**Flin源码结构**[](" \l "flin" \o "Permanent link)

Flink源码还是比较清晰易懂的，尤其是了解她的架构后，大部分实现都非常符合common sense。下面主要过下Flink所有的包，解释下他们的主要作用，以及需要框架里使用的主要类吧。

| **Artifect** | **功能介绍** |
| --- | --- |
| Flink-clients | FlinkCommandLine入口类(CliFrontEnd) ,解析Flink命令行  （DefaultCLI, YarnSessionCLI）,负责将从用户jar文件中main函数生成plan(PackagedProgramUtils),通过LocalExecutor或RemoteExecutor调用Flink-Optimizer生成JobGroph. 使用ClusterDescriptor启动JobManager（比如在yarn），使用并将jobGraph用户ClusterClient提交给本地或远程的JM |
| Flink-runtime | Flink 源碼的核心。 框架的核心組件都在裏面， 包括 Diskpatcher, JobMaster, TaskExecutor, ResourceManager, ClusterEntryPoint, WebUI以及他們依賴的子組件以及核心數據模型，JobGroup, ExecutionGraph, Execution, Task, Invokable, Operator, Driver, Function, NetworkBufferPool, ChannelManager, IOMemory, MemoryManager, PRCService, HAService, HeartbeatService , CheckPointCoordinator , BackPressure, etc . 這些類的名字和作用在上一章都或多或少提到過， 最好結合架構的介紹， 理解他們的代碼。 |
| Flink-runtime-web | Flink Web monitor 的界面和handler . handler會調用Dispacher 的方法處理客戶請求， 比如sumitJob. |
| Flink-java | 用Java 和 scala實現的Flink Batch API , Dataset, ExecutionEnrionment, etc . |
| Flink-streaming-java | 用Java 和 scala實現的Flink stream API , DataStream, ExecutionEnrionment, , StreamExecutionEnvironment, windowing，和對streaming提供支持的runtime, Checkpoint, StreamTask, StreamPartitioner , BarrierTracker, WindowOperator |
| Flink-optimizer | 主要功能是优化plan和生成JobGraph。Flink-optimizer是一个Client端很重要的库，它决定了从客户端代码application到jobGraph形成过程。我个人绝对的也是调查和解决application问题的根源，比如application的写法是不合适滴，有问题的，或最优的等等。在前面架构里面，并没有谈及Flink-optimizer，所以这里简单介绍下。Flink application开发者使用Flink API编写application，flink-client为了将application最终能在Flink cluster运行起来，它首先通过用户jar文件中的main函数生成plan：以及DataSink为根的一个或多个树结构，树的节点都是application使用API operator，每一个节点的输入来自于树的下一层节点。不难想象树的叶子节点都是DataSource：他们没有输入节点，当有用于读取数据源的inputFormat。跟节点是DataSink：他们既有输入节点，也有用于写入目标系统的outputFormat，中间节点都会有一个多个的输入节点。Plan数据结构描述了同用户的application完全相同的数据流的节点，但它只是一个逻辑树状结构，并没有对连接点的边做描述，也没有对application编写的数据流做任何修改。然后使用Flink-optimizer将plan数据根据优化测绿设置节点的边上的数据传输方式，并同时使用OptimizerNode生成多种优化方案，最后使用cost最小的方案产生的方案作为OptimizedPlan。比如将数据装载方式（ShipStrategyType）设置FORWARD，而不是PARTITION\_HASH而FORWARD的网络cost最低(0)。OptimizedPlan是一个图结构，图中的顶点(PlanNode)记录了自身的cost以及从source开始到它的累积cost。OptimizedPlan主要针对与join和interation操作。然后Flink-optimizer将OptimizedPlan编译成jobGraph。编译过程应该基本上是一对一的翻译（从PlanNode到JobVertex），但是一串PlanNode满足Chaining条件（比如数据在每个opertot都不需要重新分区，流过operator之后，直接forward到下一个operator），Flink-optimizer就像这些operator连接到一块，然后在JobGraph里面只创建一个ChainedOperator jobVertex，ChainedOperator同Spark里面的stage概念类似，是优化的一部分。最后flink-client将jobGraph提交给FlinkCluster，jobGraph变形为ExecutionGraph在JM和TM上执行。可以从Optimizer的compile和JobGraphGenerator的CompileJobGraph展开看，他们分别compile的是Plan和OptimizedPlan。Flink-optimizer优化的是partition的选择以及算法的选择，而不是DAG的workflow |
| Flink-table | 用户可以用flink-java、flink-stream-java里的api编写flink application，也可以用Flink-Table的table api和flink-sql写application。Flink-Table将数据源（Dataset, DataStream）都generalize 成Table（row based）,用户可以用类似关系型数据库的操作方式操作Flink的数据源，这种方式虽然屏蔽了一些flink-api的特性（比如 broadcast），但极大的降低了application的开发难度，减少了客户程序的代码量，从而极大的提高系统的重用度。Flink-table的底层还是依赖dataset、datastream api, 所以基于table api或SQL的程序（Program）最终会翻译成Flink-optimizer的Plan，经过前面所述同样的编译优化过程，最终一个EG的形式运行在JM和TM上。当然，在被翻译成Plan之前，Flink-table的Program也会有自身的优化过程，比如SQL Plan optimization。代码需要看，如何实现一个Table：StreamTableSource, BatchTableSource,BatchTableSink,AppendStreamTableSink，如何扩充FlinkSQL：UserDefinedFunction，ScalarFunction,TableFunction,AggregateFunction TableEnvironment |
| Flink-yarn Flink-container | Flink 怎么支持异构环境的， 包括不同环境里的, 主要是异构环境里的不同ResourceManager （启动TM ）,以及 ClusterDescriptor (启动JM)如何实现的。 |
| flink-connector flink-format | 连接外围数据系统（数据源和输出)系统的InputFormat, OutputFormat . |
| flink-filesystem | Flink支持的分布式文件系统， hadoop, s3, mapr |
| flink-metrics | flink 的metrics 系统 |
| flink-statebackends | flink 的 state的存储 |
| flink-test flink-jepsen | Flink的UT和集成测试。 |
|  |  |