ZooKeeper Administrator's Guide中文版

存档 `zookeeper`

> 本文档主要介绍ZooKeeper部署及日常管理

安装部署

本章节包含与ZooKeeper部署有关的内容，具体来说包含下面三部分内容：

- 系统软硬件需求
- 集群部署（安装）
- 单机开发环境部署（安装）

前两部分主要介绍如何在数据中心等生产环境上安装部署ZooKeeper，第三部分则介绍如何在非生产环境上（如为了评估、测试、开发等目的）安装部署ZooKeeper。

系统软硬件需求

支持的OS平台

ZooKeeper框架由多个组件组成，有的组件支持全部平台，而还有一些组件只支持部分平台，详细支持情况如下：

- **Client**：它是一个 `Java` 客户端连接库，上层应用系统通过它连接至ZooKeeper集群。
- **Server**：它是运行在ZooKeeper集群节点上的一个 `Java` 后台服务程序。
- **Native Client**：它是一个用 `C` 语言实现的客户端连接库，其与 `Java` 客户端库一样，上层应用（非 `Java` 实现）通过它连接至ZooKeeper集群。
- **Contrib**：它是指多个可选的扩展组件。

操作系统	Client	Server	Native Client	Contrib
GNU/Linux	D + P	D + P	D + P	D + P
Solaris	D + P	D + P	/	/
FreeBSD	D + P	D + P	/	/
Windows	D + P	D + P	/	/
Mac OS X	D	D	/	/

> **D：支持开发环境， P：支持生成环境， /：不支持任何环境**

上表中未显式注明支持的组件在相应平台上可能不能正常运行。虽然ZooKeeper社区会尽量修复在未支持平台上发现的BUG，但并无法保证会修复全部BUG。

软件要求

ZooKeeper需要运行在JDK6或以上版本中。若ZooKeeper以集群模式部署，则推荐的节点数至少为3，同时建议部署在独立的服务器上。在Yahoo!，ZooKeeper通常部署在运行RHEL系统的服务器上（服务器配置：双核CPU、2G内存、80G容量IDE硬盘）。

集群部署（安装）

为了保证ZooKeeper服务的可靠性，您应该以集群模式部署ZooKeeper服务。只要半数以上的集群节点在线，服务将是可用的。因为ZooKeeper需要半数以上节点同意才能选举出Leader，所以建议ZooKeeper集群的节点数为奇数个。举个例子，对于有四个节点的集群只能应付一个节点宕机的异常，如果有两个节点宕机，则剩下两个节点未达到法定的半数以上选票，ZooKeeper服务将变为不可用。而如果集群有五个节点，则集群就可以应付二个节点宕机的异常。

> **提示：**
> 正如《ZooKeeper快速入门》文档中所提到的，至少需要三个节点的ZooKeeper集群才具备容灾特性，因此我们强烈建议集群节点数为奇数。
> 通常情况下，生产环境下，集群节点数只需要三个。但如果为了在服务维护场景下也保证最大的可靠性，您也许会部署五个节点的集群。原因很简单，如果集群节点为三个，当你对其中一个节点进行维护操作，将很有可能因维护操作导致集群异常。而如果集群节点为5个，那你可以直接将维护节点下线，此时集群仍然可正常提供服务（就算四个节点中的任意一个突然宕机）。
> 您的冗余措施应该包括生产环境的各个方面。如果你部署三个节点的ZooKeeper集群，但你却将这三个节点都连接至同一个网络交换机，那么当交换机宕掉时，你的集群服务也一样是不可用的。

关于如何配置服务器使其成为集群中的一个成员节点的详细步骤如下（每个节点的操作类似）：

- 1.安装JDK，你可以使用本地包管理系统安装，也可以从JDK官网下载
- 2.设置Java堆相关参数，这对于避免内存swap来说非常重要。因为频繁的swap将严重降低ZooKeeper集群的性能。您可以通过使用压力负载测试来决定一个合适的值，确保该值刚好低于触发swap的阈值。保守的做法是：当节点拥有4G的内存，则设置-xmx=3G。
- 3.安装ZooKeeper，您可以从官网下载：http://zookeeper.apache.org/releases.html
- 4.创建一个配置文件，这个文件可以随便命名，您可以先使用如下设置参数：

```
tickTime=2000
dataDir=/var/lib/zookeeper/
clientPort=2181
initLimit=5
syncLimit=2
server.1=zoo1:2888:3888
server.2=zoo2:2888:3888
server.3=zoo3:2888:3888
```

> 您可以在配置参数章节中找到上面这些参数及其他参数的解释说明。这里简要说一下：集群中的每个节点都需要知道集群中其它节点成员的连接信息。通过上述配置文件中格式为 `server.id=host:port:port` 的若干行配置信息您就可以实现这个目标。 `host` 与 `port` 意思很简单明了，不多作说明。而 `server.${id}` 代表节点ID，你需要为每个节点创建一个名为 `myid` 的文件，该文件存放于参数 `dataDir` 所指向的目录下。

- 5. `myid` 文件的内容只有一行，其值为配置参数 `server.${id}` 中 `${id}` 的实际值。即服务器1对应的 `myid` 文件的内容为1，这个值在集群中必须保证其唯一性，同时必须处于[1, 255]之间。
- 6.如果你已经创建好配置文件（如zoo.cfg），你就可以启动ZooKeeper服务：

```
$ java -cp zookeeper.jar:lib/slf4j-api-1.6.1.jar:lib/
slf4j-log4j12-1.6.1.jar:lib/log4j-1.2.15.jar:conf \
org.apache.zookeeper.server.quorum.QuorumPeerMain zoo.cfg
```

> QuorumPeerMain启动一个ZooKeeper服务，JMX管理bean也同时被注册，通过这些JMX管理bean，你就可以在JMX管理控制台上对ZooKeeper服务进行监控管理。ZooKeeper的JMX文档有更详细的介绍信息。同时，你也可以在 `$ZOOKEEPER_HOME/bin` 目录下找到ZooKeeper的启动脚本 `zkServer.sh` 。

- 7.接下来你就可以连接至ZooKeeper节点来测试部署是否成功。
  在 `Java` 环境下，你可以运行下面的命令连接至已启动的ZooKeeper服务节点并执行一些简单的操作

```
$ bin/zkCli.sh -server 127.0.0.1:2181
```

单机开发环境部署介绍

如果你想部署ZooKeeper以便于开发测试，那你可以使用单机部署模式。然后安装Java或C客户端连接库，同时在配置文件中将服务器信息与开发机绑定。
具体安装部署步骤也集群部署模式类似，唯一不同的是zoo.cfg配置文件更简单一些。你可以从《ZooKeeper快速入门》文档中的相关章节获取详细的操作步骤。
关于安装客户端连接库的相关信息，你可以从ZooKeeper Programmer’s Guide文件的Bindings章节中获取。

维护管理

这部分包含ZooKeeper运行与维护相关的信息，其包含如下几个主题：

ZooKeeper集群部署规划

ZooKeeper可靠性依赖于两个基本的假设：

- 一个集群中只会有少数服务器会出错，这里的出错是指宕机或网络异常而使得服务与集群中的多数服务器失去联系。
- 已部署的机器可以正常运行，所谓正常运行是指所有代码可以正确的执行，有适当且一致的工作时钟、存储、网络。

为了最大可能的保证上述两个前提假设能够成立，这里有几个点需要考虑。一些是关于跨服务器（节点之间）需求的，而另一些是关于集群中每个节点服务器的考虑。

跨机器要求

为了启用ZooKeeper服务，集群中半数以上的节点需要能够相互通信。为了创建一个可以支撑F个节点异常的集群，你需要部署2xF+1个节点。即一个包含三个节点的集群可以应对一个节点异常的情况，一个包含五个节点的集群则可以应对二个节点异常的情况，依此类推。需要注意的是，一个包含六个节点的集群也只能应对二个节点失败的情况，因为三个节点并未超过半数。因此，ZooKeeper集群中的节点数通常为奇数。

为了实现最大限度的容灾能力，你应该尽力使集群节点发生异常时不影响其它节点（即保持节点部署上尽可能的独立）。举个例子，当大部分的机器共享同一个交换机，若这个交换机挂了将使关联的服务都下线。共享电源电路、冷却系统等也是同样的道理（有同样的单点问题）。

单机要求

如果ZooKeeper服务需要与其它应用竞争存储、CPU、网络、内存等资源时，其性能将显著下降。机器必须为ZooKeeper服务提供持久化保障，因为ZooKeeper需要先使用存储设备来保存变更日志，然后才允许变更操作提交。如果你想确保ZooKeeper操作不会因存储而挂起，那你应该意识到这个依赖关系并重视起来。这里有几个事情可以最小化这类问题所带来的消极影响。

- ZooKeeper的事务日志必须存储在专用的设备上（一个专用分区是不够的）ZooKeeper以串行的方式写入日志。如果与其它进程共享日志存储设备将导致磁盘寻址与IO竞争，这最终将导致几秒的时延。
- 不要将Zookeeper放置在可能引起swap的环境。为了保证Zookeeper的实时性，它几乎不允许swap。因此，请确保分配给ZooKeeper的最大堆内存大小不能大于ZooKeeper可用的真实的内存大小。关于这一点的更多信息，请参数后续章节 Things to Avoid。

Provisioning

无

ZooKeeper的优势与局限

无

管理

无

维护

ZooKeeper的长期维护需求几乎没有，然而你仍然需要注意如下几件事情：

持续的数据目录清理

ZooKeeper的数据目录包含集群上特殊存储结点znode数据的持久化拷贝文件。文件中包含快照与事务日志文件。znode上数据的变更将同时会被追加至事务日志中。当这些日志不断增长时，所有znode的当前状态的快照文件将被写回文件系统。这个快照将取代之前的日志。

ZooKeeper服务器默认情况下不会移除旧的快照与日志文件，删除快照与日志这项工作是管理员的责任。因为每个服务环境是不一样的，所以管理这些快照与文件的需求也是不同一。 `PurgeTxnLog` 工具实现了一个简单的保留策略，管理员可以使用这个工具，API docs 中包含有详细的使用说明。下面的例子的作用为：保留最近 `<count>` 个快照及相关的日志，其它的快照及相关日志则删除。 `<count>` 的值通常要大于3（虽然不是必须的，但提供3个备份时将极大降低日志损坏的可能性）。您可以在ZooKeeper服务器上运行 `cron` 脚本来每天清除过期日志。

```
$ java -cp zookeeper.jar:lib/slf4j-api-1.6.1.jar:lib/slf4j-log4j12-1.6.1.
jar:lib/
log4j-1.2.15.jar:conf org.apache.zookeeper.server.PurgeTxnLog <dataDir> <
snapDir> -n <count>
```

在V3.4.0版本中引入的自动清除快照及相关事务日志的特性可以通过配置参数 `autopurge.snapRetainCount` 与 `autopurge.purgeInterval` 来启用。更多的使用见后续的配置参数（高级配置）章节。

调试日志清理 (log4j)

通过本文档的日志章节可知，ZooKeeper将使用Log4j内置的日志滚动覆盖功能最多生成N个日志文件。Log4j的配置示例可在发行版本tar包的 `conf/log4j.properties` 文件中找到。

Supervision

你需要一个监督进程来管理ZooKeeper服务进程（本质上为一个Java进程）。ZooKeeper服务具有“快速失败”的特性，即只要服务发生不可恢复的错误，则ZK进程直接退出。因为ZK集群是高可用的，所有集群中的一个ZK服务虽然异常退出了，但整个集群仍然是可用的，仍然可对外提供服务。同时由于集群可以自行恢复特性，一旦此前异常退出的服务器重启后，它将自动重新加入集群中而不需要任何的人工干预。

因此，我们就需要一个类似 `daemontools或SMF` 的工具来管理ZK服务，确保进程异常退出后可以自动被重启并快速重新加入集群。

Monitoring

ZK服务可以使用如下两种方式进行监控：

- ZK提供的4个字母的命令工具
- JMX（详见ZK官网提供的JMX相关文档）

日志

ZK使用V1.2版本的Log4j作为其日志组件。ZK默认的日志配置文件位于发行版本的conf目录下。Log4j需要 `log4j.properties` 文件，这个文件要么位于ZK运行时所在的目录，要么位于classpath目录。

更多的信息见 Log4j Default Initialization Procedure 。

Troubleshooting

Server not coming up because of file corruption
A server might not be able to read its database and fail to come up because of some file corruption in the transaction logs of the ZooKeeper server. You will see some IOException on loading ZooKeeper database. In such a case, make sure all the other servers in your ensemble are up and working. Use “stat” command on the command port to see if they are in good health. After you have verified that all the other servers of the ensemble are up, you can go ahead and clean the database of the corrupt server. Delete all the files in datadir/version-2 and datalogdir/version-2/. Restart the server.

配置参数（高级配置）

ZooKeeper’s behavior is governed by the ZooKeeper configuration file. This file is designed so that the exact same file can be used by all the servers that make up a ZooKeeper server assuming the disk layouts are the same. If servers use different configuration files, care must be taken to ensure that the list of servers in all of the different configuration files match.

最小配置

Here are the minimum configuration keywords that must be defined in the configuration file:
clientPort
the port to listen for client connections; that is, the port that clients attempt to connect to.
dataDir
the location where ZooKeeper will store the in-memory database snapshots and, unless specified otherwise, the transaction log of updates to the database.

> Be careful where you put the transaction log. A dedicated transaction log device is key to consistent good performance. Putting the log on a busy device will adversely effect performance.

tickTime
the length of a single tick, which is the basic time unit used by ZooKeeper, as measured in milliseconds. It is used to regulate heartbeats, and timeouts. For example, the minimum session timeout will be two ticks.

高级配置

The configuration settings in the section are optional. You can use them to further fine tune the behaviour of your ZooKeeper servers. Some can also be set using Java system properties, generally of the form zookeeper.keyword. The exact system property, when available, is noted below.

dataLogDir
(No Java system property)
This option will direct the machine to write the transaction log to the dataLogDir rather than the dataDir. This allows a dedicated log device to be used, and helps avoid competition between logging and snaphots.

> Having a dedicated log device has a large impact on throughput and stable latencies. It is highly recommened to dedicate a log device and set dataLogDir to point to a directory on that device, and then make sure to point dataDir to a directory not residing on that device.

globalOutstandingLimit
(Java system property: zookeeper.globalOutstandingLimit.)
Clients can submit requests faster than ZooKeeper can process them, especially if there are a lot of clients. To prevent ZooKeeper from running out of memory due to queued requests, ZooKeeper will throttle clients so that there is no more than globalOutstandingLimit outstanding requests in the system. The default limit is 1,000.

ZK命令：4个字母

ZooKeeper responds to a small set of commands. Each command is composed of four letters. You issue the commands to ZooKeeper via telnet or nc, at the client port. Three of the more interesting commands: “stat” gives some general information about the server and connected clients, while “srvr” and “cons” give extended details on server and connections respectively.

conf
New in 3.3.0: Print details about serving configuration.
cons
New in 3.3.0: List full connection/session details for all clients connected to this server. Includes information on numbers of packets received/sent, session id, operation latencies, last operation performed, etc…
crst
New in 3.3.0: Reset connection/session statistics for all connections.
dump
Lists the outstanding sessions and ephemeral nodes. This only works on the leader.
envi
Print details about serving environment
ruok
Tests if server is running in a non-error state. The server will respond with imok if it is running. Otherwise it will not respond at all.
A response of “imok” does not necessarily indicate that the server has joined the quorum, just that the server process is active and bound to the specified client port. Use “stat” for details on state wrt quorum and client connection information.
srst
Reset server statistics.
srvr
New in 3.3.0: Lists full details for the server.
stat
Lists brief details for the server and connected clients.
wchs
New in 3.3.0: Lists brief information on watches for the server.
wchc
New in 3.3.0: Lists detailed information on watches for the server, by session. This outputs a list of sessions(connections) with associated watches (paths). Note, depending on the number of watches this operation may be expensive (ie impact server performance), use it carefully.
wchp
New in 3.3.0: Lists detailed information on watches for the server, by path. This outputs a list of paths (znodes) with associated sessions. Note, depending on the number of watches this operation may be expensive (ie impact server performance), use it carefully.
mntr
New in 3.4.0: Outputs a list of variables that could be used for monitoring the health of the cluster.

```
$ echo mntr | nc localhost 2185
zk_version 3.4.0
zk_avg_latency 0
zk_max_latency 0
zk_min_latency 0
zk_packets_received 70
zk_packets_sent 69
zk_outstanding_requests 0
zk_server_state leader
zk_znode_count 4
zk_watch_count 0
zk_ephemerals_count 0
zk_approximate_data_size 27
zk_followers 4 - only exposed by the Leader
zk_synced_followers 4 - only exposed by the Leader
zk_pending_syncs 0 - only exposed by the Leader
zk_open_file_descriptor_count 23 - only available on Unix platforms
zk_max_file_descriptor_count 1024 - only available on Unix platforms
```

The output is compatible with java properties format and the content may change over time (new keys added). Your scripts should expect changes.
ATTENTION: Some of the keys are platform specific and some of the keys are only exported by the Leader. The output contains multiple lines with the following format:
`key \t value`

Here’s an example of the ruok command:

```
$ echo ruok | nc 127.0.0.1 5111
imok
```

Data File Management

ZooKeeper stores its data in a data directory and its transaction log in a transaction log directory. By default these two directories are the same. The server can (and should) be configured to store the transaction log files in a separate directory than the data files. Throughput increases and latency decreases when transaction logs reside on a dedicated log devices.

The Data Directory

This directory has two files in it:

- myid - contains a single integer in human readable ASCII text that represents the server id.
- snapshot. - holds the fuzzy snapshot of a data tree.

Each ZooKeeper server has a unique id. This id is used in two places: the myid file and the configuration file. The myid file identifies the server that corresponds to the given data directory. The configuration file lists the contact information for each server identified by its server id. When a ZooKeeper server instance starts, it reads its id from the myid file and then, using that id, reads from the configuration file, looking up the port on which it should listen.

The snapshot files stored in the data directory are fuzzy snapshots in the sense that during the time the ZooKeeper server is taking the snapshot, updates are occurring to the data tree. The suffix of the snapshot file names is the zxid, the ZooKeeper transaction id, of the last committed transaction at the start of the snapshot. Thus, the snapshot includes a subset of the updates to the data tree that occurred while the snapshot was in process. The snapshot, then, may not correspond to any data tree that actually existed, and for this reason we refer to it as a fuzzy snapshot. Still, ZooKeeper can recover using this snapshot because it takes advantage of the idempotent nature of its updates. By replaying the transaction log against fuzzy snapshots ZooKeeper gets the state of the system at the end of the log.

The Log Directory

The Log Directory contains the ZooKeeper transaction logs. Before any update takes place, ZooKeeper ensures that the transaction that represents the update is written to non-volatile storage. A new log file is started each time a snapshot is begun. The log file’s suffix is the first zxid written to that log.

File Management

The format of snapshot and log files does not change between standalone ZooKeeper servers and different configurations of replicated ZooKeeper servers. Therefore, you can pull these files from a running replicated ZooKeeper server to a development machine with a standalone ZooKeeper server for trouble shooting.

Using older log and snapshot files, you can look at the previous state of ZooKeeper servers and even restore that state. The LogFormatter class allows an administrator to look at the transactions in a log.

The ZooKeeper server creates snapshot and log files, but never deletes them. The retention policy of the data and log files is implemented outside of the ZooKeeper server. The serveritself only needs the latest complete fuzzy snapshot and the log files from the start of that snapshot. See the maintenance section in this document for more details on setting a retention policy and maintenance of ZooKeeper storage.

> The data stored in these files is not encrypted. In the case of storing sensitive data in ZooKeeper,necessary measures need to be taken to prevent unauthorized access. Such measures are external to ZooKeeper (e.g., control access to the files) and depend on the individual settings in which it is being deployed.

Things to Avoid

Here are some common problems you can avoid by configuring ZooKeeper correctly:

inconsistent lists of servers
The list of ZooKeeper servers used by the clients must match the list of ZooKeeper servers that each ZooKeeper server has. Things work okay if the client list is a subset of the real list, but things will really act strange if clients have a list of ZooKeeper servers that are in different ZooKeeper clusters. Also, the server lists in each Zookeeper server configuration file should be consistent with one another.

incorrect placement of transasction log
The most performance critical part of ZooKeeper is the transaction log. ZooKeeper syncs transactions to media before it returns a response. A dedicated transaction log device is key to consistent good performance. Putting the log on a busy device will adversely effect performance. If you only have one storage device, put trace files on NFS and increase the snapshotCount; it doesn’t eliminate the problem, but it should mitigate it.

incorrect Java heap size
You should take special care to set your Java max heap size correctly. In particular, you should not create a situation in which ZooKeeper swaps to disk. The disk is death to ZooKeeper. Everything is ordered, so if processing one request swaps the disk, all other queued requests will probably do the same. the disk. DON’T SWAP.

Be conservative in your estimates: if you have 4G of RAM, do not set the Java max heap size to 6G or even 4G. For example, it is more likely you would use a 3G heap for a 4G machine, as the operating system and the cache also need memory. The best and only recommend practice for estimating the heap size your system needs is to run load tests, and then make sure you are well below the usage limit that would cause the system to swap.

Publicly accessible deployment
A ZooKeeper ensemble is expected to operate in a trusted computing environment. It is thus recommended to deploy ZooKeeper behind a firewall.

最佳实践

For best results, take note of the following list of good Zookeeper practices: For multi-tennant installations see the section detailing ZooKeeper “chroot” support, this canbe very useful when deploying many applications/services interfacing to a single ZooKeeper cluster.