2012

CloudVista: a Framework for Interactive Visual Cluster Exploration of Big Data in the Cloud

Zhen Li
Wright State University

CloudVista: a Framework for Interactive Visual Cluster Exploration of Big Data in the Cloud

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Engineering

by

Zhen Li
B.E.,Dalian Jiaotong University, 2008

2012
Wright State University

Wright State University
SCHOOL OF GRADUATE STUDIES

September 21, 2012

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Zhen Li ENTITLED CloudVista: a Framework for Interactive Visual Cluster Exploration of Big Data in the Cloud BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science in Computer Engineering.

Keke Chen, Ph.D
Thesis Director

Mateen Rizki, Ph.D
Chair, Department of Computer Science and Engineering

Committee on
Final Examination

Keke Chen, Ph.D

Guozhu Dong, Ph.D

Thomas Wischgoll, Ph.D

Andrew Hsu, Ph.D
Dean, School of Graduate Studies

ABSTRACT

Li, Zhen. M.S.C.E, Department of Computer Science and Engineering, Wright State University, 2012. *CloudVista: a Framework for Interactive Visual Cluster Exploration of Big Data in the Cloud.*

With the development and deployment of ubiquitous information sensing, mobile devices, wireless sensor networks, RFID readers, simulation, and computer generated software logs, big data have become precious resources for scientific study, business intelligence, and national security. As one of the most intuitive and effective analysis methods, visual cluster analysis remains as a significant challenge for big datasets. First, existing visualization models need to be updated to process big data in parallel. Second, processing big data inevitably bring large latency, which conflicts the requirement of interactivity. In this thesis, we develop the CloudVista framework to address the common problems with data reduction methods and the conflict between the latency caused by processing big data and the interactivity desired by visual cluster exploration. There are a number of components in the framework: (1) the data structure *visual frame* and the previously developed VISTA visualization model for parallel processing; (2)the *RandGen* algorithm that generates batches of meaningful visual frames; and (3) a workflow to minimize the cost of big data processing. The CloudVista demonstration system is designed and implemented with web services and Hadoop/MapReduce, assuming the entire big data stored in the cloud. Finally, we show some visualization results and performance evaluation results based on the demonstration system.

Contents

List of Figures

List of Tables

Acknowledgement

Foremost, my sincere gratitude goes to my advisor Prof. Keke Chen for his kindness, motivation, expertise, and most of all, for his patience. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Master study.

In addition, I would like to thank the rest of my thesis committee members: Prof. GuoZhu Dong and Prof. Thomas Wischgoll, for their insightful comments, and hard questions.

Finally, I would like to thank my lab mates: Shumin Guo and Huiqi Xu for helping me develop the demonstration system.

Dedicated to

my dear parents, Shenghe Li and Shujing Chen;

without whom it would never have been accomplished

Introduction

1.1 Overview

Data Intensive Computing in the Cloud. With the development of information technology, a large amount of data are generated by sensor networks,social networks, web applications, astronomy, military, medical records and so on. Every day, 2.5 quintillion bytes of data are created and 90% of the data in the world today was created within the past two years[1]. "big data" now refers to terabytes, petabytes, or exabytes, and the definition keeps evolving. With the increase of data, making sense of data also becomes increasingly difficult. Data clustering is an important tool for exploring and understanding data, which groups similar objects and separate dissimilar ones. In particular, interactive visual cluster analysis invites users into the clustering process and helps user quickly understand big data, filer big data and capture unique patterns.

1.2 Challenges of Clustering Extreme Scale Data

A clustering algorithm depends on the similarity measure to define groups and separate groups. Clustering and cluster analysis is a highly complicated, time consuming process. Users have numerous choices of similarity measures and clustering algorithms. They often need to examine a number of candidate similarity measures and algorithms to see whether the resultant patterns are useful.

[1] http://www-01.ibm.com/software/data/bigdata/

Traditionally, when a large data is involved, the three-phase framework is used to reduce the cost of iterative cluster analysis. Figure 1.1 shows the three-phase framework. The sampling/summarization step is used to reduce the size of the dataset. The clustering and cluster analysis step often happens in one powerful local visualization workstation, and this step is highly iterative, involving parameter tuning, re-clustering, and cluster understanding, often conducted by domain experts. After the final clustering structure is determined, the intermediate clustering structure is used to label the cluster membership for all records in the entire large dataset, which is typically a linear-complexity process.

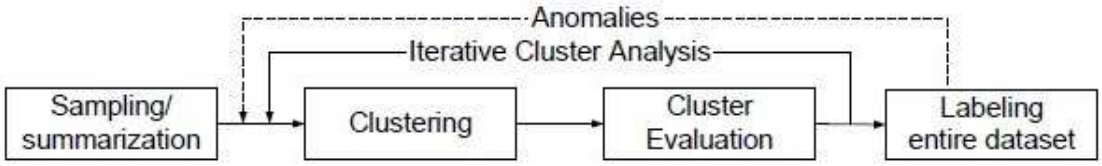

Figure 1.1: Three phases for cluster analysis of large datasets

An extremely low sample rate will cause serious mismatch between the clustering structure in the sample or summary dataset and that in the entire dataset. To make the three-phase framework meaningful, the sampling rate cannot be too low. As described by previous study, the appropriate sample rate is also determined by the complexity of the clustering structure [22]. Since the workstation's processing power in the second phase is limited, the sampling size keeps about the same. As a result, with the increase of the whole dataset, sampling rate is decreasing. The low sample rate causes severe problems, such as abnormal visual cluster patterns, missing small clusters, unclear secondary clustering structures, and cluster boundary extension [7].

Challenges with Interactive Visual Cluster Exploration Many believe that interactive visual analytics is one of the best approaches to addressing the big data challenges. In particular, interactive visual exploratory data analysis [44, 45] can help users quickly understand the big data, filter data, and capture unique patterns [20]. However, working big data also brings several unique challenges. (1) It requires visualization algorithms scalable

to big data and large-scale parallel processing infrastructures. (2) As big datasets are often stored and processed remotely in the cloud, the latency caused by network and batch processing severely conflicts with the requirement of interactivity. (3) Exploring data in the cloud also needs to address the unique economics problem - how to minimize the financial cost while ensuring the quality of service not compromised.

Scope and Contributions The proposed framework for interactive cluster visualization framework is called **CloudVista**, which is used to address the problems with exploring big data in the cloud. In particular, the limitation brought by the sampling-based approaches is addressed by the whole-data exploration approach, which, however, raises the conflict between the processing latency and the interactivity of visual cluster exploration.

The CloudVista framework clearly divides the processing responsibilities between the data cloud and the visualization workstation. The large dataset is stored and processed in the cloud. Parallel processing algorithms are used to reduce the big data to a key structure "visual frame". An important property is that the size of visual frame does not depend on the big dataset, but on the resolution of visualization. Visual frames can be generated in batch in the cloud, and then sent to the workstation. Users work with the workstation locally to render visual frames and interactively explore them.

It is important to choose an appropriate visualization model. In the current version of framework, we use the VISTA visualization model [6] because it can be easily parallelized. Previous studies have shown that this model is effective in validating clustering structures, incorporating domain knowledge in previous studies [6], and handling moderately large scale data with the three-phase framework [7].

We address the latency problem with an automatic batch frame generation algorithm - the RandGen algorithm. The goal is to efficiently generate a series of meaningful visual frames without the user's intervention. With the initial parameter setting determined by the user, the RandGen algorithm will automatically generate the parameters for the subsequent visual frames, so that these frames are also continuously and smoothly changed. We show

that the statistical properties of this algorithm can help identify the clustering structure. In addition to this algorithm, we also support a hierarchical exploration model to further reduce the cost and need of cloud-side processing.

We also implement a demonstration system based on Hadoop/MapReduce [50]. Some preliminary experimental evaluation is done to show the advantages of the whole-data visualization approach and the performance of the batch frame processing.

Related Work

2.1 Interactive multidimensional data visualization

Since data exploration has to involve the user in the analysis loop, most data exploration tools use interactive visual user interface [4]. Many visual data exploration techniques are specifically designed for exploring multidimensional data, such as Grand Tour [1, 52], Projection Pursuit [10], Star Coordinates [32], RadViz [27], HD-Eye [26], and VISTA [6]. These tools are often used to discover patterns, validating clustering structures, and detect anomalies. A few tools are specifically designed to visualize algorithmic clustering results, such as IHD [51] and Hierarchical Clustering Explorer [44].

The scalability of visual design is greatly limited by the dimensions of data. Parallel Coordinates system [29, 30] and Scatterplot matrix are probably fine for exploring less than ten dimensions. Star coordinates systems can scale up to tens of dimensions [6, 7]. We have shown that the VISTA system can comfortably handle 68 dimensions (for the Census data in UCI KDD database) with a normal PC display [7]. With higher dimensions, dimensionality reduction techniques such as linear ones [31, 16, 19, 28, 10, 48] and non-linear ones [25, 46, 37, 43, 38, 42, 2, 14] have to be applied to reduce the dimensionality to a manageable number. Note that the multidimensional scaling algorithms [11, 15, 12, 49] we discussed in the proposal are also used to reduce dimensionality.

For large datasets, pixel-based [35, 36] and density-based [34, 8, 26] visualization techniques are more appropriate, because of the heavy overlapping of visual objects. In practice, the three-phase framework "sampling [3, 23] or summarization [53] – cluster-

ing/cluster analysis – disk labeling [23, 5]" enables the application of high complexity algorithms or visual exploration on smaller representative datasets. Compression [17, 40] and multi-resolution [24, 9] techniques are also used to work with the data reduction techniques. However, data reduction techniques will result in problems such as missing small clusters, rare events, and distorting the clusters [7].

2.2 Big Data Analysis with Hadoop/MapReduce

A number of MapReduce-based data mining algorithms have been proposed or developed recently, aiming at big data in the cloud. The use of Hadoop/MapReduce [13] enables reliable and efficient processing of big data. However, it is challenging to redesign the algorithm logic to fit in the MapReduce processing framework. PLANET [41] was developed for parallel tree ensemble learning on big data. PEGASUS [33] is designed for mining peta-scale graphs, and Lin, et al. [39] applies MapReduce programs in text mining. MapReduce is also applied to visualize scientific data (typically, low dimensional) [21]. However, to our knowledge, there is no work on visualizing multidimensional big data in the cloud, except for our recent development [8].

There are two modes to run a MapReduce job, in either a private Hadoop cluster or a Hadoop cluster in the public cloud. In a private Hadoop, users just submit a MapReduce job and wait to be scheduled. A private Hadoop normally has thousands of compute nodes, which is a big investment. In most companies, research institutes, and government agencies, there are no in-house large-scale private Hadoop clusters. Therefore, running Hadoop/MapReduce on top of the public cloud is a more realistic and in fact economical method for most users. Amazon has developed the Elastic MapReduce[1], which allows users to run on-demand Hadoop/MapReduce clusters using Amazon EC2 nodes. Scripts[2] also available for users to manually setup Hadoop clusters on EC2 nodes.

[1] aws.amazon.com/elasticmapreduce/.
[2] e.g., wiki.apache.org/hadoop/AmazonEC2

Running a Hadoop cluster on top of the public cloud is totally different from on a private Hadoop cluster. First, normally the user start a dedicated Hadoop cluster for each job, because it is convenient to do so with the cloud resources. As a result, there is no need to consider multi-user or multi-job resource competition problems. Second, the user is responsible for setting up the on-demand cluster. In particular, she/he needs to set the appropriate number of virtual nodes for the cluster. It is difficult because the optimal setting may differ from application to application, the type of virtual nodes, and the amount of input data. We have developed an resource provisioning and optimization method for running Hadoop jobs in public clouds [47].

Preliminaries

3.1 Hadoop and MapReduce

Our framework uses Hadoop/MapReduce to process big data in parallel and generate visualization. In this section, we briefly introduce these two components.

3.1.1 Hadoop

Hadoop is an open source software framework. It aims to build massive-scale parallel processing infrastructure on commodity computers based on Google's technique: Google file system (GFS) [18] and MapReduce [13]. Correspondingly, it has two components: the distributed file system (HDFS) and the MapReduce processing and programming model. The size of a Hadoop cluster can range from several servers to thousands of servers. A Hadoop cluster consists of a single master node and multiple worker nodes. The master node manages MapReduce jobs and the HDFS storage system. It includes a JobTracker process and a NameNode process. Each slave (worker) node receives commands from the master node to handle data. A slave node has a DataNode process and a TaskTracker process. Figure 3.1 shows the architecture of Hadoop cluster.

HDFS is a distributed, scalable, and portable file system running on commodity hardware. HDFS system divides every file into multiple fixed size blocks , and these blocks are saved through the DataNodes of a Hadoop cluster. The figure 3.2 shows the architecture of HDFS.

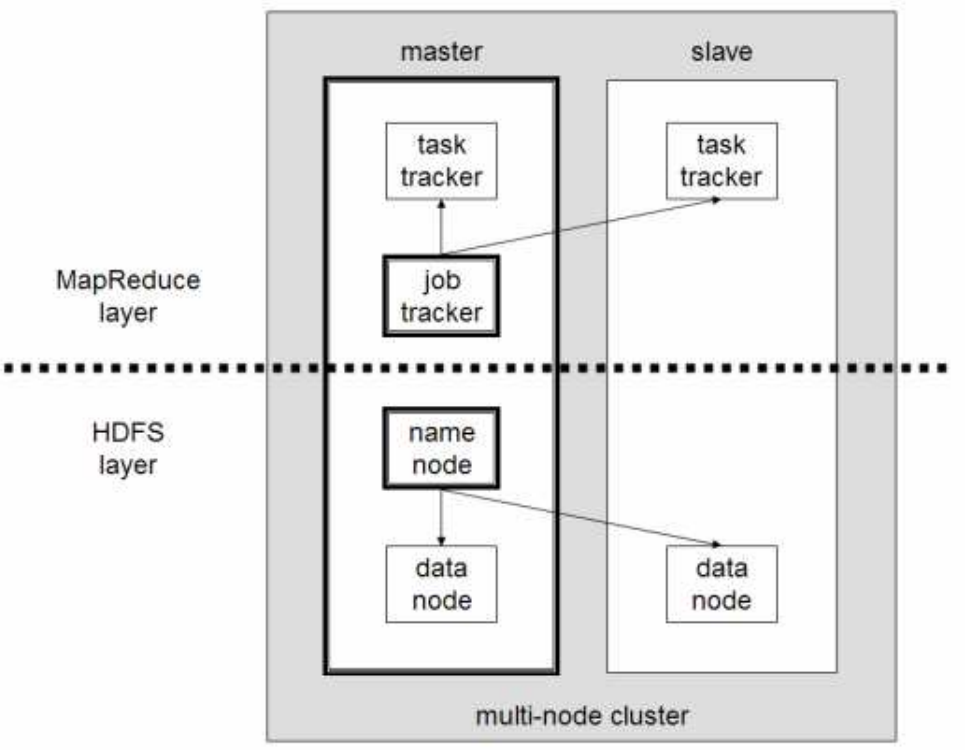

Figure 3.1: Architecture of Hadoop, cited from Hadoop's website (hadoop.apache.org).

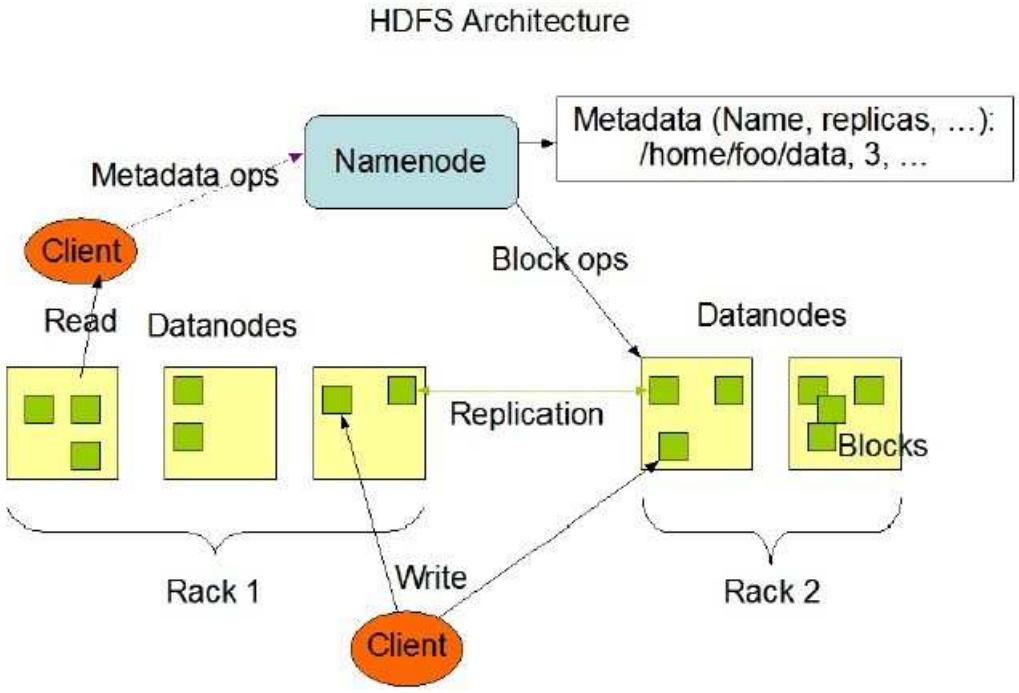

Figure 3.2: Architecture of HDFS, cited from Hadoop's website (hadoop.apache.org)

3.1.2 MapReduce

MapReduce is a programming model and also a processing framework. The users only need to implement the Map and Reduce functions for lager scale datasets. This figure 3.3 shows the whole process of MapReduce. First, the source data are partitioned into into a set of m splits. Each split will be handled by one Map task. The user designed map function will process the data and generate intermediate pairs <key, value>, and each reduce task will aggregate all the assigned intermediate pairs <key, value> associated with the same intermediate key.

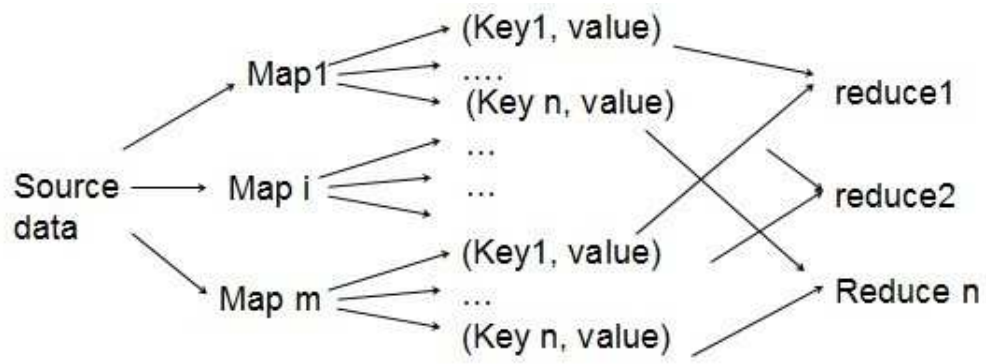

Figure 3.3: An illustration of MapReduce processing.

MapReduce programs are automatically parallelized by the infrastructure, distributed and executed on a large cluster of commodity servers (or virtual nodes in public clouds). The run-time system (e.g., Hadoop) takes care of partitioning the input data, scheduling the jobs and tasks execution across a set of nodes, managing the required inter-machine communication, and handling system failures. Programmers with no knowledge on distributed computing can easily program and utilize the resources of a large distributed system.

3.2 Amazon cloud

Amazon Elastic Compute Cloud (Amazon EC2) is a major part of Amazon Web Services.Amazon EC2 allows users to rent virtual computers preloaded with a variety of oper-

ating systems. User can create a virtual machine by using Amazon Machine Image, which is called an “virtual machine instance. A user can create, launch, and terminate instances according to their needs. EC2 employs a pay-as-you-go billing model, i.e., you pay only for the amount of time and the number of instances you use. Each instance is charged by hours. We will eventually deploy our system on top of Amazon EC2 to conveniently scale up for big data processing.

Amazon Simple Storage Service(Amazon S3) provides a simple web services interface that can be used to store and retrieve data, at any time, from anywhere on the web. A data object in Amazon S3 is stored in a *bucket* and named by a *key*. Every data object in the Amazon S3 can also be accessed by a URL. In our framework, data will be stored in S3 and accessed by dynamically created Hadoop clusters in the cloud.

Amazon Simple Queue Service(Amazon SQS) offers a reliable, highly scalable, hosted queue for storing messages as they travel between computers. By using Amazon SQS, users can coordinatethe distributed components of their applications. Amazon SQS makes it easy to build an automated workflow, working in close conjunction with other Amazon cloud services and components outside of Amazon. . We will use SQS as the communication channel between the client and the cloud.

The CloudVista Framework

4.1 CloudVista Framework and Components

In this section, we will describe the CloudVista framework and its components in detail.

4.1.1 The CloudVista Framework

The CloudVista approach has a three-tier architecture: the interactive visualization client, the application server, and the Hadoop cluster (the latter two are possibly in the cloud), as shown in Figure 4.1. It processes multidimensional numerical data for exploratory cluster analysis. The data and computing intensive tasks are conducted in the Hadoop cluster. The visualization is encoded in the intermediate visual representation - the visual frame, and generated by the MapReduce program, RandGen. According to the size of the interested subset of the data, the cloud side may return the user selected subset or a sample of the subset. The application server handles the communication between the client and the cloud. It delivers the output of the cloud, e.g, the visual frames and subset information to the client. It accepts cloud processing commands and invokes the corresponding MapReduce programs. It may also compress or encrypt data for transmission. The client side visualizes the visual frames, allows users to interactively explore the dataset. If the cloud returns a small dataset, e.g., a subset selected by the user, the local VISTA visualization system [6] is used to explore the data. The communication between the client and the application server is via some methods, either a web service or a message queue service.

There are some key technical components in this architecture, including the VISTA visualization model, the key data structure "visual frame", the cloud-side parallel data processing algorithms for generating visual frames or subsets, and the major workflow of user exploration. Figure 4.1 shows these components.

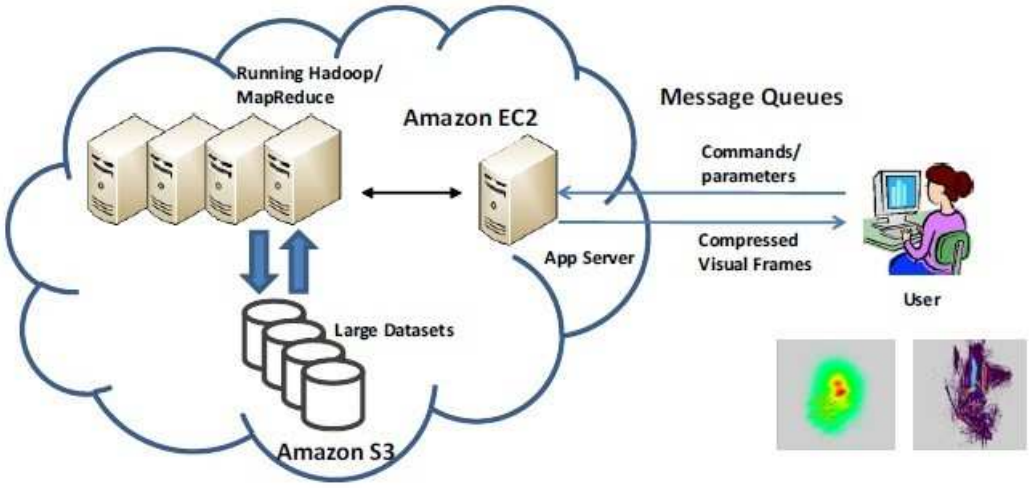

Figure 4.1: The CloudVista Framework [8].

4.1.2 Visualization Model

We use the VISTA visualization model [6] in our prototype. It is built on top of star coordinates 4.2. Let $\mathbf{s}_i \in \mathbb{R}^2, i = 1, \ldots, k$ be unit vectors arranged in a "star shape" around the origin on the display, and a k-dimensional point $\mathbf{x} = (x_1, \ldots x_i, \ldots, x_k)$, where x_i is appropriately normalized (often using standardization or max-min normalization). $\alpha = (\alpha_1, \ldots, \alpha_k)$, $\alpha_i \in [-1, 1]$ are the dimensional weights and $c \in \mathbb{R}^+$ (i.e., positive real) is a scaling factor. The mapping is defined as

$$f(\mathbf{x}, \alpha, \{\mathbf{s}_i\}, c) = c \sum_{i=1}^{k} \alpha_i x_i \mathbf{s}_i. \tag{4.1}$$

This model is essentially a random projection model if we randomly select the parameters α_i. We have shown that close points will be surely mapped to close 2D points, while distant points might also be mapped to close 2D points [8], which creates visual cluster overlapping

and needs the user to interactively explore and distinguish. By tuning α values, the user can interactively find the possible overlapping.

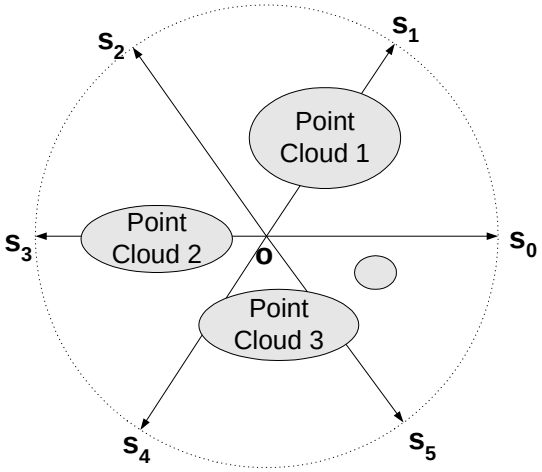

Figure 4.2: Illustration of star coordinates [6]

In summary, there are several important properties with this model. (1) The model is linear and thus very efficient to apply. (2) It preserves the dense areas, but may cause visually overlapped dense areas. (3) The α parameters can be adjusted to create an animation. (4) Since this model is applied to individual records, it is naturally parallel and can be implemented with MapReduce [13].

4.1.3 The Visual Frame Structure

Visual frame is the key structure in the CloudVista framework. It serves as the communication package between the client and the cloud. The visualization result is generated via parallel processing in the cloud side and encoded as visual frames. Compared to the original big data, it is space-efficient and not determined by the size of the original dataset.

The structure of visual frame is determined by the display space. It partitions the two-dimensional display space into a grid structure, where each cell encodes the density information, i.e., the number of points mapped to the cell, as shown in Figure 4.3. Thus, the size of a visual frame is bounded by the number of non-empty cells - we call it resolution. The setting of resolution is a tradeoff between the cost and the details of visualization.

Normally, a rectangle display area for a normal PC display contains about one thousand by one thousand pixels. We can set high resolution to 1000 by 1000, for instance. With this setting, a few megabytes will be sufficient to represent a visual frame. In contrast, a big dataset is magnitudes larger than a visual frame. There is a natural data reduction process, when we map the large dataset to a visual frame, and the well-known MapReduce programming and processing model can nicely fit in.

In the following, we briefly discuss how a visual frame is generated with the MapReduce model. The Map process accepts the parameters such as the parameters for the VISTA visualization model, and the resolution of the generated visual frame. With the resolution, e.g., 1000 by 1000, the visual space is partition and aligned with the cell coordinates. Because the VISTA model applies to each record, it can be implemented as a naturally parallel process. The Map process simply applies the VISTA model and map the record to a tuple $\langle(fid, u_1, u_2), 1\rangle$, where fid is the frame ID, (u_1, u_2) is the coordinate of the cell, and 1 means the contribution to the cell. In the Reduce process, the tuples are aggregated by (fid, u_1, u_2) to generate $\langle(fid, u_1, u_2), d\rangle$, where d is the density of the cell for the frame fid. The cells are filled sparsely, which means the actual size of a visual frame might be even smaller than megabytes. If low resolutions are used, the size of visual frame can be reduced further.

Such a visual frame can be easily visualized with the heatmap method, where higher-density cells have warmer colors. The following MapReduce code snippet sketches the MapReduce program for generating a frame, where the function f is the VISTA model. This program can be extended to handle multiple frames, for example, the RandGen algorithm that we will discuss later.

4.1.4 Batch Frame Generation Algorithm: RandGen.

In local exploration of small datasets, the user can tune the α values to immediately observe the changing visualization to find the clue of possible visual overlapping. However,

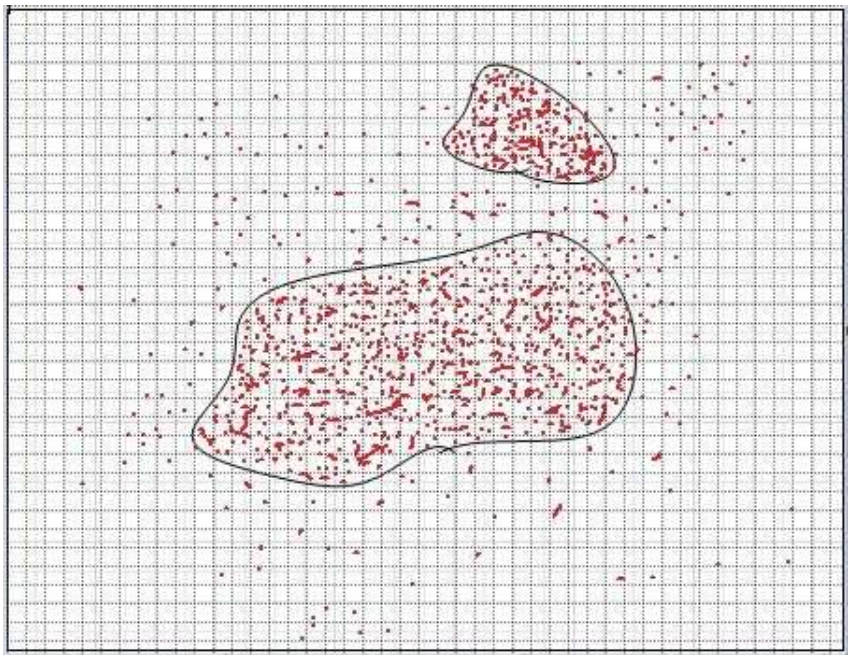

Figure 4.3: The Structure of Visual Frame

1: **map**$(fid, i, \mathbf{x}, p, r)$
2: fid: frame id, i: record id, $\mathbf{x}$: k-d record,
p: mapping parameters including α, θ and c,
r: resolution.
3: $(u_1, u_2) \leftarrow f(\mathbf{x}, \alpha, \theta, c)$;
4: EmitIntermediate((fid, u_1, u_2), 1)

1: **reduce**$((fid, u_1, u_2), v)$
2: fid: frame id, (u_1, u_2): coordinate, v: list of counts.
3: $d \leftarrow 0$;
4: **for** each v_i in v **do**
5: $\quad d \leftarrow d + v_i$;
6: **end for**
7: Emit$(\langle (fid, u_1, u_2), d \rangle)$;

---

this is impractical for exploring big data in the cloud due to the latency of updating the visualization. We develop the RandGen algorithm to generate the continuously changing visualization in batches of visual frames. The basic idea is to allow the user to quickly capture the global patterns from randomly generated visual frames. Then, they can drill down to the subsets for detailed exploration, which can be done locally.

To generate a meaningful set of frames in batch, we use the RandGen algorithm. This algorithm applies a randomization process to generate a set of related α values. The algorithm starts from the initial setting of α values, which can be determined by default values, e.g., setting all $alpha$ to 0.5, or by using a random selection to boost the RandGen algo-

rithm. The RandGen algorithm then applies the following perturbation to all dimensional weights simultaneously to generate the next set of α values, which are certainly bounded by the range [-1, 1]. Let α_i^ϕ represent the α parameter for dimension i in frame ϕ, the update can be defined as follows, where $\alpha_i^{\phi+1}$ is the α_i setting for the next frame.

$$\alpha_i^{\phi+1} = \begin{cases} 1 & \text{if } \alpha_i^\phi + \delta_i > 1 \\ \alpha_i^\phi + t \times B & \text{if } \alpha_i^\phi + \delta_i \in [-1, 1] \\ -1 & \text{if } \alpha_i^\phi + \delta_i < -1, \end{cases} \tag{4.2}$$

where t is a predefined step length, set to a small value, e.g., $0.01 \sim 0.05$, and B is a uniform random variable return -1 or +1. δ_i is generated independently at random for each dimension. This process repeats until it reaches the user defined number of steps. Because the adjustment is very small, the resultant visual frames change smoothly. The user can observe an animation by playing the visual frames.

We have proved that with a sufficiently large number of continuously changing frames (e.g., 100 frames), the clustering structure can be statistically preserved and visually observed by the user [8].

4.1.5 Subset exploration

We use two methods to guarantee efficient exploration of big data. One is the batch-interaction model, supported by the RandGen algorithm. The other is the "details-on-demand" exploration [4], which can exponentially reduce the size of data to be visualized. In the framework, we allow the user to select a subset of data points for exploration, based on a certain visual frame. When a subset is selected, multiple strategies can be applied.

Concretely, the following algorithm is used to to support the details-on-demand exploration. Figure 4.4 shows the flowchart how this model is supported in the framework. Depending on the size of the selected subset of data, denoted by ν records, the cloud side

can have three processing strategies. If the selected data is very small, e.g., smaller than the client's visualization capacity, denoted as μ records, the subset is simply returned directly and the local exploration can be applied using the existing VISTA system (Case 1). If the rate $\mu/\nu > \xi$, where ξ is the lowest sampling rate acceptable by the user, often 1%-5%, the selected subset is sampled to get μ records (Case 2). Similarly, the sample set can be explored with the VISTA system locally. If the rate $\mu/\nu < \xi$ which means sampling is not applicable, the RandGen algorithm is applied to generate a batch of frames for that subset only (Case 3). We have estimated how many cloud-side operations are needed if this exploration model is supported [8].

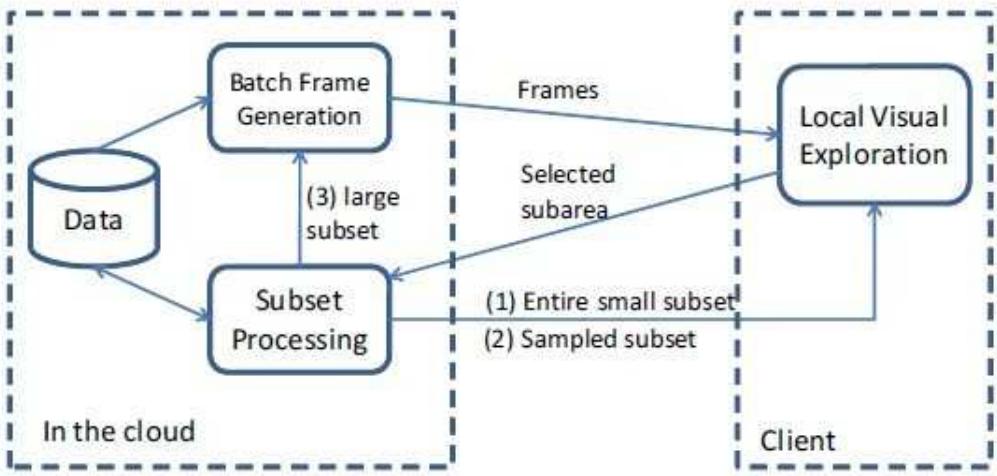

Figure 4.4: Interactions between the client and the cloud

A MapReduce program is developed for the key operation, subset selection and sampling. It simply filters out the selected records based on the user selected area (a rectangle area on the selected frame). The sampling and the RandomGen algorithm can also be appropriately integrated into this step.

4.2 Extended Study of the CloudVista Framework

This section includes the ongoing work for the CloudVista framework, which is extended for other visualization models.

4.2.1 Other Visualization Models

The CloudVista framework also works for other visualization models. In the following, we briefly describe how to apply Parallel Coordinates [29, 30] and Scatterplot Matrix in the framework.

Parallel Coordinates. The parallel coordinates model organizes the multidimensional coordinates in a parallel way (Figure 4.5). Thus, each point is represented as a segmented line linking the dimensional coordinates. Though it is a simple representation, parallel coordinates can be used to represent geometric objects such as points, lines, planes, or surfaces in multidimensional spaces [30]. In MapReduce formulation, the Map takes in a list of multidimensional points and the coordinate locations. It maps each pair of neighboring coordinate values of each point to a line segment. Assume (y_i, y_{i+1}) represents the two neighboring dimensions X_i and X_{i+1} for a k-dimensional point $(y_1, y_2, \ldots, y_k)$. If we use x_i and x_{i+1} to represent the intersection of these two coordinates with the x axis. Then, the line is represented as $y = (x - x_i)\frac{(y_{i+1}-y_i)}{(x_{i+1}-x_i)} - y_i$. This line is mapped to the aggregation cells. The Map task needs to emit all the covered cells by the line. The dimensional coordinates might be arranged in any orders, which results in totally different visualizations.

Scatter-plot Matrix. A scatter plot simply maps two dimensional points, of which the two dimensions can be any two dimensions of the multidimensional data selected by the user. It is widely used as the major visualization method for low dimensional data. By enumerating all the combinations of pair dimensions, we get a scatter-plot matrix. The scatter plot approach can be very effective for identifying outliers in many applications. We demonstrate how scatter plot can be easily formulated with MapReduce (Figure 4.6). The

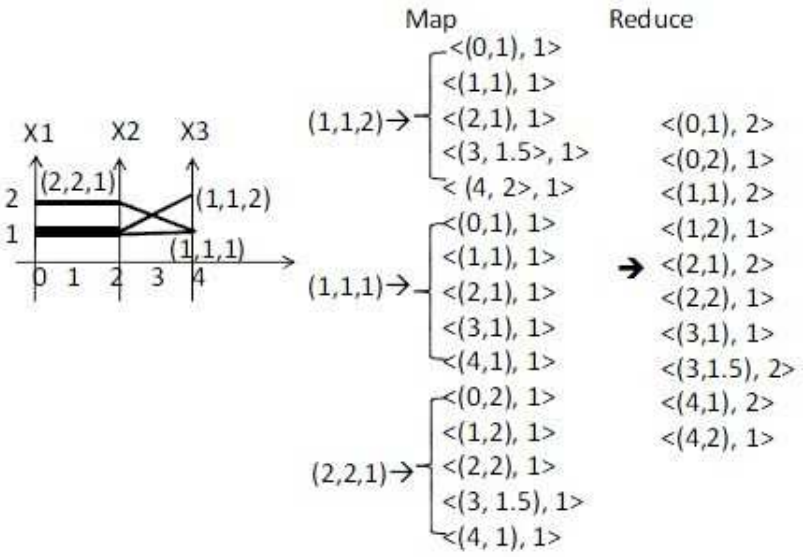

Figure 4.5: Parallel Coordinates

Map takes in a list of multidimensional points and the desired pair of dimensions. It maps each point to the cell coordinate and emits the pair < `cell id, 1`>. With the simple aggregate Reduce function, it results in the density-based scatter plot. It can be easily generalized to scatter plot matrix by emitting a pair < `(plot id, cell id), 1`> for each point and each pair of dimensions, where *plot id* identifies the pair of dimensions. The Reduce will group the pairs by the key `(plot id, cell id)`.

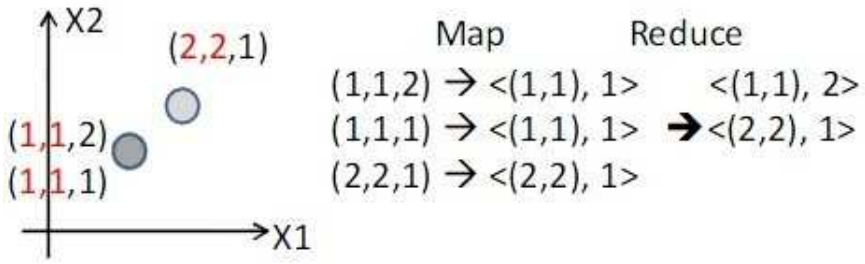

Figure 4.6: Scatter Plot

With the MapReduce formulation of the visualization models, and some batch frame processing and predication algorithms, we can integrate these visualization models in the CloudVista framework.

The CloudVista Demo System

This demonstration will show the novel features of CloudVista and also provide an intuitive way of understanding the limits of existing approaches in handling big data. Through the demonstration, users will observe the performance of RandGen algorithm to generate batches of visual frames and the utility of the batch frames. We implemented the CloudVista prototype with the VISTA visualization model [6] and hadoop/MapReduce programs, and the targeted deployment environment is the Amazon Cloud. At current stage, we have temporarily used the local Hadoop cluster (the nimbus cluster) to host the big data and the MapReduce processing. We also use a Tomcat server to implement the application server. Ultimately, when the system is deployed in the Amazon Cloud, the original datasets will be stored in S3; the hadoop cluster is provisioned dynamically in Amazon EC2 to generate visual frames or process subsets; and the simple queue service (SQS) is possibly used as the communication channel between the application server and the client software. The demonstration consists of two parts:

Live System: A fully interactive demonstration of the entire CloudVista prototype will be presented. The user will be able to use the client-side system to load a dataset, run RandGen to generate batches of visual frames, interactively explore visual frames, and drill down the selected subsets. Various system statistics will be fed back from the cloud to the client to help understand the cloud side cost.

Comparison with Existing Approaches: The existing approaches depend on the data reduction approaches to obtain a manageable size of data, which will severely downgrade

the visualization quality. We will compare the visualization results on the whole big dataset with those on sample sets or summary sets, to show the uniqueness of cluster exploration on big data.

5.1 Live System

The live system consists of a preliminary version of the client side user interface, the application server, and the MapReduce programs. The user will be able to use them to explore a sample dataset.

Client User Interface (UI). Figure 5.1 shows the UI of the client-side system. The visual frame is visualized with the heatmap method. The UI includes frame browsing (the lower slider and control widgets), frame zooming (the left vertical slider), and view moving (the four-direction widget). The top right also shows the status and statistics for some operations.

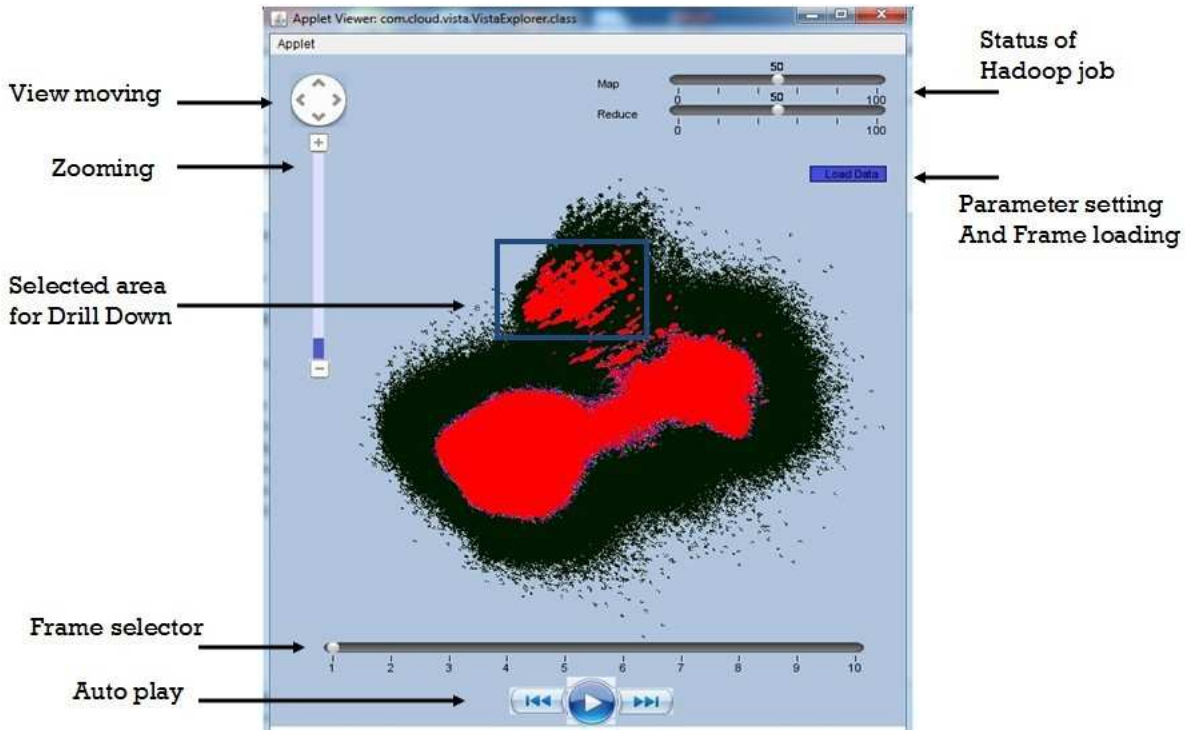

Figure 5.1: The client-side user interface

Visual Frame Generation: In the beginning of exploring a dataset, the user needs to select the dataset in the hadoop and set the necessary parameters. Figure 5.2 shows the parameter configuration. The "resolution" parameter defines the number of cells in the u and v directions of the display area (e.g., 1000×1000). The "number of frames" defines the number of frames that will be generated in each batch (e.g., 100). The "max sample size" is the maximum number of sample records that the original VISTA system can comfortably handle locally in the client side (e.g., 50,000). The "sample rate" is the user acceptable minimum sample rate that can preserve sufficient details of the clustering structure (e.g., 0.05). The top right corner of the main UI will show the progress and statistics of the RandGen algorithm, including the number seconds spent to generate the frames, the amount of data transferred to the client, the number of frames and the resolution. Once the user is not satisfied with the current batch of frames, she/he can generate another batch.

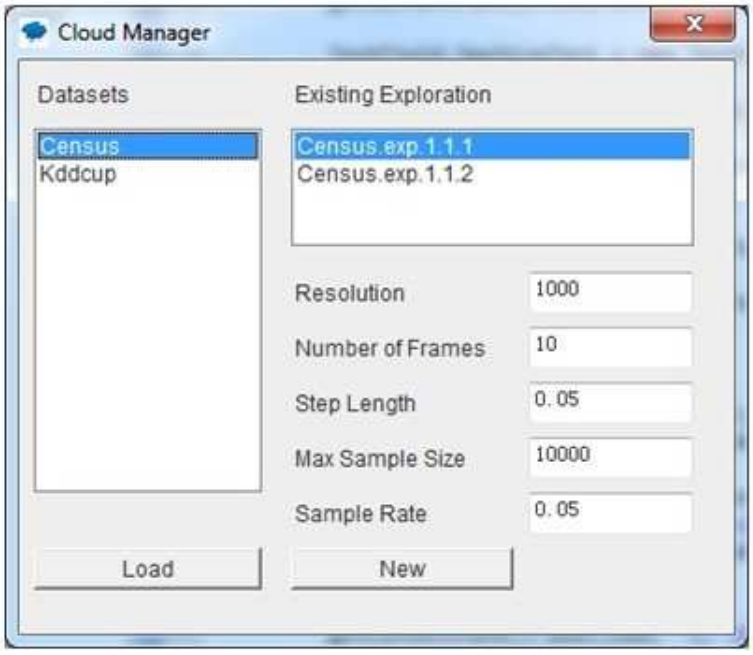

Figure 5.2: Parameter setting and frame loading

Basic Interactions. After the frames are generated, transferred, and loaded into the client system. The user can "autoplay" the frames or select any frame to observe its clustering details. The zooming widget allows the user to zoom in/out of the view. The

view moving widget allows the user to move the view to certain part of the frame. This is especially useful when the view is zoomed in.

Drill-down. The drill-down operation supports the hybrid exploration model that can efficiently reduce the size of data to be processed. If the user is interested in exploring more details in some area of a visual frame, she/he can drag the mouse to select the interested rectangle area. The drill-down operation may cause three different subset operations at the backend, according to the size of the selected subset: (1)subset RandGen for subsets still too large, (2) sampling for medium subsets, if sampling with the acceptable rate will generate an acceptable sample size that the client can handle locally, and (3) directly returning the entire subset for small subsets. The subset processing strategy and statistics are shown on the top right part of the window, including the time to finish the operation and the amount of data transferred. The subset exploration is also encoded as "dataset.exp_id.layer$_1$_seq.layer$_2$_seq...", where layer$_i$ means the drill-down layer, and "layer$_i$_seq" represents the sequential number of drill-down operations at the layer i. With this organization it is easy to roll back from the exploration of low-level subsets. All the intermediate results are buffered on the client side. Once a roll-back operation happens, the system can retrieve the related dataset based on the naming convention.

5.2 Comparative Study withVisualizing Reduced Data

We also want to compare the visualization results of the CloudVista entire-data approach with those of reduced data. Two common methods, sampling and summarization, will be used to reduce the big data.

Sampling. The uniform sampling algorithm is applied to get acceptable sample sizes (e.g., 50,000) from the big data in the cloud. Then, we will use the existing VISTA system [6] to visualize the sample datasets.

Summarization. We implement the BIRCH approach [53] for summarization. BIRCH

uses a clustering-feature (CF) tree to process the data records. Each data record is absorbed into one of the leaves of the CF tree. At last, each entry in the leaf node represents a cluster in the dataset. Depending on the height and the fanout of the tree, we can derive a number of small clusters in the end. These clusters are described by the number of points, the centroid (mean), and the variance. Each cluster can be approximated by a multidimensional normal distribution with the same mean and variance. We develop a MapReduce program proportionally draw samples from these multidimensional distributions to generate the density map, which are then visualized with the CloudVista frame viewer.

Experiments

The CloudVista framework enables users to interactively visualize the remote big datasets. A number of problems are critical to this framework. First, since the CloudVista framework is motivated by the sampling approach for handling big data, we want to understand the benefits brought by exploring the whole dataset instead of samples. Second, the conflict between the latency caused by processing big data and the required interactivity is critical to the usability of the framework. We want to understand how serious the latency is. We conduct a number of experiments to evaluate these two aspects.

6.1 Setup

We build the experimental prototype system with the in-house hadoop cluster, which consists of 15 worker nodes and 1 master node. We also put the application server on the master node, which may cause some performance issues in practice, but does not affect our experiments. Each of these nodes has two quad-core AMD CPUs, 16 GB memory, and two 500GB hard drives, which is connected to other nodes via a gigabit ethernet switch. According to the rule of thumb for Hadoop configuration - one core can have one Map and Reduce slot [50], we configure each worker node with eight map slots and six reduce slots. We assume the client side desktop computer can handle 50 thousands records with up to 100 dimensions as we have shown [7].

Two existing large scale datasets are extended to larger scale to evaluate the ability of processing large datasets. These two datasets are KDD Cup 1999 dataset with 41 attributes

and about 750 MB in total, and the Census 1990 dataset with 68 attributes and about 350 MB in total. Both can be found in UCI machine learning database. This resampling and perturbation method can approximately preserve the clustering structure when the dataset is extended. We describe the method as follows. First, the categorical attributes are replaced with a sequence of integers (an arbitrary mapping is defined between the categorical values and integer numbers), and then all attributes are normalized with simple methods such as max-min normalization. Second, a random sampling with replacement (i.e., the sampled record is not removed from the pool) is applied. For each sampled record, a random noise with normal distribution $N(0, 0.01)$ is added to each dimensional value to produce a new record. This process repeats for a number of times until we get sufficient number of records. Because the added noise is small enough, and the sampling is uniform, the clustering structure is preserved well. We denote these generated datasets with Census$_{ext}$ and KDD$_{ext}$ respectively.

6.2 Comparing Whole Data Visualization and Sample Data Visualization

The whole data visualization and the sample data visualization is compared to show the benefits of visualizing the whole dataset. The Census dataset is used in this experiment. The sample data visualization is based on 10,000 sample records, which are visualized with the VISTA system; the whole data visualization uses an extended dataset of 25 million records (5.3 GB in total), which are processed in the Hadoop cluster to generate the visual frames.

Figure 6.1 is one snapshot of visualizing the sample data with the VISTA system. We can observe about three dense areas - they are the clusters. These clusters form a hierarchical structure, marked as C1, C2.1, and C2.2, where C2.1 and C2.2 are close to each other. There are a number of questions unclear with this visualization. (1) C1 seemingly

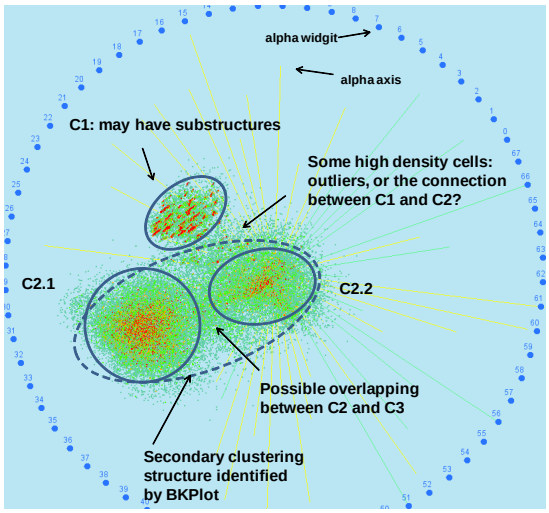

Figure 6.1: Visualization and Analysis of Census data (from [8]).

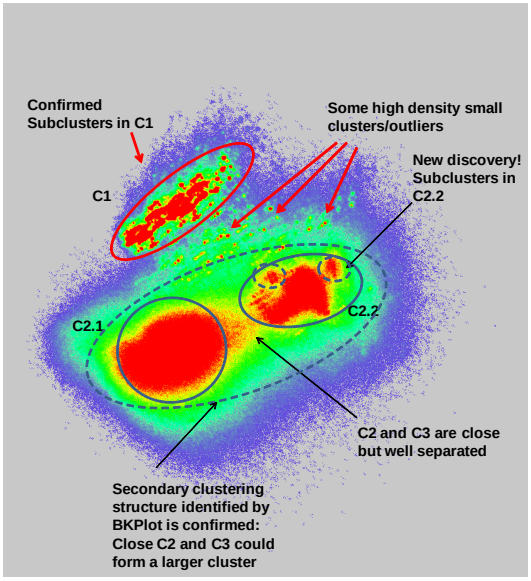

Figure 6.2: Visualization and Analysis of 25 Million Census records from [8]).

has some small clusters. (2) How the outliers are distributed? (3) How close are C2.1 and C2.2? Due to the small sample size, we cannot answer these questions.

In contrast, we can answer these questions with the whole-data visualization. Figure 6.2 is one of the visual frames that is most similar to the sample-data visualization. The previous questions can be answered. (1) We can confirm that C1 has many small clusters. Drilling down C1 may help us find more details of C1. (2) There are small clusters and outliers distributed between C1 and C2.2. (3) C2.1 and C2.2 are close to each other, but they are well separated - a density gap is observed between them. In addition, C2.2 also contains small sub-clusters, which are impossible to be observed in Figure 6.1.

It is clear that the whole-data approach preserved much more details of the cluster distribution. It is in particular good for observing small clusters and outliers. However, small sample sets are sufficient to find the big clusters.

6.3 Latency Evaluation

In previous discussion, we understand that the resolution will significantly affect the size of the visual frame, and may also impact the processing performance. Compared to Figure 6.2 in previous discussion, which is with an 1000x1000 resolution, the result of 250x250 resolution in Figure 6.3 has slightly reduced visual quality. Thus, it could be acceptable in the top level exploration, where the sketch of distribution is important. For any details, a drill-down operation can be applied.

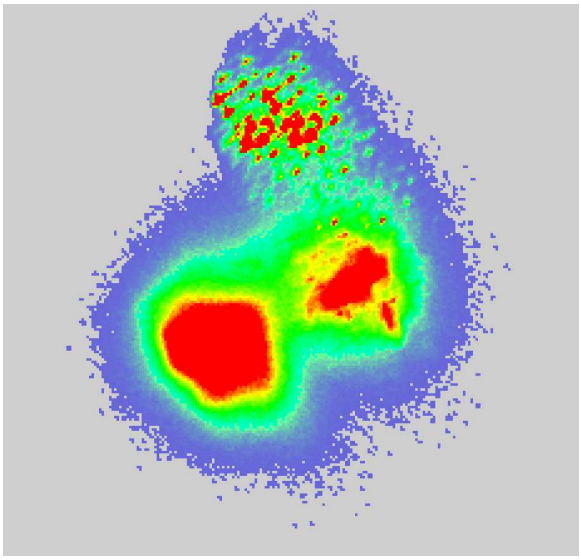

Figure 6.3: 12 Million Census records with 250x250 resolution.

The following latency evaluation compares these two resolutions and the impact of resolution to latency. As the cloud side cost can be split into three parts: MapReduce processing, data transmission cost from the cluster to the app server, result compressing and transferring to the client. Two extended datasets are used in experiments: 25 million records of Census (Census$_{ext}$) data and 40 million records of KDD Cup (KDD$_{ext}$) data. We also evaluate two resolutions 1000x1000 and 250x250. The result is presented in Table 6.1. "Frame size" represents the number of covered cells on average in each frame. The high-resolution frames have about 320 thousands of cells are covered on average per frame

for census data, while only 143 thousands for KDD cup data. Because of the reduced number of cells, the covered cells are about 10 times smaller for 250x250 resolution than the 1000x1000 resolution. We use the size of the application server's output (the compressed frames) to represent the cost of transmitting data from the cloud to the client. Again, the low resolution ones about 10 times smaller, which means faster cloud-client transmission. The "total time" is the total cost for cloud-side processing. Thus, low resolution results represent a significant cost saving compared to the high resolution ones.

	resolution	frame size	compressed frames	total time(sec)
Census$_{ext}$	High	320K	100MB	247
	Low	25K	9.7MB	141
KDD$_{ext}$	High	143K	45MB	265
	Low	12K	4.6MB	188

Table 6.1: Summary of the RandGen experiment.

Conclusion and Future Work

We address the limitation of the existing three-phase framework for cluster analysis on big data with the CloudVista framework. This framework assumes the data is in the cloud, which enables economical on-demand massive-scale processing on the whole dataset. The whole-data approach can preserve fine structures such as small clusters and outliers and give users higher confidence on the visualization results. However, the cloud-side processing on the whole data may bring high latency, which conflicts with the requirement of interactivity. We developed the batch-interaction model supported with the RandGen algorithm to achieve a balance between the high throughput and the interactivity. The results show that the whole-data approach can be satisfactorily implemented under the CloudVista framework.

We plan to conduct the following future work. (1) We will experiment with larger Hadoop clusters in Amazon EC2, and evaluate the performance with the public cloud setting. (2) We will apply the resource provisioning model developed in another project [47] to optimize the batch requesting and processing strategies. (3) We will extend the CloudVista framework to handle more multidimensional visualization models such as Parallel Coordinates and Scatterplot Matrix as mentioned in Section 4.2.1.

Bibliography

[1] D Asimov. The grand tour: A tool for viewing multidimensional. *SIAM Journal of Scientific and Statistical Computing*, 6(1):128–143, 1985.

[2] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Proceedings Of Neural Information Processing Systems (NIPS)*, pages 585–591. MIT Press, 2001.

[3] Paul S. Bradley, Usama M. Fayyad, and Cory Reina. Scaling clustering algorithms to large databases. In *Proceedings of ACM SIGKDD Conference*, pages 9–15, 1998.

[4] Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufman, 1999.

[5] Keke Chen and Ling Liu. Clustermap: Labeling clusters in large datasets via visualization. In *Proceedings of ACM Conference on Information and Knowledge Management (CIKM)*, pages 285–293, 2004.

[6] Keke Chen and Ling Liu. VISTA: Validating and refining clusters via visualization. *Information Visualization*, 3(4):257–270, 2004.

[7] Keke Chen and Ling Liu. iVIBRATE: Interactive visualization based framework for clustering large datasets. *ACM Transactions on Information Systems*, 24(2):245–292, 2006.

[8] Keke Chen, Huiqi Xu, Fenggung Tian, and Shumin Guo. Cloudvista: Visual cluster exploration for extreme scale data in the cloud. In *Proceedings of International Conference on Scientific and Statistical Database Management (SSDBM)*, 2011.

[9] Paolo Cignoni, Claudio Montani, Enrico Puppo, and Roberto Scopigno. Multiresolution representation and visualization of volume data. *IEEE Transactions on Visualization and Computer Graphics*, 3(4), 1997.

[10] Dianne Cook, Andreas Buja, Javier Cabrera, and Catherine Hurley. Grand tour and projection pursuit. *Journal of Computational and Graphical Statistics*, 23:155–172, 1995.

[11] Trevor F. Cox and Michael A. A. Cox. *Multidimensional Scaling*. Chapman and Hall/CRC, Boca Raton, FL, US, 2001.

[12] Vin De Silva and Joshua B. Tenenbaum. Global Versus Local Methods in Nonlinear Dimensionality Reduction. In *Advances in Neural Information Processing Systems 15*, volume 15, pages 705–712, 2003.

[13] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. In *USENIX Symposium on Operating Systems Design and Implementation*, 2004.

[14] David L. Donoho and Carrie Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences*, 102(21):74267431, 2005.

[15] Christos Faloutsos and King-Ip (David) Lin. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proceedings of ACM SIGMOD Conference*, pages 163–174, 1995.

[16] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[17] James Fowler and Roni Yagel. Lossless compression of volume data. In *The 1994 Symposium on Volume Visualization*, pages 43–50, 1994.

[18] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. In *USENIX Symposium on Operating Systems Design and Implementation*, 2003.

[19] Richard L. Gorsuch. *Factor Analysis*. Lawrence Erlbaum, 1983.

[20] Georges Grinstein, Mihael Ankerst, and Danial A. Keim. Visual data mining: Background, applications, ad drug discovery applications. In *Proceedings of ACM SIGMOD Conference*, 1999.

[21] Keith Grochow, Bill Howe, Roger Barga, and Ed Lazowska. Client + cloud: Seamless architectures for visual data analytics in the ocean sciences. In *Proceedings of International Conference on Scientific and Statistical Database Management (SSDBM)*, 2010.

[22] S. Guha, R. Rastogi, and K. Shim. Cure: An efficient clustering algorithm for large databases. In *In Proceedings of ACM SIGMOD Conference*, pages 73–84, 1998.

[23] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: An efficient clustering algorithm for large databases. In *Proceedings of ACM SIGMOD Conference*, pages 73–84, 1998.

[24] Baining Guo. A multiscale model for structure-based volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(4), 1995.

[25] Trevor Hastie and Werner Stuetzle. Principal curves. *Journal of the American Statistical Association*, 1989.

[26] Alexander Hinneburg, Daniel A. Keim, and Markus Wawryniuk. Visual mining of high-dimensional data. In *IEEE Computer Graphics and Applications*, pages 1–8, 1999.

[27] Patrick Hoffman, Georges Grinstein, Kenneth Marx, Ivo Grosse, and Eugene Stanley. Dna visual and analytic data mining. In *IEEE Visualization*, pages 437–442, 1997.

[28] Aapo Hyvarinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10(3):626–634, 1999.

[29] Alfred. Inselberg. Multidimensional detective. In *IEEE Symposium on Information Visualization*, pages 100–107, 1997.

[30] Alfred Inselberg. *Parallel Coordinates: Visual Multidimensional Geometry and Its Applications*. springer, 2009.

[31] J. Edward Jackson. *A User's Guide to Principal Components*. Wiley, 2003.

[32] Eser Kandogan. Visualizing multi-dimensional clusters, trends, and outliers using star coordinates. In *Proceedings of ACM SIGKDD Conference*, pages 107–116, 2001.

[33] U Kang, Charalampos E. Tsourakakis, and Christos Faloutsos. Pegasus: Mining peta-scale graphs. *Knowledge and Information Systems (KAIS)*, 2010.

[34] Danial Keim. Visual exploration of large data sets. *ACM Communication*, 44(8):38–44, 2001.

[35] Daniel A. Keim. Designing pixel-oriented visualization techniques: Theory and applications. *IEEE Transactions on Visualization and Computer Graphics*, 6:59–78, January 2000.

[36] Daniel A. Keim, Jörn Schneidewind, and Mike Sips. Scalable pixel based visual data exploration. In *Proceedings of the 1st first visual information expert conference on*

Pixelization paradigm, VIEW'06, pages 12–24, Berlin, Heidelberg, 2007. Springer-Verlag.

[37] Teuvo Kohonen. *Self Organizing Map*. Springer, 2001.

[38] Stephane Lafon and Ann B. Lee. Diffusion maps and coarse-graining: a unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1392–1403, 2006.

[39] Jimmy Lin and Chris Dyer. *Data-intensive text processing with MapReduce*. Morgan and Claypool Publishers, 2010.

[40] Paul Ning and Lambertus Hesselink. Fast volume rendering of compressed data. In *Proceedings of Visualization*, 1993.

[41] Biswanath Panda, Joshua S. Herbach, Sugato Basu, and Roberto J. Bayardo. Planet: Massively parall learning of tree ensembles with mapreduce. In *Proceedings of Very Large Databases Conference (VLDB)*, 2009.

[42] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[43] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[44] Jinwook Seo and Ben Shneiderman. Interactively exploring hierarchical clustering results. *IEEE Computer*, 35(7):80–86, 2002.

[45] Ben Shneiderman. Inventing discovery tools: Combining information visualization with data mining. *Information Visualization*, 1:5–12, 2002.

[46] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(22), 2000.

[47] Fenggung Tian and Keke Chen. Towards optimal resource provisioning for running mapreduce programs in public clouds. In *IEEE Conference on Cloud Computing*, 2011.

[48] Santosh S. Vempala. *The Random Projection Method*. American Mathematical Society, 2005.

[49] Jason Tsong-Li Wang, Xiong Wang, Dennis Shasha, and Kaizhong Zhang. Metricmap: an embedding technique for processing distance-based queries in metric spaces. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, pages 973–987, 2005.

[50] Tom White. *Hadoop: The Definitive Guide*. O'Reilly Media, 2009.

[51] Jing Yang, Matthew O. Ward, and Elke A. Rundensteiner. Interactive hierarchical displays: a general framework for visualization and exploration of large multivariate datasets. *Computers and Graphics Journal*, 27:265–283, 2002.

[52] Li Yang. Interactive exploration of very large relational datasets through 3d dynamic projections. In *Proceedings of ACM SIGKDD Conference*, pages 236–243, 2000.

[53] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: An efficient data clustering method for very large databases. In *Proceedings of ACM SIGMOD Conference*, pages 103–114, 1996.