Visual Analysis of Cloud Computing Performance Using Behavioral Lines

Chris Muelder, Biao Zhu, Wei Chen, *Member, IEEE,* Hongxin Zhang, *Member, IEEE,* Kwan-Liu Ma, *Fellow, IEEE,*

Abstract—Cloud computing is an essential technology to Big Data analytics and services. A cloud computing system is often comprised of a large number of parallel computing and storage devices. Monitoring the usage and performance of such a system is important for efficient operations, maintenance, and security. Tracing every application on a large cloud system is untenable due to scale and privacy issues. But profile data can be collected relatively efficiently by regularly sampling the state of the system, including properties such as CPU load, memory usage, network usage, and others, creating a set of multivariate time series for each system. Adequate tools for studying such large-scale, multidimensional data are lacking. In this paper, we present a visual based analysis approach to reasoning and understanding the performance and behavior of cloud computing systems. Our design is based on similarity measures and a layout method to portray the behavior of each compute node over time. When visualizing a large number of behavioral lines together, distinct patterns often appear suggesting particular types of performance bottleneck. The resulting system provides multiple linked views, which allow the user to interactively explore the data by examining the data or a selected subset at different levels of detail. Our case studies using datasets collected from two different cloud systems show that this visual based approach is effective in identifying trends and anomalies of the systems.

Index Terms—Cloud computing, multidimensional data, performance visualization, visual analytics.

1 Introduction

Big Data analytics and discovery relies on efficient cloud computing facilities. Like many high-performance computing systems, a cloud computing service consists of a large number of internetworked machines. But whereas most high-performance computation is task-oriented (e.g. explicitly allocate n nodes to task k exclusively for a time period), cloud computers are more service-oriented (e.g. run an ongoing service continually and add/remove compute nodes dynamically). In order to maximize the efficiency of such a cloud service, it is important to monitor the compute nodes' usage and behavior, in order to identify services that are over/under-allocated, potentially inefficient services that could be optimized, or outlying services or nodes that are misbehaving or failing. However, the privacy of the users should also be respected.

For some goals, the objective is finding similar behavior at differing points in time (e.g. two different runs of the same application under different configurations). To this end, many approaches focus on correlating particular behaviors, regardless of when it occurred. However, similar to many other real-time or streaming applications, cloud computer performance monitoring needs to preserve synchronicity or chronology, as it is interested in concurrent or subsequent behavior. Specifically, some of the main objectives are to identify group behaviors or outlying behaviors as quickly as possible, such as detecting compute node failures, inefficient use of system resources, or even malicious or suspicious behaviors.

- *Chris Muelder and Kwan-Liu Ma are with Department of Computer Science, University of California, Davis, CA, 95616, USA. E-mail: ma@cs.ucdavis.edu*
- *Biao Zhu, Wei Chen, and Hongxin Zhang are with State Key Lab of CAD&CG, Zhejiang University, Hangzhou, Zhejiang, 310058, China.*

Manuscript received October 1, 2015; revised January 4, 2016.

This can be done by tracking general usage metrics, such as CPU load, network load, memory usage, disk read/write load, etc. However, these individual metrics can all be very noisy (i.e. has high variance), and the system is in constant flux. The high frequency patterns make traditional line charts nigh unreadable if all lines are plotted at once. Smoothing the data can alleviate this problem, at the expense of sacrificing high variance patterns, though high variance patterns are frequently important to the analysts.

In time critical situations, it is important for a visual system to very succinctly convey such overall trends whilst preserving time as a dimension. In particular, it is more important to summarize this than to present individual dimensions. While traditional line charts generally suffice for a single dimension at a time, associations between different plots can be difficult or even impossible.

Here, we present an effective visual analytic design for characterizing the evolution of cloud computing activities, based on comparing entities' behavioral similarity over time. We employ windowed signal processing techniques to smooth and summarize noisy behaviors. Then, we use these statistics to derive a force-directed line chart, where lines corresponding to similarly behaving compute nodes are bundled together, and differing behaviors repel each other. This reduces the complex, multidimensional time-series into a simple, intuitive representation of the evolution of these behaviors over time. Since the tasks on such a system are typically parallelized to be running identical code, nodes that are working on the same task would generally exhibit very similar behavior, and thus be bundled together in our similarity line chart. On the other hand, outlying behaviors become quite apparent, as a misbehaving node would separate itself from the rest of the lines. The resulting picture gives the user a high level summary of the system's utilization. This view is combined with several other more conventional views to manage and explore

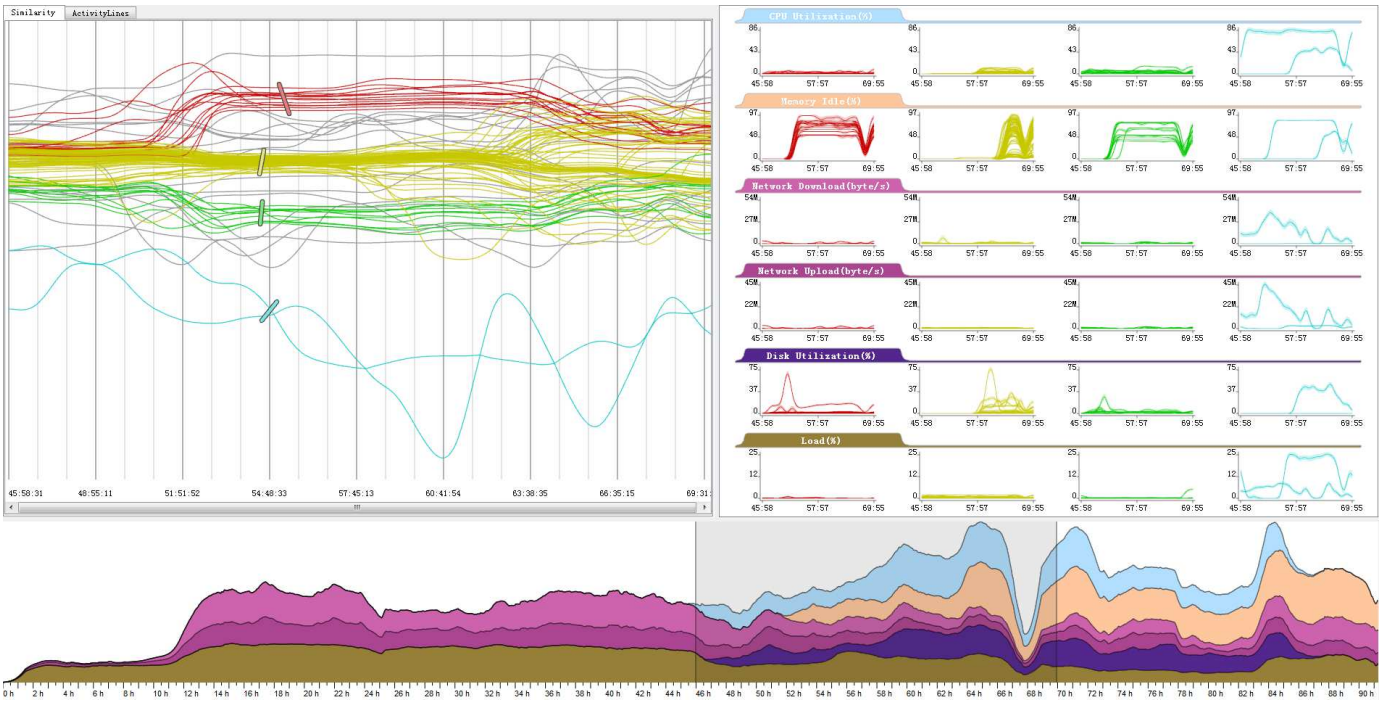
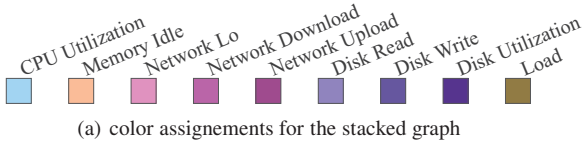


Fig. 1. The three main views of the visual analytics system. The timeline view at bottom summarizes the aggregate behavior of the entire cloud computer per dimension over time. When a time range is selected, the system generates a behavioral similarity line plot (top left), where each line is an individual compute node. These lines are bundled according to their similarity, so that similar nodes group together and anomalous nodes seperate out. Individual line bundles can be selected via brushing, and their properties can be inspected in detail (top right). Color legends are shown in Figure 2.



(a) color assignements for the stacked graph

(b) color assignments for brushing selections

Fig. 2. Color legends for the stacked graph and brushing selection.

the data, allowing for efficient exploration of large and complex performance data sets. The result is an interactive visual analysis tool that can effectively help maintain and optimize large-scale cloud computing services.

2 Related Work

The analysis of time-varying, multidimensional data [1] is an important topic in many fields of research, including data mining [2], visual analytics [3], and GIScience [4]. Most of these focus on detecting trends in time varying trajectories, such as stops or interactions between trajectories. Many of them also focus on aggregating trajectories to produce summaries. The ability to explore such datasets interactively at different resolutions is key to gaining deep insight into them. Jaja et al. [5] develop indexing techniques and search algorithms to efficiently handle temporal range value querying of multidimensional time-varying data. Another proposed method to visualize multidimensional time-varying data is through the estimation of symbolic sequences that are based on both the spatial and temporal attributes of the data [6]. These symbolic sequences reduce the loss of data and at the same time are used for various supervised and unsupervised data-mining tasks. [7] aims at efficient analysis of similar trajectories based on the Longest Common Subsequence (LCSS) model. This in turn increases the robustness of the systems against noisy data making it applicable to real world data like GPS tracking, wireless applications, video tracking [8] and motion capture [9]. Franciosi and Menconi [10] use entropy and statistical linguistic techniques to analyse multi-dimensional time-varying data. Similar to [6], the data is first translated into a multi dimensional symbolic sequence and markers are then defined to encode the characteristics of the time series. The trend of the data is then derived from its entropy with respect to a moving window analysis.

There are several other papers that have contributed towards temporal pattern search. Fail et al. [3] design a visual interface for finding temporal patterns in multivariate temporal data. This approach was employed in [11] for searching similar temporal patterns in clinical history. While [3] and [11] did not have automated methodologies for the mining procedure, [12] employed automated FPM algorithms to the sequences returned by a visual query to automatically view the mined results. Mining of multidimensional time-series helps to understand causal relationships in time-varying data. Mohammad and Nishida [13] propose that causal relations in time-varying data can be unearthed by analyzing meaningful events in time series rather than the raw data itself. In order to do this, they use the RSST (Robust Singular Spectrum Transform) algorithm to identify these points in the time series. The Granger-causality test is then applied to validate the occurrences of the basic events. When the proposed model was

applied to mine records of a real world application, the causality graphs proved very helpful in representing underlying relations between the cause and effect of the dataset.

Many tools exist that can collect low-level trace data from large parallel systems, recording event-level details of parallel processes [14], [15], [16]. A number of visualization techniques have been developed for analysis of such trace data. Often, they employ interactive timelines with line or point plots [17], [18]. Others use animation to represent time [19]. And others use network representations to summarize communication patterns/trends [20]. But these approaches primarily rely on trace-level data, which is unfeasible to collect constantly for an entire system.

Profile level data on the other hand, summarizes the data at a higher level, such as per-process or per-function values averaged over timesteps. ParaProf [21] is an example of a visual system that is designed to study such profile information. Real-time dashboards and other monitoring systems have to rely on profile data because the low level information would be overwhelming. Virtue [22] is one such real-time visualization system which uses immersive 3D techniques to monitor the current state of a system geographically. But these approaches generally only look at one data dimension at a time.

The idea of plotting dimensionless lines over time to exhibit behavior has also been explored in the past. Storylines [23], [24], [25], [26] are a popular discrete form of this, where cluster membership is shown directly by line bundling. However, for the kind of data we are discussing, such a discrete representation of group membership could reduce the fidelity of the input data, as the results would depend on a clustering algorithm. Analog variants of this kind of approach have also been shown to be effective for movement trace data [27]. In this work, the authors employed a dimensional reduction projection to generate the behavioral line summary. However, this approach only works well if the whole dataset range is available at once, and is rigid with its layout, making it less suitable for noisy, highly dynamic, or streaming data.

A force-directed layout [28], [29] of data is adopted in many visualization designs. Besides offering an aesthetically pleasing visualization of the data, the force-direct layout may bring out essential structures in the data. In [30], network data is derived from high dimensional data and visualized with a force-directed layout to guide the following parallel coordinate visualization for assessing the correlation in the data. In [31], multidimensional weather data is first represented as a weighted complete graph and then visualized by laying out the graph based on a force model to reveal the overall relationship between the dimensions, from which the user can further explore the details using other visualization techniques. Our approach also offers an overview of the data based on a force-directed method to guide the following visual exploration of selected data subsets. Unlike most visualization works which employ 2D layouts, our work uses 1D layout.

3 System Design & Methodology

At most cloud computing facilities, the state and performance of devices and computing are typically monitored through simple dashboards which small multiples of individual metrics, or a Gantt chart, which plots one metric over time for all processing units' activities as rectangles or as a heat map over a chosen time period. For large-scale computing, neither of these approaches are visually scalable to large numbers of process or multiple metrics,

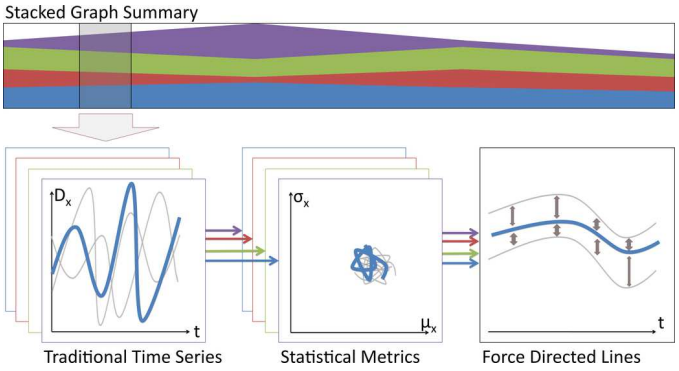


Fig. 3. The overall process of computing behavioral similarity lines for a collection of entities with time-varying metrics. A time range is selected from the timeline view, which depicts overall aggregates over all entities. Each node's time series could traditionally be rendered as individual lines per dimension. These plots can be very noisy, but windowed signal processing metrics map the entities to a 2D statistical space, where similarly behaving entities are consistantly placed. By considering distances in this statistical space, we can compute forces between entities at each time step, and use these forces to lay out a smooth line plot over time.

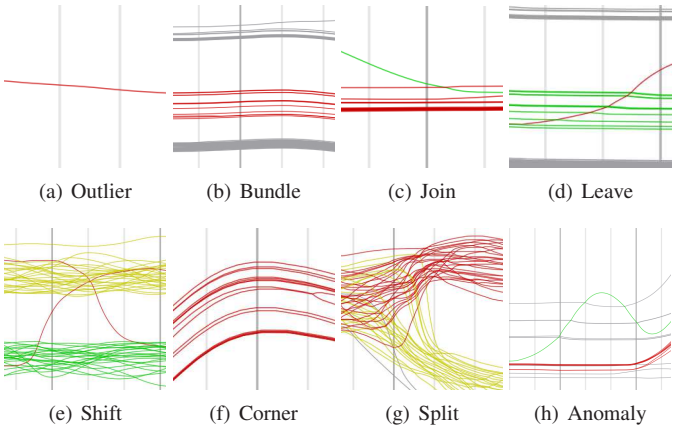


Fig. 4. Example behavioral line patterns. The y axis is dimensionless, so understanding of the plot comes from observing the lines positioning and interaction relative to each other. While the meaning of the patterns that these lines form can be domain dependant, there are a number of common patterns that are fairly general. Lines can either be independent or in a group of other lines. They can join, leave, or switch between groups dynamically. Large groups of lines can change behavior together. And odd or anomalous behavior stands out.

hampering analysts' ability to explore and find connections among all nodes. Rather than trying to present all the low level data directly, we found it better to visually organize nodes according to higher level behaviors of interest, which allows the analysts to comprehensively group processes of a similar behavior, and more quickly identify, then track down and understand the cause or impact of anomalous behavior.

To analyze large complex data, our system is comprised of multiple, tightly linked views as shown in Figure 1. The core technique of our approach is a behavioral plot, which plots similarity between entities as closeness between the lines, comparable to storyline visualizations [26]. There is an underlying analytic process required to generate this plot, as summarized in Figure 3. The user selects a range of time to load, which is pulled from a database. The underlying data can be represented per dimension as sets of 1D time series. These time series are then transformed into 2D traces in a statistical space, consisting of the windowed mean

and standard deviation of each series over time. The traces in this space are now smooth enough to be compared against each other, so for each time step, we use a Euclidean metric to summarize the distance between entities, and compute 1D forces between entities at each time step to lay out lines in a new, behavioral plot.

The vertical axis of this behavioral plot is thus dimensionless, and the horizontal axis is time. Reading this plot is not a matter of reading individual lines on their own, but rather observing their interactions in the context of the other lines. For instance, a single line distinct from the rest of the lines is exhibiting outlier behavior (Figure 4(a)), whereas a bundle of lines would be an example of a cluster of very similar behavior (Figure 4(b)). As entities behaviors change, they often join existing bundles (Figure 4(c)), leave them (Figure 4(d)), or even shift immediately between two groups of behaviors (Figure 4(e)). Sometimes an entire bundle of entities change behavior suddenly, resulting in a sharp corner in the plot (Figure 4(f)). Sometimes entire bundles split in half (Figure 4(g)). This could be indicative of one job finishing only to be replaced by two smaller jobs. And lastly, some behaviors are just unpredictable(Figure 4(h)), which could indicate an anomalous event that affected a particular entity.

We also expose each step in the transformation process as auxilliary views, so that the user can investigate the provenance of an observed behavioral pattern. Each view focuses on a different aspect of the data, at different levels of detail. Selection or brushing in any view is linked to the display of data in the other views, allowing for dataset exploration. As each timestep is transformed and layed out incrementally using temporally local data, the approach is also apt for supporting streaming data.

3.1 Stacked Graph Timeline

The stacked graph timeline (shown in Figure 5) is a broad overview of the data, and the first visualization the user sees upon loading a data set. It shows metrics aggregated accorss the entier cloud machine over time. Each layer of the graph represents one metric, which is normalized and colored according to Figure 2.

From this graph, the user can perceive some macro patterns over time such as a spike or a valley, which may indicate anomalies or bottlenecks within the cloud computing system. While this view is relatively simple, it effectively brings attention to interesting time ranges in the dataset, suggesting the user to conduct further analyses together with other linked views.

3.2 Traditional Scatter/Line Plots

At the other extreme is a detailed view of one dimension over time. While traditionally, such time varying data is represented as line plots, the temporal resolution of the data we were investigating do not scale well in line plot representations, particularly when there are lines with high variance. For example, in Figure 6(a), the selected entity's line is relatively stable, but in Figure 6(b), the selected entity exhibits a large amount of variance, and simply drawing the one line obscures most of the plot. Because of this, our system does not render all the lines at once. Rather, we render the data points in a scatter plot fashion, and only connect the points of a user-selected entity. That, combined with the limitation that such a plot only shows one dimension at a time, makes this representation only acceptable for detailed inspection. Various lensing and interactive fish-eye distortion techniques are also utilized, but they were not found to be sufficient to handle such convoluted data alone. However, even in data entities with high variance, regular cyclic patterns were observable, which led us to investigate the applicability of signal processing techniques.

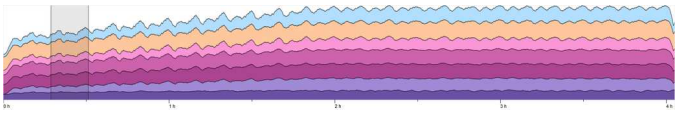
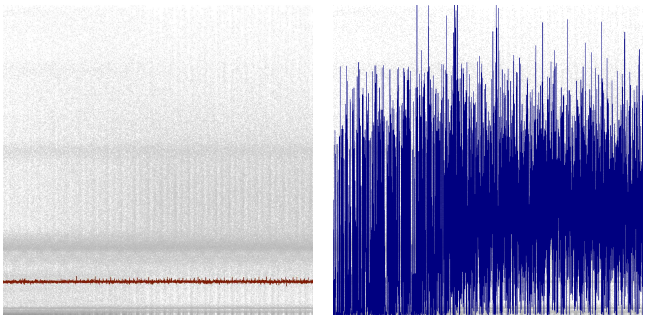


Fig. 5. The stacked graph timeline. The x axis is time, and the height of each layer is the average value of the represented metric over a period of time. This particular data set contains 4 hours of logs.



(a) Stable (b) Noisy

Fig. 6. Even plotting one dimension of the data over time can be problematic when the data is noisy. In some cases, the data is fairly stable and traditional line plots would work (a). However, in many cases, the lines themselves are quite noisy (b). While drawing all the lines would be useless, rendering as points can provide context, and individual traces drawn on top for detailed analysis.

3.3 Statistical Plots

One common technique in signal processing for dealing with noisy, but trending, data is to smooth it out by sampling local means or medians. However, smoothing the data sufficiently enough to show trends sacrifices the information pertaining to the noisiness itself. To counter this, we consider not only local means but also local standard deviations. That is, for each entity i, dimension d, and time t we consider a sliding window of $t \pm a$. In order to avoid sharp jumps as data points enter or exit the window, we compute a gaussian kernel, as is generally considered appropriate for temporal sampling [32]. That is, for a set of data D_i, d, t we compute the weighted mean $\mu_{i,d}(t)$ and standard deviation $\sigma_{i,d}(t)$ as:

$$\mu_{i,d}(t) = \frac{\sum_{k=-a}^{a} D_{i,d,t+k} * w(t,k)}{\sum_{k=-a}^{a} w(t,k)}$$

$$\sigma_{i,d}(t) = \sqrt{\frac{\sum_{k=-a}^{a} (D_{i,d,t+k}^2 - \mu_{i,x}^2(t)) * w(t,k)}{\sum_{k=-a}^{a} w(t,k)}}$$

where

$$w(t,k) = e^{\frac{-(t-k)^2}{k^2}}$$

Once these values are computed, it is possible to plot μ versus σ as a line plot (Figure 7) and clusters readily become apparent. However, such a representation loses the context of time, as it becomes impossible to differentiate if two entities actually crossed paths or not. Such a view is also still limited to one input dimension at a time.

3.4 Behavioral Similarity Plots

While the statistical process can clean up one dimension at a time, our objective is to handle multiple dimensions. In order to combine all the desired dimensions, we compute a similarity metric S as:

$$S_{ij} = \left(\sum_{m=1}^{n} \sqrt{\left(\frac{\mu_{m,i} - \mu_{m,j}}{\mu_{m,max} - \mu_{m,min}} \right)^2 + \left(\frac{\sigma_{m,i} - \sigma_{m,j}}{\sigma_{m,max} - \sigma_{m,min}} \right)^2} \right)^{-1}$$

between every two nodes i, j over time via the aforementioned localised statistics for each timestep. n is the number of user-selected dimensions. The user can control which dimensions to include or exclude. Currently, we normalize the data in each dimension, and employ a standard Euclidean metric. This is appropriate because we want to group entities with both similar μ and σ, and separate nodes that differer in either. We accumulate Euclidean distance of all the selected metrics m to derive the overall similarity. We then use this similarity matrix as input to a force directed calculation:

$$F_a = k_1 \times S_{ij} \times d$$

$$F_r = \frac{k_2 \times (1 - S_{ij})}{d}$$

in order to generate a behavioral similarity timeline plot. Here, F_a is the attractive force (Hooke's law), and F_r is the repulsive force (Coulomb's law). Coefficients k_1 and k_2 are chosen to guarantee that two nodes have very similar F_a and F_r when they reach the minimum distance d_{min}.

We compute a 1D force-directed layout for each time-step, and use this to lay out a behavioral line for each compute node. Similar behaviors bundle together, while outliers diverge. Like many force-directed approaches, we employ an iterative calculation based on Fruchterman-Reingold [29]. For the first timestep, the initial line heights are randomized. While this means the plots are not necessarily deterministic, the end results tend to look similar enough, without the need for more complex initialization. However, subsequent time-steps are initialized by reusing the previous time-steps' layout. This offers a good amount of stability to the layout, and also reduces the number of iterations that are needed. In order to further increase stability and account for gradual differences between the timesteps, we found it helpful to apply standardization to scale each time step, which is defined by subtracting the mean value and dividing by the standard deviation. This primarily helps keep the plot centered over time, and compensates for the drift due to extra iterations of force directed computation in later time steps. Excessively iterating the initial timestep could also potentially solve this drift problem, but would require greatly increased computation time.

The upper left view in Figure 1 is an example of such a plot. The same view appears in Figure 11. The y axis in this view is computed according to the similarity of the entities in each timestep. Thus, in this view, lines that run close together indicate entities that were similar in behavior over a certain period of time, such as compute nodes running identical code and working on the same task. Lines that branch off indicate anomalous entities, such as compute nodes that may be under/over-utilized for instance. In this manner, this view provides a succinct summary of the behavioral patterns across the whole system for a range of time. It can also indicate regions of the data that warrant further investigation in the detailed views.

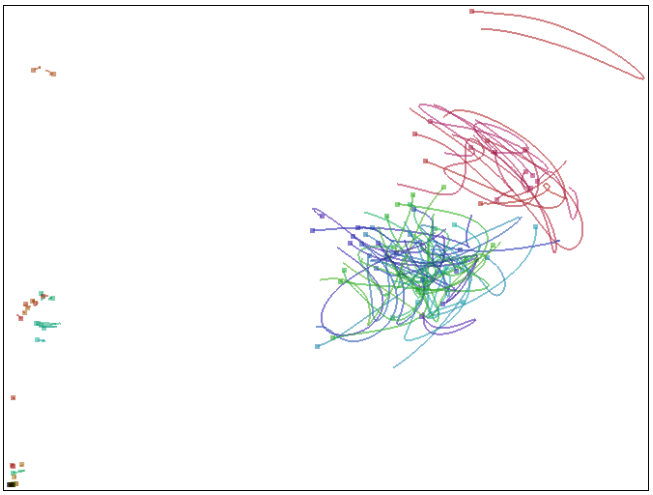


Fig. 7. Signal processing techniques can extract structure from the noisy data. Here we plot the windowed mean (on y) versus a windowed standard deviation (on x) of one data dimension as a line for each entity. Very clear clusters are visible along the left of the plot (low variance), while noisier clusters and outliers are to the right. Colors used here encode nothing but are used to distinguish lines.

As the number of lines grows, such behavioral line plots often create visual clutter, i.e., lines tend to overlap with one another. In order to improve the scalability and usability of such visualization, we provide a magnifying lens for the user to untangle the clutter. As shown in Figure 8(a), we can hardly see any detail inside the line bundle. With the magnifying lens(Figure 8(b)), lines are spread out and thus can be better distinguished, also making it possible to select lines from a bundle.

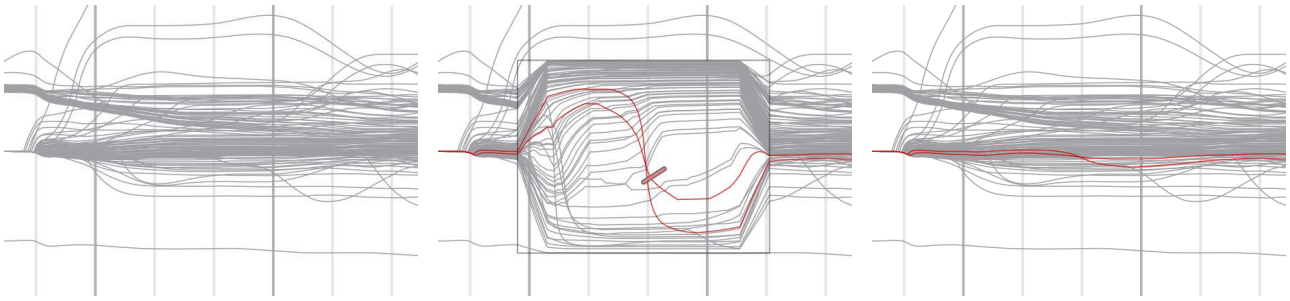
3.5 Detail Plots

The behavioral lines abstract away all the original dimensional information. However, when the user finds interesting patterns or outliers in the behavioral lines, the system should enable investigation into the detailed data. While naïvely rendering the raw data is not that effective, we can plot the smoothed statistical lines over time to depict the underlying multidimensional time series. To this effect, we implemented a *small multiples* view that plots the smoothed mean curve in each dimension for each selected group of compute nodes. That is, for each selection in the behavioral lines view, we populate a column of detail views, plotting the statistics over time for the corresponding row's dimension.

In order to visualize these statistics, we borrow techniques from uncertainty visualization [33]. Rather than just plotting the raw data or the smoothed mean curve, we plot the semitransparent range $[\mu_{i,d}(t) - \sigma_{i,d}(t), \mu_{i,d}(t) - \sigma_{i,d}(t)]$ over time for each selected node i. This can be seen in Figure 9, where one of these plots is blown up for closer view.

4 Case Studies

Here, we demonstrate visual analysis of performance profile data collected from two live cloud systems. One set of data is collected from a system with 476 nodes over a period of 2 weeks, at a 10 second resolution. And the other data is a high-resolution datasets are of a system of around 70 compute nodes, sampled at 2 second resolution for 2 periods of time: a 4 hour period and a 24 hour period. The metrics used in these datasets are shown in Table 1, and are colored according to the color map in Figure 2.



(a) (b) (c)

Fig. 8. magnifying lens (Cartesian distortion) applied to behavioral lines. Overlapping is severe in (a), we can not see any detail of the bundle. After applying the magnifying lens (b), individual lines are spread out for viewing and selection. The two red lines in (c) would normally be impossible to isolated.

Metric	476 node system	70 node system
CPU Utilization	√	√
Memory Idle	√	√
Network Lo		√
Network Download	√	√
Network Upload	√	√
Disk Read		√
Disk Write		√
Disk Utilization	√	
Load	√	

TABLE 1
Metrics collected from the two cloud systems for case studies.

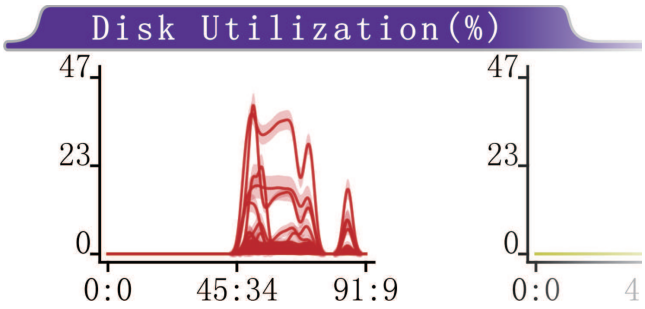


Fig. 9. The detailed view consists of an array of small multiples per dimension and per selection. Each plot renders semitransparent areas over time to convey the smoothed statistical metrics. The thickness of the area corresponds with the variance of the timeseries.

Figure 10(a) shows the full time range of the same collected dataset used in Figure 1. In this dataset, there are a number of interesting behaviors. The system is quite uniform at first, then the lines are split into 2 main behavioral clusters (the red one and the yellow one). From the detailed view, it is found that the strong split is because the compute nodes corresponding to the red lines surge in CPU, I/O, and memory usage about 10 hours before the yellow ones. Once the nodes corresponding to the yellow lines join in, there is a point where the two clusters almost rejoin, before remaining mostly distinct. This could indicate that the yellow nodes have a different system configuration, which allowed them to catch up and then surpass the red nodes. Also, we can easily find out two abnormal nodes (the cyan one and the green one), they are away from all other lines because their CPU utilization is much lower than others, as we can see from the detailed view. However, their standard deviations are relatively small, so we refer to the statistical line view, as shown in Figure 10(d), where the maximum standard deviation of both nodes (highlighted in Figure 10(b-c)) is lower than that of other nodes, indicating that both nodes act more stable than others. And we can also find out the two outliers easily from the statistical line view (Figure 10(d)).

Figure 11 shows a selection of the middle portion of the 4 hour dataset. For the most part, the lines bundle into clusters, indicating a smoothly running system, with an exception of the green nodes. From the detailed view, we can tell that the two green nodes are much like the yellow cluster; they may have just been trapped in the force directed layout for a while before self-correcting. For the red and yellow clusters, the primary difference between them is the CPU and memory usage, we can see that from the detail plots, both CPU and memory utilization of the red cluster are a little higher than the yellow one. It indicates that they may be working on different jobs or have a different hardware configurations. For the blue cluster, it is tightly bundled because all the nodes belong to this cluster seem to be idle, as shown in the detailed view.

The 24 hour dataset exhibits much more interesting behaviors, particularly since there is a gap in the middle of the dataset where the cloud system was inaccessible - possibly due to maintenance or an update. Figure 12(a) shows the timeline stacked chart of the overall system behavior. The gap is quite obvious, but there are a number of other notable spikes and behavior shifts. We explore some of these in the more detailed view.

Early in the overall time period, as we can see in Figure 12(b), the system is fairly constant, some nodes have periodic behavior, and some disk-heavy nodes suddenly drop to 0, which lead to a shift. Then, a little after 2:00, we can perceive a drastically change in the layout, which turns out to be caused by a spike in disk reads in the yellow and green clusters.

In Figure 12(c), at around 3:15 and 3:40, two valleys emerge for the red cluster due to CPU spikes. After that there comes the second spike, primarily affected by the green and yellow clusters, mostly hitting their CPU load. Fringe nodes, such as those in the cyan cluster, are pushed away, even though their pattern doesn't change.

Leading up to the outage, the 2 primary clusters exhibit more sporadic but synchronous behavior Figure 12(d), with the exception at 5:50, when each cluster diverges briefly in disk reads. For the outage, it is noteworthy that this affects all nodes seemingly equivalently, indicating that it is a system-originated effect, like a system-wide out of power, and not likely job or user originated.

After the outage (Figure 12(e)), the two primary clusters begin

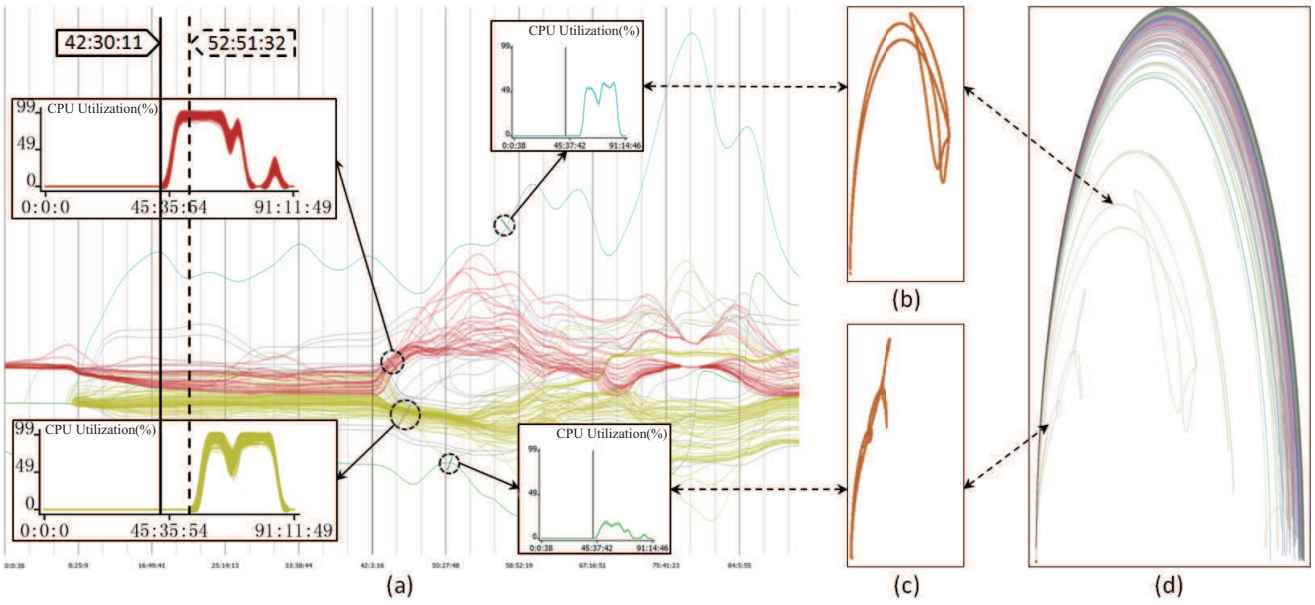


Fig. 10. The two week dataset. The behavior line view (a) shows an overall trend where one bundle splits into two, with a number of outliers. Investigating some of the outliers reveals that the underlying statistical lines (b-c) have very unique patterns, particularly with respect to the context of the entire dataset (d).

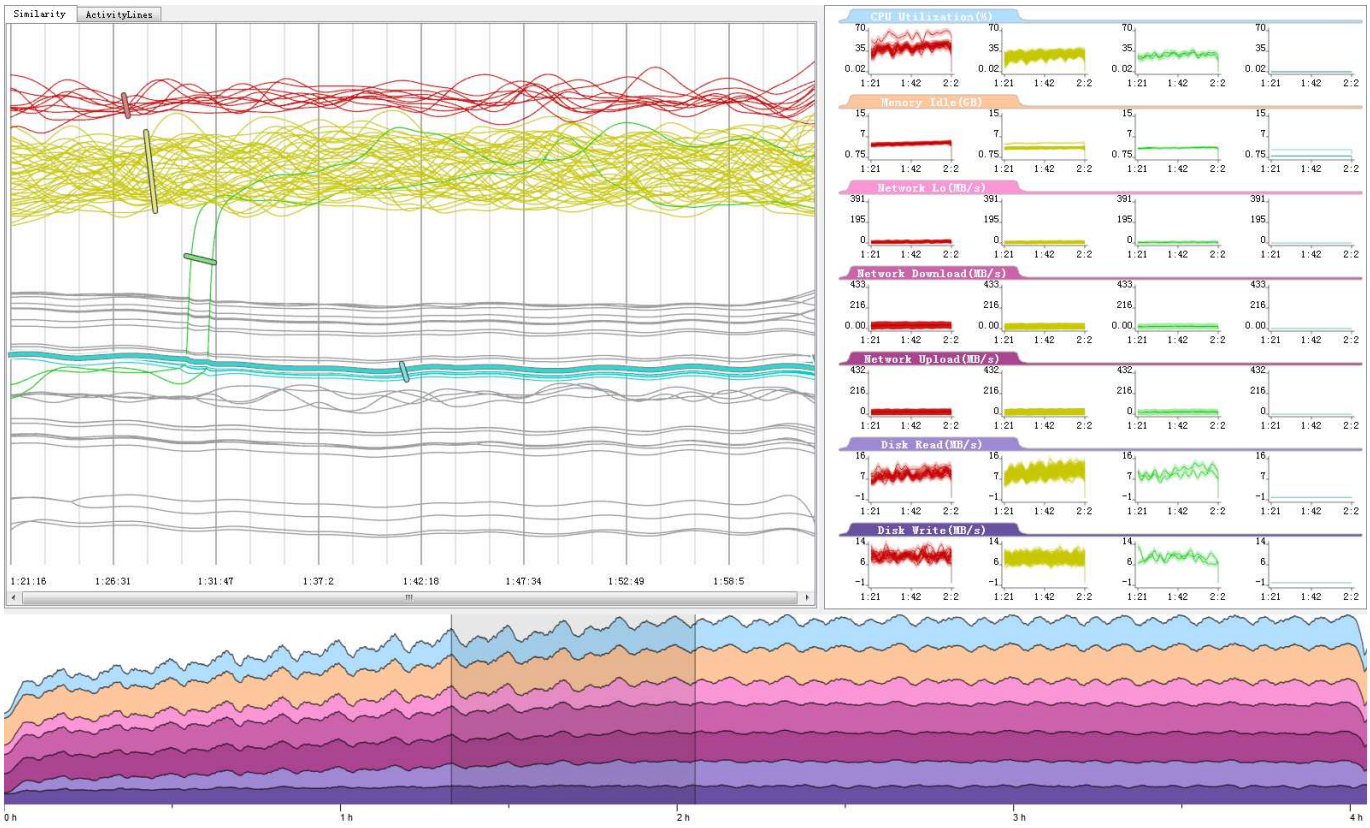
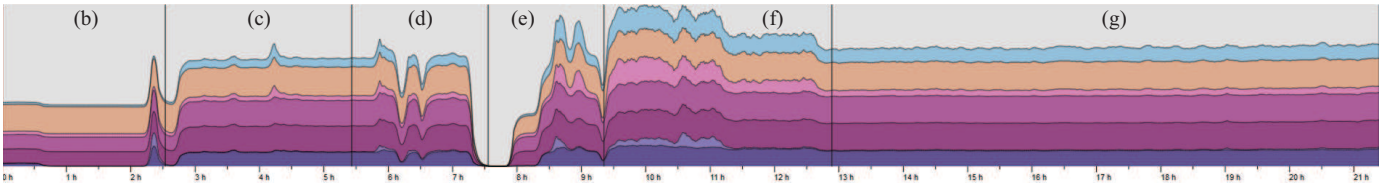
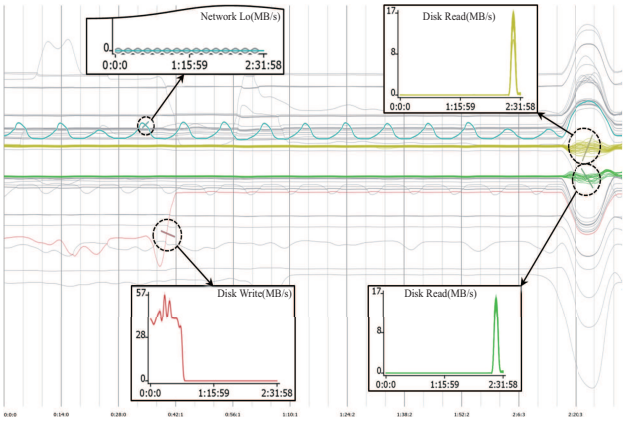


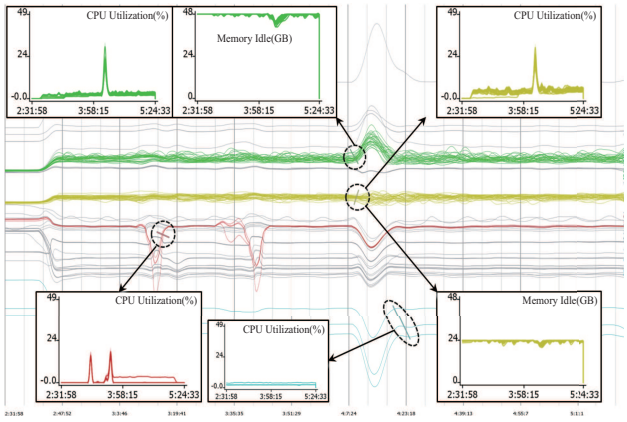
Fig. 11. A time-range selection from the 4 hour data set. The behavior is overall very stable, grouped into several clusters. The red group and the yellow group are very similar, with a little difference in CPU and memory usage. The two green nodes are much like and finally shift into the yellow group. The blue group is tightly bundled together, because those nodes seem to be idle and all metrics are very much the same.



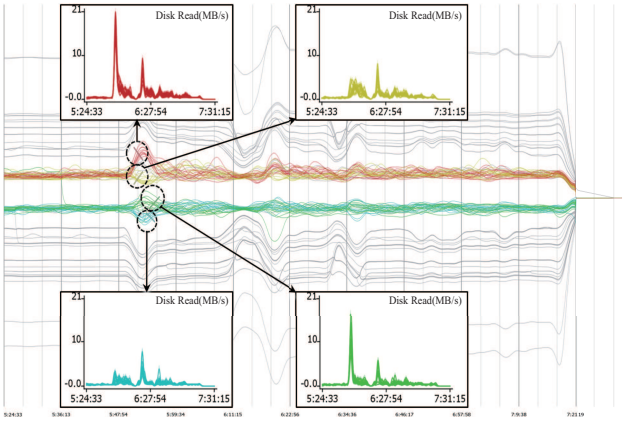
(a) 24 Hour Timeline



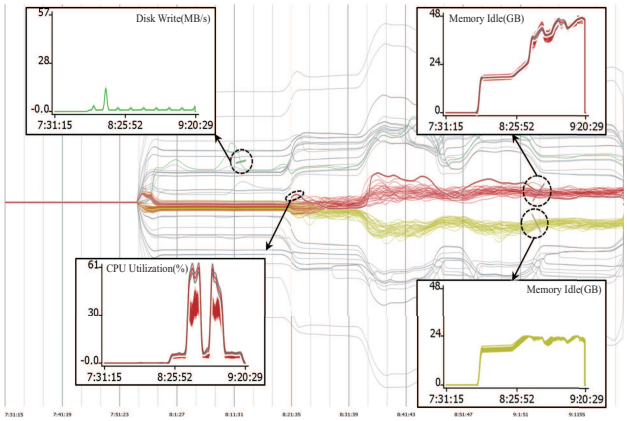
(b) Early in the day, some nodes have periodic behavior, and there are some disk-heavy nodes that suddenly drop to 0. Then, a little after 2:00AM, a spike in disk reads occurs in 2 clusters



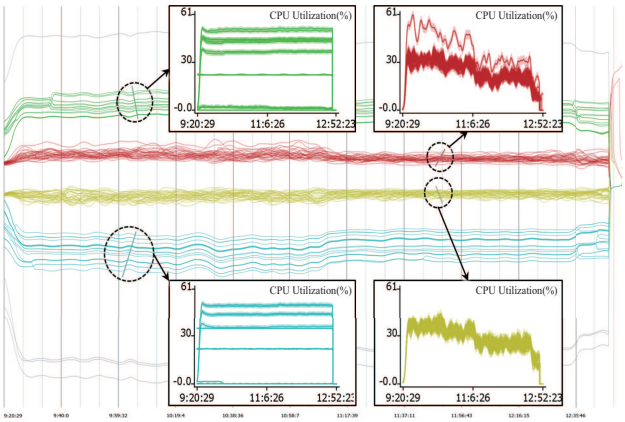
(c) Around 3:15 and again at 3:40, one minor cluster exhibits CPU spikes. At 4:10, the two major clusturs both exhibit spikes in CPU usage. Fringe nodes are pushed away, even though their pattern doesn't change.



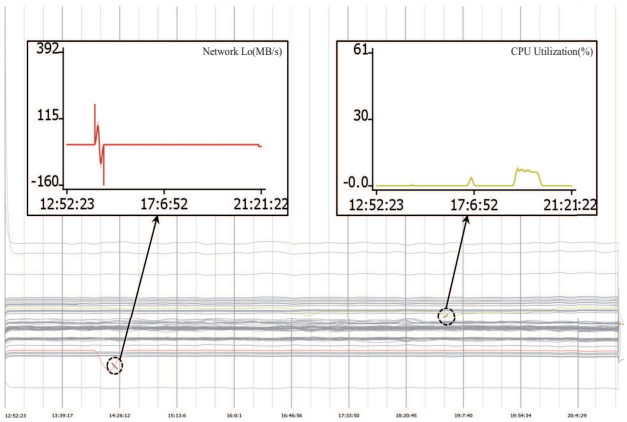
(d) Leading up to the outage, the 2 primary clusters exhibit more sporadic but synchronous behavior, with the exception of at 5:50AM each cluster diverges briefly in disk reads.



(e) After the outage, the two primary clusters begin synchronized, before splitting into two distinct groups based on memory usage. They also exhibit strong CPU spikes. Meanwhile, the fringe nodes exhibit more regular behavior, such as periodic small writes.



(f) Eventually, the system moves into a more synchronized pattern. While the two primary clusters are changing CPU load behavior over time, they remain in sync.



(g) Finally, the system settles into a steady state for the remainder of the time range, with the exception of a few small spikes in fringe nodes.

Fig. 12. A cloud computer system, over a 24 hour period with an outage in the middle. There are some significant events leading up to the outage, and a period of flux after the outage as the system returns to a steady state. Excerpts from the detail view are inset for space to describe the more interesting patterns.

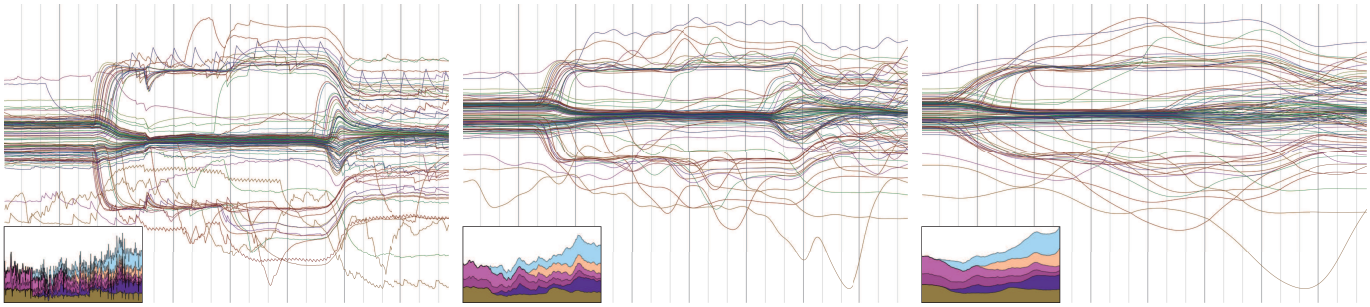


Fig. 13. The effect of different smoothing window sizes. From left to right, the smoothing window sizes are 0, 16, and 64, respectively. Color of the behavioral lines here encodes nothing but is used to distinguish from each other.

synchronized, before splitting into two distinct groups based on different memory usage. Outliers, the green node and the highlighted red node, can be easily recognized, because any anomaly in any metric of the node is exhibited in its behavior line.

Then the system moves into a more synchronized pattern (Figure 12(f)). While the two primary clusters are changing CPU load behavior over time, the changes synchronize well, leading to the stability of the layout.

Finally, the whole system settles into a steady state for the remainder of the time range (Figure 12(g)), with several anomalies in the red and yellow nodes.

5 Discussion and Future Work

The main contribution of our design is the presentation of a succinct overview of the whole cloud computation system, which provides effective indicators for the analysts to more easily locate system anomalies or bottlenecks. Nevertheless, it is possible for the overview to possibly miss some subtle events due to the similarity metric used. For example, two processing nodes could shift in different metrics such that the behavioral lines of these two nodes tend to be bundled together when they should not, but such occurrences are not likely. Allowing the users to assign different weights to different dimensions may reduce this problem.

Before we compute a force-directed layout for the behavioral similarity plot, we first need to smooth the original data, because the original, noisy data is not that useful, as shown Figure 6. The data used in the stacked graph timeline view is smoothed in the same way as the data used in the behavioral similarity plot view. The choice of the smoothing window size could be critical to the results; a small window size might not provide enough readability while a large window size could lose too much of the original information contained in the data. Figure 13, shows an exploration of the parameter space; from left to right, the window size is 0, 32, and 128, respectively. Line patterns better reveal themselves as the window size increases, but some features can become less clear or almost disappear in the extreme case. While this provided insight to find a decent default value, we also allow users to interactively adjust the window size in the stacked graph timeline view. After an interesting pattern is identified with one window size, the user can adjust the window size according to the corresponding time range, which is then applied to smooth the data used to compute the behavioral lines. In this way, our approach enables users to find a good window size to smooth the data contextually.

Our approach utilizes basic signal processing techniques. However, other applications could greatly benefit from combining our approach with more sophisticated signal processing techniques, such as wavelets or Fourier analyses. The force directed approach we implemented is rather naïve and inefficient, as it has to compute a full $O(n^2)$ repulsive forces, and it may encounter problems with local minima – particularly in 1D. More efficient layout approaches could be incorporated to increase the scalability to larger data sets. Dynamic clustering approaches could be incorporated to aid in this. We plan to develop a hierarchical force-directed algorithm with a time complexity of $O(n \log n)$ for faster layout computing to better be capable of handling streaming data. And when a line plot such as the one we have used grows too dense, additional visual summarization techniques could be explored. Lastly, while we have demonstrated some applications of our visual-based approach through case studies, a more extensive study that allows us to validate the findings experimentally would further prove the effectiveness of our approach.

6 Conclusion

Time varying multidimensional data commonly found in many Big Data applications can be very difficult to reason effectively. When the data is noisy, this challenge is greatly compounded. We have developed a new approach to visual summarization of such complex data in a way that is intuitive and simple. In particular, it enables visual analysis in a novel statistical space and has the potential to support streaming data analysis. We have demonstrated the visual analysis process with case studies on two datasets that existing techniques were ill equipped to handle. While these data sets were obtained from logging cloud computing, the applicability of our approach extends to innumerable other domains, and could greatly help in other complex, time critical situations. Although behavior plots may not present every facet of the data to the user in isolation, they provide good indicators to attract analysts' attention and they link well with more detailed or traditional visual analytic approaches.

Acknowledgments

This work was supported in part by the U.S. National Science Foundation through grant IIS-1320229, U.S. Department of Energy through grants DE-SC0007443 and DE-SC0012610, National 973 Program of China (2015CB352503), National Science Foundation of China (61232012 and 61422211), and the Key Natural Science Foundation of Zhejiang Province, China through grant LZ12F02002. The datasets used in this work were provided by Qihoo 360 Corporation and Northrop Grumman Corporation.

REFERENCES

[1] G. Andrienko, N. Andrienko, U. Demsar, D. Dransch, J. Dykes, S. I. Fabrikant, M. Jern, M.-J. Kraak, H. Schumann, and C. Tominski, "Space, time and visual analytics," *International Journal of Geographical Information Science*, vol. 24, no. 10, pp. 1577–1600, Oct. 2010.

[2] F. Giannotti and D. Pedreschi, *Mobility, Data Mining and Privacy: Geographic Knowledge Discovery*, 1st ed. Springer Publishing Company, Incorporated, 2008.

[3] J. A. Fails, A. K. Karlson, L. Shahamat, , and B. Shneiderman, "A visual interface for multivariate temporal data: Finding patterns of events across multiple histories," in *Proceedings of VAST 2006*, 2006, pp. 167–174.

[4] J. Gudmundsson, P. Laube, and T. Wolle, "Computational movement analysis," in *Springer Handbook of Geographic Information*, W. Kresse and D. M. Danko, Eds. Springer Berlin Heidelberg, 2012, pp. 423–438.

[5] J. F. JaJa, J. Kim, and Q. Wang, "Temporal range exploration of large scale multidimensional time series data," in *Proceedings of SSDBM*, 2004, pp. 95 – 106.

[6] S. Hidaka and C. Yu, "Spatio-temporal symbolization of multidimensional time series," in *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*, Dec 2010, pp. 249–256.

[7] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. Keogh, "Indexing multi-dimensional time-series with support for multiple distance measures," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD))*, 2003, pp. 216–225.

[8] M. Betke, J. Gips, and P. Fleming, "The camera mouse: visual tracking of body features to provide computer access for people with severe disabilities," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 10, no. 1, pp. 1–10, March 2002.

[9] M. Shimada and K. Uehara, "Discovering of correlation from multi-stream of human motion," in *Proceedings of the Third International Conference on Discovery Science*, 2000, pp. 290–294.

[10] M. Franciosi and G. Menconi, "Multi-dimensional sparse time series: feature extraction," *CoRR*, vol. abs/0803.0405, 2008. [Online]. Available: http://arxiv.org/abs/0803.0405

[11] C. Plaisant, S. Lam, B. Shneiderman, M. S. Smith, D. Roseman, G. Marchand, M. Gillam, C. Feied, J. Handler, and H. Rappaport, "Searching electronic health records for temporal patterns in patient histories: A case study with microsoft amalga," in *AMIA Annual Symposium Proceedings*, 2008, pp. 601–605.

[12] D. Gotz, F. Wang, and A. Perer, "A methodology for interactive mining and visual analysis of clinical event patterns using electronic health record data," *Journal of Biomedical Informatics*, vol. 48, no. 0, pp. 148 – 159, 2014.

[13] Y. Mohammad and T. Nishida, "Mining causal relationships in multi-dimensional time series," in *Smart Information and Knowledge Management*, ser. Studies in Computational Intelligence, E. Szczerbicki and N. Nguyen, Eds. Springer Berlin Heidelberg, 2010, vol. 260, pp. 309–338.

[14] S. S. Shende and A. D. Malony, "The Tau parallel performance system," *Int. J. High Perform. Comput. Appl.*, vol. 20, pp. 287–311, May 2006.

[15] W. E. Nagel, A. Arnold, M. Weber, H.-C. Hoppe, and K. Solchenbach, "Vampir: Visualization and analysis of MPI resources," *Supercomputer*, vol. 12, pp. 69–80, 1996.

[16] O. Zaki, E. Lusk, W. Gropp, and D. Swider, "Toward scalable performance visualization with Jumpshot," *Inernational Journal of High Performance Computing Applications*, vol. 13, pp. 277–288, Aug. 1999.

[17] C. Muelder, F. Gygi, and K.-L. Ma, "Visual analysis of inter-process communication for large-scale parallel computing," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 1129 –1136, Nov.-Dec. 2009.

[18] C. Muelder, C. Sigovan, K.-L. Ma, J. Cope, S. Lang, K. Iskra, P. Beckman, and R. Ross, "Visual analysis of I/O system behavior for high-end computing," in *Proceedings of the third international workshop on Large-scale system and application performance (LSAP)*, 2011, pp. 19–26.

[19] C. Sigovan, C. Muelder, and K. Ma, "Visualizing large-scale parallel communication traces using a particle animation technique," *Computer Graphics Forum*, vol. 32, no. 3, pp. 141–150, 2013.

[20] C. Sigovan, C. Muelder, K. Ma, J. Cope, K. Iskra, and R. B. Ross, "A visual network analysis method for large-scale parallel I/O systems," in *Proceedings of the 27th IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*, 2013, pp. 308–319. [Online]. Available: http://dx.doi.org/10.1109/IPDPS.2013.96

[21] W. Spear, A. D. Malony, C. W. Lee, S. Biersdorff, and S. Shende, "An approach to creating performance visualizations in a parallel profile analysis tool," in *Proceedings of the 2011 international conference on Parallel Processing - Volume 2*, ser. Euro-Par'11, pp. 156–165.

[22] E. Shaffer, D. Reed, S. Whitmore, and B. Schaeffer, "Virtue: performance visualization of parallel and distributed applications," *Computer*, vol. 32, no. 12, pp. 44 –51, dec 1999.

[23] M. Ogawa and K.-L. Ma, "Software evolution storylines," in *Proceedings of the 5th International Symposium on Software Visualization (SoftVis)*, 2010, pp. 35–42.

[24] C. Muelder, T. Crnovrsanin, A. Sallaberry, and K.-L. Ma, "Egocentric storylines for visual analysis of large dynamic graphs." in *Proceedings of IEEE BigData 2013*, pp. 56–62.

[25] K. Reda, C. Tantipathananandh, A. Johnson, J. Leigh, and T. Berger-Wolf, "Visualizing the evolution of community structures in dynamic social networks," in *Proceedings of the 13th Eurographics/IEEE-VGTC Conference on Visualization (EuroVis)*, 2011, pp. 1061–1070.

[26] Y. Tanahashi and K.-L. Ma, "Design considerations for optimizing storyline visualizations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2679–2688, 2012.

[27] T. Crnovrsanin, C. Muelder, C. Correa, and K.-L. Ma, "Proximity-based visualization of movement trace data," in *Proceedings of IEEE VAST 2009*, Oct, pp. 11–18.

[28] P. Eades, "A heuristic for graph drawing," *Congressus Numerantium*, vol. 42, no. 11, p. 149160, 1984.

[29] T. M. J. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," *Software: Practice and Experience*, vol. 21, no. 11, pp. 1129–1164, Nov 1991. [Online]. Available: http://dx.doi.org/10.1002/spe.4380211102

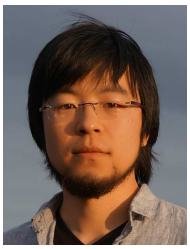
[30] Z. Zhang, K. T. McDonnell, and K. Mueller, "A network-based interface for the exploration of high-dimensional data spaces," in *Proceedings of IEEE Pacific Visualization Symposium (PacificVis)*, 2012, pp. 17–24.

[31] H. Qu, W.-Y. Chan, A. Xu, K.-L. Chung, K.-H. Lau, and P. Guo, "Visual analysis of the air pollution problem in Hong Kong," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1408–1415, 2007.

[32] H. J. Blinchikoff and A. I. Zverev, *Filtering in the time and frequency domains*. New York: Wiley, 1976, a Wiley-Interscience publication. [Online]. Available: http://opac.inria.fr/record=b1084177

[33] A. Dasgupta, M. Chen, and R. Kosara, "Conceptualizing visual uncertainty in parallel coordinates," *Computer Graphics Forum*, vol. 31, no. 3, pp. 1015–1024, June 2012.

Chris Muelder received his PhD in computer science from the University of California at Davis in 2011. He then worked as a postdoctoral researcher with the VIDI Labs at UC Davis during 20112014. He has served on both the EuroVis and InfoVis program committees. His research interests include information visualization, visual analytics, and graph layout and visualization. He is presently with Google as a software engineer.



Biao Zhu is a PhD student at the State Key Laboratory of CAD & CG, Zhejiang University. His current research focuses are information visualization and visual analytics.

Wei Chen is a professor at the State Key Lab of CAD & CG, Zhejiang University. Professor Chen received his PhD from the Zhejiang University in 2002. His research interests include visualization, visual analytics, and biomedical image computing. He has published more than 60 papers in international journal and conferences. Professor Chen served as Papers Co-Chair of IEEE PacificVis 2013 and Conference Chair of IEEE PacificVis 2015. He presently serves on the steering committee of IEEE PacificVis.

Hongxin Zhang is an associate professor of the State Key Laboratory of CAD & CG, Zhejiang University. He received his Ph.D in Applied Mathematics from the Zhejiang University in 2002. His research interests include geometric modeling, visual analytics, and machine learning.

Kwan-Liu Ma, an IEEE Fellow, is a professor of computer science and the chair of the Graduate Group in Computer Science (GGCS) at the University of California, Davis. He leads VIDI Labs and and the UC Davis Center for Visualization. Professor Ma received his PhD degree in computer science from the University of Utah in 1993. His research interests include visualization, high-performance computing, and user interface design. Professor Ma was a recipient of the NSF PECASE award in 2000, and the IEEE VGTC 2013 Visualization Technical Achievement Award. He has served as a papers chair for SciVis, InfoVis, EuroVis, and PacificVis, and also as an associate editor of IEEE TVCG (20072011) and the Journal of Computational Science and Discoveries (20092014). He is a founder of PacificVis, Ultravis, and LDAV. Professor Ma presently serves on the editorial boards of the IEEE CG&A and the Journal of Visualization.