DIEGO HUELTES

ANÁLISIS DE DATOS LOW COST CON BCOLZ / BQUERY

SOBRE MI

- Ingeniero en Informática por la Universidad de Granada
- Ingeniero Python en RavenPack (Marbella)
- Especializado en análisis de datos en la Università degli Studi di Catania (Italia)
- Profesor en el curso ejecutivo de Big Data Analytics de la Escuela de Organización Industrial (EOI)
- Data science lover
- Twitter @JdiegoH
- GitHub: DiegoHueltes

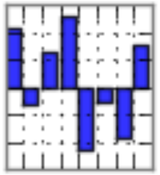
ANÁLISIS DE DATOS NUMÉRICO

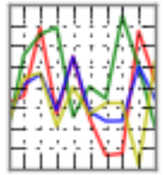
sex	age	educyr	farm	urban	hhsize	lntotal	lnmed	lnrlfood
female	68	4	no	yes	6	10.136490	11.233210	8.639339
female	57	8	no	yes	6	10.252060	8.505120	9.345752
male	42	14	no	yes	6	10.932310	8.713418	10.226330
female	72	9	no	yes	6	10.267490	9.291736	9.263722
female	73	1	no	yes	8	10.488110	7.555382	9.592890
female	66	13	no	yes	7	10.526600	9.789702	9.372034
female	73	2	no	yes	9	10.229960	8.192847	9.276959
male	46	9	no	yes	4	9.526502	5.783825	8.851970
male	50	12	no	yes	5	10.490310	7.506592	9.719610
male	45	12	no	yes	4	10.020070	5.783825	8.938481
male	75	4	no	yes	6	10.493170	8.117908	9.260901
male	44	10	no	yes	3	8.461615	6.829794	6.733994
male	51	12	no	yes	4	9.578440	7.265429	9.035526
female	48	17	no	yes	4	10.705170	7.575585	9.735068
male	54	17	no	yes	4	10.188040	7.455298	9.643557
male	43	16	no	yes	3	10.739150	4.867535	9.870209
female	41	13	no	yes	4	9.670019	6.516193	9.076807
female	70	2	no	yes	2	10.051010	6.388561	9.069571
male	67	9	no	yes	2	9.150693	6.371612	8.447347
male	62	9	no	yes	7	10.019670	6.326149	9.164073

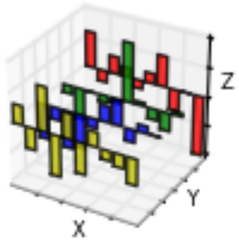
HERRAMIENTAS

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$







julia

¿POR QUÉ EN COLUMNAS?

A	B	C
A1	B1	C1
A2	B2	C2
A3	B3	C3

EJEMPLO

sex	age	educyr	farm	urban	hhsize	lntotal	lnmed	lnrlfood
female	68	4	no	yes	6	10.136490	11.233210	8.639339
female	57	8	no	yes	6	10.252060	8.505120	9.345752
male	42	14	no	yes	6	10.932310	8.713418	10.226330
female	72	9	no	yes	6	10.267490	9.291736	9.263722
female	73	1	no	yes	8	10.488110	7.555382	9.592890
female	66	13	no	yes	7	10.526600	9.789702	9.372034
female	73	2	no	yes	9	10.229960	8.192847	9.276959
male	46	9	no	yes	4	9.526502	5.783825	8.851970
male	50	12	no	yes	5	10.490310	7.506592	9.719610
male	45	12	no	yes	4	10.020070	5.783825	8.938481
male	75	4	no	yes	6	10.493170	8.117908	9.260901
male	44	10	no	yes	3	8.461615	6.829794	6.733994
male	51	12	no	yes	4	9.578440	7.265429	9.035526
female	48	17	no	yes	4	10.705170	7.575585	9.735068
male	54	17	no	yes	4	10.188040	7.455298	9.643557
male	43	16	no	yes	3	10.739150	4.867535	9.870209
female	41	13	no	yes	4	9.670019	6.516193	9.076807
female	70	2	no	yes	2	10.051010	6.388561	9.069571
male	67	9	no	yes	2	9.150693	6.371612	8.447347
male	62	9	no	yes	7	10.019670	6.326149	9.164073

GROUP BY

```
df.groupby(['sex', 'urban'])['lntotal'].sum()
```

sex	urban	lntotal
female	no	8229.449212
female	yes	6791.747925
male	no	30810.547545
male	yes	10208.281080

```
df.groupby(['age'])['lnmed', 'lnrlfood'].mean()
```

age	lnmed	lnrlfood
16	0.000000	7.016957
18	8.351847	8.837768
19	6.278522	7.887609
20	4.290405	8.310306
21	4.566683	8.364449
22	5.090194	8.534154
23	4.291696	8.308049
24	5.162307	8.472016
25	4.980300	8.382062
26	4.510228	8.341341
27	5.401488	8.528195
28	5.394319	8.504374
29	4.740650	8.540293
30	5.272670	8.528900

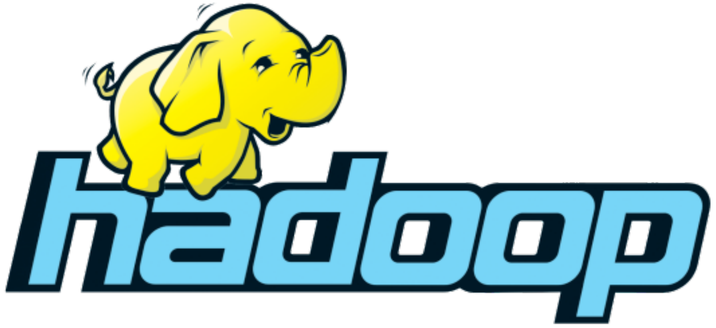
FILTRO

```
df[df['sex'] == 'female' ]
```

sex	age	educyr	farm	urban	hhsize	lntotal	lnmed	lnrlfood
female	68	4	no	yes	6	10.136490	11.233210	8.639339
female	57	8	no	yes	6	10.252060	8.505120	9.345752
female	72	9	no	yes	6	10.267490	9.291736	9.263722
female	73	1	no	yes	8	10.488110	7.555382	9.592890
female	66	13	no	yes	7	10.526600	9.789702	9.372034
female	73	2	no	yes	9	10.229960	8.192847	9.276959
female	48	17	no	yes	4	10.705170	7.575585	9.735068
female	41	13	no	yes	4	9.670019	6.516193	9.076807
female	70	2	no	yes	2	10.051010	6.388561	9.069571
female	81	1	no	yes	2	9.612206	6.856462	9.028255
female	70	0	no	yes	5	10.051030	7.827640	9.147572
female	51	9	no	yes	5	11.019560	7.393263	10.235500
female	62	3	no	yes	5	10.000500	5.273000	9.195153
female	58	5	no	yes	4	10.178800	9.010669	9.529779
female	53	9	no	yes	6	9.876953	6.551080	9.315619
female	65	0	no	yes	1	8.760125	7.064759	8.274182
female	32	15	no	yes	4	10.373930	0.000000	9.699347
female	70	5	no	yes	3	9.491340	6.476973	8.977983
female	45	12	no	yes	4	9.562902	6.476973	8.603507
female	45	12	no	yes	5	11.134990	9.122602	9.949111

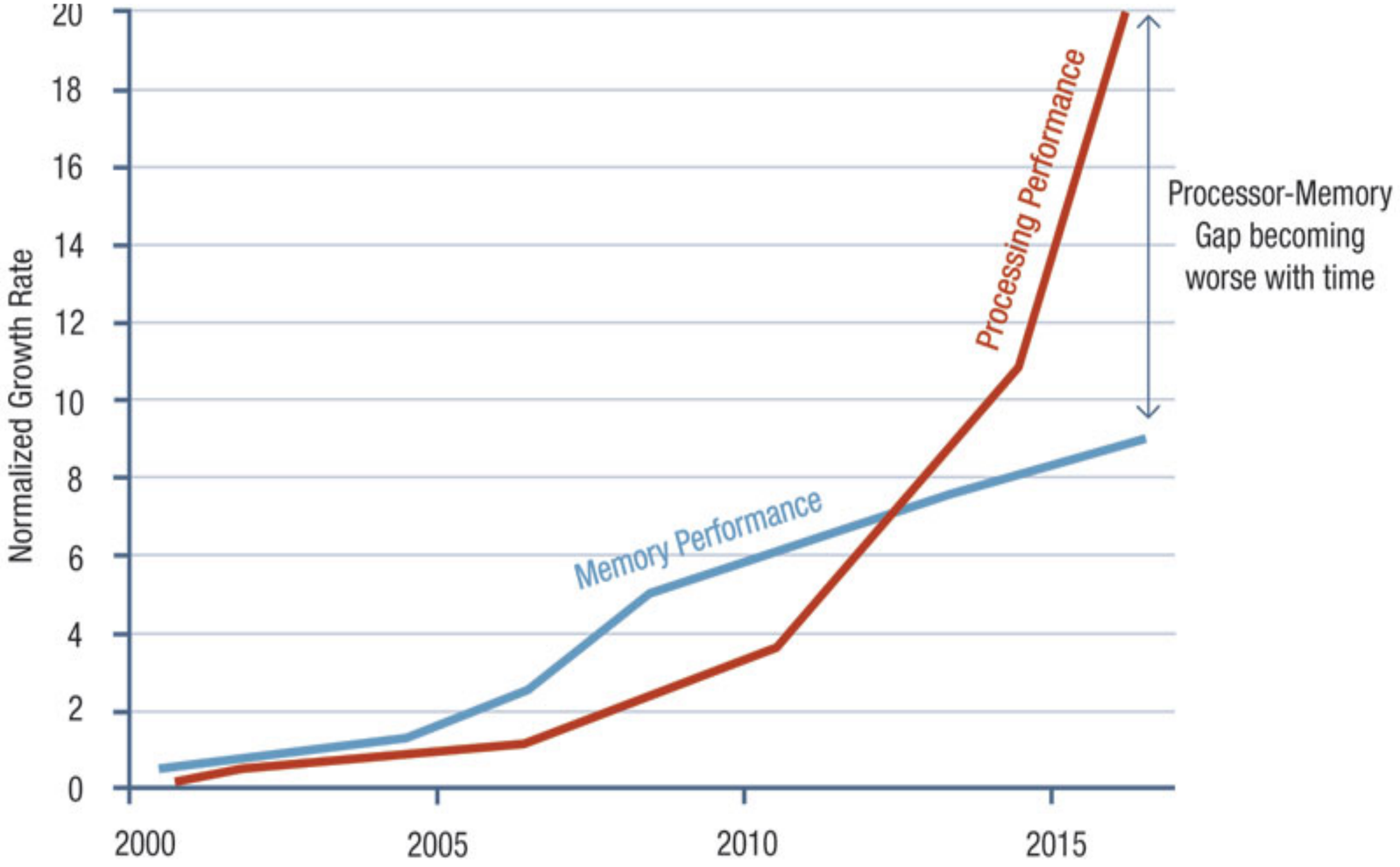
¡Bien! ¡Todo el mundo puede hacer análisis numérico con pandas!

¿Y si queremos hacer un análisis de datos sobre 100,000,000 registros?



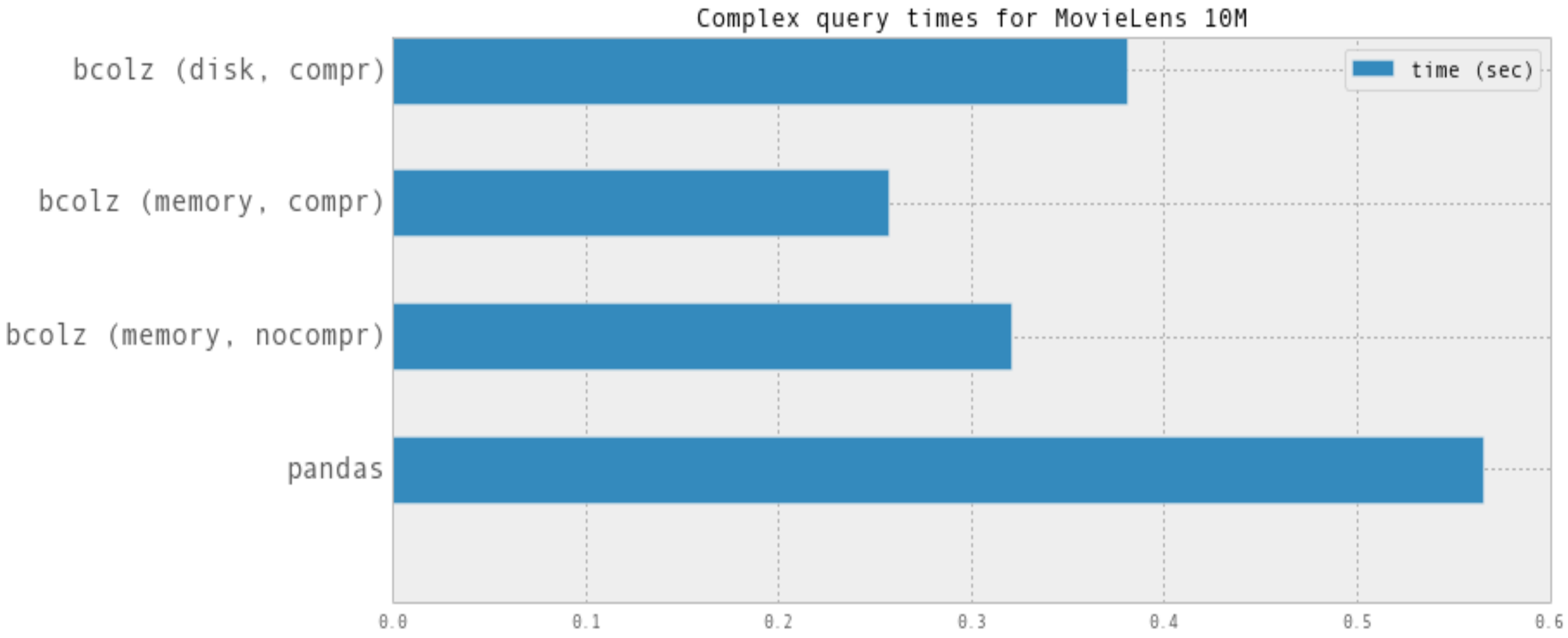
DASK

bcolz provides **columnar**, **chunked** data containers that can be **compressed** either in-memory and **on-disk**



*Graph derived from approximating aggregate processing, memory performance data from ASIC data. Includes future projections.

ANÁLISIS DE DATOS LOW COST CON BCOLZ / BQUERY



BCOLZ EXAMPLES

```
ct = bcolz.ctable(rootdir='mydir')

ct["(f0>0) & (f1<10)"]

ct.addcol(new_col)

ct = bcolz.fromiter(((i,i*i) for i in xrange(N)), dtype="i4,f8", count=N)

[row for row in ct.where("(f0>0) & (f1<10)")]

ct.eval("cos((3+f0)/sqrt(2*f1))")

bcolz.fromiter([row for row in ct.where("(f0>0) & (f1<10)")], rootdir='mydir')
```

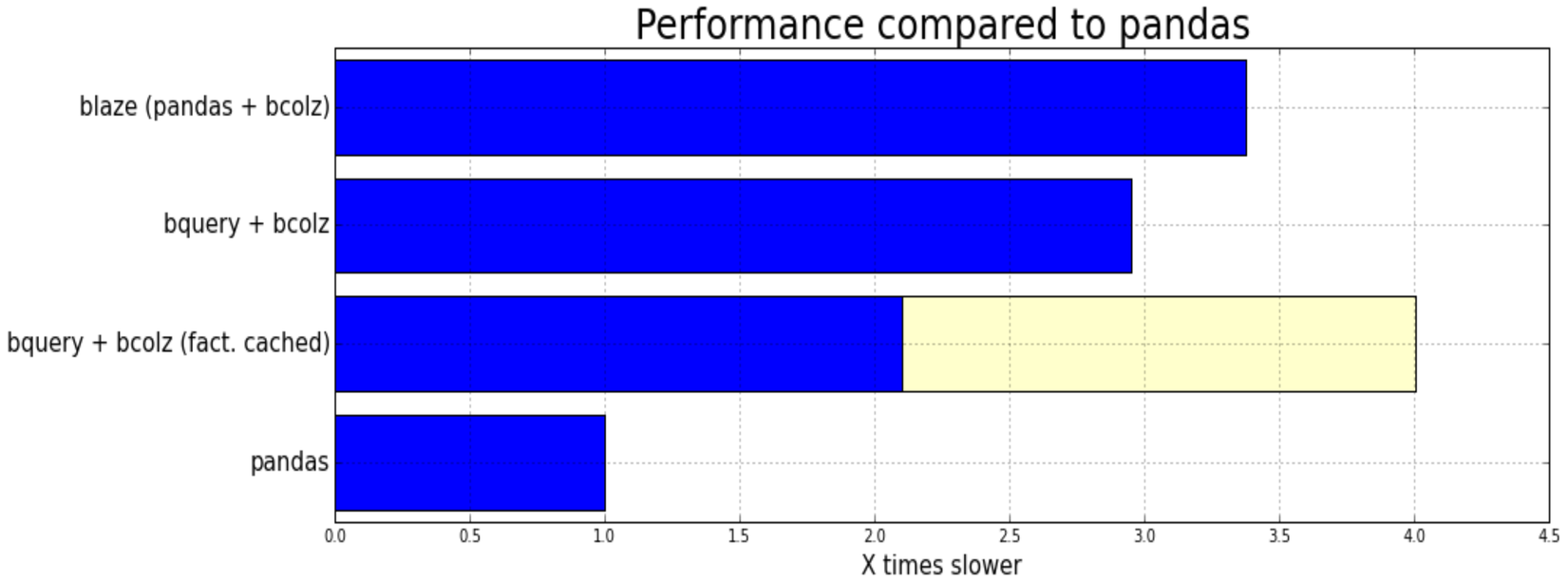
```
ct.todataframe()

fromdataframe(df)
```

groupby??

BQUERY

query and
aggregation
framework for
bcolz

```
import bquery
# assuming you have an example bcolz file called example.bcolz
ct = bquery.ctable(rootdir='example.bcolz')
ct.groupby(list_of_groupby_columns, agg_list)
ct.groupby(['f0'], [['f2', 'sum', 'f2_sum'], ['f2', 'mean', 'f2_mean']])
```



TAXI SET PERFORMANCE

- https://github.com/visualfabriq/bquery/blob/master/bquery/benchmarks/taxi/Taxi%20Set.ipynb
- 146,112,989 rows

Multi Process, All Measures

In [18]:
```
with ctime(message='CT payment_type all measure sum, ' + str(cpu_count()) + ' processors'):
    execute_query(ct_list, ['payment_type'], measure_list)

with ctime(message='CT yearmonth  all measure sum, ' + str(cpu_count()) + ' processors'):
    execute_query(ct_list, ['pickup_yearmonth'], measure_list)

with ctime(message='CT yearmonth + payment_type  all measure sum, ' + str(cpu_count()) + ' process
ors'):
    execute_query(ct_list, ['pickup_yearmonth', 'payment_type'], measure_list)
```

```
CT payment_type all measure sum, 8 processors:
(6.3044, 'sec')


CT yearmonth  all measure sum, 8 processors:
(5.3946, 'sec')


CT yearmonth + payment_type  all measure sum, 8 processors:
(7.6656, 'sec')
```

GENIAL PARA

- Análisis de datos numérico en una máquina cuando los datos no entran en memoria. Pero el volumen de datos es manejable a tiempo “real”.
- Análisis de datos numérico mas rápido que pandas (dependiendo que quieras hacer)

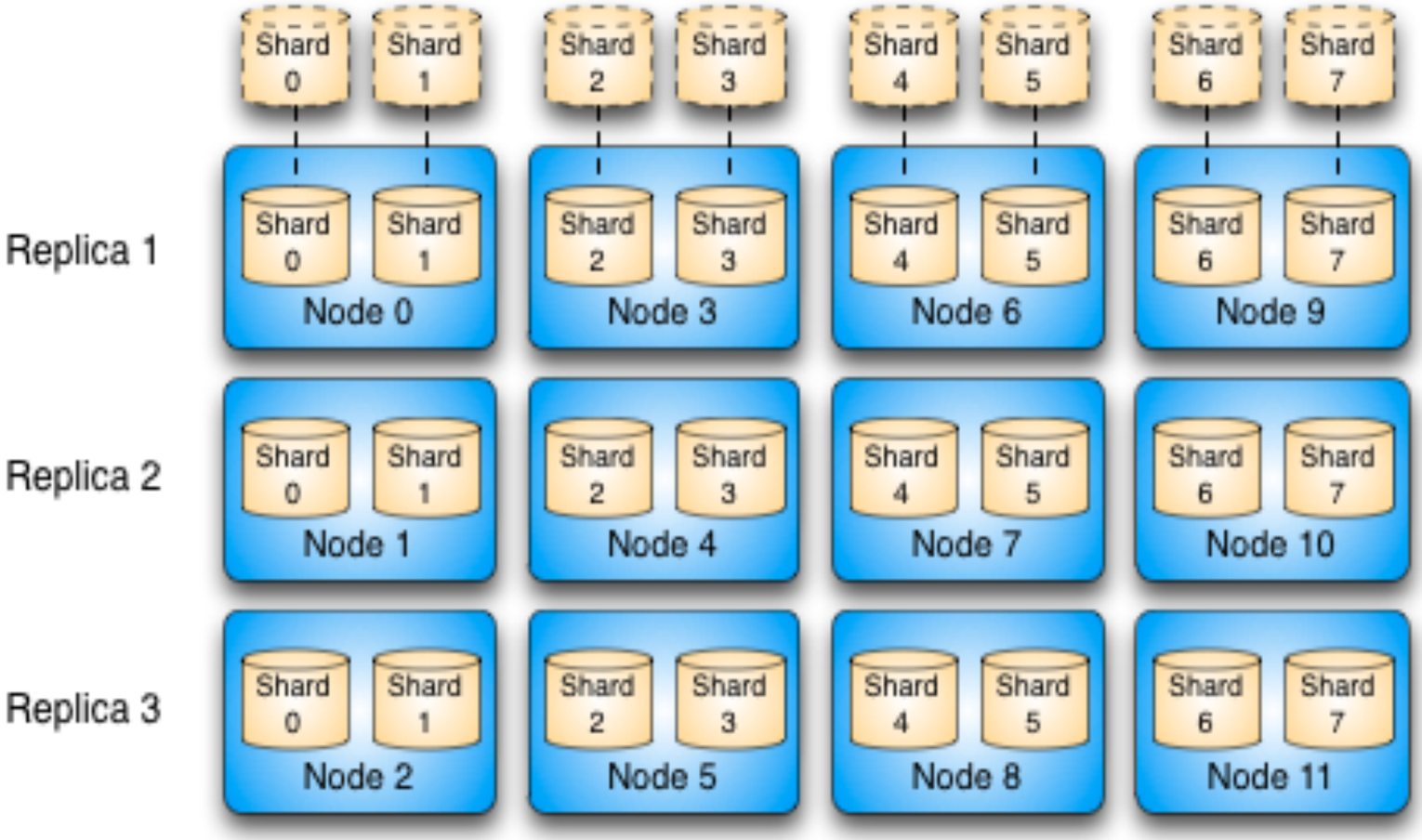
NO ES PARA

- Análisis de datos no numéricos (usa una bd)
- Datos no orientados en columnas

POR RESOLVER

- Volúmenes de datos extremadamente grandes

COMING SOON...


Replica 1
Replica 2
Replica 3
Shard 0
Shard 1
Shard 2
Shard 3
Shard 4
Shard 5
Shard 6
Shard 7
Node 0
Node 3
Node 6
Node 9
Node 1
Node 4
Node 7
Node 10
Node 2
Node 5
Node 8
Node 11

COLABORAD!

bquery:
https://github.com/visualfabriq/bquery

bcolz:
https://github.com/Blosc/bcolz

¿PREGUNTAS?

¡GRACIAS!