Scale-out Beyond MapReduce

Raghu Ramakrishnan
Cloud Information Services Lab (CISL)
Microsoft
Outline

• Big Data
  – The New Applications
  – The Digital Shoebox
    • Tiered Storage
    • Compute Fabric

• REEF
Cloud Information Services Lab (CISL)

• Applied research for Cloud and Enterprise (CE)

• Focus areas:
  – Cloud data platforms, predictive analytics and data-driven enterprise applications

• Modus innovatii:
  – Embedded with the product team
  – Engage closely with MSR
  – Balance of external and internal impact
Big Data
What’s the big deal?
What’s New?

• **What we’re doing with it!**
  – The tech is best thought of in terms of what it enables

• Why is this more than tech evolution?
  – Cloud services + advances in analytics + HW trends = **Ability to cost-effectively do things we couldn’t dream of before**
  – Uncomfortably fast evolution = **revolution**
Challenges

• Is there real technical innovation here?
  – Yes: Elastic scale-out; heterogeneous data and analysis; real-time/interactive; “instant on” cloud access
  – Many fascinating challenges, no deal breakers

• What about the social, legal and regulatory issues?
  – Will take longer to understand and resolve

• Biggest gap
  – People: data scientists, data-driven managers (McKinsey)
  – NAE CATS Big Data training workshop (planned for Jan 2014)
The “index” is keyed by concept instance, and organizes all relevant information, wherever it is drawn from, in semantically meaningful ways.
idli Restaurante near Ann Arbor, MI

1. Madras Masala ★★★★★ (14)
   madrasmasala.com
   (734) 222-9006 - 328 Maynard St, Ann Arbor, MI
   Menu: idli
   11 Reviews | Overview | Directions

2. Temptations ★★★★★ (25)
   temptationsrestaurant.com
   (734) 433-4907 - 2876 Washtenaw Rd, Ypsilanti, MI
   Menu: idli
   23 Reviews | Overview | Directions

3. Selvi's Wet Grinder / Mixie Store
   local.yahoo.com
   (734) 697-3303 - 8748 W Walden Dr, Belleville, MI
   Overview | Directions

Soy Noodle Recipe Recipe Soy Noodle Recipe: 2 Recipes
Abc Seafood Kuching Ammamma Recipes Ann Arbor Com Assam Fish Bobby Flay Cilantro ... Juice Soup Kaddu Masala Rava Idli Morcovi Muschiulet National Nutrition Month Navratri ...
en.petitchef.com/tags/recipes/soy-noodle-recipe - Cached

Soft Taco Stack Recipe Soft Taco Stack: 1 Recipes
Amish Cooking Ann Arbor Appalam Kuzhambu Baby Sweet Basundhi Bulgur Cranberry Salad Chancho Agridulce Idli Mario Batali Milagu Ponnal Mint Leaf
Content Optimization

Content Recommendation on Web Portals

Key Features

Package Ranker (CORE)
Ranks packages by expected CTR based on data collected every 5 minutes

Dashboard (CORE)
Provides real-time insights into performance by package, segment, and property

Mix Management (Property)
Ensures editorial voice is maintained and user gets a variety of content

Package rotation (Property)
Tracks which stories a user has seen and rotates them after user has seen them for a certain period of time

Key Performance Indicators

Lifts in quantitative metrics
Editorial Voice Preserved
## CORE Dashboard: Segment Heat Map

<table>
<thead>
<tr>
<th>Package</th>
<th>male</th>
<th>female</th>
<th>OMG</th>
<th>BU Auto</th>
<th>BU Ent</th>
<th>BU Fin</th>
<th>Health</th>
<th>BUSport</th>
<th>NBA</th>
<th>BUT trav</th>
<th>ALL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>406,200</td>
<td>350,043</td>
<td>270,039</td>
<td>121,060</td>
<td>325,673</td>
<td>195,796</td>
<td>350,152</td>
<td>132,186</td>
<td>123,338</td>
<td>923,611</td>
<td></td>
</tr>
<tr>
<td></td>
<td>18,440</td>
<td>14,449</td>
<td>16,940</td>
<td>7,369</td>
<td>20,012</td>
<td>7,639</td>
<td>21,451</td>
<td>9,457</td>
<td>7,996</td>
<td>36,457</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0,0452</td>
<td>0,0371</td>
<td>0,0676</td>
<td>0,0617</td>
<td>0,0614</td>
<td>0,0652</td>
<td>0,0613</td>
<td>0,0712</td>
<td>0,0416</td>
<td>0,0416</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8,417</td>
<td>-11.13</td>
<td>50,661</td>
<td>45,564</td>
<td>47,483</td>
<td>66,553</td>
<td>53,691</td>
<td>0,0000</td>
<td>0,0000</td>
<td>0,0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>18,067</td>
<td>8,669</td>
<td>8,586</td>
<td>8,885</td>
<td>9,744</td>
<td>6,067</td>
<td>3,769</td>
<td>6,500</td>
<td>2,490</td>
<td>18,137</td>
<td></td>
</tr>
<tr>
<td></td>
<td>885</td>
<td>644</td>
<td>3,529</td>
<td>885</td>
<td>9,744</td>
<td>3,769</td>
<td>6,439</td>
<td>2,490</td>
<td>1,738</td>
<td>130,143</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1,0565</td>
<td>111,405</td>
<td>189,386</td>
<td>231,465</td>
<td>221,222</td>
<td>198,362</td>
<td>911</td>
<td>330</td>
<td>0,0000</td>
<td>0,0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>153,684</td>
<td>111,405</td>
<td>189,386</td>
<td>231,465</td>
<td>221,222</td>
<td>198,362</td>
<td>911</td>
<td>330</td>
<td>0,0000</td>
<td>0,0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2,667</td>
<td>2,501</td>
<td>140,943</td>
<td>140,943</td>
<td>140,943</td>
<td>140,943</td>
<td>140,943</td>
<td>140,943</td>
<td>140,943</td>
<td>140,943</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3,326</td>
<td>3,954</td>
<td>3,521</td>
<td>1,004</td>
<td>3,016</td>
<td>2,931</td>
<td>3,180</td>
<td>3,180</td>
<td>3,180</td>
<td>3,180</td>
<td></td>
</tr>
<tr>
<td></td>
<td>249</td>
<td>78,9</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td></td>
</tr>
<tr>
<td></td>
<td>78,9</td>
<td>249</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2,667</td>
<td>2,501</td>
<td>140,943</td>
<td>140,943</td>
<td>140,943</td>
<td>140,943</td>
<td>140,943</td>
<td>140,943</td>
<td>140,943</td>
<td>140,943</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3,326</td>
<td>3,954</td>
<td>3,521</td>
<td>1,004</td>
<td>3,016</td>
<td>2,931</td>
<td>3,180</td>
<td>3,180</td>
<td>3,180</td>
<td>3,180</td>
<td></td>
</tr>
<tr>
<td></td>
<td>249</td>
<td>78,9</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td></td>
</tr>
<tr>
<td></td>
<td>78,9</td>
<td>249</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2,667</td>
<td>2,501</td>
<td>140,943</td>
<td>140,943</td>
<td>140,943</td>
<td>140,943</td>
<td>140,943</td>
<td>140,943</td>
<td>140,943</td>
<td>140,943</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3,326</td>
<td>3,954</td>
<td>3,521</td>
<td>1,004</td>
<td>3,016</td>
<td>2,931</td>
<td>3,180</td>
<td>3,180</td>
<td>3,180</td>
<td>3,180</td>
<td></td>
</tr>
<tr>
<td></td>
<td>249</td>
<td>78,9</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td></td>
</tr>
<tr>
<td></td>
<td>78,9</td>
<td>249</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2,667</td>
<td>2,501</td>
<td>140,943</td>
<td>140,943</td>
<td>140,943</td>
<td>140,943</td>
<td>140,943</td>
<td>140,943</td>
<td>140,943</td>
<td>140,943</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3,326</td>
<td>3,954</td>
<td>3,521</td>
<td>1,004</td>
<td>3,016</td>
<td>2,931</td>
<td>3,180</td>
<td>3,180</td>
<td>3,180</td>
<td>3,180</td>
<td></td>
</tr>
<tr>
<td></td>
<td>249</td>
<td>78,9</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td></td>
</tr>
<tr>
<td></td>
<td>78,9</td>
<td>249</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td></td>
</tr>
</tbody>
</table>
Kinect

• The Kinect is an array of sensors.
  – Depth, audio, RGB camera ...

• SDK provides a 3D virtual skeleton.
  – 20 points around the body, 30 fps
  – 30 frames per second
  – Between 60-70M sold by May 2013

• Exemplar of “Internet of Things”
  – Event streams from a multitude of devices, enabling broad new apps

(Slide modified from Assaf Schuster, Technion)
Kinect-based Full Body Gait Analysis

• **Non-intrusive** – Suitable for home monitoring.
  – Place Kinect anywhere in corridor or room, start measuring.
  – Measure gait as subjects go about their daily routine.

• **Comprehensive** – Extract parameters from full body.
  – Parameters extracted from 3D skeleton of entire body.
  – Simultaneously measure any part of the body.

• **Accurate** – Supervised learning overcomes errors.
  – Full body information improves accuracy.

(Slide courtesy Assaf Schuster, Technion)
Connected devices will soon be EVERYWHERE

During 2008, the number of things connected to the Internet exceeded the number of people on earth.

By 2020 there will be 50 billion.

(Slide courtesy Ratul Mahajan, MSR)

HomeOS: Another Instance of IoT

(Slide courtesy Ratul Mahajan, MSR)
Big Data
Build it—they’re here already!
One Slide MapReduce Primer

Data file
One Slide MapReduce Primer

HDFS
One Slide MapReduce Primer
One Slide MapReduce Primer

- HDFS
- Map tasks
One Slide MapReduce Primer

HDFS

Map tasks
One Slide MapReduce Primer

HDFS

Map tasks
One Slide MapReduce Primer

HDFS

Map tasks

Reduce tasks
One Slide MapReduce Primer

HDFS

Map tasks

Reduce tasks

HDFS

HDFS

HDFS
Good for scanning/sequentially writing/appending to huge files
Scales by “mapping” input to partitions, “reducing” partitions in parallel
Partitions written to disk for fault-tolerance
Expensive “shuffle” step between Map & Reduce
No concept of iteration

Hive and Pig are SQL variants implemented by translation to MapReduce

Not great for serving (reading or writing individual objects)
THE DIGITAL SHOEBOX

- Capture any data, react instantaneously, mix with data stored anywhere
  - Tiered storage management
  - Federated access
- Use any analysis tool (anywhere, mix and match, interactively)
  - Compute fabric
- Collaborate/Share selectively
Integrated Query “In-Place”
Can join and group-by tables from a relational source with tables in a Hadoop cluster without needing to learn MapReduce

Integrated BI Tools
Using Excel, end users can search for data sources with Power Query and do roll-up/drill-down etc. with Power Pivot—across both relational and Hadoop data

Interactive Visualizations
Use Power View for immersive interactivity and visualizations of both relational and Hadoop data
A COMMON VISION

The vision of supporting many kinds of scalable analytics over all of a user’s data is shared by many vendors:

- Aster/Teradata
- Berkeley Data Analytics Stack
- Cloudera
- Google
- HortonWorks
- Microsoft
- Pivotal/EMC

SQL on Hadoop panel, Aug 2013:
http://hivedata.com/real-time-query-panel-discussion/
Challenges

• **Volume**
  – Elastic scale-out
  – Multi-tenancy

• **Variety**
  – Trade-off: Shared building blocks vs. custom engines

• **Velocity**
  – Real-time and OLTP, interactive, batch
How Far Away is Data?

• GFS and Map-Reduce:
  – Schedule computation “near” data
  – i.e., on machines that have data on their disks

• But
  – Windows Azure Storage
    • And slower tiers such as tape storage ...
  – Main memory growth
    • And flash, SSDs, NVRAM etc. ...

• Must play two games simultaneously:
  – Cache data across tiers, anticipating workloads
  – Schedule compute near cached data
Compute Fabric: YARN

- **Resource manager** for Hadoop2.x
- Allocates compute containers to competing jobs
  - Not necessarily MR jobs!
  - **Containers** are the unit of resource
  - Can fail or be taken away; programmer must handle these cases
- Other RMs include Corona, Mesos, Omega
Making YARN Easier to Use: REEF

- **Evaluator**: YARN container with REEF services
  - Capability-awareness, Storage support, Fault-handling support, Communications, Job/task tracking, scheduling hooks
- **Activity**: User Code to be executed in an Evaluator
  - Monitored, preemptable, re-started as needed
  - Unique id over lifetime of job
  - Executes in an Evaluator, which can be re-used
Motivation: Machine Learning Workflow

Step I: Example Formation
Feature Extraction
Label Extraction

Step II: Modeling

Step III: Deployment (or just Evaluation)
Example: User Activity Modeling

- Large dimensionality vector describing possible user activities
- But a typical user has a sparse activity vector

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Possible Values</th>
<th>Typical values per user</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pages</td>
<td>~ MM</td>
<td>10 – 100</td>
</tr>
<tr>
<td>Queries</td>
<td>~ 100s of MM</td>
<td>Few</td>
</tr>
<tr>
<td>Ads</td>
<td>~ 100s of thousands</td>
<td>10s</td>
</tr>
</tbody>
</table>

- Hadoop pipeline to model user interests from activities
Feature and Target Windows

- **Query**: Visit Y! finance
- **Visit Y! finance**: Event of interest
- **Moving Window**
- **Feature Window**
- **Target Window**
## User Modeling Pipeline

<table>
<thead>
<tr>
<th>Component</th>
<th>Data Processed</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Acquisition</td>
<td>~ 1 Tb per time period</td>
<td>2 – 3 hours</td>
</tr>
<tr>
<td>Feature and Target Generation</td>
<td>~ 1 Tb * Size of feature window</td>
<td>4 - 6 hours</td>
</tr>
<tr>
<td>Model Training</td>
<td>~ 50 - 100 Gb</td>
<td>1 – 2 hours for 100’s of models</td>
</tr>
<tr>
<td>Scoring</td>
<td>~ 500 Gb</td>
<td>1 hour</td>
</tr>
</tbody>
</table>
Example Formation: SQL at Scale

Feature Extraction

EMail

ID
Bag of Words

Data Parallel Functions

Click Log

ID
Label

Label Extraction

Large Scale Join

ID
Bag of Words

Label

Example
Learning a Language Classifier

1. Wikipedia (en)
2. Wikipedia (de)
3. Character Frequencies
4. Feature extraction
5. Map (Reduce)
6. Training Examples
7. (Logistic) Regression
8. Learning
9. Language Classifier
10. Batch Gradient Descent
Scaling Model Building (30,000ft)

Learning is Iterative

- Apply Model to Data
- Observe Errors
- Update Model

Needs Iterative Dataflows

- Avoid forced rescheduling between iterations
- Node-local data storage / caches
- Machine learning cost is often I/O dominated
- Efficient means of communication *within* an iteration
The Challenge

- Fault Tolerance
- Row/Column Storage
- High Bandwidth Networking

SQL / Hive → ... → ... → Machine Learning

YARN / HDFS
The Challenge

- Fault *Awareness*
- Local data caching
- Low Latency Networking

YARN / HDFS
Can we share more than just Resource Management?
Take-Away

Machine Learning applications require a plethora of systems
Example Formation: Well covered by ‘standard’ DAG engines
Modeling: Much less standardized (the jury is still out)
Deployment: Very domain specific

All of these systems share lots of mechanisms at the bottom end
Bad for systems builders: Duplicated Effort
Bad for users: Cost of coordination between systems is very high
Bad for cloud providers: No elasticity between systems
REEF
Retainable Evaluator Execution Framework

The Team
MapReduce library

- Runs Hive and Pig
- Excellent starting point for M/R optimizations: Caching, Shuffle, Map-Reduce-Reduce, Sessions, ...

Machine Learning algorithms

- Scalable implementations: Decision Trees, Linear Models, Soon: SVD
- Excellent starting point for: Fault awareness in ML

Open-source release planned

Demo at VLDB
REEF in the Stack

- SQL / Hive
- ...
- ...
- Machine Learning

REEF

YARN / HDFS
REEF: Computation and Data Management

Extensible Control Flow

- Control plane implementation. **User code** executed on YARN’s Application Master
- **User code** executed within an **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
Running Example: Distributed Shell

Run ‘dir’ on these nodes!
The REEF Control Flow

Client

submit job

Name Node

YARN RM

launch container

REEF

HDFS

NM

HDFS

NM

public class DistributedShell {
  ... public static void main(String[] args) {
    ... Injector i = new Injector(yarnConfiguration);
    ... REEF reef = i.getInstance(REEF.class);
    ... reef.submit(driverConf);
  }
}
The REEF Control Flow

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());
    }
    ...
}

The REEF Control Flow

Client

Name Node | YARN RM

appears on the diagram as a client, asking for a container.

Job Driver

REEF | HDFS | NM

Request

ask for container

signed token

HDFS | NM

HDFS | NM
The REEF Control Flow
The REEF Control Flow

```
public class DistributedShellJobDriver {
    private final String cmd = "dir";

    public void onNext(RunningEvaluator eval) {
        final String activityId = [...];
        final JavaConfigurationBuilder b = [...];

        b.bind(Activity.class, ShellActivity.class);
        b.bindNamedParameter(Command.class, this.cmd);

        eval.submit(activityId, cb.build());
    }

    [...]
}
```
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
The REEF Control Flow

Client

CompletedActivity

Name Node

YARN RM

Job Driver

REEF

HDFS

NM

Activity

Evaluator

HDFS

NM

services

Retains State!
The REEF Control Flow
The REEF Control Flow

Client

CompletedActivity

Job Driver

REEF
HDFS
NM

Activity
Evaluator
HDFS
NM

Activity
Evaluator
HDFS
NM

Name Node
YARN RM

services
The REEF Control Flow

Client

Evaluator.
close()

Job Driver

REEF
HDFS
NM

CompletedEvaluator

Name Node
YARN
RM

Activity
Evaluator
HDFS
NM

services

services
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 Rx-Style asynchronous APIs
Driver files requests via non-blocking API calls
REEF fires events at user-provided event Observers (e.g. Evaluator availability, Exceptions, ...)
REEF maintains some useful state (e.g. the Evaluator used in “completed Activity” messages)
Majority of the state keeping (e.g. work queues) is maintained by the Driver.
Learning in REEF
The Task: Learn a Regression Model

Given: Dataset $X$ of Examples
Each Example consists of
  a Feature Vector $x$ (e.g. words in a document) and
  a Label $y$ (e.g. the fraction of people that clicked it).

Desired: A predictor
“Find a function $f(x)$ such that $f(x) \approx y$ for unseen data”

More precisely, find:

$$\hat{f} = \arg\min_f \sum_{(x,y) \in X} (f(x) - y)^2$$
Linear Models

The set of functions $f(x)$ is infinite $\Rightarrow$ We can’t find a solution

Approach: Restrict the set of functions to linear functions

We choose linear functions parameterized by the weight vector $w$ :

$$f(x) := f_w(x) := \langle w, x \rangle = \sum_i w_i x_i$$

Hence, we can now search for $\hat{w}$ :

$$\hat{w} = \arg\min_w \sum_{(x,y) \in X} (f_w(x) - y)^2 = \sum_{(x,y) \in X} (\langle w, x \rangle - y)^2$$
The Learning Algorithm: Batch Gradient Descent

The objective function is convex and differentiable in $w$

Algorithm (simplified):

Start with a random $w_0$

Until convergence:
  Compute the gradient

$$
\partial_w = \sum_{(x,y) \in X} 2 \langle w, x \rangle - y
$$

Apply the gradient to the model

$$
w_{t+1} = w_t - \partial_w
$$
How It Maps to REEF

Start with a random $w_0$

Until convergence:

Compute the gradient

$$\partial_w = \sum_{(x,y) \in X} 2 (\langle w, x \rangle - y)$$

Apply the gradient to the model

$$w_{t+1} = w_t - \partial_w$$

Driver

Activity (per partition)
How It Maps to REEF: Control Flow

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
Contrast: Hadoop MapReduce

Each iteration is a MapReduce Job
Map: Compute gradients per data point
Combine / Reduce: Sum the gradients up
Driver: Update model, re-submit if needed

Huge overhead *per Iteration*
Schedule mappers and Reducers
Distribute code
Read *and* parse data from HDFS

Common claim in the literature: Avoiding these = 30x speedup
Data Management Services
The Two Sides of REEF Services

Evaluator Side
- Services live in the scope of the Evaluator
- They outlive Activities
- Activities get them injected at construction time

Driver Side
- Services provide help with creating their Evaluator-Side Configuration
- Typically, via a declarative specification
Storage Service

Evaluator Side
- Two abstractions: Map and Spool
- Many different implementations
  - Local vs. Remote
  - Tiered Storage
  - Access Patterns

Driver Side
- User declares the abstraction and (expected) access pattern
- Examples
  - Spool, WORM, Don’t care about the order per pass
  - Map, linear insert, zipf-distributed reads
- The Storage Service provides Configurations that bind the abstract interfaces to optimal implementations
Spool: Sequential I/O

Two interfaces
Iterator (pull)
Accumulator (push)

Simple implementations
Buffering, durability semantics are implementation-specific
Driver chooses the appropriate implementation

Example Access Patterns

**Machine Learning**
Write once, iterate often (order is unimportant)
Write once, get differently sampled Iterators back

**MapReduce**
Write once, read once in sorted order
Shuffle: Write sorted runs, read merged data

```java
public interface Spool<T> {
    implements Iterable<T>, Accumulable<T> {
        Iterator<T> iterator();
        Accumulator<T> accumulator();
    }
}

public interface Iterator<T> {
    boolean hasNext();
    T next();
}

public interface Accumulator<T> {
    void add(T t);
    void close();
}
```
Map: Random I/O

Based on Java Map API
Missing non-scalable primitives (such as “iterator()”, “size()”)

Wide range of implementations
As with Spool, semantics and performance vary with underlying implementation

Examples
Local caches / buffer management
Distribution of job state
Key-value stores
Mailbox-style messaging (for small objects)
Network Service

Evaluator Side
- Inter-Activity Messaging
  - Name based
  - Optionally with mailbox semantics
- Group communications
  - Examples: Reduce, Broadcast

Driver Side
- Name Service
- Tracks the physical location of Activities
- Wire-up for group communications
  - E.g. tree construction for Reduce
Key REEF Network Service Properties

Identifier-based communication

all communication is done with identifiers
decouple from physical locations

```
public interface Identifier {}
```

Instant communication
Mailbox communication

decouple from temporal constraints

```
public class NameService {
  ...
  public InetSocketAddress lookup(Identifier i) {
    ...
  }
  ...
}
```
NetworkService ns = new NetworkService(...);
ns.send("Activity B", message);
```java
NetworkService ns = new NetworkService(...);
ns.getMailbox().send("Activity B", message);
```
One-to-one Activity Communication

Low latency messaging
asynchronous send, receive upcall

Bulk data transfer
spool, iterator, accumulator interface

```java
public class NetworkService<T> {
    ...
    public void send(Identifier id, T obj) {
        ...
    }
    ...
    public void handler(Receiver<T> receiver) {
        ...
    }
    public Spool<T> getSpool(Identifier id) {
        ...
    }
    ...
}

public interface Receiver<T> {
    void recv(Identifier id, T obj);
}
```
Group Activity Communication

Multicast, k-hop broadcast
1-to-N

Aggregation tree
N-to-1

Publish/subscribe
M-to-N
State Management Service

Evaluator Side
- Checkpoint Channel
- Single writer
- Returns Memento upon commit
- Access to state when presented with a Memento

Driver Side
- User select check pointing constraints
- Durability level: Memory, Disk, HDFS, WAS, ...
- Lifetime of checkpoints
- Service generates configurations satisfying these requirements
State Management

Manage (save and retrieve) the state of a computation

Objectives:

Fault Tolerance
Preempt. Scheduling Operations
Dynamic Optimizations (skew, multi-tenancy)
State Management

- Fault Tolerance
- Preempt. Scheduling Operations
- Dynamic Optimizations (skew, multi-tenancy)
State Management

Fault Tolerance

App Code

Logging and Recovery Policies and Mechanisms

Preempt. Scheduling Operations

App Code

Preemption Policies

Preemption Mechanisms / Stats Collection

Checkpoint Service

Dynamic Optimizations (skew, multi-tenancy)

App Code

Optimization Policies

Also used in: Apache Yarn/MapReduce
Checkpoint Service API

**public interface** CheckpointService {
  **public** CheckpointWriteChannel create();
  **public** Memento commit(CheckpointWriteChannel ch);
  **public** void abort(CheckpointWriteChannel ch);
  **public** CheckpointReadChannel open(Memento mem);
  **public** boolean delete(Memento mem);
}

**public interface** WritableByteChannel extends Channel {
  **public** int write(ByteBuffer src);
}

**public interface** ReadableByteChannel extends Channel {
  **public** int read(ByteBuffer dst);
}

Atomic, append-only, single-writer, write-once

Support for HDFS and Local Filesystem

Job-level quotas and garbage collection (via Hadoop staging)

(Configured via Tang or directly)
public interface ResumeableActivity<T, M extends Memento> extends Activity<T> {
  public M suspend();
  public T resume(M memento);  
}
Preemption: REEF Control Flow

Client

Job Driver

Activity

Evaluator

services

CheckpointService cs = Tang config;
CheckpointWriteChannel cwc = cs.create();
cwc.write(state);
Memento mem = cs.commit(cwc);
return mem;

public Memento suspend()
{
  
}

Node

Yarn

RM

Name

HDFS

NM

node1

node2

node3

node4
Preemption: REEF Control Flow

public void resume(Memento mem) {
    CheckpointReadChannel crc = cs.open(mem);
    crc.read(...buffer...);
}
Scheduling in Hadoop
(Curino, Douglas, Rao)
(Amoeba paper, SOCC 2012)

Popular schedulers

CapacityScheduler

FairScheduler

Deadline-oriented scheduling

New idea:

Support work-preserving preemption
(via) checkpointing → more than preemption
Dynamic Optimization

Leveraging checkpointing for parsimonious scheduling in MR
Killing Tasks vs. Preemption

33% Improvement

% Complete vs. Time (s)
Adding Preemption to YARN
And Open-Sourcing to Apache

Use of Preemption
Context:
Outdated information
Delayed effects of actions
Multi-actor orchestration

Interesting type of preemption:
RM declarative request
AM binds it to containers

Collaborative application
Policy-based binding for flexible preemption requests

Client

PreemptionMessage {
    Strict { Set<ContainerID> }
    Flexible { Set<ResourceRequest>, Set<ContainerID> }
}

Job1

RM
Scheduler

NodeManager

App Master

Task

NodeManager

Task

NodeManager

Task

NodeManager

Task

NodeManager
Changes throughout YARN

Common Checkpoint Service
WriteChannel cwc = cs.create();
cwc.write(...state...);
CheckpointID cid = cs.commit(cwc);
ReadChannel crc = cs.open(cid);

When can I preempt?
tag safe UDFs or user-saved state

@Preemptable
public class MyReducer{
  ...
}
Contributing to Apache

Engaging with OSS
- talk with active developers
- show early/partial work
- small patches
- ok to leave things unfinished

Contribution:
- Client
- Job1
- RM
- Scheduler
- NodeManager
- App Master
- Task
- NodeManager
- MR-5176
- MR-5189
- MR-5192
- MR-5194
- MR-5197
- MR-5189
- YARN-45
- YARN-567
- YARN-568
- YARN-569
- YARN-5196

Talk with active developers, show early/partial work, small patches, and it's okay to leave things unfinished.
There is More, But Not Today

Configuration Management: Tang
Using Dependency Injection / Inversion of control
Automated checks before spinning up remote machines
Cross-Framework: Java and .NET

Event framework: Wake
Inspired by Rx and Seda: Observer / Observable but with static wire-up
Automated thread management
Remoting: Used for Driver <-> Evaluator messaging
Cross-Framework: Java and .NET
CONCLUSIONS

If you want to implement the next Mahout, consider building on REEF

• Data is the new gold, data mining the new Klondike

• The next generation of data platforms will fuse traditional data management, scale-out systems like Hadoop, and cloud capabilities

• Convergence of analytic toolsets, blurring market boundaries
Collaborations

- AIP
- GSL
- Isotope team
- Galen Hunt’s team (Drawbridge)
- MSR, XCG