Things to learn

Login to ambari and check jobs and spark UI information.

Check verison of all apps

Read all APIs of dataset

==========================================================================================================================

About spark submit command

Which version of spark you are using 2.4

Which version of kafka you are using kafka\_2.11 0.10.1.1

spark-submit

--master yarn

--deploy-mode cluster

--queue spark\_queue

--class org.dell.saie.analytics.SACSaAgentFileExtractor

--name $job\_name

--files configapplication.yaml,configgeneric.properties

--verbose

--num-executors 9

--executor-memory 24g

--executor-cores 5

--conf spark.driver.memory=8g

--conf spark.yarn.executor.memoryOverhead=4096m

$job\_name-1.3.0.jar application.yaml generic.properties

& ${LOG}

## What kind of hive table used in SAIE project?

Internal table.

How will you create spark session so that we can connect to hive?

thriftdellhadoop2-master1.us.dell.com9083

hiveDynamicPartition true

hiveDynamicPartitionMode nonstrict

SparkSession sparkSession = SparkSession.builder().config(sparkConf).config(hive.metastore.uris, thriftURI)

.config(hive.exec.dynamic.partition, hiveDynamicPartition)

.config(hive.exec.dynamic.partition.mode,hiveDynamicPartitionMode)

.enableHiveSupport()

.getOrCreate();

If you want to create JavaSpark context from spark session then below is the code. Spark context can be used to create accumulator and bradcoast.

JavaSparkContext javaSparkContext = JavaSparkContext.fromSparkContext(sparkSession.sparkContext());

===========================================================================================================================================

Read json data from kafka using spark session

DatasetRow jsonData = sparkSession.read().format(org.apache.spark.sql.kafka010.KafkaSourceProvider)

.option(kafka.bootstrap.servers, brokers) dellhadoop2-edge1.us.dell.com6667,dellhadoop2-edge2.us.dell.com6667

.option(group.id, consumerGroup)

.option(subscribe, kafkaTopic)

.option(startingOffsets, startingOffset) {topic {partition1 offset , partition2 offset, partitionN offset}

.option(endingOffsets, endingOffset)

.load()

.selectExpr(CAST(value as STRING ));

===========================================================================================================================================

How to convert json data set to tabular data set

schemaStringStructType=upload\_mfg\_id upload\_serial\_number upload\_type file\_config\_id file\_status run\_key\_id anti\_virus\_status upload\_asset\_tag business\_process\_type import\_date upload\_case\_number file\_src\_path file\_coll\_id create\_date file\_size\_bytes is\_periodic kafka\_key\_id upload\_asset\_tag\_type client\_version registration\_id file\_type api\_user\_id business\_master\_id asset\_tag sub\_client\_type

Prepare schema from json keys

ListStructiField fields = new ArrayList();

for (String fieldName schemaStringStructType.split( )) {

StructField field = DataTypes.createStructField(fieldName, DataTypes.StringType, true);

fields.add(field);

}

StructType schema = DataTypes.createStructType(fields);

create new data set with schema

DatasetString kafkaMessageDSString = kafkaMessageDS.as(Encoders.STRING());

DatasetRow kafkaMessageDSSchema = sparkSession.read().schema(schema).json(kafkaMessageDSString);

return kafkaMessageDSSchema;

===========================================================================================================================================

Which all DataSet APis are being used in this project

a. dataset.dropDuplicates(run\_key\_id); Here run key id is column name in DataSetRow

b. dataset.repartition(partitionCount); we can repartition the dataset using repartition method

c. dataset.count(partitionCount)

===========================================================================================================================================

How will you run hive query in spark

DatasetRow runSeqNumRawDS = sparkSession.sql(watermarkQuery);

Below are Some of the queries that can be executed using sql method

watermarkQuery = select NVL(run\_seq\_num, 0) from saie\_audit.process\_job where p\_job\_name = 'sac-saagent-etl-file-parser'

INSERT OVERWRITE TABLE saie\_audit.process\_job PARTITION (p\_job\_name) SELECT ' + processJobRunSeqNum + ', 'kafka', ' + kafkaTopic + ', ' + consumerGroup + ', ' + totalLag + ', ' + batchsize + ', ' + latestOffsets + ',' + committedOffsets + ', ' + finalOffsets.toString() + ', 'SUCCESS', pj.created\_date, pj.created\_by, CURRENT\_TIMESTAMP(), 'spark-user', pj.p\_job\_name FROM saie\_audit.process\_job pj WHERE p\_job\_name = ' + jobName + '

boolean isRunSequenceNotEmpty = (runSeqNumRawDS != null && runSeqNumRawDS.count() 0);

String maxNum;

if (isRunSequenceNotEmpty) {

maxNum = runSeqNumRawDS.collectAsList().get(0).get(0).toString();

} else {

maxNum = 0;

}

BigInteger maxRunSeqNum = new BigInteger(maxNum);

return maxRunSeqNum;

===========================================================================================================================================

How will you creare temporary view from data set

kafkaMessageDSSchema1.createOrReplaceTempView(Name of the temp table name);

Once dataset is created as temporary view we can run queries on top of it just like hive tables

===========================================================================================================================================

For debugging purpose we can log some of the details like spark application id, user who is executing spark job, and quer etc using below APIs. All these information is stored at context level.

String sparkAppId = sparkSession.sparkContext().applicationId();

String userName = sparkSession.sparkContext().sparkUser();

===========================================================================================================================================

Which is the main transformation used in this project

flatMap

DatasetCollectionFlattenedData collectionDataDS = kafkaDataSetFromView.flatMap(new ExtractCollectionFilesLambda(filesToBeExtractedArr, xmlFileSep, collectionFileExtentionArr, mountPath, xmlFormattingRegex, sparkAppId, accumulator, userPlusApplicationId), Encoders.bean(CollectionFlattenedData.class));

public class ExtractCollectionFilesLambda implements FlatMapFu vc nctionRow, CollectionFlattenedData

{

public IteratorCollectionFlattenedData call(Row row) throws Exception {

ListCollectionFlattenedData collectionFlattenedDataList = new ArrayList();

Business logic

fileListFromCollectionIterator = collectionFlattenedDataList.iterator();

return fileListFromCollectionIterator;

}

{

CollectionFlattenedData is pojo class which will store data in key value format.

===========================================================================================================================================

We are using persisit method on datasetCollectionFlattenedData why

We faced performance issue becase this data set has data that will be written to 3 tables.

to write to tables we were creating temporary table from this data set and wrote 3 queries to load data into different tables.

So here sql queries are actions so when each action is called flat map transformation was called 3 times as result all tedious operation we were doing was executed thrice. As a result we face big performace issue. means application was taking almost 3 times . So when we analyze the DAG we found that each time flat map and its lambda function was called.

Solution to this issue is persit dataset before creating temp tables and running queries on top of it.

collectionDataDS.persist(getStorageLevel(StorageLevel.MEMORY\_AND\_DISK())); there are many storage level options

===========================================================================================================================================

Spark accumulator is also used in this scenario how it used and why it is used

TO create accumulator we need java spark context

JavaSparkContext javaSparkContext = JavaSparkContext.fromSparkContext(sparkSession.sparkContext());

LongAccumulator accumulator = javaSparkContext.sc().longAccumulator();

This accumulator created in driver code should be serialized to executors

accumulator.add(1); this will be done in each executors.

Long errorCountValue = accumulator.value(); This code will be executed in executor

Why are using in this project

We are processing collections in batches say 1 or 2 lakh .. in that particular batch if we find more than configuratble percent (for example 1 or 2) of collection corrupt due to zip issues. or all files inside collections are empty or unparsable, basically if we can not get even single line of data then we have to fail our job. so that we will come to know what is the issue and rectify it.

so in order to keep count of failed collections in distributed environemnt we are using accumulator

===========================================================================================================================================

How are you connecting to kafka to get last commited offset and latest offset

Properties props = new Properties();

props.put(bootstrap.servers, hosts); comma seperated ips

props.put(group.id, saie-offset-util-group);

props.put(key.deserializer, org.apache.kafka.common.serialization.StringDeserializer);

props.put(value.deserializer, org.apache.kafka.common.serialization.StringDeserializer);

create kafka consumer

KafkaConsumerString, String consumer = new KafkaConsumer(props);

ListPartitionInfo partitions = this.consumer.partitionsFor(topic);

-- committed offset

Iterator var6 = partitions.iterator();

while(var6.hasNext()) {

PartitionInfo partition = (PartitionInfo)var6.next();

OffsetAndMetadata offsetMeta = this.consumer.committed(new TopicPartition(topic, partition.partition()));

Long offset = this.handleNullOffsets(offsetMeta);

offsets.add(new Tuple2(partition.partition(), offset));

}

topicOffsets.put(topic, offsets);

-- Latest offset

this.consumer.assign(topicPartitions);

this.consumer.seekToEnd(topicPartitions);

endOffsets = new HashMap();

Iterator var3 = topicPartitions.iterator();

while(var3.hasNext()) {

TopicPartition tp = (TopicPartition)var3.next();

Long offset = this.consumer.position(tp);

((Map)endOffsets).put(tp, offset);

}

}

return this.consolidatedOffsets((Map)endOffsets);

===========================================================================================================================================

How are we commiting offset Iterated MapInt, Long committedOffset (INT Partition number, Long is offset numbers)

private ListTuple2Integer, Long partitionOffsets = new LinkedList();

put all partition and offset in map

MapTopicPartition, OffsetAndMetadata commitMap = new HashMap();

commitMap.put(new TopicPartition(topicName, PARTITION\_NUMBER ), new OffsetAndMetadata((OFFSET, saie-tool-consumer));

this.consumer.commitSync(commitMap);

===========================================================================================================================================

Sample queries used in this project

loadDDVCollectionDataHQL=INSERT INTO saie\_owner.sac\_saagent\_ddv\_collection\_data PARTITION (p\_import\_date,p\_file\_src\_type) SELECT d.runseqnum\_d AS run\_seq\_num, rundtlseqnum\_d AS run\_dtl\_seq\_num, d.runkeyid\_d AS run\_key\_id, d.mfgid\_d AS mfg\_id, d.assettag\_d AS asset\_tag, d.assettagtype\_d AS asset\_tag\_type, d.importdate\_d AS import\_date, d.filename\_d AS file\_name, d.filecollectiondateutc\_d AS file\_collection\_date\_utc, d.collectiondateutc\_d AS collection\_date\_utc, d.relationtree\_d AS relation\_tree, d.tgtobjectclass\_d AS tgt\_object\_class, d.tgtobjectclassinstance\_d AS tgt\_object\_class\_instance, d.objectclass1\_d AS object\_class\_1, d.objectclass2\_d AS object\_class\_2, d.objectclass3\_d AS object\_class\_3, d.objectclass4\_d AS object\_class\_4, d.objectclass5\_d AS object\_class\_5, d.objectclass6\_d AS object\_class\_6, d.objectclass7\_d AS object\_class\_7, d.objectclass8\_d AS object\_class\_8, d.objectclass9\_d AS object\_class\_9, d.objectclass10\_d AS object\_class\_10, d.attributename\_d AS attribute\_name, d.attributevalue\_d AS attribute\_value, createdate\_d AS create\_date, createby\_d AS create\_by,sparkappid\_d AS spark\_app\_id, pimportdate\_d AS p\_import\_date, d.pfilesrctype\_d AS p\_file\_src\_type FROM sac\_saagent\_ddv\_collection\_data\_temp d WHERE d.recordtype='1'

loadDDVCollectionMetaHQL=INSERT INTO saie\_owner.sac\_saagent\_ddv\_collection\_meta PARTITION (p\_import\_date) SELECT m.runseqnum\_m AS run\_seq\_num, m.filecollid\_m AS file\_coll\_id, m.runkeyid\_m AS run\_key\_id, m.fileconfigid\_m AS file\_config\_id, m.uploadcasenumber\_m AS upload\_case\_number, m.uploadmfgid\_m AS upload\_mfg\_id, m.uploadserialnumber\_m AS upload\_serial\_number, m.uploadassettag\_m AS upload\_asset\_tag, m.uploadassettagtype\_m AS upload\_asset\_tag\_type, m.uploadtype\_m AS upload\_type, m.collectionfilestatus\_m AS collection\_file\_status, m.collectionantivirusstatus\_m AS collection\_anti\_virus\_status, m.collectionfilesrcpath\_m AS collection\_file\_src\_path, m.collectionfilesizebytes\_m AS collection\_file\_size\_bytes, m.collectiondateutc\_m AS collection\_date\_utc, m.filecollectiondateutc\_m AS file\_collection\_date\_utc, m.importdate\_m AS import\_date,m.registrationid\_m AS registration\_id,m.downloadsource\_m AS download\_source, m.filename\_m AS file\_name, m.mfgid\_m AS mfg\_id, m.assettag\_m AS asset\_tag, m.assettagtype\_m AS asset\_tag\_type, m.isperiodic\_m AS is\_periodic, m.isalert\_m AS is\_alert, m.businessprocesstype\_m AS business\_process\_type, m.clientversion\_m AS client\_version, m.clienttype\_m AS client\_type, m.subclienttype\_m AS sub\_client\_type, m.eula\_m AS eula, m.entitlement\_m AS entitlement, m.customerfilepresent\_m AS customer\_file\_present, m.warrantyexpirydate\_m AS warranty\_expiry\_date, m.ddvrevision\_m AS ddv\_revision, m.motherboardeppid\_m AS motherboard\_eppid, m.hardwareinfoguid\_m AS hardware\_info\_guid, m.systemmodel\_m AS system\_model, m.biosversion\_m AS bios\_version, m.processorinformation\_m AS processor\_information, m.videocontroller\_m AS video\_controller, m.operatingsystem\_m AS operating\_system, m.oscaption\_m AS os\_caption, m.oseditionid\_m AS os\_editionid, m.osarchitecture\_m AS os\_architecture, m.osbuildbranch\_m AS os\_build\_branch, m.osreleaseid\_m AS os\_release\_id, m.osversion\_m AS os\_version, m.oscurrmajorvernum\_m AS os\_curr\_major\_ver\_num, m.oscurrminorvernum\_m AS os\_curr\_minor\_ver\_num, m.oscurrversion\_m AS os\_curr\_version, m.oscurrbuildnum\_m AS os\_curr\_build\_num, m.osupdatebuildrevision\_m AS os\_update\_build\_revision, m.lastrestarttime\_m AS last\_restart\_time, m.lastshutdowntime\_m AS last\_shutdown\_time, createdate\_m AS create\_date, createby\_m AS create\_by,sparkappid\_m AS spark\_app\_id, pimportdate\_m AS p\_import\_date FROM sac\_saagent\_ddv\_collection\_data\_temp m LEFT OUTER JOIN saie\_owner.sac\_saagent\_ddv\_collection\_meta meta ON (m.runkeyid\_m = meta.run\_key\_id AND m.pimportdate\_m = meta.p\_import\_date) WHERE meta.run\_key\_id is null AND m.recordtype='2'

loadDDVCollectionErrorHQL=INSERT INTO saie\_audit.process\_job\_detail\_log PARTITION (p\_import\_date,p\_job\_name) SELECT e.runseqnum\_e AS run\_seq\_num,e.srcrunseqnum\_e AS src\_run\_seq\_num, e.srcrundtlseqnum\_e AS src\_run\_dtl\_seq\_num, e.runkeyid\_e AS run\_key\_id, e.importdate\_e AS import\_date, e.filename\_e AS file\_name, e.jobname\_e AS job\_name, e.errcode\_e AS error\_code, e.errmessage\_e AS error\_message, e.collectionsource\_e AS collection\_source, createdate\_e AS create\_date, createby\_e AS create\_by, e.filecollid\_e as file\_coll\_id, e.fileconfigid\_e as file\_config\_id, e.uploadtype\_e as upload\_type, e.uploadcasenumber\_e as upload\_case\_number, e.uploadmfgid\_e as upload\_mfg\_id, e.uploadserialnumber\_e as upload\_serial\_number, e.uploadassettag\_e as upload\_asset\_tag, e.isperiodic\_e as is\_periodic, e.collectionfilestatus\_e as collection\_file\_status, e.collectionantivirusstatus\_e as collection\_anti\_virus\_status, e.collectionfilesrcpath\_e as collection\_file\_src\_path, e.collectionfilesizebytes\_e as collection\_file\_size\_bytes, e.businessprocesstype\_e as business\_process\_type, e.clientversion\_e as client\_version, e.clienttype\_e as client\_type, e.subclienttype\_e as sub\_client\_type, e.entitlement\_e as entitlement, e.ddvrevision\_e as ddv\_revision , e.assettag\_e as asset\_tag, e.errorgroup\_e as error\_group,pimportdate\_e AS p\_import\_date, e.pjobname\_e AS p\_job\_name FROM sac\_saagent\_ddv\_collection\_data\_temp e WHERE e.recordtype='3'

===========================================================================================================================================

How to calculate total lag and calculating endoffsets

public Long getTotalLag(MapInteger, Long committedOffsets, MapInteger, Long latestOffsets) {

return committedOffsets.entrySet().stream().map(e - (latestOffsets.get(e.getKey()) - e.getValue()))

.reduce(0L, (e1, e2) - e1 + e2);

}

===========================================================================================================================================

Things to do later

Download and understand kafka-util code