SACC

卓越 5周年 变迁

2013中国系统架构师大会

SYSTEM ARCHITECT CONFERENCE CHINA 2013

大数据下的IT架构变迁

高性能存储引擎TokuDB剖析

关于我

- 一工@淘宝核心系统
- 开源爱好者、存储引擎探索和实践者
- 博客：logN.me
- 微博：@BohuTANG_
- Gtalk: overred.shuttler@gmail.com

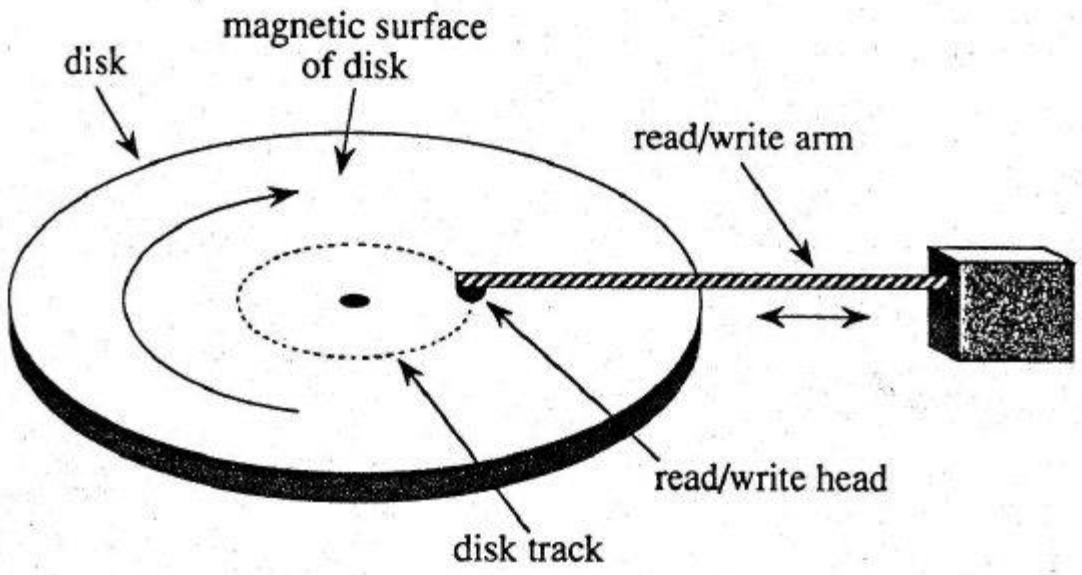
大纲

- TokuDB简介
- TokuDB的索引结构
- TokuDB的特点
- 不适用场景

TokuDB简介

- Tokutek 公司研发(2007--2013年open source)
- 目前5个研发，1个性能测试，1个科学家(MIT)
- 支持MySQL/MariaDB，GPL开源协议

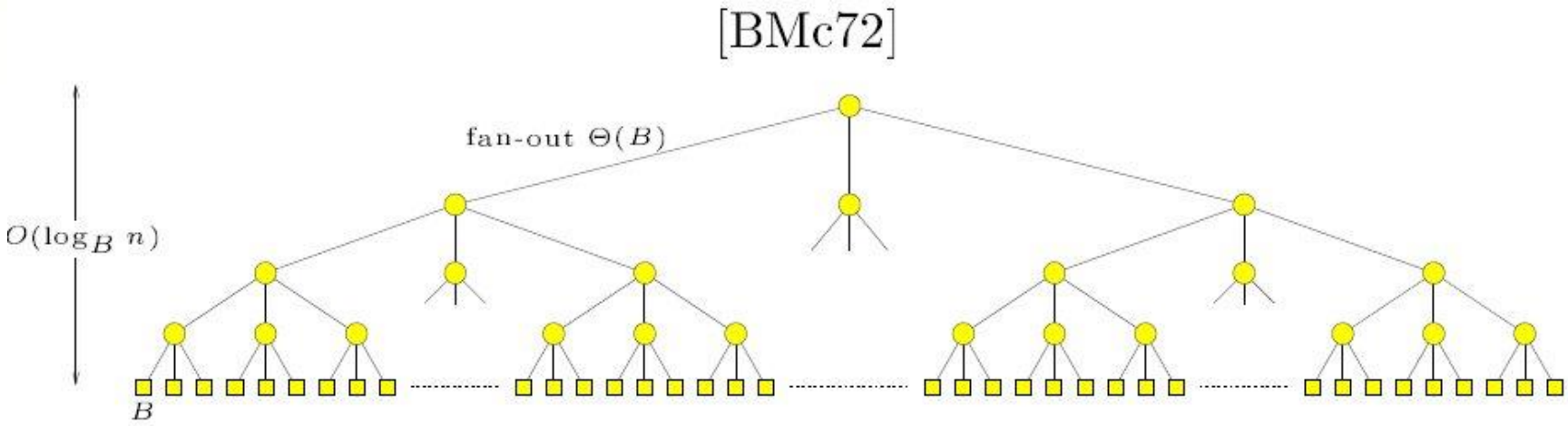
TokuDB的索引结构



随机读/写时间 = 寻道时间 + 读/写时间

TokuDB的索引结构

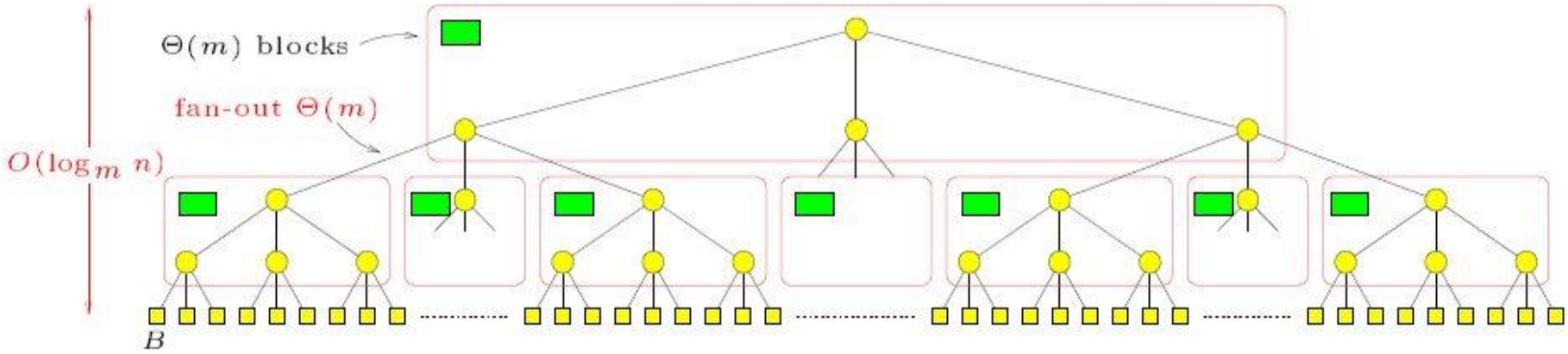
B-tree索引的缺陷



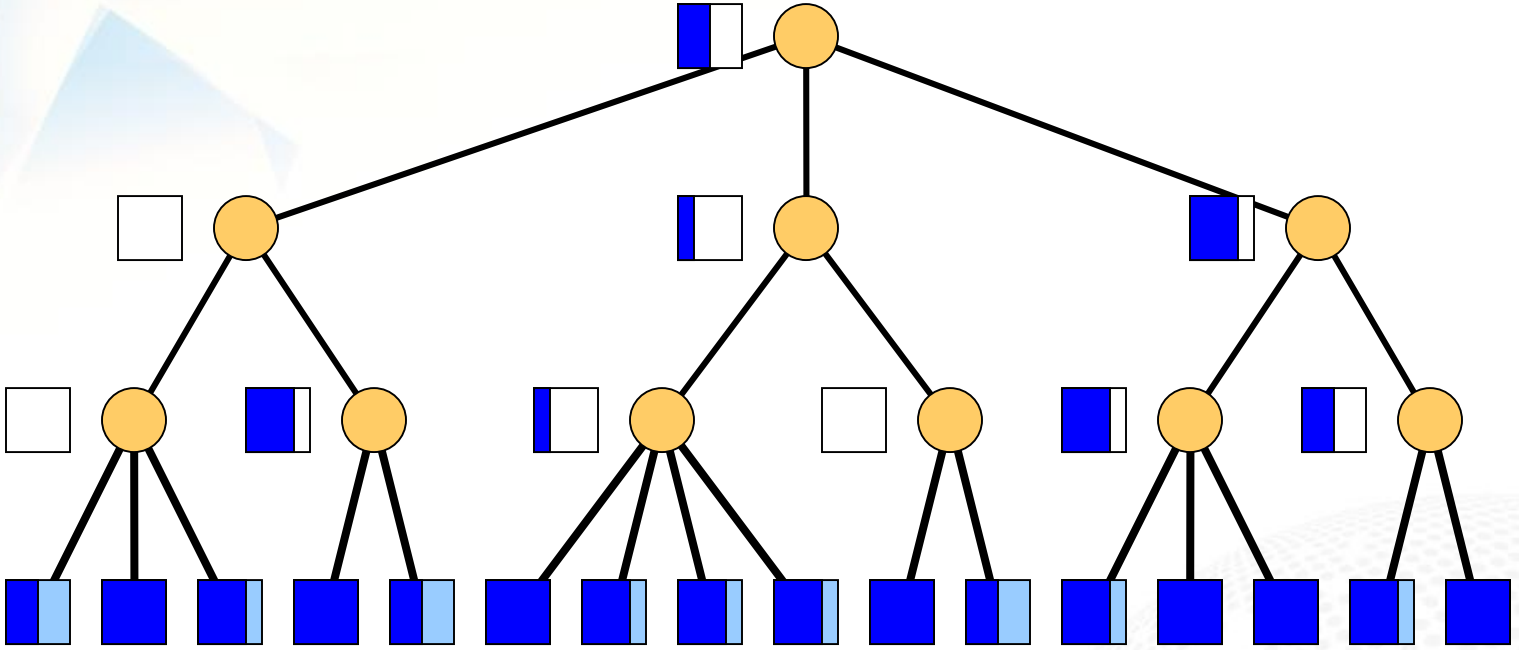
假设：B = 16KB
50GB / 16KB ~ 300w个node，太多了！
缺点：不适合随机读写，大部分是寻道时间！

TokuDB的索引结构

Fractal-Tree® 索引

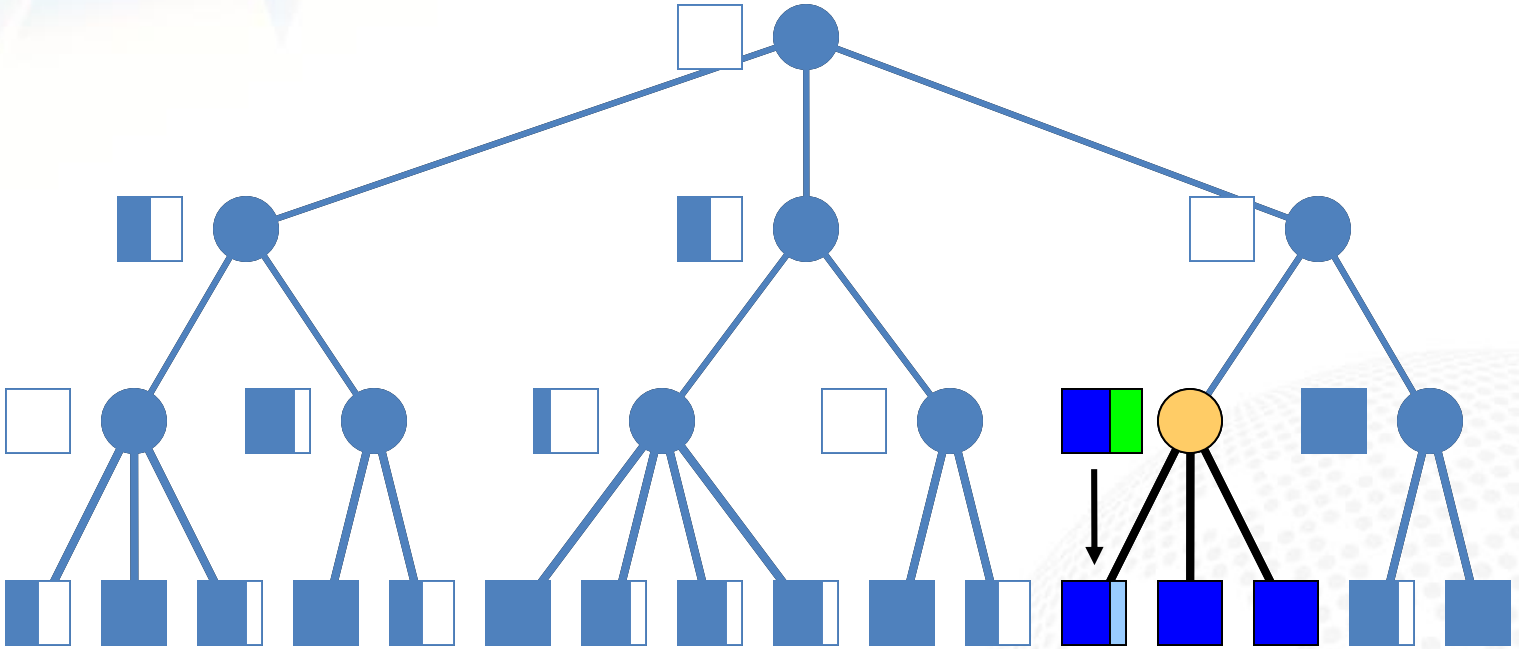


B = 4MB（块大，整块压缩，~1MB)
50GB / 4MB ~ 1w个node，node少！



- 每个node有4-16个子节点
- 每个node都有一个message buffer
- 内节点message buffer是FIFO结构，减少cmp
- 叶节点message buffer是OMT结构，弱平衡的二叉树

INSERT



TokuDB的索引结构

- Fractal-Tree® ，由TokuTek注册专利
- 它其实是Buffer-Tree的一个“变种”
- Buffer-Tree是由Lars最早提出
- TokuDB对细节做了不少优化:CPU+Memory

Lars Arge

Curriculum Vitæ
January 2013

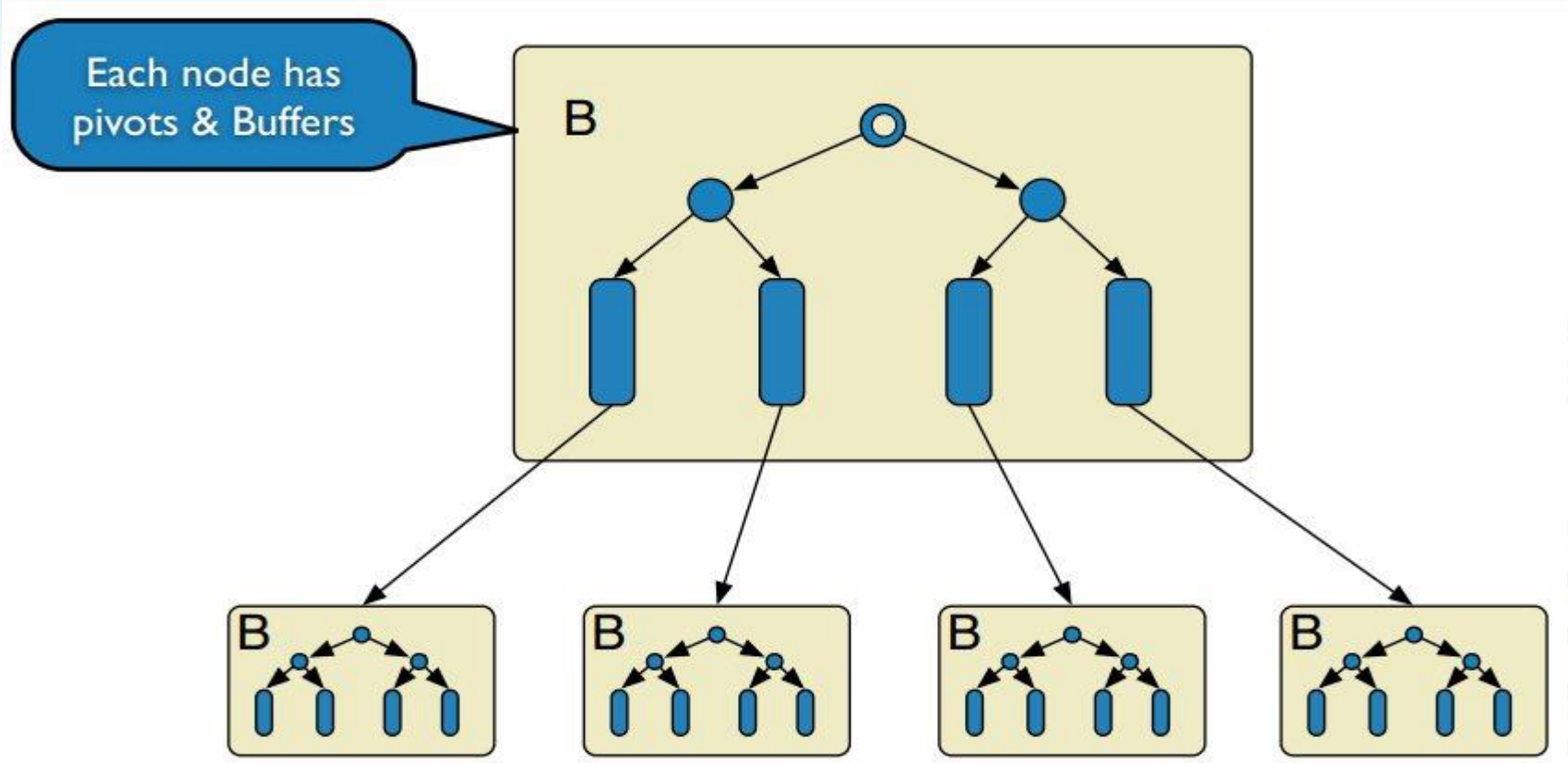
TokuDB的特点

一切的好处均要归功于message buffer

TokuDB的特点

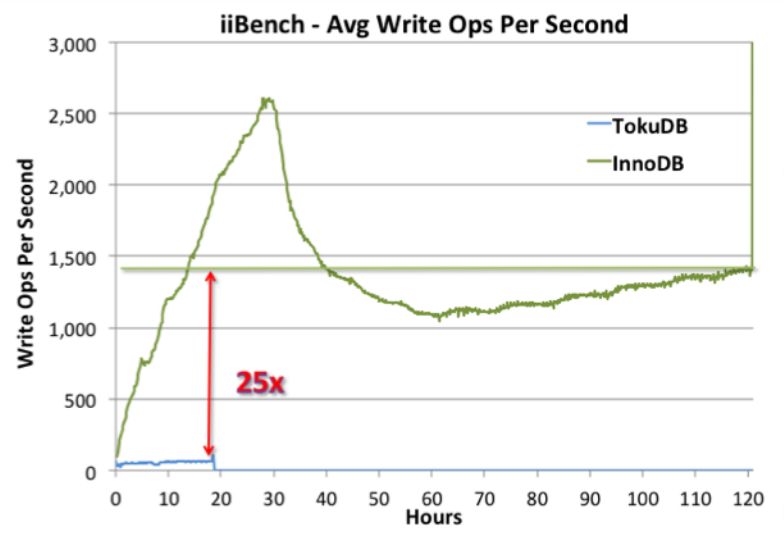
- Insert/Update/Delete/Column等，均是message
- lazy式操作，自上而下逐步Flush到leaf节点
- 天然多版本，无需做undo log
- Fast insert/ Fast update，延迟小

TokuDB的特点



TokuDB的特点

- node块大，适合压缩
- IOPS少，适用于SSD



TokuDB的特点

Log

- log manager来管理log文件，InnoDB的rotate log效果
- 为了并发性能，log在实现上，分: in buffer和out buffer
- 支持group commit
- 只有redo log ，不压缩，顺序写文件

TokuDB的特点

Transaction

- ACID
- Fractal-tree在事务实现上有优势，无undo log
- Leaf节点做MVCC，实现最终多版本
- 内存维护活动事务(live transaction)链
- tokudb.rollback记事务逻辑日志

TokuDB的特点

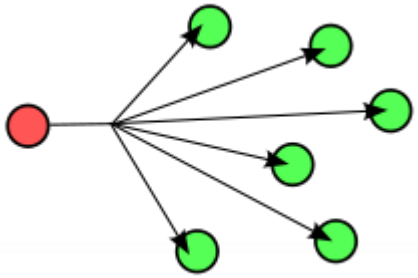
CheckPoint

- 类型是sharp checkpoint
- 后台线程定期执行
- Begin checkpoint:
  - 1) 给所有node打pending标记
  - 2) 获取当前checkpoint的LSN
- End checkpoint:
  - 3) 把dirty node写盘
  - 4) 把ft meta信息写盘
- 中途Crash问题

TokuDB的特点

Hot Schema

- Add/Delete Column， message类型为BROADCAST
- Lazy式
- 每个node遇到BROADCAST，都要把它带给子节点



TokuDB的特点

Indexing

- 索引分两种：
  a) clustering index
  b) covering index
- 每个索引均是一个单独的fractal-tree结构文件
- 表foo:

```
mysql> create table foo(a int primary key, b int, c int, d int);
```

TokuDB的特点

Indexing

covering index:

```
mysql> alter table foo add index cvr_bc(b, c);
```

此时索引文件里一条记录的结构：

```
{key={len=xx data="b-c-a"} val={len=0 data=""}}
```

索引key有covering key组成，value空

TokuDB的特点

Indexing

clustering index:

```
mysql> alter table foo add clustering index cst_bc(b,c);
```

此时索引文件里一条记录的结构：

```
{key={len=xx data="b-c-a"} val={len=xx data="d"}}
```

TokuDB的特点

Indexing

Covering index

```
{key={len=xx data="b-c-a"} val={len=0 data=""}}
```

vs

Clustering index

```
{key={len=xx data="b-c-a"} val={len=xx data="d"}}
```

- 都是物理的
- Clustering index包含全数据
- Clustering index可是表的另一种排序方式

不适用场景

- select count操作，需做表扫描
- affected rows操作，需做表扫描

Thanks!


SequeMedia
盛拓传媒
IT168.com
www.it168.com
ChinaUnix.net
ITPUB