

Todo el big data es igual

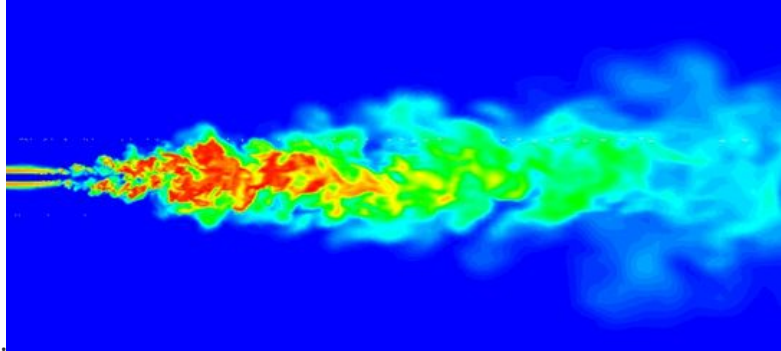
UOC Data Day. Madrid
23 de octubre de 2018



Sobre mí

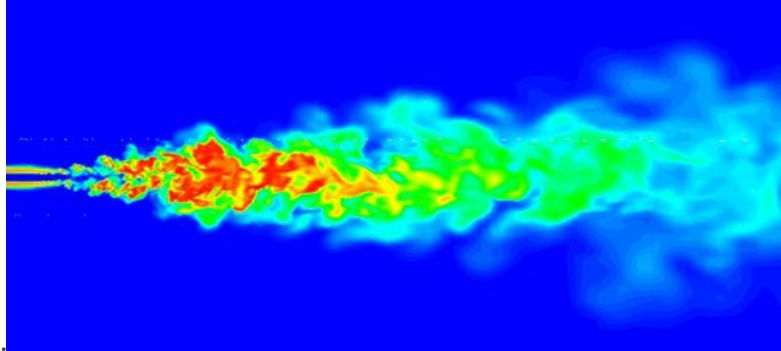
- Doctor en Ingeniería Aeroespacial, Ingeniero Aeronáutico
 - Hice la tesis en Teoría y Simulación Numérica de flujos turbulentos.
 - @guillemborrell
 - guillemborrell.es
- Ahora Team Lead en Kernel Analytics
 - Proyectos taylor-made de analítica de datos.
 - Partner estratégico en muchos de nuestros clientes.
 - Data Science + Data Engineering + Web Development
- En Big Data y Analytics desde cuando sólo se llamaba Cálculo Numérico.
 - R se usaba sólo en las facultades de Matemáticas, Python era un lenguaje de juguete y Scala no “escalaba”
 - Las cosas en serio se hacían con C++ o Fortran
 - El cálculo distribuido tenía poca implantación industrial, nula implantación en marketing y ventas.
 - La supercomputación era una un concepto maduro tecnológicamente
 - Sistemas de ficheros distribuidos, redes de altas prestaciones, clusters de cálculo con Linux...

Sobre mí



- Doctor en Ingeniería Aeroespacial, Ingeniero Aeronáutico
 - Hice la tesis en Teoría y Simulación Numérica de flujos turbulentos.
 - @guillemborrell
 - guillemborrell.es
- Ahora Team Lead en Kernel Analytics
 - Proyectos tailor-made de analítica de datos.
 - Partner estratégico en muchos de nuestros clientes.
 - Data Science + Data Engineering + Web Development
- En Big Data y Analytics desde cuando sólo se llamaba Cálculo Numérico.
 - R se usaba sólo en las facultades de Matemáticas, Python era un lenguaje de juguete y Scala no “escalaba”
 - Las cosas en serio se hacían con C++ o Fortran
 - El cálculo distribuido tenía poca implantación industrial, nula implantación en marketing y ventas.
 - La supercomputación era una un concepto maduro tecnológicamente
 - Sistemas de ficheros distribuidos, redes de altas prestaciones, clusters de cálculo con Linux...

Sobre mí



- Doctor en Ingeniería Aeroespacial, Ingeniero Aeronáutico
 - Hice la tesis en Teoría y Simulación Numérica de flujos turbulentos.
 - @guillemborrell
 - guillemborrell.es
- Ahora Team Lead en Kernel Analytics
 - Proyectos taylor-made de analítica de datos.
 - Partner estratégico en muchos de nuestros clientes.
 - Data Science + Data Engineering + Web Development
- En Big
 -
 -
 -
 -
 -
 -

MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.

Finalmente se pudo industrializar la computación distribuida

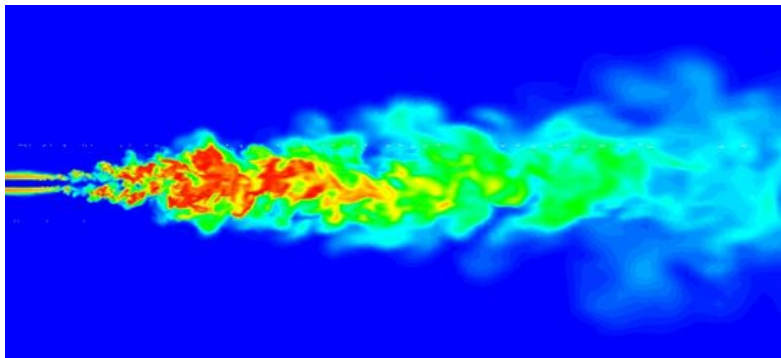
Simplified Data Processing on Large Clusters

- Simplified → no queremos un sistema de cálculo general, sino una serie de algoritmos específicos
- Data Processing → algoritmos para la carga, proceso y transformación de datos
- On Large Clusters → de manera distribuida y tolerante a fallos

Finalmente se pudo industrializar la computación distribuida

Simplified Data Processing on Large Clusters

- Simplified → no queremos un sistema de cálculo general, sino una serie de algoritmos específicos
- Data Processing → algoritmos para la carga, proceso y transformación de datos
- On Large Clusters → de manera distribuida y tolerante a fallos



≠

HR Information		Contact		
Position	Salary	Office	Extn.	
Accountant	\$162,700	Tokyo	5407	
Chief Executive Officer (CEO)	\$1,200,000	London	5797	
Junior Technical Author	\$86,000	San Francisco	1562	
Software Engineer	\$132,000	London	2558	
Software Engineer	\$206,850	San Francisco	1314	
Integration Specialist	\$372,000	New York	4804	
Software Engineer	\$163,500	London	6222	
Pre-Sales Support	\$106,450	New York	8330	
Sales Assistant	\$145,600	New York	3990	
Senior Javascript Developer	\$433,060	Edinburgh	6224	

Lo importante está ya en el título

Simplified Data Processing on Large Clusters

- Simplified → no queremos un sistema de cálculo general, sino una serie de algoritmos específicos
- Data Processing → algoritmos para la carga, proceso y transformación de datos
- On Large Clusters → de manera distribuida y tolerante a fallos





Blue Gene/P ANL

≠



Google NC

La revolución ha venido al diseñar toda una plataforma para resolver un problema específico, más sencillo que el general.

M  **: Simplified**  **ssing on Large Clusters**

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.



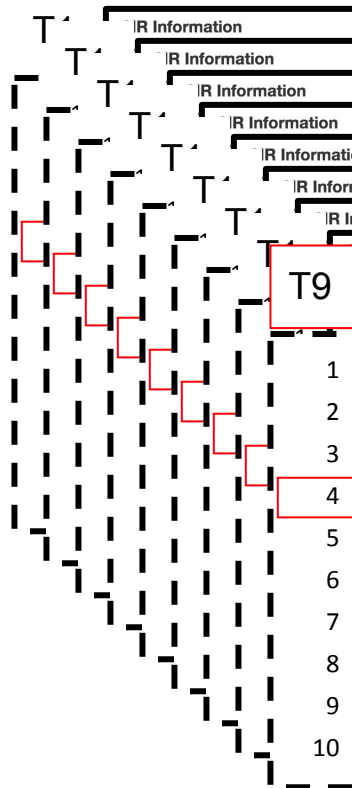


HR Information		Contact	
Position	Salary	Office	Extn.
Accountant	\$162,700	Tokyo	5407
Chief Executive Officer (CEO)	\$1,200,000	London	5797
Junior Technical Author	\$86,000	San Francisco	1562
Software Engineer	\$132,000	London	2558
Software Engineer	\$206,850	San Francisco	1314
Integration Specialist	\$372,000	New York	4804
Software Engineer	\$163,500	London	6222
Pre-Sales Support	\$106,450	New York	8330
Sales Assistant	\$145,600	New York	3990
Senior Javascript Developer	\$433,060	Edinburgh	6224

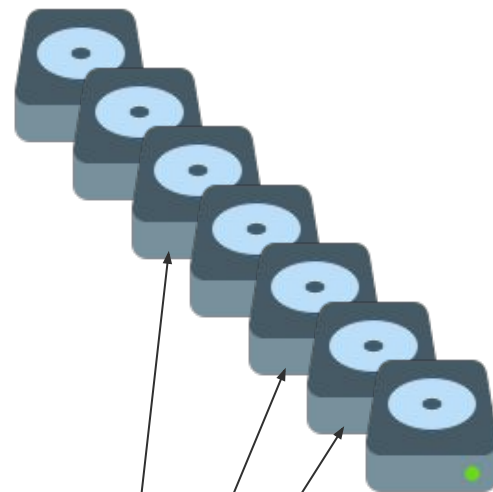
T1

1
2
3
4
5
6
7
8
9
10

HR Information		Contact	
Position	Salary	Office	Extn.
1 Accountant	\$162,700	Tokyo	5407
2 Chief Executive Officer (CEO)	\$1,200,000	London	5797
3 Junior Technical Author	\$86,000	San Francisco	1562
4 Software Engineer	\$132,000	London	2558
5 Software Engineer	\$206,850	San Francisco	1314
6 Integration Specialist	\$372,000	New York	4804
7 Software Engineer	\$163,500	London	6222
8 Pre-Sales Support	\$106,450	New York	8330
9 Sales Assistant	\$145,600	New York	3990
10 Senior Javascript Developer	\$433,060	Edinburgh	6224



	HR Information	Contact
1	Accountant	\$162,700 Tokyo 5407
2	Chief Executive Officer (CEO)	\$1,200,000 London 5797
3	Junior Technical Author	\$86,000 San Francisco 1562
4	Software Engineer	\$132,000 London 2558
5	Software Engineer	\$206,850 San Francisco 1314
6	Integration Specialist	\$372,000 New York 4804
7	Software Engineer	\$163,500 London 6222
8	Pre-Sales Support	\$106,450 New York 8330
9	Sales Assistant	\$145,600 New York 3990
10	Senior Javascript Developer	\$433,060 Edinburgh 6224



Guardo el esquema a parte

T9, 4: {"Software Engineer", "\$132,000", "London", "Extn. 2558"}

El par clave-valor

T9, 4: {"Software Engineer", "\$132,000", "London", "Extn. 2558"}

1. Esquema. Qué puede contener
 2. **Clave**. Qué es. Dónde está ubicado
 3. **Valor**. Contenido
- La clave es única **para todo el almacenamiento**
 - El esquema es mucho más pequeño que los datos y lo puedo dejar donde quiera
 - **Es equivalente a llevarnos el registro de la tabla**
 - Permite trivialmente su almacenamiento **distribuido**

Las funciones puras

Una función pura sólo depende de las variables de entrada que no puede cambiar y no depende de un estado externo.

```
def min(x, y):  
    if x < y:  
        return x  
    else:  
        return y
```

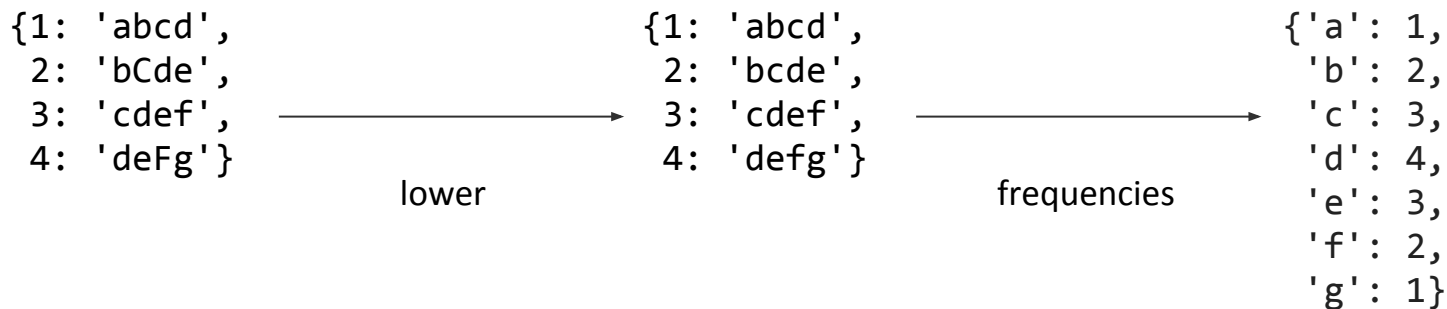
```
exponent = 2  
  
def powers(L):  
    for i in range(len(L)):  
        L[i] = L[i]**exponent  
    return L
```

La composición de funciones

Podemos conseguir que dos funciones $f(x)$ y $g(x)$ se puedan componer $f(g(x))$ o $g(f(x))$

1. Si retornan el mismo tipo que el argumento de entrada. **Como un par clave - valor.**
2. Si son funciones puras

Puedo implementar prácticamente cualquier algoritmo como contar la frecuencia de ocurrencia de letras en una lista de frases

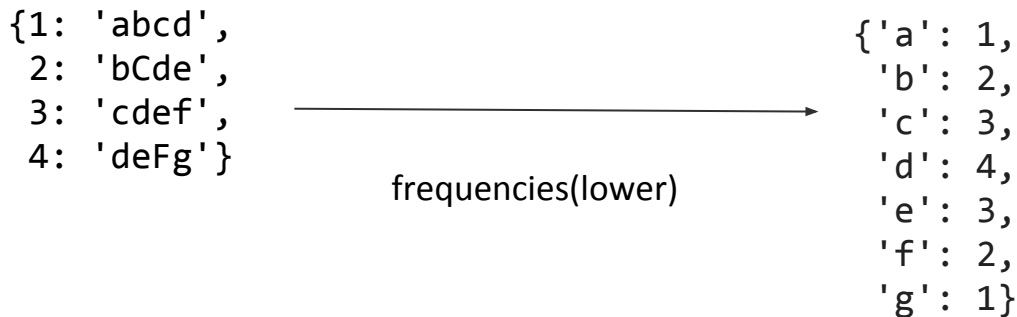


La composición de funciones

Podemos conseguir que dos funciones $f(x)$ y $g(x)$ se puedan componer $f(g(x))$ o $g(f(x))$

1. Si retornan el mismo tipo que el argumento de entrada. **Como un par clave - valor.**
2. Si son funciones puras

Puedo implementar prácticamente cualquier algoritmo como contar la frecuencia de ocurrencia de letras en una lista de frases

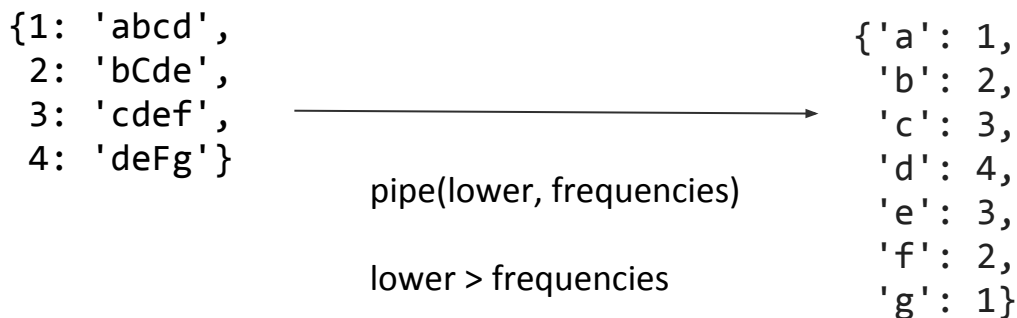


La composición de funciones

Podemos conseguir que dos funciones $f(x)$ y $g(x)$ se puedan componer $f(g(x))$ o $g(f(x))$

1. Si retornan el mismo tipo que el argumento de entrada. **Como un par clave - valor.**
2. Si son funciones puras

Puedo implementar prácticamente cualquier algoritmo como contar la frecuencia de ocurrencia de letras en una lista de frases



Lazy evaluation

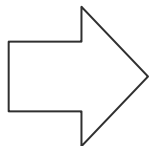
Consumir todo lo que sea iterable (listas, tuplas, diccionarios, objetos, tablas...) en el momento en el que sea necesario.

```
>>> a = [1,2,3,4]
>>> b = ['a', 'b', 'c', 'd']
>>> zip(a, b)
<zip at 0x7fa166876708>
```

```
>>> for ai, bi in zip(a, b):
...     print(ai, bi)
1 a
2 b
3 c
4 d
```


5 principios o garantías del Cheap Data

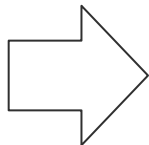
Tablas e índices
Clave - valor
Funciones puras
Composición de funciones
Lazy evaluation



Damos estructura a cualquier registro (Qué es...)
Doy una identidad para llegar al dato en cualquier parte del sistema (Dónde está...)
Puedo evaluar en “unidades” de ejecución sin estado (Me da igual dónde evaluar...)
Cualquier salida de una evaluación puede ser la entrada de la siguiente (Homogeneidad)
La infraestructura puede decidir cuándo y dónde evaluar (Declarativo)

5 principios o garantías del Cheap Data

Tablas e índices
Clave - valor
Funciones puras
Composición de funciones
Lazy evaluation



Damos estructura a cualquier registro (Qué es...)
Doy una identidad para llegar al dato en cualquier parte del sistema (Dónde está...)
Puedo evaluar en “unidades” de ejecución sin estado (Me da igual dónde evaluar...)
Cualquier salida de una evaluación puede ser la entrada de la siguiente (Homogeneidad)
La infraestructura puede decidir cuándo y dónde evaluar (Declarativo)

¡Programación funcional!

Todo el big data se construye con los mismos bloques



- Tipo: Clave - valor
- Flujo: Map → **Shuffle** → Reduce
- Modo de operación: *batches*



- Tipo: RDD
- Flujo: **DAG** donde cada nodo ejecuta una función pura
- Modo de operación: *batches*, *microbatches*



- Tipo: Tupla
- Flujo: **DAG** donde cada nodo ejecuta una función pura
- Modo de operación: *stream*



- Tipo: Bag, Array, DataFrame
- Flujo: **DAG** donde cada nodo ejecuta una función pura
- Modo de operación: *batch*, *stream*



- Tipo: Tensor
- Flujo: **DAG** donde cada nodo ejecuta una función pura + autodiff
- Modo de operación: *batch*

- El streaming tiene que ver con los datos, no con el cálculo
 - Tuplas, mappings, listas sí soportan streaming
 - RDD, Dataframes, no lo soportan
- Muchos resultados tienen versión batch y streaming.

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$$

$$M_n = M_{n-1} \frac{x_n - M_{n-1}}{n}$$

Semántica declarativa

	Provincia	Municipio	Localidad	Código postal	Longitud	Latitud	Precio gasolina 95	Precio gasolina 98	Precio gasóleo A	Precio gasóleo B
0	ÁLAVA	ALEGRÍA-DULANTZI	ALEGRIA-DULANTZI	1240	-2.519194	42.842917	1.349	NaN	1.289	NaN
1	ÁLAVA	ALEGRÍA-DULANTZI	ALEGRIA-DULANTZI	1240	-2.509361	42.846028	NaN	NaN	1.259	0.726
2	ÁLAVA	AMURRIO	LARRINBE	1468	-2.989111	43.044333	1.279	1.379	1.219	NaN
3	ÁLAVA	AMURRIO	LEZAMA	1450	-2.967611	43.031889	1.339	1.454	1.275	NaN
4	ÁLAVA	ARRAIA-MAEZTU	MAEZTU/MAESTU	1120	-2.477917	42.753194	1.320	1.395	1.215	NaN

¿Cuál es el precio medio de la gasolina 95 para todas las provincias cuyo nombre empieza por P?

Semántica declarativa

	Provincia	Municipio	Localidad	Código postal	Longitud	Latitud	Precio gasolina 95	Precio gasolina 98	Precio gasóleo A	Precio gasóleo B
0	ÁLAVA	ALEGRÍA-DULANTZI	ALEGRIA-DULANTZI	1240	-2.519194	42.842917	1.349	NaN	1.289	NaN
1	ÁLAVA	ALEGRÍA-DULANTZI	ALEGRIA-DULANTZI	1240	-2.509361	42.846028	NaN	NaN	1.259	0.726
2	ÁLAVA	AMURRIO	LARRINBE	1468	-2.989111	43.044333	1.279	1.379	1.219	NaN
3	ÁLAVA	AMURRIO	LEZAMA	1450	-2.967611	43.031889	1.339	1.454	1.275	NaN
4	ÁLAVA	ARRAIA-MAEZTU	MAEZTU/MAESTU	1120	-2.477917	42.753194	1.320	1.395	1.215	NaN

¿Cuál es el precio medio de la gasolina 95 para todas las provincias cuyo nombre empieza por P?

```
p_provincias = gasolinas.Provincia.str.startswith('P')
p_precios = gasolinas[p_provincias]
for provincia, group in p_precios.groupby('Provincia'):
    print(provincia, group['Precio gasolina 95'].mean())
```

```
PALENCIA 1.287107
PALMAS (LAS) 1.1005437
PONTEVEDRA 1.37257
```

Semántica declarativa

	Provincia	Municipio	Localidad	Código postal	Longitud	Latitud	Precio gasolina 95	Precio gasolina 98	Precio gasóleo A	Precio gasóleo B
0	ÁLAVA	ALEGRÍA-DULANTZI	ALEGRIA-DULANTZI	1240	-2.519194	42.842917	1.349	NaN	1.289	NaN
1	ÁLAVA	ALEGRÍA-DULANTZI	ALEGRIA-DULANTZI	1240	-2.509361	42.846028	NaN	NaN	1.259	0.726
2	ÁLAVA	AMURRIO	LARRINBE	1468	-2.989111	43.044333	1.279	1.379	1.219	NaN
3	ÁLAVA	AMURRIO	LEZAMA	1450	-2.967611	43.031889	1.339	1.454	1.275	NaN
4	ÁLAVA	ARRAIA-MAEZTU	MAEZTU/MAESTU	1120	-2.477917	42.753194	1.320	1.395	1.215	NaN

¿Cuál es el precio medio de la gasolina 95 para todas las provincias cuyo nombre empieza por P?

```
(gasolinas.loc[lambda df: df.Provincia.str.startswith('P')]  
  .groupby('Provincia')  
  .agg({'Precio gasolina 95': np.mean}))
```

Precio gasolina 95	
Provincia	
PALENCIA	1.287108
PALMAS (LAS)	1.100544
PONTEVEDRA	1.372571

Semántica declarativa

	Provincia	Municipio	Localidad	Código postal	Longitud	Latitud	Precio gasolina 95	Precio gasolina 98	Precio gasóleo A	Precio gasóleo B
0	ÁLAVA	ALEGRÍA-DULANTZI	ALEGRIA-DULANTZI	1240	-2.519194	42.842917	1.349	NaN	1.289	NaN
1	ÁLAVA	ALEGRÍA-DULANTZI	ALEGRIA-DULANTZI	1240	-2.509361	42.846028	NaN	NaN	1.259	0.726
2	ÁLAVA	AMURRIO	LARRINBE	1468	-2.989111	43.044333	1.279	1.379	1.219	NaN
3	ÁLAVA	AMURRIO	LEZAMA	1450	-2.967611	43.031889	1.339	1.454	1.275	NaN
4	ÁLAVA	ARRAIA-MAEZTU	MAEZTU/MAESTU	1120	-2.477917	42.753194	1.320	1.395	1.215	NaN

¿Cuál es el precio medio de la gasolina 95 para todas las provincias cuyo nombre empieza por P?

```
select Provincia,  
       avg([Precio gasolina 95]) from gasolinas  
where substr(Provincia, 1, 1) = "P" group by Provincia
```

Precio gasolina 95	
Provincia	
PALENCIA	1.287108
PALMAS (LAS)	1.100544
PONTEVEDRA	1.372571

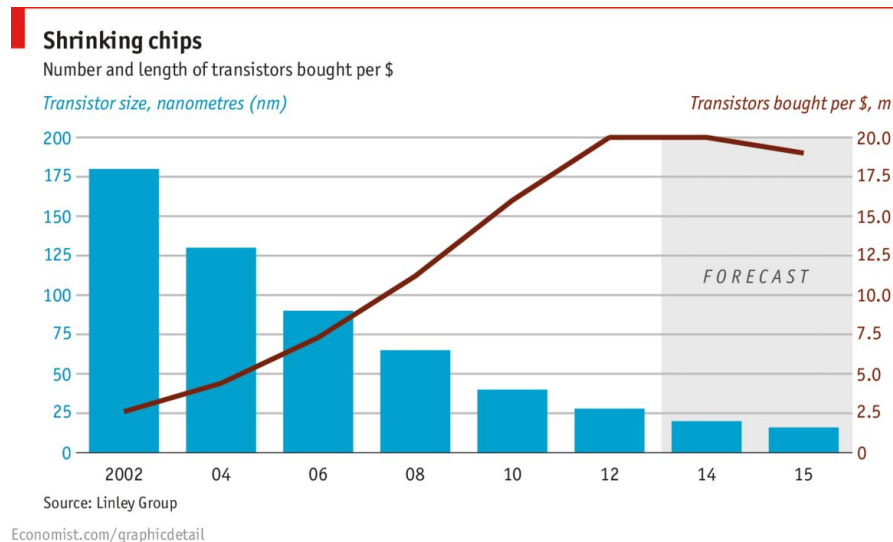


El futuro post-Moore

- ↑ Datos
- ↑ Cada vez más casos de negocio
- ↑ Complejidad de la analítica
- ↑↑ Expectativas
- = Número de transistores

Sólo nos queda optimizar lo que ya tenemos

- Herramientas más complejas
- Arquitecturas heterogéneas / específicas (half float)
- Planificadores + optimizadores
- [10 things I hate about Pandas](#)
- Mejores drivers, compresión, UDF compiladas
- Cada vez más funcional
- Cada vez más declarativo



SQL?

```
SELECT *,
        acq_ipos / num_investments AS acq_rate
FROM (
    SELECT CASE WHEN i.investor_name IS NULL THEN 'NO INVESTOR'
               ELSE i.investor_name
            END AS "Investor name",
           COUNT(DISTINCT c.permlink) AS num_investments,
           COUNT(DISTINCT
                CASE WHEN c.status IN ('ipo', 'acquired') THEN c.permlink
                END) AS acq_ipos
    FROM crunchbase\_companies
    LEFT JOIN crunchbase\_investments
        ON c.permlink = i.company_permlink
    GROUP BY 1
    ORDER BY 2 DESC
) t
```

[Referencia](#)

DataFrame / Tabla

```
(crunchbase_companies
.merge(crunchbase_investments,
      how='left',
      left_on='permalink',
      right_on='company_permalink')
.assign(investor_name=lambda df: df.investor_name.fillna('NO INVESTOR'),
        acq_ipos=
            lambda df:[row.permalink if row.status in ('ipo', 'acquired') else np.nan
                        for row in df.itertuples()])
.eval('num_investments = permalink')
.groupby('investor name', as_index=False)
.agg({'investor_name': 'nunique',
      'acq_ipos': 'nunique'})
.eval('acq_rate = acq_ipos / num_investments')
.sort_values('num_investments', ascending=False)
)
```

DataFrame / Tabla

```
(crunchbase_companies
.merge(crunchbase_investments,
      how='left',
      left_on='permalink',
      right_on='company_permalink')
.assign(investor_name=lambda df: df.investor_name.fillna('NO INVESTOR'),
        acq_ipos=
            lambda df:[row.permalink if row.status in ('ipo', 'acquired') else np.nan
                        for row in df.itertuples()])
.eval('num_investments = permalink')
.groupby('investor name', as_index=False)
.agg({'investor_name': 'nunique',
      'acq_ipos': 'nunique'})
.eval('acq_rate = acq_ipos / num_investments')
.sort_values('num_investments', ascending=False)
)
```



UDF (Función pura)

Pandas, R, Spark, PySpark, SparklyR, Flink, Arrow...

¡Muchas Gracias!

Guillem Borrell

Team lead

guillem.borrell@kernel-analytics.com

Kernel Analytics, S.L.

Barcelona

Balmes 89, Planta 6º, pta. 4ª, 08008 +34 932506437

Madrid

Joaquín Bau, 2 1º C 28036 Madrid +34 915022390

For further information:

www.kernel-analytics.com

info@kernel-analytics.com

Follow us:

twitter.com/kernelanalytics

linkedin.com/company/kernel-analytics