**一、导言（背景、Introduction）**

微服务的背景：

Microservice-based systems aims to integrate a suite of smaller services into an application. Each service runs in its own process1 and can evolve independently. These services are typically built around business capabilities and are responsible for their own functionalities

单体应用面临的挑战。改造的迫切性。

微服务的拆分和软件的重构分解很类似，

**目标**

面向数据的、自底向上的微服务化拆分方法，

**二、相关工作、Motivation**

以LeaveWord的拆分过程引出影响单体微服务化拆分的因素、与传统模块化重构的不同

**三、拆分方案生成过程**

假设：所提出的技术认为，单片企业应用程序有三个主要部分：客户端用户界面、服务器端应用程序和数据库。它还认为大型系统是在较小的子系统上构建的，每个子系统都有一套定义明确的业务职责。我们还假设每个子系统都有一个单独的数据存储。

Scenario和trace级别是考虑到最终拆分的微服务应当是一个独立的业务功能聚合体；

而SQL级别更多的是考虑到减少数据的耦合和拆分的开销。

概念定义：Trace、Scenario

改造Kieker监控代码执行路径和SQL语句的执行[2][3]、数据处理与存储、

每条调用链都有唯一的ID标识，即使同一个场景下产生的两条调用链对应的代码执行路径完全相同（即所经过的方法和SQL语句及其顺序完全相同），这两条调用链也是不同的。

**0. SQL/Trace/Scenario的权重计算、包含Table的判断方法**

单个场景的权重定义为，由设计人员根据业务场景的重要度进行设置，默认全部为1。对于一些比较重要或者使用频率很高的业务场景，例如火车站订票系统的订票场景、购物网站的下单流程，可以适当增加权重。本文实验的所有系统，在了解系统功能和业务流程的基础上，分配给各个场景的权重取值范围为[1, 2]。d

单个方法调用链的权重定义为，直接等于这条链所属的场景的权重。

单条SQL语句的权重定义为，等于每个场景权重与该场景调用次数的乘积的累加和，其中定义为第k个场景调用第i条SQL语句的次数：

**1. 两个数据表之间的关联度计算，目前考虑SQL、Trace和Scenario三个级别：**

* 定义，SQL级别的关联度定义为，等于同时操作这两个数据表的SQL语句的累加权重与操作其中任意一个数据表的SQL语句的累加权重之比：
* 定义，调用链级别的关联度定义为，等于同时操作这两个数据表的调用链的累加权重与操作了其中任意一个数据表的调用链的累加权重之比：
* 定义，场景级别的关联度定义为，等于同时操作这两个数据表的场景的累加权重与操作了其中任意一个数据表的场景的累加权重之比：
* 最终得到三个层次的数据表关联度矩阵、和。

每个矩阵有如下特征：

* + 为对称矩阵，矩阵的第i行第j列表示table i与table j之间的关联度
  + 对角线上元素值为1，其他元素的取值区间为[0,1]。

**2. table的共享程度（SharingDegree），同一个table，考虑SQL/Trace/Scenario三个层次：**

* SQL层次的共享：
* 将场景分类、划分成不同的模块, 模块层次的共享：
* 场景层次的共享：
* 综合三个层次的共享度高低（公式），提取共享度高的table（规则），同时根据这些table之间的关联度分组，得到若干组共享table。

**3. 图聚类**

* 输入：
* 三个关联度矩阵融合成一个矩阵，目前
* 调整共享度高的table所连边的权重：
  + 与该table在同一个共享群组中的table，两table之间的边更新为1.5，即这两个table之间的关联度非常强，增加它们被划分到同一个微服务中的概率；
  + 与该table有边相连但不在同一个共享群组中，两table之间的边削减为原来的0.2倍，即降低它们的关联程度，减小被放到同一个微服务中的可能性。
* 最终会得到一个的对称矩阵，矩阵中每个元素的取值在[0,1.7]之间
* 算法实现：

Girvan Newman算法，每次删减边介数最大的边，计算聚类结果和对应的模块度，取模块度最高的作为最优方案输出。

* 输出：

模块度最高的table划分结果。

**4. 拆分开销的计算**

从SQL、Method、Class三个代码层面自底向上进行分析，如果一条SQL操作的所有table最终归属于两个或两个以上不同的微服务，则该条SQL需要进行拆分；同理，：

输出为特定拆分方案需要修改的SQL、Method、Class以及对应的总数量，以及一个最终计算得出的拆分开销分数。

认为SQL拆分只会影响直接调用这条SQL的方法，调用这个方法的方法不需要再拆了。

**5. 模块度与拆分开销的多目标优化选择**

推荐模块度与拆分开销分数综合排名最高的一个拆分方案

**6. 用户反馈调整：**

1. 开始拆分之前，增删共享表群组

2. 每次迭代之间，调整服务数量

3. 可以调整模块度和拆分开销的比例，重新生成推荐方案

4. 确认总体方案之后，手动调整单表所属的群组

**7. 最终的拆分建议：**

1）table的划分方案

2）每个微服务包含的Class/Method/SQL

3）要拆分的Class/Method/SQL应该拆分到哪几个微服务中

-------------------------------------------------------------------------------------

**Evaluation**

RQ0: **正确性**，推荐的拆分方案是否是符合服务的高内聚、低耦合特性的，以及是否符合人的主观判断？

=> [1]中使用的指标、[4]中的coupling criteria、[6]中的质量指标

可以像[1]一样自定义一些指标，例如

**场景、频率的合理性**，对比有场景与频率参数和没有场景与频率参数的最终方案有什么不同，以此说明加入场景概念的优势->**语义耦合度**

[3]中使用的Cargo Tracking系统是DDD设计的sample，可以用来验证拆分的合理性？

RQ1：可行性：3-4个开源项目拆分

**实用性**，这个拆分工具是否可以实际运用到现实的工业项目中？

=> Case study：找一些大点的项目

RQ2.1: 效率，是否比手动分析源码、得出拆分方案更快？

=> 找人找系统做**对比试验**、计时

拆分开销倾向性、人工给出不同倾向性的方案，与工具给的方案对比

**RQ2.2：算法性能测试**，Intel i5 2.5GHz CPU、8GB RAM

大型系统的效率，纯说，人工分析不了

RQ2.3: 拆分方案推荐的贡献（分析动态执行路径的优势、自底向上拆分的优势、反馈调整的优势）

=> **问卷调查/用户反馈**：[4]：user feedback、Pros and Cons

**Threats to Validity**

- 场景频率的设置对于最终结果影响较大，不宜设置得差别过大

- 只实验了几个目标系统，我们的方法可能对其它的系统不具有泛化能力

- 目前改造的监控工具对系统的语言和框架有限制，必须使用Java语言、Spring框架，且使用Mybatis作为ORM框架，如果要对其它语言或框架的系统做监控，需要继续完善监控工具。

- 实验的参与者都是学生而不是专业的软件开发者，可能结果也会有所不同

- 我们假设工具的使用者对拆分系统的业务功能和结构有足够的了解

**意义、贡献**

0. 提出一种根据场景驱动的半自动化拆分方法

1. 拆分粒度从类细化到方法级别，更能降低服务之间的耦合

2. 给出从SQL、方法到类的完整的拆分方案

3. 允许用户反馈调整，综合考虑数据/代码耦合关系和拆分代价，给出一个现实的能够接受的方案

**Future work**

- 对数据的读和写的权重不同

- 对数据的拆分粒度细化到字段级别

- 对代码进行自动化拆分

**可以扩展的点：**

- Neo4j的详细介绍

- 工具的详细使用说明

**参考文献：**

1. W Jin, T Liu, Q. Zheng, et al. Functionality-Oriented Microservice Extraction Based on Execution Trace Clustering. ICWS, 2018: 211-218.
2. A. van Hoorn, J. Waller, W. Hasselbring. Kieker: a framework for application performance monitoring and dynamic software analysis. ICPE, 2012, pp. 247-248.
3. Kieker. http://kieker-monitoring.net/.
4. M Gysel, L Kölbener, W Giersche, et al. Service Cutter: A Systematic Approach to Service Decomposition. ESOCC, 2016, pp. 185-200.
5. Zhong CX, Li SS, Zhang H, Zhang C. Evaluating Granularity of Microservices-oriented System Based on Bounded Context. Ruan Jian Xue Bao/Journal of Software, (in Chinese). http://www.jos.org.cn/1000-9825/5797.htm

附中文参考文献：

[6] 钟陈星, 李杉杉, 张贺, 章程.基于限界上下文的微服务粒度界定研究.软件学报. http://www.jos.org.cn/1000-9825/5797.htm