Declarative Systems for Large Scale Machine Learning

Markus Weimer, Tyson Condie, Raghu Ramakrishnan

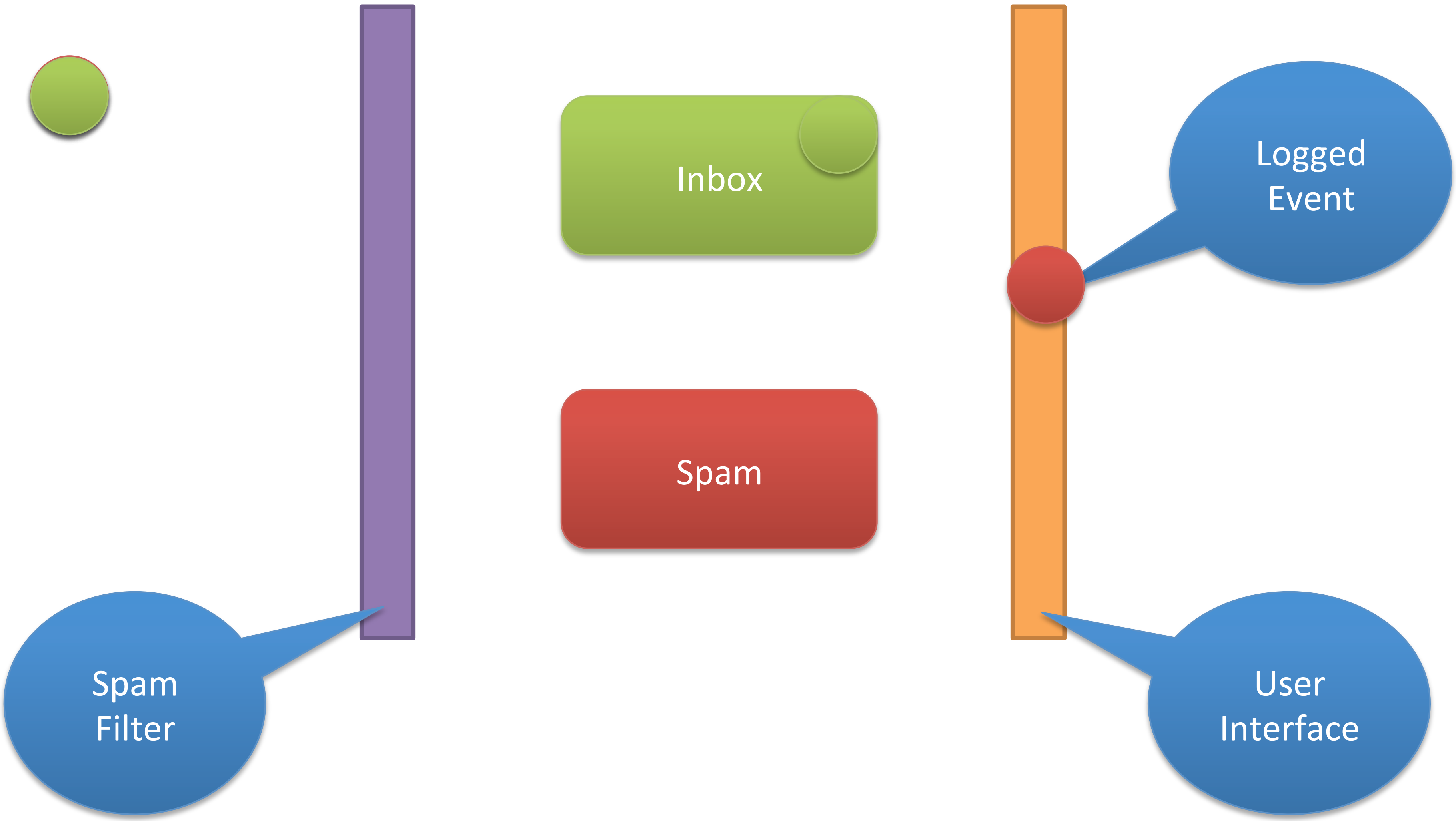
Cloud and Information Services Laboratory
Microsoft

Joint work with …

Yingyi Bu, Vinayak Borkar, Michael J. Carey
University of California, Irvine

Joshua Rosen, Neoklis Polyzotis
University of California, Santa Cruz

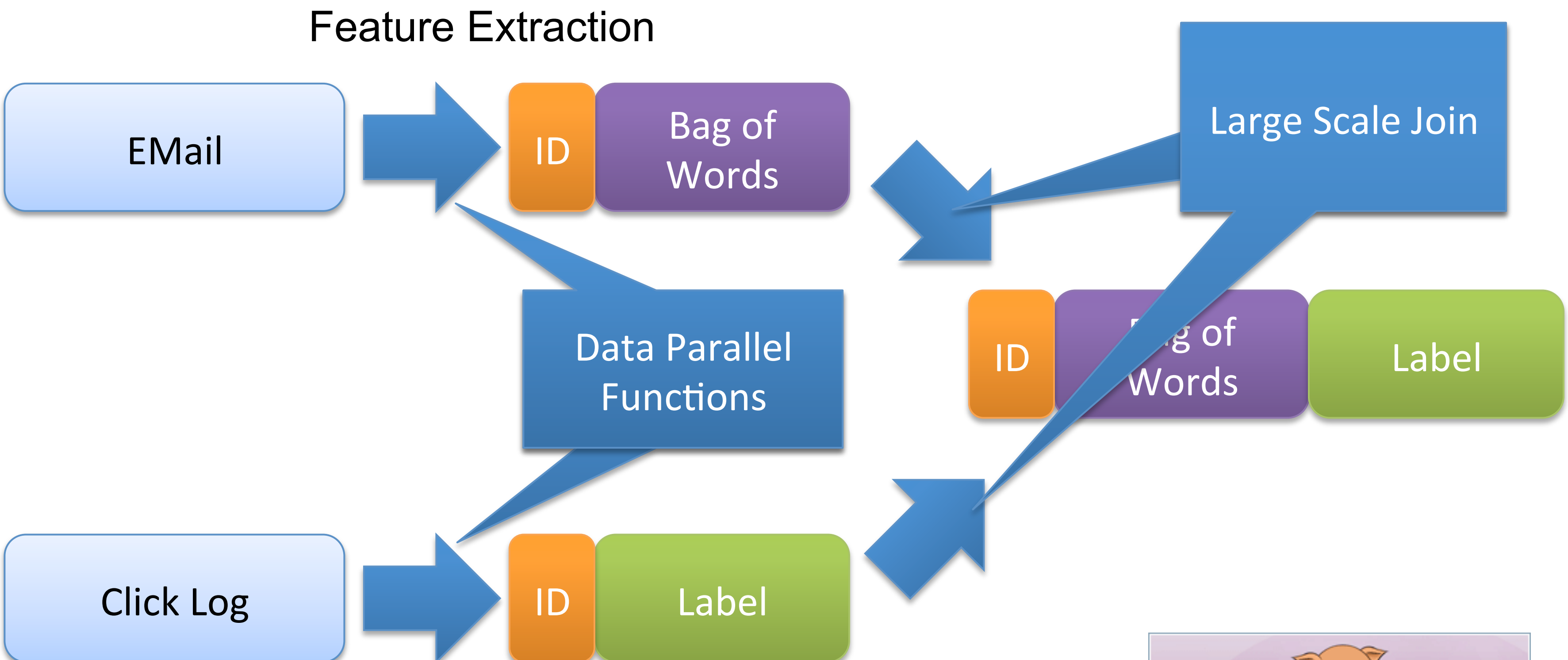
Example: Spam Filter



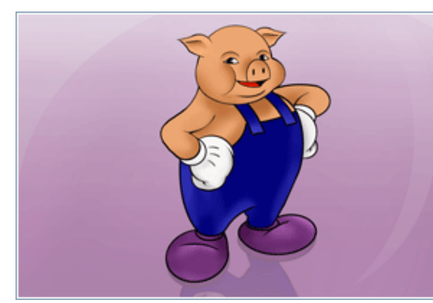
Machine Learning Workflow

- **Step I: Example Formation**
  - Feature Extraction
  - Label Extraction
- **Step II: Modeling**
- **Step III: Deployment (or just Evaluation)**

Example Formation



Label Extraction



Modeling

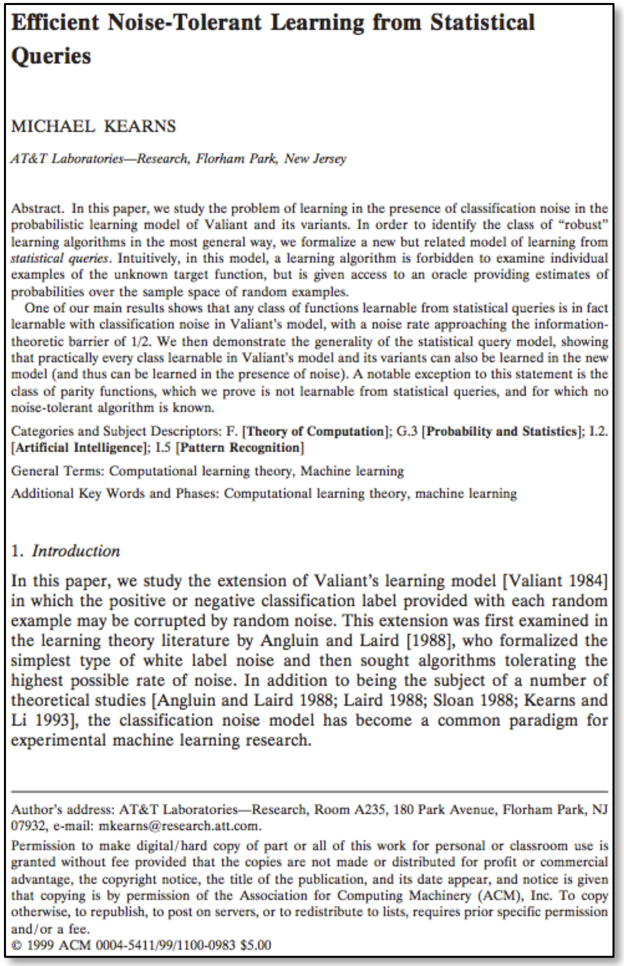
- **Many Algorithms are inherently sequential**
  - Apply model to data → Look at Errors → Update Model

- **Common solutions**
  - Subsampling
  - Train on partitions, merge results
  - Rephrasing of algorithms in MapReduce

MapReduce for Modeling

- Learning algorithm access the data only through **statistical querys**

- A statistical query returns an estimate of the expectation of a function f(x,y) applied to the data.



Efficient Noise-Tolerant Learning from Statistical Queries

MICHAEL KEARNS

AT&T Laboratories—Research, Florham Park, New Jersey

Abstract. In this paper, we study the problem of learning in the presence of classification noise in the probabilistic learning model of Valiant and its variants. In order to identify the class of "robust" learning algorithms in the most general way, we formalize a new but related model of learning from *statistical queries*. Intuitively, in this model, a learning algorithm is forbidden to examine individual examples of the unknown target function, but is given access to an oracle providing estimates of probabilities over the sample space of random examples.

One of our main results shows that any class of functions learnable from statistical queries is in fact learnable with classification noise in Valiant's model, with a noise rate approaching the information-theoretic barrier of 1/2. We then demonstrate the generality of the statistical query model, showing that practically every class learnable in Valiant's model and its variants can also be learned in the new model (and thus can be learned in the presence of noise). A notable exception to this statement is the class of parity functions, which we prove is not learnable from statistical queries, and for which no noise-tolerant algorithm is known.

Categories and Subject Descriptors: F. [**Theory of Computation**]; G.3 [**Probability and Statistics**]; I.2. [**Artificial Intelligence**]; I.5 [**Pattern Recognition**]

General Terms: Computational learning theory, Machine learning

Additional Key Words and Phases: Computational learning theory, machine learning

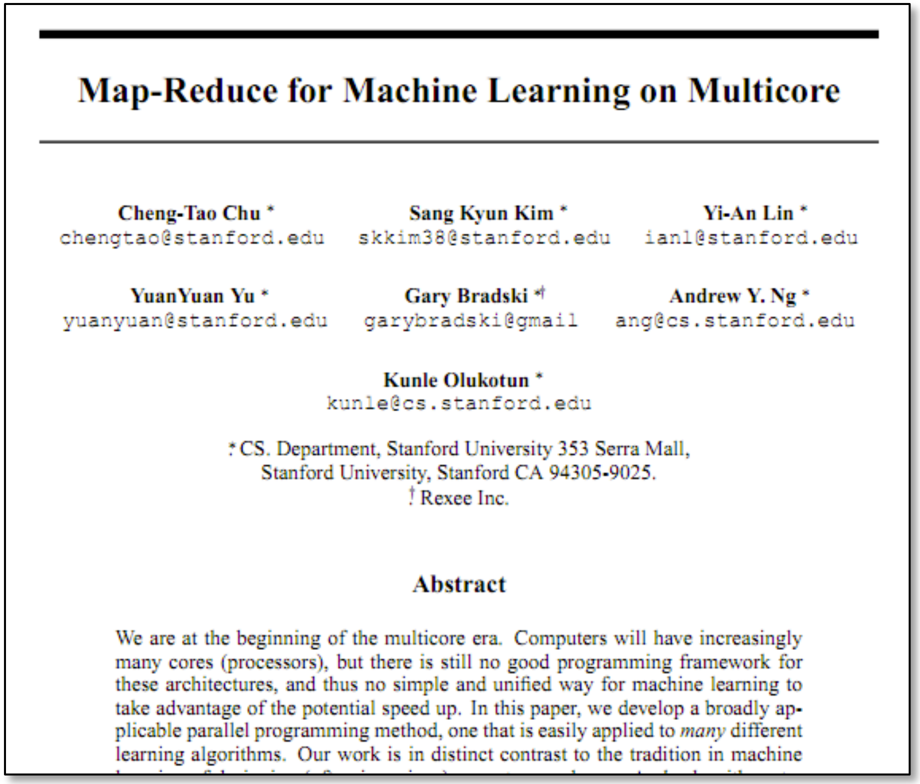
1. *Introduction*

In this paper, we study the extension of Valiant's learning model [Valiant 1984] in which the positive or negative classification label provided with each random example may be corrupted by random noise. This extension was first examined in the learning theory literature by Angluin and Laird [1988], who formalized the simplest type of white label noise and then sought algorithms tolerating the highest possible rate of noise. In addition to being the subject of a number of theoretical studies [Angluin and Laird 1988; Laird 1988; Sloan 1988; Kearns and Li 1993], the classification noise model has become a common paradigm for experimental machine learning research.

Author's address: AT&T Laboratories—Research, Room A235, 180 Park Avenue, Florham Park, NJ 07932, e-mail: mkearns@research.att.com.

MapReduce for Modeling

- Rephrase query in summation form.
- **Map:** Calculate function estimates over data partitions
- **Reduce:** Aggregate the function estimates.



Map-Reduce for Machine Learning on Multicore

Cheng-Tao Chu * chengtao@stanford.edu

Sang Kyun Kim * skkim38@stanford.edu

Yi-An Lin * ianl@stanford.edu

YuanYuan Yu * yuanyuan@stanford.edu

Gary Bradski *† garybradski@gmail

Andrew Y. Ng * ang@cs.stanford.edu

Kunle Olukotun * kunle@cs.stanford.edu

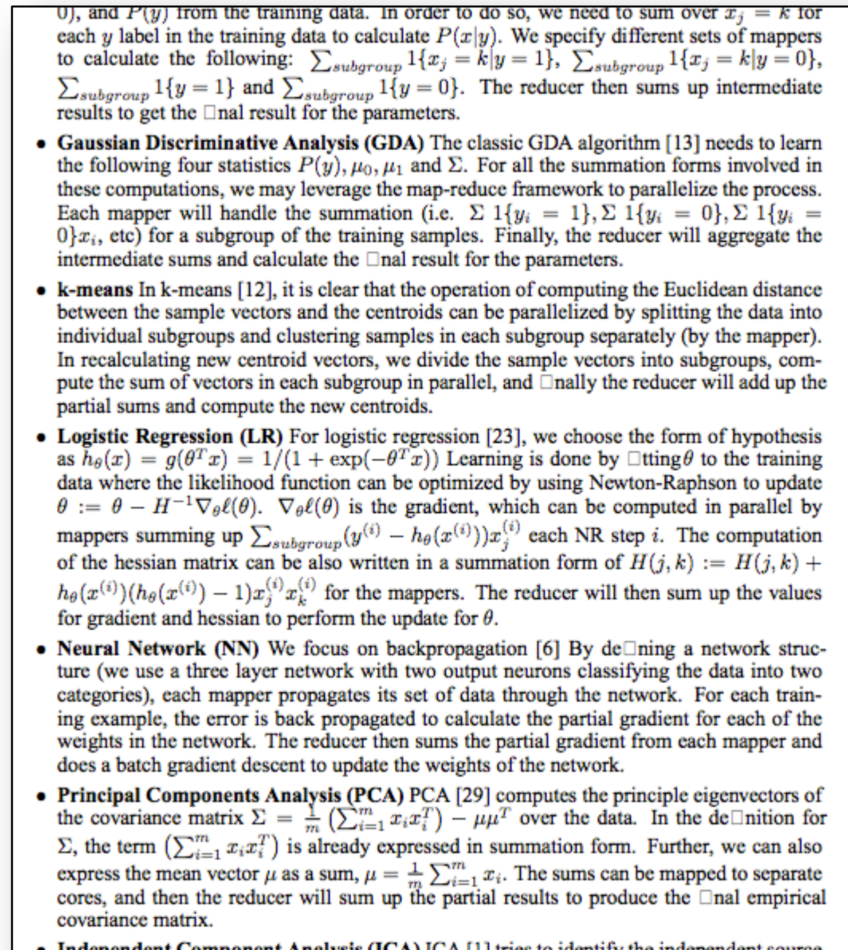
* CS. Department, Stanford University 353 Serra Mall,
Stanford University, Stanford CA 94305-9025.
† Rexee Inc.

Abstract

We are at the beginning of the multicore era. Computers will have increasingly many cores (processors), but there is still no good programming framework for these architectures, and thus no simple and unified way for machine learning to take advantage of the potential speed up. In this paper, we develop a broadly applicable parallel programming method, one that is easily applied to *many* different learning algorithms. Our work is in distinct contrast to the tradition in machine

Example Methods

- Convex Optimization
  - (Logistic) Regression
  - Support Vector machines
  - …
- K-Means Clustering
- Naïve Bayes
- Neural Networks
- …



0), and $P(y)$ from the training data. In order to do so, we need to sum over $x_j = k$ for each y label in the training data to calculate $P(x|y)$. We specify different sets of mappers to calculate the following: $\sum_{subgroup} 1\{x_j = k|y = 1\}$, $\sum_{subgroup} 1\{x_j = k|y = 0\}$, $\sum_{subgroup} 1\{y = 1\}$ and $\sum_{subgroup} 1\{y = 0\}$. The reducer then sums up intermediate results to get the □nal result for the parameters.

- **Gaussian Discriminative Analysis (GDA)** The classic GDA algorithm [13] needs to learn the following four statistics $P(y), \mu_0, \mu_1$ and Σ. For all the summation forms involved in these computations, we may leverage the map-reduce framework to parallelize the process. Each mapper will handle the summation (i.e. $\Sigma\ 1\{y_i = 1\}, \Sigma\ 1\{y_i = 0\}, \Sigma\ 1\{y_i = 0\}x_i$, etc) for a subgroup of the training samples. Finally, the reducer will aggregate the intermediate sums and calculate the □nal result for the parameters.
- **k-means** In k-means [12], it is clear that the operation of computing the Euclidean distance between the sample vectors and the centroids can be parallelized by splitting the data into individual subgroups and clustering samples in each subgroup separately (by the mapper). In recalculating new centroid vectors, we divide the sample vectors into subgroups, compute the sum of vectors in each subgroup in parallel, and □nally the reducer will add up the partial sums and compute the new centroids.
- **Logistic Regression (LR)** For logistic regression [23], we choose the form of hypothesis as $h_\theta(x) = g(\theta^T x) = 1/(1 + \exp(-\theta^T x))$ Learning is done by □ttingθ to the training data where the likelihood function can be optimized by using Newton-Raphson to update $\theta := \theta - H^{-1}\nabla_\theta \ell(\theta)$. $\nabla_\theta \ell(\theta)$ is the gradient, which can be computed in parallel by mappers summing up $\sum_{subgroup}(y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)}$ each NR step i. The computation of the hessian matrix can be also written in a summation form of $H(j,k) := H(j,k) + h_\theta(x^{(i)})(h_\theta(x^{(i)}) - 1)x_j^{(i)}x_k^{(i)}$ for the mappers. The reducer will then sum up the values for gradient and hessian to perform the update for θ.
- **Neural Network (NN)** We focus on backpropagation [6] By de□ning a network structure (we use a three layer network with two output neurons classifying the data into two categories), each mapper propagates its set of data through the network. For each training example, the error is back propagated to calculate the partial gradient for each of the weights in the network. The reducer then sums the partial gradient from each mapper and does a batch gradient descent to update the weights of the network.
- **Principal Components Analysis (PCA)** PCA [29] computes the principle eigenvectors of the covariance matrix $\Sigma = \frac{1}{m}\left(\sum_{i=1}^m x_i x_i^T\right) - \mu\mu^T$ over the data. In the de□nition for Σ, the term $\left(\sum_{i=1}^m x_i x_i^T\right)$ is already expressed in summation form. Further, we can also express the mean vector μ as a sum, $\mu = \frac{1}{m}\sum_{i=1}^m x_i$. The sums can be mapped to separate cores, and then the reducer will sum up the partial results to produce the □nal empirical covariance matrix.
- **Independent Component Analysis (ICA)** ICA [1] tries to identify the independent source

Example: Batch Gradient Descent (BGD)

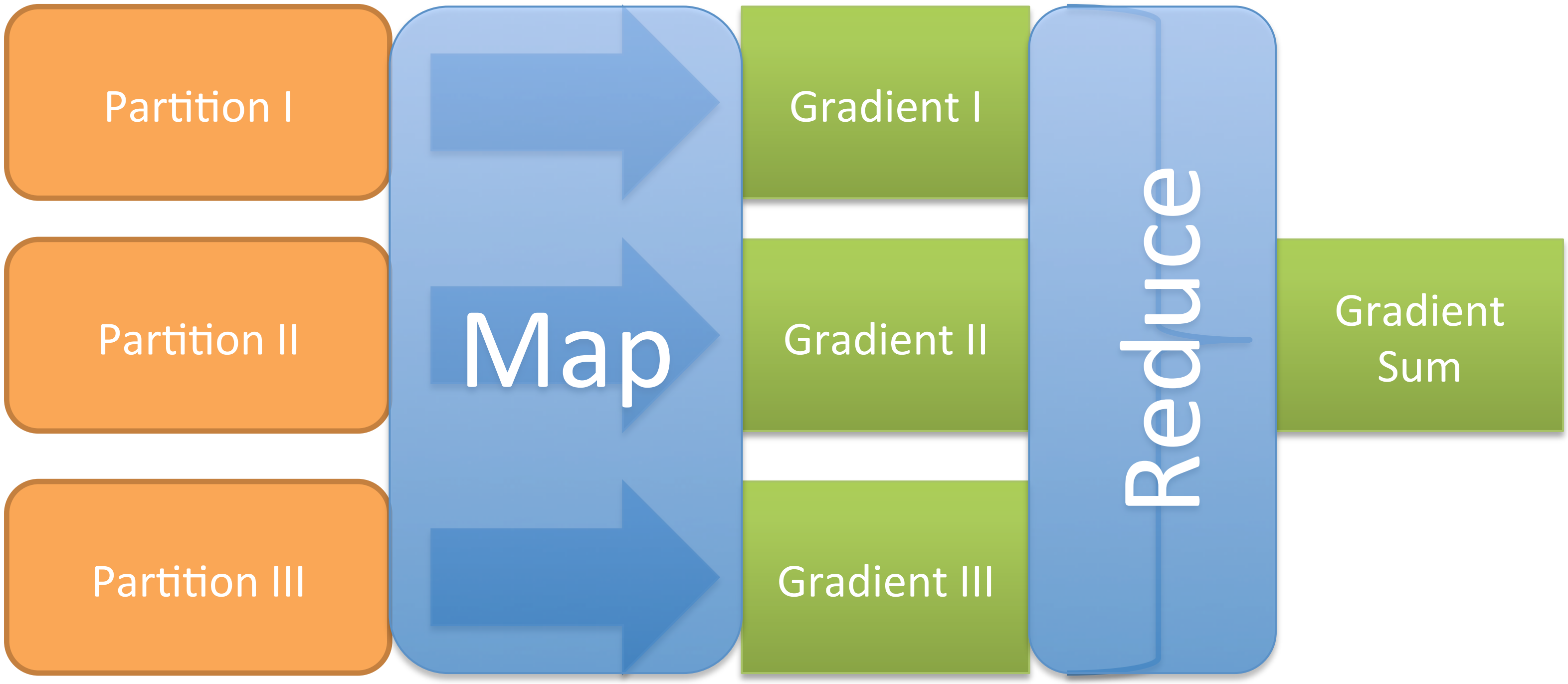
Until Convergence:

$$w_{t+1} = \underbrace{(1.0 - \eta\lambda) * \Big(w_t}_{\text{Regularization}} - \eta \underbrace{\sum_{(x,y)} \partial_w l\big(y, \langle w_t, x \rangle\big)}_{\text{Data Parallel Sum}} \Big)$$

w_t: Current Model
x: Data
y: Label

l: loss function (e.g. squared error)
∂: Gradient operator

Example: Gradient Computation



Modeling on Hadoop MapReduce?

- **Machine learning algorithms are iterative**
  - Each iteration contains multiple Statistical Queries

- **Overhead per MapReduce Job**
  - Each statistical query is a job
  - A job entails Scheduling, Data reading, State transfer, …
  - Especially bad on shared clusters

More than Map Reduce

- **Complete Job DAGs**
  - Beyond the fixed map-groupby-reduce
  - Arbitrary length and complexity
- **More Operators**
  - Join, Filter, Project, …
- **Examples**
  - Dryad (Microsoft Research)
  - Hyracks (UC Irvine)
  - Stratosphere (TU Berlin)

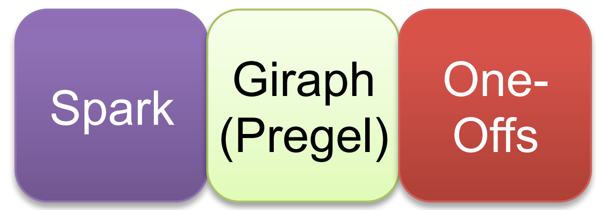
More than Map Reduce

- **Complete Job DAGs**
  - Beyond the fixed map-groupby-reduce
  - Arbitrary length and complexity
- **More Operators**
  - Join, Filter, Project, …
- **Examples**
  - Dryad (Microsoft Research)
  - Hyracks (UC Irvine)
  - Stratosphere (TU Berlin)

Machine Learning is Cyclic!

Applied Large Scale ML requires …

- **A Relational Algebra**
  - Join, Filter, Map, …
  - For feature and label extraction
- **Iterative computation**
  - Loops over data
  - Incremental model updates



- **Scalability / High Performance**
  - Jobs must execute successfully irrespective of the data set size / runtime cluster configuration
  - More favorable cluster setups must be used for speed-ups (e.g. cache data in memory)

Take-away

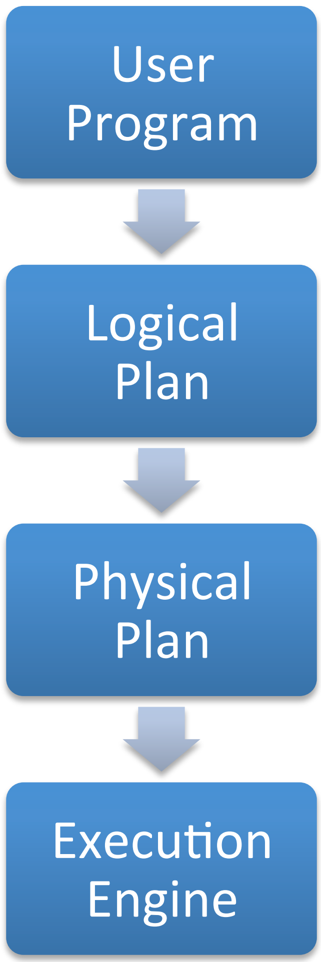
- Usability is bad
  - Developing a single model takes months
  - Requires many tools and technologies

- Pick your poison on a way to a subpar solution
  - Subsampling hurts model fidelity
  - Training on MapReduce often too slow

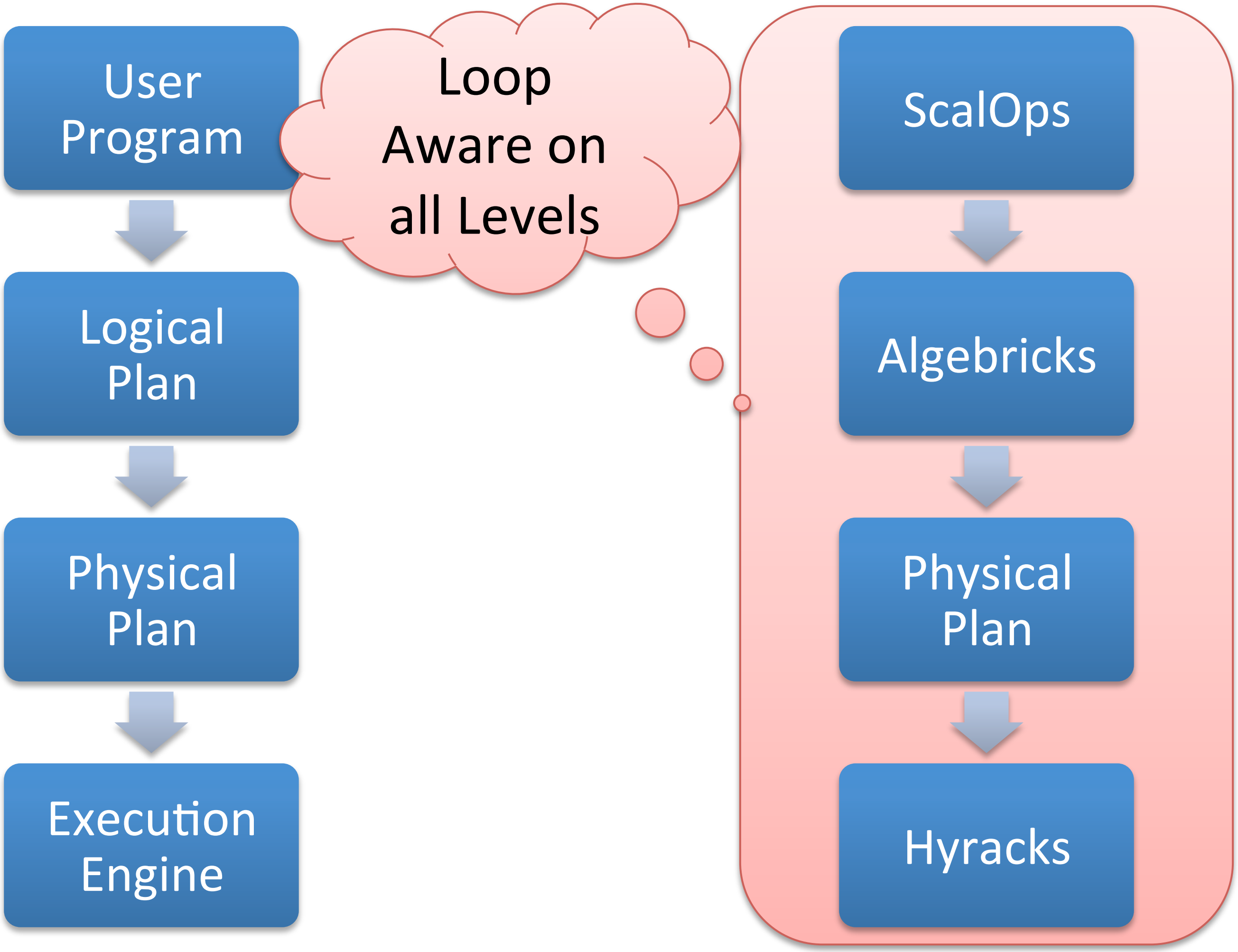
Goals

- **Integrate modeling and ETL workflows**
  - All Pig operators
  - Iteration is a first class citizen
  - Unify MPI, Pregel, MapReduce, … on a **single** runtime
- **Improve productivity**
  - Free the Programmer from runtime details (like MapReduce)
  - Facilitate easier job composition
  - IDE support
  - UDFs as first class citizens (unlike Pig)

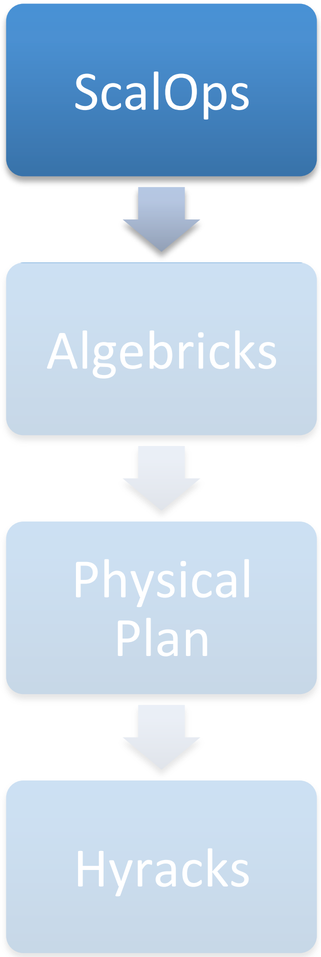
Vision



Vision


User Program
Logical Plan
Physical Plan
Execution Engine
Loop Aware on all Levels
ScalOps
Algebricks
Physical Plan
Hyracks

ScalOps – The Language



ScalOps – Overview

- Embedded Domain Specific Language in Scala
- All Pig Operators (Filter, Join, GroupBy, …)
- Iteration support
- Rich UDF support
  - Inline Scala function calls / literals
  - Everything callable from a JVM can be a UDF
- Support in major IDEs

Example: Batch Gradient Descent (BGD)

Until Convergence:

$$w_{t+1} = \underbrace{(1.0 - \eta\lambda) * \Big(w_t}_{\text{Regularization}} - \eta \underbrace{\sum_{(x,y)} \partial_w l\big(y, \langle w_t, x\rangle\big)}_{\text{Data Parallel Sum}}\Big)$$

---

w_t: Current Model
x: Data
y: Label

l: loss function (e.g. squared error)
∂: Gradient operator

BGD in ScalOps

```
def t
  class E    ctorType, la    doubleT    delta:Doubl    extends Environment
  val initi   alue = new Env(Vec    Typ  eros(                    le.MaxValue)
  loop(initialValue, (env: Env) => env.delt
      val gradient = xy.map(x=>compute_gra
      val loss     = xy.map(x=>compu  e_los
      env.w       -= gradient
      env.delta    = env.lastLoss - loss
      env.lastLoss = loss
      env
    }
  }
}
```

Training data; Table is
pe

Initializer

Loop Condition

Loop Body

Computes a gradient

Computes the loss

Native UDFs

Spark!?

- Scala DSL and runtime for data analytics

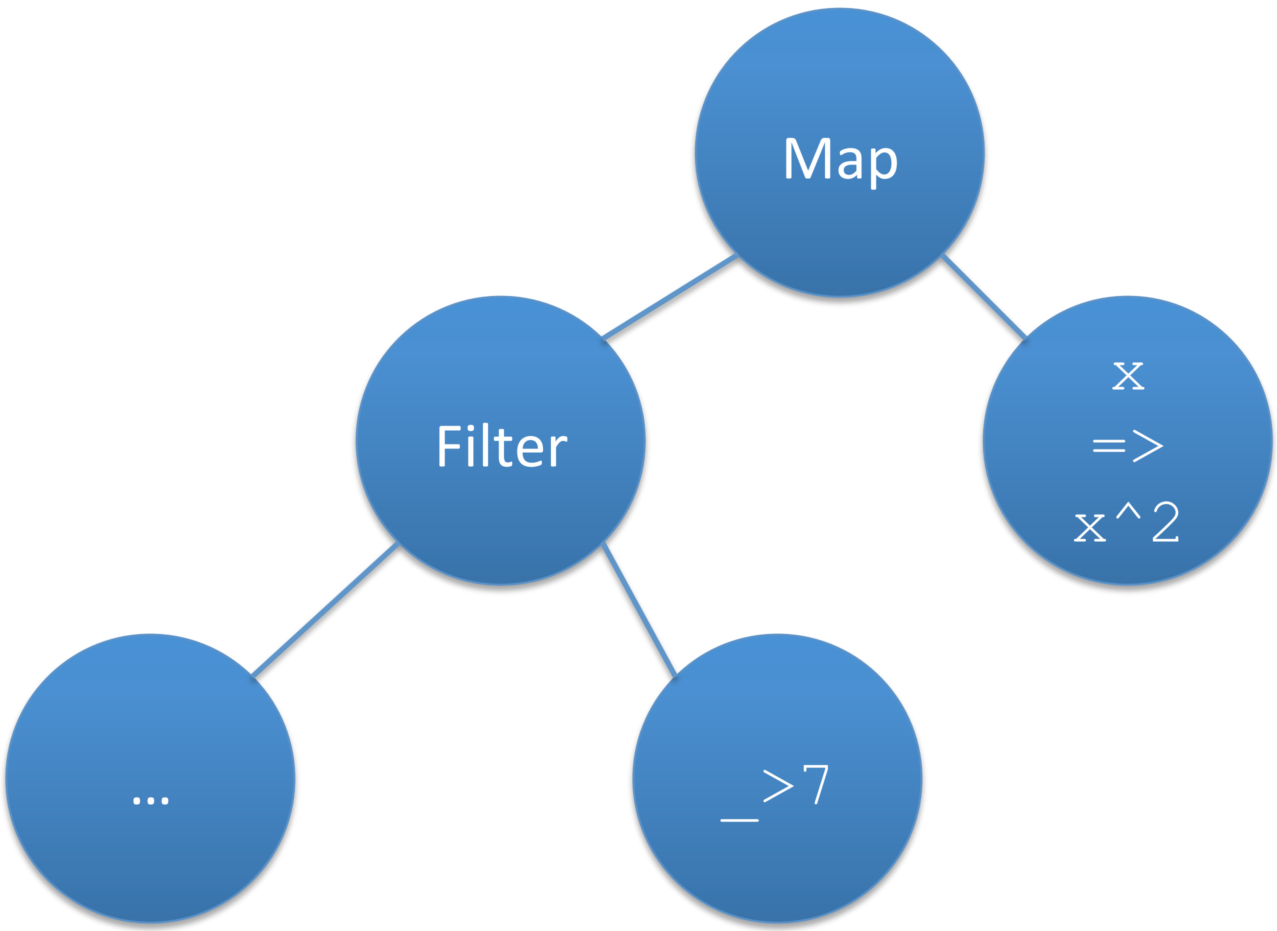
```
val points = spark.textFile(...).
                   map(parsePoint).
                   partitionBy(HashPartitioner(NODES)).
                   cache()
```

Physical Layer

```
      (1 / (1 + exp(-p.y*(w dot p.x))) - 1) * p.y * p.x  ).
      reduce(_ + _)
    w -= gradient
  }
```

Parse Tree Extraction Example

```
table.filter(_>7).map(x=>x^2)
```



Automatic Optimizations

```
def train(xy:Table[Example],
          compute_grad:(Example, Vector) => Vector,
          compute_loss:(Example, Vector) => Double) = {

  class Env(w:VectorType, lastError:DoubleType, delta:DoubleType) extends Environment

  val initialValue = new Env(VectorType.zeros(1000), Double.MaxValue, Double.MaxValue)

  loop(initialValue, (env: Env) => env.delta < eps) { env => {
      val gradient = xy.map(x=>compute_grad(x,env.w)).reduce(_+_)
      val loss     = xy.map(x=>compute_loss(x,env.w)).reduce(_+_)
      env.w       -= gradient
      env.delta    = env.lastLoss - loss
      env.lastLoss = loss
      env
    }
  }
}
```

Automatic Optimizations

```
def train(xy:Table[Example],
          compute_grad:(Example, Vector) => Vector,
          compute_loss:(Example, Vector) => Double)

  class Env(w:VectorType, lastError:DoubleType, delta.                         nvironment

  val initialValue = new Env(VectorType.zeros(1000), Double.MaxValue, Double.MaxValue)

  loop(initialValue, (env: Env) => env.delta < eps) { env => {
      val gradient = xy.map(x=>compute_grad(x,env.w)).reduce(_+_)
      val loss     = xy.map(x=>compute_loss(x,env.w)).reduce(_+_)
      env.w       -= gradient
      env.delta    = env.lastLoss - loss
      env.lastLoss = loss
      env
    }
  }
}
```

Merge into one MapReduce Step

Automatic Optimizations

```
def train(xy:Table[Example],
          compute_grad:(Example, Vector) => Vector,
          compute_loss:(Example, Vector) => Double) = {

  class Env(w:VectorType, lastError:DoubleType, delta:DoubleType) extends Environment

  val initialValue = new Env(VectorType.zeros(1000), Double.MaxValue, Double.MaxValue)

  loop(initialValue, (env: Env) => env.delta < eps) { env => {
      val gradient = xy.map(x=>compute_grad(x,env.w)).reduce(_+_)
      val loss     = xy.map(x=>compute_loss(x,env.w)).reduce(_+_)
      env.w       -= gradient
      env.delta    = env.lastLoss - loss
      env.lastLoss = loss
      env
    }
  }
}
```

Automatic Optimizations

Merge into one Operator

```
def train(xy:Table[Example],
          compute_grad:(Example, Vector) => Vector,
          compute_loss:(Example, Vector) => Double) =

  class Env(w:VectorType, lastError:DoubleType, delta:Do        Environment

  val initialValue = new Env(VectorType.zeros(1000), Double.MaxValue, Double.MaxValue)

  loop(initialValue, (env: Env) => env.delta < eps) { env => {
      val gradient = xy.map(x=>compute_grad(x,env.w)).reduce(_+_)
      val loss     = xy.map(x=>compute_loss(x,env.w)).reduce(_+_)
      env.w        -= gradient
      env.delta     = env.lastLoss - loss
      env.lastLoss = loss
      env
    }
  }
}
```

Automatic Optimizations

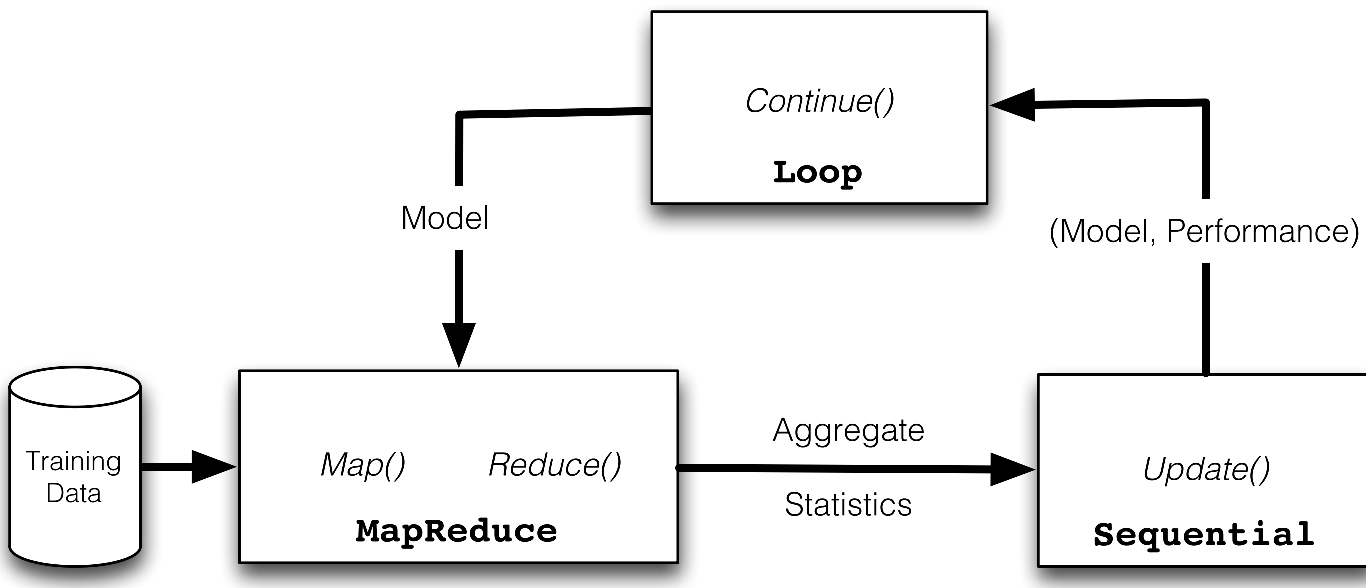
```
def train(xy:Table[Example],
          compute_grad:(Example, Vector) => Vector,
          compute_loss:(Example, Vector) => Double) = {

  class Env(w:VectorType, lastError:DoubleType, delta:DoubleType) extends Environment

  val initialValue = new Env(VectorType.zeros(1000), Double.MaxValue, Double.MaxValue)

  loop(initialValue, (env: Env) => env.delta < eps) { env => {
      val gradient = xy.map(x=>compute_grad(x,env.w)).reduce(_+_)
      val loss     = xy.map(x=>compute_loss(x,env.w)).reduce(_+_)
      env.w       -= gradient
      env.delta    = env.lastLoss - loss
      env.lastLoss = loss
      env
    }
  }
}
```

Automatic Optimizations

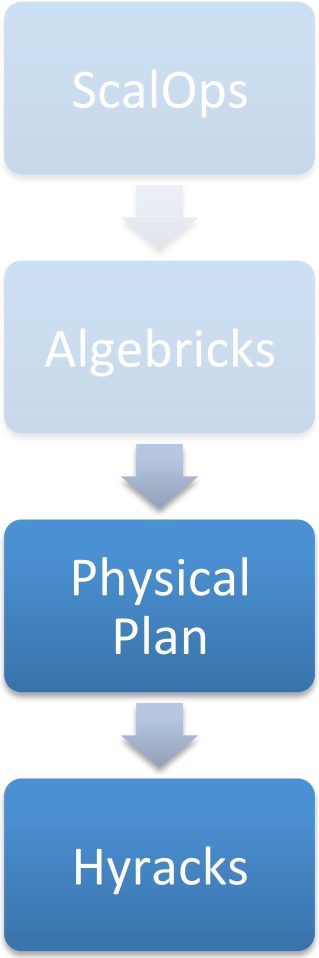
```
def train(xy:Table[Example],
          compute_grad:(Example, Vector) => Vector,
          compute_loss:(Example, Vector) => Double) = {

  class Env(w:VectorType, lastError:DoubleType, delta:DoubleType) extends Environment

  val initialValue = new Env(VectorType.zeros(1000), Double.MaxValue, Double.MaxValue)

  loop(initialValue, (env: Env) => env.delta < eps) { env => {
      val gradient = xy.map(x=>compute_grad(x,env.w)).reduce(_+_)
      val loss     = xy.map(x=>compute_loss(x,env.w)).reduce(_+_)
      env.w       -= gradient
      env.delta    = env.lastLoss - loss
      env.lastLoss = loss
      env
    }
  }
}
```

Cache **xy** in main memory, if possible

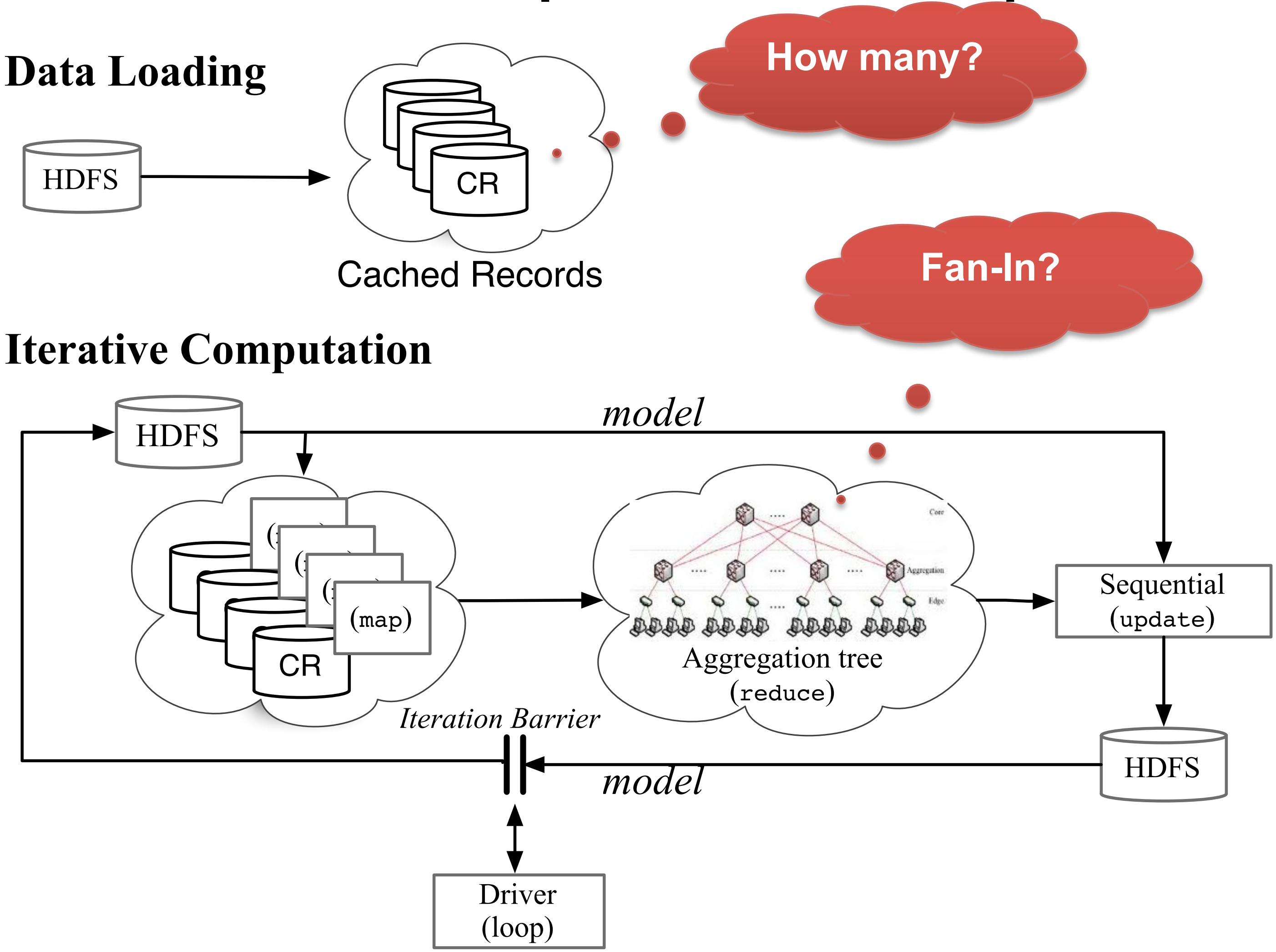
Result: Logical Plan



Physical Optimizer



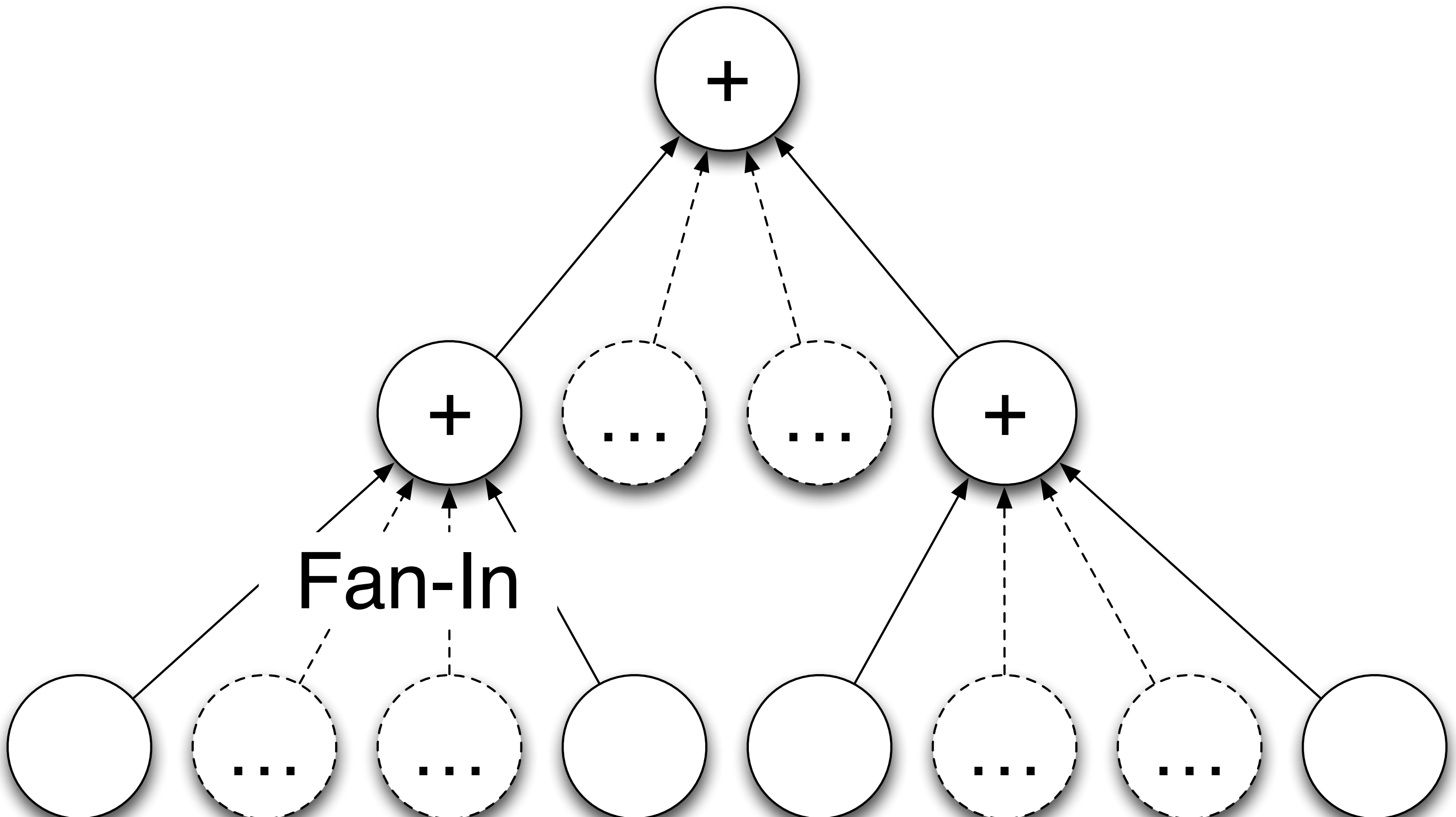
Iterative Map-Reduce-Update



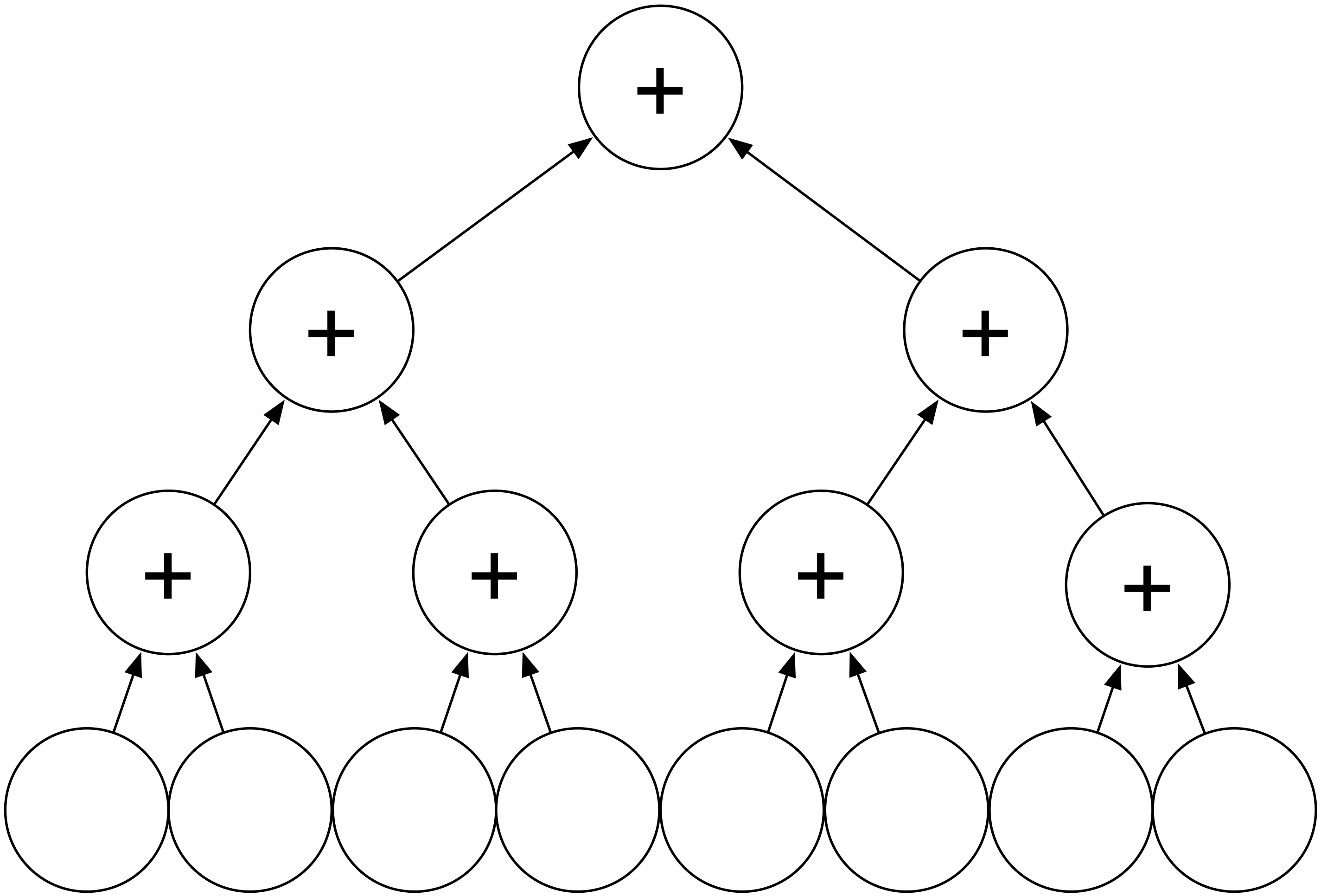
Other "Optimizations"

- Caching, "Rocking"
- Data-Local Scheduling
- Iteration-Aware Scheduling
- Avoid (de-)serialization
- Minimize #network connections
- Pipelining
- …

Optimal Aggregation Tree Fan-In

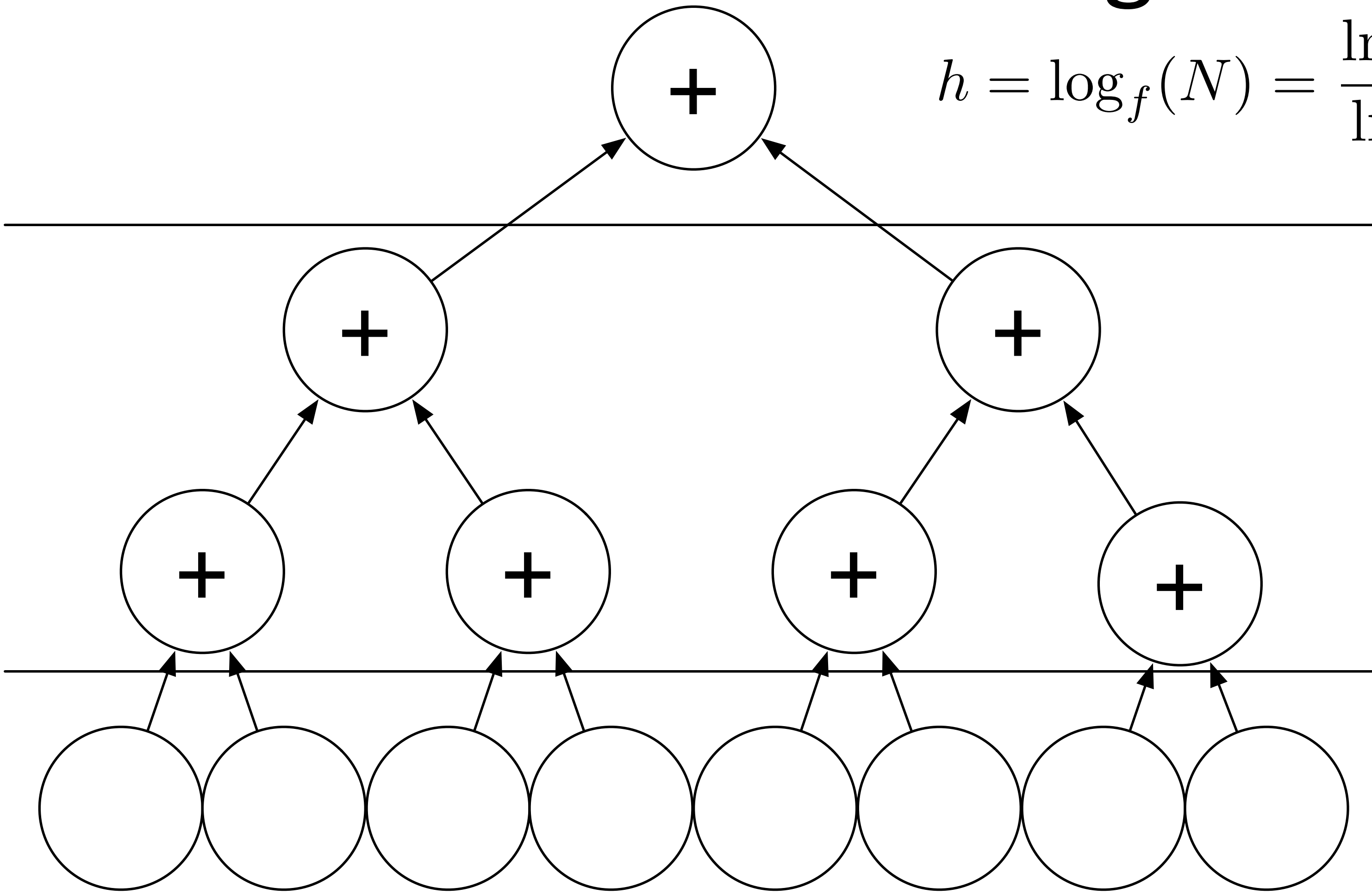


Tree Fan-In



Tree Fan-In: Blocking

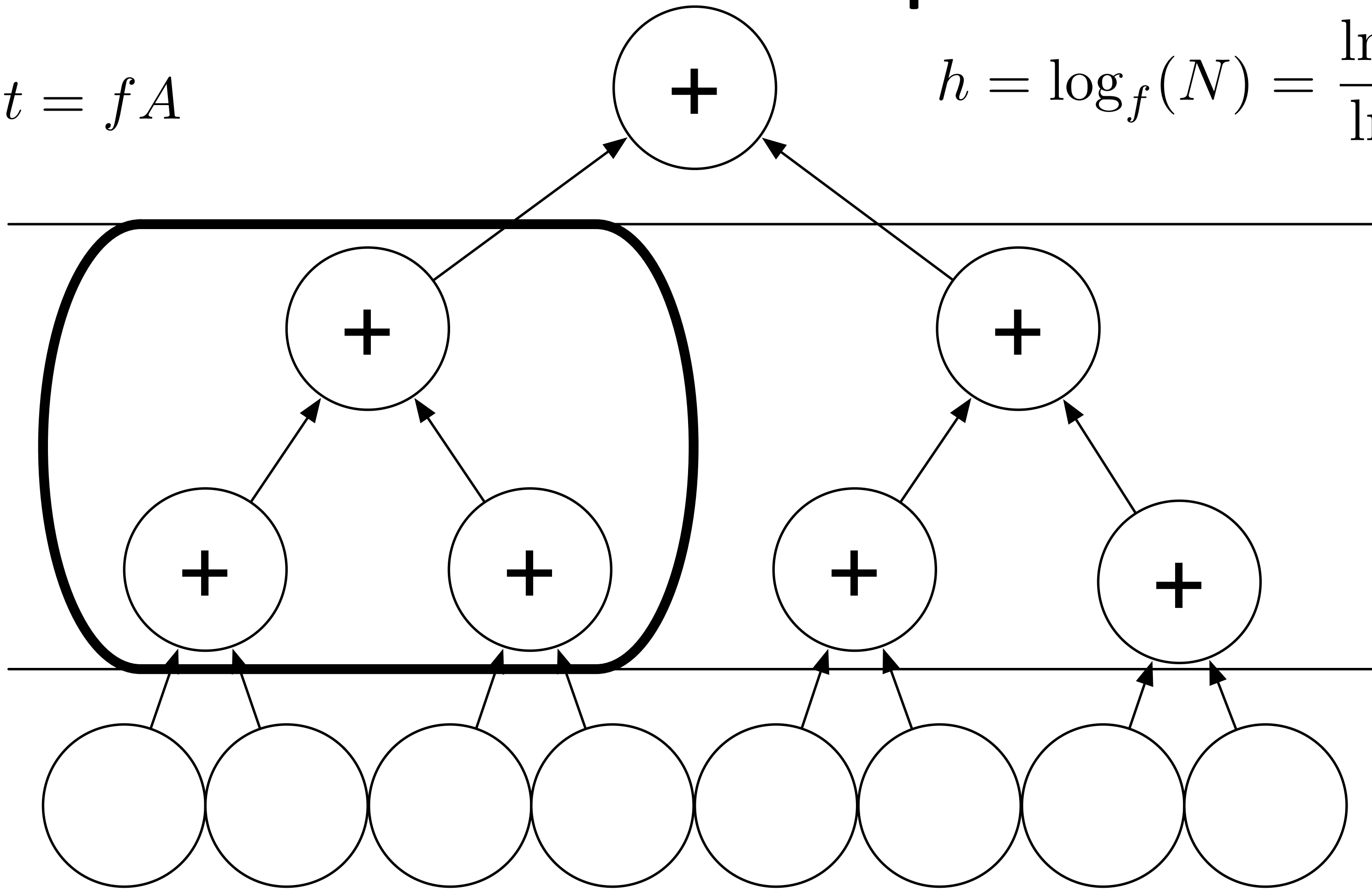
$$h = \log_f(N) = \frac{\ln(N)}{\ln(f)}$$



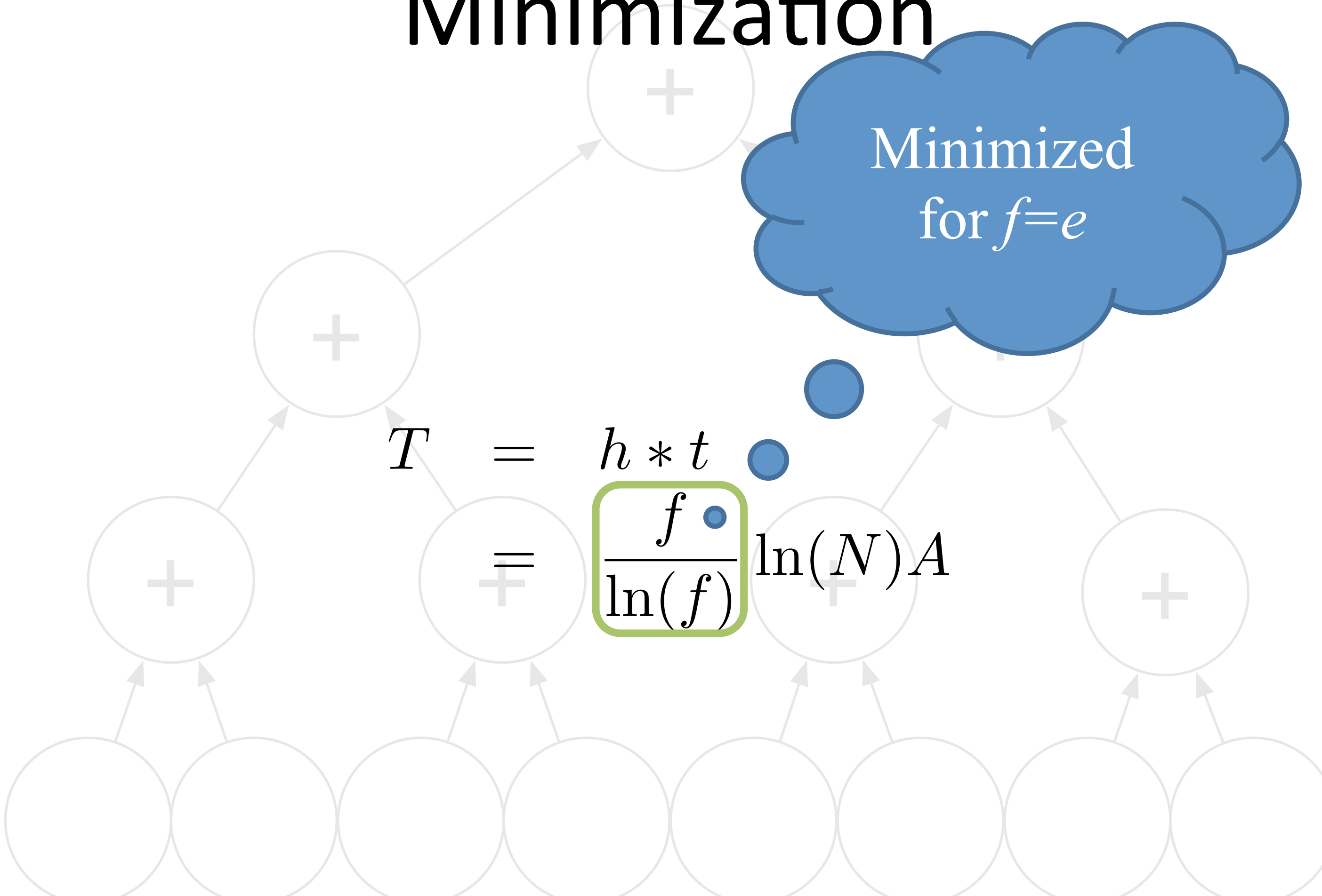
Tree Fan-In: Time per level

$t = fA$

$h = \log_f(N) = \frac{\ln(N)}{\ln(f)}$



Overall Aggregation Time Minimization



Minimized for f=e

$$
\begin{aligned}
T &= h * t \\
&= \boxed{\frac{f}{\ln(f)}} \ln(N) A
\end{aligned}
$$

Optimal Partitioning: Time per Iteration

Aggregation time

Map time

Processing Time

$$
\begin{aligned}
T &= T_A(N) + T_M(N) \\
&= Ae\ln(N) + \frac{RD + RP}{N} - MD
\end{aligned}
$$

Time for not loading from Disk

Load Time

Symbol	Meaning
R	# Records
M	Cache capacity per CPU
P	Map time per record
D	Load time per record
A	Aggregation time per record

Optimal Choices (Summary)

- **Minimal Wall Clock Time**
  - Balance aggregation & map time
  - Almost always: Use as many machines as you can

- **Minimal Cost** (time x #machines)
  - If your data fits into distributed RAM: do that
  - Else: It’s complicated

Time Optimal Partitioning

Let $R \leq MN$. The **time-minimal** number of machines for an Iterative Map-Reduce-Update operator is

$$\hat{N}_1 = \frac{RP}{Ae}$$

Let $R > MN$. The **time-minimal** number of machines for an Iterative Map-Reduce-Update operator is

$$\hat{N}_1 = \frac{RD + RP}{Ae}$$

Symbol	Meaning
R	# Records
M	Cache capacity per CPU
P	Map time per record
D	Load time per record
A	Aggregation time per record

Most often: Use as many machines as you have

Cost Optimal Partitioning

Let $R \leq MN$. The **cost-minimal** number of machines for an Iterative Map-Reduce-Update operator is

$$\hat{N}_1 = \frac{R}{M}$$

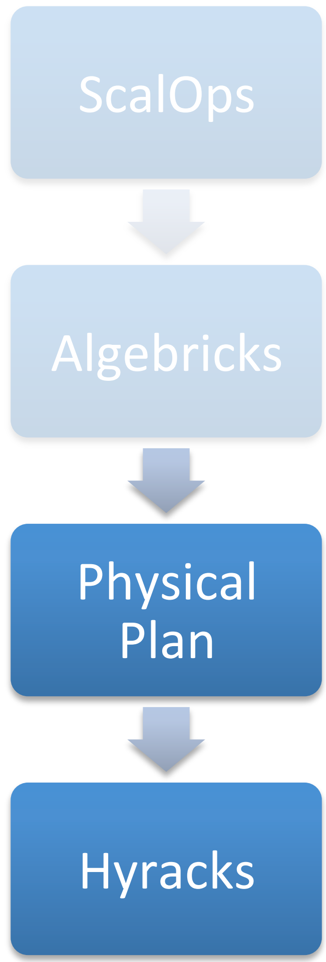
Let $R > MN$. The **cost-minimal** number of machines for an Iterative Map-Reduce-Update operator is

$$\hat{N}_1 = e^{\frac{MD}{Ae}}$$

The solution heavily depends on your job

Symbol	Meaning
R	# Records
M	Cache capacity per CPU
P	Map time per record
D	Load time per record
A	Aggregation time per record

Evaluation



Evaluation Methodology

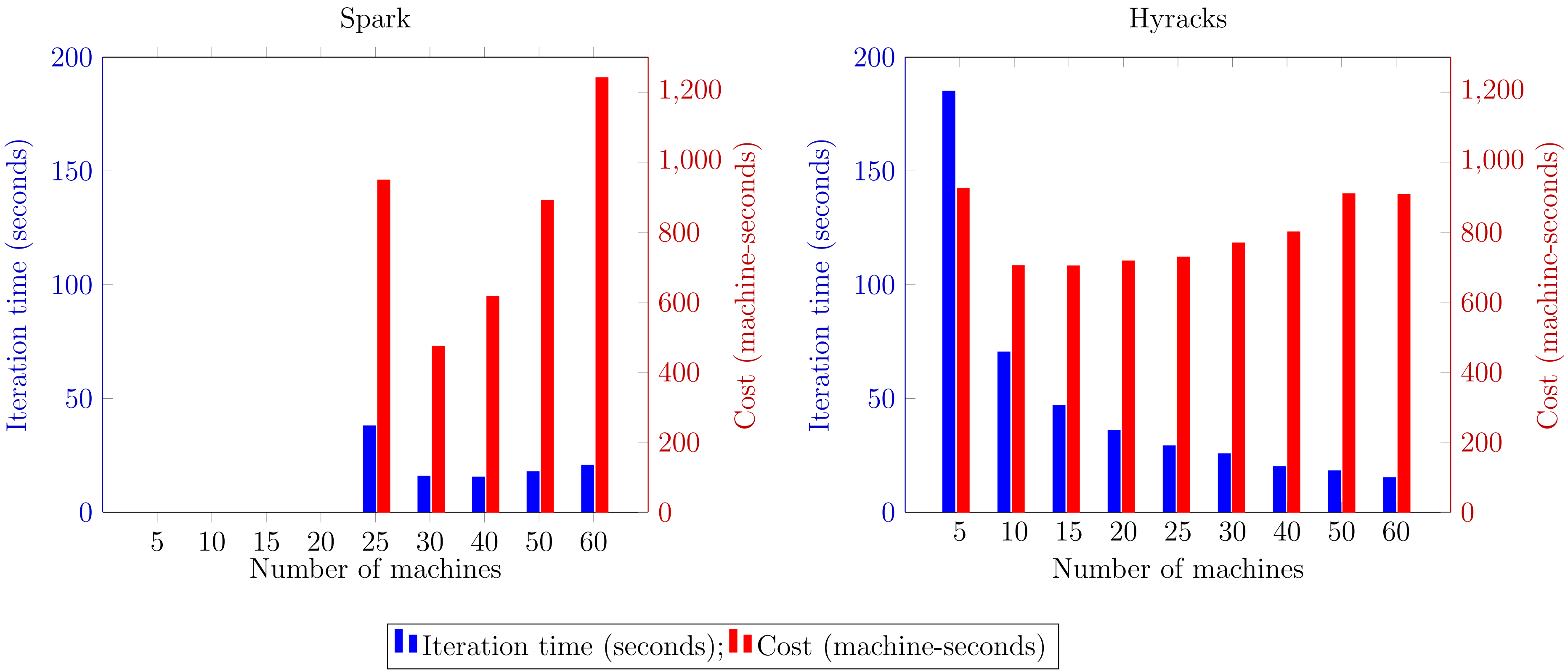
- **Metrics**
  - Iteration time
  - Cost: iteration time x number of machines
- **Speed-up**
  - Fix the data size and scale up # of machines
  - Goal: identify cost optimal # of machines
- **Scale-up**
  - Start with cost optimal configuration
  - Proportionally increase data size and # of machines


Hi, Tyson | Sign Out | Help
Make Y! My Homepage
Mail | My Y! | Yahoo!
YAHOO! NEWS
Search
Search Web
HOME
U.S.
WORLD
BUSINESS
ENTERTAINMENT
SPORTS
TECH
POLITICS
SCIENCE
HEALTH
BLOGS
LOCAL
POPULAR
News Search
News Search
Videos
Photos
Driven
Future Is Now
Trending Now
Vitality
Who Knew?
Power Players
Remake America
YOUR FRIENDS' ACTIVITY
Let your friends be your guide to great contents on Yahoo! News by connecting to Facebook.
By connecting you'll be able to see friends' activities and add your own. Learn more »
1 of 28
Photos: Thousands march in Florida for Trayvon Martin
3 winners, over 100 million Mega Millions losers
Santorum's tone on Ryan shifts after Romney endorsement
YOU ON YAHOO! NEWS
Hi there
Login with Facebook to personalize your Yahoo! News experience. Learn more »
NEWS FOR YOU
Public opinion shifts on Trayvon Martin case
7 California boys arrested in attack on teen
7 people to blame for the Trayvon Martin hysteria
Workers restoring Russian mansion find treasure
Calif. doctor accused of giving daughter propofol
Top Stories
ABC News
Latest News
Slideshows
AP
Reuters
AFP
Hillary Clinton: Time running out for diplomacy with Iran AP - 2 hrs 49 mins ago
U.S. Secretary of State Hillary Rodham Clinton made clear Saturday that time is running out for diplomacy over Iran's nuclear program and said talks aimed at preventing Tehran from acquiring a nuclear weapon would resume … More »
ICC: Viral video will spur Kony arrest this year AFP - 32 mins ago
The International Criminal Court's chief prosecutor voiced confidence Saturday that fugitive Ugandan rebel chief Joseph Kony will be arrested this year, praising the role of a
1962-2012
Milgard
50th ANNIVERSARY
Clearly the best
Get a $50 rebate on every Milgard window or door you purchase.
Hurry! Only lasts until March 31st
Click Here for more info.

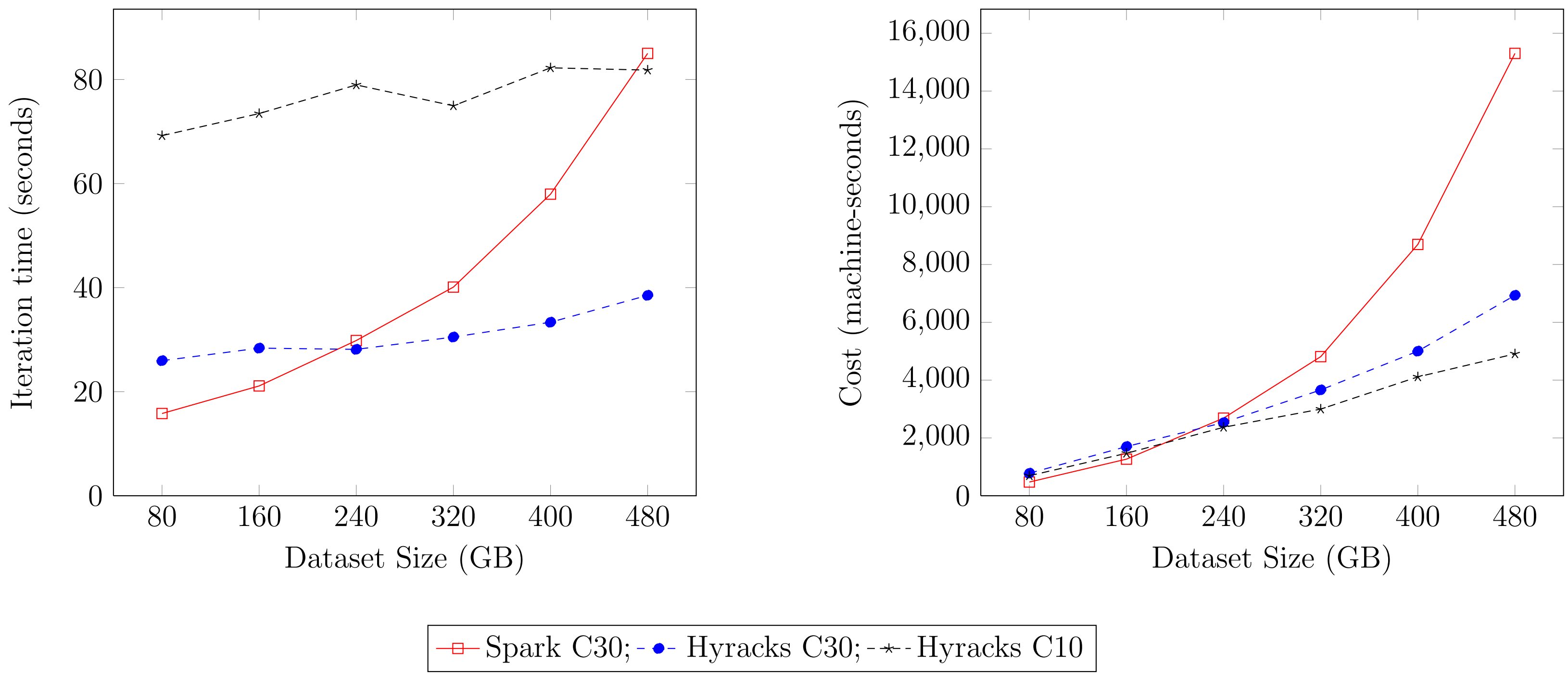
News Recommendation

- **Task**
  - Predict news click-through rate
  - Linear Model
- **Data**
  - 120GB in libsvm text format
- **Hardware**
  - 150 Machines in 5 Rack, 1Gbps Ethernet
  - Each machine: 8 Cores, 4 Disks, 16GB RAM

Spark vs. Hyracks Speedup



Spark vs. Hyracks Scale-up

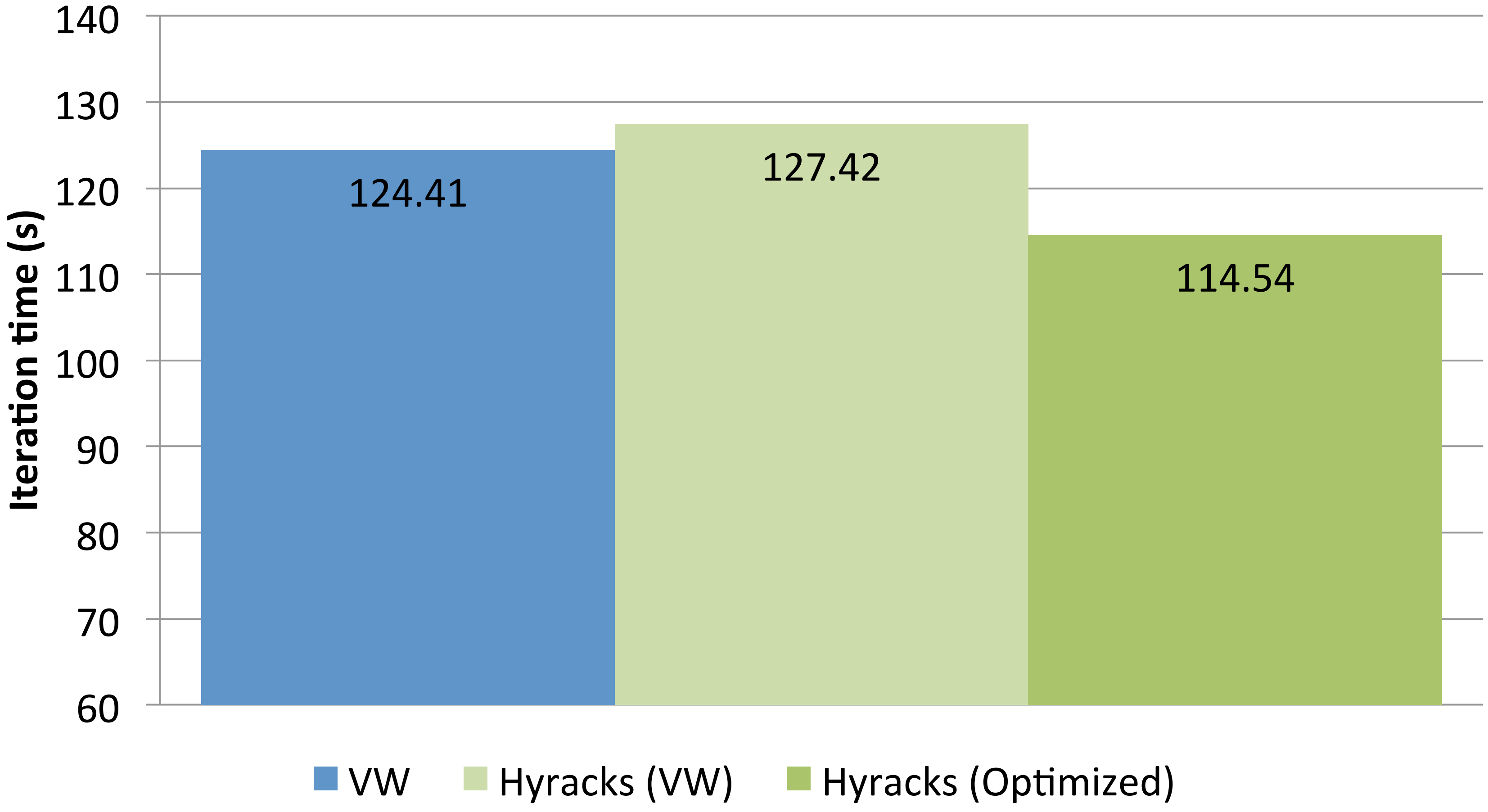



Hi, Tyson | Sign Out | Help
Make Y! My Homepage
Mail | My Y! | Yahoo!
YAHOO! NEWS
Search
Search Web
HOME U.S. WORLD BUSINESS ENTERTAINMENT SPORTS TECH POLITICS SCIENCE HEALTH BLOGS LOCAL POPULAR
News Search
News Search
Videos Photos Driven Future Is Now Trending Now Vitality Who Knew? Power Players Remake America
YOUR FRIENDS' ACTIVITY
Let your friends be your guide to great contents on Yahoo! News by connecting to Facebook.
By connecting you'll be able to see friends' activities and add your own. Learn more »
1 of 28
Photos: Thousands march in Florida for Trayvon Martin
3 winners, over 100 million Mega Millions losers
Santorum's tone on Ryan shifts after Romney endorsement
Top Stories ABC News Latest News Slideshows AP Reuters AFP
Hillary Clinton: Time running out for diplomacy with Iran AP - 2 hrs 49 mins
U.S. Secretary of State Hillary Rodham Clinton made clear Saturday that time is run
out for diplomacy over Iran's nuclear program and said talks aimed at preventing Tehr
from acquiring a nuclear weapon would resume … More »
ICC: Viral video will spur Kony arrest this year AFP - 32 mins ago
The International Criminal Court's chief prosecutor voiced confidence Saturday that
fugitive Ugandan rebel chief Joseph Kony will be arrested this year, praising the role of a
YOU ON YAHOO! NEWS
Hi there
Login with Facebook to personalize your Yahoo!
News experience. Learn more »
Login with Facebook
NEWS FOR YOU
Public opinion shifts on Trayvon Martin case
7 California boys arrested in attack on teen
7 people to blame for the Trayvon Martin hysteria
Workers restoring Russian mansion find treasure
Calif. doctor accused of giving daughter propofol
1962-2012
Milgard
50th
ANNIVERSARY
Clearly the best
Get a $50 rebate on every Milgard
window or door you purchase.
Hurry! Only lasts until March 31st
Milgard
Click Here

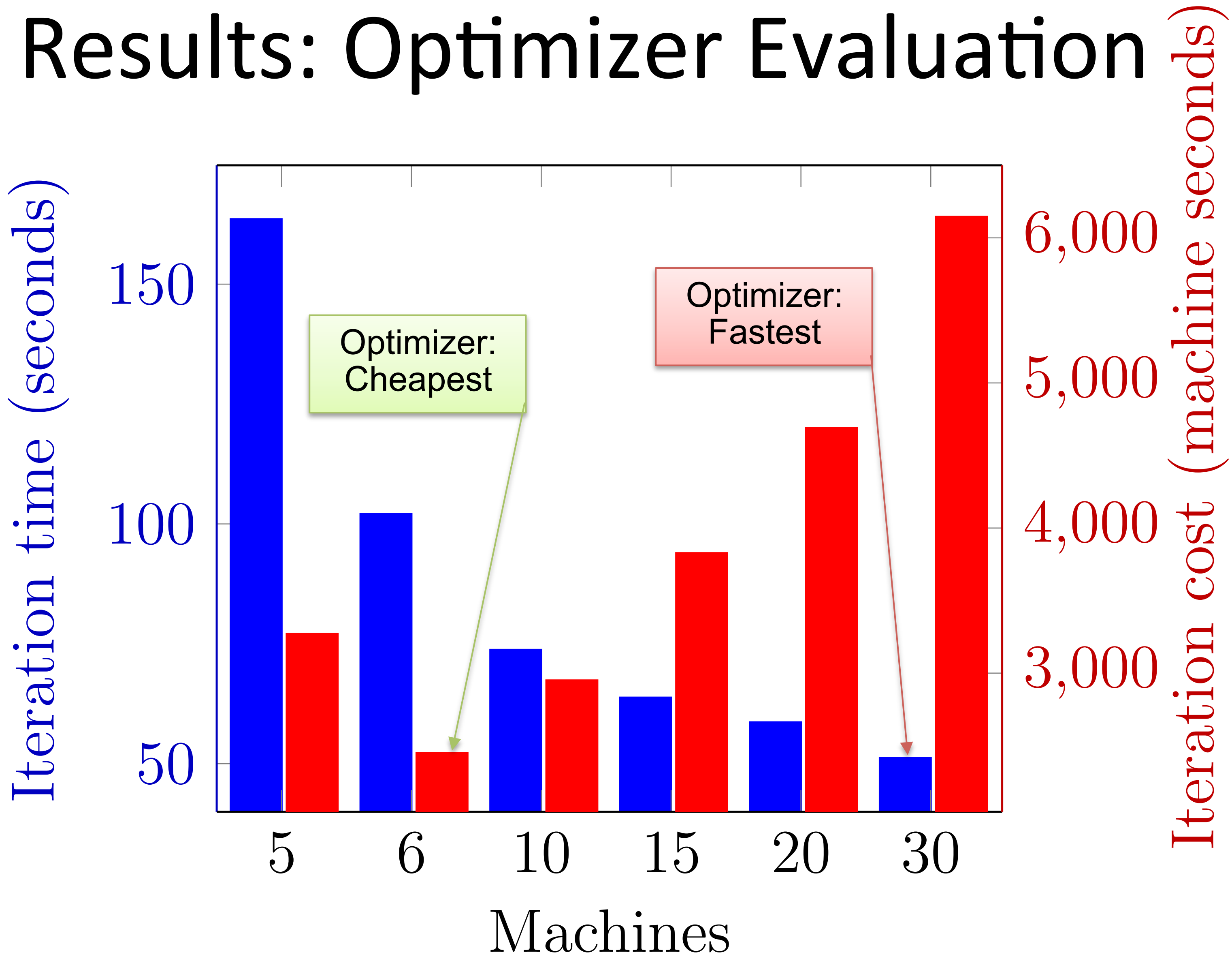
Personalized Advertisement

- **Task**
  - Predict ad click-through rate
  - Linear Model, learned with BGD
- **Data**
  - 500GB in VW text format
- **Hardware**
  - 30 Machines in one Rack 1Gbps Ethernet
  - Each machine: 8 Cores, 4 Disks, 16GB RAM

Grounding Experiment



Results: Optimizer Evaluation



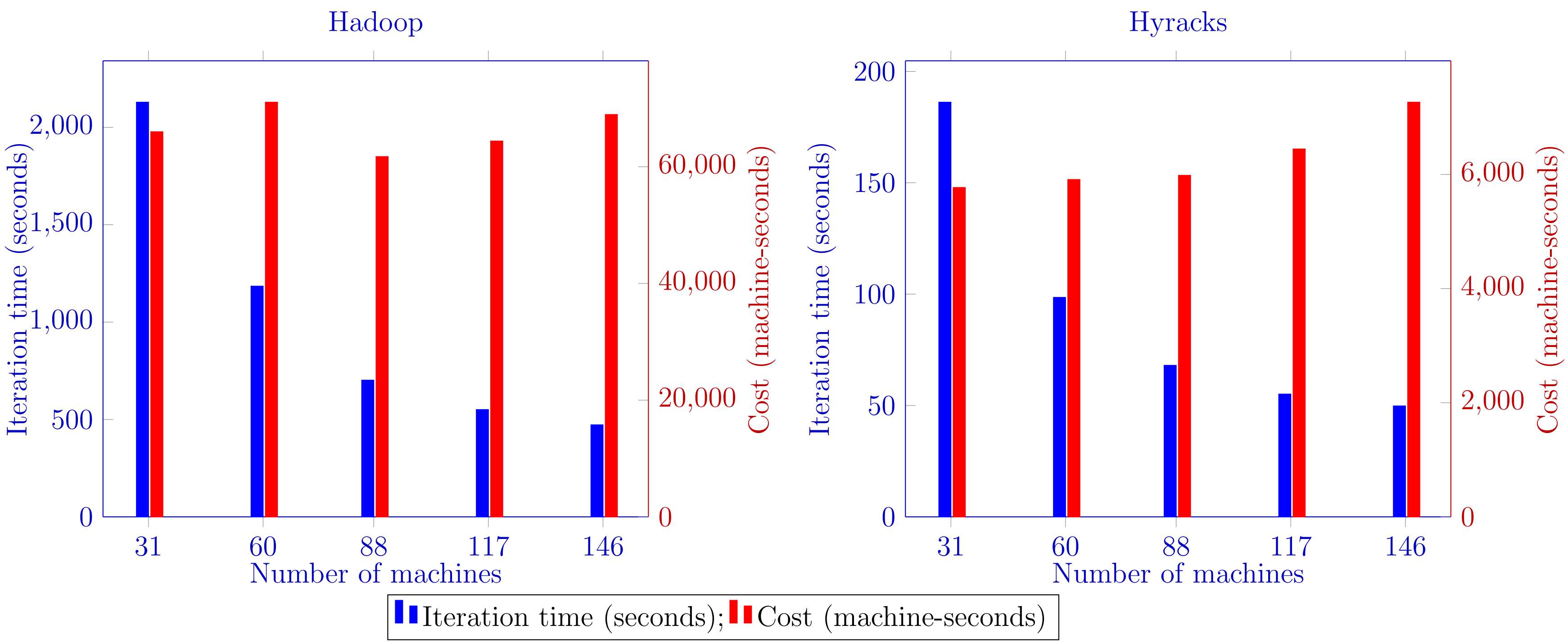
Experiments in the Pregel Model

- **Task**
  - Compute PageRank
- **Data**
  - Yahoo! Webmap as available on Webscope
  - 1.4B nodes, 8GB on disk
- **Cluster**
  - 150 Machines in 5 Rack, 1Gbps Ethernet
  - Each machine: 8 Cores, 4 Disks, 16GB RAM

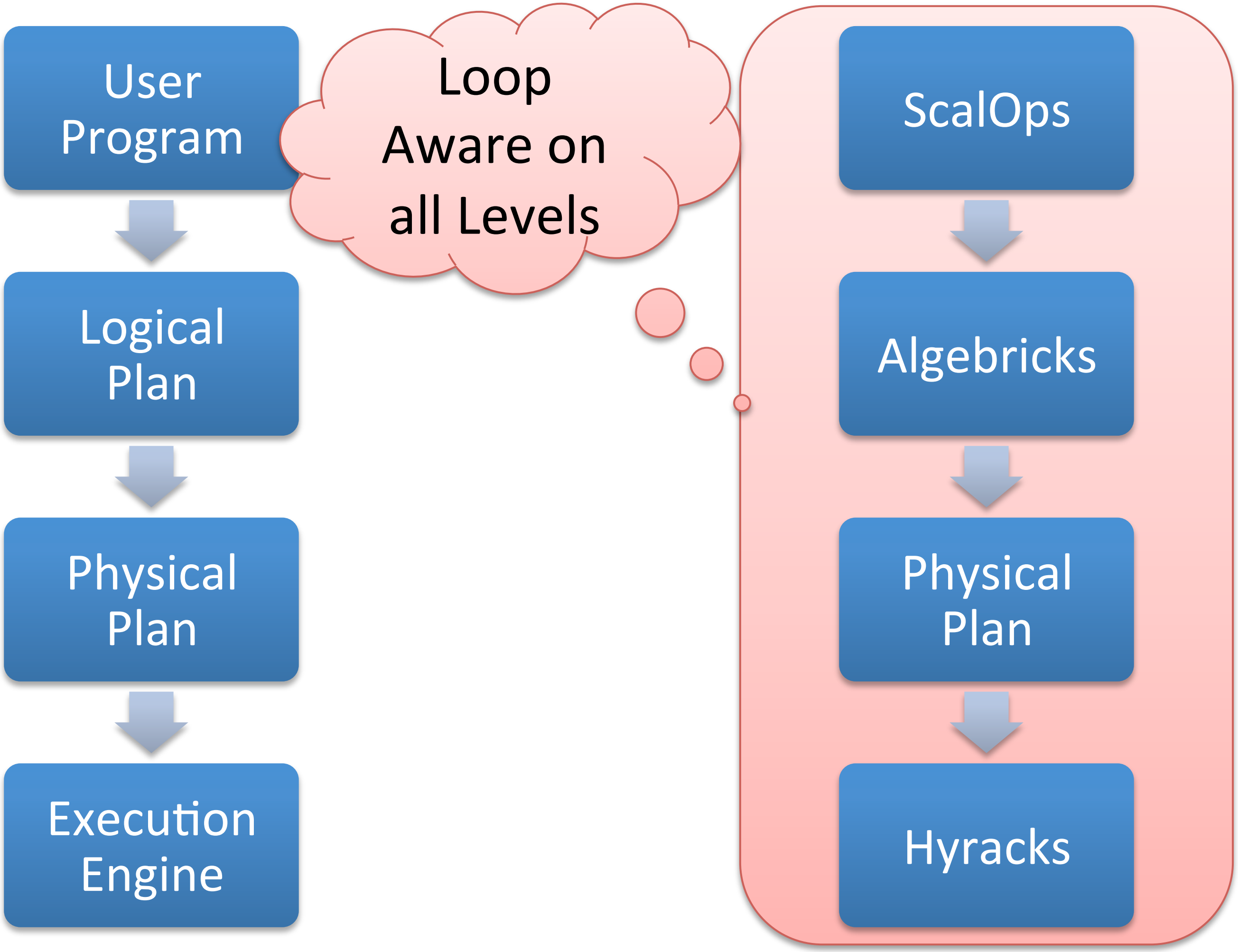
Related Work

- **Three OSS systems can run the task**
  - Hadoop
  - Hyracks
  - GraphLab 2 (different computation model)
- **Several systems failed despite 3.2TB RAM**
  - Giraph/Golden Orb (by transitive closure)
  - Spark (despite Matei's help)
  - Mahout

Hyracks vs. Hadoop Pagerank Speedup



Conclusion



User Program
Logical Plan
Physical Plan
Execution Engine
Loop Aware on all Levels
ScalOps
Algebricks
Physical Plan
Hyracks

Benefits

- Unifies both ETL and Iterative Computation in a **single** framework
  - Simplifies Job Composition
- Optimizable Execution Plans
  - Imperative for compute clouds
  - Supports different optimization goals

Future Work

- **Build & package it for consumption**
- **Optimizer for recursive data flows**
  – Example: Auto-detect the need for caching
- **Expose runtime policies to the DSL layer**
  – Example: Make fault tolerance optional
- **Support Asynchronous Computation**
  – Important for Graphical Models

Coordinates

- **Hyracks**
  - http://code.google.com/p/hyracks/
  - http://asterix.ics.uci.edu/
- **Markus Weimer**
  - mweimer@microsoft.com
  - @markusweimer
  - http://cs.markusweimer.com