博文

系统设计黄金法则：简单之美 精选

已有 51563 次阅读 2012-4-23 11:03 | 个人分类:科研点滴 | 系统分类:科研笔记 | 系统设计, 黄金法则, 简单之美

【注：本文已发表在2012年第5期《中国计算机学会通讯》。】

最近多次看到系统设计与实现的文章与讨论，再加上以前读过的其他资料以及自己的一些实践教训，让我觉得应该把这些资料汇总整理一下。如果**要从讨论不同系统的众多资料中总结一条黄金法则的话，那只有一个词——“简单”；如果用一个英语单词来表达的话，那就是——KISS (Keep It Simple, Stupid!)。**

麻省理工方法与新泽西方法(MIT Approach vs. New Jersey Approach)【1】【2】

这个观点来自一篇很经典的文章，Richard Gabriel在1989年写的文章中的一节“The Rise of 'Worse is Better'”。说来惭愧，我是直到2011年5月在IBM T.J. Watson实验室听报告才第一次听说，当时便印象深刻。后来上普林斯顿的高级系统设计课程，发现这篇文章也在Reading List中，要求所有学生阅读然后在课上讨论。

“The Rise of 'Worse is Better”对比了以LISP系统为代表的麻省理工方法和以Unix/C为代表的新泽西(贝尔实验室)方法。Gabriel发现相比于LISP/CLOS系统完美的设计，Unix/C只是一味追求实现简单，但事实却证明Unix/C像终极计算机病毒那样快速蔓延，奠定了今天计算机系统的基础。

让我们来看看这两种不同的设计哲学。

1）MIT Approach

- 简单性：设计必须简单，这既是对实现的要求，也是对接口的要求。接口的简单要比实现的简单更加重要。
- 正确性：设计在任何值得注意的方面都要保证正确。不正确是绝对不允许的。
- 一致性：设计必须保持一致兼容。设计可以允许轻微少量的不简单和不完整，来避免不一致。一致性和正确性同等重要。
- 完整性：设计必须覆盖到实际应用的各种重要场景。所有可预料到的情况都必须覆盖到。简单性不能过度的损害完整性。

2）New Jersey Approach

- 简单性：设计必须简单，这既是对实现的要求，也是对接口的要求。实现的简单要比接口的简单更加重要。简单是设计中需要第一重视的因素。

包云岗
加为好友 给我留言
打个招呼 发送消息

扫一扫，分享此博文

作者的精选博文 全部
- 我心中的“创新”
- 伯克利科研模式的启发
- 我的2018：微信朋友圈的24个
- 人工智能在中国是否存在泡
- 与新晋图灵奖得主的虚拟对话
- 假期陪伴孩子学习的一点体会

作者的其他最新博文 全部
- 我心中的“创新”
- 伯克利科研模式的启发
- 处理器芯片开源设计与敏捷开
- 一个嘉奖真心做事认真做事的
- 非“首创”研究等于低价值吗?

精选博文导读 全部
- 本科毕业论文怎么写
- 科学史越全面，科学的未来...
- 论文写作中综述的价值
- 你健康吗?
- 平行电池：智能生态化电池...
- 写邮件

相关博文
- 【智库分析】爱思唯尔2021...
- Deadline—青年学者健康的“...
- 《自动化学报》多名作者入...
- 美捷登主编夏华向教授入选“...
- 杂谈：对于普通大众、研究...
- 榜上有名！MIR 11位编委、...

- 正确性：设计在任何值得注意的方面都要求正确。为了简单性，正确性可以做轻微的让步。
- 一致性：设计不能过度不兼容一致。为了简单，一致性可以在某些方面做些牺牲，但与其允许设计中的这些处理不常见情况的部分去增加实现的复杂性和不一致性，不如丢掉它们。
- 完整性：设计必须覆盖到实际应用的各种重要场景。所有可预料到的情况都应该覆盖到。为了保证其它几种特征的品质，完整性可以作出牺牲。事实上，一旦简单性受到危害，完整性必须做出牺牲。一致性可以为实现的完整性作出牺牲；最不重要的是接口上的一致性。

如果觉得这种哲学描述太抽象的话，原文中有一个关于Unix中断处理的例子，非常生动。一位MIT的教授一直困恼于Syscall处理时间过长出现中断时如何保护用户进程某些状态，从而让用户进程能继续执行。他问新泽西人，Unix是怎么处理这个问题。新泽西人说，Unix只支持大多数Syscall处理时间较短的情况，如果时间太长出现中断Syscall不能完成，那就会返回一个错误码，让用户重新调用Syscall。但MIT人不喜欢这个解决方案，因为这不是“正确的做法”。

Unix/C开发于1970年前后，那时离1964年刚推出的IBM System/360没几年，软件刚摆脱硬件束缚，能移植到不同的机器上，从而变成了一种可单独出售的产品。就是这样的一个软件产业的萌芽期，这种“实现简单”的理念被证明是更有效的。那么在今天的互联网时代，这种理念还有效吗？我们再来看下一篇文章。

来自互联网巨头们的教训【3】

这是最近看到的一篇文章，作者从High Scalability Blog上总结了几大互联网在设计后台数据中心所遇到的教训（这篇文章总结的非常好，强烈推荐大家读一下）。文章开头就总结了七个互联网公司（Google, YouTube, Twitter, Amazon, eBay, Facebook and Instagram）都提到的6点教训：

- **Keep it simple** - complexity will come naturally over time.
- **Automate everything**, including failure recovery.
- **Iterate your solutions** - be prepared to throw away a working component when you want to scale it up to the next level.
- **Use the right tool for the job**, but don't be afraid to roll your own solution.
- **Use caching**, where appropriate.
- **Know when to favor data consistency over data availability**, and vice versa.

第一点就是“简单”，但和New Jersey Approach的原因和内涵有所不同。不同于Unix时代相对简单的单机系统，互联网时代的大公司的系统往往都是成千上万台机器，在这样的系统上部署、管理服务（软件）是一项非常有挑战的任务。而为大规模用户提供的一项服务往往会涉及到众多模块、若干步骤。此时***“简单”就是要求每个阶段、每个步骤、每个子任务尽量采用最简单的解决方案，这是由于大规模系统内在的不确定性导致的复杂性决定的***。

即使做到了每个环节最简单，但由于不确定性的存在，整个系统还是会出现不可控的复杂性。比如，Google Fellow、美国工程院院士Jeff Dean最近在UC Berkeley有个报告【4】介绍他们努力缓解大规模数据中心中的Long-Tail Latency难题。问题简单描述如下：假设一台机器

处理请求的平均响应时间为1ms，只有1%的请求处理时间会大于1s (99th-Percentile)。如果一个请求需要由100个这样的节点一起处理，那么就会出现63%的请求响应时间大于1s，这样的系统完全是不可接受的。面对这个复杂的不确定性问题，Google他们做了很多工作，权衡各种Tradeoff，具体请看这个报告【4】。

大规模数据中心，看起来似乎和我们普通的开发人员离得比较远。但最近看Paul Graham写的《Hackers and Painters》这本介绍硅谷创业公司的书，发现Graham也在多处强调“简单”。

Paul Graham的《Hackers and Painters（黑客与画家）》

Paul Graham被称为“硅谷创业之父”。他在1995年和MIT的Robert Morris教授创办了Viaweb，于1998年被Yahoo!以4900万美元收购。2005年，他又创办了Y Combinator创业孵化器公司，帮助80多家创业公司成长起来，其中包括Dropbox(市值大于40亿美元)、Airbnb(市值大于13亿美元)等。显然，Graham有丰富的创业经验。

Graham在“设计者的品味”一章中写到，“好的设计是简单的”、“简单就是美，正如漂亮的数学证明往往是简短而巧妙的那种”。他提到，有些创业者希望第一版就能推出功能齐全的产品，满足所有的用户需求，但这种想法是致命的。***在硅谷创业最忌讳的就是“Premature Optimization”***。因为一方面用户需求是多样的，不同人群都有不同的需求；另一方面开发者想象的需求往往和真实的用户需求有偏差。所以，Graham推崇那种有用户参与反馈的迭代优化的方式。

无独有偶，最近至少听到两个报告提到了Facebook的开发模式。当Facebook开发一个新的服务，会先让一个小用户群使用，根据用户的反馈来修改功能，同时可以调试程序中的bug。然后下一版让更大一些的用户群使用，收集用户反馈继续修改程序。如此反馈几次，最后再推向所有用户。这种模式要求再最初设计时尽量简单，从而只需几个月的时间就能推出一个新的功能，然后再不断地优化完善。

到目前为止，谈的工业界偏多一些，但其实在系统领域的学术研究，“简单”法则同样适用。

李凯教授：KISS原则

美国工程院院士、普林斯顿大学计算机系李凯教授是“KISS”原则的坚决贯彻者。几乎每次和李凯老师讨论，他都会强调“Keep it Simple”。李凯老师的做事方式是——只抓住大方向，其他问题尽量简化。

但真正要做到KISS原则其实并不容易。我在遇到问题时，往往会从各个方面去考虑问题，其中难免包含了各种细枝末节，这种方式导致问题经常会变得非常复杂。之前讲过这个例子，在移植TCP/IP协议栈到用户态时，我觉得有约10个功能需要考虑。和李老师讨论，他让我把那些功能分成两类：“必须有(Must Have)”和“可以有(Nice-to-Have)”。当我试了这种方法，发现原来Must-Have的功能其实也不过2~3个而已。而最近的例子则是在要设计一个功能让TCP/IP连接的Server在模拟器中、Client在真实机器。我考虑是尽量减少模拟器上OS的开销，所以打算采用自己写一个设备、然后让用户态程序Bypass Kernel直接访问该设备的方

案。但李凯老师在了解OS开销以后，认为容忍开销、尽量直接使用模拟器自己带的功能，让开发更简单。

这些教训也让我不断地去思考为什么要用KISS原则。慢慢地我体会到，***KISS原则目的其实是——“快速推进、逐步优化”***。我们设计一个算法，往往可以在大脑中预先思考好，然后直接编程写出来。但是，我们设计实现一个系统，当系统的复杂度超出我们大脑的工作记忆容量时，就无法在大脑中去“模拟”每一个细节。此时，我们应该用最快的速度去把系统建起了，然后再对各个环节进行优化。

这个KISS理念并不是计算机系统领域特有的，最早是来源于研制飞机时提出的设计理念。而在其他领域，如果一个任务涉及多个步骤，也同样有效，比如生物研究。我去年前曾看过施一公教授写的一篇文章中也提到了这一点。

施一公教授："耗费时间的完美主义阻碍创新进取 "【5】

2011年9月清华大学施一公教授在科学网上发布了一篇博客《如何做一名优秀的博士生：（二）方法论的转变》，其中介绍到完美主义的危害，其实是从另一个角度来强调“简单”。

施教授讲了他博士后期间的一个故事。一次他的任务是纯化一个蛋白。两天下来，虽然纯化了，但是产量只有20%。
他不好意思地对导师说，“产率很低，我计划继续优化蛋白的纯化方法，提高产率”。
但导师反问：“你为什么想提高产率？已有的蛋白不够你做初步的结晶实验吗？”
他回敬：“我有足够的蛋白做结晶筛选，但我需要优化产率以得到更多的蛋白。”
导师不客气地打断：“不对。产率够高了，你的时间比产率重要。请尽快开始结晶。”
实践最后证明他导师的建议是对的。

对于这个故事，施一公教授总结如下：

> "在大刀阔斧进行创新实验的初期阶段，对每一步实验的设计当然要尽量仔细，但一旦按计划开始后对其中间步骤的实验结果不必追求完美，而是应该义无反顾地把实验一步步推到终点，看看可否得到大致与假设相符的总体结果。如果大体上相符，你才应该回过头去仔细地再改进每一步的实验设计。如果大体不符，而总体实验设计和操作都没有错误，那你的假设（或总体方向）很可能是有大问题的。
>
> ……
>
> 从1998年开始自己的独立实验室到现在，我告诉所有学生：切忌一味追求完美主义。我把这个方法论推到极限：只要一个实验还能往前走、一定要做到终点、尽量看到每一步的结果、之后需要时再回头看、逐一解决中间遇到的问题。 "

结语

我想从各个角度去阐释“简单之美”，但到最后感觉这篇文章就是一个大杂烩。既然如此，那我就再加一点料。

Elon Musk是现实世界中的钢铁侠，他先后创办了网络支付公司PayPal、电动汽车公司Tesla以及空间探索公司Space X。目前Space X研制的“猎鹰”火箭已成功试飞，并得到NASA 16亿美元的合同。为了减低成本和提供可靠性，Space X设计的火箭也到处渗透着“简单”【6】：“在他们的猎鹰1号运载火箭上，并没有很多专利，科学家们不在乎，只要火箭能飞就行。火箭用的主发动机也不是21世纪的最新设计，而是1960年代的老古董，只有一个燃料喷射器。它很老，但很可靠。”

参考资料

【1】“The Rise of 'Worse is Better'”，http://www.jwz.org/doc/worse-is-better.html
【2】“"差点的更好"设计理念的兴起”，http://www.aqee.net/the-rise-of-worse-is-better/
【3】“Scalability Lessens from Google, YouTube, Twitter, Amazon, eBay, Facebook and Instagram”
http://www.dodgycoder.net/2012/04/scalability-lessons-from-google-youtube.html
【4】“Achieving Rapid Response Times in Large Online Services”, Jeff Dean, Google.
http://research.google.com/people/jeff/latency.html
【5】“如何做一名优秀的博士生：（二）方法论的转变”，施一公，科学网博客
http://blog.sciencenet.cn/home.php?mod=space&uid=46212&do=blog&id=486270
【6】“硅谷企业家开设私人火箭工厂 目标直指火星”，新浪科技，http://tech.sina.com.cn/d/2012-02-09/16086703213.shtml

上一篇：美国工程院院士李凯教授侧记
下一篇：计算机历史博物馆之旅

评论 (17 个评论) 该博文允许注册用户评论 请点击登录

[17]zhucele 2014-12-24 23:03 | 赞

包先生真知灼见，思想犀利，洞穿迷雾。佩服。

[16]郭红星 2012-9-3 10:14 | 赞

先简单以立，立然后完美。好！

[15]刘进平 2012-5-2 05:17 | 赞

good

[14]aiisf 2012-4-23 22:30 | 赞 赞

最近就在研究系统设计

[13]yl87165501 2012-4-23 18:26 | 赞 赞

写得很好，很受启发

[12]齐占会 2012-4-23 17:46 | 赞 赞

顶！这篇文章写的非常好，非常好！学习到了很多

[11]asirsimon 2012-4-23 17:29 | 赞 赞

追求简单，这个和“奥卡姆剃刀”所提倡的一致。

[10]hfgz 2012-4-23 15:54 | 赞 赞

实验室里研发阶段可以，商业交付和量产阶段不行。

[9]丛远新 2012-4-23 14:31 | 赞 赞

新书理念分享——系统设计金则：简约！
http://blog.sciencenet.cn/home.php?mod=space&uid=324673&do=blog&quickforward=1&id=562679
嘀嘀嘀

[8]everybodycome 2012-4-23 14:01 | 赞 赞

可以发散思维

[7]李毅伟 2012-4-23 13:51 | 赞

赞

施所说的有一定道理，但应具体问题具体分析。

[6]谢蓝 2012-4-23 13:24 | 赞

Simple is best, but simple is not easy.

[5]罗淼 2012-4-23 13:16 | 赞

好文章！

[4]白图格吉扎布 2012-4-23 12:36 | 赞

两点之间直线最短。

[3]lihx1798 2012-4-23 12:23 | 赞

"在大刀阔斧进行创新实验的初期阶段，对每一步实验的设计当然要尽量仔细，但一旦按计划开始后对其中间步骤的实验结果不必追求完美，而是应该义无反顾地把实验一步步推到终点，看看可否得到大致与假设相符的总体结果。如果大体上相符，你才应该回过头去仔细地再改进每一步的实验设计。如果大体不符，而总体实验设计和操作都没有错误，那你的假设（或总体方向）很可能是有大问题的。"

[2]侯德鑫 2012-4-23 12:23 | 赞

不错。
不过为何我这边看到的博文最后几段格式很乱？相邻两行有重叠.....

[1]linsushing 2012-4-23 11:58 | 赞

该博文很有启发。谢谢！我正在做一个试验，用一个顶级微生物群落（菌种）加入到小曲白酒酿造中，只要其结果是还能具有一定出酒率。其方向将是正确的。

1/1 | 总计:17 | 首页 | 上一页 | 下一页 | 末页 | 跳转

返回顶部