GoRedis介绍

陌陌/李志威/CTO

“GoRedis是基于RocksDB，使用Go语言编写的高效Redis Server，提供Redis所不具备的众多特性。”

-goredis.io

RocksDB

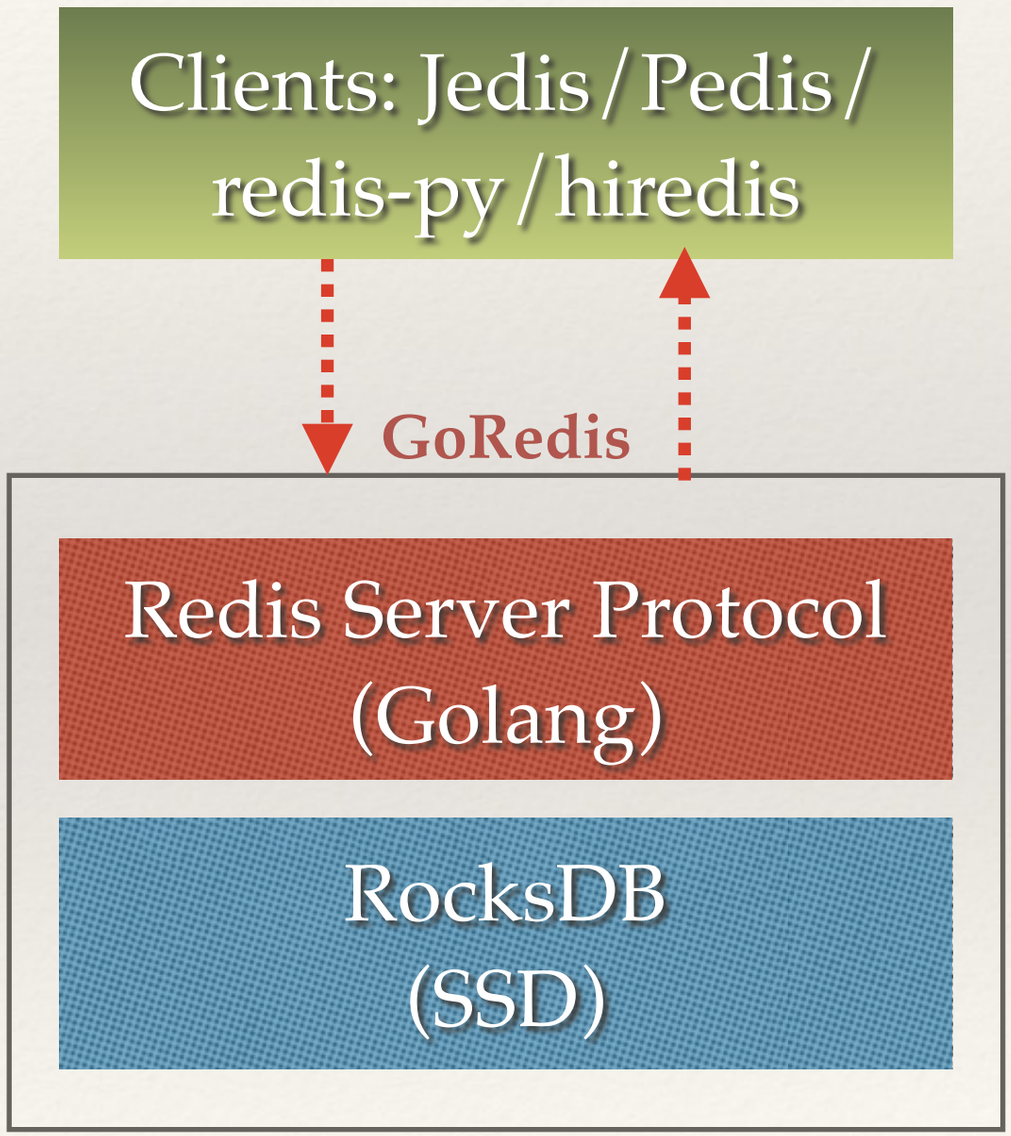
- RocksDB由Facebook基于LevelDB改进而来，能充分利用多核CPU，并提供系列的扩展特性
- LevelDB是Google开源的高性能Key-Value数据库，基于文件系统，主要解决高并发随机写，以及海量存储的需求，配合SSD能提供非常高的随机读性能
- Key有序存储，让RocksDB实现Redis数据结构提供了可行性

Go语言

- Go是Google在近年推出的一个全新的编程语言
- 充分利用多CPU；强类型；简化开发；貌美C/C++的速度

Redis Server

- 实现了Redis协议，并支持大部分数据指令
- 对开发者来说，服务端是GoRedis或是Redis基本无异



GoRedis特性

- 基于RocksDB获得大量特性
- “零”内存，数百万Field的Hash，几千万Member的Sorted Set，都不会消耗额外的内存
- 海量存储，基于RocksDB能提供亿级的数据存储
- 性能可观，在SSD下，Get/Set 6w+/s

GoRedis特性

- 快速启动，没有类似Reload RDB的过程，可以快速重启实例
- 增量同步，GoRedis主从情况下，从库断开后不会全量同步（参考MongoDB）
- MultiSlaveOf，一个GoRedis可以同时作为多个Redis的从库，实现集群备份
- 完善日志，为DBA提供完善的各类日志输出

适用场景

- 海量数据存储
- 安全数据存储
- 节省内存

性能测试

- 以下测试基于Flash卡
- redis-benchmark -p 1602 -n 1000000 -c 20 -t set,get -r 100000000
- SET：68742.70 requests per second
- GET：72150.07 requests per second

性能测试

- redis-benchmark -p 1602 -n 1000000 -c 20 -r 100000000 rpush mylist __rand_int__
- RPUSH： 25012.51 requests per second

磁盘IO

参数/型号	SAS (15K/Raid1)	SSD MLC (Raid1)	SSD SLC (Raid1)	Flash
容量	600G	200G	200G	600G
随机读带宽(64K)	45MB/s	512MB/s	705MB/s	2.5GB/s
随机写带宽(64K)	30MB/s	195MB/s	138MB/s	962MB/s
随机读(4KB) IOPS	854	59887	102694	410369
随机写(4KB) IOPS	584	20470	9677	241557
顺序读(4KB) IOPS	79137	67205	124113	337635
顺序写(4KB) IOPS	180171	120899	163866	247138
随机读写(R:W 3:1)	**790**	**45660**	**32128**	**413989**

以上数据为特定配置和环境下小样本测试，仅供参考

“使用RocksDB实现各种Redis数据结构。”

存储结构

- 每个Key在RocksDB里都有这样的数据结构
- +[key]type = value
- 对于String以外的复杂结构，会使用额外的数据来存储
- 不同指令，性能不同

Redis结构	RocksDB结构
String	+[key]string
Hash/Set	+[key]hash
List	+[key]list
Sorted Set	+[key]zset

String

- Redis: SET name latermoon
- RocksDB: Put(‘+[name]string’, ‘latermoon’)

key	value
+[name]string	latermoon

Hash/Set

- Redis: HMSET user name latermoon age 28 sex Male
- RocksDB:
- Put(‘+[user]hash’, ‘’)
- Put(‘_h[user]name’, ‘latermoon’)
- Put(‘_h[user]age’, ’28’)
- Put(‘_h[user]sex’, ‘Male’)

key	value
+[user]hash	**null**
_h[user]name	latermoon
_h[user]age	28
_h[user]sex	Male

List

- Redis: LPUSH mylist a b c
- RockDB:
  - Put(‘+[mylist]list’, ‘0,2’)
  - Put(‘_l[mylist]#0’, ‘a’)
  - Put(‘_l[mylist]#1’, ‘b’)
  - Put(‘_l[mylist]#2’, ‘c’)

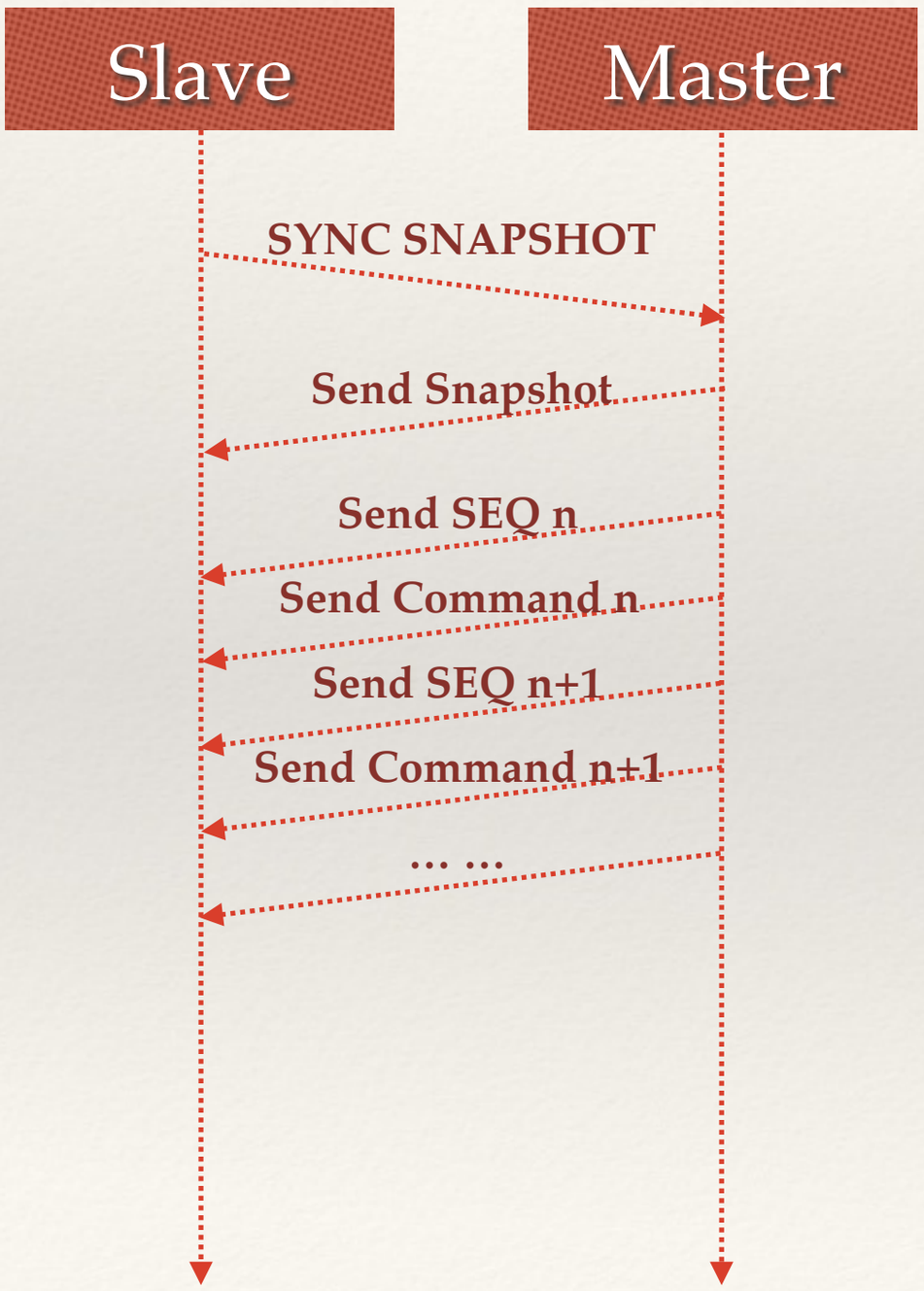
key	value
+[mylist]list	**0,2**
_l[mylist]#0	a
_l[mylist]#1	b
_l[mylist]#2	c

Sorted Set

- Redis: ZADD myzset 1 a 2 b
- RockDB:
- Put('+[myzset]zset', '2')
- Put('_z[myzset]m#a', '1')
- Put('_z[myzset]m#b', '2')
- Put('_z[myzset]s#1#a', '')
- Put('_z[myzset]s#2#b', '')

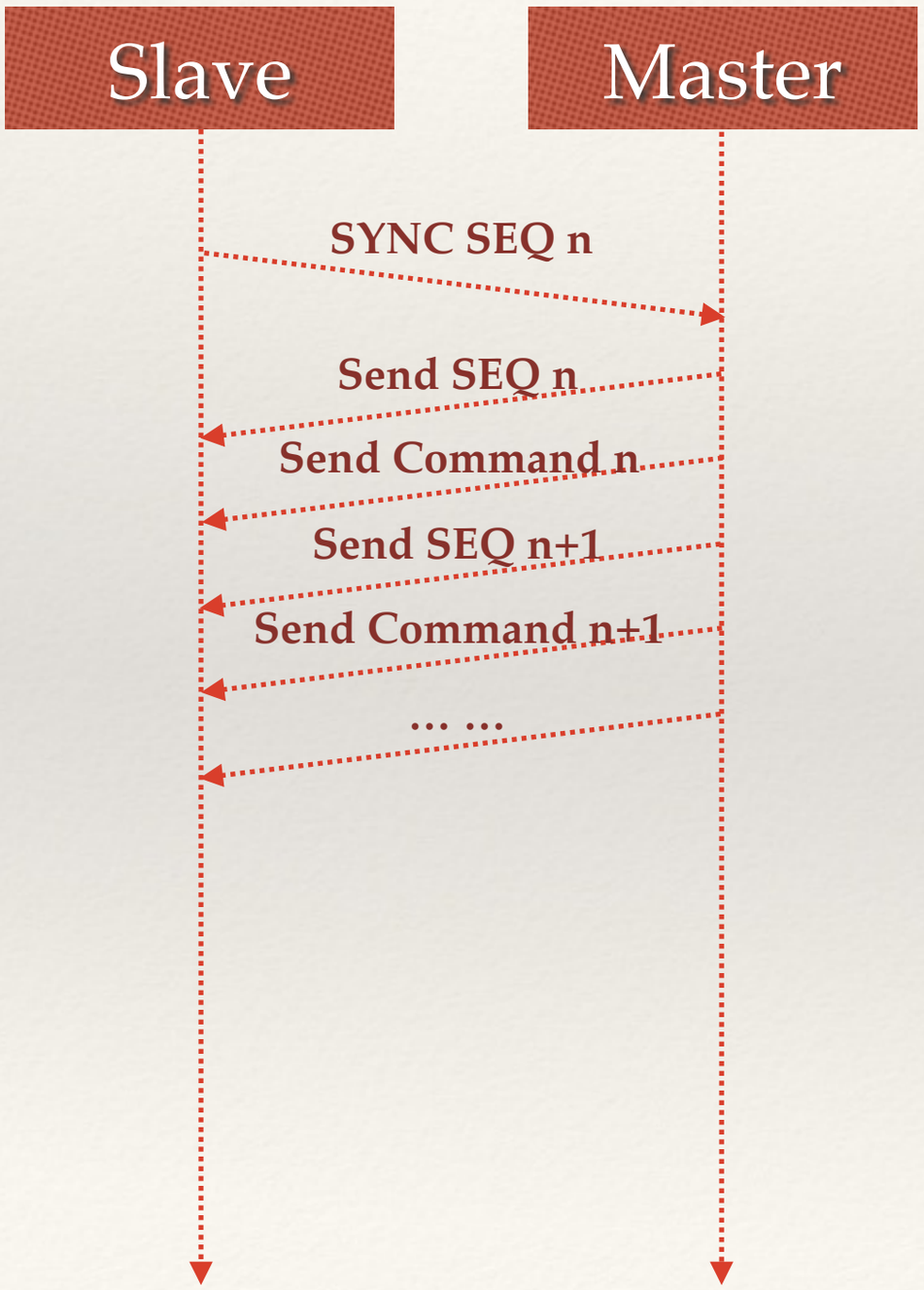
key	value
+[myzset]zset	**2**
_z[myzset]m#a	1
_z[myzset]m#b	2
_z[myzset]s#1#a	null
_z[myzset]s#2#b	null

增量同步



- 首次同步，Slave发出SYNC SNAPSHOT指令
- Master先把完整的数据快照发送给Slave
- 然后把实时指令，及序号SEQ实时发送给Slave，实现同步

增量同步



- Slave重连时，会把最后一次接收到的序号SEQ发给Mater，请求接收增量数据
- Master从本地日志队列里，找到对应的SEQ，补发给Slave
- 补发完成后，会继续发送实时数据，实现增量同步

日志

- stdlog.log 标准输出
- cmd.log 指令日志
- cmd.string.log 字符串指令日志
- cmd.hash.log Hash指令日志
- cmd.list.log List指令日志
- cmd.zset.log SortedSet指令日志
- exec.time.log 指令耗时日志
- leveldb.io.log rocksdb指令数日志
- seq.log 主从同步日志
- slow.log 慢查询日志

time	LPUSH	RPUSH	LPOP	RPOP	LINDEX	LLEN	LRANGE	LTRIM
03:47:59	50	0	0	0	0	0	185	50
03:48:00	34	0	0	0	0	0	141	34
03:48:01	46	0	0	0	0	0	164	46
03:48:02	50	0	0	0	0	0	184	50
03:48:03	33	0	0	0	0	0	137	33
03:48:04	39	0	0	0	0	0	147	39
03:48:05	50	0	0	0	0	0	178	50
03:48:06	40	0	0	0	0	0	152	40
03:48:07	50	0	0	0	0	0	162	50
03:48:08	37	0	0	0	0	0	148	37

time										ion
03:46:49	1102	34	0	0	206	0	538	323	0	0	164
03:46:50	1499	54	0	0	288	0	775	382	0	0	164
03:46:51	1064	36	0	0	204	0	530	294	0	0	164
03:46:52	1215	43	0	0	242	0	619	311	0	0	164
03:46:53	1641	51	0	0	300	0	825	465	0	0	164
03:46:54	1423	50	0	0	252	0	744	377	0	0	164
03:46:55	1432	48	0	0	260	0	736	388	0	0	164
03:46:56	1569	48	0	0	284	0	784	453	0	0	164
03:46:57	1413	43	0	0	248	0	730	392	0	0	164
03:46:5											

time	get	set	batch	enum	del	lru_hit	lru_miss
03:48:39	504	622	573	602	0	744	257
03:48:40	531	695	637	619	0	802	257
03:48:41	411	514	471	494	0	616	212
03:48:42	563	725	666	651	0	836	272
03:48:43	425	498	451	496	0	586	233
03:48:44	374	494	451	416	0	539	180
03:48:45	490	600	552	554	0	675	251
					0	789	278
					0	493	179
					0	765	245

time	<1ms	1-5ms	6-10ms	11-30ms	>30ms
03:50:49	1133	2	0	0	0
03:50:50	1221	6	0	0	0
03:50:51	1158	4	1	0	0
03:50:52	1471	1	0	0	0
03:50:53	1314	5	0	0	0
03:50:54	1653	5	0	0	0
03:50:55	1303	4	0	0	0
03:50:56	1402	6	0	0	0
03:50:57	1370	3	1	0	0
03:50:58	1396	2	0	0	0

扩展指令

- 为GoRedis定制指令，实现更多实用功能
- doc指令集提供面向Document的数据结构
- key指令集提供数据枚举功能
- info指令提供丰富的状态信息

Doc指令集

- 对一个Key提供面向Document的DOC_SET和DOC_GET指令
- doc_set(key, {"name":"latermoon"})
- doc_set(key, {"$rpush":["photos", "d.jpg", "e.jpg"]}})
- doc_set(key, {"$incr":["version", 1]})doc_set(key, {"setting.mute.start":23, "setting.mute.end":8})
- doc_set(key, {"$del":["name", "setting.mute.start"]})
- doc_get(key)
- doc_get(key, "name,sex,photos,setting.mute,version")

数据：

```
user:300000:profile = {
        name: "latermoon", // string
        sex: 1 // int
        photos: ["a.jpg", "b.jpg", "c.jpg"],
        setting: { // hash
                mute: { start: 23, end: 8 }
        },
        is_vip: true, // bool
        version: 172 // int
}
```

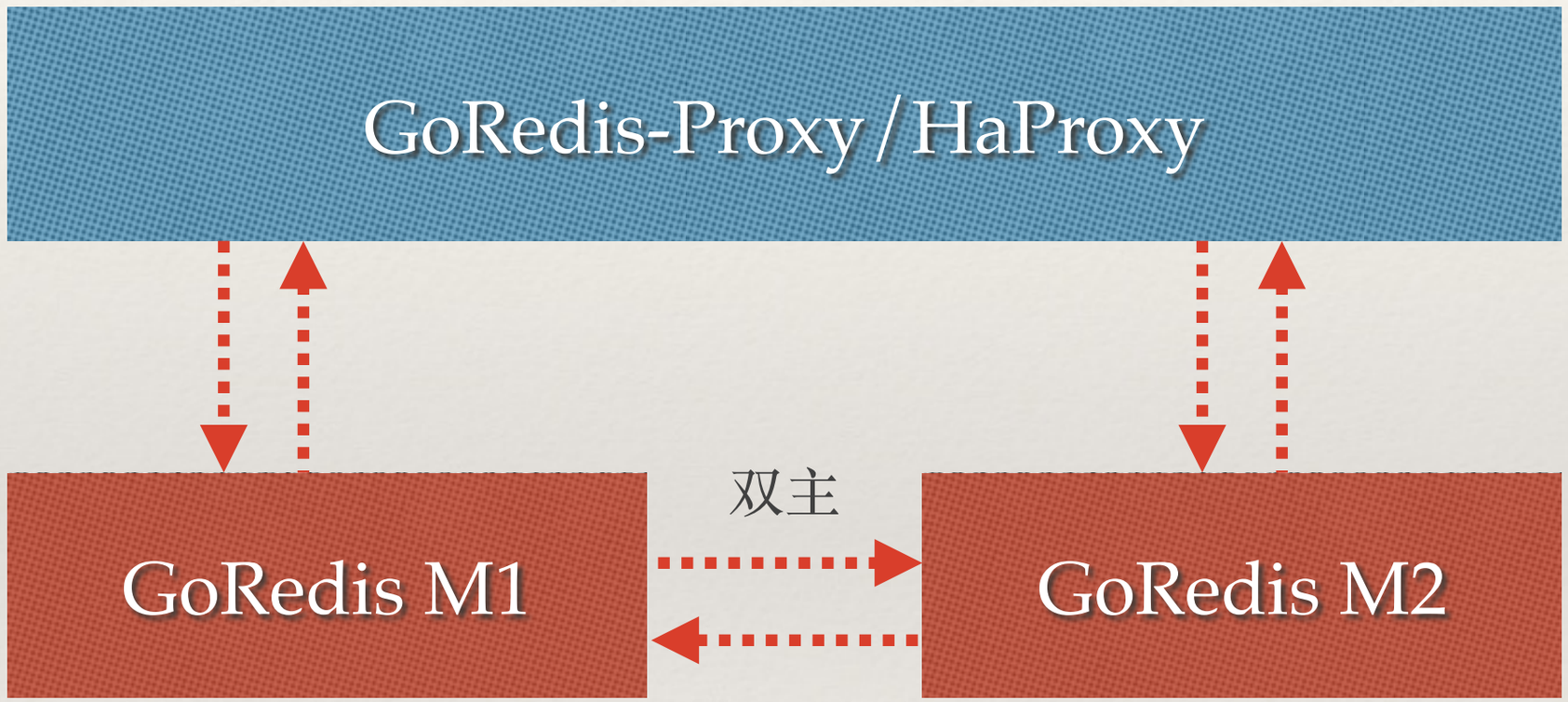
Keys支持

- 在大量数据的情况下，官方Redis的Keys操作会阻塞进程，处于不可以状态
- 而RocksDB的Key是有序可扫描的，因此可以提供
- key_next [seek] [count] [withtype] [withvalue]
- 实现稳定的全表扫描

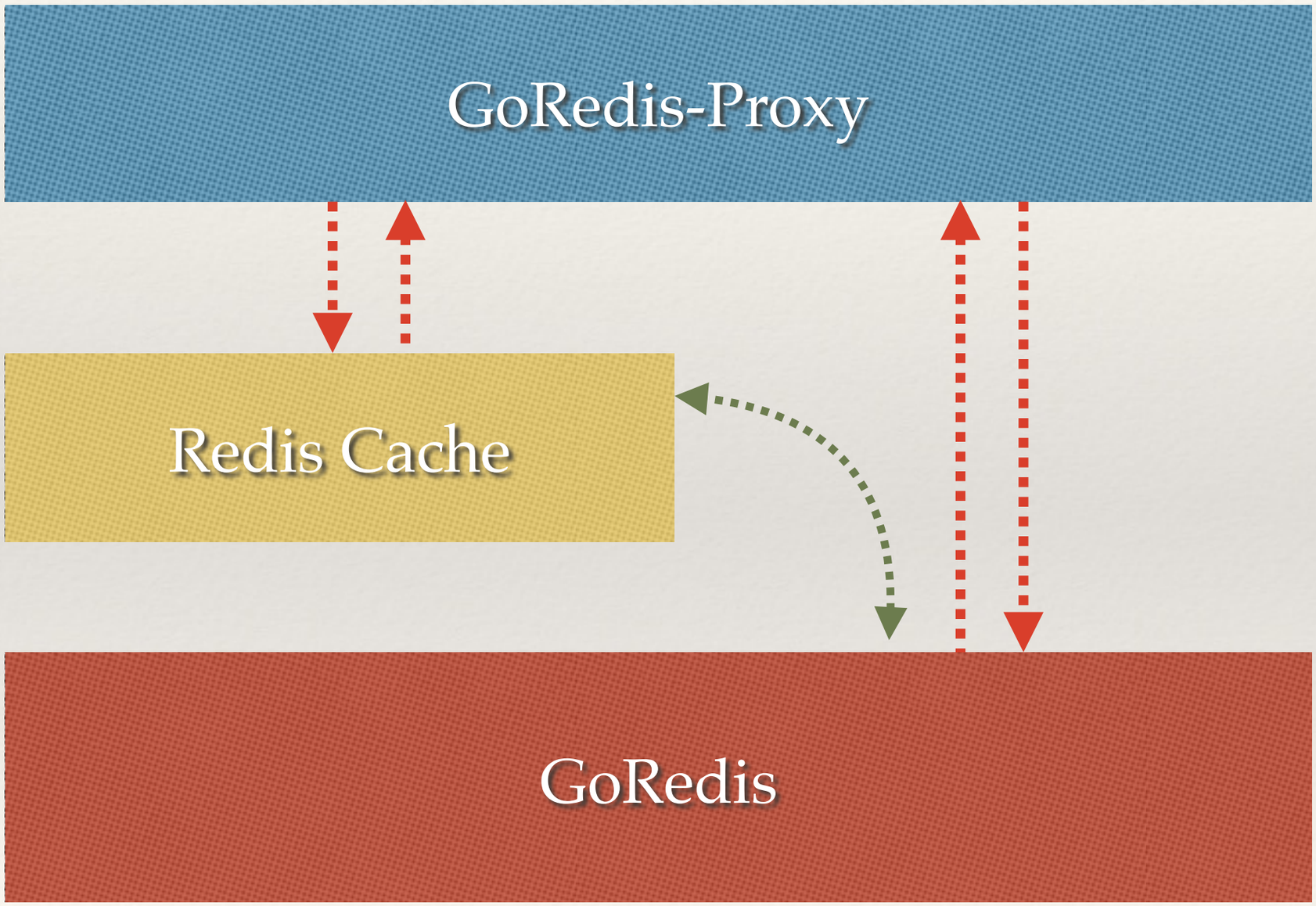
AOF指令

- AOF [yes/no]
- 生成数据副本

高可用



Cache模式



二次开发

- Go语言版本的RedisServer框架
- 添加OnXXX(cmd *Command)即可添加自定义指令

```go
func (server *GoRedisServer) OnPING(cmd *Command) (reply *Reply) {
    reply = StatusReply("PONG")
    return
}
```

```go
func (server *GoRedisServer) OnHSET(cmd *Command) (reply *Reply) {
    key := cmd.StringAtIndex(1)
    hash := server.levelRedis.GetHash(key)
    field, _ := cmd.ArgAtIndex(2)
    value, _ := cmd.ArgAtIndex(3)
    hash.Set(field, value)
    return IntegerReply(1)
}
```

http://goredis.io

http://github.com/momotech/GoRedis

陌陌/李志威/*latermoon@qq.com*