


Flink 实战宝典

Flink 理论与实战

作者：左元

组织：尚硅谷

时间：April 12, 2020

版本：0.1



完成比完美更重要！

目录

| | |
|-----------------------------|-----------|
| 第一部分 Flink 的使用 | 1 |
| 1 第一章，流式处理理论概述 | 2 |
| 2 第二章，Flink 框架快速上手 | 3 |
| 3 第三章，Flink DataStream API | 4 |
| 4 第四章，基于时间和窗口的算子 | 5 |
| 5 第五章，Flink 状态编程 | 6 |
| 6 第六章，Flink DataSet API | 7 |
| 7 第七章，Flink 与外部系统的交互 | 8 |
| 8 第八章，Flink Table API & SQL | 9 |
| 9 第九章，Flink CEP 库的使用 | 10 |
| 第二部分 Flink 的部署与运维 | 11 |
| 10 第十章，Flink 应用的监控 | 12 |
| 11 第十一章，如何部署 Flink 集群 | 13 |
| 12 第十二章，Flink 集群的高可用 | 14 |
| 第三部分 Flink 内核与优化 | 15 |
| 13 第十三章，Flink 运行时架构 | 16 |
| 14 第十四章，Flink 状态的原理 | 17 |
| 15 第十五章，Flink 的容错机制 | 18 |
| 16 第十六章，Flink 作业的调度 | 19 |
| 17 第十七章，Flink 内存管理的特点 | 20 |

| | |
|---------------------------------|---------------|
| 18 第十八章, Flink 的网络 IO 机制 | 21 |
| 19 第十九章, Flink 常见优化措施 | 22 |
| 19.1 在项目启动的时候 | 22 |
| 19.2 在分析需求的时候 | 22 |
| 19.3 在开发的时候 | 23 |
| 19.4 维护 | 23 |
| 20 第二十章, 流的去重及其优化 | 24 |
| 21 第二十一章, 如何解决数据倾斜 | 25 |
| 第四部分 Flink 实时数仓项目 | 26 |

第一部分

Flink 的使用

第一章 第一章，流式处理理论概述

第二章 第二章, Flink 框架快速上手

第三章 第三章, Flink DataStream API

第四章 第四章，基于时间和窗口的算子

第五章 第五章, Flink 状态编程

第六章 第六章, Flink DataSet API

第七章 第七章, Flink 与外部系统的交互

第八章 第八章, Flink Table API & SQL

第九章 第九章, Flink CEP 库的使用

第二部分

Flink 的部署与运维

第十章 第十章, Flink 应用的监控

第十一章 第十一章，如何部署 Flink 集群

第十二章 第十二章，Flink 集群的高可用

第三部分

Flink 内核与优化

第十三章 第十三章，Flink 运行时架构

第十四章 第十四章，Flink 状态的原理

第 十五 章 第十五章，Flink 的容错机制

第十六章 第十六章，Flink 作业的调度

第十七章 第十七章, Flink 内存管理的特点

第十八章 第十八章, Flink 的网络 IO 机制

第十九章 第十九章，Flink 常见优化措施

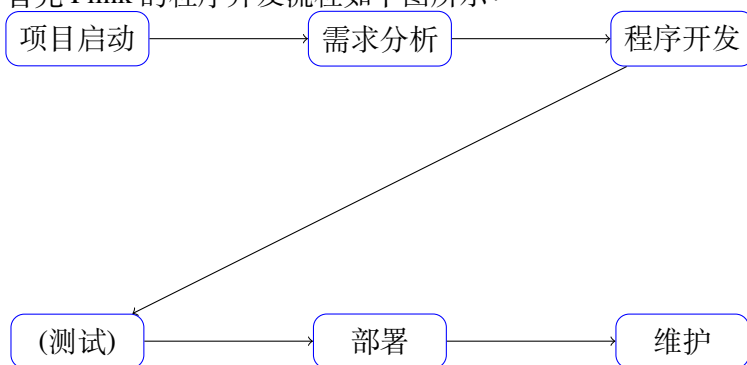
过早优化是万恶之源。

高德纳

任何程序的编写其实遵循三个阶段。第一，编写出一个可以运行的程序。第二，确保这个程序是正确的。第三，优化这个程序。所以高德纳才说：“过早优化是万恶之源”。而 Flink 程序与其他的程序的优化策略的基本道理是一样的，也就是尽量减少计算密集型和 IO 密集型这两种运算。

我们先来看一下在开发 Flink 程序时常犯的一些错误：

首先 Flink 的程序开发流程如下图所示：



19.1 在项目启动的时候

- 不使用一种迭代式的开发，而是想要毕其功于一役。
- 刚学了一些 Flink 基础知识，就想用 Flink 来解决公司最难的需求。
- 事先没有流处理的知识，直接开始使用 Flink 进行编程。
- 事先没有经过 Flink 编程方面的训练，例如：基本 API 的熟悉，基本的调优训练等等。
- 不和社区进行交流，不在社区中搜索可能已经存在的答案。例如不使用 StackOverflow、Flink Documentation、Flink Jira 等常见工具。

19.2 在分析需求的时候

- 不考虑程序的一致性，例如是否允许丢失数据，是否允许数据重复计算。
- 不考虑程序后续的迭代和程序
- 不考虑需要解决的问题的规模，例如数据量的大小，数据流高峰期的数据量等等。
- 没有细致的分析真实的业务需求

19.3 在开发的时候

- 没有细致思考到底使应该使用 `DataStream API` 还是 `Table API & SQL`?
- 对 `Flink` 的类型系统理解不深，盲目使用了嵌套过深或者非常复杂的数据结构（例如深度嵌套的 `POJO CLASS` 或者 `case class`）
- 胡乱使用 `keyBy()` 函数
- 在不同的任务之间共享静态变量，从而造成了死锁、竞争等同步问题。
- 在 `UDF` 函数中随意开启新的线程，会造成非常难以调试的 `bug`，例如检查点相关的问题。
- 随便自定义窗口函数，而没有使用 `Flink` 默认的函数。
- 没有将初始化的代码放在富函数的 `open` 方法中，而是放在了例如 `flatMap`、`map`、`filter` 函数中。

19.4 维护

由于 `Flink` 更新非常频繁，所以不要随便对程序进行升级。

第 二 十 章 第 二 十 章，流的去重及其优化

第二十一章 第二十一章，如何解决数据倾斜

第四部分

Flink 实时数仓项目