+

软件设计说明书

SSC1.0.1.0（上海 SC-性能数据采集）

文档版本号：		文档编号：	
文档密级：	保密	归属部门/项目：	
产品名：		子系统名：	
编写人：		编写日期：	

修订记录：

版本号	修订人	修订日期	修订描述
V1.0A	EPG 团队	2013-03-27	初稿
V1.0B	陈凌	2013-9-23	增加需求跟踪
V2.0	李雪峰	2023-03-10	参考公司的标准模板，对于动环项目进行补充和裁剪 2.1　概述，增加对重要需求点的列表阐述，避免设计时遗漏 2.4　模块间交互图，为了避免把大量的逻辑揉在一个时序图，分成子章节分别对业务流程进行设计
V2.1	孙科	2023-07-20	根据部门设计准入标准，增加执行检查点 增加 2.6 公共设计 增加 2.5 影响范围评估 增加 2.9.3 网络资源需求梳理 增加 2.9.4 安全性检查

派发清单：

发文人/部门	日期	电话/传真

受文人/部门	动作类型*	日期	电话/传真

*动作类型：批准、审核、通知、归档、参与会议，其它（请说明）

目 录

1 简介

1.1 目的

[本文档是对方案设计提供一个高层和底层的综合描述，有些系统无需将设计分为高层和底层两个阶段，而是直接用一个设计阶段完成高层和底层设计的工作，则可采用此模板。]

软件设计的目的是根据产品需求、软件需求文档中对需求的描述，结合架构设计文档中对方案的描述，对具体实现方案进行细化，精确到各模块的功能、具体的实现及其模块之间的交互，明确模块中如何通过源代码实现相关的接口和功能，描述其中关键的对外接口及内部函数的实现，关键的全局变量，使用到的数据库对象（包括表、数据、存储过程等）。本文档对编码和单元测试工作进行直接指导，并对测试工作起到辅导作用。

其他情况请另外说明。]

1.2 文档范围

[例如，本文档适用的产品、模块，覆盖的范围等，受这份文档影响的相关模块等，不在该文档覆盖范围内的但可能引起疑义的问题。]

1.3 预期的读者和阅读建议

[说明此文档的阅读对象，对读者的要求。简要说明此文档中其它章节包含的内容与文档组织方式，对于不同读者的阅读方式建议。

如：

目标读者是系统最终用户、系统分析员、项目经理、产品经理、市场人员等。

此文档的第 2 章描述…..]

本文档组织方式：

第一章 简介，描述文档的目的；

第二章 描述设计原则，开发人员在开发过程中必须遵循这些原则；

第三章 描述设计策略；

第四章 描述模块重用设计，设计人员和开发人员需要根据这部分的描述重用以前的设计并关注可供将来重用的设计部分；

第五章 方案设计部分描述设计方案，包括逻辑结构、模块划分、模块间关系及交互图、状态图等设计图例，共设计人员和开发人员了解方案；

第六章 模块设计，详细描述各个模块的具体设计及其单元设计，可指导开发人员进行模块设计和相关测试工作；

第七章 系统容错处理，描述可能的错误信息和处理措施；

第八章 描述系统可维护性设计；

第九章 描述设计的假设前提；

第十章 描述设计中的风险，相关人员应当关注风险对产品可能造成的影响；

第十一章 附录。

1.4 参考文档

[适当时，提供相关的包含文档及参考文档。

软件设计说明书的参考文档应当包括但不限于：产品需求说明书，软件需求说明书，架构设计说明书，数据库设计说明书，高层设计说明书、设计规范、编码规范等；

同时，文档中说明为引用、参考的文档也应该在这里列出。

参考文档请按包含、相关的关系分别在下面列出。]

1.4.1 包含文档

包含的文档名称及其版本	文档路径	对应的章节名称

［包含文档：作为本软件设计的一部分，不可分割的组成部分，读者阅读本设计说明书时必须同时也阅读的文档。如数据库设计说明书（如果数据库设计有变更）。

通常情况下，软件设计说明书没有包含文档。］

如有接口文档该章节为必填，请补充接口文档名称及 SVN 归档路径

【检查点：系统内部接口】是否涵盖内部接口调用设计内容，该设计内容是否清晰合理，依赖耦合性是否较低，是否存在相互依赖的网状调用、是否存在同在同层级调用、是否存在不该依赖的接口调用等

【检查点：外部系统接口】是否涵盖外部系统接口交互设计内容，该设计内容是否清晰合理，是否复用当前接口，是否迫于压力以对方接口为准新增接口，是否有考虑推动以我方为主提供接口规范，接口协议是否合理，接口安全性是否有考虑，接口性能是否有考虑

1.4.2 相关文档【必填】

参考的文档名称及其版本	文档路径	对应的章节名称

[相关文档：作为引用而包含的文档。读者在阅读本设计说明书时如果有必要可以参考阅读的文档。如产品需求说明书、设计规范文档等]

如有该章节为必填，请补充需求文档名称及 SVN 归档路径，如非独立需求文档请补充具体章节；如涉及外部接口文档请一并补充

2 方案设计

2.1 概述

性能数据采集。

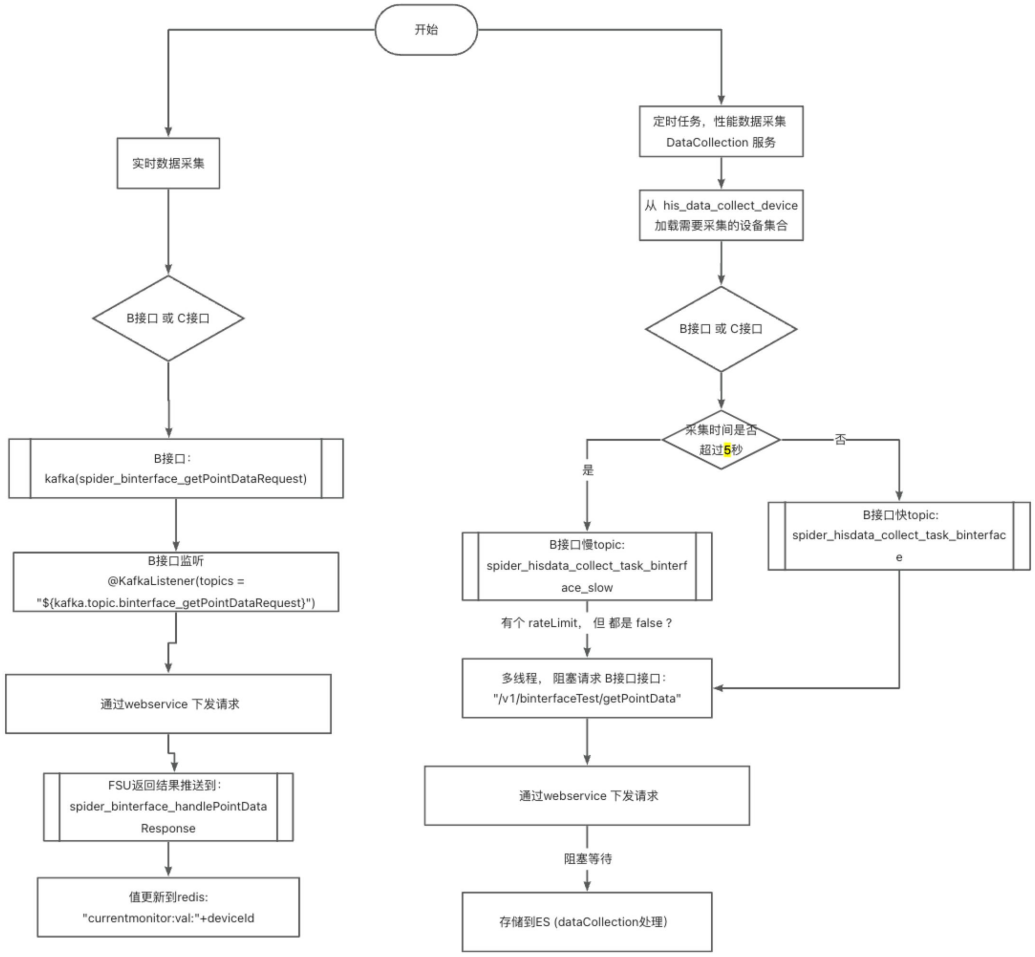
解决以下问题：

（1）当发起大量的性能数据采集时，会对实时数据查询（监控视图）造成影响（C 接口接入的设备明显）

（2）采集较慢或无法采集的设备会造成阻塞，导致积压，能够正常采集的设备也无法较快响应。

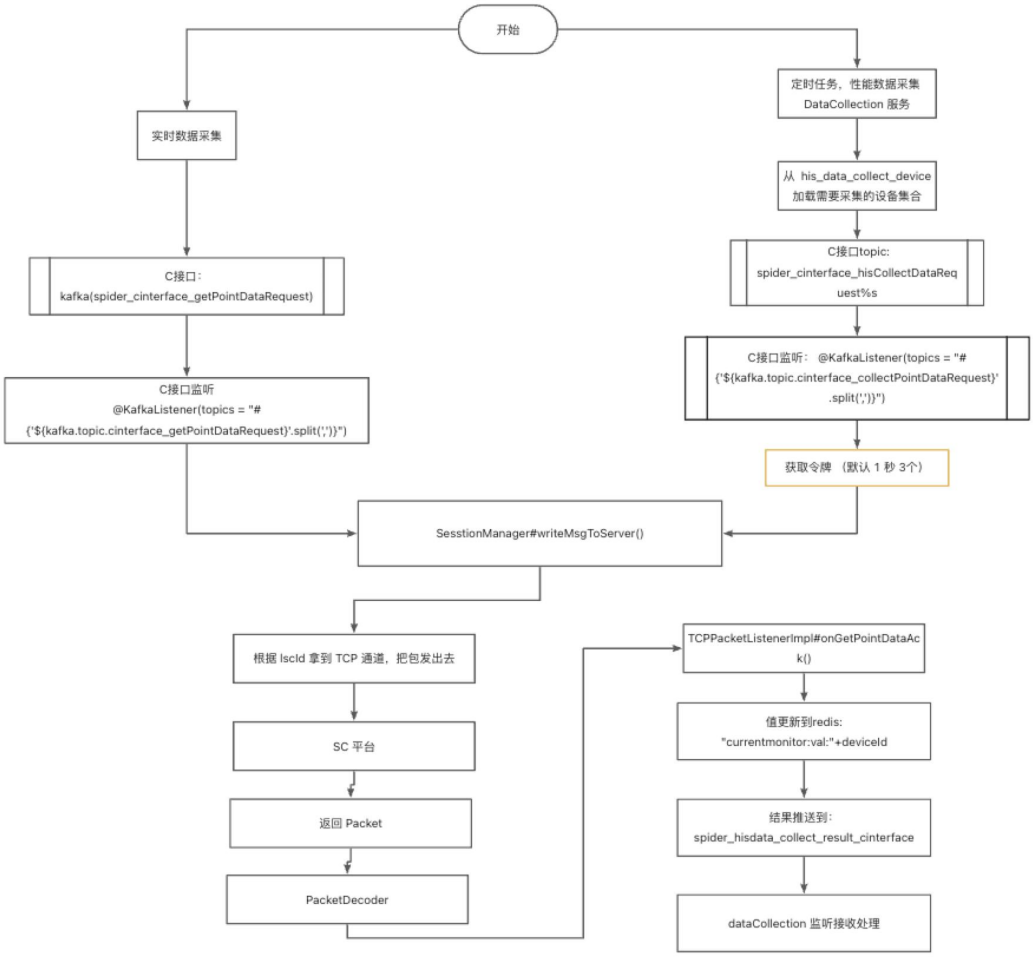
2.1.1 实时数据采集与定时性能数据采集-现有逻辑：

B 接口-现有逻辑



比对项	实时采集	定时采集
topic	spider_binterface_getPoint DataRequest	B 接口快 topic: spider_hisdata_collect_task_binterface B 接口慢 topic: spider_hisdata_collect_task_binterface_slow
限速	不限	令牌桶限速

C 接口现有逻辑：



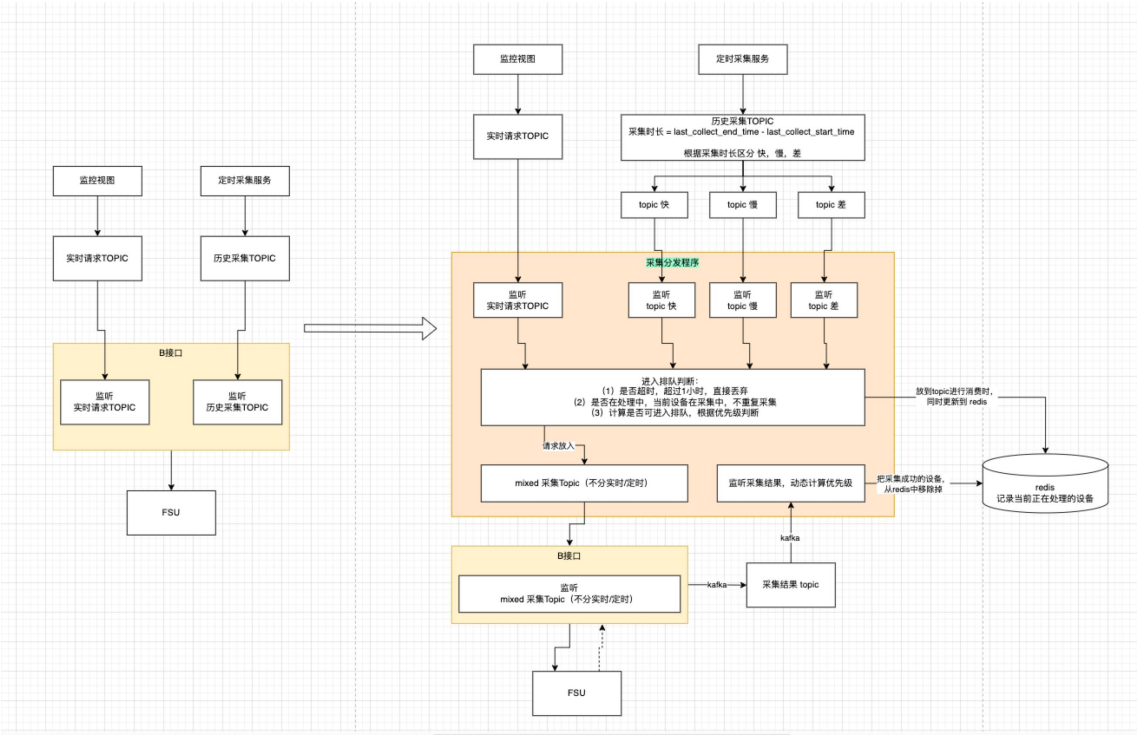
比对项	实时采集	定时采集
topic	spider_cinterface_getPointDataRequest%s	spider_cinterface_hisCollectDataRequest%s
限速	不限	令牌桶限速

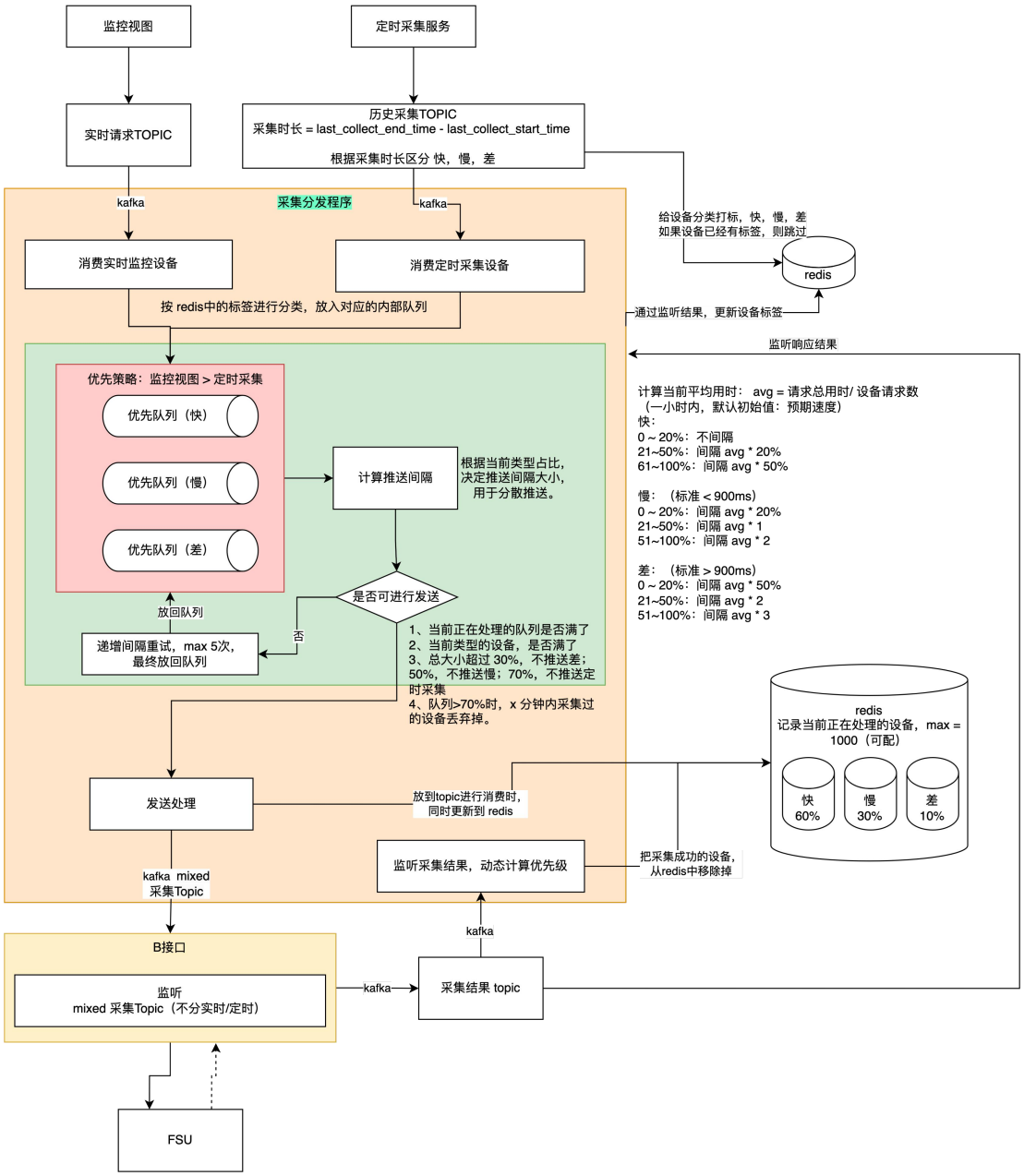
2.1.2 本次设计考虑：

针对等待采集的设备根据优先级算法进行排队下发。底层 B/C 接口处理逻辑不变，中间添加

“采集分发程序”完成该部分逻辑：

调整前后比对概览（B 接口为例，C 接口同理）





2.1.3 优化方向说明：

按采集场景分为：实时采集，定时采集两种

按设备采集速度分为：快，慢，差 三种

（1）按优先级排队：按设备维度动态计算优先级，处理场景越优先的设备越优先

推送到 Kafka 队列中，采集速度越快的设备越优先推送到 kafka 队列中

（2）使用聚合队列作为 B/C 接口新的监听 topic，只区分接入类型（B 接口接入，C 接口接入），不再区分场景，进入到此队列后，就按该队列进行队列处理（实时采集或定时采集）

（3）重复下发限制：

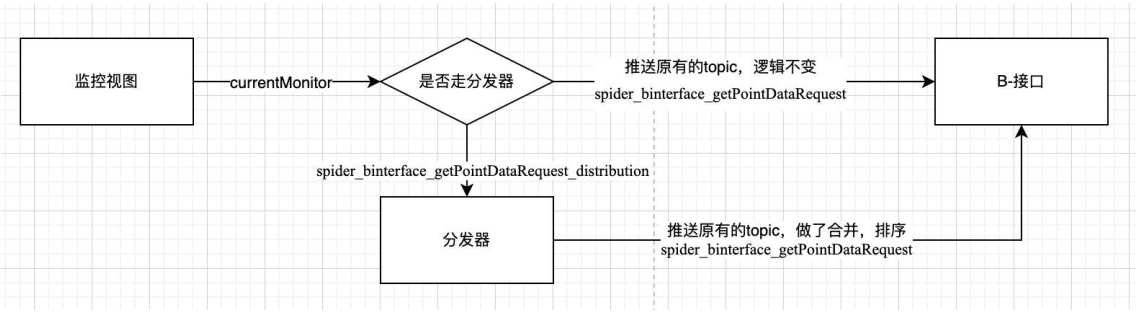
通过记录下发与响应的报文，得到“处理中”状态的设备

采集中的设备，避免重复下发（通过请求下发和请求响应，可以得到“采集中的设备”）

采集时间有效期内的设备，避免重复下发（比如设备 A 采集响应时间未超过 5 秒，又被发起了采集，则不重复发起）

（4）超时任务丢弃：排队超过一小时的设备，进行丢弃处理，避免 kafka 队列积压，造成 rebalance

调整



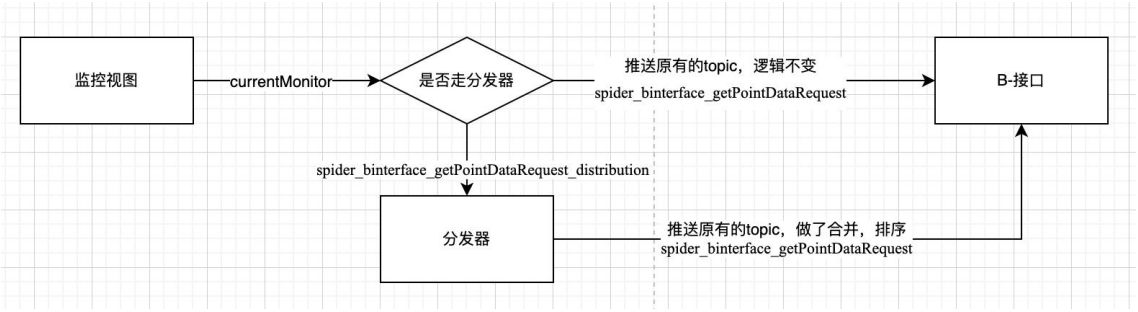
Topic 说明

2.1.4 B 接口-实时请求

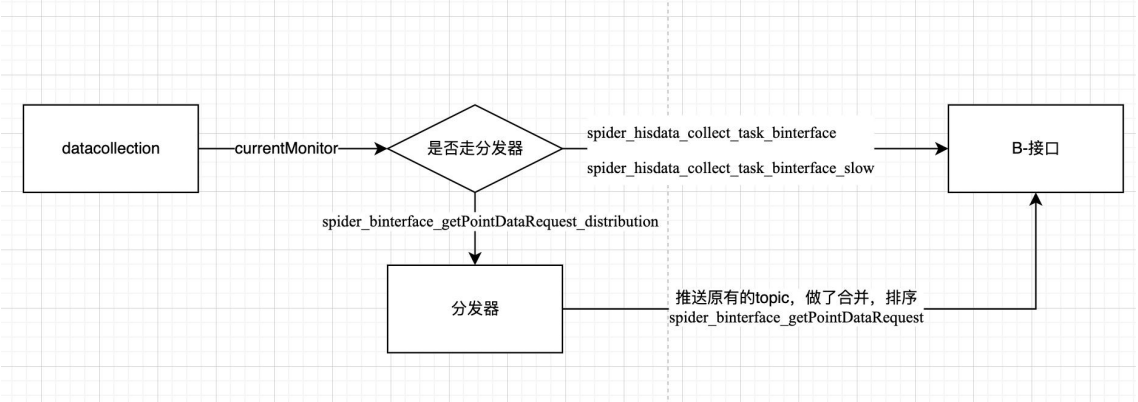
实时请求 TOPIC: spider_binterface_getPointDataRequest （currentmonitor)

分发器监听 topic: spider_binterface_getPointDataRequest_distribution （新增，分发器）

采集结果响应 topic: spider_binterface_handlePointDataResponse (不变）



2.1.5 B 接口-定时采集



定时采集请求 TOPIC:

spider_hisdata_collect_task_binterface 、 spider_hisdata_collect_task_binterface_slow（datacollection 生产，B 接口消费）

分发器监听 topic: spider_binterface_getPointDataRequest_distribution （新增，分发器）

采集结果响应 topic: spider_binterface_handlePointDataResponse (不变）

2.1.6 设备采集耗时统计方式

从 distribute-service 将请求推送到 kafka 开始计算，

将请求批次记录到 redis

当 distribute-service 从 kafka 监听到对应批次的响应数据，则表示此批准数据完成，

更新设备的耗时，更新到 redis，供下次设备采集分类做参考。

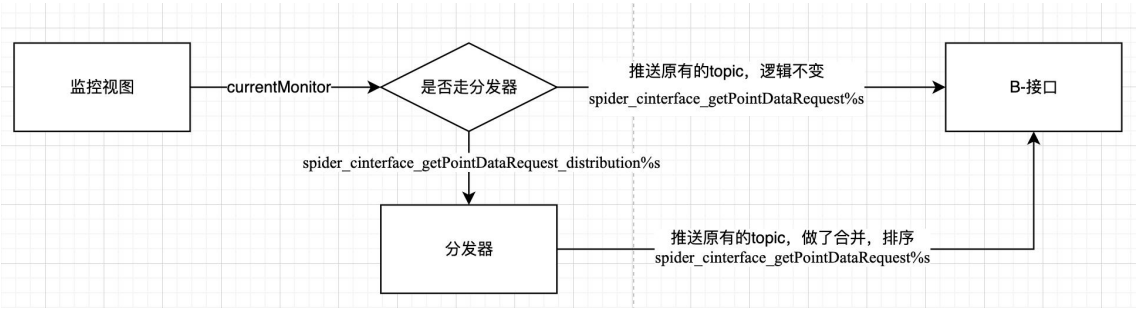
redis-key: device_data_collect:cost

key: ${device_id}

value: {last_collect_start_time:xx, last_collect_end_time: xx, cost: xx, tag: fast/slow/poor}

key: md5(device_id + List<MeteCode>)

2.1.7 C 接口-实时请求

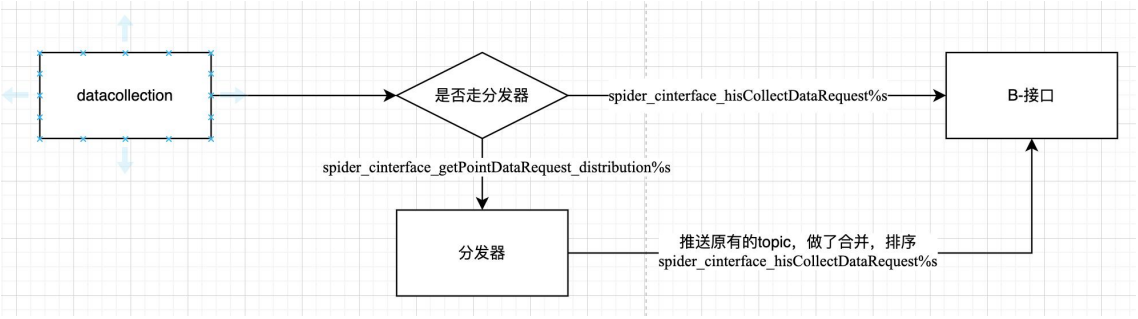


实时请求 TOPIC: spider_cinterface_getPointDataRequest% （currentmonitor)

分发器监听 topic: spider_cinterface_getPointDataRequest_distribution%s （新增，分发器）

采集结果响应 topic: spider_binterface_handlePointDataResponse (不变）

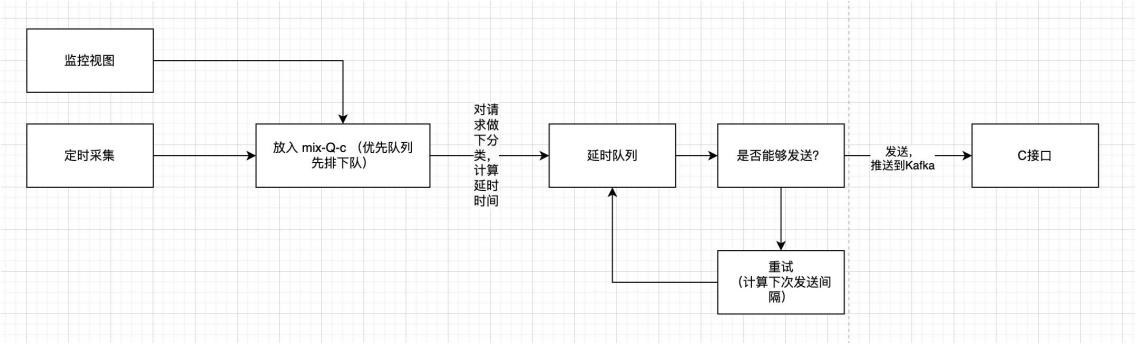
2.1.8 C 接口-定时采集



实时请求 TOPIC: spider_cinterface_hisCollectDataRequest%s （datacollection)

分发器监听 topic: spider_cinterface_hisCollectDataRequest_distribution%s （新增，分发器）

采集结果响应 topic: spider_cinterface_handlePointDataResponse (不变）



优先级计算：

实时与定时的优先级：2:1

快,慢,差排队优先级： 6:2:1

定时采集的设备会有 快，慢，差

监控视图里面来的设备也会有 快，慢，差，所以计算好优先级后，不再进行区分场景（实时查询或定时采集）

实时类

实时快：优先级 = 实时优先级 × 快优先级 = 2 × 6 = 12

实时慢：优先级 = 实时优先级 × 慢优先级 = 2 × 2 = 4

实时差：优先级 = 实时优先级 × 差优先级 = 2 × 1 = 2

定时类

定时快：优先级 = 定时优先级 × 快优先级 = 1 × 6 = 6

定时慢：优先级 = 定时优先级 × 慢优先级 = 1 × 2 = 2

定时差：优先级 = 定时优先级 × 差优先级 = 1 × 1 = 1

总优先级 = 12 + 4 + 2 + 6 + 2 + 1 = 27

即同一时间内，在等待处理的设备类型，不能超过以下占比

实时快占比 = 12/27 ≈ 44.4%

实时慢占比 = 4/27 ≈ 14.8%

实时差占比 = 2/27 ≈ 7.4%

定时快占比 = 6/27 ≈ 22.2%

定时慢占比 = 2/27 ≈ 7.4%

定时差占比 = 1/27 ≈ 3.7%

下一个设备，可以是当前实际占比比目标占比差距最大的设备，然后排上去。

即谁点的比例最低，谁就往上排。

比如 当前 实时快当前点比 40%，定时慢点比 0%， 那么，就应该先把一个定时慢的设备排上去。

1、初始化：

2、策略调整，当队列里面达到 500，600，700，1000 不再接收差，慢，中的设备（这个必须要有，与占比比例结合，两种策略并存）

3、快，慢，差是按项目时间需求来计算的（比如 5 分钟要完成 2000 个，那这个平均时间就是快的标准。）

4、队列满了，尝试 5 次（100，200，400，800...）如果都失败，放回原队列

5、丢弃策略：当>700，5 分钟内采集过的设备，丢弃。

6、请求批次合并（x 秒内的请求），排序，再放 kafka（一个批次 2800，处理了 700 个，还有 2100 在内存里面等着，下一个批次过来的时候，和原有的 2100 个，合并去重，再排队，再往 kafka 里面放）

问题考虑：

1、定时请求可能是一大批一下子过来，监控视图是一个一个过来比较分散的。所以很可能是因为定时任务扎堆，把监控视图的给堵住，这个要怎样处理。实时请求可能一下子来了 500 个设备的采集，那这 500 个很可能就下发下去了，监控视图这个时候再来一个，如果底端是顺序处理的话，上层无论怎样控制，这个监控视图的设备都得等前面 500 个处理完之后才到它。这是底层逻辑导致上层无法调解的。上层最多只能把这个，“处理中”的量调小，让在处理中的量比较小的时候，才能保证监控视图不用等太久。

简而言之，如果底层性能很差，这个是不可调和的问题。

这个分发程序只是结合不同 LSC 平台的实际情况，调整下发比例，优先级，丢弃策略，重试这几个方面来优化，达到尽量不要让监控视图因为性能数据采集影响到，采集快的设备不要受到采集慢的设备太多的影响。

2.1.9 C 接口代码调整

1、监控视图请求

（1）给 currentmonitor 添加配置： sendToDistribution: false，可以选择直接把请求推送给 B 接口 / C 接口，也可以选择推送给 distribution。先默认关掉，待测试有效稳定，再切换成默认开。

（2）在 currentmonitor 给合开关，进行判断，如果开关打开，就要把消息推送给 distribution 的 topic：spider_binterface_getPointDataRequest_distribution，要给对象添加 source 属性。

（3）如果开关不打开，走原来的逻辑，不用做任何的调整。

（4）如果开关打开，请求推送给分发器，由分发器进行合并处理，再推送到原来的 topic: spider_cinterface_getPointDataRequest%s，最终 C 接口都是原来的逻辑。

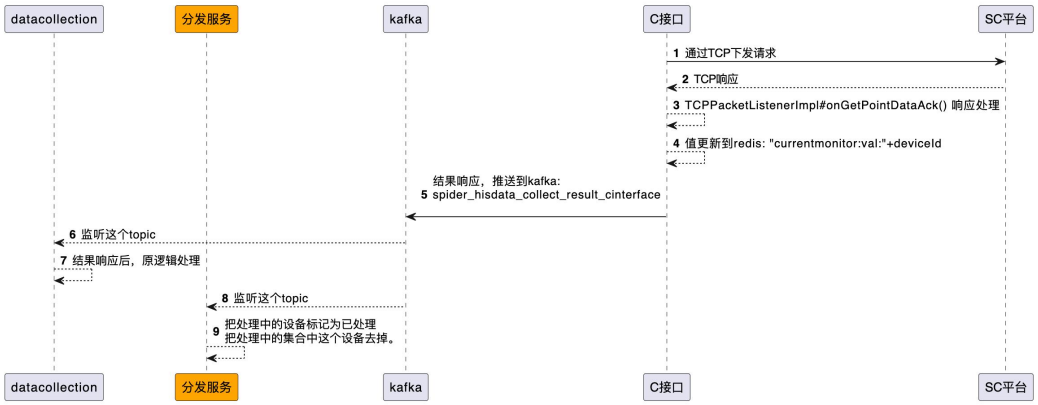
（5）distribution 的 nacos 配置：application-distribution.yml

① kafka.topic.cinterface_getPointDataRequest:
spider_cinterface_getPointDataRequest%s （与 C 接口一致）

② kafka.topic.binterface_getPointDataRequest:
spider_binterface_getPointDataRequest（与 B 接口一致）

请求响应处理

C 接口请求响应处理



2.2 逻辑结构

[使用框线图或者 UML 描述本系统或子系统的逻辑结构，包括模块的划分与各模块间的依赖关系。]

程序设计规范性：是否按照公司标准模板进行设计（包括**对象、时序、包、类、方法**等），是否有遗漏，同时满足公司软件设计规范要求

2.3 模块定义

[描述模块的划分及各模块的总体功能]

模块的外部接口和入参/出参，基于接口的设计，就先不强制在设计文档中体现，但是需要在代码中把外部接口、外部接口涉及的 view 对象和内部的分离，并且添加注释

模块名称	模块描述*[主要描述模块的功能及其意义]*	模块状态*[重用/变更/新增]*	模块所属[集团/陕西/云南/GEMC]
distribute	性能数据采集分发服务	新增	GEMC
b-interface	B 接口	变更	GEMC
c-interface	C 接口	重用	GEMC

2.4 模块间交互图

[列出本系统或子系统所有模块的交互（SEQUENCE）图，并进行必要的文字描述，需要反映出对需求关键点的设计实现分解，如果涉及到多个关键流程，那么需要多个时序图]

2.5 影响范围评估

程序设计是否完全覆盖了版本需求内容，改造开发点是否有考虑对其他程序的影响并给出合理的设计

特别是已有功能优化，老功能改造该章节为必填。

阈值下发逻辑直接复用现有的逻辑，页面新增实时下发接口；对数据有效性稽核原有报表新增/改造定义的字段不复用原有的数据以及数据运算逻辑；在 B 接口添加额外的写入参数巡检表校验逻辑，该逻辑添加开关，默认设置为关；

2.6 公共设计

2.6.1 关键技术考虑

难点、重点技术是否有考虑、是否有研究、是否有针对该技术业务的设计、是否可行能落地、是否提供实现方式

2.6.2 公共机制考虑

是否有考虑公用抽象工具类、公共类/组件，是否存在为了避免麻烦将可公共的地方写在了私有包或者项目工程中

动环项目新需求开发，**禁止复制 V 版本中的代码到项目中**，组件及工具类不满足的应进行扩展

2.6.3 重用性考虑

系统重用、组件复用考虑，是否已考虑、进行系统、组件重用、复用设计

动环所有新项目都必须复用能源平台 V 版本，新需求开发都需要考虑与 V 版本的关系，产品化要求及组件化可行性

2.7 数据流图

[列出本系统或子系统所有模块的数据流图，并进行必要的文字描述。]

2.8 数据存储设计

2.8.1 逻辑模型

[这里要用框线图按照业务涉及的业务实体（非物理模型的表），列出不同业务实体的关系（几对几），业务实体要包括主要的业务属性]

2.8.2 物理模型列表

[数据库设计采用数据库设计模板，并编写成单独文档。在这里请说明引用的数据库设计文档。

此节从本系统或子系统的角度来列举与本高层设计有关的数据库对象变更等。]

检查点:数据结构规范性考虑

2.9 非功能设计

2.9.1 指标

按照当前系统的规模和容量，估算并承诺一个季度内的设计目标

非功能指标分类	非功能指标项	指标值	补充说明
安全	是否涉及外部接口		*是否涉及外部接口*
	是否包含身份认证/报文防篡改		*外部接口必须有 appID/appKey 身份认证，外部接口必须有报文的散列值防篡改*
	是否包含铭感数据		*外部接口涉及敏感数据必须用 https 协议*
性能	接口 TPS(每秒)		
	接口响应时间(毫秒)		
	风险规模考虑	*是否需要水平/垂直拆涉及存储的表，是否需要换存储*	
	高性能高并发大数据量读写场景	**检查点：如不涉及填不涉及**	*是否有识别高性能高并发大数据量读写场景，并且对该场景进行专项设计，并且就设计技术合理性、工作量、设计内容均具备可*

			落地性
	高可用场景考虑	**检查点：如不涉及填不涉及**	*服务是否具备高可用、弹性扩展能力，是否更多地以集群热备为主而非冷备，同时在服务在高可用环境下是否存在多程序运行冲突*

2.9.2 第三方引用

[描述各模块所引用的第 3 方库、工具等，包括库、工具的名称、开发者/商、版本等信息，并描述其引用方式、接口等。]

2.9.3 网络资源需求梳理

网络资源规划是否涵盖，是否合理，涉及的对外网络开通策略需要给出，并且需要满足正确性、可行性、完整性

【检查点】如不涉及对接接口，且无需额外网络策略写不涉及；如需开通网络策略，参考下表补充网络策略信息

源地址	源端口	源地址所属业务系统	目的 IP 地址	目的端口	目的地址所属业务系统	协议（TCP，UDP）	动作（开通，关闭）	开通策略的用途
10.242.135.0/24（原有） 10.242.238.0/24（新增）	ALL	网络线 4A	188.0.211.40	8080	卓望动环工作台	TCP	开通	4A 图形堡垒机访问资源

2.9.4 安全性

2.9.4.1 公司安全开发规范满足度

是否满足公司安全开发规范要求

原则上必须满足公司安全开发规范要求，如因可观因素暂不满足或需延迟满足的，说明不满足的点、不满足原因、可预期风险及应对措施、后续满足计划

2.9.4.2 垂直越权考虑及实现

垂直越权基于 webbas 架构实现

对于新的资源/接口，要保证完成以下动作：

（1）维护资源管理 xml（前端进行维度，文档发布于 SVN），xml 文件上管理菜单，按钮，资源权限（即接口）

（2）开发人员（后端），将 xml 导入，更新 wb_resources 表，

（3）只有进行了配置的接口才能被调用，否则会出现 403 的错误提示。

2.9.4.3 水平越权考虑及实现

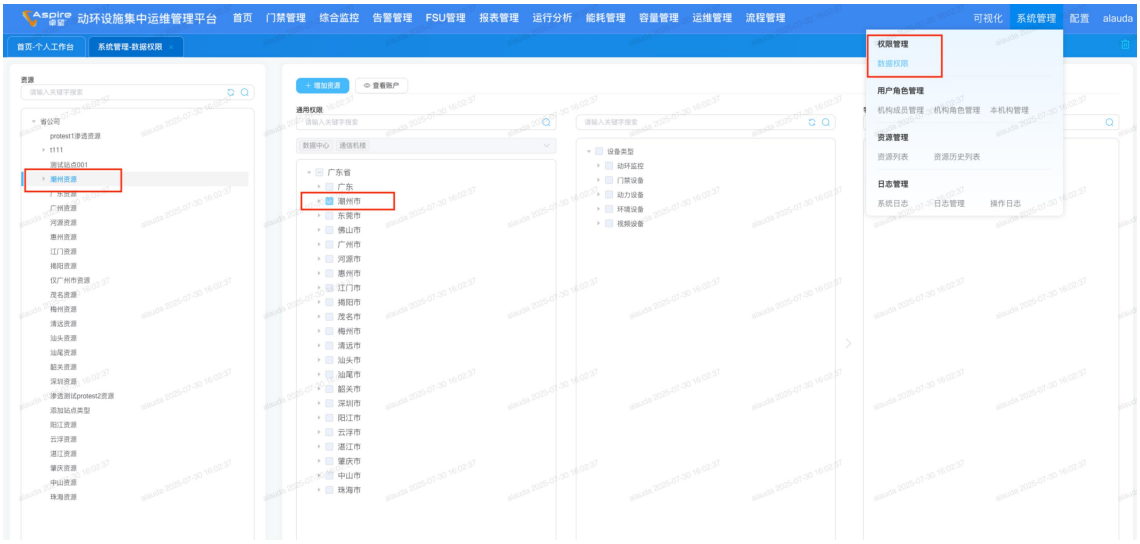
通过账号拥有的角色区域权限，进行区域维度的限制。

1、数据权限配置：

（1）新增数据权限（如果沿用已有数据权限，跳过该步骤）

配置入口：系统配置-权限管理-数据权限，

比如下图，配置了一个“潮州资源”，可以访问的数据范围是“潮州市及以下”的所有数据



（2）给账号绑定数据权限

入口：系统管理-用户角色管理-本机构管理，勾选账号，给成员分配数据权限

2、新增接口添加数据权限

2.9.4.4 敏感数据加密考虑及实现

敏感数据加密考虑及实现是否完成，如不涉及的写不涉及

2.9.4.5 重放攻击考虑及实现

重放攻击考虑及实现是否完成

2.9.4.6 应用、中间件配置及认证安全

应用、中间件配置及认证安全是否完成实现

2.9.4.7 文件上传下载安全

文件上传下载安全是否完成实现

当前动

2.9.4.8 未经授权访问 API 防护

未经授权访问 API 防护是否完成实现

2.9.5 扩展性

【检查点】程序是否具备未来功能、非功能场景的扩展能力，是否在程序中存在硬编码配置，是否在功能改造时需要“大动干戈”或“无从下手”

2.9.6 风险

3 系统可部署性设计（可选）

［说明为了简化部署环节，缩短部署时间而在程序内部设计中作出的安排。］

4 系统可维护性设计

［说明为了系统维护的方便而在程序内部设计中作出的安排，包括在程序中专门安排用于系统的检查与维护的检测点、日志、屏幕输出、专用模块、工具等。］