Processor Concurrency

Conduit allows you to process records concurrently using the processor'sworkers parameter. This can be particularly beneficial for processors performing time-consuming computations on a fast stream of records.

Ordering guarantee When processors are configured to process records concurrently, they guarantee that the order of records won't change. Although records may be processed out of order, the resulting order of records coming out of the processor is guaranteed to be unchanged. Consider a pipeline that needs to process 100 records per second. One of the processors in the pipeline needs to retrieve data from a server, and this operation typically takes 1 second to complete. Since this operation is executed for every record, sequential processing won't keep up with the stream. However, by configuring the processor to use 100 workers, we can execute the slow operation concurrently on 100 records at a time. This enables us to process 100 records per second.

Note that in reality we would want to configure an even higher number of workers to provide some room for handling larger bursts.

Configuration

You can configure a processor to run concurrently by specifying the number of workers. By default, the number of workers is set to 1, which means that records are processed sequentially.

Here is an example pipeline configuration that configures thejs processor to run in parallel using 10 workers:

version :

2.2 pipelines : -

id : example - pipeline connectors :

define source and destination connectors

...

processors : -

id : example - js type : js workers :

10

configure 10 workers that run concurrently

settings : script :

> function process(record) { record.Metadata["foo-key"] = "foo-value"; return record; }Edit this page Previous Referencing Record Fields Next Connector List