Ceph RGW Cache Prefetching for Batch Jobs

Xun Lin
Yang Qiao
Tianyi Tang
Gang Wei
Zhangyu Wan

Project Overview

- Goal:
  - Improve the runtime of Spark batch jobs by prefetching data from Ceph RGW cache
- MVP:
  - Extract DAG out of Spark/Hive Applications
  - Find the job dependency path with maximum runtime reduction
  - Prefetch data from Ceph RGW

Progress

- Problems in the last Demo:
  - DAG not understandable
  - Unable to connect Spark to Ceph through S3a

Progress

DAG:

- Extract DAG out of HiBench Spark benchmarks
- Draw the dependencies from the DAG

Kariz(New):

- Convert DAG string into Kariz graph class

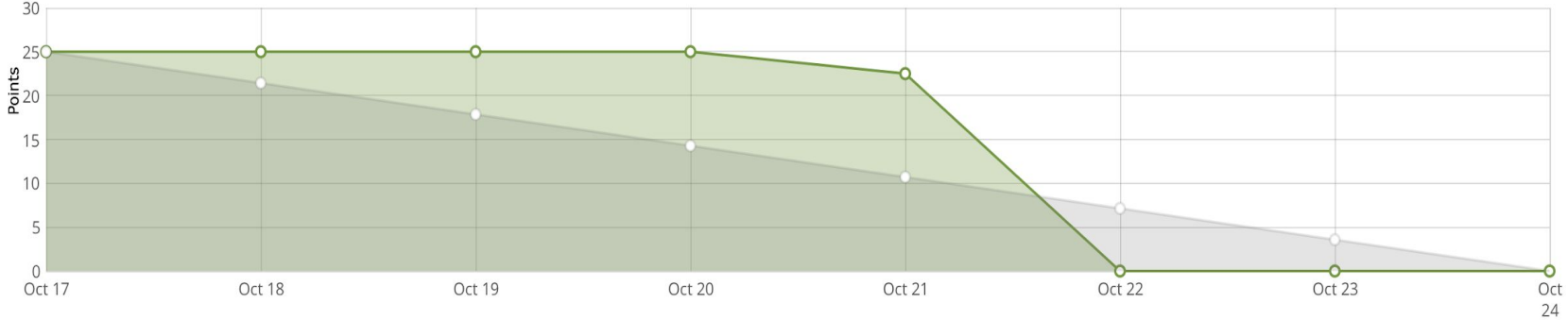
Infrastructure:

- Spark can access Ceph through S3a
- Spark applications can work on multiple VMs

Progress

DEMO 3 BU-CEPH RGW CACHE PREFETCHIN... 17 OCT 2019-24 OCT 2019

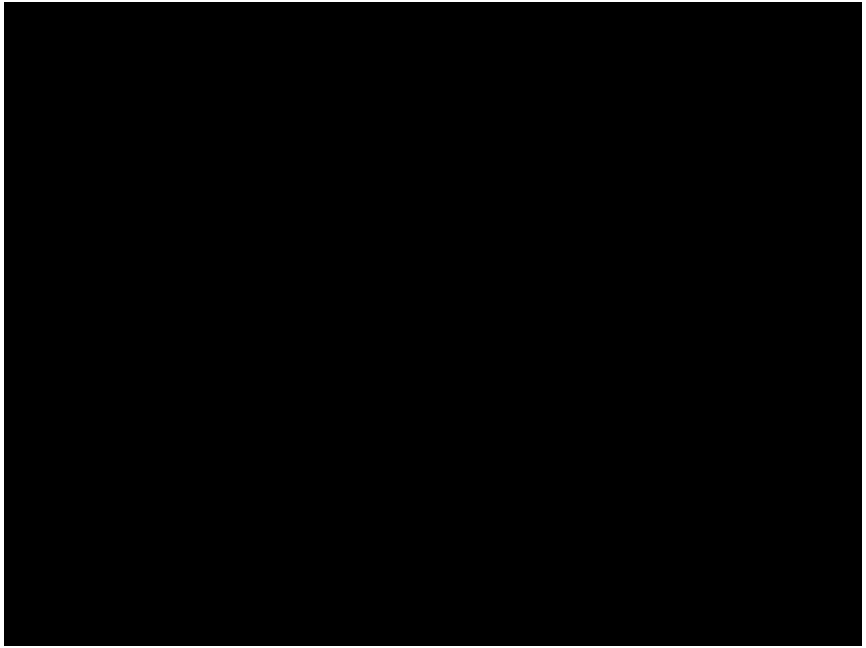
100% 25 total points 25 completed points 0 open tasks 6 closed tasks 0 iocaine doses



HiBench: Spark Benchmarks

- Consists of a set of Hadoop/Spark programs that help evaluate the big data framework
- Micro Benchmarks: Sort, WordCount, TeraSort, Sleep, etc
- Version Dependency:
  - Spark 1.6
  - Scala 2.0

Demo: Spark DAG Generation with HiBench



DAG

Extract Spark RDD DAG in different stages of MapReduce

DAG.txt UNREGISTERED

```
(8) MapPartitionsRDD[2] at map at IOCommon.scala:44 []
 |  MapPartitionsRDD[1] at sequenceFile at IOCommon.scala:44 []
 |  hdfs://sandbox.hortonworks.com:8020/HiBench/Wordcount/Input HadoopRDD[0] at sequenceFile at IOCommon.scala:44 []
(8) MapPartitionsRDD[3] at flatMap at ScalaWordCount.scala:44 []
 |  MapPartitionsRDD[2] at map at IOCommon.scala:44 []
 |  MapPartitionsRDD[1] at sequenceFile at IOCommon.scala:44 []
 |  hdfs://sandbox.hortonworks.com:8020/HiBench/Wordcount/Input HadoopRDD[0] at sequenceFile at IOCommon.scala:44 []
(8) MapPartitionsRDD[4] at map at ScalaWordCount.scala:46 []
 |  MapPartitionsRDD[3] at flatMap at ScalaWordCount.scala:44 []
 |  MapPartitionsRDD[2] at map at IOCommon.scala:44 []
 |  MapPartitionsRDD[1] at sequenceFile at IOCommon.scala:44 []
 |  hdfs://sandbox.hortonworks.com:8020/HiBench/Wordcount/Input HadoopRDD[0] at sequenceFile at IOCommon.scala:44 []
(8) ShuffledRDD[5] at reduceByKey at ScalaWordCount.scala:48 []
 +-(8) MapPartitionsRDD[4] at map at ScalaWordCount.scala:46 []
    |  MapPartitionsRDD[3] at flatMap at ScalaWordCount.scala:44 []
    |  MapPartitionsRDD[2] at map at IOCommon.scala:44 []
    |  MapPartitionsRDD[1] at sequenceFile at IOCommon.scala:44 []
    |  hdfs://sandbox.hortonworks.com:8020/HiBench/Wordcount/Input HadoopRDD[0] at sequenceFile at IOCommon.scala:44 []
```

Dependency Path Graph


```
['sequenceFile', 'sequenceFile', 'map', 'flatMap', 'map', 'reduceByKey', 'sequenceFile', 'sequenceFile',
'map', 'flatMap', 'map', 'sequenceFile', 'sequenceFile', 'map', 'flatMap', 'sequenceFile', 'sequenceFile',
'map']

{0: {'output': 'HadoopRDD[0]', 'inputs': ['hdfs://sandbox.hortonworks.com:8020/HiBench/Wordcount/Input']}, 1:
{'output': 'MapPartitionsRDD[1]', 'inputs': ['', 'HadoopRDD[0]']}, 2: {'output': 'MapPartitionsRDD[2]',
'inputs': ['', 'MapPartitionsRDD[1]']}, 3: {'output': 'MapPartitionsRDD[3]', 'inputs': ['',
'MapPartitionsRDD[2]']}, 4: {'output': 'MapPartitionsRDD[4]', 'inputs': ['', 'MapPartitionsRDD[3]']}, 5:
{'output': 'ShuffledRDD[5]', 'inputs': ['', 'MapPartitionsRDD[4]']}, 6: {'output': 'HadoopRDD[0]', 'inputs':
['hdfs://sandbox.hortonworks.com:8020/HiBench/Wordcount/Input', 'ShuffledRDD[5]']}, 7: {'output':
'MapPartitionsRDD[1]', 'inputs': ['', 'HadoopRDD[0]']}, 8: {'output': 'MapPartitionsRDD[2]', 'inputs': ['',
'MapPartitionsRDD[1]']}, 9: {'output': 'MapPartitionsRDD[3]', 'inputs': ['', 'MapPartitionsRDD[2]']}, 10:
{'output': 'MapPartitionsRDD[4]', 'inputs': ['', 'MapPartitionsRDD[3]']}, 11: {'output': 'HadoopRDD[0]',
'inputs': ['hdfs://sandbox.hortonworks.com:8020/HiBench/Wordcount/Input', 'MapPartitionsRDD[4]']}, 12:
{'output': 'MapPartitionsRDD[1]', 'inputs': ['', 'HadoopRDD[0]']}, 13: {'output': 'MapPartitionsRDD[2]',
'inputs': ['', 'MapPartitionsRDD[1]']}, 14: {'output': 'MapPartitionsRDD[3]', 'inputs': ['',
'MapPartitionsRDD[2]']}, 15: {'output': 'HadoopRDD[0]', 'inputs': ['hdfs://sandbox.hortonworks.com:8020/HiBench
/Wordcount/Input', 'MapPartitionsRDD[3]']}, 16: {'output': 'MapPartitionsRDD[1]', 'inputs': ['', 'HadoopRDD[
0]']}, 17: {'output': 'MapPartitionsRDD[2]', 'inputs': ['', 'MapPartitionsRDD[1]']}}
```

Kariz

- A distributed cache management project by our mentor
- Currently only support PIG

- Improves end-to-end execution time of DAG by informed caching and prefetching
- Prefetches inputs that reduce the runtime the most.

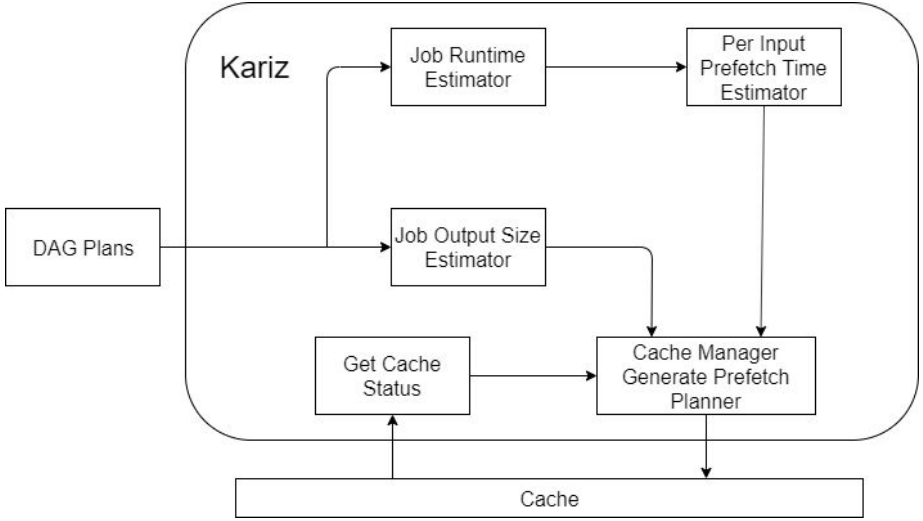
Kariz

What we are currently working on:

- Convert DAG string into Kariz input format (connect Spark with Kariz)
- Adapt PIG DAG planner into Spark DAG planner

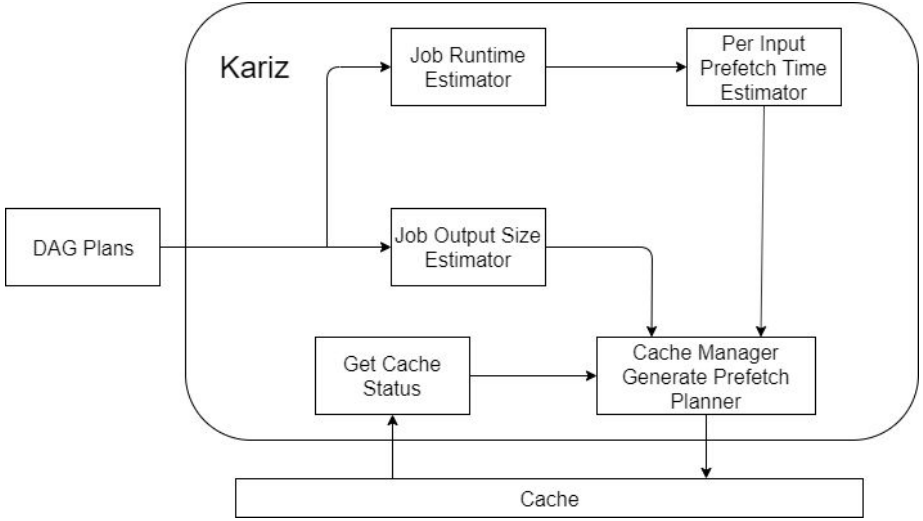
Kariz DAG Planner

- Predict the runtime for each job
- Plan shows input files and their size
- Identify the cache blocks needs to be in the cache.



Kariz Cache Planner

- Prefetch plans: plans of the future stages that should be started to prefetch now to be satisfied by the start of their stage.
- Cache plans: all plans of the current stage.
- Sort the cache plans and prefetch plans based on the fraction of their improvement to DAG runtime to their size

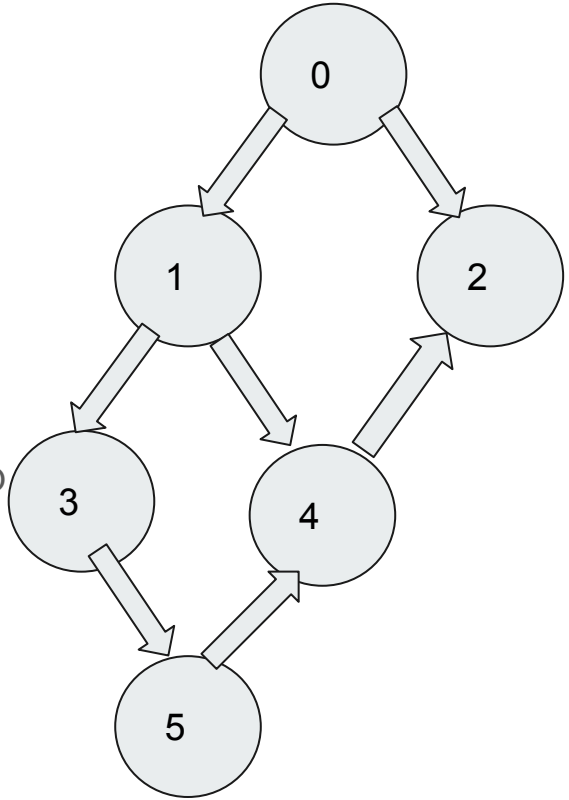


example

In spark

Node: functions

Edges: outputs RDD



Feature: Won't process until all of its dependencies are DONE.

What is the order?

scala> val wordCount = sc.textFile("README.md").flatMap(_.split("\\s+")).map((_, 1)).reduceByKey(_ + _)

wordCount.toDebugString:

```
(8) ShuffledRDD[5] at reduceByKey at ScalaWordCount.scala:48 []
 +-(8) MapPartitionsRDD[4] at map at ScalaWordCount.scala:46 []
    |  MapPartitionsRDD[3] at flatMap at ScalaWordCount.scala:44 []
    |  MapPartitionsRDD[2] at map at IOCommon.scala:44 []
    |  MapPartitionsRDD[1] at sequenceFile at IOCommon.scala:44 []
    |  hdfs://sandbox.hortonworks.com:8020/HiBench/Wordcount/Input HadoopRDD[0] at sequenceFile at
    IOCommon.scala:44 []
```

Kariz DAG Planner

Which is the one reduces the runtime the most?

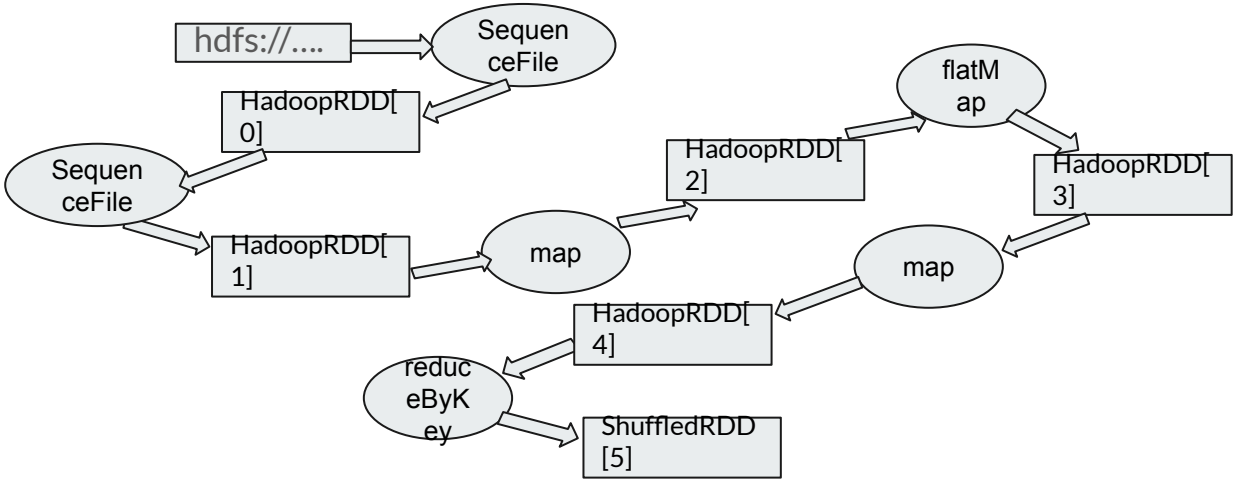
Algorithm: find the longest path in the DAG graph (Dijkstra)

“_”

Graph notation

Function list:
['sequenceFile', 'sequenceFile', 'map', 'flatMap', 'map', 'reduceByKey']

Graph (key is function ID):
{0: {'inputs': ['hdfs://sandbox.hortonworks.com:8020/HiBench/Wordcount/Input'], 'output': 'HadoopRDD[0]'}, 1: {'inputs': ['HadoopRDD[0]'], 'output': 'MapPartitionsRDD[1]'}, 2: {'inputs': ['MapPartitionsRDD[1]'], 'output': 'MapPartitionsRDD[2]'}, 3: {'inputs': ['MapPartitionsRDD[2]'], 'output': 'MapPartitionsRDD[3]'}, 4: {'inputs': ['MapPartitionsRDD[3]'], 'output': 'MapPartitionsRDD[4]'}, 5: {'inputs': ['MapPartitionsRDD[4]'], 'output': 'ShuffledRDD[5]'}}



Demo: Spark with Ceph

Demo: Spark with Ceph (radosgw.8000.log)

wang — centos@ceph-machine:~/ceph/build/out — ssh 128.31.27.218 -i ./cloud-ceph-rgw -l centos -A — 106×30

```
2019-10-23 17:21:36.314 7f5556ff0700 20 CONTENT_TYPE=application/x-www-form-urlencoded; charset=utf-8
2019-10-23 17:21:36.314 7f5556ff0700 20 HTTP_AUTHORIZATION=AWS 0555b35654ad1656d804:OkJhAra4UPQshL2VfXxnOe
IL1vA=
2019-10-23 17:21:36.314 7f5556ff0700 20 HTTP_CONNECTION=Keep-Alive
2019-10-23 17:21:36.314 7f5556ff0700 20 HTTP_DATE=Wed, 23 Oct 2019 17:21:36 GMT
2019-10-23 17:21:36.314 7f5556ff0700 20 HTTP_HOST=127.0.0.1:8000
2019-10-23 17:21:36.314 7f5556ff0700 20 HTTP_USER_AGENT=aws-sdk-java/1.7.4 Linux/3.10.0-957.5.1.el7.x86_64
 OpenJDK_64-Bit_Server_VM/25.201-b09/1.8.0_201
2019-10-23 17:21:36.314 7f5556ff0700 20 HTTP_VERSION=1.1
2019-10-23 17:21:36.314 7f5556ff0700 20 REMOTE_ADDR=127.0.0.1
2019-10-23 17:21:36.314 7f5556ff0700 20 REQUEST_METHOD=HEAD
2019-10-23 17:21:36.314 7f5556ff0700 20 REQUEST_URI=/hadoop1/
2019-10-23 17:21:36.314 7f5556ff0700 20 SCRIPT_URI=/hadoop1/
2019-10-23 17:21:36.314 7f5556ff0700 20 SERVER_PORT=8000
2019-10-23 17:21:36.314 7f5556ff0700  1 ====== starting new request req=0x7f5556febdd0 =====
2019-10-23 17:21:36.314 7f5556ff0700  2 req 6 0.000s initializing for trans_id = tx00000000000000000006-0
05db08c20-102b-default
2019-10-23 17:21:36.314 7f5556ff0700 10 rgw api priority: s3=6 s3website=5
2019-10-23 17:21:36.314 7f5556ff0700 10 host=127.0.0.1
2019-10-23 17:21:36.314 7f5556ff0700 20 subdomain= domain= in_hosted_domain=0 in_hosted_domain_s3website=0
2019-10-23 17:21:36.314 7f5556ff0700 20 final domain/bucket subdomain= domain= in_hosted_domain=0 in_hoste
d_domain_s3website=0 s->info.domain= s->info.request_uri=/hadoop1/
2019-10-23 17:21:36.314 7f5556ff0700 20 get_handler handler=25RGWHandler_REST_Bucket_S3
2019-10-23 17:21:36.314 7f5556ff0700 10 handler=25RGWHandler_REST_Bucket_S3
2019-10-23 17:21:36.314 7f5556ff0700  2 req 6 0.000s getting op 3
2019-10-23 17:21:36.314 7f5556ff0700 10 op=25RGWStatBucket_ObjStore_S3
2019-10-23 17:21:36.314 7f5556ff0700  2 req 6 0.000s s3:stat_bucket verifying requester
2019-10-23 17:21:36.314 7f5556ff0700 20 req 6 0.000s s3:stat_bucket rgw::auth::StrategyRegistry::s3_main_s
trategy_t: trying rgw::auth::s3::AWSAuthStrategy
2019-10-23 17:21:36.314 7f5556ff0700 20 req 6 0.000s s3:stat_bucket rgw::auth::s3::AWSAuthStrategy: trying
```

Next steps

- Integrate Hive and Spark
- Cache planner in Kariz