Towards Systems Level Prognostics in the Cloud

Assuring Availibility and Quality of Service of Cloud hosted Systems

Budhaditya Deb, Mohak Shah*, Scott Evans
GE Global Research
Niskayuna, NY and San Ramon CA*
{deb, mohak, evans}@ge.com

Manoj Mehta, Anthony Gargulak, and Tom Lasky
GE Healthcare, Life Care Solutions
Milwaukee, WI
{Mehta, Gargulak, Lasky}@ge.com

***Abstract*—Many application systems are transforming from device centric architectures to cloud based systems that leverage shared compute resources to reduce cost and maximize reach. These systems require new paradigms to assure availability and quality of service. In this paper, we discuss the challenges in assuring Availability and Quality of Service in a Cloud Based Application System. We propose machine learning techniques for monitoring systems logs to assess the health of the system. A web services data set is employed to show that variety of services can be clustered to different service classes using a k-means clustering scheme. Reliability, Availability, and Serviceability (RAS) logs and Job logs dataset from high performance computing system is employed to show that impending fatal errors in the system can be predicted from the logs using an SVM classifier. These approaches illustrate the feasibility of methods to monitor the systems health and performance of compute resources and hence can be used to manage these systems for high availability and quality of service for critical tasks such as health care monitoring in the cloud.**

Keywords—Systems Prognostics, Cloud Systems

I. INTRODUCTION

Prognostics and Health monitoring (PHM) is a vast field sometimes intersecting with fields of condition-based maintenance and monitoring and diagnostics – see [1] for a good overview of the field and [2]-[5] for surveys of the state of the art. Prognostics from pro-gnosis, means "knowing ahead" and is the heart of GE's recent thrust into analytics – we seek to have no unpredicted failures of systems and have made significant investment into analytics technologies towards this end [9].

Prognostic algorithms are generally either data driven – formed by mining historical data, or physics based – driven by physical models that estimate system response. In the domain of Cloud type of computational systems both type of prognostics algorithms have been employed – the former in mining logs for prediction of failure, the later in assessing electronics equipment for physical predictors of impending loss (see [6]-[8] for results related to a large data center prognostics project). However relatively few papers have looked at systems level prognostics for cloud based critical systems. Furthermore, large-scale systems that aim to connect edge devices to the cloud are relatively novel system concepts which directly enable data access via client devices as well as potential aggregate analysis. Consequently, availability and quality of service assurance are not just extremely important but also acquire new meanings as a result of the scale, complexity and additional dependencies introduced by various data transmission phases in the end-to-end system. Hence, detecting and localizing QoS degradation and loss of availability is non-trivial and pose significantly higher challenges.

In this paper we explore data driven prognostics of large computer systems using publically available data sets (logs) that we mine to form models capable of detecting impending failure. This is a first step in determining whether the QoS and log data from these large-scale systems are amenable to aggregate analysis towards building a quality assurance framework. These evaluations are also aimed at highlighting the requirements in terms of system monitoring and identifying hardware and software sensor requirements so as to enable QoS status monitoring. Note that while the quality requirements for such large-scale systems may take similar form as traditional systems (e.g., service response time), the composite QoS behavior metrics and characterizations need not be the same. Often, as illustrated by the experiments below, the QoS in such cases results from complex interactions of quality parameters.

The parameters and the associated dependencies are sometimes well studied for specific parts of the system (e.g., data transmission from edge devices to a central server via wireless routing). However, their inter-relationship with the rest of the components in the stream is understudied partly because these system concepts are still in a nascent stage of design and deployment. The experiments in this work are aimed at confirming whether inter-relationships between various quality and performance parameters can be studied together to correlate them with the QoS. To this end, we study two datasets. A web services data set **Error! Reference source not found.-Error! Reference source not found.** is employed to study whether a variety of services can be grouped in service classification based on their quality parameters. The Reliability, Availability, and Serviceability (RAS) logs and Job logs dataset [10]-[18] is employed to determine whether the RAS and system job logs can act as reliable indicators for loss of service due to fatal errors in the system.

The rest of the paper is organized as follows. In section II we describe the data sets that are explored. In Section III we describe the machine learning algorithms we use to train and apply system prognostic models. Results are described in section IV, and the paper concludes with recommendation for next steps in Section V.

TABLE I. RAS LOG FIELDS

RAS log Fields	Comments	Example
RECID	is the sequence number	1371890
MSG_ID	Source of the message	CARD_0411
COMPONENT	Application, Kernel, MC, MMCS, Baremetal, Card, Diags	CARD
SUBCOMPONENT	Functional area that generated the message	PALOMINO_S
ERRCODE	Fine-grained event type information	DetectedClockCardErrors
SEVERITY	Debug, Trace, Info, Warning, Error, Fatal	FATAL
EVENT_TIME	Start of the event	2008-04-14-15.08.12.285234
LOCATION	Location of the event	R-04-M0-S
MESSAGE	A brief overview of the event condition	An error was detected by Clock Card: Error=Loss of Reference input

TABLE II. JOB LOG FIELDS

	Header
	; Computer: Blue Gene/P
	; Installation: Intrepid - Argonne National Laboratory
	; Acknowledge: Narayan Desai (desai@mcs.anl.gov), Susan Coghlan (smc@alcf.anl.gov), ANL
	; Information: http://www.alcf.anl.gov/resources/storage.php
	; Conversion: Wei Tang (wtang6@iit.edu) Feb 16, 2010
	; MaxJobs: 68936
	; MaxRecords: 68936
	; Preemption: No
	; UnixStartTime: 1231135224
	; TimeZoneString: America/Chicago
	; StartTime: Mon Jan 5 00:00:24 GMT-6 2009
	; EndTime: Tue Sep 1 23:42:06 GMT-6 2009
	; MaxNodes: 40960
	; MaxProcs: 163840

Log Fields	Job #	Submit Time	Wait Time	Running Time	# Alloc Processors			Req Procs	Req running time			User ID			Queue Num			
	1	0	6680	7560	2048	-1	-1	2048	10800	-1	-1	1	-1	-1	1	-1	-1	-1
	2	7	14297	7568	2048	-1	-1	2048	10800	-1	-1	1	-1	-1	1	-1	-1	-1
	3	1590	17322	7561	2048	-1	-1	2048	10800	-1	-1	1	-1	-1	1	-1	-1	-1
	4	2205	61	14972	8192	-1	-1	8192	21600	-1	-1	2	-1	-1	2	-1	-1	-1
	5	2566	17357	7571	2048	-1	-1	2048	10800	-1	-1	1	-1	-1	1	-1	-1	-1
	6	2751	36	3653	256	-1	-1	64	3600	-1	-1	3	-1	-1	3	-1	-1	-1
	7	5485	12162	6147	65536	-1	-1	16384	10800	-1	-1	4	-1	-1	4	-1	-1	-1
	8	6553	39	2733	256	-1	-1	64	3600	-1	-1	3	-1	-1	3	-1	-1	-1
	9	10355	48	565	256	-1	-1	64	3600	-1	-1	3	-1	-1	3	-1	-1	-1
	10	14158	50	911	256	-1	-1	64	3600	-1	-1	3	-1	-1	3	-1	-1	-1
	11	14313	6693	7556	2048	-1	-1	2048	10800	-1	-1	1	-1	-1	1	-1	-1	-1
	12	21941	54508	7577	2048	-1	-1	2048	10800	-1	-1	1	-1	-1	1	-1	-1	-1
	13	26546	51863	7590	2048	-1	-1	2048	10800	-1	-1	1	-1	-1	1	-1	-1	-1
	14	27521	50898	7580	2048	-1	-1	2048	10800	-1	-1	1	-1	-1	1	-1	-1	-1
	15	28617	49817	7566	2048	-1	-1	2048	10800	-1	-1	1	-1	-1	1	-1	-1	-1

II. DATA SETS

Two data sets in the public domain are explored to determine the feasibility of systems prognostics of large compute systems. Section A describes a Web services data set, and Section B a system log data set.

A. *Web services data set*

The web services dataset **Error! Reference source not found.-Error! Reference source not found.** consists of various Quality of Web Service (QWS) measurements on 2507 real web service implementations. The services were collected using a crawler engine with the QWS measurements conducted in March 2008 using a web service broker framework. The QoS attributes covered in the study for each service included Response time, Availability, Throughput, Successability, Reliability, Compliance, Best Practices, Latency and Documentation.

TABLE III. PARAMETERS AND UNITS OF THE WEB SERVICES DATASET OF AL-MASRI AND MAHMOOUD (2008).

ID	Parameter Name	Description	Units
1	Response Time	Time taken to send a request and receive a response	ms
2	Availability	No. of successful invocations/total invocations	%
3	Throughput	Total number of invocations for a given period of time	Invokes/sec
4	Successability	Number of responses / number of request messages	%
5	Reliability	Ratio of the number of error messages to total messages	%
6	Compliance	The extent to which a WSDL document follows WSDL specification	%
7	Best Practices	The extent to which a web service follows WS-I Basic Profile	%
8	Latency	Time taken for the server to process a given request	ms
9	Documentation	Measure of documentation (i.e. description tags) in WSDL	%

Table III shows the QoS attributes of the web services dataset along with their description and the units of measurement. The original study also consisted of calculating a web services relevancy function using a weighted sum of normalized attribute values. This function was subsequently utilized to generate four service classifications based on the overall quality rating. In our study, these attributes are used in an unsupervised learning approach using k-means clustering to group different services in different cases.

B. *BlueGeneP RAS Log from ANL*

Recent thrust on high performance computing (HPC) environment to provide cloud services has led to numerous studies on failure analysis. In order to maintain and monitor HPCs, typically service logs are generated to indicate the health of the system and its components. Some literature already exists which analyzes the service logs to characterize errors, provide root cause analysis and predict failures [10-16].

The data set used for this study is based on the IBM Blue Gene/P system at Argonne National Labs. It consists of a 273-day RAS and Job log collected in the ANL system. The logs and associated information are also available for download from [17] and [18]. More details about the HPC system and the log data used for the analysis can be found in [10]. Here we summarize the main attributes of the data se.

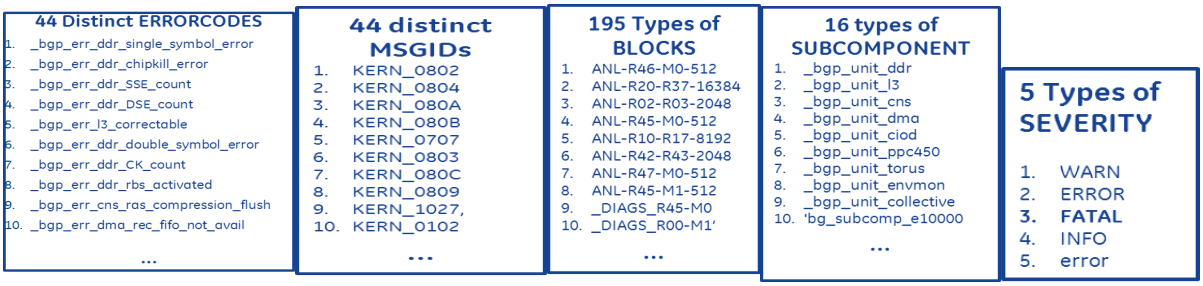
RAS Log: The Reliability, Availability and Serviceability Logs on Blue Gene/P are collected using the Core Monitoring and Control Systems (CMCS) which monitors compute nodes, I/O nodes and various other network components. The collected fields in the RAS logs are given in the Table I. In this work we consider the SEVERITY field and in particular where the severity is denoted as FATAL. A fatal event usually denotes a failure of a critical component which leads to an application crash, a hardware crash or severe loss of a service. Using the log entries preceding a fatal error, we study whether the fatal errors can be predicted.

Job Log: We also consider the Job Log or the records of jobs scheduled on the HPC by the scheduler Cobalt Note: (http://trac.mcs.anl.gov/projects/cobalt/). An example of the Job log is given in the Table II. While the RAS logs provides the notable events occurring in the HPC cluster, the job log provides application level information to further delve into the root cause analysis of a particular problem. The job log may also help differentiate between software and hardware failures by localizing a fault to a particular application since the RAS logs typically provide only system level messages. A joint

TABLE IV. TYPICAL LINES IN THE RAS LOGS

RECID	MSG_ID	COMPONENT	SUBCOMPONENT	ERRCODE	SEVERITY	EVENT_TIME	FLAGS	PROCESSOR	NODE	BLOCK	LOCATION	SERIALNUMBER	ECID	MESSAGE
26123930	KERN_0802	KERNEL	_bgp_unit_ddr	_bgp_err_ddr_single_symbol_error	WARN	2009-01-05-00.02.51.162211	-	0	-	ANL-R46-M0-512	R46-M0-N01-J33	3575YL12M80156ZH	x'02405004902DA518080377D308AE'	ECC-correctable single symbol error: DDR Controller 0, failing SDRAM address 0x00466e2a0, BPC pin ER118, transfer 0, bit 157, BPC module pin L04, compute trace MEMORY0DATA157, DRAM chip U01, DRAM pin D9.
26123943	KERN_0804	KERNEL	_bgp_unit_ddr	_bgp_err_ddr_chipkill_error	WARN	2009-01-05-00.06.44.106651	-	0	-	ANL-R20-R37-16384	R21-M0-N10-J05	44V3572YL12K73050CT	x'02407D34C1045713100B674608A2'	ECC-correctable chipkill error: DDR Controller 1, failing SDRAM address 0x03afb4180, chipkill location 0x008, either X8 compute DRAM chip U15 or U34.
26123976	KERN_0804	KERNEL	_bgp_unit_ddr	_bgp_err_ddr_chipkill_error	WARN	2009-01-05-00.19.36.103137	-	0	-	ANL-R20-R37-16384	R20-M1-N06-J14	4V3575YL12K731304S	x'024028601C69180A0A1047D346AD'	ECC-correctable chipkill error: DDR Controller 0, failing SDRAM address 0x035770da0, chipkill location 0x020, either X8 compute DRAM chip U02 or U07.
26123987	KERN_0804	KERNEL	_bgp_unit_ddr	_bgp_err_ddr_chipkill_error	WARN	2009-01-05-00.22.30.905278	-	0	-	ANL-R20-R37-16384	R20-M0-N00-J15			
												44V3575YL12K731305H	x'024028601B89780A0F085796C8A8'	ECC-correctable chipkill error: DDR Controller 1, failing SDRAM address 0x01573c580, chipkill location 0x010, either X8 compute DRAM chip U32 or U18.
26124008	KERN_080A	KERNEL	_bgp_unit_ddr	_bgp_err_ddr_SSE_count	WARN	2009-01-05-00.35.53.068598	-	0	-	ANL-R02-R03-2048	R03-M1-N15-J04	44V3575YL12M7270WJD	x'02402C600839D80F0C0E576748A4'	DDR controller 0, chipselect 0 single symbol error count 9
26124021	KERN_080A	KERNEL	_bgp_unit_ddr	_bgp_err_ddr_SSE_count	WARN	2009-01-05-00.41.28.753741	-	0	-	ANL-R45-M0-512	R45-M0-N09-J13	44V3575YL12M7346J9N	x'024050048F70F5180F0667848AA5'	DDR controller 0, chipselect 0 single symbol error count 48222

TABLE V. DIFFERENT KEYWORDS WITH INDEXES EXTRACTED FROM PARSED LOG ENTRIES.



analysis of both RAS and Job logs similar to [10] is planned for the future. In this study we look at them individually.

III. ALGORITHMS

A. *Unsupervised Learning of Web Services Logs*

The original study of Al-Masri et. al [19, 20] treated the web services dataset in a supervised machine learning setting by the virtue of ranking (and hence, instance labels) generated using the relevancy function. In contrast, we treat the data independent of labels. The main reason for not employing the relevancy function is because it was not availability with the data set. The relevancy function, involved a weighted sum of normalized attribute values. The weights in this case correspond to the relative importance of the QoS attributes in the final QoS scoring as determined by the user. While an importance ranking of attributes can be obtained based on the priority of the users, we cannot compute and confirm their relative weights without additional information. As a result, we aim to discover if the data reveals a "natural groupings" allowing us to categorize the web services for further analysis. In practice such an approach may be more suited to a wider array of service classifications since it is completely data driven and unsupervised.

To categorize services into different classes, we study the histogram profiles of selected attributes to investigate the attribute value landscape across the web services. Since the scales of the captured QoS attributes are not directly comparable, we normalize the attribute values in [0,1] interval. We then employ principal component analysis (PCA) [22] to obtain the three major directions of variance for visualization purposes. PCA involves an orthogonal transformation of data by eigenvalue decomposition so that the resulting data dimensions are linearly uncorrelated. These dimensions, known as principal components, are obtained such that they are arranged in decreasing order of their variance. We use the top three dimensions (variables) with the highest variance to visually explore the data.

In order to discover the natural groupings of data instances in the original attribute space, we map the problem to an unsupervised machine learning problem. The motivation was to validate if the data clusters in the original space allowing us to group services by their "similarity" (as defined by the similarity metric used by the clustering method). To this end, we employ a k-means clustering algorithm [22] to perform data clustering in the original 9-dimensional data space. A k-means clustering algorithm is aimed to divide the dataset into k groups. The group that each data instance belongs to is determined as the one with the closest mean. While determining the best k, and hence the optimal number of groups to fit the data, is non-trivial, we chose k=4 so as to be consistent with the groupings obtained in the original study in four service classification using the relevancy function.

Note that we do not currently perform clustering on PCA transformed data. Given that PCA has a symbiotic relationship with k-means clustering method, the next step would be to cluster data on PCA transformed space. We use the PCA and k-means clustering algorithm implementations available as a part of the scikit learn library for Python [23].

TABLE VI. RAS LOGS WITH KEYWORDS INDEXED

RECID		MSG ID	COMPONENT	SUBCOMPONENT	ERR CODE	SEVERITY	EVENT TIME	FLAGS	NODE		BLOCK	LOCATION		LOCATION					
1	[]	1	1	1	1	1	733778	'-'	0	'-'	1	1	'44V3575YL12M80156ZH'	'x"02405004902DA518 080377D308AE"'	'ECC-correctable'	'single'	'symbol'	'error:'	'DDR'
2	[]	2	1	1	2	1	733778	'-'	0	'-'	2	2	'44V3572YL12K73050CT'	'x"02407D34C1045713 100B674608A2"'	'ECC-correctable'	'chipkill'	'error:'	'DDR'	'Controller'
3	[]	2	1	1	2	1	733778	'-'	0	'-'	2	3	'44V3575YL12K731304S'	'x"024028601C69180A 0A1047D346AD"'	'ECC-correctable'	'chipkill'	'error:'	'DDR'	'Controller'
4	[]	2	1	1	2	1	733778	'-'	0	'-'	2	4	'44V3575YL12K731305H'	'x"024028601B89780A 0F085796C8A8"'	'ECC-correctable'	'chipkill'	'error:'	'DDR'	'Controller'
5	[]	3	1	1	3	1	733778	'-'	0	'-'	3	5	'44V3575YL12M7270WJD'	'x"02402C600839D80F 0C0E576748A4"'	'DDR'	'controller'	'0,'	'chipselect'	'0'
6	[]	3	1	1	3	1	733778	'-'	0	'-'	4	6	'44V3575YL12M7346J9N'	'x"024050048F70F518 0F0667848AA5"'	'DDR'	'controller'	'0,'	'chipselect'	'0'
7	[]	3	1	1	3	1	733778	'-'	0	'-'	4	7	'44V3575YL12M7345J1M'	'x"02407D34C0101713 0D086871C8BB"'	'DDR'	'controller'	'1,'	'chipselect'	'0'
8	[]	4	1	1	4	1	733778	'-'	0	'-'	4	6	'44V3575YL12M7346J9N'	'x"024050048F70F518 0F0667848AA5"'	'DDR'	'controller'	'0,'	'chipselect'	'0'
9	[]	1	1	1	1	1	733778	'-'	0	'-'	4	6	'44V3575YL12M7346J9N'	'x"024050048F70F518 0F0667848AA5"'	'ECC-correctable'	'single'	'symbol'	'error:'	'DDR'
10	[]	5	1	2	5	1	733778	'-'	0	'-'	5	8	'44V3572YL12K80031JY'	'x"024004700911380F 0F0B4757C8A1"'	'L3'	'Correctable'	'error'	'L3'	'controller=0x00000001'

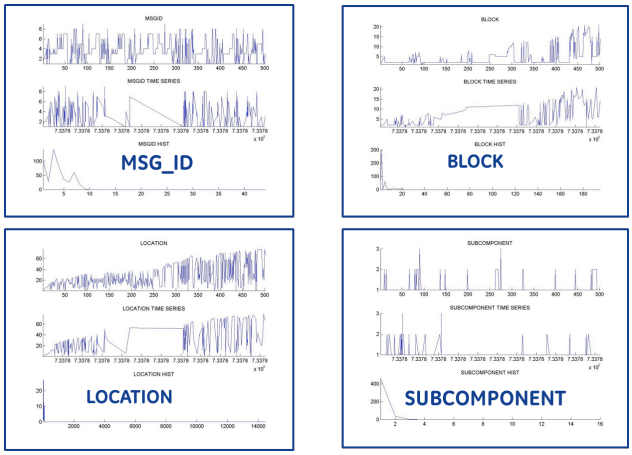


Figure 1: Figure showing the sequence of keywords appearing in a block log entries for different RAS fields along with the histogram of the keyword frequency in each field.

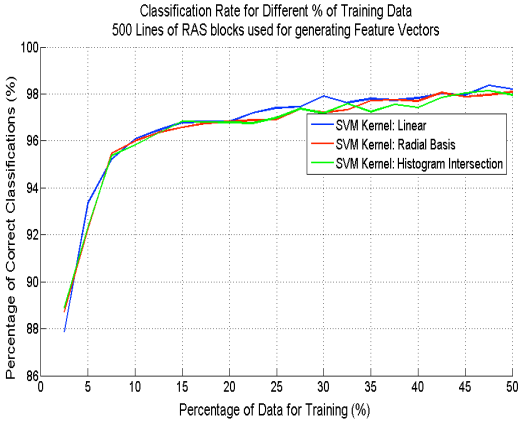


Figure 2: for the figure 345 distinct blocks leading fatal errors and 1000 non-fatal blocks were sued. The graph shows the classification accuracy with different fraction of the dataset used in the training period.

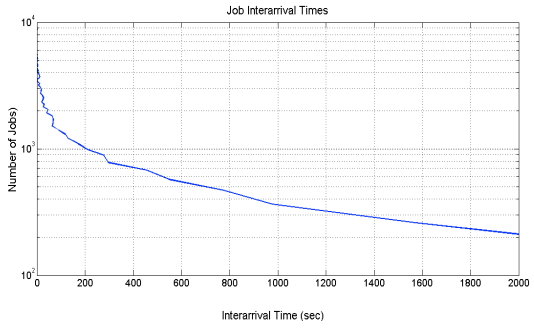


Figure 3: Figure showing the process interarrival times

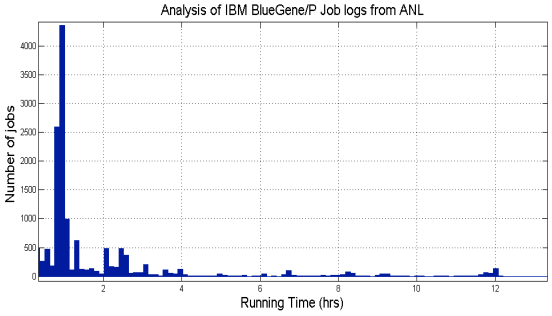


Figure 4: Figure showing histogram of process run time

B. *Predicting FATAL Errors on RAS logs using Support Vector Machines.*

In order to predict fatal errors we consider a list of log entries preceding it, and use Support Vector Machine [16] based classifier to decide whether the log entries suggest an impending fatal error or a normal state (with non-fatal or no errors) of the system. SVM is a standard supervised learning technique which has been extensively used for classification and regression analysis of data. It has proved particularly useful when the relative utility of different attributes (from a classification point of view) is not well understood for a complex system with a high dimensional feature space.

Before we can employ the SVM based classification, we need to do some preprocessing of the RAS logs to extract relevant information from the RAS logs. Table IV shows how RAS log entries appear as semi-structured text. We parse the log one line at a time to extract the values of the different fields such as MSG_ID, SUBCOMPONENT etc. For each unique value of each RAS field a unique code index is generated. Table V gives examples of unique words or values extracted for each field along with the index number for each of them. Finally using the indexes, the RAS log is converted into a codebook with the actual words associated with each field converted to an index entry as shown in Table VI.

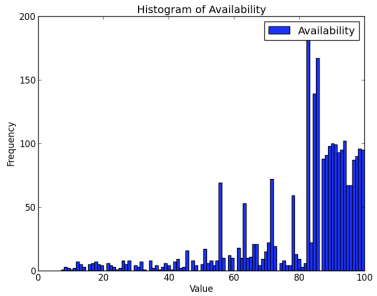


Figure 5: Histogram profile of Availability attribute of web services data

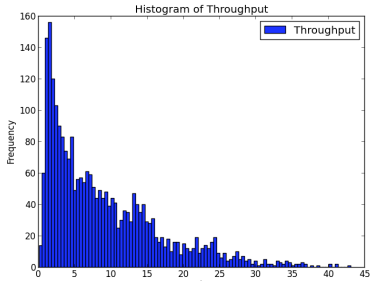


Figure 6: Histogram profile of Throughput attribute of web services data

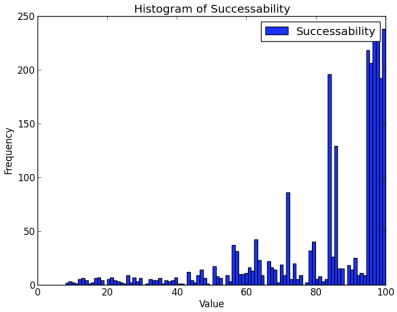


Figure 7: Histogram profile of Successability attribute of web services data

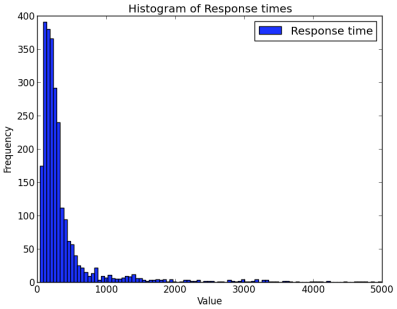


Figure 8: Histogram profile of Response times of web services

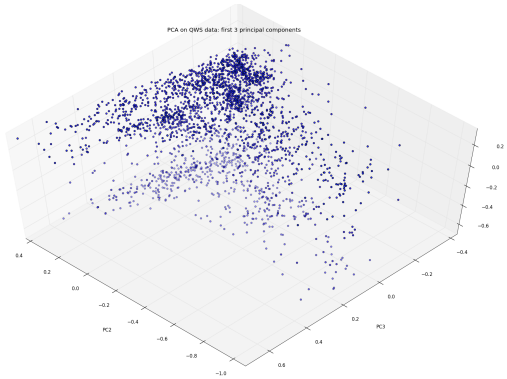


Figure 9: Visualization of PCA transform of web services data (top three principal components)

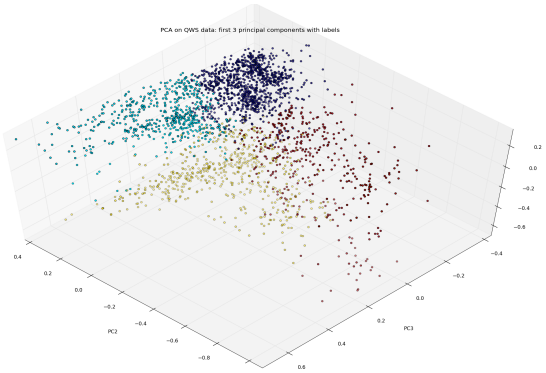


Figure 10: Results of k-means clustering (k=4) of web services data

Next, we look at the predictability of FATAL errors (i.e. lines with SEVERITY entry as FATAL) using the log entries preceding them. We consider a fixed window of log entries preceding each fatal error. The window may consist of a fixed number of lines or a fixed time window. In this work we used a fixed window of 500 lines to describe a block of logs.

Our intuition is that the group of log entries would contain enough indicator messages which can be used to predict that a fatal error is imminent. In particular we assume that certain keywords appearing in the preceding log entries in each field can be used to predict the fatal error. Thus keywords, denoted by index entries in the codebook may be used as features to classify if a preceding block of log entries denote a block leading to a fatal error or a non-fatal error.

Using the keyword entries in the codebook, we create feature vectors for different blocks of log entries. Our feature vectors are simply histogram of keywords appearing in a fixed length block of log entries. The histograms signify the relative frequency of certain keywords appearing in the preceding logs and high frequency of certain keywords (for example a sequence of non-fatal errors) may signify that a severe or fatal error is impending.

The Log entry blocks are divided into two classes as *Fatal* and *Non-Fatal (*with any warning which is not a fatal error). For fatal blocks, we consider all entries with the severity field equal to Fatal and generate feature vectors with log entries preceding the fatal event. For the results, a group of 500 log entries preceding each fatal error were used. For the non-Fatal entries, we randomly consider an entry in the log whose severity is not fatal and which does not have a fatal event within 500 lines of logs before or after it. Thus we get a block with no fatal entries in it and create feature vectors belonging to the non-fatal class.

The two feature sets belonging to fatal and non-fatal errors further are subdivided into training and testing sets. Finally, the training set is used to train a SVM based binary classifier. While at this point we are unable to cross-validate the groupings with some service class ground truth, preliminary results suggests good separation in the classes. A formal

evaluation of the goodness metrics of the clusters is subject of future work.

IV. RESULTS

A. *Web services dataset*

Figures 5-8 present histogram profiles of Availability, Throughput, Successability and Response time attributes for the web services data. Figure 9 shows the top three principal components to illustrate the data distribution along the three most variable directions.

Figure 10 shows the groupings (clusters) obtained using the k-means clustering algorithm on the web services data in the original data space. In order to visualize the groupings, we superimpose the labels (each color on data point represents a cluster) on the PCA visualization of the data.

B. *SVM based classification of RAS logs*

As of date we have processed only a small fraction of the entire recorded log due to certain data and parsing irregularities but the initial results are promising. For the results we were able to extract and analyze 345 distinct fatal errors. For the non-fatal case we randomly generated 1000 distinct blocks. In order to evaluate whether we can distinguish the two different sets using the feature vectors, we divided the data set into training and testing set. Figure 2 above shows the classification accuracy for different SVM kernels used. We see that even with a very small set of training examples; we can predict an imminent fatal event with extremely high accuracy.

We have also started evaluating the job logs and plotted some statistics related to processes in Figure 3 and Figure 4. As described in [10, 11], information such as running time of a job has strong correlation with the probability of failure. For example [1] shows that longer jobs (and not larger jobs) are more susceptible to failures. We believe that addition of such information available from simultaneous from job logs would improve the error characterization and predictability. This is subject of future work.

V. CONCLUSIONS AND NEXT STEPS

The work presents our preliminary investigation on the problem of predicting Quality of Service and failures in cloud based architecture supporting critical services such for healthcare monitoring. We consider a two-pronged approach to achieve this. First, web service attributes are used for classification and clustering of services that provide application level tool and parameters to characterize and predict quality of service. Second we analyze RAS logs from HPC clusters that would support the services, and study if failures leading to loss or degradation of services can be predicted. Our initial results are promising and suggest a methodology and technique suitable for this purpose.

This paper presents an initial foray into the analysis of cloud based critical services and health monitoring. As future work, we would conduct a complete system level analysis involving application level monitors, end-to-end service quality attributes and system level hardware and software monitors to accurately predict service quality in cloud based critical services architecture

REFERENCES

[1] G. Vachtsevanos, F. L. Lewis, M. Roemer, A. Hess and B. Wu, Intelligent Fault Diagnosis and Prognosis for Engineering Systems, John Wiley & Sons, Inc., 2006.

[2] M. Schwabacher (2005), "A Survey of Data-Driven Prognostics", Proceedings of the AIAA Infotech@Aerospace Conference, Arlington, Virginia, September 26-29, 2005.

[3] M. Schwabacher and K. Goebel (2007), "A Survey of Artificial Intelligence for Prognostics", Proceedings of the AAAI Fall Symposium, Arlington, Virginia, 2007.

[4] J. B. Coble and J. W. Hines (2008), "Prognostic Algorithm Categorization with PHM Challenge Application", Proceedings of the 2008 International Conference on Prognostics and Health Management, October 6-9, 2008.

[5] A. B. Chandola and V. Kumar (2009), "Anomaly Detection: A Survey", ACM Computing Surveys, Vol. 43, No. 3, 2009.

[6] Urmanov, A. and Bougaev, A., " Prognostics in Data Centers," http://www.stanford.edu/class/ee392m/Lecture7Urmanov.pdf

[7] Urmanov, "Electronic Prognostics for Computer Servers," Reliability and Maintainability Symposium, 2007. RAMS '07, 22-25 Jan. 2007

[8] Bougaev, Urmanov, " R-functions Based Classification for Abnormal SoftwareProcess Detection," Lecture Notes in Computer Science, Volume 3801/2005, Computational Intelligence and Security, Springer Berlin / Heidelberg

[9] Garvey, Dustin; Evans, Scott; Iyer, Naresh; Varma, Anil; Yan, Weizhong; Bouqata, Bouchra, "Iscale, and Introduction," GRC Technical report 2012GRC608

[10] Z. Zheng, L. Yu, W. Tang, Z. Lan, R. Gupta, N. Desai, S. Coghlan, D. Buettner, Co-analysis of RAS Log and Job Log on Blue Gene/P, International Parallel & Distributed Processing Symposium (IPDPS), 2011

[11] Y. Liang , Y. Zhang , M. Jette , A. Sivasubramaniam , R. Sahoo, BlueGene/L Failure Analysis and Prediction Models, International Conference on Dependable Systems and Networks, 06

[12] Mengliao Wang, Xiaoyu Shi, Ken Wong, , Learning Configuration Files for Automatic Fault Diagnosis, International Conference on Program Comprehension (ICPC), 2011

[13] Wei Zhou, Jianfeng Zhan, Dan Meng: Multidimensional Analysis of System Logs in Large-scale Cluster Systems. CoRR abs/0906.1328 (2009)

[14] Z. Lan, Y. Li, P. Gujrati, Z. Zheng, R. Thakur, and J. White, "A Fault Diagnosis and Prognosis Service for TeraGrid Clusters", Proc. of TeraGrid'07 , 2007.

[15] Resource management on Cloud systems with Machine Learning , Master's Thesis, Zhenyu Fang, Technical University of Catalonia

[16] Cortes, Corinna; and Vapnik, Vladimir N.; "Support-Vector Networks", Machine Learning, 20, 1995.

[17] Parallel workloads archive, http://www.cs.huji.ac.il/labs/parallel/workload

[18] Usenix Computer Failure Data Repository, http://cfdr.usenix.org

[19] E. Al-Masri, and Q. H. Mahmoud, "QoS-based Discovery and Ranking of Web Services", IEEE 16th International Conference on Computer Communications and Networks (ICCCN), 529-534, 2007.

[20] E. Al-Masri, and Q. H. Mahmoud, "Investigating Web Services on the World Wide Web", 17th International Conference on World Wide Web (WWW), 795-804, 2008.

[21] QWS Data URL: http://www.uoguelph.ca/~qmahmoud/qws/index.htm

[22] Bishop, C. M., Pattern Recognition and Machine Learning, Springer, 2006

[23] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011