... perhaps the fastest NoSQL database in the world

What is Redis?

Remote **Di**ctionary **S**erver

- Often described as **Key / Value** NoSQL database
- Better description: **Data structures** server
- **"Swiss Army Knife"**, collection of small useful data structures

Redis most important feature: **High performance**

- In-memory database
- Small codebase (20000 lines), native C
- Connection via TCP or UNIX socket
  (no REST interface)
- Limited choice of key mappings
  (5 types, 2 special types)
- No nested data structures

More Redis features:

- Persistence via **Snapshotting** and/or **Journalling**
- **Master/Slave chain** database replication
- **Sentinel** server monitoring
- For real clustering -> **redis-cluster** project
  (beta, not production-ready yet)
- Keys can have **expiry time**
- **Publish / Subscribe** system

Sounds cool, but what are the **Use Cases?**

- memcached++
- Statistics collection (downloads, hits, dwell times ...)
- Log buffers
- Task queues
- Share state between processes (common 'blackboard')
- Inter-process communication (channels)

Starting up Redis (on Debian) ...

- Start the server: `$> /etc/init.d/redis-server start`
- Configuration in: `/etc/redis/redis.conf`
- Log files in: `/var/log/redis/redis-server.log`
- Database dumps: `/var/lib/redis/dump.rdb`
- Journal: `/var/lib/redis/appendonly.aof`

Accessing Redis ...

- Command line tool: `$> redis-cli`
- Socket access: `$> telnet <redishost> 6379`
- Libraries:
  - redis-py (Python)
  - redis-rb (Ruby)
  - hiredis (C)
  - Jedis (Java)
  - ... and many more (http://redis.io/clients)

More Redis tools ...

- Benchmarking tool:

```
$> redis-benchmark
```

- Database consistency check:

```
$> redis-check-dump <database dump>
```

- Journal consistency check:

```
$> redis-check-aof <aof file>
```

General commands:

```
redis> PING                    # Check server connection
PONG
redis> INFO                    # Get server information
# Server
redis_version:2.8.13
[...]
redis> HELP                    # Command help
[...]
redis> SELECT 0                # Select database #no
OK
redis> FLUSHDB                 # Clear current database
OK
redis> SHUTDOWN                # Shutdown the server
redis> QUIT                    # Quit the session
```

Redis stores data in **key / value** pairs, **<value>** one of:

- `String`
- `Hash`
- `List`
- `Set`
- `Sorted Set`

and two **special** structures (derivative of *String*):

- `Bitmaps`
- `HyperLogLogs`

Basic **key / value** pairs ('Strings') :

```
redis> SET currentuser Max
OK
redis> GET currentuser
"Max"
redis> SET count 1
OK
redis> INCR count
(integer) 2
redis> GET count
"2"
```

Hashes:

```
redis> HSET users:123 name Max
(integer) 1
redis> HSET users:123 password Super_secret
(integer) 1
redis> HGET users:123 name
"Max"
redis> HGET users:123 password
"Super_secret"
redis> HKEYS users:123
1) "name"
2) "password"
```

Lists:

```
redis> RPUSH users Max John Peter
(integer) 3
redis> LLEN users
(integer) 3
redis> LRANGE users 0 -1
1) "Max"
2) "John"
3) "Peter"
redis> LPOP users
"Max"
redis> LINDEX users 1
"Peter"
```

Sets:

```
redis> SADD favorites:news BBC NYT Wired
(integer) 3
redis> SADD favorites:tech Wired Heise
(integer) 2
redis> SINTER favorites:tech favorites:news
1) "Wired"
redis> SUNION favorites:tech favorites:news
1) "Heise"
2) "Wired"
3) "NYT"
4) "BBC"
```

Sorted Sets: (Entries sorted by *Score*)

```
redis> ZADD favorites 15 BBC 10 NYT 80 Heise 50 Wired
(integer) 4
redis> ZSCORE favorites NYT
"10"
redis> ZRANGEBYSCORE favorites 40 inf
1) "Wired"
2) "Heise"
```

Key expiry:

```
redis> SET icecream "Like ice in the sunshine"
OK
redis> EXPIRE icecream 20
(integer) 1
[... after 6 seconds ...]
redis> TTL icecream
(integer) 14
redis> GET icecream
"Like ice in the sunshine"
[... after half a minute ...]
redis> TTL icecream
(integer) -1
redis> GET icecream
(nil)
```

Publish / Subscribe:

Subscriber

Publisher

1
```
redis> SUBSCRIBE channel
Reading messages...
(press Ctrl-C to quit)
1) "subscribe"
2) "channel"
3) (integer) 1
[...]
```

2
```
redis> PUBSUB CHANNELS
1) "channel"
[...]
```

3
```
redis> PUBLISH channel
"Hi there!"
(integer) 1
```

4
```
1) "message"
2) "channel"
3) "Hi there!"
```

Bitmaps:

```
redis> SET bitkey "\xff"
OK
redis> SETBIT bitkey 4 0
(integer) 1
redis> GET bitkey
"\xf7"
redis> BITCOUNT bitkey
(integer) 7
```

Lua scripting:

```lua
-- usernames.lua
-- Get field "name" from all users

local users = redis.call('KEYS', 'users:*')
local names = {}
for num,key in ipairs(users) do
  names[num] = redis.call('HGET', key, 'name')
end
return names
```

```
$> redis-cli EVAL "$(cat usernames.lua)" 0
```

Hyperloglogs:

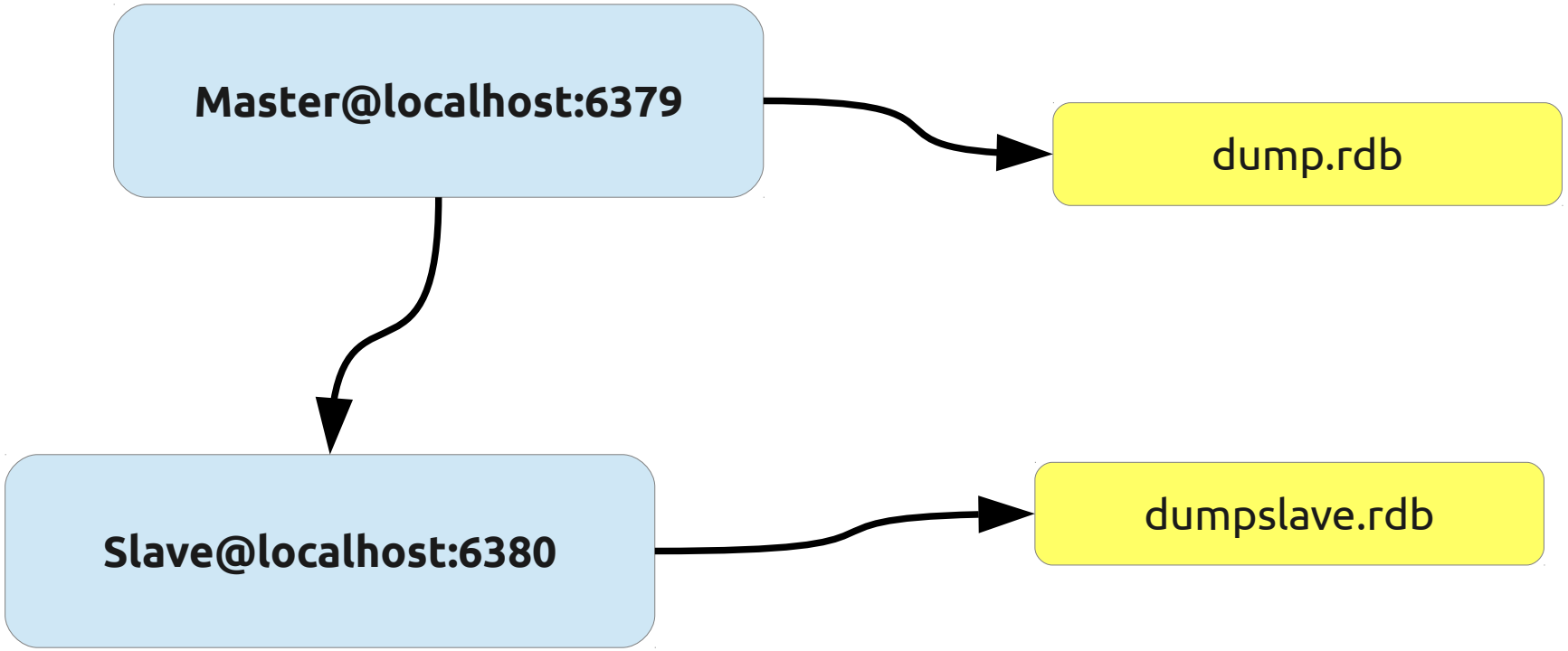
(approximate cardinality of huge sets with small memory demand)

```
redis> PFADD hll a b c d a a
(integer) 1
redis> PFCOUNT hll
(integer) 4
redid> PFADD hll a a a a a
(integer) 0
redis> PFCOUNT hll
(integer) 4
```

redis

Master / Slave DB replication:

```
$> redis-server --port 6380 --slaveof localhost 6379
--dbfilename dumpslave.rdb
```



Redis homepage:

- www.redis.io

Books:

- The little Book of Redis (free)
- Redis in Action (Manning)
- Redis Cookbook (O'Reilly)

Questions from the SoCraTes 2014 session:
(and my attempts at an answer, please feel free to correct)

- Is it possible to access sets via wildcards?

  -> Doesn't seem so. Looks like only the **KEYS** command accepts wildcards.

- Is there a performance penalty when accessing different database numbers?

  -> Didn't find anything, but there shouldn't be any significant penalty.

Questions from the SoCraTes 2014 session (cont.):
(and my attempts at an answer, please feel free to correct)

- Is there a mechanism to notify a client when a key is changed?

  -> **Yes**, 2.8 introduced Keyspace Notifications, see:

  http://redis.io/topics/notifications