

通信亚健康自愈决策算法设计和验证

摘要 针对微服务架构跨 POD 通信亚健康问题，业务层可从选择部分受影响的 POD 进行隔离，以避开通信亚健康链路，实现不依赖底层能力的业务自愈。在选择隔离 POD 集合时，需满足业务冗余和容灾要求，且尽可能降低隔离带来的业务影响，提升自愈效率。对此全遍历搜索求解耗时较长，无法满足快速自愈要求。我们通过对问题的形式化，转化和建模为带约束的最优化问题，在通用求解算法之上针对性设计迭代求解算法，在保障最优性的前提下提升了求解效率。

关键词 网元，通信亚健康，最优化

1 引言

5G 网元采用微服务架构，微服务实例部署到容器中，一个或多个容器组成 POD，跨不同 POD 的微服务实例之间通过网络通信。受网络设备拥塞、故障等因素影响，POD 间通信可能出现亚健康状态，易导致业务 KPI 下降。针对这类 POD 对通信亚健康场景，业务层可选择部分受影响的 POD 进行隔离，以避开通信亚健康链路，实现不依赖底层能力的业务自愈。在通信亚健康场景下部分实体间的连接对处于亚健康状态，POD 隔离方案影响业务自愈的效果和耗时。为了降低对业务的影响，POD 隔离恢复可以分成多批次进行。此时面临如何选择最优的实体集合进行隔离恢复的问题。在多种约束条件和寻优目标之下，全遍历搜索效率较低。如遍历所有 POD 子集，按约束条件和优化目标进行筛选和排序。时间复杂度为 $O(n2^n)$ ，执行时间过长，无法满足业务快速自愈的要求。

基于对问题的理解，我们首先对问题描述和选择规则进行形式化，后建立带约束的最优化问题模型，使用迭代算法进行求解，在保障求解最优性的前提下提升了求解效率。本文如下组织后续章节。第2节介绍了我们对问题建立的合理化假设以及在此基础上的形式化。第3节给出了带约束的最优化问题模型和适用于本问题的求解算法。第4节描述了实验方案与数据生成策略，并展示了实验评估结果。第5节对本文进行了总结。

2 问题形式化

2.1 问题描述

一系列 POD 在某网络拓扑结构上，一部分 POD 属于业务链路。POD 类型分主备和负荷分担两种，

其中负载分担类型 POD 有冗余数和容灾数。每种 POD 间存在双向通信关系，部分通信关系（POD 对）处于亚健康状态。

自愈决策算法输入网络中的 POD 信息：是否属于业务链路，类型，冗余数、容灾数（仅针对负载分担类型）；以及当前的亚健康 POD 对集合。以通信亚健康的源和目的 POD 集合作为全集，从中根据如下选择规则选出待恢复的 POD 子集，要求所选 POD 子集满足约束条件（规则 1/2/3），且按优化目标（优先级：规则 4>5>6>7）排序后得到解。

1. 主备 POD，不允许主备同时隔离
2. 负荷分担 POD，被隔离数小于冗余数
3. 负荷分担 POD，被隔离数小于冗余 + 容灾数
4. 覆盖的通信亚健康链路数量最多（覆盖面最大）
5. 隔离批次最少（自愈恢复最快）
6. 优先隔离不承载业务链路的 POD 类型
7. 隔离 POD 数量最少（业务影响范围最小）

2.2 假设简化

假设 1. 主备类型 POD 包含一个主 POD，和一个备 POD。

假设 2. 冗余和容灾在优化目标上没有区别。

假设 3. 在不发生 Pod 隔离的情况下，链路状态只会从健康变为亚健康，而不会从亚健康自动恢复到健康。隔离期间，无新的突发故障或因自愈操作而产生新的故障点，即随隔离操作，亚健康通信对减少。

由假设 1，可将主备类型 POD 转换成冗余数为 1 的负荷分担类型 POD。

由假设 2，规则 2 和规则 3 唯一区别是不等式右侧是否有容灾数，满足 2 必满足 3，可视容灾为

冗余的一部分，合并为一条规则。可通过两次求解，变更输入的冗余数来优先规则二。

由假设 3，算法能够正确得知故障随隔离方案的变化情况，故可在单批次隔离无法满足约束时规划多批次隔离，一次计算出多批隔离策略，分批实现故障自愈，减少重复计算，节省资源。

2.3 规则形式化

记所有 POD 的种类构成集合 I ，第 $i \in I$ 类 pod 构成集合 P_i ，此类 POD 的冗余数为 $0 \leq R_i < |P_i|$ 。由此定义 $P = \bigcup_{i \in I} P_i$ 表示全部 POD 构成的全集， $M \subseteq P$ 表示属于业务链路的所有 pod 构成的集合 (major)。 $E \subseteq P \times P$ 表示 POD 通信亚健康通信 (有序) 对构成的集合，满足通信对只存在于不同类 POD 之间。

我们首先考虑单批次隔离的问题建模，忽略规则 5，后推广到多批次隔离。算法输入 P, R, M, E ，输出每一类 POD 中下一批需隔离的 POD 集合 S_i ，有 $S_i \subseteq P_i$ 。令 $S = \bigcup_{i \in I} S_i$ 表示下一批需隔离的所有 POD。

由假设 1 和假设 2，可统一处理 POD 类别和冗余与容灾数，将规则 1、2、3 形式化为约束条件 1a。规则 4 要求覆盖的通信亚健康链路数量最多，可形式化为最大化条件 1b。规则 6 要求优先隔离不承载业务链路的 POD 类型，可形式化为最小化条件 1c。规则 7 要求隔离 POD 数量最少，可形式化为最小化条件 1d。

$$\forall i \in I, |S_i| \leq R_i \quad (1a)$$

$$\max |\{(x, y) \in E : x \in S \vee y \in S\}| \quad (1b)$$

$$\min |S \cap M| \quad (1c)$$

$$\min |S| \quad (1d)$$

3 方法设计与分析

3.1 最优化问题建模

设 $i \in P, x_i = [i \in S] \in \{0, 1\}$ 表示 i 所表示的 POD 是否被选入隔离集合。另设隔离批次数量为 k 。我们使用 x_i, k 表达约束条件和最优化条件。

使用 k 个批次，则意味着最多第 i 类 POD，最多可以选 kR_i 个进行隔离 (拆成每次隔离 R_i 个)。故约束条件表达为公式 2a。覆盖的通信亚健康链路数量 (最大化) T_1 表达为公式 2b，隔离批次 (最小化) T_2 表达为公式 2c，被隔离业务链路 POD 数量 (最小化) T_3 表达为公式 2d，被隔离的 POD 数

量 (最小化) T_4 表达为公式 2e。令 c_1, c_2, c_3, c_4 表示优化目标的权重常系数，有 $c_1 > c_2 > c_3 > c_4 > 0$ ，则目标函数 T (最大化) 可表示为公式 2f。

$$\sum_{j \in P_i} x_j \leq kR_i \quad \forall i \in I \quad (2a)$$

$$\begin{aligned} T_1 &= |\{(x, y) \in E : x \in S \vee y \in S\}| \\ &= \sum_{(i,j) \in E} (x_i + x_j - x_i x_j) \end{aligned} \quad (2b)$$

$$T_2 = k \quad (2c)$$

$$T_3 = |S \cap M| = \sum_{i \in M} x_i \quad (2d)$$

$$T_4 = |S| = \sum_{i \in P} x_i \quad (2e)$$

$$T = c_1 T_1 - (c_2 T_2 + c_3 T_3 + c_4 T_4) \quad (2f)$$

由此我们可以将此问题建模为以下最优化问题。

$$\begin{aligned} \max \quad & c_1 \sum_{(i,j) \in E} (x_i + x_j - x_i x_j) \\ & - c_2 k - c_3 \sum_{i \in M} x_i - c_4 \sum_{i \in P} x_i \\ \text{s.t.} \quad & x_i \in \{0, 1\} \quad \forall i \in P \\ & \sum_{j \in P_i} x_j \leq kR_i \quad \forall i \in I \end{aligned} \quad (3)$$

3.2 多批次规划求解

对于单批次 $k = 1$ 问题，最优化问题 3 退化为 01 二次规划问题 4，可使用现有规划求解器求解^[1]。

$$\begin{aligned} \max \quad & c_1 \sum_{(i,j) \in E} (x_i + x_j - x_i x_j) \\ & - c_3 \sum_{i \in M} x_i - c_4 \sum_{i \in P} x_i \\ \text{s.t.} \quad & x_i \in \{0, 1\} \quad \forall i \in P \\ & \sum_{j \in P_i} x_j \leq R_i \quad \forall i \in I \end{aligned} \quad (4)$$

对于多批次问题，由于规则 4 (覆盖数量最多) 优先于规则 5 (批次最少)，可找到覆盖边数最多的前提下，最小的批次数 \hat{k} ，固定 $k = \hat{k}$ 后转化为单批次问题 4，后将单批次问题的解拆分为 \hat{k} 批次的解，将选中的隔离 POD 按顺序放进可容纳的批次内即可 (由 $\forall i \in I, \sum_{j \in P_i} x_j \leq kR_i$ 约束条件，拆分方案一定存在)。

对于 \hat{k} 的确定，随着批次 k 增加，最优解可覆盖的亚健康通信对数量会单调增加，直至达一上界 (如覆盖所有亚健康通信对) 后不再增加。可利用

表 1 随机故障实验结果

故障点类型	单点故障	两点故障	三点故障
POD	1.8s / 2.8s / 47M	3.4s / 4.5s / 62M	4.4s / 5.6s / 74M
主机端口	10.2s / 19.5s / 134M	58.6s / 184.9s / 230M	182.4s / 470.5s / 343M
主机设备	21.2s / 30.5s / 253M	83.0s / 349.2s / 423M	166.7s / 454.4s / 645M
TOR 端口	1.7s / 9.5s / 52M	7.9s / 29.2s / 112M	28.2s / 103.1s / 116M
TOR 设备	2.7s / 5.8s / 46M	3.7s / 8.3s / 48M	20.7s / 40.8s / 55M
EOR 端口	4.9s / 12.3s / 89M	7.7s / 20.6s / 107M	55.5s / 264.5s / 144M
任意位置	3.5s / 13.1s / 68M	4.2s / 12.3s / 70M	21.0s / 54.5s / 97M

这一单调性，对 k 进行二分搜索提升查找效率，对每个固定的 $k = k'$ 转为二次规划问题 5 求解得到能覆盖的通信对数量，直至找到最小的 k 覆盖了最多的通信对，最多求解 $\log(|E|)$ 次二次规划问题 5 即可确定 \hat{k} 。

$$\begin{aligned}
 \max \quad & \sum_{(i,j) \in E} (x_i + x_j - x_i x_j) \\
 \text{s.t.} \quad & x_i \in \{0, 1\} \quad \forall i \in P \\
 & \sum_{j \in P_i} x_j \leq k' R_i \quad \forall i \in I
 \end{aligned} \quad (5)$$

4 实验设计与结果

4.1 实验方案

某网元的 POD (A-576pod/3 冗余, B-48pod/1 冗余, C-160pod/1 冗余, D-208pod/1 冗余, E-64pod/1 冗余, 共计 1056 POD) 随机部署在 50 台 Host 上, 每机柜包含 2 台主机。由两个 TOR 交换机连接同一机柜的不同主机的物理网卡 (TOR-0 连接到 Host0 和 Host1 的某个物理网卡)。所有 TOR 的网卡均衡连接到一对 EOR 交换机上, 根据业务要求, POD 间双向通信为 A-B, A-C, A-D, E-B, E-C。

建立网络拓扑后, 我们根据多种可能的故障情况注入单点或多点故障: POD 故障, 主机端口故障, 主机设备故障, TOR 端口故障, TOR 设备故障, EOR 端口故障, 任意位置故障。进一步根据故障情况得到 POD 间通信亚健康概率, 并生成亚健康通信对。

4.2 数据生成

假设 4. 每个 POD 通信对每次通信均随机 (等概率) 选择可行的通信链路。

假设 5. POD 间可能的通信链路, 不会重复经过同一网卡, 且链路长度最短。

基于假设 4, 根据指定的故障点, 统计每个 POD 对间所有可能的通信链路, 和其中经过故障点的链路数量, 两者比值可作为此 POD 通信对是亚健康状态的概率, 并以此生成亚健康 POD 通信对。

基于假设 5, 我们可以将无穷的可能通信链路合理简化为有限和便于计算的数量, 使用最短路算法统计 POD 对间所有可能链路。

4.3 实验结果

实验环境为 Intel(R) Xeon(R) W-2235 CPU @ 3.80GHz, 12 核心, 64 GB 内存, Ubuntu 18.04 系统。使用 Python 3 实现, 二次规划求解器使用 SCIPOpt^[1]。限制单次求解耗时上限为 10 分钟, 内存上限为 1GB, 每种实验配置进行 10 次随机数据生成并求解。表 1 给出了不同故障点类型下的单点、两点、和三点故障的平均耗时, 最大耗时, 以及最大内存占用数据。超过三点故障的测试中出现超时 (耗时大于 10 分钟), 强制结束求解过程, 没有得到结果, 故未在表中列出。

5 结论

本文通过对通信亚健康自愈决策问题的分析和形式化, 将其建模为带约束的最优化问题, 并结合通用求解算法与迭代求解算法解决多批次决策问题, 在保障最优性的前提下提升了求解效率, 在多种随机故障场景下实验验证了方法效果。

参考文献

- [1] BERLIN Z I. Scip[EB/OL]. 2023. <https://www.scipopt.org/>.