Scal(a)ing up Machine Learning and Big Graph Analytics

Tyson Condie, Markus Weimer and Raghu Ramakrishnan
Yahoo! Research

Vinayak Borkar, Yingyi Bu and Mike Carey
U.C. Irvine

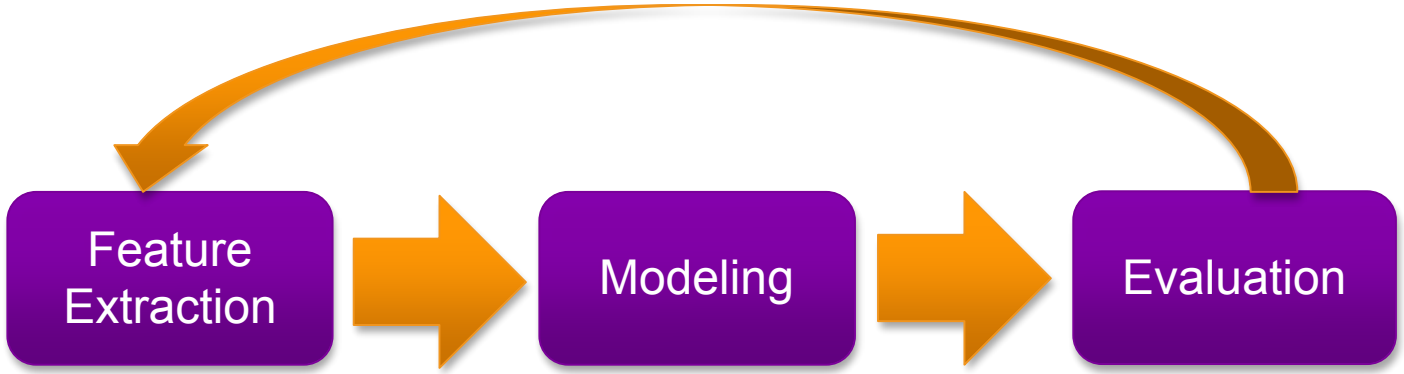
Joshua Rosen and Neoklis Polyzotis
U.C. Santa Cruz

Develop a platform for Machine Learning and Graph analytics on a cluster of shared-nothing machines

- Programming environment
  - Specific to ML and Graph domain
- Data-parallel runtime
  - Supports ML and Graph analytic workflows

Characteristic Workflow for ML Analytics

- Feature Extraction
  - ETL workflow
- Modeling
  - Iterative algorithm that fits a model to the data
- Evaluation
  - Checks model fidelity



What is a Model?

- Global Model
  - Aggregate summary of all data points
  - Small in size compared to data set
  - e.g., regression, classification

- Local Model
  - Interdependent parameters for each data point
  - Size proportional to the data set
  - e.g., topic (graphical) models, clustering, PageRank

Outline

- Case Study
- Modeling in the Cloud
- Our work
- Project roadmap

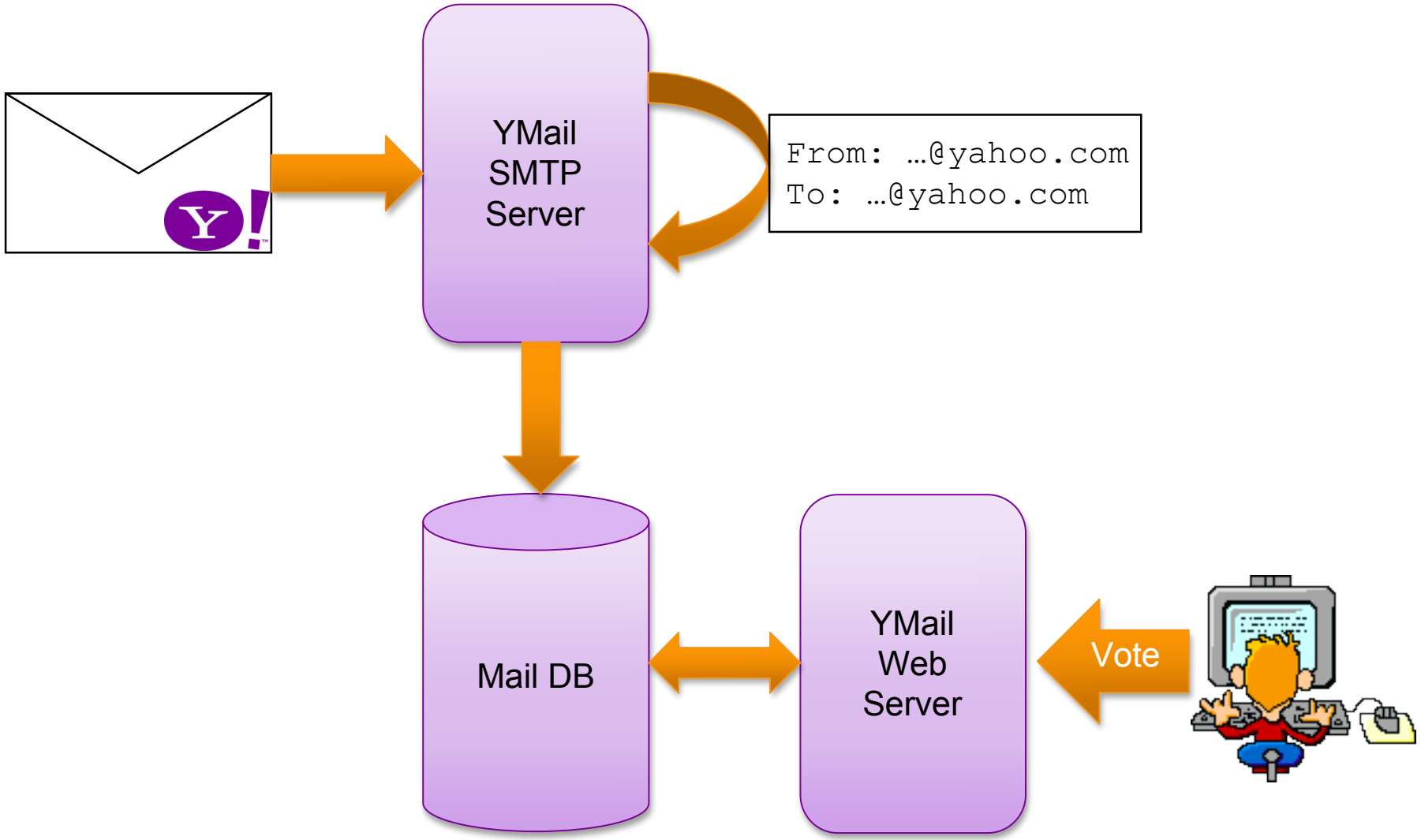
YAHOO! MAIL

Outbound Spam Filter for Yahoo! Mail

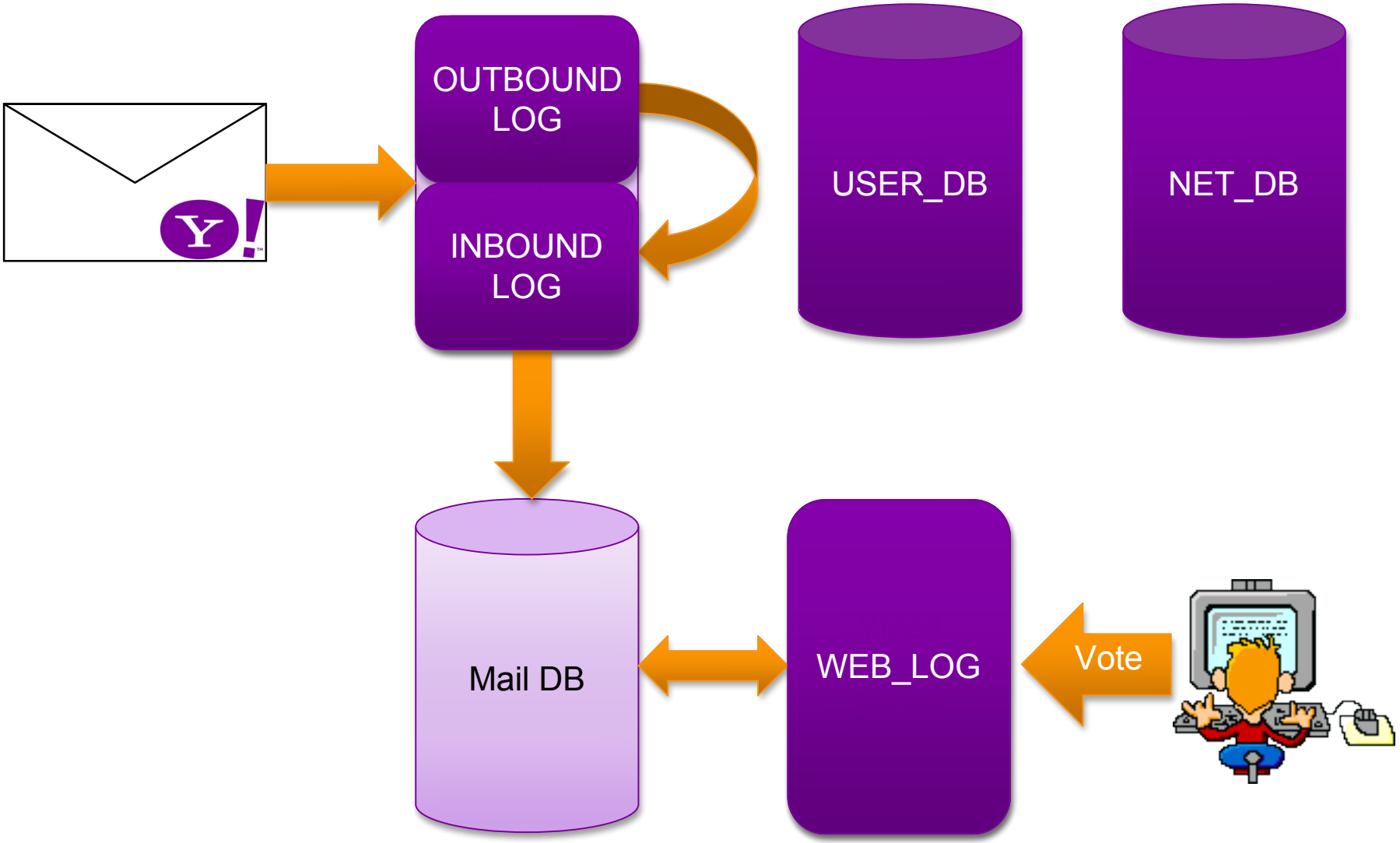
Ingredient: Client Classification

- **Goal:**
  - Train a spam classifier that can classify **outbound** messages
- **Problem:**
  - Other mail services won't tell us if our users send spam
- **Solution:**
  - Take the mail from Yahoo! to Yahoo! and observe how it was classified by our users

EMail Flow

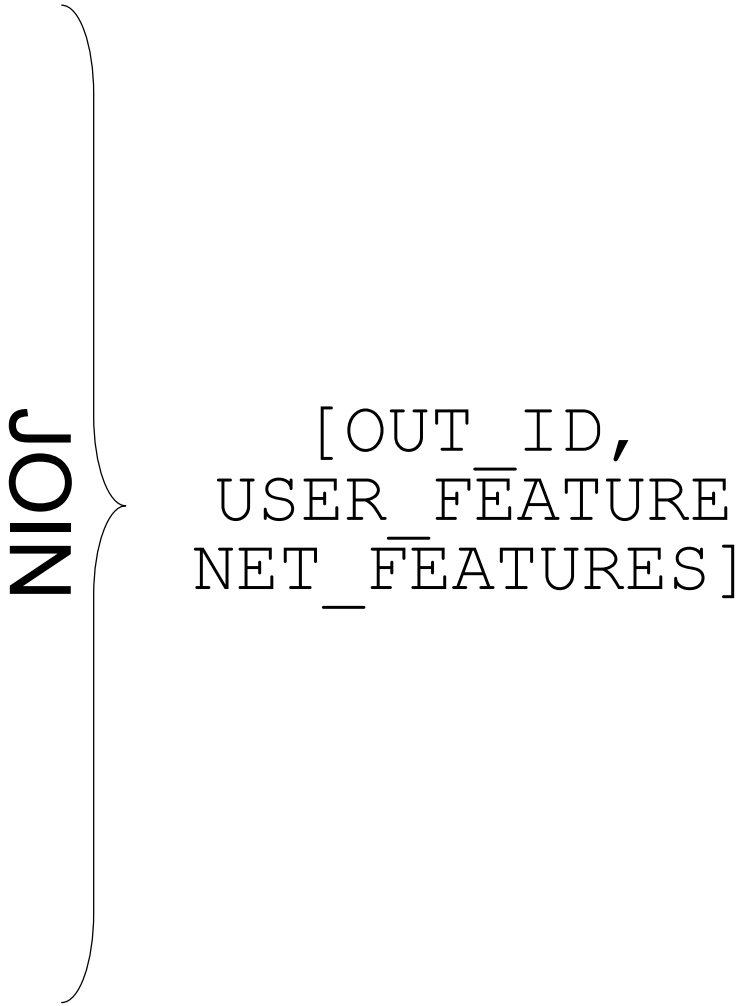

YMail
SMTP
Server
From: …@yahoo.com
To: …@yahoo.com
Mail DB
YMail
Web
Server
Vote

EMail Flow: Data Sets



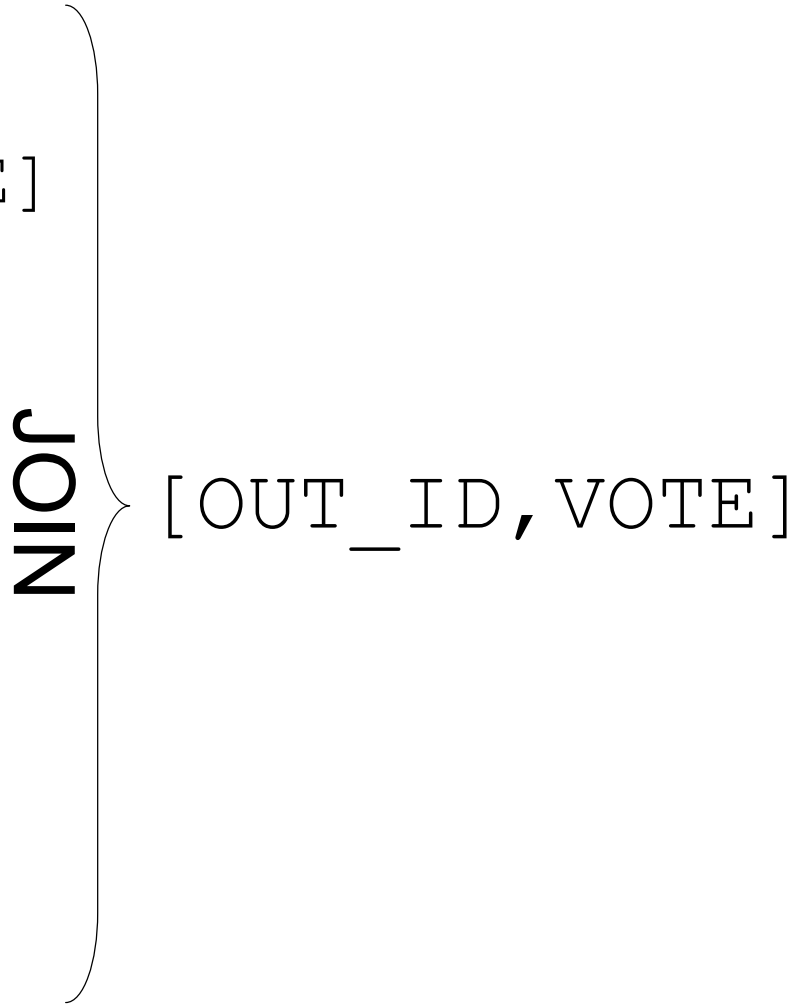
Step I Extract Features

- OUTBOUND_LOG
  - [OUT_ID, FROM, IP]
- USER_DB
  - [USER_ID, …]
- NET_DB
  - [IP, …]



Step II: Extract Labels

- OUTBOUND_LOG
  - [OUT_ID,FROM,TO,TIME]
- INBOUND_LOG
  - [IN_ID,FROM,TO,TIME]
- WEB_LOG
  - [IN_ID,VOTE]



Step II: Finding a Label

- OUTBOUND_LOG
  - [OUT_ID, **FROM, TO, TIME**]
- INBOUND_LOG
  - [IN_ID, **FROM, TO, TIME**]
- WEB_LOG
  - [IN_ID, VOTE]

JOIN

This needs a fuzzy join

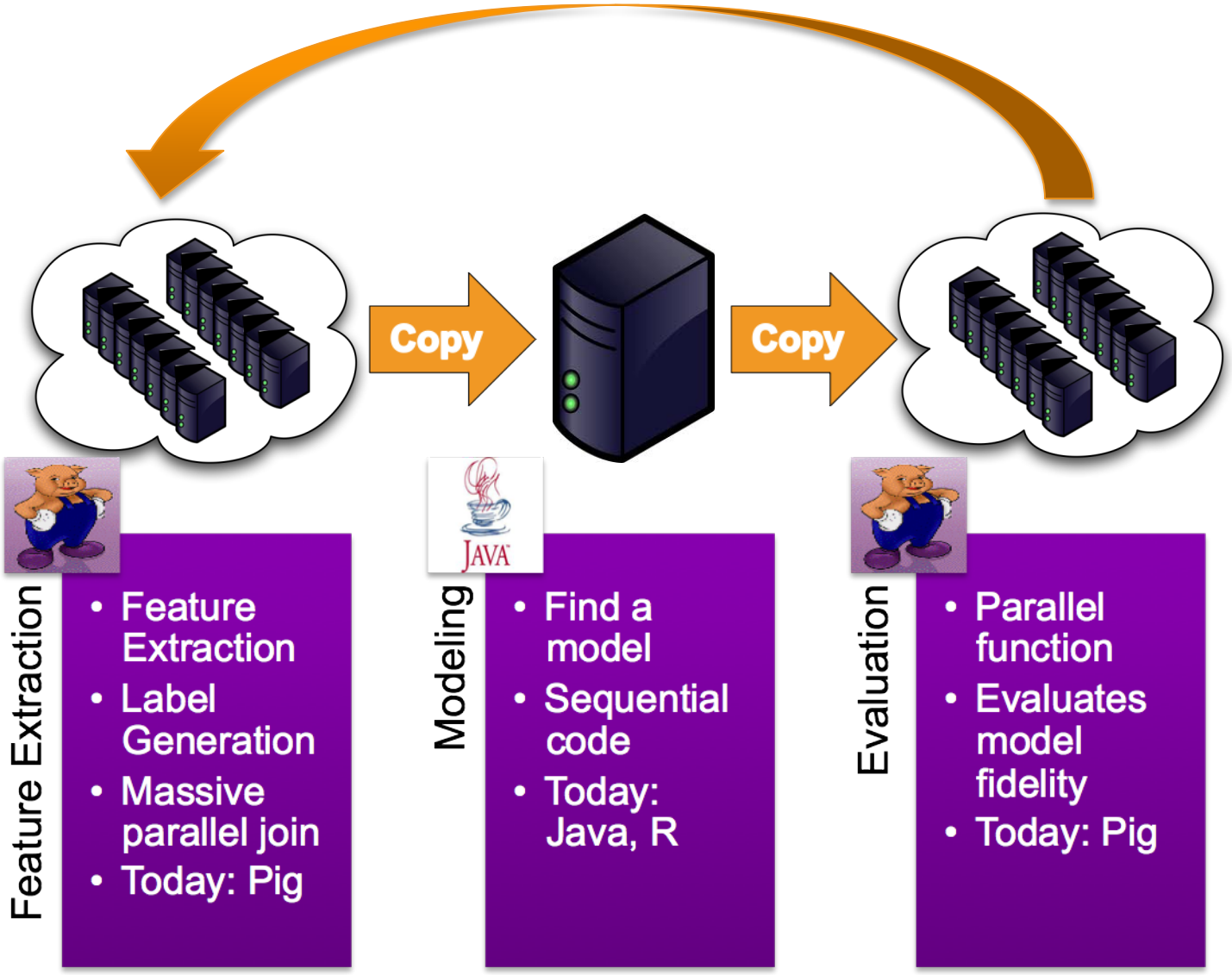
Step III: Creating the training data

- JOIN labels and features
- **Subsample** the training data
- Copy to a single machine

Step IV: Training the model

- Until a satisfactory model is found:
  - Train the model using sequential code
  - Evaluate the found model in Pig again
  - Apply insights to feature extraction, label generation and learning algorithm

Recap


Copy
Copy
Feature Extraction
• Feature Extraction
• Label Generation
• Massive parallel join
• Today: Pig
Modeling
• Find a model
• Sequential code
• Today: Java, R
Evaluation
• Parallel function
• Evaluates model fidelity
• Today: Pig

Take-Away

- **Usability is bad** (took an intern 3 months)
  - Many tools and technologies needed
  - Fractured workflow, captured in e.g. Oozie

- **Subpar solution**
  - Subsampling hurts classifier fidelity
  - Copying data to a single machine is slow

Now: Parallelize the middle

Feature Extraction

- Feature Extraction
- Label Generation
- Massive parallel join
- Today: Pig

Modeling

- Find a model
- Iterative data-parallel plan
- Cluster of machines
- Today: Hadoop

Evaluation

- Parallel function
- Evaluates model fidelity
- Today: Pig

What do people do?

- **Gradient Boosted Decision Trees**
  - Popular non-linear modeling algorithm
  - Used in Yahoo! search engine to learn the ranking function

- **Gradient Descent**
  - A method for minimizing objective functions
  - Written as a sum of differentiable functions

- **Latent Dirichlet Allocation (LDA)**
  - Example of a topic model
  - Usually expressed via Graphical Model

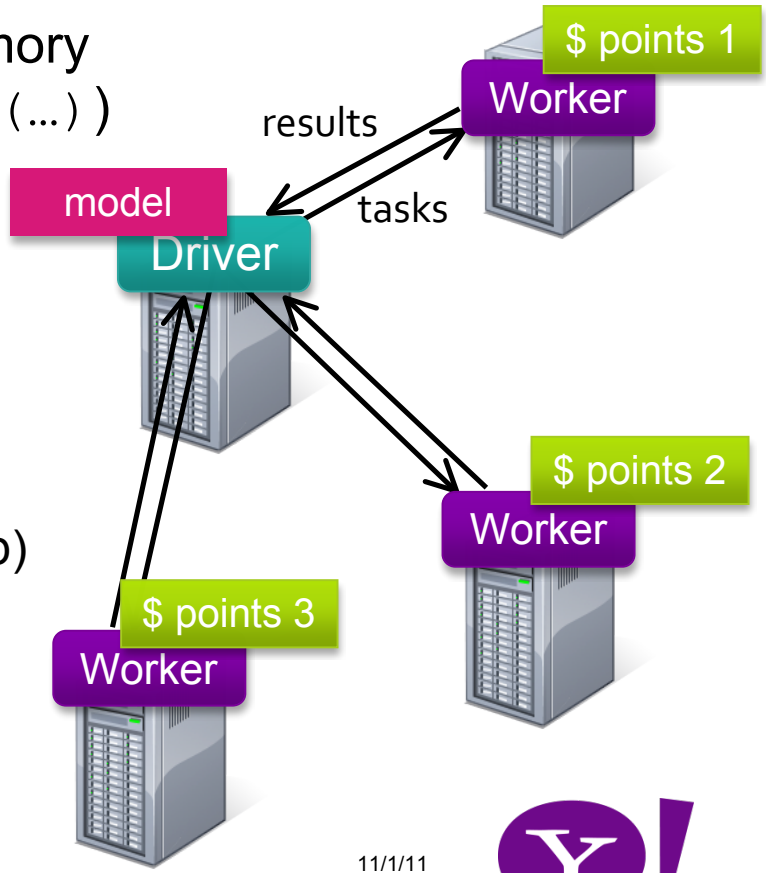
What do people do?

- **Gradient Boosted Decision Trees**
  - Popular non-linear modeling algorithm
  - Used in Yahoo! search engine to learn the ranking function
  - **Solution: Fake mappers that implement MPI**
- **Gradient Descent**
  - A method for minimizing objective functions
  - Written as a sum of differentiable functions
  - **Solution: Fake mappers that run aggregation trees**
- **Latent Dirichlet Allocation (LDA)**
  - Example of a topic model
  - Usually expressed via Graphical Model
  - **Solution: Fake mappers that implement this graphical model**

Framework I (Global Models): Spark

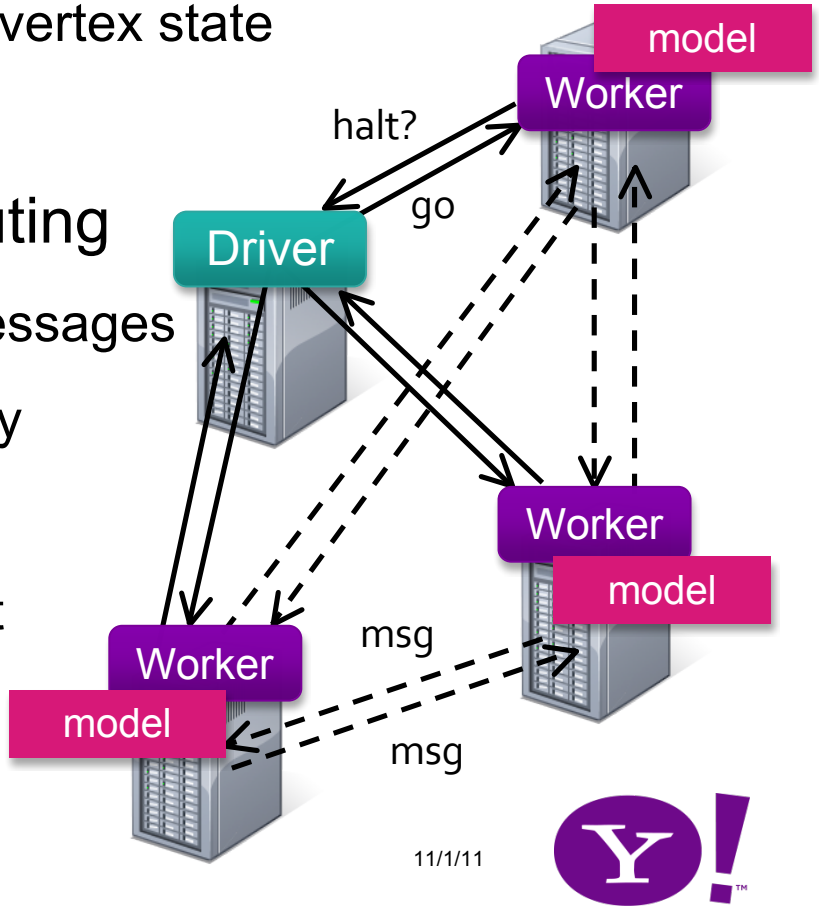
- Data Model: Resilient Distributed Datasets
  - Support for hash and range partitioning
  - Transformations or Actions can be added to them
  - The results can be materialized in memory (`RDD.cache()`) or to disk (`RDD.save(…)`)
- Caching allows fast iteration

```
points.cache()
var w = Vector.random(D)
for (i <- 1 to ITERATIONS) {
  val gradient = points.map { p => f(p, w) }
                       .reduce ((a, b) => a + b)
  w -= gradient
}
```



Framework II (Local Models): Pregel

- Graph-oriented API
  - UDF that implements single vertex update
  - Input messages from neighboring vertices
  - Output: outgoing messages and new vertex state
- Bulk synchronous parallel computing
  - Runtime executes all vertices with messages
  - Model updated and cached in memory
  - Checkpoint for fault-tolerance
  - Continues until everyone votes to halt

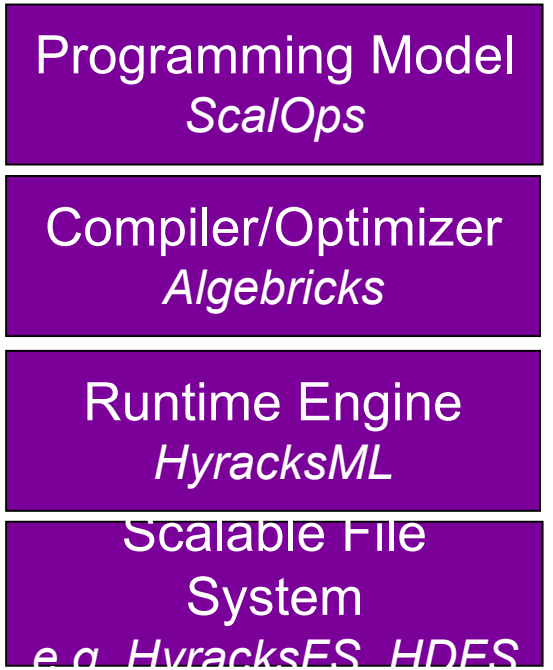


Framework: Pros/Cons

- **Pros**
  - Adds the notion of a working set for fast iterations
  - Exports a clean API specific to the problem domain
  - Abstracts (many) low-level implementation details

- **Cons**
  - Point solutions that mainly focus on modeling
  - Limited optimization support
  - No external memory operators

ScalOps + HyracksML



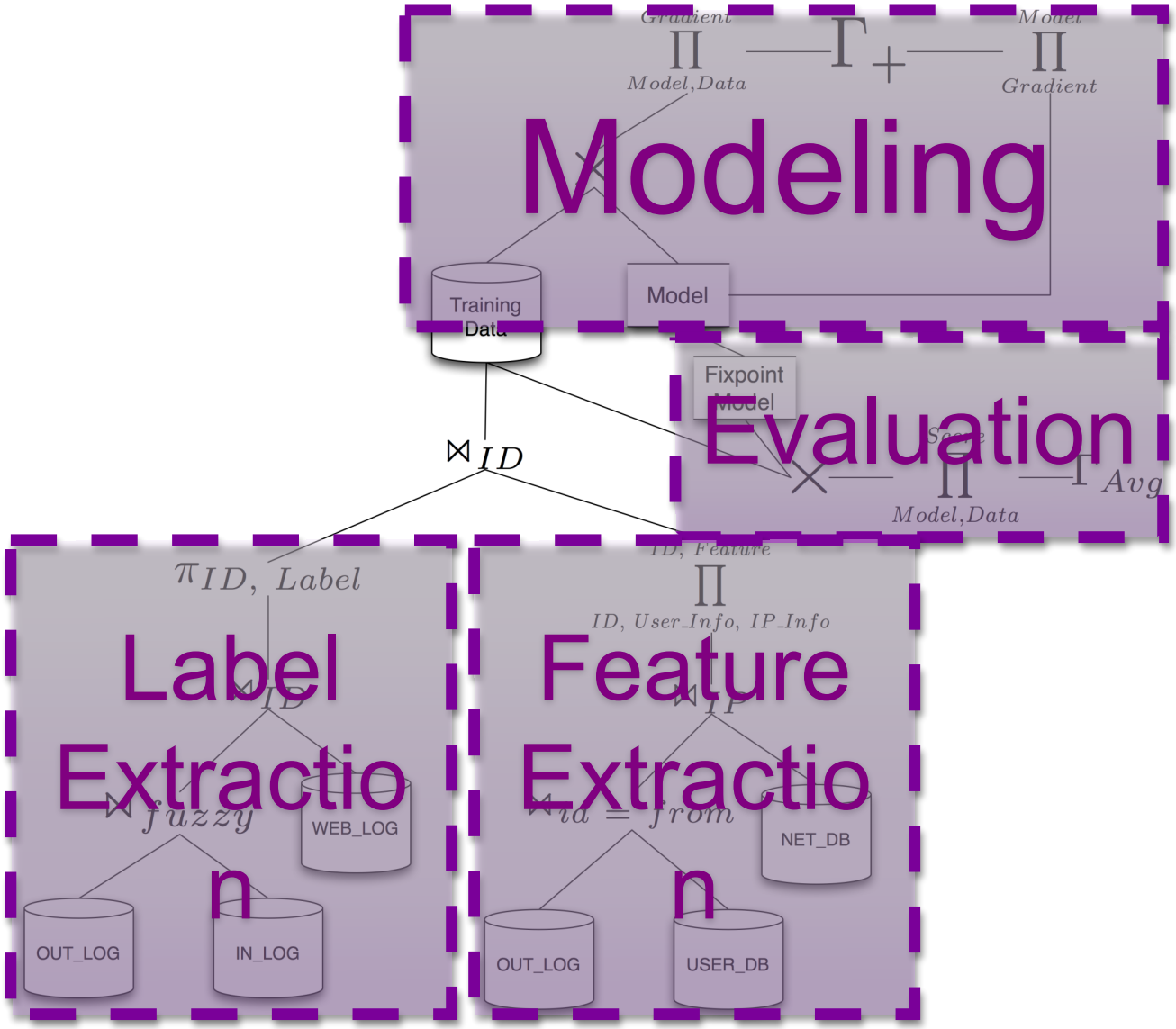
- ScalOps
  - Scala DSL for the entire analytic pipeline
  - Supports Pig Latin
  - Looping construct that captures iteration

- HyracksML + Algebricks
  - Data-parallel runtime that runs on a cluster of shared-nothing machines
  - Deductive database extensions to directly support iteration
  - Relational algebra and query optimizer that captures the entire workflow

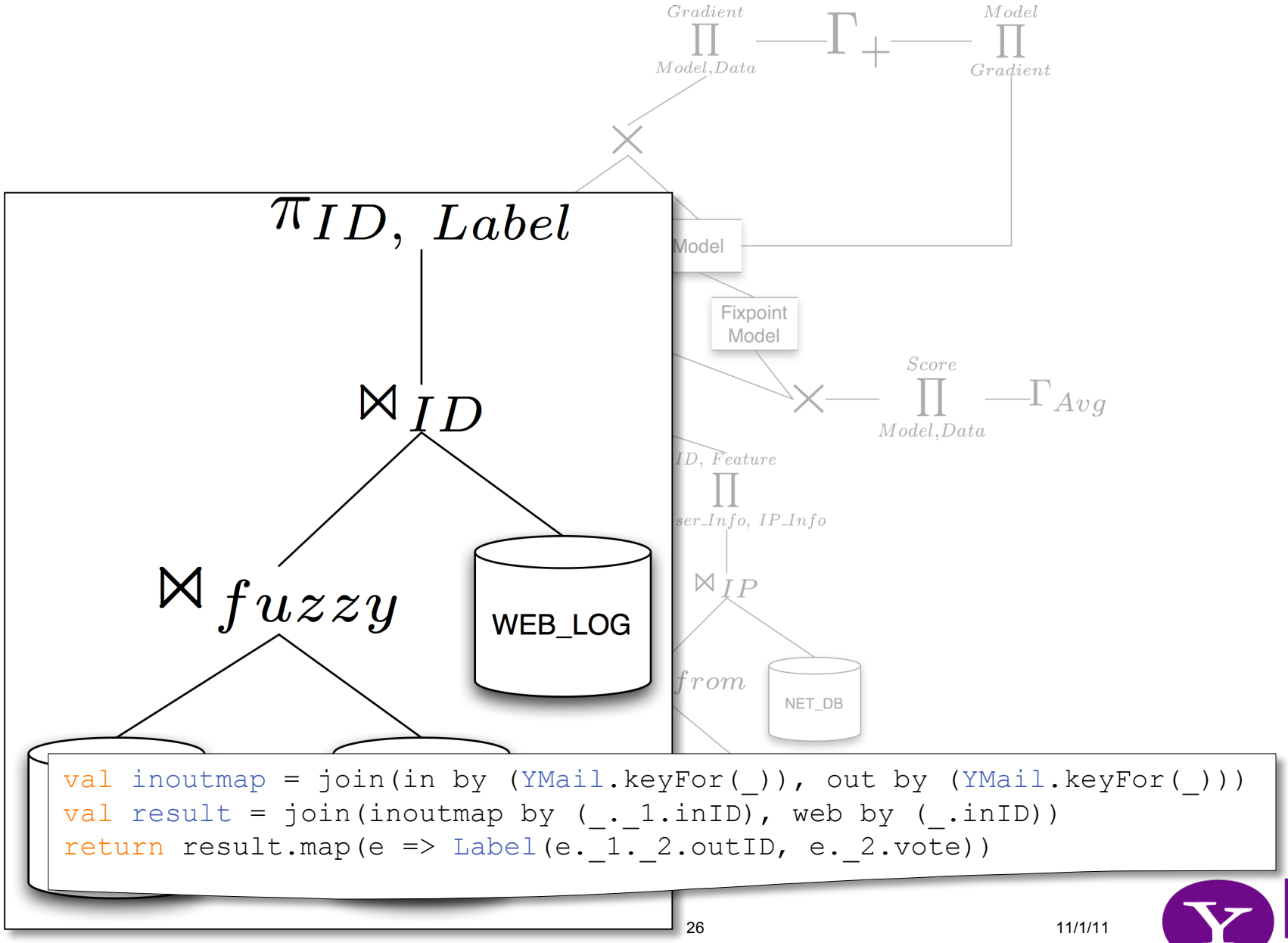
Spam Filter: ScalOps

```
object OutboundSpamFilter extends Query {

  // Join Targets
  case class Features(outID: String, features: Vector)
  case class Label(outID: String, label: Double)

  def getLabels(inboundLogs: Queryable[YMail.InboundRecord], outboundLogs: Queryable[YMail.OutboundRecord], webLogs: Queryable[YMail.WebRecord]) =
{
    val inoutmap = join(inboundLogs by (YMail.keyFor(_)), outboundLogs by (YMail.keyFor(_)))
    join(inoutmap by (_._1.inID), webLogs by (_.inID)).map(e => Label(e._1._2.outID, e._2.vote))
  }

  def getFeatures(outboundLogs: Queryable[YMail.OutboundRecord], userDB: Queryable[YMail.User], netDB: Queryable[YMail.IP]) = {
    case class IPInformation(outID: String, ipInfo: YMail.IP)
    case class UserInformation(outID: String, userInfo: YMail.User)
    val users = join(outboundLogs by (_.from), userDB by (_.id)).map(e => UserInformation(e._1.outID, e._2))
    val ips = join(outboundLogs by (_.ip), netDB by (_.ip)).map(e => IPInformation(e._1.outID, e._2))
    join(outboundLogs by (_.outID), users by (_.outID), ips by (_.outID)).map(e => Features(e._1.outID, YMail.extractFeatures(e._1, e._2.userInfo,
e._3.ipInfo)))
  }

  def run {
    val inboundLogs = load[YMail.InboundRecord]("hdfs://ymail/inbound.dat").filter(YMail.isYMail)
    val outboundLogs = load[YMail.OutboundRecord]("hdfs://ymail/outbound.dat").filter(YMail.isYMail)
    val webLogs = load[YMail.WebRecord]("hdfs://ymail/web.dat")
    val userDB = load[YMail.User]("hdfs://y/users.dat")
    val netDB = load[YMail.IP]("hdfs://y/network.dat")

    // Extract labels
    val labels = getLabels(inboundLogs, outboundLogs, webLogs)

    // Extract Features
    val features = getFeatures(outboundLogs, userDB, netDB)

    // Assemble a dataset
    val dataset = join(labels by (_.outID), features by (_.outID)).map(e => new Example(e._1.outID, e._1.label, e._2.features))

    // Train a model
    val model = ML.train(dataset)

    // Evaluate
    val score = dataset.map(ML.evaluate(model, _)).reduce(_ + _) / dataset.count

  }

}
```


Spam Filter: Hyracks RQ

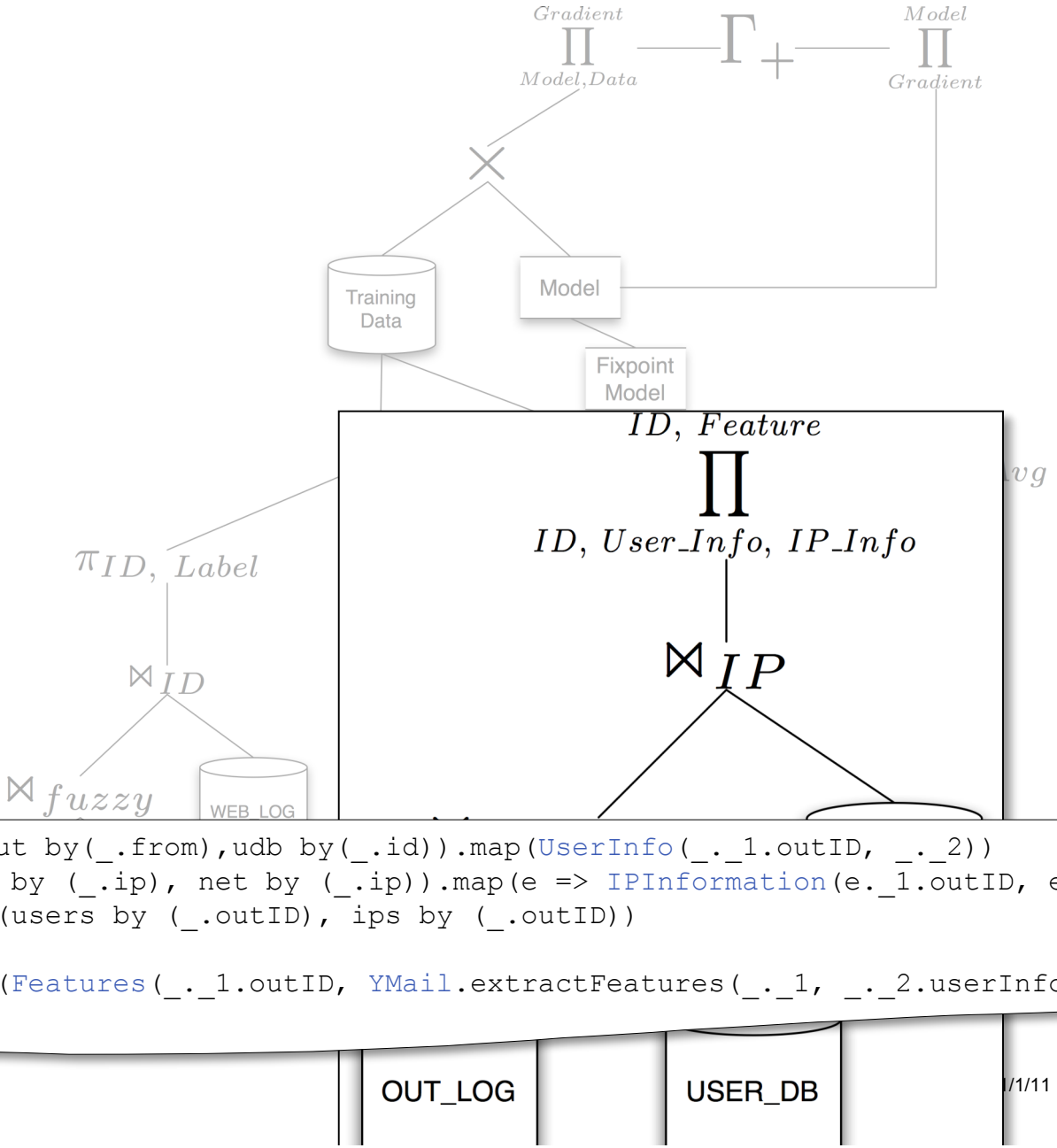


Spam Filter: Label Extraction



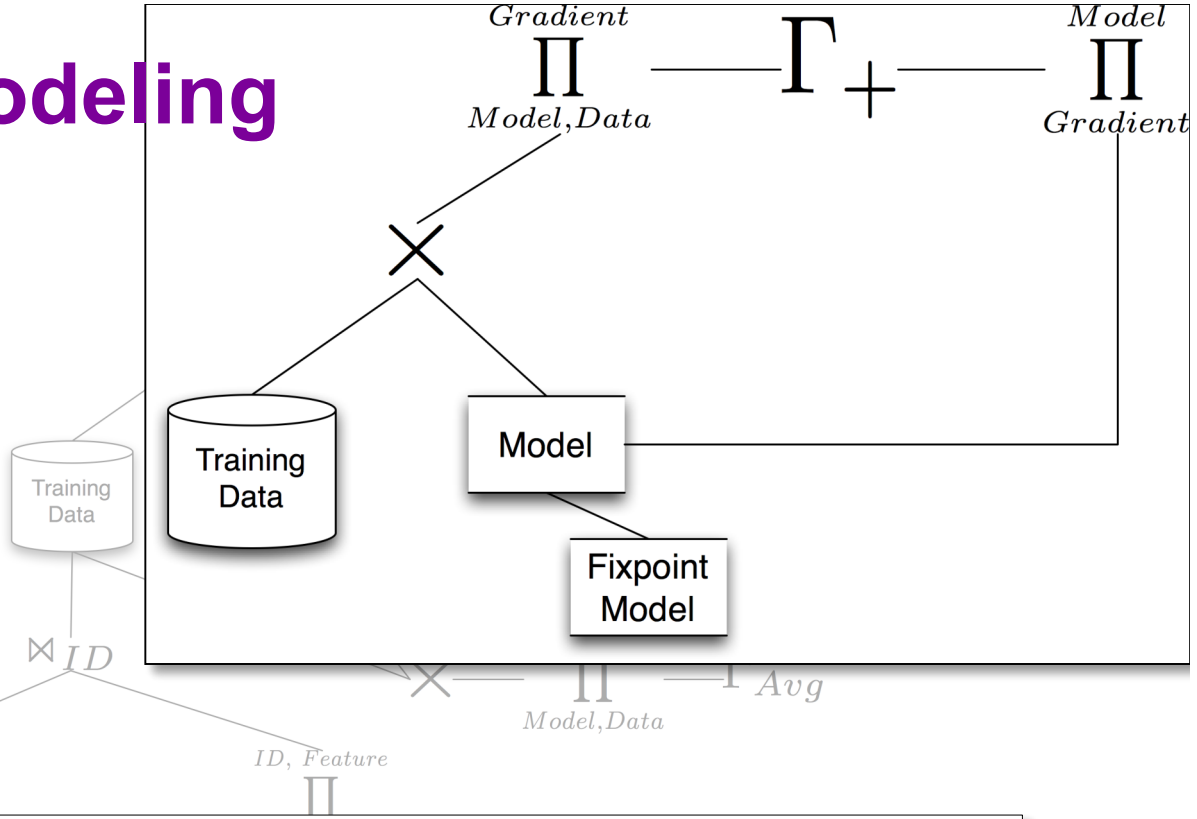

```
val inoutmap = join(in by (YMail.keyFor(_)), out by (YMail.keyFor(_)))
val result = join(inoutmap by (_._1.inID), web by (_.inID))
return result.map(e => Label(e._1._2.outID, e._2.vote))
```

Spam Filter: Feature Extraction



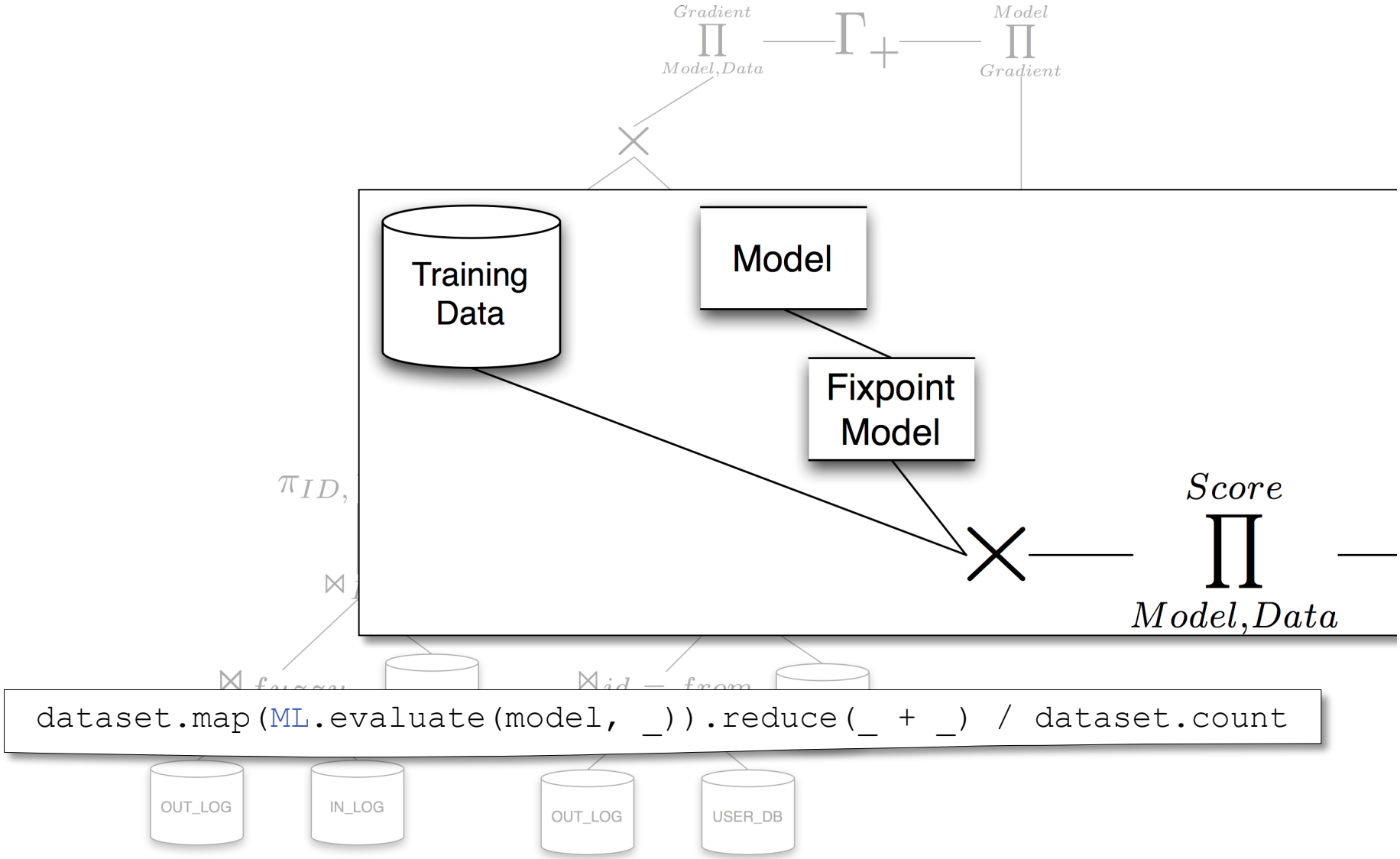

```
val users = join(out by(_.from),udb by(_.id)).map(UserInfo(_._1.outID, _._2))
val ips = join(out by (_.ip), net by (_.ip)).map(e => IPInformation(e._1.outID, e._2))
val results = join(users by (_.outID), ips by (_.outID))

return results.map(Features(_._1.outID, YMail.extractFeatures(_._1, _._2.userInfo, _._3.ipInfo)))
```

Spam Filter: Modeling




```
class MyModel(w: VectorType, funcValue: DoubleType) extends Model
val init = new MyModel(VectorType.zeros(1000), Double.MaxValue)
val result = loop(init, (mdl: Model) => mdl.funcValue > eps) { mdl =>
  val gradients = data.map(x => computeGradient(x, mdl))
  mdl.funcValue = gradients.map(x => x._1).reduce(_+_)
  mdl.funcValue += lambda * env.w.norm2
  mdl.w -= gradients.map(x => x._2).reduce(_+_) / gradients.count
  mdl.w *= (1.0 - 0.5 * lambda)
  mdl
}
return result
```

Spam Filter: Evaluation



Conclusion: Target Algorithms

- **Linear Models**
  - Batch Gradient Descent
  - Stochastic Gradient Descent
  - Logistic Regression
- **Clustering**
  - K-Means
- **Matrix Factorization**
- **Graph Analysis**
  - PageRank
  - LDA (Graphical Models)

Coming Attractions

- Recursive data-parallel runtime (HyracksML)
  - Deductive Databases, BOOM, Flying Fixed-Point

   Yingyi Bu and Joshua Rosen
- Extended Relational Algebra (Algebricks)
  - Captures the entire workflow in an optimizer friendly representation

   Vinayak Borkar
- High-level language (ScalOps)
  - Abstractions for Machine learning and Graph-based algorithms

   Markus Weimer

- Executive Producers

Mike Carey, Tyson Condie, Neoklis Polyzotis and Raghu Ramakrishnan