# Effective Big Data Visualization

Murali Mani
University of Michigan Flint
mmani@umflint.edu

Si Fei
University of Michigan Flint
sfei@umflint.edu

## ABSTRACT

In the last several years, big data analytics has found an increasing role in our everyday lives. Data visualization has long been accepted as an integral part of data analytics. However, data visualization systems are not equipped to handle the complexities typically found in big data. Our work examines effective ways of visualizing big data, while also realizing that most visualization processes are interactive. During an interactive visualization session, an analyst issues several visualization requests, each of which builds on prior visualizations. In our approach, we integrate a distributed data processing system that can effectively process big data with a visualization system that can provide effective interactive visualization but for smaller amounts of data. The analyst's current request is used to infer contextual information about the analyst such as their expertise and tolerance for delay. This information is used to carefully determine additional data that can be sent to the visualization system for decreasing the response time for future requests, thus providing a better experience for the analyst and increasing their productivity.

## CCS CONCEPTS

• **Information systems → Data analytics**; • **Human-centered computing → Visual analytics**;

## KEYWORDS

Big Data, Data Visualization, Data Analytics

**ACM Reference format:**
Murali Mani and Si Fei. 2017. Effective Big Data Visualization. In *Proceedings of IDEAS '17, Bristol, United Kingdom, July 12-14, 2017,* 6 pages.
https://doi.org/10.1145/3105831.3105857

## 1 INTRODUCTION

Data analysis refers to the process of discovering potential clues, reliable facts, and possible outcomes via the use of analytical techniques on data. It helps individuals extract information from data, which can be used to make informed decisions [5]. Two main categories of data analysis include direct and exploratory [5]. In direct analysis, we start with a specific question that we want to answer, and then find the answer through data analytical techniques. It has a clear precondition that we must know what we want to analyze, and it is purposeful. On the other hand, exploratory analysis is not purposeful and is used when we do not have a predetermined question to answer. Exploratory analysis starts to simply look at data and search for possible interesting findings. Techniques such as data visualization are used in both direct and exploratory analysis.

Data analytics has become even more important with the advent of big data [1], as extensive analysis is needed before we can extract useful information from big data. In [1], the authors identify astronomy, biological sciences, education, healthcare, urban planning, transportation, energy, homeland security, computer security, social sciences, and financial systems as some of the applications of big data. Further the volume, variety and velocity (referred to as the 3Vs) of big data [8] make traditional approaches of data analytics not useful for big data analytics (note that more Vs have been added to describe big data since the original 3 Vs).

Big data analytics has been made possible largely because of the tremendous improvements made in distributed processing in the last few years, since map-reduce programming model was introduced [4]. In map-reduce programming, the software developers have only a small learning curve – a program is specified as a sequence of map and reduce sub-routines. The task of distributing the map and reduce tasks across machines and executing them in parallel, managing all communications and data transfers, providing for redundancy and fault tolerance are all automatically done by frameworks such as Apache Hadoop [13], and the more recent Apache Spark [15]. These frameworks have been found to scale up to several thousands of machines.

Data visualization has a critical role in data analytics [5], and an even more critical role for big data analytics, where it is almost always impossible for the user to see the information in the data without effective analysis. The HCI and data visualization communities have long recognized the importance of fast visualization for productive analysis, including for big data analytics [7, 9]. In addition to quick response times, researchers have identified that the analysis process is typically iterative, where an analyst issues follow-up refinements to prior questions [3, 7]. In [7], the authors state: "*The goal of interactive analysis tools is to empower data analysts to formulate and assess hypotheses in a rapid, iterative manner*".

In our work, we tackle the problem of interactive big data visualization. There has been prior work on interactive big data analytics based on progressive processing (also called incremental processing, incremental visualization), where the system produces results based on partially processed data and progressively refines these results as more data is incrementally processed [3, 6]. Other prior work rely on fast and approximate big data processing and visualization using data reduction techniques such as sampling [2, 10].

For effective big data visualization, we integrate two systems: (a) a data processing system that can effectively process big data despite its various complexities, and (b) a visualization system that can effectively provide interactive visualization, but cannot typically handle the complexities of big data. Visualization requests that can be processed with the data already available at the visualization system result in much smaller wait times for the analyst. Our system sends additional data to the visualization system that can be useful for future requests. The space of future requests that an analyst is likely to explore is predicted based on the contextual information about the user's expertise inferred from the current request. In addition, different users have different tolerance for delay. For instance, an analyst beginning to explore a dataset might be more tolerant for delays in hope that future requests will have lesser delay, whereas a company executive who is executing one request in order to incorporate the result in a corporate presentation may be much less tolerant for delays. The tolerance for delay is also predicted by our system based on the current request.

Our contributions include: (a) we illustrate how the user request can be used to infer contextual information, specifically the user's expertise (which indicates the space of possible future requests) and the user's tolerance for delay, (b) we devise an algorithm that uses the contextual information about the user, along with the knowledge of the visualization system resources available to determine the additional data to be sent to the visualization system, (c) we validate our approach with experimental studies.

**Outline:** The rest of the paper is structured as follows. In Section 2, we given an overview of big data processing and data visualization. Section 3 describes our approach, where we describe how contextual information about the user is inferred and how this information is used to determine the additional data to be sent to the visualization system. In Section 4, we discuss our experimental studies. Section 5 discusses related work, and Section 6 concludes our work.

## 2 BACKGROUND

In this section, we examine the two components of our system: big data processing systems, and visualization systems.

### 2.1 Big Data Processing

Map-reduce programming model [4] has revolutionized distributed data processing with its ability to use a large number of machines for a data processing task easily. Here, programmers write their task as map and reduce sub-routines. Frameworks such as Apache Hadoop [13] and Apache Spark [15] split the input dataset into several data segments, which are processed by these map and reduce sub-routines in a parallel manner. Map-reduce has scaled up data processing to several thousands of machines as reported at https://wiki.apache.org/hadoop/PoweredBy. Our integrated system can use either Spark/Hadoop that process the entire data, or any incremental big data processing system or data reduction system [2, 3, 6, 10].

### 2.2 Data Visualization

Data visualization is an integral part of data analysis, and it is ideal for both directed and exploratory analysis [5]. Data visualization concentrates on use of tables and graphs for presenting data, especially quantitative information, to the user and for the user to communicate with the data. Exploring data using images (different kind of graphs) enables our eyes to easily locate meaningful information through trends, patterns, relationships etc. In addition, an image of the relevant data is a good way to find hidden information.

Different types of visualization correspond to different kinds of information. Three most commonly used types of visualization include line graph, bar graph and scatter plot. Appropriate visualizations should be used for a particular task; otherwise we may come to incorrect conclusions. For instance, line graph works perfectly for connecting values through time, such as monthly or daily analysis. However, line graph is not appropriate to analyze categorical items because categorical items are typically independent and do not have obvious relationship between each other. Instead, a bar graph is more suited to explore categorical items. In addition, to handle complex data, sometimes multiple types of visualizations are combined for analysis [5].

## 3 SMART SYSTEM

As mentioned earlier, we integrate data processing and visualization systems. Figure 1 illustrates the interaction between analysts, visualization system and data processing system in our scenario.

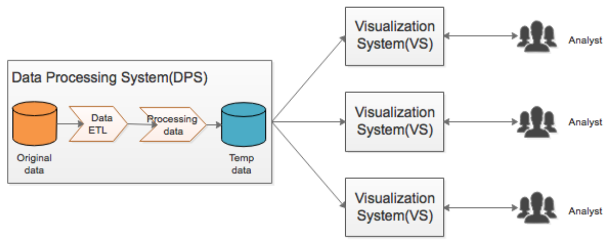

**Figure 1: Multiple analysts are interacting with their own visualization systems. The visualizations systems interact with a centralized data processing system. The data processing system processes the raw original data based on the request and sends the results to the visualization system that performs and displays the visualization to the analyst.**

A simplistic integration of data processing and visualization systems could push all the processing to the data processing system. This ensures that the resource-constrained visualization system does minimal work, requires minimal amount of resources and handles minimal amount of data, while the resource-rich data processing system does the heavy-lifting of processing the large amount of data. However such an integration does not support effective interaction as every visualization request will require the data processing system to perform the processing. This typically results in high latency: (a) the data processing system has to process a large amount of data, and (b) the results of the data processing needs to be transferred to the visualization system. The user experience can be improved vastly if the user request can be handled by the visualization system using a decreased amount of data, without requiring any processing by the data processing system.

In our work, we propose intelligent integration of a data processing system and a visualization system that minimizes the number of requests that require processing by the data processing system.

Additional data is sent to the visualization system when a user performs a visualization, and this data can be used for future visualization requests without any processing performed by the data processing system. Figure 2 shows our integrated framework.

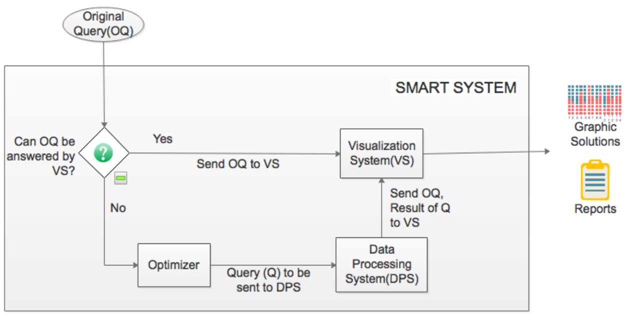
Original Query(OQ)
SMART SYSTEM
Can OQ be answered by VS?
Yes
Send OQ to VS
Visualization System(VS)
Graphic Solutions
Reports
No
Send OQ, Result of Q to VS
Optimizer
Query (Q) to be sent to DPS
Data Processing System(DPS)

**Figure 2: SMART System: when an analyst issues a request (OQ), the system first checks whether it can be answered with the data available at VS. If yes, then OQ is sent to VS and VS produces the necessary visualization results. If no, then the SMART optimizer comes up with a query (Q) to be sent to the DPS. DPS sends OQ and the results of Q to VS, and VS produces the necessary visualization results. The results of Q might have additional data to answer future requests.**

It may appear that the user has the best experience if the visualization system has as much data as possible. However, that is not true. If the visualization system has more data than what will be used in future requests, then the user experience will suffer as the time taken by the visualization system for processing each request will be higher. Further, if additional data is sent to the visualization system than needed for the request, it may help interaction and future requests, but it will have a negative effect on user experience for this request. In short, we want the visualization system to have the least amount of data, while still ensuring that most future requests from the user can be processed by the visualization system without any processing by the data processing system.

The SMART optimizer (Figure 2) determines the processing done by the data processing system and the additional data sent to the visualization system. Note the following.

- The user experiences the least latency if the answer to the question *Can OQ be answered by VS?* in Figure 2 is Yes. Here, the user request is handled entirely by the visualization system, and no processing is done by the data processing system.
- If a user request requires processing by the data processing system, then the user latency is the least if the data processing system does the maximum processing. In this case, least amount of data is transferred to the visualization system, and least amount of processing is done by the visualization system. However, future requests will likely require processing by the data processing system, increasing their latency.

## 3.1 Visualization Request

A visualization request can be considered as having three parts (a) zero or more dimensions (b) one or more measures and (c) zero or more additional features such as filters. We represent a visualization request as a 3-tuple (D, M, F), where the data is filtered based on the filters F, grouped by based on the dimensions D, and the measures M are aggregated. It represents the following SQL:

```
SELECT D, AGG(M)
FROM Data
WHERE F
GROUP BY D
```

Figure 3 shows yearly values for children in poverty and unemployment in CA from the dataset available at [16]. The schema for this dataset (simplified) is: Data (State, County, Year, Feature, Value) Here, D = {Year, Feature}; M = {Value}; F = {(State = 'CA'), (Feature IN {'Unemployment', 'Children in poverty'})}.

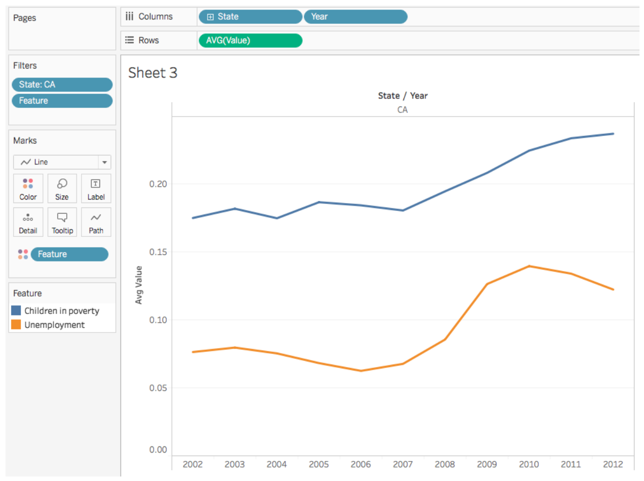

**Figure 3: Example visualization: line graph showing yearly children in poverty and unemployment rates in CA state.**

## 3.2 Information about the User

The SMART system tries to understand more about the user from a user request. A user is classified along two dimensions: (a) user's expertise with the data, and (b) user's tolerance for delay. If a user issues a request with very few dimensions, very few measures and very few filters, the user is likely to have little experience with this dataset. Such a user also typically has high tolerance for delay as they are exploring the dataset before beginning to do more in-depth analysis. As the number of dimensions, measures and filters in the user request increases, the user is likely to have more experience with this dataset and to issue follow-up requests. Such a user is likely to be interested in the delay over the entire user session and not just the delay for this one request. Note that a user's request is used to classify the user, and no user login is required.

## 3.3 SMART Algorithm

If the visualization system does not have the data to answer a visualization request, and the user has high tolerance for delay, additional data potentially useful for future requests will be sent to the visualization system. On the other hand, if a user has low tolerance for delay, as little data as needed will be sent to the visualization system. The SMART system knows the resources available at the visualization system, and the visualization system is not sent more data than what it can handle. Algorithm 1 is used by the SMART system. The parameters for the algorithm include:

OrigQ = the current user request
VSResources = resources available at the visualization system
DelayTolerance = the tolerance for delay for this user (expressed as time that the user is willing to wait for the visualization result)
UserExpertise = quantified representation of the user expertise

The output of the algorithm is the query Q that our SMART system calculates to be the best query to be executed by the data processing system. OrigQ and the result of Q are sent to the visualization system, which then performs the remaining processing and displays the final visualization result of OrigQ to the user. The SMART algorithm uses the following functions that are defined:

ComputeResources (Q1, Q2): estimates the resources needed at the visualization system to handle the user request Q1 when the data processing system executes Q2
TimeNeeded (Q1, Q2): estimates the total time needed to handle user request Q1 when the data processing system executes Q2
ExpertLevel (Q1, Q2): estimates the expertise needed for a user whose original request is Q1 to issue Q2 as a follow-up request (Q2 may be/may not be issued immediately after Q1)

Two more additional variables (for book-keeping) include: Q = currQuery to be sent to the data processing system (initialized to origQ); Q′ = query that is being checked by our system to be sent to the data processing system (also initialized to origQ).

**Algorithm 1** SMART Optimizer Algorithm

```
1: while (VSResources ≥ ComputeResources (origQ, Q′) AND DelayTolerance ≥ TimeNeeded (origQ, Q′) AND UserExpertise ≥ ExpertLevel (origQ, Q′)) do
2:     Q = Q′;
3:     if (F is not empty) then
4:                                         ▷ first remove a filter from Q if present
5:         Q′ = Q after correctly removing one of the filters in Q;
6:     else if there are more related dimensions in data not present in Q then    ▷ second add a related dimension
7:         Q′ = Q after adding the dimension closest related to one of the dimensions in Q;
8:     else if there are more un-related dimensions in data not present in Q then    ▷ third add a not related dimension
9:         Q′ = Q after adding 1 more un-related dimension;
10:    else if there are more measures in data not present in Q then    ▷ fourth, add a measure
11:        Q′ = Q after adding 1 more measure;
12: return Q;
```

Algorithm 1 uses "related" and "un-related" dimensions. Two dimensions D1 and D2 are related to each other if one of the two functional dependencies is defined: D1 → D2 (or) D2 → D1. If no such functional dependency exists between D1 and D2, then D1 is un-related to D2. For instance, city and state are related (city → state); quarter and month are related (month → quarter); gender and age are un-related. Also for related dimensions, the degree of closeness is defined: Consider dimensions D2 and D3 that are both related to D1 because of functional dependencies D1 → D2, and D1 → D3. D2 is closer to D1 than D3, if D2 → D3. Similarly, if D2 → D1, and D3 → D1, then D2 is closer to D1 than D3, if D3 → D2. For example, quarter is more closely related to year than month.

Algorithm 1 first removes filters one by one (for correctness, when a filter is removed, the columns in the filter are added to the set of dimensions), then adds related dimensions one by one, then adds un-related dimensions one by one, and finally adds measures one-by-one. This ordering is based on our observations of typical user interactions. No order is specified between filters (if there are many filters), between related dimensions, between un-related dimensions, or between measures. Investigating the different orders, including studying whether a dimension can be added before a filter is removed, are outside the scope of our current project, and could be interesting future work.

**Example:** Consider our example of comparing the trends in yearly unemployment and children in poverty. The schema and origQ are:
Data (State, County, Year, Feature, Value)
OrigQ: D = {Year, Feature}; M = {Value}; F = {(State = 'CA'), (Feature IN {'Unemployment', 'Children in poverty'})}

Removing filters implies getting the data for all states, and also for all features (so an analyst can find other correlations as well). When the filter (State = 'CA') is removed, the column State is added to D. Then the algorithm tries to add the dimension county. This simplistic example has no more measures. In short, if the visualization system has lot of resources, and the user's expertise and tolerance for delay are very high, then our algorithm comes up with the following query for the data processing system:
Q: D = {Year, Feature, State, County}; M = {Value}; F = {}

```
SELECT Year, Feature, State, County, AGG(Value)
FROM Data
GROUP BY Year, Feature, State, County
```

If the results of Q are sent to the visualization system, then the visualization system can answer many requests directly, without requiring any processing by the data processing system including comparing across states or counties, and comparing other features.

## 4 EVALUATION

In this section, we compare the performance of different ways of integrating the data processing system and the visualization system. We use the TPC-H benchmark data [17], specifically Supplier, Customer and Part tables (Supplier/Customer tables are joined with Nation and Region tables; so for each supplier/customer, we know the nation and region). We used a scale factor of 1 (the total size of the data across all the eight tables is 1 GB). Even at this scale factor, our visualization system cannot perform group by on the returnflag values in the Lineitem table which has more than 6 million rows. For each of the tables, we came up with a original request and a follow-up request. Our test requests have no filters, and the measure used is to count the number of rows in each group. We considered three options for each table. The tables, the requests and the SQL for the original request corresponding to the three options are given below.

| Table | # rows in Table | original request dimensions (D1) | follow-up request dimensions (D2) |
|---|---|---|---|
| Supplier | 10,000 | nation name | region name |
| Customer | 150,000 | nation name | region name |
| Part | 200,000 | manufacturer | brand |

| Option | Data processing system query for original request |
|---|---|
| Option1 | SELECT * FROM Table |
| Option2 | SELECT D1, COUNT(*) FROM Table GROUP BY D1 |
| Option3 | SELECT D1, D2, COUNT(*) FROM Table GROUP BY D1, D2 |

For Options 1 and 3, the original and follow-up requests can be handled by the results of the data processing system query for the original request. Therefore, the data processing system does no processing for answering the follow-up request. This is not the case for Option 2, and the follow-up request requires processing by the data processing system as well. For handling this follow-up request, we assume that the data processing system will perform as much processing as possible.

For our experimental study, we use R [14] for both data processing and for visualization. Any other data processing and visualization systems can be used instead of R as well. We picked the same software for both data processing and visualization so that the differences in implementation details of different software do not skew our results. To validate this choice, we compared R and Spark. For the query: SELECT region name, COUNT (*) FROM Supplier, R took 0.084 seconds, whereas Spark took 0.370 seconds. For visualizing the above result as a table, R took 0.003 seconds whereas Spark took 18.235 seconds. As the options we study involve performing the same operations sometimes in the data processing system or in the visualization system, we use the same software (we picked R) for both data processing and visualization.

For identical reasons as above, we used the same machine for both data processing and data visualization – a 64-bit 8-core machine with a 2.4 GHz Intel i7 chip running Windows 7. We simulated the data transfer between the data processing and visualization systems by piping the output of the data processing system to a text file, which is then read by the visualization system.

Figure 4 shows the total amount of time (this includes processing by the data processing and visualization systems and data transfer) for the three options for the three tables. The trend is almost identical for all the three cases. Option 1 needs a large amount of time for the original request, because of the large amount of data transferred, and because the visualization system processes a much larger amount of data. Option 2 has the smallest amount of time for the original request. Option 3 needs marginally more time than Option 2, as more data is transferred in Option 3 and the visualization system does more processing, as one more dimension is added in Option 3. However, the dimension added is a related dimension and the increase in data transferred is quite small. Actually, for Supplier and Part tables, no extra rows are transferred in Option 3 as compared to Option 2. For Part, more rows are transferred in Option 3 than Option 2. Table 1 shows the detailed numbers for each option for each table. For the follow-up request, both Option 1 and Option 3 do not require any processing by the data processing system and outperforms Option 2 by a huge margin. However, Option 3 outperforms Option 1 because Option 1 has too much extra data at the visualization system. For Supplier, Option 3 (25 rows at VS) needs 44% of the time needed by Option 1 (10,000 rows at VS). For Customer, Option 3 (25 rows at VS) needs less than 4% of the time needed by Option 1 (150,000 rows at VS).

We can see in our experimental studies that Option 3 is preferable to other options, as it results in a better experience for the user – there is a slightly increased delay for the original request, but the delay for future requests is reduced significantly. Option 3 carefully sends extra data to the visualization system in response to the original request. To summarize, if we carefully send extra data to the visualization system in response to the original request, it is possible to cut down delays for future requests and significantly improve user experience.

## 5 RELATED WORK

There has been prior work that study visualization of very large databases. These techniques rely on data reduction using sampling. For instance [2] uses uniform random sampling, whereas [10] uses a visualization-aware sampling specifically targeting regression, density estimation and clustering. While visualization-aware sampling provided better visualization results over uniform random sampling, it is not general and needs to be investigated for the many different types of analyst questions. Data reduction approaches are orthogonal to our solution, and data processing systems that employ data reduction can be used by our system as well.

Techniques that can be used as part of visualization systems to handle large datasets have also been investigated by other researchers. For instance, parallel rendering has been studied in [11], incremental visualization has been studied in [6]. Again, these approaches are orthogonal to our solution, and visualization systems that employ similar techniques can be incorporated into our system.

A system that integrates data processing and visualization systems is SparkR, available as part of Spark [15]. However, SparkR pushes as much processing as possible to the data processing system, which gives excellent experience for stand-alone requests. Our approach could result in more processing done at the visualization system than SparkR, but it results in a better experience for the user in an interactive session with multiple requests.

## 6 CONCLUSIONS AND FUTURE WORK

In this paper, we studied effective ways of integrating data processing systems that can handle the complexities of big data with visualization systems that can provide interactive visualization so that we can achieve the goal of interactive big data visualization and big data analytics. Our system studied how a request from an analyst can be used to infer contextual information about the analyst such as their expertise and tolerance for delay. This user knowledge is used by our system to carefully determine the additional data sent to the visualization system in response to a user request; this results in an increased delay for the current request, but there could be potentially significant time savings for future requests. Our experimental evaluation validates our approach.

There are several avenues of future work that are interesting. First, note that given a request, our system removes filters, then adds dimensions and measures in that order. This is based on observations of typical user interactions during data analytics. However,

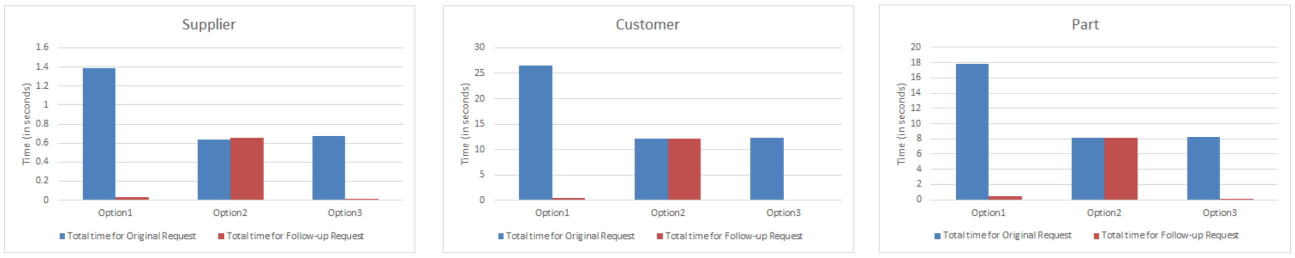

**Figure 4: The total time for the original and follow-up requests for Supplier, Customer and Part tables for the three options.**

| | Supplier | | | Customer | | | Part | | |
|---|---|---|---|---|---|---|---|---|---|
| | O1 | O2 | O3 | O1 | O2 | O3 | O1 | O2 | O3 |
| DPS-Orig | 0.618 | 0.582 | 0.6 | 11.852 | 12.124 | 12.188 | 7.616 | 8.086 | 8.184 |
| VS-Orig | 0.054 | 0.008 | 0.024 | 0.398 | 0.014 | 0.014 | 0.476 | 0.014 | 0.016 |
| Transfer-Orig | 0.716 | 0.044 | 0.05 | 14.32 | 0.052 | 0.052 | 9.77 | 0.046 | 0.054 |
| Total-Orig | 1.388 | 0.634 | 0.674 | 26.57 | 12.19 | 12.254 | 17.862 | 8.146 | 8.254 |
| Rows-Orig | 10K | 25 | 25 | 150K | 25 | 25 | 200K | 5 | 25 |
| KB-Orig | 2,018 | 0.8 | 1.4 | 32,572 | 0.8 | 1.5 | 24,111 | 0.1 | 0.8 |
| DPS-Next | 0 | 0.59 | 0 | 0 | 12.048 | 0 | 0 | 8.05 | 0 |
| VS-Next | 0.036 | 0.012 | 0.016 | 0.422 | 0.018 | 0.016 | 0.472 | 0 | 0.014 |
| Transfer-Next | 0 | 0.056 | 0 | 0 | 0.046 | 0 | 0 | 0.056 | 0 |
| Total-Next | 0.036 | 0.658 | 0.016 | 0.422 | 12.112 | 0.016 | 0.472 | 8.106 | 0.014 |
| Rows-Next | 0 | 5 | 0 | 0 | 5 | 0 | 0 | 25 | 0 |
| KB-Next | 0 | 0.2 | 0 | 0 | 0.2 | 0 | 0 | 0.4 | 0 |

**Table 1: Table showing the detailed numbers from our experiments. O1, O2, O3 refer to the three options. DPS-Orig, VS-Orig, Transfer-Orig, Total-Orig refer to the different times for the original request – time taken by the data processing system, time taken by the visualization system, time taken for transfer, and the total time taken. DPS-Next, VS-Next, Transfer-Next, Total-Next refer to the times taken for the follow-up request. Rows-Orig refers to the number of rows of data transferred from the data processing system to the visualization system, KB-Orig refers to the size of this data transfer in KB. Rows-Next, KB-Next refer to the data transferred for the follow-up request. All the times are in seconds, and the size of data is in KB.**

this needs to be investigated more, as an analyst might want to add a dimension before removing a filter, or removing one filter might be more relevant than removing a different filter. In addition, our system considered a data processing system that gives exact results to a request. To handle the complexities of big data, techniques such as incremental evaluation of a request, sampling and data reduction etc have been studied as part of data processing. It will be interesting to investigate how to incorporate such data processing techniques with a visualization system as it brings in another factor to be considered for giving a good experience for the analyst.

## ACKNOWLEDGMENTS

The authors would like to thank Dr. Halil Bisgin and other members of the database group at UM Flint for their valuable suggestions.

## REFERENCES

[1] D. Agrawal, P. Bernstein, E. Bertino, S. Davidson, U. Dayal, M. Franklin, J. Gehrke, L. Haas, A. Halevy, J. Han, H. V. Jagadish, A. Labrinidis, S. Madden, Y. Papakonstantinou, J. M. Patel, R. Ramakrishnan, K. Ross, C. Shahabi, D. Suciu, S. Vaithyanathan and J. Widom. *Challenges and Opportunities with Big Data âĂŞ A community white paper developed by leading researchers across the United States.* http://cra.org/ccc/docs/init/bigdatawhitepaper.pdf, Mar 2012.

[2] L. Battle, M. Stonebraker and R. Chang. IEEE Big Data Conference, Santa Clara, CA, 1 - 8, October 2013.

[3] M. Barnett, B. Chandramouli, R. DeLine, S. Drucker, D. Fisher, J. Goldstein, P. Morrison and J. Platt. *Stat! – An Interactive Analytics Environment for Big Data* ACM SIGMOD Conference, New York, NY, 1013-1016, June 2013.

[4] J. Dean and S. Ghemawat. *MapReduce: Simplified Data Processing on Large Clusters* OSDI, 2004, 137-150.

[5] S. Few. *Now You See It: Simple Visualization Techniques for Quantitative Analysis.* Analytics Press, 2009.

[6] D. Fisher, I. O. Popov, S. M. Drucker and M. C. Shraefel. *Trust Me, I'm Partially Right: Incremental Visualization Lets Analysts Explore Large Datasets Faster.* ACM SIGCHI Conference, Austin, TX, 1673-1682, May 2012.

[7] J. Heer and S. Kandel. *Interactive analysis of big data.* ACM Crossroads, 19(1): 50 - 54, 2012.

[8] D. Laney. *3D Data Management: Controlling Data Volume, Velocity and Variety.* Gartner, Feb 6, 2001.

[9] R. B. Miller. *Repose time in man-computer conversational transactions* Fall Joint Computer Conference, San Francisco, CA, 1968.

[10] Y. Park, M. Cafarella and B. Mozafari. *Visualization-Aware Sampling for Very Large Databases.* IEEE ICDE Conference, Helsinki, Finland, 755 - 766, May 2016.

[11] H. Piringer, C. Tominski, P. Muigg and W. Berger. *A Multi-Threading Architecture to Support Interactive Visual Exploration.* IEEE Transactions on Visualization and Computer Graphics, 15(6): 1113-1120, 2009.

[12] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker and I. Stoica. *Apache Spark: A Unified Engine for Big Data Processing.* ACM Communications, 59(11): 56 - 65, 2016.

[13] Apache Hadoop. http://hadoop.apache.org

[14] The R Project for Statistical Computing. https://www.r-project.org/

[15] Apache Spark. http://spark.apache.org

[16] Sample Data Sets from Tableau. https://public.tableau.com/en-us/s/resources

[17] TPC Benchmark H (Decision Support) http://www.tpc.org/tpch/