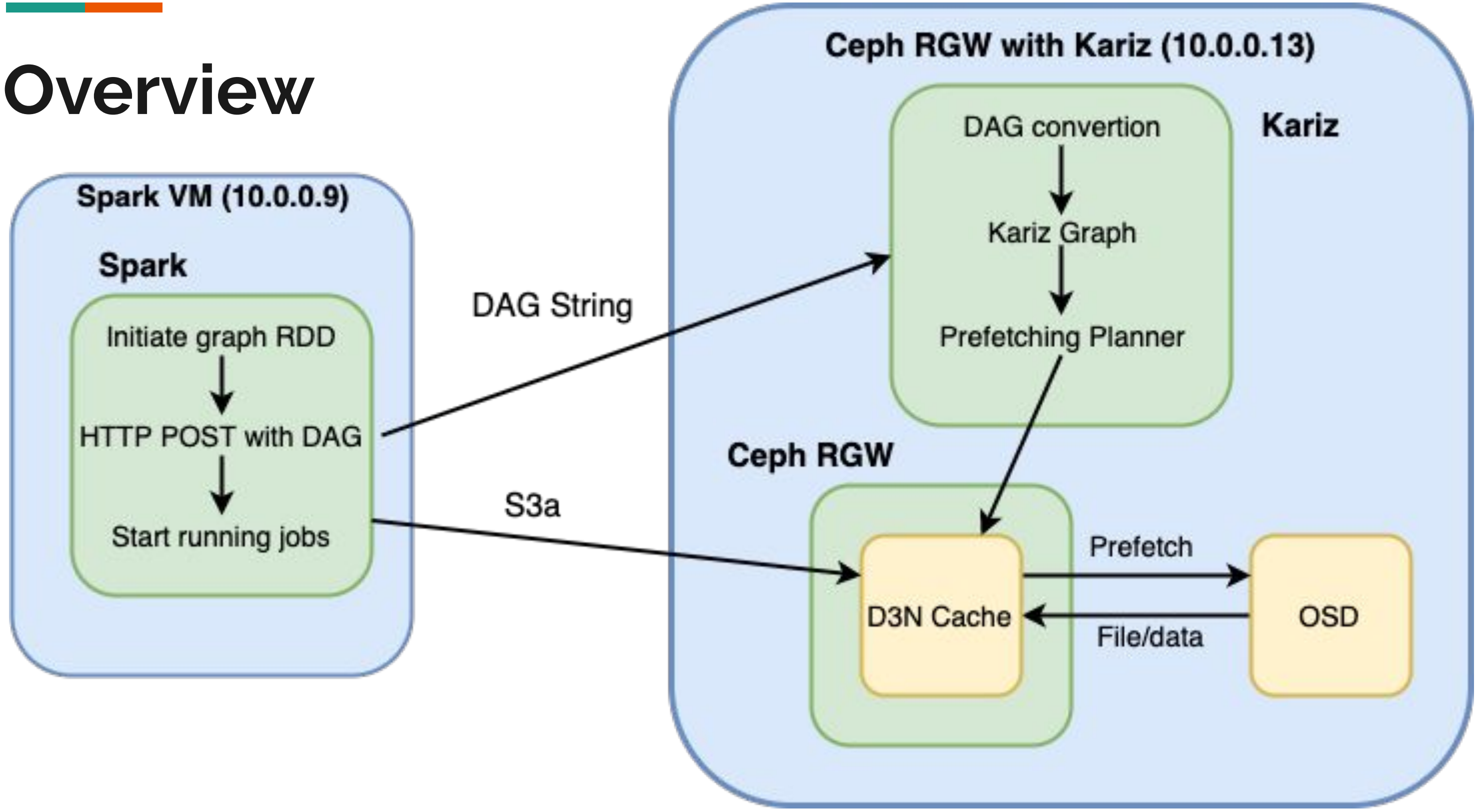
Ceph RGW Cache Prefetching for Batch Jobs

Xun Lin
Yang Qiao
Tianyi Tang
Gang Wei
Zhangyu Wan

Overview

- MVP:
    - Extract DAG out of Spark Applications
    - Find the job dependency path, generate cache planner
    - Prefetch Files/data while running batch jobs
    - Performance evaluation (with & without prefetching)

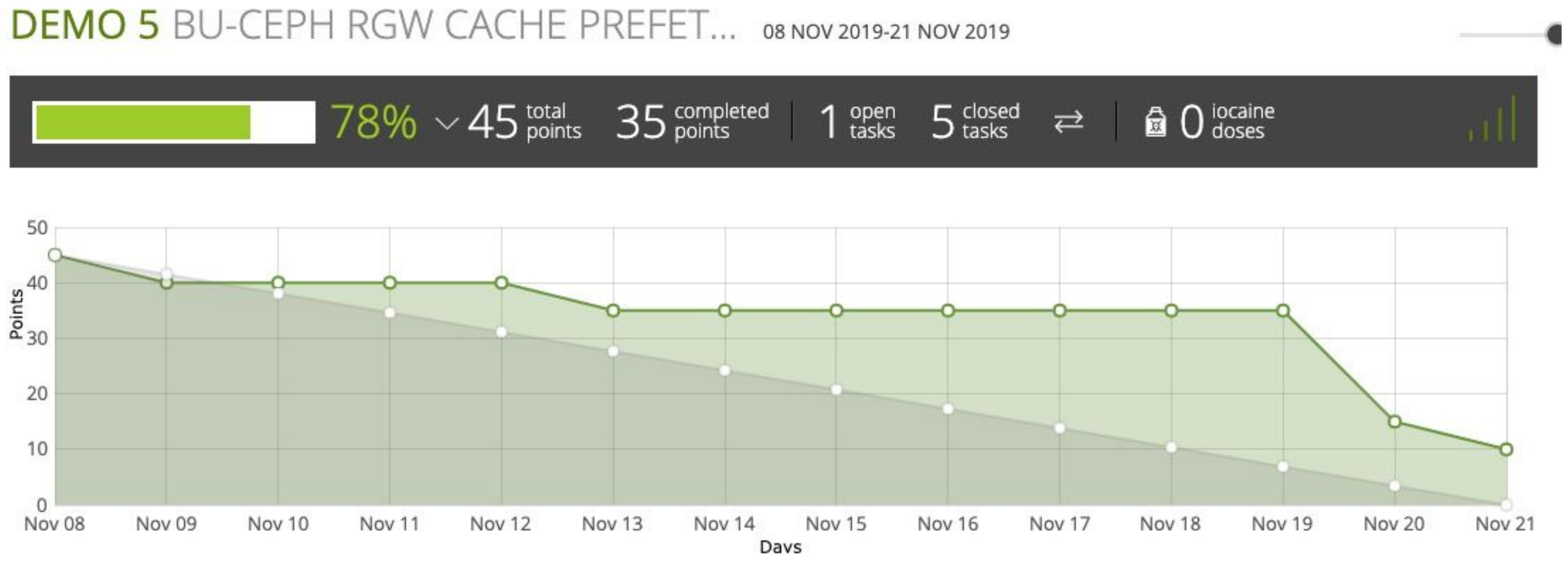
Overview


Spark VM (10.0.0.9)
Spark
Initiate graph RDD
HTTP POST with DAG
Start running jobs
DAG String
S3a
Ceph RGW with Kariz (10.0.0.13)
Kariz
DAG convertion
Kariz Graph
Prefetching Planner
Ceph RGW
D3N Cache
Prefetch
File/data
OSD

Progress

- **Finish End to End implementation**
    - Spark
        - Integration of Spark, Ceph and Kariz
    - Kariz cache planner
        - Parse requests to graph and planner
    - Prefetching
        - Prefetch files/data based on planner

Taiga



DEMO 5 BU-CEPH RGW CACHE PREFET... 08 NOV 2019-21 NOV 2019

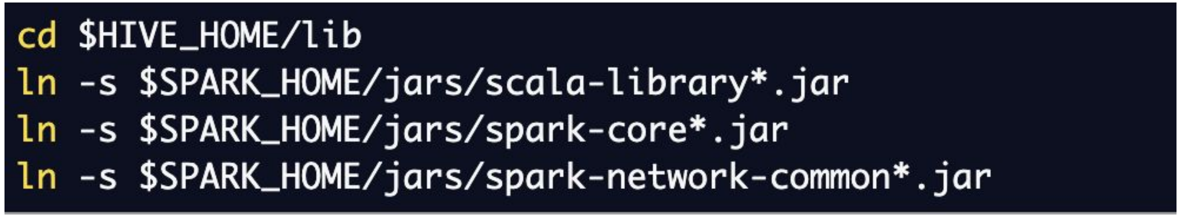
78% 45 total points 35 completed points 1 open tasks 5 closed tasks 0 iocaine doses

System Integration

- Environment
  - Spark/Hive in 10.0.0.9
  - Kariz/Ceph RGW in 10.0.0.13
- Communication
  - Spark - Kariz
  - Spark - Ceph RGW
  - Kariz - Ceph RGW

Hive with Spark

1. Start Spark in standalone mode and set environment variable
2. Link Scala and Spark jars in Hive lib folder

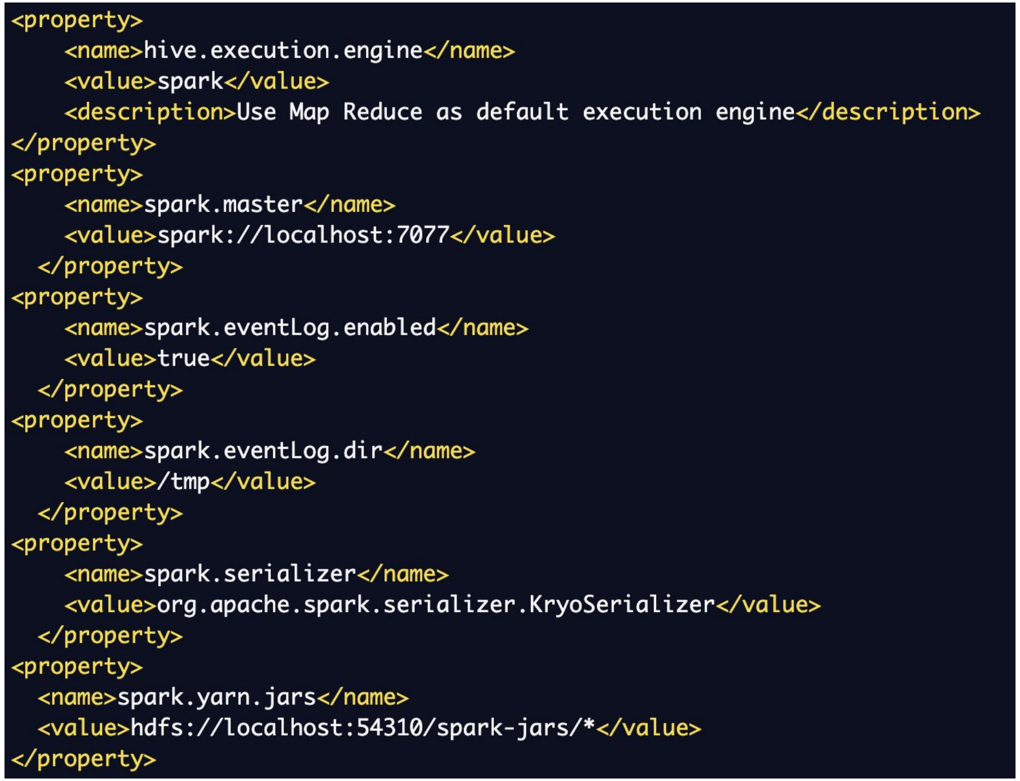


```
cd $HIVE_HOME/lib
ln -s $SPARK_HOME/jars/scala-library*.jar
ln -s $SPARK_HOME/jars/spark-core*.jar
ln -s $SPARK_HOME/jars/spark-network-common*.jar
```

Hive with Spark

3. Set the hive.execution.engine to Spark in hive-site.xml (configuration file)

4. Set Spark parameters in hive-site.xml and move jar dependencies to HDFS for hive



```xml
<property>
    <name>hive.execution.engine</name>
    <value>spark</value>
    <description>Use Map Reduce as default execution engine</description>
</property>
<property>
    <name>spark.master</name>
    <value>spark://localhost:7077</value>
  </property>
<property>
    <name>spark.eventLog.enabled</name>
    <value>true</value>
  </property>
<property>
    <name>spark.eventLog.dir</name>
    <value>/tmp</value>
  </property>
<property>
    <name>spark.serializer</name>
    <value>org.apache.spark.serializer.KryoSerializer</value>
  </property>
<property>
  <name>spark.yarn.jars</name>
  <value>hdfs://localhost:54310/spark-jars/*</value>
</property>
```

Spark with Ceph

1. Add dependencies in pom.xml
   a. aws-java-sdk 1.7.4
   b. hadoop-aws 2.7.3
2. Configure spark-defaults.conf
3. Configure spark-env.sh
4. Setup hadoop configuration in Spark application
   a. access key
   b. secrete key
   c. endpoint
   d. NativeS3FileSystem

Spark with Ceph

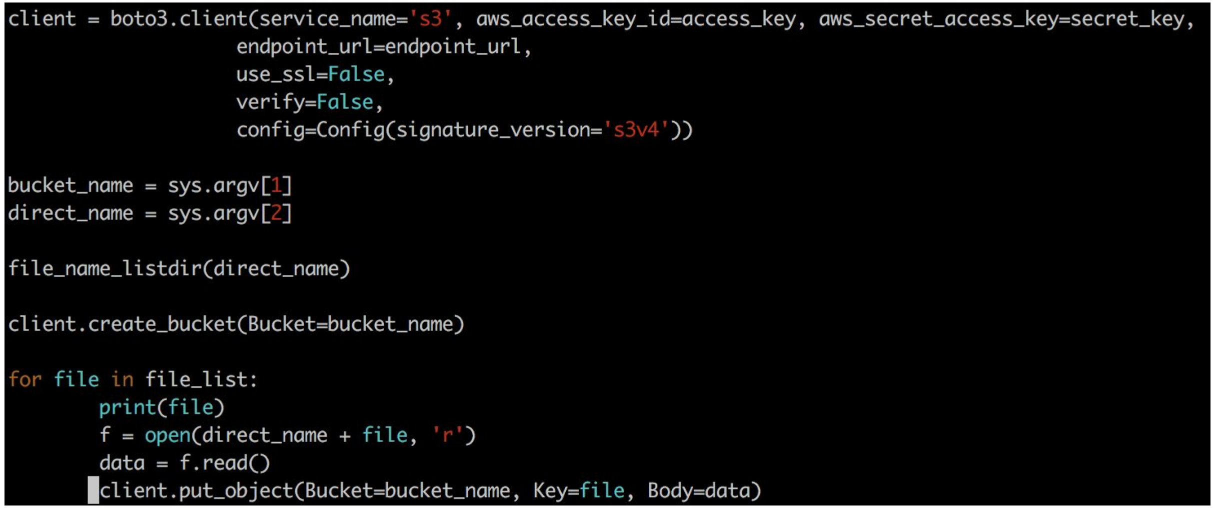
Two steps:

a. Creating ceph bucket and upload input files into ceph

b. Using s3a to access data in Ceph for running spark jobs

Spark with Ceph

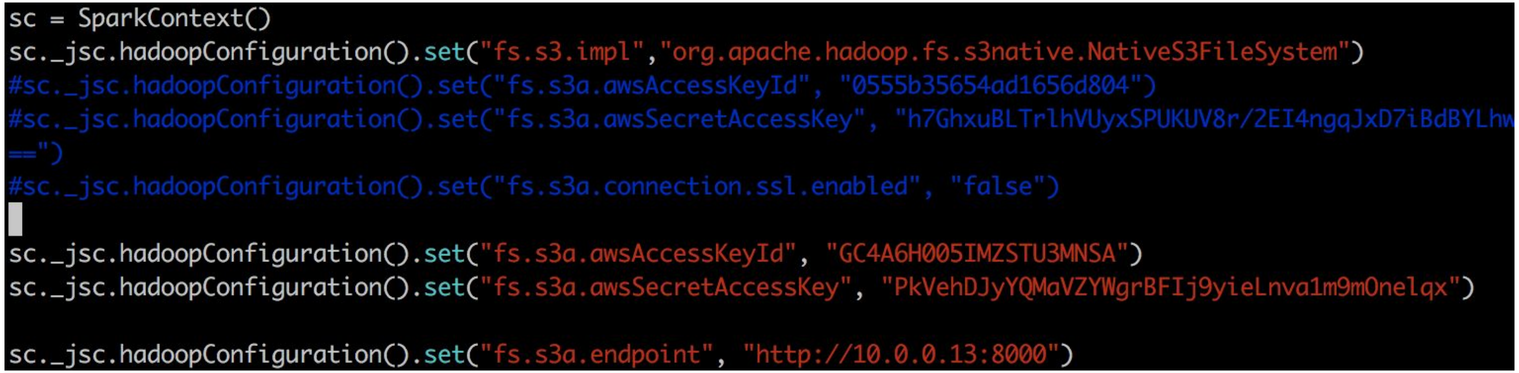
Creating ceph bucket and upload input files into ceph



```
client = boto3.client(service_name='s3', aws_access_key_id=access_key, aws_secret_access_key=secret_key,
                    endpoint_url=endpoint_url,
                    use_ssl=False,
                    verify=False,
                    config=Config(signature_version='s3v4'))

bucket_name = sys.argv[1]
direct_name = sys.argv[2]

file_name_listdir(direct_name)

client.create_bucket(Bucket=bucket_name)

for file in file_list:
        print(file)
        f = open(direct_name + file, 'r')
        data = f.read()
        client.put_object(Bucket=bucket_name, Key=file, Body=data)
```

Spark with Ceph

Using s3a to access data in Ceph for running spark jobs



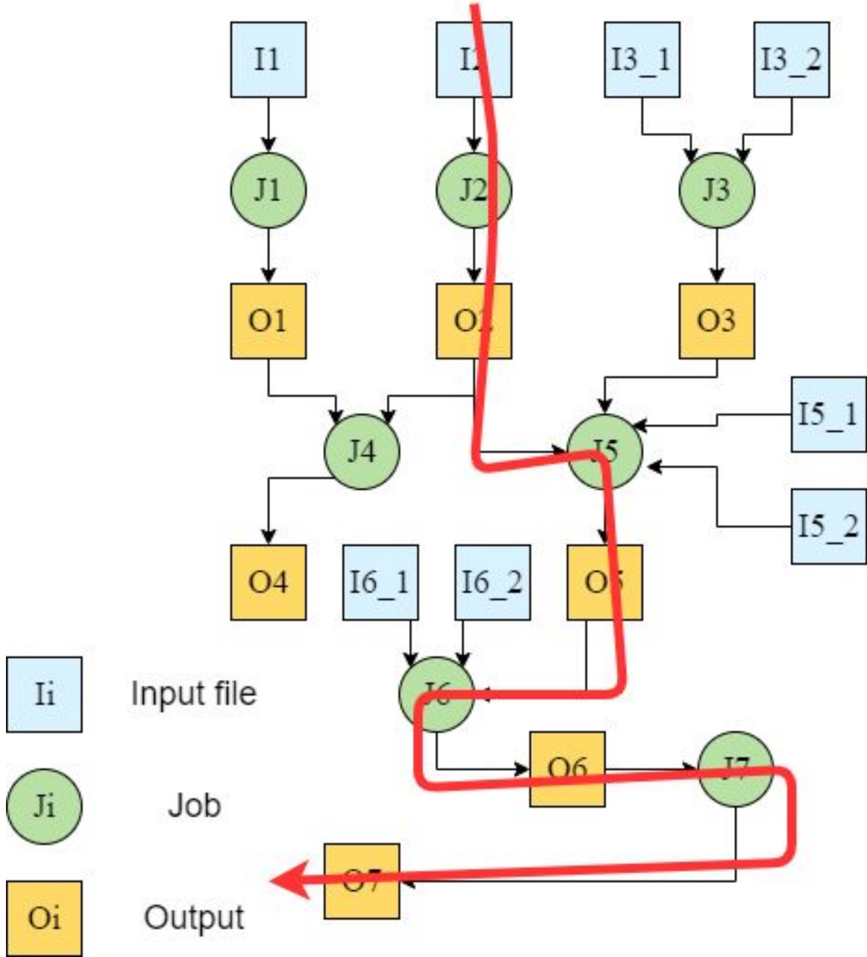
```
sc = SparkContext()
sc._jsc.hadoopConfiguration().set("fs.s3.impl","org.apache.hadoop.fs.s3native.NativeS3FileSystem")
#sc._jsc.hadoopConfiguration().set("fs.s3a.awsAccessKeyId", "0555b35654ad1656d804")
#sc._jsc.hadoopConfiguration().set("fs.s3a.awsSecretAccessKey", "h7GhxuBLTrlhVUyxSPUKUV8r/2EI4ngqJxD7iBdBYLhw
==")
#sc._jsc.hadoopConfiguration().set("fs.s3a.connection.ssl.enabled", "false")

sc._jsc.hadoopConfiguration().set("fs.s3a.awsAccessKeyId", "GC4A6H005IMZSTU3MNSA")
sc._jsc.hadoopConfiguration().set("fs.s3a.awsSecretAccessKey", "PkVehDJyYQMaVZYWgrBFIj9yieLnva1m9m0nelqx")

sc._jsc.hadoopConfiguration().set("fs.s3a.endpoint", "http://10.0.0.13:8000")
```

Spark with Kariz

1. Change the endpoint to Ceph RGW’s endpoint
2. Kariz daemon uses port 3188 and cache daemon uses port 3187
3. Run these two servers
    a. ${KARIZ_ROOT}/plans/kariz/api/server.py
    b. ${KARIZ_ROOT}/cache/server.py
4. Run Spark application
    a. Make sure Ceph is running correctly

Kariz Cache planner

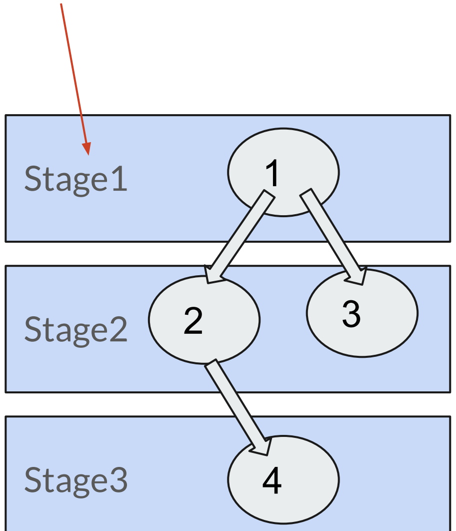
- Find the path with longest running time by Dijkstra.
- Assuming the red line is the longest path, plan to prefetch I2, I5_1, I5_2, I6_1, I6_2
- Get data size and file location



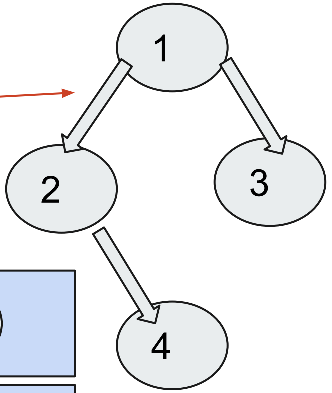
Prefetch

Now we have all we need: what /when to cache, files on longest path, data size, kariz with spark, and kariz with ceph.   Also : **Kariz already have the pig prefetching module.**

PIG: Next stage have to wait until all the node in last stage finish



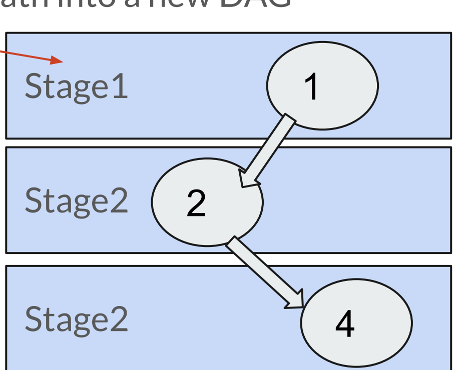
Spark: No such stage dependency



Trick: fit the longest path into a new DAG

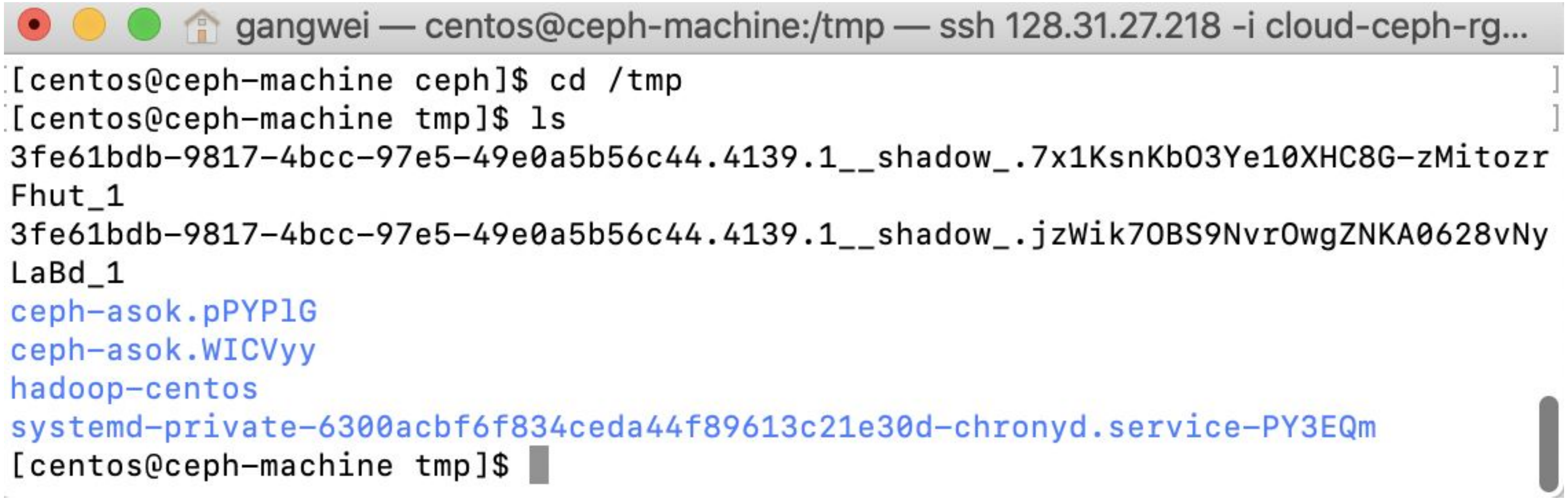
then use pig

Prefetching module.



Demo

Prefetched Data

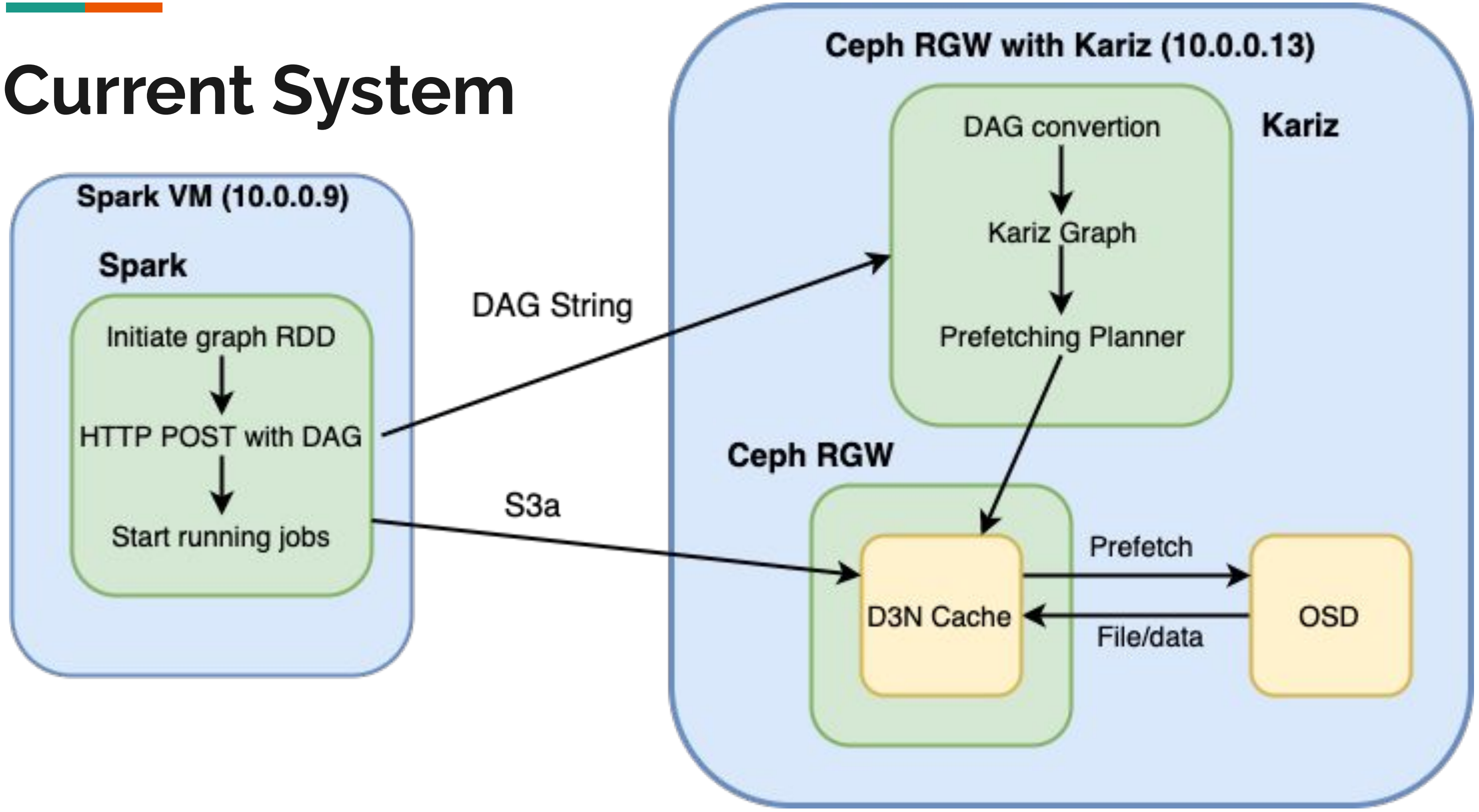

gangwei — centos@ceph-machine:/tmp — ssh 128.31.27.218 -i cloud-ceph-rg...

```
[centos@ceph-machine ceph]$ cd /tmp
[centos@ceph-machine tmp]$ ls
3fe61bdb-9817-4bcc-97e5-49e0a5b56c44.4139.1__shadow_.7x1KsnKbO3Ye10XHC8G-zMitozr
Fhut_1
3fe61bdb-9817-4bcc-97e5-49e0a5b56c44.4139.1__shadow_.jzWik7OBS9NvrOwgZNKA0628vNy
LaBd_1
ceph-asok.pPYPlG
ceph-asok.WICVyy
hadoop-centos
systemd-private-6300acbf6f834ceda44f89613c21e30d-chronyd.service-PY3EQm
[centos@ceph-machine tmp]$
```

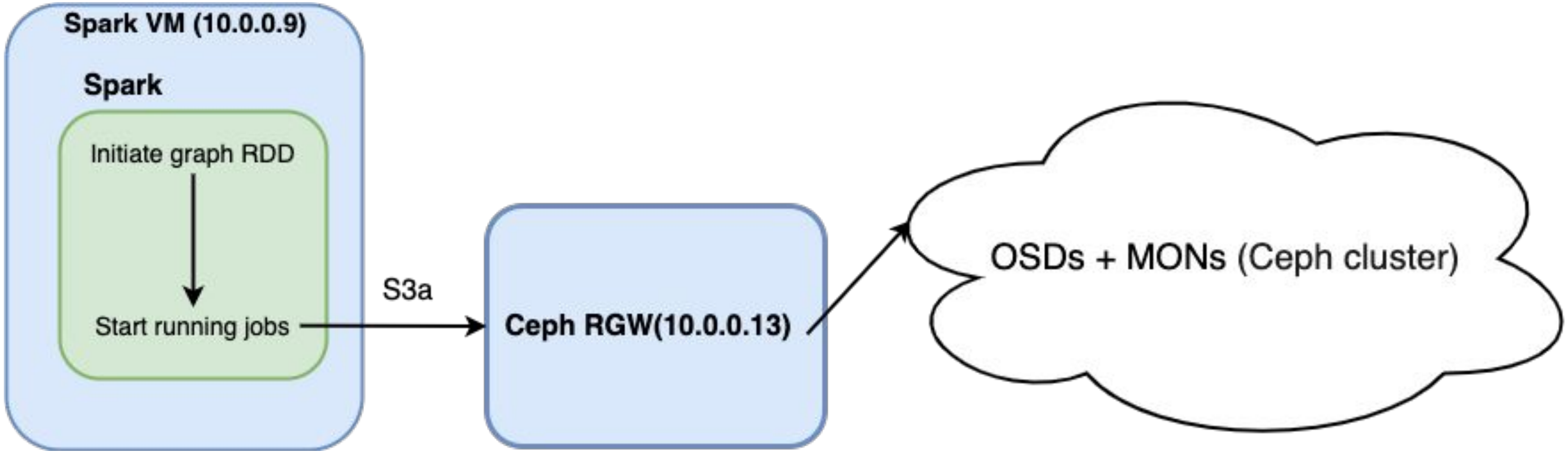
Next Step

1. Hive on Spark?
2. Performance Evaluation
    a. System adjustment
    b. Runtime comparison

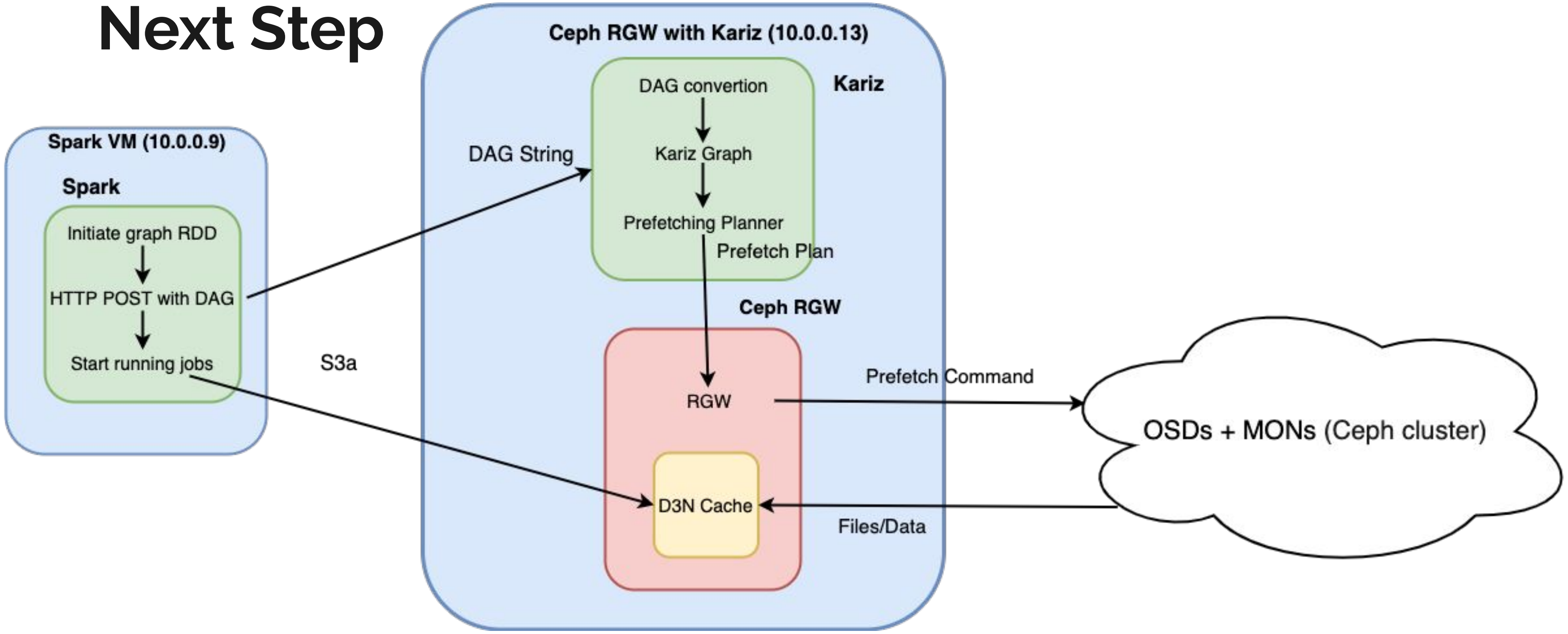
Current System



Next Step


Spark VM (10.0.0.9)
Spark
Initiate graph RDD
Start running jobs
S3a
Ceph RGW(10.0.0.13)
OSDs + MONs (Ceph cluster)

Next Step


Ceph RGW with Kariz (10.0.0.13)
Kariz
DAG convertion
Kariz Graph
Prefetching Planner
Prefetch Plan
DAG String
Spark VM (10.0.0.9)
Spark
Initiate graph RDD
HTTP POST with DAG
Start running jobs
S3a
Ceph RGW
RGW
D3N Cache
Prefetch Command
Files/Data
OSDs + MONs (Ceph cluster)