字节跳动混沌工程实践之场景化主动实验

原创

字节跳动技术团队 于 2021-03-25 11:51:01 发布 4760 收藏 9
文章标签： 大数据 分布式 人工智能 java 编程语言

本文链接：https://blog.csdn.net/ByteDanceTech/article/details/115222625

背景

从 2010 年 Netflix 上线 Chaos Mokey 的第一个版本到现在，虽然混沌工程发展已历时十年，但其实只在少数大厂里面有较成熟的落地，对绝大部分研发同学来说，混沌工程还是一个比较陌生的领域。

分布式和微服务化已经成为主流的系统架构设计方案，大规模分布式系统的可用性保障能力越来越成为关注的重点。混沌工程也开始如雨后春笋般在各大企业内部萌芽生长，但大部分还处于初期的探索阶段，在实践过程中也遇到了这样或那样的问题，有技术上也有认知层面上的，这些问题难免会对混沌工程的快速落地产生阻力。

下面介绍一下字节跳动在混沌工程实践过程中的一个关键阶段：**场景化主动实验**。希望本文可以帮助大家加深对混沌工程价值的了解，对设计混沌工程实验、落地混沌工程建设提供更多的思路。

什么是场景化主动实验

混沌工程的高级原则要求能够在生产环境自动的运行实验，这个目标并不是一蹴而就的。

根据混沌工程成熟度模型(CMM)[4]说明，要分别从“熟练度”和“应用度”两个维度同时进行建设。其中，“熟练度”体现了混沌工程系统的有效性和安全性，“应用度”衡量了混沌工程实验覆盖的广度和深度。在混沌工程建设的中前期，这两点都是混沌工程成功落地的关键路径。

在混沌工程的初级阶段，通常都会建设一个故障注入测试平台（FIT，Fault Inject Testing），集成一些常见的故障场景或异常事件的模拟能力，由业务或 QA 同学设计并执行实验来验证系统的韧性能力。

在这个阶段，基础架构和业务系统的实现都可能处于比较粗放状态，混沌工程平台的故障注入能力需要兼容各种业务架构的实现方案和软硬件环境，执行实验时，业务同学不仅要设计实验的故障场景（机房网络故障、下游服务宕机等）、配置演练环境（目标服务、实验集群等控制实验的爆炸半径），还要找到能够描述实验时服务状态的稳定性的指标（如 metrics、日志或告警等），然后手动启动实验，执行人还要不停的观察稳定性指标的变化，判断系统的容灾逻辑或弹性策略是否被正确触发、业务系统的表现是否符合预期等等。如果在执行过程中发现异常，需要立刻终止实验，收敛实验影响。

整个实验过程的人力成本较高，实验的操作门槛也较高，再加上这个阶段业务同学对混沌工程价值和理念的认知还处于较初级水平，很难会主动对自己的服务设计实验，更无法保证实验的常态化执行。因此，混沌工程实验的时效性和业务系统的弹性容灾策略持续有效就比较难以保证了。

如何突破这个阶段、成功抵达混沌工程的终极目标呢？

通过不断的思考，我们认为混沌工程建设需要一个过渡阶段，即场景化主动演练。

所谓场景化主动演练，就是在明确混沌工程的终极建设目标的前提下，以终为始，分阶段去设计混沌工程的实验标准、定义技术规范，搭配工程化能力，逐步将人和业务引导到混沌工程建设的高速公路上，共同推进 CMM 模型的熟练度和应用度。

所以，场景化主动实验是通向混沌工程自动化建设的关键路径。

混沌工程演变图

如何建设场景化主动实验

首先需要明确混沌工程的最终目标，以终为始，反推当前阶段应该建设什么样的技术规范和标准能力。

然后，根据业务当前的基础架构现状和实验诉求构建一个通用的实验场景，由混沌工程平台方在保证实验风险可控的条件下主动对业务系统进行实验。这样，在满足业务需要的同时又可以推动相关技术规范和基础能力的建设，而且对业务同学的资源依赖较少。

以字节跳动为例，要实现在生产环境自动的执行可控的混沌工程实验，当前阶段应该具备的能力包括：

能够在生产环境持续的运行实验并具备实验爆炸半径的控制能力

选定一个命中业务痛点且通用的实验场景，构建通用的自动化执行实验能力

能够描述服务稳定性的通用指标

自动检测稳定性指标的变化

自动终止实验【在实验爆炸半径可控的情况下，非必须】

场景主动实验能力拓扑图

主动实验

FIT 平台需要业务同学分析业务容灾场景并制定实验，然后执行实验进行验证，混沌工程平台方只是给予一些技术支持和建议。在业务同学对混沌工程认知度不高情况下，实验的主动性、覆盖率、时效性都很难保证，如果再想让业务同学去配合建设一些混沌工程的基础能力难度就可想而知。

转换一下思路，将业务主动改为平台主动！由混沌平台针对业务系统的某个场景主动执行混动实验来验证服务的弹性能力，业务同学只需关注实验结果。只要实验场景契合业务痛点、实验结果对业务构建弹性系统有意义，业务同学自然会认可混沌工程的价值，也会更加积极的参与混沌工程实验和混沌工程的基础建设。

实验场景

混沌工程的价值是发现业务系统中潜在薄弱环节，提升业务系统韧性能力，即服务可用性和稳定性，所以主动实验场景也应该满足这个前提。

在字节跳动，主动实验落地的第一个场景是验证服务调用链的强弱依赖关系。所谓强弱依赖关系，就是当访问的下游服务异常（服务宕机、响应超时、接口返回失败等）时，调用方的稳定性和可用性是否会受到影响。例如，查询缓存 miss 或失败后可以继续回源访问数据库，缓存的问题并不会影响服务可用性，这个缓存就是个弱依赖。

为什么要选择这个场景呢？公司的很多线上运营事故都是因为不合理的依赖关系导致的。字节跳动有很多高 QPS 的海量服务系统，为了保证用户体验，对高可用高稳定性的要求很高，会通过各种缓存、服务兜底、数据兜底等容灾策略来减少强依赖服务的数量。另外，在字节跳动的服务治理体系中，会根据服务间的强弱依赖调用关系为某些故障场景预设自动容灾降级的策略。比如，当服务治理系统检测到系统负载过高时，能够自动对弱依赖的下游执行降级，通过 FailFast 来缓解系统的负载。因此，业务负责人通常都会比较关注服务的强弱依赖关系。

服务依赖调用链

自动化

主动设计和执行实验意味着将业务侧的人力成本转嫁给了混沌平台，混沌平台必须要借助通用化、自动化的执行能力来降低人力成本。需要强调的是，这里的通用化和自动化一定是和混沌工程的终极目标一致的，能够承载过渡阶段的职责。

可以从混沌工程的高级原则[2]进行剖析：

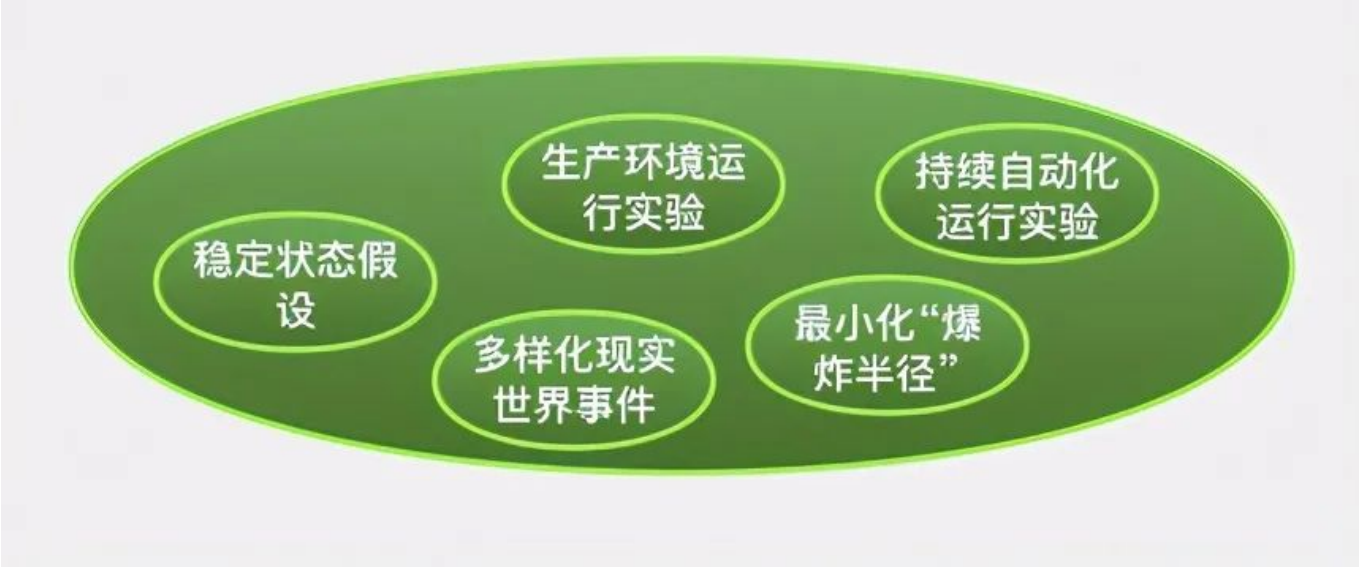
> **服务的稳定状态假设**：找出能够描述服务稳定状态的通用指标，可以是 metrics、告警等。执行实验前，这些稳定状态指标应处于稳定状态；执行实验时，如果服务的可用性或稳定性受到影响时，稳定性指标会发生变化，比如 metrics 曲线的剧烈波动、触发告警等；当实验结束后，稳定性指标又会恢复到之前的稳定状态。如果执行实验时，稳定状态假设发生了变化，说明这是个强依赖。

生产环境运行实验：在风险可控的条件下推荐在生产环境运行实验，因为系统的行为会根据环境和流量模式的不同而有所不同，系统用户也不会像你所预期的那样与你的系统进行交互。当然，在生产环境执行混沌工程实验很可能是有损的，为了避免对用户体验产生较大影响，一定要控制好实验的爆炸半径，比如筛选出少量的实验流量，这就要求混沌平台具备实验流量筛选与调度的能力。

多样化现实世界事件：也就是各种故障的模拟能力。因为是场景化实验，所以对多样化的诉求倒不像 FIT 那样强烈。

最小化“爆炸半径”：如果选择在线上环境执行实验，需要具备筛选实验流量的能力，即筛选出满足实验所需的最小流量即可，严格控制“爆炸”的影响范围。

持续自动化运行实验：通过工程化能力将实验目标过滤、实验流量筛选、执行实验、稳定状态检测、生成实验报表、实验结果反馈收集等流程串联成自动化的执行流，减少人力依赖。另外，混沌工程实验从来都不是一锤子买卖，应该通过常态化的运行实验来持续的保证服务的高可用和弹性机制符合预期。



混沌工程高级原则

服务标准与技术规范

一个公司内部通常会有很多条产品线，每个产品又会包含很多微服务，还会依赖到中台服务或基础组件，这些不同模块的开发语言、服务框架、技术栈等可能五花八门。要想实现混沌工程通用化与自动化的能力，就必须制定一些通用的服务标准和技术规范。

混沌工程不仅需要了解业务系统当前的技术栈，还要能够预测未来的技术发展趋势，帮助业务提前规划切实可行的技术规范并协助进行建设。

举个例子，如何定义服务的稳定状态？

在字节跳动，我们以调用方视角定义了一个服务级别的 metrics 指标来描述服务的稳定性状态：

被调方在响应包中添加一个通用的 stability 字段来标识本次请求的处理结果是否成功

调用方从响应包中解析出 stability 字段并上报 metrics。某些异常场景（如被调方无响应）导致调用方无法解析出 stability 字段，会上报一个特殊的不稳定值。

之所以要在响应包中单独定义一个 stability 字段是为了区分开系统错误和业务错误。因为业务错误并不表示服务出现了问题，所以我们更关注系统错误。举个例子，删除好友时被调方返回好友不存在的错误，这个错误并不表示系统的稳定性出现了问题。

这个规范最开始只在字节跳动的少数核心服务中使用，其他服务通常都有自己的稳定性指标，有些服务可能需要多个 metrics 指标一起来描述服务的稳定性。因为混沌工程的通用性要求应该具备一种通用的指标来描述所有服务的稳定性状态，经过评估，我们将 stability metrics 作为了混沌工程的服务通用稳定性描述指标，并计划将其推广到全公司的业务，无论什么服务框架和开发语言都可以低成本的快速实现。

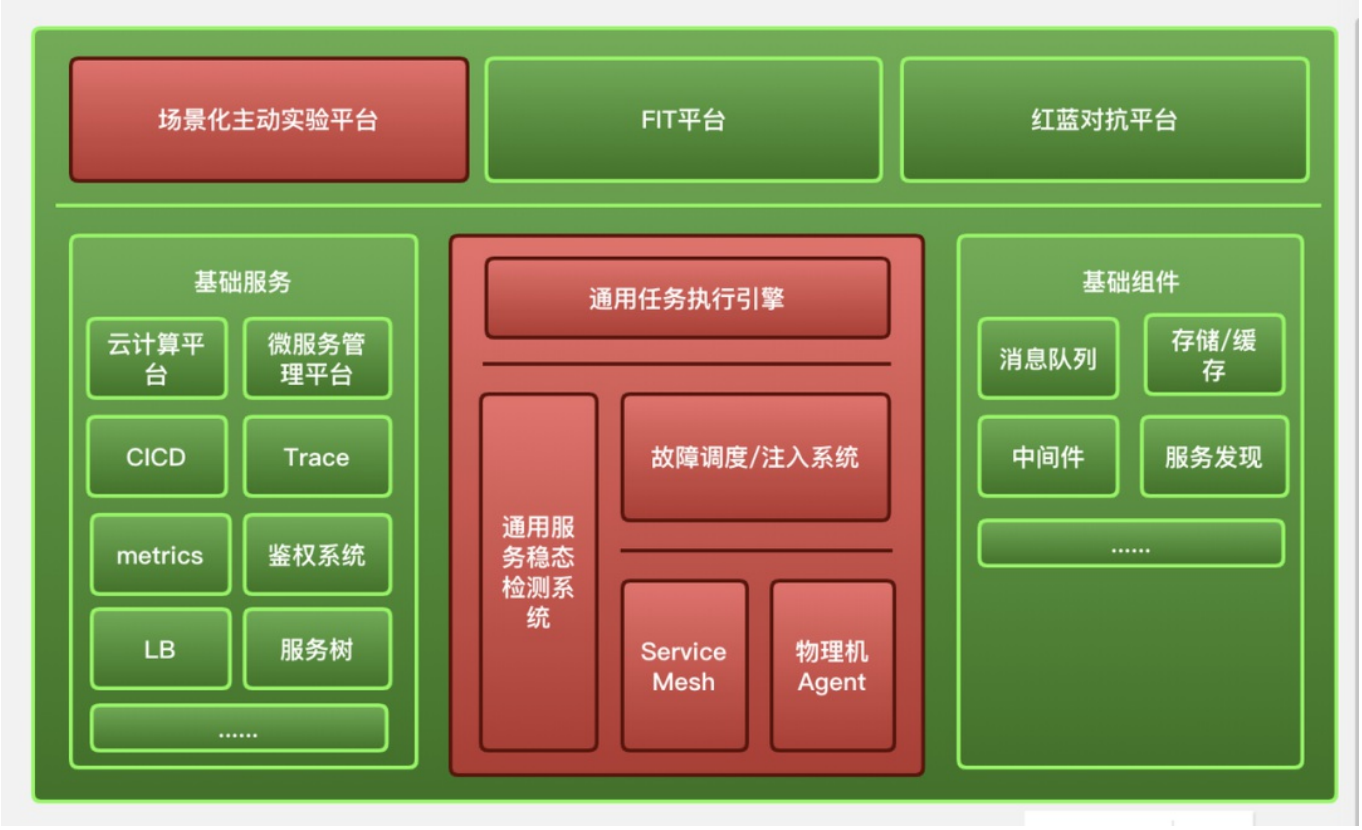
字节跳动的混沌工程概况

字节跳动的混沌工程体系主要包括场景化主动实验平台、FIT 平台和红蓝对抗平台。

场景化主动实验平台：以平台视角主动对业务发起混沌实验，在验证业务系统的高可用和弹性能力的同时推动服务标准化和技术标准化建设，提升混沌工程价值产出和影响力，为实现混沌工程的终极目标做好铺垫。

FIT 平台：业务同学自助验证服务容灾机制的正确性，具备各种异常事件和故障模拟能力，提供了灵活的可视化实验编排能力。

红蓝对抗平台：以第三方视角，用系统化、随机化的对抗实验方式来验证业务系统的高可用水平、监控和告警的有效性，以及异常情况下人工介入时机、故障定位和故障恢复时间是否达标等。



字节跳动混沌工程整体架构图

字节跳动的场景化主动实验建设方案

场景化主动实验的流程图

1. 实验目标

同时推进混沌工程建设的“熟练度”和“应用度”，以生产环境自动执行混沌工程实验为目标构建场景化主动实验的流程和标准，实验场景选定为验证服务间的强弱依赖调用关系，可自动化运行实验，具备实验爆炸半径控制能力。

通过验证强弱依赖调用关系的正确性反推服务的稳定性指标是否规范，推动稳定性指标的规范落地。

2. 实验对象

优先在字节跳动的核心服务落地，树立模范标杆，然后扩展到更大范围。

3. 稳定性指标&波动自动化检测

稳定指标：

因为 stability metrics 已经被多数核心服务当做通用的稳定性描述指标，所以场景化主动演练将 stability metrics 作为稳定性描述的核心指标，同时辅助判断接口成功 qps、失败 qps、调用下游成功 qps、失败 qps、pct99 等指标。

指标波动检测方案：

自动化执行实验要求混沌平台能够自动检测稳定性指标的变动，因为不同服务的指标曲线是不一样的，同一服务的不同时刻的指标曲线也是不一样的，所以预置曲线波动的阈值上限的效果肯定不会太好。因此，在项目启动阶段我们就直接探索自动化的动态检测 metrics 曲线波动的方案：

一种是 Netflix 介绍的 AB test 方案，对比实验曲线和样本曲线的差异

一种是实时计算曲线的变化趋势

由于方案一要求具备更灵活的流量筛选能力和实验环境隔离能力，当前阶段的建设难度和成本较高，所以我们选择了方案二。

自动化波动检测：

起初我们参考线上报警的检测方案：在执行主动演练时，先通过机器学习为稳定性指标实时训练检测模型，然后用模型实时检测指标曲线的变化。但是，因为主动实验的时间较短（一个实验节点只有 60~120 秒）、metrics 数据点稀疏（一个节点的实验时间只能采集到 2~4 个数据点）、实验流量较低（爆炸半径控制在 5~10 QPS），所以基于机器学习的检测效果并不理想。

于是，我们改成组合多种统计规则的检测算法，根据最近一段时间的历史数据动态生成曲线的合理波动范围阈值，然后在实验过程中实时检测增量数据点的波动范围。如果数据点超出了波动范围阈值，就被判定为不稳定。

经过不断调优，最终把这个场景下的指标检测效果优化到了预期水平。

优化方案：

在实践中，我们发现单个稳定性指标的曲线会偶现非预期的波动噪音，这些噪音会影响曲线检测结果的准确率。

于是，我们增加了一个噪音过滤策略：通过对比有相关性的多条稳定性指标曲线的波动相似度来过滤噪音。举个例子，对下游依赖服务注入故障后，调用下游服务失败的 metrics 曲线会上升，如果稳定性指标曲线也上升，而且这两个曲线的变化趋势也相似时，才会认为曲线变化是受实验的影响。这个策略能够比较有效的过滤掉偶现的曲线波动噪音。

具备了稳定状态的检测能力，在场景化主动实验时就可以根据稳定状态的检测结果自动推断下游依赖服务是强依赖还是弱依赖。

4. 最小化爆炸半径控制

为了保证实验的通用性和降低构造实验流量的成本，我们选定在生产环境执行实验，因此最小化爆炸半径控制能力就非常重要了。

字节跳动的生产环境有个用于服务灰度上线的金丝雀集群，在服务上线时会先升级这个集群来验证服务的正确性。这个集群的实例比较少，且支持通过修改集群权重来调整进入集群流量。

经过验证，实验流量在 5~10QPS 时就可以保证稳定性 metrics 指标检测的准确率。所以，执行实验时，先从服务的金丝雀集群中随机选择一个实例作为实验目标，计算实例流量与预期流量的偏差重新生成权重值，然后通过修改集群权重来调整实例流量。

5. 验证方式

字节跳动的微服务管理平台通过聚合服务的 Trace 日志生成了服务间的调用拓扑图，通过 OpenAPI 可以查询到某个服务的所有一级下游依赖服务列表。

然后，逐个对下游依赖服务注入一段时间的宕机故障，同时检测服务的稳定性指标是否出现异常波动来推断下游依赖服务是强依赖还是弱依赖。

需要注意的是，因为是逐个对下游依赖服务执行实验，为了避免前一个实验对下一个实验产生干扰，我们在两次实验间增加了一个间隔时间，具体的间隔时长依赖所使用的指标检测算法。

6. 实验报告

实验执行结束，平台会将下游依赖服务的强弱依赖推断结果、执行上下文、稳态指标监控视图和检测结果等汇总成一个实验报告发送给服务负责人。

服务负责人确认实验报告，如果发现实验结果不符合预期，可以通过执行上下文和监控视图等信息来辅助定位问题。同时，可以在实验报告中备注不符合预期原因、问题修复方案和修复进度等，平台会定时跟进修复进度并提醒服务负责人更新结果。

如果实验结果符合预期或问题已完成修复，实验报告会进入结单状态，同时记录服务的强弱依赖推断结果，并输出给服务治理平台，应用于服务在线治理。

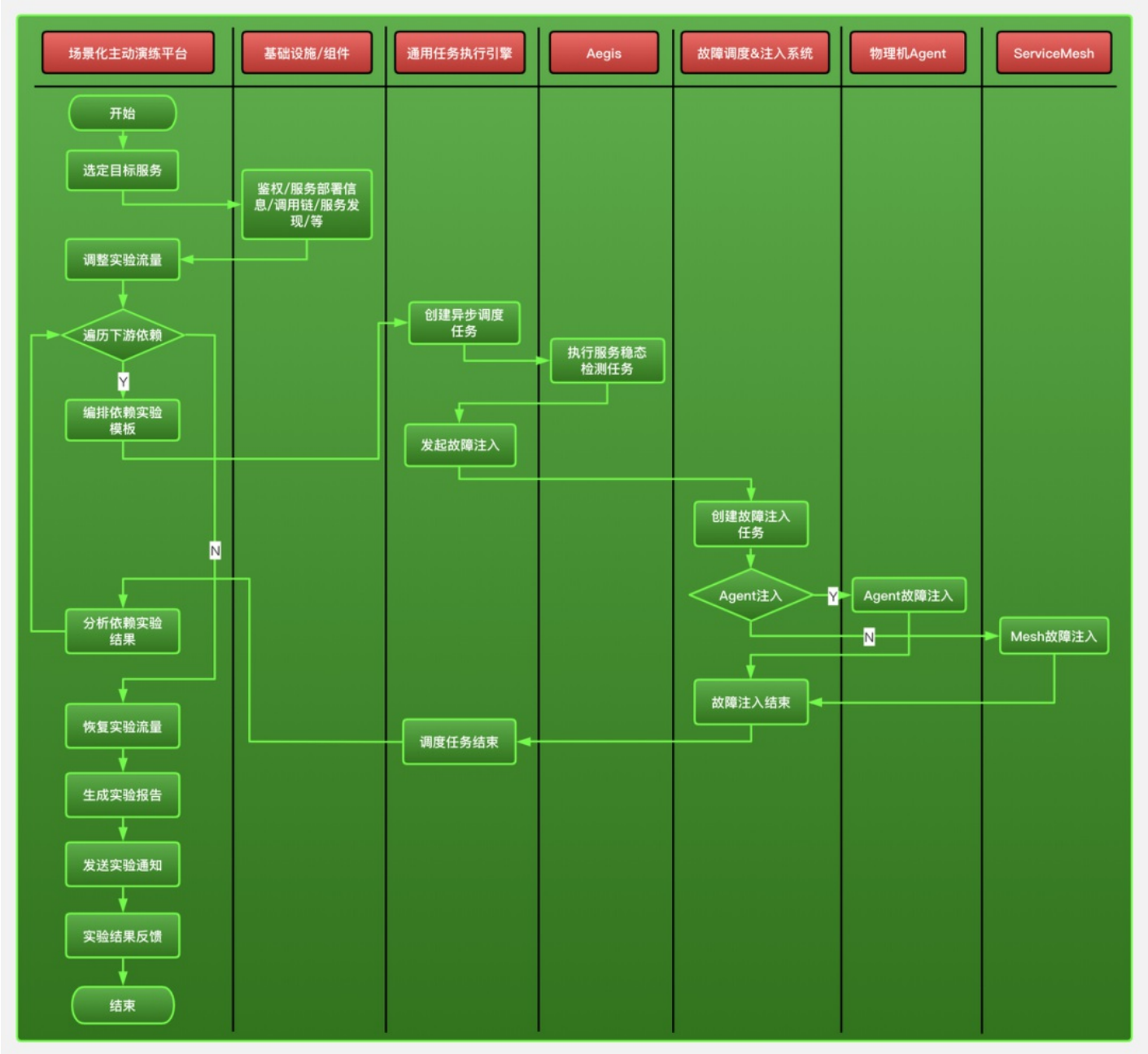
7. 实验结果自动保准

业务系统是持续迭代的，下游依赖关系也是动态变化的。如果非预期的引入了强依赖，会给系统增加可用性风险，因此场景化主动实验也需要常态化执行。

已完成的主动实验都可以开启试验结果自动保准，每隔一定时间会自动执行实验验证下游依赖服务的强弱依赖关系，并与历史实验结果进行对比，并将变动的部分通过实验报告发送给服务负责人。

经过服务负责人确认后，新的验证结果会被更新到存储和发送给服务治理平台。

8. 完整的实验流程



场景化主动实验流程图

场景化主动实验的价值

我们的验证范围从核心服务开始、逐渐向外围服务扩张。

核心服务的验证结果基本符合预期，但还是有少数个案存在稳定性指标不规范、容灾逻辑不符合预期的情况。通过实验结果自动保准机制，也多次及时发现因代码变更所导致的容灾逻辑失效情况。

外围服务验证出来的问题就比较多了，稳定性指标严重不规范导致无法评估服务的稳定性、容灾逻辑覆盖率低导致过多的强依赖使得服务可用性风险较高等等。需要持续的推动业务进行优化升级，并给予一些高可用弹性系统的建设思路或参考方案。

场景化主动实验近期工作

最近，我们新上线了在结果保准时自动对弱依赖服务注入随机故障来进一步探索弱依赖的抗风险能力。我们认为弱依赖服务在任何故障场景下都不应该影响服务的可用性。

另外，场景化主动实验还接入了服务的上线流程，在服务灰度上线时就触发实验，让服务负责人能够更及时的发现不符合预期的系统变更，如果有必要可立刻终止或回滚上线变更。

场景化主动实验的未来规划

通过场景化主动实验，我们已具备持续保障字节跳动核心服务的稳定性指标规范性、强弱依赖正确性，以及弱依赖的抗风险能力等等。

接下来，我们会把场景化主动实验扩展到更大服务范围和更多业务场景，让场景化主动实验发挥出更大的价值。

另外，我们也在思考打通服务上下游，从点到线再到面，以更高维度的系统视角来探索启发式的智能混沌实验。

参考文献

Netflix 混沌工程开源项目 - Chaos Monkey：

https://github.com/Netflix/chaosmonkey

PRINCIPLES OF CHAOS ENGINEERING：

https://principlesofchaos.org/?lang=ENcontent

《混沌工程：Netflix 系统稳定性之道》：

https://www.oreilly.com/library/view/chaos-engineering/9781491988459/

《Chaos Engineering》：

https://www.infoq.cn/article/M3EktXxYGRYYm*t5vKga

阿里巴巴 混沌工程开源项目 - ChaosBlade：

https://github.com/chaosblade-io/chaosblade

PingCAP 混沌工程开源项目 - Chaos Mesh：

https://github.com/pingcap/chaos-mesh

字节跳动混沌工程实践总结：

https://mp.weixin.qq.com/s/kZ_sDdrbc-_trVLNCWXyYw

点击阅读原文，快来加入我们吧！