Ovis-2: A Robust Distributed Architecture for Scalable RAS

J. M. Brandt, B. J. Debusschere, A. C. Gentile, J. R. Mayo,
P. P. Pébay, D. Thompson, M. H. Wong
Sandia National Laboratories, Livermore CA 94550 U.S.A.
{brandt,bjdebus,gentile,jmayo,pppebay,dcthomp,mhwong}@sandia.gov

Abstract

Resource utilization in High Performance Compute clusters can be improved by increased awareness of system state information. Sophisticated run-time characterization of system state in increasingly large clusters requires a scalable fault-tolerant RAS framework. In this paper we describe the architecture of Ovis-2 *and how it meets these requirements. We describe some of the sophisticated statistical analysis, 3-D visualization, and use cases for these. Using this framework and associated tools allows the engineer to explore the behaviors and complex interactions of low level system elements while simultaneously giving the system administrator their desired level of detail with respect to ongoing system and component health.*

keywords: RAS, fault-tolerance, failure prediction, scalable analysis, distributed analysis, cluster monitoring

1. Introduction

Traditional monitoring systems (e.g., [3, 2, 1, 6]) are very simplistic in that they periodically receive metric data from the monitored devices and compare the values with applicable threshold limits. If the limit is reached or exceeded the monitoring system may have the ability to invoke a simple system response, typically shutdown or reboot of the device. If the limit is not reached then no action is taken. In other words this methodology is generally reactive to faults that have already occurred and hence typically invokes an immediate response with drastic consequences in order to prevent further failure. While this is effective in terms of preventing a cascade of catastrophic device failures it places the entire burden for fault tolerance on the application using the devices. This mode of operation can be very costly in terms of resource utilization as the most common method of coping with this uncertainty is for an application to periodically checkpoint its data so that in the event of underlying hardware failure the application can restart from the latest checkpoint and hence make progress.

In contrast, Ovis [4] extends the facility for setting such failsafe thresholds by providing a variety of analytical tools for the statistical exploration of metric data taken over large numbers of similar elements or devices. Using learned statistical distributions, correlations and probabilistic models derived from this data, the engineer or system administrator can set thresholds in probabilistic terms in order to identify probabilistic outliers which can be indicative of degradation or impending failure. Additionally these tools allow tracking of how the models and distributions drift over time which can be indicators of environmental change or component aging. Such outlier identification can be used to signal to the operating system, scheduler/resource manager, application, and system administrator in advance of failure and thus allow time for non-catastrophic reaction including but not limited to checkpointing data that would be affected by failure of the component(s) identified, removal of the component(s), allocation of replacement resources, and restart of the application. This by comparison can be considered a pro-active system. Note that while some of the aforementioned systems (e.g., [2] and [1] via [5]) may provide some level of simple statistical calculations (e.g., max, min, mean) and display of such, these systems do not provide probabilistic characterizations nor use such characterizations as the basis of further run-time monitoring decision support. In section 3 we give several use cases as motivation for this type of analysis.

We introduced the first version of Ovis at SMTPS

These authors were supported by the United States Department of Energy, Office of Defense Programs. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed-Martin Company, for the United States Department of Energy under contract DE-AC04-94-AL85000.

in 2006 [7][†]. In that paper we described real-world applications of the visual display and of the statistical analysis techniques to abnormality detection, particularly in non-uniform environments. OVIS-2 is an entire redesign of the first version of OVIS to enable scalable, fault-tolerant monitoring and analysis necessary for the real-time management of large clusters. OVIS-2 has enhanced analytical capabilities, including increased abilities to compare current metric values to past statistical calculations, multi-variate correlations, and bivariate Bayesian modeling. Finally, OVIS-2 has a more interactive interface supporting multiple simultaneous cluster views, including 3-D views, and drag and drop selection of metrics for display.

In this paper, we describe the new OVIS-2 framework, with emphasis on the advances that address scalability and fault-tolerance; the new analytical capabilities and interface; and give examples of the use of the new features. We conclude with plans for future work.

2. Architecture

This section describes the OVIS-2 architecture. This includes the cluster specification, the framework architecture, the analysis architecture, and the user interface. A diagram of the architecture is shown in Figure 1 and the components of the architecture will be described in more detail below.

2.1. Cluster Specification

OVIS-2 is easily configurable to support new clusters. The cluster is specified via an XML description of the types of components in the cluster, metric data about the cluster that can and should be collected, the number of components of each given type, and the associations between components. Associations are used to specify

- containment information, such as the location of nodes within racks and the placement of racks within the room,
- network layout with a mapping of switch ports to networked components,
- power management information with a mapping of outlets to the components they power,
- serial console information with a mapping of compute node serial ports to ports on serial consoles, and

[†]Our pre-OVIS consideration of statistical analysis for cluster monitoring can be found in [8].

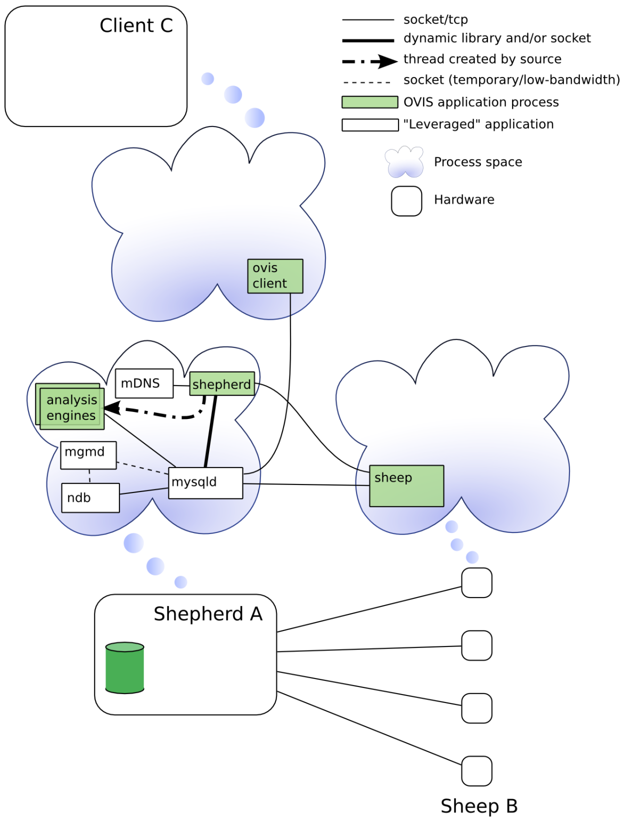


Figure 1. Block diagram of the OVIS-2 architecture. Sheep are responsible for collecting and inserting metric values. Shepherds maintain the database and perform the analyses. The diagram also shows example non-OVIS-2 components that are relied upon for operation, such as a MySQLCluster database.

- metric collection information, which specifies which components collect metric information remotely for other components in the cluster (e.g., by SNMP).

The XML description is used to create and populate database tables that are used to store a chronology of the cluster's state for later analysis and visual examination.

2.2 Data Collection and Storage

OVIS-2 currently stores metric data in a replicated database in order to provide fault-tolerance should a database node fail. In the longer term, only metadata – such as the component types, numbers, and associations – will be replicated so that data collection can be scaled further. Processes called "sheep" run on components or on behalf of a set of components and collect metric data for those components. "Shepherd" processes run on machines hosting the database. Shepherd processes advertise themselves using mDNS so that sheep need not store any configuration information themselves. Sheep then contact a shepherd (chosen at random to balance load) in order to find a database in which to insert their data. When contacted by a sheep, a shepherd may accept it or redirect it to another shepherd in order to balance load while maintaining some coherence (spatially, so that nearby sheep are usually attached to a particular shepherd, or temporally, so that a sheep tends to be given the same shepherd whenever that shepherd is available). When a sheep is accepted by a shepherd it then inserts data into the database on that shepherd's machine directly. If a shepherd node goes down, sheep will detect this via mDNS notification of a change in the advertised service and will contact a new shepherd from the list of those still advertising.

Currently, the metrics to be collected and their initial collection rate are stored in the database, having been specified in the cluster specification XML file. The framework provides a well-defined API for metric collectors, which include methods for instantiating collectors, invoking them at the correct rate, and for inserting data into the database. OVIS-2 provides collectors for metrics in well-known locations such as within the `/proc` hierarchy in UNIX. Readers also exist for accessing data via IPMI, Infiniband, smartctl, SNMP, and Ganglia/RRDTool interfaces, among others. Readers are supplied with OVIS-2 where allowed by the licensing constraints of the data interfaces. In future releases we will support dynamic discovery of metrics by searching for metric data files in well-known locations. In addition to sampling at regular intervals, metric collectors may push data into the database at any time so that changes in conditions that should invoke an immediate response are not delayed.

Each metric collector class is built into every sheep process. Sheep attempt to instantiate every metric collector at startup and only retain those which have a database table present. The sheep signal the remaining collectors to collect data at regular intervals specified in the database. In the event of the database being unable to handle the frequency of transactions, shepherds can notify the sheep to change their metric insertion rate or redirect sheep to other shepherds. Because the initial choice of shepherd is made randomly by sheep it is expected that the load, in terms of sheep per shepherd, will be well balanced. A shepherd, knowing both its current load and those of the other shepherds, can make the decision to redirect a sheep requesting service to another shepherd process. Significant load imbalance, in terms of processor and/or storage subsystem, can still occur in the system due to several factors:

1. changes in shepherd population due to failure or startup of shepherds;
2. sheep processes that act on behalf of many remote components insert data for all of those components via a database connection to a single shepherd and can thus account for several times the traffic of other sheep;
3. variable frequency of insertions, depending on the component and the type of data being saved;
4. other processes running on a shepherd (such as analysis requests that read from the database) may place differential load on particular shepherds.

Thus it may be insufficient to just address gross imbalance in the number of sheep and indeed such an imbalance may be needed in order to balance the actual loads on the shepherd in terms of CPU utilization and/or database transactions. Although OVIS-2 provides a framework for redirecting sheep and throttling metric collection rates, different ways to measure and classify load imbalance are still being explored.

2.3 Analysis Architecture

As mentioned in the introduction, OVIS-2 provides ways to Learn descriptive statistics (minimum, maximum, mean, variance, skewness, and kurtosis of a metric), correlative statistics (mean, variance, covariance, linear correlation coefficient, and linear regression lines calculated on pairs of metrics), and bivariate Bayesian parametric models (fits of a normal, log-normal, or exponential distribution whose parameter(s) is/are function(s) of two metrics). Therefore, Learn can either mean that the calculation is done analytically (*e.g.*, descriptive and correlative statistics), or is based on approximations such as quadratures (bivariate Bayesian modeling). Generically speaking, the output of a Learn analysis is a *model*, which may be used in several ways.

First, a model may be taken as ground truth and used to find outliers in a given dataset – which may, but does not have to, be acquired at times subsequent to the dataset used in computing the model. This mode of execution is called Monitor. For instance, in the case of descriptive statistics, Monitor evinces outliers based on prescribed extremal bounds and/or nominal value and acceptable deviation; these values may be obtained from a previous execution of a descriptive statistics analysis in Learn mode using training data – but they can also come from any other origin such as manufacturer specification, expert knowledge, or simply be a desired range of operation. On the other hand, for bivariate Bayesian statistics, the Monitor mode identifies events whose probability of occurrence is calculated as being low with respect to a given probabilistic threshold. This threshold provides the user with means to steer the sensitivity of the Monitor.

Conversely, the validity of a model may be tested by assuming a dataset – which again may, but does not have to, use a different set than what was used initially – should have the same characteristic as the model and assessing how well the model matches the dataset. This mode of execution is thus called *Validate* and is currently implemented only for Bayesian calculations, but will soon be extended to all types of analysis. The main use of this mode of execution is to inform the user as to the applicability of a model to a particular dataset. Note that it can be utilized, by repeatedly running Validate analyses using the same model, but with successive datasets, to assess the progressive "drift" of baseline behavior with respect to an initial model. Depending on the use case, such a drift may be entirely normal – and even expected – or indicate that something is deteriorating. Therefore the Validate mode of execution can serve as a diagnostics tool as well as Monitor mode.

To summarize, these 3 modes of execution are intended to cover in particular the following use cases:

1. There is no existing model, or an existing model has been found to not fit current data and so a new model is being calculated.

2. A specified window of time has passed and a model is being validated against the incoming data.

3. Incoming data is being compared for classification against a valid model. Note that several of these use cases rely on the tasks above being performed at regular intervals.

The tasks involved in these use cases must run to completion within that interval in order to be of use. Ovis-2 parallelizes these analyses so that appropriate selection of the ratio of sheep to shepherds will allow the tasks to complete in time.

Since MPI was not designed for fault tolerant operation, we use a different scheme to perform parallel calculations. Each calculation task is queued by inserting a row into a database table – the requests table. Each shepherd available at the time of the insertion performs a portion of the calculation and inserts its results into a second database table – the intermediate results table. Each time new results are placed into the intermediate results table a database trigger updates a row in a third database table – the final results table – by updating new values with any existing intermediate results – these updates may take the form of a sum, or of taking the minimum (or maximum) between the existing intermediate result and the new result. At any point in the calculation, any subset of the shepherds may fail and yet the calculation will complete. If shepherd(s) fail during a calculation, the results will be degraded. However, the severity of the degradation will vary with: 1) the relative size of the data lost as compared to the entire dataset, 2) the statistical variations within the dataset, and 3) the way with which computational work has been distributed amongst shepherds.

Values in the intermediate results table are simple sums over portions of the data along with the number of samples summed. The quantities being summed depend on the task. As an example, these sums are the raw statistical moments of metric data when learning descriptive statistics parameters. Each shepherd is assigned a portion of the components in the request and computes sums of statistical moments of the metric of interest over the time interval specified in the request for those components. The user interface is responsible for converting the raw moments into the mean, variance, skewness, and kurtosis. Because the intermediate results are independent of one another, no communication between shepherds is required and the speedup is linear in the number of shepherds. Only the final sum is performed in series. Ideally, all of the shepherds would perform some part of the work combining intermediate results to obtain the final result but in practice – since the combination is a simple sum of 5-10 numbers from each shepherd – no need for this has arisen and so a database trigger (run on a single shepherd) does this work.

2.4. Graphical User Interface

Ovis-2 provides a user interface targeted at both system administrators and engineers/analysts. The goal is to provide both high-level visual and statistical information about the state of the cluster as well as precise information about the state of individual components in the system. While system administrators may be largely concerned with addressing system failures, other users are expected to be interested in information that will help tune application performance. The user interface allows the user to interact with the environment in passive monitoring mode as well as active data exploration.

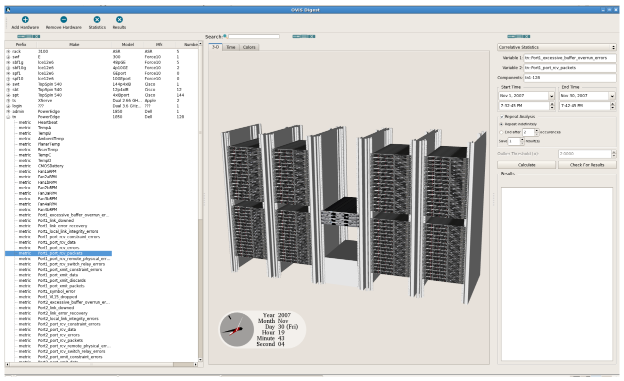


Figure 2. A screenshot of Ovis-2 with several views active (left to right: metric list, see §2.4.3; 3-D physical, see §2.4.1; correlative statistics, see §2.4.2). Views will be split out for clarity in the figures that follow.

Information for a particular cluster is provided in a single window split into panes as shown in Figure 2. Each pane provides an independent view of the same cluster and may be moved, resized, and split by the user. When a pane is split, its contents are kept in one of the two new panes that replace the old pane. The other pane shows a list of view types so that the user can choose what type of information should be displayed. Panes may be split vertically or horizontally.

2.4.1 3-D Physical View

The physical view shows a graphical 3-D representation of the cluster, as in Figure 3. All information in the rendering, including the shape and texture data, is retrieved from the database. The purpose of this display is to aid users in developing models of performance or reliability which can then be tested using the statistical tools Ovis-2 provides.

Each component in the cluster can be colored individually, with its hue based on metric values describing its state at some point in time and its saturation based on the amount of time elapsed since the last measurement, allowing temporal behavior to be explored in the face of missing data. The spatial effects of the environment on the cluster are often easier to see when the cluster is presented in its actual physical configuration. The temporal behavior of metrics can be explored using a time widget associated with each display; by dragging a clock widget's second, minute, hour, day, or year hands (see the bottom left of Figure 2), the variation of values over time can be explored. Each display

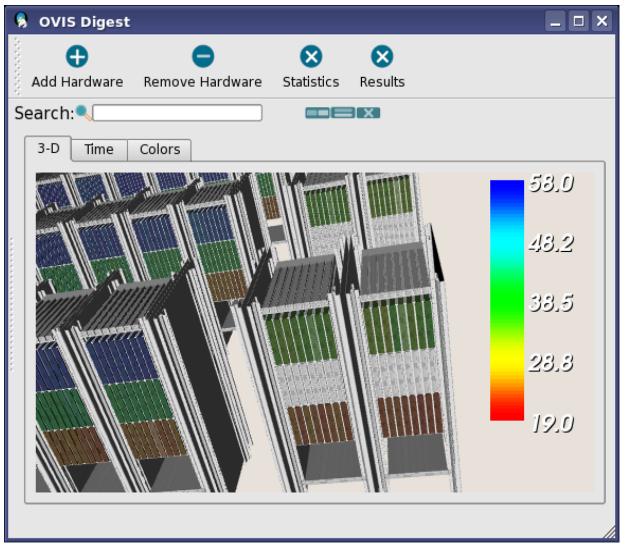


Figure 3. A screenshot of Ovis-2 with a physical view containing nodes colored by a metric value of a contained component (in this case, a microcontroller temperature) Note the strong dependence on height within the rack.

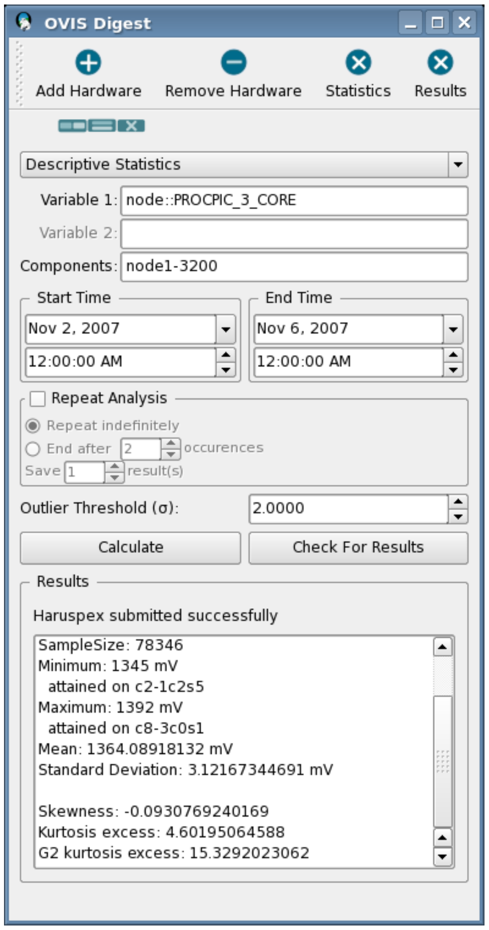
can also be set to automatically advance through time at some fixed rate so that system administrators can view the cluster state changing in real time and application engineers can play back historical data at a pace that is useful in diagnosing problems with application performance. A list of components may be selected by name or number in a physical view; any component in the list is drawn pushed out along an axis normal to its front face – so that selected nodes appear to "pop out" from the rack they inhabit.

Multiple panes containing different 3-D physical views can be displayed at the same time. This allows for simultaneous differing perspectives of the cluster, differing metrics being visualized, and the comparison of the same metric at different points in time.

2.4.2 Analysis Views

Though a user can examine the behavior of a cluster using the interface described above, it is also possible to drill down and retrieve specific quantifiable information by requesting a statistical analysis be performed. To request an analysis, the user splits an existing pane and chooses an analysis input view for the newly-created space in the cluster window.

If descriptive or correlative statistics are requested (Figure 4), the user must specify a range of time, a set of components, and the metric(s) of interest before



```
SampleSize: 78346
Minimum: 1345 mV
  attained on c2-1c2s5
Maximum: 1392 mV
  attained on c8-3c0s1
Mean: 1364.08918132 mV
Standard Deviation: 3.12167344691 mV

Skewness: -0.0930769240169
Kurtosis excess: 4.60195064588
G2 kurtosis excess: 15.3292023062
```

Figure 4. The descriptive statistics view panel where parameters are specified and results are displayed.

submitting the request. Once submitted, the analysis is performed in parallel on the nodes which contain the metric database. There is space below the input parameters where results are displayed.

Bayesian analysis currently has a dedicated view for specifying input parameters since they differ significantly from the simpler types of analysis. However, the mechanics of selecting an analysis are much the same.

It is also possible to browse previous analysis results via a third type of view. Both the input parameters and the results of the analysis are displayed. Finally, any analysis request may be set to repeat at regular intervals. In this way, if a model is developed that predicts node failure, the forecast can be kept up to date with new measurements as time goes on. When an analysis is repeated, the user may select the number of repetitions, or for analyses which should be repeated *ad infinitum*, the number of results to keep.

2.4.3 Descriptive Views

The list of available metrics for each type of component appear in a tree view, which along with a 3-D view form the initial two panes in any cluster's window. These metrics not only provide a reference for the analyst, but they can be dragged and dropped onto the 3-D physical view and the analysis input views to specify how nodes should be colored or which values should be considered in an analysis, respectively.

3 Use Cases

In this section we discuss applications of the methodologies of the Ovis project as applied in Ovis-2 to cluster state situations where traditional threshold-based monitoring methods would be inadequate. We consider not only abnormality detection to be used for early indication of potential failure but also possibilities for statistical consideration of system state to be applied to more efficient use of system resources.

3.1 Spatial Dependencies as Root Cause Indicators

In Figure 3 we show a particular cluster, where the nodes are colored by the temperature of a microcontroller contained within each node. From the figure, one can see that the temperatures of the considered component increase over 10 degrees as one moves up in height from one chassis to the next. This relationship can also be discovered by the correlation analysis of this value with component height. Knowledge of the spatial dependency is critical in indicating that this may be an effect of the airflow due to the cluster design. This is further indicated by consideration of the two outer racks in each row, where the middle chassis does not contain compute nodes and hence the upper chassis' values are less than those in other racks where the middle chassis is populated.

Since it is well known that temperature can have strong effect on electronics components' longevity and performance, we are in the process of checking for failures and/or abnormalities in the current and historical operating behaviors of the upper nodes as compared to those of the lower ones. Note that since the values are within the traditional threshold limits, the vendor supplied monitoring system does not give indication of this condition. Further, simple min/max/mean consideration is insufficient to pick up the spatial dependency which could be crucial to root cause analysis.

3.2 Extreme Statistical Abnormalities in a Large System

In this case, we explored the voltage distributions of various elements in a cluster consisting of 920 nodes and 3680 processors. For one of the core voltages we discovered that over the 920 similar elements there were two significant outliers. In this case the voltage distribution over a range of time was seen to be a reasonable fit to a normal distribution (see Figure 4 for statistics over a more limited range of time) with a mean of 1364mV and a standard deviation of approximately 3mV. One outlier was approximately ten standard deviations off the mean on the high side with a maximum voltage of 1393mV and another approximately 6 standard deviations off the mean on the low side with a voltage of 1342mV. We then looked at the distributions of voltage values of these two outlier elements alone over time and found that the mean for the high outlier element was 1387mV with a standard deviation of 2mV and the mean for the low outlier element was 1347mV with a standard deviation of 1.7mV. Thus the means of these outlier elements are 7.7 and 5.7 standard deviations outside of the mean of the entire set respectively.

While these voltages are not out of bounds as far as the absolute thresholds are concerned, as evidenced by the nodes still being in service, they should certainly be classified as anomalous because they contain elements whose values lie very far out in the tails of the distribution for that metric. In this particular case we only had a week's worth of data at the time of this writing, but the voltages looked stable over that time independent of what was running on the nodes. Also their distributions appeared to be in line with other single element distributions of this metric over the same time period. It will be interesting to follow this and see if further degradation occurs or if errors are eventually reported for these nodes. Here the high voltage is of concern as it may lead to premature component failure and the low voltage is of concern as it could cause timing issues though we have not yet discussed this with the vendor and do not know what the design tolerances are.

3.3 Model Change Implications and Aging

One of the components OVIS-2 can monitor is hard disk drive parameters via SMART data (if implemented). Using this mechanism we are collecting data in order to model REALLOCATED SECTOR COUNT vs. POWER ON HOURS and additionally the rate of change per unit time. In one system, we are currently monitoring 160 500GB hard drives. Initially there is some count which represents fabrication defects which were remapped at the factory. Since we are looking for surface degradation we subtract the initial count as an offset which then yields a count of zero when we start. Thus the initial model will be simple, zero in this case, and exhibit no temporal dependency. The need for automatically checking the model against the data and building a new model when the current one doesn't fit is very evident in this type of scenario where it is expected that as the component ages the applicable model will change. As we start to experience read/write errors (which we have not so far) sector reallocations will start to occur in order to map them out. Though SMART data has a threshold to compare against for when to signal the need to replace the disk it gives no warning until that threshold is crossed and no hint as to how fast the degradation is progressing. OVIS-2 can build applicable models in near real time as data is collected. Using these models as the basis for comparison quantitative predictions can be made as to the relative health and rate of degradation of individual drives long before the "replace" threshold is reached.

Such data monitoring and analysis can also be used for making run-time decisions for resource allocation. Based on this type of data OVIS-2 could give hints as to which disks are more reliable in order to allow applications to make choices based on the importance and expected longevity of their data being stored.

3.4 Job-Centric Information

Job-centric information and statistics display is currently under development, but is a high-value target for the OVIS project's methodologies as statistical consideration of such information can be used not only for enabling preemptive failure mechanisms but also for providing statistical insight into non-failure related application behavior and cluster utilization.

To first order, OVIS-2 will read job data from scheduler log files; later OVIS-2 will provide an API by which the scheduler can provide that information to OVIS-2. In the latter case, the interaction can be invoked, for example, in the job prologue and epilogue scripts.

OVIS-2 will provide different types of job-centric information to satisfy the needs of a variety of users. Visual monitoring of global cluster utilization would be of interest to cluster administrators, while, for a cluster user, visual monitoring of the communications amongst the nodes using a logical connectivity display of the nodes can give important information about the optimal job placement. OVIS-2 will also support statistics by job group, including both metric statistics pertaining to components comprising a job as well as statistics of job-related data, such as queue time and run time. Such job-related information is of value to cluster administrators in the establishment of queuing policies.

4. Future Work

Though we believe that Ovis-2 addresses a much needed capability for HPC platforms with its advanced statistical analysis in a robust scalable framework, we think that there is still room for advance in terms of metric data storage and retrieval efficiency, analysis, visualization techniques, and system response. Additionally, appropriate APIs allowing better communication between Ovis-2 the scheduler/resource manager, and running applications could facilitate a more scalable fault tolerant platform in the future. Below we discuss some of our plans for future work in these areas.

1. Time series analysis using Bayesian inference to characterize temporal behavior patterns of elements and correlations of these behavior patterns with application and environment. Of particular interest are network and I/O behavioral patterns in that knowledge of these for a particular application could facilitate more efficient use of platform resources if it were used at job launch time to optimize resource allocation.

2. Root cause analysis using Bayesian networks. Using these techniques we plan to use historic failure information coupled with domain knowledge in order to help the system administrator identify possible causes of failure and quantify the probabilities of these. As a historic repository of failures and their causes is built for a particular platform, this type of analysis should enable efficient targeting of root cause mechanisms even as outliers are detected and save system administrators the time typically needed to track this down manually.

3. Currently Ovis-2 uses a spatially accurate rendering of an HPC platform together with a color map overlay based on metric or analysis data to allow the user to visualize spatial and temporal behaviors. In the area of networking and storage we are experimenting with various graph based mechanisms to facilitate similar understanding. Discovery of an application's dominant communication patterns coupled with a low level understanding of how this affects contention in the networks given a particular topology could facilitate resource allocations that would minimize such contention and hence improve overall performance.

4. Database optimization. We are investigating performance tradeoffs in the table definitions and in the data storage methodologies. This involves designing to support both data inserts and analyses' queries of a variety of complexity, depending on the analysis type. Further, there is the question of what to keep in the active searchable database as opposed to what to store longer term. We will be investigating options for dynamically dropping detailed data and keeping only metadata descriptions, including model results, for older data, with the ability to reload the detailed data and integrate it into the active data for analyses over long time periods.

5. Conclusion

As cluster size continues to increase, so does the need for scalable RAS methodologies. Additionally, the increased likelihood for component failure during large, long-term jobs puts a burden on applications to provide their own fault-tolerance if none is provided by the system. In this paper, we have described the new scalable and distributed architecture for our Ovis-2 cluster monitoring and analysis tool. We have given use cases where the unique statistical characterization approach used in Ovis-2 can be used not only for anomaly detection as an early indicator of potential performance degradation and/or failure, but also as a mechanism to enable more efficient use of system resources.

References

[1] Ganglia. `http://ganglia.sourceforge.net`.

[2] HP Cluster Management Utility. `http://h20311.www2.hp.com/HPC/cache/412128-0-0-0-121.html and system documentation therein`.

[3] IBM Cluster Systems Management. `http://www-03.ibm.com/systems/clusters/software/csm/ and system documentation therein`.

[4] OVIS. `http://ovis.ca.sandia.gov`.

[5] RRDtool. `http://www.rrdtool.org`.

[6] smartmontools. `http://smartmontools.sourceforge.net`.

[7] J. M. Brandt, A. C. Gentile, D. J. Hale, and P. P. Pébay. Ovis: A tool for intelligent, real-time monitoring of computational clusters. In *Proc. 20th IEEE International Parallel & Distributed Processing Symposium (SMTPS)*, Rhodes, Greece, Apr. 2006.

[8] J. M. Brandt, A. C. Gentile, Y. M. Marzouk, and P. P. Pébay. Meaningful automated statistical analysis of large computational clusters. In *IEEE Cluster 2005*, Boston, MA, Sept. 2005. Extended Abstract.