DiSh: Dynamic Shell-Script Distribution

Tammam Mustafa
Konstantinos Kallas
Pratyush Das
Nikos Vasilakis

binpa.sh
github.com/binpash/dish
Shells 🐚 are everywhere
Shells 🐚 are everywhere

from the 2022 state of the octoverse: https://octoverse.github.com
A general data processing script
An example: Temperature Analysis

```
TEMPS="temps.txt"
cat $TEMPS | cut -c 89-92 | grep -v 999 | sort -rn | head -n1 > max.txt
```

Takes < 1s
Data is exploding

Volume of data created and replicated worldwide (source: IDC)
An example: Temperature Analysis

- Data from National Oceanic and Atmospheric Administration (NOAA).
- Stored in HDFS (Hadoop distributed file system).

```bash
TEMPS="temps.txt"
HDFS dfs -cat $TEMPS | cut -c 89-92 | grep -v 999 | sort -rn | head -n1 > max.txt
```
The works but ...
TEMPS="temps.txt"

HDFS dfs -cat $TEMPS | cut -c 89-92 | grep -v 999 | sort -rn | head -n1 > max.txt

> 95%

< 5%

The works but ...

200s

from pyspark import SparkConf, SparkContext

# Configure and create Spark context
conf = SparkConf().setAppName("FindMaxTemp")
sc = SparkContext(conf=conf)

# Read the data from the HDFS file
data = sc.textFile("hdfs:///path/to/temps.txt")

# Extract the desired characters, filter out '999', and convert to integers
filtered_data = data.map(lambda line: line[88:92]).filter(lambda x: x != "999").map(int)

# Find the maximum value
max_temp = filtered_data.max()

# Save the result to a local file
with open("max.txt", "w") as output_file:
    output_file.write(str(max_temp))
TEMPS="temps.txt"
HDFS dfs -cat $TEMPS | cut -c 89-92 | grep -v 999 | sort -rn | head -n1 > max.txt

> 95%
< 5%

from pyspark import SparkConf, SparkContext

# Configure and create Spark context
conf = SparkConf().setAppName("FindMaxTemp")
sc = SparkContext(conf=conf)

# Read the data from the HDFS file
data = sc.textFile("hdfs:///path/to/temps.txt")

# Extract the desired characters, filter out '999', and convert to integers
filtered_data = data.map(lambda line: line[88:92]).filter(lambda x: x != 999).map(int)

# Find the maximum value
max_temp = filtered_data.max()

# Save the result to a local file
with open("max.txt", "w") as output_file:
    output_file.write(str(max_temp))

import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class MaxTemperature {
    public static class TokenizerMapper extends Mapper<Object, Text, IntWritable, IntWritable> {
        private IntWritable temperature = new IntWritable();

        public void map(Object key, Text value, Context context)
            throws IOException, InterruptedException {
            String line = value.toString();
            if (line.length() > 92) {
                int temp = Integer.parseInt(line.substring(88, 92));
                if (temp != 9999) {
                    temperature.set(temp);
                    context.write(new IntWritable(1), temperature);
                }
            }
        }
    }

    public static class IntMaxReducer extends Reducer<IntWritable, IntWritable, IntWritable, IntWritable> {
        public void reduce(IntWritable key, Iterable<IntWritable> values, Context context)
            throws IOException, InterruptedException {
            int maxValue = Integer.MIN_VALUE;
            for (IntWritable value : values) {
                maxValue = Math.max(maxValue, value.get());
            }
            context.write(key, new IntWritable(maxValue));
        }
    }

    public static void main(String[] args)
        throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "max temperature");
        job.setJarByClass(MaxTemperature.class);
        job.setMapperClass(TokenizerMapper.class);
        job.setCombinerClass(IntMaxReducer.class);
        job.setReducerClass(IntMaxReducer.class);
        job.setOutputKeyClass(IntWritable.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
TEMPS="temps.txt"

HDFS dfs -cat $TEMPS | cut -c 89-92 | grep -v 999 | sort -rn | head -n1 > max.txt
TEMPS="temps.txt"

HDFS dfs -cat $TEMPS | cut -c 89-92 | grep -v 999 | sort -rn | head -n1 > max.txt
Performance Comparison

On premise cluster -- 100 Mbps link speed -- 3.6gb data

- Bash: 200s
- PaSh: 190s
- Hadoop Streaming: 90s
- DiSh: 7s
DiSh

seq.sh

Dataflow conversion

Transformations

Dataflow Model

Cluster Workers

No tight coupling: Could work on top of any shell!
DiSh Overview

script

Dynamic Orchestration

User Shell

Compiler

Scheduler

Annotations

Node
  Worker
    Shell

Node
  Worker
    Shell
DiSh Overview

- Script
- Dynamic Orchestration
- User Shell
- Compiler
- Scheduler
- Annotations
- Node
  - Worker
    - Shell
  - Worker
    - Shell
The shell is extremely Dynamic

```
OUT=${OUT:-$TOP/out}
for input in $(ls ${IN}); do
cat "${IN}/$input" |
  tr -sc '[A-Z][a-z]' '[\012*]' |
  sort > "${OUT}/${input}.out"
done
```
DiSh Overview

- Script
- Dynamic Orchestration
- User Shell
- Compiler
- Scheduler
- Annotations
- Node
  - Worker
  - Shell
- Node
  - Worker
  - Shell
Dataflow Transformations

grep hello ./in.txt | sort

Order Aware Dataflow Model

An Order-aware Dataflow Model for Parallel Unix Pipelines
DiSh Overview

script

Dynamic Orchestration

User Shell

Compiler

Scheduler

Annotations

Node
  Worker
    Shell

Node
  Worker
    Shell
DiSh Overview

- Script
- Dynamic Orchestration
- User Shell
- Compiler
- Scheduler
- Annotations
- Node
- Worker
- Shell
- Node
- Worker
- Shell
DiSh Overview

- Script
- Dynamic Orchestration
- User Shell
- Compiler
- Scheduler
- Annotations
- Node
  - Worker
  - Shell
- Node
  - Worker
  - Shell
DISH in Action - Temperature Analysis

TEMPS="temps.txt"

HDFS dfs -cat $TEMPS | cut -c 89-92 | grep -v 999 | sort -rn | head -n1 > max.txt

Data flow transformation
DISH in Action - Temperature Analysis

HDFS file expansion

```
/temps.txt → hdfs cat → cut → grep → sort → head → max.txt
```

```
/n1/block1 → cut → grep → sort → head → max.txt
/n2/block2
```
DISH in Action - Temperature Analysis

/n1/block1

cut

grep

sort

head

max.txt

Parallelization

/n2/block2

/cut

/grep

/sort -m

/head

max.txt
DISH in Action - Temperature Analysis

Graph splitting and worker assignment

Worker 1

Worker 2

Host

max.txt
Data movement

Worker 1:
/n1/block1
  cut → grep → sort
     → sort -m → head

Worker 2:
/n2/block2
  cut → grep → sort

Host:
max.txt
<100M
1 line
Performance Results

- Hadoop Streaming: 7.2x avg speedup
- DiSh: 13.6x avg speedup
Not only fast but also correct!

Out of the 408 tests:

- DiSh and Bash only differ in 2 tests.
- Both return with an error, though different code

<table>
<thead>
<tr>
<th>Command</th>
<th>Bash and X differ</th>
</tr>
</thead>
<tbody>
<tr>
<td>dash</td>
<td>20</td>
</tr>
<tr>
<td>ksh</td>
<td>22</td>
</tr>
<tr>
<td>mksh</td>
<td>29</td>
</tr>
<tr>
<td>yash</td>
<td>20</td>
</tr>
</tbody>
</table>
What DiSh doesn’t do
DiSh..
DiSh Offers

● A system that automatically distributes shell scripts
● Significant performance gains without developer effort
● Fully compatible with existing shells scripts

binpa.sh  github.com/binpash/dish