Scalding:
Big Data Programming with Scala

Taewook Eom

Data Infrastructure Team
SK planet

taewook@sk.com

2015-01-29

Big Data Processing

hadoop MapReduce

HIVE

CASCADING

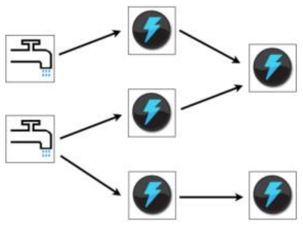
cloudera IMPALA

Spark

TAJO

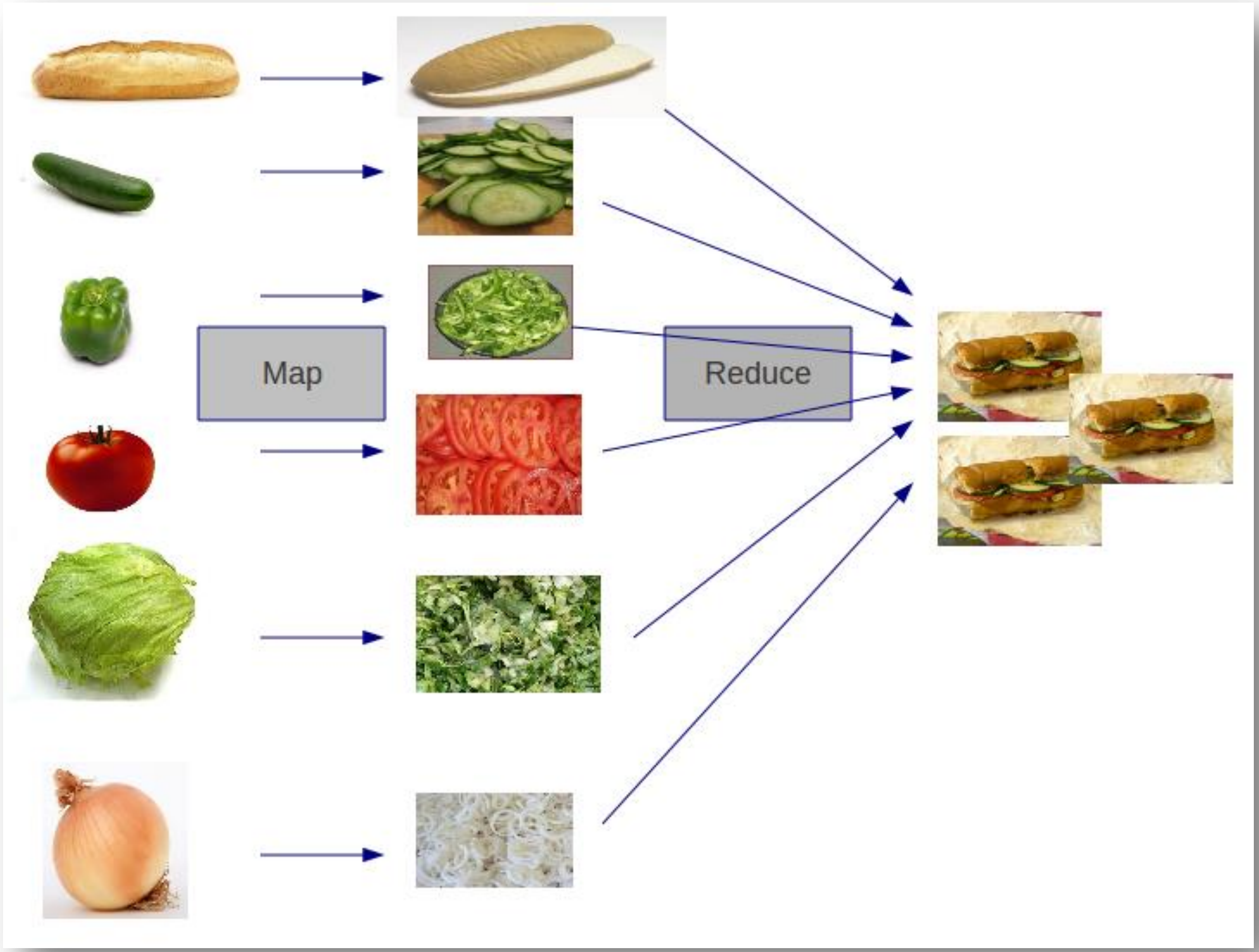
Apache Hama

Apache Storm



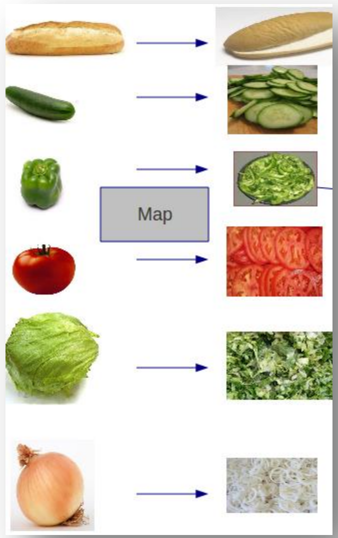
samza

MapReduce, MR, Map-Reduce

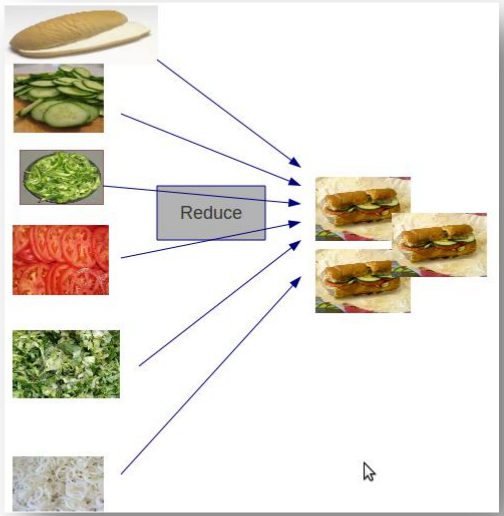


https://twitter.com/brianabelson/status/506933310787186688

Data Processing Pattern with MR



select
function
where(filter)



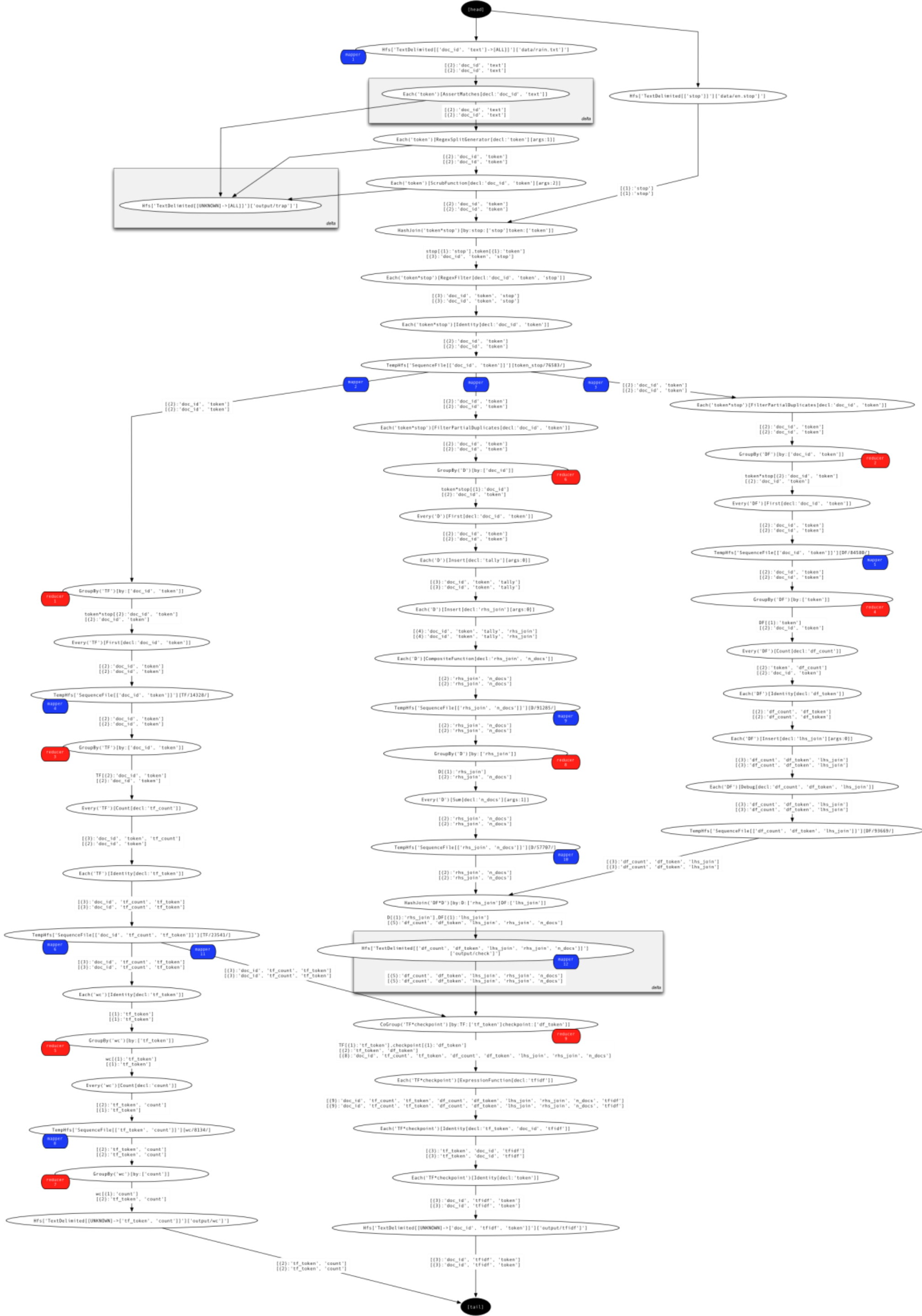
group by
Join
order by
windowing function
analytics function

M = M+
MR = M+RM*
MRMR... = (M+RM*)+

➡ **Workflow** management

Data Workflow = DAG

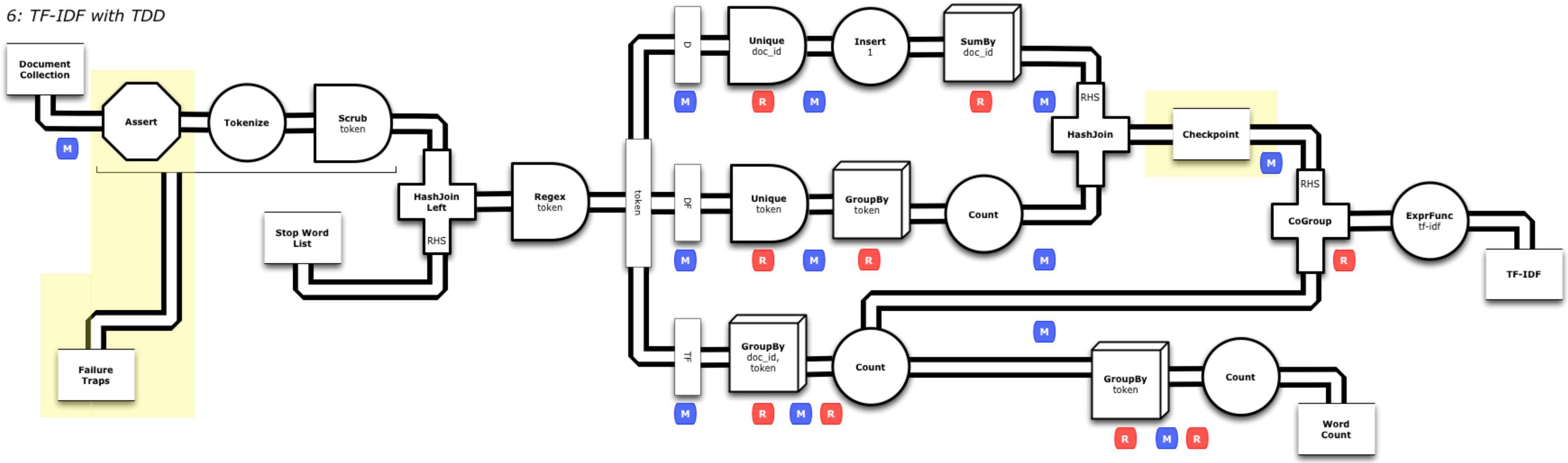
(Directed Acyclic Graph)



http://docs.cascading.org/impatient/impatient6.html

Cascading

http://www.cascading.org/

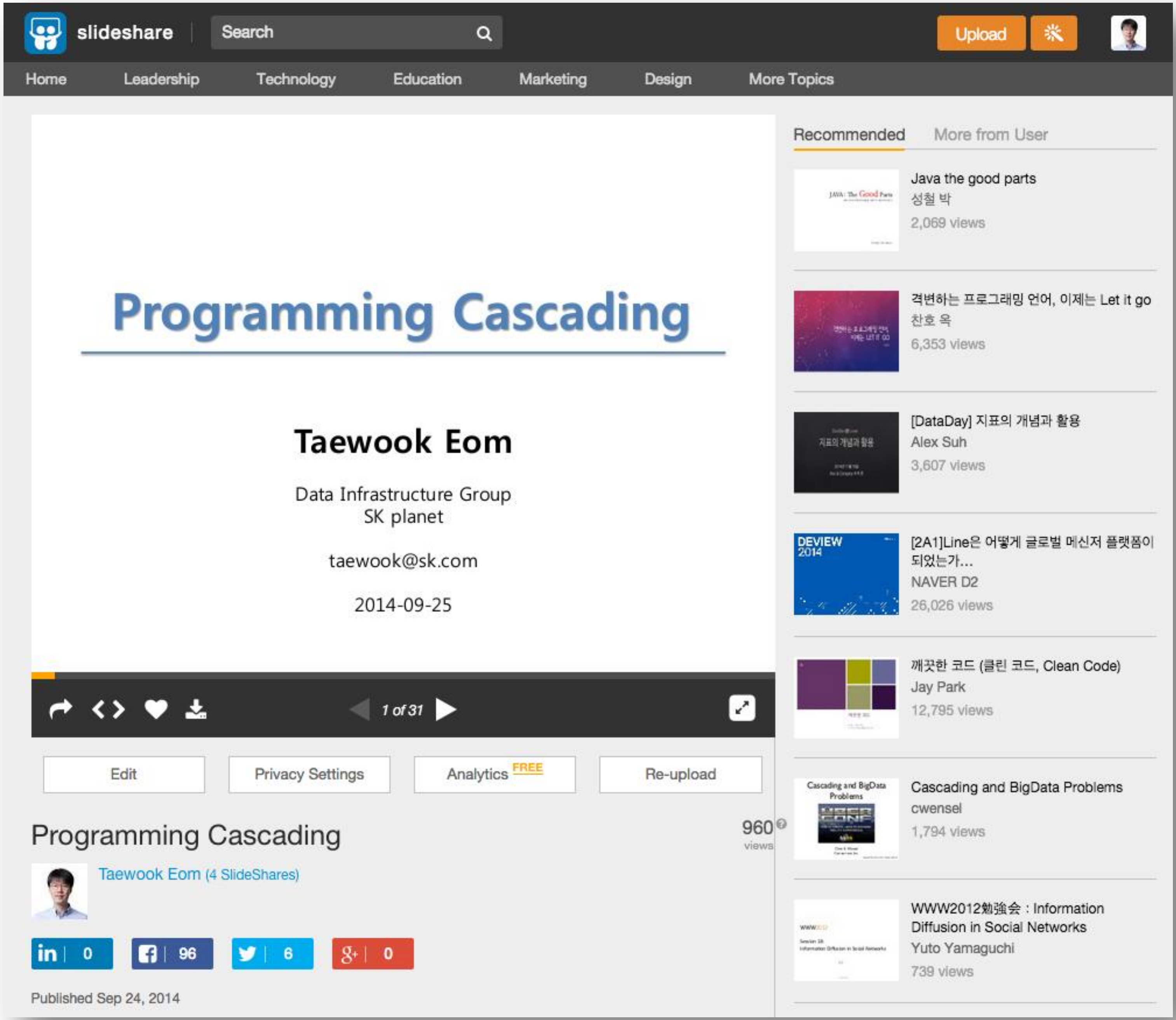


- **Pipe** abstraction = Plumbing
- Operators like **SQL**
- **DAG** based **workflow** management

http://docs.cascading.org/impatient/impatient6.html

http://www.stuartplumbing.com.au/wp-content/uploads/2014/02/pipes_253.jpg http://acpfl.co/wp-content/uploads/2014/11/Plumbers.png
http://sommerdyke.com/wp-content/uploads/2014/11/plumbing3.jpg/ http://grammarchicblog.files.wordpress.com/2013/08/plumber-manchester.jpg

http://www.slideshare.net/taewook/programming-cascading



Programming Cascading

Taewook Eom

Data Infrastructure Group
SK planet

taewook@sk.com

2014-09-25

Programming Cascading

Taewook Eom (4 SlideShares)

960 views

in 0 | 96 | 6 | 0

Published Sep 24, 2014

Java the good parts
성철 박
2,069 views

격변하는 프로그래밍 언어, 이제는 Let it go
찬호 옥
6,353 views

[DataDay] 지표의 개념과 활용
Alex Suh
3,607 views

[2A1]Line은 어떻게 글로벌 메신저 플랫폼이 되었는가...
NAVER D2
26,026 views

깨끗한 코드 (클린 코드, Clean Code)
Jay Park
12,795 views

Cascading and BigData Problems
cwensel
1,794 views

WWW2012勉強会 : Information Diffusion in Social Networks
Yuto Yamaguchi
739 views

Object-Oriented vs. Functional

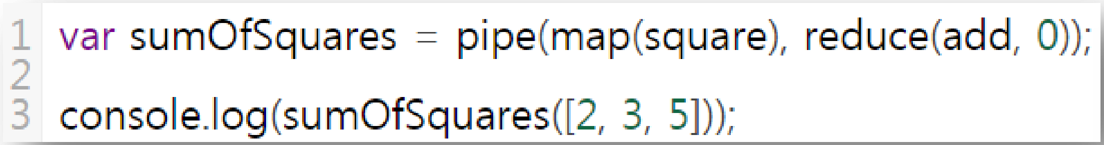
OOP focuses on the **differences in the data**

Data and the **operations** upon it are tightly **coupled**
The central model for abstraction is the **data** itself

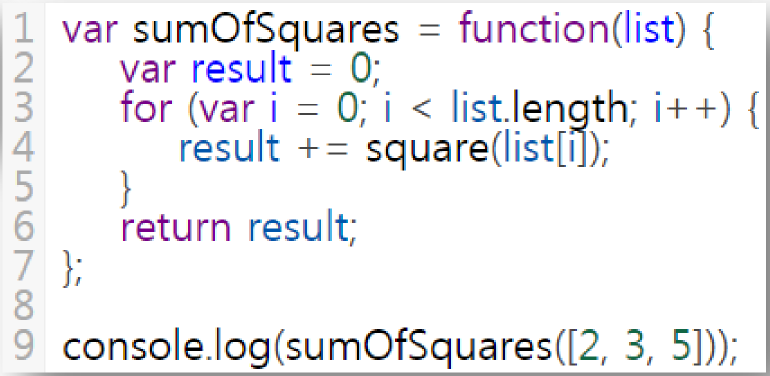
FP concentrates on **consistent data structures**

Data is only loosely coupled to functions
The central model for abstraction is the **function**, not the data structure

FP describe what they want done, **not how** to do it
OOP uses mostly **imperative** techniques



```
var sumOfSquares = pipe(map(square), reduce(add, 0));

console.log(sumOfSquares([2, 3, 5]));
```



```
var sumOfSquares = function(list) {
    var result = 0;
    for (var i = 0; i < list.length; i++) {
        result += square(list[i]);
    }
    return result;
};

console.log(sumOfSquares([2, 3, 5]));
```

http://scott.sauyet.com/Javascript/Talk/2014/01/FuncProgTalk/#slide-10 http://scott.sauyet.com/Javascript/Talk/2014/01/FuncProgTalk/#slide-13

Data Processing, Functional Programming

SQL

uses a **consistent data structure** (table: rows x cols)
uses **functions** that can be **combined**
is **declarative** not imperative

Data is Immutable

➔ Transformable

by Composable Functions

http://scott.sauyet.com/Javascript/Talk/2014/01/FuncProgTalk/#slide-6

Why Scala?

http://www.scala-lang.org/

Scalable **La**nguage
➔ **Big Data**

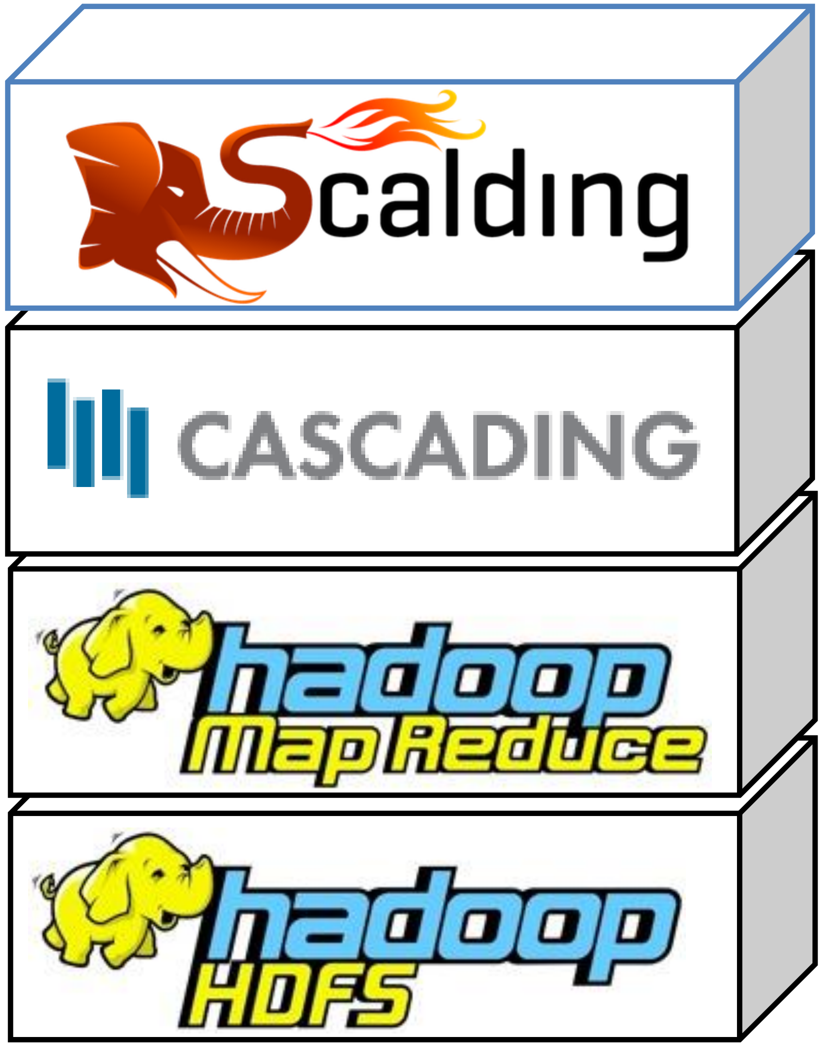
Seamless **Java Interop**
➔ **Hadoop** runs on the **JVM**

Functional
➔ **Data Processing**

REPL(Read-Evaluate-Print Loop)
➔ **Interactive** data analysis

Scalding

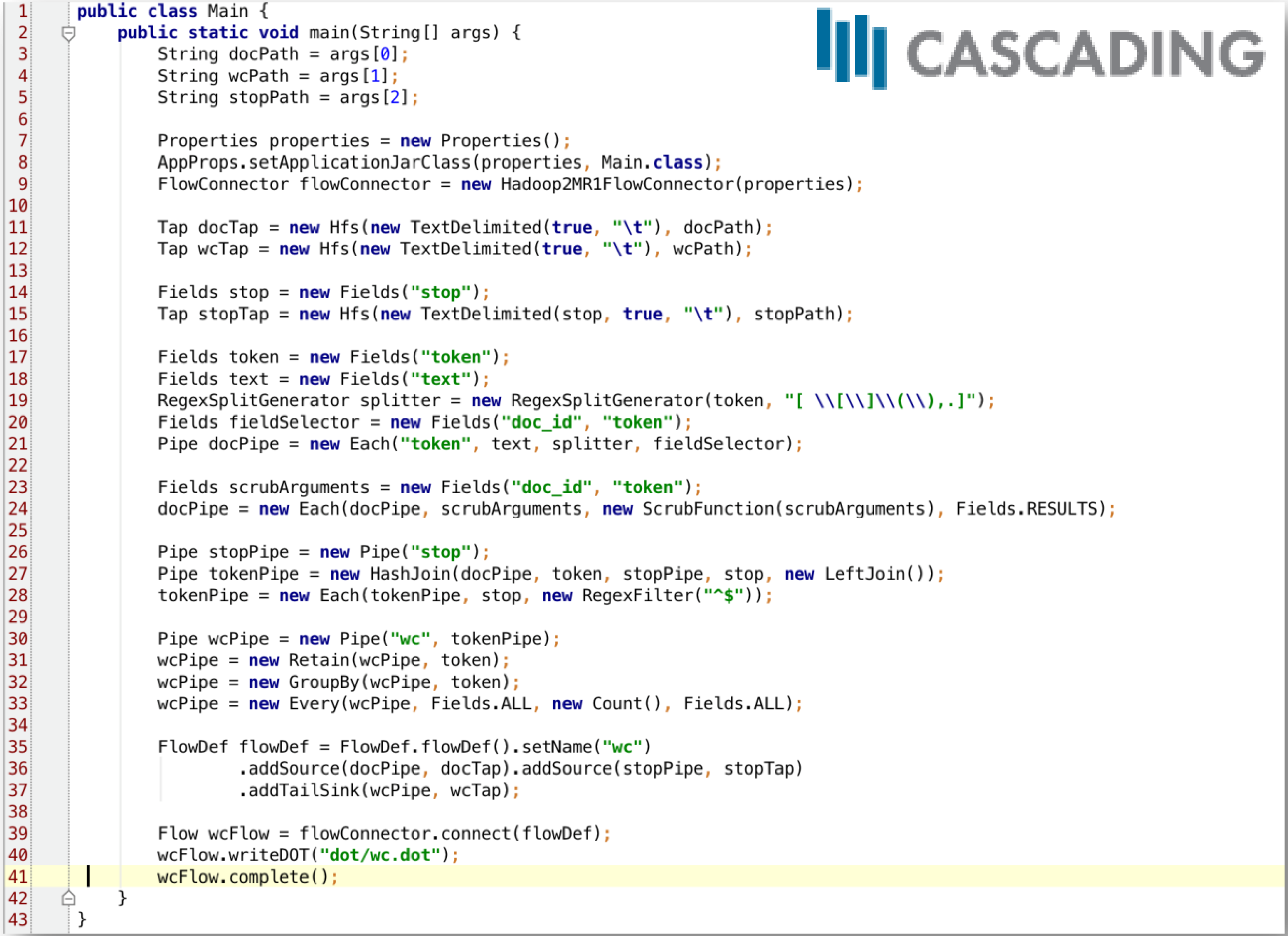
https://github.com/twitter/scalding



Scala DSL for Cascading

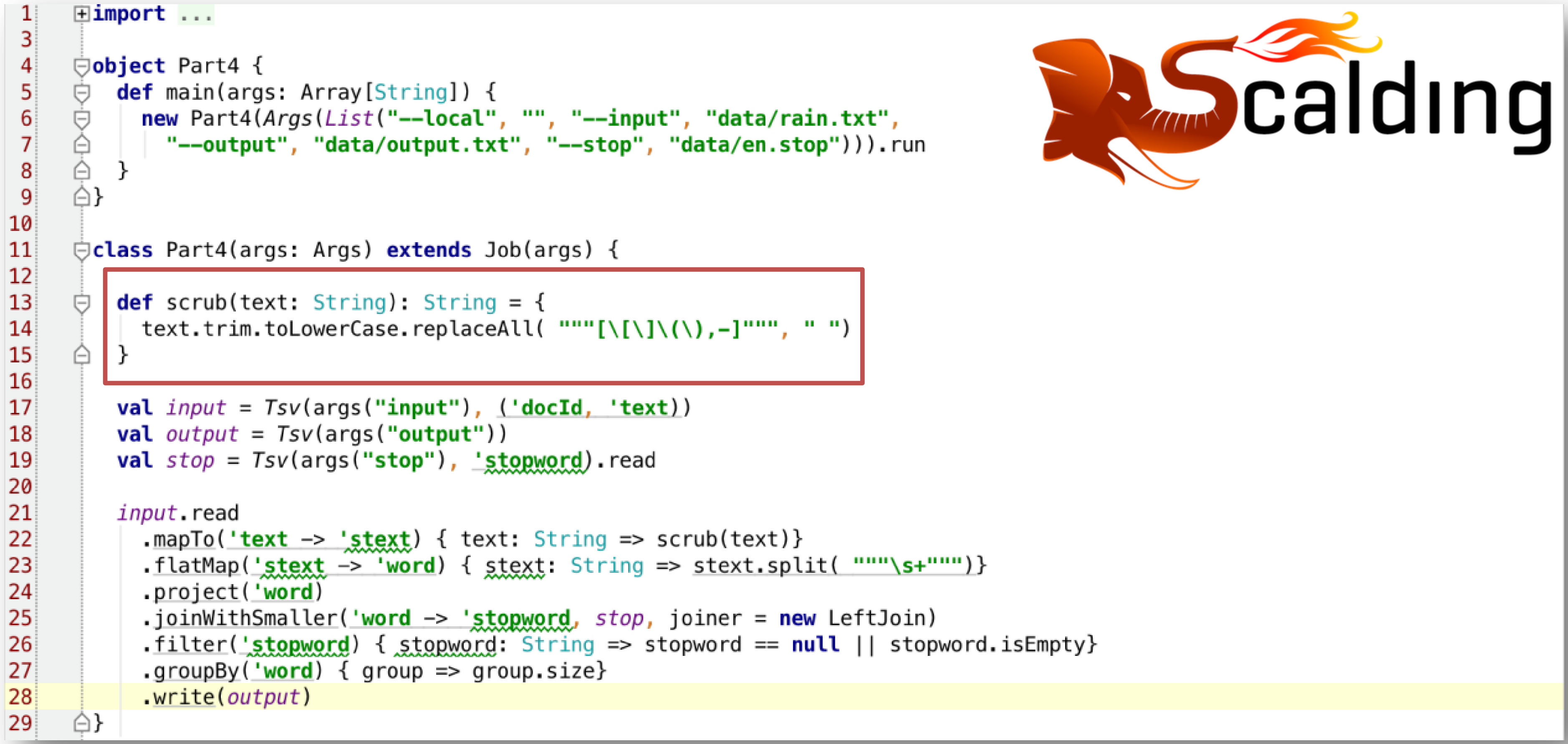
Simple and **concise** syntax

maintained by Twitter



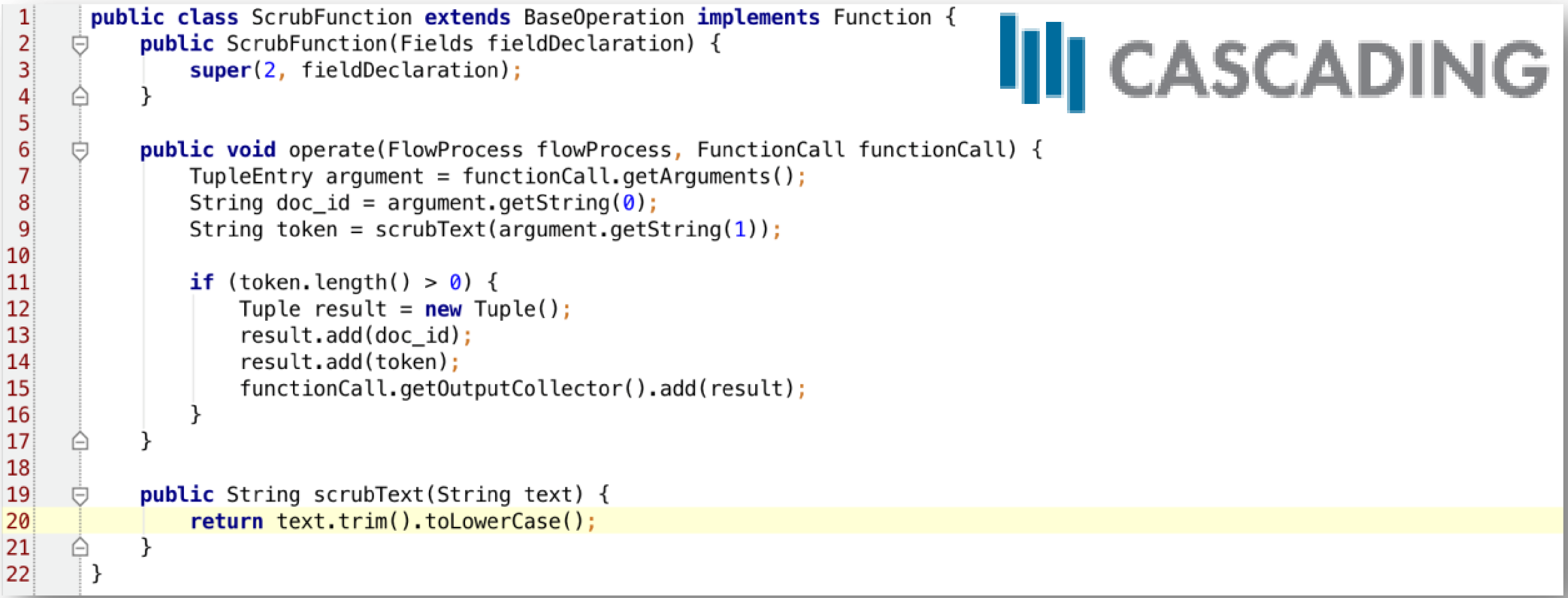
```java
public class Main {
    public static void main(String[] args) {
        String docPath = args[0];
        String wcPath = args[1];
        String stopPath = args[2];

        Properties properties = new Properties();
        AppProps.setApplicationJarClass(properties, Main.class);
        FlowConnector flowConnector = new Hadoop2MR1FlowConnector(properties);

        Tap docTap = new Hfs(new TextDelimited(true, "\t"), docPath);
        Tap wcTap = new Hfs(new TextDelimited(true, "\t"), wcPath);

        Fields stop = new Fields("stop");
        Tap stopTap = new Hfs(new TextDelimited(stop, true, "\t"), stopPath);

        Fields token = new Fields("token");
        Fields text = new Fields("text");
        RegexSplitGenerator splitter = new RegexSplitGenerator(token, "[ \\[\\]\\(\\),.]");
        Fields fieldSelector = new Fields("doc_id", "token");
        Pipe docPipe = new Each("token", text, splitter, fieldSelector);

        Fields scrubArguments = new Fields("doc_id", "token");
        docPipe = new Each(docPipe, scrubArguments, new ScrubFunction(scrubArguments), Fields.RESULTS);

        Pipe stopPipe = new Pipe("stop");
        Pipe tokenPipe = new HashJoin(docPipe, token, stopPipe, stop, new LeftJoin());
        tokenPipe = new Each(tokenPipe, stop, new RegexFilter("^$"));

        Pipe wcPipe = new Pipe("wc", tokenPipe);
        wcPipe = new Retain(wcPipe, token);
        wcPipe = new GroupBy(wcPipe, token);
        wcPipe = new Every(wcPipe, Fields.ALL, new Count(), Fields.ALL);

        FlowDef flowDef = FlowDef.flowDef().setName("wc")
                .addSource(docPipe, docTap).addSource(stopPipe, stopTap)
                .addTailSink(wcPipe, wcTap);

        Flow wcFlow = flowConnector.connect(flowDef);
        wcFlow.writeDOT("dot/wc.dot");
        wcFlow.complete();
    }
}
```

https://github.com/Cascading/Impatient/blob/master/part4/src/main/java/impatient/Main.java



```scala
import ...

object Part4 {
  def main(args: Array[String]) {
    new Part4(Args(List("--local", "", "--input", "data/rain.txt",
      "--output", "data/output.txt", "--stop", "data/en.stop"))).run
  }
}

class Part4(args: Args) extends Job(args) {

  def scrub(text: String): String = {
    text.trim.toLowerCase.replaceAll( """[\[\]\(\),-]""", " ")
  }

  val input = Tsv(args("input"), ('docId, 'text))
  val output = Tsv(args("output"))
  val stop = Tsv(args("stop"), 'stopword).read

  input.read
    .mapTo('text -> 'stext) { text: String => scrub(text)}
    .flatMap('stext -> 'word) { stext: String => stext.split( """\s+""")}
    .project('word)
    .joinWithSmaller('word -> 'stopword, stop, joiner = new LeftJoin)
    .filter('stopword) { stopword: String => stopword == null || stopword.isEmpty}
    .groupBy('word) { group => group.size}
    .write(output)
}
```

https://github.com/sujitpal/hia-examples/blob/master/scala/scalding-impatient/src/main/scala/com/mycompany/impatient/Part4.scala



```
public class ScrubFunction extends BaseOperation implements Function {
    public ScrubFunction(Fields fieldDeclaration) {
        super(2, fieldDeclaration);
    }

    public void operate(FlowProcess flowProcess, FunctionCall functionCall) {
        TupleEntry argument = functionCall.getArguments();
        String doc_id = argument.getString(0);
        String token = scrubText(argument.getString(1));

        if (token.length() > 0) {
            Tuple result = new Tuple();
            result.add(doc_id);
            result.add(token);
            functionCall.getOutputCollector().add(result);
        }
    }

    public String scrubText(String text) {
        return text.trim().toLowerCase();
    }
}
```

UDF(User-defined Function)

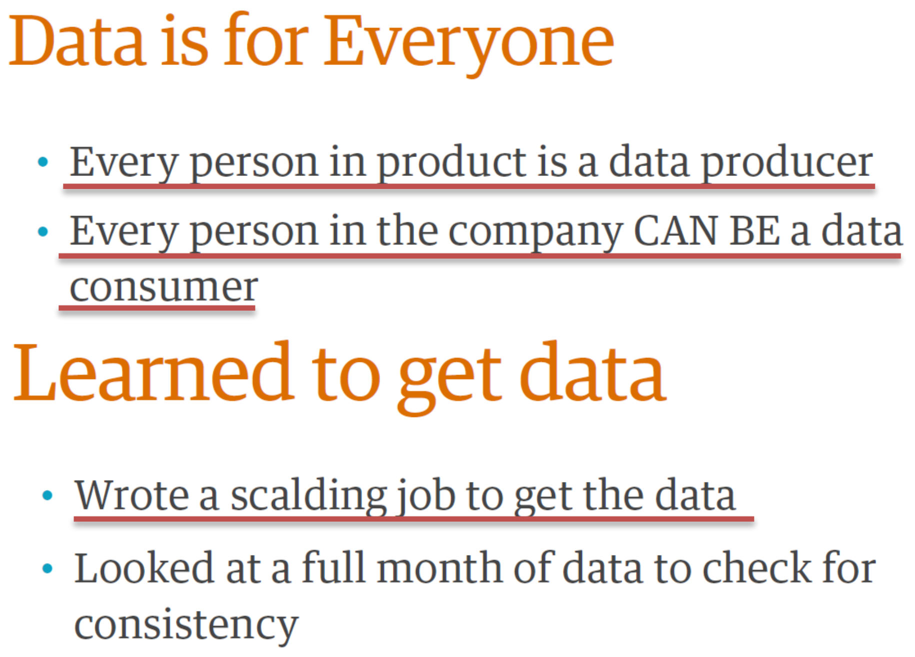
https://github.com/Cascading/Impatient/blob/master/part4/src/main/java/impatient/ScrubFunction.java

**“If you need to write UDF’s all the time,
something is wrong with you.”**

- Various authors of non-scalding frameworks
who happened to be completely WRONG

Etsy's Data-Driven Culture

At **Etsy**, it's not just **engineers** who **write** and **deploy code** – our **designers** and **product managers** regularly do too.

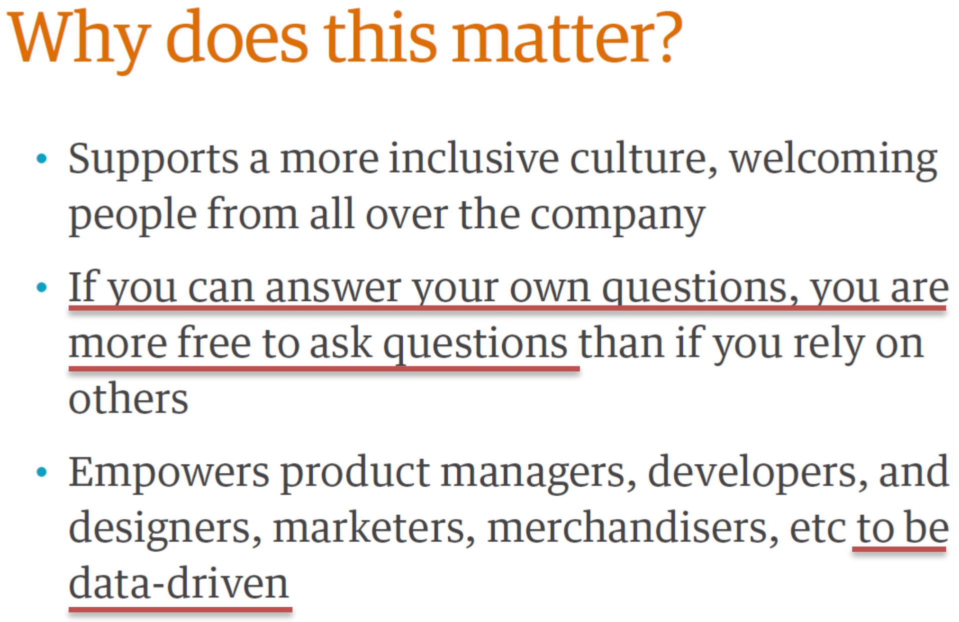


Data is for Everyone

- Every person in product is a data producer
- Every person in the company CAN BE a data consumer

Learned to get data

- Wrote a scalding job to get the data
- Looked at a full month of data to check for consistency

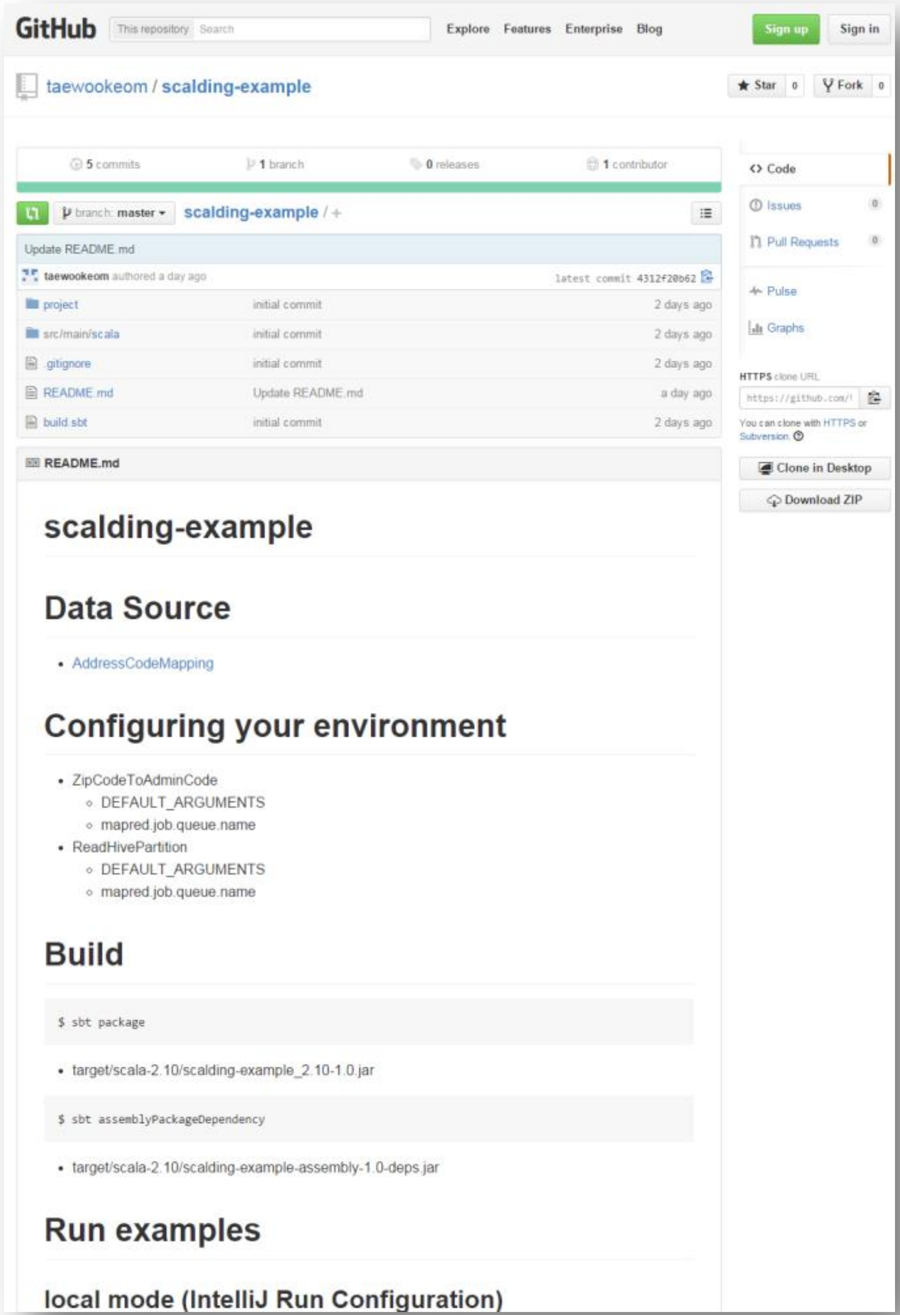


Why does this matter?

- Supports a more inclusive culture, welcoming people from all over the company
- If you can answer your own questions, you are more free to ask questions than if you rely on others
- Empowers product managers, developers, and designers, marketers, merchandisers, etc to be data-driven

https://codeascraft.com/2014/12/22/engineering-rotation/ We Invite Everyone at Etsy to Do an Engineering Rotation: Here's why
http://strataconf.com/strataeu2014/public/schedule/detail/37250 Data Consumers are better Data Producers

https://github.com/taewookeom/scalding-example



taewookeom / scalding-example

5 commits · 1 branch · 0 releases · 1 contributor

branch: master · scalding-example / +

Update README.md

taewookeom authored a day ago · latest commit 4312f20b62

project	initial commit	2 days ago
src/main/scala	initial commit	2 days ago
.gitignore	initial commit	2 days ago
README.md	Update README.md	a day ago
build.sbt	initial commit	2 days ago

HTTPS clone URL

https://github.com/!

You can clone with HTTPS or Subversion.

README.md

scalding-example

Data Source

- AddressCodeMapping

Configuring your environment

- ZipCodeToAdminCode
  - DEFAULT_ARGUMENTS
  - mapred.job.queue.name
- ReadHivePartition
  - DEFAULT_ARGUMENTS
  - mapred.job.queue.name

Build

```
$ sbt package
```

- target/scala-2.10/scalding-example_2.10-1.0.jar

```
$ sbt assemblyPackageDependency
```

- target/scala-2.10/scalding-example-assembly-1.0-deps.jar

Run examples

local mode (IntelliJ Run Configuration)

SBT Build script
- build.sbt, project/plugins.sbt
- libraryDependencies
- Main-Class in META-INF/MANIFEST.MF

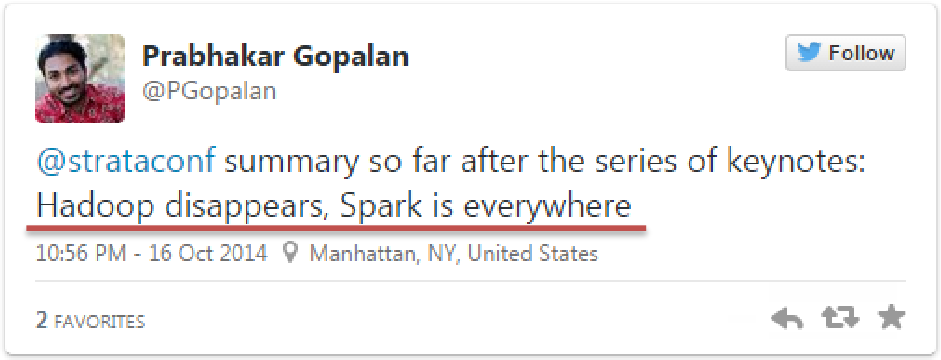
Splitting **project** and **deps JARs**

Run **command** and **arguments**

Next Try

https://spark.apache.org/

Apache Spark™ is a fast and general engine for large-scale data processing.



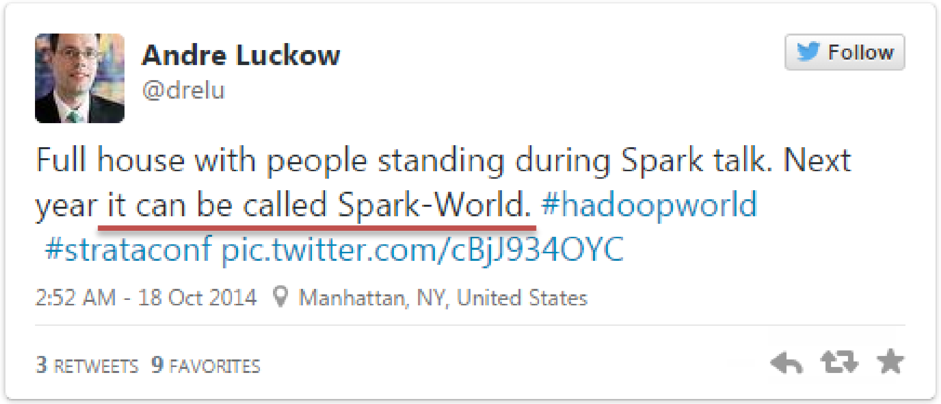
Prabhakar Gopalan
@PGopalan

Follow

@strataconf summary so far after the series of keynotes: Hadoop disappears, Spark is everywhere

10:56 PM - 16 Oct 2014 Manhattan, NY, United States

2 FAVORITES



Andre Luckow
@drelu

Follow

Full house with people standing during Spark talk. Next year it can be called Spark-World. #hadoopworld #strataconf pic.twitter.com/cBjJ934OYC

2:52 AM - 18 Oct 2014 Manhattan, NY, United States

3 RETWEETS 9 FAVORITES

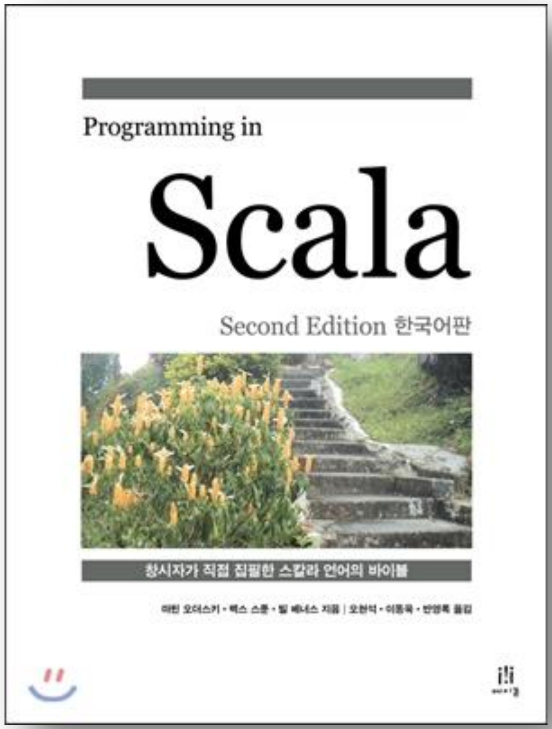
https://twitter.com/PGopalan/status/522747857288183808

https://twitter.com/drelu/status/523169685815042049

Questions?

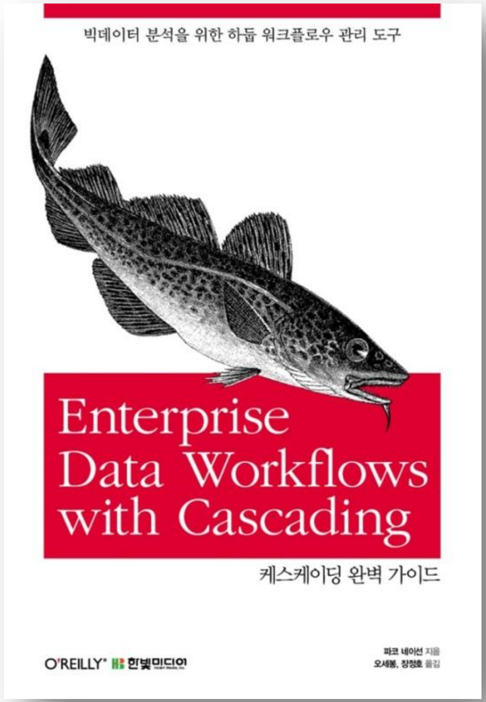
Questions.foreach(answer(_))

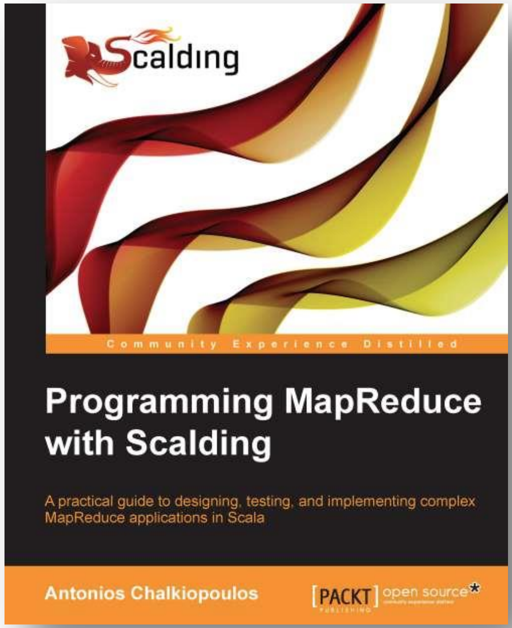
Learning Scala



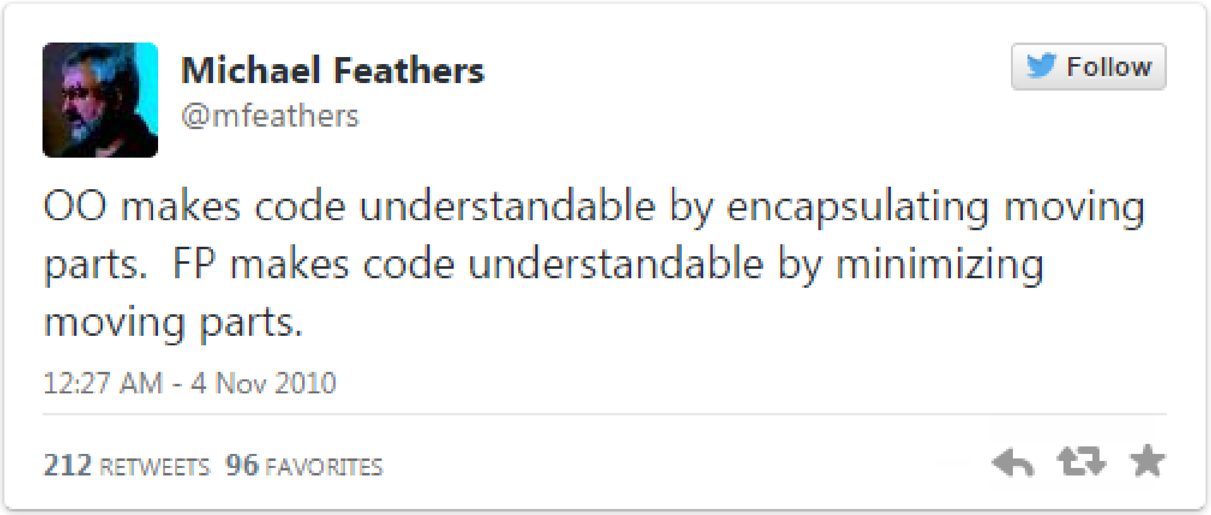
http://www.slideshare.net/deview/a4de-view2012-scalamichinisougu Scala, 미지와의 조우
http://www.slideshare.net/kthcorp/scala-15041890 꽃보다 Scala http://goo.gl/O382Fh
https://twitter.github.io/scala_school/ko/index.html 스칼라 학교!
http://refcardz.dzone.com/refcardz/scala Refcardz: Getting Started with Scala
http://wrobstory.gitbooks.io/python-to-scala/ Python To Scala
http://mbonaci.github.io/scala/ Java developer's Scala cheatsheet

Learning Scalding


빅데이터 분석을 위한 하둡 워크플로우 관리 도구
Enterprise Data Workflows with Cascading
케스케이딩 완벽 가이드
O'REILLY 한빛미디어


Scalding
Community Experience Distilled
Programming MapReduce with Scalding
A practical guide to designing, testing, and implementing complex MapReduce applications in Scala
Antonios Chalkiopoulos
[PACKT] open source

http://docs.cascading.org/tutorials/scalding-data-processing/
https://github.com/twitter/scalding/wiki/Getting-Started
https://github.com/twitter/scalding/wiki/Fields-based-API-Reference
https://github.com/twitter/scalding/tree/master/tutorial
https://github.com/scalding-io/ProgrammingWithScalding
http://sujitpal.blogspot.kr/2012/08/scalding-for-impatient.html
https://github.com/snowplow/scalding-example-project



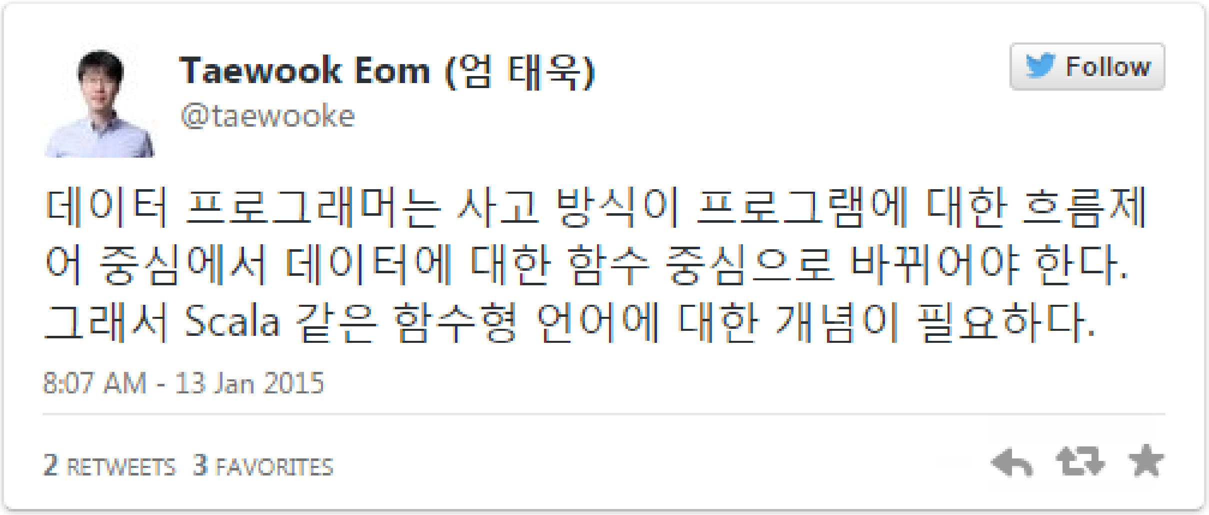
Michael Feathers
@mfeathers

OO makes code understandable by encapsulating moving parts. FP makes code understandable by minimizing moving parts.

12:27 AM - 4 Nov 2010

212 RETWEETS 96 FAVORITES

https://twitter.com/mfeathers/status/29581296216



Taewook Eom (엄 태욱)
@taewooke

데이터 프로그래머는 사고 방식이 프로그램에 대한 흐름제어 중심에서 데이터에 대한 함수 중심으로 바뀌어야 한다. 그래서 Scala 같은 함수형 언어에 대한 개념이 필요하다.

8:07 AM - 13 Jan 2015

2 RETWEETS 3 FAVORITES

https://twitter.com/taewooke/status/554776724290813953