

履约单能力自闭环建设复盘文档

C-3 创建: 王志浩, 最后修改: 王志浩 06-07 10:52

目录

- 1 项目概况
- 2 目标回顾
- 3 结果评估
 - 3.1 投入成本统计
 - 3.2 关键指标对比
 - 3.3 实际结果评估
- 4 因素分析
 - 4.1 实施过程回顾
 - 4.2 成功因素总结
 - 4.3 失败因素总结
- 5 方法/方案总结
 - 5.1 务虚方法论总结
 - 5.2 务实技术方案总结
 - 5.2.1 数据一致性保证方案
 - 5.2.2 Eagle资源评估方案
 - 5.2.3 加快数据回溯解决方案
- 6 后续计划

1 项目概况

项目名称	履约单能力自闭环建设
项目背景	背景：目前团长履约团队的大多数业务场景（包括订单查询和检索）都是直接使用交易侧的能力，结合过去的产品需求以及产品2022年H1规划和当前系统现状，当前的技术架构已经不能满足团长端迭代和稳定性的诉求，故启动履约单能力自闭环建设项目。 目标：更高效得支撑团长履约业务和技术的迭代，业务上90%以上和交易相关需求都能够快速闭环处理，技术上降低系统复杂度并提升系统的性能和稳定性。
起止时间	2022.02.23 – 2022.05.31
主R人员	RD：王志浩、徐杨、黄诗涵 QA：唐蕴凯、相恒奎
人力成本	RD：50PD QA：10PD
交付情况	进度：比技术文档最开始的进度延迟一个月左右，发生了四次预期外的情况导致延期，其中三次偏客观因素，其中一次偏主观因素。 质量：未产生对业务有影响的故障。
项目wiki主页	20220307-履约单能力自闭环建设技术文档

2 目标回顾

维度	目标描述	衡量标准（数据指标）
业务视角	业务迭代效率的提升：绝大多数需求都可以在履约系统内独立完成，减少跨组协作。	订单相关需求开发效率至少提高30%。
	业务支持能力的提升：一旦订单数据落库到履约系统，产品关于订单相关的需求和分析都能在履约系统内自闭环完成。	90%以上需求都能够在履约系统内闭环处理。
	业务场景的丰富：未来基于订单闭环可以将逆向能力及更多的履约业务场景进行落地。	
技术视角	系统可维护性的提升：团客领域的功能和数据内聚性更高，降低履约系统和交易侧的耦合，之前和交易侧的复杂调用关系降为创建履约订单时的交互。	依赖交易侧的接口数量和调用次数降低80%。
	系统性能&稳定性的提升：减少了和外部的交互，系统的性能和稳定性都能得到提升，避免因为下游波动导致的履约业务波动。	订单相关核心接口tp999优化20%。
	架构持续演进的能力提升：未来履约系统可以在此基础上在架构上更加清晰，数据模型更加丰富，对产品的支持能力更灵活，同时我们有了检索能力可以提供更丰富的业务能力。	

3 结果评估

3.1 投入成本统计

人力成本：

角色	人数	人天
前端	0	0
后端	3	50
测试	2	10
总计	5	60

服务器成本：

新增了Eagle集群，新增了13台16C32G服务器。

3.2 关键指标对比

类目	具体指标	实施前的值 (2022.3.2数据)	实施后值 (2022.5.28数据)	解读
数据库	容量	总空间：2.83 TB 利用率：48.04% 剩余空间：1.47T	总空间：5.74 TB 利用率：40% （增加700G使用量） 剩余空间：3.5T 为防止数据量突然增大，将团长所在数据库从3T都升级为6T的数据库。	增加了一个履约单堆维度数据表，履约单字段进行了扩充，空间使用增长大约700G，上涨符合预期。
	写TPS峰值	1500	2300 （增加800TPS）	增加了履约单堆维度数据表的写入，上涨符合预期。
	读QPS峰值	1100	2500 （增加1400QPS）	对交易的依赖迁移到了对DB的依赖，上涨符合预期。
	履约单单表行数	500w左右	500w左右	履约单数量维持不变
	履约单总行数	500w * 100 = 5亿行	500w * 100 = 5亿行	履约单数量维持不变
	单日慢查询量	150左右	1800左右 （增加1650）	因为查询能力切换到了履约侧，履约侧一期还未增加缓存能力，在二期增加缓存能力解决该问题。
系统服务	依赖交易的接口数	16	7 （减少11）	目前还调用交易侧的接口为： 1. 数据落库时反查交易侧 2. 司团、联络、纸质单4个接口还未迁移 （优化项中进行迁移） 3. 依赖交易侧的动态数据 （优化项逐步推进） 4. 有一些无poi的场景还未迁移，后续迁移到履约Eagle中
	依赖交易单日调用总次数	接近2亿次	8700w左右 （减少1.13亿次）	原因同上，符合预期
	依赖交易峰值QPS	4000左右	1000左右 （减少3000QPS）	原因同上，符合预期
性能	订单列表TP999	325.6ms	286.53ms （优化12%）	因为目前还有部分动态数据依赖交易侧，该优化主要为推动售后得到的优化。 基于新的履约单可以进一步进行优化，目前该优化还未上线，预计性能还能优化10%。
	订单详情TP999	317.47ms	202.12ms （优化36%）	因为目前还有部分动态数据依赖交易侧，该优化主要为推动售后和调用接口切换得到的优化。
	整单提货TP999	180.29ms	161.48ms （优化10%）	因为目前还有部分动态数据依赖交易侧，该优化主要为优化mafka发送得到的优化。
	查询待提货数量TP999	29.04ms	6.78ms （优化76.7%）	待提货数据已完全切换至履约侧，所以性能优化较明显。
	发送提货通知消息TP999	58.8ms	33.14ms （优化43.6%）	待提货数据已完全切换至履约侧，所以性能优化较明显。

3.3 实际结果评估

超出预期的	<div>1. 履约单能力的对外支持：开发过程中和交易侧沟通，交易侧会在后续将履约相关的订单查询需求交付我们来处理，后续履约单将服务更多场景，团长基础需求待接入，需要查询一个门店未来60天有没有已经支付的预售订单；团长治理需求已接入，查询退款的履约单详情。</div> <div>2. 可测性提升：在切换完后订单相关的需求开发中实际体验切换完履约单对于测试的效率提升十分显著，之前订单构建的流程为：用户支付成功的订单（会有所属的团点）-> 物流侧形成物流履约单 -> 然后根据所属的仓（即用户订单的团点对应的履约仓）对应的集单 -> 集单之后会形成调拨单 -> tms进行排队 -> 排队成功回调网格站 -> 网格站回调document -> document调用团长端进行收货单的创建 -> 用收货单生成对应的订单，切换完之后我们直接就可以变更/新增订单相关信息，和上游进行解耦，可以快速构建测试所需场景。</div>
持平预期的	<div>1) 业务视角：</div> <div>1. 业务迭代效率提升：大部分订单相关需求都可以在履约系统内独立完成，比如订单排序功能，迭代效率得以提升。</div> <div>2. 业务支持能力的提升：之前无法无法支持的需求已经用切换后的履约单检索能力进行支持，团长端订单列表和纸质单订单列表排序一致已开发完，目前正在灰度中。</div> <div>2) 业务视角：</div> <div>1. 系统可维护性提升：依赖交易接口目前从16个降低到了7个，订单展示和提货相关功能都依赖于履约自身能力，代码逻辑和复杂性降低，系统可维护性得以提升。</div> <div>2. 系统性能&稳定性的提升：单日调用总次数从2亿次降低到了8700w，qps峰值从4000降低到了1000，订单核心接口这阶段都有超过10%以上的优化，同时避免因为下游波动导致的履约业务波动，系统的性能和稳定性都能得到提升。</div>
不达预期的	<div>1. 增加了预期外的慢查询（单日平均增加1650个慢查询）。</div> <div>2. 最终完成时间比最晚晚了接近1个月。</div> <div>3. 在最终的代码清理阶段发生了一次因为误删代码导致的线上问题：https://coe.mws.sankuai.com/detail/197139（无实际业务影响）。</div>

4 因素分析

4.1 实施过程回顾

	阶段	里程碑	关键策略	完成情况	相关文档
1	方案阶段	2022.02.25 完成方案设计	通过问题剖析、现状说明、行业调研、方案论证来进行初步的方案设计。	按时完成。	20220307-履约单能力自闭环建设技术文档
2		2022.03.08 完成方案第一次评审		按时完成。	2022.3.8-启动会
3		2022.03.14 完成方案最终评审	进行更大范围的调研，对外卖、快驴、买菜类似场景通过各方面的调研来论证本方案的可行性。	方案最终确定时间比预期推迟一周。 因为在项目启动初期虎哥对该方案有存疑的地方，花了一周的时间来进行更详细的调研和评估，最终与虎哥和交易侧达成一致。	2022.3.14-交易履约讨论
4	实施阶段	2022.04.08 完成开发	将开发任务详细梳理，横向拆分为三个阶段：基础建设、数据落地和读切换；纵向拆分为两条支线，一条Mysql开发支线，一条Eagle开发支线，先解耦各自独立开发，再合并起来进行联调测试，每个人的开发目标比较明确，互相影响较小。	开发完成时间比预期推迟一周。 因为为3月-4月之间业务需求增多导致业务压力增大，过程中优先保障业务迭代，既定项目参与人与优先支持产品业务需求，对本项目进度造成一定的影响。	03-详细设计时间线
5		2022.04.11 完成写入部分上线并切换	将Mysql部分和Eagle部分解耦上线，先不关注Eagle数据的准确性问题，最终通过数据回溯做到最终的数据一致和准确。 各部分上线都通过开关关闭上线代码，然后点评实验室开启进行线上回归验证，然后按照团点或者城市按照比例进行灰度并留足观察时间，同时监控系统各个指标，保证平滑切换。	按时完成。	00-写灰度放量过程 创建新项目流程SOP
6		2022.04.22 最近40天全量数据回溯完成	回溯前进行数据容量评估，系统压力评估，回溯配置优化等各个手段，在不影响线上业务的前提下，尽可能快速的完成数据回溯，避免影响下一步计划的实施。	按时完成。	01-数据回溯过程
7		2022.04.28 完成读部分上线	因为调用交易侧接口较多，所以在履约系统中进行切面处理，在调用交易的接口之前都进行灰度判断来控制使用交易接口还是履约DB。	测试完成时间比预期推迟一周。 原因为QA人员变动，QA同学手头业务测试压力较大，导致测试时间比预期推迟一周。	履约单能力闭环整体上线流程
8		2022.05.12 开启全量对比	在对比和切换过程中按照poi和方法来进行对比和切换，整个切换过程中对线上监控定时查看分析不一致情况。	开启全量对比比预期推迟3天。 因为在对比过程中出现较多不一致情况，最开始判断失误，以为解决完当前不一致的问题即可进行下一步，结果因为各种预期外的对比不一致的问题的	20220506-接口对比失败总结
9		2022.05.20 完成全量切换		开启全量切换比预期推迟2天。 原因为为保证线上稳定性，增加了每个灰度阶段的观察时间。	20220529-接口迁移进度
10	收尾阶段	2022.05.31 清理收尾	清理闭环过程中产生的中间代码。	清理时间比预期推迟2天，因为为中间上线了一个新的需求，增加了一个新的接口，在对该接口完成对比和切换后进行的代码清理。	

4.2 成功因素总结

	角色	成功因素	关联事例
1	研发	前期调研充分：通过业务痛点、技术现状、建设成本、迭代效率、行业情况、未来展望等多个角度，调研外卖、买菜、快驴、用户交易等多个场景的业务场景和技术选型，通过尽可能多的方面来论证方案的可行性与合理性，各方集思广益使得我们对于履约单的建设目标更加明确，定位更加清晰，实施起来准备更加充分，为后续的实施阶段以及未来更多的业务扩展打下坚实的基础。	关键问题回答 履约单现状调研 履约单与交易单差异说明 团长侧建设履约单情况说明
2		任务拆解合理：将实施工作开展前对开发任务进行抽象分解，将开发部分进行解耦，按照先后顺序和开发任务所属系统将整个项目分为了三个阶段两条支线，两名同学各自主R一条支线并行开发保证开发进度，在开发完成后进行流程合并和联调。 方案论证清晰：每个关键技术点都有技术文档在组内进行详细论证，包括数据库改造方案、Eagle整体设计和高可用设计、灰度切换方案、历史数据回溯方案、数据沉淀方案、数据清理方案、数据一致性等多个关键节点的设计方案，做到最终整个系统的顺利上线。	00-整体流程图 01-数据库改造技术方案 03-通用方案 04-方案讨论
3		线上监控全面：履约单是团长履约核心功能之一，任何错误都可能导致团长履约线上化的失败，所以需要有一个纵观全局的监控系统，所以建设了项目的整体监控大盘，包括Mysql数据库层面（包括读写压力，主从慢查询、各个表的读写监控）、Eagle层面、Makfa消费层面、BCP监控（包括check量和repair量）、业务系统层面、依赖上下游的压力等多个层面进行全方面立体性监控，保证系统所有变化都在掌握之中。	履约切换大盘
4		测试工具辅助：履约单能力自闭环建设整体改动较大，影响团客交接核心流程，手工测试覆盖面小难以对系统质量提供有效保障，同时使用BCP平台辅助验证、自动化用例测试、DIFF工具回归等方法完成对功能验证和系统整体回归。	

4.3 失败因素总结

	角色	失败因素	关联事例	规避措施
1	研发	1. 方案设计阶段排期按照最理想方式排期，没有预留空余buffer防止出现预期外情况。 2. 对比过程中判断失误，误以为通过有限几次上线就可以解决全部对比不一致的问题，没有进行长时间收集对比不一致情况系统解决问题。	整体排期比最开始方案设计时延期1个月左右，其中主观原因导致的一次延期是线上对比过程出现较多预期外对比不一致的情况，导致对比过程时长超过预期。	1. 在客观因素无法避免的前提下，尽量通过提前规划和事项并行的方式来减少客观因素带来的负面影响。 2. 线上对比过程中出现的问题应该在不影响线上业务的前提下尽可能放长对比的时间，多采集问题系统得分析并解决，避免点状处理问题造成的多次上线。 3. 在方案设计阶段对排期更加细化，提前预留一些Buffer防止发生预期外情况。
2		1. 代码清理阶段没有详细核实下线逻辑。 2. 对于新增变更没有详细CR，CR没有落实到位。 3. 对于核心逻辑的重视程度不够，质量意识不足。	代码清理阶段发生一次线上问题（无实际业务影响）：因为误删除收货单状态相关代码导致履约订单没有更新到收货单号： https://coe.mws.sankuai.com/detail/197139	1. 清理/改动代码时仔细核对内部实现细节，检查全部外部调用点。 2. 对参数名称、方法、类命名时遵循开发规范，对于方法秉持“一个方法只干一件事”，避免为后人挖坑。 3. 技术需求改动无论大小必须CR到位。 4. 业务需求与技术需求尽可能分开上线，避免出现回滚会导致优化 / 业务不可用的问题。 5. 对于核心业务逻辑的代码必须添加灰度开关。 6. 上线图稳不图快，越到最后关头越要注意上线质量。
3		1. 没有提前在线下搭建和线上一致的测试库，导致SQL和索引没有按照预期内执行。 2. 线下没有同等数据量的测试库导致需要通过多次上线来调整观察索引变更，耗费时间较长。	数据库慢查询切换前单日增加了1650条： 20220516-慢查询	1. 前期规划中有缓存能力的建设，但是没有量化的数据，如今全部切换完慢查询每天增加了约1650条，在此基础上进行二期的缓存能力建设。 2. 可以提前将线上一个表的数据导到线下，在线下进行SQL的充分测试，避免通过上线来观察SQL和索引变更的效果。
4	测试	1、测试人员由于排期问题未能连贯介入整体项目 2、线下测试交易侧存在太多脏数据干扰测试	1、写部分由履约方向qa介入，读部分由治理方向qa介入 2、测试读部分时，由于存在脏数据问题，只能将测试分支与master数据进行对比，对状态的验证存在干扰	1. 履约治理方向定期互相分享业务，减少临时协助学习成本 2. 建设数据构造工具，提升测试效率

5 方法/方案总结

5.1 务虚方法论总结

- 前期调研要充分：基于业务战略/项目、技术现状/问题作为输入产生的规划往往偏向于短中期，能够较好适应短中期业务和系统上的一些诉求。而中长期的技术规划往往需要基于行业对标来做: 看看标杆怎么做的、来倒推我们目前所处的位置和阶段，制定中长期的技术演进路线，本项目在前期调研中进行了行业内多维度的调研，为后续和未来的技术演进奠定了夯实的基础。
- 多用系统化思维：在数据对比阶段要使统化得处理问题，避免「打地鼠」式得处理问题，线上对比过程中出现的问题应该在不影响线上业务的前提下尽可能放长对比的时间，多采集问题系统得分析并解决，避免点状处理问题造成的多次上线。
- 多使用量化分析：本项目在各个模块在实施前都进行了充分的量化分析，对系统现状和上线后要达到的效果都有具体的指标分析，上线后一旦偏离了事前的量化分析代表不符合预期，需要找出问题，使系统的运行情况一直在预期内。
- 应急方案要完备：每次上线都要考虑上线风险和最坏的情况，针对可能出现的问题都有对应的解决方案，防止线上出现问题后无法快速应对带来更大的影响。
- 监控手段要全面：大型技术项目要有整体的监控大盘，避免观察不全面和丢失观察维度，通过大盘各个维度的组合能够更加全面的观察系统运行时指标，做到系统随时可方便监测。

5.2 务实技术方案总结

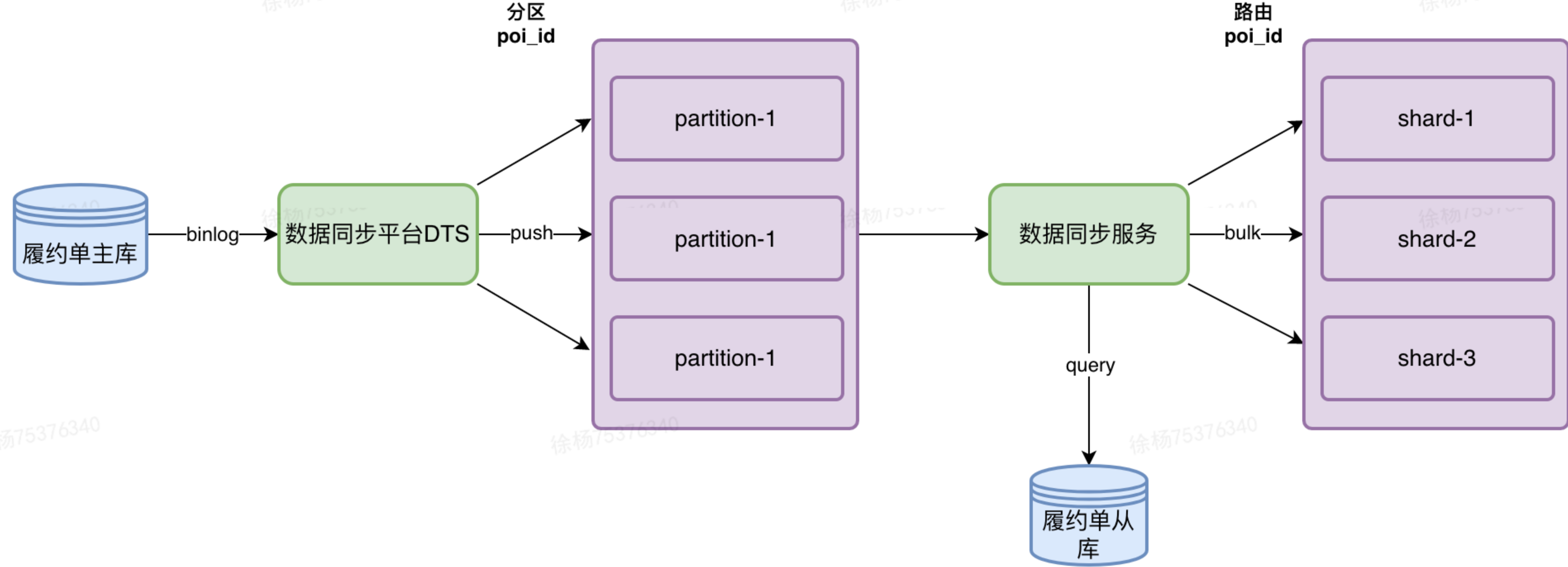
5.2.1 数据一致性保证方案

背景：履约单能力自闭环建设项目通过履约单产生的binlog，将订单数据同步到Eagle中。Eagle中的订单数据直接提供给团长使用，而团长是用户履约重要的一个环节。本项目不仅要为团长提供更快的查询性能和能力，更要保证团长查询数据的准确性和一致性，从而保证团长能够「百分百履约」。保证数据同步的准确性、时效性和数据一致性。

在不能对业务DB与ES产生较大的压力情况下实现保证：

- 保障数据一致性：保证Mysql与Eagle订单数据最终一致性。
- 实现数据自修复：发现数据不一致时可以自动修复Eagle中的脏数据。
- 解决数据并发更新：解决数据并发更新时新老数据覆盖问题和版控问题。

要做到如上三点需要在数据同步中和数据同步后两个维度保证数据的顺序性和一致性，数据同步流程如下：

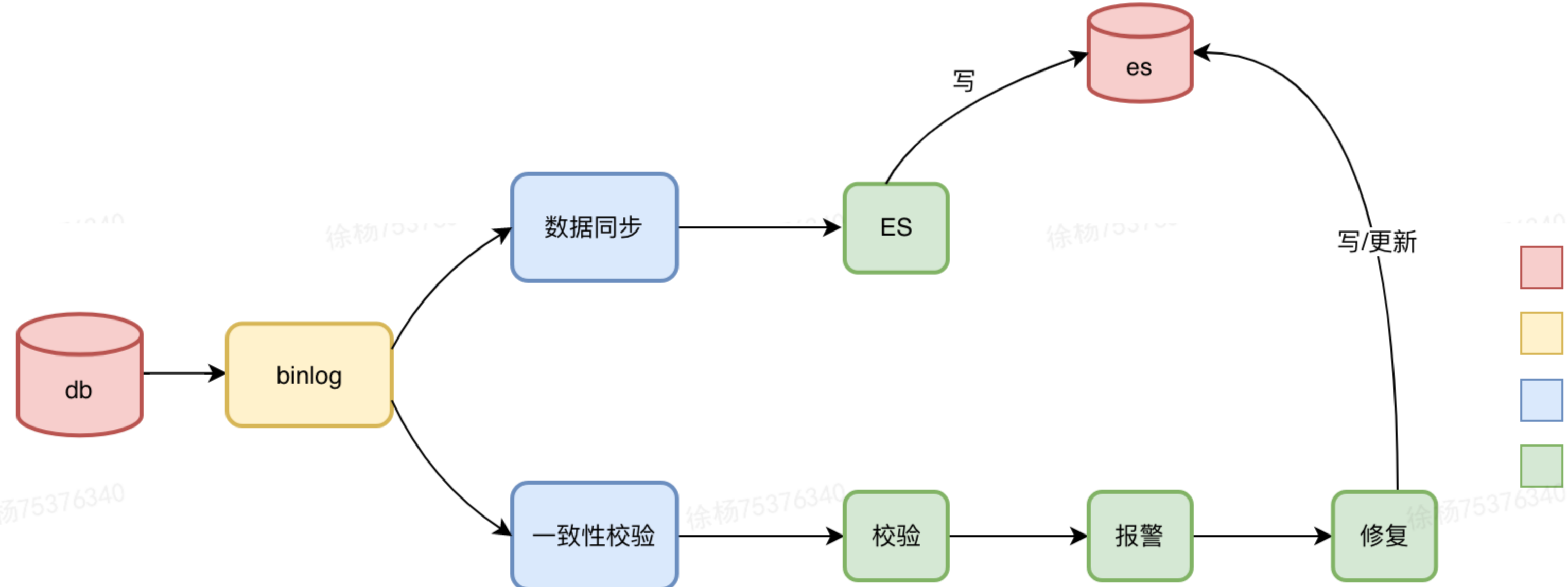


保证数据同步过程中有序性的关键节点为：

- dts同步阶段有序：DTS读取binlog是按日志写入文件的顺序，一行一行读的，为了保持往下游发送的顺序和binlog文件里的日志顺序一致，就需要单线程处理，这样来看性能会比较差。为了提升往下游发送的性能，又不破坏binlog日志的顺序，我们就需要把有冲突的数据串行发送，没有冲突的数据并行发送。分片字段就是用来决定哪些数据是可以并行发送的，比如选择了分片字段是「poi_id」，那么对同一个团点的操作，会由同一条线程来串行处理，对不同团点的操作，可能由多条线程来并发处理。
- mq消费阶段有序：通过将线程池消费线程consumer-consume-thread=1保证消费有序，同时为了加快消费速度，履约单商品数据的消费使用批量消费模式：[Mafka-高阶功能](#)。
- bulk写入阶段有序：采用bulk写入机制可以保证写入的有序性。
- 保障数据同步并发更新：通过Es自带的乐观锁机制来保证并发更新时会发生的冲突问题，详见：[es乐观锁相关](#)。

保证数据同步后一致性的关键节点为：

- 发现不一致：依赖BCP的触发流，将db产生的binlog推送到业务代码中，业务代码将通过泛化调用的方式调用目标接口触发相关的一致性校验逻辑。
- 自动修复不一致：如果每次发生不一致的问题都人工介入成本较高，所以需要引入自动修复的方案，在检测到不一致问题后通过BCP泛化调用repair进行自动修复。



5.2.2 Eagle资源评估方案

背景：检索能力的建设范围相对较广，本方案遇到的一个较为实际的问题是Eagle资源如何评估较为合理，如何用最少的资源满足最大的需求是本方案一个较关键的地方。

以下为各种数据存储的对比：

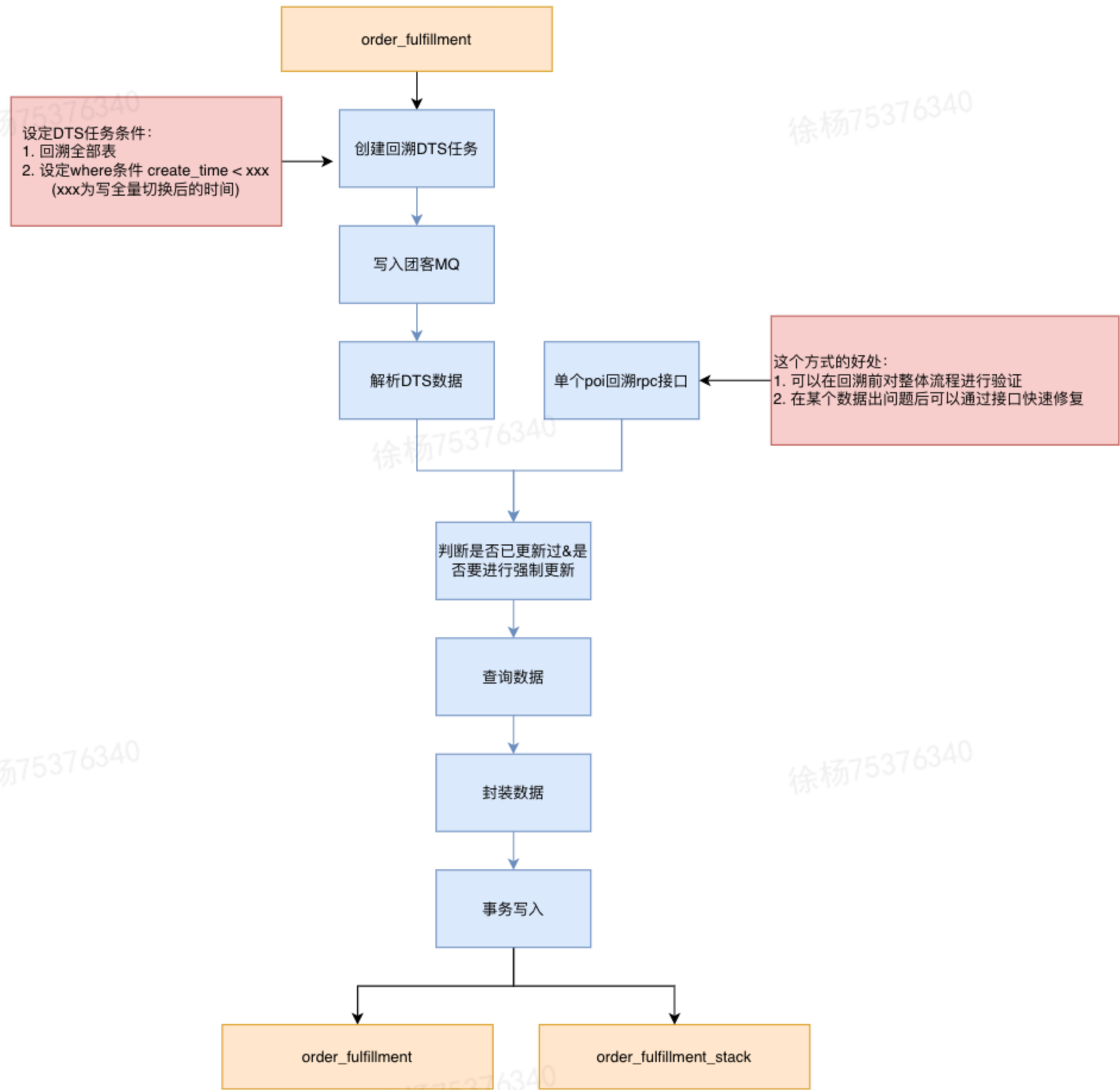
比较项	Blade 2.0	ES	Mysql
存储方式	行存, 适合 OLTP	Inverted Index, 适合各种检索	行存
扩展性	水平扩展	水平扩展	水平扩展
事务	支持	不支持（插入的数据，不是即可以查询到，建索引有延迟）	支持
副本间一致性	强一致	可配置	强一致
二级索引	支持	支持	支持
全文检索	不支持	支持	从5.6版本开始，InnoDB也支持全文索引（fulltext），从5.7开始新增ngram插件以支持对中文的全文索引
成本	高（归档机型除外）	中	低

基于各个业务方实际使用实践和Eagle官方给出的建议，业务上使用时要关注的指标为：

- 分片大小：单个分片应该介于10G–20G之间，根据此来调整单个索引分片的数量，此配置一定要**提前计算好**，因为在创建索引时就定义了主分片数量大小，对于线上业务后续变更成本会非常高。
- 索引副本数：一般线上业务索引分片数设置为1即可，对于数据丢失容忍的场景索引副本数可以调整为0。
- 磁盘使用率：**磁盘使用率**低于60%，**集群读写性能**是最优，影响ES磁盘空间的因素有以下几点：
 - 副本数量：副本有利于增加数据的可靠性，但同时会增加存储成本。默认和建议的副本数量为1，对于部分可以承受异常情况导致数据丢失的场景，可考虑设置副本数量为0。
 - 索引开销：除原始数据外，ES 需要存储索引、列存数据等，在应用编码压缩等技术后，一般膨胀10%。（_all等未计算）。
 - 操作系统预留：操作系统默认会保留**5%**的文件系统供用户处理关键流程、系统恢复以及磁盘碎片等。
 - ES内部开销：ES 占用约20%的磁盘空间，用于 segment 合并、ES Translog、日志等。
 - 安全阈值：通常至少预留**15%**的安全阈值，避免节点没有充分时间进行调整。
 - 根据以上因素得到：**最小磁盘总大小 = 源数据大小【例如业务数据库表的物理大小】 * 3.4。**
 - 磁盘总大小 = 源数据 * (1 + 副本数量) * (1 + 索引开销) / (1 – 操作系统预留空间) / (1 – ES内部开销) / (1 – 安全阈值) = 源数据 * (1 + 副本数量) * 1.7 = 源数据 * 3.4
 - 现在我们有10台1.2T的数据节点，1.2T * 10 / 3.4 / 0.2 T = 17个月的数据，目前我们可以存最近一年半的履约单数据
- CPU Busy：低于50%是安全的，数据回溯量级最大情况下目前我们最高用到8%。
- disk.io：低于70%是安全的，在回溯过程中最高用到70%，回溯完成后日常峰值为40%。

5.2.3 加快数据回溯解决方案

背景：在本项目中进行了多次历史数据回溯行为，在如何加快数据回溯上有一些可供参考的地方。



提高回溯QPS的一些手段：

- 消费者均衡模式partition.assign.mode，保障消费机器尽量均衡的分配partition。
- 调高消费线程，设置consumer.parallel.num=3，因为我们的消息回溯场景是订单维度，不用考虑顺序性问题。
- 增加topic partition，从4变更为10。
- 调整消费者生产端分配策略，调整producer.cluster.dispatch.type为全部集群，使所有消费者都能进行消费。
- 提高dts数据同步任务的消息生产线程数，目前设置在15个生产线程。
- 根据业务系统的特性，识别流量规律，配置回溯流量自动变化策略，不影响线上的前提下尽量高效得用系统资源。

注意：

- 在回溯QPS升高的同时要提前周知并实时关注下游压力，防止超过正常业务流量导致下游出现容量问题。
- 要保留一定的缓冲区，防止有突然的流量进入系统导致线上业务发生故障，至少要保留20%的冗余流量处理能力。

6 后续计划

重要程度：由低到高为1星到5星

紧急程度：由低到高为1星到5星

事项优先级：结合重要程度和紧急程度综合考量，由低到高为 P2（平均星以下）、P1（平均3星到4星）、P0（平均4星以上）

	计划事项	事项描述	优先级	跟进人	预计达成时间	Ones跟进
1	履约单历史数据同步到Blade中	将履约依赖交易的历史数据切换到履约侧，最近6个月的数据都能够通过履约侧进行查询。	P1	@ 王志浩 @ 徐杨	2022.6.30	
2	履约单QDC工具建设	一期为通过更改履约单的各种数据来快速构建测试所需数据； 二期为通过履约单的构造和编排构造实际业务中的各种场景进行快速回归测试。	P1	@ 唐蕴锴	2022.6.30	
3	优化履约订单模型	履约单模型中加入退款、佣金数据； 将依赖交易侧的动态数据逐步落库到履约侧。	P2	@ 王志浩 @ 徐杨		
4	优化履约单状态机流转机制	目前履约单状态机变更存在重复变更的问题，团长提货后更新了交易侧和履约侧，履约侧还会订阅交易侧的提货状态，导致状态更新机制重复。	P2	@ 王志浩		
5	缓存能力建设	目前查询Mysql产生了预期外的慢查询情况，需要建立缓存机制或者类似手段解决Mysql的慢查询问题。	P2	@ 徐杨		
6	降级能力建设	目前履约单检索能力无降级能力，在Eagle挂掉了无backup机制，需要建立对应的降级和应急能力。	P2	@ 王志浩		
7	迁移司团、联络和纸质单依赖交易的服务到团客服务	目前团客相关的能力已经闭环，但是履约侧其他服务目前还依赖于交易侧的能力，这部分需要迁移到团客服务。	P2	@ 王志浩		

🔒 仅供内部使用，未经授权，切勿外传



暂无赞赏，鼓励一下

评论(0)

最早 ▾ | 浏览 69 次 共 17 人浏览



写点你要说的