JobPlot: Visual Analysis of Load-Balance in Large-Scale Distributed System

Shaolun Ruan, Jiansu Pu, Qiang Guan, Yuwei Zhang, Tingting Zhang, Yunbo Rao, Hui Shao

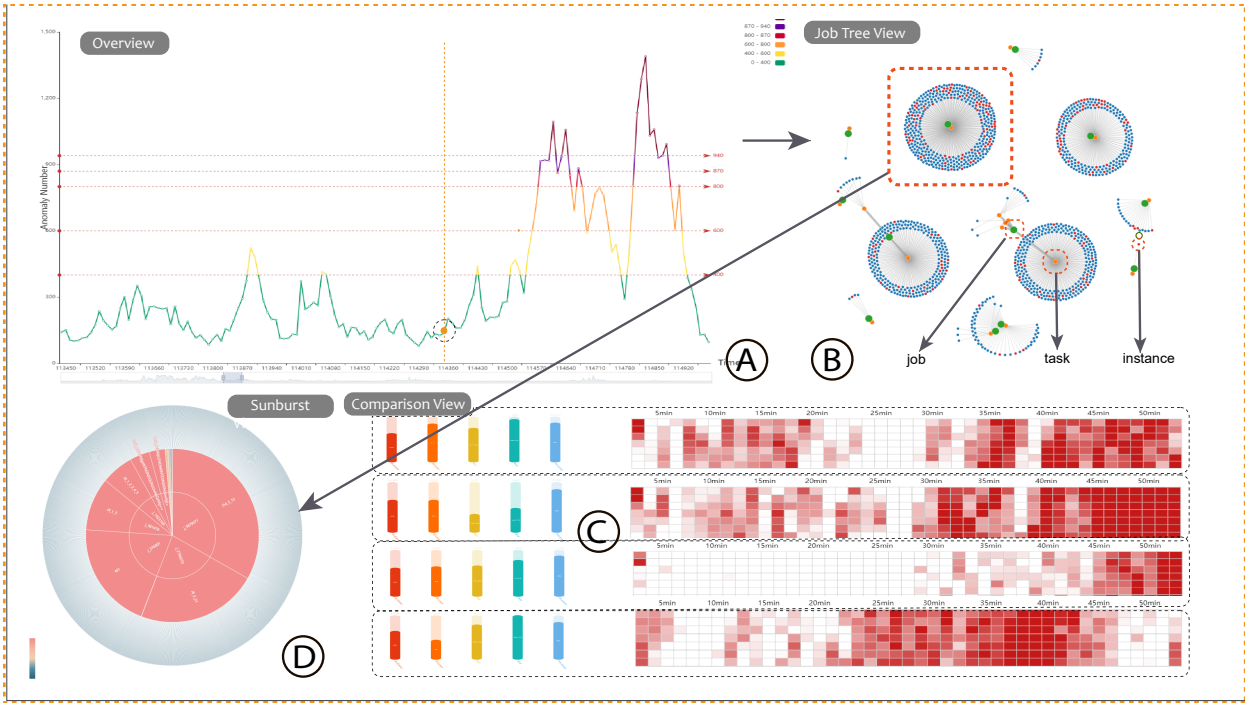

Fig. 1. A screenshot of JobPlot consisting of four areas: (A) Overview shows an overview of the number of high-load abnormal servers in each time stamp. (B) Job Tree View shows the distribution of job scheduling via forced-directed tree graph and the value HLCC of each job. (C) Comparison View shows the status of the node's entire process and parameters in executing requests. (D) Sunburst View shows the number of job scheduling nodes distribution.

Abstract—With the increasing amount of data transmission in large-scale e-commerce industry, managing such an enormous amount of requests simultaneously becomes a key problem. It is an effective method to handle the requests by applying task scheduling on the distributed server system. However, high load on the servers is inevitable in scheduling when requests of a server node process extreme large amount of tasks in a short period of time. It is difficult to maintain load balance without an effective high-load surveillance approach. To best of our knowledge, existing methods of monitoring high-load abnormal nodes are neither flexible or intuitive and are not capable of detecting server node anomalies such as positioning unreasonable requests sent by the clients. In this paper, we propose a job scheduling monitoring method based on visual analysis from a real dataset, which allows the monitoring personnel to know the status of running nodes in the area and observe the suspected requests causing a high load of servers via various view components we inject into the system.

Index Terms—Task scheduling, visual analysis, abnormal detection

1 Introduction

E-commerce is becoming more popular, creating both a risk and an opportunity. The risk is that the servers on the back end tend to run at high load when the number of requests initiated by users reaches a great number, especially for some commercial corporations which have a big scale. To address this problem, an effective approach is, for example, perform real-time task scheduling on requests sent by users. Client machines are often divided into different containers by the scheduler, and requests are encapsulated into different instances and run by different server nodes on the server side at the same time. However, it is a tough job to monitor the machines at a high load or in risk. The operator cannot know all the server nodes that are running abnormally at

- *Shaolun Ruan, Department of Computer Sci Tech, University of Electronic Science and Technology of China. E-mail: haywardryan@foxmail.com.*
- *Jiansu Pu, Department of Computer Sci Tech, University of Electronic Science and Technology of China. E-mail: jiansu.pu@uestc.edu.cn.*
- *Qiang Guan, Computer Science, Kent State University, Kent, Ohio, United States. E-mail: qguan@kent.edu.*
- *Yuwei Zhang, Department of Computer Sci Tech, University of Electronic Science and Technology of China. E-mail: yuweizhang623@gmail.com.*
- *Tingting Zhang, Department of Computer Sci Tech, University of Electronic Science and Technology of China. Email: 2414550385@qq.com.*
- *Yunbo Rao, School of Information and Software Engineering, University of Electronic Science and Technology of China. Email: raoyb@uestc.edu.cn.*
- *Hui Shao, Department of Computer Sci Tech, University of Electronic Science and Technology of China. Email: sophyond@163.com.*

a certain moment in a short time. Meanwhile, the monitoring machine can't display exactly which unreasonable user action caused all the server exceptions running certain jobs. The opportunity is that some thoughtful sponsors provide a comprehensive dataset to the public. As a result, we can do a lot of research in this domain via the useful graphic research tools to address the problems above.

The main contribution of this study is to simulate the real-time monitoring of the whole process of the server working state through data visualization means, overcome the disadvantages of the previous human operation and expand the function as follows:

- Understand the operating status parameters (CPU utilization, memory utilization, disk utilization, etc.) of the normal and abnormal servers under the time node and various hardware indicators (CPU number, memory size, etc.) of the server itself timely and intuitively.
- Find out the suspected request making the server run abnormally through the task, job, and instance information carried by the server, or which user's abnormal operation resulted in it through the visual components. Besides, find the law and analyze the cause deeply.
- Visually observe the inclusion number and relationship between a requests tasks, jobs and instances, meanwhile, get the job information that mainly occupies the server resources in time.
- Quickly grasp the working state of a server node in the past period of time through the refresh of the visual component chronologically, assist the staff to find the law and judge the error motive of a abnormal node.
- Understand the running status of all server nodes on the entire recorded timeline and the number of server anomalies throughout the platform at a certain moment, locate high-load peaks.

In short, the innovation of the method this study proposed is to make the operation process simple and smooth on the basis of the traditional monitoring system by adding visual elements. On the basis of this idea, various functions are added to help the monitoring personnel to locate anomalies properly. Through visual means to analyze the cause of the abnormality and locate the source of the fault, thereby effectively controlling the operating state of the server in real time, and escorting the conventional operation of the huge commercial system.

In this paper, we propose to collect valuable trace data about the E-commerce domain for our study. On the basis of figuring out the basic framework of the dataset, we refine subsets that suit our study from it and construct a new data organization. All the research we do work from a BS-architecture real-time simulation system based on the dataset we select. The explanation of the data will be expanded in Section 3.

2 Background

2.1 Distributed Job Scheduling

Job scheduling is a computer feature for controlling unattended background program execution of jobs, which is commonly called the batch scheduling. Job schedulers are required to orchestrate the integration of real-time business activities with traditional background IT processing across different operating system platforms and business application environments. Job scheduler needs to manage the job queue for a cluster of computers at the client end via several parameters such as job priority, computer source availability or the number of simultaneous jobs allowed for a user. This job scheduling method has become a common method for large-scale e-commerce business platforms to handle user concurrent requests and as shown in the XXX it works stably and well.

Figure 2. illustrates a model of a job scheduler assumed in this paper. In the figure, a job submitted by a user arrives at the shared job queue, and the job scheduler obtains the number of processors requested by the job (job size). No other information is given to the job scheduler.

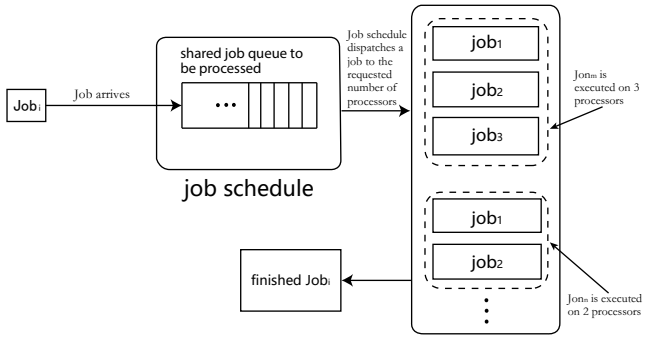

Fig. 2. A model for a job schedule

The job scheduler gathers the status of processors, idle or busy, and dispatches a job in the shared job queue to idle processors. Here, there are two policies to execute a job. In the first policy, a job scheduler dispatches a job to S processors and guarantee the job to be executed on S processors until its completion. Here, S denotes job size. In the second policy, a job scheduler does not guarantee the job to be executed on S processors, that is, the number of processors to execute the job depends on a congestion level of jobs on the parallel computer. A job executed in the first policy is called a rigid job.

Meanwhile, once an operator processed the workflow is instantiated, it is referred to as a task. The instantiation defines specific values when calling the abstract operator, and the parameterized task becomes a node in a DAG. Based on this layer processing, the task is split into several pieces that can be dispatched to the server for processing, which is commonly called task instance. A task instance represents a specific run of a task and is characterized as the combination of a dag, a task, and a point in time. Task instances also have an indicative state, which could be running, success, failed, skipped, up for retry, etc. Because of the abstraction and complexity of the process, Figure 2. shows us how a job instance is split into task instances step by step.

As shown in Figure 3, a request from a client node is processed into several jobs and one of them was sent to the next layer. The job schedule executed a relative algorithm to dispatch it as two task packages. But the server node cannot handle this kind of command, and as a result, each task is split into several task-instance (Hereinafter, task instance is simply referred to as instance). Every instance is distributed on different servers. Until this time, the process of job scheduling is over.

3 Related Work

Faced with the problem of abnormal detection [1], Xiaowei Qin et al proposed an object-oriented detection framework [2], which has two-step clustering called hourglass clustering. Two parameters, key quality index and causal parameter, are used to cluster them into different types by combining the hybrid algorithm of self-organizing map (SOM) and k-medoids.[3] Pei Yang et al. propose using generated antagonistic networks (GANs) to detect anomalies.[4] Daojing He et al. introduce the advantages of using a software-defined network (SDN)[5] to detect traffic anomalies.[6] A. R. Jakhale uses data mining [7][8][9]technology with a sliding window model and clustering technology to check

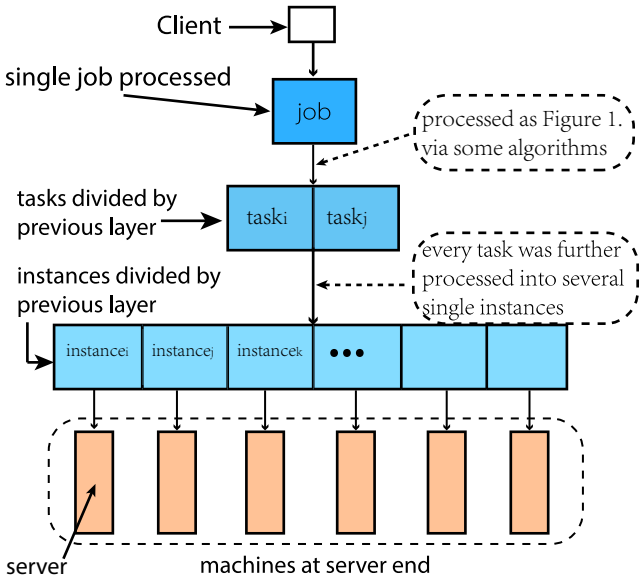

Fig. 3. Workflow from the Job Layer

abnormal data packets of network flows.[10] Alireza Tajary et al. propose a throughput-aware transient fault detection method, which takes advantage of the characteristics of multi-core server processors.[11] To identify anomalies [12][13] in large, dynamic and heterogeneous data, Nan Cao et al. introduce a visual interaction [14][15][16][17][18] system and framework which is called Voila. The system mainly realizes on-line monitoring [19] and interaction with users.[20]

Y.B. Luo et al. propose an anomaly detection method based on a flow-level limited penetrable visibility graph (FL-LPVG). This method constructs a complex network based on network flow sequence, extracts network flow feature sequence by mining the structural behavior patterns of correlation graphs, transforms statistical feature sequence [21] into association graph by using LPG, and detects abnormal traffic through data mining and information theory technology based on entropy. Its advantage is that this method greatly simplifies the process of anomaly detection, and effectively reduces the dimensionality of high-dimensional data. But to improve the efficiency of this system, we must fully mine the behavior characteristics from the massive data [22]. As a result, it certainly brings the challenges of how to deal with large data and how to extract effective information.[23]

Josef Kittler et al. introduce the concept of domain anomaly when solving the problem of anomaly detection [24][25]. There are many aspects of an anomaly, and each domain is one of many aspects of an anomaly. On this basis, they refer to Bayesian probabilistic reasoning [26] device and propose a unified anomaly detection framework to identify and distinguish anomalies in each domain. The device provides a classification of domain anomaly events by defining various anomaly properties. The innovative feature of the framework is that it exposes the multifaceted nature of the anomaly, and can make it possible to identify the various causes that may cause anomalous events, as well as the corresponding detection mechanism.[27]

4 Data Design and Requirments

4.1 Data Introduction And Processing

The dataset cluster-trace-v2018[1] used in this study is tracked and sampled from one of our production clusters which is part of the Alibaba Open Cluster Tracking Program [2] . The Program was released by the Alibaba Group and proposed in 2017. By providing cluster tracking from actual production, the program helps researchers, students and those interested in the field better understand the features and workloads of modern Internet data centers (IDCs). The project has been available in Version 2017[3] and Version 2018 so far. The Version 2017 is the prototype of the program, which records a 12-hour tracing of 3000 server nodes. For the comprehensiveness and completeness of the research, we intend to study based on Version 2018.

The dataset cluster-trace-v2018 traced 4000 nodes during 8-day, which consists of 6 separate data files and each of them records a sort of information in actual production including meta information of servers, batch recording, and container recording. The point is the amount of row data, 206GB in total, is too enormous for a regular system to process via common methods. As a result, we applied some fresh idea in python script to handle the problem which is not supposed to be an obstacle for us all. The algorithm is illustrated as follow:

Algorithm 1.data file reader
Input: data_path
1:**initialize**:reader=csvReader(data_path)
2: **for** row **in** reader do
3: filter by Conditions
4: write(row)
5:**end for**

In order to simulate real-time data flow in real production which cannot contain an enormous amount of data to render and refresh views in the system, we choose a part of origin data, the 2nd day within the 8-day period, to be the source data in our study. The selected dataset has complete and representative data structures of origin data to be the source of simulated data flow, meanwhile, all of the data is stored in the database which can deliver the stream to the graphic components quickly and accurately.

4.2 Anomaly Determination

The dataset itself has a number of parameters to illustrate the status of a server node in the back end. The big-scale server clusters anomaly determination algorithm is similar to the processing mode for anomaly identification in cloud computing infrastructures. The method, most relevant principal components(MRPC)[zhu], cannot apply in our study because of the lack of the key parameters to normalize the unit and scale dynamically. Control threshold is an effective method to determine the abnormal domain which depends on the three valuable fields: utilization of CPU, utilization of memory and utilization of disk. By assigning weights to these three parameters the system could be determined whether the node is at a high load in processing the task instance. Algorithm 2 depicts the approach to identify an anomaly rendering on the view in our study:

$$Value_{hl} = x \cdot \frac{S_{(\beta+X)}}{S} + y \cdot \frac{S_{(\alpha+X)}}{S} + z \cdot \frac{S_{(\alpha+\beta)}}{S} \quad (1)$$

In the algorithm above α, β , and γ represent the threshold of CPU, memory, disk respectively. We use the idea of mathematical probability statistics to set proper thresholds to hold the proportion of the abnormal high-load nodes at nearly 10%. The value of high load can be calculated like this.

4.3 Components Design

4.3.1 Entry View

In the entire system, there must be a data overview module that maps the entire timeline. This module is the core of the entire user UI interface and represents the state of all server nodes throughout the timeline. Since it is intended to show the operator the trend of all anomalous

[1][https://github.com/guanxyz/clusterdata/blob/master/cluster-trace-v2018/trace_2018.md]

[2][The Alibaba Cluster Trace Program is published by Alibaba Group. By providing cluster trace from real production, the program helps the researchers, students and people who are interested in the field to get a better understanding of the characteristics of modern internet data centers (IDC's) and the workloads. https://github.com/guanxyz/clusterdata]

[3][https://github.com/guanxyz/clusterdata/tree/master/cluster-trace-v2017]

servers, in the expression of the trend, a line graph is used to simulate the number of all anomalous server nodes contained in each time stamp. The component is also an entry component for the operator to use the system. In this component, the user can filter the node according to the trend expressed by the image for a special moment or a time that he wants to observe, and refresh other views according to the passed parameters. Thereby achieving a response to the entire layout of the page, the so-called Level-0 component.

Figure 5. shows the number of anomalies of server nodes from a time sequence. The view is designed as follows: the horizontal axis represents time (the original data file timestamp field), and the vertical axis represents the number (¡4000) of anomalous nodes at that time. The inflection point of the curve is used to represent the number of anomalies, and then all the inflection points are connected to form a non-breakpoint curve. According to the characteristics of the original data set, the control range of the horizontal axis is between (100000, 172000), the unit is seconds, and the whole day is recorded; the vertical axis is the number of abnormal nodes between (0, 4000), and regular changes are made as time progress.

We control the number of abnormal nodes within 5% through the processing of the algorithm, meanwhile, the abnormal level of the node is represented by the value of the color block through the mapping of the visual channel as the Figure 4. shows. Besides, The operator is supposed to trigger the event of the inflection point and so other view components could be refreshed according to the time stamp the entry view delivered.

4.3.2 Job Anomaly Detection

We will design a progressive response module Level-1 based on the Level-0 component. This type of module is the progressive of the previous module, and further reveals a more detailed graphic view through the user's operation on the previous module. This level is intended to design two views, one of which is a tree of a force-directed layout, and the other is a sunburst with a strong response function. All of them are to explain the affiliation between jobs, tasks, and instances through the specific expression of distributed task scheduling, thus helping to determine the cause of the user causing the anomaly.

The main processing module for abnormal positioning is shown in Figure 5. below. This sort of node set is divided into three levels. It is necessary to reflect the relationship between each level (job, instance, task) in the task scheduling through the view. The basic element is intended to adopt the node set of the tree structure, and each set represents the distribution of tasks and instances under the parent node job in the center logically.

The point is reflecting the anomalous server node in the view component except the display of the job scheduling. As shown in Figure 3 above, each task instance is processed by a solitary server machine independently, and we display the load level of the server node processing the instance of a task. This method seems to be random, but through a graphic approach, the operator can get the law of the layout of the anomalous server node and get the idea of some abnormal job nodes further.

4.3.3 Time-series Supplement

The graphic approach showed above is for all back-end server nodes throughout the timeline. However, they appear to be not so convincing when the operator intends to observe the status of a single node, and the so-called Level 1 and Level 2 components just illustrate status at a certain moment at the same time. We add new components to enhance the timing description and enrich the description of a specified abnormal node we selected from the view in the previous layer. We intend to depict the status of a certain server in the past period of time via the heating-map and display various operating parameters of the machine via a sort of histogram. Through the two components of the heat map and histogram, we make the sequential description of special nodes more comprehensive and detailed.

Figure 6. depicts the change of the status of a separate node and the difference of the color block illustrates the state of the performance of a server in the past 60 minutes.

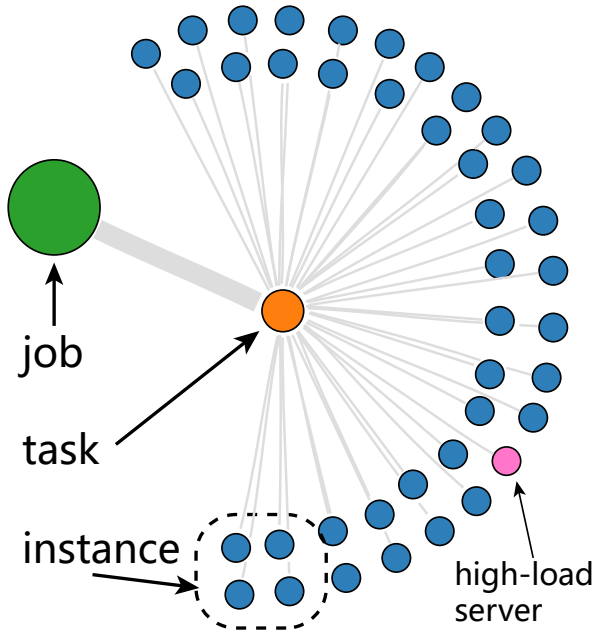

Fig. 4. The Model of Anomaly Detection

Each of a block in Figure 6.represents a span of 10 seconds in the past period of time and the area consist of all the blocks represents a timeline of an hour. The view component illustrates the operator the performance of a single server node temporally via a map of the visual channel. Those blocks with low saturation represent the good performance of the hardware of a server and those blocks in deep red represent the performance of the server is not so superior with a high load value of hardware at the moment.

For instance, the Area-1 in Figure-6 illustrates the server A, for example, was in a high load state for the past 15 minutes, especially serious in about 14 minutes before the time you selected to observe. Then within 17 minutes after that, the performance of a server running status remains at a normal level as shown in the Area-2, and after that, in the Area-3 the server has another high load phase somehow.

This level of view component enhances time scalability of a particular server node through an intuitive abstraction of performance and enhances the detailed description of nodes with another view component histogram which will analyze together with other modules in the next chapter.

5 Use Cases

In order to prove the feasibility of our study, we simulated the use and operation in the actual environment based on the dataset cluster-trace-v2018. The dataset is valuable for our simulation because the dataset is traced and obtained from the actual distributed system operation. To prove the feasibility and effectiveness of our research, we choose a typical and convincing case from the process we used. Figure 6. is an example of the Level-1 view component, which depicts an actual situation of job scheduling at time stamp 114550.which is capable of reflecting the suspected job causing the high load of server nodes in the distributed system, meanwhile, it is the entry view of the Level-2 component. As shown in Figure 6. there are eight job units and seven job-clusters processed at the time stamp 114550(the green node represents the job and the orange node represents the task), and each of a job has one child task only. We can see there are several instances nodes respected by the blue dots under each task node. At the same time, the server machines processing the instance nodes in red represents a high load status at the moment. To best of our knowledge a job scheduler schedules every job request into several different server machines, in other words, each machine has no less than one instance running on

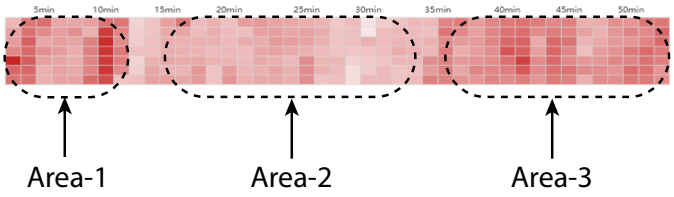

Fig. 5. Workflow from the Job Layer

itself. As a result, we cannot judge a job node surrounded by high load instance nodes in red as a suspected job node which causing an abnormal status of a server machine, as job-B shown in Figure 6. The job-B cluster has several server machines in high load status processing its task instance. We can estimate the job-B is not the unreasonable request sent by clients due to a relatively small amount of red nodes. Here we present a new concept: High Load Correlation Coefficient(HLCC) which represents the correlation between the central job and the high-load machines running the instance scheduled by the job scheduler. The HLCC can be calculated by the following algorithm:

$$HLCC = \frac{N_{ab}}{N} \tag{2}$$

As shown in the algorithm above, N represents the number of instances of a job in total. Nab represents the number of abnormal server machines processing the instances. In this case, we can make sense the definition of a suspected job that the HLCC of job-A is 90/168(0.536), nearly three times as much as the HLCC of job-B 36/220(0.164). The group-C is full of normal job clusters and almost no nodes in high-load status. The meaning of setting this control group is to illustrate it is most likely that job-A causes abnormal status of almost every server nodes running the instances at the moment, at the same time, job-B is a great and healthy request sent from the client. Although there are several abnormal nodes in red surround it, it cannot depict that job-B is the suspected abnormal job. It is most likely all of the high-load server nodes are due to any other task instance the server is now processing, or the server itself has poor performance of hardware. Thus, we have completed the abnormal mapping from the server to the user. However, we cannot conclude it is environment problem caused the anomaly such as during the client request peak period because there are normal sample job exists actually. Figure 6. is a screenshot of JobPlot at timestamp 114550. In this section, we will illustrate the process of how JobPlot works via a simulation of manipulations. As an entry component of JobPlot, area A shows the operator the distribution of the high-load servers amount. For example, we can see the value is maintained at a stable level under 400 from 113450 to 114500 and rises dramatically to 1000 plus from then on. The time stamp controlling bar makes it possible to observe the status under an enormous amount of timestamp through a limited timeline. We selected a typical moment 114550 in the are A and then area B Job Tree View is refreshed according to the time we chose. As we illustrated above, all of the job clusters in the graph are running steadily except job-A whose HLCC is much higher than any other job clusters in the view. Although the HLCC of other jobs is not zero, its value is within an acceptable range that we can conclude the central job node is not the reason causing these servers anomaly.

Area D is a sunburst view of job schedule distribution at the same time stamp as area B. As shown in the graph, eleven job clusters are recorded in the sunburst view, which is able to display the amount and proportion of job clusters intuitively. Operators can view the overall distribution of instances under individual jobs or tasks through interactive events, at the same time, can observe the detailed information about job schedule distributions. All of these characteristics make it more reasonable and practical to interact with the comparison view component as shown in area C.

From the three views described above, we have a comprehensive understanding of the status of all servers in a big-scale distributed system and server nodes worthy of attention. As a view component for an individual node, area C depicts an intuitive display of key parameters of running and status of nodes in time-series.

The bar in area C represents the CPU utilization, disk utilization, number of incoming network packages, number of outcoming network packages and memory utilization of the selected server machine from left to right. The calendar graph is placed on the left side of the bar chart, which reflects the status record of the server node in the past every ten seconds(each block represents ten seconds). According to visual mapping, the deeper the red, the greater the saturation, the higher the load of the node. As shown in area C, there are four individual modules in the body of Comparison View. C1 and C2 modules represent two servers in high load under circled job in area B and C3 and C4 modules are tracing records of normal server machines. As a result, we have completed the comprehensive time-series description of servers executing the request. As we expected, the overall running status of the nodes in the high load group is not as good as the status of the comparison group, whose blocks in red are less and more sparse than the top two modules intuitively. At the same time, the value of the bar charts is smaller than the high load one as well.

In this way of observing, the time-series view makes it comprehensive to find out the high load abnormal period throughout the past one hour. It appears that the timestamp after thirty minutes is the common abnormal period regardless of whether the server processing the instruction is in a high load state. As a result, we can see that the server in the distributed system has an abnormal peak in the past 30 minutes, which may be caused by the peak usage time of the user or some factors such as voltage or weather conditions in the area.

Fortunately, there is another proof confirming our visual design. The observed job cluster is traced from time stamp 114510 and lasted nearly 80 seconds. The moment 114510 can be checked in the C1 and C2 modules, including four blocks in deep red from 114510 to 114550(current). As a result, the moment when the server is abnormal is exactly the same as the time when the job is scheduled to execute the execution request. This is a valuable discovery to illustrate the quantity HLCC we proposed is indeed reasonable and interpretable.

6 Conclusion

In this paper, we have presented a modular visual analytics framework for the progressive job task monitoring in the big-scaled distributed system. We injected four kinds of visual components to make a full description of the scheduled job clusters, and the high load servers processing the requests in the back system. As shown in the previous section, we proposed a new quantity HLCC(high load correlation coefficient) to measure the relationship between the job itself and the cause of high load. The higher the HLCC, the greater the likelihood that the job will cause a processor abnormal. We demonstrated a valuable Job Tree View component to reflect the relationship of actual job scheduling along with HLCC. In the next layer of the visual module, we can observe the running status of an individual server node in the distributed system based on Job Tree View and Sunburst View. In this way, we can know the operating parameters and timing information of the node throughout the timeline, which is an effective approach to

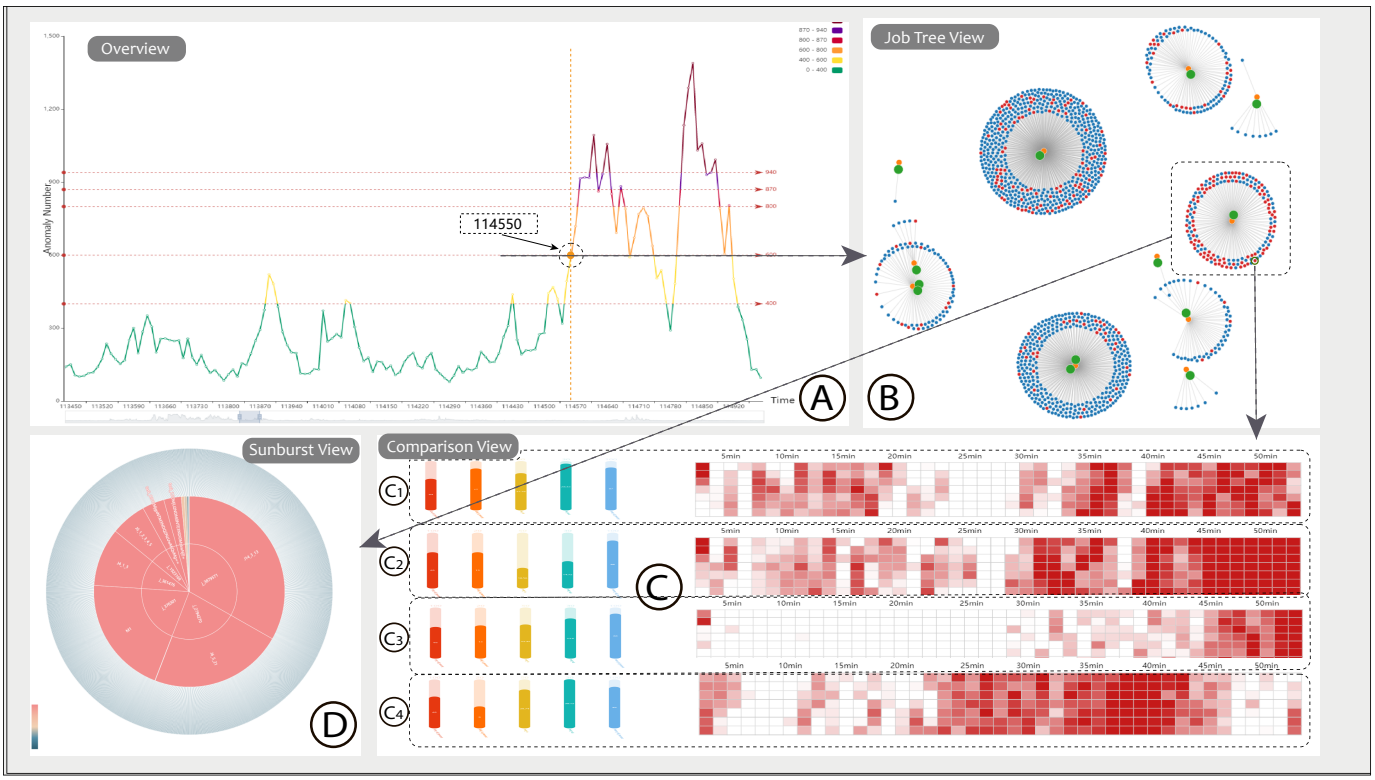

Fig. 6. Workflow from the Job Layer

know the states of all running servers intuitively.

As future research, we would like to look deeper into dataflow and the processing speed of the system. The most ideal approach is to improve the system into in Situ model, which renders the view components by directly passing in the data stream. Another task is to optimize the module structure of the system to meet enterprise requirements, adapt to a variety of operating environments and various concurrent operations. We are also working on implementing an option to present users with guidance on refinement actions and optimization possibilities which would most likely result in model improvements, based on the internal model stress level and certainty. We are planning to make this guidance more tailored to the data and task at hand through active learning from the user interactions and relevance feedback.

REFERENCES

[1] Ma Ning, Peng Yu, Wang Shaojun, Gao Wei, A weight SAE based hyperspectral image anomaly targets detection, 2017 13th IEEE International Conference on Electronic Measurement Instruments (ICEMI), 2017.

[2] J. Al Dallal, P. Sorenson, System testing for object-oriented frameworks using hook technology, Proceedings 17th IEEE International Conference on Automated Software Engineering, 2002.

[3] Xiaowei Qin, Shuang Tang, Xiaohui Chen, Dandan Miao, Guo Wei, SQoE KQIs anomaly detection in cellular networks: Fast online detection framework with Hourglass clustering, China Communications, pp.25-37, 2018.

[4] Pei Yang, Weidong Jin, Peng Tang, Anomaly Detection of Railway Catenary Based on Deep Convolutional Generative Adversarial Networks, 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2018.

[5] Arpita Prajapati, Achyut Sakadasariya, Jitisha Patel, Software defined network: Future of networking, 2018 2nd International Conference on Inventive Systems and Control (ICISC), 2018.

[6] Daojing He, Sammy Chan, Xiejun Ni, Mohsen Guizani, Software-Defined-Networking-Enabled Traffic Anomaly Detection and Mitigation, IEEE Internet of Things Journal, 2017.

[7] Vania Bogorny, Shashi Shekhar, Spatial and Spatio-temporal Data Mining, 2010 IEEE International Conference on Data Mining, 2010.

[8] Hetal Thakkar, Barzan Mozafari, Carlo Zaniolo, A Data Stream Mining System, 2008 IEEE International Conference on Data Mining Workshops, 2008.

[9] G. Williams, R. Baxter, Hongxing He, S. Hawkins, Lifang Gu, A comparative study of RNN for outlier detection in data mining, 2002 IEEE International Conference on Data Mining, 2002. Proceedings, 2002.

[10] A. R. Jakhale, Design of anomaly packet detection framework by data mining algorithm for network flow, 2017 International Conference on Computational Intelligence in Data Science (ICCIDS), 2017.

[11] Alireza Tajary, Hamid R. Zarandi, An Efficient Soft Error Detection in Multicore Processors Running Server Applications, 2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP), 2016.

[12] Zicheng Liao, Yizhou Yu, Baoquan Chen, Anomaly detection in GPS data based on visual analytics 2010 IEEE Symposium on Visual Analytics Science and Technology, 2010.

[13] Huamin Ren, Thomas B. Moeslund, Abnormal event detection using local sparse representation, 2014 11th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 2014.

[14] Chihiro Sakazume, Hiroyuki Kitagawa, Toshiyuki Amagasa, DIO: Efficient interactive outlier analysis over dynamic datasets, 2017 Twelfth International Conference on Digital Information Management (ICDIM), 2017.

[15] Xiaodong Zhu, Ji Zhang, FRIOD: A Deeply Integrated Feature-Rich Interactive System for Effective and Efficient Outlier Detection, IEEE Access, 2017.

[16] J. Heer, S. K. Card, J. A. Landay, "prefuse: a toolkit for interactive information visualization", Proceedings of Computer Human Interaction, pp. 421-430, 2005.

[17] Weijie Wang, Baijian Yang, Victor Yingjie Chen, A visual analytics approach to detecting server redirections and data exfiltration, 2015 IEEE International Conference on Intelligence and Security Informatics (ISI), 2015.

[18] Z. Liao, Y. Yu, and B. Chen. Anomaly detection in gps data based on visual analytics. In IEEE Symposium on Visual Analytics Science and Technology (VAST), pp. 51-58, 2010.

[19] Ruoning Song, Fang Liu, Real-time anomaly traffic monitoring based on dynamic k-NN cumulative-distance abnormal detection algorithm, 2014 IEEE 3rd International Conference on Cloud Computing and Intelligence

Systems, 2014.
[20] Nan Cao, Chaoguang Lin, Qiuhan Zhu, Yu-Ru Lin, Xian Teng, Xidao Wen, Voila: Visual Anomaly Detection and Monitoring with Streaming Spatiotemporal Data, IEEE Transactions on Visualization and Computer Graphics, pp. 23 33, 2018.
[21] Cheong Hee Park, Anomaly Pattern Detection on Data Streams, 2018 IEEE International Conference on Big Data and Smart Computing (BigComp), 2018.
[22] Li Xinran, Research on massive data 3D-visualization, 2012 IEEE International Conference on Computer Science and Automation Engineering, 2012.
[23] Y.B. Luo, B.S. Wang, Y.P. Sun, B.F. Zhang, X.M. Chen, FL-LPVG: An approach for anomaly detection based on flow-level limited penetrable visibility graph, 2013 International Conference on Information and Network Security (ICINS 2013), 2013.
[24] Hongbin Xia, Wenbo Xu, Research on Method of Network Abnormal Detection Based on Hurst Parameter Estimation, 2008 International Conference on Computer Science and Software Engineering, 2008.
[25] Zhixin Sun, Jin Gong, Anomaly Traffic Detection Model Based on Dynamic Aggregation, 2010 Third International Symposium on Electronic Commerce and Security, 2010.
[26] Yong Li, Wei-Yi Liu, Backward probabilistic logic reasoning algorithm for decision problem with Conditional Event ALgebra on Bayesian networks, 2008 International Conference on Machine Learning and Cybernetics, 2008.
[27] Josef Kittler, William Christmas, Tefilo de Campos, David Windridge, Fei Yan, John Illingworth, Magda Osman, Domain Anomaly Detection in Machine Perception: A System Architecture and Taxonomy, IEEE Transactions on Pattern Analysis and Machine Intelligence, pp.845-859, 2014.