03 Diseño

Tolerancia a fallos

El concepto de mayoría es clave porque es el que desencadena las elecciones para elegir un nuevo primario cuando éste cae.

La mayoría es definido por más de la mitad de los miembros del set (incluyendo los que estén caídos en ese momento).

Number of members in the set	Majority of the set
1	1
2	2
3	2
4	3
5	3
6	4
7	4

El motivo de que la mayoría se defina sobre el total de miembros es evitar que en una partición de red parcial pudiera llegar a haber 2 primarios al mismo tiempo.

Con una configuración de 5 miembros en la que se caigan 3 incluyendo el primario, los dos restantes quedarían como secundarios y no podrían elegirse entre ellos porque no tienen mayoría sobre el total: el cluster quedaría en modo solo lectura o fuera de servicio.

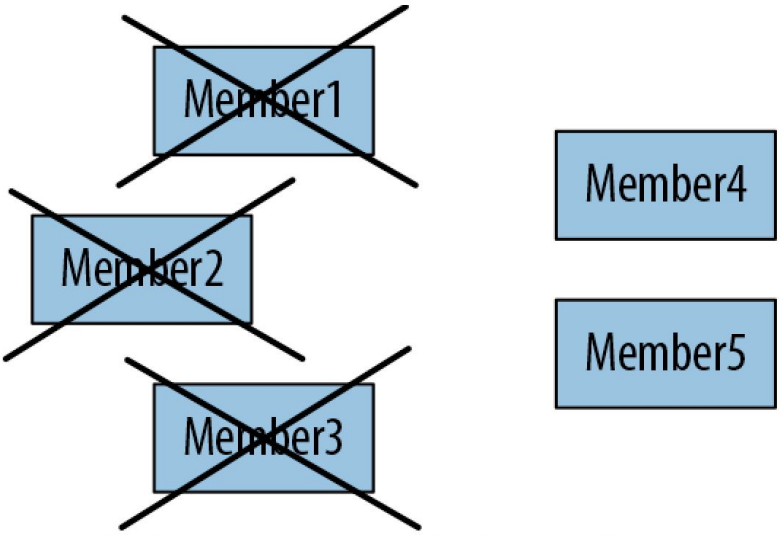


Figure 10-1. With a minority of the set available, all members will be secondaries

Esto es así porque en una partición de red en la que 3 quedarán aislados, al tener mayoría elegirán un primario, si el otro grupo de dos también pudieran elegirse entre ellos, podría darse la circunstancia de existir dos primarios y generar inconsistencias.

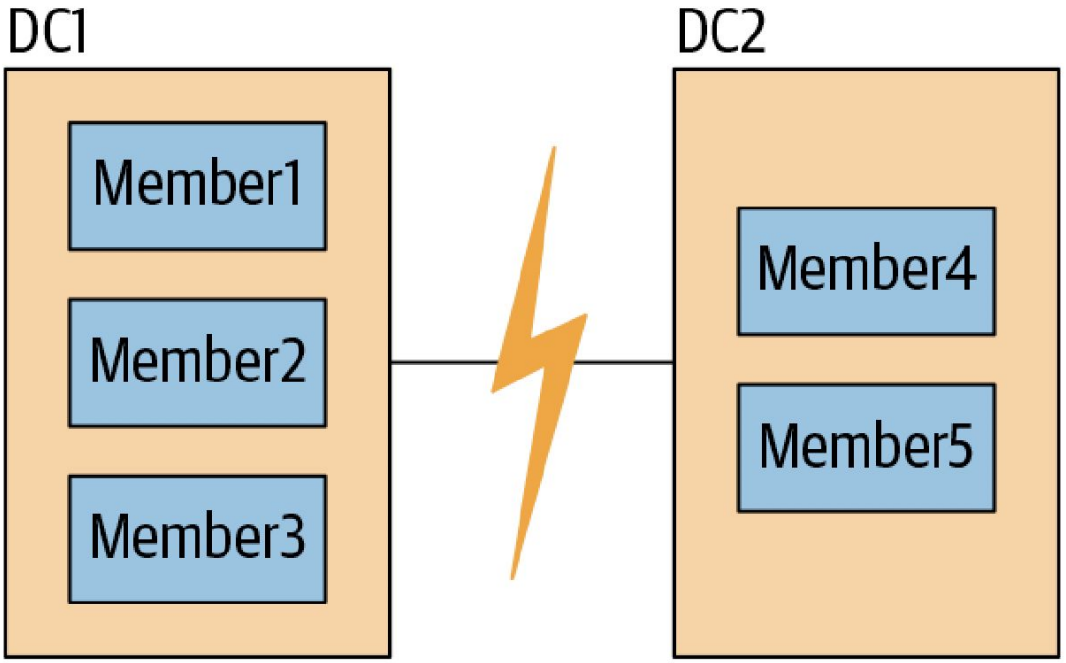


Figure 10-2. For the members, a network partition looks identical to servers on the other side of the partition going down

De esta manera, la tolarencia a fallos de un replica set es el número de miembros que pueden llegar a estar no disponibles para mantener la posibilidad de que se elija un nuevo primario. En otras palabras, la diferencia entre el número de miembros totales y la mayoría de miembros necesaria para generar las elecciones.

Number of Members	Majority Required to Elect a New Primary	Fault Tolerance
3	2	1
4	3	1
5	3	2
6	4	2

Adding a member to the replica set does not *always* increase the fault tolerance. However, in these cases, additional members can provide support for dedicated functions, such as backups or reporting.

Como se ve en la tabla anterior, añadir un miembro puede no incrementar la tolerancia a fallos, aunque podría servir para usarlos como otros usos dedicados, backups, reporting, etc.

Balanceo de carga en despliegues con alta demanda de operaciones lectura

En despliegues con alta demanda de lectura se pueden distribuir las lecturas a miembros secundarios.

Distribución de los miembros en diferentes datacenters

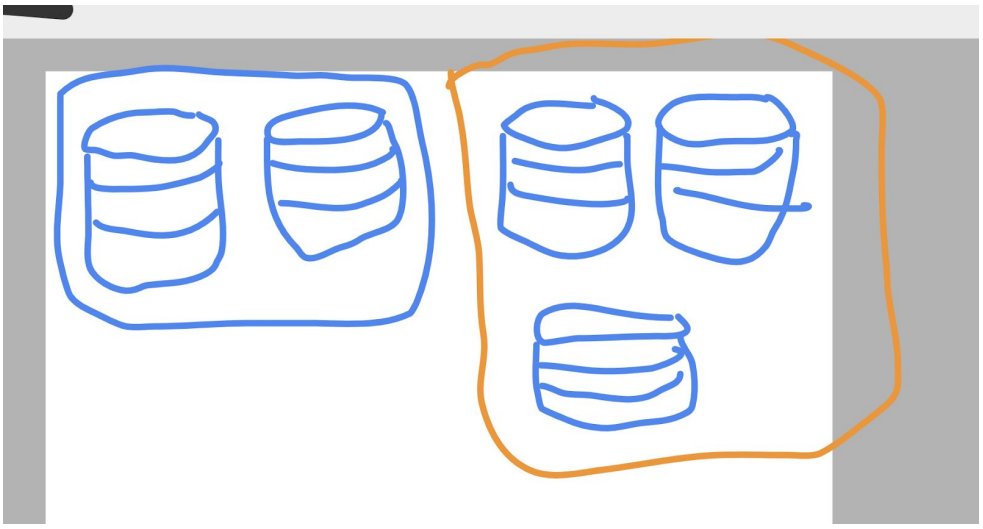
Beneficios

- Si uno de los data centers cae, los datos se mantendrán al menos disponibles para lecturas en el otro.
- Si es el centro de datos con minoría el que cae, se mantendrán tanto las operaciones de escritura como de lectura.

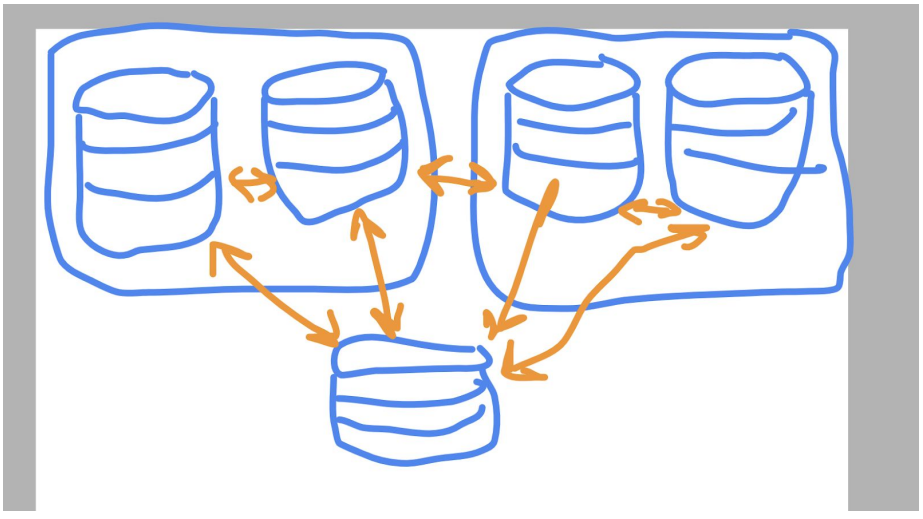
Si es posible, se deben distribuir miembros en al menos tres centros de datos (o más, según el número de miembros). Si el coste del tercer centro de datos es prohibitivo, una posibilidad de distribución es distribuir de manera uniforme los miembros con datos en los dos centros de datos y almacenar el miembro restante en la nube si la política de su empresa lo permite.

Mayoría en el despliegue en varios datacenters

- La mayoría del conjunto en un centro de datos, como en la Figura 10-2, sería un buen diseño si tiene un centro de datos primario donde siempre desea que se ubique el primario del conjunto de réplicas. Mientras su centro de datos primario esté en buen estado, tendrá uno primario. Sin embargo, si ese centro de datos no está disponible, su centro de datos secundario no podrá elegir un nuevo primario y el cluster entraría en modo lectura.



- Un número igual de servidores en cada centro de datos, más un servidor de desempate en una tercera ubicación. Este es un buen diseño si sus centros de datos son "iguales" con preferencia, ya que generalmente los servidores de cualquiera de los centros de datos podrán ver la mayoría del conjunto. Sin embargo, implica tener tres ubicaciones separadas para los servidores.



Los requisitos más complejos pueden requerir diferentes configuraciones, pero debe tener en cuenta cómo su conjunto adquirirá una mayoría en condiciones adversas.

Todas estas complejidades desaparecerían si MongoDB admitiera tener más de un primario. Sin embargo, esto traería su propia serie de complejidades. Con dos primarias, tendría que manejar escrituras en conflicto (por ejemplo, si alguien actualiza un documento en una primaria y alguien lo elimina en otra primaria). Hay dos formas populares de manejar conflictos en sistemas que admiten múltiples escritores: la reconciliación manual o hacer que el sistema elija arbitrariamente un "ganador". Ninguna de estas opciones es un modelo muy fácil para que los desarrolladores codifiquen, ya que no puede ser asegúrese de que los datos que ha escrito no cambien debajo de usted. Por lo tanto, MongoDB eligió admitir solo tener un único primario. Esto facilita el desarrollo, pero puede generar períodos en los que el conjunto de réplicas es de solo lectura.