2016 International Conference on Collaboration Technologies and Systems

A New Workflow to Interact with and Visualize Big Data for Web Applications

Rui Wu* Jose T. Painumkal+ Nimrat Randhawa+ Lisa Palathingal*
Sage R. Hiibel~ Sergiu M. Dascalu* Frederick C. Harris, Jr*
*+Department of Computer Science and Engineering
University of Nevada
Reno, USA
~Department of Chemical & Materials and Engineering
University of Nevada
Reno, USA
*{rui, lisa, dascalus, fred.harris}@cse.unr.edu +{josepainumkal, nrandhawa}@nevada.unr.edu ~shiibel@unr.edu

Abstract*— Interaction and visualization are two significant methods for both business people and scientists to find "gold nuggets" buried in raw data. These two methods can simplify complex theories and make it easier for people from different research areas to cooperate. Many prevalent web-based data interaction and visualization tools and libraries are not as effective as before because of big data. Most of the traditional client/server web application visualization tools and libraries process visualization and interaction in the client side. This workflow requires the server side to transfer data to the client side. If the data size is very big, the data transferring time is unbearable. In this paper, we propose a fast and new method for client/server web application to interact and visualize big data. The method visualizes data in the server side with multiple CPU cores and transfers result images to the client side. The client side collects users' interaction information and the server side updates visualization results based on the interaction information. We tested the workflow with large volume datasets and it is much faster than traditional workflows.

Keywords: Big data interaction, big data visualization, web application, client/server

I. INTRODUCTION

Big data is not far from us. It is the digital era now and people generate data with amazing speed. Ninety per cent of all the data in our world was generated in the last a few years [1]. There are 300 petabytes of data stored in Facebook warehouses and the daily incoming rate is around 600 terabytes [2]. Around 300 hours of video materials are uploaded to YouTube every minute [3]. Interaction and visualization are two efficient and effective methods to discover the valuable rules, tendencies, and theories buried in raw big data. These findings can be treasures for both business people and scientists. Interaction and visualization can also make complex theories and rules easier to be understood. We can use these two methods to introduce ideas to our collaborators from other research areas clearly with less of time.

If users have large datasets, it is inconvenient and sometimes impossible to store all the data in one computer. The web-based client/server architecture is a good choice to interact and visualize large datasets. Also, users do not need to store the dataset in their machines (client side). There are many mature and prevalent traditional interaction & visualization tools and libraries for web application. However, they are not as effective as before because of the big data. As [4] points out, the most prevalent visualization tools and libraries can only be ran on the client side, such as Google Chart, Open Flash Chart (OFC), Adobe Flex, and D3.js. Also, only parts of the popular tools and libraries can process large datasets. Generally, the performance will drop dramatically when there are more than 200,000 data points. For Adobe Flex, the visualization results stop responding occasionally when there are more than 50,000 records. To solve this problem, we designed a new workflow to move as much work as possible to the server side to improve the performance.

Our research aims to help business people and scientists to interact and visualize their large volume data quickly and effectively. We proposed our new workflow and a prototype system in this paper. We did some experiments to compare the performance between our workflow and the traditional workflow systems. The results proved our workflow is better for big data interaction and visualization.

The rest of this paper is organized as follows: Section II introduces the background of big data and traditional web-based client/server applications workflow, Section III presents our proposed workflow, Section IV includes how we implemented our prototype system, Section V compares the performance of traditional workflow and our proposed workflow, Section VI concludes our ideas and introduces our future work.

II. BACKGROUND

The web-based client/server architecture is widely used. We prefer to use this architecture because it does not require users to install anything for most cases. Users only need to open a browser and visit our website. Also, this architecture can always guarantee a good performance, if it is designed reasonably (allocate most heavy tasks to servers). For example, some websites offer images processing services. Users can upload a bunch of images to the websites and choose the process methods. After the websites finish the jobs, it will send a notice email to the users and then the users can download the result images from the websites. Users can also process the images with their own computers. However, the procedure may be very slow if their computer is not good enough.

CSV (comma-separated values) is a file format, which is widely used in scientific researches. For example, some sensors output the collected data into a csv file. CSV files separate data with comma for most cases. Some csv files also contain metadata part, which introduce some information about the data, such as time format, copyright, and column header descriptions.

978-1-5090-2300-4/16 $31.00 © 2016 IEEE
DOI 10.1109/CTS.2016.61

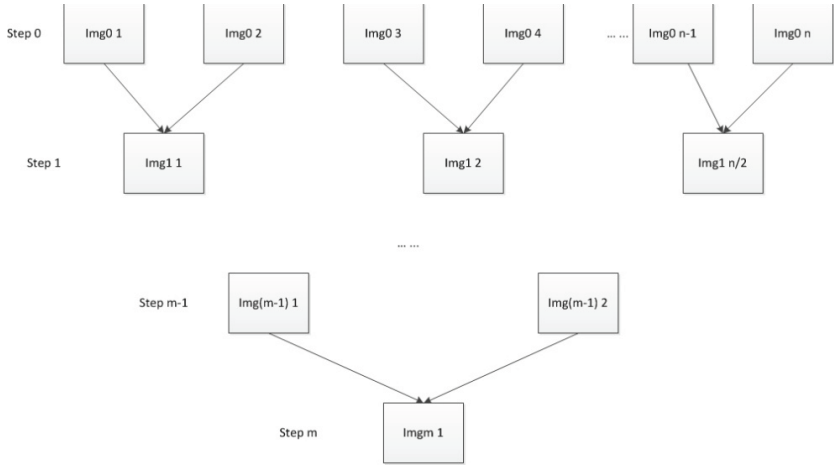

Figure 1. Image combination method

About big data, there are many definitions. For example, the company SAS defines big data as a popular term used to describe the exponential growth, availability and use of information, both structured and unstructured [5]. Most of the definitions are based on the three well-known big data features from [6]: large and growing volume (volume), fast input and output speed (velocity), different varieties of data (variety). Recently some companies, such as LinkedIn [7], added another two big data characteristics: messiness or trustworthiness of the data (veracity) and turn data into value or it is useless (value). Because of these characteristics, most of the traditional tools and libraries are not as effective as before. For example, some tools push all the tasks to the client machines. The client machines may work fine with small dataset. However, they can be very slow if the chosen dataset is large and the task is complex, which means bad user experiences.

Figure 2. Traditional workflow

Most traditional web-based client/server applications use the workflow as Figure 2 presents [4]. When the client side sends a visualization request to the server side, the server side will send data to the client side. All interaction and visualization operations are done in the client side. If we use this workflow to interact and visualize big data, there will be two problems: 1) data transfer time can be unacceptable. For example, if users have 1TB data and 10MB/s downloading speed, it will take around a half an hour to transfer data; 2) most traditional tools and libraries load data into the client machines' memory to guarantee the performance. This does not work for big data. For example, the chosen data is 1TB and this requires the client side machine to have 1TB memory spaces. This is impractical for most machines. We designed an improved workflow to move the interaction and visualization tasks to the server side that reduce the client side machines' burden. We introduce more details about this workflow in New Workflow section.

III. OUR PROPOSED NEW WORKFLOW

The basic idea about the new workflow is that it separates all interaction & visualization tasks into subtasks and arranges as many subtasks as possible in the server side. The most remarkable difference between the new workflow and the traditional workflow is that it transfers visualization result images from the server side to the client side instead of data. The server side updates the visualization result images based on the interaction information collected from the client side.

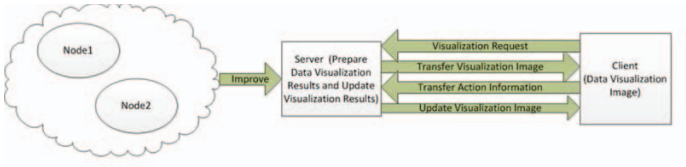

Figure 3. New workflow

Figure 3 displays the workflow details. The server side is a distributed system containing many nodes. Each node can be a computer or a CPU-core. The large volume dataset is separated into small pieces. Each node is in charge of a slice of data and generates a result image of the assigned data. One of the nodes combines all the result images as a final result image. The image combination method is shown in Figure 1. This can be done in parallel. In Step 0, we have n images in total. Each step we use one thread to combine two images. After all the threads combine images, the next step will be started. The image combination is done in parallel too.

After the server side generates the final result image, it will send the final result image to the client side. The client side collects users' interaction information (such as mouse coordinates, mouse button click, and mouse button release) and transfers the information back to the server side. The server side

will update the result images based on the users' interaction information.

The new workflow relies more on the server side machines and that results in the system performing more steadily. It is hard to predict machines situated on the client side. If we use traditional workflow and push most of the data interaction and visualization jobs to the client side, we may have different problems deal to big data. For example, the client side machine does not have advanced hardware and it performs more slowly to interact and visualize large datasets. In contrast to the traditional workflow, our proposed workflow uses distributed systems and process data in parallel. Therefore, the new workflow is always faster to deal with big data. Users do not need to have an advanced computer because the client side machine only needs to display result images and collect the user interaction information.

IV. PROTOTYPE SYSTEM

We created a prototype server with eight CPU cores in one computer and the server side can only visualize csv files with line charts for the current version. In this section, we introduce how to zoom in, zoom out, download the chosen part of data, and add/remove a csv column from the visualization result image.

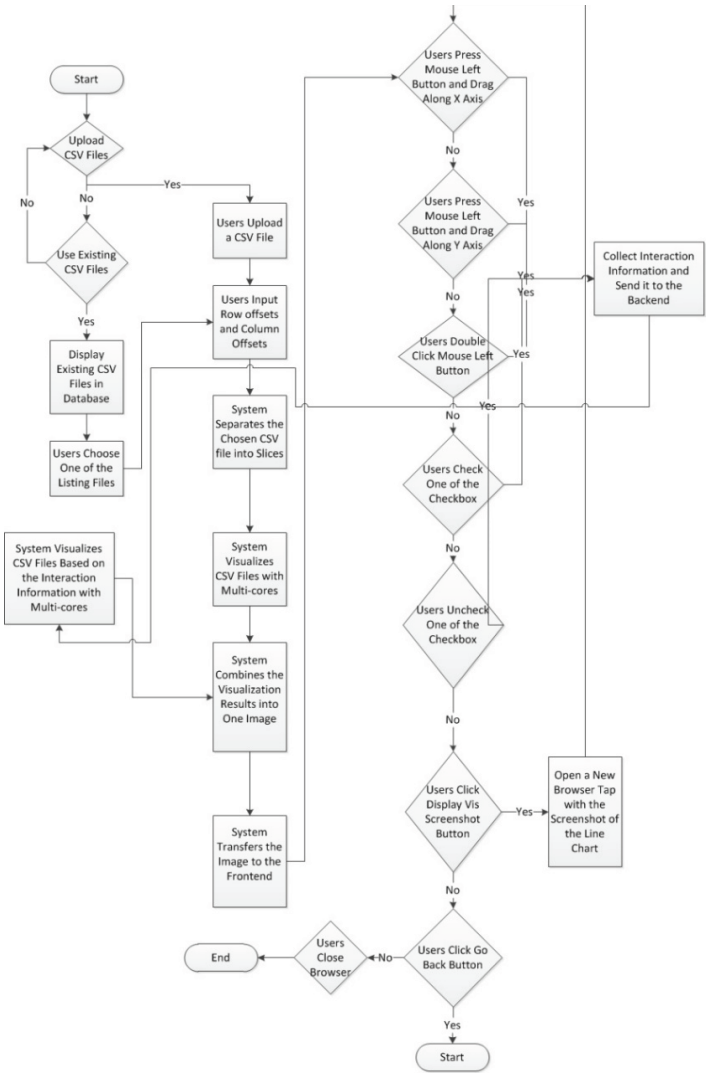

Figure 4. Prototype system new CSV visualization workflow

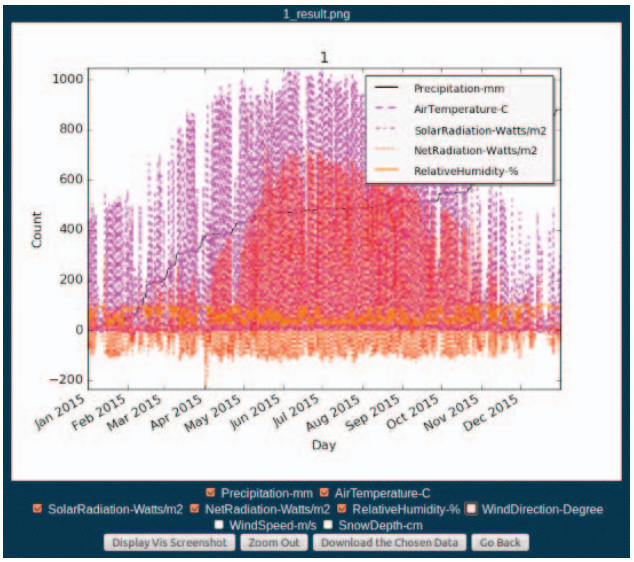

Figure 5. CSV file visualization example

In the new workflow, there are four basic steps to interact and visualize a file:

1. Users choose a file from the server database or upload a file to the server by themselves.
2. The server prepares the visualization result image and sends it to the client side.
3. Users interact with the result image and the client computer collects the interaction information.
4. The server side updates the visualization result image based on the interaction information.

Based on these four basic steps, we designed our prototype system. Figure 4 displays more details about our system. The system is event driven, which means the system flow is decided by different events. For example, users press the download button, the system will prepare the chosen data for users to download.

Figure 5 is a screenshot of a csv file visualization. The line chart part image is created by merging eight images shown in Figure 6. Because each CPU core generates a result image, we have eight images in total. The procedure is that one of the CPU-cores separate the chosen csv file into eight parts with the help of a Python library named Pandas [8]. Then each CPU-core visualizes a sub csv file with Matplotlib [11]. The result images are named: img0, img1, img2, …, img7. In the image merging stage, the system uses four CPU-cores as seen in Step One. CPU-core 0 is in charge of combining img0 and img1; CPU-core 1 is in charge of combining img2 and img3; …; CPU-core 3 is in charge of combining img6 and img7. Similarly, the final result image is generated after Step Four. This stage uses the method introduced in Figure 1.

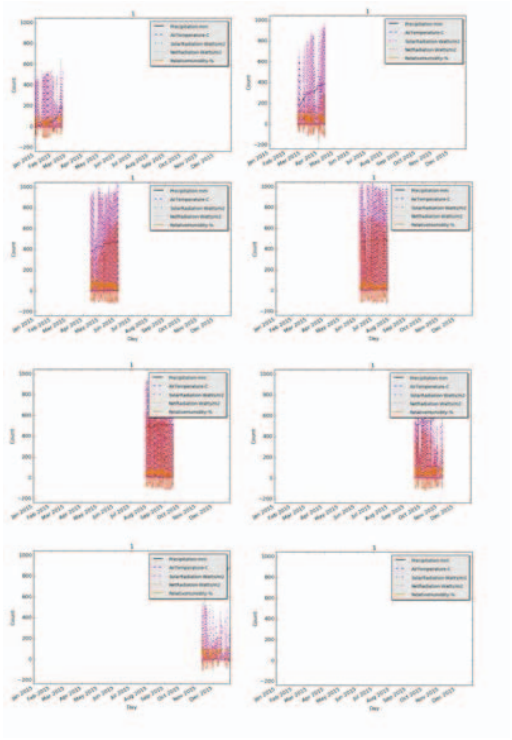

Figure 6. Visualization result images created by eight CPU cores

There are some checkboxes below the line chart part in Figure 5. Each of the checkboxes represent a column in the csv file. If the user checks one checkbox, the system will add the corresponding csv column in the line chart, which includes adding a line in the line chart and a matching legend in the right top corner of the resulting image. If the user wants to remove the corresponding csv column, they need to uncheck the checkbox, which includes removing the line from the line chart and the matching legend in the right top corner of the final result image.

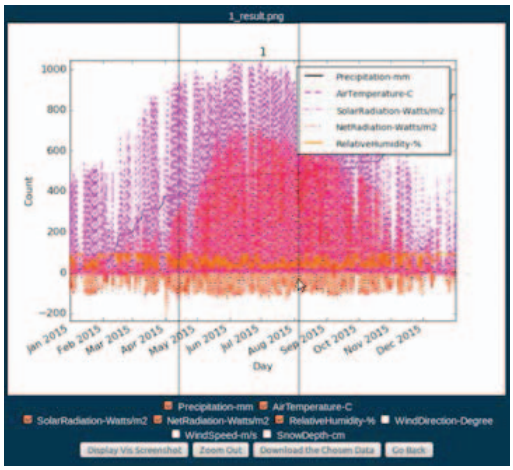

Figure 7. Users choose an area on the line chart

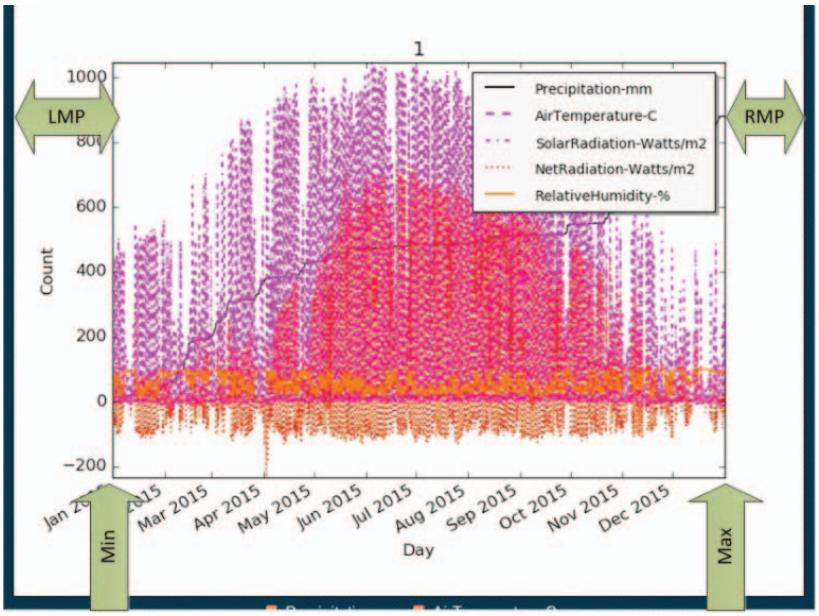

Figure 8. LMP, RMP, Min, and Max

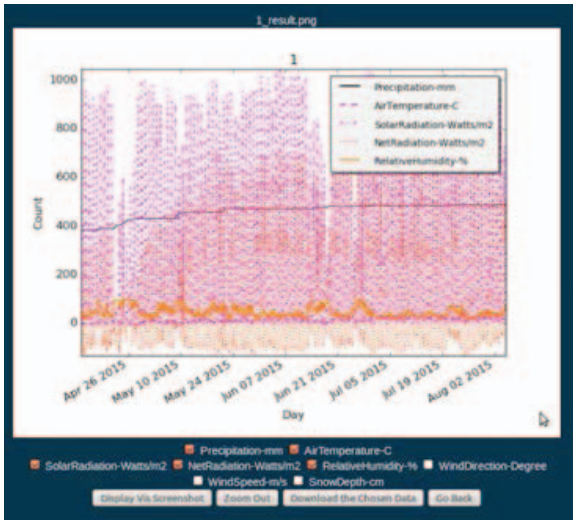

Figure 9. Zoomed in line chart

Users can zoom in the line chart by clicking the mouse left button, dragging along the line chart, and releasing the mouse left button. The system will draw a rectangle on the line chart as Figure 7 shows. After the client side receives the updated visualization result image from the server side, it will display the updated line chart image, as Figure 9 displays. If users want to zoom out on the line chart, they need to click the "Zoom out" button. We implemented the "Zoom in" function by tracking the coordinates of users' cursors and "mouse click" and "mouse release" events. Based on the information, the client side draws rectangles on the visualization result images and transfers the coordinates to the server side. We created an algorithm to convert the coordinates into x-axis values of the chosen csv files. The algorithm mainly has three steps:

1. Convert the cursor coordinates into percentages, which is named as CP. For example, when users click the middle part of the line chart, then CP should be 50% (from the image left side).
2. Get the left and right image margin percentage of the whole image. The left margin is named LMP and the right margin is named RMP. For example, the left margin is 10% width of the whole image and the right margin is 15% width of the whole image. Then LMP is 10% and RMP is 15%.
3. Obtain the corresponding x-axis values. The minimum value of the x-axis is named Min and the maximum value of the x-axis is named Max. Figure 8 is an example about LMP, RMP, Min, and Max. In this example, LMP is 15%, RMP is 10%, Min is January 1, 2015, and Max is January 1, 2016. Figure 10 is the algorithm that we used to convert CP into x-axis values.

Algorithm 1 Convert CP into x-axis values

Require: $CP \in [0,1]$, $LMP \in [0,1]$, $RMP \in [0,1]$, $Min < Max$

```
function ConvertCoorIntoXValue(CP, LMP, RMP, Min, Max)
    if CP < LMP then                ▷ The user's cursor is in the image left margin
        return Min
    else if CP > 1 − RMP then       ▷ The user's cursor is in the image right margin
        return Max
    else                            ▷ The user's cursor is in x-axis range
        return ((CP-LMP)*(Max-Min)/(1-LMP-RMP)) + Min
    end if
end function
```

Figure 10. Convert CP into x-axis values

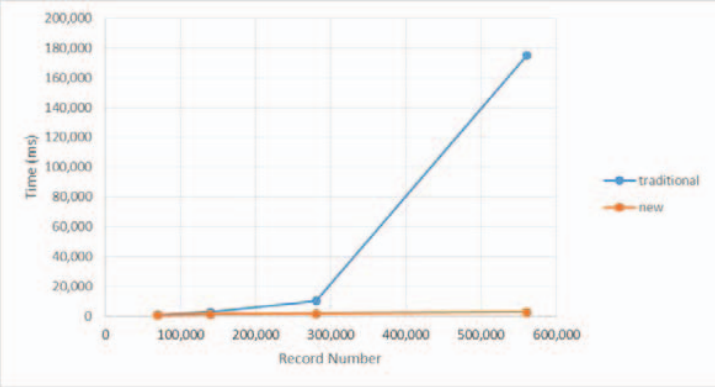

Figure 11. Traditional workflow and new workflow data visualization time consumption comparisons

The user's cursors may be placed in three sections: left margin, right margin, and in the x-axis range. If the cursor is in the left margin, then, as Figure 10 shows, the function will return Min as the x-axis value. If the cursor is in the right margin, then, as Figure 10 shows, the function will return Max as the x-axis value. If the cursor is in the x-axis range, then the corresponding x-axis value should be ((CP-LMP)*(Max-Min)/(1-LMP-RMP))+Min. This value is from the following equation:

$$\frac{CP - LMP}{x - Min} = \frac{1 - LMP - RMP}{Max - Min} \quad (1)$$

In Equation (1), x is the user's cursor x-axis value. (CP-LMP) means the percentage between the user's mouse cursor and Min; (x-Min) means the x-axis value between the user's cursor and Min; (1-LMP-RMP) means the x-axis range percentage; (Max-Min) means the x-axis range. The ratio between percentage and x-axis values should be the same.

Here is how we zoom in the chosen part data. The client side sends LMP, RMP, and CP information of "mouse click" and "mouse release" events to the server side. The server side stores Min and Max information and obtains the x-axis values of "mouse click" and "mouse release" events using the algorithm introduced in Figure 10. The server side extracts data based on the x-axis values from the csv file; stores the data in a temp file; separates the temp file into N (the node number in the server side) parts; each node of the server side processes one part; merges the process results; sends the final result image to the client side.

In the client side, we embedded visualization result images into "canvas" (an html element) in the client side. By doing this, we can avoid users from moving the images when they drag the cursors on the images.

We created an easy data extraction method. Users zoom in the wanted part on the line chart and click "Download the Chosen Data" button. The server side will provide the chosen part data as a csv file for users to download. We implemented this function because some of our collaborators are environmental scientists. Different servers received huge datasets and stored the data into the database. They expected us to create an effective and easy method to extract data from a large data file.

We used MongoDB for the data management. The basic idea is to store data files into a file system and the file paths in the MongoDB. If the system receives a query request, it will search the file path from the MongoDB based on the file name and then grab the file from the file system based on the file path.

For the parallel programming part, we used a Python library named Multi-progressing [9]. There are several reasons that we used processes instead of threads for our new workflow system [9]:

1. Processes and threads are both executed independently. Processes run in separate memory spaces and threads have a shared memory space. The parallel parts of our system do not require each core to communicate with each other and we do not want to have any writing memory conflicts problem.
2. Because of Python Global Interpreter Lock (GIL), which is a mutex (preventing multiple native threads from executing python codes at the same time), the performance can be worse than expectations.
3. Multi-processing python library will use all the processors simultaneously for computation.

Even if the current version prototype system can only visualize csv files with line charts, our proposed workflow works with most data files and visualization methods by using the four basic steps, which are mentioned in the second paragraph of this section.

V. PERFORMANCE COMPARISON

We did some experiments to prove that our workflow is faster than the traditional workflow. Here are our server and client machines hardware and operating system descriptions:

- 8 * Intel (R) Core (TM) i7-4770 CPU @ 3.4GHz
- 12.0 GB DDR3 RAM
- Ubuntu 12.04

For our new workflow, we used Flask [10] to build the backend and Matplotlib [11] to visualize data. Flask is a

powerful Python based microframework and Matplotlib is an effective Python based 2D data visualization tool. For the traditional workflow, we used Flask [8] to build the backend and dygraph.js [12] to interact and visualize data in the frontend. Digraph.js is a JavaScript library that is easy to use and efficient. There are more details about how we built the traditional workflow system in [13]. All the data files used in this paper are generated by a scientific model named Isnobal [14].

For all of the following experiments, the server and the client are installed in the same machine. Therefore, the data transfer time is very short for both the new and the traditional workflows. In real life, data transfer time is one of the most important factor that affects the user experience. For most of the big data cases, the traditional workflow costs more time than our new workflow. This is because the visualization result image is usually smaller than the visualized data itself. For example, 100 MB data can be visualized with a 5 MB jpg file. The image size is decided by resolutions and image formats. Therefore, the new workflow is faster than the traditional workflow in data transfer.

Figure 11 compares the traditional workflow and our workflow data visualization time consumptions. When there are less than 100,000 records (floats), the traditional workflow uses less time than the new one. This is because data size is small and data transfer time is short for the traditional workflow. The new workflow needs to separate files into small pieces, visualize each of them, and then merge all the visualization result images. These steps are not effective for small data files. However, when data size grows, the traditional workflow turns slower than the new workflow. Especially when we have more than 290,000 records (floats), the traditional workflow performance drops dramatically. In fact, when we tried a large file with more than 10,000,000 records, the traditional workflow ran for more than one hour and popped up an error.

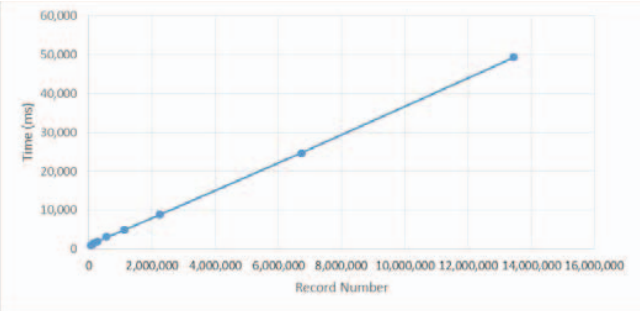

Figure 12. New workflow time consumptions

Figure 12 presents the new workflow time consumptions. It is almost linear which means the more data we have, the more time it will take. To improve the performance, we want to use Hadoop with more nodes.

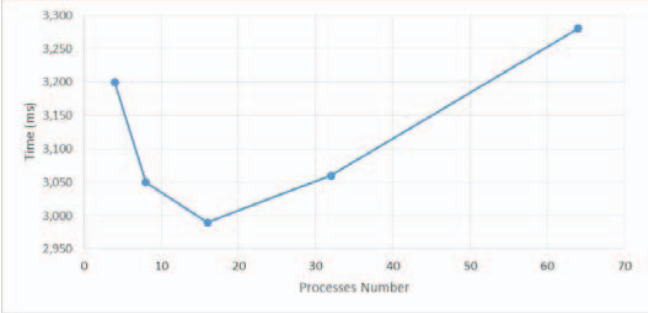

Figure 13. Visualize 560640 records with different number processes, when process number grows, performance goes up and then goes down

We also tested our system with different number of processes. Figure 13 shows that the time consumption goes down first and then goes up, which means when we used more processes, the performance of the system turns better first and then turns worse. The system performs best when we used 16 processes. This is because we used 8 * Intel (R) Core (TM) i7-4770 CPU @ 3.4GHz. The Intel core uses hyper-threading technique, which allows a computer's operating system or hypervisor to access two logical processors for each physical core [15]. Therefore, the 8 Intel core equals 16 logical processors.

VI. Conclusion and Future Work

In this paper, we proposed a new workflow to interact and visualize big data for web-based client/server applications. The basic idea is to visualize and update the visualization results in the server side and only transfer visualization result images to the client side. This is different from the traditional workflow in that it transfers data and finishes interaction and visualization tasks in the client side. We did some experiments that showed that the traditional workflow is better than the new workflow if users have small dataset (less than 100,000 floats). If users have a large dataset, the new workflow performs much better than the traditional workflow. When we used the traditional workflow to process very large amount of data, the performance will drop dramatically. The prototype can be found in [16].

In the future, we plan to use Hadoop with more nodes to improve the new workflow performance. Also, we want to use GPUs in each of the nodes to accelerate the server's performance further. Most of our collaborators are environmental scientists. They have to deal with csv and NetCDF files for most cases. Therefore, we will improve our system that support more scientific data file types, such as NetCDF.

Acknowledgment

This material is based upon work supported by the National Science Foundation under grant number IIA-1301726. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] IBM - What is big data? [Online]. Available: http://www-01.ibm.com/software/data/bigdata/what-is-big-data.html. [Accessed: 15- Sep- 2016].

[2] Scaling the Facebook data warehouse to 300 PB [Online]. Available: https://code.facebook.com/posts/229861827208629/scaling-the-facebook-data-warehouse-to-300-pb/. [Accessed: 15- Sep- 2016].

[3] Statistics – YouTube [Online]. Available: https://www.youtube.com/yt/press/statistics.html. [Accessed: 15- Sep- 2016].

[4] S. Lee, J. Y. Jo, and Y. Kim, "Performance testing of web-based data visualization", In *Proceedings of 2014 IEEE International Conference on Systems, Man and Cybernetics (SMC)* 2014, pp. 1648-1653.

[5] What is Big Data and why it matters [Online]. Available: http://www.sas.com/big-data/. [Accessed: 15- Sep- 2016].

[6] D. Laney, 3D Data Management: Controlling Data Volume, Velocity, and Variety. META Group Research Note 6: 70, February, 2001.

[7] Big Data: The 5 Vs Everyone Must Know [Online]. Available: https://www.linkedin.com/pulse/20140306073407-64875646-big-data-the-5-vs-everyone-must-know. [Accessed: 15- Sep- 2016].

[8] Python Data Analysis Library — pandas: Python Data Analysis Library [Online]. Available: http://pandas.pydata.org/. [Accessed: 15- Sep- 2016].

[9] D. Hellmann, Multi-Processing Techniques in Python. *Python Magazine*, vol. 1, number 10, October, 2007.

[10] Welcome | Flask (A Python Microframework) [Online]. Available: http://flask.pocoo.org/. [Accessed: 15- Sep- 2016].

[11] J. Hunter, "Matplotlib: A 2D Graphics Environment", *Computing in Science and Engg.*, vol. 9, no. 3, pp. 90-95, 2007.

[12] Dygraphs [Online]. Available: http://dygraphs.com/. [Accessed: 15- Sep- 2016].

[13] R. Wu, S. Dascalu, and F. Harris,"Environment for Datasets Processing and Visualization Using SciDB". In *Proceedings of the 24th International Conference on Software Engineering and Data Engineering (SEDE 2015)*, pp 226-229, October 12-14, 2015, San Diego, CA.

[14] D. Marks, J. Domingo, and J. Frew. Software tools for hydro-climatic modeling and analysis: Image processing workbench, ARS-USGS Version 2. *ARS Technical Bulletin 99*-1, Idaho, USA.

[15] D. Marr, "Hyper-Threading Technology in the Netburst® Microarchitecture" In *14th Hot Chips, Stanford*, CA, August, 2002.

[16] Prototype Github Repository [Online]. Available: https://github.com/ruiwu1990/big_data_visualization_interaction. [Accessed: 15- Sep- 2016].