- Sessionize the web log by IP. Sessionize = aggregrate all page hits by visitor/IP during a fixed time window. https://en.wikipedia.org/wiki/Session_(web_analytics (https://en.wikipedia.org/wiki/Session_(web_analytics))
- Determine the average session time
- Determine unique URL visits per session. To clarify, count a hit to a unique URL only once per session.
- Find the most engaged users, ie the IPs with the longest session times
- For this dataset, complete the sessionization by time window rather than navigation. Feel free to determine the best session window time on your own, or start with 15 minutes.
- The log file was taken from an AWS Elastic Load Balancer: http://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/access-log-collection.html#access-log-entry-format (http://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/access-log-collection.html#access-log-entry-format)

Notes

- This worksheet is completed within Jupyter using Spark Kernel that connects to a standalone Spark cluster. `spark context` is pre-loaded as `sc`.
- For simplicity, the session time is rendered as Int. This may result inaccuracy in computing the average session time. 15 minutes is used for the session window.

Procedure

- Three fields are needed from the orignial log in order to answer the questions, IP, URL, timestamp.
- One of the ways to sessionize the log is to re-organize the log into a collection of `<K,V>` pairs, where `K` is IP address and `V` is a list of `sessions`. Each `session` is a tuple 3 that includes `last_access_time`, `Set(URL, URL,...)`, and `total_session_time`.
- Step 1
  - Load the orignial log into `initRDD`, using IP as the key without the port number. The value is initially a tuple `(Long, String)`, in which the first element is the millisecond from epoch, converted from the timestamp, and the second element is the URL.
- Step 2
  - Run aggregateByKey and combine the tuples produced in previous step into `List[(Long, String)]`. The list elements are sorted by the millisecond from epoch. By the end of this step, the `<K,V>` pair is in the form

of <IP, List[(Long, String)]>

- Step 3 - sessionize
  - Run mapValues on each K. Tuples in the list are merged into a session if the difference of timestamp between two neighbouring tuples is less than 15 minutes, URLs are unioned into a set, and the total session time is accumulated. The final format of value V is ListBuffer[(Long, Set[String], Int)]. The elements of the tuple (Long, Set[String], Int) are last_access_time, Set(URL, URL,...), and total_session_time as described earlier.
- Step 4,5,6
  - These steps yield the final answers based on the sessionRDD. They are self-explanatory.

Step 1

In [1]:
```
val tmp = sc.textFile("~/data/paytmlab/Weblog/data/2015_07_22_mktplace_shop_web_log_sample.log")

val initRDD = tmp.map(s => (s.split(" ")(2).split(":")(0),
    (new java.text.SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSSSSS'Z'").parse(s.split(" ")(0)).getTime,
         s.split(" ")(12))))
```

In [4]:
```
initRDD.take(3) foreach println
```

```
(123.242.248.130,(1437570047143,https://paytm.com:443/shop/authresponse?code=f2405b05-e2ee-4b0d-8f6a-9fed0fcfe2e0&state=null))
(203.91.211.44,(1437570921580,https://paytm.com:443/shop/wallet/txnhistory?page_size=10&page_number=0&channel=web&version=2))
(1.39.32.179,(1437570912745,https://paytm.com:443/shop/wallet/txnhistory?page_size=10&page_number=0&channel=web&version=2))
```

Step 2

In [2]:
```
var reducedRDD = initRDD.aggregateByKey(List[(Long, String)]())(
    (acc, value) => value :: acc, (partial_1, partial_2) => partial_1 ::: partial_2 )
reducedRDD = reducedRDD.mapValues(_.sortBy(_._1))
```

In [6]:
```
reducedRDD.take(3) foreach println
```

```
(27.97.100.77,List((1437570537245,https://paytm.com:443/shop?utm_source=Affiliates&utm_medium=OMG&utm_campaign=OMG&utm_term=762154_)))
(115.114.78.170,List((1437570200618,https://paytm.com:443/shop/wallet/txnhistory?page_size=10&page_number=0&channel=web&version=2), (1437575648379,https://paytm.com:443/shop/v1/frequentorders?channel=web&version=2), (1437575714244,https://paytm.com:443/), (1437575792454,https://paytm.com:443/shop/cart?channel=web&version=2), (1437575893327,https://paytm.com:443/shop?utm_source=affiliate&utm_medium=promocodeclub&utm_campaign=promocodeclub-generic&utm_term=pccptm), (1437575972847,https://paytm.com:443/shop/cart?channel=web&version=2), (1437575987724,https://paytm.com:443/papi/v1/expresscart/verify), (1437576060188,https://paytm.com:443/papi/v1/expresscart/verify), (1437576122564,https://paytm.com:443/shop?utm_source=affiliate&utm_medium=promocodeclub&utm_campaign=promocodeclub-generic&utm_term=pccptm), (1437576181890,https://paytm.com:443/papi/v1/expresscart/verify), (1437576199007,https://paytm.com:443/shop/wallet/txnhistory?page_size=10&page_number=0&channel=web&version=2), (1437576202384,https://paytm.com:443/shop/wallet/balance?channel=web&version=2), (1437576473675,https://paytm.com:443/shop/v1/frequentorders?channel=web&version=2), (1437576583659,https://paytm.com:443/shop/cart?channel=web&version=2)))
(103.15.250.10,List((1437563413211,http://www.paytm.com:80/?utm_source=Affiliates&utm_medium=Paritycube&utm_campaign=Paritycube), (1437570478896,https://paytm.com:443/shop/cart?channel=web&version=2), (1437570748043,https://paytm.com:443/bus-tickets/search/Bangalore/Pondicherry(Puducherry)/2015-07-22/1), (1437570884086,https://paytm.com:443/favicon.ico), (1437575909963,https://paytm.com:443/shop/cart?channel=web&version=2), (1437576095151,http://www.paytm.com:80/), (1437576175501,https://paytm.com:443/shop/v1/frequentorders?channel=web&version=2), (1437576244596,https://paytm.com:443/shop/login?isIframe=true&theme=mp-web), (1437576265301,https://paytm.com:443/shop/cart?channel=web&version=2), (1437576393291,https://paytm.com:443/shop/authresponse?code=8e1daa2b-0c46-4c3a-99d7-887a1d2670ab&state=), (1437576420482,https://paytm.com:443/shop?utm_source=Affiliates&utm_medium=Payoom&utm_campaign=Payoom), (1437576569337,https://paytm.com:443/shop/logout?channel=web&version=2), (1437576638391,https://paytm.com:443/shop/v1/frequentorders?channel=web&version=2), (1437576778001,https://paytm.com:443/shop), (1437576813746,https://paytm.com:443/shop/cart?channel=web&version=2)))
```

Step 3 - Sessionize the Web Log by IP

In [1]:
```
/*K,V pair now looks like IP, List((session_last_access_time, set(URL), total_session_time))*/
import scala.collection.mutable.ListBuffer

def mapFunc(l: List[(Long, String)]) : ListBuffer[(Long, Set[String], Int)] = {
    var ret = ListBuffer[(Long, Set[String], Int)]()
    val size = l.length
    for ((ele, index) <- l.zipWithIndex) {
        if ( ret.isEmpty )
            ret += ((ele._1, Set(ele._2), 15))
        if ( index + 1 < size ) {
            var timeSpan = (l(index+1)._1 - ret.last._1)/1000/60
            if ( timeSpan < 15   ) {
                ret(ret.length - 1) = (
                    l(index+1)._1,
                    ret(ret.length - 1)._2 + l(index+1)._2,
                    ret(ret.length - 1)._3 + timeSpan.toInt )
            } else {
                ret += ((l(index+1)._1, Set(l(index+1)._2), 15))
            }
        }
    }
    ret
}

var sessionRDD = reducedRDD.mapValues(mapFunc)
sessionRDD.cache()
```

Out[1]: MapPartitionsRDD[12] at mapValues at <console>:34

In [6]:
```
sessionRDD.take(4) foreach (println)
```

(27.97.100.77,ListBuffer((1437570537245,Set(https://paytm.com:443/shop?utm_source=Affiliates&utm_medium=OMG&utm_campaign=OMG&utm_term=762154_),15)))
(103.15.250.10,ListBuffer((1437563413211,Set(http://www.paytm.com:80/?utm_source=Affiliates&utm_medium=Paritycube&utm_campaign=Paritycube),15), (1437570884086,Set(https://paytm.com:443/shop/cart?channel=web&version=2, https://paytm.com:443/bus-tickets/search/Bangalore/Pondicherry(Puducherry)/2015-07-22/1, (https://paytm.com:443/bus-tickets/search/Bangalore/Pondicherry(Puducherry)/2015-07-22/1,) https://paytm.com:443/favicon.ico),21), (https://paytm.com:443/favicon.ico),21),) (1437576813746,Set(https://paytm.com:443/shop/v1/frequentorders?channel=web&version=2, https://paytm.com:443/shop/authresponse?code=8e1daa2b-0c46-4c3a-99d7-887a1d2670ab&state=, (https://paytm.com:443/shop/authresponse?code=8e1daa2b-0c46-4c3a-99d7-887a1d2670ab&state=,) https://paytm.com:443/shop?utm_source=Affiliates&utm_medium=Payoom&utm_campaign=Payoom, (https://paytm.com:443/shop?utm_source=Affiliates&utm_medium=Payoom&utm_campaign=Payoom,) https://paytm.com:443/shop/login?isIframe=true&theme=mp-web, (https://paytm.com:443/shop/login?isIframe=true&theme=mp-web,) http://www.paytm.com:80/, (http://www.paytm.com:80/,) https://paytm.com:443/shop/logout?channel=web&version=2, (https://paytm.com:443/shop/logout?channel=web&version=2,) https://paytm.com:443/shop, (https://paytm.com:443/shop,) https://paytm.com:443/shop/cart?channel=web&version=2),27))) (https://paytm.com:443/shop/cart?channel=web&version=2),27))))
(115.114.78.170,ListBuffer((1437570200618,Set(https://paytm.com:443/shop/wallet/txnhistory?page_size=10&page_number=0&channel=web&version=2),15), (1437576583659,Set(https://paytm.com:443/shop/v1/frequentorders?channel=web&version=2, https://paytm.com:443/shop/wallet/balance?channel=web&version=2, (https://paytm.com:443/shop/wallet/balance?channel=web&version=2,) https://paytm.com:443/, (https://paytm.com:443/,) https://paytm.com:443/papi/v1/expresscart/verify, (https://paytm.com:443/papi/v1/expresscart/verify,) https://paytm.com:443/shop?utm_source=affiliate&utm_medium=promocodeclub&utm_campaign=promocodeclub-generic&utm_term=pccptm, (https://paytm.com:443/shop?utm_source=affiliate&utm_medium=promocodeclub&utm_campaign=promocodeclub-generic&utm_term=pccptm,) https://paytm.com:443/shop/wallet/txnhistory?page_size=10&page_number=0&channel=web&version=2, (https://paytm.com:443/shop/wallet/txnhistory?page_size=10&page_number=0&channel=web&version=2,) https://paytm.com:443/shop/cart?channel=web&version=2),26))) (https://paytm.com:443/shop/cart?channel=web&version=2),26))))
(115.118.136.29,ListBuffer((1437576422880,Set(https://paytm.com:443/papi/rr/products/4383445/statistics?channel=web&version=2, https://paytm.com:443/shop/v1/frequentorders?channel=web&version=2, (https://paytm.com:443/shop/v1/frequentorders?channel=web&version=2,) https://paytm.com:443/shop/wallet/balance?channel=web&version=2, (https://paytm.com:443/shop/wallet/balance?channel=web&version=2,) https://paytm.com:443/shop/orderhistory?pagesize=10&channel=web&version=2, (https://paytm.com:443/shop/orderhistory?pagesize=10&channel=web&version=2,) https://paytm.com:443/papi/v1/promosearch/product/4383445/offers?parent_id=13490884&price=1960&channel=web&version=2, (https://paytm.com:443/papi/v1/promosearch/product/4383445/offers?parent_id=13490884&price=1960&channel=web&version=2,) https://paytm.com:443/papi/nps/merchantrating?merchant_id=25370&channel=web&version=2, (https://paytm.com:443/papi/nps/merchantrating?merchant_id=25370&channel=web&version=2,) https://paytm.com:443/shop/authresponse?code=050610d2-6368-4d91-9ecb-a0f296fdb7ba&state=null, (https://paytm.com:443/shop/authresponse?code=050610d2-6368-4d91-9ecb-a0f296fdb7ba&state=null,) https://paytm.com:443/shop/cart?channel=web&version=2),26), (https://paytm.com:443/shop/cart?channel=web&version=2),26),) (1437577844279,Set(https://paytm.com:443/shop/logout?channel=w

```
eb&version=2, https://paytm.com:443/shop, (https://paytm.com:443/shop,) https://paytm.com:443/shop/car
t?channel=web&version=2),22))) (https://paytm.com:443/shop/cart?channel=web&version=2),22))))
```

Step 4 - Determine the Average Session Time

In [3]:
```
/* Get the average session time across all IPs. */

var c = sessionRDD.mapValues( x => x.foldLeft(0) { (acc, value) => acc + value._3 } )

val ans = c.aggregate((0,0))( (acc, value) => (acc._1 + value._2, acc._2 + 1),
                  (p1, p2) => (p1._1 + p2._1, p1._2 + p2._2))

// The final answer is 25 minutes
ans._1 / ans._2
```

Out[3]:
```
25
```

Step 5 - Determine Unique URL Visits per Session

In [4]:
```
/* , for example, the following output shows
IP 115.114.78.170 had 2 sessions, each of which visited 1 and 7 unique URL, respectively.*/

var d = sessionRDD.mapValues( x => x.map(_._2.size) )
d.take(5) foreach println
```

```
(27.97.100.77,ListBuffer(1))
(103.15.250.10,ListBuffer(1, 3, 8))
(115.114.78.170,ListBuffer(1, 7))
(115.118.136.29,ListBuffer(8, 3))
(103.26.57.44,ListBuffer(1))
```

Step 6 - Find the Most Engaged Users

In [7]:
```
/* IP 107.23.255.12 appeared to have the longest session time, 266 minutes, in total */

val maxAcrossKey = c.max()(new Ordering[Tuple2[String, Int]]() {
  override def compare(x: (String, Int), y: (String, Int)): Int =
      Ordering[Int].compare(x._2, y._2)
})

maxKey
```

Out[7]:
```
(107.23.255.12,266)
```