Dificultades

La primera dificultad fue tener que trabajar con fechas. Los datos por lo general no suelen tener un formato de fecha común, por lo que antes de trabajar con las fechas, es necesario limpiar el campo y convertirlas en un formato que acomode.
En nuestro caso, para mongodb, se utilizo el campo *created_at* para realizar las consultas. El problema fue que el formato que viene por defecto no permite realizar la operación *aggregate* que posee mongosdb, por lo que se tuvo que cambiar el formato de la fecha.
Para esto se utilizo el siguiente *script*:

```
db.icccualquiercosa.find({}).forEach(function (doc) {
doc.created_at = new Date(doc.created_at);
db.icccualquiercosa.save(doc);
});
```

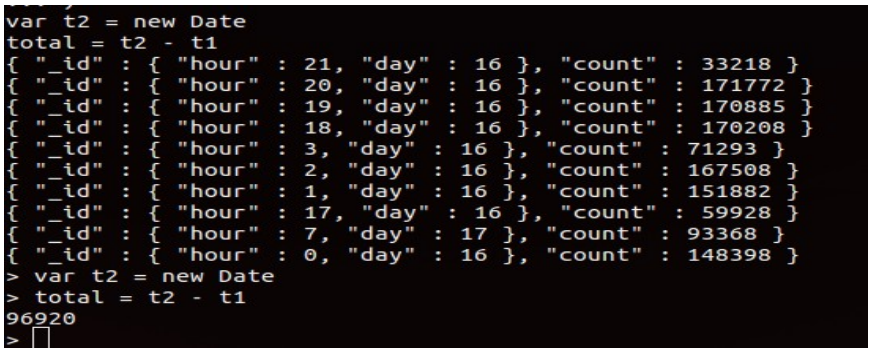
Por desgracia, este error ocurrió cunado ya tenia todo el set de datos de *tweets* descargados, por lo que dicha conversión se debió hacer con todos los datos, tomando un tiempo mayor a los 40 minutos.

Otra dificultad fue al momento de migrar los datos a una base de datos relacional. Debido a que una de las características principales de mongodb es poder almacenar los datos como documentos independiente del formato que traigan, es que se tuvieron problemas con los campos a definir en la base de satos relacional SQL. No todos los campos están presentes en todos los *tweets*, por lo que se tubo que realizar un prefiltrado para saber que campos eran mas comunes y cuales eran los datos que realmente necesitábamos para neutras métricas.
No obstante, hubieron algunos *tweets* que no se lograron replicar en la base de datos SQL debido a la falta de campos que traían, o a la calidad de los datos que estos campos contenían, es por esto que la cantidad de *tweets* almacenados en mongodb no es la misma que la cantidad de *tweets* almacenados en SQL.

Mongodb

1. La primera consulta que se realizo, tomo un tiempo de 97 segundos. En esta consulta se realizo un conteo de cuantos *tweets* habían llegado en una hora. Para esto, se utilizo el campo *create_at* que crea automáticamente mongo, asumiendo que la hora en que llego el *tweet* fue la hora en que se almaceno.

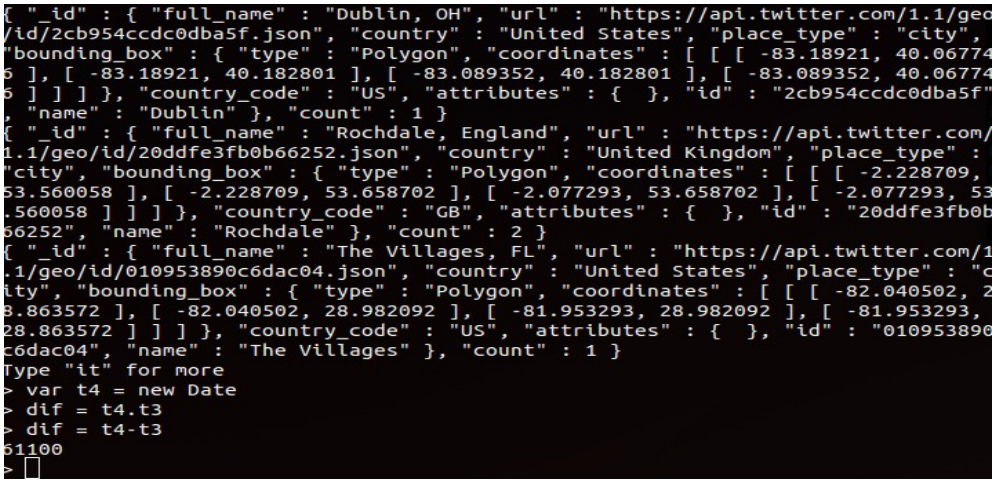
   Resultados:



   Notamos que si bien, la consulta puede parecer lenta, estamos trabajando con un gran numero de tweets, y para realizar esta consulta, se tienen que ir revisando cada uno de los *tweets* para acceder a la fecha en la que fue creado.

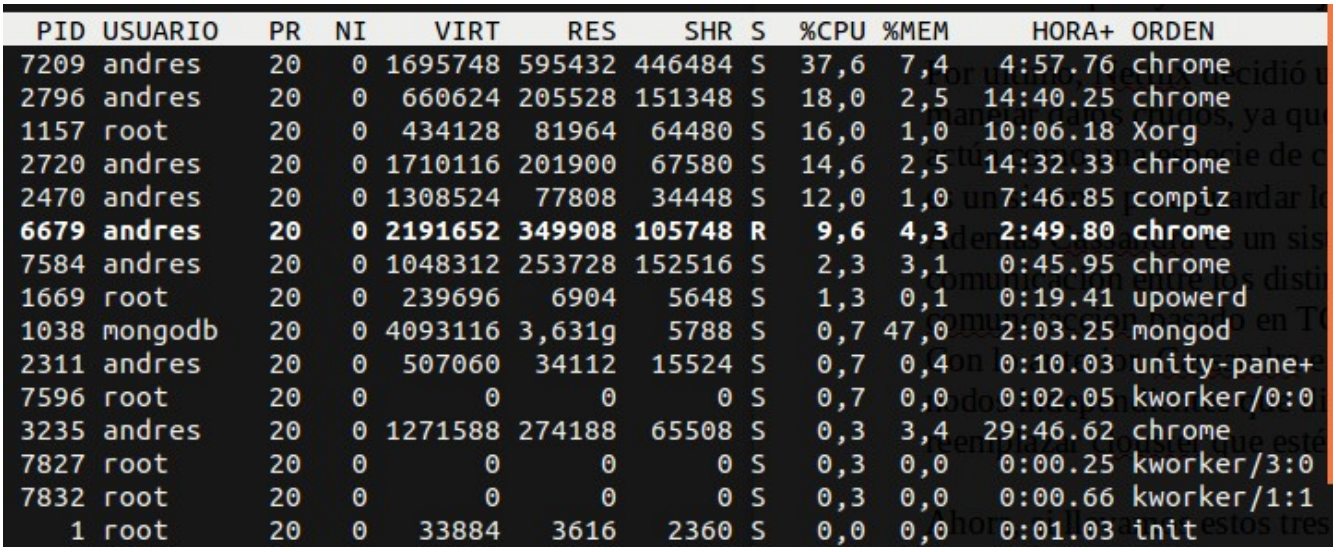
2. La segunda consulta que se realizo fue para poder verificar la ubicaron geográfica en la cual se había realizado el *tweet*. Para esto se utilizo el campo place que viene en el objeto *tweet*. Esta consulta tomo 61 segundos.

   Resultados



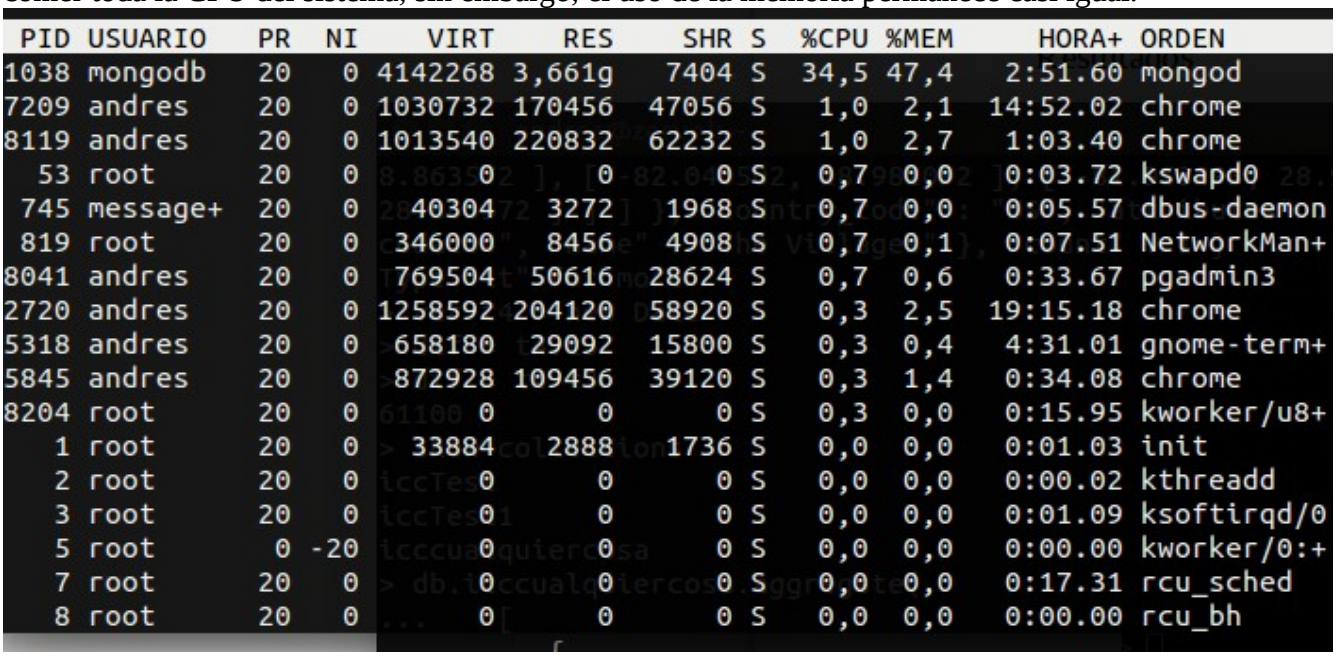
Notamos que ambas consultas pueden llegar a tomar bastante tiempo, aún trabajando con un set de datos pequeños, por lo que es impensable que una empresa como Twitter utilice métodos como estos para realizar sus estadísticas.

En cuanto al rendimiento del equipo, notamos que inicialmente, si bien tenemos ciertos procesos que están utilizando bastante de nuestra CPU, no hay ninguno que este acupando la memoria a un nivel muy alto, salvo mongodb.



PID	USUARIO	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	HORA+	ORDEN
7209	andres	20	0	1695748	595432	446484	S	37,6	7,4	4:57.76	chrome
2796	andres	20	0	660624	205528	151348	S	18,0	2,5	14:40.25	chrome
1157	root	20	0	434128	81964	64480	S	16,0	1,0	10:06.18	Xorg
2720	andres	20	0	1710116	201900	67580	S	14,6	2,5	14:32.33	chrome
2470	andres	20	0	1308524	77808	34448	S	12,0	1,0	7:46.85	compiz
6679	**andres**	**20**	**0**	**2191652**	**349908**	**105748**	**R**	**9,6**	**4,3**	**2:49.80**	**chrome**
7584	andres	20	0	1048312	253728	152516	S	2,3	3,1	0:45.95	chrome
1669	root	20	0	239696	6904	5648	S	1,3	0,1	0:19.41	upowerd
1038	mongodb	20	0	4093116	3,631g	5788	S	0,7	47,0	2:03.25	mongod
2311	andres	20	0	507060	34112	15524	S	0,7	0,4	0:10.03	unity-pane+
7596	root	20	0	0	0	0	S	0,7	0,0	0:02.05	kworker/0:0
3235	andres	20	0	1271588	274188	65508	S	0,3	3,4	29:46.62	chrome
7827	root	20	0	0	0	0	S	0,3	0,0	0:00.25	kworker/3:0
7832	root	20	0	0	0	0	S	0,3	0,0	0:00.66	kworker/1:1
1	root	20	0	33884	3616	2360	S	0,0	0,0	0:01.03	init

Al momento de realizar las consultas, notamos como el proceso relacionado con mongodb se empieza a comer toda la CPU del sistema, sin embargo, el uso de la memoria permanece casi igual.



PID	USUARIO	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	HORA+	ORDEN
1038	mongodb	20	0	4142268	3,661g	7404	S	34,5	47,4	2:51.60	mongod
7209	andres	20	0	1030732	170456	47056	S	1,0	2,1	14:52.02	chrome
8119	andres	20	0	1013540	220832	62232	S	1,0	2,7	1:03.40	chrome
53	root	20	0	0	0	0	S	0,7	0,0	0:03.72	kswapd0
745	message+	20	0	40304	3272	1968	S	0,7	0,0	0:05.57	dbus-daemon
819	root	20	0	346000	8456	4908	S	0,7	0,1	0:07.51	NetworkMan+
8041	andres	20	0	769504	50616	28624	S	0,7	0,6	0:33.67	pgadmin3
2720	andres	20	0	1258592	204120	58920	S	0,3	2,5	19:15.18	chrome
5318	andres	20	0	658180	29092	15800	S	0,3	0,4	4:31.01	gnome-term+
5845	andres	20	0	872928	109456	39120	S	0,3	1,4	0:34.08	chrome
8204	root	20	0	0	0	0	S	0,3	0,0	0:15.95	kworker/u8+
1	root	20	0	33884	2888	1736	S	0,0	0,0	0:01.03	init
2	root	20	0	0	0	0	S	0,0	0,0	0:00.02	kthreadd
3	root	20	0	0	0	0	S	0,0	0,0	0:01.09	ksoftirqd/0
5	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	kworker/0:+
7	root	20	0	0	0	0	S	0,0	0,0	0:17.31	rcu_sched
8	root	20	0	0	0	0	S	0,0	0,0	0:00.00	rcu_bh