移动应用在数据库上所需的适应与改变

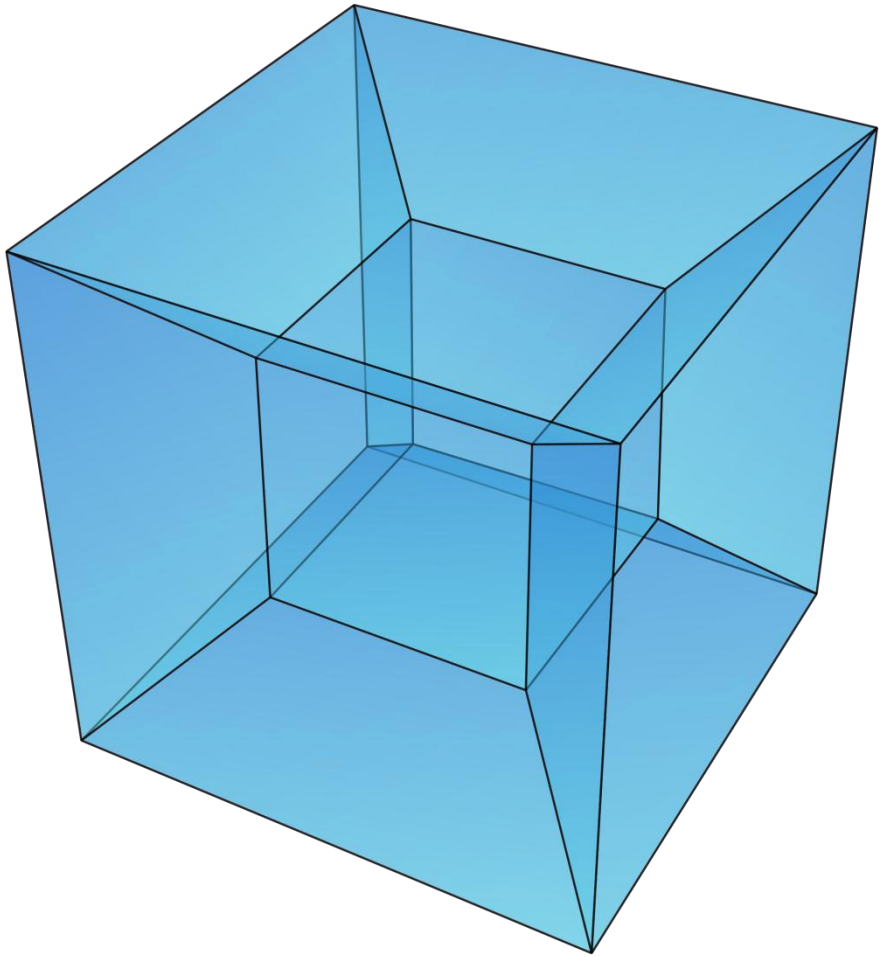
jingmi@gmail.com

2012/03/20

Agenda

- LBS 的应用特点与面临的问题
- 当 RDBMS 与 NOSQL 遇到 LBS
- 街旁的解决之道：CrabDB

多一维的视角



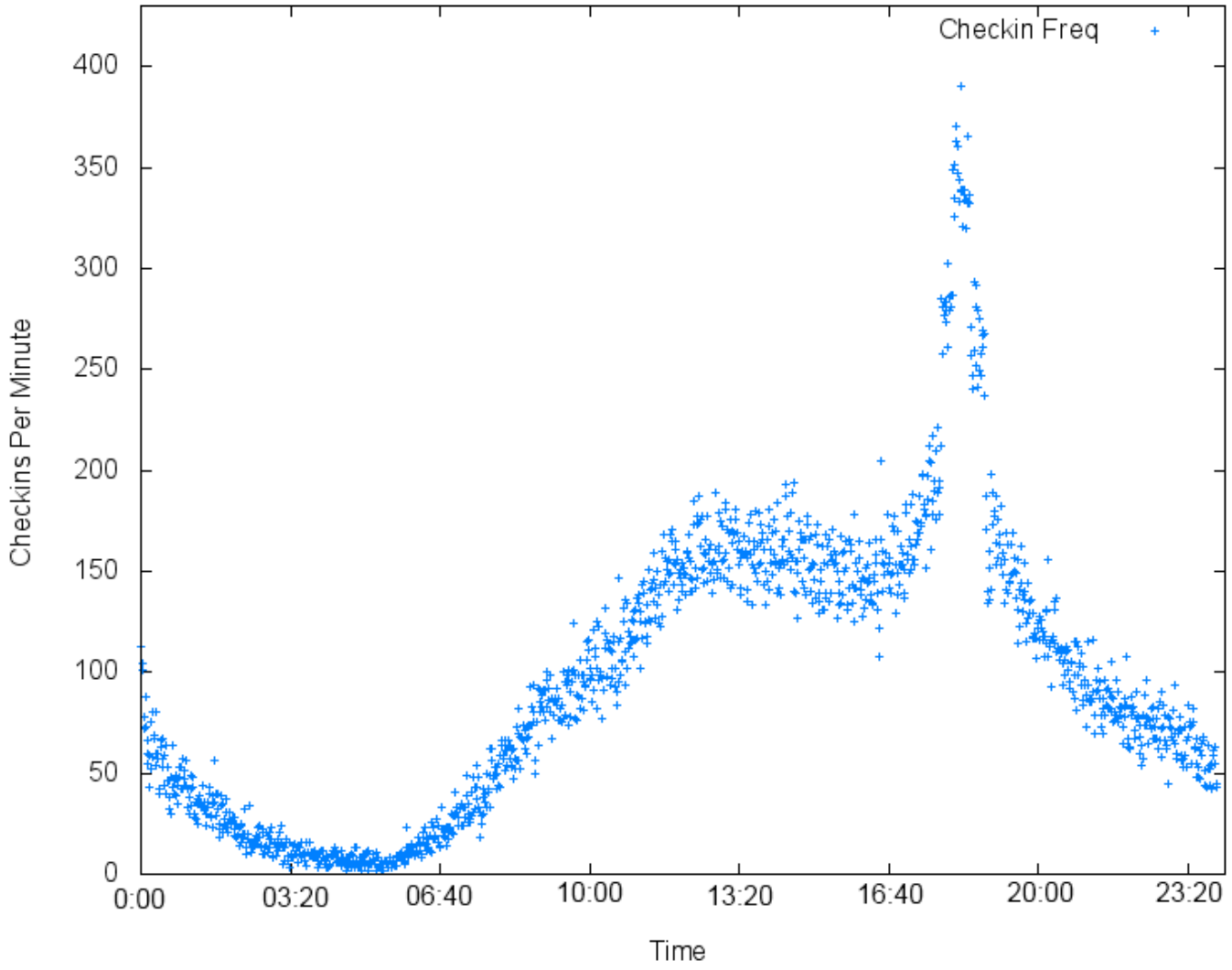
实时的签到计算

- 签到
- 计算地主
- 计算惊喜
- 弹出攻略
- 去过的好友
- 计算积分
- 计算徽章

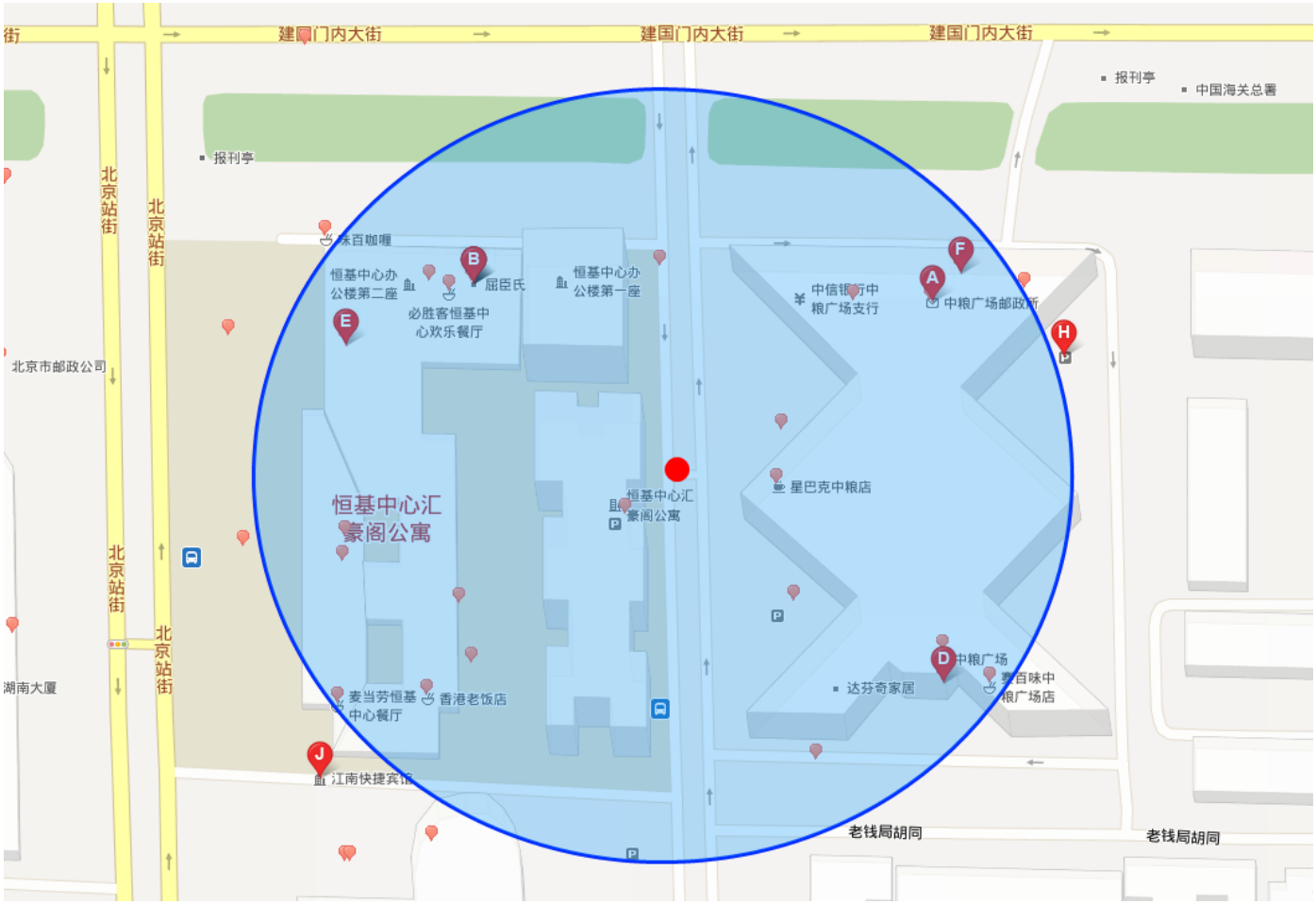
实时的签到计算(Cont.)

实时的签到计算(Cont.)

迅猛的压力高峰


Checkin Freq
Checkins Per Minute
400
350
300
250
200
150
100
50
0
0:00
03:20
06:40
10:00
13:20
16:40
20:00
23:20
Time

与众不同的广告类型


建国门内大街
建国门内大街
建国门内大街
报刊亭
中国海关总署
报刊亭
北京站街
北京站街
北京站街
北京站街
北京市邮政公司
湖南大厦
恒基中心办公楼第二座
屈臣氏
必胜客恒基中心欢乐餐厅
恒基中心办公楼第一座
中信银行中粮广场支行
中粮广场邮政所
恒基中心汇豪阁公寓
恒基中心汇豪阁公寓
星巴克中粮店
中粮广场
达芬奇家居
麦当劳恒基中心餐厅
香港老饭店
江南快捷宾馆
老钱局胡同
老钱局胡同

快速的需求变化

- LBS 的快速发展
- 未来产品的发展
- 性能的需求
- 低成本的硬件平台

Agenda

- LBS 的应用特点与面临的问题
- 当 RDBMS 与 NOSQL 遇到 LBS
- 街旁的解决之道：CrabDB

RDBMS 如何处理？

- 一张简单的表 loc_post （示例）

id	loc_id	user_id	private	……
0	1	3	0	
1	1	3	1	
2	1	3	2	

- 普通用户查询

SELECT Count(DISTINCT user_id) FROM loc_post WHERE loc_id = 1 and private = 0;	1
SELECT Count(user_id) FROM loc_post WHERE loc_id = 1 and private = 0;	1

- 朋友查询

SELECT Count(DISTINCT user_id) FROM loc_post WHERE loc_id = 1 and private <= 1;	1
SELECT Count(user_id) FROM loc_post WHERE loc_id = 1 and private <= 1;	2
SELECT Count(DISTINCT user_id) FROM loc_post WHERE loc_id = 1 and private <= 1 and user_id in (1, 2, 3);	1

- 自己查询

RDBMS 如何处理？(Cont.)

- 查询/计算类型很多

  SELECT … WHERE …

  SELECT … IN …

  SELECT … ORDER BY …

  SELECT Count(…) … GROUP BY … ORDER BY …

  UPDATE …

  INSERT …

- 索引过多
- 数据一致性要求高，不能批量/推迟计算，IO 过多，缓存失效很快

RDBMS 如何处理？(Cont.)

- 的确可以做很多优化（分区、设计更合理的 schema、数据库调优、etc）
- 应用开发工程师需要注意的优化情况太多
- 不利于产品的快速开发和迭代

NOSQL @ Foursquare(2011.06)

- > 9M users
- ~3M checkins/day(52 checkin/sec)
- > 20M places
- MongoDB, PostgreSQL (legacy, migrating off)
- all on EC2
- ~40 machines (68GB, m2.4xl on EC2)

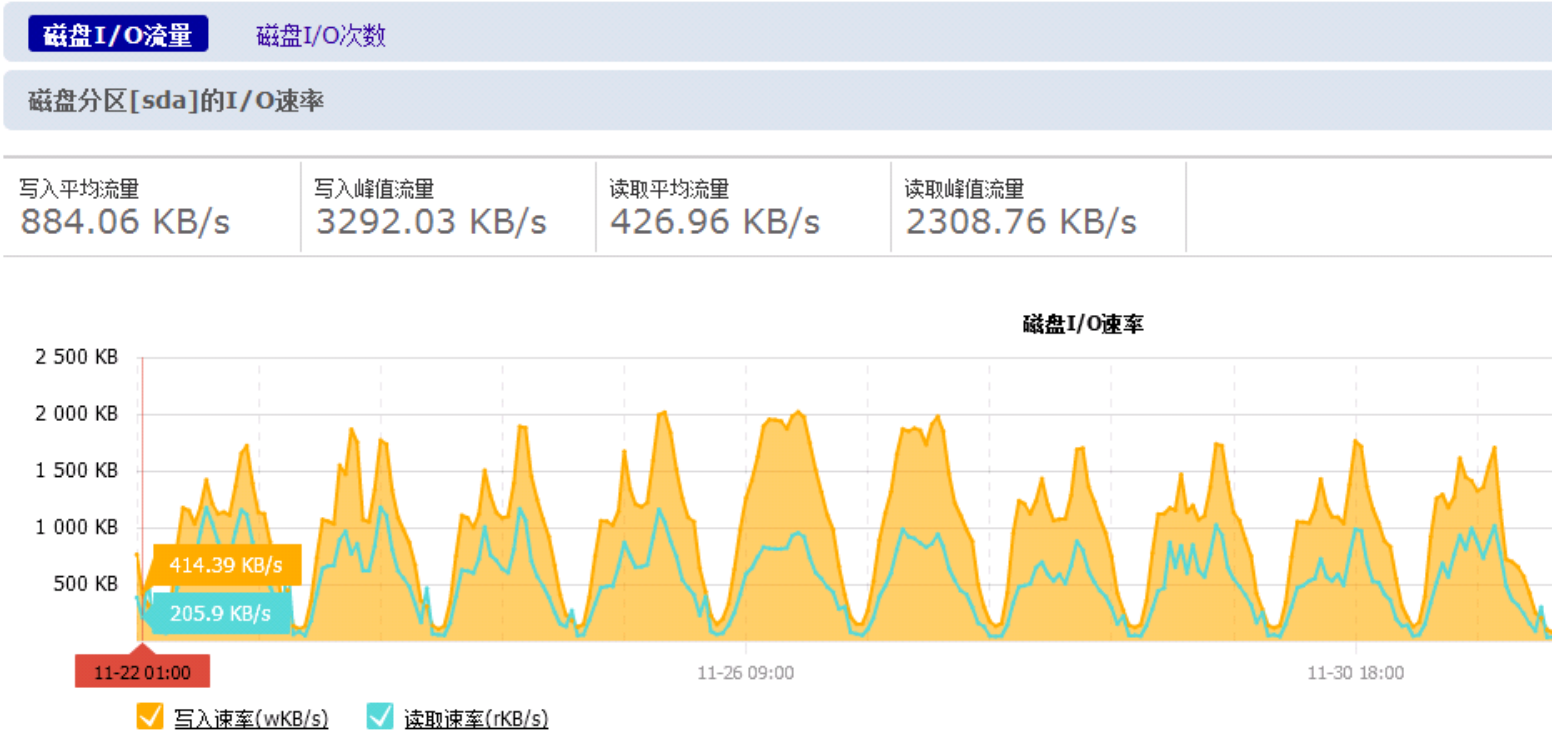
http://engineering.foursquare.com/2011/06/17/presentation-on-how-foursquare-uses-mongodb/

选择MongoDB

- Schema-Free
- 减少 join
- Replica-sets 与容灾
- GEO-Indices
- 高性能

为什么迁移出 MongoDB

- 数据总量大（超出内存），IO 压力比较高
- CPU load avg: > 4（16 核 64G 内存）

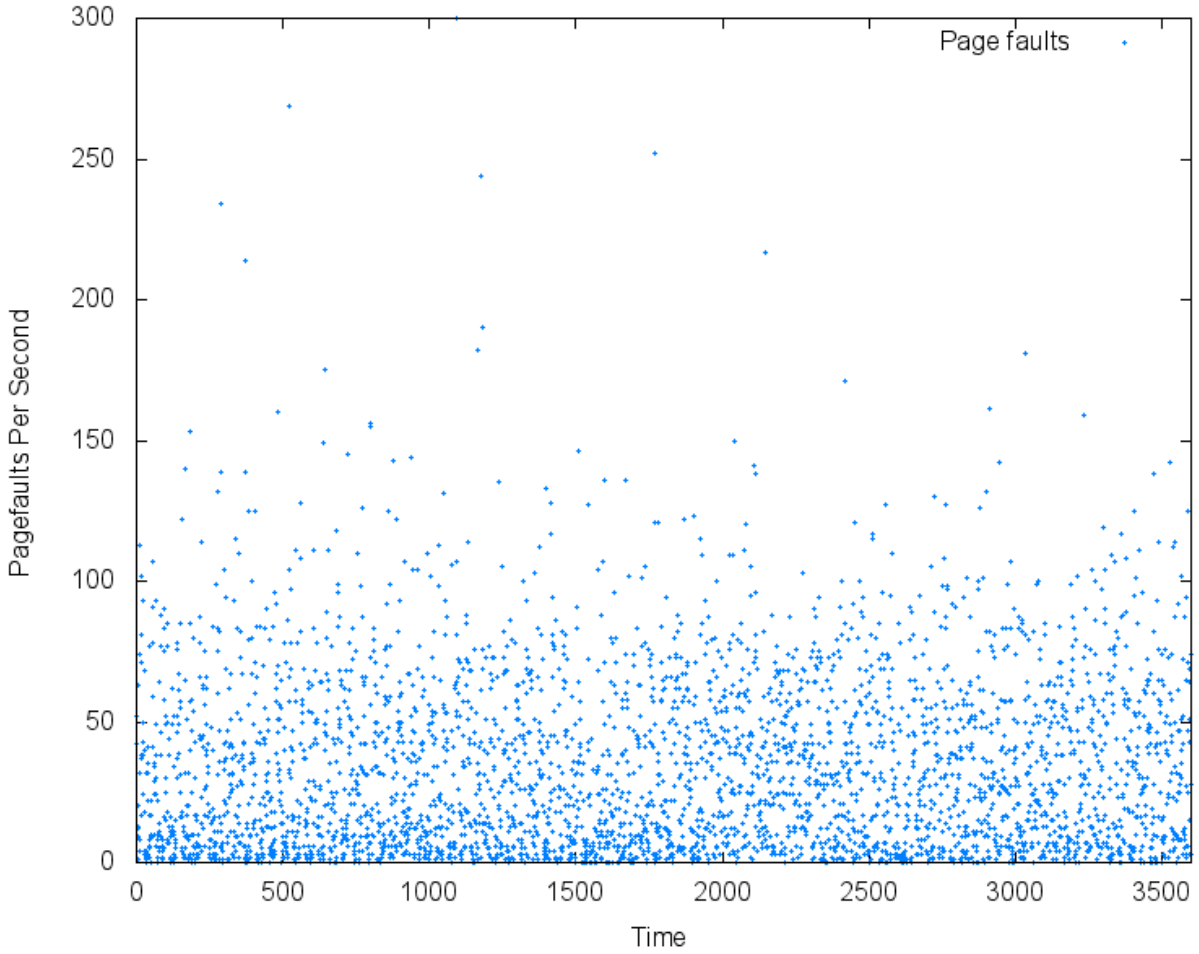


为什么迁移出 MongoDB (Cont.)

Pagefaults 频繁

Median = 26 pagefaults/sec

Avg = 33.86 pagefaults/sec



MongoDB 不适用分析(1)：数据膨胀

- BSON 是把双刃剑
  - Schema Free
  - 空间开销巨大
- 索引过多
- 数组索引

MongoDB 不适用分析(2):内存有限

- 不同版本 Linux 内核实现 mmap(2) 的 writeback 机制差异
- 缓存粒度难以有效控制（OS负责页面换入换出）
- LBS 中对大量长尾数据的访问造成随机读写
- vmtouch 告诉我们只有 35.5% 的数据被缓存

MongoDB 不适用分析(3):数据分布

- 数据分布不可控，按照非 _id 连续访问记录并不具备优势
  - LBS 计算中存在大量选择所有属于同一个user_id 的记录并计算的需求
  - RDBMS 中可以使用 Clustered Index

Agenda

- LBS 的应用特点与面临的问题
- 当 RDBMS 与 NOSQL 遇到 LBS
- 街旁的解决之道：CrabDB

CrabDB 的数据结构

- 简单的 key -> list

```
loc_id:[ { user_id  :  {id,  private,  …} },
         { user_id  :  {id,  private,  …} },
         { user_id  :  {id,  private,  …} },
         ……]
```

- 只保存数据
- 定长存储

CrabDB的查询与计算

- 多种查询（类似于 MongoDB 的调用）

  db.find(Expr)

  db.group(Expr)

  db.sort(Expr)

  db.limit(Expr)

  db.count()

- 条件表达式支持

  tcc 动态编译表达式请求（比 MongoDB 中 js 实现表达式计算更快）

  更加方便快速的实现复杂的表达式

  group(datetime / 86400): 将存储的 datetime 类型转换为天进行聚合

  find(R.id * 2 == 6)

- 快速的查询与计算

一些用例及对比

- 对比

CrabDB	MongoDB
crab.user_post[3].find(R.type == 3)	db.post.find({'u': 3, 'type': 3})

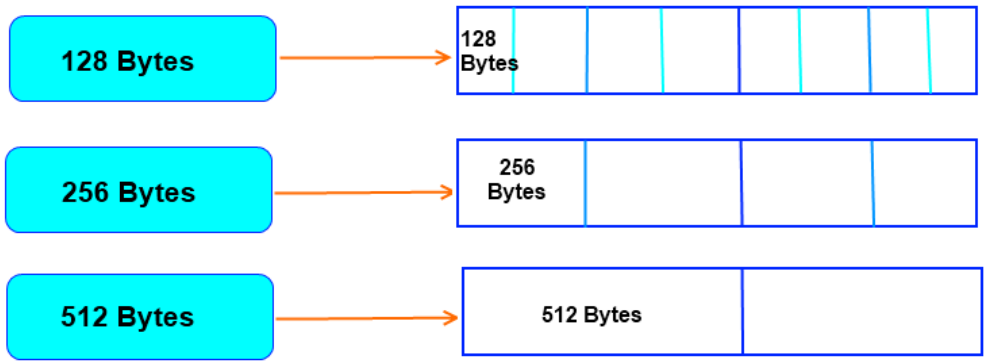
- 更多查询（Python 调用）

```
crab.location_post[1].find((R.privacy == 0) |
  R.user_id.in([1,2,3,4,5])).count()


crab.location_post[1].find(R.type == 2,
  R.user_id.in([123, 456, 678])).sort([R.is_great,
  -R.num_likes, -R.post_id])
```

CrabDB 的存储分配

- Slab 分配策略
- 按照比例扩容
- 只保存数值，内存利用率高，便于压缩
- 放在内存中，减少IO



CrabDB 的分布式实现

- 分布式对象系统
- Master / Slave
- Slab 块在后台异步扩容

CrabDB 的分布式实现(Cont.)

- 扩展性

  扩展计算

  扩展存储

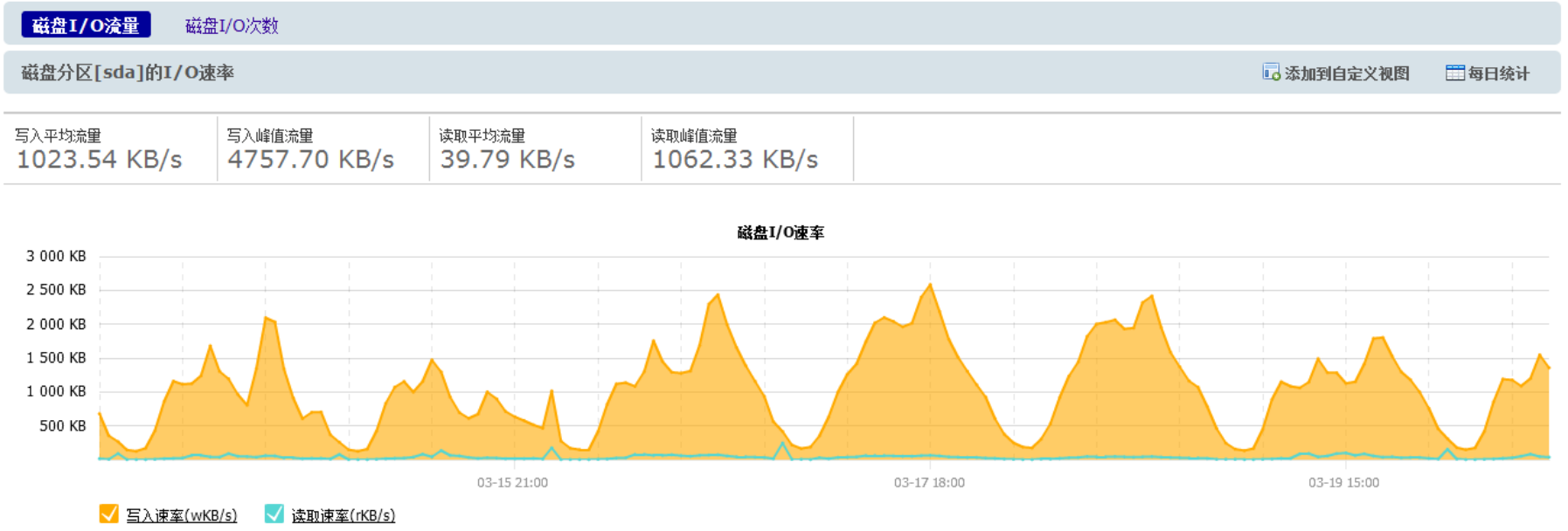
- 容灾与恢复

  恢复计算

  恢复存储

CrabDB 上线后的成果

- 系统响应更快
- MongoDB 压力大幅降低，线上机器资源大幅节省
- 写入未减少是因为数据被备份至 MongoDB

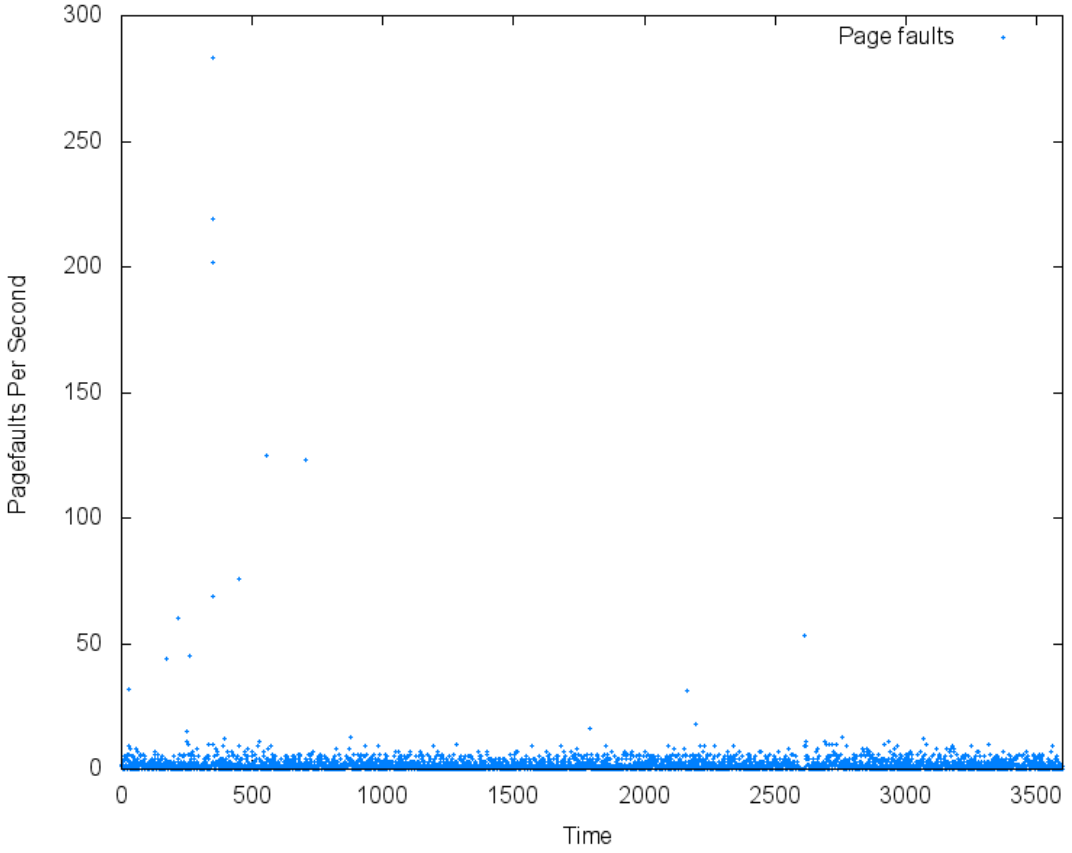


CrabDB 上线后的成果（Cont.）

Pagefaults 大幅减少

Median = 1 pagefaults/sec

Avg = 2.33 pagefaults/sec



让 MongoDB 做最擅长的事情

- 足够好的性能
- 更快速的开发支持
- Schema Free
- 地理位置索引

THANKS