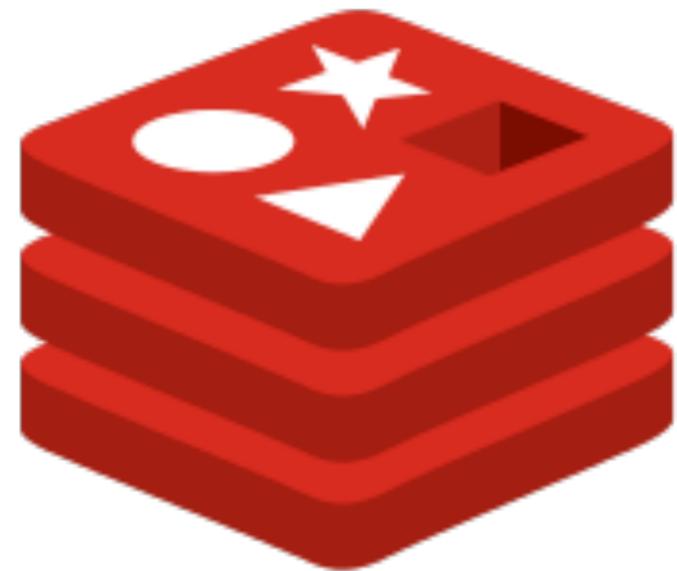


Intro to Redis

A Support Overview



redis

Matt Stancliff

Redis Engineer

NYC

mstancliff@gopivotal.com

Today

Today

What is Redis?

Today

What is Redis?

How does Redis work?

Today

What is Redis?

How does Redis work?

How do we configure Redis?

Today

What is Redis?

How does Redis work?

How do we configure Redis?

How do Redis commands work?

Today

What is Redis?

How does Redis work?

How do we configure Redis?

How do Redis commands work?

How do we manage Redis?

Today

What is Redis?

How does Redis work?

How do we configure Redis?

How do Redis commands work?

How do we manage Redis?

How does Redis break?

Today

What is Redis?

How does Redis work?

How do we configure Redis?

How do Redis commands work?

How do we manage Redis?

How does Redis break?

How do we fix a broken Redis?

What is Redis?

Big Picture Overview

“Disk is the new tape.”

“Memory is the new disk.”

“Memory is the new disk.”

Task: Read 10,000 database records concurrently from a web app

“Memory is the new disk.”

Task: Read 10,000 database records concurrently from a web app

Disk

“Memory is the new disk.”

Task: Read 10,000 database records concurrently from a web app

Disk

300 reads per second
(randomly located)

“Memory is the new disk.”

Task: Read 10,000 database records concurrently from a web app

Disk

300 reads per second
(randomly located)

(3 ms to 7 ms per read)

“Memory is the new disk.”

Task: Read 10,000 database records concurrently from a web app

Disk

300 reads per second
(randomly located)

(3 ms to 7 ms per read)

30 seconds to 70 seconds total

“Memory is the new disk.”

Task: Read 10,000 database records concurrently from a web app

Disk

300 reads per second
(randomly located)

(3 ms to 7 ms per read)

30 seconds to 70 seconds total

RAM

“Memory is the new disk.”

Task: Read 10,000 database records concurrently from a web app

Disk

300 reads per second
(randomly located)

(3 ms to 7 ms per read)

30 seconds to 70 seconds total

RAM

36,000,000 reads per second
(randomly located)

“Memory is the new disk.”

Task: Read 10,000 database records concurrently from a web app

Disk

300 reads per second
(randomly located)

(3 ms to 7 ms per read)

RAM

36,000,000 reads per second
(randomly located)

(20 ns to 60 ns per read)

30 seconds to 70 seconds total

“Memory is the new disk.”

Task: Read 10,000 database records concurrently from a web app

Disk

300 reads per second
(randomly located)

(3 ms to 7 ms per read)

30 seconds to 70 seconds total

RAM

36,000,000 reads per second
(randomly located)

(20 ns to 60 ns per read)

0.2 ms to 0.6 ms total

30 s vs 0.0002 s

Task: Read 10,000 database records concurrently from a web app

Disk

RAM

30 s vs 0.0002 s

Task: Read 10,000 database records concurrently from a web app

Disk

RAM

Using RAM-only, you get results

than disk-backed storage.

30 s vs 0.0002 s

Task: Read 10,000 database records concurrently from a web app

Disk

RAM

Using RAM-only, you get results

150,000x faster

than disk-backed storage.

30 s vs 0.0002 s

Task: Read 10,000 database records concurrently from a web app

Disk

RAM



Using RAM-only, you get results

150,000x faster

than disk-backed storage.

“Memory is [still] the new disk.”

Task: Read 10,000 database records concurrently from a web app

RAM

36,000,000 reads per second
(randomly located)

(20 ns to 60 ns per read)

0.2 ms to 0.6 ms total

“Memory is [still] the new disk.”

Task: Read 10,000 database records concurrently from a web app

SSD

RAM

36,000,000 reads per second
(randomly located)

(20 ns to 60 ns per read)

0.2 ms to 0.6 ms total

“Memory is [still] the new disk.”

Task: Read 10,000 database records concurrently from a web app

SSD

100,000 reads per second
(randomly located)

RAM

36,000,000 reads per second
(randomly located)

(20 ns to 60 ns per read)

0.2 ms to 0.6 ms total

“Memory is [still] the new disk.”

Task: Read 10,000 database records concurrently from a web app

SSD

100,000 reads per second
(randomly located)

(0.1 ms per read)

RAM

36,000,000 reads per second
(randomly located)

(20 ns to 60 ns per read)

0.2 ms to 0.6 ms total

“Memory is [still] the new disk.”

Task: Read 10,000 database records concurrently from a web app

SSD

100,000 reads per second
(randomly located)

(0.1 ms per read)

100 ms total

RAM

36,000,000 reads per second
(randomly located)

(20 ns to 60 ns per read)

0.2 ms to 0.6 ms total

“Memory is [still] the new disk.”

Task: Read 10,000 database records concurrently from a web app

SSD

100,000 reads per second
(randomly located)

(0.1 ms per read)

100 ms total

300x faster than disk

RAM

36,000,000 reads per second
(randomly located)

(20 ns to 60 ns per read)

0.2 ms to 0.6 ms total

“Memory is [still] the new disk.”

Task: Read 10,000 database records concurrently from a web app

SSD

100,000 reads per second
(randomly located)

300x faster than disk

500x slower than RAM

0.1 to 0.5 ms total

RAM

100,000 reads per second
(randomly located)

(20 ns to 60 ns per read)

0.2 ms to 0.6 ms total

disk vs. SSD vs. RAM recap

disk vs. SSD vs. RAM recap

disk

SSD

RAM

disk vs. SSD vs. RAM recap

disk

SSD

RAM

300

reads per second

disk vs. SSD vs. RAM recap

disk

300

reads per second

SSD

100,000

reads per second

RAM

disk vs. SSD vs. RAM recap

disk

300

reads per second

SSD

100,000

reads per second

RAM

36,000,000

reads per second

disk vs. SSD vs. RAM recap

disk

300

reads per second

SSD

100,000

reads per second

RAM

36,000,000

reads per second

300x faster than disk

disk vs. SSD vs. RAM recap

disk

300

reads per second

SSD

100,000

reads per second

RAM

36,000,000

reads per second

300x faster than disk

150,000x faster than disk

disk vs. SSD vs. RAM recap

disk

300

reads per second

SSD

100,000

reads per second

RAM

36,000,000

reads per second

300x faster than disk

150,000x faster than disk

500x faster than SSD

disk vs. SSD vs. RAM recap

disk

300

reads per second

SSD

100,000

reads per second

RAM

36,000,000

reads per second



300x faster than disk

150,000x faster than disk

500x faster than SSD

What does Redis do?

Redis is

Redis is

a **data structure** server

Redis is
a **data structure** server
with **replication**

Redis is
a **data structure server**
with **replication**
persistent storage

Redis is

a **data structure** server

with **replication**

persistent storage

async queues

Redis is

a **data structure** server

with **replication**

pub/sub

persistent storage

async queues

atomic scripting (transactions)

Redis is

a data structure server

with **replication**

pub/sub

persistent storage

async queues

atomic scripting (transactions)

clustering [soon]

Redis is

a data structure server

with **replication**

pub/sub

persistent storage

async queues

Redis is

Redis is

complementary infrastructure

Redis is

complementary infrastructure

use **with** PostgreSQL

Redis is

complementary infrastructure

use **with** PostgreSQL

MySQL

Redis is

complementary infrastructure

use **with** PostgreSQL

MySQL

Hive/Pig/HBase/Hadoop

structured data cache

Redis is

complementary infrastructure

use **with** PostgreSQL

MySQL

Hive/Pig/HBase/Hadoop

structured data cache queueing system

Redis is

complementary infrastructure

use **with** PostgreSQL

MySQL

Hive/Pig/HBase/Hadoop

structured data cache queueing system

pub/sub messaging bus

Redis is

complementary infrastructure

use **with** PostgreSQL

MySQL

Hive/Pig/HBase/Hadoop

Redis is

Redis is
standalone infrastructure

Redis is

standalone infrastructure

primary datastore for

Redis is

standalone infrastructure

primary datastore for

following/friending graphs

Redis is

standalone infrastructure

primary datastore for

following/friending graphs

timelines

Redis is

standalone infrastructure

primary datastore for

following/friending graphs

timelines

newsfeeds

ranking users (top scores)

Redis is

standalone infrastructure

primary datastore for

following/friending graphs

timelines

newsfeeds

ranking users (top scores)

low latency analytics

Redis is

standalone infrastructure

primary datastore for

following/friending graphs

timelines

newsfeeds

Redis is

Redis is
in-memory

Redis is
in-memory

all storage kept live in RAM

Redis is **in-memory**

all storage kept live in RAM

150,000x faster than HDDs

Redis is **in-memory**

all storage kept live in RAM

150,000x faster than HDs

500x faster than SSD

Redis is

Redis is
 durable

Redis is **durable**

tunable **commit-every-update-to-disk** options

Redis is **durable**

loads previous memory state from disk on startup

tunable **commit-every-update-to-disk** options

automatic copying to **multiple standby replicas**

Redis is
durable

loads previous memory state from disk on startup

tunable **commit-every-update-to-disk** options

automatic copying to **multiple standby replicas**

[soon] **clustering** data across dozens of hosts

Redis is
durable

loads previous memory state from disk on startup

tunable **commit-every-update-to-disk** options

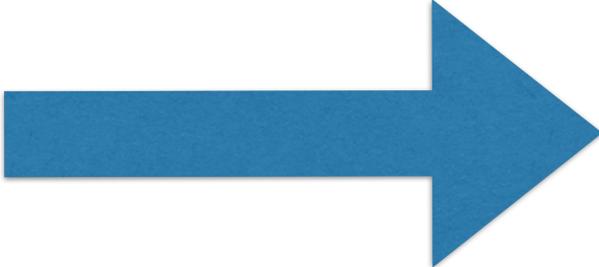
How does Redis work?

How does Redis work? Startup

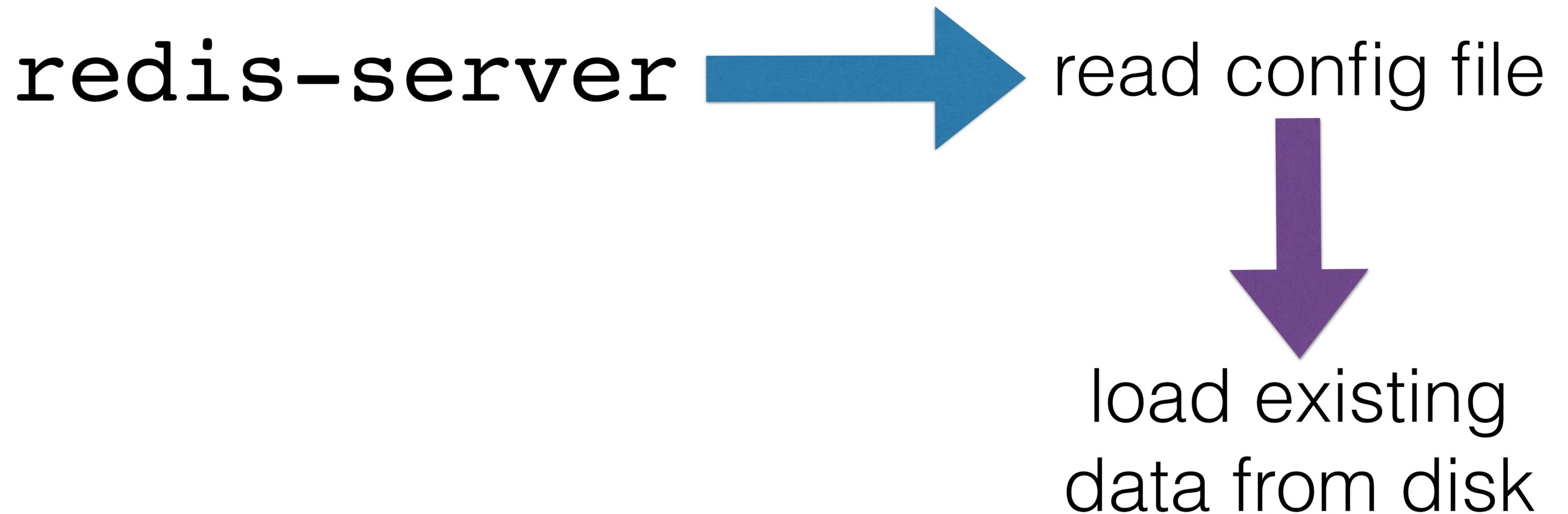
How does Redis work? Startup

```
redis-server
```

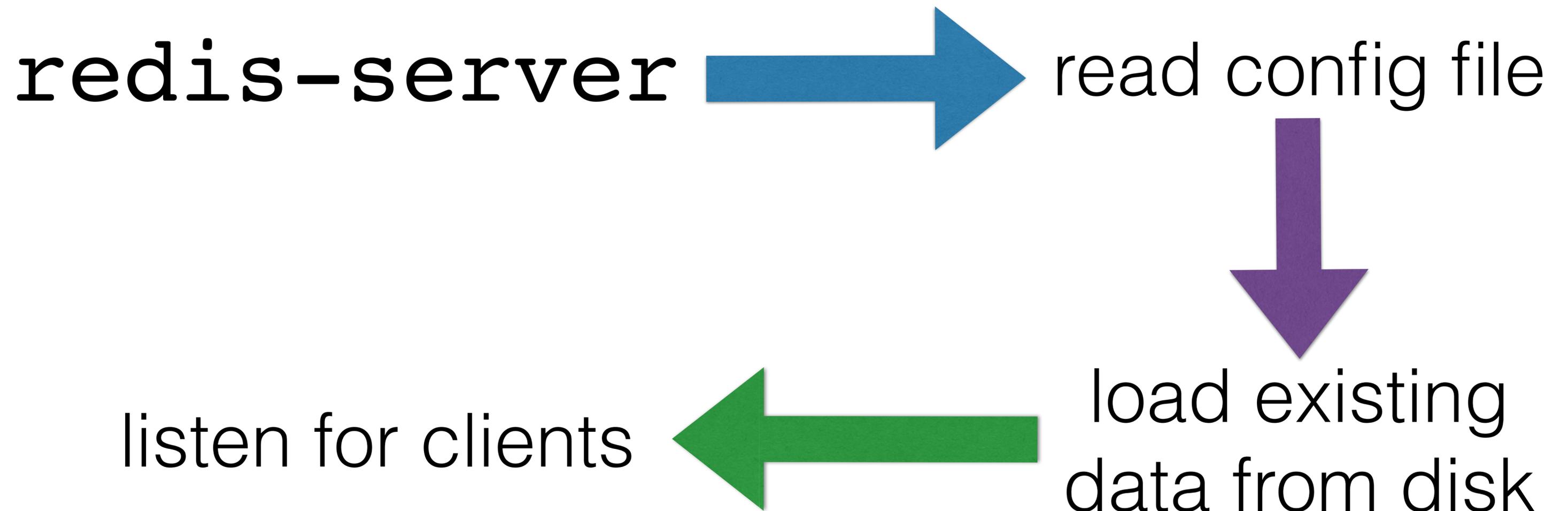
How does Redis work? Startup

`redis-server`  read config file

How does Redis work? Startup



How does Redis work? Startup



How does Redis work?

How does Redis work?
asynchronous

How does Redis work?

asynchronous

single-threaded

How does Redis work?

asynchronous

single-threaded

event loop

event loop

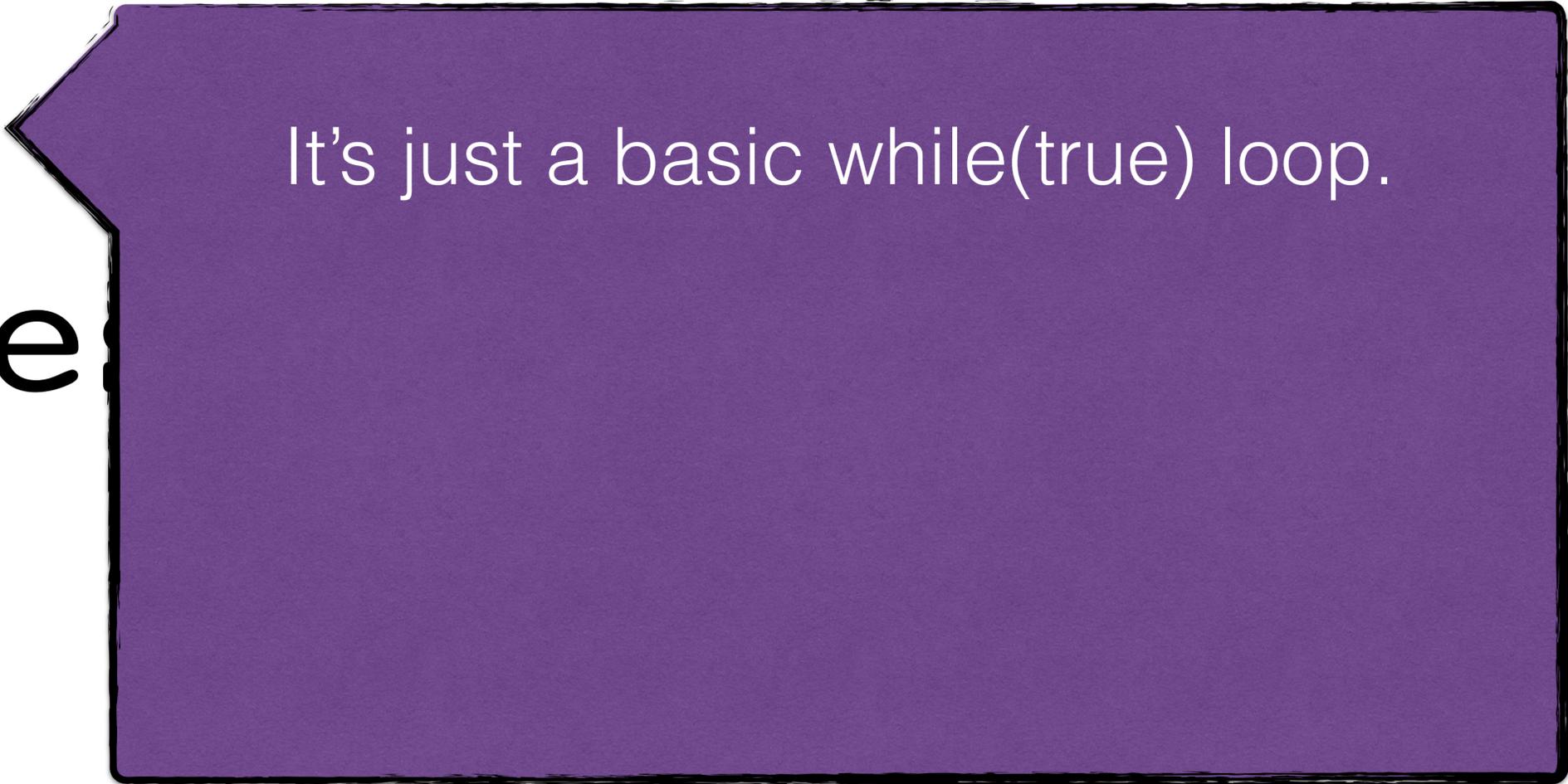
```
while ( !stopped ) {  
    processEvents ( ) ;  
}
```

event loop

```
while
```

```
  proces
```

```
}
```



It's just a basic while(true) loop.

event loop

```
while  
  proce  
}
```

It's just a basic `while(true)` loop.

`while(true)` runs 500 million iterations per second on my 2.6GHz laptop.

event loop

```
while
```

```
  process
```

```
}
```

It's just a basic `while(true)` loop.

`while(true)` runs 500 million iterations per second on my 2.6GHz laptop.

That's 2 nanoseconds per loop.

event loop

```
while (!stopped) {  
  processEvents();  
}
```

redis handles all operations in this loop.

event loop

```
while (!stopped) {  
    processEvents();  
}
```

redis handles all operations in this loop.
result:

event loop

```
while (!stopped) {  
    processEvents();  
}
```

redis handles all operations in this loop.

result:

redis is single threaded.

event loop

```
while (!stopped) {  
    processEvents();  
}
```

redis handles all operations in this loop.

result:

redis is single threaded.

redis uses only one core.

processEvents()

`processEvents()`

check for network events

`processEvents()`

check for network events

new clients (IPv4, IPv6, domain sockets)

processEvents ()

check for network events

new clients (IPv4, IPv6, domain sockets)

connected clients running commands

processEvents()

check for network events

- new clients (IPv4, IPv6, domain sockets)

- connected clients running commands

process scheduled events

processEvents()

check for network events

- new clients (IPv4, IPv6, domain sockets)

- connected clients running commands

process scheduled events

- 10 times per second:

processEvents ()

check for network events

- new clients (IPv4, IPv6, domain sockets)

- connected clients running commands

process scheduled events

- 10 times per second:

 - replication sanity check

processEvents ()

check for network events

- new clients (IPv4, IPv6, domain sockets)

- connected clients running commands

process scheduled events

- 10 times per second:

 - replication sanity check

 - force-expire keys

processEvents ()

check for network events

- new clients (IPv4, IPv6, domain sockets)

- connected clients running commands

process scheduled events

- 10 times per second:

 - replication sanity check

 - force-expire keys

 - persistence sanity check

processEvents()

check for network events

- new clients (IPv4, IPv6, domain sockets)

- connected clients running commands

process scheduled events

- 10 times per second:

 - replication sanity check

 - force-expire keys

 - persistence sanity check

 - ... and a dozen other things.

How does Redis work?
Data

How does Redis work?
Data

Redis stores **all** data in memory

How does Redis work? Data

Redis stores **all** data in memory

optional backup-to-disk settings for:

How does Redis work?

Data

Redis stores **all** data in memory

optional backup-to-disk settings for:

complete DB snapshot to disk

How does Redis work? Data

Redis stores **all** data in memory

optional backup-to-disk settings for:

complete DB snapshot to disk

writes entire dataset to disk

How does Redis work?

Data

Redis stores **all** data in memory

optional backup-to-disk settings for:

append-only journal file

complete DB snapshot to disk

writes entire dataset to disk

How does Redis work? Data

Redis stores **all** data in memory

optional backup-to-disk settings for:

append-only journal file

every change appended to a file

(like binlogs)

complete DB snapshot to disk

writes entire dataset to disk

How does Redis work? Data

Redis stores **all** data in memory

optional backup-to-disk settings for:

append-only journal file

every change appended to a file

(like binlogs)

complete DB snapshot to disk

writes entire dataset to disk

more compact than append-only file

Redis is **single-threaded**

Redis is **single-threaded**

(except for this)

Redis is **single-threaded**

(except for this)

append-only journal:

Redis is **single-threaded**

(except for this)

append-only journal:

written using a background thread

Redis is **single-threaded**

(except for this)

append-only journal:

written using a background thread

(the only thread in Redis)

Redis is **single-threaded**

(except for this)

append-only journal:

written using a background thread

(the only thread in Redis)

complete DB snapshot:

Redis is **single-threaded**

(except for this)

append-only journal:

written using a background thread

(the only thread in Redis)

complete DB snapshot:

forks a new `redis-server` process

Redis is **single-threaded**

(except for this)

append-only journal:

written using a background thread

(the only thread in Redis)

complete DB snapshot:

forks a new `redis-server` process

serializes the child's frozen/"snapshot" memory to disk

Redis is **single-threaded**

(except for this)

append-only journal:

written using a background thread

(the only thread in Redis)

complete DB snapshot:

forks a new `redis-server` process

serializes the child's frozen/"snapshot" memory to disk

can cause performance hiccups on EC2 or large data sets

Redis is **single-threaded**

Redis is **single-threaded**

advantage:

Redis is **single-threaded**

advantage:

no locks

Redis is **single-threaded**

advantage:

- no locks

- your command is the only one running

Redis is **single-threaded**

advantage:

- no locks

- your command is the only one running

disadvantage:

Redis is **single-threaded**

advantage:

- no locks

- your command is the only one running

disadvantage:

- your command is the only one running

Redis is **single-threaded**

advantage:

- no locks

- your command is the only one running

disadvantage:

- your command is the only one running

- “bad” commands can block the server for seconds

Redis is **single-threaded**

advantage:

- no locks

- your command is the only one running

disadvantage:

- your command is the only one running

- “bad” commands can block the server for seconds

- poorly designed in-server scripts can block forever

How does Redis
configure?

redis.conf

redis.conf

plain text file

redis.conf

plain text file [name] [value]

redis.conf

plain text file

[name] [value]

sample entries:

redis.conf

plain text file [name] [value]

sample entries:

```
# Accept connections on the specified port, default is 6379.  
# If port 0 is specified Redis will not listen on a TCP socket.  
port 6379
```

redis.conf

plain text file [name] [value]

sample entries:

```
# Accept connections on the specified port, default is 6379.  
# If port 0 is specified Redis will not listen on a TCP socket.  
port 6379  
save 900 1  
save 300 10  
save 60 10000
```

redis.conf

plain text file

[name] [value]

sample entries:

```
# Accept connections on the specified port, default is 6379.  
# If port 0 is specified Redis will not listen on a TCP socket.
```

```
port 6379
```

```
save 900 1
```

```
save 300 10
```

```
save 60 10000
```

```
# The filename where to dump the DB
```

```
dbfilename dump.rdb
```

Top 5 Config Settings

Network

Top 5 Config Settings

Network

```
port 6379
```

Top 5 Config Settings

Network

```
bind ::1
```

Top 5 Config Settings

Network

```
unixsocketperm 755
```

Top 5 Config Settings

Persistence

Top 5 Config Settings

Persistence

```
# Save DB if at least <changes> happen in <seconds>:  
#  
#   save <seconds> <changes>
```

Top 5 Config Settings

Persistence

```
# Save DB if at least <changes> happen in <seconds>:  
#  
#   save <seconds> <changes>  
save 900 1  
save 300 10  
save 60 10000
```

Top 5 Config Settings

Persistence

```
# Save DB if at least <changes> happen in <seconds>:  
#  
#   save <seconds> <changes>  
save 900 1  
save 300 10  
save 60 10000
```

```
# Directory path to store DB, AOF, and  
# replication/cluster metadata.  
# Redis instances *must not* share directories.  
dir ./
```

Top 5 Config Settings

More Persistence

Top 5 Config Settings

More Persistence

`appendonly yes`

Top 5 Config Settings

More Persistence

```
appendonly yes
```

```
# appendfsync always  
appendfsync everysec  
# appendfsync no
```

Top 5 Config Settings

More Persistence

```
appendonly yes
```

```
# appendfsync always  
appendfsync everysec  
# appendfsync no
```

```
# Automatically rewrite the log file implicitly calling  
# BGREWRITEAOF when the AOF size grows by percentage.  
auto-aof-rewrite-percentage 100  
auto-aof-rewrite-min-size 64mb
```

Top 5 Config Settings

Memory Limits

Top 5 Config Settings

Memory Limits

```
# Don't use more memory than the specified amount of bytes.  
# When the memory limit is reached Redis will try to remove keys  
# accordingly to the eviction policy selected (see maxmemory-policy).
```

Top 5 Config Settings

Memory Limits

```
# Don't use more memory than the specified amount of bytes.  
# When the memory limit is reached Redis will try to remove keys  
# accordingly to the eviction policy selected (see maxmemory-policy).
```

```
maxmemory <bytes>
```

Top 5 Config Settings

More Memory Limits

Top 5 Config Settings

More Memory Limits

```
# volatile-lru -> remove the key with an expire set using an LRU algorithm
```

Top 5 Config Settings

More Memory Limits

```
# allkeys-lru -> remove any key accordingly to the LRU algorithm
```

Top 5 Config Settings

More Memory Limits

```
# volatile-random -> remove a random key with an expire set
```

Top 5 Config Settings

More Memory Limits

```
# allkeys-random -> remove a random key, any key
```

Top 5 Config Settings

More Memory Limits

```
# volatile-ttl -> remove the key with the nearest expire time (minor TTL)
```

Top 5 Config Settings

More Memory Limits

```
# noeviction -> don't expire at all, just return an error on write operations
```

Top 5 Config Settings

More Memory Limits



safe

```
# noeviction -> don't expire at all, just return an error on write operations
```

Top 5 Config Settings

More Memory Limits

safe

```
# noeviction -> don't expire at all, just return an error on write operations
```

also safe

Top 5 Config Settings

More Memory Limits

safe

weird

also safe

function -> expire at all, just return an error on write operations

Top 5 Config Settings

More Memory Limits

safe

weird

kinda dumb

also safe

ction -> expire at all, just return an error on write operations

Top 5 Config Settings

More Memory Limits

“pretend memcache”

safe

weird

kinda dumb

also safe

function -> expire at all, just return an error on write operations

Top 5 Config Settings

More Memory Limits

“pretend memcache”

safe

weird

kinda dumb

also safe

action -> expire at all, just return an error on write operations

```
maxmemory-policy volatile-lru
```

Top 5 Config Settings

Replication

Top 5 Config Settings

Replication

```
# Use slaveof to make a Redis instance a copy of another Redis server.
```

Top 5 Config Settings

Replication

```
slaveof <masterip> <masterport>
```

Top 5 Config Settings

Replication

```
slaveof <masterip> <masterport>
```

```
slave-serve-stale-data yes
```

Top 5 Config Settings

Replication

```
slaveof <masterip> <masterport>
```

serves data while
disconnected from master *and*
while a sync is in progress

```
slave-serve-stale-data yes
```

Top 5 Config Settings

Replication

```
slaveof <masterip> <masterport>
```

serves data while disconnected from master *and* while a sync is in progress

```
slave-serve-stale-data yes
```

(may have only partial replica of master)

Top 5 Config Settings

Replication

```
slaveof <masterip> <masterport>
```

serves data while disconnected from master *and* while a sync is in progress

```
slave-serve-stale-data yes
```

(may have only partial replica of master)

```
slave-read-only yes
```

Top 5 Config Settings

Replication

```
slaveof <masterip> <masterport>
```

serves data while disconnected from master *and* while a sync is in progress

```
slave-serve-stale-data yes
```

(may have only partial replica of master)

```
slave-read-only yes
```

```
min-slaves-to-write 3  
min-slaves-max-lag 10
```

Top 5 Config Settings

Replication

```
slaveof <masterip> <masterport>
```

serves data while disconnected from master *and* while a sync is in progress

```
slave-serve-stale-data yes
```

(may have only partial replica of master)

```
slave-read-only yes
```

```
min-slaves-to-write 3  
min-slaves-max-lag 10
```

for reliable installations deny writes if fewer than 3 in-sync replicas are live

Top 5 Config Settings

Cluster

Top 5 Config Settings Cluster

```
# Normal Redis instances can't be part of a Redis Cluster; only nodes that are  
# started as cluster nodes can. In order to start a Redis instance as a  
# cluster node enable the cluster support uncommenting the following:
```

```
cluster-enabled yes
```

Top 5 Config Settings Cluster

```
# Normal Redis instances can't be part of a Redis Cluster; only nodes that are  
# started as cluster nodes can. In order to start a Redis instance as a  
# cluster node enable the cluster support uncommenting the following:
```

```
cluster-enabled yes
```

```
cluster-config-file node-6379.conf
```

Top 5 Config Settings

Cluster

```
# Normal Redis instances can't be part of a Redis Cluster; only nodes that are  
# started as cluster nodes can. In order to run Redis as a  
# cluster node enable the cluster support by setting:  
cluster-enabled yes
```

must be a unique file per server.

```
cluster-config-file node-6379.conf
```

Top 5 Config Settings

Cluster

```
# Normal Redis instances can't be part of a Redis Cluster; only nodes that are  
# started as cluster nodes can. In order to run Redis as a  
# cluster node enable the cluster support by setting:  
cluster-enabled yes
```

cluster-config-file must be a unique file per server.

Only the server writes to cluster-config-file.

```
cluster-config-file node-6379.conf
```

Top 5 Config Settings

Cluster

```
# Normal Redis instances can't be part of a Redis Cluster; only nodes that are  
# started as cluster nodes can. In order to run Redis as a  
# cluster node enable the cluster support by setting:  
cluster-enabled yes
```

cluster-config-file must be a unique file per server.

Only the server writes to `cluster-config-file`.

(persists cluster membership information
across restarts)

```
cluster-config-file node-6379.conf
```

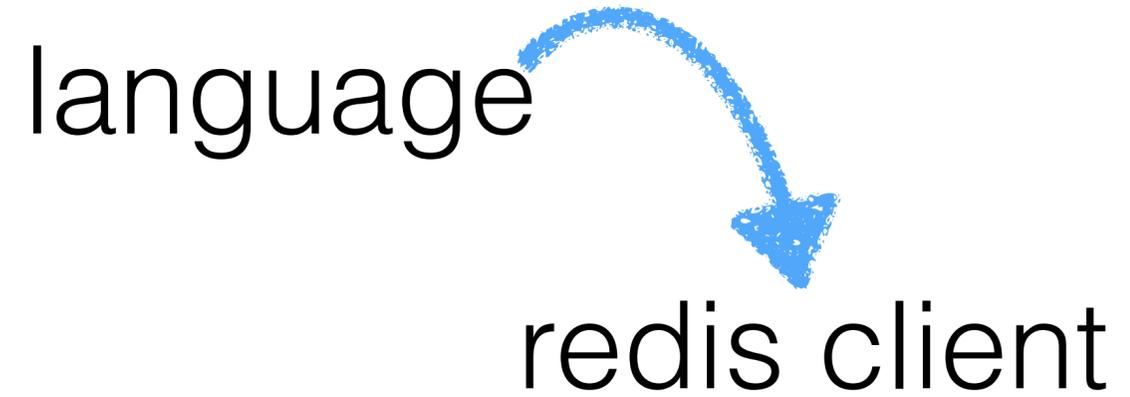
How Redis program?

Programmer's View of Redis

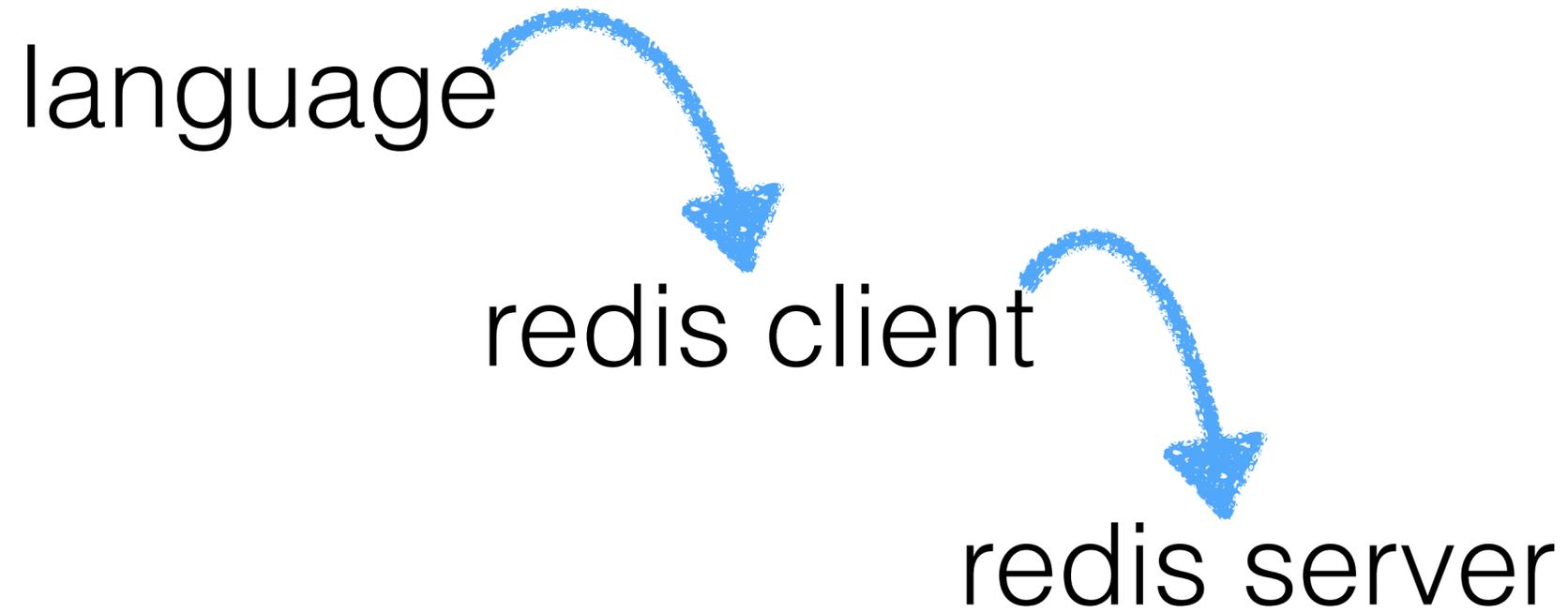
Programmer's View of Redis

language

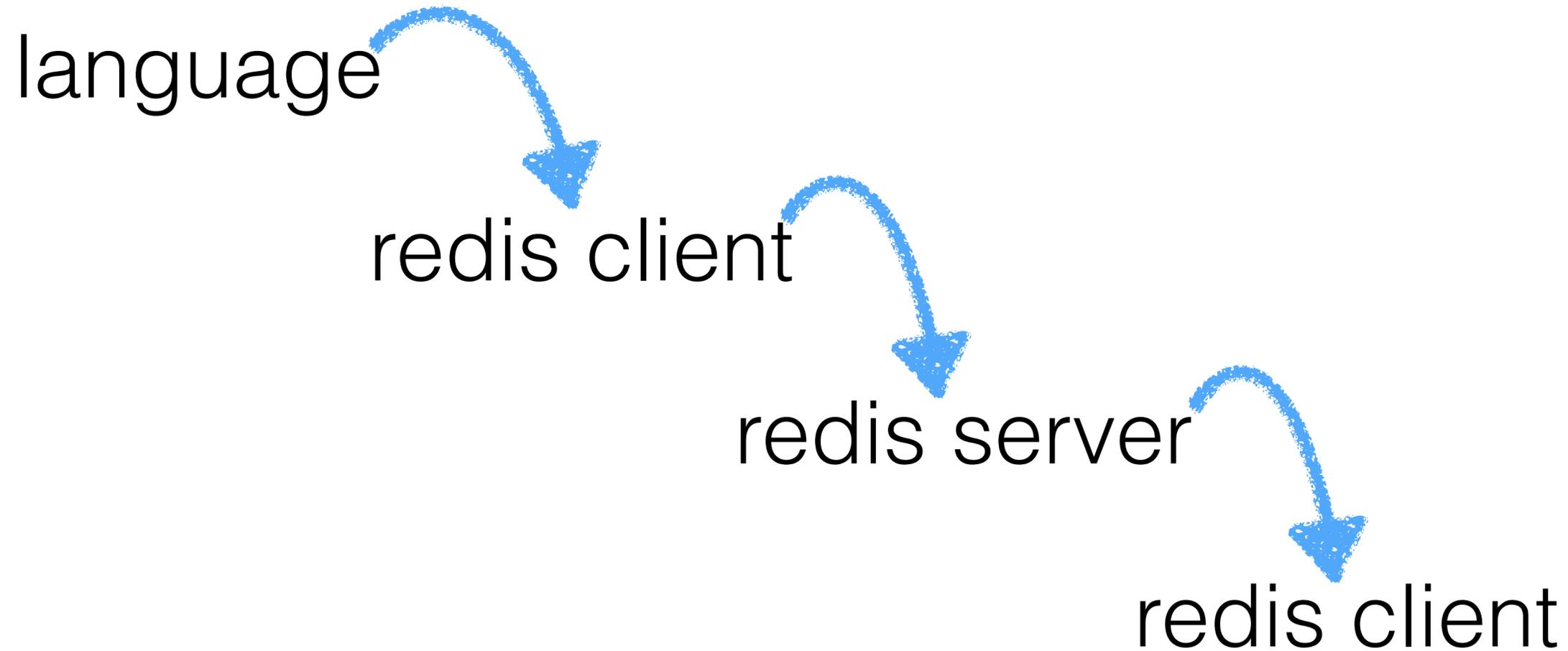
Programmer's View of Redis



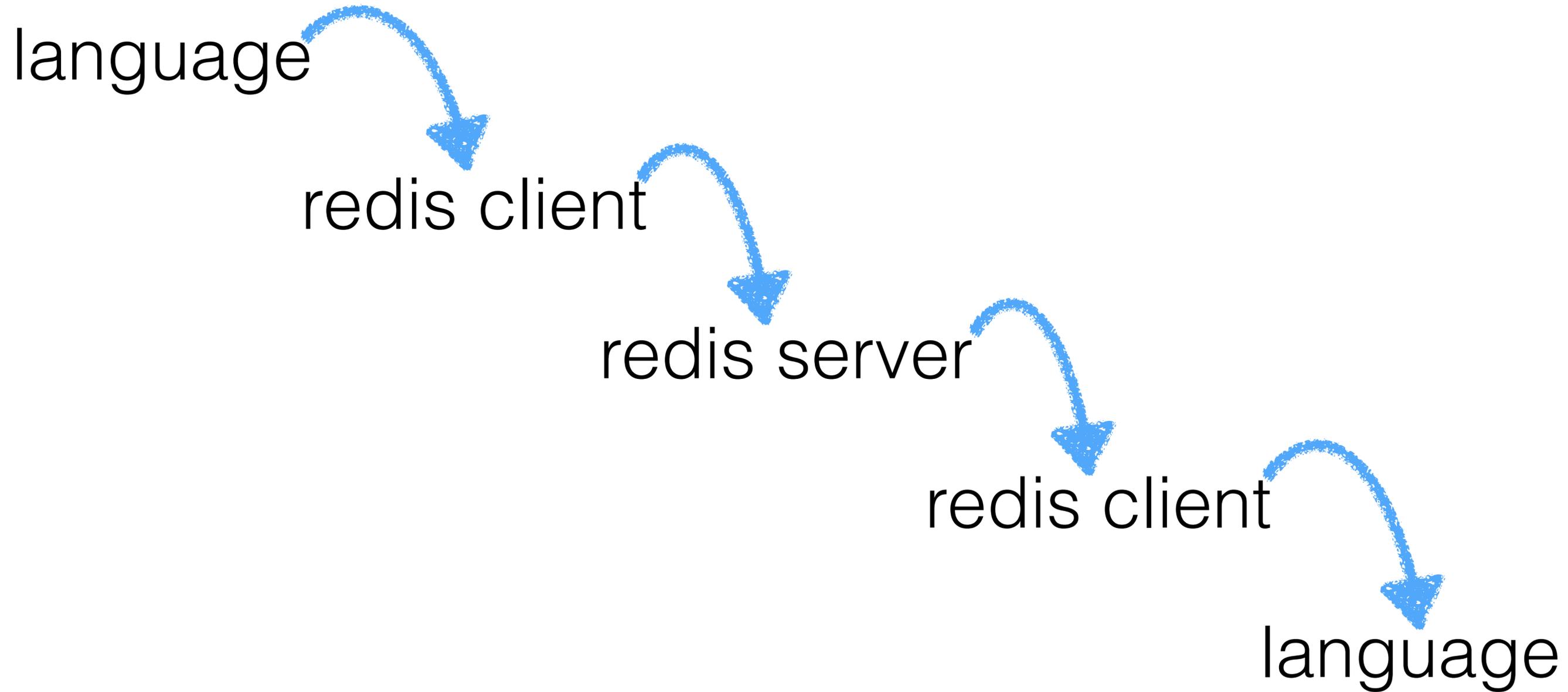
Programmer's View of Redis



Programmer's View of Redis



Programmer's View of Redis



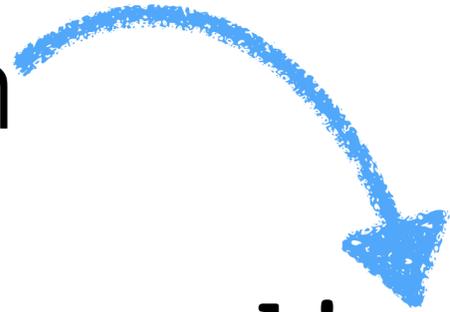
Programmer's View of Redis

Programmer's View of Redis

python

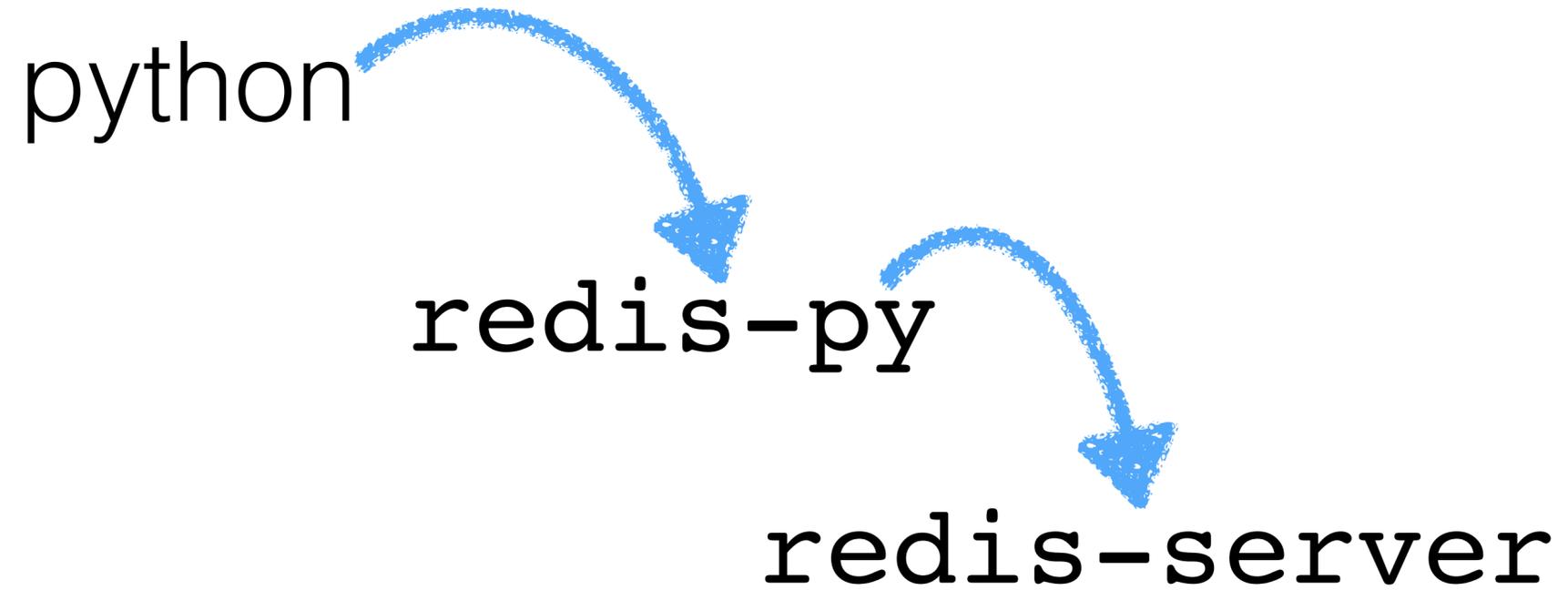
Programmer's View of Redis

python

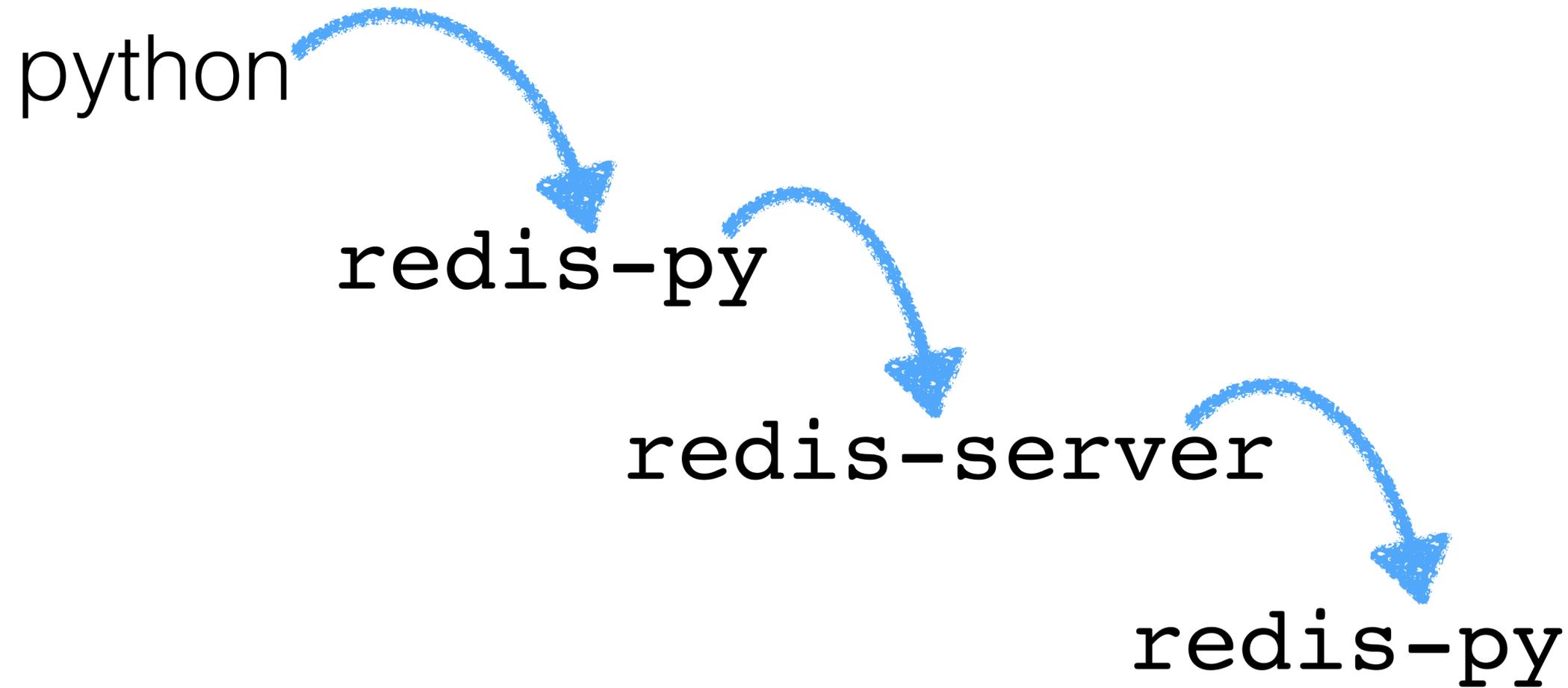


redis-py

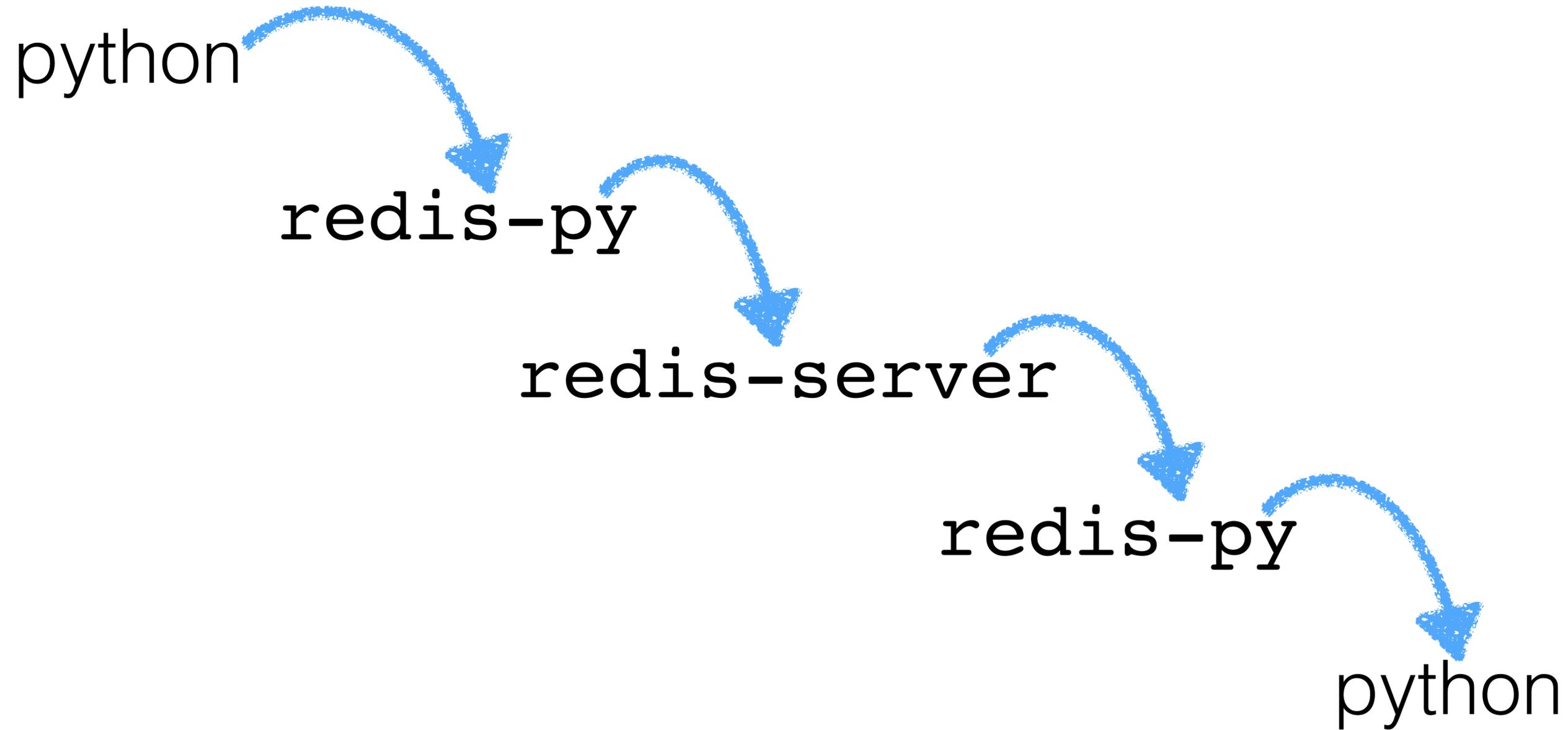
Programmer's View of Redis



Programmer's View of Redis



Programmer's View of Redis



Programmer's View of Redis

Two Types of Clients

Programmer's View of Redis

Two Types of Clients



Programmer's View of Redis

Two Types of Clients

every-command-is-a-function



Programmer's View of Redis

Two Types of Clients

every-command-is-a-function

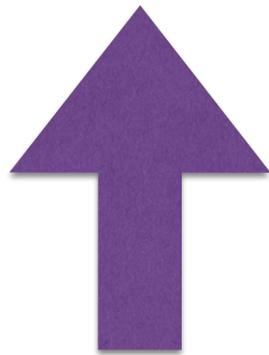
```
r = redis.Redis()  
r.set('bing', 'baz')
```

Programmer's View of Redis

Two Types of Clients

every-command-is-a-function

```
r = redis.Redis()  
r.set('bing', 'baz')
```



Programmer's View of Redis

Two Types of Clients

every-command-is-a-function

```
r = redis.Redis()  
r.set('bing', 'baz')
```



cleaner, has language feel



Programmer's View of Redis

Two Types of Clients

every-command-is-a-function

```
r = redis.Redis()  
r.set('bing', 'baz')
```



cleaner, has language feel

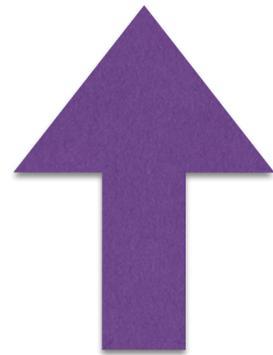
free command entry fields

Programmer's View of Redis

Two Types of Clients

every-command-is-a-function

```
r = redis.Redis()  
r.set('bing', 'baz')
```



cleaner, has language feel

free command entry fields

```
r = redis.Redis()  
r.cmd("SET bing baz")
```

Programmer's View of Redis

Two Types of Clients

every-command-is-a-function

```
r = redis.Redis()  
r.set('bing', 'baz')
```



cleaner, has language feel

free command entry fields

```
r = redis.Redis()  
r.cmd("SET bing baz")
```



Programmer's View of Redis

Two Types of Clients

every-command-is-a-function

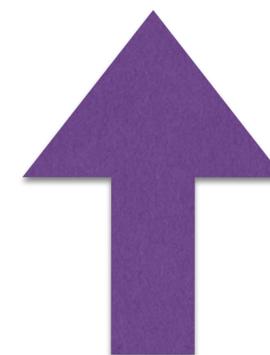
```
r = redis.Redis()  
r.set('bing', 'baz')
```



cleaner, has language feel

free command entry fields

```
r = redis.Redis()  
r.cmd("SET bing baz")
```



more flexible,
supports arbitrary commands

Programmer's View of Redis

Two Types of Clients

every-command-is-a-function

```
r = redis.Redis()  
r.set('bing', 'baz')
```



cleaner, has language feel

free command entry fields

```
r = redis.Redis()  
r.cmd("SET bing baz")
```



more flexible,
supports arbitrary commands

Programmer's View of Redis

Two Types of Clients

every-command-is-a-function

free command entry fields

Programmer's View of Redis

Two Types of Clients

every-command-is-a-function

```
r = redis.Redis()  
r.set('w', 34.3)  
c = r.incrbyfloat('w', 0.1)
```

free command entry fields

Programmer's View of Redis

Two Types of Clients

every-command-is-a-function

```
r = redis.Redis()  
r.set('w', 34.3)  
c = r.incrbyfloat('w', 0.1)
```



free command entry fields

Programmer's View of Redis

Two Types of Clients

every-command-is-a-function

```
r = redis.Redis()  
r.set('w', 34.3)  
c = r.incrbyfloat('w', 0.1)
```



c is now the **float** 34.4

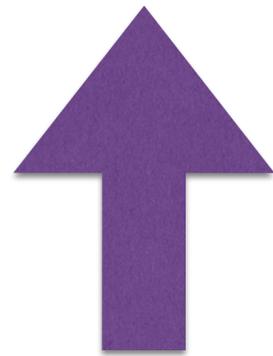
free command entry fields

Programmer's View of Redis

Two Types of Clients

every-command-is-a-function

```
r = redis.Redis()  
r.set('w', 34.3)  
c = r.incrbyfloat('w', 0.1)
```



c is now the **float** 34.4

free command entry fields

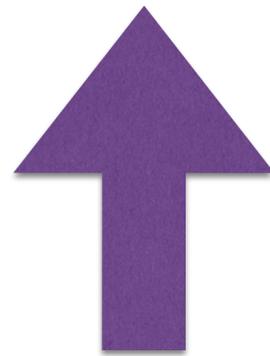
```
r = redis.Redis()  
r.cmd("SET w 34.3")  
c = r.cmd("INCRBYFLOAT w 0.1")
```

Programmer's View of Redis

Two Types of Clients

every-command-is-a-function

```
r = redis.Redis()  
r.set('w', 34.3)  
c = r.incrbyfloat('w', 0.1)
```



c is now the **float** 34.4

free command entry fields

```
r = redis.Redis()  
r.cmd("SET w 34.3")  
c = r.cmd("INCRBYFLOAT w 0.1")
```

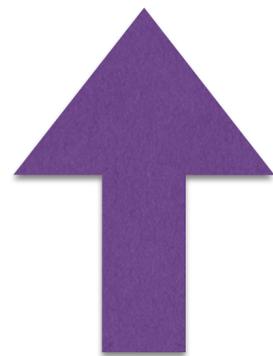


Programmer's View of Redis

Two Types of Clients

every-command-is-a-function

```
r = redis.Redis()  
r.set('w', 34.3)  
c = r.incrbyfloat('w', 0.1)
```



c is now the **float** 34.4

free command entry fields

```
r = redis.Redis()  
r.cmd("SET w 34.3")  
c = r.cmd("INCRBYFLOAT w 0.1")
```



depending on the client,
c is now the **string** "34.4"

- or -

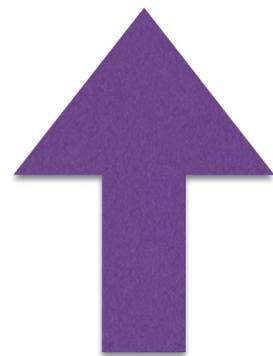
c is now the **float** 34.4

Programmer's View of Redis

Two Types of Clients

every-command-is-a-function

```
r = redis.Redis()  
r.set('w', 34.3)  
c = r.incrbyfloat('w', 0.1)
```



c is now the **float** 34.4

free command entry fields

```
r = redis.Redis()  
r.cmd("SET w 34.3")  
c = r.cmd("INCRBYFLOAT w 0.1")
```



depending on the client,
c is now the **string** "34.4"

- or -

c is now the **float** 34.4

Programmer's View of Redis

Two Types of Clients

every-command-is-a-function

free command entry fields

Programmer's View of Redis

Two Types of Clients

every-command-is-a-function

```
r = redis.Redis()  
r.set('w', 34.3)  
r.incrbyfloat('w', 0.1)  
c = r.get('w')
```

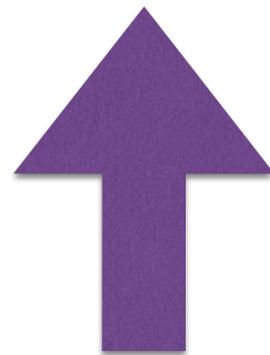
free command entry fields

Programmer's View of Redis

Two Types of Clients

every-command-is-a-function

```
r = redis.Redis()  
r.set('w', 34.3)  
r.incrbyfloat('w', 0.1)  
c = r.get('w')
```



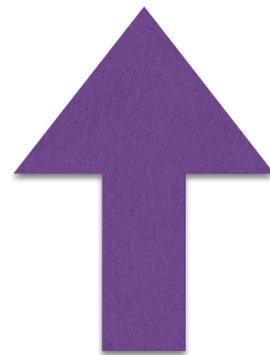
free command entry fields

Programmer's View of Redis

Two Types of Clients

every-command-is-a-function

```
r = redis.Redis()  
r.set('w', 34.3)  
r.incrbyfloat('w', 0.1)  
c = r.get('w')
```



c is now the **string** 34.4

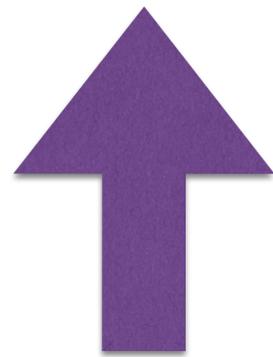
free command entry fields

Programmer's View of Redis

Two Types of Clients

every-command-is-a-function

```
r = redis.Redis()  
r.set('w', 34.3)  
r.incrbyfloat('w', 0.1)  
c = r.get('w')
```



c is now the **string** 34.4

free command entry fields

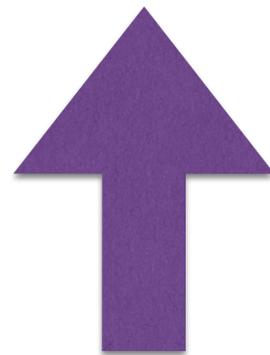
```
r = redis.Redis()  
r.cmd("SET w 34.3")  
r.cmd("INCRBYFLOAT w 0.1")  
c = r.cmd("GET w")
```

Programmer's View of Redis

Two Types of Clients

every-command-is-a-function

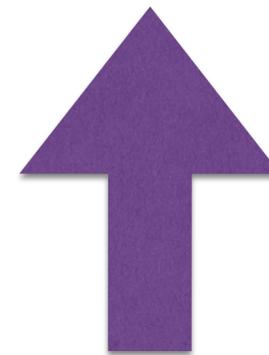
```
r = redis.Redis()  
r.set('w', 34.3)  
r.incrbyfloat('w', 0.1)  
c = r.get('w')
```



c is now the **string** 34.4

free command entry fields

```
r = redis.Redis()  
r.cmd("SET w 34.3")  
r.cmd("INCRBYFLOAT w 0.1")  
c = r.cmd("GET w")
```

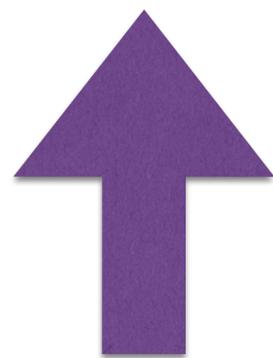


Programmer's View of Redis

Two Types of Clients

every-command-is-a-function

```
r = redis.Redis()  
r.set('w', 34.3)  
r.incrbyfloat('w', 0.1)  
c = r.get('w')
```



c is now the **string** 34.4

free command entry fields

```
r = redis.Redis()  
r.cmd("SET w 34.3")  
r.cmd("INCRBYFLOAT w 0.1")  
c = r.cmd("GET w")
```



c is now the **string** 34.4

Programmer's View of Redis

Clients

<http://redis.io/clients>

C: hiredis

PHP: Predis

Java: Jedis

Python: redis-py

Perl: Redis

Ruby: redis-rb

Redis commands aren't always **simple**

```
SORT key [BY pattern] [LIMIT offset count] [GET pattern  
[GET pattern ...]] [ASC|DESC] [ALPHA] [STORE  
destination]
```

```
SORT key [BY pattern] [LIMIT offset count] [GET pattern  
[GET pattern ...]] [ASC|DESC] [ALPHA] [STORE  
destination]
```

can be represented in languages with optional parameters:
`sort(name, start=None, num=None, by=None,
get=None, desc=False, alpha=False, store=None)`

```
MGET key1 key2 key3 key4 ...
```



multi-get

```
MGET key1 key2 key3 key4 ...
```

```
MSET key1 val1 key2 val2 key3 val3 key4 val4
```

```
MSET key1 val1 key2 val2 key3 val3 key4 val4
```



multi-set

```
SET key value [EX seconds] [PX milliseconds] [NX|XX]
```

```
SET key value [EX seconds] [PX milliseconds] [NX|XX]
```



optional
arguments

Redis Command Types

sorted sets

Redis Command Types

sets

sorted sets

Redis Command Types

sets

sorted sets

hashes

Redis Command Types

sets

sorted sets

hashes

Redis Command Types

lists

sets

sorted sets

hashes

Redis Command Types

strings

lists

sets

key
management

sorted sets

hashes

Redis Command Types

strings

lists

sets

key
management

sorted sets

hashes

server
management

Redis Command Types

strings

lists

sets

key
management

sorted sets

hashes

server
management

scripting

Redis Command Types

strings

lists

sets

key
management

sorted sets

hashes

server
management

scripting

Redis Command Types

strings

pub/sub

lists

sets

key
management

sorted sets

hashes

server
management

scripting

Redis Command Types

strings

pub/sub

lists

transactions

sets

key
management

sorted sets

hashes

server
management

scripting

Redis Command Types

strings

pub/sub

lists

transactions

Redis Data Types

Basics

Redis Data Types

Basics

key = *value*

Redis Data Types

Basics

key = value

keys are strings

strings are binary safe

strings have a max size of 512 MB

Redis Data Types

Basics

key = value

keys are strings

strings are binary safe

strings have a max size of 512 MB

values have a type:

string, list, hash, set, or sorted set

Redis Data Types

Basics

Quick Note:

Redis Data Types

Basics

Quick Note:

<http://redis.io/> embeds
live redis sessions

Redis Data Types

Strings

Redis Data Types

Strings

```
127.0.0.1:6379> SET location:kitten "in a tree"
```

Redis Data Types

Strings

OK

Redis Data Types

Strings

```
127.0.0.1:6379> GET location:kitten
```

Redis Data Types

Strings

```
"in a tree"
```

Redis Data Types

Strings

```
127.0.0.1:6379> STRLEN location:kitten
```

Redis Data Types

Strings

```
(integer) 9
```

Redis Data Types

More Strings

Redis Data Types

More Strings

```
127.0.0.1:6379> APPEND location:kitten " in the park"
```

Redis Data Types

More Strings

```
(integer) 21
```

Redis Data Types

More Strings

```
127.0.0.1:6379> GET location:kitten
```

Redis Data Types

More Strings

```
"in a tree in the park"
```

Redis Data Types

More Strings

```
127.0.0.1:6379> SETRANGE location:kitten 17 mall
```

Redis Data Types

More Strings

```
(integer) 21
```

Redis Data Types

More Strings

```
127.0.0.1:6379> GET location:kitten
```

Redis Data Types

More Strings

```
"in a tree in the mall"
```

Redis Data Types

Even More Strings

Redis Data Types

Even More Strings

BITCOUNT

GETBIT

SETBIT

BITOP

SETEX

PSETEX

INCR

INCRBY

INCRBYFLOAT

DECR

DECRBY

GETSET

SETNX

MSETNX

MGET

MSET

Redis Data Types

Lists

Redis Data Types

Lists

```
LPUSH alist hello
```

Redis Data Types

Lists

```
LPUSH alist hello
```

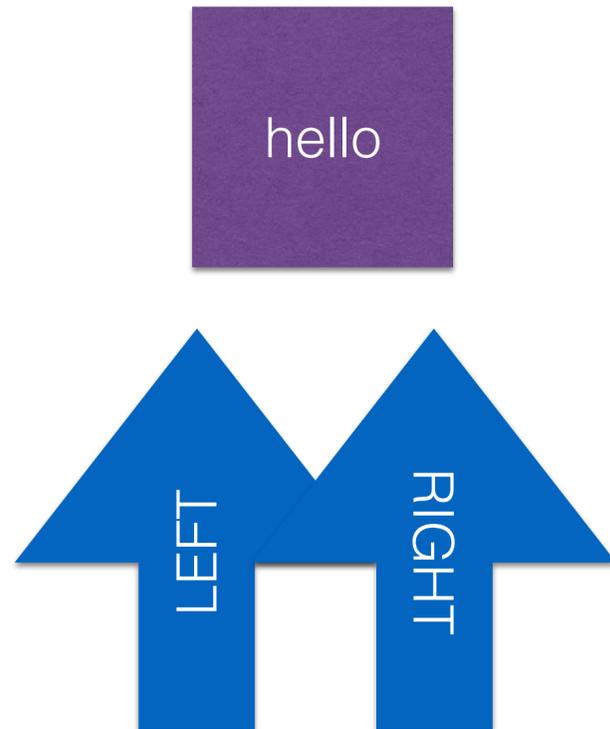


hello

Redis Data Types

Lists

```
LPUSH alist hello
```

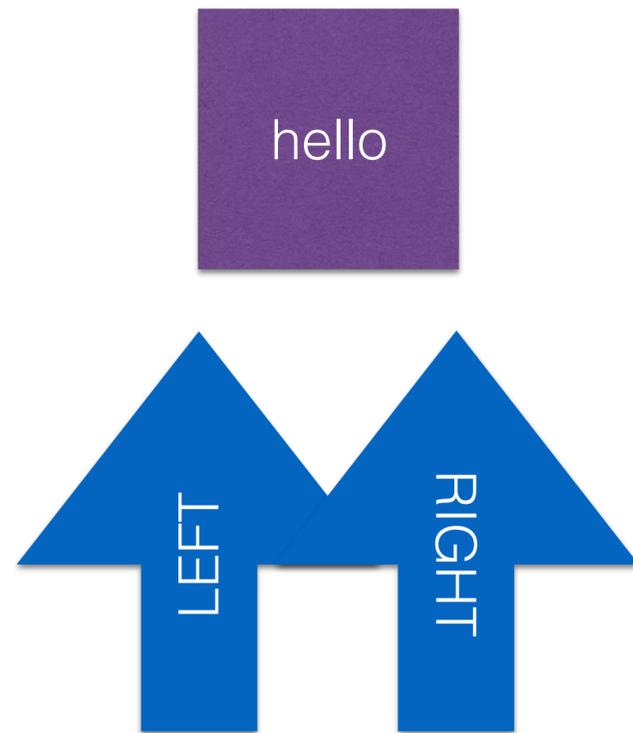


Redis Data Types

Lists

Redis Data Types

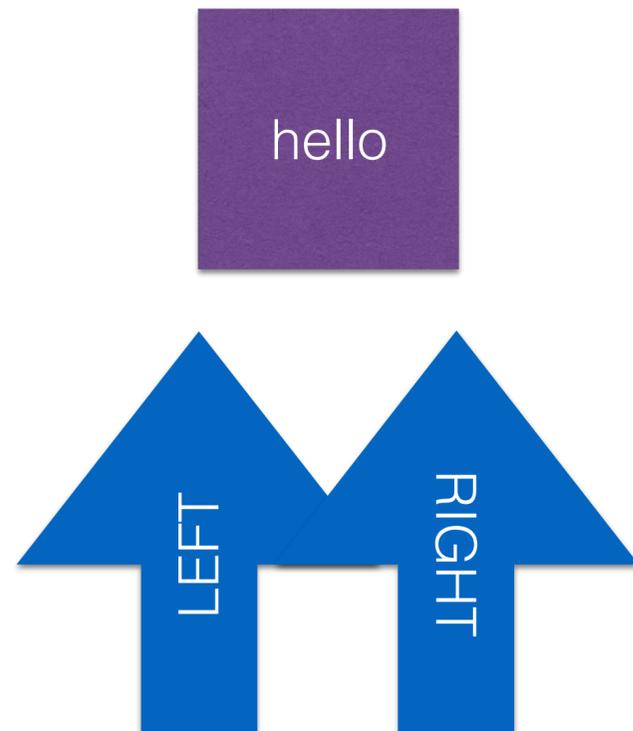
Lists



Redis Data Types

Lists

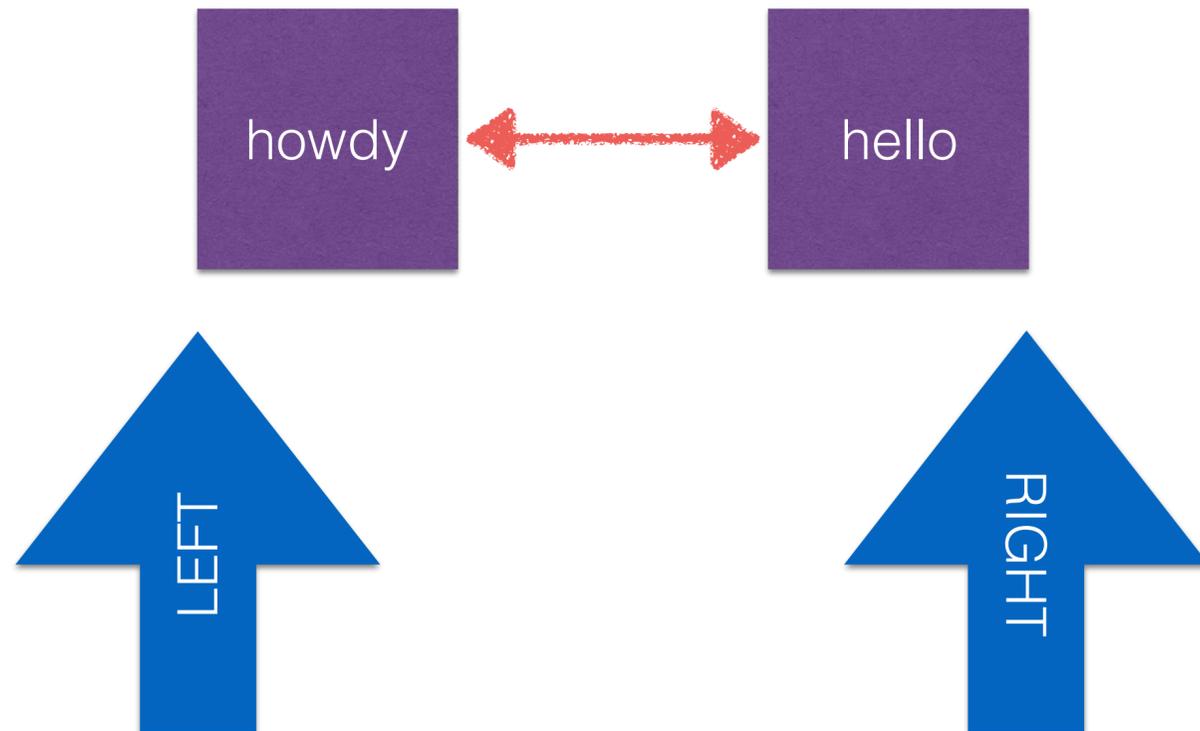
`LPUSH alist howdy`



Redis Data Types

Lists

`LPUSH aList howdy`

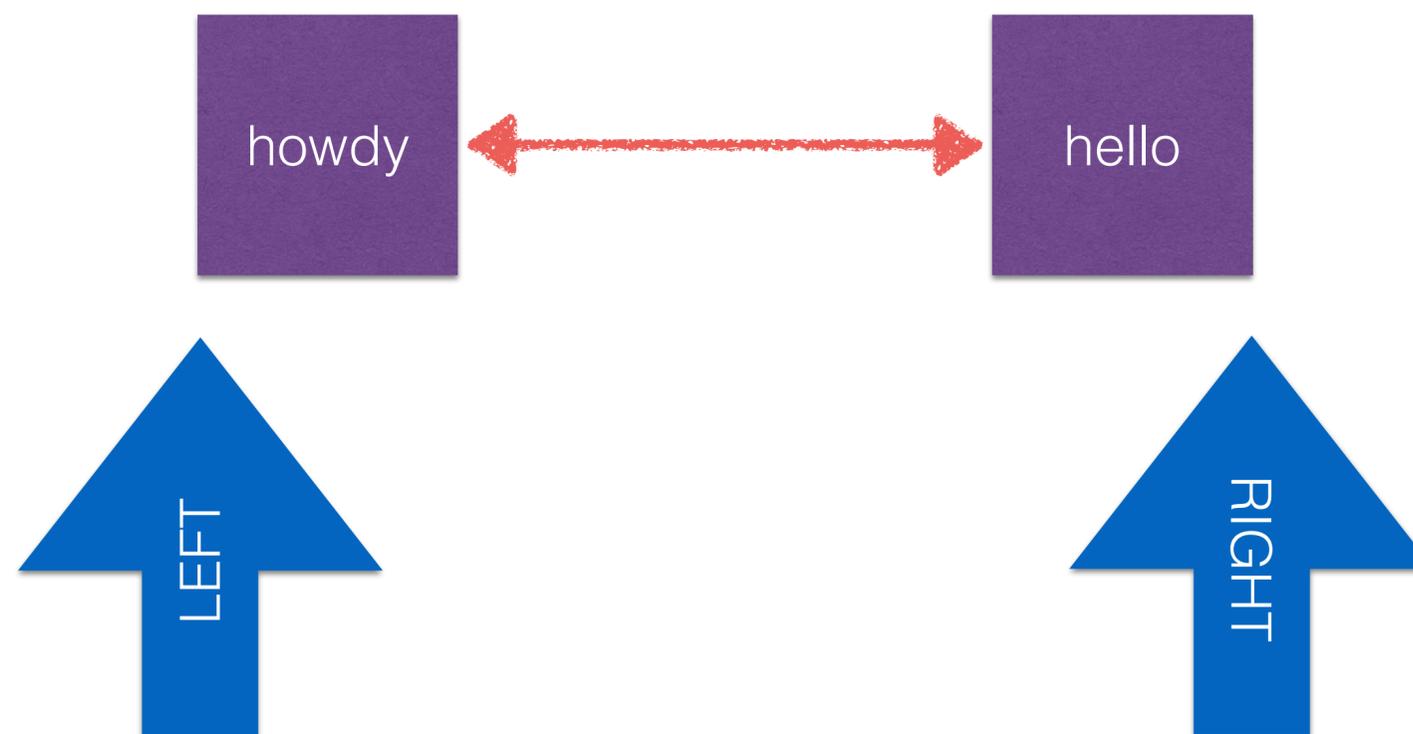


Redis Data Types

Lists

Redis Data Types

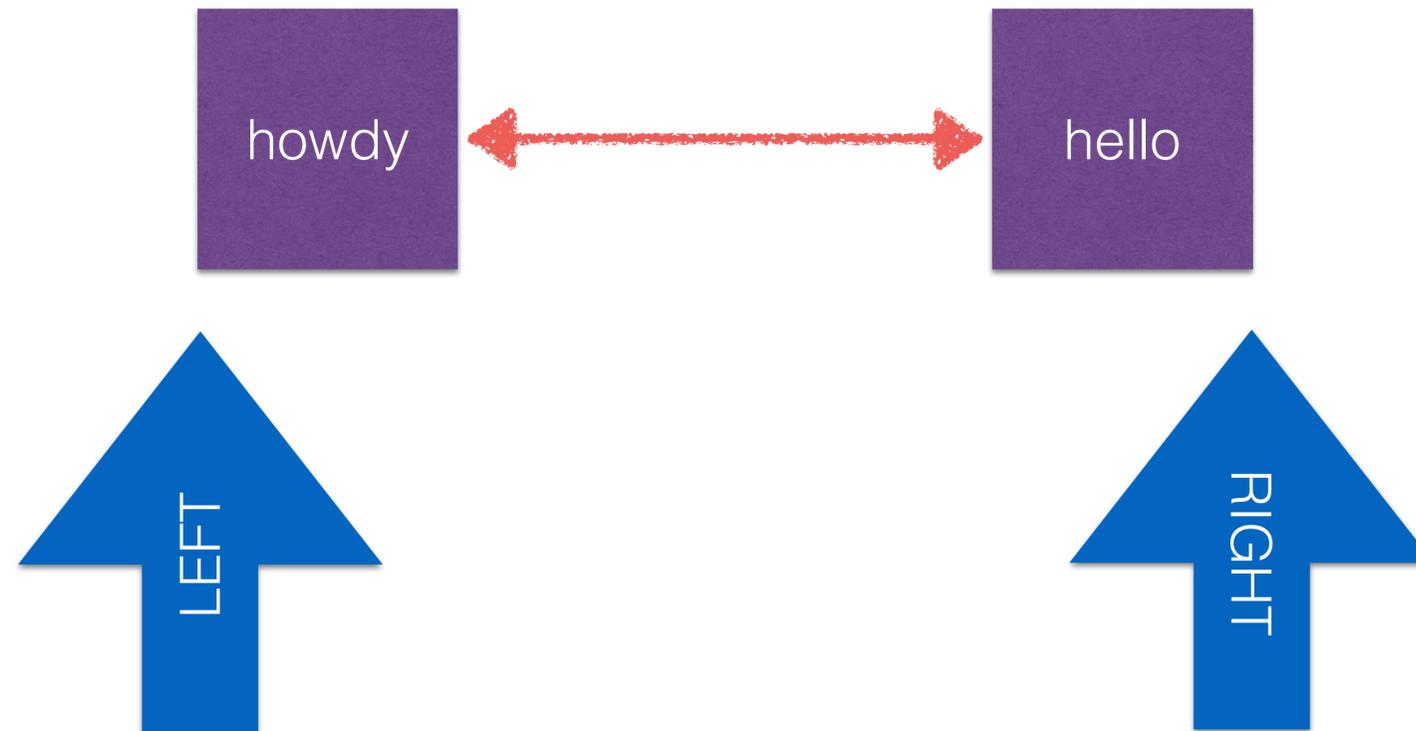
Lists



Redis Data Types

Lists

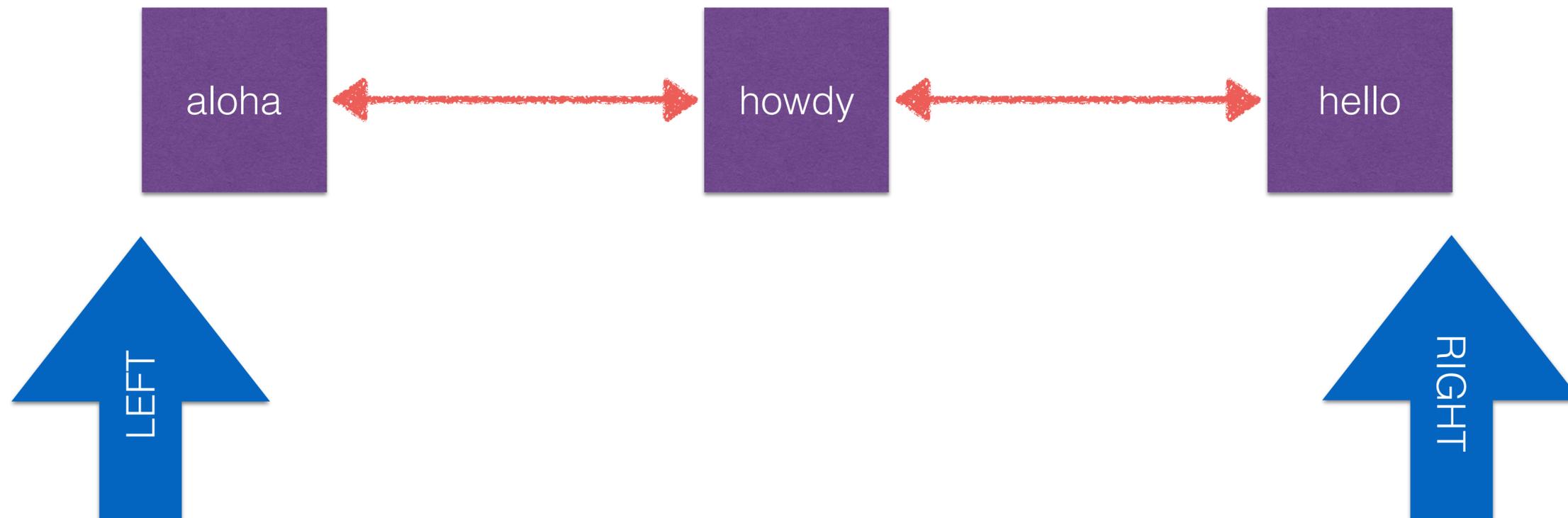
`LPUSH alist aloha`



Redis Data Types

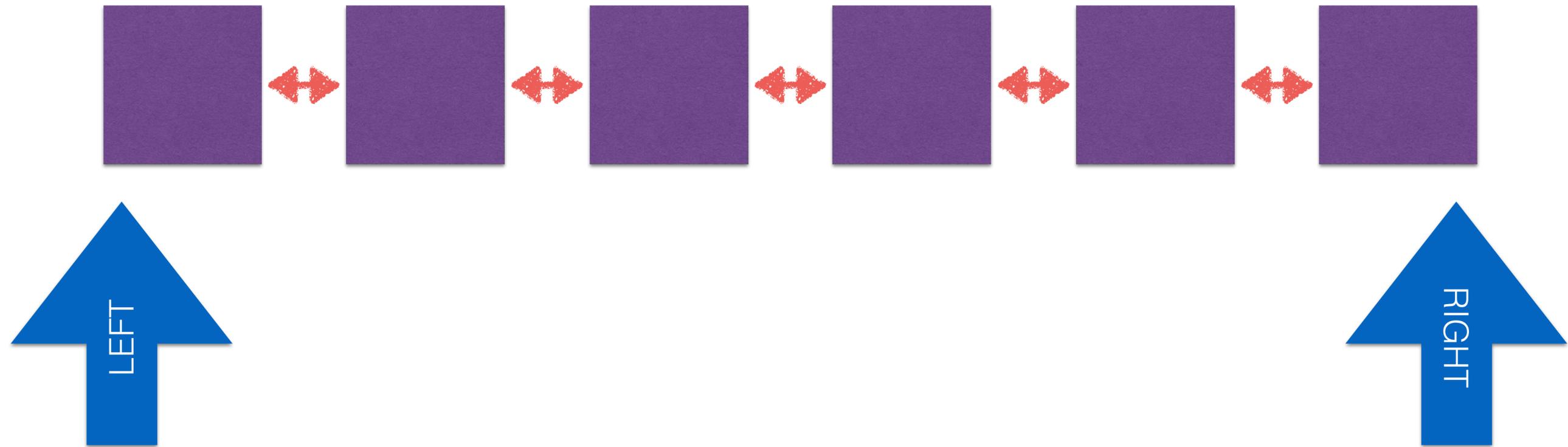
Lists

`LPUSH alist aloha`



Redis Data Types

Lists



Redis Data Types

Lists

Redis Data Types

Lists

```
127.0.0.1:6379> LPUSH mention:redis redis.io
```

Redis Data Types

Lists

```
(integer) 1
```

Redis Data Types

Lists

```
127.0.0.1:6379> LPUSH mention:redis news.ycombinator.com
```

Redis Data Types

Lists

```
(integer) 2
```



Redis Data Types

Lists

```
127.0.0.1:6379> LPUSH mention:redis github.com/antirez
```



Redis Data Types

Lists

```
(integer) 3
```



Redis Data Types

Lists

```
127.0.0.1:6379> LRANGE mention:redis 0 -1
```



Redis Data Types

Lists

3) "redis.io"



Redis Data Types

Lists



Redis Data Types

Lists

```
127.0.0.1:6379> LPOP mention:redis
```



Redis Data Types

Lists

```
"github.com/antirez"
```



Redis Data Types

Lists

```
127.0.0.1:6379> RPOP mention:redis
```



Redis Data Types

Lists

```
"redis.io"
```

```
news
```

Redis Data Types

Lists

```
127.0.0.1:6379> LPOP mention:redis
```



news

Redis Data Types

Lists

```
"news.ycombinator.com"
```

Redis Data Types

More Lists

Redis Data Types

More Lists

```
127.0.0.1:6379> LPOP mention:redis
```

Redis Data Types

More Lists

```
(nil)
```

Redis Data Types

More Lists

```
127.0.0.1:6379> BLP0P mention:redis 3
```

Redis Data Types

More Lists

(3.66s)

Redis Data Types

Even More Lists

Redis Data Types

Even More Lists

RPOPLPUSH
BRPOPLPUSH

BLPOP
BRPOP

LLEN

LREM
LSET
LTRIM

LINDEX
LINSERT

LPUSHX
RPUSHX

Redis Data Types

Hashes

Redis Data Types

Hashes

dictionaries

Redis Data Types

Hashes

dictionaries

maps

Redis Data Types

Hashes

dictionaries

maps

hash tables

Redis Data Types

Hashes

dictionaries

maps

hash tables

collections

Redis Data Types

Hashes

Redis Data Types

Hashes

key =

Redis Data Types

Hashes

key =

```
field1 = val1  
field2 = val2  
field3 = val3  
field4 = val4
```

```
·  
·
```

Redis Data Type

Hashes

values are
only strings

key =

```
field1 = val1  
field2 = val2  
field3 = val3  
field4 = val4  
.  
.
```

Redis Data Types

Hashes

Redis Data Types

Hashes

Logically:

Redis Data Types

Hashes

Logically:

hash with {field1, field2, ... fieldN}

Redis Data Types

Hashes

Redis Data Types

Hashes

Same as strings:

Redis Data Types

Hashes

Same as strings:

string:field1

string:field2

...

string:fieldN

Redis Data Types

Hashes

Redis Data Types

Hashes

Strings

Redis Data Types

Hashes

Strings

account:3391:field1 → value1

account:3391:field2 → value2

account:3391:field3 → value3

account:3391:field4 → value4

account:3391:field5 → value5

account:3391:field6 → value6

Redis Data Types

Hashes

Strings

repetition

wasted bytes

account:3391:field1 → value1

account:3391:field2 → value2

account:3391:field3 → value3

account:3391:field4 → value4

account:3391:field5 → value5

account:3391:field6 → value6

Redis Data Types

Strings

repetition

wasted bytes

excess pointers

(8 bytes each)



Redis Data Types

Hash

Strings

repetition

wasted bytes

excess pointers

(8 bytes each)



Redis Data Types

Strings

repetition

wasted bytes

excess pointers

(8 bytes each)



Hash

account:3391



field1
value1
field2
value2
field3
value3
field4
value4
field5
value5
field6
value6

Redis Data Types

Strings

repetition

wasted bytes

excess pointers

(8 bytes each)

account:3391:field1 → value1
account:3391:field2 → value2
account:3391:field3 → value3
account:3391:field4 → value4
account:3391:field5 → value5
account:3391:field6 → value6

Hash

one key

account:3391

field1
value1
field2
value2
field3
value3
field4
value4
field5
value5
field6
value6

Redis Data Types

String

repetition

wasted bytes

excess pointers

(8 bytes each)



Hash

one key

account:3391

one pointer

multiple fields

field1
value1
field2
value2
field3
value3
field4
value4
field5
value5
field6
value6

Redis Data Types

Strings

repetition

wasted bytes

excess pointers

(8 bytes each)

account:3391:field1 → value1
account:3391:field2
account:3391:field3
account:3391:field4
account:3391:field5 → value5
account:3391:field6 → value6

compact,
pointer-less*
representation

Hash

one key

account:3391

one pointer

multiple fields

field1
value1
field2
value2
field3
value3
field4
value4
field5
value5
field6
value6

Redis Data Types

Hashes

Redis Data Types

Hashes

```
127.0.0.1:6379> HSET user:matt name Matt
```

Redis Data Types

Hashes

```
(integer) 1
```

Redis Data Types

Hashes

```
127.0.0.1:6379> HSET user:matt company GoPivotal
```

Redis Data Types

Hashes

```
(integer) 1
```

Redis Data Types

Hashes

```
127.0.0.1:6379> HGETALL user:matt
```

Redis Data Types

Hashes

4) "GoPivotal"

Redis Data Types

More Hashes

Redis Data Types

More Hashes

```
127.0.0.1:6379> HINCRBY user:matt loginCount 1
```

Redis Data Types

More Hashes

```
(integer) 1
```

Redis Data Types

More Hashes

```
127.0.0.1:6379> HINCRBY user:matt loginCount 1
```

Redis Data Types

More Hashes

```
(integer) 2
```

Redis Data Types

More Hashes

```
127.0.0.1:6379> HGETALL user:matt
```

Redis Data Types

More Hashes

6) "2"

Redis Data Types

More Hashes

Redis Data Types

More Hashes

```
127.0.0.1:6379> HMSET user:matt created 2013-10-28 lastSeen 1385393884 geohash dr5rm7w
```

Redis Data Types

More Hashes

OK

Redis Data Types

More Hashes

```
127.0.0.1:6379> HGETALL user:matt
```

Redis Data Types

More Hashes

12) "dr5rm7w"

Redis Data Types

Even More Hashes

Redis Data Types

Even More Hashes

HDEL

HKEYS

HVALS

HINCRBYFLOAT

HGET

HMGET

HSETNX

HEXISTS

Redis Data Types

Sets

Redis Data Types

Sets

A collection of strings

Redis Data Types

Sets

A collection of strings
no duplicates allowed

Redis Data Types

Sets

A collection of strings
no duplicates allowed
no order preserved

Redis Data Types

Sets

Redis Data Types

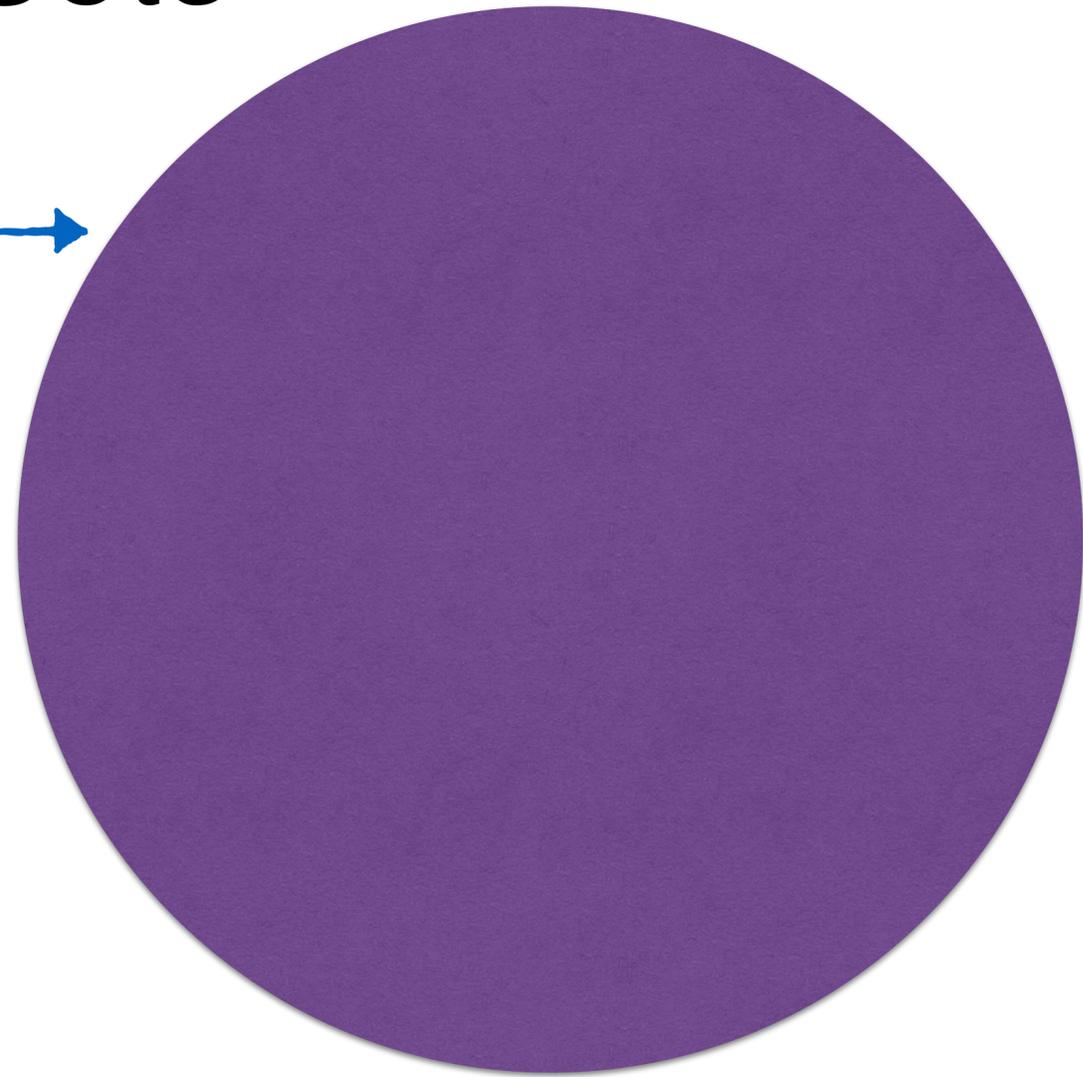
Sets



Redis Data Types

Sets

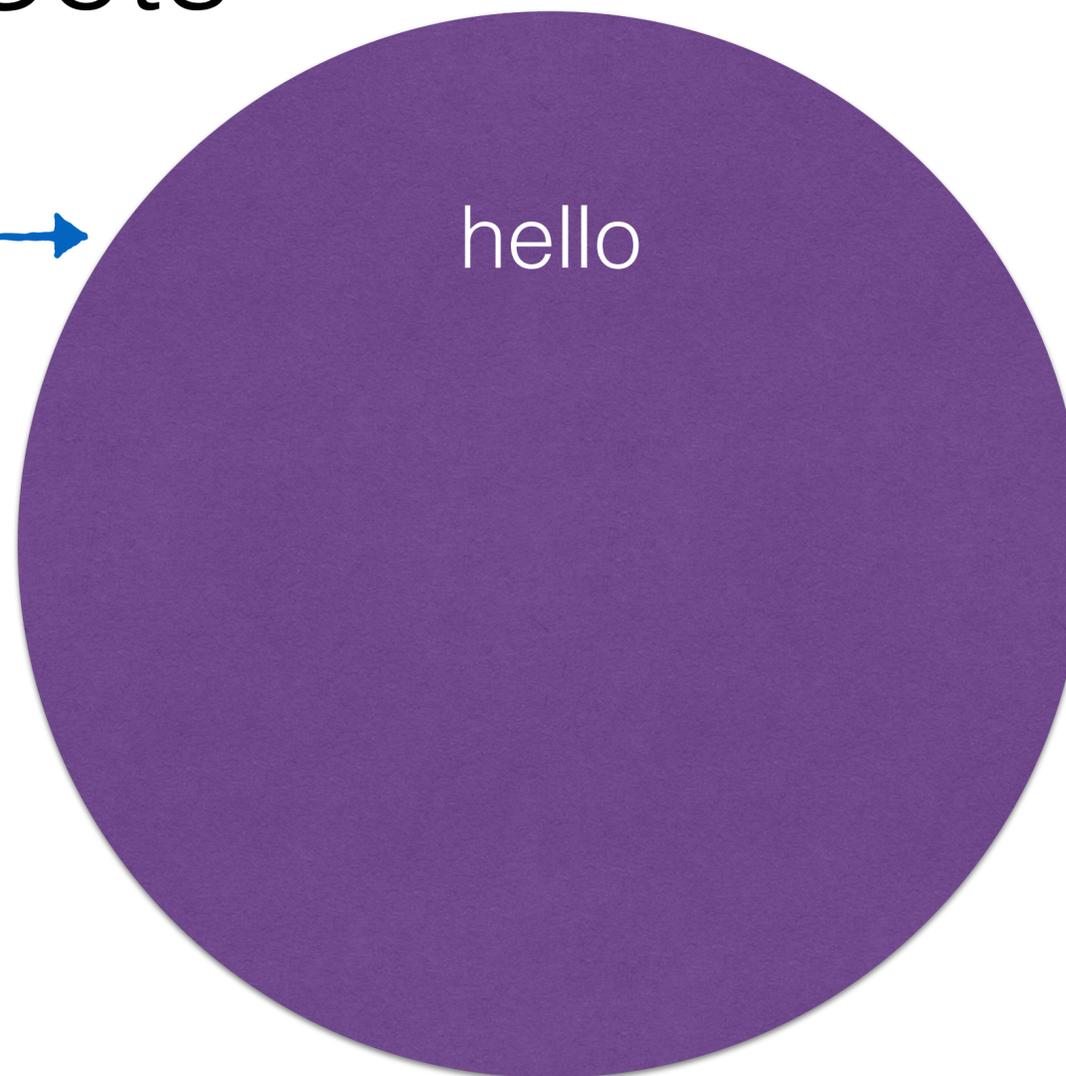
hello



Redis Data Types

Sets

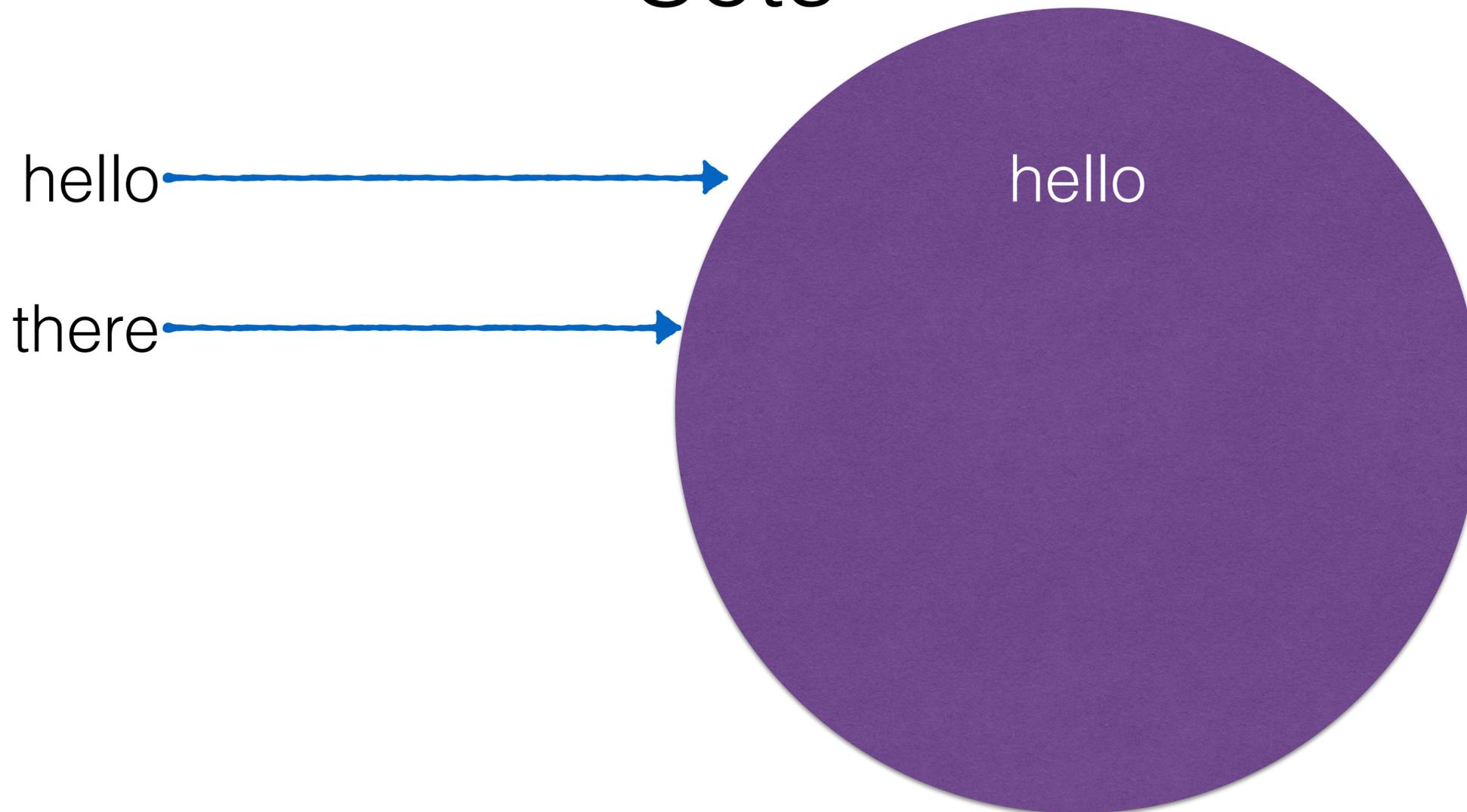
hello



hello

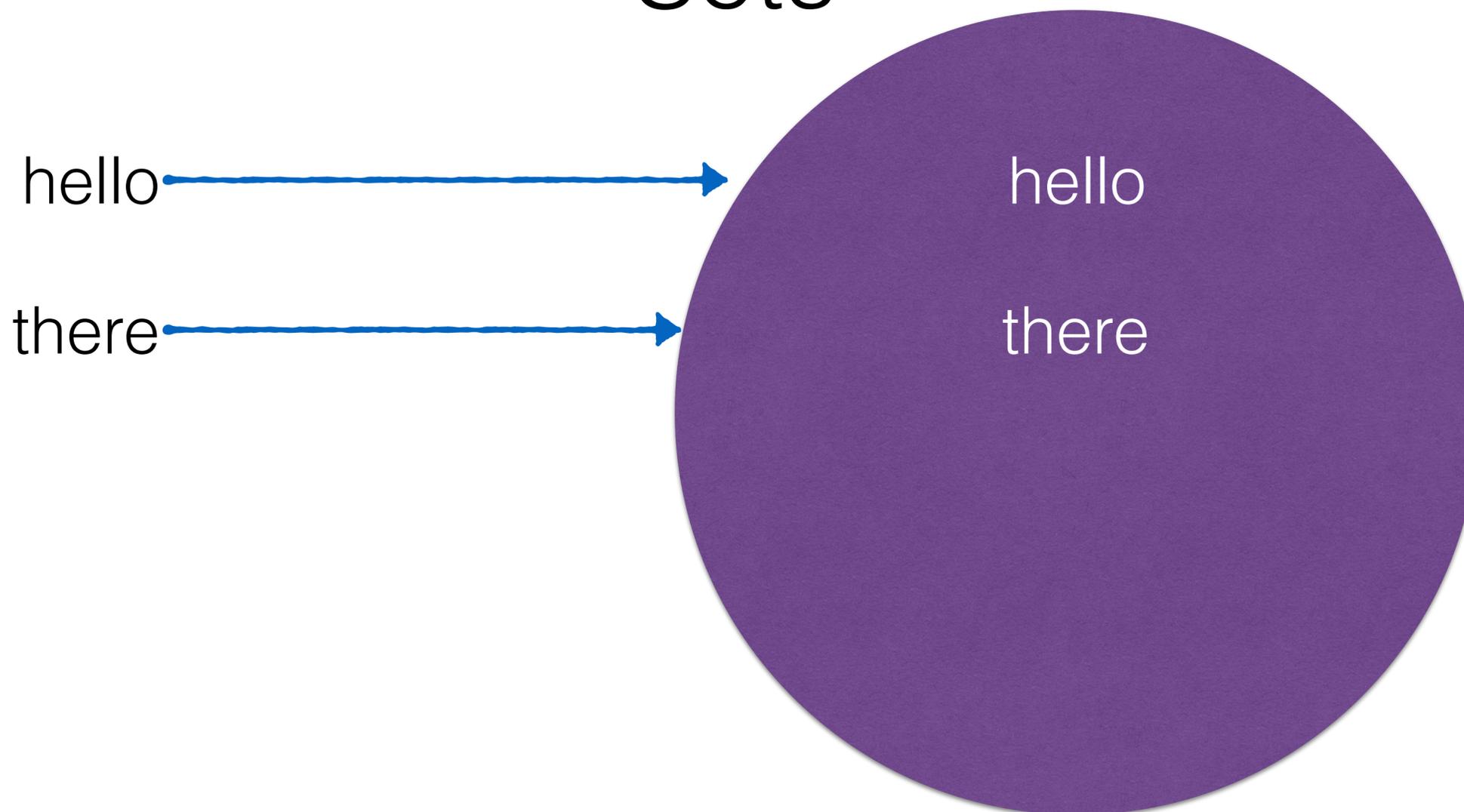
Redis Data Types

Sets



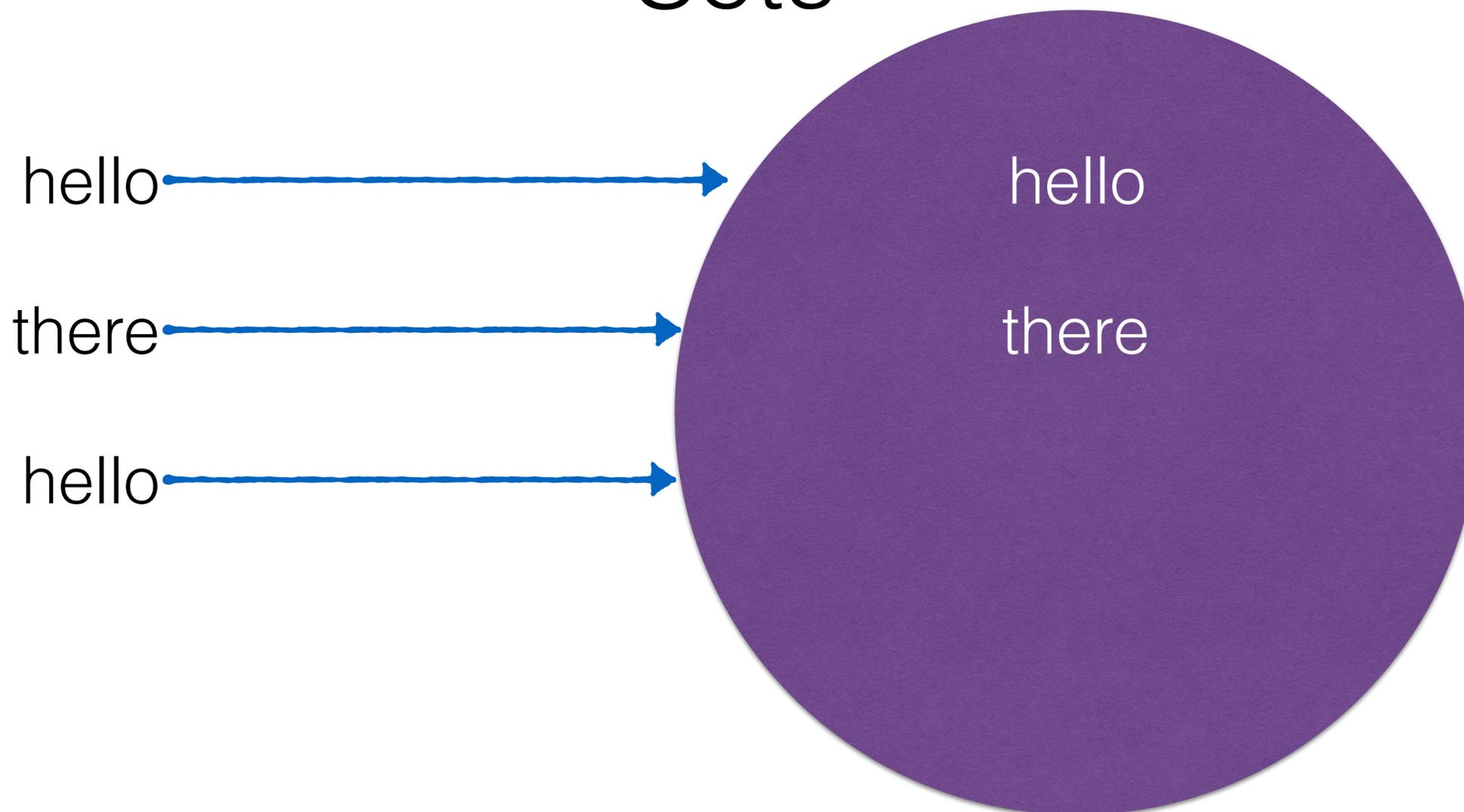
Redis Data Types

Sets



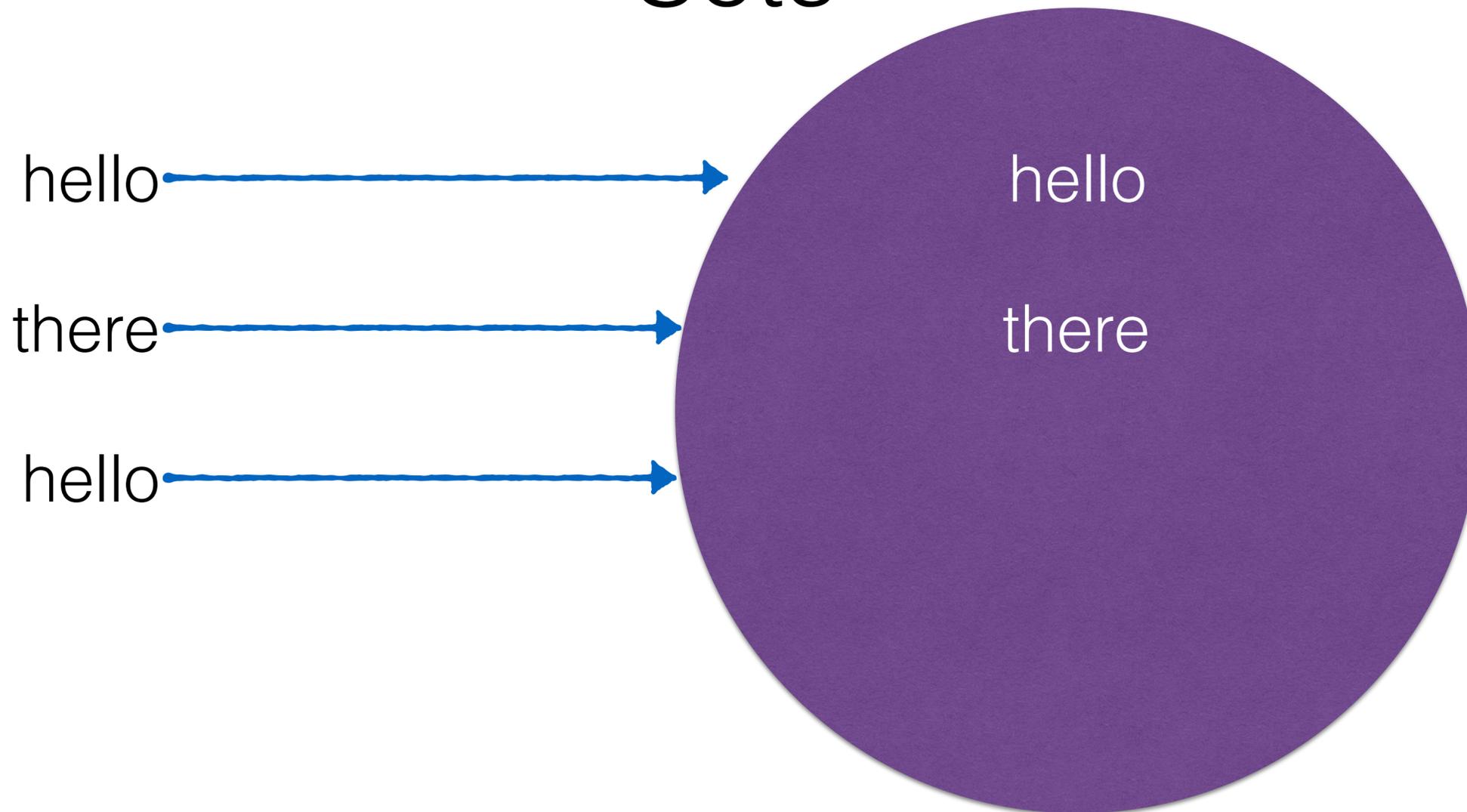
Redis Data Types

Sets



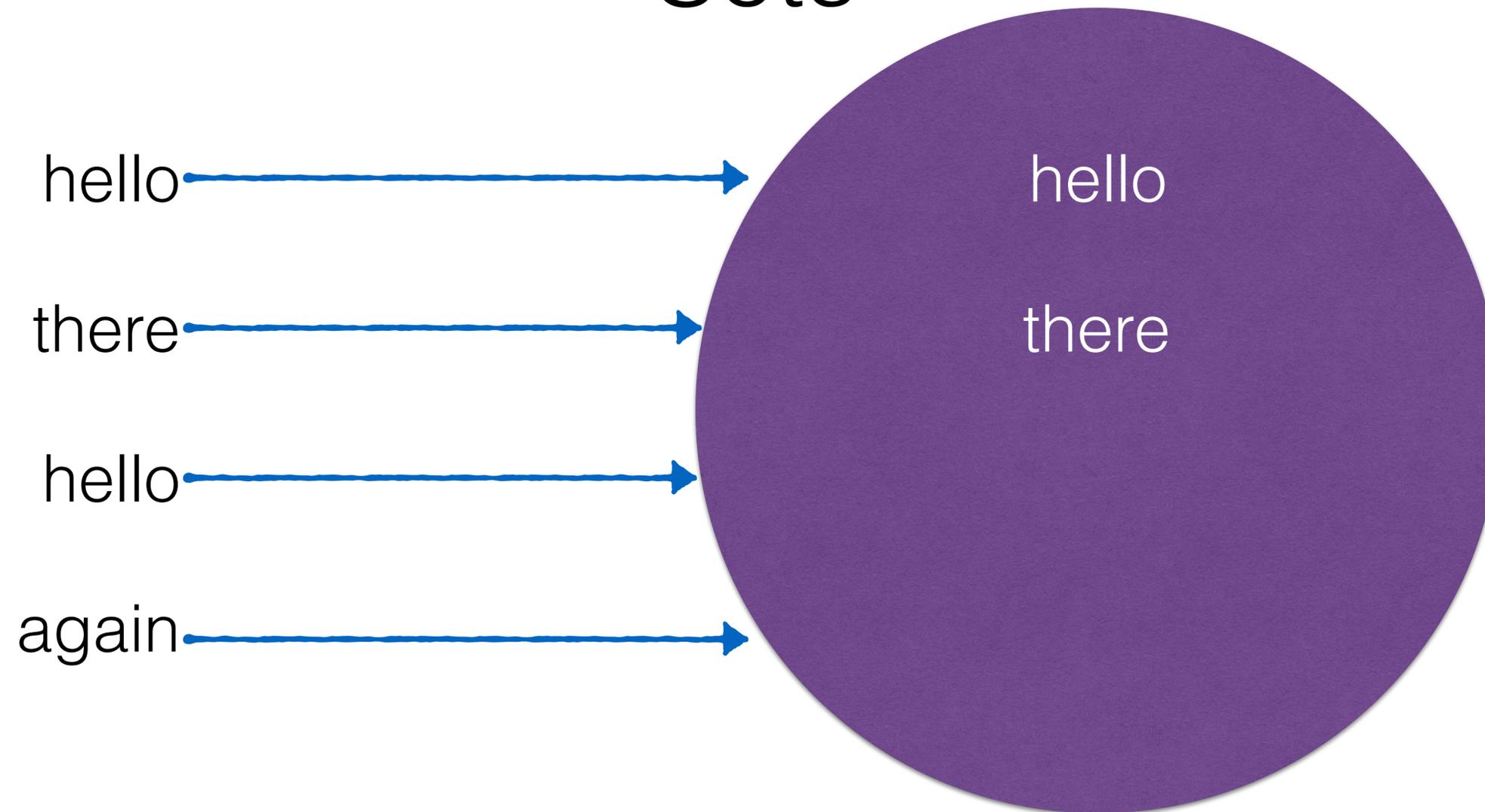
Redis Data Types

Sets



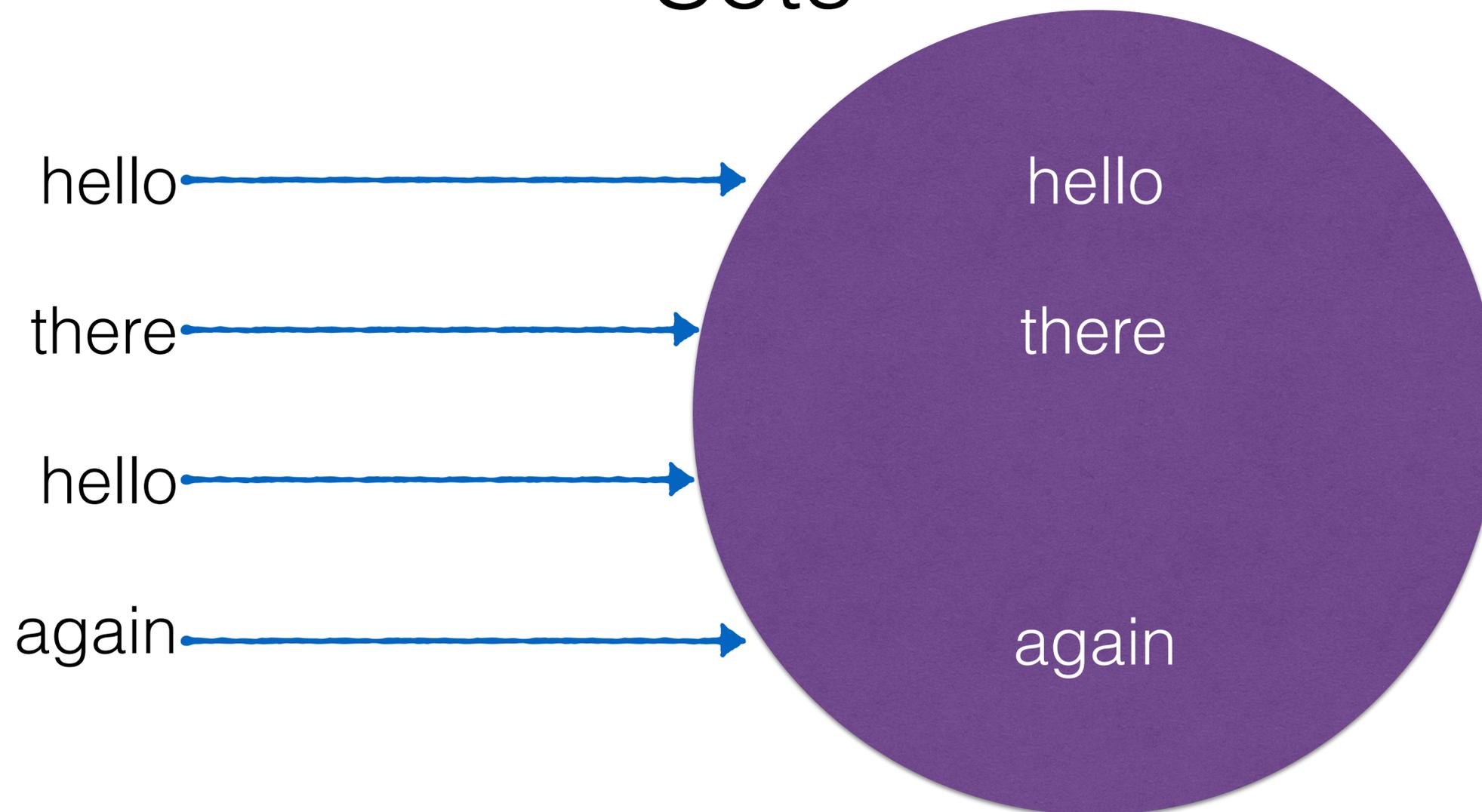
Redis Data Types

Sets



Redis Data Types

Sets



Redis Data Types

Sets

Redis Data Types

Sets

```
127.0.0.1:6379> SADD purpleCircle hello
```

Redis Data Types

Sets

```
(integer) 1
```

Redis Data Types

Sets

elements
added

```
(integer) 1
```

Redis Data Types

Sets

elements
added

```
127.0.0.1:6379> SADD purpleCircle there
```

Redis Data Types

Sets

elements
added

```
(integer) 1
```

Redis Data Types

Sets

elements
added

```
127.0.0.1:6379> SADD purpleCircle hello
```

Redis Data Types

Sets

elements
added

```
(integer) 0
```

Redis Data Types

Sets

elements
added

duplicate.
nothing added.

(integer) 0

Redis Data Types

Sets

elements
added

duplicate.
nothing added.

```
(integer) 1
```

Redis Data Types

Sets

Redis Data Types

Sets

```
127.0.0.1:6379> SMEMBERS purpleCircle
```

Redis Data Types

Sets

3) "again"

Redis Data Types

More Sets

Redis Data Types

More Sets

```
127.0.0.1:6379> SADD circle hello there hello again
```

Redis Data Types

More Sets

```
(integer) 3
```

Redis Data Types

More Sets

elements
added

```
(integer) 3
```

duplicate.
nothing added.

Redis Data Types

More Sets

elements
added

```
127.0.0.1:6379> SMEMBERS circle
```

duplicate.
nothing added.

Redis Data Types

More Sets

elements
added

3) "again"

duplicate.
nothing added.

Redis Data Types

Even More Sets

Redis Data Types

Even More Sets

SCARD

SREM

SMOVE

SPOP

SISMEMBER

SINTER

SINTERSTORE

SRANDMEMBER

SDIFF

SDIFFSTORE

SUNION

SUNIONSTORE

Redis Data Types

Sorted Sets

Redis Data Types

Sorted Sets

A collection of strings

Redis Data Types

Sorted Sets

A collection of strings
no duplicates allowed

Redis Data Types

Sorted Sets

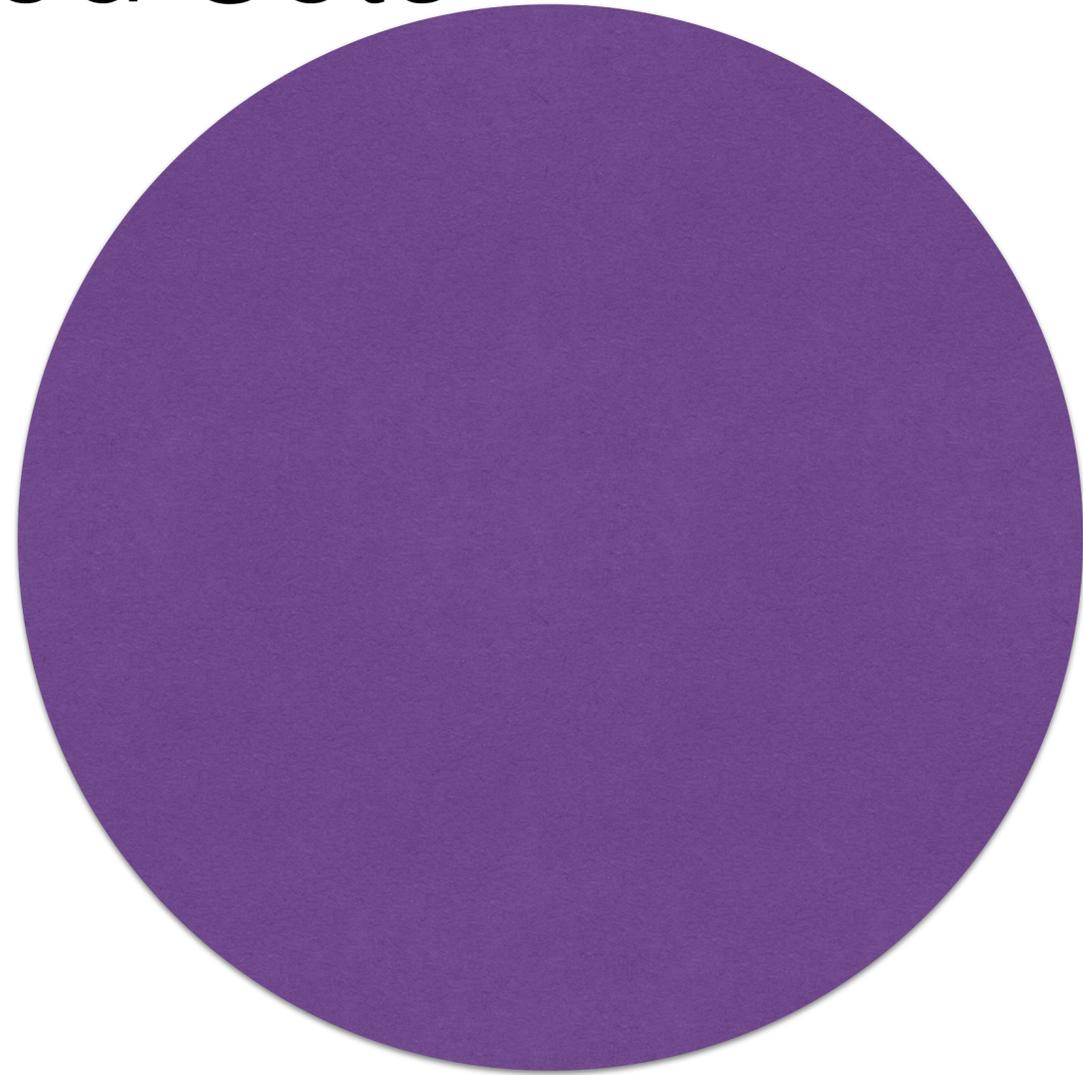
A collection of strings
no duplicates allowed
user-defined ordering

Redis Data Types

Sorted Sets

Redis Data Types

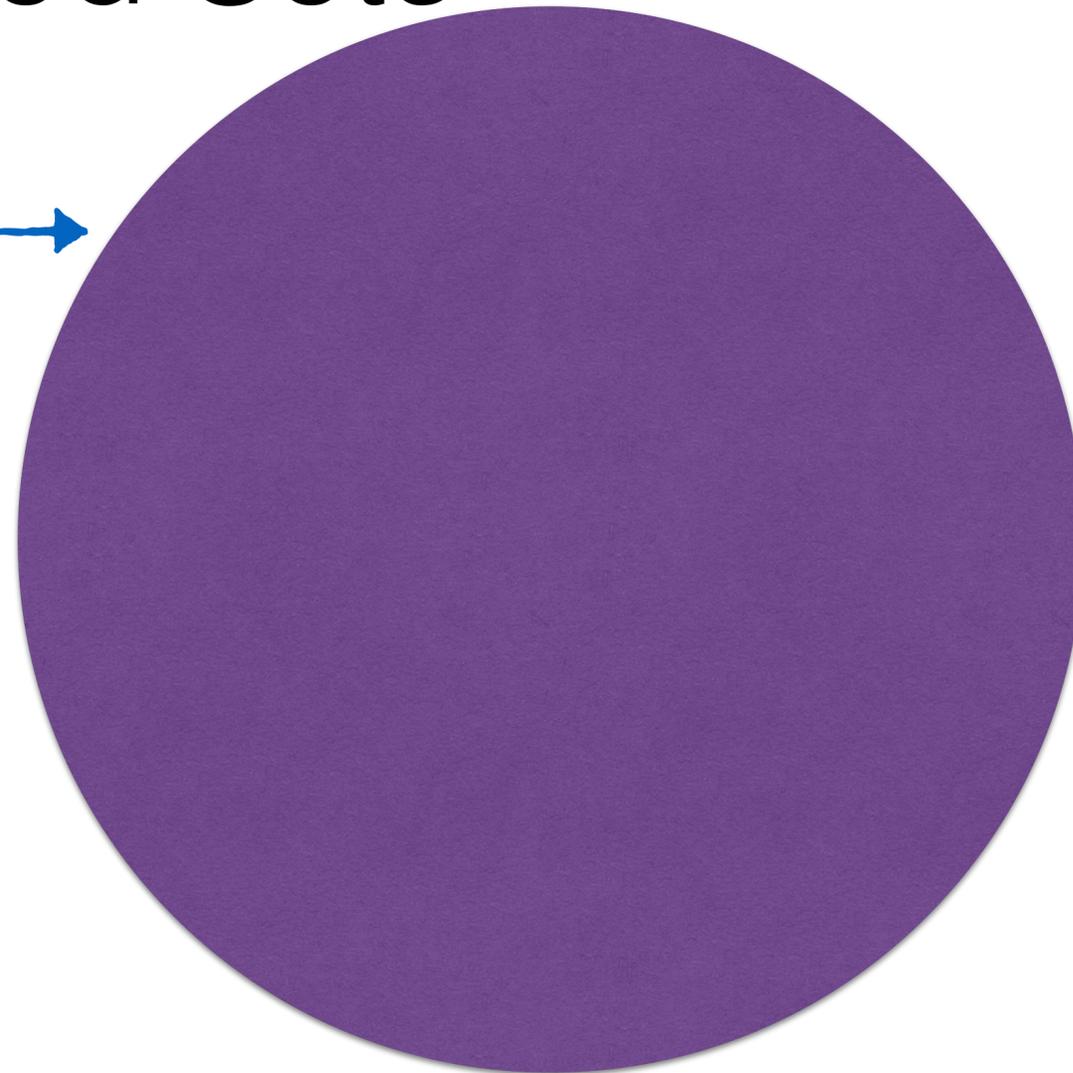
Sorted Sets



Redis Data Types

Sorted Sets

hello; score=-20.7



Redis Data Types

Sorted Sets

hello; score=-20.7



hello

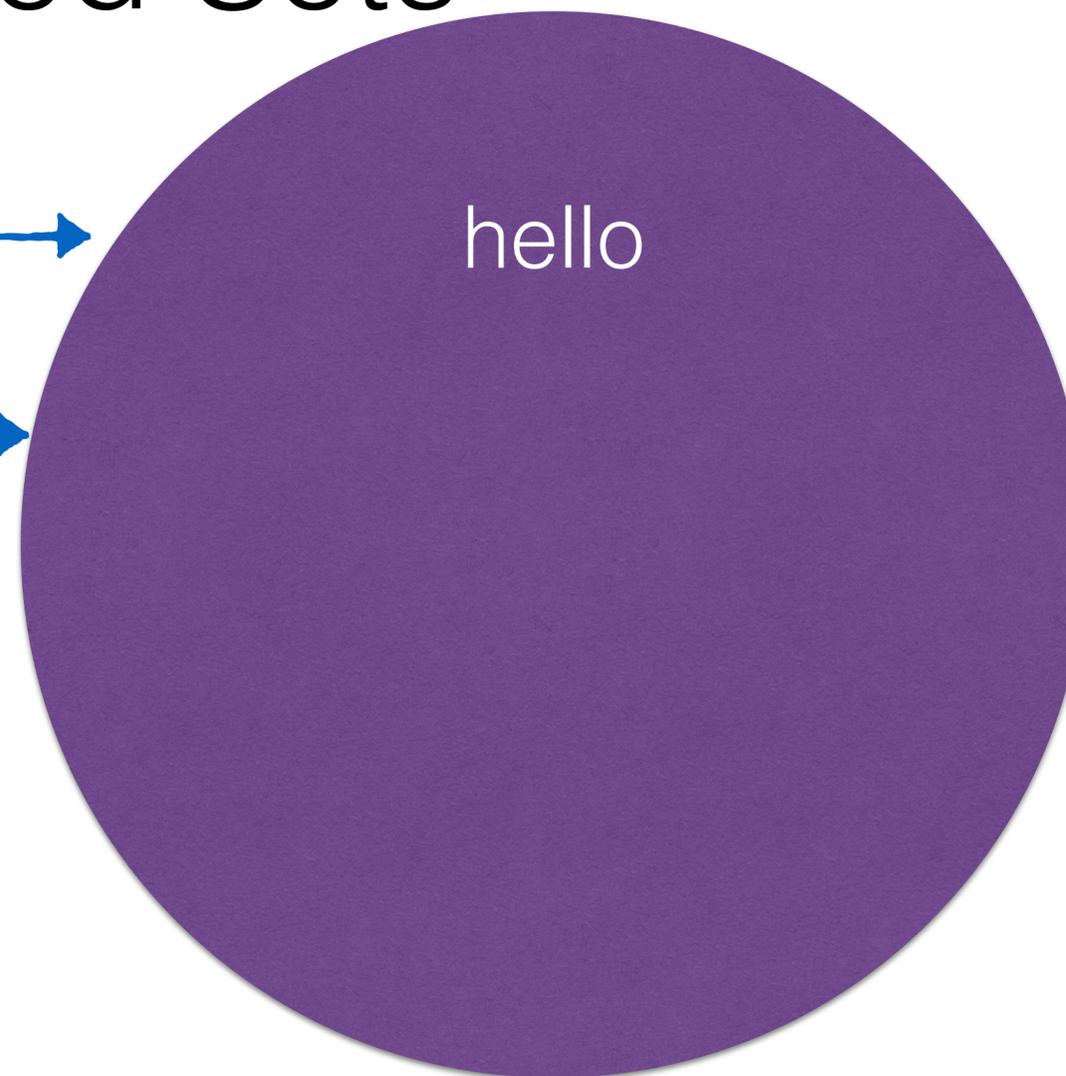
Redis Data Types

Sorted Sets

hello; score=-20.7



there; score=12



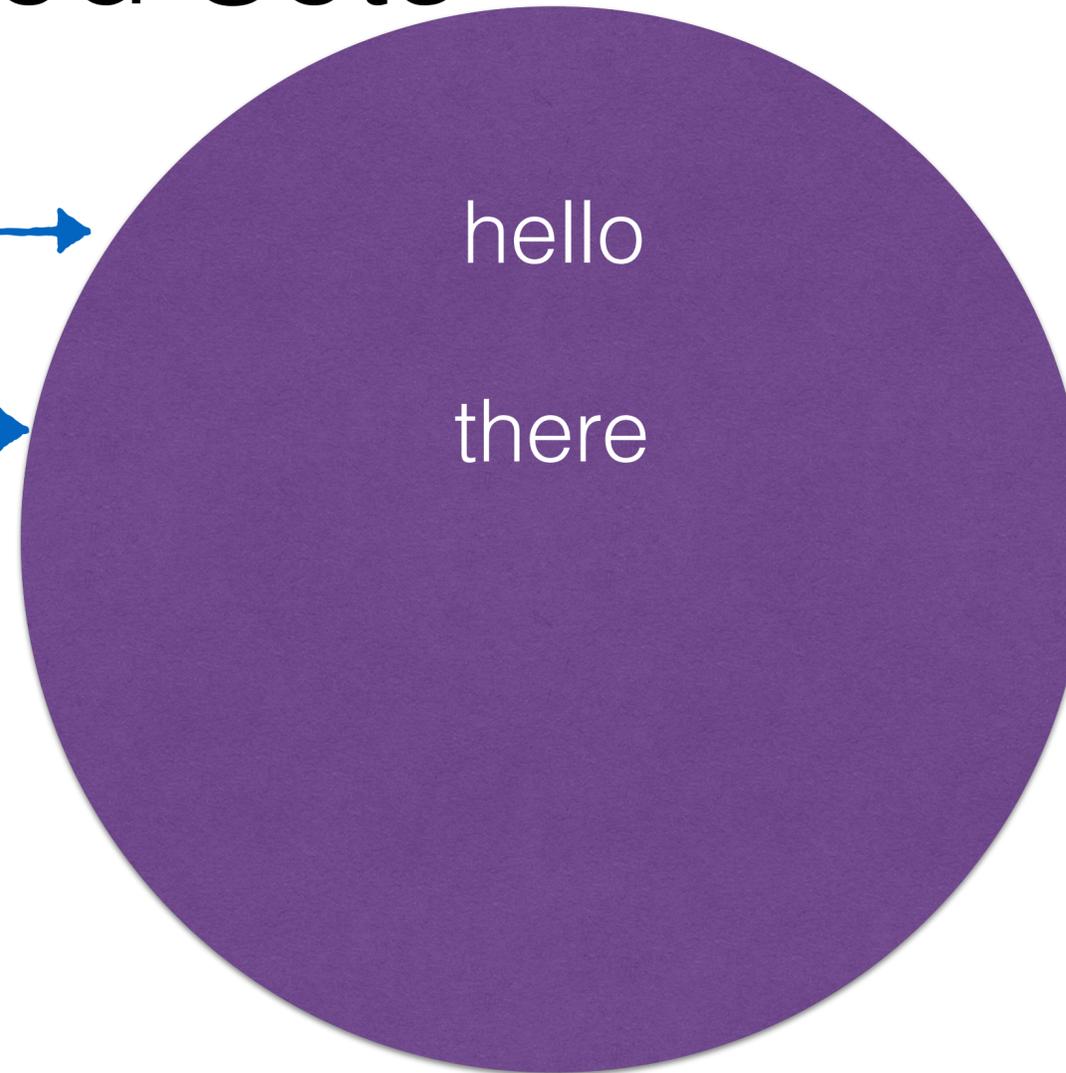
Redis Data Types

Sorted Sets

hello; score=-20.7



there; score=12



hello

there

Redis Data Types

Sorted Sets

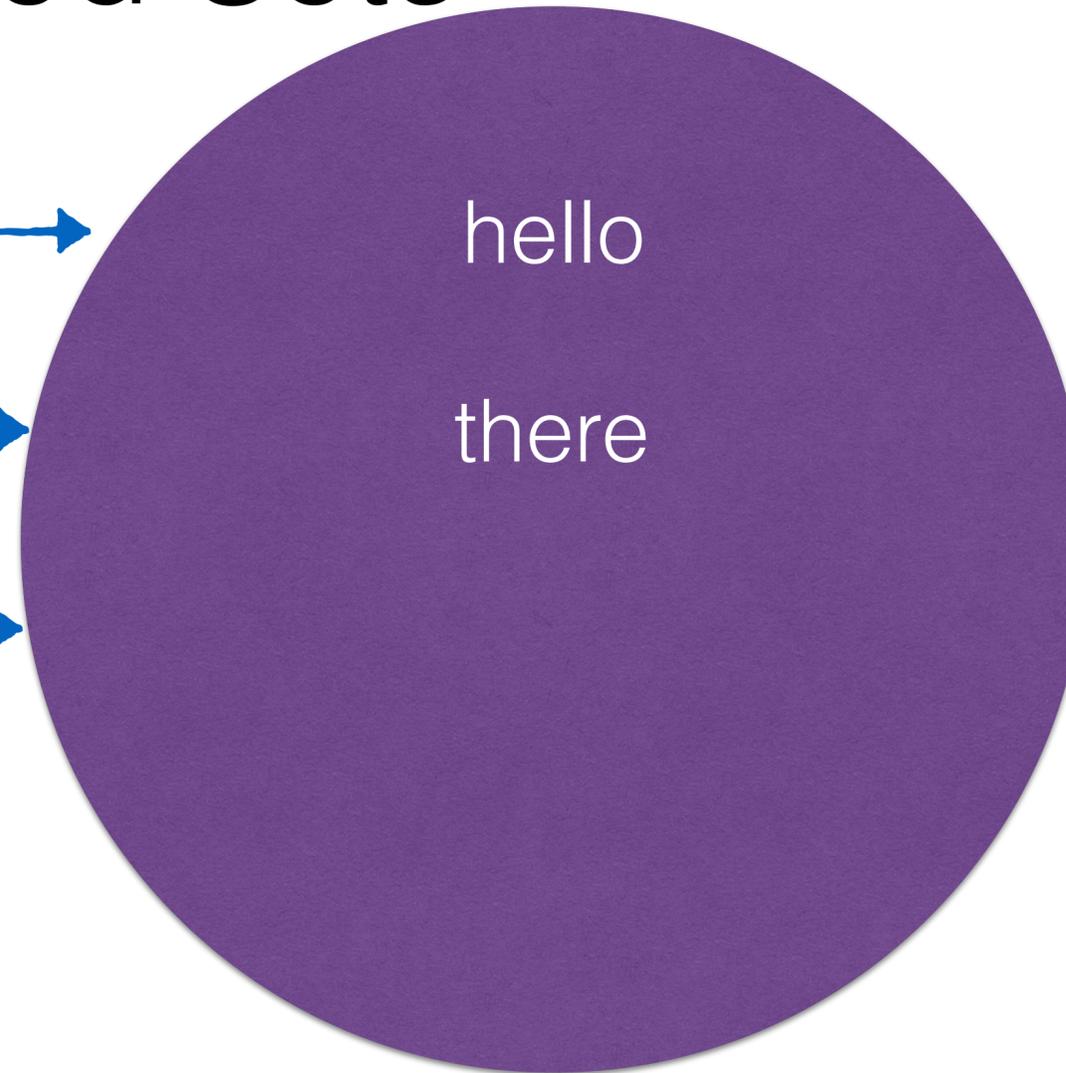
hello; score=-20.7



there; score=12



hello; score=81



Redis Data Types

Sorted Sets

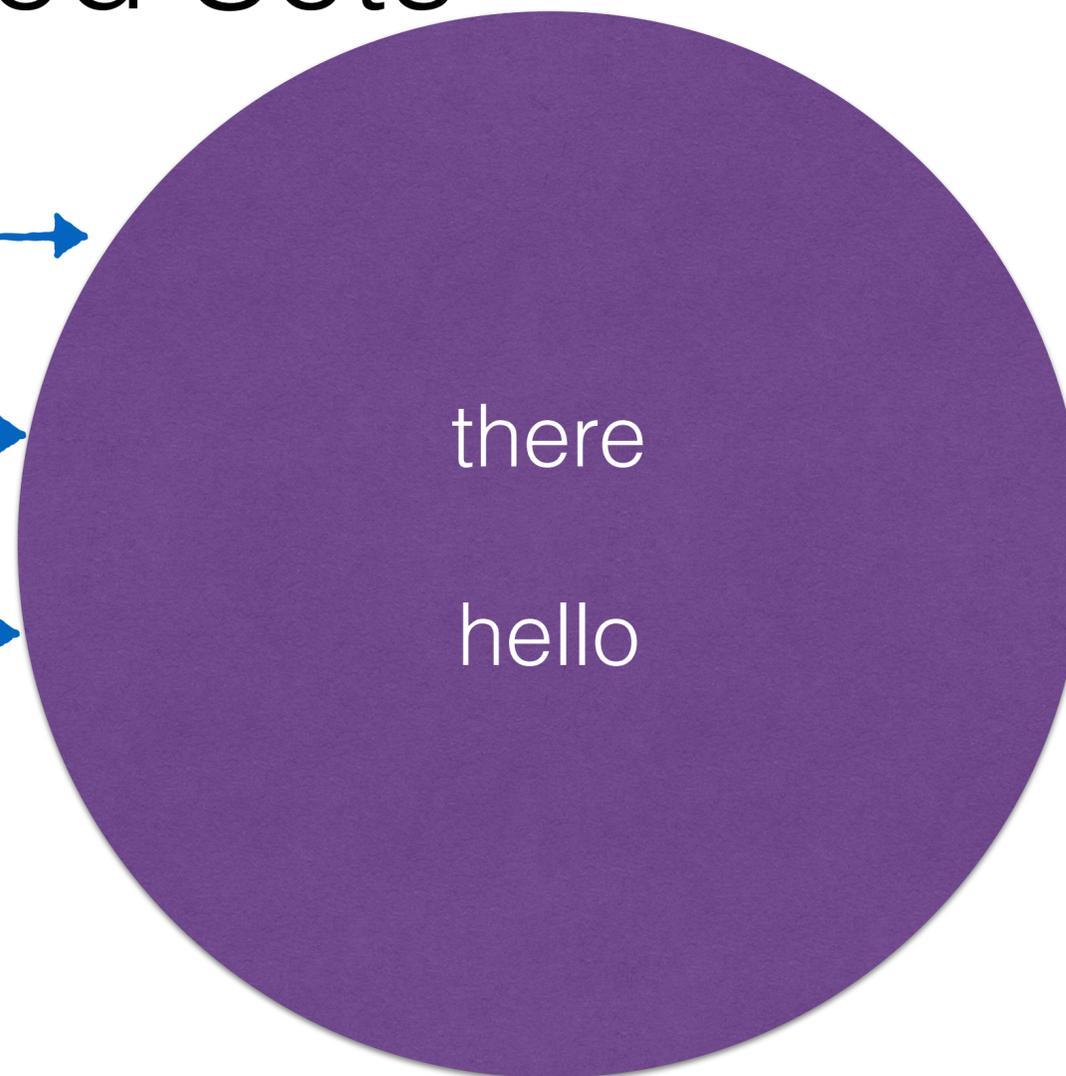
hello; score=-20.7



there; score=12



hello; score=81



there

hello

Redis Data Types

Sorted Sets

hello; score=-20.7



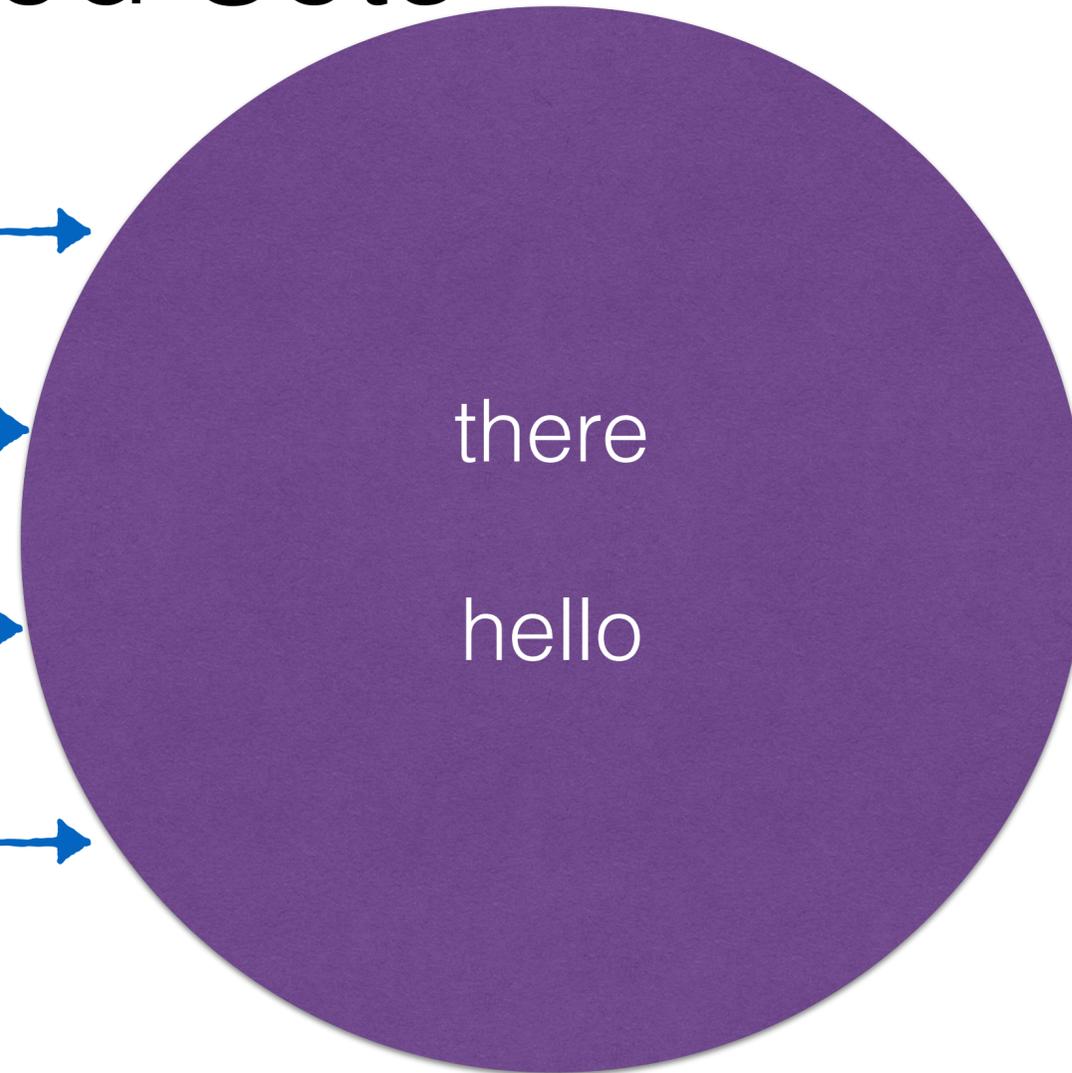
there; score=12



hello; score=81

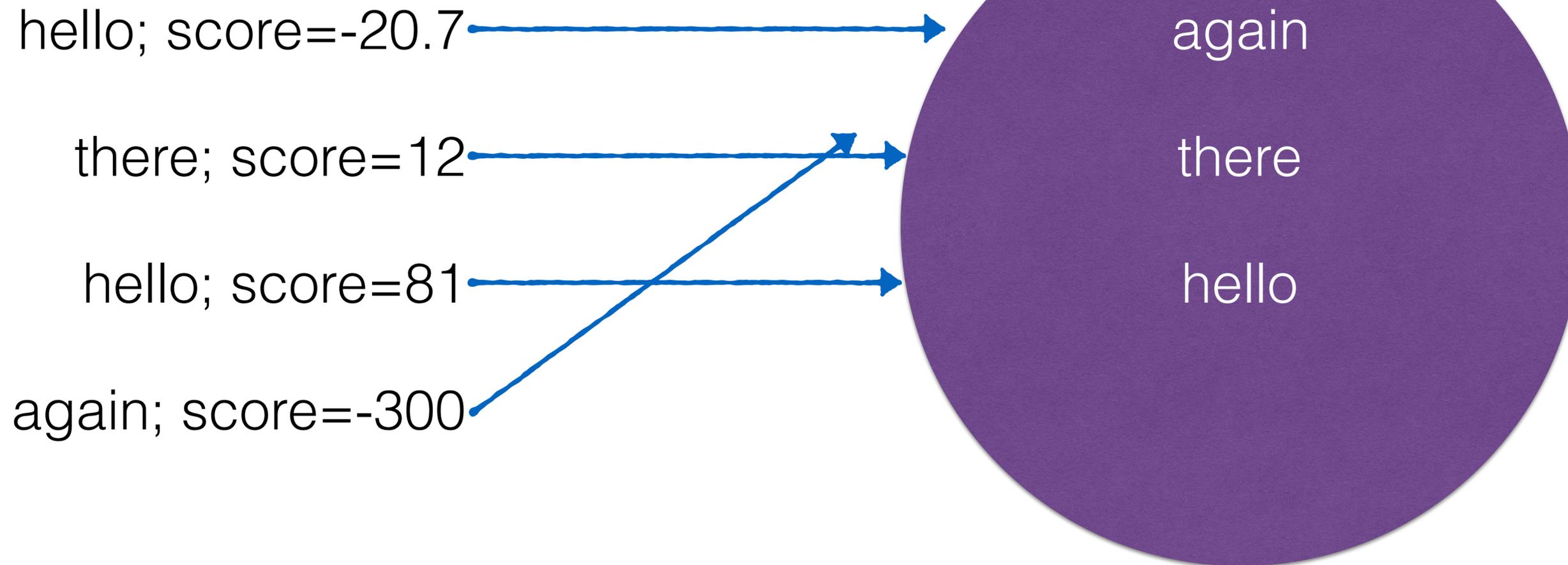


again; score=-300



Redis Data Types

Sorted Sets



Redis Data Types

Sorted Sets

Redis Data Types

Sorted Sets

```
127.0.0.1:6379> ZADD purpleCircle -20.7 hello
```

Redis Data Types

Sorted Sets

(error) WRONGTYPE Operation against a key holding the wrong kind of value

Redis Data Types

Sorted Sets

```
127.0.0.1:6379> DEL purpleCircle circle
```

Redis Data Types

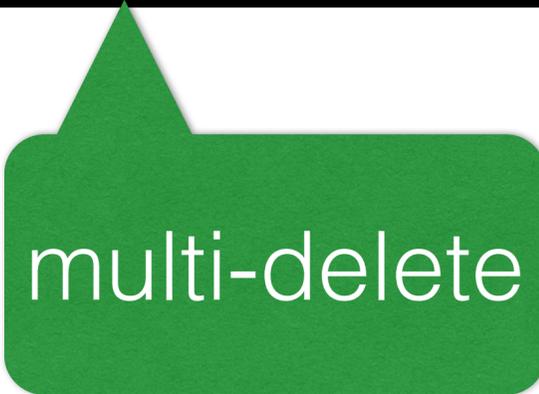
Sorted Sets

(integer) 2

Redis Data Types

Sorted Sets

```
(integer) 2
```



multi-delete

Redis Data Types

Sorted Sets

Redis Data Types

Sorted Sets

```
127.0.0.1:6379> ZADD purpleCircle -20.7 hello
```

Redis Data Types

Sorted Sets

```
127.0.0.1:6379> ZADD purpleCircle -20.7 hello
```

score

element

Redis Data Types

Sorted Sets

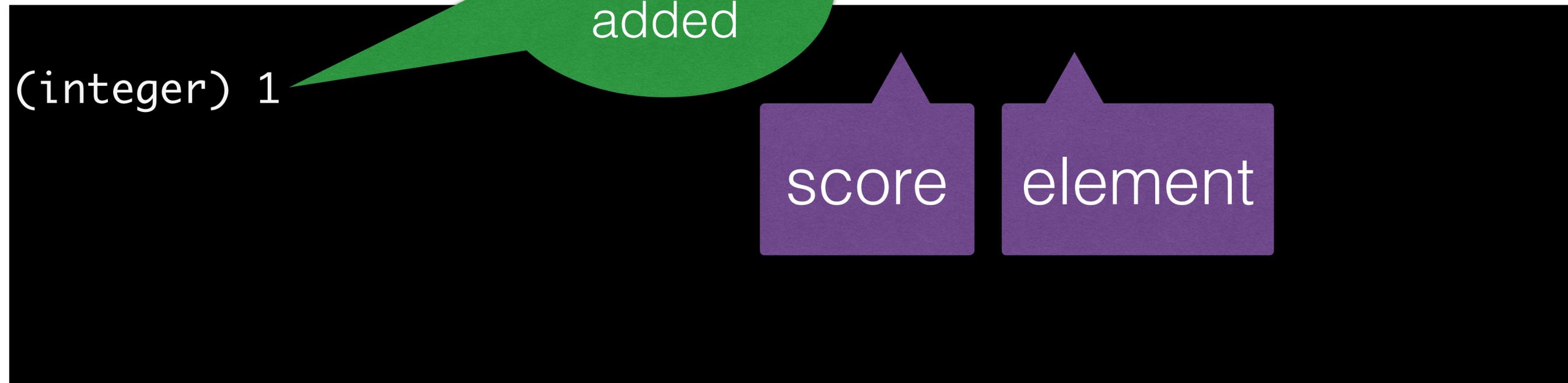
(integer) 1

score

element

Redis Data Types

Sorted Sets



Redis Data Types

Sorted Sets

A green speech bubble with a tail pointing towards the terminal command. It contains the text "elements added".

elements
added

```
127.0.0.1:6379> ZRANGEBYSCORE purpleCircle -inf +inf
```

Redis Data Types

Sorted Sets



elements
added

```
1) "hello"
```

Redis Data Types

Sorted Sets

A green speech bubble with a tail pointing towards the bottom left, containing the text "elements added".

elements
added

```
127.0.0.1:6379> ZRANGEBYSCORE purpleCircle -inf +inf WITHSCORES
```

Redis Data Types

Sorted Sets



elements
added

```
2) "-20.699999999999999999"
```

Redis Data Types

Sorted Sets

elements
added

```
2) "-20.699999999999999999"
```

because floating point

Redis Data Types

More Sorted Sets

Redis Data Types

More Sorted Sets

```
127.0.0.1:6379> ZADD purpleCircle 12 there 81 hello -300 again
```

Redis Data Types

More Sorted Sets

```
127.0.0.1:6379> ZADD purpleCircle 12 there 81 hello -300 again
```

Redis Data Types

More Sorted Sets

```
127.0.0.1:6379> ZADD purpleCircle 12 there 81 hello -300 again
```

Redis Data Types

More Sorted Sets

```
127.0.0.1:6379> ZADD purpleCircle 12 there 81 hello -300 again
```

Redis Data Types

More Sorted Sets



(integer) 2

Redis Data Types

More Sorted Sets



(integer) 2

hello already
existed

Redis Data Types

More Sorted Sets



(integer) 2

hello already
existed

score got
updated

Redis Data Types

More Sorted Sets

Redis Data Types

More Sorted Sets

```
127.0.0.1:6379> ZRANGE purpleCircle 0 -1
```

Redis Data Types

More Sorted Sets

start
position

end
position

```
127.0.0.1:6379> ZRANGE purpleCircle 0 -1
```

Redis Data Types

More Sorted Sets

start
position

end
position

```
3) "hello"
```

Redis Data Types

More Sorted Sets

start
position

end
position

```
127.0.0.1:6379> ZRANGE purpleCircle 0 -1 WITHSCORES
```

Redis Data Types

More Sorted Sets

start
position

end
position

6) "81"

Redis Data Types

More Sorted Sets

start
position

end
position

6) "81"

replaced
-20.7

Redis Data Types

Even More Sorted Sets

Redis Data Types

Even More Sorted Sets

ZREM

ZREMRANGEBYRANK

ZREMRANGEBYSCORE

ZRANK

ZREVRANK

ZCARD

ZINTERSTORE

ZRANGE

ZRANGEBYSCORE

ZREVRANGE

ZUNIONSTORE

ZREVRANGEBYSCORE

How Redis manage?

Running

Running

You can set config parameters:

Running

You can set config parameters:

on the command line (as arguments)

Running

You can set config parameters:

on the command line (as arguments)

or

Running

You can set config parameters:

on the command line (as arguments)

or

in the config file (the “normal” way)

Running

Running

You can modify most parameters
live during runtime with **CONFIG SET**

Running

You can modify most parameters
live during runtime with `CONFIG SET`

Read settings with `CONFIG GET [name]`

or

Running

You can modify most parameters live during runtime with **CONFIG SET**

Read settings with **CONFIG GET [name]**

or

CONFIG GET *

for all current settings

Running

Running

Redis can update your existing config file

Running

Redis can update your existing config file

CONFIG REWRITE

Running

Redis can update your existing config file

CONFIG REWRITE

Comments, ordering, and structure get preserved.

Stand Alone

Stand Alone

`redis-server`

Stand Alone

redis-server

lonely |'lōnlē| adjective (**lonelier** , **loneliest**)
sad because one has no friends or company

Stand Alone

redis-server



lonely |'lɒnlē| adjective (**lonelier** , **loneliest**)
sad because one has no friends or company

Stand Alone

```
matt@ununoctium:/Volumes/matt/repos/redis/src% ./redis-server
```

[17997] 25 Nov 19:13:00.979 * The server is now ready to accept connections on port 6379

Replication

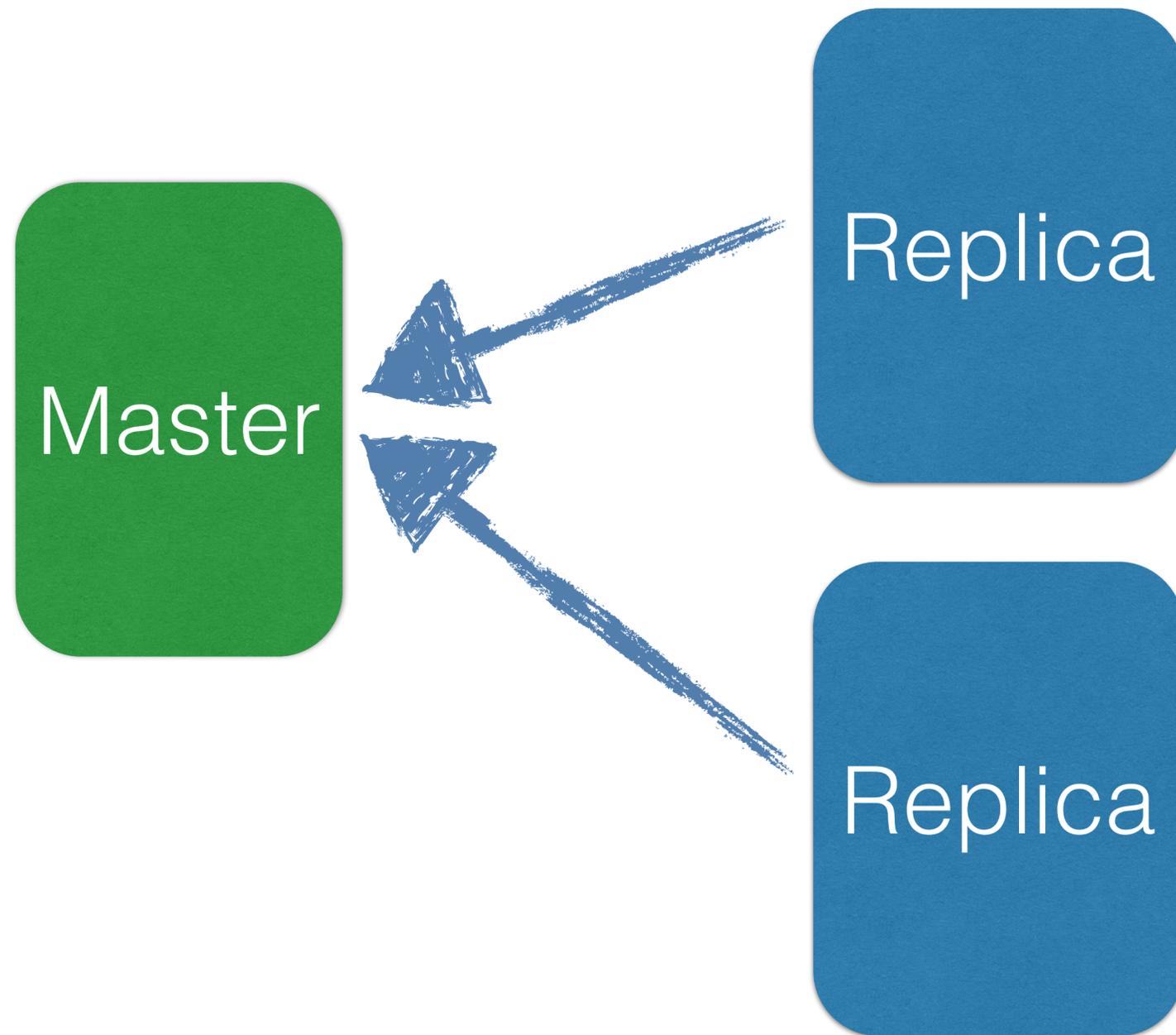
Replication



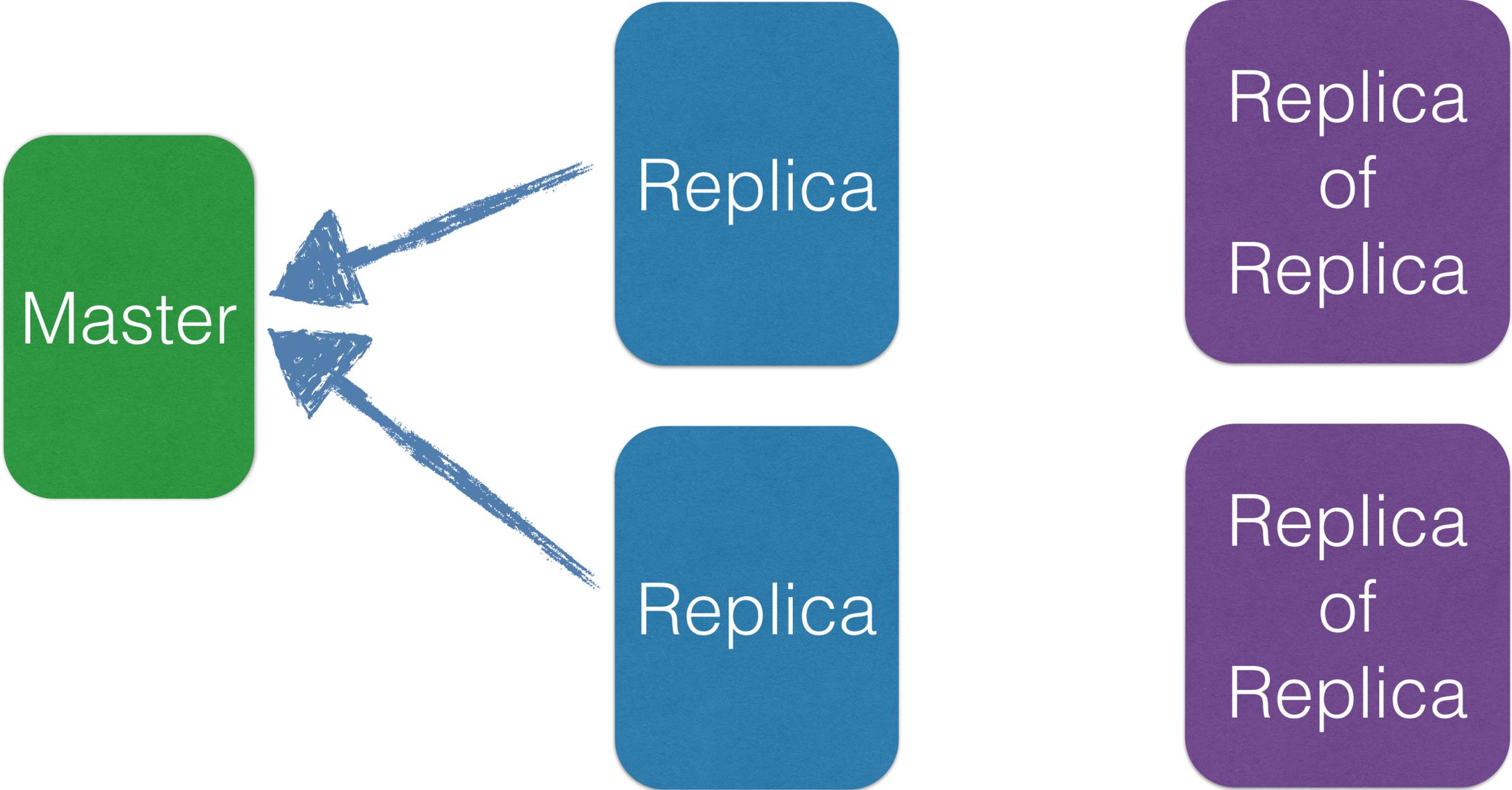
Replication



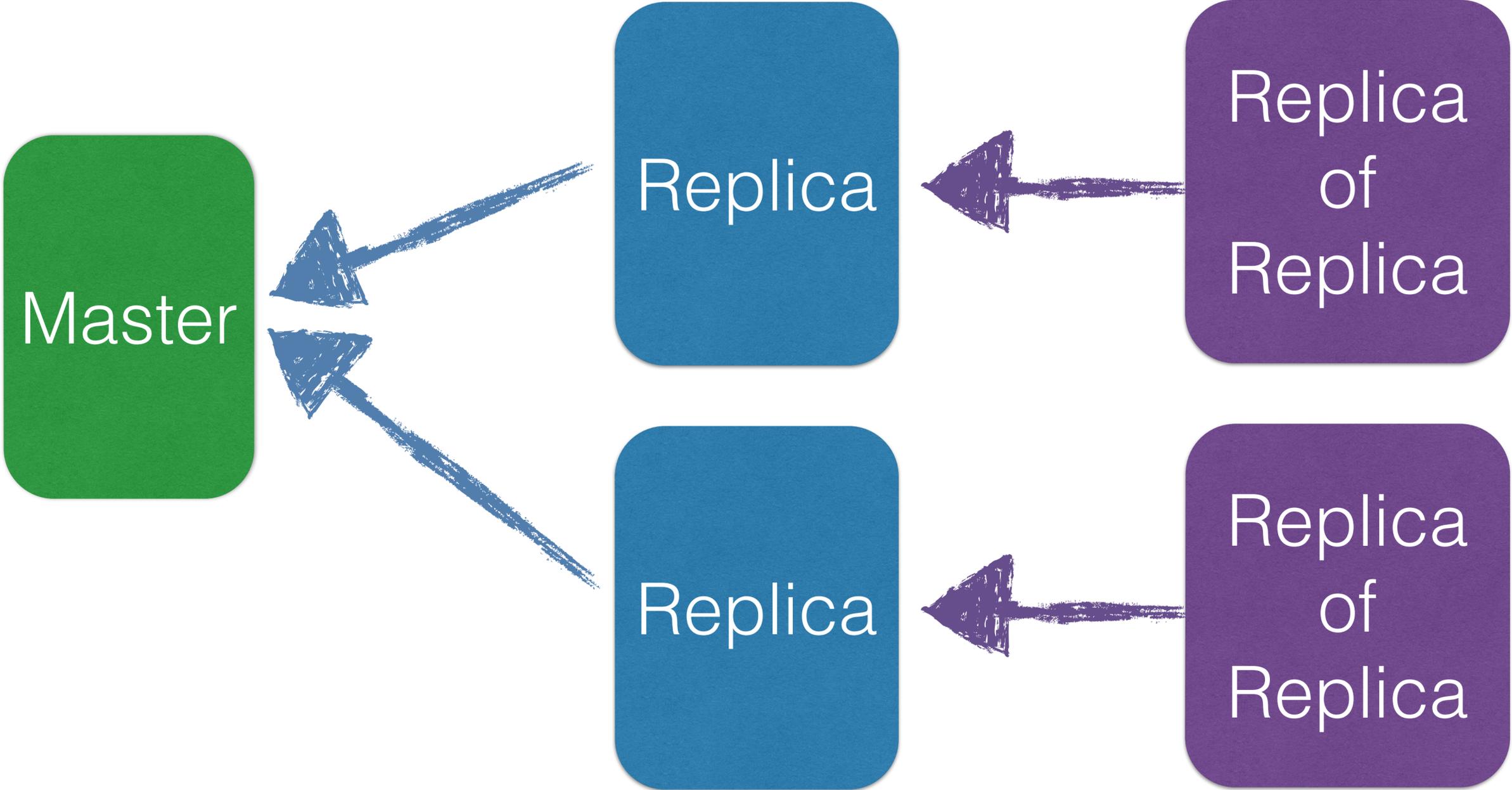
Replication



Replication



Replication



Replication

On Replica

Replication

On Replica

```
127.0.0.1:3700> KEYS *
```

Replication

On Replica

```
(empty list or set)
```

Replication

On Replica

```
127.0.0.1:3700> SLAVEOF 127.0.0.1 6379
```

Replication

On Replica

OK

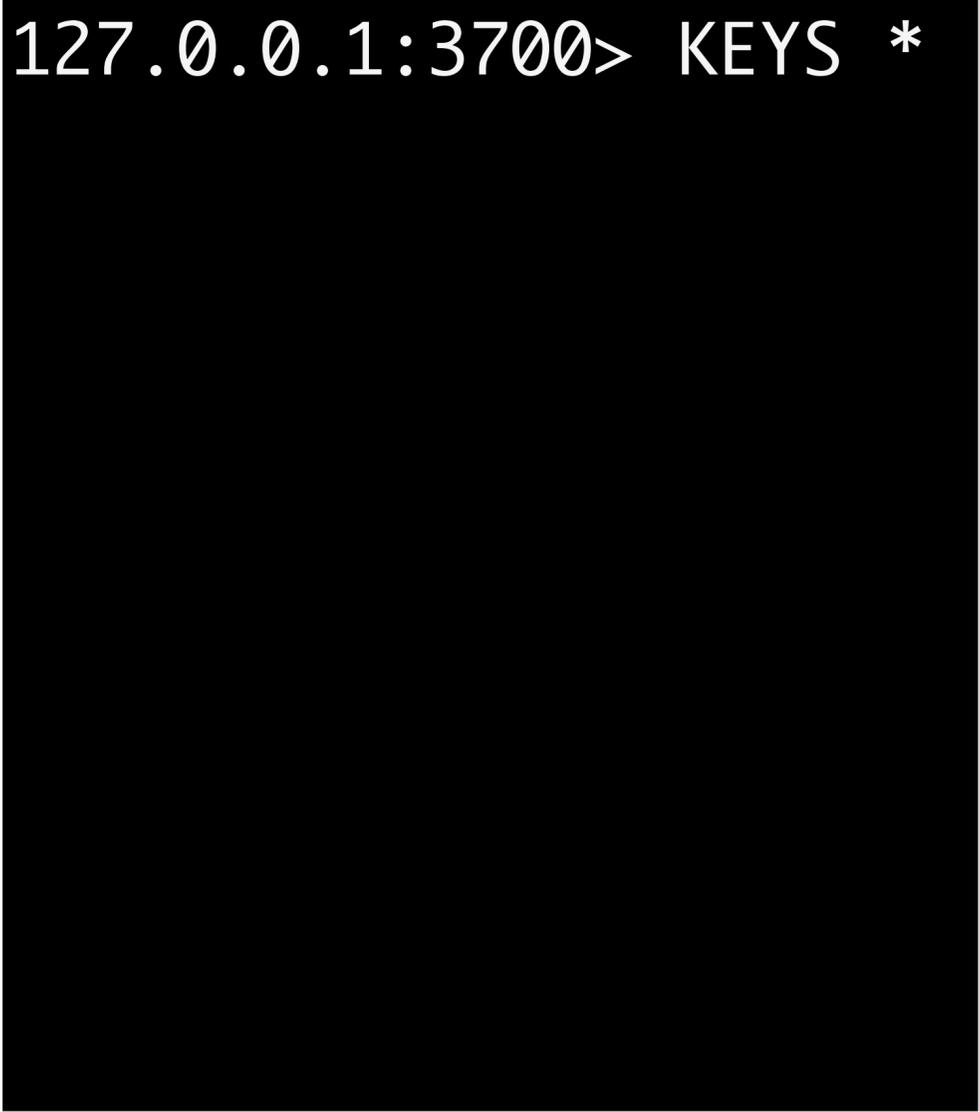
Replication

On Replica

Replication

On Replica

```
127.0.0.1:3700> KEYS *
```



Replication

On Replica

11) "name"

Replication

On Replica

Replication

On Replica

```
[17594] 25 Nov 16:45:58.498 * The server is now ready to accept connections on port 3700
```

Replication

On Replica

```
[17594] 25 Nov 16:46:33.897 * MASTER <-> SLAVE sync: Finished with success
```

Replication

On Master

Replication

On Master

```
[4752] 25 Nov 16:58:51.185 * Synchronization with slave succeeded
```

Replication

On Replica

Replication

On Replica

```
127.0.0.1:3700> SLAVEOF NO ONE
```

Replication

On Replica

OK

Replication

On Replica

Replication

On Replica

```
[17636] 25 Nov 16:55:44.413 * Discarding previously cached master state.
```

Replication

On Replica

```
[17636] 25 Nov 16:55:44.413 * MASTER MODE enabled (user request)
```

Replication

On Replica

```
[17636] 25 Nov 16:55:44.413 * MASTER MODE enabled (user request)
```

SLAVEOF NO ONE = MASTER MODE

On Replica; Master Down Failed Replication

On Replica; Master Down Failed Replication

```
[17636] 25 Nov 16:59:51.698 * MASTER <-> SLAVE sync: Finished with success
```

Replication Management Sentinel

Replication Management Sentinel

Beta since June 2012, rewritten November 2013.

Replication Management Sentinel

Beta since June 2012, rewritten November 2013.

Manages redis replication and availability with redis.

Replication Management Sentinel

Beta since June 2012, rewritten November 2013.

Manages redis replication and availability with redis.

Provides auto-promotion of replicas.

Replication Management Sentinel

Beta since June 2012, rewritten November 2013.

Manages redis replication and availability with redis.

Provides auto-promotion of replicas.

Provides a service where you ask for the current redis master servers.

Replication Management Sentinel

Beta since June 2012, rewritten November 2013.

Manages redis replication and availability with redis.

Provides auto-promotion of replicas.

Provides a service where you ask for the current redis master servers.

```
SENTINEL GET-MASTER-ADDR-BY-NAME userDB
```

Replication Management Sentinel

Replication Management Sentinel

Alternative to hard-coding database IPs in your config files.

Replication Management Sentinel

Alternative to hard-coding database IPs in your config files.

Just ask Redis Sentinel for the current write master address.

Replication Management Sentinel

Alternative to hard-coding database IPs in your config files.

Just ask Redis Sentinel for the current write master address.

Sentinel auto-notifies all clients about replica promotions to master.

Replication Management Sentinel

Alternative to hard-coding database IPs in your config files.

Just ask Redis Sentinel for the current write master address.

Sentinel auto-notifies all clients about replica promotions to master.

Direct knowledge of DB state.

Replication Management Sentinel

Alternative to hard-coding database IPs in your config files.

Just ask Redis Sentinel for the current write master address.

Sentinel auto-notifies all clients about replica promotions to master.

Direct knowledge of DB state.

No waiting for timeouts or load balancers to switch over.

Replication

Replication

That's it for replication.

Replication

That's it for replication.

Replicas can replicate other replicas.

Replication

That's it for replication.

Replicas can replicate other replicas.

The replica instance stays in-sync (async) with its master node.

Replication

That's it for replication.

Replicas can replicate other replicas.

The replica instance stays in-sync (async) with its master node.

The replica is an exact copy of its master as long as `slave-read-only` **yes**

Clustering

preview

Clustering

Almost ready.

preview

Clustering

Almost ready.

Under development for three years.

Clustering

Almost ready.

Under development for three years.

Distributes keys across master instances.

Clustering

Almost ready.

Under development for three years.

Distributes keys across master instances.

Each master instance has multiple identical replicas.

Clustering

Almost ready.

Under development for three years.

Distributes keys across master instances.

Each master instance has multiple identical replicas.

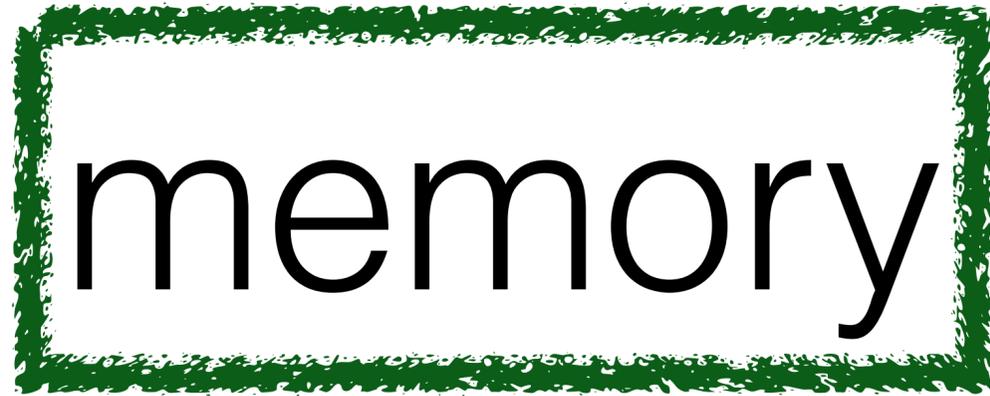
Replicas sanely promote to master if failure is detected.

Capacity Planning

(and capacity-related failure scenarios)

Capacity Planning

(and capacity-related failure scenarios)



memory

Capacity Planning

(and capacity-related failure scenarios)

memory

network

Capacity Planning

(and capacity-related failure scenarios)

memory

network

cpu

Capacity Planning

(and capacity-related failure scenarios)

memory

network

cpu

disk

Capacity Planning

Capacity Planning

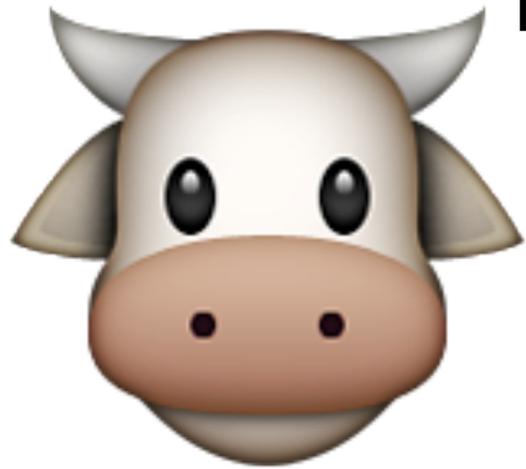
memory

Capacity Planning

memory

牛

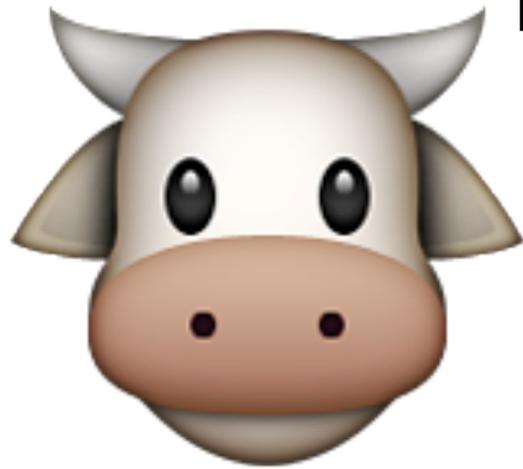
Capacity Planning



memory

牛

Capacity Planning

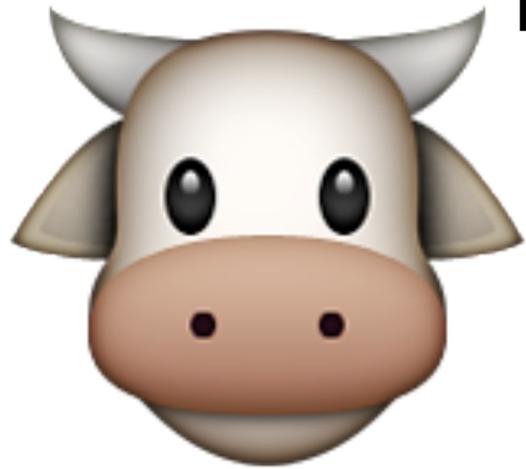


memory

牛

COW

Capacity Planning



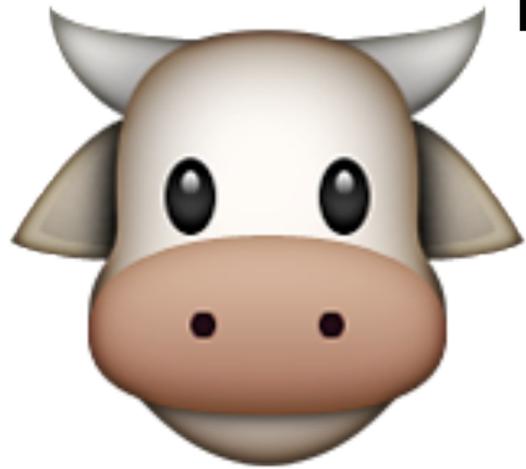
memory

牛

COW

Copy On Write

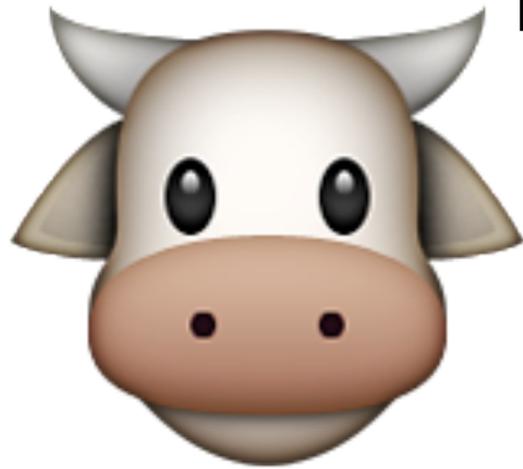
Capacity Planning



memory

牛

Capacity Planning

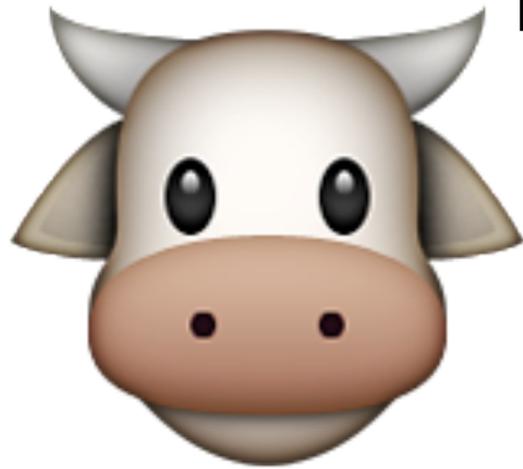


memory

牛

64 GB
RAM DB

Capacity Planning



memory

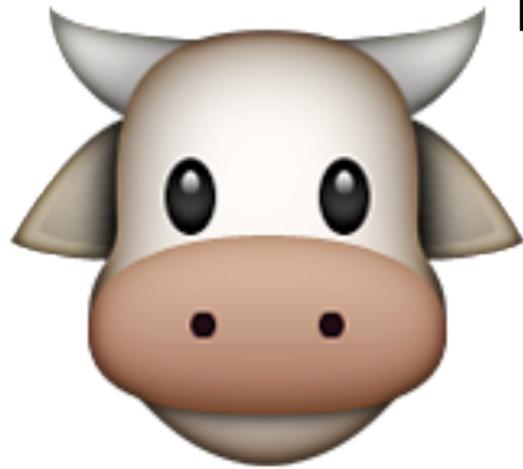
牛

64 GB
RAM DB

64 GB
RAM DB

BGSAVE

Capacity Planning



memory

牛

64 GB
RAM DB

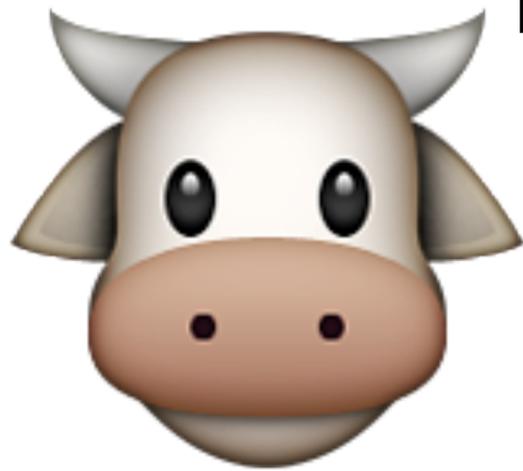


64 GB
RAM DB

references
all data at time
of BGSAVE

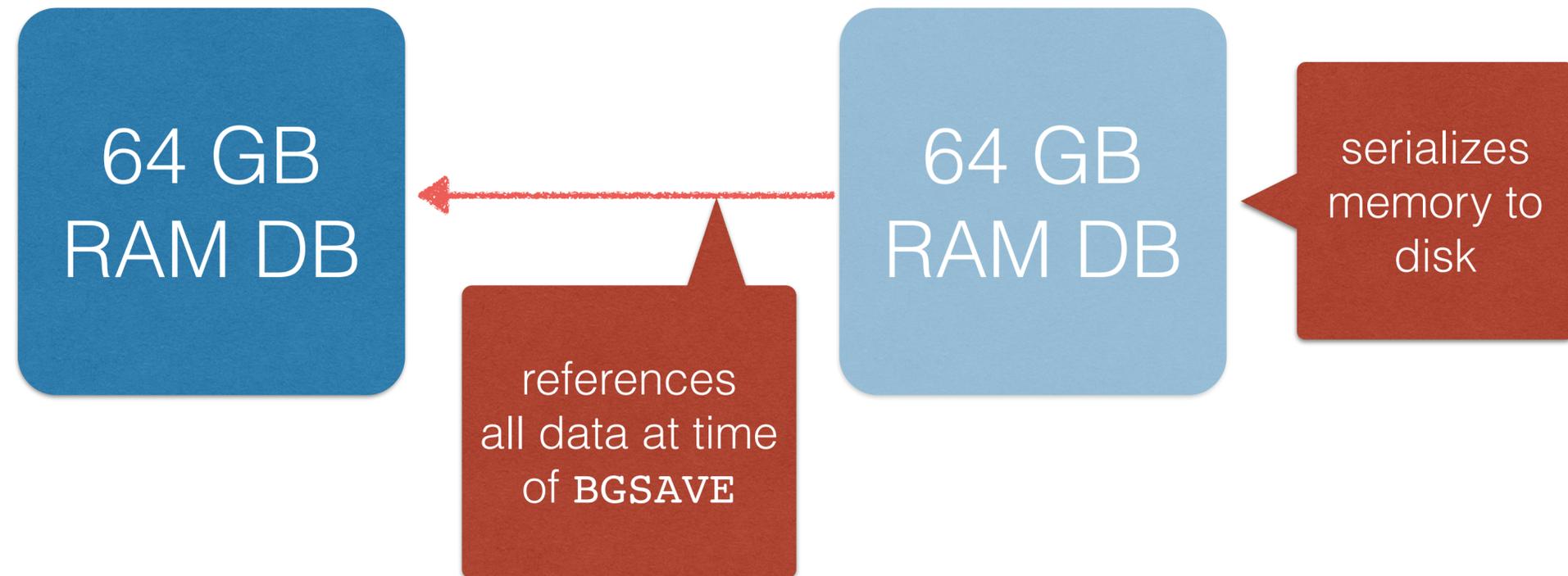
BGSAVE

Capacity Planning



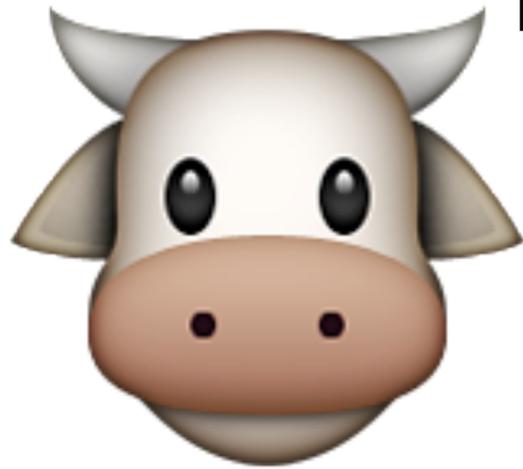
memory

牛



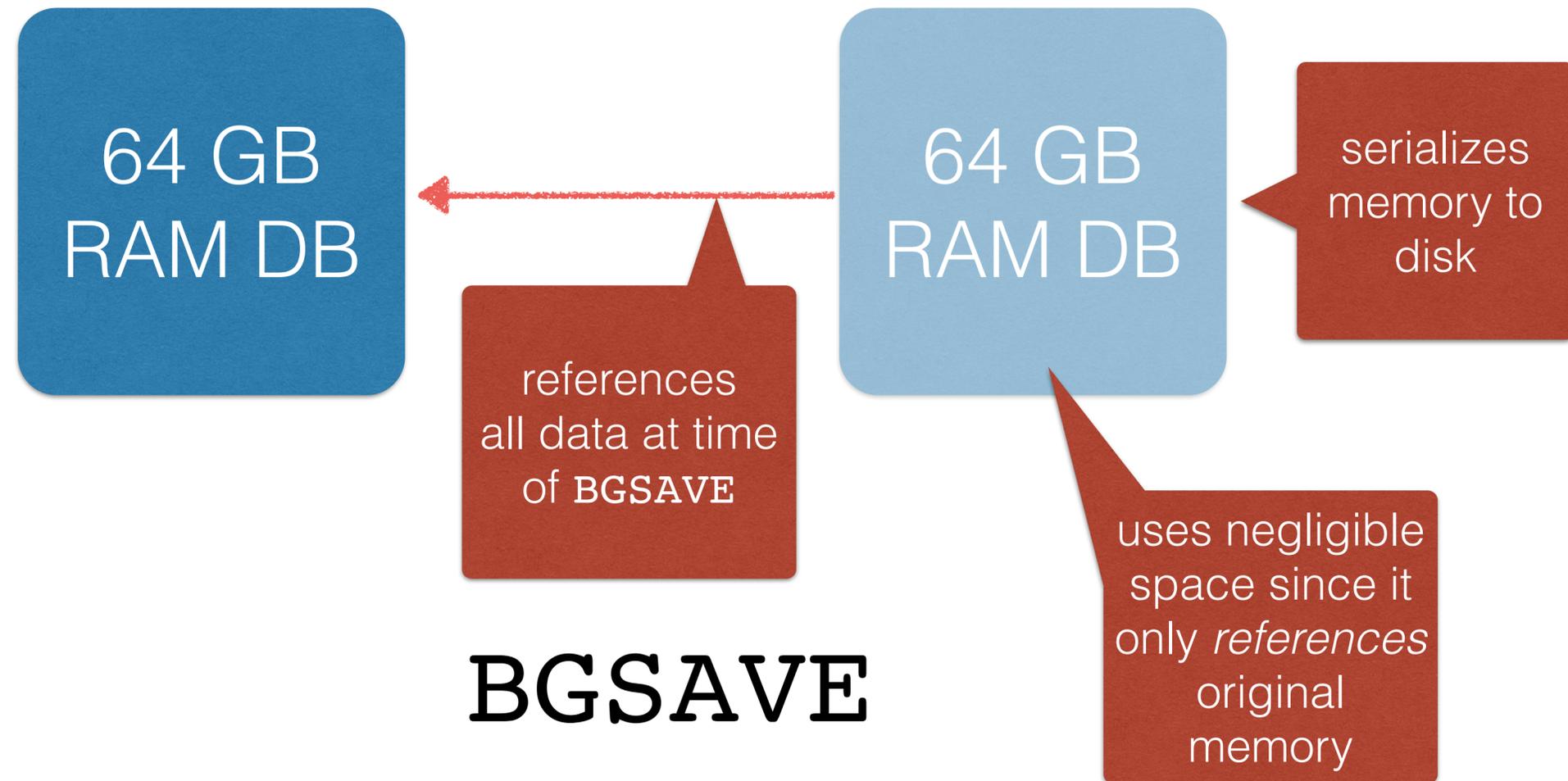
BGSAVE

Capacity Planning

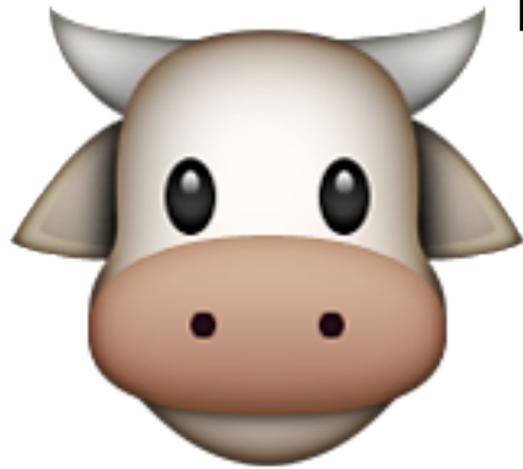


memory

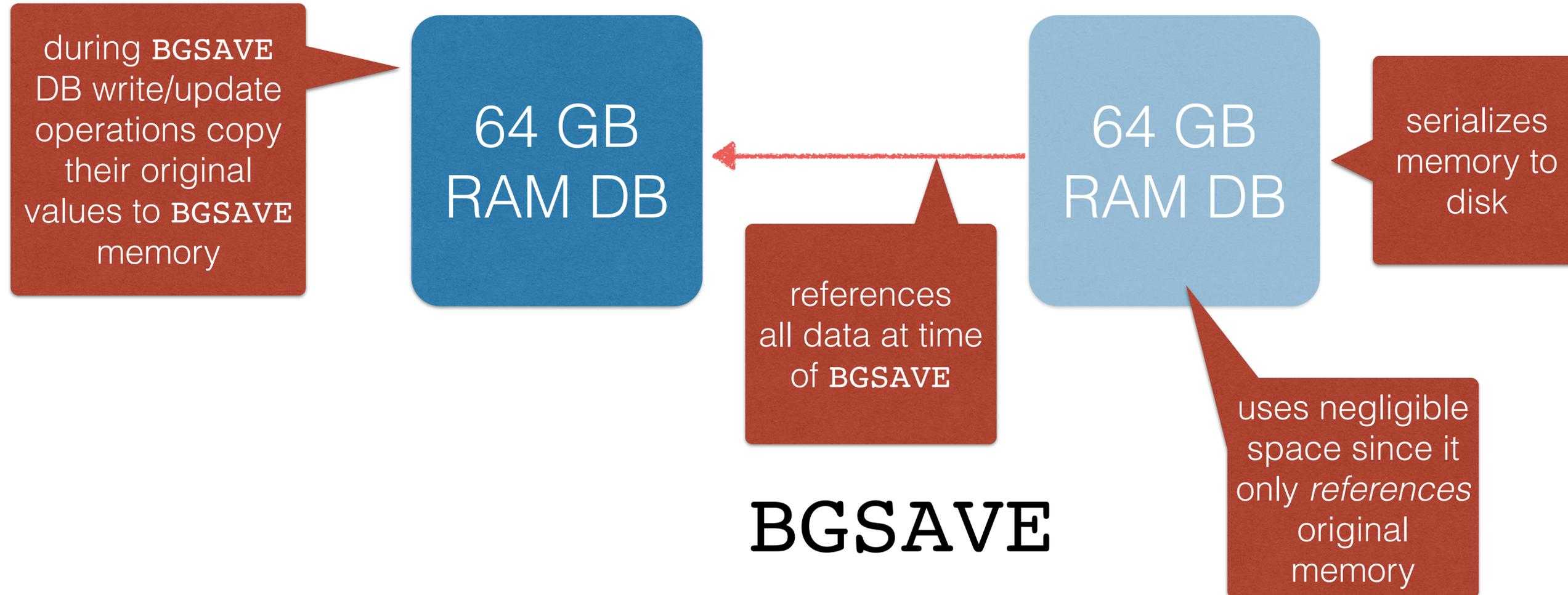
牛



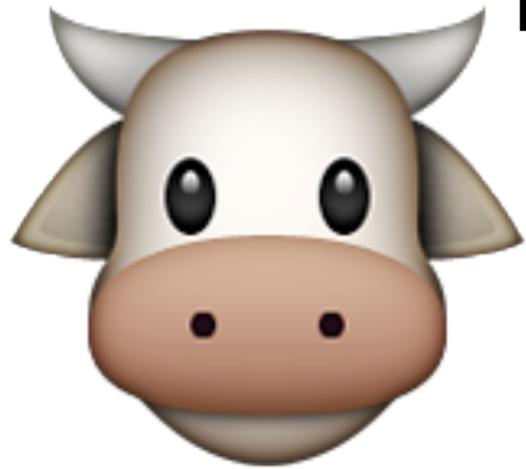
Capacity Planning



memory



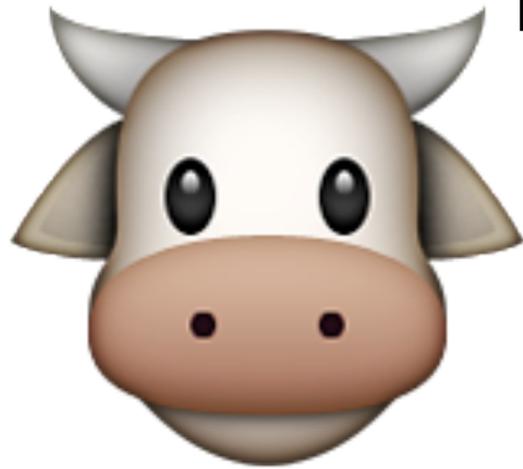
Capacity Planning



memory

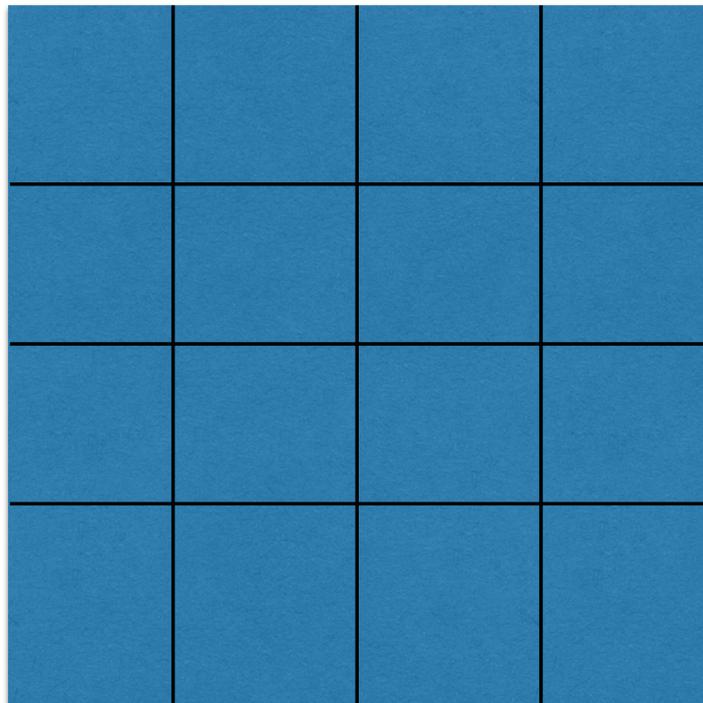
牛

Capacity Planning



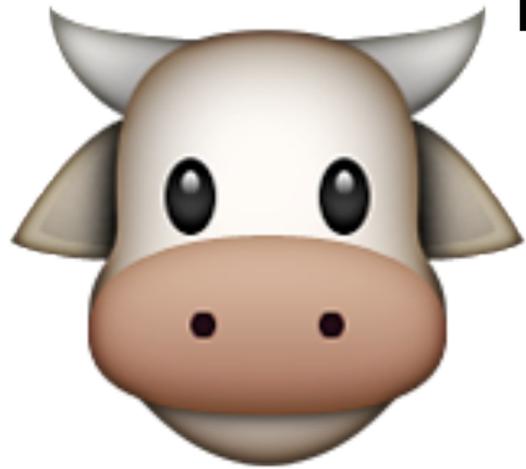
memory

牛

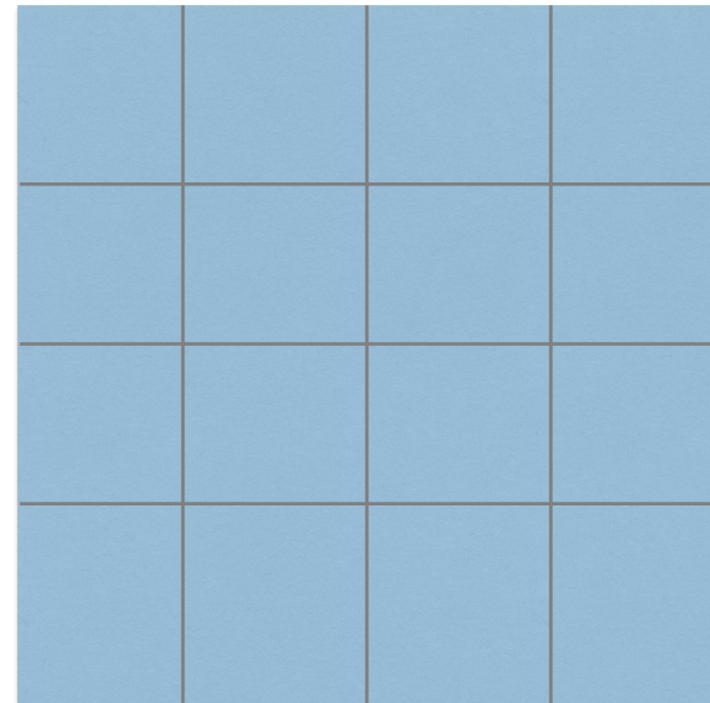
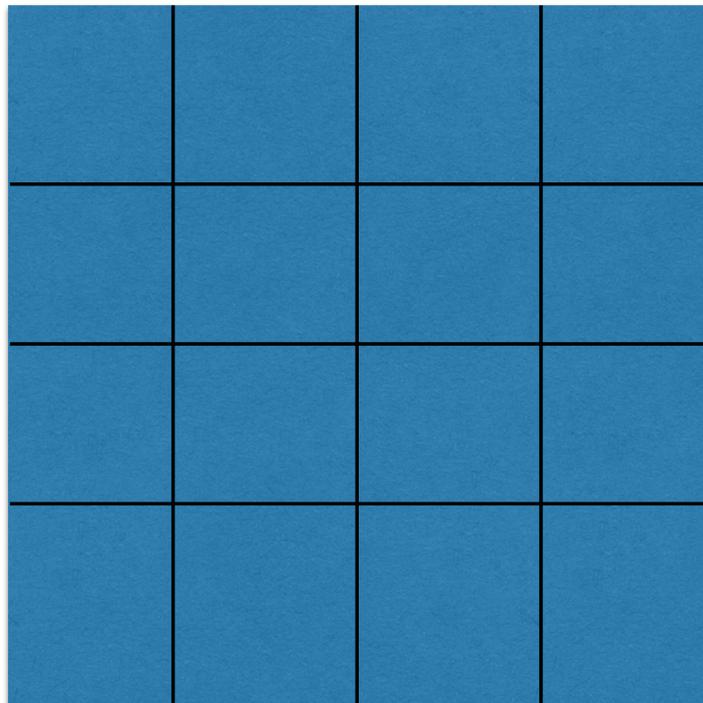
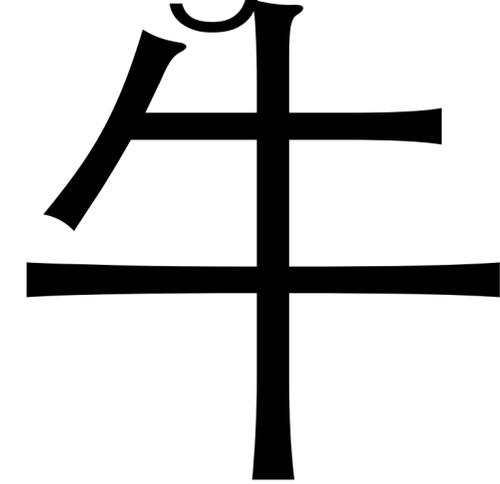


Something writes 4K to DB during BGSAVE

Capacity Planning

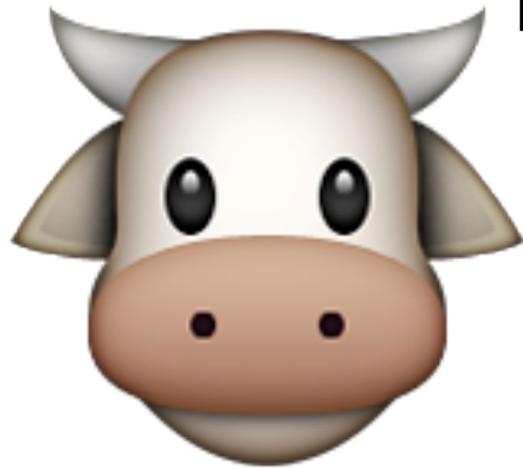


memory

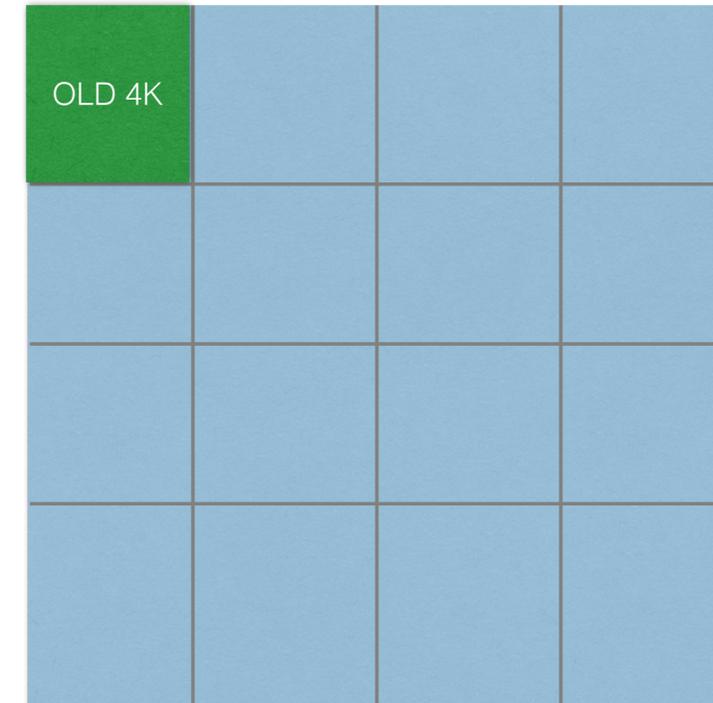
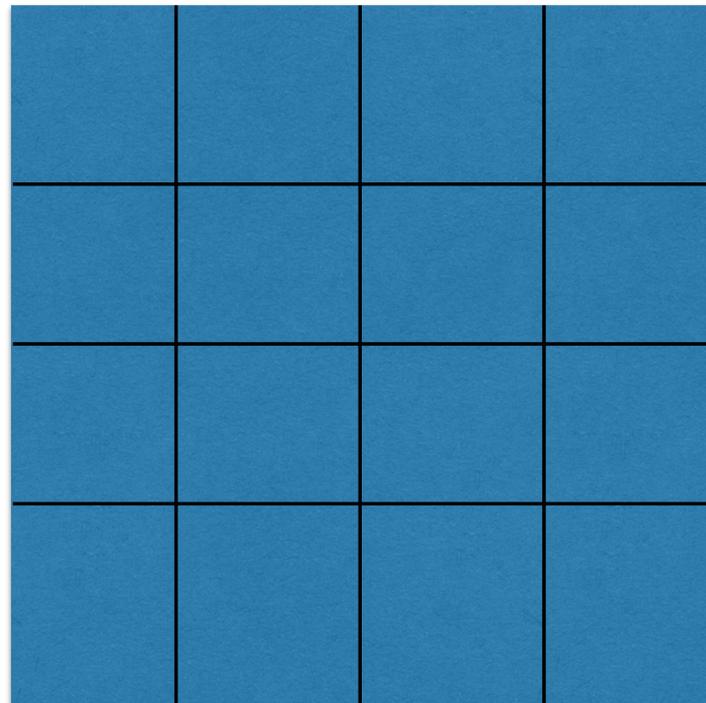
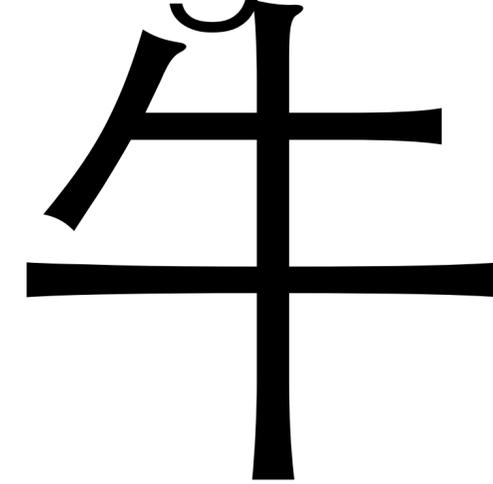


Something writes 4K to DB during **BGSAVE**

Capacity Planning

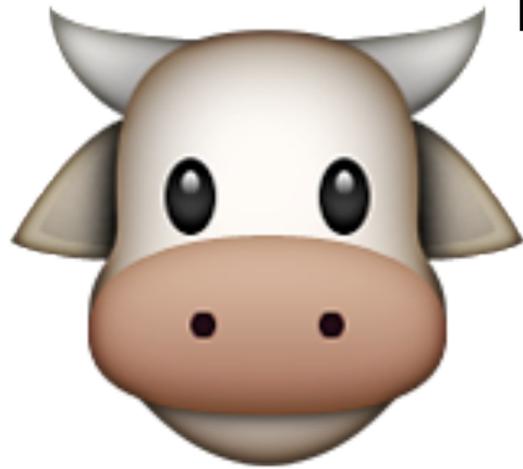


memory

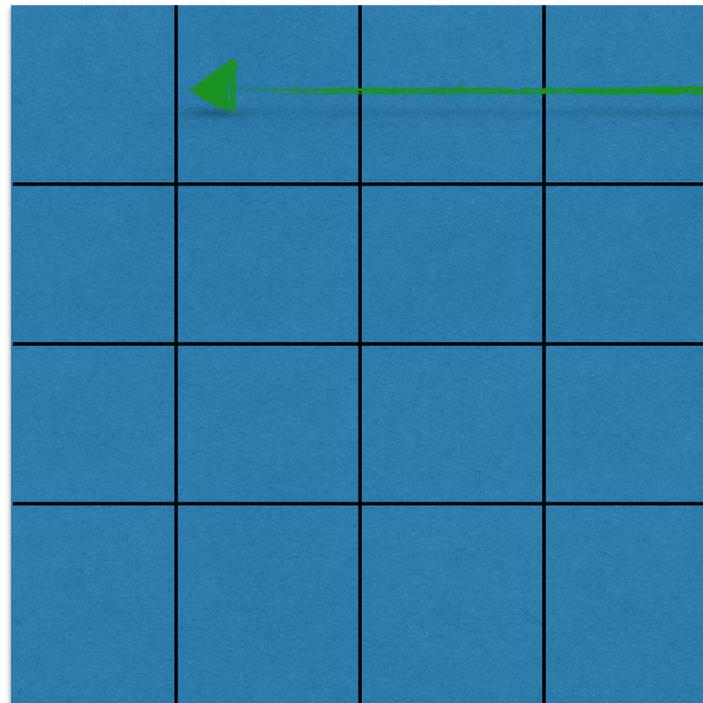
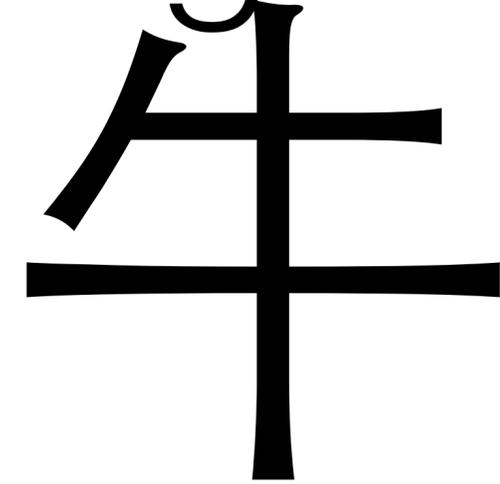


Something writes 4K to DB during **BGSAVE**

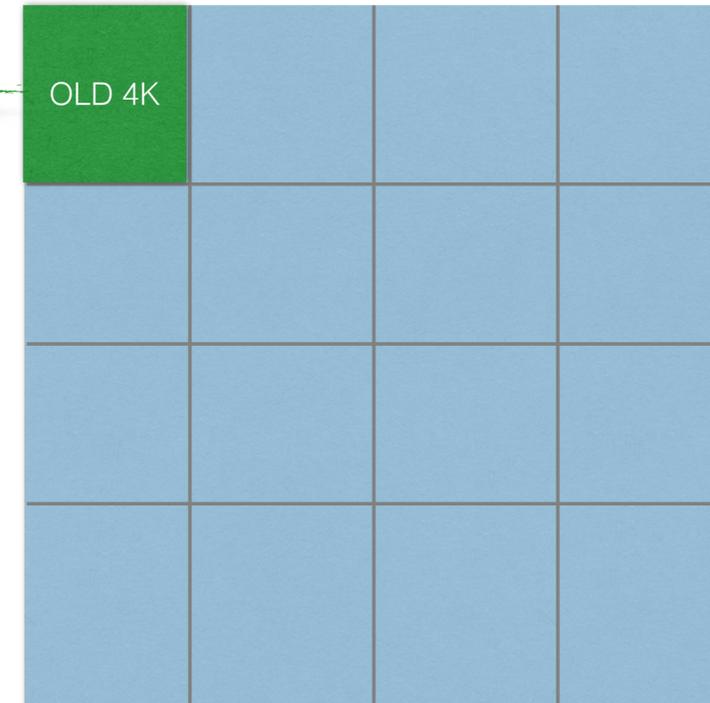
Capacity Planning



memory

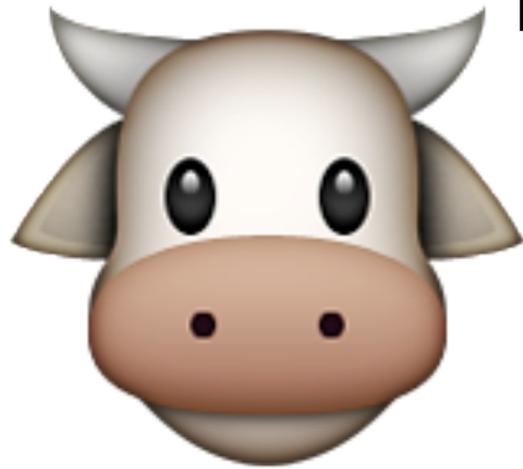


copied from

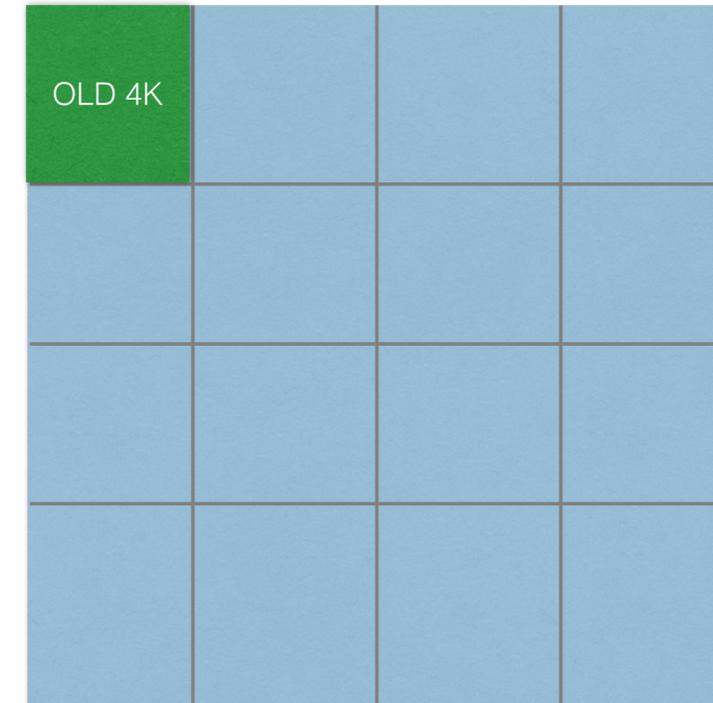
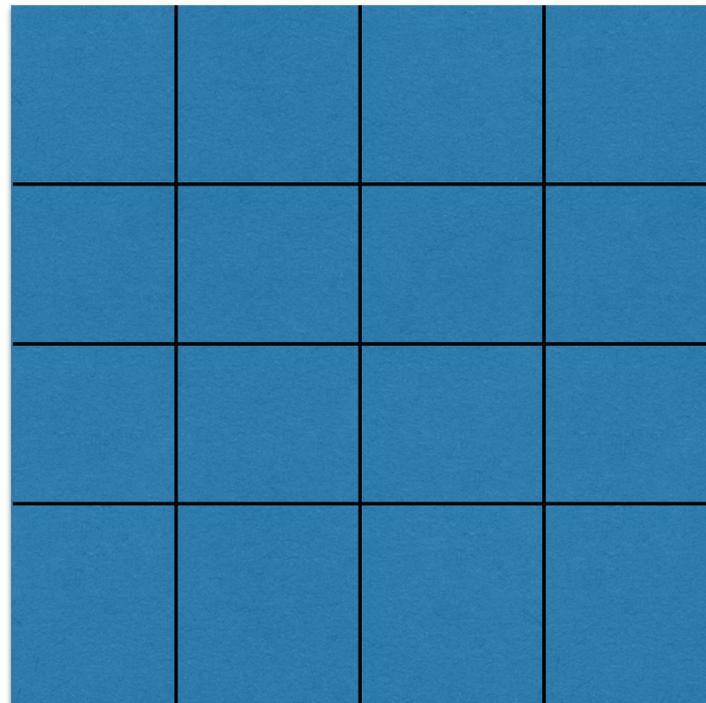
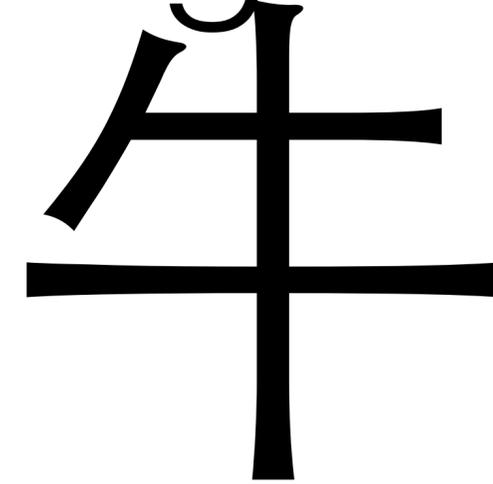


Something writes 4K to DB during **BGSAVE**

Capacity Planning

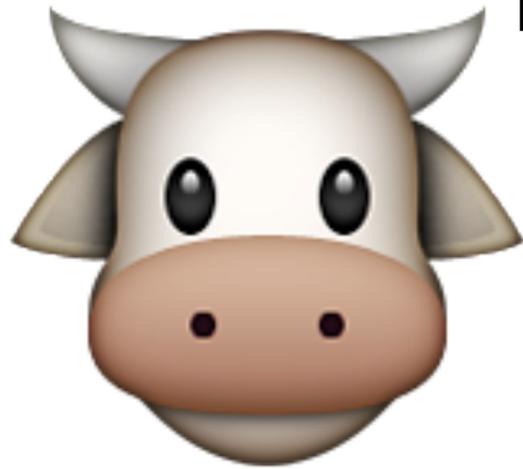


memory

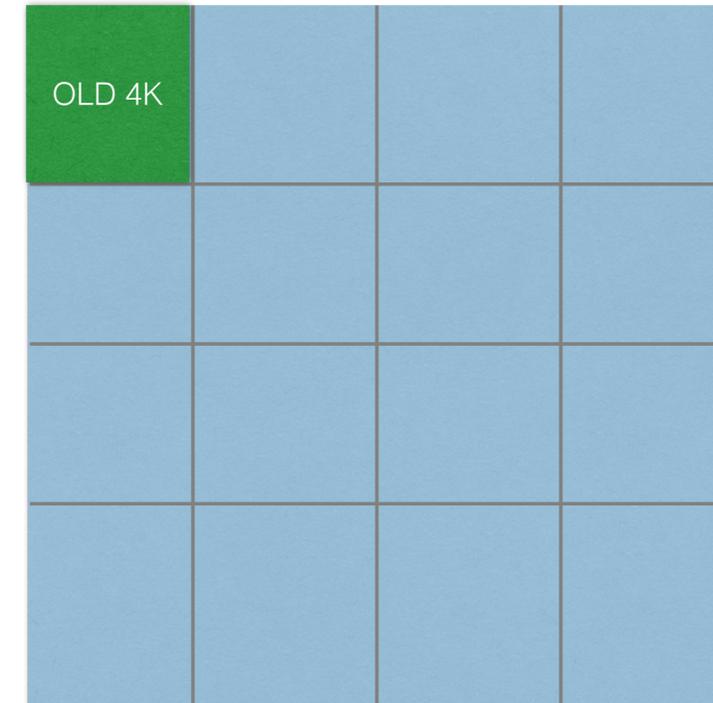
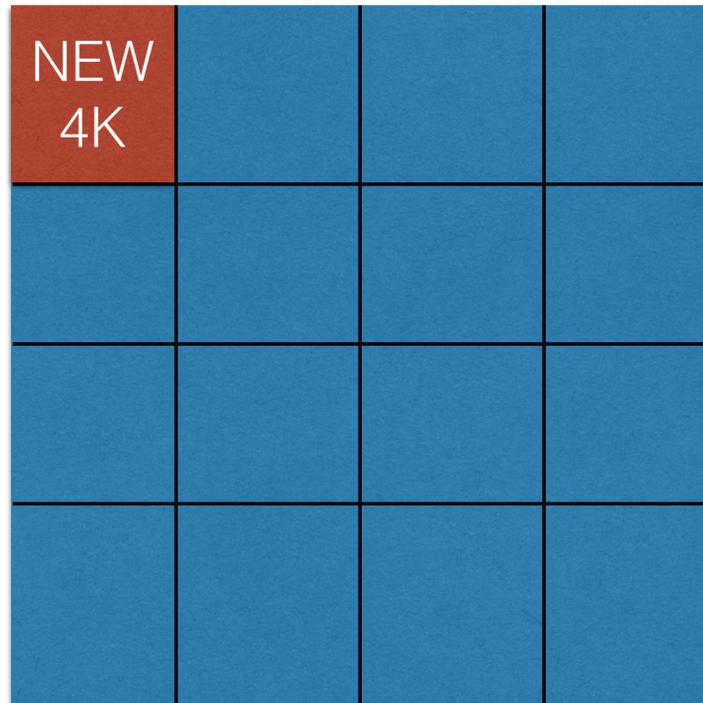
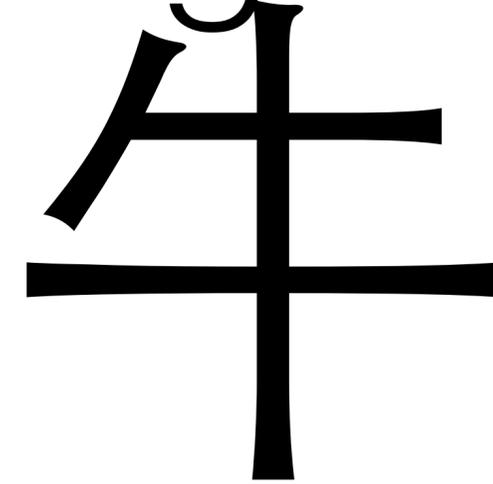


Something writes 4K to DB during **BGSAVE**

Capacity Planning

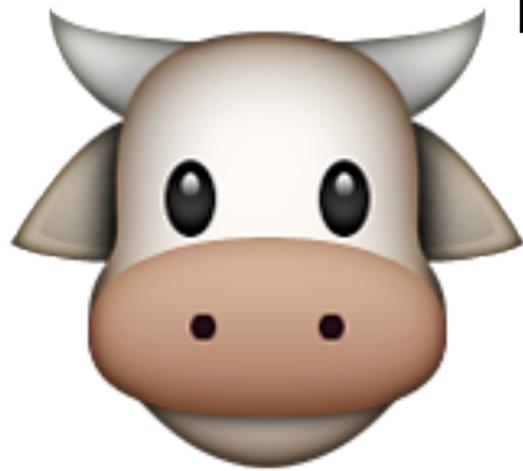


memory

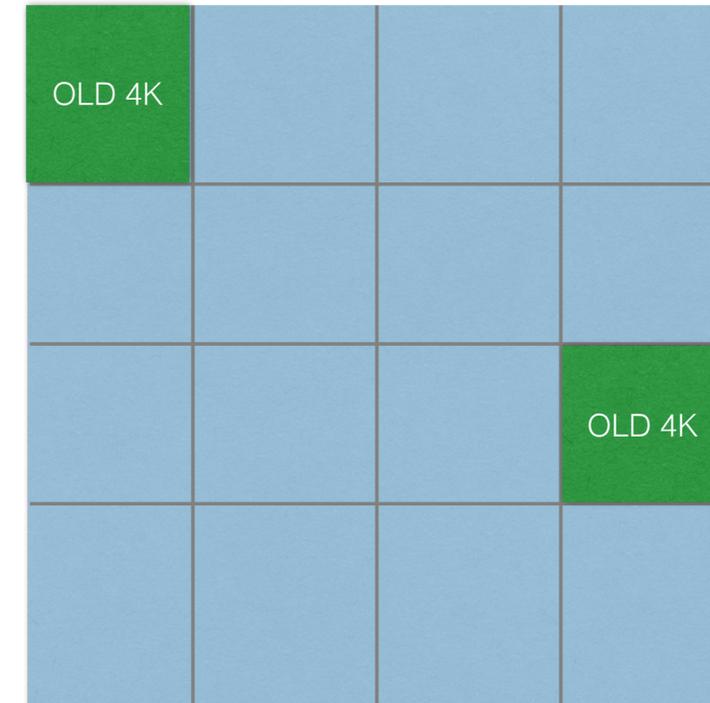
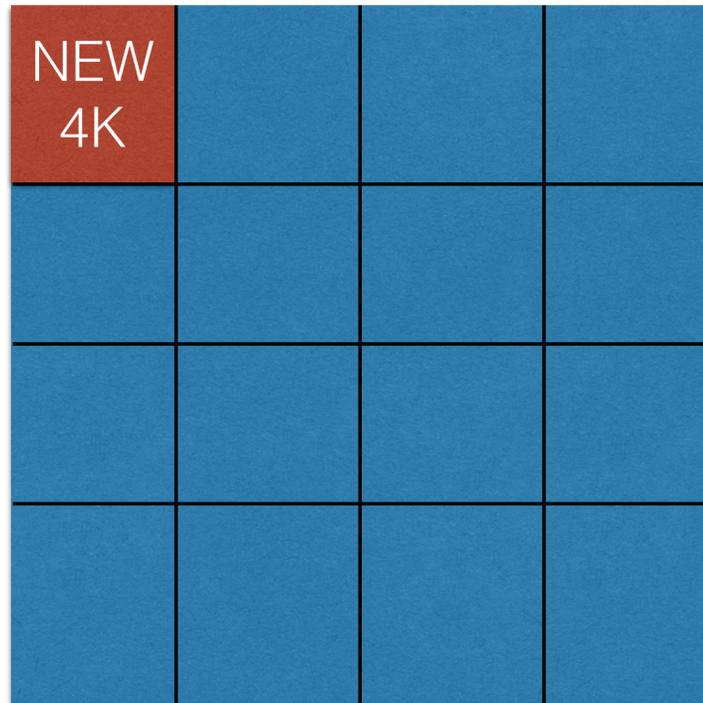
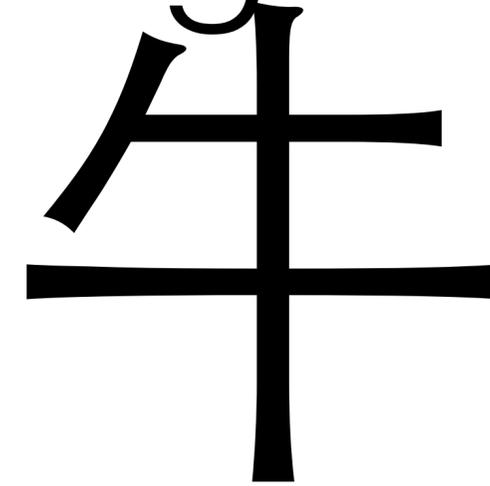


Something writes 4K to DB during **BGSAVE**

Capacity Planning

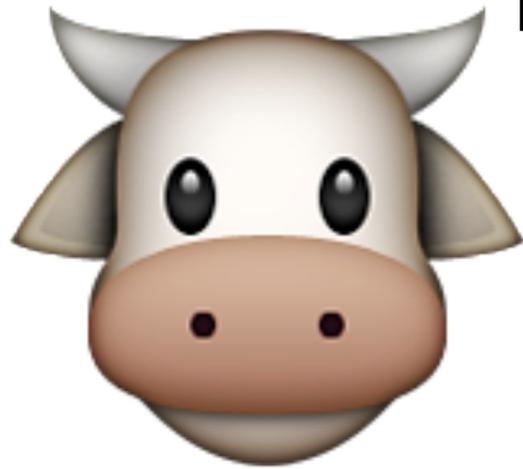


memory

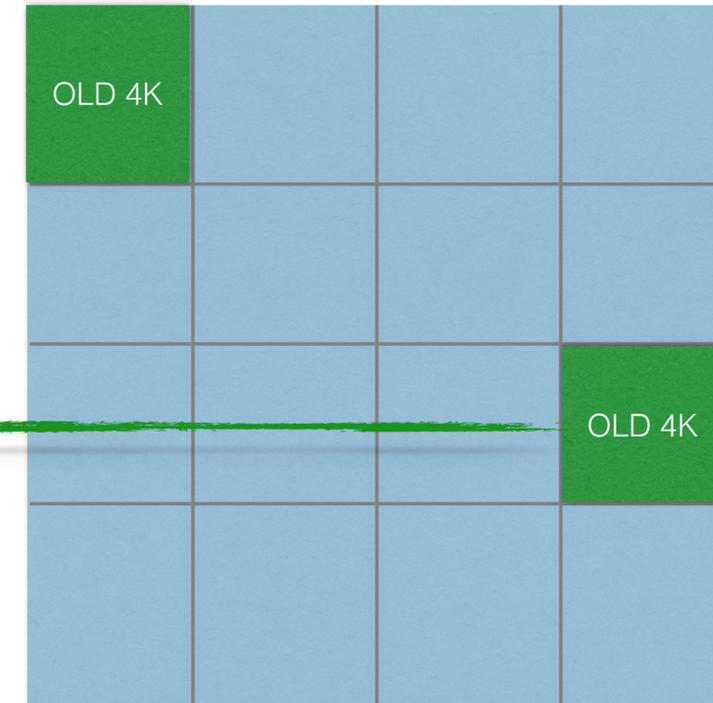
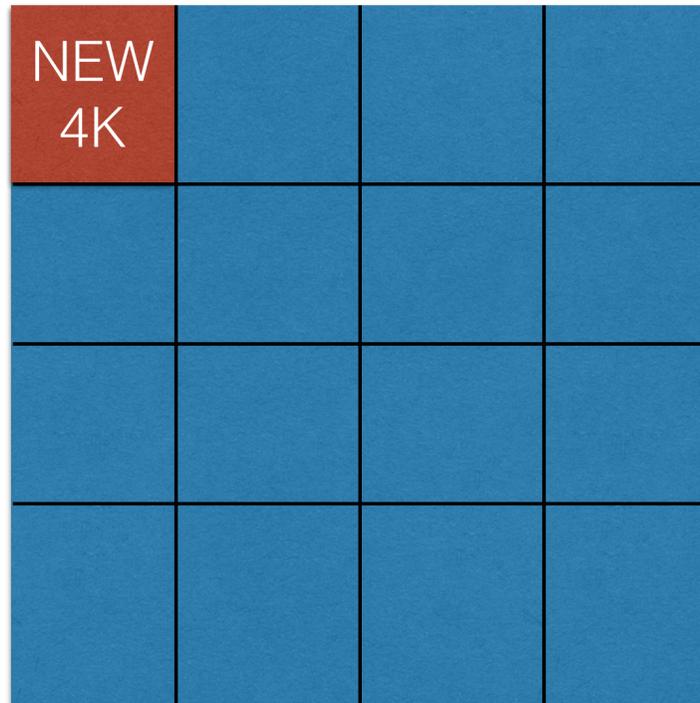
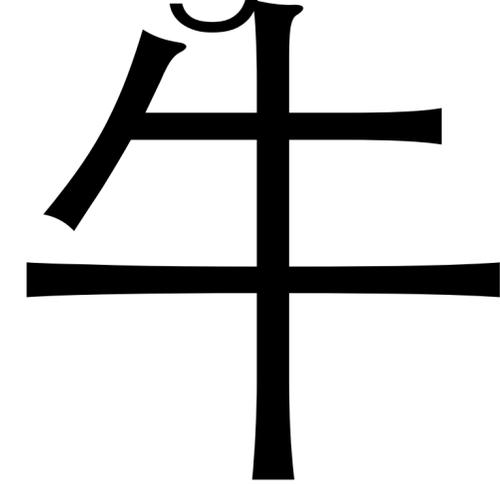


Something writes 4K to DB during **BGSAVE**

Capacity Planning



memory

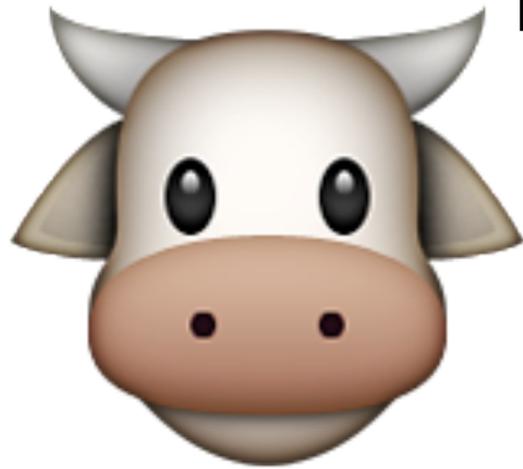


copied from

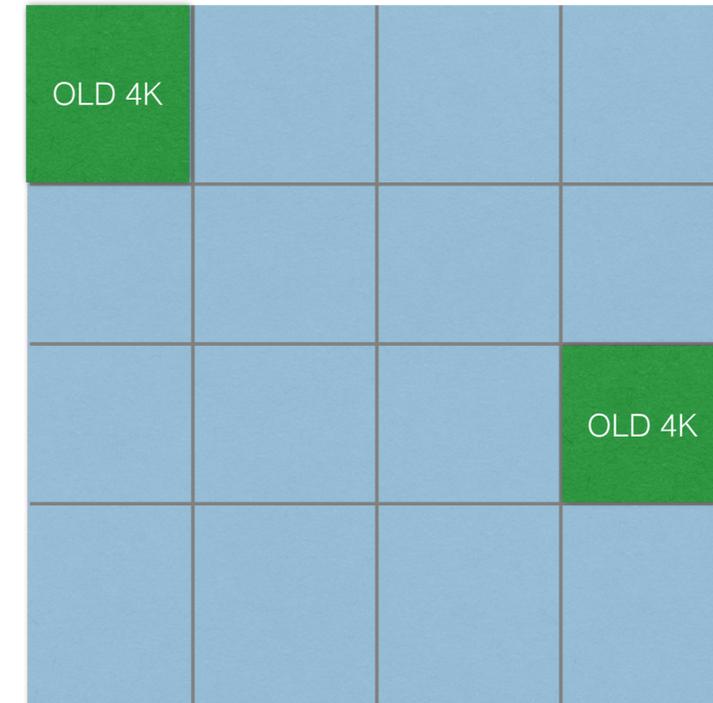
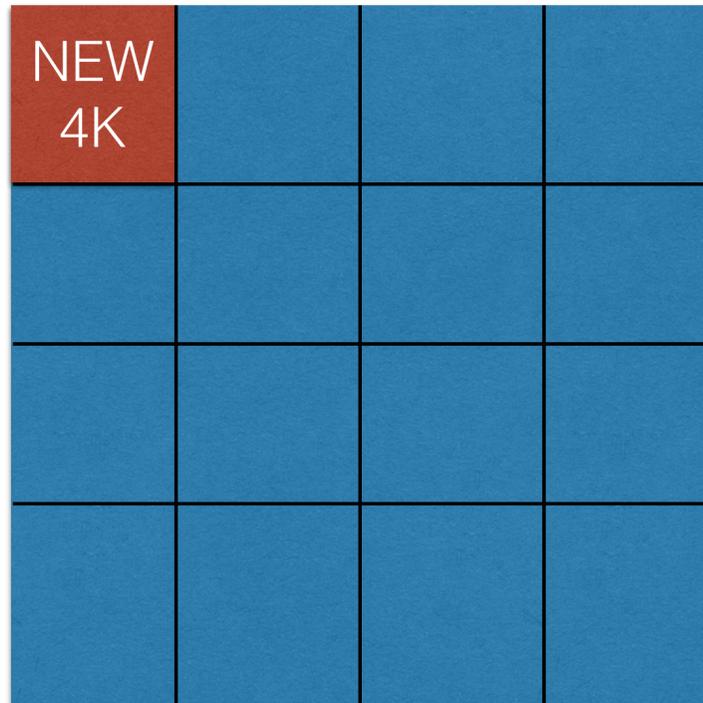
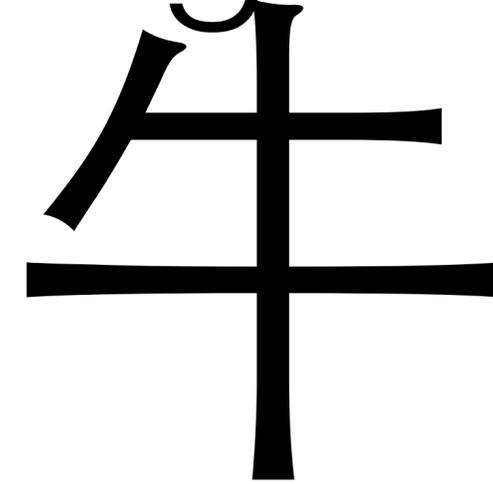


Something writes 4K to DB during **BGSAVE**

Capacity Planning

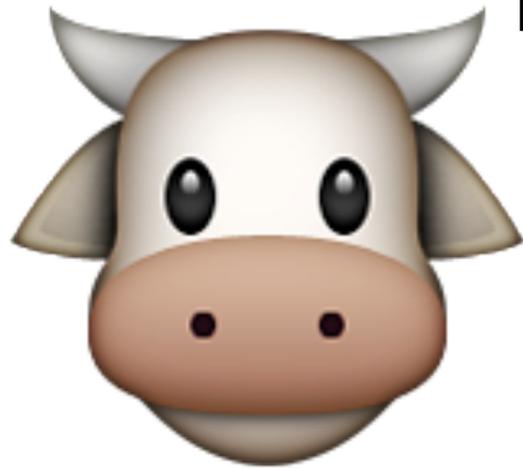


memory

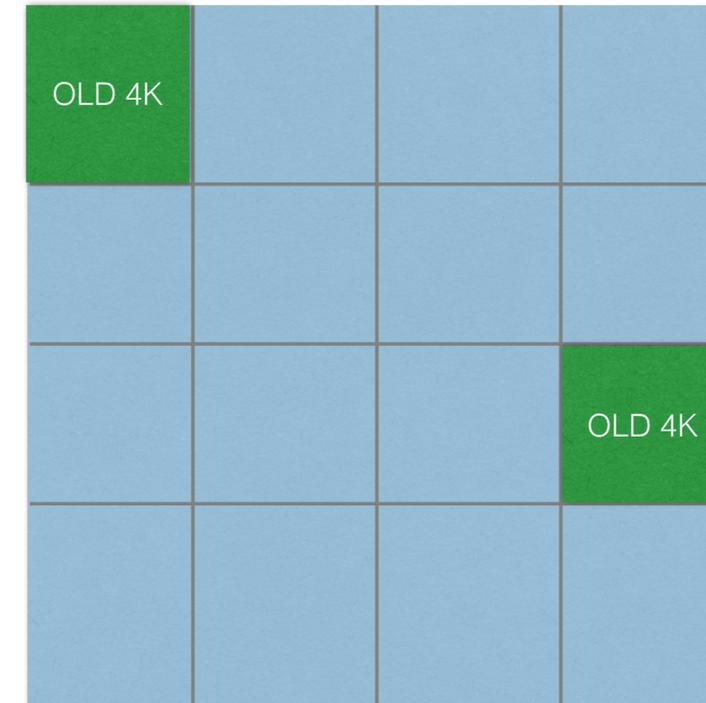
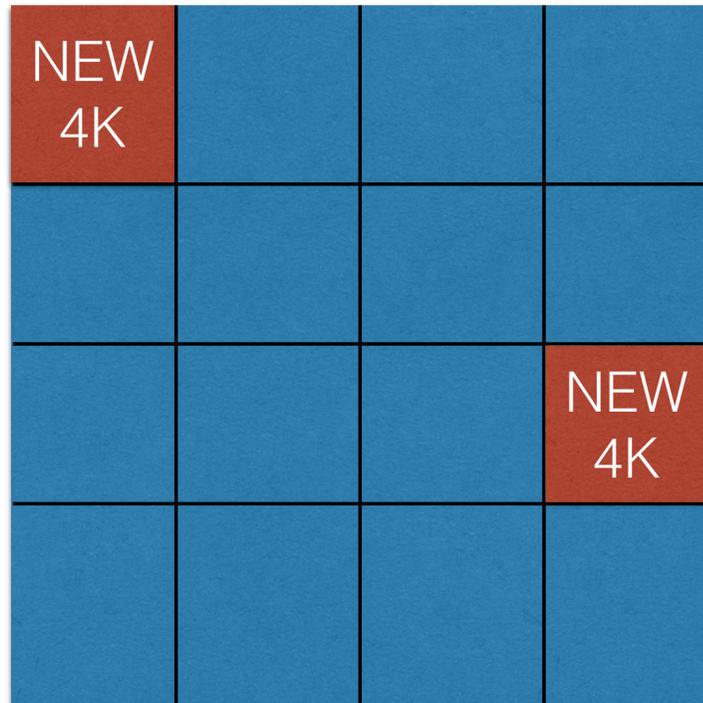
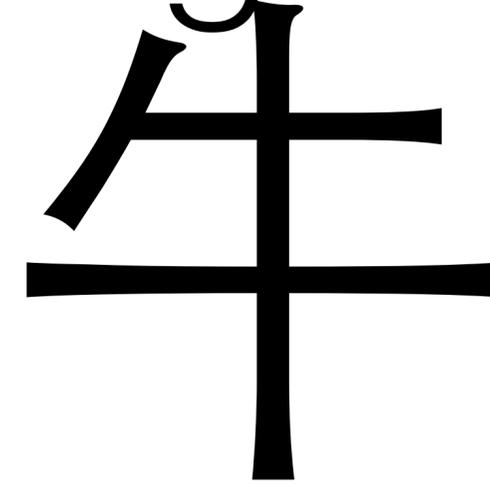


Something writes 4K to DB during **BGSAVE**

Capacity Planning

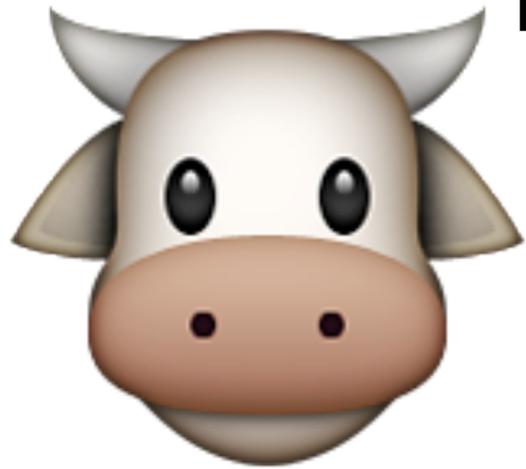


memory



Something writes 4K to DB during **BGSAVE**

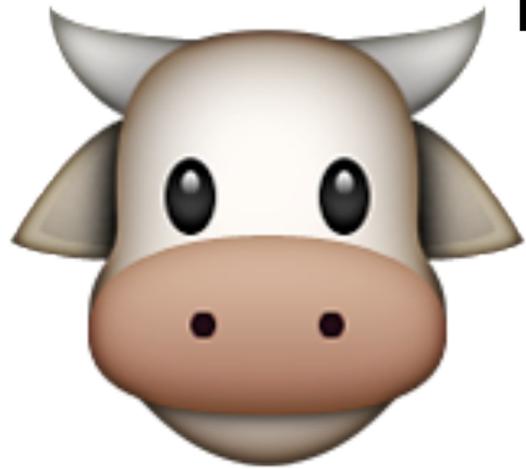
Capacity Planning



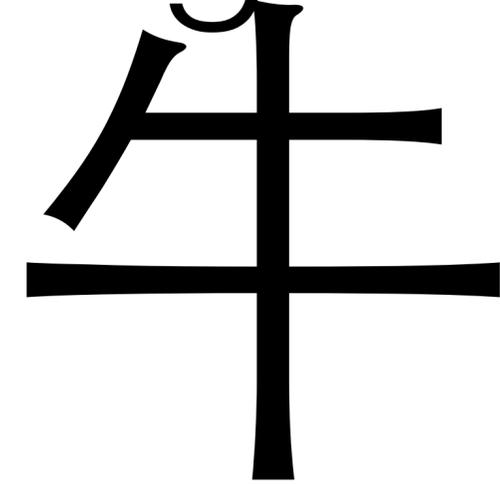
memory

牛

Capacity Planning

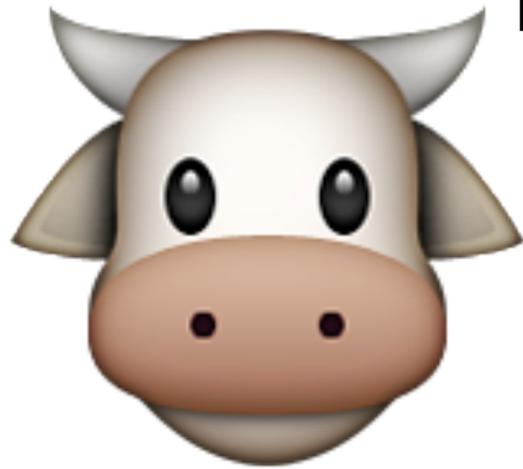


memory

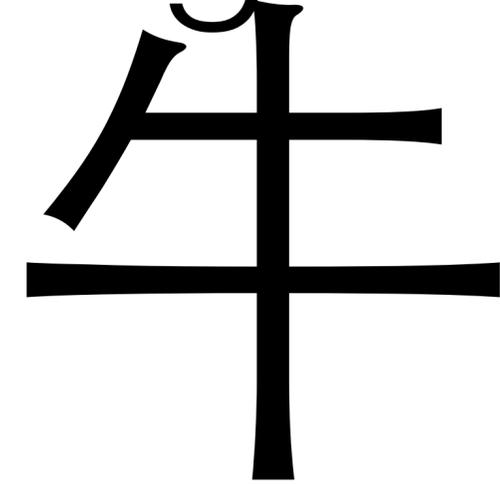


Implications

Capacity Planning



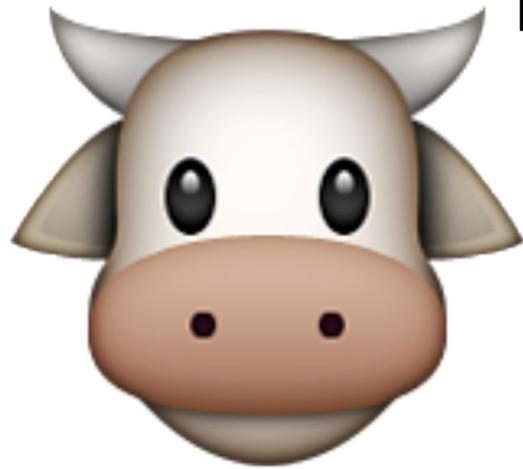
memory



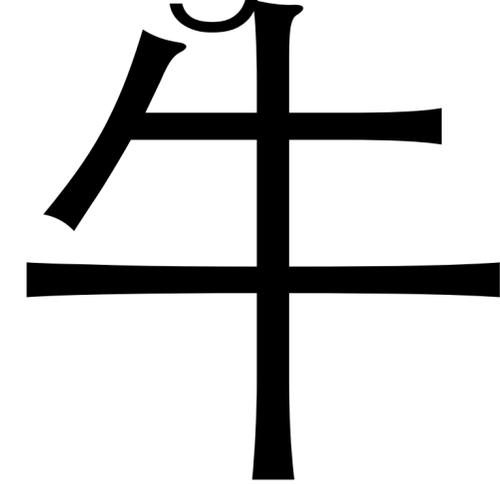
Implications

OS must enable memory overcommit

Capacity Planning



memory

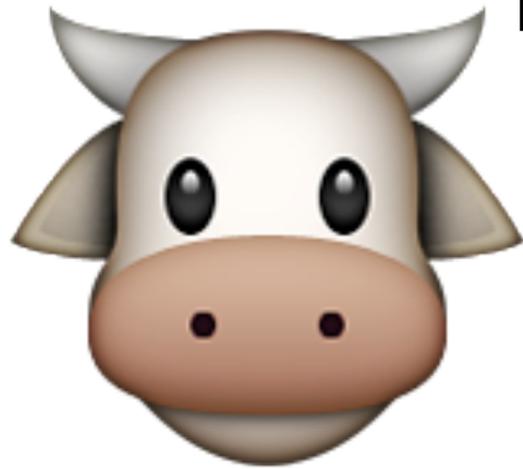


Implications

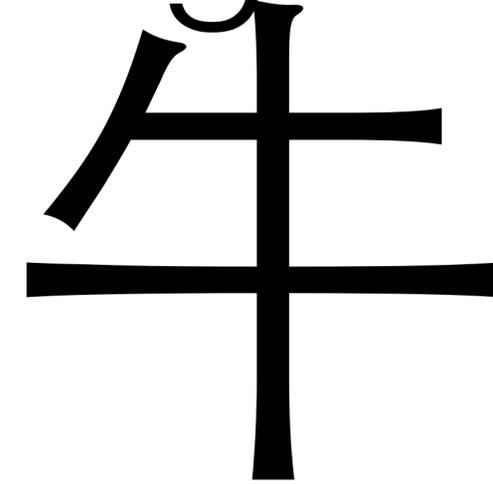
OS must enable memory overcommit

64 GB usage + 64 GB fork \neq 128 GB physical usage

Capacity Planning



memory



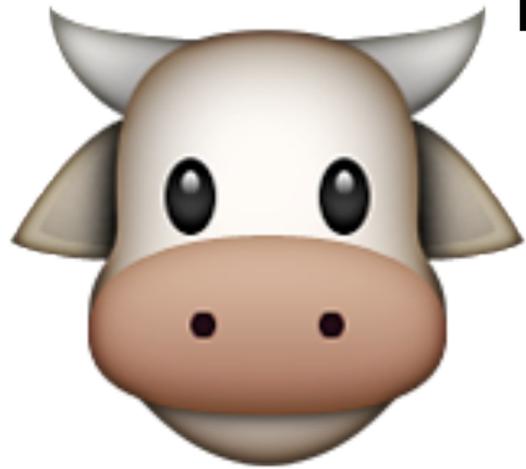
Implications

OS must enable memory overcommit

64 GB usage + 64 GB fork \neq 128 GB physical usage

```
[5358] 14 Nov 11:25:09.466 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
```

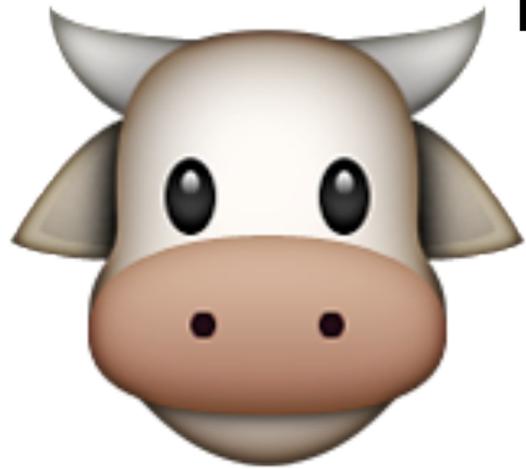
Capacity Planning



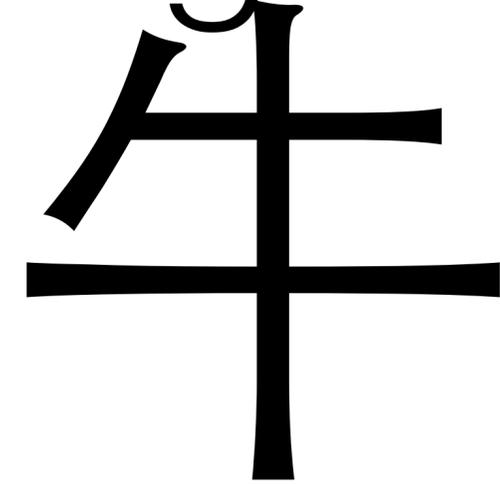
memory

牛

Capacity Planning

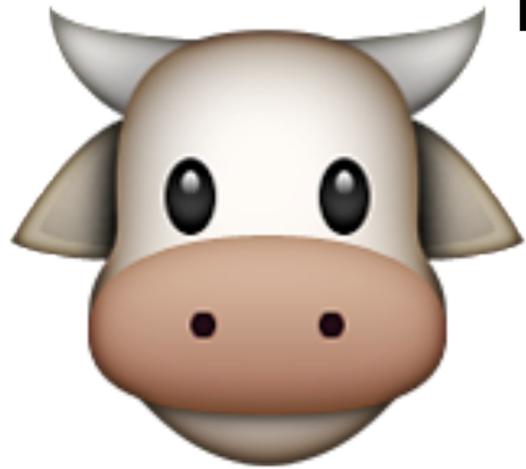


memory

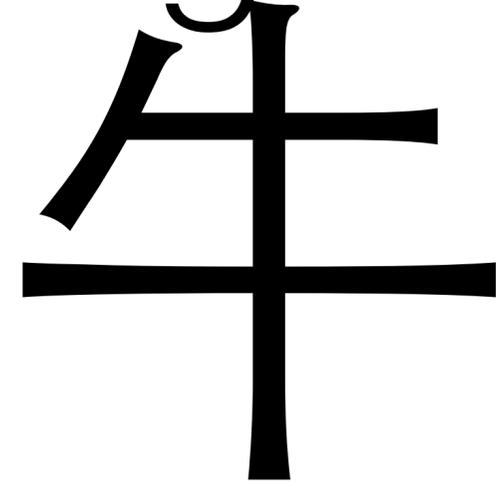


Implications

Capacity Planning



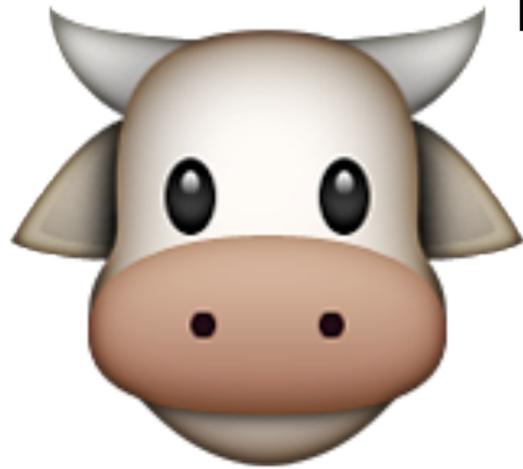
memory



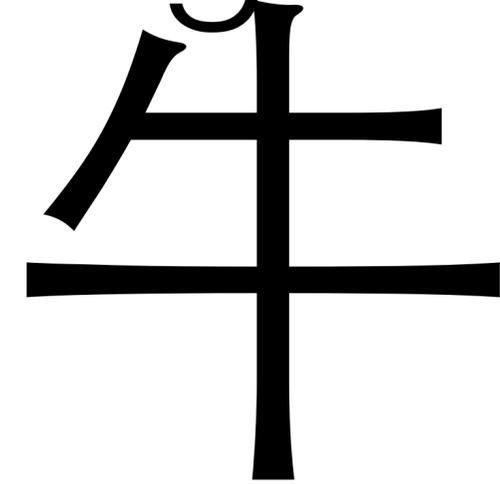
Implications

High-write DBs near memory limits can break

Capacity Planning



memory

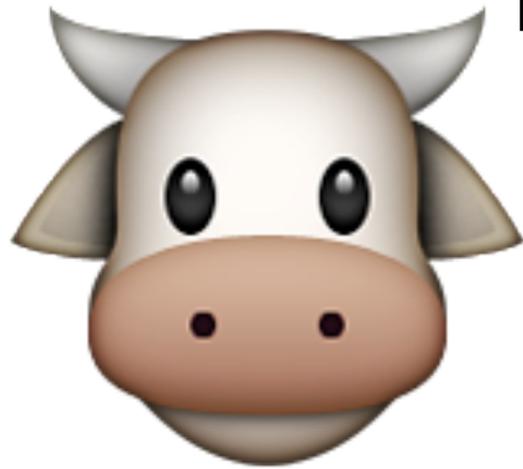


Implications

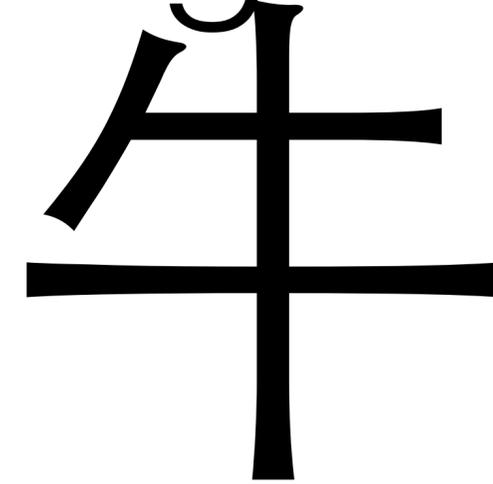
High-write DBs near memory limits can break

64 GB usage + 1 GB/sec updates + 2 sec **BGSAVE**

Capacity Planning



memory



Implications

High-write DBs near memory limits can break

64 GB usage + 1 GB/sec updates + 2 sec **BGSAVE**

64 GB memory + 2 GB  = 66 GB = OOM Killer

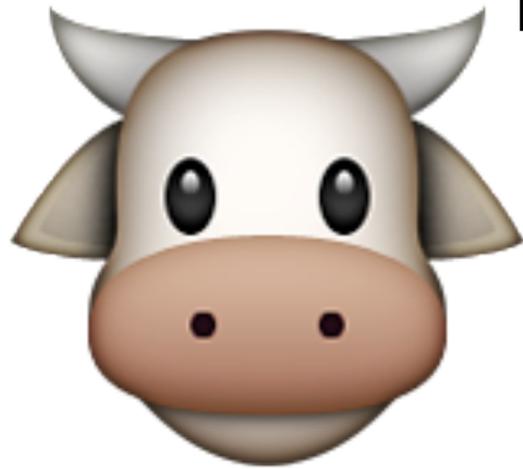
Capacity Planning



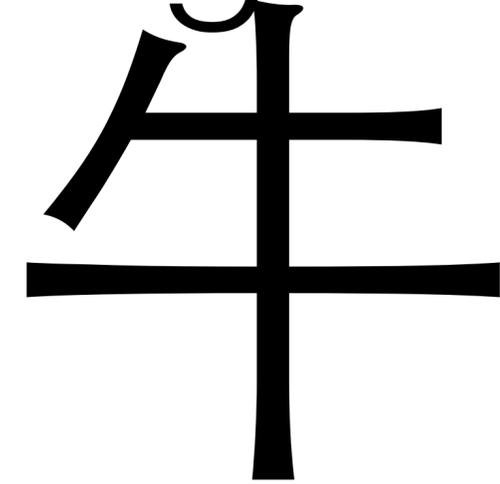
memory

牛

Capacity Planning

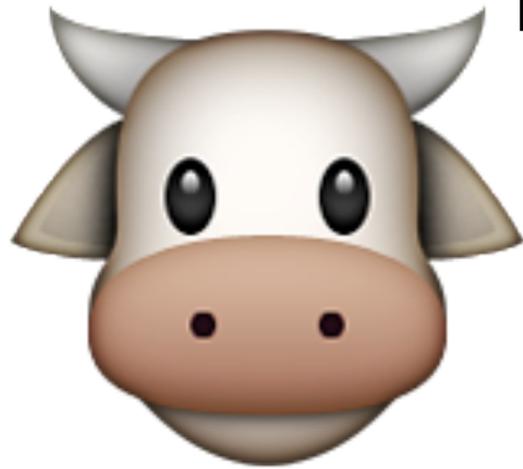


memory

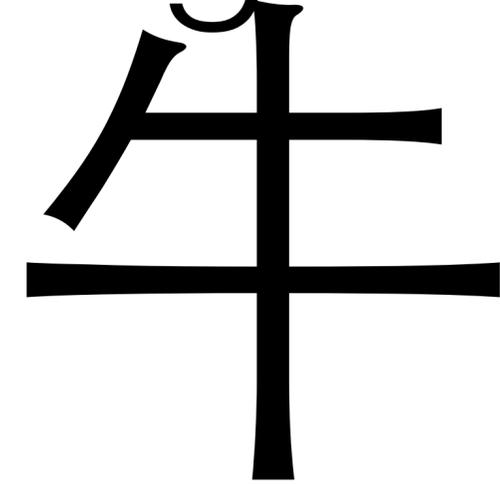


Note

Capacity Planning



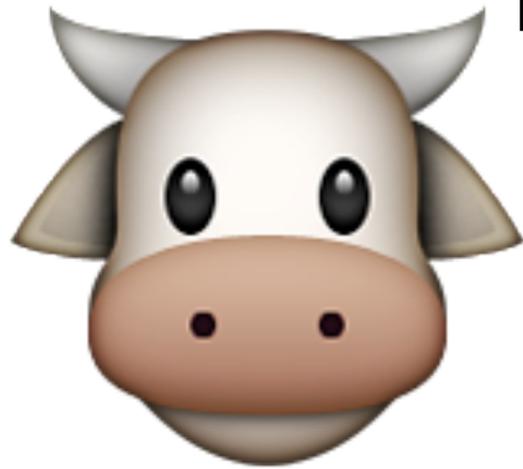
memory



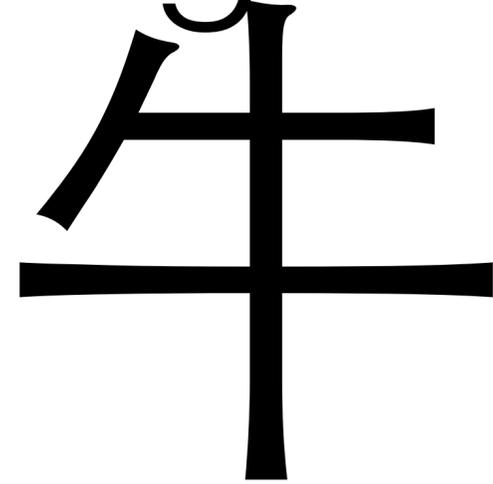
Note

Xen has horrible forking performance.

Capacity Planning



memory

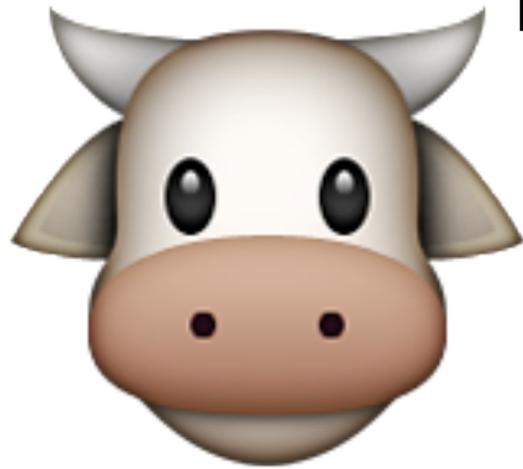


Note

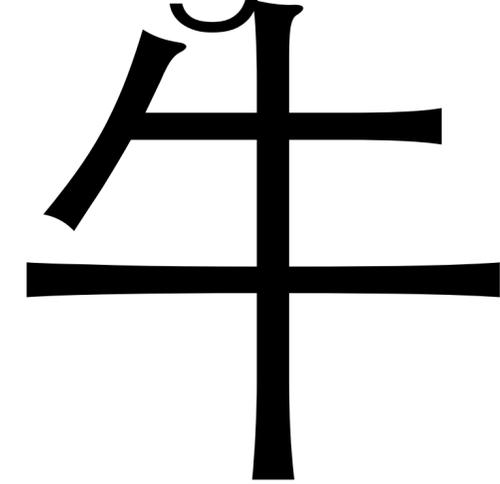
Xen has horrible forking performance.

Hardware fork latency: 80ms

Capacity Planning



memory



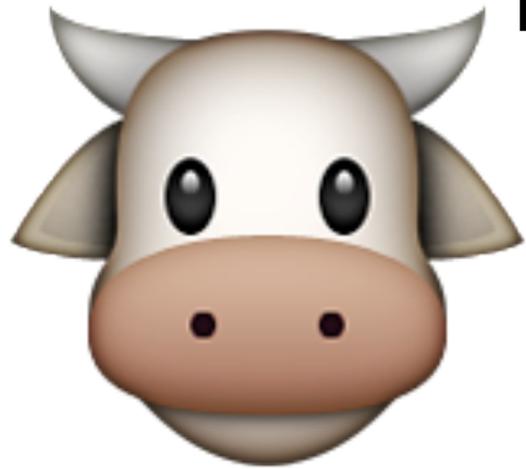
Note

Xen has horrible forking performance.

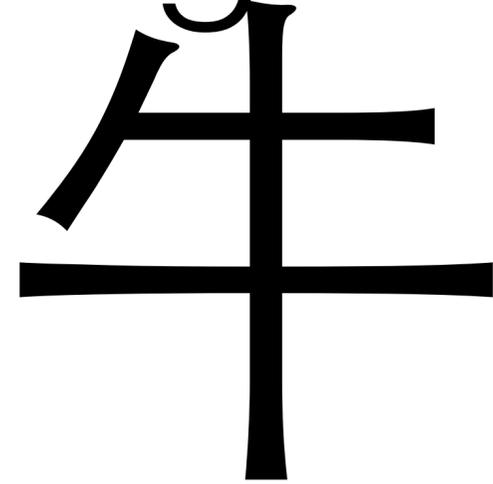
Hardware fork latency: 80ms

VMware fork latency: 77ms

Capacity Planning



memory



Note

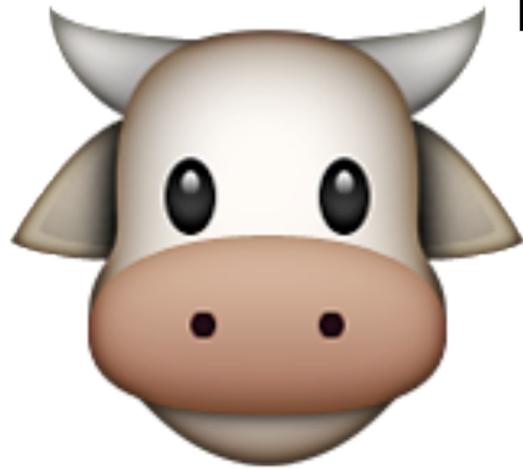
Xen has horrible forking performance.

Hardware fork latency: 80ms

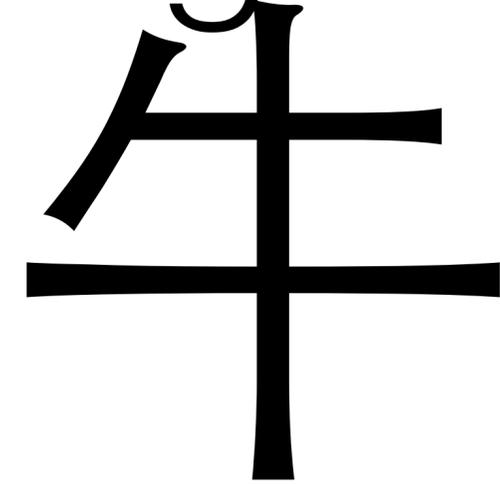
VMware fork latency: 77ms

Xen fork latency: 1460ms

Capacity Planning

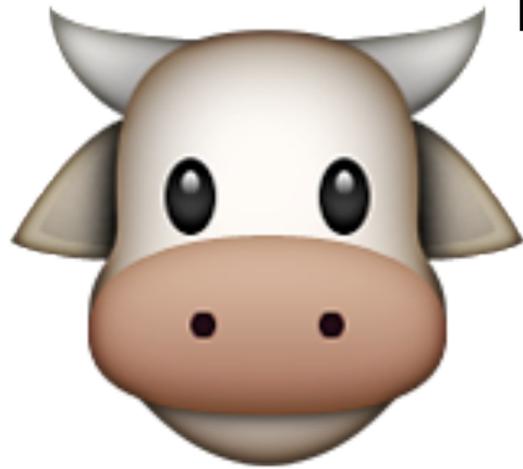


memory

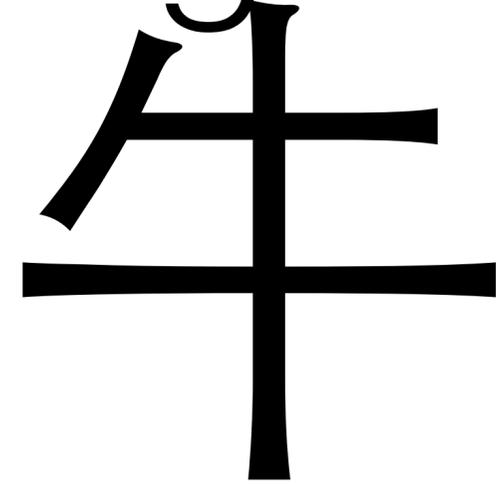


Note

Capacity Planning



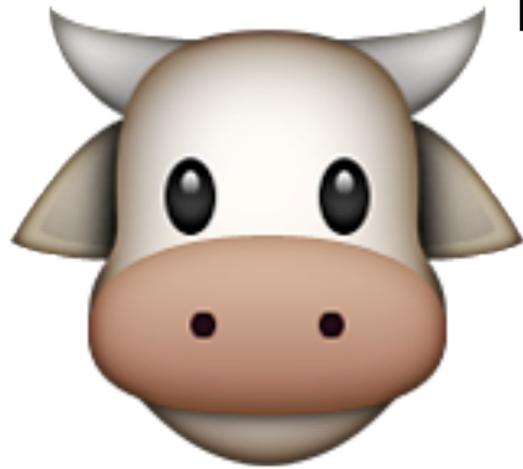
memory



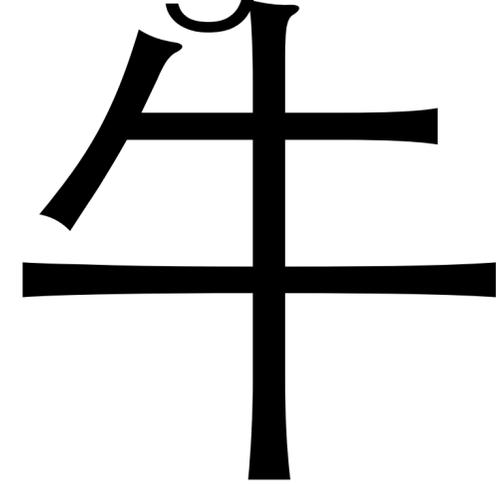
Note

A process is blocked until `fork` returns.

Capacity Planning



memory

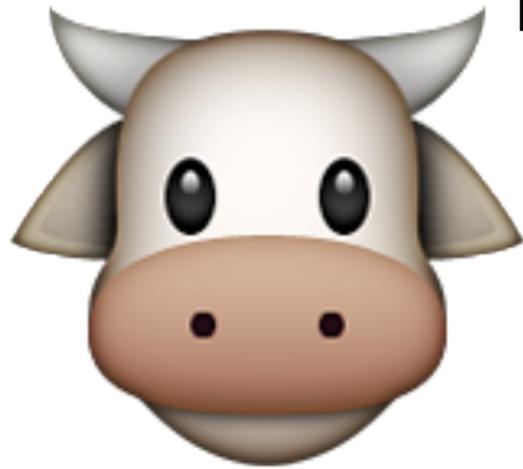


Note

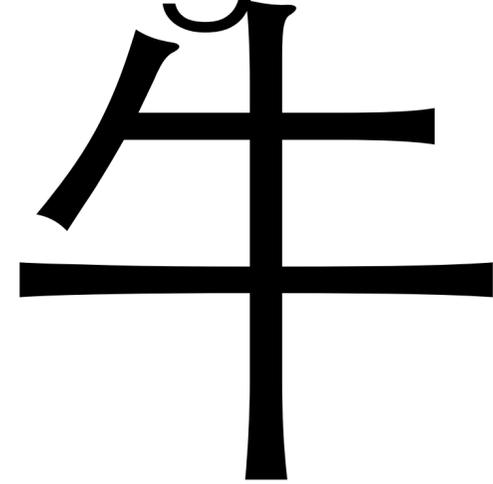
A process is blocked until `fork` returns.

Normally a tiny less-than-100ms hiccup.

Capacity Planning



memory



Note

A process is blocked until `fork` returns.

Normally a tiny less-than-100ms hiccup.

On Xen, you can notice a multiple second “outage.”

Capacity Planning

Capacity Planning

network

Capacity Planning

network

Recommendations

Capacity Planning

network

Recommendations

Have a network.

Capacity Planning

network

Recommendations

Have a network.

Know where you are in your network.

Capacity Planning

network

Recommendations

Have a network.

Know where you are in your network.

Try to at least be in the same building as your DBs.

Capacity Planning

network

Protocol

Capacity Planning

network

Protocol

* [argument count] \r \n

Capacity Planning

network

Protocol

```
*[argument count]\r\n
```

```
#[byte count of argument]\r\n
```

Capacity Planning

network

Protocol

```
*[argument count]\r\n
```

```
#[byte count of argument]\r\n
```

```
[data]\r\n
```

Capacity Planning

network

Protocol

```
*[argument count]\r\n
```

```
#[byte count of argument]\r\n
```

```
[data]\r\n
```

.

.

```
#[byte count of last argument]\r\n
```

```
[data]\r\n
```

Capacity Planning

network

Protocol

Capacity Planning

network

Protocol

```
SET name Matt
```

Capacity Planning

network

Protocol

*3

```
SET name Matt
```

Capacity Planning

network

Protocol

* 3

\$ 3

SET name Matt

Capacity Planning

network

Protocol

* 3

\$ 3

SET name Matt

SET

Capacity Planning

network

Protocol

* 3

\$ 3

SET name Matt

SET

\$ 4

Capacity Planning

network

Protocol

* 3

\$ 3

SET name Matt

SET

\$ 4

name

Capacity Planning

network

Protocol

* 3

\$ 3

SET name Matt

SET

\$ 4

name

\$ 4

Capacity Planning

network

Protocol

* 3

\$ 3

SET name Matt

SET

\$ 4

name

\$ 4

Matt

Capacity Planning

network

Protocol

* 3

\$ 3

SET name Matt

SET

\$ 4

name

\$ 4

Matt

Capacity Planning

network

Protocol

```
SET name Matt
```

```
*3\r\n$3\r\nSET\r\n$4\r\nname\r\n$4\r\nMatt\r\n
```

Capacity Planning

network

Protocol

```
SET name Matt
```

```
*3\r\n$3\r\nSET\r\n$4\r\nname\r\n$4\r\nMatt\r\n
```

33 bytes

Capacity Planning

network

Protocol

Capacity Planning

network

Protocol

SADD names Matt

Capacity Planning

network

Protocol

SADD names Matt

SADD names GoPivotal

Capacity Planning

network

Protocol

SADD names Matt

SADD names GoPivotal

SADD names Ireland

Capacity Planning

network

Protocol

SADD names Matt

SADD names GoPivotal

SADD names Ireland

SADD names Barcelona

Capacity Planning

SADD names Matt
SADD names GoPivotal
SADD names Ireland
SADD names Barcelona

network

Protocol

Capacity Planning

```
SADD names Matt  
SADD names GoPivotal  
SADD names Ireland  
SADD names Barcelona
```

network

Protocol

```
*3\r\n$4\r\nSADD\r\n$5\r\nnames\r\n$4\r\nMatt\r\n
```

Capacity Planning

```
SADD names Matt  
SADD names GoPivotal  
SADD names Ireland  
SADD names Barcelona
```

network

Protocol

```
* 3 \r\n$4 \r\nSADD\r\n$5 \r\nnames \r\n$4 \r\nMatt\r\n
```

```
* 3 \r\n$4 \r\nSADD\r\n$5 \r\nnames \r\n$9 \r\nGoPivotal\r\n
```

Capacity Planning

```
SADD names Matt  
SADD names GoPivotal  
SADD names Ireland  
SADD names Barcelona
```

network

Protocol

```
* 3 \r\n$4 \r\nSADD\r\n$5 \r\nnames \r\n$4 \r\nMatt\r\n
```

```
* 3 \r\n$4 \r\nSADD\r\n$5 \r\nnames \r\n$9 \r\nGoPivotal\r\n
```

```
* 3 \r\n$4 \r\nSADD\r\n$5 \r\nnames \r\n$7 \r\nIreland\r\n
```

Capacity Planning

```
SADD names Matt
SADD names GoPivotal
SADD names Ireland
SADD names Barcelona
```

network

Protocol

```
*3\r\n$4\r\nSADD\r\n$5\r\nnames\r\n$4\r\nMatt\r\n
```

```
*3\r\n$4\r\nSADD\r\n$5\r\nnames\r\n$9\r\nGoPivotal\r\n
```

```
*3\r\n$4\r\nSADD\r\n$5\r\nnames\r\n$7\r\nIreland\r\n
```

```
*3\r\n$4\r\nSADD\r\n$5\r\nnames\r\n$9\r\nBarcelona\r\n
```

Capacity Planning

```
SADD names Matt
SADD names GoPivotal
SADD names Ireland
SADD names Barcelona
```

network

Protocol

```
*3\r\n$4\r\nSADD\r\n$5\r\nnames\r\n$4\r\nMatt\r\n
```

34
bytes

```
*3\r\n$4\r\nSADD\r\n$5\r\nnames\r\n$9\r\nGoPivotal\r\n
```

39
bytes

```
*3\r\n$4\r\nSADD\r\n$5\r\nnames\r\n$7\r\nIreland\r\n
```

37
bytes

```
*3\r\n$4\r\nSADD\r\n$5\r\nnames\r\n$9\r\nBarcelona\r\n
```

39
bytes

Capacity Planning

```
SADD names Matt  
SADD names GoPivotal  
SADD names Ireland  
SADD names Barcelona
```

network

Protocol

```
* 3 \r\n$4\r\nSADD\r\n$5\r\nnames\r\n$4\r\nMatt\r\n\r\n34 bytes  
* 3 \r\n$4\r\nSADD\r\n$5\r\nnames\r\n$9\r\nGoPivotal\r\n\r\n39 bytes  
* 3 \r\n$4\r\nSADD\r\n$5\r\nnames\r\n$7\r\nIreland\r\n\r\n37 bytes  
* 3 \r\n$4\r\nSADD\r\n$5\r\nnames\r\n$9\r\nBarcelona\r\n\r\n39 bytes
```

149 bytes total

Capacity Planning

```
SADD names Matt
SADD names GoPivotal
SADD names Ireland
SADD names Barcelona
```

network

Protocol

```
*3\r\n$4\r\nSADD\r\n$5\r\nnames\r\n$4\r\nMatt\r\n
*3\r\n$4\r\nSADD\r\n$5\r\nnames\r\n$9\r\nGoPivotal\r\n
*3\r\n$4\r\nSADD\r\n$5\r\nnames\r\n$7\r\nIreland\r\n
*3\r\n$4\r\nSADD\r\n$5\r\nnames\r\n$9\r\nBarcelona\r\n
```

34
bytes

39
bytes

37
bytes

39
bytes

149 bytes total

Capacity Planning

SADD names Matt
SADD names GoPivotal
SADD names Ireland
SADD names Barcelona

network

Protocol

SADD names Matt GoPivotal Ireland Barcelona

Capacity Planning

```
SADD names Matt  
SADD names GoPivotal  
SADD names Ireland  
SADD names Barcelona
```

network

Protocol

```
SADD names Matt GoPivotal Ireland Barcelona
```

```
*6\r\n$4\r\nSADD\r\n$5\r\nnames\r\n$4\r\nMatt\r\n$9\r\n\r\nGoPivotal\r\n$7\r\nIreland\r\n$9\r\nBarcelona\r\n\r\n
```

Capacity Planning

```
SADD names Matt  
SADD names GoPivotal  
SADD names Ireland  
SADD names Barcelona
```

network

Protocol

```
SADD names Matt GoPivotal Ireland Barcelona
```

```
*6\r\n$4\r\nSADD\r\n$5\r\nnames\r\n$4\r\nMatt\r\n$9\r\n\r\nGoPivotal\r\n$7\r\nIreland\r\n$9\r\nBarcelona\r\n\r\n
```

78 bytes total

Capacity Planning

network

Protocol

Capacity Planning

network

Protocol

```
SADD names Matt  
SADD names GoPivotal  
SADD names Ireland  
SADD names Barcelona
```

Capacity Planning

network

Protocol

SADD names Matt
SADD names GoPivotal
SADD names Ireland
SADD names Barcelona

VS.

Capacity Planning

network

Protocol

SADD names Matt
SADD names GoPivotal
SADD names Ireland
SADD names Barcelona

VS.

SADD names Matt GoPivotal Ireland Barcelona

Capacity Planning

network

Protocol

SADD names Matt
SADD names GoPivotal
SADD names Ireland
SADD names Barcelona

VS.

SADD names Matt GoPivotal Ireland Barcelona

149 bytes total vs. 78 bytes total

Capacity Planning

network

Protocol

SADD names Matt
SADD names GoPivotal
SADD names Ireland
SADD names Barcelona

VS.

SADD names Matt GoPivotal Ireland Barcelona

149 bytes total

vs.

78 bytes total

55% less network traffic

Capacity Planning

Pipelining

network

Protocol

Capacity Planning

Pipelining

network

Protocol

→ SISMEMBER names Matt

Capacity Planning

Pipelining

network

Protocol

→ SISMEMBER names Matt

→
1

Capacity Planning

Pipelining

network

Protocol

→ SISMEMBER names Matt

→
1

→ SISMEMBER names GoPivotal

Capacity Planning

Pipelining

network

Protocol

➔ SISMEMBER names Matt

➔
1

➔ SISMEMBER names GoPivotal

➔
1

Capacity Planning

Pipelining

network

Protocol

→ SISMEMBER names Matt

→
1

→ SISMEMBER names GoPivotal

→
1

→ SISMEMBER names Croatia

Capacity Planning

Pipelining

network

Protocol

→ SISMEMBER names Matt

→
1

→ SISMEMBER names GoPivotal

→
1

→ SISMEMBER names Croatia

→
0

Capacity Planning

Pipelining

network

Protocol

→ SISMEMBER names Matt

→
1

→ SISMEMBER names GoPivotal

→
1

→ SISMEMBER names Croatia

→
0

→ SISMEMBER names Barcelona

Capacity Planning

Pipelining

network

Protocol

→ SISMEMBER names Matt

→
1

→ SISMEMBER names GoPivotal

→
1

→ SISMEMBER names Croatia

→
0

→ SISMEMBER names Barcelona

→
1

Capacity Planning

Pipelining

network

Protocol

→ SISMEMBER names Matt

→
1

→ SISMEMBER names GoPivotal

→
1

→ SISMEMBER names Croatia

→
0

→ SISMEMBER names Barcelona

→
1

8 network
round trips

Capacity Planning

Pipelining

network

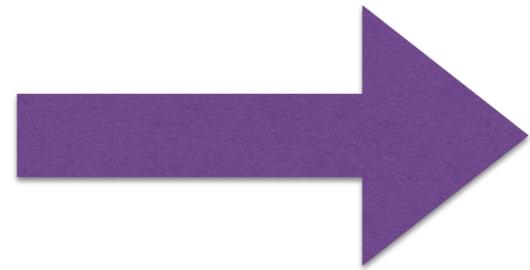
Protocol

Capacity Planning

Pipelining

network

Protocol



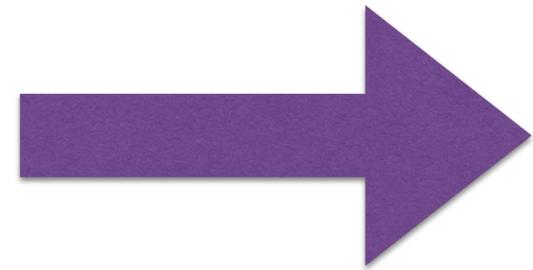
```
SISMEMBER names Matt  
SISMEMBER names GoPivotal  
SISMEMBER names Croatia  
SISMEMBER names Barcelona
```

Capacity Planning

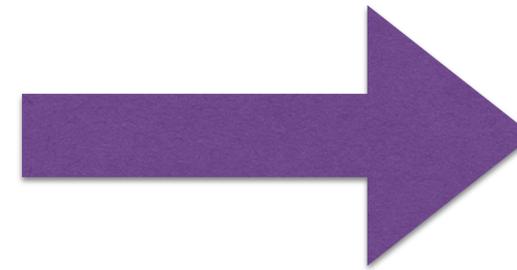
Pipelining

network

Protocol



```
SISMEMBER names Matt  
SISMEMBER names GoPivotal  
SISMEMBER names Croatia  
SISMEMBER names Barcelona
```



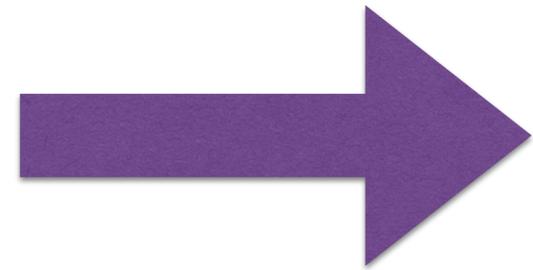
```
1  
1  
0  
1
```

Capacity Planning

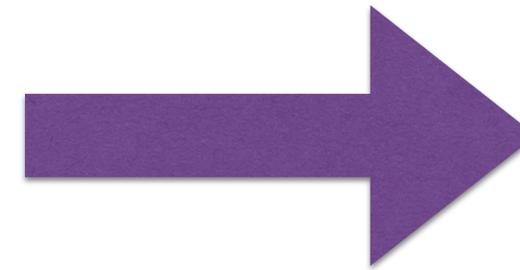
Pipelining

network

Protocol



```
SISMEMBER names Matt  
SISMEMBER names GoPivotal  
SISMEMBER names Croatia  
SISMEMBER names Barcelona
```



```
1  
1  
0  
1
```

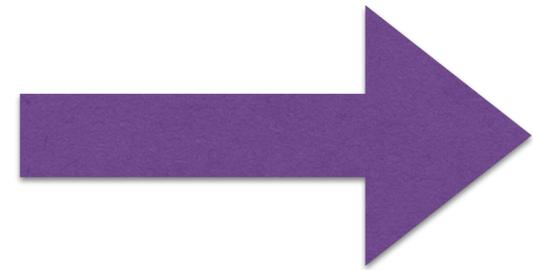
results returned
in execution
order

Capacity Planning

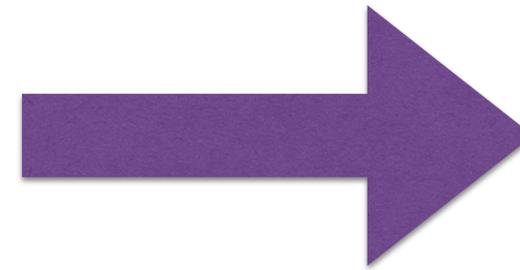
Pipelining

network

Protocol



```
SISMEMBER names Matt  
SISMEMBER names GoPivotal  
SISMEMBER names Croatia  
SISMEMBER names Barcelona
```



1
1
0
1

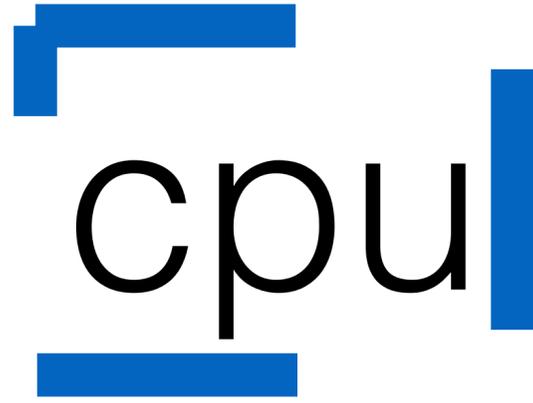
results returned
in execution
order

2 network round trips

Capacity Planning

cpu

Capacity Planning



Redis uses one thread for data manipulation.

Capacity Planning

cpu

Redis uses one thread for data manipulation.

Redis uses one thread for background **AOF** writes.

Capacity Planning

cpu

Redis uses one thread for data manipulation.

Redis uses one thread for background **AOF** writes.

Redis forks for **BGSAVE**.

Capacity Planning

cpu

Capacity Planning

cpu

Run one redis-server instance per core.

Capacity Planning

cpu

Run one redis-server instance per core.

Leave a few cores free for `AOF` and `BGSAVE` scheduling.

Capacity Planning

cpu

Run one redis-server instance per core.

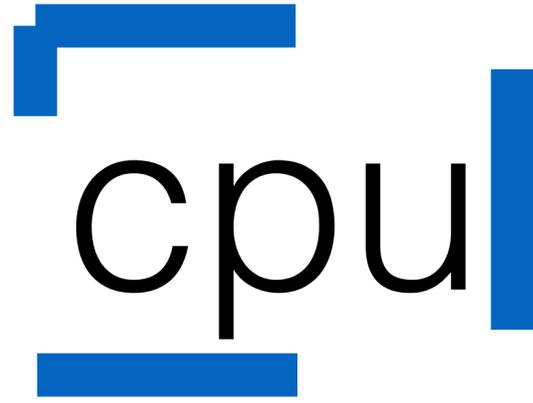
Leave a few cores free for `AOF` and `BGSAVE` scheduling.

Cluster will make running multiple instances per host simpler.

Capacity Planning

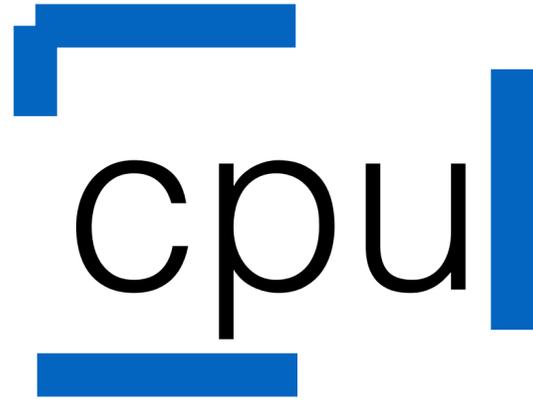
cpu

Capacity Planning



Each Redis command has individual performance characteristics.

Capacity Planning



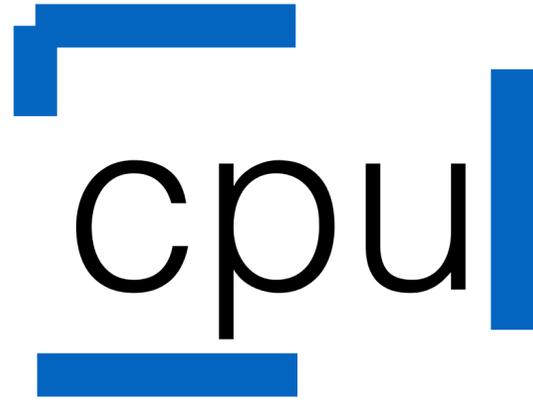
Each Redis command has individual performance characteristics.

The Big-Oh of each command is listed in the documentation.

Capacity Planning

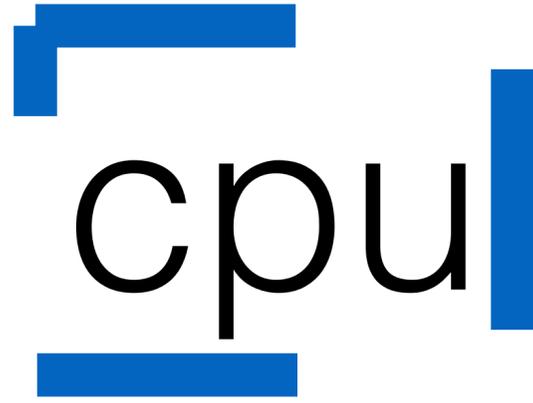
cpu

Capacity Planning



Redis provides excellent performance and latency for:

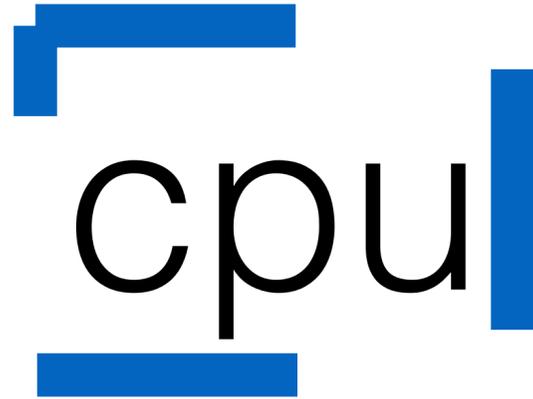
Capacity Planning



Redis provides excellent performance and latency for:

- 1) $O(1)$ operations.

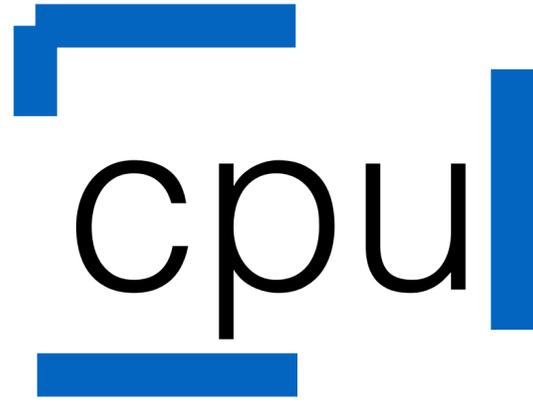
Capacity Planning



Redis provides excellent performance and latency for:

- 1) $O(1)$ operations.
- 2) Logarithmic operations (most basic sorted sets operations, including ZRANK).

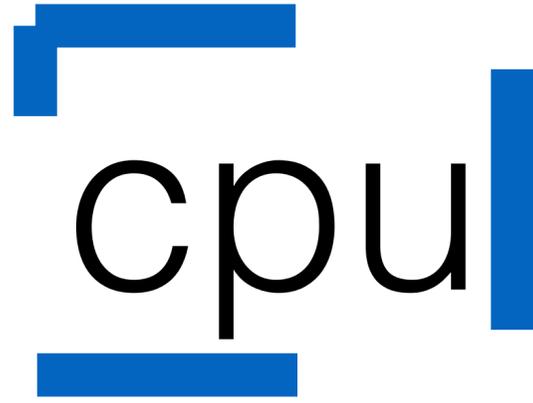
Capacity Planning



Redis provides excellent performance and latency for:

- 1) $O(1)$ operations.
- 2) Logarithmic operations (most basic sorted sets operations, including ZRANK).
- 3) $O(N)$ seek + $O(M)$ work (for example LTRIM) every time you can make sure to take M small. Example: capped collections implementation.

Capacity Planning



Redis provides excellent performance and latency for:

- 1) $O(1)$ operations.
- 2) Logarithmic operations (most basic sorted sets operations, including ZRANK).
- 3) $O(N)$ seek + $O(M)$ work (for example LTRIM) every time you can make sure to take M small. Example: capped collections implementation.
- 4) $\log(N)$ seek + $O(M)$ work (for example removing ranges of elements from a sorted set).

Capacity Planning



Capacity Planning



Disks are useful for saving data and storing configs.

Capacity Planning



Disks are useful for saving data and storing configs.

`AOF` and `BGSAVE` and `redis.conf` and
`sentinel.conf` and `cluster.conf`

Capacity Planning



Capacity Planning



Redis needs to own a directory with write access.

Capacity Planning



Redis needs to own a directory with write access.

The `dir` config parameter.

Capacity Planning



Redis needs to own a directory with write access.

The `dir` config parameter.

Redis saves `AOF/BGSAVE`/cluster state to the `dir`.

How Redis fail?

Failure Scenarios

Failure Scenarios

Failure Scenarios

memory corruption

Failure Scenarios

memory corruption

configuration not matching expectations

Failure Scenarios

memory corruption

configuration not matching expectations

coding errors

Failure Scenarios

memory corruption

configuration not matching expectations

coding errors

users abusing the DB

Failure Scenarios

the error report

Failure Scenarios

the error report

```
=== REDIS BUG REPORT START: Cut & paste starting from here ===  
[2285] 04 Nov 15:19:02.148 # Redis 2.7.104 crashed by signal: 11  
[2285] 04 Nov 15:19:02.148 # Failed assertion: (:0)  
[2285] 04 Nov 15:19:02.148 # --- STACK TRACE  
redis-rdb-bgsave *:6379(logStackTrace+0x3e)[0x443c1e]  
redis-rdb-bgsave *:6379(rdbSaveStringObject+0x0)[0x42c2d0]
```

Failure Scenarios

the error report

```
=== REDIS BUG REPORT END. Make sure to include from START to END. ===
```

Failure Scenarios

the error report

Failure Scenarios

the error report

sections:

Failure Scenarios

the error report

sections:

server

clients

memory

persistence

stats

replication

CPU

command stats

cluster

keyspace

list of clients

list of client state

register contents

Failure Scenarios

memory corruption

Failure Scenarios

memory corruption

cause:

Failure Scenarios

memory corruption

cause:

not using ECC memory

Failure Scenarios

memory corruption

cause:

not using ECC memory

these days nobody knows
what hardware they use.

Failure Scenarios

memory corruption

Failure Scenarios

memory corruption

symptoms:

Failure Scenarios

memory corruption

symptoms:
unexplained segfaults

Failure Scenarios

memory corruption

symptoms:
unexplained segfaults
gibberish in error reports

Failure Scenarios

memory corruption

symptoms:

unexplained segfaults
gibberish in error reports

usually at the same time

Failure Scenarios

memory corruption

Failure Scenarios

memory corruption

```
used_memory:18374686491318515000  
used_memory_human:pk!D  
used_memory_rss:14293835776  
used_memory_peak:11877760896  
used_memory_peak_human:11.06G  
used_memory_lua:47104  
mem_fragmentation_ratio:0.00  
mem_allocator:libc
```

Failure Scenarios

what's that? memory corruption

```
used_memory:18374686491318515000  
used_memory_human:pk!D  
used_memory_rss:14293835776  
used_memory_peak:11877760896  
used_memory_peak_human:11.06G  
used_memory_lua:47104  
mem_fragmentation_ratio:0.00  
mem_allocator:libc
```

Failure Scenarios

what's that? memory corruption

```
used_memory:183746864913  
used_memory_human:pk!D  
used_memory_rss:14293835776  
used_memory_peak:11877760896  
used_memory_peak_human:11.06G  
used_memory_lua:47104  
mem_fragmentation_ratio:0.00  
mem_allocator:libc
```

what's that?

Failure Scenarios

what's that?

not even
regular p

```
used_memory:183746864913  
used_memory_human:pk!D  
used_memory_rss:14293835776  
used_memory_peak:11877760896  
used_memory_peak_human:11.06G  
used_memory_lua:47104  
mem_fragmentation_ratio:0.00  
mem_allocator:libc
```

what's that?

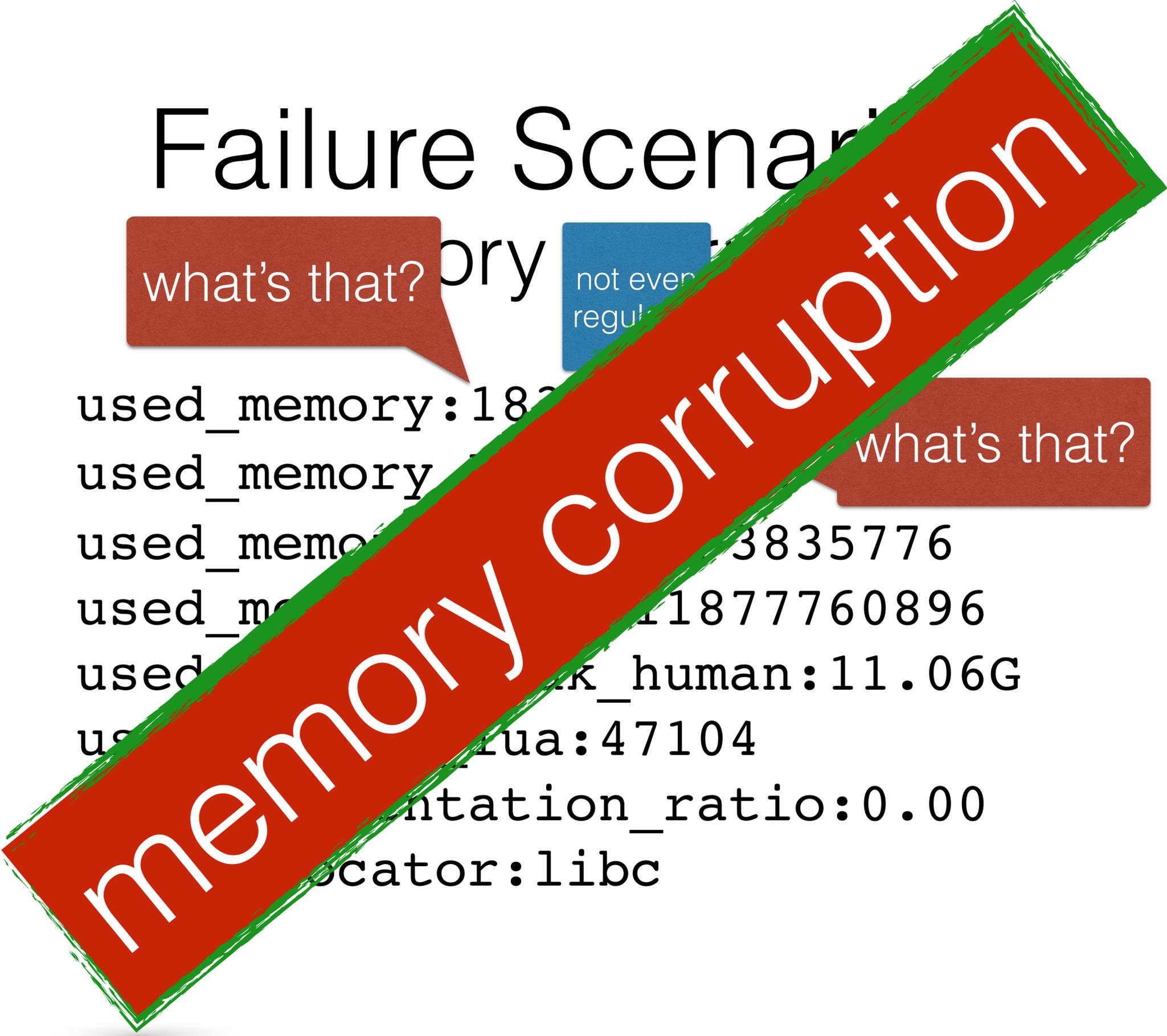
Failure Scenario

what's that?

not even
regul

```
used_memory:182  
used_memory:  
used_memory:3835776  
used_memory:11877760896  
used_memory_human:11.06G  
used_memory_lua:47104  
memory_reservation_ratio:0.00  
indicator:libc
```

what's that?



Failure Scenarios

memory corruption

Failure Scenarios

memory corruption

actions:

Failure Scenarios

memory corruption

actions:

If Redis crashes and memory is suspected,

Failure Scenarios

memory corruption

actions:

If Redis crashes and memory is suspected,
without rebooting after the crash,

Failure Scenarios

memory corruption

actions:

If Redis crashes and memory is suspected,
without rebooting after the crash,

```
redis-server -test-memory [MB]
```

\

memory corruption

example:

```
redis-server -test-memory 16384
```

Failure Scenarios

memory corruption

Failure Scenarios

memory corruption

check `dmesg` for memory errors

Failure Scenarios

memory corruption

check `dmesg` for memory errors

if available, check `ipmi` output:

Failure Scenarios

memory corruption

check `dmesg` for memory errors

if available, check `ipmi` output:

```
ipmitool sel list
```

Failure Scenarios

memory corruption

check `dmesg` for memory errors

if available, check `ipmi` output:

```
ipmitool sel list
```

```
Memory #0x01 | Correctable ECC  
Memory #0x02 | Uncorrectable ECC
```

Failure Scenarios

memory corruption

Failure Scenarios

memory corruption

better test:

Failure Scenarios

memory corruption

better test:

if user has hardware access, run MemTest86

Failure Scenarios

memory corruption

better test:

if user has hardware access, run MemTest86

pre-boot memory testing

Failure Scenarios

memory corruption

better test:

if user has hardware access, run MemTest86

pre-boot memory testing

<http://www.memtest86.com/>

Failure Scenarios

memory corruption

Failure Scenarios

memory corruption

Most reported Redis crashes are due to hardware memory corruption.

Failure Scenarios

memory corruption

Most reported Redis crashes are due to hardware memory corruption.

Always recommend running with ECC memory.

Failure Scenarios

persistence problems

Failure Scenarios

persistence problems

BGSAVE degrading performance

Failure Scenarios

persistence problems

BGSAVE degrading performance

AOE corruption

Failure Scenarios

persistence problems

Failure Scenarios

persistence problems

BGSAVE = binary, compressed, DB snapshot

Failure Scenarios

persistence problems

BGSAVE = binary, compressed, DB snapshot

AOF = command-by-command DB reconstruction

Failure Scenarios

persistence problems

Failure Scenarios

persistence problems

BGSAVE = fork () latency varies by platform.

Failure Scenarios

persistence problems

BGSAVE = `fork()` latency varies by platform.
hardware = 10ms per GB.

Failure Scenarios

persistence problems

BGSAVE = `fork()` latency varies by platform.
hardware = 10ms per GB.
xen = 250ms per GB.

Failure Scenarios

persistence problems

BGSAVE = `fork()` latency varies by platform.

hardware = 10ms per GB.

xen = 250ms per GB.

AOE = flushes to disk every second

Failure Scenarios

persistence problems

BGSAVE = `fork()` latency varies by platform.

hardware = 10ms per GB.

xen = 250ms per GB.

AOE = flushes to disk every second
(or always or never)

Failure Scenarios

persistence problems

Failure Scenarios

persistence problems

fork latency reported as:

Failure Scenarios

persistence problems

fork latency reported as:

`latest_fork_usec` in `INFO` output

Failure Scenarios

persistence problems

Failure Scenarios

persistence problems

if AOF gets corrupted, you can edit it

Failure Scenarios

persistence problems

if AOF gets corrupted, you can edit it

remove the end-of-file corruption

Failure Scenarios

persistence problems

if AOF gets corrupted, you can edit it

remove the end-of-file corruption
(lose or recover the last key)

Failure Scenarios

persistence problems

Failure Scenarios

persistence problems

BGSAVE = faster restarts

Failure Scenarios

persistence problems

BGSAVE = faster restarts
(writes DB to disk based on usage)

Failure Scenarios

persistence problems

BGSAVE = faster restarts
(writes DB to disk based on usage)

AOF = more up-to-date persistence

Failure Scenarios

persistence problems

BGSAVE = faster restarts
(writes DB to disk based on usage)

AOFL = more up-to-date persistence
(flushes to disk once a second)

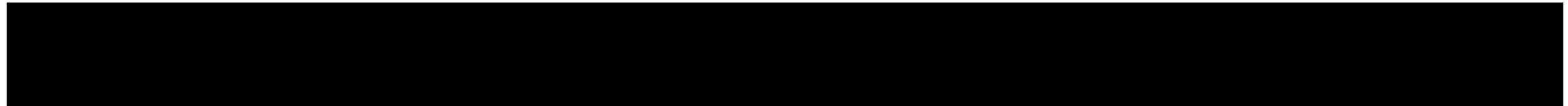
Failure Scenarios

configuration delusions

Failure Scenarios

configuration delusions

Redis is telling me I can't write.



Failure Scenarios

configuration delusions

Redis is telling me I can't write.

Did `BGSAVE` fail while you're running with
`stop-writes-on-bgsave-error yes`

Failure Scenarios

configuration delusions

Failure Scenarios

configuration delusions

Redis is telling me I can't write.



Failure Scenarios

configuration delusions

Redis is telling me I can't write.

Are you running with disconnected replicas and these configs set

```
min-slaves-max-lag 10
```

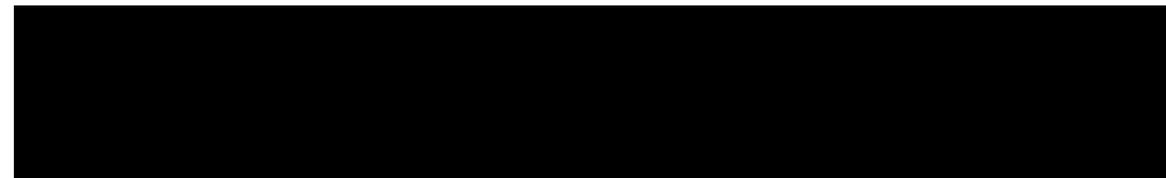
Failure Scenarios

configuration delusions

Failure Scenarios

configuration delusions

Redis is telling me I can't write.



Failure Scenarios

configuration delusions

Redis is telling me I can't write.

Did you overrun your memory limit?

maxmemory 16GB

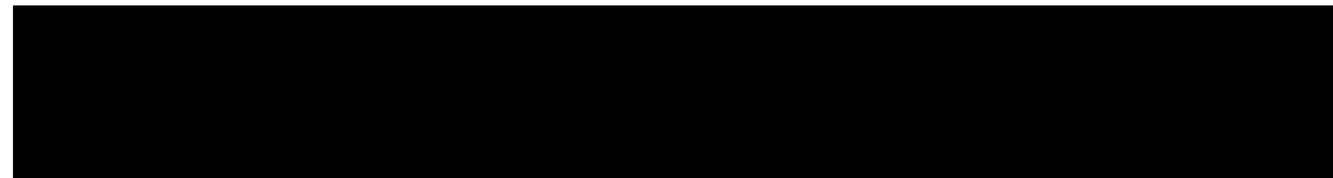
Failure Scenarios

configuration delusions

Failure Scenarios

configuration delusions

Redis is telling me I can't connect.



Failure Scenarios

configuration delusions

Redis is telling me I can't connect.

Did you overrun your client limit?

```
maxclients 10000
```

Failure Scenarios

configuration delusions

Failure Scenarios

configuration delusions

Redis is telling me I can't read.



Failure Scenarios

configuration delusions

Redis is telling me I can't read.

Are you reading a disconnected replica with

slave-serve-stale-data no

Failure Scenarios

users abusing the DB

Failure Scenarios

users abusing the DB

Redis is intermittently stalled.

Failure Scenarios

users abusing the DB

Redis is intermittently stalled.

```
CONFIG SET slowlog-log-slower-than [μs]
```

Failure Scenarios

users abusing the DB

Redis is intermittently stalled.

```
CONFIG SET slowlog-log-slower-than [μs]  
CONFIG SET slowlog-max-len [entries]
```

Failure Scenarios

users abusing the DB

Redis is intermittently stalled.

```
CONFIG SET slowlog-log-slower-than [μs]
```

```
CONFIG SET slowlog-max-len [entries]
```

```
SLOWLOG GET 10
```

Failure Scenarios

users abusing the DB

Redis is intermittently stalled.

```
CONFIG SET slowlog-log-slower-than [μs]
```

```
CONFIG SET slowlog-max-len [entries]
```

```
SLOWLOG GET 10
```

```
SLOWLOG RESET
```

Failure Scenarios

users abusing the DB

Failure Scenarios

users abusing the DB

Redis is stalled.

Failure Scenarios

users abusing the DB

Redis is stalled.

Did someone run `KEYS *` on a large DB?

Failure Scenarios

users abusing the DB

Redis is stalled.

Did someone run `KEYS *` on a large DB?

(`KEYS` is deprecated in favor of the Redis 2.8 `SCAN` interface)

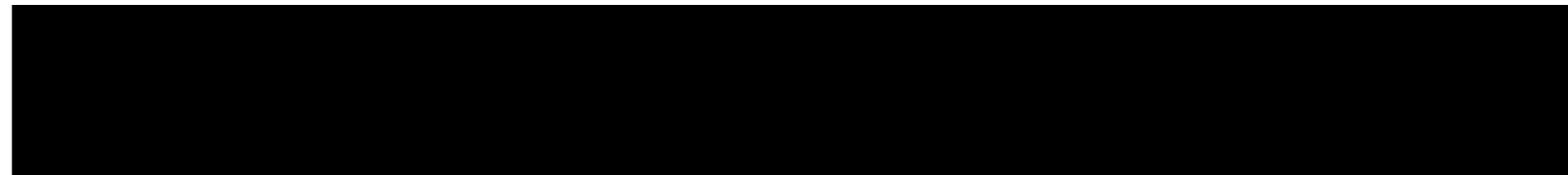
Failure Scenarios

users abusing the DB

Failure Scenarios

users abusing the DB

Redis is stalled.



Failure Scenarios

users abusing the DB

Redis is stalled.

Is a poorly behaved script running?

```
lua-time-limit 5000
```

Failure Scenarios

users abusing the DB

Failure Scenarios

users abusing the DB

`lua-time-limit` does **not** kill scripts.

Failure Scenarios

users abusing the DB

`lua-time-limit` does **not** kill scripts.

After `lua-time-limit` is reached,
the server accepts commands again, but only
allows `SCRIPT KILL` or `SHUTDOWN NOSAVE`

Failure Scenarios

users abusing the DB

Failure Scenarios

users abusing the DB

Watch out for latency complaints when Redis is paging out to swap space.

Failure Scenarios

users abusing the DB

Failure Scenarios

users abusing the DB

For serious, production-level DB usage:

Failure Scenarios

users abusing the DB

For serious, production-level DB usage:

Run dedicated DB hardware with swap off.

Failure Scenarios

users abusing the DB

Failure Scenarios

users abusing the DB

Check how much of Redis is paged to disk:

Failure Scenarios

users abusing the DB

Check how much of Redis is paged to disk:

```
cat /proc/<pid of redis-server>/smaps
```

Failure Scenarios

users abusing the DB

Failure Scenarios

users abusing the DB

Servers should not swap.

Failure Scenarios

users abusing the DB

Servers should not swap.

Servers should not have swap enabled.

Failure Scenarios

users abusing the DB

Servers should not swap.

Servers should not have swap enabled.

If your server is mis-sized,

Failure Scenarios

users abusing the DB

Servers should not swap.

Servers should not have swap enabled.

If your server is mis-sized,
you deserve the OOM Killer.

How Redis debug?

Automated data collection

Debug Info

Debug Info

If you failed, you have an error report.

Debug Info

Debug Info

```
=== REDIS BUG REPORT END. Make sure to include from START to END. ===
```

Debug Info

If you aren't failed, grab **INFO**

Debug Info

Debug Info

```
db0:keys=12,expires=0,avg_ttl=0
```

Debug Info

Quick Overview of `INFO` Fields

Debug Info

Sections

- `server`: General information about the Redis server
- `clients`: Client connections section
- `memory`: Memory consumption related information
- `persistence`: RDB and AOF related information
- `stats`: General statistics
- `replication`: Master/slave replication information
- `cpu`: CPU consumption statistics
- `commandstats`: Redis command statistics
- `cluster`: Redis Cluster section
- `keyspace`: Database related statistics

Debug Info

Server

- `redis_version`: Version of the Redis server
- `redis_git_sha1`: Git SHA1
- `redis_git_dirty`: Git dirty flag
- `os`: Operating system hosting the Redis server
- `arch_bits`: Architecture (32 or 64 bits)
- `multiplexing_api`: event loop mechanism used by Redis
- `gcc_version`: Version of the GCC compiler used to compile the Redis server
- `process_id`: PID of the server process
- `run_id`: Random value identifying the Redis server (to be used by Sentinel and Cluster)
- `tcp_port`: TCP/IP listen port
- `uptime_in_seconds`: Number of seconds since Redis server start
- `uptime_in_days`: Same value expressed in days
- `lru_clock`: Clock incrementing every minute, for LRU management

Debug Info

Clients

- `connected_clients`: Number of client connections (excluding connections from slaves)
- `client_longest_output_list`: longest output list among current client connections
- `client_biggest_input_buf`: biggest input buffer among current client connections
- `blocked_clients`: Number of clients pending on a blocking call (BLPOP, BRPOP, BRPOPLPUSH)

Debug Info

Memory

- `used_memory`: total number of bytes allocated by Redis using its allocator (either standard **libc**, **jemalloc**, or an alternative allocator such as **tcmalloc**)
- `used_memory_human`: Human readable representation of previous value
- `used_memory_rss`: Number of bytes that Redis allocated as seen by the operating system (a.k.a resident set size). This is the number reported by tools such as **top** and **ps**.
- `used_memory_peak`: Peak memory consumed by Redis (in bytes)
- `used_memory_peak_human`: Human readable representation of previous value
- `used_memory_lua`: Number of bytes used by the Lua engine
- `mem_fragmentation_ratio`: Ratio between `used_memory_rss` and `used_memory`
- `mem_allocator`: Memory allocator, chosen at compile time.

Debug Info

Persistence I (General)

- `loading`: Flag indicating if the load of a dump file is on-going
- `rdb_changes_since_last_save`: Number of changes since the last dump
- `rdb_bgsave_in_progress`: Flag indicating a RDB save is on-going
- `rdb_last_save_time`: Epoch-based timestamp of last successful RDB save
- `rdb_last_bgsave_status`: Status of the last RDB save operation
- `rdb_last_bgsave_time_sec`: Duration of the last RDB save operation in seconds
- `rdb_current_bgsave_time_sec`: Duration of the on-going RDB save operation if any
- `aof_enabled`: Flag indicating AOF logging is activated
- `aof_rewrite_in_progress`: Flag indicating a AOF rewrite operation is on-going
- `aof_rewrite_scheduled`: Flag indicating an AOF rewrite operation will be scheduled once the on-going RDB save is complete.
- `aof_last_rewrite_time_sec`: Duration of the last AOF rewrite operation in seconds
- `aof_current_rewrite_time_sec`: Duration of the on-going AOF rewrite operation if any
- `aof_last_bgrewrite_status`: Status of the last AOF rewrite operation

Debug Info

Persistence II (AOF)

- `aof_current_size`: AOF current file size
- `aof_base_size`: AOF file size on latest startup or rewrite
- `aof_pending_rewrite`: Flag indicating an AOF rewrite operation will be scheduled once the on-going RDB save is complete.
- `aof_buffer_length`: Size of the AOF buffer
- `aof_rewrite_buffer_length`: Size of the AOF rewrite buffer
- `aof_pending_bio_fsync`: Number of fsync pending jobs in background I/O queue
- `aof_delayed_fsync`: Delayed fsync counter

Debug Info

Persistence III (Loading)

- `loading_start_time`: Epoch-based timestamp of the start of the load operation
- `loading_total_bytes`: Total file size
- `loading_loaded_bytes`: Number of bytes already loaded
- `loading_loaded_perc`: Same value expressed as a percentage
- `loading_eta_seconds`: ETA in seconds for the load to be complete

Debug Info

Stats

- `total_connections_received`: Total number of connections accepted by the server
- `total_commands_processed`: Total number of commands processed by the server
- `instantaneous_ops_per_sec`: Number of commands processed per second
- `rejected_connections`: Number of connections rejected because of maxclients limit
- `expired_keys`: Total number of key expiration events
- `evicted_keys`: Number of evicted keys due to maxmemory limit
- `keyspace_hits`: Number of successful lookup of keys in the main dictionary
- `keyspace_misses`: Number of failed lookup of keys in the main dictionary
- `pubsub_channels`: Global number of pub/sub channels with client subscriptions
- `pubsub_patterns`: Global number of pub/sub pattern with client subscriptions
- `latest_fork_usec`: Duration of the latest fork operation in microseconds

Debug Info

Replication

- `role`: Value is "master" if the instance is slave of no one, or "slave" if the instance is enslaved to a master. Note that a slave can be master of another slave (daisy chaining).
- `connected_slaves`: Number of connected slaves

Debug Info

Replication (if replica/slave)

- `master_host`: Host or IP address of the master
- `master_port`: Master listening TCP port
- `master_link_status`: Status of the link (up/down)
- `master_last_io_seconds_ago`: Number of seconds since the last interaction with master
- `master_sync_in_progress`: Indicate the master is SYNCing to the slave

Debug Info

Replication (if currently `SYNCing`)

- `master_sync_left_bytes`: Number of bytes left before `SYNCing` is complete
- `master_sync_last_io_seconds_ago`: Number of seconds since last transfer I/O during a `SYNC` operation

Debug Info

Replication (if master unreachable)

- `master_link_down_since_seconds`: Number of seconds since the link is down

Debug Info

CPU

- `used_cpu_sys`: System CPU consumed by the Redis server
- `used_cpu_user`: User CPU consumed by the Redis server
- `used_cpu_sys_children`: System CPU consumed by the background processes
- `used_cpu_user_children`: User CPU consumed by the background processes

Debug Info

Commandstats (for each command)

- `cmdstat_XXX:calls=XXX,usec=XXX,usec_per_call=XXX`

Debug Info Keyspace

For each database, the following line is added:

- `dbXXX:keys=XXX,expires=XXX`

Debug Info

Details about each INFO field:

<http://redis.io/commands/info>

Manual data collection

Manual Data Collection

Node topology

Manual Data Collection

Node topology



Manual Data Collection

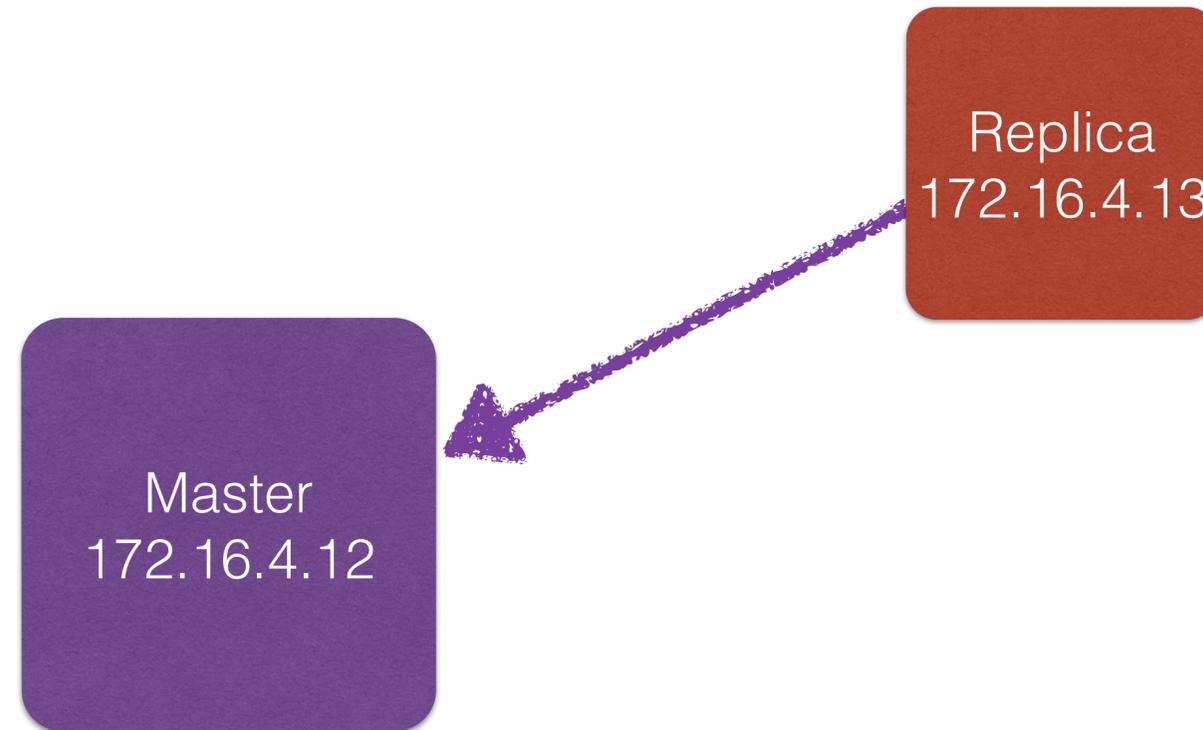
Node topology

Master
172.16.4.12

Replica
172.16.4.13

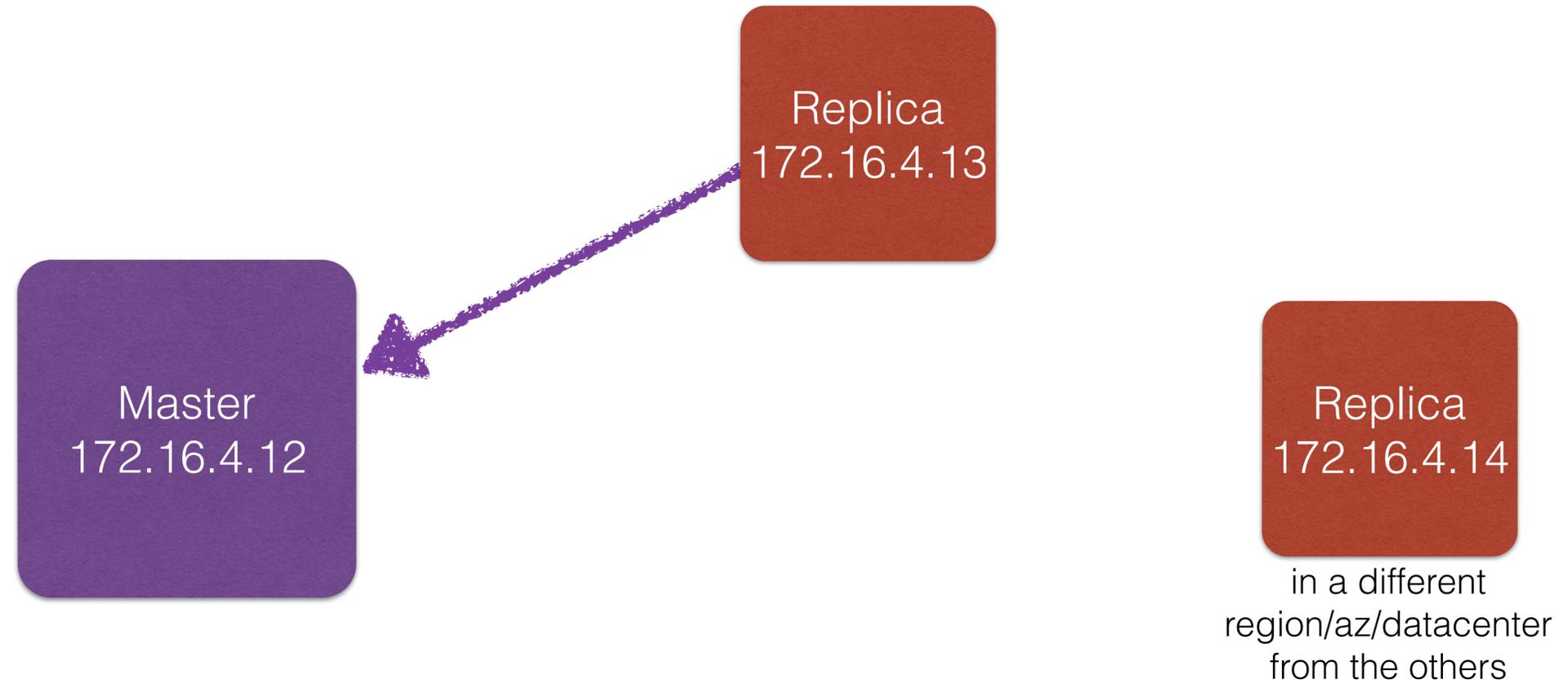
Manual Data Collection

Node topology



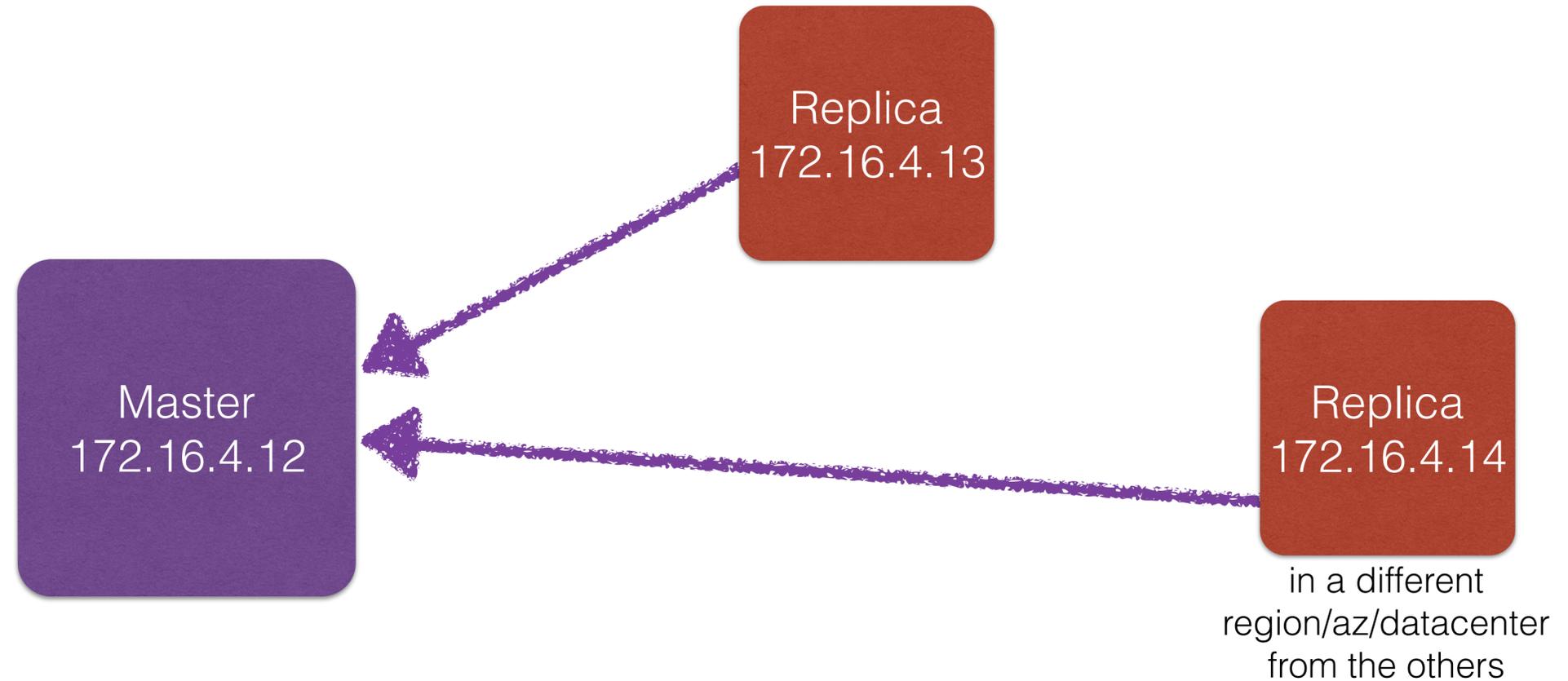
Manual Data Collection

Node topology



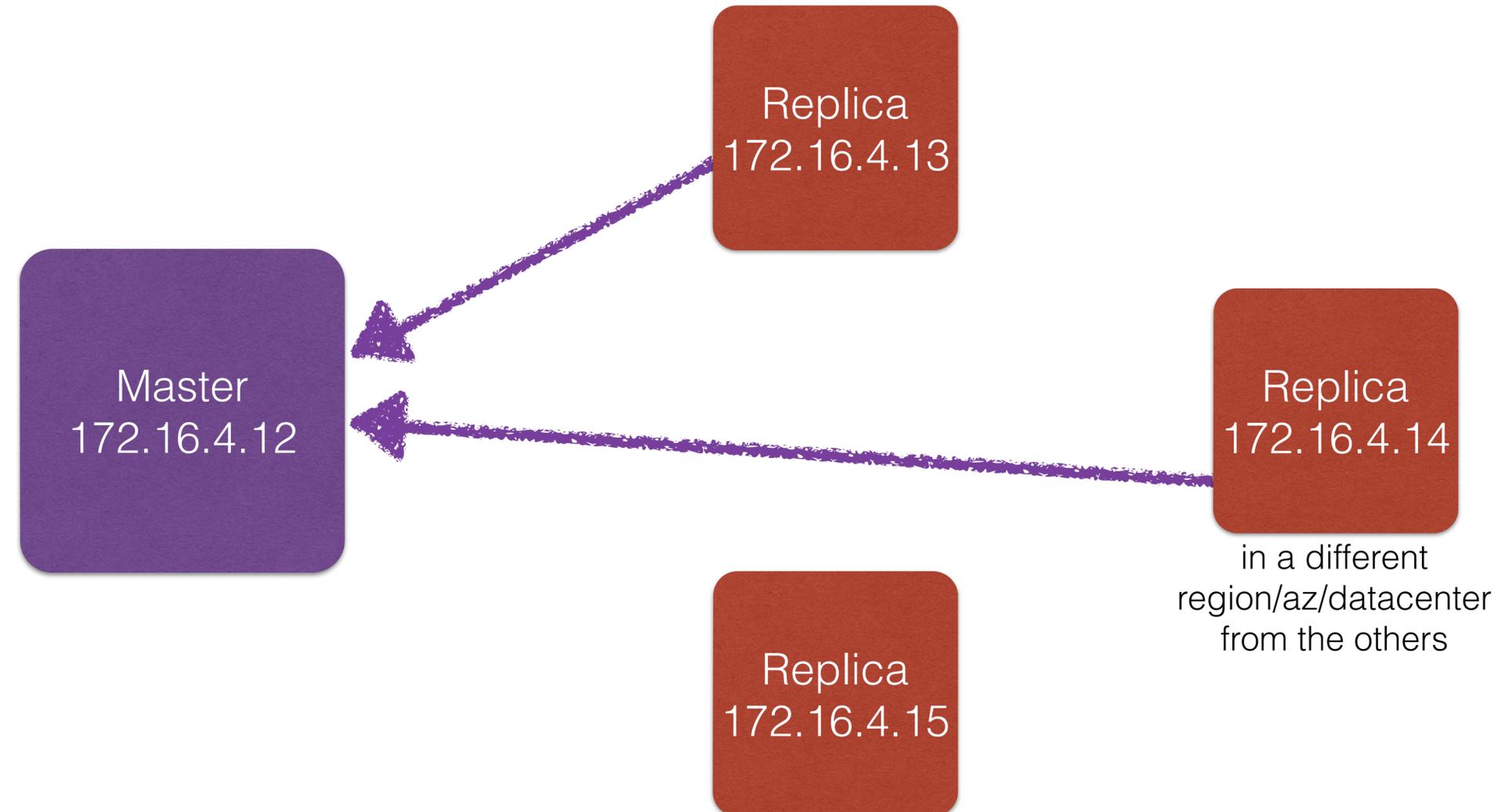
Manual Data Collection

Node topology



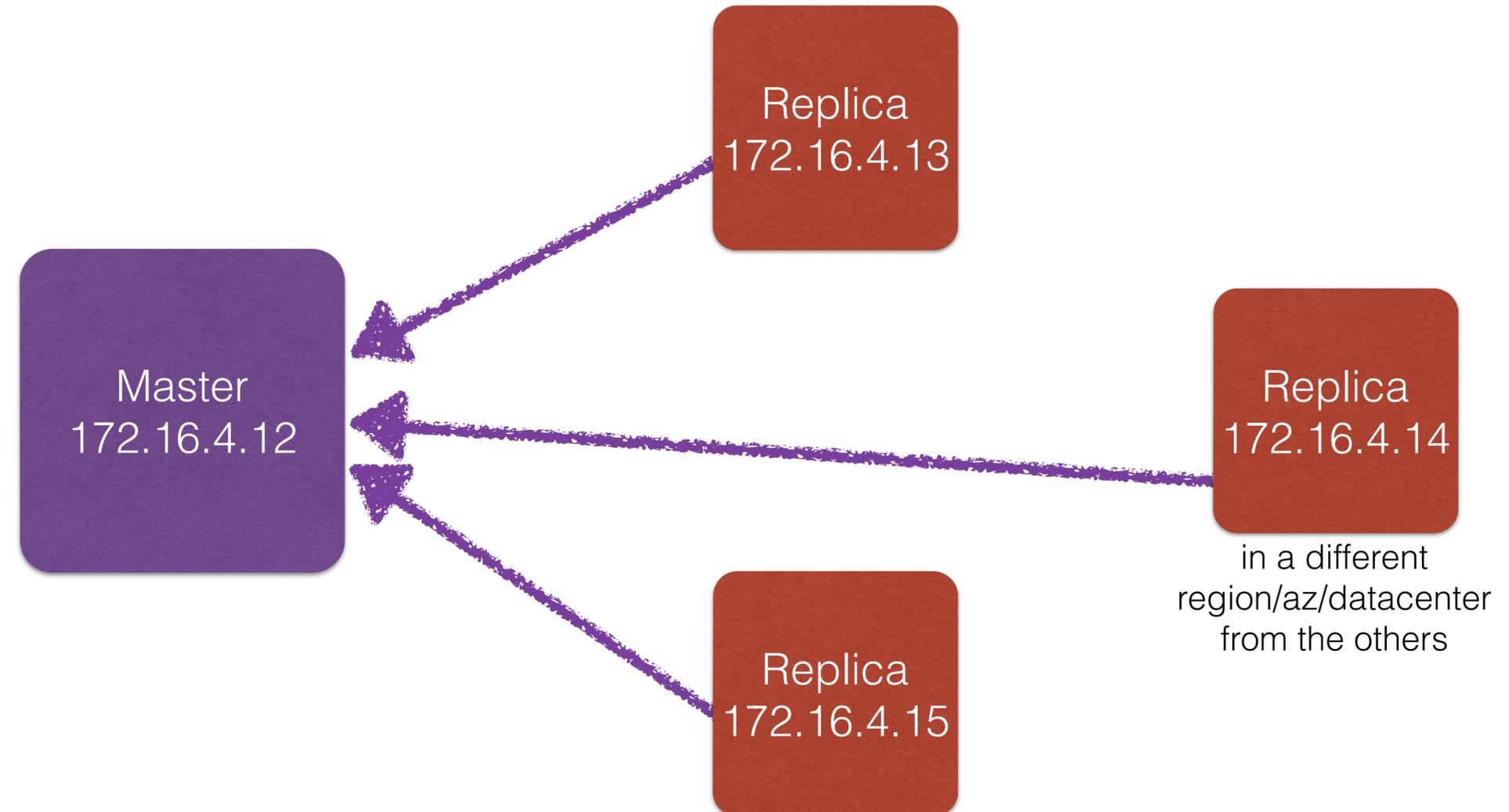
Manual Data Collection

Node topology



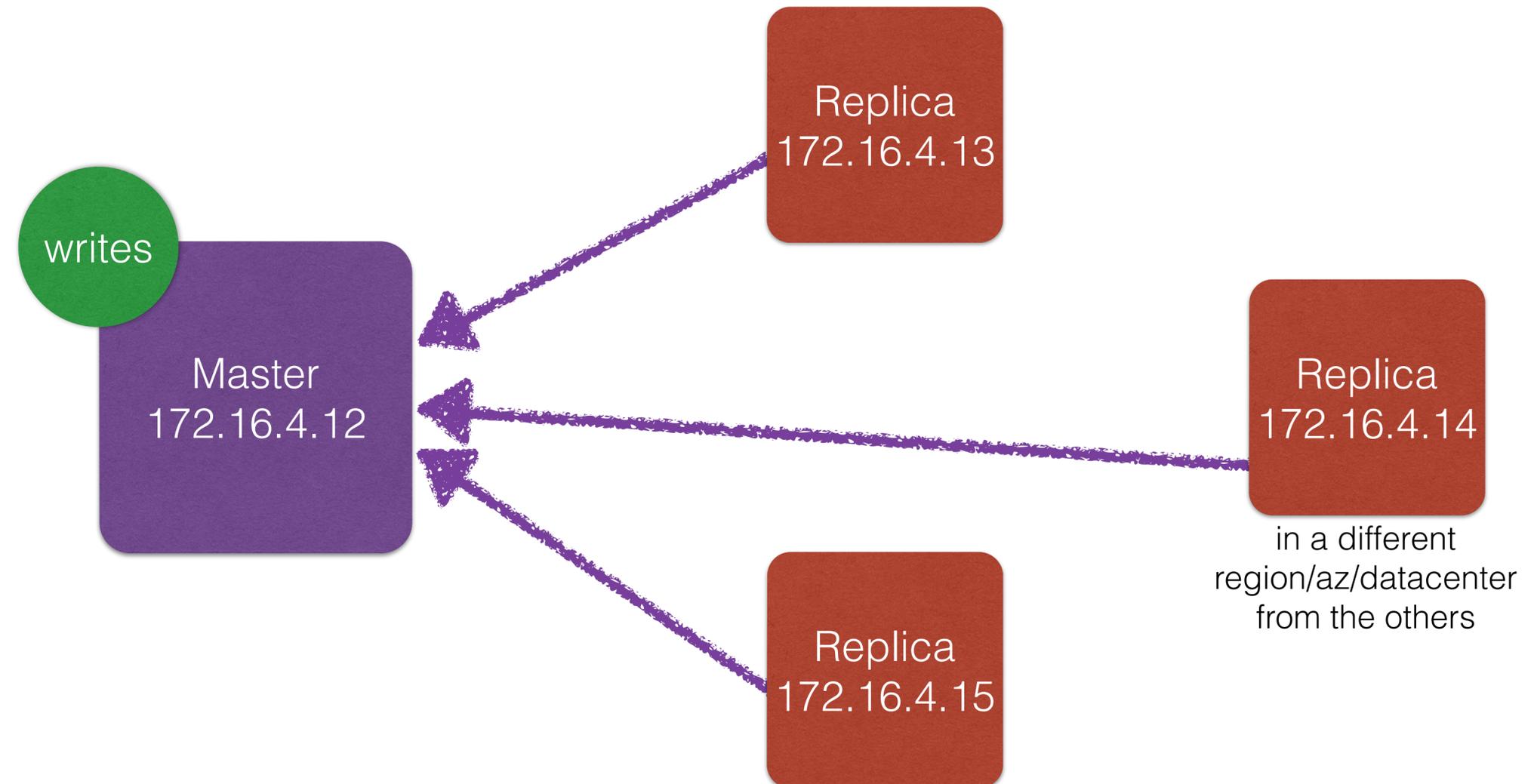
Manual Data Collection

Node topology



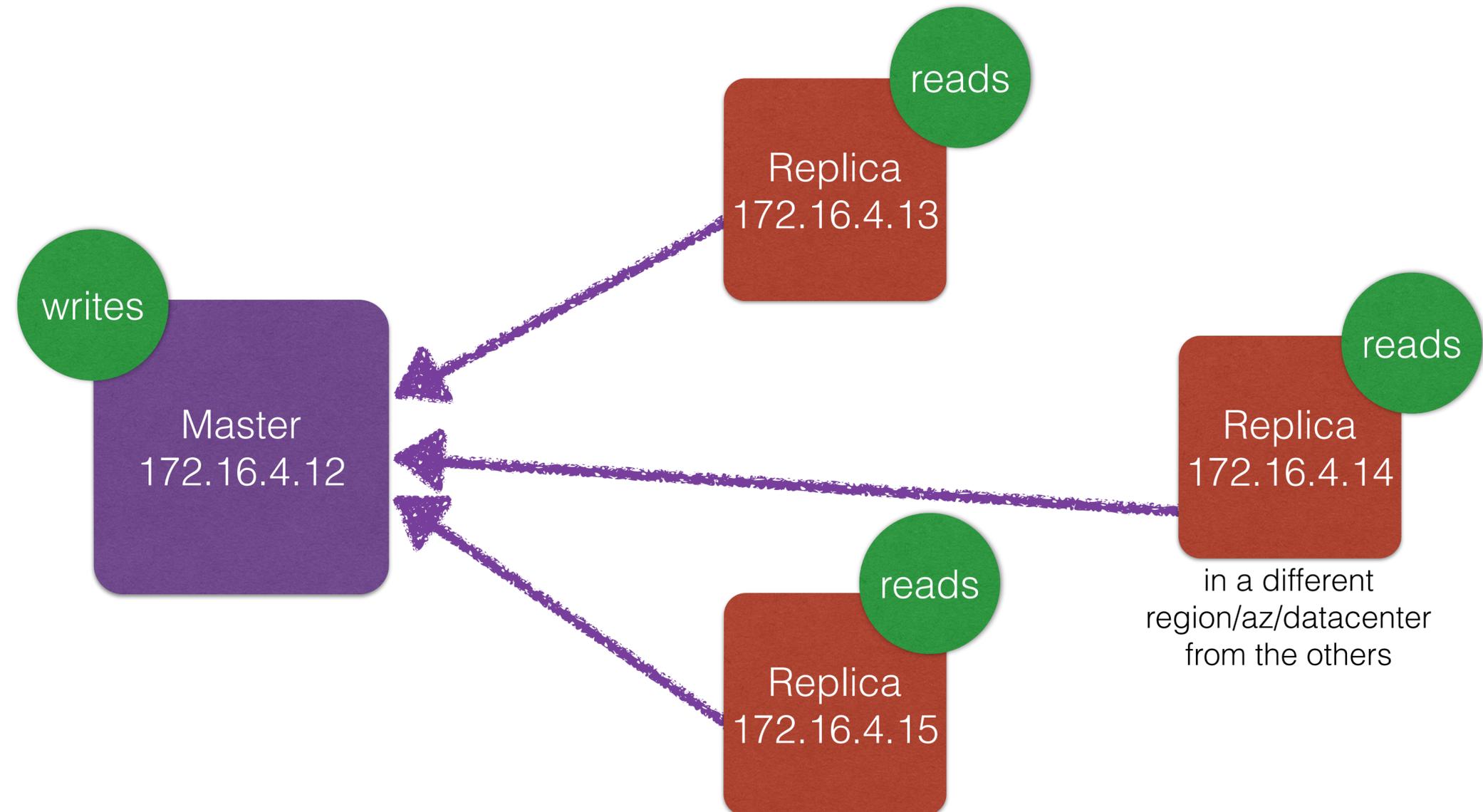
Manual Data Collection

Node topology



Manual Data Collection

Node topology



Manual Data Collection

Node Info

Manual Data Collection

Node Info

Make pictures

Manual Data Collection

Node Info

Make pictures

Config file from each instance

Manual Data Collection

Node Info

Make pictures

Config file from each instance

INFO or error report from each instance

Manual Data Collection

Client Info

Manual Data Collection

Client Info

Programming language version

Manual Data Collection

Client Info

Programming language version

Redis client and its version

Manual Data Collection

Client Info

Programming language version

Redis client and its version

Direct communication to Redis?
Using a proxy? Sharding? Hashing?

Manual Data Collection

System Info

Manual Data Collection

System Info

OS vendor with version

Manual Data Collection

System Info

OS vendor with version

Memory information

Manual Data Collection

System Info

OS vendor with version

Memory information

```
Linux: vmstat -SM 1 10; vmstat -SM -s;  
free -m; top -b -d 3 -n 3 -M -m; df -h; iostat -k 1 12
```

Manual Data Collection

System Info

OS vendor with version

Memory information

```
Linux: vmstat -SM 1 10; vmstat -SM -s;  
free -m; top -b -d 3 -n 3 -M -m; df -h; iostat -k 1 12  
OS X: memory_pressure; vm_stat -c 10 1;  
top -o rsize -i 3 -l 3; df -h; iostat -c 12
```

Manual Data Collection

Network Info

Manual Data Collection

Network Info

On-site vs. hosted vs. VPS?

Manual Data Collection

Network Info

On-site vs. hosted vs. VPS?

```
ping -c 20 [redis server]  
redis-cli --latency -h [host] -p [redis port]
```

Manual Data Collection

Network Info

On-site vs. hosted vs. VPS?

```
ping -c 20 [redis server]  
redis-cli --latency -h [host] -p [redis port]
```

Any VPN or ssh tunneling?
(including, but not limited to: stunnel, stud, spiped)

Figure it out

Figure it out

Figure it out

<http://redis.io/documentation>

Figure it out

<http://redis.io/documentation>

Explains specific Redis details
with proper usage examples
(and fixes)

Figure it out

Figure it out

<http://redis.io/commands>

Figure it out

<http://redis.io/commands>

Detailed, accurate

Figure it out

<http://redis.io/commands>

Detailed, accurate

Notes changes across versions

Figure it out

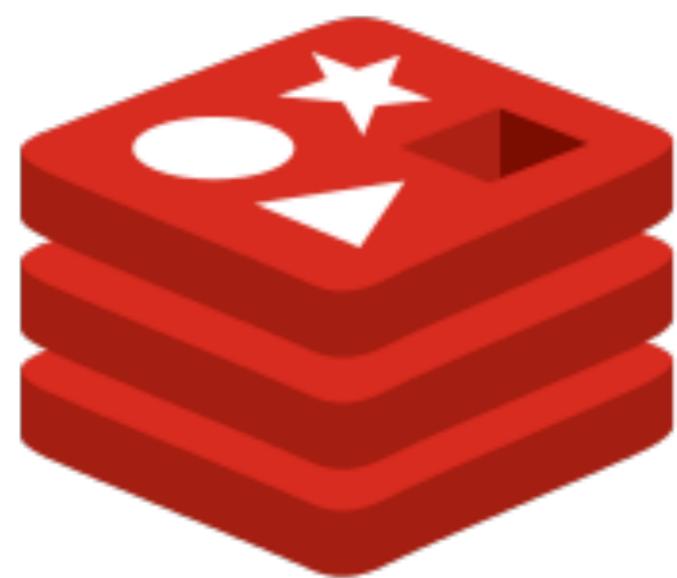
Figure it out

Ask Google

Figure it out

Figure it out

Ask Us



redis