|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **序号** | **年份** | **作 者** | **文 章 题 目** | **针对的软件和衰退影响** | **衰退检测/评估方法** | **预测衰退(剩余可用寿命)方法** | **抗衰策略选择、实施技术** | **关 键 词** | **不 足** |
| **基于时间模型(衰退分析：基于建模的方法，抗衰策略：定时抗衰，抗衰方法：)** | | | | | | | | | |
|  | 1994 | D. L. Parnas | Software Aging | 软件产品过时。内存泄露、线程未终止、存储空间碎片、数据溢出；操作(参数配置)不当 | | 预防措施：执行软件工程流程 |  |  |  |
|  | 1995 | Yennun Huang, C. Kintala | [Software Rejuvenation：Analysis Module and Applications](../00-参考论文/(1995)%20Software%20Rejuvenation：Analysis%20Module%20and%20Applications（隐蔽性软件故障）.pdf) | 电话计费系统： | 无此步骤 | 连续马尔科夫链建模 | 考虑时间的、系统级重启 | uncertain,elusive bug,  transient缺陷 | 未考虑负载的影响；需要衰退先验知识 |
|  | 1995 | Garg. S | Analysis of software rejuvenation using Markov regenerative stochastic Petri net | C/S的server;  失效率增加 | 无此步骤 | 马尔科夫再生随机petri网 | 时间&负载 |  |  |
|  | 1996 | A. Pfening, S. Garg, A. Puliafito, K.S. Trivedi | Optimal Software Rejuvenation for Tolerating Soft Failures | 电话通讯系统的server;  服务率下降 (事务丢失) | 无此步骤 | 马尔科夫决策过程-基于阈值. look -ahead-n函数，最优最小成本函数 |  |  | 现有的大部分方法都缺乏固定的/一致的控制基础[] |
|  | 1997 | S. Garg, A. Puliafito, M. Telek, K. S. Trivedi | On the analysis of software rejuvenation policies | 基于事务的系统;  失效率增加和服务率下降 | 无此步骤 |  | 时间&负载. Measure: availability, loss Prob. | 软件的分析模型；bug瞬时性 | 假设条件较多；可用性如何评价？ |
|  | 1997 | Avritzer A., E.J. Weyuker | Monitoring Smoothly Degrading Systems for Increased Dependability |  |  |  |  |  |  |
|  | 1998 | Garg. S | analysis of preventive maintenance in transactions based software systems | 基于事务的软件系统的server(不含DBM);  失效率上升,服务率下降(丢失事务) | 无此步骤 | 马尔科夫再生模型：建模软件行为={t时刻软件的状态,t时刻队列容纳力,t} | 时间&负载; Measure: 可用性, loss Prob. | Transient; does not recur upon re-execution;  1+负载 | 假设失效发生时，到来的事务会丢失；影响模型精度的因子:考虑的衰退影响 |
|  | 2000 | T Dohi, K．S．Trivedi | Statistical non-parametric algorithms to estimate the optimal software rejuvenation schedule | 基于事务的软件系统 | 基于时间的衰退分析  半马尔科夫模型建模软件状态迁移过程 | 基于趋势分析的性能预测，采用**随机非参数化算法**评估最优抗衰策略 |  | statistical non-parametric algorithm | 依赖于大量失效时间数据[8]，并假设失效率已知[16] |
|  | 总结 | 基于时间模型(系统建模)（定时抗衰） | | 性能=丢失的事务数/概率；失效率增加=可用性降低 |  | 建模软件数学模型，求解各状态驻留时间和转移概率；求最佳抗衰周期 | **优点:简便,易实施.**  **不足:固定周期,不顾实际,造成不必要的恢复成本；** | | **3.****依赖于失效先验知识，而实际上软件是在不断变化的（自身升级、更新，运行环境变化[]）** |
| **基于度量/预测的方法（视情抗衰）** | | | | | | | | | |
| **序号** | **年份** | **作 者** | **文 章 题 目** | **针对的软件和衰退影响** | **衰退检测/评估方法** | **预测衰退(剩余可用寿命)方法** | **抗衰策略选择、实施技术** | **关 键 词** | **不 足** |
|  | **1998** | **Garg. S** | **A methodology for detection and estimation of software aging** | 1. **Unix工作站-操作系统** 2. **操作系统资源耗尽** | **离线的分析指标时序数据中的趋势，检测衰退的存在，评估衰退对系统资源的影响：**  **时序分析-强局部加权回归(加权最小二乘法估计)-Seasonal Kendall test-斜率评估技术；分析参数的线性、周期性依赖** | **依赖单指标****全局趋势和预计耗尽时间。本文只正式衰退的存在和衰退对系统资源的影响** |  | **不确定性大** | **方法不具备普适性。利用视觉判断时序数据是否有周期性、全局趋势等特殊特性，不同情况下，需要用不同的分析模型去掉其他特性；假设资源耗损情况有全局趋势.**  **分开考虑每个资源的趋势；而且是线性的，无法分析非线性的、复杂的衰退情况。忽略了负载对系统资源耗费的影响;** |
|  | **1999** | **K Vaidyanathan, K.S. Trivedi** | **A measurement-based model for estimation of resource exhaustion in operational software systems** | **Unix操作系统**  **系统资源参数+系统负载(CPU活动和文件系统I/O)** | **聚类系统负荷,得到负荷状态空间;用半马尔科夫反馈模型，建立各状态下各系统参数的反馈函数;求解得到各参数的趋势、耗竭率和耗竭时间** | **基于趋势分析的性能预测:线性回归通过计算最小二乘估计来评估true斜率** |  | **考虑了负载** | **假设有线性趋势** |
|  | **2001** | **V. Castelli, K.S. Trivedi** | **Proactive management of software aging** | **IBM x系列服务器集群系统**  **操作系统资源耗竭**  **(偏具体实现)** | **简单线性回归和扩展的线性回归方法（h断点线性回归，对数线性回归，对数k断点分段线性回归），使用Mallow’s Cp index 和启发式标准Wp选择最优模型** | **线性回归预测20步长以后的(采集间隔：5分钟)。（采集350个小时，前208个小时的数据用来建模；剩余数据用于验证模型。）** | **以基于时间的为用户默认策略、基于预测的（基于阈值）；** | **软件抗衰天生适合集群环境；实验架构可参考** | **线性回归无法处理非线性、更复杂的时序数据；孤立分析参数；有些复杂衰退无法用OS资源表达** |
|  | 2001 | K.Vaidyanathan , K. S. Trivedi | Analysis and Implementation of Software Rejuvenation in Cluster Systems |  |  |  |  |  |  |
|  | 2002 | Vaidyanathan, K.  博士论文 | Proactive management of software systems: Analysis and implementation | 集群系统；失效 |  | 预计耗尽时间。 | 随机回报网.  马尔科夫再生过程理论 |  |  |
|  | **2002** | [**Cassidy, K.J.**](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=Authors:.QT.Cassidy,%20K.J..QT.&newsearch=partialPref)**;** [**Gross, K.C.**](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=Authors:.QT.Gross,%20K.C..QT.&newsearch=partialPref)**;** [**Malekpour, A.**](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=Authors:.QT.Malekpour,%20A..QT.&newsearch=partialPref)**;** | **Advanced pattern recognition for detection of complex software aging phenomena in online transaction processing servers** | 1. **OLTP servers** 2. **共享内存池latch contention(抢占锁)** 3. **共享内存池总剩余内存、锁丢失、DB统计数据** | **MSET(多元状态估计/模式识别技术)从长期监测的正常行为数据中构建经验模型；从观测指标偏离正常运行情况，检测衰退。观测到有13个变量偏离了正常值** |  | **无** | **统计模式识别；找不到能度量的操作系统资源参数** | **借助商业工具；选取反应共享内存池锁抢占的参数；不足：没有考虑参数间的关联性，方法不具一般性.实验结果显示预测误报率有点高，需要额外方法确认、纠错** |
|  | **2002** | **Li L., K.S. Trivedi** | **An Approach for Estimation of Software Aging in a Web Server** | **Web server系统资源耗竭或性能下降(剩余物理内存、已使用交换空间、5分钟平均CPU负载、磁盘请求数、页面换出数、新进程数、响应时间)** | **在[]基础之上，采用非参数化方法计算斜率(类似于数值分析)** | **基于趋势分析的性能预测.时序分析-ARMA模型** | **无** | **非线性时序分析；参数CCF(交叉相关函数)** | **假设软件衰退有一个总的趋势；目测周期性分析；基于先验知识选择衰退相关的指标 [39]；同一软件，不同负载条件下(不同数据集)，反应衰退的参数不同** |
|  | **2003** | K．Vaidyanathan | MSET performance optimization for detection of software aging | **电商应用的大型基于Unix的多处理器服务器的系统资源耗竭** |  |  |  |  |  |
|  | 2003 | Y. Bao, X. Sun, K.S. Trivedi | Adaptive Software Rejuvenation: Degradation Model and Rejuvenation Scheme | 系统资源泄露：内存泄露；  自适应衰退模型：可以用基于模型的、也可用基于度量的方法 | 自适应评估模型： | 自适应评估模型： | 自适应抗衰安排策略 | 自适应衰退模型 | 现有的大部分方法都缺乏固定的/一致的控制基础[] |
|  | 2003 |  | Proactive detection of software aging mechanisms in performance critical computers |  | 软件衰退机制的预防式检测 |  |  |  |  |
|  | 2003 | R. Sahoo et al. | Critical event prediction for proactive management in large-scale computer clusters | 大规模聚类 |  | 基于预测；关键事件和系统参数：不同的时序分析方法和基于规则的分类算法 |  |  |  |
|  | **2003**  **[M]** | K.Y. Cai et al. | **An overview of software cybernetics（控制学）** | **网络环境下的软件**  **服务的性能和质量**  **网络流量的变化** | **软件抗衰是一个软件控制学问题？如何适应性地执行抗衰？** |  |  | **adaptively** | **综述性论文，提出软件抗衰是潜在的研究方向，未提出确切的方法** |
|  | 2004 | Wei Xie, K.S. Trivadi | Software Rejuvenation Policies for Cluster Systems under Varying Workload |  |  | 基于建模技术的性能预测： |  |  |  |
|  | 2004 | M. Malek et al. | Self Rejuvenation：An Effective Way to High Availability |  |  | 基于建模技术的性能预测： |  |  |  |
|  | 2005 | Bao, Y. Trivedi, K. S. | A workload-based analysis of software aging and rejuvenation | 资源泄漏 |  | 非齐次马尔柯夫链：构建衰退模型 | 时间和工作负荷的；半马尔科夫过程 | elusive |  |
|  | 2005 | M. Grottke and K.S. Trivedi | Software Faults, Software Aging and Software Rejuvenation | 若干种bug | 解释了（即使代码中没有额外的故障）衰退是如何发生的，以及若干种bug及其对衰退的影响 |  |  |  |  |
|  | 2005 | K. Vaidyanathan et al. | A comprehensive model for software rejuvenation |  |  |  |  |  |  |
|  | **2006** | **M.Grottke,K. S. Trivedi** | **Analysis of Software Aging in a Web Server** | **Web server: apache.**  **内存泄露和性能(responseTime)** | **考虑了单个资源趋势的周期性** | **数个不同的非参数化统计方法，计算历史数据中的衰退趋势；然后用时序分析方法预测未来任何资源的值，并计算它是否会耗尽。在预测资源消耗时，考虑了数据的周期特性（关注点：如何建模周期特性、如何确定模型order）** |  |  | **用已知的异常数据拟合[],不足以发现未知异常、衰退** |
|  | 2006 | R. Matias Jr. , Paulo J. F. Filho | An Experimental Study on Software Aging and Rejuvenation in Web Servers |  | 衰退对web server的影响 |  |  |  |  |
|  | 2006 | Seong (Steve) R. Yu | A software replication model for rejuvenation transparency to clients in a single computer environment | 单服务器环境的 |  |  | 热备抗衰技术=抗衰+完整冗余(hot-passive软件复制） | 单服务器环境下，抗衰实施不用刻意避开高峰期 |  |
|  | 2006 | 徐建 | 软件抗衰若干关键技术研究(PhD thesis) |  |  |  |  |  |  |
|  | 2006 | L. Silva et al. | [Software Aging and Rejuvenation in a SOAP-based Server](../../00-参考论文/软件衰退%20抗衰/(2006)%20Software%20Aging%20and%20Rejuvenation%20in%20a%20SOAP-based%20Server.pdf) |  |  |  |  |  |  |
|  | 2007 | 徐建 | 软件抗衰研究综述 |  |  | 预测软件性能：认为时序分析方法从本质上讲都是线性模型，且统计学方法大都需要一些难以证明的先验知识。故一些较为复杂的时间序列数据(如软件的性能预测数据)，这类模型的预测结果不能满足应用的需求。 | 清空缓存、内存垃圾收集，重新初始化内核表、清理文件系统、收集碎片等 | Heisenbug(瞬时性、隐蔽性bug) |  |
|  | 2007 | Luis Silva, A.Andrzejak | Deterministic Models of Software Aging and Optimal Rejuvenation Schedules | SOAP server- Apache.  请求率能力=function(服务的请求数).   1. 动态、自动为指定的致衰indicators（y=f(x) ,只考虑一个指标x）找指标，并用统计方法自动验证其正确性。比较通用、参数少 |  | 自适应抗衰？ |  | 为指定的致衰因素找最适合的一个指标。评价上述方法的方法可能值得借鉴 | 本文方法成立的假设：大多数情况，衰退过程能够被一个确定的衰退模型足够精确的捕获；且假设性能下降的主要indicators可以被逼近为一些“work”相关的度量标准（metric），如自前一次抗衰起服务的请求数。如果内存泄露或其他资源未释放是衰退的主要原因的话。而且只考虑了单个指标，没有考虑多指标的关联(复杂的衰退情况是有关联的)；且未考虑时间和软件负荷 |
|  | 2007 | Moura Silva et al. | Using Virtualization to Improve Software Rejuvenation | 应用服务器(Tomcat /Axis) -内存泄露 | 基于阈值的方法决定何时需要执行抗衰技术 | | 只要有一个QoS指标违反阈值，就自动触发抗衰活动。基于虚拟化的 | Threshold，  Virtualization | 只采用简单的阈值技术，只要有一个QoS指标违反阈值，就触发抗衰。丢弃了很多对后续制定规则有帮助的参数，根据经验选择的阈值也不可能适用于所有场景。一些攻击和作业可能保持CPU使用足够高，来逃避虚拟机的抗衰.不追踪前一些times[37] |
|  | 2008 | A.Andrzejak, L. Silva | Using machine learning for non-intrusive modeling and prediction of software aging | SOA应用-应用服务器;  处理的最大请求数/秒(性能下降) | 机器学习 | 机器学习方法(决策树、支持向量机、朴素贝叶斯)建模和预测确定性软件异常 |  | non-intrusive:外置，非内置； transient; adaptive, robust | 作者未验证其方法在动态配置或引起软件异常的多资源耗竭上的效果[] |
|  | 2008 | Cherkasova et al. | Anomaly? Application Change? or Workload Change? Towards Automated Detection of Application Performance Anomaly and Change |  |  |  |  |  | 假设软件可用静态模型建模，模型只依赖工作负荷且不会衰退。 |
|  | 2010 | Javier Alonso et al. | Adaptive on-line software aging prediction based on Machine Learning |  |  |  |  |  |  |
|  | 2010 | D.Simeonov, D. Avresky | **Proactive software rejuvenation based on machine learning techniques**  **（对我立论有用）** | Web server；引起服务器崩溃的异常：  内部异常-内存泄露(内存和cpu负载)，外部异常-拒绝服务攻击 | 考虑一段时期的参数的动态特性，以发现复杂多变的衰退情况；用Lasso Regularization特征提取，降低数据的维数；使机器学习算法能更好的拟合训练数据，同时避免过分拟合 | 机器学习-支持向量机、标准最小二乘法算法，利用异常数据（打标了预测的失效时间）训练规则（但本文未实现） | 虚拟化技术实现软件复制和抗衰；重启虚拟机（或重启应用对应的进程，效果评价：系统参数要和重启虚拟机的参数值一致） | linear regression, Non-linear、complicated behavior, feature selection, a wide range of anormalies | 1）Lasso Regularization是线性回归，对于非线性的、更复杂的行为，需要用其他更高级的机器学习算法(标准最小二乘法/决策树等)，在Lasso Regularization之上，进行学习，以降低分类器的FP、FN。2）学习多种异常的分类器。机器学习方法依赖于异常数据，异常数据不完备，就不能识别其他的、未知的异常。 |
|  | 2011 | Kourai, K. & Chiba, S. | Fast Software Rejuvenation of Virtual Machine Monitors | 虚拟机系统中-VM monitor |  |  | warm-VM reboot |  |  |
|  | 2011 | J. Alonso, Belanche, L. & Avresky, D. R. | Predicting Software Anomalies Using Machine Learning Techniques | 1. 多层次在线书店(Apache Tomcat& mysql); 2. 软件失效：tomcat内存、 | **Lasso 正规化技术（稀疏回归方法）,减少用于构建预测模型的监测指标的数目(最好情况:60%)**  把预测转成检测状态异常;  机器学习算法，将系统状态聚成3类：绿（ok）、黄（警告，红色警告出现前5分钟）、红（危险，软件崩溃前5分钟） | 机器学习算法( R statistical language, Rpart（Decision Tree）,Naive Bayes,Support Vector machines Classifiers,K-nearest neighbors(knn),**Random Forest(随机森林)**, LDA（线性决策分析）/QDA（二次方决策分析）),选择依据：拟合测试数据最好的 |  | 预测资源耗尽引起的异常并非易事；**动态和不确定性软件异常；资源消耗曲线非线性；趋势动态变化；** | 预测失效时间，而非剩余可用寿命，没考虑可用性；  预测剩余寿命太难，转为检测告警(表示即将发生的失效)；具体ML预测算法的选择依赖于训练、测试数据的选择，量又不大。特征提取的目的:为了降低模型的构建成本和复杂度，以及采集任务的难度，而不是考虑参数间的关联对分析结果的影响 |
|  | 2011 | Jean Araujo et al. | Software aging issues on the eucalyptus cloud computing infrastructure | Eucalyptus云计算平台- 节点控制器;  内存泄露 |  |  | 现有解决方案都是针对特定的每个监测环境量身定制的 |  |  |
|  | 2011 | Jean Araujo et al. | Experimental Evaluation of Software Aging Effects on the Eucalyptus Cloud Computing Infrastructure | Eucalyptus云计算平台- 节点控制器;  内存泄露和内存碎片 | 线性回归 | 基于阈值的方法，不预测 | Apache进程重启 |  | 会出现在两个采样时间点之间，内存达到上限、服务崩溃，产生额外的宕机的情况 |
|  | 2011 | Jean Araujo et al. | Software Rejuvenation in Eucalyptus Cloud Computing Infrastructure: A Method Based on Time Series Forecasting and Multiple Thresholds | Eucalyptus云计算平台- 节点控制器;  内存泄露（虚拟内存） |  | 基于阈值的方法 + 时序预测 | Apache进程重启 |  | 只针对内存泄露；未考虑负载对衰退的影响；无法应对资源使用突变的情况 |
|  | 2011  综述 | Cotroneo et al. | Software Aging and Rejuvenation:Where We Are and Where We Are Going | 1. 安全关键的系统(7%)、非安全关键的系统(业务、任务关键:web server, DBMS) 2. **内存泄漏**(物理内存、交换空间)，**性能异常**(吞吐量、响应延迟)；or不具体某一种资源的耗竭。 3. **未来研究：**  * 分析更复杂的衰退表现/情况，其他类型的致衰bug：**算术错误(舍入误差等)**、存储相关的bug、与管理system-depended数据结构相关的bug(如DBMS的共享内存池、JVM的虚拟内存等)；   综合分析真实的致衰bugs. | | 大多关注预测**失效时间**。分为:   1. 基于模型的分析方法：用马尔科夫决策过程和随机Petri网等方法,建立系统的数学模型,解最优抗衰实施时间 2. 基于度量的分析方法：用时序分析和机器学习识别资源耗竭和性能下降的趋势 3. **未来研究**  * 开发混合方法，实现在线监测和抗衰评估框架 | 1. **决策实施抗衰的最优安排：**  * 基于分析模型的方法(如基于时间的抗衰) * 基于度量的方法（基于阈值/预测的抗衰）  1. **目前实施的抗衰技术：**  * 应用无关的（各级重启：组件、应用、VM、节点重启,集群） * 应用相关的：垃圾回收(如java的GC)、内核表刷新,需支持   关注更复杂的衰退表现； | | 1. **未来研究：**  * 更有效的抗衰技术，降低抗衰成本(在线、微重启) * 解释state of the application * 实现额外的机制，通过有选择的部分执行抗衰来最小化宕机时间   OS和中间件、编程语言中融入抗衰策略 |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  | 总结 |  | 衰退的影响是逐步的性能下降和失效，不同的研究针对的影响不同，目前主要针对**内存泄漏**(物理内存、交换空间)和**性能异常**(服务率：吞吐量、响应延迟)。   1. **未来研究[]：**  * 更多地在真实系统中实践 * 分析更复杂的衰退表现/情况，其他类型的致衰bug：**算术错误(****舍入误差等)**、存储相关的bug、与管理system-depended数据结构相关的bug(如DBMS的共享内存池、JVM的虚拟内存等)； * 综合分析真实的致衰bugs.   这就需要有一个通用的刻画软件可用性/状态的方法，能够智能的选择关键参数。 | | 目前预测抗衰时机主要是针对失效(剩余寿命)，可以说抗衰是失效敏感的；而实际软件系统也比较关心性能。对于运行时软件失效的分布和性能下降的nature具有随机性。   1. **基于模型的分析方法**：用马尔科夫决策过程和随机Petri网等方法,建立系统的数学模型,解最优抗衰实施时间 2. **基于度量的分析方法**：用时序分析和机器学习识别资源耗竭和性能下降的趋势 3. **未来研究****[]**  * 开发混合方法，实现在线监测和抗衰评估框架 * 预测抗衰时机不应只针对失效（热备、集群很普及，它们主要就是失效触发的），虽然这对抗衰不应只是失效触发的，也应该是性能耗竭触发的。   这就需要预测时，不只是预测软件何时会失效（剩余寿命），还要根据软件性能要求(输入)，预测软件何时性能不够用(剩余可用寿命) | | 负载等环境因素对软件的状态有较大影响。   1. **决策实施抗衰的最优安排：**  * 基于分析模型的方法(如基于时间的抗衰) * 基于度量的方法（如基于预测的抗衰）  1. **目前实施的抗衰技术：**  * 应用无关的（各级重启：组件、应用、VM、节点重启,集群） * 应用相关的：垃圾回收(如java的GC)、内核表刷新,需支持  1. **未来研究[]**  * 更有效的抗衰技术，降低抗衰成本(在线、微重启) * 解释state of the application * 实现额外的机制，通过有选择的部分执行抗衰来最小化宕机时间   OS和中间件、编程语言中融入抗衰策略 | | |
|  | **抗衰实施技术** | | | | | | | | |
|  | 2004 | G. Candea ,S. Kawamoto, Y. Fujiki, G. Friedman,Fox, A. | Microreboot - A technique for cheap recovery  (ACM) |  |  |  |  |  |  |

1. Proactive management of software aging

Some typical causes of this degradation are memory bloating and leaking, unterminated threads, unreleased ﬁle-locks, data corruption, storage-space fragmentation, and accumulation of round-off errors.

Aging occurs because software is extremely complex and never wholly free of errors. It is almost impossible to fully test and verify that a piece of software is bug-free. This situation is further exacerbated by the fact that software development tends to be extremely time-to-market-driven, which results in applications which meet the short-term market needs, yet do not account very well for long-term ramifications such as reliability. Hence, residual faults have to be tolerated in the operational phase. These residual faults can take various forms, but the ones that we are concerned with cause long-term depletion of system resources such as memory, threads, and kernel tables. The essentially economic problem of developing and producing bug-free code is not the problem at hand; instead we address one of the problems that arises from the prevailing approach to developing software, and one approach to attacking that problem is software rejuvenation.

Most current fault-tolerance techniques are reactive in nature. Proactive fault management, on the other hand, takes suitable corrective action to prevent a failure before the system experiences a fault. Although this technique has long been used on an ad hoc basis in physical systems, it has only recently gained recognition and importance for computer systems. Software rejuvenation is a specific form of proactive fault management which can be performed at suitable times, such as when there is no load on the system, and thus typically results in less downtime and cost than the reactive approach. Since proactive fault management incurs some overhead, an important research issue is to determine the optimal times to invoke it in operational software systems. Proactive fault management can be greatly enhanced by the ability to predict the fault far enough in advance that one can take action to avoid or mitigate its effects. Resource exhaustion by its very nature offers clues that failure is imminent, in the form of parameters that can be monitored, extrapolated, and compared to thresholds via suitable algorithms.

One of the benefits of clustering is its natural redundancy of hardware and software components. A number of single points of failure can be removed by cluster systems.实际上，集群软硬件冗余避免单点失效的成功架构，对应用有很高的依赖性；而且为了保证有能力实现故障转移，**足够的冗余资源必须有能力承受/适应被转移过来的负载**。Another advantage of a clustered system is its ability to improve system maintenance.如果集群中有一个资源需要修理、修复，且有其他冗余资源可以使用，那么就可以有计划的把需要修复的资源上的负载转移出去，然后停掉、撤下来修复。

软件抗衰相比软件冗余等事后容错方法，是一种预防性的容错方法；它也被称为“软件再生”。

1. 软件衰退发现与应对要解决的本质问题

检测、评估软件的性能/健康状态，预测系统性能恶化/失效的时间，选择合适粒度的抗衰策略，并执行。

1. 现有方法已解决的本质问题
2. 未来研究尚需要解决的关键问题
3. 软件老化潜在的原因

软件老化潜在的原因有内存泄露、未释放的文件锁、未终止的线程的累积、数据越界错误的增长、文件系统碎片、共享内存池封锁（死锁？）等等[18]

1. 徐建，软件抗衰若干关键技术研究，2007

数据分析方法：

1、朴素贝叶斯：可针对多行为建模？