Interfaces^3

Reynold Xin
Aug 22, 2014 @ Databricks Retreat

Repurposed Jan 27, 2015 for Spark community

Spark's two improvements over Hadoop MR

- Performance: "100X" faster than Hadoop MR

- Programming model: easier to use

Hadoop MR

```
public static class WordCountMapClass extends MapReduceBase
  implements Mapper<LongWritable, Text, Text, IntWritable> {

  private final static IntWritable one = new IntWritable(1);
  private Text word = new Text();

  public void map(LongWritable key, Text value,
                  OutputCollector<Text, IntWritable> output,
                  Reporter reporter) throws IOException {
    String line = value.toString();
    StringTokenizer itr = new StringTokenizer(line);
    while (itr.hasMoreTokens()) {
      word.set(itr.nextToken());
      output.collect(word, one);
    }
  }
}

public static class WorkdCountReduce extends MapReduceBase
  implements Reducer<Text, IntWritable, Text, IntWritable> {

  public void reduce(Text key, Iterator<IntWritable> values,
                     OutputCollector<Text, IntWritable> output,
                     Reporter reporter) throws IOException {
    int sum = 0;
    while (values.hasNext()) {
      sum += values.next().get();
    }
    output.collect(key, new IntWritable(sum));
  }
}
```

Spark

```
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
val spark = new SparkContext(master, appName, [sparkHome], [jars])
val file = spark.textFile("hdfs://...")
val counts = file.flatMap(line => line.split(" "))
                 .map(word => (word, 1))
                 .reduceByKey(_ + _)
counts.saveAsTextFile("hdfs://...")
```

"It has been the easiest learning experience that I went through"

- Burak coerced by Reynold

- Undergrad CS education cares more about implementation of functionality

- PhD research cares more about prototyping and validating ideas

- Neither requires thinking hard about interface design

"The most important aspect of any module is not how it implements the facilities it provides, but the way in which it provides those facilities in the first place."

– Damian Conway on "Ten Essential Development Practices"

in·ter·face
/ˈintərˌfās/ 🔊

noun

1. a point where two systems, subjects, organizations, etc., meet and interact.
   "the **interface between** accountancy and the law"

2. COMPUTING

   a device or program enabling a user to communicate with a computer.

verb

1. interact with (another system, person, organization, etc.).
   "his goal is to get people interfacing with each other"

2. COMPUTING

   connect with (another computer or piece of equipment) by an interface.

Example of Interfaces

- public programming APIs (e.g. RDD)

- external modules we expose (matplotlib)

- default imports in notebooks

- internal module methods (e.g. tree store)

- command line arguments

- configuration options

Why is interface design important?

- If you write code, you are already doing design

- Interfaces can be our biggest asset

- or biggest liabilities!

Public Interfaces as Assets

- Great public interfaces capture emotions and in turn capture customers

- Customers invest heavily in (public) interfaces

  - Cost of switching interfaces is HIGH: rewriting & retraining

  - **Network effect**: each "customer" brings value to another by writing apps and talking about it

Internal Interfaces as Assets

- Great internal interfaces capture emotions and in turn capture developers

- Developers reinforce our leadership

- Well designed internal interfaces enable us to move faster

  - e.g. compression codec vs connection manager

Interfaces as Liabilities

- Bad public interfaces increase support burden

  - groupByKey anyone?

- Bad internal interfaces increase cost of maintenance and innovation

Good Interfaces

- Easy to learn & use

- Sufficiently powerful

- Anticipating an inability to know future needs

- Backward compatible

"Other than hiring Reza and buying him drinks,
how do I get better at it?"

–Andy Konwinski

Process

1. Identify modules: **separation of concerns**

2. For each module: don't sweat implementation details but take time to **identify** interfaces, **minimize** them, and think how they **evolve**

3. Design, prototype & **program using** the interfaces

4. Write out a short **design doc** and ask for feedback

5. Implement the interface, and re-iterate

Guidelines

Keep it simple, stupid (KISS)

- Easier to learn / use

- Easier to document

- Easier to implement (less bugs)

- Easier to optimize narrow interfaces

- Easier to throw out / re-implement

- Easier to support long term

Ways to Simplify Design

Remove

Hide

Organize

Displace

Ways to Simply Design

Remove: Get rid of anything that isn't essential to the application. This could mean content, too; like the language you use in the navigation labels.

Organize: Arrange the elements of the interface so that they fit into logical chunks. This might mean based on a person's mental model (how they think), or tie in to a more familiar interface pattern.

Hide: Place the most important elements within reach (make them obvious), and hide the others, making them accessible through navigation.

Displace: Pushing some of the functionality to another device, or feature, so that the one interface isn't responsible for displaying every possible interaction.

Name Matters

- Class, variable, method names should be self-explanatory

- Avoid cryptic names (e.g. operator overloading)

- Be consistent

Periodic Table of Dispatch Operators

All operators of Scala's marvelous Dispatch library on a single page

Handle content directly

∧	<<? (values)	POST	>> ((in, charset) => result)	as_source
:/ (host, port)	/ (path)	PUT	>> ((in) => result)	as_str
:/ (host)	<<< (text)	DELETE	>~ ((source) => result)	>>> (out)
/ (path)	<<< (file, content_type)	HEAD	> ((text) => result)	>:> ((map) => result)
url (url)	<<< (values)	secure	>>~ ((reader) => result)	>+ (block)
	<< (text)	<& (request)	<> ((elem) => result)	~> ((conversion) => result)
	<< (values)	>\ (charset)	</> ((nodeseq) => result)	>+> (block)
	<< (text, content_type)	to_uri	># ((json) => result)	>! (listener)
	<< (bytes)		>|	
	<:< (map)			
	as (user, passwd)			
	as_! (user, passwd)			
	gzip			

Bad Examples in Spark

ExecutorLauncher

ExecutorRunner

DriverRunner

DriverWrapper

Client

Client (another one)

Client Base

AppClient

Bad Examples in Spark

ExecutorLauncher	yarn-client
ExecutorRunner	standalone
DriverRunner	standalone
DriverWrapper	standalone
Client	standalone
Client (another one)	yarn
Client Base	yarn
AppClient	standalone

Documentation Matters

```scala
26 ▼ /**
27    * A heartbeat.
28    */
29 ▼ private[spark] case class Heartbeat(
30       executorId: String,
31       taskMetrics: Array[(Long, TaskMetrics)],
32       blockManagerId: BlockManagerId)
```

```scala
def executorHeartbeatReceived(
    execId: String,
    taskMetrics: Array[(Long, Int, Int, TaskMetrics)],
    blockManagerId: BlockManagerId): Boolean = {
```

Documentation Matters

```scala
26 ▼  /**
27     * A heartbeat from executors to the driver. This is a shared message
28     * used by several internal components to convey liveness or execution
29     * |information for in-progress tasks.
30     */
31 ▼  private[spark] case class Heartbeat(
32        executorId: String,
33        taskMetrics: Array[(Long, TaskMetrics)], // taskId -> TaskMetrics
34        blockManagerId: BlockManagerId)
```

+ Explicit typing for public interfaces also part of the doc

Minimize Accessibility

- Make classes and members as private as possible, even for internal modules

- This maximizes **information hiding**

- Enables modules to be used, understood, built, tested, and debugged independently

- A bad habit of many Scala developers to leave everything wide open

```scala
/**
 * :: DeveloperApi ::
 * Tracks task-level information to be displayed in the UI.
 *
 * All access to the data structures in this class must be synchronized on the
 * class, since the UI thread and the EventBus loop may otherwise be reading and
 * updating the internal data structures concurrently.
 */
@DeveloperApi
class JobProgressListener(conf: SparkConf) extends SparkListener with Logging {

  import JobProgressListener._

  // How many stages to remember
  val retainedStages = conf.getInt("spark.ui.retainedStages", DEFAULT_RETAINED_STAGES)

  // Map from stageId to StageInfo
  val activeStages = new HashMap[Int, StageInfo]

  // Map from (stageId, attemptId) to StageUIData
  val stageIdToData = new HashMap[(Int, Int), StageUIData]

  val completedStages = ListBuffer[StageInfo]()
  val failedStages = ListBuffer[StageInfo]()

  // Map from pool name to a hash map (map from stage id to StageInfo).
  val poolToActiveStages = HashMap[String, HashMap[Int, StageInfo]]()

  val executorIdToBlockManagerId = HashMap[String, BlockManagerId]()

  var schedulingMode: Option[SchedulingMode] = None

```

Principle of least astonishment

- Use your common sense; interfaces should not surprise users

- e.g. Tachyon format command accidentally deletes file

Composability

- LogisticRegressionWithSGD

- LogisticRegressionWithADMM

- LogisticRegressionWithLBFGS

- LogisticRegressionWithNewton

- LinearRegressionWithSGD

- …

Composability

- LogisticRegression.fit(data, method="admm")

Long-term Maintainability

- When in doubt, leave it out

  - Every interface added increases complexity

  - Easier to add than remove in the future

- Avoid exposing dependency on 3rd party libraries

  - e.g. MLlib's use of Breeze (+)

  - e.g. Spark's use of Guava Optional (-)

- Don't let implementation details impact interface design

- KISS

  - Remove, hide, organize, displace

  - Name matters

  - Documentations matter

  - Minimize accessibility

- Compose interfaces for expressivity

- Long-term maintainability

- …

Interface Design

- Years of effort; impossible to do overnight

- Critical in building out a strong platform

- Critical in ensuring the long-term pace of innovation

- We scored better than anybody else out there, but still a long way to go

References

- Eric S. Raymond, Basics of the Unix Philosophy http://www.faqs.org/docs/artu/ch01s06.html

- Joshua Bloch, How to Design a Good API and Why it Matters http://lcsd05.cs.tamu.edu/slides/keynote.pdf

- Richard Gabriel, The Rise of ``Worse is Better'' http://www.jwz.org/doc/worse-is-better.html (I don't actually agree with the article)