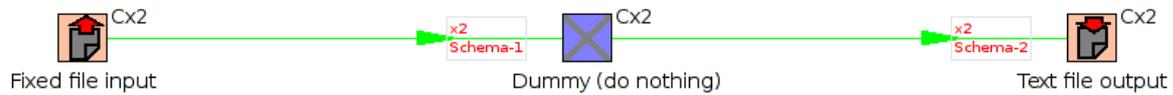
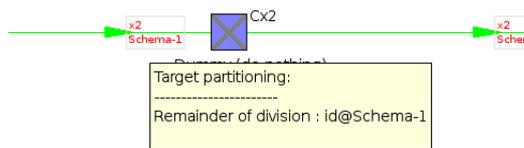


Repartitioning in a clustered environment

The following example reads a file in parallel and then moves the data to the dummy step according to a first partitioning schema specification. (hash on id)



Then the data is repartitioned on another partitioning schema and written to 2 separate text files. If you mouse over the new partitioning hints on the hops you'll see more info on the partitioning used:



At this time repartitioning is only activated between clustered nodes. The other borderline conditions such as re-partitioning from master to slave nodes or vice-versa is not yet active. (I figured this was the most common use-case) If you need that functionality, consider adding a simple clustered dummy step on either end.

The transformation is translated into N transformations for N partitions on N slave servers.

NOTE: At this time, we do NOT support a number of partitions different from the number of slave servers. (MC: It is my intention to re-enable this soon)

This is what the slave transformations look like:



As you can see there are $(N-1)$ extra remote output steps for each step that (re-)partitions. There are obviously also $(N-1)$ extra remote input steps for each step that is partitioned.

Solved (b)locking problem

I had one use-case (the one shown) where I re-partitioned on the same key. (stupid tests are always the best) The problem there was that all data is partitioned and ends up on a single node. Re-partitioning on the same key leads to all data being sent to the always same output RowSet.

Because up until now we read alternating from each input RowSet until we got data, this was causing a head-ache because there would never be a single row on (N-1) input rowsets. This was solved by reading up to 100 rows from a single rowset before switching to another. It switches also after a short timeout. That would re-try the other (N-1) rowsets every 100 rows to limit the overhead of checking after every row.

I hope that this doesn't have a negative impact on performance, but my first tests indicate that it doesn't. This deviates from the standard behaviour in version 2.5, but unlike the round-robin system for the output we never guaranteed the same behaviour on the input. (on the contrary)

Warning: Complex code ahead

As you can imagine, with the virtually limitless possibilities in a Kettle transformation, it is very hard to come up with every possible test scenario in a few days or weeks time. Please let me know use-cases that work for you and especially those that DON'T work as soon as possible so that we can iron out the largest issues.

Suggestions welcome

It can not be stressed hard enough that suggestions are as always very much needed and welcome in any stage of the development. Suggestions and/or code to improve the useability, GUI, anything is always appreciated.

Matt

mcasters@pentaho.org