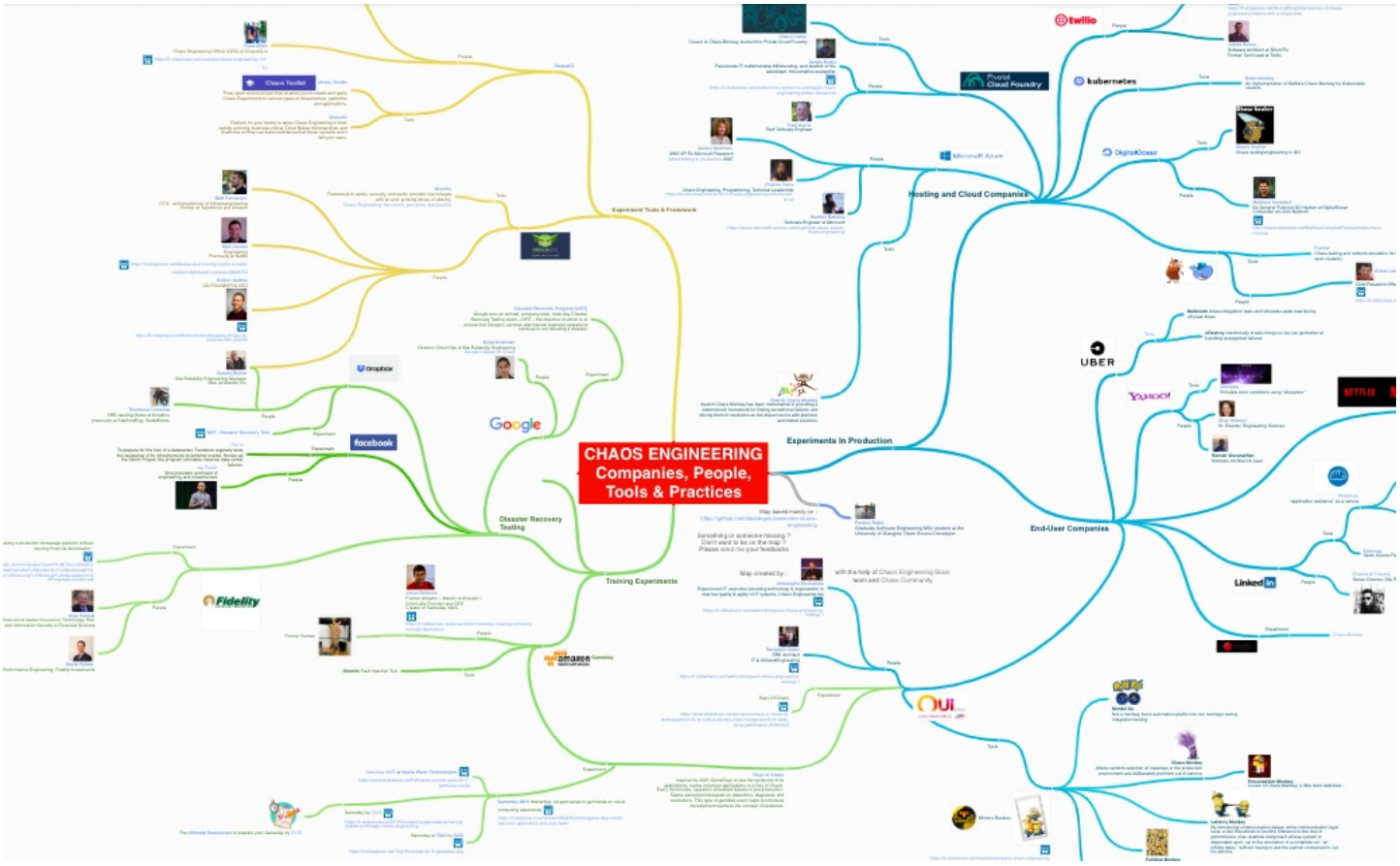
知乎 写文章



阿里巴巴在混沌工程领域的实践和思考

应用高可用
专注内容原创，转载麻烦注明出处“知乎-应用高可用”。

导读：

混沌工程（Chaos Engineering）：是在分布式系统上进行实验的学科, 目的是建立对系统抵御生产环境中失控条件的能力以及信心。最早由Netflix及相关团队提出。

故障演练（MonkeyKing）：是阿里巴巴在混沌工程领域的产品，目标是沉淀通用的故障模式，以可控成本在线上重放，以持续性的演练和回归方式运营来暴露问题，不断推动系统、工具、流程、人员能力的不断前进。

通过本文，您将了解到：

- 为什么需要混沌工程
- 阿里巴巴在该领域的实践和思考
- 未来的计划
- 时间线和参考资料

关键词：混沌工程、故障演练、最小化爆炸半径

一、为什么需要混沌工程（翻译自Chaos Engineering电子书）

1.1 混沌工程与测试的区别

混沌工程、故障注入和故障测试在关注点和工具中都有很大的重叠。

混沌工程和其他方法之间的主要区别在于，混沌工程是一种生成新信息的实践，而故障注入是测试一种情况的一种特定方法。当您想要探索复杂系统可能出现的不良行为时，注入通信延迟和错误等失败是一种很好的方法。但是我们也想探索诸如流量激增，激烈竞争，拜占庭式失败，以及消息的计划外或不常见的组合。如果一个面向消费者的网站突然因为流量激增而导致更多收入，我们很难称之为错误或失败，但我们仍然对探索系统的影响非常感兴趣。同样，故障测试以某种预想的方式破坏系统，但没有探索更多可能发生的奇怪场景，那么不可预测的事情就可能发生。

测试和实验之间可以有一个重要的区别。在测试中，进行断言：给定特定条件，系统将发出特定输出。测试通常是二进制态的，并确定属性是真还是假。严格地说，这不会产生关于系统的新知识，它只是将效价分配给它的已知属性。实验产生新知识，并经常提出新的探索途径。我们认为混沌工程是一种实验形式，可以产生关于系统的新知识。它不仅仅是一种测试已知属性的方法，可以通过集成测试更轻松地进行验证。

混沌实验的输入示例：

- 模拟整个区域或数据中心的故障。
- 部分删除各种实例上的Kafka主题。
- 重新创建生产中发生的问题。
- 针对特定百分比的交易服务之间注入一段预期的访问延迟。
- 基于函数的混乱（运行时注入）：随机导致抛出异常的函数。
- 代码插入：向目标程序添加指令和允许在某些指令之前进行故障注入。
- 时间旅行：强制系统时钟彼此不同步。
- 在模拟I/O错误的驱动程序代码中执行例程。
- 在 Elasticsearch 集群上最大化CPU核心。

混沌工程实验的机会是无限的，可能会根据您的分布式系统的架构和您组织的核心业务价值而有所不同。

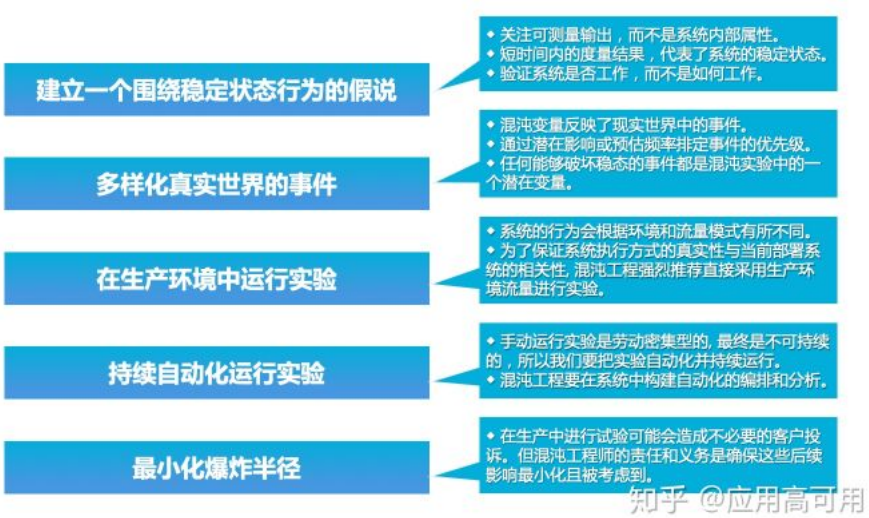
1.2 实施混沌工程的先决条件

要确定您的组织是否已准备好开始采用Chaos Engineering，您需要回答一个问题：您的系统是否能够适应现实世界中的事件，例如服务故障和网络延迟峰值？

如果您知道答案是“否”，那么在应用本书中的原则之前，您还有一些工作要做。Chaos Engineering非常适合揭露生产系统中未知的弱点，但如果您确定混沌工程实验会导致系统出现严重问题，那么运行该实验就没有任何意义。先解决这个弱点。然后回到Chaos Engineering，它将发现你不了解的其他弱点，或者它会让你更有信心你的系统实际上是有弹性的。混沌工程的另一个基本要素是可用于确定系统当前状态的监控系统。如果不了解系统的行为，您将无法从实验中得出结论。

1.3 混沌工程原则

为了具体地解决分布式系统在规模上的不确定性，可以把混沌工程看作是为了揭示系统弱点而进行的实验。破坏稳态的难度越大，我们对系统行为的信心就越强。如果发现了一个弱点，那么我们就有了一个改进目标。避免在系统规模化之后问题被放大。以下原则描述了应用混沌工程的理想方式，这些原则来实施实验过程。对这些原则的匹配程度能够增强我们在大规模分布式系统的信心。



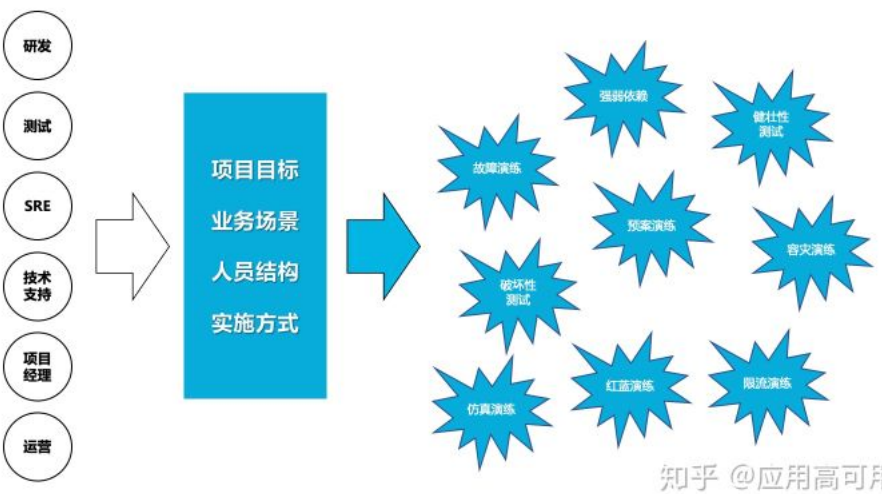
二、阿里巴巴在混沌工程领域的实践 - 故障演练

> **混沌工程** 属于一门新兴的技术学科，行业认知和实践积累比较少，大多数IT团队对它的理解还没有上升到一个领域概念。阿里电商域在2010年左右开始尝试故障注入测试的工作，开始的目标是想解决微服务架构带来的强弱依赖问题。后来经过多个阶段的改进，最终演进到 **MonkeyKing（线上故障演练平台）**。从发展轨迹来看，阿里的技术演进和Netflix的技术演进基本是同时间线的，每个阶段方案的诞生都有其独特的时代背景和业务难点，也可以看到当时技术的局限性和突破。
>
> 为了让更多的朋友了解这门新兴学科，接下来我们会在 **“知乎 - 应用高可用” 推出系列文章，**结合混沌工程原则来分析我们解决的问题和其演进过程，便于做稳定性的朋友做技术选型。

2.1 建立一个围绕稳定状态行为的假说

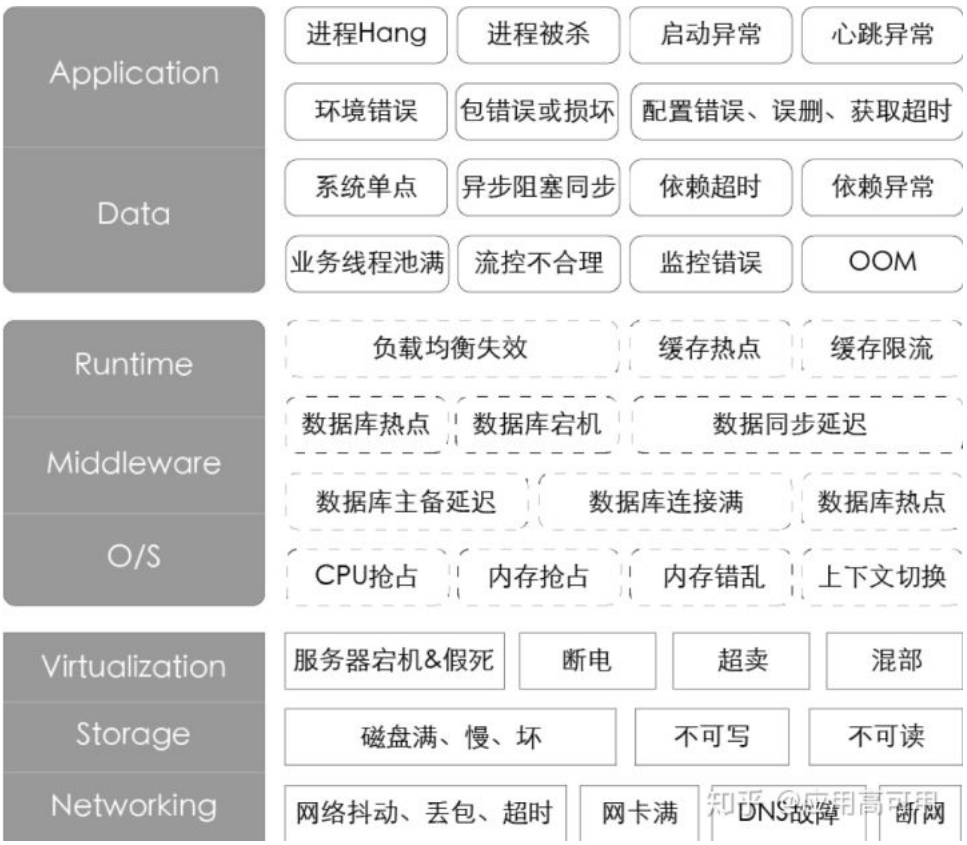
目前阿里巴巴集团范围内的实践偏向于故障测试，即在一个具体场景下实施故障注入实验并验证预期是否得到满足。这种测试的风险相对可控，坏处是并没有通过故障注入实验探索更多的场景，暴露更多的潜在问题，测试结果比较依赖实施人的经验。当前故障测试的预期比较两级分化，要么过于关注系统的内部细节，要么对于系统的表现完全没有预期，与混沌工程定义的稳态状态行为差异比较大。

引起差异的根本原因还是组织形态的不同。2014年，Netflix团队创建了一种新的角色，叫作混沌工程师（Chaos Enigneer），并开始向工程社区推广。而阿里目前并没有一个专门的职位来实施混沌工程，项目目标、业务场景、人员结构、实施方式的不同导致了对于稳定状态行为的定义不太标准。



2.2 多样化真实世界的事件

阿里巴巴因为多元化的业务场景、规模化的服务节点及高度复杂的系统架构，每天都会遇到各式各样的故障。这些故障信息就是最真实的混沌工程变量。为了能够更体感、有效率的描述故障，我们优先分析了P1和P2的故障（P是阿里对故障等级的描述），提出一些通用的故障场景并按照IaaS层、PaaS层、SaaS层的角度绘制了故障画像。



从故障的完备性角度来看，上述画像只能粗略代表部分已出现的问题，对于未来可能会出现的新问题也需要一种手段保持兼容。在更深入的进行分析之后，我们定义了另一维度的故障画像：

- 任何故障，一定是硬件如IaaS层，软件如PaaS或SaaS的故障。 并且有个规律，硬件故障的现象，一定可以在软件故障现象上有所体现。
- 故障一定隶属于单机或是分布式系统之一，分布式故障包含单机故障。
- 对于单机或同机型的故障，以系统为视角，故障可能是当前进程内的故障，比如：如FullGC，CPU飙高；进程外的故障，比如其他进程突然抢占了内存，导致当前系统异常等。
- 同时，还可能有一类故障，是人为失误，或流程失当导致，这部分我们今天不做重点讨论。

从故障注入实现角度，我们也是参照上述的画像来设计的。之前我们是通过Java字节码技术和操作系统层面的工具来分别模拟进程内和进程外的故障。随着Serverless、Docker等新架构、新技术的出现，故障实现机制和承接载体也将会有一些新的变化。

2.3 在生产环境中运行实验

从功能性的故障测试角度来看，非生产环境去实施故障注入是可以满足预期的，所以最早的强弱依赖测试就是在日常环境中完成的。不过，因为系统行为会根据环境和流量模式有所不同，为了保证系统执行方式的真实性与当前部署系统的相关性，推荐的实施方式还是在生产环境（仿真环境、沙箱环境都不是最好的选择）。

很多同学恐惧在生产环境执行实验，原因还是担心故障影响不可控。实施实验只是手段，通过实验对系统建立信心是我们的目标。关于如何减少实验带来的影响，这点在"最小化爆炸半径"部分会有阐述。

2.4 持续自动化运行实验

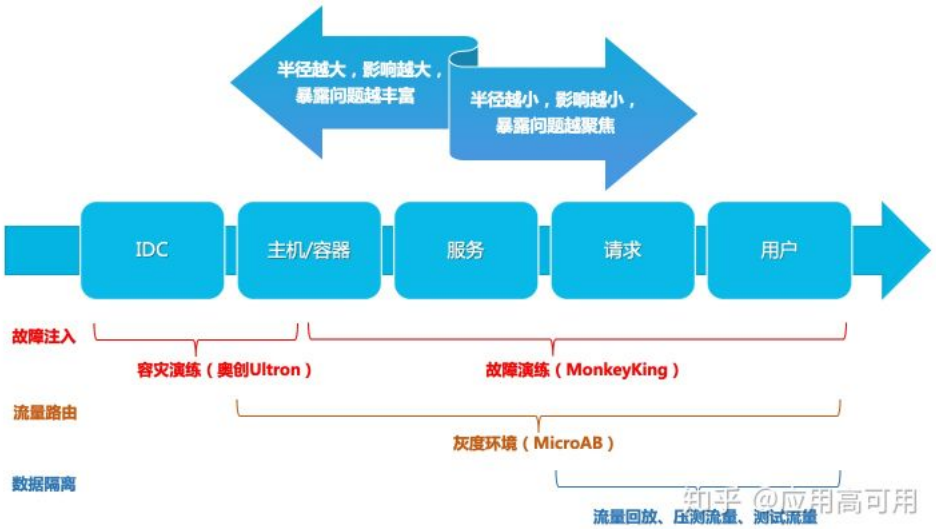
2014年，线下环境的强弱依赖测试用例是默认在每次发布后自动执行的。2015年，开始尝试在线上进行自动化回归。不过发展到最近两年，手动实验的比例逐渐变高。原因也不复杂，虽然故障注入自动化了，业务验证的成本仍然比较高。在业务高速发展、人员变化较快的环境之下，保持一套相对完善的线上回归用例集对是见非常难的事情。虽然也出现了流量录制技术，不过因为混沌工程实验本身会打破系统已有的行为，基于入口和出口的流量比对的参考度就下降许多。

为了解决测试成本问题，2017年初开始推进线上微灰度环境的建设。基于业务、比例来筛选特征流量，通过真实的流量来替换原来的测试流量，通过监控&报警数据来替代测试用例结果。目前已经有部分业务基于微灰度+故障演练的模式来做演练验证（比如：盒马APOS容灾演习）。

因为故障演练之前是作为一个技术组件被嵌入到常态和大促的流程中，所以在系统构建自动化的编排和分析方面的产品度并不高。演练可视化编排和能力开放会是我们团队未来的一个重点，下文中的规划部分会有所阐述。

2.5 最小化爆炸半径

在生产中进行试验可能会造成不必要的客户投诉，但混沌工程师的责任和义务是确保这些后续影响最小化且被考虑到。对于实验方案和目标进行充分的讨论是减少用户影响的最重要的手段。但是从实际的实施角度看，最好还是通过一些技术手段去最小化影响。Chaos Monkey和FIT的核心区别在于：是否可以进一步减小故障的影响，比如微服务级别、请求级别甚至是用户级别。在EOS的3.0~4.0阶段（详细请见最后时间线部分），已经可以实现请求级别的微服务故障注入。虽然那个时候演练实施的主要位置在测试环境，但初衷也是为了减少因为注入故障而导致的环境不稳定问题。除了故障注入，流量路由和数据隔离技术也是减少业务影响的有效手段。



三、未来的计划

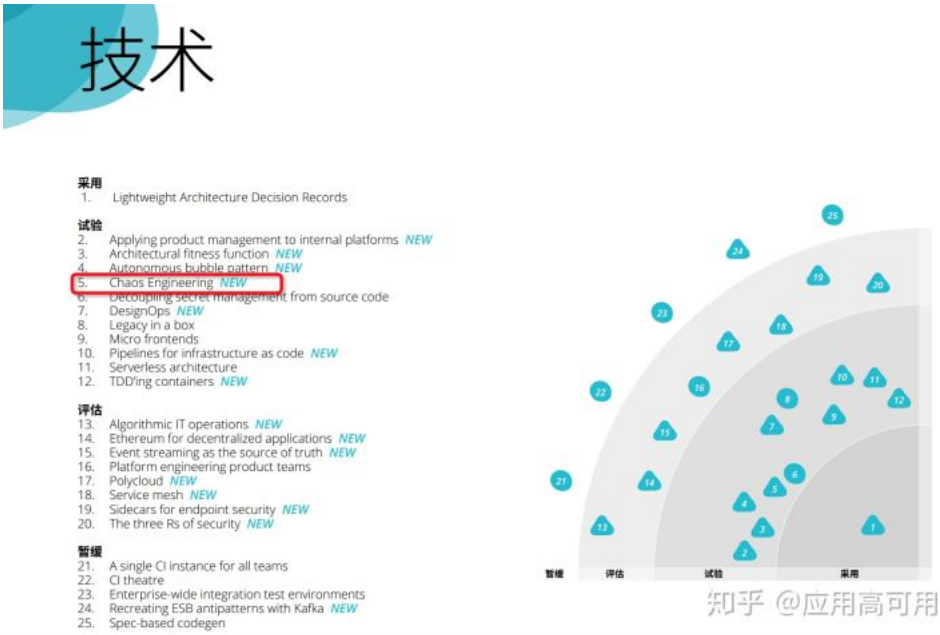
线上故障演练发展到今天是第三年，随着阿里安全生产的大环境、业务方的诉求、研发迭代模式的变化，以及大家对混沌工程的接受和认识程度的提高。集团的演练领域会向着未来的几个目标发力：

- 建立高可用专家库，结构化提高应用容错能力（解决"稳定状态定义"的问题）
- 建设故障注入实现标准，集团内开源，提升故障模拟的广度和深度（拓宽"多样化真实世界的事件"的广度）
- 规模化覆盖核心业务（提升"在生产环境中运行实验"的规模）
- 以产品化、平台化思路开放演练能力（探索"自动化运行实验"的方式）

四、其他

4.1 混沌工程的行业发展时间线

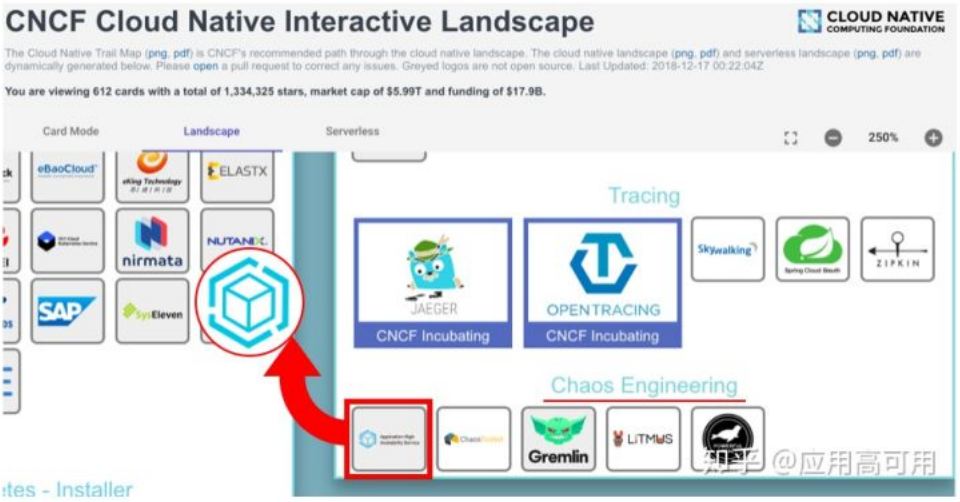
- 2010年，Netflix Eng Tools团队开发出了Chaos Monkey。当时Netflix从物理基础设施迁移到AWS上，为了保证AWS实例的故障不会给Netflix的用户体验造成影响，他们开发了这个工具，用来测试系统。
- 2011年，Simian Army诞生，在Chaos Monkey的基础上增加了故障注入模式，可以测试更多的故障场景。Netflix 认为，云的特点是冗余和容错，但没有哪个组件能够保证100%的可用性，所以他们必须设计出一种云架构，在这种架构里，个体组件的故障不会影响到整个系统。
- 2012年，Netflix在 GitHub上开源了Chaos Monkey，并声称他们“已经找到了应对主要非预期故障的解决方案。通过经常性地制造故障，我们的服务因此变得更有弹性。”
- 2014年，Netflix团队创建了一种新的角色，叫作混沌工程师。Bruce Wong 发明了这个角色，并由 Dan Woods 在 Twitter上向广大的工程社区推广。
- 2014年，Chaos Monkey的升级版本FIT诞生，实现微服务级别的故障注入。
  2016年，混沌工程(Chaos Engineering)的概念通过《Principles Of Chaos》这本书被提出。面向失败设计也逐渐成为构建一个高可用架构产品的基本要求，混沌工程也在更多公司进行实践。
- 2017年，Chaos Monkey的3.0版本ChAP（Chaos Automation Platform）诞生，可以配合环境、监控实现部分场景的自动化演练。
  概念：强弱依赖系统（EOS），主要为通过程序自动化的帮助应用依赖梳理、系统降级提供基础数据，减少人肉成本，定位是线下环境的测试工具。
- 2017年，Chaos Engineering被收录到ThoughtWorks Technology Radar。
- 2018年，混沌工程（Chaos Engineering）成为CNCF的一个新的技术领域。



4.2 阿里巴巴实践的发展时间线

- 2012年，Eos 1.0(强弱依赖系统统)开发完成，在线下环境进行应用依赖的检测。
- 2013年2月，Eos 2.0，尝试在二套环境下面进行检测，通过SCM脚本自动化安装应用，减少人工安装应用成本。
- 2013年5月，完成二套环境下面的改造，应用检测的自动化完成。
- 2013年7月，开始和持续集成团队合作，复用测试用例，做接口级别的依赖检测。
- 2014年7月，进入Eos 3.0，基于RPC进行故障模拟，不依赖具体的环境，更加丰富的注解，实现强弱依赖的自动化。
- 2015年8月，Eos 4.0，全部基于ASM字节码技术进行故障模拟，不依赖具体的环境，丰富的注解，支持根据用户ID进行屏蔽和检测，主要围绕自动化、常态化、降低成本几个方面发展，实现强弱依赖的常态化运行。
- 2015年12月，与线上测试团队产品进行合作，识别压测数据，尝试在线上进行故障演练。
- 2016年，故障演练项目立项（GOC+中间件），重新设计架构和产品流程，确定产品名为MonkeyKing，在交易和中间件链路尝试演练。
- 2016年11月，开始筹备和推进线上微灰度环境的建设。
- 2017年3月，具备前后端灰度隔离的能力，可以基于业务、比例来筛选特征流量，通过真实的流量来替换原来的测试流量。
- 2017年，成立双11故障演练项目组（GOC+中间件），优化产品流程和使用体验，把演练推广

到更多业务线，演练场景3.9k个。
- 2018年，演练场景7.6k个。
- 2018年9月，应用高可用服务（AHAS）在阿里云公测，故障演练服务面向公有云客户输出。
- 2018年10月，阿里巴巴实践混沌工程的实践产品AHAS也被列入CNCF混沌工程云原生全景图中。



4.3 参考资料

- Chaos Engineering（O'Reilly出版）
- Principles Of Chaos
- 超全总结 | 阿里电商故障治理和故障演练实践
- 国外实践Chaos Engineering的一些公司
- Chaos Engineering 的历史、原则以及实践
- ChAP: Chaos Automation Platform
- FIT: Failure Injection Testing
- Fault injection Wikipedia Page
- 阿里云应用高可用服务AHAS

编辑于 2018-12-17

分布式系统 高可用 混沌

推荐阅读

两阶段提交的工程实践

郁白 发表于分布式与存...

第二十五章.世界一片混沌，人们看不清未来

武侯大道的精分暗战，将不再专注于毒品犯罪。而专注于冰毒对我们社会风气的影响。书中犯罪将会少描写，而会着重于描写冰毒色情业以及各层次吸鬼的生活。 回到C市的那个下午，妻还在上班，她…

暮呈渊

【Numberphile】从混沌游中得到的是什么

无序中存在着有序，偶然中蕴含必然，按照看似混沌无序的方式进，我们却可以得到有序规则的果？
youtu.be/kbKtFN71Lfs&t
译：一身是（海）胆的汤姆 校正

圆桌字幕组 发表于圆桌字

还没有评论

写下你的评论...