MongoDB中文社区

MongoDB时序的设计与应用

MongoDB中文社区南京分会主席
陈亮亮

MongoDB 中文社区

大纲

- 解析MongoDB新特性-时序
- 如何在MongoDB中使用时序?
- 聊聊MongoDB时序集合性能
- MongoDB时序iot场景设计

MongoDB
中文社区

解析MongoDB新特性-时序

解析MongoDB新特性-时序

MongoDB时序集合是MongoDB5.0新推出的功能，他能快速将一段时间内的数据写入磁盘，并且提供快速时序检索的集合。

与普通集合相比，时序集合在数据插入的过程中，自动将数据按照时间维度组织成最优的存储格式，也为后面应用程序对时序数据提高了查询效率。

解析MongoDB新特性-时序

MongoDB 中文社区

MongoDB 数据桶模式：

假设我们有一个传感器每分钟测量温度并将其保存到数据库中，我们需要写入数据库中的数据流：

```
{_id: ObjectId(),deviceid: 1,date: ISODate("2019-11-10"),samples : [{ temperature: 10, time: 1573833152},]},
{_id: ObjectId(),deviceid: 1,date: ISODate("2019-11-10"),samples : [{ temperature: 15, time: 1573833153},]},
{_id: ObjectId(),deviceid: 1,date: ISODate("2019-11-10"),samples : [{ temperature: 14, time: 1573833154},]},
{_id: ObjectId(),deviceid: 1,date: ISODate("2019-11-10"),samples : [{ temperature: 20, time: 1573833155},]}
```

解析MongoDB新特性-时序

MongoDB 中文社区

桶模式设计数据模型：

```
{
    _id: ObjectId(),
    deviceid: 1,
    date: ISODate("2019-11-10"),
    first: 1573833152,
    last: 1573833155,
    samples : [
        { temperature: 10, time: 1573833152},
        { temperature: 15, time : 1573833153},
        { temperature: 14, time: 1573833154},
        { temperature: 20, time : 1573833155}
    ]
}
```

字段解释：

- id —文档的ID，这个ID具备唯一性。
- deviceId —查询的设备ID
- date—样品的日期；我们可以将其存储在此处以简化聚合
- first —在存储桶中读取的最旧数据的时间戳
- last —存储桶中读取的最新数据的时间戳
- samples —数据容器

解析MongoDB新特性-时序

MongoDB 中文社区

用例中桶模式的优势：

- 节省索引的大小
- 简化数据结构
- 可以将需要采集的数据按照时间维度集中在一起，方便快速范围检索
- 提升数据写入速度

MongoDB
中文社区

如何在MongoDB中使用时序?

如何在MongoDB中使用时序?

MongoDB 中文社区

显示指定创建的集合为时序集合:

```
db.createCollection(
    "weather",
    {
        timeseries: {
            timeField: "timestamp",
            metaField: "metadata",
            granularity: "hours"
        }
    }
)
```

字段含义介绍:

- timeField 是时间参数, 必须为 BSON data。
- metaField 影响维度基数, 好的 metaField 应该选择低基数的, 有选择性的指标, 高基数必然带来性能的下降。
- granularity 是可选的聚合粒度参数, 数据库会将一个时间段的数据聚合存放, 这个参数影响性能, 不影响功能。
- expireAfterSeconds 影响数据的过期, 是默认通过每 60s 一次的检测实现的。过期时间可配置。

如何在MongoDB中使用时序?

MongoDB 中文社区

增:

```
db.weather.insertMany({"metadata": { "sensorId": 5578, "type": "temperature" },"timestamp": ISODate("2021-05-18T00:00:00.000Z"),"temp": 12});
```

批量插入集合的方式:

```
db.weather.insertMany([{"metadata": { "sensorId": 5578, "type": "temperature" },"timestamp": ISODate("2021-05-18T00:00:00.000Z"),"temp": 12},
                       {"metadata": { "sensorId": 5578, "type": "temperature" },"timestamp": ISODate("2021-05-18T04:00:00.000Z"),"temp": 11},
                       {"metadata": { "sensorId": 5578, "type": "temperature" },"timestamp": ISODate("2021-05-18T08:00:00.000Z"),"temp": 11},
                       {"metadata": { "sensorId": 5578, "type": "temperature" },"timestamp": ISODate("2021-05-18T12:00:00.000Z"),"temp": 12}]);
```

删:

删除条件中必须包含metaField

改:

更新条件中必须包含metaField字段值

如何在MongoDB中使用时序?

MongoDB 中文社区

查:

计算时序集合时段平均值:

```
db.weather.aggregate( [
    {
        $project: {
            date: {
                $dateToParts: { date: "$timestamp" }
            },
            temp: 1
        }
    },
    {
        $group: {
            _id: {
                date: {
                    year: "$date.year",
                    month: "$date.month",
                    day: "$date.day"
                }
            },
            avgTmp: { $avg: "$temp" }
        }
    }
] )
```

```
{
  "_id" : {
    "date" : {
      "year" : 2021,
      "month" : 5,
      "day" : 18
    }
  },
  "avgTmp" : 12.714285714285714
}
{
  "_id" : {
    "date" : {
      "year" : 2021,
      "month" : 5,
      "day" : 19
    }
  },
  "avgTmp" : 13
}
```

如何在MongoDB中使用时序?

MongoDB 中文社区

注意点:

1.时序集合底层存储依然是 WiredTiger,;

2.没有为时序查询定制太多新的语法, 各种聚合依然需要通过 aggregate 进行;

3. 时序集合已经按照常用的查询模式, 对数据进行了存储模型上的优化。在索引上如果有自己的针对 metafield 的过滤需求, 可以正常创建二级索引;

4.MongoDB时序集合在更新和删除中需要添加指定条件。 另外, 在当前版本里, 时序集合不支持分片（6.0支持分片）。

聊聊MongoDB时序集合性能

MongoDB
中文社区

聊聊MongoDB时序集合性能

MongoDB 中文社区

写入性能（4C 8G 128G ssd）：

```
[OVERALL], RunTime(ms), 371870
[OVERALL], Throughput(ops/sec), 26816.895705813706
[TOTAL_GCS_PS_Scavenge], Count, 2392
[TOTAL_GC_TIME_PS_Scavenge], Time(ms), 2867
[TOTAL_GC_TIME_%_PS_Scavenge], Time(%), 0.6828303998856788
[TOTAL_GCS_PS_MarkSweep], Count, 0
[TOTAL_GC_TIME_PS_MarkSweep], Time(ms), 0
[TOTAL_GC_TIME_%_PS_MarkSweep], Time(%), 0.0
[TOTAL_GCs], Count, 2392
[TOTAL_GC_TIME], Time(ms), 2867
[TOTAL_GC_TIME_%], Time(%), 0.6828303998856788
[CLEANUP], Operations, 200
[CLEANUP], AverageLatency(us), 37.695
[CLEANUP], MinLatency(us), 0
[CLEANUP], MaxLatency(us), 7347
[CLEANUP], 95thPercentileLatency(us), 2
[CLEANUP], 99thPercentileLatency(us), 6
[INSERT], Operations, 10000000
[INSERT], AverageLatency(us), 8315.8053209
[INSERT], MinLatency(us), 1262
[INSERT], MaxLatency(us), 3031039
[INSERT], 95thPercentileLatency(us), 18719
[INSERT], 99thPercentileLatency(us), 64287
[INSERT], Return=OK, 10000000
```

聊聊MongoDB时序集合性能

MongoDB 中文社区

读写混合压测性能：

读写比 80：20

MongoDB规格	并发数 threads	吞吐量 throughput(ops/sec)	平均读延迟 RAL(us)	平均写延迟 WAL(us)
2核4G	100	19010	6869	68
2核4G	200	23407	13174	6521
2核8G	100	19278	6732	87
2核8G	200	24580	12225	6034
4核8G	100	38742	2856	121
4核8G	200	39584	5352	2843
4核16G	100	45907	2557	31
4核16G	200	48653	5023	2507
8核16G	100	56011	2089	33
8核16G	200	59764	4019	2033
8核32G	100	105014	1097	30
8核32G	200	143344	2477	2094
16核64G	100	162876	2459	2272
16核64G	200	206616	622	42

聊聊MongoDB时序集合性能

MongoDB 中文社区

磁盘占用：

MongoDB对数据的压缩支持snappy、zstd和zlib算法，在以往线上真实的数据空间大小与真实磁盘空间消耗进行对比，可以得出以下结论：

压缩算法	真实数据量	真实磁盘空间消耗
snappy压缩算法	3.5T	1 – 1.5T
zstd压缩算法	3.5T	0.6 – 0.9T
zlib压缩算法	3.5T	0.5 – 0.7T

Hbase默认采用的是snappy算法，MongoDB时序集合默认采用zstd压缩算法，所以相同数据量，MongoDB磁盘占用更低。

聊聊MongoDB时序集合性能

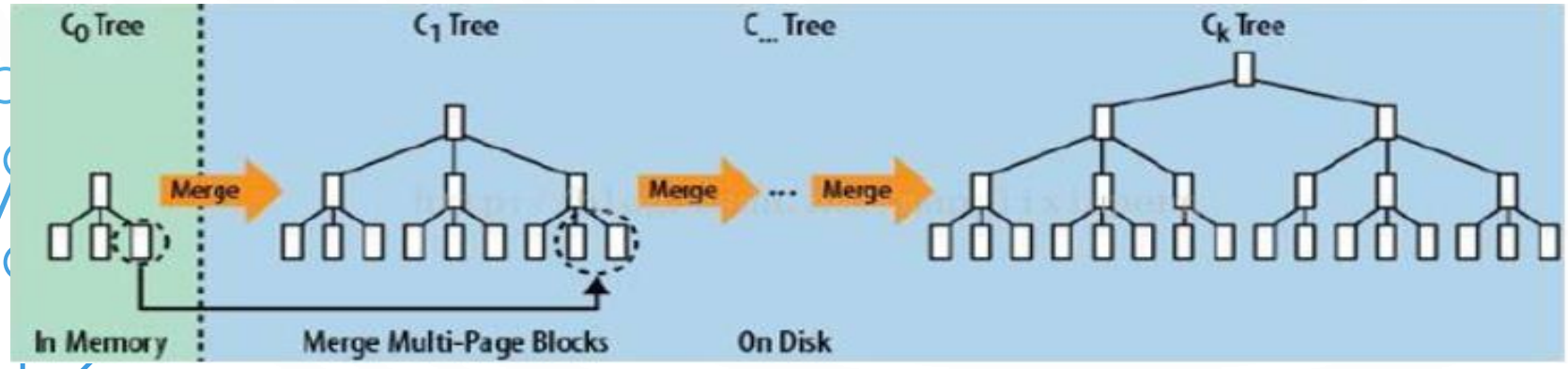
MongoDB 中文社区

LSM Tree 和 B Tree 性能对比

Lsm典型代表：Hbase，RocketDB，InfluxDB等

读写：在大批量写入的场景下，写入速度快，但大批量写入情况下，对CPU和内存消耗较大。原则是用部分性能换取高速写性能。

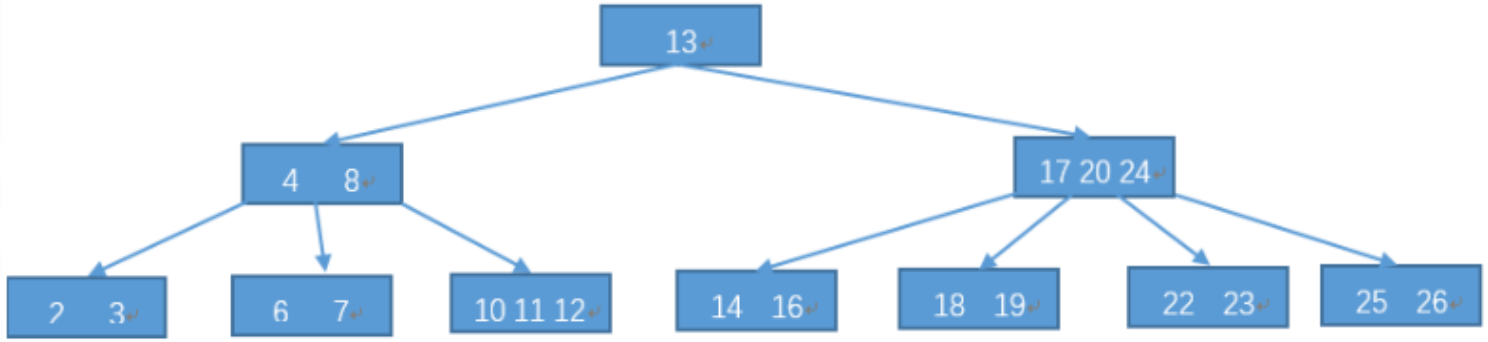
更新和删除：由于LSM-Tree只能追加写，不能原地更新和删除，只能在Segment Compaction的时候进行真正地更新和删除。



聊聊MongoDB时序集合性能

MongoDB 中文社区

B Tree



读写：优势是支持高效的读，范围搜索查询优势明显。但在大规模写请求下，B树需要维护树结构，节点分裂。读磁盘的随机读写概率就会变大，性能会逐渐减弱。

更新和删除：一个key只会出现一个Page页里面，能做到原地更新和删除。

聊聊MongoDB时序集合性能

MongoDB中文社区

MongoDB时序集合使用限制：

- 客户端加密
- ChangeStream
- ReIndex重建索引
- Tiggers
- 更新和删除限制

MongoDB时序iot场景设计

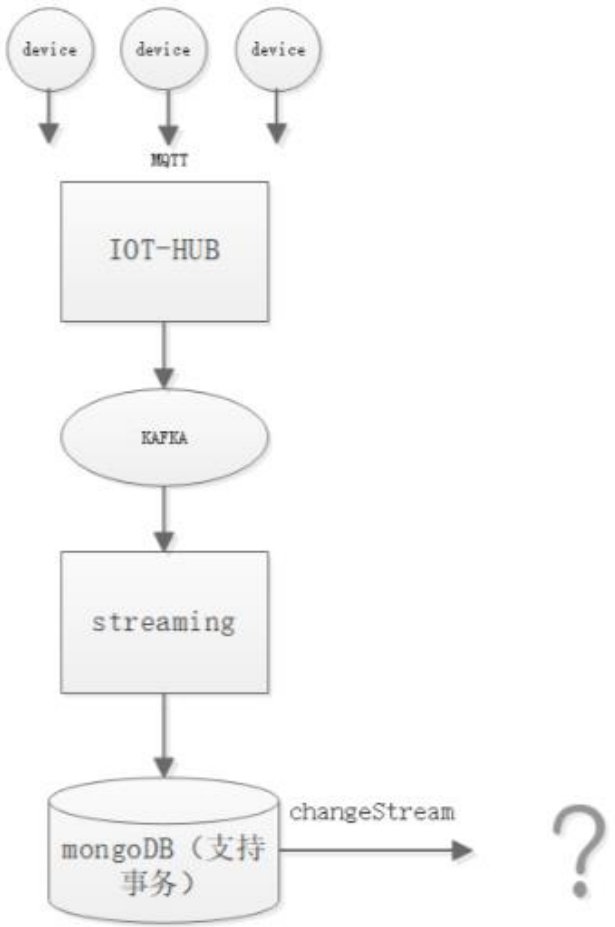
MongoDB时序iot场景设计

MongoDB 中文社区

场景需求：

数据质量，实时消费kafka数据，并经过流式计算后，需要对数据进行展示，如右侧流程图：

1. 时序集合
2. 读写分离
3. ChangeStream 分流查询



MongoDB时序iot场景设计

MongoDB 中文社区

过期数据清理：

1.可以采用时序集合原生态的TTL索引进行自动过期。

2.可以通过新旧集合替换的方式，对旧集合直接删除的方式。

MongoDB
中文社区

Thank You!

Mongoing中文社区

微信扫描二维码，关注我的公众号