Big Data

Diplomado de Datos 2021

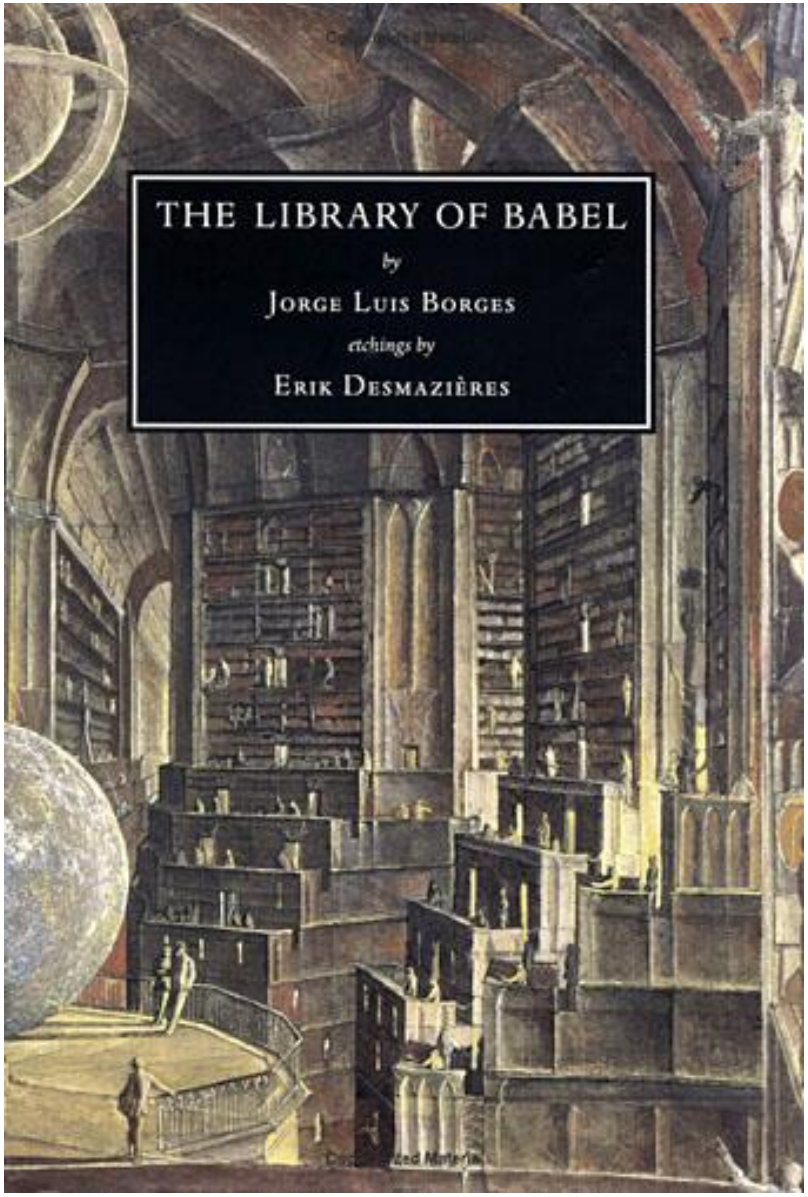
Clase 5: Recuperación de Información a gran escala (I)

Aidan Hogan
aidhog@gmail.com

Manejando datos de texto

Sobrecarga de información

Si no se pudiera usar buscadores …

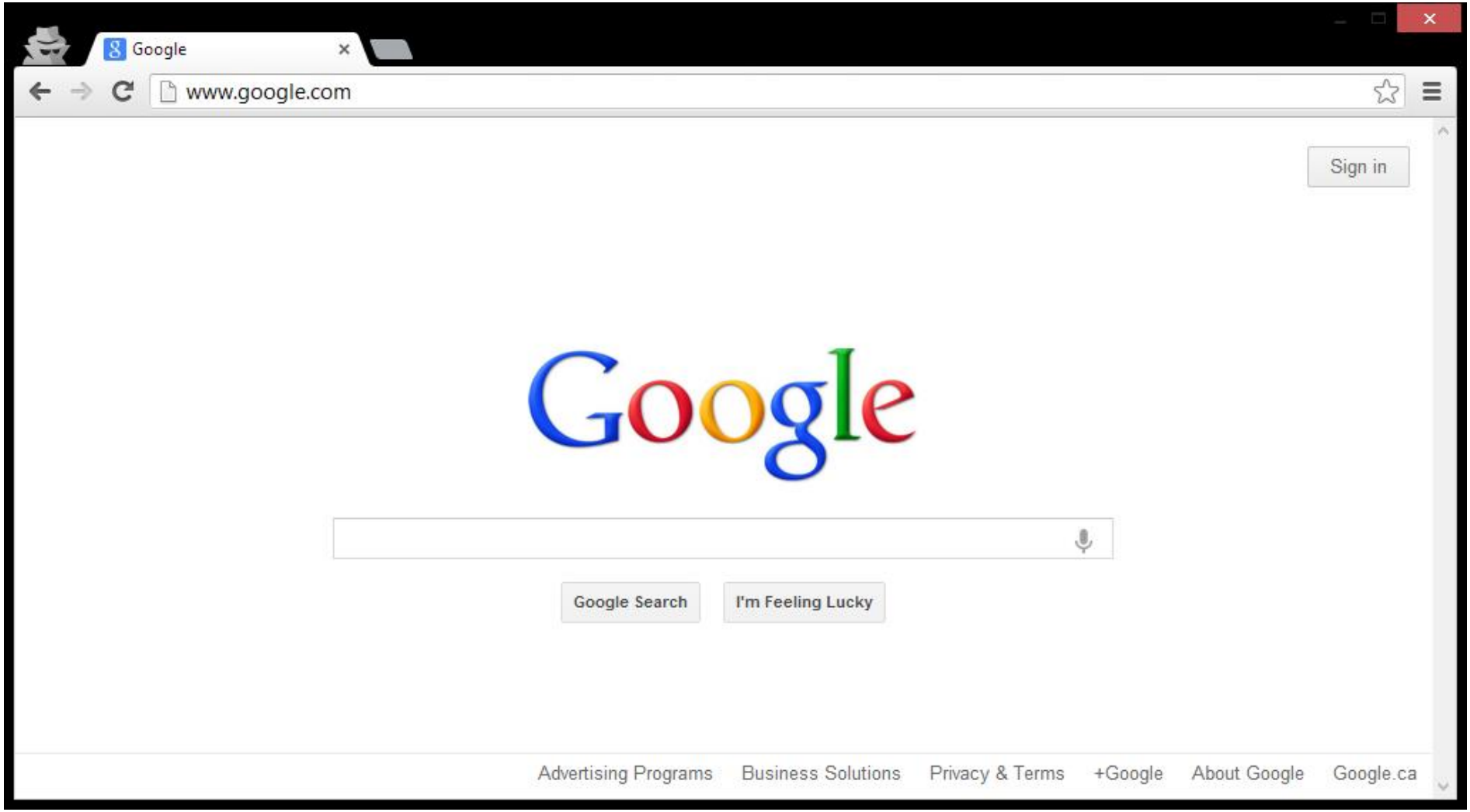


- Contiene todos los libros posibles con
  - 25 caracteres en el alfabeto
  - 80 caracteres por línea
  - 40 líneas por página
  - 410 páginas
  - 410 x 40 x 80 = 1.312.000 car.
  - $25^{1.312.000}$ libros

- Contendría cada libro imaginable
  - Incluyendo un libro con la ubicación de todos los libros útiles

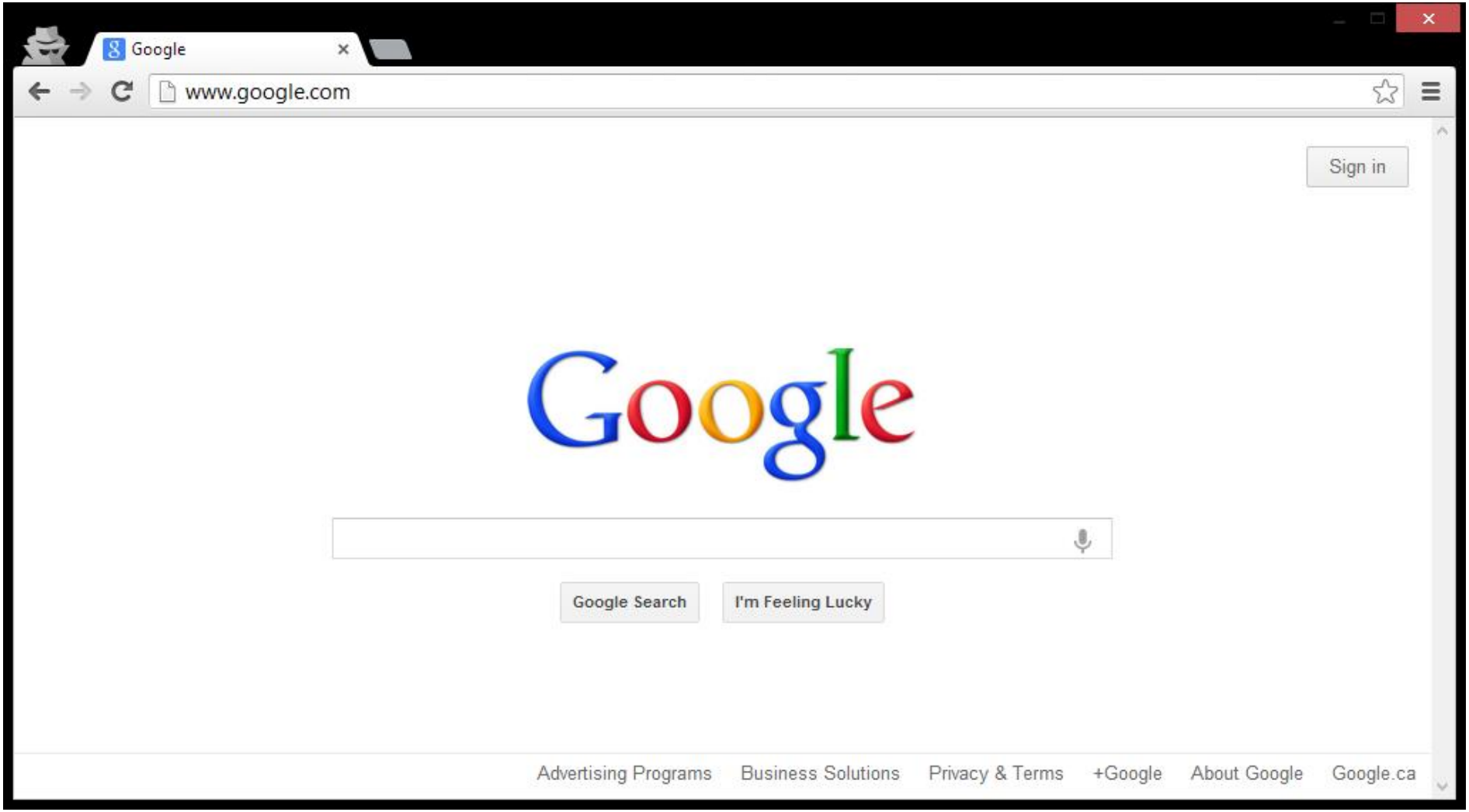
Información total = Cero información

El libro que indexa la biblioteca

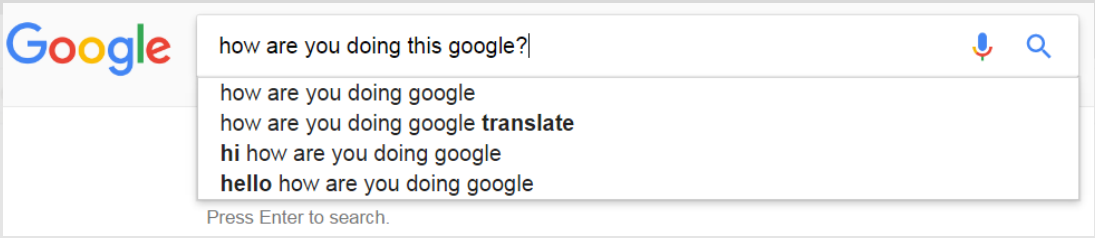


Búsqueda en la Web

Implementando la búsqueda de Google



Implementando la búsqueda de Google



¿Qué procesos y algoritmos necesita Google para implementar su búsqueda de la Web?

Crawling

1. Parsear enlaces de las páginas
2. Ordenar los enlaces para descargar
3. Descargar páginas, GOTO 1

Indexación

1. Parsear keywords de las páginas
2. Indexear sus keywords
3. Administrar actualizaciones

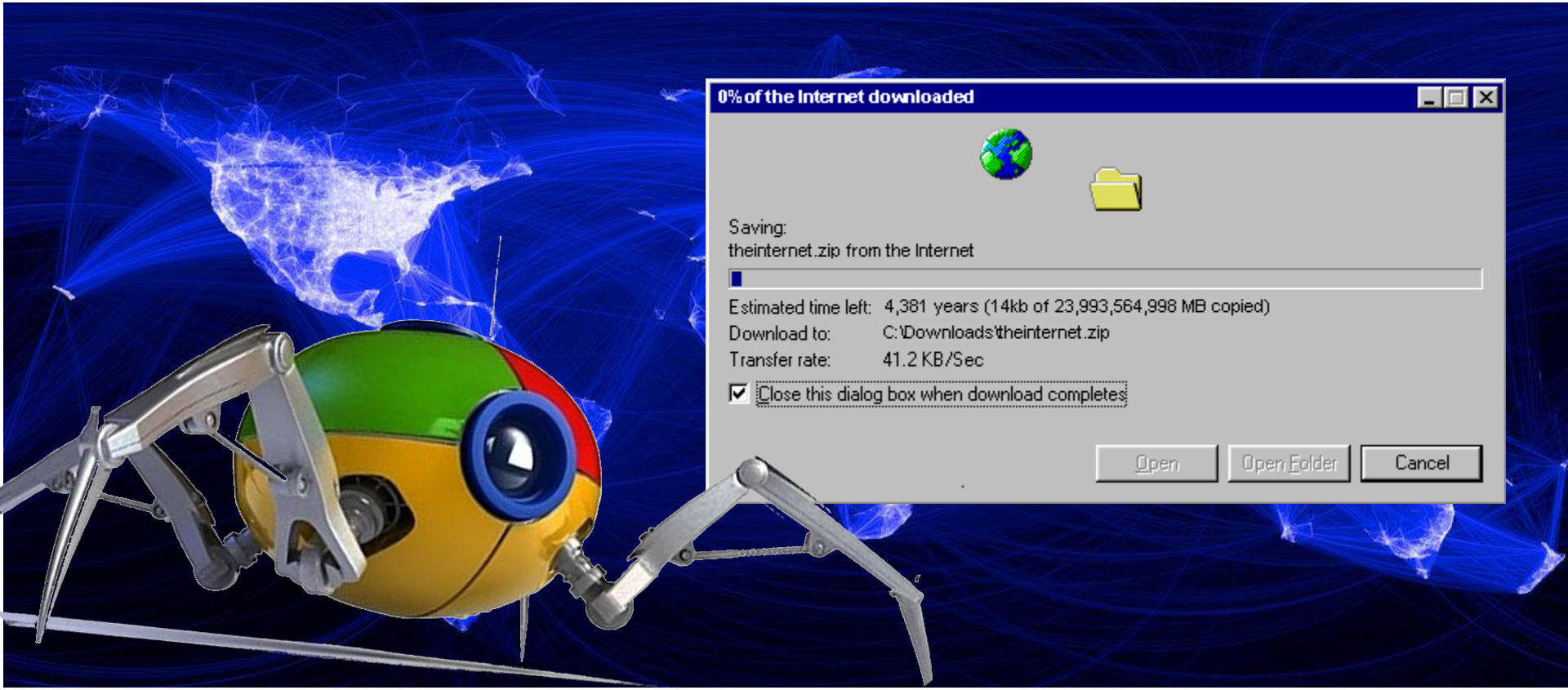
Ranking

1. ¿Qué tan relevante es una página?
2. ¿Qué tan importante es?
3. ¿Cuántos clics tiene?

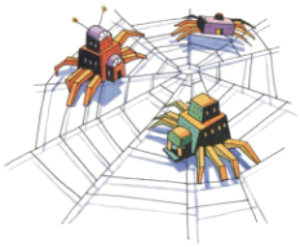
…

Recuperación de información: Crawling

¿Cómo sabe Google acerca de la Web?



Crawling

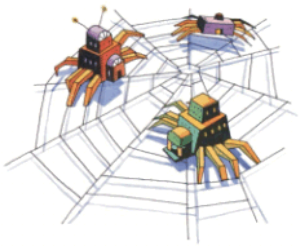


Descargar la Web ☺

```
crawl(list seedUrls)
    frontier_i = seedUrls
    while(!frontier_i .isEmpty())
        new list frontier_i+1
        for url : frontier_i
                page = downloadPage(url)
                frontier_i+1.addAll(extractUrls(page))
                store(page)
        i++
```

¿Qué le falta a este código?

Crawling: Evitar Ciclos

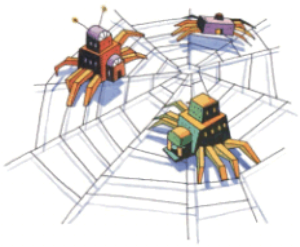


Descargar la Web ☺

```
crawl(list seedUrls)
    frontier_i = seedUrls
    new set done
    while(!frontier_i .isEmpty())
        new list frontier_i+1
        for url : frontier_i
                page = downloadPage(url)
                done.add(url)
                frontier_i+1.addAll(extractUrls(page).removeAll(done))
                store(page)
        i++
```

¿Cómo es el rendimiento?

Crawling: Rendimiento



Descargar la Web ☺

```
C:\Users\Aidan>ping twitter.com

Pinging twitter.com [199.16.156.198] with 32 bytes of data:
Reply from 199.16.156.198: bytes=32 time=118ms TTL=50
Reply from 199.16.156.198: bytes=32 time=120ms TTL=50
Reply from 199.16.156.198: bytes=32 time=120ms TTL=50
Reply from 199.16.156.198: bytes=32 time=125ms TTL=50

Ping statistics for 199.16.156.198:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 118ms, Maximum = 125ms, Average = 120ms

C:\Users\Aidan>
```

```
done.add(url)
```

- La mayoría del tiempo se gastará esperando conexiones
- El uso del disco duro/CPU será casi 0
- El ancho de banda no será maximizado

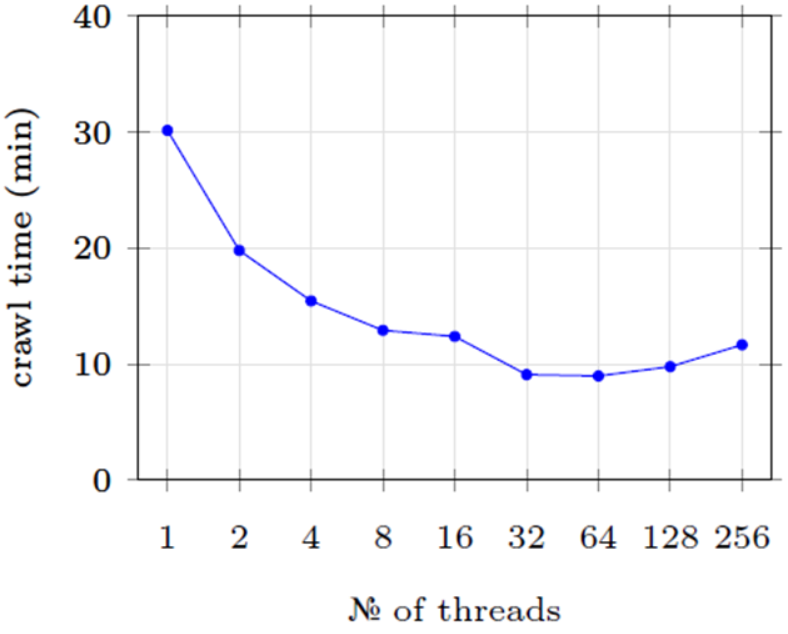
¿Cómo es el rendimiento? ?

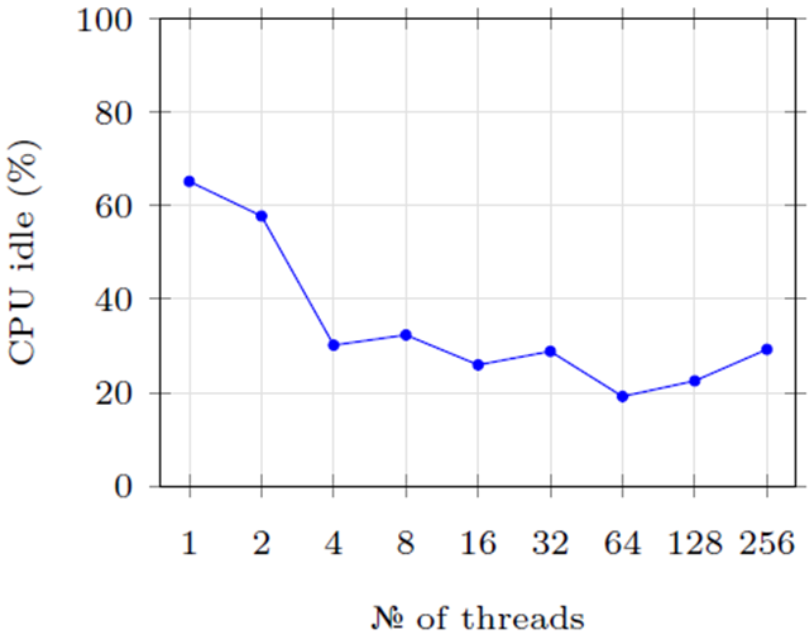
Crawling: "Multi-threading" es importante

```
crawl(list seedUrls)
    frontier_i = seedUrls
    new set done
    while(!frontier_i .isEmpty())
        new list frontier_i+1
        new list threads
        for url : frontier_i
                thread = new DownloadThread.run(url,done,frontier_i+1)
                threads.add(thread)
        threads.poll()
        i++
  DownloadThread: run(url,done,frontier_i+1)
      page = downloadPage(url)
      synchronised: done.add(url)
      synchronised: frontier_i+1.addAll(extractUrls(page).removeAll(done))
      synchronised: store(page)
```

Crawling: "Multi-threading" es importante

(por ejemplo) Haciendo un crawl de mil URLs …



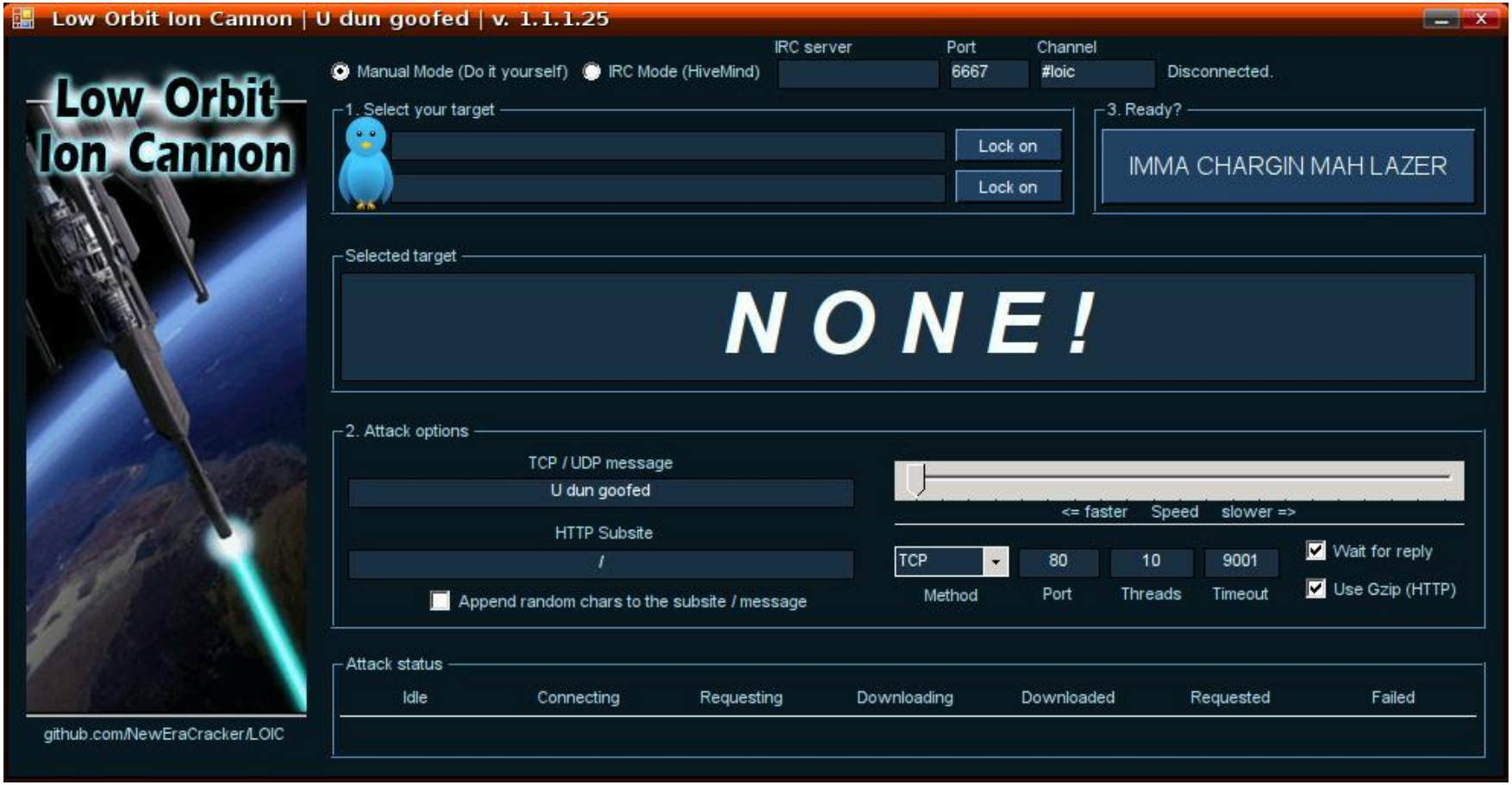


¿Cuál es el peligro de usar muchos hilos?

Crawling: ¡Es importante ser cortés!

Ataque de Denegación de Servicio (Distribuido)

"(Distributed) Denial of Server Attack": (D)DoS



Crawling: Evitar hacer un (D)DoS

Christopher Weatherhead

- ¡Preso por 18 meses!

… pero más probable que tu IP será baneada

Crawling: Planificador de sitio web

```
crawl(list seedUrls)
    frontier_i = seedUrls
    new set done
    while(!frontier_i .isEmpty())
        new list frontier_i+1
        new list threads
        for url : schedule(frontier_i)
                    # maximiza el tiempo entre dos peticiones al mismo sitio
                thread = new DownloadPageThread.run(url,done,fronter_i+1)
                threads.add(thread)
        threads.poll()
        i++
  DownloadPageThread: run(url,done,frontier_i+1)
      page = downloadPage(url)
      synchronised: done.add(url)
      synchronised: frontier_i+1.addAll(extractUrls(page).removeAll(done))
      synchronised: store(page)
```

Protocolo de exclusión de robots

http://website.com/robots.txt

```
User-agent: *
Disallow: /
```

No se permiten bots en este sitio web

```
User-agent: *
Disallow: /user/
Disallow: /main/login.html
```

No se permiten bots en la carpeta `/user/` ni la página `/main/login.html`

```
User-agent: googlebot
Disallow: /
```

Banea solo el bot con el “user-agent” `googlebot`.

Robots Exclusion Protocol (non-standard)

```
User-agent: googlebot
Crawl-delay: 10
```

Le dice al **googlebot** que haga una petición no más de una vez cada 10 segundos

```
User-agent: *
Disallow: /
Allow: /public/
```

Banea todo menos la carpeta **/public/** para todos los bots

```
User-agent: *
Sitemap: http://example.com/main/sitemap.xml
```

Da un enlace al "site map"

Crawling: Puntos importantes

- Lista de semillas: Puntos de entrada para el crawling
- Frontera: Los próximos URLs a descargarse
- Lista de vistos: Para evitar ciclos
- Multihilos: Mantiene a las máquinas ocupadas
- Cortesía: No sobrecargar los sitios web
  - Aplicar un retraso entre dos peticiones
  - Seguir lo que se dice en el archivo `robots.txt`
  - Revisar si hay un "site-map"

Crawling: Distribución

¿Cómo implementaríamos un crawler distribuido?

```
for url : frontier_i-1
      map(url,count)
```

Beneficios similares al uso de multihilos

¿Cuál será el cuello de botella al aumentar el número de máquinas?

El bando ancho o la cortesía (los retrasos entre peticiones)

Crawling: ¿Toda la Web?

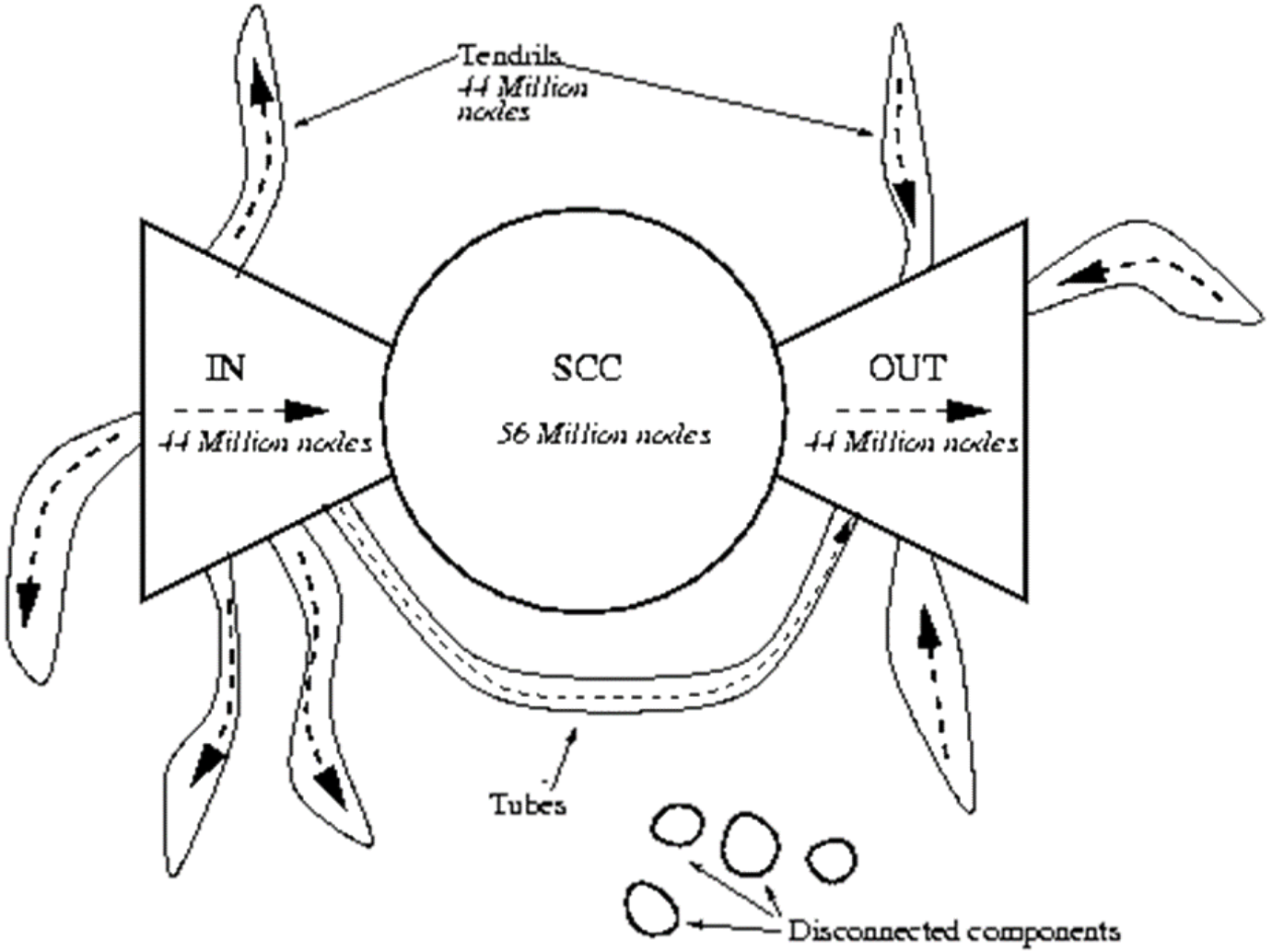
¿Podemos hacer un crawl de toda la Web?

Crawling: All the Web?

¿~~Podemos~~ hacer un crawl de toda la Web?

¿Puede Google hacer un crawl de toda la Web?

Crawling: Inaccesibilidad (Corbatín)



Broder et al. "Graph structure in the web," Comp. Net., vol. 33, no. 1-6, pp. 309–320, 2000

Crawling: Inaccesibilidad ("Deep Web")

¿Qué es la "Deep Web"?

Crawling: Inaccesibilidad ("Deep Web")

¿Qué es la "Deep Web"?

- Contenido generado dinámicamente

Crawling: Inaccesibilidad ("Deep Web")

¿Qué es la "Deep Web"?

- Contenido generado dinámicamente
- Protegido con contraseña

Crawling: Inaccesibilidad ("Deep Web")

¿Qué es la "Deep Web"?

- Contenido generado dinámicamente
- Protegido con contraseña
- "Dark Web" (usa criptografía)

Crawling: Inaccesibilidad ("Deep Web")

¿Qué es la "Deep Web"?

- Contenido gene
- Protegido con c
- "Dark Web" (usa

Se inventan 42% de las estadísticas en el momento

Crawling: All the Web?

¿~~Podemos~~ hacer un crawl de toda la Web?

¿Puede Google hacer un crawl de toda la Web?

¿Puede Google hacer un crawl de si mismo?

Apache Nutch

- Framework open-source para hacer crawling
- Compatible con Hadoop

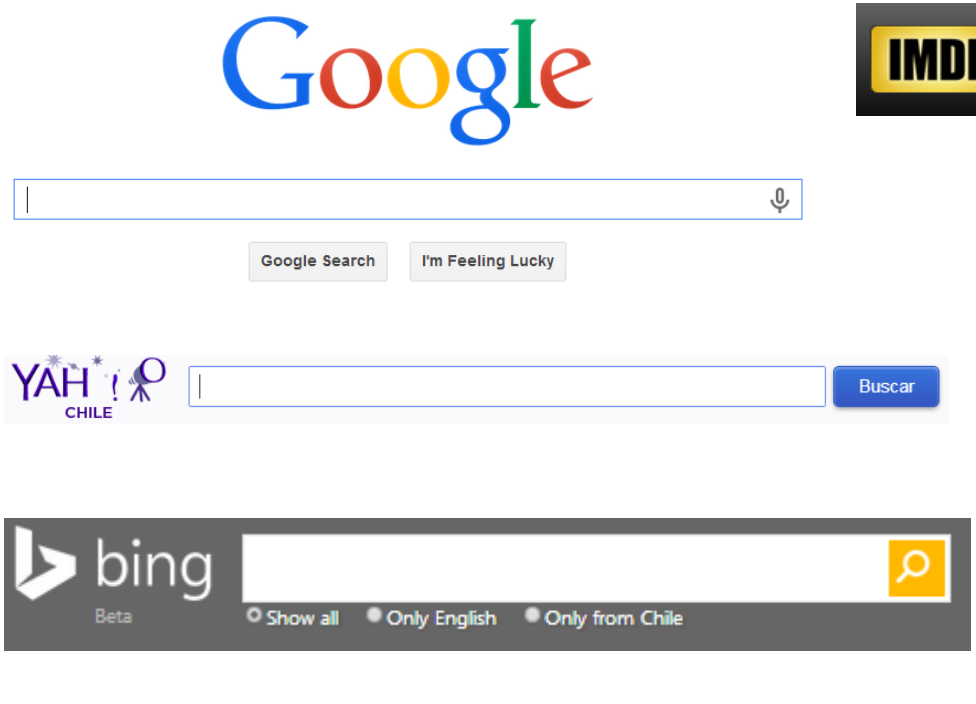
https://nutch.apache.org/

Recuperación de información: Indexación invertida

Índices invertidos

- Un mapa de palabras a documentos
  - "Invertido" pues normalmente documentos mapean a palabras

¿Cuáles son las aplicaciones?



Índices invertidos: Un ejemplo

1

en.wikipedia.org/wiki/Fruitvale_Station

Fruitvale Station

From Wikipedia, the free encyclopedia

Fruitvale Station is a 2013 American drama film written and directed by Ryan Coogler.

Índice invertido:

Term List	Posting List
a	(1,2,…)
american	(1,5,…)
and	(1,2,…)
by	(1,2,…)
directed	(1,2,…)
drama	(1,16,…)
…	…

Índices invertidos: Un ejemplo de búsqueda

american drama

- AND: Intersección de "posting lists"
- OR: Unión de "posting lists"
- PHRASE: ???

¿Cómo deberíamos implementar PHRASE? 

Índice invertido:

Term List	Posting List
a	(1,2,…)
american	(1,5,…)
and	(1,2,…)
by	(1,2,…)
directed	(1,2,…)
drama	(1,16,…)
…	…

Índices invertidos: Frases

1

Fruitvale Station

From Wikipedia, the free encyclopedia

1 10 18 21 23 28 37 43 47 55 59 68 71 76

Fruitvale Station is a 2013 American drama film written and directed by Ryan Coogler.

Índice invertido:

Term List	Posting Lists
a	(1,[21,96,103,…]), (2,[…]), …
american	(1,[28,123]), (5,[…]), …
and	(1,[57,139,…]), (2,[…]), …
by	(1,[70,157,…]), (2,[…]), …
directed	(1,[61,212,…]), (4,[…]), …
drama	(1,[38,87,…]), (16,[…]), …
…	…

Índices invertidos: Niveles de indexación

"Record-level" (nivel de registro)

Mapea palabras a documentos sin sus posiciones en el documento

Term List	Posting List
a	(1,2,…)
american	(1,5,…)
and	(1,2,…)
by	(1,2,…)
directed	(1,2,…)
drama	(1,16,…)
…	…

"Word-level" (nivel de palabra)

Mapea palabras a documentos con sus posiciones en el documento

Term List	Posting Lists
a	(1,[21,96,103,…]), (2,[…]), …
american	(1,[28,123]), (5,[…]), …
and	(1,[57,139,…]), (2,[…]), …
by	(1,[70,157,…]), (2,[…]), …
directed	(1,[61,212,…]), (4,[…]), …
drama	(1,[38,87,…]), (16,[…]), …
…	…

Índices invertidos: Normalización de palabras

drama america

¿Cómo podemos resolver este problema?

Índice invertido:

Term List	Posting Lists
a	(1,[21,96,103,…]), (2,[…]), …
american	(1,[28,123]), (5,[…]), …
and	(1,[57,139,…]), (2,[…]), …
by	(1,[70,157,…]), (2,[…]), …
directed	(1,[61,212,…]), (4,[…]), …
drama	(1,[38,87,…]), (16,[…]), …
…	…

Índices invertidos: Normalización de palabras

drama america

¿Cómo podemos resolver este problema?

Normalizar palabras:

"Stemming" corta los sufijos de las palabras según reglas genéricas:

`{ America , American , americas , americanise } → { america }`

Índices invertidos: Normalización de palabras

drama america

¿Cómo podemos resolver este problema?

Normalizar palabras:

"Stemming" corta los sufijos de las palabras según reglas genéricas:

`{ America , American , americas , americanise } → { america }`

"Lematización" usa conocimiento de la palabra para normalizarla

`{ better , goodly , best } → { good }`

Índices invertidos: Normalización de palabras

drama america

¿Cómo podemos resolver este problema?

Normalizar palabras:

"Stemming" corta los sufijos de las palabras según reglas genéricas:

{ America , American , americas , americanise } → { america }

"Lematización" usa conocimiento de la palabra para normalizarla

{ better , goodly , best } → { good }

Reemplazar palabras sinónimas

{ film , movie } → { movie }

- La normalización es específica al lenguaje
- Hay que usar la misma normalización para el documento y la consulta

Índices Invertidos: Distribución

Índices Invertidos: Distribución

Term	Posting
and	1,3,4,5,6
ate	1,2,3
cat	3,4,6
dog	3,5,6,7
the	1,2,3,4,5,6,7
vet	4

¿Cómo deberíamos distribuir un índice invertido?

Distribuir por palabra

Term	Posting
dog	3,5,6,7

Term	Posting
and	1,3,4,5,6
vet	4

Term	Posting
ate	1,2,3
the	1,2,3,4,5,6,7

Term	Posting
cat	3,4,6

¿Desventajas?

- Complicaciones para balancear la carga considerando palábras comunes
- Con AND y PHRASE habrá que hacer intersecciones entre varias máquinas
- Puede ser más difícil de indexar y actualizar

Índices Invertidos: Distribución

Term	Posting
and	1,3,4,5,6
ate	1,2,3
cat	3,4,6
dog	3,5,6,7
the	1,2,3,4,5,6,7
vet	4

¿Cómo deberíamos distribuir un índice invertido?

Distribuir por documento

Term	Posting
ate	2
cat	6
dog	6
the	2,6

Term	Posting
and	3,5
ate	3
cat	3
dog	5
the	3,5

Term	Posting
and	1
ate	1
the	1

Term	Posting
and	4
dog	7
the	7
vet	4

¿Desventajas?

- Búsquedas con palabras simples van a necesitar uniones sobre varias máquinas
- Puede haber problemas con balancear la carga si hay documentos muy grandes
- Hay que replicar términos en varias máquinas

Índices invertidos: Implementaciones

Apache Lucene

- Índice invertido
  - Open Source
  - Java

Doug Cutting (arriba) & Mike Cafarella (abajo)

lucene™

nutch

hadoop

Lucene, Solr y Elasticsearch

- Índice invertido
- Open source (Apache)
- Java
- Una máquina

Apache Solr

- Lucene + HTTP API
- Open source (Apache)
- Java
- Una máquina*
- Algunas extensiones
- Mejor para datos estáticos

elasticsearch

- Lucene + HTTP API
- Open source (Elastic)
- Java
- Varias máquinas
- Muchas extensiones
- Mejor para datos dinámicos

* Existe una extensión que se llama SolrCloud, que ofrece distribucion

ELASTICSEARCH

Documentos

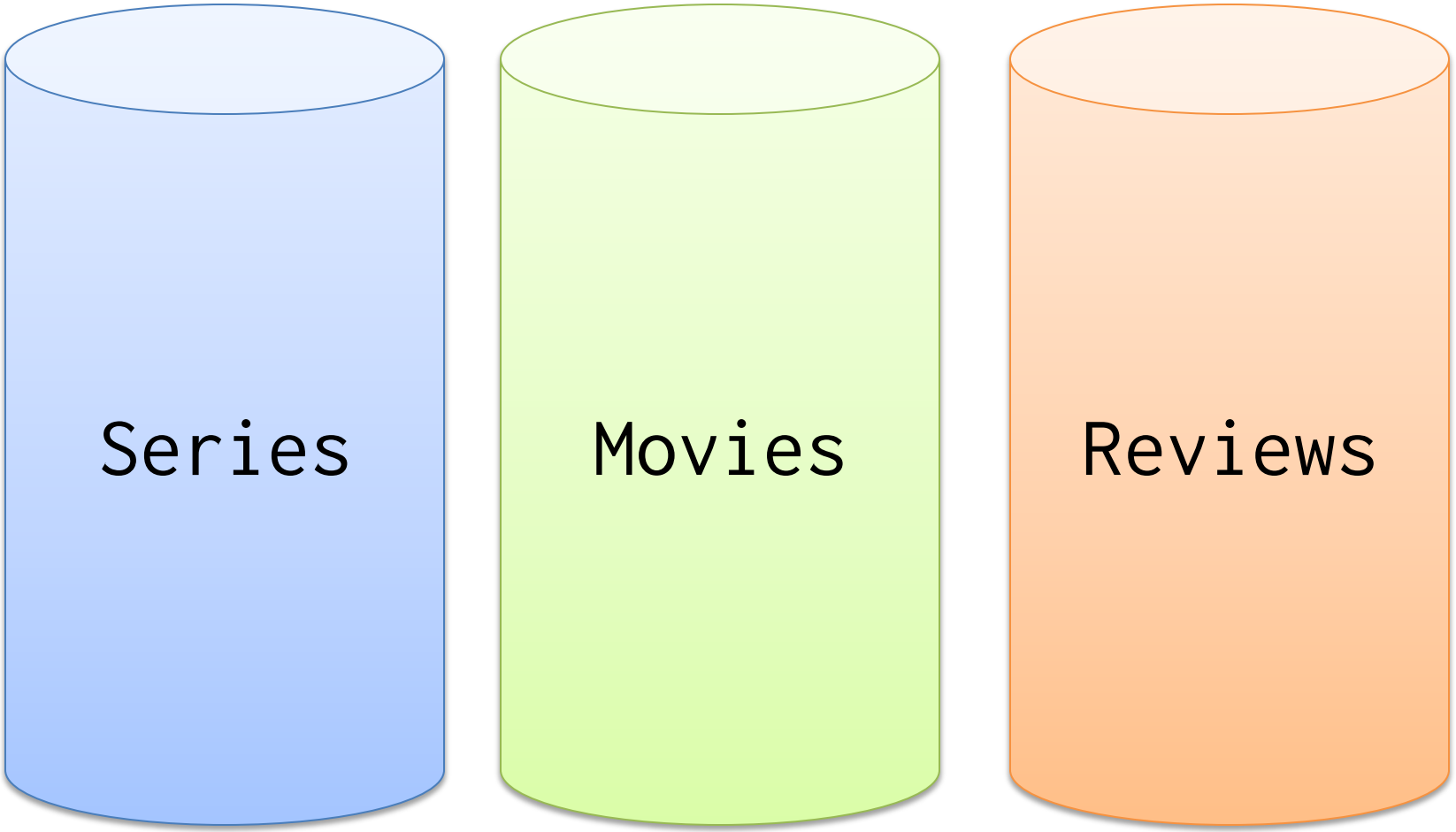
```
{
  "name": "Dark",
  "description": "A crime thriller with elements of
      science-fiction set in a fictional German town.",
  "language": "German",
  "genres": [ "Science Fiction", "Thriller", "Crime" ],
  "start": "2017-12-01",
  "episodes": 26,
  "seasons": [
    {
      "num": 1,
      "epsiodes": 10
    },
    {
      "num": 2,
      "epsiodes": 8
    },
    {
      "num": 3,
      "epsiodes": 8
    }
  ],
  "rating": {
    "average": 8.8,
    "count": 12924
  }
}
```

Tipos

```
{
  "mappings": {
    "series" : {
      "properties": {
        "name": {
           "type": "text",
           "fields": {
             "raw": {
               "type": "keyword"
             }
           }
        },
        "description": { "type": "text" },
        "language": { "type": "keyword" },
        "genres": { "type": "keyword" },
        "start": { "type": "date" },
        "episodes": { "type": "integer" },
        "seasons": { "type": "nested" },
        "rating": { "type": "object" }
      }
    }
  }
}
```

... definen un esquema para un tipo de documento

Índices



... cada índice indexa un tipo de documento

Elasticsearch: Distribución

Distribución por documento

Term	Posting
and	1,3,4,5,6
ate	1,2,3
cat	3,4,6
dog	3,5,6,7
the	1,2,3,4,5,6,7
vet	4

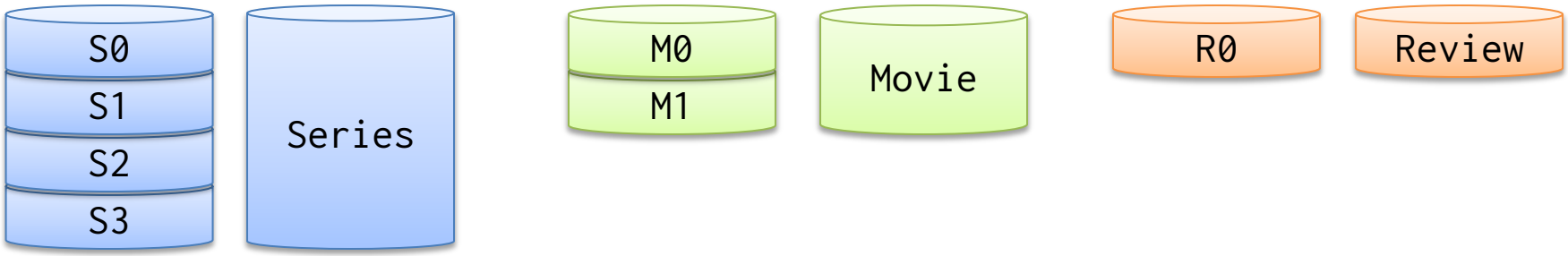
Term	Posting
ate	2
cat	6
dog	6
the	2,6

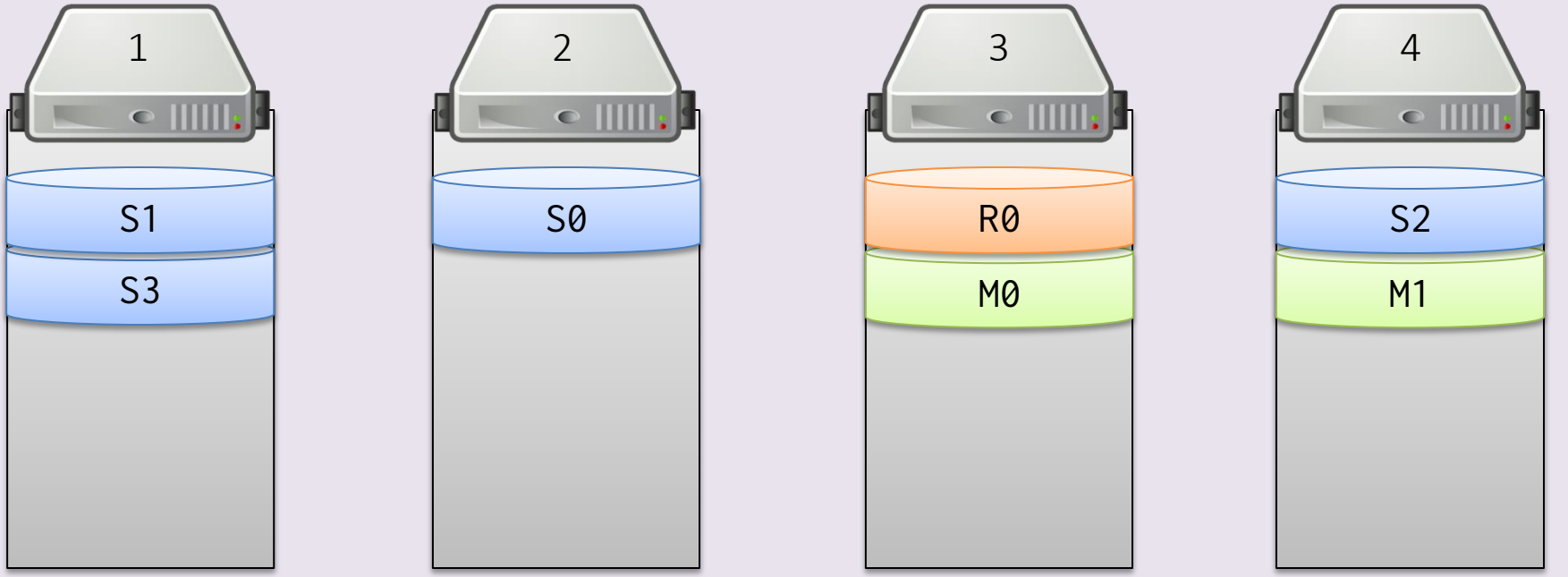
Term	Posting
and	3,5
ate	3
cat	3
dog	5
the	3,5

Term	Posting
and	1
ate	1
the	1

Term	Posting
and	4
dog	7
the	7
vet	4

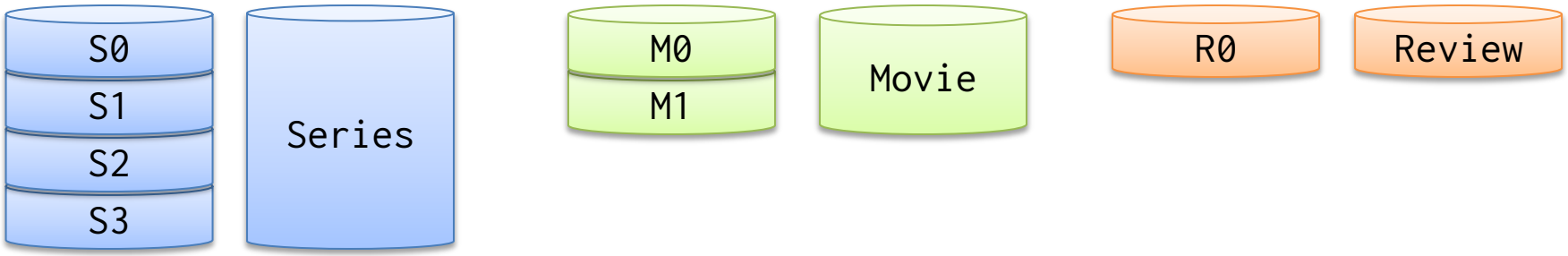
Shards

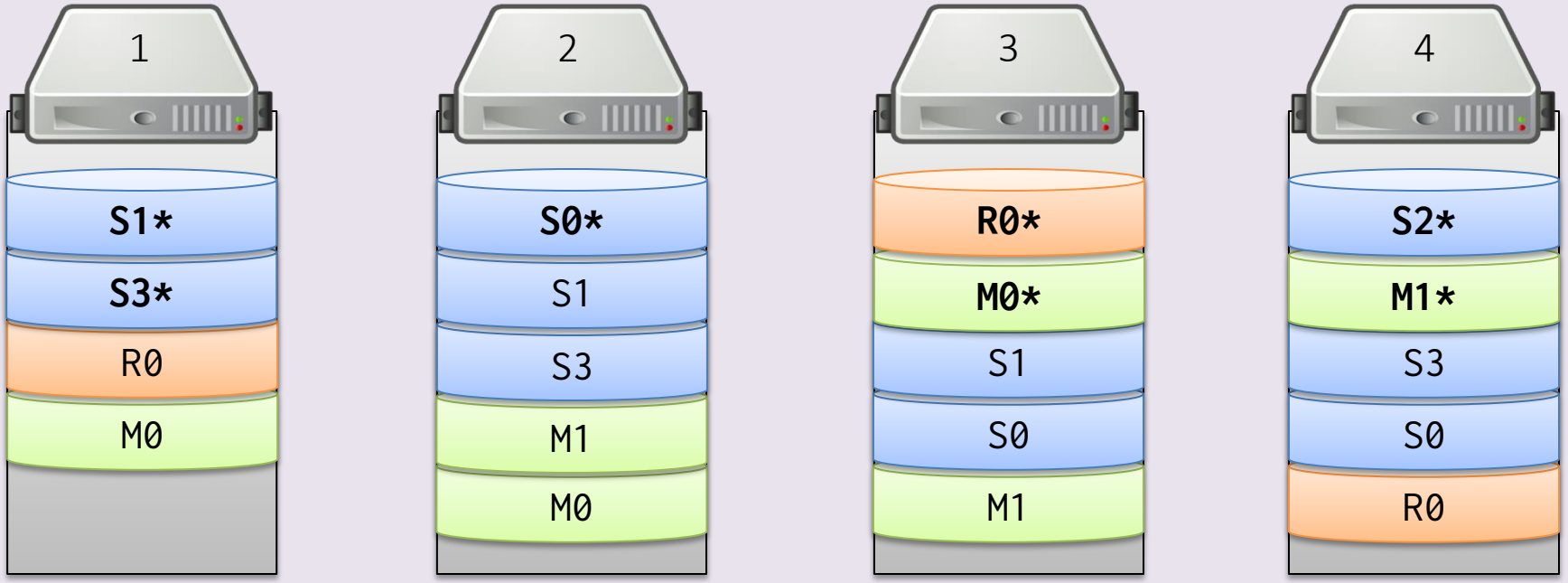




… un maestro asigna los shards (y las replicas) a las máquinas

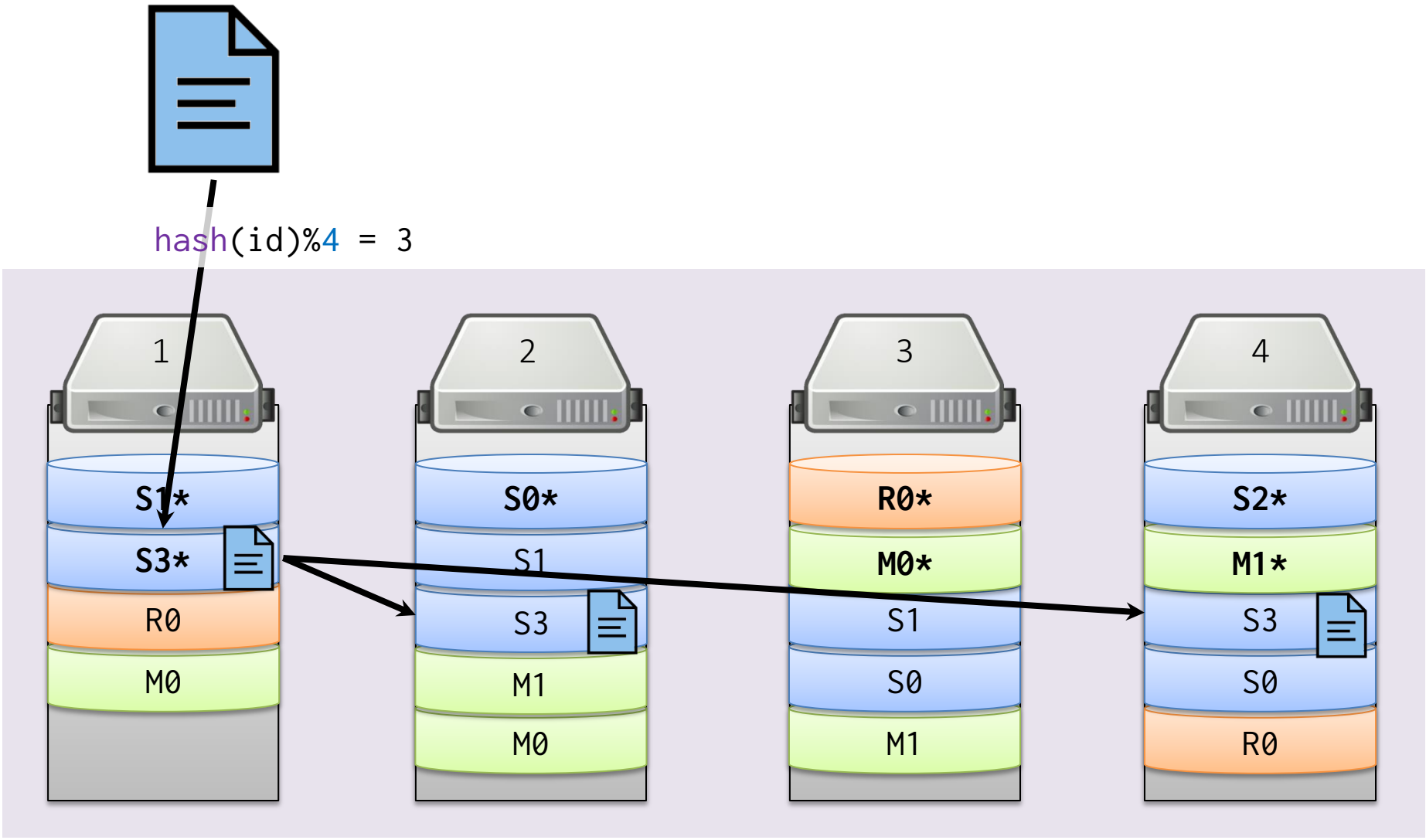
Shards **Primario*** / Replica (2)



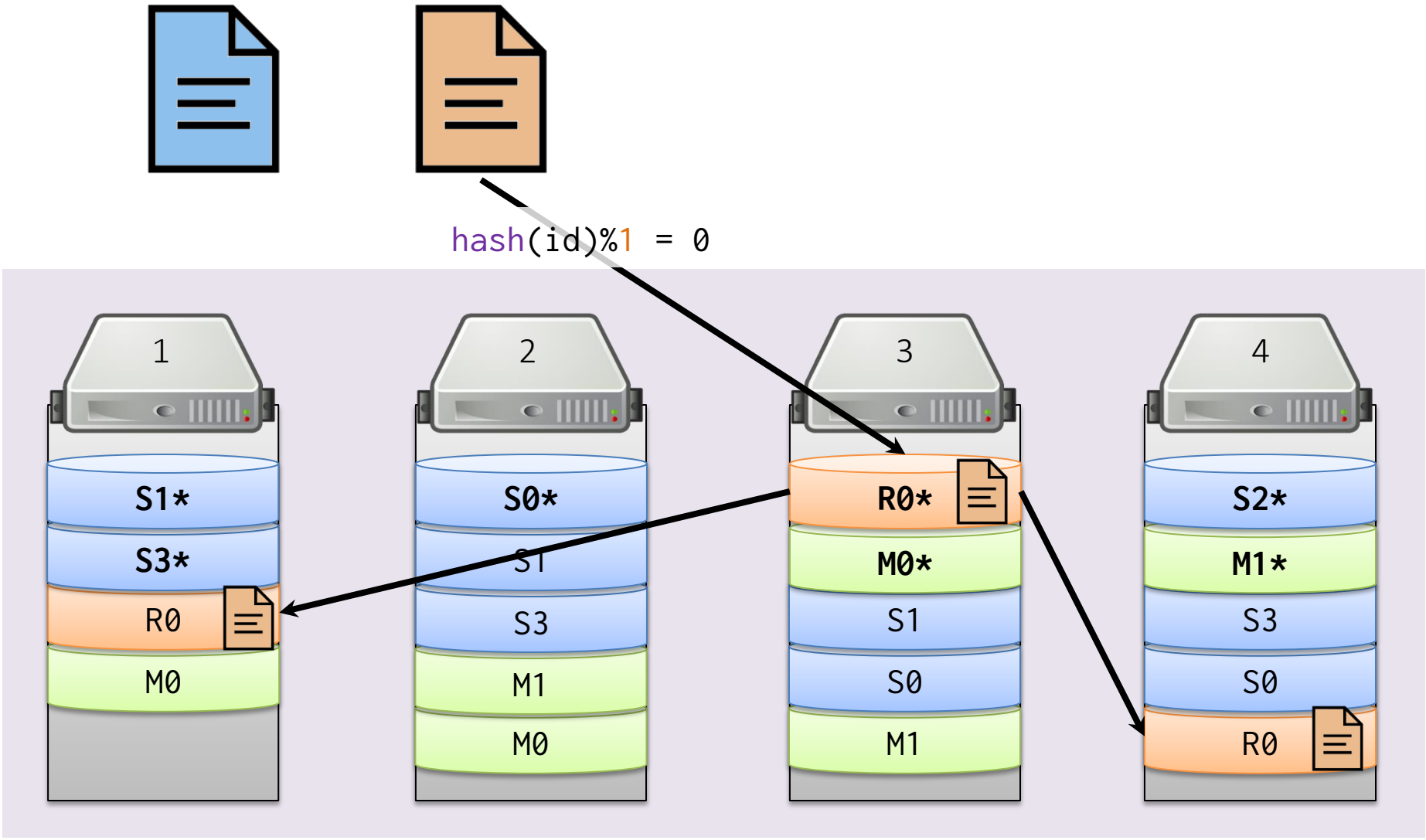


… en caso de fallos, replicará los datos y elegirá otro primario

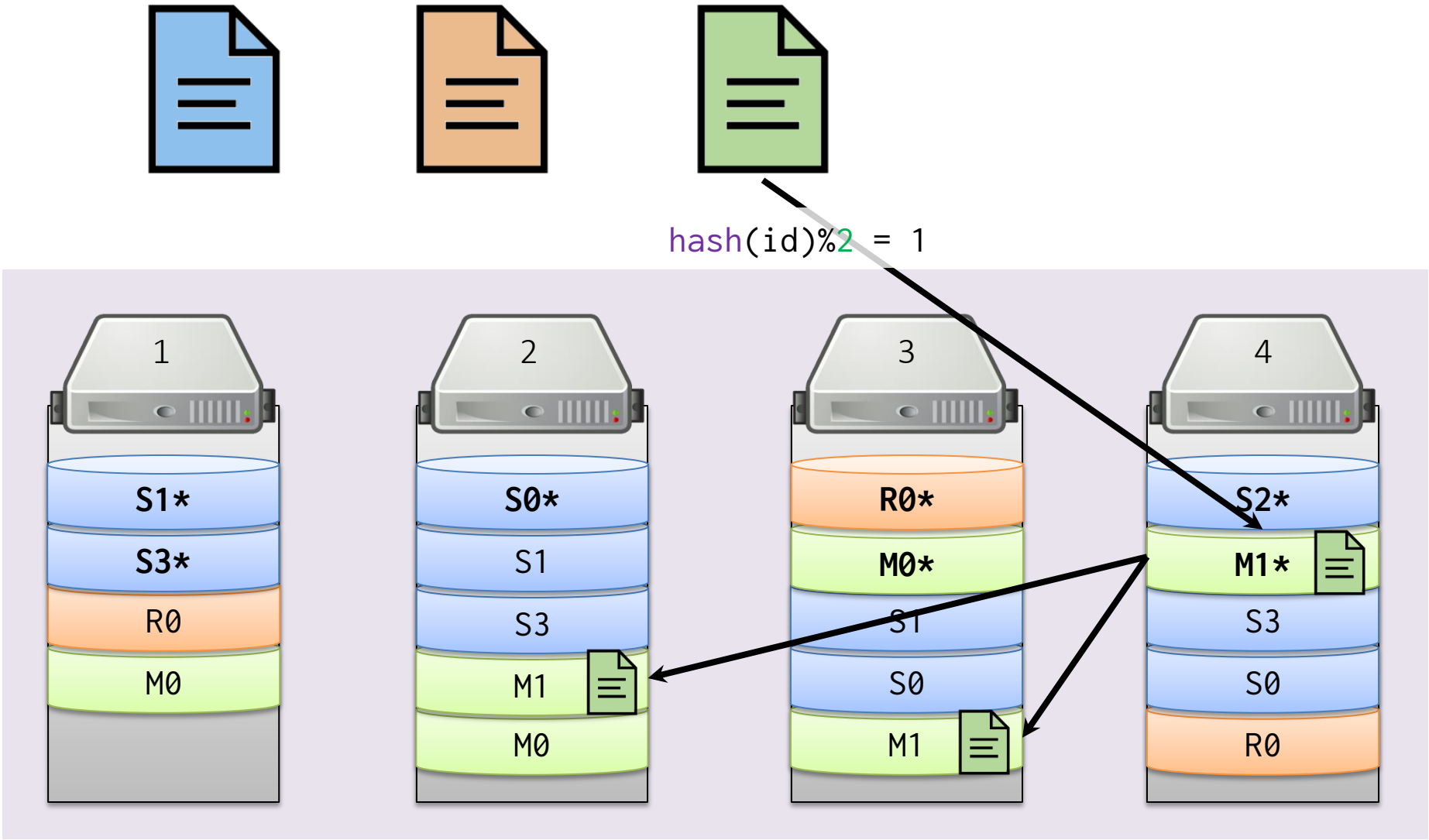
Shards: Escrituras



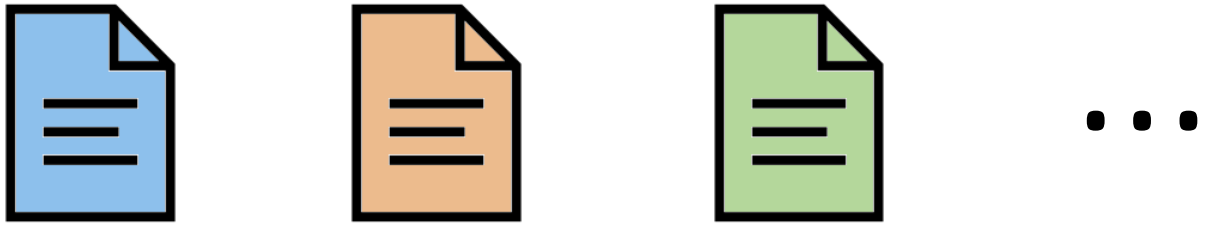
Shards: Escrituras

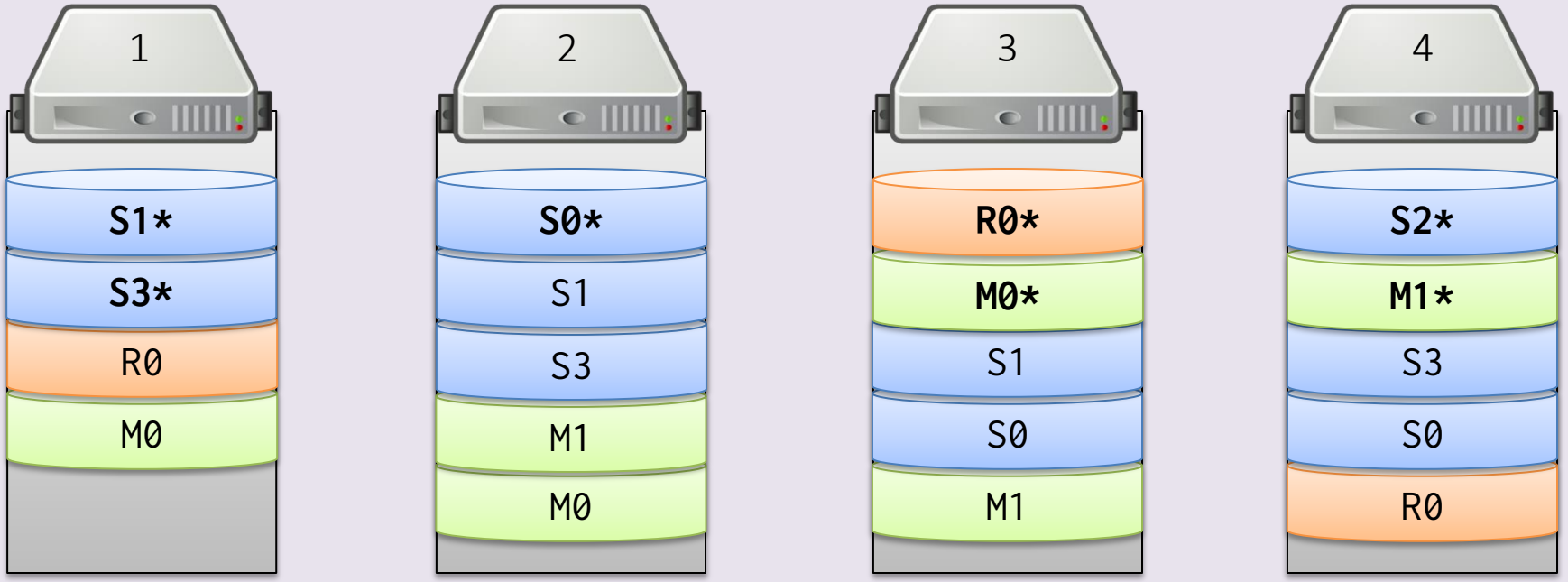


Shards: Escrituras



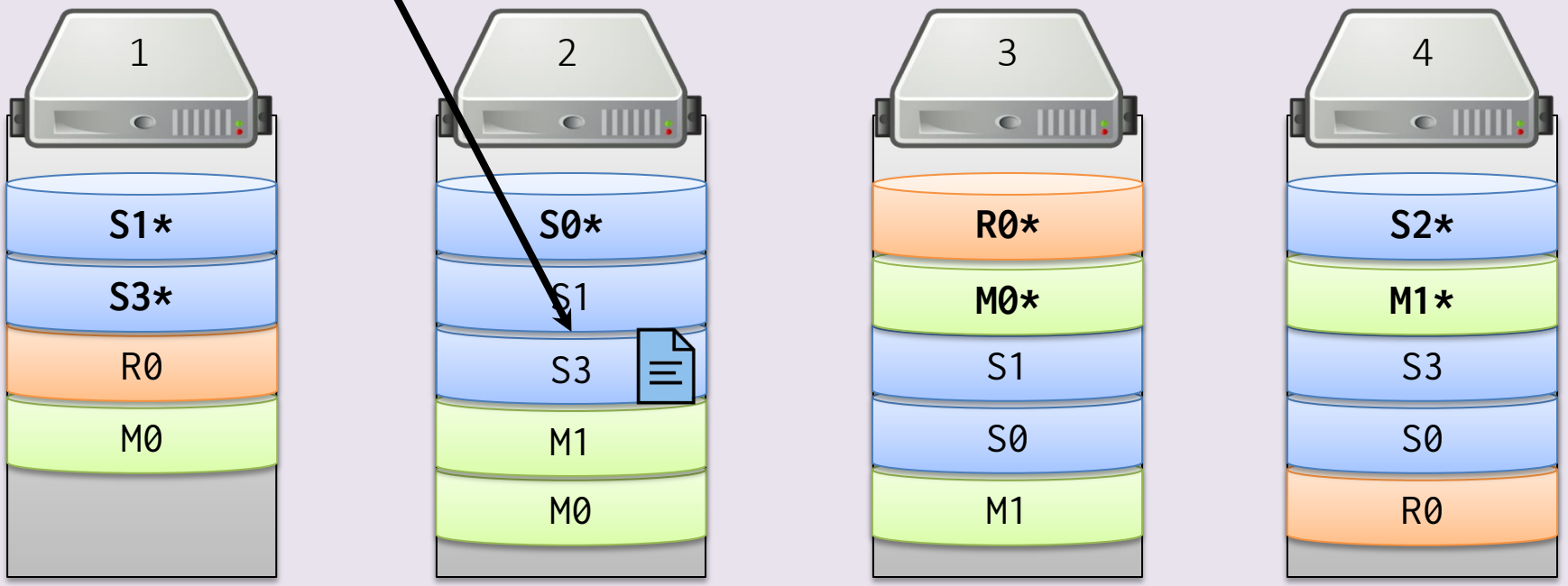
Shards: Escrituras





Shards: Lecturas (ID)

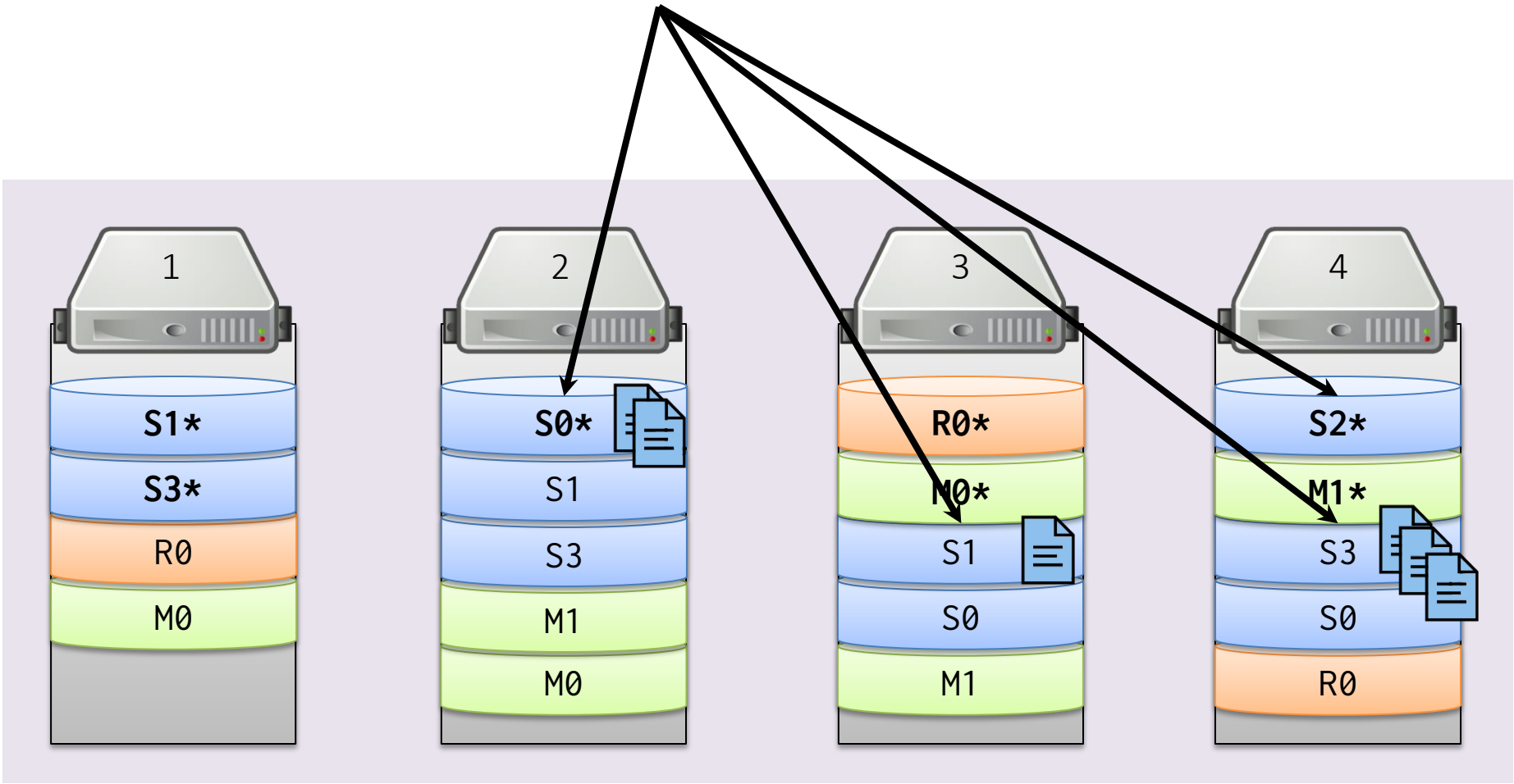
```
[Series] id: 1332

  hash(id)%4 = 3
```



... lee de cualquier replica (o primaria) del shard 3

Shards: Lecturas (Búsqueda)

```
[Series] description: “german science fiction”
```



... busca en cualquier replica (o primaria) de cada shard

Elasticsearch: Interfaces

Interfaz: HTTP/RESTful

- Documento:
  - `INDEX GET DELETE UPDATE`, p.ej.:
    - `GET Series/_doc/54`
    - `PUT Series/_doc/42` , etc.

- Multi-documento:
  - `MULTI-GET BULK DELETE-BY-QUERY UPDATE-BY-QUERY REINDEX` , p.ej.:
    - `GET /_mget`

```
{ “docs” [
   { “_index”: “Series”, “_id”: “42” },
   { “_index”: “Series”, “_id”: “54” }
] }
```

https://www.elastic.co/guide/en/elasticsearch/reference/current/docs.html

Interfaz: HTTP/RESTful

- Índice:
  - `CREATE-INDEX UPDATE-INDEX GET-INDEX`, p.ej.:
    - `PUT /VideoGame`
    - `DELETE /VideoGame`, etc.

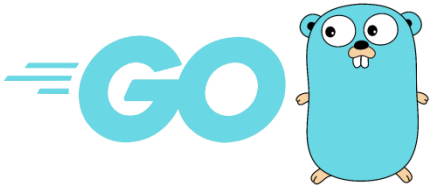
- Búsqueda:
  - `SEARCH`, p.ej.:
    - `GET /Series/_search`

```
{ “query” [
   { “term” : { “description”: “nuclear”  } }
] } , etc.
```

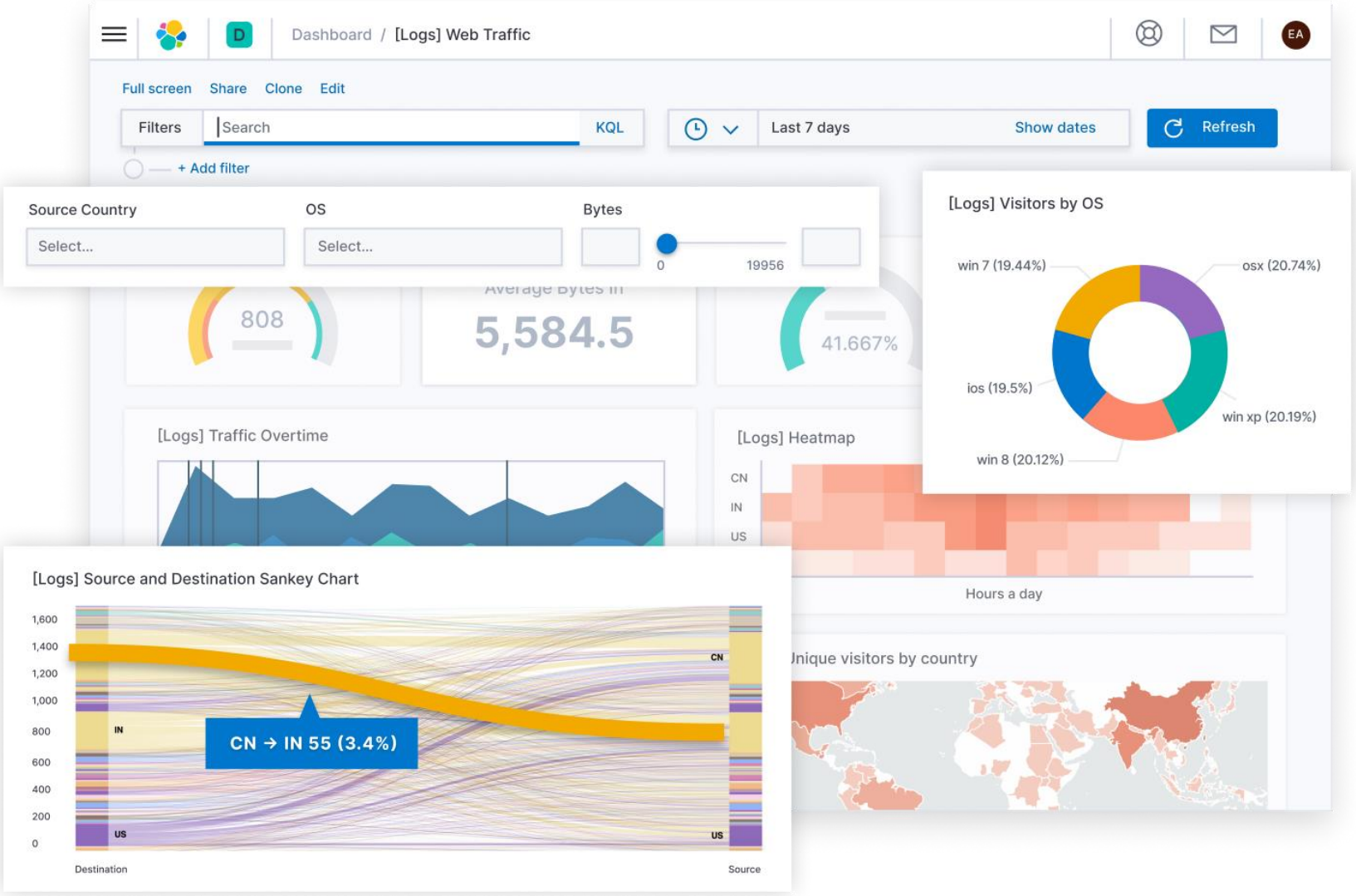
. . .

https://www.elastic.co/guide/en/elasticsearch/reference/current/docs.html

Interfaz: Programática

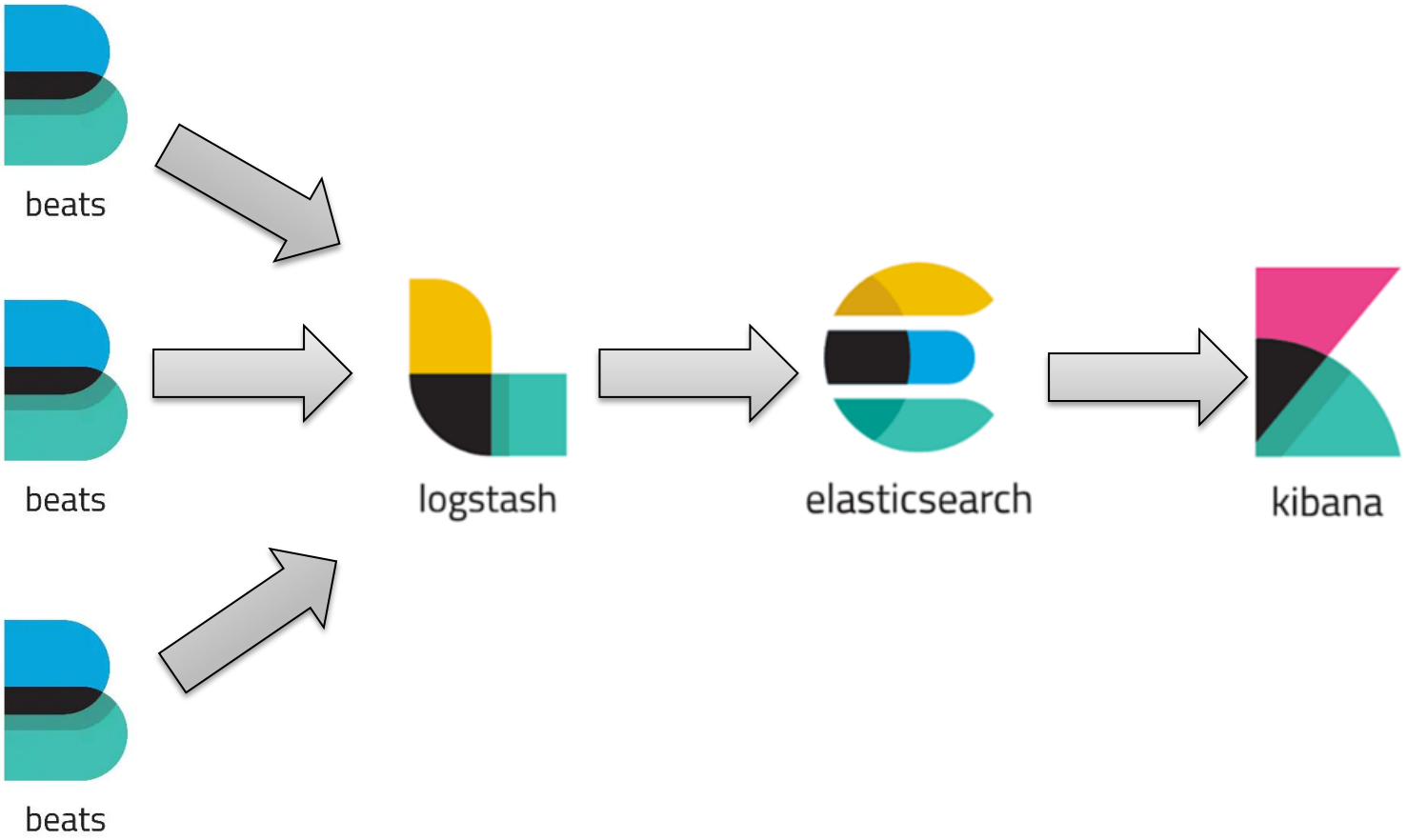


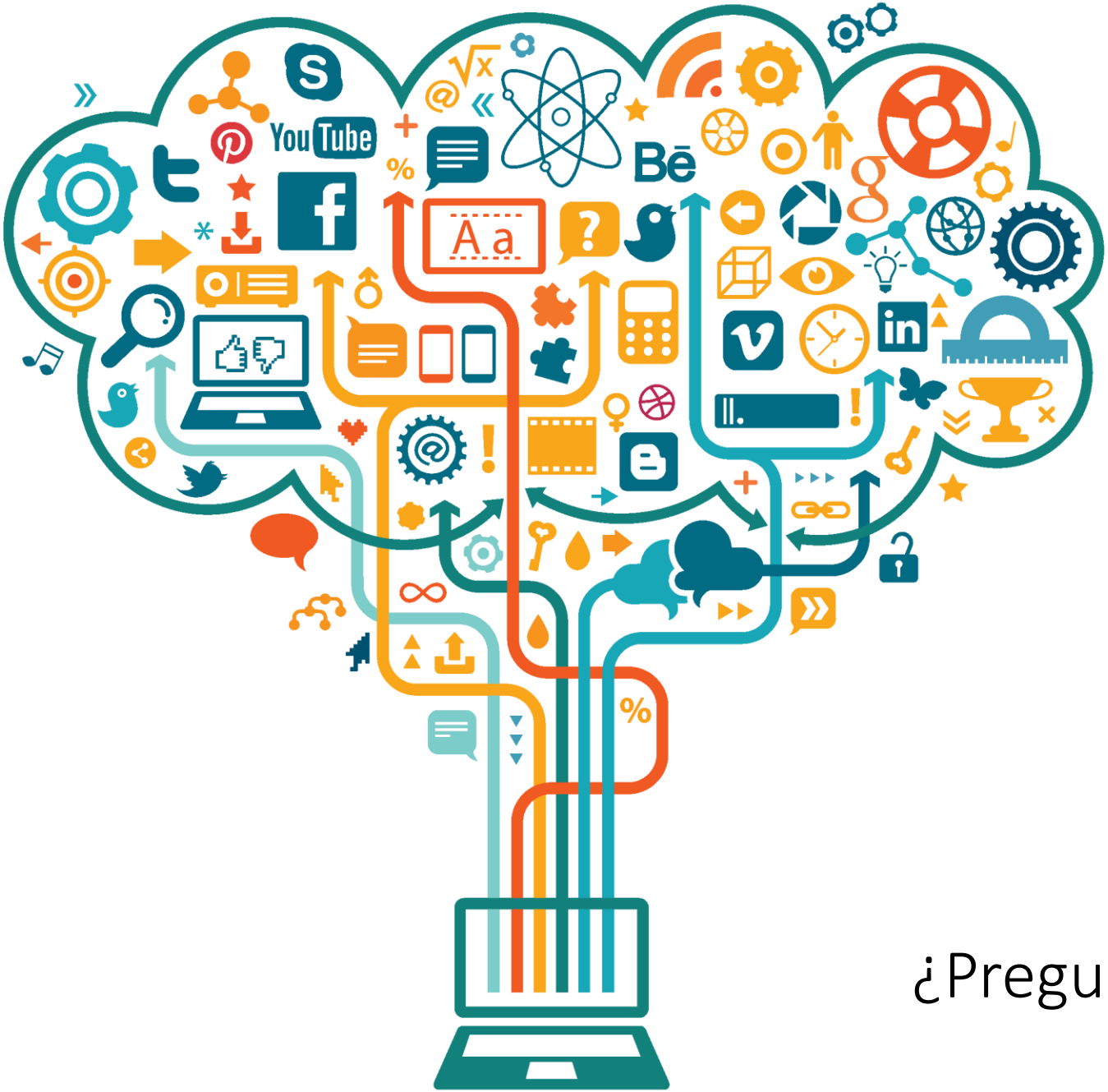
Interfaces: Análitica (Kibana)



ELK Framework: (ElasticSearch+LogShash+Kibana)

- Para procesar y analizar “streams” de datos, por ejemplo, mensajes de log





¿Preguntas?