## 项目功能开发情况汇报

### 1. 异常处理与断点续传（Error handling）

* **完成情况**：已完成
* **具体方案**：
  + Kafka本身默认提供基于offset的断点续传能力，可自动记录消费进度，确保异常后可无缝恢复消费。
  + 本次特别针对消费过程中可能存在的长耗时任务，优化了Kafka的关键参数，如将max.poll.interval.ms从默认的300000毫秒调整到600000毫秒，有效避免了因消费耗时过长而触发Kafka Rebalance的问题，显著提升了异常场景下系统的稳定性。

### 2. 性能优化（Performance）

* **完成情况**：已完成
* **具体方案**：
  + 新增了Kafka的自动提交间隔配置（auto.commit.interval.ms），优化至每秒一次自动异步提交，大幅降低了消息重复消费的风险。
  + 数据拉取后由单线程改为多线程并行批量处理（基于自定义线程池CustomThreadPool），整体处理性能显著提升。
  + 当前系统实测性能达到5分钟内处理百万条数据以上，系统整体吞吐能力显著提高。

### 3. 去除HBase依赖（Remove dependence on HBase）

* **完成情况**：已完成
* **具体方案**：
  + 已成功去除系统中对HBase的依赖，所有数据访问与缓存操作现已完全迁移至Redis和MongoDB。

### 4. 单元测试覆盖率（UT Sonar & Code Coverage）

* **完成情况**：进行中（当前覆盖率30%+）
* **下一步计划**：
  + 目标明确为提升单元测试覆盖率至80%以上，持续进行单元测试的编写与SonarQube代码质量检查，确保代码的高质量和可维护性。

### 5. 系统监控（Monitor - Grafana & Prometheus）

* **完成情况**：暂未开始
* **下一步计划**：
  + 明确将在下一阶段使用Prometheus进行系统指标采集，利用Grafana进行监控可视化呈现，以便快速定位并解决线上问题。

### 6. 关联Cusip数据的合并处理（Combine Cusip for related Cusip）

* **完成情况**：已完成
* **具体方案**：
  + 将related Cusip关系数据缓存于Redis，相关的Cusip refData数据存储于MongoDB，通过高效整合Redis和MongoDB数据实现实时数据合并与推送。

### 7. 实时订阅模式与扩展性（Live-live mode & Extensible）

* **完成情况**：部分完成，尚需完善
* **具体方案与后续规划**：
  + 当前尚未实现对MongoDB配置变化的实时监听与动态加载能力，后续明确计划实现此功能，使系统支持实时动态添加或变更规则（Rule）。
  + 后续还将增加API能力，以支持动态启停指定的Kafka监听器，进一步提升系统的灵活性。
  + 系统整体架构已支持通过扩展Pod实现水平扩容，具备高度的扩展性。

### 8. 系统恢复时间（Recovery time）

* **完成情况**：已完成
* **具体方案**：
  + 通过优化Kafka客户端的重试与重连策略（如将reconnect.backoff.max.ms优化为10秒），显著提升了异常情况下的快速恢复能力。

## 项目Review完成情况汇总表：

| **Review项目** | **完成情况** | **备注与下一步动作** |
| --- | --- | --- |
| 1. 异常处理与断点续传 | ✅ 已完成 | Kafka参数优化，有效降低了消息重复消费风险 |
| 2. 性能优化 | ✅ 已完成 | 多线程批量处理，5分钟处理百万级数据 |
| 3. 去除HBase依赖 | ✅ 已完成 | 已迁移至Redis和MongoDB |
| 4. 单元测试覆盖率 | ⚠️ 进行中 (30%+) | 目标80%以上，持续推进 |
| 5. 系统监控（Grafana & Prometheus） | ❌ 尚未开始 | 下一阶段明确使用Prometheus与Grafana |
| 6. 关联Cusip数据合并处理 | ✅ 已完成 | Redis缓存与MongoDB数据高效合并 |
| 7. 实时订阅模式与扩展性 | ⚠️ 部分完成 | 后续计划支持动态rule及API启停监听器，Pod水平扩容已支持 |
| 8. 系统恢复时间 | ✅ 已完成 | Kafka重试和重连策略已优化，系统恢复迅速 |