An Introduction to PySpark

Alex Ware

An Introduction to PySpark

- What is PySpark?
- Can it solve all of my data problems?
- Are you sure I can’t just use pandas instead?

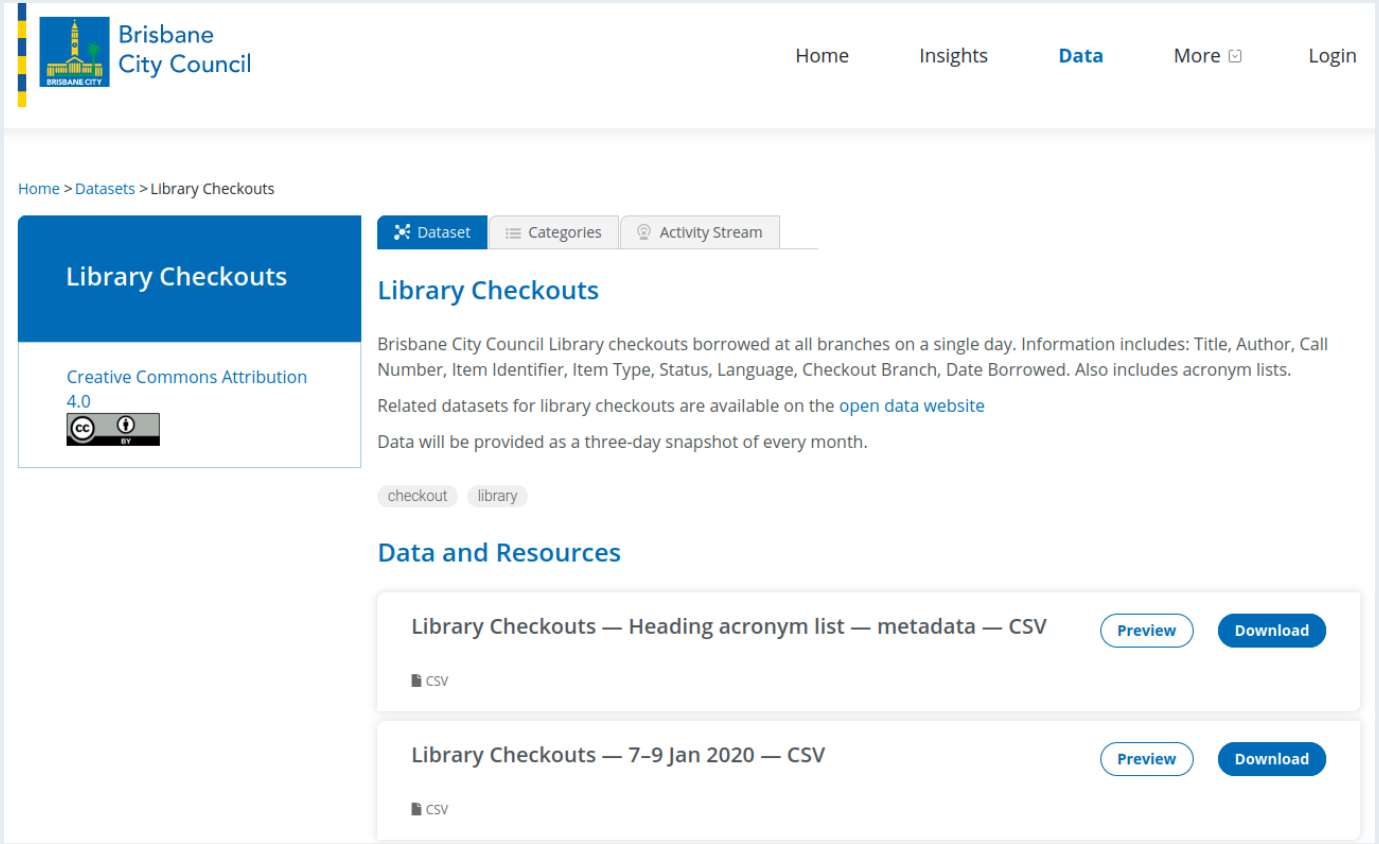
Alex Ware

- Software Engineer at Geoscape Australia
- Previously a Data Engineer in the Australian Public Service (APS)
- Co-organiser of the Canberra Python User Group and co-organiser of the recent Django Girls Canberra Workshop

Alex Ware

- Software Engineer at Geoscape Australia
- Previously a Data Engineer in the Australian Public Service (APS)
- Co-organiser of the Canberra Python User Group and co-organiser of the recent Django Girls Canberra Workshop

DISCLAIMER



https://www.data.brisbane.qld.gov.au/data/dataset/library-checkouts-branch-date

Pandas

```
import pandas as pd

df = pd.read_csv("data/library_checkouts_all_branches_jul_2023.csv")

df[["checkout_library", "author", "item_type", "language", "age"]].head()
```

✓ 0.0s

Pandas

```
import pandas as pd

df = pd.read_csv("data/library_checkouts_all_branches_jul_2023.csv")

df[["checkout_library", "author", "item_type", "language", "age"]].head()
```

✓ 0.0s

	checkout_library	author	item_type	language	age
0	BSQ	Cano, Carles,	LOTE-BOOK	SPANISH	JUVENILE
1	BSQ	Cano, Carles,	LOTE-BOOK	SPANISH	JUVENILE
2	BSQ	Dupuis, Sylvia,	LOTE-BOOK	SPANISH	JUVENILE
3	BSQ	Maudet, Matthieu.	LOTE-BOOK	SPANISH	JUVENILE
4	BSQ	Foreman, Michael,	LOTE-BOOK	SPANISH	JUVENILE

PySpark

```
from pyspark.sql import SparkSession, functions as f, Window

spark = (SparkSession
            .builder
            .appName("Demo")
            .getOrCreate())

df = spark.read.csv("data/library_checkouts_all_branches_jul_2023.csv", header=True, escape="\"")

df.select("checkout_library", "author", "item_type", "language", "age").show(5)
```

✓ 6.0s

PySpark

```
from pyspark.sql import SparkSession, functions as f, Window

spark = (SparkSession
            .builder
            .appName("Demo")
            .getOrCreate())

df = spark.read.csv("data/library_checkouts_all_branches_jul_2023.csv", header=True, escape="\"")

df.select("checkout_library", "author", "item_type", "language", "age").show(5)
```

✓ 6.0s

```
+---------------+----------------+---------+--------+--------+
|checkout_library|          author|item_type|language|     age|
+---------------+----------------+---------+--------+--------+
|            BSQ|    Cano, Carles,|LOTE-BOOK| SPANISH|JUVENILE|
|            BSQ|    Cano, Carles,|LOTE-BOOK| SPANISH|JUVENILE|
|            BSQ|  Dupuis, Sylvia,|LOTE-BOOK| SPANISH|JUVENILE|
|            BSQ|Maudet, Matthieu.|LOTE-BOOK| SPANISH|JUVENILE|
|            BSQ|Foreman, Michael,|LOTE-BOOK| SPANISH|JUVENILE|
+---------------+----------------+---------+--------+--------+
only showing top 5 rows
```

Pandas

```
print("\nQuick stats\n")

print("Number of checkouts:", df.shape[0])

df_language = (df[["language"]].loc[~df.language.isnull()]
                               .drop_duplicates())

print("Number of languages:", df_language.shape[0])

df_pop = (df[["age", "item_id"]].groupby(by=["age"])
                                .count()
                                .rename(columns={"item_id": "count"})
                                .sort_values("count", ascending=False))

print("\nPopularity by age:")
df_pop.head()
```

✓ 0.0s

```
Quick stats

Number of checkouts: 50487
Number of languages: 31

Popularity by age:
```

age	count
JUVENILE	24740
ADULT	24477
YA	1260
POLICY NOT FOUND	10

PySpark

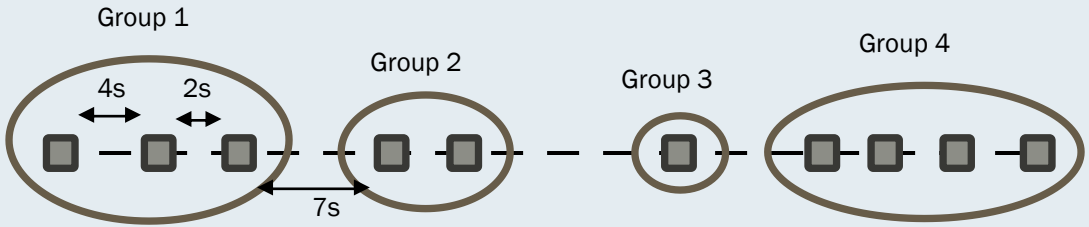
```
print("\nQuick stats\n")

print("Number of checkouts:", df.count())

df_language = (df.select("language")
               .filter(f.col("language").isNotNull())
               .distinct())

print("Number of languages:", df_language.count())

df_age = df.groupby("age").count().orderBy("count", ascending=False)

print("\nPopularity by age:")
df_age.show()
```

✓ 1.3s

```
Quick stats

Number of checkouts: 50487
Number of languages: 31

Popularity by age:
+----------------+-----+
|             age|count|
+----------------+-----+
|        JUVENILE|24740|
|           ADULT|24477|
|              YA| 1260|
|POLICY NOT FOUND|   10|
+----------------+-----+
```

Scenario

Group any checkouts that occur within 5 seconds of each other in the same library



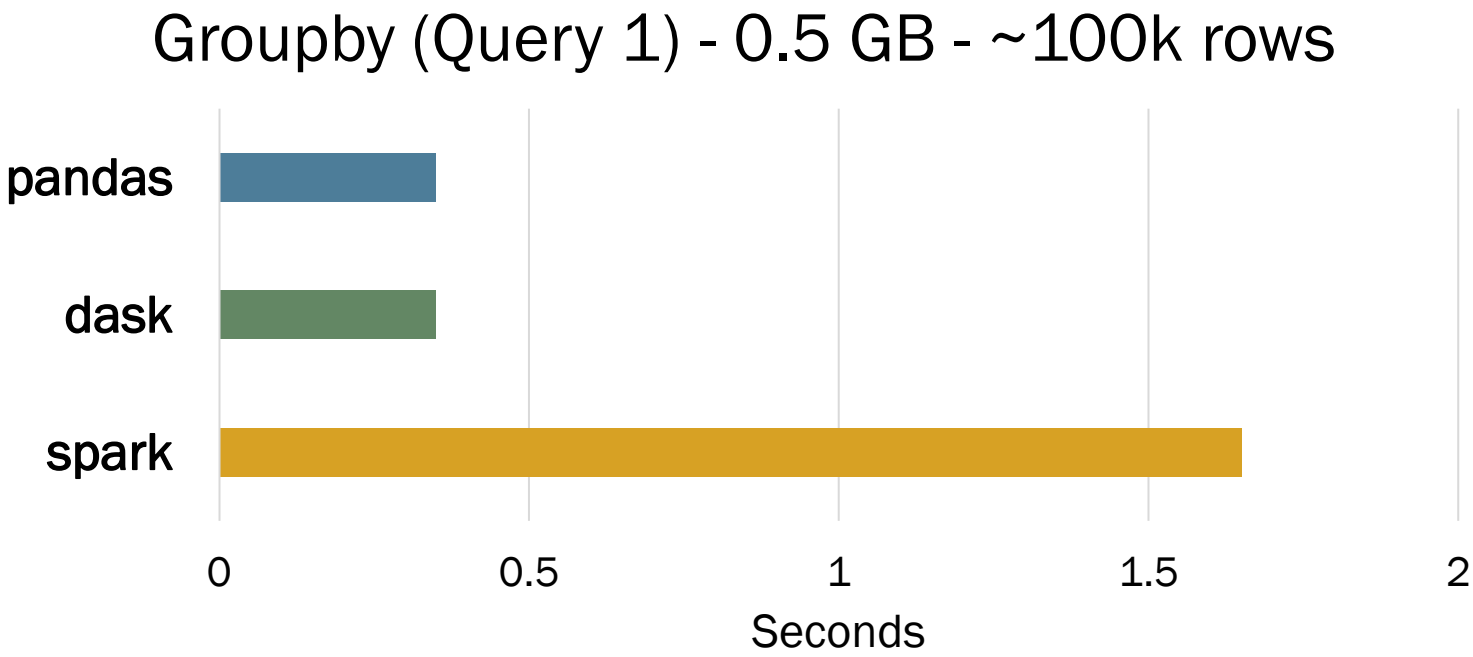
PySpark

```python
def group_records_into_checkouts(df, allowed_gap_in_seconds):
    """Assumes that two books that are borrowed out within x seconds of eachother in the same library can be grouped into a single checkout"""

    # Grab the time of the immediately previous checkout
    library_partition = Window.partitionBy("checkout_library").orderBy("date", "item_id")
    df_checkout = df.withColumn("previous_checkout", f.lag(f.col("date")).over(library_partition))

    # Only consider it to be the same checkout if the previous occurred within the previous X seconds
    df_checkout = (df_checkout.withColumn("time_between_checkouts", f.unix_timestamp("date") - f.unix_timestamp("previous_checkout"))
                              .withColumn("is_same_checkout", f.col("time_between_checkouts") <= allowed_gap_in_seconds)
                              .withColumn("new_checkout_increment", f.when(f.col("is_same_checkout"), 0).otherwise(1))
                              .drop("is_same_checkout"))

    # The checkout group IDs can then be generated by doing a cumulative sum
    df_checkout = df_checkout.withColumn("group_id", f.sum("new_checkout_increment")
                                                      .over(library_partition.rangeBetween(Window.unboundedPreceding, 0)))

    return df_checkout.drop("row_number", "previous_checkout", "new_checkout_increment")
```

✓ 0.0s

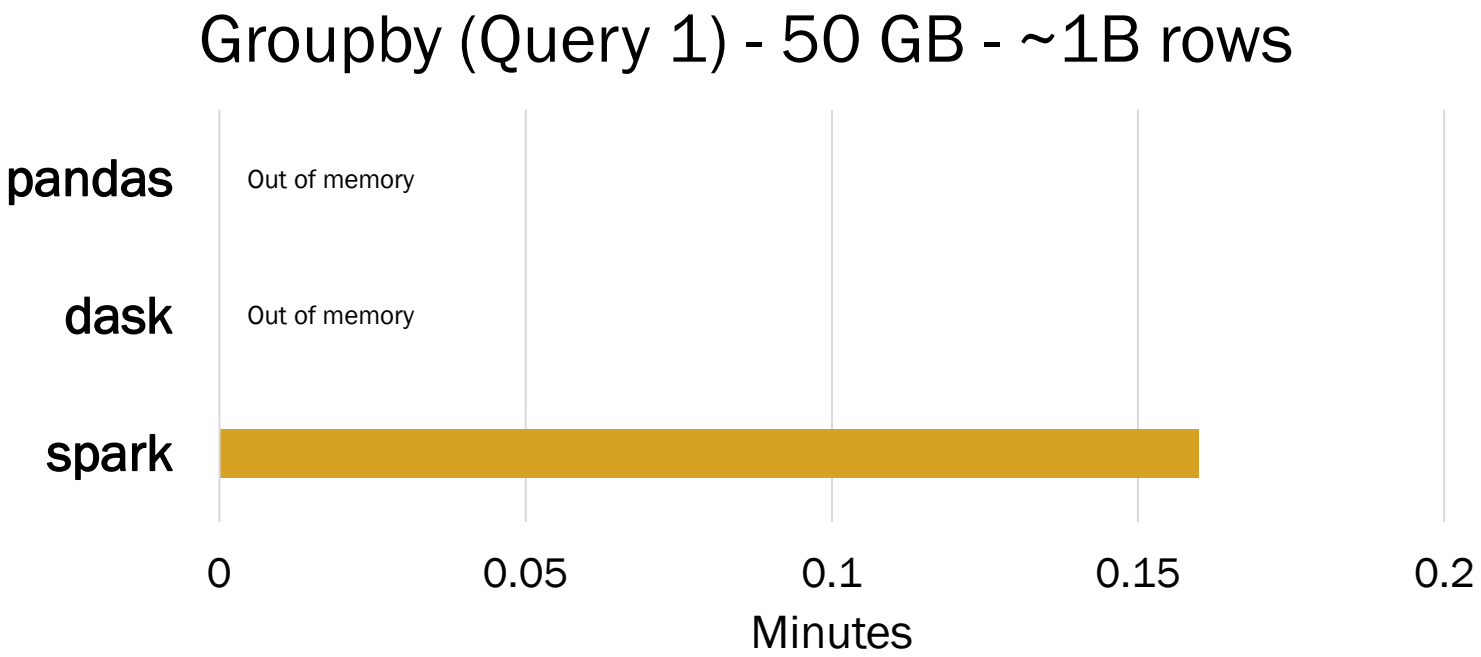
checkout_library	title	author	item_type	age	date
CDE	FASTBACK - Really weird!	Do, Anh,	FBKIDS	JUVENILE	2023-07-09 12:13:01
CDE	The spectacular revenge of Suzi Sims /	French, Vivian,	JU-PBK	JUVENILE	2023-07-09 12:13:03
CDE	FASTBACK - Camping time!	Do, Anh,	FBKIDS	JUVENILE	2023-07-09 12:13:05
CDE	FASTBACK - Hotdog!	Do, Anh,	FBKIDS	JUVENILE	2023-07-09 12:13:07
CDE	FASTBACK - Messy weird!	Do, Anh,	FBKIDS	JUVENILE	2023-07-09 12:13:09
CDE	FASTBACK - Crazy weird!	Do, Anh,	FBKIDS	JUVENILE	2023-07-09 12:13:11
CDE	FASTBACK - Treehouse tales too silly to be told until now!	Griffiths, Andy,	FBKIDS	JUVENILE	2023-07-09 12:13:14
CDE	FASTBACK - Circus time!	Do, Anh,	FASTBACK	JUVENILE	2023-07-09 12:13:15
CDE	Trolls world tour : the movie novel /	Lewman, David,	JU-PBK	JUVENILE	2023-07-09 12:13:18
CDE	Dragon's merry Christmas /	Pilkey, Dav,	JU-PBK	JUVENILE	2023-07-09 12:13:19
CDE	FASTBACK - Splashy weird!	Do, Anh,	FBKIDS	JUVENILE	2023-07-09 12:13:20
CDE	Race to Crashpoint Tower /	Older, Daniel Jos,	JU-PBK	JUVENILE	2023-07-09 12:13:23
CDE	Shark school /	Ocean, Davy,	JU-PBK	JUVENILE	2023-07-09 12:13:24
CDE	The one and only Ivan /	Applegate, Katherine.	JU-PBK	JUVENILE	2023-07-09 12:13:26
CDE	Conjuror cow : with flaps to lift /	Donaldson, Julia,	PICTURE-BK	JUVENILE	2023-07-09 12:13:30
CDE	Dippy and the dinosaurs /	French, Jackie,	PICTURE-BK	JUVENILE	2023-07-09 12:13:30
CDE	How do you say I love you? /	Barton, Ashleigh,	PICTURE-BK	JUVENILE	2023-07-09 12:13:30
CDE	Room on the broom : touch and feel book /	Donaldson, Julia,	JU-BOARD	JUVENILE	2023-07-09 12:13:30
CDE	Tricky's bad day /	Lester, Alison,	PICTURE-BK	JUVENILE	2023-07-09 12:13:30
CDE	Rocky to the rescue! /	McManus, Rove,	JU-PBK	JUVENILE	2023-07-09 12:13:31
CDE	Yucky mucky manners /	Lloyd, Sam,	PICTURE-BK	JUVENILE	2023-07-09 12:13:31
CDE	The cartoons that saved the world /	Ellen, Tom,	JU-PBK	JUVENILE	2023-07-09 12:13:35
CDE	I survived the sinking of the Titanic, 1912 /	Ball, Georgia,	JU-GRAPHIC	JUVENILE	2023-07-09 12:13:38
CDE	Doom's day camp.	Hauke, Joshua,	JU-GRAPHIC	JUVENILE	2023-07-09 12:13:40
CDE	Miles Morales : shock waves /	Reynolds, Justin A.,	JU-PBK	JUVENILE	2023-07-09 12:13:43
CDE	Wings of fire. the graphic novel /	Sutherland, Tui,	JU-PBK	JUVENILE	2023-07-09 12:13:44
CDE	Deltora quest.	Niwano, Makato,	GRAPHICNOV	YA	2023-07-09 12:13:47
CDE	Best of pests /	Harris, Tim,	JU-PBK	JUVENILE	2023-07-09 12:13:48
CDE	Apocalypse cow! /	Edmonds, Guy,	JU-PBK	JUVENILE	2023-07-09 12:13:51
CDE	Don't worry, Murray /	Stein, David Ezra,	PICTURE-BK	JUVENILE	2023-07-09 12:13:55
CDE	Mr. Cool /	Hargreaves, Roger.	PICTURE-BK	JUVENILE	2023-07-09 12:13:59
CDE	Barbie in a mermaid tale : Barbie in a mermaid tale 2	null	DVD	JUVENILE	2023-07-09 12:14:00
CDE	UglyDolls	null	DVD	JUVENILE	2023-07-09 12:14:00
CDE	Little Miss Scary /	Hargreaves, Roger,	PICTURE-BK	JUVENILE	2023-07-09 12:14:05
CDE	Mr. Happy /	Hargreaves, Roger,	PICTURE-BK	JUVENILE	2023-07-09 12:14:05
CDE	Mr. Tall /	Hargreaves, Roger.	PICTURE-BK	JUVENILE	2023-07-09 12:14:05
CDE	Little Bunny's book of thoughts /	Smallman, Steve,	PICTURE-BK	JUVENILE	2023-07-09 12:14:09
CDE	Mr. Rush /	Hargreaves, Roger.	PICTURE-BK	JUVENILE	2023-07-09 12:14:09

So, what should I use?



https://h2oai.github.io/db-benchmark/

So, what should I use?



https://h2oai.github.io/db-benchmark/

Java?

pyspark.sql.DataFrame.count

```
    def count(self) -> int:                                                    [docs]
        """Returns the number of rows in this :class:`DataFrame`.

        .. versionadded:: 1.3.0

        Examples
        --------
        >>> df.count()
        2
        """
        return int(self._jdf.count())
```

Java?

pyspark.sql.DataFrame.collect

```
    def collect(self) -> List[Row]:                                    [docs]
        """Returns all the records as a list of :class:`Row`.

        .. versionadded:: 1.3.0

        Examples
        --------
        >>> df.collect()
        [Row(age=2, name='Alice'), Row(age=5, name='Bob')]
        """
```

Java?!

pyspark.sql.DataFrame.collect

```
    def collect(self) -> List[Row]:                                    [docs]
        """Returns all the records as a list of :class:`Row`.

        .. versionadded:: 1.3.0

        Examples
        --------
        >>> df.collect()
        [Row(age=2, name='Alice'), Row(age=5, name='Bob')]
        """
        with SCCallSiteSync(self._sc):
            sock_info = self._jdf.collectToPython()
        return list(_load_from_socket(sock_info, BatchedSerializer(CPickleSerializer())))
```

https://www.databricks.com/glossary/pyspark

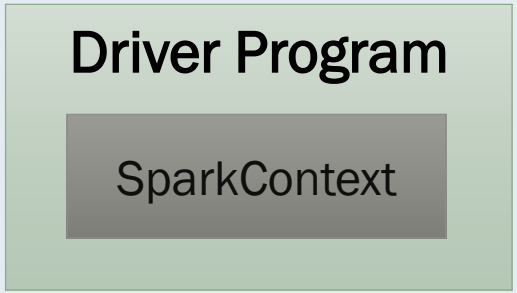
What if…

functional programming…

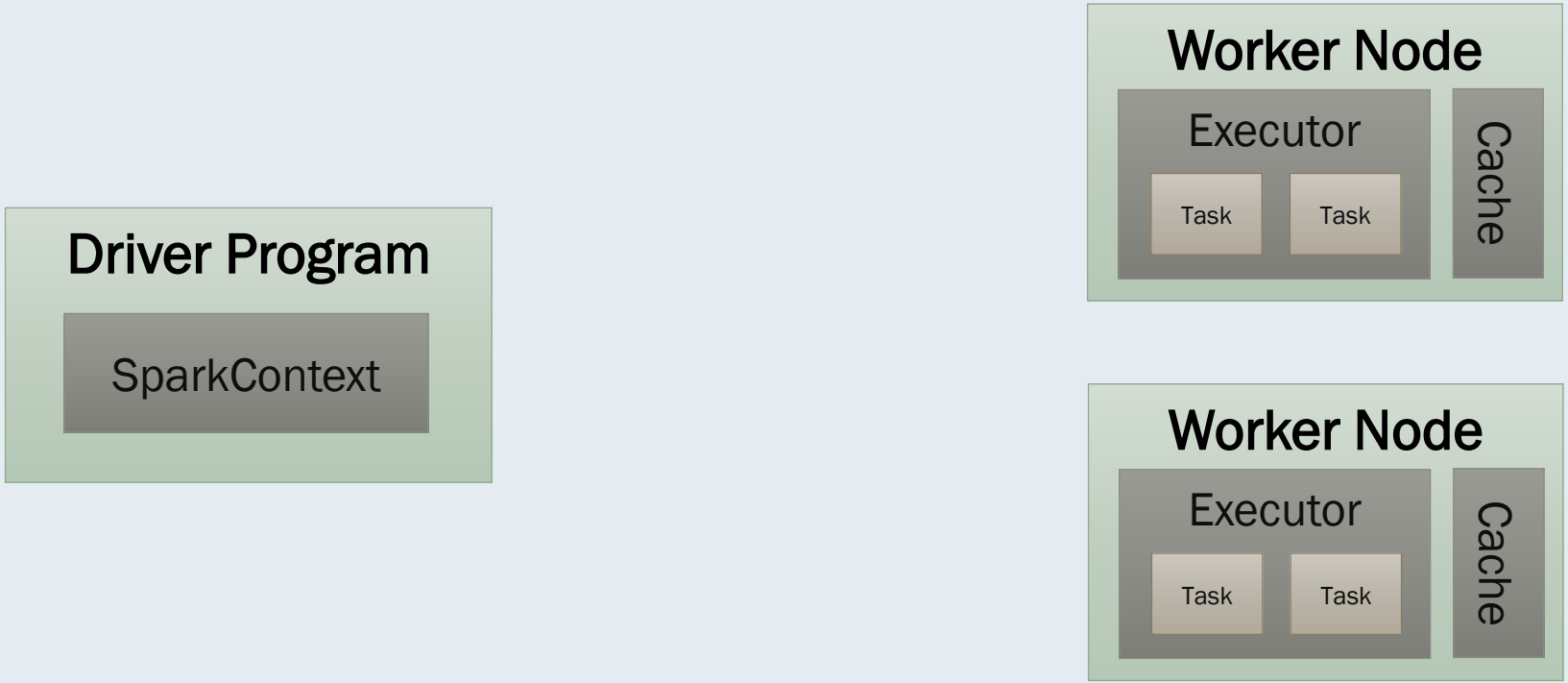
on data…

and distributed!

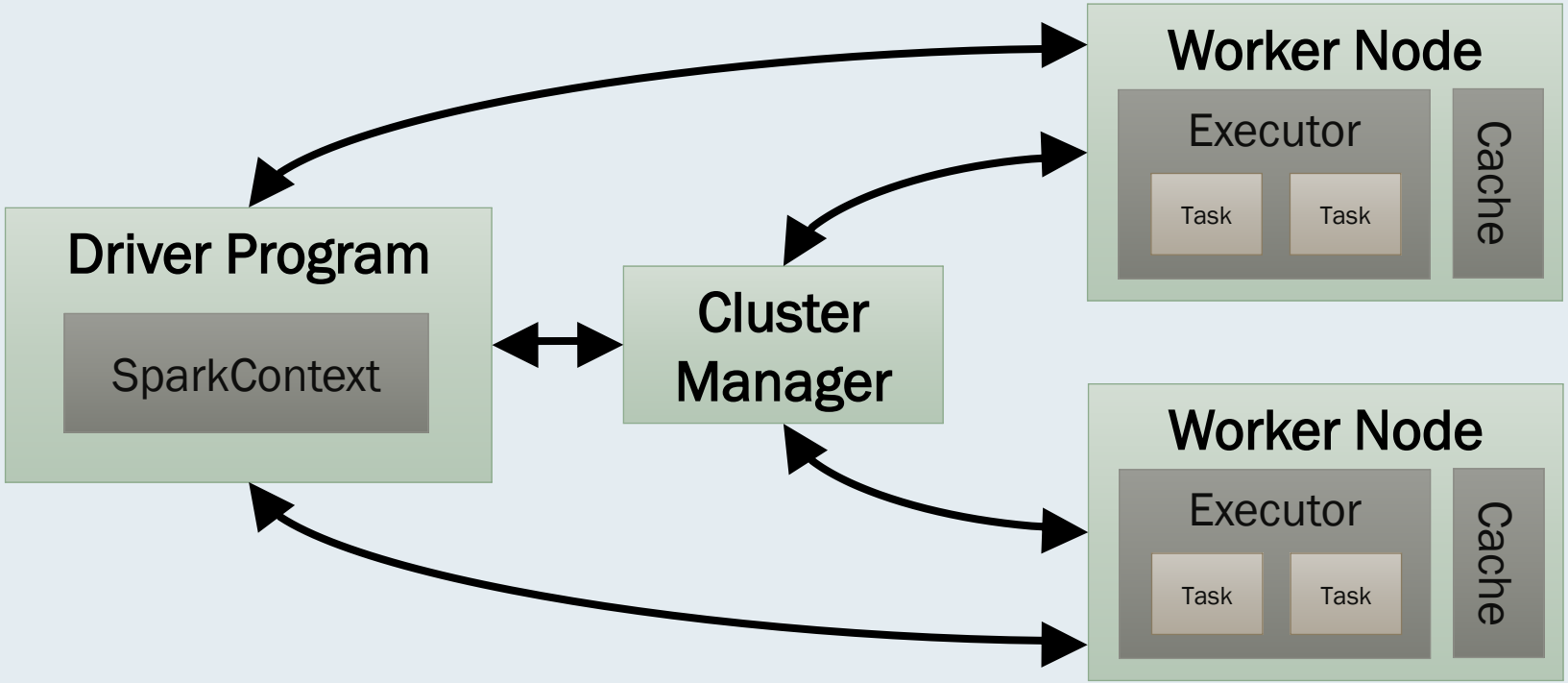
Distribution!



Distribution!



Distribution!



Spark Master at spark://

URL:
Alive Workers: 0
Cores in use: 0 Total, 0 Used
Memory in use: 0.0 B Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

▾ Workers (0)

Worker Id	Address	State	Cores	Memory	Resources

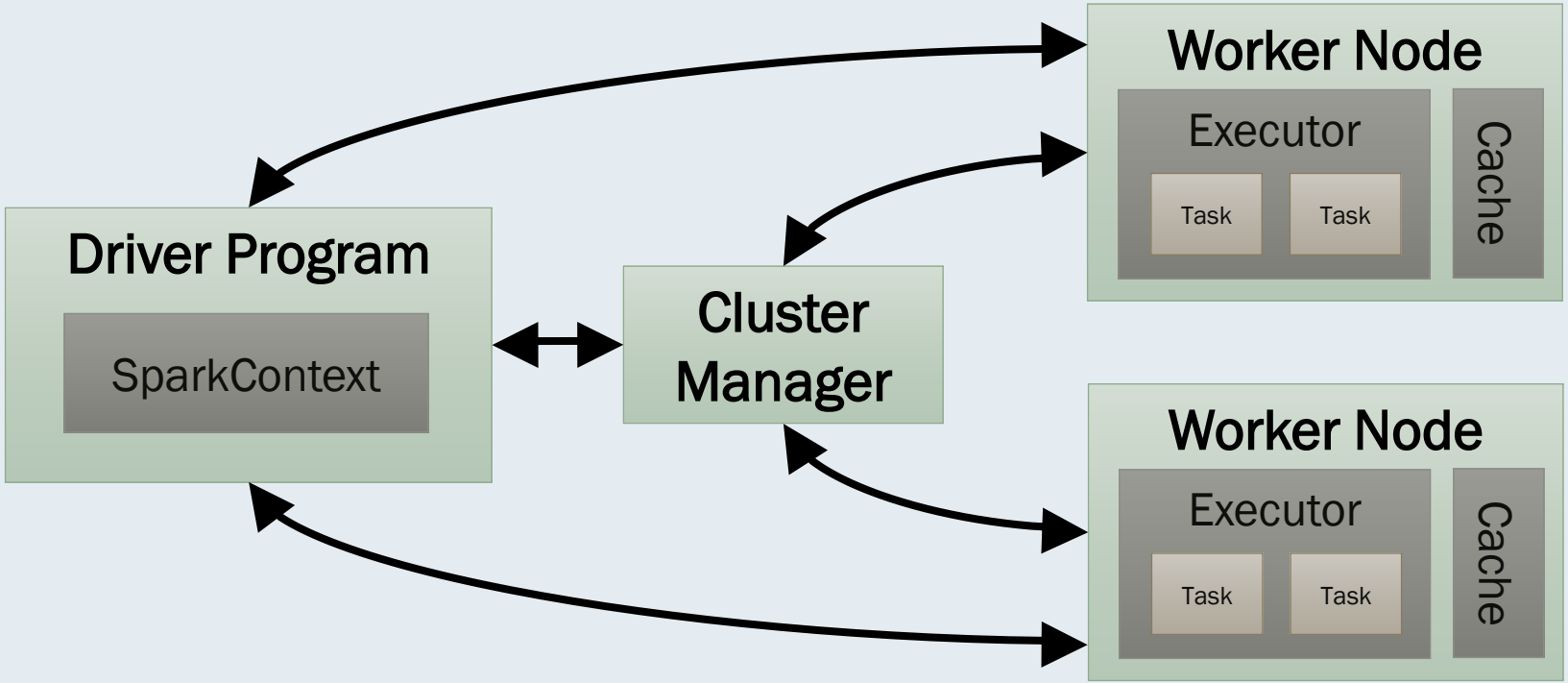
▾ Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

▾ Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Distribution!



Because Matei Zaharia wanted

*to do **MapReduce** but better*

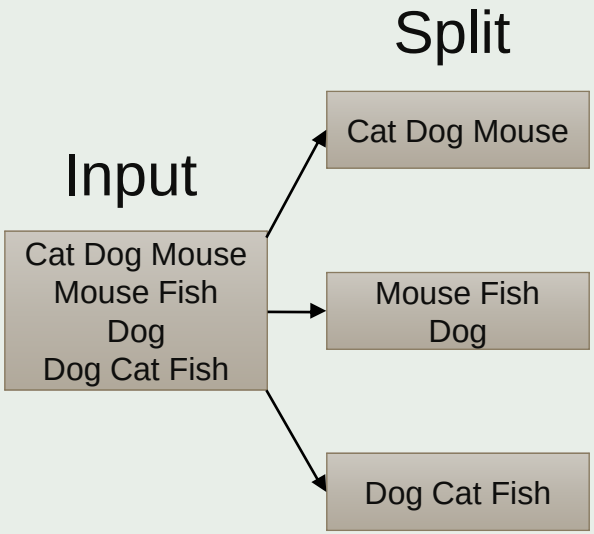
Let’s count words

Let’s count words

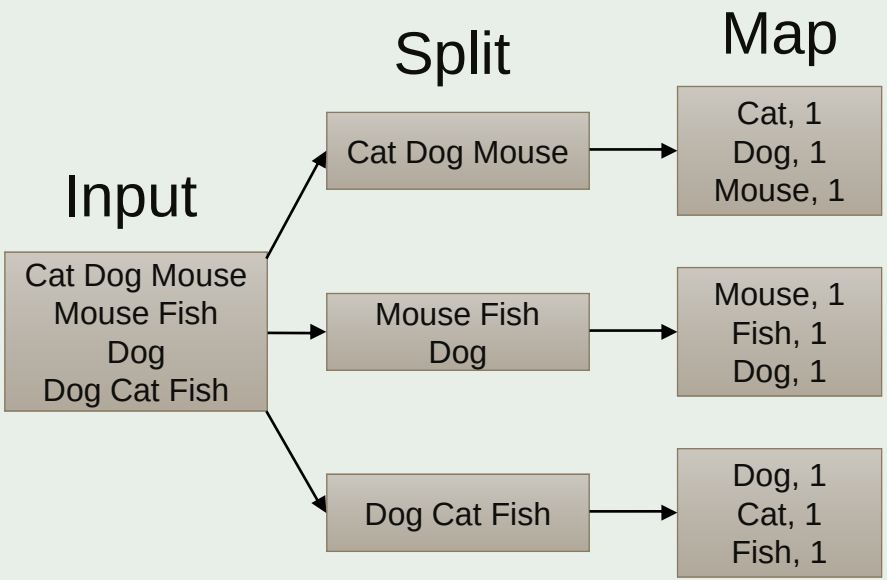
Input

```
Cat Dog Mouse
Mouse Fish
Dog
Dog Cat Fish
```

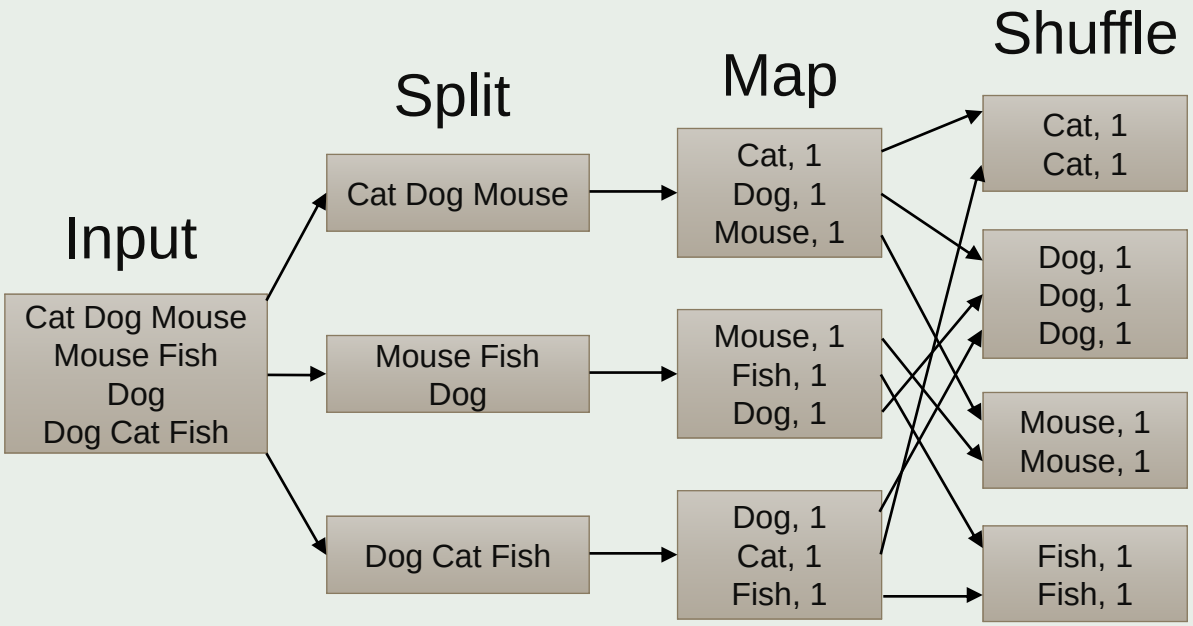
Let’s count words



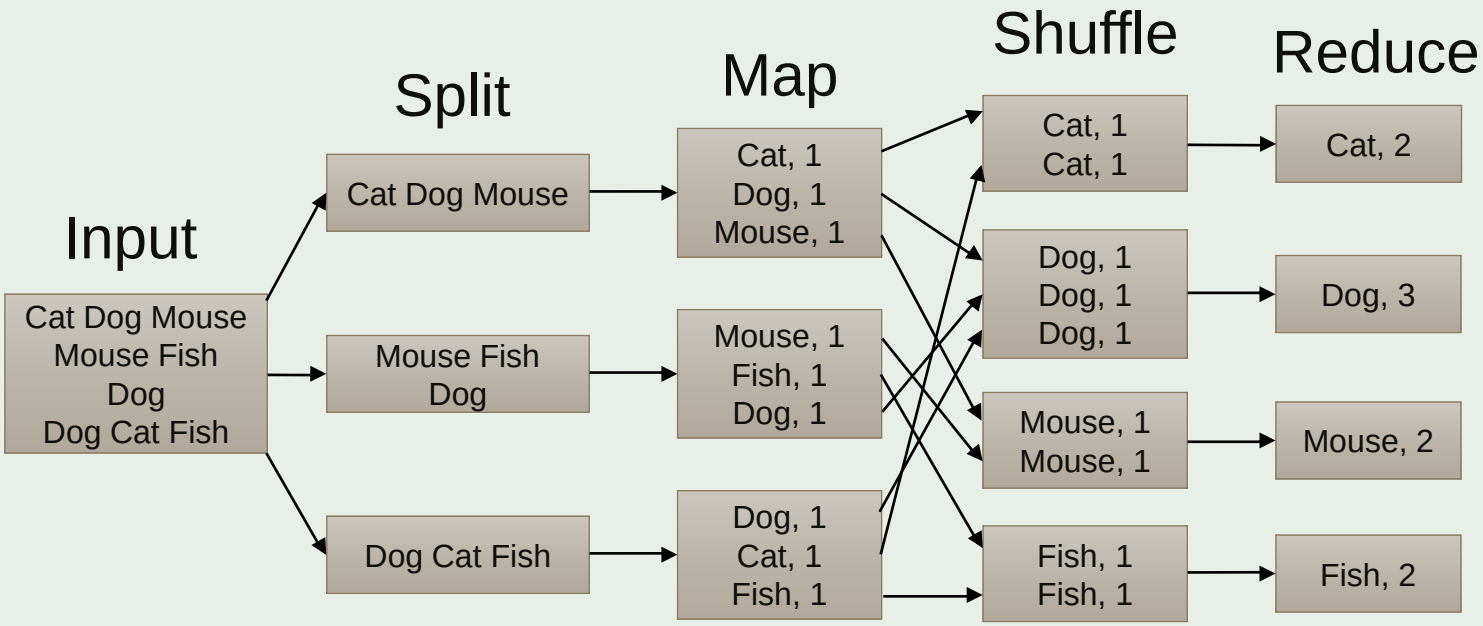
Let’s count words



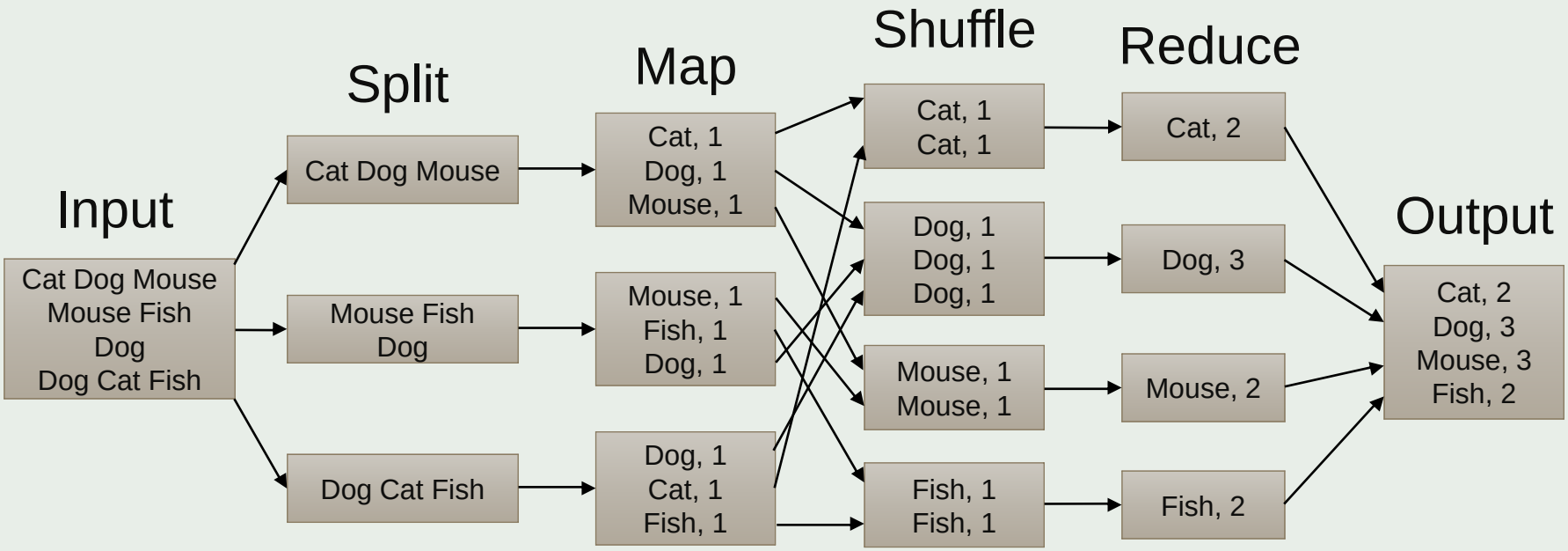
Let’s count words



Let’s count words



Let’s count words



Because Matei Zaharia wanted

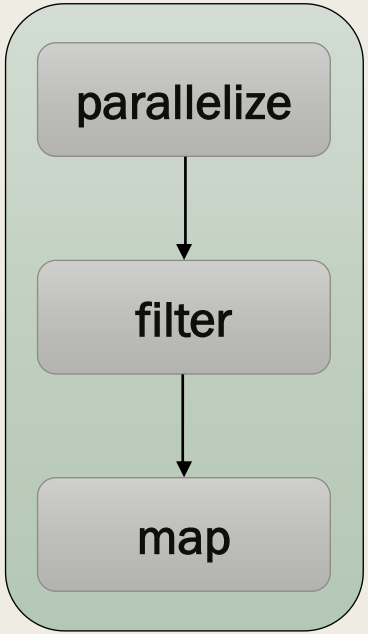
*to do **MapReduce** but better*

Transformations *- lazy, added to the plan, generally take a dataframe and return a dataframe*

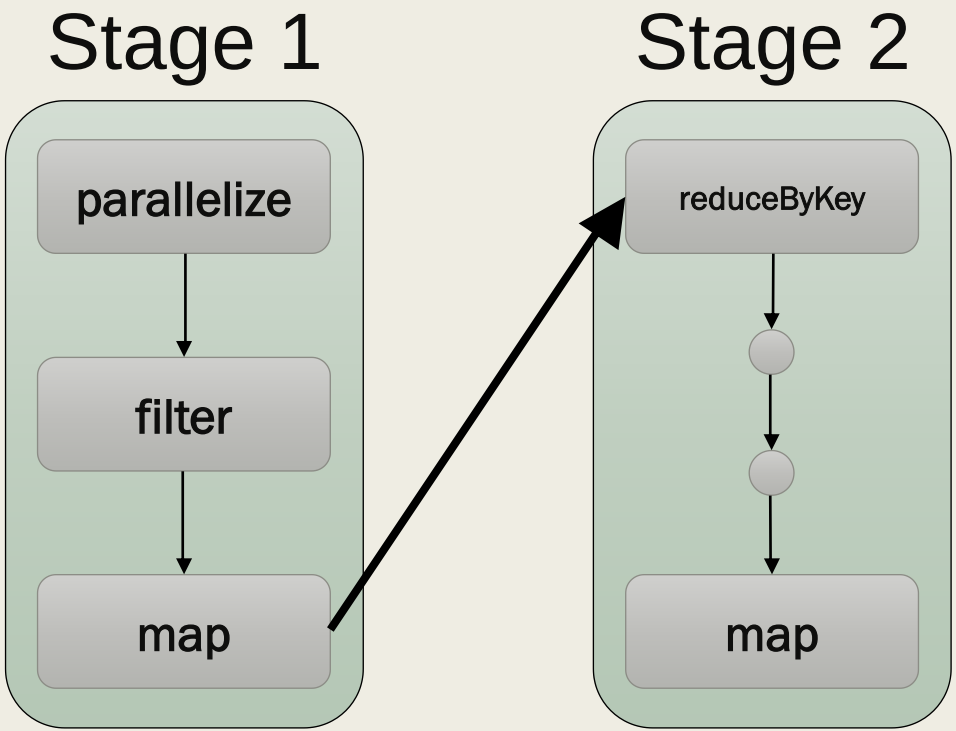
Directed Acylic Graphs

Directed Acylic Graphs

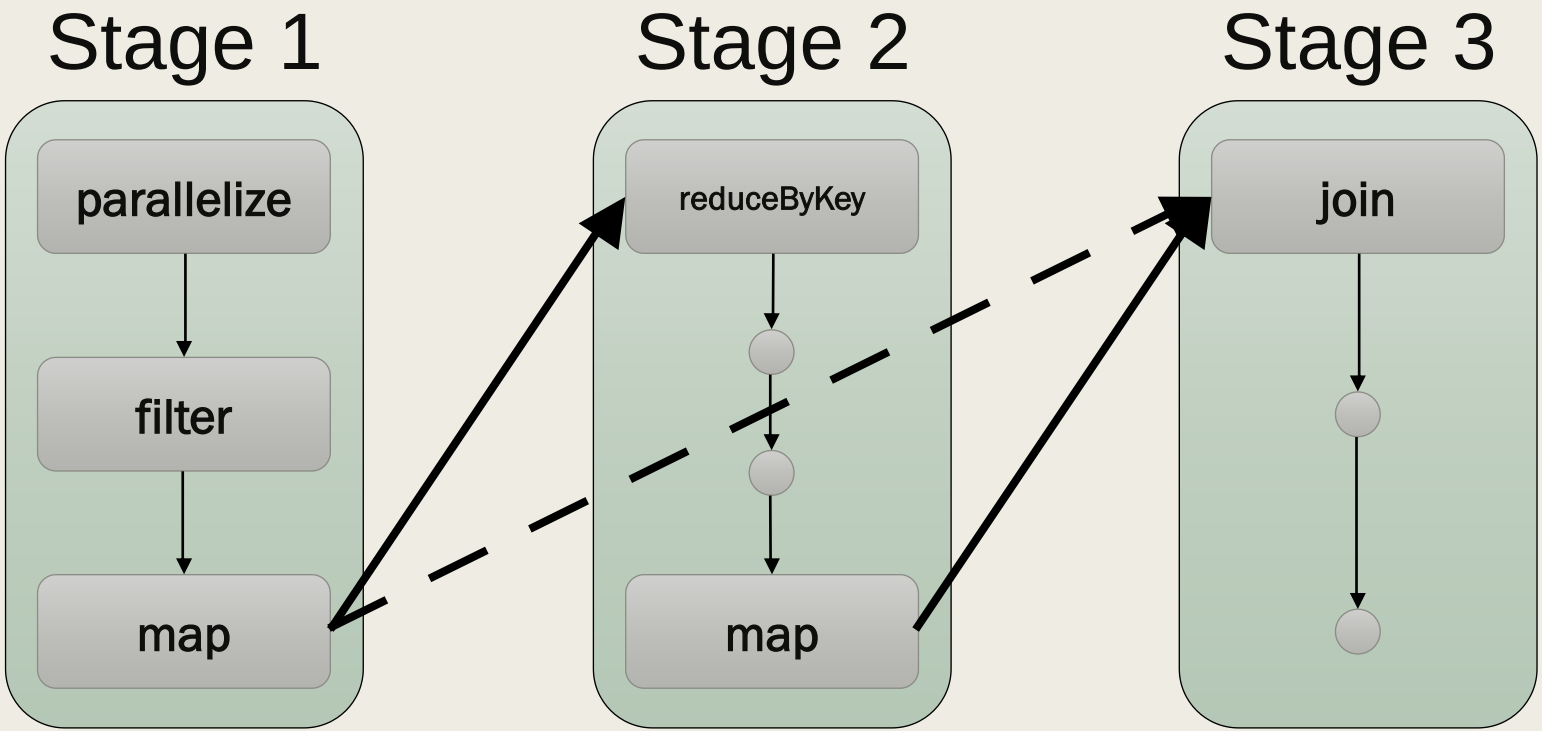
Stage 1



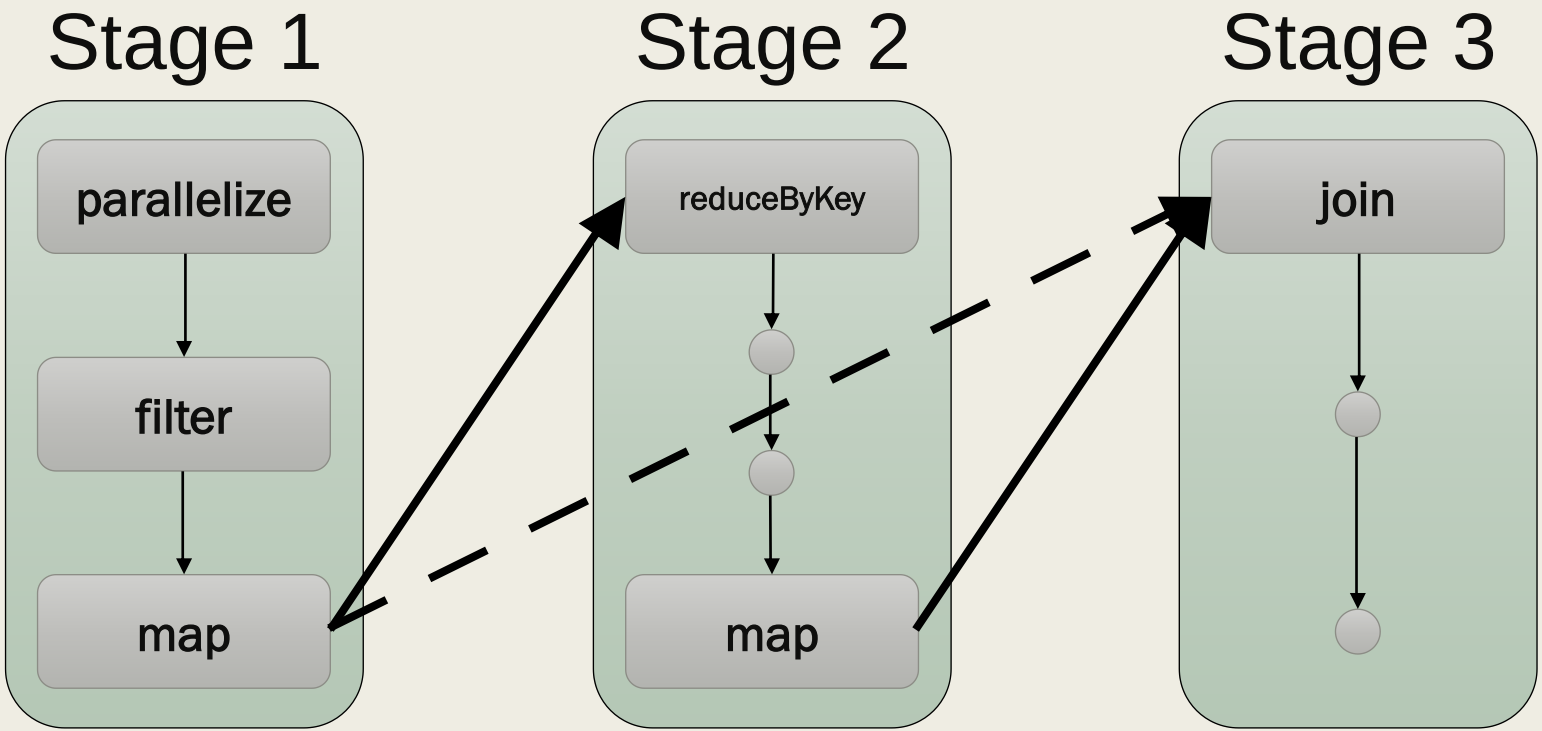
Directed Acylic Graphs



Directed Acylic Graphs

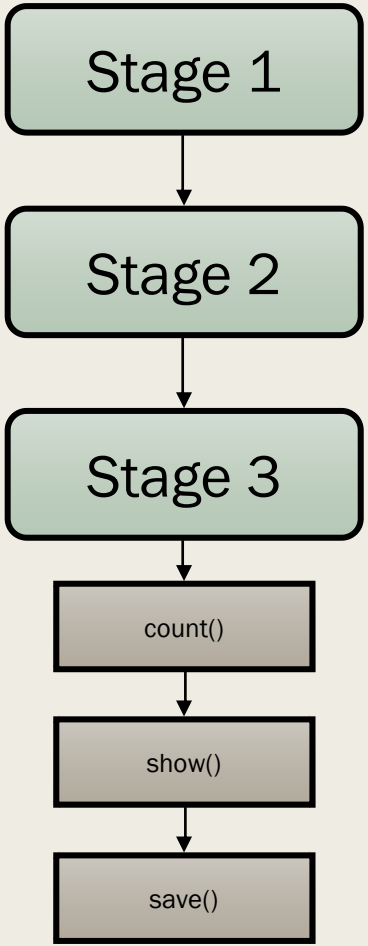


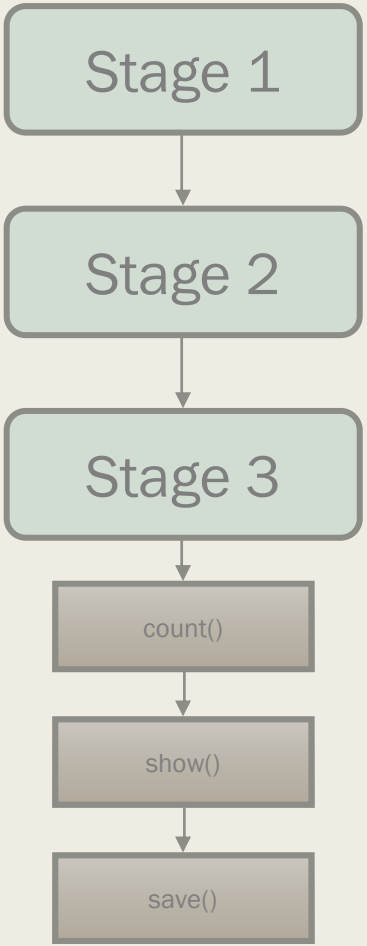
java.lang.OutOfMemoryError

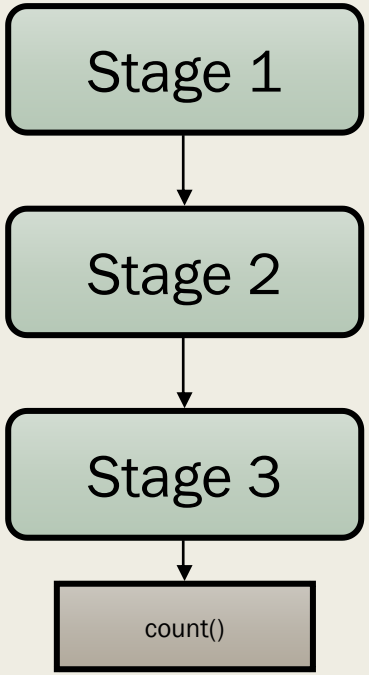


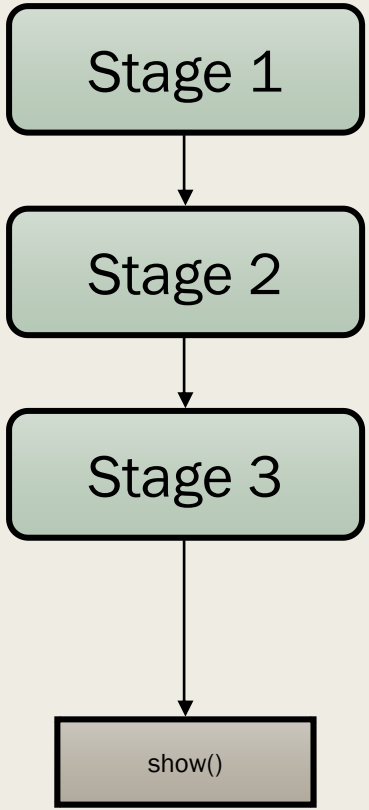
Transformations *- lazy, added to the plan, generally take a dataframe and return a dataframe*

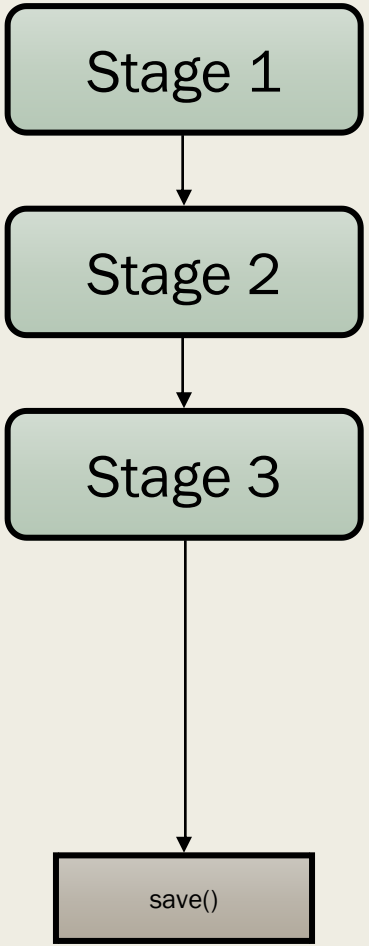
Actions *- eager, force evaluation of the plan*


Stage 1
Stage 2
Stage 3
count()
show()
save()


Stage 1
Stage 2
Stage 3
count()
show()
save()


Stage 1
Stage 2
Stage 3
count()


Stage 1
Stage 2
Stage 3
show()


Stage 1
Stage 2
Stage 3
save()

Memory Management

Cache/persist

- *Lazy or eager*
- *Maintain lineage*
- *Choose storage level*

Checkpoint

- *Destroys the lineage*
- *Maintained after spark application terminated*

Save as …

- *(e.g. saveAsTable)*

Memory Management

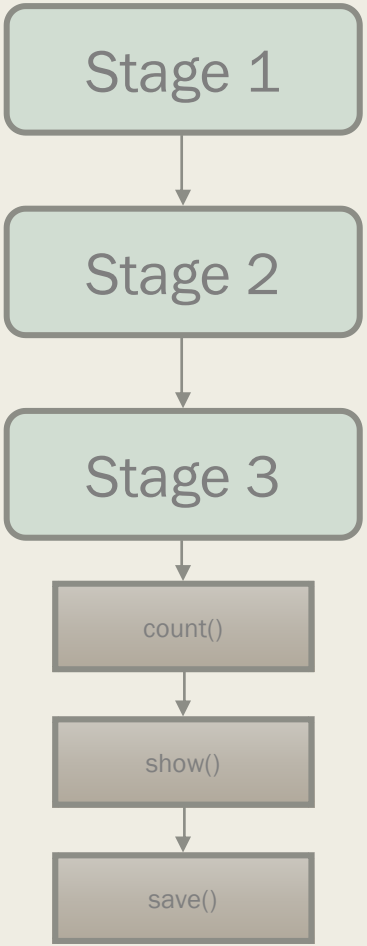
Completed Jobs: 5

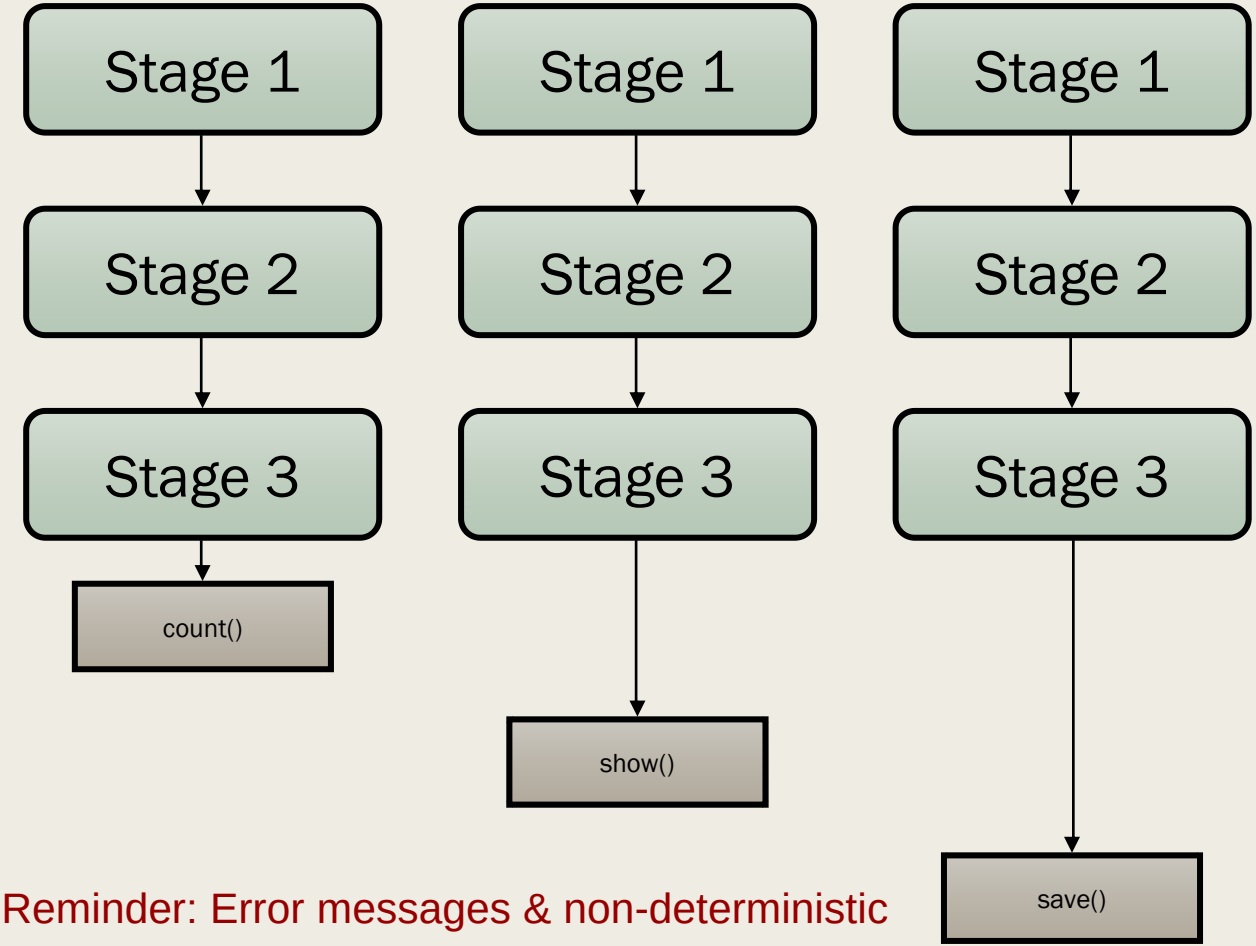
▸ Event Timeline

▾ **Completed Jobs (5)**

Job Id (Job Group)	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
4 (0fe0ca6f-d4a4-4d1f-9637-fc14ab69ad76)	id = 576456da-5fee-4d72-9844-38535128bf0f runId = 0fe0ca6f-d4a4-4d1f-9637-fc14ab69ad76 batch = 3 start at ThrottlerKickStarter.scala:182	2020/04/14 16:22:00	7 s	7/7 (11 skipped)	1021/1021 (1234 skipped)
3 (0fe0ca6f-d4a4-4d1f-9637-fc14ab69ad76)	id = 576456da-5fee-4d72-9844-38535128bf0f runId = 0fe0ca6f-d4a4-4d1f-9637-fc14ab69ad76 batch = 2 start at ThrottlerKickStarter.scala:182	2020/04/14 16:21:20	8 s	7/7 (7 skipped)	1021/1021 (623 skipped)
2 (0fe0ca6f-d4a4-4d1f-9637-fc14ab69ad76)	id = 576456da-5fee-4d72-9844-38535128bf0f runId = 0fe0ca6f-d4a4-4d1f-9637-fc14ab69ad76 batch = 1 start at ThrottlerKickStarter.scala:182	2020/04/14 15:55:17	12 s	7/7 (3 skipped)	832/832 (201 skipped)
1 (0fe0ca6f-d4a4-4d1f-9637-fc14ab69ad76)	id = 576456da-5fee-4d72-9844-38535128bf0f runId = 0fe0ca6f-d4a4-4d1f-9637-fc14ab69ad76 batch = 0 start at ThrottlerKickStarter.scala:182	2020/04/14 15:47:30	2 s	3/3 (1 skipped)	410/410
0 (0fe0ca6f-d4a4-4d1f-9637-fc14ab69ad76)	id = 576456da-5fee-4d72-9844-38535128bf0f runId = 0fe0ca6f-d4a4-4d1f-9637-fc14ab69ad76 batch = 0 start at ThrottlerKickStarter.scala:182	2020/04/14 15:47:27	3 s	3/3	211/211

https://stackoverflow.com/questions/61206084/does-skipped-stages-have-any-performance-impact-on-spark-job





Reminder: Error messages & non-deterministic

So, what should I use?

- How much data do you have?
- What does your current code base look like?
- How much time do you want to spend on infrastructure?
- Who is working on your code?
- Any personal preferences?

In Conclusion

- What is PySpark?
- Can it solve all of my data problems?
- Are you sure I can’t just use pandas instead?

In Conclusion

- What is PySpark?

See previous slides

- Can it solve all of my data problems?

- Are you sure I can’t just use pandas instead?

In Conclusion

- What is PySpark?

See previous slides

- Can it solve all of my data problems?

Kind of, but you'll also get fun new ones

- Are you sure I can't just use pandas instead?

In Conclusion

- What is PySpark?

See previous slides

- Can it solve all of my data problems?

Kind of, but you'll also get fun new ones

- Are you sure I can't just use pandas instead?

https://www.infoq.com/news/2021/11/apache-spark-pandas-api/

Holden Karau

Holden Karau is a transgender Canadian open source developer advocate at Google focusing on Apache Spark, Beam, and related big data tools. Previously, she worked at IBM, Alpine, Databricks, Google (yes, this is her second time), Foursquare, and Amazon. Holden is the coauthor of *Learning Spark*, *High Performance Spark*, and another Spark book that's a bit more out of date. She is a committer & PMC member for Apache Spark and committer on Apache SystemML and Apache Mahout projects. When not in San Francisco, Holden speaks internationally about different big data technologies (mostly Spark). She was tricked into the world of big data while trying to improve search and recommendation systems and has long since forgotten her original goal. Outside of work she enjoys playing with fire, riding scooters, and dancing. You can follow her on Twitter @holdenkarau, Twitch (holdenkarau), and on YouTube.

https://www.oreilly.com/people/holden-karau/