“系统架构部”周刊2020年01期

1. 发刊词

系统架构部“2019年总结会”认为，技术分享，除了培训，我们还可以来个“周刊”。周刊中，记录一些我们平时的技术交流，思想碰撞，产品发布，经验分享等。于是我们决定，参考科技爱好者周刊的形式，从2020年开始，每周发布一期“系统架构部”周刊。

周刊的材料供给，除了本部门的同事会积极参与之外，也欢迎各位同仁躬身入局。请在Issue中提交内容,可以包括：

1. 一些对工作有益的“句子”
2. 一种不错的“方法论”
3. 比较不错的小工具介绍
4. 可能是一种典型问题的分析排查过程
5. 一段不错的shell脚本
6. 一些思想的碰撞
7. “从基础开始，让你飞”系列
8. 一个不错的JAVA/GO项目，给点介绍，再“有图有真相”一把
9. 一种自己尝试过的不错的最佳实践
10. 等等等

记录每周值得分享的内容，周五发布。欢迎投稿，推荐或自荐文章/软件/资源，请提交 issue，或者联系主编黄老邪(微信号bingoohuang) 。

2. 2020年，程序员本命年，加油！

2020年注定是一个不平凡的一年。2020年是公历闰年，2月份是29天，一年是366天。

- 2020 = 1024 + 996。
- 2020 = 404 + 404 + 404 + 404 + 404。

1024是什么梗？在数学上，1024是2的10次方，$1024=2^{10}$。在计算机中，1GB=1024MB，1MB=1024KB，1KB=1024Byte。因此1024多指互联网和科技公司，经常表示程序员，另外还表示一级棒的意思（1GB）。

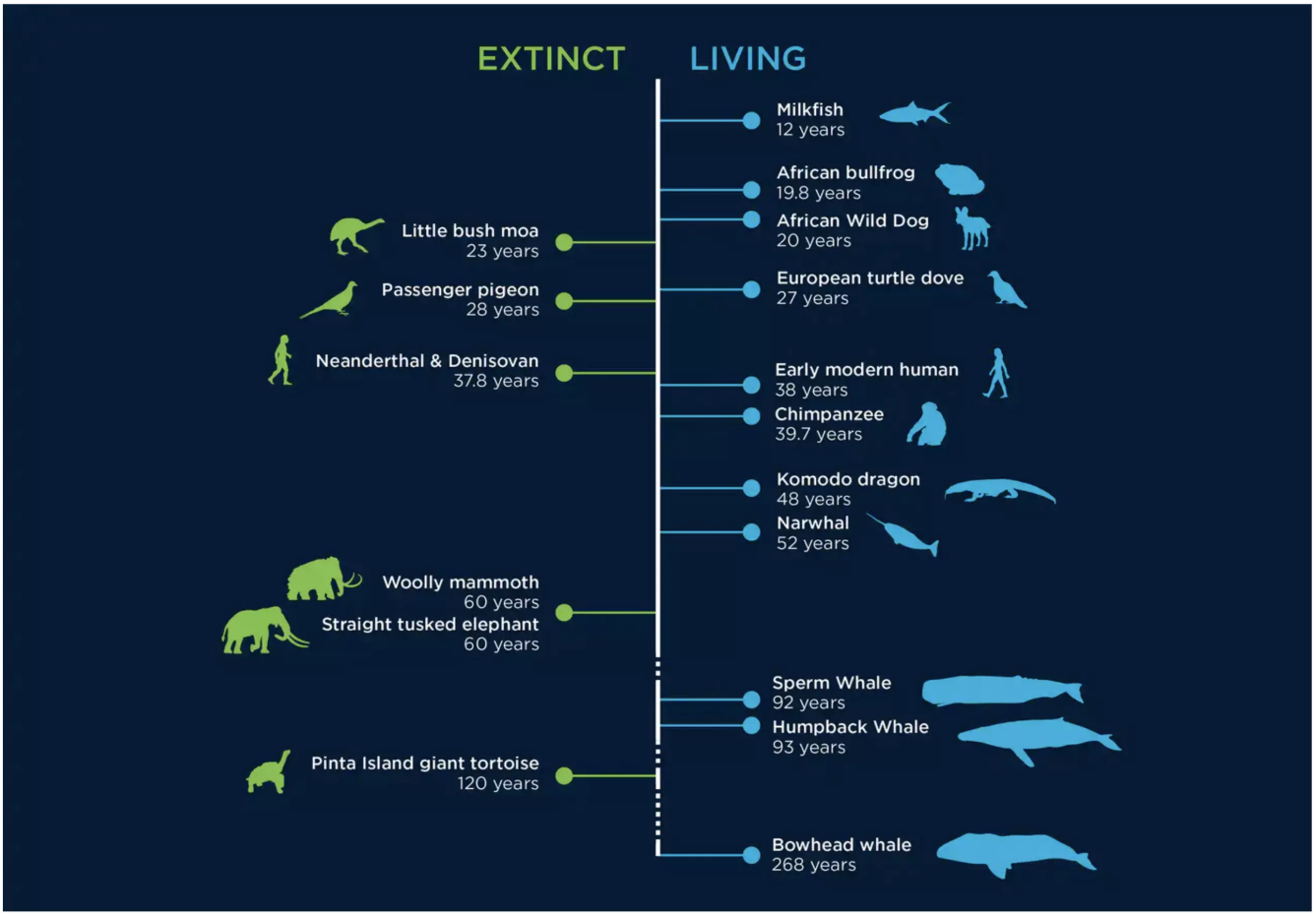
996是个工作制，表示工作时间从早上9点到晚上9点，每周工作6天。也就是说，每周要工作60个小时。很多互联网公司都有一个潜规则，那就是996工作制，因此程序员之间就有一种说法是，工作996， 生病ICU，颇为辛酸却无奈。

2020 = 1024 + 996，冥冥之中竟然暗合程序员本命年之数！想到这里，眼前一亮变成了眼前一黑，这是天意吗？

我命由我不由天，无论如何都要爱惜身体，少熬夜，多锻炼。

2020年，程序员本命年，加油！

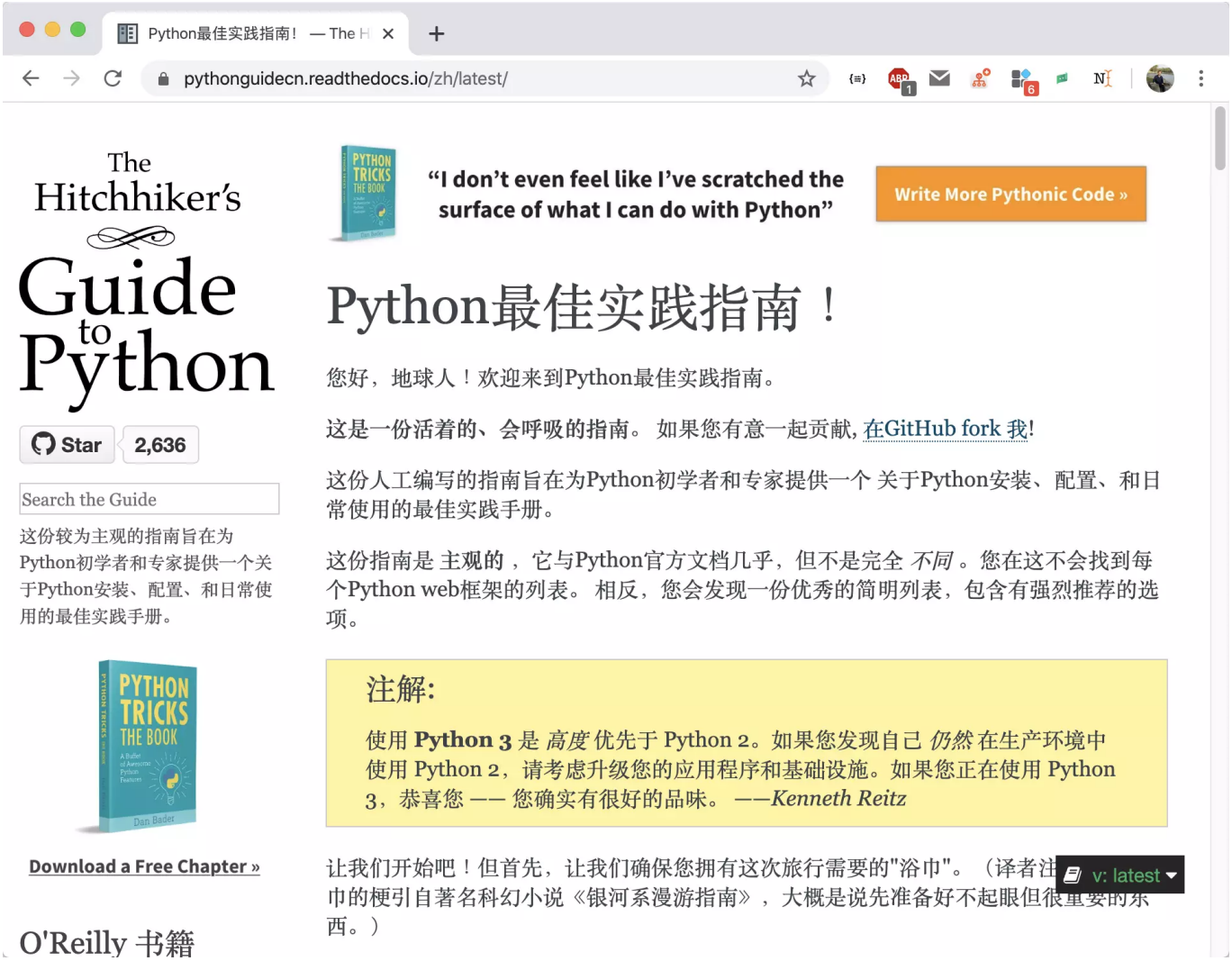
3. 人类的自然寿命是38岁



Using DNA analysis, scientists can now estimate the lifespans of long-lived and extinct species. 通过 DNA 分析，科学家现在可以估计出长寿物种和灭绝物种的寿命 CSIRO 联邦科学与工业研究组织, Author provided 作者提供

澳大利亚科学家在《自然》杂志发表论文，认为人类的自然寿命是38岁。所谓自然寿命，就是在野生状态下人类的平均寿命。目前，人均寿命的提高完全是生活水平和医疗条件改善导致的。 研究人员发现，基因的 DNA 有42处与哺乳动物的最大寿命有关。他们一共研究了252种脊椎动物的基因组，发现最长寿的哺乳动物是弓头鲸，寿命为268岁。巨型海龟的寿命为120岁，非洲象为65岁。

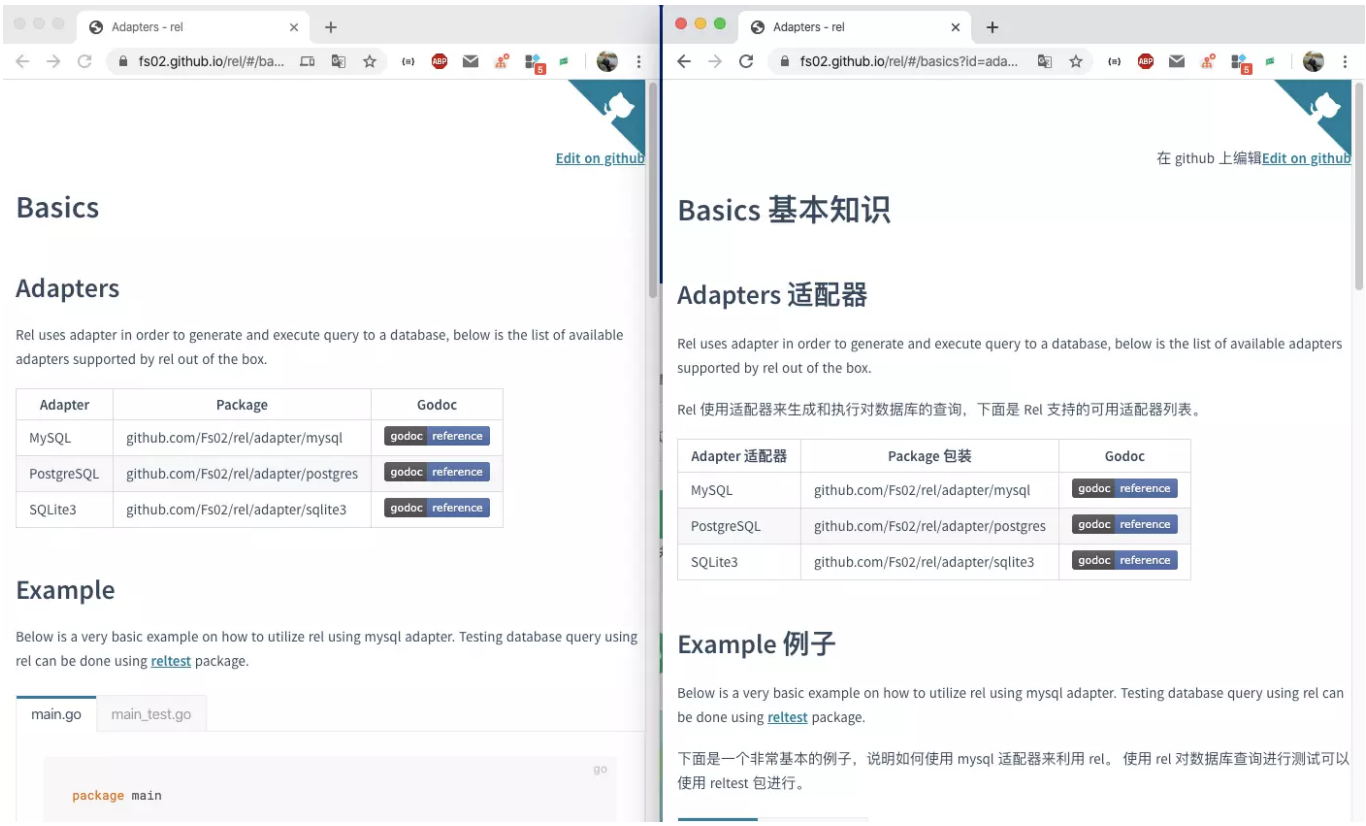
4. 读书《Python最佳实践指南（中文版）》



读书有个方法，先10分钟“快速刷一遍”，此时追求葫芦吞枣，不求甚解。回过头，再对感兴趣的章节细嚼慢咽。因为一般书的精华部分不会超过20%（8020原则）。

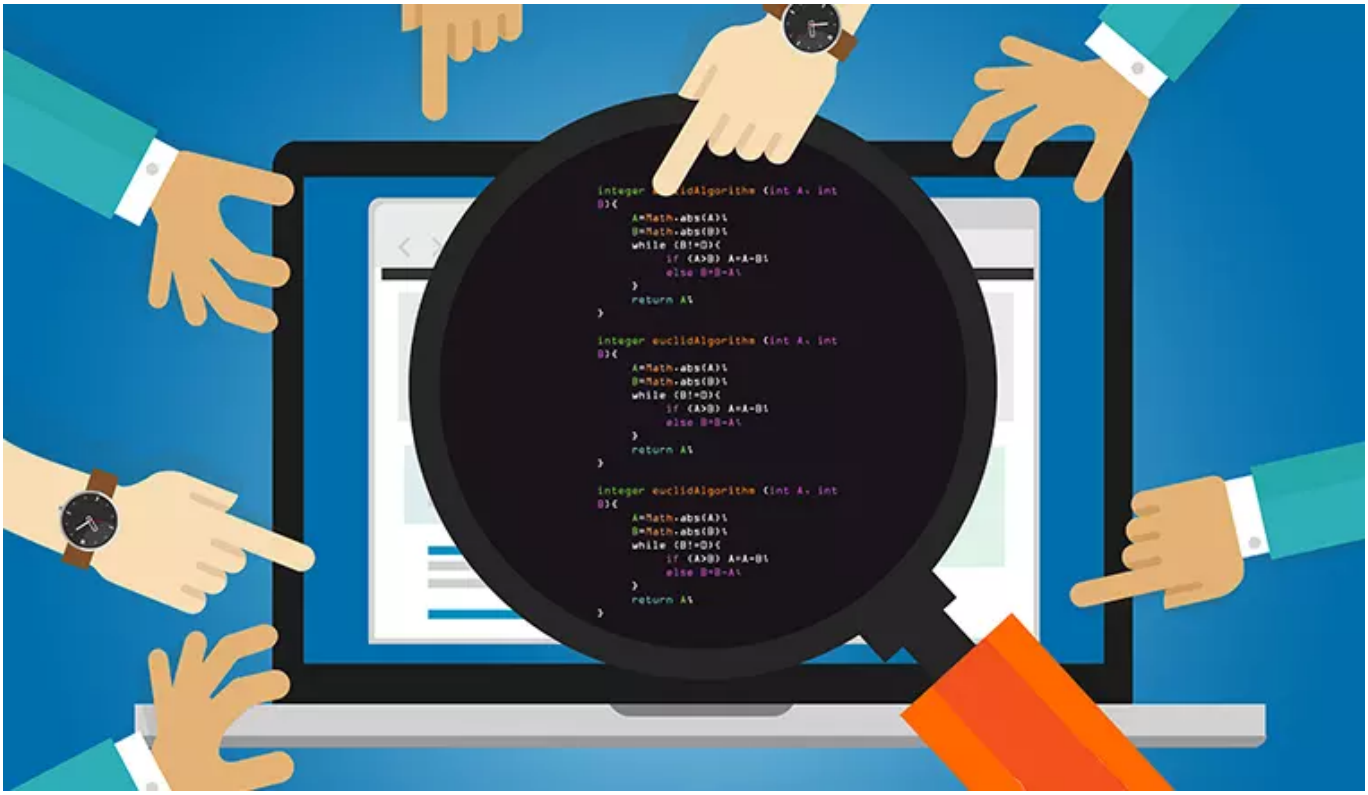
5. 谷歌浏览器插件推荐-彩云小译，妈妈从此不用担心我英文没学好了

下图左边是原始网页，右边是彩云小译“火力全开”后的网页：



注：插件网址，请科学上网

6. 来自谷歌的代码回顾最佳实践中的评审者指南



Code Review建议翻译成“代码回顾”，而不是“代码审查”。

代码回顾的主要目的是始终保证随着时间的推移，代码越来越健康，所有代码回顾的工具和流程也是针对于此设计的。

高级原则: 通常而言，只要代码对系统有明显的提升且正常工作，即便不完美，评审者也应该倾向于通过这次变更。

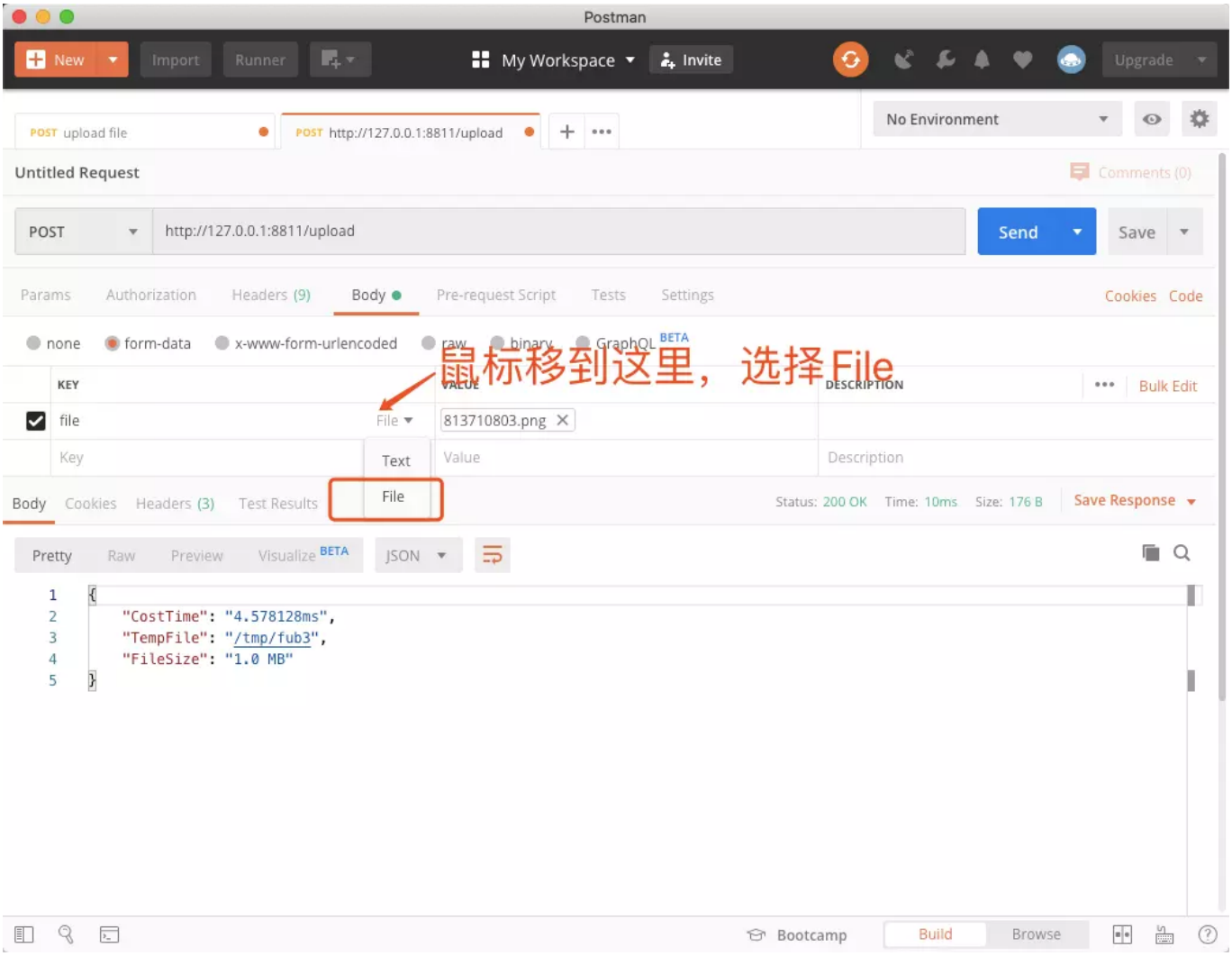
没有完美无缺的代码，只有越来越好的代码。代码评审者绝不应该要求开发者打磨好CL中的每个细节才予以通过，相反，评审者应该权衡项目进度和他们给出建议的重要性，适当放宽要求。评审者应该追求 持续提高 ，而不是追求完美。那些可以提升整个系统可维护性、可读性和可以理解性的变更不应该因为代码不够完美而被推迟几天甚至几周。

指导性: 代码回顾有个重要的作用，那就是可以教会开发者关于语言、框架或者通用软件设计原理。

解决代码冲突: 如果Code Review中有任何冲突，开发人员和评审人员都应该首先根据开发者指南和评审者指南中其他文档的内容，尝试达成一致意见。当很难达成一致时，开发者和评审者不应该在Code Review评论里解决冲突，而是应该召开面对面会议或者找个权威的人来协商。（如果你在评论里协商，确保在评论里记录了讨论结果，以便日后其他人翻阅。）如果这样都解决不了问题，那解决问题的方式就应该升级了。通常的方式是拉着团队一起讨论、让团队主管来权衡、参考代码维护者的意见，或者让管理层来决定。

7. POSTMAN怎么测试文件上传（来自同事潘柯宇的问题）

需要鼠标移动到相应位置才浮现，否则找不到哦。



8. 为啥某台MySQL写入慢了20倍？（来自同事万凯星的问题）

那天，小万同学来问我，为啥两台机器上的MySQL数据库速度不一样呢，都是写入5000条数据，23号机器只需要1秒不到，251号却需要30秒。使用bssh结合pump工具，实测如下：

```
$ bssh -H l23 -H l251 "./pump -d='127.0.0.1:9633 root/8BE4' -b10 -r5000 -t
aaabbb.wtest"
Select Server :l23,l251
Run Command   :./pump -d='127.0.0.1:9633 root/8BE4' -b10 -r5000 -t
aaabbb.wtest
l23  ::  time="2019-12-26T15:47:26+08:00" level=info
msg="ds:8BE4@tcp(127.0.0.1:9633)/?
charset=utf8mb4&parseTime=true&loc=Local"
l23  ::  time="2019-12-26T15:47:26+08:00" level=info msg="batchSQL:insert
into aaabbb.wtest(ip) values(?),(?),(?),(?),(?),(?),(?),(?),(?),(?)"
         aaabbb.wtest 5000/5000 639.07977ms
[============================================================================]
100%

l251 ::  time="2019-12-26T16:03:06+08:00" level=info
msg="ds:8BE4@tcp(127.0.0.1:9633)/?
charset=utf8mb4&parseTime=true&loc=Local"
l251 ::  time="2019-12-26T16:03:06+08:00" level=info msg="batchSQL:insert
into aaabbb.wtest(ip) values(?),(?),(?),(?),(?),(?),(?),(?),(?),(?)"
         aaabbb.wtest 5000/5000 30.99761299s
[============================================================================]
100%
```

```
CREATE TABLE `wtest` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `ip` varchar(255) DEFAULT NULL,
  UNIQUE KEY `测试唯一索引` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

排除了数据库配置参数的原因之外，怀疑是硬盘的差别，硬盘写入测试了一下，使用Linux中用dd命令来测试硬盘读写速度，果然不一样。

> 测试磁盘写能力 `time dd if=/dev/zero of=/tmp/testw.dbf bs=4k count=100000` 因为/dev/zero是一个伪设备，它只产生空字符流，对它不会产生IO，所以，IO都会集中在of文件中，of文件只用于写，所以这个命令相当于测试磁盘的写能力。命令结尾添加oflag=direct将跳过内存缓存，添加oflag=sync将跳过hdd缓存。

```
$ bssh -H l23 -H l251 "time dd if=/dev/zero of=/tmp/testw.dbf bs=4k
count=100000"
Select Server :l23,l251
Run Command   :time dd if=/dev/zero of=/tmp/testw.dbf bs=4k count=100000
l251 ::  记录了100000+0 的读入
l251 ::  记录了100000+0 的写出
l251 ::  409600000字节(410 MB)已复制，1.86862 秒，219 MB/秒
l251 ::
l251 ::  real   0m3.434s
l251 ::  user   0m0.036s
```

```
l251 ::  sys    0m0.458s
l23  ::  记录了100000+0 的读入
l23  ::  记录了100000+0 的写出
l23  ::  409600000字节(410 MB)已复制，0.448683 秒，913 MB/秒
l23  ::
l23  ::  real   0m0.496s
l23  ::  user   0m0.036s
l23  ::  sys    0m0.367s
```

实测结果是，251比23写入速度慢了3倍啊。

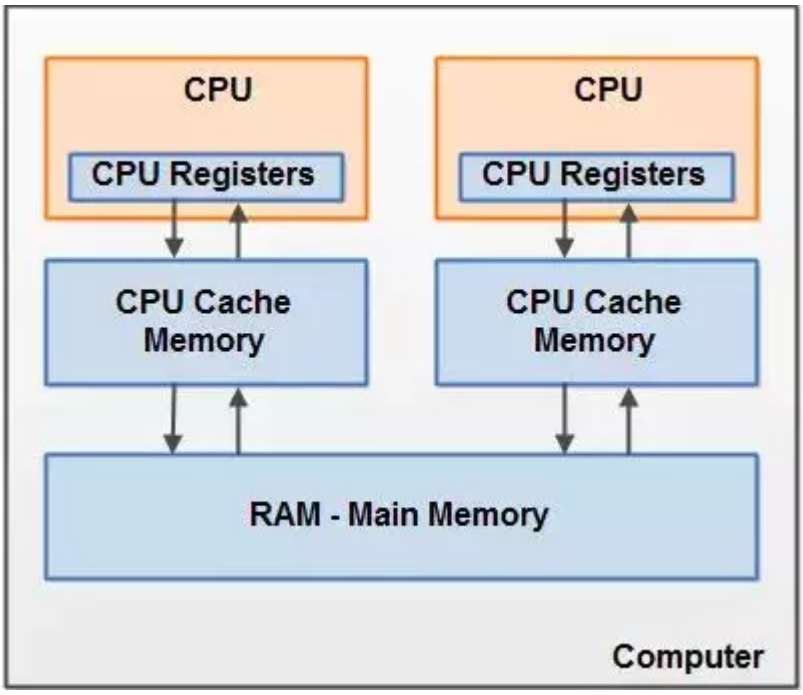
调查下来，*原来251就1块硬盘没做raid，23是2块硬盘做了raid，所以硬盘写入速度，最终导致了数据库的写入速度啊*。

9. 压轴连载 《舒威讲：JAVA多线程》之“何谓线程”?

图片来源12

线程（英语：thread）是操作系统能够进行运算调度的最小单位。大部分情况下，它被包含在进程之中，是进程中的实际运作单位。一条线程指的是进程中一个单一顺序的控制流，一个进程中可以并发多个线程，每条线程并行执行不同的任务。在Unix System V及SunOS中也被称为轻量进程（lightweight processes），但轻量进程更多指内核线程（kernel thread），而把用户线程（user thread）称为线程。

The Java memory model(JMM) describes how threads in the Java programming language interact through memory. Together with the description of single-threaded execution of code, the memory model provides the semantics of the Java programming language.



- 多CPU：一个现代计算机通常由两个或者多个CPU。其中一些CPU还有多核。从这一点可以看出，在一个有两个或者多个CPU的现代计算机上同时运行多个线程是可能的。每个CPU在某一时刻运行一个线程是没有问题的。这意味着，如果你的Java程序是多线程的，在你的Java程序中每个CPU上一个线程可能同时（并发）执行。
- CPU寄存器：每个CPU都包含一系列的寄存器，它们是CPU内内存的基础。CPU在寄存器上执行操作的速度远大于在主存上执行的速度。这是因为CPU访问寄存器的速度远大于主存。
- 高速缓存cache：由于计算机的存储设备与处理器的运算速度之间有着几个数量级的差距，所以现代计算机系统都不得不加入一层读写速度尽可能接近处理器运算速度的高速缓存（Cache）来作为内存与处理器之间的缓冲：将运算需要使用到的数据复制到缓存中，让运算能快速进行，当运算结束后再从缓存同步回内存之中，这样处理器就无须等待缓慢的内存读写了。CPU访问缓存层的速度快于访问主存的速度，但通常比访问内部寄存器的速度还要慢一点。每个CPU可能有一个CPU缓存层，一些CPU还有多层缓存。在某一时刻，一个或者多个缓存行（cache lines）可能被读到缓存，一个或者多个缓存行可能再被刷新回主存。
- 内存：一个计算机还包含一个主存。所有的CPU都可以访问主存。主存通常比CPU中的缓存大得多。
- 运作原理：通常情况下，当一个CPU需要读取主存时，它会将主存的部分读到CPU缓存中。它甚至可能将缓存中的部分内容读到它的内部寄存器中，然后在寄存器中执行操作。当CPU需要将结果写回到主存中去时，它会将内部寄存器的值刷新到缓存中，然后在某个时间点将值刷新回主存。
- 指令重排序问题：为了使得处理器内部的运算单元能尽量被充分利用，处理器可能会对输入代码进行乱序执行（Out-Of-Order Execution）优化，处理器会在计算之后将乱序执行的结果重组，保证该结果与顺序执行的结果是一致的，但并不保证程序中各个语句计算的先后顺序与输入代码中的顺序一致。因此，如果存在一个计算任务依赖另一个计算任务的中间结果，那么其顺序性并不能靠代码的先后顺序来保证。与处理器的乱序执行优化类似，Java虚拟机的即时编译器中也有类似的指令重排序（Instruction Reorder）优化