SAS® GLOBAL FORUM 2016

IMAGINE. CREATE. INNOVATE.

Working with Big Data in Near Real Time Using SAS® Event Stream Processing

Kishore Konudula

#SASGF

Working with Big Data in Near Real Time Using SAS® Event Stream Processing

Kishore Konudula

Kavi Global

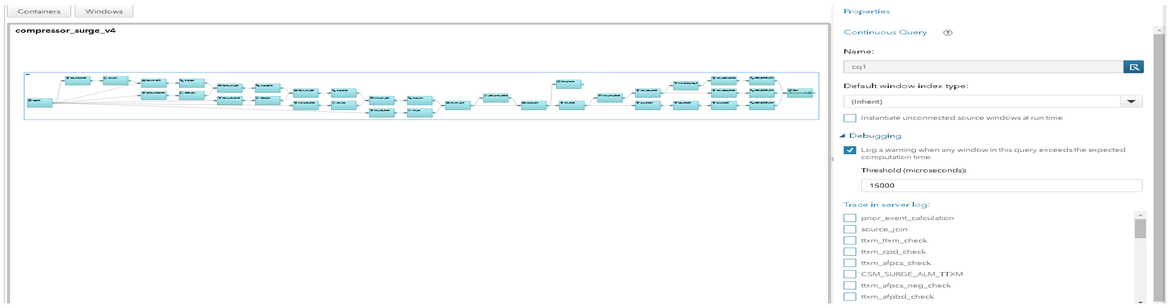
ABSTRACT

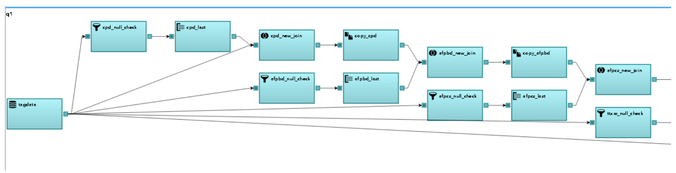
- Data explosion is very real and many organizations are producing data in the order of terabytes as opposed to megabytes a few years ago.
- Data processing technology has not caught up, until now!
- SAS® Event Stream Processing helps in processing huge amounts of data in near real time with low latency.
- SAS® ESP can be used to connect to various state of the art data storage mediums like Hadoop, Cassandra and various message systems.
- The data can be transformed, analyzed and scored instantaneously thanks to DS2 integration out of the box.

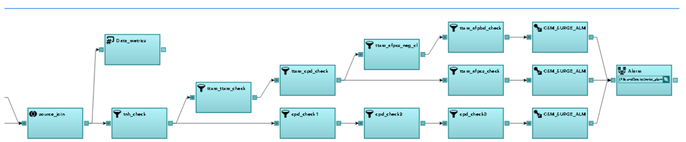
METHODS

- SAS® ESP comes with native connectivity to Big Data sources like HDFS.
- ESP also connects to Apache Camel to be able to talk to other kinds of data sources, in this case Apache Cassandra which is extensively used for TimeSeries data.
- This poster looks at ways to connect to HDFS and Cassandra and also dives into the performance metrics of these ingestion mediums.
- The model contains multiple windows including source window, filter window, aggregate window, join window, copy window, compute window and union window.
- TimeSeries sensor data is stored in HDFS and Cassandra which is ingested into the ESP model using the Source window and then the n-1 and n-2 values of these sensor readings are computed using the *lag* function.
- This is done to be able to detect surge in the sensor equipment.

- SAS® ESP model





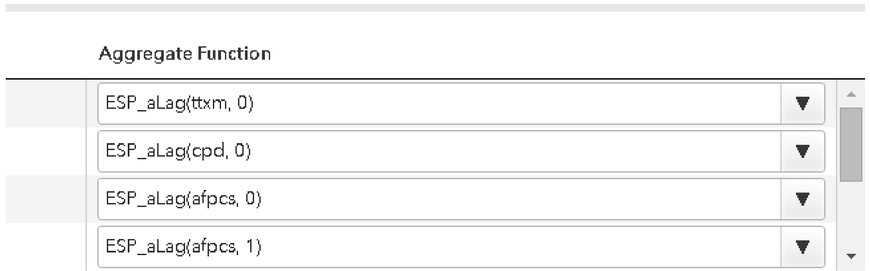


Working with Big Data in Near Real Time Using SAS® Event Stream Processing

Kishore Konudula

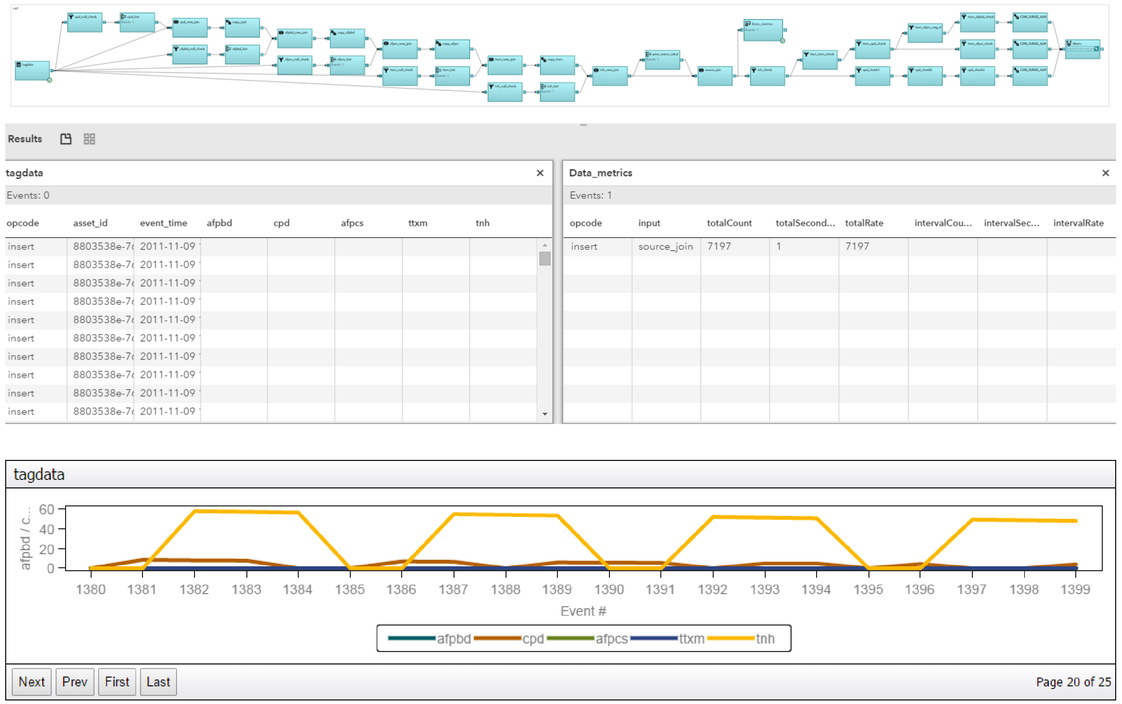
Kavi Global

METHODS CONTINUED



RESULTS

- More than 7000 events were processed in 1 second when connecting to a csv file on the server



Working with Big Data in Near Real Time Using SAS® Event Stream Processing

Kishore Konudula

Kavi Global

RESULTS CONTINUED

- Connecting to HDFS also yields similar performance. However over larger datasets HDFS publisher is far more efficient the file system adapter used in the previous example.

```
[sas@ip-10-202-90-9 bin]$ dfesp_hdfs_publisher -u "dfESP://localhost:41003/compressor_surge_v4/cq1/tagdata" -f "hdfs://10.202.90.248:8020"
```

- Reading data from Cassandra requires routing through Apache Camel. There may be slight latency involved. However reading the same sensor events through Cassandra took around 2 seconds. Most of the additional time was the time it took Camel to connect and push data into the ESP model

Data_metrics

Events: 1

opcode	input	totalCount	totalSecond...	totalRate
insert	source_join	7197	2	3598.5

- The maximum throughput achieved when reading data from Cassandra through Apache Camel was around 9k events per second.

Data_metrics

Events: 1

opcode	input	totalCount	totalSecond...	totalRate
delete	source_join	17986	2	8993
updateblock	source_join	69999	8	8749.88
insert	source_join	17986	2	8993

CONCLUSIONS

- SAS® Event Stream Processing is vast improvement over available tools today to churn through huge amounts of data and sensor events stored in various data sources.
- With improvements in connectivity ESP is reducing the latency.
- The connectivity to HDFS and conventional filesystem is already highly efficient. A native adapter to Cassandra can be developed to reduce the latency introduced by Camel.

Working with Big Data in Near Real Time Using SAS® Event Stream Processing

Kishore Konudula

Kavi Global

APPENDIX

```
<project name="compressor_surge_v4" pubsub="auto" threads="10" use-tagged-token="true">
  <description><![CDATA[testing connection to cassandra and trying to maintain the order]]></description>
  <contqueries>
    <contquery name="cq1" timing-threshold="15000">
      <windows>
        <window-aggregate name="prior_event_calculation" index="pi_HASH" insert-only="true" output-insert-only="false" collapse-updates="true">
          <schema>
            <fields>
              <field name="asset_id" type="string" key="true"/>
              <field name="ttxm_n1" type="double"/>
              <field name="cpd_n1" type="double"/>
              <field name="afpcs_n1" type="double"/>
              <field name="afpcs_n2" type="double"/>
              <field name="afpbd_n1" type="double"/>
              <field name="cpd_n2" type="double"/>
              <field name="tnh_n1" type="double"/>
            </fields>
          </schema>
          <output>
            <field-expr><![CDATA[ESP_aLag(ttxm, 0)]]></field-expr>
            <field-expr><![CDATA[ESP_aLag(cpd, 0)]]></field-expr>
            <field-expr><![CDATA[ESP_aLag(afpcs, 0)]]></field-expr>
            <field-expr><![CDATA[ESP_aLag(afpcs, 1)]]></field-expr>
            <field-expr><![CDATA[ESP_aLag(afpbd, 0)]]></field-expr>
            <field-expr><![CDATA[ESP_aLag(cpd, 1)]]></field-expr>
            <field-expr><![CDATA[ESP_aLag(tnh, 0)]]></field-expr>
          </output>
        </window-aggregate>
        <window-join name="source_join" index="pi_EMPTY" insert-only="false" output-insert-only="true" collapse-updates="true">
          <join type="inner" no-regenerates="true">
            <conditions>
              <fields left="asset_id" right="asset_id"/>
            </conditions>
          </join>
          <output>
            <field-selection name="ttxm" source="l_ttxm"/>
            <field-selection name="afpcs" source="l_afpcs"/>
            <field-selection name="tnh_n1" source="r_tnh_n1"/>
            <field-selection name="afpbd" source="l_afpbd"/>
            <field-selection name="ttxm_n1" source="r_ttxm_n1"/>
            <field-selection name="cpd_n1" source="r_cpd_n1"/>
            <field-selection name="cpd_n2" source="r_cpd_n2"/>
            <field-selection name="afpcs_n1" source="r_afpcs_n1"/>
            <field-selection name="afpcs_n2" source="r_afpcs_n2"/>
```

```
<?xml version="1.0"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:osgi="http://camel.apache.org/schema/osgi"
       xmlns:osgix="http://www.springframework.org/schema/osgi-compendium"
       xmlns:camel="http://camel.apache.org/schema/spring"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/
                           http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel
                           http://camel.apache.org/schema/osgi http://camel.apache.org/schema/osgi/camel-osg
                           http://www.springframework.org/schema/osgi-compendium http://www.springframework.

    <!--          <from uri="direct:start"/>
      <routeBuilder ref="camelCassandraBuilder" />
     <bean id="camelCassandraBuilder" class="com.github.oscerd.camel.cassandra.CamelCassandraRouteBuilder"/>
    -->
    <camelContext id="camel1" xmlns="http://camel.apache.org/schema/spring">

        <endpoint id="csvData" uri="file">
            <property key="fileName" value="C:\test.csv" />
        </endpoint>

        <endpoint id="cassandraPublish" uri="esp://10.202.90.9:41003">
            <property key="project" value="compressor_surge_v4" />
            <property key="contquery" value="cq1" />
            <property key="window" value="tagdata" />
            <property key="blocksize" value="1" />
             <property key="format" value="json" />
        </endpoint>

        <route>
            <from uri="timer:foo?period=600s"/>
            <to uri="readFromCassandra"/>
            <to uri="ref:cassandraPublish"/>
```

```
$ dfesp_fs_adapter -k pub -h "dfESP://localhost:41003/compressor_surge/cq1/tagdata_sample_csv" -f /home/e
```

SAS® GLOBAL FORUM 2016

IMAGINE. CREATE. INNOVATE.

LAS VEGAS | APRIL 18-21

#SASGF