大型电商分布式系统实践 第11周

DATAGURU专业数据分析社区

上节内容回顾

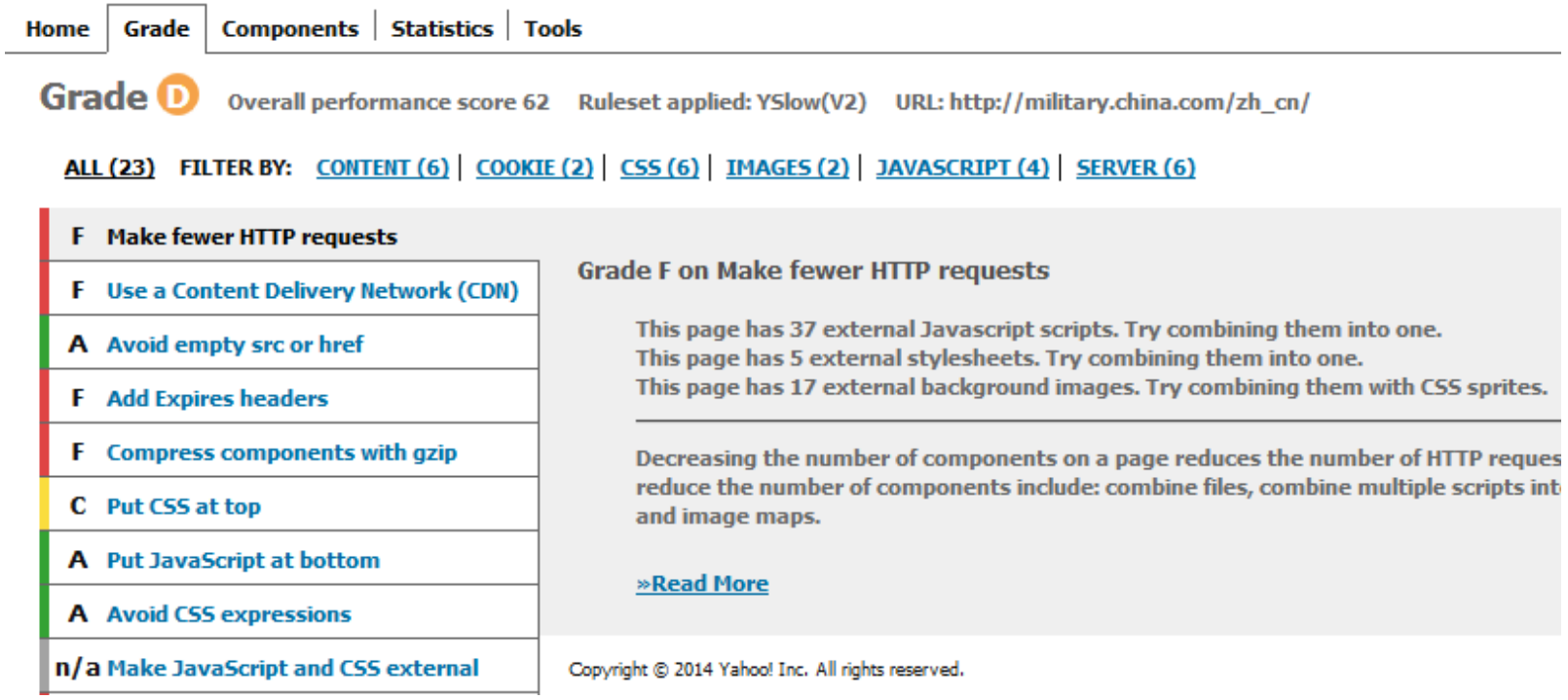
如何寻找性能瓶颈

对于性能优化来说，第一步也是最重要的一步，便是寻找可以优化的点，也就是所谓的性能瓶颈，性能瓶颈实际上就是木桶原理中最短的那一块木板，只有补上这块短板，才能够更好的发挥应用的整体性能。

WEB的性能优化涉及到包括前端优化，服务端优化，操作系统优化，数据库查询优化，JVM调优等等众多领域的知识，每个领域如果深入去发掘，都可以编著成一本相关的书籍。因此，本小节将只是介绍一些常用的方法以及工具，来帮助快速定位性能的瓶颈。

前端优化工具--yslow

YSlow是yahoo!提供的用于网页性能分析的浏览器插件，它通过一系列由yahoo！提出并且得到业界充分认可的页面性能评估规则，来对当前页面进行性能检测，提供F-A 6个级别的评分，F代表最差，A表示最好，并且给出分析所得到的相关数据以及优化的具体建议。我们可以通过相关数据，快速发现页面的不足，并对自己的网站和服务器做相应的优化。



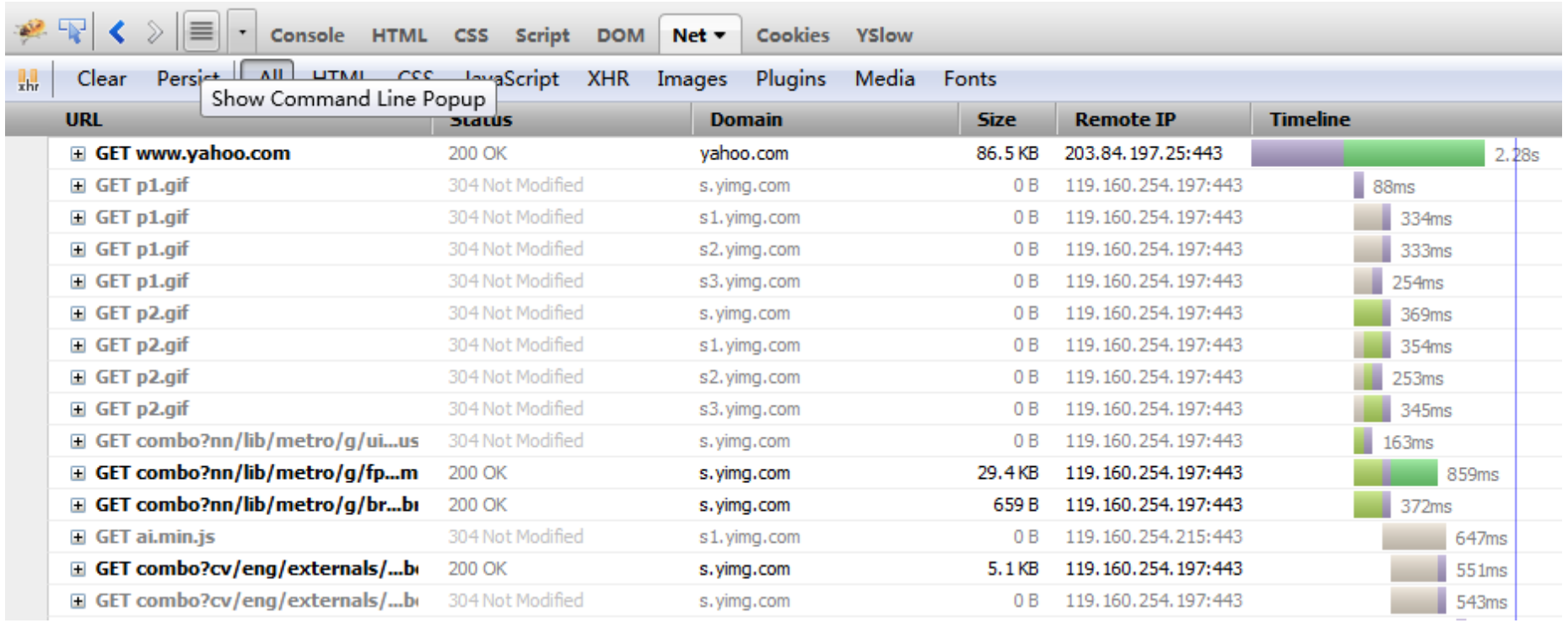
前端性能优化

yahoo!曾对网站速度优化提出了非常著名34条准则，后精简为更为直观的23条，这些著名的规则最后都反映在YSlow的评分体系上，YSlow会针对每一条来进行评分，如：

页面的HTTP请求数量
是否使用CDN网络
是否使用压缩
样式放在页面首部加载
避免CSS表达式
减少DNS查找
将脚本放在页面底部加载
。。。。。。

页面响应时间

服务端单个请求的响应速度，跟整个页面的响应时间以及页面的加载速度密切相关，它也是衡量页面性能的一个重要指标。借助另外一个工具**firebug**，我们能够清楚的看到，整个页面的加载时间，以及具体每一个请求耗费的时间。这样我们便能够快速找到响应慢的请求，分析原因，进行优化。



服务端优化—方法响应时间

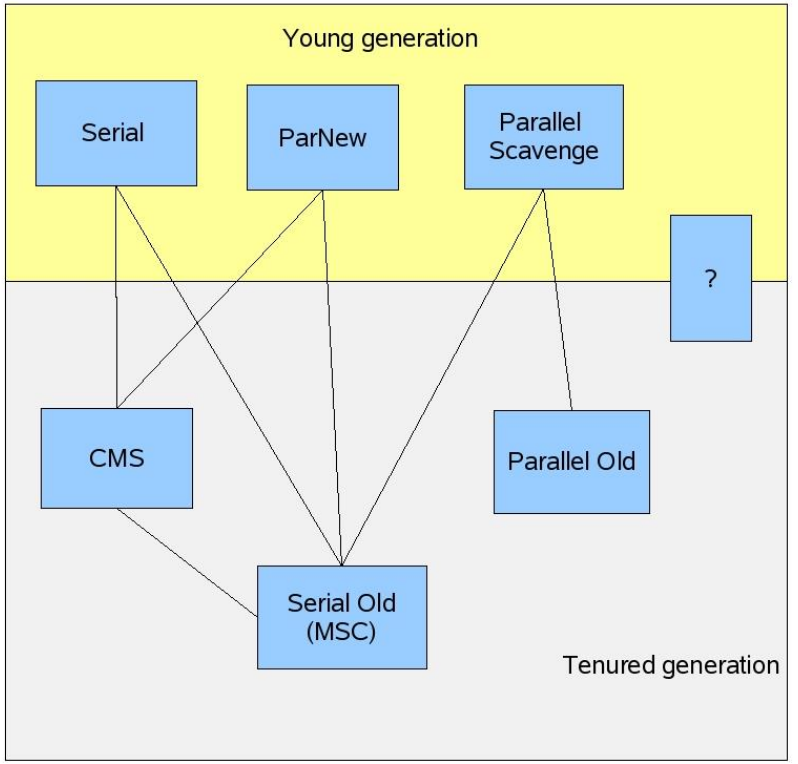
定位到响应慢的请求以后，接下来便需要深入发掘导致请求响应慢的原因，并且定位到具体的代码。通过对代码的检查分析，能够定位到具体的方法和代码行，但是通常来说这种方式比较浪费时间，且容易遗漏，通过java环境下的一个十分有效的动态跟踪工具--**btrace**，能够快速的定位和发现耗时的方法。

一段测试代码：

```
@Override
protected void doPost(HttpServletRequest req, HttpServletResponse resp)
                    throws ServletException, IOException {
        PrintWriter out =  resp.getWriter();
        try {
                Thread.sleep(500L);
        } catch (InterruptedException e) {}
        out.write("success");
}
```

服务端优化—GC日志分析

GC日志能够反映出java应用执行内存回收详细情况，如Minor GC的频繁程度，Full GC的频繁程度，GC所导致应用停止响应的时间，引起GC的原因等等。根据程序吞吐量优先还是响应时间优先的不同，sun HotSpot虚拟机1.6版在服务器端提供Parallel Scavenge/Parallel Old以及ParNew/CMS两种比较常用的垃圾收集器的组合，其中Parallel Scavenge和ParNew为新生代的垃圾收集器，而Parallel Old和CMS为老年代的垃圾收集器。



服务端优化—GC优化

Case1，下面这段GC日志：
2012-02-19T05:00:46.461+0800: 139.170: [Full GC [PSYoungGen: 1907328K->0K(1910144K)] [PSOldGen: 2665961K->1211211K(2899968K)] 4573289K->1211211K(4810112K) [PSPermGen: 94244K->94244K(98560K)], 4.9259400 secs]
[Times: user=5.19 sys=0.00, real=4.93 secs]

导致GC的原因是由于YoungGen的空间难以满足新对象创建的需要，并且，由于Parallel Scavenge垃圾收集器的悲观策略，每次晋升到OldGen的平均大小如果大于当前OldGen的剩余空间，则触发一次FullGC。上述情况如果频繁发生，则可以通过-Xmx与-Xms参数调整整个堆的大小，以增加OldGen的大小，YoungGen对应的-Xmn保持不变。

服务端优化—GC优化

Case2，下面这段GC日志：
2012-07-13T11:21:45.423+0800: 4070.053: [Full GC [PSYoungGen: 53055K->0K(2488128K)] [ParOldGen: 350072K->279976K(2682880K)] 403128K->279976K(5171008K) [PSPermGen: 262143K->132624K(262144K)], 1.8700750 secs] [Times: user=10.46 sys=0.00, real=1.87 secs]

通过日志可以发现，导致FullGC的原因是由于PermGen空间被占满，PermGen通常用来存放已被虚拟机加载的类信息，以及常量、静态变量、即时编译器编译后的代码等数据。PermGen的空间由于内存回收条件十分苛刻，在应用启动后一般都比较稳定，并且通过GC回收的内存也十分有限。如果频繁因为PermGen的空间不够用而发生FullGC，一种情况可能是由于PermGen设置的确实过小，对于Groovy一类的动态语言来说，会频繁的进行类型的加载操作，这时需要调整-XX:PermSize和-XX:MaxPermSize两个参数的大小，就可以解决，另一种情况则可能是由于错误的代码导致的频繁类加载，需要使用jmap将堆dump下来进行分析，以定位具体的错误代码位置。

服务端优化—数据库查询

通过mysql的配置文件my.cnf，可以修改慢日志的相关配置：

log_slow_queries = /var/log/mysql/mysql-slow.log
long_query_time = 1

```
# Here you can see queries with especially long duration
log_slow_queries        = /var/log/mysql/mysql-slow.log
long_query_time = 1
```

其中，log_slow_queries用来指定慢日志的路径，而long_query_time用来指定慢于多少秒的SQL会被记录到日志当中。

查询优化—合理使用索引

索引是影响数据库性能的一个重要因素，一旦索引使用不当，将严重影响数据库的性能。mysql提供了explain命令，用来解释和分析SQL查询语句，通过explain命令，可以模拟查询优化器执行SQL语句，从而知道mysql是如何执行你的SQL语句的。

explain select * from order_info where order_id = 1;

```
mysql> explain select * from order_info where order_id = 1;
+----+-------------+------------+-------+---------------+---------+---------+-------+------+-------+
| id | select_type | table      | type  | possible_keys | key     | key_len | ref   | rows | Extra |
+----+-------------+------------+-------+---------------+---------+---------+-------+------+-------+
|  1 | SIMPLE      | order_info | const | PRIMARY       | PRIMARY | 4       | const |    1 |       |
+----+-------------+------------+-------+---------------+---------+---------+-------+------+-------+
```

查询优化—合理使用索引

建了索引就一定能够使用到么，不见得，某些情况下，查询的列即使建了索引，也不一定能够用上。

表user: a，b，c三个字段建立组合索引

可以使用索引：

```
select * from user  where   a=1
select * from user  where   a=1 and b=2
select * from user  where   a=1 and b=2  and c=3
```

不能使用索引：

```
select * from user  where   a>1  and b<2
```

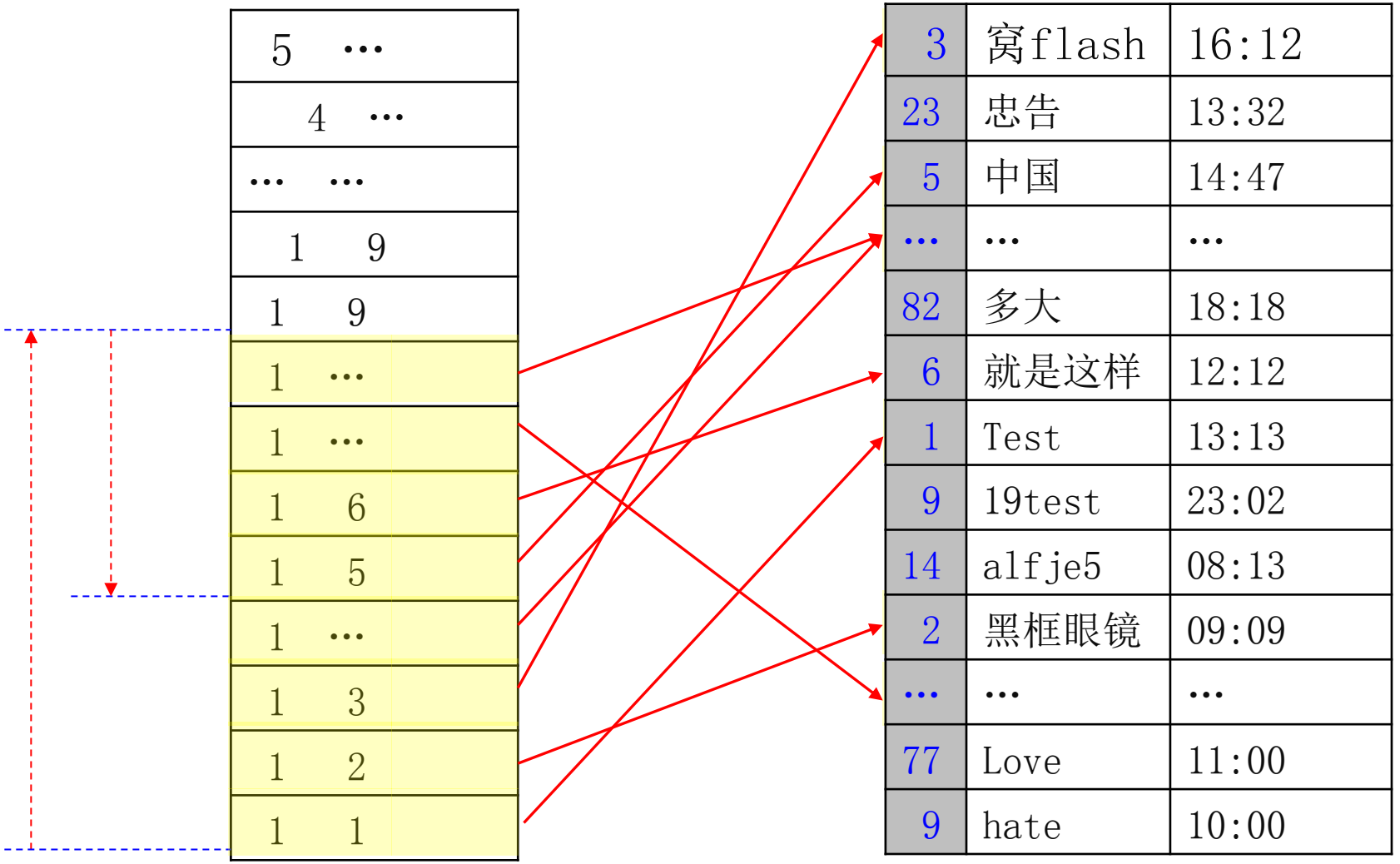
查询优化—普通分页

```
– SELECT *
–    FROM feed_0000
– WHERE num_id = 50842480985
–   AND suspended = 0
–   AND rate = 1
– ORDER BY id
– LIMIT 4, 4
```

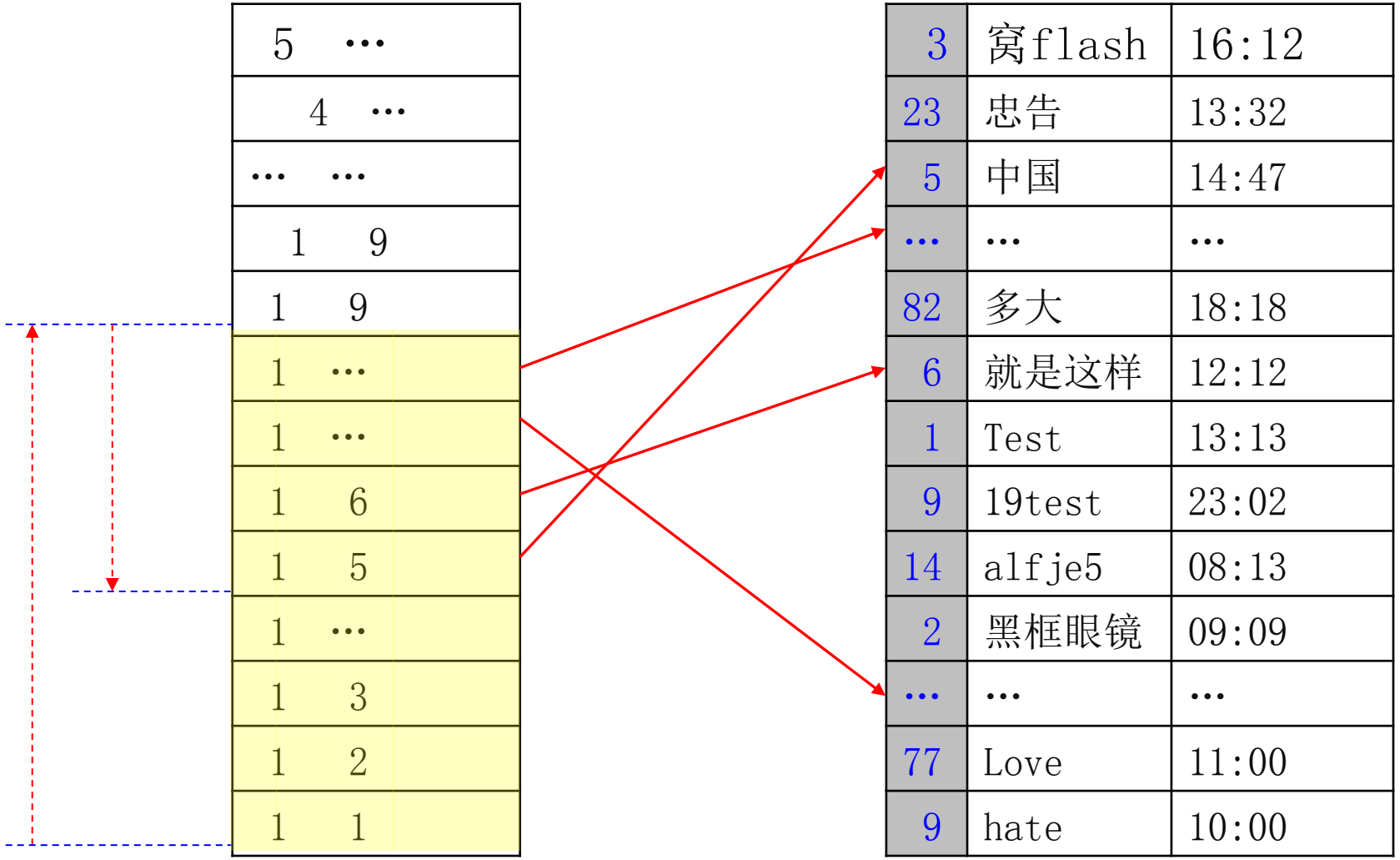
查询优化—优化分页

```
–SELECT *
–    FROM ( SELECT id
–           FROM feed_0000
–          WHERE num_id = 3606022462
–            AND suspended = 0
–            AND rate = 1
–          ORDER BY id
–          LIMIT 4, 4 ) a, feed_0000 b
– WHERE B.id = A.id
– ORDER BY B.ID
```

查询优化—普通分页查询路径



查询优化—优化后分页查询路径



查询优化—SQL编写总结

- 尽量条件确定
- 尽量使用等号
- 访问尽量少的数据
- 尽量避免聚合运算
- 尽量避免表关联
- 关联表的分页尽量在一张表中完成

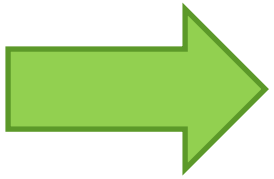
服务端优化—反范式设计

关系数据库理论所提出的范式设计，要求在表的设计过程中尽可能的减少数据冗余，这样会带来很大好处。但是，对于大多数复杂的业务场景来说，数据展现的维度不可能是单表的，因此，在进行查询操作时，需要进行表的关联，这不仅代价高昂，且由于查询条件指定的列可能并不在同一个表中，因此也无法使用到索引，这将导致数据库的性能严重下降。

为了尽可能的避免关联查询带来的性能损耗，有人提出了反范式设计，即将一些常用的需要关联查询的列进行进行冗余存储，以便减少表关联带来的随机I/O和全表扫描。

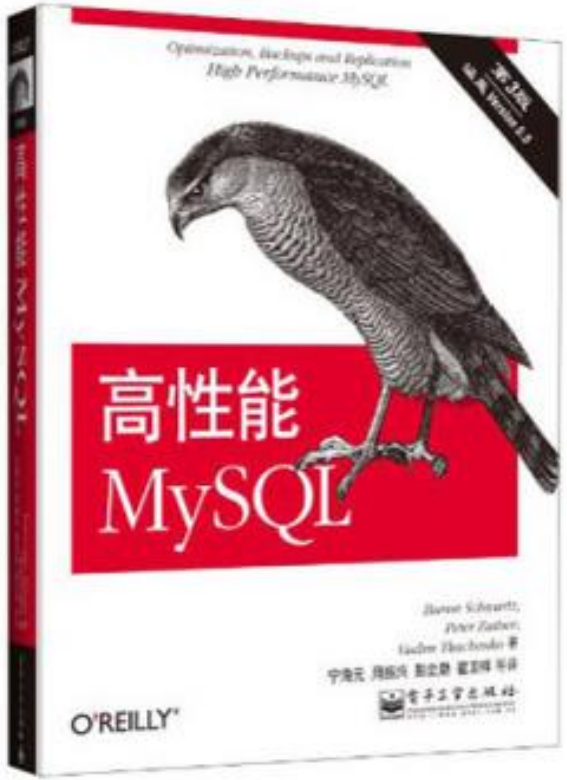
服务端优化—反范式设计

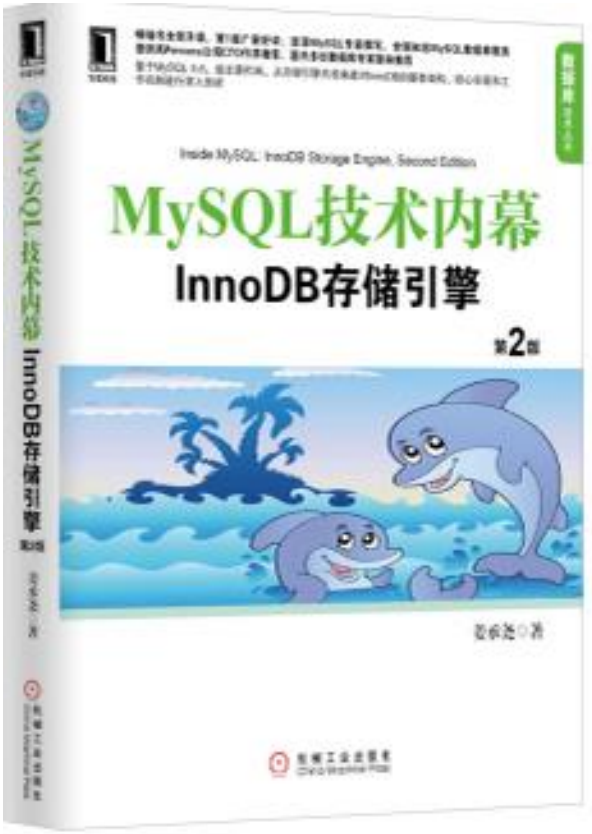
```
create table user(
user_id int primary key auto_increment,
user_nick varchar(100) ,
sex int,
age int,
introduce varchar(400)
);

create table order_info(
order_id int primary key auto_increment,
user_id int ,
price int,
good_id int,
good_title varchar(100),
good_info varchar(500)
)
```



```
create table order_info_ext(
order_id int primary key auto_increment,
user_id int ,
user_nick varchar(100) ,
sex int,
age int,
price int,
good_id int,
good_title varchar(100),
good_info varchar(500)
);
```

推荐两本mysql相关的书籍





为什么要性能测试

性能测试指的是通过一些自动化的测试工具模拟多种正常、峰值以及异常负载条件来对系统的各项性能指标进行测试，系统在上线运行之前，需要经过一系列的性能测试，以确定系统在各种负载下的性能指标的变化，发现系统潜在的一些瓶颈和问题，通过性能测试，能够得到应用性能的基准线，即系统能够承载的峰值访问，高位运行时系统的稳定性，以及系统响应时间等一系列关键指标，给系统的上线运维提供了重要的参考依据。

性能测试工具—ab、jmeter、LoadRunner

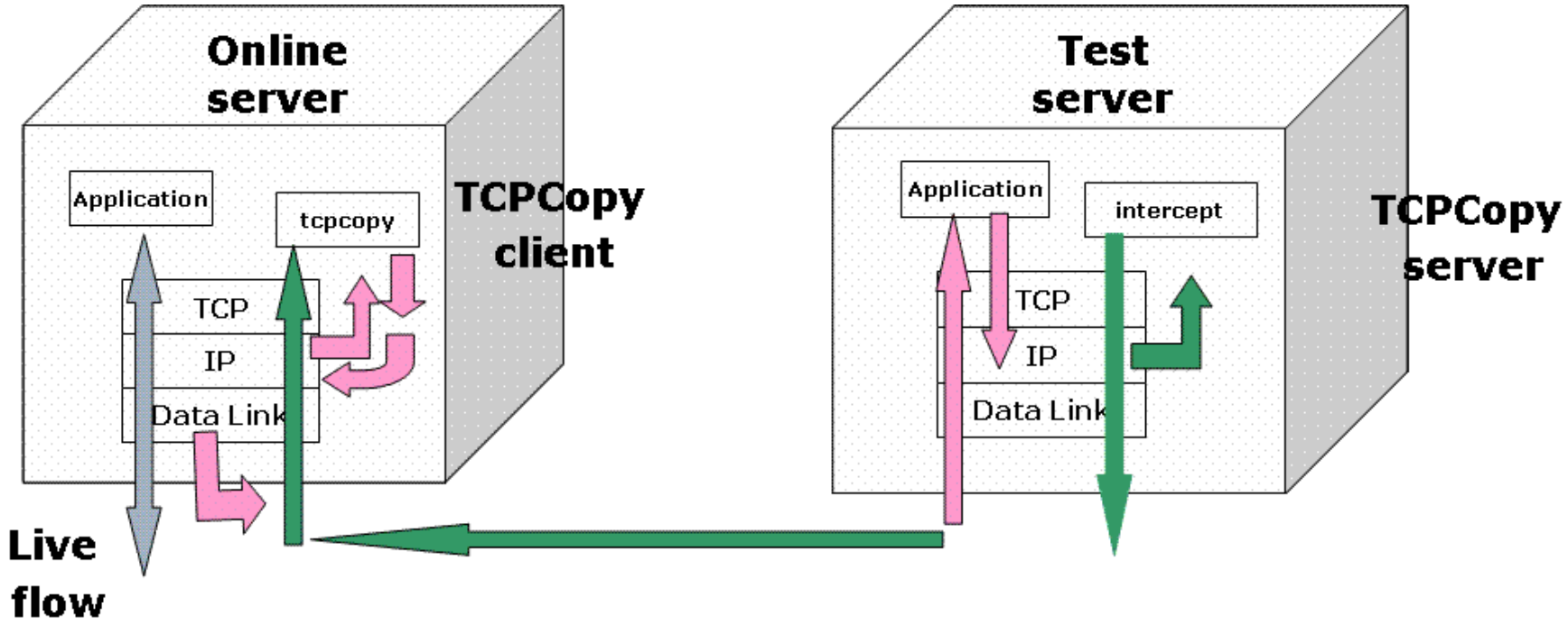
ab的全称为ApacheBench，是apache基金会提供的一款专门用来对HTTP服务器进行性能测试的小工具，可以模拟多个并发请求对服务器进行压力测试，得出服务器在高负载下能够支持的qps以及应用的响应时间，为系统设计者提供参考依据。

jmeter是apache基金会提供的另一个开源性能测试工具，它的功能比ab更为强大，采用纯java实现，支持多种协议的性能基准测试，如HTTP、SOAP、FTP、TCP、SMTP、POP3等等，可以用于模拟在服务器、网络或者其他对象上施加高负载，以测试他们的压力承受能力，或者分析他们在不同负载的情况下的性能表现，能够灵活的进行插件化的扩展，支持通过脚本方式的回归测试，并且提供各项指标的图形化展示。

Load Runner是惠普(HP)公司研发的一款功能极为强大的商业付费性能测试工具，它通过模拟大量实际用户的操作行为及实时性能检测的方式，帮助更加快速的查找和确认问题，此外，LoadRunner能够支持最为广泛的协议标准，适应各种体系架构，几乎是应用性能测试领域的行业标准。

性能测试工具—TCPCopy

TCPCopy是网易技术部于2011年9月所开源的一个项目，它是一款请求复制工具，能够将在线请求复制到测试机器，模拟真实环境，达到程序在不上线的情况下承担线上真实流量的效果，目前已广泛用于国内各大互联网公司。



性能测试工具----适合分布式环境的性能测试工具

1.能够通过分布式集群模拟大规模的流量

2.通过JVM的接口、操作系统的接口，能够实时观测和记录被压测机器状态，包括GC、load、网络流量、qps、rt时间，热点方法等

3.自动化的机器准备、环境搭建、账号生成等等，并且测试用例可以回放

4.测试环境能够模拟的场景有限，因此，工具需要能够支持线上日志回放，线上真实环境引流压测，全链路压测的功能

性能测试工具—性能环境与真实环境的差异

Thanks

FAQ时间