# Part 1 Base

## 1.1 性能指标

PV QPS TPS

* 并发数：Concurrency。能同时处理的用户数。高并发
* 响应时间：发出请求到收到相应数据的时间
* 吞吐量：系统单位处理的请求数。QPS（query per second）
* 性能计数器：包括System Load、对象和线程数、内存、CPU、磁盘、网络IO

## 1.2 性能测试方法

* 性能测试：性能预期
* 负荷测试：找到性能安全临界线
* 压力测试：超过安全符合继续施压，获得最大压力承受能力

## 1.3 Protocol

### HTTP 1.1

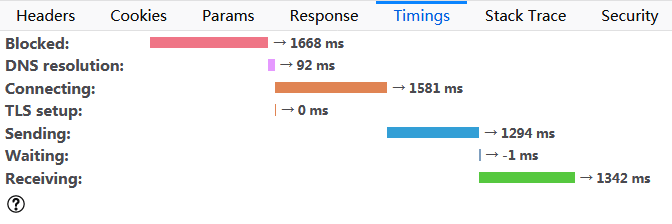
**最大并发数限制**

并发指建立的TCP连接数。可以一个连接发送多个请求。Server端按发送顺序处理。

# Part 2 性能监控

## 2.1 Timing

### 2.1.1 FireFox



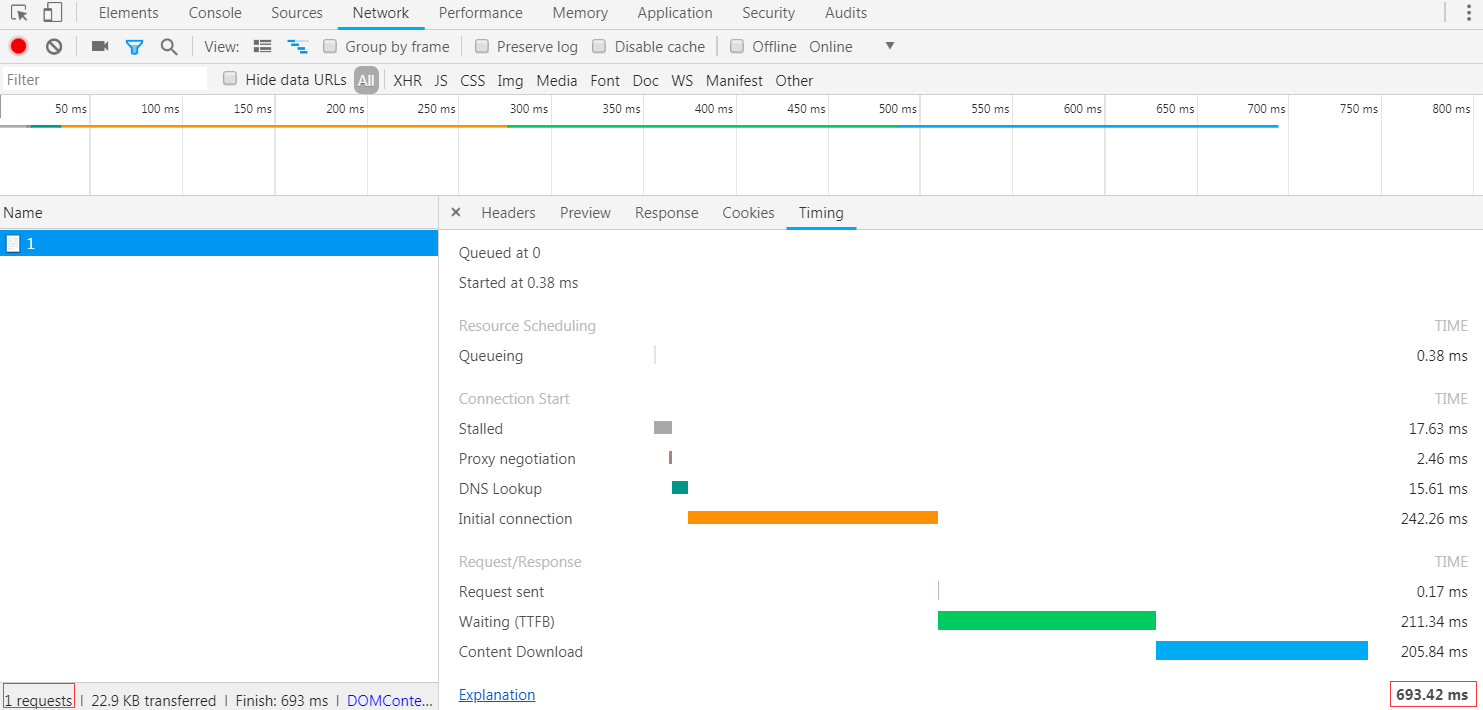
Connecting:TCP建立连接时间

Waiting : Waiting for a response from the server.

Receiving :read entire response from server

### 2.1.2 Chrome

[Reference](https://developers.google.com/web/tools/chrome-devtools/network-performance/understanding-resource-timing)



Queuing：阻塞的原因

Finish： timeline里所有request总时间

DOMContentLoaded ： 加载html dom内容，不包括js，css等

Load： 整个页面的时间

## 2.2 DB

连接

mysql –uroot [-h slc11fsp] -p

### ProcessList

SHOW FULL PROCESSLIST



每一条代表一个thread，一个tcp连接

### Lock

1.show status like '%Innodb\_row\_lock%

统计



Innodb\_row\_lock\_current\_waits 当前阻塞线程数

Innodb\_row\_lock\_waits 总共wait次数

2.select \* from information\_schema.INNODB\_LOCKS;

查看lock具体数据



lock\_trx\_id找到事务

lock\_table+lock\_data找到具体锁定的数据。

Lock\_mode: X/S

lock\_type： RECORD行锁。

### Transaction

#### View

开启事务 start transaction;

SELECT \* FROM information\_schema.INNODB\_TRX;





trx\_id

trx\_state: RUNNING/LOCK WAIT

trx\_stared

trx\_wait\_stated:block事务wait stated

trx\_query:

trx\_rows\_locked 锁定row行数

trx\_tables\_locked 锁定table数量

select trx\_id, trx\_state, trx\_started, trx\_wait\_started, trx\_query, trx\_rows\_locked from information\_schema.INNODB\_TRX;

#### Autocommit



show variables like 'autocommit';

set autocommit = 0;仅对当前连接生效

autocommit VS start transaction

[Reference](https://blog.csdn.net/qq_26941173/article/details/77872038)

set autocommit=0 所有语句自动开启事务，commit结束事务。  
start transaction 启动一个新事务。

 START TRANSACTION语句"挂起"自动提交模式

#### ReadOnly

查看 show global variables like "%read\_only%";

设置

#### Authority

grant select,insert,update,delete on s18.\* to 'test'@'127.0.0.1' identifi

ed by '123456';

## 2.3 TOMCAT

Server

超过最大连接数，响应时间变长

最大连接数越大，cup load约高

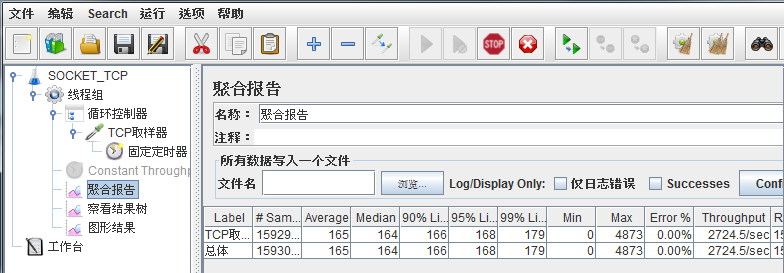
线程数越大，超过500，响应时间变长。线程切换时间？

吞吐量是否应该在服务端测，排除传输时间干扰。

# Part 3 性能测试

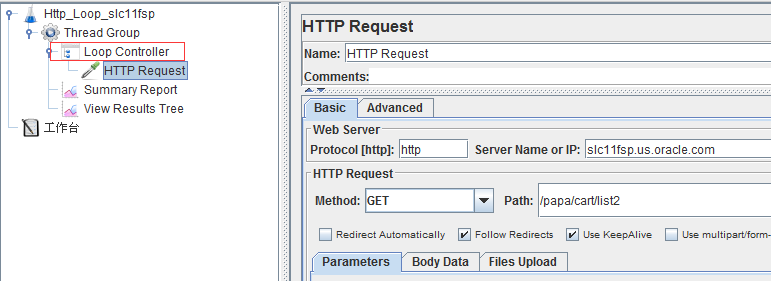
## 3.1 JMeter

### TCP



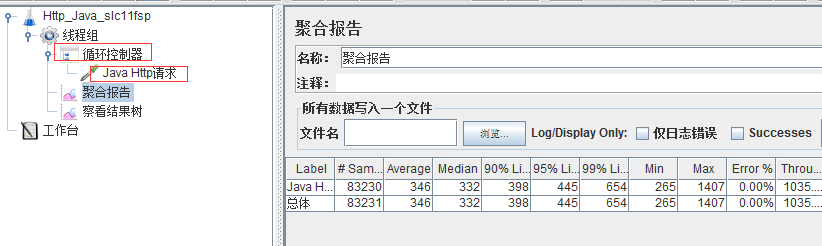
### HTTP

短连接



此处是自带的HTTP Request，虽然Loop，但每个request都会重新建立TCP连接，使用完close。

长连接



java中不要close连接

## 3.2 JLoad

Report

BIO 模式

ThreadNum：800

Samples:1186769,finish:253813 s,Average:213 ms,Min:0 ms,Max:6480 ms,Throughput:3553,Error:0

Nio模式

ThreadNum：800

Samples:1064169,finish:238516 s,Average:224 ms,Min:0 ms,Max:8231 ms,Throughput:3367,Error:0

# Part 4 系统优化

### 前端优化

* 合并css、js，减少Http请求
* 启用压缩，减少数据传输
* 使用浏览器缓存。生成新的js时，只需更新Html中引用。
* CSN加速：缓存静态资源，如图片、文件、CSS、JavaScript脚本、静态网页等。
* 反向代理

### 应用服务器

系统架构

* 缓存：加快数据读取，降低DB访问压力。网站遇到性能瓶颈，第一个想到的解决方案
* 异步操作：加快请求响应及实现削峰。
* 使用集群：使用负载均衡，有更好的响应延迟，提高吞吐能力。

代码优化

* 多线程,最大发挥CPU和处理IO阻塞。
* 使用NIO(非阻塞，基于事件驱动)
* 资源复用，比如单例和线程池。
* 合理的数据结构
* 减少垃圾回收。
* 尽量不使用锁,减小锁的力度。