性能测试体系（邱功圆）

- 进阶
  - 高并发
    - HTTP缓存
    - 应用级缓存
    - 多级缓存
    - 池化
    - 异步并发
    - 扩容
    - 队列
  - 高可用
    - 负载均衡
    - 限流
    - 降级
    - 隔离
    - 超时重试
    - 回滚
    - 压测预警
  - 业务/产品模型梳理
    - 确定业务访问总量、业务种类和业务占比
    - 根据不同时段业务访问建立对应业务模型
    - 确定业务对应资源消耗
    - 确定突变业务（大促或者活动）对整体业务影响
    - 线上问题分析改进业务模型
  - 性能测试平台
  - 监控平台
    - 推荐使用Open-Falcon、zabbix
    - APM应用性能监控平台实时监控和管理应用软件的性能和可用性是以后趋势
- 技术知识体系
  - 测试环境部署
    - 测试/生产环境架构一致
    - 测试/生产环境软件版本一致
      - 操作系统、中间件、数据库、应用
    - 测试/生产环境参数配置一致
    - 测试/生产环境数据量级一致
    - 测试环境相对生产可等比缩小
  - 性能测试工具
    - loadrunner、jmeter、 Web Bench
    - Grinder tsung、kylinpet
  - 性能监控
    - 中间件
      - 操作系统层监控
        - nmon、top、iostat、free等
      - 前端性能测试工具
        - YSlow、PageSpeed、Httpwacht
      - 中间件性能监控
        - Profiler工具分析代码
          - 工具
            - Java
              - jvisualvm、JProfiler、MAT
            - C/C++
              - Linux
                - GProf
                - OProfile
                - CallGrind
                - vTune
              - Windows
                - Visual Studio
                - vTune
                - DevPartner
            - .NET
              - Visual Studio
              - DevPartner
          - 过程
            - 执行多次，查看CallTree
            - 分析占用时间比例最高的若干调用
      - 数据库监控
        - spotlight
        - awr
        - slow
      - 监控平台
        - zabbix、nagio、cacti、open-falcon
  - 性能分析
    - 一般常识
      - CPU、TPS曲线图存在明显波动则存在瓶颈
      - 并发时服务日志级别需调整为ERROR级别
      - 通常一个请求由一个线程负责执行，占用一个逻辑CPU
      - 若并发量增加而CPU使用率未增加则存在瓶颈
      - CPU负荷集中在应用服务器和数据库服务器上
      - 内存负荷集中在应用服务器和数据库服务器上
      - 磁盘负荷集中在数据库/文件服务器上
      - 对外网络流量集中在负载均衡器（nginx、LVS）上
      - 响应时间（参考）
    - 环境检查
      - 目的
        - 确认系统是否是最优配置，否则测试数据不可靠
      - 操作系统参数检查
      - 中间件
        - 版本
        - 内存大小
        - 最大连接数
        - 工作线程数
        - DB连接池大小
        - 缓存大小
      - nginx、LVS
      - 数据库
        - 连接数
        - 内存
        - 其他参数优化
    - 瓶颈分析
      - 实际响应时间数据收集
        - 分别用不同并发路数进行测试，收集实际数据，进行比较
        - 若实际响应时间明显大于以上估算，则需要分析
      - 根据监控分析硬件资源使用情况
        - nmon
        - top
        - free
        - iostat
      - 建立诊断分析经验库
      - 分层诊断分析
        - 前端性能分析
          - YSlow、PageSpeed、Httpwacht
        - 应用诊断
          - 应用隔离
          - 代码性能分析
        - 数据库
          - 数据库SQL语句性能分析
            - 执行计划
            - 慢查询日志
        - 缓存
  - 性能优化
    - 硬件资源优化
    - 参数优化
      - OS
        - 最大连接数
        - 最大打开文件数
        - 连接重用、快速回收等
      - 中间件
        - 工作进程数、线程数、数据库连接数、内存等
    - 代码优化
      - 异步
      - 对高频率的调用进行合并
      - 缓存
      - 使用更高效的机制或算法
      - 减少日志输出
    - 数据库优化
      - sql优化
        - 根据执行计划进行优化
        - 小结果集驱动大的结果集
        - 只取出自己需要的数据（字段和行）
        - 使用合适的数据类型
        - 使用最有效的过滤条件
        - 避免使用函数
        - 避免使用 is null
        - 避免复杂的Join 和子查询（MySQL）
      - 大字段垂直拆分
      - 大表水平拆分
      - 表分区
      - 分库、读写分离
      - 缓存
    - 业务逻辑优化
    - 架构设计优化
- 能力体系
  - 性能测试理论体系
    - 性能需求分析
    - 性能指标
    - 性能测试环境
    - 性能测试业务模型
    - 性能测试数据量
    - 性能测试策略
    - 性能测试报告
    - 性能测试风险
  - 知识库
  - 培养计划
  - 人员规划
    - 专业划分
      - 性能监控研究员
      - 性能工具研究员
      - 前端性能研究员
      - 中间件性能研究员
      - 数据库性能研究员
    - 成熟度划分
      - 初级性能测试工程师
      - 中级性能测试工程使
      - 高级性能测试工程师
- 工作体系
  - 性能测试流程
    - 性能需求分析
      - 根据客户要求分析系统的实际性能需求，合理定义性能指标
    - 性能设计（方案）
      - 通过分析系统各项功能被执行的情况，进行针对性的设计，找到性能瓶颈，用尽可能低的硬件成本来满足系统性能需求
    - 测试用例设计
      - 针对常用功能
      - 估算各功能使用率
    - 基础（常用）参数优化
      - 目的：确认系统为最优配置，避免测试数据不可靠，重复测试（该过程容易被很多人忽视）
      - os、nginx、db、线程数、连接数、内存
    - 测试执行
      - 先对基本配置进行调整，将各种限制参数放到很大（最大连接数、工作线程数）
      - 单元性能测试
        - 对每种服务器单独测试
      - 集成性能测试
        - 从系统总入口进行整体测试
    - 性能分析与优化
      - 分析系统瓶颈并进行优化，降低响应时间，支撑更多的用户
    - 测试数据整理
      - 确认有无错误
      - 确认响应时间
      - 确认TPS
    - 达标判定
      - 规定用户数下的总体响应时间
      - 各服务器的负载情况
  - 性能测试产物
  - 性能测试规范
  - 性能工作概述
    - 立项
      - 确定合理的阶段性 市场目标，包括关键性能目标
    - 需求
      - 分析系统业务场景，定义性能需求，明确性能指标
    - 架构设计
      - 性能设计（软硬件技术选型，实现方式设计，性能扩展机制设计）
      - 性能验证
    - 开发
      - 采用良好的数据结构和算法来实现架构设计
      - 评审代码的性能问题
    - 测试
      - 验证系统总体性能
      - 评估、优化、提供扩展建议
    - 运维
      - 监控性能，必要时进行扩展