浙 江 大 学

硕士学位论文开题报告

**（专业学位）**

论文题目：基于Web的医疗保险信息管理系统的性能测试与

分析

姓 名： 王艳灵

学 号： Z134325199

专 业： 软件工程

院 别： 软件学院

导 师： 李善平

二零一四年 十一月

**目 录**

[1.课题来源及类型 - 1 -](#_Toc17289)

[2.课题的意义及国内外现状分析 - 1 -](#_Toc3501)

[2.1基于Web的医疗保险信息管理系统概述 - 1 -](#_Toc26009)

[2.2软件性能测试概述 - 2 -](#_Toc829)

[2.3医疗保险信息管理系统性能测试的提出 - 2 -](#_Toc8387)

[3.课题的研究目标、研究内容和拟解决的关键问题 - 3 -](#_Toc28875)

[3.1课题研究目标 - 3 -](#_Toc8984)

[3.2课题研究内容 - 4 -](#_Toc4213)

[3.3拟解决的关键问题 - 5 -](#_Toc14660)

[4.课题的研究方法、设计及试验方案，可行性分析 - 6 -](#_Toc13079)

[4.1性能测试方法 - 6 -](#_Toc21513)

[4.1.1强度测试 - 6 -](#_Toc17501)

[4.1.2负载测试 - 6 -](#_Toc16022)

[4.1.3压力测试 - 7 -](#_Toc24638)

[4.2课题的设计方案 - 8 -](#_Toc13432)

[4.3课题的可行性分析 - 8 -](#_Toc3495)

[5.课题计划进度和预期成果 - 9 -](#_Toc15417)

[5.1计划进度 - 9 -](#_Toc3862)

[5.2预期成果 - 9 -](#_Toc5106)

[5.2.1性能测试结果分析 - 10 -](#_Toc374)

[5.2.2系统性能调优方案提出 - 11 -](#_Toc10064)

# 1.课题来源及类型

随着软件行业的发展，计算机软件的使用规模越来越大。Web系统的应用也迅速普及，用户群的迅速扩大，对于大量用户访问Web系统的性能提出了更高的要求。整个社会对软件质量的要求越来越高，软件性能测试成为了提高软件质量的一个关键保障。基于此，本文结合投保人医疗保险信息系统的特点以及软件测试的管理，研究如何在Web项目的开发过程中进行性能测试，以此提高软件开发的整体质量。

# 2.课题的意义及国内外现状分析

## 2.1基于Web的医疗保险信息管理系统概述

随着社会的发展，人们对健康越来越关注，医院和药店的业务不断增长，处理越来越复杂，简单的人工经营管理，已无法满足医院和药店的需要，成为医院和药店发展的瓶颈，人们对医疗相关的各项服务的要求也越来越高。传统的手工操作模式，病人帐目繁杂，病人查帐困难，职工的医疗保健系统和报销的问题出现错误现象严重且无法追踪，人工传送各种记帐单容易漏、错帐，人工划价收费差错难以避免，因此，如何提高服务水平、提高工作效率，成为迫待解决的问题。因此医疗保险信息系统恰好满足了这些需求。

医疗保险信息管理系统（Health insurance information management system）是为了使病人获得基本的医疗保险服务，又能给医院和药店提供快速便捷的信息交互功能。这个系统涉及到的部门非常广，如参保单位、银行、税务部门、定点医疗机构、定点零售药店等，各部门的密切配合，使资源得到充分合理的利用，通过与定点医疗机构、定点零售药店建立网络连接，来实现投保人在就诊和购药时，使用医保卡就能实时支付的功能。最后通过建立计算机管理信息系统，实现医保业务处理计算机化。本系统是医院管理信息系统的子系统。在本系统中要实现的系统模块是投保人信息管理和保单生成的医疗保险信息管理系统，主要包括投保人的信息、投保药品、药店、和投保人理赔参数的设置等，达到一个提取信息的功能。另外，由于基于Web服务器的应用系统由于提供浏览器界面而无须安装，大大降低了系统部署和升级成本，得以普遍应用。目前，很多企业的核心业务系统均是Web应用，用户可以享用更丰富的功能和体验，所以，在这里研究软件测试的结合项目是基于Web的医疗保险信息管理系统。

## 2.2软件性能测试概述

系统的性能是个很大的概念，覆盖面非常广泛，对一个软件系统而言，包括执行效率、资源占用、稳定性、安全性、兼容性、可扩展性、可靠性等等。也即是，性能是指表明软件系统或构件对于其及时性要求符合程度的指标。属于产品的特性范畴，可以用响应时间、吞吐量、每秒点击数等参数指标来进行衡量。对于互联网应用软件，性能是其质量的一个非常重要的组成部分。作为解决软件性能问题的重要手段，软件性能测试已经广为人们所熟悉，并受到很高的关注。性能测试本质上是一项规范，它利用有关产品、人员和过程的信息来减少应用程序、升级程序或修补程序部署中的风险。主要是通过自动化的测试工具模拟多种正常、峰值以及异常负载条件对系统的各项性能指标进行的一种测试。Web性能测试是借助测试工具来模仿真实用户对Web系统进行施压，在正常和疲劳情况下检测系统的各项指标，预测系统是否满足特定的场景和发现系统瓶颈的过程。主要目的是在并发环境下，评估系统的承受力、发现系统弱点，系统优化和保证系统长时间运行的可靠、安全和稳定。

性能测试可以从狭义和广义两方面考虑，狭义的性能测试是通过模拟生产运行的业务压力或用户使用场景来测试系统的性能是否满足生产性能的需求。广义的性能测试则是压力测试、负载测试、强度测试、并发测试、容量测试、配置测试、可靠性测试等。

到目前为止，性能测试仍然是控制性能的非常有效的手段，在软件的能力验证、能力规划、性能调优、缺陷修复等方面都发挥着重要作用。

## 2.3医疗保险信息管理系统性能测试的提出

目前，作为和每个人切身利益相关的医疗保险信息系统被大家尤为关注，那么这其中，投保人的医疗保险的各项信息的整合和管理就变得比较复杂，传统的手工模式根本无法应付这项工程，于是，医疗保险信息系统就自然而然的出现在人们的面前。随着各种信息管理系统Web应用的兴起，基于B/S结构的软件日益强劲发展，正在成为未来软件模式的趋势，所以，这里的医疗保险信息系统也是基于这种模式的。然而，当一个Web应用被开发并展现在用户、供应商或合作伙伴的面前时，尤其是即将被部署到实际运行环境之前，用户往往会疑问：这套Web应用能否承受大量并发用户的同时访问？系统对用户的请求响应情况如何？在长时间的使用下系统是否运行稳定？系统的整体性能状况如何？如果存在性能瓶颈，那么是什么约束了系统的性能？而这些正是Web性能测试解决的问题，如何有效进行Web性能测试，目前并没有一个系统和完整的回答。那么，如何对一个基于Web的医疗保险信息管理系统有效的进行性能测试，更是没有一个系统而完整的答案。此外，由于紧凑的开发计划和复杂的系统架构，Web应用的测试经常是被忽视的，即使进行了测试，其关注点也主要放在功能测试上。但是，近年来Web性能测试越来越引起重视，成为Web系统必不可少的重要测试内容。本文的研究就是基于这种需求，进行基于目标的Web性能测试方法。

开发的软件质量有好有坏，参差不齐，如果想对这些软件的质量进行把关，或者说想尽可能的提高软件的质量，就需要在软件工程过程中对软件进行测试。因此一些企业或者软件开发人员会对软件进行一些功能测试，但功能测试只是对系统的基本功能做相关的测试，并没有考虑系统性能相关问题，一旦因为性能问题系统崩溃，将会给用户带来很不愉快的上网体验，也会因此给相关机构和企业带来损失。

针对基于Web的医疗保险信息管理系统的性能测试的专门研究又不是很系统，因此，本文为基于Web的医疗保险信息管理系统的质量，而尝试进行一些性能测试的研究是很有意义的。

# 3.课题的研究目标、研究内容和拟解决的关键问题

## 3.1课题研究目标

本文以JMeter为测试工具，对基于Web的医疗保险信息管理系统进行性能测试，以发现性能瓶颈，以提出提高其质量的调优方案。

研究目标是：结合这个基于Web的医疗保险信息管理系统的特征，不仅用测试工具去运行一些测试脚本来证明产品是否可以达到性能指标，更关键是要发现产品性能上的缺陷，并解决定位问题，这才是软件性能测试的真正目的。其目标可分为以下两种级别：

（1）性能测试总体目标。找出这个医疗保险信息管理系统可能存在的性能瓶颈或者软件缺陷，确认其是否可以达到用户的需求。收集测试结果并分析产生缺陷原因，提交总结报告，让软件开发方对此系统进行性能改进。

（2）性能测试具体目标。此目标又可以分为：

确定此系统的总体性能参数，包括所支持的最大并发用户数、事务处理成功率、请求相应的往返延迟等；

确定在各个级别的负载及压力测试下服务器输出的具体性能参数；

系统耐力强度的判定。

这些测试目标驱动了整个测试过程的进行，因而在此系统性能测试中起着至为关键的核心作用。

## 3.2课题研究内容

本文主要是研究的是使用JMeter针对基于Web的投保人医疗保险信息系统性能进行测试，找出系统中需要改进的地方，使系统尽可能的完美。

本文主要内容包括：

1、业务场景选取

熟悉系统应用，根据系统开发的逻辑来了解整个系统的架构，这样才能知道测试中需要模拟的应该是什么样的请求。

了解和分析测试需求，确定测试业务场景和和性能测试指标。

确定测试业务场景：根据用户需求中所描述的系统架构来分析性能测试的重点表现在哪几个方面，如：并发访问的性能、批处理业务执行效率、系统处理的稳定性等，同时结合系统架构分析可能存在的性能瓶颈，这个也是选择测试业务场景的基础。

根据用户需求并结合实际，尽量做到对不同业务类型和操作类型的覆盖，如；对数据库的读操作（简单查询、复杂查询）、数据库的写操作（如：插入、删除和更新），选取一些具有代表性的业务操作作为测试业务场景。

分析性能测试指标：

（1）根据用户需求和所选取的典型业务场景，首先选取性能指标项，如：并发用户数、交易响应时间、每秒交易数（TPS）、服务器资源占用率、网络吞吐量等；

（2）在选取性能指标项后，根据用户需求或总体设计文档对性能指标进行分析，获得各个性能指标项值，如：用户需求中说明了某个业务使用的用户数为N，但没有说明并发用户数，可以按照10%N~20%N来设置并发用户数。

2、搭建测试环境：

（1）测试环境搭建尽量与真实运行环境大致一样，这样测试结果才有参考价值。

（2）营造独立的测试环境，被测系统在预期性能测试执行期间应保证其资源独占性，即测试过程中要确保我们的测试环境独立，避免测试环境被占用，影响测试进度及测试结果，比如设备连网后，如果其他开发组或测试组也在共用，这样就可能影响我们的测试结果。有时开发人员为确定问题会使用我们的测试环境，这样会打乱我们的测试活动，更严重的是影响测试进度。因此需要为本次测试搭建独立的测试环境。

（3）构建可复用的测试环境，项目实际执行过程中，测试环境是经常变化，比如测试软件版本更新、测试人员流失等等，需要随时跟踪和改进，尽量将可控的资源进行分类整理。可控资包括：测试环境配置手册、测试硬件信息、环境变更记录等等，目的是尽量将测试环境进行备份，方便出现未知问题时快速的还原。当刚搭建好测试环境，安装测试软件之前及测试过程中，对操作系统及测试环境进行备份是必要的，这样一来可以为我们下轮测试时直接恢复测试环境，避免重新搭建测试环境花费时间，二来在当测试环境遭到破坏时，可以恢复测试环境，避免测试数据丢失，重现问题。

3、测试数据准备

在实施性能测试时，需要运行系统相关业务，这时需要一些数据支持才可运行业务，这部分数据即为初始测试数据，数据准备和清理的工作量是非常大的，需要在测试前提前考虑。为更加真实的模拟现实运行环境，我们在测试过程中，应尽可能准备与真实业务执行相一致的初始数据，如系统用户数据、业务数据、辅助数据等。

（1）系统用户数据：登陆系统使用的帐户名、口令等，数量与虚拟用户数一致；

（2）业务数据：每个虚拟用户模拟真实用户进行操作时使用到的数据；

（3）辅助数据：为保证业务操作的正常进行而设置的基本信息资料。此外，测试数据可分可重用和不可重用数据：

（4）可重用数据：如客户信息等查询类的数据，此类数据只需一次准备即可；

（5）不可重用数据：此类数据为一次性消耗数据，不可重用，一般应用在数据增加或修改类业务交易，此类数据如增加客户标识、帐户标识等。

4、性能测试实施

录制脚本、修改脚本参数、调试脚本和执行脚本。

5、性能测试结果分析

分析测试结果，获得执行脚本过程中所收集的性能指标值的分析报告，可以通过分析报告中的数据，得知系统的性能情况。

最后，本文在总结性能测试结果，并分析性能瓶颈，得出建议优化措施，进而提高该系统的性能，使软件质量得到进一步的保障。

## 3.3拟解决的关键问题

基于Web的医疗信息管理系统的各方面功能已经被测试过，趋于稳定，但是性能问题有待解决，比如Web页面反应缓慢、系统事务响应速率低的用户需求响应及时度等的性能问题，这些问题有待性能测试之后提出性能调优方案，优化件的稳定性与可靠性，来进一步保证软件的质量。

本课题中将会根据此系统的特点使用JMeter性能测试工具，采用压力测试、负载测试、疲劳测试的方法对系统性能进行测试。生成相应的测试结果（如查看结果树、聚合报告等），但是JMeter生成的结果不够直观，这里会使用JMeter插件生成相应的结果图，然后根据测试结果对该系统的性能情况有个初步的判定，分析该系统性能缺陷，得到优化措施，提高系统的质量和稳定性，降低系统发布后不必要的风险。

# 4.课题的研究方法、设计及试验方案，可行性分析

## 4.1性能测试方法

### 4.1.1强度测试

通常是为了确定系统可以维持持续的预期的负载。在测试期间，内存利用率是用来检测系统潜在的缺陷。同样重要，但人们往往忽视，那就是系统的性能退化。即确保吞吐量和/或响应时间经过长时间的持续活动和刚开始测试时一样好或者比刚开始测试时还要好。这实际上涉及到重要负载系统申请延长，重要的一段时间。我们的目标是发现系统在持续使用时有什么样的反应行为。

疲劳强度测试可以被当作是一个长期的负载或压力测试，它是采用系统稳定运行情况下能够支持的大并发用户数，持续执行一段时间业务，通过综合分析交易执行指标和资源监控指标来确定系统处理大工作量强度性能的过程。疲劳强度测试可以采用工具自动化的方式进行测试，也可以手工编写程序测试，其中后者占的比例较大。一般情况下以服务器能够正常稳定响应请求的大并发用户数进行一定时间的疲劳测试，获取交易执行指标数据和系统资源监控数据。如出现错误导致测试不能成功执行，则需要及时调整测试指标，如降低用户数、缩短测试周期等。还有一种情况的疲劳测试是对当前系统性能的评估，用系统正常业务情况下并发用户数为基础，进行一定时间的疲劳测试。

### 4.1.2负载测试

负载测试是一种主要为了测试软件系统是否达到需求文档设计的目标，譬如软件在一定时期内，最大支持多少并发用户数，软件请求出错率等，测试的主要是软件系统的性能。其实，负载测试与压力测试很相似，也是寻找系统性能指标的极大值。例如500用户点击提交按钮，服务器内存使用率达到98%，可以初步认为内存是瓶颈，为下一步详细判断提供依据。负载测试则侧重于连续测试的时间。在这个系统测试过程中，会按一般负载测试和渐增负载测试两种情况进行：

最基本的一种负载测试用于确定Web应用程序的行为在正常和预期的峰值负载条件下，负载测试有助于识别应用程序的最大操作能力以及任何可能妨碍其操作能力的瓶颈。

一般负载测试：是模拟场景时，使负载是平时正常负载的多倍。

渐增负载测试：当开始负载测试，先从一个小数量的虚拟用户，然后逐步增加负荷从正常到峰值。然后可以观察应用程序在这个逐渐增加负载条件下是有什么样的翻译。最终，会超过一个性能目标阈值限制值。例如，你可能会继续增加的负载，直到服务器处理器利用率达到75%时，或者当最终用户响应时间超过8秒。

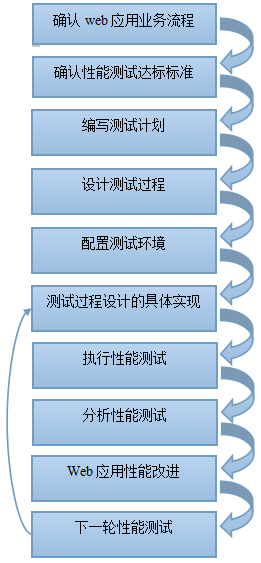
### 4.1.3压力测试

压力测试偏重于施压，软件压力测试的基本思路很简单：不是在常规条件下运行手动或自动测试，而是在计算机数量较少或系统资源匮乏的条件下运行测试。通常要进行软件压力测试的资源包括内部内存、CPU可用性、磁盘空间和网络带宽。压力测试主要是为了测试硬件系统是否达到需求文档设计的性能目标，譬如在一定时期内，系统的CPU利用率，内存使用率，磁盘I/O吞吐率，网络吞吐量等，压力测试和负载测试最大的差别在于测试目的不同。负载测试是模拟实际软件系统所承受的负载条件的系统负荷，通过不断加载（如逐渐增加模拟用户的数量）或其它加载方式来观察不同负载下系统的响应时间和数据吞吐量、系统占用的资源（如CPU、内存）等，以检验系统的行为和特性，以发现系统可能存在的性能瓶颈、内存泄漏、不能实时同步等问题。负载测试更多地体现了一种方法或一种技术。压力测试是在强负载（大数据量、大量并发用户等）下的测试，查看应用系统在峰值使用情况下操作行为，从而有效地发现系统的某项功能隐患、系统是否具有良好的容错能力和可恢复能力。压力测试分为高负载下的长时间（如24小时以上）的稳定性压力测试和极限负载情况下导致系统崩溃的破坏性压力测试。

确定出现的应用程序问题，或只有在极端条件下才变得明显的应用程序问题。这些条件包括高负载、高并发、或有限的计算资源。适当的压力测试是有用的在寻找同步和定时错误，联锁问题，优先级的问题，和资源损失错误。这个想法是强调一个系统的突破点，寻找漏洞，将打破潜在危害。该系统预计不会处理过载没有足够的资源，但行为(例如，失败)在一个可接受的方式(例如，不损坏或数据丢失)。

## 4.2课题的设计方案

首先，在一个系统进行性能测试之前，要了解系统的逻辑结构、物理结构和系统结构，以方便确定在性能测试中建立什么样的需求；然后熟悉业务流程，确定关键路径和导航路径；然后根据需求和业务流程，确定性能测试的目标以及性能测试的各项指标；设计测试计划，并配置测试环境，使用JMeter测试工具进行测试，然后根据JMeter生成的结果图和JMeter插件生成的结果图进行分析，得出系统性能缺陷，提出优化方案，并在系统性能改进后进行重测，以确定系统得到优化，并可能发现新的性能问题。具体过程如图4.1：



**图4.1性能测试流程**

## 4.3课题的可行性分析

基于Web的医疗保险信息管理系统的性能测试与分析的可行性分析可以从以下几方面进行：

（1）经济可行性分析

经济可行性分析中最重要的内容之一是成本-效益分析。对于一个基于Web系统的性能测试要在经济方面评价其是否合理，成本-效益分析要估计出系统性能测试过程中的花费并与收益衡量比较。由于JMeter是开源软件，测试工具成本可忽略不计，并且它可以消除手动测试那种既耗时又单调，而且需要投入大量的人力资源的问题，使用JMeter能达到预期的效益，所以本课题在经济上是可行的。

（2）技术可行性分析

就目前使用的性能测试技术来说，系统的性能测试目标能够达到；利用现有的技术在规定的期限内软件性能测试工作基本能够完成，所以技术上是可行的。

（3）操作可行性分析

该性能测试旨在找出系统的性能瓶颈，提高系统性能质量。所以操作上是可行的。

基于以上分析结果，我们认为本文的研究方法和设计方案切实可行，可操作性高，研究结果也具有很高的实用性，课题具有一定的研究价值。

# 5.课题计划进度和预期成果

## 5.1计划进度

|  |  |  |
| --- | --- | --- |
| **开始时间** | **结束时间** | **主要工作内容** |
| 2014年10月 | 2014年11月 | 查阅文献资料，编写课题开题报告 |
| 2014年12月 | 2015年01月 | 进行整理性能测试需求工作，确定各项指标，和测试计划 |
| 2015年01月 | 2015年03月 | 配置测试环境，进行测试实践工作 |
| 2015年03月 | 2015年04月 | 分析测试结果，并整理分析报告 |
| 2015年04月 | 2015年05月 | 撰写论文正文 |

## 5.2预期成果

总结全文，本文预期取得的主要研究成果如下：

### 5.2.1性能测试结果分析

由于不同的应用系统，不同的测试目的，不同的性能关注点，所以在性能测试中需要具体问题具体分析，这也是指导性能测试的分析原则。查找性能瓶颈需要按照由易到难的的顺序进行，顺序如下：服务器硬件瓶颈、网络瓶颈（对局域网，可以不考虑）、服务器操作系统瓶颈（参数配置）、中间件瓶颈（参数配置，数据库，web服务器等）、应用瓶颈（SQL语句、数据库设计、业务逻辑、算法等）。但是以上过程并不是每个分析中都需要的，要根据测试目的和要求来确定分析的深度。对一些要求低的，分析到应用系统在将来大的负载压力（并发用户数、数据量）下，系统的硬件瓶颈在哪儿就够了。在性能分析中，分段排除法很有效果，分析的信息来源包括：根据场景运行过程中的错误提示信息；根据测试结果搜集到的监控指示信息。其中监控指标数据分析包含：最大并发用户数、业务操作响应时间、服务器资源监控指标（内存，处理器，磁盘I/O）、数据库服务器资源监控。

根据每次测试过程中监控到的系统资源使用情况，以及借助于自动化工具给出的测试结果分析报告或采用脚本(或分析程序)对日志文件进行统计分析的结果分析报告，对照测试目的，从而得出测试的结果是否通过的结论。同时对测试分析与统计的结果要进行评估，评估测试结果与真实理论结果之间的偏差。并写成测试报告。如果测试结果与期望值差距太大，要在测试报告中分析原因并写出自己的改进建议供研发人员参考。

本文是基于JMeter工具进行性能测试，以下内容详细说明下该工具中的测试分析结果。断言（Assertion）通常是用于对每一个请求取样器（request sampler）进行额外验证的工具，它可以对测试结果进行简单的分类。当进行一个功能的性能测试时，需要在对应的请求取样器下面添加断言，执行一次脚本，测试结果就会存放到.jtl文件中，这也是本研究课题预期成果的数据来源。通过JMeter生成的聚合报告（Aggregate Report）、树状列表（View Results Tree）可以显示得到通常意义上性能测试所关心的几个结果，这些都是研究成果的数据组成部分。而且在JMeter中的.jtl文件存放了测试的所有结果，总的来说，对JMeter的结果分析，主要就是对.jtl文件中原始数据的整理。研究课题最终通过分析以上的数据得出以下的成果：①收集性能测试的各项指数，对系统性能情况进行判定，并找出测试中可能存在的问题；②根据系统性能报表找出系统中存在的缺陷和瓶颈，并撰写测试报告，最终以文档形式展现出来。

### 5.2.2系统性能调优方案提出

根据测试工具的性能测试结果分析报告中得出的系统缺陷与性能瓶颈，在根据其性能指标，以及所处于层级决定选择优化的方式方法。本文最终提出该系统相关的调优的方案。

具体的调优方法如下：数据库优化；应用层优化；Web优化；架构优化递进。很多应用，很多应用，优化数据库往往是最直接、最方便、见效最显著的，但并非所有的系统性能都处在瓶颈，或者数据库瓶颈解决之后，可能应用层瓶颈，Web层瓶颈，甚至架构瓶颈都会冒出来了，所以数据库优化十分重要，但往往很多人理解系统优化就是数据库优化，是不全面的。优化角色一般推荐具备较深数据知识的程序员，或者专业的DBA，而不只是会CRUD开发人员。应用层优化侧重于应用层本身的逻辑优化，算法优化，代码优化等，优化的角色可以是熟悉应用的程序员。Web优化偏向于熟悉前端开发的技术人员。架构优化更侧重偏向于物理层面，涉及变更架构的，需要应用具有良好的拓展性，考验架构师平时的功底。但并非架构做的越强大、越灵活、越可配置、越易水平拓展就是越好的，其一：考虑此应用的投入产出比，换言之，是否值得投入这么多架构设计成本；其二架构设计也是具有一定的时效性的，IT速度太快了，今天的好东西未必是明天的好东西，越灵活的架构，就意味着存在更多的配置项，从某一方面，反而增加了系统的复杂度。很多大型成熟的应用，也并非一蹴而就，而是通过不断的调整优化，不断变更架构的。圣人也并非天生的，而是不断的总结，提炼，优化，重构。

性能优化未必一次性就能满足的，可能此处瓶颈消失了，系统一旦运转快速后，在其他地方又发现新的性能瓶颈，所以性能优化是一个迭代的工作。直至满足系统需要的性能指标。而且在性能优化中还必须考虑优化成本的问题，系统性能设计或优化是否可以一步升天，按照最好的分布式架构进行设计和优化呢？单个节点一直运转健康，理论上是可以达到的，但实际实施层面不可取，必须结合实际的非功能需求进行设计和优化，否则一步到极致的话，系统的成本太过于庞大，光是性能设计和优化的成本就高于系统本身给客户所提供的价值，也造成研发成本开销过大。