

ABSTRACT

The study of software failures has now become more important since it has been recognized that computer system outages are more due to software faults than due to hardware faults. Recently, the phenomenon of "software aging", one in which the state of the software system degrades with time, has been reported in widely used software and also in high-availability and safety-critical systems. The primary causes of this degradation are the exhaustion of operating system resources, data corruption and numerical error accumulation. This may eventually lead to performance degradation of the software or crash/hang failure or both. To counteract this phenomenon, a proactive approach to fault management, called "software rejuvenation" has been proposed. This essentially involves gracefully terminating an application or a system and restarting it in a clean internal state. This process removes the accumulated errors and frees up operating system resources. The preventive action can be done at optimal times (for example, when the load on the system is low) so that the overhead due to planned system downtime is minimal. This method therefore avoids unplanned and potentially expensive system outages due to software aging.

In this talk, we start with a classification of software faults. We then discuss methods of evaluating the effectiveness of proactive fault management in operational software systems and determining optimal times to perform rejuvenation. This is done by developing stochastic models which tradeoff the cost of unexpected failures due to software aging with the overhead of proactive fault management. Next we show measurements from a real system that show aging. We then discuss how to predict the time to resource exhaustion using the measured data. Finally, we describe the use of rejuvenation in cluster systems and its implementation in a major commercial system.