Khipu 的存储引擎
Kesque

@jolestar

Khipu 是什么

Khipu ['kʰipu], 记簿/结谱），
是古印加人的一种结绳记事的方法，
用来计数或者记录历史。

A Scala/Akka implementation of the Ethereum protocol

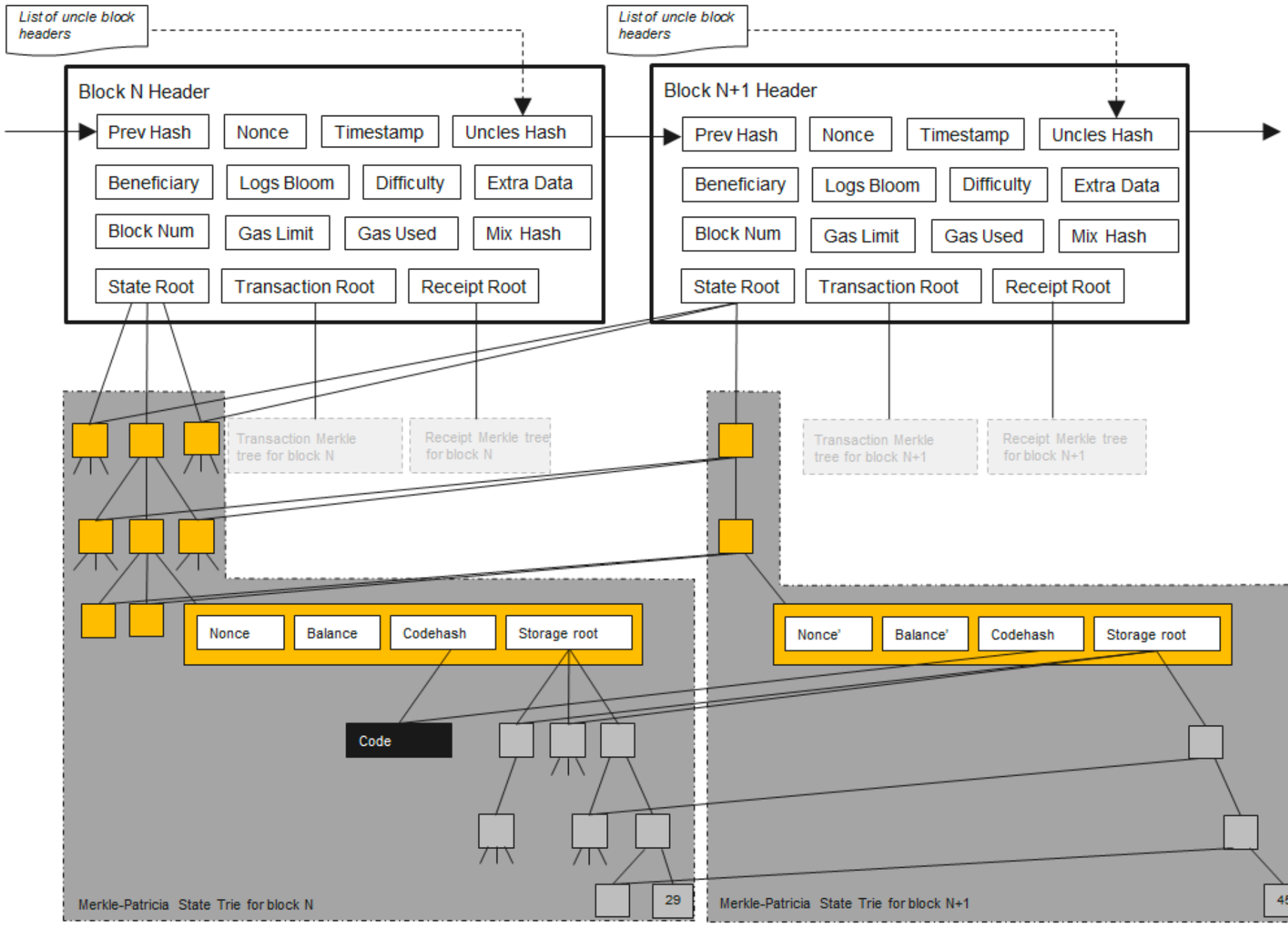
https://github.com/khipu-io/khipu

- 并行合约执行
- 存储引擎 Kesque

为什么需要一个新的存储引擎?

理解以太坊存储

- State Trie/ Storage Trie
- Trie Node to Key-Value



以太坊存储的特点

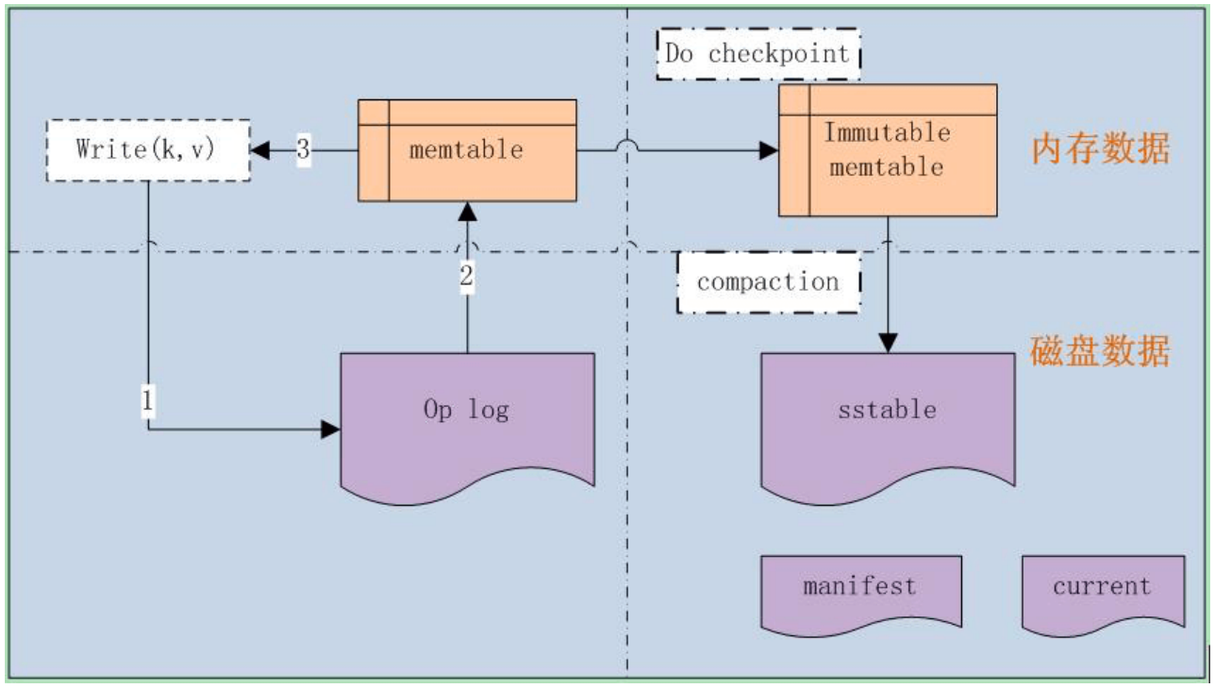
- Key 主要是 Hash（随机分布），Key 随着 Value 变化
- 一个节点变化可能触发多次写操作
- 一个读操作可能触发多次读操作，并且读操作前后相关
- 几乎不存在更新的需求，主要是增删

Everything is identified as Hash. 除了本身就是 Hash 的一个 Address，以太坊里所有东西都在变动中，一点最微小的变动，前一秒的“你”就不再是“你”，因为新“你”的 key 就是新“你”的 hash。历史长河里留下一个个过去的“你”，你无法修改它们、或把它们删去。

–邓草原

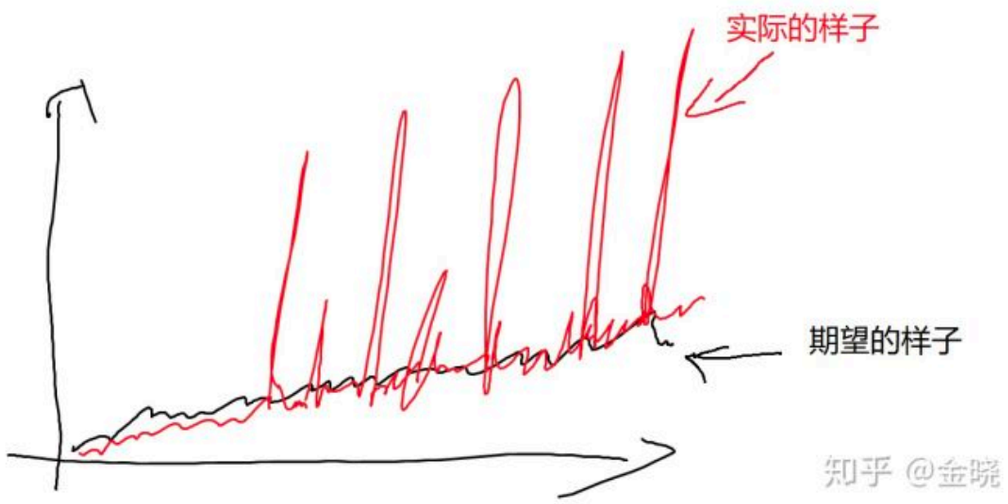
LevelDB 的特点

- 数据按 Key 有序存储，利于按 Key 顺序检索



- 写操作性能优越（AppendLog，Memtable）

- 写放大/读放大导致性能不稳定



Kesque

```
/**
 * Kesque (Kafkaesque)
 *
 * What's Kafkaesque, is when you enter a surreal world in which all your
 * control patterns, all your plans, the whole way in which you have configured
 * your own behavior, begins to fall to pieces, when you find yourself against a
 * force that does not lend itself to the way you perceive the world. You don't
 * give up, you don't lie down and die. What you do is struggle against this
 * with all of your equipment, with whatever you have. But of course you don't
 * stand a chance. That's Kafkaesque. - Frederick R. Karl
 * https://www.nytimes.com/1991/12/29/nyregion/the-essence-of-kafkaesque.html
 *
```

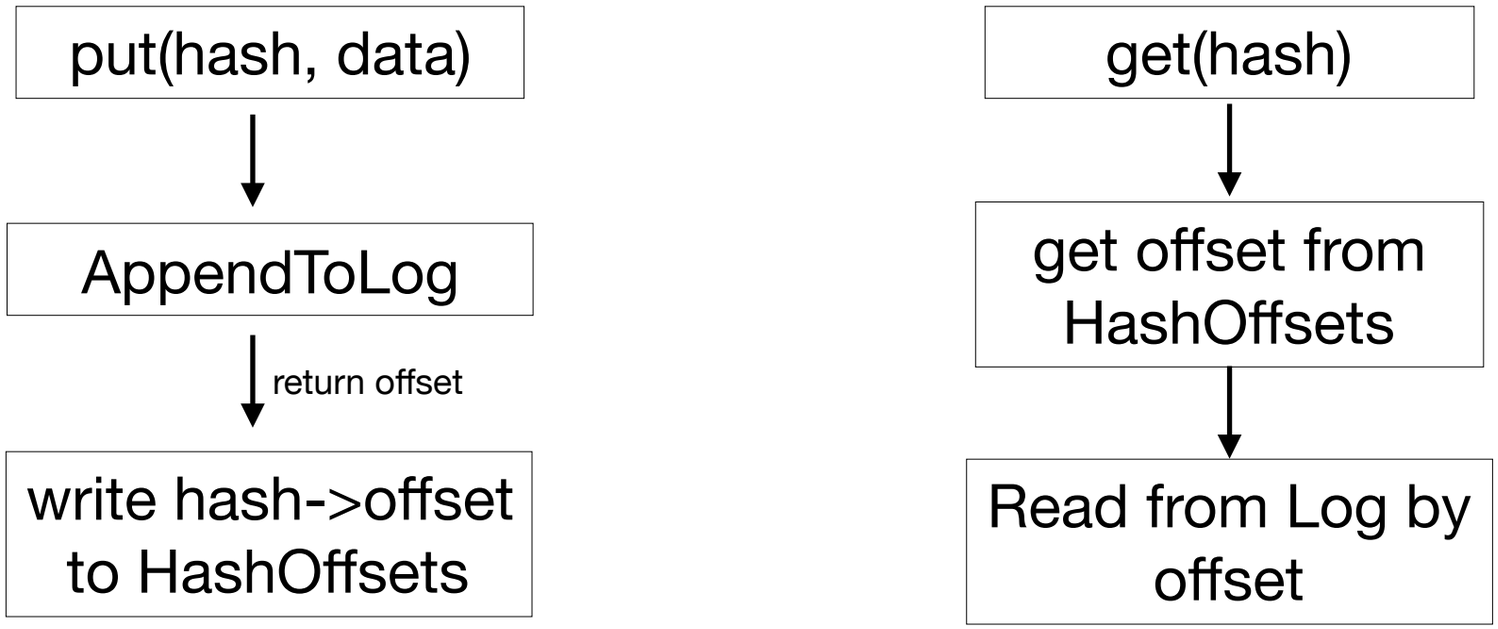
『卡夫卡式』是当你进入一个超现实的世界，而你所有的支配模式、所有的计划、编制自身行为的整套方法都开始分崩离析。你能做的就是用自身拥有的全部资源与其对抗。不过当然，你沒有任何机会成功。这才是卡夫卡式。

Kesque

- Kafka LogManager（AppendEnd to Log)
- HashKeyValueTable(index,cache)
- HashOffsets（All in memory)

Kesque

数据落盘的 HashTable



Kesque

HashOffsets<int(hash[0:4]),offset> int->int, int->ints

```
/**
 * It's acutally an Int to Ints map
 * We introduce one extra pairs of fields - for key=0, which is used as 'used' flag
 * Memoey usage:
 *   Int -> Int[]
 * Key (Int) in bytes: 4
 * Value (Int[]) in bytes: (16(reference) + 4(length) + 4(align)) + 4 * N = 24 + 4 * N
 * KV in Bytes: 28 + 4 * N
 *
 * There are about 99.6% keys have only 1 value, i.e. 28 + 4 = 32 bytes.
 * Thus 100,000,000 kvs in bytes: 100,000,000 * 32 / 1024 / 1024 / 1024 = 3G
 */
```

当前以太坊大约有 2.5 亿个 kv

在百G级别数据的情况下，kesque 随机读的速度比 leveldb 高一个数量级

Tree vs Hash

- 索引成本
- 访问方式
- 数据特点

Khipu 值得学习的地方

- EventStream/Actor 设计模式
- 合约的并行处理
- Scala

邓草原

9月28日 13:41 来自 微博 weibo.com

我需要人手帮忙，可以先从提 bug 开始，有 PR 更好。工作量太大了。。。//@响马: 与其等 eth，不如等邓老师。

@邓草原

经过近半年的开发，khipu 的性能已经能到目前最快的以太坊节点软件 parity 的 2/3。在一台 32G 内存、SATA SSD 的机器上，parity 每秒能执行的 gas 量约为 14.5 mgas/s，khipu 约为 9.8 mgas/s。我还没时间去弄明白为什么 parity 的 db 能维持在 32G 左右，khipu 和 geth 的这个量应该都在 100G 以上。

9月28日 12:10 来自 微博 weibo.com 已编辑　14　6　10