs filtering and sorting (such as sorting students by first name into queues, one queue for each name) and a Reduce() procedure that performs a summary operation (such as counting the number of students in each queue, yielding name frequencies). The MapReduce System orchestrates by marshaling the distributed servers, running the various tasks in parallel, managing all communications and data transfers between the various parts of the system, and providing for redundancy and fault tolerance.

The model is inspired by the map and reduce functions commonly used in functional programming, although their purpose in the MapReduce framework is not the same as in their original forms. Additionally, the key contributions of the MapReduce framework are not the actual map and reduce functions, but the scalability and fault-tolerance achieved fro a variety of applications by optimizing

the execution engine once. MapReduce libraries have been written in many programming languages, with different levels of optimization. A popular open-source implementation is Apache Hadoop.

Map step : The master node takes the input, divides it into smaller sub-problems, and distributes them to worker nodes. A worker node may do this again in turn, leading to a multi-level tree structure. The worker node processes the smaller problem, and passes the answer back to its master node.

Reduce step: The master node then collects the answers to all the sub-problems and combines them in some way to form the output the answer to the problem it was originally trying to solve.

MapReduce allows for distributed processing of the map and reduction operations. Provided that each mapping operation is independent of the others, all maps can be performed in parallel though in practice this is limited by the number of independent data sources and/or the number of CPUs near each source. Similarly, a set of 'reducers' can perform the reduction phase, provided that all outputs of the map operation that share the same key are presented to the same reducer at the same time, or that the reduction function is associative. While this process can often appear inefficient compared to algorithms that are more sequential, MapReduce can be applied to significantly larger datasets than "commodity" servers can handle a large server farm can use MapReduce to sort a petabyte of data in only a few hours.[citation needed] The parallelism also offers some possibility of recovering from partial failure of servers or storage during the operation: if one mapper or reducer fails, the work can be rescheduled assuming the input data is still available.

Lets look at a simple example. Assume we have five files, and each file contains two columns that represent a city and the corresponding temperature recorded in that city for the various measurement days. In this example, city is the key and temperature is the value.

```
Beijing, 20
Shenzhen, 25
Shanghai, 22
Tianjin, 32
Beijing, 4
Tianjin, 33
Shanghai, 18
```

Out of all the data we have collected, we want to find the maximum temperature for each city across all of the data files. Using the MapReduce framework, we can break this down into five map tasks, where each mapper works on one of the five files and the mapper task goes through the data and returns the maximum temperature for each city. For example, the results produced from one mapper task for the data above would look like this:

```
(Beijing, 20) (Shenzhen, 25) (Shanghai, 22) (Tianjin, 33)
```

Lets assume the other four mapper tasks (working on the other four files not shown here) produced the following intermediate results:

```
(Beijing, 18) (Shenzhen, 27) (Shanghai, 32)
(Tianjin, 37)(Beijing, 32) (Shenzhen, 20)
(Shanghai, 33) (Tianjin, 38)(Beijing, 22)
(Shenzhen, 19) (Shanghai, 20) (Tianjin, 31)
(Beijing, 31) (Shenzhen, 22) (Shanghai, 19) (Tianjin, 30)
```

All five of these output streams would be fed into the reduce tasks, which combine the input results and output a single value for each city, producing a final result set as follows:

```
(Beijing, 32) (Shenzhen, 27) (Shanghai, 33) (Tianjin, 38)
```

As an analogy, we can think of map and reduce tasks as the way a census was conducted in Roman times, where the census bureau would dispatch its people to each city in the empire. Each census taker in each city would be tasked to count the number of people in that city and then return their results to the capital city. There, the results from each city would be reduced to a single count (sum of all cities) to determine the overall population of the empire. This mapping of people to cities, in

parallel, and then combining the results is much more efficient than sending a single person to count every person in the empire in a serial fashion.

4 Knowledge Acquisition

Under the Big Data trend, tabular data on the Web has become a rich source of structured data that is useful for ordinary users to explore. According to a recent Google study [1], there are a total of 14 billion raw HTML tables and, among those, 150 million relational data tables, easily making Web tables one of the richest structured data sources. There are many real-world applications that can benefit from high quality Web tables. First, search engines can utilize Web tables to improve the result quality. Existing search engines usually return relevant Web pages for queries and require users to mine answers from the returned Web pages. Obviously if there are many relevant high-quality Web tables, search engines can utilize these high-quality structured data to directly compute the answers and do not require users to search answers from documents. Query-answer (QA) systems can also utilize Web tables to easily generate answers for queries. Second, many systems want to provide Internet users with OLAP functionality, which is not supported in traditional keyword-based search systems. For example, if a user wants to know the average price of iPhone 5 in Beijing. Search engines cannot answer such queries. If we can extract Web tables about prices of iPhone 5 in Beijing, we can utilize these Web tables to answer such queries. Third, knowledge bases can use Web tables to enrich themselves by adding more concepts, entities and relationships between columns.

Due to the potential, tables on the Web have recently attracted a number of studies with the goals of understanding the semantics of those Web tables and providing effective search and exploration mechanisms over them [1,2,3,4]. Table understanding is to identify, recognize and interpret tabular structures to enable a variety of tasks such as data extraction, data interpretation, data integration, and search and analysis.

There are many research challenges in Web table understanding. The first challenge is how to identify large-scale high-quality Web tables. On one hand, there are billions of Web pages and only some of them contain tabular data. On the other hand, the Web data is rather dirty (due to typing errors or different representations for the same entity). Thus it is non-trivial to identify high-quality Tables from large numbers of Web pages. The second one is how to recognize Web tables. There is usually no description information for each Web table. The column names of Web tables are also usually incomplete or inaccurate because the same concept/entity may have different representations. Thus it is rather hard to determine the subject of a Web table (the main content the Web table describes), the concept of each column, and the entity of each cell value in the table. The third challenge is how to integrate multiple Web tables from different sources. Since Web tables from different sources may be relevant, we want to integrate them to generate more accurate and complete structured data. The fourth challenge is how to use Web tables to effectively support search and analysis applications. Internet users are only familiar with keyword queries and do not understand structured query languages. We need to bring the unstructured queries and the underlying structured data much closer so as to help users explore Web tables.

To address these challenges, several methods have been proposed, such as machine learning based method, traditional information retrieval method and human-machine hybrid method for understanding tables on the Web. These methods combine knowledge bases, crowds, and algorithms together and develop effective techniques to improve the quality of web table understanding. In the following parts, we focus on three main approaches mentioned above to explain how to understand web table problems respectively.

One main approach proposed new machine learning (ML) techniques to annotate table cells with entities that they likely mention, table columns with types from which entities are drawn for cells in the column, and relations that pairs of table columns seek to express. ML-based approach introduce a new graphical model for making all these labeling decisions for each table simultaneously, rather than make separate local decisions for entities, types and relations. This kind of ML based method could obtain high precision in processing Web table extraction and Web table interpretation, when utilizing formal knowledge base, such as YAGO catalog, Free Base, DBPedia, to table annotation table cells and columns. From the typical experimental result of ML based method, it showed a boost of F1 accuracy from 65% using baseline IR method to 70% using this approach. Intuitively,

this approach achieves better quality, but is in general not very scalable due to the expensive learning model being employed.

Another classic approach utilized traditional information retrieval (IR) technology, which enabled us to produce column concepts that are as good as those produced by ML based method, but in a much more scalable fashion. More specifically, IR based method identifies top k similar concepts from the knowledge base, where the similarity between a concept and a column is quantified based on the matching between the entities in the concept and the cell values of the column. Fuzzy matching is enabled between the cell values and knowledge entities, such that George Bush and George W. Bush can be matched. A more general set of similarity functions can be adopted to allow more flexible ranking of the candidate concepts than simple majority-rule. This method aimed to provide a solution that allows both fuzzy matching and ranking of the concepts based on more general similarity functions, without sacrificing scalability.

New approach has been suggested by several scholars called human-machine hybrid method which utilized combining technology between Crowdsourcing and machine. Crowdsourcing recently attracted significant attention from both industrial and academic communities. There are about millions of active users and we can conduct real experiments on this platform. But due to the high-money-cost and large-latency limitations in crowdsourcing, it is rather expensive and time consuming to use crowds to annotate and integrate Web tables. It calls for new methods to support large numbers of Web tables. So they combine machine learning algorithms, knowledge bases, and crowds to integrate multiple Web tables. First, the crowds only annotate some Web tables that are hard to correctly annotate for algorithms. Then they use machine leaning based algorithms to use crowds answers to annotate other Web tables so as to reduce the number of questions submitted to crowds. Iteratively, this method can integrate large numbers of Web tables with less money cost and high quality.

5 Next generation model

Besides the huge quantity of data, it is also a serious problem to deal with the rapid and high dimensional data. In such a background, the data streaming model was proposed. It is a widely used model not only on IT but also in astronomic, communicating and economic fields. The first appearance of this model could be traced back to 20th century, but it is one of the hottest topic in recent years since the scale of the data becomes the bottleneck of modern IT development. Based on such model, the data streaming algorithms have the characteristics of high timeliness, low complexity and approximation.

The approximation is the core of the data streaming algorithm. The real data in internet is actually occupied by the large quantity of redundance. In this section we introduce several classical data streaming algorithms to give a sketch how this model fit the modern data processing.

5.1 Conception

In data streaming model, the source signal could be regards as a high dimension vector $\vec{A}$. Initially, $\vec{A}$ equals $\vec{0}$. The data is a continuous tuple sequence $< a_1, ..., a_n >$ denoting a updating of the source signal. According to the type of tuple, it can be classified by:

1. Time Series Model : a_i is an integer indicating update A_i by $a_i = A_i$.
2. Cashier Register Model : a_i is a two-wise tuple $(j, D_i), D_i \geq 0$, which indicating an updating $A_i[j]$ by $A_{i-1}[j] + D_i$
3. Turnstile Model : a_i is a two-wise tuple $(j, D_i), D_i \in R$, which indicating an updating $A_i[j]$ by $A_{i-1}[j] + D_i$

This paper only concern about the cashier model. Consider an IP tracker, with each visiting counted once by $(IP, 1)$, thus it is a representational application of cashier model. Moreover, assume all the data is pass-efficient.

5.2 Methods

The typical methods in streaming model consist of *Sampling* and *Sketching*. *Sampling* method will draw the data with a specific probability and the distribution of data will be reconstruct by the samples. While the *Sketching* method mapping the arbitrary data with a hash function to a length fixed space in order to decrease the space complexity. Moreover the sketching method is simple to build but well performed. Especially, the error of approximation could be bounded with a small factor of L_p norm.

Human beings could be regarded as a "biological" sketching machine. The sensors of human receive a consequently high dimensional signal(such as audio and video signal) from the nature rapidly and process it with sketching method. Therefore people usually could not remember every thing happened before, but they might have a vague idea about that. Differ from sampling, the sketching method is sensitive to abnormal data which implies the sketching could be used into anomaly detection.

5.2.1 Flajolet-Martin

Flajolet-Martin is used to estimate distinct elements of data. Suppose the sequence of streaming equals $\delta =< a_1, ..., a_n >$, for each a_i we have $a_i = (j \in [n], D_i = 1)$. Therefore it is a streaming description of frequency vector $f = (f_1, .., f_n)$ and the result algorithm's output should be $d = |\{j : f_j > 0\}|$.

Since the dimension of source signal could be unsolvable large, then the algorithm is supposed to output an approximation with small error bounded. Formally, suppose the output is $\hat{d}$, for any $\epsilon, \delta > 0$, $Pr[|\hat{d} - d| > \epsilon]$ should be less equal than δ.

Let $zeor(x)$ be the quantity of 0s in the tail of binary representation. Here we provide the algorithm:

Flajolet-Martin Algorithm

1. Initialization
   Choose a hash function $h : [n] \to [n]$ from 2-wise independent hash function family.
   Initialize $z = 0$
2. Processing Data
   Update $z = \max\{z, zeros(h(j))\}$
3. Output
   $2^{z+\frac{1}{2}}$

Although Flajolet-Martin Algorithm is a simple algorithm but the reason it works is not trivial. To analyze the algorithm, suppose $z = t$ when the algorithm terminate. Let $Y_r = \sum_{j:f_i>0} X_{r,j}$, then $Y_r \geq 0 \Leftrightarrow t \geq r$. Equivalently, $Y_r = 0 \Leftrightarrow t \leq r - 1$. Since h maps an arbitrary number in $[n]$ to a binary string with length $log(n)$. Therefore:

$$E[X_{r,j}] = Pr[zeros(h(j)) \geq r] = Pr[2^r divides h(j)] = \frac{1}{2^r}$$

$$E[Y_r] = \sum_{j:f_i>0} E[X_{r,j}] = \frac{d}{2^r}$$

Also notice h is pair-wise independent, we have

$$Var[Y_r] = \sum_{j:f_j>0} Var[X_{r,j}] \leq \sum_{j:f_j>0} E[X_{r,j}^2] = \sum_{j:f_j>0} E[X_{r,j}] = \frac{d}{2^r}$$

Substitute the equation to *Markov Inequality* and *Chebyshev Inequality*, it implies:

$$Pr[Y_r > 0] = Pr[Y_r \geq 1] \leq \frac{E[Y_r]}{1} = \frac{d}{2^r}$$

and

$$Pr[Y_r = 0] = Pr[|Y_r - E[Y_r]| \geq \frac{d}{2^r}] \leq \frac{2^r}{d}$$

To give a reasonable error bound of $\hat{d} = 2^{t+\frac{1}{2}}$, let a be the smallest integer satisfy $2^{t+\frac{1}{2}} \geq 3d$, b be the largest integer satisfy $2^{t+\frac{1}{2}} \leq \frac{d}{3}$. Thus it is guaranteed with probability at least 53% $\hat{d}$ is in interval $[\frac{d}{3}, 3d]$.

Since the algorithm is supposed to bound the error with arbitrary ϵ. A statistical trick called *Median Trick* will be used here. The intuition of Median Trick is to maintain k procedures parallelly and promise the distribution of k estimator is approximate Gaussian distribution. Therefore we could pick $\hat{d}_{k/2}$ as the final estimator instead. More precisely, we need introduce a crucial bound in probability theory, the *Chernoff Bound*.

Chernoff Bound: $X_1, X_2, ..., X_n$ are random variables follows distribution with parameters p. Even S denote more than half of X equals 1. Then $Pr(S) \geq 1 - exp\{-\frac{1}{2p}n(p-\frac{1}{2})^2\}$.

Let X_i denote $\hat{d}_i \geq 3d$, substitute $n = k, p \leq \frac{\sqrt{2}}{3}$ we have $Pr(S) = 2^{-\Omega(k)} = \delta$. Maintain $k = \Theta(log(\frac{1}{\delta}))$ procedures parallelly, and output $\hat{d}_{1/2}$, the algorithm gives a $(O(1), \delta)$ approximation with space complexity $O(log(\frac{1}{\delta})log(n))$.

5.2.2 Count Sketch

Count Sketch is a crucial streaming data structure. It supports point, range, inner product and even quantile queries.

Restricted in cashier model, the algorithm maintain a t-rows k-columns table. Each row contains a vector in k dimension indicating the projection space of the hash function. Utilize the median trick, the algorithm is provided:

Count Sketch Algorithm For Point Query

1. Initialization

   build a table C with t rows and k columns, $t = \lceil log(\frac{1}{\delta}) \rceil$, $k = \frac{3}{\epsilon^2}$. Set all the elements as zero.

   Select t independent hash functions $h : [n] \to [k]$ from 2-universal randomly

   Select t independent hash functions $g : [n] \to \{-1, +1\}$ from 2-universal randomly

2. Processing Data

   $for\ i = 1 \to\ t$ **do** $C[i][h_i(j)]\ \leftarrow\ C[i][h_i(j)] + cg_i(j)$

3. Output

   $\hat{f}_a = median_{1 \leq i \leq t}\ g_i(a)C[i][h_i(a)]$

To analyze the algorithm, consider a specific row of the table C_t. Let $X_t = \hat{f}_a$ be the output of input a, the indicator variable Y_j indicate $h(j) = h(a)$. Since $C_t[h(a)]$ is updated only when $h_t(j) = h_t(a)$, the accumulated contribution is $f_j \cdot g_t(j)$, Therefore

$$X = g_t(a) \sum_{j=1}^{n} f_j g_t(j) Y_j = f_a + \sum_{j \in [n] \setminus \{a\}} f_j g_t(a) Y_j$$

According to the properties of expectation, we have

$$E[g_t(j)Y_j] = E[g_t(j)]E[Y_j] = 0$$

which implies

$$E[X_t] = f_a + \sum_{j \in [n] \setminus \{a\}} f_j g_t(a) Y_j = f_a + 0 = f_a$$

As it shows the output is an unbiased estimator of f_a. Since $g(a)^2 = 1$, it implies $Var[x] = \sum_{j \in [n] \setminus \{a\}} \frac{f_j^2}{k} = \frac{\|\mathbf{f}\|_2^2 - f_a^2}{k}$. Take it into Chernoff bound and let $t = O(log(\frac{1}{\delta}))$, then for any given $\epsilon, \delta > 0$, count sketch promise $Pr[|\hat{f}_a - f_a| \geq \epsilon \parallel \mathbf{f}_{-a} \parallel_2] \leq \delta$.

References

[1] Cormode G, Muthukrishnan S. An improved data stream summary: the count-min sketch and its applications[J]. Journal of Algorithms, 2005, 55(1): 58-75.

[2] Cormode G, Garofalakis M. Sketching streams through the net: Distributed approximate query tracking[C]//Proceedings of the 31st international conference on Very large data bases. VLDB Endowment, 2005: 13-24.

[3] Chiky R, Hbrail G. Summarizing distributed data streams for storage in data warehouses[M]//Data Warehousing and Knowledge Discovery. Springer Berlin Heidelberg, 2008: 65-74.

[4] Papapetrou O, Garofalakis M, Deligiannakis A. Sketch-based querying of distributed sliding-window data streams[J]. Proceedings of the VLDB Endowment, 2012, 5(10): 992-1003.