20th May – Task3

Problem:

- Implement the code of Future objects in Scala and check where it lacks for RDDs in Python. attach snapshots of code implemented.
- Explore about the java and Scala Future objects used for concurrency of partitions in executors.

Walkthrough:

We know that spark used rdd to partition a collection of data and parallelly processes it. But it does not signify that the individual tasks does not depend on each other and are independent.

Using future objects we can do parallel programming where a unit of work is allowed to run separately from the primary application thread. When the work is complete, it notifies the main thread about the completion or failure of the worker thread.

Scala Future Objects:

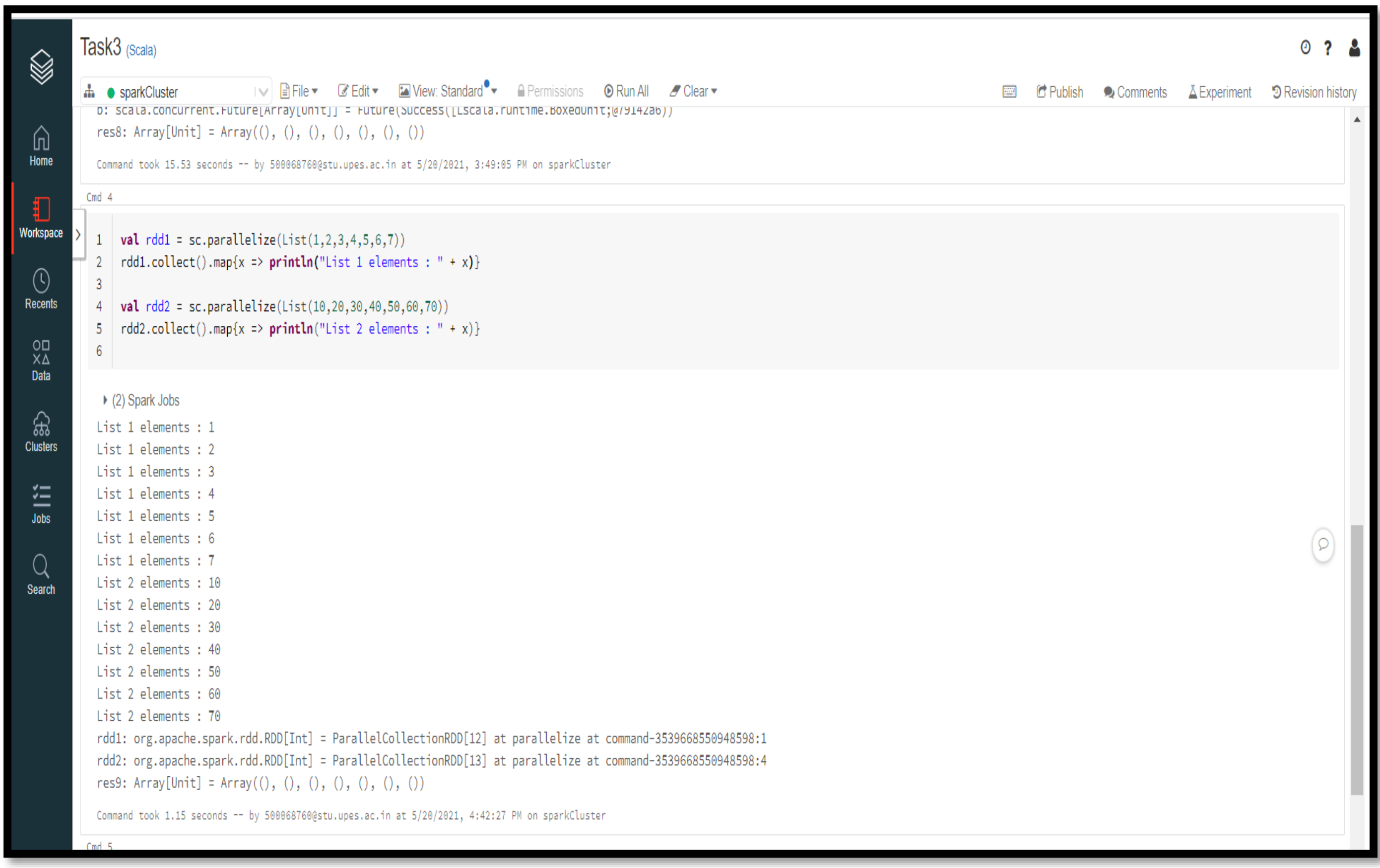
A Future gives you a simple way to run a job inside your spark application concurrently.

Scala support Thread pools, Thread pools is nothing but tarting a thread required us to allocate a memory region for its call stack and context switch from one Thread to another. Which consumes much more time than work in the concurrent task. For this reason, most concurrency frameworks have facilities that maintain a set of Threads in a waiting state and start running when concurrently executable work tasks become available, and we call this as Thread Pools.

In Scala, by importing the packages mentioned as:

- import scala.concurrent.ExecutionContext.Implicits.global
- import scala.concurrent.Future
- import scala.Success
- import scala.Failure

Snapshots:

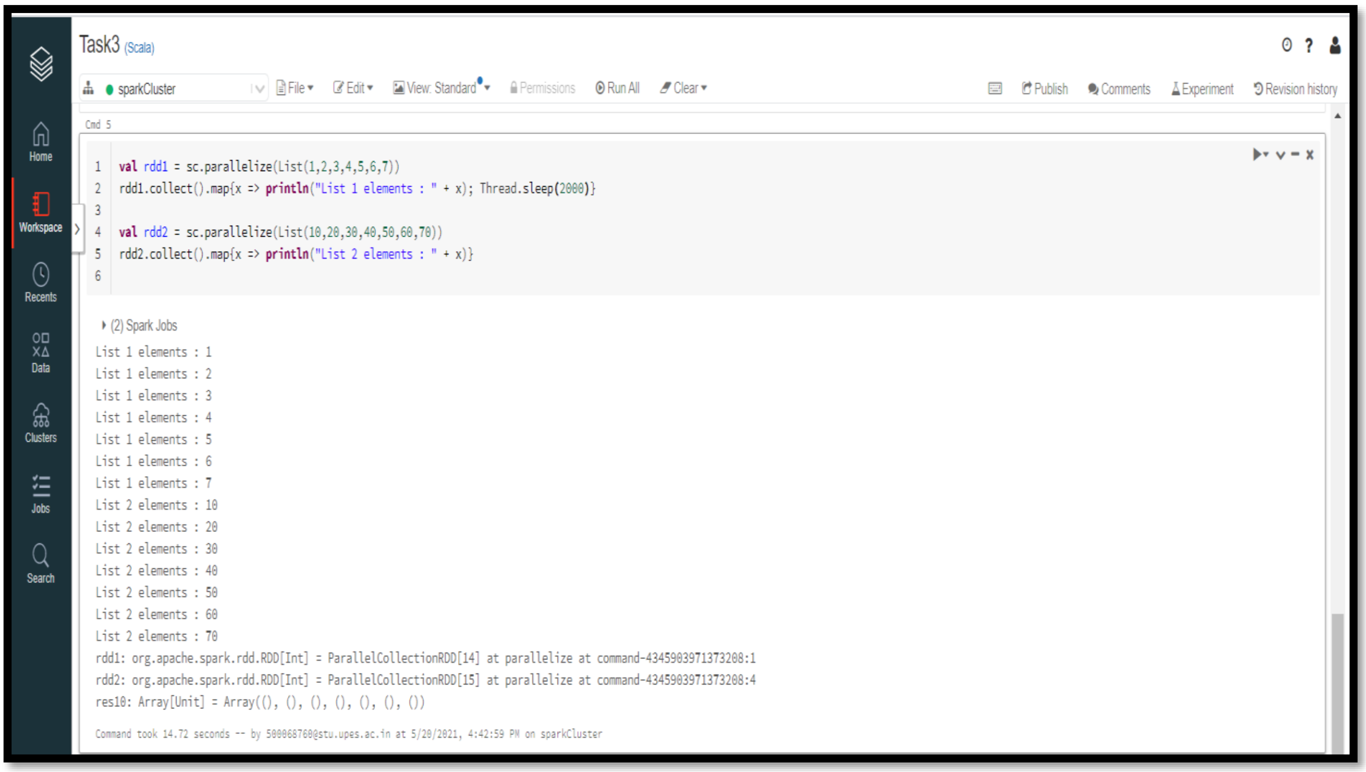


```
b: scala.concurrent.Future[Array[Unit]] = Future(Success([Lscala.runtime.BoxedUnit;@79142a6))
res8: Array[Unit] = Array((), (), (), (), (), (), ())

Command took 15.53 seconds -- by 500068760@stu.upes.ac.in at 5/20/2021, 3:49:05 PM on sparkCluster
```

```
val rdd1 = sc.parallelize(List(1,2,3,4,5,6,7))
rdd1.collect().map{x => println("List 1 elements : " + x)}

val rdd2 = sc.parallelize(List(10,20,30,40,50,60,70))
rdd2.collect().map{x => println("List 2 elements : " + x)}

```

```
List 1 elements : 1
List 1 elements : 2
List 1 elements : 3
List 1 elements : 4
List 1 elements : 5
List 1 elements : 6
List 1 elements : 7
List 2 elements : 10
List 2 elements : 20
List 2 elements : 30
List 2 elements : 40
List 2 elements : 50
List 2 elements : 60
List 2 elements : 70
rdd1: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[12] at parallelize at command-3539668550948598:1
rdd2: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[13] at parallelize at command-3539668550948598:4
res9: Array[Unit] = Array((), (), (), (), (), (), ())

Command took 1.15 seconds -- by 500068760@stu.upes.ac.in at 5/20/2021, 4:42:27 PM on sparkCluster
```

Here we see that two rdds (rdd1 and rdd2) are executed and rdd1 is printed before rdd2 i.e **sequentially.**



```scala
val rdd1 = sc.parallelize(List(1,2,3,4,5,6,7))
rdd1.collect().map{x => println("List 1 elements : " + x); Thread.sleep(2000)}

val rdd2 = sc.parallelize(List(10,20,30,40,50,60,70))
rdd2.collect().map{x => println("List 2 elements : " + x)}
```

```
List 1 elements : 1
List 1 elements : 2
List 1 elements : 3
List 1 elements : 4
List 1 elements : 5
List 1 elements : 6
List 1 elements : 7
List 2 elements : 10
List 2 elements : 20
List 2 elements : 30
List 2 elements : 40
List 2 elements : 50
List 2 elements : 60
List 2 elements : 70
rdd1: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[14] at parallelize at command-4345903971373208:1
rdd2: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[15] at parallelize at command-4345903971373208:4
res10: Array[Unit] = Array((), (), (), (), (), (), ())

Command took 14.72 seconds -- by 500068760@stu.upes.ac.in at 5/20/2021, 4:42:59 PM on sparkCluster
```

Even if we increased the time to be executed in rdd1, it stills executes it first, so as spark executes the jobs sequentially it may take time and till then other process need to wait.

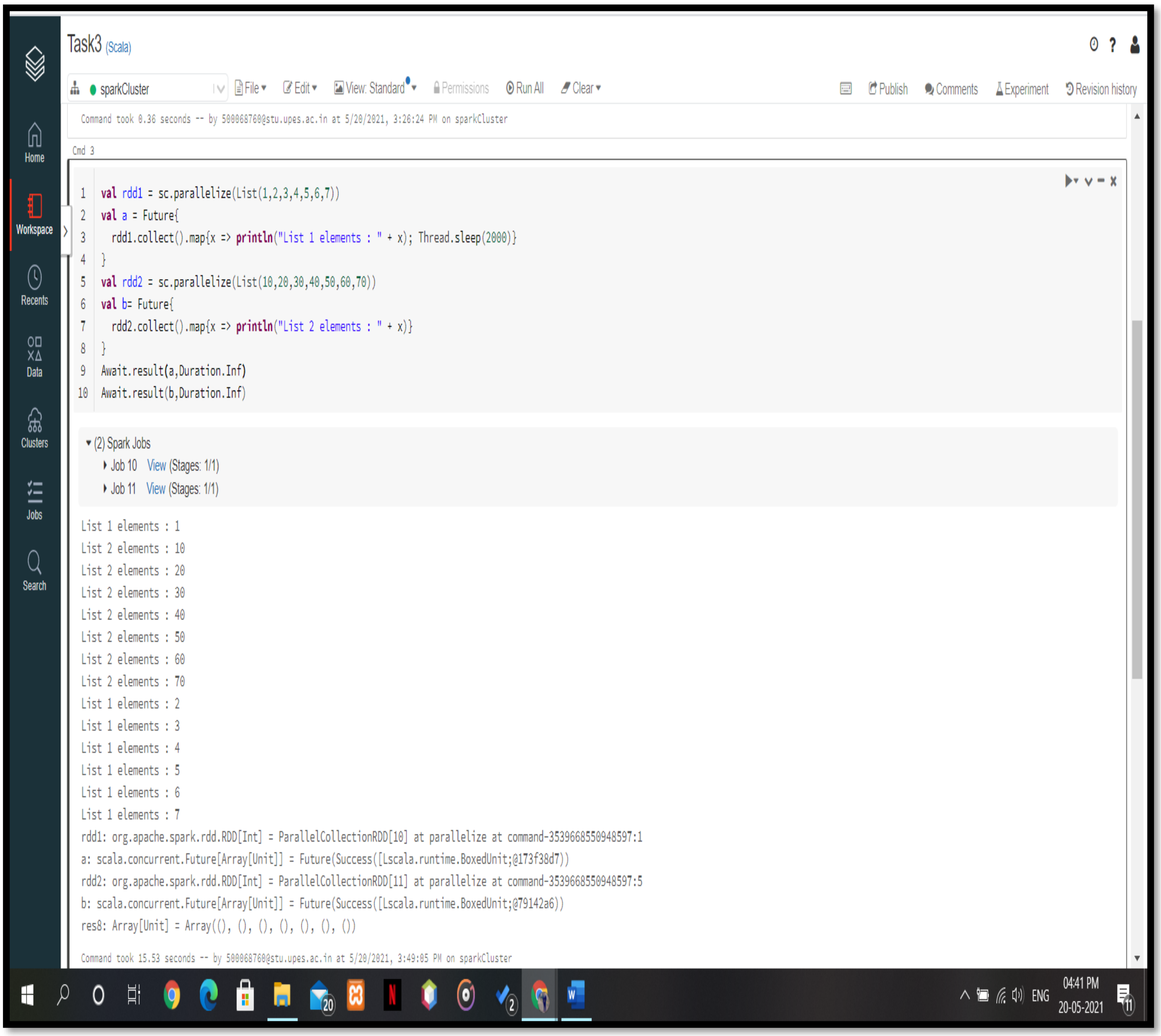
To achieve this concurreny of process in spark, using scala future objects we create thread for a particular task and that thread does not depend on other tasks and hence we get our parallelism.

This future objects need execution context which is available in below packages so we need to import them first and along with that we may have call-back functions in future to indicate success or failure so we will import that also.

- import scala.concurrent.{Future} // to import the future executioncontext
- import scala.concurrent.ExecutionContext.Implicits.global //
- import scala.util.{Failure, Success}
- import scala.concurrent.duration.Duration
- import scala.concurrent.{Await, ExecutionContext, Future}

await and duration is to give time to a thread and using duration we can specify the infinite time.

<u>Here comes the Future scala object:</u>



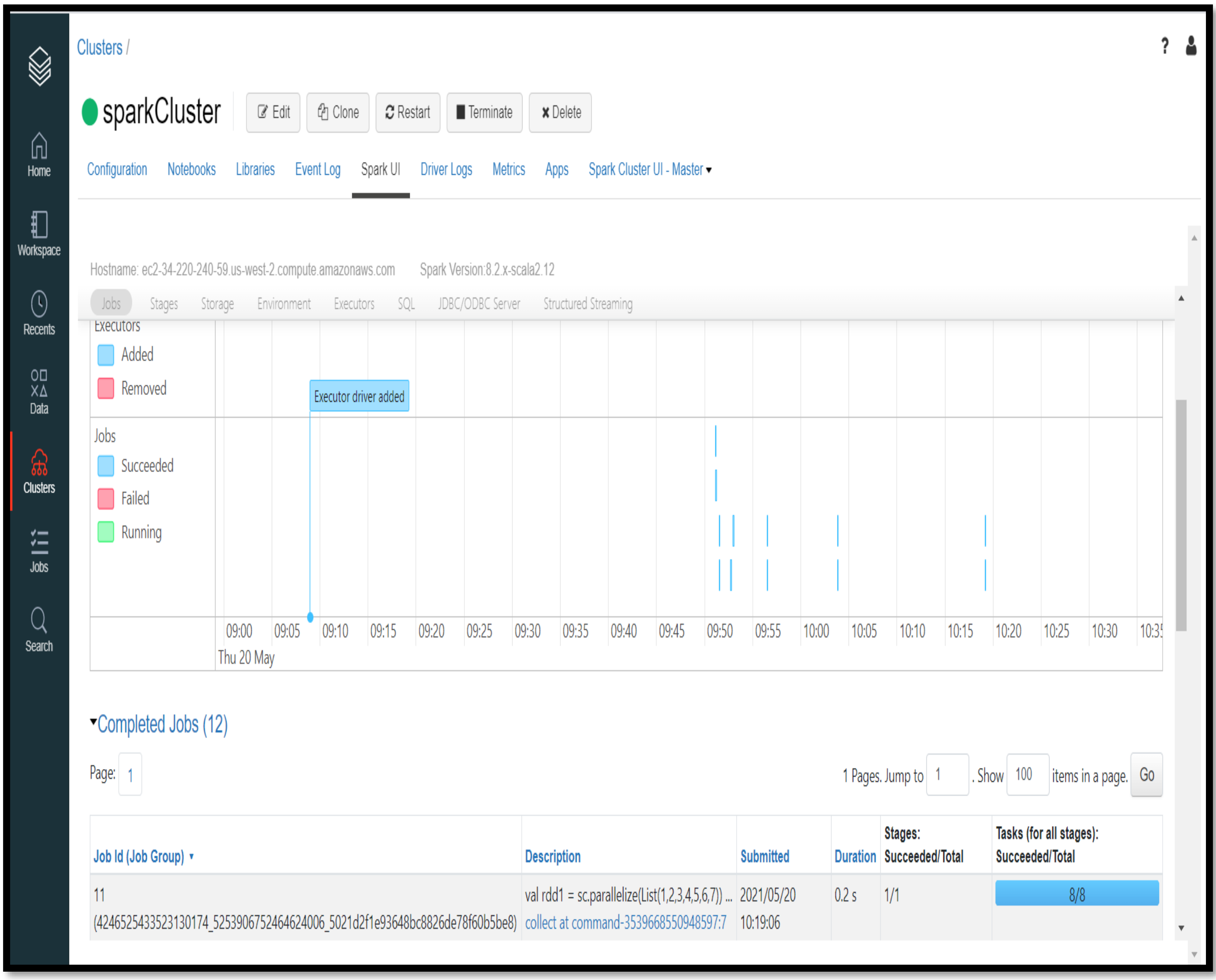
```
Command took 0.36 seconds -- by 500068760@stu.upes.ac.in at 5/20/2021, 3:26:24 PM on sparkCluster
```

```scala
val rdd1 = sc.parallelize(List(1,2,3,4,5,6,7))
val a = Future{
  rdd1.collect().map{x => println("List 1 elements : " + x); Thread.sleep(2000)}
}
val rdd2 = sc.parallelize(List(10,20,30,40,50,60,70))
val b= Future{
  rdd2.collect().map{x => println("List 2 elements : " + x)}
}
Await.result(a,Duration.Inf)
Await.result(b,Duration.Inf)
```

```
(2) Spark Jobs
  Job 10  View (Stages: 1/1)
  Job 11  View (Stages: 1/1)
List 1 elements : 1
List 2 elements : 10
List 2 elements : 20
List 2 elements : 30
List 2 elements : 40
List 2 elements : 50
List 2 elements : 60
List 2 elements : 70
List 1 elements : 2
List 1 elements : 3
List 1 elements : 4
List 1 elements : 5
List 1 elements : 6
List 1 elements : 7
rdd1: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[10] at parallelize at command-3539668550948597:1
a: scala.concurrent.Future[Array[Unit]] = Future(Success([Lscala.runtime.BoxedUnit;@173f38d7))
rdd2: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[11] at parallelize at command-3539668550948597:5
b: scala.concurrent.Future[Array[Unit]] = Future(Success([Lscala.runtime.BoxedUnit;@79142a6))
res8: Array[Unit] = Array((), (), (), (), (), (), ())

Command took 15.53 seconds -- by 500068760@stu.upes.ac.in at 5/20/2021, 3:49:05 PM on sparkCluster
```

Here we see that as the rdd1 takes 2 sec to print the next element of rdd and till the rdd2 executes its element so here we can see the resources are utilised and as the thread created is different from other so it does not affect other.

Here is the snapshot of executor where we can see that the jobs are performed independent of the other and the first one keeps iterating after every 2 sec break we gave.



Hostname: ec2-34-220-240-59.us-west-2.compute.amazonaws.com   Spark Version:8.2.x-scala2.12

Completed Jobs (12)

Job Id (Job Group) ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
11 (4246525433523130174_5253906752464624006_5021d2f1e93648bc8826de78f60b5be8)	val rdd1 = sc.parallelize(List(1,2,3,4,5,6,7)) ... collect at command-3539668550948597:7	2021/05/20 10:19:06	0.2 s	1/1	8/8