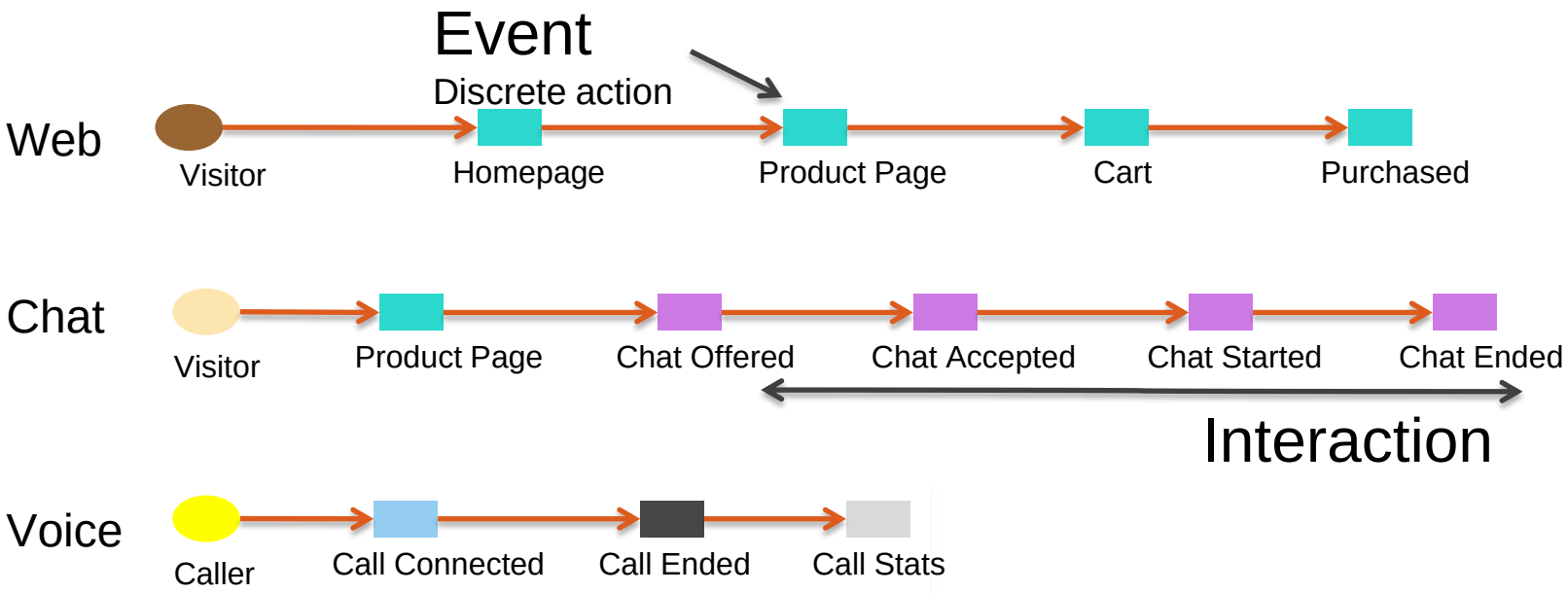
Automated Machine Learning @[24]7

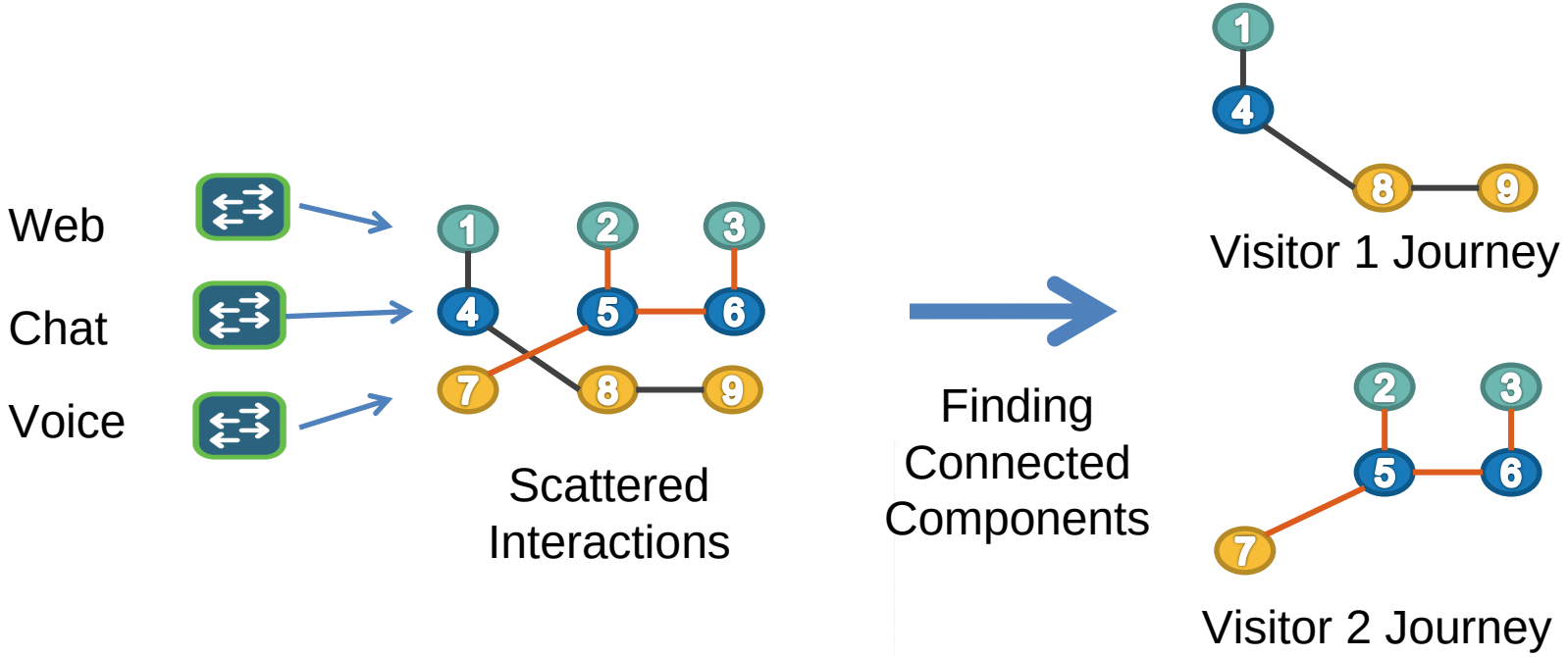
Sourabh Chaki

Anticipate, Simplify, Learn

A Proactive Framework for Intuitive Customer Care



Graph Problem



Graph processing in Hadoop

- HashToMin for Connected Component
  - http://arxiv.org/pdf/1203.5387.pdf
- Log(n) mapreduce iterations for connecting *n* diameter graph
- Challenges:
  - Thinking graph problem in map and reduce
  - Not fit for iterative algorithms

GraphX

- Spark fits for iterative programming
- In-memory data
- Think as edges and vertices
- High level APIs for graph processing

Connected Component in GraphX

- Vertices >> Edges
  - Graph from only edges
  - Interactions as external KVP RDD
- val cc = graph.connectedComponents()
- Join cc.vertices with interactions
- Group interactions with leaders

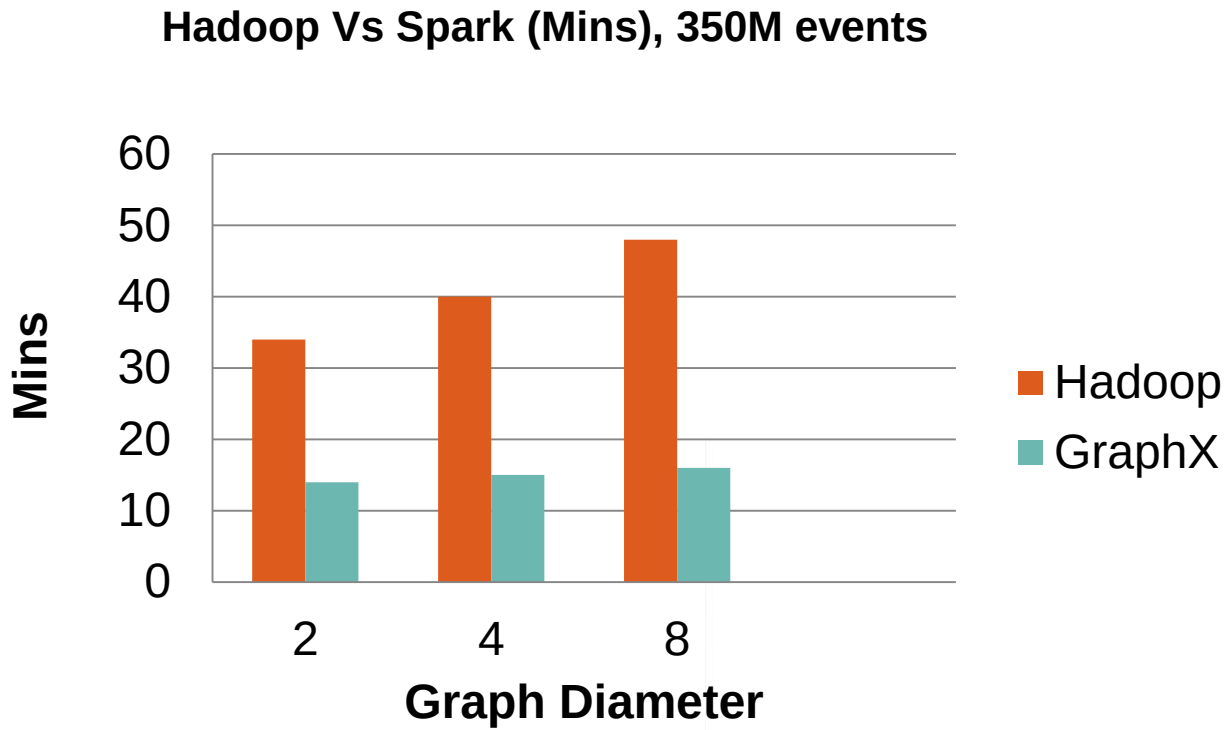
Join

- Broadcast join
- Hash Partition Join
- Adaptive
  - Small data: Broadcast
  - Large data: HashPartition

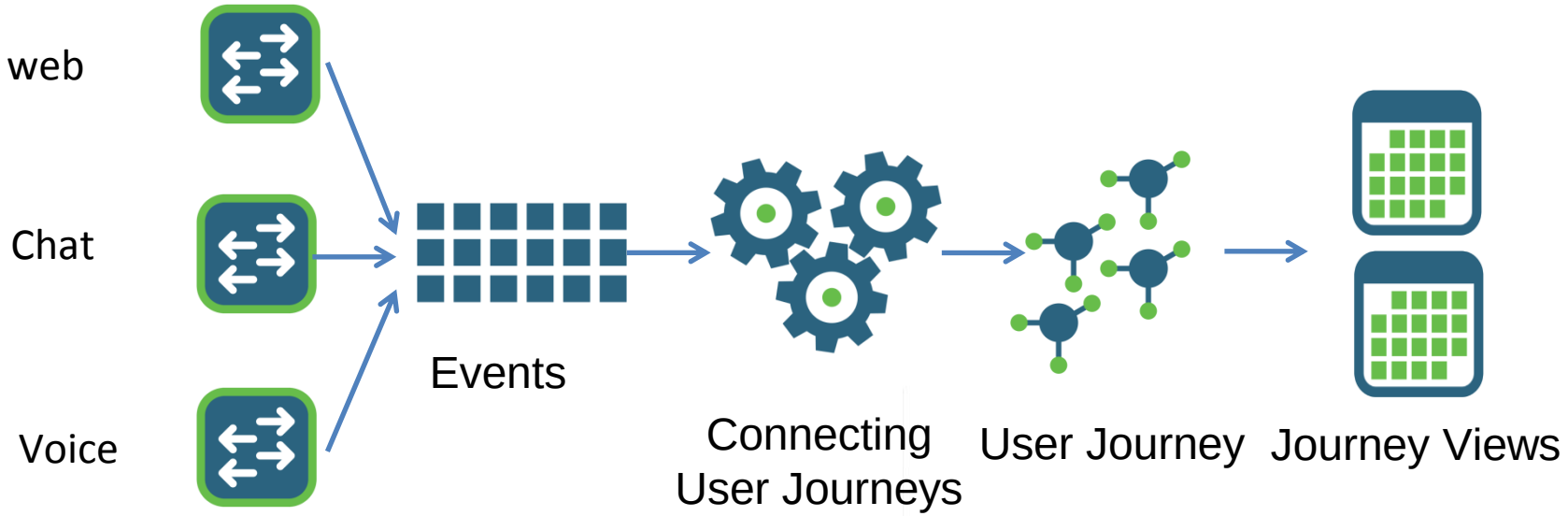
Memory Share

- Default shuffle memory 20%
  - Low for shuffle heavy application
- Shuffle memory: 40%, storage memory 40%

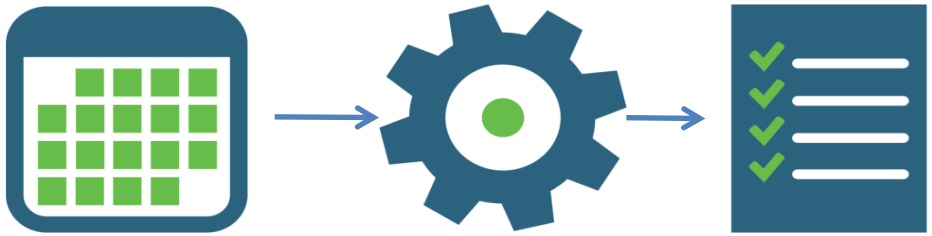
Performance Gain



Interactions to Journey



User Journey to Model



Journey View

Pre-processing and model building

Model

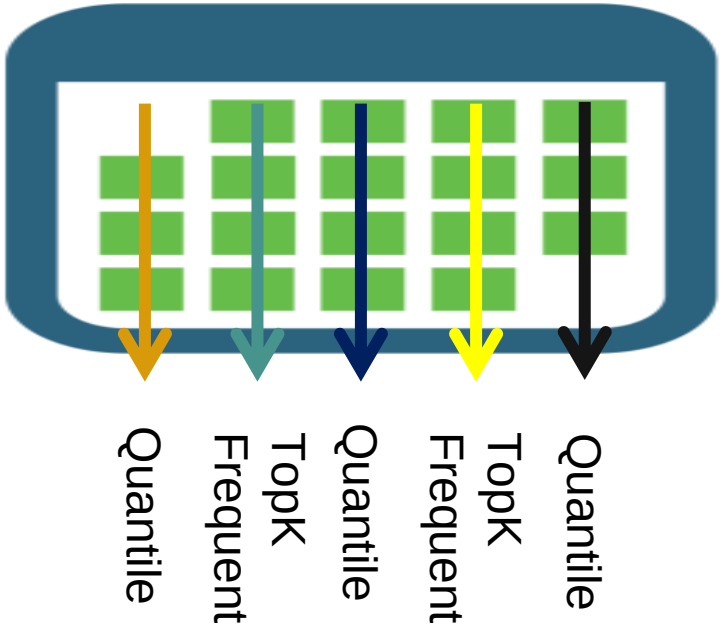
Feature Engineering

- Training Set/Test Set
- Balanced Sampling
- Frequent Items
- Quantile Bins
- Transformation Function

Feature Engineering Config

```
trainTest=60,40
#features
catVarIndices=1,2
contVarIndices=3,4
labelIndex=0
#transformation
catVar.1.bin=5
catVar.2.bin=4
contVar.3.bin=5
contVar.4.bin=6
#sampling
label.1.wt=10
label.2.wt=100
```

Column wise data analysis



- ✓ Sum (set1 , set2) = Sum(set1) + Sum(set2)
- ✖ Quantile(Set1 + Set2) = Quantile(Set1) + Quantile(Set2)
- ✖ TopKFrequent(set1, set2) = TopKFrequent(set1) + TopKFrequent(set2)

Problems

- Non distributed
- Need data shuffle
- Shuffle on different columns
- High disk and network IO

Algebird with Spark

F(set1 + set2) ≈ F(set1) + F(set2)

- QTree
- CountMinSketch

https://github.com/twitter/algebird

Binning Product Price

Product price => low / high value product

```
val aggregator= new QTreeSemigroup[Double](8)
val qtree = journeyViewRDD
                .map(row=>QTree(row(productPriceIndex)))
                .reduce(aggregator.plus(_,_) )
val median = qtree.quantileBounds(0.5)._1
```

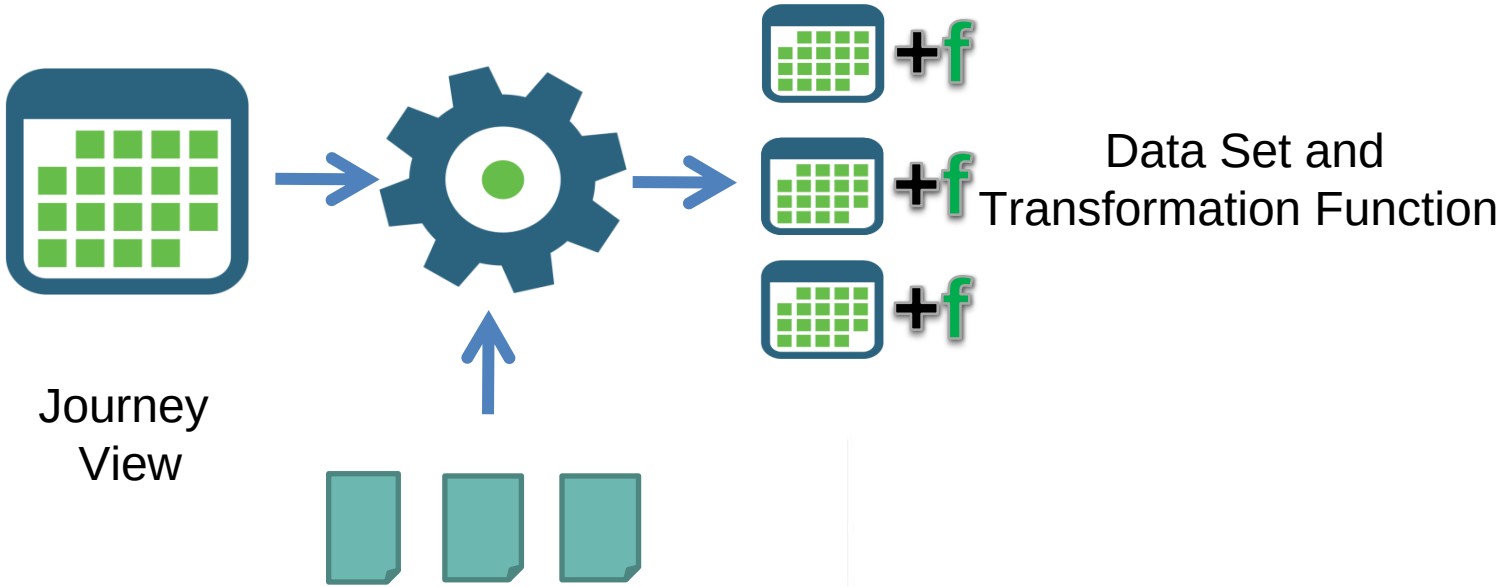
Binning Product Price

Product price => low / high value product

```
class TransformationFunc(val median:Double) extends Serializable {
   def transform (value:Double): String = {
     if(value<median) "low"
     else "high"
   }
 }

val transFunc = new TransformationFunc(median)
val priceCat = transFunc.transform(productPrice)
```

Automated Feature Engineering



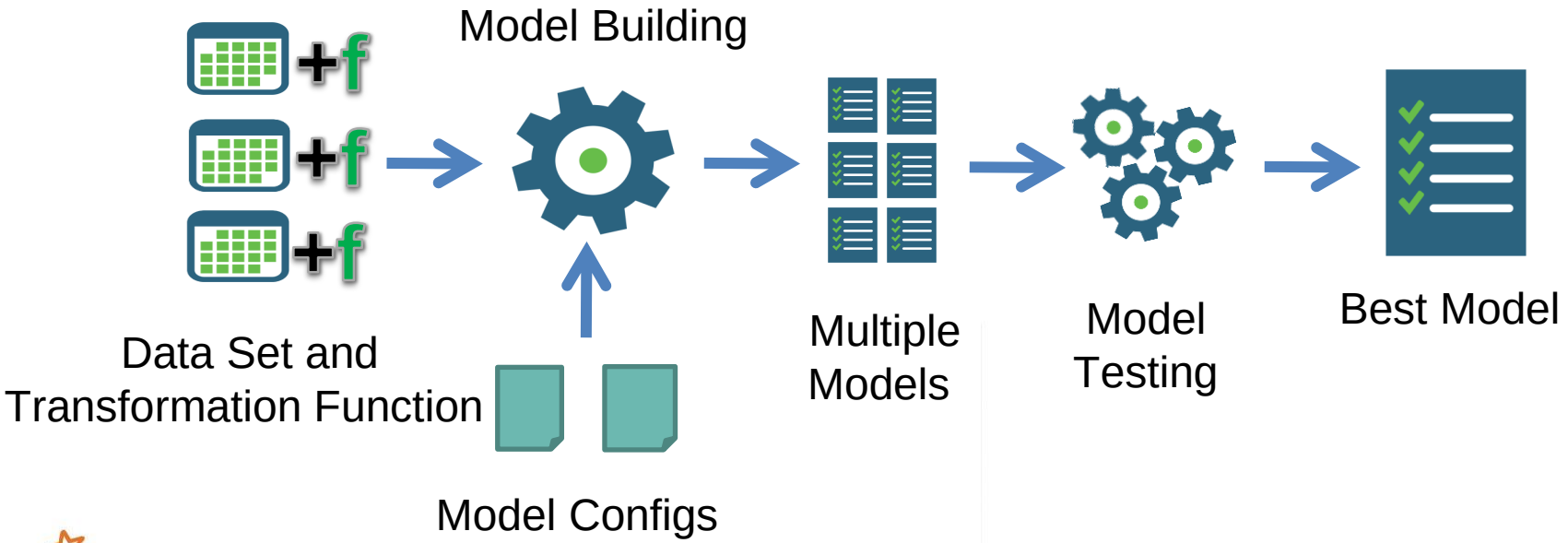
Model Building

- Spark MLlib
- Random Forest for classification
- Model Testing
- Performance Metrics

Model Config

```
#model
model=randomforest
randomforest.algo=gini
randomforest.bin=20
randomforest.classes=2
randomforest.depth=10
randomforest.numTrees=10
randomforest.featureSubsetStrategy=auto
#model metric
modelMetric=ROC
```

Automated Model Building



Prediction Entity

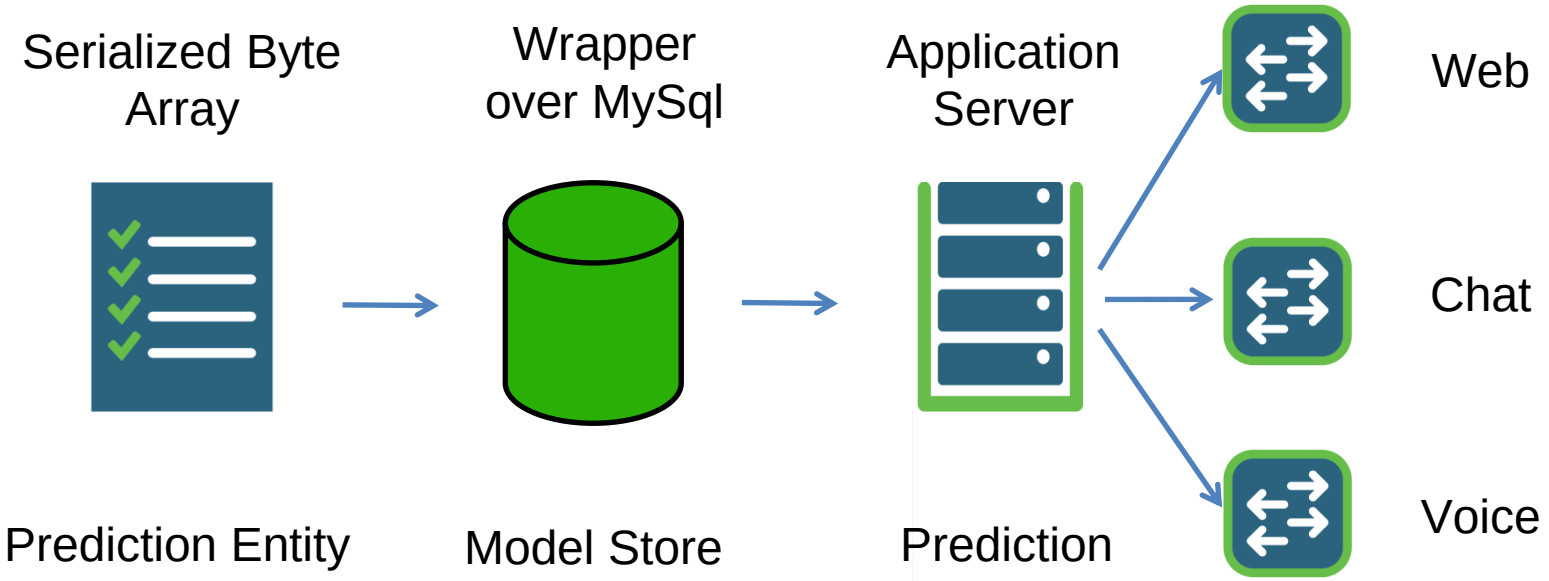
- Transformation Function Object
- Random Forest Model Object

```
class PredictionEntity(model:RandomForestModel,
                       tansformationFunc:TransformationFunc)
                       extends Serializable {
  def predict(vector:Vector):Double = {
    //transform the vector using transformationFunc
    //predict using model
  }
}
```

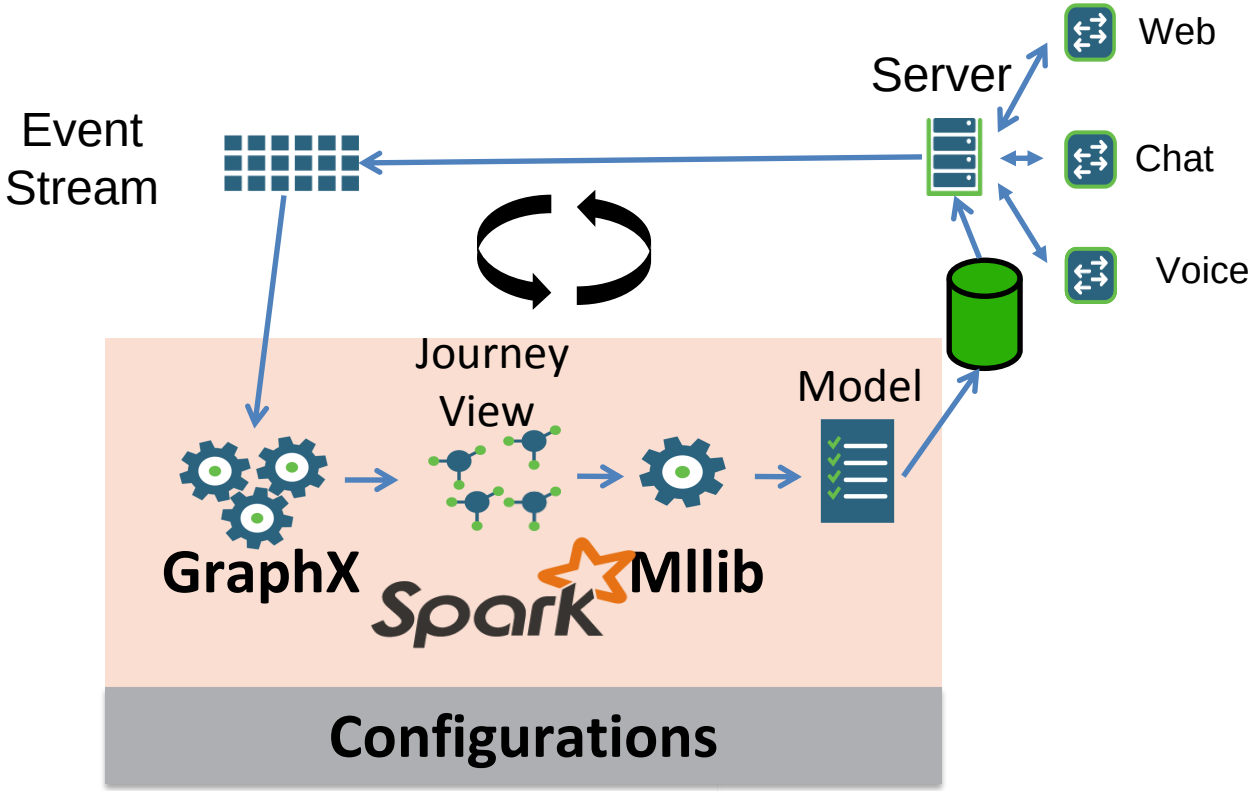
Prediction outside Spark

- Need export
  - No PMML support
- Model store
- Synchronous prediction call

Prediction outside Spark



Machine Learning Cycle



Thank You