Flink StateBackend (1) - Overview

> 如果要说 Flink 作为一个流式计算引擎，有什么很不错的地方，我觉得 State / StateBackend 算是一个。

Overview

通常，从我的观点来看，如果一个无状态分布式程序跑在 Flink 上，我会简单地称它只是单纯地利用了 Flink 的分布式特性而已，并没有真正地在利用 Flink 这个框架。比如简单的无须容错的数据导入导出服务。

在 Overview 中我将简单地介绍现有 Flink 中 StateBackend 的种类，和 StateBackend 的组成部分。之后由浅入深全方位讲解各个 StateBackend 的背后原理。

组成部分

通常一个 StateBackend 由以下两者组成：

- OperatorStateBackend
- KeyedStateBackend

其中，OperatorStateBackend 主要用于无 Key 场景下的状态存储，如 Kafka Offset，常见的有 ListState / UnionState / BroadcastState。KeyedStateBackend 主要用于 Key 场景下的状态存储，如某个 uid 用户在 10min 窗口下的行为明细，常见的有 ValueState / ListState / MapState 等。

不管是 OperatorStateBackend 和 KeyedStateBackend，本质作用都是存储状态，所以他们都有如下功能：

- 恢复状态
- 修改状态
- 持久化状态

之后逐一讲解 StateBackend 时也会从这几个方面去讲解。

分类

一般情况下，KeyedStateBackend 所使用的频次以及状态的大小都超过 OperatorStateBackend，所以大家对 KeyedStateBackend 的关注度往往比较大。

通常我们说的 StateBackend 有三种：

- MemoryStateBackend
- FsStateBackend
- RocksDBStateBackend

而这三种 StateBackend 对应的 OperatorStateBackend 和 KeyedStateBackend 分别为：

StateBackend	OperatorStateBackend	KeyedStateBackend
MemoryStateBackend	DefaultOperatorStateBackend	HeapKeyedStateBackend
FsStateBackend	DefaultOperatorStateBackend	HeapKeyedStateBackend
RocksDBStateBackend	DefaultOperatorStateBackend	RocksDBKeyedStateBackend

看到这里可能要问，MemoryStateBackend 和 FsStateBackend 有什么区别？这个具体后面文章会讲，因为 KeyedStateBackend 本身只定义 State 的存储方式和序列化方式，至于状态存储到哪里，如何去存储，Checkpoint 怎么操作，还是由 StateBackend 来控制的。所以 MemoryStateBackend 和 FsStateBackend 都是将状态放入内存中但是在 Checkpoint / Savepoint 的方式会有所不同。

在这里简单描述一下三种 StateBackend 的用途，和官方博客描述的一致，有兴趣可以看 Ververica Blog。

> MemoryStateBackend

MemoryStateBackend 主要用来做本地调试，可以用于存储小状态，如 KB 级别的状态数据。

> FsStateBackend

FsStateBackend 可以用于生产环境，具有 HA 特性。

> RocksDBStateBackend

RocksDBStateBackend 适用于超大型 State，也是诸多应用场景下最主流的选择，具有 HA 特性，支持增量 Checkpoint。