TapDB 技术白皮书

V6.0 版本

深圳钛铂数据有限公司

2024

Contents

1 什么是 TapDB

TapDB 是 TapData 提供的国产分布式文档数据库；全面兼容文档数据库协议，适配国产芯片鲲鹏，海光，海思, 以及麒麟等操作系统。

1.1 文档数据库

TapDB 中的记录是一个文档，它是由字段和值对组成的数据结构。TapDB 文档类似于 JSON 对象。字段值可以包含其他文档、数组和文档数组。

使用文档的优点是：

- 文档对应于许多编程语言中的原生数据类型。
- 嵌入式文档和数组可以减少成本高昂的的连接操作。
- 动态模式支持流畅的多态性。

1.1.1 集合/视图/按需物化视图

TapDB 将文档存储在集合中。集合类似于关系数据库中的表。除了集合之外，TapDB 还支持：- 只读视图 - 按需物化视图

1.2 主要功能

1.2.1 高性能

TapDB 提供高性能数据持久性。尤其是：

- 对嵌入式数据模型的支持减少了数据库系统上的 I/O 活动。
- 索引支持更快的查询，并且可以包含嵌入式文档和数组的键。

1.2.2 查询 API

TapDB 查询 API 支持读写操作 (CRUD) 以及：

- 数据聚合
- 文本搜索和地理空间查询。

1.2.3 高可用性

TapDB 的复制工具（称为副本集）提供：

- 自动故障转移
- 数据冗余。副本集是一组维护相同数据集的 TapDB 服务器，可提供冗余并提高数据可用性。

1.2.4 横向可扩展性

TapDB 的核心功能之一是提供横向可扩展性：

- 分片将数据分布在机器集群上。
- 从 3 开始。4，TapDB 支持根据分 片键 创建数据 区域。在均衡集群中，TapDB 仅将区域覆盖的读取和写入定向到区域内的那些分片。有关更多信息，请参阅 区域 手册页。

1.2.5 支持多种存储引擎

TapDB 支持多种存储引擎：

- WiredTiger 存储引擎（包括对静态加密的支持）
- 内存存储引擎。

2 产品介绍

2.1 数据库和集合

TapDB 将数据记录存储为文档（特别是 BSON 文档），这些文档聚集在集合中。数据库存储一个或多个文档集合。

2.1.1 数据库

在 TapDB 中，数据库用于保存一个或多个文档集合。要选择要使用的数据库，请在 TapDB shell 中执行 `use <db>` 语句，如下例所示：

```
use myDB
```

2.1.1.1 创建数据库

如果数据库不存在，TapDB 会在您首次向该数据库存储数据时创建该数据库。因此，您可以切换到一个不存在的数据库，并在 TapDB shell

```
use myNewDB
db.myNewCollection1.insertOne( { x: 1 } )
```

`insertOne()` 操作会同时创建数据库 myNewDB 和集合 myNewCollection1 如果尚不存在)。确保数据库和集合名称都遵循 TapDB 命名限制。

2.1.2 集合

TapDB 将文档存储在集合中。集合类似于关系数据库中的表。

2.1.2.1 创建集合

如果集合不存在，TapDB 会在您首次存储该集合的数据时创建该集合。

```
db.myNewCollection2.insertOne( { x: 1 } )
db.myNewCollection3.createIndex( { y: 1 } )
```

如果相应的集合尚不存在，则 `insertOne()` 和 `createIndex()` 操作都会创建相应的集合。确保集合名称遵循 TapDB 命名限制。

2.1.2.2 显式创建

TapDB 提供了 `db.createCollection()` 方法来显式创建具有各种选项的集合，例如设置最大大小或文档验证规则。如果不指定这些选项，则无需显式创建集合，因为 TapDB 会在您首次存储集合数据时创建新集合。

2.1.2.3 文档验证

默认情况下，集合不要求其文档具有相同的模式，即单个集合中的文档不需要具有相同的字段集，并且在集合内的不同文档中，字段的数据类型可以不同。

从 TapDB 3 开始。2 ，但是，您可以在更新和插入操作期间对集合实施文档验证规则。

2.1.2.4 修改文档结构

若要更改集合中文档的结构，例如添加新字段、删除现有字段或将字段值更改为新类型，请将文档更新为新结构。

2.1.2.5 唯一标识符

系统为集合分配了一个不可变的 UUID。集合 UUID 在副本集的所有节点和分片集群的分片中保持不变。

要检索集合的 UUID，请运行 `listCollections` 命令或 `db.getCollectionInfos()` 方法。

2.2 文档 (Document)

TapDB 将数据记录存储为 BSON 文档。BSON 是 JSON 文档的二进制表示形式，但它包含的数据类型比 JSON 更多。有关 BSON 规范，请参阅 bsonspec.org 。另请参阅 BSON 类型。

2.2.1 文档结构

TapDB 文档由成对的字段和字段值组成，并具有以下结构：

```
{
field1: value1,
field2: value2,
field3: value3,
...
fieldN: valueN
}
```

字段的值可以是任何 BSON 数据类型，包括其他文档、数组和文档数组。例如，以下文档包含不同类型的值：

```
var mydoc = {
               _id: ObjectId("5099803df3f4948bd2f98391"),
               name: { first: "Alan", last: "Turing" },
               birth: new Date('Jun 23, 1912'),
               death: new Date('Jun 07, 1954'),
               contribs: [ "Turing machine", "Turing test", "Turingery" ],
               views : NumberLong(1250000)
             }
```

上述字段具有以下数据类型：

- _id 包含一个 ObjectId。
- name 包含一个嵌入式文档，其中包含字段 first 和 last。
- birth 和 death 保存日期类型的值。
- contribs 保存着一个字符串数组。
- views 保存 NumberLong 类型的值。

2.2.1.1 字段名称

字段名称为字符串，文档对字段名称有以下限制：

- 字段名称 _id 保留用作主键；它的值在集合中必须是唯一的、不可变的，并且可以是除数组之外的任何类型。如果 _id 包含子字段，则子字段名称不能以 ($) 符号开头。
- 字段名称不能包含 null 字符。
- 服务器允许存储包含点 (.) 和美元符号 ($) 的字段名称。
- TapDB 5.0 改进了对在字段名称中使用 ($) 和 (.) 的支持。有一些限制。有关详细信息，请参阅字段名称注意事项。

TapDB 查询语言不支持具有重复字段名称的文档。虽然某些 BSON 构建器可能支持创建具有重复字段名称的 BSON 文档，但即使插入成功或看似成功，也不支持将这些文档插入 TapDB。例如，通过 TapDB 驱动程序插入具有重复字段名称的 BSON 文档可能会导致驱动程序在插入之前静默删除重复值，或者导致插入包含重复字段的无效文档。对任何此类文档的查询都会导致任意且不一致的结果。

2.2.2 点符号

TapDB 使用点符号来访问数组的元素和访问嵌入式文档的字段。

2.2.2.1 数组

要通过从零开始的索引位置指定或访问数组的元素，请用点号 (.) 将数组名称和从零开始的索引位置连接，并用引号引起来：

```
"<array>.<index>"
```

例如，假设文档中包含以下给定字段：

```
{
    ...
    contribs: [ "Turing machine", "Turing test", "Turingery" ],
    ...
}
```

要指定 contribs 数组中的第三个元素，使用点符号 “contribs.2”。

有关查询数组的示例，请参阅：

- 查询数组
- 查询嵌入式文档数组

2.2.2.2 嵌入式文档

要使用点符号指定或访问嵌入式文档的字段，请将嵌入式文档名称与点 (.) 和字段名称连接起来，并用引号引起来：

```
<embedded document>.<field>
```

例如，假设文档中包含以下给定字段：

```
{
    ...
    name: { first: "Alan", last: "Turing" },
    contact: { phone: { type: "cell", number: "111-222-3333" } },
    ...
}
```

- 要在 name 字段中指定名为 last 的字段，请使用点符号 “name.last”。
- 要在 contact 字段中指定 phone 文档中的 number，请使用点符号 “contact.phone.number”。

> ▷ 提示：
> 分区字段不能使用包含点 (.) 的字段名称。

2.2.3 文档限制

文档具有以下属性：

2.2.3.1 文档大小限制

BSON 文档的大小限制为 16 MB。

最大文档大小有助于确保单个文档不会使用过多的 RAM，或者在传输过程中不会使用过多的带宽。要存储大于最大大小的文档，TapDB 提供了 GridFS API。

2.2.3.2 文档字段顺序

与 JavaScript 对象不同，BSON 文档中的字段为有序字段。

2.2.3.2.1 查询中的字段顺序

对于查询，字段顺序行为如下：

- 比较文档时，字段排序很重要。例如，在查询中将文档与字段 a 和 b 进行比较时：
  - `{a: 1, b: 1}` 等于 `{a: 1, b: 1}`
  - `{a: 1, b: 1}` 不等于 `{b: 1, a: 1}`
- 为了高效执行查询，查询引擎可能会在查询处理期间对字段重新排序。在其他情况下，处理这些投影操作符时可能会发生字段重新排序：$project 、$addFields 、$set unset 。
  - 字段重新排序可能发生在中间结果以及通过查询返回的最终结果中。
  - 由于某些操作可能会对字段重新排序，因此对于使用前面列出的投影操作符的查询，您不应依赖于这些查询结果中返回的特定字段排序。

2.2.3.2.2 写入操作中的字段顺序

对于写入操作，TapDB 会保留文档字段的顺序，但以下情况除外：

- _id 字段始终是文档中的第一个字段。
- 包含字段名称 renaming 的更新可能会导致文档中的字段重新排序。

2.2.3.3 _id 字段

在 TapDB 中，存储在集合中的每个文档都需要一个唯一的 _id 字段作为主键。如果插入的文档省略了 _id 字段，则 TapDB 驱动程序会自动为 ObjectId 字段生成 _id 。

这也适用于通过带有 upsert: true 的更新操作插入的文档。

_id 字段具有以下行为和约束：

- 默认情况下，TapDB 在创建集合期间会在 _id 字段上创建唯一索引。
- `_id` 字段始终是文档中的第一个字段。如果在服务器收到的文档中，`_id` 不是第一个字段，则服务器会将该字段移动到开头。
- 如果 _id 包含子字段，则子字段名称不能以 ($) 符号开头。
- _id 字段可以包含任何 BSON 数据类型的值，数组、正则表达式或未定义除外。

> ▷ 提示：
> 为了确保复制功能正常，请勿在 _id 字段中存储 BSON 正则表达式类型的值。

以下是存储 _id 值的常见选项：

- 使用 ObjectId。
- 使用自然唯一标识符（如果可用）。这样可以节省空间并避免附加索引。
- 生成一个自动递增的数字。
- 在应用程序代码中生成 UUID。为了更有效率地将 UUID 值存储在该集合和 _id 索引中，请将 UUID 存储为 BSON BinData 类型的值。

  如果满足以下条件，则 BinData 类型的索引键可以更有效地存储在索引中：

  - 二进制子类型值的范围是 0-7 或 128-135，并且
  - 字节数组的长度为：0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 16, 20, 24 或 32。
- 使用驱动程序的 BSON UUID 工具生成 UUID。请注意，驱动程序实现可能会以不同的方式实现 UUID 序列化和反序列化逻辑，这可能与其他驱动程序不完全兼容。有关 UUID 互操作性的信息，请参阅驱动程序文档。

> ▷ 提示：
> 大多数 TapDB 驱动程序客户端将包含 _id 字段并生成 ObjectId ，然后再将插入操作发送到 TapDB；但是，如果客户端发送的文档没有 _id 字段，则 TapDB 将自动添加 _id 字段并生成 ObjectId 。

2.2.4 文档结构的其他用途

除定义数据记录外，TapDB 还始终使用这种文档结构，包括但不限于：查询过滤器、更新规范文档和索引规范文档

2.2.4.1 查询筛选器文档

查询筛选器文档指定条件，确定选择哪些记录进行读取、更新和删除操作。

您可以使用 <field>:<value> 表达式指定相等条件和查询运算符表达式。

```
{
<field1>: <value1>,
<field2>: { <operator>: <value> },
...
}
```

示例请参见：

- 查询文档

- 对嵌入/嵌套文档的查询
- 查询数组
- 查询嵌入式文档数组

2.2.4.2 更新规范文档

更新规范文档使用更新操作符来指定在更新操作期间要对特定字段执行的数据修改。

```
{
<operator1>: { <field1>: <value1>, ... },
<operator2>: { <field2>: <value2>, ... },
...
}
```

有关示例，请参阅更新规范。

2.2.4.3 索引规范文档

索引规范文档定义待索引的字段和索引类型：

```
{ <field1>: <type1>, <field2>: <type2>, ...  }
```

3 安装 TapDB

3.1 Debian 上安装

本文介绍如何在 Debian Linux 平台上，通过下载的.tgz 文件安装 TapDB v6.0。

3.1.1 注意事项

在生产环境中部署 TapDB 之前，请考虑生产说明文档，其中提供了针对生产 TapDB 部署的性能注意事项和配置建议。

3.1.2 安装 TapDB

3.1.2.1 前提条件

使用以下命令安装 TapDB .tgz Tarball 所需的依赖项：

Debian 11 (Bullseye)，Debian 10 (Buster)

```
sudo apt-get install libcurl4 openssl liblzma5
```

3.1.2.2 操作步骤

请按照以下步骤从.tgz 手动安装 TapDB。

1. 联系技术支持获取安装包。
2. 通过 tar 命令解压安装包。

   ```
   tar -zxvf tapdb-linux-*-6.0.tgz
   ```

3. 确保二进制文件位于 PATH 环境变量中列出的目录下。

   TapDB 二进制文件位于 tarball 的 bin/ 目录中。您可以执行以下任一操作：

   - 将二进制文件复制到 PATH 变量中列出的目录中，例如 /usr/local/bin（根据需要使用您的安装目录来更新 /path/to/the/tapdb-directory/）

     ```
     sudo cp /path/to/the/tapdb-directory/bin/* /usr/local/bin/
     ```

   - 创建指向 PATH 变量中所列目录的二进制文件的符号链接，例如 /usr/local/bin（根据需要使用您的安装目录更新 /path/to/the/tapdb-directory/）：

     ```
     sudo ln -s  /path/to/the/tapdb-directory/bin/* /usr/local/bin/
     ```

3.1.3 运行 TapDB

3.1.3.1 ulimit 注意事项

大多数类 Unix 操作系统都会限制进程可以使用的系统资源。这些限制可能会对 TapDB 操作产生负面影响，应该进行调整。

> ▷ 提示：
> 如果打开文件数的 ulimit 值低于 64000，TapDB 会生成启动警告。

3.1.3.2 操作步骤

请按照以下步骤运行 TapDB。这些说明假设您使用的是默认设置。

1. 创建 TapDB 数据与日志目录：

```
sudo mkdir -p /var/lib/tapdb
sudo mkdir -p /var/log/tapdb
```

启动 TapDB 进程的用户必须具有对这些目录的读取和写入权限。例如，如果你打算自己运行 TapDB：

```
sudo chown `whoami` /var/lib/tap     # Or substitute another user
sudo chown `whoami` /var/log/tapdb   # Or substitute another user
```

2. 运行 TapDB。

要运行 TapDB，请在系统提示符下运行 tapdb 进程。

```
tapdb --dbpath /var/lib/tap --logpath /var/log/tapdb/tapdb.log --fork
```

有关命令行选项-dbpath 和-logpath 的详细信息，请参阅选项。

3. 验证 TapDB 是否已成功启动。

检查日志文件 /var/log/TapDB/tapdb.log 中以下行的进程输出，验证 TapDB 是否成功启动：

```
[initandlisten] waiting for connections on port 27017
```

4. 开始使用 TapDB。

tap 在与相同的主机上启动 tapdb 会话。您可以运行不带任何命令行选项的 tap，以连接到使用默认端口在本地主机上运行的 27017。

```
tap
```

3.1.3.3 默认绑定本地主机

默认情况下，TapDB 启动时会将 bindIp 设置为 127.0.0.1 ，从而绑定到本地主机网络接口。这意味着 TapDB 只能接受来自同一计算机上运行的客户端的连接。远程客户端将无法连接到 tapdb ，并且 TapDB 将无法初始化副本集，除非将此值设置为可从远程客户端访问的有效网络接口。

该值可通过以下任一方式配置：

- 在 TapDB 配置文件中使用 bindIp ，或
- 通过命令行参数 --bind_ip

在绑定到非本地主机（例如可公开访问）的 IP 地址之前，请确保您已保护集群免遭未经授权的访问。

3.2 Red Hat/CentOS 上安装

本文介绍如何在 Red Hat Enterprise Linux、CentOS Linux 或 Oracle Linux 平台上，通过下载的.tgz 文件安装 TapDB v6.0。

早期版本的部署方式，见附录章节。

3.2.1 注意事项

在生产环境中部署 TapDB 之前，请考虑生产说明文档，其中提供了针对生产 TapDB 部署的性能注意事项和配置建议。

3.2.2 安装 TapDB

3.2.2.1 前提条件

使用以下命令安装 TapDB .tgz Tarball 所需的依赖项：

```
sudo yum install libcurl openssl xz-libs
```

3.2.2.2 操作步骤

请按照以下步骤从.tgz 手动安装 TapDB。

1. 联系技术支持获取安装包。
2. 通过 tar 命令解压安装包。

   ```
   tar -zxvf tapdb-linux-*-6.0.tgz
   ```

3. 确保二进制文件位于 PATH 环境变量中列出的目录下。

   TapDB 二进制文件位于 tarball 的 bin/ 目录中。您可以执行以下任一操作：

   - 将二进制文件复制到 PATH 变量中列出的目录中，例如 /usr/local/bin（根据需要使用您的安装目录来更新 /path/to/the/tapdb-directory/）

     ```
     sudo cp /path/to/the/tapdb-directory/bin/* /usr/local/bin/
     ```

   - 创建指向 PATH 变量中所列目录的二进制文件的符号链接，例如 /usr/local/bin（根据需要使用您的安装目录更新 /path/to/the/TapDB-directory/）：

```
sudo ln -s  /path/to/the/tapdb-directory/bin/* /usr/local/bin/
```

3.2.3 运行 TapDB

3.2.3.1 前提条件

大多数类 Unix 操作系统都会限制进程可以使用的系统资源。这些限制可能会对 TapDB 操作产生负面影响，应该进行调整。

> ▷ 提示：
> 如果打开文件数的 ulimit 值低于 64000，TapDB 会生成启动警告。

3.2.3.2 目录路径

3.2.3.2.1 使用默认目录

默认情况下，TapDB 使用 tapdb 用户帐户运行，并且使用以下默认目录：

- /var/lib/tapdb（数据目录）
- /var/log/tapdb（日志目录）

创建 TapDB 数据与日志目录：

```
sudo mkdir -p /var/lib/tapdb
sudo mkdir -p /var/log/tapdb
```

默认情况下，TapDB 使用 tapdb 用户帐户运行。创建一个 tapdb 和一个 TapDB 群组。确保 tapdb 属于该群组，然后将这些目录的所有者和群组设为 tapdb：

```
sudo chown -R tapdb:tapdb /var/lib/tapdb
sudo chown -R tapdb:tapdb /var/log/tapdb
```

3.2.3.2.2 使用非默认目录

要使用除默认目录外的数据目录和/或日志目录：

1. 创建新目录。
2. 编辑配置文件 /etc/tapdb.conf 并相应修改以下字段：

- storage.dbPath 指定新的数据目录路径（例如/some/data/directory）
- systemLog.path 指定新的日志文件路径（例如/some/log/directory/tapdb.log）

3. 确保运行 TapDB 的用户有权访问这些目录：

```
sudo chown -R tapdb:tapdb <directory>
```

如果更改运行 TapDB 进程的用户，必须赋予新用户访问这些目录的权限。

4. 如果已强制执行，请配置 SELinux。

3.2.3.2.3 配置 SELinux

配置不当的 SELinux 策略可能会不安全，或者可能会阻止您的 tapdb 实例运行。

如果 SELinux 处于 enforcing（强制执行）模式，则须为 TapDB 自定义 SELinux 策略以

- 允许访问 cgroup
- 允许访问 netstat

3.2.3.3 操作步骤

请按照以下步骤在您的系统上运行 TapDB。参照这些操作说明的前提是您在使用默认设置。

1. 创建数据和日志目录。

创建 TapDB 实例存储其数据的目录。例如：

```
sudo mkdir -p /var/lib/tapdb
```

创建 TapDB 实例用于存储日志的目录。例如：

```
sudo mkdir -p /var/log/tapdb
```

启动 TapDB 进程的用户必须具有对这些目录的读取和写入权限。例如，如果你打算自己运行 TapDB：

```
sudo chown `whoami` /var/lib/tapdb    # Or substitute another user
sudo chown `whoami` /var/log/tapdb    # Or substitute another user
```

2. 运行 TapDB

要运行 TapDB，请在系统提示符下运行 tapdb 进程。

```
tapdb --dbpath /var/lib/tapdb --logpath /var/log/tapdb/tapdb.log --fork
```

有关命令行选项-dbpath 和-logpath 的详细信息，请参阅选项。

3. 验证 TapDB 是否已成功启动。

   检查日志文件 /var/log/TapDB/tapdb.log 中以下行的进程输出，验证 TapDB 是否成功启动：

   ```
   [initandlisten] waiting for connections on port 27017
   ```

4. 开始使用 TapDB。

   tap 在与相同的主机上启动 tapdb 会话。您可以运行不带任何命令行选项的 tap tapdb，以连接到使用默认端口 在本地主机上运行的 27017。

   ```
   tap
   ```

3.2.3.4 默认绑定本地主机

默认情况下，TapDB 启动时会将 bindIp 设置为 127.0.0.1 ，从而绑定到本地主机网络接口。这意味着 tapdb 只能接受来自同一计算机上运行的客户端的连接。远程客户端将无法连接到 TapDB ，并且 TapDB 将无法初始化副本集，除非将此值设置为可从远程客户端访问的有效网络接口。

该值可通过以下任一方式配置：

- 在 TapDB 配置文件中使用 bindIp ，或
- 通过命令行参数 --bind_ip

在绑定到非本地主机（例如可公开访问）的 IP 地址之前，请确保您已保护集群免遭未经授权的访问。

3.3 Ubuntu 上安装

本文介绍如何在 Ubuntu Linux LTS（长期支持）平台上，通过下载的.tgz 文件安装 TapDB v6.0。

3.3.1 注意事项

在生产环境中部署 TapDB 之前，请考虑生产说明文档，其中提供了针对生产 TapDB 部署的性能注意事项和配置建议。

3.3.2 安装 TapDB

3.3.2.1 前提条件

使用以下命令安装 TapDB Community Edition .tgz Tarball 所需的依赖项：

Ubuntu 22.04 (Jammy)

```
sudo apt-get install libcurl4 libgssapi-krb5-2 libldap-2.5-0 libwrap0
↪  libsasl2-2 libsasl2-modules libsasl2-modules-gssapi-mit snmp openssl
↪  liblzma5
```

Ubuntu 20.04 (Focal)，Ubuntu 18.04（Bionic）

```
sudo apt-get install libcurl4 openssl liblzma5
```

3.3.2.2 操作步骤

请按照以下步骤从.tgz 手动安装 TapDB。

1. 联系技术支持获取安装包。
2. 通过 tar 命令解压安装包。

   ```
   tar -zxvf tapdb-linux-*-6.0.tgz
   ```

3. 确保二进制文件位于 PATH 环境变量中列出的目录下。

   TapDB 二进制文件位于 tarball 的 bin/ 目录中。您可以执行以下任一操作：

   - 将二进制文件复制到 PATH 变量中列出的目录中，例如 /usr/local/bin（根据需要使用您的安装目录来更新 /path/to/the/tapdb-directory/）

     ```
     sudo cp /path/to/the/tapdb-directory/bin/* /usr/local/bin/
     ```

   - 创建指向 PATH 变量中所列目录的二进制文件的符号链接，例如 /usr/local/bin（根据需要使用您的安装目录更新 /path/to/the/tapdb-directory/）：

     ```
     sudo ln -s  /path/to/the/tapdb-directory/bin/* /usr/local/bin/
     ```

3.3.3 运行 TapDB

3.3.3.1 ulimit 注意事项

大多数类 Unix 操作系统都会限制进程可以使用的系统资源。这些限制可能会对 TapDB 操作产生负面影响，应该进行调整。

> ▷ 提示：
> 如果打开文件数的 ulimit 值低于 64000，TapDB 会生成启动警告。

3.3.3.2 操作步骤

请按照以下步骤运行 TapDB。这些说明假设您使用的是默认设置。

1. 创建 TapDB 数据与日志目录：

   ```
   sudo mkdir -p /var/lib/tapdb
   sudo mkdir -p /var/log/tapdb
   ```

   启动 TapDB 进程的用户必须具有对这些目录的读取和写入权限。例如，如果你打算自己运行 TapDB：

   ```
   sudo chown `whoami` /var/lib/tap     # Or substitute another user
   sudo chown `whoami` /var/log/tapdb   # Or substitute another user
   ```

2. 运行 TapDB。

   要运行 TapDB，请在系统提示符下运行 tapdb 进程。

   ```
   tapdb --dbpath /var/lib/tap --logpath /var/log/tapdb/tapdb.log --fork
   ```

   有关命令行选项-dbpath 和-logpath 的详细信息，请参阅选项。

3. 验证 TapDB 是否已成功启动。

   检查日志文件 /var/log/TapDB/tapdb.log 中以下行的进程输出，验证 TapDB 是否成功启动：

   ```
   [initandlisten] waiting for connections on port 27017
   ```

   可能会在进程输出中看到非严重警告。只要看到上述日志行，便可在 TapDB 初次计算期间安心地忽略这些警告。

4. 开始使用 TapDB。

   tap 在与 相同的主机上启动 tapdb 会话。您可以运行不带任何命令行选项的 tap，以连接到使用默认端口 在本地主机上运行的 27017 }。

   ```
   tap
   ```

3.3.3.3 默认绑定本地主机

默认情况下，TapDB 启动时会将 bindIp 设置为 127.0.0.1 ，从而绑定到本地主机网络接口。这意味着 tapdb 只能接受来自同一计算机上运行的客户端的连接。远程客户端将无法连接到 tapdb ，并且 tapdb 将无法初始化副本集，除非将此值设置为可从远程客户端访问的有效网络接口。

该值可通过以下任一方式配置：

- 在 TapDB 配置文件中使用 bindIp ，或
- 通过命令行参数-bind_ip

在绑定到非本地主机（例如可公开访问）的 IP 地址之前，请确保您已保护集群免遭未经授权的访问。

4 CRUD 操作

4.1 CRUD 操作

CRUD 操作用于创建、读取、更新和删除文档。

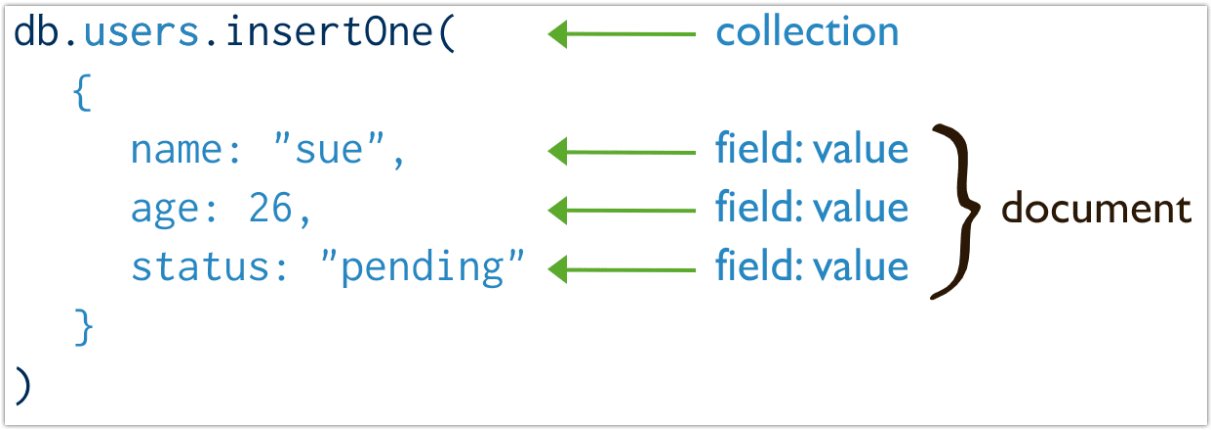
4.1.1 创建操作

创建或插入操作用于将新文档添加到集合中。如果集合当前不存在，插入操作会创建集合。

TapDB 提供以下方法将文档插入到集合中：

- `db.collection.insertOne()`
- `db.collection.insertMany()`

在 TapDB 中，插入操作针对的是单个集合，所有写入操作在单个文档级别上具有原子性。



4.1.2 读取操作

读取操作用于从集合中检索文档，即查询集合中的文档。TapDB 提供 `db.collection.find()` 方法来读取文档，您可以指定查询过滤器或条件来识别要返回的文档。

4.1.3 更新操作

更新操作用于修改集合中的现有文档，TapDB 提供以下方法来更新文档：

- db.collection.updateOne()
- db.collection.updateMany()
- db.collection.replaceOne()

在 TapDB 中，更新操作针对的是单个集合，所有写入操作在单个文档级别上具有原子性。

您可以指定条件或过滤器来识别要更新的文档，这些过滤器使用与读取操作相同的语法。

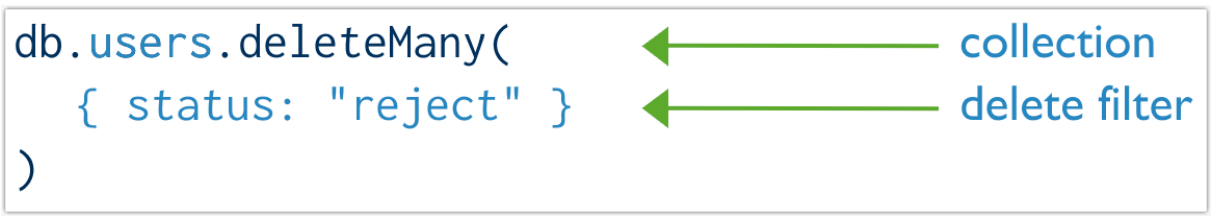
4.1.4 删除操作

删除操作用于从集合中删除文档，TapDB 提供以下方法来删除文档：

- db.collection.deleteOne()
- db.collection.deleteMany()

在 TapDB 中，删除操作针对的是单个集合，所有写入操作在单个文档级别上具有原子性。

您可以指定条件或过滤器来识别要删除的文档，这些过滤器使用与读取操作相同的语法。



4.1.5 批量写入

TapDB 提供批量执行写入操作的功能，相关信息，见批量写入操作。

4.2 批量写入

TapDB 提供批量执行写入操作的功能，允许客户端对单个集合进行多种写入操作。批量写入操作影响单个集合，并允许应用程序确定所需的可接受确认级别。

使用 `db.collection.bulkWrite()` 方法可以执行批量插入、更新和删除操作，TapDB 还支持通过 `db.collection.insertMany()` 方法进行批量插入。

4.2.1 有序/无序操作

- 有序操作：TapDB 以串行方式执行操作。如果在处理某个写入操作期间出现错误，TapDB 将停止并返回错误，不处理后续操作。
- 无序操作：TapDB 可以并行执行操作，但不保证这种行为。如果在处理某个写入操作期间出现错误，TapDB 将继续处理列表中剩余的写入操作。

在分片集合上执行有序操作列表通常比无序操作列表慢，因为每个有序操作都必须等待前一个操作完成。

默认情况下，`bulkWrite()` 执行有序操作。要指定无序写入操作，请在选项文档中设置 `ordered: false`。

4.2.2 bulkWrite () 方法

`bulkWrite()` 支持以下写入操作：

- insertOne
- updateOne
- updateMany
- replaceOne
- deleteOne
- deleteMany

每个写操作都会作为数组中的一个文档传递给 `bulkWrite()`。

4.2.3 示例

本节的示例使用 pizzas 集合：

```
db.pizzas.insertMany( [
    { _id: 0, type: "pepperoni", size: "small", price: 4 },
    { _id: 1, type: "cheese", size: "medium", price: 7 },
    { _id: 2, type: "vegan", size: "large", price: 8 }
] )
```

以下 bulkWrite() 示例对 pizzas 集合执行以下操作：

- 使用 insertOne 添加两个文档。
- 使用 updateOne 更新一个文档。
- 使用 deleteOne 删除一个文档。
- 使用 replaceOne 替换一个文档。

```
try {
    db.pizzas.bulkWrite( [
        { insertOne: { document: { _id: 3, type: "beef", size: "medium",
↪   price: 6 } } },
        { insertOne: { document: { _id: 4, type: "sausage", size: "large",
↪   price: 10 } } },
        { updateOne: {
            filter: { type: "cheese" },
            update: { $set: { price: 8 } }
        } },
        { deleteOne: { filter: { type: "pepperoni"} } },
        { replaceOne: {
            filter: { type: "vegan" },
            replacement: { type: "tofu", size: "small", price: 4 }
        } }
    ] )
} catch( error ) {
    print( error )
}
```

输出示例，包括已完成操作的摘要：

```
{
    acknowledged: true,
    insertedCount: 2,
    insertedIds: { '0': 3, '1': 4 },
    matchedCount: 2,
    modifiedCount: 2,
    deletedCount: 1,
    upsertedCount: 0,
    upsertedIds: {}
}
```

4.2.4 批量插入分片集合的策略

大批量插入操作（包括初始数据插入或例行数据导入）可能会影响分片集群的性能。对于批量插入，请考虑以下策略：

4.2.4.1 预拆分集合

如果分片集合为空，此集合将只有一个初始数据段，此数据段位于单个分片上。随后，TapDB 必须花一些时间接收数据、创建分割以及将分割的数据段分发给可用的分片。为了避免这一性能开销，可以预先分割集合，如分割分片集群中的数据段。

4.2.4.2 对 taps

要提高分片集群的写入性能，请使用 `bulkWrite()` 并将可选参数 `ordered` 设置为 `false`。taps 可以尝试将写入操作同时发送到多个分片。对于空集合，先按照分片集群中的拆分数据段。

4.2.4.3 避免单调限速

如果分片键在插入期间单调增加，则所有已插入数据都会进入集合中的最后一个数据段，该数据段将始终出现在单个分片上。因此，集群的插入容量永远不会超过该单个分片的插入容量。

如果插入量大于单个分片可以处理的容量，并且无法避免分片键的单调增加，则可以考虑对应用程序进行以下修改：

- 反转分片键的二进制位。这样将保留信息，并避免将插入顺序与递增的值序列相关联。
- 交换第一个和最后一个 16 位字，“随机打乱”插入。

以下示例（采用 C++ 编写）交换生成的 BSON ObjectId 的前导和尾随 16 位字，以使其不再单调递增。

```
using namespace tap;
    OID make_an_id() {
    OID x = OID::gen();
    const unsigned char *p = x.getData();
    swap( (unsigned short&) p[0], (unsigned short&) p[10] );
    return x;
}
void foo() {
    // create an object
    BSONObj o = BSON( "_id" << make_an_id() << "x" << 3 << "name" << "jane" );
    // now we may insert o into a sharded collection
}
```

▷ 提示：

另请参阅分片键以了解有关选择分片键的信息。

5 聚合操作

聚合操作处理多个文档并返回计算结果。您可以使用聚合操作进行以下操作：

- 组合多个文档中的值。
- 对分组数据执行操作，返回单一结果。
- 分析一段时间内的数据变化。

若要执行聚合操作，您可以使用：

- 聚合管道：这是执行聚合的首选方法，允许进行复杂的数据处理。
- 单一目的聚合方法：这些方法操作简单，但功能不如聚合管道丰富。

5.1 聚合管道

聚合管道由一个或多个处理文档的阶段组成：

- 每个阶段都对输入文档执行一个操作。例如，某个阶段可以过滤文档、对文档进行分组并计算值。
- 从一个阶段输出的文档会传递到下一阶段。
- 聚合管道可以返回对文档组的综合结果，例如总值、平均值、最大值和最小值。

如使用通过聚合管道更新

> ▷ 提示：
> 使用 db.collection.aggregate() 方法运行的聚合管道不会修改集合中的文档，除非管道包含 $merge 或 $out 阶段。

5.1.1 聚合管道示例

以下聚合管道示例包含两个阶段，并返回按披萨名称分组后各款中号披萨的总订单数量：

```
db.orders.aggregate( [
    // Stage 1: Filter pizza order documents by pizza size
    {
        $match: { size: "medium" }
    },
    // Stage 2: Group remaining documents by pizza name and calculate total
     ↪  quantity
    {
        $group: { _id: "$name", totalQuantity: { $sum: "$quantity" } }
    }
] )
```

$match 阶段：

- 过滤出尺寸为中号的披萨订单文档。
- 将过滤后的文档传递到下一阶段 $group。

$group 阶段：

- 按披萨名称对文档进行分组。
- 使用 $sum 计算每种披萨的总订单数量，并将总数存储在聚合结果的 totalQuantity 字段中。

5.2 单一目的聚合方法

单一目的聚合方法用于聚合集合中的文档。这些方法简单易用，但功能相对有限。

方法	说明
db.collection.estimatedDocumentCount()	返回集合或视图中文档的近似数量。
db.collection.count()	返回集合或视图中文档的数量。
db.collection.distinct()	返回具有指定字段的不同值的文档数组。

6 索引

索引在 TapDB 中用以支持高效查询。如果没有索引，TapDB 必须扫描集合中的每个文档才能返回查询结果；而有了适当的索引，TapDB 可以限制扫描的文档数，从而提高查询性能。虽然索引可以提升查询性能，但添加索引会影响写入操作的性能，因为每次插入操作都必须更新所有相关的索引，因此在高写入率的集合需要评估其索引更新成本。

6.1 用例

如果应用程序对相同字段重复运行查询，则可以为这些字段创建索引以提高性能。例如，以下是一些常见的场景及其对应的索引类型：

场景	索引类型
人力资源部门经常通过员工 ID 查找员工信息。您可以在员工 ID 字段上创建索引以提高查询性能。	单字段索引
销售人员经常按位置查找客户信息，位置存储在一个嵌入式对象中，包括 state、city 和 zipcode 等字段。您可以在 location 对象上创建索引。	单字段索引在嵌入式文档上
杂货店经理按名称和数量查找库存商品，以确定哪些商品库存不足。您可以同时为 item 和 quantity 字段创建复合索引。	复合索引

> 提示：
> 在嵌入式文档上创建索引时，仅对整个嵌入式文档的查询使用该索引。对文档中特定字段的查询不使用该索引。

6.2 开始体验

您可以使用驱动程序方法或 TapDB Shell 创建和管理索引。

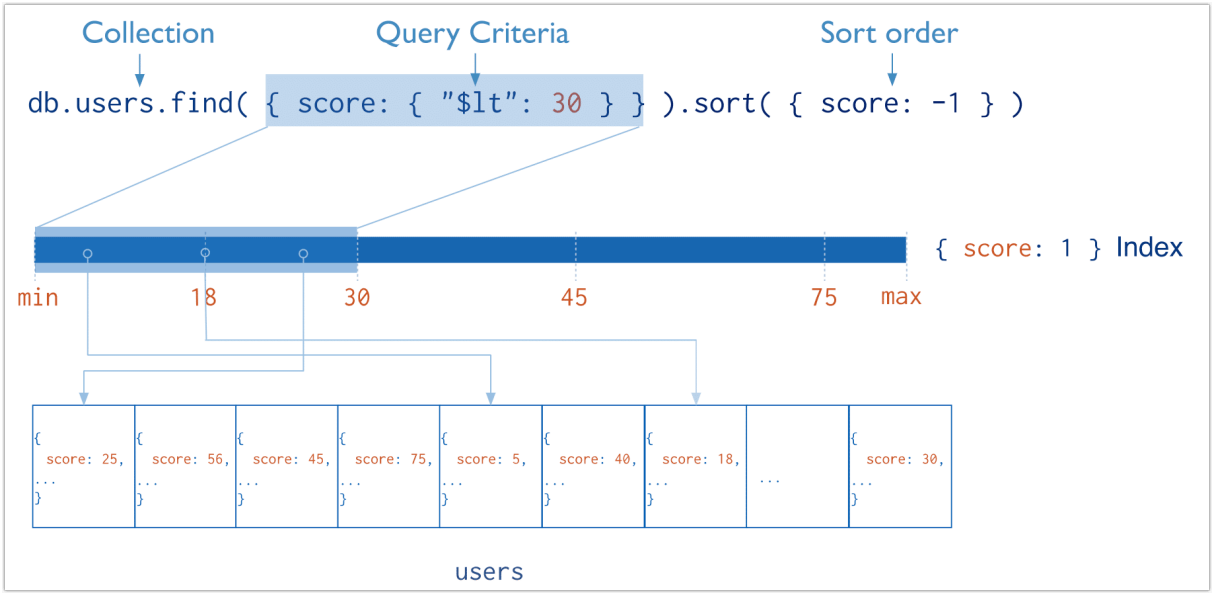
6.2.1 创建和管理索引

您可以使用驱动程序方法或 TapDB Shell 来创建和管理索引。

6.3 详情

索引是一种特殊的数据结构，它以易于遍历的形式存储一小部分集合数据。TapDB 索引使用 B-Tree 数据结构。索引存储特定字段或多个字段的值，并按字段值排序。这种排序支持高效的等值匹配和基于范围的查询操作。此外，TapDB 还可以使用索引顺序来返回排序后的结果。

下图说明了使用索引来选择匹配文档并为其排序的查询：



TapDB 中的索引与其他数据库系统中的索引类似，您可以在集合级别定义索引，也可以对集合中文档的任何字段或子字段建立索引。

6.4 默认 _id 索引

TapDB 在创建集合时会自动对 _id 字段创建唯一索引，该索引可防止插入两个具有相同 _id 字段值的文档，且不能删除此索引。

> ▷ 提示：
> 在分片集群中，如果不使用 _id 字段作为分片键，应用程序必须确保 _id 字段中值的唯一性，以防止出错，通常可以使用自动生成的标准 ObjectId 实现。

6.5 创建索引

要在 TapDB Shell 中创建索引，请使用 db.collection.createIndex() 方法。

```
db.collection.createIndex( <key and index type specification>, <options> )
```

以下示例在 name 字段上创建单键降序索引：

```
db.collection.createIndex( { name: -1 } )
```

db.collection.createIndex() 方法仅在相同规范的索引不存在时创建索引。

6.5.1 索引名称

索引的默认名称是索引键和索引中每个键方向（即 1 或 -1）的连接，使用下划线作为分隔符。例如，在 { item: 1, quantity: -1 } 上创建的索引的名称为 item_1_quantity_-1。

您可以使用自定义名称创建索引，从而创建具有业务意义的索引名称。下述案例中，通过 createIndex() 方法在 item 和 quantity 上创建一个名为 query for inventory 的索引：

```
db.products.createIndex(
    { item: 1, quantity: -1 } ,
    { name: "query for inventory" }
)
```

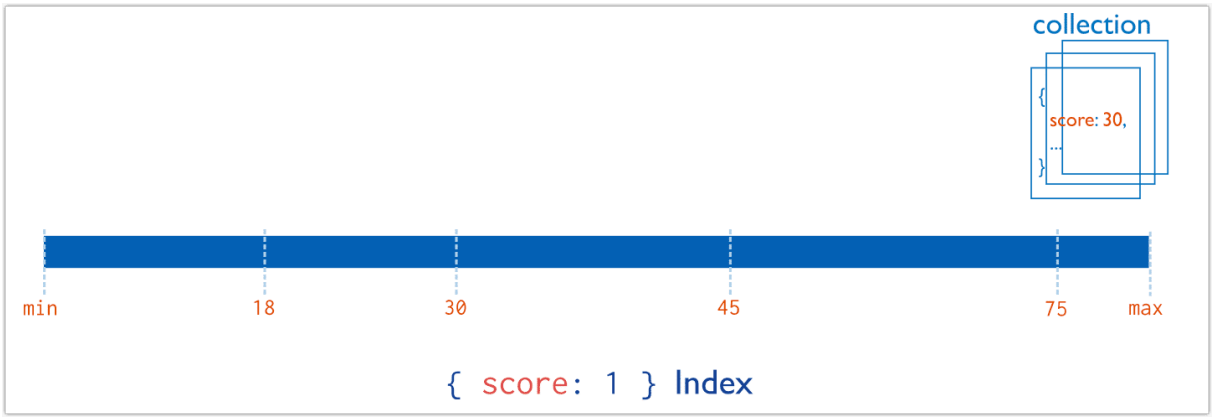
您可以使用 db.collection.getIndexes() 方法查看索引名称，索引一旦创建就无法重命名。

6.6 索引类型

TapDB 提供了许多不同的索引类型来支持特定的数据和查询类型。

6.6.1 单个字段

除了 TapDB 定义的 `_id` 索引外，TapDB 还支持对文档的单个字段创建用户定义的升序/降序索引。

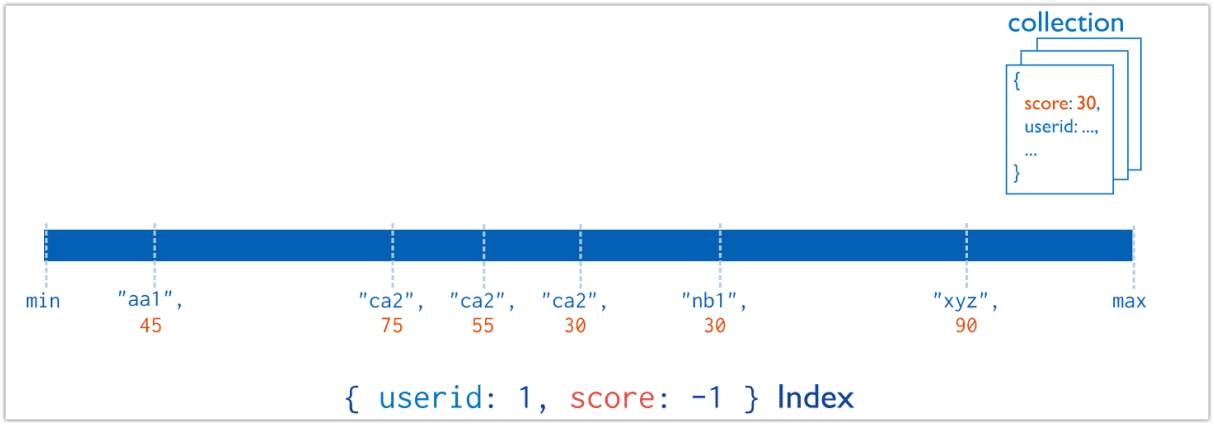


对于单字段索引和排序操作，索引键的排序顺序（即升序或降序）并不重要，因为 TapDB 可以沿任一方向遍历索引。

6.6.2 复合索引

TapDB 还支持针对多个字段的用户自定义索引，即复合索引。

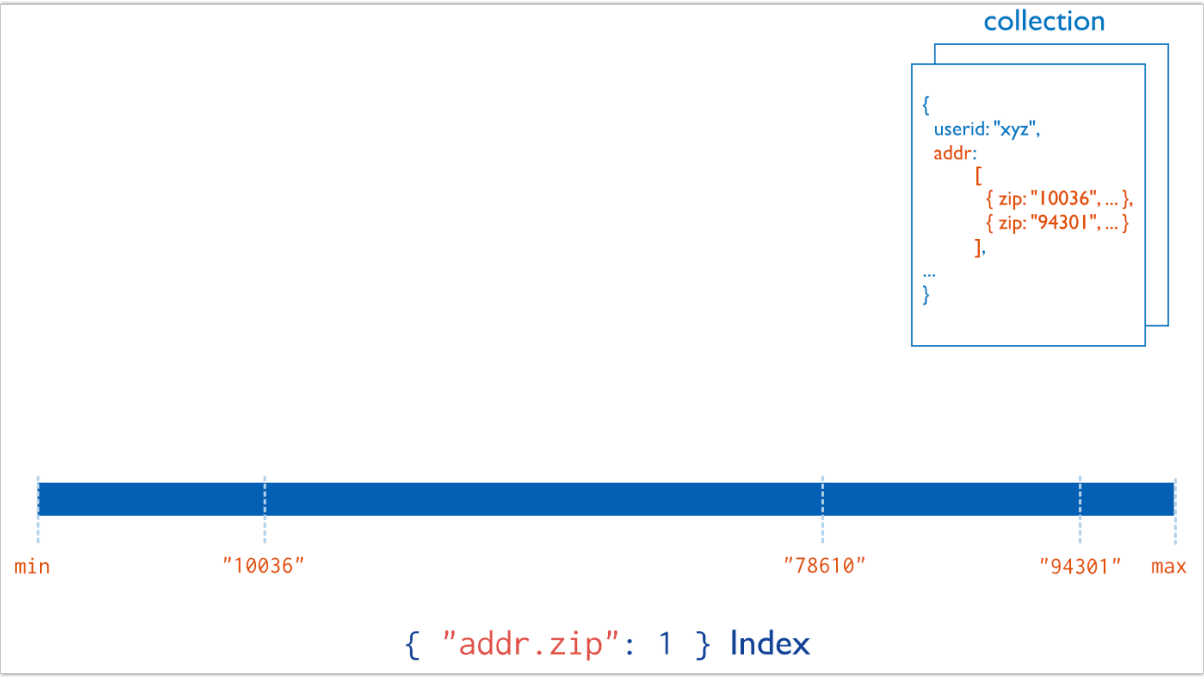
复合索引中列出的字段顺序很重要。例如，复合索引 `{ userid: 1, score: -1 }` 按 `userid` 排序，然后在每个 `userid` 值中按 `score` 排序。



对于复合索引和排序操作，排序顺序（即 升序或降序）可以决定索引是否可以支持排序操作。

6.6.3 多键索引

TapDB 使用多键索引来索引存储在数组中的内容。如果对保存数组值的字段建立索引，TapDB 会为数组的每个唯一元素创建单独的索引项。多键索引允许查询通过匹配数组的一个或多个元素来选择包含数组的文档。如果索引字段包含数组值，TapDB 会自动判断是否创建多键索引，无需显式指定多键类型。



6.6.4 地理空间索引

TapDB 提供了两种特殊索引来支持地理空间坐标数据的高效查询：返回结果时使用平面几何的 2d 索引和使用球面几何返回结果的 2dsphere 索引。

该索引为稀疏索引。如果要使用稀疏索引创建复合索引，请先查看使用稀疏复合索引的特殊注意事项。

6.6.5 文本搜索索引

TapDB 提供 text 索引类型，以支持在集合中搜索字符串内容。要了解有关文本索引的更多信息，请参阅文本索引。

该索引为稀疏索引。如果要使用稀疏索引创建复合索引，请先查看使用稀疏复合索引的特殊注意事项。

6.6.6 哈希索引

TapDB 提供 hashed 索引类型，用于支持基于哈希的分片。这些索引在其范围内的值分布更随机，但仅支持等值匹配，不支持基于范围的查询。

6.6.7 聚集索引

从 TapDB 5.3 开始，您可以使用集群索引创建集合。使用集群索引创建的集合称为“集群化集合”。

6.6.8 通配符索引

从 TapDB 4.2 开始，您可以使用通配符索引支持对多个字段、任意字段或未知字段的查询。创建通配符索引时，使用 $** 表示所有字段或所有值，以便使用单个命令对以下任何内容进行索引：

- 字段的所有值
- 文档中的所有字段
- 文档中除特定字段路径（Field Path）之外的所有字段
- 文档中的多个特定字段
- 要了解更多信息，请参阅通配符索引。

6.7 索引属性

6.7.1 唯一索引

唯一索引确保 TapDB 拒绝索引字段的重复值。除唯一约束外，唯一索引在功能上与其他 TapDB 索引类似。

6.7.2 部分索引

部分索引仅对符合指定过滤表达式的文档进行索引，降低存储要求和索引创建与维护的性能成本。

部分索引提供了比稀疏索引更全面的功能，因此应优先选择部分索引。

6.7.3 稀疏索引

稀疏索引确保索引仅包含具有索引字段的文档条目，跳过没有索引字段的文档。

您可以将稀疏索引选项与唯一索引选项结合使用，以防止插入索引字段具有重复值的文档，并跳过索引缺少索引字段的文档。

6.7.4 TTL 索引

TTL 索引是一种特殊索引，TapDB 可以使用它在一定时间后自动删除集合中的文档。这适用于仅需在数据库中保留有限时间的数据，如日志和会话信息。

6.7.5 隐藏索引

从 4.4 版本开始，隐藏索引对查询规划器不可见，不用于支持查询。隐藏索引允许评估删除索引的潜在影响，索引在隐藏期间保持完全维护，一旦取消隐藏即刻可用。

除 `_id` 索引外，您可以隐藏任何索引。

6.8 索引使用

索引可以提高读取操作的效率。分析查询性能教程提供了带索引和不带索引的查询执行统计信息示例。

6.9 索引和排序规则

排序规则允许用户为字符串比较指定特定于语言的规则，例如字母大小写和重音符号规则。要使用索引进行字符串比较，操作还必须指定相同的排序规则。否则，索引将无法支持该操作。

> ▷ 提示：
> 由于配置了排序规则的索引是通过 ICU 排序规则键来实现排序，因此，相比未配置排序规则的索引的索引键，有排序规则感知的索引键可能会更大。

例如，集合 `myColl` 在字符串字段 `category` 上具有一个索引，排序规则语言环境为 `"fr"`：

```
db.myColl.createIndex( { category: 1 }, { collation: { locale: "fr" } } )
```

以下查询操作指定了与索引相同的排序规则，因此可以使用索引：

```
db.myColl.find( { category: "cafe" } ).collation( { locale: "fr" } )
```

而以下查询操作默认使用“简易的”二进制排序器，因此无法使用索引：

```
db.myColl.find( { category: "cafe" } )
```

如果复合索引的前缀键不是字符串、数组或嵌入式文档，即使查询操作指定了与索引不同的排序规则，它仍然可以利用该复合索引来支持对其前缀键的比较。

例如，集合 `myColl` 在数值字段 `score` 和 `price` 以及字符串字段 `category` 上有一个复合索引；该索引使用排序规则语言环境 `"fr"` 创建，用于进行字符串比较：

```
db.myColl.createIndex(
    { score: 1, price: 1, category: 1 },
    { collation: { locale: "fr" } } )
```

以下操作使用 "simple" 二进制排序规则进行字符串比较，它们可以使用索引：

```
db.myColl.find( { score: 5 } ).sort( { price: 1 } )
db.myColl.find( { score: 5, price: { $gt: NumberDecimal( "10" ) } } ).sort(
↪  { price: 1 } )
```

以下操作使用 "simple" 二进制排序规则对 category 字段进行字符串比较，但只能利用索引的 score: 5 部分：

```
db.myColl.find( { score: 5, category: "cafe" } )
```

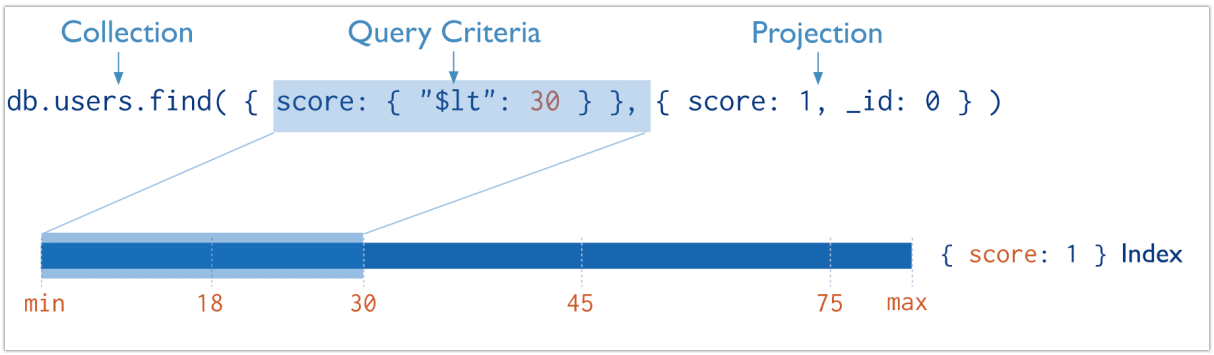
> ▷ 提示：
> 与文档键（包括嵌入式文档键）的匹配使用简单的二进制比较。这意味着类似 “foo.bár” 的键的查询不会匹配 “foo.bar” 键，无论您为 strength 参数设置了什么值。

以下索引只支持简单的二进制比较，不支持排序规则

- 文本索引，
- 2d 索引，以及
- geoHaystack 索引。

6.10 覆盖查询

当查询条件和查询投影仅包含索引字段时，TapDB 会直接从索引返回结果，而无需扫描任何文档或将文档调入内存。这些覆盖查询可以非常高效。



6.11 索引并集

TapDB 可以使用索引的交集来完成查询。对于指定复合查询条件的查询，如果一个索引可以满足查询条件的一部分，而另一个索引可以满足查询条件的另一部分，那么 TapDB 就可以使用这两个索引的交集来完成查询。使用复合索引和使用索引交集的效率取决于具体的查询和系统。

6.12 限制

索引适用某些限制，例如索引键的长度或每个 collection 的索引数量。

6.13 其他注意事项

虽然索引可以提高查询性能，但索引也会带来一些操作注意事项。

在索引构建期间，应用程序可能会遇到性能下降或集合的读/写权限受限的问题。

有关索引构建过程的更多信息，请参阅填充集合上的索引构建文档，特别是复制环境中的索引构建部分。

某些驱动程序使用 `NumberLong(1)` 而不是 `1` 来指定索引顺序，生成的索引是相同的。

7 事务

在 TapDB 中，操作单个文档是具有原子性的。您可以使用嵌入式文档和数组来捕获单个文档结构中的数据关系，而无需跨多个文档和集合进行标准化。这种单文档原子性在很多情况下消除了使用分布式事务的必要性。

对于需要对多个文档（可能跨多个集合）的操作具有原子性的情况，TapDB 支持多文档事务。您可以使用分布式事务来跨多个操作、集合、数据库、文档和分片进行操作。

7.1 事务 API

下面的示例展示了事务 API 的关键组件，特别是回调 API。回调 API 的功能包括：

- 启动事务
- 执行指定操作
- 提交结果（或在出错时中止）

回调 API 包含特定错误的重试逻辑。例如，服务器在遇到 `TransientTransactionError` 或 `UnknownTransactionCommitResult` 错误后会重试事务。从 TapDB 6.0 开始，服务器在遇到 `TransactionTooLargeForCache` 错误时不会重试事务。

▷ 提示：

- 使用适合您的 TapDB 版本的 TapDB 驱动程序。
- 使用驱动程序时，事务中的每个操作都必须将会话传递给每个操作。
- 事务中的操作使用事务级读关注、事务级写关注和事务级读取偏好。
- 您可以在事务中隐式或显式创建集合。请参阅在事务中创建集合和索引。

7.2 事务和原子性

对于需要对多个文档（可能跨多个集合）进行原子性读取和写入的情况，TapDB 支持分布式事务，包括副本集和分片集群上的事务。

分布式事务的原子性表现为：

- 事务要么应用所有数据更改，要么回滚所有更改。
- 在事务提交时，事务中所做的所有数据更改都会保存并变得可见。在事务外部，事务提交前的更改不可见。

  不过，当事务写入多个分片时，并非所有外部读取操作都需等待已提交事务的结果在各个分片上可见。例如，如果事务已提交并且写入 1 在分片 A 上可见，但写入 2 在分片 B 上尚不可见，则读关注 `"local"` 处的外部读取可以在不看到写入 2 的情况下读取写入 1 的结果。
- 事务中止后，事务中所做的所有数据更改会被丢弃且不可见。例如，如果事务中的任何操作失败，事务就会中止，事务中所做的所有数据更改将被丢弃且不可见。

▷ 提示：

与单文档写入操作相比，分布式事务通常具有更高的性能开销，因此应优先考虑有效的模式设计。非规范化的数据模型（嵌入式文档和数组）在很多情况下仍是最佳选择。适当的数据建模将最大限度地减少对分布式事务的需求。有关其他事务使用注意事项（如运行时间限制和 oplog 大小限制），另请参阅生产注意事项

7.3 事务和操作

分布式事务可以跨多个操作、集合、数据库、文档和分片使用。

- 在事务中可以创建集合和索引。
- 事务使用的集合可以位于不同的数据库中。

▷ 提示：

不能在跨分片写事务中创建新集合。例如，如果在一个分片中写入现有集合，并在另一个分片中隐式创建集合，TapDB 将无法在同一事务中执行这两个操作。

- 不能写入固定大小集合。
- 从集合读取时不能使用读关注。(从 TapDB 5.0 开始)
- 不能在 `config`、`admin` 或 `local` 数据库中读取/写入集合。
- 不能写入 `system.*` 集合。
- 不能使用 `explain` 或类似命令返回受支持操作的查询计划。
- 对于在 ACID 事务外部创建的游标，无法在 ACID 事务内部调用 `getMore`。
- 对于在事务中创建的游标，无法在事务外部调用 `getMore`。
- 您不能将 killCursors 指定为事务
- 有关事务中不支持的操作列表，请参阅限制操作。

> 提示：
> 在启动事务之前创建或删除集合时，如果在事务内访问该集合，请发出带有写关注 `"majority"` 的创建或删除操作，以确保事务可以获取所需的锁。

7.3.1 在事务中创建集合和索引

如果事务不是跨分片写入事务，则可以在分布式事务中执行以下操作：

- 创建集合。
- 在同一事务中创建的新空集合上创建索引。

在事务中创建集合时：

- 您可以隐式创建一个集合，例如：
  - 对不存在的集合进行插入操作，或
- 对不存在的集合使用 `update/findAndModify` 操作进行 `upsert: true`。
- 可以使用命令或其辅助程序 `db.createCollection()` 显式创建集合。

在事务内创建索引时，要创建的索引必须位于以下位置之一：

- 不存在的集合。集合作为操作的一部分创建。
- 同一事务中创建的新空集合。

您还可以对现有索引运行 `db.collection.createIndex()` 和 `db.collection.createIndexes()` 以检查其是否存在。这些操作成功返回而不创建索引。

7.3.1.1 限制

- 您无法在跨分片写事务中创建新集合。例如，如果您在一个分片中写入一个现有集合，并在另一个分片中隐式创建一个集合，TapDB 将无法在同一事务中执行这两个操作。
- 当以分片集合为目标时，您无法在事务中使用 $graphLookup 阶段。
- 要在事务内显式创建集合或索引，事务读关注级别必须为 "local"

要显式创建集合和索引，请使用以下命令和方法：

命令	方法
create	db.createCollection()
createIndexes	db.collection.createIndex(), db.collection.createIndexes()

7.3.2 计数操作

要在事务内执行计数操作，请使用 $count 聚合阶段或 $group（带有 $sum 表达式）聚合阶段。

TapDB 驱动程序提供集合级 API countDocuments(filter, options) 作为辅助方法，该方法使用 $group 和 $sum 表达式来执行计数。

TapDB shell 提供 db.collection.countDocuments() 辅助方法，该方法使用 $group 和 $sum 表达式进行计数。

7.3.3 去重操作

如要在事务中执行不同的操作：

- 对于未分片集合，可以使用 db.collection.distinct() 方法或 distinct 命令，以及带有 $group 阶段的聚合管道。
- 对于分片集合，不能使用 db.collection.distinct() 方法或 distinct 命令。要查找分片集合的不同值，请改用带有 $group 阶段的聚合管道。例如：
  - 不使用 db.coll.distinct("x")，而是使用

```
db.coll.aggregate([
    { $group: { _id: null, distinctValues: { $addToSet: "$x" } } },
    { $project: { _id: 0 } }
])
```

- 不使用 db.coll.distinct("x", { status: "A" }), 而是使用

```
db.coll.aggregate([
    { $match: { status: "A" } },
    { $group: { _id: null, distinctValues: { $addToSet: "$x" } } },
    { $project: { _id: 0 } }
])
```

管道返回一个指向文档的游标 { "distinctValues" : [2, 3, 1] }, 迭代游标以访问结果文档。

7.3.4 信息操作

事务中允许使用诸如 hello、buildInfo、connectionStatus（及其辅助方法）之类的信息命令，但这些命令不能是事务中的第一项操作。

7.3.5 限制性操作

事务中不允许执行以下操作：

- 在跨分片写事务中创建新集合。例如，如果在一个分片中写入现有集合，并在另一个分片中隐式创建集合，TapDB 将无法在同一事务中执行这两项操作。
- 使用除 local 之外的读关注级别创建集合（如 db.createCollection()）和索引（如 db.collection.createIndex()）。
- 使用 listCollections 和 listIndexes 命令及其辅助方法。
- 执行非 CRUD 和非信息性操作（如 createUser、getParameter 和 count）及其辅助程序。

7.4 事务和会话

- 事务与会话关联。
- 一个会话一次最多可以具有一个未结事务。
- 使用驱动程序时，事务中的每项操作都必须与会话关联。详见驱动程序特定文档。
- 如果会话结束且具有未结事务，该事务将中止。

7.5 读关注/写关注/读取偏好

7.5.1 事务和读取偏好

事务中的操作使用事务级读取偏好。使用驱动程序时，您可以在事务启动时设置事务级读取偏好：

- 如果未设置事务级读取偏好，事务将使用会话级读取偏好。
- 如果未设置事务级和会话级读取偏好，事务将使用客户端级读取偏好。默认情况下，客户端级读取偏好为 `primary`。

分布式事务包含读取操作时，必须使用读取偏好 `primary`。

7.5.2 事务和读关注

事务中的操作使用事务级读关注。集合和数据库级别设置的任何读关注在事务中都会被忽略。

如果未设置事务级和会话级读关注，事务级读关注默认为客户端级读关注。默认情况下，对于主节点上的读取，客户端级读关注是 `"local"`。

事务支持以下读关注级别：

`"local"`

- 读关注 `"local"` 返回节点中可用的最新数据，但可以回滚。
- 对于分片集群上的事务，读关注 `"local"` 无法保证数据来自跨分片的同一快照视图。如果需要快照隔离，请使用读关注 `"snapshot"`。
- 您可以在事务中创建集合和索引。要显式创建集合或索引，事务必须使用读关注 `"local"`。隐式创建集合时，可以使用任何事务支持的读关注。

`"majority"`

- 如果事务以写关注 `"majority"` 提交，读关注 `"majority"` 返回多数副本集节点确认且不可回滚的数据。否则，读关注 `"majority"` 不保证读取多数提交的数据。
- 对于分片集群上的事务，读关注 `"majority"` 无法保证数据来自跨分片的同一快照视图。如果需要快照隔离，请使用读关注 `"snapshot"`。

`"snapshot"`

- 事务使用提交，则读关注会从多数已提交数据的快照中返回数据。
- 如果事务不使用写关注 “majority” 提交，则 `"snapshot"` 读关注不保证读操作会使用大多数已提交数据的快照。
- 对于分片集群上的事务，数据的 `"snapshot"` 视图会在各分片之间同步。

7.5.3 事务和写关注

事务使用事务级写关注来提交写入操作。事务内的写入操作必须在没有明确写关注规范的情况下执行，并须使用默认的写关注。在提交时，使用事务级写关注来提交写入。

> 提示：
> 请勿为事务中的各个写入操作显式设置写关注。为事务内的各个写入操作设置写关注会返回错误消息。

您可以在事务启动时设置事务级写关注。

- 如果未设置事务级写关注，事务级写关注默认为会话级写关注。
- 如果未设置事务级和会话级写关注，事务级写关注默认为客户端级写关注。
  - `w: "majority"`（在 TapDB 5.0 及更高版本中），包含仲裁节点的部署有所不同。详见隐式默认写关注。
  - `w: 1`

事务支持所有写关注 w 值，包括：

`w: 1`

- 写关注 `w: 1` 会在提交应用于主节点后返回确认信息。

  :::tip

  使用 `w: 1` 提交时，如果发生故障转移，则可以回滚

  :::

- 使用 `w: 1` 写入关注提交时，事务级 `"majority"` 读关注无法保证事务中的读操作会读取大多数已提交数据。
- 使用 `w: 1` 写关注提交时，事务级 `"snapshot"` 读关注无法保证事务中的读操作会使用大多数已提交数据的快照。

`w: "majority"`

- 在将提交应用于大多数投票节点后，写关注 `w: "majority"` 会返回确认消息。
- 使用 `w: "majority"` 写关注提交时，事务级 `"majority"` 读关注可以保证操作已读取大多数已提交数据。对于分片集群上的事务，大多数已提交数据的视图不会在各分片之间同步。
- 使用 `w: "majority"` 写关注提交时，事务级 `"snapshot"` 读关注可以保证操作已从大多数已提交数据的同步快照中读取。

> 提示：
> 无论为事务指定了何种写关注，分片集群事务的提交操作始终包括一些使用 {w: "majority", j: true} 写关注的部分。
> 服务器参数 coordinateCommitReturnImmediatelyAfterPersistingDecision 控制何时将事务提交决策返回给客户端。该参数在 MongoDB 5.0 中引入，默认值为 true。在 TapDB 6.0 和 5.0.10 中，默认值更改为 false，即分片事务协调器会等待所有成员确认多文档事务提交，然后再将提交决策返回给客户端。
> 如果为 "majority" 多文档事务指定写关注，且该事务无法复制到计算出的多数副本集成员，则该事务可能不会立即回滚副本集成员。副本集最终将保持一致。事务始终会在所有副本集节点上应用或回滚。
> 无论为事务指定了何种写关注，驱动程序在重试 commitTransaction 时都会应用 w: "majority"。

7.6 基本信息

以下各节将介绍有关事务的更多注意事项。

7.6.1 仲裁节点

如果任何事务操作读取或写入包含仲裁节点的分片，跨多个分片的写事务将出现错误并中止。

7.6.2 分片配置限制

您无法在具有将 writeConcernMajorityJournalDefault 设置为 false 的分片（例如具有使用内存中存储引擎的投票节点的分片）的分片集群上运行事务。

> 提示：
> 无论为事务指定了何种写关注，分片集群事务的提交操作始终包括一些使用 {w: "majority", j: true} 写关注的部分。

7.6.3 诊断

要获取事务状态和指标，请使用以下方法：

源	返回
db.serverStatus() 方法 serverStatus 命令	返回事务指标。

源	返回
$currentOp 聚合管道	返回：$currentOp.transaction 如果操作属于事务的一部分。有关在事务中持有锁的非活动会话的信息。$currentOp.twoPhaseCommitCoordinator 涉及写入多个分片的分片事务的指标。
db.currentOp() 方法 currentOp 命令	返回：currentOp.transaction 如果操作属于事务的一部分。currentOp.twoPhaseCommitCoordinator 涉及写入多个分片的分片事务的指标。
tapdb 和 taps 日志消息	在 TXN 日志组件中包含有关慢速事务（即超过 operationProfiling.slowOpThresholdMs 阈值的事务）的信息。

7.6.4 特征兼容性版本 (FCV)

要使用事务，所有部署节点的 featureCompatibilityVersion 必须至少为：

部署	最低 featureCompatibilityVersion
副本集 (Replica Set)	`4.0`
分片集群	`4.2`

要检查成员的 FCV，请连接到该成员并运行以下命令：

```
db.adminCommand( { getParameter: 1, featureCompatibilityVersion: 1 } )
```

更多信息，请参阅 `setFeatureCompatibilityVersion` 参考页。

7.6.5 存储引擎

分布式事务副本集和分片集群支持的

- 主节点使用 WiredTiger 存储引擎，而
- 从节点使用 WiredTiger 存储引擎或内存存储引擎。

> ▷ 提示：
> 您无法在具有将 `writeConcernMajorityJournalDefault` 设置为 `false` 的分片上运行事务。

7.6.6 限制关键部分等待时间

从 TapDB 5.2（和 5.0.4）开始：

- 当查询访问分片时，数据段迁移或 DDL 操作可能会占用集合的关键部分。
- 要限制分片在事务中等待关键部分的时间，请使用 `metadataRefreshInTransactionMaxWaitBehindCritSecMS` 参数。

8 复制

TapDB 中的副本集是一组维护相同数据集的 TapDB 进程。副本集提供冗余和高可用性，是所有生产部署的基础。本节介绍 TapDB 中的副本及其组件和架构，还提供与副本集相关的常见任务教程。

8.1 冗余和数据可用性

复制提供冗余并提高数据可用性。通过在不同的数据库服务器上设置数据副本，复制为单个数据库服务器的故障提供了一定程度的容错能力。

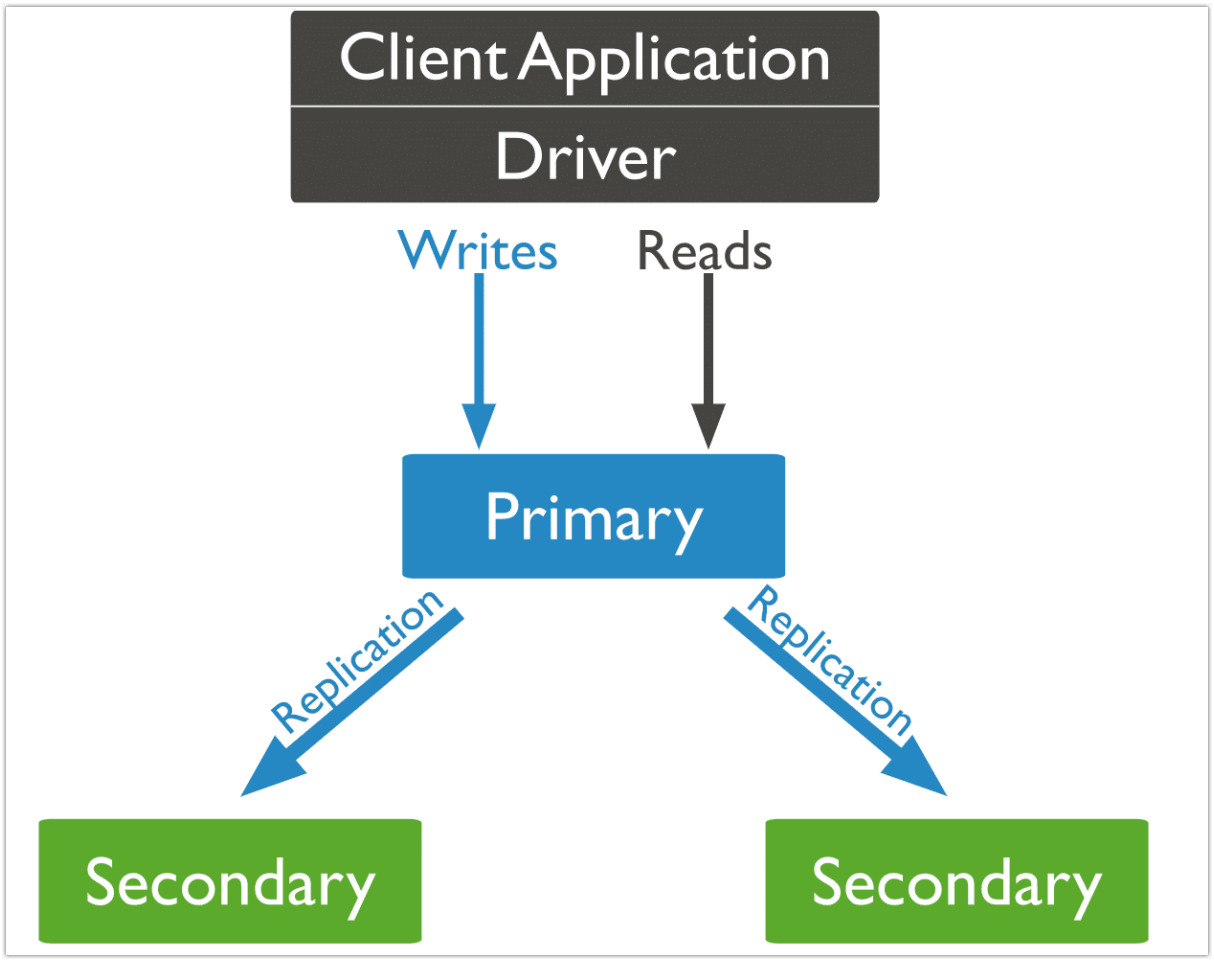
在某些情况下，复制还可以增加读取容量，因为客户端可以将读取请求发送到不同的服务器。在不同数据中心维护数据副本可以提高分布式应用程序的数据局部性和可用性。您还可以根据特定用途维护额外的副本，如灾难恢复、报告或备份。

8.2 TapDB 中的复制

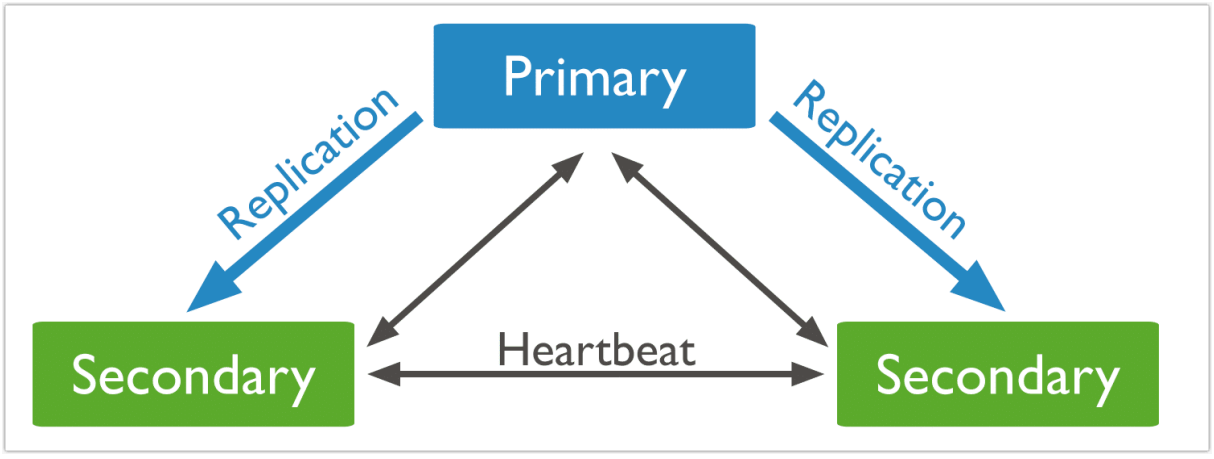
副本集是一组维护相同数据集的 TapDB 实例。副本集包含多个数据承载节点和一个可选的仲裁节点。在数据承载节点中，只有一个主节点，其他为从节点。

> ▷ 提示：
> 每个副本集节点必须且只能属于一个副本集。副本集节点不能属于多个副本集。

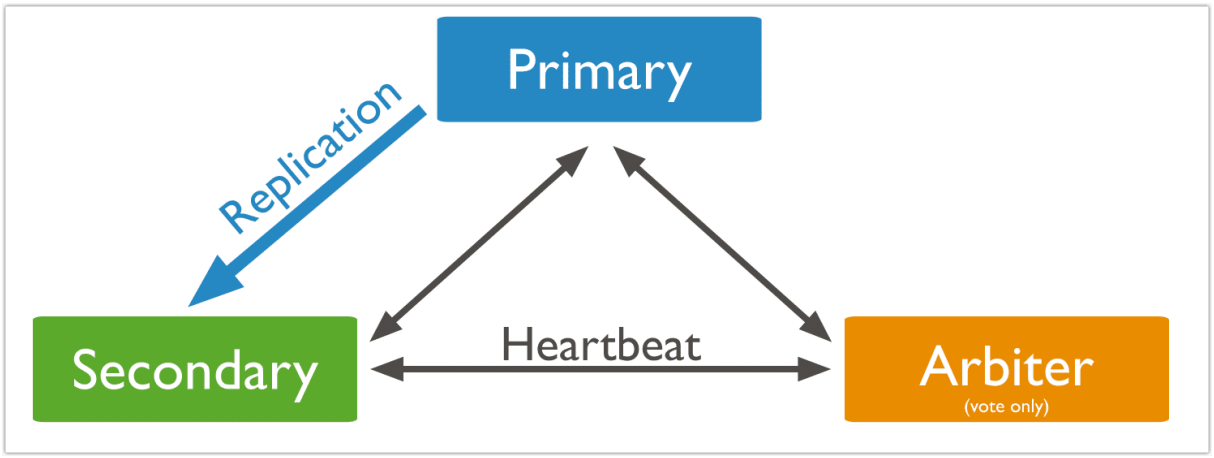
主节点接收所有写入操作。副本集中只能有一个主节点使用 `{ w: "majority" }` 写关注级别对写入请求进行确认；尽管在某些情况下，另一个 TapDB 实例可能会暂时将自身视为主节点。主节点在其操作日志（即 oplog）中记录对其数据集的所有更改。



从节点复制主节点的 oplog，并将这些操作应用于其数据集，使其数据集反映主节点的状态。如果主节点不可用，则某个符合条件的从节点将被选举为新的主节点。



在某些情况下（如只有一个主节点和一个从节点，但因成本限制无法添加更多从节点），可以选择将一个 TapDB 实例作为仲裁节点加入副本集。仲裁节点参与选举，但不持有数据（即不提供数据冗余）。



在选举期间，仲裁节点始终是仲裁节点，而主节点可能会降级为从节点，从节点也可能被选为主节点。

8.3 异步复制

从节点会复制主节点的 oplog 并异步应用于其数据集。通过使从节点的数据集反映主节点的数据状态，即使一个或多个节点出现故障，副本集也可继续运行。

8.3.1 慢操作

现在，副本集的从节点会记录应用时间超过慢操作阈值的 oplog 条目，这些慢 oplog 消息：

- 记录在诊断日志中
- 记录在 REPL 组件下，包含 `applied op: <oplog entry> took <num>ms` 文本
- 不依赖日志级别（系统级别或组件级别）
- 不依赖于分析级别。
- 受 slowOpSampleRate 限制

分析器不会捕获慢 oplog 条目。

8.3.2 复制延迟和流量控制

复制延迟是指主节点上的操作与将该操作从 oplog 应用到从节点之间的延迟。一些小的延迟是可以接受的，但随着复制延迟的增加，会出现严重的问题，包括在主节点上创建缓存压力。

从 TapDB 4.2 开始，管理员可以限制主节点应用写入的速率，目标是将 `majority committed` 延迟保持在可配置的最大 `flowControlTargetLagSeconds` 值以下。

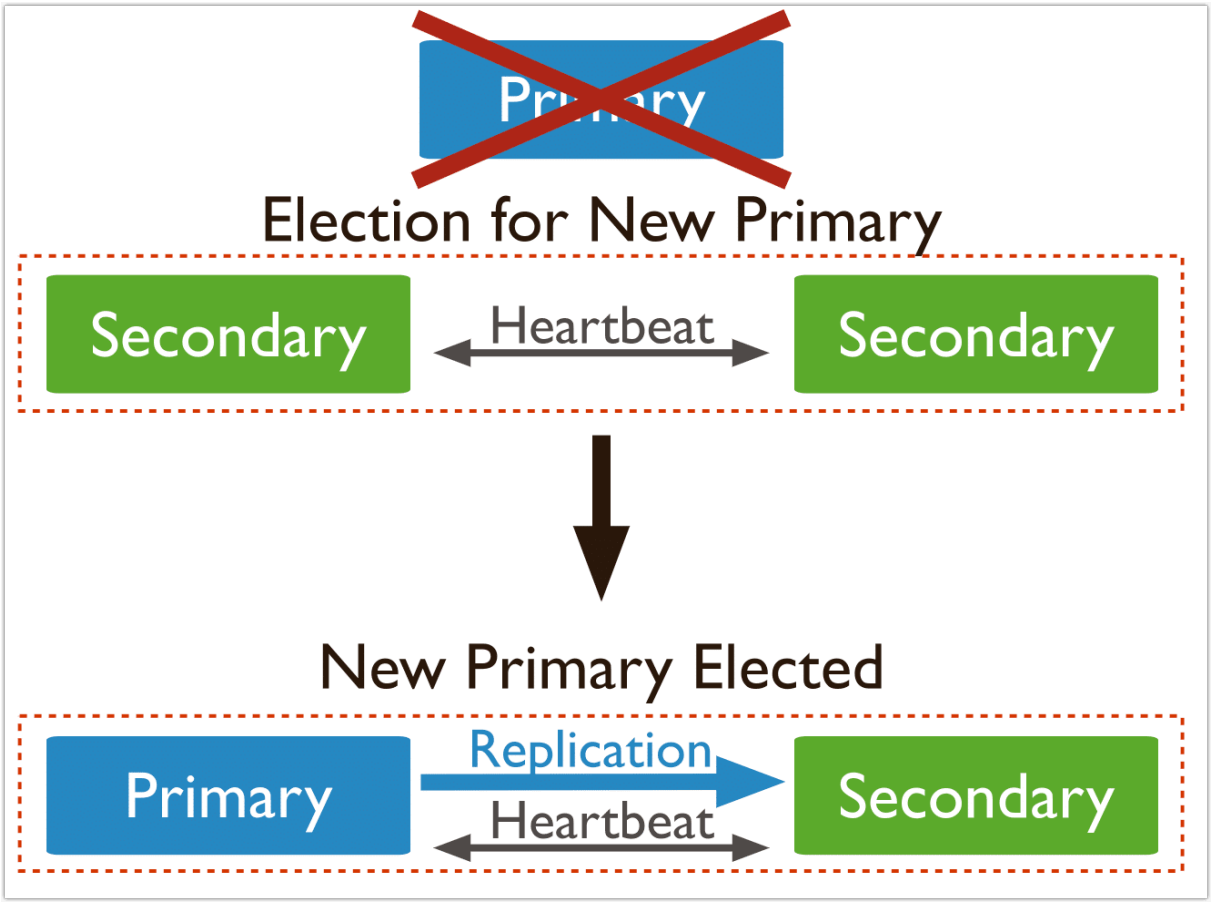
默认情况下，流量控制是启用的。

> 提示：
> 要启用流量控制，副本集/分片集群的 featureCompatibilityVersion (fCV) 必须为 4.2，且读关注（read concern）为 majority。

启用流量控制后，随着延迟接近 flowControlTargetLagSeconds，主节点上的写入操作必须先获取票据，然后才能获取锁以应用写入操作。通过限制每秒发出的票据数量，流量控制机制将尝试将延迟保持在目标延迟以下。

8.4 自动故障转移

当主节点在超过配置的 electionTimeoutMillis 时间段（默认 10 秒）内未与副本集中的其他节点通信时，一个符合条件的从节点将发起选举，并提名自己成为新的主节点。集群将尝试完成新主节点的选举并恢复正常运转。



在成功完成选举之前，副本集无法处理写操作。如果将读取查询配置为在主节点离线时在从节点上运行，那么副本集可以继续为读取查询提供服务。

假设采用默认的副本配置设置，集群选举新主节点之前的平均时间通常不应超过 12 秒。这包括将主节点标记为不可用以及召集和完成选举所需的时间。可以通过修改 settings.electionTimeoutMillis 复制配置选项来调整

该时间段。网络延迟等因素可能会延长副本集选举完成所需的时间，这反过来又会影响集群在没有主节点的情况下运行的时间。这些因素取决于具体集群架构。

将 `electionTimeoutMillis` 复制配置选项从默认的 `10000`（10 秒）降低，可以更快地检测到主节点故障。然而，由于临时网络延迟等因素，即使主节点在其他方面是健康的，集群也可能会更频繁地进行选举。这可能导致 `w: 1` 写入操作的回滚次数增加。

您的应用程序连接逻辑应包括对自动故障转移和后续选举的容忍度。TapDB 驱动程序可检测到主节点丢失，并一次性自动重试某些写入操作，从而为自动故障转移和选举提供额外的内置处理功能：

兼容的驱动程序默认启用可重试写入。

TapDB 提供镜像读功能，用最近访问的数据预热可选从节点缓存。预热从节点的缓存有助于在选举后更快地恢复性能。

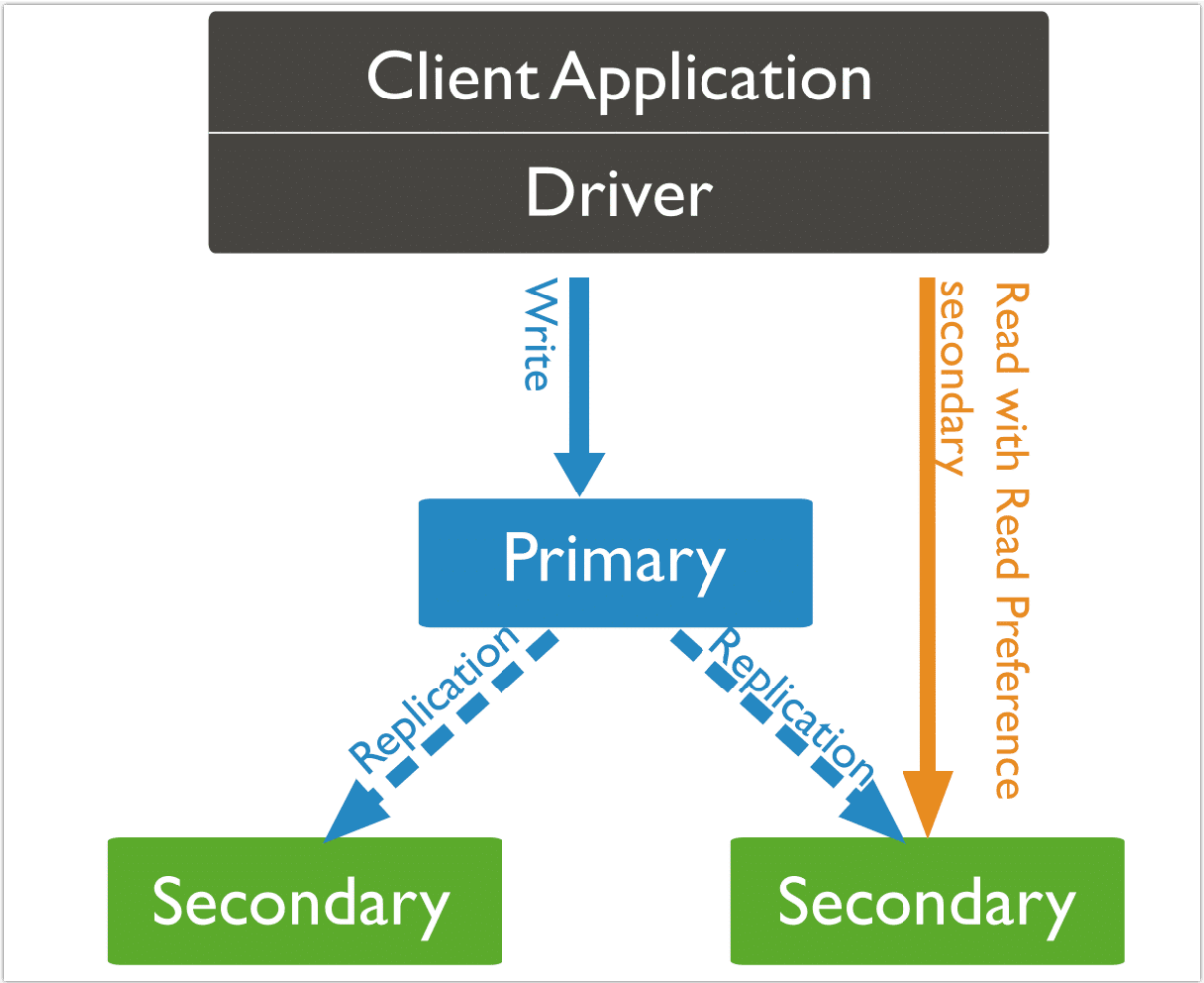
要了解有关 TapDB 故障转移过程的更多信息，请参阅：

- 副本集选举
- 可重试写入
- 副本集故障转移期间的回滚

8.5 读取操作

8.5.1 读取偏好

默认情况下，客户端从主节点读取；但是，客户端可以指定读取偏好以向从节点发送读取操作。



异步复制到从节点意味着在从节点读取到的数据可能不会反映主节点上的最新状态。

分布式事务包含读操作的 `primary` 必须使用读取偏好

8.5.2 数据可见性

根据读关注，客户端可在写入操作持久化后读取数据：

- 无论写入操作的写关注级别如何，使用 `"local"` 或 `"available"` 读关注的其他客户端均可在写入操作被发起它的客户端确认之前，看到该操作的结果。
- 使用 `"local"` 或 `"available"` 读关注的客户端可读取数据，而这些数据可能会在副本集故障转移期间进行回滚。

对于多文档事务中的操作，当事务提交时，该事务中进行的所有数据更改都将保存并在事务外部可见。换言之，一个事务不会在回滚其他事务的同时提交某些更改。

在事务提交前，事务中的数据更改在事务外不可见。

不过，当事务写入多个分片时，并非所有外部读取操作都需等待已提交事务的结果在各个分片上可见。例如，如果事务已提交并且写入 1 在分片 A 上可见，但写入 2 在分片 B 上尚不可见，则读关注 "local" 处的外部读取可以在不看到写入 2 的情况下读取写入 1 的结果。

有关 TapDB 读取隔离性、一致性和新近度的更多信息，请参阅读取隔离性、一致性和新近度。

8.5.3 镜像读取

镜像读取可减少由于中断或计划维护后主节点选举对系统的影响。在副本集发生故障转移后，接管成为新主节点的从节点会在新查询请求传入时更新其缓存。在缓存预热期间，性能可能会受到影响。

镜像读会预热 `electable` 从节点副本集成员的缓存。为了预热可选举从节点的缓存，主节点会将其接收到的受支持操作的示例镜像到可选举的从节点。

可以使用 `mirrorReads` 参数来配置接收镜像读的 `electable` 从节点副本集节点的子集的大小。

> ▷ 提示：
> 镜像读取不会影响主节点对客户端的响应。主节点镜像复制到从节点的读取是“发后即忘”类型的操作。主节点不会等待响应。

8.5.4 支持的操作

镜像读取支持以下操作：

- count
- distinct
- find
- findAndModify （具体来说，过滤器将作为一次镜像读取请求发送）
- update （具体来说，过滤器将作为一次镜像读取请求发送）

8.5.5 启用/禁用镜像读取

默认情况下启用镜像读取并使用默认的采样率 (0.01)。要禁用镜像读取，请将 `mirrorReads` 参数设置为 `{ samplingRate: 0.0 }`：

```
db.adminCommand( {
    setParameter: 1,
    mirrorReads: { samplingRate: 0.0 }
} )
```

当采样率大于 0.0 时，主节点将把支持的读取操作镜像复制到一部分 electable 的从节点。当采样率为 0.01 时，主节点会将它接收到且受支持的读取操作的百分之一镜像复制到可参与选举的从节点。

例如，对于一个包含一个主节点和两个从节点的副本集，如果主节点接收到 1000 个可镜像的操作且采样率为 0.01，主节点会将大约 10 个操作镜像到从节点，每个从节点接收一小部分操作。主节点将每个镜像读取请求随机发送到可用的从节点。

8.5.6 更改镜像读取采样率

要更改镜像读取的采样率，请将 mirrorReads 参数设为 0.0 和 1.0 之间的数字：

- 采样率设置为 0.0 将禁用镜像读。
- 采样率介于 0.0 和 1.0 之间时，主节点会以指定的采样率向可参选从节点随机转发支持的读取操作。
- 采样率 1.0 会导致主节点将所有支持的读取操作转发到可参选从节点。

8.5.7 镜像读取指标

如果您在下列操作中指定以下字段，serverStatus 命令和 db.serverStatus() Shell 方法会返回 mirroredReads 指标：

```
db.serverStatus( { mirroredReads: 1 } )
```

8.6 事务

从 TapDB 4.0 开始，多文档事务可用于副本集。

分布式事务包含读操作的 primary 必须使用读取偏好。

在事务进行提交前，在事务中所做的数据更改在事务外不可见。

不过，当事务写入多个分片时，并非所有外部读取操作都需等待已提交事务的结果在各个分片上可见。例如，如果事务已提交并且写入 1 在分片 A 上可见，但写入 2 在分片 B 上尚不可见，则读关注 "local" 的外部读取可以在不看到写入 2 的情况下读取写入 1 的结果。

8.7 变更流

从 TapDB 3.6 开始，变更流（Change Streams）可用于副本集和分片集群。变更流允许应用程序访问实时数据变化，而无需实时跟踪 oplog，这可能相当复杂且容易出错。应用程序可以使用变更流来订阅一个或多个集合上的所有数据变更。

8.8 其他功能

副本集提供多个选项来支持应用程序需求。例如，可以部署一个跨多个数据中心的副本集，或通过调整某些节点的 `members[n].priority` 来控制选举结果。副本集还支持用于报告、灾难恢复或备份功能的专用节点。

请参阅优先级为 0 的副本集节点、隐藏副本集节点和延时副本集节点以了解更多信息。

在某些情况下，副本集中的两个节点可能会暂时都认为自己是主节点，但最多只有一个节点能完成 `{ w: "majority" }` 写关注的写入操作。能够完成 `{ w: "majority" }` 写入操作的节点是当前的主节点，另一个节点是尚未意识到自己已降级（通常由于网络分区）的前主节点。发生这种情况时，即使已请求了读取偏好为 `primary`，连接到前主节点的客户端依然可能观察到过时的数据，并且对前主节点的新的写入操作最终将被回滚。

9 分片

分片是一种跨多台机器分布数据的方法。TapDB 通过分片支持超大数据集和高吞吐量操作的部署。

大型数据集或高吞吐量应用程序可能对单个服务器的容量构成挑战。例如，较高的查询速率可能会耗尽服务器的 CPU 容量，而大于系统 RAM 的工作集大小会对磁盘驱动器的 I/O 容量造成压力。

解决系统增长问题有两种方法：垂直扩展和水平扩展。

- 垂直扩展（Vertical Scaling）：增加单个服务器的容量，例如使用更强大的 CPU、添加更多 RAM 或增加存储空间。但单台机器的硬件限制和云提供商的配置上限，使得垂直扩展存在实际的最大值。
- 水平扩展（Horizontal Scaling）：将系统数据集和负载划分到多台服务器上，并按需增加服务器以提高容量。虽然单台机器的容量可能有限，但多台机器一起处理工作负载，往往比单台高性能服务器更高效。水平扩展增加了基础设施和维护的复杂性，但扩展的成本通常低于高端硬件的成本。

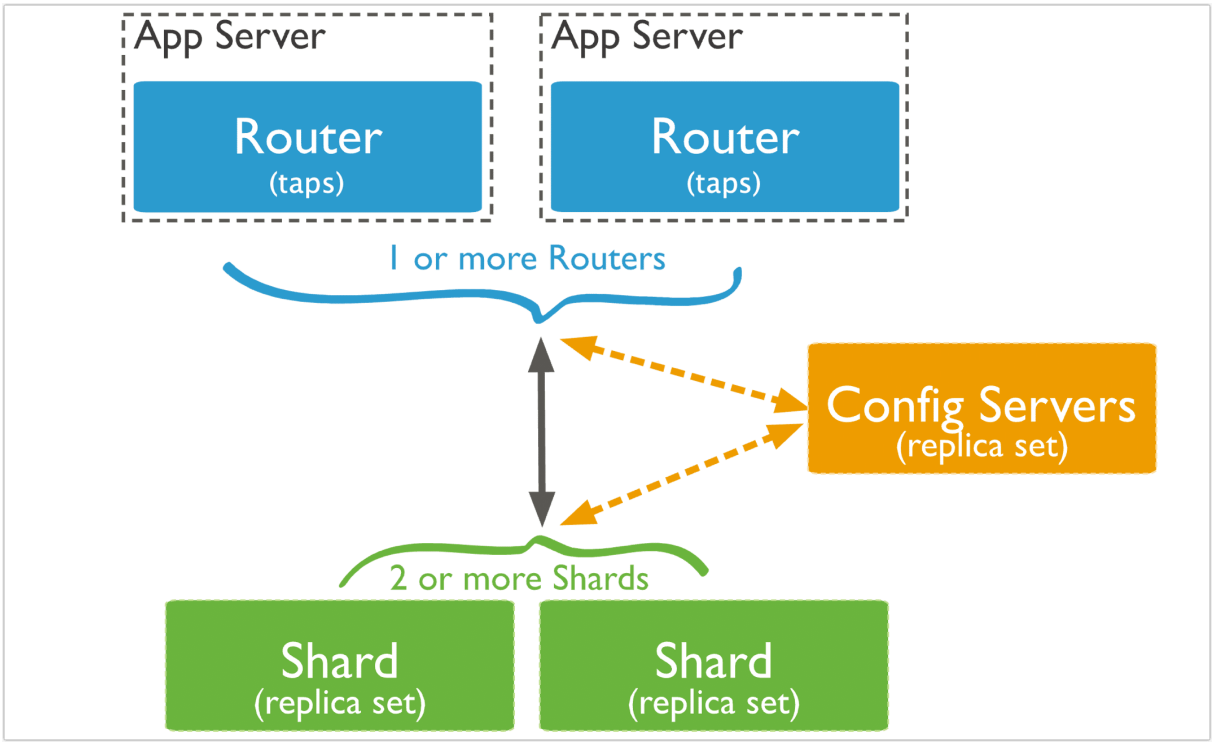
TapDB 支持通过水平扩展进行分片。

9.1 分片集群

TapDB 分片集群由以下组件构成：

- 分片：每个分片都包含分片数据的一个子集，每个分片都必须作为一个副本集。
- **taps**：充当查询路由器，在客户端应用程序和分片集群之间提供接口。taps 支持对冲读，从而最大限度地降低延迟。
- 配置服务器：存储集群的元数据和配置设置。从 TapDB 3.4 开始，配置服务器必须作为副本集（CSRS）部署。

下图描述了分片集群内各组件之间的交互：



TapDB 在集合级别对数据进行分片，将集合数据分布到集群中的各个分片上。

9.2 分片键

TapDB 使用分片键在分片之间分发集合的文档。分片键由文档中的一个或多个字段组成。

分片集合中的文档可能缺少分片键字段。跨分片分发文档时，缺少的分片键字段将视为具有 null 值，但在路由查询时则不会。

从 TapDB 5.0 开始，您可以通过更改集合的分片键对集合重新分片，还可以通过向现有分片键添加字段来优化分片键。

分片键值决定了文档在分片中的分布。您可以更新文档的分片键值，除非分片键字段是不可变的 `_id` 字段。

9.2.1 分片键索引

对已填充的集合进行分片，该集合必须具有以分片键开头的索引。对空集合进行分片时，如果该集合没有指定分片键的索引，TapDB 会创建支持索引。

9.2.2 分片键策略

分片键的选择影响分片集群的性能、效率和可扩展性。即使硬件和基础架构最佳，选择不当的分片键也会造成瓶颈。分片键及其索引的选择也会影响集群可使用的分片策略。

9.3 块

TapDB 将分片数据划分为多个块，每个块都有一个基于分片键的范围（包含下限和排除上限）。

9.4 平衡器和均匀数据分布

为了使集群中所有分片的数据均匀分布，平衡器在后台运行，以跨分片迁移数据范围。

9.5 分片的优点

9.5.1 读/写

TapDB 将读写工作负载分布在各个分片上，使每个分片处理一部分操作。通过增加分片，可以跨集群水平扩展读写工作负载。

对于包含分片键或复合分片键前缀的查询，taps 可以将查询定位到特定分片。这些有针对性的操作比向集群中的每个分片广播更高效。

taps 支持对冲读取以最大限度地减少延迟。

9.5.2 存储容量

分片将数据分布在各个分片上，使每个分片包含整个集群数据的一个子集。随着数据集的增长，额外的分片增加了集群的存储容量。

9.5.3 高可用性

将配置服务器和分片部署为副本集可以提高可用性。

即使一个或多个分片副本集不可用，分片集群仍可继续执行部分读取和写入。虽然无法访问不可用分片上的数据，但可用分片上的读取或写入仍可成功。

9.6 分片前的注意事项

分片集群基础设施要求和复杂性需要仔细规划、执行和维护。

一旦集合被分片，TapDB 不提供取消分片的方法。虽然稍后可以重新分片集合，但务必仔细考虑分片键的选择，以避免可扩展性和性能问题。

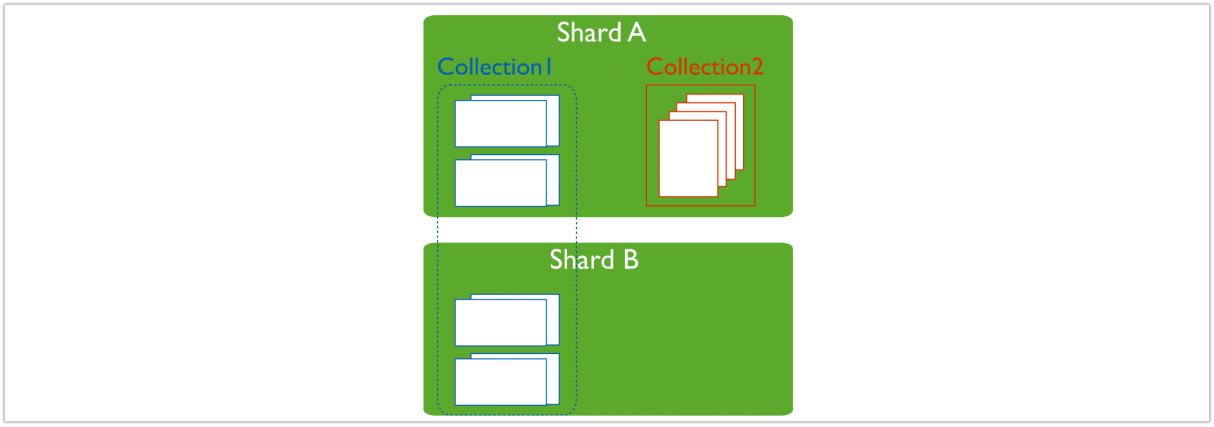
如果查询不包含分片键或复合分片键的前缀，taps 执行广播操作，查询分片集群中的所有分片。这些分散/聚集查询可能是长时间运行的操作。

从 TapDB 5.1 开始，启动、重新启动或添加具有集群范围写关注 (CWWC) 的分片服务器时，必须设置 `sh.addShard()`。

如果 CWWC 未设置，并且分片配置为默认写入关注 `{ w: 1 }`，则分片服务器将无法启动或添加，并返回错误。

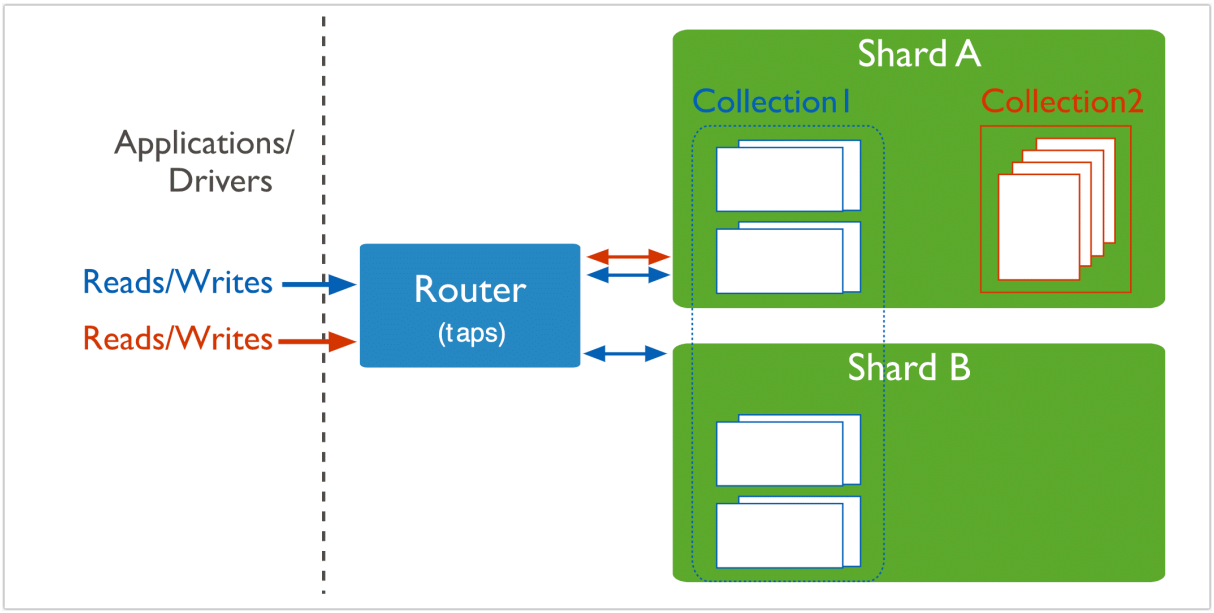
9.7 分片集合和非分片集合

数据库可以混合使用分片集合和非分片集合。分片集合被分区并分布在集群中的分片上。非分片集合存储在主分片上。每个数据库都有自己的主分片。



9.8 连接到分片集群

您必须连接到 taps 路由器才能与分片集群中的任何集合进行交互。这包括分片集合和非分片集合。客户端不应连接到单个分片来执行读取或写入操作。



您可以使用 TapDB shell 连接到 TapDB 实例的相同方式，或使用 TapDB 驱动程序连接到 taps。

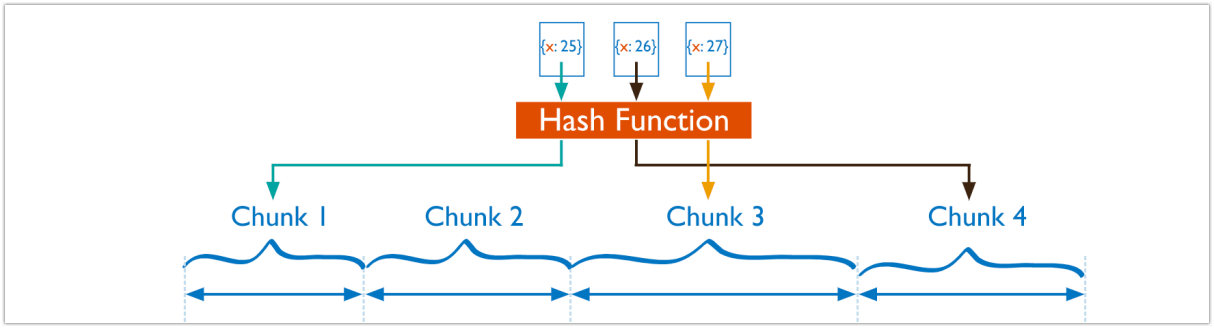
9.9 分片策略

TapDB 支持两种分片策略来在分片集群之间分发数据。

9.9.1 哈希分片

哈希分片计算分片键字段值的哈希值，并根据哈希值将数据分配到块中。

> 提示：
> TapDB 在使用哈希索引解析查询时会自动计算哈希值，应用程序无需计算哈希。

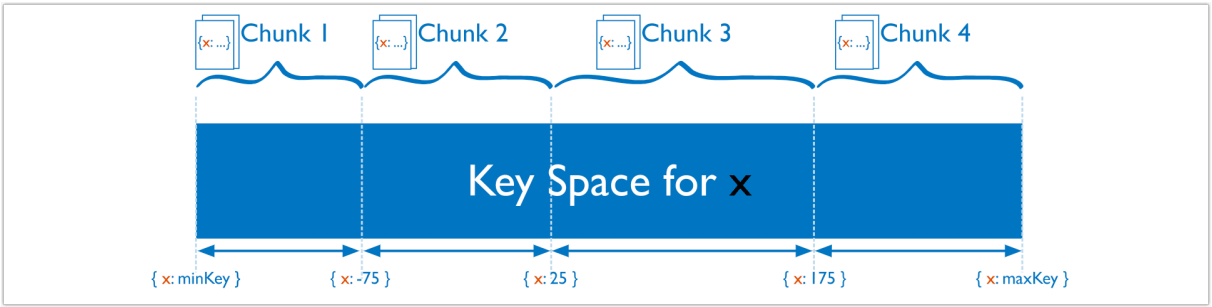


虽然分片键的范围可能“接近”，但其哈希值不太可能位于同一块上。哈希分布有利于更均匀的数据分布，尤其是在分片键单调变化的数据集中。

然而，哈希分布导致对分片键的范围查询不太可能针对单个分片，从而导致更多的集群范围广播操作。

9.9.2 范围分片

范围分片根据分片键值将数据划分为不同范围，并根据分片键值为每个块分配范围。



一系列值“接近”的分片键更有可能位于同一块上。这允许进行有针对性的操作，因为 taps 可以将操作仅路由到包含所需数据的分片。

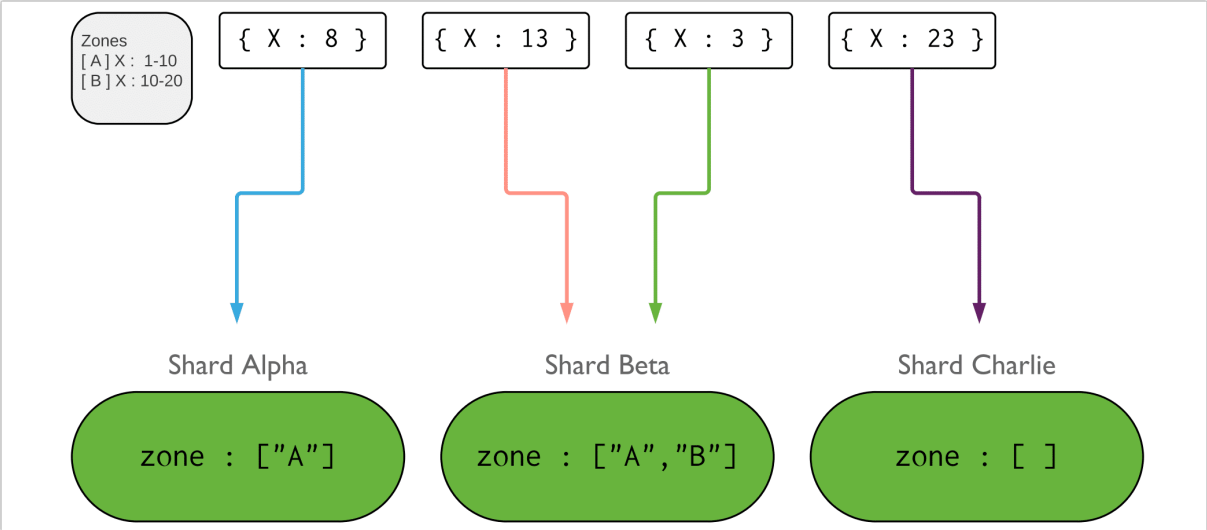
范围分片的效率取决于分片键的选择。选择不当的分片键可能会导致数据分布不均匀，进而导致性能瓶颈。

9.10 分片集群中的区域

区域可以帮助改善跨多个数据中心的分片集群的数据局部性。

在分片集群中，您可以根据分片键创建分片数据区域，并将每个区域与集群中的一个或多个分片关联。一个分片可以与任意数量的区域关联。在平衡集群时，TapDB 仅将区域覆盖的块迁移到与该区域关联的分片。

每个区域涵盖一个或多个分片键值范围，并始终包括其下边界但不包括其上边界。



定义区域覆盖的新范围时，必须使用分片键中包含的字段。如果使用复合分片键，则范围必须包含分片键的前缀。更多详细信息，请参阅区域中的分片键。

在选择分片键时，考虑将来可能使用的区域。

> ▷ 提示：
> 从 TapDB 4.0.3 开始，在对空集合或不存在的集合进行分片之前设置区域和区域范围，可以更快地设置区域分片。

9.11 分片中的排序规则

使用 `shardCollection` 带有选项的命令对具有默认排序规则 `collation : { locale : "simple" }` 的集合进行分片。成功分片需要：

- 集合必须有一个以分片键为前缀的索引。
- 索引必须具有排序规则 `{ locale: "simple" }`。

当使用排序规则创建新的集合时，请确保在对集合进行分片之前满足这些条件。

> ▷ 提示：
> 对分片集合的查询继续使用为集合配置的默认排序规则。要使用分片键索引的排序规则 `simple`，请在查询的排序规则文档中指定 `{ locale : "simple" }`。

9.12 变更流

从 TapDB 3.6 开始，副本集和分片集群可以使用变更流（Change Streams）。变更流允许应用程序访问实时数据变更，而无需担心跟踪 oplog 的复杂性和风险。应用程序可以使用变更流订阅一个或多个集合上的所有数据变更。

9.13 事务

随着分布式事务的引入，分片集群上可以进行多文档事务。

在事务提交之前，事务中所做的数据更改在事务外部不可见。

但是，当事务写入多个分片时，并非所有外部读取操作都需要等待已提交事务的结果在所有分片上可见。例如，如果事务已提交，并且写入 1 在分片 A 上可见，但写入 2 在分片 B 上尚不可见，则读取关注 `"local"` 的外部读取可以读取写入 1 的结果，而无需查看写入 2。

10 管理 TapDB

10.1 生产说明

本页详细介绍了影响 TapDB 系统配置的因素，尤其是在生产环境中运行时的注意事项。

> ▷ 提示：
> TapDB 4.2 删除了已弃用的 MMAPv1 存储引擎，建议将 MMAPv1 存储引擎更改为 WiredTiger 存储引擎。

10.1.1 平台支持说明

> ▷ 提示：
> 在 macOS 10.12.x、10.13.x 和 10.14.0 上非正常关闭期间，TapDB 4.0 可能会丢失数据。Apple 已在 macOS 10.14.1 中修复此问题。

10.1.1.1 x86_64 架构

TapDB 需要满足以下最低配置要求的 x86_64 微架构：

对于 **Intel x86_64**：

- Sandy Bridge 或更高版本的酷睿处理器
- Tiger Lake 或更高版本的赛扬或奔腾处理器

对于 **AMD x86_64**：

- Bulldozer 或更高版本的处理器

从 TapDB 5.0 开始，TapDB、taps 和旧版 TapDB shell 不再支持不符合最低微架构要求的 x86_64 平台。

- TapDB 仅支持运行 Red Hat Compatible Kernel (RHCK) 的 Oracle Linux。TapDB 不支持 Unbreakable Enterprise Kernel (UEK)。
- TapDB 5.0 需要使用 AVX 指令集，该指令集在部分 Intel 和 AMD 处理器上可用。

10.1.1.2 arm64 架构

在 arm64 上，TapDB 需要 ARMv8.2-A 或更高版本的微架构。

从 TapDB 5.0 开始，TapDB、taps 和旧版 TapDB shell 不再支持不符合最低微架构要求的 arm64 平台。

> ▷ 提示：
> TapDB 不再支持缺乏适当 CPU 架构的单板硬件（如 Raspberry Pi 4）。

10.1.2 平台支持列表

10.1.2.1 x86_64 架构

- Ubuntu 22.04，20，18
- CentOS 8.5，7.2
- Debian 10
- Kylin v10-sp2
- openEuler-22.03-sp3

10.1.2.2 arm 架构

- CentOS 8.5
- Kylin v10-sp1

10.1.3 dbPath

dbPath 目录中的文件必须与配置的存储引擎 tapdbdbPath--storageEngine 相对应。必须拥有指定 dbPath 的读取和写入权限。

如果您使用防病毒 (AV) 扫描程序或端点检测和响应 (EDR) 扫描程序，请将扫描程序配置为从扫描中排除 database storage path 和 database log path 。

database storage path 中的数据文件已压缩。此外，如果使用加密存储引擎，数据文件也会被加密。扫描这些文件的 I/O 和 CPU 成本可能会显着降低性能，而不提供任何安全优势。

如不排除 database storage path 和 database log path 中的目录，扫描程序可能会隔离或删除重要文件。丢失或隔离的文件可能会损坏数据库并使 TapDB 实例崩溃。

10.1.4 并发

10.1.4.1 WiredTiger

WiredTiger 支持读取者和写入者对集合中的文档进行并发访问。客户端可以在写入操作进行时读取文档，并且多个线程可以同时修改集合中的不同文档。

10.1.5 数据一致性

10.1.5.1 日记

TapDB 使用预写式日志记录到磁盘上的日志。日志保证在由于崩溃或其他严重故障而终止的情况下，TapDB 可以快速恢复已写入日志但未写入数据文件的 写入操作 TapDB。

保持日志功能处于启用状态，以确保 TapDB 能够在崩溃后恢复其数据文件并使数据文件保持在有效状态。有关更多信息，请参阅日志。

从 TapDB 4.0 ，您不能为使用 WiredTiger 存储引擎的副本集成员指定 `--nojournal` 选项或 `storage.journal.enabled: false` 。

10.1.5.2 读关注 (read concern)

从 TapDB 3.6，如果写入请求确认，则可以使用因果一致会话来读取自己的写入。

TapDB 3.6 之前的版本，为了读取您自己的写入，您必须使用 `{ w: "majority" }` 写关注发出写入操作，然后使用 `primary` 读取偏好以及 `"majority"` 或 `"linearizable"` 读关注发出读取操作。

10.1.5.3 写关注

写关注描述了 TapDB 为写入操作请求的确认级别。写关注的级别会影响写入操作的返回速度。当写入操作具有较弱的写关注时，它们会快速返回。对于更强的写关注，客户端在发送写入操作后必须等待，直到 TapDB 在请求的写关注级别确认写入操作。由于写关注不足，写入操作可能在客户端看来已成功，但在某些服务器故障的情况下可能不会持久化。

10.1.6 网络

10.1.6.1 使用可信网络环境

务必在可信环境中运行 TapDB，使用网络规则阻止来自所有未知计算机、系统和网络的访问。与任何依赖于网络访问的敏感系统一样，您的 TapDB 部署应该只能由需要访问的特定系统访问，例如应用程序服务器、监控服务和其他 TapDB 组件。

> ▷ 提示：
> 默认情况下，未启用授权，并且 tapdb 假定环境可信。根据需要启用 authorization 模式。

10.1.6.2 禁用 HTTP 接口

在版本 3.6 中进行了更改：TapDB 3.6 删除了已弃用的 HTTP 接口和 TapDB 的 REST API。

早期版本的 TapDB 提供一个 HTTP 接口来检查服务器的状态，且可选地运行查询。该 HTTP 接口默认禁用。请勿在生产环境中启用该 HTTP 接口。

10.1.6.3 管理连接池大小

通过调整连接池大小以适应您的使用案例，避免 TapDB 或 taps 实例的连接资源过载。从当前数据库请求典型数量的 110 - 115 % 开始，并根据需要修改连接池大小。

connPoolStats 命令返回有关分片集群中 taps 和 tapdb 实例的当前数据库的打开连接数的信息。

10.1.7 硬件考虑因素

TapDB 专门针对商用硬件而设计，几乎没有硬件要求或限制。TapDB 的核心组件在小端硬件上运行，主要是 x86/x86_64 处理器。客户端库（即驱动程序）可以在大端或小端系统上运行。

10.1.7.1 分配足够的 RAM 和 CPU

至少应确保每个 tapdb 或 taps 实例都能访问两个实际核心或一个多核物理 CPU。

10.1.7.1.1 WiredTiger

WiredTiger 存储引擎是多线程的，可以利用额外的 CPU 内核。具体来说，相对于可用 CPU 数量的活动线程（即并发操作）总数可能会影响性能：

- 吞吐量随着并发活动操作数量增加而增加，最高可达 CPU 数量。
- 当同时活跃的操作数量超过 CPU 数量且超出的数量达到一定阈值时，吞吐量便会降低。

阈值取决于您的应用程序。您可以通过试验和测量吞吐量来确定应用程序的最佳并发活动操作数量。`tapstat` 的输出在 (ar|aw) 列中提供了关于活动读取/写入数量的统计信息。

借助 WiredTiger，TapDB 可同时利用 WiredTiger 内部缓存和文件系统缓存。

默认情况下，WiredTiger 对所有集合使用 Snappy 区块压缩，对所有索引使用前缀压缩。压缩默认值可以在全局级别进行配置，也可以在集合和索引创建期间针对每个集合和每个索引进行设置。

WiredTiger 内部缓存和磁盘格式中的数据使用不同的表示形式：

- 文件系统缓存中的数据与磁盘上的数据格式相同，并且同样拥有数据文件压缩带来的好处。操作系统使用文件系统缓存来减少磁盘 I/O。
- WiredTiger 内部缓存中加载的索引具有与磁盘上格式不同的数据表示形式，但仍可利用索引前缀压缩来减少 RAM 使用量。索引前缀压缩会对被索引字段中的常用前缀去重。
- WiredTiger 内部缓存中的集合数据未压缩，并使用与磁盘上格式不同的表示形式。区块压缩可大幅节省磁盘上存储空间，但数据必须解压缩才能由服务器操作。

要调整 WiredTiger 内部缓存的大小，请参阅 storage.wiredTiger.engineConfig.cacheSizeGB 和-wiredTiger-CacheSizeGB 。避免将 WiredTiger 内部缓存大小增加到超过其默认值。

> 提示：
> storage.wiredTiger.engineConfig.cacheSizeGB 限制 WiredTiger 内部缓存的大小。操作系统使用可用的空闲内存进行文件系统缓存，这允许压缩的 TapDB 数据文件保留在内存中。此外，操作系统还使用任何空闲 RAM 来缓冲文件系统区块和文件系统缓存。
> 为了容纳额外的 RAM 用户，您可能必须减少 WiredTiger 的内部缓存大小。

默认 WiredTiger 内部缓存大小值假定每台计算机有一个 tapdb 实例。如果一台计算机包含多个 TapDB 实例，则应减少该设置以容纳其他 tapdb 实例。

如果在容器（例如 tapdb 、lxc 、Docker 等）中运行 cgroups ，而该容器无法访问系统中的所有可用 RAM，则必须将 storage.wiredTiger.engineConfig.cacheSizeGB 设置为一个值小于容器中可用的 RAM 量。确切的数量取决于容器中运行的其他进程。请参阅 memLimitMB 。

要查看有关缓存和逐出率的统计信息，请参阅 `wiredTiger.cache` 命令返回的 `serverStatus` 字段。

10.1.7.1.2 压缩和加密

使用加密时，配备 AES-NI 指令集扩展的 CPU 表现出显着的性能优势。

10.1.7.1.3 使用固态硬盘 (SSD)

TapDB 在搭配 SATA SSD（固态硬盘）的情况下具有很好的效果以及出色的性价比。

如果可用且注重经济性，请使用固态硬盘 (SSD)。

商用 (SATA) 旋转驱动器通常是不错的选择，因为价格更昂贵的旋转驱动器提供的随机 I/O 性能提升并不明显（仅为 2 倍左右）。使用固态硬盘或增加 RAM 可能更有效地提高 I/O 吞吐量。

10.1.7.1.4 TapDB 和 NUMA 硬件

在具有非统一内存访问 (NUMA) 的系统上运行 TapDB 可能会导致许多操作问题，包括时段性的性能缓慢和系统进程使用率较高。

在 NUMA 硬件上运行 TapDB 服务器和客户端时，应当配置内存交织策略，以便主机以非 NUMA 方式运行。当部署在 Linux（自版本 2.0 起）和 Windows（自版本 2.6 起）计算机上时，TapDB 会在启动时检查 NUMA 设置。如果 NUMA 配置可能会降低性能，TapDB 会打印警告消息。

在 Windows 上配置 NUMA

在 Windows 上，必须通过计算机的 BIOS 启用内存交织。有关详细信息，请参阅您的系统文档。

在 Linux 上配置 NUMA

在 Linux 上，您必须禁用区域回收，并确保 tapdb 和 taps 实例由 numactl 启动，这通常是通过平台的初始化系统配置的。您必须执行这两个操作，才能正确禁用 TapDB 的 NUMA。

1. 使用以下命令禁用区域回收：

```
echo 0 | sudo tee /proc/sys/vm/zone_reclaim_mode
```

```
sudo sysctl -w vm.zone_reclaim_mode=0
```

2. 确保 tapdb 和 taps 由 numactl 启动。这通常是通过平台的初始化系统配置的。运行以下命令，确定平台上正在使用的初始化系统：

```
ps --no-headers -o comm 1
```

-如果为 `systemd`，则您的平台使用 systemd init 系统，并且您必须 按照下面的 systemd 标签页中的步骤来编辑 TapDB 服务文件。

- 如果为 “`init`”，则您的平台使用 SysV Init 系统且无需执行此步骤。SysV Init 的默认 TapDB 初始化脚本包括默认通过 `numactl` 来启动 TapDB 实例的必要步骤。
- 如果您管理自己的初始化脚本（即不使用这些初始化系统中的任何一个），则必须按照下面 Custom init scripts（自定义初始化脚本）选项卡中的步骤来编辑您的自定义初始化脚本。

10.1.7.2 磁盘和存储系统

10.1.7.2.1 交换

在可以避免交换或将交换保持在最低限度的情况下，TapDB 性能最佳，因为从交换中检索数据总是比访问 RAM 中的数据慢。但是，如果托管 TapDB 的系统耗尽 RAM，则交换可以阻止 Linux OOM Killer 终止 `tapdb` 进程。

一般来说，您应选择以下交换策略之一：

1. 在您的系统上分配交换空间，并将内核配置为仅允许在高内存负载下进行交换，或是
2. 请勿在系统上分配交换空间，并将内核配置为完全禁用交换

请参阅设置 vm.swappiness，了解在 Linux 系统上按照这些指南配置交换的说明。

> **提示：**
> 如果您的 TapDB 实例托管在还运行其他软件（例如网络服务器）的系统上，则应选择第一个交换策略。在此情况下，请勿禁用交换。如果可能，强烈建议在自己的专用系统上运行 TapDB。

10.1.7.2.2 RAID

为了在存储层方面获得最佳性能，请使用 RAID-10 支持的磁盘。RAID-5 和 RAID-6 通常无法提供足够的性能来支持 TapDB 部署。

10.1.7.2.3 远程文件系统 (NFS)

使用 WiredTiger 存储引擎，如果远程文件系统符合 ISO/IEC 9945-1:1996 (POSIX.1)，则 WiredTiger 对象可以存储在远程文件系统上。由于远程文件系统通常比本地文件系统慢，因此使用远程文件系统进行存储可能会降低性能。

如果决定使用 NFS，请将以下 NFS 选项添加到 `/etc/fstab` 文件中：

- bg
- hard
- nolock
- noatime
- nointr

根据您的内核版本，其中一些值可能已设置为默认值。有关更多信息，请参阅您的平台文档。

10.1.7.2.4 将组件分离到不同存储设备

为了提高性能，请考虑根据应用程序的访问和写入模式将数据库的数据和日志分布存放到不同的存储设备上。将组件安装为单独的文件系统，并使用符号链接将每个组件的路径映射到存储它的设备。

对于 WiredTiger 存储引擎,您还可以将索引存储在不同的存储设备上。请参阅 `storage.wiredTiger.engineConfig.dir` 。

> ▷ 提示：
> 使用其他存储设备会影响创建数据快照式备份的能力，因为这些文件会位于不同的设备和卷上。

10.1.7.2.5 调度

虚拟设备或云托管设备的调度

对于通过虚拟机监控程序连接到虚拟机实例或由云托管提供商托管的本地区块设备，客户机操作系统应使用 cfq 调度程序以获得最佳性能。cfq 调度器允许操作系统将 I/O 调度推迟到底层虚拟机监控程序。

▷ 提示：

如果满足以下所有条件，则可以使用 noop 调度器进行调度：

- 虚拟机监视器为 VMware。
- 使用副本集拓扑结构或分片集群。
- 虚拟机位于同一虚拟主机上。
- 包含 DBpaths 的底层存储是一个常见的 LUN 块存储。

调度物理服务器

对于物理服务器，操作系统应该使用截止时间调度程序。截止时间调度程序限制每个请求的最大延迟，并保持良好的磁盘吞吐量，这是磁盘密集型数据库应用程序的最佳选择。

10.1.8 架构

10.1.8.1 副本集

有关副本集部署的架构注意事项的概述，请参阅副本集架构文档。

10.1.8.2 分片集群

有关针对生产部署的推荐分片集群架构的概述，请参阅分片集群生产架构。

10.1.9 压缩

WiredTiger 可以使用以下一种压缩库压缩集合数据：

- snappy

  提供低于 zlib 或 zstd 的压缩率，但 CPU 成本比二者均低。

- ZLIB

  提供高于 snappy 的压缩率，但 CPU 成本高于 snappy 和 zstd。

- zstd （从 TapDB 4 开始可用。2 ）

  提供高于 snappy 和 zlib 的压缩率，且 CPU 成本低于 zlib。

默认情况下,WiredTiger 使用 snappy 压缩库。要更改压缩设置,请参阅 `storage.wiredTiger.collectionConfig.bl`

。

WiredTiger 默认对所有索引使用前缀压缩。

10.1.10 时钟同步

TapDB 组件 保留逻辑时钟以支持时间相关的操作。使用 NTP 同步主机时钟可降低组件之间时钟漂移的风险。组件之间的时钟漂移会增加时间相关操作出现错误或异常行为的可能性，例如：

- 如果任何给定 TapDB 组件的底层系统时钟与同一部署中的其他组件偏差一年或更长时间，则这些节点之间的通信可能会变得不可靠或完全停止。

  maxAcceptableLogicalClockDriftSecs 参数控制组件之间可接受的时钟漂移量。maxAcceptableLogical-ClockDriftSecs 值较低的集群对时钟漂移的容忍度也相应较低。

- 对于返回当前集群或系统时间的操作，例如 Date() 、NOW 和 CLUSTER_TIME ，具有不同系统时钟的两个集群成员可能会返回不同的值。
- 依赖计时的功能在集群中可能会出现不一致或不可预测的行为，并且 TapDB 组件之间存在时钟漂移。

使用 Wired Tiger 存储引擎运行低于 或 的 TapDB 的部署需要 NTP3.4.6 3.2.17 同步，其中时钟漂移可能会导致 检查点挂起。此问题已在 TapDB3 中修复。4.6+ 变更日志和 TapDB3.2.17+ 发布说明，并在 TapDB3.6, 4.0, 4.2 和更高版本的所有单点版本中解决。

10.1.11 平台特定注意事项

10.1.11.1 Linux 上的 TapDB

10.1.11.1.1 内核和文件系统

在 Linux 上的生产环境中运行 TapDB 时，应使用 Linux kernel 2.6.36 版本或更高版本，使用 XFS 或 EXT4 文件系统。尽可能使用 XFS，因为它通常在 TapDB 中表现更好。

使用 WiredTiger 存储引擎时，强烈建议对数据承载节点使用 XFS，以避免将 EXT 4 与 WiredTiger 一起使用时可能出现的性能问题。

- 一般来说，如果您使用 XFS 文件系统，请至少使用 Linux 内核版本 2.6.25。
- 如果使用 EXT4 文件系统，请至少使用 Linux 内核版本 2.6.28 。
- 在 Red Hat Enterprise Linux 和 CentOS 上，至少应使用 Linux 内核版本 2.6.18-194。

10.1.11.1.2 系统 C 库

TapDB 使用 GNU C 库 (glibc)（在 Linux 上运行）。通常，每个 Linux 发行版都提供自己的经过审查的该库版本。为获得最佳结果，请使用此系统提供版本可用的最新更新。您可以使用系统的软件包管理器检查是否安装了最新版本。例如：

- 在 RHEL / CentOS 上，以下命令会更新系统提供的 GNU C 库：

```
sudo yum update glibc
```

- 在 Ubuntu / Debian 上，以下命令会更新系统提供的 GNU C 库：

```
sudo apt-get install libc6
```

10.1.11.1.3 fsync() 用于目录

> 提示：
> TapDB 需要的文件系统要能够在 fsync() 支持 目录上。例如，HGFS 和 Virtual Box 的共享文件夹不 支持此操作。

10.1.11.1.4 将 vm.swappiness 设置为 1 或 0

“Swappiness” 是一个影响虚拟内存管理器行为的 Linux 内核设置。vm.swappiness 设置范围从 0 到 100：值越大，它就越倾向于将内存页面交换到磁盘，而不是从 RAM 中删除页面。

- 设置为 0 会完全禁用交换 [4] 。
- 设置为 1 允许内核仅进行交换以避免内存不足问题。
- 设为 60 可指示内核频繁交换到磁盘，也是众多 Linux 发行版上的默认值。
- 设置为 100 将通知内核主动交换到磁盘。

在可以避免交换或将交换保持在最低限度的情况下，TapDB 性能最佳。因此，您应根据应用程序需求和集群配置，将 vm.swappiness 设置为 1 或 0。

> 提示：
> 大多数系统和用户进程在 cgroup 中运行，默认情况下将 vm.swappiness 设置为 60。如果运行的是 RHEL/CentOS，请将 vm.force_cgroup_v2_swappiness 设置为 1，以确保指定的 vm.swappiness 值覆盖任何 cgroup 默认值。

[4] 对于 3.5 之前的 Linux 内核版本或 RHEL 之前的 2.6.32-303 / CentOS 内核版本，将 vm.swappiness 设置为 0 时，内核仍能在某些紧急情况下进行交换。

> 提示：
> 如果 TapDB 实例托管在还运行其他软件（例如网络服务器）的系统上，则应将 vm.swappiness 设置为 1。如果可能，强烈建议在自己的专用系统上运行 TapDB。

- 要检查系统上的当前交换设置，请运行：

```
cat /proc/sys/vm/swappiness
```

- 要更改系统上的交换状态，请执行以下操作：

  1. 编辑 /etc/sysctl.conf 文件并添加以下行：

```
vm.swappiness = 1
```

  2. 运行如下命令以应用设置：

```
sudo sysctl -p
```

> ▷ 提示：
> 如果正在运行 RHEL/CentOS 并使用 tuned 性能配置文件，则还必须编辑所选配置文件以将 vm.swappiness 设置为 1 或 0。

10.1.11.1.5 建议配置

对于所有 TapDB 部署：

- 使用网络时间协议 (NTP) 在主机之间同步时间。这在分片集群中尤其重要。

对于 WiredTiger 存储引擎，请考虑以下建议：

- 关闭包含 atime 的存储卷的数据库文件
- 根据 ulimit 参考中的建议，调整平台的 ulimit 设置。当高强度使用时，较低的 ulimit 值会对 TapDB 产生负面影响，并可能导致与 TapDB 进程的连接失败和服务丢失。
- 禁用透明大页。TapDB 在使用普通（4096 字节）虚拟内存页面时性能更好。请参阅透明大页设置。
- 在 BIOS 中禁用 NUMA。如果不可行，请参阅 NUMA 硬件上的 TapDB。
- 如果不使用默认的 TapDB 目录路径或端口，请为 TapDB 配置 SELinux。

对于 WiredTiger 存储引擎：

- 无论存储介质类型（旋转磁盘、SSD 等）为何，均应将预读大小设为 8 到 32 之间。

  更高的预读通常有利于顺序 I/O 操作。由于 TapDB 磁盘访问模式通常是随机的，因此使用更高的预读设置的好处有限，或者可能会导致性能下降。因此，除非测试表明更高的预读值具有可测量、可重复且可靠的优势，否则为了获得最佳的 TapDB 性能，请将预读值设置在 8 到 32 之间。TapDB 商业支持部门可提供有关备用预读配置的建议和指导。

10.1.11.1.6 TapDB 和 TLS/SSL 库

在 Linux 平台上，您可能会在 TapDB 日志中看到以下语句之一：

```
<path to TLS/SSL libs>/libssl.so.<version>: no version information available
 ↪  (required by /usr/bin/tapdb)
<path to TLS/SSL libs>/libcrypto.so.<version>: no version information
 ↪  available (required by /usr/bin/tapdb)
```

这些警告表明系统的 TLS/SSL 库与编译 tapdb 所针对的 TLS/SSL 库不同。通常，这些消息不需要干预；但是，您可以使用以下操作来确定 tapdb 期望的符号版本：

```
objdump -T <path to tapdb>/tapdb | grep " SSL_"
objdump -T <path to tapdb>/tapdb | grep " CRYPTO_"
```

这些操作将返回类似于以下行之一的输出：

```
0000000000000000      DF *UND*       0000000000000000  libssl.so.10 SSL_write
0000000000000000      DF *UND*       0000000000000000  OPENSSL_1.0.0
 ↪  SSL_write
```

此输出中的最后两个字符串是符号版本和符号名称。将这些值与以下操作返回的值进行比较，以检测符号版本不匹配情况：

```
objdump -T <path to TLS/SSL libs>/libssl.so.1*
objdump -T <path to TLS/SSL libs>/libcrypto.so.1*
```

此过程既不精确也不详尽：tapdb 库中的 libcrypto 使用的许多符号并非以 CRYPTO_ 开头。

10.1.11.2 Windows 上的 TapDB

对于使用 WiredTiger 存储引擎的 TapDB 实例，Windows 上的性能与 Linux 上的性能相当。

10.1.11.3 虚拟环境中的 TapDB

本部分介绍在一些比较常见的虚拟环境中运行 TapDB 时的注意事项。

10.1.11.3.1 VMware

TapDB 与 VMware 兼容。

VMware 支持内存复用，因而可为虚拟机分配比物理机可用内存更多的内存。当内存超量时，虚拟机监控程序会在各虚拟机之间重新分配内存。VMware 的气球驱动程序 (vmmemctl) 会回收被视为价值最低的页面。

气球驱动程序驻留在客户机操作系统中。在某些配置下，当气球驱动程序扩展时，它可能会干扰 TapDB 的内存管理并影响 TapDB 的性能。

为了防止气球驱动程序和内存过量承诺功能对性能产生负面影响，请为运行 TapDB 的虚拟机保留全部内存量。为虚拟机预留适当数量的内存，可以防止本地操作系统中的气球在管理程序存在内存压力时膨胀。

尽管气球驱动程序和内存超配功能可能会在某些配置下对 TapDB 性能产生负面影响，但请勿禁用这些功能。如果禁用这些功能，虚拟机监控程序可能会使用其交换空间来满足内存请求，这会对性能产生负面影响。

通过设置 VMware 关联性规则，确保虚拟机停留在特定 ESX/ESXitapdb 主机上。如果您必须手动将虚拟机迁移到其他主机，并且虚拟机上的 实例是 主节点，则必须先 step down 主节点，然后 `shut down the instance`。

遵循 vMotion 网络最佳实践 和 VMKernel 。不遵循最佳实践可能会导致性能问题，并影响 副本集 和 分片集群 的高可用性机制。

您可以克隆运行 TapDB 的虚拟机。您可以使用此功能部署新的虚拟主机，将其添加为副本集的成员。

10.1.11.3.2 KVM

TapDB 与 KVM 兼容。

KVM 支持内存复用，因而可以为虚拟机分配比物理机可用内存更多的内存。当内存超量时，虚拟机监控程序会在各虚拟机之间重新分配内存。KVM 的气球驱动程序会回收被视为价值最低的页面。

气球驱动程序驻留在客户机操作系统中。在某些配置下，当气球驱动程序扩展时，它可能会干扰 TapDB 的内存管理并影响 TapDB 的性能。

为了防止气球驱动程序和内存过量承诺功能对性能产生负面影响，请为运行 TapDB 的虚拟机保留全部内存量。为虚拟机预留适当数量的内存，可以防止本地操作系统中的气球在管理程序存在内存压力时膨胀。

尽管气球驱动程序和内存超配功能可能会在某些配置下对 TapDB 性能产生负面影响，但请勿禁用这些功能。如果禁用这些功能，虚拟机监控程序可能会使用其交换空间来满足内存请求，这会对性能产生负面影响。

10.1.12 性能监测

10.1.12.1 iostat

在 Linux 上，使用 iostat 命令可检查磁盘 I/O 是否是数据库的瓶颈。运行 iostat 时应指定秒数，以避免显示涵盖服务器启动以来的时间的统计信息。

例如，以下命令将显示扩展统计信息以及每个显示报告的时间，流量以 MB/s 为单位，间隔时间为一秒：

```
iostat -xmt 1
```

iostat 中的关键字段：

- %util：这是快速检查时最有用的字段，表示设备/驱动程序正在使用的时间百分比。
- avgrq-sz：平均请求大小。此值的数字越小，则表示随机 I/O 操作越多。

10.1.12.2 bwm-ng

bwm-ng 是一个用于监控网络使用情况的命令行工具。如果怀疑存在基于网络的瓶颈，可以使用 bwm-ng 开始诊断过程。

10.1.13 备份

要备份 TapDB 数据库，请参阅TapDB 备份方法概述。

10.2 TapDB 备份方法

10.2.1 使用 TapData（推荐）

TapData 是一款面向数据服务的平台化产品，提供数据复制、数据转换、数据开发等多种能力，不仅可以满足 TapDB 副本集、分片集的全量和增量备份，更是一个以低延迟数据复制为核心优势构建的实时异构数据集成和数据服务平台。

典型用例包括数据异地灾备、数据迁移、数据入湖、构建实时数据管道及通用 ETL 处理等。

10.2.2 通过复制数据文件进行备份

10.2.2.1 使用 AES256-GCM 的加密存储引擎的注意事项

对于使用 AES256-GCM 加密模式的存储引擎，AES256-GCM 要求每个进程使用带有密钥的唯一计数器区块值。注意事项如下：

- 从热备份恢复

  从 4.2 版本开始，如果从 “热” 备份（即 TapDB 正在运行时获取的备份）中恢复数据，TapDB 可以在启动时检测 “脏” 密钥，并自动更新数据库密钥以避免重复使用初始化向量（IV）。

- 从冷备份恢复

  如果从“冷”备份（即 TapDB 未运行时获取的备份）中恢复数据，TapDB 无法在启动时检测“脏”密钥，重复使用 IV 会导致机密性和完整性保证失效。

  从 4.2 版本开始，为了避免从冷备份恢复后重复使用密钥，TapDB 添加了新的命令行选项 `--eseDatabaseKeyRollover`。使用该选项启动时，TapDB 实例会更新使用 AES256-GCM 加密配置的数据库密钥并退出。

10.2.2.2 使用文件系统快照进行备份

可以通过复制 TapDB 底层数据文件来创建 TapDB 部署的备份。

如果 TapDB 存储数据文件的卷支持时间点快照，您可以使用这些快照创建特定时刻的 TapDB 系统备份。文件系统快照是操作系统卷管理器的功能，而不是 TapDB 特有的功能。操作系统可以使用文件系统快照创建卷的快照，用作数据备份的基准。快照机制取决于底层存储系统。例如，在 Linux 上，Logical Volume Manager (LVM) 可以创建快照。同样，Amazon EC2 的 EBS 存储系统也支持快照。

要获得正在运行的 TapDB 进程的正确快照，必须启用日志功能，并且日志必须与其他 TapDB 数据文件位于同一逻辑卷上。如果未启用日志功能，则无法保证快照的一致性或有效性。

要获得分片集群的一致快照，必须禁用负载均衡器，并在大约相同的时间点从每个分片和配置服务器捕获快照。

10.2.2.3 使用 cp 或 rsync

如果存储系统不支持快照，可以使用 `cp` 或 `rsync` 等工具直接复制文件。由于复制多个文件不是原子操作，因此必须在复制文件之前停止对 TapDB 的所有写入。

通过复制底层数据生成的备份不支持副本集的时间点恢复，并且难以管理较大的分片集群。此外，这些备份更大，因为它们包括索引和重复的底层存储填充和碎片。相比之下，`tapdump` 创建的备份较小。

10.2.3 使用 tapdump

`tapdump` 从 TapDB 数据库读取数据并创建高保真 BSON 文件，而 `taprestore` 工具可以使用这些文件来填充 TapDB 数据库。`tapdump` 和 `taprestore` 是备份和恢复小型 TapDB 部署的简单而高效的工具。

`tapdump` 和 `taprestore` 针对正在运行的 TapDB 进程进行操作，并且可以直接操作底层数据文件。默认情况下，`tapdump` 不捕获本地数据库的内容。

`tapdump` 仅捕获数据库中的文档。生成的备份节省空间，但 `taprestore` 或 TapDB 必须在恢复数据后重建索引。

连接到 TapDB 实例时，`tapdump` 可能会对 TapDB 性能产生不利影响。如果数据大于系统内存，查询会将工作集挤出内存，从而导致页面错误。

在 tapdump 捕获输出时，应用程序可以继续修改数据。对于副本集，tapdump 提供 --oplog 选项，以在其输出的 Oplog 条目中包含 tapdump 操作期间出现的条目。这允许相应的 taprestore 操作重放捕获的 Oplog。要恢复使用 --oplog 创建的备份，请使用 taprestore 和 --oplogReplay 选项。
