

# MSET Performance Optimization for Detection of Software Aging

Kalyan Vaidyanathan and Kenny Gross  
RAS Computer Analysis Laboratory  
Sun Microsystems  
San Diego, CA 92121, USA  
{kalyan.vaidyanathan,kenny.gross}@sun.com

## 1. Introduction

Software aging [2] is a phenomenon observed in a software application executing continuously for a long period of time, where exhaustion of operating system resources (memory leaks), data corruption and numerical error accumulation eventually lead to performance degradation, hang/crash failures or both. To counteract this problem, Huang et al. [2] proposed the technique of software rejuvenation, which involves occasionally stopping the software application, removing the accrued error conditions and then restarting the application in a clean environment. For time-based rejuvenation policies, which are simpler to implement, state restoration is performed at regular deterministic intervals. For prediction-based rejuvenation, dynamic resource metrics are continuously monitored and rejuvenation is attempted only when the onset of aging is deemed highly probable. The second approach allows the most efficient overall system operation. This approach, however, requires that some performance metrics be identified that can be monitored to detect the onset of aging [3].

Our objective in this paper is to study the performance tradeoffs in using the Multivariate State Estimation Technique (MSET) for proactive annunciation of software aging in large, Unix-based multiprocessor servers that are used in mission-critical and business-critical e-commerce applications.

## 2. MSET

MSET [1] is a nonlinear, nonparametric modeling method that was originally developed by Argonne National Laboratory (ANL) for high-sensitivity proactive fault monitoring applications in commercial nuclear power applications. The implementation of MSET that was used in this study was the SureSense tool developed by Expert Microsystems. In the first step, a training procedure is used to characterize the monitored equipment using historical operating data which contain all modes and ranges of operation

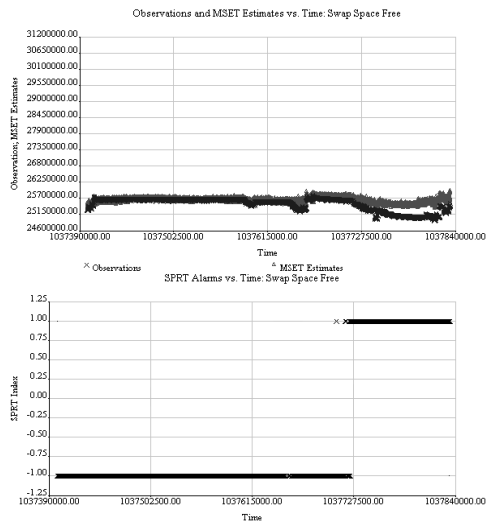
and does not contain any operating anomalies. This procedure builds a model out of a subset of observations (called training vectors) that are determined to best characterize the system's expected operational state for the selected signals. In the second step, called the monitoring step, newly acquired observations are used together with the previously trained MSET model to estimate the expected values of the signals.

The difference between a signal's predicted value and its directly sensed value, called a residual, is then computed. The software's fault detection procedure employs a Sequential Probability Ratio Test (SPRT) technique to determine whether the residual error value is uncharacteristic of the learned process model and thereby indicative of a sensor or equipment fault. For sudden, gross failures of a sensor or a subsystem, this procedure announces the disturbance as fast as a conventional threshold limit check. However, for slow degradation as in the case of software aging, this test can detect the disturbance long before it would be apparent with conventional threshold limits.

The main factors that determine the running time of a training model are the number of training vectors (user-settable in SureSense) and the number of signals in the model. The model training time is linearly proportional to the number of training vectors and to the square of the number of signals in the model. Generally, estimates produced by using a lesser number of training vectors tend to be less accurate and less reliable than those created by using more training vectors. Under some circumstances, using too many training vectors can capture signal noise and lead to undesirable effects (the "overtraining" syndrome).

### 2.1 Application of MSET to software aging

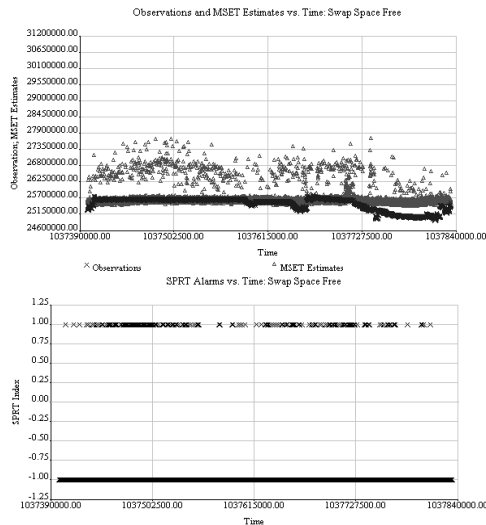
Figures 1 and 2 show examples of MSET applied to detect the onset of software aging. A data set consisting of approximately 35000 observations of 29 software variables (signals) collected from a Unix-based multiprocessor system was used for the models. The upper plot of Figure 1



**Figure 1. Model with 1000 training vectors**

shows the MSET estimates and observations of signal 'free swap space', for a model built using 1000 training vectors. A subtle memory leak was simulated at approximately the mid-point of the observations using the built-in fault injection capability of the SureSense tool. The corresponding lower plot shows the SPRT alarms generated by the tool. Sustained positive alarms can be seen soon after the simulated memory leak starts.

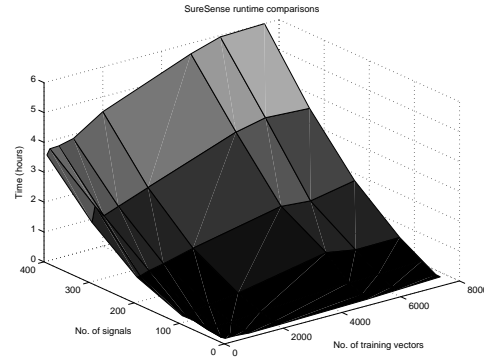
Figure 2 shows MSET estimates and observations from the same data set from a model created using only 100 training vectors. As we can see from the upper plot, the estimates are not as tight as in the previous model and tend to be noisy. Also, the lower plots show sporadic and intermittent false alarms.



**Figure 2. Model with 100 training vectors**

### 3 Multiparametric Tradeoff Study in MSET

Next, the model training times for MSET were studied with various number of training vectors and signals. Figure 3 shows the results from this multiparametric analysis. The training times rise sharply with increase in the number of signals but rise more gently with increase in number of training vectors. This verifies the theoretical estimates of training time dependence on the number of signals and training vectors. Results from this sensitivity analysis are being used as a guideline for deciding tradeoffs issues in setting up optimized MSET models on new server platforms.



**Figure 3. Multiparametric plot**

### 4 Summary and Conclusions

In this paper, we demonstrated the applicability of using MSET to detect the widespread software aging problem and studied the performance versus sensitivity tradeoff issues associated with it. More training vectors results in a better model but takes a longer time to train, while fewer training vectors perform the training procedure quickly but result in less accurate estimates. Future experiments could be conducted along these lines by systematically varying the number of training vectors and the number of model signals, using the approach established in this investigation.

### References

- [1] K. C. Gross et al. Application of a Model-Based Fault Detection System to Nuclear Plant Signals. In *Proc. of ISAP-97*, July 2001.
- [2] Y. Huang, C. Kintala, N. Kolettis, and N. D. Fulton. Software Rejuvenation: Analysis, Module and Applications. In *Proc. of FTCS-25*, June 1995.
- [3] K. S. Trivedi, K. Vaidyanathan and K. Goševa-Popstojanova. Modeling and Analysis of Software Aging and Rejuvenation. In *Proc. of ANSS-2000*, April 2000.