Project Report of Information Visualization for Configuration of Distributed Systems

Kewen Wang
Computer Science and Engineering Department
University of Connecticut
Storrs, USA
Email: kewen.wang@uconn.edu

***Abstract* — Tuning configuration of distributed systems provides an approach for system performance improvement. But distributed systems have large amount of configuration. For instance, Apache Cassandra has 107 configuration parameters, how to tune these configurations is not an easy task. In this paper, we design a web-based tool to present direct and hidden information for configurations and the programs of the distributed platform, to give some hints to users for configuration tuning and related trouble shooting issues. This tool takes advantage of web services to visualize the extracted text data in the web pages, and applies JavaScript to provide animation effect for user interaction. Moreover, it utilizes layering mechanism to shrink the information scale and uses dividing technique to present specified information according to user focus. And we design flexible web framework for information display, which could accommodate to various configuration sets in distinct distributed platform.**

Keywords- Configuration, Cassandra, Visualization, WebService

I. INTRODUCTION

For distributed system, there are many configuration parameters that can affect the execution of the program in the platform. Tuning these configurations could solve the configuration-related system problem or improve the system performance [1]. However, there are many parameters for distributed system platform, and these configurations have complex correlation with each other. For configuration of distributed system platform such as Apache Cassandra [2, 3], there are 107 parameters, and about one thousand source codes files. Facing large amount of configurations, clearly presenting all the information to users is not an easy task. Moreover, it is even harder to find out how these configurations are used in the programs.

In this paper, we design a tool to visualize the configuration information and display these detailed information for system configuration. This tool applies web service to visualize configuration information from various aspects, and to provide animation effect through using JavaScript. To improve the display effect of this information, this tool uses an open source JavaScript library: JavaScript InfoVis Toolkit [4] to provide better animation effect for user to get desired information for configuration, and better interfaces for information exploration.

This web-based tool extracts the important configuration information through static source codes analysis. This information includes basic information and indirect information. Basic information provides brief background information for this configuration to help users have a basic knowledge about the configurations. Indirect information is extracted from the source codes of the system platform, and is about the connection information between the configuration and source codes files.

For this tool, it uses two mechanisms: layering and dividing to achieve neat visualization effect. It applies layering mechanism to classify the configurations into different sets and groups to shrink the information scale of each level. And it provides two perspectives to display the information of specified configuration parameter, or the related configuration information for selected source codes file. Dividing is used to display all the related information for specified configuration or source codes file. This divided information is easy to be organized, and could be presented in flexible layout through flexible web framework design. These two mechanisms combined could speed up the display of the configuration information.

The rest of the paper is organized as follows. Section II introduces related work. Section III presents the overview of this tool. Section IV describes the preprocessing stage for the original data. Section V and Section VI explains two mechanisms applied in this tool. Section VII evaluates the performance of this tool. Finally this paper is concluded in Section VIII.

II. Related Work

Information Visualization (InfoVis) aims to aid users in exploring, understanding, and analyzing data through progressive, iterative visual exploration [5]. InfoVis first collects the original data, and transfers this data into structured data through analysis, and then filter out focus data, then maps it to geometric data which could be further transformed into image data through rendering. And user can interact with the image data through some interfaces to explore and understand the data information from multiple perspectives.

The related research has different methodologies such as modeling and interaction. Modeling is important to design effective visualization. Visual representation models are introduced to handle different perception problems such as the uncertainty problem [6]. The data-driven models are applied to visualize various data such as high-dimensional data [7] and geographic data [8]. Besides, some researchers focus on the advanced interaction like visual comparison [9] and interest-driven navigation [10]. For different data and specific applications, there are specialized techniques for visualization. Edge bundling [11] technique is applied in graph visualization to show the European follower graph for GitHub. Whisper [12] uses locations of graphical elements to reflect the geographic and time attributes of documents. BirdVis [13] displays spatiotemporal distribution of birds by combining choropleth maps with different visual components.

Although there is much previous work in this research field, few of them have focused on the information visualization for distributed system configuration.

III. Overview of System

Configuration of distributed system has huge problem scale for the number of configuration parameters and their various effects on programs. In this paper, we design a tool to visualize the configuration of Apache Cassandra. This tool utilizes web services to implement the information display for configuration, and designs interfaces for user interaction. As showed in Fig.1, this tool preprocesses the source codes and configuration files to obtain configuration information including basic and indirect information. This preprocessing step is finished off-line, and no execution of programs is needed because of its static analysis feature. The further mining and analyzing is implemented in JavaServlet to map the extracted configuration information to structured data from different perspectives. Finally, the resulting data in the format of JSON data will be displayed in html web pages by using jQuery [14].

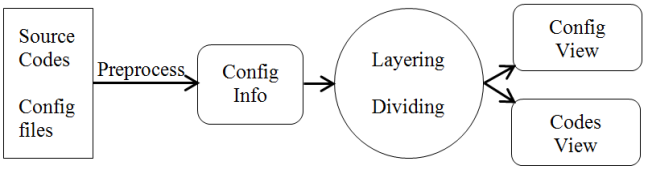

Figure 1. System Overview

During the display of these data, we apply layering mechanism to view this information from different aspects: configuration view and source codes view, and to classify the configurations into different levels and groups to narrow down the information scale of each level. And we apply dividing mechanism to present the full information for specified configuration or source codes file, which can be explored from configuration view and source codes view. These two mechanisms combined could speed up the display of complex information and provide clearer views of them.

IV. Preprocessing

In this preprocessing stage, the source codes and configuration files are preprocessed through static analysis method to obtain configuration information, which includes basic and indirect information.

The basic information is extracted by searching through the configuration files and related source codes file that implements the java object construction for these configuration parameters. The basic information includes configurations list, data type, default values in the configuration files, and other information about which of them are used in the programs in default configuration setting. Corresponding results are saved as text files.

Indirect information is obtained by analyzing the call graph of the programs for the source codes. Actually, all these configurations are encapsulated in a configuration object, and other functions or classes in the programs call these configurations by using corresponding getting methods for the configuration object. This stage of analysis finds out which functions or classes call the getting methods to use the configuration parameters, this is the connection information between configurations and the involved programs. Moreover, the location (line numbers) of the classes or functions where the configuration getting method is called is also highlighted for the convenience of further display of the information. This information is also extracted out as text files for next step analysis.

V. LAYERING

For the configuration of Apache Cassandra, it contains 107 parameters. For source codes file f_1 and configuration parameter A, f_1 is assumed to have used configuration A and A is classified as program configuration if the getting method of A occurs in file f_1. If one codes file f_1 has used two configuration parameters A and B, we assume parameters A and B are connected through f_1, and parameters A and B are classified as connected program configurations. Based on the assumption, Apache Cassandra has 74 program configurations and 68 of them are connected. To make the overview and the display clearer, we design two views for the information display: configurations view, and the source codes view.

For the configuration view as showed in Fig.2, we show the configurations list and classify them into two sets: program configurations that are used by source codes, and system configurations that are not used in the programs. And we design graphs to display the connection between configurations. For these 68 connected program configurations, it will be a mess to show all the connections in one graph. We design grouping method to divide these configurations into several groups according to their connection relation. For example, configuration A and B are used in one source codes file but are not used in files that has used configuration C, then A and B will be clustered in one group while C in another one. The original graph could be mapped to an abstract graph consisting of several groups while each group is represented by a node.

We use an open source JavaScript library: JavaScript InfoVis to display the configuration information in graph and make some modifications to enhance the animation effect. As showed in Fig. 3, after clicking on the group node 05 in the graph, this node will expand and two nodes of group 05 will be displayed in the form of a small graph while each node represents a configuration parameter. For better display effect, nodes and lines in each small graph of one group are painted with same color, different from the colors of other groups. It will help user have clearer overview about the graph.

In the configurations list as showed in Fig. 4, we also paint the name of the configurations in different colors according to their different groups, and these colors are corresponding to the color in the graph. For the configuration list, we add a function to sort all these configurations in the order of group as showed in the left of Fig. 4. This provides a convenient method for user to look up the configurations in the list and in the graph at the same time.

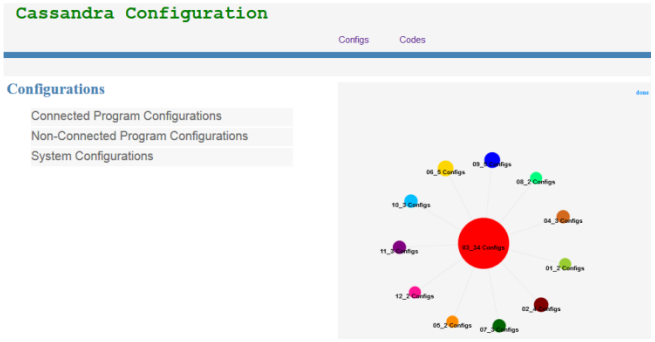

Figure 2. Configuration View

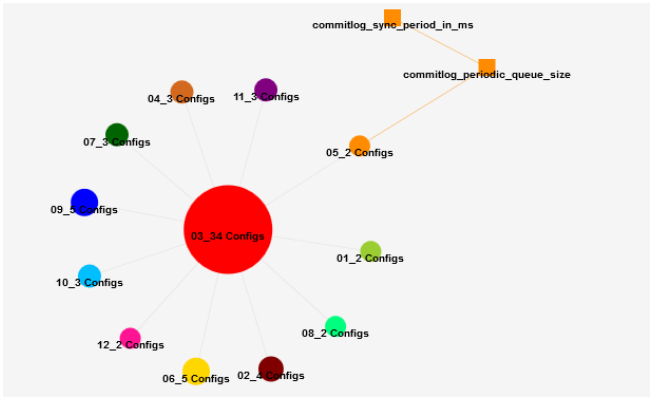

Figure 3. Group Expanding

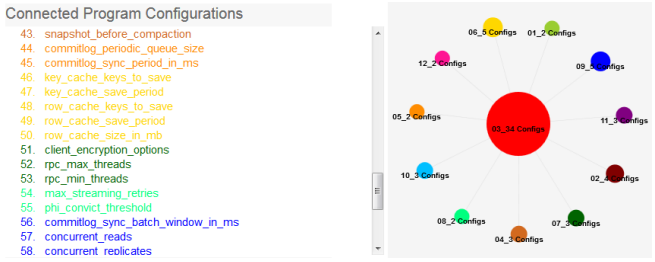

Figure 4. Sorting by Group

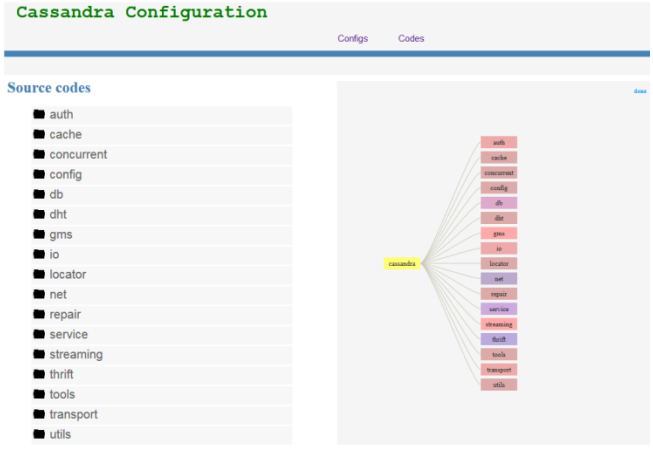

Figure 5. Source Codes View

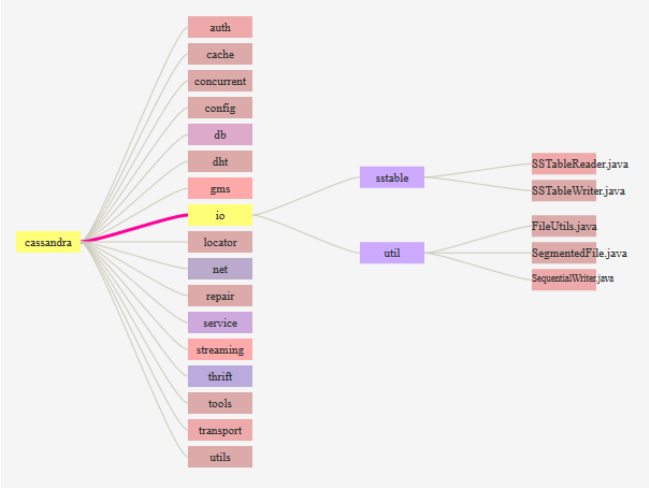

Figure 6. Source Codes File Tree

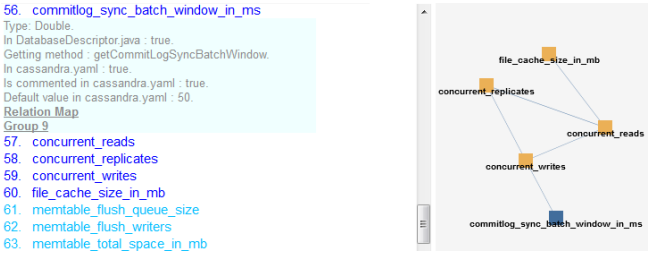

Figure 7. Configuration Information

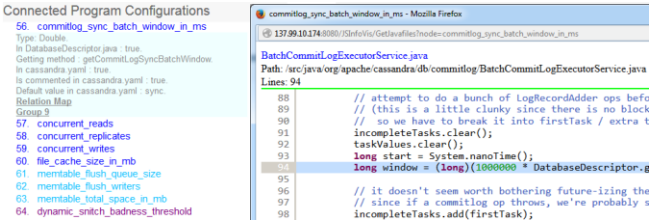

Figure 8. Source Codes

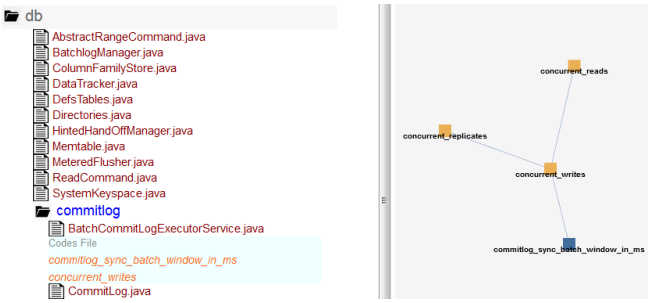

Figure 9. Connection Graph of Configurations

Moreover, layering mechanism is also applied in the source codes view as showed in Fig. 5. For those programs that have used the configurations, we arrange them according to their original position in the source codes folder. This list is similar to the file lists of the common file system, folder followed by sub-folders or files. Furthermore, we apply the JavaScript library to display this layering relation between these codes files in the form of tree in which each folder and codes file is represented as one node showed in Fig. 6. By clicking on the node, the sub-tree of this node will expand to display sub-folders or the codes files.

These layering methods: classifying configurations into different sets and subsets, grouping them into different groups according to their connection features in the programs, and source codes file tree could narrow down the scale of the configuration information at one level, and finally show this information in a legible scale.

VI. DIVIDING

To obtain the information of specified configuration or source codes file is essential for understanding this configuration and its related programs. This tool applies dividing mechanism to focus on the information of specified configuration or codes file.

In the configuration view, the basic information is displayed below each configuration by clicking the configuration name as showed in Fig. 7. This basic information includes the getting method for this configuration in file "DatabaseDescriptor.java" which encapsulates the configurations as an object, whether this configuration is used in the default configuration file "Cassandra.yaml", and the default value in this file. Moreover, it also provides a link "Group 9" to display the graph of the group where the specified configuration exists. For one configuration, it may be used by different source codes files, which may use other configurations as well, so one configuration may have several configurations connected to it. This connection information is also displayed in the connection graph of specified configuration by clicking the link "Relation Map". Furthermore, the list of the codes files that uses the configuration will be showed in a new window by clicking on the configuration node in the graph. As showed in Fig. 8, the source codes of the listed codes files are presented in coding style by using a JavaScript tool: SyntaxHighlighter [15]. Moreover, the location (line numbers) where the configuration is used in the codes file is highlighted.

Dividing mechanism is also applied in the source codes view. As showed in Fig. 9, it lists the configurations that are used in the source codes file "BatchCommitLogExecutorService.java". And the configurations are also listed in tree view as showed in Fig. 10. Clicking the configuration name listed, the connection information of the configuration is also provided through showing the connection graph. And

the source codes will also pop up by clicking the "Codes File" link. Moreover, this tool applies JavaScript to provide flexible web page framework to accommodate different kinds of configuration sets in different distributed system platform. Thus it can be applied for different configuration sets of different platforms by using one set of web pages.

VII. EVALUATION

This tool uses a layering mechanism to display the complex connection information for these configurations. It can narrow down the scale of the configuration information. For configuration view, this tool provides an abstract graph consisting of different connection groups. This abstract graph as showed in the left of Fig. 11 provides a clearer overview of this connection information of these configurations compared to the original connection graph for those program configurations as showed in the right of Fig. 11. Moreover, the speed for displaying the abstract graph has about 5 times speed-up over the display of the original graph which contains 10 times more node and line items than the abstract graph. For the source codes view, it provides the partial view of the codes files, and the further information could be displayed through user interaction. It provides a cleaner layout at the first glance.

This tool also uses a dividing mechanism, which displays the information for specified configuration. For one configuration, the related information is much small compared to the whole information, it is more efficient to show this information for specified configuration. From the source codes view, it is also more efficient to list the related configuration information for one specified codes file.

Moreover, the coverage rate is another metric for the evaluation. This tool provides basic and indirect information for every configuration in the platform, which means 100% coverage rate for the configuration information in the platform. And the design of flexible web framework makes this tool more efficiently accommodate to different platforms without manually modifying the web pages.

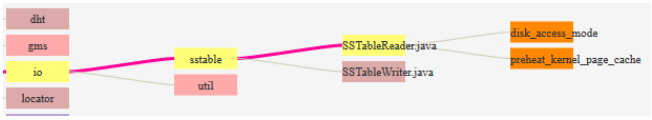

Figure 10. Related Configurations

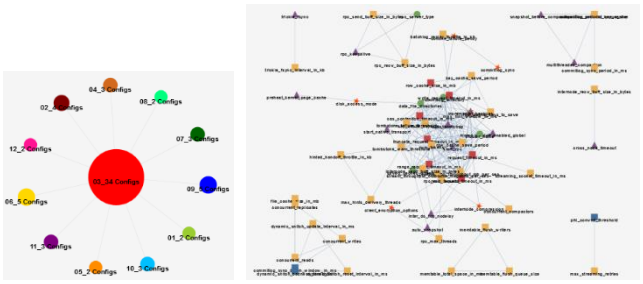

Figure 11. Abstract Graph and Original Graph

VIII. CONCLUSION

This web-based tool provides a convenient method to visualization of configuration information. It presents the overview and details of this information to users, and the web interface is efficient for user interaction to explore the lists, images and graphs to understand the data. To view the information from different aspects, this tool applies layering to provide configuration view and source codes view. And dividing mechanism is used to display certain data for specified configuration. Especially the grouping technique could narrow down the scale and complexity of the configuration connection graph, and shorten the time for graph refreshing, which is demonstrated in evaluation. In addition, this tool uses flexible web framework to visualize configurations of different distributed platform.

However, the application of this tool is not achieved in more different distributed systems to demonstrate the advantage of its flexible design. This should be finished in the future to add more distributed systems. In addition, the information about how some configurations can affect the flows of the programs during the real time execution is not visualized by this tool. It will be beneficial for the understanding of the relation between configurations and programs execution if this further information could be presented to the users in the future.

REFERENCES

[1] Chen, H., Jiang, G., Zhang, H., and Yoshihira, K. "Boosting the performance of computing systems through adaptive configuration tuning." Proceedings of the 2009 ACM symposium on Applied Computing. ACM, 2009.

[2] The Apache Cassandra Project. http://cassandra.apache.org. (2014, Nov.10)

[3] Lakshman, Avinash, and Prashant Malik. "Cassandra: a decentralized structured storage system." ACM SIGOPS Operating Systems Review 44.2 (2010): 35-40.

[4] JavaScript InfoVis Toolkit. https://philogb.github.io/jit. (2014, Oct.15).

[5] Liu, S., Cui, W., Wu, Y., and Liu, M. "A survey on information visualization: recent advances and challenges." The Visual Computer (2014): 1-21.

[6] Y. Wu, G.-X. Yuan, and K.-L. Ma. Visualizing ow of uncertainty through analytical processes. IEEE Trans. Vis. Comput. Graph., 18(12):2526-2535, 2012.

[7] E. Bertini, A. Tatu, and D. Keim. Quality metrics in high-dimensional data visualization: An overview and systematization. IEEE Trans. Vis. Comput. Graph., 17(12):2203-2212, 2011.

[8] D. Lloyd and J. Dykes. Human-centered approaches in geovisualization design: Investigating multiple methods through a long-term case study. IEEE Trans.Vis. Comput. Graph., 17(12):2498-2507, 2011.

[9] C. Tominski, C. Forsell, and J. Johansson. Interaction support for visual comparison inspired by natural behavior. IEEE Trans. Vis. Comput. Graph., 18(12):2719-2728, 2012.

[10] C. G. Healey and B. M. Dennis. Interest driven navigation in visualization. IEEE Trans. Vis. Comput. Graph., 18(10):1744-1756, 2012.

[11] D. Selassie, B. Heller, and J. Heer. Divided edge bundling for directional network data. IEEE Trans. Vis. Comput. Graph., 17(12):2354-2363, 2011.

[12] N. Cao, Y.-R. Lin, X. Sun, D. Lazer, S. Liu, and H. Qu. Whisper: Tracing the spatiotemporal process of information diusion in real time. IEEE Trans. Vis. Comput. Graph., 18(12):2649-2658, 2012.

[13] N. Ferreira, L. Lins, D. Fink, S. Kelling, C. Wood, J. Freire, and C. Silva. Birdvis: Visualizing and understanding bird populations. IEEE Trans. Vis. Comput. Graph., 17(12):2374-2383, 2011.

[14] jQuery. http://jquery.com/. (2014, Oct.5).

[15] Syntax Highlighter. http://alexgorbatchev.com/SyntaxHighlighter/. (2014, Oct.10).