LANL Supercomputing Data Analysis

Clay Baenziger, Bruce Bugbee, Ryan Ford, Charlie Grammon

Mathematics and Computer Science Department
Colorado School of Mines
Golden CO, 80401

Abstract

With the increase in large-scale supercomputing projects, the analysis of data regarding supercomputer failures is becoming more important. Los Alamos National Lab recently released a large dataset regarding various supercomputer systems including information on usage, failures, position, and automated event logs. Our team performed many different statistical practices to characterize the data as well as search for possible correlations between failure and other (potentially predictive) data. We created a custom workflow consisting of a Solaris machine, Perl scripts, scripts in the R statistical programming language, and databases provided in a MySQL relational database to perform our analysis. Apart from our analysis we found numerous logical fallacies and corrupt fields in the data provided by the client. Our analysis produced interesting results, many of which supported conventional wisdom of how supercomputers behave under varying circumstances.

1 Introduction

With the growing complexity and necessity of mathematical models in today's world, the development of reliable supercomputing systems is a priority of the scientific community. As with any problem, the best way to increase reliability is through in-depth analysis of data pertaining to all aspects of supercomputing [RR00]. This is very challenging due to the often sensitive nature of the research that utilizes supercomputers. This causes private organizations and governmental agencies alike to either horde this data or not even record it [BS07]. This attitude of secrecy was the standard until Los Alamos National Lab (LANL) recently released a large, comprehensive dataset encompassing many different aspects of supercomputing to the academic community in hopes of using the resulting analysis to improve their research systems. This dataset contains information on usage, failures, automated event logs, and positional data. The rarity of this data combined with the potential improvements to supercomputing practices provides fertile grounds for original analysis on the topic.

The scaricity of comparable data sets have left us with very little previous research with which to base our work upon. The most prominent analysis of this particular dataset was done by Garth Gibson and Bianca Schroedier of Carnegie Mellon University (CMU). The bulk of their research so far has been focused upon the subset of the LANL-released data pertaining to failures. Their work provides a basic characterization of the failure data as well as theoretical inquiry into distribution fitting and modeling

The goal of this paper is to build upon the previous work done at CMU and expand it to provide statistical characterization of usage, automated event log, and positional data as well as describe patterns and correlations found between the subsets. We first describe the systems that are used in our analysis as well as the format and collection methods of the various data subsets (Section 2). Section 3 provides details concerning the technical aspects of our project as well as our statistical methods. We then provide various characterizations including user-based and temporal concerning the usage data (Section 4). Section 5 provides similar characterization of the auto-

mated event log data. The subsequent sections provide analysis concerning correlations between usage and failure data (Section 6) as well as positional and failure data (Section 7). Section 8 concludes the paper and Section 9 provides acknowledgments.

2 Description of the Data and Systems

2.1 System Profiling

The majority of our research has been conducted with respect to five different systems at LANL. These are Systems 8, 15, 16,20, 23. Although LANL hosts many more systems than this, the usage data which we were provided is only for these five systems and so we limited our scope accordingly. The following section details the system information for each of these five systems.

System 8 is NUMA (Non-Uniform Memory Access) cluster system with 164 nodes at two CPUs per node and 446 distinct users. It was installed in March 2001 and put into production in April 2001. It is still running at LANL. It is used primarily as an open-science system typically running smaller jobs. It is also used as a development cluster.

System 15 is single node NUMA machine with 256 CPUs and a total of 74 distinct users. It was installed and put into production in November 2004 and is still currently in operation. System 15 is used for open-science work, typically assigned to a single research project.

System 16 is a NUMA cluster system with 16 nodes at 128 CPUs per node for a total of 2048 CPUs. It hosted 925 distinct users. It is a much older system than the other four detailed in this report as it was installed in October 1996 and put into production during December 1996. It was decommissioned in September 2002. System 16 was classified as an open science system.

System 20 is labeled as a NUMA cluster system with 512 nodes at 4 CPUs per node. Installed in October 2001 and placed in production in December 2001, System 20 is still currently running at LANL. It has 612 distinct users documented in its usage data. This system proved particularly interesting due to the fact that it is used primarily as a weapons research system.

System 23 is a NUMA cluster machine with five nodes. Its first node contained 32 processors and each of the remaining four had 128 processors each giving a total of 544 processors. It hosted 800 distinct users. Installed in October 1998 and then put into production that same month, System 23 is the second oldest system of the five. It was decommissioned in December of 2004. Machine 23 acted as both an open-science and a weapons system with minor amounts of visualization activities as well.

2.2 Usage Data

The main focus of this paper is the analysis of supercomputer usage data provided by LANL. The data is a log of each job ran on the system, recording specific information such as user ID, submit time, suggested start time, deadline time, dispatch time, end time, amount of process time spent on executing user code, amount of process time spent on executing system kernel code, number of processors requested, and which nodes of the system the job ran on. Submit time, suggested start time, deadline time, dispatch time, and end time are all given in the standard UNIX timestamp format (number of seconds since 1/1/1970) and both process time measures are given in seconds. Each log entry is created by the batch queue system responsible for the running, tracking, and delegation of resources for all jobs for the particular system in question. This system is fully automated and thus the usage data is totally independent of the system administrators for each system. An example of a usage data entry from the LANL readme file is:

1038350873 3405 4 1038350856 0 0 1038350856
0.079056 0.035136 0.114192 4 0 d2 1 d2 2 d2 3 d2
1129913079 25731 4 1129912866 0 0 1129913037
*124.000000 1.000000 125 1 0 4*d2*

More information on the format of the usage data can be found in the readme files of that accompany the LANL dataset. The usage data provided contains information regarding five systems of various research interests: System 8, System 15, System 16, System 20, and System 23.

2.3 Failure data

The failure data used is based on a "remedy" database that was created in 1996 as part of a policy shift at LANL. This new policy was designed to have each failure that required the attention of a system administrator entered into this log in the hope of aiding future researchers. Each record contains the date and time of the failure, the system and node it occurred on, the type of workload that was running on the particular node (defined as *compute* for computational workloads, *graphics* for visualization workloads, and for *fe* front-end/interactive workloads), as well as the cause for the failure. The cause for each failure was chosen from six categories: Hardware; Software; Network; Human Error; Facility, including power outages, environmental damage, and cooling problems; and Unknown. In addition to the root cause of the error, a more detailed description of the problem is provided; such as which hardware component failed for failures classified as Hardware. The protocol for creating this log was created by a team of engineers, administrators, and operations staff members at LANL. The process for creating an entry begins with an automated detection system alerting the system administrator of a problem. The system administrator then investigates the problem and tries to determine the root cause along with additional information. If the root cause can not be determined it is marked as Unknown. An example of a failure data entry from the LANL readme file:

2, cluster, 49,6152,80,0,0,5-Apr,5-Jun, current,part,80,1,1,0,graphics.fe,6/21/2005 10:54,6/21/2005 11:00,6,,Graphics Accel Hdwr,,,,,N

The failure data provided contains information regarding 23 different systems. For the analysis in this paper, the systems that also have usage data provided will be the main focus. Therefore, we will limit failure analysis when used with usage data to the same systems and timeframes that the usage data spans. More information regarding the format of the entries can be found in the readmy files that have been released with LANL's data.

2.4 Event Data

For System 20, an automated event log was provided for analysis by LANL. This event log maintains records of various events that happen on System 20, where events can be loosely categorized as anything that changes the overall state of the system. Examples of this include high temperature warnings, fan failures, and file system problems. System failures can also be categorized into the event log. It should be noted that each entry in the log is not necessarily unique. A single event can be reported numerous times because it affects various nodes. Each entry contains an event ID (a generic number used for reporting), the subsystem affected, the classification of the event, the type of event, the timestamp of when the event was reported, if the event was handled by an administrator, and a description of how the event was handled. An example of a typical event log entry for System 20 with no node iformation is:

460903,resourcemgmtdaeomon node-25, server,subsys,1145552216,1, failed to configure resourcemgmt subsystem err = 10

More information on the format of the event log data can be found in the LANL-created readme files that were released alongside of the dataset.

2.5 Positional Data

The positional data used is based on a basic three dimensional grid system to describe the location of the nodes within a certain machine room. The coordinate system used begins at (0,0) with the values describing relative North-South and East-West positions respectively. The North-South (East-West) orientations were divided into groups of major and minor rows (columns) which varied depending on the system Also provided, was information on a node's height in the stack for systems that had multiple nodes in a single vertical housing as well as information regarding lateral stack position for systems that housed nodes on the same shelf. The positional data spans three separate machine rooms that house various numbers of systems. An example of the format of the positional entries as provided in the data files by LANL is:

Machine 8,Bldg 1,Room 1, RackPosition, RackPosition, Position in Rack, NODE NUM, East/West, North/South,Vertical Position, N number, 1 to 26, 28 to 35,"1 to 37, top to bottom"

In total there were 14 different systems for which positional data was provided. Of the systems for

which usage data was provided, only System 8 and System 20 also had positional data.

2.6 Data Cleaning

While the majority of the data provided is accurate, it is to be expected that in a dataset this large there are some areas in need of sanity checking to ensure that there is no corrupt data or faulty entries. Surprisingly, most of the anomalies in the data stem from the automatically collected usage data rather than the administrator-entered failure data. Problems encountered in the usage data included submit times, dispatch times, and end times that were outside of the range of the data (pre-1995), job end times that occur before their respective submit times, and jobs that were dispatched before they were submitted or after they had already ended. When investigated further, we were told by LANL that this was simply corrupt data and it was therefore acceptable to be thrown out of future analysis. We have provided a cleaned version of the data back to LANL with documentation concerning all errors and descrepencies found. A through explaination of what errors were found and corrected can be found in Appendix B.4.

3 Analysis Background

3.1 Analysis Tools

All datasets were given to our team in standard comma-separated text files that included readme files describing the format. These datasets are quite large (in excess of 300MB text files in some instances) so it was extremely important for us to implement a combination of technologies to handle and analyze the data. We set up a Solaris Nevada workstation with 12 GB of RAM and two AMD Opteron 2.5 GHZ processors as our main workstation for handling our project. To get the data into a usable format, we used custom built Perl and shell scripts to parse the data into a MySQL relational database [Dav07]. For our statistical work, we chose to use the R open source programming language [R D07]. R was chosen for many reasons including its free cost, ease of use, and its growing acceptance in the statistical analysis community. By using the RMySQL library for R, we created an environment in which we can execute MySQL queries within R scripts and then directly run analysis including graphical methods on the results. We have adopted a paradigm of using MySQL to do all of the data extraction. This enables us to avoid a commonly noted pitfall of R in that it tends to break down when performing data extraction on large datasets such as the ones we are using. Within this project, the role of R is to provide the statical evaluation techniques and graphical output needed for our analysis. We have also implemented another system for analyzing the data by using the software package MiniTab in a Windows XP environment. While our team prefers our combination of R and MySQL, we have found MiniTab easier to use for creating graphics and analyzing time series data in some cases (most notably the positional data). Further explanation regarding our analysis tools as well as examples can be found in Appendix B.

3.2 Statistical Methods

To thoroughly analyze the data provided to us, our team chose to implement a variety of statistical techniques. This was done in an effort to overcome the shortcomings of any single method and gain as much insight into the data as possible. Our techniques can be divided into two categories: numerical and graphical. Numerical methods such as correlation were used to provide a quantifiable description of various characteristics within the data. Despite their value, numerical methods alone prove insufficient for this dataset. This deficiency is compensated by the use of graphical methods. Graphical methods, while not as quantitative, provide a way to easily visualize data from large sets. Since the goal of this project is to analyze general patterns within the various datasets provided by LANL, the bulk of our analysis is in the form of graphical methods. Examples of the graphical methods we used include histograms,barplots,scatter plots, box plots, contour graphs, and correlation matrix plots. These methods simplified complex univariate and multivariate pattern analysis to an acceptable level. It is only by employing both types of analysis that we were able to readily and efficiently analyze the data provided.

Due to the unique nature of the data, our team faced some challenges adapting these methods to provide quality analysis. The most common problem was the sheer size of the datasets. This made graphically representing the data in an understandable fashion difficult. To compensate for this, our

team analyzed subsets of the data that were easier to visualize and modified graphical parameters to allow for better display results. We also ran into problems with the outliers in the data. Often times, outliers would occur in the usage data that would represent a large job. These data points would be numerically distant from the rest of the data and thus would skew any graphical representation. When acceptable, we would use a quantile clamping method to compensate for the outliers. The quantile clamping method is simply analyzing a smaller quantile of the data that does not contain the problematic outliers of the original set. This method was used sparingly as to not neglect pertinent and legitimate data that could be confused with outliers. Analysis challenges have been well documented throughout our research and are discussed further throughout this paper.

Types of Analysis

4 Usage Data

4.1 Usage Attribute Correlation

The usage data provided contains many measures that can be used to characterize each system. Measures such as CPU seconds system, number of CPUs requested, total number of jobs, and job length can be used to create a usage profile of each system and analyze the various strains placed upon it. An area of particular interest is how each of these measures affect each other.

To analyze this, our team gathered the mean values of each measure from each node on the system and created a simple correlation matrix pitting each measure against all others. It should be noted that System 15 was omitted due to the fact that it only has a single node. After plotting the correlation matrix, there were no prominent patterns that appeared across all systems. This result was expected since each system is used in very different manners and thus their usage attributes would affect each other in different manners as well.

While there are no global patterns within the usage attribute correlation that are readily seen, there is a system specific pattern found on System 16 that is interesting. On System 16, it was observed that there is a slight correlation (0.28) between the average number of CPUs requested and the average total time of a job. What is interesting about this is the plot that describes these two attributes indicate that as the average number of CPUs requested increased, the average total time increases until a medium number of CPUs is reached. At this point, the slope of the plot changes sign and indicates that as the number of CPUs increase the average total time decreases. This indicates that for many middle-length jobs, it might be more efficient to have them utilize more processors in an effort to decrease total time and allow for more jobs to be run.

4.2 Temporal Analysis

With regards to time, the general pattern that all systems exhibit is that there exists a brief time period during the infancy of the system in which the usage is dramatically low. This observation is not unexpected, in that it is anticipated that each system will take a significant amount of time to get set up and configured before normal usage traffic commences. After this period of low usage, the system experiences a ramp-up of traffic. At this point, the usage of the systems stays somewhat stable for the remainder of the lifespan of the machine. While the various usage characteristics do fluxuate slightly, they tend to stay close to the average values with very few exceptions. It should be noted that System 23 experienced a significant drop in usage towards the end of its life. This was attributed to the opening of a new general science system to which many of System 23's began using as a replacement.

Analysis indicates that usage traffic is definitely affected by both time of day and day of the week for each system. The bulk of the usage occurs during the normal work week (Mon-Fri). Machines tend to be very active between 9am and 12am. System 8 normally spikes between 10am and 12pm and then drops down for the next three hours. It then picks back up and remains consistent until about 1am. System 15 starts picking up around 1pm to midnight before it drops off again. The drop on System 15 is much more gradual than on the other systems with the difference between peek usage times and lull

times being very small compared to the other systems. System 20 sees usage around 7am but the real workload starts around 12pm and goes until 12am. System 23 has a dramatic spike at 11am which continues as an elevated workload until around 1pm. It then falls back down and stays much closer to the mean for the remainder of the day

System workloads tend to vary depending on when in their lifespan they are most heavily used. System 8 is most heavily used towards the end of its life, System 15 is most heavily used towards the end of its life, System 16 is most heavily used during the beginning of its life, and System 20's usage is fairly even throughout its lifespan with noticeable spikes happening at seemingly random times. System 23 is somewhat unique in that there is a noticeable increase in usage towards the middle of its life. This has been attributed to the reclassification of the system from a computer science system to a general science system that was open to a larger community. This reclassification also decreased the number of large jobs running on the system due to the fact that researchers essentially have to "share" their system more. Towards the beginning of its life, before the reclassification, the total number of CPU's requested per month was only slightly lower than the post-reclassification measures.

4.3 Analysis of Queue Efficiency

The intent of this section was to quantify and measure the efficiency of the queue system employed by LANL for the dispatch of their jobs. To do this, we first took a count of the number of CPUs used for jobs during the lifespan of the machine. We then divided that count into what hour of day the CPUs were being dispatched and similarly when they were finished being used. This number was then divided by the total count of CPUs used to result in a percent. We then subtracted the CPUs finished by those being dispatched for their respective hours to yield a rough measure of efficiency. Essentially, what this provided was either positive values, which represented more CPUs finishing jobs then being dispatched during that hour, or negative values which meant the opposite. For example, of all the CPUs that have ran on System 23, 7.75% of them were run at 3:00pm. However, at 3:00pm only about 4.95% of the CPUs are finishing jobs. This results in a 2.8% deficit of CPUs becoming free and so the system is becoming more heavily utilized. We can then look at similar trends across the whole 24 hour day and find when the machine is most inundated, or else, has the most free CPUs. By using this method to view queue efficiency, we can easily tell how far each machine has been from the "ideal" efficiency for its lifespan. This ideal efficiency is that the same percent of CPUs are becoming free as are then being used, which would result in a difference of zero across the whole span of the day.

After calculating these measurements, we then represented the data in bar graphs which yielded some interesting trends. The most obvious trend of note was that, with the exception of machine 16, all of the machines became much busier around noon or 1:00pm each day and then would not return to positive counts until around 10:00pm to Midnight. This was not unexpected because similar trends were observed in usage data where more jobs were being submitted during these hours. What this tells us is that it would be more efficient to run some of these jobs normally submitted during the day at less busy times later that night. For example, if you anticipated your job to run for about 7 or 8 hours, you might want it dispatched at 5:00am so that it will be running during the hours that are continuing to accumulate free CPUs. This would result in less strain on the system and perhaps a faster run time. System 16 was unique, however, because while the other machines had a very diurnal period with one high and one low, 16 proved more semi-diurnal. At around 1:00pm, 16 becomes particularly busy until about 6:00pm. Then at 9:00pm CPU use suddenly steeply falls off for three hours and then returns back into the negative until 7:00am where it experiences one final positive count of jobs submitted. One idea that we drew from this trend is that it might be wise to submit a lot of shorter jobs (less then 3 hours) during the huge spike from 9:00pm to 12:00am.

4.3.1 User based analysis

One of the more interesting aspects of the usage data is the field of unique user IDs attached to each job submission. This allows us to move past analysis of generic jobs and characterize both the work and the people on these systems. Within the realm of usage characterization, looking at the users who are more dominate on a system gives us an idea of the distribution of system resources, different user roles,

and characterization of the typical jobs certain users run on the system. For the purpose of our analysis, we have provided a categorical definition of what constitutes a prolific user for each system. A prolific user is defined as any user that uses 15% or more of the average allocation of total time for the system under perfect conditions. An illustrative example is on a system of 100 users, each user would only use 1% of the total time of the system in a perfect world where all users are equal. A prolific user for this system would be someone who uses 15% or more of the total time of the system. This measure was used to distinguish prolific users from non-prolific users on each system. Once this distinction was made, we then used other measures such as the percentage of jobs submitted, the average job length, and the average wait time, to further characterize the data. It should be noted that for sake of simplicity, all time values were rounded to the nearest second and all percentage values were rounded to the nearest hundredth. Due to the scale of the data provided, this should be an immaterial practice.

We also implemented a measure of our own creation in an effort to distinguish between jobs of the same length that require varying numbers of processors. This measure was defined to create a user-based percentage for the entire system. This measure will be referred to as CPU weighted job length percentage for the rest of the analysis.

Upon initial observations, the prolific users on each system are disjoint from each other with the exception of user 9432. This user is the sole prolific user on System 15 and is listed as one of the 16 prolific users on System 16. The likely explanation of this is that since System 15 is relatively smaller in both user base and system resources than other systems, user 9432 was a part of the project that was the main focus of System 15, but did a large amount of research on System 16 as well.

In general, the prolific users on all systems with the exception of System 15 dominate the resources of the system. The aggregated measure of total time of the system used by prolific users for each system tends to be approximately 50% with this measure reaching as height as 70% on some systems. Due to this system resource domination, analysis of the prolific users is not only valid but valuable to understanding the behavior of each system.

System 8: Analysis of System 8 yielded a list of six prolific users, all of whom use a significant amount of system resources. The top two prolific users, 2820 and 4785 respectively, are of particular interest, and quite evident in a visualization of usage for System 8 (see Figure 1). User 2820 used 23.35% of the total time of the system while submitting only 1.52% of the jobs with an average job length of approximately seven hours. This implies that user 2820 was running large scale jobs that are indicative of complex computation. While this is expected with supercomputer users, the juxtaposition between user 2820 and 4785 is very interesting. While user 2820 seems to uphold the standard image of a supercomputer user, user 4785 does not. User 4785 was responsible for 18.54% of the total time of the system while submitting 43.05% of the jobs on the system. This observation combined with an observed average job length of approximately twelve minutes indicates that user 4785 might operate in some form of a support role. Possibilities of what this support role could entail include creating smaller and less computationally complex jobs for use within a larger research project, performing small scale research on the system itself rather than using the system as a tool for other research, or acting as a system administrator and dealing with system upkeep and troubleshooting. However, one interesting observation for System 8 is that nearly all nodes have a large percentage of all one CPU jobs as seen in Figure 4. The full list of prolific users for System 8 can be found in Table 5 on page 22.

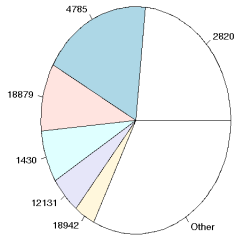

Figure 1: Prolific users for system 8. Notice user 2820.

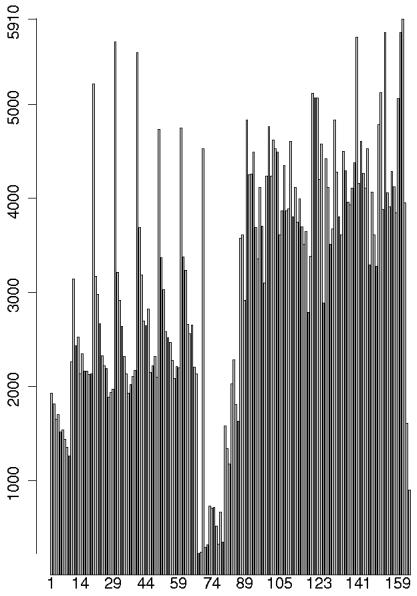

Figure 2: Plot showing count of one CPU jobs on System 8 versus which node they executed on (notice the very smooth distribution of jobs).

System 15: As discussed earlier, System 15 is a relatively small system that was focused on a single research project for the majority of its lifetime. User 9432 is the sole prolific user with 21.57% of the total time of the system and 12.83% of the jobs submitted. The average job length of user 9432 was approximately four hours with an average wait time just under an hour and a CPU weighted job length percentage of 20.59%. This user was most likely one of the research leaders for the project due to the large amount of resources utilized on such a small system. The existence of a single prolific user on a system of this size is expected.

System 16: Analysis of System 16 yields 16 prolific users and, as discussed earlier, is one of the largest systems we researched. The most interesting user out of these 16 is the third most prolific user, 2291. User 2291 exhibits many of the same characteristics as user 4785 on System 8 in the fact that it submits a large portion of the jobs submitted (35.53%) with an average job length of approximately ten minutes. It follows from this that user 2291's role in a research group is most likely focused on support rather than original work much like user 4785's. The complete breakdown of prolific users for System 16 can be found in Table 6 on page 23.

System 20: System 20's user analysis yields 13 prolific users all of whom use between 2% and 8% of the total time of the system. Each user normally submits less than 2% of the total jobs of the system with an average job length of one to two hours. These measures indicate that all of the prolific users on System 20 fit the profile of researchers that execute large-scale, resource demanding jobs rather than jobs that perform support functions. This further is demonstarted by jobs requesting only one CPU as illustrated in Figure **??**. Further more, all System 20 prolific users are listed in Table 7 on page 24.

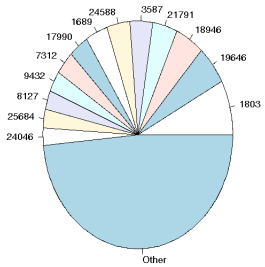

Figure 3: Prolific users for system 20.

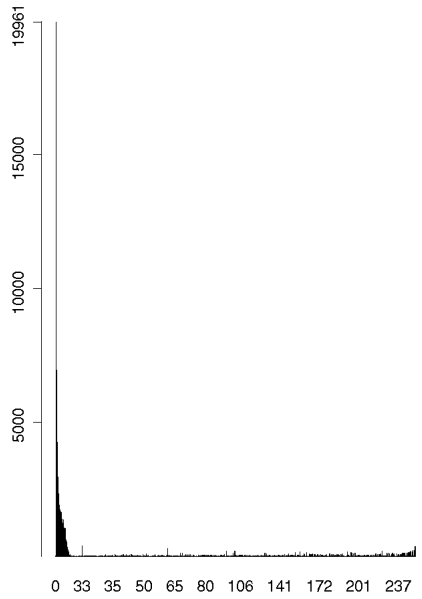

Figure 4: Plot showing count of one CPU jobs on System 20 versus which node they executed on. Notice the very large low numbered node distribution (and slight recovery for very high numbered nodes).

System 23: Analysis of System 23 uncovered 11 users that fall into our definition of prolific. With the exception of the most prolific user, user 19573, all the prolific users exhibit the characteristics of a large scale researcher (much like the prolific users of System 20). User 19573 exhibits these same characteristics, but is distinguished from the other prolific users by occupying 31.50% of the total time of the system. For reference, the next most prolific user only uses 6.39% of the total time of the system. This domination of the total time of the system by user 19573 indicates a more senior classification than other users (such as a research group leader, senior engineer, etc.). The complete list of prolific users with their usage measures can be found in Table 8 on page 24.

5 Event Characterization

In a similiar vein as the usage data, we performed analysis on the automated event log data for System 20 in an effort to provide basic characterization that aids in future analysis. Due to the dramatically smaller size and the nature of the automated event log data, the majority of the characterization performed is limited to temporal analysis. As with the usage data, we analyzed the occurance of events on a daily and weekly basis in an effort to find an underlying pattern within the data. With regards to the number of events by day of the week it was obvious that Thursdays were vastly more prominent than other days. Friday was the only day with comparable counts to Thursday. The other five days of the week all seemed very similar to each other. This trend is interesting in that it is not what we expected to see after having such clear distinction between weekdays and weekends in both usage and failure data.

We also divided the counts of events into the hour of day in which they occurred. Although this data proved very homogeneous, there where a few noticeable trends in the data. The four highest counts for the event data occurred between 1pm and 5 pm. This is interesting because much of the event data has to do with temperature warnings and 1pm to 5pm is the hottest time of the day. Similarly, some of the lowest points were from 2am through the night. These observations seem a little more in character when compared to the usage data, but again, these trends are not nearly as pronounced as in the usage.

We also preformed some time series analysis for the occurrences of events. This very clearly showed that there where many more events at the beginning of our data set which then leveled off after about six months. After that, the data seemed fairly level with only a few spikes for the remainder of our data. It is worth noting that three of the four spikes noticeable in the monthly division on the time series plots occurred during the third month of the year. When related to the fact that these months also correlate with increasing temperatures it might not be hard to imagine that these event spikes are caused by a cooling system that is still lax from the previous winter months.

6 Usage and failure correlation

Apart from basic characterization, the main goal of our project was to find a correlation between usage data and failure data. This is an extremely important portion of our analysis due to the real world implications it could have. If a strong usage-failure

correlation was detected, it is quite possible that the resulting analysis could yield information that would aid in the future design of both data backup strategies and supercomputer usage protocols. Due to its importance, we analyzed three distinct areas of usage-failure correlation: a numerical attribute correlation between usage and failure on a nodal and temporal basis, an analysis of the ratio of user time vs. system time with respect to failures, and graphical overlay method of temporal data for usage and failure rates.

6.1 Usage and failure attribute correlation

Several methods were used to do a basic attribute correlation between usage and failures. Breaking up the failures on a nodal basis and looking at the usage of each node over the life of the system, breaking up the usage and failures on a per week basis and looking at the correlation between them, and lastly looking at the failures on a per users basis. This analysis was predominantly done with the use of a correlation matrix and correlation matrix plots.

By Node: The nodal analysis was done by comparing different usage characteristics with failures summed or averaged over the life of the systems. For the usage characteristics the total number of jobs submitted, average job length, average CPU seconds user, average CPU seconds system, average total time, and average number of CPUs were used to describe the usage of each node. These usage characteristics were then correlated with the number of total, hardware, CPU, software, site software, vender hardware, disk failures, memory, cooling, interconnect, and PCI failures on each node. From this analysis no global trends were found but some interesting system specific trends were noticed. System 15 only had one node, making it impossible to use in this analysis with this method.

While no global patterns were found with the nodal analysis, interesting trends were uncovered on System 8, System 16, and System 20. On System 8 the best correlation between system usage and failures that was found was 0.26 between average CPU seconds user and the total number of failures. This is a low correlation value, but it still proves to be very statistically significant using a p-test measure. Another statistically significant correlation was between average number of CPUs requested and CPU failures with a 0.25 correlation. System 16 had much better correlations across the board with some correlations being as high as the 0.73 found between the total time and software failures. System 20 also shows statistically significant correlation between software failures and total time. While System 23 it had some extremely high correlations, none of them which statistically significant due to the fact System 23 only had 5 nodes.

For a more detailed look at the correlations see the subset of the correlation plots attached in the appendix.

Time Based: The time based analysis was done similarly to the nodal analysis except the usage and failures were tallied on a weekly basis and a limited subset of usage characteristics were compared. The usage and failures were compared on the total CPU seconds user, the average job length, the ratio of user and system time, and the total number of jobs verses the total, hardware and software failures. The only correlation of note across all systems was CPU seconds user and number of hardware failures. This correlation ranged between .17 and .33. Other than that there was little visual correlation between usage and failures on a per week basis.

6.2 User Time Vs System Time

A hypothesized possible failure predictor depends on user processor time and system processor time. It has been hypothesized that as correctable errors take more time in the kernel[WL97], as the system corrects the error (or attempts to), this can lead to an increase in kernel time in relation to user time. Investigation of this hypothesis was fortuitiously possible thanks to the system and user time data available in the usage tables. The next question was as to how long an increase in system time might continue and vary before a failure is actually seen. This data would be useful for prediction, and whether level or rate changes are most indicative of possible failure. Example plots for various time intervals of failures versus the ratio of user time to system time can be seen in Figure **??**. These show an obvious pattern between the failures and times, but are for the

yearly data, which is largely statistically insignificant. Looking into the data aggregated quarterly, monthly, and weekly show largely lowering correlation values when analyzed mechanically. Unfortunately, graphical investigation becomes difficult at these levels, as the large increase in data points is difficult to decypher. Certainly, this larger data set should be more useful for analysis, containing the data to generate these yearly averages. To attempt gaining this further understanding from the data, autocorrelation plots should be used to determine possible lags, and to look for how reliable he data is across various systems.

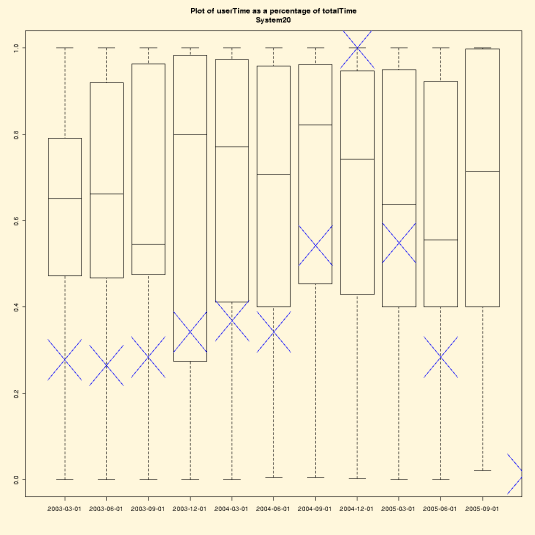

Figure 5: Example plot of user versus system time as boxplots with normalized hardware failure counts.

6.3 Temporal system usage/failure rates

Two approaches were used to look at system usage over time. One approach was bar plots of usage overlaid with line plots of the number of failures per month. The second approach involved looking at the ratio between different types of failures and usage over the life of the system. Both of these approaches led to discovery of some expected, but still interesting trends.

Graphical Overlay: The most prevalent global trend noticed was that during the start-up time for the given system, the failures are at or above the average failure level for the life of the system, yet the usage during this time period is hardly noticeable. The presence of this increased failure period has been discussed in previous literature and was expected [BS]. The presence of a similar but inverse usage period that coincides with the increased failure period makes anecdotal sense in that when a system is set up, it is expected that there will be a period of time in which the system administrators and the users have to get used to the new machine. This would discourage people from using the system due to its unfamiliarity and as mentioned previously, an increased number of failures will occur due to the system administrators not being familar with the system [BS].

A challenge with this data and method was that most of the the trends were system specific. Examples of this include System 8, where the system usage and failures appeared to independent outside of the system start-up time. System 15 appeared to have a slight inverse trend between system usage and failures. On System 16, there appeared to be a strong correlation between system usage and hardware failures during the second half of the system's life. Similar to System 15, System 20 has a noticeable inverse correlation between usage and hardware failures and software and unknown failures appear to be uncorrelated. On System 23 there are not any noticeable trends between usage and failures except at the end of the systems life, where there is low usage but the number of hardware failures remains roughly constant while both software and unknown failures drop off.

Usage vs Failures Ratio Plots: Apart from the bar plot overlays, we also graphically analzyed a time series plot of the ratio between the number of failures that happen with the amount the system was used for given time periods. This was done in an effort to capture an overall "health" metric of the system, in that if the ratio is high, there are more failures occuring per usage measure and vice versa.

Across all systems it was evident that during the start-up period the system was less reliable, but a similar trend at the end of a systems life was not observed. The general trend across systems was that during the startup period the systems were unreliable and for the next portion of the systems life they were somewhat reliable. Almost all systems had a spike during the middle of there life where there reliable drooped for a period of time and then the machines best reliability was near the end of a systems

life. These trends were especially noticeable when observing software failures.

Like many other parts of our work, our analysis uncovered system-specific trends more readily than global trends. System 8 had the most variability of reliability, but followed the same general trend that the reliability was better later in its life. System 20 followed the general trends except it had a reliability drop at the end of its life due to an increase in hardware failures. System 23 is a perfect example of the above stated general trends.

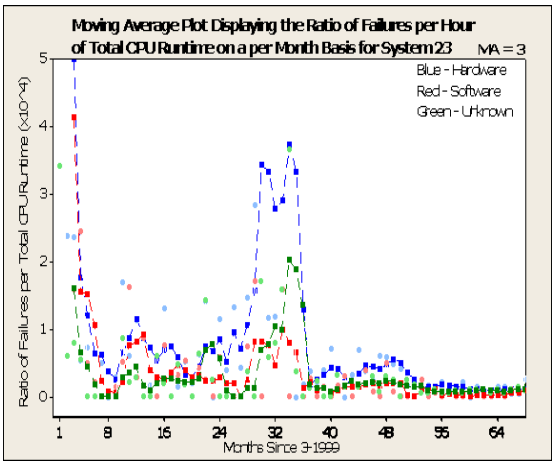

Figure 6: Ratio of failures per hours of cpu time System 23.

Another thing we tried to do was correlate the reliability between systems. This was approached two ways: one was looking at if there were any correlations between systems failures on a calendar basis and secondly the errors were correlated as months since the systems introduction. The following image show the errors as months from the start of the system's life and clearly illustrates the global trends across the systems.

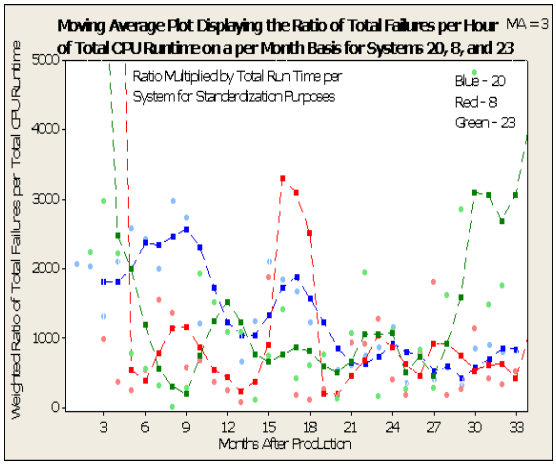

Figure 7: Count of failures by month since system installation.

A brief analysis of failure correlation over all systems was performed with respect to time in an effort to uncover any global problems (e.g. file system errors) that could be correlated to errors on all systems.

7 Location and failure analysis

7.1 Building 3-1

North-South: Failure is distinctly separated between major N-S rows. The first major row maintains a fairly steady count just below average. Row 2 proves the least prone to failure. At row 3, the count jumps back up to a level a little over the average and maintains that trend at a fairly constant level until it finally ramps up at the farthest southern rows. Among this trend in the major row distinction, there are also a number of spikes located around the beginning and middle of the major rows defined often at locations .01 and .07-.09. These spikes seem to be, largely, results of spikes in the trend of software failures. However, after reanalyzing the data with interactive nodes excluded, these spikes disappear and leave software failure to be rather uninteresting.

East-West: The failure data is far less characterized by any obvious trend when divided into east to west columns. Total failure is largely similar across the room with four rows falling well under the average count. This is likely due to the fact that, when the data is sorted with relation to total down time as

opposed to number of failures, these rows show up as the top four rows. Which is to say, they fail less but stay down longer.

Height: When looking at the total count of failures, the height from bottom to top follows a parabolic trend where its maxes occure at the lowest and highest nodes and the middle nodes form a concave center. After dividing up this failure into hardware and software failures, it becomes more obvious that the failures occurring at the bottom of the stack are attributed to software and unknown, where as the top of the stack falls victim to more hardware failures. This is an interesting division, especially when we consider that the cooling method for this room is floor vents which would leave the top of the stack hotter and perhaps in turn cause higher hardware failures.

Shelf: A number of the systems in this room have multiple nodes located left to right on shelves for each height in the stack. However, this separation seems fairly uninteresting in that there doesn't seem to be any easily noticeable trends in the failure data.

Hardware: When we just look at the failures ascribed to hardware and environment, we see that this division is, firstly, much more prominent than either software or unknown, and secondly, it contributes most noticeably to more homogenous trends in the failure data. In the N-S division, hardware is very level across each major row where as software and unknown is not. Also, as mentioned in the section concerning height, hardware failures are more prone to occur at the top of the stack.

Software: Software failures contribute a much smaller portion of the total failure data than does hardware. It is interesting however, in that it is defined largely as spikes in the data which have proven to occur almost exclusively in interactive node locations or at server nodes. Software spikes occur, in the N-S orientation, at the beginning and middle of major rows, (.01 and .07-.08) which is also where all of the interactive nodes are laid out amongst the different systems. Also, in height, software is much higher in the bottom of the stack than it is anywhere else in the stack. This might be attributed to the fact that the node on the bottom of the stack is often a server or domain server node that runs software on behalf of the rest of the stack. Consequently, this bottom node is responsible for running more complicated, and in turn, more error prone software. Finally, software illuminates a little bit of characterization in the shelf layout failure in that it has the most failures on the far right node and the least in the center. Again, this is likely due to the far right node acting on behalf of the other four nodes.

7.2 Building 1 System 8 and 20

It should be noted that Node 0 in System 20 acts as an interactive node and consequently has hugely disproportionate amount of failures to the rest of the system. Consequently, it skews other trends in the positional data and so it is convenient to discard this node. However, before completely excluding this node it might also be interesting to mention that it contains over 55% software failures as compared to about 30% Hardware failures. This property is not characteristic of the majority of the remaining nodes in machine 20, or for that matter, the nodes of other systems in this report. That being said, for the remaining sections of the positional data, node 0 has been excluded concerning failure trends. This is to help us gain insight into other trends within the data.

N-S: On System 8 the N-S data is composed of five rows which seem to represent a bowl in terms of counts of failure. When the data is subdivided into different categories, this trend appears to fit with hardware, software, and unknown failures alike. Unknown however helps accent this trend with a large spike on the furthest south row. Similarly, software has its highest value at the furthest north row. System 20, which contains 26 distinct rows, there is less of an obvious trend but there are certainty perceivable spikes primarily due to software failures at row 20 and 29 (there is no data for rows 21-28 making these two rows appear as ends of major groups and so it seems likely that these nodes also act as interactive nodes. Furthermore, the node 0 data amplifies this spike even more severely).

E-W: Because System 8 has only one column of E-W data, only System 20 is of interest in this respect. However, System 20 only has two columns and both are incredibly similar with respect to all types of failures with only software showing a slightly higher count on the more eastern column.

Height: System 8's vertical position data is composed of 37 different heights. This data seems largely void of an obvious trend, however, there are software spikes at the top and bottom and an unknown spike at height 18, around the middle of the stack. System 20 has 5 heights also with no easily noticeable trend except that the middle is higher than the other four heights.

8 Future Research Possibilities

Due to the constrained time frame that our project must operate within, we did not have time to explore all of the potential areas of the data that we feel would yield interesting results. We are including a list of potential research that could be based off of our current work and quite possible prove interesting.

- The ratio of user time vs. system time has shown potential of being a predictor for correctable hardware failures due to extra operating system time needed to detect and correct failure.
- Given more information regarding the machine rooms in which the systems are housed, our positional data indicates that a potential relayout of the systems could help decrease failures of various types.
- Due to the nature and dominance of the prolific users on each system, potential correlation seems to exist between the prolific users and certain error types. Future research could

9 Summary

Many members of the scientific community have been championing the idea of increased frequency in releases of datasets pertaining to the usage and problems incurred during supercomputing. We whole heartedly support this attitude and hope that the brief but substantial analysis we have provided encourages even more research on the subject. Below we summarize a few of our findings.

- The usage data provided by LANL, while large and created via automated means, contains some errors and inconsistencies that most likely stem from the declassification process.
- All systems experience a start-up period (in their infancy) where usage traffic is dramatically low and failure rates are substantially high.
- System usage follows distinct patterns that reflect the normal work schedule of users at LANL in both day of the week and time of day.
- There exist prolific users on each system that use a majority of the resources.
- The automated event log data does not seem to follow the same daily and weekly distribution as both the usage and failure data.
- While there were system specific numeric correlations between usage characteristics and failure mearsures, there does not appear any dominant global correlations amongst the systems
- The ratio of user time versus system time appears to predicate replaceable hardware failures.
- Spikes in the failure frequency happen around the beginning and middle of the major North-South rows in the positional data.
- Increased failure rates occur as one moves towards the ends of each stack with increased hardware failures occuring at the bottom, while increased software and unknown failures occur near the top.

10 Acknowledgments

Our team would like to acknowledge everyone who helped us in this project. Special thanks goes to Gary Girder from Los Alamos National Lab, Roman Tankelevich, Cyndi Rader, the students who participated in the peer review process, and the Solaris organization at Sun Microsystems for supporting the group with the hardware to complete this project. Extra special thanks goes to Alberto Villareal, who proved invaluable throughout the entire research process.

References

[BS] Garth A. Gibson Bianca Schroeder, *A large-scale study of failures in high-performance computing systems*.

[BS07] Garth A. Gibson Bianca Schroeder, *Disk failures in the real world: What does an mttf of 1,000,000 hours mean to you?*, FAST'07 (2007).

[Dav07] David Axmark, Allan Larsson and Michael "Monty" Widenius, *Mysql relational database*, MySQL AB, 2007.

[R D07] R Development Core Team, *R: A language and environment for statistical computing*, R Foundation for Statistical Computing, Vienna, Austria, 2007, ISBN 3-900051-07-0.

[RR00] Kenneth J. Ryan and C. Shane Reese, *Estimating reliability trends for the world's fastest computer*.

[Ste05] Jon Stearley, *Defining and measuring supercomputer reliability, availability, and serviceability defining and measuring supercomputer reliability, availability, and serviceability (ras)*, 6th LCI International Conference on Linux Clusters (2005).

[WL97] Chang-Hong Wu and Gary Lauterbach, *US Patent 5,912,906: Method and apparatus for recovering from correctable ecc errors*, June 1997.

A Definitions

- ARIMA (Autoregressive Integrated Moving Average) model: A generalized version of an ARMA (Autoregressive Moving Average) model, used for the understanding and possible prediction of future points in time series data.
- available: "An item in the condition of availability" [Ste05]
- availability: "The fraction of a time period that an item is in a condition to perform its intended function upon demand" [Ste05]
- correlation: A statistical measure indicating the departure of two variables from independence
- domain server nodes: A server node that acts on behalf of nodes within a certain domain.
- event: "Any occurrence which affects the state of an item"
- failure: "The termination of the ability of an item to perform a required function. External corrective action is required in order to restore the ability of an item to perform a required function." [Ste05]
- interactive node: A node that a user has direct access to and has to use in order to execute jobs on the system (usually denoted as node 0 or 1)
- interconnect: A hardware component that handles communication between nodes
- interrupt: "The suspension of a process to handle an event external to the process." [Ste05]
- item: "all-inclusive term to denote any level of unit, including system and component" [Ste05]
- job: "A user-defined unit of work that is to be accomplished by a computer"
- machine: A synonym for system
- machine room: A physical room used to house supercomputers or groups of supercomputers.
- maintenance: "the act of sustaining an item in or restoring it to a condition to perform its intended function" [Ste05]
- node: "A hardware component consisting of one or more CPUs and capable of communicating with other nodes in order to perform parallel computations" [Ste05]
- NUMA (Non Uniform Memory Access): A computer memory design for use with multiprocessor system where memory access time is dependent upon memory location
- outlier: A data point that is numerically distant from the majority of the data points from the same dataset.
- position: The relative location of an item within its machine room regarding North/South, East/West,vertical, and lateral orientation.
- process: Synonym for job
- processor time: the amount of time a processor spends performing a given task
- prolific: a user who uses 15 times or more of a system's aggregated total time (e.g. on a 100 user system any user who uses 15% or more of the system's total time is defined as prolific)
- reliability: "The probability that an item will function without failure under stated conditions for a specified amount of time." [Ste05]
- repair: "the act of restoring an item to a condition to perform its intended function" [Ste05]
- server node: A node that acts on behalf of other nodes to perform system tasks
- serviceability: "the probability that an item will be retained in, or restored to, a condition to perform its intended function within a specified period of time.
- shelf: A subsection of a stack in which a single node or multiple nodes can be located with the same vertical orientation but varying lateral position.
- stack: The vertical casing in which a single node or multiple nodes are housed in.

- start-up time the qualitatively deduced period of time at the beginning of a system's life where initial setup and configuration occur. This period is often characterized by abnormally high failure rates as well as low usage values.
- supercomputer: "Any group of computers that have the fastest processing speeds available at a given time. Generally speaking, the intended function of a supercomputer is to quickly perform computations for users." [Ste05]
- system: "A collection of components organized to accomplish a specific function or set of functions." A system is comprised of one or more nodes capable of communicating with each other to perform given tasks." [Ste05]
- system processor time: amount of processor time spent on executing system kernel code for a certain job
- strain: The state of an item in which a task or tasks require a large percentage of resources greater than the average allocation.
- time series: A collection of data points measures at successive (often uniform) time intervals
- total time: an aggregated total of user processor time and system processor for a specific job
- usage: The utilization of an item's resources towards a certain task or group of tasks
- user: A unique person or group of persons that submits jobs to a system
- user processor time: amount of processor time spent on executing user code for a certain job
- wall time: "Regular time as displayed on a wall clock" [Ste05]
- work: A generic term representing the usage of item resources to perform a given task or tasks
- workload: A measure of the amount of work being done on an item at any given time.

B Technical Overview

Though not incredibly large for modern workstations, our main data was 4,395,440 records, split up between usage data for five systems, failure data for all systems and an event log for system 20. To handle the task of achieveing fast data access and analysis we selected a relational database to hold and manipulate the data, while most group members used the GNU R statistical package to perform data analysis and visualization. One member chose to use Minitab using an ODBC connection back to the relational database. All of this software (execpt Minitab) was hosted off a Sun Microsystems workstation. The workstation was chosen for the the multiuser capibility of a Unix environment based around a central server.

B.1 Hardware

A Sun Microsystem's W2100z workstation was used to host the database used by our team, and for three users, to also run the GNU R statistical package. The system was equipped with two 2.6 Ghz AMD Opteron CPUs and with 12 GB of system memory, and one 146 GB SCSI Ultra360 disk. The machine was installed with Sun Solaris Nevada build 55a.

B.2 Software

The database our team chose to use was MySQL, version 5.0.41-log, as built by the Blastwave project[**?** , Blastwave] The database configuration was tuned from default to more effectively use available system memory. The statistical packages chosen reflected group member comfort with various software packages. One member chose Minitab due to his familiarity with the package, and his relative inexperience with a Unix machine. The other three used the GNU R package version 2.5.0 (2007-04-23). Both Minitab and R used MySQL as a backing data store to ensure consistancy of results and the speed benifits of allowing MySQL to filter and join data as desired.

B.3 Data Import

To data was provided by Los Alamos[**?** , InstituteDataLANL]n a variety of comma separated value (**CSV**) and space delimitered files. The data was massaged by a PERL script (see Appendix **??**) tuned to output a CSV file for each input file from Los Alamos. This data was then fed into MySQL. The location data was only handled in Minitab, however, and all import was done following usual import methods.

B.4 Bad Data Handling

Some data as provided appeared to be flawed or inexplicable. For this data MySQL was used to generate a new table using a query designed to remove any data felt to be incorrect. This data, however, was a very small percentage of the total. Our criteria for determining what data was bad can be seen in table B.4. The total which was considered bad was quite small; out of the 4,502,519 records of the original data, 4,382,953 were considered clean, a loss of only 2.66%.

While some data appeared to have subtle errors as listed in table B.4, some data was obviously corrupted. Through which means the corruption occured was not obvious. The data passed through checks at Los Alamos to ensure it contained no sensitive data. These checks may have introduced the corruption, as such some records in the data were also removed. For the system 16 data, there were 75 records removed. All records removed consisted of simply two fields one ”555” and one ”0”. Similarly, the system 23 data had one record corrupted with the ASCII data ”JOB_FINISH” corrupting one or more fields, and one record with a fractional num_cpus value. Similarly, some failure data records are missing any entry for the failure type, no hardware, software, network, human error, facilities, or unknown description. Advice from Los Alamos was this data should be taken as having an unknown reason, on the logic all other data was valid. This data consisted of 453 records. Further more, some data needs further explaination on the meaning of various entries. Much of this information can be found in Appendix D.

Tables	Fields	Criteria	Number of Rows Affected	Disposition
All Usage Data	submit_time	After Jan. 1, 1995	156	Thrown Out
All Usage Data	dispatch_time	After Jan. 1, 1995	369	Thrown Out
All Usage Data	end_time	After Jan. 1, 1995	155	Thrown Out
All Usage Data	submit_time, dispatch_time, end_time	submit_time<=dispatch_time<= end_time	110	Thrown Out
All Usage Data	cpu_sec_user, cpu_sec_system, total_time	cpu_sec_user+cpu_sec_sys> 0.95×total_time cpu_sec_user+cpu_sec_sys< 1.05×total_time	28	Thrown Out
All Usage Data (execpt system 20 node data)[1]	num_cpus	cpu_nums=$\sum_i^{num\_nodes}$ node$_i$	11,320	Thrown Out
System 20 Event Log	event_timestamp	After Jan. 1, 1995	5	Thrown Out
Failure Data	unknown, software	unknown=”Security”	15	Recatagorized to software as ”Security Software”[2]

Table 1: Criteria for correcting and determining bad data.

B.5 Categorization

While looking at the numerous descriptions of the hardware and software failures, it was apparent that some categorization needed to be done to the data to allow further analysis. We grouped the descriptions in various categories so that the analysis could look at broader categories of failures. The result was 13 categories for hardware failures, and 5 categories for software failures. The breakdown can be seen in Tables B.5, 3.

[1]Data for system 20 grouped by node lacks cpu, and node usage for the first 40% of the node file so these were not assumed bad.
[2]Received instructions that the data was miscategorized.

Hardware Failure Catagories

Category	Description	Count
Primary IC Logic	CPU	5931
	System Controller	7
	OCP	1
	IOS CPU	16
	WACS Logic	1
Maintenance	Maintenance	734
Misc.	Other	710
Networking, Machine Access	Fddi	3
	Ethernet Switch	80
	Ethernet Cable	9
	1 GBit Ethernet Card	10
	PCI GBIT Ethernet Board	28
	Site Ethernet Switch	42
	100 MBit Ethernet Card	2
	PCI Ethernet Board	5
	Ethernet Copper Cable	6
	Ethernet Fibre Cable	1
	Ethernet Fiber Cable	1
	Gig E Switch	22
	Gig E Connection	6
	Router Board	84
	Site Network Interface	1
	Console Network Device	16
	Console Interface Module	4
	Term Server	1
Memory	Memory Module	55
	Memory Dimm	2880
	SSD Memory Module	6
	SSD Logic	1
	IOS Buffer Memory	2
	MMB	22
Disk	IOS Disk Logic	2
	KGPSA	1
	Disk Drive	272
	SCSI Controller	31
	SCSI Adapter Card	8
	PCI SCSI Controller	15
	SCSI Card	1
	SCSI Drive	40
	IDE Cable	1
	Fibre Drive	33
	Fibre HBA	40
	Fibre Raid Controller	82
	PCI Fibre Channel Adapter	100
	PCI Fibre Channel Adaptor	37
	Fibre Channel Port Adapter	6
	Fibre Cable	14
	Fibre Raid LCC card	2
	Fibre Raid Midplane	1
	Drive Cage	7
	CD ROM	1
Shared Storage	SAN Shelf	6
	SAN Appliance	1
	SAN GBIC	1
	SAN Switch	1
	SAN Fiber Cable	1
	SAN Controller	27
	SAN Disk Drive	11
	Disk Cabinet	22
Interconnect	Interconnect	113
	Interconnect Soft Error	175
	Interconnect Misc	1

Continued on Next Page. . .

Category	Description	Count
	Interconnect Switch	73
	Interconnect Interface	425
	Interconnect Cable	45
	Vhisp	9
	Xtown Board	2
Graphics	Graphics Accel Hdwr	39
	Graphics Video Card	1
	RM Board	11
	GE Board	14
	DG Board	2
	MIA	39
	IO6	14
	Ktown Board	5
Power and Distribution	GFX Power Supply	2
	Fibre Raid Power Supply	2
	Power Supply	593
	Rack Power Distribution Unit	1
	Power cord	3
	Wire Harness	3
	Module Assembly	1
Cooling	Fan	20
	40MM Cooling Fan	11
	Fan Assembly	103
	Temp Probe	5
	Heatsink bracket	1
PCI	PCI Back Plane	138
	PCI IO Module	12
	PCI Shoebox	2
Misc Boards	Node Board	915
	System Board	125
	Bach Plane Assembly	2
	Mid-plane	73
	Riser Card	1
	MSC Board	23

Table 3: Hardware Failure Catagories

Software Failure Catagories

Category	Description	Count
Maintenance	DST-Upgrd/Install OS sftw	75
	DST	1127
	DST-Upgrd/Instl 3rdParty Sftw	5
	DST-Scan for errors /scratch	106
	Upgrade/Install OS sftw	109
	Modify kernel parameters	41
	Patch Install	57
	Security Software	1
	Upgrade/Install 3rd Party Sftw	8
	Modify system config files	27
FS SW	Cluster File System	141
	Parallel File System	451
	NFS	67
	Scratch FS	32
	Vizscratch FS	1
	Disk IO, firmware and storage	48
	Scratch Drive	1
Misc SW	Other Software	1174
Vendor SW	Interconnect	113
	Network	528
	Kernel software	228
	OS	1010

Continued on Next Page. . .

Category	Description	Count
	IOS Software	5
Site SW	Scheduler Software	187
	Cluster Software	41
	Resource Mgmt System	128
	User code	94
	MPI, PVM, Array services	31
	Compilers and libraries	1

B.6 Data Import PERL Script labelImportScript

```
#!/bin/perl
if((scalar(@ARGV) < 4) || (scalar(@ARGV) > 5)){
print "Usage:\tPlease provide nodes, CPU/node, CPU index, filename, [Print Batch ID Flag]\n
exit(1);
}
$NODES=$ARGV[0];
$CPU=$ARGV[1];
$FIRST_CPU=$ARGV[2];
$PRINT_BATCH_ID=$ARGV[4];

open(FH,"$ARGV[3]") || die("Can't open file $ARGV[3]: $!\n");
while(<FH>) {
# $batchID only batch ID on System 20 data (use Print Batch ID option)
(my $end_time, my $user_id, my $num_processors, my $submit_time,
 my $suggested_start, my $deadline_time, my $dispatch_time,
 my $cpu_seconds_user, my $cpu_seconds_system, my $total_time,
 my $batchID, my $jobPairs) = split(/ /,$_,12);
my $disposition;
my %cpus;
my @jobs = split(" ",$jobPairs);
my $total_cpu=0;
for (my $i=0; $i < scalar(@jobs)-1;$i+=2) {
# Remove "d" From d## Sys. 20 Format
if($jobs[$i+1] =~ /d[0-9]*/) {
$jobs[$i+1] =~ s/d//;
}
# See If Disposition Format Of Sys. 20
if($jobs[$i] =~ /[a-z]/) {
$disposition = $jobs[$i];
# Trust The Job Info For Num CPUs
$total_cpu=$num_processors;
}
# See If System 20 Format With Node Ranges
elsif($disposition) {
# Should Fall Through To Execute Loop Once
# If No "-" In The Node Field
my $low, $high;
($jobs[$i] =~ '-')?
($low, $high)=split(/-/,$jobs[$i],2):
$low=$high=$jobs[$i];
foreach ($low..$high) {
```

```
# Boolean (1-Node Used,0-Node Not Used)
$cpus{$_}=1;
}
# We Only Want To Skip A Field If It's The Closing "/"
$i-- unless ($jobs[$i+1] =~ "/");
}
# See If Format Of Sys. 15/20
elsif($jobs[$i+1] =~ /\*/) {
($_,$jobs[$i+1])=split(/\*/,$jobs[$i+1]);
$cpus{sprintf("%u",$jobs[$i+1])}=$_;
                        $total_cpu+=$_;
}
# See If We're Using All Procs On This Node
elsif($jobs[$i] =~ /\*/) {
$cpus{sprintf("%u",$jobs[$i+1])}=$CPU;
$total_cpu+=$CPU;
} else {
# Just Incriment By One
$cpus{sprintf("%u",$jobs[$i+1])}++;
$total_cpu++;
}
}
print "$submit_time,$suggested_start,$deadline_time,$dispatch_time,$end_time,$total_time,$c
foreach ($FIRST_CPU..$NODES-1+$FIRST_CPU)
{ exists($cpus{$_})?print ",$cpus{$_}":print ",0"; }
chomp $batchID;
print ",$batchID,$disposition" if $PRINT_BATCH_ID;
print "\n";
}
```

C Tables

Table 4: System 8 Prolific Users

User ID	% of Total Time	% Total Jobs	Avg. Job Length	Avg. Wait Time	% CPU Weighted Job Length
2820	23.35	1.52	26617	30733	2.72
4785	18.54	43.05	777	18892	2.25
18879	9.63	0.63	27086	847	4.20
1430	7.39	0.19	67872	3373	0.92
12131	5.21	0.73	16832	167280	15.72
18942	3.52	0.23	33978	4306	2.97

Summary of prolific user data

Total Prolific User Time:	68%
Total Prolific User Request:	46.3%
Avg. Prolific User Job Length:	28860 sec
Avg. Prolific User Wait Time:	37572 sec
Total Prolific User CPU Weighted:	28.78%

Table 5: System 16 Prolific Users

User ID	% of Total Time	% Total Jobs	Avg. Job Length	Avg. Wait Time	% CPU Weighted Job Length
3154	5.85	1.06	16897	1148	5.13
7739	5.34	2.52	1462	3518	1.71
2291	5.13	35.54	594	314	3.67
1602	4.87	0.59	20116	3210	5.34
9575	3.75	0.34	11224	8211	0.98
1993	3.6	1.64	15696	24557	11.7
11014	3.06	3.81	2373	1910	3.11
2330	3.05	3.09	2944	2115	1.34
2220	2.72	0.69	9039	3382	0.86
4288	2.63	0.19	13560	12170	1.11
4016	2.42	0.37	10850	6837	1.23
6387	2.17	0.33	20569	660	1.22
9432	2.09	0.35	18696	5816	0.68
13686	1.81	0.2	16694	52718	0.8
5701	1.78	0.26	18636	10108	1.07
6148	1.72	0.21	13124	2772	0.72

Summary of prolific user data

Total Prolific User Time:	52%
Total Prolific User Request:	51.19%
Avg. Prolific User Job Length:	12030 sec
Avg. Prolific User Wait Time:	8715 sec
Total Prolific User CPU Weighted:	40.67%

Table 6: System 20 Prolific Users

User ID	% of Total Time	% Total Jobs	Avg. Job Length	Avg. Wait Time	% CPU Weighted Job Length
1803	7.79	0.45	12303	10217	7.07
19646	5.49	0.72	17527	19894	8.02
18946	4.99	1	13166	32228	4.89
21791	4.25	0.92	11431	27365	4.62
3587	3.87	0.57	10319	72951	3.71
24588	3.87	0.15	8755	39264	3.48
1689	3.82	2.99	6259	13553	4.14
17990	3.37	0.36	9504	7028	2.11
7312	3.33	0.33	21628	117626	3.24
9432	2.92	0.92	16091	5860	2.59
8127	2.69	0.24	10441	104217	2.02
25684	2.65	1.44	6841	13380	1.62
24046	2.48	0.45	4356	3556	2.37

Summary of prolific user data

Total Prolific User Time:	52%
Total Prolific User Request:	10.54%
Avg. Prolific User Job Length:	11432 sec
Avg. Prolific User Wait Time:	35934 sec
Total Prolific User CPU Weighted:	49.88%

Table 7: System 23 Prolific Users

User ID	% of Total Time	% Total Jobs	Avg. Job Length	Avg. Wait Time	% CPU Weighted Job Length
19573	31.5	1.06	25435	73875	10.49
7253	6.39	0.11	67872	143267	3.45
14485	6.36	1.23	21300	49524	12.08
9270	4.99	0.31	18077	8261	0.65
23805	4.87	0.2	31684	11871	1.63
8216	4.25	0.23	48286	22527	3.59
2793	2.76	0.93	43926	129515	2.74
1690	2.72	0.18	47207	10739	2.79
3257	2.17	0.34	22381	7120	0.32
2291	2	0.38	10890	3761	0.43
6607	1.93	0.18	19469	12834	0.31
6607	1.93	0.18	19469	12834	0.31

Summary of prolific user data

Total Prolific User Time:	70%
Total Prolific User Request:	5.15%
Avg. Prolific User Job Length:	32412 sec
Avg. Prolific User Wait Time:	43027 sec
Total Prolific User CPU Weighted:	38.48%

D FAQ

During our project we complied a list of FAQ to answer some of the common concerns with the data

1. Are problems reported even if the system is not running something (i.e. the off chance it's totally idle), or is the automated system only online when jobs are (so should we normalize failures over time for idle time)?

   The systems are monitored all the time so there will be reporting during idle time, however, the ability to catch problems with nodes or other gear is somewhat load dependent itself, for example we wont know if network file system is down if no-one is trying to use it etc. So for the most part I think you can assume that failure will be logged during idle time but not all failure will be detected, I would guess that most will be detected.

2. Were all the disks in system 20 homogenous? For example, all from the same manufacturer and model line (i.e. Sun uses both Fujitsu and Seagate - and Cheetah, Barracuda just from Seagate). From my lab experience failure is very different amongst different types (and would suggest some multi-modal distributions if the data is from heterogeneous disks).

   The disks are all same model and manufacturer but they are only the internal node disks. Essentially no user data ever hit these disks, they were used for booting the nodes and node operation. Users could write to the /tmp on the local disk but it would be wiped after their job so almost no one ever did. The systems were not booted often either. There was a global parallel file system attached to this system and that hardware/disks etc. is not represented in this data, so the failure rates for these disk drives is not representative of really heavily used disk systems. This is evident in the CMU disk

failure paper where our failure rate was closer to actual manufacturer specs than any other site, other sites had much worse failure rate because they were reporting failure for disks that were actively used for applications.

3. I'm not sure why the break up of System 20's data between nodes and domains both showing very similar data.

   (a) Was the system only reservable in domain sized chunks (or only available as such for some period of time)? It doesn't appear so, as I see jobs with requests for 200, 252 CPUs, etc.

   The system was managed in clumps (this is typical - hierarchical mgmt mechanisms), there were 32 node clumps of nodes. Node 0 and 1 for each domain were "manager/file system" nodes, so OS images and other things were pushed from these nodes. Node 1 was suppose to be the fail over for Node 0 in each domain for these "mgmt server" functions. This sometimes worked and sometimes didnt, but failure of these nodes could spell disaster for the other nodes in the domain. The nodes in a domain also shared a common management network switch and other common parts like you could guarantee that you were on the same interconnect switch meaning that latency between nodes in a domain was slightly better than nodes from different domains, especially for collective operations (all reduce etc.). The nodes were not scheduled in total domain sizes, you could ask for any number of nodes, but the queue mgr did have many queues setup and typically a queue would front several domains worth of nodes, and it would try to give you consecutive nodes in a domain if it could. Additionally, some domains were special - one of them (maybe two, I cant remember which) were behind an interactive job queue, so people could do interactive debugging and the like, others were totally behind batch queues. There is probably different failure rates for these domains I would guess.

   (b) I see that both files start with the same entry, so for completeness, would it make sense to take the domain based data and use it for a lower resolution usage, and the available per node data as a high resolution usage?

   It should be the same job records with the main difference being the "where did the job run on the machine" which I think is the last few fields. In the domain file it should say x nodes in domain 0 and y nodes in domain 1 4 * d0 5 * d1 . In the node file it is node numbers in the "where did the job run".

   (c) Does batchID tell me anything? Does a job keep it's ID, so an aborted job would have the same ID? How about a job run successfully twice?

   I would attach no significance to batchID, I expect there will be duplicates, probably restarting after a system restart or after an upgrade or some such thing.

4. The positional data has what appear to be grid coordinates (N/S,E/W), is this correct? Does a machine room start at 0,0?

   Correct

5. We notice that jobs get submitted in large groups. For example, on system 8 of 763,293 jobs there are only 526,372 distinct times, and on system 20 of the 489,376 jobs there are only 458,014 distinct times. Any institutional reasons you can think of for this?

Sure, a user has gotten a job stream working and is ready to start a large number of runs for a parameter study or is doing a bunch of 1d or 2d runs as a follow up for a 3d run, or the like. The write a script that loads up the queue with all they work they know they want to do. So the script submits a few hundred jobs with different parms. It is just a way of work for the users.

6. Machine 23 was decommissioned on December 2004, but we have 1,913 jobs from the usage data which were submitted after 12/1/04, are these bogus, or was the decommission date just a date for some users to move on?

   Not bogus, queues were closed to general users, but there is sometimes the really important user that just HAS to have a bit more time to get that last study done to write that award winning paper, or to move data off of the machine last minute etc.

7. System 23 seems to have been becoming really busy after 12/01, but around 10/03 its use suddenly and dramatically dropped off, yet the machine wasn't decommissioned until 12/04 is there any thing which you might be able to relate on external factors affecting this?

   Machine 23 was not a weapons code machine, it was used for more open science type users, so it is not terribly representative. Its early days were dominated by computer science type people trying out new computer science things. It was run by a computer science org (not by our production computing org) from 1998 to 2001. In 2001, the machine was opened up to regular open science use in 2001. In 2003, a much much larger system was put in (not in the data) and the open science users migrated to that platform quickly except for a few die hards. This sort of thing happens on open science machines, but typically doesnt happen on our weapons program machines.

8. The system 20 event log has a "handled" field which appears binary 0-false, 1-true, however, there's a -1 value which appears 3,165 times. Further more, what handled these events? How? Also, is this the log which alerts administrators to the problems of system 20, or is there a different system which does that, or other data for that which isn't included?

   Ok, the documentation for the system in question has handled int whether event was handled (1) or not (0) So this is undefined in the documentation. Scripts/executables were registered to handle event types, 1 is there was a registered script/executable that was found and run when the event occurred 0 is no script was registered or found to run during the event. What was actually done was dependent on the script/executable, could be logged on syslog, could be started a fail over action, could be a no-op, I dont have the scripts/executables so I dont know. We know the event occurred and either started or didnt start some handler that could have done something.

9. I've noticed in data for systems 8, 15, 16, and 23 users having entries where num cpus is zero and indeed no nodes are in use for the job, even though the job is more than 10 minutes long. Is this bad data?

   I would toss those records, I can't explain them.

10. Similarly, we've noticed dispatch times of 0, 1, and 4 the latter two only on system 20. There's only about 250 entries with these, the rest are all epoch based time-stamps. Is this bad data, or was that a dispatch time of 0, 1 and 4 seconds?

    I would toss this as well, this is defined in the documentation as a time stamp.

11. Is the failure data labeled num interconnects a summation of PCI back planes, infiniband and network links, etc. or what type of interconnect is described there?

External interconnect links per node (like 1 infiniband or 2 - or 1 myrinet or 2 etc.)

12. Is the date and time in the failure data GMT, or New Mexico time, and does it reflect daylight savings, if New Mexico? Likewise, are the Unix epoch times in GMT or NM time?

    Mountain time (New Mexico). Yes clocks were changed for daylight savings on those week ends.

    For events: UTC This denotes a UTC time value stored in an integer field. Client programs should convert time to local time and output the result as a string.

    For usage this is what the documentation says: Time the event was logged (in seconds since the epoch) - not sure if this is Mountain time or not I think it is Mountain time since the scheduler works based on local time for releasing jobs, but I cant be totally sure though.

13. System 23 sees 40% of its failures on Tuesday, any reason for this? Also can you tell us what DST means for system 23 under the software description, or what the maintenance was, as both were primarily Tuesday events and comprise 308/1128 failures for the system.

    DST is dedicated system time for putting on maintenance to the OS or other maintenance activities, DST was scheduled for this machine on Tuesday. DST did kill jobs that were running at the time - signals were sent to apps but it did take the machine down for scheduled maintenance.

14. There are 906 failure records with no HW Desc., SW Desc., Facilities Desc., Human Err. Desc., Network Desc., nor Unknown Entry. Should I map these to unknown or are they potentially bogus?

    If the date/time and system/node info is correct map them to unknown

15. The system 20 event data has in the subsystem name column some things which are very non-obvious to us, or could possibly explain more, if we knew more, can you give us an idea on these:

    (a) node-ms0

    (b) node-ms0-C

    node-ms0 is a management station the cluster had a management station used for managing the cluster, so if there is an event with that in it, it is changing of state of something on the management station I don't know what the -C stands for but it still has something to do with the management station.

16. System 20 Event Log

    (a) Sub-System: "full"

        I think the full denotes when a partition was blocked or not. Blocked means nothing could be dispatched on that partition; unfortunately partition information is not consistent for the life of the machine. It changed from time to time, so you can say that some part of the machine was blocked to new jobs, but you cant correlate that to what part of the machine. These records may be somewhat useless. Additionally you can't tell why it was blocked. It could have been some administrative action or something else. I think this is probably useless.

    (b) Sub-System: "[0-9][0-9][0-9][0-9]"

This is likely the process ID of a command running on the system. Likely, they were install, reboot and othercatastrophic commands.

17. Can you provide us with which systems were on the se interconnects so we could correlate which systems were impacted: Interconnect-[0,1]N0[0-3]

    Interconnect is the low latency interconnect this machine had 2 so interconnect0 is the first full fat tree interconnect1 was the other so every machine had 2 interconnect cards There were 4 low level switches on each network (first phase of fat tree) 0,1,2,3 so interconnect0-2 is interconnect fabric 0 switch 2 interconnect1-0 is interconnect fabric1 switch 0

    Now I think (not sure but you can probably tell this) switch 0 should be nodes 0-63 switch 1 should be 64-127 and so on up to node 255 (but I could have that backwards too, not sure but you can probably tell from the data.

    So gigeN where n is 0-something is a gige switch logging an error N designates the switch number, there is also a mapping to nodes but I dont know what it is for sure, I think it is node 0-31 gige0 and 32-63 gige1 but I could be off on that

18. System 15 has about 6 months of usage before the first failure record (usage began on 8/9/04, first failure was 12/2/04) is this data incomplete, or was system 15 simply very reliable?

    System 15 is an oddball, it was not managed by our production systems team until later in its life 6 months or so after initial usage, but it was scheduled by our production scheduler, so we dont have failure for this machine during its first few months of usage, even though there were no doubt failures.

19. In the failure data seen before seems to be "No" for all the records in the file, any reason for the field to be there, if they're all "No"?

    Well, there was suppose to be some more information where one failure was related to another but the data was very unreliable, so I would not use that field no matter what it says. You should be able to see related errors by setting a time window and seeing where there are large clumps of things that went down, that is an indication that some common hardware died or something.

20. What happens when a long job dies and restarts from its last checkpoint as far as the usage data is concerned? Is it a new job, or does the old job just continue, or does it depend?

    If the job dies, MPI will normally abort the communications which should cause the job to exit out of the system and the next job runs in the queue based on the queue mgmt structure at the time. Most likely it is the same job and it picks up from the last restart, but is up to the user to do that. Most do it that way. It is possible that a job may fail in such a way that it hangs (rarely but it does happen), in that case it would run out of time on the allocation it has and then be kicked off and then we are back to the queue mgr.

21. Also what are the meanings of the unknown field being set to "Security" (there are 15 instances of this)?

Those should actually be in the next column and it was a security software related interrupt.

22. We have been looking a little at simultaneous failures in building 3 room 1 and in a couple occurrences we have noticed that when, say, a power failure occurs the first node (node 0) on many of the different machines dies but not the rest of the nodes.

    Is this because, assuming a total power outage, only the first nodes would be in the report on behalf of the entire machine or is there another underlying reason? If power is taken down or a total loss it is reported on node 0.

    If power is taken down or a total loss it is reported on node 0.

23. It seems system 14's failure data has node numbers offset by 129 (i.e. its range is 129-255) should we offset these back down to zero?

    These were the actual node numbers, this machine physically had 256 nodes but the first 1/2 were used for I/O function, so they were not part of the user computational portion, the 2nd half were used for user jobs, so the numbers are actually correct.

24. There are event log entries of the form: "Fan speeds (3534 3534 3375 4655 3497 3479 are these a measure of various fan speeds (in RPM)?

    Yes, but if it doesnt say what the equipment is, I dont know what it was, but that would be fan speeds for the 6 fans in some box.

25. System 23 is reported to have 5 nodes with 128 CPU's each but with only a total cpu count of 544 (as opposed to the calculated 640).

    System 23 had 5 nodes. The first node had 32 processors, and the other 4 had 128 processors.

26. How were the varous machines used (i.e. open science, or weapons research)?

    Systems 8, 15, 16, and to a great extent 23 were all open machines (not classified) and were not weapons science machines for the most part (except for 23 a little bit)

    - System 8 Open science for the most part, but only for smaller jobs, many used it as a development cluster
    - System 15 Open science, one primary project
    - System 16 Open science, run by non production group for a while at first
    - System 20 Combo of open science and weapons work that could be done in the open Combo of open science and weapons work that could be done in the open also with some open graphics content

27. What happens when a long job dies and restarts from its last checkpoint as far as the usage data is concerned? Is it a new job, or does the old job just continue, or does it depend?

    If the job dies, MPI will normally abort the communications which should cause the job to exit out of the system and the next job runs in the queue based on the queue mgmt structure at the time. Most likely it is the same job and it picks up from the last restart, but is up to the user to do that. Most do it that way. It is possible that a job may fail in such a way that it hangs (rarely but it does happen), in that case it would run out of time on the allocation it has and then be kicked off and then we are back to the queue mgr.