Terabytes at your Fingertips

Distributed coding at scale with PySpark

Sam Penrose

Sam Penrose[1], Data Engineer at Mozilla[2]

- Your ~~teacher~~ fellow student
- Does “data engineering” for Mozilla (distributed, terabyte-scale)

1. Sam / Mozilla: github.com/SamPenrose
2. Sam / life: www.sampenrose.net | twitter.com/sampenrose

We will learn

1. How distributed coding at scale (“DCS”) is **new**
2. The **three problems** you encounter and their **five root causes**
3. A **core technique** for interactive DCS coding

“Normal” coding

```
>>> def extract_and_transform(d):
...     value = d['field']
...     return float(value)
...
>>> def test():
...     data = [{'field': '1.1'}, {'field': '1.02'}]
...     expected = [1.1, 1.02]
...     actual = map(extract_and_transform, data)
...     assert actual == expected
...
>>> test()
>>>
```

Tests pass—ship it!

```
>>> import json
>>> with open('production_data.json') as f:
...     production_data = json.load(f)
...
>>> loaded = []
>>> for d in production_data:
...     value = extract_and_transform(d)
...     loaded.append(value)
...
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
  File "<stdin>", line 3, in extract_and_transform
TypeError: float() argument must be a string or a number
```

Let’s debug interactively

```
>>> def debug_etl_with_print(d):
...     value = d['field']
...     print "value is:", value
...     return float(value)
...
>>> for d in production_data:
...     _ = debug_etl_with_print(d)
...
value is: 1.07
value is: 1.13
value is: None
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
  File "<stdin>", line 4, in debug_etl_with_print
TypeError: float() argument must be a string or a number
```

What did we just do?

1. Noticed a problem.
2. **Displayed the program's state**.
3. Identified the problem.

How old is this technique?

1989: Python

1970: Unix shell

1962: Lisp REPL

1951: add print statements

Ritchie and Thompson(sitting) at PDP-11 in Bell Labs

No, really: 1951

5-2 Location of mistakes in a program.

It might be thought that a good way of finding errors in a program would be to make the machine proceed order by order under the control of the "Single E.P." button (see Section 6-4), and to study the numbers in the machine by watching the monitors attached to the arithmetical unit and store. This, however, usually turns out to be a very slow and inefficient process, especially as the numbers are displayed in binary form. Methods have therefore been developed which permit the machine to proceed unhindered by the operator, whilst printing on the teleprinter a permanent record that can be studied at leisure, and that will assist in understanding the nature of the mistake.

One such method is to wait until the machine has stopped (or to stop it deliberately) and then, without clearing the whole store, to insert (by pressing the starter button again) a small program which will print, in suitable form, the contents of part of the store. This has come to be known as the "post-mortem" method. Tapes are kept available near the EDSAC for printing the function letters, or address parts, of orders in consecutive storage locations. Programmers may also prepare their own post-mortem tapes.

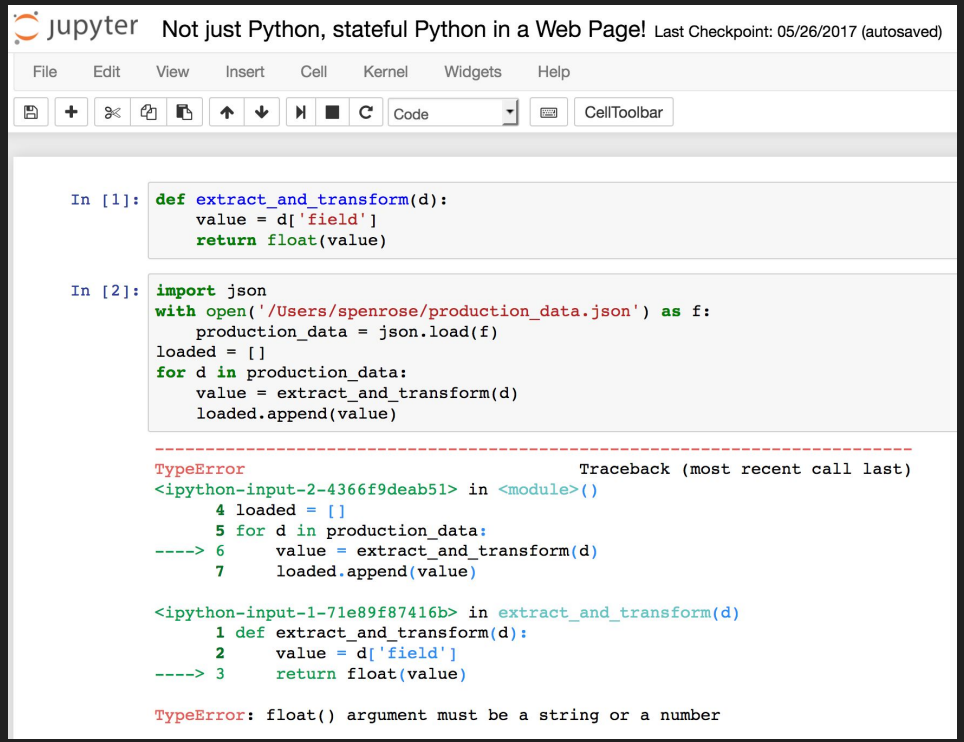
This method yields only a static picture of the store as it was when the calculation stopped. Other methods have been derived to provide information about the whole course of the calculation. These necessarily involve modifying the program to cause the extra printing, and therefore a new tape must be prepared and presented to the machine. This, however, is no hardship, since the machine will read an average tape in about a minute, and the preparation need not take more than a few minutes of the programmer's time.

5-21 Method using extra output orders. One simple and very useful plan is to place an output order at the beginning of the master routine and in front of each subroutine so that the completion of the various stages of the program

The Preparation of Programs for an Electronic Digital Computer, Maurice Wilkes, 1951

Notebooks: brave new (?) world

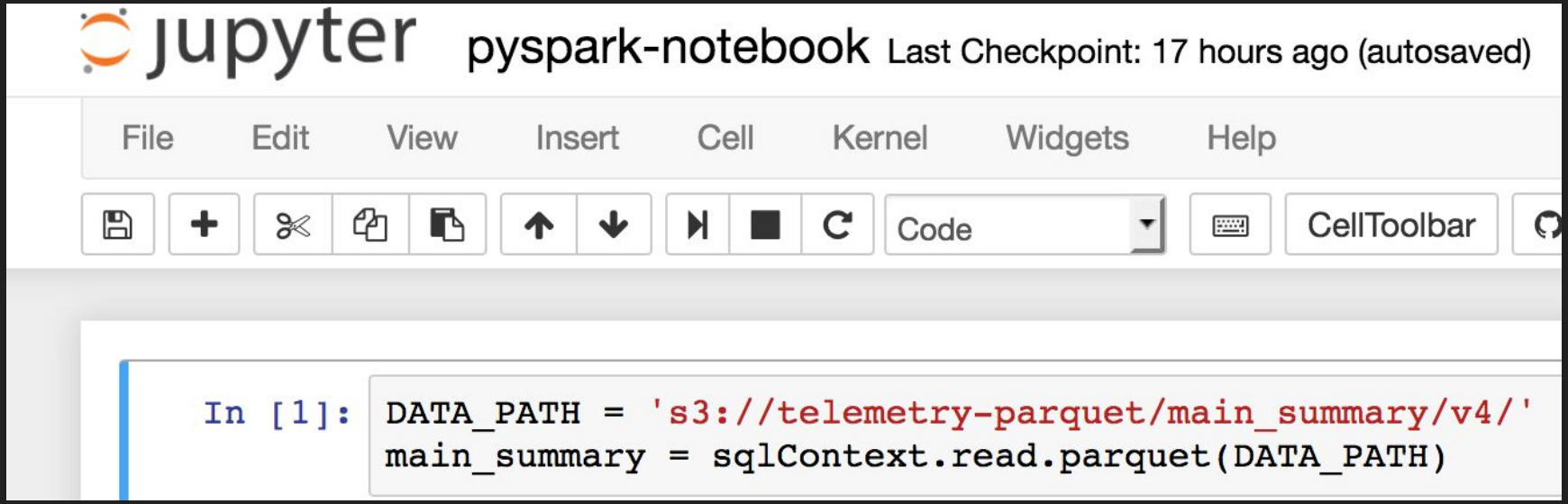
```
>>> import json
>>> with open('production_data.json') as f:
...     production_data = json.load(f)
...
>>> loaded = []
>>> for d in production_data:
...     value = extract_and_transform(d)
...     loaded.append(value)
...
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
  File "<stdin>", line 3, in extract_and_transform
TypeError: float() argument must be a string or a number
```



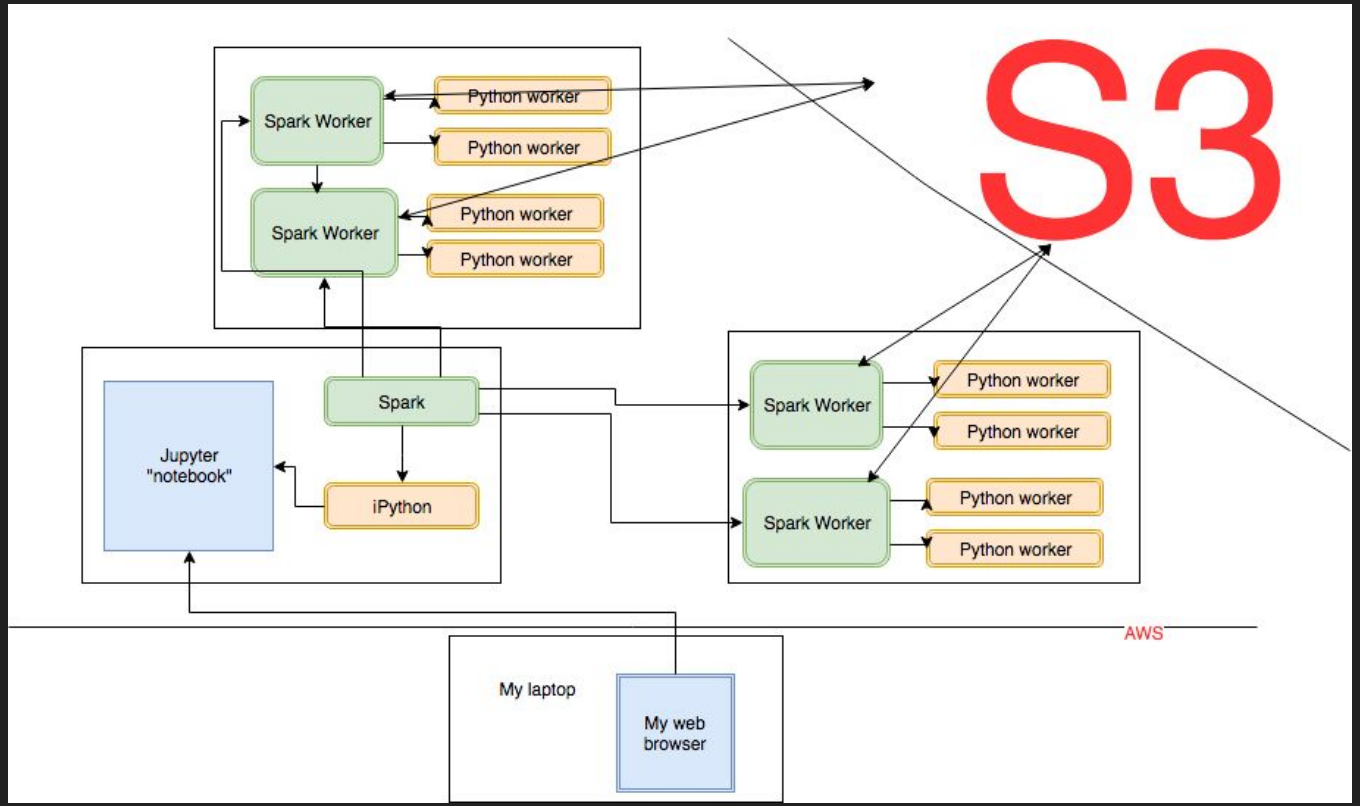
Coding: 1951 -> 201x

1. Run
2. Observe
    a. By print() if necessary
3. Edit
    a. GOTO: 1

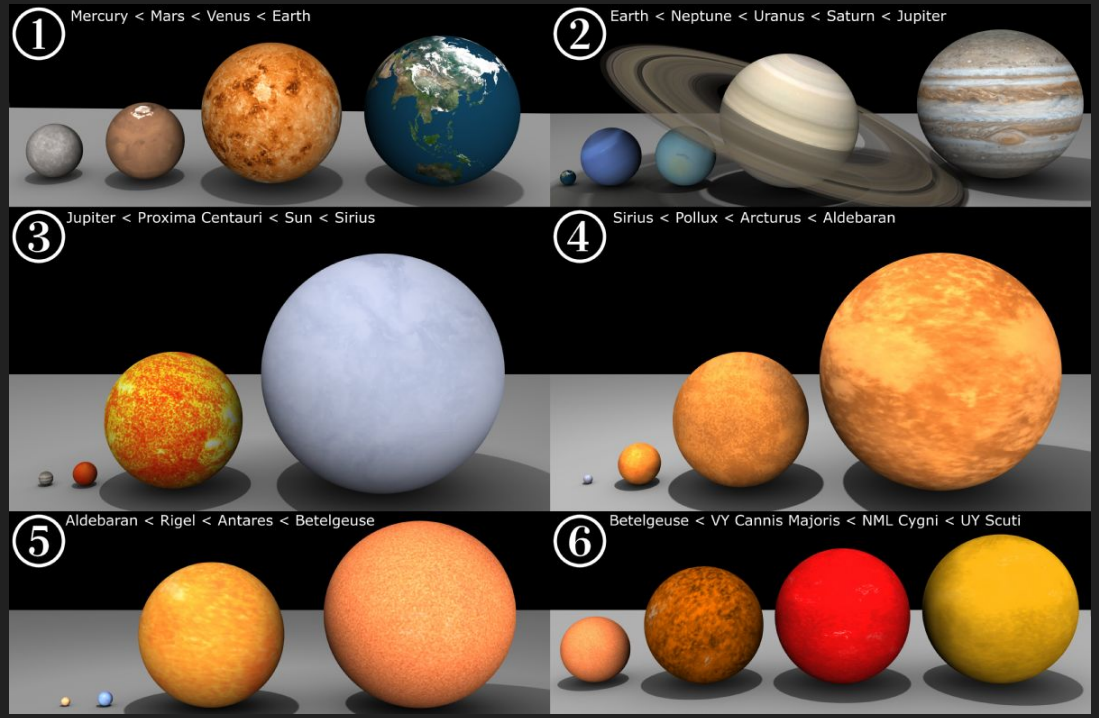
2017: 2 lines -> 500TB on 30 nodes



Everything has changed … except the UI

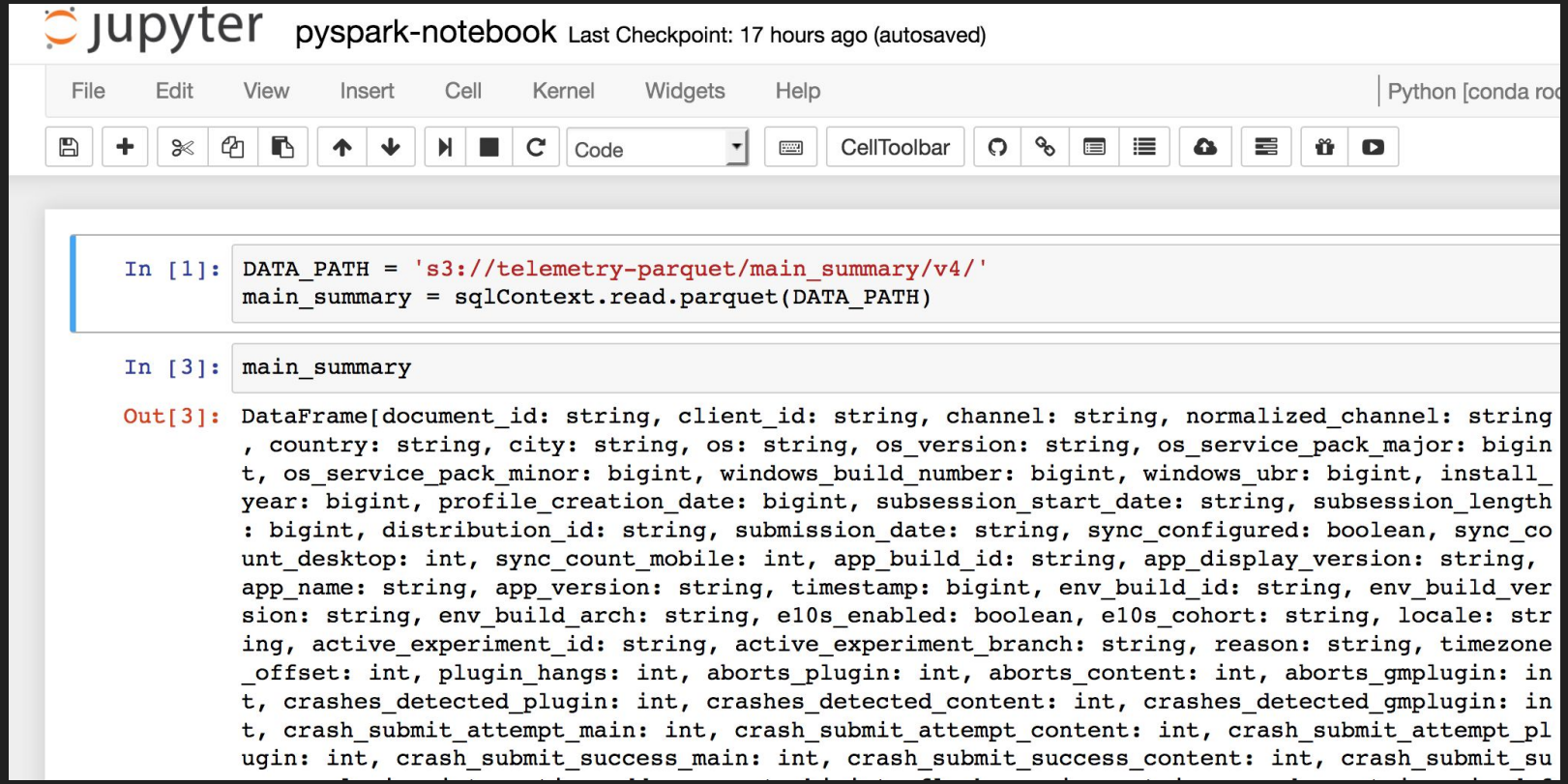


Our minds don’t do scale



“Relative sizes of the planets in the solar system and some of the largest stars” by Jcpag2012

Not Python (or Scala), not a variable, not printable



Enormously powerful pseudo-statements

```
In [5]: from datetime import date, timedelta
        week_ago = date.today() - timedelta(days=7)
        start = week_ago.isoformat()
        clause = "submission_date > '{}'".format(start)
        recent = main_summary.where(clause)

In [6]: recent.count()

Out[6]: 53866405276
```

Well invalidate my priors and call me perplexed!

```
In [5]: from datetime import date, timedelta
        week_ago = date.today() - timedelta(days=7)
        start = week_ago.isoformat()
        clause = "submission_date > '{}'".format(start)
        recent = main_summary.where(clause)

In [6]: recent.count()

Out[6]: 53866405276

In [7]: main_summary.count()

Out[7]: 164284288426

        Wait, 1/3 of data in last week?
        Something's wrong.
```

Using the DCS[1] prompt: take()

```
Out[18]: [Row(active_experiment_id=u'e10s-enabled-beta-20151214@experiments.mozilla.org'),
          Row(active_experiment_id=u'e10s-beta45-withoutaddons@experiments.mozilla.org'),
          Row(active_experiment_id=u'e10s-beta45-withoutaddons@experiments.mozilla.org'),
          Row(active_experiment_id=u'e10s-beta45-withoutaddons@experiments.mozilla.org'),
          Row(active_experiment_id=u'e10s-beta45-withoutaddons@experiments.mozilla.org'),
          Row(active_experiment_id=u'e10s-beta45-withoutaddons@experiments.mozilla.org'),
          Row(active_experiment_id=u'e10s-beta45-withoutaddons@experiments.mozilla.org'),
          Row(active_experiment_id=u'e10s-beta45-withoutaddons@experiments.mozilla.org'),
          Row(active_experiment_id=u'e10s-beta45-withoutaddons@experiments.mozilla.org'),
          Row(active_experiment_id=u'e10s-beta45-withoutaddons@experiments.mozilla.org')]

In [22]: e10s = recent.rdd.filter(lambda row: row['active_experiment_id'].startswith('e10s'))
         ten_e10s = e10s.take(10)
```

1. “DCS”: distributed computing at scale

Tracebacks at the DCS prompt (1)

```
Out[18]: [Row(active_experiment_id=u'e10s-enabled-beta-20151214@experiments.mozilla.org'),
          Row(active_experiment_id=u'e10s-beta45-withoutaddons@experiments.mozilla.org'),
          Row(active_experiment_id=u'e10s-beta45-withoutaddons@experiments.mozilla.org'),
          Row(active_experiment_id=u'e10s-beta45-withoutaddons@experiments.mozilla.org'),
          Row(active_experiment_id=u'e10s-beta45-withoutaddons@experiments.mozilla.org'),
          Row(active_experiment_id=u'e10s-beta45-withoutaddons@experiments.mozilla.org'),
          Row(active_experiment_id=u'e10s-beta45-withoutaddons@experiments.mozilla.org'),
          Row(active_experiment_id=u'e10s-beta45-withoutaddons@experiments.mozilla.org'),
          Row(active_experiment_id=u'e10s-beta45-withoutaddons@experiments.mozilla.org'),
          Row(active_experiment_id=u'e10s-beta45-withoutaddons@experiments.mozilla.org')]
```

click to expand output; double click to hide output

```
In [22]: e10s = recent.rdd.filter(lambda row: row['active_experiment_id'].startswith('e10s'))
         ten_e10s = e10s.take(10)
```

```
Py4JJavaErrorTraceback (most recent call last)
<ipython-input-22-334b6583877b> in <module>()
      1 e10s = recent.rdd.filter(lambda row: row['active_experiment_id'].startswith('e10s'))
----> 2 ten_e10s = e10s.take(10)
```

Tracebacks at the DCS prompt (2)

```
In [22]: el0s = recent.rdd.filter(lambda row: row['active_experiment_id'].startswith('el0s'))
         ten_el0s = el0s.take(10)

Py4JJavaErrorTraceback (most recent call last)
<ipython-input-22-324b6583877b> in <module>()
      1 el0s = recent.rdd.filter(lambda row: row['active_experiment_id'].startswith('el0s'))
----> 2 ten_el0s = el0s.take(10)

/usr/lib/spark/python/pyspark/rdd.py in take(self, num)
   1308
   1309             p = range(partsScanned, min(partsScanned + numPartsToTry, totalParts))
-> 1310             res = self.context.runJob(self, takeUpToNumLeft, p)
   1311
   1312             items += res

/usr/lib/spark/python/pyspark/context.py in runJob(self, rdd, partitionFunc, partitions, allowLocal)
    931         # SparkContext#runJob.
    932         mappedRDD = rdd.mapPartitions(partitionFunc)
--> 933         port = self._jvm.PythonRDD.runJob(self._jsc.sc(), mappedRDD._jrdd, partitions)
    934         return list(_load_from_socket(port, mappedRDD._jrdd_deserializer))
    935

/usr/lib/spark/python/lib/py4j-0.10.3-src.zip/py4j/java_gateway.py in __call__(self, *args)
   1131         answer = self.gateway_client.send_command(command)
   1132         return_value = get_return_value(
-> 1133             answer, self.gateway_client, self.target_id, self.name)
   1134
   1135         for temp_arg in temp_args:

/usr/lib/spark/python/pyspark/sql/utils.py in deco(*a, **kw)
     61     def deco(*a, **kw):
     62         try:
---> 63             return f(*a, **kw)
     64         except py4j.protocol.Py4JJavaError as e:
     65             s = e.java_exception.toString()

/usr/lib/spark/python/lib/py4j-0.10.3-src.zip/py4j/protocol.py in get_return_value(answer, gateway_client, target_id,
name)
    317                 raise Py4JJavaError(
    318                     "An error occurred while calling {0}{1}{2}.\n".
--> 319                     format(target_id, ".", name), value)
    320             else:
    321                 raise Py4JError(

Py4JJavaError: An error occurred while calling z:org.apache.spark.api.python.PythonRDD.runJob.
: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 10.0 failed 4 times, most recent
failure: Lost task 0.3 in stage 10.0 (TID 290479, ip-172-31-16-157.us-west-2.compute.internal): org.apache.spark.api.
python.PythonException: Traceback (most recent call last):
  File "/mnt/yarn/usercache/hadoop/appcache/application_1495754612025_0001/container_1495754612025_0001_01_000211/pys
park.zip/pyspark/worker.py", line 172, in main
    process()
  File "/mnt/yarn/usercache/hadoop/appcache/application_1495754612025_0001/container_1495754612025_0001_01_000211/pys
park.zip/pyspark/worker.py", line 167, in process
    serializer.dump_stream(func(split_index, iterator), outfile)
  File "/mnt/yarn/usercache/hadoop/appcache/application_1495754612025_0001/container_1495754612025_0001_01_000211/pys
park.zip/pyspark/serializers.py", line 263, in dump_stream
    vs = list(itertools.islice(iterator, batch))
  File "/usr/lib/spark/python/pyspark/rdd.py", line 1306, in takeUpToNumLeft
  File "<ipython-input-22-324b6583877b>", line 1, in <lambda>
AttributeError: 'NoneType' object has no attribute 'startswith'

        at org.apache.spark.api.python.PythonRunner$$anon$1.read(PythonRDD.scala:193)
        at org.apache.spark.api.python.PythonRunner$$anon$1.<init>(PythonRDD.scala:234)
        at org.apache.spark.api.python.PythonRunner.compute(PythonRDD.scala:152)
        at org.apache.spark.api.python.PythonRDD.compute(PythonRDD.scala:63)
        at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:319)
        at org.apache.spark.rdd.RDD.iterator(RDD.scala:283)
        at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:70)
        at org.apache.spark.scheduler.Task.run(Task.scala:86)
        at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:274)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
        at java.lang.Thread.run(Thread.java:745)

Driver stacktrace:
        at org.apache.spark.scheduler.DAGScheduler.org$apache$spark$scheduler$DAGScheduler$$failJobAndIndependentStag
es(DAGScheduler.scala:1454)
        at org.apache.spark.scheduler.DAGScheduler$$anonfun$abortStage$1.apply(DAGScheduler.scala:1442)
        at org.apache.spark.scheduler.DAGScheduler$$anonfun$abortStage$1.apply(DAGScheduler.scala:1441)
        at scala.collection.mutable.ResizableArray$class.foreach(ResizableArray.scala:59)
        at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:48)
        at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:1441)
        at org.apache.spark.scheduler.DAGScheduler$$anonfun$handleTaskSetFailed$1.apply(DAGScheduler.scala:811)
        at org.apache.spark.scheduler.DAGScheduler$$anonfun$handleTaskSetFailed$1.apply(DAGScheduler.scala:811)
        at scala.Option.foreach(Option.scala:257)
        at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:811)
        at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:1667)
        at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:1622)
        at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:1611)
        at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:48)
        at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:632)
        at org.apache.spark.SparkContext.runJob(SparkContext.scala:1873)
        at org.apache.spark.SparkContext.runJob(SparkContext.scala:1886)
        at org.apache.spark.SparkContext.runJob(SparkContext.scala:1899)
        at org.apache.spark.api.python.PythonRDD.runJob(PythonRDD.scala:441)
        at org.apache.spark.api.python.PythonRDD.runJob(PythonRDD.scala)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:237)
        at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)
        at py4j.Gateway.invoke(Gateway.java:280)
        at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)
        at py4j.commands.CallCommand.execute(CallCommand.java:79)
        at py4j.GatewayConnection.run(GatewayConnection.java:214)
        at java.lang.Thread.run(Thread.java:745)
Caused by: org.apache.spark.api.python.PythonException: Traceback (most recent call last):
  File "/mnt/yarn/usercache/hadoop/appcache/application_1495754612025_0001/container_1495754612025_0001_01_000211/pys
park.zip/pyspark/worker.py", line 172, in main
    process()
  File "/mnt/yarn/usercache/hadoop/appcache/application_1495754612025_0001/container_1495754612025_0001_01_000211/pys
park.zip/pyspark/worker.py", line 167, in process
    serializer.dump_stream(func(split_index, iterator), outfile)
  File "/mnt/yarn/usercache/hadoop/appcache/application_1495754612025_0001/container_1495754612025_0001_01_000211/pys
park.zip/pyspark/serializers.py", line 263, in dump_stream
    vs = list(itertools.islice(iterator, batch))
  File "/usr/lib/spark/python/pyspark/rdd.py", line 1306, in takeUpToNumLeft
  File "<ipython-input-22-324b6583877b>", line 1, in <lambda>
AttributeError: 'NoneType' object has no attribute 'startswith'

        at org.apache.spark.api.python.PythonRunner$$anon$1.read(PythonRDD.scala:193)
        at org.apache.spark.api.python.PythonRunner$$anon$1.<init>(PythonRDD.scala:234)
        at org.apache.spark.api.python.PythonRunner.compute(PythonRDD.scala:152)
        at org.apache.spark.api.python.PythonRDD.compute(PythonRDD.scala:63)
        at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:319)
        at org.apache.spark.rdd.RDD.iterator(RDD.scala:283)
        at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:70)
        at org.apache.spark.scheduler.Task.run(Task.scala:86)
        at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:274)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
        ... 1 more
```

Tracebacks at the DCS prompt (3)

```
  File "<ipython-input-22-334b6583877b>", line 1, in <lambda>
AttributeError: 'NoneType' object has no attribute 'startswith'
```

This reminds me of something ...

```
  File "<ipython-input-22-334b6583877b>", line 1, in <lambda>
AttributeError: 'NoneType' object has no attribute 'startswith'
```

```
>>> import json
>>> with open('production_data.json') as f:
...     production_data = json.load(f)
...
>>> loaded = []
>>> for d in production_data:
...     value = extract_and_transform(d)
...     loaded.append(value)
...
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
  File "<stdin>", line 3, in extract_and_transform
TypeError: float() argument must be a string or a number
```

Root cause #1: data variance

In [16]:
```
def is_e10s_experiment(row):
    if row['active_experiment_id'] is None:
        return False
    return row['active_experiment_id'].startswith('e10s')
e10s = recent.rdd.filter(is_e10s_experiment)
ten_e10s = e10s.take(10)
```

In [17]:
```
ten_e10s[0]['active_experiment_id']
```

Out[17]:
```
u'e10s-beta45-withoutaddons@experiments.mozilla.org'
```

Why was I able to debug that?

Did I display the program state? **No.**

Ritchie and Thompson(sitting) at PDP-11 in Bell Labs

What about the previous issue?

```
In [5]: from datetime import date, timedelta
        week_ago = date.today() - timedelta(days=7)
        start = week_ago.isoformat()
        clause = "submission_date > '{}'".format(start)
        recent = main_summary.where(clause)

In [6]: recent.count()

Out[6]: 53866405276

In [7]: main_summary.count()

Out[7]: 164284288426

        Wait, 1/3 of data in last week?
        Something's wrong.
```

Maybe I can do something about that, too!

```
Wait, 1/3 of data in last week?
Something's wrong.
```

```
In [8]: tinytinylittleeensyweensybit = main_summary.take(10)
```

```
In [9]: for row in tinytinylittleeensyweensybit:
            print row['submission_date'],
```

```
20160620 20160620 20160620 20160620 20160620 20160620 20160620 201
```

```
In [10]: start
```

```
Out[10]: '2017-05-19'
```

```
In [11]: properly_formatted = week_ago.isoformat().replace('-', '')
         properly_formatted
```

```
Out[11]: '20170519'
```

```
In [12]: clause = "submission_date > '{}'".format(properly_formatted)
         recent = main_summary.where(clause)
```

```
In [13]: recent.count()
```

```
Out[13]: 2119866345
```

Why was I able to debug these two problems?

1. ~~I displayed the DCS **program state**.~~
2. In the first case, the **traceback** displayed the **problem**.
3. In the second case, the **problem** was displayable in **every row**.

The core technique*

1. **Localize** the **problem** state to observe it.

*part one

Problem states, localized and observed

```
File "<ipython-input-22-334b6583877b>", line 1, in <lambda>
AttributeError: 'NoneType' object has no attribute 'startswith'
```

Wait, 1/3 of data in last week?
Something's wrong.

In [8]:
```
tinytinylittleeensyweensybit = main_summary.take(10)
```

In [9]:
```
for row in tinytinylittleeensyweensybit:
    print row['submission_date'],
```

```
20160620 20160620 20160620 20160620 20160620 20160620 20160620 201
```

In [10]:
```
start
```

Out[10]:
```
'2017-05-19'
```

In [11]:
```
properly_formatted = week_ago.isoformat().replace('-', '')
properly_formatted
```

Out[11]:
```
'20170519'
```

In [12]:
```
clause = "submission_date > '{}'".format(properly_formatted)
recent = main_summary.where(clause)
```

In [13]:
```
recent.count()
```

Out[13]:
```
2119866345
```

Modularize fix, because software engineering

```
In [16]: def is_e10s_experiment(row):
             if row['active_experiment_id'] is None:
                 return False
             return row['active_experiment_id'].startswith('e10s')
         e10s = recent.rdd.filter(is_e10s_experiment)
         ten_e10s = e10s.take(10)

In [17]: ten_e10s[0]['active_experiment_id']

Out[17]: u'e10s-beta45-withoutaddons@experiments.mozilla.org'
```

hadoop@i...-31-0-97:~ | bash ... | java

File Edit Options Buffers Tools Python Help

```
def is_e10s_experiment(row):
    if row['active_experiment_id'] is None:
        return False
    return row['active_experiment_id'].startswith('e10s')
```

Test, because software engineering

```
In [19]: # Modularize and test module locally
         import my_local_utilities
         worked = filter(my_local_utilities.is_e10s_experiment,
                         tinytinylittleeensyweensybit) == []
         worked

Out[19]: True
```

Engineering problem (1)

```
In [20]: # Now distribute *the test data* and try the code on it.
         tiny_distributed = sc.parallelize(tinytinylittleeensyweensybit)
         works_distributed = tiny_distributed.filter(
             my_local_utilities.is_e10s_experiment).collect() == []
```

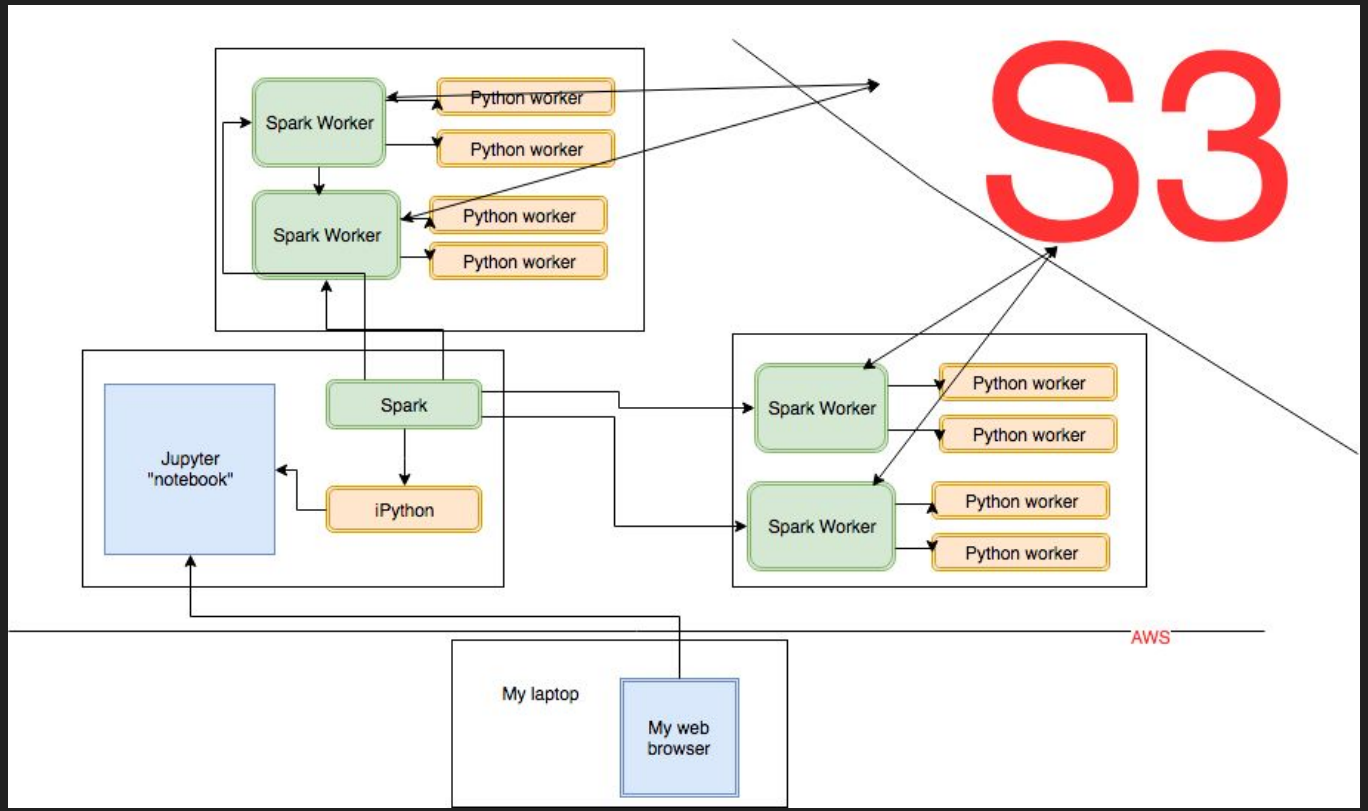
```
Py4JJavaErrorTraceback (most recent call last)
<ipython-input-20-db53628f212c> in <module>()
      2 tiny_distributed = sc.parallelize(tinytinylittleeensyweensybit)
      3 works_distributed = tiny_distributed.filter(
----> 4     my_local_utilities.is_e10s_experiment).collect() == []
```

click to scroll output; double click to hide

Engineering problem (2)

```
Py4JJavaError: An error occurred while calling z:
.
: org.apache.spark.SparkException: Job aborted du
led 4 times, most recent failure: Lost task 0.3 i
s-west-2.compute.internal): org.apache.spark.api.
nt call last):
  File "/mnt/yarn/usercache/hadoop/appcache/appli
12025_0001_01_000164/pyspark.zip/pyspark/worker.p
    func, profiler, deserializer, serializer = re
  File "/mnt/yarn/usercache/hadoop/appcache/appli
12025_0001_01_000164/pyspark.zip/pyspark/worker.p
    command = serializer._read_with_length(file)
  File "/mnt/yarn/usercache/hadoop/appcache/appli
12025_0001_01_000164/pyspark.zip/pyspark/serializ
    return self.loads(obj)
  File "/mnt/yarn/usercache/hadoop/appcache/appli
12025_0001_01_000164/pyspark.zip/pyspark/serializ
    return pickle.loads(obj)
ImportError: No module named my_local_utilities
```

I just imported it … locally



Root cause #2: heterogenous framework state

```
In [19]: # Modularize and test module locally
         import my_local_utilities
         worked = filter(my_local_utilities.is_e10s_experiment,
                         tinytinylittleeensyweensybit) == []
         worked

Out[19]: True
```

```
In [20]: # Now distribute *the test data* and try the code on it.
         tiny_distributed = sc.parallelize(tinytinylittleeensyweensybit)
         works_distributed = tiny_distributed.filter(
             my_local_utilities.is_e10s_experiment).collect() == []

Py4JJavaErrorTraceback (most recent call last)
<ipython-input-20-db53628f212c> in <module>()
      2 tiny_distributed = sc.parallelize(tinytinylittleeensyweensybit)
      3 works_distributed = tiny_distributed.filter(
----> 4     my_local_utilities.is_e10s_experiment).collect() == []
```

click to scroll output; double click to hide

The core technique, completed

1. **Localize problem** state to observe it.
2. Scale out and up in **steps**.

Scaling out and up in steps

1. Get the code working with a small local dataset.
2. Scale **out**: distribute **that small local dataset** and **that code**. Still works?
3. Scale **up**: iteratively run that code on more and more of the full dataset.
    a. E.g. 1%, 10%, 20%, 40%
    b. On each loop, look for a **scale fail**.

Mo’ data, mo’ problems

Three flavors of scale fail:

1. Traceback reporting a **your-code-vs-your-data** problem.
2. Exponential **slowdown**.
3. Traceback reporting a **toolchain panic**.

Your code vs. your data: “only bring me problems”

In [14]:
```
def fragile_is_e10s_experiment(row):
    """I will raise an AttributeError."""
    return row['active_experiment_id'].startswith('e10s')
def is_this_row_naughty(row):
    try:
        row['active_experiment_id'].startswith('e10s')
    except AttributeError:
        return True # or row
    return False
problem_rows = recent.rdd.filter(is_this_row_naughty)
problem_made_locally_observable = problem_rows.take(10)
```

Root cause #3: mental models of control flow

```
def to_sessions((client_id, pinglist)):
    client = {
        'client_id': client_id,
        'known_bad_pings': [],
        'sessions': [],
        'breaks_by_psc': 0,
        'breaks_by_ssid': 0,
        'breaks_by_sid': 0
        }
    def mark_is_finished(session):
        try:
            session['to_end'] = session['pings'][-1]['meta']['reason'] in ("shutdown", "aborted-session")
            session['complete'] = session['from_start'] and session['to_end']
        except Exception:
            session['to_end'] = False
            session['complete'] = False
    def make_session(first, previousSessionId=None):
        session = {'pings': [first]}
        try:
            session['from_start'] = first.get('payload', {}).get('info', {}).get(
                'subsessionCounter') == 1
        except Exception:
            session['from_start'] = False
        mark_is_finished(session)
        if previousSessionId:
            if first['payload']['info']['previousSessionId'] != previousSessionId:
                client['breaks_by_sid'] += 1
        return session
    def psc(d):
        result = -99
        try:
            result = d.get('payload', {}).get('info', {}).get(
                'profileSubsessionCounter', result)
        except Exception:
            pass
        return result
    pinglist.sort(key=psc)
    previous = pinglist[0]
    session = make_session(previous)
    if not pinglist[1:]:
        mark_is_finished(session)
        client['sessions'].append(session)
    for d in pinglist[1:]:
        try:
            if previous['payload']['info']['profileSubsessionCounter'] + 1 != \
                d['payload']['info']['profileSubsessionCounter']:
                client['breaks_by_psc'] += 1
            if previous['payload']['info']['subsessionId'] != \
                d['payload']['info']['previousSubsessionId']:
                client['breaks_by_ssid'] += 1
            if previous['payload']['info']['sessionId'] == \
                d['payload']['info']['sessionId']:
                session['pings'].append(d)
            else:
                mark_is_finished(session)
                client['sessions'].append(session)
                session = make_session(d, previous['payload']['info']['sessionId'])
            previous = d
        except Exception:
            client['known_bad_pings'].append(d)

    return client
```

If the results surprise you, quantify the control flow

You have a mental model of your code's control flow.

Localize and observe it to see if it is the problem.

In [21]:
```
def quantify_control_flow():
    almost_every_time = sc.accumulator(0)
    once_in_a_blue_moon = sc.accumulator(0)
    def process(row):
        if row['active_experiment_id'] is not None:
            almost_every_time.add(1)
            pass # Do the real work.
        else:
            once_in_a_blue_moon.add(1)
            pass
    return process, almost_every_time, once_in_a_blue_moon
quantifer, almost_every_time, once_in_a_blue_moon = quantify_control_flow()
```

In [22]:
```
_ = recent.rdd.map(quantifer).collect()
```

In [23]:
```
print almost_every_time # -> "should" be huge, is tiny
print once_in_a_blue_moon # "should" be tiny, is huge
```

```
12781730
2107084615
```

Slowness

Two new root causes:

- #4: exponential behavior
- #5: heavy load (high ratio of data to computational resources)

Reallocate

1. Are you comparing every row to every row?
2. Yes: you are (probably) running an exponential algorithm.
    a. First step: repartition on the comparison key(s).
    b. (Make sure the algorithm lacks “inner loop: fetch_across_network()”)
3. No: see if the data is skewed

Repartition on your comparison key

```
def partitioner(d):
    return hash(d.get('meta', d).get('clientId', 'MISSING'))
def get_key(d):
    return d.get('meta', d).get('clientId', 'MISSING')
PARTITION_FACTOR = 10
def group_by_client_id(rdd):
    size = rdd.getNumPartitions() * PARTITION_FACTOR
    rdd.repartitionAndSortWithinPartitions(size, partitioner, True, get_key)
    return rdd.groupBy(lambda d: d.get('clientId', 'MISSING'))
```

Divide and conquer

1. Is your processing function associative?
2. If so, turn that frown upside down!
    a. data*2 -> delay*4 :-(
    b. data/2 -> delay/4 :-)!

Sample

1. 10^6 rows: 800,000 ‘a’, 200,000 ‘b’ or ‘c’
2. Take 10% sample -> 10^5 rows
    a. “What is the ratio of ‘a’ to (‘b’ or ‘c’)?” -> “About 4:1.”
    b. Sampling worked great!
3. 10^1 rows: [a, a, a, c, a, b, a, a, a, a]
    a. Take 10% sample: useless.
4. More information:
    a. ~~Read an article.~~
    b. ~~Read a book.~~
    c. Take a class.

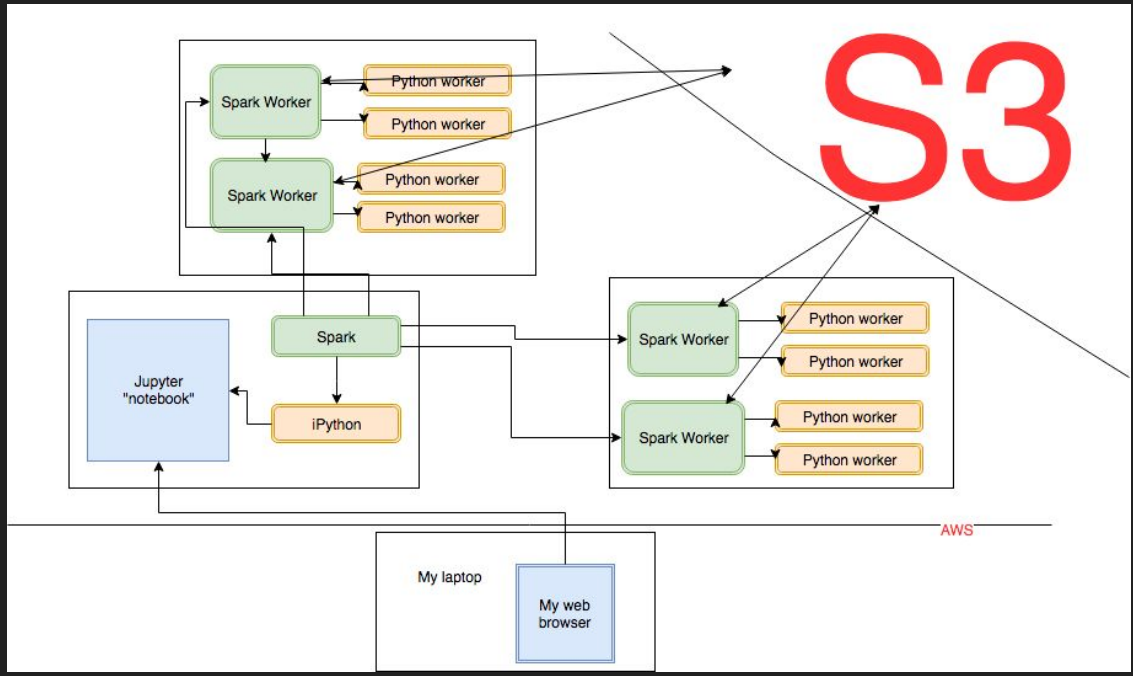
Spend

Toolchain panics: recognition

- **Py4J**NetworkError: An error occurred while trying to connect to the Java server
- **org.apache.spark**.SparkException: Job 330 cancelled; SparkContext was shut down
- **org.apache.spark**.sql.catalyst.errors.package$TreeNodeException
- **org.apache.spark**.SparkException: Kryo serialization failed: Buffer overflow.

Toolchain panics: recovery

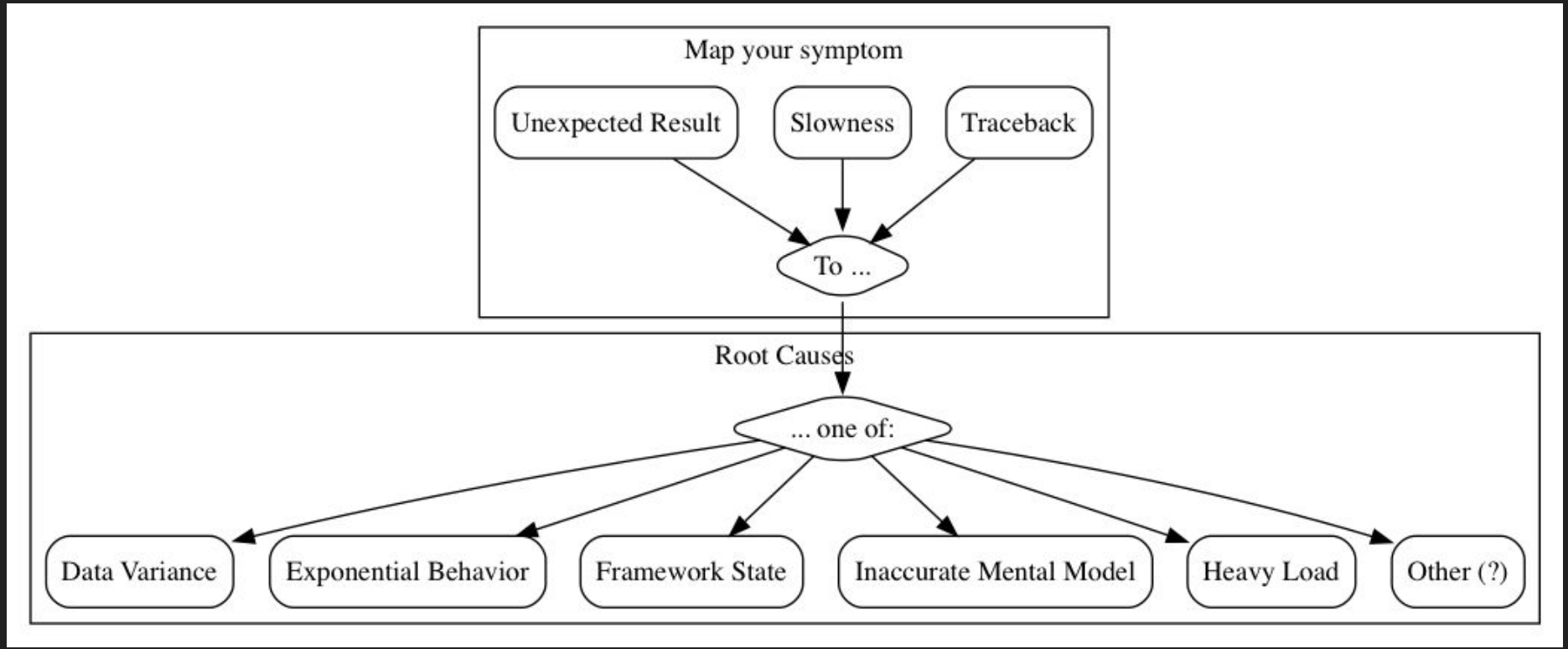
- **Lighten** the load
  - Reallocate, sample, ...
- Check toolchain **edges**
  - Spark console, etc.



Synopsis (1)

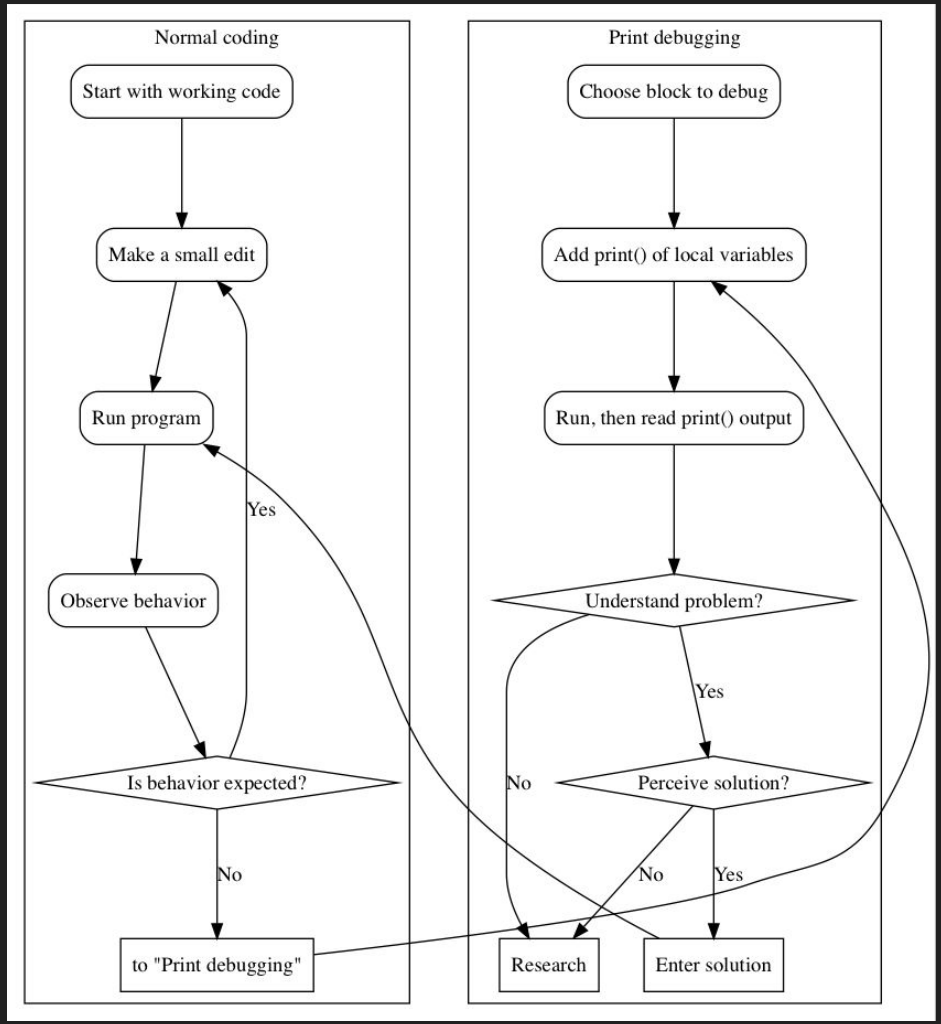
- Three **symptoms**:
  - Crashes, slowness, unexpected results
- Five **causes**:
  - Variance, framework state, mental models, exponential behavior, heavy load
- One **core technique**:
  - **Localize** the **problem state** to observe it
  - **Scale** out and up in **steps**

Synopsis (2)


Map your symptom
Unexpected Result
Slowness
Traceback
To ...
Root Causes
... one of:
Data Variance
Exponential Behavior
Framework State
Inaccurate Mental Model
Heavy Load
Other (?)

Markup to flowchart

```
"Start with working code" -> "Make a small edit"
"Make a small edit" -> "Run program"
"Run program" -> "Observe behavior"
"Observe behavior" -> "Is behavior expected?"
"Is behavior expected?" ->
                "Make a small edit"[label="Yes"]
"Is behavior expected?" ->
                "to \"Print debugging\""[label="No"]
```



In conclusion

A **possibility** you may not have considered ...

Thank you

www.sampenrose.net | @sampenrose