故障排查及其要求与过程

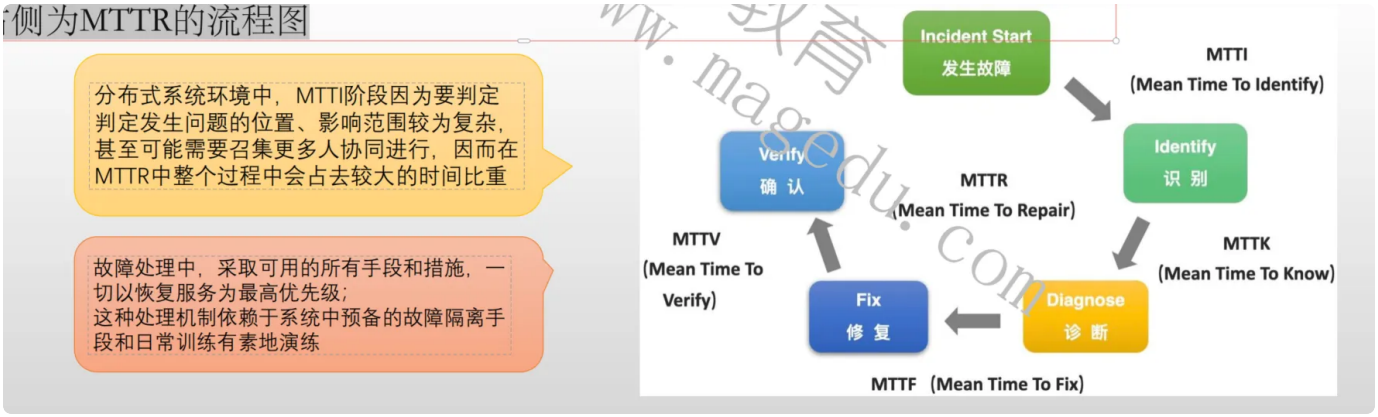
故障排查是运维分布式计算系统的一项关键技能，它并非天赋，而是可以靠后天学习获得

新手们常常不能有效地进行故障排查，是因为这个过程，在理想情况下同时需要两个条件

- 对通用的故障排查过程的理解（不依靠任何特定的系统）
- 对发生故障的系统足够了解
  - 理解系统内部运行的原理
  - 理解系统的设计方式和构建原理

依靠通用性的流程和手段也可以处理系统中的一些问题，但这样通常很低效

右侧为MTTR的流程图



故障排查的方法

故障排查过程的定义

■反复采用“假设—排除”手段的过程

■针对某系统的一些观察结果和对该系统运行机制的理论认知，基　　于收到的问题报告，不断地提出造成该问题的假设，进而针对这些假设进行测试和排除

排查过程

■从收到的系统故障报告中描述的问题开始

■通过观察系统的监测指标、日志信息和请求跟踪信息了解系统目前的状态

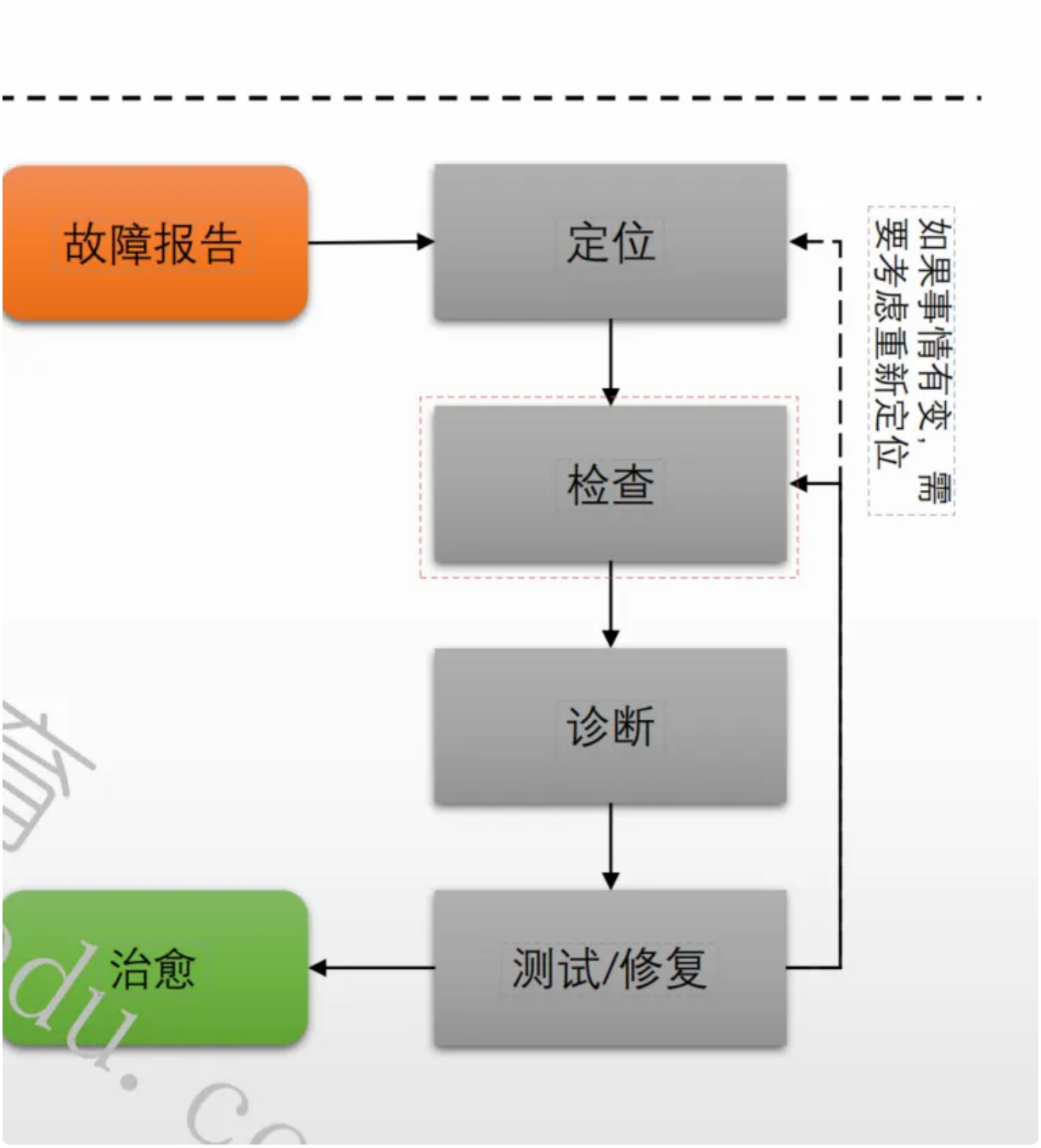
■再结合掌握的系统构建原理、运行机制，以及失败模型等提出一些可能导致失败的原因，提出假设

■测试假设是否成立

◆将假设与系统的实际状态对比，找出支持或不支持假设的证据

◆尝试修复该系统，对系统进行可控的配置调整后，观察操作的结果

■当所有的可能性都存在时，要优先考虑最简单的情形



故障排查过程

收到有关系统故障的报告

■故障报告可能源自告警系统、用户反馈或者舆情监控等

◆告警系统通常只应该发送事关SLO指标的告警信息

■有效的故障报告应该清晰记录了预期、实际结果和重现方式（但未必能实现）

■这些报告应该采用一致的格式，存储在一个支持搜索、自动发送报告等操作的专用系统中，功能丰富一些的

该类系统甚至还可以为常见问题提供一个自服务分析工具或者自服务修复工具

判定问题严重程度

■合理判定一个问题的严重程度，需要On-Call工程师具有较好的心理素质、丰富的经验和良好的判断能力

■根据问题的严重程度采取合理的应对级别

◆仅影响特定用户的问题：直接尝试故障排查和修复测试

◆影响整个服务可用性的紧急问题

●进行响应升级，立即根据流程召集更多人参与进来

●**缓解系统问题是第一要务，需要优先尽最大可能地让系统恢复服务**

开始问题排查

■逐一检查每个组件的工作状态

■可用素材包括监控、日志和请求链路跟踪数据

诊断

■根据检查的结果，正确诊断出问题所在的一个重要前提是对系统设计和原理有着清晰、深入的理解

■另外，也存在一些通用的手段帮助完成诊断

◆简化和缩略：检查组件间的连接及组件间传递的数据是否符合预期

◆关注异常操作：检查系统上的非预期内的操作，以及操作的原因和系统资源的消耗去向

◆ 检查故障前的最后一个变更：正常运行状态的系统因惯性很少直接出错，直到某个外力因素的出现，例如变更

◆开发和使用专用的诊断工具：对于重要的服务，由SRE开发出专用的诊断工具，能提效提升效率

测试和修复

■经过检查和诊断，通常能够得出一个引起故障的可能原因列表，而后就要通过一系列测试，以找出真正的问题所在

■以下是在测试中常用的一些指导法则

◆先测试最可能的情况

◆设计进行的各测试间应该具有互斥性

◆测试的结果可能存在误导，要注意对结果的二次判定

- 测试有可能存在副作用，它可能会加重故障的后果
- 测试较多时，尽量使用文档记录已经完成的测试和测试的结果

排查思路示例：服务器故障

网站服务器崩溃的可能性原因

- 服务器硬件故障或者系统内核Bug
- 并发的多线程出现了死锁，或者后端数据库崩溃
- 存放业务数据变动的磁盘空间耗尽
- 流量过高，系统超载
  - 服务器硬件配置过低，正常流量增长下的系统超载

    正常的短暂性流量突增
  - 遭受攻击，出现异常流量尖峰

- 触发了应用程序未曾测试出的潜在Bug，例如死循环或内存泄露等导致系统资源耗尽
- 系统参数设置不合理，例如fd数量过低，或者是并发连接数上限过低
- 人为误操作等

处理思路

■若处于“假死”状态，可以kill并restart业务进程；

■分析监控数据，尤其是注意排查宕机前的异常指标数据，比如CPU或内存尖峰等

■查看系统日志，获取详细信息

◆查看/var/log/messages文件，分析宕机前后的系统日志，注意排查错误信息

◆若启用了kdump，还要分析宕机生成的crash文件

◆硬件故障相关的日志通常位于/var/log/dmesg

根因修复和故障复盘

根因定位

■结合监控、日志、链路跟踪相关的记录，通过工具验证、连通性测试、代码复审等步骤完成

■常规的原因多见于隐秘的Bug或各种外力，例如配置变更、发布、资源部署，甚至是误操作等

根因解决

■导致问题的原因可能有很多，常见的可大体归结为可用性问题和性能问题两类

■可用问题需要系统设计时即考虑面向失败的可用措施，而性能问题的解决主要依赖于容量规划、系统扩容以及代码优化等手段

故障复盘

■复盘的目的，既是要沉淀问题处理的经验，也是为改进系统提出合理方案

■复盘时的指导原则

- ◆列举导致故障的所有主要原因
- ◆为各问题找到避免再次出现的解决措施
- ◆为下次处理类似问题提升效率沉淀出经验

故障判定时可遵循的原则

健壮性原则

■系统中的每个部件自身都要面向失败设计，让自身具有一定的自愈能力，例如主备、集群、限流、降级和重试等

■因此，若被依赖方拥有自愈能力或被判定为非核心应用的情况下，依赖方出的问题要自身承担主要责任

第三方默认无责

■系统稳定性一定要做到相对自我可控，而不能完全依赖于外部

■若出故障的系统依赖于第三方服务，例如公有云的各类服务，包括IaaS、PaaS、CDN、DBaaS等，第三方默认无责

■可用措施包括主备、双活、多活等

分段判定原则

■对于复杂问题，可采用的原则，例如发生衍生故障，或者故障蔓延的原因与触发原因不同时，则可以分段分别进行判定