**基本模块**

**数据流编程框架**

用户界面的菜单栏中加入了COStream Run菜单项，在该菜单项下存在两个子菜单项，分别是compile和run，代表数据流程序执行的两个部分，程序员新建COStream工程，在代码编辑框编写好流程序后，点击compile菜单项，插件会调用流语言编译器将COStream源程序转换成C++目标代码，目标代码会存放在后台的临时文件夹中，然后执行run菜单项，插件会找到上一步生成的目标代码文件夹，编译并链接生成的文件，得到流语言的可执行程序，运行这一程序得到流语言源程序的结果。流语言开发与Eclipse默认的java语言的开发有着明显的差异性，因此，在插件中加入针对流程序的COStream透视图，如下图1所示，在该透视图下，工程视图模块，代码编辑模块和流图视图模块的比例是1:3:1，菜单栏中的新增项只在COStream透视图中显示。

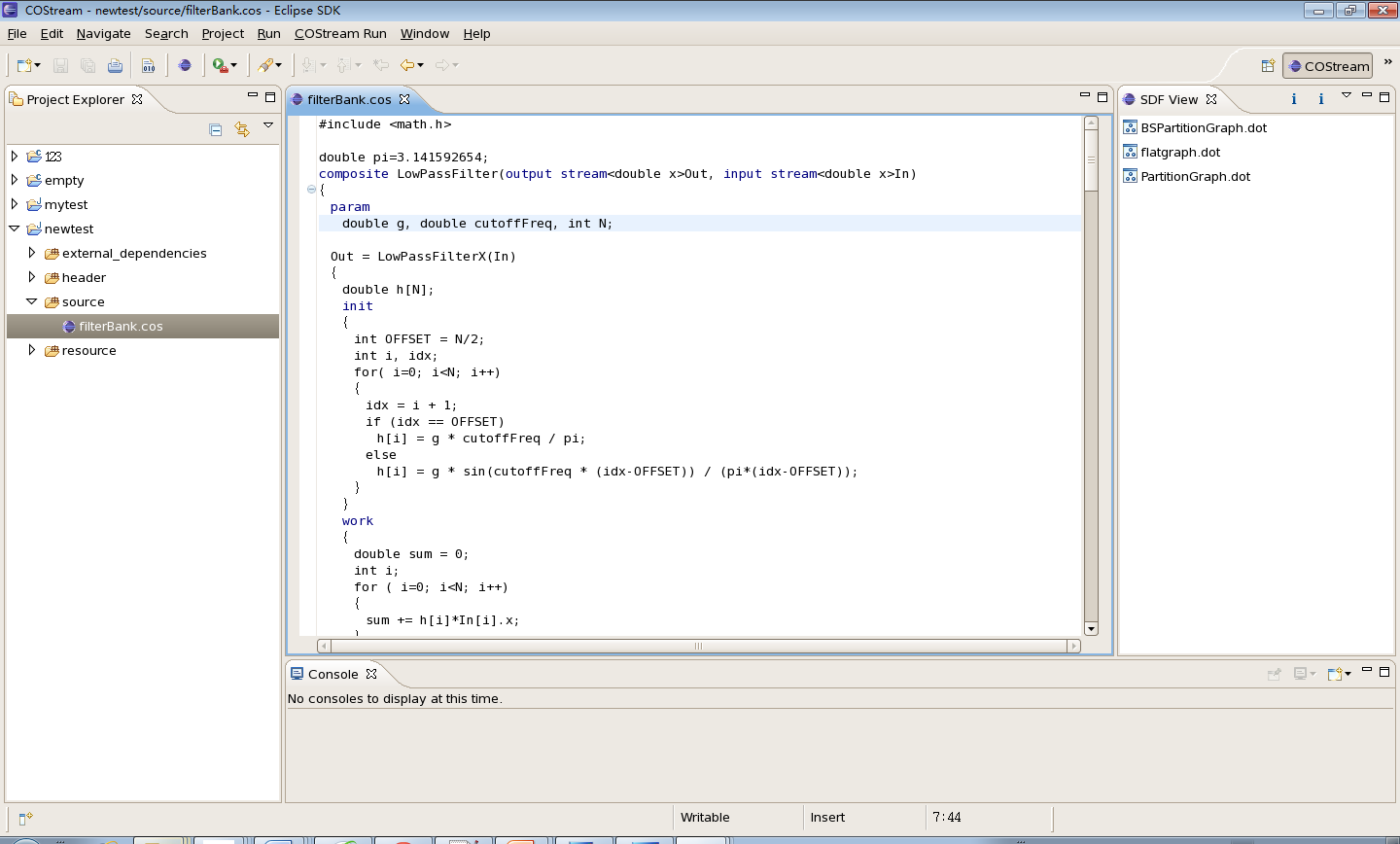


图1：数据流可视化编程环境透视图

**代码编辑模块**

在Eclispe-plugin开发环境中提供了对textEditor的扩展接口，本项目使用这一扩展接口继承实现了cosEditor，在这一编辑器中将composite，operator，stream，add，init，work，window等关键字进行高亮显示。为实现cosEditor中的错误预提示，需要用到COStream编译器中的错误输出结果。在COStream编译器中，设计了当流程序编译出错时的错误输出，并指定了错误类型以及产生错误的位置信息，代码编辑模块遍历流编译器编译产生的所有错误信息，找出错误行数以及错误类型，并在cosEditor上标注出这些信息，流程序员在编写完COStream数据流程序后按保存键，代码编辑模块会在后台调用流编译器，执行上述过程，在cosEditor中显示错误信息，从而简化了程序员的编程，让程序员减少不必要细节的时间开销。

**流语言编程接口模块**

在Eclipse插件开发中，官方提供了针对插件开发的编程接口，要改变工程的相关参数，有两种方式，第一种是在菜单栏的project菜单项下的工程属性部分添加，这种修改只会改变当前的单个工程的配置属性；第二种是在windows菜单项的首选项子菜单当中添加，这种方式改变的是整个开发环境的配置参数。如下图2所示，考虑到流程序接口配置的优先级，将流语言参数相关设定放在首选项中。

Eclipse-plugin针对首选项提供了preferencePages扩展点，在该扩展点下完成流语言编程接口相应参数的添加，包括机器数和机器上参与运行的核数，后端平台的选择以及一些其他的选项。若运行在X86或者GPU后端，则默认可用机器数为1，只需要设置可运行的核数，若目标代码运行在集群后端，则运行的机器数是可以指定的，同时可以指定每个机器上参与运行的物理核的个数。

要实现上述功能，必须在COStream编译器内部预留了相应的接口，则通过首选项的响应为COStream编译器添加相应的参数，则当前流语言编译时执行的是带运行选项的编译命令，因此可以让程序员可视化的操作流语言编译选项。

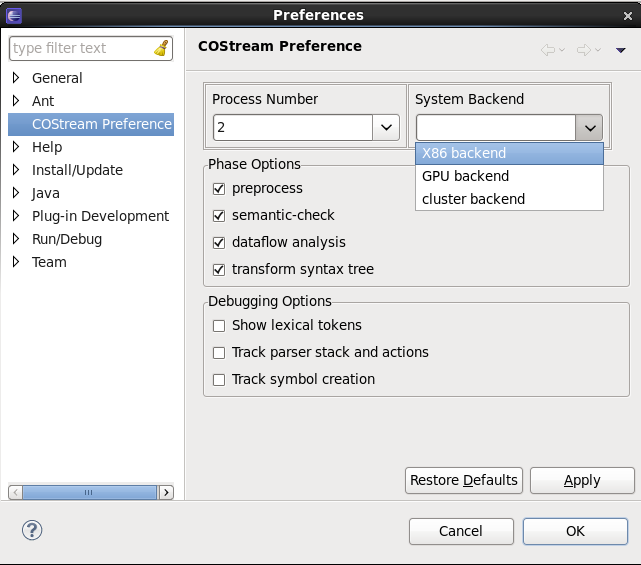


图2：数据流插件编程接口图

**流图显示模块**

在COStream数据流程序中，通过静态数据流图(SDF)来表示流程序展开后的结构，SDF图包含流程序的各个计算节点(actor)以及连接actor的通行边，数据流程序员通过分析SDF图的结构，能够清晰的了解各个actor工作量，流图划分情况。因此，SDF图的显示是COStream流编程步骤当中的重要一环，在插件中添加流图显示模块有助于程序员分析流程序的结构特点，对编写的程序作出改进。若需要得到某一个SDF图，则需点击SDF视图中相应的列表项，则插件会跳转到Image视图，如下图3所示，对相应的SDF图在Image视图中显示。

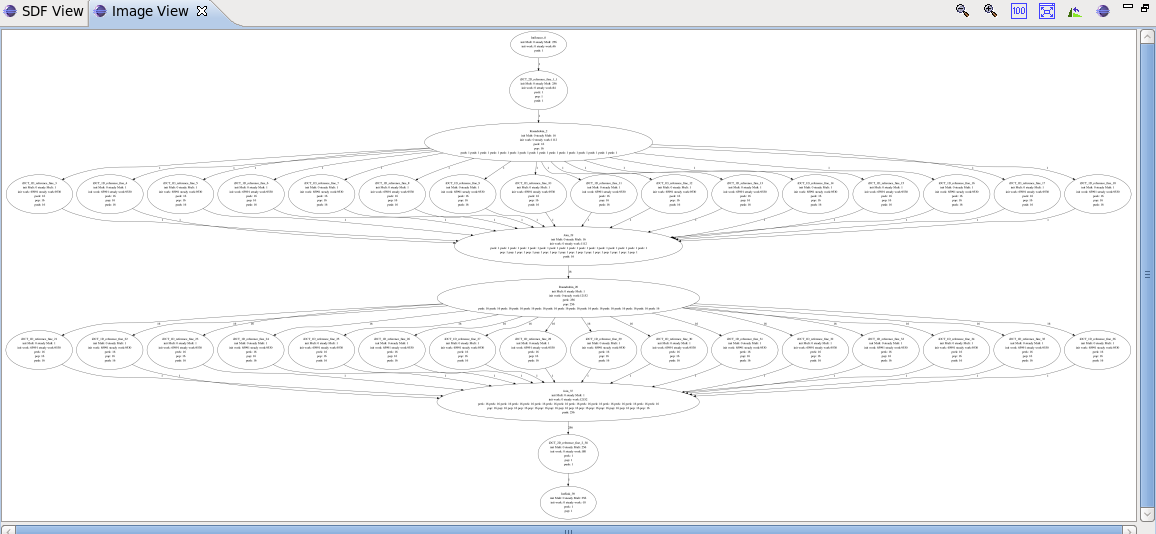


图3：SDF视图和Image视图

**性能分析模块**

当流程序编写完成后，性能分析工具会分析首选项中选中的后端信息，来调用不同的分析工具，如果流程序运行在X86后端，在该程序编译运行的过程中性能分析工具会先分析流程序计算，同步，通信等模块的时间开销，并统计每一个核运行这几部分所需的时间以及所需时间占的百分比，结果显示在性能分析视图中，集群后端的性能分析界面如下图4所示。在流程序运行结束后性能分析工具并不会结束，性能分析工具会在后台运行同一个程序的串行执行版本，并计算串行执行的总时间，通过计算上述时间，系统会得到程序的加速比，并在性能分析视图中显示。同时，流程序员还可以在性能分析视图中设定扩大执行的次数，得到计算资源充足时的流程序加速比。性能分析视图在COStream透视图中并不会默认出现在用户界面，需要在windows菜单栏下的COStream Performance视图中调出，COStrea -m Performance视图右键的refresh菜单项可以刷新视图中的性能分析结果。

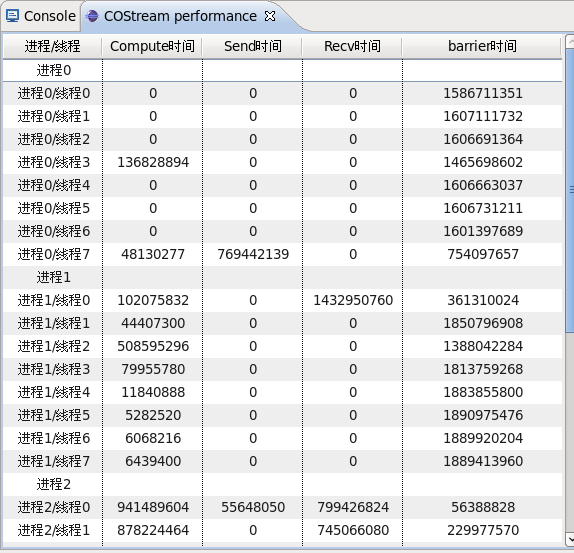


图4：流程序编程环境集群后端性能分析图

**流程序调试模块**

通过对COStream语言和C++目标代码之间建立映射关系，可以在COStream编译器的代码生成模块添加代码，在每一个目标代码的生成行之前加入#line directives信息，标示当前生成行在COStream源代码中是在哪一行。而在可视化开发的过程中加入对后缀名为.cos的COStream流程序的断点支持。可视化开发环境用gdb调试器作为流程序调试的默认调试器。当程序员调试COStream程序时，开发工具找到程序的下一个中断位置，并在跨语言映射哈希表中找到对应的COStream程序的位置，#line命令是gdb调试器中的跨语言调试命令，如下图5所示，调试器通过这一命令可以找到当前断点的所有可用变量的信息，并加入到Eclipse的变量视图中，在下图6中，程序员可打开变量视图找到COStream断点处的可见变量的值。

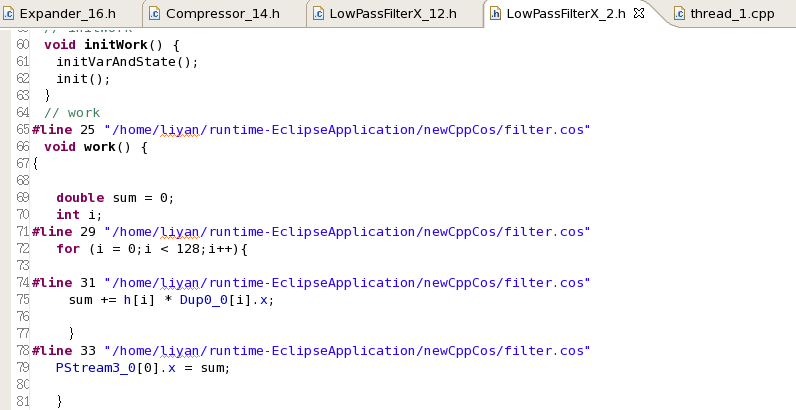


图5：流程序gdb调试功能目标代码映射关系

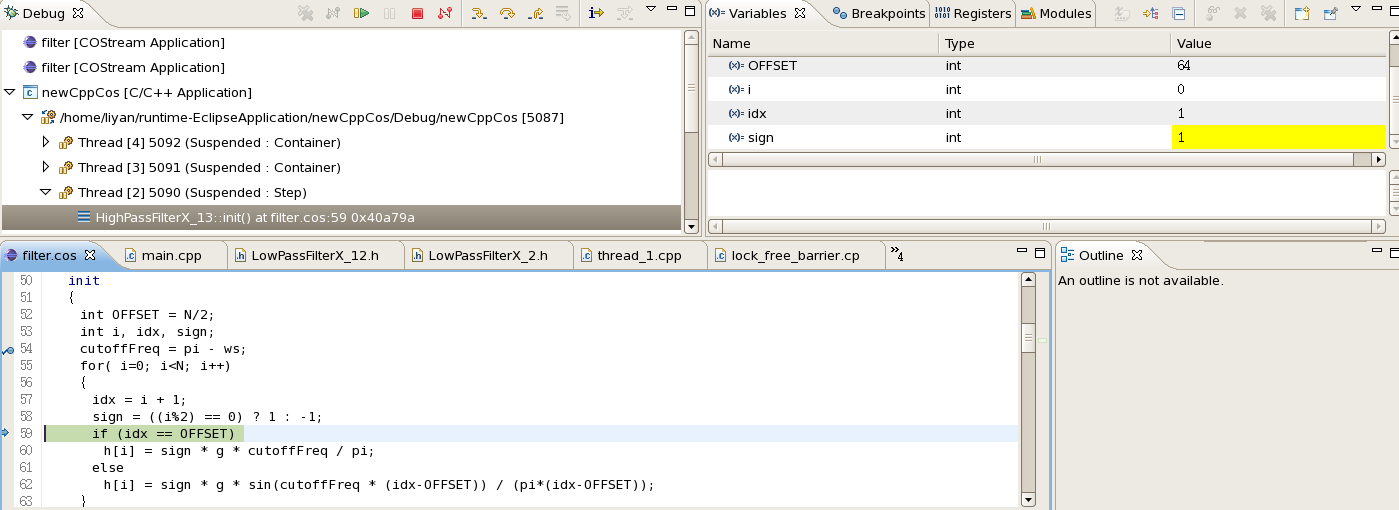


图6：流程序调试示例图