

LET'S BREAK APACHE SPARK WORKSHOP (USING DOCKER)



Grzegorz Gawron, head of data science



REQUIREMENTS

A laptop with:

LET'S BREAK APACHE SPARK WORKSHOP (USING DOCKER)



- at least 10GB disk space free, 4 cores, 8GB of free RAM
- installed Docker Community Edition
<https://docs.docker.com/engine/installation/>
- installed Docker Compose <https://docs.docker.com/compose/install/>
- installed git (<https://gist.github.com/derhuerst/1b15ff4652a867391f03>)

Grzegorz Gawron



Run:

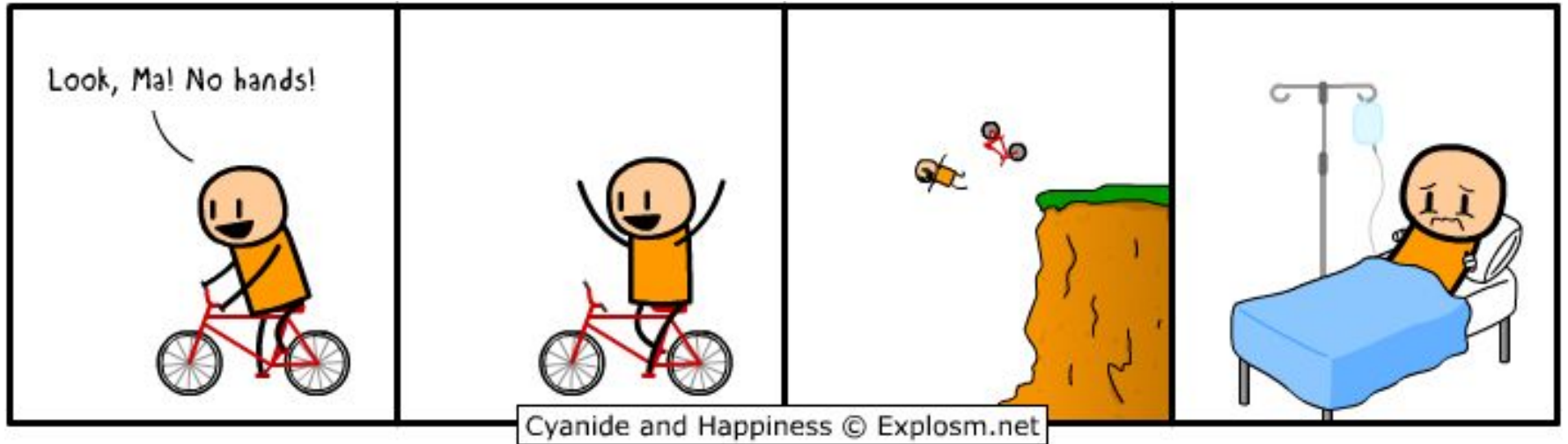
- ❏ `docker pull dimajix/jupyter-spark`
 - ❏ `git clone https://github.com/dimajix/docker-jupyter-spark.git`
- * some useful stuff: <https://github.com/gregaw/workshop-break-spark>

ABOUT ME

Grzegorz Gawron
ggawron@virtuslab.com

- currently running a large data engineering project for a global retailer
- greenishfield
 - Spark ML pipelines (forecasting, recommendation)
 - Spark Streaming
- large hadoop cluster (1000s cores + TBs RAM)
- spark, scala, python

MOTIVATION: KIDS



AGENDA

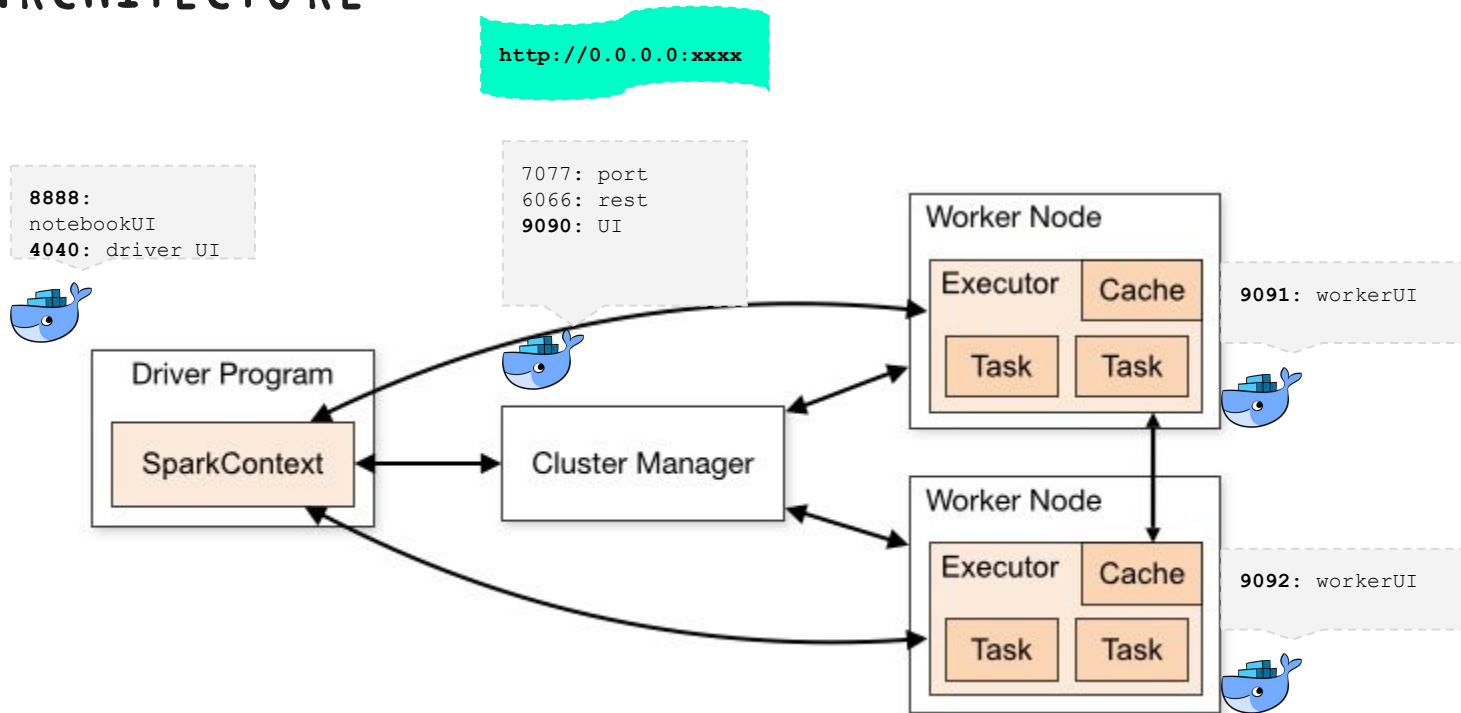
- run a simple distributed python spark app in jupyter notebook and via shell
- see it running healthily on spark UIs
- repeat the **break-and-feel** cycle multiple times

QUESTIONS WE'LL BE ASKING

- healthy system
 - what does it look like?
- how to
 - make a spark job hang temporarily
 - ... or hang it for good?
 - get a spark job live-locked?
- what happens if
 - you kill the master? slaves? master and slaves?
 - ! the slaves/master come back?
 - the notebook kernel dies?

ARCHITECTURE

SPARK ARCHITECTURE



OTHER WAYS OF RUNNING SPARK

Out of Scope

- mesos
- yarn
- kubernetes

CONFIGS: DOCKER-COMPOSE.YML

```
Machito:docker gregaw$ cat docker-jupyter-spark/docker-compose.yml
version: "3"
```

```
services:
```

```
  jupyter-notebook:
    hostname: jupyter-notebook
    container_name: jupyter-notebook
    image: dimajix/jupyter-spark:latest
    command: notebook
    build:
      context: .
      args:
        http_proxy: ${http_proxy}
        https_proxy: ${https_proxy}
    env_file:
      - docker-compose.env
    environment:
      - http_proxy=${http_proxy}
      - https_proxy=${https_proxy}
    expose:
      - 8888
    ports:
      - 8888:8888
      - 4040:4040
    volumes:
      - ./:/shared
```

```
  spark-master:
    hostname: spark-master
```

CONFIGS: DOCKER-COMPOSE.ENV

```
Machito:docker gregaw$ cat docker-jupyter-spark/docker-compose.env
#...
```

```
SPARK_MASTER_HOST=spark-master
SPARK_MASTER_PORT=7077
SPARK_WEBUI_PORT=9090
SPARK_WORKER_CORES=2
SPARK_WORKER_MEMORY=2G
SPARK_LOCAL_DIRS=/tmp
SPARK_WORKER_DIR=/tmp
```

```
SPARK_MASTER=spark://spark-master:7077
```

```
# Additional Spark variables used by notebook
```

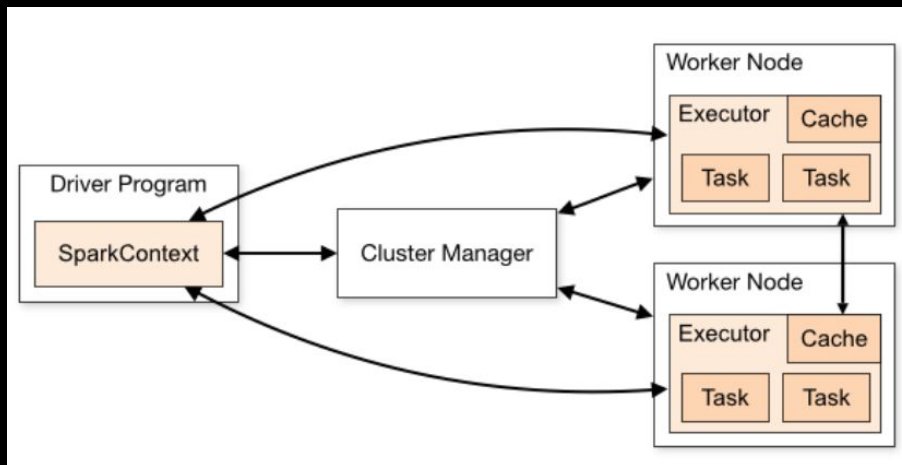
```
SPARK_DRIVER_MEMORY=1G
SPARK_EXECUTOR_MEMORY=1G
SPARK_EXECUTOR_CORES=1
SPARK_NUM_EXECUTORS=4
```

```
# AWS S3 Configuration
```

```
#S3_ENDPOINT=s3.eu-central-1.amazonaws.com
#S3_PROXY_HOST=<your-proxy-host>
#S3_PROXY_PORT=<your-proxy-port>
```

```
#AWS_ACCESS_KEY_ID=<your-aws-access-key>
```

```
#AWS_SECRET_ACCESS_KEY=<your-aws-access-secret>
```



CONFIGS [OPTIONAL]: KERNEL.JSON

```
Machito:docker gregaw$ cat docker-jupyter-spark/conf/jupyter-kernels/PySpark/kernel.json
{
  "display_name": "PySpark 2.1 (Python 3.5)",
  "language": "python",
  "argv": [
    "[% ANACONDA_HOME %]/bin/python3",
    "-m", "ipykernel",
    "-f", "{connection_file}"
  ],
  "env": {
    "TZ": "UTC",
    "SPARK_MAJOR_VERSION": "${SPARK_MAJOR_VERSION}",
    "SPARK_HOME": "[% SPARK_HOME %]",
    "PYTHONPATH": "[% SPARK_HOME %]/python:[% SPARK_HOME %]/python/lib/py4j-0.10.4-src.zip",
    "PYTHONSTARTUP": "[% SPARK_HOME %]/python/pyspark/shell.py",
    "PYTHONHASHSEED": "0",
    "SPARK_YARN_USER_ENV": "PYTHONHASHSEED=0",
    "PYSPARK_PYTHON": "[% ANACONDA_HOME %]/bin/python3",
    "PYSPARK_SUBMIT_ARGS": "--master [% SPARK_MASTER %] --driver-memory=[% SPARK_DRIVER_MEMORY %] --executor-cores=[%
SPARK_EXECUTOR_CORES %] --executor-memory=[% SPARK_EXECUTOR_MEMORY %] --num-executors=[% SPARK_NUM_EXECUTORS %]
pyspark-shell"
  }
}
```

HEALTHY SYSTEM

THE TOOLBOX

> **cd docker-jupyter-spark**

> **docker network create dimajix**

> **docker-compose up**

> **docker-compose down**

> docker ps

> docker inspect jupyter-notebook

> docker images

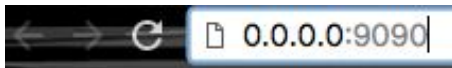
> docker stop jupyter-spark-master

> docker start jupyter-spark-master

SHOW SOME LOGS

```
> docker exec -it jupyter-spark-master tail -f  
/opt/spark/logs/spark--org.apache.spark.deploy.master.Master-1-spark-master.out  
  
> docker exec -it jupyter-spark-slave-1 tail -f  
/opt/spark/logs/spark--org.apache.spark.deploy.worker.Worker-1-spark-slave-1.out  
  
> docker exec -it jupyter-spark-slave-2 tail -f  
/opt/spark/logs/spark--org.apache.spark.deploy.worker.Worker-2-spark-slave-2.out
```

SPARK MASTER



Spark Master at spark://spark-master:7077

URL: spark://spark-master:7077

REST URL: spark://spark-master:6066 (*cluster mode*)

Alive Workers: 2

Cores in use: 4 Total, 0 Used

Memory in use: 8.0 GB Total, 0.0 B Used

Applications: 0 [Running](#), 13 [Completed](#)

Drivers: 0 Running, 0 Completed

Status: ALIVE

Workers

Worker Id	Address	State	Cores	Memory
worker-20171023065727-172.18.0.2-39315	172.18.0.2:39315	ALIVE	2 (0 Used)	4.0 GB (0.0 B Used)
worker-20171023065728-172.18.0.3-43191	172.18.0.3:43191	ALIVE	2 (0 Used)	4.0 GB (0.0 B Used)

SPARK WORKER



Spark Worker at 172.18.0.3:43191

ID: worker-20171023065728-172.18.0.3-43191

Master URL: spark://spark-master:7077

Cores: 2 (0 Used)

Memory: 4.0 GB (0.0 B Used)

[Back to Master](#)

Running Executors (0)

ExecutorID	Cores	State	Memory	Job Details	Logs
------------	-------	-------	--------	-------------	------

Finished Executors (12)

ExecutorID	Cores	State	Memory	Job Details	Logs
0	1	KILLED	2.0 GB	ID: app-20171023073418-0000 Name: pyspark-shell User: root	stdout stderr
1	1	KILLED	2.0 GB	ID: app-20171023073418-0000 Name: pyspark-shell User: root	stdout stderr

SPARK DRIVER

[Jobs](#)[Stages](#)[Storage](#)[Environment](#)[Executors](#)[SQL](#)[pyspark-shell app](#)

Spark Jobs (?)

User: root

Total Uptime: 16 min

Scheduling Mode: FIFO

Active Jobs: 1

Completed Jobs: 11

▶ [Event Timeline](#)

Active Jobs (1)

Job Id ▾	Description		Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
11	reduce at <ipython-input-12-fec2a75ec3fb>:14	(kill)	2017/10/23 14:30:44	5.7 min	0/1	0/10

Completed Jobs (11)

Job Id ▾	Description		Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
10	reduce at <ipython-input-11-fec2a75ec3fb>:14		2017/10/23 14:29:17	2 s	1/1	10/10
9	reduce at <ipython-input-10-fec2a75ec3fb>:14		2017/10/23 14:28:45	2 s	1/1	10/10

JUPYTER NOTEBOOK

← → ↻ ⓘ 0.0.0.0:8888/tree



Files

Running

Clusters

Conda

Select items to perform actions on them.

Upload

New ▾



☐ spark-warehouse

☐ Untitled.ipynb

☐ Untitled1.ipynb

Text File

Folder

Terminal

Notebooks

PySpark 2.1 (Python 3.5)

Python 3.5

Python [conda root]

Python [default]

DRIVERS

USE SPARK-SUBMIT

```
> docker exec -it jupyter-notebook bash

# cd /opt/spark-2.2.0-bin-without-hadoop/

# bin/spark-submit --executor-memory=1G --conf "spark.driver.memory=1G" --conf
"spark.cores.max=10" --conf "spark.executor.cores=2" --master
spark://spark-master:7077 examples/src/main/python/pi.py
```

USE JUPYTER NOTEBOOK

```
# based on pyspark standard example
from random import random
from operator import add

def within_circle_quarter (_):
    # x, y in [0,1)
    x, y = random(), random()
    return 1 if x**2 + y**2 <= 1.0 else 0

spark = SparkSession.builder.appName( 'breakit' ).getOrCreate()
spark.sparkContext.setLogLevel( "INFO" )
n = 10000000
partitions = 10
quarter_count = spark.sparkContext.parallelize( range(n), partitions).map(within_circle_quarter).reduce(add)

pi = 4.0 * quarter_count / n

print("pi={}".format(pi))
```



GET FAMILIAR WITH
RUNNING JOBS, PLAY
WITH PARAMS

(SPARK-SUBMIT)



GET FAMILIAR WITH TWEAKING YOUR CLUSTER

(DOCKER-COMPOSE.ENV +
DOCKER-COMPOSE UP)



GET FAMILIAR WITH
INTRODUCING HAVOC
(DOCKER STOP <CONTAINER-NAME>)



MAKE A JOB HANG



MAKE A JOB HANG
FOR GOOD



GET A SPARK JOB
LIVE-LOCKED



WHAT IF... YOU KILL
THE MASTER?



WHAT IF...WORKERS
ARE TOO GREEDY?



WHAT IF... YOUR
DRIVER GETS TOO
LITTLE MEMORY?



HOW TO FIX THE LIVELOCK PROBLEM



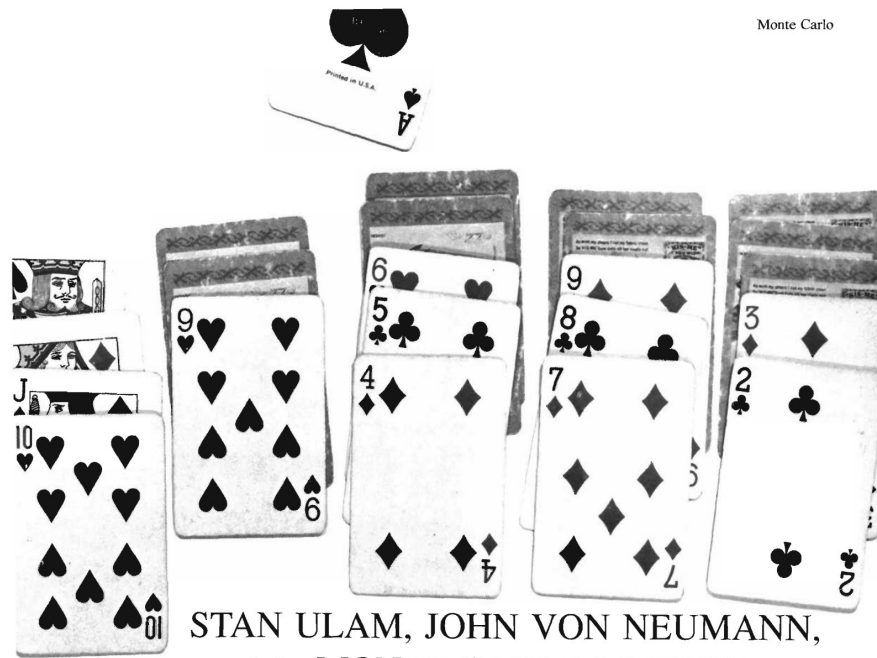
🔍 spark source code github|



IMPLEMENT SOLITAIRE
SUCCESS PROBABILITY
ESTIMATION USING MONTE
CARLO IN SPARK



Monte Carlo



STAN ULAM, JOHN VON NEUMANN,
and the MONTE CARLO METHOD

by Roger Eckhardt



🔍 stanislaw ulam solitaire monte carlo



COME UP WITH
YOUR OWN TASK

