databricks Join Performances

Disclaimer

The "data" - For clarification, not in the scope of exam

- For this example Marelli environment has been used
- There the output of the reading operation is a custom object with several attributes and methods
- Read rdd are stored inside the “.data” attribute

Tests done with 5VM (1 master + 4 workers) for a total of 70Gb RAM and 20 Core

```
from Mco.Marc.ProcessEngine.sparkProcEng import DatabricksFullProcessEngine
from datetime import datetime, timedelta
from Mco.Marc.Base import STRINGS
from Mco.Marc.Base.abstractTracer import LOG_LEVELS

mco = DatabricksFullProcessEngine("Unimi","Join tests")
```

```
2020-11-30T08:50:40.396320Z - INFO,Unimi,Join tests,Mco.Marc.Base.abstractTrac
er,DefaultTracer.__init__,Log started
2020-11-30T08:50:40.473348Z - INFO,Unimi,Join tests,Mco.Marc.Base.abstractTrac
er,DefaultTracer.__init__,File created
2020-11-30T08:50:40.608467Z - INFO,Unimi,Join tests,Mco.Marc.ProcessEngine.spa
rkProcEng,DatabricksFullProcessEngine.__init__, is ready!
2020-11-30T08:50:40.670725Z - WARNING,Unimi,Join tests,Mco.Marc.ProcessEngine.
sparkProcEng,DatabricksFullProcessEngine.__init__,Using Databricks Spark versi
on with full access to primitives!
```

```
masterLookup = {}
nameLookup = {"POSNR":"Delivery_Item","MANDT":"Client","VBELN":"Delivery_Id",
            "MATNR":"Material_Number",
"WERKS":"Receiving_Plant","LFIMG":"Quantity",
             "MEINS":"Unit_Measure","ORT01":"Customer Address",
              "VBTYV":"Document_Category","VTWEG":"Distribution_Channel",
              "SPART": "Business_Unit", "KDMAT": "Material_Id", "BWART":
"Movement_Type", "MTART": "Mat_Type",
              "ARKTX": "Sales_Order_Text", "VKORG": "Sales_Organization",
"KUNNR": "Customer_Id", "WADAT_IST": "Delivery_Date"}


nameLookupRdd = sc.parallelize([(key,nameLookup.get(key)) for key in
nameLookup])
nameLookupRdd.take(5)
```

```
Out[5]: [('POSNR', 'Delivery_Item'),
 ('MANDT', 'Client'),
 ('VBELN', 'Delivery_Id'),
 ('MATNR', 'Material_Number'),
 ('WERKS', 'Receiving_Plant')]
```

Data Input SAP Tables (SAP is a very popular E

```
LIPSObj = mco.read("/data/raw_data/global/SAP/P52/LIPS", top=10)
LIKPObj = mco.read("/data/raw_data/global/SAP/P52/LIKP", top=10)
KNA1Obj =  mco.read("/data/raw_data/global/SAP/P52/KNA1", top=10)
```

```
2020-11-30T08:55:35.092773Z - INFO,Unimi,Join tests,Mco.Marc.MarcX.reader,Spar
kReader.read,Trying to read the path data/raw_data/global/SAP/P52/LIPS
2020-11-30T08:55:35.366925Z - ERROR,Unimi,Join tests,Mco.Marc.MarcX.reader,Spa
rkReader._read_process,Exception: Trying to read with _read_process: data/proc
essed_data/dev/DatabricksFullProcessEngine/Unimi/Join tests/process.json
2020-11-30T08:55:36.238725Z - INFO,Unimi,Join tests,Mco.Marc.MarcX.reader,Spar
kReader.read,Reading first 10 files
2020-11-30T08:56:36.312768Z - INFO,Unimi,Join tests,Mco.Marc.ProcessEngine.pro
cessEngine,DatabricksFullProcessEngine.read,639980 lines read in 61.2201220989
2273
2020-11-30T08:56:36.440644Z - INFO,Unimi,Join tests,Mco.Marc.MarcX.reader,Spar
kReader.read,Trying to read the path data/raw_data/global/SAP/P52/LIKP
2020-11-30T08:56:36.506135Z - ERROR,Unimi,Join tests,Mco.Marc.MarcX.reader,Spa
rkReader._read_process,Exception: Trying to read with _read_process: data/proc
essed_data/dev/DatabricksFullProcessEngine/Unimi/Join tests/process.json
2020-11-30T08:56:36.831634Z - INFO,Unimi,Join tests,Mco.Marc.MarcX.reader,Spar
kReader.read,Reading first 10 files
2020-11-30T08:57:34.583191Z - INFO,Unimi,Join tests,Mco.Marc.ProcessEngine.pro
```

```
cessEngine,DatabricksFullProcessEngine.read,639980 lines read in 58.1422417163
8489
2020-11-30T08:57:34.637981Z - INFO,Unimi,Join tests,Mco.Marc.MarcX.reader,Spar
kReader.read,Trying to read the path data/raw_data/global/SAP/P52/KNA1
2020-11-30T08:57:34.705452Z - ERROR,Unimi,Join tests,Mco.Marc.MarcX.reader,Spa
rkReader._read_process,Exception: Trying to read with _read_process: data/proc
essed_data/dev/DatabricksFullProcessEngine/Unimi/Join tests/process.json
2020-11-30T08:57:35.047156Z - INFO,Unimi,Join tests,Mco.Marc.MarcX.reader,Spar
kReader.read,Reading first 10 files
2020-11-30T08:57:51.014115Z - INFO,Unimi,Join tests,Mco.Marc.ProcessEngine.pro
cessEngine,DatabricksFullProcessEngine.read,23492 lines read in 16.37534523010
254
```

```
print("LIPSObj:{LIPSObj}, \nLIKPObj:{LIKPObj}, \nKNA1Obj:
{KNA1Obj},\nnameLookupRdd:{nameLookupRdd}".format(LIPSObj=LIPSObj.numelem,

LIKPObj=LIKPObj.numelem,

KNA1Obj=KNA1Obj.numelem,

nameLookupRdd=nameLookupRdd.count()))
```

```
LIPSObj:639980,
LIKPObj:639980,
KNA1Obj:23492,
nameLookupRdd:18
```

```
def keepColumn(row, columnList,ts_key=False,ts_format=False):
  newRow = {}
  for column in columnList:
    newRow[column] = row.get(column)
  if not ts_key:
    ts_key='HammerGW.ts_load'
  if not ts_format:
      ts_format=STRINGS.DATETIME_FORMAT
  newRow["ts"] = datetime.strptime(row[ts_key],ts_format)
  return newRow


def createSnapshot(obj,ts_load=False):
  snapshotObj = obj.copy()
  if not ts_load:
    ts_load = obj.header["ts_key"]
  keyList = snapshotObj.header.get("row_keys")
  getSnapshot = lambda rdd: rdd.map(lambda x: (tuple([x.get(key) for key in
keyList]),(x.get(ts_load),x)))\
            .reduceByKey(lambda x,y: x if x[0]>y[0] else y).map(lambda x: x[1]
[1])
  snapshotObj.update(getSnapshot,"getSnapshot")
  return snapshotObj


columns = ["MANDT", "VKORG", "KUNNR", "WERKS", "WADAT_IST","VBELN","TS"]
keepF = lambda rdd: rdd.map(lambda x: keepColumn(x, columns,
ts_key=False,ts_format=False))
LIKPObj.restore()
LIKPObj.update(keepF,"keepF")

smallLIKPObj = createSnapshot(LIKPObj)
```

```
2020-11-30T08:58:33.403826Z - INFO,Unimi,Join tests,Mco.Marc.CustomTypes.json
L,JsonL.update,[keepF] Number of lines: 639980 in 0.005741040229797364 seconds
2020-11-30T08:58:43.218594Z - INFO,Unimi,Join tests,Mco.Marc.CustomTypes.json
L,JsonL.update,[getSnapshot] Number of lines: 639980 in 0.009451396942138672 s
econds
```

```
columns = ["MANDT", "MATNR", "VTWEG", "SPART","POSNR", "WERKS", "KDMAT",
"LFIMG", "BWART", "MTART", "ARKTX", "VTWEG","MEINS","VBELN","TS","VBTYV"]
keepF = lambda rdd: rdd.map(lambda x: keepColumn(x, columns,
ts_key=False,ts_format=False))
LIPSObj.restore()
LIPSObj.update(keepF,"keepF")
smallLIPSObj = createSnapshot(LIPSObj)
```

```
2020-11-30T08:58:48.842015Z - INFO,Unimi,Join tests,Mco.Marc.CustomTypes.json
L,JsonL.update,[keepF] Number of lines: 639980 in 0.005509872198104859 seconds
2020-11-30T08:58:58.039786Z - INFO,Unimi,Join tests,Mco.Marc.CustomTypes.json
L,JsonL.update,[getSnapshot] Number of lines: 639980 in 0.008786808252334594 s
econds
```

```
columns = ["MANDT","KUNNR",'ORT01',"HammerGW.ts_load"]

keepF = lambda rdd: rdd.map(lambda x: keepColumn(x, columns,
ts_key=False,ts_format=False))
KNA1Obj.restore()
KNA1Obj.update(keepF,"keepF")
smallKNA1Obj = createSnapshot(KNA1Obj,ts_load="HammerGW.ts_load")
```

```
2020-11-30T08:58:59.711679Z - INFO,Unimi,Join tests,Mco.Marc.CustomTypes.json
L,JsonL.update,[keepF] Number of lines: 23492 in 0.0015163912773132325 seconds
2020-11-30T08:59:03.121638Z - INFO,Unimi,Join tests,Mco.Marc.CustomTypes.json
L,JsonL.update,[getSnapshot] Number of lines: 23473 in 0.003033855438232422 se
conds
```

```
smallLIPSrdd = smallLIPSObj.data
smallLIKPrdd = smallLIKPObj.data
smallKNA1rdd = smallKNA1Obj.data
```

First Test: Two Big Rdd

```
smallLIPSrdd.first()
```

```
Out[12]: {'MANDT': '050',
 'MATNR': '00519614100',
 'VTWEG': ' ',
 'SPART': ' ',
 'POSNR': '000649',
 'WERKS': 'PI36',
 'KDMAT': ' ',
 'LFIMG': 70.0,
 'BWART': '101',
 'MTART': 'ROH',
 'ARKTX': '520 - DISTANZIALE BARRA STABILIZZATRICE',
 'MEINS': 'PZ',
 'VBELN': '0180033789',
 'TS': '20190509181619',
 'VBTYV': ' ',
 'ts': datetime.datetime(2019, 5, 21, 3, 17, 51, 458000)}
```

Prepare Join - No Action

```
keyList = ["MANDT", "VBELN"]
plainSmallLIPSrdd = smallLIPSrdd.map(lambda x: (tuple([x.get(key) for key in
keyList]),x))
plainSmallLIKPrdd = smallLIKPrdd.map(lambda x: (tuple([x.get(key) for key in
keyList]),x))
```

```
plainSmallLIKPrdd.first()
```

```
Out[16]: (('050', '0081748519'),
 {'MANDT': '050',
  'VKORG': 'SI35',
  'KUNNR': '0000603004',
  'WERKS': ' ',
  'WADAT_IST': '20170825',
  'VBELN': '0081748519',
  'TS': '20170825053415',
  'ts': datetime.datetime(2019, 6, 4, 13, 5, 36, 157000)})
```

Direct Join - Baseline

```
plainSmallLIPSrdd.join(plainSmallLIKPrdd).count()
```

```
Out[17]: 223690
```

Lets see if Spark Cache something

```
plainSmallLIPSrdd.join(plainSmallLIKPrdd).count()
```

```
Out[18]: 223690
```

```
joined = plainSmallLIPSrdd.join(plainSmallLIKPrdd).persist()
joined.count()
```

```
Out[19]: 223690
```

```
joined.count()
```

```
Out[20]: 223690
```

Repartition

```
distributedSmallLIPSrdd = plainSmallLIPSrdd.repartition(100).persist()
distributedSmallLIKPrdd = plainSmallLIKPrdd.repartition(100).persist()
distributedSmallLIPSrdd.count(), distributedSmallLIKPrdd.count()
```

```
Out[21]: (639980, 639980)
```

Repartitioned Join

```
distributedSmallLIPSrdd.join(distributedSmallLIKPrdd).count()
```

```
Out[22]: 223690
```

Repartition + Repartitioned Join

```
distributedSmallLIPSrdd = plainSmallLIPSrdd.repartition(100).persist()
distributedSmallLIKPrdd = plainSmallLIKPrdd.repartition(100).persist()
distributedSmallLIPSrdd.count(), distributedSmallLIKPrdd.count()
distributedSmallLIPSrdd.join(distributedSmallLIKPrdd).count()
```

```
Out[14]: 223690
```

Collect as Map

```
distributedSmallLIPSmap = distributedSmallLIPSrdd.collectAsMap()
```

```
for key in distributedSmallLIPSmap.keys():
  break
key,distributedSmallLIPSmap.get(key), len(distributedSmallLIPSmap) #145.564
keys
```

```
Out[32]: (('050', '0082195107'),
 {'MANDT': '050',
  'MATNR': '00521138980',
  'VTWEG': 'OE',
  'SPART': 'CS',
  'POSNR': '000660',
  'WERKS': 'PI35',
  'KDMAT': '00521138980',
  'LFIMG': 21.0,
  'BWART': '631',
  'MTART': 'FERT',
  'ARKTX': '520 SEMIC ANT SX',
  'MEINS': 'PZ',
  'VBELN': '0082195107',
  'TS': '20190408092745',
  'VBTYV': ' ',
  'ts': datetime.datetime(2019, 5, 21, 3, 22, 11, 177000)},
 145564)
```

Get

```
distributedSmallLIKPrdd.map(lambda x:
(distributedSmallLIPSmap.get(x[0]),x[1])).count()
```

```
Out[16]: 639980
```

Collect As Map + Get

```
distributedSmallLIPSmap = distributedSmallLIPSrdd.collectAsMap()
distributedSmallLIKPrdd.map(lambda x:
(distributedSmallLIPSmap.get(x[0]),x[1])).count()
```

```
Out[17]: 639980
```

Broadcast Map

```
distributedSmallLIPSbroadcastMap = sc.broadcast(distributedSmallLIPSmap)
```

Broadcasted Map

```
distributedSmallLIKPrdd.map(lambda x:
(distributedSmallLIPSbroadcastMap.value.get(x[0]),x[1])).count()
```

```
Out[34]: 639980
```

Collect + Broadcast + Broadcasted Get

```
distributedSmallLIPSbroadcastMap =
sc.broadcast(distributedSmallLIPSrdd.collectAsMap())
distributedSmallLIKPrdd.map(lambda x:
(distributedSmallLIPSbroadcastMap.value.get(x[0]),x[1])).count()
```

```
Out[35]: 639980
```

Second Test: One Big Rdd and a small one

Prepare Join - no Action

```
keyList = ["MANDT","KUNNR"]
plainSmallKNA1rdd = smallKNA1rdd.map(lambda x: (tuple([x.get(key) for key in
keyList]),x)) #23K lines
plainSmallLIKPrdd = smallLIKPrdd.map(lambda x: (tuple([x.get(key) for key in
keyList]),x)) #149K lines
```

Baseline - Direct Join

```
plainSmallKNA1rdd.join(plainSmallLIKPrdd).count()
```

```
Out[38]: 625568
```

Lets try to reverse the join order

```
plainSmallLIKPrdd.join(plainSmallKNA1rdd).count()
```

```
Out[39]: 625568
```

Repartition

```
distributedSmallKNA1rdd = plainSmallKNA1rdd.repartition(100).persist()
distributedSmallLIKPrdd = plainSmallLIKPrdd.repartition(100).persist()
distributedSmallKNA1rdd.count(), distributedSmallLIKPrdd.count()
```

```
Out[40]: (23473, 639980)
```

Repartitioned Join

```
distributedSmallKNA1rdd.join(distributedSmallLIKPrdd).count()
```

```
Out[41]: 625568
```

Repartition+Join

```
distributedSmallKNA1rdd = plainSmallKNA1rdd.repartition(100).persist()
distributedSmallLIKPrdd = plainSmallLIKPrdd.repartition(100).persist()
distributedSmallKNA1rdd.count(), distributedSmallLIKPrdd.count()
distributedSmallKNA1rdd.join(distributedSmallLIKPrdd).count()
```

```
Out[26]: 625568
```

Collect As Map

```
distributedSmallKNA1map = distributedSmallKNA1rdd.collectAsMap()
```

Get instead of Join

```
distributedSmallLIKPrdd.map(lambda x:
(distributedSmallKNA1map.get(x[0]),x[1])).count()
```

```
Out[43]: 639980
```

Collect As Map + Get

```
distributedSmallKNA1map = distributedSmallKNA1rdd.collectAsMap()
distributedSmallLIKPrdd.map(lambda x:
(distributedSmallKNA1map.get(x[0]),x[1])).count()
```

```
Out[44]: 639980
```

Broadcast Map

```
distributedSmallKNA1SbroadcastMap = sc.broadcast(distributedSmallKNA1map)
```

Broadcasted Get

```
distributedSmallLIKPrdd.map(lambda x:
(distributedSmallKNA1SbroadcastMap.value.get(x[0]),x[1])).count()
```

```
Out[46]: 639980
```

Map + Distribute + Boradcasted Get

```
distributedSmallKNA1SbroadcastMap =
sc.broadcast(plainSmallKNA1rdd.collectAsMap())
distributedSmallLIKPrdd.map(lambda x:
(distributedSmallKNA1SbroadcastMap.value.get(x[0]),x[1])).count()
#5.82 sec to 0.74 sec ==> 1/3 reduction 1K€ per month ==> 300€
```

```
Out[48]: 639980
```

Third test: Big Registry

Read Data

```
bigLIPSObj = mco.read("/data/raw_data/global/SAP/P52/LIPS",top=40) #2.559.920
lines
bigLIKPObj = mco.read("/data/raw_data/global/SAP/P52/LIKP",top=40) #776.015
```

```
2020-11-30T05:43:27.487010Z - INFO,Unimi,Join tests,Mco.Marc.MarcX.reader,Spar
kReader.read,Trying to read the path data/raw_data/global/SAP/P52/LIPS
2020-11-30T05:43:27.690293Z - ERROR,Unimi,Join tests,Mco.Marc.MarcX.reader,Spa
rkReader._read_process,Exception: Trying to read with _read_process: data/proc
essed_data/dev/DatabricksFullProcessEngine/Unimi/Join tests/process.json
2020-11-30T05:43:29.002994Z - INFO,Unimi,Join tests,Mco.Marc.MarcX.reader,Spar
kReader.read,Reading first 40 files
2020-11-30T05:46:10.397813Z - INFO,Unimi,Join tests,Mco.Marc.ProcessEngine.pro
cessEngine,DatabricksFullProcessEngine.read,2559920 lines read in 162.91100716
59088
2020-11-30T05:46:10.558833Z - INFO,Unimi,Join tests,Mco.Marc.MarcX.reader,Spar
kReader.read,Trying to read the path data/raw_data/global/SAP/P52/LIKP
2020-11-30T05:46:10.625282Z - ERROR,Unimi,Join tests,Mco.Marc.MarcX.reader,Spa
rkReader._read_process,Exception: Trying to read with _read_process: data/proc
essed_data/dev/DatabricksFullProcessEngine/Unimi/Join tests/process.json
2020-11-30T05:46:11.042704Z - INFO,Unimi,Join tests,Mco.Marc.MarcX.reader,Spar
kReader.read,Reading first 40 files
2020-11-30T05:47:25.378108Z - INFO,Unimi,Join tests,Mco.Marc.ProcessEngine.pro
cessEngine,DatabricksFullProcessEngine.read,776015 lines read in 74.8196990489
9597
```

Clean Useless Columns

```
columns = ["MANDT", "VKORG", "KUNNR", "WERKS", "WADAT_IST","VBELN","TS"]
keepF = lambda rdd: rdd.map(lambda x: keepColumn(x, columns,
ts_key=False,ts_format=False))
LIKPObj.restore()
LIKPObj.update(keepF,"keepF")

smallBigLIKPObj = createSnapshot(bigLIKPObj)

columns = ["MANDT", "MATNR", "VTWEG", "SPART","POSNR", "WERKS", "KDMAT",
"LFIMG", "BWART", "MTART", "ARKTX", "VTWEG","MEINS","VBELN","TS","VBTYV"]
keepF = lambda rdd: rdd.map(lambda x: keepColumn(x, columns,
ts_key=False,ts_format=False))
LIPSObj.restore()
LIPSObj.update(keepF,"keepF")
smallBigLIPSObj = createSnapshot(bigLIPSObj)
```

```
2020-11-30T05:47:29.467191Z - INFO,Unimi,Join tests,Mco.Marc.CustomTypes.json
L,JsonL.update,[keepF] Number of lines: 639980 in 0.003922449588775635 seconds
2020-11-30T05:48:10.379028Z - INFO,Unimi,Join tests,Mco.Marc.CustomTypes.json
```

```
L,JsonL.update,[getSnapshot] Number of lines: 772072 in 0.04051255869865417 se
conds
2020-11-30T05:48:14.787247Z - INFO,Unimi,Join tests,Mco.Marc.CustomTypes.json
L,JsonL.update,[keepF] Number of lines: 639980 in 0.004343063354492188 seconds
2020-11-30T05:49:47.509455Z - INFO,Unimi,Join tests,Mco.Marc.CustomTypes.json
L,JsonL.update,[getSnapshot] Number of lines: 2559920 in 0.09234606504440307 s
econds
```

Prepare Join and Make Join - Baseline

```
keyList = ["MANDT", "VBELN"]
plainBigLIPSrdd = smallBigLIPSObj.data.map(lambda x: (tuple([x.get(key) for key
in keyList]),x))
plainBigLIKPrdd = smallBigLIKPObj.data.map(lambda x: (tuple([x.get(key) for key
in keyList]),x))
plainBigLIPSrdd.join(plainBigLIKPrdd).count()
```

```
Out[41]: 2405262
```

Repartition

```
distributedSmallBigLIPSrdd = plainBigLIPSrdd.repartition(100).persist()
distributedSmallBigLIKPrdd = plainBigLIKPrdd.repartition(100).persist()
distributedSmallBigLIPSrdd.count(), distributedSmallBigLIKPrdd.count()
```

```
Out[42]: (2559920, 772072)
```

Join Repartitioned

```
distributedSmallBigLIPSrdd.join(distributedSmallBigLIKPrdd).count()
```

```
Out[43]: 2405262
```

Reversed Join Repartitioned

```
distributedSmallBigLIKPrdd.join(distributedSmallBigLIPSrdd).count()
```

```
Out[44]: 2405262
```

Collect As Map

```
distributedSmallBigLIPSmap = distributedSmallBigLIPSrdd.collectAsMap()
```

```
org.apache.spark.SparkException: Job aborted due to stage failure: Total siz
e of serialized results of 41 tasks (4.0 GiB) is bigger than spark.driver.ma
xResultSize 4.0 GiB.
```

Small Registry Collect As Map

```
distributedSmallBigLIKPmap = distributedSmallBigLIKPrdd.collectAsMap()
```

```
ConnectException: Connection refused (Connection refused)
```

Get

```
plainBigLIPSrdd.map(lambda x:
(distributedSmallBigLIKPmap.get(x[0]),x[1])).count()
```

```
NameError: name 'plainBigLIPSrdd' is not defined
```

...

```
distributedSmallBigLIKPbroadcast = sc.broadcast(distributedSmallBigLIKPmap)
```

```
NameError: name 'distributedSmallBigLIKPmap' is not defined
```

...

```
NameError: name 'plainBigLIPSrdd' is not defined
```