東方通 | TongTech®

400-650-7088

TongRDS V2.2.1 安装配置手册

北京东方通科技股份有限公司

目录

1 安装环境要求

- TongRDS 为纯 java 应用，可完美支持国产软、硬件环境。
- TongRDS 运行需 java1.8 以上版本环境支持，OracleJDK 1.8、OpenJDK 1.8 或国产 JDK 1.8 环境均可，目前测试 OpenJDK 11 可正常运行。

2 安装及目录说明

2.1 RDS 服务节点安装

2.1.1 软件安装

程序发行版打包 TongRDS-2.2.x.x.Node.tar.gz 格式文件，在用户主目录下解包（linux 下 tar xvfz 命令）会创建 pmemdb 主目录，程序运行所需文件均在此目录下。

pmemdb/bin 下面是启动脚本 StartServer.sh 和 StopServer.sh 为 linux 下启停脚本（如果没有执行权限，可使用 chmod +x 命令赋执行权限），StartServer.bat 为 Windows 下的启动脚本（Windows 下停止程序关闭启动窗口即可）;SM4.sh(SM4.bat)为国密算法 SM4 的独立加解密实现;Client.sh（Client.sh）为客户端连接例程。

2.1.2 专用机（rpm/deb 包）安装

为实现专用机安装要求，提供 rpm/deb 打包方式。该包由专用机管理软件安装，安装成功后软件会解压安装到/opt/rds/pmemdb 目录下。同时将安装/user/lib/systemd/system/RdsNode.service 文件

2.2 中心节点安装

2.2.1软件安装

中心节点发行版打包 TongRDS-2.2.x.x.MC.tar.gz 格式文件，在用户主目录下解包（linux 下 tar xvfz 命令）会创建 pcenter 主目录，程序运行所需文件均在此目录下。

pcenter/bin 下面是启动脚本 StartCenter.sh 和 StopCenter.sh 为 linux 下启停脚本（如果没有执行权限，可使用 chmod +x 命令赋执行权限），StartCenter.bat 为 Windows 下的启动脚本（Windows 下停止程序关闭启动窗口即可）。

2.2.2License 安装

中心节点需要安装 License 授权文件才可启动运行。license 以名为“center.lic”文件的形式由厂家提供。pcenter 目录下的子目录有 bin、etc、lib、logs 等目录，程序的许可证文件“center.lic”放在 pcenter 主目录下。

2.2.3专用机（rpm/deb 包）安装

为实现专用机安装要求，提供 rpm/deb 打包方式。该包由专用机管理软件安装，安装成功后软件会解压安装到/opt/rds/pcenter 目录下。同时将安装/user/lib/systemd/system/RdsCenter.service 文件

2.3 目录说明

2.3.1bin

bin 目录内是启动、停止服务程序的 sh 脚本，包含 StartServer.sh、

StopServer.sh、StartCenter.sh、StopCenter.sh 等。

2.3.1.1 服务节点 bin 文件说明

文件名	用途
Client.bat/ Client.sh	Client 应用启动脚本
external.vmoptions	Linux 下 jvm 参数配置文件
InstallService.bat	Windows 服务安装脚本
UninstallService.bat	Windows 服务卸载脚本
JavaService.exe	Windows 服务加载程序
RdsNode.service	Linux 服务配置脚本
RestartSentinel.sh	独立哨兵服务重启脚本
sentinel.sh	独立哨兵程序加载脚本
StartSentinel.bat	独立哨兵服务启动脚本
StartSentinel.sh	独立哨兵服务启动脚本
StopSentinel.sh	独立哨兵服务停止脚本
RestartServer.sh	服务节点重启脚本
starter.sh	服务节点加载脚本
StartServer.bat	服务节点启动脚本
StartServer.sh	服务节点启动脚本
StopServer.bat	服务节点停止脚本
StopServer.sh	服务节点停止脚本
SM3.bat/ SM3.sh	SM3 演算脚本
SM4.sh/ SM4.bat	SM4 演算脚本
Version.bat/ Version.sh	版本、许可证查询

2.3.1.2 中心节点 bin 文件说明

文件名	用途
InstallService.bat	Windows 服务安装脚本
UninstallService.bat	Windows 服务卸载脚本
JavaService.exe	Windows 服务加载程序
external.vmoptions	Linux 下 jvm 参数配置文件
RdsCenter.service	Linux 服务配置脚本
RestartCenter.sh	Center 重启脚本
StartCenter.bat	Center 启动脚本
StartCenter.sh	Center 启动脚本
StopCenter.bat	Center 停止脚本
StopCenter.sh	Center 停止脚本
SM3.bat/ SM3.sh	SM3 演算脚本
Version.bat/ Version.sh	版本、许可证查询

2.3.2etc

etc 目录保存配置文件，RDS 服务节点有两个文件：cfg.xml 文件是静态参数配置，包括内存表的配置、监听端口等；dynamic.xml 文件为动态参数配置，主要配置复制节点地址、中心管理节点地址等。

中心节点为 config.properties 、 active.properties 、sync.properties、acl.properties 四个文件。其中：config.properties 是静态配置文件，包括端口号等，修改后需要重启生效；active.properties 为 ActiveCode 授权码文件，文件内容修改后立即生效；sync.properties 为各 center 节点的同步配置，动态生效。acl.properties 为企业版新增分用户角色权限控制，动态生效。

2.3.3lib

lib 目录是程序运行库文件。包含 core.2.2.x.x.jar（服务节点）、cener.2.2.x.x.jar（中心节点）、第三方支持库（如：netty 框架库）等等。

2.3.4logs

logs 目录保存程序运行日志。

2.3.5var

var 目录保存系统的 redo 日志和持久化的数据文件。目前系统支持同步列表的 redo 日志。

每个表保存 1 套 redo 日志，每套同步列表的 redo 日志共 6 个文件，其中 tb{n}_redo.cfg 文件中保存当前 redo 日志文件的编号；tb{n}_redo.rd1—tb{n}_redo.rd5 为 redo 日志文件。

3 系统启、停

3.1 启动 RDS 服务节点

启动服务使用 pmemdb/bin 目录下的 StartServer.sh 脚本，在 bin 目录下执行此脚本。

如果需要以后台服务形式启动，可执行如下在 bin 目录下执行：

```
nohup ./StartServer.sh > /dev/null 2>&1 &
```

前置条件：java1.8 版本可用。

3.2 停止 RDS 服务节点

停止服务请在 pmemdb/bin 目录下执行 StopServer.sh 脚本。

3.3 启动中心节点

启动服务使用 pcenter/bin 目录下的 StartCenter.sh 脚本，在 bin 目录下执行此脚本。

如果需要以后台服务形式启动，可执行如下在 bin 目录下执行：

```
nohup ./StartCenter.sh > /dev/null 2>&1 &
```

前置条件：java1.8 版本可用。

3.4 停止中心节点

停止服务请在 pcenter/bin 目录下执行 StopCenter.sh 脚本。

4 RDS 服务节点配置

系统启动时通过读取${MEMDB_PATH}/etc/cfg.xml 文件获得配置信息。配制内容举例如附录 2，说明如下。

4.1 通用参数

Server.Common.RuntimeModel：程序的运行状态。该项配置为“debug”时，相关的统计日志会更快的输出。

Server.Common.Service：当前服务节点所属的服务组名。

Server.Common.StartWaiting：主线程启动时等待其他节点同步数据的时

间，单位秒，缺省为 10 秒。

Server.Common.DataDump：自动持久化数据到磁盘的时间间隔，单位是分钟。最小值是 1 分钟，配置为 0（或缺省时），不开启自动持久化功能。

Server.Common.DataDumpAppending：针对 Redis 仿真协议开启命令增量记录功能（类似 Redis 的 AOF），缺省为不开启。设置为“true”开启增量文件功能，会将写命令（只记录写命令）记录到增量文件 var/append-<timestamp>.aof 中，该配置生效的前提是开启自动持久化功能（Server.Common.DataDump 配置大于 0）。程序每次执行自动持久化后会删除过期的增量文件（每次执行 save 全量保存时会删除过期的增量文件，否则增量文件会持续增加，所以必须开启自动持久化功能才允许增量保存）。

Server.Common.TotalVariableAllowedSize：长度超过 value 限制的数据占用的总内存量。

Server.Common.MaxValueLength：长度超过 value 限制的数据的单条最大长度。

Server.Common.MaxItemsInCompress：复杂类型（如：list、set、hash、zset 等）以压缩模式存储的项目的最大数量，缺省是 256，超过此数量则采用 BigValue 对象方式存储。

Server.Common.DynamicValueCompress：对于采用动态存储方式的大 value 值，是否压缩存储，压缩存储时数据以字节数组形式保存，否则采用 java 的字符串形式保存（字符串 1 个字符会占用 2 个字节），缺省为 true。该配置从版本 2.2.1.2_P3 开始废弃不用，数据均采用压缩模式存储。虽然压缩模式会使用稍多的 CPU 但会大大减少内存占用，该配置已无实用价值。

Server.Common.SlowOperationThreshold：慢响应的阈值，执行时间高于该值的命令会在日志文件中输出类似“Time consuming {} ms for {}”的警告日志。

Server.Common.BinaryCompatible：定义 Redis 仿真端口是否支持 2 进制的 value 数据，缺省为“false”（该配置从 2.2.1.2_P3 版本开始缺省改为“true”）。由于改进了兼容算法，RDS 自有协议（缺省为 6200 端口）从 P3 版本开始支持此

配置。

Server.Common.DefaultRedisPort：当配置了集群时，该配置定义各集群节点的缺省 redis 端口。创建集群时，各节点会尝试通信，如果通信成功则会读取实际的 redis 端口，在未连通之前会采用此配置作为缺省端口。该配置的缺省值时 6379。

Server.Common.JmxUrl：指定 RDS 业务管理功能的 jmx 地址。缺省时 RDS 不启动 jmx 服务，配置此项后 RDS 会启动 jmxrmi 服务对外提供核心数据监控接口。该配置可以只配主机地址和端口号，格式为“ip:port”，程序会自动补全为“service:jmx:rmi:///jndi/rmi://ip:port/jmxrmi”；也可以直接配置完整的 url（完整的 url 必须以“service:jmx”开头。

Server.Common.DataReadOnly：当前节点数据是否为只读状态。当该项配置为 true 时，总是不允许执行写操作的命令（无论节点角色是主节点还是从节点）；配置为 false 时，根据角色不同，当角色为主节点是允许写操作，角色为从节点时不允许写操作；缺省时（未配置此项时），总是允许写操作命令（无论节点角色是主节点还是从节点）。

Server.Common.MaxItemsReturned：限制 lrange、smembers、hgetall 等命令的返回最大数量，超过此值返回错误信息，0（或小于 0）为不限制，缺省限制为 10000 条。

Server.Common.Statistics：开启统计功能，缺省为“false”不开启，主要影响查询命中率等统计信息。

Server.Common.FullGcInterval：程序运行期间每过多长时间（单位：秒）强制做 1 次 FullGC（调用 System.gc()方法），分辨率为 10 秒，缺省（或配置为 0）不做强制 GC。

Server.Common.AntiRedis：去掉程序返回信息里面的 redis 标记，目前配置该项可屏蔽 info 命令中的 redis_version 字段，缺省为‘false’。

Server.Common.RedisVersion：设置 info 命令中返回的 redis_version 的内容，在 Server.Common.AntiRedis 参数未配置时有效，缺省为“5.2.13”。

Server.Common.DangerousCommandFilter：屏蔽 redis 危险命令，设置该项为‘true’时，危险命令禁止执行（目前屏蔽 flushdb、flushall 命令），缺省为‘false’。

Server.Common.NeedRescue:程序启动时是否尝试恢复数据，如果为真或未配置，则程序启动时首先尝试读取 var 目录下的文件数据，然后再根据 dynamic.xml 文件中的同步配置尝试从其他程序更新数据。缺省为“true”。

Server.Common.SslCiphers：当启用 SSL 连接时，指定密钥交换使用的算法。早期版本的 SSL 存在“SSL/TLS 服务器瞬时 Diffie-Hellman 公共密钥过弱”漏洞。可通过此配置禁止使用 Diffie-Hellman 算法。建议配置：

TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384, TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA,
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA,
TLS_KRB5_WITH_DES_CBC_MD5, TLS_KRB5_WITH_3DES_EDE_CBC_SHA,
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA, TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA,
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,
TLS_RSA_WITH_AES_128_CBC_SHA

Server.Common.DataStoragePath：指定存盘文件和日志输出的主目录。程序运行时会在该目录下创建 var 和 logs 目录，缺省时与 Rds 主目录一致，可配置指定将数据存放到其它目录。如果该配置以字符“/”开头则认为时绝对路径配置，否则认为是从 Rds 主目录起始的相对路径。**注意：如果指定其它路径请确保 Rds 对该目录有读写权限同时不能危及到操作系统安全。**

Server.Common.MaxKeyLength：当 table 表中配置的 key 的类型为 bytes2（2.2.1.2_P3 新增）时，该配置限制 key 可能的最大长度。

Server.Common.MasterDeterminate：主节点切换策略配置。缺省时程序按照 dynamic.xml 中 Synchronize 配置的顺序检测首个健康节点为主节点；当该配置为“config”时将 Synchronize 配置中的第一项作为主节点（不检测此节点是否健康）。使用该配置可阻止备份节点在检测到主节点异常时自主切换为主。

4.2 端口监听参数

Server.Listen.Port：程序主服务启动后监听的端口号。

Server.Listen.Threads：程序为接收数据创建的线程池的大小。因程序采用异步 IO 处理数据，所以线程数不再对应打开的连接个数，该线程池实事上是负责程序业务处理的主线程，建议对于独占主机的应用可以配置为 CPU 核心数的 2 到 3 倍，最多不要超过 32。

Server.Listen.MaxConnections：程序允许同时连接的最大客户端数量，缺省值为 1000，合法的最小值为 10，最大值目前设置是 1 千万。

Server.Listen.Backlog:出现大量并发连接请求时程序允许堆积的 lianjie 请求的上限，缺省值为 256。

Server.Listen.IoProcessor：（已废弃）网络访问处理器类型，可选项为 traditional 和 concurrent。Traditional 为传统处理方式，兼容性更好；concurrent 为多 cpu 高并发模式,对于现代主机性能更好。TongRDS 版本 2.2.1.2 以后缺省为 concurrent（版本 2.2.1.1 及以前版本缺省为 traditional）。TongRDS 版本 2.2.1.2_P1 以后废弃此配置。

Server.Listen.Secure：监听端口的安全级别，0：telnet 连接，无安全通道；1：ssl 加密连接，用 ssl 连接；2： password 密码认证连接；3：ssl+password 加密连接+密码认证，连接时采用 ssl 连接，然后发送密码，密码通过后才是有效连接。密码通过 Server.Listen.Password 项配置。

Server.Listen.Password：连接主服务的密码。如果配置项 Server.Listen.Secure=2 或 3，则创建连接后客户端会首先发送该密码进行认证。

注意：系统内的所有服务程序的配置文件中的配置项 Server.Listen.Secure 和 Server.Listen.Password 必须一致，否则数据同步将因为无法建立正确的连接而不能成功。

Server.Listen.IdleTimeout：Redis 仿真端口连接空闲超时时间。当端口在此配置时间内没有读写操作则会关闭连接。

4.3 日志参数

Server.Log.File：系统输出的日志文件名，缺省为“memdb.log”。文件目录为$MEMDB_HOME/logs/。

Server.Log.Level：系统日志级别，目前支持 debug、info、warn 等配置。

Server.Log.BackDates：日志保存天数，超过时间的日志会被删除，缺省为 0（不删除），1 为只保留当天日志，以此类推。

Server.Rescure.NeedRescure：程序启动时是否从其他服务器恢复数据，缺省为“true”，恢复数据。程序恢复数据时会按照本机配置文件中的同步服务器列表依次尝试连接，并找到启动时间最早的服务器尝试恢复；如果恢复过程未正常完成（通信异常等）程序会重复上述过程（最多 3 次）；如果程序启动时未找到可恢复数据的服务器（如同步服务器列表中的 server 都连接不上），则放弃恢复数据操作。当该配置配成“false”时，程序不恢复数据直接完成启动。

注：日志文件按天切换，单个不大于 1g，超过会自动切分。数据文件大小大约为总数据量（key+value）的 130%。

4.4 事件监听参数

Server.Notify 段配置用于事件监听功能和消息通知配置。

Server.Notify.SyncListNumber：同步 publish 命令发布的消息（到其他节点）的缓存列表数量。缺省为 1，配置为 0 关闭同步功能（同步消息功能只在集群模式下有效，非集群模式下配置为 0 可提高性能）。

Server.Notify.SyncListLength：同步 publish 命令发布的消息（到其他节点）的缓存列表长度。缺省为 1000。

Server.Notify.Event：指定需要监听的事件类型，合法配置有 keyspace 和 keyevent，其中：keyspace 对应 __keyspace@<db>__ 开头的事件；keyevent 对应 __keyevent@<db>__ 开头的事件。这 2 个至少需要配置一个，否则不能开启事件监听功能。

Server.Notify.Message：指定需要监听的事件内容，合法配置有：expired（key 过期触发事件）、evicted（由于数据过多造成 key 被驱逐时触发此时间）、generic（key 操作命令，如 del、expire 等）、string（String 类型数据操作命令，如 set 等）、list（List 类型数据操作命令，如 lpush、rpush 等）、set（Set 类型数据操作命令，如 sadd 等）、hash（HashMap 类型数据操作命令，如 hadd 等）、zset（Sorted 数据类型操作命令，如 zadd 等）、cmds（全部操作命令事件，等效于同时设置 generic, string, list, set, hash, zset）、all（全部事件，等效于同时设置 expired, evicted, generic, string, list, set, hash, zset）。

4.5 防火墙参数

Server.Firewall.AuthFailedTimes：同一个 client 端密码验证允许连续失败的次数，超过此次数会断开连接并在 5 秒内阻止该 client 端的新建连接。

Server.Firewall.Whitelist：白名单地址列表，只有列表内的 client 端才允许接入，多项地址由逗号（,）分隔。注意：黑白名单只能设置 1 个，其中白名单的优先级高于黑名单，如果同时设置了黑、白名单，白名单生效。

Server.Firewall.Blacklist：黑名单地址列表，黑名单内的 client 端禁止接入，多项地址由逗号（,）分隔。注意：黑白名单只能设置 1 个，其中白名单的优先级高于黑名单，如果同时设置了黑、白名单，白名单生效。

4.6 Redis 适配参数

Server.Listen.RedisPort：程序提供 Redis 兼容协议的服务端口。

Server.Listen.RedisPlainPassword：登入 Redis 服务使用的密码在配置文件中的保存方式，该项为‘true’时为明文保存，缺省需要加密保存（该配置是否生效需要参考 Server.Listen.Secure 的配置，该配置为 2 时需要密码认证）。

Server.Listen.RedisPassword：登入 Redis 服务使用的密码，依据上述配

置，可以是明文或密文（该配置是否生效需要参考 Server.Listen.Secure 的配置，该配置为 2 或 3 时需要密码认证）。

注：密码中如果有 xml 文件中定义的特殊字符需要进行转义。例如：密码是“123&456”，配置文件中应配成“123&456”，需转义的字符如下表：

原始字符	转移后
"	"
'	'
<	<
>	>
&	&

4.7 数据表配置

4.7.1表参数

Server.Tables：定义系统配置的内存表的数量。

Server.Table{n}.Rows：表 n 的缓存 list 的总长度。基于效率和多线程并发访问等因素的考虑，程序创建 m 个队列，每个队列的长度是（Rows/m+1），每个队列有自己的内存锁，数据通过对 key 做 hash 分到不同的队列里。优点是并发和效率，缺点是由于数据是通过 hash 分到不同的队列了，所以每个队列的数据分配会不均，会使程序能保存的数据总数达不到 Rows 的配置。另外一个问题是由于实现机制问题，程序对 Server.CacheLength 有最大值限制。

Server. Table{n}.DataExpireTime：数据过期时间，单位‘秒’。程序从写

入数据开始计时。过期后自动清理。该参数缺省值是 0，数据不过期。

Server. Table{n}.UpdateModel：程序对已有数据是否自动更新。缺省为自动更新方式，即当收到的数据的 key 在系统内已经存在时，程序会首先删除旧数据，然后再增加新数据。如果配成‘insert’，即不自动更新，则当输入的 key 已存在时，拒绝更新并报 27 号错误更新失败。

Server. Table{n}.AutoOverlay：当内存表已满时，程序是否用收到的新数据覆盖掉系统内较旧的数据（程序的算法从列表尾开始覆盖，但不能保证被覆盖的是最旧的数据）。缺省为自动覆盖，当配置为“false”时不覆盖旧数据并报 28 号错误插入数据失败（更新已有数据不受此功能影响）。

4.7.2字段参数

Server. Table{n}.Key：定义 key 字段的类型和长度。程序可理解的类型有 bytes、bytes2(针对 key 的可变长度类型)、integer、ipv4、long、mac、biginteger、ipv6 等几种。其中 bytes 类型、bytes2 类型和 biginteger 类型需要定义长度。bytes 的长度单位是 8bit 的 byte，缺省为 32 字节；bytes2 的长度定义和 bytes 相同，区别在于当输入的 key 超过配置的长度时 bytes 类型会报错，而 bytes2 类型会申请动态空间存储；biginteger 的长度单位是 32bit 的整数，缺省为 4 个整数。ipv4 类型的字符串表示格式为： XXX.XXX.XXX.XXX，其中 X 为十进制数 mac 类型表示为：XX:XX:XX:XX:XX:XX，其中 X 为 16 进制数，double 类型为双倍长整形数，表示格式为：XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX，其中 X 为定长的 32 个 16 进制数，ipv6 类型为双倍长整形数的子类，表示格式为：XXXX:XXXX:XXXX:XXXX:XXXX:XXXX:XXXX:XXXX，其中 X 为 16 进制数。例如：

<Key>bytes, 128</Key>

Server. Table{n}.Value：定义 value 字段的类型和长度。定义描述同 Server.Key。例如：

<Value>bytes, 512</Value>

Server. Table{n}.Indexes：定义非唯一索引的最大配置数，缺省为 0。

Server. Table{n}.Index<1…n>：定义第 n 个非唯一索引的类型、长度和索引算法。Index 的定义与 Server.Key 相比，多了第三项索引算法的定义，索引算法支持 3 种：none：无索引；list：链表索引，适合少量重复数据的索引；tree：二叉树索引，适合有大量重复数据的索引。缺省的索引算法为二叉树（tree）索引。Index 类型和长度的定义描述同 Server.Key。例如：

<Index1>mac</Index1>

<Index2>ipv6,, tree</Index2>

<Index3>bytes, 512, list</Index3>

<Index4>biginteger, 4</Index4>

4.7.3数据同步

由于程序采用内存方式保存数据，无法持久。为了解决单点，采用多 s e r v e r 并行工作，数据相互同步的工作方式。程序同步数据时依靠数据的时间戳判断新、旧，要求各服务器的时钟同步。相关配置如下：

Server. Table{n}.Sync.ListNumbers：同步队列数量。为了同步操作不影响主业务，采用同步队列方式由后台线程实现（严格的讲应该叫准实时同步）。该配置定义同步队列的个数，需要同步的数据 hash 到不同的队列。更多的队列配置可以增加同步速度但会消耗更多的线程资源。

Server.Table{n}.Sync.ListLength：每个同步队列的长度，较长的队列可靠性高但会消耗更多的内存资源。

Server.Table{n}.Sync.Mode：程序是否采用可靠方式同步数据，有效配置为 must 和 recode。数据同步采用列表缓冲的准实时方式实现，缺省情况下如果同步数据的对端服务异常导致同步列表写满时会丢弃旧数据，此模式下不影响本机写操作的继续执行，但对端服务恢复后因为有丢弃的同步数据，可能造成部分数据的不一致。当配置成 must 模式时，如果同步列表写满后程序会拒绝随后的写/更新操作（但不影响读操作），只有当有数据同步完成后才会允许继续更新数据；当配置成 recode 时，当同步列表写满后需要丢弃旧数据时，会将丢弃的数

据记录到 var/synclost.dat.XXX 文件中。

Server.Table{n}.Sync.NeedRedo：同步列表是否写 redo 日志。如果该项配置为“true”则所有写入同步列表的数据会同时写到 var 目录的 redo 文件中，在程序重启时会按照原来的列表顺序恢复同步数据。Redo 文件有 1 个控制文件和 5 个日志文件组成：控制文件记录当前日志文件的序号；5 个日志文件被循环使用，其总记录条数等于列表长度。日志文件每 1 秒钟或每 16K 做 1 次磁盘写操作，理论上 1 秒以内的数据有可能会丢失，但丢失数据总量不会超过 8K。

Server. Table{n}.Sync.Warning：同步列表的警告线，当同步列表中的数据占总表长度的百分比超过该配置时，在日志中会有 warn 级别的告警

可配置多个同步节点。

4.8 中心服务器配置

中心管理节点提供集中管理功能，包括 license 集中管理、各服务节点状态监控等功能。同步服务器配置在 dynamic.xml 文件中，可在程序运行时动态修改，动态刷新。

Server.Center.Server.host：中心节点主机的地址

Server.Center.Server.port：中心节点的服务端口号，对应中心节点的 service.port 配置。

可配置多个中心节点。

4.9 集群动态变更与数据迁移配置

集群配置配置支持动态调整（扩容和缩容），可通过修改中心节点的集群配置动态增加或减少节点，也可以根据节点内存使用情况动态调整各节点占用的 slot 数量。

集群配置调整后客户端将按照新的 slot 分配规则访问各节点数据，调整前的数据由于 slot 分配的变更有可能访问不到，此时可利用 Server.Clusters.MigrateData 等于 true 手工触发一次数据迁移(将集群内原有数据按照新的 slot

配置重新分配（各节点均需要配置）。配置举例如下：

```
<Clusters>
    <MigrateData>true</MigrateData>

    ......
</Clusters>
```

程序发现该配置后会尝试迁移数据，迁移完成后会从配置文件中删除此配置。

5 中心节点配置

中心节点是缓存中间提供的集中管理模块，负责各服务节点以及接入的客户端的集中监控管理。

5.1 核心配置

5.1.1配置说明

中心节点相关配置说明如下：

server.log.file：日志名，日志会存放到主目录的 logs 子目录下。

server.log.level:日志级别。可选级别有:nothing, error, warning, info, debug, dump，依次越来越详细。

server.log.backdates：日志保存天数，超过时间的日志会被删除，缺省为 0（不删除），1 为只保留当天日志，依此类推。

server.service.port：主服务提供端口，用于各节点上传数据和客户端接入缓存集群（主服务不支持非 SSL 方式接入，因此没有 SSL 相关配置）。

server.service.max_connection：主服务端口允许的最大连接数，可配置的最小值为 1，缺省 1000，支持动态更新。

server.sentinel.port：Redis 哨兵仿真端口，模拟 Redis 哨兵的功能，提供客户端查询各服务主节点、订阅主节点变更消息等。

server.sentinel.ssl：Redis 哨兵仿真端口是否启用 SSL 安全连接，true 为启用安全连接。

server.sentinel.max_connection：哨兵仿真端口允许的最大连接数，可配置的最小值为 1，缺省 1000，支持动态更新。

server.host.accessible_address： Center 所在主机可被访问的 IP 地址，此配置将替换可能返回给客户端的“127.0.0.1”地址。

server.platform:指定平台类型,缺省为“native”,容器云环境配置为“k8s”

server.platform_service：容器云环境下 center 服务（必须是 headless 类型）名。

server.raft.async_write：内部程序在写入集群数据时是否采用异步方式，缺省为 true。如果配成 false 每次写数据需要等大多数节点写成功（或超时错误）才会返回。程序设计上集群数据具备自恢复功能，不需要等待写集群成功，建议该项使用缺省配置。

server.raft.easy_group：Center 各节点通过 Raft 协议组同步数据时，是否要求 Raft 以宽松的方式判断多节点规则，缺省为“false”。缺省时 Raft 将严格按照 sync.properties 中配置的 Center 节点作为总的节点数，只有超过半数节点活跃 Raft 组才有效，Raft 组有效时 Center 节点才可对外提供服务，如果达到半数的节点宕机则整个 Reaft 组进入 crash 模式，全部 Center 节点中止服务。但是，对于某些特殊应用场景（比如：两中心异地容灾），某些长江下无法满足多数节点活跃，则开启此配置。开启此配置后 Raft 将以活跃节点数代替配置中的总节点数，保证容灾场景出现时 Center 扔可正常提供服务。

server.service.node.expire_time：集群中服务节点失联下线的超时时间，缺省为 2000 毫秒。如果该时间内服务节点数据未被更新则认为该节点已经下线。

server.service.node.livings_first： 各服务节点间的同步配置项（dynamic.xml 中的 sync 段）的顺序是否遵循活跃节点优先原则（根据活跃节点和活跃的先后顺序调整同步列表顺序），缺省是配置优先原则（即按照配置的顺序）。

server.admin.port：Restful 协议管理控制台的监听端口，该接口提供标准

Restful 形式的对外接口，可查询 license 使用情况、各服务节点运行情况、客户端接入情况等信息。

server.admin.protocol：Restful 协议管理控制台端口采用的协议，缺省为 http，配置为 https 时将采用安全连接（系统自带证书）。

server.admin.auth：登录控制台端口认证方式，缺省不认证，目前支持 token 方式认证，该项配置为“token”激活。

server.admin.auth.token.name：http 请求头中 token 使用的属性名。

server.admin.auth.token.content：http 请求头中 token 的内容。

server.admin.auth.token.validity_period：http 请求头中当前 token 的有效时间，过期失效。缺省为 30 分钟。

server.ssl.ciphers：当启用 SSL 连接时，指定密钥交换使用的算法。早期版本的 SSL 存在“SSL/TLS 服务器瞬时 Diffie-Hellman 公共密钥过弱”漏洞。可通过此配置禁止使用 Diffie-Hellman 算法。建议配置：

TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,

TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384,

TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA,

TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA,

TLS_KRB5_WITH_DES_CBC_MD5, TLS_KRB5_WITH_3DES_EDE_CBC_SHA,

TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA,

TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA,

TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,

TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_128_CBC_SHA

server.storage.path：指定存盘文件和日志输出的主目录。启动时程序从该目录下读取 center.lic 文件，且运行时会在该目录下创建 var 和 logs 目录，缺省时与 pcenter 主目录一致，可配置指定将数据存放到其它目录。如果该配置以字符“/”开头则认为时绝对路径配置，否则认为是从 pcenter 主目录起始的相对路径。

注意：如果指定其它路径请确保软件对该目录有读写权限同时不能危及到操

作系统安全。

server.sentinel.idle_timeout：哨兵仿真端口连接空闲超时时间。当端口在此配置时间内没有读写操作则会关闭连接。

5.1.2配置举例

配置文件举例如下：

```
server.log.file = center.log
# nothing, error, warning, info, debug, dump.
server.log.level = debug
server.log.backdates = 7
service.port=6300
sentinel.port=26379
```

5.2 同步配置

5.2.1配置说明

同步配置用于配置其它的中心节点的服务地址，实现各中心节点间的数据同步，依靠多个节点实现高可用。

中心节点的数据同步除了同步 client 的连接信息、服务节点的信息以外，还负责维护中心节点群落间的配置一致性（采用自有实现的 raft 算法保证群落内配置的一致性）。要实现上述功能，需在每个服务节点的 sync.properties 文件中把全部中心节点都配进去。

配置说明如下：

sync.servers：定义中心节点的总数量（编号从 1 开始）

sync.server1.host：第一台需要同步数据的中心节点的地址

sync.server1.port：第一台需要同步数据的中心节点的端口

5.2.2配置举例

配置举例如下：

```
sync.servers=2
sync.server1.host=localhost
sync.server1.port=6301
sync.server2.host=localhost
sync.server2.port=6302
```

5.3 授权码配置

5.3.1配置说明

授权码（AcviteCode）用于控制客户端能访问 RDS 集群的权限。配置方式为 AcviteCode = 以逗号分隔的服务列表。Client 使用 ActiveCode 作为密码接入中心节点（中心节点类似于 Redis 的哨兵）。

授权码是 properties 文件的 key 可接受的任意字符串，长度小于 128 字节。

“=“后面是以逗号分隔的服务名列表，服务名对应 RDS 服务节点中的 Server.Common.Service 配置。”*“为通配符，如果 AcviteCode = *，则用此 AcviteCode 接入的 client 有访问所有服务的权限。

5.3.2配置举例

ActiveCode 配置举例如下：

```
default = *
acioweor_483kja03np4h8238G = WebSession, AuthService,
```

5.4 部署管理配置

配置文件 cluster.properties 用于配置中心节点管理范围内的 RDS 服务节点的工作模式（部署模式）。可通过配置设置被管理的服务内的 RDS 节点处于哨兵模式或处于集群模式。

5.4.1配置说明

配置项列表如下：

配置项	说明
{服务名}. type	指定服务的多节点部署模式，有效配置为：“sentinel”（哨兵模式）；“cluster”（集群模式）。
{服务名}.nodes	哨兵模式时配置的节点总数
{服务名}.node[0-n]	各节点的地址端口配置（n 不能大于 nodes 配置的值），格式为：IP:PORT
{服务名}. shards	集群模式时的分组（切片）数
{服务名}.shard[0-n].nodes	各分组内包含的 RDS 服务节点信息，格式为：IP(ALIAS):PORT，多项由逗号分隔。括号内是 ip 地址的别名，可选项，如果有配置别名在节点端执行 cluster nodes 等命令时，节点地址返回的是别名，否则返回节点的 IP 地址。
{服务名}.shard[0-n].slots	各分组内包含的数据的切片值。合法的切片值从 0 到 16383 之间。切片值配置支持指定值和区间范围配置，区间配置格式为：start-stop，多项由逗号分隔。

5.4.2配置举例

cluster.properties 配置文件举例：

```
WebService.type=sentinel
WebService.secure=0
```

```
WebService.password=454d51192b1704c60e19734ce6b38203
WebService.nodes=3
WebService.node0=192.168.0.60:6200
WebService.node1=192.168.0.61:6200
WebService.node2=192.168.0.60:6202

OnlineData.type=cluster
OnlineData.secure=0
OnlineData.password=454d51192b1704c60e19734ce6b38203
OnlineData.shards=3
OnlineData.shard0.nodes=192.168.0.60:6200,192.168.0.60:6200
OnlineData.shard0.slots=0-4999
OnlineData.shard1.nodes=localhost:6200,192.168.0.60:6201
OnlineData.shard1.slots=5000-11000
OnlineData.shard2.nodes=192.168.0.60:6202,192.168.0.60:6202
OnlineData.shard2.slots=11001-16383
```

5.5 用户访问控制（ACL）

用户访问控制通过设置访问控制列表（ACL），对接入用户通过用户名、密码进行验证，并对用户可执行的命令以及可访问的数据进行授权以达到对用户命令权限和数据权限的控制。

用户访问权限配置在中心节点中的 acl.properties 文件配置，中心节点读取配置并下发到服务节点实现用户接入控制。整个配置包括：角色配置和用户配置 2 大部分。

5.5.1 角色配置

角色配置用于配置不同角色的访问权限。配置项为：

acl.roles：acl 列表中角色的下标的最大值，程序从 0 开始循环读取各角色。

acl.role1.cmd_list：配置角色 1 可访问的命令列表。配置可具体指定命令，或设置命令组。合法配置举例："acl.role1.cmd_list=@all"，角色 1 可使用全部命令；"acl.role2.cmd_list=@read，@sortedset -@list"，角色 2 可使用全部只读命令、

全部 zset 命令，且不能使用全部 list 命令；“acl.role3.cmd_list=get，set，sadd，-set”角色 3 可使用 get 和 sadd 命令（授权执行过程为：+get，+set，+sadd，-set）。

acl.role1.key_prefix：配置角色 1 可访问的数据 key 的前缀。例如：“acl.role3.key_prefix=aa”，角色 1 可访问以 aa 开头的 key 对应的数据。

5.5.2用户配置

用户配置用于指定每个服务可访问的用户和用户所属角色（一个用户可配置多个角色，权限是叠加关系）。配置项如下：

WebSession.users：配置 WebSession 服务（对应各服务节点中的 Server.Common.Service 配置）中的用户总数量。

WebSession.user1.username：WebSession 服务中用户 1 的用户名

WebSession.user1.password：WebSession 服务中用户 1 的密码，密码采用 SM3 算法加密。

WebSession.user1.role_id：WebSession 服务中用户 1 的角色，可以配置多项，各项用“,”分隔。

WebSession.user1.active：WebSession 服务中用户 1 当前是否可用，active 为 true 时用户可用，为 false 时禁止登陆。

5.5.3配置举例

Acl.properties 配置举例如下：

```
acl.roles=5
acl.role1.cmd_list=@all,
acl.role2.cmd_list=@read, @sortedset -@write
acl.role3.cmd_list=get, set, sadd, -set, ,
acl.role3.key_prefix=aa

#WebSession.users=5
#WebSession.user1.username=user1
#// SM3 加 密 ,
“6E0F9E14344C5406A0CF5A3B4DFB665F87F4A771A31F7EDBB5C72874A32B2957”的明文为
```

```
"123"
    #WebSession.user1.password=6E0F9E14344C5406A0CF5A3B4DFB665F87F4A771A31F7E
DBB5C72874A32B2957
    #WebSession.user1.role_id=1，2
    #WebSession.user1.active=true
    #WebSession.user2.username=user2
    #WebSession.user2.password=6E0F9E14344C5406A0CF5A3B4DFB665F87F4A771A31F7E
DBB5C72874A32B2957
    #WebSession.user2.role_id=3
```

6 独立哨兵服务配置样例

RDS 提供独立的哨兵组件。独立的哨兵组件包含在服务节点包中随随服务节点发行。相比中心节点附带的哨兵功能，该模块部署更灵活，客户端适应性更强（例如：中心节点的哨兵功能要求登录必须使用密码，但有的早期客户端不支持带密码的哨兵，独立的哨兵组件允许无密码接入）。

使用该组件可帮助用户更灵活的解决 redis 时需要哨兵的问题，独立模块哨兵有如下特点：

- ✧ 独立于 RDS-Node 节点运行
- ✧ 运行时无需 license 授权
- ✧ 根据配置主动检测服务节点状态
- ✧ 通过配置，一个哨兵可同时监控多个 RDS 服务组
- ✧ 可同时运行多个哨兵程序，根据各自的配置文件独立运行。

6.1 哨兵配置

哨兵模块采用独立的配置文件 sentinel.xml。文件内容例如：

```
<?xml version="1.0" encoding="UTF-8"?>
<Server>
```

```
        <Common>
            <SslCiphers>
                TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384,
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,
TLS_RSA_WITH_AES_256_CBC_SHA,          TLS_KRB5_WITH_DES_CBC_MD5,
TLS_KRB5_WITH_3DES_EDE_CBC_SHA,      TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA,
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA,
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,
TLS_RSA_WITH_AES_128_CBC_SHA
            </SslCiphers>
        </Common>
        <Log>
            <!-- nothing, error, warn, info, debug, dump. >
            <    error is the default                    -->
            <Level>info</Level>
        </Log>
        <Listen>
            <Port>26379</Port>
            <Threads>4</Threads>

            <!-- 0: telnet; 1: SSL. -->
            <Secure>0</Secure>
            <IsPlainPassword>true</IsPlainPassword>
            <!--
<Password>454d?51192b1704c60e19734ce6b38203</Password>-->
            <Password>123</Password>
        </Listen>
        <Sentinels>
            <Sentinel>
                <Host>localhost</Host>
                <Port>26379</Port>
            </Sentinel>
            <Sentinel>
                <Host>localhost</Host>
                <Port>26380</Port>
            </Sentinel>
            <Sentinel>
                <Host>localhost</Host>
                <Port>26381</Port>
            </Sentinel>
        </Sentinels>
```

```
        <Services>
            <WebSession>
                <!-- 0：telnet；1：SSL；2：password；3：SSL + password. -->
                <Secure>0</Secure>
                <IsPlainPassword>true</IsPlainPassword>
                <Password>123</Password>
                <EndPoints>192.168.0.87:6379, 192.168.0.86:6379</EndPoints>
            </WebSession>
        </Services>
    </Server>
```

6.1.1通用配置

Server.Common.SslCiphers：当启用 SSL 连接时，指定密钥交换使用的算法。早期版本的 SSL 存在“SSL/TLS 服务器瞬时 Diffie-Hellman 公共密钥过弱”漏洞。可通过此配置禁止使用 Diffie-Hellman 算法。建议配置：

TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,

TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384,

TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA,

TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA,

TLS_KRB5_WITH_DES_CBC_MD5, TLS_KRB5_WITH_3DES_EDE_CBC_SHA,

TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA,

TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA,

TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,

TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_128_CBC_SHA

6.1.2日志

Server.Log 部分用于哨兵程序的日志配置。

File：日志文件名，缺省为“sentinel.log”

Level：日志级别，缺省为“warn”

BackDates：日志保存天数，超过时间的日志会被删除，缺省为 0（不删除），1 为只保留当天日志，以此类推。

6.1.3Listen

Listen 部分是哨兵模块监听端口、登录哨兵时的密码等配置（登录哨兵必须要密码认证，不允许无认证方式登录）

Port 为哨兵监听的端口，缺省为 26389

Secure 定义是否开启 ssl 安全连接

Password 定义登录哨兵的密码，

IsPlainPassword 决定保存的密码是否被加密过，true 为明文。

IdleTimeout 定义哨兵端口连接空闲超时时间。当端口在此配置时间内没有读写操作则会关闭连接。

当前配置时，客户端采用普通连接，密码为 123 登录哨兵。

6.1.4Sentinels

Sentinels 部分用于配置哨兵组中的其他哨兵的地址，目前用于“SENTINEL snetinels master”等命令时获取其他独立哨兵的信息（多台独立哨兵可以配置相同，程序自动判断并标记指向自己的配置项）。

程序会采用自身 Listen 节中配置的安全等级尝试连接其他哨兵程序，因此各哨兵的安全等级和密码等配置项必须相同。

Host：其他哨兵程序的服务器地址。

Port：其他哨兵程序使用的端口。

6.1.5Services

Services 部分为被监控的服务的配置，其下面的配置都是针对 RDS-Node 服务节点的。Services 下可同时配置多个服务，每个服务配置一段，名称对应 RDS 的服务名（cfg.xml 中的 Server.Common.Service 配置项）。举例中配置了一个名为 WebSession 的服务。配置项含义如下：

Secure：安全等级，和 RDS 服务节点 cfg.xmlServer.Listen.Secure 含义相同，此配置项需要与被连接的 RDS 的配置值相同。

Username：连接 RDS 服务节点认证时采用的用户名。当 RDS 启用了 ACL 认证时，需要使用用户名、密码登录，本配置用于指定登录 RDS 的用户名。由于哨兵需要通过PING命令判断RDS节点是否正常，因此ACL需要配置该用户执行PING命令的权限。（**注意：如认证只需要密码则不能配置该项**）

Password：连接 RDS 服务节点认证时采用的密码，如果 Secure 配置需要认证则该项不能为空。

IsPlainPassword：上述配置的密码是否为明文，如果不是明文程序再需要使用密码时会首先尝试解密。

EndPoints：RDS 服务节点的网络地址，格式为 IP:Port，同时配多项用“.”分隔。

6.2 服务节点配置

启用独立的哨兵模块时，TongRDS 服务节点无需特殊配置。

6.3 客户端 jedis（以 Version3.5.0 为例）连接

按照上述配置，jedis 客户端采用如下参数连接（其中哨兵模块运行再 192.168.0.86：26389 上）：

```
JedisSentinelPool pool = new JedisSentinelPool("WebSession",
new HashSet<String>() {{
```

```
            this.add("192.168.0.86:26379");
        }}, (String)null, "123");
        Jedis jedis = pool.getResource();

        jedis.set("aaa", "ddd");
        System.out.println("aaa = " + jedis.get("aaa"));

        pool.close();
        jedis.close();
```

其中构造 JedisSentinelPool 的最后 2 个参数分别是连接 RDS 服务节点的密码和连接哨兵的密码，用例中的参数含义为 RDS 不需要密码，哨兵密码是“123”。独立的哨兵模块也可以配置为无密码接入。

7 Center 哨兵配置样例

本样例由中心节点管理 2 个服务节点，这 2 个服务节点工作在主备模式，并由 Center 中心节点提供哨兵功能。

7.1 样例环境

本次测试采用 2 台主机，分别为服务器 1（192.168.0.86）和服务器 2（192.168.0.87），服务器 1 主机上运行中心（Center）节点和服务节点 1，服务器 2 主机上运行服务节点 2。

两台主机采用相同的 ntp 服务器同步时钟。

java 版本为 OpenJdk 1.8（openjdk version "1.8.0_302"）。

验证用 Jedis 版本为 3.7.0

7.2 软件安装

将 TongRDS-2.2.1.2.MC.tar.gz 中心节点程序包和 TongRDS-2.2.1.2.Node.tar.gz 上传到服务器 1，将

TongRDS-2.2.1.2.Node.tar.gz 上传到服务器 2。

分别解 tar 包，生成 pmemdb 目录和 pcenter 目录。其中 pmemdb 目录为服务节点程序，pcenter 目录下为中心节点程序。

7.3 服务节点配置

首先，检查一下 cfg.xml 中的 Server.Service 的配置为 WebSession。该服务名必须和 Center 节点中的配置对应，无法对应则节点不允许接入。

然后修改 dynamic.xml 配置文件如下：

```
<?xml version="1.0" encoding="UTF-8"?>

<Server>
  <Center>
    <Password>454d51192b1704c60e19734ce6b38203</Password>
    <EndPoint>
      <Host>192.168.0.86</Host>
      <Port>6300</Port>
    </EndPoint>
  </Center>
</Server>
```

其中 Server.Center.Password 为连接 Center 时的认证密码，测试时可不用修改。

Sever.Center.EndPoint 配置修改 IP 地址为 192.168.0.86，端口为 6300。如果有多台 Center 节点，EndPoint 可配置多条，例如：

```
<?xml version="1.0" encoding="UTF-8"?>

<Server>
  <Center>
    <Password>454d51192b1704c60e19734ce6b38203</Password>
    <EndPoint>
      <Host>192.168.0.86</Host>
      <Port>6300</Port>
    </EndPoint>
    <EndPoint>
      <Host>192.168.0.87</Host>
      <Port>6300</Port>
```

```
    </EndPoint>
  </Center>
</Server>
```

7.4 中心节点配置

7.4.1config. properteis

首先检查config.properteis文件中的service.port=6300(Center的主服务端口)，sentinel.port=26379 （Redis 哨兵的仿真接口， 26379 为哨兵的缺省端口），server.password=454d51192b1704c60e19734ce6b38203（节点接入时的认证密码，采用 SM4 加密)。

7.4.2cluster.properties

然后修改 cluster.properties（该文件用于定义服务节点的工作状态，如集群状态、哨兵状态等)， cluster.properties 配置修改如下（哨兵模式)：

```
WebSession.type=sentinel
WebSession.nodes=2
WebSession.node0=192.168.0.86:6200
WebSession.node1=192.168.0.87:6200
```

其中 WebSession.type 定义名为 WebSession(对应服务节点中的 Server.Service)的服务状态为哨兵模式。

WebSession.nodes 定义 2 个服务节点的地址和端口（此处配置需要和节点的实际的运行配置对应)。

7.4.3active.properties

检查 active.properties 文件中是否有 WebSession 的定义，例如：

```
acioweor_483kja03np4h8238G = WebSession, AuthService
```

其中的“acioweor_483kja03np4h8238G”是激活码，对应哨兵接入时的密码，等号后面的内容是该激活码对应的服务名称，多个名称用“,”分隔。

7.4.4sync.properties

如果有多台中心节点集群，可修改此配置文件。例如配置 2 台 Center 集群：

```
sync.servers=2
sync.server1.host=192.168.0.86
sync.server1.port=6300
sync.server2.host=192.168.0.87
sync.server2.port=6300
```

本例可不配置此文件。

7.5 启动服务

分别启动中心节点和两个服务节点。

启动成功后，服务节点的 dynamic.xml 文件将被修改为类似如下：

```
<?xml version="1.0" encoding="UTF-8"?>

<Server>
  <Center>
    <Password>454d51192b1704c60e19734ce6b38203</Password>
    <EndPoint>
      <Host>node1</Host>
      <Port>6300</Port>
    </EndPoint>
  </Center>
  <Synchronize>
    <EndPoint>
      <Host>192.168.0.86</Host>
      <Port>6200</Port>
    </EndPoint>
    <EndPoint>
      <Host>192.168.0.87</Host>
      <Port>6200</Port>
    </EndPoint>
```

```
    </Synchronize>
</Server>
```

其中的 Synchronize 部分是从 Center 节点的 cluster.properties 配置中同步来的。

7.6 测试

7.6.1 测试目的

使用 jedis 客户端的哨兵模式接入，验证 RDS 中心节点模拟哨兵的功能；采用 jedis 多次接入，验证 RDS 模拟主节点功能。

本例使用 jedis 3.7.0（group：'redis.clients'，name：'jedis'，version：'3.7.0'）测试通过。

注：jedis 不同版本存在连接哨兵的 bug，例如 3.6.x 版本，无法采用有密码方式连接哨兵。如果测试不成功请首先检查 jedis 版本

7.6.2 Jedis 接入（jedis 版本 3.7.0）

创建一个 java 类，输入如下代码：

```
    public static void main(String args[]) {
        JedisSentinelPool pool = new JedisSentinelPool("WebSession",
new HashSet<String>() {{
            this.add("192.168.0.86:26379");
        }}, (String) null, "acioweor_483kja03np4h8238G");

        Jedis jedis = pool.getResource();

        jedis.set("aaa", "ddd");

        System.out.println("aaa = " + jedis.get("aaa"));

        jedis.close();
        pool.close();
    }
```

其中："WebSession"是服务名，需要与中心节点、服务节点的配置一致；"192.168.0.86:26379"为 Center 节点仿真哨兵的端口位置；"acioweor_483kja03np4h8238G"是哨兵接入的密码，对应 Center 的 active.properties 中的配置。

运行程序，在服务器 1（192.168.0.86）上观察到如下日志：

```
CacheServer::set() Set aaa<> = ddd ok
CacheServer::process_get() Get aaa = 'ddd' ok.
```

在服务器 2（192.168.0.87）上观察到如下日志：

```
CacheServer::sync() Sync aaa<> = ddd at 1629777554660 ok(0).
```

日志分析可知，jedis 从哨兵端口获得了主节点的访问端口，并成功完成读写操作，节点 2 获得同步数据。

注：Center 的哨兵功能不允许无密码接入，较低版本的 jedis 客户端无法使用。

7.6.3验证主节点保持

继续上例测试，多次运行程序，观察读写操作均出现在节点 1 的日志中，节点 2 中始终是同步日志，说明正常情况下每次接入的操作均发生在一个节点上，另外的节点只负责备份。

7.6.4备份节点异常测试

将备份节点杀掉。再次运行上述程序，观察主节点日志有正常的读写记录，说明服务正常。

将备份节点恢复，再次运行程序，读写仍然发生在主节点，说明备份节点的

启动停止不会引起主备切换。

7.6.5主节点异常测试

将主节点杀掉，再次运行程序，程序可正常完成。观察 2 节点日志发现，set 和 get 的操作日志出现在备份节点，说明中心节点做了主备切换。

将主节点恢复（等待其启动完成），再次运行程序，程序可正常完成。观察节点日志，set 和 get 操作的日志出现在主节点（节点 1），节点 2 上仍然是同步日志，说明 Center 将主节点切换回了节点 1。

8 集群配置样例

本样例验证用 Center 节点统一管理的集群配置方式（以常见的 3 主 3 从配置为例）。实现时，全部有关集群配置均配置到 Center 节点，服务节点启动时连接 Center 节点，由 Center 节点下发并生效集群配置。

本例 Center 节点安装在 192.168.0.86 主机，服务主端口为 6300。

Service 服务节点分别部署在 192.168.0.10、192.168.0.11、192.168.0.20、192.168.0.21、192.168.0.30、192.168.0.31 等 6 台主机，端口均为 6200（RDS 协议端口）和 6379（Redis 仿真端口）。

8.1 服务节点配置

注：服务节点的集群方式配置非常类似于哨兵方式的配置

首先，检查一下 cfg.xml 中的 Server.Service 的配置为 WebSession。该服务名必须和 Center 节点中的配置对应，无法对应则节点不允许接入。

然后修改 dynamic.xml 配置文件如下：

```
<?xml version="1.0" encoding="UTF-8"?>

<Server>
  <Center>
    <Password>454d51192b1704c60e19734ce6b38203</Password>
    <EndPoint>
      <Host>192.168.0.86</Host>
      <Port>6300</Port>
    </EndPoint>
  </Center>
</Server>
```

其中 Server.Center.Password 为连接 Center 时的认证密码，测试时可不用修改。

Sever.Center.EndPoint 配置修改 IP 地址为 192.168.0.86，端口为 6300。如果有多台 Center 节点，EndPoint 可配置多条，例如：

```
<?xml version="1.0" encoding="UTF-8"?>

<Server>
  <Center>
    <Password>454d51192b1704c60e19734ce6b38203</Password>
    <EndPoint>
      <Host>192.168.0.86</Host>
      <Port>6300</Port>
    </EndPoint>
    <EndPoint>
      <Host>192.168.0.87</Host>
      <Port>6300</Port>
    </EndPoint>
  </Center>
</Server>
```

8.2 中心节点配置

8.2.1config. properteis

首先检查config.properteis文件中的service.port=6300(Center 的主服务端口)，server.password=454d51192b1704c60e19734ce6b38203（节点接入时的认证密码，

采用 SM4 加密)。

8.2.2cluster.properties

然后修改 cluster.properties（该文件用于定义服务节点的工作状态，如集群状态、哨兵状态等），cluster.properties 配置修改如下（集群模式）：

```
WebSession.type=cluster
WebSession.shards=3
WebSession.shard0.nodes=192.168.0.10:6200, 192.168.0.11:6200
WebSession.shard0.slots=0-5000
WebSession.shard1.nodes=192.168.0.20:6200, 192.168.0.21:6200
WebSession.shard1.slots=5001-10000
WebSession.shard2.nodes=192.168.0.30:6200, 192.168.0.31:6200
WebSession.shard2.slots=10001-16383
```

其中 WebSession.type=cluster 含义为：服务“WebSession”(对应服务节点中的 Server.Service=WebSession 配置）的状态为集群（cluster）模式。

WebSession.shards 定义了服务“WebSession”包含 3 个服务节点分组，WebSession.shard0.nodes 定义第一个 clustar 分组的服务节点地址，排在前面的是主节点，后面的是从节点（此处的配置需要和实际的服务节点的运行情况对应）；WebSession.shard0.slots 定义第一个分组处理的数据切片的范围。

8.2.3sync.properties

如果有多台中心节点集群，可修改此配置文件。例如配置 2 台 Center 集群：

```
sync.servers=2
sync.server1.host=192.168.0.86
sync.server1.port=6300
sync.server2.host=192.168.0.87
sync.server2.port=6300
```

本例可不配置此文件。

8.3 启动集群

1、 启动 Center 节点

2、 分别启动 6 台服务节点。服务节点启动成功会连接 Center 节点并获得 Cluster 配置，各节点的 dynamic.xml 文件将被修改为类似如下：

```
<?xml version="1.0" encoding="UTF-8"?>

<Server>
  <Center>
    <Password>454d51192b1704c60e19734ce6b38203</Password>
    <EndPoint>
      <Host>192.168.0.86</Host>
      <Port>6300</Port>
    </EndPoint>
  </Center>
  <Synchronize>
    <EndPoint>
      <Host>192.168.0.10</Host>
      <Port>6200</Port>
    </EndPoint>
    <EndPoint>
      <Host>192.168.0.11</Host>
      <Port>6200</Port>
    </EndPoint>
  </Synchronize>
  <Clusters>
    <Cluster>
      <EndPoints>192.168.0.10:6200, 192.168.0.11:6200/EndPoints>
      <Slots>0-5000</Slots>
    </Cluster>
    <Cluster>
      <EndPoints>192.168.0.20:6200, 192.168.0.21:6200</EndPoints>
      <Slots>5001-10000</Slots>
    </Cluster>
    <Cluster>
      <EndPoints>192.168.0.30:6200, 192.168.0.31:6200</EndPoints>
      <Slots>10001-16383</Slots>
    </Cluster>
  </Clusters>
```

```
</Server>
```

8.4 测试

采用 jedis（版本 3.7.0）连接 RDS 集群，并进行简单的数据读写操作，验证集群的可用性以及仿真 Redis。

新建一个 java 对象，加入如下 main 方法并执行：

```
public static void main(String[] args){
    HashSet<HostAndPort> nodes = new HashSet<>();
    ArrayList list = null;
    nodes.add(new HostAndPort("192.168.0.10", 6379));
    nodes.add(new HostAndPort("192.168.0.11", 6379));
    JedisCluster jedisCluster = new JedisCluster(nodes, 1000, 1000, 3, "123", new
GenericObjectPoolConfig());

    System.out.println(jedisCluster.set("aaa", "1001"));
    System.out.println(jedisCluster.get("aaa"));

    System.out.println(jedisCluster.set("aba", "1002"));
    System.out.println(jedisCluster.get("aba"));

    System.out.println(jedisCluster.set("abc", "1003"));
    System.out.println(jedisCluster.get("abc"));

    System.out.println(jedisCluster.set("bbb", "1011"));
    System.out.println(jedisCluster.get("bbb"));

    System.out.println(jedisCluster.set("bab", "1012"));
    System.out.println(jedisCluster.get("bab"));

    System.out.println(jedisCluster.set("bac", "1013"));
    System.out.println(jedisCluster.get("bac"));

    System.out.println(jedisCluster.set("bbc", "1014"));
    System.out.println(jedisCluster.get("bbc"));

    System.out.println(jedisCluster.set("ccc", "1021"));
    System.out.println(jedisCluster.get("ccc"));
```

```
        System.out.println(jedisCluster.set("cac", "1022"));
        System.out.println(jedisCluster.get("cac"));

        System.out.println(jedisCluster.set("cbc", "1023"));
        System.out.println(jedisCluster.get("cbc"));

        jedisCluster.subscribe(new MyPubsub(),"channel");

        jedisCluster.publish("channel","aaa");

         jedisCluster.close();
    }
```

9 故障排除

9.1 主备节点数据不同步

当配置完系统测试时发现主备节点不能同步数据，可从以下几个方面检查：

1. 检查各节点是否有 license 授权（使用 info 命令查看 Server 节的 license 内容），无授权的节点不能进行主备数据同步
2. 检查各节点 cfg.xml 中的 Listen 中的 Secure 配置和 Password 配置是否一致
3. 检查 cfg.xml 中 Sync 的配置，ListNumbers 或 ListLength 是否配成了 0，
4. 查 dynamic.cml 中 Synchronize 配置列表中是否有需要同步的全部节点（如果连接的 Center，不能直接改此处配置，需要改 Center 配置）
5. 检查主、备节点上的 cfg.xml 中的 BinaryCompatible 和 BinaryCompatibleKey 配置是否一致。