ODF 2017 开源数据库论坛(北京)
OPEN-SOURCE DATABASE FORUM(BEIJING)

开源数据库正在改变世界

2017年8月24日-25日　北京－京仪大酒店

Redis based In-Memory Data Grid and Distributed Locks in Java

Nikita Koksharov, Founder of Redisson

Redisson overview (Redis based In-Memory Data Grid)

Distributed Collections	Distributed Objects
Distributed Locks and Synchronizers	Distributed Services
Local Cache	Integration with frameworks

Distributed objects

1. Object holder
2. Binary stream holder
3. Geospatial holder
4. BitSet
5. AtomicLong
6. Atomic Double
7. Publish Subscribe
8. Bloom filter

Distributed collections

1. Map (supports eviction)
2. Multimap (supports eviction)
3. Set (supports eviction)
4. SortedSet / ScoredSortedSet
5. List
6. Queue / Deque
7. Blocking Queue / Deque
8. Priority Queue / Deque
9. Delayed Queue

Distributed locks and synchro...

1. Lock
2. FairLock
3. MultiLock
4. RedLock
5. ReadWriteLock
6. Semaphore
7. CountDownLatch

Distributed services

1. Remote Service
2. Live Object Service
3. Executor Service
4. Scheduler Service
5. Map Reduce Service

Integration with frameworks

1. Spring framework
2. Spring Cache
3. Hibernate 2nd Level Cache
4. JCache API (JSR-107) implementation
5. Tomcat Session Manager
6. Spring Session

Local cache support

1. Read operations up to 45x faster
2. Integrated into
    1. Map
    2. Map with expiring entries
    3. Spring Cache
    4. Hibernate Cache

Map

```
ConcurrentMap<Integer, MyObject> map = new ConcurrentHashMap<>();

map.put(20, new MyObject("oldobj"));

map.putIfAbsent(20, new MyObject("newobj"));

map.containsKey(1);
```

Redisson Map

```
ConcurrentMap<Integer, MyObject> map = redisson.getMap("someMap");
map.put(20, new MyObject("oldobj")); // wraps HSET command
map.putIfAbsent(20, new MyObject("newobj")); // wraps LUA-script
map.containsKey(1);
```

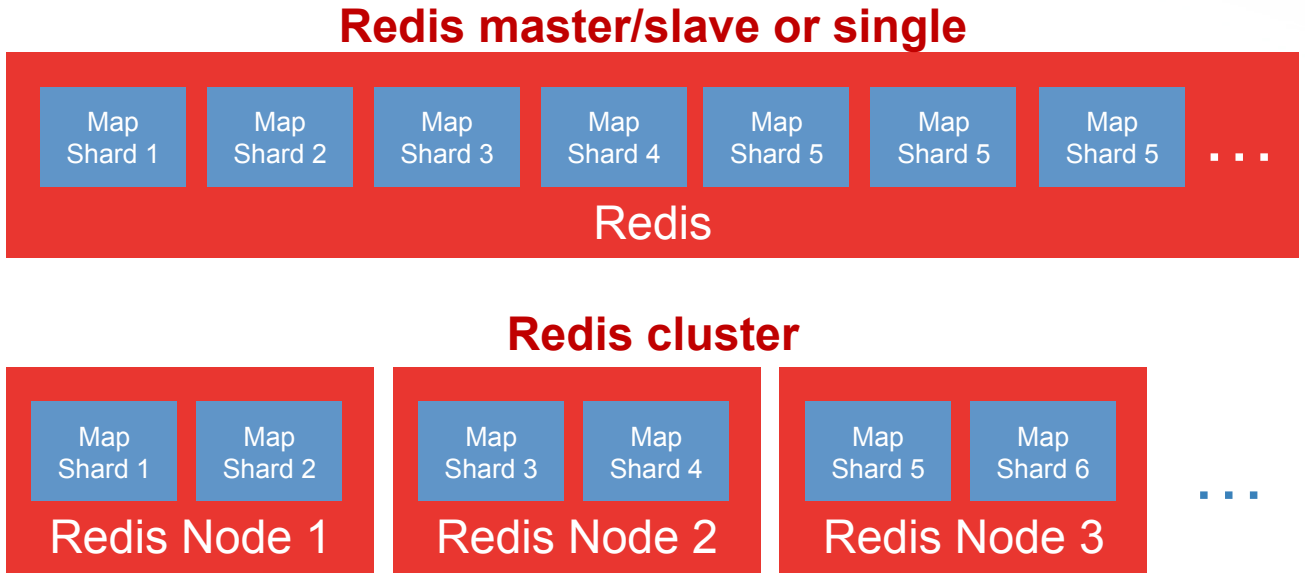
ConcurrentMap.putIfAbsent Lua Script Implementation

```
"if redis.call('hsetnx', KEYS[1], ARGV[1], ARGV[2]) == 1 then "
   "return nil; "
"else "
   "return redis.call('hget', KEYS[1], ARGV[1]); "
"end;“

// KEYS[1] = map name
// ARGV[1] = map key
// ARGV[2] = map value
```

Redisson Map Eviction

```
// implements java.util.concurrent.ConcurrentMap

RMapCache<Integer, String> map = redisson.getMapCache("someMap");

// maxIdleTime = 20 minutes

map.put(20, "oldobj", 20, TimeUnit.MINUTES);

// maxIdleTime = 20 minutes, timeToLive = 10 minutes

map.put(20, "oldobj", 20, TimeUnit.MINUTES, 10, TimeUnit.MINUTES);
```

Redisson Sharded Map



Redisson Map with Local Cache

```
ConcurrentMap<Integer, MyObject> map =
                                    redisson.getLocalCachedMap("someMap");
```

Local cached Map entry

Eviction policy

1. Last Recently Used
2. Last Frequently Used
3. Soft Reference
4. Weak Reference

Local cached Map entry Invalidation policy

1. On change
2. On change + clears whole cache on reconnect
3. On change + clears changed entries only on reconnect

Redisson references

```
RMap<String, RMap<String, String>> settings = redisson.getMap(“settings”);


RMap<String, String> options1 = redisson.getMap(“settings_server1”);
options1.put(“name”, “s1”);
options1.put(“cpu”, “80”);
// store “options” map inside “settings” map
settings.put(“server1”, options1);
// read “options” map from “settings” map
RMap<String, String> options = settings.get(“server1”);
```

Redisson lock

```
// implements java.util.concurrent.locks.Lock
RLock lock = redisson.getLock("lock");
lock.lock();
// or
lock.lock(10, TimeUnit.MINUTES);
//...
lock.unlock();
```

Locking algorithm

1. Tries to acquire lock by invoking Lua-script
2. If lock couldn't be acquired it subscribes to LockChannel
3. Does in loop:
    1. Tries to acquire lock again
    2. If lock couldn't be acquired it waits for "unlock"event from LockChannel published by current lock owner
4. Unsubscribes from LockChannel if lock has been acquired
5. During unlock method invocation it deletes lock state and publishes "unlock"event to LockChannel

Redisson fair lock

```
// implements java.util.concurrent.locks.Lock
RLock lock = redisson.getFairLock("lock");
lock.lock();
// or
lock.lock(10, TimeUnit.MINUTES);
//...
lock.unlock();
```

Fair locking algorithm

1. Tries to acquire lock by invoking Lua-script
2. If lock couldn't be acquired it subscribes to LockChannel:redissonId:threadId and stores this id into threads queue
3. Does in loop:
    1. Tries to acquire lock again
    2. If lock couldn't be acquired it waits for "unlock"event from LockChannel:redissonId:threadId published by current lock owner
4. Unsubscribes from LockChannel:redissonId:threadId if lock has been acquired
5. During unlock method invocation it deletes lock state object and publishes "unlock"event to the first LockChannel:redissonId:threadId obtained from threads queue

Redisson multilock

```
RLock lock1 = redisson1.getLock("lock1");
RLock lock2 = redisson2.getLock("lock2");
RLock lock3 = redisson3.getLock("lock3");

// implements java.util.concurrent.locks.Lock
RLock lock3 = new RedissonMultiLock(lock1, lock2, lock3);
lock.lock();
//...
lock.unlock();
```

Multilocking algorithm

1. Acquires lock one by one:
    1. Tries to acquire lock
    2. If lock on current step can’t be acquired all previously acquired locks are unlocked and locks iterator is reset to the first lock

Redisson Remote Service. Service registration

```
RRemoteService remoteService = redisson.getRemoteService();


ServiceImpl service = new ServiceImpl();
// Can handle only 1 invocation concurrently
remoteService.register(SomeService.class, service);
// Can handle up to 12 invocations concurrently
remoteService.register(SomeService.class, service, 12);
```

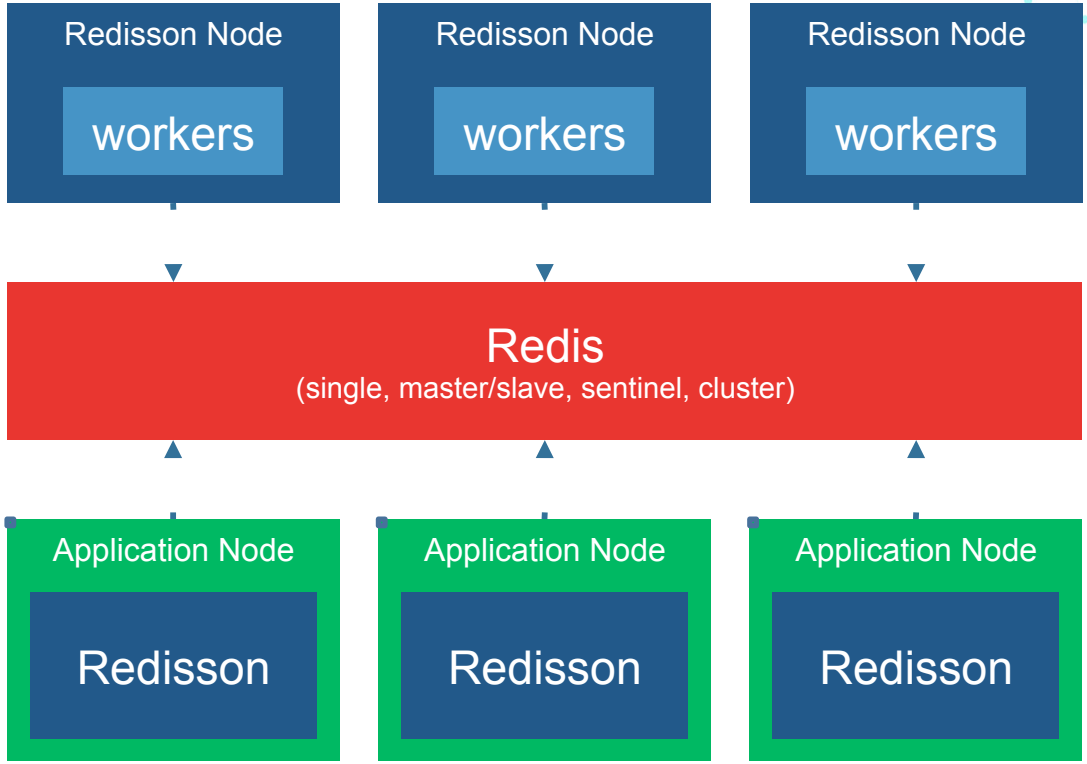
Redisson Remote Service. Service invocation

```
RRemoteService remoteService = redisson.getRemoteService();


SomeService service = remoteService.get(SomeService.class);

String result = service.doSomeStuff(1L, “secondParam”, new AnyParam());
```

Redisson Remote Service. Asynchronous calls

```
public interface Service {

    Long someMethod(Long param1, String param2);

    MyObject someMethod();

}


@RRemoteAsync(Service.class)

public interface ServiceAsync {

    RFuture<Long> someMethod(Long param1, String param2);

    RFuture<Void> someMethod();

}
```

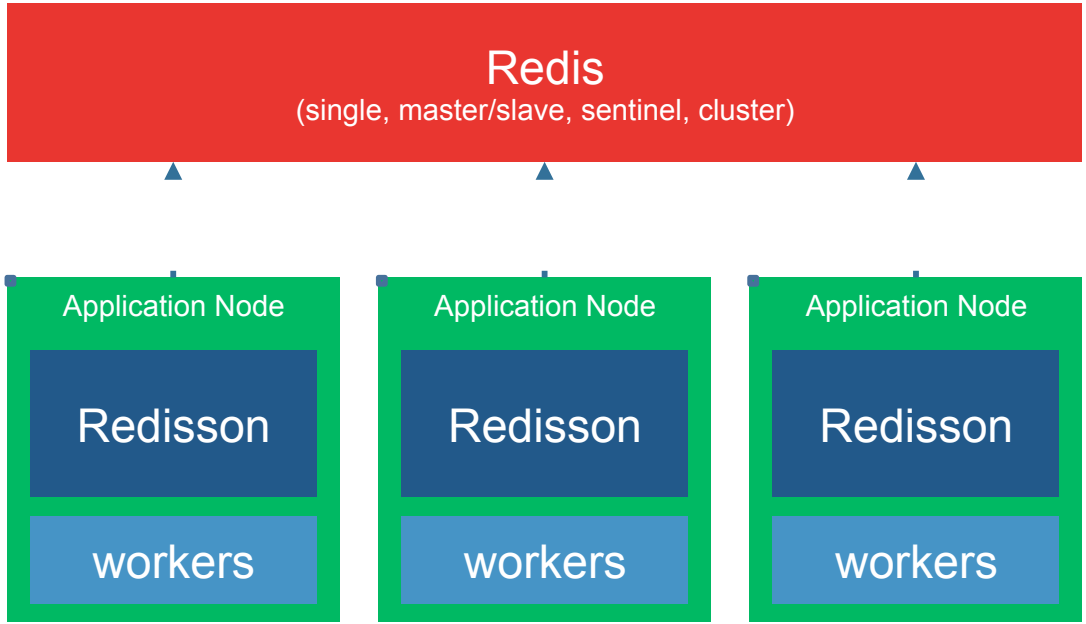
Tasks execution architecture



Tasks execution without Redisson Node

```
Redisson redisson = …

// custom executor workers

redisson.getExecutorService("myExecutor").registerWorkers(16);

// map-reduce workers

redisson.getExecutorService(MAPREDUCE_NAME).registerWorkers(16);
```

Workers on application side



Task definition / Callable

```
public class CallableTask implements Callable<MyResult> {
    @RInject
    private RedissonClient redissonClient;
    private String param;

    @Override
    public MyResult call() throws Exception {
        // task body
    }
}
```

Task definition / Runnable

```
public class RunnableTask implements Runnable {
    @RInject
    private RedissonClient redissonClient;

    @Override
    public void run() {
        // task body
    }

}
```

Submitting tasks for execution

```
// implements java.util.concurrent.ExecutorService
RExecutorService executorService = redisson.getExecutorService("myExecutor");


executorService.submit(new RunnableTask());
// or with parameters
executorService.submit(new RunnableTask(41, "someParam"));


executorService.submit(new CallableTask());
// or with parameters
executorService.submit(new CallableTask(53, "someParam"));
```

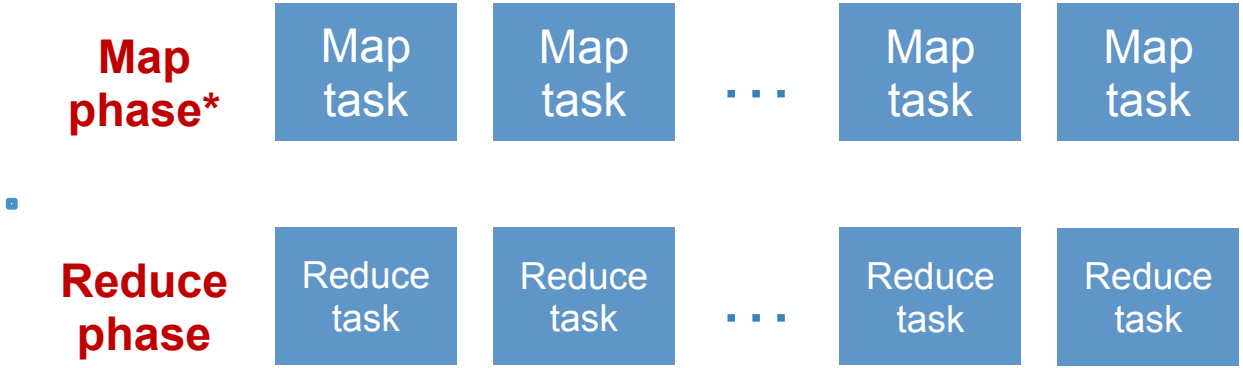
Submitting tasks for scheduled execution

```
// implements java.util.concurrent.ScheduledExecutorService
RScheduledExecutorService es = redisson.getExecutorService("myExecutor");


es.schedule(new CallableTask(), 10, TimeUnit.MINUTES);
// or
es.schedule(new RunnableTask(), 5, TimeUnit.SECONDS);


es.scheduleAtFixedRate(new RunnableTask(), 10, 25, TimeUnit.HOURS);
// or
es.scheduleWithFixedDelay(new RunnableTask(), 5, 10, TimeUnit.HOURS);
```

Scheduling tasks with cron expressions

```
RScheduledExecutorService es = redisson.getExecutorService("myExecutor");
// uses Quartz cron format
es.schedule(new RunnableTask(), CronSchedule.of("10 0/5 * * * ?"));
// or
es.schedule(new RunnableTask(),  CronSchedule.dailyAtHourAndMinute(10, 5));
```

Map Reduce overview



*splits to several tasks and run in parallel
only for RShardedSet and RShardedMap objects

Map Reduce

Data source example

```
RMap<String, String> map = redisson.getMap("wordsMap");
map.put(“page1:line1",    "Alice was beginning to get very tired");
map.put("page1:line2",    "of sitting by her sister on the bank and");
map.put("page1:line3",    "of having nothing to do once or twice she");
map.put("page1:line4",    "had peeked into the book her sister was reading");
map.put("page1:line5",    "but it had no pictures or conversations in it");
map.put("page1:line6",    "and what is the use of a book");
map.put("page1:line7",    "thought Alice without pictures or conversation");
```

Mapper definition

```
public class WordMapper implements
                                RMapper<String, String, String, Integer> {

    @Override
    public void map(String key, String value,
                                        RCollector<String, Integer> collector) {

        String[] words = value.split(" ");
        for (String word : words) {
            collector.emit(word, 1);
        }
    }

}
```

Reducer definition

```
public class WordReducer implements RReducer<String, Integer> {

    @Override
    public Integer reduce(String word, Iterator<Integer> iter) {
        int sum = 0;
        while (iter.hasNext()) {
            Integer i = (Integer) iter.next();
            sum += i;
        }
        return sum;
    }

}
```

Collator definition

```
public class WordCollator implements RCollator<String, Integer, Integer> {

    @Override
    public Integer collate(Map<String, Integer> result) {
        int totalWordsAmount = 0;
        for (Integer count : result.values()) {
      totalWordsAmount += count;
        }
        return totalWordsAmount;
    }

}
```

Running Map Reduce process

```
RMap<String, String> map = redisson.getMap("wordsMap");
RMapReduce mapReduce  = map.mapReduce()
                                    .mapper(new WordMapper())
                                    .reducer(new WordReducer())
                                    .timeout(60, TimeUnit.SECONDS);

// count occurrences of words
Map<String, Integer> wordToNumber = mapReduce.execute();

// count total words amount
Integer totalWordsAmount = mapReduce.execute(new WordCollator());
```

How to start

```
// 1. Create config object
Config = new Config();
config.useClusterServers()
        .addNodeAddress("myserver.com:7000", "myserver.com:7001");
// 2. Create Redisson instance
RedissonClient redisson = Redisson.create(config);
// 3. Get object you need
Map<String, String> map = redisson.getMap("myMap");
```

Connection modes

1. AWS Elasticache nodes
2. Azure cache nodes
3. Cluster nodes
4. Sentinel nodes
5. Master and Slave nodes
6. Single node

Data serialization

1. Jackson JSON
2. CBOR
3. MsgPack
4. Snappy
5. Kryo
6. FST
7. LZ4
8. JDK Serialization
9. Amazon Ion
10. Avro
11. Smile

Asynchronous command execution

```
RMapAsync<Integer, String> map = redisson.getMap("someMap");
Future<String> putIfFuture = map.putIfAbsentAsync(20, "object");
Future<String> getFuture = map.getAsync(20);


getFuture.addListener(new FutureListener<Boolean>() {
    @Override
    public void operationComplete(Future<Boolean> future)
        throws Exception {
        //...
    }
});
```

Reactive command execution

```
RedissonReactive redisson = Redisson.createReactive(config);

RMapReactive<Integer, String> map = redisson.getMap("someMap");
Publisher<String> putRes = map.put(20, "object");
Publisher<String> value = map.get(20);
```

Redisson vs Other IMDG or Cache Solution

	Redisson + Redis	Java based IMDG or cache solution (hazelcast, ehcache …)
Distributed objects and services	✓	✓
Large memory amount handling	✓	have issues
Fully-managed service support	✓	✗

Used by


EA


nab


NETFLIX


SAP


invaluable
The world's premier auctions and galleries


ticketmaster®


Bai du 百度


New Relic.
SYNTHETICS™


infor


Singtel


BROOKHAVEN
NATIONAL LABORATORY


Crimson Hexagon


datorama


optionshouse
by E*TRADE

Thanks

关注开源数据库论坛