Infraestructura

Almacenamiento y procesamiento de datos

Gonzalo Rivero

Big Data en investigación social y opinión pública

22 a 26 de julio, 2019

Por qué vamos a hablar de esto

- Almacenamiento de datos
  1. Necesitamos almacenar mayor volumen de datos
  2. Los datos no siempre son rectangulares
- Procesamiento de datos
  1. Modelos computacionalmente caros
  2. Mayor demanda de baja latencia

Qué veremos hoy

- Almacenamiento
  1. Archivos de texto plano
  2. Archivos binarios
  3. Bases relacionales (SQL)
  4. Base no-relacionales (No-SQL)
- Procesamiento de datos
  1. Paralelismo

Formatos de texto plano

1. Comma-separated values
   - Delimited values
2. Archivos no delimitados
   2.1 Separación de datos es menos ambiguo (puede usar comas)
   2.2 Necesita diccionario externo

Archivos de texto plano

+ Legible por humanos
+ Facil de editar manualmente
+ La mayoria de software puede leerlo y escribirlo
- No incluye metadatos:
  0.1 No define el tipo de cada variable
  0.2 No aporta información sobre categoricas
- Lento de escribir y escribir
- Impreciso (convertir float a string)

Hojas de cálculo

1. Nunca usar Excel

Archivos binarios

+ Específico (y optimizado) para cada lenguaje
  - Contiene información sobre tipos
  - Datos comprimidos
    - Mayor velocidad I/O
    - Menor espacio
- Puede dificultar compartir datos
- Posibilidad de que los datos no sean accesibles

Bases de datos relacionales

- Estructuras centralizadas para almacenar datos
- Comunes en entornos empresariales
- Conceptos (y lenguaje) útiles en gestión de datos

Una *base de datos* es:

- Una colección lógica de datos significativos
- Una colección de programas que ayudan a un usuario a creater y mantener una base de datos.

Ventajas

1. Control de acceso a usuarios
2. Permite concurrencia controlada en el acceso
3. Hace cumplir restricciones de integridad
4. Provee de métodos de respaldo y recuperación
5. Permite limitación de inconsistencias

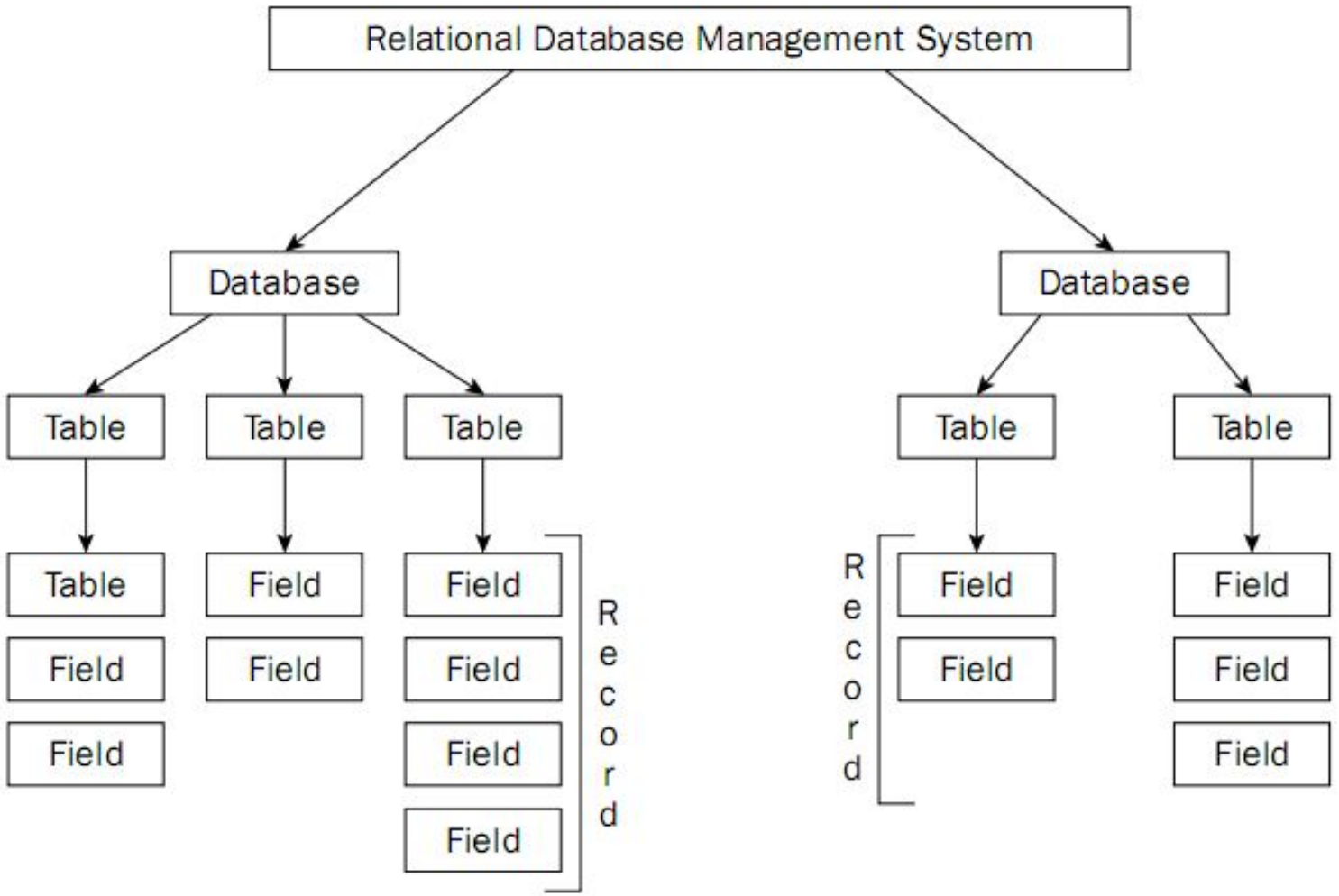
Desventajas

1. Nuevo sistema de funcionalidades que los usuarios deben comprender
2. Alto coste inicial
3. Riesgos de fallo por estructura descentralizada
4. Alto consumo de disco

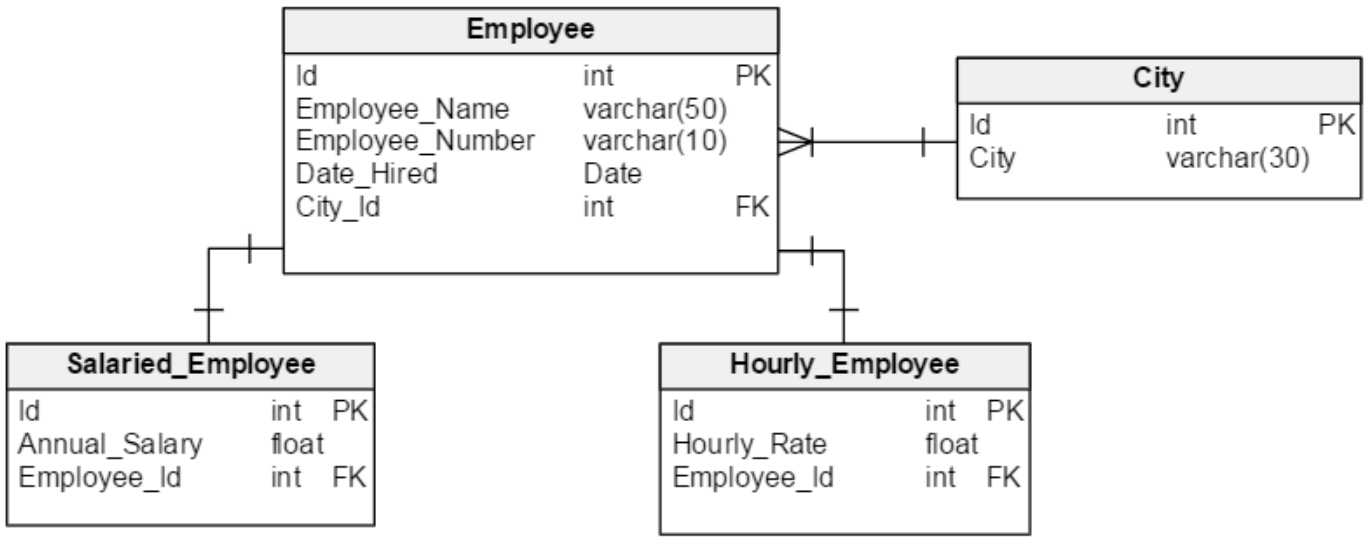
Base de datos relacional (RDBMS)

1. Las relaciones entre valores se almacenan como una tabla
2. Almacenamiento de datos en *tablas*
3. Cada tabla tiene una clave principal (*primary key*) que identifica cada tupla de forma única
4. Los datos están normalizados en relación a un esquema
5. Define restricciones de integridad
6. Es posible distribuir almacenamiento
7. Diseñada para organizaciones que gestionan grandes volúmenes de datos y sirven a múltiples usuarios

Un ejemplo en abstracto



Un ejemplo de tablas



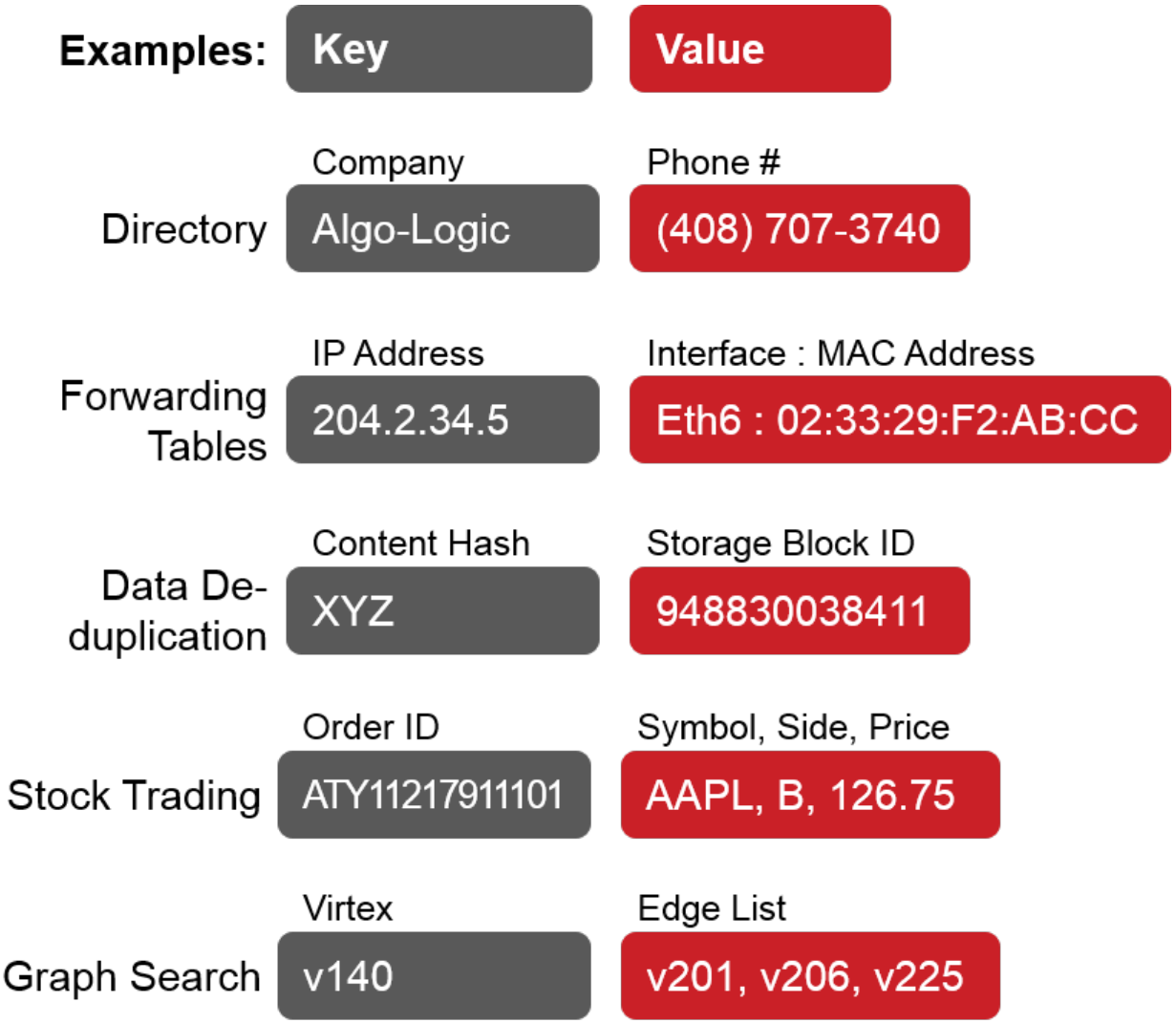
NoSQL

- NoSQL es una base de datos diseñada para almacenar datos desestructurados que pueden no tener una estructura fija
- Es útil pensar en NoSQL como si fuese una base de datos que no es RDBMS
- La ventaja es que, relajando restricciones ACID, facilita leer y escribir y simplifica distribución (disponibilidad)
- Permite almacenar datos con estructuras que cambian a lo largo del tiempo o cuando no es posible comprometerse a una determinada estructura

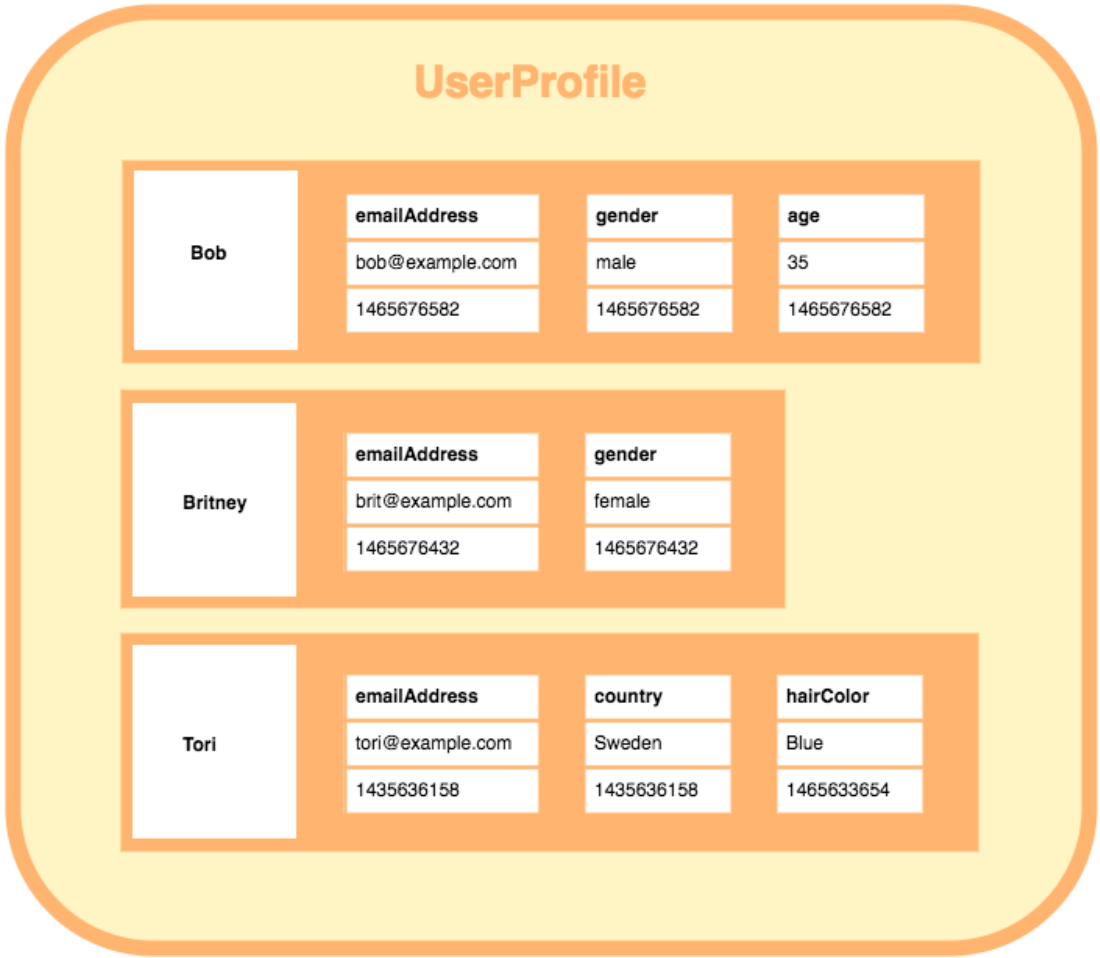
Tipos de bases de datos NoSQL

1. Clave/Valor – Redis
   1.1 Almacenan los datos como pares clave/valor y por tanto no tienen un lenguaje de búsqueda. Solo necesitan los verbos `get`, `put`, and `delete`. Eso explica su alto rendimiento
2. Columnas anchas — Cassandra o HBase.
   2.1 Las columnas correspondientes a cada entrada se almacenan juntas. Las filas no tienen que tener el mismo número de columnas. Las columnas pueden añadirse a cada fila sin tener que modificar todas las filas.
3. Documentos — MongoDB
   3.1 Similar a clave/valor pero almacena un documento (como JSON). No permite relaciones entre registros o fusiones.
4. Grafos — Neo4J.
   4.1 Almacena las relaciones entre dos nodos que están relacionados de alguna manera.

Clave/valor


Examples:
Key
Value
Company
Phone #
Directory
Algo-Logic
(408) 707-3740
IP Address
Interface : MAC Address
Forwarding Tables
204.2.34.5
Eth6 : 02:33:29:F2:AB:CC
Content Hash
Storage Block ID
Data De-duplication
XYZ
948830038411
Order ID
Symbol, Side, Price
Stock Trading
ATY11217911101
AAPL, B, 126.75
Virtex
Edge List
Graph Search
v140
v201, v206, v225

Columna ancha



Documento

Key	Document

1001	<pre>{
	  "CustomerID": 99,
	  "OrderItems": [
	  { "ProductID": 2010,
	  "Quantity": 2,
	  "Cost": 520
	  },
	  { "ProductID": 4365,
	  "Quantity": 1,
	  "Cost": 18
	  }],
	  "OrderDate": "04/01/2017"
	}</pre>

1002	<pre>{
	  "CustomerID": 220,
	  "OrderItems": [
	  { "ProductID": 1285,
	  "Quantity": 1,
	  "Cost": 120
	  }],
	  "OrderDate": "05/08/2017"
	}</pre>

Grafo

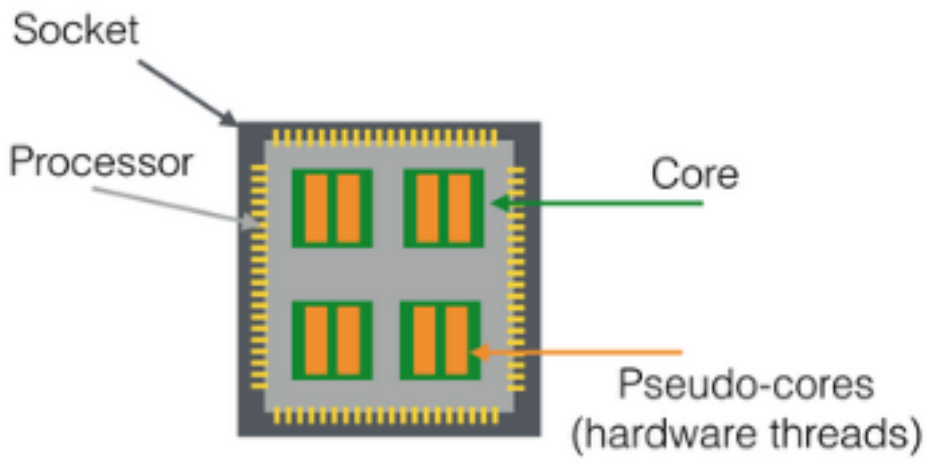

IS FRIEND OF
LIKES
name:
Zush1 Zam
neo4j
SERVES
LOCATED IN
cuisine:
1Sush1
IS FRIEND OF
LIKES
SERVES
location:
New York
name:
1Sush1
LOCATED IN

Procesar datos: Procesamiento paralelo

- Con datos digitales habitualmente tenemos problemas
  - Limitaciones por CPU: Los cálculos son lentos
  - Limitaciones por memoria: Los datos no caben en memoria
  - Limitaciones por I/O: Leer y escribir lleva mucho tiempo
  - Limitaciones por network: Transmisión de datos es costosa
- Para muchas tareas estadísticas, R es lo bastante rápido pero
  - Nuevos datos digitales suelen ser de mayor tamaño
  - Modelos de ML suelen ser complejos
  - Workflows suelen requerir muchas repeticiones
- Arquitecturas modernas pueden aprovechar existencia varios núcleos

CPUs

- Antigüamente los ordenadores tenían una única unidad de procesamiento central (CPU)
- Lo habitual es que tengan más de un procesador/núcleo
  - Mi computador: Dos procesadores con dos núcleos cada uno
  - Servicios comunes de EC2: 12 procesadores de 4 núcleos



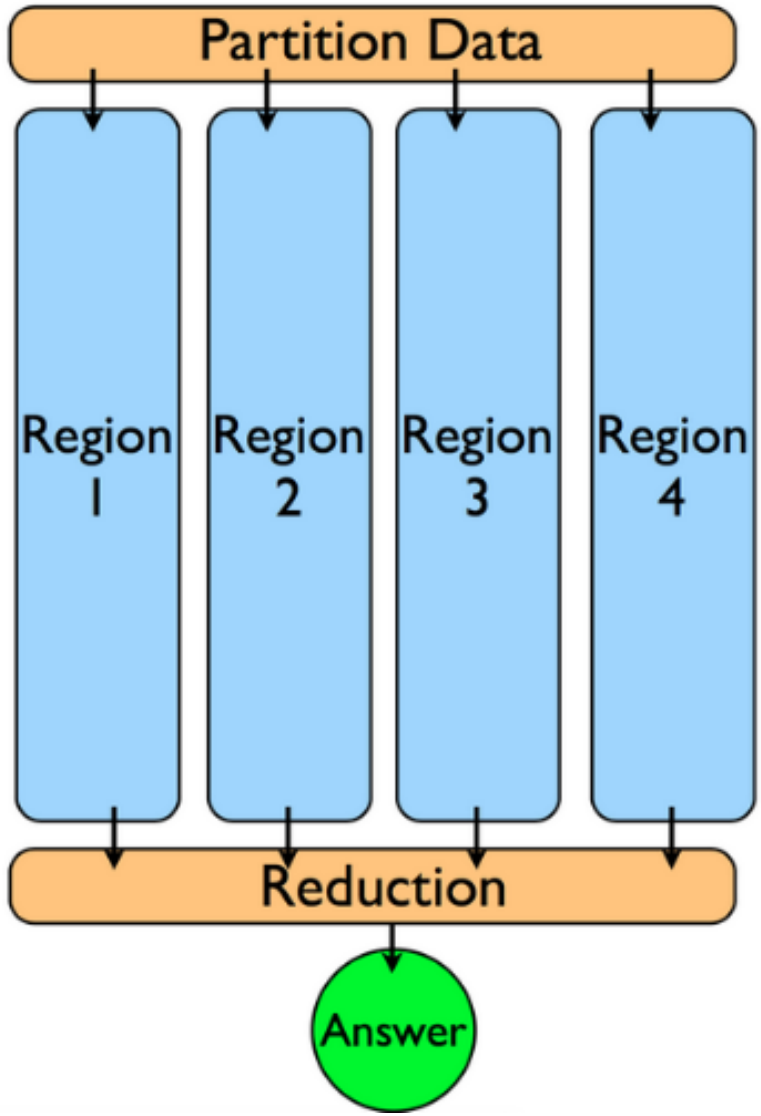
- Es posible ejecutar más operaciones al mismo tiempo

Dos tipos de problemas

- Mi programa es demasiado lento:
  - Aumentar el número de núcleos para aumentar el número de operaciones
  - `parallel` o `multicore`
- Mi problema es demasiado grande:
  - Aumentar el número de computadores para aumentar memoria
  - `parallel` o `snow`

Paralelismo imperfecto

- Paralelizar en si mismo es una operación costosa
- Usar 4 núcleos no implica que 1/4 del tiempo de usar un núcleo
- Dividir el trabajo entre núcleos y recombinarlos lleva tiempo
- Más todavía si tiene que hacerse sobre network
- Crecimiento es menos que linear y paralelizar puede llevar más que usar un único núcleo



Forking/Threads

- R solo ejecuta un único hilo
- Paralelismo está basado en "forks" que crean diferentes procesos
- No hay memoria compartida: cada nuevo proceso es un clon del anterior y lleva sus propios datos.
- Forking no existe para Windows.
- En diferentes procesadores (en un cluster) R genera nuevos procesos (no copias).