主讲老师: Fox

```
文档：5.MongoDB建模调优&change stream实战....
链接：http://note.youdao.com/noteshare?id=06f45164f11e98d0ec81f86129a4ceac&sub=4463961331D54DFB8F273188AD59FBB5
```


MongoDB开发规范

（1）命名原则。数据库、集合命名需要简单易懂，数据库名使用小写字符，集合名称使用统一命名风格，可以统一大小写或使用驼峰式命名。数据库名和集合名称均不能超过64个字符。

（2）集合设计。对少量数据的包含关系，使用嵌套模式有利于读性能和保证原子性的写入。对于复杂的关联关系，以及后期可能发生演进变化的情况，建议使用引用模式。

（3）文档设计。避免使用大文档，MongoDB的文档最大不能超过16MB。如果使用了内嵌的数组对象或子文档，应该保证内嵌数据不会无限制地增长。在文档结构上，尽可能减少字段名的长度，MongoDB会保存文档中的字段名，因此字段名称会影响整个集合的大小以及内存的需求。一般建议将字段名称控制在32个字符以内。

（4）索引设计。在必要时使用索引加速查询。避免建立过多的索引，单个集合建议不超过10个索引。MongoDB对集合的写入操作很可能也会触发索引的写入，从而触发更多的I/O操作。无效的索引会导致内存空间的浪费，因此有必要对索引进行审视，及时清理不使用或不合理的索引。遵循索引优化原则，如覆盖索引、优先前缀匹配等，使用explain命令分析索引性能。

（5）分片设计。对可能出现快速增长或读写压力较大的业务表考虑分片。分片键的设计满足均衡分布的目标，业务上尽量避免广播查询。应尽早确定分片策略，最好在集合达到256GB之前就进行分片。如果集合中存在唯一性索引，则应该确保该索引覆盖分片键，避免冲突。为了降低风险，单个分片的数据集合大小建议不超过2TB。

（6）升级设计。应用上需支持对旧版本数据的兼容性，在添加唯一性约束索引之前，对数据表进行检查并及时清理冗余的数据。新增、修改数据库对象等操作需要经过评审，并保持对数据字典进行更新。

（7）考虑数据老化问题，要及时清理无效、过期的数据，优先考虑为系统日志、历史数据表添加合理的老化策略。

（8）数据一致性方面，非关键业务使用默认的WriteConcern：1（更高性能写入）；对于关键业务类，使用WriteConcern：majority保证一致性（性能下降）。如果业务上严格不允许脏读，则使用ReadConcern：majority选项。

（9）使用update、findAndModify对数据进行修改时，如果设置了upsert：true，则必须使用唯一性索引避免产生重复数据。

（10）业务上尽量避免短连接，使用官方最新驱动的连接池实现，控制客户端连接池的大小，最大值建议不超过200。

（11）对大量数据写入使用Bulk Write批量化API，建议使用无序批次更新。

（12）优先使用单文档事务保证原子性，如果需要使用多文档事务，则必须保证事务尽可能小，一个事务的执行时间最长不能超过60s。

（13）在条件允许的情况下，利用读写分离降低主节点压力。对于一些统计分析类的查询操作，可优先从节点上执行。

（14）考虑业务数据的隔离，例如将配置数据、历史数据存放到不同的数据库中。微服务之间使用单独的数据库，尽量避免跨库访问。

（15）维护数据字典文档并保持更新，提前按不同的业务进行数据容量的规划。

MongoDB调优

三大导致MongoDB性能不佳的原因：

1. 慢查询
2. 阻塞等待
3. 硬件资源不足

1,2通常是因为模型/索引设计不佳导致的

排查思路：按1-2-3依次排查

影响MongoDB性能的因素

https://www.processon.com/view/link/6239daa307912906f511b348

MongoDB建模小案例分析

MongoDB性能监控工具

性能问题排查参考案例

记一次 MongoDB 占用 CPU 过高问题的排查

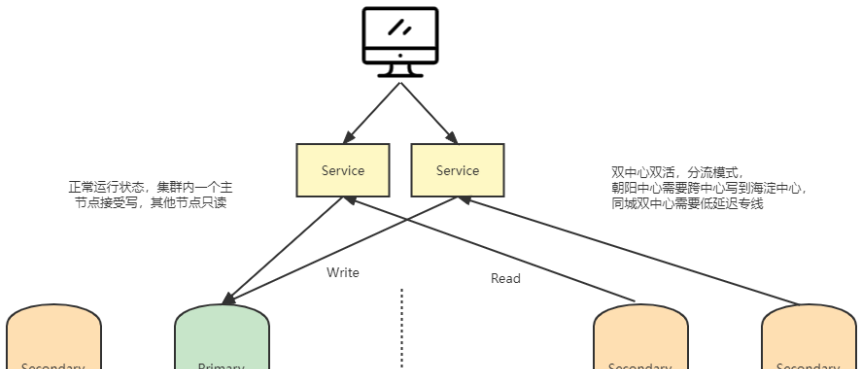
MongoDB线上案例：一个参数提升16倍写入速度

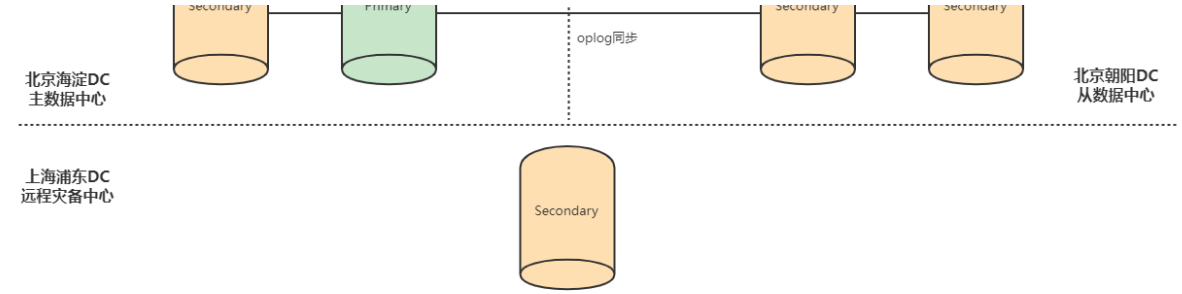
MongoDB高级集群架构

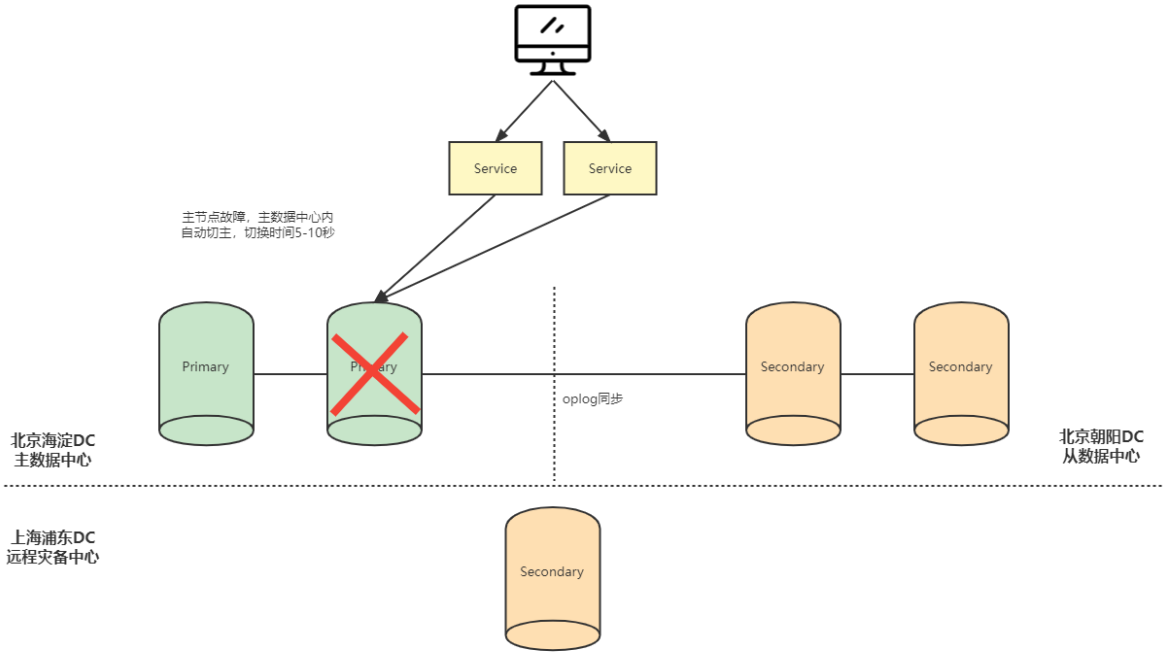
两地三中心集群架构

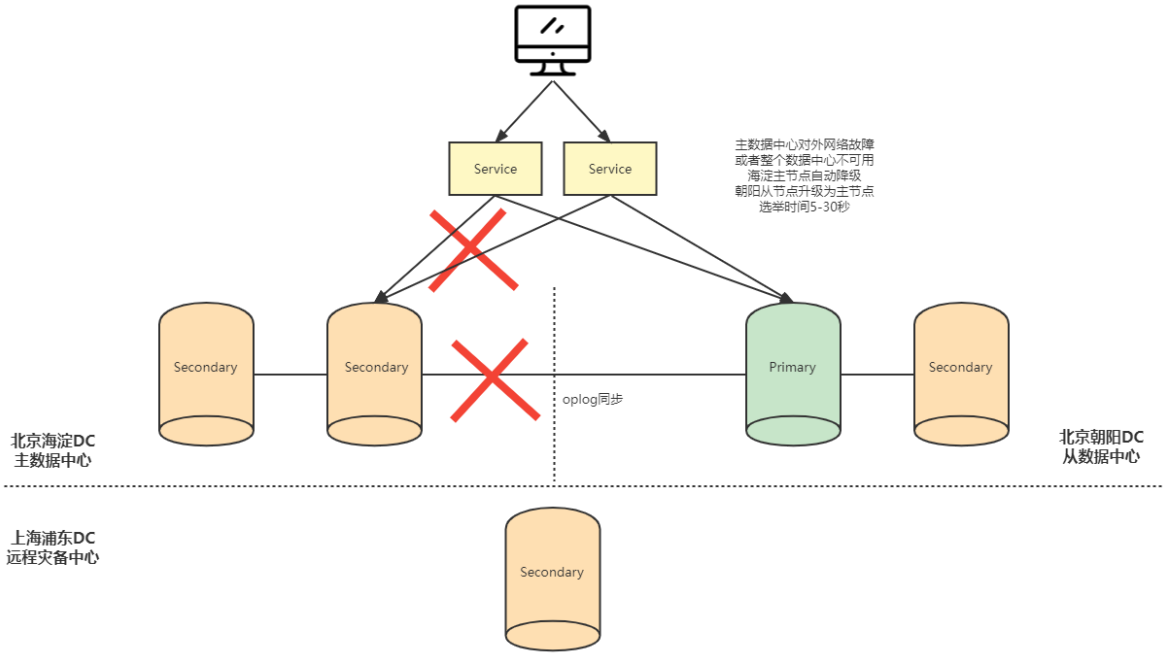
https://www.processon.com/view/link/6239de401e085306f8cc23ef

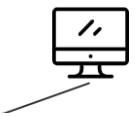
双中心双活 + 异地热备=两地三中心

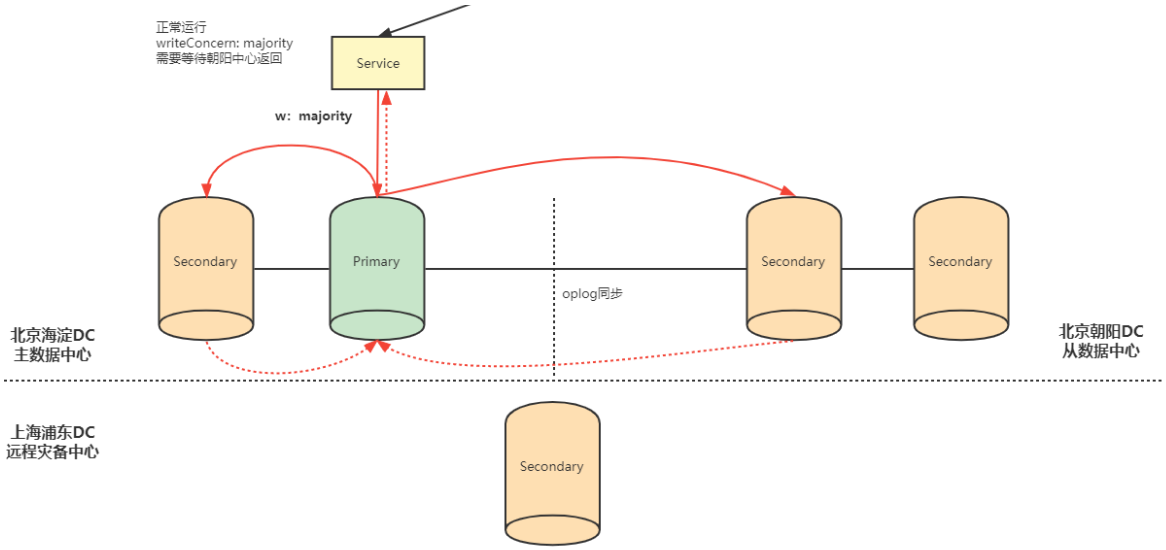



oplog同步
北京海淀DC
主数据中心
北京朝阳DC
从数据中心
上海浦东DC
远程灾备中心
Secondary


Service
Service
主节点故障，主数据中心内
自动切主，切换时间5-10秒
Primary
Secondary
Secondary
oplog同步
北京海淀DC
主数据中心
北京朝阳DC
从数据中心
上海浦东DC
远程灾备中心
Secondary

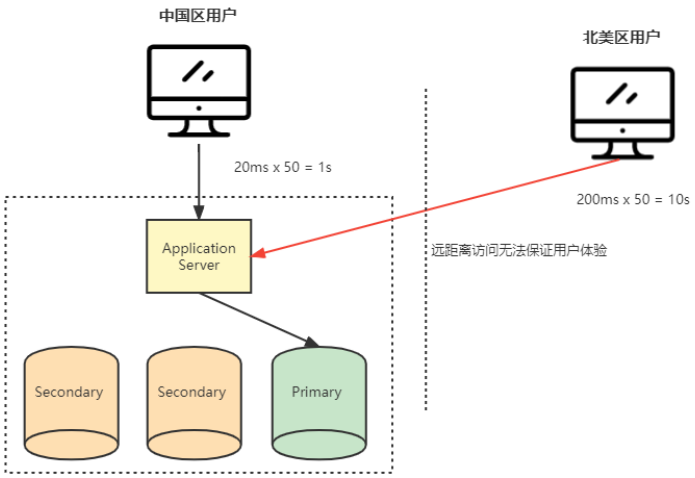

Service
Service
主数据中心对外网络故障
或者整个数据中心不可用
海淀主节点自动降级
朝阳从节点升级为主节点
选举时间5-30秒
Secondary
Secondary
Primary
Secondary
oplog同步
北京海淀DC
主数据中心
北京朝阳DC
从数据中心
上海浦东DC
远程灾备中心
Secondary

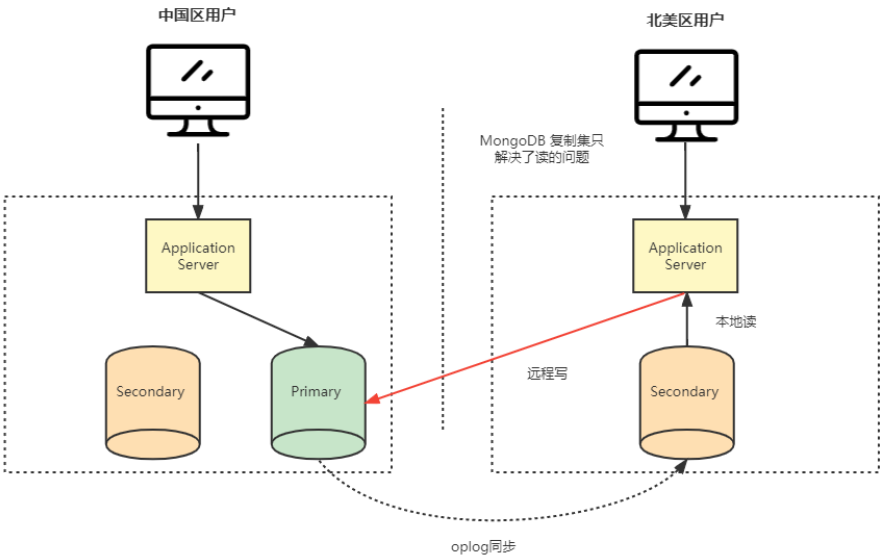


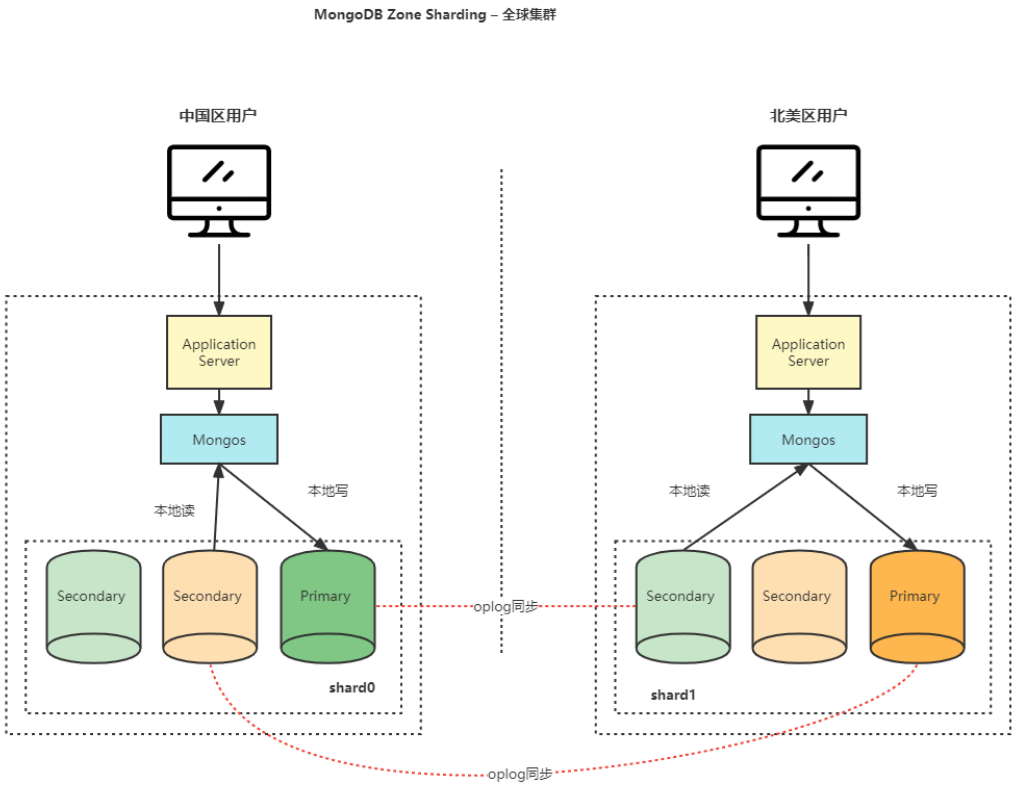


全球多写集群架构

https://www.processon.com/view/link/6239de277d9c08070e59dc0d





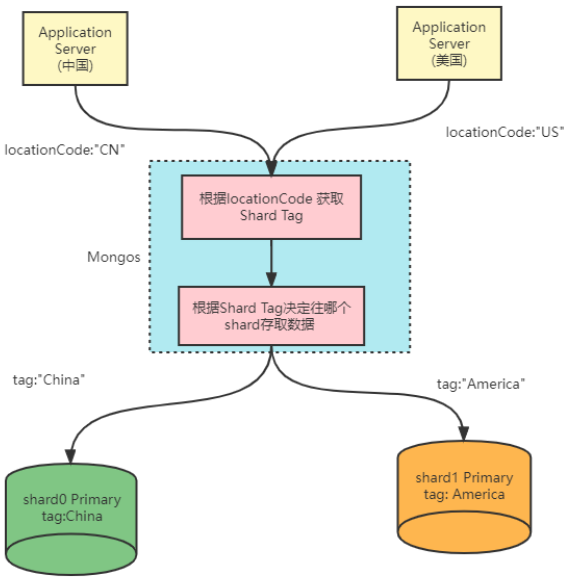


Zone Sharding 设置步骤
1.针对每个要分片的数据集合，模型中增加一个区域字段
{
key:value,
...
locationCode:"CN" //应用程序按照规则，自动加上这个字段值
}

2.给集群的每个分片加区域标签
sh.addShardTag("shard0" , " China")

3.给每个区域指定属于这个区域的分片块范围（chunk range）

```
sh.addTagRange( "crm.orders",
{ "locationCode" : "CN", "order_id" : MinKey },
{ "locationCode" : "CN", "order_id" : MaxKey },
 "China" )
```



MongoDB整合SpringBoot

文档CRUD操作&聚合操作

视频教程： https://vip.tulingxueyuan.cn/detail/p_622d92aee4b066e9608ee2c9/6

事务操作

官方文档： https://docs.mongodb.com/upcoming/core/transactions/

编程式事务

```
/**
 * 事务操作API
 * https://docs.mongodb.com/upcoming/core/transactions/
 */
@Test
public void updateEmployeeInfo() {
 //连接复制集
 MongoClient client = MongoClients.create("mongodb://fox:fox@192.168.65.174:28017,192.168.65.174:28018,192.168.65.174:28019/test?authSource=admin&replicaSet=rs0");

 MongoCollection<Document> emp = client.getDatabase("test").getCollection("emp");
 MongoCollection<Document> events = client.getDatabase("test").getCollection("events");
 //事务操作配置
 TransactionOptions txnOptions = TransactionOptions.builder()
 .readPreference(ReadPreference.primary())
 .readConcern(ReadConcern.MAJORITY)
 .writeConcern(WriteConcern.MAJORITY)
 .build();
 try (ClientSession clientSession = client.startSession()) {
 //开启事务
 clientSession.startTransaction(txnOptions);

 try {

 emp.updateOne(clientSession,
 Filters.eq("username", "张三"),
 Updates.set("status", "inactive"));

 int i=1/0;

 events.insertOne(clientSession,
 new Document("username", "张三").append("status", new Document("new", "inactive").append("old", "Active")));

 //提交事务
```

```
 clientSession.commitTransaction();

 }catch (Exception e){
 e.printStackTrace();
 //回滚事务
 clientSession.abortTransaction();
 }
 }
}
```

声明式事务

配置事务管理器

```
@Bean
MongoTransactionManager transactionManager(MongoDatabaseFactory factory){
 //事务操作配置
TransactionOptions txnOptions = TransactionOptions.builder()
 .readPreference(ReadPreference.primary())
 .readConcern(ReadConcern.MAJORITY)
 .writeConcern(WriteConcern.MAJORITY)
 .build();
 return new MongoTransactionManager(factory);
}
```

编程测试service

```
@Service
public class EmployeeService {

 @Autowired
 MongoTemplate mongoTemplate;

 @Transactional
 public void addEmployee(){
 Employee employee = new Employee(100,"张三", 21,
  15000.00, new Date());
  Employee employee2 = new Employee(101,"赵六", 28,
  10000.00, new Date());

  mongoTemplate.save(employee);
  //int i=1/0;
  mongoTemplate.save(employee2);

  }
```

```
19 }
```

测试

```
1 @Autowired
2 EmployeeService employeeService;
3
4 @Test
5 public void test(){
6  employeeService.addEmployee();
7
8 }
```

change stream实战