Big Data Learning Systems

Tyson Condie

Cloud and Information Services Lab, Microsoft

Collaborators: Markus Weimer, Rusty Sears, Byung-Gon Chun, Shravan Narayanamurthy, Sriram Rao, Carlo Curino, Raghu Ramakrishnan
“Machine Learning is Programming by Example”

Used when:
Programming is hard (e.g. topic detection, bioinformatics)
Program changes all the time (recommender systems, antispam)
Machine Learning

- **Supervised**
  - Classification
  - Regression
  - Recommender

- **Unsupervised**
  - Clustering
  - Dimensionality reduction
  - Topic modeling

Big: TiB - PiB

Data → Learning → Model

Small: MiB - GiB
Machine Learning Workflow

Step I: Example Formation
Feature and Label Extraction

Step II: Modeling

Step III: Evaluation (and eventually Deployment)
Example Formation at Scale

Feature Extraction

EMail

Click Log

Data Parallel Functions

Label Extraction

ID
Bag of Words

ID
Label

ID
Label

Example

(Large Scale) Join

ID
Bag of Words

Label
Machine Learning Workflow

Step I: Example Formation
Feature and Label Extraction

Step II: Modeling

Step III: Evaluation
Step I: Example Formation
   Feature and Label Extraction

Step II: Modeling

Step III: Evaluation
Modeling (30,000ft)

Learning is Iterative

Apply Model to Data

Observe Errors

Update Model

There isn’t one computational model (yet)

**Statistical Query Model:** Algorithm operates on statistics of the dataset

**Graphical Models:** Heavy message passing, possibly asynchronous.

**Many more:** Custom solutions
Machine Learning Workflow

Step I: Example Formation
Feature and Label Extraction

Step II: Modeling
Cumbersome & Not scalable

Step III: Evaluation
Distributed Learning

Machine Learning in MapReduce?

+ MapReduce model fits statistical query model learning

- Hadoop MR does not support iterations (30x slowdown compared to others)
- Hadoop MR does not match other forms of algorithms

“Solution”: Map only jobs

1. Allocate a set of map tasks
2. Instantiate learning algorithm
3. Execute iterative algorithm until convergence
4. Release mappers

➡ Hadoop Abuse
Hadoop Abusers 1: (All)Reduce and friends

Decision Trees on Hadoop
Jerry Ye et al.
Runs OpenMPI on a map only job
Uses HDFS for coordination/bootstrap

Vowpal Wabbit
John Langford et al.
AllReduce on a map-only job
Uses custom server for coordination/bootstrap
Constructs a binary aggregation tree
Optimizes node selection via redundant tasks

Worker/Aggregator Abstraction
Markus Weimer and Sriram Rao
Iterative Map-Reduce-Update on a map-only job
Uses Zookeeper for coordination/bootstrap
Custom communication between workers and aggregator
Hadoop Abusers 2: The Graph View

Apache Giraph
Avery Chen et al.
Open source implementation of Pregel on a map-only job
Uses Zookeeper for coordination/bootstrap
Graph computation using “think like a vertex” UDFs
Executes BSP message passing algorithm

Yahoo! LDA (YLDA)
Alex Smola and Shravan Narayanamurthy
Instantiate Graphical Model on a map-only job
Uses HDFS for coordination/bootstrap
Coordinate global parameters via shared memory
Problems with this Approach

Problems for the Abusers

Fault Tolerance Mismatch
MapReduce fault tolerance actually gets in the way

Resource Model Mismatch
MapReduce’s resource selection often suboptimal for the job at hand (data local vs. communication)

Cumbersome integration with M/R

Every Abuser has to implement ...
Networking
Cluster Membership
Bulk data transfers
...

Problems for the Cluster

Abusers Violate MapReduce assumptions
Network usage bursts in (All)Reduce
Local disk use in VW

The Abusers are disrespectful of other users
  e.g. production workflows
  Hoarding of resources (even worse as Hadoop does not support preemption)
Rise of the Resource Managers
YARN: Hadoop v2
Resource Allocation = list of (node type, count, resource)

E.g.
{ (node1, 1, 1GB), (rack-1, 2, 1GB),(*, 1, 2GB) }
YARN: Hadoop v2

Client → Monitor Job → Resource Manager

E.g., MapReduce Scheduler

Resource Pool → Node Manager

E.g., Map Task

Resource Pool → Node Manager

E.g., Reduce Task

Resource Pool → Node Manager

E.g., Reduce Task

Resource Pool → Node Manager

E.g., Map Task

Resource Pool → Node Manager

E.g., Reduce Task

Resource Pool → Node Manager

E.g., Map Task

Resource Pool → Node Manager

E.g., Reduce Task

Resource Pool → Node Manager

E.g., Map Task

Resource Pool → Node Manager

E.g., Reduce Task

Resource Pool → Node Manager

E.g., Map Task

Resource Pool → Node Manager

E.g., Reduce Task

Resource Pool → Node Manager

E.g., Map Task

Resource Pool → Node Manager

E.g., Reduce Task

Resource Pool → Node Manager

E.g., Map Task

Resource Pool → Node Manager

E.g., Reduce Task

Resource Pool → Node Manager

E.g., Map Task

Resource Pool → Node Manager

E.g., Reduce Task

Resource Pool → Node Manager

E.g., Map Task

Resource Pool → Node Manager

E.g., Reduce Task

Resource Pool → Node Manager

E.g., Map Task

Resource Pool → Node Manager

E.g., Reduce Task

Resource Pool → Node Manager

E.g., Map Task

Resource Pool → Node Manager

E.g., Reduce Task

Resource Pool → Node Manager

E.g., Map Task

Resource Pool → Node Manager

E.g., Reduce Task
YARN: Hadoop v2
YARN: Hadoop v2
YARN: Hadoop v2
YARN: A step in the right direction

Disentangles resource allocation from the computational model
  YARN manages the cluster resource allocations
  Application masters manage computation on allocated resources

Low-level API
  Containers are empty processes (with resource limits)
  No assumed higher-level semantics
REEF: Retainable Evaluator Execution Framework
Goals for REEF

Ease development on resource managers (like YARN)
Cluster membership: Heartbeats, Failure notification, etc.
Networking: Naming, Message Passing, Group Communications, etc.
State management: Checkpointing, Storage, etc.
...

Unify different computations on a single runtime
e.g. Map/Reduce followed by MPI followed by stream processing
Hand-over of resources (containers on the machines)
Hand-over of data and state (ideally, in RAM)
The Challenge

SQL / Hive  ...  ...  Machine Learning

YARN / HDFS
The Solution: add a layer of indirection
REEF in the Stack (Future)
REEF: Computation and Data Management

Extensible Control Flow

- **Job Driver**: Control plane implementation. **User code** executed on YARN’s Application Master.
- **Activity**: **User code** executed within an **Evaluator**.
- **Evaluator**: Execution Environment for **Activities**. One **Evaluator** is bound to one YARN Container.

Data Management Services

- **Storage**
  - Abstractions: Map and Spool
  - Local and Remote
- **Network**
  - Message passing
  - Bulk Transfers
  - Collective Communications
- **State Management**
  - Fault Tolerance
  - Checkpointing
REEF Control Flow
Running Example: Distributed Shell

Run `ls` on these nodes!
public class DistributedShell {
    ...
    public static void main(String[] args) {
        ...
        Injector i = new Injector(yarnConfiguration);
        ...
        REEF reef = i.getInstance(REEF.class);
        ...
        reef.submit(driverConf);
    }
}
public class DistributedShell {
    ...
    public static void main(String[] args) {
        ...
        Injector i = new Injector(yarnConfiguration);
        ...
        REEF reef = i.getInstance(REEF.class);
        ...
        reef.submit(driverConf);
    }
}
public class DistributedShellJobDriver {
    private final EvaluatorRequestor requestor;
    ...

    public void onNext(StartTime time) {
        requestor.submit(EvaluatorRequest.Builder()
            .setSize(SMALL).setNumber(2)
            .build());
    }
    ...
}
public class DistributedShellJobDriver {
    private final EvaluatorRequestor requestor;
    ...

    public void onNext(AllocatedEvaluator eval) {
        Configuration contextConf = ...;
        eval.submitContext(contextConf)
    }
    ...
}
The REEF Control Flow

Client

Job Driver
REEF
HDFS
NM

Context Evaluator
HDFS
NM

context config

Name Node
YARN RM

HDFS
NM
The REEF Control Flow
```java
public class DistributedShellJobDriver {
    private final String cmd = "ls";

    public void onNext(ActiveContext ctx) {
        final String activityId = [...];
        Configuration activityConf = Activity.CONF
            .set(IDENTIFIER, "ShellActivity")
            .set(ACTIVITY, ShellActivity.class)
            .set(COMMAND, this.cmd)
            .build();
        ctx.submitActivity(activityConf);
    }
}
```
class ShellActivity implements Activity {

    private final String command;
}

@Inject
ShellActivity(@Parameter(Command.class) String c) {
    this.command = c;
}

private String exec(final String command){
    ...
}

@Override
public byte[] call(byte[] memento) {
    String s = exec(this.cmd);
    return s.getBytes();
}
The REEF Control Flow

Client

Name Node
YARN RM

Activity
Context
Evaluator
HDFS
NM

Job Driver
REEF
HDFS
NM

Activity
Context
Evaluator
HDFS
NM

heartbeat()
The REEF Control Flow

Client

Name Node
YARN RM

Job Driver
REEF
HDFS
NM

Activity
Context
Evaluator
HDFS
NM

Activity
Context
Evaluator
HDFS
NM
The REEF Control Flow

Client

CompletedActivity

Job Driver

REEF
HDFS
NM

Name Node
YARN RM

Activity
Context Evaluator
HDFS
NM

Activity
Context Evaluator
HDFS
NM
REEF Control Flow: Summary

Control Flow is centralized in the Driver
Evaluator allocation & configuration
Activity configuration & submission

Error Handling is centralized in the Driver
When an Activity throws an Exception, we ship & throw it at the Driver
When an Evaluator dies, we throw an Exception at the Driver

All APIs are asynchronous
Driver files requests via non-blocking API calls
REEF fires events at user (e.g. Evaluator availability, Exceptions, ...)
Goal: REEF is stateless for fault-tolerant drivers

Easy to reason about and debug
Scalable
Checkpoint Services

Activity
suspend()

preempt cont.
on node4

suspend()

Job Driver
REEF
HDFS NM

Evaluator
Activity

services

HDFS NM

HDFS NM

node1

node2

node3

node4
Checkpoint Services

Name Node  Yarn RM  HDFS NM  REEF  HDFS NM  HDFS NM  Job Driver

Client

node1  node2  node3  node4

Activity

Evaluator

Services

HDFS  NM

Retrieve Checkpoint
Learning in REEF
1. Driver Launches
2. Driver Launches Evaluators
3. Driver submits LoadActivity
4. Activity loads Data
5. Activity finishes

6. Until Converged:
   Driver submits ComputeGradient
   Gradient is shipped to the Driver
Conclusion
Logical/Physical Separation

Logical Layer

Physical Layer

- Observation #1: Enables query optimization
- Can we automate this translation?
Logical/Physical Separation

Logical Layer

ML algorithm

Physical Layer

SQM

Graph Analysis

Select, Project, Join, Group

MapReduce

MPI

• Observation #2: Systems have to solve the same problems and adopt similar solutions

• Can we isolate these solutions in reusable modules?
A Unifying Design

SQM → ML algorithm

Logical query over training data

Query optimizer

Parallel Recursive Dataflow

REEF

Graph Analysis
Recursion is built into the language  
Amenable to optimizations  
Lots of existing work that we can leverage

- J. Eisner and N. Filardo. Dyna: Extending datalog for modern AI. In Datalog ‘10  
- D. Deutch, C. Koch, T. Milo. On Probabilistic Fixpoint and Markov Chain Query Languages. In PODS ‘10  
Version 0.1

- Implementation over Hyracks
- Supports both Iterative-MRU and Pregel
- Standard optimizations + some new tricks
• Provenance for triage
  • “My model misbehaves - why?”
• Fault-awareness policies
• Incremental learning

Storage/Networking
• Dynamic resources
  • Elastic operators

Query Processing
• Cost estimation for recursive computation
  • Cost models (time vs money)

State management
• Interactive Query Processing
  • Provenance for triage
  • “My model misbehaves - why?”

Fault-awareness policies
• Incremental learning

Caching policies
Conclusion

• Open source release soon
  • Apache 2 license
  • MapReduce support (including Hive)

• Machine learning libraries supported
  • Iterative Map-Reduce-Update
  • MPI (Graphical Models)
  • Mahout compatibility?

• Contact: Tyson Condie
  • tcondie@microsoft.com
  • tcondie@cs.ucla.edu