High Performance Heterogeneous Computing for Collaborative Visual Analysis

Jianping Li* Jia-Kai Chou† Kwan-Liu Ma‡
University of California, Davis

Abstract

Visual analysis of large and complex data often requires multiple analysts with diverse expertise and different perspectives to collaborate in order to reveal hidden structures and gain insight in the data. While collaborative visualization allows multiple users to collectively work on the same analytic task, the user side computing devices can be used to share the computation workload for demanding data transformations and visualization calculations. In this paper, we present a heterogeneous computing framework for effective utilization of all the connected client devices to enhance the usability of online visualization applications.

Keywords: heterogeneous computing, collaborative visualization, information visualization, visual analytics, parallel data transformation

1 Introduction

Visualization is effective to extract useful information from large data. Collaborative visualization allows analysts to combine their effort to solve challenging problems [Isenberg et al. 2011]. Visual analysis of large and complex data often requires statistical methods to aggregate the data into meaningful overviews and visual summaries. For exploratory visual analysis, the user can interactively change the parameters of the underlying algorithms to generate different visualizations for discovering hidden structures and patterns in the data. Due to the large size of datasets and high complexity algorithms, the computations needed for data transformations and visualizations require high performance computing to minimize the wait time of the analysts.

While remote collaborative visualizations are done online and in a synchronous fashion, the users are performing analysis on a shared visualization using their own devices. Due to the recent advancements in personal and mobile computing, devices such as laptops and smartphones have multi-core processors and high bandwidth network connections, which can be leveraged to accelerate computations. In this paper, we present a heterogeneous computing framework for building web-based collaborative visualization tools that can utilize client devices to effectively perform data transformations and visualizations in parallel. To evaluate the applicability and performance, we use our framework to build a prototype system for collaborative visual analysis of massive time series. The test results show that our framework can effectively improve the performance of collaborative visualization systems.

*lij@cs.ucdavis.edu
†jkchou@ucdavis.edu
‡ma@cs.ucdavis.edu

2 Related Work

Our work is related to research in collaborative visualization and distributed computing for visual analytics applications.

2.1 Collaborative Visualization

Research on collaborative visualization has a long history and is gaining more momentum recently. The recent advancements in web technologies lead to more web-based tools are built for collaborative visualization. CoWebViz [Kaspar et al. 2013] is a clinical visualization tool which supports synchronous collaboration with real-time interactions, which it uses a central server to support multiple clients for viewing 3D stereoscopic volume data as 2D cross-section images on web browsers. ManyEyes [Viegas et al. 2007] is a social website for asynchronous collaborative visualization and data analysis with commenting and annotation tools. [Mouton et al. 2011] reviewed many collaborative visualization systems and examined the trends in related technologies. The review showed that the rapid and relentless advancement in web technologies and internet connections has made the web browser an ideal platform for building collaborative visualization applications.

2.2 Distributed and Parallel Computing

Many work have been dedicated to improve the performance of visualization systems using distributed computing. [Chan et al. 2008] built a system for analyzing massive time series data by distributing data processing to a cluster of database servers and use predictive caching mechanism to hide system latency. CBRAIN [Sherif et al. 2014] uses grid systems to distribute data and computations for supporting collaborations on neuroimaging visualization. While these approaches can scale to large data, they would require expensive investment for installing and maintaining multiple dedicated servers. As an alternative, low-cost solution, we apply the idea of crowd computing [Fernando et al. 2012] to collaborative visualization. By combining the computing resources of the server and multiple client devices, we leverage heterogeneous parallel computing to improve the overall system performance.

3 Methods and Technologies

Our framework is aimed to provide high performance computing to web-based collaborative visualization applications that are computation-intensive. By leveraging standard modern web technologies, users can easily get connected with their collaborators using web browsers without any client side software or plug-in. By utilizing user devices to accelerate computations, our framework can effectively improve system performance when multiple users are connected to explore and analyze the same dataset.

3.1 Architecture

In our framework, the server node manages all the status and peer-to-peer(P2P) connections of the client nodes. When a new client is connected, the server distributes the data and creates P2P connections from all the existing client nodes to the new client node.

For reducing the overhead of HTTP communication, WebSocket[1] is used for faster server-client communications and efficient distributions of data. To leverage all the computing power of the personal and mobile computing devices in the client side, WebWorkers[2] is used for multi-threading and parallel computing. In addition, a newly standardized web technology, WebRTC[3], is used for client-to-client communications between the web browsers. Without WebRTC, the server needs to handle every network communication, which results in higher latency for client-to-client message passings and data transfers. Therefore, enabling direct communications between clients can greatly reduce the communication overhead. Figure 1 shows how the server and clients are connected to form a cluster of heterogeneous systems for accelerating the computations for a collaborative visualization application. The application server is built using NodeJS, a server-side JavaScript framework for building server side applications. By using the same language in both the server and client sides, the server and the client nodes can share a computing task together seamlessly. For rendering, HTML5 SVG[4] is used for generating visualizations on the web browser. In the process of distributed parallel computing, the results in the form of SVG elements are exchanged between the clients nodes, and the final visualization is assembled from the SVG elements.

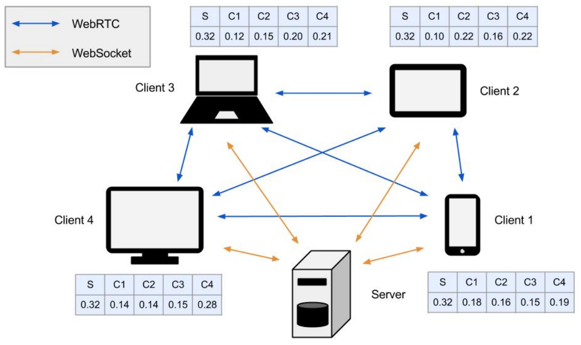


Figure 1: *The server and the client nodes are connected to form a heterogeneous cluster to accelerate data transformations and visualizations.*

3.2 Distribution of Computations

To achieve high performance computing with a heterogeneous cluster, a gating factor is to properly distribute the computations to balance the running time of each node. The nodes with more computing power and higher network bandwidth should perform more computations. [Beaumont et al. 2001] provide a simple algorithm for ensuring optimal distribution of n independent units of computation to p heterogeneous processors. For processor P_i with speed s_i, the algorithm finds the optimal distribution, n_i, by first distributes n units of computations evenly to each processor and then iteratively distributes one of the remaining units to the processor k, where $\frac{n_k+1}{s_k}$ has the minimum value for $1 \leq k \leq p$. To use this algorithm for distributing computations to multiple client nodes, we need to know the computing speed of each client device. However, hardware information of the client devices, such as the speed of the processor, is unknown to web applications due to the limitation of the browser. In addition, the heterogeneity in the software stack and network connections makes acquiring the actual computing speed of each client device nontrivial.

[1]http://www.w3.org/TR/websockets
[2]http://www.w3.org/TR/workers
[3]http://www.w3.org/TR/webrtc
[4]http://www.w3.org/TR/SVG

In our framework, we use a test program to estimate the computing speed of each client device. As a client is connected to the server, the test program distributes an equal amount of computations to each client, and the client devices perform the computations with the optimal number of threads according to the number of processor cores. Based on the completion time of each client, the relative computing speeds including communication overhead can be estimated. Because each client can initiate computations to the cluster, each client node has a different estimation of the relative speeds. For example, the client node that requests the computation would have zero communication overhead, so it should be responsible for relatively more workload while the remote nodes should receive relatively less workload due to the network latency. As illustrated in Figure 1, every node in the heterogeneous cluster uses its own estimation of relative computing speeds to distribute computations. With s'_i denoting the normalized relative computing speeds, Algorithm 1 is used to distribute computations to the server and client devices in our framework.

```
$n_i = \lfloor s'_i \times n \rfloor$, for $1 \leq i \leq p$;
while ($\sum_{i=1}^{p} n_i < n$) do
    if $\frac{n_k+1}{s'_k} = \min_{i=1}^{p} \frac{n_i+1}{s'_i}$ then
        $n_k = n_k + 1$;
    end
end
```

Algorithm 1: Algorithm to determine the optimal distribution of computations to each node in a heterogeneous cluster.

Our approach not only optimizes the system response time for the client node that initiates the computation, which is most sensitive to latency, but also improves the synchronization performance for all client nodes. Comparing to the conventional approach where all results are first sent back to be merged in the server and then broadcast to all clients, our framework leverages WebRTC to enable effective P2P data transfers, so the results can be efficiently exchanged among the client nodes without the need of round trips to the server for synchronizations.

3.3 Data Partition and Distribution

To use the client devices for accelerating computations, partitions of the raw data need to be distributed to the clients when they connect to the server. Ideally, each client should only receive the part of data that is required for its own computation based on the computing speeds and the relationship between the computation complexity and the data size. Because the relative computing speed ratios are different at each client node, the server needs to check all the relative computing speed ratios and properly distribute the needed portion of data to each client. In addition, dynamically distributing the data partitions to different client nodes is challenging as the clients may connect or disconnect at any time. We can make the distribution of data to be run as a background task. When a client node connects to the server, it cannot be used for computing before the server completes the rearrangement of the data distribution, while the new client can still see the visualizations and participate in the collaborative analysis. When a client node disconnects, the server needs to temporarily take over the workload of the disconnected node until the rearrangement of the data distribution is completed. To simplify the implementation for our preliminary study, currently we distribute the whole dataset to every client node when the client connects to the server for the first time. Dynamic and optimal distributions of data will be studied in our future work.

4 Evaluation

To experimentally study our framework design, we have built a prototype system for collaborative visual analysis of massive time series data. Time series data are commonly found in many applications such as finance, healthcare, and movement tracking. The test data we have used is compose of electronic health records. Each item in the time series represents the health history of a person over a twenty year period with a 30-day time interval. Each data point contains logs of the diseases and medications for the corresponding time interval.

4.1 Parallel Transformation and Visualization of Massive Time Series

When analyzing time series data, data operations such as segmenting, aligning, filtering and clustering are needed to extract trends, correlations, and causalities from the data. To visualize the data, view transformations are performed to encode and map the aggregated data into visual structures. In our study, we use Sankey diagram to visualize trends of clusters and patterns of transitions. With x-axis representing the time, each node(i.e. the vertical bar) in the Sankey diagram represents a cluster of the time series at a specific discrete time, and the curve edge shows how the time series move from one cluster to another over time. Figure 2 shows how our system distributes the time series data and the associated computations to the server and the connected client devices. Because the edges in the Sankey diagram are computed by finding the difference between the clusters at a previous time and the clusters at a latter time, the data and computations distributed to the computing devices are overlapped by one time interval in order to compute all the edges in the Sankey diagram.

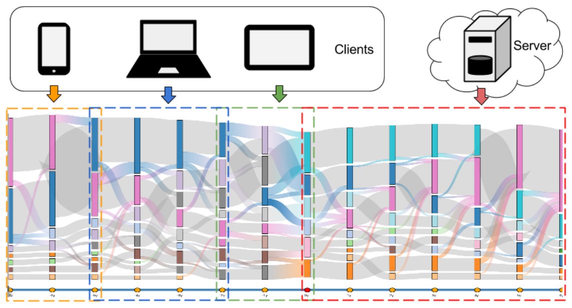


Figure 2: *The computations for data transformations and visualization calculations are distributed to the server and client devices according to their relative computing speeds.*

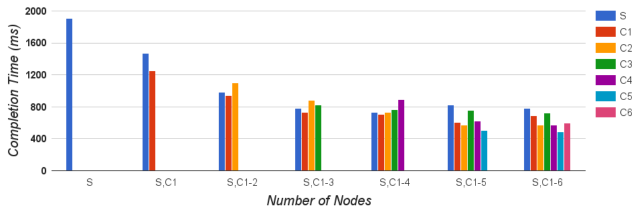
4.2 Performance

For evaluating performance, we test how our system performs as we increase the number of client nodes and the size of the dataset. The devices that we used in this performance test are listed in Table 1. The web browser used in all the devices is Chrome version 42. Figure 3 (a) and (b) show the completion time of each client for 300K and 750K number of records, respectively. The completion time is measured from the time when the first client node (C1) requests a new computation task to the time when it receives all the results from other nodes. The test results show that our method for distributing the computation is effective in balancing the computing time on each node in a heterogeneous cluster. Since the system performance depends on the worst completion time of the client nodes, balancing the computation time is important to achieving the best performance with parallel computing using heterogeneous devices. In addition, larger problem sizes would have more performance gain due to the overhead of communications between the client nodes. As we note in Figure 3 (a) and (b), the performance gain is significantly larger for the data size of 750K records than the data size of 300K records.

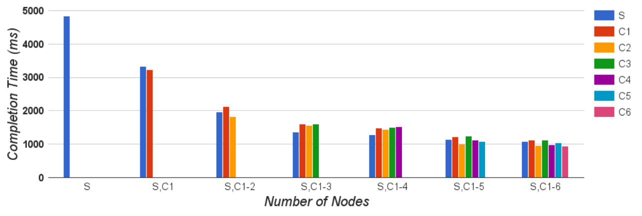
Figure 4 shows the system performance for different data sizes with different number of client nodes used in computations. As the test results show, the completion time for transforming and visualizing that data can be significantly reduced as more client nodes are used for computations. The speedup depends on the number of client nodes and their computing power. Note that C4 is a smartphone, and we can see that it only provides a very small performance gain in test case of S,C1-4. If we compare the performance for the case which only the server was used for computing versus the case which the server and 3 client nodes(C1,2,3) are all used for computing, the results show that the speedup is more than 2X for the data size of 600K records and about 3X for 1M records. As we test with more client nodes, the performance gain is only noticeable for larger data size. For the test cases with 5 and 6 client nodes, the speedup is 3.1X and 3.2X, respectively, for 600K records, and the speedup is 3.4X and 3.9X, respectively, for 1M records.

Node	Type	Processor	Speed	cores / threads
S	server	AMD FX-8320	3.2 GHz	8 / 8
C1	laptop	Intel Core i5	1.4 GHz	2 / 4
C2	desktop	Intel Core i7	3.6 GHz	4 / 8
C3	desktop	Intel Core i7	2.4 GHz	4 / 8
C4	phone	Qualcomm Snapdragon	2.5 GHz	4 / 4
C5	desktop	Intel Core i7	3.5 GHz	4 / 8
C6	desktop	Intel Core i7	4.0 GHz	4 / 8

Table 1: *A list of devices used for performance testing.*



(a) *300K records*



(b) *750K records*

Figure 3: *Completion time of each client when different number of client nodes are involved in the computation.*

5 Discussion

Our preliminary study allows us to identify both limitations of our current implementation and opportunities for further work.

5.1 Limitations

Our approach may only suitable for applications with moderate data size and computational requirements that can be handled by a small

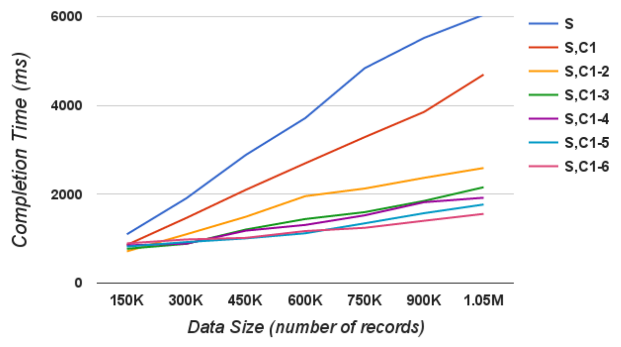


Figure 4: *Performance evaluation of increasing data size for different number of client nodes.*

number of personal computing devices. For larger data size and more intensive computation, server side scaling is required. When there are multiple servers, our approach still can be used to combine the computing power of multiple servers and client devices to accelerate computations. For problem sizes that can be handled efficiently by a small number of personal computing devices, all demanding computations can be completely offloaded to the client nodes without consuming any computing resources of the server, which allows the server to host multiple services and support multiple groups of users.

A consideration with our approach is that the computing resources and network bandwidths of the client nodes may change at any time due to other user activities. This makes the initial estimation of computing speeds inaccurate and may result in poor system performance due to uneven completion time among the client nodes. To minimize the negative effect of this problem, we can periodically adjust the workloads of the client nodes using the new completion time.

5.2 Future Work

In future work, we plan to use our framework in more collaborative visualization applications. We will also look for ways to improve performance. Currently, we only use the CPU on the client nodes for computing. By using WebGL or WebCL for data processing, web application can also use the GPU on the client node for general purpose computing [Liu et al. 2013]. We can extend our framework to not only use the CPU but also the GPU on the client nodes for parallel data transformation and visualization.

For dynamic data distribution as we discussed in Section 3.3, we will conduct further research to optimally distribute data to the client devices as the clients may join or leave the cluster at any time. Each client node should only receive the part of data that is needed to perform its own computation. Dynamically and optimally distributing data would reduce the network traffics of the server for transferring data.

Another future step is to support collaborative systems with many users analyzing different data sets. This requires an effective task scheduling to manage multiple computation tasks requested by different users simultaneously. With this extension, it would be possible to use our framework for supporting data analysis and visualization in social websites where many users are exploring different data sets [Viegas et al. 2007].

6 Conclusion

High performance computing is crucial to an interactive visualization system to enable effective data exploration. While traditional collaborative visualization tools rely on centralized servers for processing complex computation and neglect the potential use of client devices for computation acceleration, we contribute a web based framework for high performance heterogeneous computing through effective utilization of capable personal computing devices. We have built a prototype system using our framework for visual collaborative analysis of massive time series data. The test results show that our framework can effectively improve system performance and reduce user wait time on computation intensive operations. As more collaborative tools will be web based in the future, our framework is expected to help improve the performance and enhance the usability of collaborative visualization applications.

Acknowledgements

This research is supported in part by the National Science Foundation via grants NSF IIS-1320229 and Department of Energy via grant DE-FC02-12ER26072.

References

BEAUMONT, O., BOUDET, V., PETITET, A., RASTELLO, F., AND ROBERT, Y. 2001. A proposal for a heterogeneous cluster scalapack (dense linear solvers). *Computers, IEEE Transactions on 50*, 10, 1052–1070.

CHAN, S.-M., XIAO, L., GERTH, J., AND HANRAHAN, P. 2008. Maintaining interactivity while exploring massive time series. In *Visual Analytics Science and Technology, 2008. VAST'08. IEEE Symposium on*, IEEE, 59–66.

FERNANDO, N., LOKE, S. W., AND RAHAYU, W. 2012. Mobile crowd computing with work stealing. In *Network-Based Information Systems (NBiS), 2012 15th International Conference on*, IEEE, 660–665.

ISENBERG, P., ELMQVIST, N., SCHOLTZ, J., CERNEA, D., MA, K.-L., AND HAGEN, H. 2011. Collaborative visualization: definition, challenges, and research agenda. *Information Visualization 10*, 4, 310–326.

KASPAR, M., PARSAD, N. M., AND SILVERSTEIN, J. C. 2013. An optimized web-based approach for collaborative stereoscopic medical visualization. *Journal of the American Medical Informatics Association 20*, 3, 535–543.

LIU, Z., JIANG, B., AND HEER, J. 2013. immens: Real-time visual querying of big data. In *Computer Graphics Forum*, vol. 32, Wiley Online Library, 421–430.

MOUTON, C., SONS, K., AND GRIMSTEAD, I. 2011. Collaborative visualization: current systems and future trends. In *Proceedings of the 16th International Conference on 3D Web Technology*, ACM, 101–110.

SHERIF, T., RIOUX, P., ROUSSEAU, M.-E., KASSIS, N., BECK, N., ADALAT, R., DAS, S., GLATARD, T., AND EVANS, A. C. 2014. Cbrain: a web-based, distributed computing platform for collaborative neuroimaging research. *Frontiers in neuroinformatics 8*.

VIEGAS, F. B., WATTENBERG, M., VAN HAM, F., KRISS, J., AND MCKEON, M. 2007. Manyeyes: a site for visualization at internet scale. *Visualization and Computer Graphics, IEEE Transactions on 13*, 6, 1121–1128.