性能测试

--性能测试过程

内容回顾

- 计划测试—定义测试目标
  - 通过用户数据进行分析
    - 用户数据收集（服务器端开启日志收集功能）
    - 日志分析：使用分析工具（如：WLExpertSetup）
  - 分析日志得到：
    - 用户量使用变化规律
    - 峰值数据出现在什么时间
    - 从数据分析中得到性能测试场景

内容回顾

- 分析关键性能指标：平均并发、平均带宽等
- 根据日志数据分析，得到新场景，运行后，继续对比结果，确认设计的场景是否合理，得到的数据是否符合预期，如果不符合，分析系统瓶颈在哪

内容回顾

- 通过同类业务进行分析：找出性能指标参考数据
- 通过80/20原则进行分析：找出主要功能点，确定被测功能点

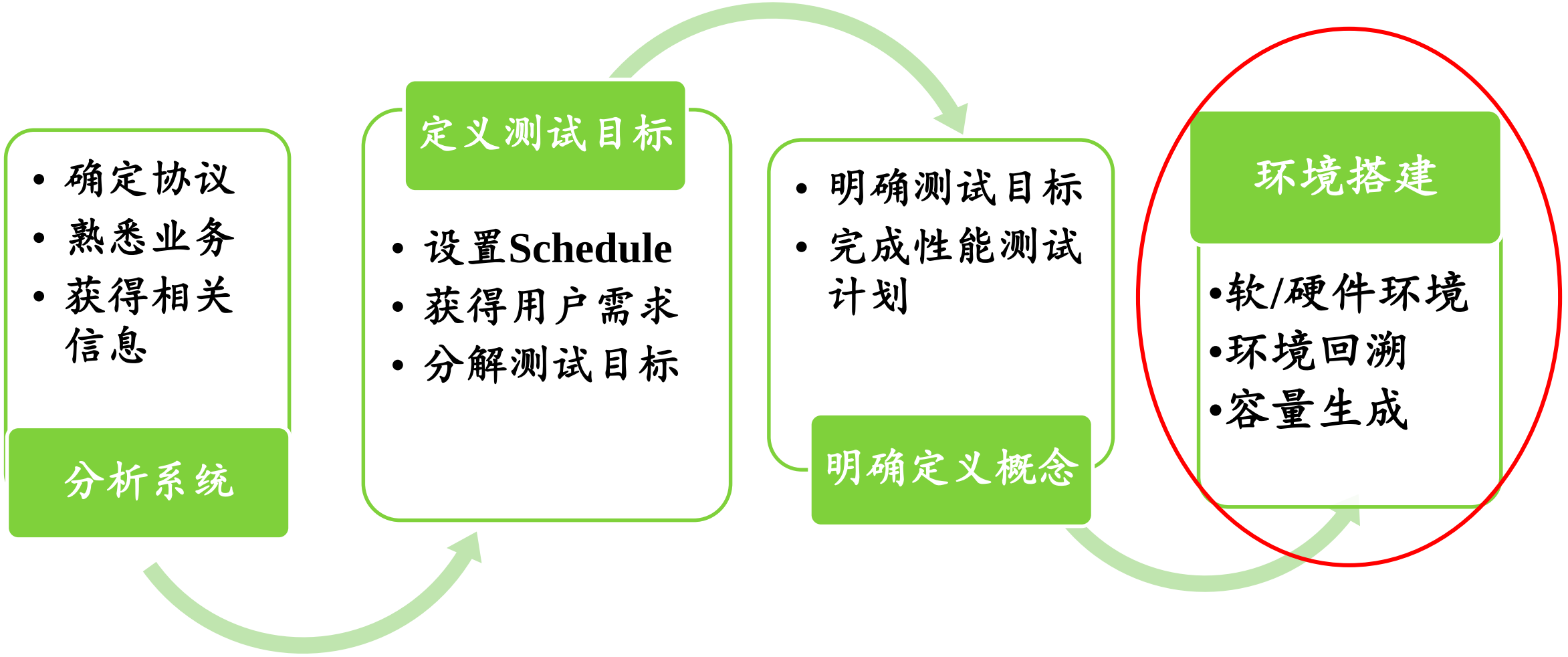
■ 编写性能测试计划

- 文档目的、项目背景、术语、参考文档、系统运行环境、测试内容、非测试内容、角色和职责、性能测试工具、进度安排、出口标准、交付物、风险

内容回顾

- 编写性能测试方案
  - 文档目的、测试目的、测试策略、业务抽取（测试脚本）、需要进行测试的主要业务、用户行为模型及性能指标（测试场景）、监控方式、场景检查
- 编写性能测试用例
  - 与功能测试用例相比，增加事务、检查点、集合点、关联等的说明

计划测试流程


分析系统
• 确定协议
• 熟悉业务
• 获得相关信息
定义测试目标
• 设置Schedule
• 获得用户需求
• 分解测试目标
明确定义概念
• 明确测试目标
• 完成性能测试计划
环境搭建
•软/硬件环境
•环境回溯
•容量生成

目录

搭建测试环境

- 需要搭建哪些环境
  - 硬件环境
  - 软件环境
- 测试平台评估
  - 评估硬件配置
    - CPU、内存、硬盘、带宽等分别需要什么规格的

搭建测试环境

- 通过集群方式进行计算
  - 对于海量数据请求，需要进行负载均衡
  - 对集群上的一个节点进行性能测试，得出该节点的处理能力
  - 计算每增加一个节点的性能指标情况

搭建测试环境

- 如在单台服务器上获得具体的性能指标，每台服务器能够承受2000人在线、平均TPS为80、响应时间为2秒。添加负载均衡策略，测试负载均衡下的数据

负载服务器个数	平均每台负载服务器在线用户数	平均TPS	响应时间/s
1	2 000	80	2
1 添加负载均衡策略	1 950	78	2
2 使用负载均衡	1 950	78	2
3 使用负载均衡	1 900	76	2
4 使用负载均衡	1 900	75	2

搭建测试环境

- 软件环境
  - 为了保证测试结果数据的客观公正，在每次测试前，都需要保证相同的软件环境，所以测试环境备份非常重要
  - 可以使用虚拟机自带的Snapshot快照功能来实现测试备份和还原
  - 形成测试环境搭建手册

搭建测试环境—搭建手册

- 虚拟机设置
- 安装介质说明
- 安装步骤
- 快照制作规范及基础配置
- 数据生成规则
- 最优对比平台
- 基准测试数据

数据生成

- 如果被测系统已有数据500万条帖子，3万个注册用户，与新搭建的系统，里面只有3、5条帖子，1个注册用户，运行结果是否一致？
  - 一定不一致
  - 怎样生成测试数据
    - 数据库中写执行过程
    - 使用DataFactory快速生成

目录

脚本业务报告

VuGen Script Business Processes

Author: Cassie

Script name: noname1

Date: 2019 年 4 月 1 日

Comments:

目录

性能测试过程实践

1 搭建Apache + Mysql + PHP环境

2 将Discuz源文件放在WWW目录下

3 在浏览器中输入localhost:端口号/discuz，根据提示进行安装

4 安装完成后，设置管理员密码（备注：在pre_ucenter_members（ultrax）表中）

性能测试过程实践

5 手工使用系统，简要写出搭建环境记录

6 写出简要测试方案：至少包含测试策略、业务抽取、用户行为模型及性能指标（测试场景）

7 写出简要测试用例

8 执行并监控测试过程

9 保存每种场景生成的测试结果（Analysis结果）

测试场景监控

场景checklist

检查项编号	检查项目	检查人	备注
1	场景类型		
2	场景脚本		
3	场景运行设计		
4	运行设置		
5	集合点策略		
6	负载生成器		
7	SLA策略		
8	系统监控及计数器管理		
9	运行结果		
10	环境备份		

测试场景监控

序号	执行时间	测试项目	被测服务器IP	场景结果文件	测试项目数据	备注
1	2019/3/15 14:00	发帖	192.168.1.5	res_smessage	实际发帖人数500	
2	2019/3/15 15:00	查询	192.168.1.6	res_search	查询帖子数50000	
3	2019/3/15 16:00	登录	192.168.1.7	res_login	实际登录人数1000	
4	……					
5						
6						

目录

性能分析

- 场景运行结束后，通过Analysis组件可以得到对应的性能测试报告，但是光靠这个报告不能反映真实问题
  - 还需要人为对这些数据进行分析
  - 便于进行瓶颈定位和下一步优化
- 对于整个软件系统来说，最终能反映到单体用户的响应时间上，响应时间是由哪些内容组成的呢？

前端性能分析与调优—生活中的例子

- 思考生活中的例子：张三每天早上乘交通工具上班要花费多长时间？这个时间组成有哪些？如何调优？
  - 分析：从家到公司的时间由哪些部分组成
    - 从家到交通工具所在站点
    - 从交通工具到达目的地
    - 从目的地进入公司
    - 张三上班经常迟到，想改变上班策略，做到不迟到（生活中性能调优）

前端性能分析与调优—生活中的例子

- 假设调优前7:30出门，走15分钟到达公交站，一般等待5分钟上公交（偶尔挤不上去），在正常情况下会在7:50坐上公交，然后坐50分钟公交，在8:40下公交，走路8分钟到公司楼下，等5分钟电梯，在8:53到达公司，通常情况下在上班路上会消耗83分钟

前端性能分析与调优—生活中的例子

- 调优方法1：对所有内容进行分析，确定哪些可以优化，哪些不能优化

几乎不能优化的内容	
公交车50分钟 等车5分钟	无法优化
比较难优化的内容	
等电梯5分钟	如果楼层低可以走上去，6楼以下，只需3分钟
可以优化的部分	
出门走到公交站15分钟 下公交车走到公司楼下8分钟	如果这两个时间都由走路改成小跑，那么到公交站可以节约7分钟，下车到公司节约4分钟。所以使用这种方式优化，优化效果节约了11分钟

前端性能分析与调优—生活中的例子

- 调优方法1总结：
  - 通过分析上班途中的时间，确定可以优化的内容，进行优化，最终节约了11分钟
  - 这种方法方便快捷，但效果不一定很好，而且经常不能解决本质问题

前端性能分析与调优—生活中的例子

- 调优方法2：首先分析最值得调优的部分（最浪费时间的部分）

项目	消耗时间
出门到公交站	15分钟
等待公交	5分钟
坐公交	50分钟
下公交到公司	8分钟
坐电梯	5分钟

前端性能分析与调优—生活中的例子

- 有没有替换公交的交通工具
  - 地铁
  - 假设：从坐上公交到地铁站需4分钟，等地铁需要3分钟，坐地铁需要24分钟，从地铁出站走到公司需要8分钟

项目	消耗时间
公交到地铁	4分钟
等待地铁	3分钟
坐地铁	8分钟

  - 交通工具消耗的总时间：31分钟，比乘坐公交减少了19分钟，而且比坐公交更稳定

前端性能分析与调优—生活中的例子

- 调优方法2总结：
  - 调优后成本提高了，因为多坐一次地铁
  - 这次不是找最容易优化的部分下手，而是找最浪费时间的部分下手
  - 这种调优方式相对来说比较困难，需要更多的技术和经验，但优化效果较好

前端性能分析与调优—生活中的例子

- 对比前两种优化结果

项目	调优前（分钟）	调优1（分钟）	调优2（分钟）
出门到公交站	15	8	15
等待公交	5	5	5
坐公交	50	50	31（坐公交+等、坐地铁时间）
下公交到公司	8	4	8
坐电梯	5	5	5
总计	83	72	64

前端性能分析与调优—生活中的例子

- 如果将两种调优方式结合在一起，效果又会如何呢？
  - 调优方法3：

项目	调优前（分钟）	调优后（分钟）
出门到公交站	15	8
等待公交	5	5
坐公交	50	31
下公交到公司	8	4
坐电梯	5	5
总计	83	53

前端性能分析与调优—生活中的例子

- 继续分析：出门到公交站需要8分钟，等待公交需要5分钟，坐公交到地铁需要4分钟，这些项目上能不能继续优化？
  - 如果选择骑车，会怎样

项目	调优前（1+2）	骑车方案
出门到公交站	8	3
等待公交	5	0
公交到地铁	4	5（包含停车时间）

  - 结果：通过这种方式将时间再次优化了9分钟

前端性能分析与调优—生活中的例子

- 最终结果：

项目	调优结果
出门到地铁	8分钟
等待地铁	3分钟
坐地铁	24分钟
下地铁到公司	4分钟
坐电梯	5分钟
总计	44分钟

前端性能分析与调优—生活中的例子

- 除了这种优化策略以外，还有别的思路吗？
  - 早点上班，错开高峰期
  - 换到离公司近的地方居住或交通更方便的地方居住
  - 骑车去上班
  - 换一个离家近的公司

前端性能分析与调优—生活中的例子

- 调优方法3总结：
  - 通过对前面两种调优方式进行调优效果进行综合，并且对每一个部件进行分析，找出之间的联系和互相影响的部分，进行进一步分析和尝试，直到最优结果
- 对于一个软件系统来说，从成本和难易程度来说一般的调优顺序是怎样的呢？

性能分析与调优

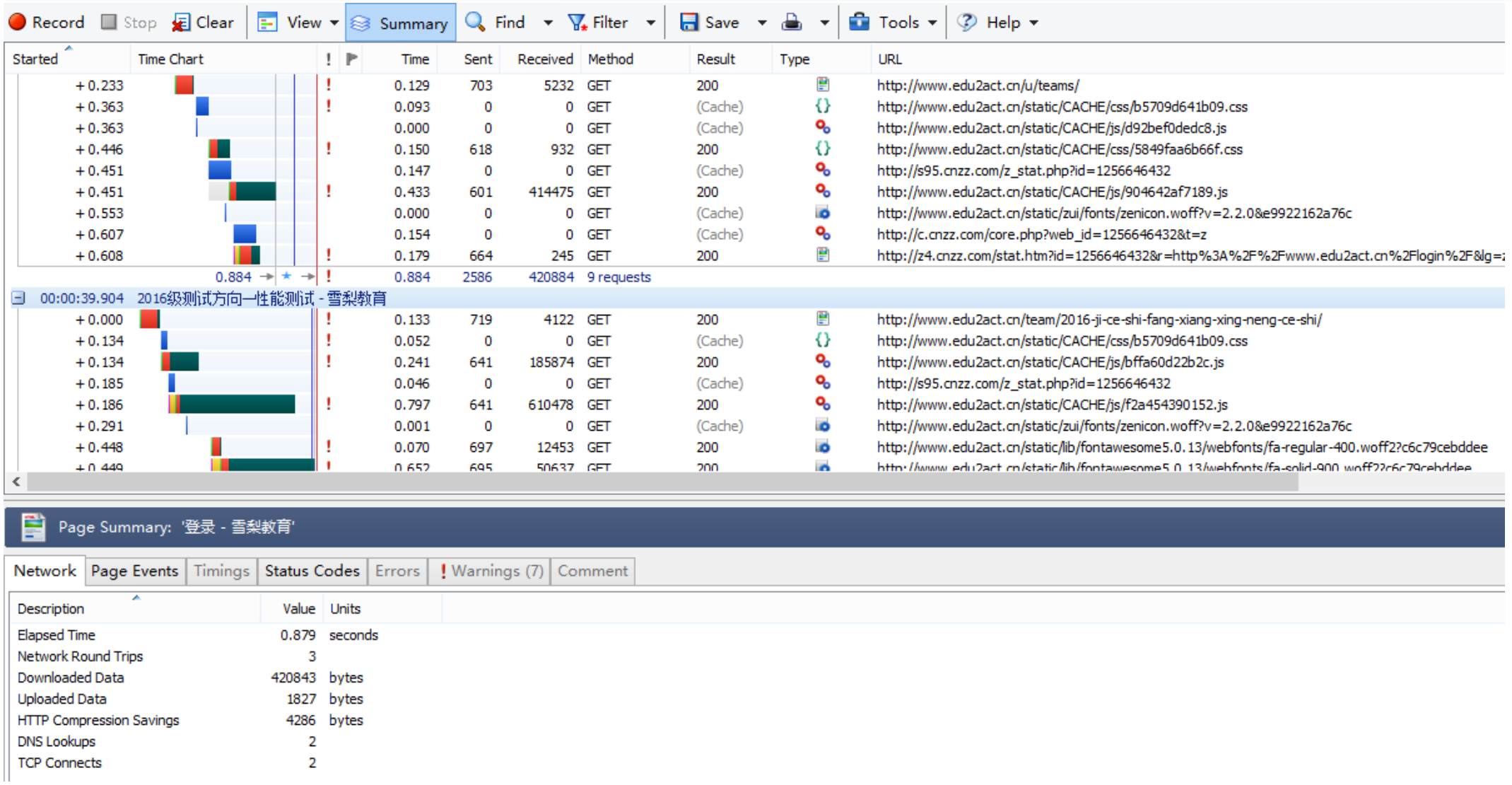
软件系统调优顺序

- 软件平台设置
  - 对软件服务平台的设置，可以更好的利用硬件资源，在不开销任何成本的基础上提升性能指标
- 硬件平台
  - 通过配置测试和基准测试可以很快地确定硬件更新所带来的效果

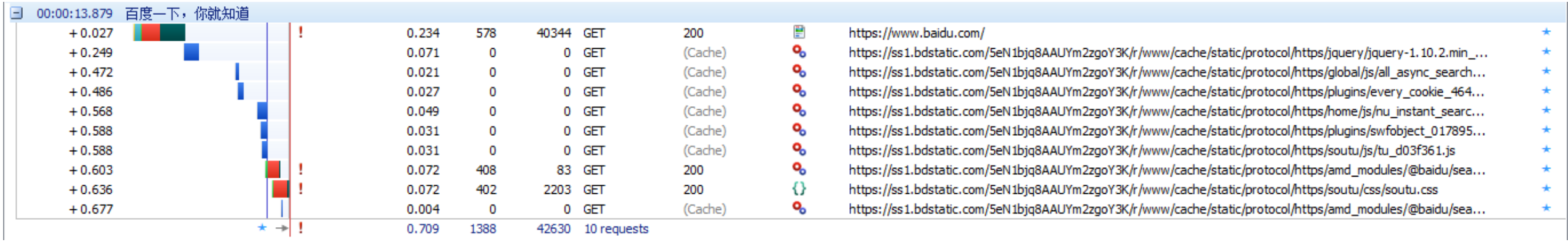
性能分析与调优

- 代码或SQL语句
  - 通过静态分析得到负载较大代码或SQL语句，将其进行修改，降低逻辑复杂度及成本
- 架构或需求
  - 如果上述方法无法完成性能调优，需要考虑对架构进行一定调整
  - 一般在设计初期应该对架构进行科学负载和并发测试，确定架构的正确性

前端性能分析与调优


Record
Stop
Clear
View
Summary
Find
Filter
Save
Tools
Help
Started
Time Chart
Time
Sent
Received
Method
Result
Type
URL
+0.233 0.129 703 5232 GET 200 http://www.edu2act.cn/u/teams/
+0.363 0.093 0 0 GET (Cache) http://www.edu2act.cn/static/CACHE/css/b5709d641b09.css
+0.363 0.000 0 0 GET (Cache) http://www.edu2act.cn/static/CACHE/js/d92bef0dedc8.js
+0.446 0.150 618 932 GET 200 http://www.edu2act.cn/static/CACHE/css/5849faa6b66f.css
+0.451 0.147 0 0 GET (Cache) http://s95.cnzz.com/z_stat.php?id=1256646432
+0.451 0.433 601 414475 GET 200 http://www.edu2act.cn/static/CACHE/js/904642af7189.js
+0.553 0.000 0 0 GET (Cache) http://www.edu2act.cn/static/zui/fonts/zenicon.woff?v=2.2.0&e9922162a76c
+0.607 0.154 0 0 GET (Cache) http://c.cnzz.com/core.php?web_id=1256646432&t=z
+0.608 0.179 664 245 GET 200 http://z4.cnzz.com/stat.htm?id=1256646432&r=http%3A%2F%2Fwww.edu2act.cn%2Flogin%2F&lg=
0.884 0.884 2586 420884 9 requests
00:00:39.904 2016级测试方向一性能测试 - 雪梨教育
+0.000 0.133 719 4122 GET 200 http://www.edu2act.cn/team/2016-ji-ce-shi-fang-xiang-xing-neng-ce-shi/
+0.134 0.052 0 0 GET (Cache) http://www.edu2act.cn/static/CACHE/css/b5709d641b09.css
+0.134 0.241 641 185874 GET 200 http://www.edu2act.cn/static/CACHE/js/bffa60d22b2c.js
+0.185 0.046 0 0 GET (Cache) http://s95.cnzz.com/z_stat.php?id=1256646432
+0.186 0.797 641 610478 GET 200 http://www.edu2act.cn/static/CACHE/js/f2a454390152.js
+0.291 0.001 0 0 GET (Cache) http://www.edu2act.cn/static/zui/fonts/zenicon.woff?v=2.2.0&e9922162a76c
+0.448 0.070 697 12453 GET 200 http://www.edu2act.cn/static/lib/fontawesome5.0.13/webfonts/fa-regular-400.woff2?c6c79cebddee
Page Summary: '登录 - 雪梨教育'
Network
Page Events
Timings
Status Codes
Errors
Warnings (7)
Comment
Description Value Units
Elapsed Time 0.879 seconds
Network Round Trips 3
Downloaded Data 420843 bytes
Uploaded Data 1827 bytes
HTTP Compression Savings 4286 bytes
DNS Lookups 2
TCP Connects 2

访问百度首页时间分布



00:00:13.879 百度一下，你就知道

+0.027		0.234	578	40344	GET	200		https://www.baidu.com/
+0.249		0.071	0	0	GET	(Cache)		https://ss1.bdstatic.com/5eN1bjq8AAUYm2zgoY3K/r/www/cache/static/protocol/https/jquery/jquery-1.10.2.min_...
+0.472		0.021	0	0	GET	(Cache)		https://ss1.bdstatic.com/5eN1bjq8AAUYm2zgoY3K/r/www/cache/static/protocol/https/global/js/all_async_search...
+0.486		0.027	0	0	GET	(Cache)		https://ss1.bdstatic.com/5eN1bjq8AAUYm2zgoY3K/r/www/cache/static/protocol/https/plugins/every_cookie_464...
+0.568		0.049	0	0	GET	(Cache)		https://ss1.bdstatic.com/5eN1bjq8AAUYm2zgoY3K/r/www/cache/static/protocol/https/home/js/nu_instant_searc...
+0.588		0.031	0	0	GET	(Cache)		https://ss1.bdstatic.com/5eN1bjq8AAUYm2zgoY3K/r/www/cache/static/protocol/https/plugins/swfobject_017895...
+0.588		0.031	0	0	GET	(Cache)		https://ss1.bdstatic.com/5eN1bjq8AAUYm2zgoY3K/r/www/cache/static/protocol/https/soutu/js/tu_d03f361.js
+0.603		0.072	408	83	GET	200		https://ss1.bdstatic.com/5eN1bjq8AAUYm2zgoY3K/r/www/cache/static/protocol/https/amd_modules/@baidu/sea...
+0.636		0.072	402	2203	GET	200		https://ss1.bdstatic.com/5eN1bjq8AAUYm2zgoY3K/r/www/cache/static/protocol/https/soutu/css/soutu.css
+0.677		0.004	0	0	GET	(Cache)		https://ss1.bdstatic.com/5eN1bjq8AAUYm2zgoY3K/r/www/cache/static/protocol/https/amd_modules/@baidu/sea...
		0.709	1388	42630	10 requests			

访问bing首页时间分布

Started	Time Chart	!	Time	Sent	Received	Method	Result	Type	URL	Blocked
00:00:09.821	微软 Bing 搜索 - 国际版									
+ 0.891		!	0.726	874	35829	GET	200		http://cn.bing.com/	
+ 1.767			0.035	0	0	GET	(Cache)		http://www.bing.com/rs/5p/1mH/cj,nj/2213d9b6/b50738ca.js?setmkt=zh-CN	
+ 1.873		!	0.058	1076	209	GET	200		http://cn.bing.com/fd/ls/l?IG=9B2D7A79260443338870D456AD90CFE5&Type=Event.CPT&DATA={%22pp%22:{...	0.001
+ 1.876			0.203	1276	198	POST	204		http://cn.bing.com/fd/ls/lsp.aspx?	0.002
+ 1.963		!	0.004	0	0	GET	(Cache)		http://www.bing.com/rb/5p/cj,nj/6d2f1af3/261929f9.js?bu=EvQekx-8Hr8e3gTNHs8enx_RHtge3R6LH4kf_R7vHf...	
+ 1.979		!	0.228	1012	390	GET	200		http://cn.bing.com/notifications/render?bnptrigger=%7B%22PartnerId%22%3A%22HomePage%22%2C%22II...	0.002
+ 2.080			0.007	0	0	GET	(Cache)		http://www.bing.com/rs/2T/jO/cj,nj/64c6b209/6c8d22b8.js?setmkt=zh-CN	
+ 2.081			0.009	0	0	GET	(Cache)		http://www.bing.com/rs/2T/j2/cj,nj/b2a824f7/b8c6c09d.js?setmkt=zh-CN	
+ 2.082			0.013	0	0	GET	(Cache)		http://www.bing.com/rs/2T/fN/cj,nj/d83a28bc/699c87d7.js?setmkt=zh-CN	
+ 2.085		!	0.163	374	1227	GET	200		http://www.bing.com/rs/30/1H/cj,nj/5983aa50/f8c6dd44.js?setmkt=zh-CN	0.003
+ 2.094			0.184	928	1835	GET	200		http://cn.bing.com/HPImageArchive.aspx?format=hp&idx=0&n=1&nc=1554191344377&pid=hp&quiz=1&og=1&...	0.002
+ 2.100		!	0.251	856	5508	GET	200		http://cn.bing.com/hpm?IID=SERP.1000&IG=9B2D7A79260443338870D456AD90CFE5	0.002
+ 2.147			0.114	374	454	GET	200		http://www.bing.com/rs/30/1X/cj,nj/4c7364c5/40e1b425.js?setmkt=zh-CN	0.061
+ 2.207		!	0.066	399	2920	GET	200		http://www.bing.com/rb/16/cj,nj/1b7dfb88/cc8437ad.js?bu=DikuXGxwdGhgZKwBsAEuoAEu&setmkt=zh-CN	
+ 2.327			0.091	812	1227	GET	200		http://www.bing.com/rs/30/1H/cj,nj/5983aa50/f8c6dd44.js?setmkt=zh-CN	0.002
+ 2.475			0.003	0	0	GET	(Cache)		http://www.bing.com/rs/6n/k9/cj,nj/6240f061/6fb5e8ee.js?setmkt=zh-CN	
+ 2.581			0.051	812	1270	GET	200		http://www.bing.com/rs/30/1X/cj,nj/4c7364c5/40e1b425.js?setmkt=zh-CN	0.005
+ 2.762			0.004	0	0	GET	(Cache)		http://www.bing.com/rs/2T/lq/cj,nj/4761a975/efd13843.js?setmkt=zh-CN	
+ 2.763			0.046	837	8181	GET	200		http://www.bing.com/rb/16/cj,nj/1b7dfb88/cc8437ad.js?bu=DikuXGxwdGhgZKwBsAEuoAEu&setmkt=zh-CN	0.002
+ 2.768		!	0.248	858	3431	GET	200		http://cn.bing.com/hptr/?IG=9B2D7A79260443338870D456AD90CFE5&IID=SERP.1300	0.001
+ 2.833		!	0.090	1019	209	GET	200		http://cn.bing.com/fd/ls/l?IG=9B2D7A79260443338870D456AD90CFE5&Type=Event.PPT&DATA={%22S%22:23...	0.001
+ 3.620			0.094	4751	198	POST	204		http://cn.bing.com/fd/ls/lsp.aspx	0.003
+ 3.975		!	0.088	2681	233	GET	200		http://a4.bing.com/fd/ls/l?IG=9B2D7A79260443338870D456AD90CFE5&TYPE=Event.ClientInst&DATA=%5B%7...	0.002
+ 5.696			0.062	1369	198	POST	204		http://cn.bing.com/fd/ls/lsp.aspx	0.002
	1.827 5.758	!	5.758	20308	63517	24 requests				
00:00:40.333	微软 Bing 搜索 - 国际版									
+ 0.000			0.117	1733	198	POST	204		http://cn.bing.com/fd/ls/lsp.aspx	0.002
	0.117		0.117	1733	198	1 request				

访问首页时间分析

	baidu	Bing
duration	0.709s	0.117s
send	1388字节	1733字节
receive	42630字节	198字节

请求回应时间分析

- 一个请求从发送给服务器到服务器返回给本地，由哪些时间组成
  - 客户端到服务器的来回时间损耗
  - 服务器端处理时间

具体网络时间：

Network | Page Events | Timings | Status Codes | Errors | ! Warnings (18) | Comment

Key	Description	Minimum	Maximum	Average	Total	Units
	Blocked	0.001	0.061	0.006	0.091	secs
	DNS Lookup	< 0.001	0.017	0.006	0.017	secs
	Connect	0.032	0.122	0.061	0.245	secs
	Send	< 0.001	0.001	< 0.001	0.007	secs
	Wait	0.018	0.422	0.126	2.145	secs
	Receive	0.001	0.164	0.015	0.258	secs
	Cache Read	0.003	0.035	0.011	0.075	secs
	TTFB	0.040	0.562	0.142	2.414	secs
	Network	0.044	0.726	0.157	2.672	secs

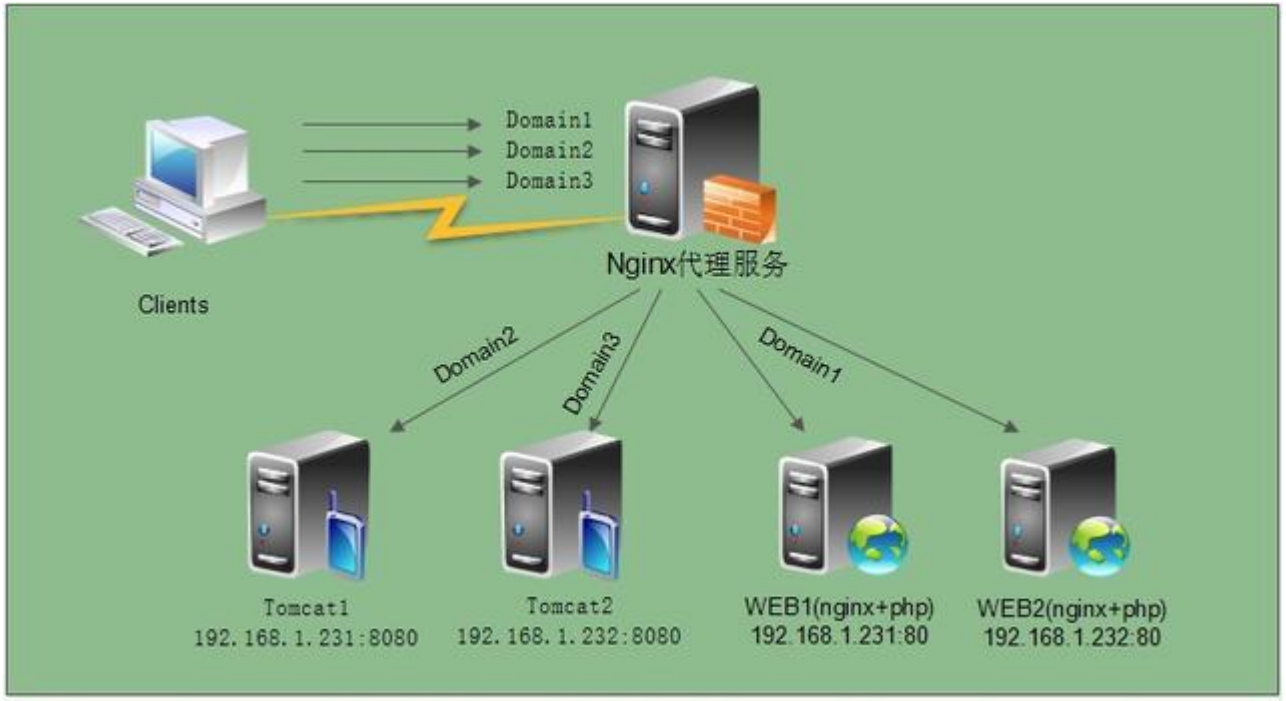
- **Wait:服务器处理时间**

前端性能分析与调优

- 页面级优化：减少HTTP请求数
- 将外部脚本置底
- Lazy Load Javascript（只有在需要加载的时候加载，在一般情况下并不加载信息内容。）
- 将CSS放在HEAD中
- 等等

后端性能分析

- 后端包括：
  - Web服务器
  - App服务器
  - DB服务器
  - 负载均衡服务器



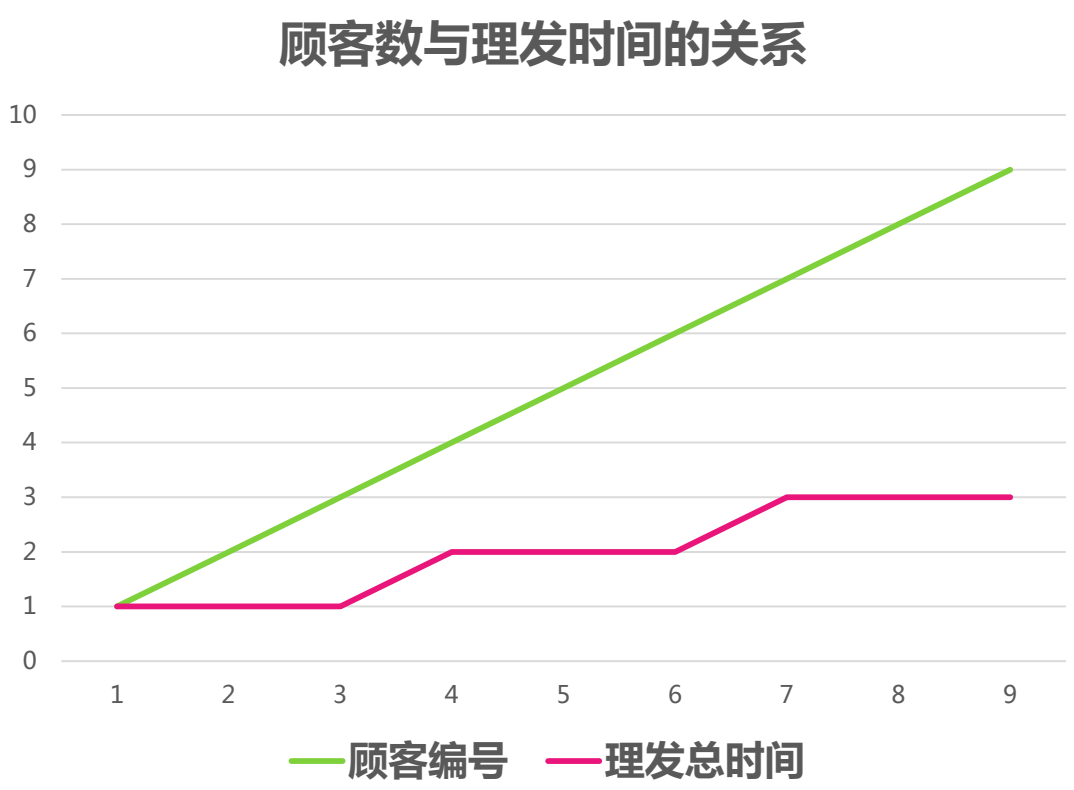
- 怎样知道系统慢时，是哪里出的问题
  - 发请求做跟踪分析

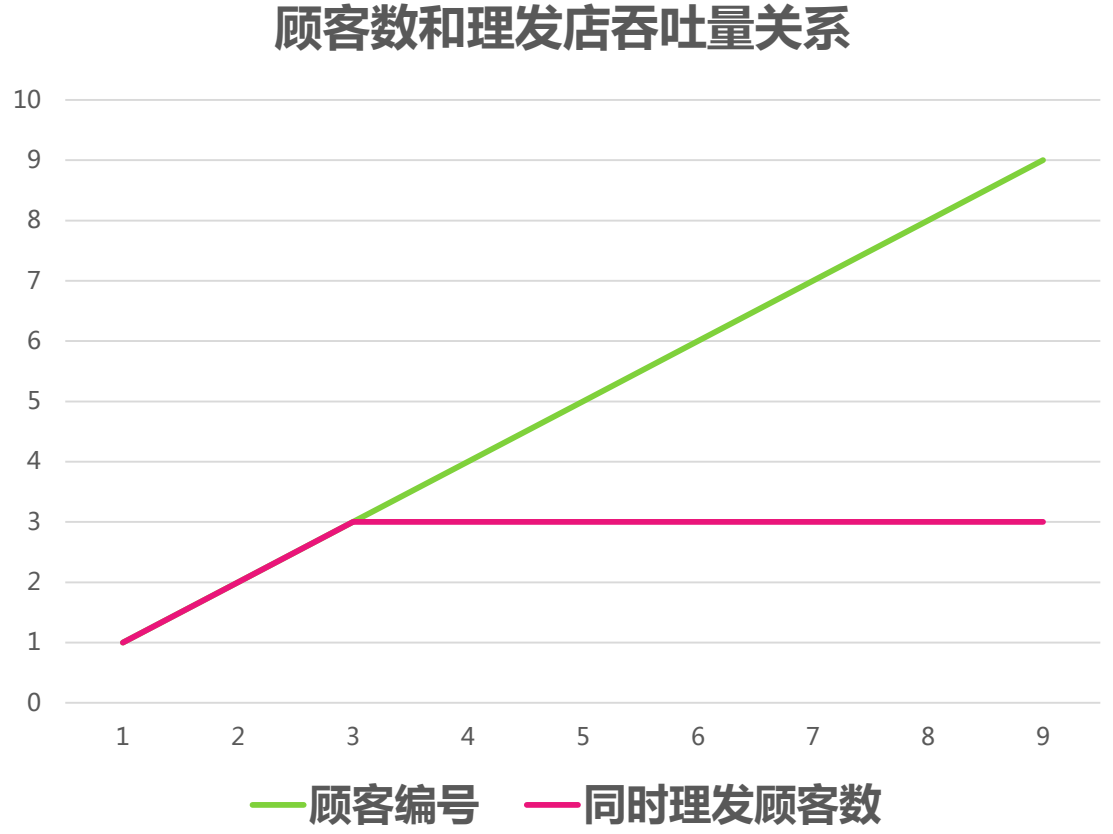
后端性能分析—生活中的例子

理发师模型—三个理发师情况

顾客编号	理发总时间	等待时间	同时理发顾客数	理发时间	师傅使用率
1	1	0	1	1	33%
2	1	0	2	1	66%
3	1	0	3	1	100%
4	2	1	3	1	100%
5	2	1	3	1	100%
6	2	1	3	1	100%
7	3	2	3	1	100%
8	3	2	3	1	100%
9	3	2	3	1	100%
10	/	/	/	/	/

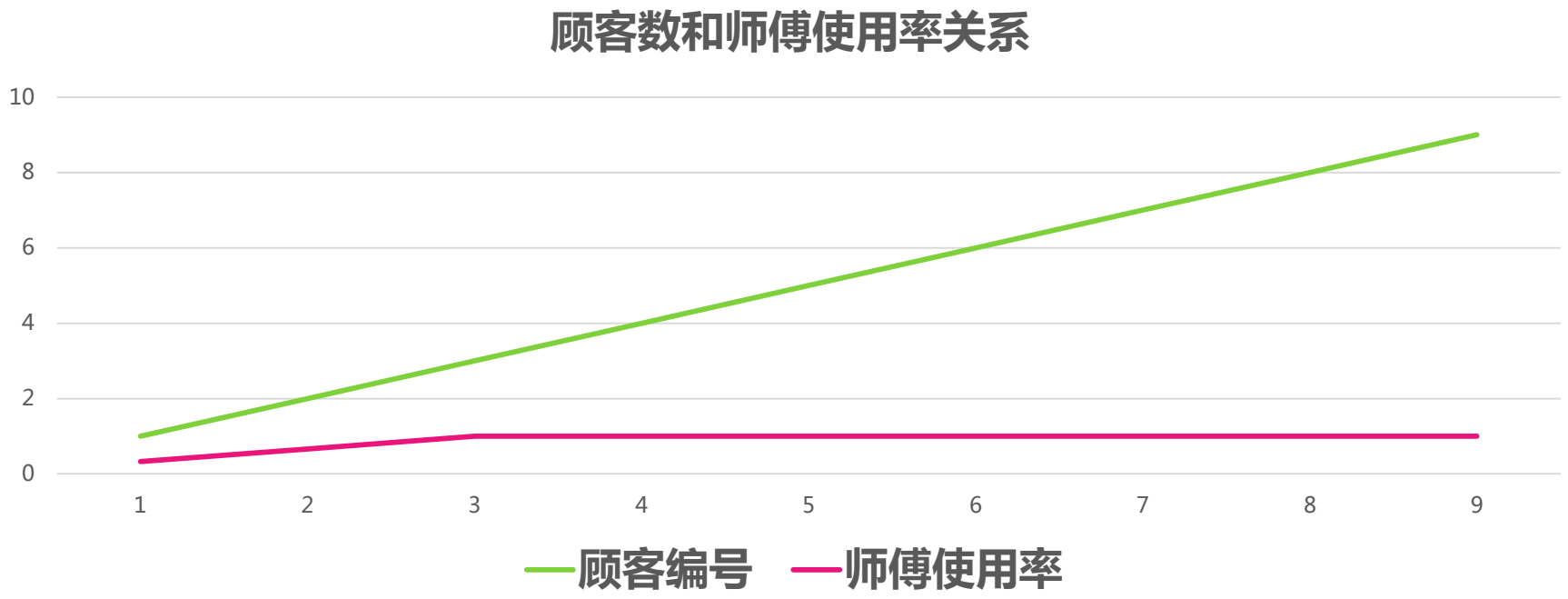
后端性能分析—生活中的例子





后端性能分析—生活中的例子

- 理发顾客数增加到峰值后就稳定了，因为没有更多师傅了
- 分析顾客数和师傅使用率关系



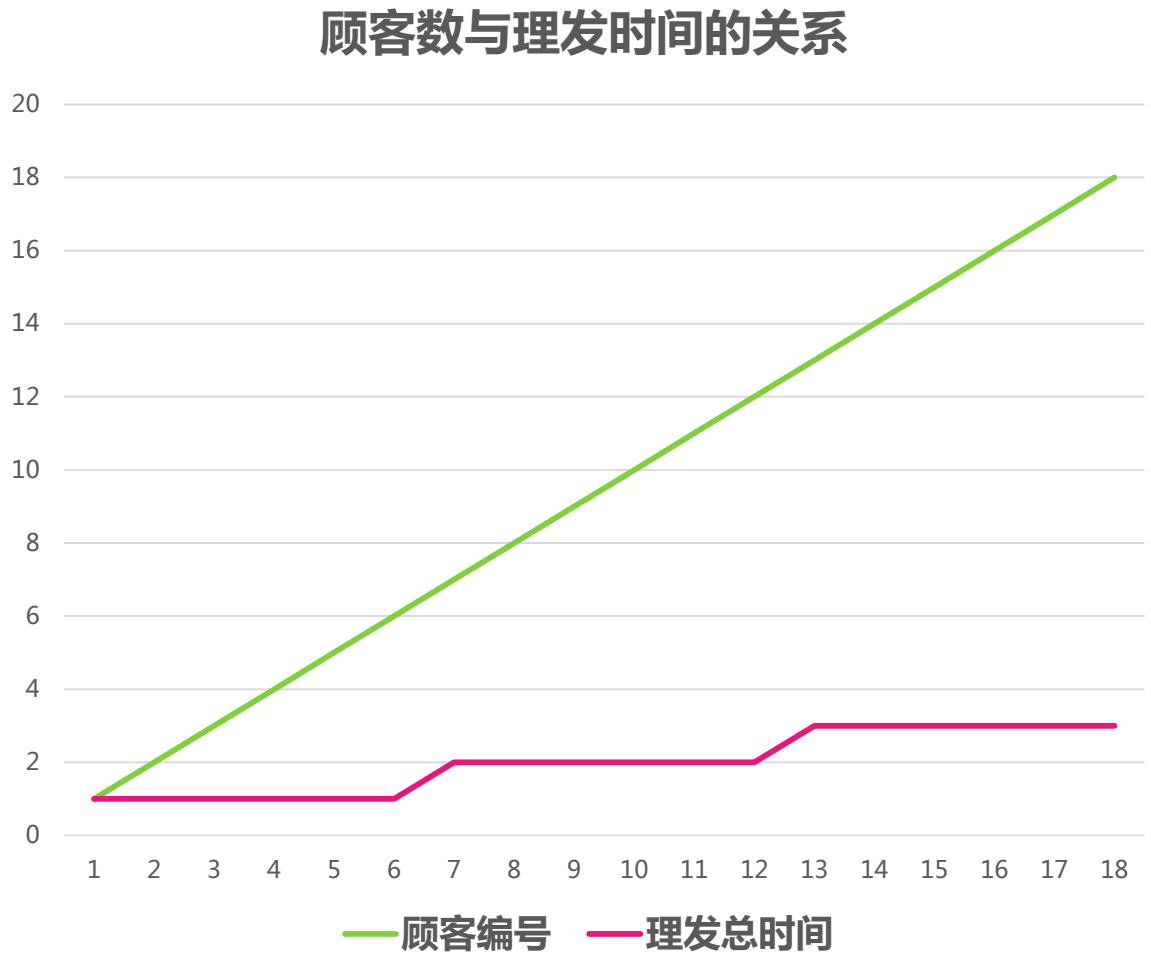
后端性能分析—生活中的例子

- 从图中可以看出，当有3个顾客在店时，师傅使用率到达了峰值，而此时理发总时间发生了波动
- 另一方面，最大顾客数也达到了峰值，可以认为由于资源的满负荷导致了同时理发的处理能力瓶颈，另一方面响应时间变长
- 如何解决这个问题？
  - 增加资源
  - 再请3名理发师

后端性能分析—生活中的例子

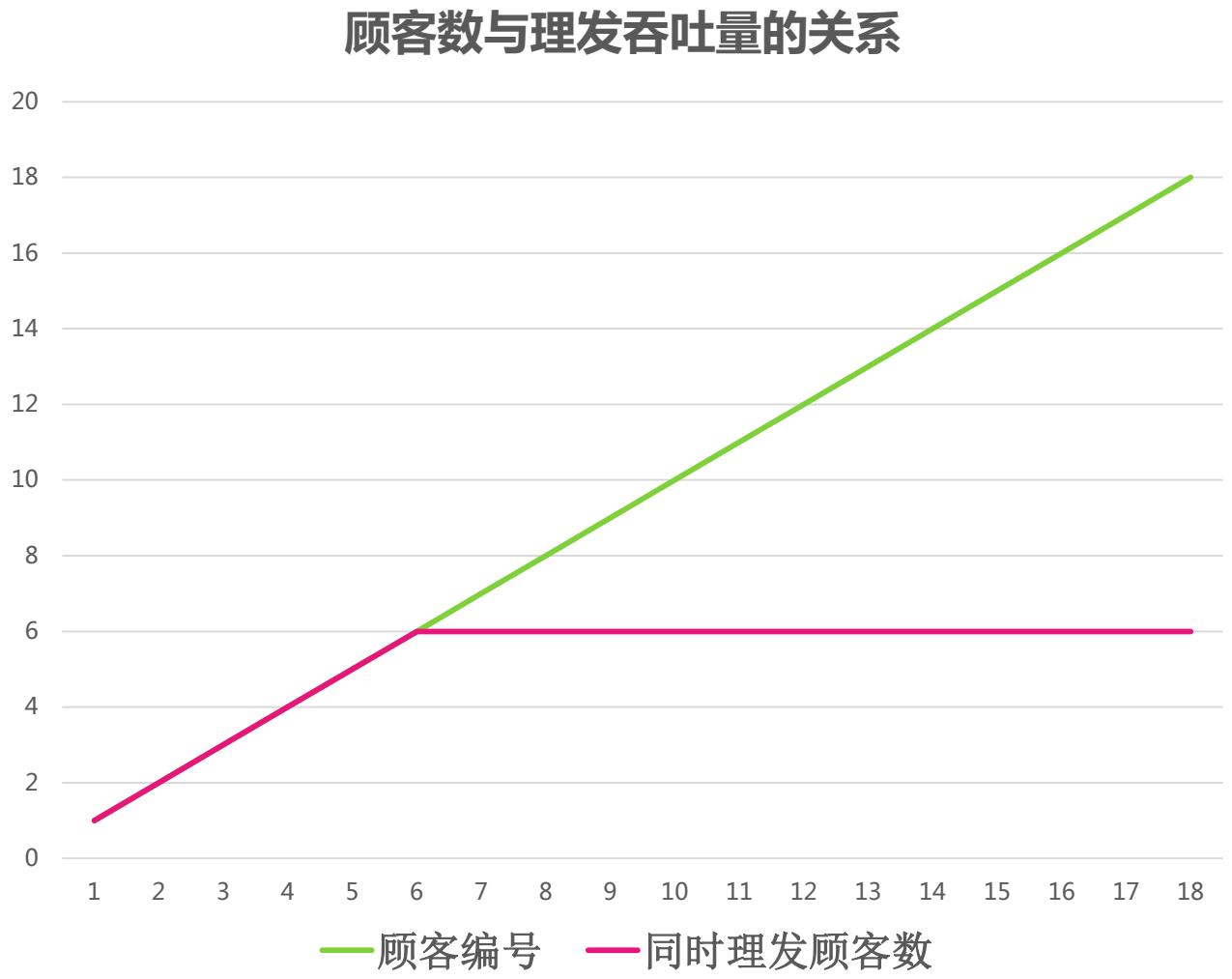
顾客编号	理发总时间	等待时间	同时理发顾客数	理发时间	师傅使用率
1	1	0	1	1	17%
2	1	0	2	1	33%
3	1	0	3	1	50%
4	1	0	4	1	67%
5	1	0	5	1	83%
6	1	0	6	1	100%
7	2	1	6	1	100%
8	2	1	6	1	100%
9	2	1	6	1	100%
10	2	1	6	1	100%
11	2	1	6	1	100%
12	2	1	6	1	100%
13	3	2	6	1	100%
14	3	2	6	1	100%
15	3	2	6	1	100%
16	3	2	6	1	100%
17	3	2	6	1	100%
18	3	2	6	1	100%
19	/	/	6	/	/

后端性能分析—生活中的例子



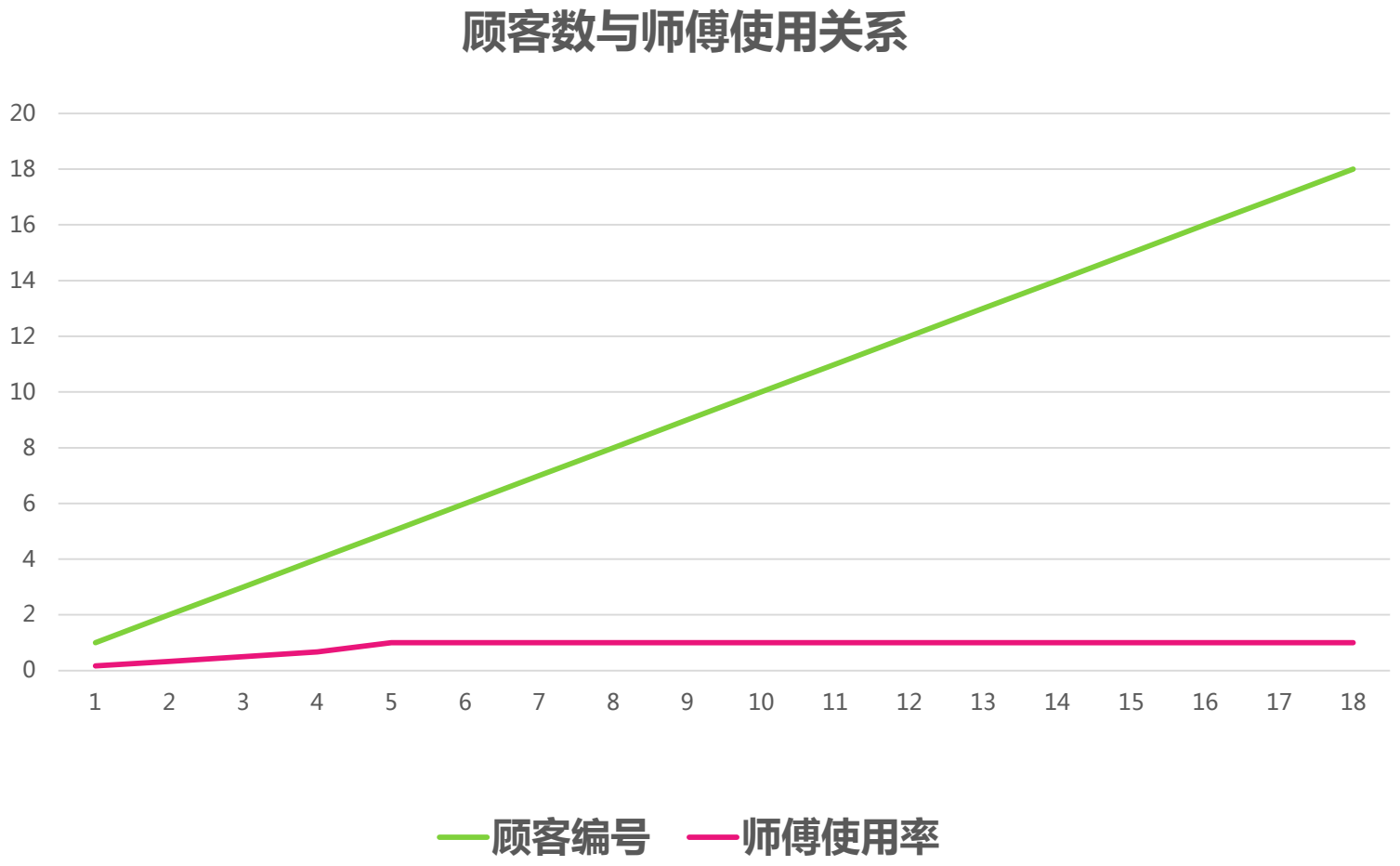
- 从该图可以看到理发总时间比刚才平稳了一些，理发店在顾客可以接受的时间范围内能够受理18个顾客同时理发，理发总时间仍然随着用户的增加而增加，只是更平稳了

后端性能分析—生活中的例子



- 随着用户量的增加，同时理发数量增加

后端性能分析—生活中的例子



- 随着顾客数量的增加，师傅使用率上升，当资源全部占用时，响应时间开始增加，而处理能力到达峰值，这是顾客数与响应时间、吞吐量和资源占用率的关系

后端性能分析—生活中的例子

- 顾客数与响应时间、吞吐量的关系：
  - 随着顾客数的增加资源利用率上升甚至完全占用
  - 当资源完全占用时，顾客会进入队列，响应时间变长，吞吐量达到峰值

后端性能分析—生活中的例子

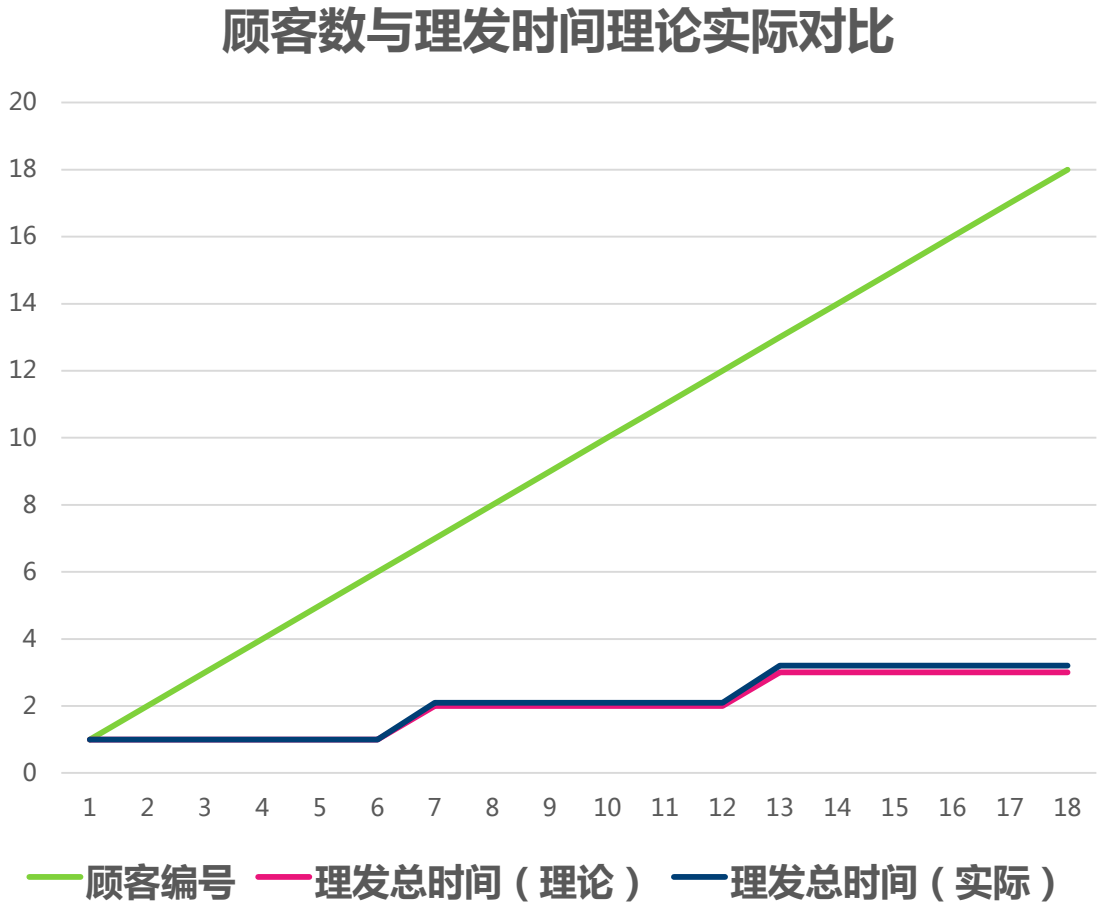
- 是否只要增加师傅数量，理发店就能提高处理能力
  - 如何知道哪些顾客先来，哪些顾客后来
  - 增加发号和叫号机制
  - 假如叫号过程耽误6分钟

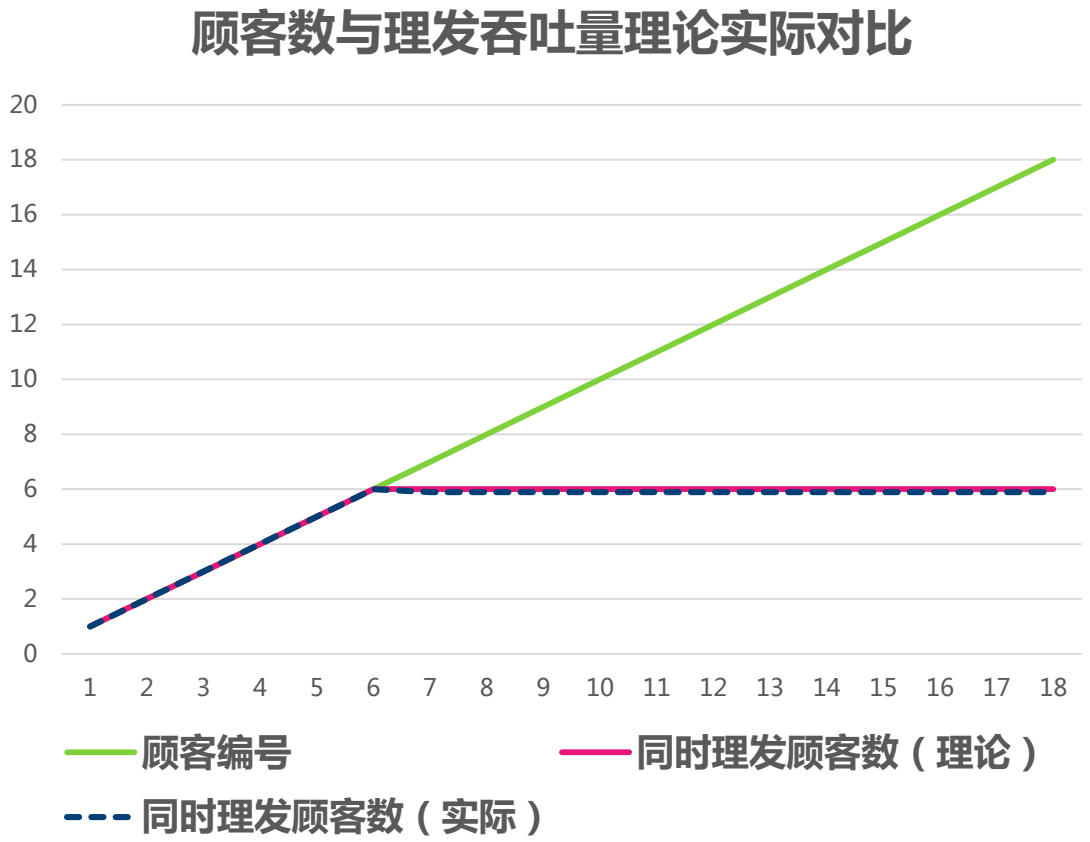
后端性能分析—生活中的例子

顾客编号	理发总时间	等待时间	同时理发顾客数	理发时间	师傅使用率
1	1	0	1	1	17%
2	1	0	2	1	33%
3	1	0	3	1	50%
4	1	0	4	1	67%
5	1	0	5	1	83%
6	1	0	6	1	100%
7	2.1	1.1	5.9	1	100%
8	2.1	1.1	5.9	1	100%
9	2.1	1.1	5.9	1	100%
10	2.1	1.1	5.9	1	100%
11	2.1	1.1	5.9	1	100%
12	2.1	1.1	5.9	1	100%
13	3.2	2.2	5.9	1	100%
14	3.2	2.2	5.9	1	100%
15	3.2	2.2	5.9	1	100%
16	3.2	2.2	5.9	1	100%
17	3.2	2.2	5.9	1	100%
18	3.2	2.2	5.9	1	100%
19	/	/	5.9	/	/

后端性能分析—生活中的例子

- 增加0.1小时叫号时间后，单位时间内理发的顾客数是低于6的





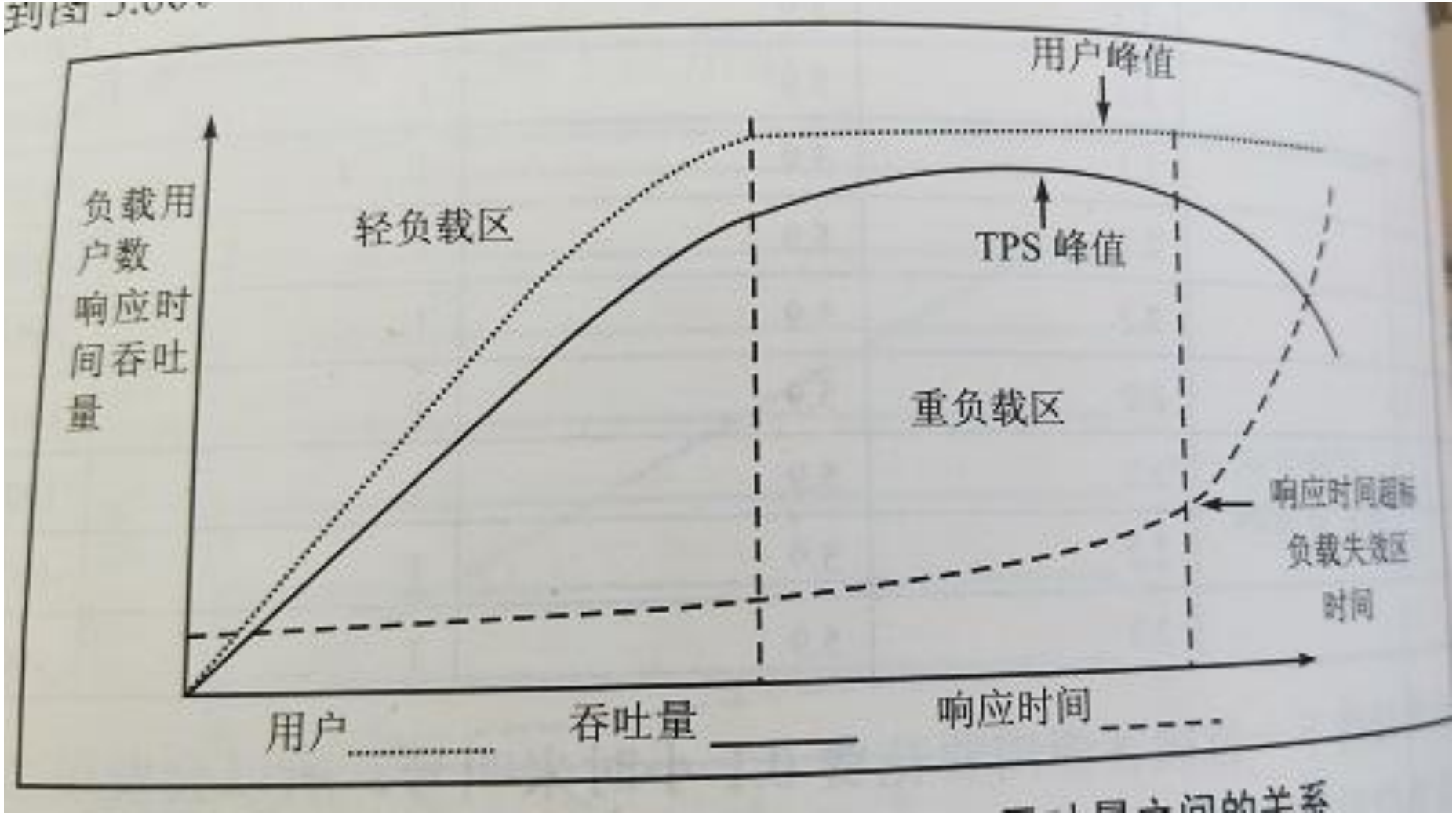
后端性能分析—生活中的例子

- 由于排队机制的开销，实际响应时间会比理论值略低，同时系统处理能力会先到达理论峰值

后端性能分析—性能指标总结

- 响应时间：
  - 通常开始平稳，后随负载的增加而逐步增加
- 吞吐量：
  - 反映系统处理能力指标。随负载量变化，吞吐量往往增长到一个峰值后下降，队列变长
- 服务器资源占用
  - 反映系统能耗指标。随着用户和吞吐量上升，服务器资源会被占用的越来越多，直到服务器资源被完全占用

后端性能分析



- 随着用户上升，响应时间与吞吐量的变化关系

后端性能分析

- 轻负载区
  - 随着用户数量上升，响应时间基本没有太大变化
  - 吞吐量随着用户增加而增加
  - 系统资源是足够的，系统能完全轻松处理业务，所以称为轻负载区

后端性能分析

- 重负载区
  - 当用户量上升，响应时间开始明显上升
  - 吞吐量上升速度变慢，并且到达峰值
  - 系统处理能力到达峰值，由于资源匮乏，吞吐量下降，响应时间变长
  - 处理能力达到极限

后端性能分析

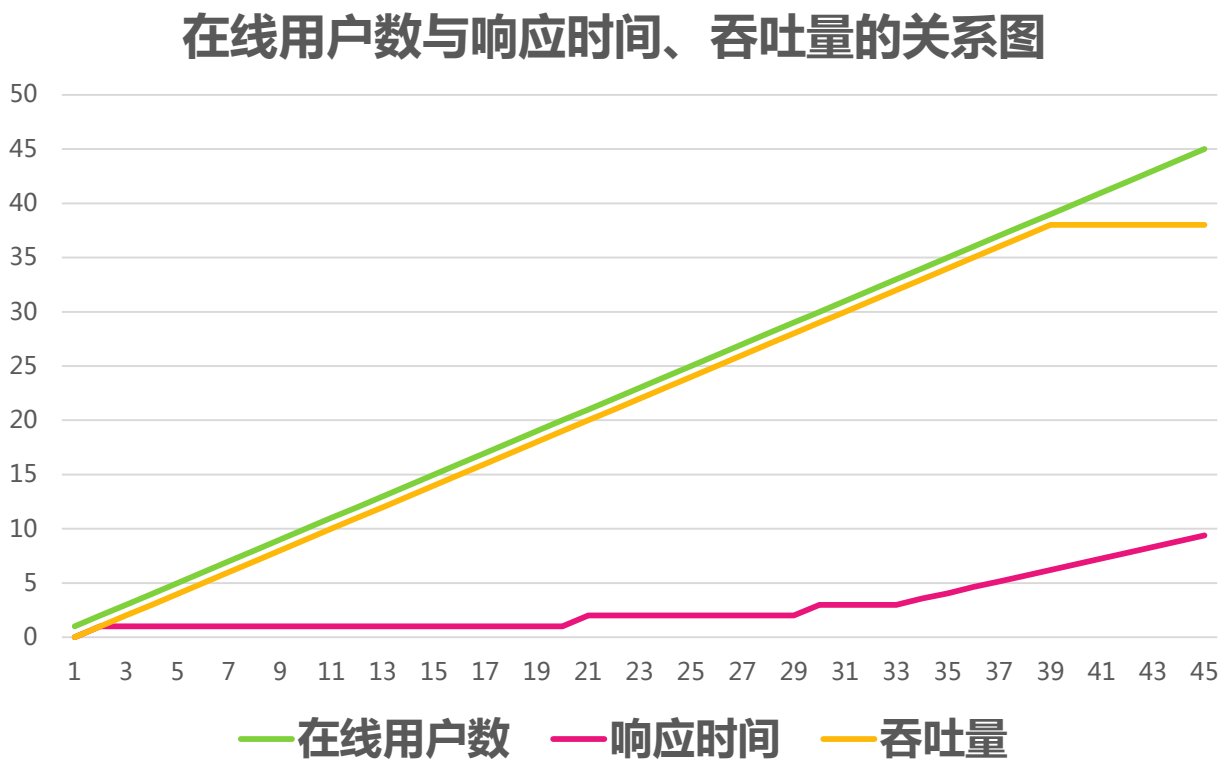
- 轻负载区、重负载区及分界点的分析：
  - 轻负载区到重负载区分界点用户数：系统最优的高性能用户数，系统资源被高效分配和利用
  - 重负载区吞吐量峰值：这个峰值是系统的最高处理能力，而同时的用户数也是系统所能达到的高性能处理承受的用户数，这个时刻的资源利用率应该正好达到峰值
  - 重负载区到负载失效区分界点用户数：系统所能达到的性能需求最大的在线用户数，超过这个数目的用户将无法正常访问系统

后端性能分析

- 负载失效区：当用户量继续增加时，响应时间会大幅下降，资源被消耗殆尽。当响应时间超出用户能够承受的范围时，这部分用户将会选择放弃访问

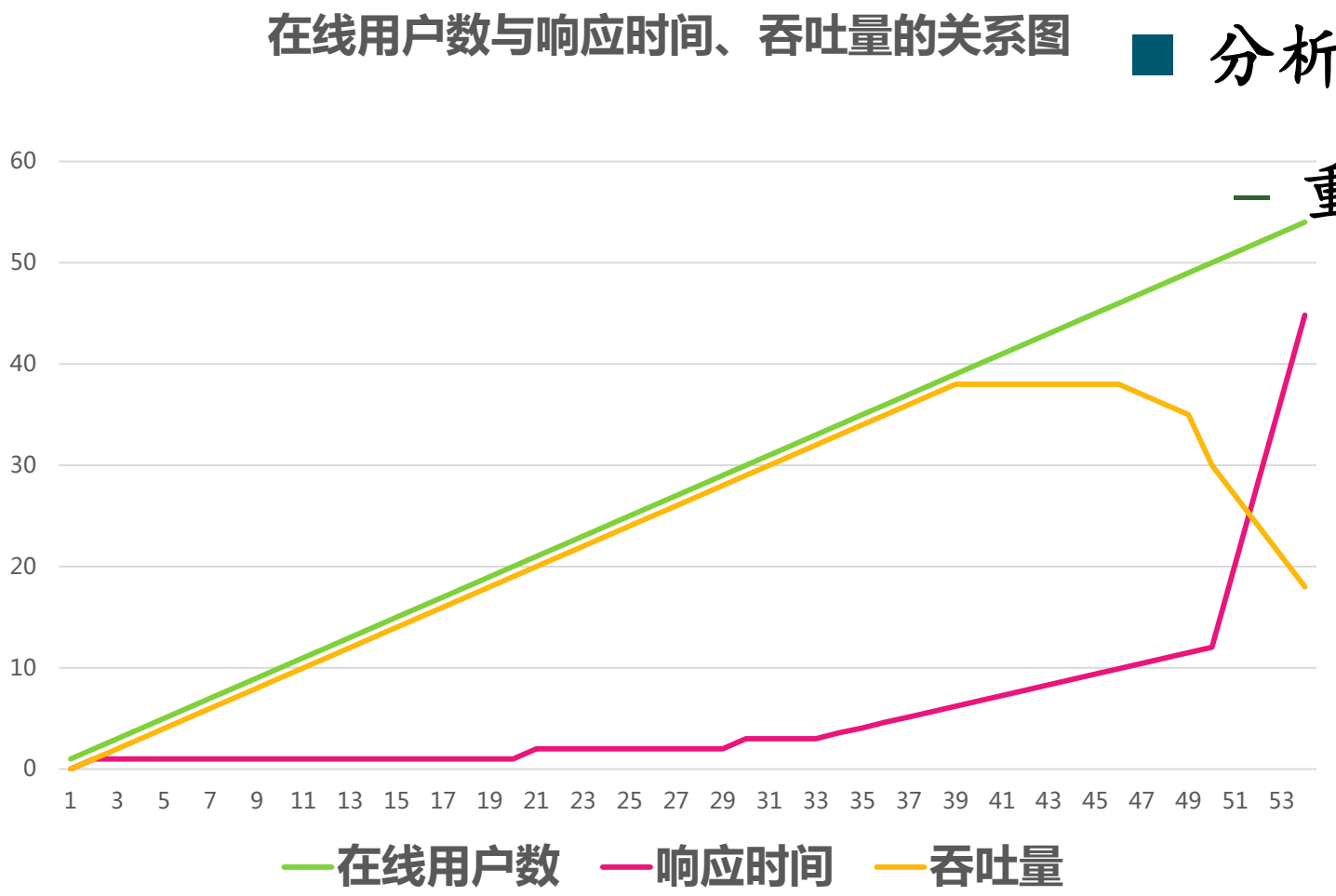
- 总结：一个系统最好的工作在哪个区域
  - 轻负载区，接近重负载区即可，不能出现系统进入重负载失效区的情况
  - 在做性能测试时，只要能够清楚地处理这3个区域，就能分析得到当前系统的负载状态及可能存在的瓶颈

后端性能分析



- 分析该图是哪种负载情况
  - 轻负载情况

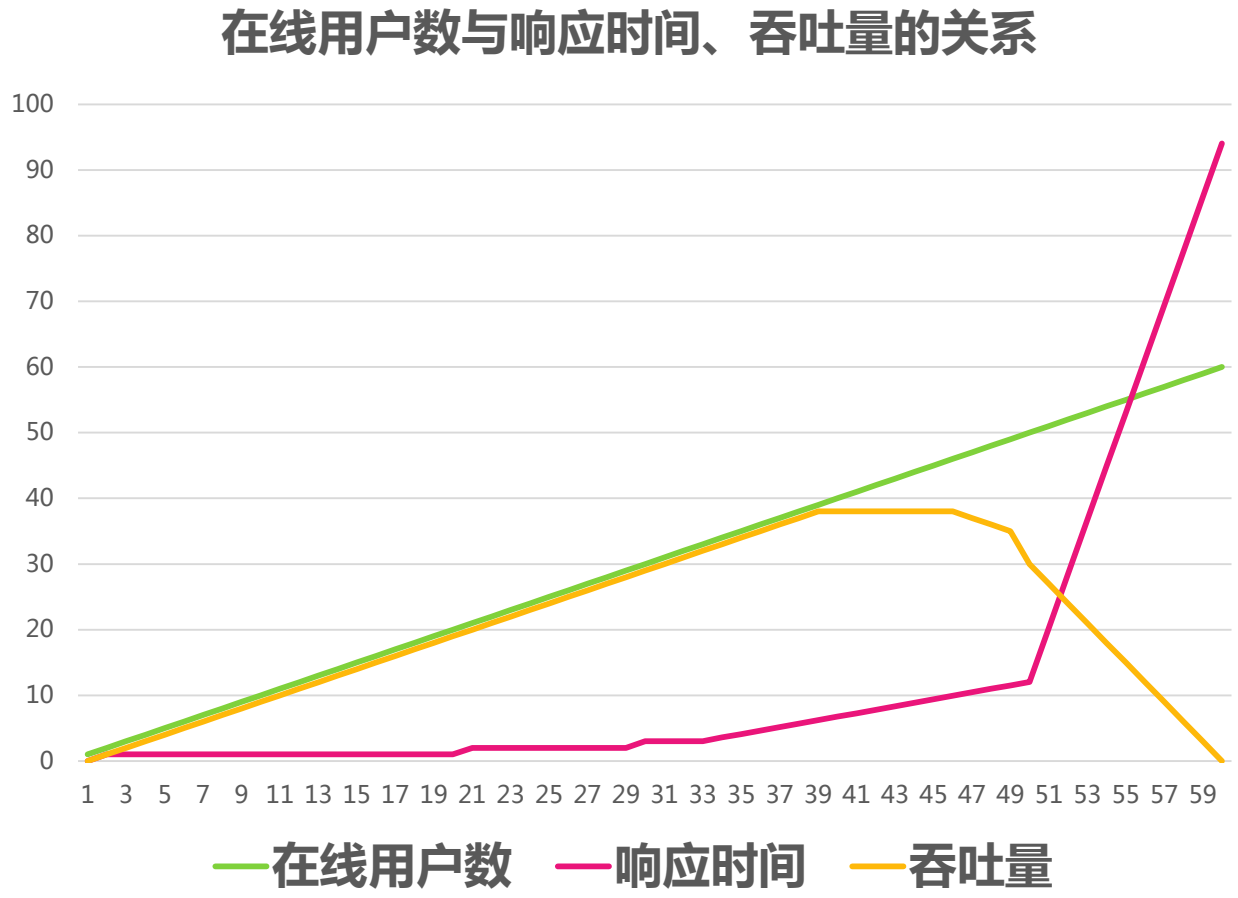
后端性能分析



分析该图是哪种负载情况

重负载情况

后端性能分析



- 分析该图是哪种负载情况
  - 超负载情况

LoadRunner中前端和后端的分析

- 使用Web Page Breakdown进行前后端性能分析
- 使用工具分析，找出哪方面性能不满足，结合理发师的例子进行系统调优
  - 基本性能指标值分析
  - 增加资源，查看各项性能指标的值
  - 找出系统瓶颈，然后针对性修改

后端性能分析方法

- 手工编写计数代码帮助定位
  - 分析应用层、数据层处理所耗费的时间
- 层层监控
  - 在表示层、应用层、数据层的各个服务器上监控，来剥离一个请求在各个服务器上的时间开销
- 通过理发师模型调优
  - 了解性能数据相关影响，设置手工场景，随着用户负载逐渐上升，观察响应时间和吞吐量的关系，来确定系统是否进入重载区

后端性能分析方法

- 当进入重载区时，检查相关资源计数器，确定资源瓶颈，在排除了硬件资源瓶颈后，可以进一步定位软件瓶颈

目录

书写测性能试报告

- 性能测试报告三大类
  - 定性型报告
  - 分析型报告
  - 比较型报告

书写测性能试报告

- 定性型
  - 强调系统性能最终是否能满足用户的性能需求。例如：测试目标为系统能够达到5000用户同时在线，响应时间在3秒以内
  - 一般采用目标场景
  - 一般应用在验收测试

书写测性能试报告

- 分析型性能测试报告
  - 给出对于一个软件版本详细分析结论
  - 指出该版本的优劣
  - 但由于缺少参考标准，这种报告不能说明什么，只介绍性能测试方式和结果，得到负载测试的数据参考
  - 采用手工Basic方式运行
  - 通过逐渐增加负载，找到当前系统的负载情况，最终得到最优访问量、失效访问量等关键数据，从中提出可以导致瓶颈的原因，还需要通过配置测试后的比较型报告来证实

书写测性能试报告

- 比较型性能测试报告
  - 明确各个版本之间的优缺点，并给出相关选择的建议
  - 通过对比两个或多个产品线，给用户一个高、中、低配置选择
  - 并提出性价比最高的选择方案
- 比较型性能测试报告另一种使用场景：同一个软件在不同配置下（例如：代码调优），通过运行相同的业务，对比修改后的结果，确认系统调优是否有效

书写测性能试报告

- 比较型性能测试报告使用注意事项
  - 脚本的工作原理应该完全相同
  - 场景必须完全相同
  - 每次对比测试的环境必须相同

书写测性能试报告

- 测试报告格式
  - 性能测试目的
  - 测试环境
  - 测试工具及测试方法的说明
  - 测试结果数据列表
  - 比较分析
  - 总结性能测试结论

内容总结

- 搭建测试环境
  - 注意事项
- 脚本业务报告
- 性能测试过程实践
  - 熟悉需求，搭建环境，写计划，写方案，写用例，设计测试脚本，设计测试场景、执行并监控测试场景、写测试报告

内容总结

- 性能分析与调优
  - 前端性能分析
  - 后端性能分析
- 书写性能测试报告
  - 报告类型
  - 每种类型适用场合
  - 每种报告的书写

Question