



DIPPING INTO THE BIG DATA **RIVER**

STREAM ANALYTICS AT SCALE

RADEK OSTROWSKI



bit.do/topt







TODAY'S STORY





TODAY'S STORY

Challenge



Let's use streaming data to recommend products!



[HTTPS://PBS.TWIMG.COM/MEDIA/BTTEGLHIYAALZJR.JPG](https://pbs.twimg.com/media/BTTEGLHIYAALZJR.jpg)



TODAY'S STORY

Solution



TECH STACK



Falcon

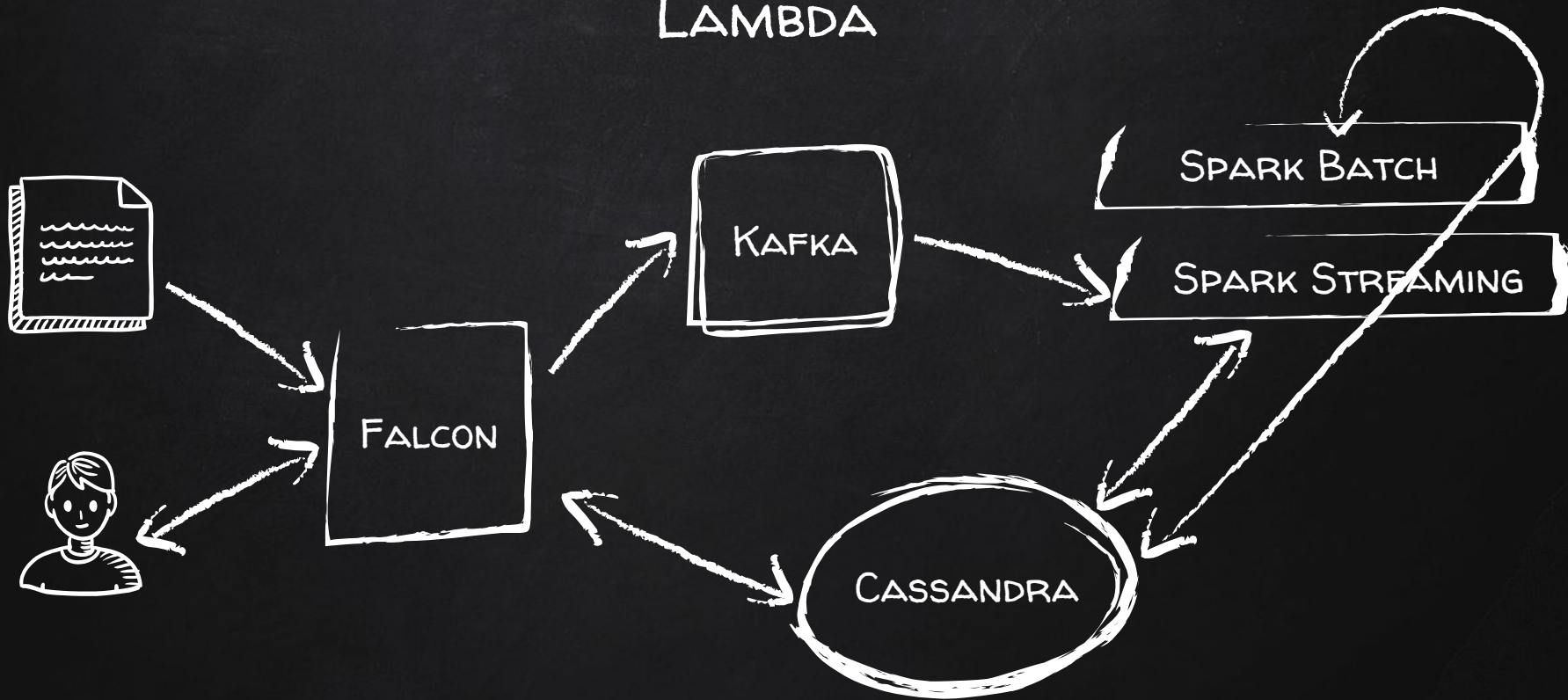


Expedia Hotel Recommendations

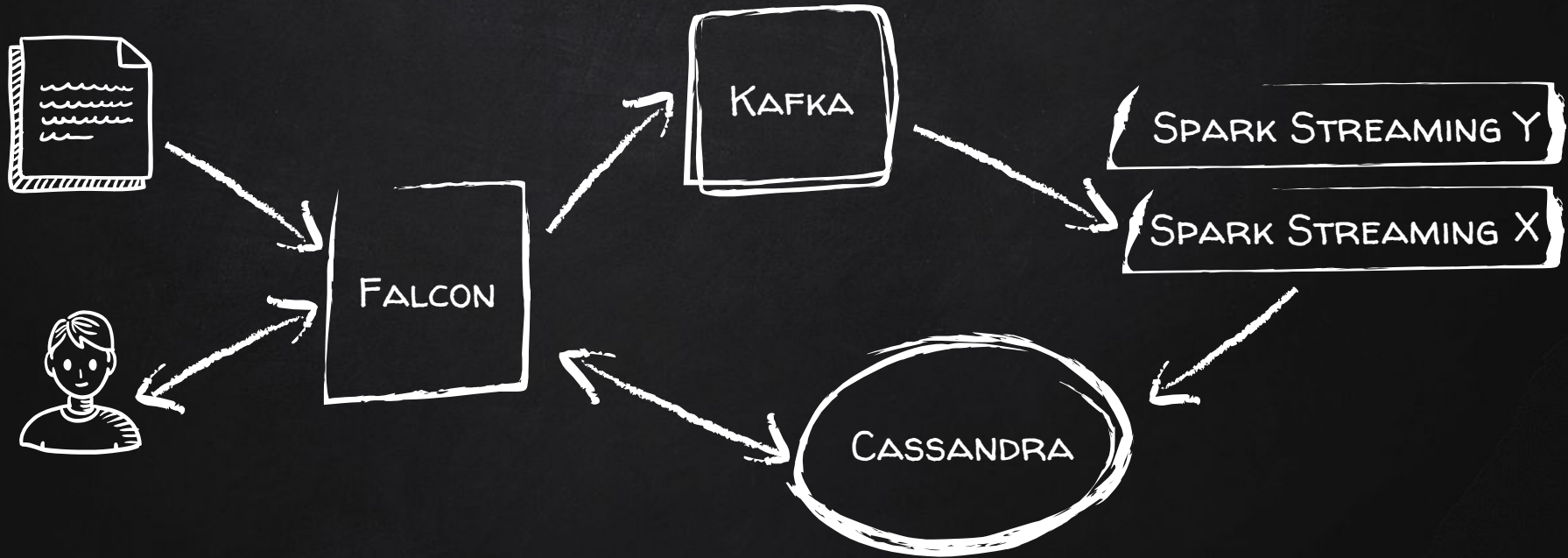
Which hotel type will an Expedia customer book?
\$25,000 · 1,974 teams · a year ago

Overview [Data](#) Kernels Discussion Leaderboard Rules

λ
LAMBDA



K
KAPPA





LAMBDA VS KAPPA

Why Lambda?

Batch business as usual, but addition of lightweight stream brings latest information

Why not?

Complexities in maintaining two separate tech stacks/infrastructures/code

Client needs to combine both outputs

Why Kappa?


























Just one tech stack/infrastructure to maintain

Why not?

Full on stream process could still be challenging, you'll be working at the cutting edge





[HTTPS://GERARDNICO.COM/WIKI/_DETAIL/DATA_MINING/
RATING_COLLABORATIVE_FILTERING.PNG](https://gerardnico.com/wiki/_detail/data_mining/rating_collaborative_filtering.png)

jupyter Kappa - Collaborative Filtering

Last Checkpoint: a minute ago (autosaved)



File Edit View Insert Cell Kernel Widgets Help

Trusted

Python 2



```
In [ ]: def buildCFModel(train):
    def isProductToRating(productCount, clickCount):
        return (productCount * 3.0) + clickCount

    ratings = train.rdd.\
        map(lambda r: Rating(r.user_id, r.product, isProductToRating(r.purchased_count, r.clicked_count)))
    rank = 10
    numIterations = 20
    lambdaFactor = 0.01
    alpha = 0.01
    seed = 42
    return ALS.trainImplicit(ratings, rank, numIterations, alpha, seed=seed)

In [ ]: def recommendTopProducts(dfModel):
    numberOfRecommendationsRequired = 5
    rdd = dfModel.recommendProductsForUsers(numberOfRecommendationsRequired)
    recommendations = rdd.map(lambda (user, ratings): (user, map(lambda r: r.product, ratings)))
    topRecommendationsSchema = StructType([
        StructField("user_id", IntegerType(), False),
        StructField("recommended_products", ArrayType(IntegerType()), False)
    ])
    return sql.createDataFrame(recommendations, topRecommendationsSchema)

In [ ]: def processStream(rdd):
    df = sql.read.json(rdd)
    if len(df.columns):
        #store updated counters in C*
        df.withColumn('c', separateClicks_udf(df['is_purchase'])).\
            select("user_id", "product", "c.purchased_count", "c.clicked_count").\
            write.format("org.apache.spark.sql.cassandra").mode('append').\
            options(table="users_interests", keyspace="bdr").save()

        #read all data from C*
        usersInterests = sql.read.format("org.apache.spark.sql.cassandra").\
            options(table="users_interests", keyspace="bdr").load().cache()

    dfModel = buildCFModel(usersInterests.select("user_id", "product", "clicked_count", "purchased_count"))
    top5 = recommendTopProducts(dfModel)
    top5.show()
    top5.write.format("org.apache.spark.sql.cassandra").mode('append').options(table="cf", keyspace="bdr").save()
```



TODAY'S STORY

Summary



SUMMARY



Don't just store your data. Take advantage of it when it's in motion.



Kappa vs Lambda will depend on your use case/culture/platform.



Use technologies that scale
e.g. SMACK stack.



Docker facilitates quick development and portability.

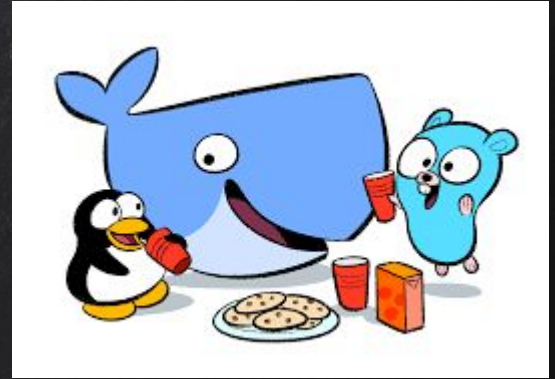


Machine Learning is more accessible than ever before. Use it!



Not sure yet? Run a POC.
It's easier than it sounds.

CREDITS



GitHub: <http://github.com/radek1st/BigDataRiver>

Free presentation template by SlidesCarnival

Unsplash photos by Jérôme Prax, Jeremy Bishop and
Matthew Sleeper

THANKS!

ANY QUESTIONS?

rostrow@gmail.com

