



24 | FMEA方法，排除架构可用性隐患的利器

李运华

- 00:05 / 13:37

我在前面的专栏分析高可用复杂度的时候提出了一个问题：高可用和高性能哪个更复杂，大部分同学都分析出了正确的答案：高可用更复杂一些，主要原因在于异常的场景很多，只要有一个场景遗漏，架构设计就存在可用性隐患，而根据墨菲定律“可能出错的事情最终都会出错”，架构隐患总有一天会导致系统故障。因此，我们在进行架构设计的时候必须全面分析系统的可用性，那么如何才能做到“全面”呢？

我今天介绍的**FMEA方法**，就是保证我们做到全面分析的一个非常简单但是非常有效的方法。

FMEA介绍

FMEA（Failure mode and effects analysis，故障模式与影响分析）又称为失效模式与后果分析、失效模式与效应分析、故障模式与后果分析等，专栏采用“故障模式与影响分析”，因为这个中文翻译更加符合可用性的语境。FMEA是一种在各行各业都有广泛应用的可用性分析方法，通过对系统范围内潜在的故障模式加以分析，并按照严重程度进行分类，以确定失效对于系统的最终影响。

FMEA最早是在美国军方开始应用的，20世纪40年代后期，美国空军正式采用了FMEA，尽管最初是在军事领域建立的方法，但FMEA方法现在已广泛应用于各种各样的行业，包括半导体加工、餐饮服务、塑料制造、软件及医疗保健行业。FMEA之所以能够在这些差异很大的领域都得到应用，根本原因在于FMEA是一套分析和思考的方法，而不是某个领域的技能或者工具。

回到软件架构设计领域，FMEA并不能指导我们如何做架构设计，而是当我们设计出一个架构后，再使用FMEA对这个架构进行分析，看看架构是否还存在某些可用性的隐患。

FMEA方法

在架构设计领域，FMEA的具体分析方法是：

- 给出初始的架构设计图。
- 假设架构中某个部件发生故障。
- 分析此故障对系统功能造成的影响。
- 根据分析结果，判断架构是否需要进一步优化。

FMEA分析的方法其实很简单，就是一个FMEA分析表，常见的FMEA分析表格包含下面部分。

1. 功能点

当前的FMEA分析涉及的功能点，注意这里的“功能点”指的是从用户角度来看的，而不是从系统各个模块功能点划分来看的。例如，对于一个用户管理系统，使用FMEA分析时“登录”“注册”才是功能点，而用户管理系统中的数据库存储功能、Redis缓存功能不能作为FMEA分析的功能点。

2. 故障模式

故障模式指的是系统会出现什么样的故障，包括故障点和故障形式。需要特别注意的是，这里的故障模式并不需要给出真正的故障原因，我们只需要假设出现某种故障现象即可，例如MySQL响应时间达到3秒。造成MySQL响应时间达到3秒可能的原因很多：磁盘坏道、慢查询、服务器到MySQL的连接网络故障、MySQL bug等，我们并不需要在故障模式中一一列出来，而是在后面的“故障原因”一节中列出来。因为在实际应用过程中，不管哪种原因，只要现象是一样的，对业务的影响就是一样的。

此外，故障模式的描述要尽量精确，多使用量化描述，避免使用泛化的描述。例如，推荐使用“MySQL响应时间达到3秒”，而不是“MySQL响应慢”。

3. 故障影响

当发生故障模式中描述的故障时，功能点具体会受到什么影响。常见的影响有：功能点偶尔不可用、功能点完全不可用、部分用户功能点不可用、功能点响应缓慢、功能点出错等。

故障影响也需要尽量准确描述。例如，推荐使用“20%的用户无法登录”，而不是“大部分用户无法登录”。要注意这里的数字不需要完全精确，比如21.25%这样的数据其实是没有必

20% 40%

要的，我们只需要预估影响是 还是 。

4.严重程度

严重程度指站在业务的角度故障的影响程度，一般分为“致命/高/中/低/无”五个档次。严重程度按照这个公式进行评估：严重程度 = 功能点重要程度 × 故障影响范围 × 功能点受损程度。同样以用户管理系统为例：登录功能比修改用户资料要重要得多，80%的用户比20%的用户范围更大，完全无法登录比登录缓慢要更严重。因此我们可以得出如下故障模式的严重程度。

- 致命：超过70%用户无法登录。
- 高：超过30%的用户无法登录。
- 中：所有用户登录时间超过5秒。
- 低：10%的用户登录时间超过5秒。
- 中：所有用户都无法修改资料。
- 低：20%的用户无法修改头像。

对于某个故障的影响到底属于哪个档次，有时会出现一些争议。例如，“所有用户都无法修改资料”，有的人认为是高，有的人可能认为是中，这个没有绝对标准，一般建议相关人员讨论确定即可。也不建议花费太多时间争论，争执不下时架构师裁定即可。

5.故障原因

“故障模式”中只描述了故障的现象，并没有单独列出故障原因。主要原因在于不管什么故障原因，故障现象相同，对功能点的影响就相同。那为何这里还要单独将故障原因列出来呢？主要原因有这几个：

- 不同的故障原因发生概率不相同

例如，导致MySQL查询响应慢的原因可能是MySQL bug，也可能是没有索引。很明显“MySQL bug”的概率要远远低于“没有索引”；而不同的概率又会影响我们具体如何应对这个故障。

- 不同的故障原因检测手段不一样

例如，磁盘坏道导致MySQL响应慢，那我们需要增加机器的磁盘坏道检查，这个检查很可能不是当前系统本身去做，而是另外运维专门的系统；如果是慢查询导致MySQL慢，那我们只需要配置MySQL的慢查询日志即可。

- 不同的故障原因的处理措施不一样

例如，如果是MySQL bug，我们的应对措施只能是升级MySQL版本；如果是没有索引，我们的应对措施就是增加索引。

6.故障概率

这里的概率就是指某个具体故障原因发生的概率。例如，磁盘坏道的概率、MySQL bug的概率、没有索引的概率。一般分为“高/中/低”三档即可，具体评估的时候需要有以下几点需要重点关注。

- 硬件

硬件随着使用时间推移，故障概率会越来越高。例如，新的硬盘坏道几率很低，但使用了3年的硬盘，坏道几率就会高很多。

- 开源系统

成熟的开源系统bug率低，刚发布的开源系统bug率相比会高一些；自己已经有使用经验的开源系统bug率会低，刚开始尝试使用的开源系统bug率会高。

- 自研系统

和开源系统类似，成熟的自研系统故障概率会低，而新开发的系统故障概率会高。

高中低是相对的，只是为了确定优先级以决定后续的资源投入，没有必要绝对量化，因为绝对量化是需要成本的，而且很多时候都没法量化。例如，XX开源系统是3个月故障一次，还是6个月才故障一次，是无法评估的。

7.风险程度

风险程度就是综合严重程度和故障概率来一起判断某个故障的最终等级，风险程度 = 严重程度 × 故障概率。因此可能出现某个故障影响非常严重，但其概率很低，最终来看风险程度就低。“某个机房业务瘫痪”对业务影响是致命的，但如果故障原因是“地震”，那概率就很低。例如，广州的地震概率就很低，5级以上地震的20世纪才1次（1940年）；如果故障的原因是“机房空调烧坏”，则概率就比地震高很多了，可能是2年1次；如果故障的原因是“系统所在机架停电”，这个概率比机房空调又要高了，可能是1年1次。同样的故障影响，不同的故障原因有不同的概率，最终得到的风险级别就是不同的。

8.已有措施

针对具体的故障原因，系统现在是否提供了某些措施来应对，包括：检测告警、容错、自恢复等。

- 检测告警

最简单的措施就是检测故障，然后告警，系统自己不针对故障进行处理，需要人工干预。

- 容错

检测到故障后，系统能够通过备份手段应对。例如，MySQL主备机，当业务服务器检测到主机无法连接后，自动连接备机读取数据。

- 自恢复

检测到故障后，系统能够自己恢复。例如，Hadoop检测到某台机器故障后，能够将存储在这台机器的副本重新分配到其他机器。当然，这里的恢复主要还是指“业务”上的恢复，一般不太可能将真正的故障恢复。例如，Hadoop不可能将产生了磁盘坏道的磁盘修复成没有坏道的磁盘。

9.规避措施

规避措施指为了降低故障发生概率而做的一些事情，可以是技术手段，也可以是管理手段。例如：

- 技术手段：为了避免新引入的MongoDB丢失数据，在MySQL中冗余一份。

- 管理手段：为了降低磁盘坏道的概率，强制统一更换服务时间超过2年的磁盘。

10. 解决措施

解决措施指为了能够解决问题而做的一些事情，一般都是技术手段。例如：

- 为了解决密码暴力破解，增加密码重试次数限制。
- 为了解决拖库导致数据泄露，将数据库中的敏感数据加密保存。
- 为了解决非法访问，增加白名单控制。

一般来说，如果某个故障既可以采取规避措施，又可以采取解决措施，那么我们会优先选择解决措施，毕竟能解决问题当然是最好的。但很多时候有些问题是系统自己无法解决的，例如磁盘坏道、开源系统bug，这类故障只能采取规避措施；系统能够自己解决的故障，大部分是和系统本身功能相关的。

11. 后续规划

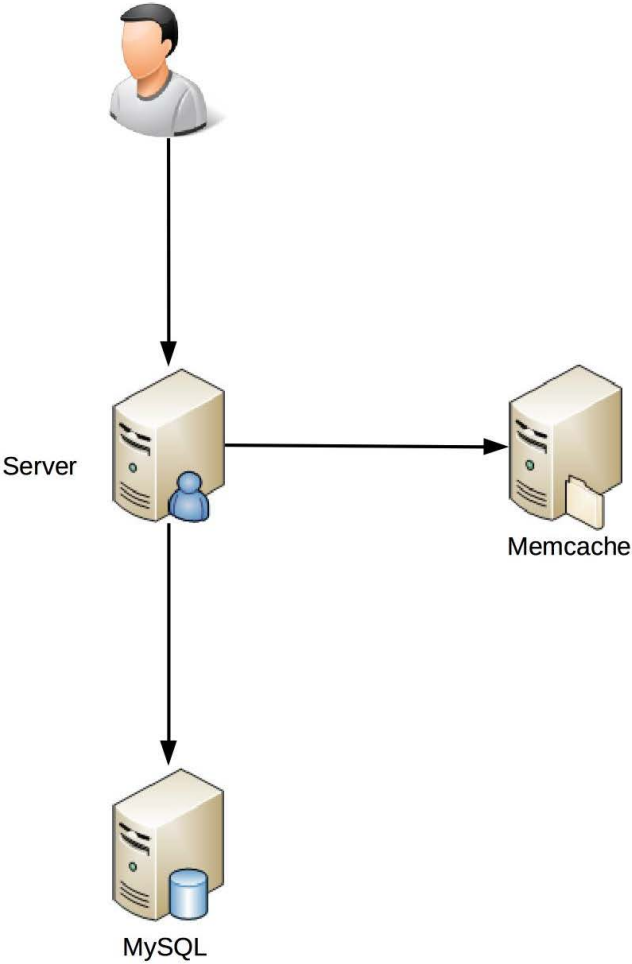
综合前面的分析，就可以看出哪些故障我们目前还缺乏对应的措施，哪些已有措施还不够，针对这些不足的地方，再结合风险程度进行排序，给出后续改进规划。这些规划既可以是技术手段，也可以是管理手段；可以是规避措施，也可以是解决措施。同时需要考虑资源的投入情况，优先将风险程度高的系统隐患解决。

例如：

- 地震导致机房业务中断：这个故障模式就无法解决，只能通过备份中心规避，尽量减少影响；而机柜断电导致机房业务中断：可以通过将业务机器分散在不同机柜来规避。
- 敏感数据泄露：这个故障模式可以通过数据库加密的技术手段来解决。
- MongoDB断电丢数据：这个故障模式可以通过将数据冗余一份在MySQL中，在故障情况下重建数据来规避影响。

FMEA实战

下面我以一个简单的样例来模拟一次FMEA分析。假设我们设计一个最简单的用户管理系统，包含登录和注册两个功能，其初始架构是：



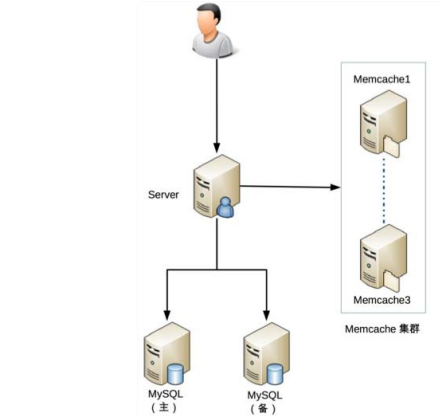
初始架构很简单：MySQL负责存储，Memcache（以下简称MC）负责缓存，Server负责业务处理。我们来看看这个架构通过FMEA分析后，能够有什么样的发现，下表是分析的样例（注意，这个样例并不完整，感兴趣的同学可以自行尝试将这个案例补充完整）。

功能点	故障模式	故障影响	严重程度	故障原因	故障概率	风险程度	已有措施	规避措施	解决措施	后续规划
登录	MySQL无法访问	当MC中无缓存时，用户无法登录。预计有60%的用户	高	MySQL服务器断电	中	中	无	无	无	增加备份MySQL
登录	MySQL无法访问	同上	高	Server到MySQL的网络连接中断	中	中	无	无	无	MySQL双网卡连接
登录	MySQL响应时间超过5秒	60%的用户登录时间超过5秒	高	慢查询导致MySQL运行缓慢	高	高	慢查询检测	重启MySQL	无	不需要
登录	MC无法访问	所有用户都收到MySQL查询信息。一旦，但MySQL压力还是能够会增大，响应会变慢	低，虽然慢，但MySQL压力还是能够会增大，响应会变慢	MC服务器断电	中	低	无	无	无	MC集群
注册	MySQL无法访问	用户无法注册	低，因为新用户注册的用户每天大约只有100个	MySQL服务器断电	中	低	无	无	无	无，因为即使增加备份机器，也无法作为主机写入
注册	MC无法访问	无影响，用户注册流程不操作MC	无	MC服务器断电	中	低	无	无	无	不需要

经过上表的FMEA分析，将“后续规划”列的内容汇总一下，我们最终得到了下面几条需要改进的措施：

- MySQL增加备机。
- MC从单机扩展为集群。
- MySQL双网卡连接。

改进后的架构如下：



小结

今天我为你讲了FMEA高可用分析方法，并且给出了一个简单的案例描述如何操作。FMEA是高可用架构设计的一个非常有用的方法，能够发现架构中隐藏的高可用问题，希望对你有所帮助。

这就是今天的全部内容，留一道思考题给你吧，请使用FMEA方法分析一下HDFS系统的架构，看看HDFS是如何应对各种故障的，并且分析一下HDFS是否存在高可用问题。

欢迎你把答案写到留言区，和我一起讨论。相信经过深度思考的回答，也会让你对知识的理解更加深刻。（编辑乱入：精彩的留言有机会获得丰厚福利哦！）



yason li	
请教老师： 根据网上查到的资料发现，经过多年的演进FMEA从定性和定量两个维度分别延伸出了FMECA和FMEDA，实际进行架构分析时是不是使用FMECA会更好一些？还有就是FMEA分析貌似比较适合系统中单点故障的评估。如果系统比较复杂完成故障的原因有可能是多点同时互相影响那么评估时候是不是使用FTA 故障树分析更合适呢？ 作者回复	2018-06-21
1. 我也是看了你的评论才知道还有FMECA和FMEDA，我查了一下资料，其实都是FMEA的扩展版，其实我们在具体实践的时候，分析纬度已经包括FMECA中的危害性分析(文中的“严重程度”)，以及FMEDA中的诊断分析(文中的“已有措施”) 2. FTA我理解是一种故障影响分析方式，例如FMEA中分析“故障影响”	2018-06-23
Hook	2018-06-21
请教老师： FMEA 实战表格中（正文的倒数第二张图），故障原因是“MySQL 服务器断电”对应了 2 个功能点，分别为：登录、注册。其中，“登录”功能点它的“后续规划”列中写的是“增加备份 MySQL”，而“注册”功能点对应的是“无”，因为即使增加备份机器，也无法作为主机写入”。 我的问题是： “注册”功能点对应的后续规划可不可以是：“增加 mysql 主从切换功能”呢？老师写“无”是从什么角度来思考“后续规划”的。 作者回复	2018-06-21
确实可以这么做，但有些场景没必要做这么复杂，注册不了段时间注册就可以了💎💎	
邱昌	2018-07-11
老师，Mysql响应慢超过5秒，是如何做出影响60%的结论？ 实际应用中这种如何评估？ 作者回复	2018-07-11
根据业务具体情况评估，不是绝对的	
13683815260	2018-07-07
这个名词虽然不熟悉，不过这几个步骤在风险管理里面很熟悉。 风险程度=严重程度*故障概率等。 13683815260	2018-07-07
HDFS可以从 网络原因分片传输 存储分片大小 DataNode故障 NameNode故障 如果两个NameNode都出现问题这个时候就需要人工介入了吧。 作者回复	2018-07-09
是的	
王磊	2018-06-21
hdfs 对于datanode failure, hdfs的应对方式是数据存在多份拷贝，当某个结点down掉后，系统会检测到 under replication, 数据的其他拷贝会在其他可用结点上再增加一份拷贝； 对于那么node failure, hdfs 的应对方式是secondary namenide。 作者回复	2018-06-23
两个namenode挂掉呢？	
云学	2018-06-21
我们公司也在用这套方法，这套方法其实就是多设计一些异常case，系统能够依然保持稳定，当然正常的case也是很重要的。 作者回复	2018-06-23
知道这套方法论的公司不多，说明你们公司技术比较厉害💎💎	

