VE572 Homework4

Yihao Liu

July 23, 2018

Ex. 1 — Processes and cgroups

1. cgroups (abbreviated from control groups) is a Linux kernel feature that limits, accounts for, and isolates the resource usage (CPU, memory, disk I/O, network, etc.) of a collection of processes.
2.

differences: the resource usage of a cgroup can be configured but that of a process can't.

similarities: they are hierarchical, and child inherit certain attributes from their parent.

Ex. 2 — MapReduce

1. See the class `Map` in file `ve572.h4.ex2.MapReduce.java`
2. See the class `Reduce` in file `ve572.h4.ex2.MapReduce.java`
3. See the function `main` in file `ve572.h4.ex2.MapReduce.java`
4.

data size	5e4	5e5	5e6	5e7
streaming (s)	2.474	3.415	13.464	128.59
java (s)	1.200	2.170	9.184	122.22

Ex. 3 Avro

1. add `avro` in `pom.xml` of `maven`

```
<!-- https://mvnrepository.com/artifact/org.apache.avro/avro -->
<dependency>
     <groupId>org.apache.avro</groupId>
     <artifactId>avro</artifactId>
     <version>1.8.2</version>
</dependency>
```

2. `ve572.h4.ex3.avsc`

```
{
  "namespace": "ve572.h4.ex3.avro",
  "type": "record",
  "name": "Grade",
  "fields": [
    {
      "name": "name",
      "type": "string"
```

```
    },
    {
      "name": "id",
      "type": "string"
    },
    {
      "name": "score",
      "type": "int"
    }
  ]
}
```

3. See the file `ve572.h4.ex3.GradeAvro.java`
4. There are 3 ways Avro can be used in MapReduce (Mixed-mode, Record-based, and Key/Value-based modes).

- Mixed-mode - in cases where you have non Avro input and generate Avro outputs, or vice versa, in which case the Avro mapper and reducer classes aren't suitable. Use AvroWrapper class.
- Record-based - in this case Avro will be used end-to-end. As Avro isn't key/value format you should use specific Mapper (AvroMapper) and Reducer (AvroReducer) classes.
- Key/Value-based - you want to use Avro as native key/value format, use the AvroKeyValue, AvroKey, and AvroValue classes to work with Avro key/value data.

Ex. 4 — Bloom filters

1. A Bloom filter is a space-efficient probabilistic data structure, conceived by Burton Howard Bloom in 1970, that is used to test whether an element is a member of a set. False positive matches are possible, but false negatives are not – in other words, a query returns either "possibly in set" or "definitely not in set". Elements can be added to the set, but not removed (though this can be addressed with a "counting" filter); the more elements that are added to the set, the larger the probability of false positives.