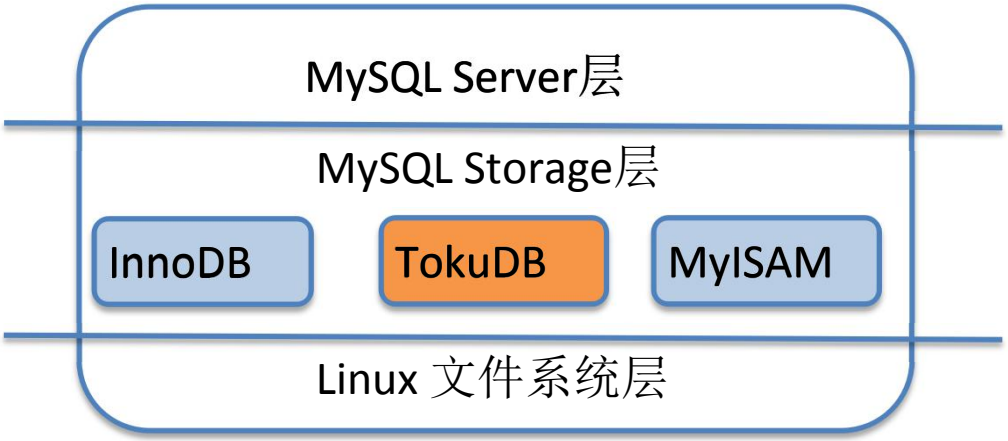
TokuDB索引结构

网易杭州研究院---胡争（博客：**openinx.github.io**）

TokuDB简介

- 基于分形树实现的MySQL存储引擎
- Tokutek公司2007年研发，2013年开源
- 2015年Percona公司收购Tokutek公司
- TokuDB内部的K-V存储引擎为ft-index
- TokuMx: ft-index + MongoDB Server层代码

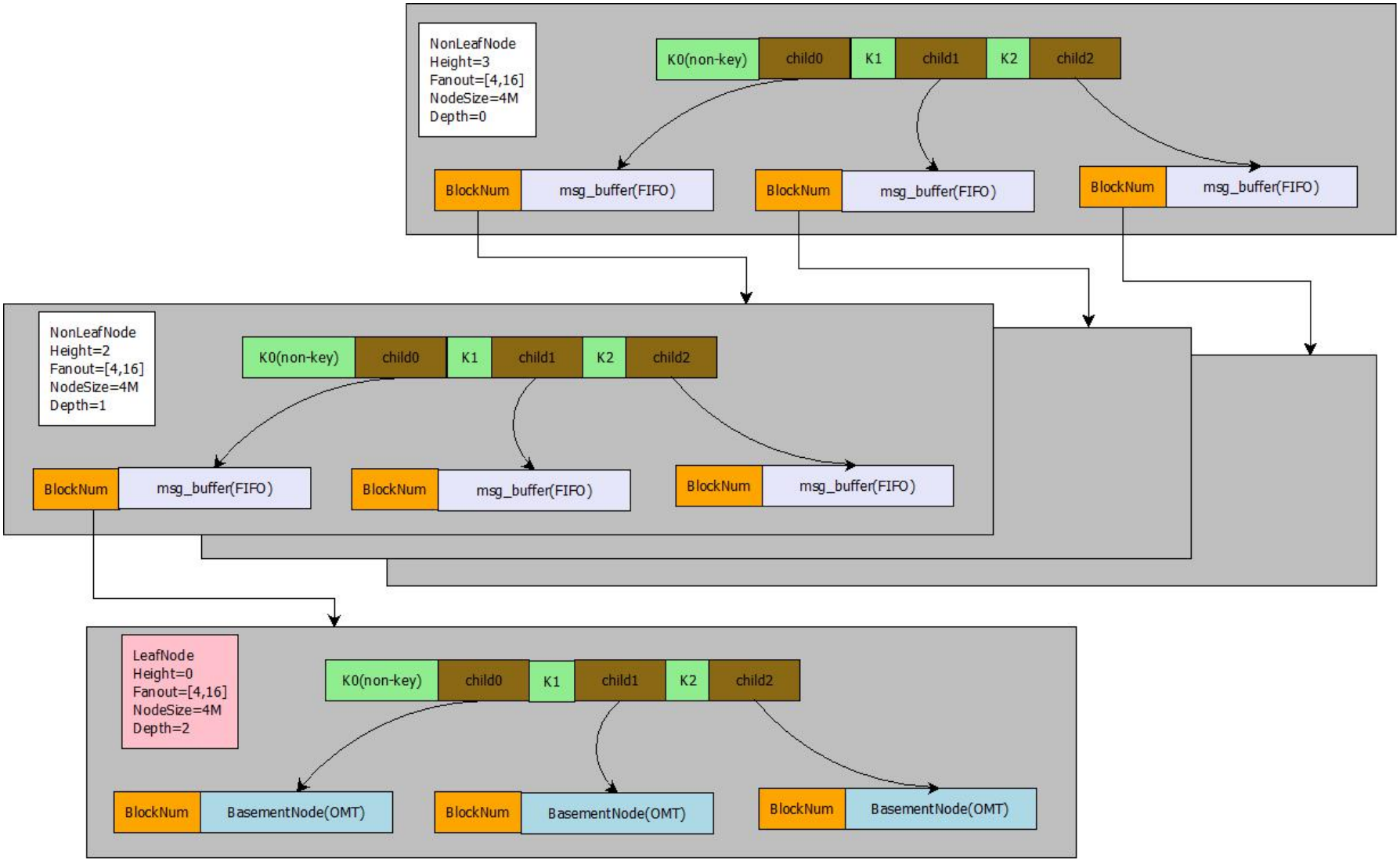


TokuDB特点

- 支持事务（ACID）的MySQL存储引擎
- 插入性能大大高于InnoDB（分形树vs B+树）
- 查询性能略低于InnoDB
- 在线执行DDL操作（不阻塞写操作）
- 超高压缩率（TokuDB 4M vs InnoDB 16K）

更高性能，更低成本！

分形树索引结构（一）


NonLeafNode
Height=3
Fanout=[4,16]
NodeSize=4M
Depth=0
K0(non-key)
child0
K1
child1
K2
child2
BlockNum
msg_buffer(FIFO)
BlockNum
msg_buffer(FIFO)
BlockNum
msg_buffer(FIFO)
NonLeafNode
Height=2
Fanout=[4,16]
NodeSize=4M
Depth=1
K0(non-key)
child0
K1
child1
K2
child2
BlockNum
msg_buffer(FIFO)
BlockNum
msg_buffer(FIFO)
BlockNum
msg_buffer(FIFO)
LeafNode
Height=0
Fanout=[4,16]
NodeSize=4M
Depth=2
K0(non-key)
child0
K1
child1
K2
child2
BlockNum
BasementNode(OMT)
BlockNum
BasementNode(OMT)
BlockNum
BasementNode(OMT)

分形树结构（二）

- msg_buffer
  - 先进先出队列
- BasementNode（OMT）
  - 弱平衡二叉树
  - 增删改查期望复杂度O(logN)
- 页大小默认4M。
- 扇出fanout默认[4,16]区间。

分形树结构（三）

- **叶子节点**
  - 数据量维持在[1M,4M]区间
  - 数据量小于1M则合并
  - 数据量大于4M则分裂。

- **非叶子节点**
  - 扇出（fanout）维持在[4,16]区间
  - 扇出小于4则合并
  - 扇出大于16则分裂。

分形树Insert/Update/Delete

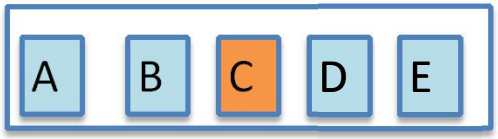
- 步骤：
  - a. 磁盘读取root节点页;
  - b. 若root节点需分裂，则root节点一分为二，提升一个新的Root节点;
  - c. 若root节点是叶子节点，则插入到basementNode；否则，append message到msg_buffer;
  - d. 返回

```
src/ydb_write.h: toku_db_put
    |- src/ydb_write.h: db_put
        |- ft/ft-ops.cc: toku_ft_insert_unique
            | - ft/ft-ops.cc: toku_ft_send_insert
                | - ft/ft-ops.cc: toku_ft_root_put_msg
```

每次插入，Put msg到根节点就返回!

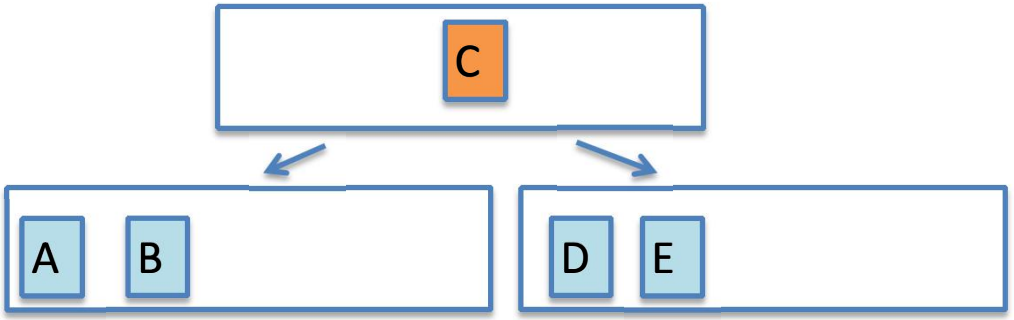
分形树Insert/Update/Delete

- **root节点分裂**

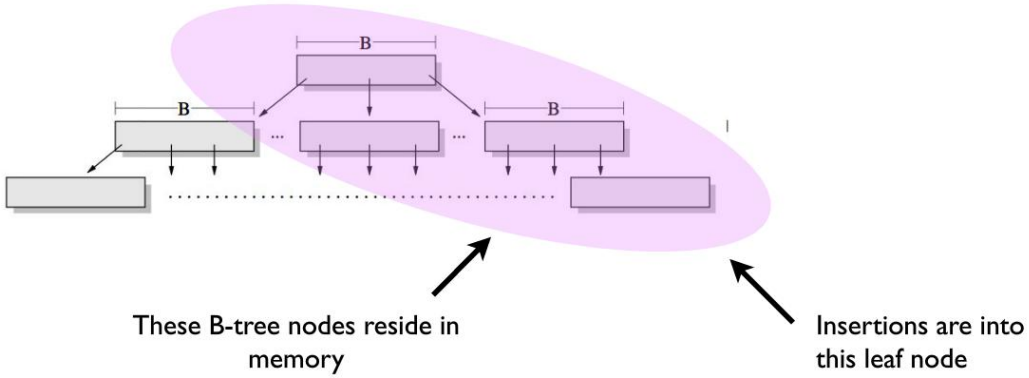


a. Root节点分裂前

b. Root节点分裂后

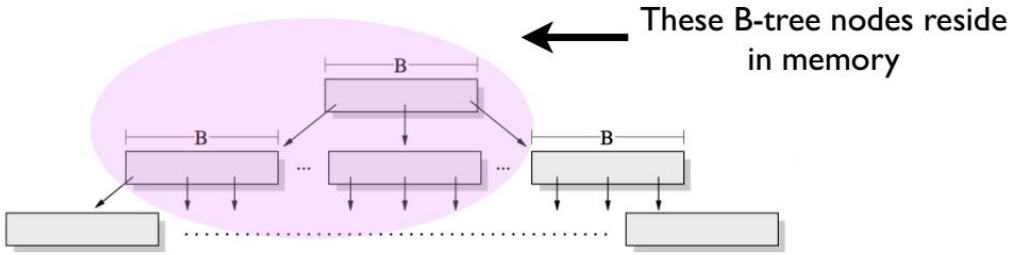


分形树Insert/Update/Delete



a. B+树顺序插入热点数据分布图

分形树对应的热点数据分布图？



b. B+树随机插入热点数据分布图

分形树Flush线程

- **Flush线程处理流程**
  - 当非叶子节点(height>0)的数据量超过4M时，表明该节点需要flush;
  - 找出该节点中所有msg_buffer数据量最大child，将(node,child)添加到flush线程中。
  - 将(node, child)节点中msg_buffer中所有的msg，都apply到child指向的儿子节点（页）。
  - 判断child指向的儿子节点(页)是否需要split或者merge。

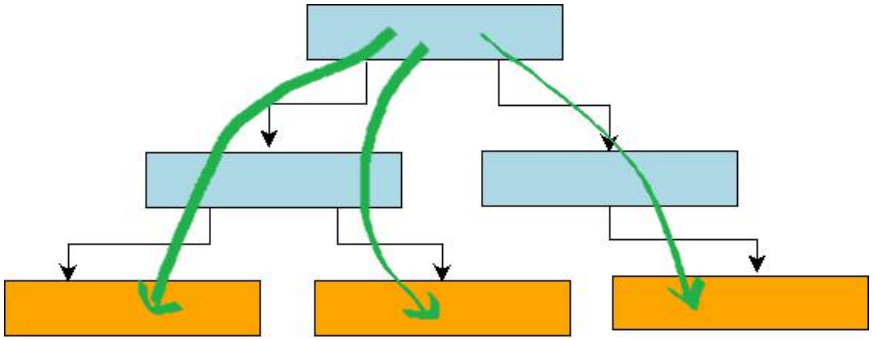
1. **异步flush msg_buffer消息到下层节点**
2. **维持树形结构和高度**
3. **牺牲写性能、提升读性能**

分形树Point-Query

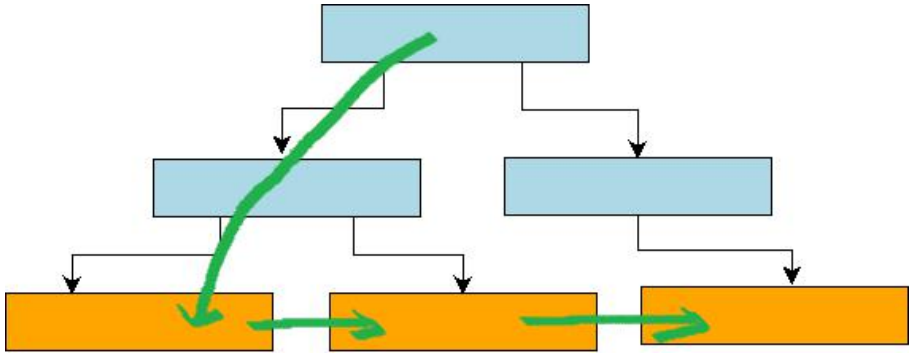
- **查找流程：**
  - 加载Root节点，通过二分搜索确定Key落在Root节点的键值区间Range, 找到对应的Range的Child指针。
  - 加载Child指针对应的的节点。 若该节点为非叶子节点，则继续沿着分形树一直往下查找，一直到叶子节点停止。若当前节点为叶子节点，则停止查找。
  - apply root节点到leaf节点上所有的消息到basementNode。
  - 返回basementNode上key对应的value。

```
src/ydb_cursor.h: toku_c_get
    | - src/ydb_cursor.cc: c_getf_first
        | - src/ydb_cursor.cc: toku_ft_cursor_first
            | - ft/cursor.cc: toku_ft_cursor_first
                | - ft/cursor.cc: ft_cursor_search
                    | - ft/ft-ops.cc: toku_ft_search
                        | - ft/ft-ops.cc: ft_search_node
                            | - ft/ft-ops.cc: ft_search_child
                            | - ft/ft-ops.cc: ft_search_basement_node
```

分形树Range-Query

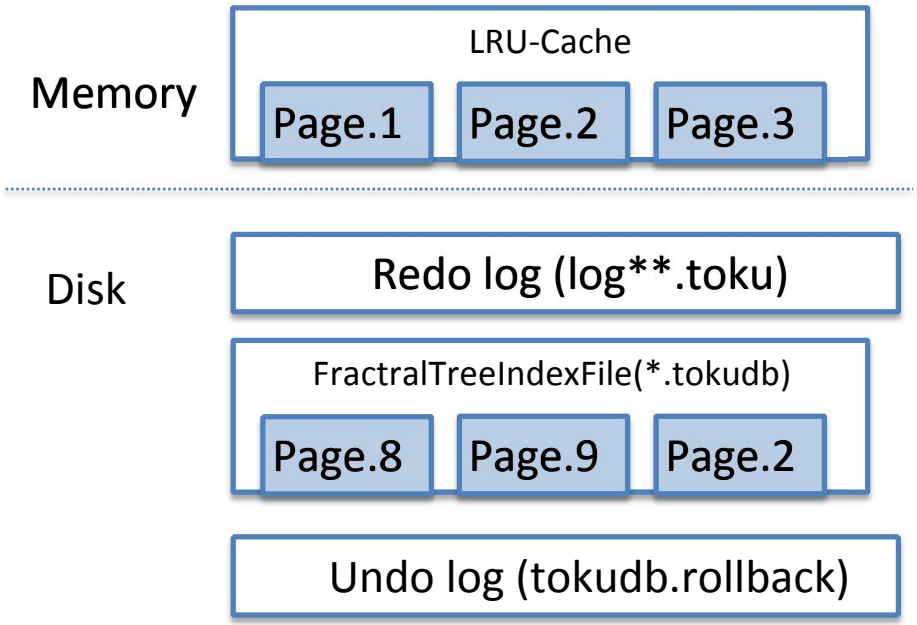


分形树范围查询



B+树范围查询

TokuDB事务（一）


Memory
LRU-Cache
Page.1
Page.2
Page.3
Disk
Redo log (log**.toku)
FractralTreeIndexFile(*.tokudb)
Page.8
Page.9
Page.2
Undo log (tokudb.rollback)

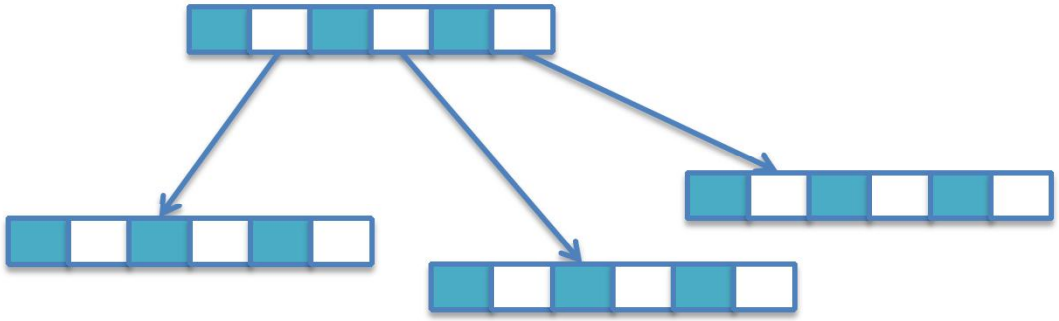
TokuDB事务（二）

- **事务举例：**
  - begin;
  - insert into student set id = 2, name='openinx';
  - commit;

TokuDB事务（三）

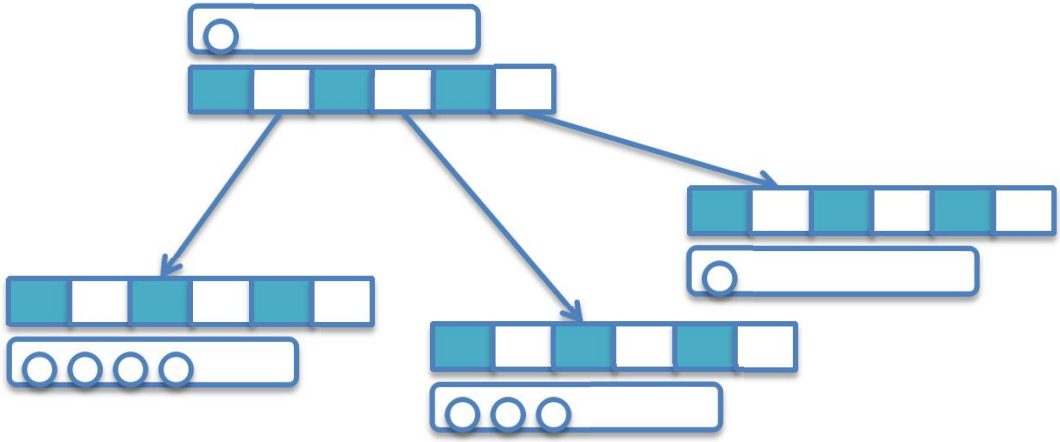
- **Begin操作：**
  - 初始化txn对象，分配xid。
  - 添加txn到事务管理器。
- **Insert操作：**
  - 插入{id=2,name='openinx'}到分形树中 =>添加消息到root节点
  - 写日志到rollback。=>写undo
  - 写xbegin操作到redolog（xbegin操作由begin之后的下一个更新操作写入到redo log）。
  - 写insert操作到redo log。
- **Commit操作：**
  - 写xcommit到redo log。
  - 写日志到rollback日志。=>写undo
  - 从事务管理器中删除txn。
  - fsync redolog到磁盘。

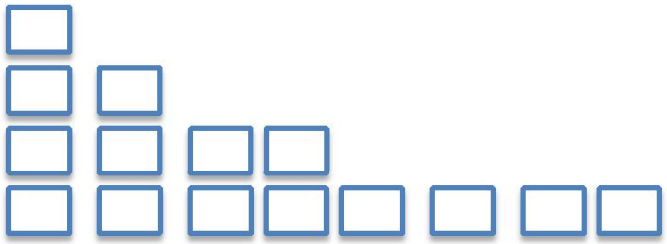
总结（一）



B+树

分形树





LSM树

总结（二）

	InnoDB(B+树)	TokuDB（分形树）	LevelDB（LSM树）
事务ACID	支持	支持	不支持
随机写	慢	快	非常快
随机读	慢	-	-
顺序写	最快	-	-
顺序读	最快	-	-
是否支持在线DDL	5.5之前不支持，5.6～5.7之后开始支持大部分的在线DDL.	支持在线DDL	不支持
undo-log	1.实现MVCC；2.实现事务回滚	仅用于实现事务回滚，MVCC由分形树结构支持	不支持事务，无undo-log
redo-log	物理日志	逻辑日志	逻辑日志
压缩比率	页16K，比TokuDB低	页4M，压缩比高	-
适用场景	读多写少的OLTP业务	写多读少的OLTP业务	写多读少的OLAP业务
页	16K，高扇出	4M，低扇出	2M，无扇出
代表产品	InnoDB/MyISAM/SQLite/XFS/MongoDB etc.	TokuDB/TokuMX/TokuFS	BigTable/HBase/Cassandra/MongDB-Wired Triger/InfluxDB

参考资料

- http://github.com/Percona/tokudb-engine
- http://openinx.github.io/2015/11/25/ft-index-implement/
- http://pan.baidu.com/s/1bnFkCEV

Thank you