SPARK SUMMIT
EUROPE 2017

Storage Engine Considerations for your Apache Spark Applications

Mladen Kovacevic, Senior Solutions Architect

Cloudera Inc.

#EUdev10

Outline

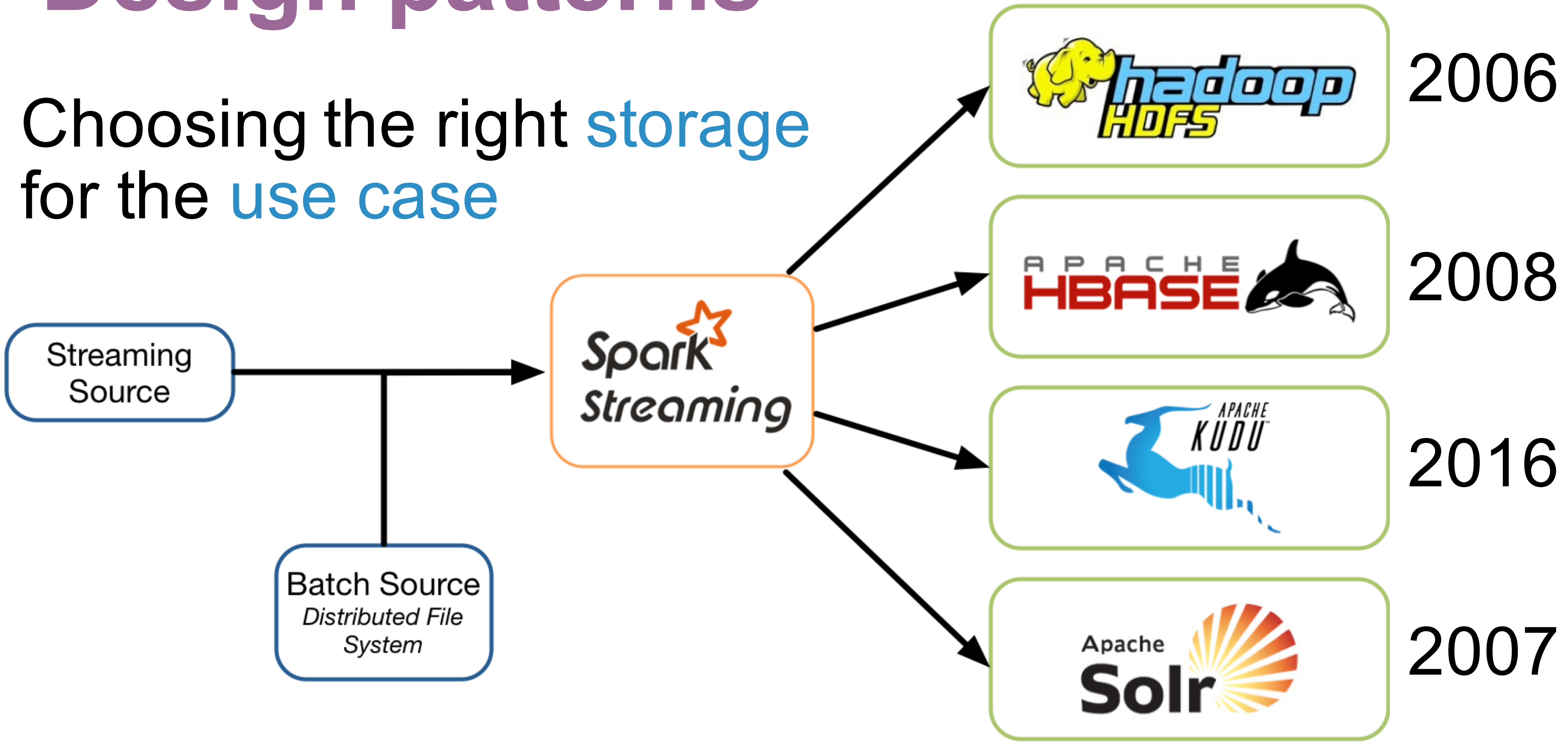
- Motivation – store your data – where *exactly*?
- Storage Capabilities:
  - HDFS
  - HBase
  - Kudu
  - Solr
- Asking the right questions
- Decide on right storage solution

Motivation

- Spark, SparkStreaming, SparkSQL – great for processing – need a place to store content
- Integration with variety of storage systems
- *Ingest* and *consumption* requirements – use case!

Design patterns

Choosing the right storage for the use case



HDFS

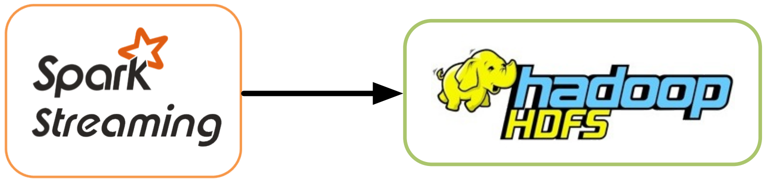
- Distributed file system – cheap, scalable, storage
- Immutable – “record” changes are painful
- Columnar file formats - ideal for analytics
- SQL overlays (SparkSQL, Hive Metastore, more) to define schema

Highlights

Very high throughput, painful random IO, batch oriented, coding overhead (ie. dealing with small files problems), any file

HDFS design pattern

```
df.write.parquet("/data/person_table")
```



- Small files accumulate
- External processes, or additional application logic to manage these files
- Partition management
- Manage metadata carefully (depends on ecosystem)
- Considerations- changing dimensions (fast/slow)
- Late arriving data

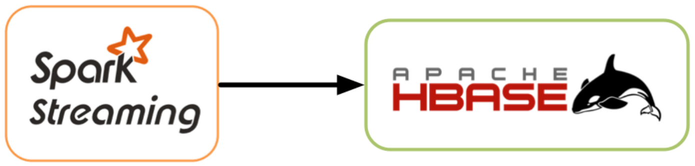
HBase

- NoSQL engine, manages files on HDFS
- Key-value, distributed storage engine
- No data types – just binary fields
- Thousands to millions of columns
- Store entity data (profiles of people, devices, accounts)

Highlights

Very fast random IO, low throughput, NRT oriented, challenging BI, no strict data types

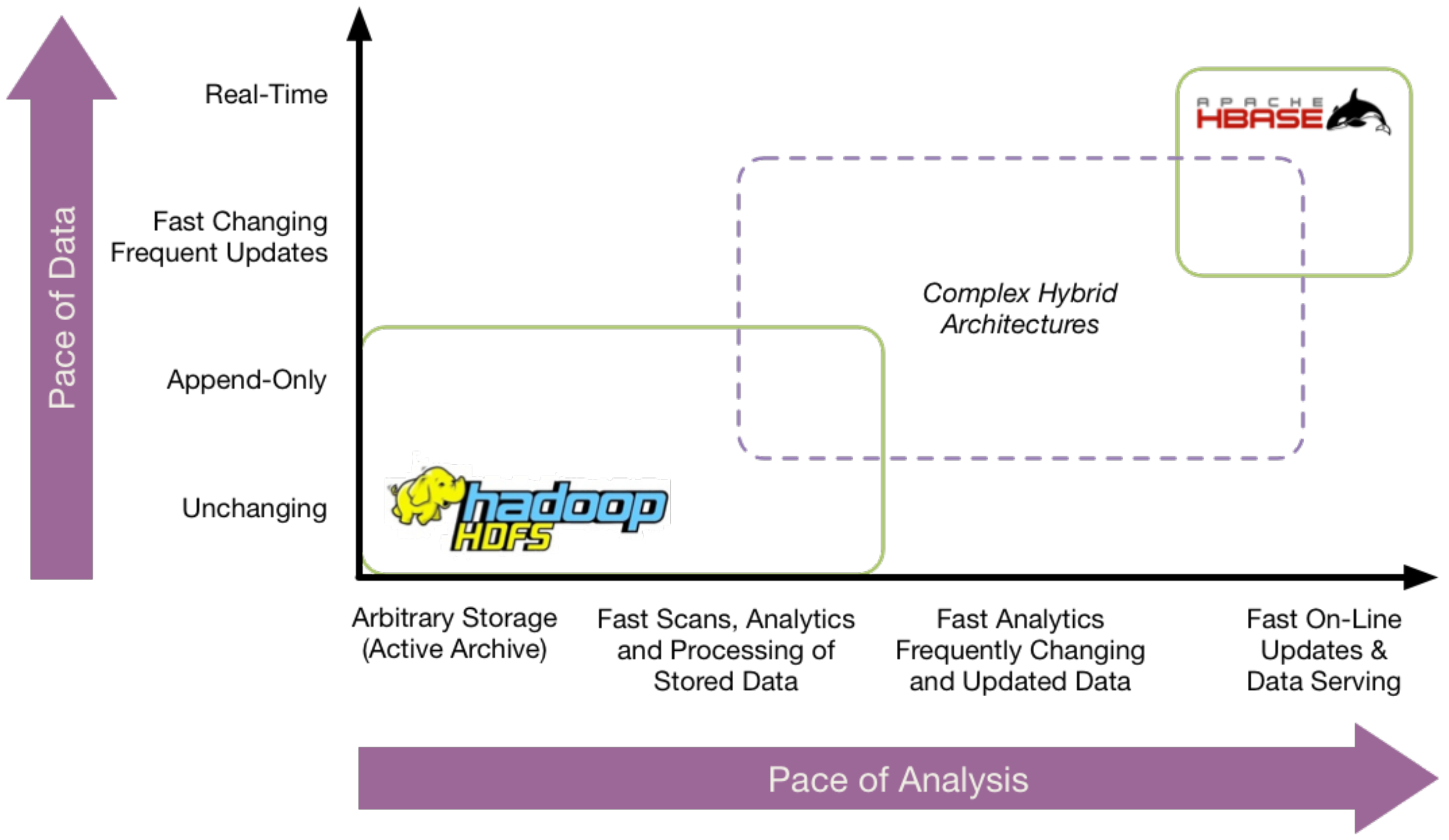
HBase design pattern



- HBase Connection anywhere in Spark/SparkStreaming app
- SparkSQL/DataFrames, Bulk Load
- Primary storage for ingestion or complementary preserving state
- NoSQL store vs. structured
- Near-real-time

CDH hbase-spark: https://github.com/cloudera/hbase/tree/cdh5-1.2.0_5.13.0/hbase-spark
CDH HBase and Spark docs: http://archive.cloudera.com/cdh5/cdh/5/hbase-1.2.0-cdh5.13.0/book.html#spark
Upstream hbase-spark (watch for updates in HBase 2.x release): https://github.com/apache/hbase/tree/master/hbase-spark/
Upstream HBase and Spark book: http://hbase.apache.org/book.html#spark

Analytic Gap


Pace of Data
Real-Time
Fast Changing Frequent Updates
Append-Only
Unchanging
hadoop HDFS
Complex Hybrid Architectures
APACHE HBASE
Arbitrary Storage (Active Archive)
Fast Scans, Analytics and Processing of Stored Data
Fast Analytics Frequently Changing and Updated Data
Fast On-Line Updates & Data Serving
Pace of Analysis

Kudu

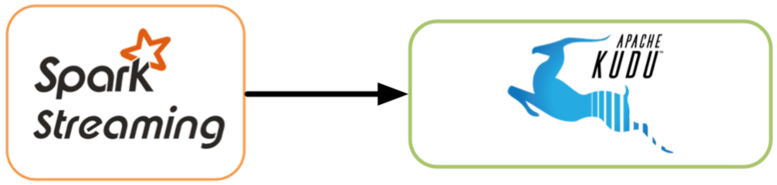
- Storage system for tables of structured data
- Bring-your-own-SQL (SparkSQL, Impala), NoSQL-like API, integration with Spark, MapReduce, more..
- Columnar, key partitioning by range and/or hash
- Limited number of columns (strongly typed)

Highlights

Fast random IO, fast throughput, NRT oriented, terrific for BI, structured data

Kudu design pattern

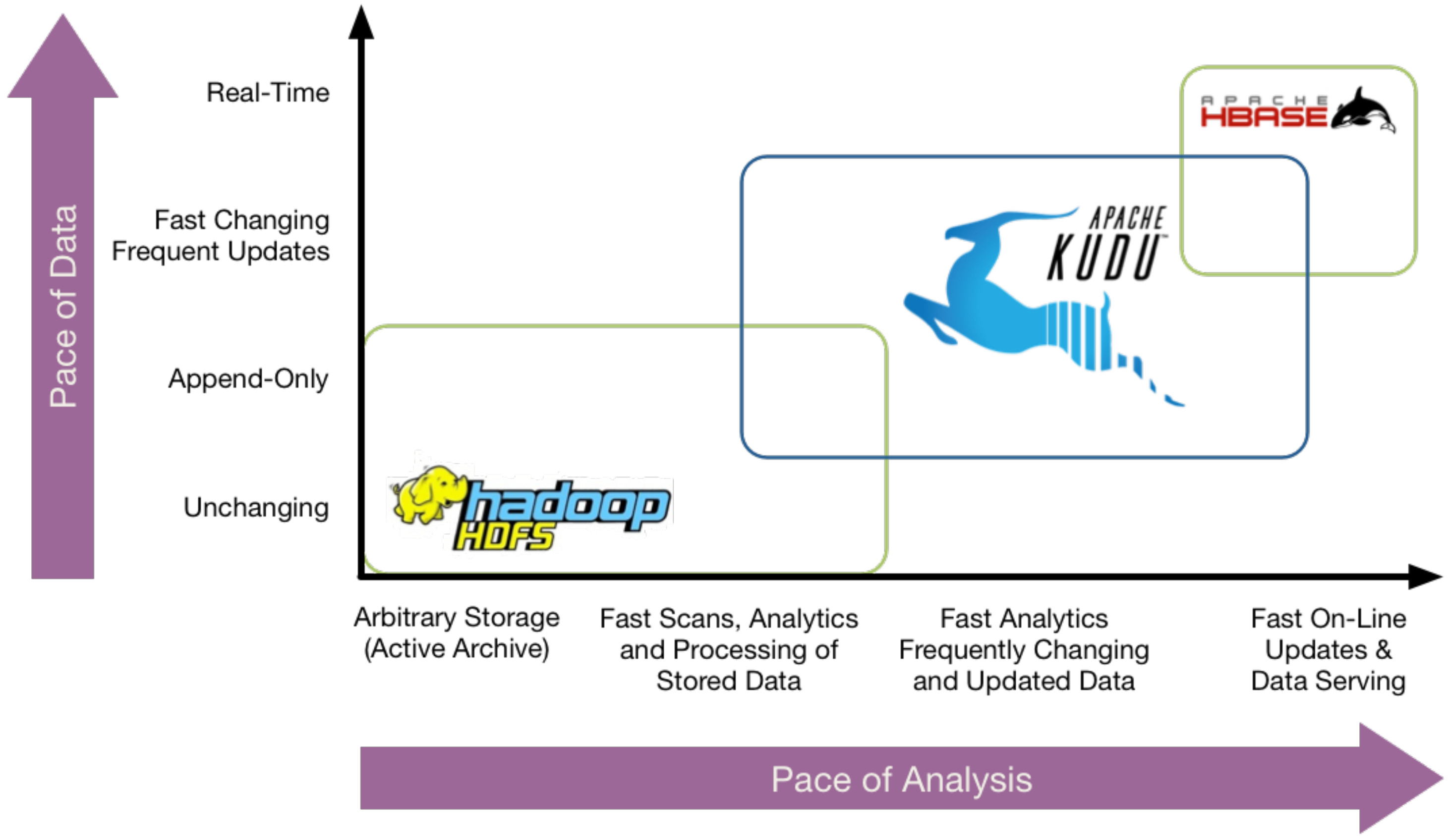
```
df.write.options(kuduOptions).mode(“append”).kudu
                         OR
               kuduContext.insertRows()
```



- DataFrame perfect match for Kudu (structured)
- Data available *immediately* to SQL engines (Impala, SparkSQL)
- Ideal case is append with moderate updates

Kudu Integration with Spark: http://kudu.apache.org/docs/developing.html#_kudu_integration_with_spark
Up and running with Apache Spark on Apache Kudu: https://blog.cloudera.com/blog/2017/02/up-and-running-with-apache-spark-on-apache-kudu/

Analytic Gap Filled



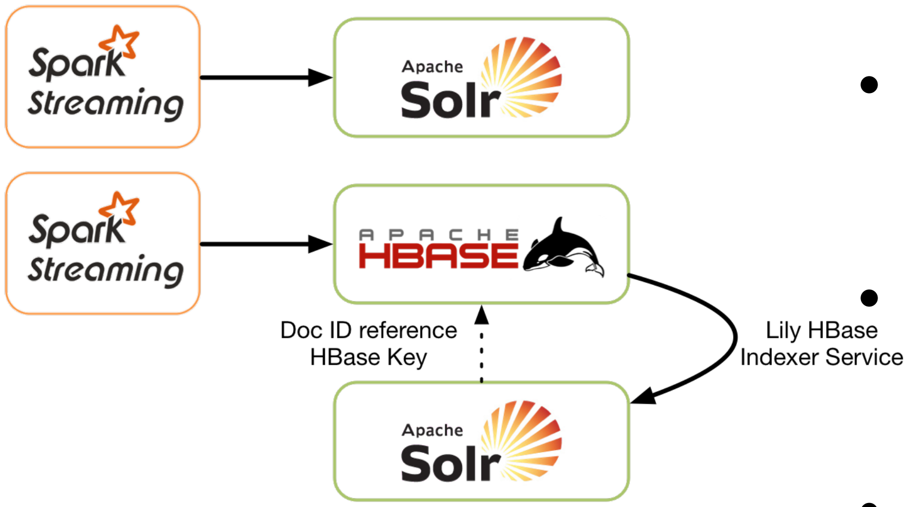
Solr

- Distributed index enabling search capabilities (Lucene)
- Typed, REST API based, search index query processing
- Search interface, faceting, integration with HBase storing content (typically) in HDFS

Highlights

High random IO, low throughput, multi-faceted use cases, NRT oriented, terrific for BI with the right tools (non-SQL), loose schema, data types

Solr design pattern



- Prepare Solr document, add to SolrCloud directly OR
- Write to HBase, leverage Lily HBase Indexer service to update Solr
- Store complete record in HBase, while indexed fields for search in Solr
- NRT availability (short soft commits)

Questions we ask (1)

- How many voters have cast their ballots by city thus far in the election, by the second?
  - streaming data into ‘voter’ table, aggregate query, immediate data availability : Kudu
- How many people watched last nights game compared to the night before?
  - daily batch, aggregate query : HDFS parquet

Questions we ask (2)

- What version is my device running and how many dropped packets do I have?
  - streaming entity profile data, metrics may change per release, many updates, specific device, NRT: HBase
- Which tweets talk to the housing market, in the 21-30 age group?
  - streaming, keyword search, facet filtering : Solr

Use case questionnaire

- Consumption interface: SQL (JDBC/ODBC) vs. API
- Near-real-time requirement for consumers
- Ingestion rate (can we keep up?)
- Entity vs. Events (time-based)
- Append-only vs moderate updates vs many updates
- Distinct values in dataset

Storage considerations (1)

Criteria	hadoop HDFS	APACHE HBASE	APACHE KUDU	Apache Solr
SQL interface	🟢	🟡	🟢	🔴
API interface	🔴	🟢	🟢	🟢
Near-real-time ingestion	🔴	🟢	🟢	🟢
Append-only + available for query	🟡	🟢	🟢	🟢
Appends with moderate updates	🔴	🟢	🟢	🟢
Mostly updates	🔴	🟢	🟡	🟢

Storage considerations (2)

Criteria	hadoop HDFS	APACHE HBASE	APACHE KUDU	Apache Solr
Entity based data	🔴	🟢	🟠	🟢
Event based data (time-series)	🟠	🟠	🟢	🟠
High distinct values	🟢	🟢	🟢	🔴
Many and unknown attributes	🔴	🟢	🔴	🟠
Binary data (Images, PDFs, etc)	🟢	🟢	🔴	🔴
Analytics	🟢	🔴	🟢	🟠

Wrap-up

- Review entire use-case *end-to-end* early
- Understand storage capabilities
- Ask the right questions (upstream/consumers)
- Consider security, architecture and development costs
- Decide on the right storage solution