**TTC配置文件说明**

1. **TTC配置文件**

TTC配置文件位于TTC编译安装目录的conf文件夹下，主要有的配置文件如下所列：

1. Agent配置文件：agent.xml
2. Cache配置文件：cache.conf
3. 热备配置文件 ： hbp.conf
4. 数据库配置文件：table.conf
5. 扩容迁移配置文件：cluster.conf
6. **TTC Agent配置文件**

XML类型的配置文件agent.xml



|  |  |  |
| --- | --- | --- |
| AGENT\_CONFIG | ClientIdleTime | 连接超时时间 |
| AdminListen | TTC端监听端口 |
| Version | 版本号 |
| MasterAddr | Master平台地址，拉取最新配置文件 |
| MODULE | Id | 模块号(BUSINESS\_MODULE下可包含多个模块) |
| ListenOn | 该模块的监听客户连接端口 |
| CACHEINSTANCE | Name | 实例名（一个模块下可包含多个实例） |
| Addr | 该实例名对应的TTC服务地址 |

在一台agent上可以部署多个模块，agent根据监听端口的区别，判断对不同的TTC模块的访问，模块ID代表一组TTC实例，可以包含后端的多个TTC服务，agent使用一致性hash判断路由的规则。

1. **Cache配置文件**

Cache配置文件为conf目录下的cache.conf文件，其包含的配置项及配置项的基本含义如下表所示：

|  |  |  |
| --- | --- | --- |
| 配置文件项 | 基本含义 | 说明 |
| **LogLevel** | 日志级别 | emerg, alert, crit, error, warning, notice, info, debug。默认日志级别是info |
| **BindAddr** | TTC服务地址 |  |
| **BindPort** | TTC服务端口 |  |
| **MaxListenCount** | 等待连接队列长度 | Backlog大小 |
| **UdpRecvBufferSize** | UDP接收缓冲区 | 当前无意义 |
| **UdpSendBufferSize** | UDP发送缓冲区 | 当前无意义 |
| **MaxRequestWindow** | 对tcp和unix socket启动异步连接处理，这个值是允许异步处理的单连接未完成请求数 |  |
| **MaxFdCount** | 设置最大的FD支持 | =0为不配置为默认值  /proc/sys/fs/file-max，一般根据内存大小配置 |
| **StartStatReporter** | 是否自动运行二级网管上报数据 | 上报TNM2数据 |
| **WatchDogTime** | 监控进程的轮询时间，单位秒 |  |
| **ServerRecovery** | 主进程crash监控 | None:主进程退出则退出虽有进程  Crash：主进程crash是重启CrashDebug：crash重起并打开core  Killed：被-9杀掉时也重启  Error：退出码非零时也重起  Always |
| **MaxIncomingPollers** | Epoll fd的最大个数 | Thread对应 |
| **MaxBarrierCount** | 系统最大的barrier个数。需要访问DB时，请求需要先放进桶，每个key值对应一个桶。它跟下面的MaxKeyCount主要是用来对db请求排序的，防止同时多个相同key的update请求由于到mysql的顺序搞乱了导致数据错误 |  |
| **MaxKeyCount** |
| **MaxTaskCount** |  | 当前版本源代码中找不到这个配置项相关代码 |
| **MaxDropCount** | 当cache的内存不够，需要淘汰数据的时候，每次淘汰多少个key节点 | 当前版本已经停用，代码被注销 |
| **IdleTimeout** | agent与ttc的长连接最大空闲时间 | 已经没什么作用 |
| **HelperTimeout** | helper处理一个请求的最长时间，helper超时 |  |
| **DisableDataSource** | 是否启动成无数据源模式。默认为0，即需要数据源 | 无源模式开关 |
| **CacheShmKey** | 缓存所使用共享内存的key |  |
| **CacheShmVersion** | 4.0版本以上默认为4 |  |
| **CacheNodeTotal** |  | 当前版本源代码中找不到这个配置项相关代码 |
| **CacheBucketTotal** |  | 当前版本源代码中找不到这个配置项相关代码 |
| **CacheChunkTotal** |  | 当前版本源代码中找不到这个配置项相关代码 |
| **CacheChunkSize** |  | 当前版本源代码中找不到这个配置项相关代码 |
| **CacheAverageDataSize** |  | 当前版本源代码中找不到这个配置项相关代码 |
| **CacheMemorySize** | 缓存空间大小 | 需要大于50M |
| **PreAllocNGNum** | 预分配Node Group的数量 | 默认不配置 |
| **CacheHashDepth** |  | 当前版本源代码中找不到这个配置项相关代码 |
| **CacheChunkDepth** |  | 当前版本源代码中找不到这个配置项相关代码 |
| **UseMatchedAsAffectedRows** | 在执行Update时，API AffectedRows()返回符合Update条件的记录数，而不是实际Update(update前后数据真正改变了)的记录数 |  |
| **DisableLRUUpdate** | LRU结点更新规则 | 0： 总是更新LRU，节点按最后访问时间排序  1：批量读不更新LRU  2：所有读请求不更新LRU，节点按最后修改时间排序  3：所有读写请求都不更新LRU，节点按产生顺序排序 |
| **DelayUpdate** | 启动或者关闭异步更新功能 | 异步版本上使用同步更新逻辑，默认配置为同步模式 |
| **MarkerPrecision** | 异步V3内存格式中的时间标记节点密度 | 已经不起作用，程序中默认写死为1 |
| **MaxFlushSpeed** | 控制并发脏数据flush的node 的最大数 |  |
| **MinDirtyTime** | 两个时间控制脏数据存活时间，脏数据量达到min时间时开始单线程flush。  # 以后逐渐增产速度。超过max时间以后以MaxFlushSpeed设定的最大速度flush |  |
| **MaxDirtyTime** |
| **LimitNodeRows** | 每个NODE（相当于一个KEY对应的数据）可以拥有的数据的行数 |  |
| **LimitEmptyNodes** | 打开空节点限制功能 | #<=0: 表示关闭这个功能，使用旧逻辑，允许无限空节点，也不统计空节点个数  #1-999: 非法，最小值必须1000，作为1000处理  #1000-1G(含）:表示最大空节点个数  #>1G ，非法，作为无限制处理，也就是0  #>=2G，非法，符号整数溢出，结果不确定 |
| **DisableAutoPurge** | 禁止自动淘汰功能 | 禁止自动淘汰，则当内存不够分配时候，扩张node的操作将失败，insert node的操作将失败。缩小或者删除node的操作可以成功 |
| **LossyDataSource** | AffectedRows的返回值以canche为准，还是以数据源为准的配置 | 主要应用在数据迁移模式下，用于确定是以前端为准，还是以后端数据为准 |
| **AutoPurgeAlertTime** | 配置AutoPurgeAlertTime必须配置时间戳字段lastcmod，即在表结构中必须有字段的DefaultValue = lastcmod | 配置AutoPurgeAlertTime= 100，表示如果当前淘汰告警设置的时间为100s。ttc内部每条数据的修改时间Tm都会被记录下来。该数据被淘汰的时间点时间为Tp。如果Tp-Tm的差值小于设定的时间100s则给予报警。 |
| **EnablePlugin** | 打开ttc内嵌so的功能，使其可以加载so |  |
| **PluginName** | 需要加载的so的路径以及文件名 |  |
| **PluginWorkerNumber** | 需要启动多少个work thread进行二次开发逻辑的运行 |  |
| **PluginAddr** | 插件的监听地址 |  |
| **PluginTimerNotifyInterval** | 使用自定义逻辑淘汰数据,指定淘汰间隔，单位为秒 |  |
| **PluginConfigFile** | 插件配置文件 |  |
| **PluginNetworkMode** | 插件SetAddress的时候如果地址跟ttc绑定的地址相同时是否依旧走网络 |  |

1. **数据库配置文件**

|  |  |  |
| --- | --- | --- |
| **DB\_DEFINE（db分布信息配置）** | **DbName** | DB名，分库则为前缀 |
| **DbNum** | (100,10) （idx/100）%10  此处定义了db的路由规则 |
| **MachineNum** | 机器数 |
| **Deploy** | 不分库分表=0, 只分库=1, 只分表=2, 既分库也分表=3 |
| **DbMax** | 一共有多少个DB |
| **SlaveGuardTime** | MySQL DB 复制保护期 |
| **CheckTableConfig** | 是否检查mysql的表定义跟table.conf一致性 |
| **EnableKeyHash** | 如果key是string或者binary类型，且要分库分表，则要指定hash算法 |
| **KeyHashSo** | 定key-hash算法的插件路径，默认为 ../lib/key-hash.so |
| **KeyHashFunction** | 指定hash算法 和待hash的string key 范围，目前提供IntHash和StringHash两种算法 |
| **Machine（db机器配置,看可以配置多个）** | **Procs** | DB读进程个数,推荐8个 |
| **WriteProcs** | DB写进程个数，推荐8个 |
| **CommitProcs** | 仅异步版本使用。淘汰脏数据的时候，写回DB进程个数 |
| **DbIdx** | 数据库后缀数字，有多个不连续段可以用逗号隔开，连续数字可以用“-”隔开 |
| HelperType | 配置数据源类型。目前支持MYSQL/TDB/CUSTOM三种，其中CUSTOM需要业务自己写插件实现helper功能 |
| **DbAddr** | Db 的ip地址 |
| **DbUser** | Db 的用户名 |
| **DbPass** | Db 的password |
| **MyCnf** | Mysql配置文件 |
| **QueueSize** | 到MYSQL请求的队列大小 |
| **WriteQueueSize** | 到MYSQL请求的写队列大小 |
| **DbAddr1** | slave地址 |
| **DbUser1** | slave用户名 |
| **DbPass1** | slave密码 |
| **Workload** | 设定复制支持下的负载分配策略 |
| **Machine级联配置** | **Procs** | 新ttc连接作为数据源的老ttc的连接数 |
| **HelperType** | 指明以ttc作为数据源 |
| **DbIdx** | 指明作为数据源的ttc的号段 |
| **DbAddr** | 指定作为数据源的ttc的ip和port，并指明//连接方式 |
| **TABLE\_DEFINE（表配置）** | **TableName** | 表的名称，分表则为前缀 |
| **FieldCount** | 字段个数 |
| **KeyFieldCount** | 设定cache存储中的key字段数 |
| **TableNum** | (1,10) （key/1）%10  此处定义了表的路由规则 |
| **ServerOrderBySQL** | 控制向DB发送Select时，加上对应的ORDER BY字符串 |
| **ServerOrderInsert** | 使用ServerOrder功能时，插入数据的处理方式  # first 插入数据放在前面，相当于按插入时间倒序。Update不做顺序调整  # last 插入数据放在最后面，相当于按插入时间正序。Update不做顺序调整  #purge 插入或者update时，purge掉cache中的数据，下次访问从DB重新拉取。这个配置可以支持任意条件排序，但是会影响cache命中率，和DB的负载 |
| **FIELD字段定义** | **FieldName** | 字段名称 |
| **FieldType** | 字段类型定义： Signed=1, Unsigned=2, Float=3, String=4, Binary=5 |
| **FieldSize** | 字段长度 |
| **ReadOnly** | 是否只读，只读不能被update |
| **DefaultValue** | 默认值 |