TiD 2017
质量竞争力大会

超大规模性能测试的云端方案及案例分享

臻云科技 任党恩

2017/07/19

下一代软件研发
SOFTWARE DEVELOPMENT

TiD2017

自我介绍

任党恩　臻云科技

大规模性能测试的云端SaaS和企业性能测试解决方案咨询

曾在IBM中国开发实验室工作16年
- WebSphere中间件系统测试架构师
- Linux技术中心高级程序员
- Rational全球系统及性能测试架构师

应用系统的性能挑战

新技术的发展

- 移动、物联网时代，接入的人与设备数目大幅增加
- 系统架构的复杂度增加（云、大数据、微服务等）

使用场景的碎片化

- 随时随地

持续交付和敏捷开发模式

- 快速迭代和交付要求
- 性能测试难以快速有效实施

TiD2017

若干测试解决方案

以运营代测试：灰度发布（快速上线、验证业务）

- 偏重验证业务，而非技术
- 发现问题晚，代价高

在线流量复制回放：TCPCopy/Gor

- 生产环境，风险高
- 接口更改，无法复制流量

传统性能测试工具：LoadRunner, JMeter等

- 小规模：最多模拟几千虚拟用户
- 低效率：手工准备和维护测试环境，数天、甚至几周完成性能测试
- 高成本：独占，人工管理计算资源；收费高

如何应对：

高并发的压力发起，海量测试结果/日志的实时处理和展现?

TiD2017

性能测试云服务

测试服务逐渐转向云端

- 移动兼容性测试：各种设备云方案
- 功能自动化：Sauce Labs等
- 其它：接口测试,安全测试, 持续交付流水线服务 ..

性能测试@云平台

- 云资源的弹性契合不同规模的性能测试
- 公网发起测试，较之内网测试更加真实，也可突破本地带宽限制
- 测试实施更为便捷，过程标准化，效率提升
- 测试机器无需人工配置和管理！→ 主要精力用于测试开发，数据分析和调优
- 按需使用，大幅节约成本

SOASTA

BlazeMeter

Cloudwise 压测宝

ONE APM

XMeter

TiD2017

构建云端性能测试方案

基本原则

- 异构IaaS之上的自服务性能测试平台
- 流行开源组件的高效集成 + 自主的调度和编排系统
- 消息驱动的分布式处理，消除单点，高度可扩展
- 支持大并发，功能完善，支持持续集成测试
- 标准化操作流程，辅助分析和结果呈现
- 完全自动化，高性价比
- 易于使用和维护

TiD2017

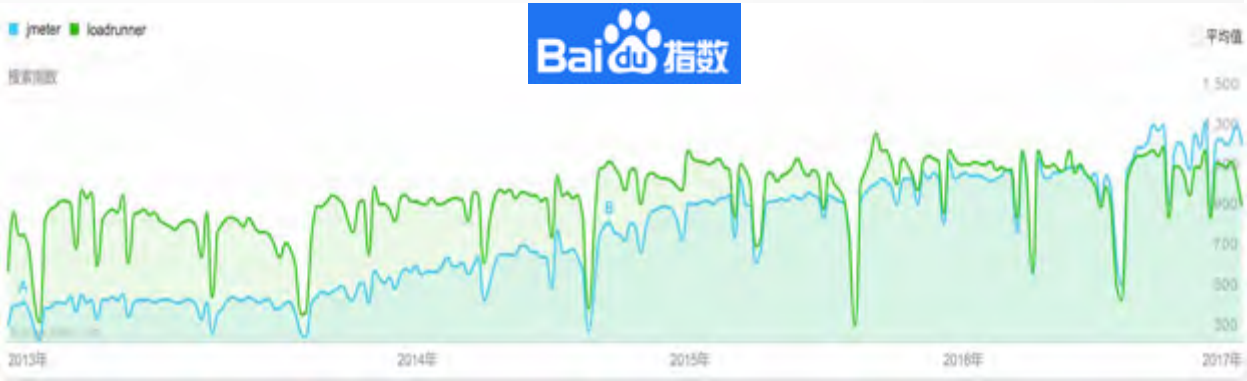
测试执行引擎的选取

考虑下列因素

- 在社区中具有一定影响力
- 架构开放可扩展，易于集成
- License友好

我们的选择：Apache JMeter

- 最流行的社区性能测试工具
- 可扩展协议支持，数据处理等
- Apache license



TiD2017

JMeter的问题

不支持大规模并发

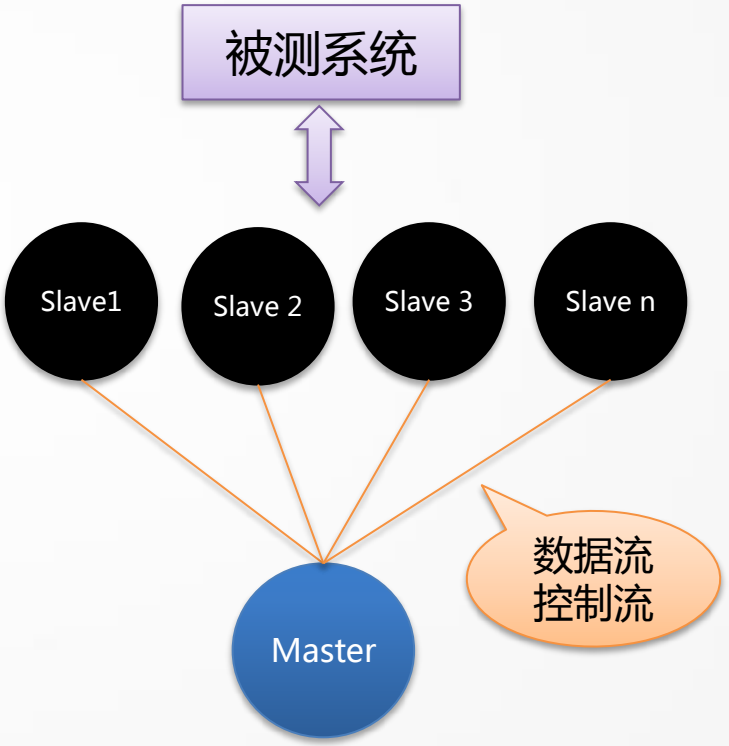
- 典型Master/Slave单点结构，无法扩展
- 测试结果数据处理瓶颈

测试报告/被测系统监控功能不完善

无法方便地做持续集成

只是工具、非平台、非解决方案

- 大量时间做测试环境准备
- 无法管理测试脚本、测试结果



改造JMeter，去中心化

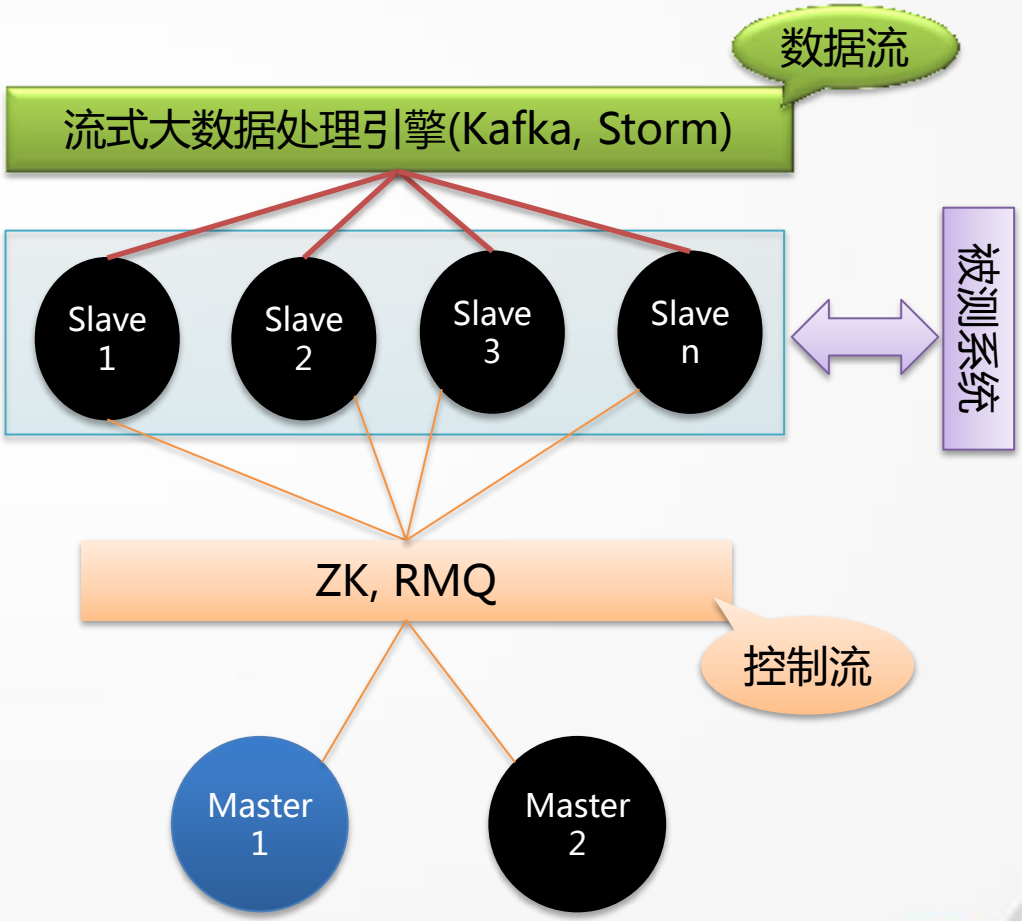
去中心化、分布式、容错

实时的流式结果处理

- JMeter Slave定期汇报测试结果数据
- Kafka+Storm并行处理
- 也可基于ZMQ自行实现

效果

- 完全水平扩展，支持超大规模
- 容错机制，系统更加健壮
- 双Master，互为备份



TiD2017

若干组件说明

Zookeeper

- Master leader选举、Slave发现 / 断开
- 测试任务分发：各Slave根据自己状态申请测试任务
- 分布式锁（Slave任务申请和状态更新等）

RabbitMQ

- 控制消息转发：各Slave测试状态报告、Master控制消息分发、测试环境状态更新
- 可水平扩展

Kafka / Storm

- 测试结果（响应时间，返回码和验证点成功）转发和持久化，避免测试结果丢失
- 多测试结果同时分析
- 测试结果保存至时间序列数据库InfluxDB
- 可水平扩展

容器化的JMeter

- 发送测试状态、接受Master控制指令
- 下载测试脚本以及动态更改分配的虚拟用户数目、变量替换、结果数据发送改造等

TiD2017

分布式组件的功能验证

复杂的系统交互：zookeeper、Redis、RMQ

复杂的应用逻辑：分布式应用带来的不确定性：Slave断开、超时、RMQ消息无法转发等

带来的问题

- 手工测试场景无法覆盖所有的情况
- 回归测试非常困难

解决办法：分而治之

- 需要与外部组件交互的功能进行接口抽象：单元测试的时候进行mockup，保证覆盖到所有分支
- 复杂逻辑的代码避免与外部系统进行交互，将需要的数据从方法参数传入，可以对这部分逻辑进行完整的单元测试

效果

- 近百个单元测试用例，对复杂的应用逻辑完全覆盖

TiD2017

测试环境的管理

根据用户的测试请求，调用云服务商API：

- 自动创建机器，网络（网络隔离）、GRE隧道等
- 自动创建自动绑定公网IP、分配带宽
- 分层架构：适配各种不同的云提供商，避免对上层应用产生太大影响
- 异构的云基础设施：消除计费方式不统一对应用产生的影响

实现方式从Shell替换成Python

- 控制更方便、功能更强（锁）
- 准备环境过程中需要通过RMQ报告状态

加快环境准备速度：环境预留

- 完全从头开始创建环境费时，采用环境预留缩短用户等待时间（等待时间缩短2/3）
- 环境是否回收、保留，利用ZK分布式锁

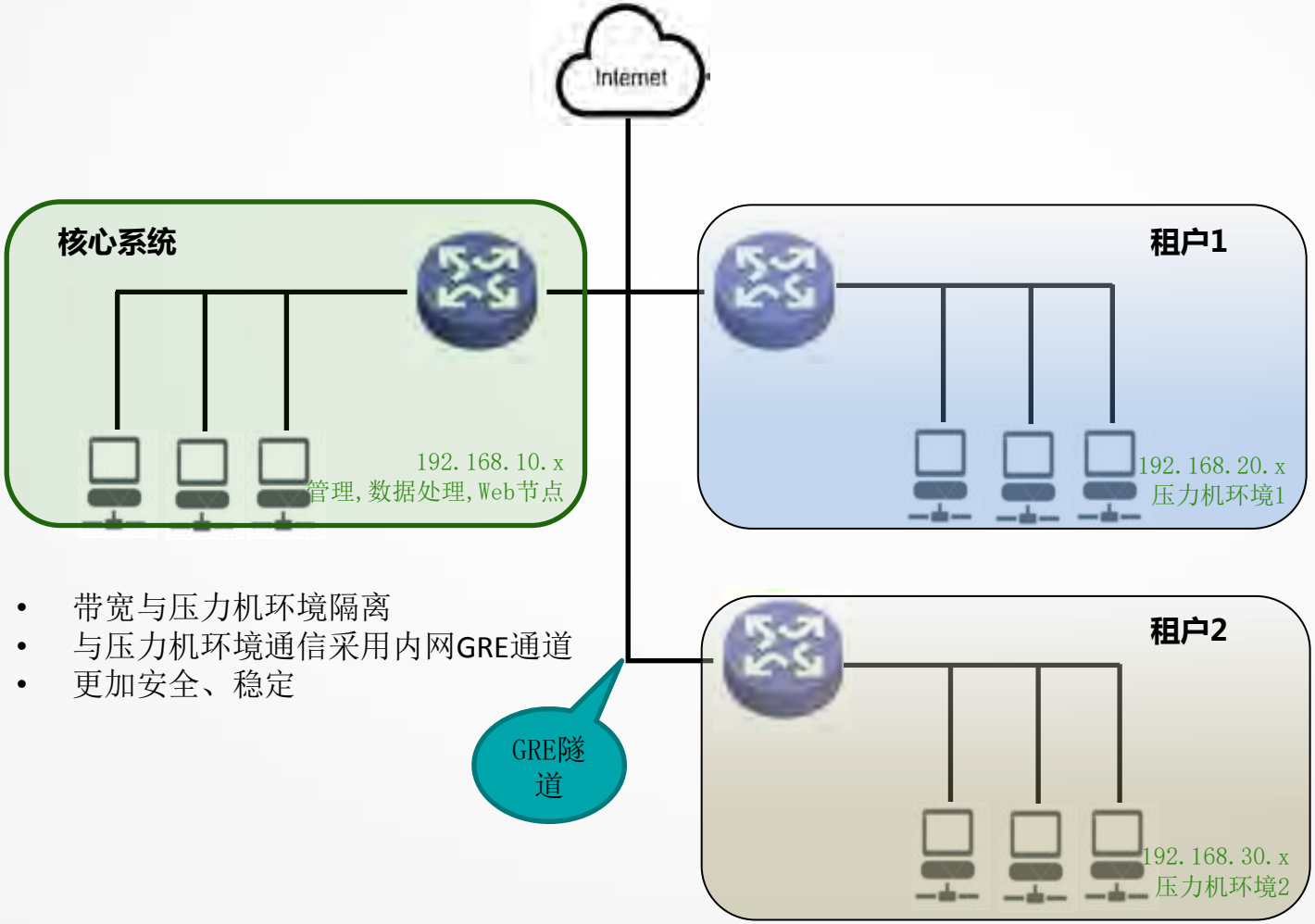
容器化

- JMeter执行单元运行于docker中，方便部署和维护

效果

- 即取即用：无需管理测试机器、大幅节约使用成本
- 环境隔离、安全
- 分钟级创建测试环境（公有云）； 秒级创建测试环境 （私有物理机器）

云端多租户环境隔离



所有测试资源自动创建/销毁

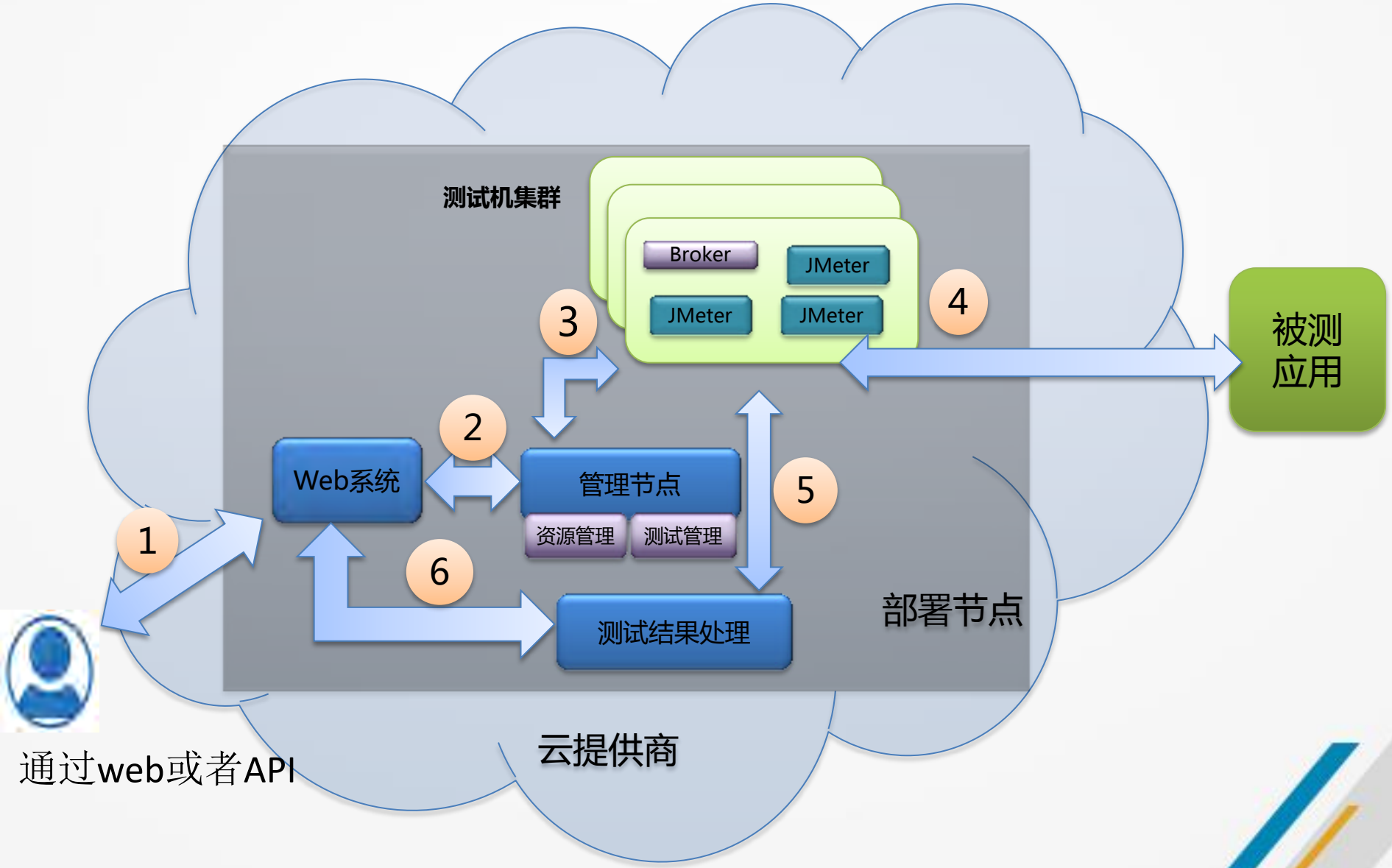
- 虚机
- 子网络
- 路由器
- GRE通道
- 公网IP地址，带宽

- 异构多站点支持

- 带宽与压力机环境隔离
- 与压力机环境通信采用内网GRE通道
- 更加安全、稳定

TiD2017

系统总体架构拓扑图



TiD2017

其它功能的完善

在线管理测试用例和结果

- 接口测试（REST API）
- ReactJS

完善监控系统

- 被测系统监控

  利用Grafana/InfluxDB/Collectd对被测系统实施监控

  提供扩展机制允许加入第三方中间件的监控

- 对测试发起机的监控

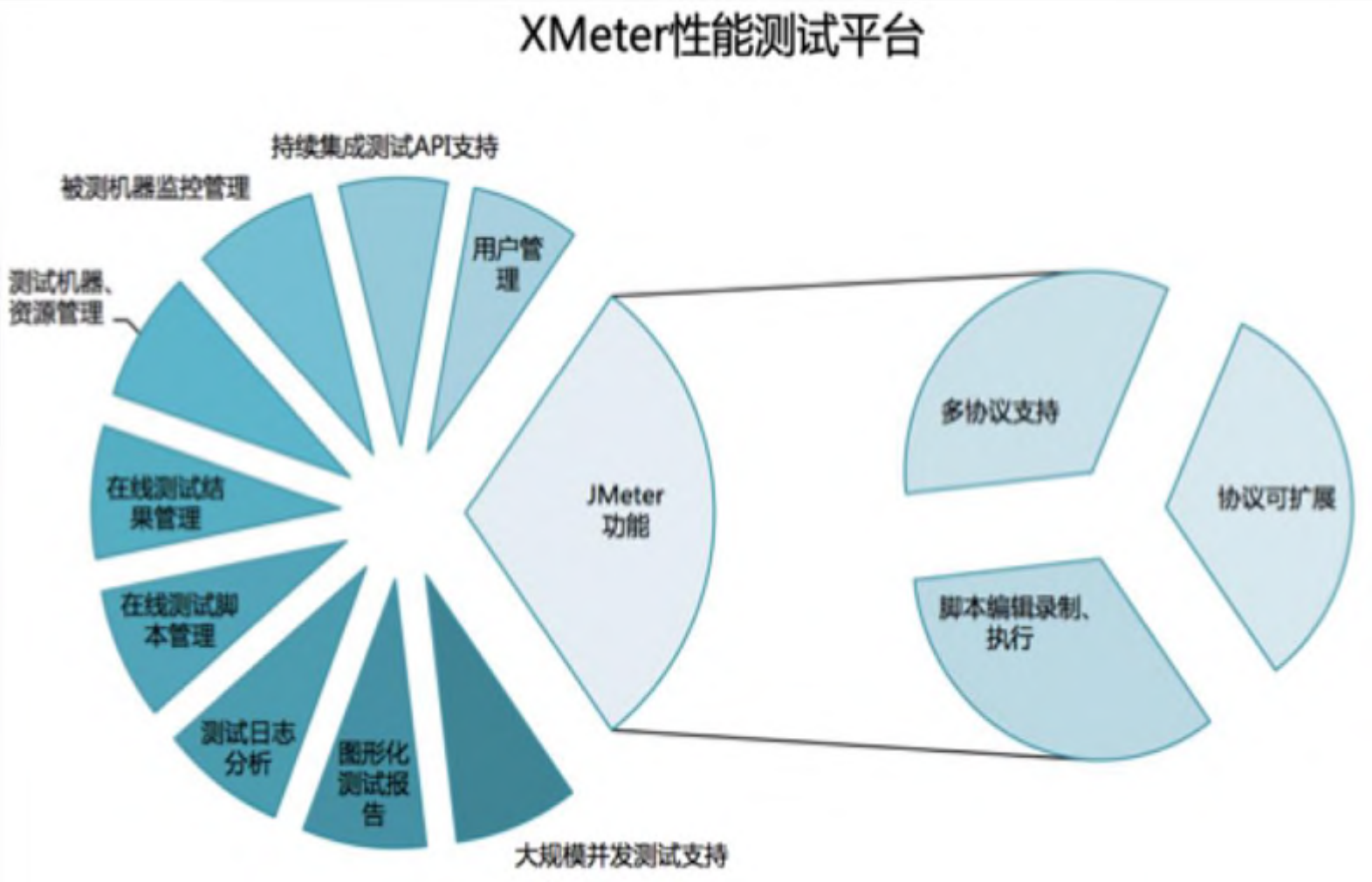
  避免压力发起机器工作负载过高导致对测试结果有影响

API支持

- 常用功能暴露REST API，提供性能测试持续集成

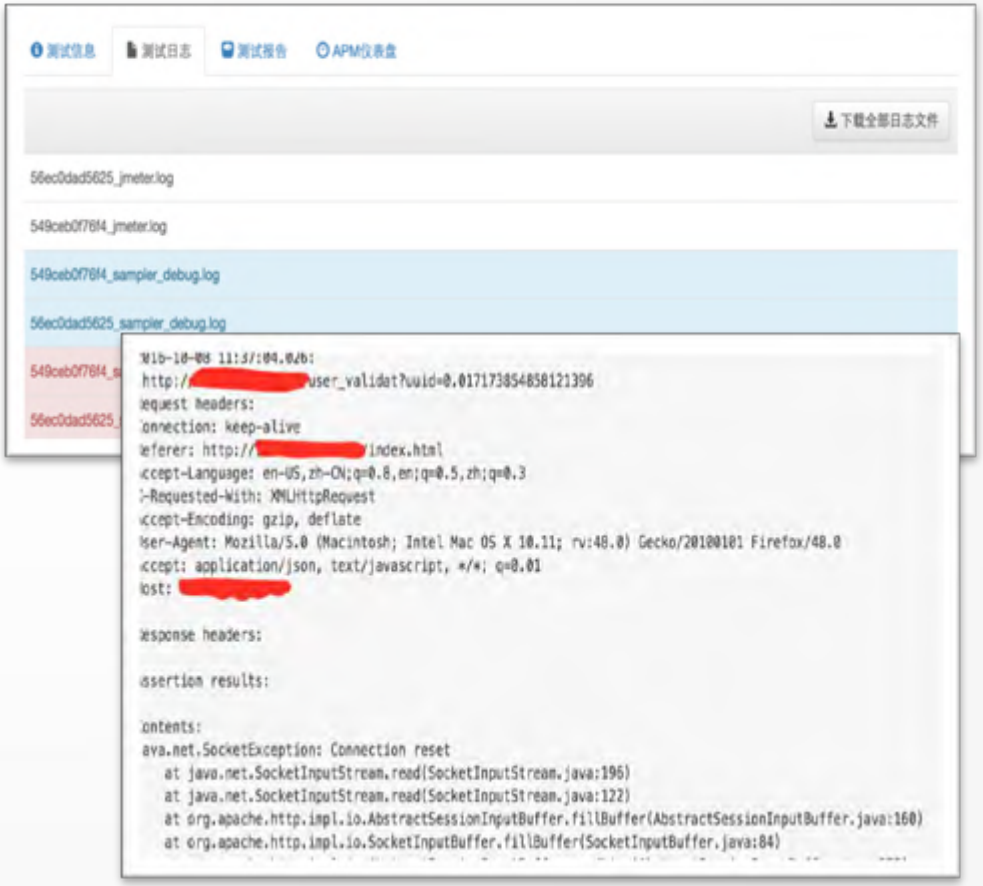
TiD2017

性能测试平台的附加价值



TiD2017

系统实际效果图

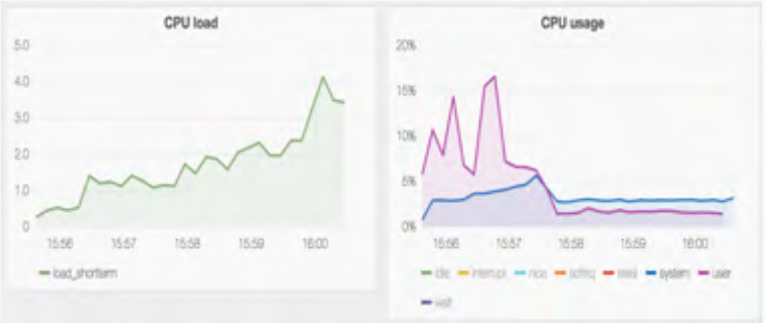


示例：性能度量指标分析

平均吞吐量	最大虚拟用户数	平均响应时间	最大响应时间
306	500	1.52 s	1.55 min
最小响应时间	**请求响应码成功率**	**验证点成功率**	**平均请求大小**
4.00 ms	100.00%	100.00%	5.71 KiB

计算实际的带宽占用

平均吞吐量	最大虚拟用户数	平均响应时间	最大响应时间
1018	500	481.47 ms	29.90 s
最小响应时间	**请求响应码成功率**	**验证点成功率**	**平均请求大小**
3.00 ms	100.00%	100.00%	5.71 KiB



测试机的资源占用

TiD2017

某省在线司法学习和考试系统

- 项目背景
- 合理的负载模型
  - 机考：集中登录，做题，集中交卷
  - 背景负载：自学课程
- 硬件升级 （阿里云 主机配置，带宽）
- CPU，磁盘IO高：频繁写日志，开发控制日志级别
- 网络带宽占用过高：调整带宽，响应内容压缩
- 并发用户上不去：调整nginx连接配置，tomcat线程池，jdbc连接池
- 响应时间长：SQL优化，表拆分，redis缓存(题目)，异步判卷
- 水平扩展：应用服务器（2台->4台），数据库服务器(2台->3台，读写分离)

经过3周的测试和调优，并发用户数从100提升到5万

TiD2017

MQTT百万连接测试

http://emqtt.io/ 开源社区最流行的MQTT中间件

- 实施EMQ基准性能测试，共计将近30个测试场景，20pd完成所有工作

- 开放性：定制开发开源的JMeter的MQTT插件 https://github.com/emqtt/mqtt-jmeter/
- 节约成本：测试时间从2周下降到1个小时，大幅降低测试所花成本
- 持续集成：XMeter测试服务作为将来版本发布的质量保证
- 易于定位问题：定制后端报告模版，实时监控EMQ服务器表现
- 样例测试报告

  中文入口 - https://github.com/emqtt/emq-xmeter-cn

  英文入口 - https://github.com/emqtt/emq-xmeter-en

  连接测试报告 - https://github.com/emqtt/emq-xmeter-cn/blob/master/connection.rst

  吞吐量测试报告 - https://github.com/emqtt/emq-xmeter-cn/blob/master/throughput.rst

TiD2017

MQTT测试实践

JMeter插件消息时延计算（异步系统时延计算）

- Pub / Sub机制，由于异步转发消息，Sub端无法得到消息发送时间，无法计算时延

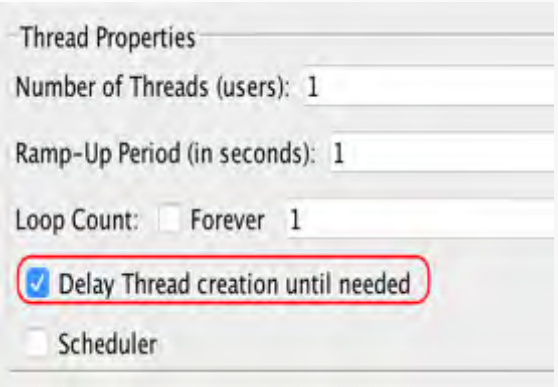
  解决方法：Pub在发送消息的时候带上所在服务器的发送时间戳
- 测试机的时间同步

系统配置更改

- 进程资源，打开文件数，线程数等：ulimit
- 连接重用设置 - net.ipv4.tcp_tw_reuse

JMeter脚本设置

- 在必要的时候创建线程



服务器运行的虚拟用户数和带宽进行调整

- 可针对不同的测试脚本调整参数

2017中国企业
敏捷实施情况调查

中国敏捷实施现状如何？
企业敏捷实施过程中会遇到哪些困难？
哪些实践是比较普遍的？

如果你已经开始实践敏捷，请扫码参与调查。同ThoughtWorks一起，为中国敏捷行业打造这份权威报告。

您需要一份权威报告！

深入了解敏捷趋势、最佳实践和经验教训，
帮助你成功实现敏捷转型。