1.1 主要目标（导读）

（可用性要求高的软件在经济、社会、科技方面的作用、地位，重视程度提高，要求提高→软件健康(可用性)的重要性

软件健康面临的威胁→入侵只是一小部分，更多的是自身的衰退/老化(age) →软件性能下降、失效率上升，不能按要求完成既定的功能：损失（以秒计）；入侵可由安全系统（杀毒软件、IDS、防火墙）解决，衰退却不行(找权威)

衰退的关注度提升（业界、学术界），但在运行时，不易排查、排除软件衰退的原因→呼唤容错

传统软件容错方法：代码多样性（N版本）、热备/集群等（银行、股票交易等系统限制/禁止使用集群[找权威]）→存在的问题(被动式，失效发生后触发;1.性能早不能满足要求；2.切换需要几分钟；3.其他服务器超载，怎么办？)→呼唤预防性/主动性容错机制：软件抗衰；

软件复杂度、复用程度的提升，面临新的问题→呼唤新的方法、思路→应满足的要求、特点）

* + 1. 目标

{

软件衰退的定义：内部缺陷（研究分类、行为特征）/不当的维护→逐渐状态退化→性能（服务率/服务质量）下降/失效率增加→表现：生理特征(系统/应用资源、服务能力)

内部缺陷的不可避免性（普适性）→衰退的必然性→研究的必要性

对可用性要求高的软件造成的损失：经济、社会生活、政治、科技方面

{宏观描述＋案例分析: [股票交易系统], [电子商务(购物、售票)], [航空航天]

总结：（目的是突出重要程度，体现研究的迫切性）}

业界、学术界加大关注→容错→传统容错存在的问题：牺牲硬件成本、被动

预防性容错机制-软件抗衰的重要性、基本思路

***新需求：新bug涌现、软件内外部环境动态变化、不确定因素[]***

学术界研究的重点（主要步骤）}

* + 1. 我
    2. 抗衰研究现状

{抗衰的基本思路，在软件可用性保障中的重要性。

现有抗衰方法的特点、作用：从分析方法{基于模型的、基于度量的、基于AIS的抗衰方法}和解决的主要问题、步骤这两个维度展开。

分析未来的趋势和可能的状况（应用服务软件的网络化、虚拟化），可能存在的新问题：复杂的致衰fault、若干fault可能一起出现并相互影响、多指标间相互关联/覆盖，新bug涌现、多样性）。

现有方法在解决问题时存在的不足分析：依赖人工经验选择软件健康评价指标，依赖衰退数据构建分析模型，未充分考虑上下文环境；研究的问题集中在内存泄露和响应时间下降方面（单独考虑）。

现有方法实现了哪些步，哪些步骤还有不足；

说明人工免疫引入的必要性、可以解决哪些问题}

上述方法获取资源使用时序数据的趋势后，结合趋势和当前实时数据，预测资源使用数据达到阈值的时间，作为软件可能失效的时间。当这种思想应用于实际软件中时，存在如下问题：保证不同类型的应用软件能够正常运行的资源阈值不易确定：大型电商应用等软件系统（尤其是云计算平台）的资源分配、回收频繁，极易造成资源使用数据瞬时突变，这种噪声数据会对较小的阈值做出挑战，产生误报（没必要的抗衰）；而较大的阈值，易造成漏报（错失抗衰时机而崩溃）。而多元模式识别方法避免了在检测敏感度和误报之间的权衡，（这些参数在监测系统中可调）使得监测系统可以引入一些标准的可用性度量标准。

该方法假设时序数据中存在全局趋势，且对时序数据的周期性考虑依赖了目测效果；同时，孤立地考虑了实验指定的单个特殊指标的衰退导致的失效，未考虑衰退的综合效果，亦未考虑多个指标间的交互性、相关性和它们对系统可用性的影响，因此方法不具通用性；

因为软件衰退现象发现的初期，研究者普遍发现衰退的主要体现是操作系统资源的耗竭，这其中最易观测到的是内存泄露问题，因此将内存泄露作为软件衰退研究的主要对象，如内存泄露、碎片，预测内存会损耗的时间（失效时间）。故多篇论文根据经验，直接选取可以刻画内存泄露的参数作为软件衰退度量的指标，如物理内存使用情况和交换空间使用情况，没有考虑性能和业务遭受的影响。这种做法可能对检测操作系统的内存泄露有效，但会带来如下不足：

1. 只针对个别衰退现象和影响，丢弃了很多对后续提炼抗衰决策规则有用的参数，而只选择一个参数不足以识别复杂的衰退现象；
2. 以人工经验选取度量指标的方法，一般根据单个指标是否达到阈值，判定软件衰退恶化的程度、是否会失效。而软件运行的环境具有动态多变性，根据经验选择的阈值不一定适用于所有场景。

应该在业务受到影响的时候，就执行一定级别的抗衰，恢复业务，或将业务转给其他虚拟机。

上述基于时序分析的方法均假设系统资源时序数据存在全局的上升或下降趋势，因此针对单参数进行衰退预测，依赖于用实际监测的数据进行模型/曲线拟合分析和投影，然后在一定的预测时间范围内，对比投影的数据和预先设定的上、下限经验阈值，可以较准确地预测如内存泄露引起的软件衰退问题。但仅掌握假设的数据的整体趋势并无法准确的预测未来资源的耗费情况（如：交换空间使用量）和执行抗衰策略的安排，原因是可能出现局部突变（变异local variation），因此需要使用多元模型分析多种资源之间的相互作用，考虑软件系统性能和业务遭受的影响。