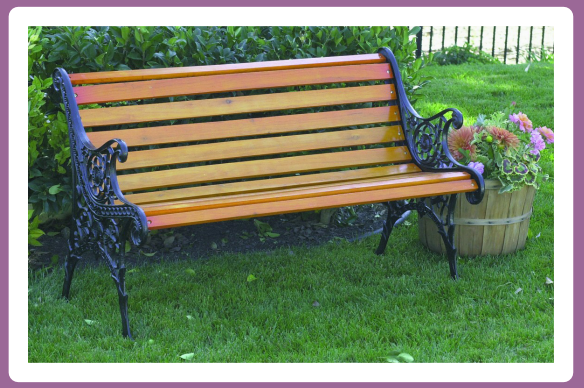
"S" +  =

Spark-Bench

Simulate, Test, Compare, Exercise, and Yes, Even Benchmark!

SPARK SUMMIT EUROPE 2017

Emily May Curtin

IBM Watson Data Platform

Atlanta, GA

@emilymaycurtin

@ecurtin

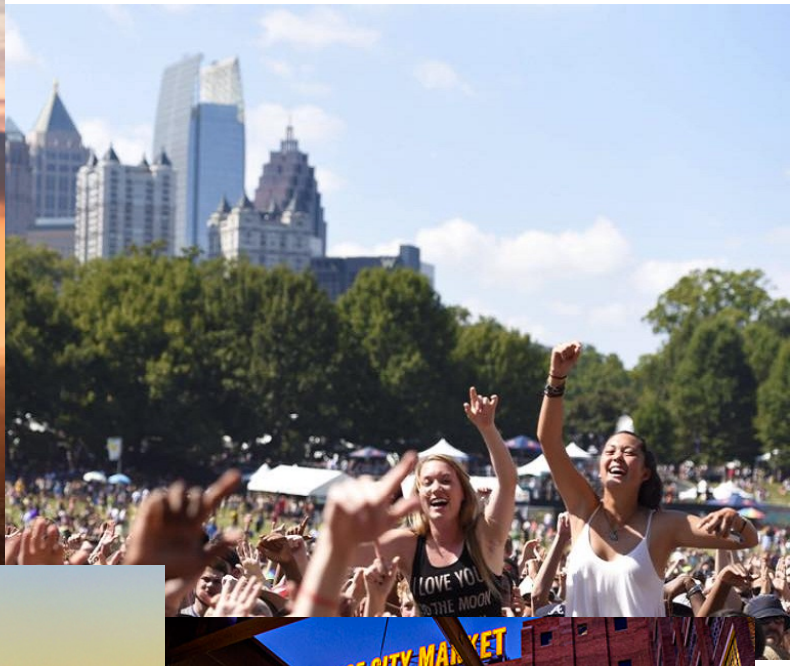
#EUeco8

#sparkbench

Who Am I

- Artist, former documentary film editor
- 2017 Spark Summit East – Spark + Parquet
- Tabs, not spaces (at least for Scala)(where 1 tab = 2 spaces)
- Software Engineer, IBM Watson Data Platform
- **Current lead developer of SparkTC/spark-bench**

Atlanta!!!!

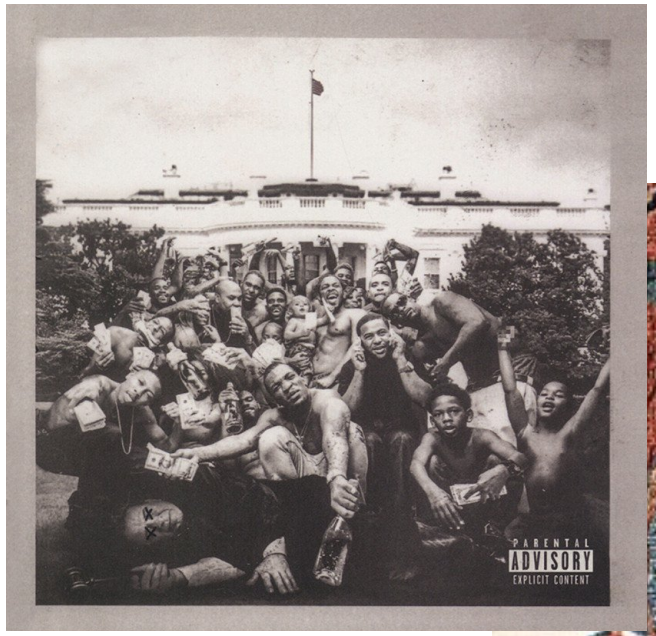


Spark-Bench

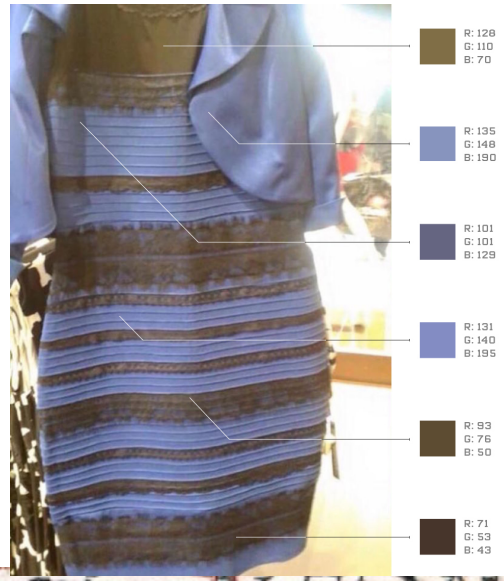
- History & Rewrite Motivation
- How Spark-Bench Works
- Detailed Example: Parquet vs. CSV Benchmarking
- More Examples
  - Cluster Hammer
  - Multiple Notebook Users
  - Varying over Spark parameters

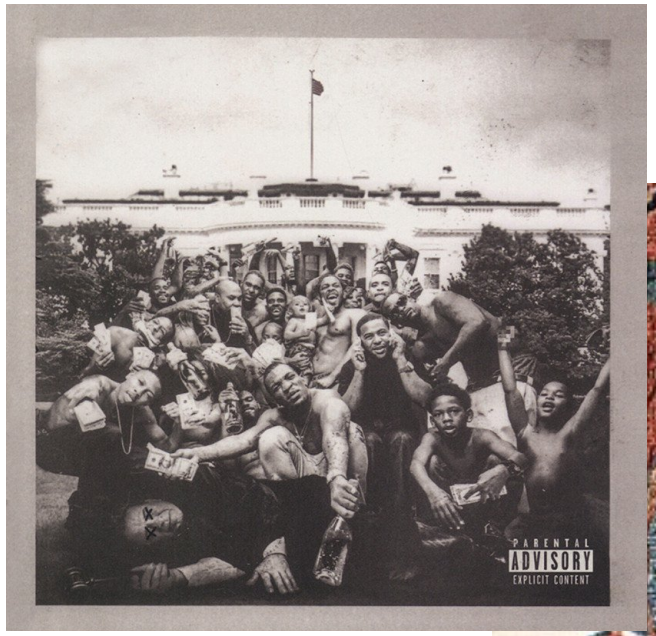
2015 A.D.

2015 A.D.



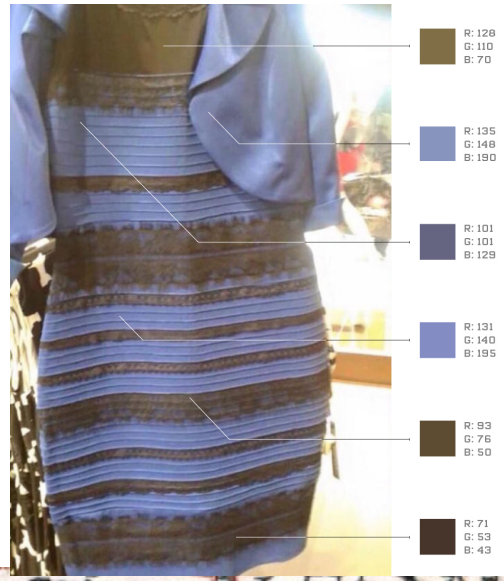
2015 A.D.

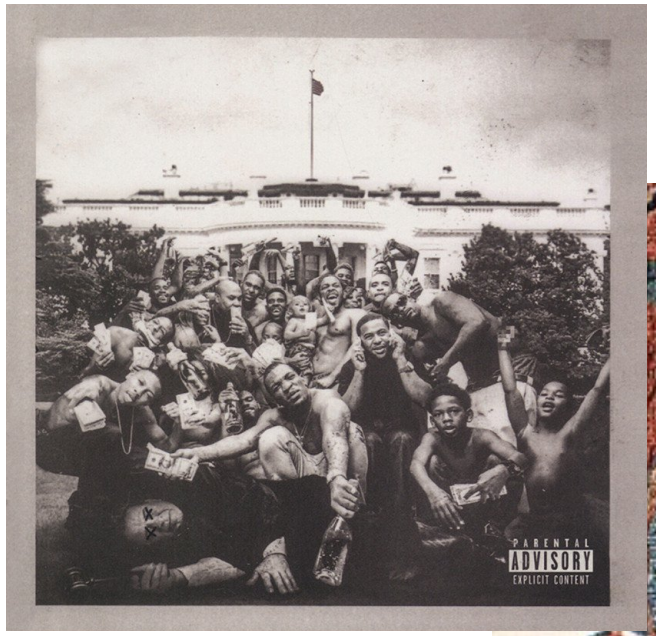

R: 128
G: 110
B: 70
R: 135
G: 148
B: 190
R: 101
G: 101
B: 129
R: 131
G: 140
B: 195
R: 93
G: 76
B: 50
R: 71
G: 53
B: 43


PARENTAL
ADVISORY
EXPLICIT CONTENT


WIL LELM.

2015 A.D.





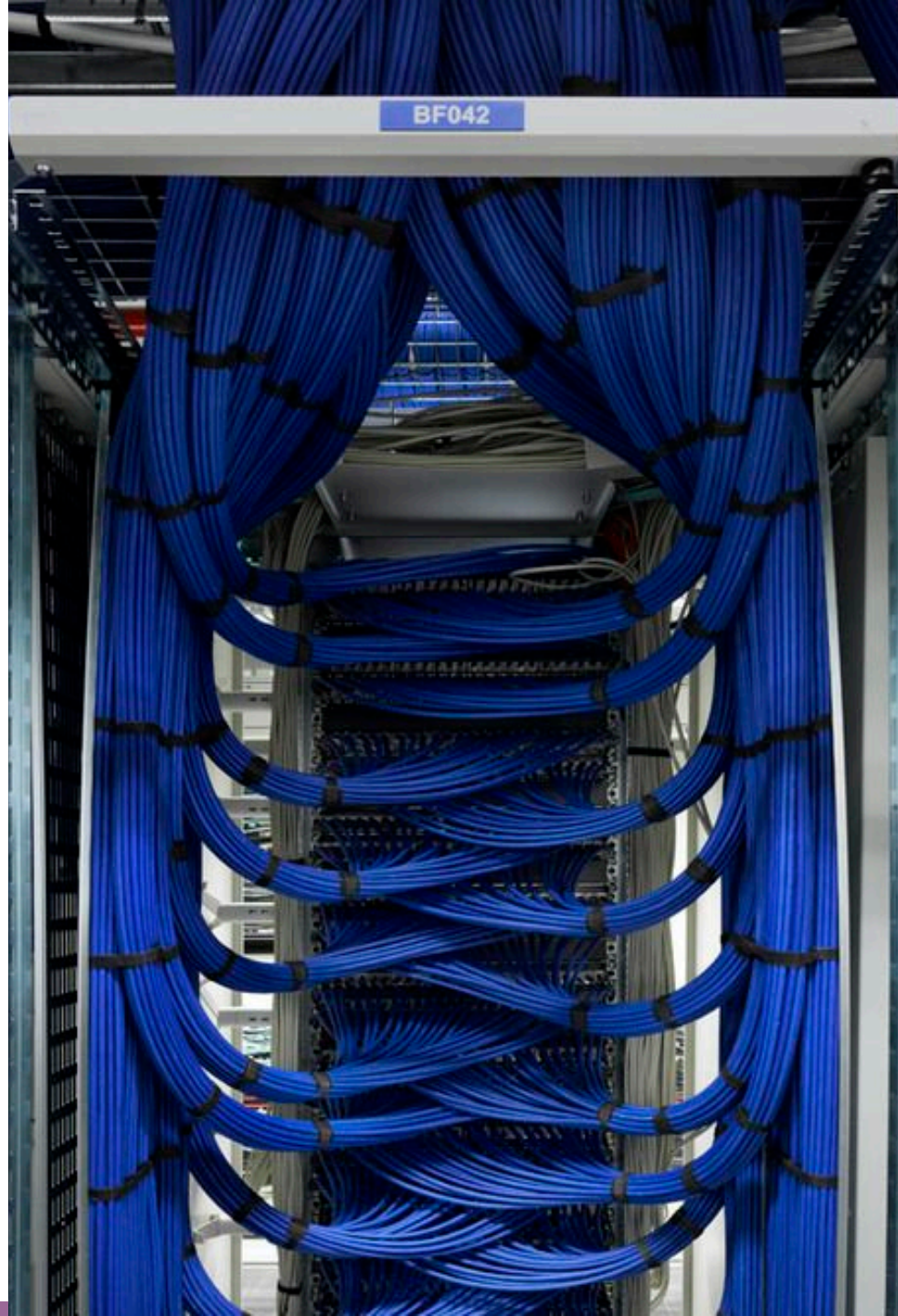
Spark 1.4.0

Spark Benchmarking circa 2015 A.D.

Spark-Bench 2015

"*SparkBench: A Comprehensive Benchmarking Suite For In Memory Data Analytics Platform Spark*"

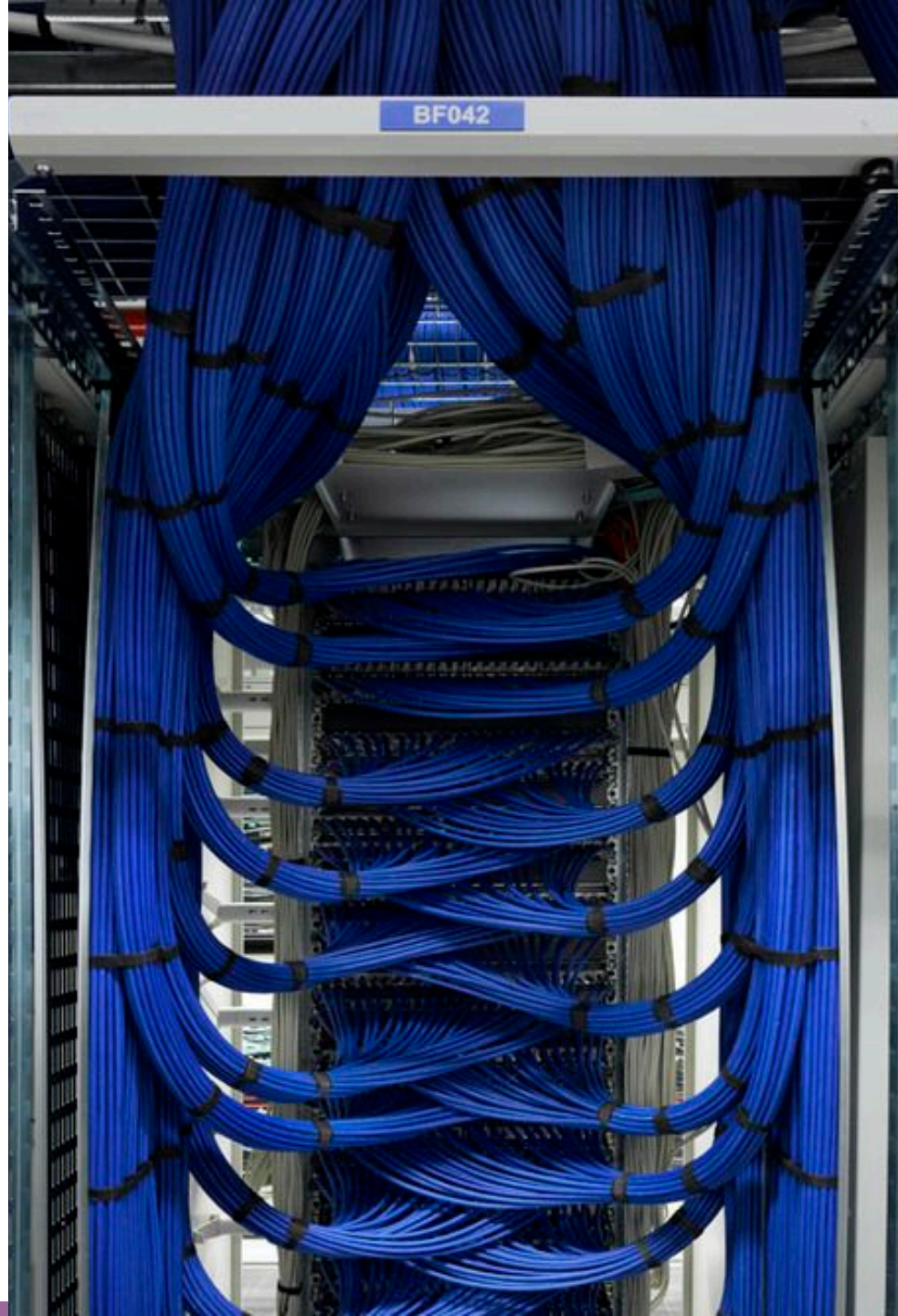
Min Li, Jian Tan, Yandong Wang, Li Zhang, Valentina Salapura IBM TJ Watson Research Center - APC 2015



Spark-Bench 2015

"*SparkBench: A Comprehensive Benchmarking Suite For In Memory Data Analytics Platform Spark*"

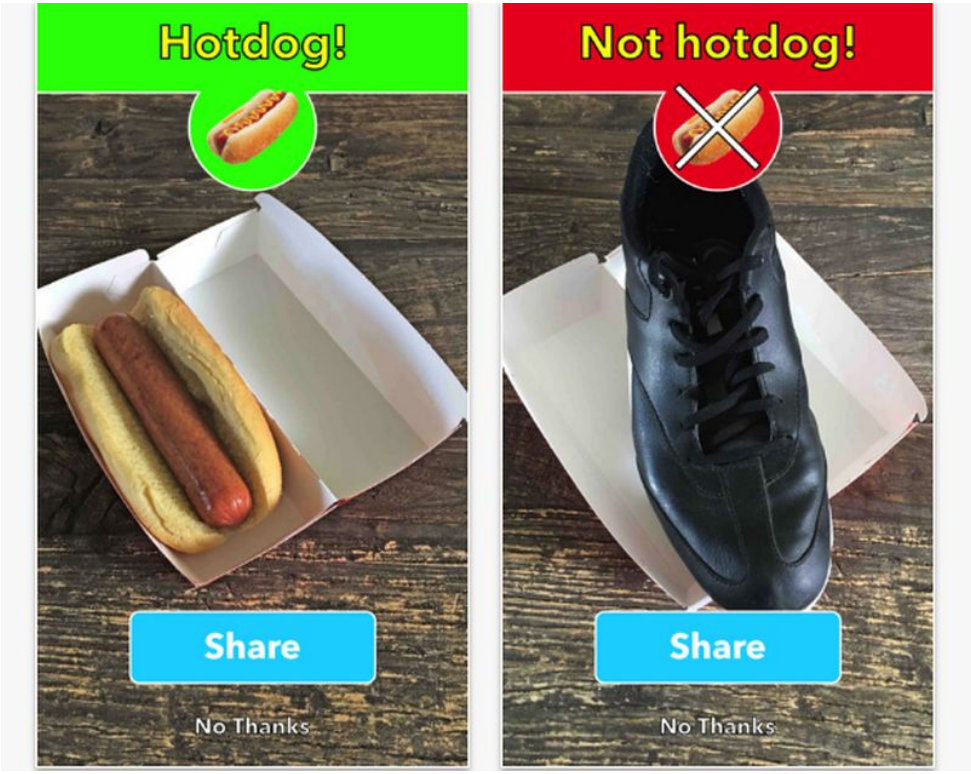
Min Li, Jian Tan, Yandong Wang, Li Zhang, Valentina Salapura IBM TJ Watson Research Center - APC 2015



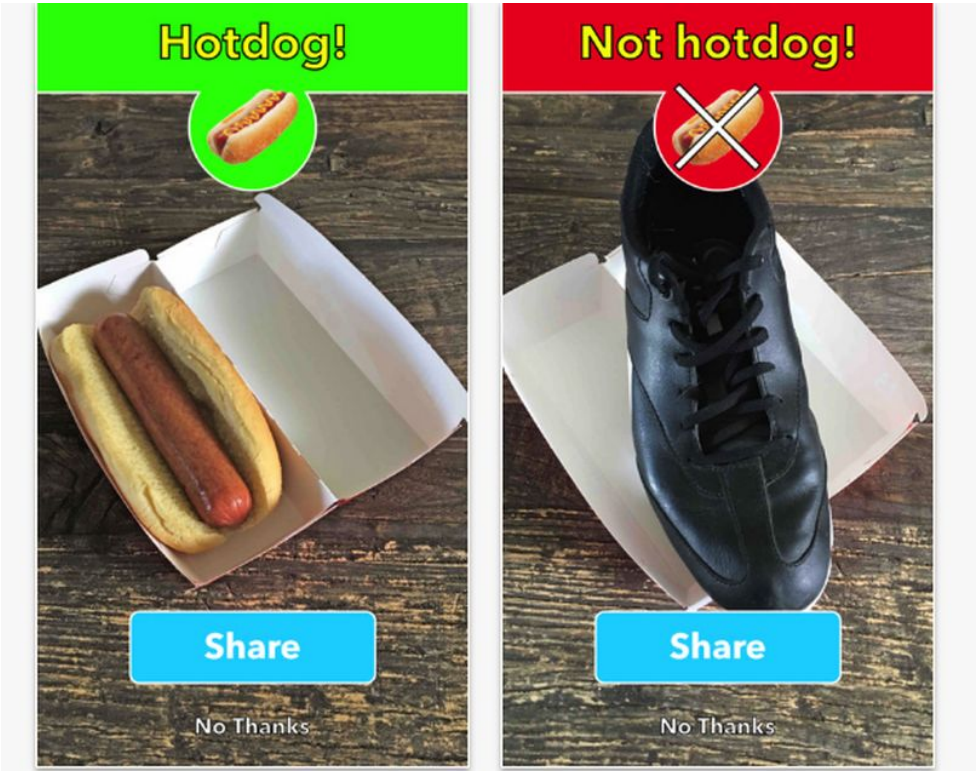
2017 A.D.

2017 A.D.

2017 A.D.



2017 A.D.



APACHE Spark™ 2.2.0

Watson Data Platform

Data Science Experience

IBM Analytics Engine (*Beta*)

Watson Machine Learning

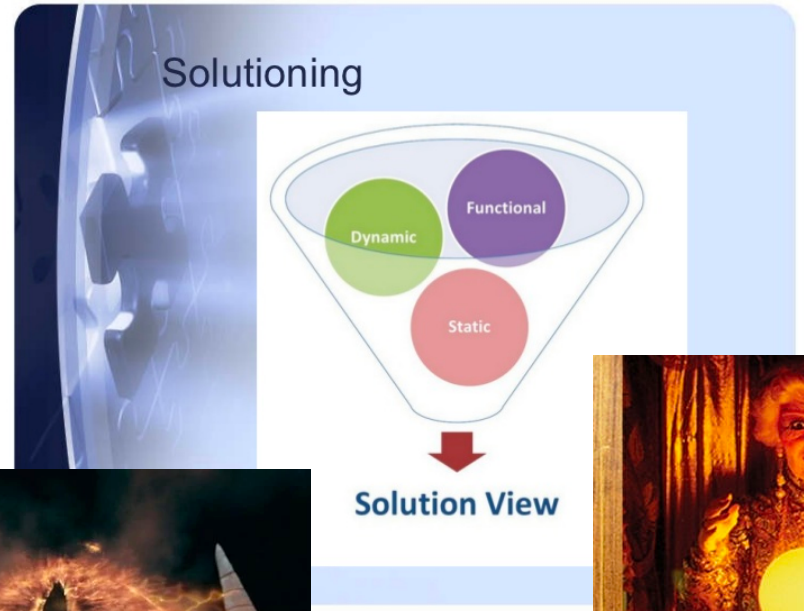
AND MUCH, MUCH MORE

As a researcher, I want to run a set of standard benchmarks with generated data against one instance of Spark, one time.

--The user story that SparkTC/spark-bench version 1.0 fulfilled

Good start, *let's keep going!*

The Vision


Solutioning
Dynamic
Functional
Static
Solution View


FUTURE AVE
VISION ST


VISION


Vision


SYNERGY


VISION
success
focus
motivation
teamwork
strategy
goal
marketing
money
creation
concept
business

As a Spark developer, I want to run regression tests with my changes in order to provide some authoritative performance stats with my PR.

As a Spark developer, I want to see regression tests run against Spark nightly builds.

As a Parquet developer, I want to compare performance between Spark built with several different versions of Parquet.

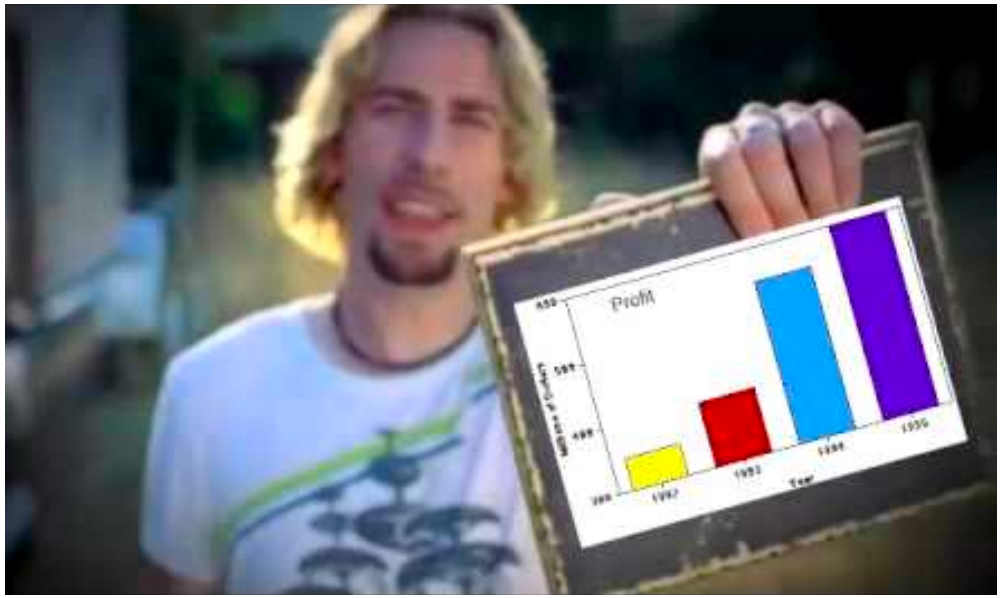
As a Spark developer, I want to simulate multiple notebook users hitting the same cluster so that I can test my changes to the dynamic allocation algorithm.

As an enterprise user of Spark, I want to make my daily batch ingest job run faster by tuning parameters.

As a DevOps professional, I want to see how performance changes for our daily jobs as a I add more nodes to our prod cluster.

As a data scientist, I want to run several machine learning algorithms over the same set of data in order to compare results.

As anybody who has ever had to make a graph about Spark, I want to get numbers easily and quickly so I can focus on making *beautiful graphs.*



How It Works

Structural Details

	Old Version	✨New Version✨
Project Design	Series of shell scripts calling individually built jars	Scala project built using SBT
Build/Release	Manually/Manually	SBT, TravisCI, auto-release to Github Releases on PR merge
Configuration	Scattered in shell script variables	Centralized in one config file with many new capabilities
Parallelism	None	Three different levels!
Custom Workloads	Requires writing a new subproject with all accompanying bash	Implement our abstract class, Bring Your Own Jar

The Config File (a first glance)

```
spark-bench = {
  spark-submit-config = [{
    workload-suites = [
      {
        descr = "One run of SparkPi and that's it!"
        benchmark-output = "console"
        workloads = [
          {
            name = "sparkpi"
            slices = 10
          }
        ]
      }
    ]
  }]
}
```

Workloads

- Atomic unit of spark-bench
- Optionally read from disk
- Optionally write results to disk
- All stages of the workload are timed
- Infinitely composable

Workloads

- Machine Learning
  - Kmeans
  - Logistic Regression
  - Other workloads ported from old version
- SQL queries
- Oddballs
  - SparkPi
  - Sleep
  - Cache Test

Workloads: Data Generators

- Graph data generator
- Kmeans data generator
- Linear Regression data generator
- More coming!

What About Other Workloads?

1. Active development, more coming!
2. Contributions welcome ☺

3. Bring Your Own Workload!

```
{
  name = "custom"
  class = "com.seriouscompany.HelloString"
  str = ["Hello", "Hi"]
}
```

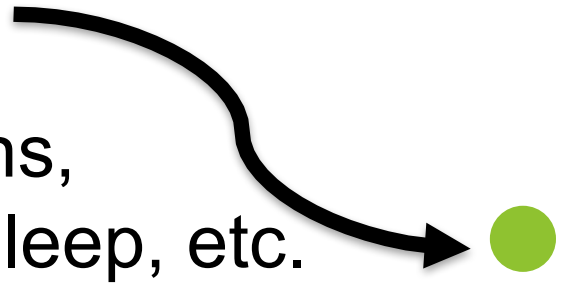
Custom Workloads

```
{
  name = "custom"
  class = "com.seriouscompany.OurGiganticIngestDailyIngestJob"
  date = "Tues Oct 26 18:58:25 EDT 2017"
  source = "s3://buncha-junk/2017-10-26/"
}
```

```
{
  name = "custom”
  class = "com.example.WordGenerator”
  output = "console”
  rows = 10
  cols = 3
  word = "Cool stuff!!"
}
```

← See our documentation on custom workloads!

Workload



Could be Kmeans,
SparkPi, SQL, Sleep, etc.

```
{
  name = "sparkpi"
  slices = 500000000
}
```

SparkTC/spark-bench Github Developer's Guide Examples Quickstart User's Guide Workloads

SparkPi

Borrowing from the classic Spark example, this workload computes an approximation of pi. From the Spark examples page: https://spark.apache.org/examples.html

```
Spark can also be used for compute-intensive tasks. This code estimates π
by "throwing darts" at a circle. We pick random points in the unit square
((0, 0) to (1,1)) and see how many fall in the unit circle. The fraction
should be π / 4, so we use this to get our estimate.
```

SparkPi is particularly useful for exercising the computing power of Spark without the consideration of heavy I/O from data-reliant workloads.

The timing result of SparkPi will include the estimate of Pi that was generated.

Parameters

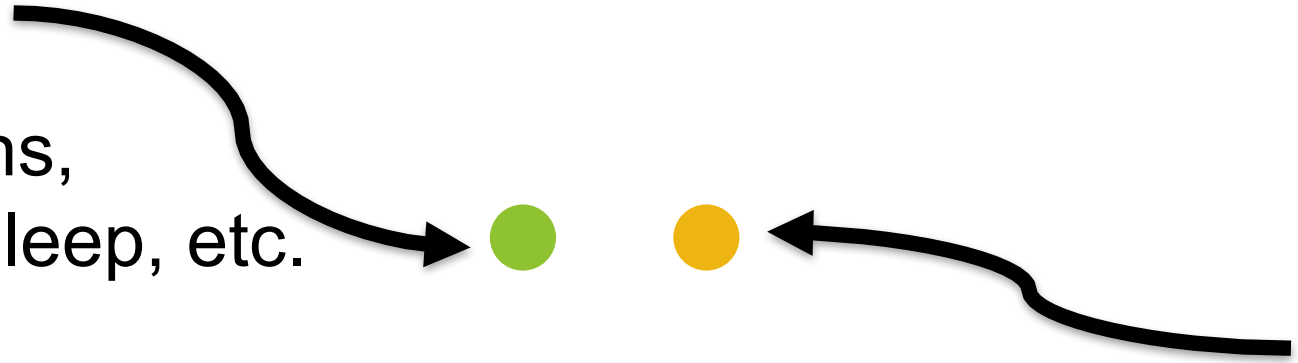
Name	Required	Default	Description
name	yes	–	“sparkpi”
slices	no	2	Number of partitions that will be spawned

Examples

```
{
```

Workload

Could be Kmeans,
SparkPi, SQL, Sleep, etc.

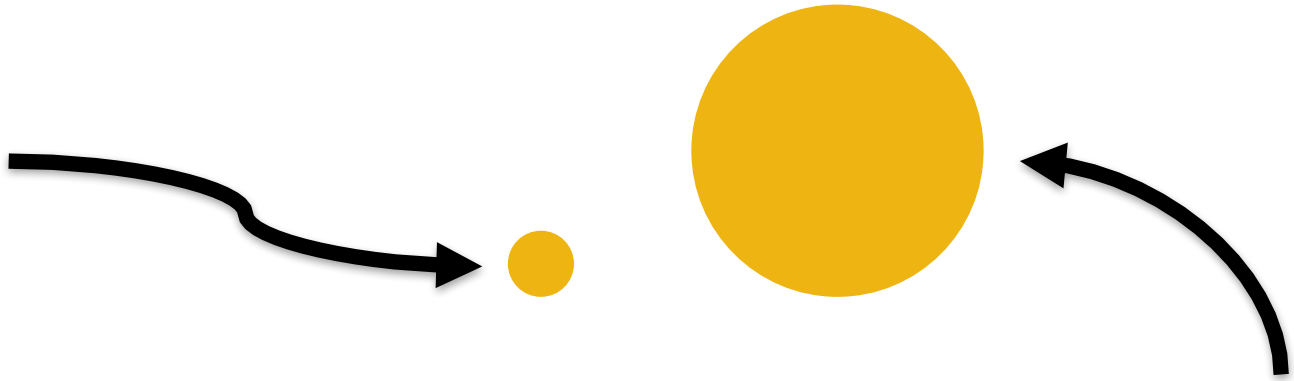


```
{
  name = "sparkpi"
  slices = 500000000
}
```

Data Generation Workload

```
{
  name = "data-generation-kmeans"
  rows = 1000
  cols = 24
  output = "/tmp/kmeans-data.parquet"
  k = 45
}
```

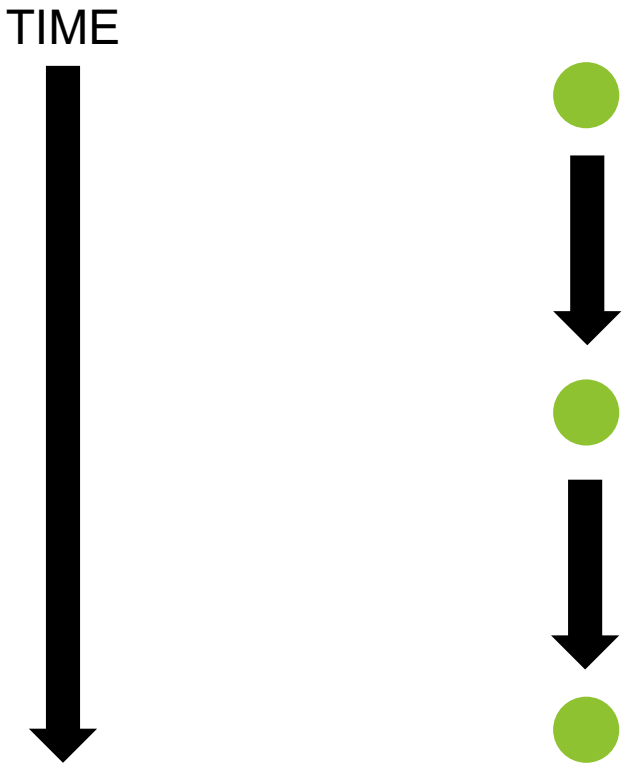
Data Generation, Small Set



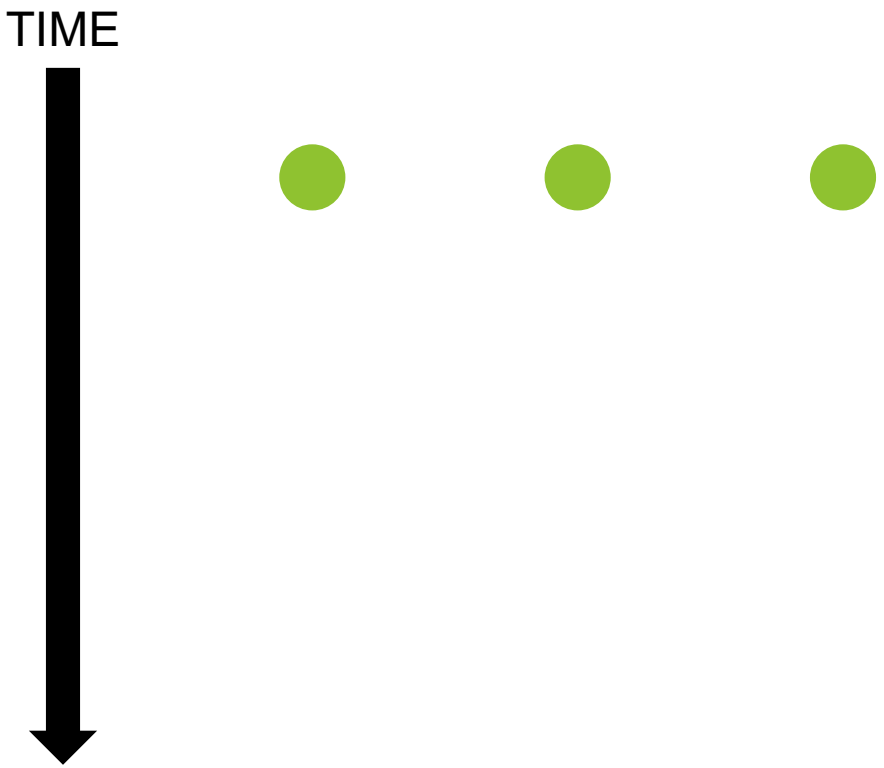
```
{
  name = "data-generation-kmeans"
  rows = 100
  cols = 24
  output = "/tmp/small-data.parquet"
}
```

Data Generation, Large Set

```
{
  name = "data-generation-kmeans"
  rows = 100000000
  cols = 24
  output = "/tmp/large-data.parquet"
}
```



Workloads running serially

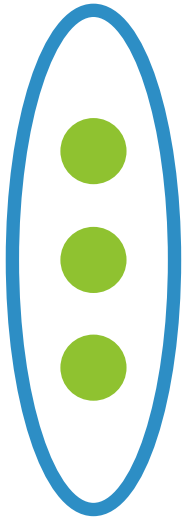


Workloads running in parallel

Workload Suites

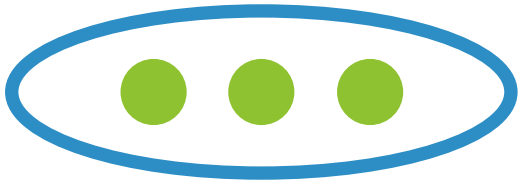
- Logical group of workloads
- Run workloads serially or in parallel
- Control the repetition of workloads
- Control the output of benchmark results
- Can be composed with other workload suites to run serially or in parallel

One workload suite
with three workloads
running **serially**



```
workload-suites = [ {
  descr = "Lol"
   parallel = false
  repeat = 1
  benchmark-output = "console"
  workloads = [
    {
      // Three workloads here
    }
  ]
}]
```

One workload suite
with three workloads
running **in parallel**

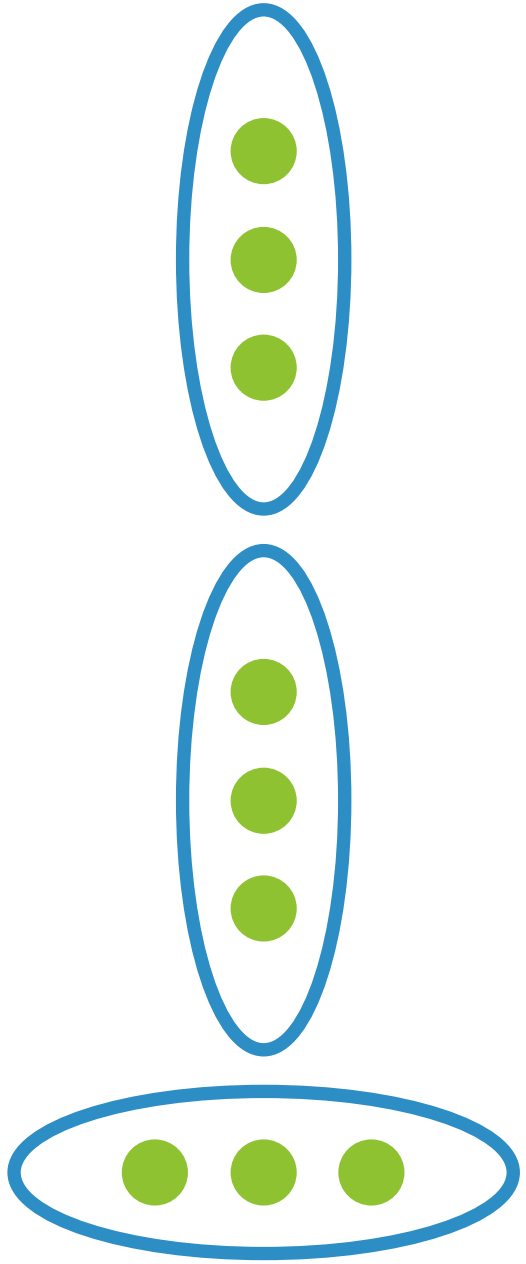


```
workload-suites = [ {
  descr = "Lol"
   parallel = true
  repeat = 1
  benchmark-output = "console"
  workloads = [
    {
      // Three workloads here
    }
  ]
}]
```

Three workload suites running **serially**.
Two have workloads running serially, one has workloads running in parallel.

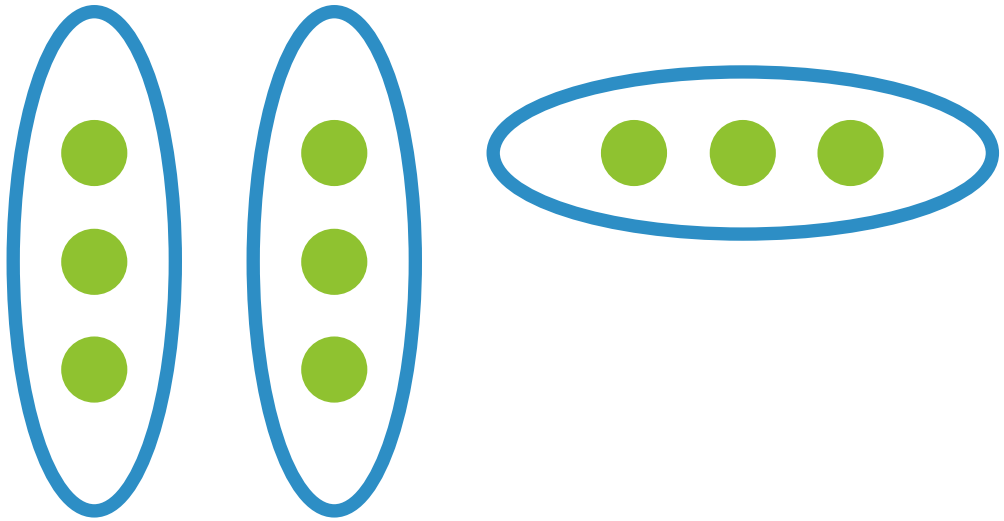
(config file is abbreviated for space)

```
suites-parallel = false
workload-suites = [
  {
    descr = "One"
    parallel = false
  },
  {
    descr = "Two"
    parallel = false
  },
  {
    descr = "Three"
    parallel = true
}]
```



Three workload suites running **in parallel.**

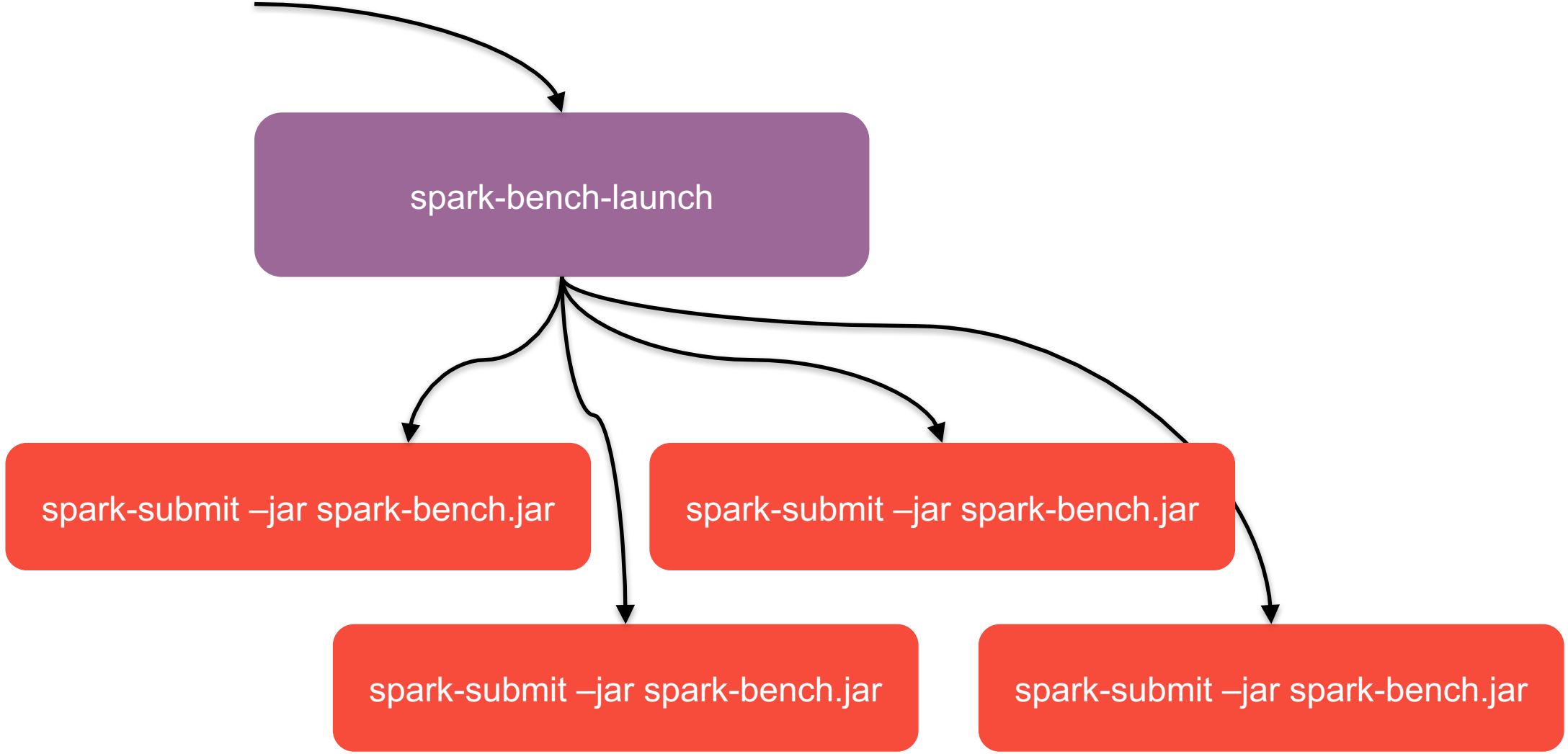
Two have workloads running serially,
one has workloads running in parallel.



(config file is abbreviated for space)

```
suites-parallel = true
workload-suites = [
  {
    descr = "One"
    parallel = false
  },
  {
    descr = "Two"
    parallel = false
  },
  {
    descr = "Three"
    parallel = true
}]
```

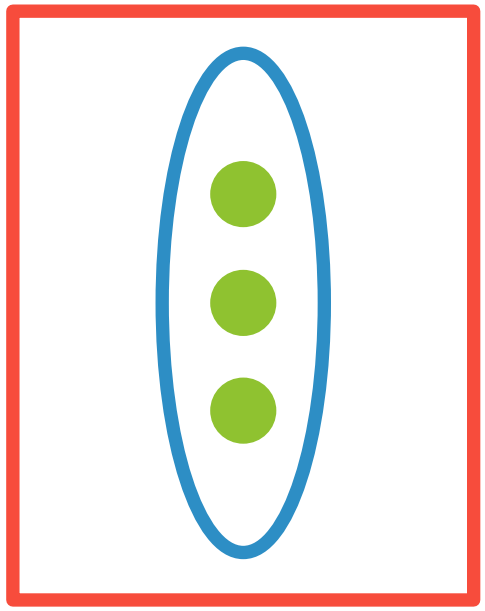
```
./bin/spark-bench.sh my-amazing-config-file.conf
```



Spark-Submit-Config

- Control any parameter present in a spark-submit script
- Produce and launch multiple spark-submit scripts
- Vary over parameters like executor-mem
- Run against different clusters or builds of Spark
- Can run serially or in parallel

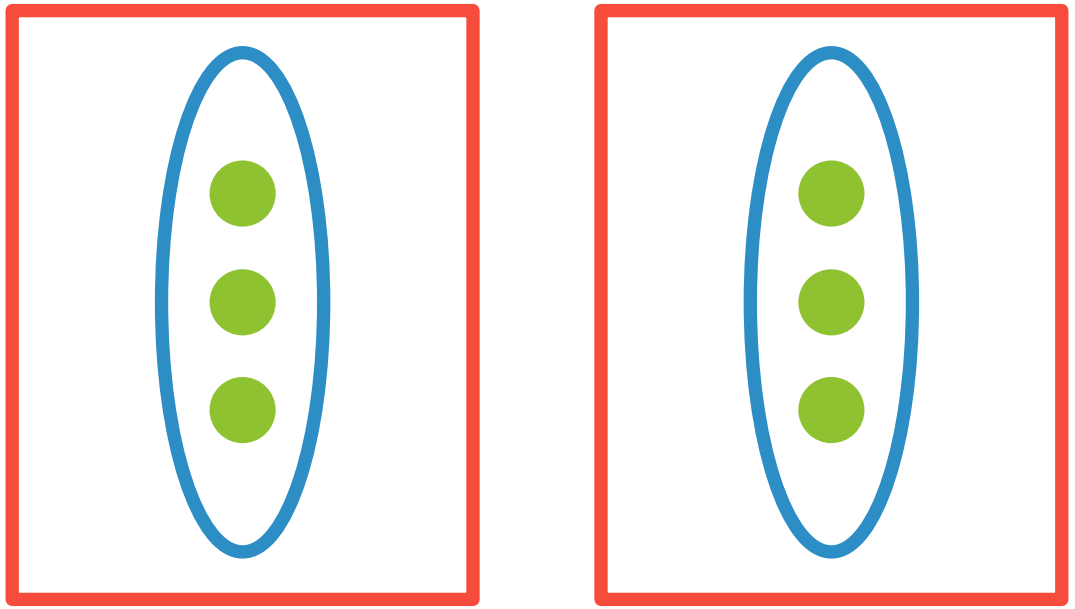
One spark-submit with one workload suite with three workloads running **serially**



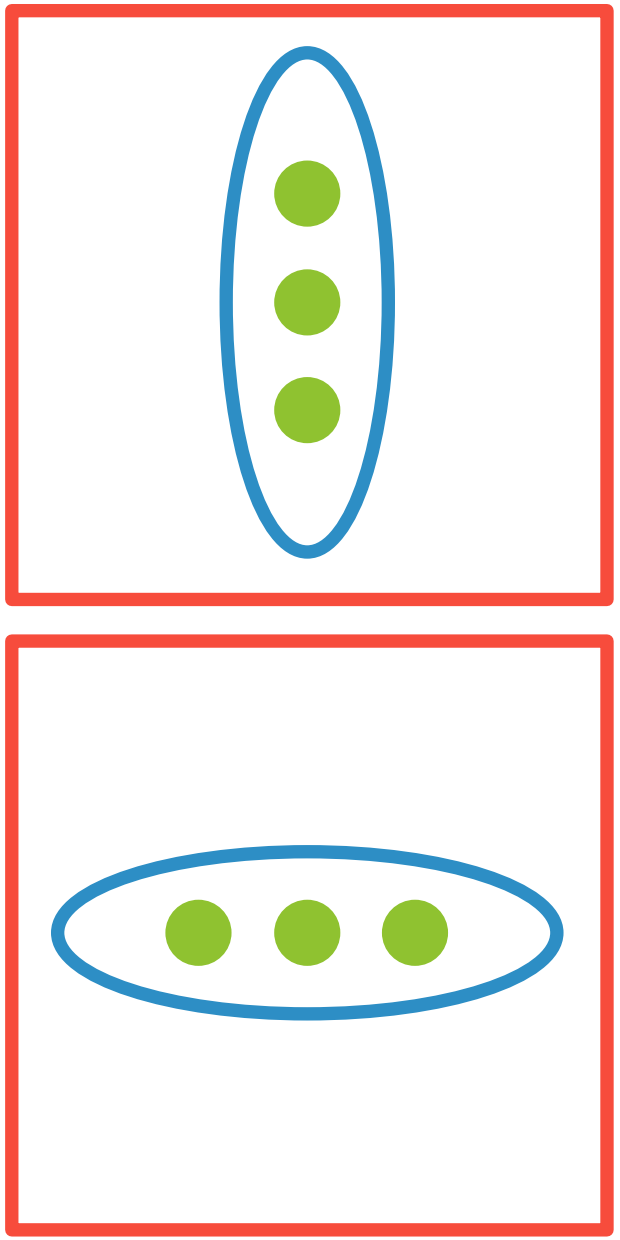
(config file is abbreviated for space)

```
spark-bench = {
  spark-submit-config = [{
    spark-args = {
      master = "yarn"
    }
    workload-suites = [{
      descr = "One"
      benchmark-output = "console"
      workloads = [
        // . . .
      ]
    }]
  }]
}
```

Two spark-submits running in parallel, each with one workload suite, each with three workloads running **serially**



Two spark-submits running **serially**,
each with one workload suite,
one with three workloads running **serially,**
one with three workloads running **in parallel**

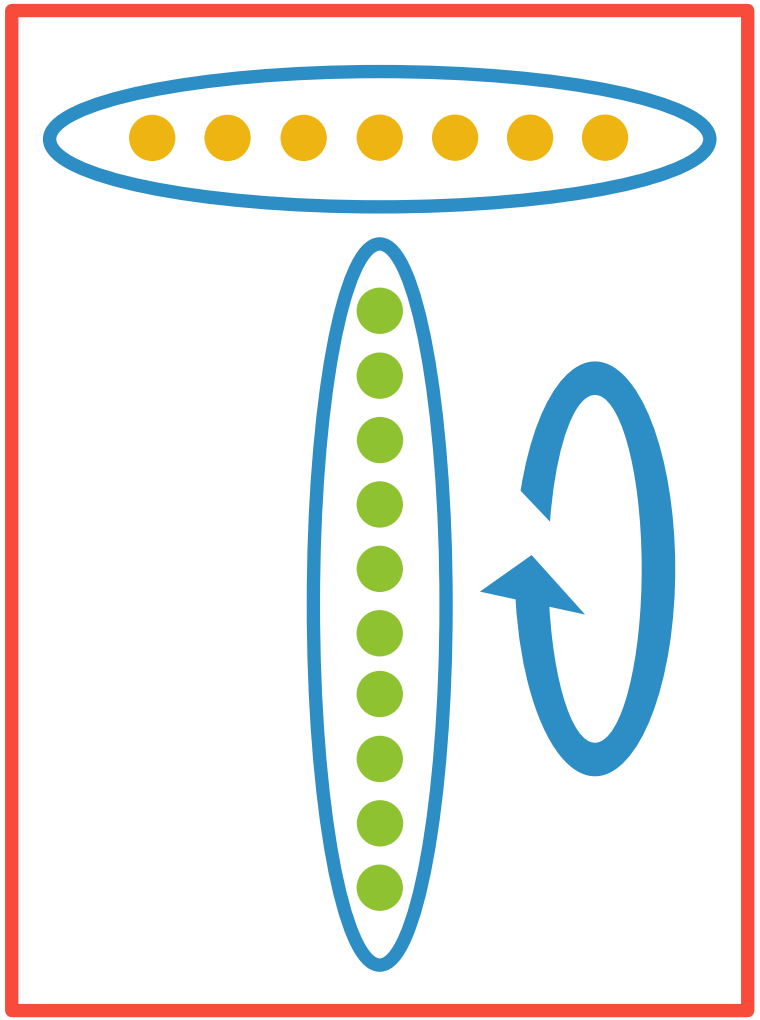


One spark-submit with two workload suites running serially.

The first workload suite generates data **in parallel**.

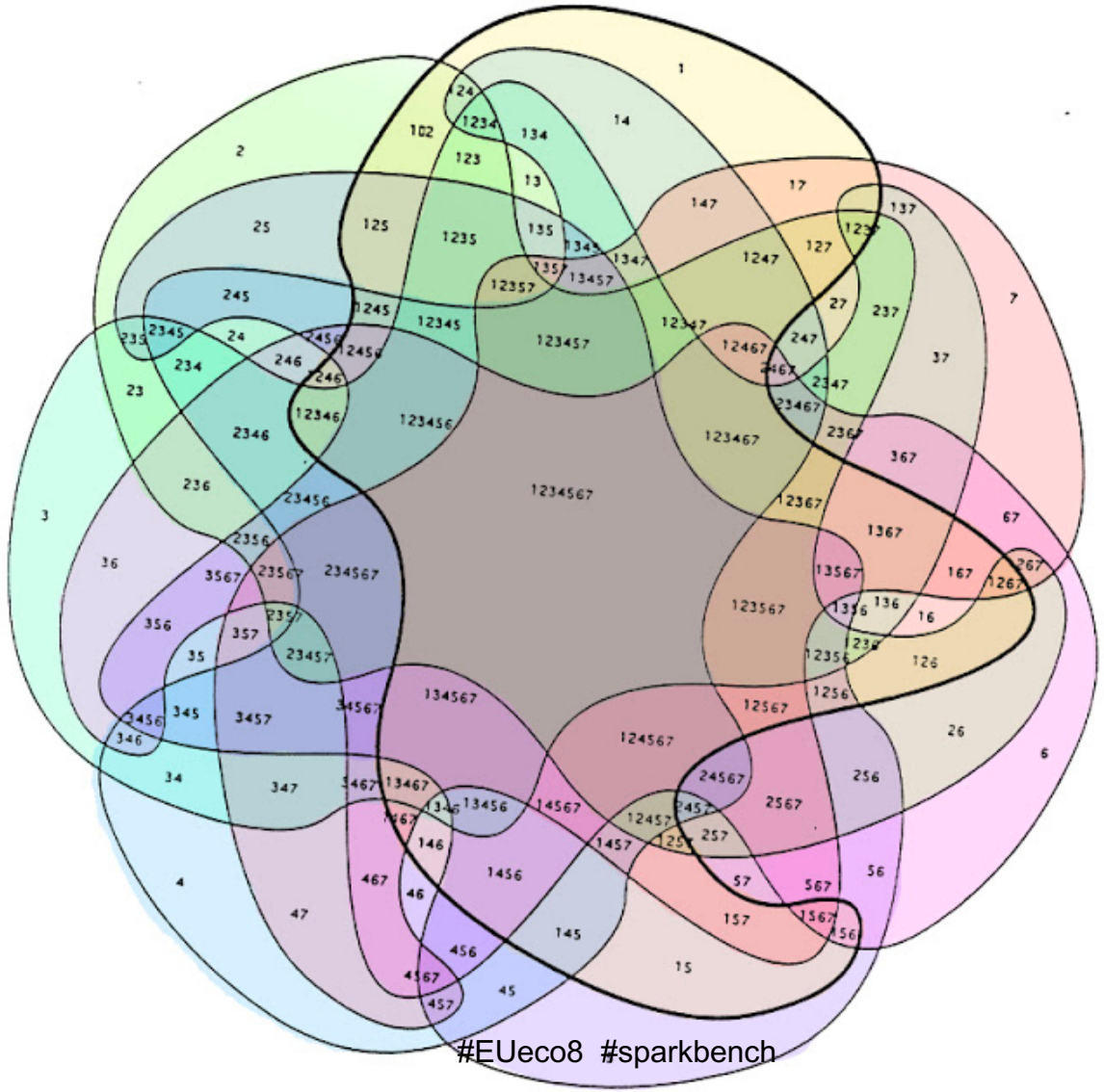
The second workload suite runs different workloads **serially**.

The second workload suite ***repeats***.



```
spark-bench = {
  spark-submit-parallel = false
  spark-submit-config = [{
    spark-args = { // . . . }
    suites-parallel = false
    workload-suites = [{
      descr = "Data Generation"
      parallel = true
      repeat = 1 // generate once and done!
      workloads = { // . . . }
    },
    workload-suites = [{
      descr = "Actual workloads that we want to benchmark"
      parallel = false
      repeat = 100 // repeat for statistical validity!
      workloads = { // . . . }
    }]
  }]
}
```

Beautiful Graphs



#EUeco8 #sparkbench

Detailed Example: Parquet vs. CSV

Example: Parquet vs. CSV

Goal: benchmark two different SQL queries over the same dataset stored in two different formats

1. Generate a big dataset in CSV format
2. Convert that dataset to Parquet
3. Run first query over both sets
4. Run second query over both sets
5. Make beautiful graphs

```
spark-bench = {
  spark-submit-config = [{
    spark-home = "/usr/iop/current/spark2-client/"
    spark-args = {
      master = "yarn"
      executor-memory = "28G"
      num-executors = 144
    }
    conf = {
      "spark.dynamicAllocation.enabled" = "false"
      "spark.dynamicAllocation.monitor.enabled" = "false"
      "spark.shuffle.service.enabled" = "true"
    }
    suites-parallel = false

// …CONTINUED…
```

```
workload-suites = [
  {
    descr = "Generate a dataset, then transform it to Parquet format"
    benchmark-output = "hdfs:///tmp/emily/results-data-gen.csv"
    parallel = false
    workloads = [
      {
        name = "data-generation-kmeans"
        rows = 10000000
        cols = 24
        output = "hdfs:///tmp/emily/kmeans-data.csv"
      },
      {
        name = "sql"
        query = "select * from input"
        input = "hdfs:///tmp/emily/kmeans-data.csv"
        output = "hdfs:///tmp/emily/kmeans-data.parquet"
      }
    ]
  },

// …CONTINUED…
```

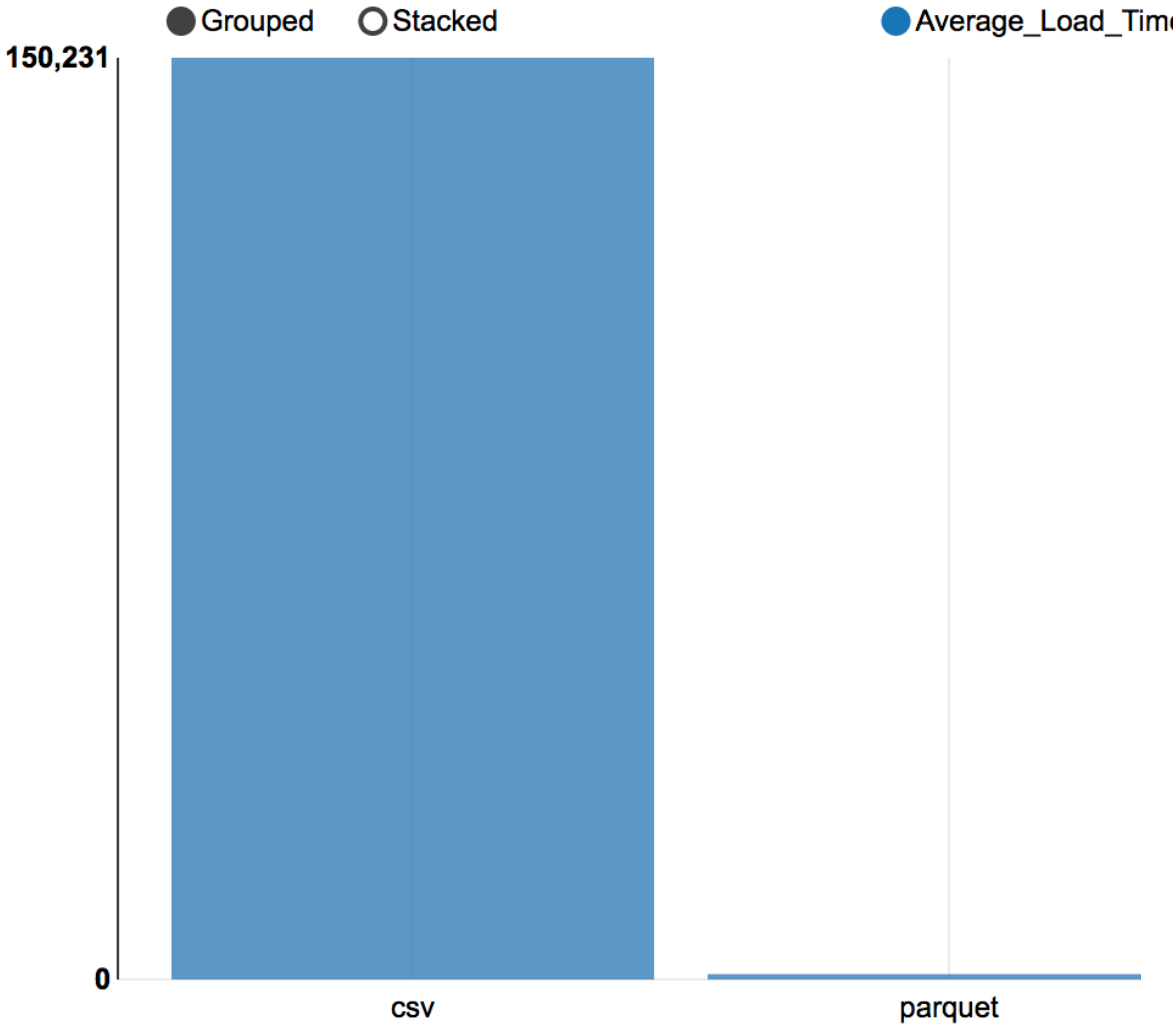
```
{
    descr = "Run two different SQL over two different formats"
    benchmark-output = "hdfs:///tmp/emily/results-sql.csv"
    parallel = false
    repeat = 10
    workloads = [
      {
        name = "sql"
        input = [
          "hdfs:///tmp/emily/kmeans-data.csv",
          "hdfs:///tmp/emily/kmeans-data.parquet"
        ]
        query = [
          "select * from input",
          "select `0`, `22` from input where `0` < -0.9"
        ]
        cache = false
      }
    ]
  }
 ]
}]
}
```

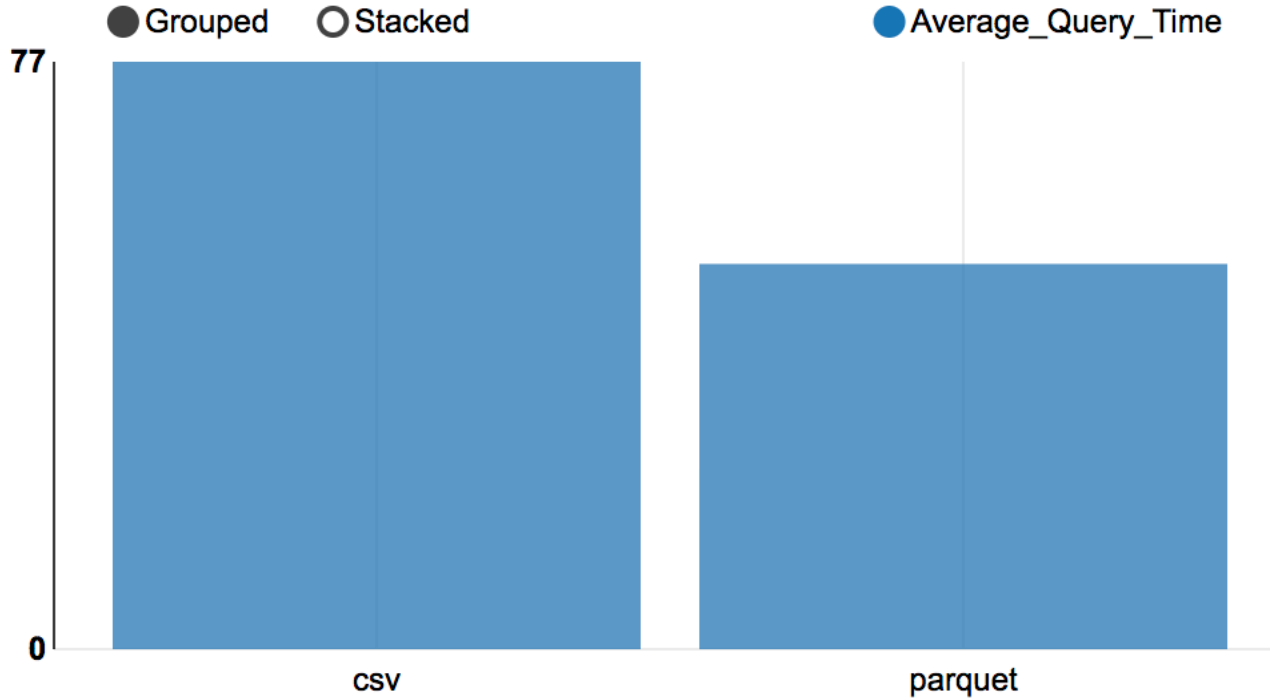
You made it through the plaintext! Here's a puppy!

Parquet vs CSV Results

name	timestamp	loadTime	queryTime	total_Runtime	run	cache	queryStr	input	spark.executor.instances	spark.submit.deployMode	spark.master	spa
sql	1.50714E+12	15895947762	8722036	15904669798	0	FALSE	select * from input	hdfs:///tmp/emily/kmeans-data.csv	144	client	yarn	28(
sql	1.50714E+12	15232039388	41572417	15273611805	0	FALSE	select `0`, `22` from input where `0` < -0.9	hdfs:///tmp/emily/kmeans-data.csv	144	client	yarn	28(
sql	1.50714E+12	139462342	5652725	145115067	0	FALSE	select * from input	hdfs:///tmp/emily/kmeans-data.parquet	144	client	yarn	28(
sql	1.50714E+12	100552926	9120325	109673251	0	FALSE	select `0`, `22` from input where `0` < -0.9	hdfs:///tmp/emily/kmeans-data.parquet	144	client	yarn	28(
sql	1.50714E+12	15221909508	6183625	15228093133	1	FALSE	select * from input	hdfs:///tmp/emily/kmeans-data.csv	144	client	yarn	28(
sql	1.50714E+12	15634952815	6820382	15641773197	1	FALSE	select `0`, `22` from input where `0` < -0.9	hdfs:///tmp/emily/kmeans-data.csv	144	client	yarn	28(
sql	1.50714E+12	93264481	5243927	98508408	1	FALSE	select * from input	hdfs:///tmp/emily/kmeans-data.parquet	144	client	yarn	28(
sql	1.50714E+12	84948895	5573010	90521905	1	FALSE	select `0`, `22` from input where `0` < -0.9	hdfs:///tmp/emily/kmeans-data.parquet	144	client	yarn	28(
sql	1.50714E+12	15270158275	8408445	15278566720	2	FALSE	select * from input	hdfs:///tmp/emily/kmeans-data.csv	144	client	yarn	28(
sql	1.50714E+12	14924461363	7999643	14932461006	2	FALSE	select `0`, `22` from input where `0` < -0.9	hdfs:///tmp/emily/kmeans-data.csv	144	client	yarn	28(
sql	1.50714E+12	86821299	13952212	100773511	2	FALSE	select * from input	hdfs:///tmp/emily/kmeans-data.parquet	144	client	yarn	28(
sql	1.50714E+12	100789178	6525153	107314331	2	FALSE	select `0`, `22` from input where `0` < -0.9	hdfs:///tmp/emily/kmeans-data.parquet	144	client	yarn	28(
sql	1.50714E+12	15115682178	7085577	15122767755	3	FALSE	select * from input	hdfs:///tmp/emily/kmeans-data.csv	144	client	yarn	28(
sql	1.50714E+12	14829435321	6452434	14835887755	3	FALSE	select `0`, `22` from input where `0` < -0.9	hdfs:///tmp/emily/kmeans-data.csv	144	client	yarn	28(
sql	1.50714E+12	78654779	5291044	83945823	3	FALSE	select * from input	hdfs:///tmp/emily/kmeans-data.parquet	144	client	yarn	28(
sql	1.50714E+12	86702691	5641506	92344197	3	FALSE	select `0`, `22` from input where `0` < -0.9	hdfs:///tmp/emily/kmeans-data.parquet	144	client	yarn	28(
sql	1.50714E+12	14881975292	7346304	14889321596	4	FALSE	select * from input	hdfs:///tmp/emily/kmeans-data.csv	144	client	yarn	28(
sql	1.50714E+12	14953655628	6749366	14960404994	4	FALSE	select `0`, `22` from input where `0` < -0.9	hdfs:///tmp/emily/kmeans-data.csv	144	client	yarn	28(
sql	1.50714E+12	102374878	4925237	107300115	4	FALSE	select * from input	hdfs:///tmp/emily/kmeans-data.parquet	144	client	yarn	28(
sql	1.50714E+12	79458610	4697124	84155734	4	FALSE	select `0`, `22` from input where `0` < -0.9	hdfs:///tmp/emily/kmeans-data.parquet	144	client	yarn	28(
sql	1.50714E+12	14908128878	7429576	14915558454	5	FALSE	select * from input	hdfs:///tmp/emily/kmeans-data.csv	144	client	yarn	28(
sql	1.50714E+12	14922630998	4596120	14927227118	5	FALSE	select `0`, `22` from input where `0` < -0.9	hdfs:///tmp/emily/kmeans-data.csv	144	client	yarn	28(
sql	1.50714E+12	79144522	4191275	83335797	5	FALSE	select * from input	hdfs:///tmp/emily/kmeans-data.parquet	144	client	yarn	28(
sql	1.50714E+12	86207011	4111788	90318799	5	FALSE	select `0`, `22` from input where `0` < -0.9	hdfs:///tmp/emily/kmeans-data.parquet	144	client	yarn	28(
sql	1.50714E+12	14909544529	5722627	14915267156	6	FALSE	select * from input	hdfs:///tmp/emily/kmeans-data.csv	144	client	yarn	28(
sql	1.50714E+12	14919548742	5643308	14925192050	6	FALSE	select `0`, `22` from input where `0` < -0.9	hdfs:///tmp/emily/kmeans-data.csv	144	client	yarn	28(
sql	1.50714E+12	85241859	4417683	89659542	6	FALSE	select * from input	hdfs:///tmp/emily/kmeans-data.parquet	144	client	yarn	28(
sql	1.50714E+12	93379964	4610765	97990729	6	FALSE	select `0`, `22` from input where `0` < -0.9	hdfs:///tmp/emily/kmeans-data.parquet	144	client	yarn	28(

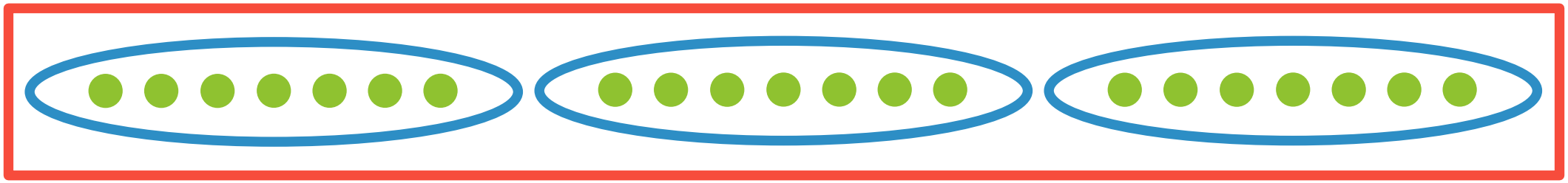
Beautiful Graphs



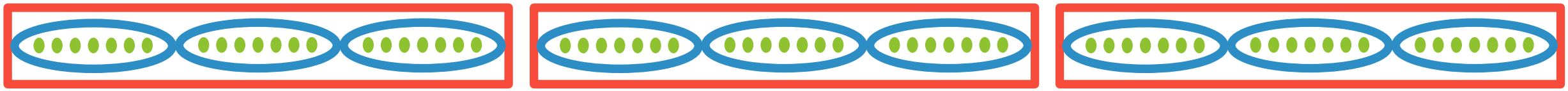


Examples: Cluster Hammer

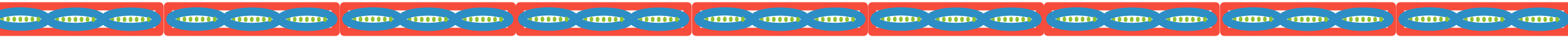
Cluster Hammer



Cluster Hammer



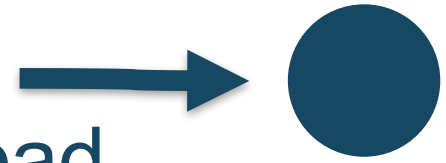
Cluster Hammer



Examples: Multi-User Notebooks

Multi-User Notebook

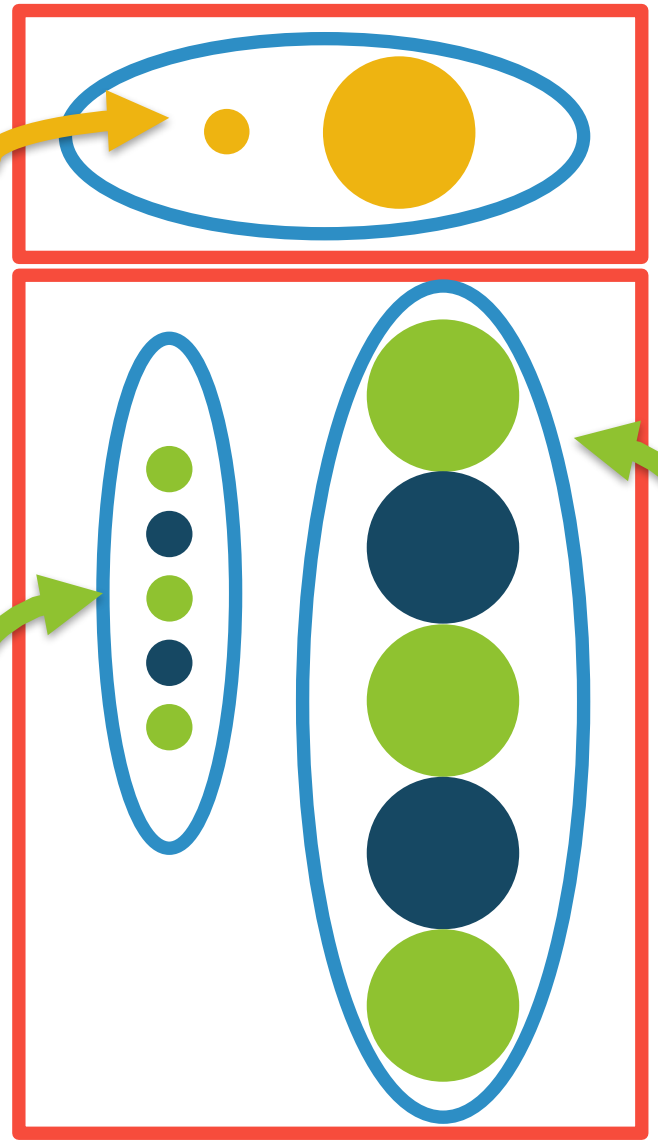
Sleep Workload



```
{
  name = "sleep"
  distribution = "poisson"
  mean = 30000
}
```

Data Generation,

sample.csv and GIANT-SET.csv



Workloads, working with the sample.csv

Same workloads, working with GIANT-SET.csv

Examples of Easy Extensions

- Add more "users" (workload suites)
- Have a workload suite that simulates heavy batch jobs while notebook users attempt to use the same cluster
- Toggle settings for dynamic allocation, etc.
- Benchmark a notebook user on a heavily stressed cluster (cluster hammer!)

Examples: Varying Spark Parameters

Different Versions of Spark

```
spark-bench = {
  spark-submit-parallel = false
  spark-submit-config = [{
    spark-home = [
      "/opt/official-spark-build/",
      "/opt/my-branch-of-spark-with-new-changes/"
    ]
    spark-args = {
      master = "yarn"
    }
    workload-suites = [{// workload suites}]
  }]
}
```

This will produce 2 spark-submits

Varying More Than One Param

```
spark-bench = {
  spark-submit-parallel = false
  spark-submit-config = [{
    spark-home = [
      "/opt/official-spark-build/",
      "/opt/my-fork-of-spark-with-new-changes/"
    ]
    spark-args = {
      master = "yarn"
      executor-mem = ["2G", "4G", "8G", "16G"]
    }
    workload-suites = [{// workload suites}]
  }]
}
```

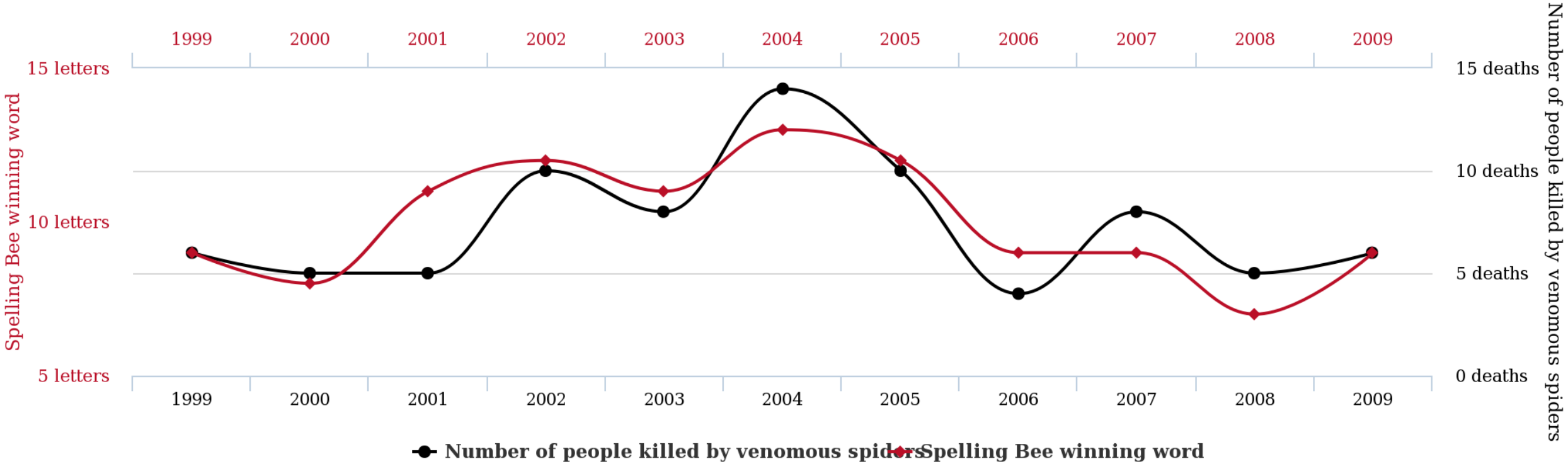
This will produce 2 X 4 = 8 spark-submits

Beautiful Graphs

Letters in Winning Word of Scripps National Spelling Bee

correlates with

Number of people killed by venomous spiders



Final Thoughts

Installation

1. Grab the latest release from the releases page on Github
2. Unpack the tar
3. Set the environment variable for $SPARK_HOME.
4. Run the examples!

```
./bin/spark-bench.sh examples/minimal-example.conf
```

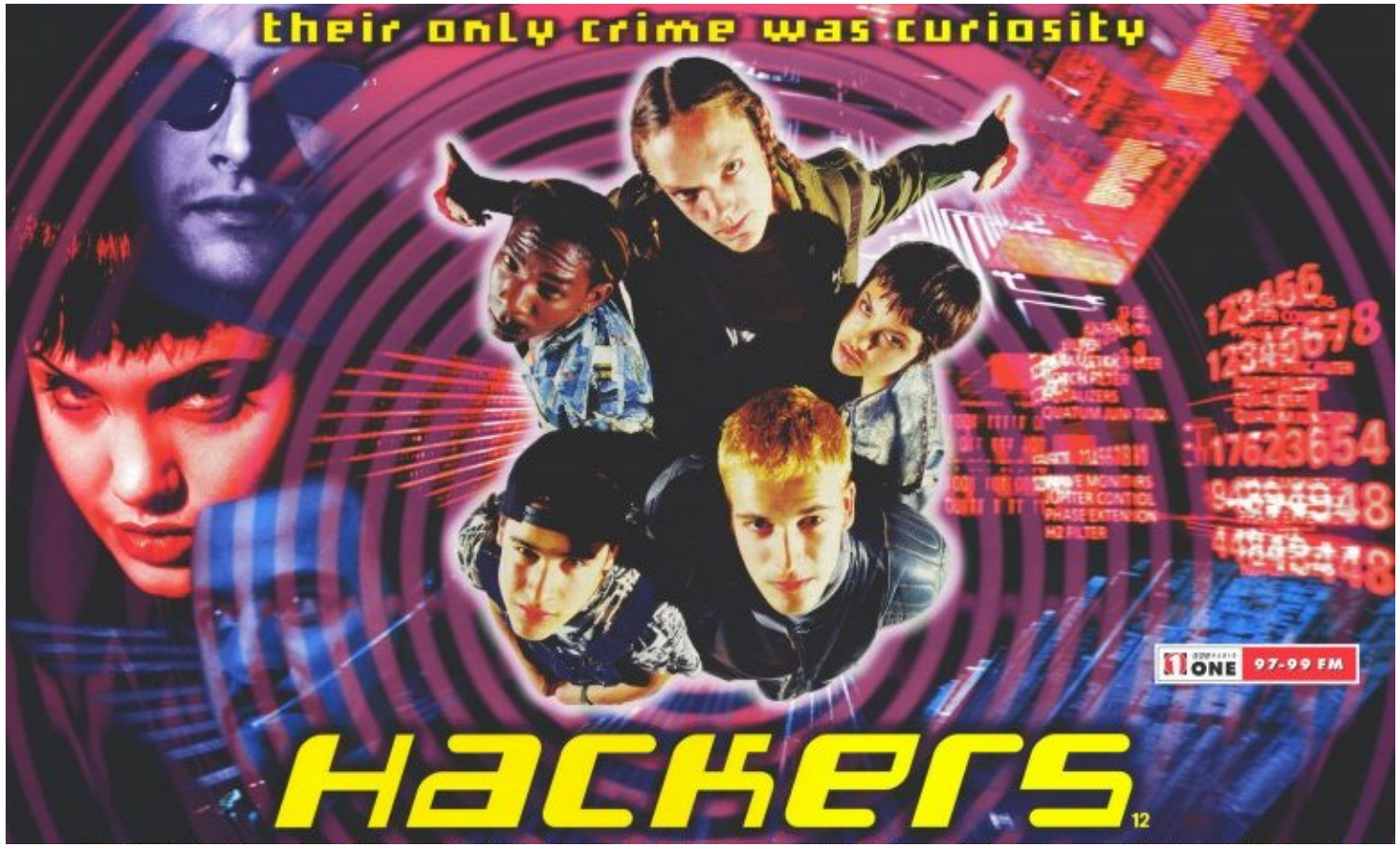
Future Work & Contributions

- More workloads and data generators
- TPC-DS (on top of existing SQL query benchmark)
- Streaming workloads
- Python

Contributions *very* welcome from all levels!

Check out our ZenHub Board for details

Shout-Out to My Team



*not an actual photo of my team

Matthew Schauer
@showermat

Craig Ingram
@cin

Brad Kaiser
@brad-kaiser

Summary

Spark-Bench is a flexible, configurable framework for

Testing, Benchmarking, Simulating, Comparing,

and more!

https://sparktc.github.io/spark-bench/


That's all Folks!

SparkTC/spark-bench

Emily May Curtin

IBM Watson Data Platform

Atlanta, GA

@emilymaycurtin

@ecurtin

#EUeco8

#sparkbench

Project

https://github.com/SparkTC/spark-bench

Documentation

https://sparktc.github.io/spark-bench/