Catchy Title

John Smith
Dept. of Computer Science
Carnegie Mellon University
jsmith@cs.cmu.edu

Tom Johnson
Dept. of Metaphysics
Confusion State University
tj@bla.csu.edu

October 11, 2011

1 Introduction

Timeseries data are prevalent in large-scale computing centers. Systems often capture sampled metrics of performance, utilization, and even sensor data like temperature. System logs can be also extracted to event signal associated with time stamp. These streams are used for monitoring, placement, optimization, and more: for example, task assignment algorithms use computer utilization to determine where to place jobs.

We propose development of a model of timeseries utilization data that incorporates correlations across time and across multiple streams. Such a model provides the potential to address a number of practical advances for datacenter efficiency:

- Improve alerting and placement algorithms by predicting future usage. Even a short-term view of future usage can be valuable for decision-making. By playing the model forward, we can obtain this data.
- Reduce the storage requirement of streams through compression. By storing only the model parameters and occasional original datapoints, potentially large timeseries data can be effectively summarized.
- Detecting potential anomalies or alert conditions. Dramatic changes in model parameters can be predictors of problems or abnormalities.

The scope of potential applications is considerable, and there is prior art in models for these operations on timeseries data (Sec. 2). We plan to focus our efforts on applying and tuning existing algorithms from the literature on the different type of timeseries data we have: rather than target relatively small sensor datasets as many have considered, we seek to apply these techniques to lengthy, bursty utilization data which are collected from real production clusters. This focus motivates our emphasis on evaluating multiple models, preprocessing, and examining scalability (Sec. 3).

2 Survey

2.1 Models

SPIRIT [5] is an algorithm for discovering patterns in multiple streams of timeseries data. This algorithm effectively tries to find correlations between the different data streams by performing principal component analysis on the the data covariance matrix. These principal components are modified and updated as each data sample is received at each time step while forgetting the samples. An interesting aspect of this algorithm is, anomaly detection which is done by detecting the changes in the number of principal components required to capture most of the energy in the signal. In [6] the authors describe a unifying model for Linear Gaussian Models which includes PCA, Kalman Filters, Gaussian Mixture Models, ICA etc. They describe a general linear state transition model along with a linear observation model corrupted by independent Gaussian noise. They describe general algorithms to do inference and also learn the parameters of the model.

DynaMMo is an algorithm for reconstructing missing timeseries data by learning a linear dynamical model [3]. This general model is learned though belief propagation and EM, and generalizes SVD, linear interpolation, and Kalman filtering. The authors demonstrate that the method captures dynamics these special cases do not based on experimentation with period flow data and motion capture traces. We plan to evaluate one of the special cases (Kalman filtering) and consider scalable approaches, an area where general approaches DynaMMo currently suffer. The paper describes how to use the model for compression by intermittently storing model parameters, an approach we may consider. Feature Extraction

2.2 Feature Extraction

One feature present in systems operations data that has not often been considered when producing sensor models is configuration information that describes what jobs are running on a cluster. Although it is time-based, it consists of events and not a continuous stream. Previous work has considered how to model such aperiodic streams in the context of computational finance; for example, Lavrenko et al [2] link events (news stories) with continuous stock price data by discretizing the continuous stream and using a sliding window to associate candidate events with changes in the stream. However, efforts in this vein do not preserve the continuous nature of performance data for predictions and compression: when using event data, we plan to convert it to a stream instead.

Zhu and Shasha [11] address the problem of detecting bursts in timeseries when their time length are unknown. The core approach is to convert time series data to wavelets, and maintain them for a given sliding window of time in a shifted wavelet tree. This data structure allows constructing a pattern of a flexible size within that window from wavelets of different sizes. In turn, it enables analyzing the timeseries data at different time scales by extracting patterns of different sizes. An online algorithm for maintaining and searching the data structure gives a constant time for updating it, while a batch algorithm trades it off for constant search times. This work focuses on efficiently detecting bursts in one timeseries. On the other hand, we would need to extend their technique and apply it to a number of timeseries. We may need to examine bursts in a group of timeseries, not only those in individual series.

Sakurai et al. [7] introduce an effective way of finding trends in large timeseries data. In particular, they propose a method for selecting an appropriate window size in time that analysts would find useful. The idea is based on first using independent component analysis (ICA), and then selecting the window size that maximizes the entropy of the weight values in the mixing matrix. By using this criterion, the technique effectively extracts repetitive patterns at different time scales from various data sets, such as Ondemand TV access records and web-click traces for a search engine. This work is complementary to the burst detection work by Zhu and Shasha, as the former suggests the best window size that exhibits repetitive patterns while the latter allows examining different window sizes efficiently. Although the paper focuses on web-click records, the general technique would also be applicable to our data center logs.

Another work by Zhu and Shasha [10] presents an efficient approach to finding correlations among a large number of data streams, in an online fashion for a given sliding window size. The key idea is to apply discrete Fourier transform (DFT) to time sequences, and hash them based on their DFT coefficients. This technique allows finding highly correlated time series for a given series, by reducing the number of candidates examined based on their grouping derived from hashing. Also, this method is highly parallelizable, which further improves its efficiency. The problem of finding correlations among a number of time series is one of our possible directions. However, the nature of data in which we are interested is different and we may accordingly need to adjust the approach; we are dealing with data center logs, whereas they focus on stock market data.

2.3 Datacenter Data

A workload prediction study [?] examines time-series measurement of workloads in datacenters with an eye towards identifying patterns to generate synthetic traces. To analyze patterns, they use two primary techniques: a frequency-domain representation of the workload (as computed from an FFT) and autocorrelation of the trace. Since strong periodic patterns may not appear in all workloads, the approach clusters streams by the goodness of fit of its computed model. We plan to run these analyses on our data to start in order to assess whether they are periodic, since most (76 of 139) patterns the authors study have a moderate degree of periodicity produced by ”interactive and/or mixed batch and interactive work.”

PAL is an anomaly localization tool to detect faulty components in distributed applications. The key idea is firstly to monitor timeseries metrics of each individual component of the distributed system and discover the change points (potentially some faulty behavior happen), and then correlated different change points based on time stamp to locate the real faulty component. Their algorithm has four steps: firstly, raw time series data needs to be smoothed. Kalman filter failed to capture the critical change points so they used 5-length moving average filter. The second step is to use CUSUM to find change points, and calculate the separation level of each change point (which measures the different of mean value of data points before the change point and after it). The outlier change points (whose separation level are outliers) is the critical change points.

Another console logs mining paper [9] extracts specific features from unstructured logs of Hadoop system [1] (A distributed system for data intensive computing) and use them to detect anomalies in the system. The tool is mainly used to detect correctness problems. The author combines logs and source to extract three different signals: message types, state variables, and message identifiers. Based on these signals, they create two matrices: one matrix has state variables as the column, and time as the row, the other matrix uses message types as the column, and message identifiers as the row. Finally, PCA is used for anomaly detection. They perform PCA on the two matrices to get a reduced set of dimensions (eigenvectors) ranked by how much variance in the data each reduced dimension explains. Anomalous rows are identified by flagging those rows whose data is largely unexplained by the first K eigenvectors, which describes 95% of the overall observed variance. After finding the anomalous rows, anomalies need to be manually interpreted to dig out the root causes. In [8], they further applied their algorithms into some of Google's production logs. Since the data is large which makes it unrealistic to perform PCA, they used frequent mining techniques to segment the long-lasting sequence into sessions.

3 Proposed Method

3.1 Data Collection

The data sets is mainly collected from one data center: OpenCloud. OpenCloud is a CMU-based academic cluster consisting of 64 compute nodes, each with dual quad-core 2.83GHz Intel Xeon processors, 16GB memory and a 10GigE NIC, and Arista 10 GigE switches. OpenCluster mainly runs distributed services like Hadoop file systems and map reduce systems. We are collecting OS-level metrics and Hadoop system logs per 5 seconds, and have data over several months.

3.2 Models

We will begin by using two existing correlation-based models on our dataset. One is SPIRIT, an incremental PCA approach that adapts the number of hidden variables [5]. The datasets we consider are significantly less periodic and more bursty than the ones explored in the SPIRIT paper, and we may need to adjust the approach accordingly.

Another approach that we consider to learn a model for time series collected in a computer cluster is Kalman Filtering. Kalman Filter models are popularly used for time series modeling. *Dynamic Data*, where temporal ordering of the data samples is important. The dynamic generative model given below, captures the state evolution dynamics of the data.

$$x_{t+1} = Ax_t + w \quad (1)$$

$$y_t = Cx_t + v \quad (2)$$

where A is the state transition matrix, C captures the observation model, w N(0,Q) and v

N(0,R) are the state evolution and observation noise models respectively. The two main subproblems associated with learning these models are *inference* and *parameter learning*. Inference deals with issues of estimating the unknown hidden variables given some observations and a given fixed model parameters. Parameter learning pertains to estimating the model parameters given only the observations. The Expectation-Maximization (EM) algorithm is commonly used for parameter learning, while inference is done via maximum likelihood estimation.

Again as mentioned above, we will treat the multi-aspect data of the computing cluster in two different ways,

- Run kalman filtering on the different data streams from each machine individually.
- Aggregate the data streams from all the machines to get a cluster level model for the data streams.

In addition to learning a model and using it to predict future patterns, we plan to do an analysis on existing data to observe important trends, such as bursts and repetitive patterns. Our main challenge in this problem is to deal with large data sets efficiently and flexibly. Our proposed direction is to build on the burst detection technique by Zhu et al. [11] and learn trends at different time scales and possibly with varied granularity of data aggregation across data center nodes.

3.3 Feature Extraction

We will perform two types of preprocessing on the raw data to produce candidate input that suits the filters we are testing. One is the log transform, in order to smooth high spikes expected in the data. The other is wavelet conversion, which would also be used for burst detection under different time windows.

3.4 Scalability

As a second phase of the project, we plan to choose the model that works best for our data on the basis of how accurately it can provide short-term predictions. As observed above, the primary challenge for many of these algorithms is efficiency and scale. In order to handle multiple streams from large scale data centers, it is necessary to parallelize our analysis algorithm. Though the precise steps we will take to improve algorithm scalability at this point are not yet completely defined, we plan to use Hadoop as the computing framework to implement the parallel version of our algorithm.

3.5 Anomaly Detection

After validating the right model for our timeseries data from the data center, we plan to apply the model and combine it with anomaly detection algorithm. The first stage is to manually inject failures to tune the model to increase the accuracy. Later on, we can apply the algorithms on historical data from OpenCloud data center to test whether the algorithm can find real anomalies in the system.

4 Plan of Work

Member	**Project Portion**	**Papers Read**
Yoshihisa Abe		[10], [11], [7]
Ilari Shafer	IPCA application/extension, scalability	[3], [**?**], [2]
Kai Ren	Data Collection + Preparation, Anomaly Detection Experiment	[9], [8], [4]
Vishnu Boddeti	Kalman Filter Models	[5], [6]

5 Timeline

A rough timeline of the project follows.

November 10 Finish testing existing models (and preparing data)

November 24 Finish working on distributing algorithm (e.g. Hadoop)

December 4 Finish Final Report

December 9 Address anomaly insertion, clean up loose ends, prepare demo

References

[1] Apache Software Foundation. Apache Hadoop, 2011.

[2] V. Lavrenko, M. Schmill, D. Lawrie, P. Ogilvie, D. Jensen, and J. Allan. Mining of concurrent text and time series. In *KDD-2000 Workshop on Text Mining*. Citeseer, 2000.

[3] Lei Li, J. McCann, N.S. Pollard, and Christos Faloutsos. Dynammo: Mining and summarization of coevolving sequences with missing values. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 507–516. ACM, 2009.

[4] Hiep Nguyen, Yongmin Tan, and Xiaohui Gu. PAL: Propagation-aware Anomaly Localization for Cloud Hosted Distributed Applications. In *SLAML*, 2011.

[5] S. Papadimitriou, J. Sun, and Christos Faloutsos. Streaming pattern discovery in multiple time-series. In *Proceedings of the 31st international conference on Very large data bases*, pages 697–708. VLDB Endowment, 2005.

[6] Sam Roweis and Zoubin Ghahramani. A Unifying Review of Linear Gaussian Models. *Neural Computation*, 11(2):305–345, February 1999.

[7] Yasushi Sakurai, L. Li, Y. Matsubara, and Christos Faloutsos. WindMine: Fast and Effective Mining of Web-click Sequences. In *SDM*, number c, 2011.

[8] Wei Xu, L. Huang, Armando Fox, David Patterson, and Michael Jordan. Experience Mining Googles Production Console Logs. In *SLAML*, 2010.

[9] Wei Xu, Ling Huang, Armando Fox, David Patterson, and M.I. Jordan. Detecting large-scale system problems by mining console logs. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pages 117–132, New York, New York, USA, 2009. ACM.

[10] Y. Zhu and D. Shasha. StatStream: Statistical monitoring of thousands of data streams in real time. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 358–369. VLDB Endowment, 2002.

[11] Yunyue Zhu and Dennis Shasha. Efficient elastic burst detection in data streams. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 336–345, New York, New York, USA, 2003. ACM.