性能测试执行策略与压测模型浅析

姓名:焦晋科

职位:测试开发工程师

部门:AI平台与研究部-AI平台部-智能对话研发部-质量部

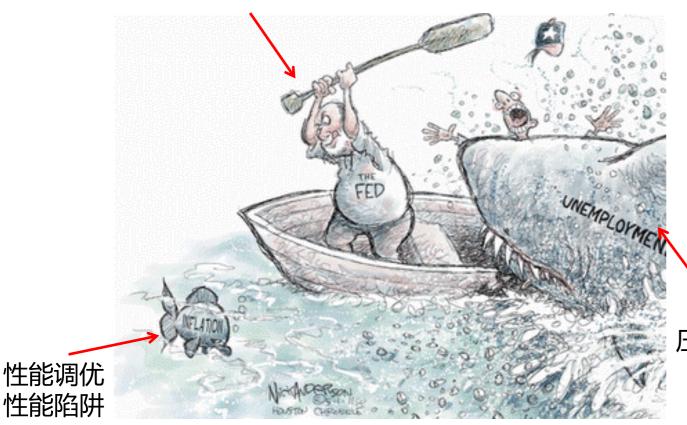
日期:2017.12.25



问题1:如何保证压测权威可靠



测试&开发



问题来了:

怎样保证压测权威、可靠? 警惕:

压测策略不当导致的误判

压测本身

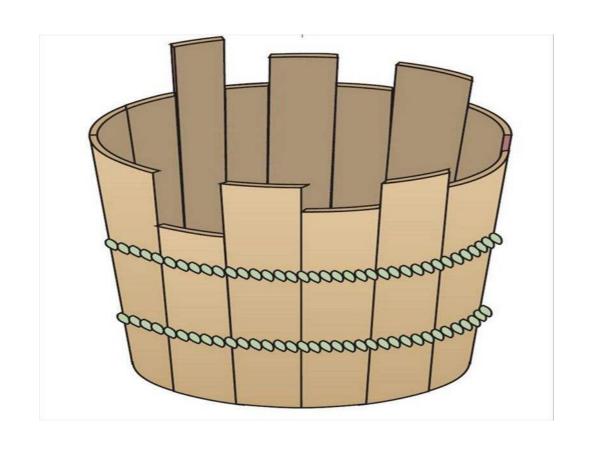
问题1:如何保证压测权威可靠



剑走偏锋: 谈谈压测自身, 执行策略与压测模型

可靠性的木桶效应





- ●发压模型
- ●脚本模型
- ●数据模型
- ●执行策略

| 发压模型 | |
|-------------|--|
| 脚本模型 | |
| 数据模型 | |
| 执行策略 | |

发压端

- 成熟可靠的发压工具
- 一问一答的模型
- 发端同步统计压测指 标(TP99,TPS)



待测对象

● 单端(仅发压端)



- 简单可靠● 适用于一问一答模型



● 不适用于复杂压测场景

问题2:如何对IM系统进行压测



IM系统:非一问一答,涉及多端

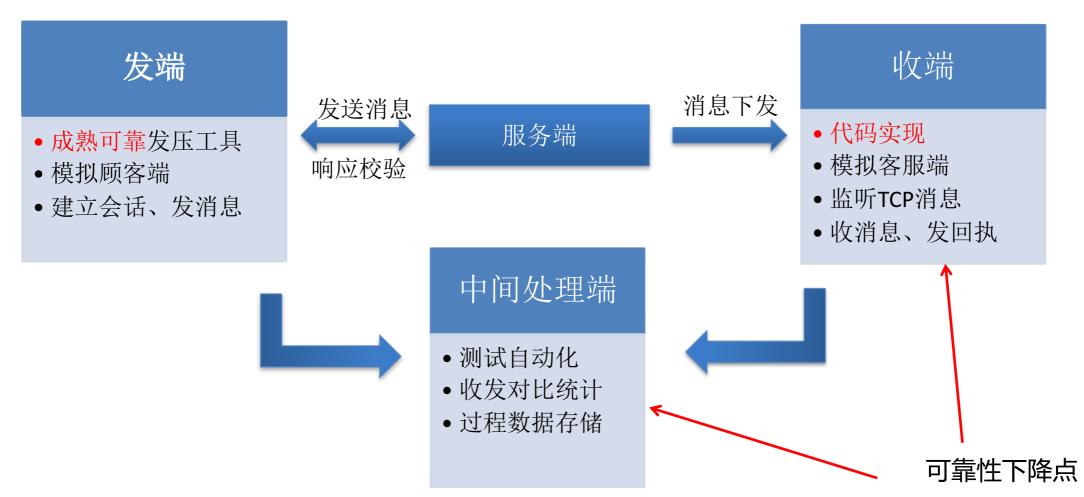


此时, 传统的单端模型已不再适用

多端压测模型



由于IM业务特殊性,典型的性能测试方案包括三部分:发端、收端、中间处理端。



组件越多,可靠性越差。

压测中的智:发压模型对比



单端压测

多端压测

简单可靠

不适用于复杂 业务场景

适用复杂场景

实现难度大

可靠性不强

| 发压模型 | |
|-------------|--|
| 脚本模型 | |
| 数据模型 | |
| 执行策略 | |

问题3:IM的大促压测如何开展



首先需明白,大促压测与日常线下压测有何不同:

| 不同点 | 大促压测 | 线下压测 |
|------|-------------------------|-------------|
| 压测目的 | 流量指标验证 | 指导性能调优 |
| 关注点 | 同时关注于会话量(UPV) 与消息量指标 | 可仅关注特定指标 |
| 脚本模型 | 单线程执行动作较多 | 单线程执行动作较为简单 |
| 时间制约 | 仅可凌晨执行 | 不受制约 |

IM大促压测:压测流程



M端,WEB端,SDK端顾客侧综合场景,覆盖商家、京东客服。 流程为:

- a. 客服登陆,保持在线(各终端通用)
- b. 顾客登陆/初始化
- c. 顾客建立会话/发送一条消息
- d. 顾客发送心跳/poll
- e. 顾客评价
- f. 顾客连续发送N条消息,消息校验

其中b-e为一个事务/PV,压测时,应首先保证事务达到预期TPS。一个事务中,任何一个请求失败,均认为事务失败。

f中的N需在保证事务量的情况下,尽量逼近预期消息量。

需要注意,b为后续步骤前置条件,若b失败,则c-f无需执行。

压测中的智:大促压测策略 v1



发端(顾客)

- Jmeter实现,分布式
- 发起TCP/HTTP请求
- 建立会话后,连续发送N条消息
- 消息打印日志



服务端

消息下发

收端(客服)

- python/Java实现
- 监听TCP消息
- 接收、打印消息、 发回执



中间处理端

- shell脚本实现
- 通过对比收发两端日志找出丢失、 重复消息

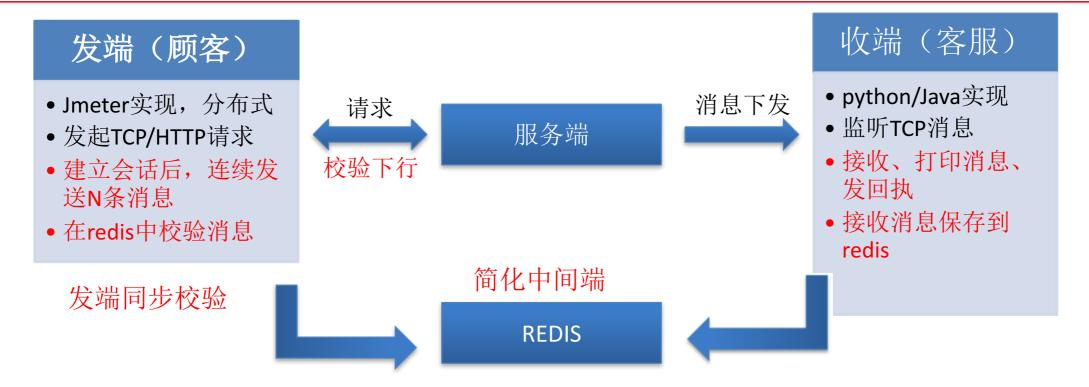


- 非实时
- 消息投递时长信息丢失
- 业务指标获取不可靠



压测中的智:大促压测策略 v2







- 实时,延时信息准确● 发端可获得业务指标

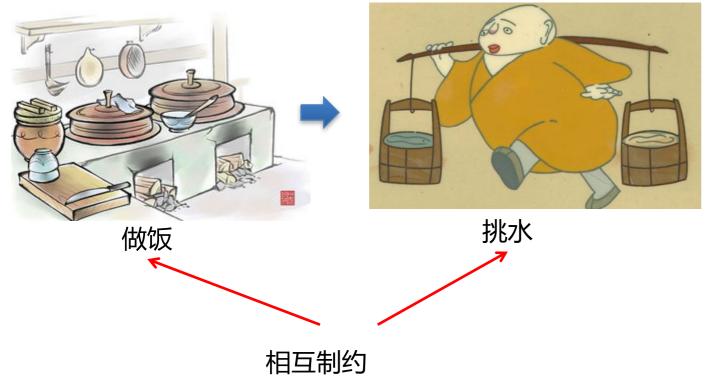


受制于水桶效应,会话数 与消息量无法同时上升, 很难达到上限

问题4:为什么会话数与消息量无法同时上升

和尚挑水做饭问题:

- 和尚1天可做3顿饭
- 和尚1天可挑9担水
- 寺院要求3顿饭和9担水同时满足

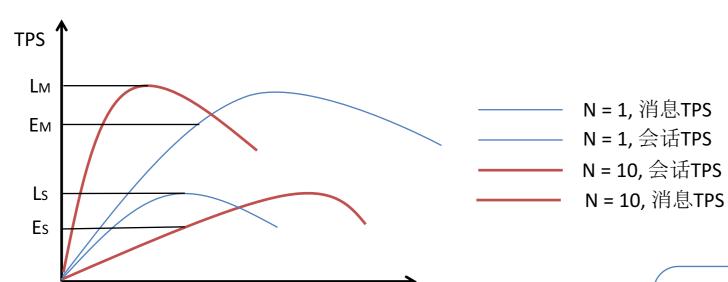


即使和尚做完饭,马上连续挑3担水,则显然也无法达到目标。

同步校验压测模型的致命缺陷

JD.COM 京东

IM大促压测:同时关注会话与消息指标,一个会话中发送N条消息



LM,消息TPS上限 EM,消息TPS预期

Ls,会话TPS上限 Es,会话TPS预期

现状:

- 绝大多数时间进行参数的调试;
- 慢请求拖累压测,无法达到上限 在双11时,同步压测模型几乎不可用

假设会话TPS先达到预期值,此时如何调节:

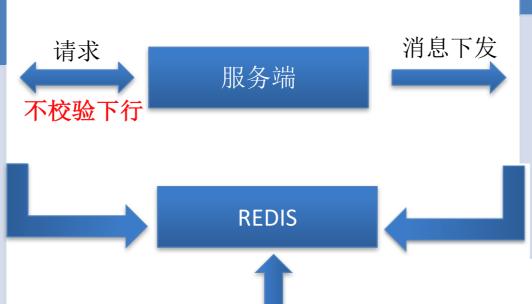
- 增加VU(线程数),则当消息TPS达到预期时,会话TPS可能已经超过了上限值;
- 增大N的值,则当消息TPS达到预期时,会话TPS可能远未达预期;

VU

压测中的智:大促压测策略 v3 异步模型

发端(顾客)

- Jmeter实现,分布式
- 发起TCP/HTTP请求
- 建立会话后,连续发 送N条消息
- 消息内容中添加时间 戳
- 消息保存在redis



收端(客服)

- python/Java实现
- 监听TCP消息
- 接收、打印消息、 发回执
- 解析接收消息
- 异步保存接收时间 戳至redis

统计放在中间端, 牺牲可靠性



- <mark>异步校验</mark>,并发量易 调节
- 业务指标定制程度高

统计端

- Java实现
- 扫描redis中所有 消息,统计得到 TP99等信息



- 牺牲部分可靠性
- 指标获取非实时

提升压测可靠性



警惕收端成为性能瓶颈

- 收端分布式部署
- 收端线程池使用优化;
- 使用NIO;
- 某些IO操作时,可先写入内存,然后由异步线程来执行IO。
- 参数化测试数据,降低压测脚本业务耦合度

压测中的智:脚本模型



发端同步 校验

异步校验

简单可靠

整体性能受限 于水桶效应

并发大, 易于 达到压测上限

实现难度大

可靠性不强

| 发压模型 | |
|-------------|--|
| 脚本模型 | |
| 数据模型 | |
| 执行策略 | |

数据来源



- 最理想的压测数据模型: 线上真实流量
- 测试数据应尽量趋近真实流量

| 数据来源 | 说明 | |
|----------|------------------------|--|
| 构造数据 | 构造模拟测试数据,某些数据可MOCK | |
| TCP-COPY | 复制线上真实流量数据,如真实顾客、真实消息等 | |

问题5:为什么TCP-COPY不适用于IM系统

真实数据

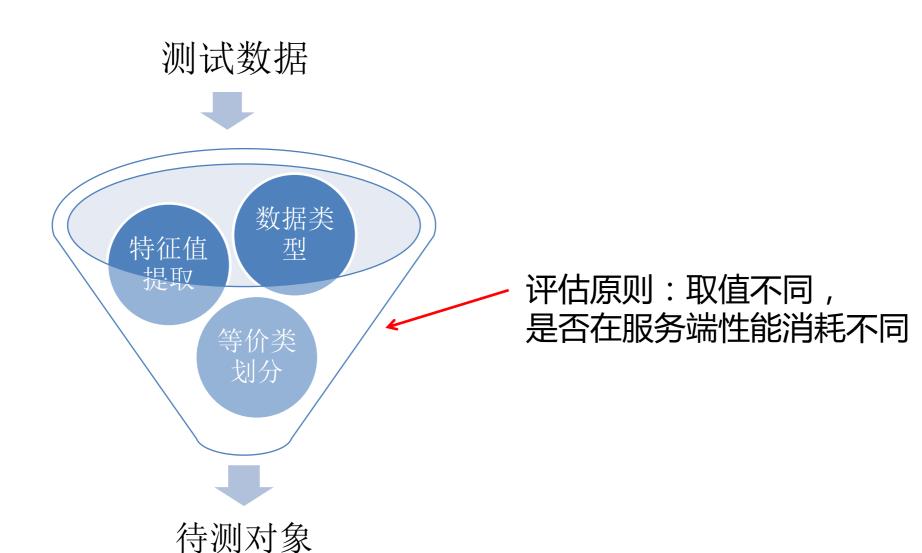


IM系统

业务约束多:

- 角色需登录
- 先建立会话,后发送消息
- 先建立会话,后评价

- 无真实密码
- 脏数据难以处理
- 易引发投诉,风险
- 不易扩展



问题5:如何准备IM压测数据





IM系统

- 分配客服
- 会话管理
- 消息转发



客服

- 顾客账号
- 顾客终端
- 咨询入口
- 咨询参数

非特征值

特征值

特征值

特征值

- 客服账号
- 客服类型
- 客服终端
- 客服组织架构
- 最大接待量

非特征值

特征值

非特征值

特征值

特征值

压测中的智:数据模型



构造数据

TCP-COPY

灵活,扩展性 强

数据真实可靠

需保证可靠性

可扩展性差

适用于复杂模型

不易善后处理

| 一 发压模型 | |
|---------------|--|
| 脚本模型 | |
| 数据模型 | |
| — 执行策略 | |



压测模型已准备就绪,不代表压测不会栽跟头

压测执行策略



• 理想的执行策略: 单次压测执行彼此独立, 无状态, 无干扰

• 无法绝对独立时:分析状态量,降低干扰



- 会话: 单对单 顾客账号分批次
- 风控、黑名单 临时关闭风控

• 客服接待量

足够大



| 出发点 | 结论 |
|------|---|
| 发压模型 | 单端压测最可靠,多端压测应用广但可靠性略低 |
| 脚本模型 | 同步压测最可靠,异步压测发压灵活但可靠性略低 |
| 数据模型 | 理想:尽量<u>趋近真实</u>流量模型基于特征值的压测数据构造方法 |
| 执行策略 | 理想:压测执行彼此独立,无状态,无干扰分析状态量,降低干扰。 |



有得必有失,应根据压测目的不同制定最适合的压测方案。



Thanks!

AI平台与研究部-AI平台部-智能对话研发部-质量部

