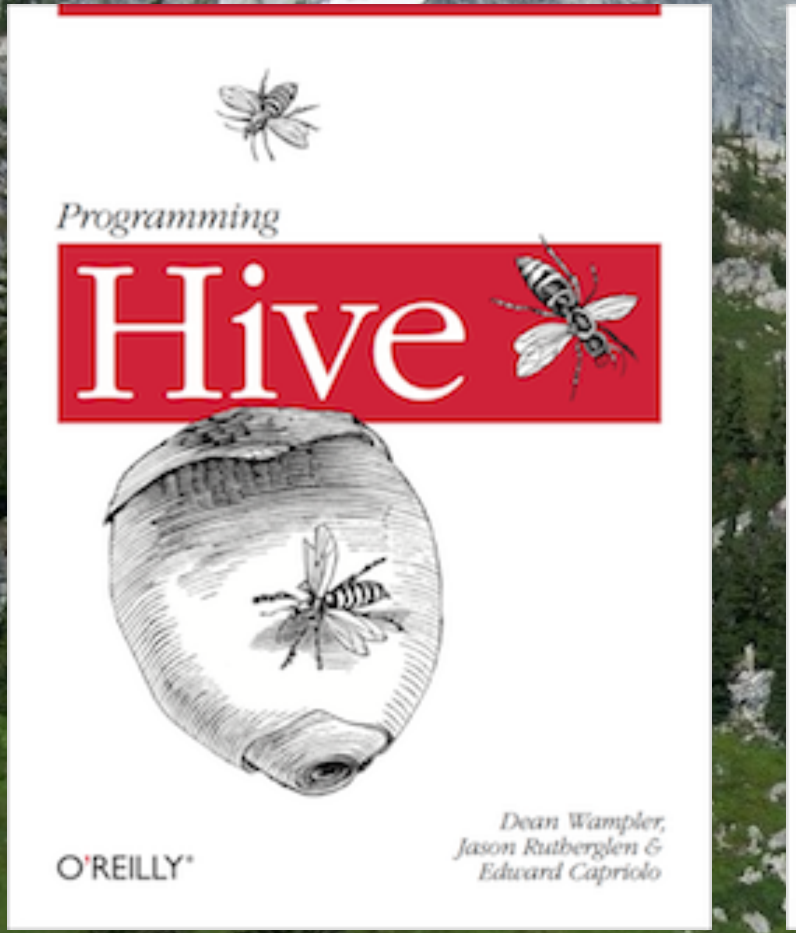
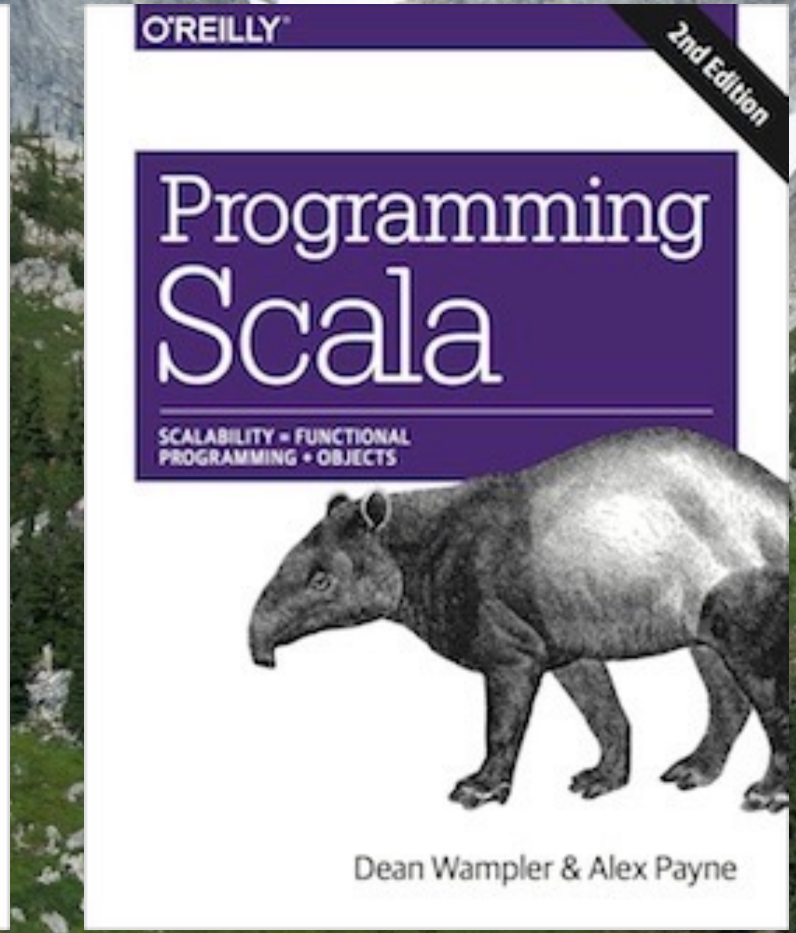
Data Science at Scale with Spark

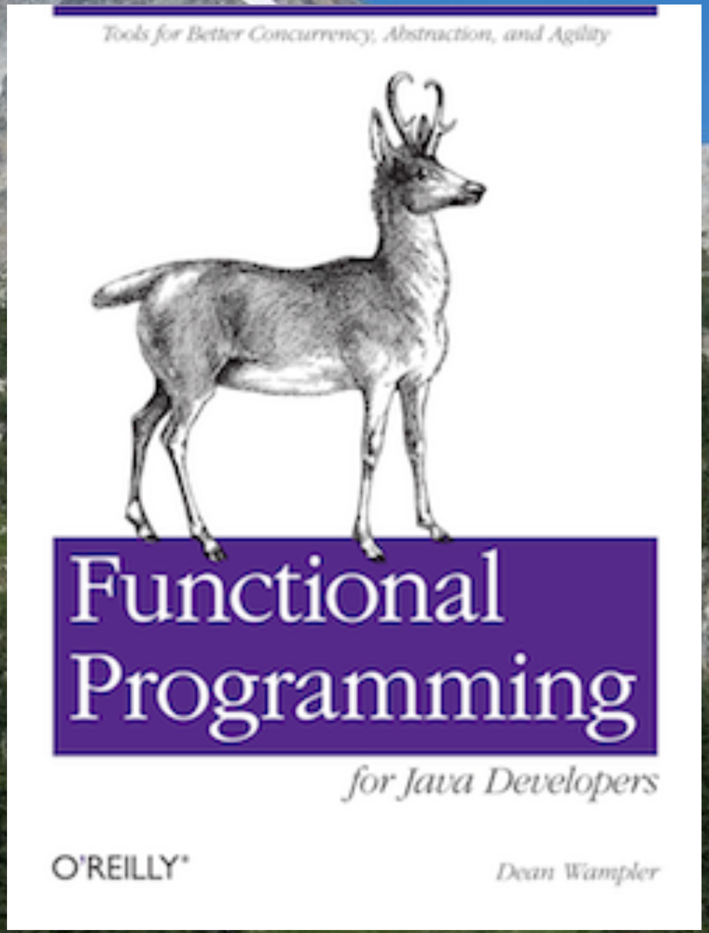
dean.wampler@lightbend.com
polyglotprogramming.com/talks

Lightbend

<shameless>
 <plug>


Programming
Hive
O'REILLY
Dean Wampler, Jason Rutherglen & Edward Capriolo


O'REILLY
2nd Edition
Programming Scala
SCALABILITY = FUNCTIONAL PROGRAMMING + OBJECTS
Dean Wampler & Alex Payne


Tools for Better Concurrency, Abstraction, and Agility
Functional Programming
for Java Developers
O'REILLY
Dean Wampler

 </plug>
</shameless>

“Trolling the Hadoop community since 2012...”

NE Scala Symposium
@deanwampler
March 9, 2012

Why Big Data Needs to Be Functional

Why the JVM?

The JVM

Java
eclipse
IntelliJIDEA

Akka
Breeze
Algebird
Spire & Cats
Axle
...

Big Data Ecosystem

Spark

Scalding

Apache Kafka
A high-throughput distributed messaging system.

hadoop

H$_2$O.ai

Cassandra

STORM

The Apache Software Foundation
http://www.apache.org/

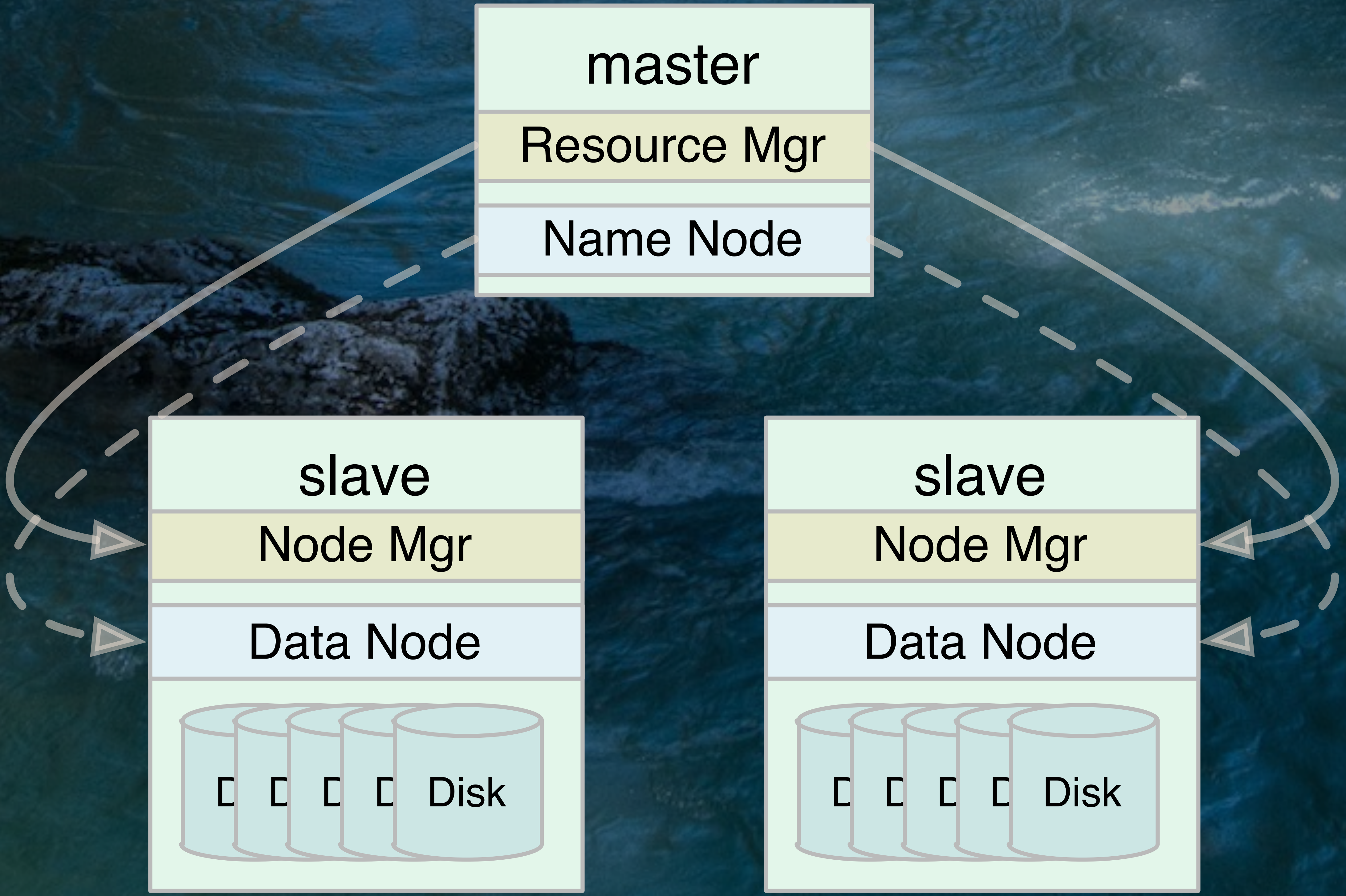
Apache Solr

samza

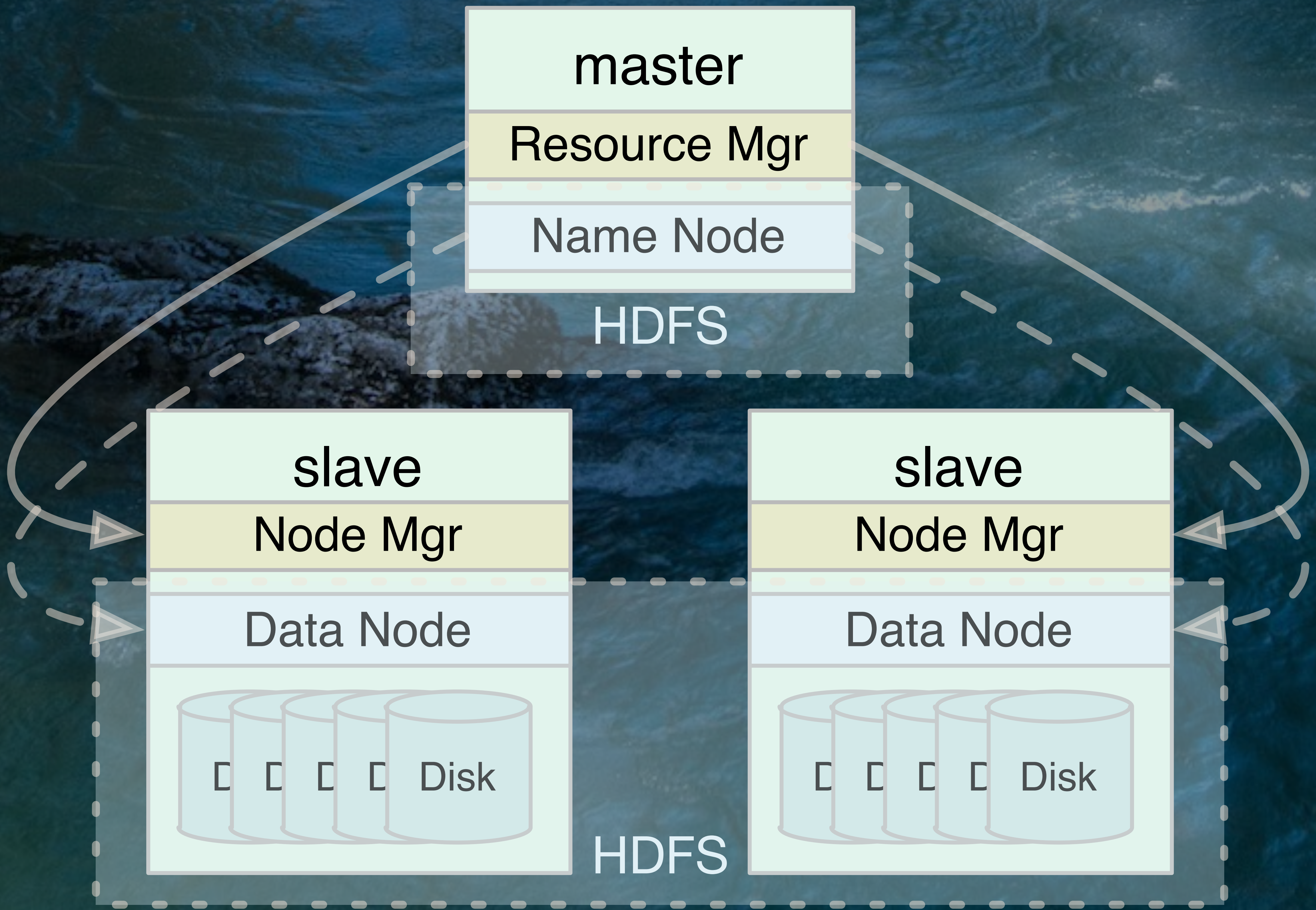
Hadoop

Hadoop

Hadoop


master
Resource Mgr
Name Node
slave
Node Mgr
Data Node
D
D
D
D
Disk
slave
Node Mgr
Data Node
D
D
D
D
Disk

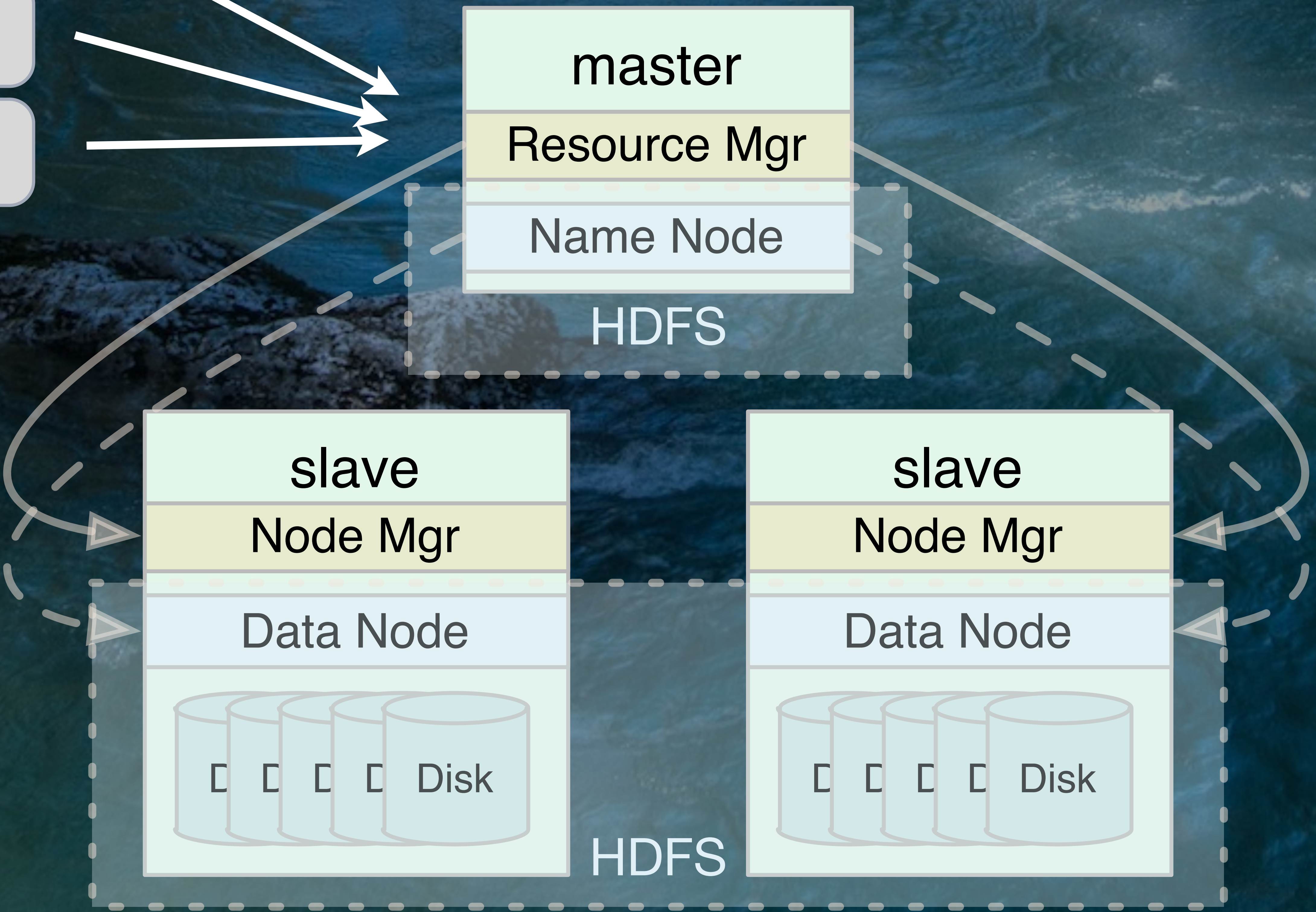
Hadoop


master
Resource Mgr
Name Node
HDFS
slave
Node Mgr
Data Node
D
D
D
D
Disk
slave
Node Mgr
Data Node
D
D
D
D
Disk
HDFS

MapReduce Job

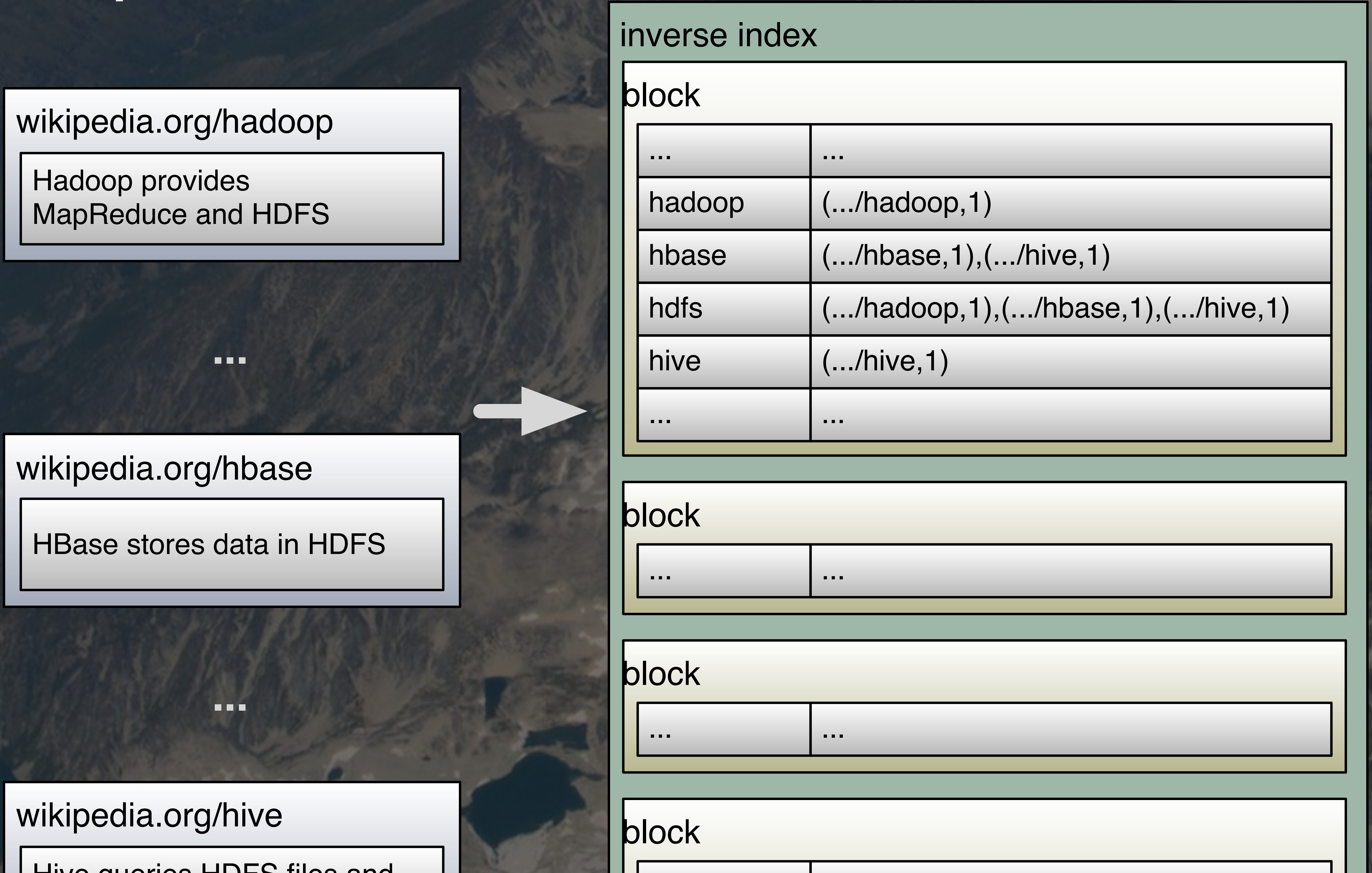
MapReduce Job

MapReduce Job



MapReduce

Example: Inverted Index



wikipedia.org/hadoop

Hadoop provides MapReduce and HDFS

...

wikipedia.org/hbase

HBase stores data in HDFS

...

wikipedia.org/hive

Hive queries HDFS files and

inverse index

block

...	...
hadoop	(.../hadoop,1)
hbase	(.../hbase,1),(.../hive,1)
hdfs	(.../hadoop,1),(.../hbase,1),(.../hive,1)
hive	(.../hive,1)
...	...

block

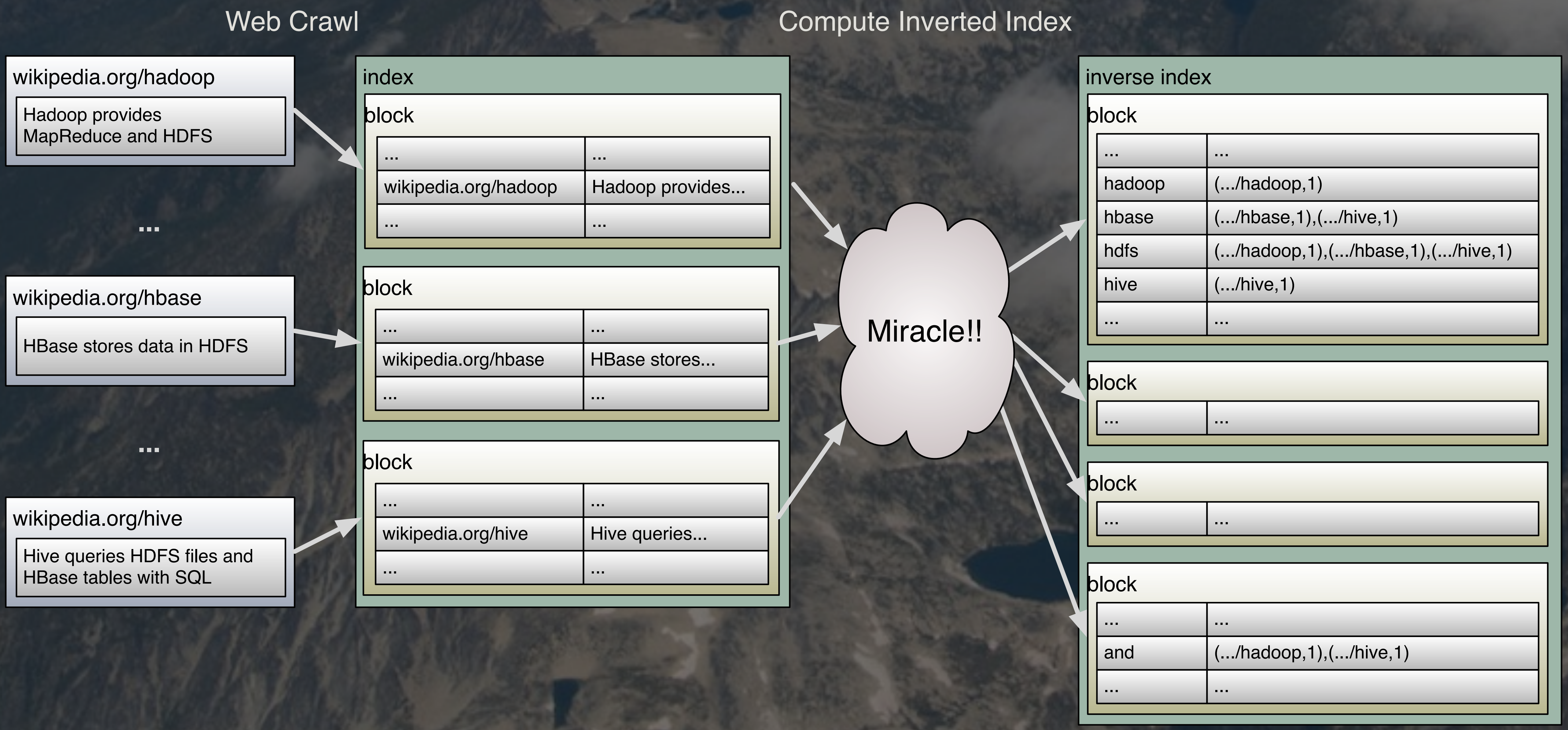
...	...

block

...	...

block

Example: Inverted Index



Web Crawl

wikipedia.org/hadoop

Hadoop provides MapReduce and HDFS

...

wikipedia.org/hbase

HBase stores data in HDFS

...

wikipedia.org/hive

Hive queries HDFS files and HBase tables with SQL

index

block

...	...
wikipedia.org/hadoop	Hadoop provides...
...	...

block

...	...
wikipedia.org/hbase	HBase stores...
...	...

block

...	...
wikipedia.org/hive	Hive queries...
...	...

Compute Inverted Index

Miracle!!

inverse index

block

...	...
hadoop	(.../hadoop,1)
hbase	(.../hbase,1),(.../hive,1)
hdfs	(.../hadoop,1),(.../hbase,1),(.../hive,1)
hive	(.../hive,1)
...	...

block

...	...

block

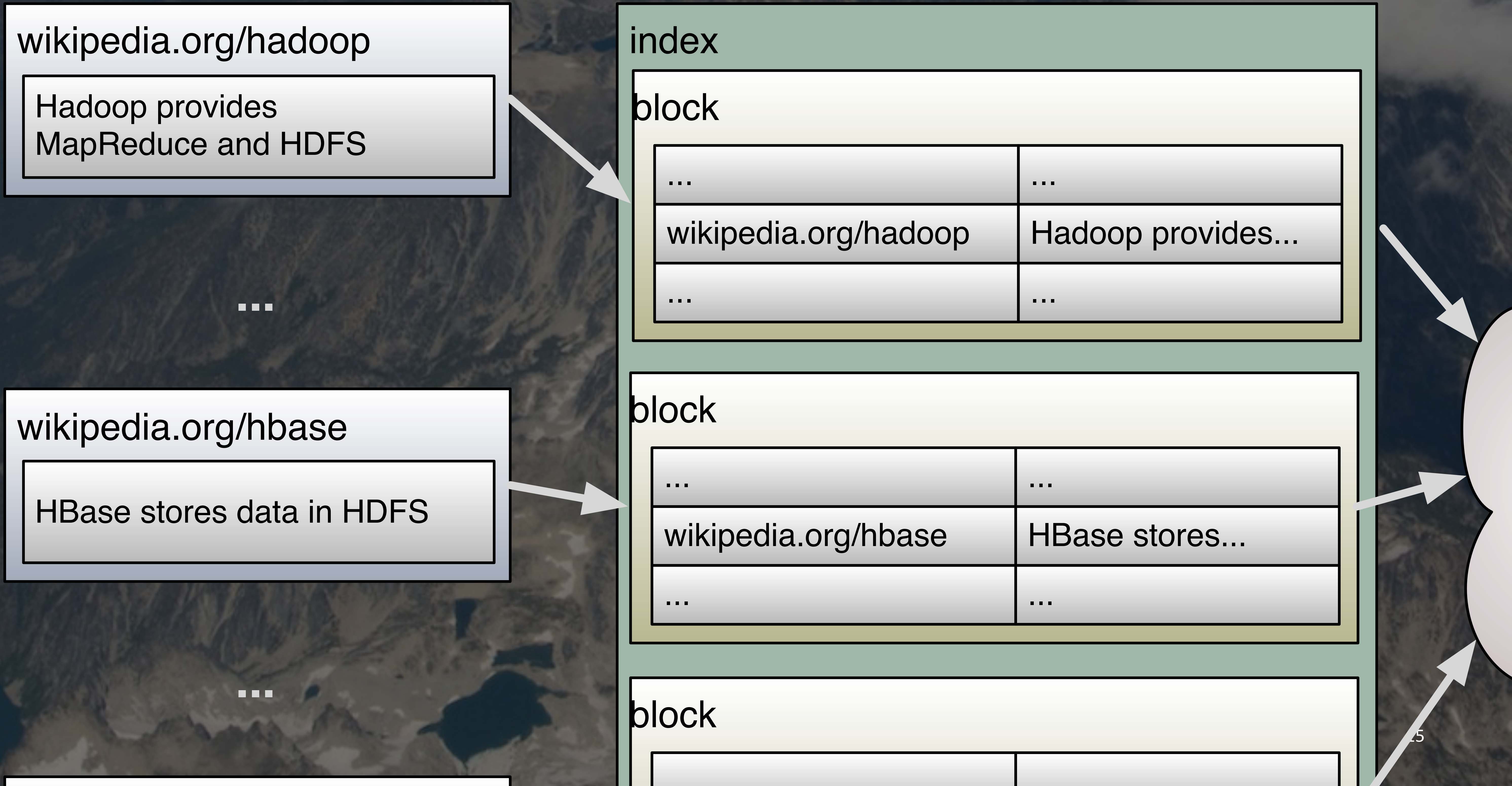
...	...

block

...	...
and	(.../hadoop,1),(.../hive,1)
...	...

Web Crawl

Compu



wikipedia.org/hadoop

Hadoop provides MapReduce and HDFS

...

wikipedia.org/hbase

HBase stores data in HDFS

...

index

block

...	...
wikipedia.org/hadoop	Hadoop provides...
...	...

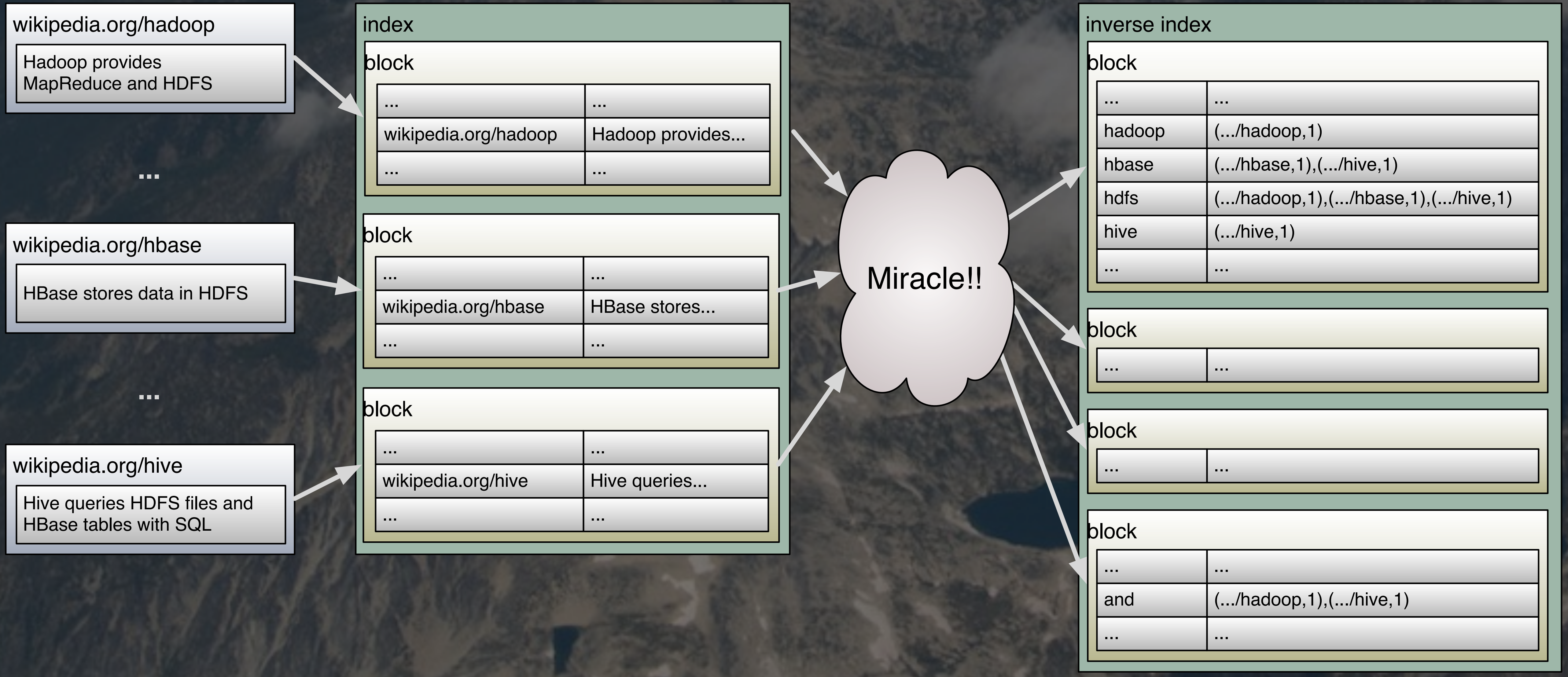
block

...	...
wikipedia.org/hbase	HBase stores...
...	...

block

Web Crawl

Compute Inverted Index



wikipedia.org/hadoop

Hadoop provides MapReduce and HDFS

...

wikipedia.org/hbase

HBase stores data in HDFS

...

wikipedia.org/hive

Hive queries HDFS files and HBase tables with SQL

index

block

...	...
wikipedia.org/hadoop	Hadoop provides...
...	...

block

...	...
wikipedia.org/hbase	HBase stores...
...	...

block

...	...
wikipedia.org/hive	Hive queries...
...	...

inverse index

block

...	...
hadoop	(.../hadoop,1)
hbase	(.../hbase,1),(.../hive,1)
hdfs	(.../hadoop,1),(.../hbase,1),(.../hive,1)
hive	(.../hive,1)
...	...

block

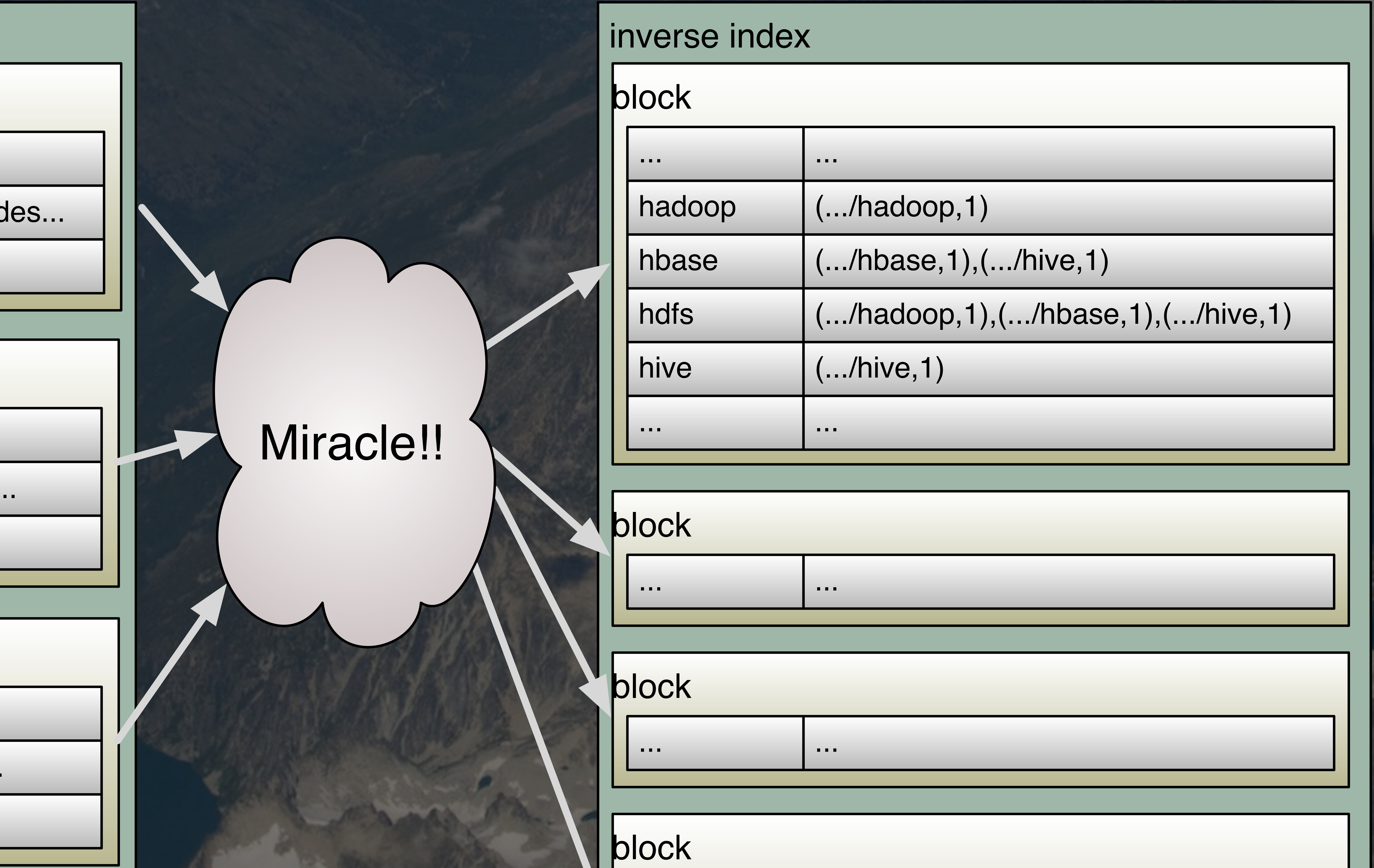
...	...

block

...	...

block

...	...
and	(.../hadoop,1),(.../hive,1)
...	...



des...

..

.

Miracle!!

inverse index

block

...	...
hadoop	(.../hadoop,1)
hbase	(.../hbase,1),(.../hive,1)
hdfs	(.../hadoop,1),(.../hbase,1),(.../hive,1)
hive	(.../hive,1)
...	...

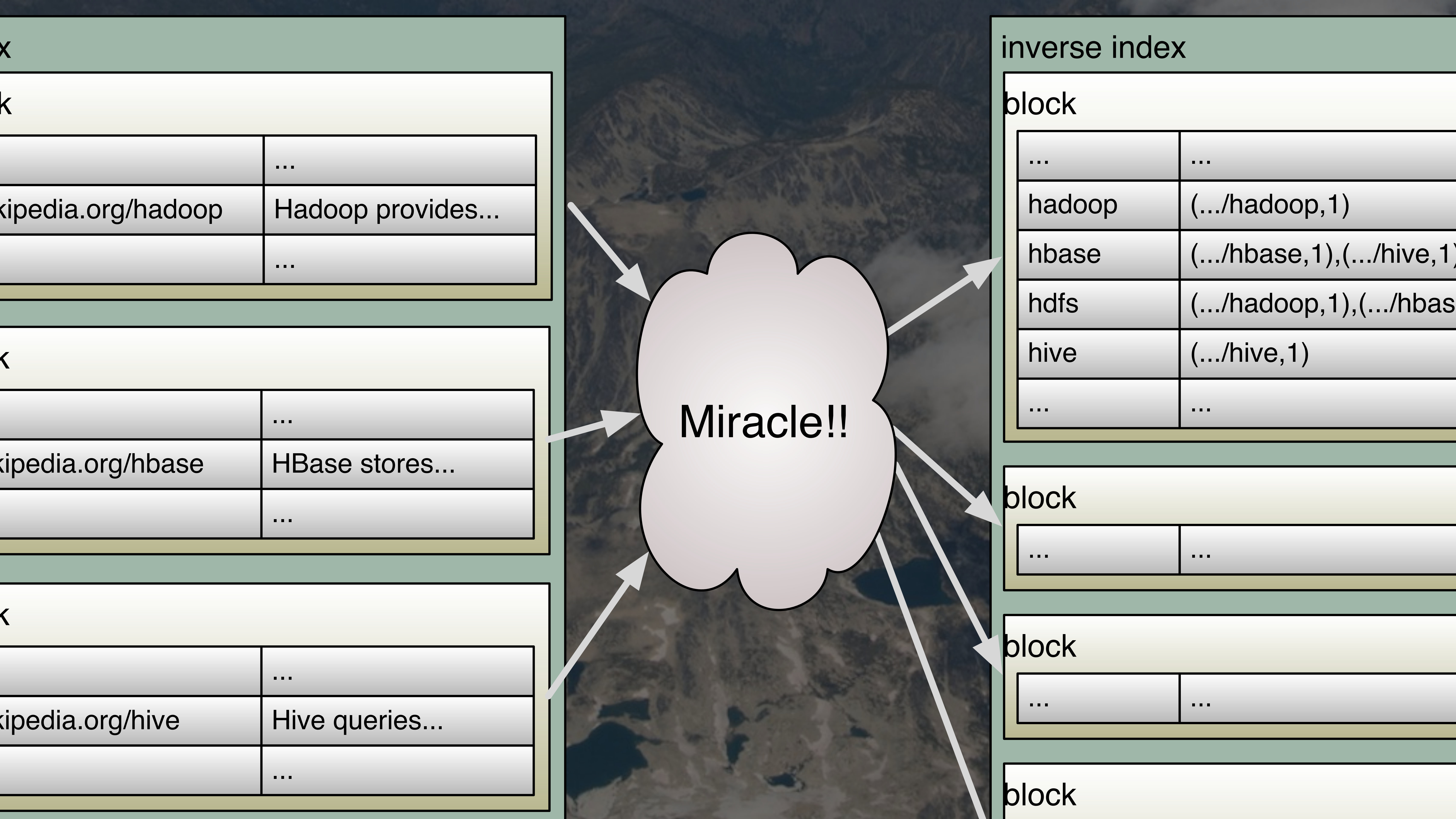
block

...	...

block

...	...

block



x

k

	...
kipedia.org/hadoop	Hadoop provides...
	...

k

	...
kipedia.org/hbase	HBase stores...
	...

k

	...
kipedia.org/hive	Hive queries...
	...

Miracle!!

inverse index

block

...	...
hadoop	(.../hadoop,1)
hbase	(.../hbase,1),(.../hive,1
hdfs	(.../hadoop,1),(.../hbas
hive	(.../hive,1)
...	...

block

...	...

block

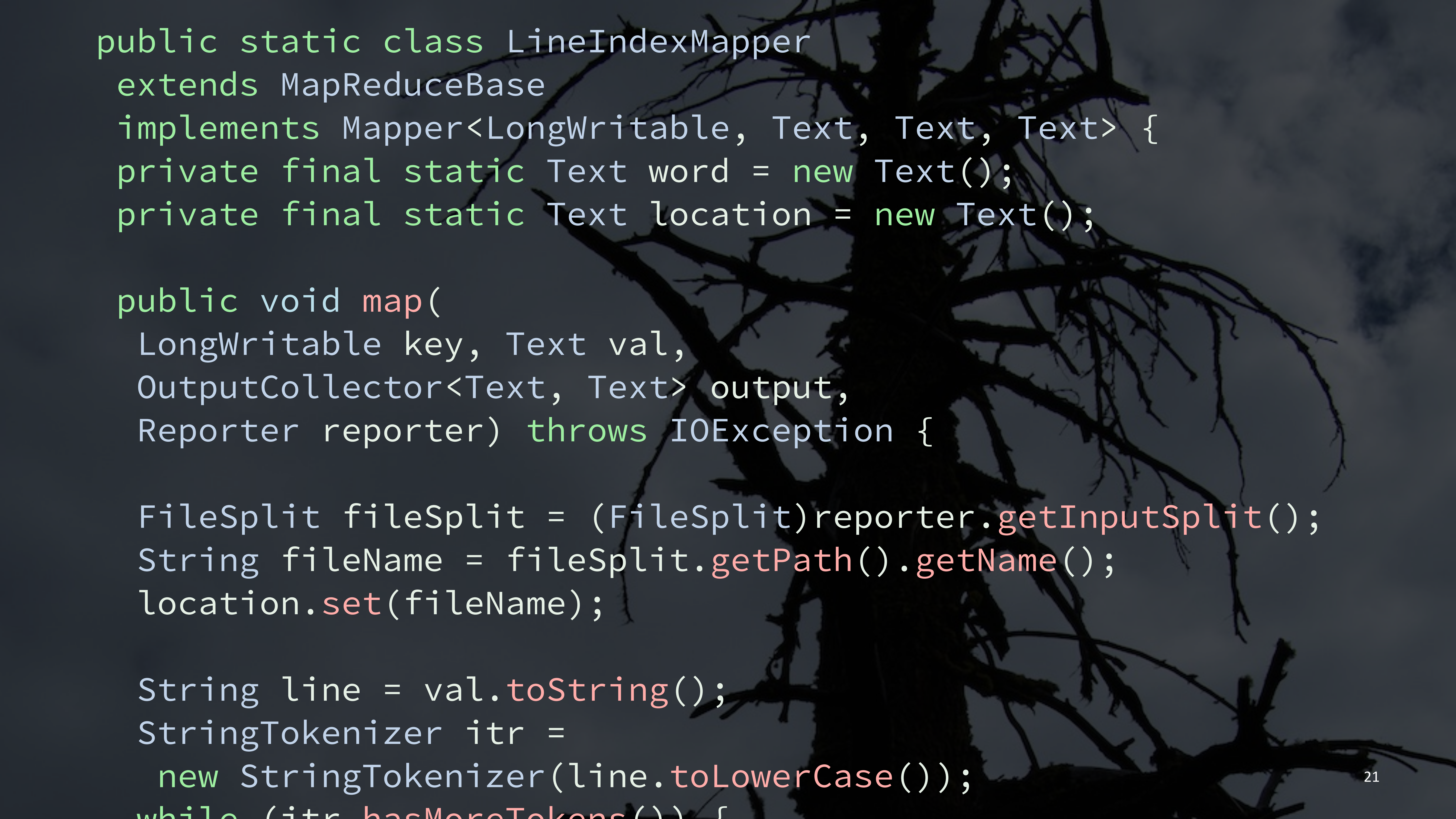
...	...

block

Java MapReduce Inverted Index

```
import java.io.IOException;
import java.util.*;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;

public class LineIndexer {

 public static void main(String[] args) {
  JobClient client = new JobClient();
  JobConf conf = new JobConf(LineIndexer.class);

  conf.setJobName("LineIndexer");
  conf.setOutputKeyClass(Text.class);
  conf.setOutputValueClass(Text.class);
  FileInputFormat.addInputPath(conf, new Path("input"));
  FileOutputFormat.setOutputPath(conf, new Path("output"));
```

```
public static void main(String[] args) {
 JobClient client = new JobClient();
 JobConf conf = new JobConf(LineIndexer.class);

 conf.setJobName("LineIndexer");
 conf.setOutputKeyClass(Text.class);
 conf.setOutputValueClass(Text.class);
 FileInputFormat.addInputPath(conf, new Path("input"));
 FileOutputFormat.setOutputPath(conf, new Path("output"));
 conf.setMapperClass(LineIndexMapper.class);
 conf.setReducerClass(LineIndexMapper.class);
 client.setConf(conf);

 try {
  JobClient.runJob(conf);
 } catch (Exception e) {
  e.printStackTrace();
 }
}
```



```
public static class LineIndexMapper
 extends MapReduceBase
 implements Mapper<LongWritable, Text, Text, Text> {
 private final static Text word = new Text();
 private final static Text location = new Text();

 public void map(
  LongWritable key, Text val,
  OutputCollector<Text, Text> output,
  Reporter reporter) throws IOException {

  FileSplit fileSplit = (FileSplit)reporter.getInputSplit();
  String fileName = fileSplit.getPath().getName();
  location.set(fileName);

  String line = val.toString();
  StringTokenizer itr =
   new StringTokenizer(line.toLowerCase());
  while (itr.hasMoreTokens()) {
```

```
 public void map(
  LongWritable key, Text val,
  OutputCollector<Text, Text> output,
  Reporter reporter) throws IOException {

  FileSplit fileSplit = (FileSplit)reporter.getInputSplit();
  String fileName = fileSplit.getPath().getName();
  location.set(fileName);

  String line = val.toString();
  StringTokenizer itr =
   new StringTokenizer(line.toLowerCase());
  while (itr.hasMoreTokens()) {
   word.set(itr.nextToken());
   output.collect(word, location);
  }
 }
}
```

Actual business logic.

```
 public static class LineIndexReducer
  extends MapReduceBase
  implements Reducer<Text, Text, Text, Text> {
  public void reduce(Text key,
   Iterator<Text> values,
   OutputCollector<Text, Text> output,
   Reporter reporter) throws IOException {
   boolean first = true;
   StringBuilder toReturn = new StringBuilder();
   while (values.hasNext()) {
    if (!first) toReturn.append(", ");
    first = false;
    toReturn.append(values.next().toString());
   }
   output.collect(key, new Text(toReturn.toString()));
  }
 }
}
```

Actual business logic.

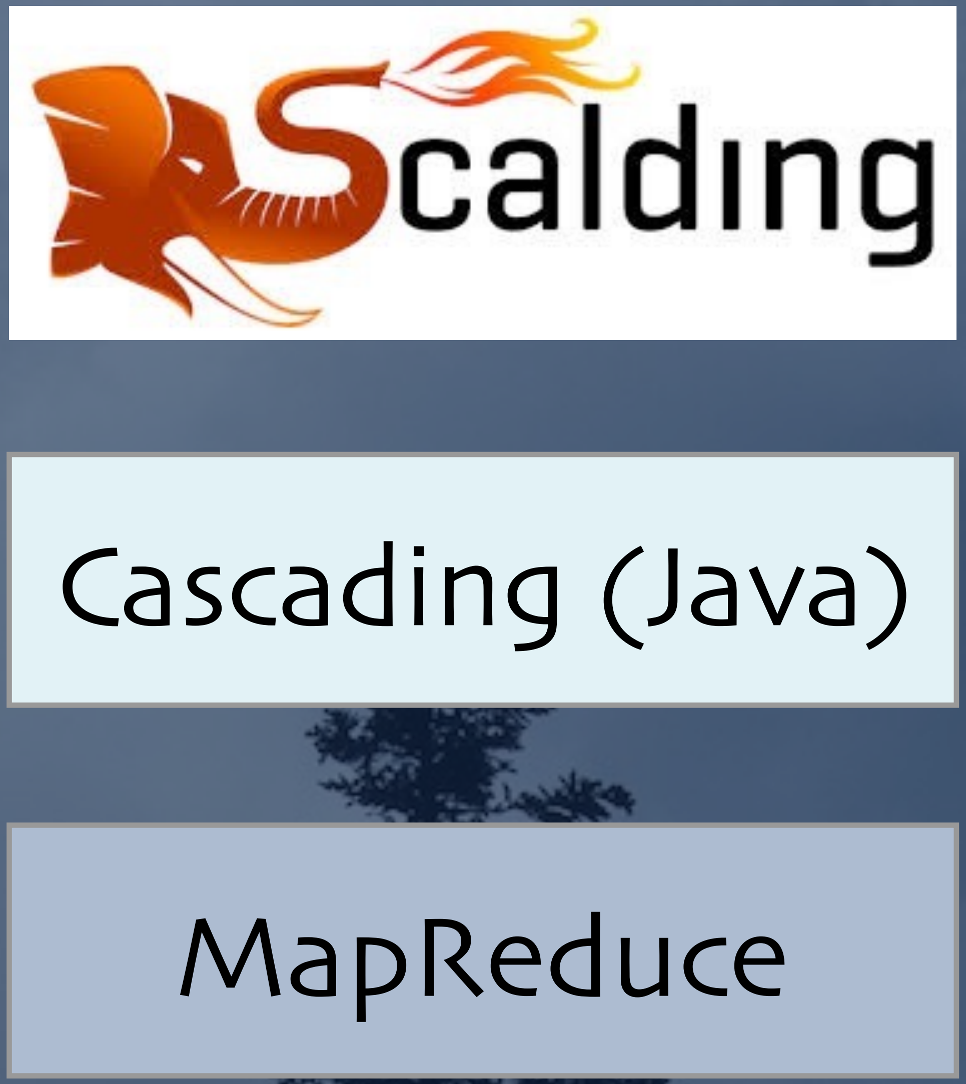
Problems

Hard to implement algorithms...

Higher Level Tools?

```
CREATE TABLE students (name STRING, age INT, gpa FLOAT);
LOAD DATA ...;
...
SELECT age, AVG(gpa)
FROM students
GROUP BY age;
```

```
A = LOAD 'students' USING PigStorage()
  AS (name:chararray, age:int, gpa:float);
B = GROUP A BY age;
C = FOREACH B GENERATE group AS age, AVG(gpa);
DUMP c;
```



Scalding

Cascading (Java)

MapReduce

```
import com.twitter.scalding._

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

  val texts = Tsv("texts.tsv", ('id, 'text))
  val wordToIds = texts
   .flatMap(('id, 'text) -> ('word, 'id2)) {
     fields: (String, String) =>
      val (id2, text) =
       text.split("\\s+").map {
        word => (word, id2)
       }
    }

  val invertedIndex = wordToTweets
   .groupBy('word)(_.toList[String]('id2 -> 'ids))
  invertedIndex.write(Tsv("output.tsv"))
}
```

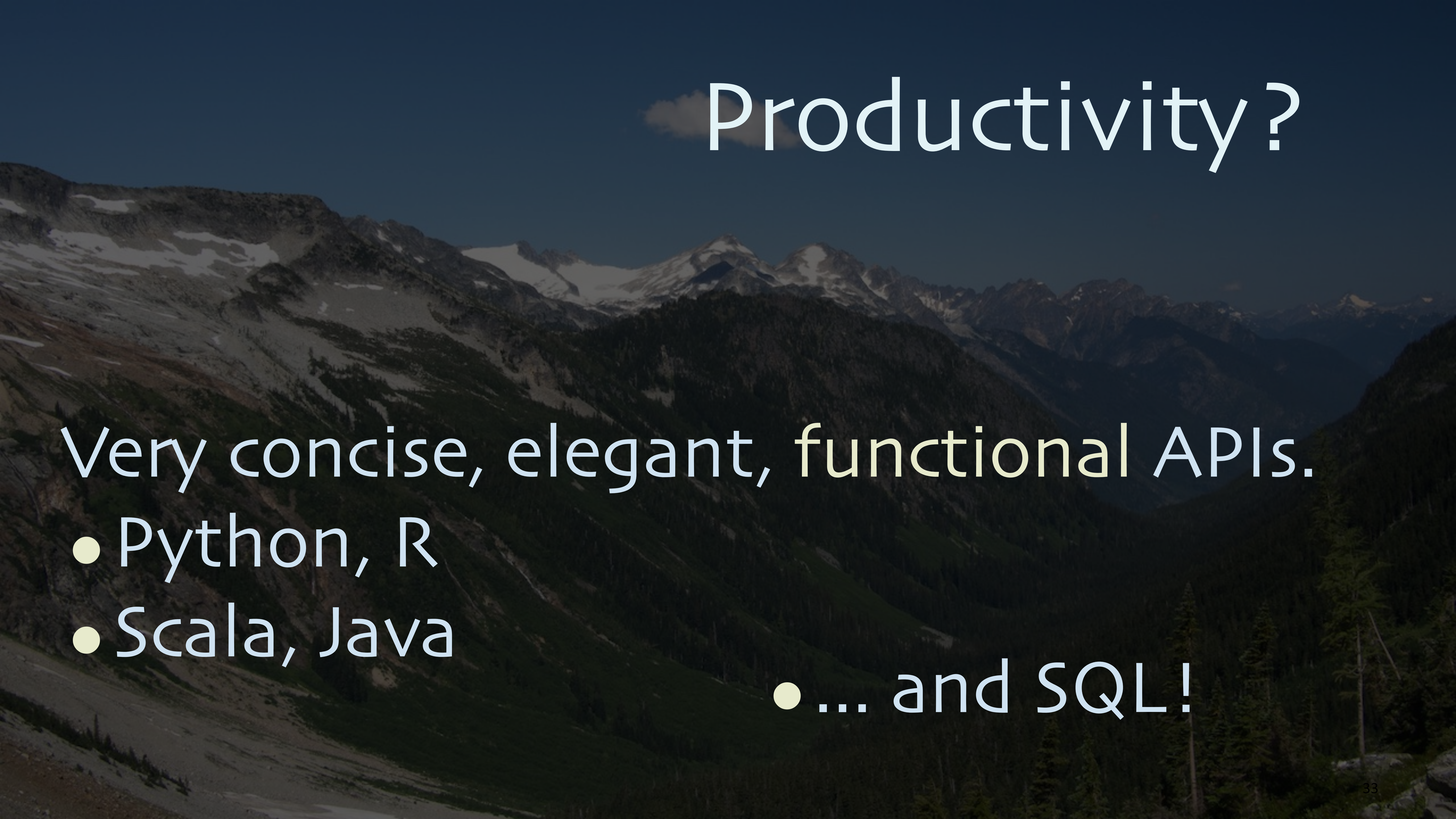
Problems

Only “Batch mode“;
What about streaming?

Problems

Performance needs to be better

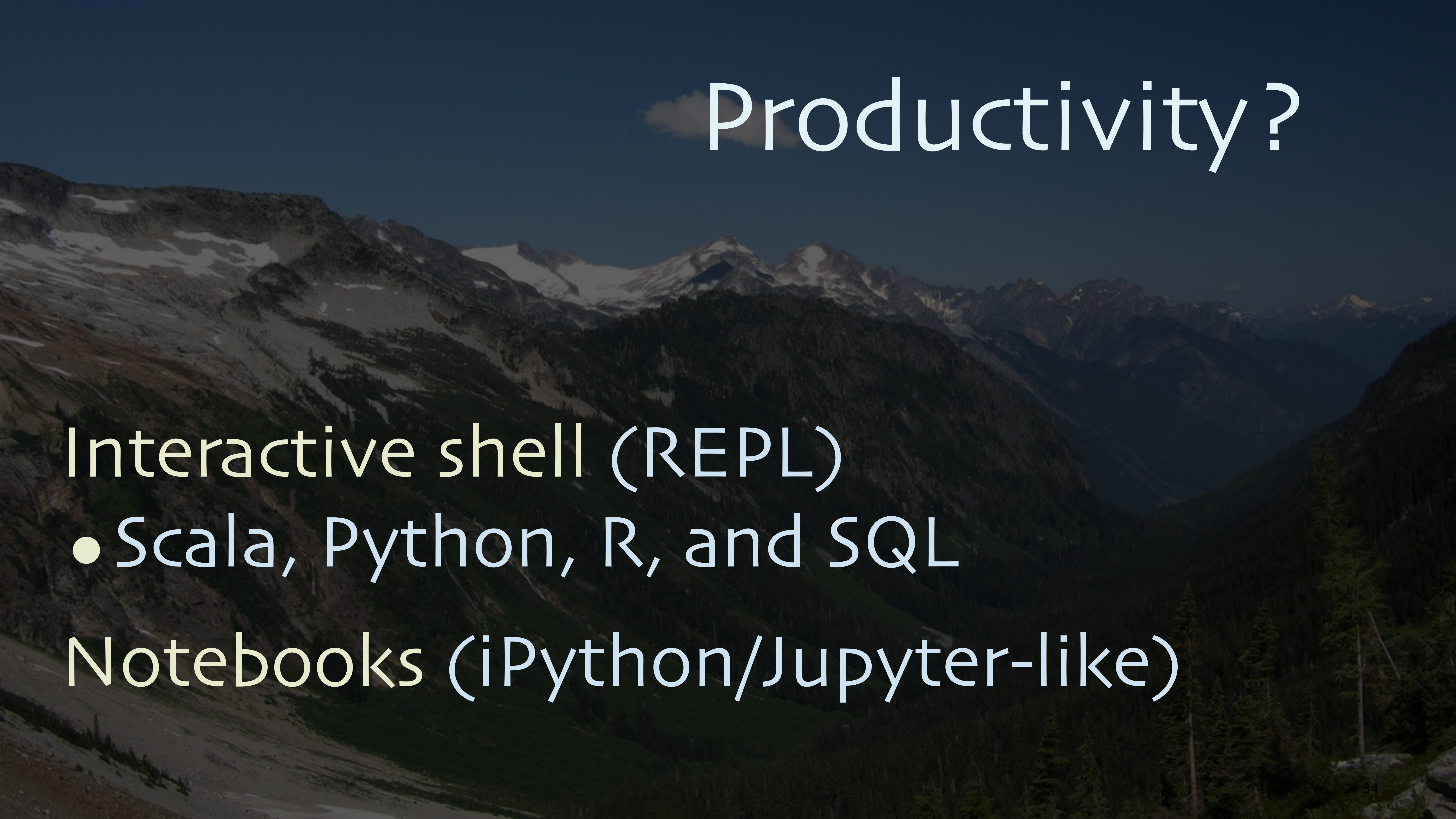
Spark



Productivity?

Very concise, elegant, functional APIs.

- Python, R
- Scala, Java
- … and SQL!

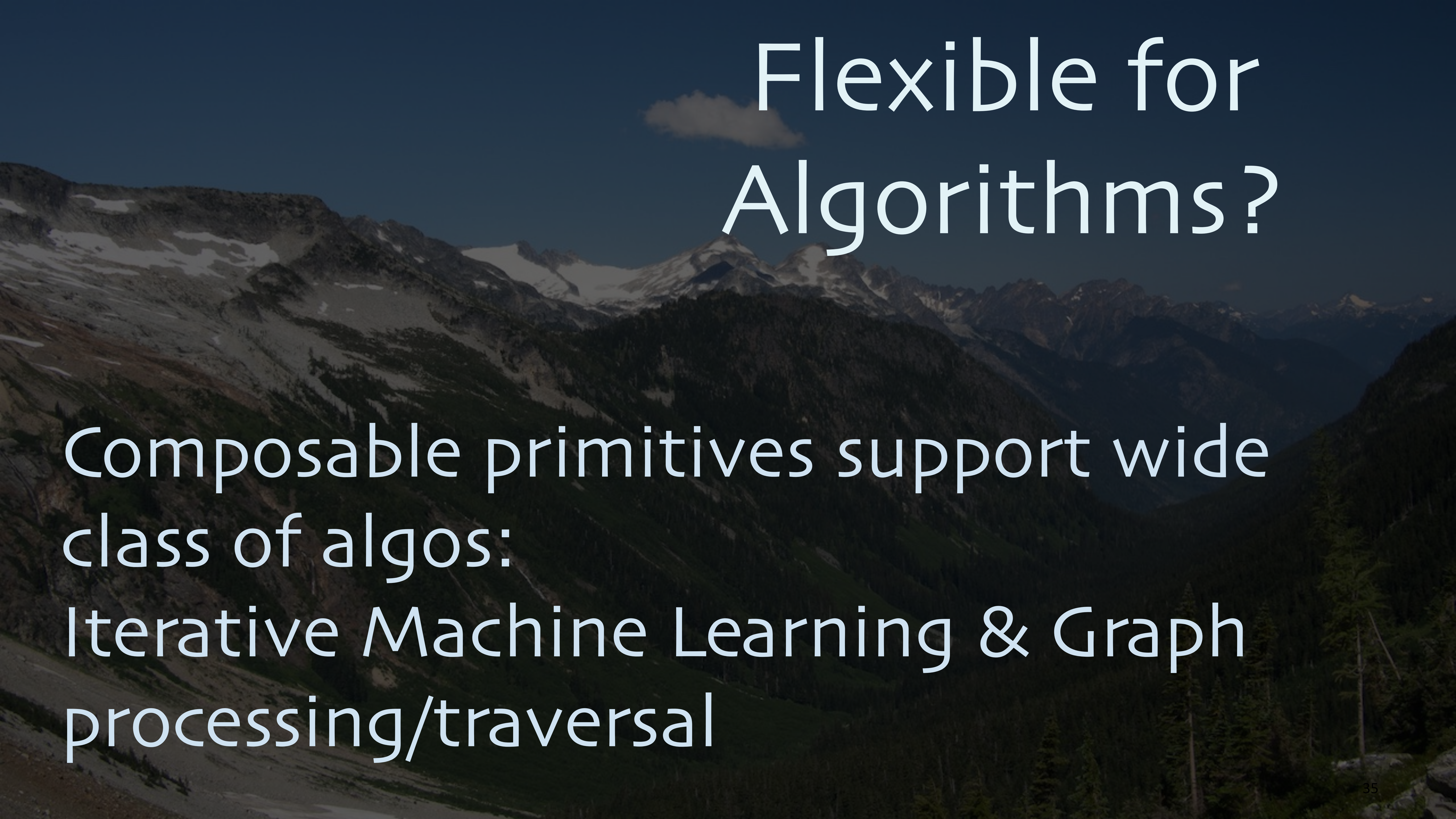


Productivity?

Interactive shell (REPL)

- Scala, Python, R, and SQL

Notebooks (iPython/Jupyter-like)



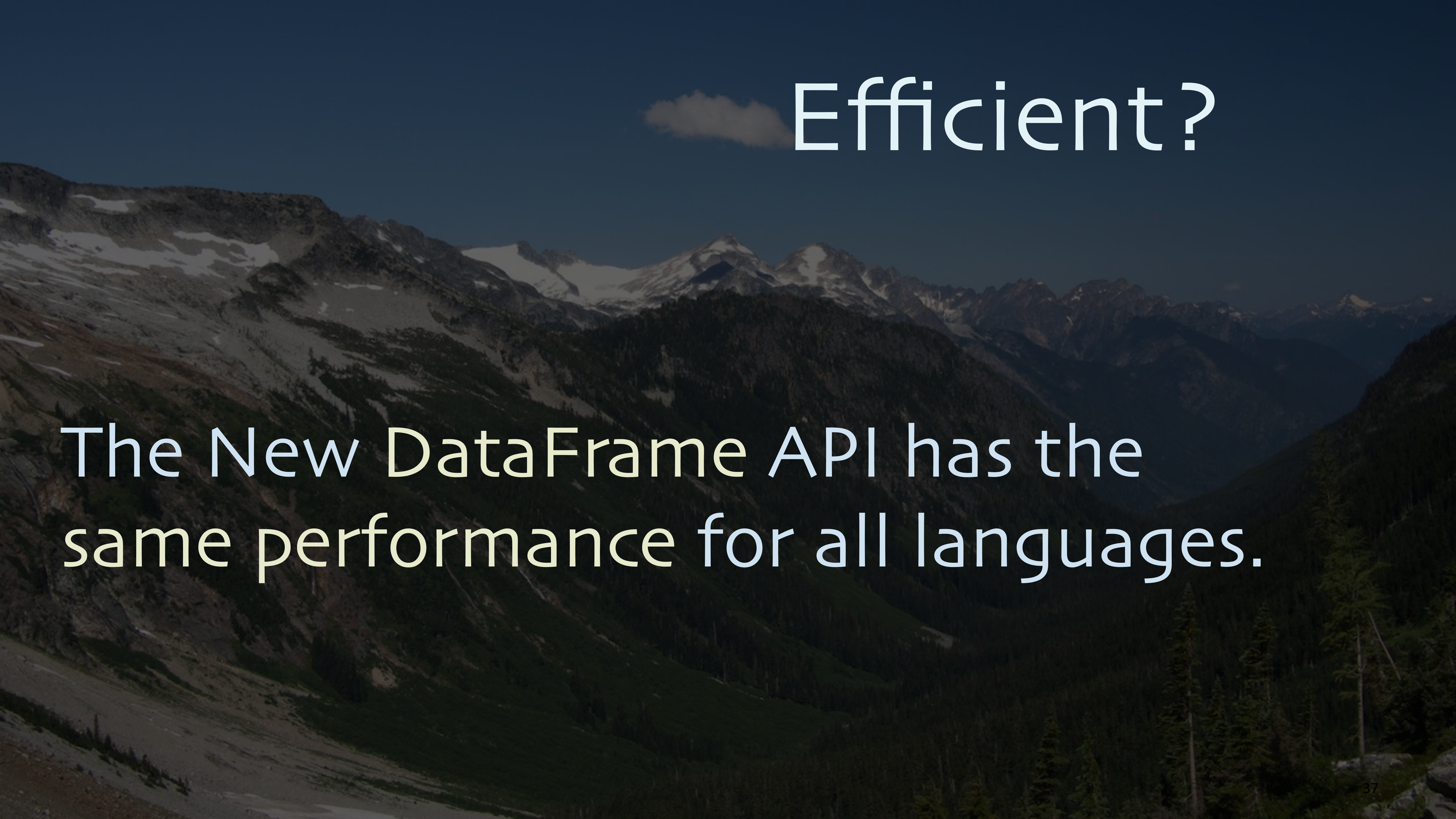
Flexible for Algorithms?

Composable primitives support wide class of algos:
Iterative Machine Learning & Graph processing/traversal

Efficient?

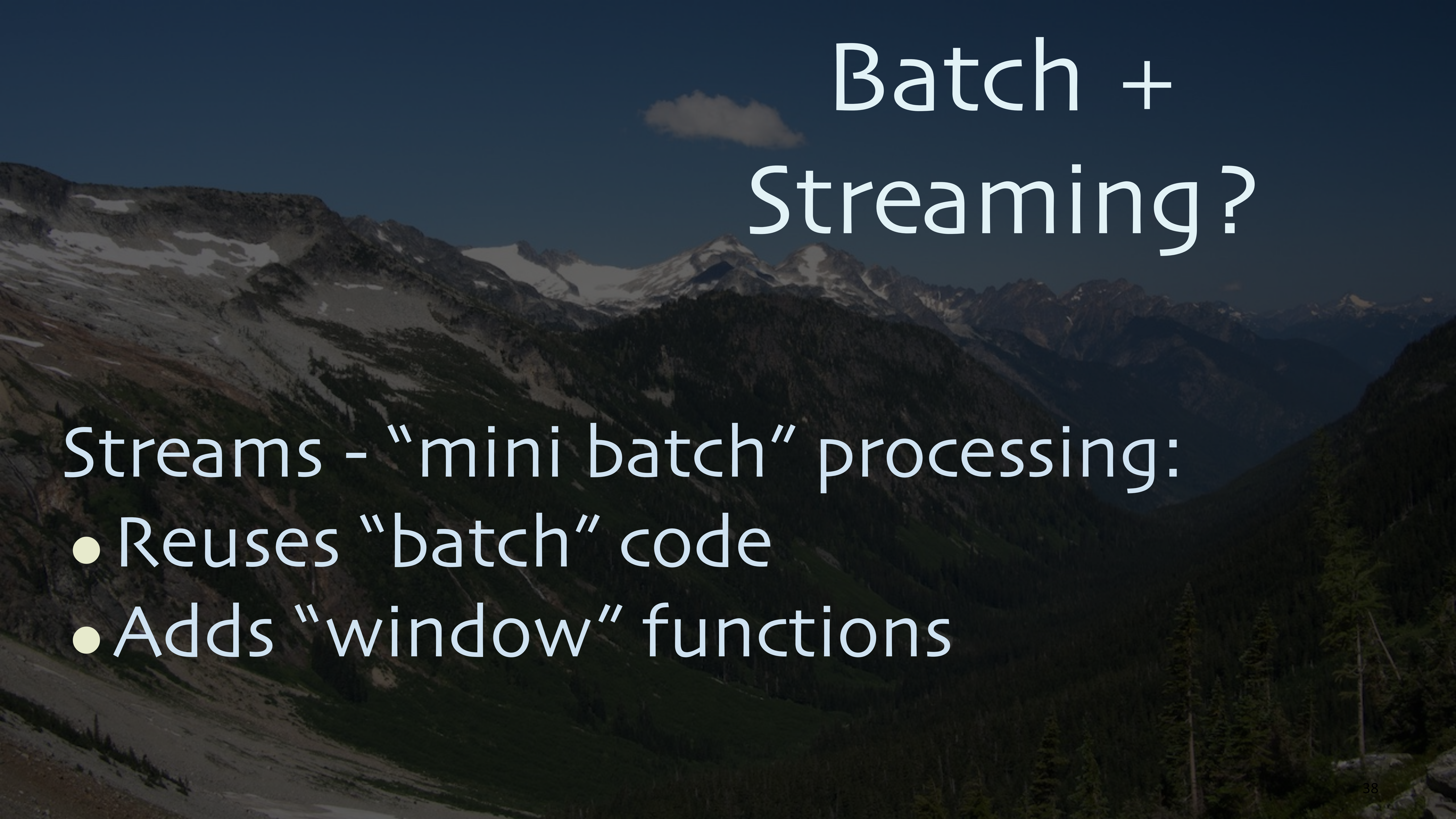
Builds a dataflow DAG:

- Combines steps into “stages”
- Can cache intermediate data



Efficient?

The New DataFrame API has the same performance for all languages.

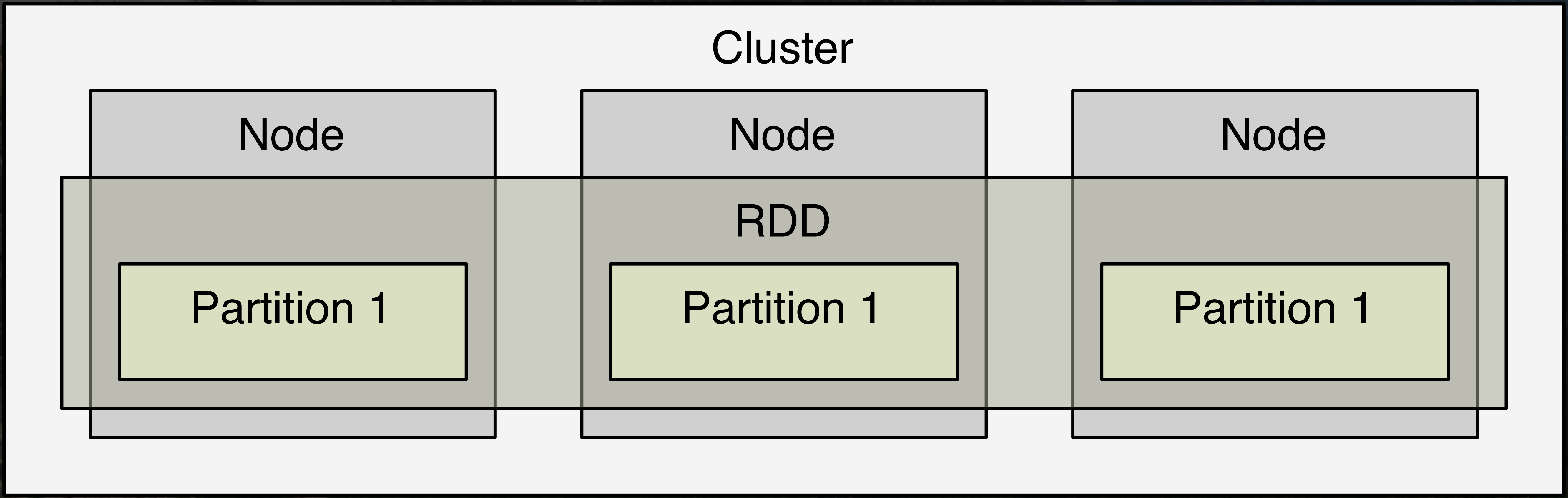


Batch + Streaming?

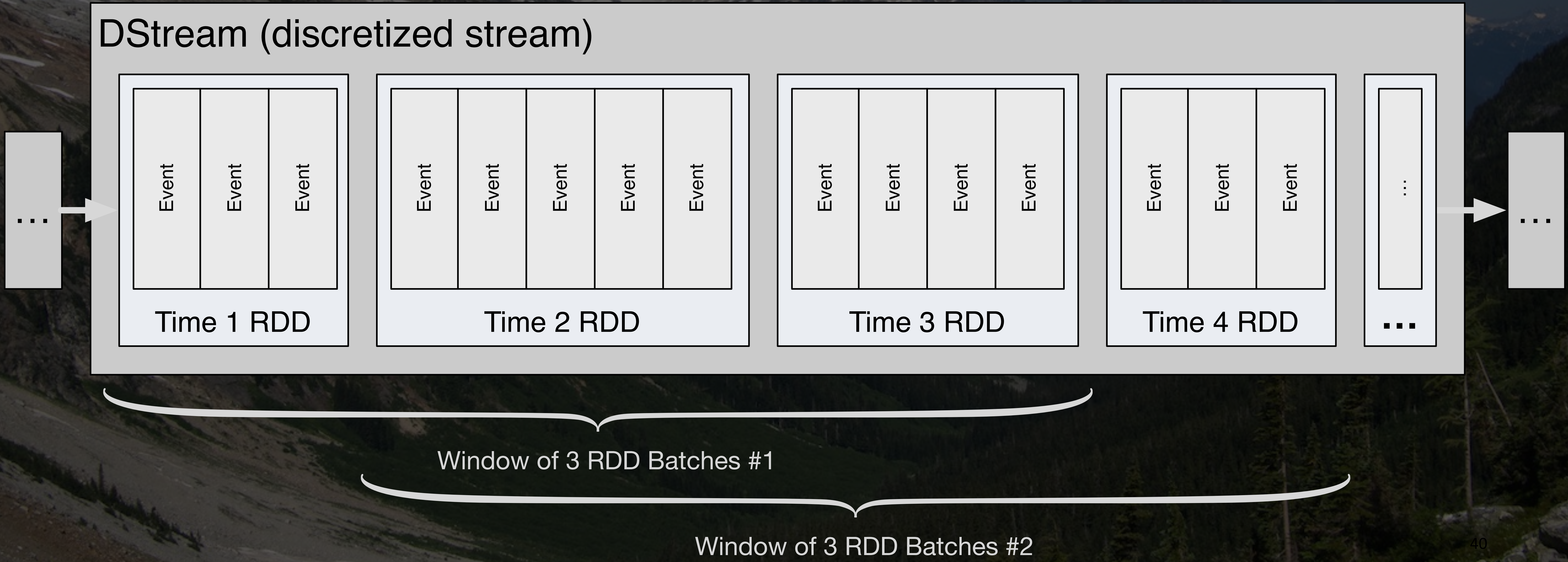
Streams - “mini batch” processing:

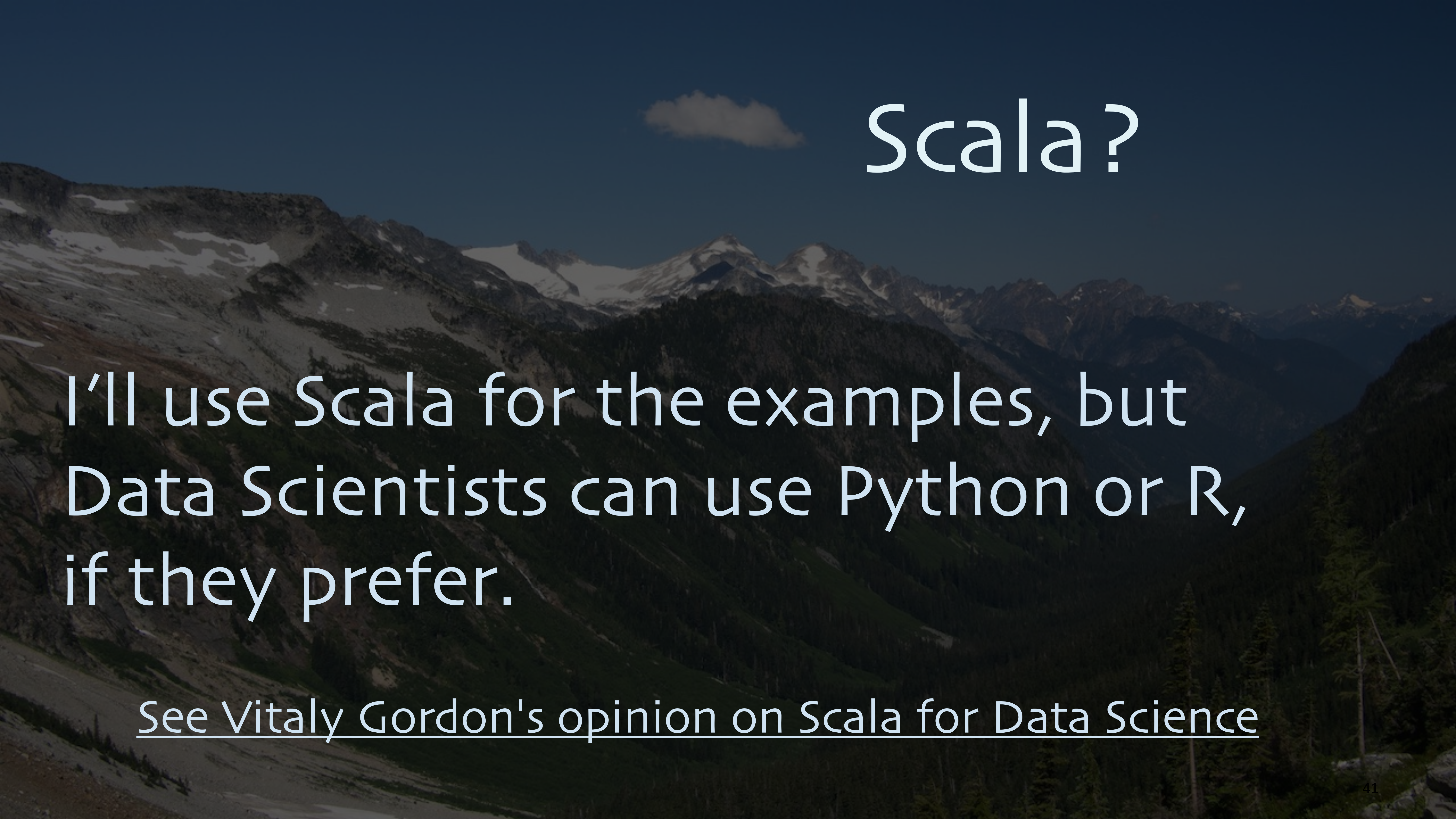
- Reuses “batch” code
- Adds “window” functions

Resilient Distributed Datasets (RDDs)



DStreams


DStream (discretized stream)
...
Event
Event
Event
Time 1 RDD
Event
Event
Event
Event
Event
Time 2 RDD
Event
Event
Event
Event
Time 3 RDD
Event
Event
Event
Time 4 RDD
...
...
...
Window of 3 RDD Batches #1
Window of 3 RDD Batches #2



Scala?

I'll use Scala for the examples, but Data Scientists can use Python or R, if they prefer.

See Vitaly Gordon's opinion on Scala for Data Science

Inverted Index

```
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._

object InvertedIndex {
 def main(args: Array[String]) = {

  val sc = new SparkContext("local", "Inverted Index")

  sc.textFile("data/crawl")
  .map { line =>
    val array = line.split("\t", 2)
    (array(0), array(1))
  }
  .flatMap {
    case (path, text) =>
```

```
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._

object InvertedIndex {
 def main(args: Array[String]) = {

  val sc = new SparkContext("local", "Inverted Index")

  sc.textFile("data/crawl")
  .map { line =>
    val array = line.split("\t", 2)
    (array(0), array(1))
  }
  .flatMap {
    case (path, text) =>
```

```
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._

object InvertedIndex {
 def main(args: Array[String]) = {

  val sc = new SparkContext("local", "Inverted Index")

  sc.textFile("data/crawl")
  .map { line =>
    val array = line.split("\t", 2)
    (array(0), array(1))
  }
  .flatMap {
    case (path, text) =>
```

```
sc.textFile("data/crawl")
  .map { line =>
    val array = line.split("\t", 2)
    (array(0), array(1))
  }
  .flatMap {
    case (path, text) =>
      text.split("""\W+""") map {
        word => (word, path)
      }
  }
  .map {
    case (w, p) => ((w, p), 1)
  }
  .reduceByKey {
```

```
sc.textFile("data/crawl")
.map { line =>
  val array = line.split("\t", 2)
  (array(0), array(1))
}
.flatMap {
  case (path, text) =>
    text.split("""\W+""") map {
      word => (word, path)
    }
}
.map {
  case (w, p) => ((w, p), 1)
}
.reduceByKey {
```

```
(word1, path1)
(word2, path2)
…
```

```
      }
  }
  .map {
    case (w, p) => ((w, p), 1)
  }
  .reduceByKey {
    (n1, n2) => n1 + n2
  }
  .map {
    case ((word,path),n) => (word,(path,n))
  }
  .groupByKey
  .mapValues { iter =>
    iter.toSeq.sortBy {
      case (path, n) => (-n, path)
    }.mkString(", ")
```

((word1, path1), N1)
((word2, path2), N2)
…

```
    }
}
.map {
  case (w, p) => ((w, p), 1)
}
.reduceByKey {
  (n1, n2) => n1 + n2
}
.map {
  case ((word,path),n) => (word,(path,n))
}
.groupByKey
.mapValues { iter =>
  iter.toSeq.sortBy {
   case (path, n) => (-n, path)
  }.mkString(", ")
```

((word1, path1), N1)
((word2, path2), N2)
…

(word1, (path1, N1))
(word2, (path2, N2))
…

```
  }
  .map {
    case ((word,path),n) => (word,(path,n))
  }
  .groupByKey
  .mapValues { iter =>
    iter.toSeq.sortBy {
     case (path, n) => (-n, path)
    }.mkString(", ")
  }
  .saveAsTextFile("output/inverted-index")

  sc.stop()
 }
}
```

(word, Seq((path1, n1), (path2, n2), (path3, n3), ...))

...

```
  }
  .map {
    case ((word,path),n) => (word,(path,n))
  }
  .groupByKey
  .mapValues { iter =>
    iter.toSeq.sortBy {
     case (path, n) => (-n, path)
    }.mkString(", ")
  }
  .saveAsTextFile("output/inverted-index")

  sc.stop()
 }
}
```

(word, “(path4, 80), (path19, 51), (path8, 12), ...”)

…

```
  }
  .map {
    case ((word,path),n) => (word,(path,n))
  }
  .groupByKey
  .mapValues { iter =>
    iter.toSeq.sortBy {
     case (path, n) => (-n, path)
    }.mkString(", ")
  }
  .saveAsTextFile("output/inverted-index")

  sc.stop()
 }
}
```

```
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._

object InvertedIndex {
 def main(args: Array[String]) = {

  val sc = new SparkContext(
   "local", "Inverted Index")

  sc.textFile("data/crawl")
  .map { line =>
    val array = line.split("\t", 2)
    (array(0), array(1))
  }
  .flatMap {
    case (path, text) =>
     text.split("""\W+""") map {
      word => (word, path)
     }
  }
  .map {
    case (w, p) => ((w, p), 1)
  }
  .reduceByKey {
    (n1, n2) => n1 + n2
  }
  .map {
    case ((word,path),n) => (word,(path,n))
  }
  .groupByKey
  .mapValues { iter =>
    iter.toSeq.sortBy {
     case (path, n) => (-n, path)
    }.mkString(", ")
  }
  .saveAsTextFile(argz.outpath)

  sc.stop()
 }
}
```

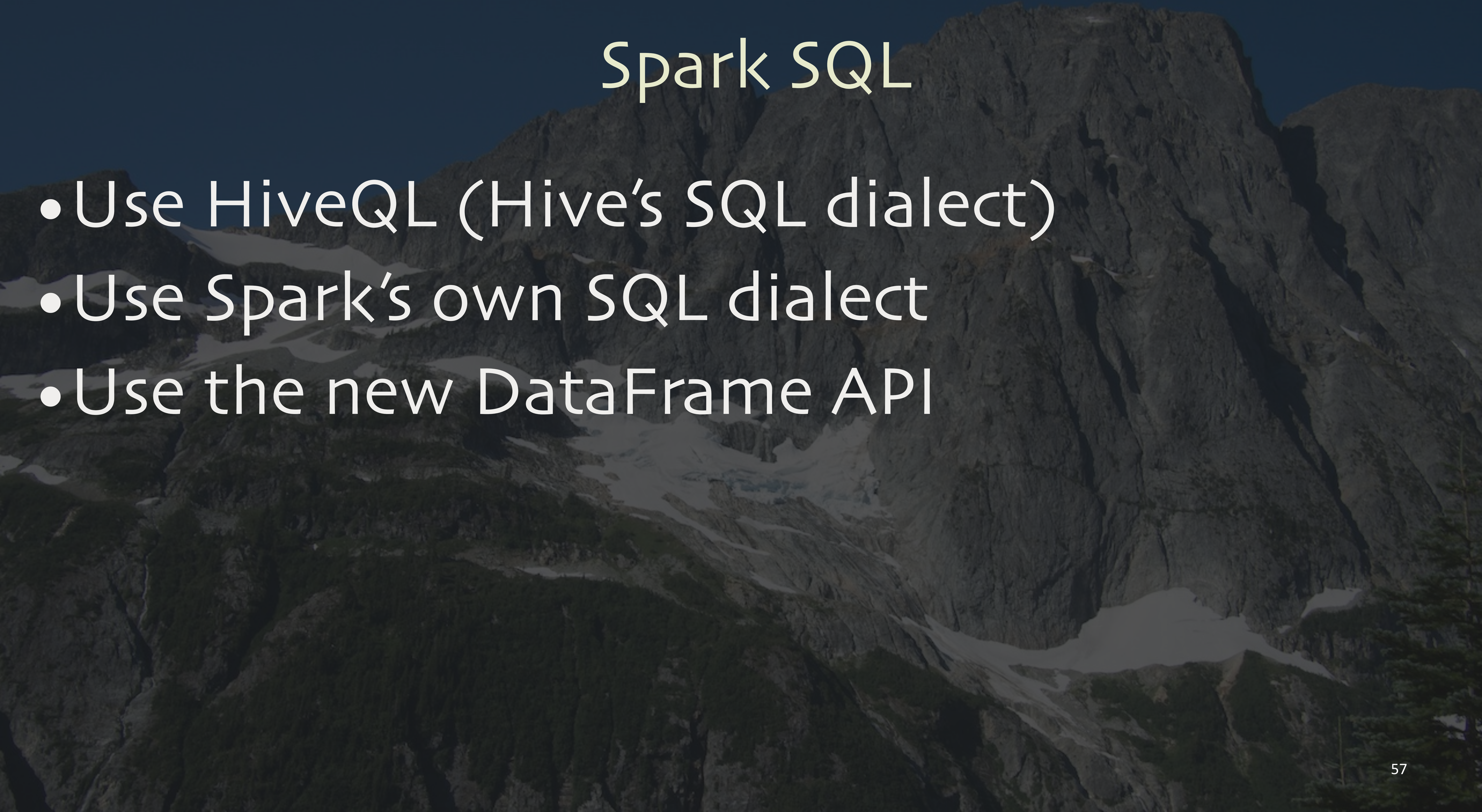
Altogether

```
    word => (word, path)
   }
}
.map {
  case (w, p) => ((w, p), 1)
}
.reduceByKey {
  (n1, n2) => n1 + n2
}
.map {
  case ((word,path),n) => (word,(path,n))
}
.groupByKey
.mapValues { iter =>
  iter.toSeq.sortBy {
   case (path, n) => (-n, path)
  }.mkString(", ")
```

Powerful, composable “operators”

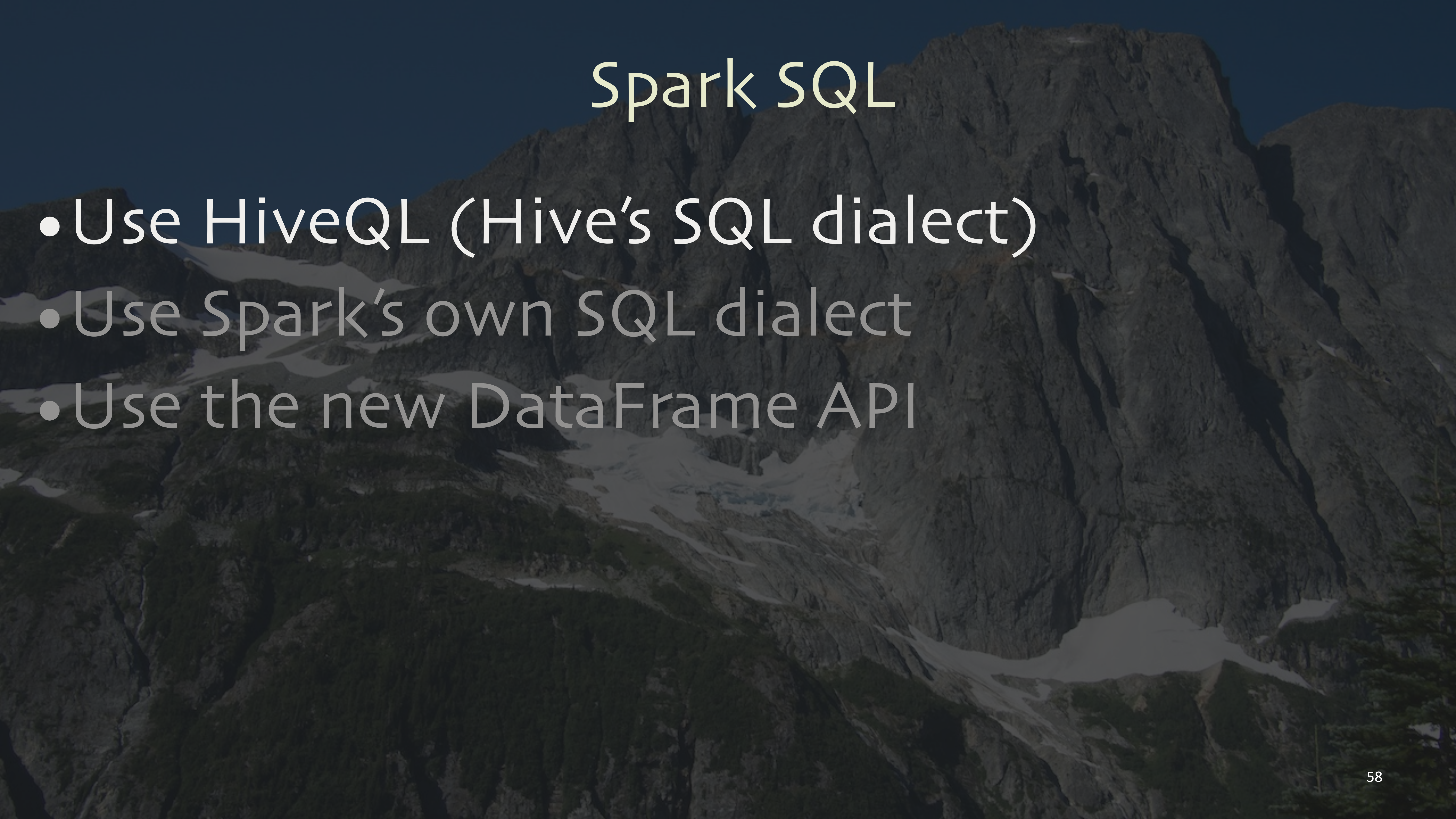
The larger ecosystem

SQL Revisited



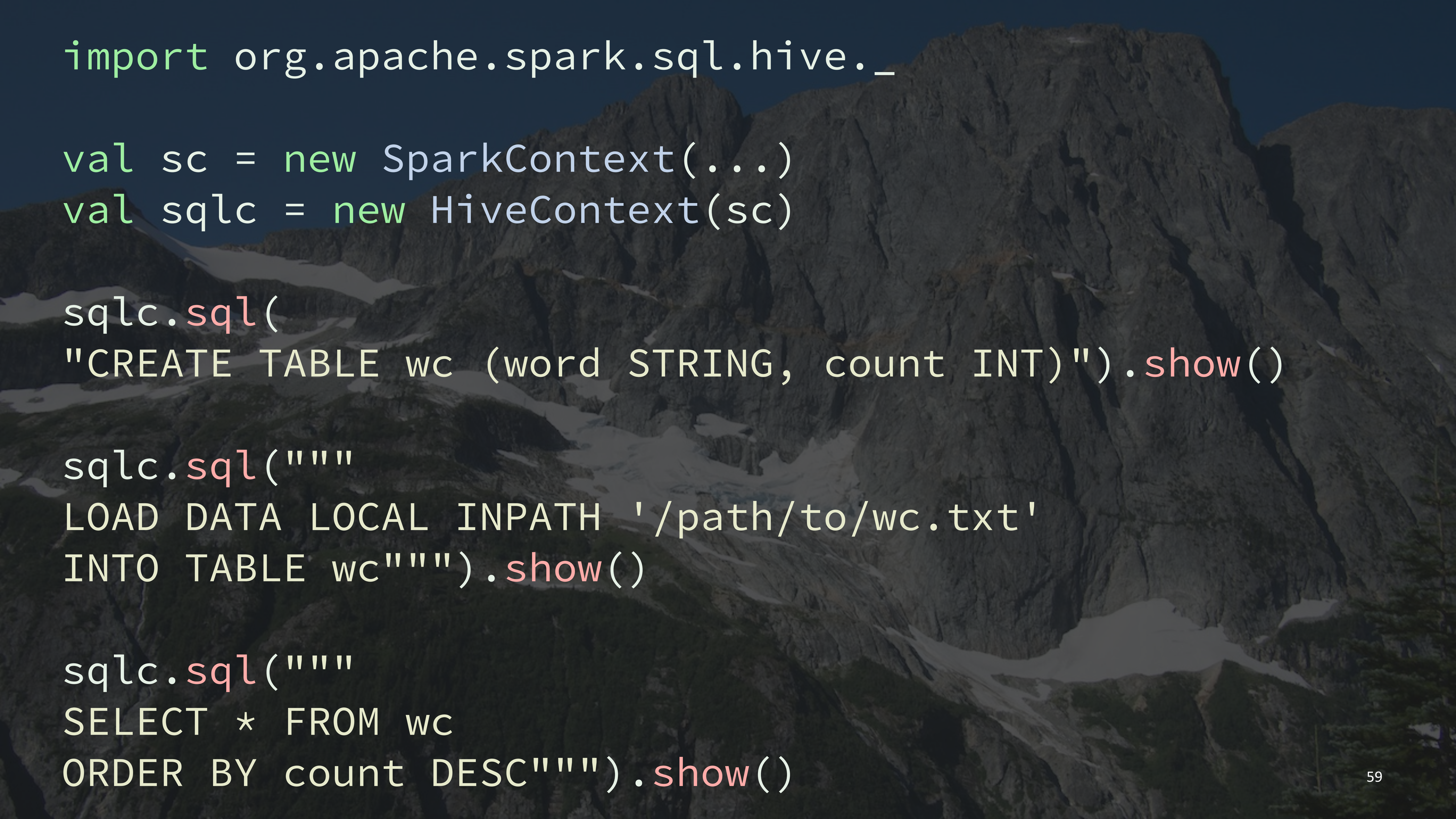
Spark SQL

- Use HiveQL (Hive's SQL dialect)
- Use Spark's own SQL dialect
- Use the new DataFrame API



Spark SQL

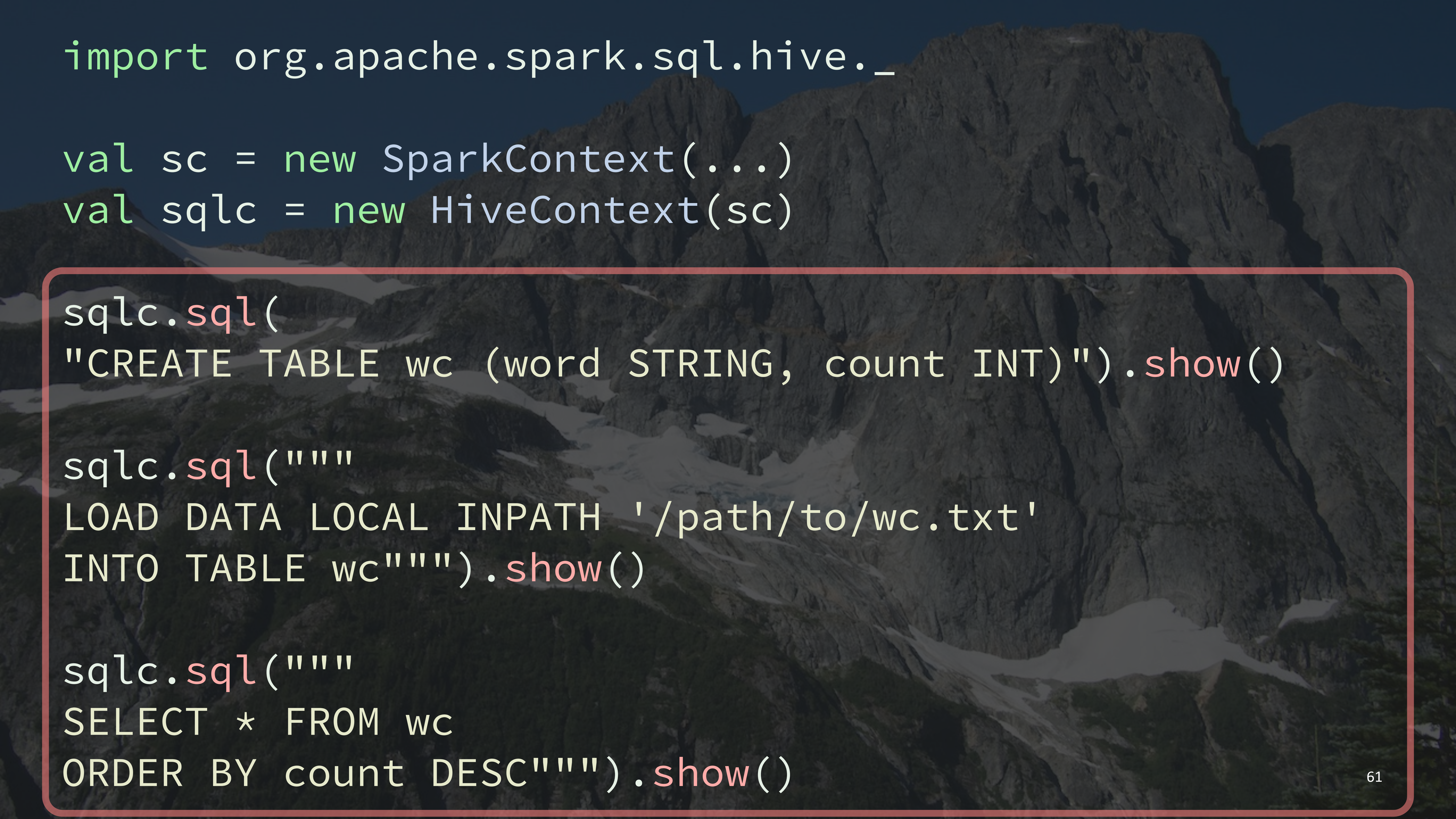
- Use HiveQL (Hive’s SQL dialect)
- Use Spark’s own SQL dialect
- Use the new DataFrame API



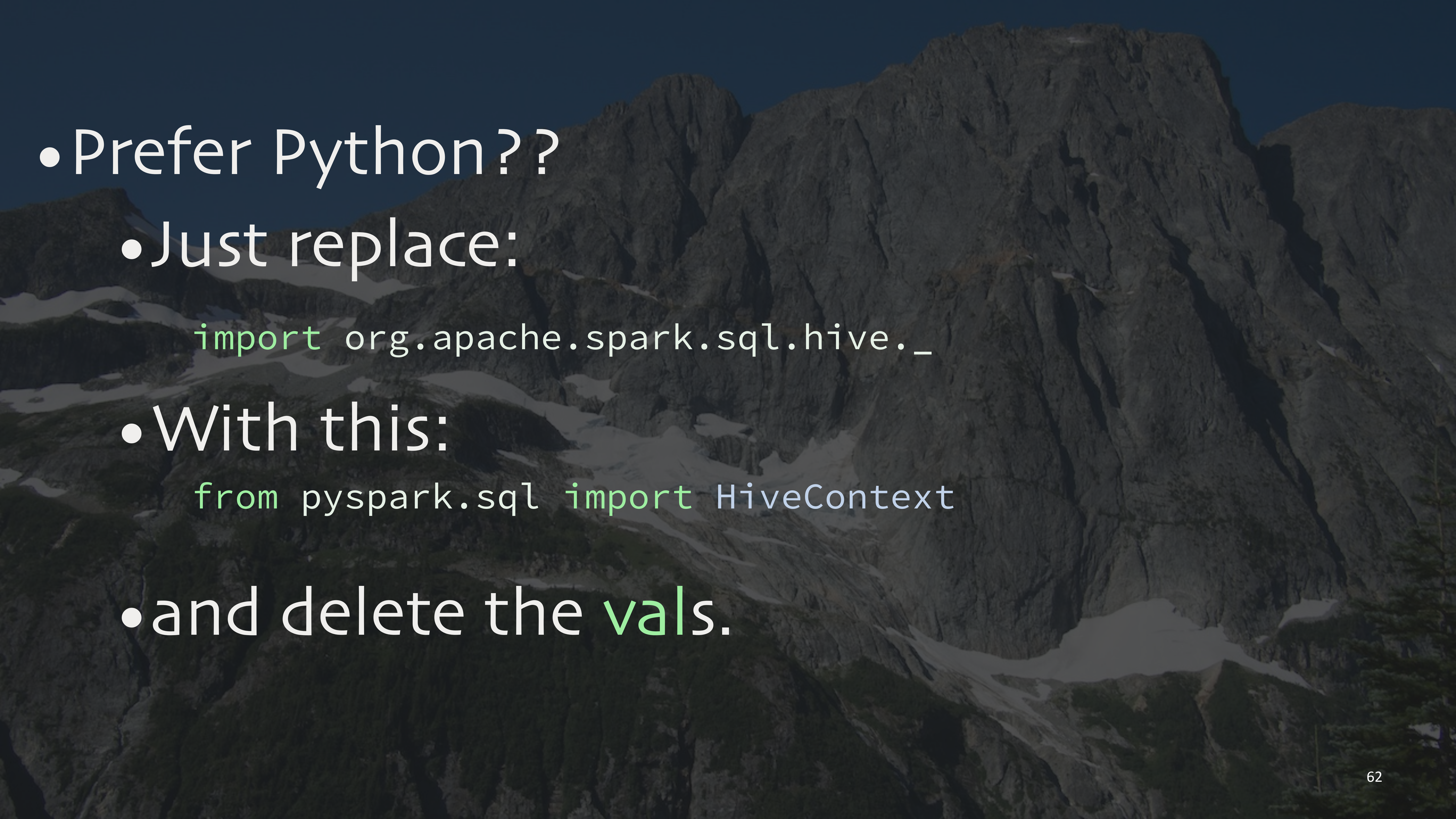
```
import org.apache.spark.sql.hive._

val sc = new SparkContext(...)
val sqlc = new HiveContext(sc)

sqlc.sql(
"CREATE TABLE wc (word STRING, count INT)").show()

sqlc.sql("""
LOAD DATA LOCAL INPATH '/path/to/wc.txt'
INTO TABLE wc""").show()

sqlc.sql("""
SELECT * FROM wc
ORDER BY count DESC""").show()
```

```
import org.apache.spark.sql.hive._

val sc = new SparkContext(...)
val sqlc = new HiveContext(sc)
```

```
sqlc.sql(
"CREATE TABLE wc (word STRING, count INT)").show()

sqlc.sql("""
LOAD DATA LOCAL INPATH '/path/to/wc.txt'
INTO TABLE wc""").show()

sqlc.sql("""
SELECT * FROM wc
ORDER BY count DESC""").show()
```



```
import org.apache.spark.sql.hive._

val sc = new SparkContext(...)
val sqlc = new HiveContext(sc)

sqlc.sql(
"CREATE TABLE wc (word STRING, count INT)").show()

sqlc.sql("""
LOAD DATA LOCAL INPATH '/path/to/wc.txt'
INTO TABLE wc""").show()

sqlc.sql("""
SELECT * FROM wc
ORDER BY count DESC""").show()
```



- Prefer Python??
  - Just replace:

```
import org.apache.spark.sql.hive._
```

  - With this:

```
from pyspark.sql import HiveContext
```

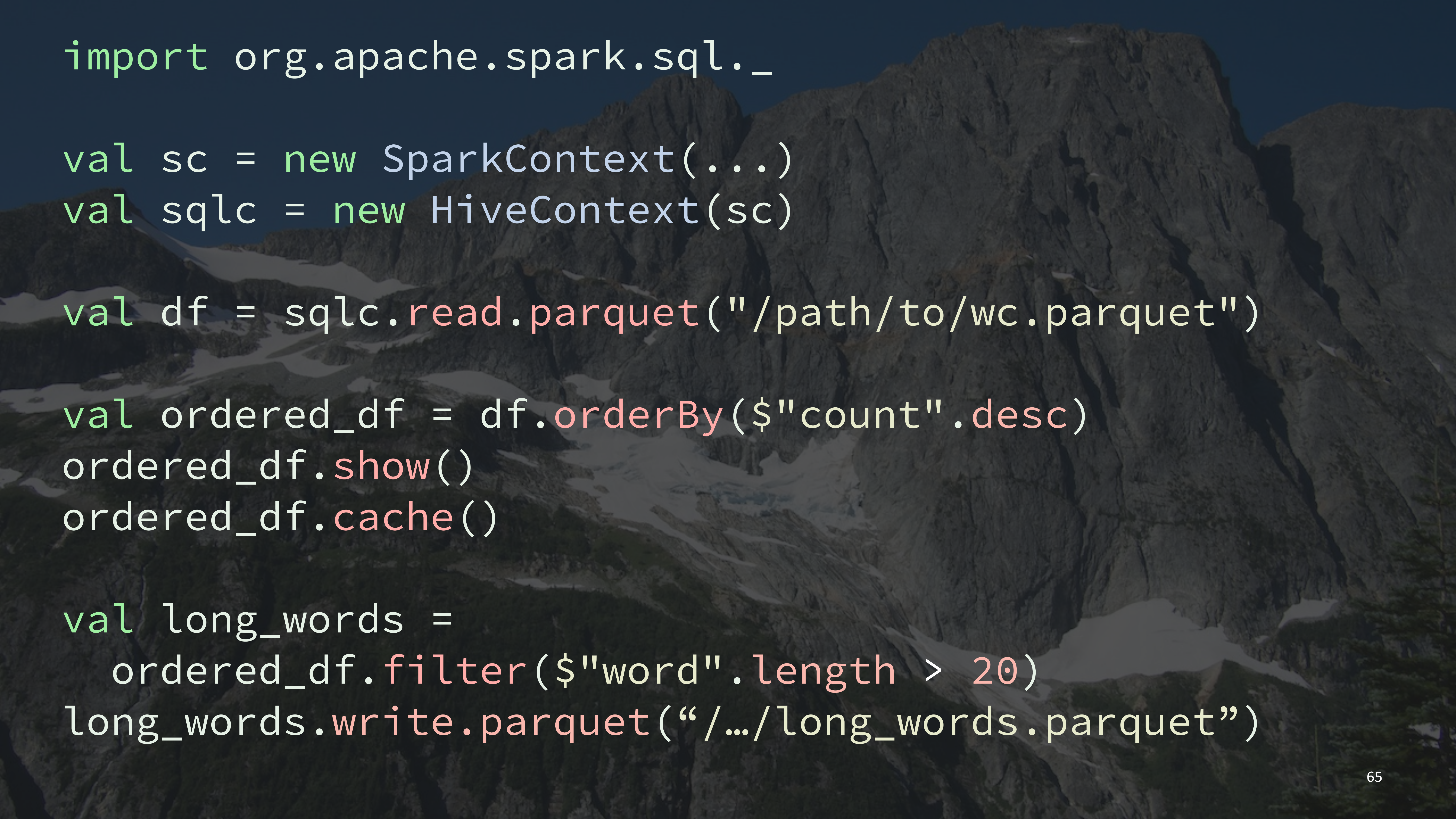
  - and delete the vals.

- Prefer Just HiveQL??
  - Use spark-sql shell script:

```
CREATE TABLE wc (word STRING, count INT);

LOAD DATA LOCAL INPATH '/path/to/wc.txt'
INTO TABLE wc;

SELECT * FROM wc
ORDER BY count DESC;
```

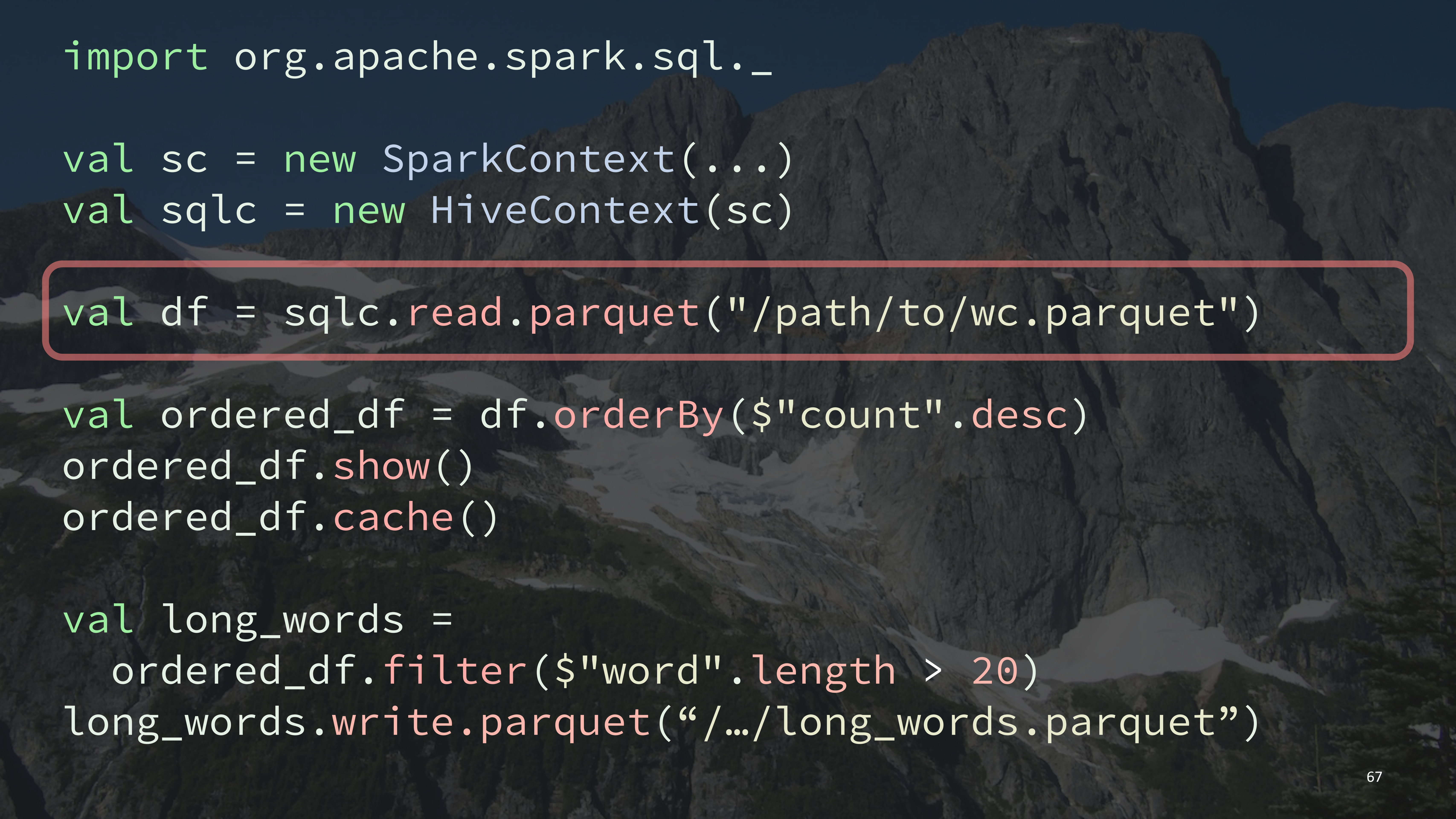
Spark SQL

- Use HiveQL (Hive's SQL dialect)
- Use Spark's own SQL dialect
- Use the new DataFrame API

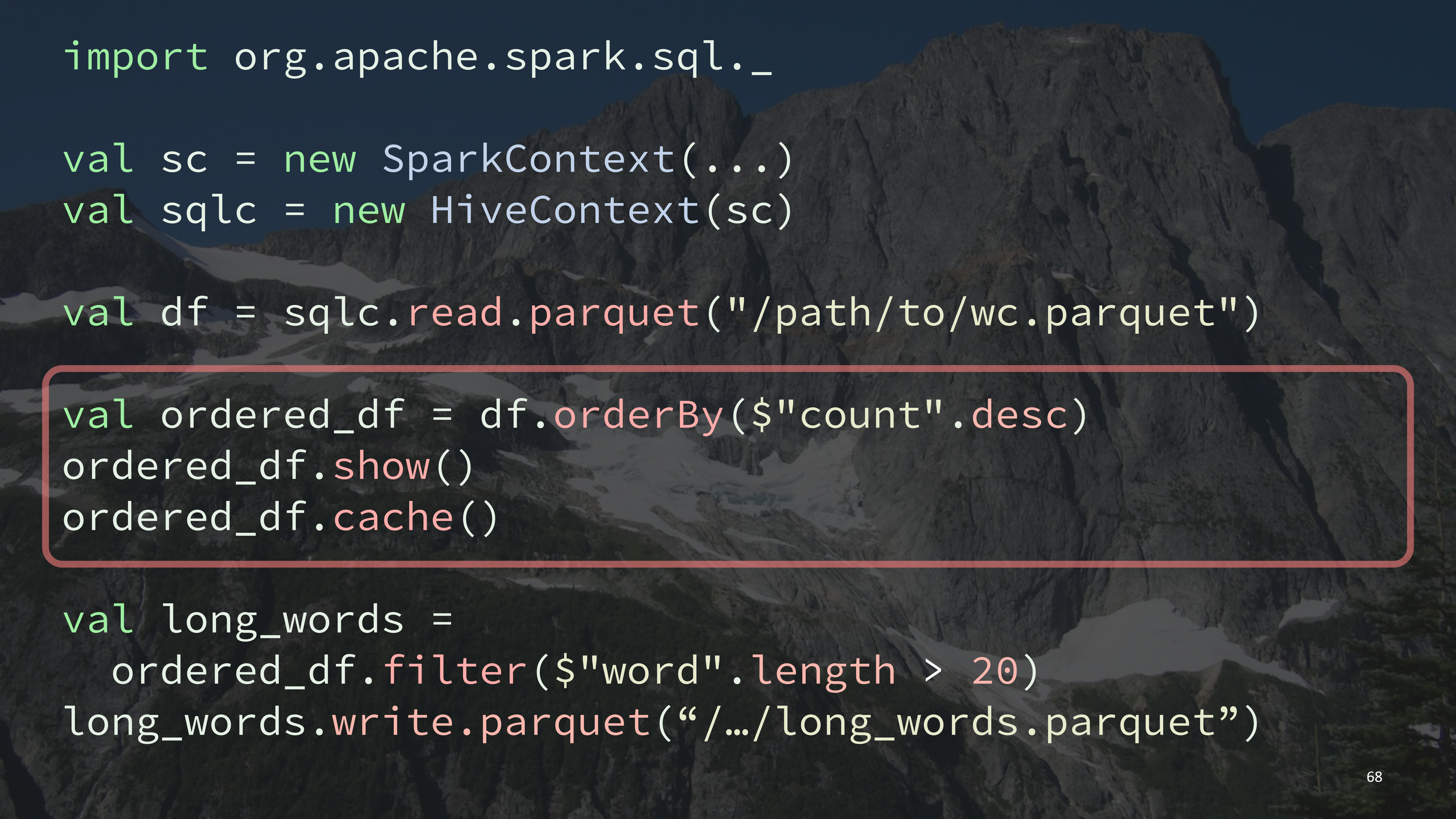


```
import org.apache.spark.sql._

val sc = new SparkContext(...)
val sqlc = new HiveContext(sc)

val df = sqlc.read.parquet("/path/to/wc.parquet")

val ordered_df = df.orderBy($"count".desc)
ordered_df.show()
ordered_df.cache()

val long_words =
  ordered_df.filter($"word".length > 20)
long_words.write.parquet(“/…/long_words.parquet”)
```

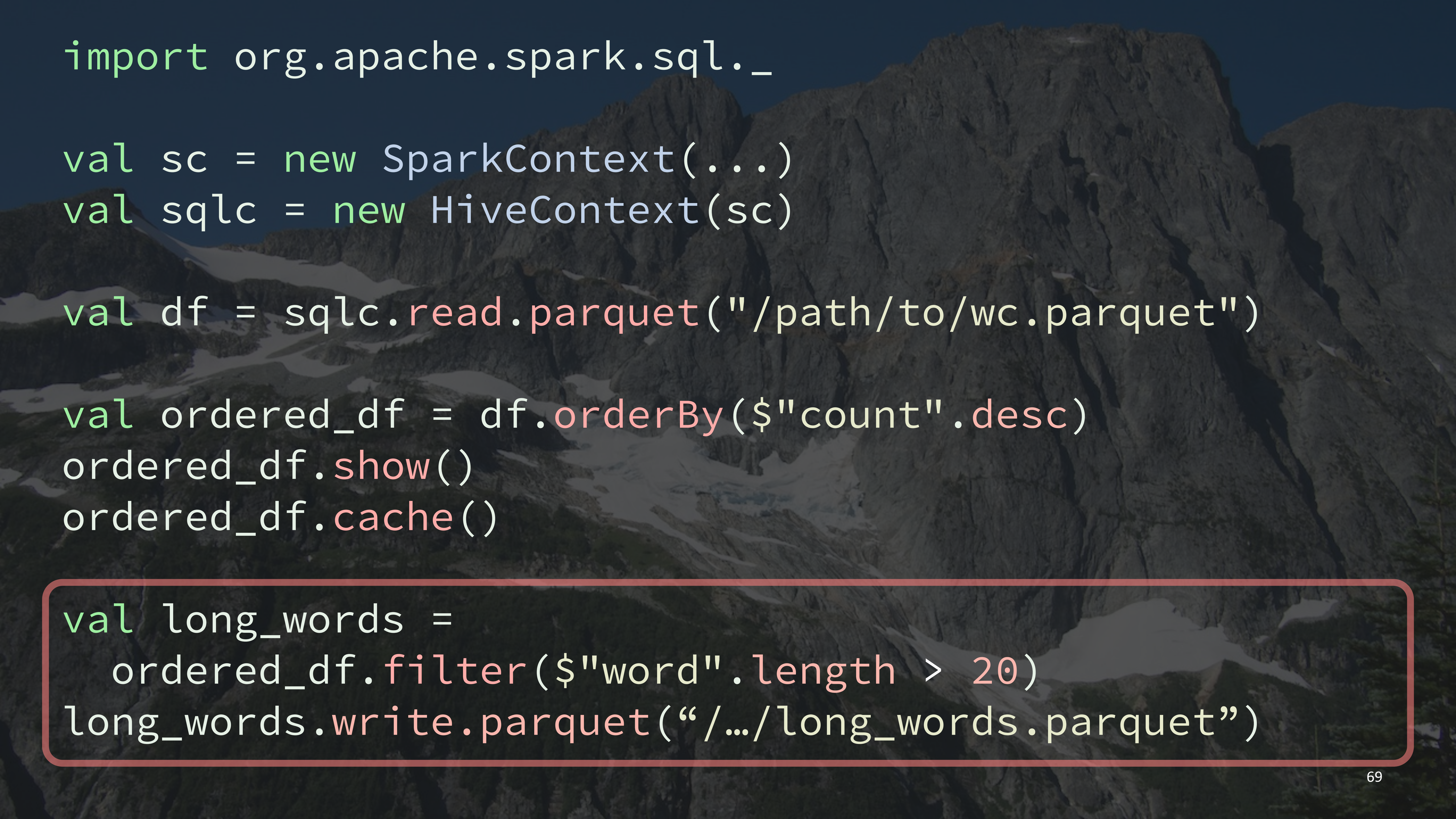
```
import org.apache.spark.sql._

val sc = new SparkContext(...)
val sqlc = new HiveContext(sc)

val df = sqlc.read.parquet("/path/to/wc.parquet")

val ordered_df = df.orderBy($"count".desc)
ordered_df.show()
ordered_df.cache()

val long_words =
  ordered_df.filter($"word".length > 20)
long_words.write.parquet(“/…/long_words.parquet”)
```



```
import org.apache.spark.sql._

val sc = new SparkContext(...)
val sqlc = new HiveContext(sc)

val df = sqlc.read.parquet("/path/to/wc.parquet")

val ordered_df = df.orderBy($"count".desc)
ordered_df.show()
ordered_df.cache()

val long_words =
  ordered_df.filter($"word".length > 20)
long_words.write.parquet("/…/long_words.parquet")
```



```
import org.apache.spark.sql._

val sc = new SparkContext(...)
val sqlc = new HiveContext(sc)

val df = sqlc.read.parquet("/path/to/wc.parquet")

val ordered_df = df.orderBy($"count".desc)
ordered_df.show()
ordered_df.cache()

val long_words =
  ordered_df.filter($"word".length > 20)
long_words.write.parquet(“/…/long_words.parquet”)
```



```
import org.apache.spark.sql._

val sc = new SparkContext(...)
val sqlc = new HiveContext(sc)

val df = sqlc.read.parquet("/path/to/wc.parquet")

val ordered_df = df.orderBy($"count".desc)
ordered_df.show()
ordered_df.cache()

val long_words =
  ordered_df.filter($"word".length > 20)
long_words.write.parquet(“/…/long_words.parquet”)
```

Machine Learning

MLlib

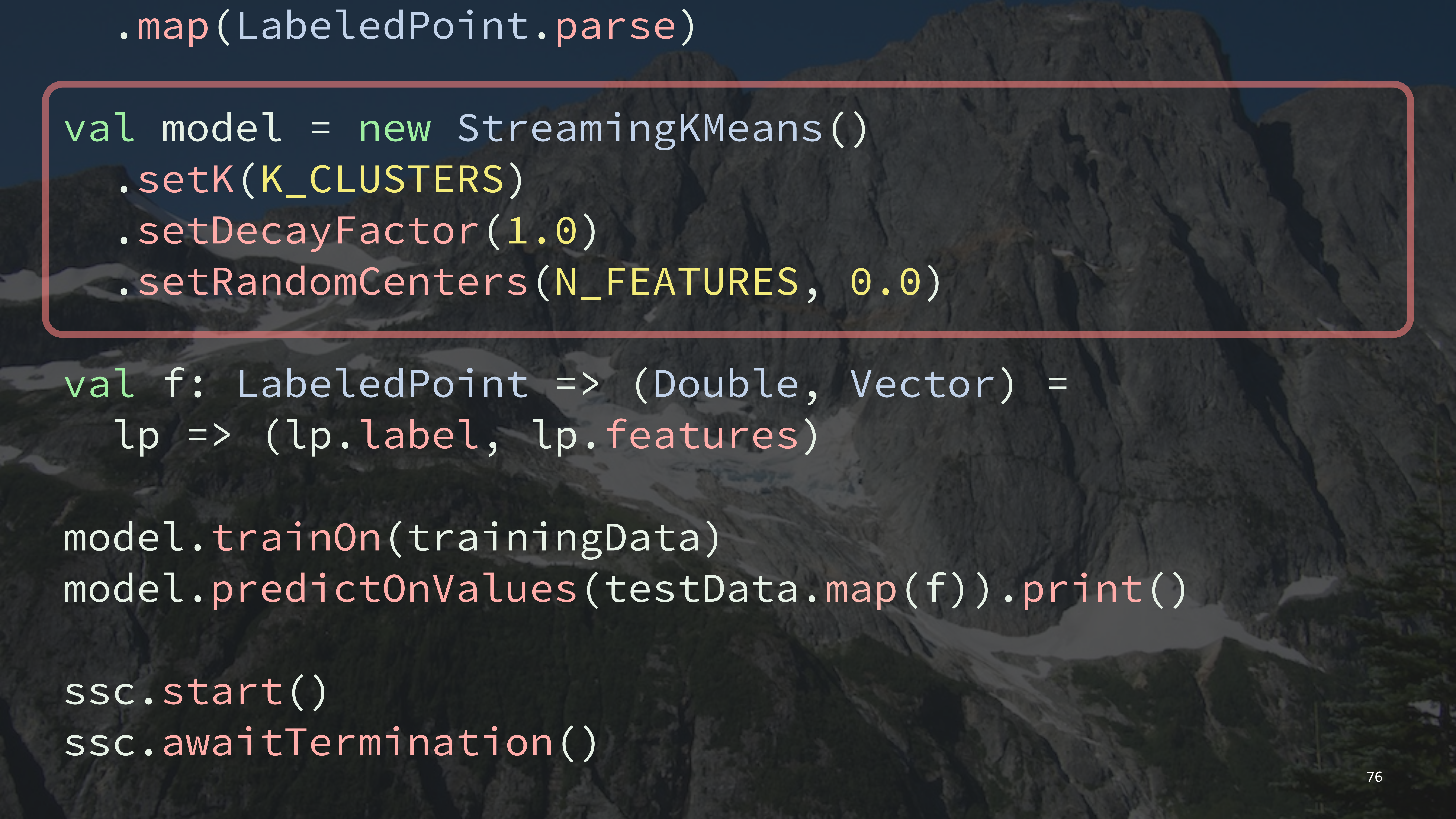
Streaming KMeans Example

```
import ...spark.mllib.clustering.StreamingKMeans
import ...spark.mllib.linalg.Vectors
import ...spark.mllib.regression.LabeledPoint
import ...spark.streaming.{
 Seconds, StreamingContext}

val sc = new SparkContext(...)
val ssc = new StreamingContext(sc, Seconds(10))

val trainingData = ssc.textFileStream(...)
  .map(Vectors.parse)
val testData = ssc.textFileStream(...)
  .map(LabeledPoint.parse)

val model = new StreamingKMeans()
  .setK(K_CLUSTERS)
```

```
import ...spark.mllib.clustering.StreamingKMeans
import ...spark.mllib.linalg.Vectors
import ...spark.mllib.regression.LabeledPoint
import ...spark.streaming.{
 Seconds, StreamingContext}

val sc = new SparkContext(...)
val ssc = new StreamingContext(sc, Seconds(10))

val trainingData = ssc.textFileStream(...)
  .map(Vectors.parse)
val testData = ssc.textFileStream(...)
  .map(LabeledPoint.parse)

val model = new StreamingKMeans()
  .setK(K_CLUSTERS)
```

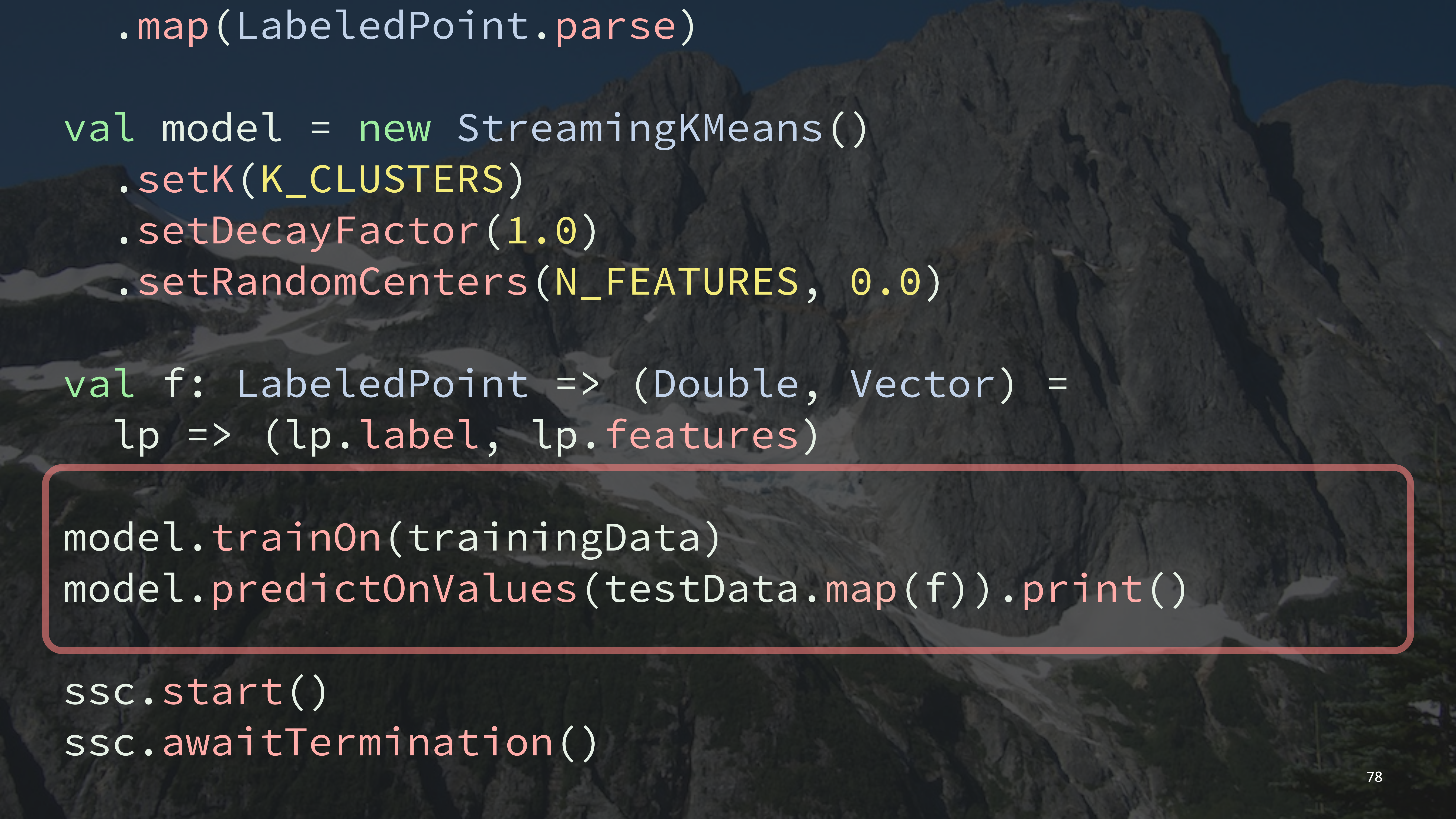
```
import ...spark.mllib.clustering.StreamingKMeans
import ...spark.mllib.linalg.Vectors
import ...spark.mllib.regression.LabeledPoint
import ...spark.streaming.{
 Seconds, StreamingContext}

val sc = new SparkContext(...)
val ssc = new StreamingContext(sc, Seconds(10))

val trainingData = ssc.textFileStream(...)
  .map(Vectors.parse)
val testData = ssc.textFileStream(...)
  .map(LabeledPoint.parse)

val model = new StreamingKMeans()
  .k(K_CLUSTERS)
```

```
import ...spark.mllib.clustering.StreamingKMeans
import ...spark.mllib.linalg.Vectors
import ...spark.mllib.regression.LabeledPoint
import ...spark.streaming.{
 Seconds, StreamingContext}

val sc = new SparkContext(...)
val ssc = new StreamingContext(sc, Seconds(10))

val trainingData = ssc.textFileStream(...)
  .map(Vectors.parse)
val testData = ssc.textFileStream(...)
  .map(LabeledPoint.parse)

val model = new StreamingKMeans()
  .k(K_CLUSTERS)
```

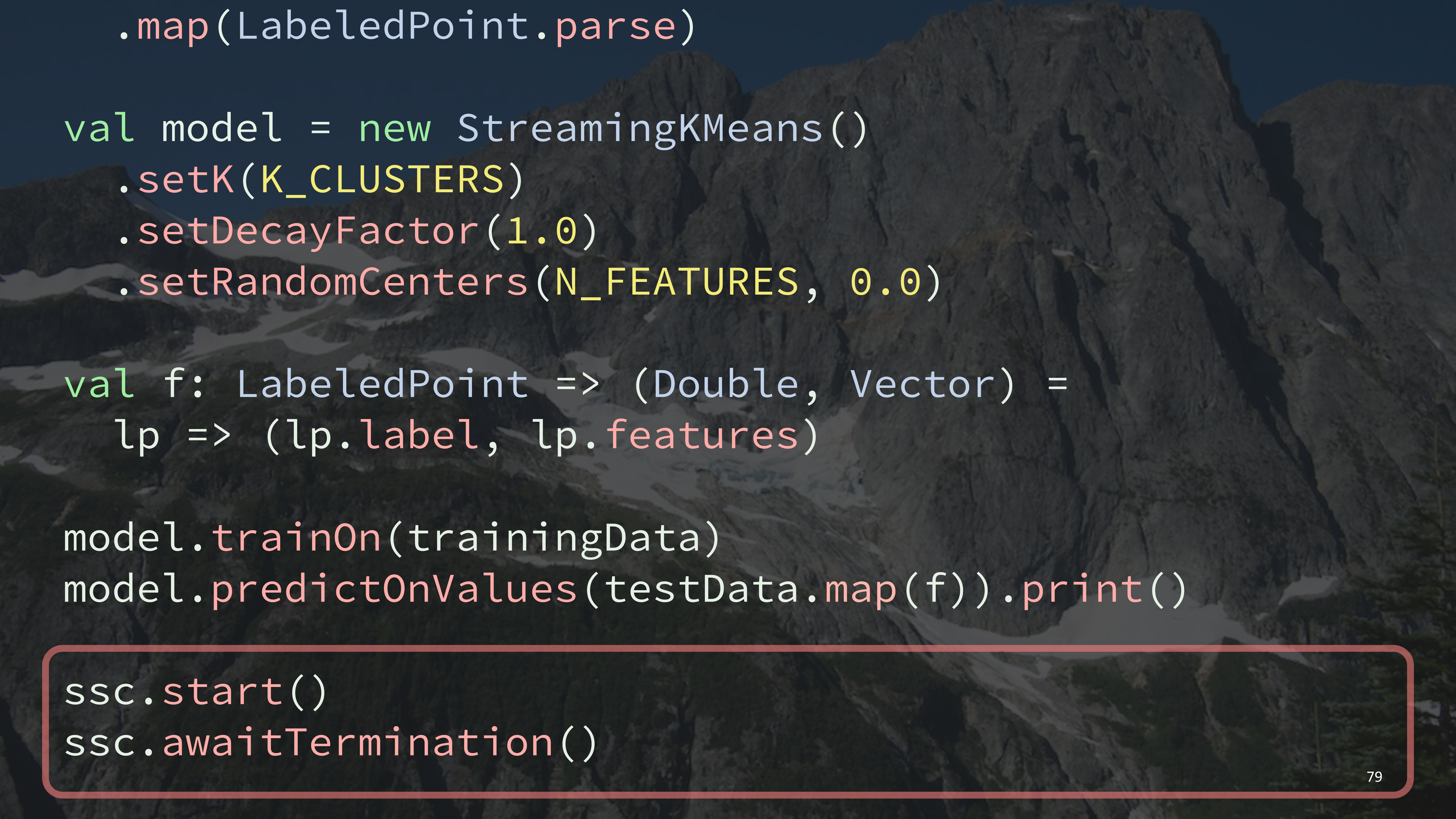


```
  .map(LabeledPoint.parse)

val model = new StreamingKMeans()
  .setK(K_CLUSTERS)
  .setDecayFactor(1.0)
  .setRandomCenters(N_FEATURES, 0.0)

val f: LabeledPoint => (Double, Vector) =
  lp => (lp.label, lp.features)

model.trainOn(trainingData)
model.predictOnValues(testData.map(f)).print()

ssc.start()
ssc.awaitTermination()
```

```
    .map(LabeledPoint.parse)

  val model = new StreamingKMeans()
    .setK(K_CLUSTERS)
    .setDecayFactor(1.0)
    .setRandomCenters(N_FEATURES, 0.0)

  val f: LabeledPoint => (Double, Vector) =
    lp => (lp.label, lp.features)

  model.trainOn(trainingData)
  model.predictOnValues(testData.map(f)).print()

  ssc.start()
  ssc.awaitTermination()
```



```
  .map(LabeledPoint.parse)

val model = new StreamingKMeans()
  .setK(K_CLUSTERS)
  .setDecayFactor(1.0)
  .setRandomCenters(N_FEATURES, 0.0)

val f: LabeledPoint => (Double, Vector) =
  lp => (lp.label, lp.features)

model.trainOn(trainingData)
model.predictOnValues(testData.map(f)).print()

ssc.start()
ssc.awaitTermination()
```



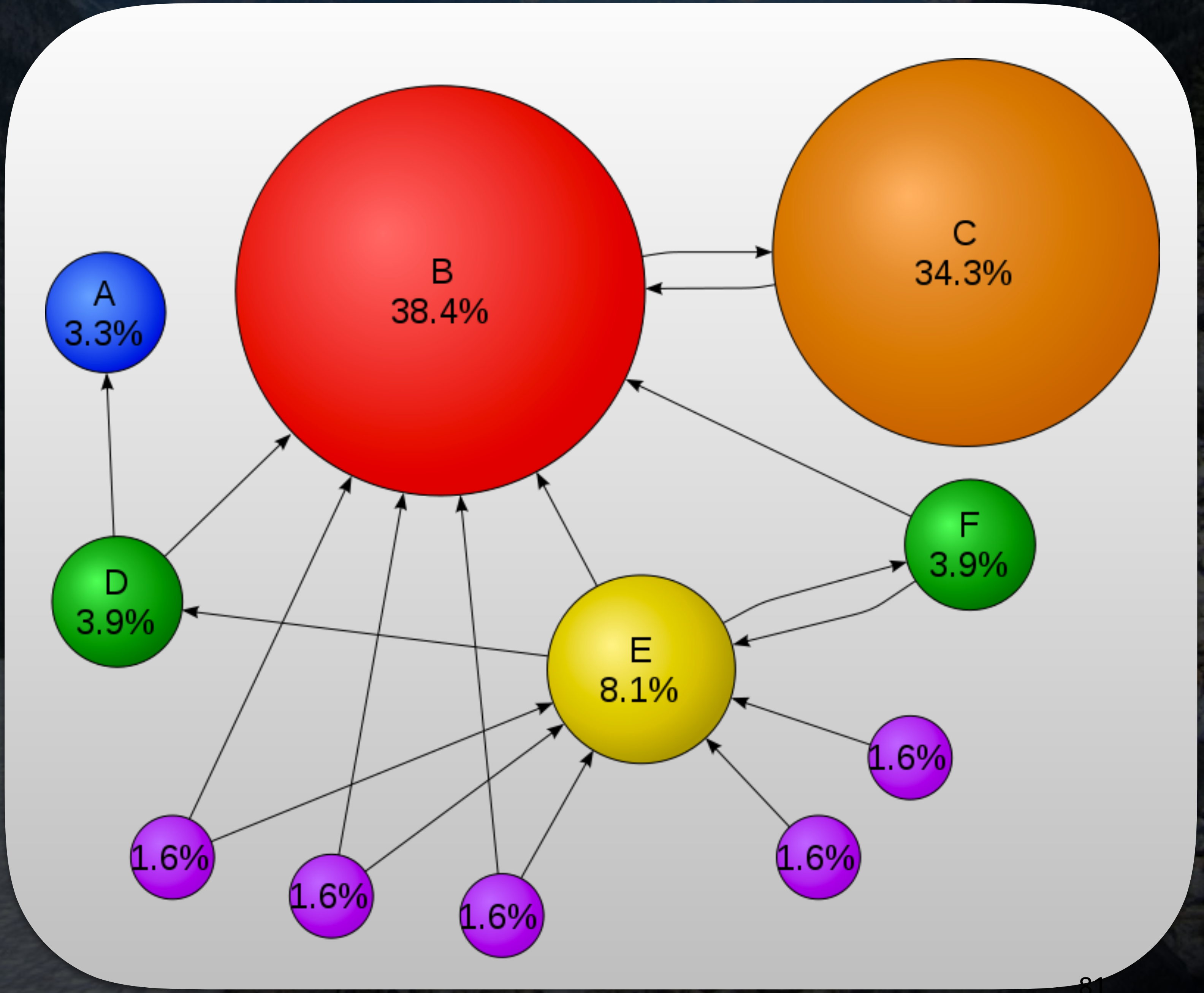
```
  .map(LabeledPoint.parse)

val model = new StreamingKMeans()
  .setK(K_CLUSTERS)
  .setDecayFactor(1.0)
  .setRandomCenters(N_FEATURES, 0.0)

val f: LabeledPoint => (Double, Vector) =
  lp => (lp.label, lp.features)

model.trainOn(trainingData)
model.predictOnValues(testData.map(f)).print()

ssc.start()
ssc.awaitTermination()
```

GraphX

Graph Processing

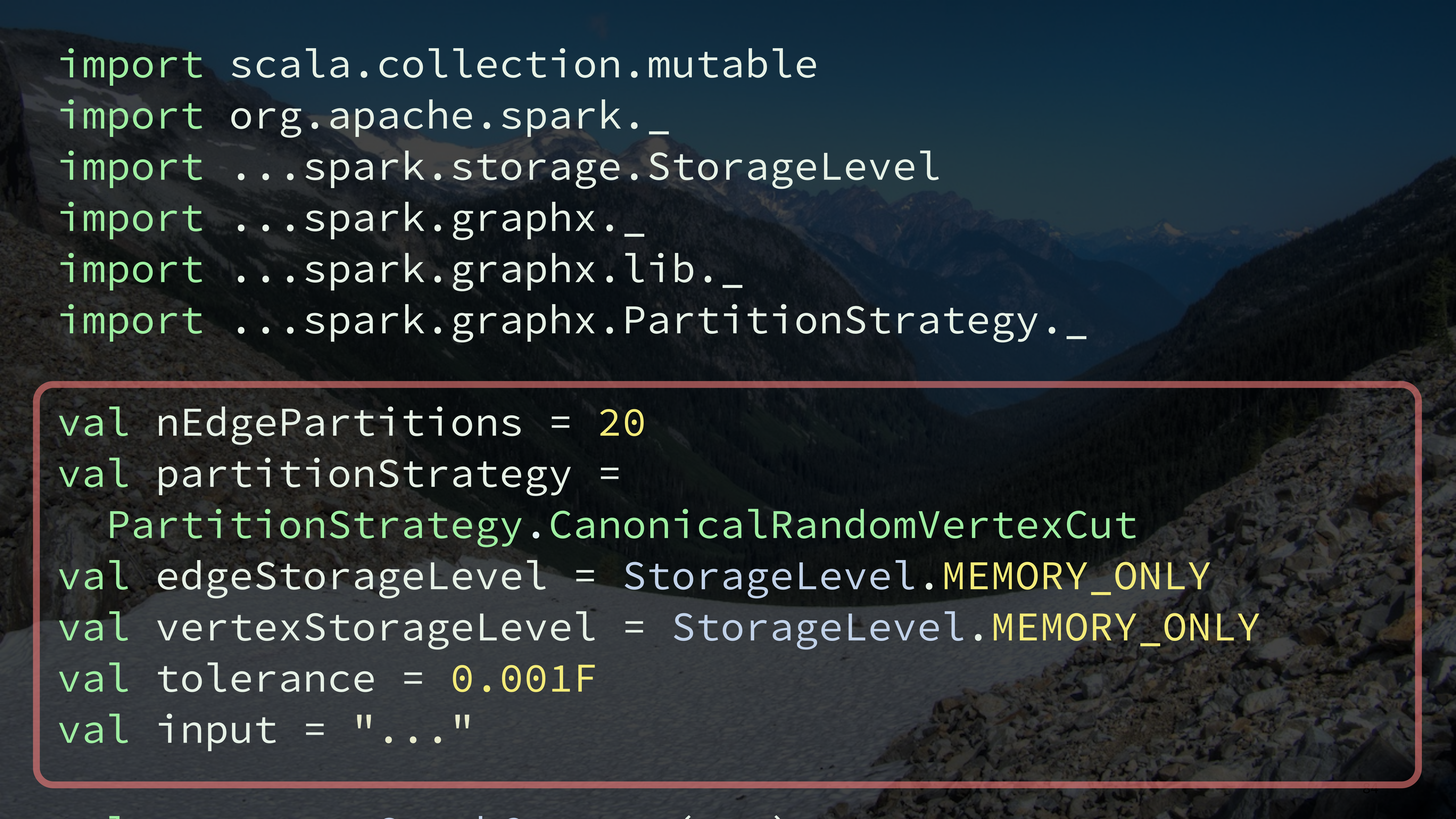
GraphX

- Social networks
- Epidemics
- Teh Interwebs
  - "Page Rank"
- …



```
import scala.collection.mutable
import org.apache.spark._
import ...spark.storage.StorageLevel
import ...spark.graphx._
import ...spark.graphx.lib._
import ...spark.graphx.PartitionStrategy._

val nEdgePartitions = 20
val partitionStrategy =
  PartitionStrategy.CanonicalRandomVertexCut
val edgeStorageLevel = StorageLevel.MEMORY_ONLY
val vertexStorageLevel = StorageLevel.MEMORY_ONLY
val tolerance = 0.001F
val input = "..."
```

```
import scala.collection.mutable
import org.apache.spark._
import ...spark.storage.StorageLevel
import ...spark.graphx._
import ...spark.graphx.lib._
import ...spark.graphx.PartitionStrategy._

val nEdgePartitions = 20
val partitionStrategy =
  PartitionStrategy.CanonicalRandomVertexCut
val edgeStorageLevel = StorageLevel.MEMORY_ONLY
val vertexStorageLevel = StorageLevel.MEMORY_ONLY
val tolerance = 0.001F
val input = "..."
```



```
import scala.collection.mutable
import org.apache.spark._
import ...spark.storage.StorageLevel
import ...spark.graphx._
import ...spark.graphx.lib._
import ...spark.graphx.PartitionStrategy._

val nEdgePartitions = 20
val partitionStrategy =
  PartitionStrategy.CanonicalRandomVertexCut
val edgeStorageLevel = StorageLevel.MEMORY_ONLY
val vertexStorageLevel = StorageLevel.MEMORY_ONLY
val tolerance = 0.001F
val input = "..."
val tolerance = 0.001f
val input = "..."

val sc = new SparkContext(...)

val unpartitionedGraph = GraphLoader.edgeListFile(
  sc, input, numEdgePartitions,
  edgeStorageLevel, vertexStorageLevel).cache

val graph = partitionStrategy.foldLeft(
  unpartitionedGraph)(_.partitionBy(_))
println("# vertices = " + graph.vertices.count)
println("# edges = " + graph.edges.count)

val pr = PageRank.runUntilConvergence(
  graph, tolerance).vertices.cache()
```

```
val tolerance = 0.001f
val input = "..."

val sc = new SparkContext(...)

val unpartitionedGraph = GraphLoader.edgeListFile(
  sc, input, numEdgePartitions,
  edgeStorageLevel, vertexStorageLevel).cache

val graph = partitionStrategy.foldLeft(
  unpartitionedGraph)(_.partitionBy(_))
println("# vertices = " + graph.vertices.count)
println("# edges = " + graph.edges.count)

val pr = PageRank.runUntilConvergence(
  graph, tolerance).vertices.cache()
  unpartitionedGraph)(_.partitionBy(_))
println("# vertices = " + graph.vertices.count)
println("# edges = " + graph.edges.count)

val pr = PageRank.runUntilConvergence(
  graph, tolerance).vertices.cache()

println("Top ranks: “)
pr.sortBy(tuple => -tuple._2, ascending=false).
  foreach(println)

pr.map {
  case (id, r) => id + "\t" + r
}.saveAsTextFile(...)
sc.stop()
```

```
  unpartitionedGraph)(_.partitionBy(_))
println("# vertices = " + graph.vertices.count)
println("# edges = " + graph.edges.count)

val pr = PageRank.runUntilConvergence(
  graph, tolerance).vertices.cache()

println("Top ranks: “)
pr.sortBy(tuple => -tuple._2, ascending=false).
  foreach(println)
```

```
pr.map {
  case (id, r) => id + "\t" + r
}.saveAsTextFile(...)
sc.stop()
```

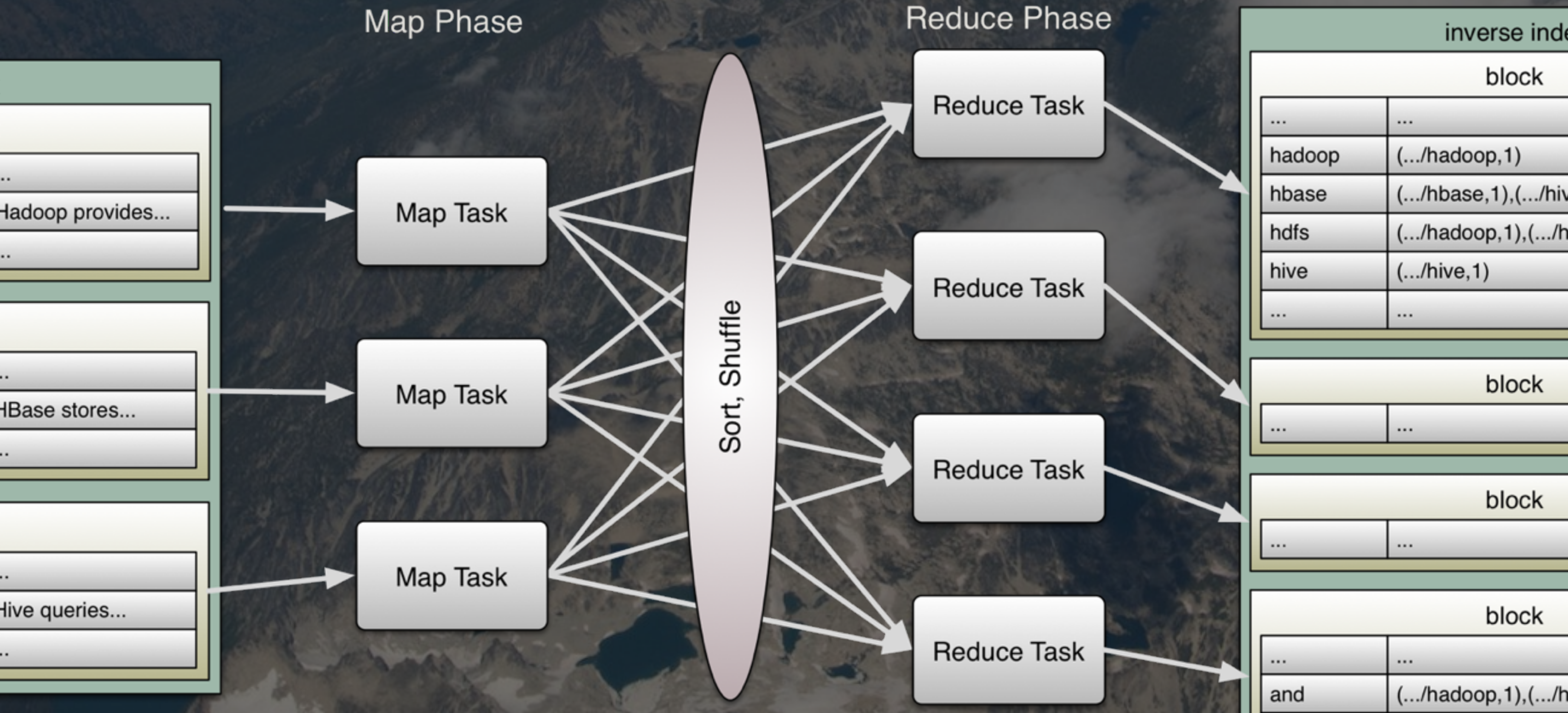
Thank You!

dean.wampler@lightbend.com
@deanwampler
polyglotprogramming.com/talks

Lightbend

Bonus Slides: Details of MapReduce implementation for the Inverted Index

1 Map step + 1 Reduce step



...
Hadoop provides...
...

...
HBase stores...
...

...
Hive queries...
...

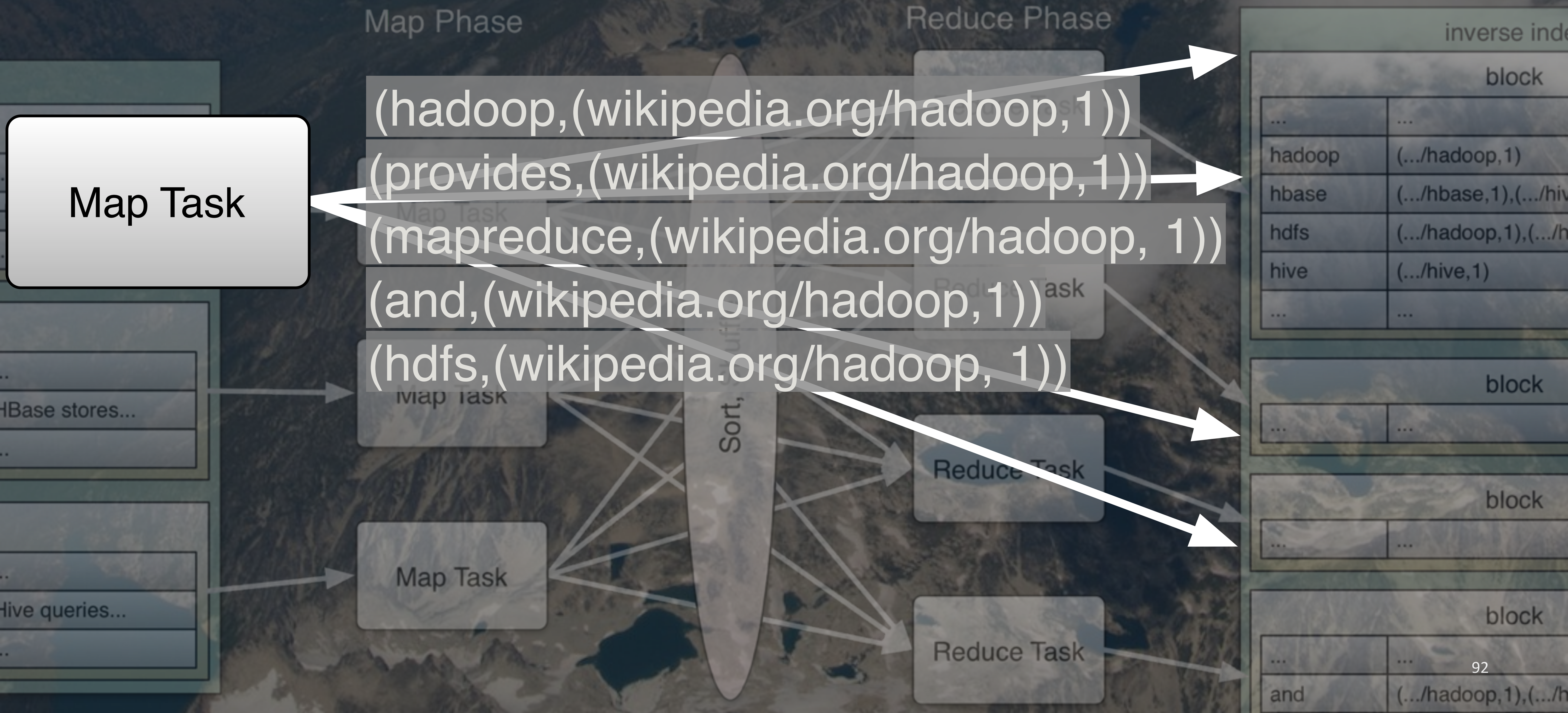
...	...
hadoop	(.../hadoop,1)
hbase	(.../hbase,1),(.../hiv
hdfs	(.../hadoop,1),(.../h
hive	(.../hive,1)
...	...

...	...

...	...

...	...
and	(.../hadoop,1),(.../h

1 Map step + 1 Reduce step

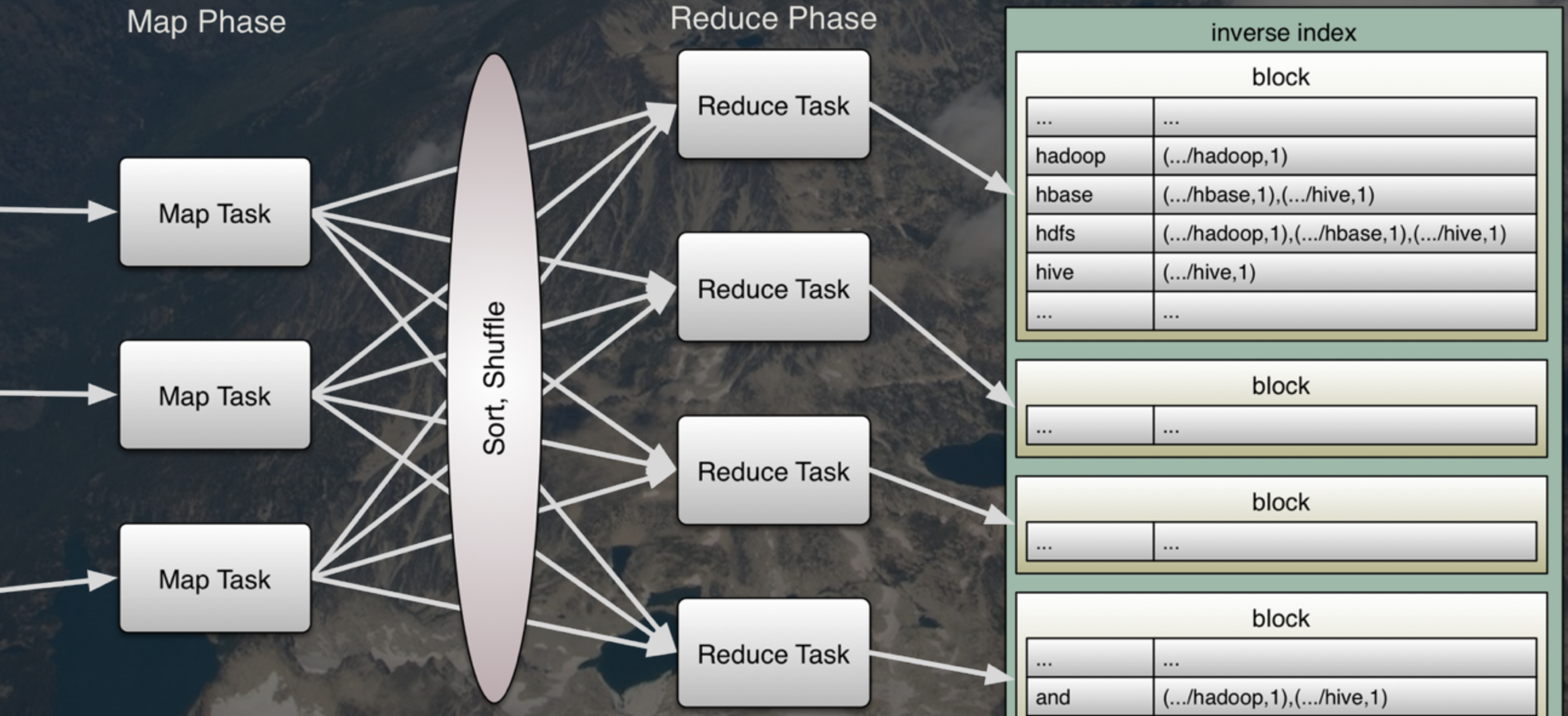


(hadoop,(wikipedia.org/hadoop,1))
(provides,(wikipedia.org/hadoop,1))
(mapreduce,(wikipedia.org/hadoop, 1))
(and,(wikipedia.org/hadoop,1))
(hdfs,(wikipedia.org/hadoop, 1))

...	...
hadoop	(.../hadoop,1)
hbase	(.../hbase,1),(.../hiv
hdfs	(.../hadoop,1),(.../h
hive	(.../hive,1)
...	...

...	...
and	(.../hadoop,1),(.../h

1 Map step + 1 Reduce step



block

...	...
hadoop	(.../hadoop,1)
hbase	(.../hbase,1),(.../hive,1)
hdfs	(.../hadoop,1),(.../hbase,1),(.../hive,1)
hive	(.../hive,1)
...	...

block

...	...

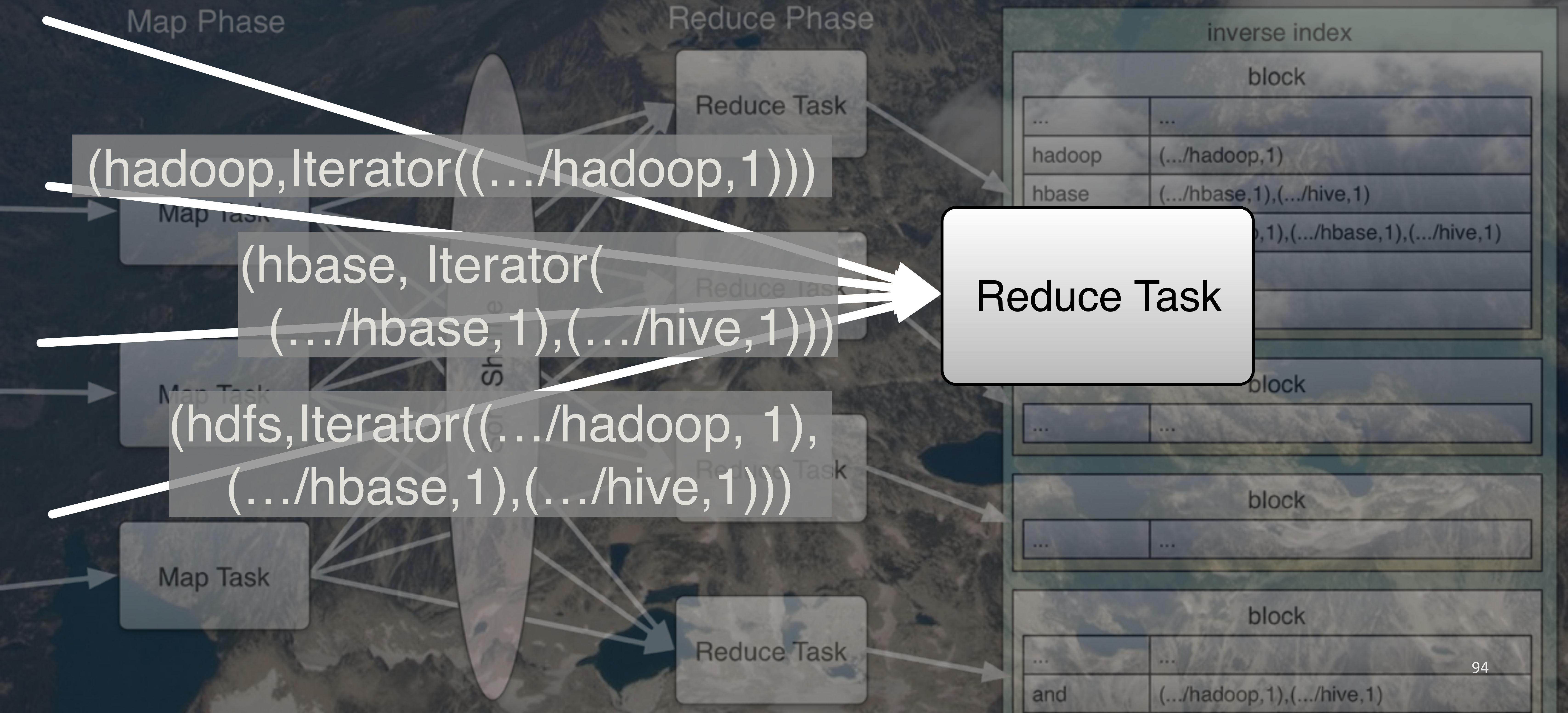
block

...	...

block

...	...
and	(.../hadoop,1),(.../hive,1)

1 Map step + 1 Reduce step



(hadoop,Iterator((…/hadoop,1)))

(hbase, Iterator(
(…/hbase,1),(…/hive,1)))

(hdfs,Iterator((…/hadoop, 1),
(…/hbase,1),(…/hive,1)))

inverse index

block

...	...
hadoop	(.../hadoop,1)
hbase	(.../hbase,1),(.../hive,1)
	,1),(.../hbase,1),(.../hive,1)

block

...	...

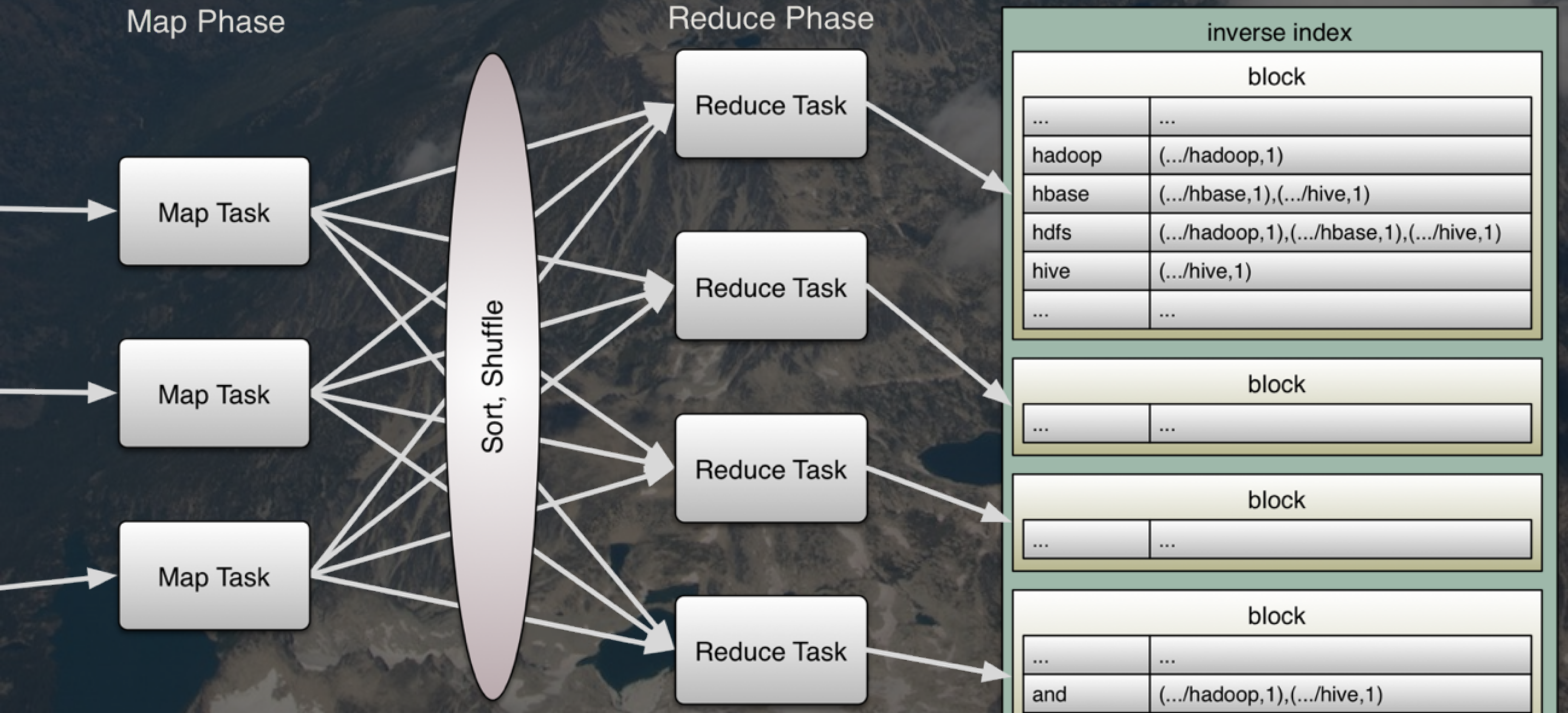
block

...	...

block

...	...
and	(.../hadoop,1),(.../hive,1)

1 Map step + 1 Reduce step



inverse index

block

...	...
hadoop	(.../hadoop,1)
hbase	(.../hbase,1),(.../hive,1)
hdfs	(.../hadoop,1),(.../hbase,1),(.../hive,1)
hive	(.../hive,1)
...	...

block

...	...

block

...	...

block

...	...
and	(.../hadoop,1),(.../hive,1)