**Pipeline和splitjoin结构**

为了便于利用composite构建流图，在composite内部引入支持流水线和并行结构化层次性的编程结构，即pipeline和splitjion两种结构。图1描述了这两种基本的流结构。pipeline结构将不同的operator顺序直线连接；splitjion结构对输入流进行分批并行处理。splitjion为了便于利用这两种结构在composite中与其他的operator混合使用，**要求这两种层次性结构都是单输入流单输出流的**。

 

图1： COStream支持的两种层次性结构

**pipeline结构**

pipeline的结构主要由pipeline的头部和pipeline的主体组成，BNF表示如：

*pipelineHead ::= ID‘=’ ‘pipeline’ opInput*

*| ‘pipeline’*

*pipelineBody ::=’{’*

*declaration\**

*comstreamStmts+*

*‘}’*

pipeline的主体部分包括两部分：第一部分是pipeline内部要使用到的变量的定义和声明；第二部分是pipeline内的操作语句，操作语句主要包括一般的C语言语句、composite调用语句以及pipeline和splitjoin结构，对于composite调用、splitjoin和pipeline结构的调用主要通过add操作符确定它们之间的数据依赖关系，add语句确定了pipeline内部的operator的连接。

如stream<int x>Out=pipeline(In)

{

add Comp1(p1);

add Comp2(p2);

add Comp3(p3);

}

由于pipeline结构是单输入流和单输出流的，我们规定能够在pipeline中被调用的composite也必须是单输入流和单输出流的。这里由于都是单输入单输出，通过“add”操作确定数据在composite间的流向。因此，composite调用以及嵌套添加pipeline和splitjoin时不需要给出输入和输出数据流的任何信息，只需要给出参数即可。

**splitjoin结构**

splitjoin的结构主要有splitjoin的头部和splitjoin的主体组成，BNF表示如：

*splitjoinHead ::= ID ‘=’ ‘splitjoin’ opInput*

*| ‘splitjoin’*

*splitStmt ::= ‘split’ ( ‘duplicate’ ‘(‘ ‘)’ | ’roundrobin’ ‘(‘ expr\*‘)’ ) ‘;’*

*joinStmt ::=‘join’ ’roundrobin’ ‘(‘ expr\* ‘)’ ‘;’*

*splitjoinBody ::=’{’*

*declaration\**

*stmt\**

*splitStmt*

*comstreamStmts +*

*joinStmt*

*‘}’*

splitjoin结构是**关键字split、join**以及一些声明和语句组成。当一个流作为splitjoin结构的输入流时这个流首先被split结点分割，split节点是根据splitjoin结构中composite调用情况产生一批满足相应条件的输出流作为不同的composite调用的输入流，然后这些composite调用的输出流作为join结点的输入流，join同样根据composite调用情况来合并各个输入流，将合并后产生的新的流作为split结点的输出，同时也作为splitjoin结构的输入流。

**在splitjoin结构中split主要有两种方式：**

（1）duplicate方式，在这种方式下所有的composite调用将会有完全相同的输入流，即split结点将split的输入流完全的复制产生一批新的输出流作为composite调用的输入。

（2）roundrobin(w1,…,wn)方式，在这种方式下split结点将其输入流中的前w1个数据发送给第一个composite调用，接下来的w2个数据发送给第二个composite调用，以此类推，将最后的wn个数据发送给最后一个composite调用。

对于join结点只有一种roundrobin方式。与pipeline结构类似，在splitjoin结构中调用的composite也要求是单输入流和单输出流的。在COStream中split和join分别用关键字split和join表示。



图2： COStream中splitjoin和pipeline应用举例

图2中的例子能够充分的说明splitjoin和pipeline的层次性特点和使用特性。在图中左边部分是COStream数据流程序的源码部分，而右边是对应的数据流图。在本例中可以看到添加了层次性的结构有助于增加数据流程序编程的灵活性和可扩展性。另外在splitjoin和pipeline中流能够被参数化，在本例composite SJP中关于LowPassFilter和HighPassFilter的调用次数可以有参数N来确定，通过N来控制splitjoin结构的宽度，同理在pipeline中也可以通过参数来控制pipeline的深度。