

混沌工程



成熟度等级

成熟度可以反映出混沌工程实验的可行性、有效性和安全性。成熟度的级别也会因为混沌工程实验的投入程度而有差异。这里将成熟度等级分为1、2、3、4、5级进行描述，等级越高成熟度越好，混沌工程实验的可行性、有效性和安全性会更有保障。

| 实验成熟度等级 | 1级 | 2级 | 3级 | 4级 | 5级 |
|-----------|-----------------------------|-------------------------------------|-----------------------------|---------------------------------|----------------------------------|
| 架构抵御故障的能力 | 无抵御故障的能力 | 一定的冗余性 | 冗余且可扩展 | 已使用可避免级联故障的技术 | 已实现韧性架构 |
| 实验指标设计 | 无系统指标监控 | 实验结果只反映系统状态指标 | 实验结果反映应用的健康状况指标 | 实验结果反映聚合的业务指标 | 可在实验组和控制组之间比较业务指标的差异 |
| 实验环境选择 | 只在开发和测试环境中运行实验 | 可在预生产环境中运行实验 | 未在生产环境中，用复制的生产流量来运行实验 | 在生产环境中运行实验 | 包括生产在内的任意环境都可以运行实验 |
| 实验自动化能力 | 全人工流程 | 利用工具进行半自动运行实验 | 自助式创建实验，自动运行实验，但需要手动监控和停止实验 | 自动结果分析，自动终止实验 | 全自动的设计、执行和终止实验 |
| 实验工具使用 | 无实验工具 | 采用实验工具 | 使用实验框架 | 实验框架和持续发布工具集成 | 并有工具支持交互式的对比实验组和控制组 |
| 故障注入场景 | 只对实验对象注入一些简单事件，如突发高CPU高内存等等 | 可对实验对象进行一些较复杂的故障注入，如EC2实例终止、可用区故障等等 | 对实验对象注入较高级别的事件，如网络延迟 | 对实验组引入如服务级别别的影响和组合式的故障事件 | 可以注入如对系统的不同使用模式、返回结果和状态的更改等类型的事件 |
| 环境恢复能力 | 无法恢复正常环境 | 可手动恢复环境 | 可半自动恢复环境 | 部分可自动恢复环境 | 韧性架构自动恢复 |
| 实验结果整理 | 没有生成的实验结果，需要人工整理判断 | 可通过实验工具的到实验结果，需要人工整理、分析和解读 | 可通过实验工具持续收集实验结果，但需要人工分析和解读 | 可通过实验工具持续收集实验结果和报告，并完成简单的故障原因分析 | 实验结果可预测收入损失、容量规划、区分出不同服务实际的关键程度 |

接纳指数

接纳指数通过对混沌工程实验覆盖的广度和深度来描述对系统的信心。暴露的脆弱点就越多，对系统的信心也就越足。类似成熟度等级，对接纳指数也定义了1、2、3、4级进行描述。

| 接纳指数 | 描述 |
|------|---|
| 1级 | 公司重点项目不会进行混沌工程实验；只覆盖了少量的系统；公司内部基本上对混沌工程实验了解甚少；极少数工程师尝试且偶尔进行混沌工程实验。 |
| 2级 | 混沌工程实验获得正式授权和批准；由工程师兼职进行混沌工程实验；公司内部有多个项目有兴趣参与混沌工程实验；极少数重要系统会不定期进行混沌工程实验。 |
| 3级 | 成立了专门的混沌工程团队；事件响应已经集成在混沌工程实验框架中以创建对应的回归实验；大多数核心系统都会定期进行混沌工程实验；偶尔以Game Day的形式，对实验中发现的故障进行复查验证。 |
| 4级 | 公司所有核心系统都会经常进行混沌工程实验；大多数非核心系统也都会经常进行混沌工程实验；混沌工程实验是工程师日常工作的一部分；所有系统默认都要参与混沌工程实验，不参与需要特殊说明。 |

收...

| | | |
|-------|--------|--|
| 稳定性分析 | 定性分析 | 比较注入故障时的系统指标和稳态指标的差异 |
| | 定量分析 | 系统性能指标：P=E/E0，E为实验组性能指标，E0为稳态时性能指标 系统恢复率：R=R'/E，ER为移除扰动后系统性能指标，E为系统稳态性能指标 |
| 系统缺陷 | 各维度原因 | 对系统弱点进行分析 对故障应对过程中的不足进行分析 对系统的故障承受能力分析 对监控告警的有效性进行分析 对模块间的依赖关系进行分析 |
| 商业价值 | 参与人反应 | 参与前后调研问卷对比 |
| | 执行实验结果 | 从稳定性分析/系统缺陷体现，最好结合已有故障的影响，针对实验遇到问题作出预估的业务侧影响。 |
| | 缺陷改善 | 已发现问题都修复 新开发程序都没有发现已知问题 |
| | 业务结果 | 长期观察系统运行情况，建立混沌工程与故障时长、频次、恢复速度的关联关系 主要依据是故障真的发生时，系统因为做过实验而逃过一劫 |

如果能以线上事故驱动混沌工程，那么就可以在用混沌工程实验和测试覆盖“线上事故”的系统缺陷和监控告警改进点后，计算所能挽回的线上事故业务不可用时长，并可换算成可挽回的经济损失。当然，这里说到的“覆盖”，发生在我们已经修复了混沌工程实验所发现的漏洞之后。只有在此时，度量成效才有意义。

可以使用下面公式来计算混沌工程实验与故障注入测试的成效：

$$\text{混沌工程实验与故障注入测试成效} = \text{所覆盖的线上事故业务不可用时长} \times \text{单位时长所造成的经济损失金额} \times \text{实验执行次数}$$

...

| 入口服务 | 下游服务 | 应用进程 | 消息服务 | 数据缓存 | 数据存储 | 系统运维 |
|------|------|------|------|------|-------|------|
| 负载均衡 | 超时重试 | 资源隔离 | 异步传递 | 热点隔离 | 读写分离 | 监控告警 |
| 流量调度 | 服务降级 | 异步调用 | 消息分级 | 热点散列 | 分库分表 | 日志跟踪 |
| 请求限流 | 调用熔断 | 热点防护 | 削峰填谷 | 主从备份 | 主从备份 | 健康检查 |
| | 强制依赖 | | 消息存罐 | | 一致性保障 | 灰度发布 |
| | 幂等处理 | | | | | 发布回滚 |
| | 最优调用 | | | | | 弹性伸缩 |
| | | | | | | 容量规划 |
| | | | | | | 服务治理 |
| | | | | | | 异地多活 |

最小化爆炸半径

根据历史经验，可以考虑通过下面方式最小化爆炸半径

| 分类 | 原则 |
|------|---|
| 演练时机 | 1、低峰期先于高峰期 2、工作日先于节假日 3、变更前先于变更时、变更后 |
| 演练规模 | 用户粒度：1、单用户先于多用户 2、测试用户先于真实用户 请求粒度：1、单请求先于多请求 2、单模块复合请求先于多模块复合请求 组件粒度：1、小比例先于全组件 系统粒度：1、边缘系统先于核心系统 2、边缘可用区优于核心可用区 网络粒度：1、子网小比例先于全子网 2、子网先于全网 |
| 演练环境 | 非生产环境优于生产环境 |
| 演练模式 | 1、有剧本的演练先于无剧本随机演练 2、通过程序固化故障注入方式 |
| 稳态指标 | 突破稳态指标立即中止演练 |
| 终止通道 | 可随时终止演练 |

故障画像

| | | | | | |
|------|-------------|----------|---------|---------|------------------|
| Saas | Application | 进程Hang | 进程被杀 | 心跳异常 | 启动异常 |
| | | 镜像构建失败 | 包错误或损坏 | 注册中心异常 | 进程内部方法调用异常 |
| Paas | Data | 异步阻塞同步 | 依赖超时 | 依赖异常 | |
| | Runtime | 业务线程池满 | 流控不合理 | 内存溢出 | |
| | Middleware | 数据库热点 | 数据同步延迟 | 数据主备延迟 | 数据库宕机 |
| Iaas | O/S | 数据库连接池满 | 数据入库延迟 | 缓存连接池满 | |
| | | 缓存宕机 | 缓存策略不合理 | 其他中间件故障 | |
| | 虚拟机/物理机 | CPU抢占 | 内存抢占 | 内存错乱 | 上下文切换 |
| | Storage | 服务器宕机/假死 | 断电 | 超卖 | 混部 |
| | Network | 磁盘满、慢、坏 | | 不可写/读 | 服务器重启 |
| | | 断网 | 网卡满 | DNS故障 | 网络抖动、丢包、超时、重复、乱序 |

...

| 工具总结分类 | 工具名称 | 核心注入故障能力 |
|------------|---------------|---|
| 代码层次 | ByteMonkey | 字节码注入JVM类应用 |
| 运维层次 | Chaos toolkit | 可集成其他注入故障工具以及监控平台，定制故障场景和监控观测指标 |
| 创建容器故障 | Pumba | 杀掉、停止容器，网络延迟、丢包 |
| | ChaosMesh | 支持pod、压力类、网络类、IO类、时钟、kernel、DNS等多种故障，具备场景编排能力 |
| | Litmus | 杀掉pod、容器，网络延迟、丢包 |
| | KubeMonkey | ChaosMonkey的k8s集群实现，随机删除集群pod |
| | ChaosMonkey | 终止容器实例 |
| | Blockade | 网络故障、网络分区 |
| 侧重虚拟机故障 | Chaosd | 支持磁盘类、进程类、压力类、网络类、JVM类等多种故障 |
| 支持容器和虚拟机故障 | chaosblade | 支持pod、磁盘、网络、压力、JVM、数据库、消息队列等多种故障，不具备场景编排能力 |