Nonparametric Multivariate Anomaly Analysis in Support of HPC Resilience[*]

G. Ostrouchov, T. Naughton, C. Engelmann, G. Vallée, and S. L. Scott
Computer Science and Mathematics Division
Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA
{ostrouchovg,naughtont,engelmannc,valleegr,scottsl}@ornl.gov

Abstract

Large-scale computing systems provide great potential for scientific exploration. However, the complexity that accompanies these enormous machines raises challeges for both, users and operators. The effective use of such systems is often hampered by failures encountered when running applications on systems containing tens-of-thousands of nodes and hundreds-of-thousands of compute cores capable of yielding petaflops of performance. In systems of this size failure detection is complicated and root-cause diagnosis difficult. This paper describes our recent work in the identification of anomalies in monitoring data and system logs to provide further insights into machine status, runtime behavior, failure modes and failure root causes. It discusses the details of an initial prototype that gathers the data and uses statistical techniques for analysis.

1. Introduction

The computational power of today's high-performance computing (HPC) systems provides enormous opportunities for scientific exploration. As large-scale HPC platforms increase in size and performance, their complexity grows as well. This increased complexity raises significant challenges for both, users and operators. These systems require great care and attention, much of which is due to a rise in failures caused by increased node/component counts.

Fault tolerance, or resilience, is a key challenge for computing and a major factor in the successful utilization of high-end scientific computing platforms. In these systems, identifying failures and diagnosing their root cause are non-trivial tasks. As part of our work in resilience we have begun to explore data gathering and analysis techniques to aid in the identification of issues, or anomalies, that may be sources of failure.

With this paper, we describe our initial work in gathering the necessary system monitoring and log data, and in using statistical methods for processing the collected data to identify anomalous events. The paper is structured as follows. Section 2 discusses related background and highlights key challenges associated with the problem of large-scale system analysis. Section 3 introduces the technical approach. Section 4 describes the developed prototype and presents results from a case study. Section 5 concludes the paper with a summary and a discussion of future work.

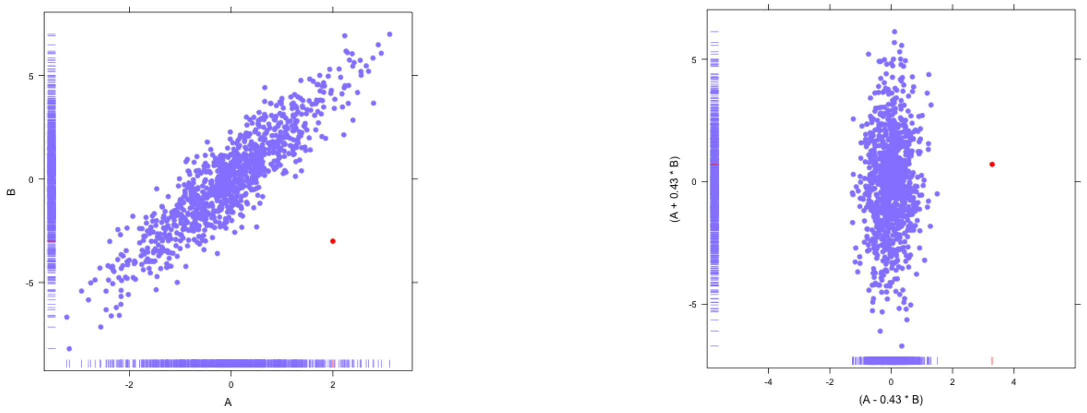
2. Background

There are several challenges when analyzing failures in large-scale systems. The system scale itself represents an issue related to *data quantity* as well as associated *sampling rates*. Note that in some instances the gathering of such data can cause the monitoring environment itself to fail due to overload, which may result in significant portions of the data being lost. The *performance requirements* of large-scale HPC systems also demand that the monitoring techniques do not perturb application execution to avoid cascading system noise issues. *Data access* is another problem as machine failure data is often considered business sensitive information and making it publicly available may require to sanitize/anonymize the data for release. Lastly, *data quality* varies significantly and in some instances the data available for study is either ill-formed or erroneous.

There are several efforts that aim to collect system data and potentially identify anomalies, notably Sisyphus [12], Ovis [1, 2], and CIFTS [7].

Sisyphus is a tool for anomaly detection in system log files that uses ideas from text analysis. Its central concept is to construct a document-term matrix, where the documents are node-hour partitions of the system

[*] This research is sponsored in part by the Department of Defense and the Extreme Scale Software Center at the Oak Ridge National Laboratory, and also by the Office of Advanced Scientific Computing Research, U.S. Department of Energy. The work was performed at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC under Contract No. De-AC05-00OR22725.



(a) The anomaly is not detected in the individual attributes.

(b) After a multivariate transformation the anomaly is detected in one of the transformed attributes.

Figure 1. Scatter and individual rug plots for two attributes (A,B) with a red anomaly point.

log. Then, the entropy value of term distribution within node-hours is used to identify anomalous log file events. Ovis provides a mechanism to collect numerical data from system nodes at regular intervals and a convenient display of the data mapped onto a representation of the system architecture. Anomalies are identified in several ways, including univariate attribute comparison to a learned reference, Bayesian parameter estimation (univariate and bivariate), and Mahalanobis based multivariate distance. Finally, CIFTS is a project that is developing a prototype of a scalable data collection backplane in a hierarchical structure of agent modules. The data will be potentially made available to various plug-in modules for analysis, monitoring, and feedback.

3. Technical Approach

The anomaly analysis presented in this paper is based on the same kind of data that Ovis collects and analyzes. In fact, we use a small part of the Ovis data collection infrastructure. We also reuse the node-hour partitioning concept from Sisyphus. However, we provide a different kind of anomaly detection analysis that is based on a nonparametric multivariate approach. It is multivariate to account for attribute dependence relationships and it is nonparametric to handle nonlinearities in this dependence.

While most anomalies may be detected in individual attributes, multivariate methods provide greater sensitivity due to their ability to account for dependence between attributes. For example, consider the scatter plot in Figure 1 (a) with two attributes A and B that display positive correlation. The individual points may be one-minute averages of processor utilization (A) and processor temperature (B). Individually, within the A distribution and the B distribution (shown by the rug plot along the axes) the red data point does not appear anomalous. However, due to the correlation structure between A and B, the red point is clearly anomalous. Detecting this anomaly requires either a multivariate approach within Figure 1 (a) or an appropriate multivariate transformation of the two variables into two new uncorrelated variables $A+0.43B$ and $A-0.43B$, shown in Figure 1 (b). In either case, a multivariate statistical technique is required. This same concept extends to higher dimensions and higher than two-way dependence.

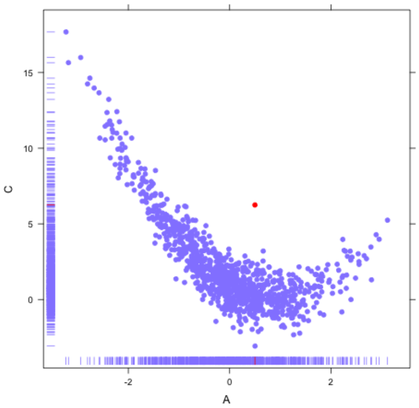


Figure 2. A nonlinear attribute relationship with an outlier that is not detectable with linear and globally Gaussian methods.

When the attribute dependence relationship is nonlinear, as between attributes A and C in Figure 2, a linear multivariate transformation can no longer provide a new attribute that detects the anomaly. At the same time, the implicit global Gaussian assumption of Mahalanobis distance would prevent anomaly detection in such nonlinear situations. The extreme points in the main cluster of Figure 2 have a greater Mahalanobis distance than the outlying red point. Such situations are handled well by nonparametric methods that use a local definition of density.

Our approach uses single-linkage clustering, which is also known as minimum spanning tree clustering [6]. While this is a very old technique, it continues to be subject of much active research [13]. It is nonparametric since no model is assumed for the clusters. Clusters are assembled based on nearest neighbor distances. Nearest neighbor techniques can be thought of as nonparametric multivariate density estimators since distance to neighbors is inversely related to local density.

4. Anomaly Analysis System

We implemented a proof-of-concept anomaly analysis prototype for large-scale HPC systems that receives information on system state about individual components and identifies anomalous component behavior, e.g., *configuration*, *performance*, or *failure* anomalies. It also provides the ability to view groups of components as statistical distributions in any computed attribute space. The developed solution primarily demonstrates that there is sufficient information available to determine some actionable anomalies among a large number of components with standard multivariate analysis techniques. If in addition good, i.e., accurate enough, failure data is available, the multivariate analysis techniques can be tuned to provide an actionable failure prediction capability, which in-turn allows anticipatory reconfiguration, such as migration of computation away from components that are about to fail [4].

4.1. Analysis Techniques

To manage the anomaly detection problem size, we select a window of interest and data granularity. For example, we can consider node-hours over a day of HPC system operation and identify those node-hours that are most different from the rest. Depending on the reaction time required and the type of analysis that is needed, it may be appropriate to consider node-seconds over the last minute or the last hour.

Given a window of interest and granularity, we make attribute selections based only on statistical properties, without injecting any expert system knowledge. A scalable implementation of our system is possible as it relies only on statistical properties of simple data summaries. Its power comes from the ability to examine high-dimensional nonlinear relationships, such as the correlation shown in Figures 1 and 2.

The developed anomaly analysis prototype is based on the R Project for Statistical Computing [11] and on the GGobi visualization software for exploring high-dimensional data [5]. Broadly, we provide the capability to identify anomalous components and anomalous time periods by applying clustering techniques and by exploring linked projections of high-dimensional representations. The clustering methodology provides automated anomaly identification while the high-dimensional visualization provides a powerful interactive exploratory tool for system data after clusters are identified. Our anomaly analysis is a capability different from the tools discussed in Section 2 (Sisyphus, Ovis, and CIFTS) and it can both benefit from these tools as well as extend their functionality. It is implemented via the following steps:

- We collect raw attribute data at an appropriate frequency and over a selected time window.
- Node-period data is then aggregated from the raw data granularity using standard statistical functions, e.g., means over time, variance over time, etc. The aggregated data forms a matrix whose rows are node-periods and columns are the attributes.
- The matrix columns are standardized to have zero mean and variance one. Constant columns are discarded as they do not contribute to the analysis. The standardization is required to compensate for different units of measurement of different attributes. For n nodes and p periods over m retained attributes, the node-period data forms an $np \times m$ matrix X.
- The node-periods and attributes are clustered via single-linkage hierarchical clustering, giving a dendrogram on the node-periods. Hierarchical clustering allows the exploration of all cluster configurations by cutting the dendrogram at various heights.
- A given dendrogram cut resulting in k clusters, partitions X into k sets of rows. For each k, a node-period *anomaly measure* can be computed from the size of the cluster to which it belongs. Although we can have $1 \leq k \leq np$, typically it is not useful to visualize k much larger than 10.
- We then display this information using different visualization techniques, e.g., node-hour dendograms, high dimensional data vizualizers, etc.
- It is also possible to cluster X^T, the transposed matrix, producing diagnostic information on the selected metrics. Those metrics that provide similar information are likely to be in the same cluster.

Intuition suggests that an anomaly measure should be high for small clusters and low for large clusters. Given a set of k clusters of sizes $n_1, n_2, \ldots, n_k$, where $\sum_{i=1}^{k} n_k = np$, we propose two possibilities. A simple inverse realtionship

$$\alpha_i = \frac{1}{n_i},$$

which gives values between 0 and 1 and is the measure used in our case study evaluation in this paper.

A better anomaly measure is formed by using the counting measure to define a cluster size probability. This is reasonable under the assumption that node-periods are sampled from some unknown distribution so that the number that falls into a cluster is proportional to the probability of that cluster (the integral of the unknown distribution over the cluster region). To compute how unusual is a given small cluster size n, we add the probability of all equal or smaller clusters. That is

$$p_i = \frac{1}{n} \sum_{n_j \leq n_i} n_j.$$

Then, converting to increase from 0 to 1 we have

$$\alpha_i = 1 - p_i$$

as a probability based measure of anomaly.

4.2. Evaluation

For proof-of-concept, we connected various components of existing software tools in a way that may not be scalable but shows the potential of such analysis. In the following, we briefly discuss these tools and the testbed used for our experimentation.

4.2.1. Testbed. The testbed for our prototype was the eXtreme TORC (XTORC) cluster at Oak Ridge National Laboratory, which is maintained by our group for system software research and development. The system is configured in a standard Beowulf fashion with 1-2 service/login nodes and 64 compute nodes[*]. All nodes are single processor Intel Pentium 4 machines connected via a Fast-Ethernet (100 Mbps) switch. The head nodes export a Network File System (NFS) share to all compute nodes, e.g., the '/home' user directories.

The XTORC software configuration varies based on the current testing and development needs of the group. In our initial our experiments, the nodes spanned several generations of Linux distributions, to include: Fedora Core 4 (FC4) & 5 (FC5), and one Red Hat 9 (RH9) installation (node53)[†]. Note, there is considerable difference in these configurations, e.g., RH9 has a Linux 2.4 kernel, and FC4&5 a 2.6 kernel. Therefore the first experiments were generally characterized as *configuration anomalies*. In later tests we rebuilt the system with a uniform software build (Ubuntu 8.04) to investigate *performance* and *failure anomalies*.

4.2.2. Tools. The Ganglia monitoring software [9] was used on all nodes in the cluster. The Ganglia architecture includes a per node daemon, `gmond`, that collects local "metrics". Additional metrics can be added via the `gmetric` utility. All nodes subscribe to a multicast channel and broadcast their current information to all other nodes in the system. Every node maintains a copy of the current metrics and values for all other nodes in the system. The data is structured using XML and is stored in memory. The `gmond` is run on all nodes providing data to the monitoring system, i.e., service and compute nodes. The service nodes run the Ganglia Meta Daemon (`gmetad`), which collects the data from nodes (multicast stream) and writes each metric to a Round Robin Database (RRD) file [9].

By default, Ganglia gathers basic system settings (os_name) and information on memory consumption, processor load, disk utilization, and network transfer statistics. We supplimented this default data with additional metrics using the `gmetric` utility, e.g., CPU temperature via LM-Sensors [8]. As part of our proof-of-concept prototype, we have used pieces of the Ovis v1.1.1 toolkit [10] (RRD reader scripts) to interface with the data maintained by Ganglia.

4.2.3. Case-study: Configuration Anomalies. The following outlines the steps used to test our prototype on 48 hours of data from 52 nodes, i.e., 2496 node-hours. These are primarily *configuration anomalies* highlighting configuration/service differences among nodes.

- Data is collected via Ganglia into RRD files on 44 attributes at 15 second intervals. Using the Ovis RRD reader scripts, we converted the RRD files into flat files, one per node.
- The flat files are converted into a set of attributes for each node-hour with a function written in R. The R system is particularly rich in statistical attributes that can be computed from data. In this experiment, we only compute the mean for each attribute node-hour, giving a 2496 by 44 matrix.
- Standardizing the matrix and discarding constant columns reduces it to 2496 by 24.
- We cluster both, the node-hours as well as the attributes. The dendrograms for the 24 attributes and a portion of the node-hours that belong to the smaller clusters are shown around the heat map plot in Figure 3. A more readable version of the attribute dendrogram together with a vertically compressed full matrix heat map is in Figure 4.
- The yellow-red heat map is the raw attribute data ordered according to the dendrogram. It shows how the attributes are similar within the node-hour clusters and within the attribute clusters.

[*] Due to the cluster's age and hardware failures, there are typically between 45-60 nodes available for testing.

[†] The full node/OS breakdown was: RH9 on node53; FC4 on node4,58,59,60; & FC5 on node1-3,5-52,61.

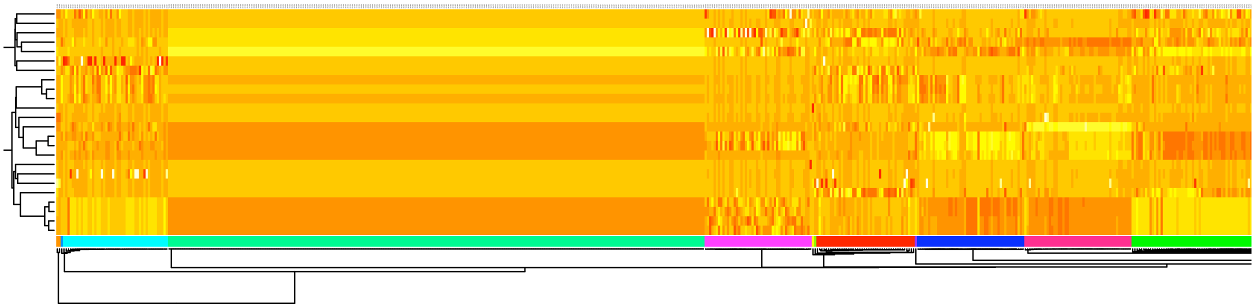


Figure 3. A portion of the transposed data matrix heat map showing the smallest 10 of 11 clusters. Attributes and node-hours are ordered by the dendrograms.

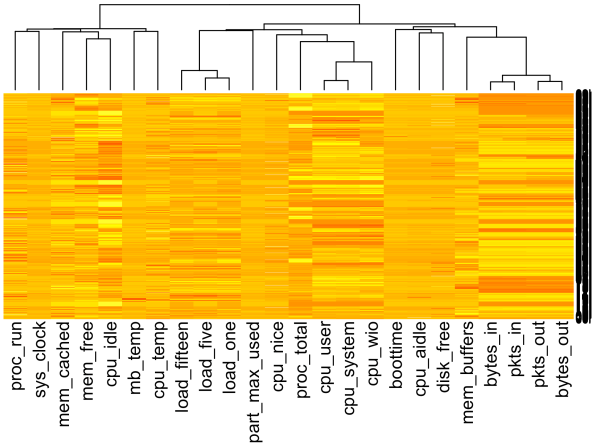


Figure 4. Attributes can be clustered according to their behaviour among node-hours. Data matrix heatmap (2496 by 24) is highly compressed in vertical direction.

During the 48-hour test period only Ganglia and Ovis processes were running in addition to normal idle-time operating system activity. After some investigation, the smallest of 11 clusters have an explanation that would normally lead to corrective action on a production HPC system. We make the following observations about the clusters identified.

- Node 0 is the most different from the rest, particularly its hours 13, 37, 46, and 47. This is the head node where most services are running. Note that hours 13 and 37 are 24 hours apart, indicating a daily service.
- Node 53 (all hours) runs the older Red Hat 9 while the others run Fedora Core 4/5.
- Nodes 12, 31, 39, 43, and 63 were all down.
- Node 13 was distinct, particularly hour 47, for undetermined reasons.
- Node 1, Node 5, and Node 30 (hour 7) were also distinct for undetermined reasons.

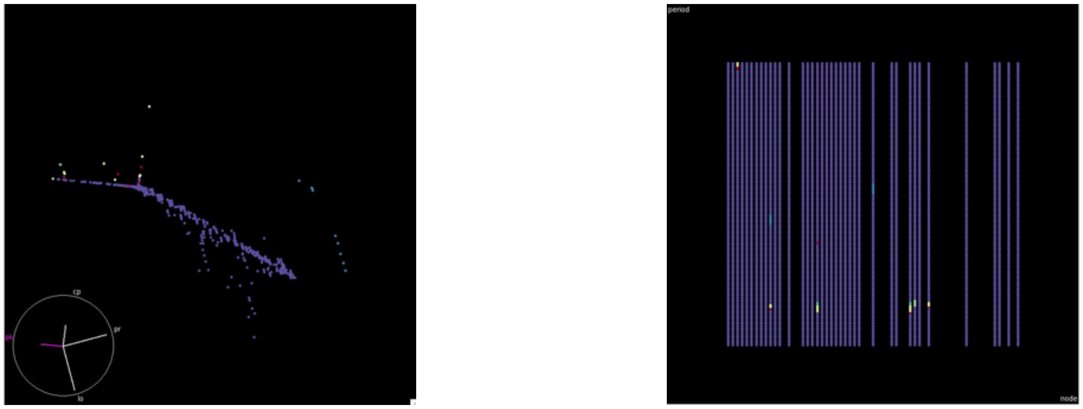
We can also interpret the attribute clusters apparent from the dendrogram in Figure 4. The first group is a set of temperature and memory related information, the second group is mostly processor related information, and the third group is mostly I/O related information. There are also finer groups, like the last four giving bytes/packets in and out. Note that the dendrogram imposes an ordering on the attributes that places similar attributes close to each other.

Another way to examine the data and clustering results is with the GGobi [14] interactive high dimensional analysis. The strengths of GGobi include the ability to browse projections of very high-dimensional data (easily 20 to 50 dimensions), linked displays that allow identifying points in different projections, and optimization for searching the projection space with various criteria. GGobi is also connected with R [3] so that the clustering resluts can be used to color-code the displayed node-periods. Here, the color-code is based on the simple anomaly measure of inverse cluster size spanning colors from purple through yellow to red. Figure 5 shows two different planar projections of the same high-dimensional attribute space. This display is highly interactive. The data in this view is based on a later *failure anomaly* experiment as discussed in the following.

5. Conclusions and Future Work

We have primarily identified anomalies which do not indicate failures, rather they indicate an unusual location or situation. The data was gathered on a mostly idle system with varied configurations. Our current work is looking into the behavior under various loads. It is not clear yet what computed features will turn out to be most useful. Also, we need to investigate what time intervals, such as node-minutes, may be optimal for given use. In order for the anomalies to focus on failures, some failures need to be observed while the monitoring is in progress. These may be induced failures, such as unplugging a link cable or disabling a fan.

In recent experiments, we have injected (performance) faults into parallel benchmarks. Figure 5 shows a projection of data gathered during such an experiment.



(a)

(b)

Figure 5. A projection of node-periods in a combination of four attributes emphasizing anomalies (a) and another projection in a node by period arrangement (b). Faults, which show up as anomalies, were injected during application execution/data collection.

In Figure 5 (b), each column represents a node (gaps reflect down/offline nodes) and the vertical axis reflects the time periods (start at bottom, stop at top). The injected faults are picked up by the clustering anomaly measure. A final anomaly is detected just before the application failed. Upon failure the application terminates and data collection is stopped as shown by the single anomaly at the end (top) of the periods. We plan to continue these controlled experiments and to improve the use of fault injection techniques to refine our detection and monitoring methods. This will also require investigation of the types of faults observed on production systems to improve the fidelity of our synthetic faults.

Processing of the data that becomes a set of node-hour attributes is currently done in a central location. For scalability, this needs to take place either on individual nodes or on distributed service nodes so that only the attribute data is forwarded to a central location. Additionally, the data gathering methods would need to be revised. The sampling methods used were rather heavy weight, such as starting a shell script to run the lm-sensor utility to gather temperature, and then starting separate utilities to log the event. More light-weight methods are needed.

References

[1] J. Brandt, B. Debusschere, A. Gentile, J. Mayo, P. Pébay, D. Thompson, and M. Wong. OVIS-2: A Robust Distributed Architecture for Scalable RAS. In *Proceedings of 4th Workshop on System Management Techniques, Processes, and Services (SMTPS)*, 2008.

[2] J. Brandt, B. Debusschere, A. Gentile, J. Mayo, P. Pébay, D. Thompson, and M. Wong. Using Probabilistic Characterization to Reduce Runtime Faults on HPC Systems. In *Proceedings of the Workshop on Resiliency in High-Performance Computing (Resilience'08)*, 2008.

[3] D. Cook and D. F. Swayne. *Interactive and Dynamic Graphics for Data Analysis: With R and GGobi (Use R)*. Springer, December 2007.

[4] C. Engelmann, G. R. Vallée, T. Naughton, and S. L. Scott. Proactive fault tolerance using preemptive migration. In *Proceedings of the* 17^{th} *Euromicro International Conference on Parallel, Distributed, and network-based Processing (PDP)*, 2009.

[5] GGobi Data Visualization System, URL: http://www.ggobi.org.

[6] J. C. Gower and G. J. S. Ross. Minimum spanning trees and single linkage cluster analysis. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 18(1):54–64, 1969.

[7] R. Gupta, P. Beckman, H. Park, E. Lusk, P. Hargrove, A. Geist, D. K. Panda, A. Lumsdaine, and J. Dongarra. CIFTS: A Coordinated infrastructure for Fault-Tolerant Systems. In *Proceedings of the* 38^{th} *International Conference on Parallel Processing (ICPP)*, 2009.

[8] LM-Sensors - Linux hardware monitoring, URL: http://www.lm-sensors.org.

[9] M. L. Massie, B. N. Chun, and D. E. Culler. The Ganglia distributed monitoring system: Design, implementation, and experience. *Parallel Computing*, 30(7), 2004.

[10] OVIS: Intelligent, Scalable, Real-time Monitoring of Large Computational Clusters, URL: http://ovis.ca.sandia.gov.

[11] R Project for Statistical Computing, URL: http://www.r-project.org.

[12] J. Stearley. Sisyphus – a log data-mining toolkit. URL: http://www.cs.sandia.gov/j̃rstear/sisyphus.

[13] W. Stuetzle and R. Nugent. Minimum spanning trees and single linkage cluster analysis. Technical Report 514, University of Washington, 2007.

[14] D. F. Swayne, D. T. Lang, A. Buja, and D. Cook. Ggobi: evolving from xgobi into an extensible framework for interactive data visualization. *Computational Statistics and Data Analysis*, 43(4):423–444, 2003.