~~>改存储为key buf加写入版本；新加搜索任务化~~

~~>k桶的map化~~

~~>数据的改写；~~

~~发起节点在搜索完成后，发起改写操作。~~

~~发送announce peer给目标节点；~~

~~设置监督节点ip,设置序列数为0；~~

~~目标节点收到后写入数据，更改序列加一；~~

~~给监督节点和发送节点返回设置成功；~~

~~发送节点没有相关任务忽略返回；~~

~~监督节点记录数据；~~

~~错误处理延后；~~

~~>数据的查询查流程连续查询多个节点；~~

~~发送给共识的c节点直接返回结果。~~

~~>storage的map化~~

~~>引入gooisp广播解决DHT网络树形化分区化的问题。~~

>节点的增加，如果是头节点要删除后一个节点的头节点标志。其他节点忽略

只有数据为空白的情况下才能写入；

后节点有的ap数据要同步；

后节点的是首节点数据要比较是否上节点是否有所有权，如果有转移首节点所有权。

>节点的删除，检测到节点删除要通知头节点发起数据的延后。注意延后的数据以最后的节点为准而不是头节点，防止较旧的数据覆盖新数据，这种触发是一个数据同步指令，而不是数据改写指令。

>cmake的安装脚本;

>引入sqlite;

>通过控制命令访问的双向通道引入json----线程分离延后

>debug信息封包化。

>脚本化写入请求和查询请求

按整个网络可用率为80%

写入流程，找到可用的C发起写入，C写入下一个A节点为写入成功，否则返回写入失败。

为了保证C节点连续同步5个A节点能够成功。因为顺序写入5个节点的成功概率为=0.32，那么每天至少要同步3次才能保证数据同步成功。那么每7个小时数据就要同步一次，按带宽10M/s算7小时可以同步的数据总量为246G。那么每台服务器可以维护的数据总量保守估计在100G左右。

在写入C节点后因为C节点又80%的概率会消失，那么需要立即发起同步，而同步的成功的概率是0.32，但只写入到下一个节点的概率为0.64。并且一旦写入成功，C和同时崩溃的概率是0.04。那么只要在C和同时崩溃前再次发起数据同步就有极大的概率成功同步其他3个节点。

Ping的连通率是x,如果连续3次ping不到达就判定服务器不可用，连续崩溃率导致服务器不可使用的概率为。因为这里的单位为每秒，那么在集群内服务器崩溃的间隔时间s为。设集群连续n次崩溃会导致集群数据丢失，那么就要在的时间内修复数据。

如果已知修复集群服务器的所需最短时间为那么服务器的连通率能够承受的上限为 。