ZDB2.0 内核简要技术介绍

数据分配表表

ZDB2.0 的存储思路采用统一空间分配表+数据组成。这一思路，与我们常用的硬盘分区的思路是一致的：定义好空间和存储单元大小，然后，开始数据读写。

ZDB2.0 分配表的存储是链条模型：分配表 1，2，3，在空间打开时，分配表 1,2,3 将会组合成一张统一分配表。该模型可以做到灵活空间配置，我们可以使用本地 hdd 做分配表 1 的存储，用网络驱动器做分配表 2 的存储，当 ZDB2.0 被打开时它会自动启动分布式存储。而大多数时候，这种模型多用于扩容需要，例如我们从 10G 扩容到 100G，我们只需要生成一个 90G 的分配表和存储空间，然后加入分配表存储链条。

关于空间呢分配表简单举个例子，

分配表 1 {1,2,3}

分配表 2 {4,5,6}

当 ZDB 空间被打开时候，这些链条形式的数据将会转换成**统一分配表 {1,2,3,4,5,6}**

ZDB2.0 的空间分配表与加密技术是互相结合的，在没有秘钥时，数据和空间分配表是不可访问的，我们在 open 空间时就会出错。

IO 接口

ZDB2.0 的底层 IO 使用了带有 Cache 机制的 umlFile 支持系列，既读写都会有 Cache 效果，在对付连续性读写 IO 操作时，umlFile 支持系列的 API 有明显提速。umlFile 支持系列可以直接在 Stream 上转存，换句话说，任何接口过 Stream 的 API 都可以作为 ZDB2.0 的底层 IO，包括 TFileStream,TMemoryStream，或则是来自 ZDB1 的 TItemStream。

数据存储原理

分配表的每个元素是存储单元，存储单元使用 ID 表示，同时它们都会指向一个物理存储地址。

当我们的数据有 100K 时，而我们的分配表每个是 10K，那么数据将会占用 10 个存储单元，同样的，会使用 10 个 ID 表示数据存储的位置。

例如写入 100K 数据如下

Write(100K 数据) 返回{1,2,3,4,5,6,7,8,9,10}，既表示这个数据的句柄

这时候，我们需要记下存放这些数据的单元 ID 才能够进行下次的读取操作

read(1,2,3,4,5,6,7,8,9,10) -> (100k 数据)

另外一种更简单方式，read(1) -> (100k 数据)，其中 read(1)是 ZDB2.0 自动寻找关联数据的机制，同样的，read(10) -> (100k 数据)

大数据和内存存储

大数据存储支持基于 Stream 工作，通常是我们使用的 FileStream。这种方式读写 Stream 不会使用过多使用内存，而是让全部数据通过 IO 传递，这种读写操作数据几乎是无限大。

内存存储是根据 TMem64 进行操作，需要预分配内存。

在使用了 Cipher 接口后，数据写操作会进行加密处理，数据读操作则进行解密处理。

大数据的存储会产生非常巨大的单元 ID 数据，上百万个 ID {1,2,3....1000000}，我们要全部记录下来将非常耗费系统资源，因此，只需要记录第一个 ID 即可，使用自动数据关联功能。

空间分配技术

ZDB2.0 的存储空间分配都是尽量采取连续性空间为主。使用探头技术按找空闲单元存储，例如分配表有 100 万个 ID，探头会高速搜索可用存储空间，然后再以连续性方式分配出来。这种分配方式优于红黑树和 HASH 表。并且，连续性空间机制在硬件和操作系统中有最天然的 Cache 支持。

数据删除和擦写

数据删除是重置空间分配表的存储单元，但是数据仍然在，这时候，ZDB2.0 会以 0 重新填充数据，达到数据保护的作用。

空间管理

空间管理是指我们在使用 ZDB2 时，需要预置分配一个存储空间，之后，我们必须在既定的空间中进行存储操作。

空间扩容是指追加一个新的空间分配表在 ZDB2.0 的数据库末尾，扩容技术是优化过的，可以在运行过程中，发现存储空间不够通过 AppendSpace 方法进行自动化扩容。

空间优化是将所有的空间分配表重新整理成一张空间分配表，并且将会对数据按线性重新构建。空间优化会发生大数据的 Copy 操作，当数据规模堆大以后，该操作需要谨慎使用。

扩展支持

在 ZDB2.0，文件压缩解压，存储单元 CRC，空间计划，遍历查询，都是作为一种扩展来支持。这些扩展广泛应用了 IO 线程，并行，线程池等技术。

完