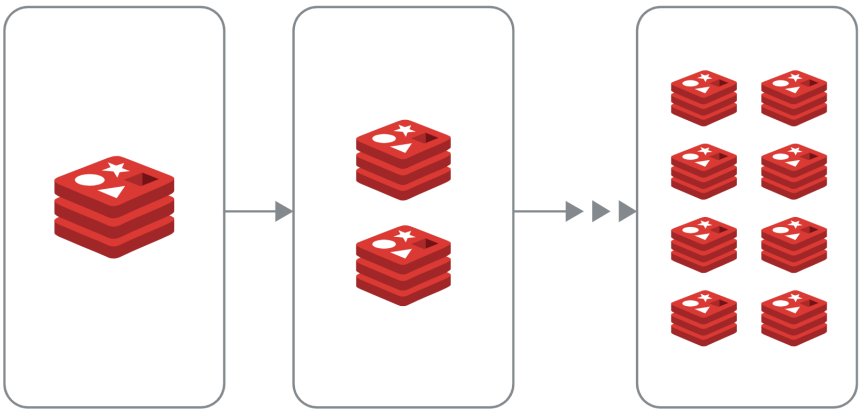
Scaling

Linear Scaling with Redis Enterprise

There are two ways to scale database shards:

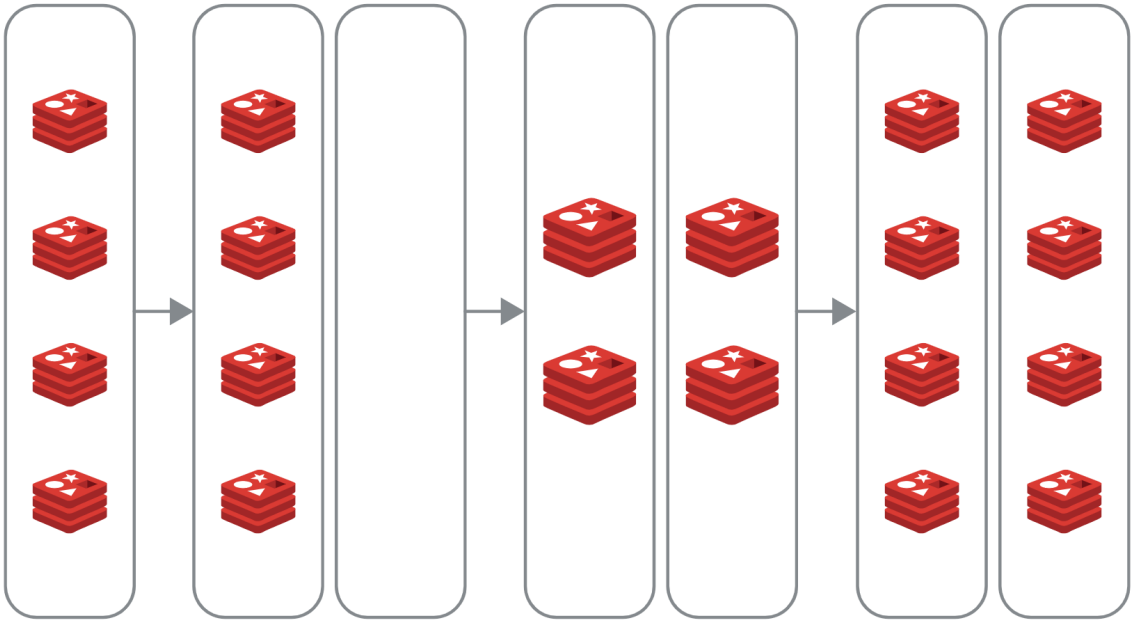
1. Scaling up by adding shards to your database without adding nodes to your cluster. This scenario is useful when there is enough un-utilized capacity in the cluster:

Scaling Up



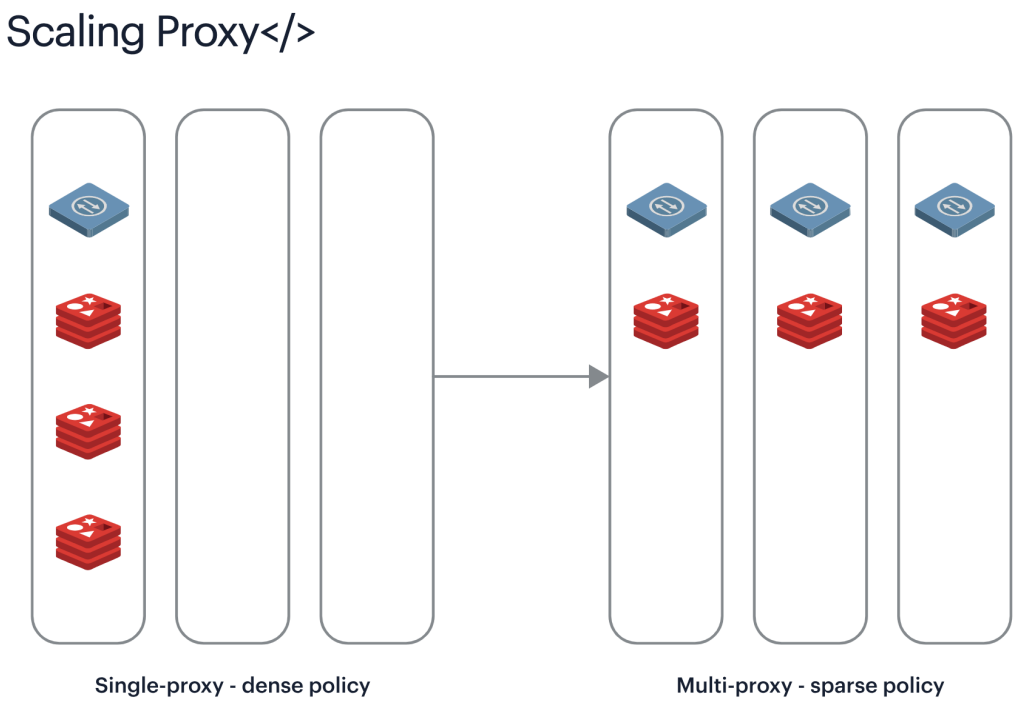
2. Scaling out by adding node(s) to the Redis Enterprise cluster, rebalancing and then resharding your database. This scenario is useful if more physical resources are

needed in your cluster in order to scale your database:



- By default, every database created over the Redis Enterprise cluster is operated with a single redundant proxy.
- Although the proxy is extremely efficient and can usually handle more than 1 million ops/sec when running on a modest cloud instance, there are some situations in which scaling the proxy is needed to deal with network bandwidth or packets-per-second limitations.
- Redis Enterprise allows you to scale the proxy across multiple nodes of the cluster without changing your application code.
- Redis clients can use a DNS round robin to select which proxy to work with every time a new connection is created.
- Furthermore, in order to optimize network traffic across the cluster nodes, Redis Enterprise selects a shard placement policy according to the proxy configuration.

- When a single proxy configuration is used, Redis Enterprise will try to put as many shards as it can on each node (this is also called a *dense* configuration). When a multi-proxy configuration is used, the shard placement policy for the cluster will be based on a *sparse* configuration.



True linear scalability with open source cluster API

Redis Enterprise supports the open source (OSS) cluster API to allow only one network hop between your client and the cluster, at any cluster size. This way, your clients are exposed to the actual placement of the shards on the cluster node using the OSS cluster protocol, as explained below:

Scaling with OSS cluster API

- From the point of view of the Redis client, each proxy represents a range of hash slots (HSs) equal to the sum of all the HSs served by the master shards hosted on the node of the proxy. In other words, from the point of view of the client, the proxy represents one big Redis instance with a master role.
- As in the OSS cluster, the client (a standard OSS client) sends requests to the relevant proxy based on the key of a key-value item and the Redis cluster hashing function.
- Once the request arrives to the proxy, it is forwarded to the relevant shard on the node by applying the same hashing function.
- In the case of a cluster topology change (i.e. shard migration, master/slave failover, etc.), the proxy updates the HSs it manages and, if necessary, redirects requests from the clients to another node in the cluster (using MOVED reply).

👉 When working with the OSS cluster API, Redis Enterprise is scaled in a true linear manner. The figure below summarizes the results of a series of benchmarks:

True Linear Scalability

6 nodes, 120 shards cluster

18 nodes, 360 shards cluster

Sub-millisecond latency is maintained across all the tests

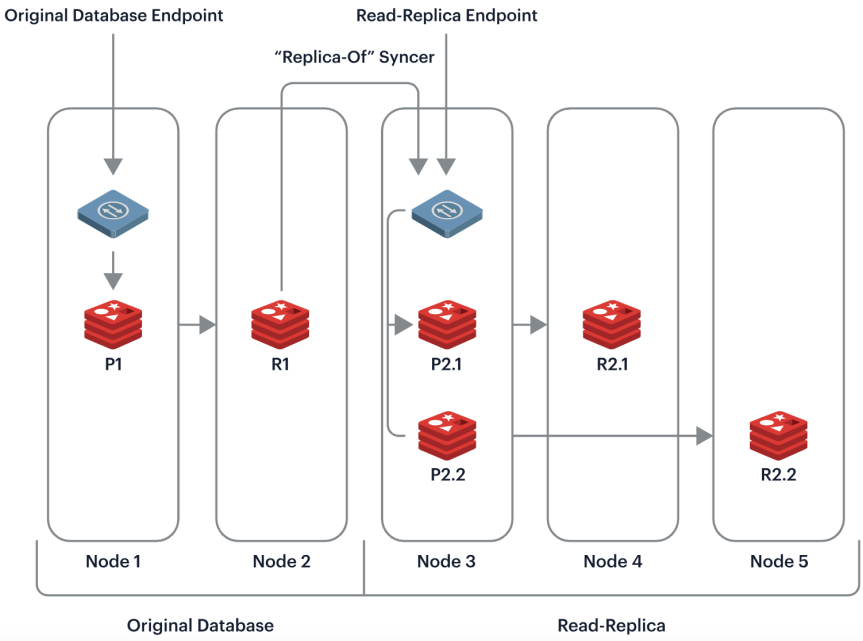
Scaling ‘read’ with read-replica using the ‘replica-of’ feature

Redis Enterprise allows you to scale your read operations by creating another database (on the same cluster) that will be served as a read-replica of your original database using the ‘replica-of’ feature.

Your read-replica is treated as a completely different database, and can be configured with a different number of shards and different availability or durability characteristics.

You can create multiple read-replicas, or just increase the number of shards in an existing read-replica in order to further scale your read operations.

Scaling ‘read’ with read-replica



Scaling shards and nodes

- The re-sharding mechanism of Redis Enterprise is based on Redis replication.
- Whenever a shard needs to be scaled out, the Redis Enterprise cluster launches another Redis instance and replicates to it half of the hash slots of the original shard
- Once the new shard is populated, traffic is sent to both shards by the proxy in a manner completely transparent to the application.

---

https://www.youtube.com/watch?v=m8HL1g9ak1k&t=37s

https://redis.com/redis-enterprise/technology/highly-available-redis/