**Assignment3**

**EX1 Using MapReduce to Process Semi-structured Data.**

1. We want a workflow that will produce a report of the number of users for each browser. We also assume that the input to the mapper is the linenumber (key) and the value is the text of that line. For simplicity, we assume that an IP address uniquely identifies a user.

Map Step:

BrowerUserMap:

For each input key-value pair (linenumber, text):

Output key-value pair (extract\_browser(text), extract\_IP(text))

* During this step, the system splits the input set into m partitions and creates m map tasks and gives each one partition. Each map task executes the map function on its partition and only completes once all map tasks are done.

Shuffle Step:

* System sorts map outputs by key and transforms all key-value pairs (k, v1), (k, v2)… with same key k to one key-value list pair (k, (v1, v2, …)) where k are browsers and v are IPs.

Reduce Step:

UserCountReduce:

For each input key-valueList pair (k, (v1, v2, … vn)):

If n > 100 : Output(k, n)

* During this step, system partitions output by key into r partitions and creates r reduce tasks and assigns each one partition. Each reduce task executes the above reduce function.

1. Here is an example of what the input and output of each mapper and reducer in the above workflow would look like.

**For Map Step:**

Input:

169.122.23.15 - frank [10/Oct/2000:13:55:36 -0700] ‘‘GET /apache pb.gif HTTP/1.0’’ 200 2326 ‘‘http://www.example.com/start.html’’ ‘‘Mozilla/4.08 [en] (Win98; I ;Nav)’’,

169.122.23.99 - frank [10/Oct/2000:13:55:36 -0700] ‘‘GET /apache pb.gif HTTP/1.0’’ 200 2326 ‘‘http://www.example.com/start.html’’ ‘‘Mozilla/4.08 [en] (Win98; I ;Nav)’’,

169.122.23.15 - frank [10/Oct/2000:13:55:36 -0700] ‘‘GET /apache pb.gif HTTP/1.0’’ 200 2326 ‘‘http://www.example.com/start.html’’ ‘’Chrome/4.08 [en] (Win98; I ;Nav)’’

(…)

Output : (Mozilla, 169.122.23.15), (Mozilla, 169.122.23.99) , (Chrome, 169.122.23.15) (…)

**For Shuffle Step:**

Input: (Mozilla, 169.122.23.15), (Mozilla, 169.122.23.99) , (Chrome, 169.122.23.15) (…)

Output: (Mozilla, (169.122.23.15, 169.122.23.99)), (Chrome, (169.122.23.15)) (…)

**For Reduce Step:**

*(For simplicity, we omit the > 100 condition here)*

Input: (Mozilla, (169.122.23.15, 169.122.23.99)), (Chrome, (169.122.23.15)) (…)

Output: (Mozilla, 2), (Chrome, 1), (…)

1. If we were allowed to use the combine functionality for the solution, it can reduce the amount of I/O. In fact, it is possible to use this functionality when the reduce function is commutative and associative which corresponds to our reduce function. We want to execute the reduce function at each mapper on partial result of the mapper. This way, we reduce data to be transferred by shuffle which reduces the amount of I/O.

In our case:

* At each mapper instance, we count the occurrence of each IP address for every browser for all documents read by this mapper instance.
* Then, for each mapper, there is only one key-value pair for each browser and the value is the number of users.

**EX2: Complex uses of MapReduce**

1. We want a workflow that will produce a report of the number of users for each browser. We also assume that the input to the mapper is the linenumber (key) and the value is the text of that line. For simplicity, we assume that an IP address uniquely identifies a user.

Map Step:

UserBrowserMap:

For each input key-value pair (linenumber, text):

Output key-value pair (extract\_IP(text), extract\_browser(text))

Example output: (169.122.23.15, Mozilla), (169.122.23.99,Mozilla) , (169.122.23.15, Chrome) (…)

* During this step, the system splits the input set into m partitions and creates m map tasks and gives each one partition. Each map task executes the map function on its partition and only completes once all map tasks are done. We are mapping all IPs to their corresponding browser.

Shuffle Step:

* System sorts map outputs by key and transforms all key-value pairs (k, v1), (k, v2)… with same key k to one key-value list pair (k, (v1, v2, …)) where k are browsers and v are IPs.

Example output: (169.122.23.15, (Mozilla, Chrome)), (169.122.23.99, (Mozilla))

Reduce Step:

The input is the output of the shuffle step

BrowserReduce:

For each input key-valueList pair (k, (v1, v1, v2, … vn)):

Output(k, v1, v2, … vn)

Example Output: (169.122.23.15, (Mozilla, Chrome)), (169.122.23.99, (Mozilla))

* During this step, the reduce task will remove any browser duplicate for each user.

UserCountReduce:

The input is the output of the previous reduce step

Count = 0

For each input key-valueList pair (k, (v1, v2, … vn)):

If n > 1: count += 1

Output Count

Example output: 1

* During this step, we count the number of users that uses more than one browser.

1. For the last reducer step of this MapReduce workflow, we will need a single reducer process since we want a single output (the number of users who use more than one browser).