


IA para la Industria: detectar problemas de manufacturing

Industriarako Adimen Artifiziala: manufacturing arazoak detektatzea

28/10/2020 – 16/12/2020



Sobre mi

 @Neuw84
@IKERLANOfficial



Ángel Conde Manjón

Líder del equipo de **Data Analytics e Inteligencia Artificial** en

*Mantenimiento
Predictivo*

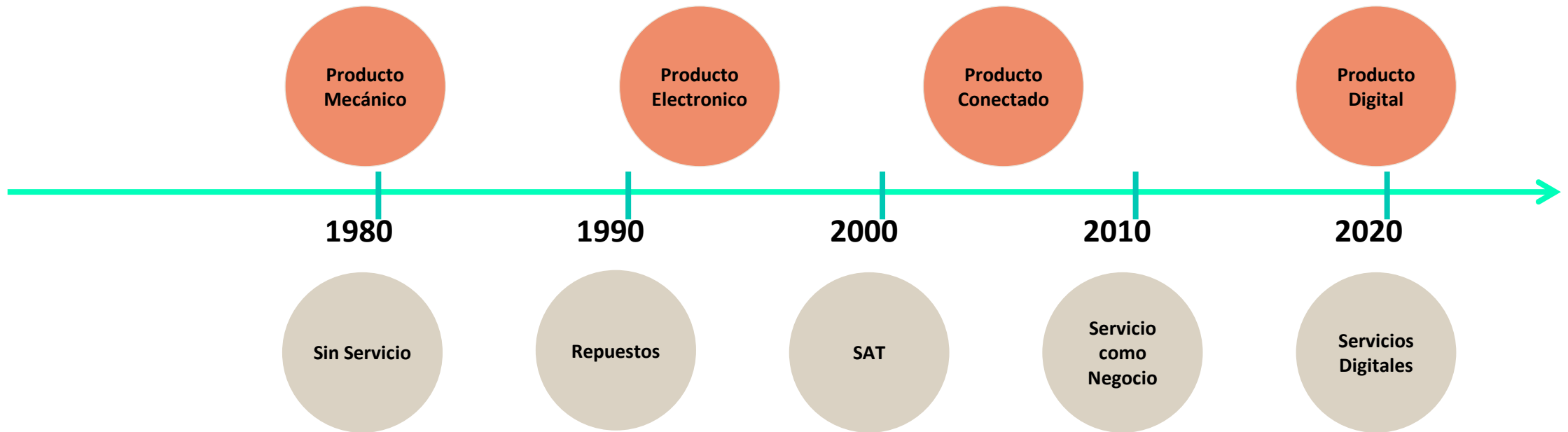
*Analítica
Avanzada*

*Monitorización
en Tiempo Real*

Plataformas Digitales



Plataformas Digitales



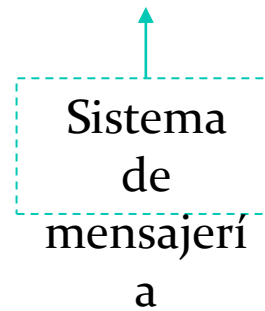
Plataformas.... Digitales?



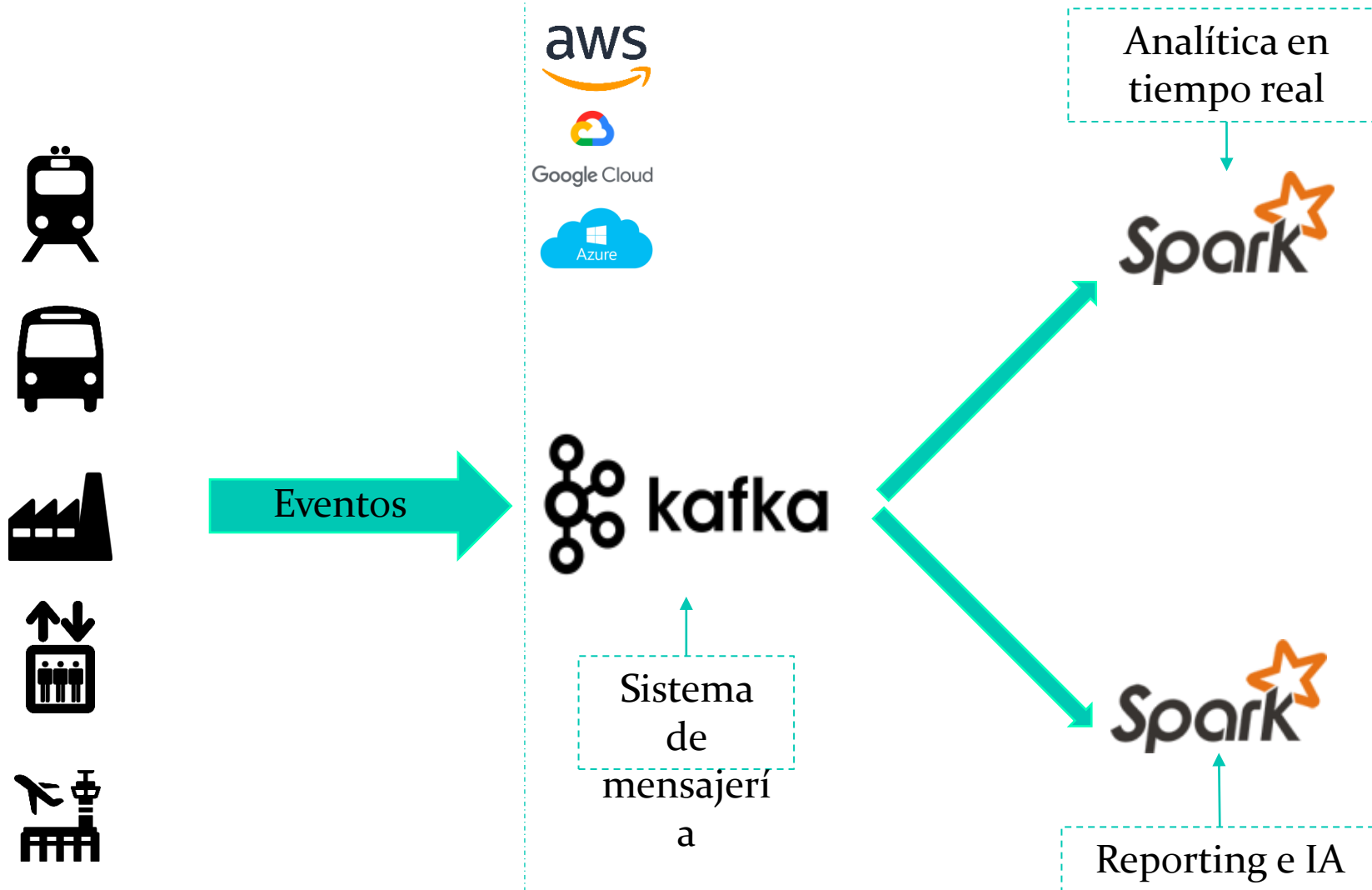
Plataformas.... Digitales?



Plataformas.... Digitales?



Plataformas.... Digitales?



Plataformas.... Digitales?



Eventos



Sistema de mensajería

Analítica en tiempo real

Spark

Data Lake



Spark

Reporting e IA

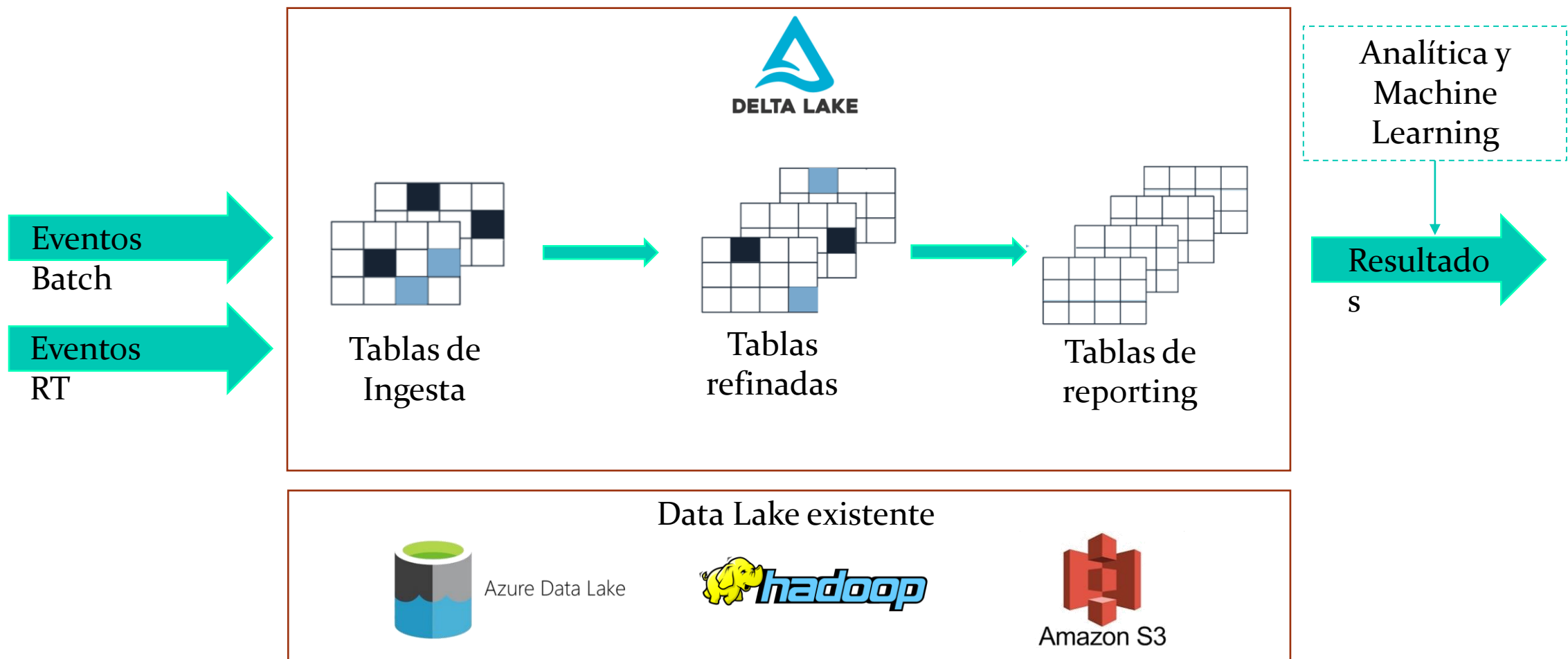
Data Lakes

- ¿Es la **realidad** así de simple?

La realidad

- ¿Cómo gestiono la **calidad de los datos**?
- ¿Cómo hago para **no** tener que **recalcular** (*arquitectura λ*)?
- Si necesito **transacciones**.... ¿Data Warehouse comercial?
- ¿**Cuántas herramientas** necesito, qué **capacidad**?

La solución



Ventajas

- Forzar chequeos de **calidad de los datos**.
- **Consistencia**, mezclar batch y streaming.
- **Atomicidad**, un trabajo acaba o no.
- **Open Source**.

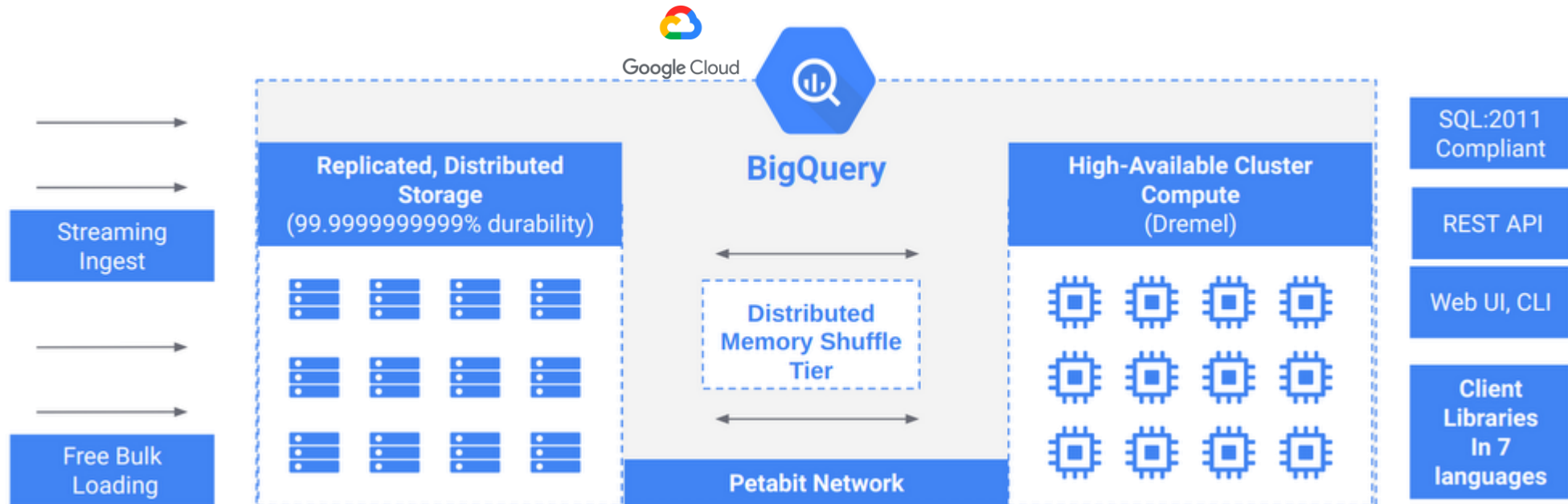
¿Aún así, podemos mejorar?

- **Sin embargo...**

- **¿Cómo gestiono capacidad y
computo?**

¿Cómo podemos simplificar?

- ¿Y si podemos **simplificar** todavía más?



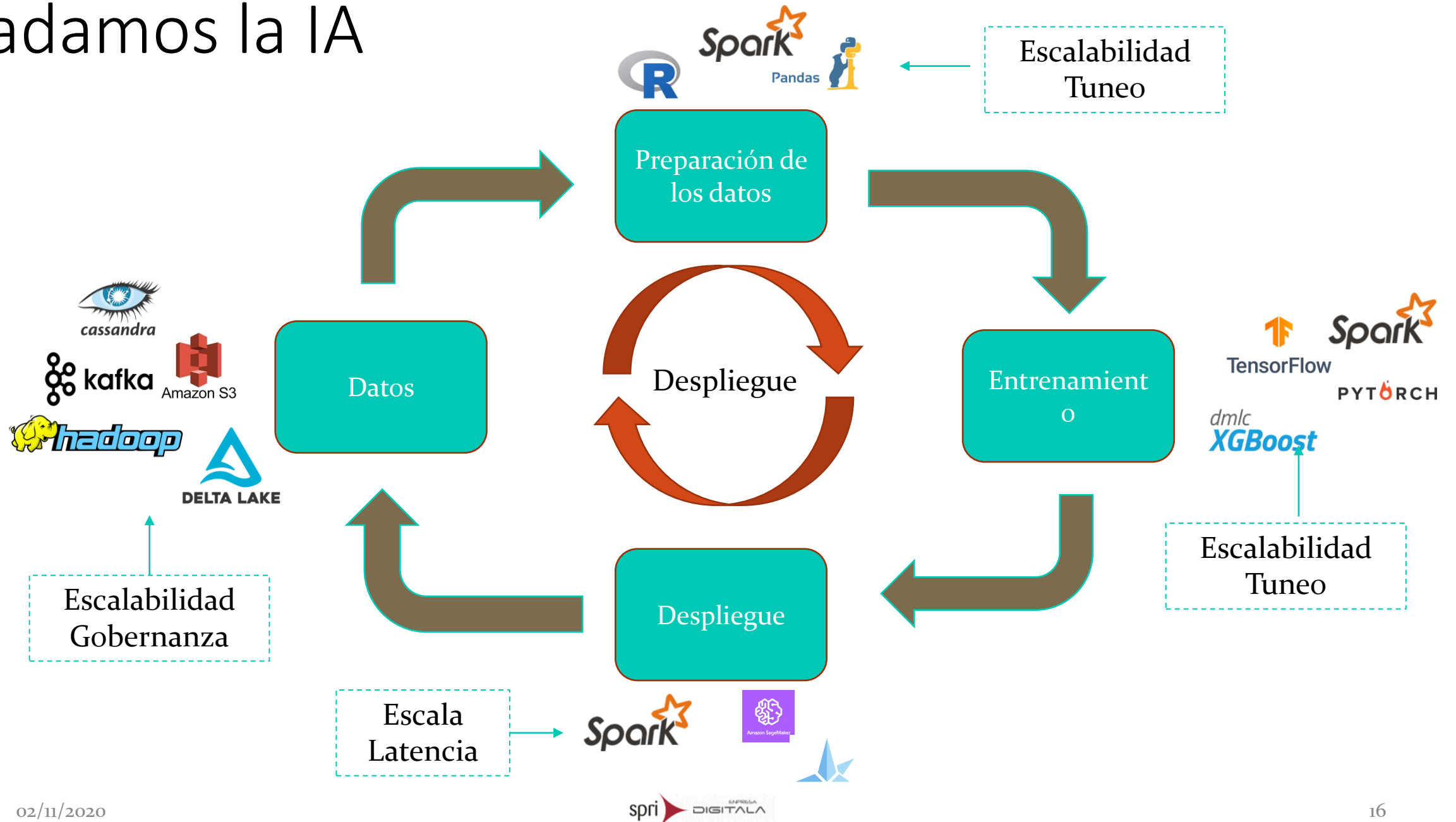
Ventajas...?

- Puedo consultar mis datos por SQL.
- No me preocupo ni del almacenamiento ni del computo.

Pero....

- La dependencia con el proveedor aumenta.
- Cálculo de costes complejo...

Añadamos la IA



La IA en producción

- **¿Cómo llevar a producción todo esto?**
- ¿Cómo puedo registrar un histórico de modelos, configuraciones y pruebas?
- ¿Cómo pongo en producción los modelos?



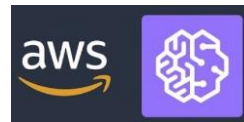
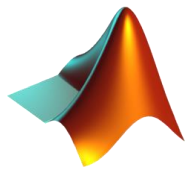
Microsoft Machine Learning for Apache Spark



Inteligencia Artificial en la nube...

- Gran capacidad de computo disponible....
- Algoritmos inteligentes de búsquedas de parámetros.
- Perdida de importancia del “feature engineering”.

AUTOMATED Machine Learning



¿Aún así podemos mejorar?

- ¿Cómo lo aplico al **mundo industrial?**

Más problemas del mundo industrial...

- Latencia... puede la nube hacer que ejecute algoritmos de control?



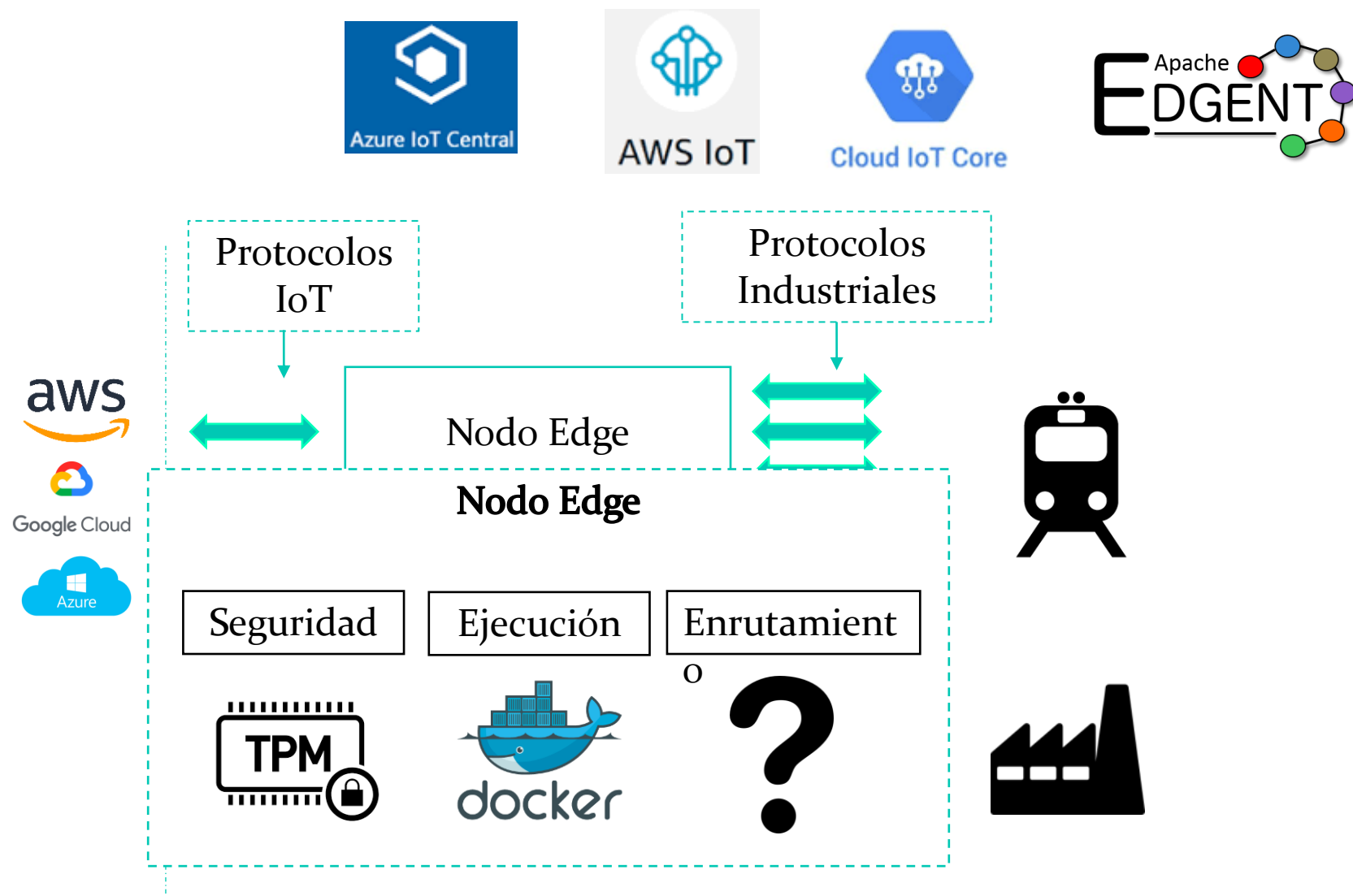
- Energía, según la energía disponible me puede interesar balancear.



- Que paradigma uso... Entrenar en nube → Despliegue en Fog/Edge?

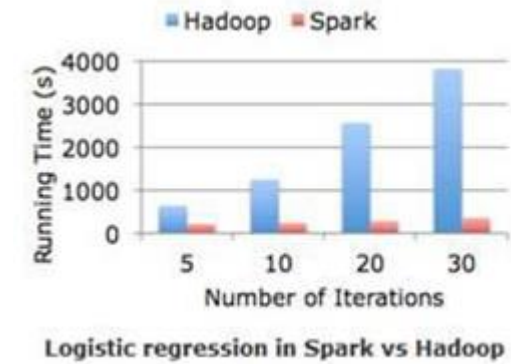
Federated learning

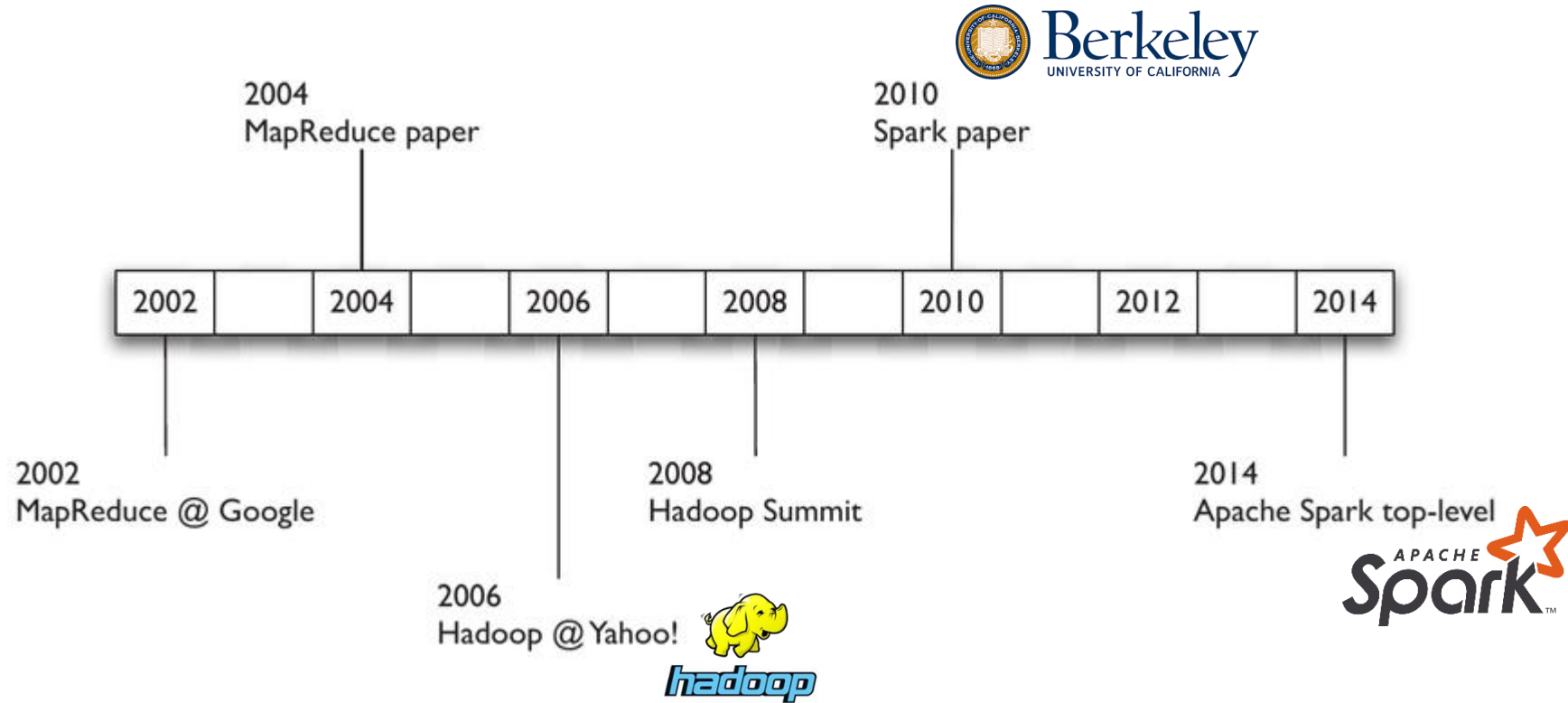
IoT e IA





- Analíticas Big data en memoria.
- Soporte SQL.
- Aprendizaje automático*.
- Procesamiento en tiempo real.
- Grafos.
- API Scala/Java/Python/R.





	Open Source	Managed Service	Auto-Awesome	Batch	Streaming	Iterative	Interactive	Pipelines	Optimizer	High-level API	Unified API	Unified Engine	Exactly Once	No Lambda	State	Timers	Watermarks	Windowing	Triggers
MapReduce				Blue									Blue						
Hadoop	Red	Red		Red									Red						
Flume			Orange	Orange				Orange	Orange	Orange			Orange						
Storm	Green			*	Green		*		*	*			*	*	*				
Spark	Blue	Blue		Blue	Blue	Blue	Blue	Blue	Blue	Blue	*	Blue	Blue	Blue	*		*	*	*
MillWheel				*	Red			Red					Red	Red	Red	Red	Red		
Flink	Orange			Orange	Orange	Orange	Orange	Orange	Orange	Orange		Orange	Orange	Orange	Orange		*	*	*
Cloud Dataflow	*	Green	Green	Green	Green			Green	Green	Green	Green		Green	Green	Green	Green	Green	Green	Green

Foundations of streaming SQL / Tyler Akiday @Google

- Estándar en el mundo Big Data para analíticas Batch.
 - Y después de Kafka, la pieza que veréis en casi todas las arquitecturas.
- Comunidad:  databricks™  Microsoft  
- Usado en todas las grandes empresas.







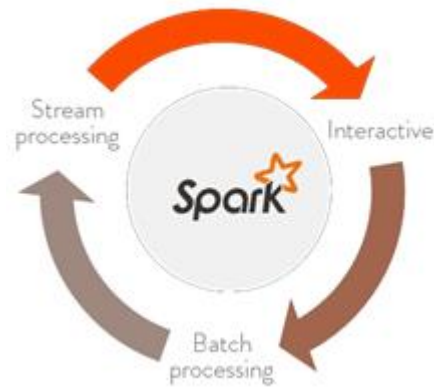




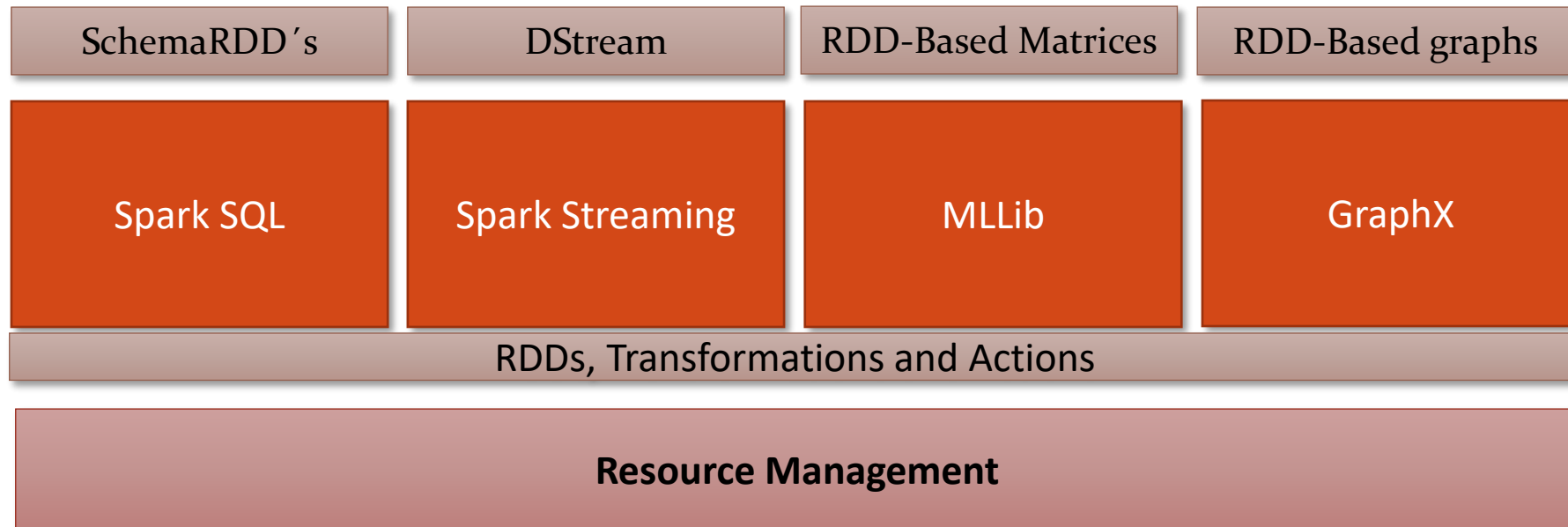




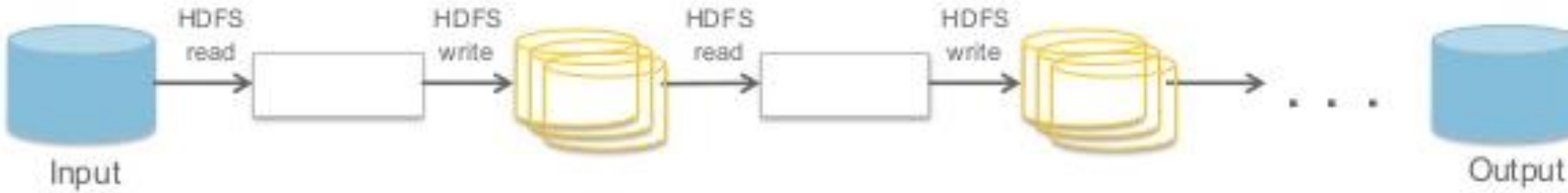




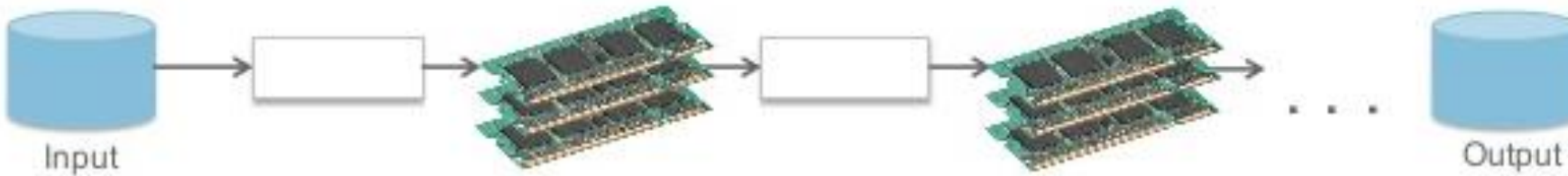
***SQL:** como interfaz común



Hadoop MapReduce: Data Sharing on Disk



Spark: Speed up processing by using Memory instead of Disks



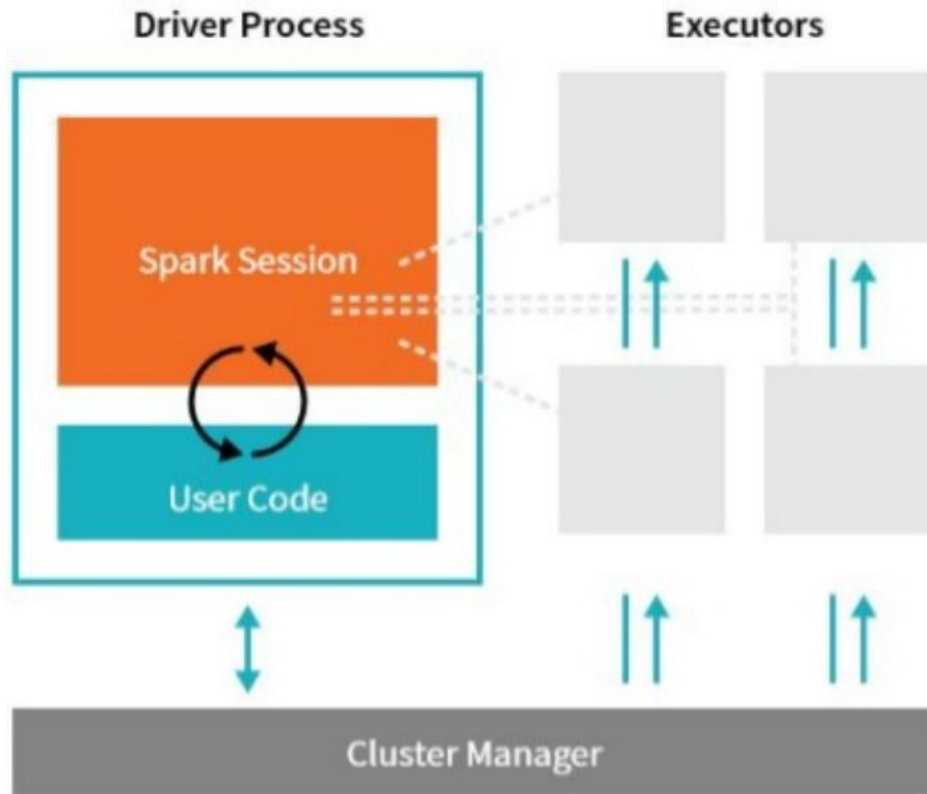
- Abaratamiento bestial de memoria RAM, ergo....
- Pasamos a guardar “estados” intermedios en **memoria**.



- Gran compatibilidad con todo el ecosistema Hadoop (conectores de entrada y salida).
- Ojo con la “calidad” y el “Locality Level”
- Si usamos S3 y compañía olvidarnos del tema.

Tasks

Index	ID	Attempt	Status	Locality Level	Executor	Launch Time	Duration	GC Time	Accumulators	Error
2	2748	0	SUCCESS	PROCESS_LOCAL	2.compute.internal	2014/09/18 00:09:56	2 ms			
1	2747	0	SUCCESS	PROCESS_LOCAL	2.compute.internal	2014/09/18 00:09:56	2 ms			
0	2746	0	SUCCESS	PROCESS_LOCAL	2.compute.internal	2014/09/18 00:09:56	3 ms			
4	2750	0	SUCCESS	PROCESS_LOCAL	2.compute.internal	2014/09/18 00:09:56	1 ms			



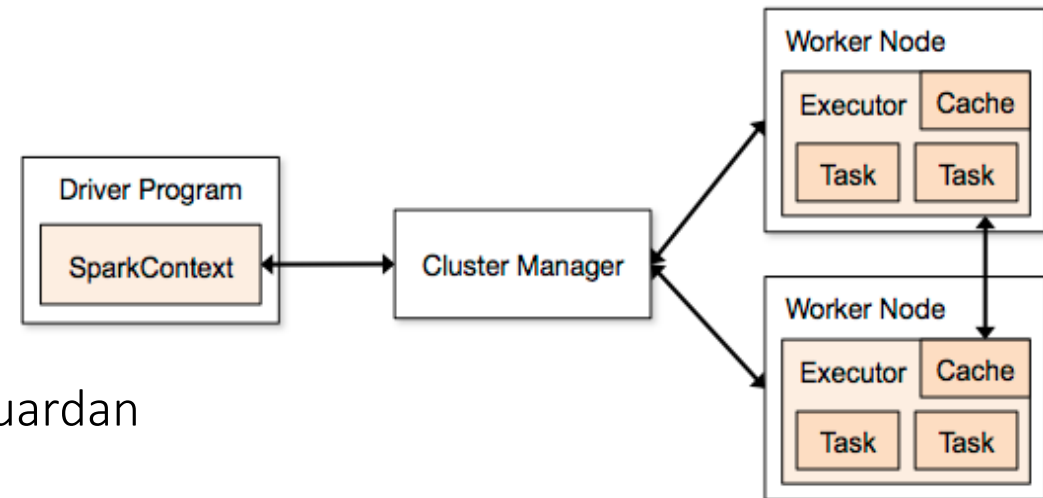
- En Spark, usamos un orquestador de clúster (YARN, Mesos, K8s o bien Spark en si mismo) para controlar las diferentes máquinas del clúster y asignar recursos y tareas a estas.
- Pueden ejecutarse múltiples aplicaciones de Spark en un clúster al mismo tiempo.

Driver

- Programa que nos permitirá inicializar un contexto o sesión de Spark usando un gestor de clúster.
- Desde aquí podremos interactuar y ejecutar operaciones mediante el contexto o sesión.
- El que almacena los resultados (ojo con el “collect”).
- Pensar en donde inicializamos las conexiones de la DB.

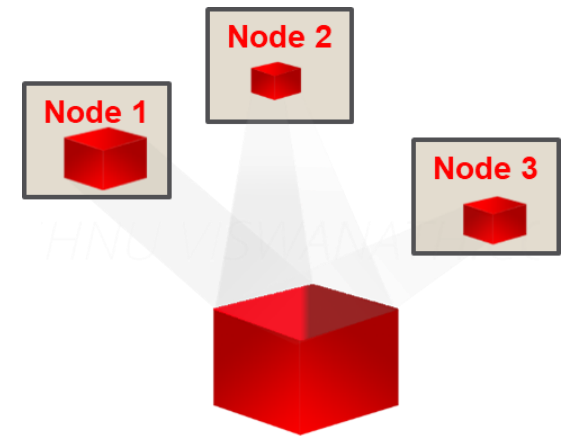
Executors

- Los que hacen el trabajo, ejecutan tareas y son los que guardan los datos.
- Controlar el reparto de memoria.
- Controlar tiempo empleado en GC.



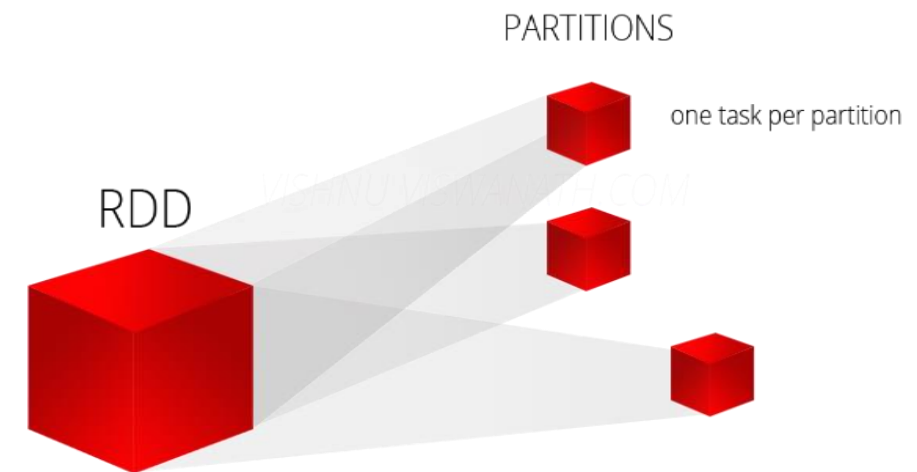
RDD o Resilient Distributed Datasets.

- **Resilient:** son inmutables y la cadena de operaciones que hagamos sobre ellos se “guarda”.
- **Distributed:** están distribuidos por el clúster.
- ***Dataset:*** contienen datos.



- Los RDDs están divididos en “Partitions”.
- Cuando ejecutamos una acción, se lanza una tarea por partición.
- Cuanto más particiones, **más posible** paralelismo.

- Spark decide las particiones de un RDD cuando lo crea.
- Esto también se puede especificar por el usuario.
- Las particiones se distribuyen por todos los nodos del clúster.



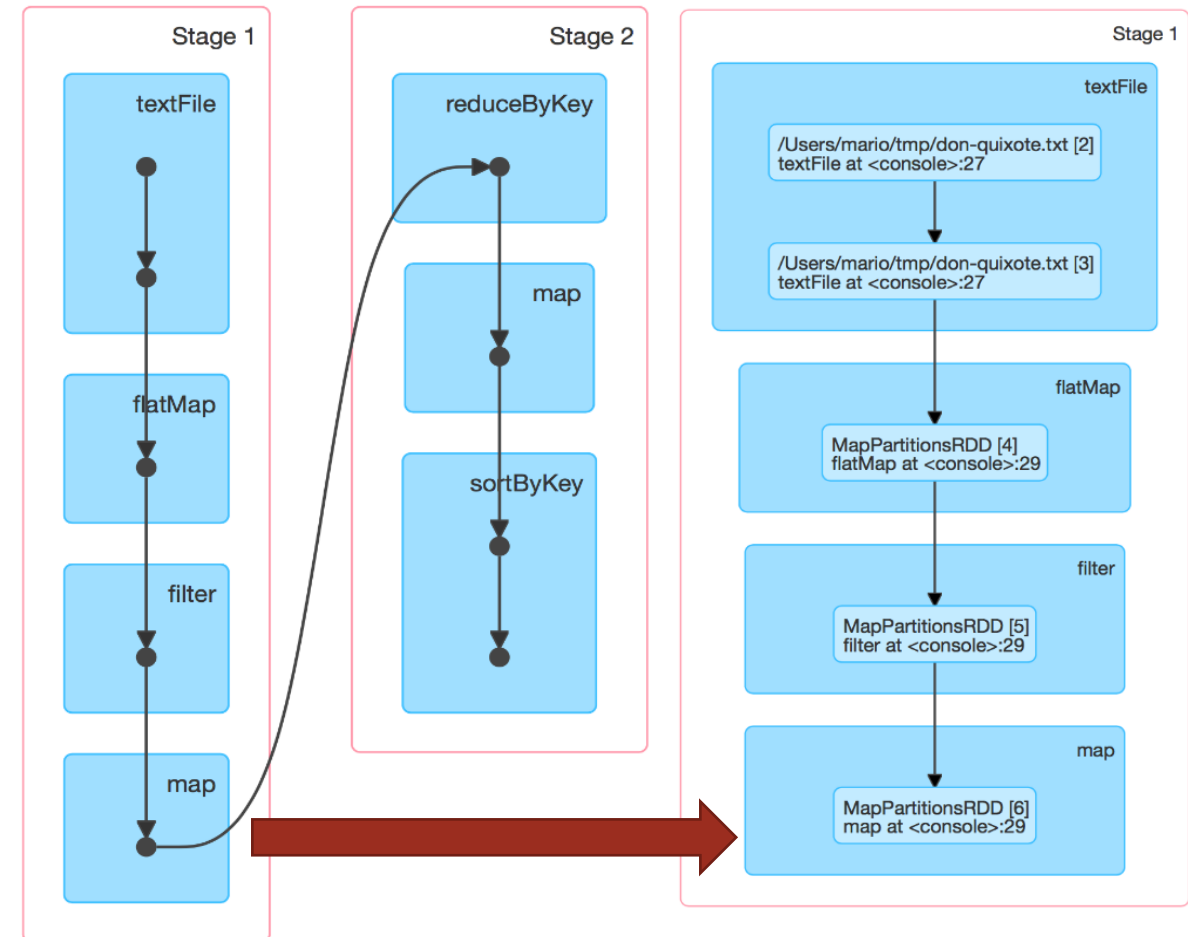
Dos tipos de operaciones en Spark:

- Transformaciones, que crean un nuevo RDD o DF/DS.
 - *e.g:* map, filter
- Acciones, que crean un valor retornado al driver después de ejecutarse.
 - *e.g:* reduce, count

Las transformaciones son “Lazy”:

- Solo se ejecutan cuando una acción lo requiere.
- Aunque hay alguna transformación que no cumple la norma sortByKey.

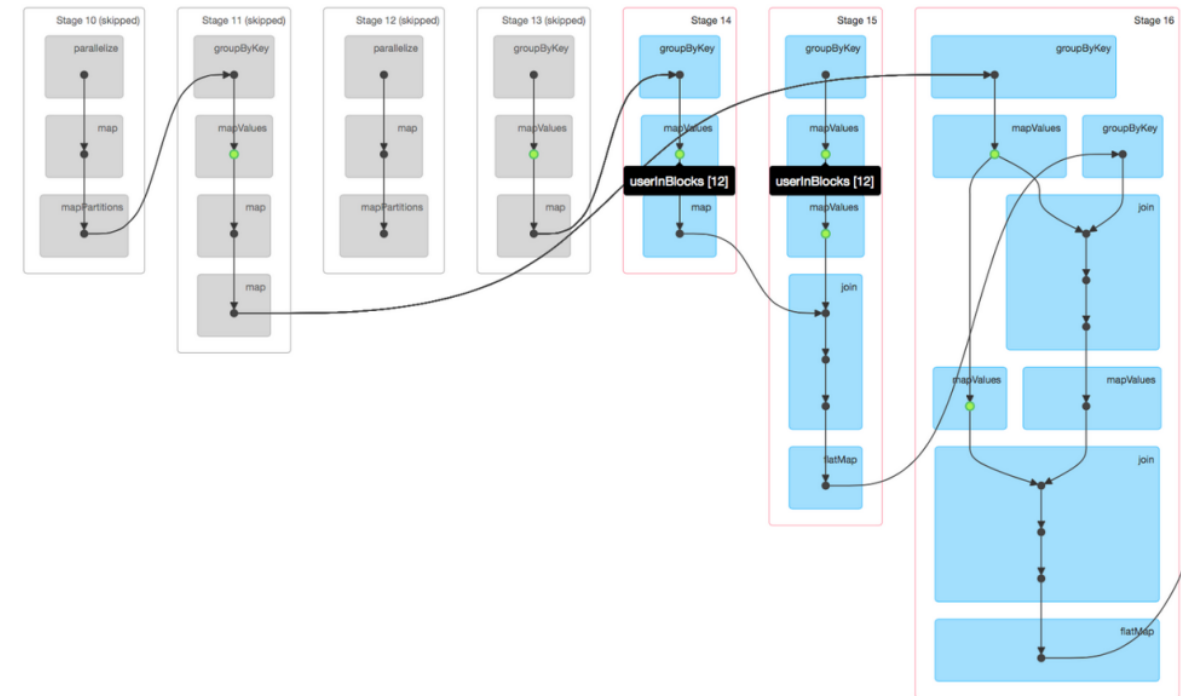
- **DAG**: un grafo dirigido finito sin ciclos.
- Contiene una secuencia de vértices y aristas en la que cada arista está directamente conectada con la siguiente en secuencia.
- Es una generalización del modelo MapReduce. Las operaciones se pueden optimizar mejor.
- La función del **DAG** se ve claramente en trabajos complejos.



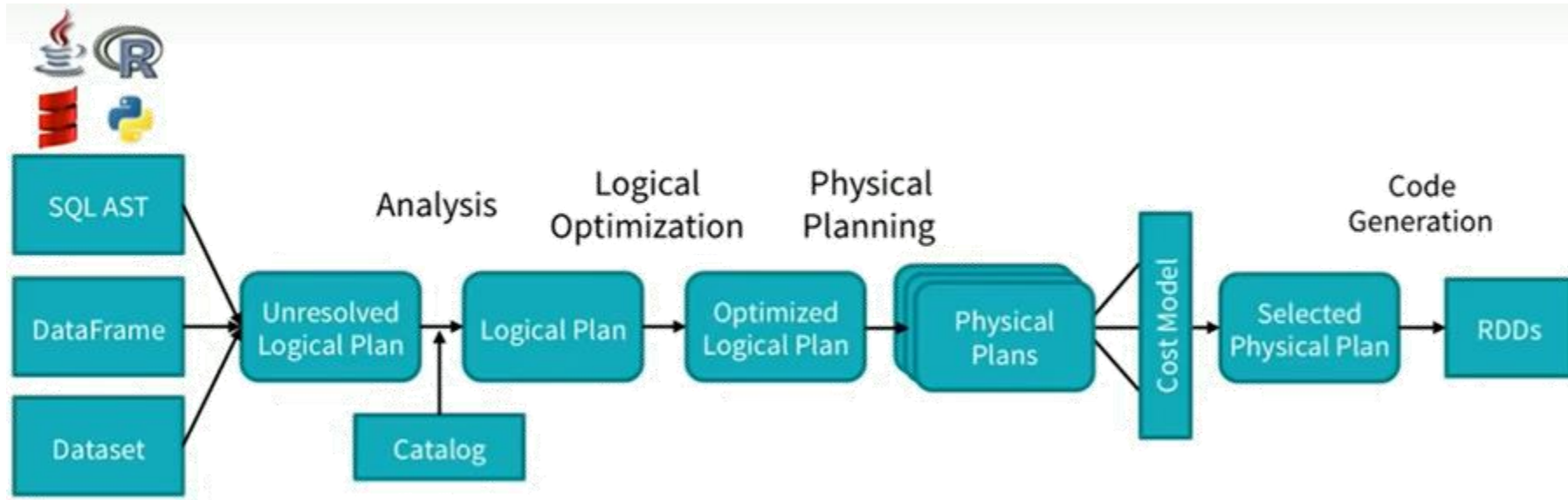
- Usar la Cache de Spark es **importante** por rendimiento, aprende como funciona.
- Cuando una acción se ejecuta, se ejecuta toda su cadena de operaciones.
- Cachear hace que rompamos esa cadena.
- Donde usarla:
 - Cuando reusamos en mismo conjunto de datos (e.g. ML).
 - Cuando una acción es costosa, nos puede ayudar a recuperarnos cuando un ejecutor falla.

Details for Job 4




Status: SUCCEEDED
 Completed Stages: 22
 Skipped Stages: 4
 ▶ Event Timeline
 ▶ DAG Visualization



- SQL ANSI 2011 completo.
- Soporte para todas las **consultas** del TPC-DS.
- Por rendimiento, usar Encoders (*Datasets* y *Kryo*).



Structuring Apache Spark 2.0: SQL, DataFrames, Datasets And Streaming / Michael Armbrust @Databricks

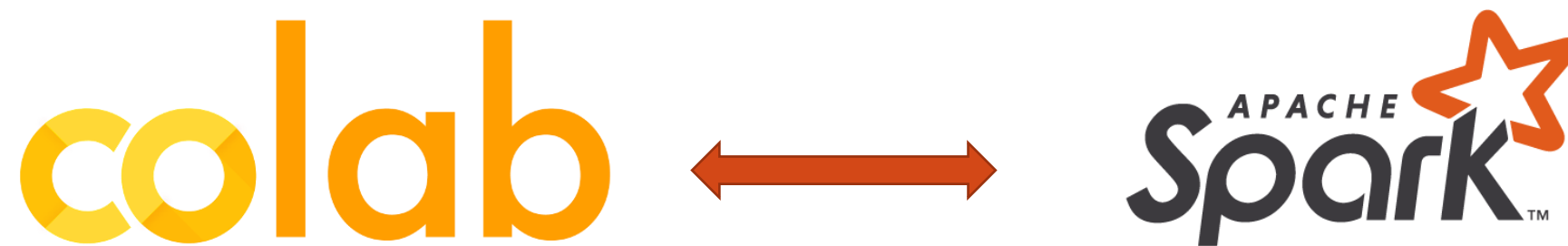
- Mejorar en **Streaming**, más fuerte  **Flink**
- Mayor Integración con  TensorFlow o 
 - ¿ En que quedará la integración de “barrier scheduling” en la 3.0.0?
- Integración ecosistema Python (Apache Arrow)?
 - Usa los UDFs de PySpark cuando no tengas muchos datos y quieras ir rápido.
 - Si necesitas rendimiento, UDFs en Scala y llamarlos desde wrappers de Python.
- Integración con nuevos sistemas de ficheros (Ozone).
- Sobrevivir al “serverless”



Google
BigQuery



Amazon Athena



<https://cutt.ly/jgRpd3K>