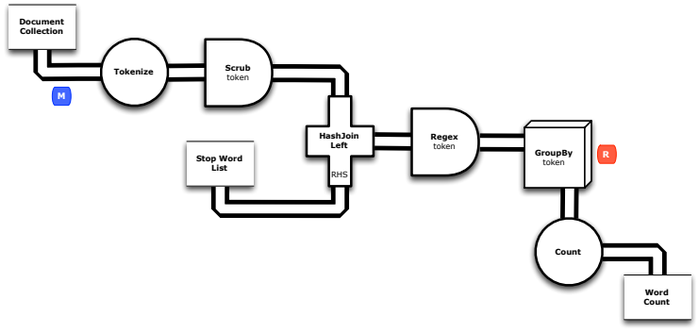
Use Case Patterns in Enterprise Big Data with Cascading

Paco Nathan
Concurrent, Inc.

pnathan@concurrentinc.com
@pacoid



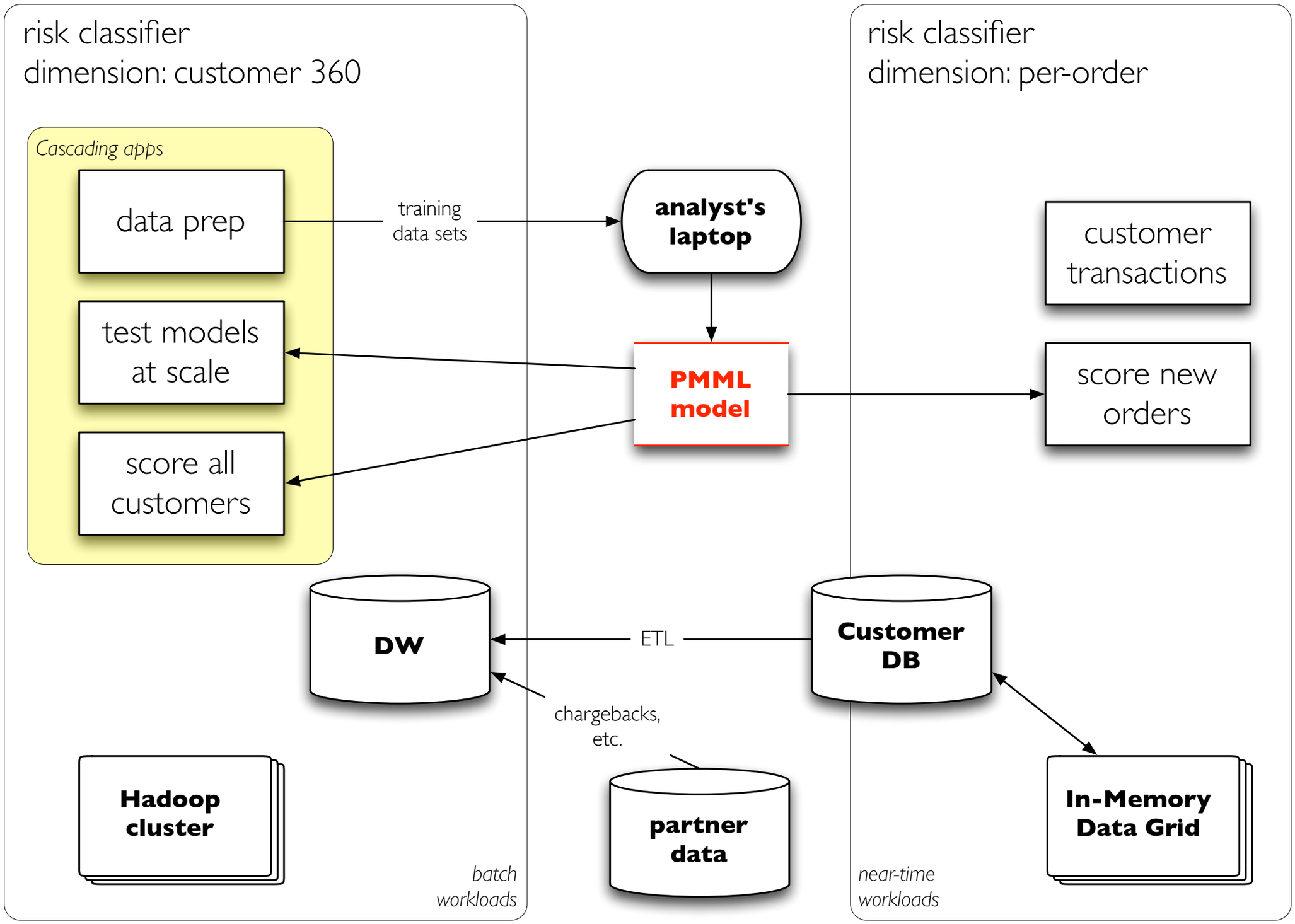
concurrent

cascading.pattern

example:

1. use customer order history as the training data set
2. train a risk classifier for orders, using Random Forest
3. export model from R to PMML
4. compile a Cascading app to execute the PMML model
5. deploy the app at scale to calculate scores

cascading.pattern



1:
“orders” data set…
train/test in R…
exported as PMML

R modeling

```
## train a RandomForest model

f <- as.formula("as.factor(label) ~ .")
fit <- randomForest(f, data_train, ntree=50)

## test the model on the holdout test set

print(fit$importance)
print(fit)

predicted <- predict(fit, data)
data$predicted <- predicted
confuse <- table(pred = predicted, true = data[,1])
print(confuse)

## export predicted labels to TSV

write.table(data, file=paste(dat_folder, "sample.tsv", sep="/"),
quote=FALSE, sep="\t", row.names=FALSE)

## export RF model to PMML

saveXML(pmml(fit), file=paste(dat_folder, "sample.rf.xml", sep="/"))
```

R output

```
     MeanDecreaseGini
var0        0.6591701
var1       33.8625179
var2        8.0290020

        OOB estimate of  error rate: 13.83%
Confusion matrix:
   0  1 class.error
0 28  5   0.1515152
1  8 53   0.1311475

[1] "./data/sample.rf.xml"
```

2: Cascading app takes PMML as a parameter...

PMML model


```
<?xml version="1.0"?>
<PMML version="4.0" xmlns="http://www.dmg.org/PMML-4_0"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.dmg.org/PMML-4_0
 http://www.dmg.org/v4-0/pmml-4-0.xsd">
 <Header copyright="Copyright (c) 2012 ceteri" description="Random Forest Tree Model">
  <Extension name="user" value="ceteri" extender="Rattle/PMML"/>
  <Application name="Rattle/PMML" version="1.2.30"/>
  <Timestamp>2012-10-22 19:39:28</Timestamp>
 </Header>
 <DataDictionary numberOfFields="4">
  <DataField name="label" optype="categorical" dataType="string">
   <Value value="0"/>
   <Value value="1"/>
  </DataField>
  <DataField name="var0" optype="continuous" dataType="double"/>
  <DataField name="var1" optype="continuous" dataType="double"/>
  <DataField name="var2" optype="continuous" dataType="double"/>
 </DataDictionary>
 <MiningModel modelName="randomForest_Model" functionName="classification">
  <MiningSchema>
   <MiningField name="label" usageType="predicted"/>
   <MiningField name="var0" usageType="active"/>
   <MiningField name="var1" usageType="active"/>
   <MiningField name="var2" usageType="active"/>
  </MiningSchema>
  <Segmentation multipleModelMethod="majorityVote">
   <Segment id="1">
    <True/>
    <TreeModel modelName="randomForest_Model" functionName="classification" algorithmName="randomForest"
splitCharacteristic="binarySplit">
     <MiningSchema>
      <MiningField name="label" usageType="predicted"/>
      <MiningField name="var0" usageType="active"/>
      <MiningField name="var1" usageType="active"/>
      <MiningField name="var2" usageType="active"/>
     </MiningSchema>
...
```

Cascading app


```
public class Main {
  public static void main( String[] args ) {
    String pmmlPath = args[ 0 ];
    String ordersPath = args[ 1 ];
    String classifyPath = args[ 2 ];
    String trapPath = args[ 3 ];

    Properties properties = new Properties();
    AppProps.setApplicationJarClass( properties, Main.class );
    HadoopFlowConnector flowConnector = new HadoopFlowConnector( properties );

    // create source and sink taps
    Tap ordersTap = new Hfs( new TextDelimited( true, "\t" ), ordersPath );
    Tap classifyTap = new Hfs( new TextDelimited( true, "\t" ), classifyPath );
    Tap trapTap = new Hfs( new TextDelimited( true, "\t" ), trapPath );

    // define a "Classifier" model from PMML to evaluate the orders
    Classifier model = ClassifierFactory.getClassifier( pmmlPath );
    Pipe classifyPipe = new Each( new Pipe( "classify" ), model.getFields(),
      new ClassifierFunction( new Fields( "score" ), model ), Fields.ALL );

    // connect the taps, pipes, etc., into a flow
    FlowDef flowDef = FlowDef.flowDef().setName( "classify" )
     .addSource( classifyPipe, ordersTap )
     .addTrap( classifyPipe, trapTap )
     .addSink( classifyPipe, classifyTap );

    // write a DOT file and run the flow
    Flow classifyFlow = flowConnector.connect( flowDef );
    classifyFlow.writeDOT( "dot/classify.dot" );
    classifyFlow.complete();
  }
}
```

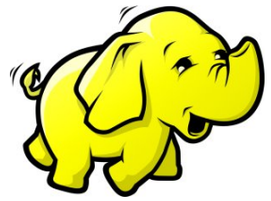
3:
app deployed on a cluster to score customers at scale...

deploy to cloud


```
elastic-mapreduce --create --name "RF" \
  --jar s3n://temp.cascading.org/pattern/pattern.jar \
  --arg s3n://temp.cascading.org/pattern/sample.rf.xml \
  --arg s3n://temp.cascading.org/pattern/sample.tsv \
  --arg s3n://temp.cascading.org/pattern/out/classify \
  --arg s3n://temp.cascading.org/pattern/out/trap
```

aws.amazon.com/elasticmapreduce/

results



```
bash-3.2$ head output/classify/part-00000
label  var0   var1   var2   order_id  predicted score
1   0   1   0   6f8e1014  1   1
0   0   0   1   6f8ea22e  0   0
1   0   1   0   6f8ea435  1   1
0   0   0   1   6f8ea5e1  0   0
1   0   1   0   6f8ea785  1   1
1   0   1   0   6f8ea91e  1   1
0   1   0   0   6f8eaaba  0   0
1   0   1   0   6f8eac54  1   1
0   1   1   0   6f8eade3  1   1
```

drill-down

blog, code/wiki/gists, JARs, community, DevOps products:

cascading.org

github.org/Cascading

conjars.org

meetup.com/cascading

goo.gl/KQtUL

concurrentinc.com

pnathan@concurrentinc.com
@pacoid