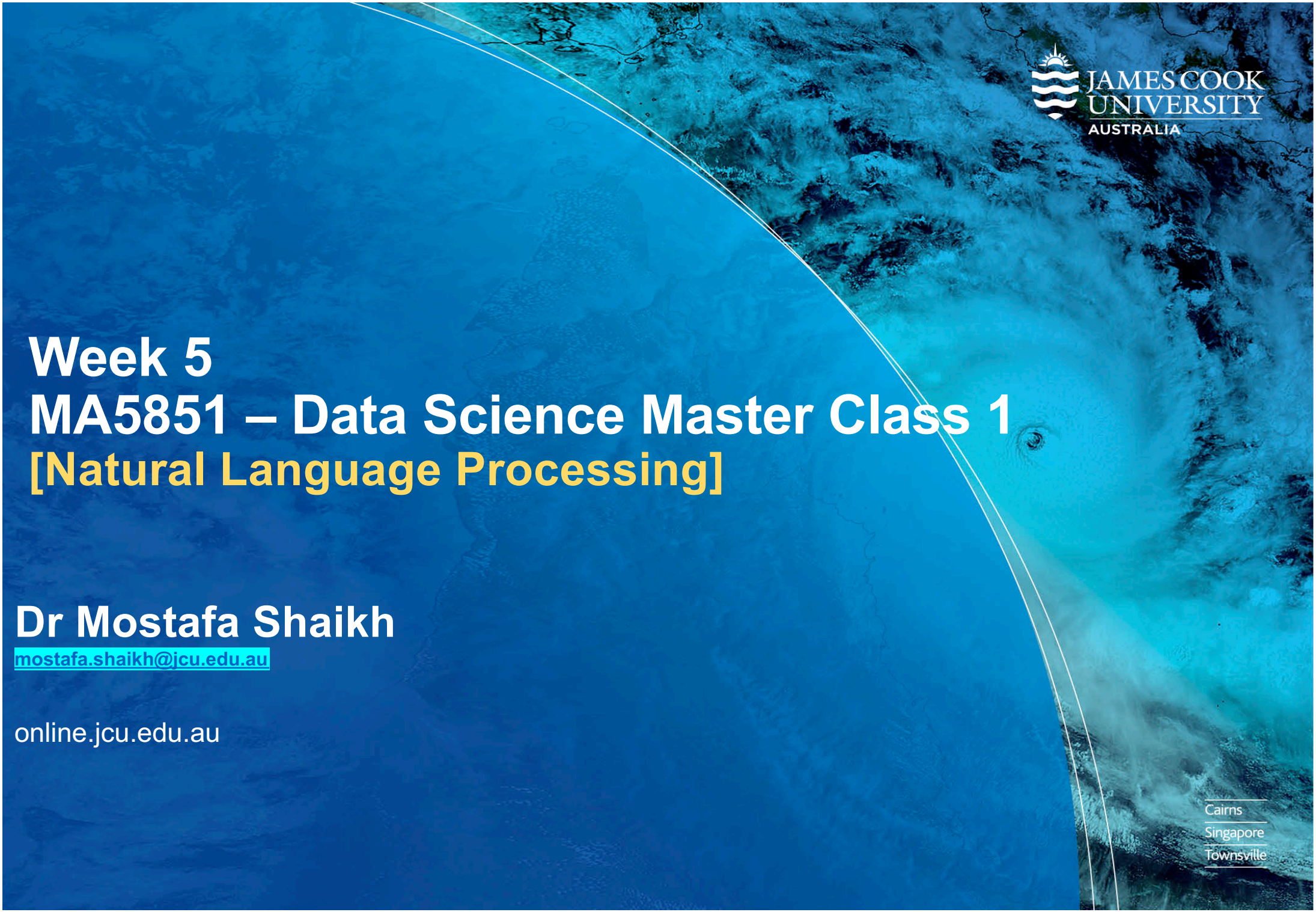

JAMES COOK UNIVERSITY
AUSTRALIA
Week 5
MA5851 – Data Science Master Class 1
[Natural Language Processing]
Dr Mostafa Shaikh
mostafa.shaikh@jcu.edu.au
online.jcu.edu.au
Cairns
Singapore
Townsville

Topic

- Apache Spark
- Spark ecosystem
- Spark vs MapReduce
- PySpark
- RDD
- Spark NLP

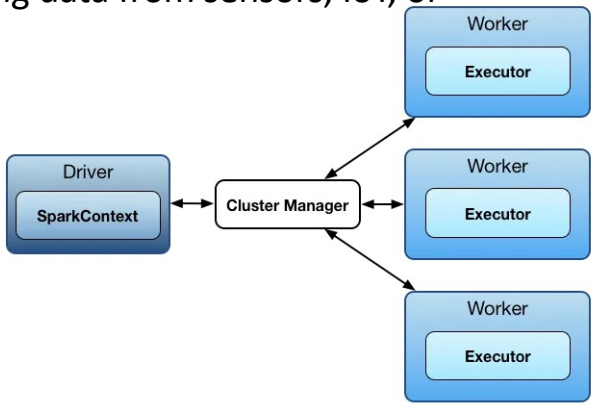
Self Practical

- Setting up Spark and PySpark
- Text processing with NLTK and PySpark
- Sentiment classification with PySpark
- Spark SQL, MLlib: Regression
- Spark NLP
- Spark Stream Processing

Announcement: The final session (week 6 session) on next Thursday from 6:00 pm (NSW time)

Apache Spark

- What is SPARK
  - Apache Spark is an open-source distributed general-purpose cluster-computing framework. Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance.
  - suitable for ETL and SQL batch jobs across large data sets, processing of streaming data from sensors, IoT, or financial systems, and machine learning tasks
- Brief History:
  - 2009: BDAS research project
  - June 2013: Accepted as an Apache Incubator project
  - February 2014: Became a top-level Apache project
  - December 2014: Spark became part of the HDP stack with version 2.2



- Why learning it?
  - **Banking**: **JPMorgan** uses Spark to detect fraudulent transactions, analyse the business spends of an individual to recommend offers, and identify patterns to decide how much to invest and where to invest.
  - **E-Commerce**: **Alibaba** uses Spark to analyse real-time transaction details, browsing history, etc. in the form of Spark jobs and provides recommendations to its users.
  - **Healthcare**: **IQVIA** is a leading healthcare company that uses Spark to analyse patient's data, identify possible health issues, and diagnose it based on their medical history.
  - **Entertainment**: Entertainment and gaming companies like **Netflix** and **Riot games** use Apache Spark to showcase relevant advertisements to their users based on the videos that they watch, share, and like.

https://spark.apache.org/powered-by.html

SPARK features

JAMES COOK UNIVERSITY AUSTRALIA

Fast processing

Spark contains Resilient Distributed Datasets (RDD) which saves time taken in reading, and writing operations and hence, it runs almost ten to hundred times faster than Hadoop

Flexible

Spark supports multiple languages and allows the developers to write applications in Java, Scala, R, or Python

Better analytics

Spark has a rich set of SQL queries, machine learning algorithms, complex analytics, etc. With all these functionalities, analytics can be performed better

In-memory computing

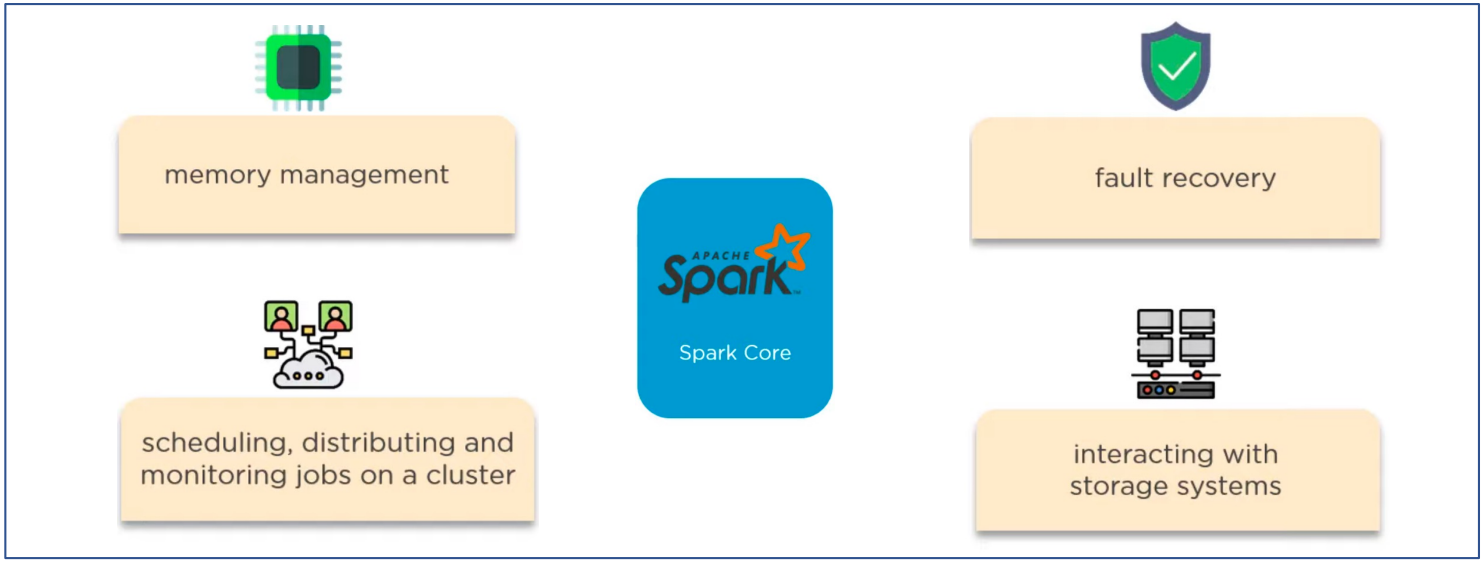
In Spark, data is stored in the RAM, so it can access the data quickly and accelerate the speed of analytics

Fault tolerance

Spark contains Resilient Distributed Datasets (RDD) that are designed to handle the failure of any worker node in the cluster. Thus, it ensures that the loss of data reduces to zero

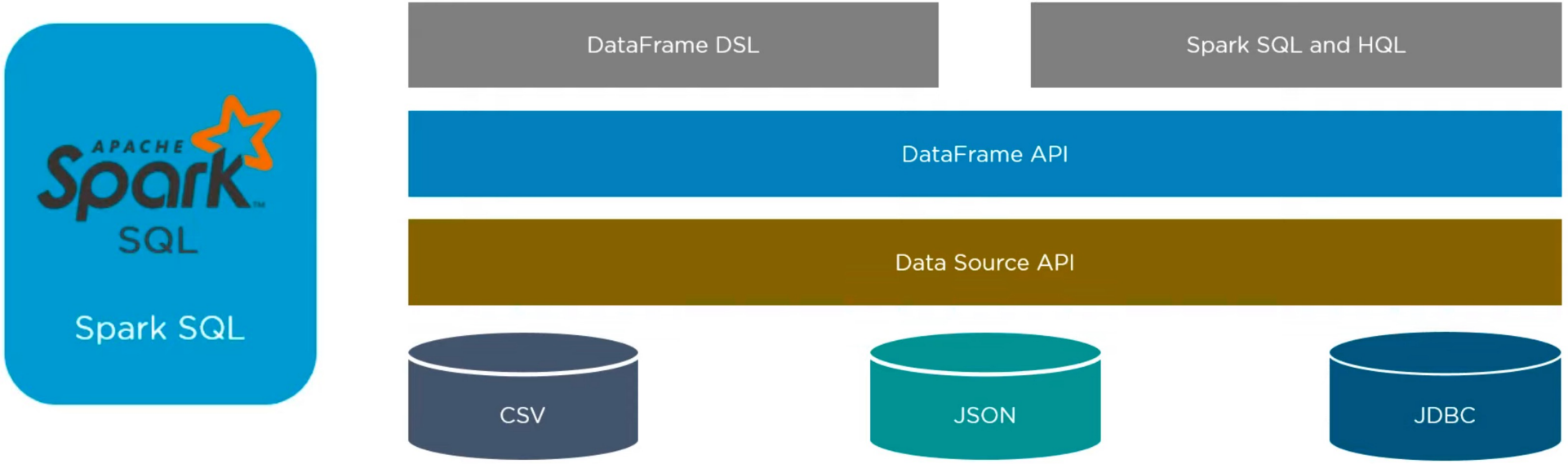
SPARK ecosystem

Core is the base engine for large-scale parallel and distributed data processing embedded with RDDs



SPARK component: Spark SQL

Spark SQL component is used for structured and semi-structured data processing



SPARK component: Spark Streaming

JAMES COOK UNIVERSITY AUSTRALIA

Spark streaming is a lightweight API allowing developers to perform batch processing and real-time streaming of data in a secure, reliable and fast manner

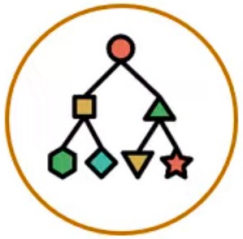
SPARK component: Spark MLlib

MLlib is a low-level machine learning library that is scalable and compatible with various programming languages

MLlib eases the deployment and development of scalable machine learning algorithms

It contains machine learning libraries that have an implementation of various machine learning algorithms

Clustering



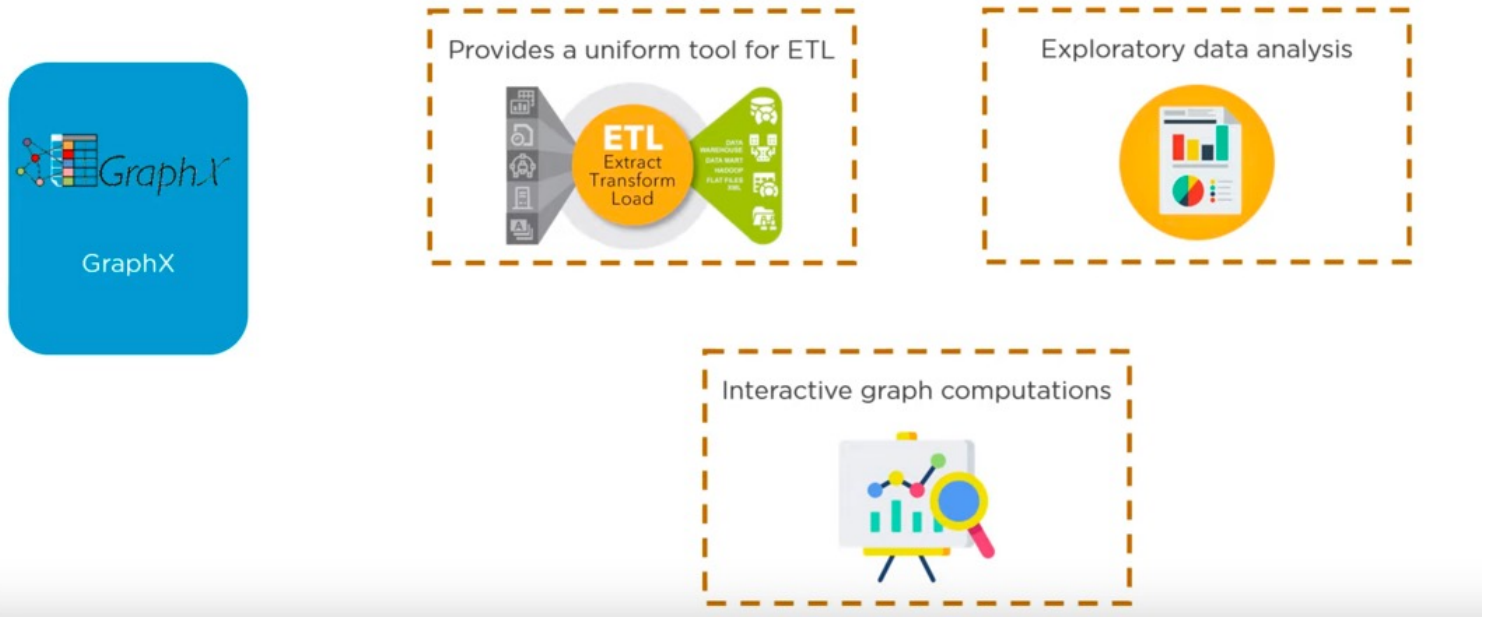
Classification

Collaborative Filtering

the DataFrame-based API is the primary Machine Learning API for Spark. So, from now MLlib will not add any new feature to the RDD based API

SPARK component: GraphX

GraphX is Spark's own graph computation engine and data store

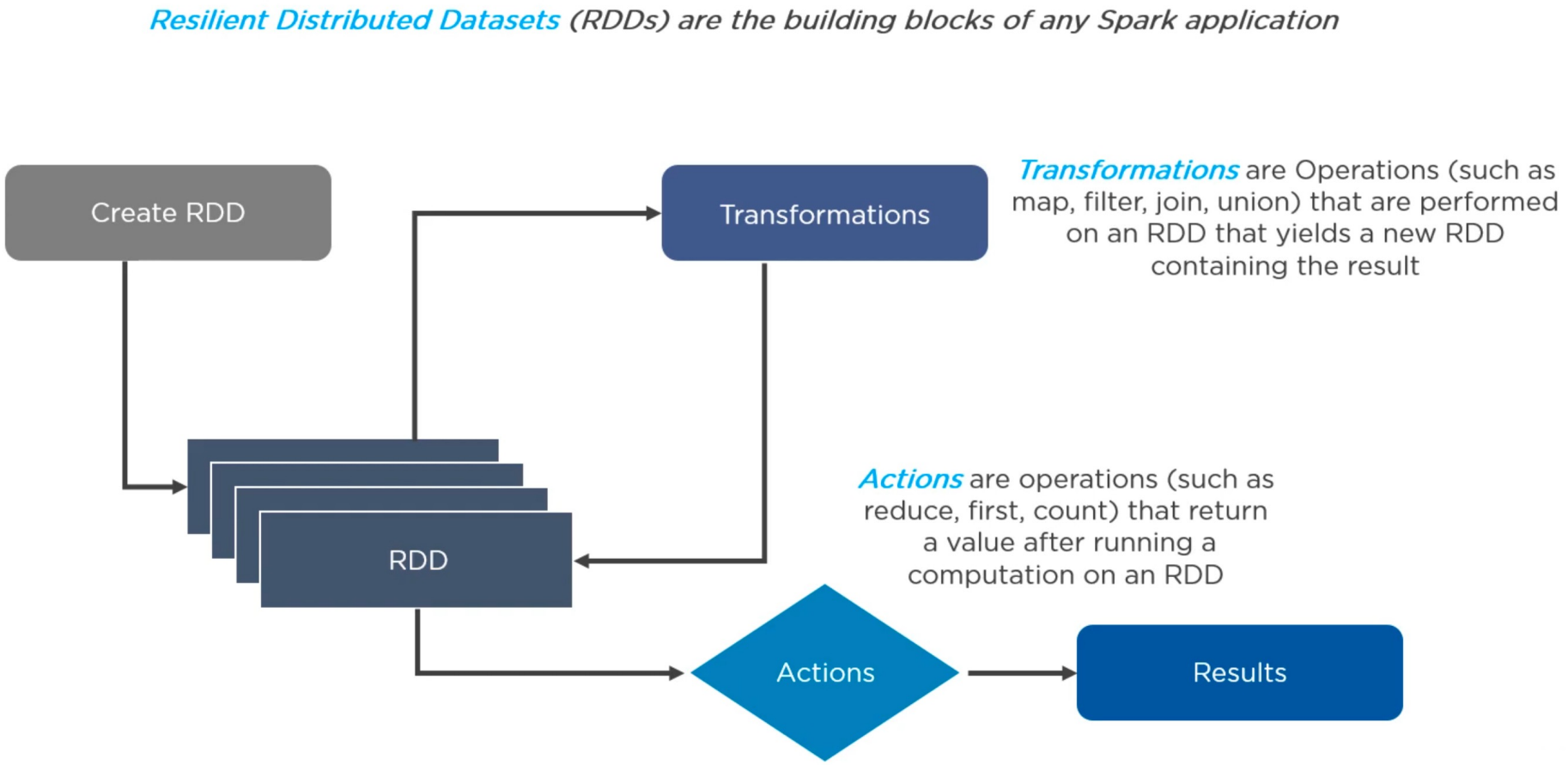


Spark and MapReduce

JAMES COOK UNIVERSITY AUSTRALIA

	
Processing data using MapReduce in Hadoop is slow	Spark processes data 100 times faster than MapReduce as it is done in-memory
Performs batch processing of data	Performs both batch processing and real-time processing of data
Hadoop has more lines of code. Since it is written in Java, it takes more time to execute	Spark has fewer lines of code as it is implemented in Scala
Hadoop supports Kerberos authentication, which is difficult to manage	Spark supports authentication via a shared secret. It can also run on YARN leveraging the capability of Kerberos

RDD

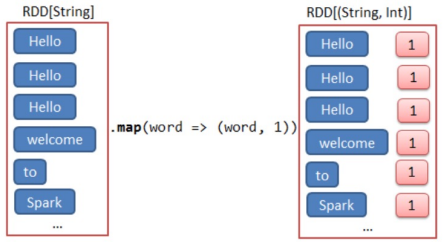
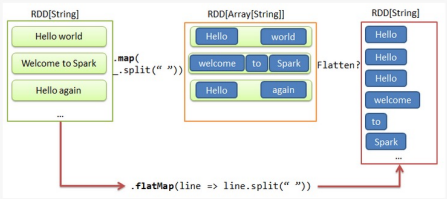


- Transformation: It is a function that produces new RDD from the existing RDDs.
- Action: In Transformation, RDDs are created from each other. But when we want to work with the actual dataset, then, at that point we use Action.

RDD -transformation

Two types of transformations:

- **Narrow transformation** – In *Narrow transformation*, all the elements that are required to compute the records in single partition live in the single partition of parent RDD.
- **Wide transformation** – In wide transformation, all the elements that are required to compute the records in the single partition may live in many partitions of parent RDD.

map (f: T => U)	Return a MappedRDD[U] by applying function f to each element of RDD.  *Figure 5.3. Transformation map example, (f: T => U) (JCU, 2019)*	Narrow
flatmap (f: T => TraversableOnce[U])	Return a new FlatMappedRDD[U] by first applying a function to all elements and then flattening the results.  *Figure 5.4. Transformation flatmap example, flatmap (f: T => TraversableOnce[U]) (JCU, 2019)*	Narrow

distinct()	Return a new RDD containing distinct elements of the source dataset. It is helpful to remove duplicate data.	Wide
groupByKey()	Being called on (K,V) RDD, return a new RDD[([K], Iterable[V])]. The data is shuffled according to the key-value K in another RDD. In this transformation, lots of unnecessary data get to transfer over the network. Spark provides the provision to save data to disk when there is more data shuffled onto a single executor machine than can fit in memory.	Wide
reduceByKey(f: (V, V) => V)	Being called on (K, V) Rdd, return a new RDD[(K, V)] by aggregating values using key: reduceByKey(_+_)	Wide
sortByKey([ascending])	Being called on (K, V) Rdd where K implements Ordered, return a new RDD[(K, V)] sorted by K	Wide
join(other: RDD[(K, W))	Being called on (K,V) Rdd, return a new RDD[(K, (V, W))] by joining them	Wide

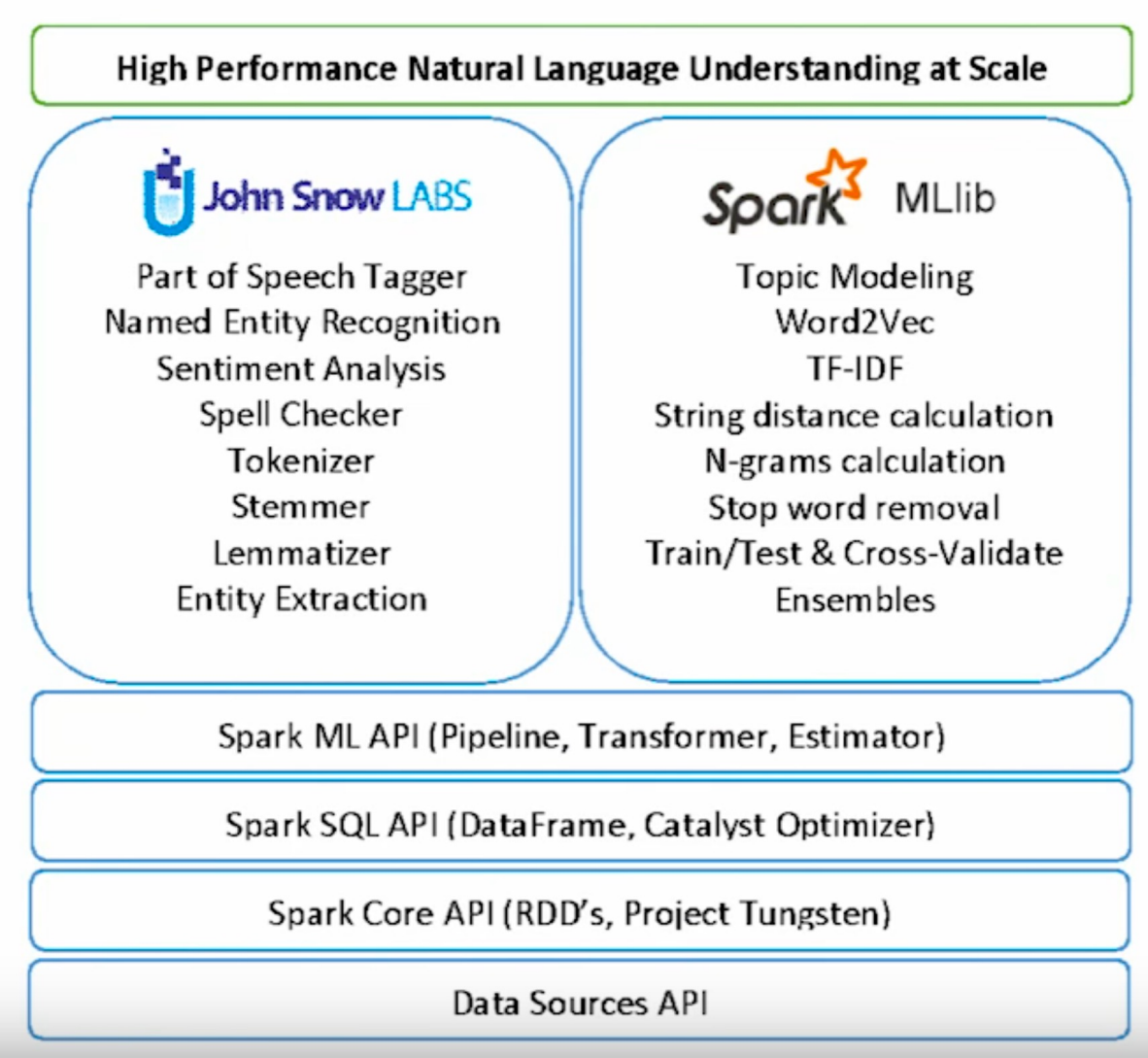
RDD actions

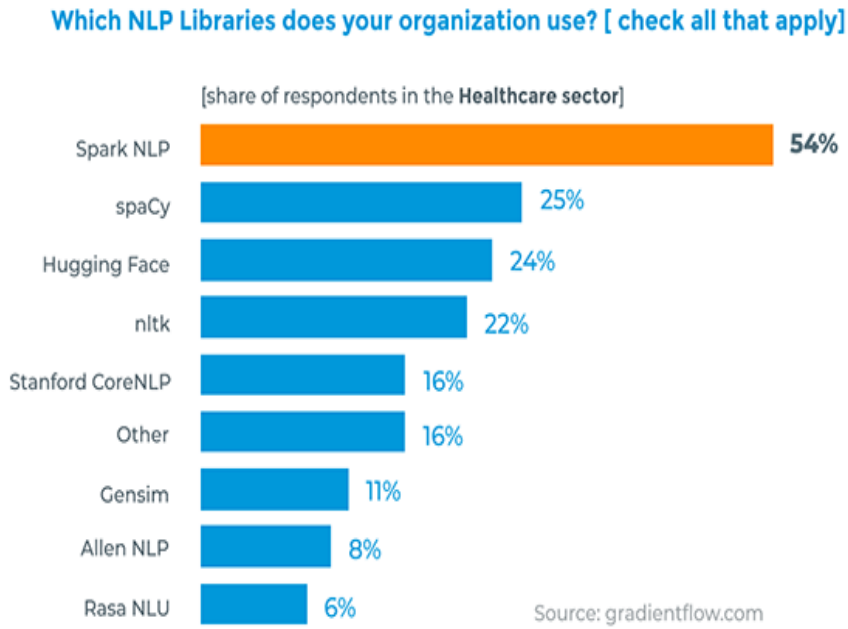
JAMES COOK UNIVERSITY AUSTRALIA

Actions are Spark RDD operations that give non-RDD values. It brings laziness of RDD into motion. An action is one of the ways of sending data from *Executer* to the *driver.*

reduce(f: (T, T) => T)	Return T by reducing the elements using specified commutative and associative binary operator
collect()	Return an Array[T] containing all elements
count()	Return the number of elements
first()	Return the first element
take(num)	Return an Array[T] taking first num elements
takeSample(withReplacement, fraction, seed)	Return an Array[T] which is a sampled subset
takeOrdered(num)(order)	Return an Array[T] having num smallest or biggest (depend on order) elements
saveAsTextFile(fileName) **saveAsSequenceFile(fileName)** **saveAsObjectFile(fileName)**	Save (serialized) Rdd
countByValue()	Return a Map[T, Long] having the count of each unique value
countByKey()	Return a Map[K, Long] counting the number of elements for each key
foreach(f: T=>Unit)	Apply function f to each element
aggregate()	It gives the flexibility to get data type different from the input type. The **aggregate()** takes two functions to get the final result. Through one function we combine the element from our RDD with the accumulator, and the second, to combine the accumulator. Hence, in aggregate, we supply the initial zero value of the type which we want to return.

Spark NLP





https://github.com/JohnSnowLabs/spark-nlp-models

Spark NLP

JAMES COOK UNIVERSITY AUSTRALIA

Name	Spark NLP	spaCy	NLTK	CoreNLP
Sentence detection	Yes	Yes	Yes	Yes
Tokenization	Yes	Yes	Yes	Yes
Stemming	Yes	Yes	Yes	Yes
Lemmatization	Yes	Yes	Yes	Yes
POS tagger	Yes	Yes	Yes	Yes
NER	Yes	Yes	Yes	Yes
Dependency parse	Yes	Yes	Yes	Yes
Text matcher	Yes	**Yes**	**No**	Yes
Date matcher	Yes	**No**	**No**	Yes
Chunking	Yes	Yes	Yes	Yes
Spell checker	Yes	**No**	**No**	**No**
Sentiment detector	Yes	**No**	**No**	Yes
Pretrained models	Yes	Yes	Yes	Yes
Training models	Yes	Yes	Yes	Yes

Features	Spark NLP	spaCy	NLTK	CoreNLP
Provides full Java API	Yes	**No**	**No**	Yes
Provides full Scala API	Yes	**No**	**No**	**No**
Provides full Python API	Yes	Yes	Yes	**No**
Supports training on GPU	Yes	Yes	**No**	**No**
Supports user-defined deep learning networks	Yes	**Yes**	**No**	**No**
Supports Spark natively	Yes	**No**	**No**	**No**
Supports Hadoop (YARN and HDFS)	Yes	**No**	**No**	**No**

Name	Language	License	Commercial use	Commercial support
Spark NLP	Java, Scala, Python	Apache 2.0	Yes	Yes
spaCy	Python	MIT	Yes	Yes
NLTK	Python	Apache 2.0	Yes	**No**
CoreNLP	Java	GNU GPL	**Paid license**	**No**
OpenNLP	Java	Apache 2.0	Yes	**No**

PySpark


```
# Reading CSV to Spark Dataframe
file_location = "/path/to/my_data.csv"
df = spark.read.format("csv").option("inferSchema", True).option("header",True).load(file_location)
display(df)


# MLlib imports
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.regression import LinearRegression

# Create a vector representation for features
assembler = VectorAssembler(inputCols=['F1', ‘F2', ‘F3', ‘F4', ‘F5', ‘F6’,  … 'Fn'], outputCol="features")
train_df = assembler.transform(df)

# Fit a linear regression model
lr = LinearRegression(featuresCol = 'features', labelCol=‘TargetVariable’)
lr_model = lr.fit(train_df)

# Output statistics
trainingSummary = lr_model.summary
print("Coefficients: " + str(lr_model.coefficients))
print("RMSE: %f" % trainingSummary.rootMeanSquaredError)
print("R2: %f" % trainingSummary.r2)
```

Spark Summary