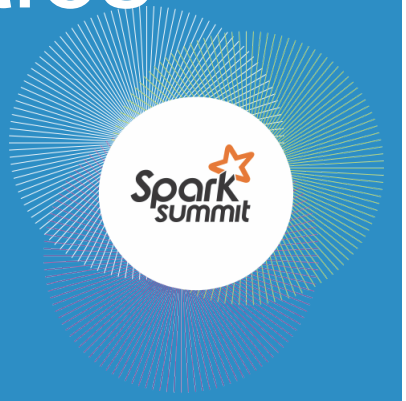
Spark @ Bloomberg: Dynamic Composable Analytics

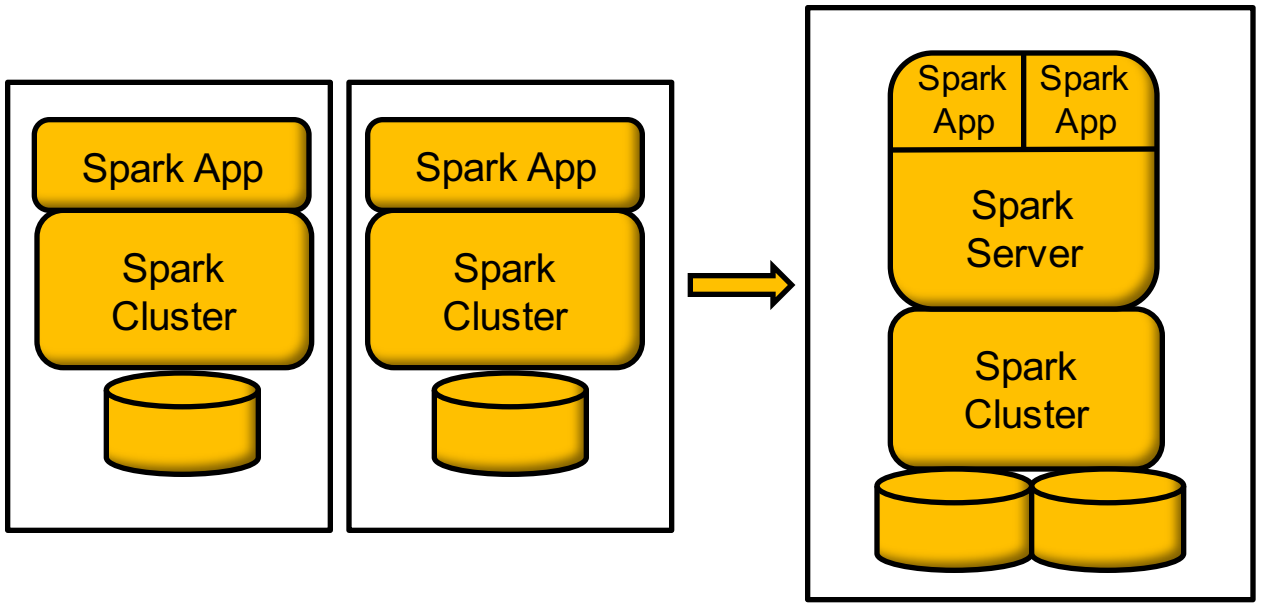
Partha Nageswaran
Sudarshan Kadambi
BLOOMBERG L.P.



SPARK SUMMIT EAST
DATA SCIENCE AND ENGINEERING AT SCALE
FEBRUARY 16-18, 2016 NEW YORK CITY

Spark at Bloomberg:
Dynamic Composable Analytics

- Adaptation of Spark in Bloomberg is evolving from crafting stand-alone Spark Apps to Serverized Spark Apps



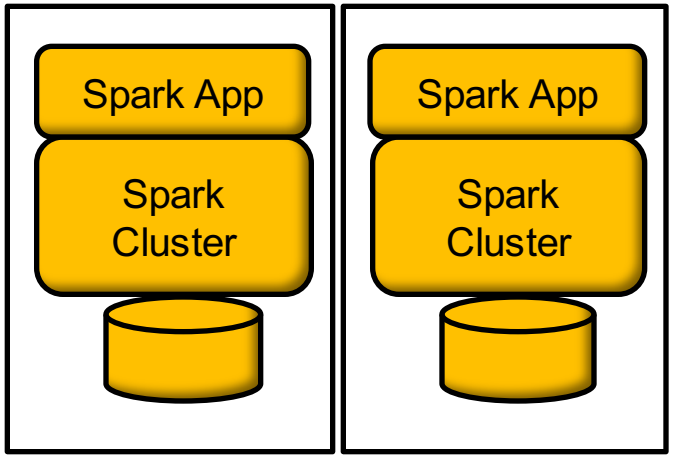
A Couple of Tenets

- Preference for on the fly calculations over pre-computed values

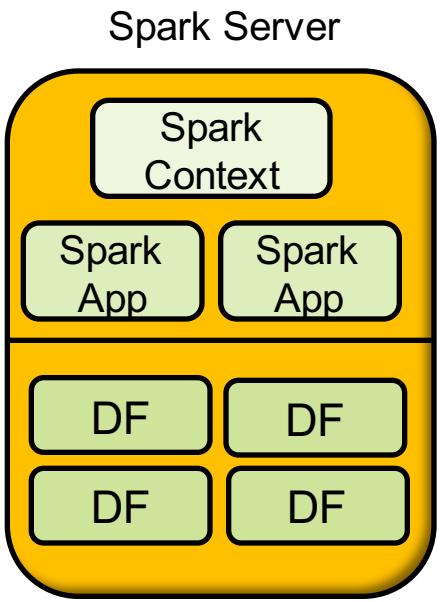
- Support analytics on analytics, ad infinitum in a dynamic manner

Spark Serverization - Motivation

- Stand-alone Spark Apps on isolated clusters pose challenges:
  - Management of clusters, replication of data, etc.
  - Analytics are confined to specific content sets making Cross-Asset Analytics much harder
  - Need to handle Real-time ingestion in each App!
  - Redundancy in:
    - » Crafting and Managing of RDDs/DFs
    - » Coding of the same or similar types of transforms/actions

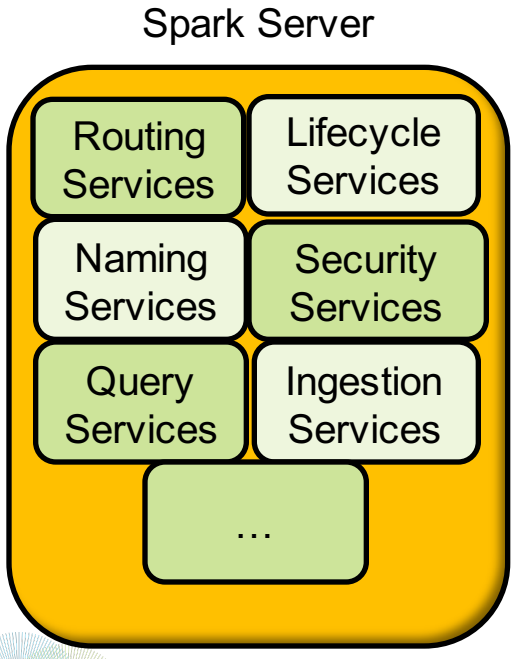


Spark Serverization - Approach



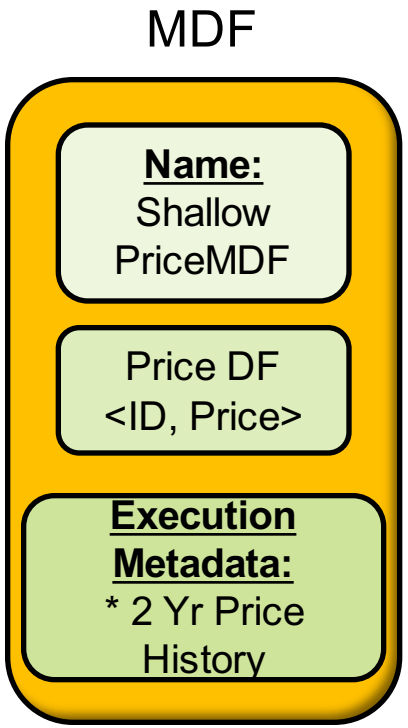
- Long-running process (Spark Server) maintains a shared SparkContext
  - Spark Apps within process share the same SparkContext
- Provide a Container based approach to capabilities such as:
  - Deploying and Managing Spark Applications
  - Deploying, Managing and Sharing RDDs / DFs across Spark Apps
  - Handling on-the-fly analytics on Streaming data
  - Declarative orchestration of higher-order analytics on RDDs / DFs and across Spark Applications

Benefits of Container Approach

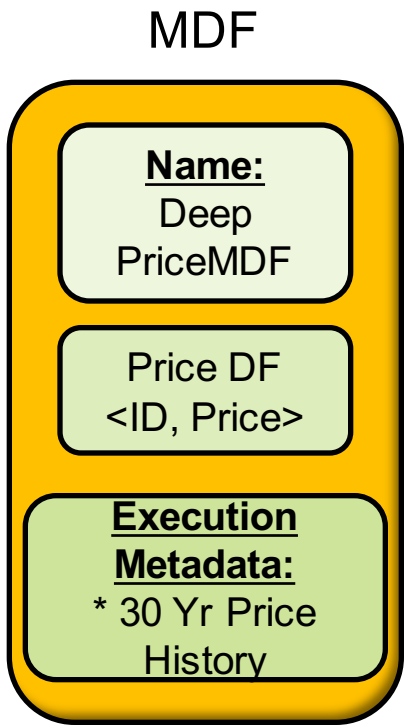


- Provide Lifecycle Management for Spark Apps
- Provide Lifecycle Management for RDDs/DFs
- Provide other declarative qualities of services, such as:
  - Routing of Requests to appropriate Spark Apps
  - Naming Service capabilities for RDDs/DFs
  - Ingestion Services
  - Securing access to Spark Apps, and RDDs/DFs

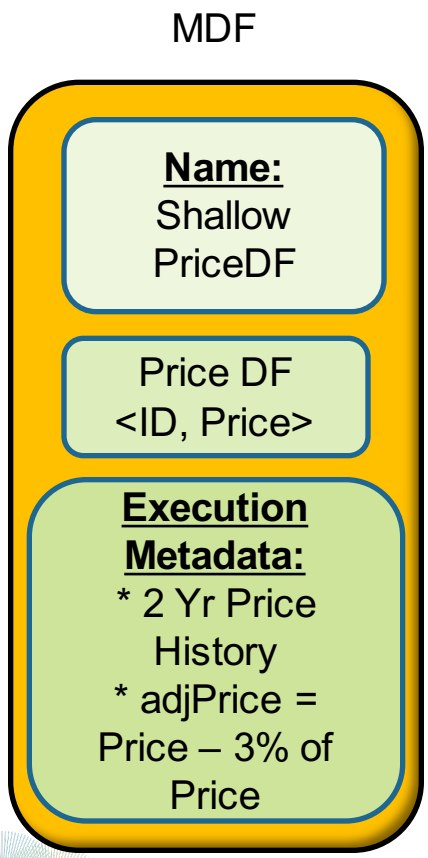
Introducing Managed DataFrames (MDFs)



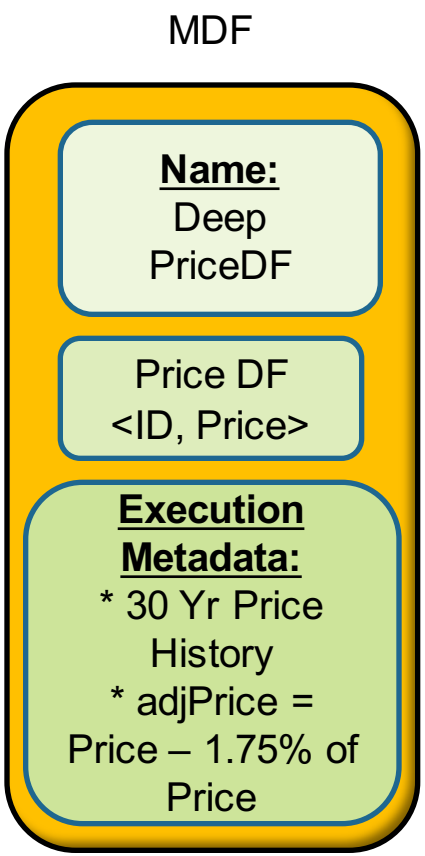
- A Managed DataFrame (MDF) is a named DataFrame, optionally combined with Execution Metadata
  - MDFs can be searched by name OR by any Column Name defined in the Schema of the corresponding DF
- Execution Metadata includes:
  - **Data Distribution** metadata captures information about the data depth, histogram information, etc.
  - E.g.: A managed DataFrame for pricing of stocks, representing 2 years of historical data and another for representing 30 years of historical data



Introducing Managed DataFrames (MDFs)

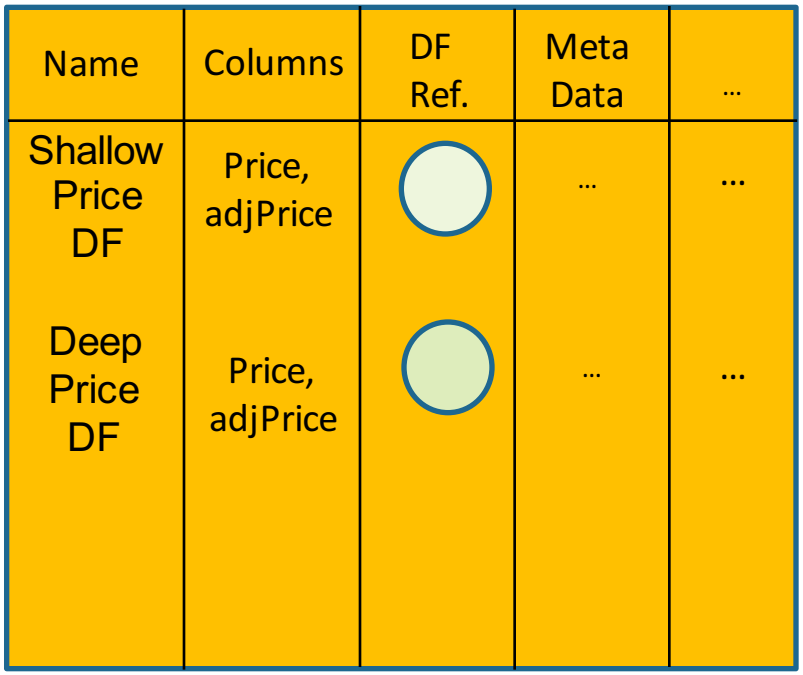


- **Data Derivation** metadata which are mathematical expressions that define how additional columns can be synthesized from existing columns in the schema
- E.g.: adjPrice is a derived Column, defined in terms of the base Price column
- In essence, an MDF with data derivation metadata have a Schema that is a union of the contained DF and the derived columns



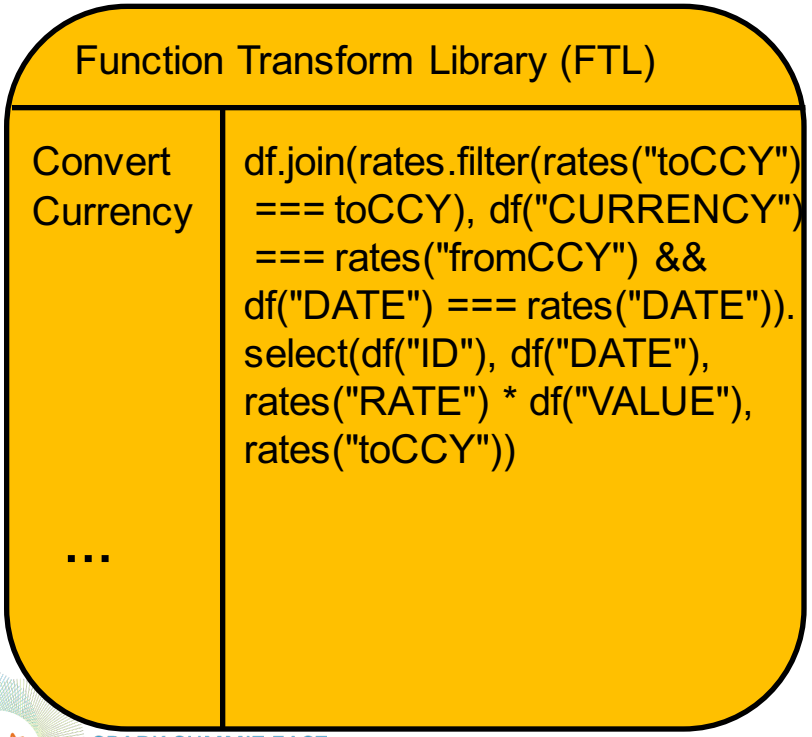
Introducing MDF Registry

MDF Registry



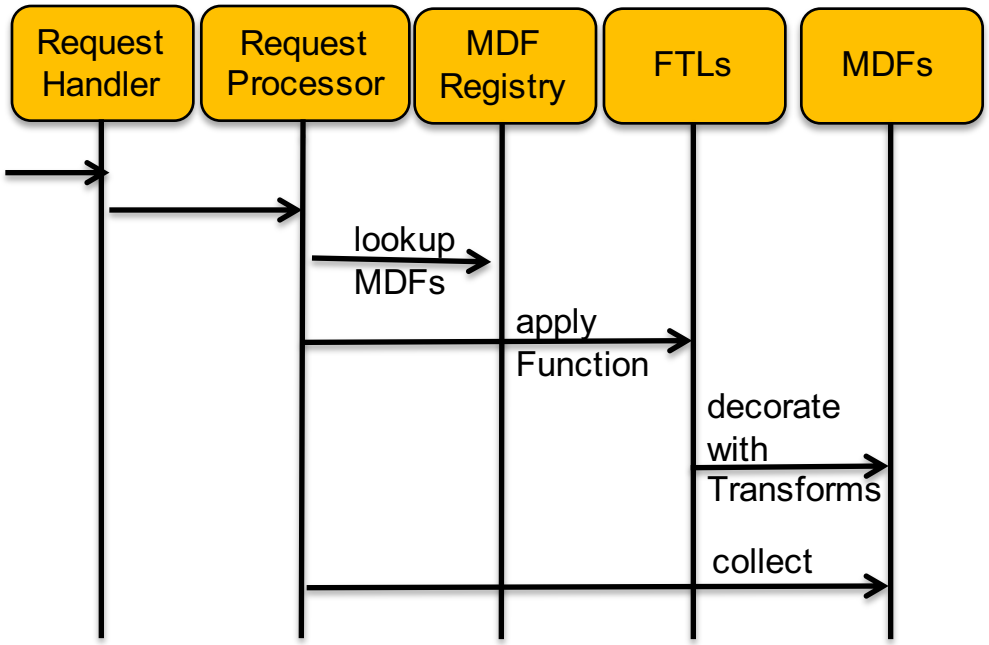
- A Registry, called the MDF Registry within the Spark Server provides support for:
  - Binding MDFs by Name
  - Looking up MDFs by Name
  - Looking up MDF by a Column Name (an element of the MDF Schema), etc.
- The MDF Registry maintains a 'table' that associates the Name of the MDF with the DF reference and Columns in the DF

Introducing DF Function Transform Libraries (FTLs)



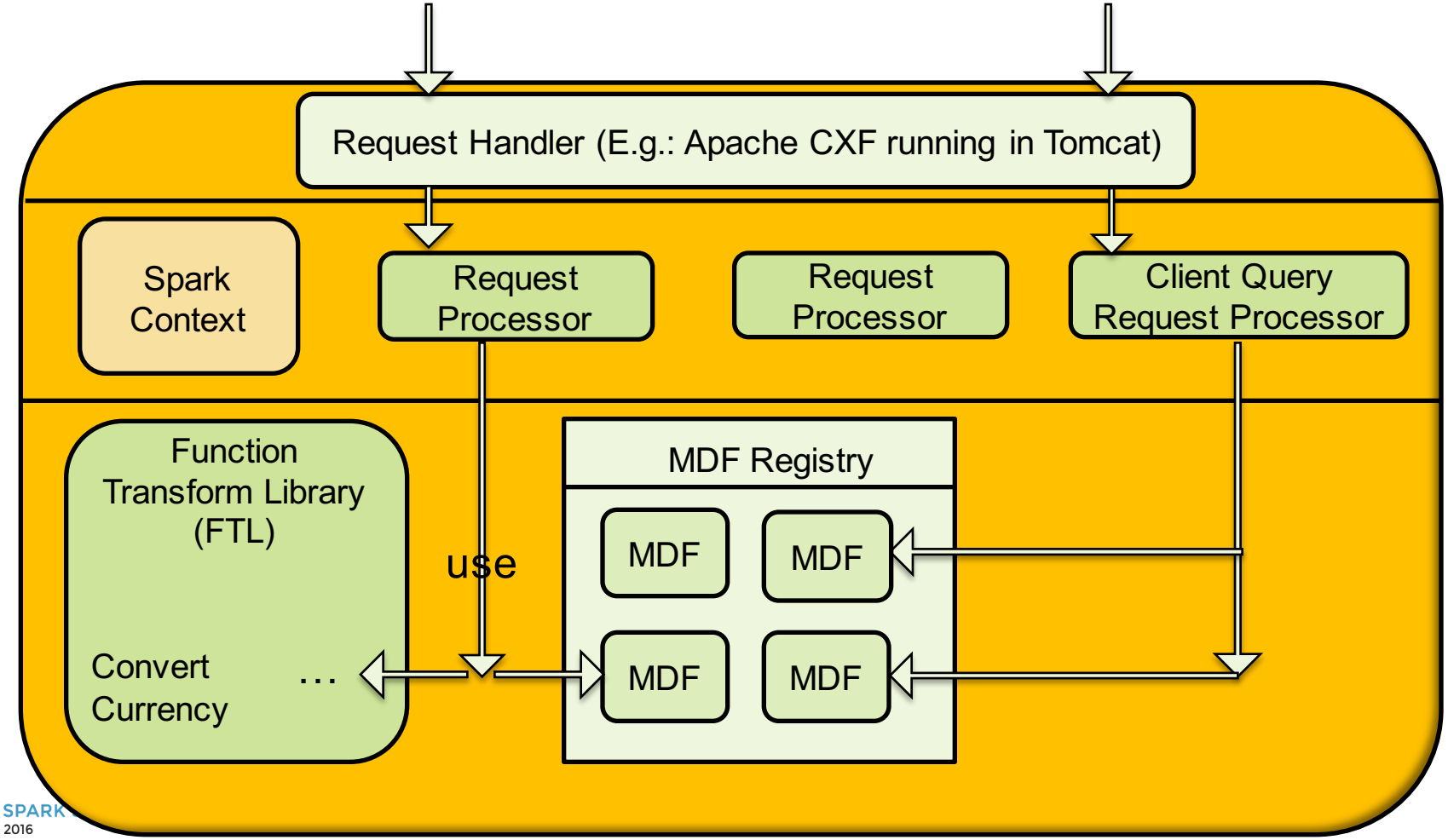
- Standard (Analytics) functions can be expressed as 'pre-canned' Spark Transforms/Actions in Function Transform Libraries (FTLs)
- Spark Apps can compose pre-canned transforms with other application logic transforms on MDFs, by looking up the pre-canned transforms from FTLs
  - E.g.: convertedPriceDF = FTL.apply(prices, "ConvertCurrency", params);

Introducing Request Processor

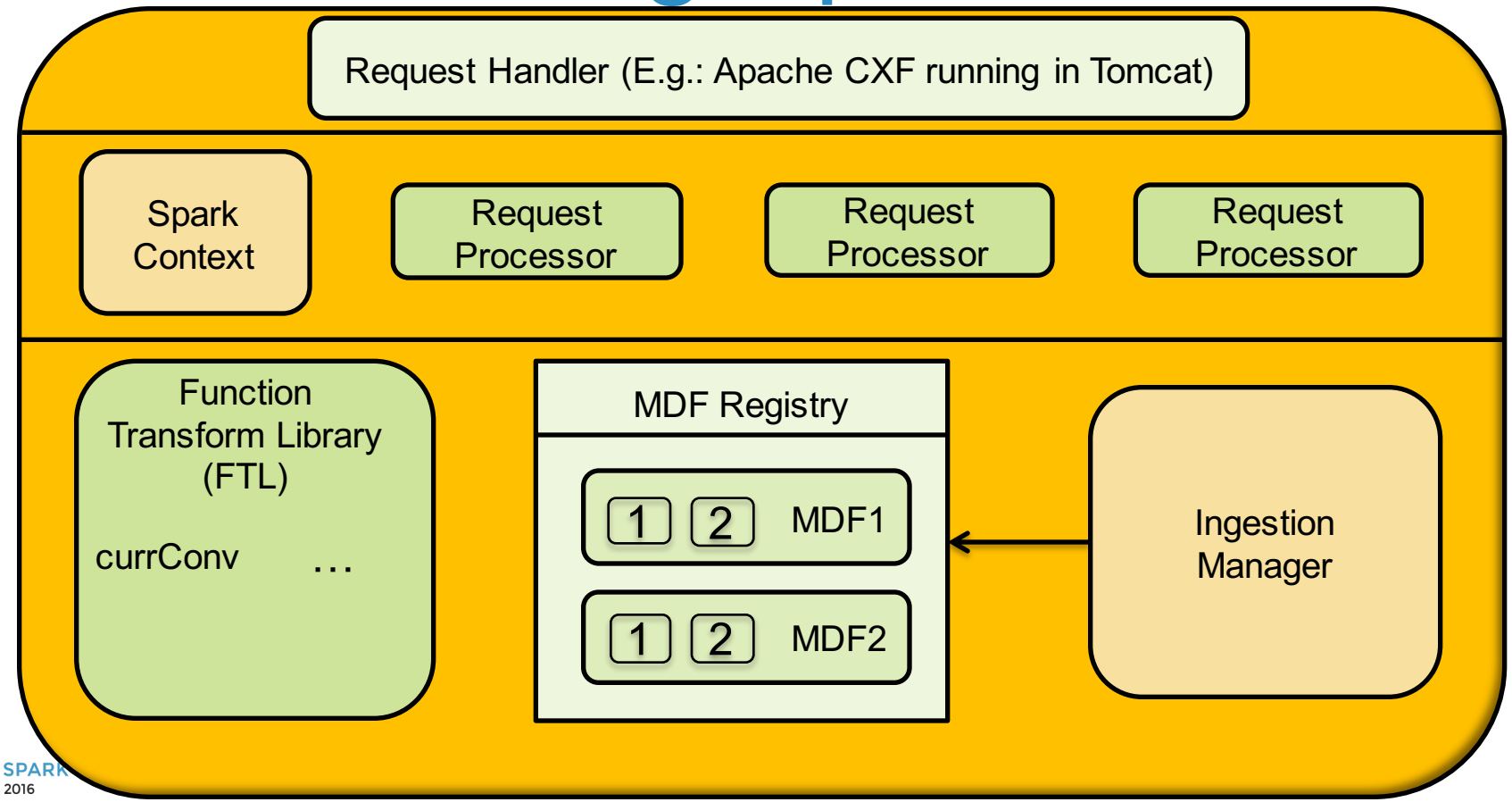


- Spark Apps (called RequestProcessors - RP) within the Spark Server are implemented compliant to specifications
  - These RPs are provided access to the Registry and FTLs
  - Are responsible for composing transforms and actions on one or more MDFs
  - May dynamically bind additional MDFs (materialized or otherwise) for use by other Apps

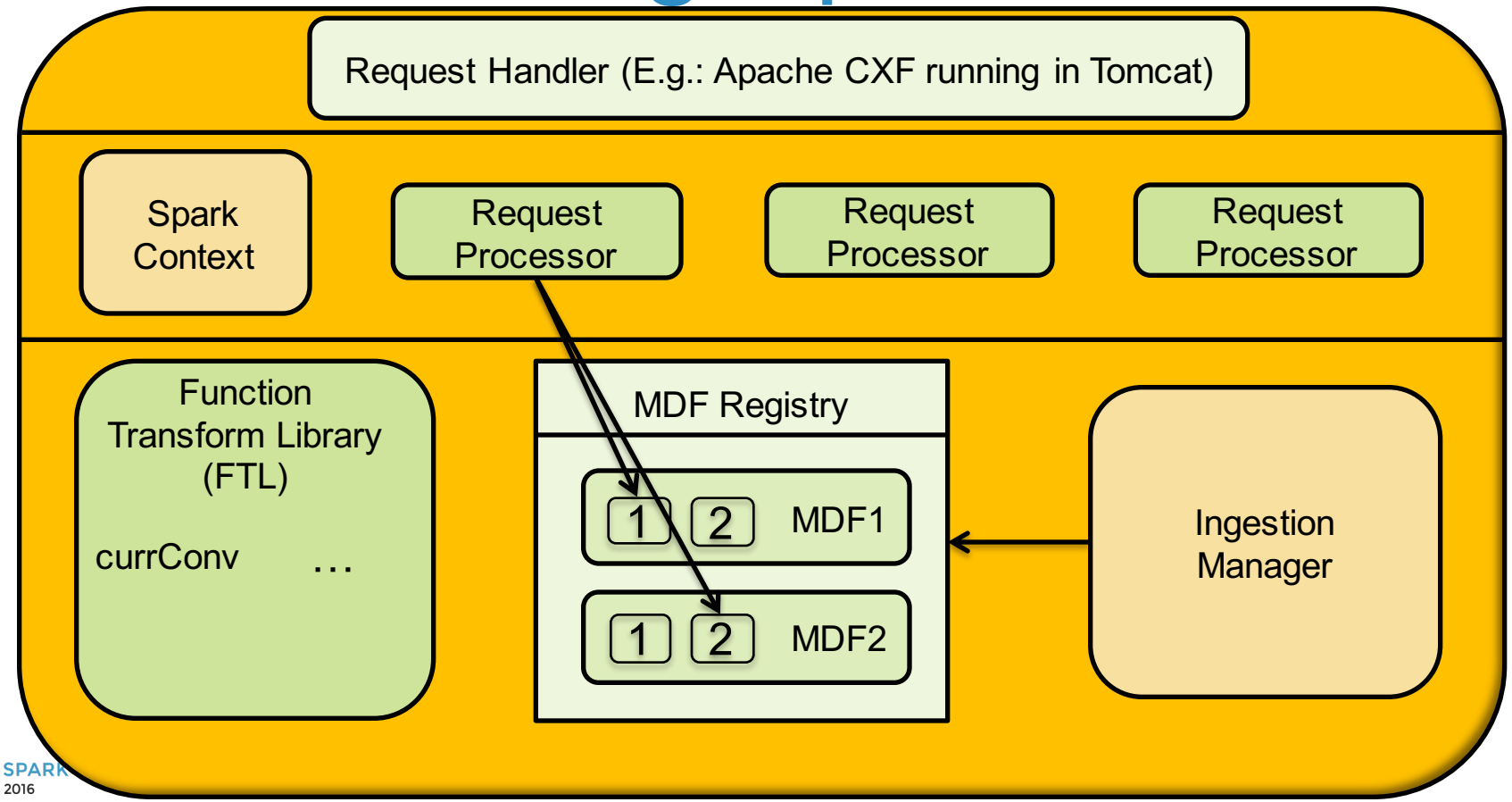
Bloomberg Spark Server



Bloomberg Spark Server



Bloomberg Spark Server



Schema Repository

- Enterprise-wide data pipeline
- External (to Spark) schema repository and service
- Enables MDF lookup by a dataset schema element
  - Analytic expressions can now be composed over data elements

Execution Metadata

- Connection Identifiers
  - Backing Stores
  - Real-time Topics
- Storage Level & Refresh Rate
- Subset Predicate, etc.

Cross-Domain Analytics

- Registration of pre-materialized DataFrames
    - Collaborative analytics between application workflows
- Dynamic creation of Managed DataFrames
    - Ad-hoc cross-domain analytics

Subsetting

- High value data sub-setted within Spark
    - Reduce cost of querying external datastore
- Specified as a filter predicate at time of registration
    - E.g. Member companies of popular indices [Dow 30, S&P 500,…] have records placed within Spark

Subsetting

- Seamless stitching between data in Spark (DF_{subset})and backing store ($DF_{subset'}$)
  - (DF_{subset} U $DF_{subset'}$).filter(query) = DF_{subset}.filter(query) U $DF_{subset'}$.filter(query)
  - Future: Predicate logic between query and subset predicates
- Dataset owners provided knobs for cost vs performance.
  - Future: LRU cache

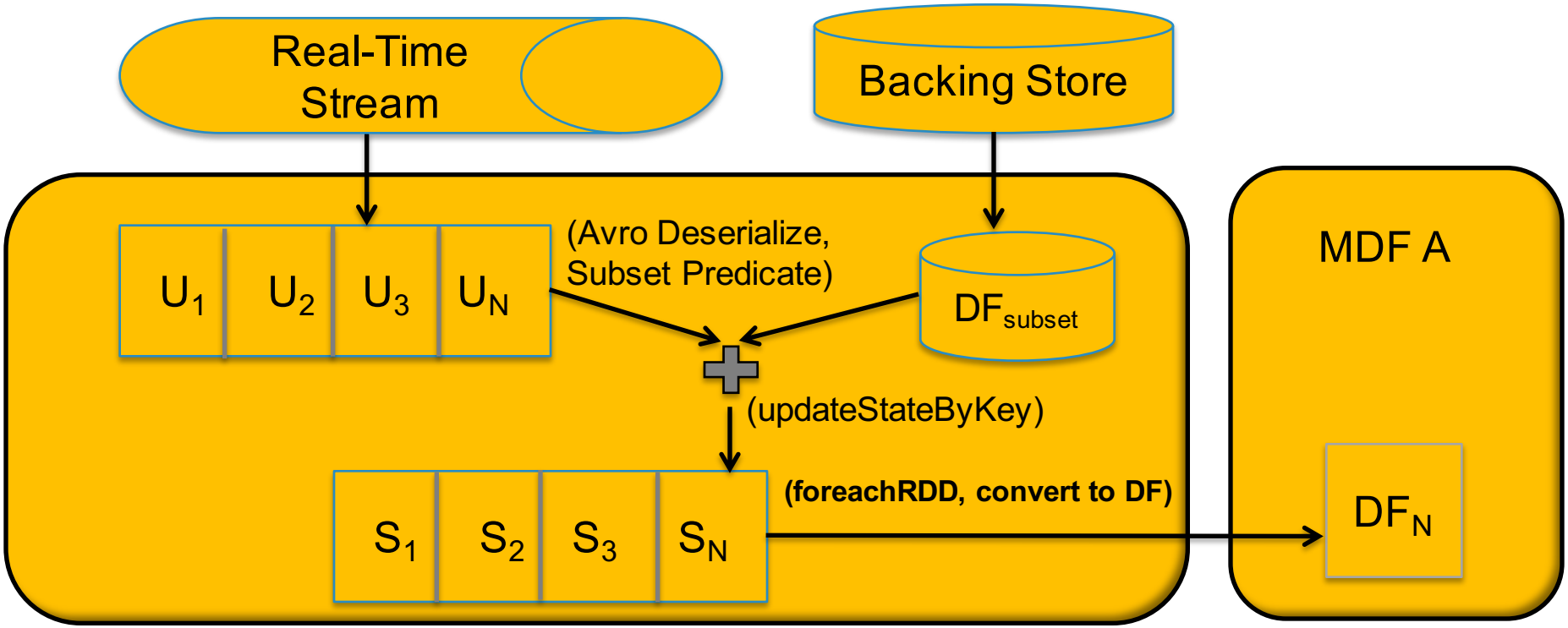
Ingestion: AB Swap

- Periodic data pull into Spark from the backing store
- Subset criteria applied during data retrieval
- Scenario when backing store kept continuously updated, external to Spark
- Avro-deserialization pushed down into various connectors

Ingestion: Stream Reconciliation

- Analytics needs to be low-latency with respect to queries, but also data freshness
- Since data is being sub-setted within Spark, need to keep the subset up to date
- Datasets published to different Kafka topics.
    - 1:1 mapping between datasets, topics and DStreams.

Ingestion: Stream Reconciliation



Reference Counting

- An MDF contains multiple generation of DFs, being generated and destroyed
- Multiple generations operated upon by RPs at given point in time
- Reference counting to keep track of what DFs are being used and by whom
- Long running queries aborted for forced reclamation

Snapshot Consistency

- Multiple queries need to operate on same snapshot of data
- How to achieve, if data constantly changing underneath?
- Each DF within MDF associated with time epoch
- Registry lookup with a reference time
- Time-align sub-setted dataframes with data in backing store

Spark Challenges

- Low latency performance consistency
- Efficient Stream reconciliation
- Spark Driver HA
- Strong consistency across contexts needs state externalization

MDF Acknowledgements

Andrew Foster

Joe Davey

Shubham Chopra

Nimbus Goehausen

Hamel Kothari

THANK YOU.

pnageswaran@bloomberg.net

skadambi@bloomberg.net

SPARK SUMMIT EAST
DATA SCIENCE AND ENGINEERING AT SCALE
FEBRUARY 16-18, 2016 NEW YORK CITY