



2017第八届中国数据库技术大会

DATABASE TECHNOLOGY CONFERENCE CHINA 2017

搜狗检索系统的性能优化

 搜狗搜索 谢远江

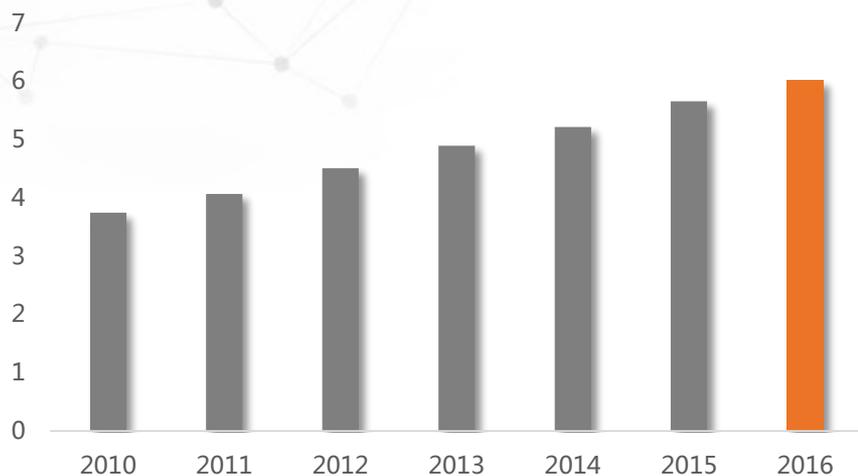
搜狗搜索 成立于2004年

拥有**5.6亿用户**，是中国**第二大搜索引擎**



搜索系统面临的挑战

PC搜索用户规模数（亿）

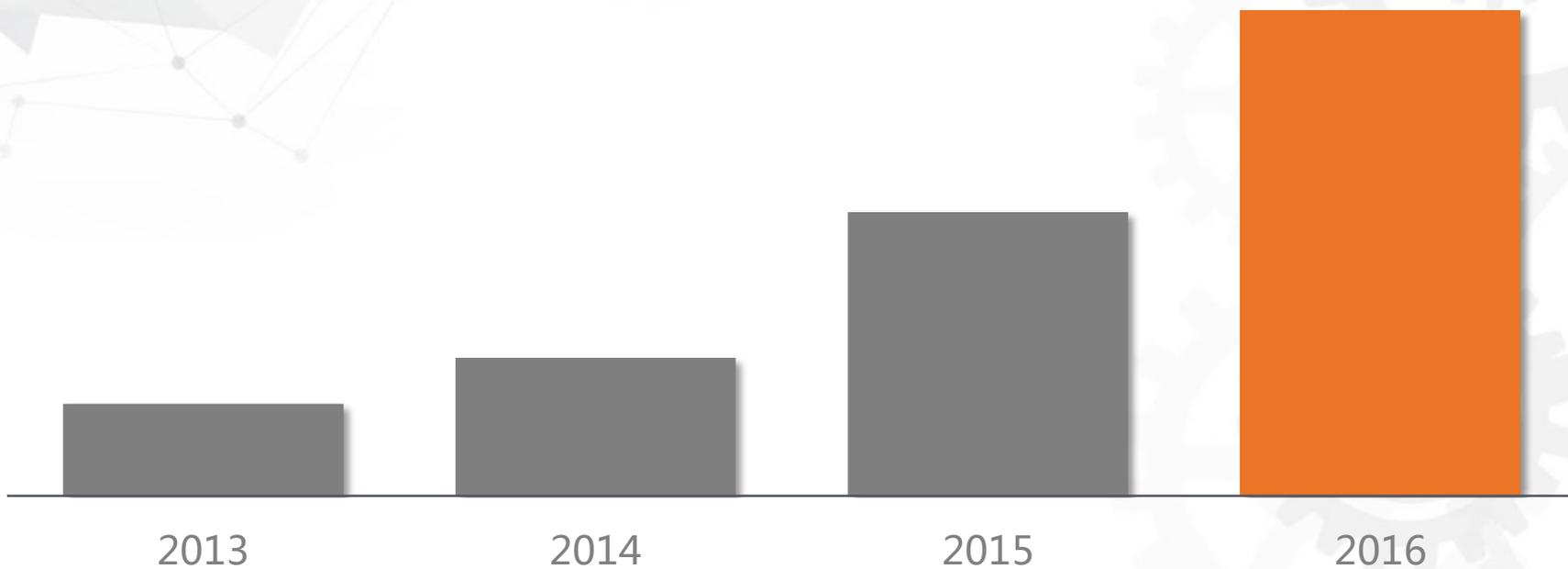


手机搜索用户规模数（亿）



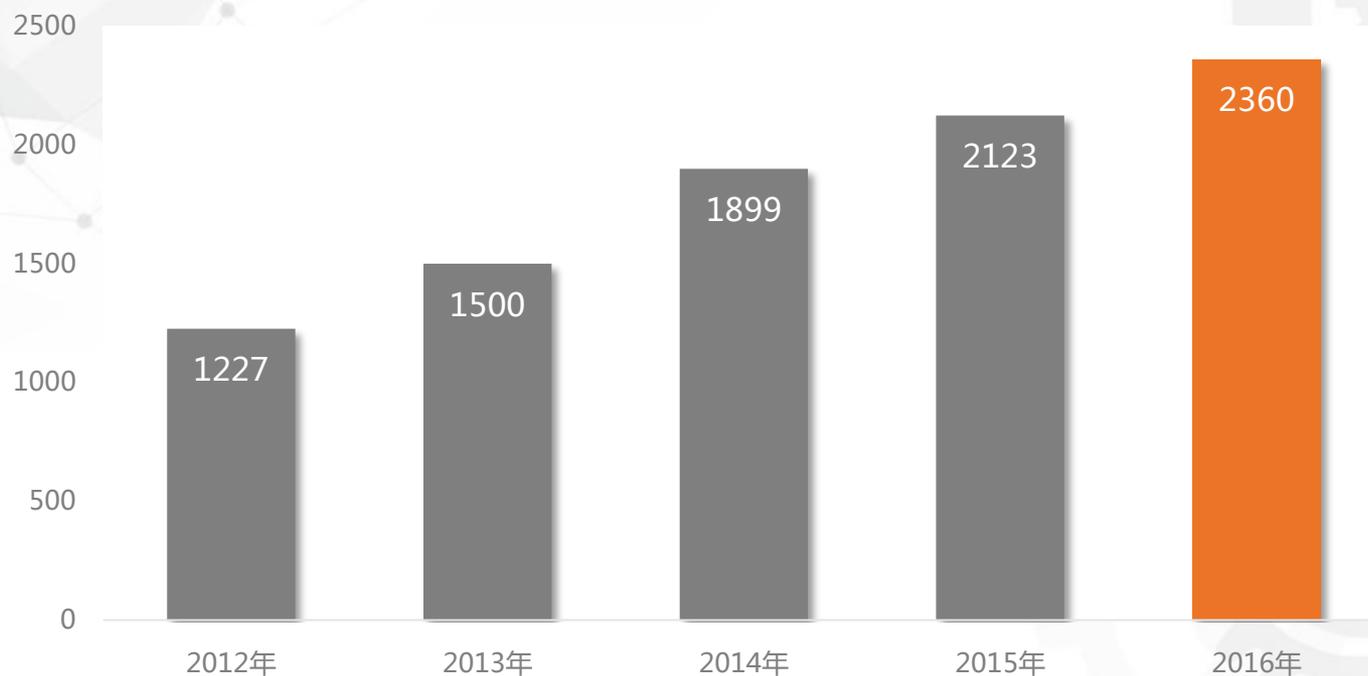
搜索系统面临的挑战

三年内移动搜索量增长800%



搜索系统面临的挑战

网页数量 (单位: 亿个)



搜索系统面临的挑战



搜索需求

新闻实事

影视娱乐

出行信息

信息释义

工具、助手

搜索系统面临的挑战



搜索系统面临的挑战



性能优化



性能优化

代码实现

➤ 锁

- 临界区大小

➤ 循环展开

```
for(int i=0; i<10000000; i++)  
    r += i;
```



```
for(int i=0; i<10000000; i+=4)  
{  
    r += i;  
    r += i+1;  
    r += i+2;  
    r += i+3;  
}
```

➤ If(A && B && C)

- A==true 70%
- B==true 20%
- C==true 5%

➤ 数据预处理

- 减少重复计算

性能优化

数据结构

➤ 选用合适的数据结构

Map



Unordered_map

➤ 根据业务场景设计数据结构

- Heap
 - pop()
 - push()
 - top()
- 1亿个整数求TOP 1W
 - 元素更新：pop + push
 - Heap::top_replace()

性能优化

体系结构

硬件升级

- cpu
- 内存
- 硬盘
- 网卡
-

专用芯片

- GPU
- TPU
- 压缩芯片
- 加密芯片
-

体系结构

- 系统环境
- 超线程
- NUMA
- SIMD&&AVX
-

性能优化

体系结构

➤ Malloc

- ptmalloc
- TcMalloc & JeMalloc
 - 减少锁争用
 - 线程缓存

➤ 文件系统缓存

- dirty_ratio
- dirty_expire_centiseecs

➤ Cpu运行频率

- 降频
- 定频

➤ 超线程

- 多线程应用
- 文件、网络、线程调度等

➤ Cache line

- Size : 64Byte
- Alignment : 64

```
struct People{  
    int user_id;  
    int name_len;  
    char name[64];  
    int age;  
    int height;  
};
```



```
struct People{  
    int user_id;  
    int age;  
    int height;  
    int name_len;  
    char name[64];  
};
```

```
struct foo{
    int x;
    int y;
};

struct foo f;

void *thread_A(void *)
{
    for(int i=0; i<10000000; i++)
        f.x += i;
}

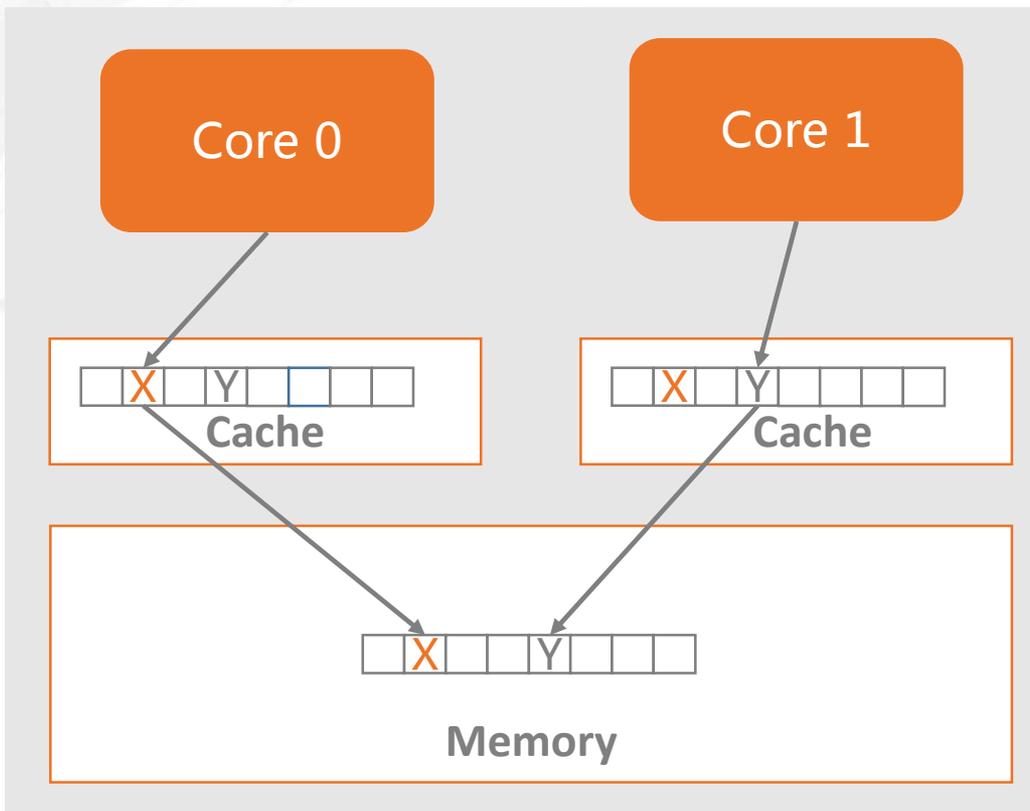
void *thread_B(void *)
{
    for(int i=0; i<10000000; i++)
        f.y += i<<1;
}
```

线程A, Cost:28ms

线程B, Cost:30ms

线程A & B, Cost:70ms

Cache False Sharing



➤ Cache False Sharing

- 增大元素间隔
 - 不同线程存取的元素位于不同的cache line上
- 线程局部变量
 - 创建全局数据的本地拷贝
 - 回写全局数据
- 数据结构分拆
 - 单独存储无交互或耦合的元素

```
struct foo{
    int x;
    char buf[60];
    int y;
};

struct foo f;

void *thread_A(void *)
{
    for(int i=0; i<10000000; i++)
        f.x += i;
}

void *thread_B(void *)
{
    for(int i=0; i<10000000; i++)
        f.y += i<<1;
}
```

线程A, Cost:28ms

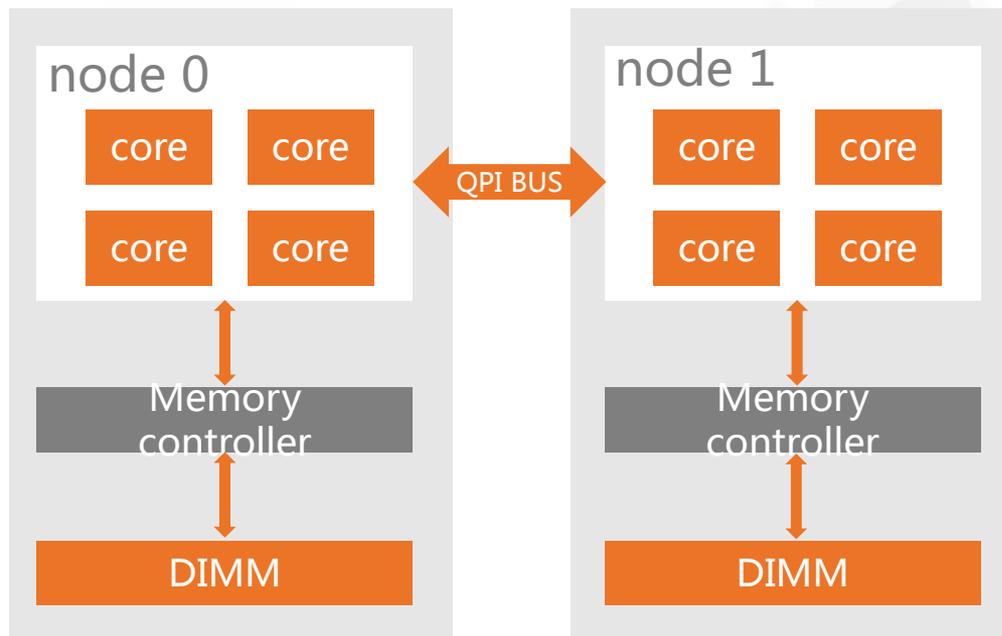
线程B, Cost:30ms

线程A & B, Cost:30ms

➤ NUMA

- 访问本地存储器的性能远高于非本地存储器

➤ 15%性能提升



性能优化

算法设计

➤ 场景

- 数据查找
- 数据格式 <uint64, uint64>
- 数据规模：百万
- 无数据更新
- Qps：10w~100w
- 命中概率：<5%

➤ Hash

➤ Hash + BitMap(BloomFilter)

0	0	1	0	0	1	0	0	0	0	1	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---

性能优化

算法设计

➤ 检索性能

- 载入内存的倒排数据
- 倒排的查找效率

➤ 倒排

- 单调递增整数数列
- {2,3,5,9,10,20,22,23,...}

➤ 倒排压缩

- 解压缩速度
- 压缩比

➤ 压缩算法

- Variable byte coding
- Elias Gamma (Delta) coding
- Golomb coding
- Simple-9
- Simple-16
- PForDelta
- NewPForDelta

性能优化

算法设计



高频

- 数量 : 0.01%
- 使用率 : 12%

中频

- 数量 : 0.1%
- 使用率 : 75%

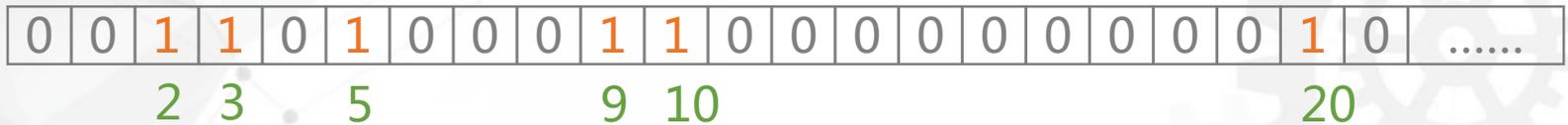
低频

- 数量 : 99.89%
- 使用率 : 13%

性能优化

算法设计

➤ Bitmap



➤ 定义Find函数

- Bitmap_find(x, res)
 - 如果bit x = 1 , res = x
 - 如果bit x = 0 , res = x往后看第一个出现1的位置
- 查找指令
 - Bsfq

➤ 优点

- 查找效率非常高

➤ 缺点

- 空间占用大

性能优化

算法设计

➤ BitCompres

➤ Delta {2,3,5,9,10,20,22,23,...} ➤ {2,1,2,4,1,10,2,1,...}

➤ Group bit

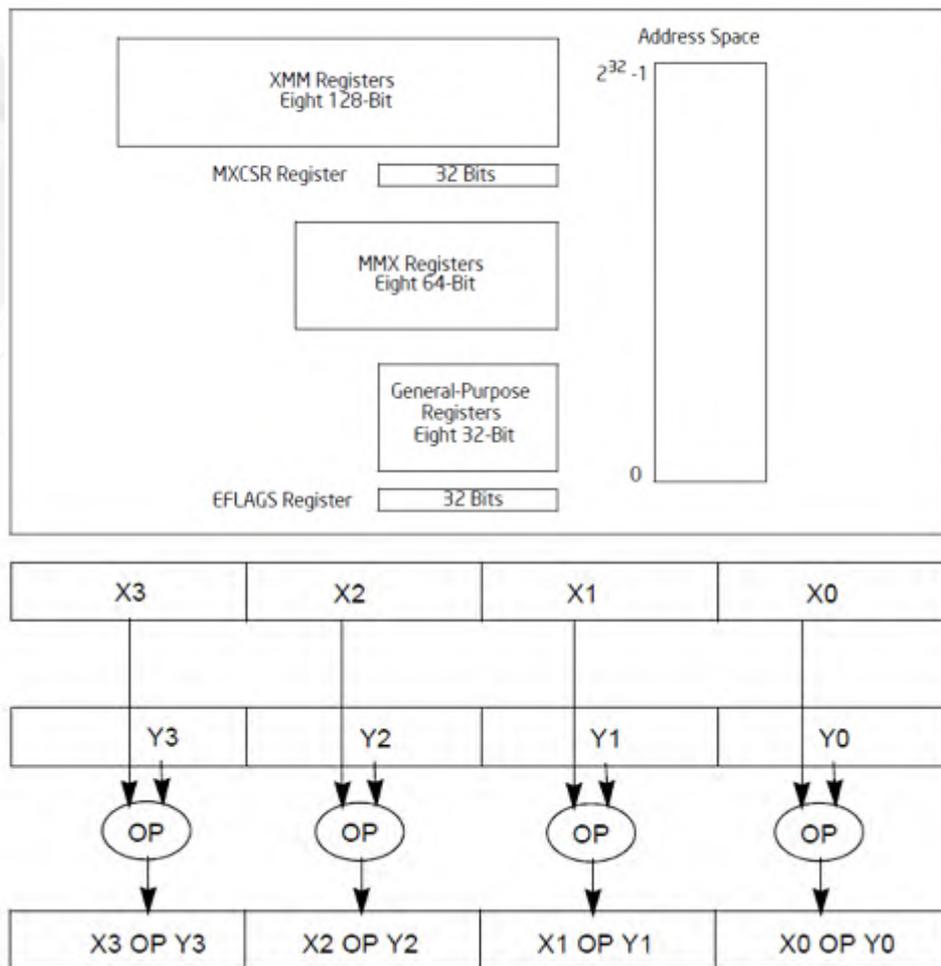
- Group : 128 (或256) 个数
- BitSize : group中最大值的位宽

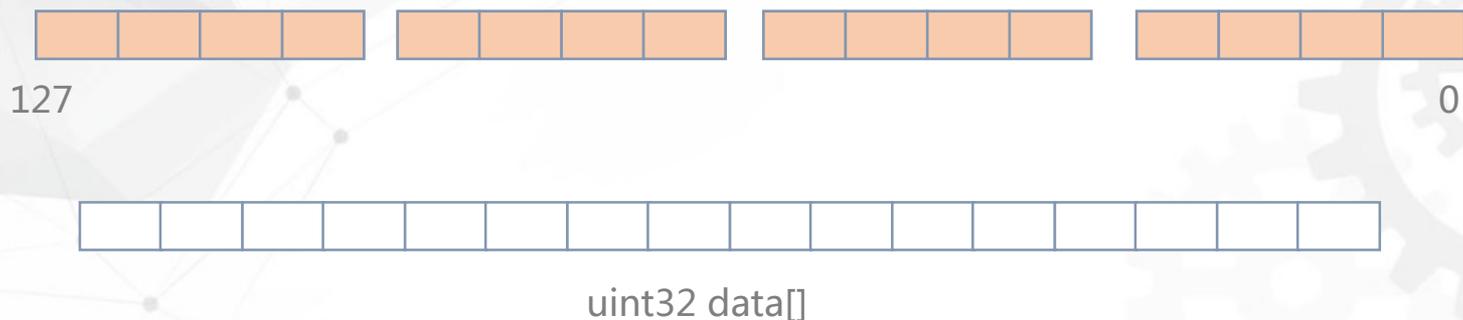


4bit

➤ GroupInfo

- BitSize
- MaxId

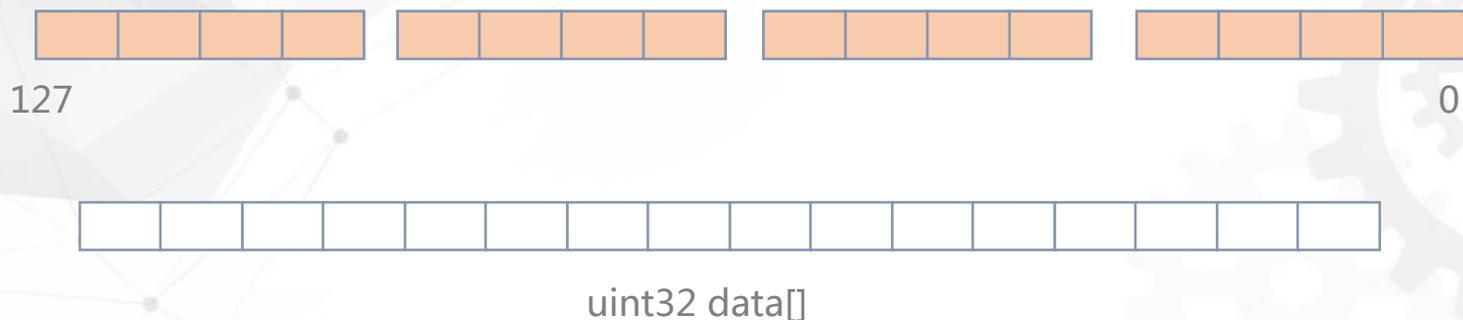




```
mask = _mm_set1_epi32((1U << 8) - 1);
```

```
ResReg = _mm_and_si128(Reg, mask);  
_mm_store_si128(res++, ResReg);
```

```
Reg = _mm_srli_epi32(Reg, 8);
```



- **Group**
 - 128/256
- **优点**
 - 并行
- **缺点**
 - 异常值影响压缩率

总结

SUMMARY



简洁高效的代码实现



合理的数据结构设计



合理利用体系结构特性



优秀的算法设计



THANKS

SequeMedia
威拓传媒

IT168.com

ITPUB

ChinaUnix