Stockholm University
DSV dept.
Academic year 2017/18

Big Data with NoSQL

Final report of the second assignment

Presented by:

- Giorgos Ntymenos
- Giacomo Bartoli

Step 1

Considering that HDFS stores file in block of 128MB, we have the following subdivision:

File1 = 350MB.
- Block A = 128/128MB
- Block B = 128/128MB
- Block C = 94/128MB

File2 = 620MB.
- Block D = 128/128MB
- Block E = 128/128MB
- Block F = 128/128MB
- Block G = 128/128MB
- Block H = 108/128MB

File 3 = 500MB.
- Block I = 128/128MB
- Block J = 128/128MB
- Block K = 128/128MB
- Block L = 116/128MB

	File name	**File size**	**Blocks**
File1	Cities.json	350 MB	A, B, C
File2	Citizens.json	620 MB	D, E, F, G, H
File3	Companies.json	500 MB	I, J,K,L

Step 2

Replication factor is 3.

DataCenter	**Rack 1**	**Rack 2**	**Rack 3**
DataNodes	1: A, F,L	6: A, E,J	11: B,F,K
	2: C, G, L	7: B, G, J	12: B, H,L
	3: C, I	8: B,G,K	13: C, H,
	4: D, I	9: A, D	14: E,I
	5: F,J	10: D, H	15: E, K

The main idea is to put one chunk on a rack and then the other two replicas on a different rack. This is due to fault tolerance and network balancing: if data on a certain node becomes unavailable they will be searched first on the same rack, which is faster and traffic on the network will be reduced, otherwise if the entire rack fails they will be still available on another rack. This is the reason why we should not have all replicas neither on the same node nor on the same rack. This latter scenario will take more time but data will still remain safe in case hardware crashes, which happens often.

In order to preserve equal disk utilization amongst racks and nodes, we want to split chunks as much as spread as possible. We can achieve it by placing new replicas on chunk servers with below-average disk space utilization.

According to what we claimed, we followed these rules:

- One replica on random node
- Second on a remote rack
- Third one on the same remote rack
- No more than one replica per node
- No more than two replicas per rack

Step 3

Increase the replication rate for citizens.json (r=5)

DataCenter	**Rack 1**	**Rack 2**	**Rack 3**
DataNodes	1: A, F, L, **H**	6: A, E,J,**F**	11: B,F,K
	2: C, G, L	7: B, G, J	12: B, H,L
	3: C, I,**D**	8: B,G,K	13: C, H.**D**
	4: D, I ,**E**	9: D, H,**E**	14: E,I,**G**
	5: F, J, **H**	10: D,J, **F**	15: E, K ,**G**

Consider that citizens.json contains blocks D,E,F,G,H, which are already replicated 3 times, we just need to add two more blocks for each one.

Aiming to achieve load balancing network and fault tolerance, the best split will result in 2 blocks on a rack, 2 other blocks on another rack and 1 final block on the third rack.

Step 4

Let's suppose that node 15 fails then blocks E,K,G are lost. We need now to move these blocks in other nodes. K's replication factor is 3, so we can put it in either rack 2 or 3, because then we will have two blocks in two racks and one block in the remain one. Concerning G and E, we have 5 blocks , as the replication factor for citizens.json is 5 ,so we will choose between the racks that have only one block of G (Racks 1 and 2). Same criteria is applied to E (Racks 1 and 3).

DataCenter	Rack 1	Rack 2	Rack 3
DataNodes	1: A, F, L, H	6: A, E,J,F	11: B,F,K
	2: C, G, L,**E**	7: B, G, J,**K**	12: B, H,L,**G**
	3: C, I,D	8: B,G,K	13: C, H.D
	4: D, I ,E	9: D, H,E	14: E,I,G
	5: F, J, H	10: D,J, F	15: **E, K ,G**

In this table, blue letters are the blocks we added after the node failed.