

Nodescape: Finding Interesting Behavior

J. Frank Roberts
jfrobe2@engr.uky.edu
December 14, 2012

1 Introduction

The Nodescape project has always been about making it easier for humans to monitor large scale computer systems. The first version of Nodescape accomplished this goal primarily by using novel presentation techniques. Continued development of Nodescape has shown us that these presentation techniques by themselves are not enough to handle the scale of modern datacenters and compute clusters. Recent development of Nodescape has focused on using software to figure out which parts of the information that we are collecting are actually interesting. We have found that this problem has two parts. First, we must develop a method for detecting anomalies in the data we are collecting. Second, we must figure out what sorts of anomalies actually qualify as anomalous in the context of computer systems monitoring.

2 Previous Work

2.1 Nodescape v1

Our initial work on Nodescape focuses on finding ways to present monitoring data so that interesting properties of the data are apparent to a viewer. The first version of Nodescape [1] (**Nodescape v1**) presents measurements by painting colors on the nodes in an image of a cluster. Warmer colors correspond to higher numbers, and cooler colors correspond to lower numbers. Nodescape v1 represents old data by speckling a node magenta (magenta is not in the normal color progression) as the respective measurement ages. The viewer may easily see from this presentation which measurements are high or low relative to the rest of the cluster, as well as how old the measurements are. Nodescape v1 does almost no analysis of the data, requiring the viewer to know what an interesting set of measurements looks like.

The presentation format used in Nodescape v1 works well for observing the values of a particular property across the nodes in a cluster at a particular moment in time, but it is not useful for observing trends or detecting abnormal behavior. Scale also presents a problem for Nodescape v1. Visually scanning the painted image of a 100 node cluster is reasonable, but this presentation would not work as well for a cluster of several thousand nodes. We have begun a second phase of research that looks for ways to overcome these deficiencies by automatically analyzing the data.

2.2 Nodescape v2

We began the second phase by reimplementing Nodescape to be modular. **Nodescape v2** [2] separates the collection and storage of data from the analysis and presentation of data. The Nodescape v2 **back-end** handles the collection and storage of data, and is comprised of two components. The **monitor** runs periodically on each machine that being monitored. Each time it runs, the monitor reads a list of commands to be executed. Each command measures some property and prints the measurement to standard output. The monitor runs these commands, captures the output, and sends the output, with

the corresponding property label, to the collector. The **collector** listens for updates from monitors. When the collector receives an update, it unpacks and stores the update in a **MySQL database**. Software handling the analysis and presentation of the data pull the data out of the MySQL database. This implementation allows us to share the same **back-end** across many different front-ends. A Nodescape v2 **front-end** is a tool which analyzes data from the MySQL database and presents it to the user.

We have implemented one front-end for Nodescape v2, called **PFE** (Plotting Front-End). For each **host-property pair** in the database, PFE calculates the mean and standard deviation over a configured time interval. If there are samples in the configured time interval which fall outside a configurable number of standard deviations, then that host-property pair is marked as containing an anomaly. For each anomalous pair, PFE plots the data over time intervals of 2 hours, 1 day, and 4 days. PFE also generates a plot over a configurable time interval.

Finally, PFE generates an HTML page containing a summary of each anomaly. The summary includes the mean, standard deviation, the sample value farthest from the mean, and the number of standard deviations at which that value falls. The HTML page includes one plot for each anomaly. Three hyperlinks appear below each summary. The first link points to a page displaying plots for all properties on the host from the anomalous pair. The second link points to a page displaying plots for all hosts for the property from the anomalous pair. The third link points to a page displaying all four plots for the anomalous host-property pair.

The PFE summary page is generated by a script which runs regularly. All linked pages are generated dynamically when they are requested. The interface is generally responsive, though the dynamically produced pages take several seconds to generate if they contain more than a handful of plots.

One goal for the development of PFE was to gain a better understanding of how to work with the MySQL database, and to gain a better understanding of how to manipulate the data. PFE was developed in an incremental fashion, without a clear technical design. The quality of the presentation suffers as a result. The plots are not well labeled, and the scale on the Y-axis often does not fit the data. These problems will not be difficult to fix in future front-ends, but the internal structure of PFE is such that fixing these problems requires a non-trivial redesign.

2.3 Conclusions from Nodescape v2

PFE also has deficiencies in the quality of its analysis. We are running the monitor on approximately eighty machines, gathering data on roughly five properties per machine. This means there are roughly four hundred host-property pairs. PFE usually marks between forty and sixty of these pairs as containing an anomaly. Upon inspection, rarely do we find that more than ten of these pairs actually demonstrate anomalous behavior.

Many of the plots marked as anomalous show extremely consistent behavior, with perhaps two instances where the value of the sample varies by a small amount. Because most of the samples in the series are actually the same value, samples that vary by even one unit may fall several standard deviations outside the mean. PFE detects the high standard deviation, and labels these pairs as containing anomalies, even though there is

nothing of interest in the series of samples.

We believe we are dealing with two problems here. First, using only standard deviation on a single series is not a good way to detect anomalies. Second, we have not constructed a good definition of what an anomaly is in the context of monitoring computer systems. We are now beginning a third phase of research, in which we are focusing on defining what it means for there to be an anomaly in computer monitoring data, and on how to detect anomalies in our data.

3 Anomaly detection

3.1 What is an anomaly?

Many computer monitoring systems rely on the administrator to determine what sorts of behavior qualify as interesting. We see two problems with this approach. First, it does not scale. If the data are presented well, a systems administrator might reasonably be able to inspect the health of up to a few hundred machines without the aid of automated analysis. However, datacenters and compute clusters often contain several thousand machines.

Some monitoring systems allow the administrator to set up alerts to be triggered when data fall outside of an acceptable range. This approach helps with the problem of scale, but is inflexible and requires the administrator to know reasonable bounds for each property for every machine in the system. Also, as we have already seen in our attempt to detect anomalies using standard deviation, just because a series contains a value outside a certain range does not mean that the series is interesting.

The second problem we see is that the administrator does not always know what behavior is interesting. For example, the number of logged in sessions on the machine `coreopsis.cs.uky.edu` varies significantly from one hour to the next, but it follows a pattern that repeats itself approximately every twenty four hours. Although the number of sessions is changing constantly, this series really is not interesting unless there is a break from the normal pattern. Even if the administrator understands that a series is not interesting just because it changes, they may not be able to identify variations from the normal pattern.

Both of these problems could be solved by using software to analyze and detect anomalies in our data. Modern computers are easily capable of quickly analyzing data from several thousand machines. There are also many anomaly detection techniques used in other fields that could be applied to machine health monitoring.

In order to detect anomalies in our data, we must first define what constitutes an anomaly. The detection algorithms we choose will partially determine this definition. However, behavior that is anomalous in general may not actually qualify as interesting in the context of computer monitoring. We must determine which anomalies we actually find interesting. We hope to develop an answer to this question during this phase of our research.

3.2 Anomaly Detection Techniques

We have begun preliminary investigation of several techniques. PFE implements an analysis using standard deviation. The method used in PFE is not effective for detecting anomalies, but we believe we may be able to design an improved analysis using standard deviation. We have also considered transforming our time series' to the frequency domain so that we can use frequency domain analysis techniques on our data. Our research on this method is not yet mature enough to conclude whether or not this technique is effective. Finally, we are researching an analysis method which converts the time series to a symbolic representation using **Symbolic Aggregate appRoXimation** (SAX) [3], and then performs analysis on the symbolic representation. We have more confidence that this method will work than we have for any other method we have considered. We will research and evaluate these techniques and some others.

3.2.1 Standard Deviation

We will design a more sophisticated technique for anomaly detection based on standard deviation. The technique used in PFE compares each series only to itself. We have observed that anomalous behaviors often correlate across properties, so our method will look for anomalous values to occur at the same time in multiple properties. We will also look for changes that occur on several hosts at the same time.

The challenge in designing this technique will be determining how to group properties and hosts. One obvious method for grouping machines is to group machines based on the group name supplied to the Nodescape back-end. For some Nodescape groups, this would work well. For example, all of the nodes in the cluster `cik` belong to the `.cik` group. These machines are physically all in the same environment. They are also part of a compute cluster, so they often are performing similar or even identical tasks at the same time. However, analysis across the `.kaos` group, which is comprised of both workstations and servers in several different locations, would not work well, as the events on one machine have little to do with the events on any other.

We may be able to determine a good grouping for properties through trial and error. For example, we have observed from PFE that memory usage, session count, and process count tend to behave similarly. We will try several manual groupings and evaluate how well they work. Ideally, we will find an automated method to group hosts and properties.

3.2.2 SAX Representation

Wei *et al.* [3] present a method for anomaly detection in time series which converts the series into a symbolic representation [4]. The transformation to the SAX representation is performed by first breaking the series into many equal length sections and finding the mean of each section. Next, each section is replaced with its mean. Finally, assign each section a symbol; symbols should be assigned such that they are equi-probable. The alphabet from which the symbols are chosen is small, often only four symbols. This assignment generates a SAX word. The SAX word is used to build a time series **bitmap**, which is a grid arrangement containing frequency counts for different sub-words of the

SAX word. The bitmap may be compared against other bitmaps; the distance between two bitmaps is defined as the sum of the squared euclidean distance between each corresponding pair of pixels in the two bitmaps.

To detect anomalies, two windows, a **lead** window and a **lag** window, are moved across the series together. The lead window covers the new, or not yet analyzed, part of the series; the lag window covers a part of the series which has already been analyzed by the lead window. Both windows are converted to the SAX representation, and the distance between the two bitmaps is computed. The farther the lead bitmap is from the lag bitmap, the more anomalous the part of the series in the lead window.

We can use this method for anomaly detection. We will use data that contains no anomalies to establish a baseline set of SAX words, and then use that set to analyze data from other hosts. We may also be able to compare SAX words across properties for some properties.

3.2.3 Frequency Domain Analysis

We have an implementation of an algorithm to find the **discrete Fourier transform** (DFT) of a time series. We have plotted the the DFT for some host-property pairs, but we do not have the expertise necessary to interpret the plots. Further research is needed to determine whether this method is viable for anomaly detection.

3.3 Evaluating Anomaly Detection Techniques

We will design a method for evaluating the effectiveness of the different anomaly detection techniques we implement. We will construct a set of test data series. Some of the tests will be synthetic, that is, the series will be artificially generated, and will contain specific anomalies at known positions in the series. We will also attempt to gather test data series which contain real anomalies. We will do this by paying close attention to our machines and to the PFE interface. When we notice an anomaly, we will pull the data for the anomalous host-property pair from the database and add it to our set of tests. A third way that we can collect test data is by artificially causing an anomaly or failure on one of the systems being monitored.

We want to have test series for a number of different situations. We want to detect a strong spike in CPU temperature as an anomaly, and we also want to detect a slow rise in CPU temperature as an anomaly. We will include test data sets for both anomalies. We will cause artificial anomalies by changing system usage so that properties like load average and memory usage are significantly affected. We can manipulate CPU temperature by changing fan speeds, by changing the environment in which a machine is situated, and by heavily loading the CPU. These tests are not the only situations we want to analyze; they only serve to show how we will build our set of tests.

We will evaluate our anomaly detection techniques by running them on our test data. Each technique will generate an anomaly score for each part of the series being analyzed. An anomaly score may be assigned to each point in the series, or we may only assign a score for each section of a series. We will look at a plot of this score, and check to see how much it rises around the anomalies in the test series. We will also check to see that

the anomaly score remains stable and low for parts of the series which do not contain anomalies.

4 Conclusion

We have developed two versions of the Nodescape monitoring system. We began this research with a focus on how to present monitoring data, and we expanded that focus to include analysis of data in our second implementation. We have found that the fundamental problem in analyzing computer system monitoring data is figuring out what behaviors are interesting. We have begun researching several techniques for anomaly detection in time series. We will implement and refine these techniques. We hope to eventually use them in future Nodescape front-ends.

References

- [1] The Aggregate's Nodescape Utility. <http://aggregate.org/NODESCAPE>. 2012.
- [2] Nodescape. <http://super.ece.engr.uky.edu:8092/nodescape/>. 2012.
- [3] Wei, Li and Kumar, Nitin and Lolla, Venkata and Keogh, Eamonn and Lonardi, Stefano and Ratanamahatana, Chotirat Ann. (2005), Assumption-Free Anomaly Detection in Time Series. Proceedings of the 17th International Conference on Scientific and Statistical Database Management: 237-242.
- [4] Lin, Jessica and Keogh, Eamonn, and Lonardi, Stefano, and Chiu, Bill. (2003), A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. Proceedings of the 8th ACM SIGMOD workshop on research issues in data mining and knowledge discovery: 2-11.