数据采集工具-基础知识讲解

数据采集工具-概念

数据采集工具是基于 Apache NiFi 的一个流编程概念的数据流系统。它支持数据路由、转换和系统中介逻辑的强大且可扩展的有向图，数据采集工具有一个基于web的用户界面，用于设计、控制、反馈和监控数据流。它在服务质量的几个维度上都是高度可配置的，例如容错与保证交付、低延迟与高吞吐量以及基于优先级的排队。NiFi为所有接收的、分叉的、连接的、克隆的、修改的、发送的数据提供了细粒度的数据源，并在到达其配置的最终状态时最终丢弃。

数据采集工具-核心概念

NIFI术语	FBP术语	描述
DataFlow Manager(DFM)		是数据采集工具用户，具有添加，删除和修改数据采集数据流组件的权限。
FlowFile	信息包	FlowFile表示在系统中移动的每个对象，对于每个对象，NiFi跟踪key/value 属性字符串的映射及其相关的零个或多个字节的内容。
FlowFile Processor	黑盒	**数据采集工具 核心组件**, 处理器可以访问给定 FlowFile 的属性及其内容。处理器可以对给定工作单元中的零或多个 FlowFile 进行操作, 并提交该工作或回滚该工作。 日常工作我们称之为处理器。
Connection processor	缓冲区	Connections （连接）用来连接 Processer（处理器）, 它们充当队列并允许各种进程以不同的速率进行交互。这些队列可以动态地进行优先级排序, 并且可以在负载均衡设置上限, 从而实现反压机制。
Flow Controllers	调度器	FlowFile Controllers维护流程如何连接, 并管理和分配所有流程使用的线程。流控制器充当代理, 促进处理器之间流文件的交换。
Process Group	分支网络	Process Group 里是一组特定的流程和连接, 可以通过输入端口接受数据并通过输出端口发送数据, 这样我们在进程组里简单地组合组件, 就可以得到一个全新功能的组件。

参照上述表格，简单来讲FlowFile是在各个节点间流动的数据；FlowFile Processor 是数据的处理模块；Connection是各个处理模块间的一个队列；Flow Controllers是复杂流程的调度；Process Group一些相关连的Processor可以封装到一个Process Group中，不同组用来表示不同流程的层次关系。

FlowFile

FlowFile代表数据采集工具中的单个数据。FlowFile由属性(attribute)和内容(content)组成。内容是FlowFile表示的数据，属性由键值对组成，提供有关数据的信息或上下文的特征。

每个FlowFile都拥有多个属性,这些属性将在FlowFile的生命周期中发生变化。FlowFile三个主要优点。

- 它允许用户在流中做出路由决策,以便满足某些条件的FlowFiles可以与其他FlowFiles进行不同地处理。这可以由RouteOnAttribute和其他类似的处理器完成的。
- 利用属性配置处理器：处理器的配置依赖于数据本身。例如,PutFile能够使用Attributes来知道每个FlowFile的存储位置,而每个FlowFile的目录和文件名属性可能不同（结合表达式语言,比如每个流都有filename属性,组件中就可以这样配置文件名：${filename},就可以获取到当前FlowFIle中filename的属性值）。
- 属性提供了有关数据的极有价值的上下文。在查看FlowFile的Provenance数据时非常有用,它允许用户搜索符合特定条件的Provenance数据,并且还允许用户在检查Provenance事件的详细信息时查看此上下文。通过简单地浏览该上下文,用户能够知道为什么以这样或那样的方式处理数据。

共同属性

每个FlowFile都有一组属性：

- filename：可用于将数据存储到本地或远程文件系统的文件名。
- path：可用于将数据存储到本地或远程文件系统的目录的名称。
- uuid：一个通用唯一标识符,用于区分FlowFile与系统中的其他FlowFiles。
- entryDate：FlowFile进入系统的日期和时间（即已创建）。此属性的值是一个数字,表示自1970年1月1日午夜（UTC）以来的毫秒数。
- lineageStartDate：任何时候克隆,合并或拆分FlowFile,都会导致创建子FlowFile。该值表示当前FlowFile最早的祖先进入系统的日期和时间。该值是一个数字,表示自1970年1月1日午夜（UTC）以来的毫秒数。
- fileSize：此属性表示FlowFile内容占用的字节数。 需要注意的是uuid,entryDate,lineageStartDate,和fileSize属性是系统生成的,不能改变。

提取属性

数据采集工具提供了几种不同的处理器,用于从FlowFiles中提取属性。这是构建自定义处理器的一个非常常见的用例，其实编写处理器是为了理解特定的数据格式，并从FlowFile的内容中提取相关信息，创建属性来保存该信息，以便可以决定如何路由或处理数据。

添加用户自定义的属性

数据采集工具除了提供能够将特定信息从FlowFile内容提取到属性中的处理器之外，数据采集工具还允许用户将自定义属性添加到每个FlowFile中的特定位置。UpdateAttribute就是专为此目的而设计。用户可以通过单击属性选项卡右上角的+按钮,在配置对话框中向处理器添加新属性。然后UI会提示用户输入属性的名称,然后输入值。对于此UpdateAttribute处理的每个FlowFile，都会添加用户自定义属性。Attribute的名称将与添加的属性的名称相同。

属性的值也可以包含表达式语言。这样就允许基于其他属性修改或添加属性。例如,如果我们想要将处理文件的主机名和日期添加到文件名之前,我们可以通过添加${hostname()}-${now():format('yyyy-dd-MM')}-${filename}来实现来实现。刚开始大家可能不太理解这是什么意思，在后续的课程中我们会进行讲解。

除了添加一组自定义的属性外,UpdateAttribute还具有一个高级UI,允许用户配置一组规则，以便在何时添加属性。要访问此功能,请在配置对话框的属性选项卡中，单击Advanced对话框底部的按钮。将弹出此处理器特定的UI界面。在此UI中，用户可以配置规则引擎，实质上是指定必须匹配的规则，以便将已配置的属性添加到FlowFile。

属性路由

数据采集工具最强大的功能之一是能够根据属性路由FlowFiles。执行此操作的主要机制是RouteOnAttribute。此处理器与UpdateAttribute一样，通过添加用户自定义的属性进行配置。通过单击处理器的配置对话框中属性选项卡右上角的+按钮,可以添加任意数量的属性。

每个FlowFile的属性将与配置的属性进行比较，以确定FlowFile是否满足指定的条件。每个属性的值应该是一个表达式语言并返回一个布尔值。下面的【表达式语言/在Property值中使用attribute】会对表达式语言进行补充。

在评估针对FlowFile的属性提供的表达式语言之后，处理器根据所选的路由策略确定如何路由FlowFile。最常见的策略是"Route to Property name"策略。选择此策略后，处理器将为配置的每个属性公开关系(可拖拽出去指向下一个处理器)。如果FlowFile的属性满足给定的表达式，则FlowFile的副本将路由到相应的Relationship。例如,如果我们有一个名为"begin-with-r"的新属性和值"$ {filename：startsWith（'r'）}"的表达式，那么任何文件名以字母'r'开头的FlowFile将是路由到那个关系。所有其他FlowFiles将被路由到"unmatched"关系。

表达式语言/在Property值中使用attribute

当我们从FlowFiles的内容中提取属性并添加用户定义的属性时，除非我们有一些可以使用它们的机制,否则它们不会作为运算符进行计算。NiFi表达式语言允许我们在配置流时访问和操作FlowFile属性值。并非所有处理器属性都允许使用表达式语言,但很多处理器都可以。为了确定属性是否支持表达式语言,用户可以将鼠标悬停在处理器配置对话框的属性选项卡中的图标上,然后会有一个提示,显示属性的描述,默认值（如果有）以及属性是否支持表达式语言。

对于支持表达式语言的属性,可以通过在 开始标记 ${ 和结束标记 } 中添加表达式来使用它。表达式可以像属性名一样简单。例如,要引用uuid Attribute,我们可以简单地使用 ${uuid}。如果属性名称以字母以外的任何字符开头,或者包含除数字,字母,句号（.）或下划线（_）以外的字符,则需要加引号。例如,${My Attribute Name} 将无效,但${'My Attribute Name'}将引用属性My Attribute Name。

除了引用属性值之外,我们还可以对这些属性执行许多功能和比较。例如,如果我们想检查filename属性是否不分大小写（大写或小写）地包含字母'r',我们可以使用表达式来完成${filename:toLower():contains('r')}。函数由冒号分隔。我们可以将任意数量的函数链接在一起，以构建更复杂的表达式。即使我们正在调用filename:toLower(),这也不会改变filename属性的值，而只是返回给我们一个新的值。

我们也可以在一个表达式中嵌入另一个表达式。如果想要将attr1 Attribute 的值与attr2 Attribute的值进行比较,我们可以使用以下表达式来执行此操作：${attr1:equals(${attr2})}。

表达式语言包含许多不同的函数,官方文档Expression Language Guide。

此外，此表达式语言指南内置于应用程序中，以便用户可以轻松查看哪些功能可用，并在输入时查看其文档。设置支持表达式语言的属性的值时，如果光标位于表达式语言的开始和结束标记内，则在关键字上按 Ctrl + Space 将弹出所有可用的函数(快捷键冲突被占用会无法使用此功能)，并将提供自动填充的功能。单击或使用键盘上下键指向弹出窗口中列出的某个功能会有提示，提示解释了该功能的作用，它所期望的参数以及函数的返回类型。

表达式语言中的自定义属性

除了使用FlowFile属性外,还可以定义表达式语言使用的自定义属性。定义自定义属性为处理和配置数据流提供了额外的灵活性。

应用的方式是：将处理数据的工序做成黑盒处理器，由多个处理器经过连接管道串成一个流程，数据逐个处理器最终以想要的形式流向目的地。

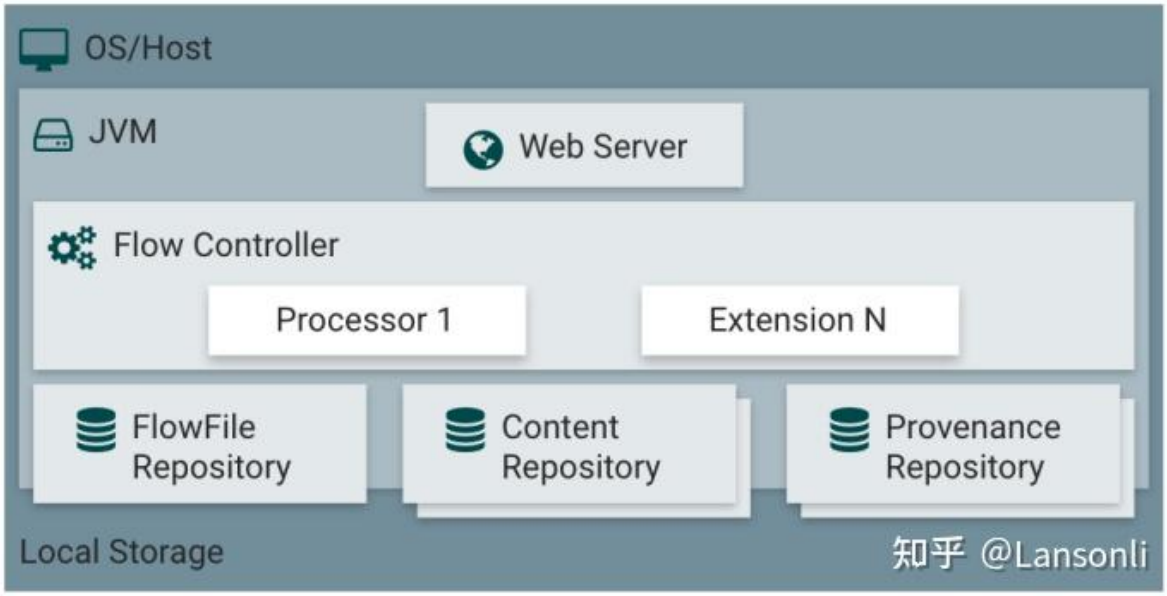
数据进入一个节点（处理器），由该节点（处理器）对数据进行处理，根据不同的处理结果将数据路由到后续的其他节点（处理器）进行处理。这是NiFi的流程比较容易可视化的一个原因。

FlowFile Processor

为了创建高效的数据流处理流程,需要了解可用的处理器（Processors ）类型，数据采集工具提供了大约近300个现成的处理器。这些处理器提供了可从不同系统中提取数据,路由,转换,处理,拆分和聚合数据以及将数据分发到多个系统的功能。如果还不能满足需求，还可以自定义处理器。

每个新的NiFi版本都会有新的处理器，下面将按照功能对处理器分类，介绍一些常用的处理器。具体可参照官网查看更多的处理器信息：https://nifi.apache.org/docs/nifi-docs/html/getting-started.html#what-processors-are-available

数据采集工具-架构



数据采集工具是基于Java开发的，所以运行在JVM之上。NiFi的核心部件在JVM中的位置如上图：

- **Web Server (Web 服务器):**

  Web服务器的目的是承载数据采集工具基于http的命令和控制API。

- **Flow Controller(流控制器):**

  Flow Controller是数据采集工具执行具体操作的大脑，负责从线程资源池中给Processor分配可执行的线程，以及其他资源管理调度的工作。

- **Extensions(扩展):**

  数据采集工具中有各种Processor及扩展。这些扩展也是运行在JVM中的。

- **FlowFile Repository(FlowFile 存储库):**

FlowFile Repository 负责保存在目前活动流中FlowFile的状态。FlowFile Repository的实现是可插拔的（多种选择，可配置，甚至可以自己实现），默认实现是使用Write-Ahead Log技术写到指定磁盘目录。

- **Content Repository(内容存储库):**

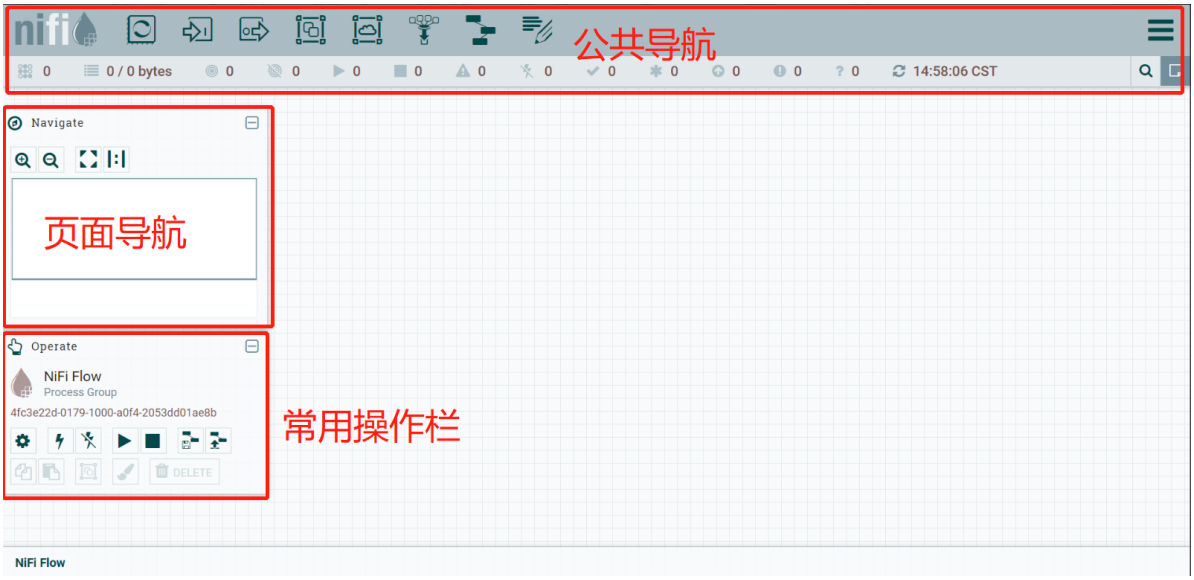
  Content Repository负责保存在目前活动流中FlowFile的实际字节内容。其功能实现是可插拔的。默认的方式是一种相当简单的机制，即存储内容数据在文件系统中。多个存储路径可以被指定，因此可以将不同的物理路径进行结合，从而避免达到单个物理分区的存储上限。

- **Provenance Repository(源头存储库):**

  源头存储库是存储所有源事件数据的地方，同样此功能是可插拔的，并且默认可以在一个或多个物理分区上进行存储，在每个路径下的事件数据都被索引，并且可被查询。

数据采集工具-WEB页面介绍

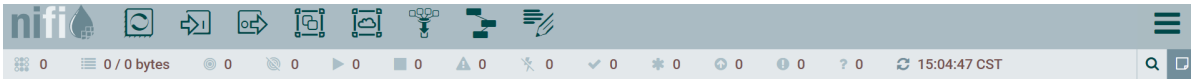
主页面



进入主页面以后，它整体就是一个画布的形式，最上方是个公共导航栏，左侧那个Navigate没啥用，不用在意，就是一个全局视角，下面的Operate是组件控制面板，可以进行单个组件的控制，也可以选中一片组件进行统一的启动，停止等等。

面板介绍

首先，刚刚已经把Nifi的整个页面理解为一个工作台，最上方就是个导航栏了，从最上面开始，这里的导航栏分为两部分，上半部分是提供给我们工作的，下半部分是对整个Nifi环境下的一个监控信息。这里简单介绍一下：



工作导航栏

导航栏中的这个菜单，我们可以理解为处理器（Processor）商城，用鼠标单击拖出到画布上，便会出现处理器（Processor）菜单

 导航栏中这个菜单，我叫它为组，什么叫组呢，当你拉了很多处理器（Processor），形成了一个完整的流程的时候，我们可以单独把这块划分成一个整体了，这时候就要用组把它包裹起来。

 有了组以后，组和组之间可能也需要联通、通信，这时候就可以用入口和出口，把它们放在组内

 这个组件需要配合 **Operate** 中的 上传使用，主要是用来迁移模板的，这块后续会专门抽章节讲一下

 这一组件，是集群Nifi进行数据通信的时候用的

 这一组件，就是个便签，用来写个备注呀啥的

 这一组件就是个漏斗，主要作用就是把四散的数据可以汇集在一起。

 全局导航栏

NIFI工作方式

这里侧重点是Nifi中的处理器应用，关于集群、组配合等等方式不在此篇记录的重点中。

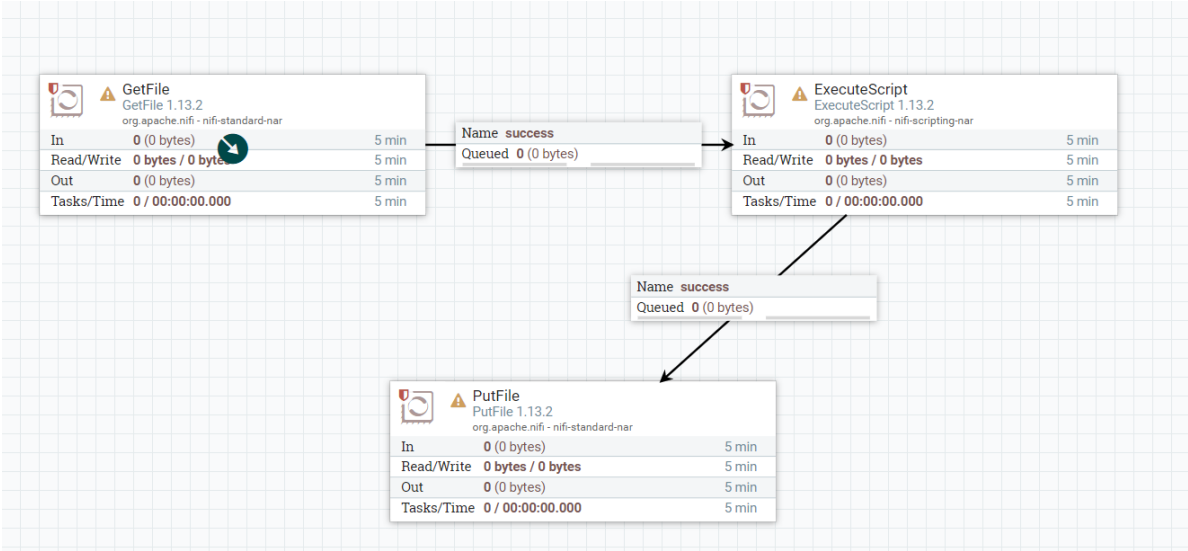
基本方式

首先回顾，Nifi其实就是**一个数据接入、处理、清洗、分发的系统**，它的工作方式就是将数据看作水管中的水，它是顺着某个流程管道流动，在这中间，可以在任意节点处堵截这个“水流”，并对它进行改造，然后放回管道继续向下流去。

这里的节点，其实就是Nifi的Processor，你叫它处理器也可以，叫他组件也好，它就是一个黑盒小模块，不同的模块有不同的功能

然后，节点和节点直接的通道，在Nifi里叫Relationship，我把它称之为管道，就像水管一样，它本身的意义就是充当水管，把上节点处理完的水传下去。

在nifi中，都是一个个的流程（处理器+管道），形成一个数据的处理通路。

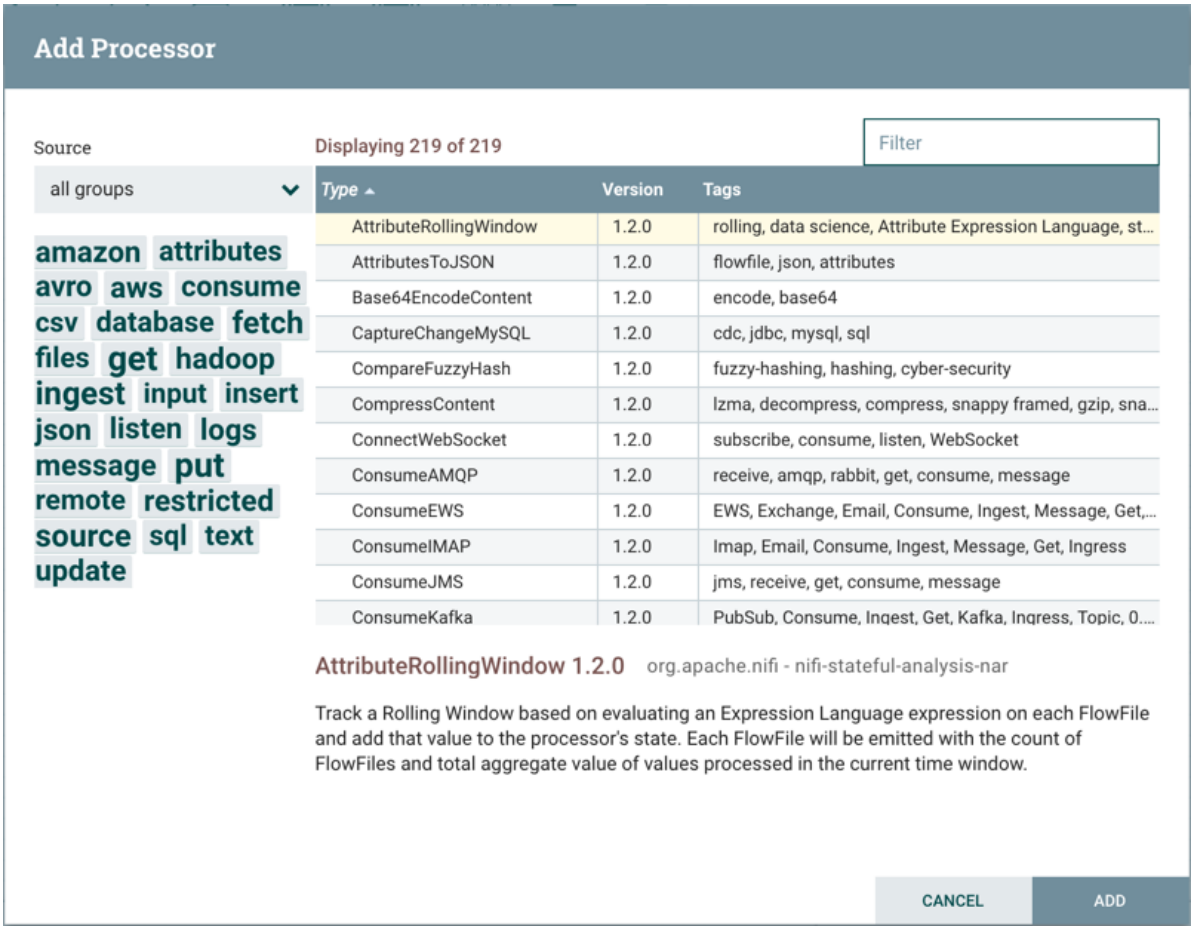


像这个例子，GetFile组件负责从一个文件里读取数据，然后把读到的数据通过管道传到ExecuteScript组件（这个组件支持用脚本代码处理数据），经过ExecuteScript之后，流向PutFile组件（将数据写入到指定文件中）。

基本流程就是： **选则一个处理器——>配置该组件至可运行状态——>关联下一组件建立管道**

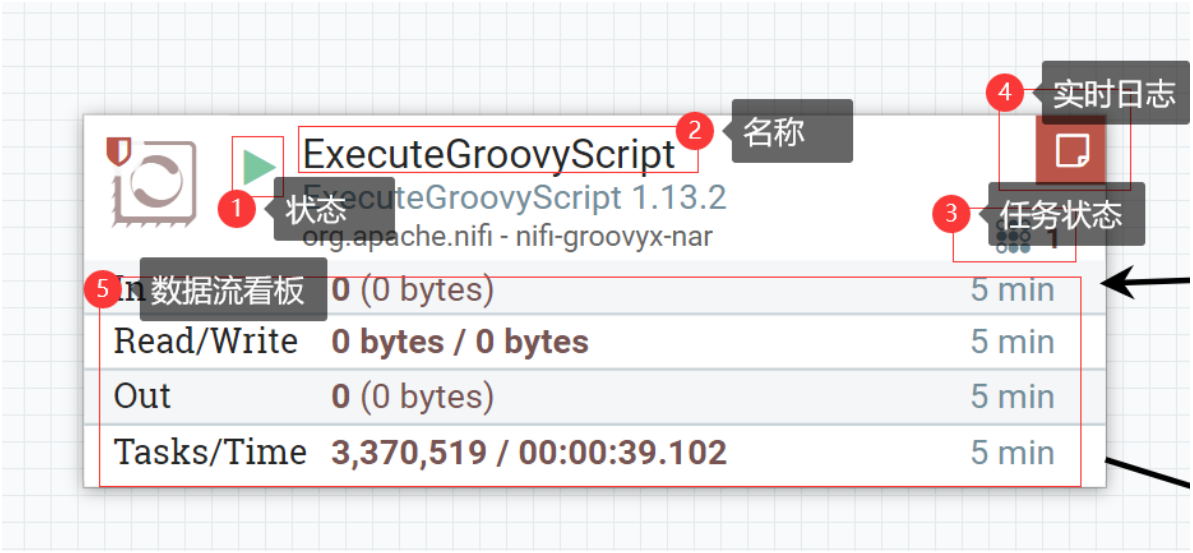
选择处理器

通过“组件商城” 图标进行处理器的选择， 处理器是最常用的组件,因为它负责数据的流入,流出,路由和操作。有许多不同类型的处理器。实际上,这是NiFi中非常常见的扩展点,这意味着许多供应商可能会实现自己的处理器来执行其所需的任何功能。将处理器拖动到画布上时,会向用户显示一个对话框：



这里可以通过处理器的包、处理器的属性、处理器的名称等维度进行组件的筛选、选择。选中后，双击则可拖拉至画布中。

处理器状态



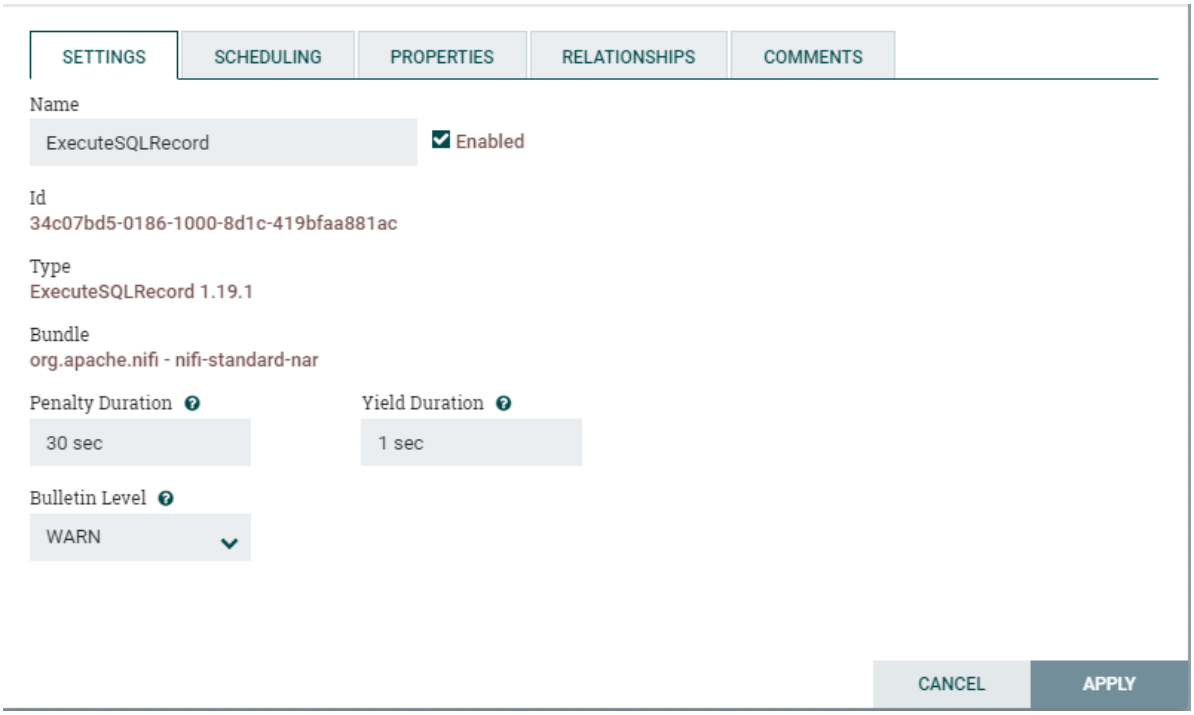
- **状态**：显示处理器的当前状态。以下指标是可能的：
  - 正在运行：处理器当前正在运行。
  - 已停止：处理器有效并已启用但未运行。
  - 无效：处理器已启用但当前无效且无法启动。将鼠标悬停在此图标上将提供工具提示,指示处理器无效的原因。一般情况下是需要我们完成必须的配置
  - 已禁用：处理器未运行,在启用之前无法启动。此状态不表示处理器是否有效。
- **名称**：这是处理器的用户定义名称。默认情况下组件的名称与它的Type相同。在示例中,此值为"ExecuteGroovyScript"，是一个专门用于执行Groovy脚本的组件。
- **任务**：此处理器当前正在执行的任务数。此数字受处理器配置对话框的计划选项卡中的并发任务设置的约束。在这里,我们可以看到处理器当前正在执行一项任务。如果NiFi实例是集群的,则此值表示当前正在集群中的所有节点上执行的任务数。
- **实时日志**：这里是用于监控当前处理器状态的，当处理器内部出现问题，一般会在此处显示错误日志
- **数据流入流出看板**：这里主要是展示处理数据过程中数据的流入流出情况，Nifi默认是5分钟更新一次页面上的看板情况，当然用户也可以在画布空白处，鼠标右键选择刷新，以达到实时查看的效果。
-
  - In：处理器从其传入处理器的队列中提取的数据量。此值表示为count/size,其中count是从队列中提取的FlowFiles的数量,size是这些FlowFiles内容的总大小
  - Read/Write：处理器从磁盘读取并写入磁盘的FlowFile内容的总大小。这提供了有关此处理器所需的I/O性能的有用信息。某些处理器可能只读取数据而不写入任何内容,而某些处理器不会读取数据但只会写入数据。其他可能既不会读取也不会写入数据,而某些处理器会读取和写入数据。
  - Out：处理器已传输到其出站连接的数据量。这**不包括处理器自行删除的FlowFiles**,也**不包括**路由到自动终止的连接的FlowFiles。与上面的"In"指标一样,此值表示为count/size,其中count是已转移到出站Connections的FlowFiles的数量,size是这些FlowFiles内容的总大小。
  - Tasks/Time：此处理器在过去5分钟内被触发运行的次数,以及执行这些任务所花费的时间。时间格式为hour：minute：second。请注意,所花费的时间可能超过五分钟,因为许多任务可以并行执行。例如,如果处理器计划运行60个并发任务,并且每个任务都需要一秒钟才能完成,则所有60个任务可能会在一秒钟内完成。但是,在这种情况下,我们会看到时间指标显示它需要60秒,而不是1秒。

处理器配置TAB

Nifi的处理器，一般都有四个标签页，分别是SETTINGS，SCHEDULING，PROPERITIES，COMMENTS，RELATIONSHIPS

除了PROPERITIES之外，另外四个几乎是通用的，这里主要说一下这四个实用的。

SETTINGS（通用配置）

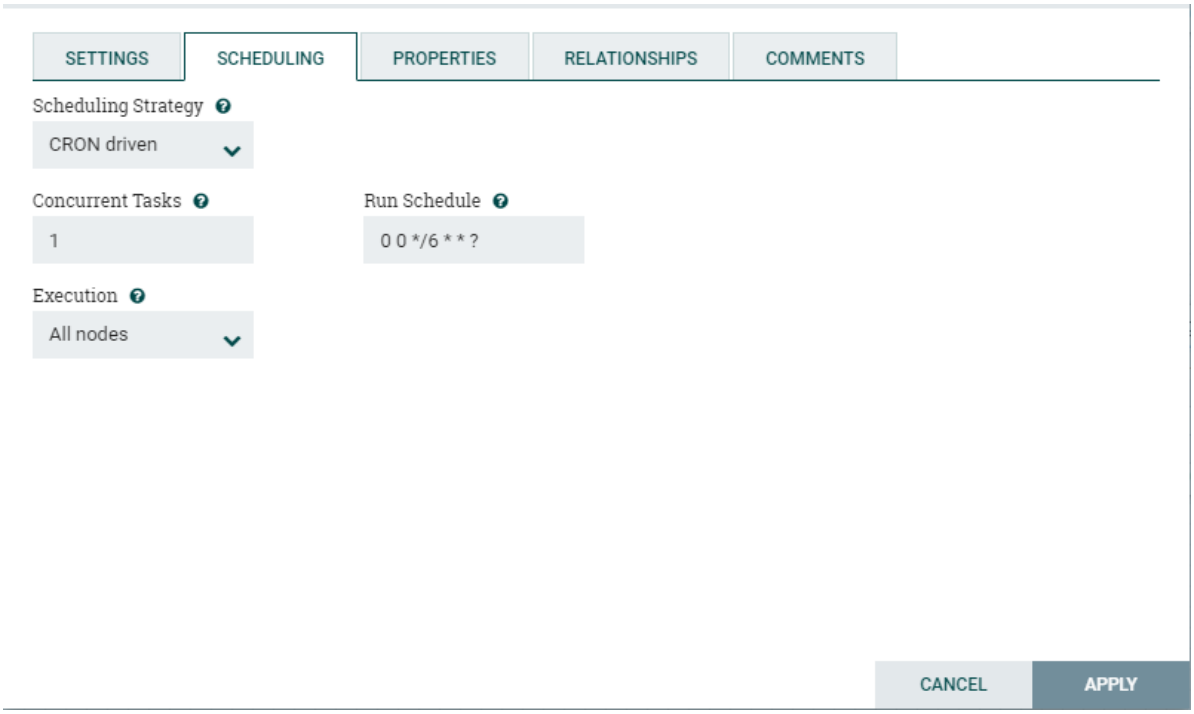


- 基本的Name这里就不说了，就是用户自定义的名称。
- Id、Type、Bundle这三个是这个处理器组件所属的代码包等基本信息，这里也不过多介绍。
- Enable这个选项，就是控制组件由启用到禁用 状态的切换。

- Penalty Duration的对话框。在处理一条数据(FlowFile)的正常过程中,可能发生事件,该事件指示处理器此时不能处理数据但是数据可以在**稍后**进行处理。在发生这种情况时,处理器可以选择Penalize FlowFile。

  这将阻止FlowFile在一段时间内被处理。例如,如果处理器要将数据推送到远程服务,但远程服务已经有一个与处理器指定的文件名同名的文件,则处理器可能会惩罚FlowFile。Penalty Duration允许DFM指定FlowFile应该受到多长时间的惩罚。默认值为30 seconds。(简单理解为推后一段时间再处理)，类似的处理器可以确定存在某种情况,处理器没法进行处理数据。例如,如果处理器要将数据推送到远程服务并且该服务没有响应。这样的话处理器应该Yield,这将阻止处理器运行一段时间。通过设置Yield Duration来指定该时间段。默认值为1 second。
- Yield Duration的对话框。类似的处理器可以确定存在某种情况,处理器**没法**进行处理数据。例如,如果处理器要将数据推送到远程服务并且该服务没有响应。这样的话处理器应该Yield,这将阻止处理器运行一段时间。通过设置Yield Duration来指定该时间段。默认值为1 second。
- Bulletin Level可以简单的理解为组件的日志输出等级的选择，有选择地进行日志等级输出

SCHEDULING处理器调度



这一标签页，代表的就是如何驱动处理器，或者说处理器的运作方式：

- 第一个配置选项是**调度策略（Scheduling Strategy）**。调度有三种可能的选项：
  - **Timer driven**：这是默认模式。处理器将定期运行。即多久运行一次，运行处理器的时间间隔由Run Schedule选项定义（当Run Schedule为0时，则代表瞬时执行）。
  - **Event driven**：选择此模式时,将由一个事件触发处理器运行,当FlowFiles进入连接此处理器的Connections时,将产生这个事件。此模式目前被认为是实验性的,并非所有处理器都支持。选择此模式时,Run Schedule选项不可配置。此外,只有此模式下Concurrent Tasks选项可以设置为0。这种情况,线程数仅受管理员配置的事件驱动线程池的大小限制。
  - **CRON驱动**：这是定时执行模式，即通过cron表达式，进行定时运行的控制。
- **线程的分配（Concurrent Tasks）**：就是配置就是线程的分配（Concurrent Tasks），这可以控制处理器将使用的线程数。

  换句话说,它控制此处理器应同时处理多少个FlowFiles。增加此值通常会使处理器在相同的时间内处理更多数据。但是,它是通过使用其他处理器无法使用的系统资源来实现此目的。这基本上提供了处理器的相对权重 - 应该将多少系统资源分配给此处理器而不是其他处理器。该字段适用于大多数处理器。但是,某些类型的处理器只能使用单个任务进行调度。

- **Execution**：执行设置用于确定处理器将被调度执行的节点。选择"All Nodes"将导致在集群中的每个节点上调度此处理器。选择"Primary Node"将导致此处理器仅在主节点上进行调度。一般单节点的情况下，我们都使用Primary Node
- **"Run Duration"选项卡**：的右侧包含一个用于选择运行持续时间的滑块。这可以控制处理器每次触发时应安排运行的时间。在滑块的左侧,标记为"Lower latency(较低延迟)",而右侧标记为"Higher throughput(较高吞吐量)"。

  处理器完成运行后,必须更新存储库才能将FlowFiles传输到下一个Connection。更新存储库的成本很高,因此在更新存储库之前可以立即完成的工作量越多,处理器可以处理的工作量就越多(吞吐量越高)。这意味着在上一批数据处理更新此存储库之前,Processor是无法开始处理接下来的FlowFiles。结果是,延迟时间会更长(从开始到结束处理FlowFile所需的时间会更长)。因此,滑块提供了一个频谱,DFM可以从中选择支持较低延迟或较高吞吐量。

PROPERITIES(属性区)

这一标签页差别较大，一般不同的组件所需要的配置各不相同，具体如果想了解对应组件的属性配置可以参考官网文档：http://nifi.apache.org/docs.html

RELATIONSHIPS（关系配置）

- 自动终止关系(Automatically Terminate Relationships)部分。此处列出了处理器定义的每个关系及其描述。为了使处理器被视为有效且能够运行,处理器定义的每个关系必须连接到下游组件或自动终止。我们可以通过选中它，例如图中选中Failure一样，来表示我们弃用这个输出，也就是不需要它指向下一个组件，这样这个处理器就变成只有一个对外输出数据的Relationship了。

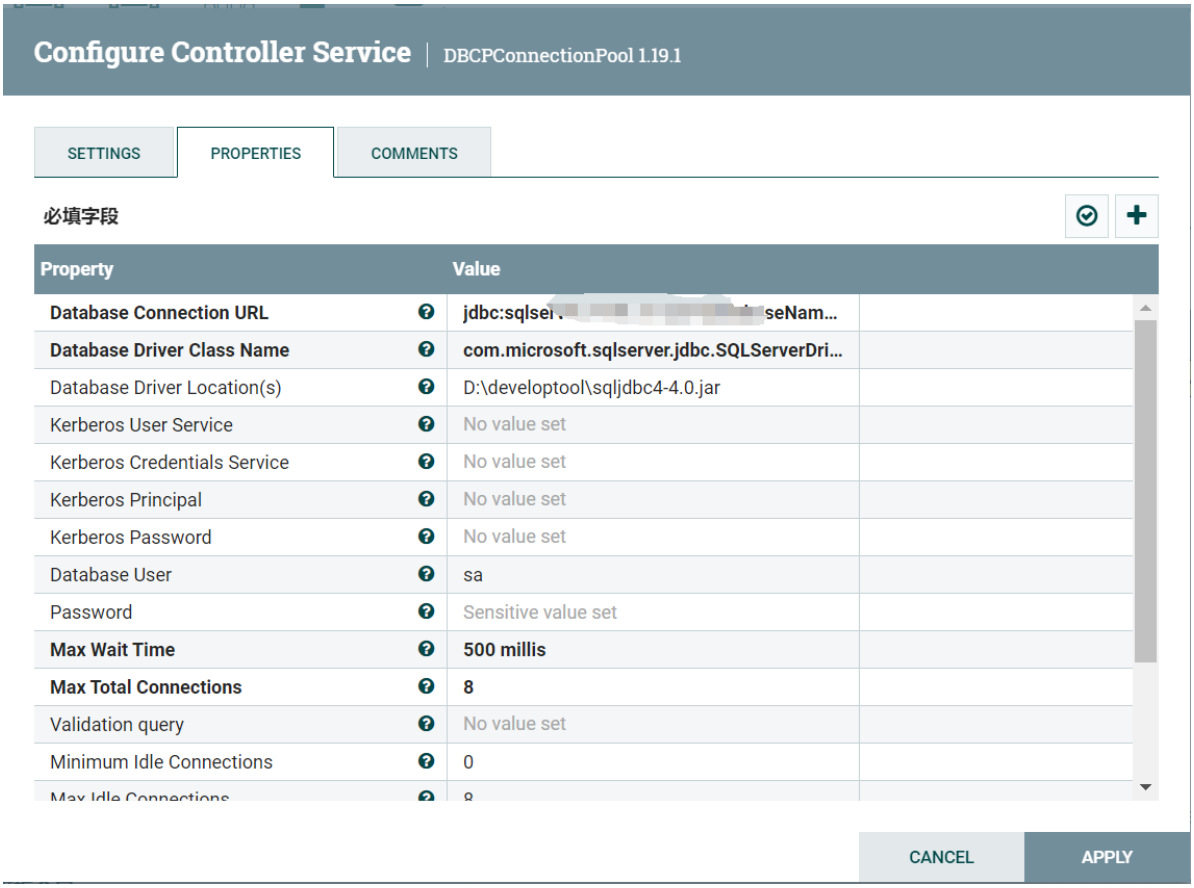
COMMENTS（备注区）

这块把它称之为”备注区“，即用来为用户提供一个区域,以包含适用于此组件的任何注释。

控制器服务配置页面

SETTINGS（通用配置）

PROPERITIES(属性区)



COMMENTS（备注区）