Ceph RGW Cache Prefetching for Batch Jobs

Xun Lin
Yang Qiao
Tianyi Tang
Gang Wei
Zhangyu Wan

Progress

- Problems in the last Demo:
  - DAG is only extracted at application level
  - Timing information is not clear for running Dijkstra

Progress

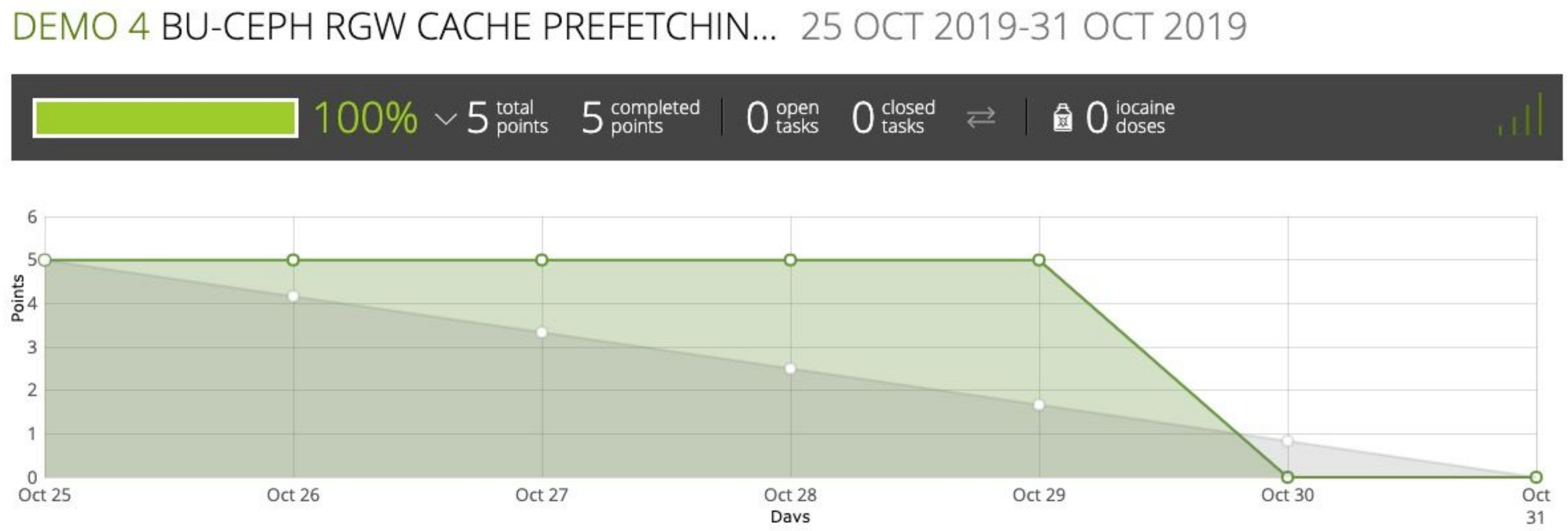
DAG:

- Change the Spark source code, compile it, extract DAG before actually running the jobs (DAG is extracted at platform level)
- Send DAG information to Kariz by HTTP request

Kariz:

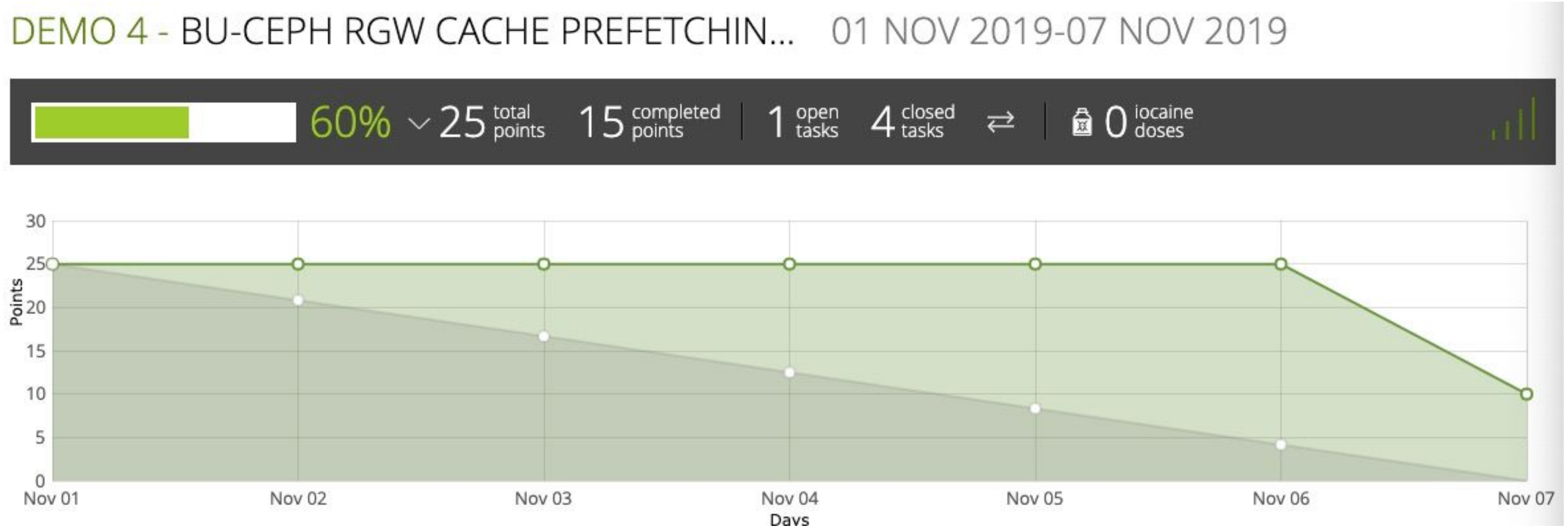
- Correct DAG string convert function
- Run a test with Kariz server, cache simulator and DAG simulator

Taiga


DEMO 4 BU-CEPH RGW CACHE PREFETCHIN... 25 OCT 2019-31 OCT 2019

100% 5 total points 5 completed points | 0 open tasks 0 closed tasks | 0 iocaine doses

Taiga



DEMO 4 - BU-CEPH RGW CACHE PREFETCHIN... 01 NOV 2019-07 NOV 2019

60% 25 total points 15 completed points 1 open tasks 4 closed tasks 0 iocaine doses

Spark Integration with Kariz

- **Why?**

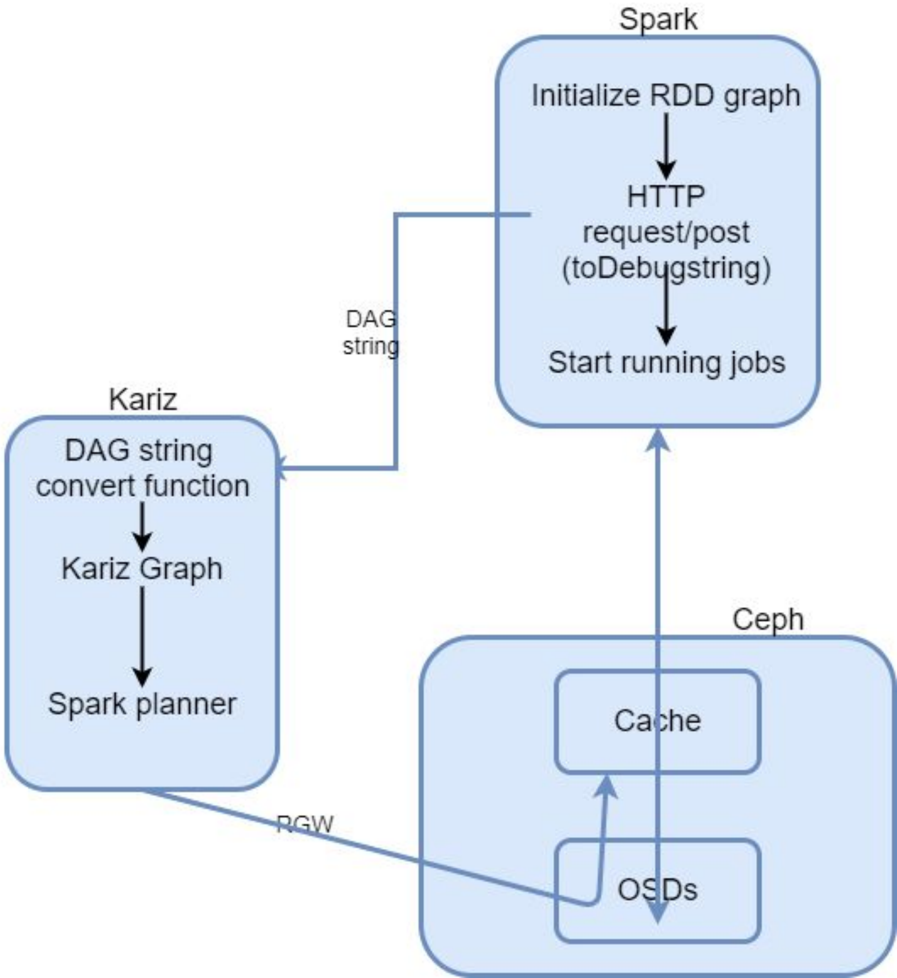
  Kariz need DAG information to determine what and when to prefetch.

- **How?**

  Send HTTP **POST** request with DAG string.

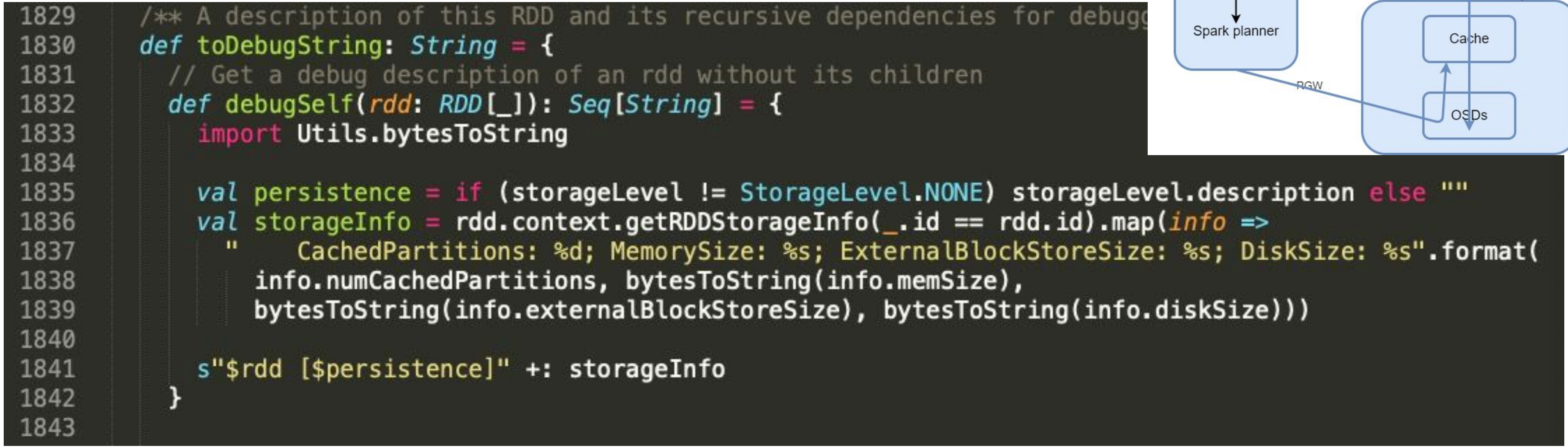
How system works

1. Spark generates the DAG before running jobs.
2. Spark sends DAG string to Kariz.
3. Kariz translates the string into its Graph class and submit the json file to its planner.
4. The planner prefetch data from Ceph into D3N cache
5. Spark runs the jobs (files are already in cache)

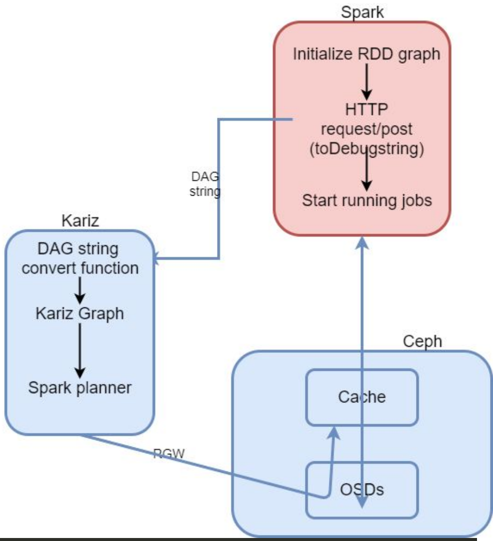


RDD.scala

(spark/core/src/main/scala/org/apache/spark/rdd/RDD.scala)

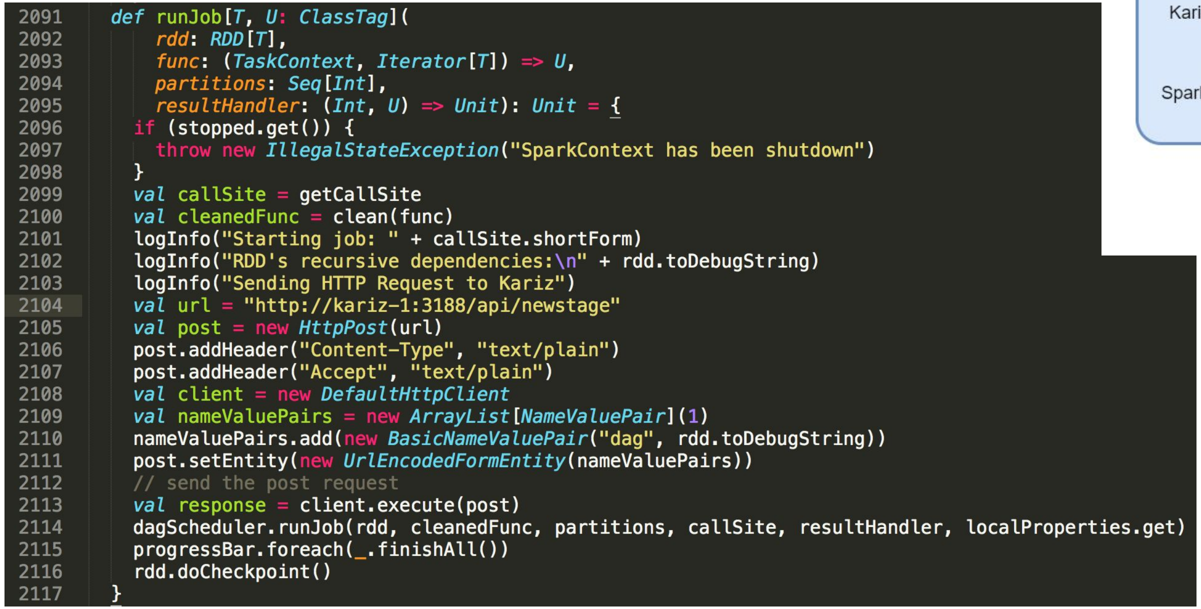


```scala
  /** A description of this RDD and its recursive dependencies for debugg
  def toDebugString: String = {
    // Get a debug description of an rdd without its children
    def debugSelf(rdd: RDD[_]): Seq[String] = {
      import Utils.bytesToString

      val persistence = if (storageLevel != StorageLevel.NONE) storageLevel.description else ""
      val storageInfo = rdd.context.getRDDStorageInfo(_.id == rdd.id).map(info =>
        "    CachedPartitions: %d; MemorySize: %s; ExternalBlockStoreSize: %s; DiskSize: %s".format(
          info.numCachedPartitions, bytesToString(info.memSize),
          bytesToString(info.externalBlockStoreSize), bytesToString(info.diskSize)))

      s"$rdd [$persistence]" +: storageInfo
    }

```

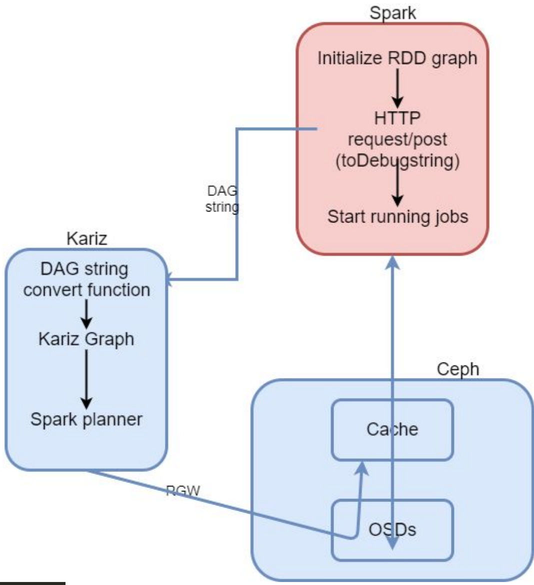


SparkContext.scala

(spark/core/src/main/scala/org/apache/spark/SparkContext.scala)



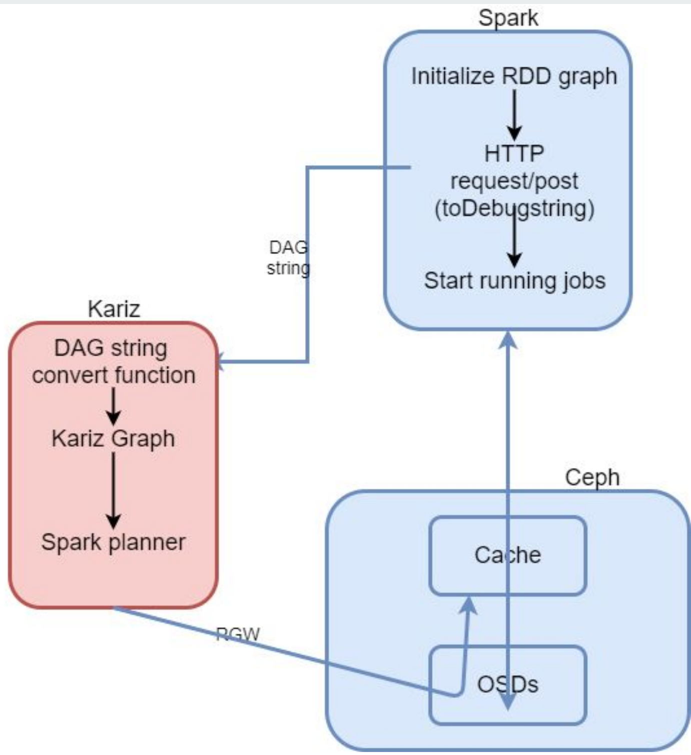
```scala
def runJob[T, U: ClassTag](
    rdd: RDD[T],
    func: (TaskContext, Iterator[T]) => U,
    partitions: Seq[Int],
    resultHandler: (Int, U) => Unit): Unit = {
  if (stopped.get()) {
    throw new IllegalStateException("SparkContext has been shutdown")
  }
  val callSite = getCallSite
  val cleanedFunc = clean(func)
  logInfo("Starting job: " + callSite.shortForm)
  logInfo("RDD's recursive dependencies:\n" + rdd.toDebugString)
  logInfo("Sending HTTP Request to Kariz")
  val url = "http://kariz-1:3188/api/newstage"
  val post = new HttpPost(url)
  post.addHeader("Content-Type", "text/plain")
  post.addHeader("Accept", "text/plain")
  val client = new DefaultHttpClient
  val nameValuePairs = new ArrayList[NameValuePair](1)
  nameValuePairs.add(new BasicNameValuePair("dag", rdd.toDebugString))
  post.setEntity(new UrlEncodedFormEntity(nameValuePairs))
  // send the post request
  val response = client.execute(post)
  dagScheduler.runJob(rdd, cleanedFunc, partitions, callSite, resultHandler, localProperties.get)
  progressBar.foreach(_.finishAll())
  rdd.doCheckpoint()
}
```



Demo

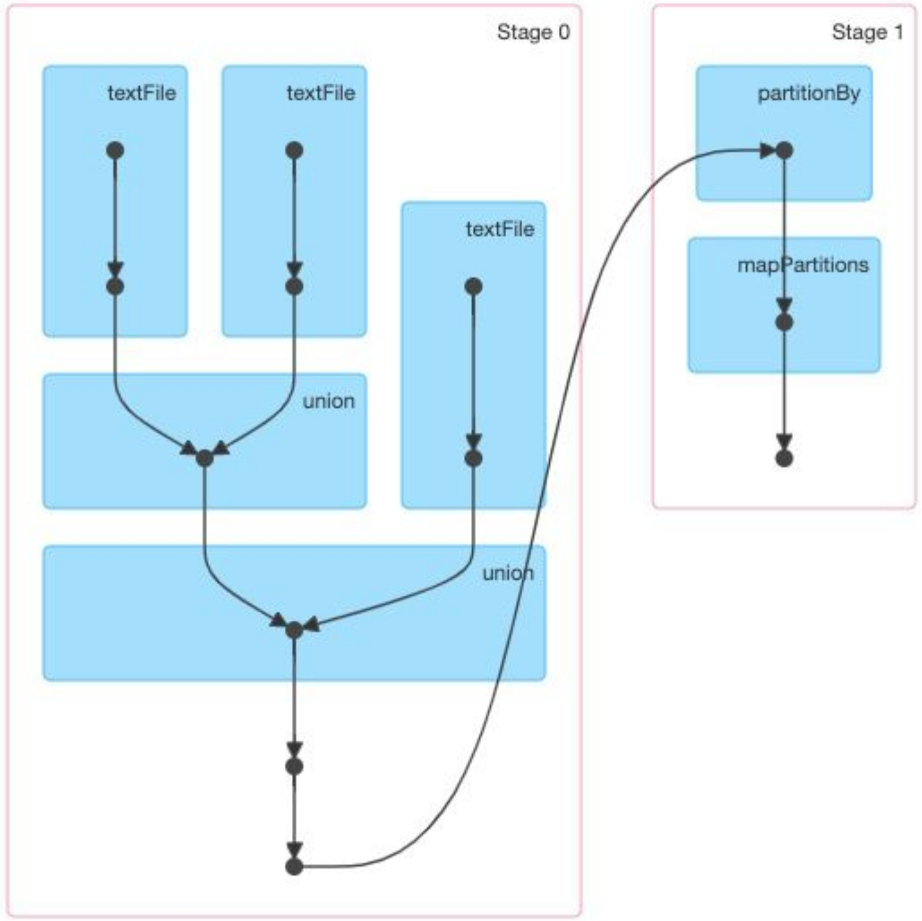
DAG string convert function

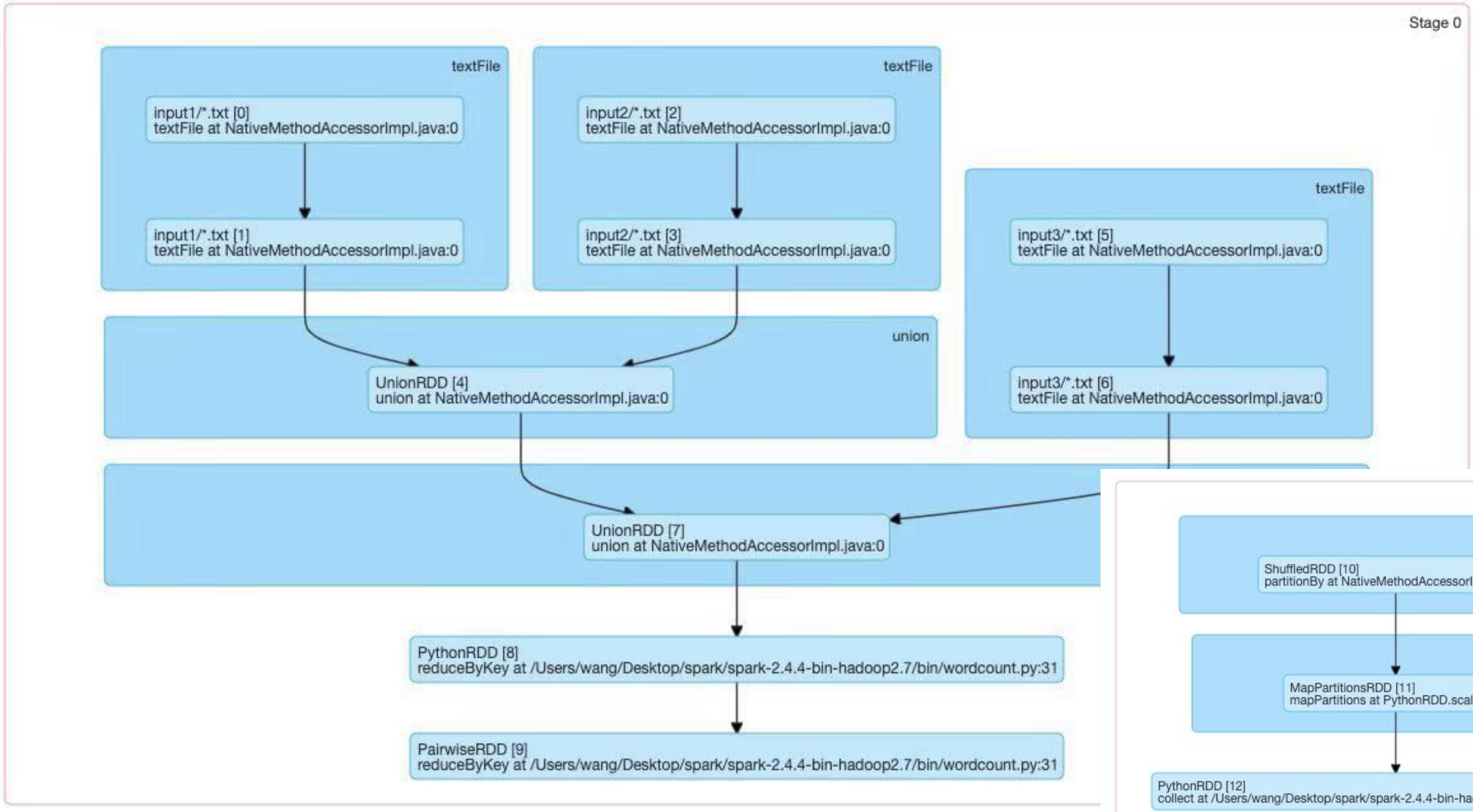
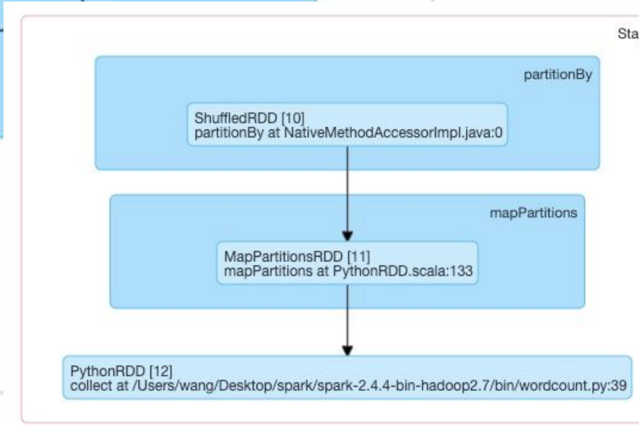
- Previously we thought the input of one node is the output rdd of the following node.
- From the Spark UI: the input of one node is all the output rdds below the node whose rdd number is smaller than the node output rdd number.



DAG visualization (multiple path)

```
(8) PythonRDD[12] at collect at
/Users/joe/Desktop/Spark_Source/spark/bin/wordcount.py:39 []
 |  MapPartitionsRDD[11] at mapPartitions at PythonRDD.scala:133 []
 |  ShuffledRDD[10] at partitionBy at NativeMethodAccessorImpl.java:0 []
 +-(8) PairwiseRDD[9] at reduceByKey at
/Users/joe/Desktop/Spark_Source/spark/bin/wordcount.py:31 []
    |  PythonRDD[8] at reduceByKey at
/Users/joe/Desktop/Spark_Source/spark/bin/wordcount.py:31 []
    |  UnionRDD[7] at union at NativeMethodAccessorImpl.java:0 []
    |  UnionRDD[4] at union at NativeMethodAccessorImpl.java:0 []
    |  input1/*.txt MapPartitionsRDD[1] at textFile at NativeMethodAccessorImpl.java:0 []
    |  input1/*.txt HadoopRDD[0] at textFile at NativeMethodAccessorImpl.java:0 []
    |  input2/*.txt MapPartitionsRDD[3] at textFile at NativeMethodAccessorImpl.java:0 []
    |  input2/*.txt HadoopRDD[2] at textFile at NativeMethodAccessorImpl.java:0 []
    |  input3/*.txt MapPartitionsRDD[6] at textFile at NativeMethodAccessorImpl.java:0 []
    |  input3/*.txt HadoopRDD[5] at textFile at NativeMethodAccessorImpl.java:0 []
```




Stage 0
textFile
input1/*.txt [0]
textFile at NativeMethodAccessorImpl.java:0
input1/*.txt [1]
textFile at NativeMethodAccessorImpl.java:0
textFile
input2/*.txt [2]
textFile at NativeMethodAccessorImpl.java:0
input2/*.txt [3]
textFile at NativeMethodAccessorImpl.java:0
textFile
input3/*.txt [5]
textFile at NativeMethodAccessorImpl.java:0
input3/*.txt [6]
textFile at NativeMethodAccessorImpl.java:0
union
UnionRDD [4]
union at NativeMethodAccessorImpl.java:0
UnionRDD [7]
union at NativeMethodAccessorImpl.java:0
PythonRDD [8]
reduceByKey at /Users/wang/Desktop/spark/spark-2.4.4-bin-hadoop2.7/bin/wordcount.py:31
PairwiseRDD [9]
reduceByKey at /Users/wang/Desktop/spark/spark-2.4.4-bin-hadoop2.7/bin/wordcount.py:31

Sta
partitionBy
ShuffledRDD [10]
partitionBy at NativeMethodAccessorImpl.java:0
mapPartitions
MapPartitionsRDD [11]
mapPartitions at PythonRDD.scala:133
PythonRDD [12]
collect at /Users/wang/Desktop/spark/spark-2.4.4-bin-hadoop2.7/bin/wordcount.py:39

Longest Path

- Why need longest path？

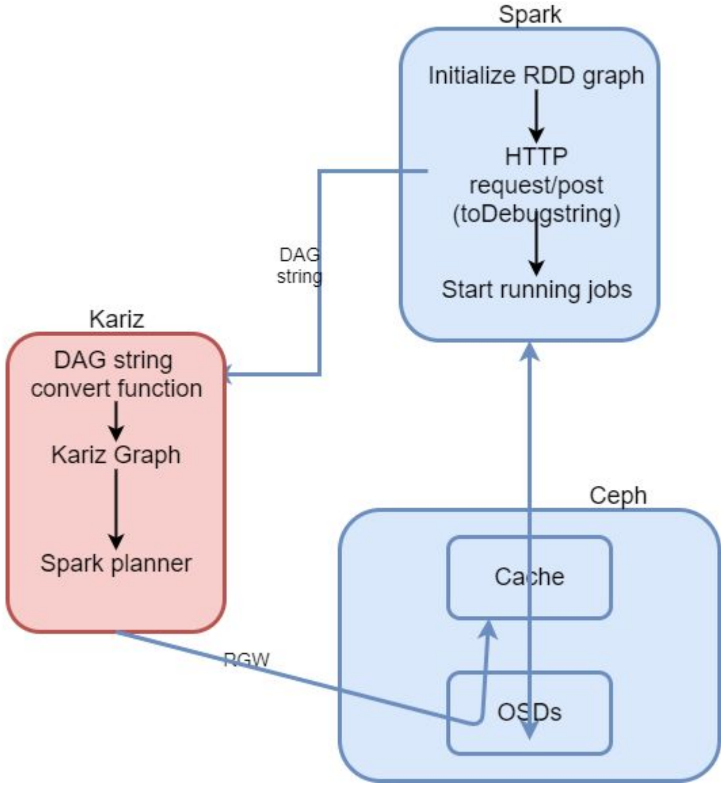
Kariz is to designed to prefetch the data that **reduce runtime the most**

How many cache we have

Statistics ,(job types) from spark → prediction of runtime

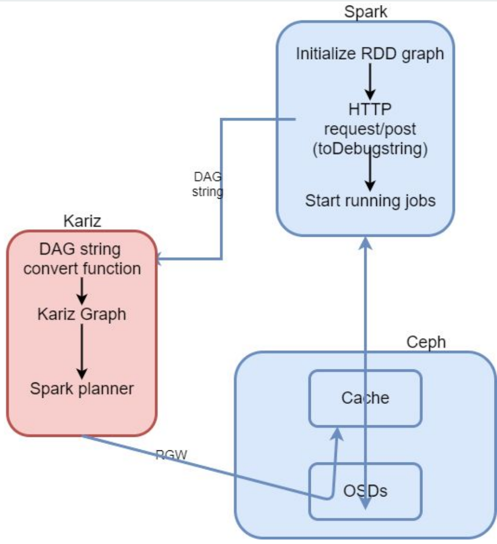
Use the longest path → cache planner for spark

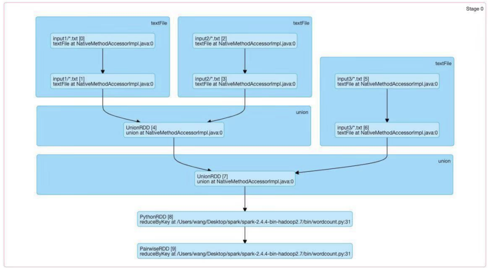
Analysis: Worst case/ Best case/ Average case

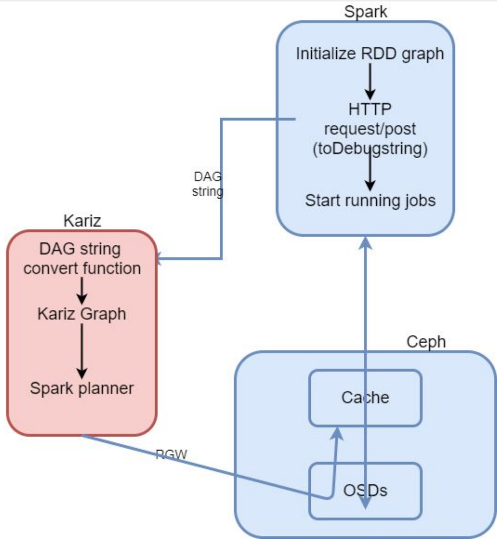


Longest Path

- Timing information of all nodes is currently randomly set.
- Individual job execution times can be predicted with data from past executions of the same job type and network and storage system bandwidth, which is measured by Kariz.
- We will make Spark send job type to Kariz and integrate the longest path into Spark planner in the next sprint.




Stage 0
textFile
input1/*.txt [0]
textFile at NativeMethodAccessorImpl.java:0
input1/*.txt [1]
textFile at NativeMethodAccessorImpl.java:0
textFile
input2/*.txt [2]
textFile at NativeMethodAccessorImpl.java:0
input2/*.txt [3]
textFile at NativeMethodAccessorImpl.java:0
textFile
input3/*.txt [5]
textFile at NativeMethodAccessorImpl.java:0
input3/*.txt [6]
textFile at NativeMethodAccessorImpl.java:0
union
UnionRDD [4]
union at NativeMethodAccessorImpl.java:0
union
UnionRDD [7]
union at NativeMethodAccessorImpl.java:0
PythonRDD [8]
reduceByKey at /Users/wang/Desktop/spark/spark-2.4.4-bin-hadoop2.7/bin/wordcount.py:31
PairwiseRDD [9]
reduceByKey at /Users/wang/Desktop/spark/spark-2.4.4-bin-hadoop2.7/bin/wordcount.py:31



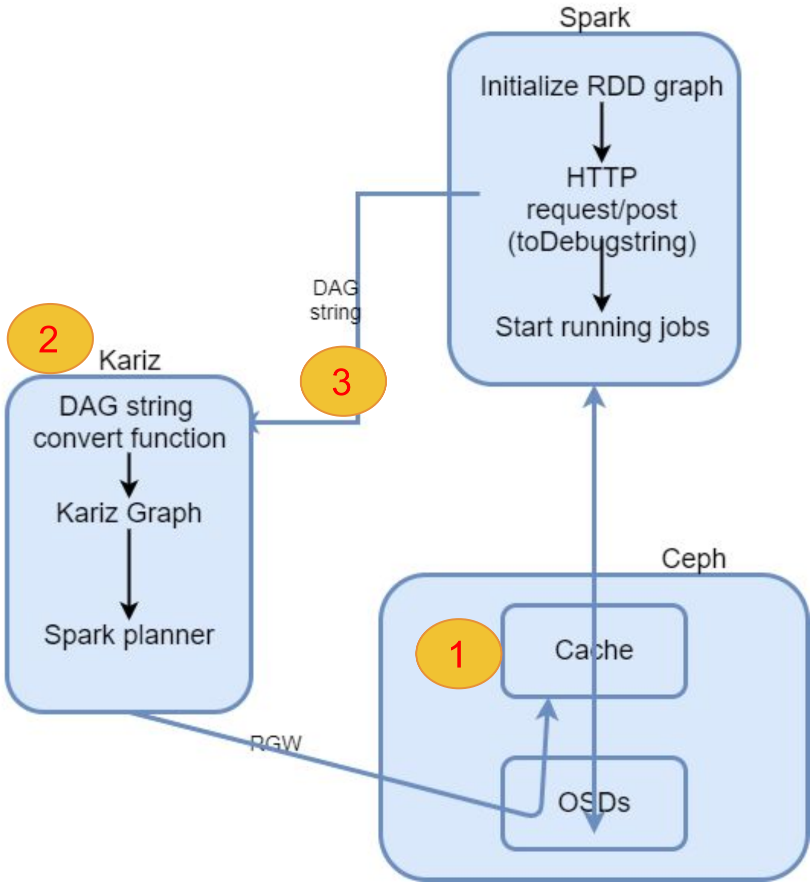
Longest Path

{0: {'output': 'HadoopRDD[5]', 'inputs': ['input3/*.txt']}, 1: {'output': 'MapPartitionsRDD[6]', 'inputs': ['HadoopRDD[5]', 'input3/*.txt']}, 2: {'output': 'HadoopRDD[2]', 'inputs': ['input2/*.txt']}, 3: {'output': 'MapPartitionsRDD[3]', 'inputs': ['HadoopRDD[2]', 'input2/*.txt']}, 4: {'output': 'HadoopRDD[0]', 'inputs': ['input1/*.txt']}, 5: {'output': 'MapPartitionsRDD[1]', 'inputs': ['HadoopRDD[0]', 'input1/*.txt']}, 6: {'output': 'UnionRDD[4]', 'inputs': ['MapPartitionsRDD[3]', 'MapPartitionsRDD[1]']}, 7: {'output': 'UnionRDD[7]', 'inputs': ['MapPartitionsRDD[6]', 'UnionRDD[4]']}, 8: {'output': 'PythonRDD[8]', 'inputs': ['UnionRDD[7]']}, 9: {'output': 'PairwiseRDD[9]', 'inputs': ['PythonRDD[8]']}, 10: {'output': 'ShuffledRDD[10]', 'inputs': ['PairwiseRDD[9]']}, 11: {'output': 'MapPartitionsRDD[11]', 'inputs': ['ShuffledRDD[10]']}, 12: {'output': 'PythonRDD[12]', 'inputs': ['MapPartitionsRDD[11]']}}

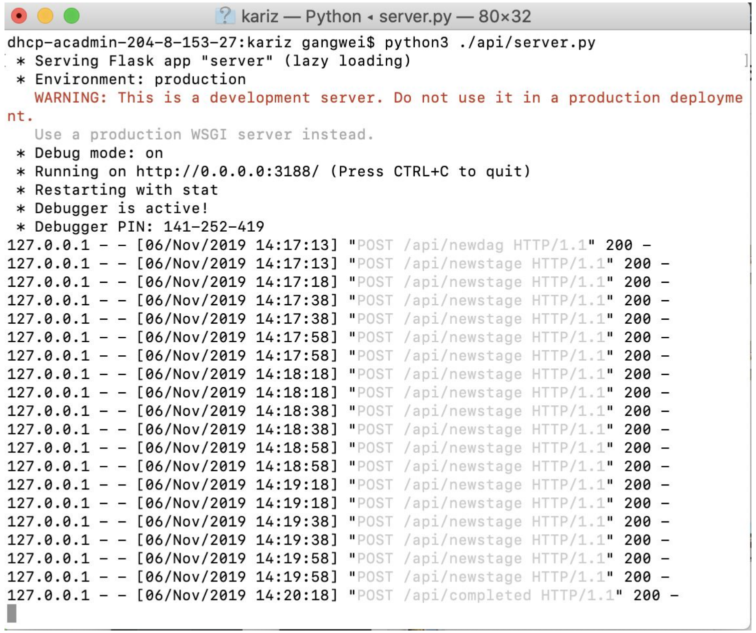
```
Longest path starting at node  0  ends at node  12  with length  -7
The longest route is  [0, 1, 7, 8, 9, 10, 11, 12]
Longest path starting at node  2  ends at node  12  with length  -8
The longest route is  [2, 3, 6, 7, 8, 9, 10, 11, 12]
Longest path starting at node  4  ends at node  12  with length  -8
The longest route is  [4, 5, 6, 7, 8, 9, 10, 11, 12]
```

Kariz simulation

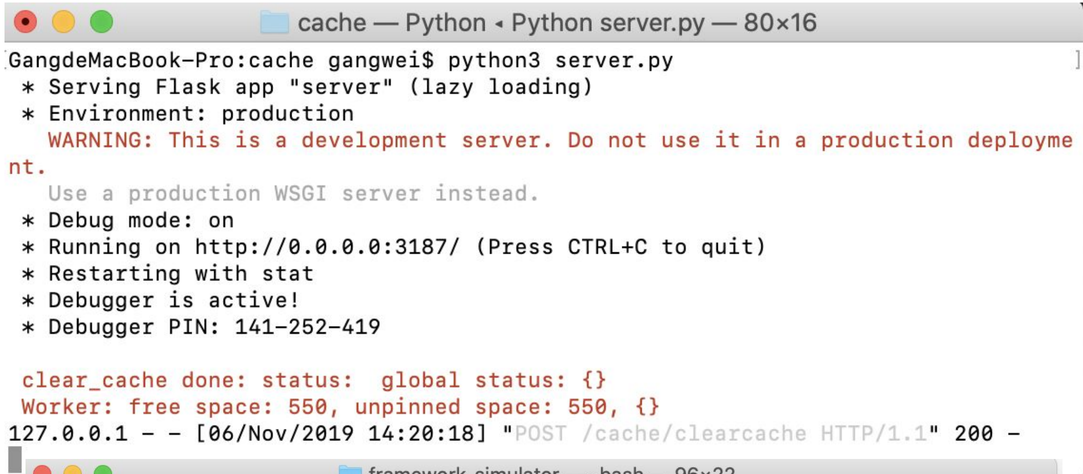
1. Cache simulator
   - Simulates as D3N cache
2. Server simulator with Kariz running on it
3. DAG simulator
   - Submits DAG to Kariz



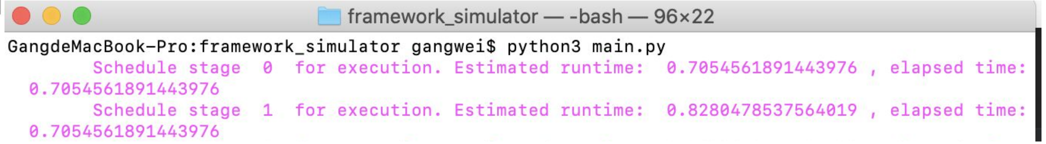
Kariz simulation


kariz — Python ◂ server.py — 80×32

```
dhcp-acadmin-204-8-153-27:kariz gangwei$ python3 ./api/server.py
 * Serving Flask app "server" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployme
nt.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://0.0.0.0:3188/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 141-252-419
127.0.0.1 - - [06/Nov/2019 14:17:13] "POST /api/newdag HTTP/1.1" 200 -
127.0.0.1 - - [06/Nov/2019 14:17:13] "POST /api/newstage HTTP/1.1" 200 -
127.0.0.1 - - [06/Nov/2019 14:17:18] "POST /api/newstage HTTP/1.1" 200 -
127.0.0.1 - - [06/Nov/2019 14:17:38] "POST /api/newstage HTTP/1.1" 200 -
127.0.0.1 - - [06/Nov/2019 14:17:38] "POST /api/newstage HTTP/1.1" 200 -
127.0.0.1 - - [06/Nov/2019 14:17:58] "POST /api/newstage HTTP/1.1" 200 -
127.0.0.1 - - [06/Nov/2019 14:17:58] "POST /api/newstage HTTP/1.1" 200 -
127.0.0.1 - - [06/Nov/2019 14:18:18] "POST /api/newstage HTTP/1.1" 200 -
127.0.0.1 - - [06/Nov/2019 14:18:18] "POST /api/newstage HTTP/1.1" 200 -
127.0.0.1 - - [06/Nov/2019 14:18:38] "POST /api/newstage HTTP/1.1" 200 -
127.0.0.1 - - [06/Nov/2019 14:18:38] "POST /api/newstage HTTP/1.1" 200 -
127.0.0.1 - - [06/Nov/2019 14:18:58] "POST /api/newstage HTTP/1.1" 200 -
127.0.0.1 - - [06/Nov/2019 14:18:58] "POST /api/newstage HTTP/1.1" 200 -
127.0.0.1 - - [06/Nov/2019 14:19:18] "POST /api/newstage HTTP/1.1" 200 -
127.0.0.1 - - [06/Nov/2019 14:19:18] "POST /api/newstage HTTP/1.1" 200 -
127.0.0.1 - - [06/Nov/2019 14:19:38] "POST /api/newstage HTTP/1.1" 200 -
127.0.0.1 - - [06/Nov/2019 14:19:38] "POST /api/newstage HTTP/1.1" 200 -
127.0.0.1 - - [06/Nov/2019 14:19:58] "POST /api/newstage HTTP/1.1" 200 -
127.0.0.1 - - [06/Nov/2019 14:19:58] "POST /api/newstage HTTP/1.1" 200 -
127.0.0.1 - - [06/Nov/2019 14:20:18] "POST /api/completed HTTP/1.1" 200 -
```


cache — Python ◂ Python server.py — 80×16

```
GangdeMacBook-Pro:cache gangwei$ python3 server.py
 * Serving Flask app "server" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployme
nt.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://0.0.0.0:3187/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 141-252-419

 clear_cache done: status:  global status: {}
 Worker: free space: 550, unpinned space: 550, {}
127.0.0.1 - - [06/Nov/2019 14:20:18] "POST /cache/clearcache HTTP/1.1" 200 -
```


framework_simulator — -bash — 96×22

```
GangdeMacBook-Pro:framework_simulator gangwei$ python3 main.py
        Schedule stage  0  for execution. Estimated runtime:  0.7054561891443976 , elapsed time:
 0.7054561891443976
        Schedule stage  1  for execution. Estimated runtime:  0.8280478537564019 , elapsed time:
 0.7054561891443976
```

Next sprint

- Kariz GET request
- Spark send job type to Kariz
- Apply the longest path to Spark planner
- Replace Cache simulator with real Ceph Cache