



下载APP



## 20 | 性能看护：如何更好地守护产品性能？

2021-07-01 尉刚强

《性能优化高手课》

课程介绍 &gt;

**讲述：尉刚强**

时长 15:46 大小 14.45M



你好，我是尉刚强。

前面几节课，我们学习了基准测试的相关技术和方法，知道了如何选择合适的性能测试工具。而所有的这些技术和手段，最终目的其实都是为了更好地看护软件的性能，并更好地支撑软件设计与优化。

但实际上，对目标软件的性能测试与看护，是一项工作量投入比较大的软件工程活动。所以在以往的咨询工作中，我就发现有不少的研发团队，虽然他们有做好软件性能看护的意识，可是在实战过程中，由于没有系统方法论的指导，中间很多工作的开展效率会比较低，而且经常会走一些弯路，从而就出现成本与收益不匹配的情况，进一步就会丧失继续投入性能测试与看护的信心。



当然，也有不少的研发团队在项目开发的过程中，并不重视性能测试与看护环节，因此就很容易陷入产品性能恶化的泥潭中不能自拔。

所以，今天我们就一起来好好思考下，怎样才能更好地实现软件产品的性能看护。

不过我也要说明一点，就是这节课我并不是要讲解具体的工具和技术，而是要**帮你建立起高效实施性能测试与看护的核心价值观**。这是因为可复用的方法，往往会比具体的工具和技术使用技巧重要许多，它可以帮你把有限的精力用到刀刃上，这样你就可以花费更少的成本获得更佳的性能测试与看护效果。

所以在这节课中，我会给你分享我在之前参与过的众多性能优化项目中，不断摸爬滚打之后提炼总结出来的经验和方法。我把这些经验方法归纳为了三条指导原则，分别是**自动化、测试前置、测试驱动**。掌握了这三条指导原则以后，你就可以有效地提高性能测试与看护的工作效率了。

那么接下来，我们就从第一条原则“自动化”开始学习吧。

## 自动化

所谓的自动化，就是把所有重复性的手动工作，尽量交给机器去自动执行，这样就可以把人解放出来，做更有价值的事情。而我们知道，在对软件系统进行性能测试的过程中，中间潜在的繁琐、重复性的工作会非常多，所以如果可以把这部分工作都自动化，就可以大大提高性能测试的效率。

那么接下来，我就通过一个例子，来带你看看如何通过自动化来影响性能测试的效率。

我刚开始参加工作不久，有一段时间主要是负责项目的性能攻关测试工作。在每一轮性能测试的过程中，我都需要同时操作十几台电脑，并在每台电脑上启动一个测试程序，然后记录不同时间段的测试结果，并重新修改测试配置，一直重复这个过程直到完成下一轮的测试。

可是这个过程实在是太麻烦了，所以到后来，我终于下定决心编写了一个测试程序，帮我自动化地进行测试，从而让我省下了很多时间去学习业务代码。

实际上，在进行性能测试的过程中，不仅中间会包含很多像我这样非常繁琐的重复性工作，而且很多时候，有些重复性工作还比较隐蔽，它们会偷偷消耗你的时间，而你还无法及时发现。

所以，在做性能测试的时候，我们如何才能得知哪些环节可能存在重复性工作呢？以及我们应该怎么去避免呢？

其实，这里我们可以直接根据性能测试的各个阶段，来主动规避重复性工作。下面我就给你具体介绍一下。

### 测试数据准备阶段

虽然不同软件产品在性能测试的过程中，获取和准备测试数据的方法都是不一样的，但是中间的大部分工作都应该尽量做到自动化，比如说自动生成数据、通过代码抓取现网数据等。

这里我举一个真实的例子，之前我在测试一个产品的性能时所需要的数据，是从 MongoDB 中 dump 出来的 JSON 文件，但它与性能测试接口的请求格式并不一样。当碰到这种情况时，聪明一点的性能测试人员可能会找一个 IDE 去批量修改这些数据，但这样依旧会花费很多时间，而且下一次测试还需要再来一次。

那么我认为最佳的方式，就是通过代码来实现这个数据的转换过程，这样一次的代码开发成本，就可以节省下后续很多性能测试中需要的数据处理时间。

其实，不光是按照测试需求来生成数据，在现网中采集数据等环节，也都有很多重复性的工作。就拿这个例子来说，从 MongoDB 中 dump 文件也是重复性工作，它也可以做到代码自动化。所以，从项目管理的视角出发，你更应该关注这个阶段的自动化。

### 测试环境准备阶段

其实，在测试环境准备阶段也有很多工作可以自动化，比如测试软件、测试脚本部署、系统环境变量设置等工作，这些都可以通过代码化管理。我就举个简单的例子，比如你可以将测试工具安装标准化到 Dockerfile 中，然后基于容器化运行，并且将环境变量配置都通

过 Shell 命令来实现，这样就可以使得整个测试环境的准备过程完全自动化，不需要手工操作。

## 测试执行阶段

然后就是测试执行阶段，对于基于网络服务的很多性能测试工作来说，你就可以选择上节课我介绍的代码化性能测试工具（如 Locust、k6），来更好地支持代码自动化执行。对于很多嵌入式设备来说，定制化的性能测试工具其实也可以朝着自动化逐步演进，这样不仅可以提升测试效率，还可以降低人力成本。

## 测试结果记录阶段

最后，就是测试结果的记录工作。由于手动记录测试结果比较方便，所以这个环节经常容易被忽视。但在我之前参与的性能测试项目的分析过程中，就出现过数次因为测试结果记录丢失或者出错，导致整个性能测试返工的情况。

所以对此你也要注意，如果可以通过代码来实现自动化的话，就可以尽量避免出现这种记录丢失或出错的情况。

总而言之，这里我总结的是在性能测试的过程中，你应该从哪些环节去尽量挖掘可以自动化的点。你在做性能测试的时候，就可以参考这些要点，去寻找和分析那些影响测试效率的重复性工作，并将它们自动化。

接下来，我们继续了解第二条提升性能测试与看护工作效率的指导原则：测试前置。

## 测试前置

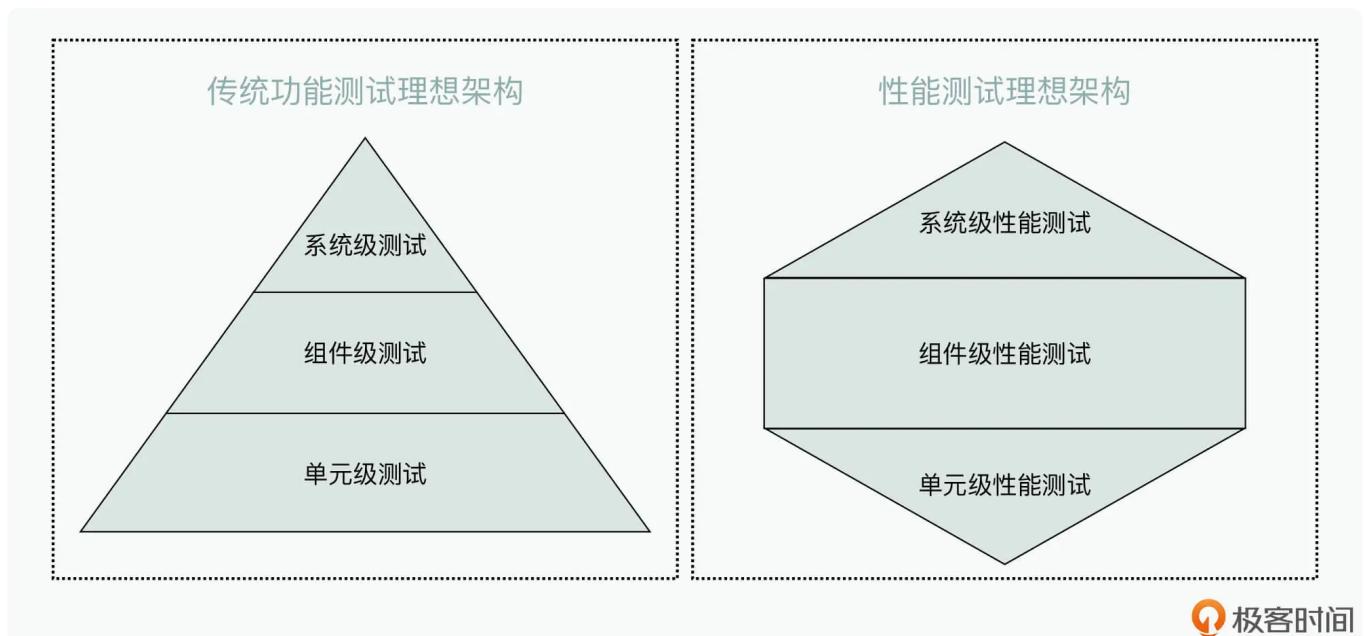
测试前置的意思就是**在软件生命周期中，尽早启动性能测试，尽早获取反馈**，而不是把性能测试只作为产品发布上线前的最后一个动作。我估计这里你可能要问了：**性能测试前置对软件性能看护来说有什么好处呢？**下面我就通过一个例子来给你分析下。

很早之前，我参与过一个嵌入式系统的性能主导重构项目，在项目的开发阶段，我们团队就搭建了针对该子系统的基准性能测试工程。虽然这套性能测试工程运行在通用 PC 上，与真实的嵌入式系统的硬件存在一定差异，但并不妨碍我们通过这套工程，来提前识别出很多系统在设计与实现的过程中潜在的性能问题。

就比如说，你可能也遇到过在代码中添加了 SQL 操作，引入慢查询的性能问题，而你应该也很清楚，这些问题在软件交付的后期会造成比较大的成本浪费。

所以，也就是从那个时候开始，我在参与接下来的性能优化项目时，就开始思考如何突破在软件后期才开展性能测试的传统思维。后来我发现，其实我们可以把性能测试工作尽快提前和尽量拆小，也就是说将更多的性能测试拆分成组件 / 服务级的性能测试，与核心模块的单元级性能测试一起来实现。而实现这个过程的指导原则正是测试前置，由此我就推导出了一个性能测试体系，可以帮助我更加高效地对系统进行性能测试与看护。

那么这个体系具体是什么呢？我们来看一张图，这是我总结的性能测试理想架构与传统功能测试理想架构的对比图。



传统功能测试从上到下大致可以分为三层，分别是系统级测试、组件级测试、单元级测试。而你可以看到，在图中的左边，功能测试的呈现层次的是金字塔形，也就是说其理想的软件测试分布规模为：大规模的单元级测试，仅次于单元测试规模的组件或服务级的测试，最后是少量规模的系统级测试。

而性能测试的理想架构如图的右侧所示，我认为应该由少量的系统级性能测试和单元模块级别的性能测试，与尽量多的组件或服务级的性能测试组成。

这是因为，对于全系统级别的性能测试来说，受制于系统的业务复杂性，容易导致测试场景和执行性能测试的成本非常大。就比如说，在我曾经参与的性能测试项目中，有些团队在全系统级别性能测试上投入了很大的精力，但最后的收益并不够理想。

而对于单元模块级别的性能测试（也就是🔗第 17 讲提到的微基准测试）来说，因为系统中的大部分业务代码并不是热点代码（依据 2/8 原则），所以对性能影响并不关键的业务逻辑而言，开发性能微基准测试用例的性价比并不是非常高。

补充：这里你要注意，对于一些系统性能的关键模块，比如核心算法模块来说，开发微基准测试其实还是比较有价值的。

那么现在，我们就来探究下为什么组件或服务级性能测试，其分布规模需要是最大的。

实际上，我在🔗第 18 讲中就提到过，如果将系统级的性能测试指标，分解到组件或服务级的性能测试上，可以降低测试的复杂度和成本，实现以大拆小的效果。而这样做，其实也更容易通过测试前置，更早地识别和发现性能问题（具体的分解方法步骤，你可以再去回顾复习下上节课的知识点）。

但是，目前大部分的软件研发团队，目光只聚焦在系统级的性能测试，而组件或服务级性能测试和单元模块级的性能测试这块是空白的，因此就比较容易陷入到系统级性能测试的各种复杂问题之中，花费很大的成本，但是收益并不高。这就是因为，团队没有采用性能测试前置的原则，去构筑更高效的性能测试体系。


## 测试驱动

OK，最后一个关键的指导原则就是测试驱动。你应该会想，这是什么意思呢？

在一般情况下，我们都会认为性能测试最主要的目标，就是获取产品的基线性能，这样在基线性能出现恶化的时候，我们就可以第一时间发现问题。那么除此之外，性能测试还有别的用处吗？

事实上，由于软件需求的不断演进和变化，为了保证软件的性能可以长期保持竞争力，我们需要从高性能设计、高性能编码实现、性能调优等多个维度一起入手，而**所有这些优化手段都应该基于性能测试来驱动进行**。

我为什么会得出这样的结论呢？其实在前面的课程中，我也已经多次给你阐述过这个观点。比如在🔗第 7 讲数据库选型设计中，我提到筛选的依据往往需要基于数据库的性能测试结果；在🔗第 2 讲并行设计架构模式中，我也讲过需要基于不同业务逻辑的性能测试结果

果来分解，而不是随便一刀切； **第 11 讲**针对数据结构和算法的选择，我也明确点出需要基于性能测试来调整优化才更有效。

所以首先，我们在确定高性能设计的关键决策点时，都应该有针对性地开发性能测试用例，并监控不同软件版本变更后的性能表现差异，这样才可以决策是否需要在软件设计上做调整。

而对于高性能编码实现来说，也是同样的道理。比如核心数据结构和算法的性能，也会随着业务数据的特性和规模改变而发生变化。因此，我们也应该针对这些核心模块，开发微基准测试用例，来监控分析性能的变化趋势。

**其实，测试驱动核心理念，就是在做软件设计优化、编码优化、性能调优的时候，都基于性能测试来驱动优化工作，而不是想当然。**

我举个例子。有的研发团队业务数据量规模非常小，查询计算逻辑比较复杂，因此开发人员就主观分析认为，使用 Elasticsearch 可以提升查询速度，然后就花费比较大的成本将业务数据都迁移到了 Elasticsearch 中，结果性能也并没有明显的改善。

但其实，如果该研发团队在决定迁移数据之前，先做好性能测试分析工作，可能就会发现这种解决方法并不能提升性能，这样就避免出现收益小且效果不好的情况了。

## 小结

今天这节课，我给你总结了曾经在很多的性能优化项目中，我实现高效性能测试的实践经验，我把它称之为高效性能看护的核心价值观。那么，在学习完今天的课程后，我希望你就可以借鉴这些经验来观察和思考下，在你的软件性能测试与看护过程中，是否也有一些环节可以改进，以此帮助提升自己的工作效率。

当然，我总结的可能并不全面，所以我更希望你可以在学完课程之后，主动去思考和总结下，在性能测试的过程中还有哪些要点对工作效率也很关键。然后，你可以提炼出一套独立的心得和经验，来指导自己或团队的工作。

## 思考题



在你参与的性能测试工作过程中，有没有哪些工作也是比较繁琐的，但是又不能很好地进行自动化执行呢？你可以留言分享出来，我们一起交流讨论，看看有没有其他的解决方法。

如果觉得有收获，也欢迎你把今天的内容分享给更多的朋友。

分享给需要的人，Ta订阅后你可得 **20 元现金奖励**

👍 赞 0

💡 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 19 | 性能测试工具：如何选择最合适的性能测试工具？

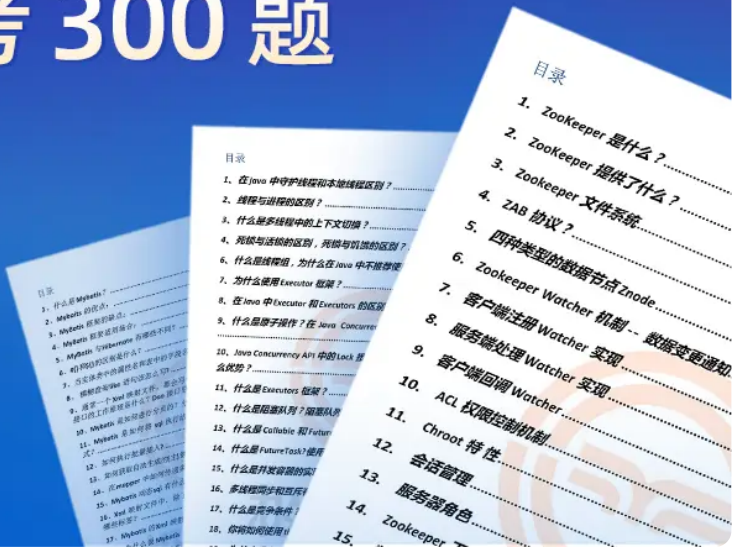
下一篇 21 | 性能CI：性能测试也可以集成到Pipeline中吗？

更多学习推荐

# Java 面试必考 300 题

## 最新汇总

限时免费领取 



### 精选留言

 写留言

由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。



