Spark API Zoo

An Overview

Hi, I'm Casey Stella!

Low Level API - RDD

- Primary API used in Spark 1.x

Low Level API - RDD

- Primary API used in Spark 1.x
- Does not presume structured data

Low Level API - RDD

- Primary API used in Spark 1.x
- Does not presume structured data
- Lowest level control

Low Level API - RDD

- Primary API used in Spark 1.x
- Does not presume structured data
- Lowest level control
  - Basic functional-style distributed computing primitives

Low Level API - RDD

- Primary API used in Spark 1.x
- Does not presume structured data
- Lowest level control
  - Basic functional-style distributed computing primitives
  - Ability to control data partition/placement

Low Level API - RDD

- Primary API used in Spark 1.x
- Does not presume structured data
- Lowest level control
  - Basic functional-style distributed computing primitives
  - Ability to control data partition/placement
  - Ability to broadcast variables

Low Level API - RDD

- Primary API used in Spark 1.x
- Does not presume structured data
- Lowest level control
  - Basic functional-style distributed computing primitives
  - Ability to control data partition/placement
  - Ability to broadcast variables

RDDs are like C: maximum control with maximum opportunity to shoot yourself in the foot.

High Level API - DataFrame/Datasets

- Primary API used in Spark 2.x

High Level API - DataFrame/Datasets

- Primary API used in Spark 2.x
- Table-like collection with well-defined rows and (nullable) columns and schemas.

High Level API - DataFrame/Datasets

- Primary API used in Spark 2.x
- Table-like collection with well-defined rows and (nullable) columns and schemas.
- These represent high level instructions which get converted into RDD transformations by a cost based optimizer.

High Level API - DataFrame/Datasets

- Primary API used in Spark 2.x
- Table-like collection with well-defined rows and (nullable) columns and schemas.
- These represent high level instructions which get converted into RDD transformations by a cost based optimizer.
- DataFrame operations/capabilities are mostly unified between streaming and batch

High Level API - DataFrame/Datasets

- Primary API used in Spark 2.x
- Table-like collection with well-defined rows and (nullable) columns and schemas.
- These represent high level instructions which get converted into RDD transformations by a cost based optimizer.
- DataFrame operations/capabilities are mostly unified between streaming and batch
- Higher level control with more focus on relational primitives

High Level API - DataFrame/Datasets

- Primary API used in Spark 2.x
- Table-like collection with well-defined rows and (nullable) columns and schemas.
- These represent high level instructions which get converted into RDD transformations by a cost based optimizer.
- DataFrame operations/capabilities are mostly unified between streaming and batch
- Higher level control with more focus on relational primitives

The attempt behind DataFrames is provides most of the flexibility of RDDs but hides some of the complexity.

High Level API - Datasets

Datasets are type-safe DataFrames.

High Level API - Datasets

Datasets are type-safe DataFrames.

- There is a performance penalty for encoding/decoding to/from Row the custom data type.

High Level API - Datasets

Datasets are type-safe DataFrames.

- There is a performance penalty for encoding/decoding to/from Row the custom data type.
- This is useful when

High Level API - Datasets

Datasets are type-safe DataFrames.

- There is a performance penalty for encoding/decoding to/from Row the custom data type.
- This is useful when
  - You have a set of objects you want to share between spark/non-spark code.

High Level API - Datasets

Datasets are type-safe DataFrames.

- There is a performance penalty for encoding/decoding to/from Row the custom data type.
- This is useful when
  - You have a set of objects you want to share between spark/non-spark code.
  - You really, really want type-safety

High Level API - Spark SQL

Spark SQL is an alternative high level API which

- Is ANSI SQL:2003 compliant

High Level API - Spark SQL

Spark SQL is an alternative high level API which

- Is ANSI SQL:2003 compliant
- Integrates with the Hive metastore

High Level API - Spark SQL

Spark SQL is an alternative high level API which

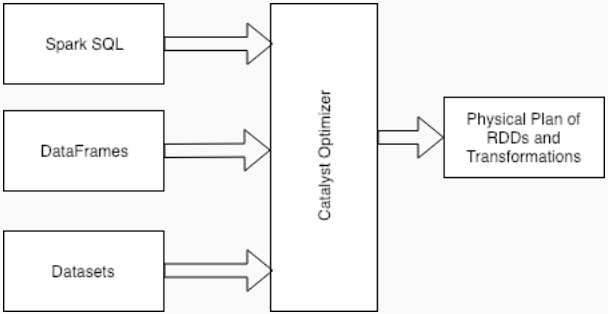
- Is ANSI SQL:2003 compliant
- Integrates with the Hive metastore
- Allows you to interact with the results of a query as a DataFrame

High Level API - Spark SQL

Spark SQL is an alternative high level API which

- Is ANSI SQL:2003 compliant
- Integrates with the Hive metastore
- Allows you to interact with the results of a query as a DataFrame

API Relationship



High Level API - Catalyst

Catalyst is a cost based optimizer which will map DataFrame operations to a physical plan. This provides some benefits:

High Level API - Catalyst

Catalyst is a cost based optimizer which will map DataFrame operations to a physical plan. This provides some benefits:

- A unified layer to apply optimizations which will affect DataFrame, SQL and Datasets

High Level API - Catalyst

Catalyst is a cost based optimizer which will map DataFrame operations to a physical plan. This provides some benefits:

- A unified layer to apply optimizations which will affect DataFrame, SQL and Datasets
- By providing an internal representation, many common transformations will operate outside of the host language.

High Level API - Catalyst

Catalyst is a cost based optimizer which will map DataFrame operations to a physical plan. This provides some benefits:

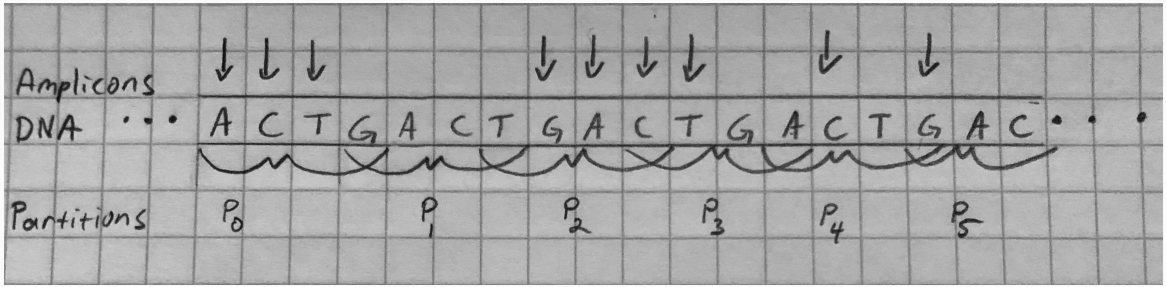
- A unified layer to apply optimizations which will affect DataFrame, SQL and Datasets
- By providing an internal representation, many common transformations will operate outside of the host language.

Key takeaway: Catalyst is a level of indirection that the community to centralize optimization work that will work across the logical APIs

Example: Off-Target Amplicon Detection

Determining off-target amplicon matches are important parts of designing custom gene sequencing.

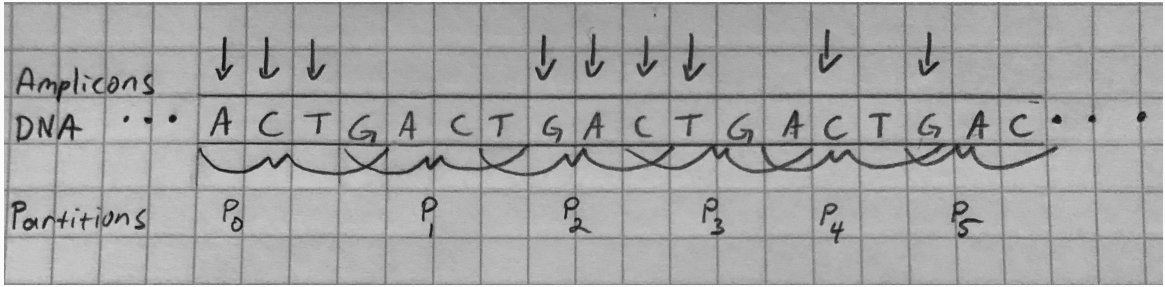
- Compute amplicon matches
- Partition with overlap via a simple k length strategy
- Determine which amplicon pairs are "valid"



Example: Off-Target Amplicon Detection

Problems

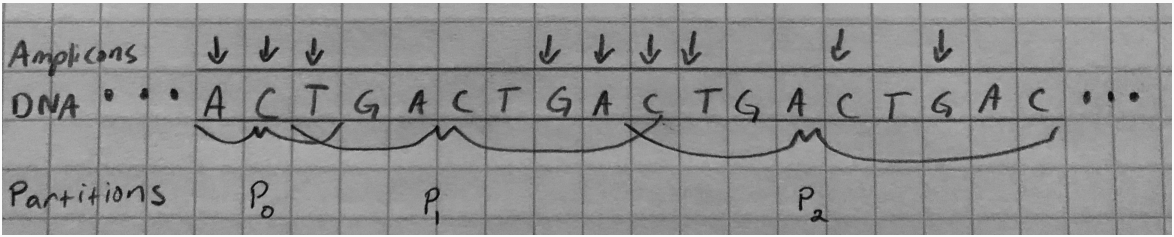
- We're in a situation where we need to do a computationally complex operation with large constant per partition.
- Lots of mostly empty partitions!



Example: Off-Target Amplicon Detection

Percentile-based Partitioning

- Compute amplicon matches
- Determine partition splits $\ni$ each partition contains an equal percent of matches
- Determine which amplicon pairs are "valid" with parallelism unit as the partition



Some Advice

- Try to use dataframes and catalyst expressions as much as you can (expecially if working in a non-JVM language).

Some Advice

- Try to use dataframes and catalyst expressions as much as you can (expecially if working in a non-JVM language).
- Don't be afraid to move up and down the stack of complexity as needed.

Some Advice

- Try to use dataframes and catalyst expressions as much as you can (expecially if working in a non-JVM language).
- Don't be afraid to move up and down the stack of complexity as needed.
- Using non-JVM languages requires care and understanding of data serialization costs.

Some Advice

- Try to use dataframes and catalyst expressions as much as you can (expecially if working in a non-JVM language).
- Don't be afraid to move up and down the stack of complexity as needed.
- Using non-JVM languages requires care and understanding of data serialization costs.
  - It may be a winning strategy to implement common custom UDFs in a JVM language and use them.

Some Advice

- Try to use dataframes and catalyst expressions as much as you can (expecially if working in a non-JVM language).
- Don't be afraid to move up and down the stack of complexity as needed.
- Using non-JVM languages requires care and understanding of data serialization costs.
  - It may be a winning strategy to implement common custom UDFs in a JVM language and use them.
- Mind memory and monitor garbage collection.

Questions

Thanks for your attention! Questions?

- This talk available on my github presentation page.[1]
- Find me at http://caseystella.com
- Twitter handle: @casey_stella

[1] http://github.com/cestella/presentations/