Kettle 中的集群

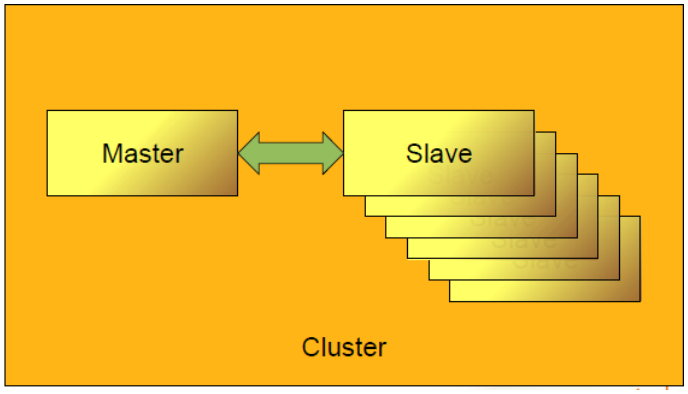
目录

Kettle 是一款开源的 ETL 工具，以其高效和可扩展性而闻名于业内。其高效的一个重要原因就是其多线程和集群功能。

Kettle 的多线程采用的是一种流水线并发的机制，我们在另外的文章中专门有介绍。这里主要介绍的是 kettle 的集群。

集群允许转换以及转换中的步骤在多个服务器上并发执行。在使用 kettle 集群时，首先需要定义的是 Cluster schema。所谓的 Cluster schema 就是一系列的子服务器的集合。在

一个集群中，它包含一个主服务器（Master）和多个从属服务器服务器(slave)。如下图所示。



子服务器（Slave servers）允许你在远程服务器上执行转换。建立一个子服务器需要你在远程服务器上建立一个叫做“Carte”的 web 服务器，该服务器可以从 Spoon(远程或者集群执行)或者转换任务中接受输入。

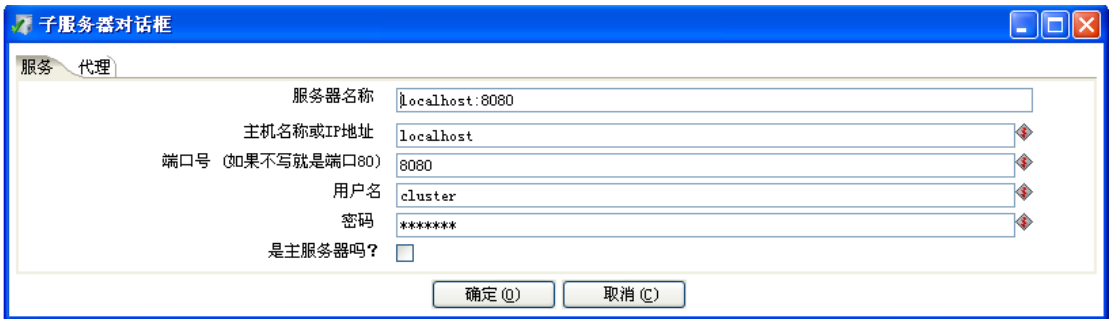
在以后的描述中，如果我们提到的是子服务器，则包括集群中的主服务器和从属服务器；否则我们会以主服务器和从属服务器来进行特别指定。

1 设计

要让转换是以集群方式执行，首先需要在 Spoon 中进行图形化的设计工作。定义一个以集群方式运行的转换，主要包括定义 cluster schema 和定义转换两个步骤。

1.1 定义 Cluster schema

1.1.1. 创建子服务器



服务 tab 选项

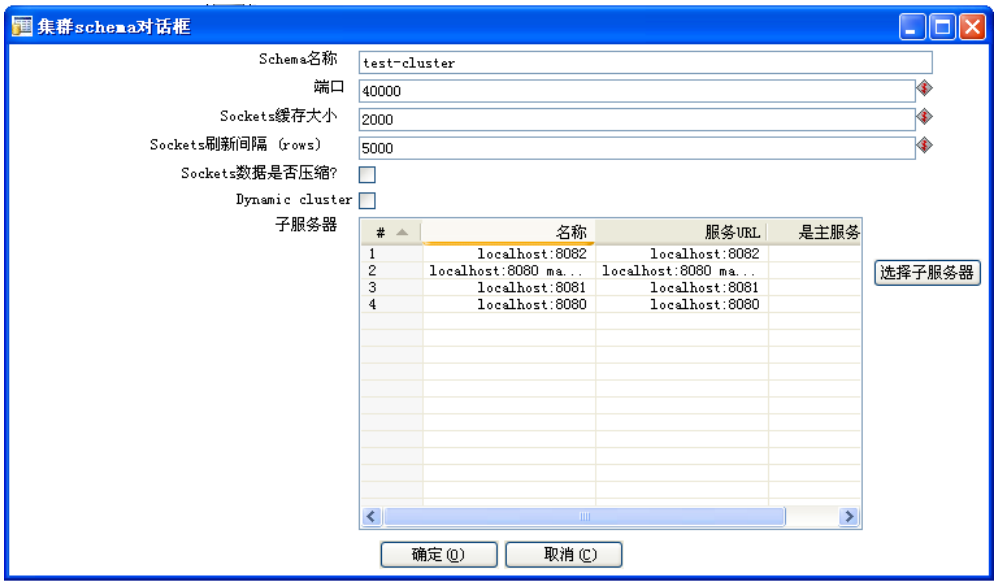
选项	描述
服务器名称	子服务器的名称
主机名称或 IP 地址	用作子服务器的机器的地址
端口号	与远程服务通信的端口号
用户名	获取远程服务器的用户名
密码	获取远程服务器的密码
是主服务器吗	在转换以集群形式执行时，该子服务器将作为主服务器

注意: 在集群环境下执行转化时，你必须有一个子服务器作为主服务器（master server）而其余所有的子服务器都作为从属服务器（slave）

Proxy tab options

选项	描述
代理服务器主机名	设置你要通过代理进行连接的主机名
代理服务器端口	设置与代理进行连接时所需的端口号
Ignore proxy for hosts: regexp|separated	指定哪些服务器不需要通过代理来进行连接。该选项支持你使用正则表达式来制定多个服务器，多个服务器之间以’|’字符来进行分割

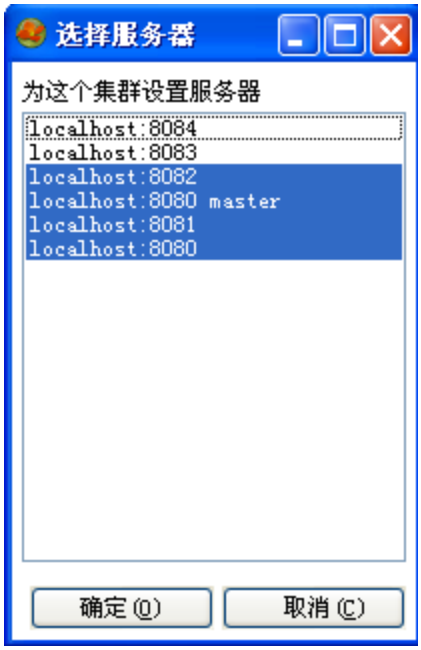
1.1.2. 创建 cluster schema



选项描述

选项	描述
Schema 名称	集群 schema 的名称
端口号	这里定义的端口号是指从哪一个端口号开始分配给子服务器。每一个在子服务器中执行的步骤都要消耗一个端口号。 注意：*确保没有别的网络协议会使用你定义的范围之类的端口，否则会引起问题*

Sockets 缓存大小	TCP 内部缓存的大小
Sockets 刷新间隔(rows)	当 TCP 的内部缓存通过网络完全发送出去并且被清空时处理的行数
Sockets 数据是否压缩	如果该选项被选中，则所有的数据都会使用 Gzip 压缩算法进行压缩以减轻网络传输量
Dynamic Cluster	动态集群指的是在运行的时候才能获知从属服务器的信息。这种情形适用于主机可以自动增加或者去除的情形，例如云计算。 主服务器的设置不变,但是它可以接受从属服务器的注册。一旦接受了某个从属服务器的注册，则每隔 30 秒去监视该从属服务器是否还处于有效状态
子服务器	这里是一个要在集群中使用的服务器列表。这个列表中包含一个主服务器和任意数目的从属服务器。 在 dynamic Cluster 的情况下，只需要选择主服务器即可



1.2 定义转换

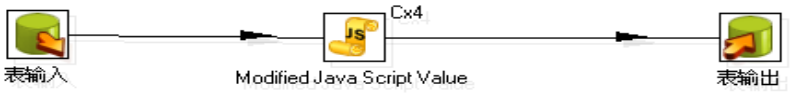
定义完了 cluster schema 后，下一步就是定义在集群环境下执行的转换。我们这里展现

的只是一个最简单的例子，完全是为了演示而用。现实情况中的集群有可能非常复杂。

首先你像平时一样创建转换，以 hop 连接连个两个步骤。然后你指定第二个步骤将在集群下执行

然后选择需要使用的集群。转换如图一样显示在 GUI 中。

注意 Cx4 显示这个步骤将在集群中运行，而这个集群中有 4 个从属服务器。假设我们将计算结果再次存入到数据表中



这个转换虽然定义了集群，但是我们同样可以让它在单机环境下执行，而且可以得到相同的结果。这意味着你可以使用普通的本地模式来测试它。

2 执行转换

要想以集群方式来运行转换或者作业，首先需要启动在 Cluster schema 中定义的主服务器和从属服务器，然后再运行转换或者作业。

2.1 启动子服务器

子服务器其实是一个嵌入式的名为Carte的小web server。要进行集群转换，首先需要启动cluster schema中的子服务器

2.1.1 脚本启动

kettle提供了carte.bat和carte.sh（linux）批处理脚本来启动子服务器，这种启动方式分为两种

2.1.1.1 使用主机号和端口号

Carte 127.0.0.1 8080

Carte 192.168.1.221 8081

2.1.1.2 使用配置文件

Carte /foo/bar/carte-config.xml

Carte http://www.example.com/carte-config.xml

如果cluster schema中定义了Dynamic cluster选项，则必须使用配置文件来进行启动，当这个子服务器启动时，它需要向配置文件中“masters”中列出的主服务器列表中汇报其运行状态（通过调用主服务器的registerSlave服务），已达到动态地设置子服务器的目的。配置文件格式

```
<slave_config>
<masters>
  <slaveserver>
    <name>master1</name>
    <hostname>localhost</hostname>
    <port>8080</port>
    <username>cluster</username>
    <password>cluster</password>
    <master>Y</master>
  </slaveserver>
</masters>

<report_to_masters>Y</report_to_masters>

<slaveserver>
  <name>slave4-8084</name>
```

```
    <hostname>localhost</hostname>
    <port>8084</port>
    <username>cluster</username>
    <password>cluster</password>
    <master>N</master>
  </slaveserver>
</slave_config>
```

这个配置文件主要包括以下几个节点

- masters: 这里列出来的服务器是当前子服务器需要向其汇报状态的主服务器。如果当前这个子服务器是主服务器，则它将连接其它的主服务器来获得这个集群中的所有子服务器。
- report_to_masters： 如果为 Y，则表示需要向定义的主服务器发送消息以表明该从属服务器存在
- slaveserver： 这里定义的就是当前 carte 实例运行时需要的子服务器的配置情况

这里定义的 username 和 password 在向主服务器调用 Register 服务时连接主服务器时提供的安全设置。 在 *〈slaveserver〉* 部分，你可以使用*〈network_interface〉* 参数，这个参数的优先级高于<hostname>参数 ，如果你的机器中安装有多个网卡，这个设置可以起作用。

2.1.2 程序启动

Kettle 提供了 org.pentaho.di.www.Carte 类，你可以通过该类提供的函数来启动或者停止子服务器。

- 启动子服务器

```
SlaveServerConfig config = new SlaveServerConfig(hostname, port, false);

Carte. runCarte(config);
```

- 停止子服务器

```
carte.getWebServer().stopServer();
```

2.1.3 子服务器内幕

我们前面提到过子服务器实际上就是一个 web server,该 web server 是基于 Jetty 这个嵌入式的开源 servlet 容器。

这个web server主要是提供转换运行的环境，另外一个重要的功能通过提供servlet来在客户端、主服务器和从属服务器之间进行通讯和控制。主服务器和从属服务器之间是通过httpClient来进行通讯的，通讯时传递的数据是xml格式。通过提供的servlet,可以实现启动、停止、暂停转换或者作业、获得转换或者作业的状态、注册子服务器、获得子服务器的列表等等

Kettle 主要提供了以下的几种基于 servlet 的服务

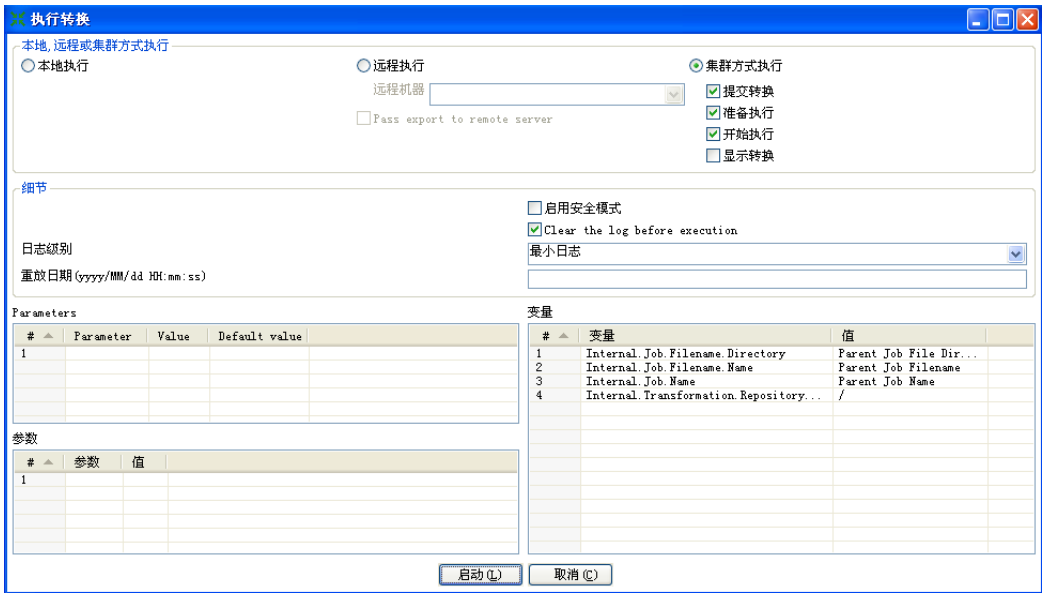
- GetRootServlet：获得 Carte 的根目录
- GetStatusServlet：获得在服务器上运行的所有的转换和作业的状态
- GetTransStatusServlet：获得在服务器上运行的某个指定的转换的每个步骤的运行状态。
- PrepareExecutionTransServlet：让服务器上的某个指定的转换做好运行的准备。
- StartTransServlet：执行服务器上的某个指定的转换

- PauseTransServlet：暂停或者重新运行某一个转换
- StopTransServlet：停止正在运行的转换
- CleanupTransServlet：清理运行转换时的环境
- AddTransServlet：向子服务器中增加某个转换。如果服务器中有正在运行或者准备运行的相同名字的转换，则抛出异常。
- AllocateServerSocketServlet：分配一个新的 socket 端口号。这个端口号是基于你在定义 cluster schema 中设置的端口号，依次加 1
- StartJobServlet：执行服务器上某个指定的作业
- StopJobServlet:停止正在运行的作业
- GetJobStatusServlet：获得某个指定作业的状态
- AddJobServlet：向当前的子服务器中添加某个作业。
- RegisterSlaveServlet：注册某个服务器的信息。服务器信息包括子服务器是否活动、最新活动的时间、最新不活动的时间。这个在 dynamic cluster 中需要用到，由从属服务器向主服务器汇报当前状态。
- GetSlavesServlet：获得集群中子服务器的信息
- AddExportServlet：以 zip 文件的形式向 caret 服务器传递作业或者转换信息，并将信息加入到服务器中。

2.2 运行转换

2.2.1 在 spoon 中运行

在 kettle 的集成设计环境 spoon 中，你可以选择转换中的“运行”菜单项，或者按 F9 快捷键，弹出以下的窗口



这里有三个选项来决定转换是以什么方式来执行

- **本地执行：**转换或者作业将在你现在使用的 JVM 中运行。
- **远程执行：**允许你指定一个想运行转换的远程服务器。这需要你在远程服务器上安装 Pentaho Data Integration（Kettle）并且运行 Carte 子服务器。
- **集群方式执行：**允许你在集群环境下执行作业或者转换

当你选择“集群方式执行”选项是，你可以选择以下的选项

- **提交转换：**分解转换并且将转换提交到不同的主服务器和从属服务器。
- **准备执行：**它将在主服务器和从属服务器上执行转换的初始化阶段。

- **开始执行:** 它将在主服务器和从属服务器中执行实际的转换任务。

- **显示转换:** 显示将要在集群上执行的生成的转换(可以参看下面的分析).

2.2.2 编程运行

你也可以通过使用 Kettle 提供的 API 通过编程来以集群的方式运行转换。

```
TransMeta transMeta = new TransMeta("cluster.ktr");
```

```
//设置执行模式
```

```
TransExecutionConfiguration config = new TransExecutionConfiguration();

config.setExecutingClustered(true);

config.setExecutingLocally(false);

config.setExecutingRemotely(false);

config.setClusterPosting(true);

config.setClusterPreparing(true);

config.setClusterStarting(true);

config.setLogLevel(LogWriter.LOG_LEVEL_BASIC);

TransSplitter transSplitter = Trans.executeClustered(transMeta, config);
```

```
long nrErrors = Trans.monitorClusteredTransformation("cluster  test",

   transSplitter, null, 1);
```

需要注意的是这段代码可以在一个独立的 JVM 中执行，而不必要在主服务器中执行。

2.2.3 运行内幕

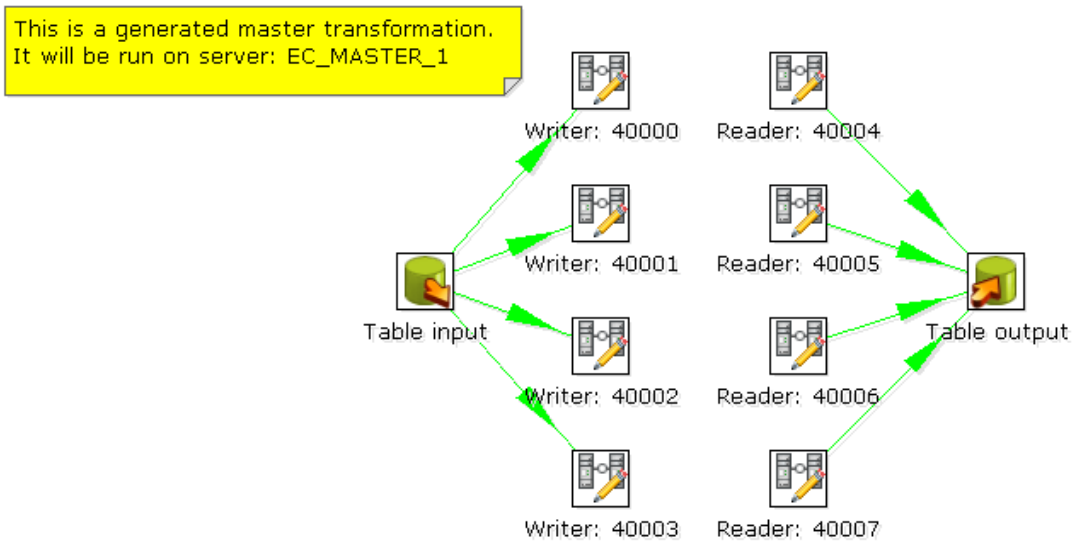
当以集群方式来运行转换时，Kettle 主要执行以下几个步骤来执行分布式的处理

- 分解转换

在定义转换的时候，如果在某个步骤中定义使用集群，那么这个步骤其实是在从属服务器(slave server)上执行的，例如我们在前面定义转换的 modify javascript value 步骤中，我们定义了使用集群，那么这个步骤将在从属服务器中执行；而 Cx4 表示这个步骤是在 4 个从属服务器上执行。如果步骤中没有定义集群，则表示该步骤是在主服务器(master server)上执行。如果前一步骤在主服务器上执行，而后一步骤需要在从属服务器上执行，或者相反，则这时需要分别在前一步骤和后一步骤之间建立一个 remoteStep 步骤，前面的 remoteStep 建立 socketWriter 进程，它负责从上一步骤中取出数据然后通过 socket 传输到对应的子服务器的 remoteStep 中。而后一步骤所在的子服务器的 remoteStep 步骤则建立一个 socketReader,负责从 socket 中获取数据，并将数据将数据传输到后一步骤中，以供后一步骤来进行后续处理。

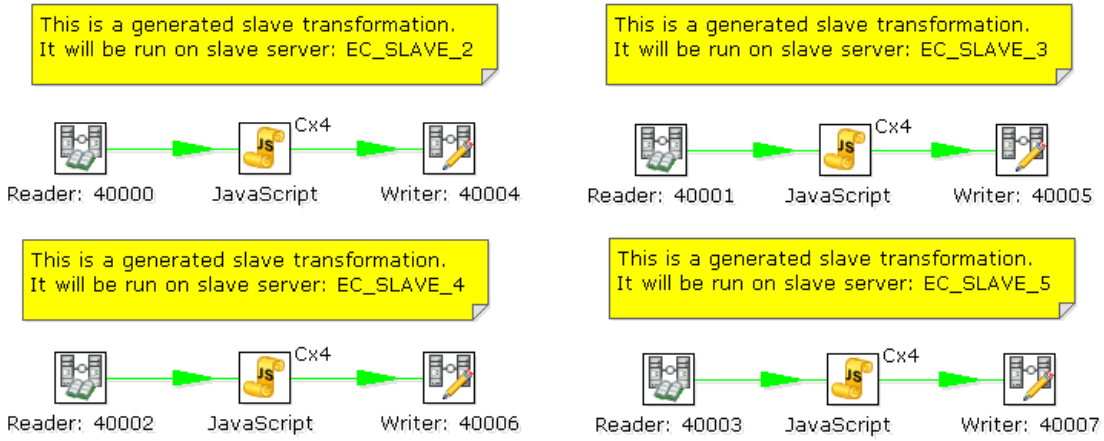
所以在以集群方式执行转换时，首要的任务是将转换分解成可以在各个子服务器上执行的转换。

我们还是以上面建立的转换来进行分析描述：



上图是在主服务器上建立的转换

And 4 slaves transformations:



上图是在 4 个从属服务器上建立的转换，我们可以注意到这四个从属服务器上的转换是一样的，除了端口号不一样。另外我们还注意到在前述 Cluster Schema 定义中我们指定了端口号为 4000，则为每一个建立的 socket 连接就是端口号 4000 开始，依次加 1。另外，还可以看到数据是通过使用 socket Writer 和 socket Reader 的 remoteStep 步骤通过 TCP/IP 的 socket 来传递数据的。

- 提交转换

对于第一步骤生成的子转换，将调用每个子服务器提供的AddTransServlet服务将转换的信息增加到每个子服务器中（包括主服务器和从属服务器）。

- 准备转换

调用每个子服务器的PrepareExecutionTransServlet服务来准备转换

- 启动转换

调用每个子服务器的StartExecutionTransServlet服务来启动转换。

- 监控转换

在各服务器的转换都启动后，调用Trans.*monitorClusteredTransformation*来监控各个服务器的运行状态（使用各子服务器提供的GetTransStatusServlet服务来获得每个子服务器的状态）。

自动分配任务？？？随机分发？

通过 remoteStep 可以看成每个转换被分割成多个，在不同服务器上执行。

Ps： remoteStep 是集群运行中自动做到事，跟我们没关系

例子

目的

做一个转换（表输入---→排序--→表输出）

然后在两台 pc 机器上实验。把集群放到排序插件上。

配置两台子服务器

创建子服务器

在主对象下的转换下的子服务器右键单击新建。

右键单击子服务器新建

填写相关的配置，用户名和密码为 cluster,如果要修改得修改 kettle 默认路径下的 pwd 下的 kettle.pwd 文件里的用户名密码。

这个是从属服务器。

配置 schemas

新建 schemas

在选择子服务器中选择这两个服务器。

开启两台机器的 carte 服务

在 10.2.4.81 机器和 10.2.4.188 机器的控制台开启 carte 服务。

F:\Kettle\pdi-ce-6.1.0.1-196 是保存 kettle 的文件夹

188 机器也和他一样。

在转换中添加集群

右击字段选择选择集群。

点击确定。

出现 cx1 代表成功。

然后运行，就 OK 了。