Bases de Datos

Clase 14: NoSQL

Hasta ahora

- Bases de datos relacionales
- SQL

Bases de datos relacionales

- Muchas estructura (un esquema fijo)
- Muchas garantías (ACID)
- Generalmente centralizadas (viven en un servidor)

NoSQL

Término común para denominar bases de datos con:

- Menos restricciones que el modelo relacional
- Menos esquema
- Menos garantías de consistencia
- Más distribución

NoSQL: ¿Por qué?

Sistemas de bases de datos relacionarles no están pensadas para un entorno altamente distribuido

- WWW, google, twitter, instagram, et

Sistemas distribuidos

Dos problemas fundamentales:

1. Datos no caben en un computador
2. Servidores pueden fallar

Datos no caben en un computador

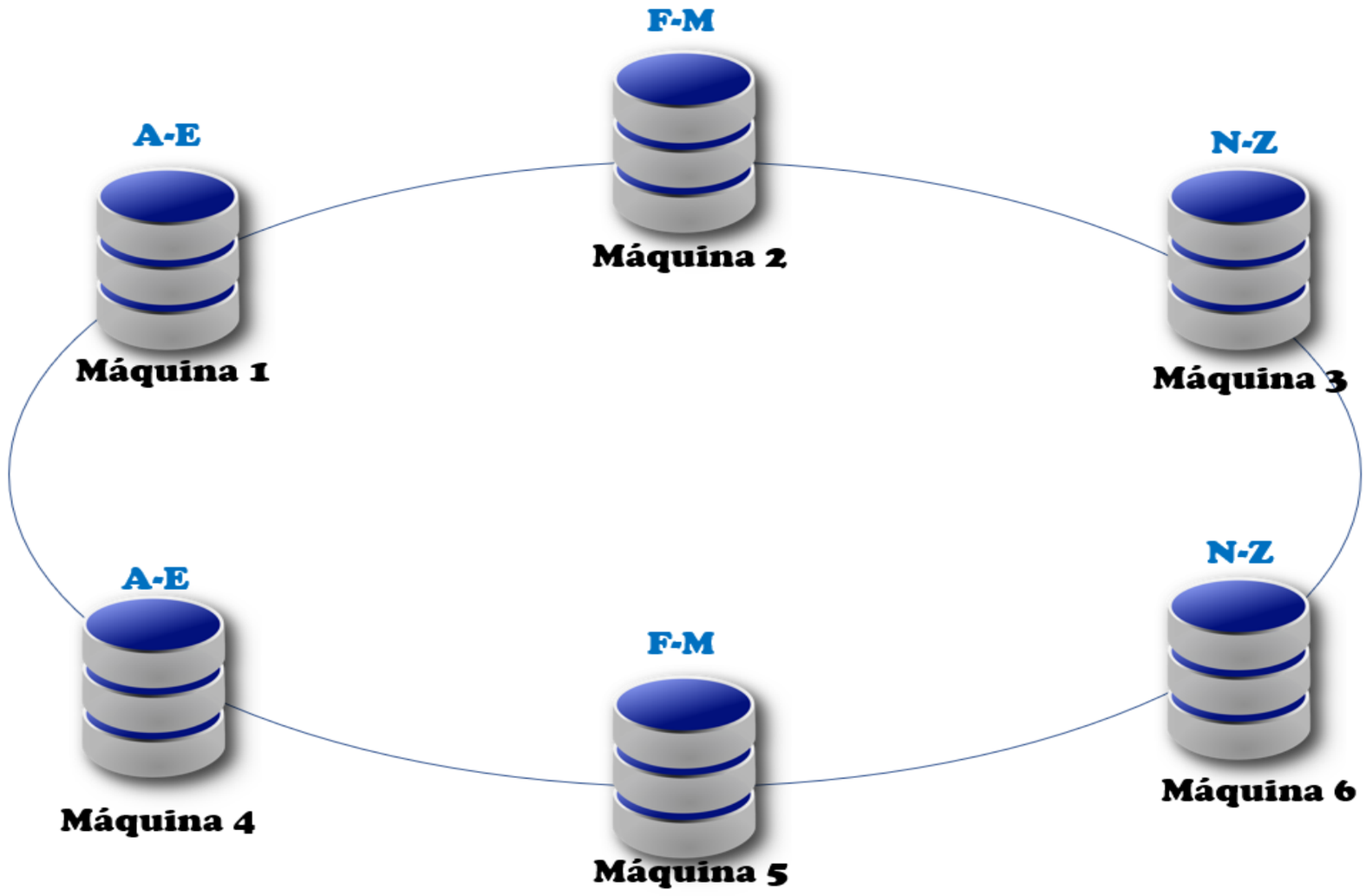
Fragmentación de los datos

Ej: **Usuarios** de twitter

Fragmentación de relación **Usuarios** en tres

Servidores fallan

Replicación de los datos



Replicación en un sistema distribuido

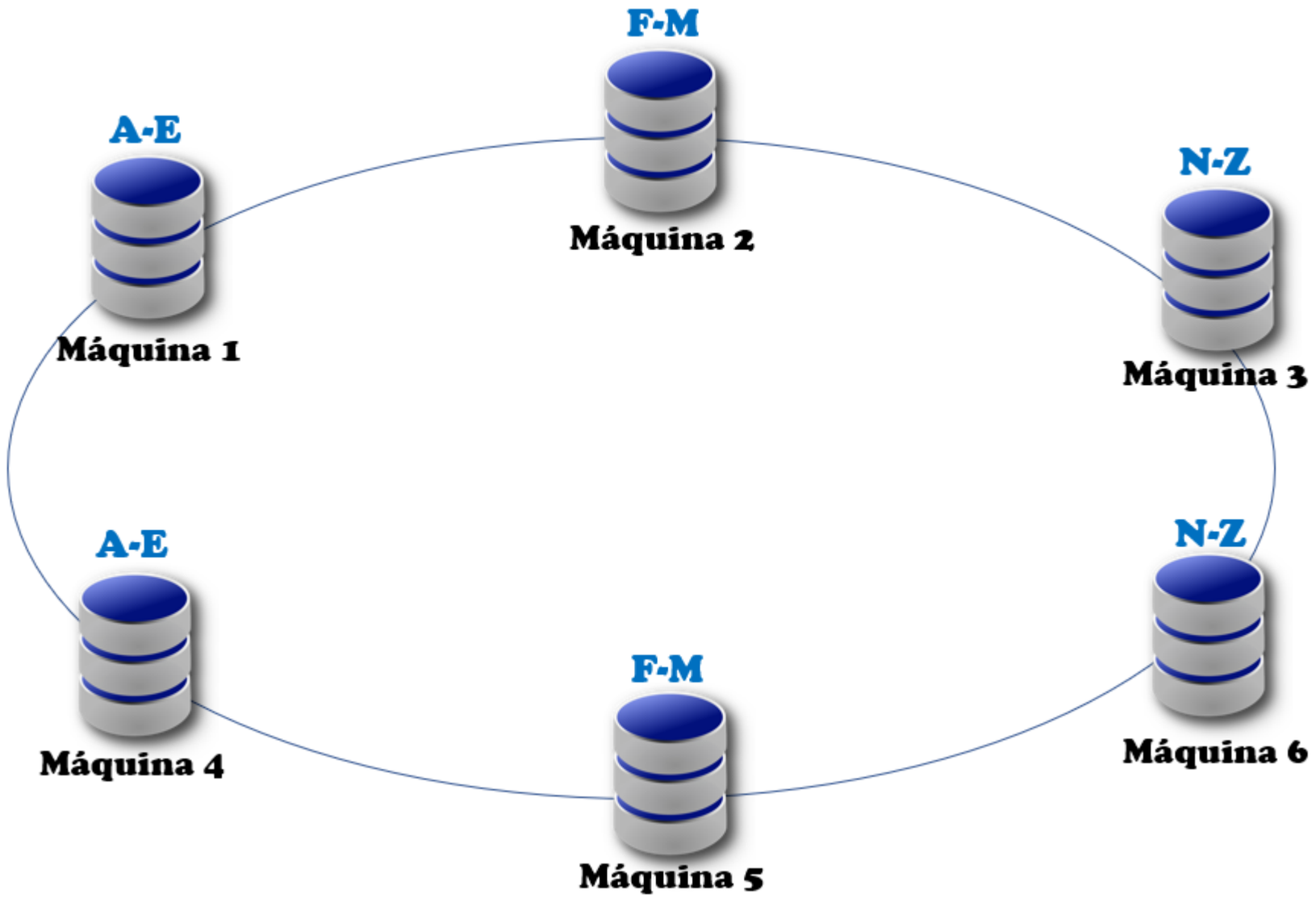
Garantías en un entorno distribuido

Tres propiedades fundamentales:

- Consistency (todos los usuarios ven lo mismo)
- Availablity (todas las consultas siempre reciben una respuesta, aunque sea error)
- Partition tolerance (el sistema funciona bien pese a estar físicamente dividido)

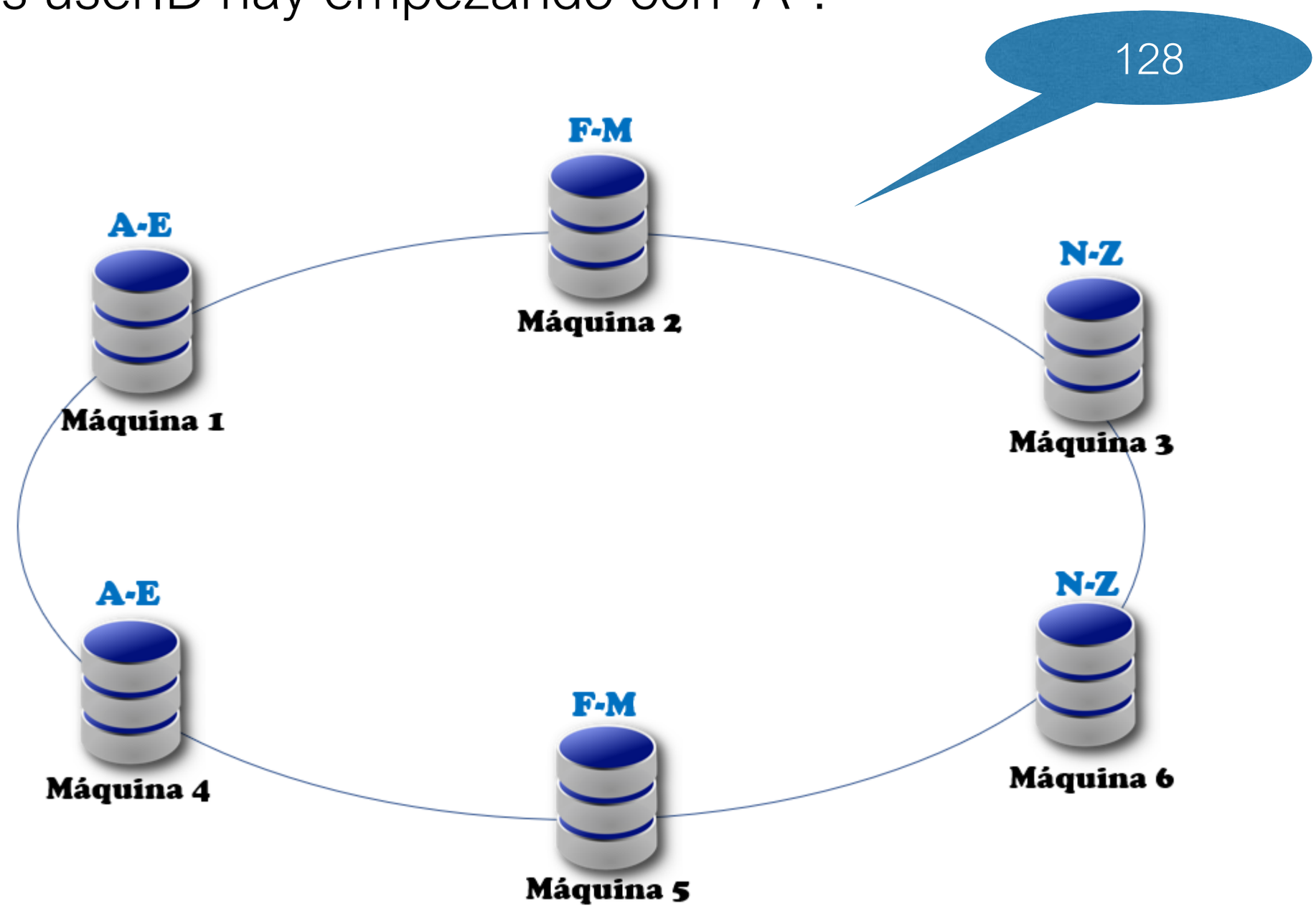
Consistencia

¿Cuántos userID hay empezando con "A"?



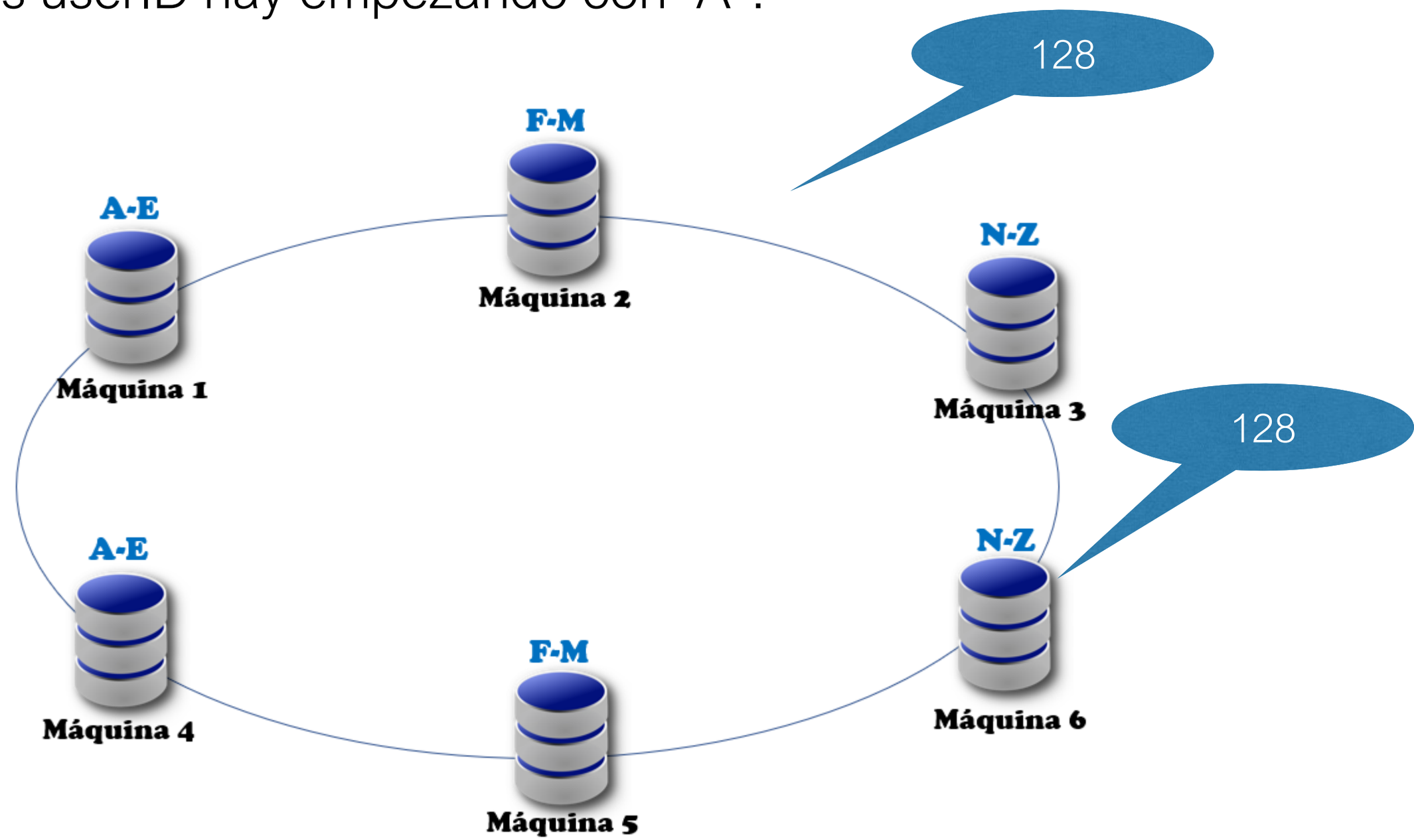
Consistencia

¿Cuántos userID hay empezando con "A"?



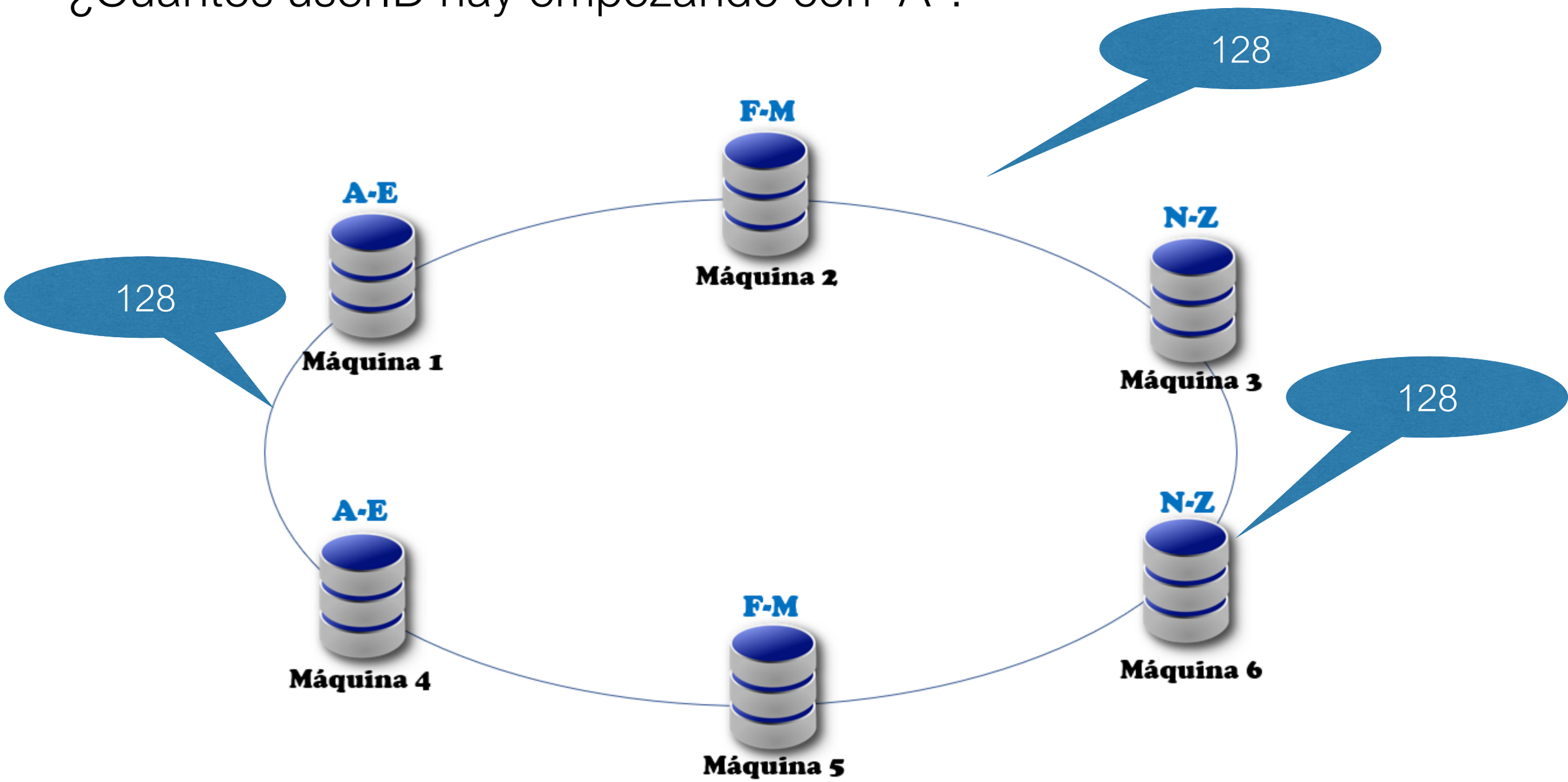
Consistencia

¿Cuántos userID hay empezando con "A"?



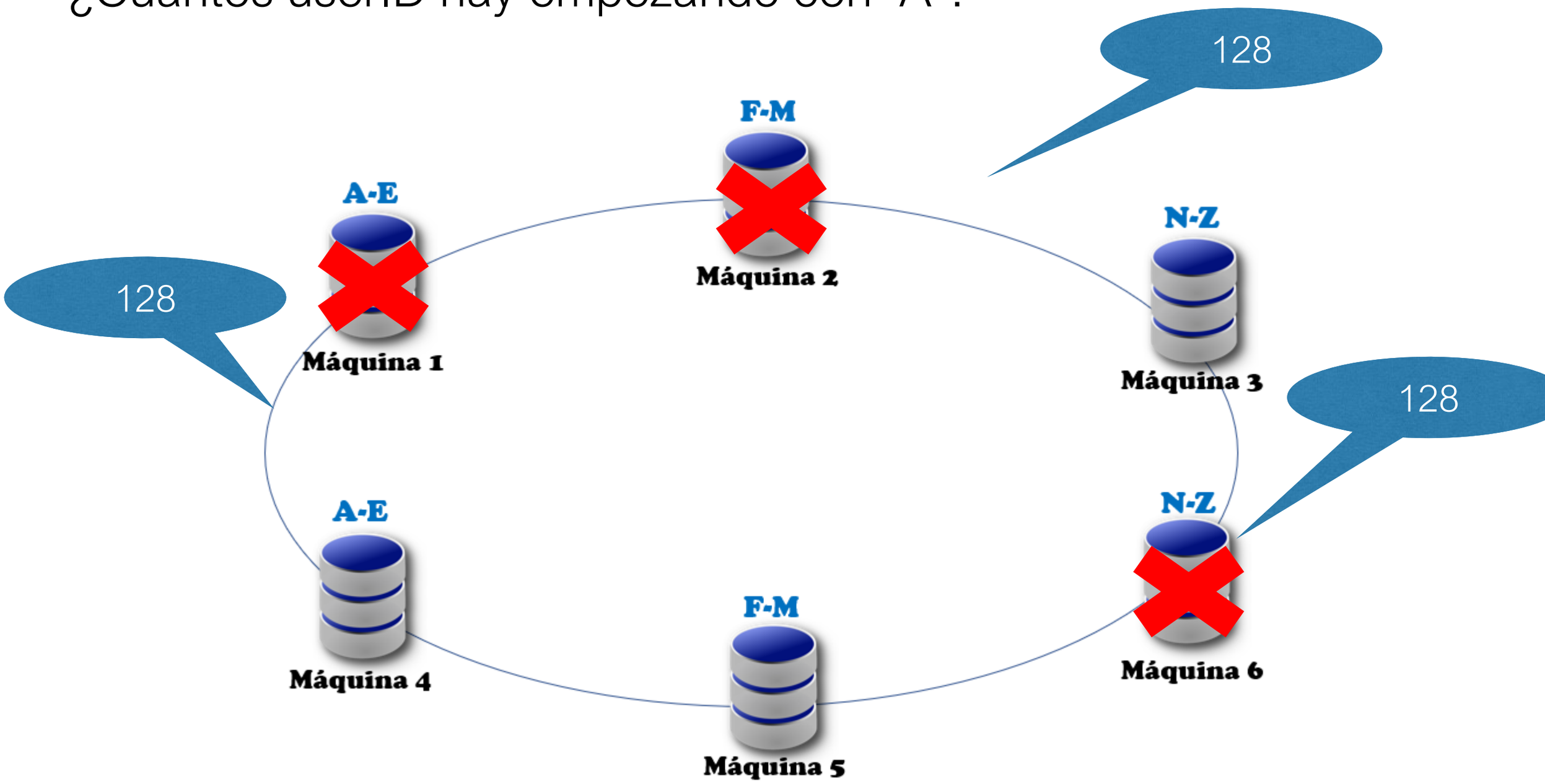
Consistencia

¿Cuántos userID hay empezando con "A"?



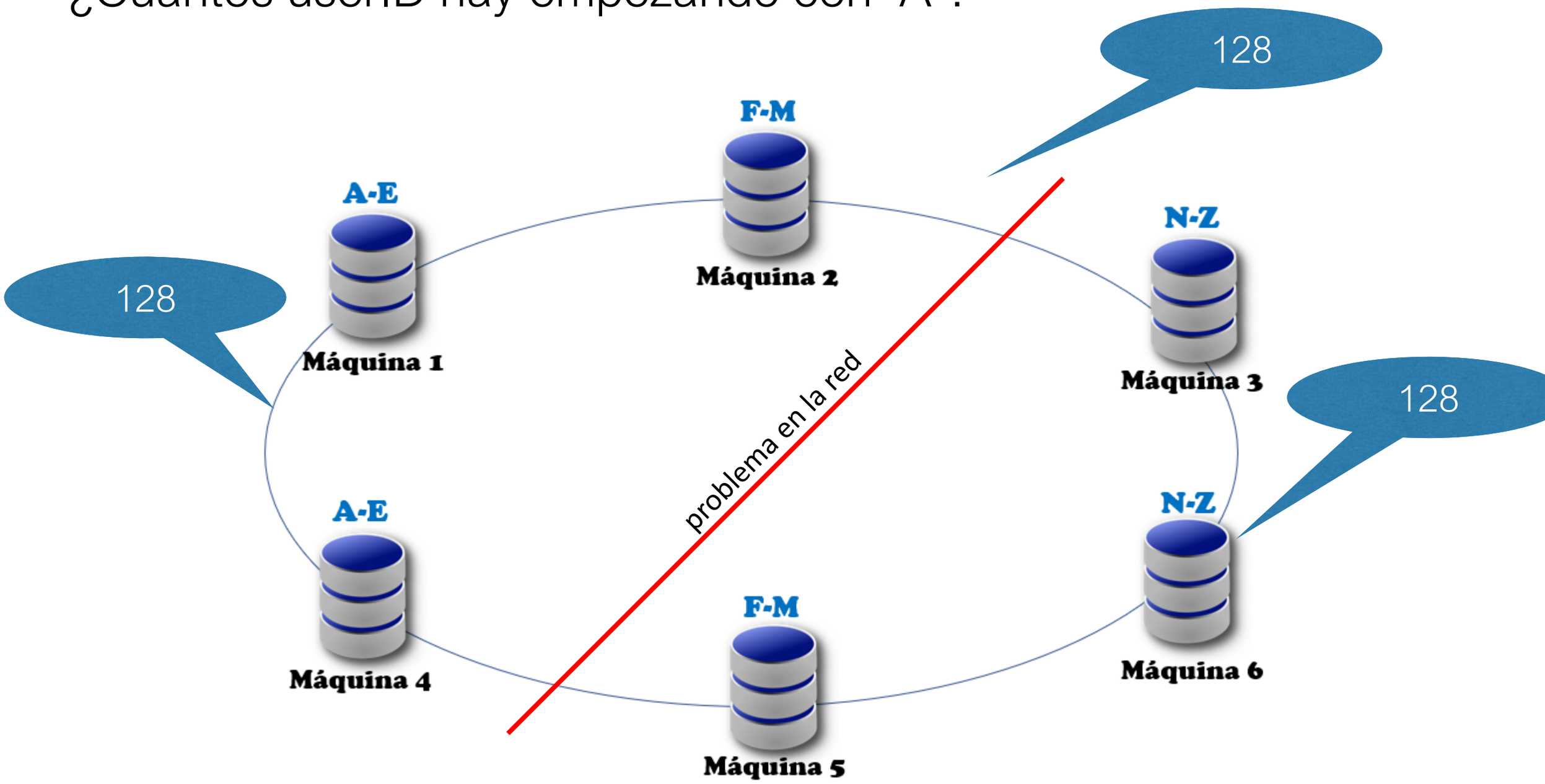
Consistencia

¿Cuántos userID hay empezando con "A"?



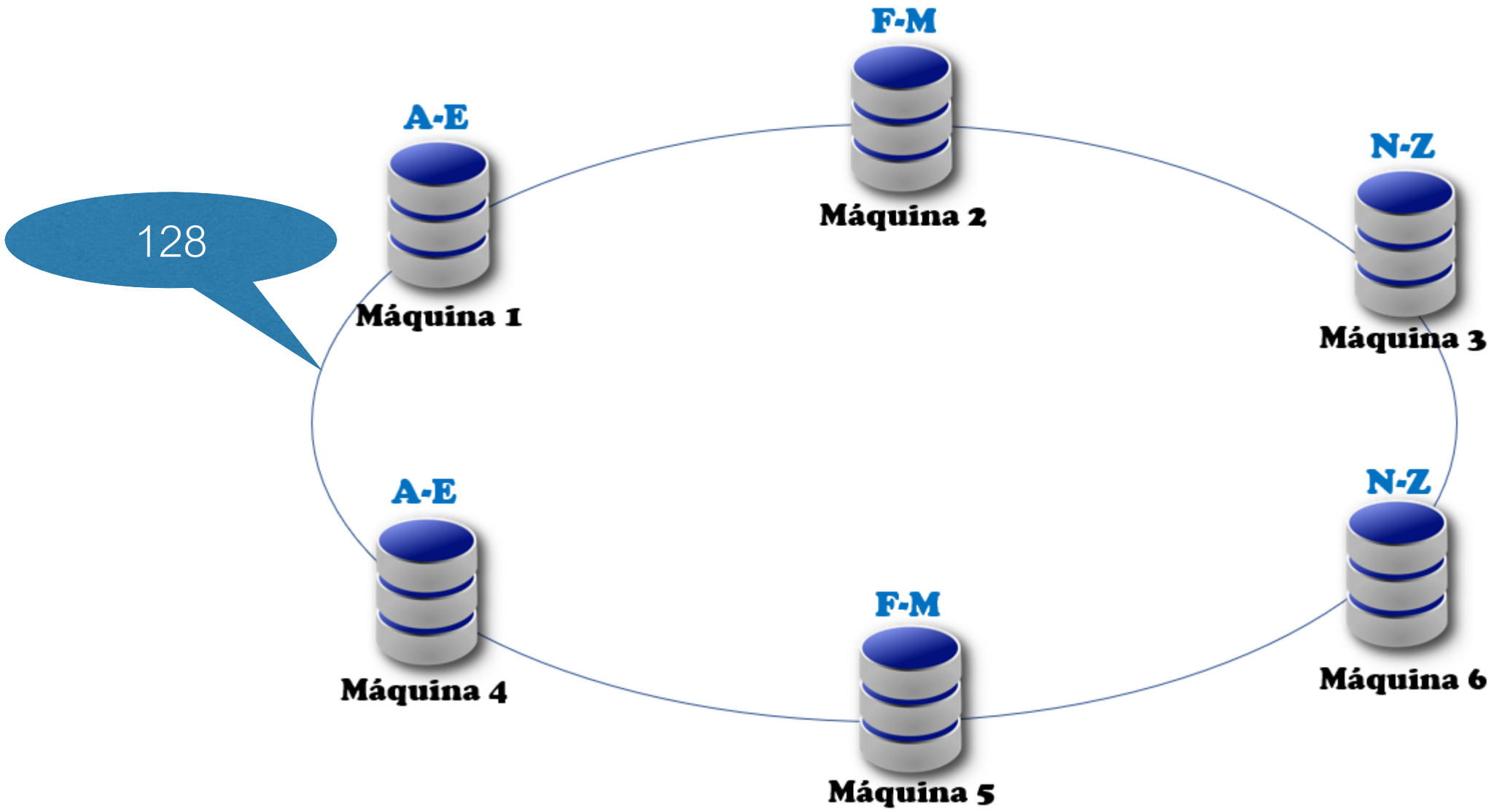
Consistencia

¿Cuántos userID hay empezando con "A"?



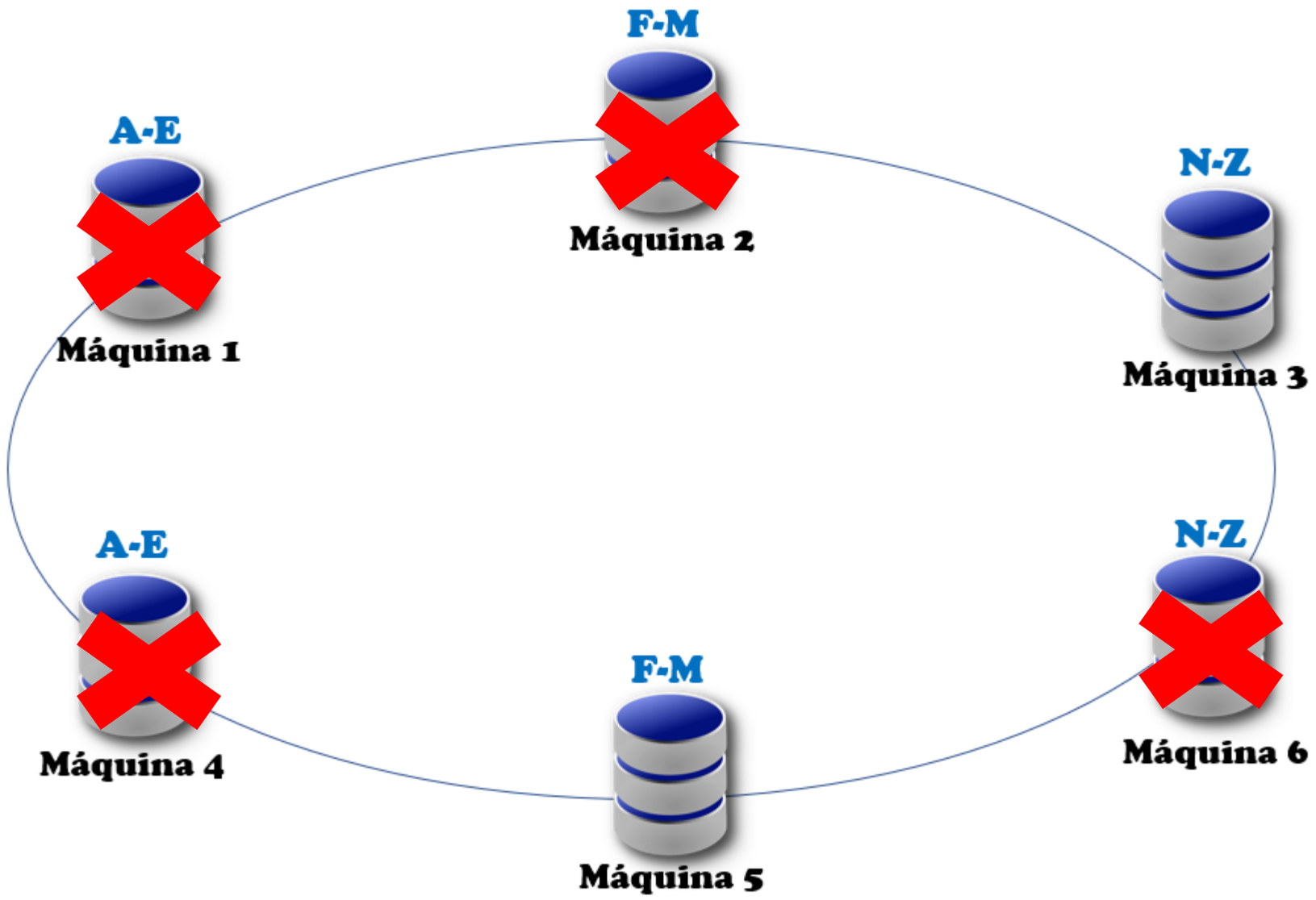
Availability

¿Cuántos userID hay empezando con "A"?



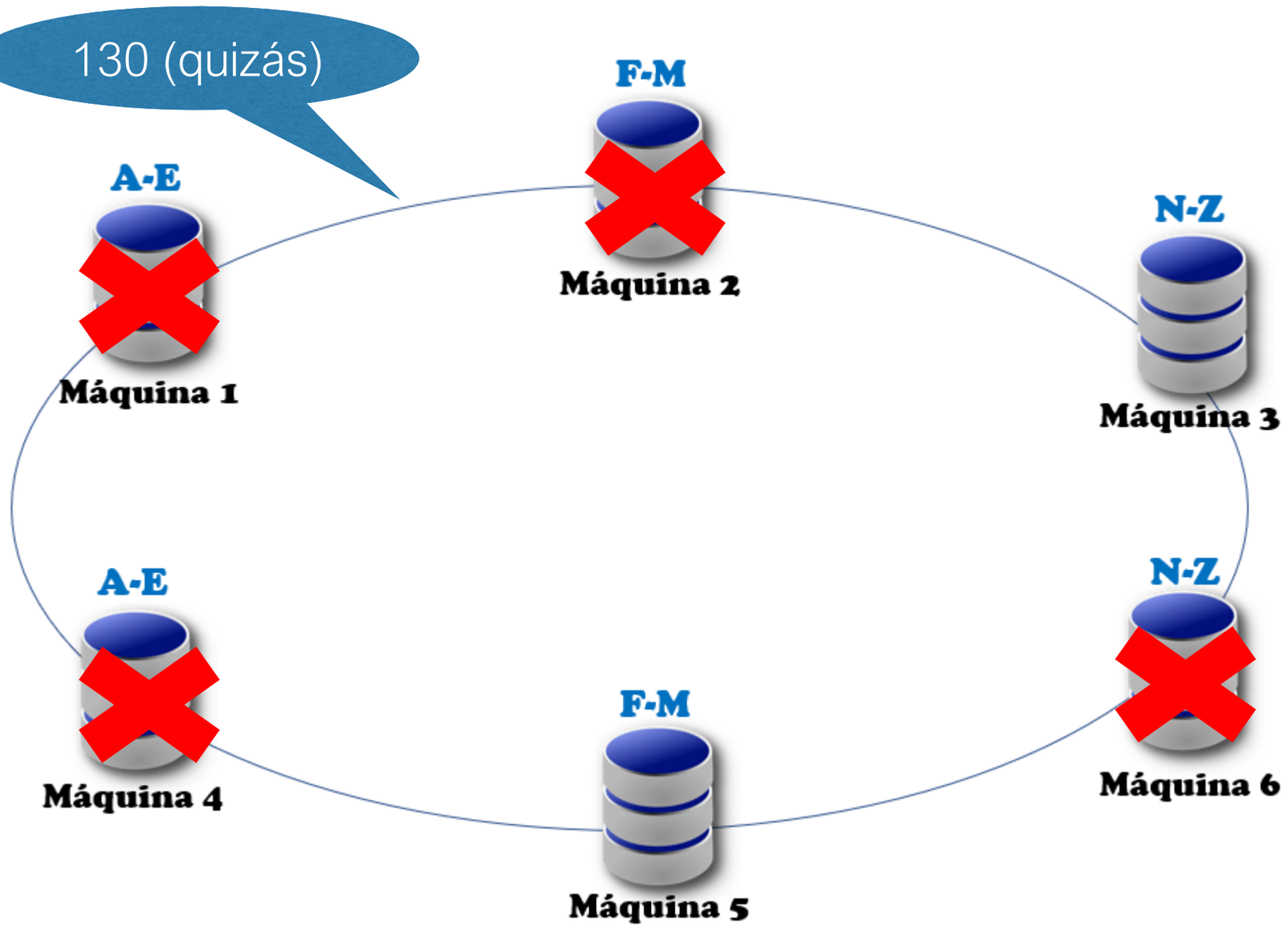
Availability

¿Cuántos userID hay empezando con "A"?



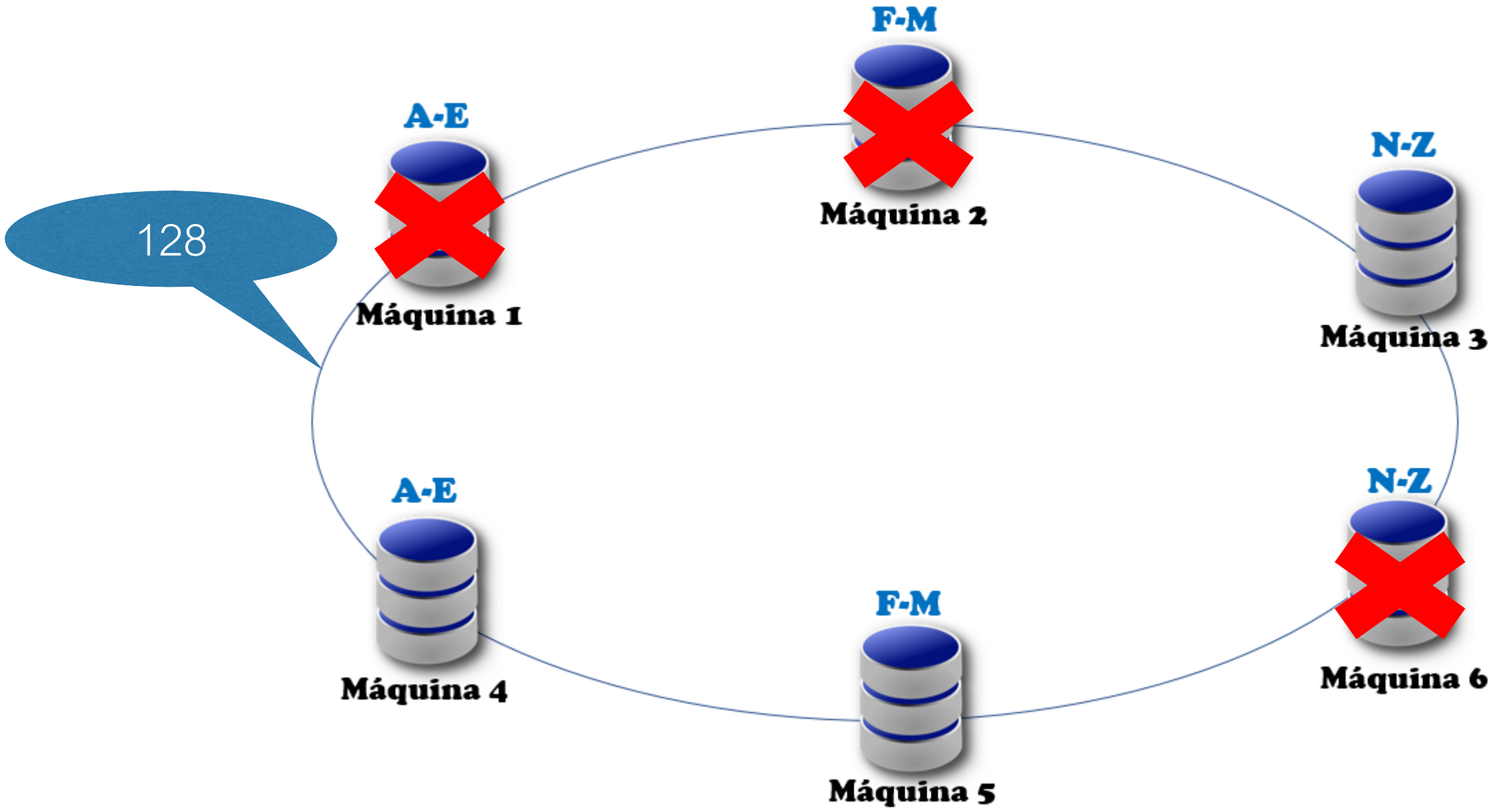
Availability

¿Cuántos userID hay empezando con "A"?



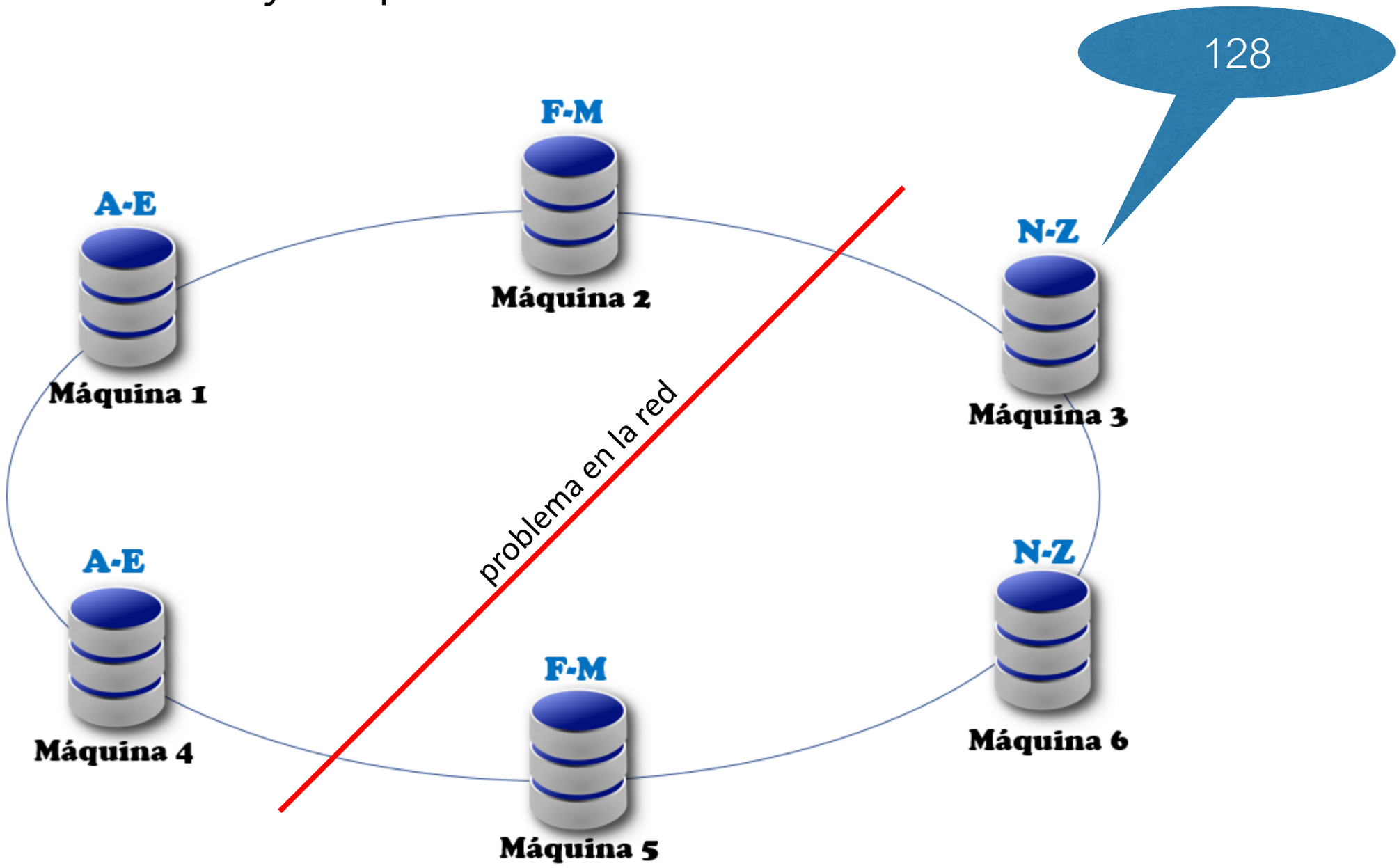
Partition tolerance

¿Cuántos userID hay empezando con "A"?



Partition tolerance

¿Cuántos userID hay empezando con "A"?



Teorema CAP

Plantea que para una base de datos distribuida es imposible mantener simultáneamente estas tres características:

- Consistency
- Availablity
- Partition tolerance

Teorema CAP

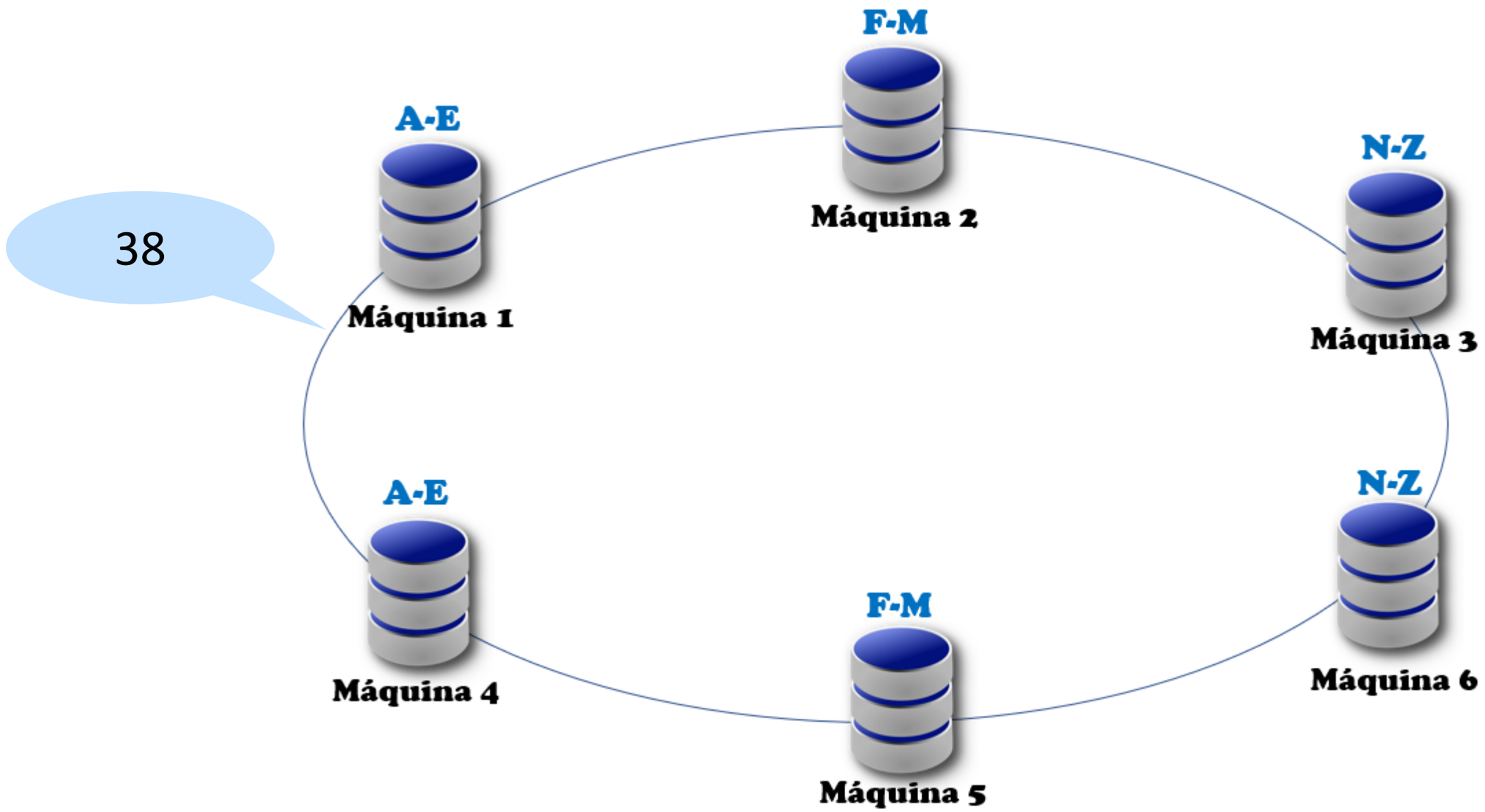
P es dado en cualquier sistema distribuido. Entonces, el Teorema CAP nos dice que hay que elegir entre:

- Consistency
- Availablity

Entonces tenemos sistemas CP y AP

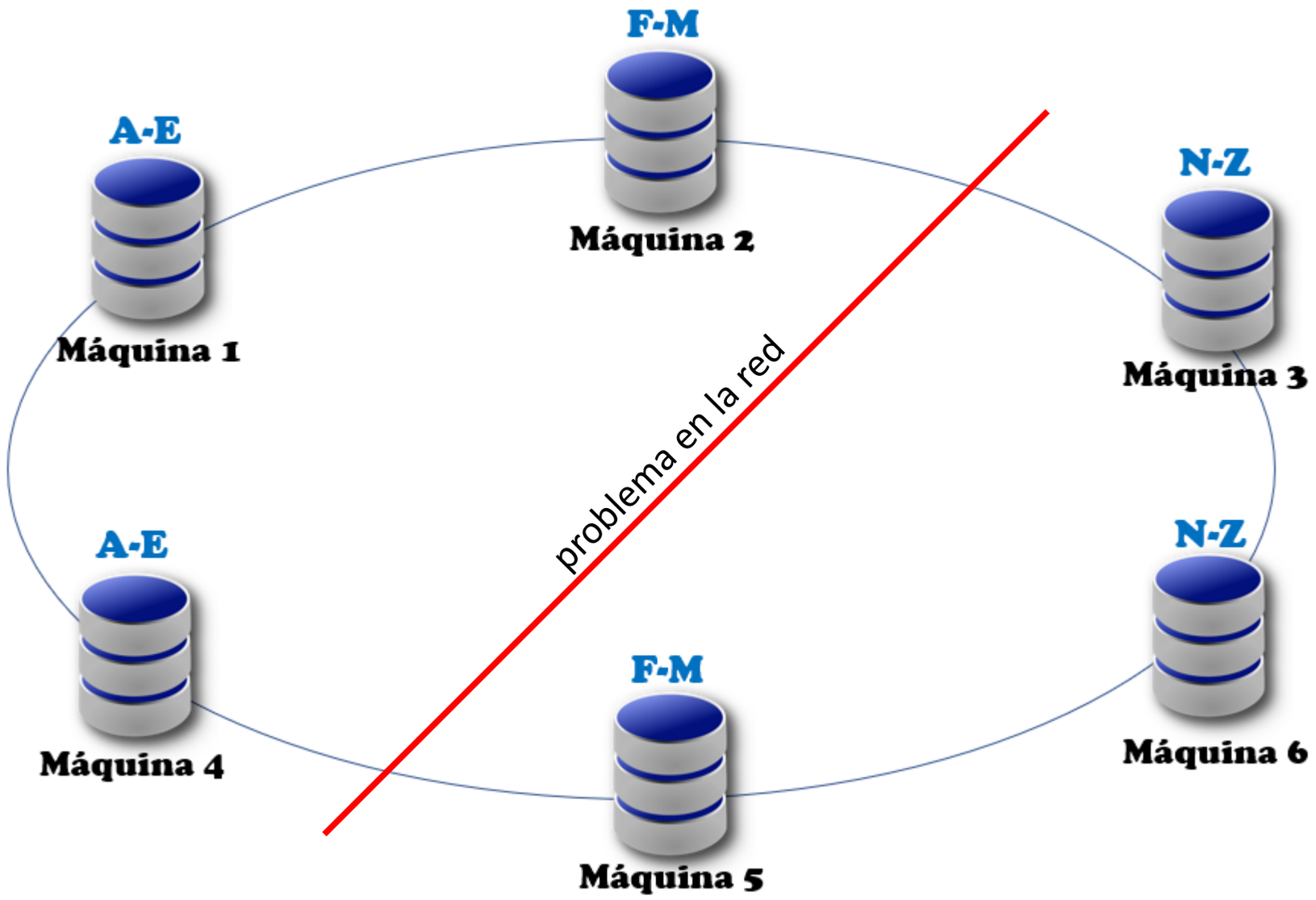
AP vs CP

¿Cuántos userID hay empezando con "Z"?



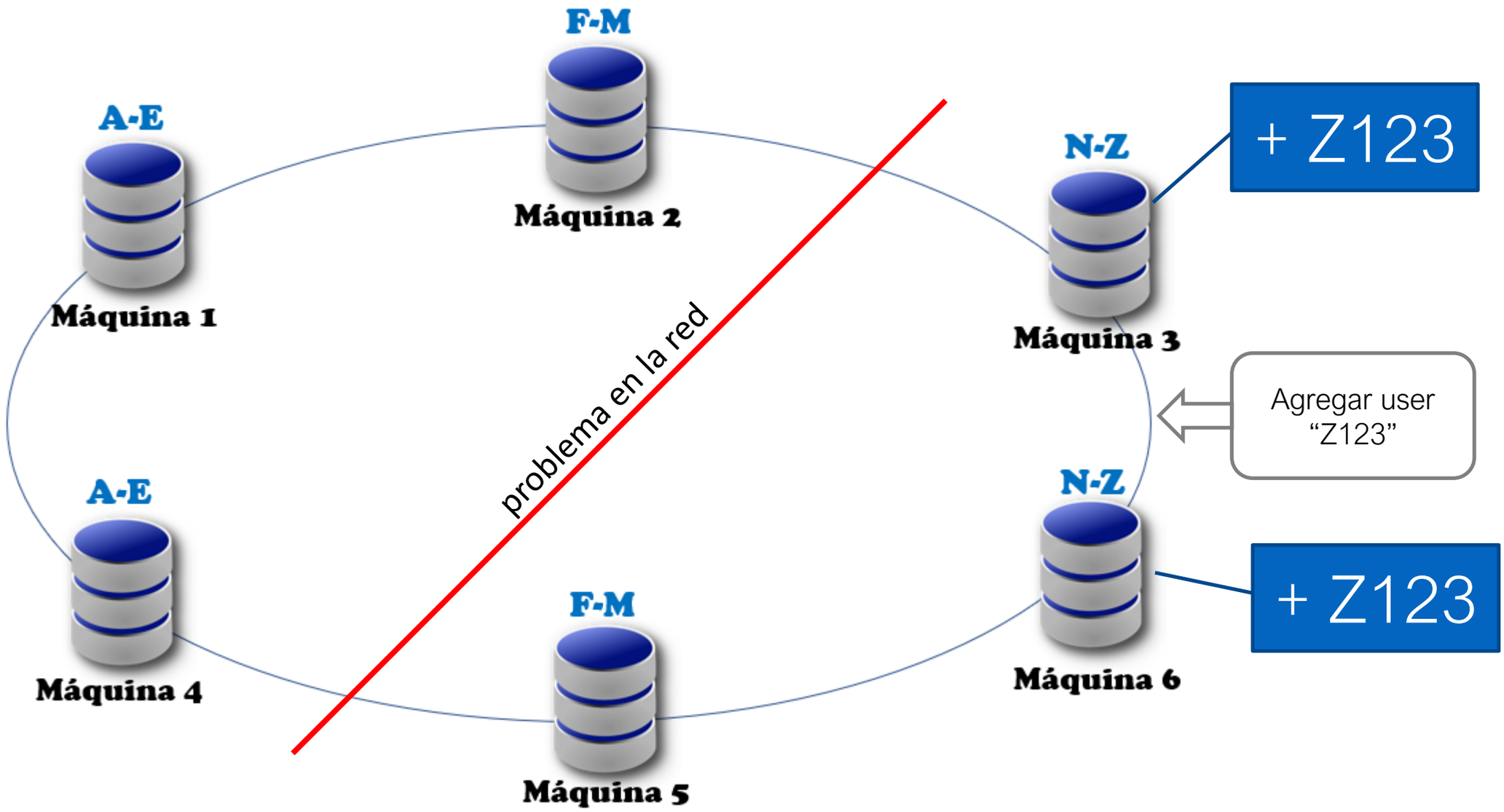
AP vs CP

¿Cuántos userID hay empezando con "Z"?



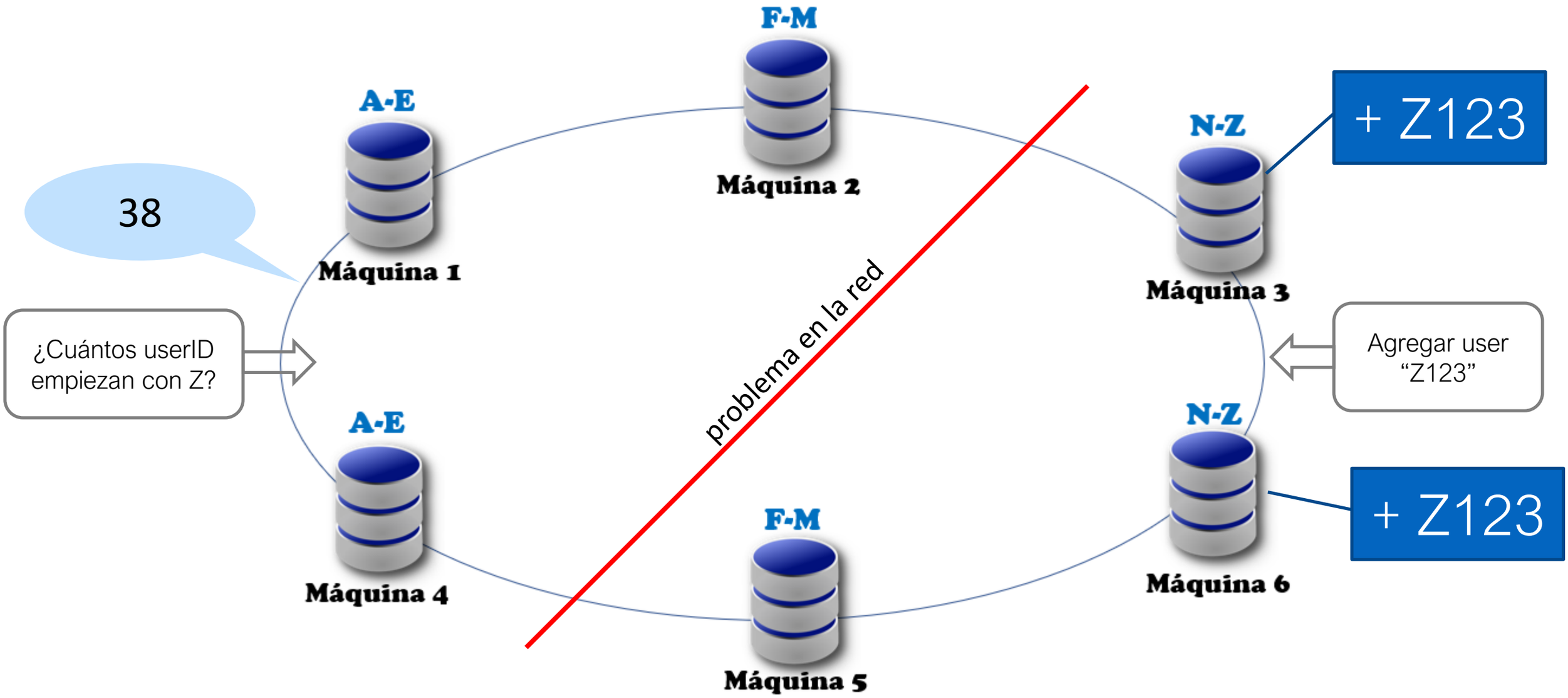
AP vs CP

¿Cuántos userID hay empezando con "Z"?



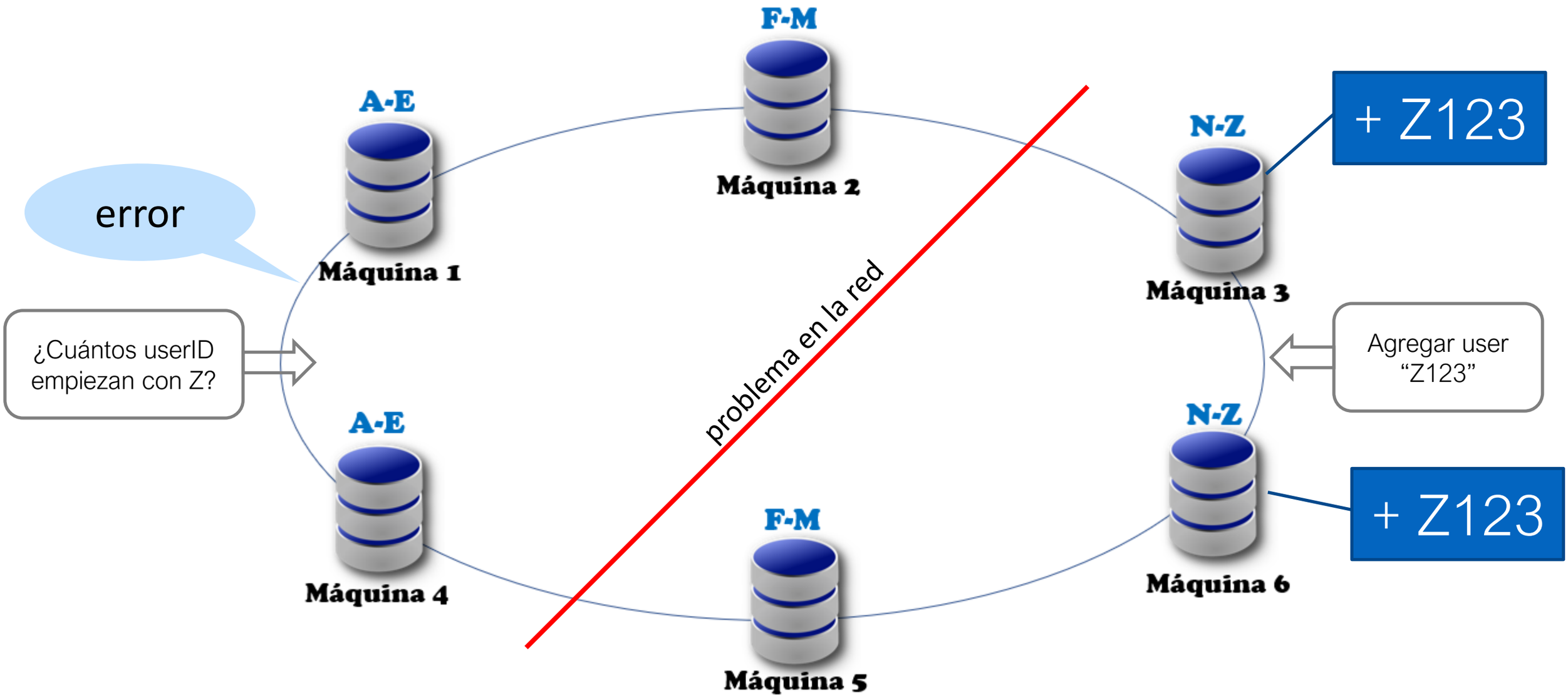
Un sistema AP

¿Cuántos userID hay empezando con "Z"?



Un sistema CP

¿Cuántos userID hay empezando con "Z"?



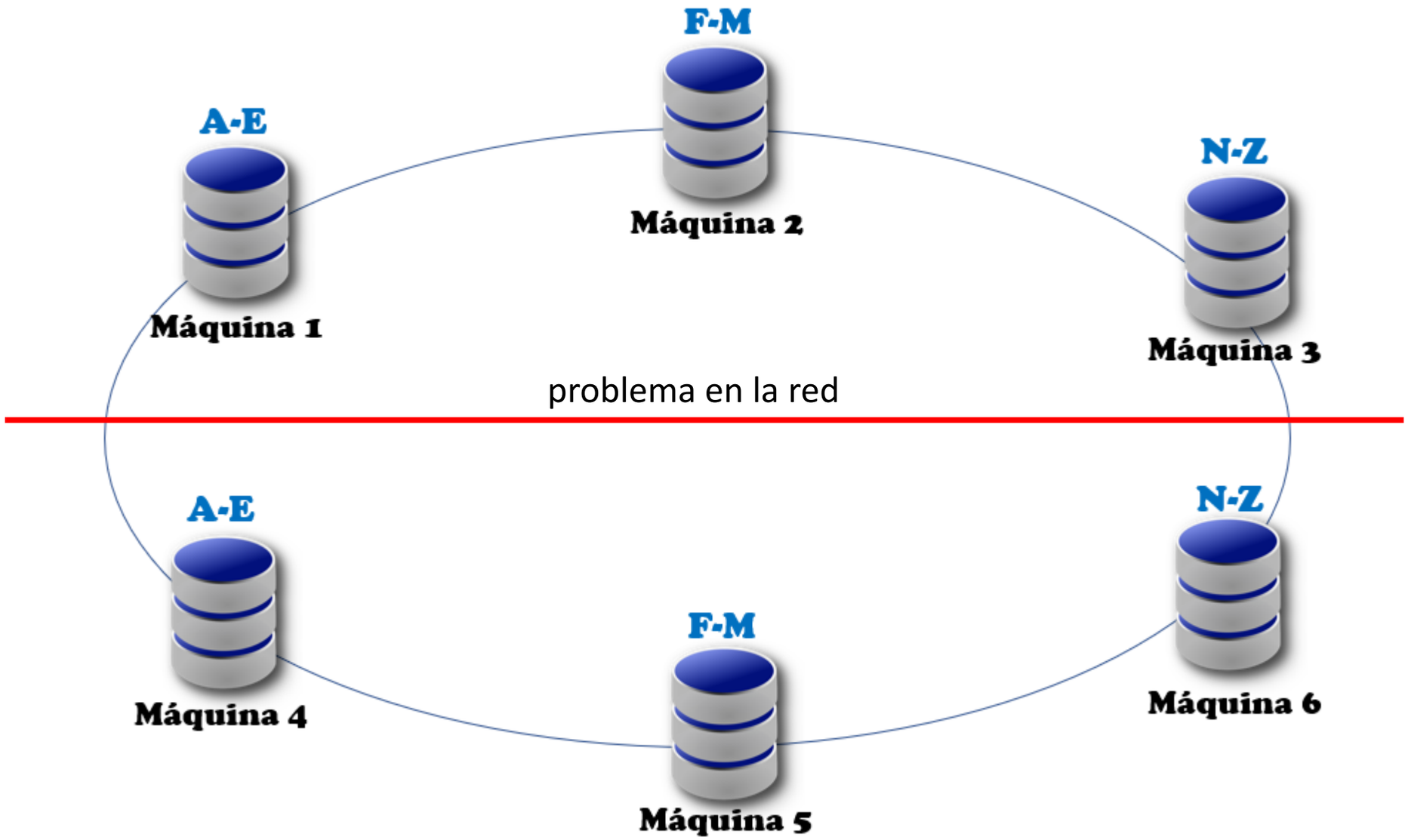
BASE

En la practica, sistemas distribuidos fijan el P, y balancean entre C y A, sin elegir uno exclusivamente.

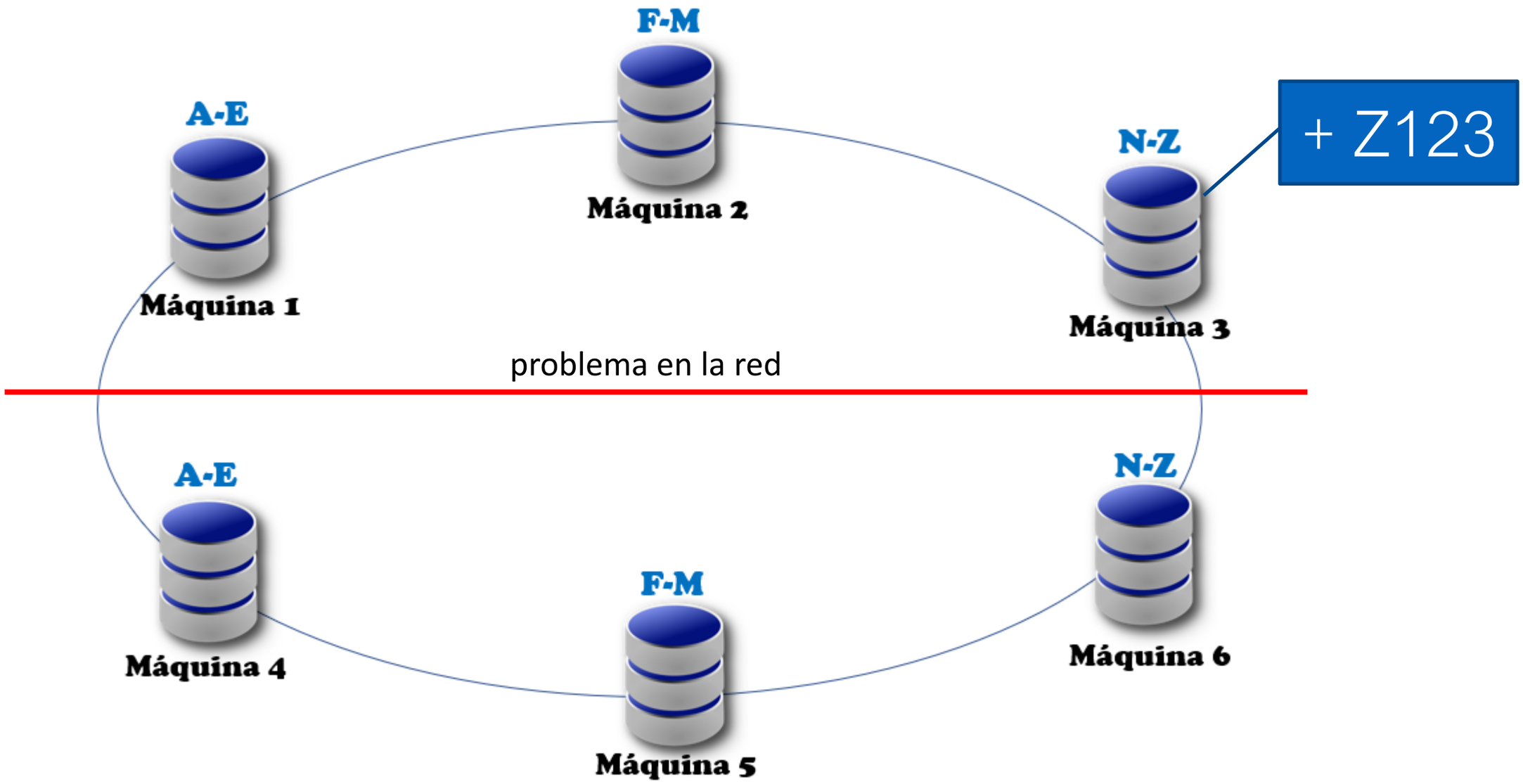
Paradigma BASE:

- Basically Available
- Soft state
- **Eventually consistent**

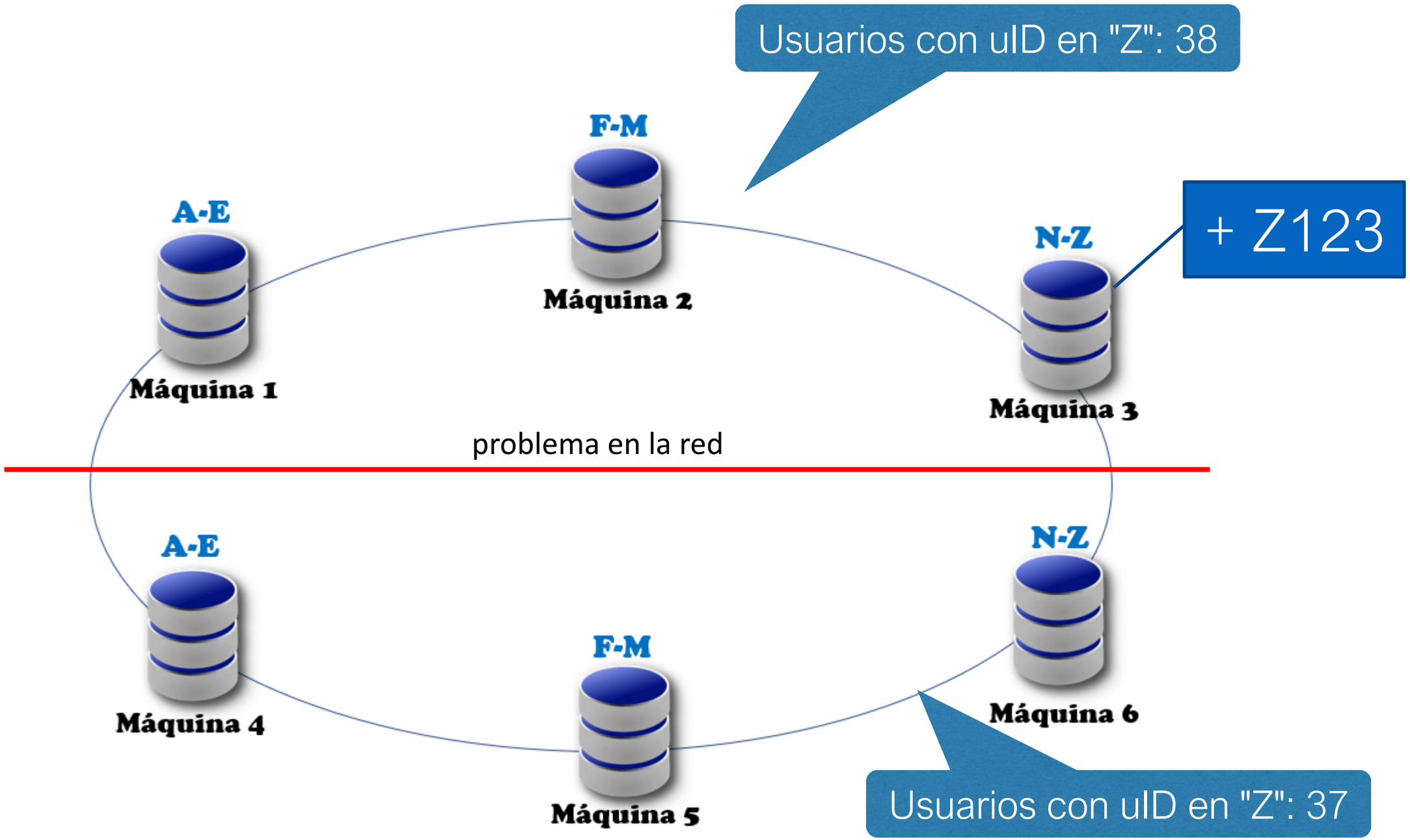
Consistencia eventual



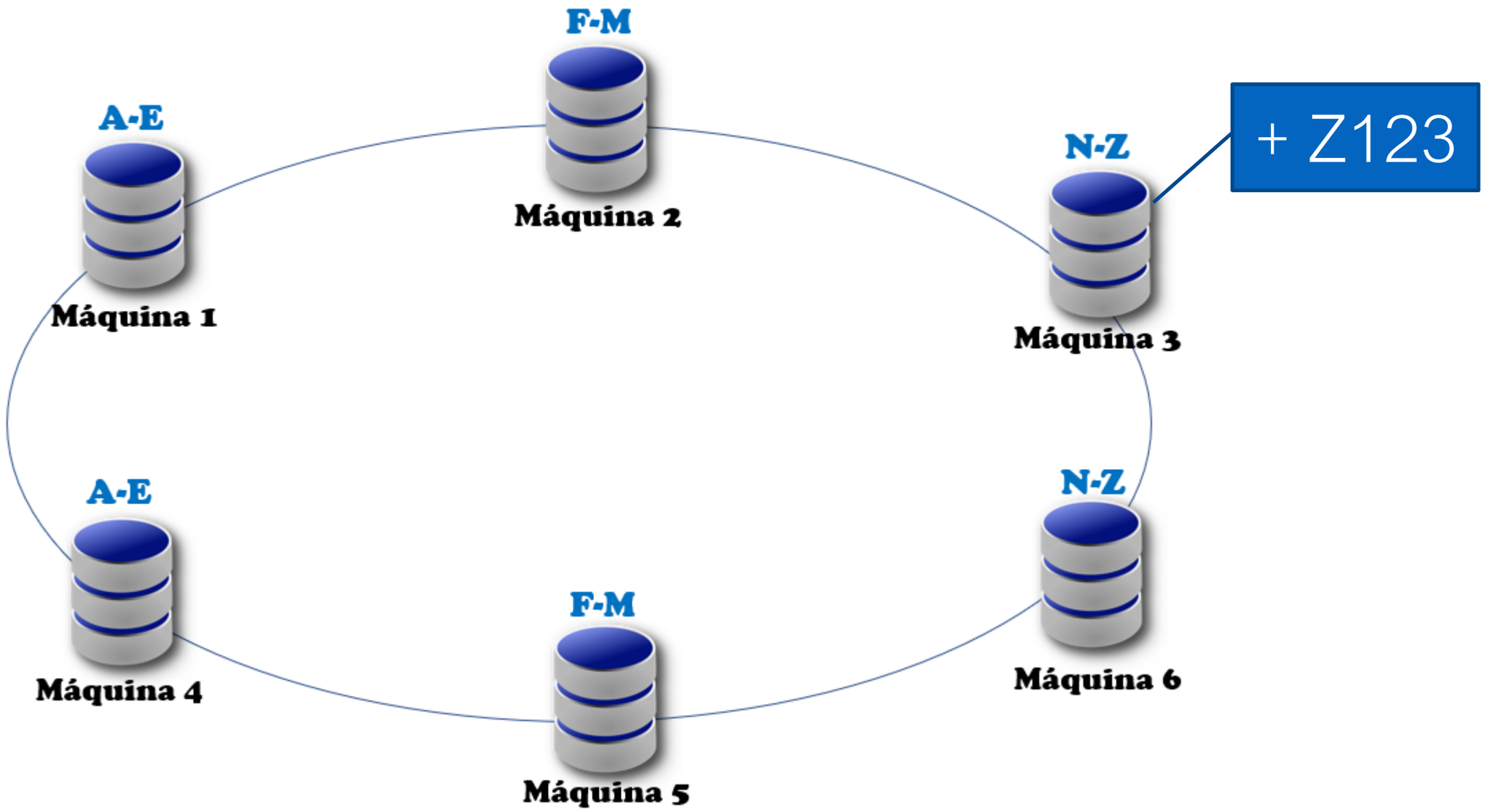
Consistencia eventual



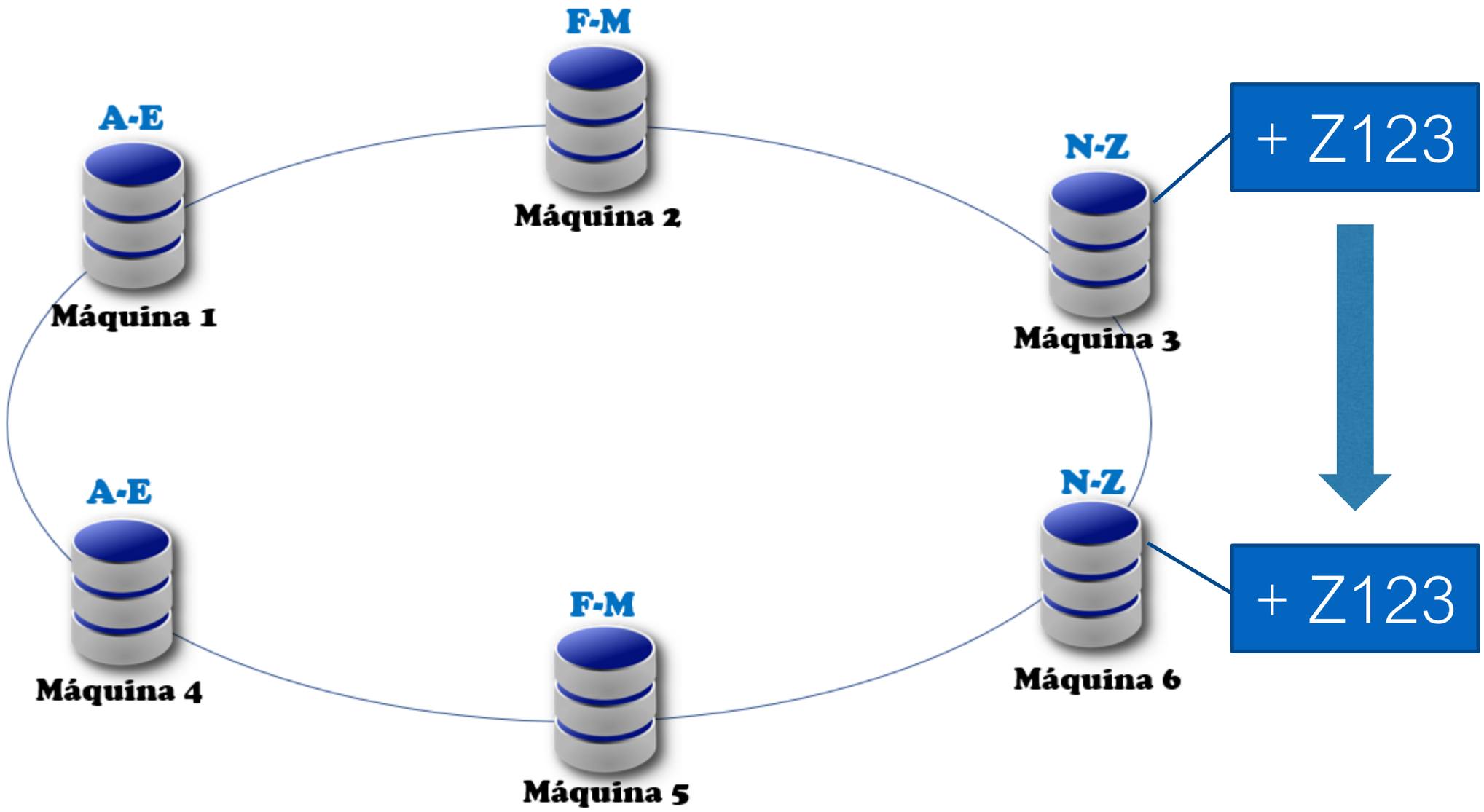
Consistencia eventual



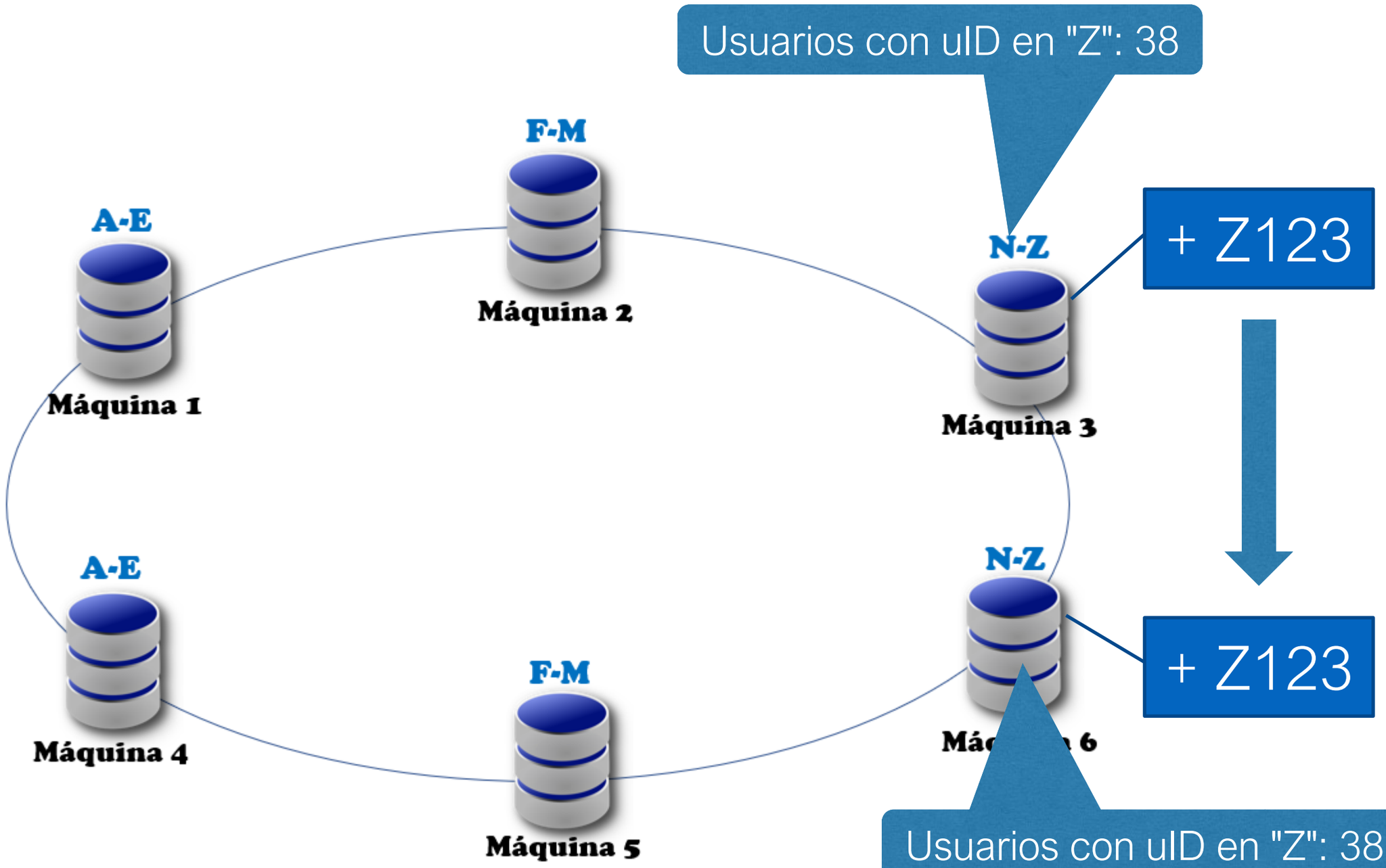
Consistencia eventual



Consistencia eventual



Consistencia eventual

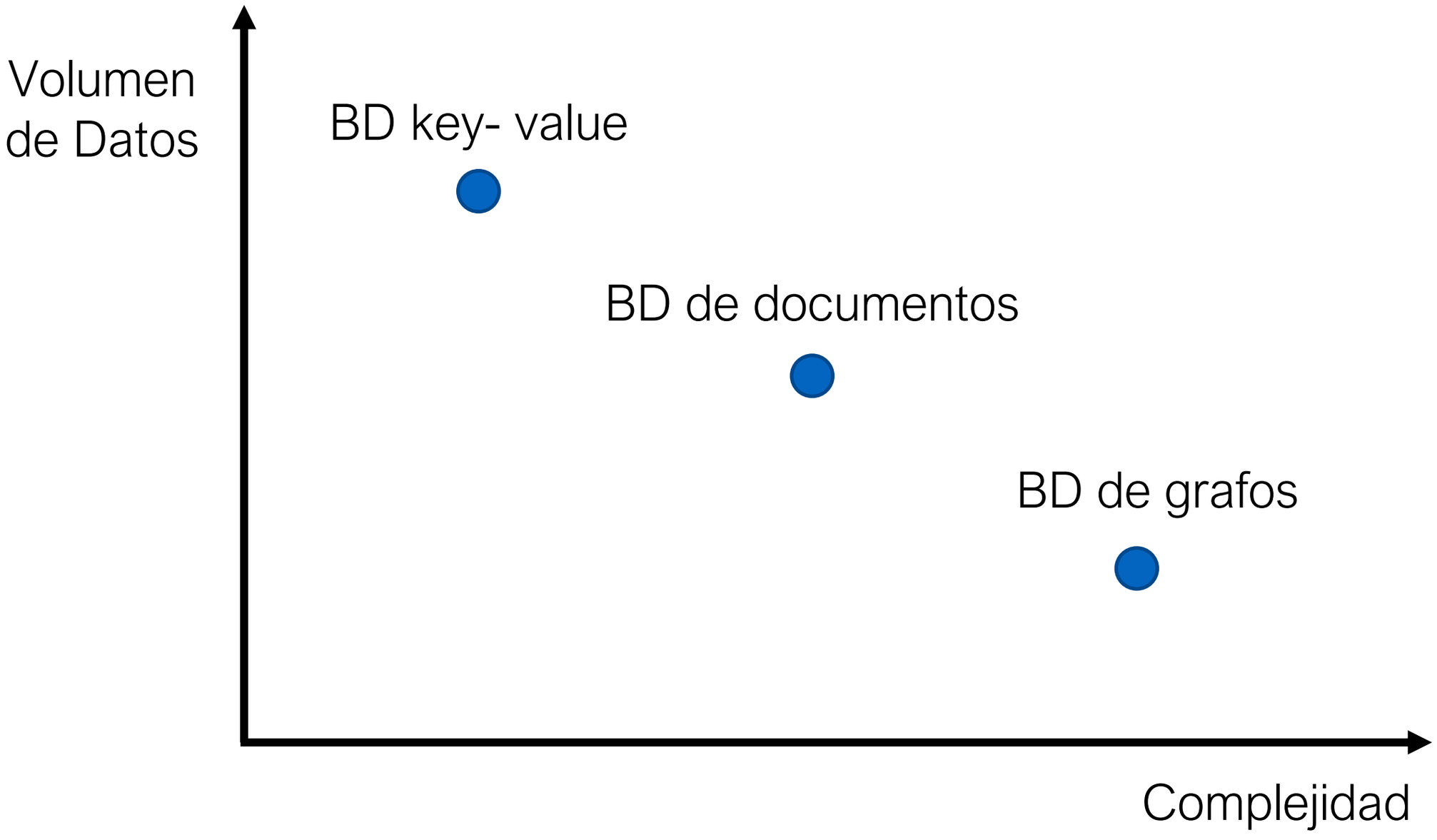


Sabores de NoSQL

Los más populares hoy en día:

- BD key-value
- BD de grafos
- BD de documentos

Sabores de NoSQL



BD Key - Value

Independientemente del esquema

- Arquitectura almacena información por medio de pares
- Cada par tiene una llave (identificador) y un valor

BD Key - Value

Operaciones cruciales:

- put(key,value)
- get(key)
- delete(key)

Key	Value
Chile	Santiago
Inglaterra	Londres
Escocia	Edinburgo
Francia	Paris
Alemania	Berlin
…	…

BD Key - Value

- Son grandes tablas de hash persistentes
- Esta categoría es difusa, pues muchas de las aplicaciones de otros tipos de BD usan key - value y hashing hasta cierto punto

Ejemplo más importante: Amazon Dynamo

BD Key - Value

Puede representar cualquier valor

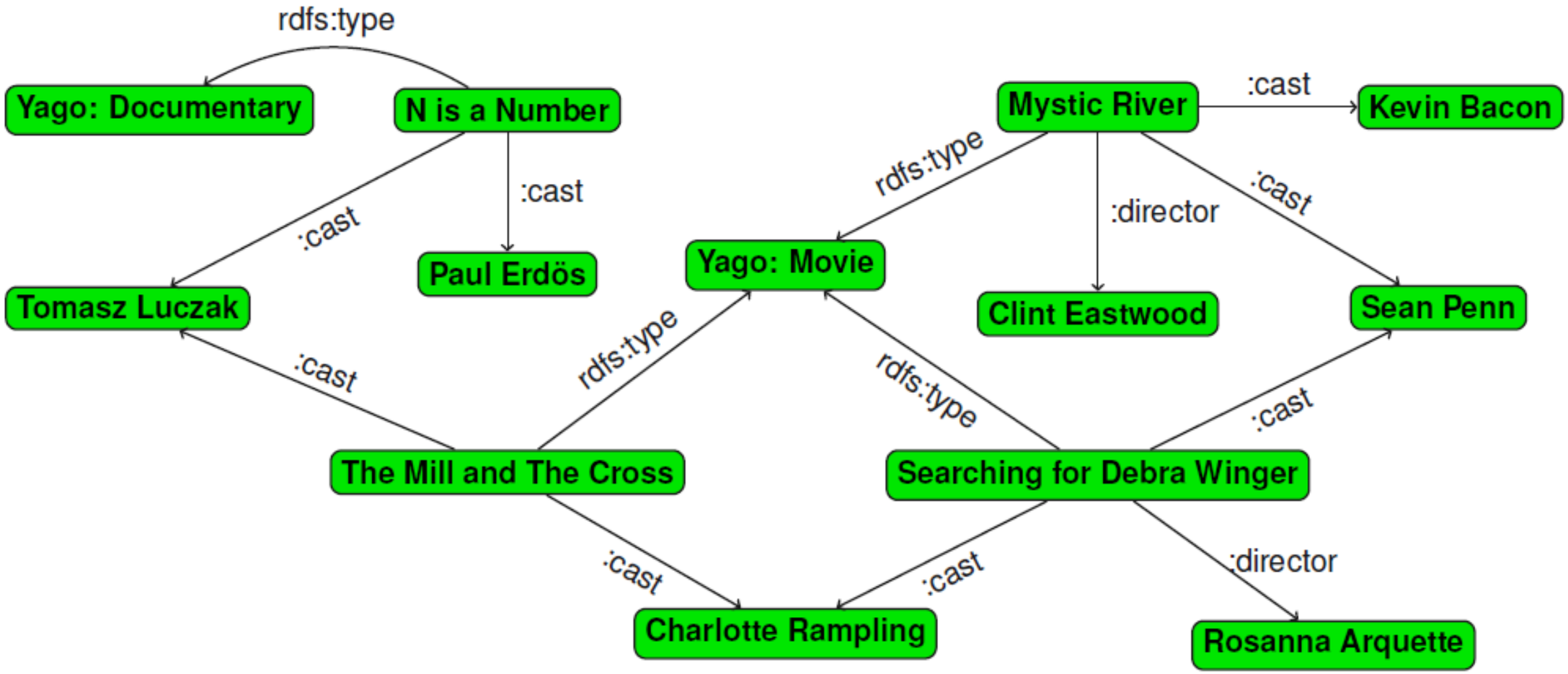
cartID	value
11789	usrID: "Juan", ítem: "Magic the Gathering Deck", value: …
12309	usrID: "Domagoj", item: "APEX XTX50 regulator set", value: …
…	…

BD de Grafos y RDF

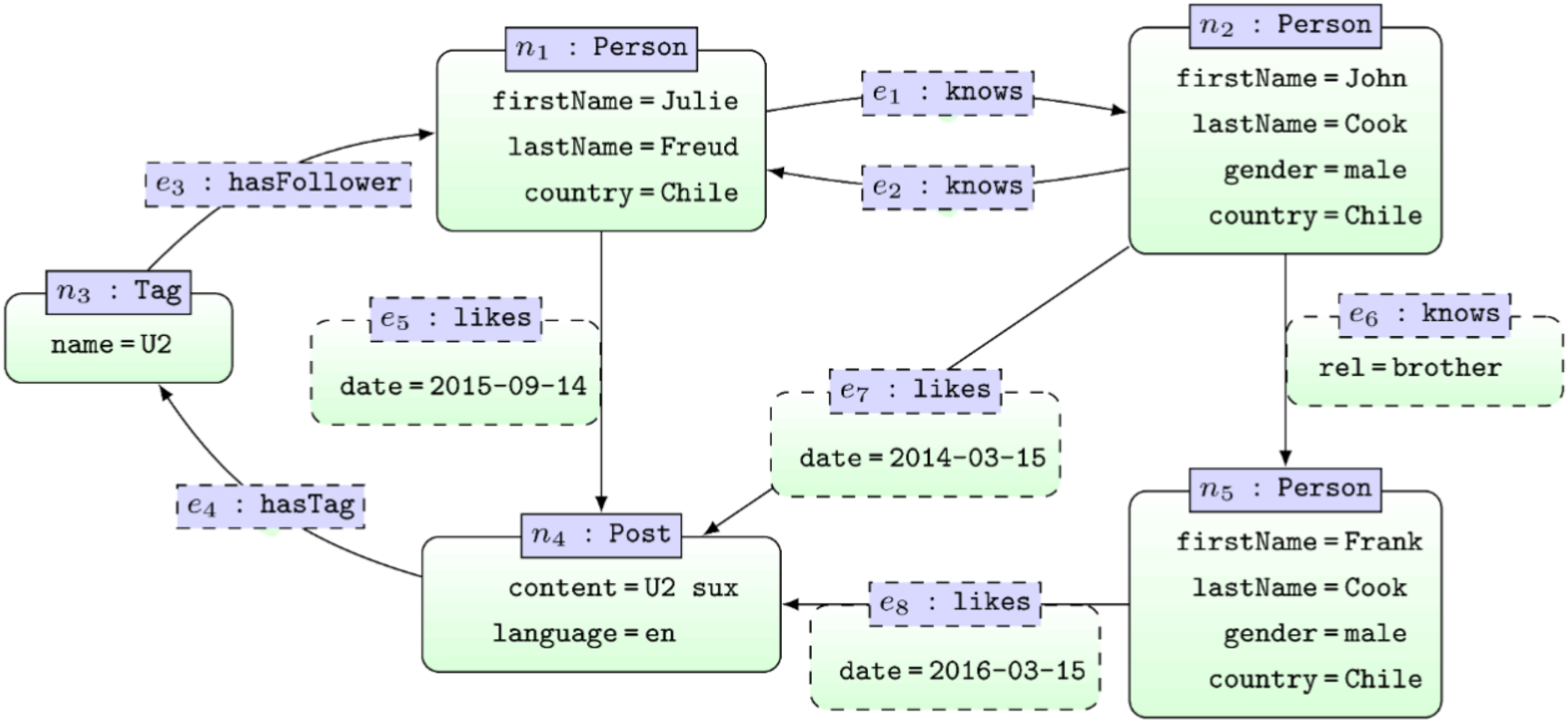
Especializadas para guardar relaciones

- En general, almacenan sus datos como property graphs
- Algunos ejemplos son Neo4J, Virtuoso, Jena, Blazegraph

BD de Grafos y RDF



BD de Grafos y RDF



BD Orientadas a Documentos

Especializadas en documentos

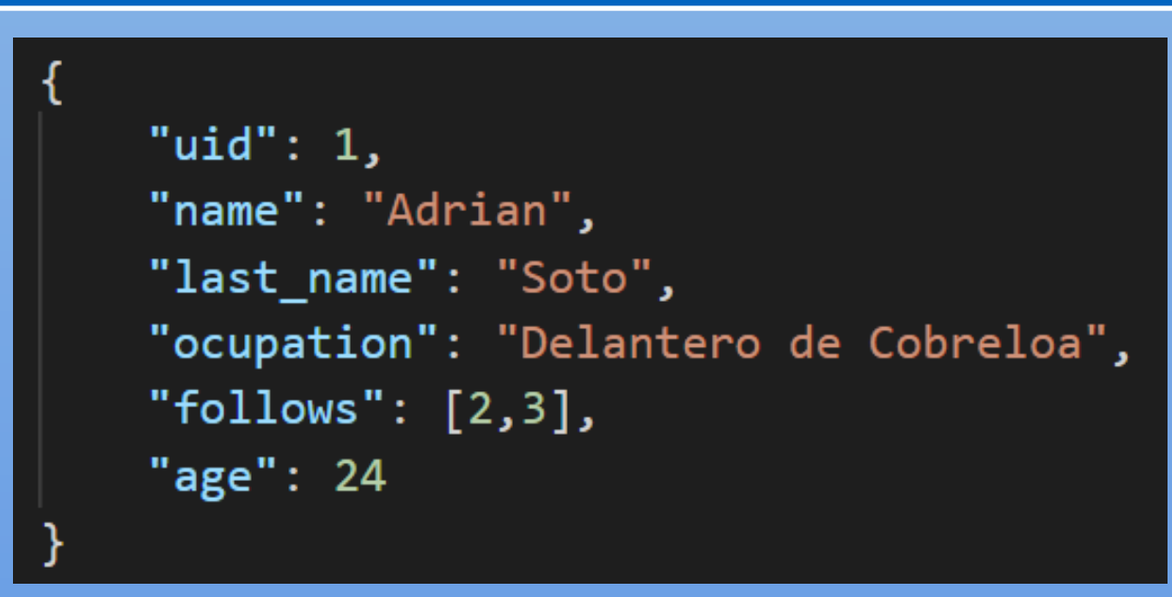
- CouchDB, MongoDB (estas y otras BD almacenan sus datos en documentos JSON)
- JSON no es el único estándar de documentos (por ejemplo, existe también XML)

BD Orientadas a Documentos

Parecidas a key-value stores:

- El valor es ahora un documento (JSON)
- Pueden agrupar documentos (colecciones)
- Lenguaje de consultas mucho más poderoso

BD Orientadas a Documentos

Usuarios	
Key	JSON
1	
2	…
…	…

JSON

Su nombre viene de JavaScript Object Notation

Estándar de intercambio de datos semiestructurados / datos en la Web

- JSON se acopla muy bien a los lenguajes de programación

JSON

Ejemplo


```
{
  "statuses": [
    {
      "id": 725459373623906304,
      "text": "@visitlondon: Have you been to any of these
                quirky London museums? https://t.co/tnrar8UttZ",
      "retweeted_status": {
        "metadata": {
          "result_type": "recent",
          "iso_language_code": "en"
        },
        "retweet_count": 239,
        "retweeted": false
      }
    }
  ]
}
```

JSON

La base son los pares key - value

```
{
  “nombre": “Matías”, “apellido": “Jünemann”
}
```

Valores pueden ser:

- Números
- Strings (entre comillas)
- Valores booleanos
- Arreglos (por definir)
- Objetos (por definir)
- null

JSON

Sintaxis

Los objetos se escriben entre {} y contienen una cantidad arbitraria de pares key - value

```
{
  “nombre": “Matías”, “apellido": “Jünemann”
}
```

JSON

Sintaxis

Los arreglos se escriben entre [] y contienen valores

```
{
  “profesores”: [
    {“nombre”: “Juan”, “apellido": “Reutter”},
    {“nombre”: “Cristian”, “apellido": “Riveros”},
    {“nombre”: “Marcelo”, “apellido": “Arenas”}
  ]
}
```

JSON vs SQL

SQL:

- Esquema de datos
- Lenguajes de consulta independientes del código

JSON:

- Más flexible, no hay que respetar necesariamente un esquema
- Más tipos de datos (como arreglos)
- Human - Readable

JSON

Lenguaje de consultas

Hay intentos de lenguajes de consulta para objetos JSON que usen su estructura de árbol:

- Por ejemplo, JSONPath

Importante: JSON está ahí para los programadores que NO buscan separar datos del código

JSON

Lenguaje de consultas

¿Por qué necesitamos esquemas para JSON?

- JSON Schema: propuesta toma fuerza el 2013 - 2014
- Harta investigación en el DataLab UC

JSON Schema

```
{
  “first_name”: “Alexis”,
  “last_name”: “Sánchez”,
  “age”: 28,
  “club”: {
    “name”: “Arsenal FC”,
    “founded”: 1886,
  }
  “first_club”: “Cobreloa”,
  “va_al_mundial”: false
}
```

JSON Schema

```
{
    "type": "object",
    "properties": {
        "first_name": { "type": "string" },
        "last_name": { "type": "string" },
        "age": { "type": "integer" },
        "club": {
            "type": "object",
            "properties": {
                "name": { "type": "string" },
                "founded": { "type": "integer" }
            },
            "required": ["name"]
        }
        “first_club": { "type": "string" },
        “va_al_mundial”: { "type": “boolean" },
    },
    "required": ["first_name", "last_name", "age", "club"]
}
```

Ver más en: http://cswr.github.io/JsonSchema/

BD de documentos

Especializadas en documentos: almacenan muchos documentos JSON

- Si quiero libros: un documento JSON por libro
- Si quiero personas: un documento JSON por persona

Notar que esto es altamente jerárquico

BD de documentos

Qué hacen bien:

- Si quiero un libro o persona en particular
- Cruce de información **simple**

Muy útiles a la hora de desplegar información en la web

BD de documentos

Pueden verse como un cache de una BD relacional
¿Por qué?

Cache de BD SQL

Students

StudentID	Nombre	Carrera
1	Alice Cooper	Computación
2	David Bowie	Todas
3	Charly García	Ingeniería Civil
…	…	…

Courses

courseID	name	year
IIC2413	Databases	2020
IMT3830	Game Theory	2020
…	…	…

Takes

courseID	StudentID
IIC2413	1
IIC2413	2
IMT3830	2
…	…

Cache de BD SQL

Lista de alumnos por curso:

- SQL tiene que hacer un join
- En BD documentos prepararemos esta información

Cache de BD SQL

Colección "**Courses**"

```
{
    "courseID": IIC2413,
    "name": "Databases",
    "year": 2020,
    "students":[
        {
            "studentID": 1,
            "name": "Alice Cooper"
        },
        {
            "studentID": 2,
            "name": "David Bowie"
        },
        ...
    ]
}
```

Cache de BD SQL

Colección "**Courses**"

```
{
    "courseID": IIC2413,
    "name": "Databases",
    "year": 2020,
    "students":[
        {
            "studentID": 1,
            "name": "Alice Cooper"
        },
        {
            "studentID": 2,
            "name": "David Bowie"
        },
        ...
    ]
}
```

```
{
    "courseID": IMT3830,
    "name": "Game Theory",
    "year": 2020,
    "students":[
        {
            "studentID": 2,
            "name": "David Bowie"
        },
        {
            "studentID": 3,
            "name": "Charly García"
        },
        ...
    ]
}
```

BD de documentos

Qué hacen bien:

- Si quiero lista de alumnos de un curso
- Si quiero nombres de todos los cursos
- Si quiero todo los cursos tomados por David

Muy útiles a la hora de desplegar información en la web

BD de documentos

Qué hacen mal:

- Manejo de información que cambia mucho
- Cruce de información no trivial

BD de documentos

Colección "**Courses**"

- Todos los alumnos que toman IIC2413 y IMT3830

```
{
    "courseID": IIC2413,
    "name": "Databases",
    "year": 2020,
    "students":[
        {
            "studentID": 1,
            "name": "Alice Cooper"
        },
        {
            "studentID": 2,
            "name": "David Bowie"
        },
        ...
    ]
}
```

```
{
    "courseID": IMT3830,
    "name": "Game Theory",
    "year": 2020,
    "students":[
        {
            "studentID": 2,
            "name": "David Bowie"
        },
        {
            "studentID": 3,
            "name": "Charly García"
        },
        ...
    ]
}
```

BD de documentos

Colección "**Courses**"

- Todos los alumnos que toman IIC2413 y IMT3830

Efectivamente hay que hacer un nested loop join:

- Iterar por todos los alumnos de IMT3830
- Iterar por todos los alumnos de IIC2413
- Ver si hacen el join

MongoDB soporta JavaScript y Python

- Se puede hace, pero no es elegante

En breve

BD de documentos:

- Útiles para despliegue de información estática
- Búsquedas simples
- Cruces muy sencillos

BD SQL:

- Información cambia mucho
- Tengo que hacer cruces cada rato
- Necesito ACID

BD Documentos y BASE

- Distintas aplicaciones en una misma base de datos acceden a distintos documentos al mismo tiempo
- En general diseñadas para montar varias instancias que (en teoría) tienen la misma información
- Propagan updates en forma descoordinada

Proveen "**Consistencia Eventual**"

Consistencia Eventual

La consistencia eventual puede generar problemas

Si dos aplicaciones intentan acceder al mismo documento en MongoDB, estas pueden ser versiones diferentes del documento