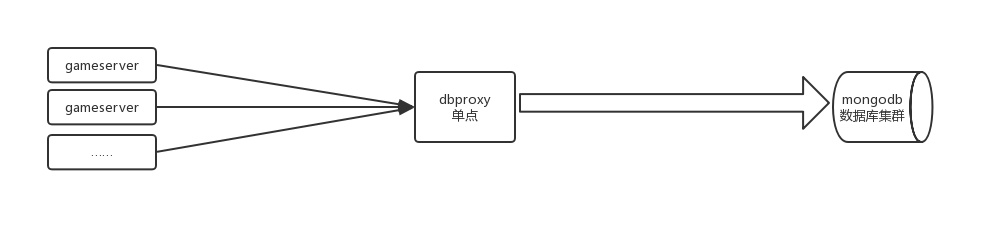
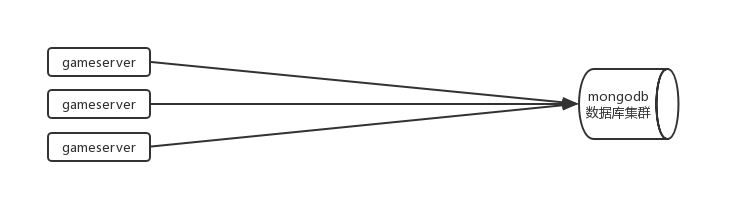
### 部署图：

通过技术中心dbproxy 接入mongodb集群



原生mongo-c driver接入mongodb集群



原生mongo-c driver连接：逻辑服务器封装异步接口直接通过官方接口操作数据库集群。

通过dbproxy连接：逻辑服务器通过网络消息与dbproxy中转操作数据库集群，部署消耗的机器增加（据技术中心之前提供的数据，每30000QPS增加一台dbproxy机器，主备模式翻倍）。

### 在线动态扩容：

|  |  |
| --- | --- |
| mongo-c driver连接 | dbproxy连接 |
| 需要在线动态扩容mongodb集群 | 需要在线动态扩容mongodb集群 |
|  | 需要新增dbproxyserver服务（旧逻辑服可能还需重启） |

原生mongo-c driver连接：直接操作数据库集群，只需要在数据量增大的时候扩容mongodb集群即可，原生支持在线动态扩容，完美满足海量数据存储功能。

通过dbproxy连接：dbproxy本身只能单点或多个单点部署，单纯扩容mongodb集群的话dbproxy将会成为瓶颈，无法做到动态扩容。

### 接入工作：

|  |  |
| --- | --- |
| mongo-c driver连接 | dbproxy连接 |
| 封装mongo-c driver相关操作接口给逻辑服使用 | 封装游戏相关操作db接口 |
|  | 封装逻辑服务器和dbproxy消息协议（这个协议本身比较难用，可能还需为他封装一套发送网络消息的接口） |
|  | lua脚本处理逻辑服消息协议 |
|  | lua脚本操作数据库 |
|  | 若dbproxy需要集群部署或主备部署需要逻辑服自己封装连接管理 |

### 集群操作接入：

|  |  |
| --- | --- |
| mongo-c driver连接 | dbproxy连接 |
| mongo-c driver与数据库集群之间原生支持url连接，只需在访问url里配上多个连接地址即可 | 需要逻辑服自己封装连接管理（极大概率引入bug和不稳定因素，非大规模并发不一定能测试出来） |

原生mongo-c driver连接：mongo-c driver原生支持mongodb集群连接，只需要通过url配置集群地址即可。

通过dbproxy连接：dbproxy本身只能单点或多个单点部署，没有提供接入sdk，需要逻辑服自己封装操作协议。若需要dbproxy主备部署或集群部署,则只能逻辑服自己封装连接管理，工作量极大，而且极大概率引入bug和不稳定因素，并且非大规模并发不一定能测试出来。

### 稳定性：

|  |  |
| --- | --- |
| mongo-c driver连接 | dbproxy连接 |
| 运维保障mongodb集群本身的稳定性 | 运维保障mongodb集群本身的稳定性 |
| mongo-c driver本身的稳定性 | mongo-go driver本身的稳定性 |
|  | dbproxy本身引入的不确定性 |
|  | dbproxy集群部署或主备部署时逻辑服自己封装的连接管理带来的大量不稳定性 |

原生mongo-c driver连接：运维保障mongodb集群本身的稳定性。mongo-c driver是经过大量工程验证过的官方代码，许多其他语言的mongo操作库都是通过 mongo-c driver封装而来，mongo-c driver本身的稳定性可以相信。

通过dbproxy连接：运维保障mongodb集群本身的稳定性。mongo-go driver的稳定性和mongo-c driver稳定性类似，可以相信。但是dbproxy本身会引入不确定性，逻辑服自己封装相应的协议、lua操作库都会引入不确定性。若是集群部署和主备部署dbproxy，则会面对多链接管理带来的大概率不确定性。

### 功能灵活性：

|  |  |
| --- | --- |
| mongo-c driver连接 | dbproxy连接 |
| 官方提供大量精细化db操作：方便的单个属性修改查询、原子操作、事务操作 | 通过dbproxy中转后在lua操作dbproxy：若需要更多精细化操作则需要添加大量dbproxy协议和lua代码，原子操作和事务操作未研究是否支持 |

### 逻辑维护：

|  |  |
| --- | --- |
| mongo-c driver连接 | dbproxy连接 |
| 各模块维护好数块序列化和反序列化 | 各模块维护好数块序列化和反序列化 |
|  | 修改相应dbproxy协议 |
|  | 修改相应lua脚本 |

### 整体流程和性能：

|  |  |
| --- | --- |
| mongo-c driver连接 | dbproxy连接 |
|  | 各模块数据序列化成dbproxy网络协议 |
|  | 通过网络层发送给dbproxy服务器 |
| 各模块数据序列化成bson结构 | lua脚本将协议序列化成bson结构 |
| 调用数据库线程的mongo-c driver数据操作接口 | 调用mongo-go driver数据操作接口 |
| 返回bson结构 | 返回bson结构 |
| 逻辑层直接解析bson数据为模块数据 | lua脚本解析bson结构 |
|  | lua脚本构造返回的dbproxy协议 |
|  | dbproxy通过网络层发送给逻辑服 |
|  | 逻辑服解码返回协议成pb |
|  | 逻辑层将pb数据处理成模块数据 |

### 数据操作时序

|  |  |
| --- | --- |
| mongo-c driver连接 | dbproxy连接 |
| 多个数据库线程，相同玩家ID分配固定线程 | lua脚本分配操作线程 |
| 公共数据通过原子操作和事务模式保证数据正确 | dbproxy本身是否支持事务和原子操作 |

### 新增Mysql支持成本

|  |  |
| --- | --- |
| mongo-c driver连接 | dbproxy连接 |
| 引擎封装相应的Mysql操作库 | dbproxy服务器封装相应的Mysql操作库 |
| 各数据模块封装Mysql格式数据序列化和反序列化 | 各数据模块封装mysql操作协议 |
|  | lua脚本编解码相应协议 |
|  | lua脚本支持各协议mysql操作 |