DTCC

2017第八届中国数据库技术大会

DATABASE TECHNOLOGY CONFERENCE CHINA 2017

宽表列存储在大数据分析中的应用与优化

卞昊穹

中国人民大学-数据库与智能信息检索实验室

博士研究生　　导师：杜小勇、陈跃国

https://github.com/dbiir/rainbow

宽表列存储在大数据分析中的应用与优化

什么样的查询

什么样的数据

怎么存？

哪种存储介质

哪种平台

啥样的数据格式/布局？

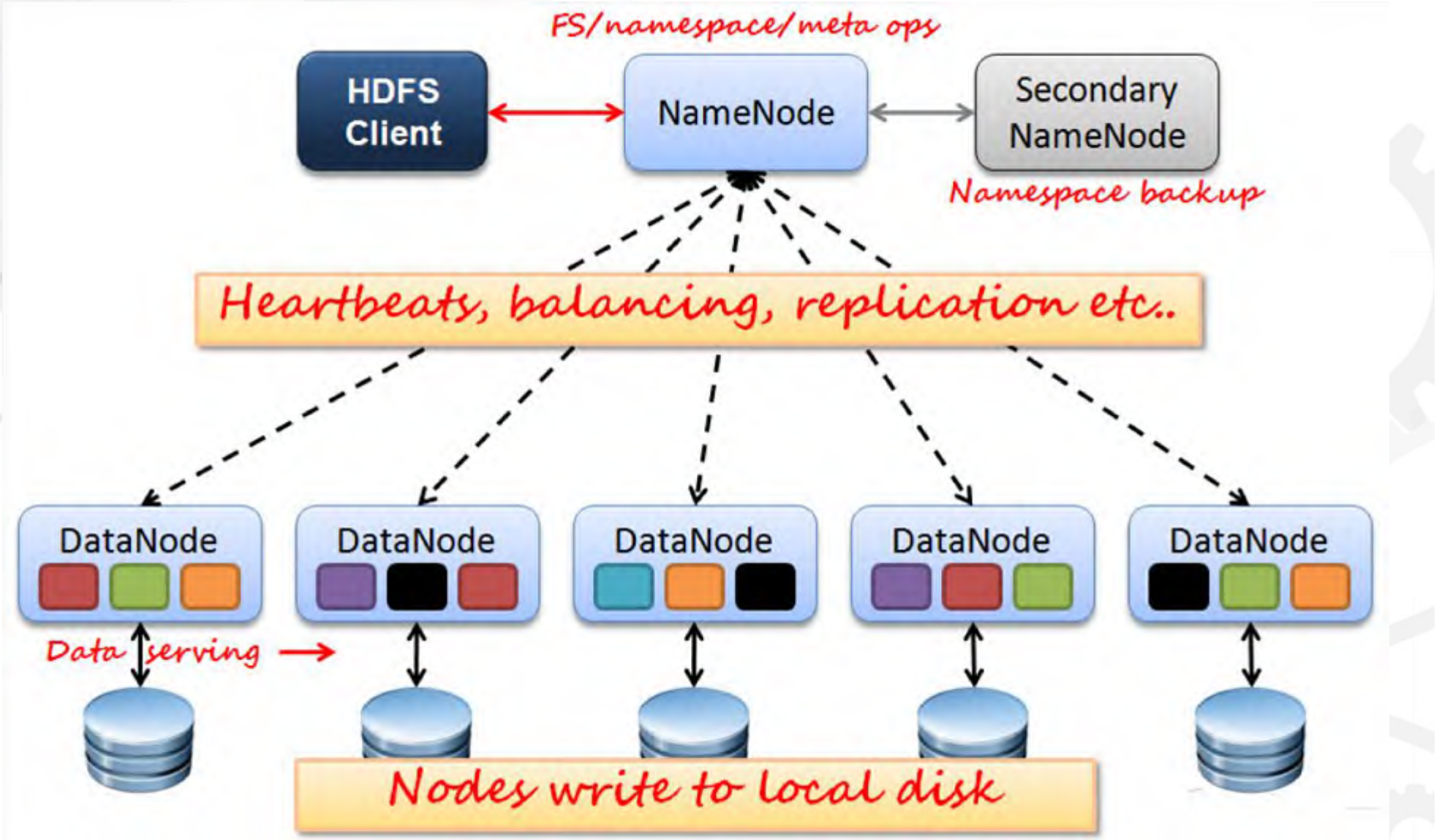

开发


改良


优化

各种数据都存在HDFS上

HDFS


FS/namespace/meta ops
HDFS Client
NameNode
Secondary NameNode
Namespace backup
Heartbeats, balancing, replication etc..
DataNode
DataNode
DataNode
DataNode
DataNode
Data serving
Nodes write to local disk

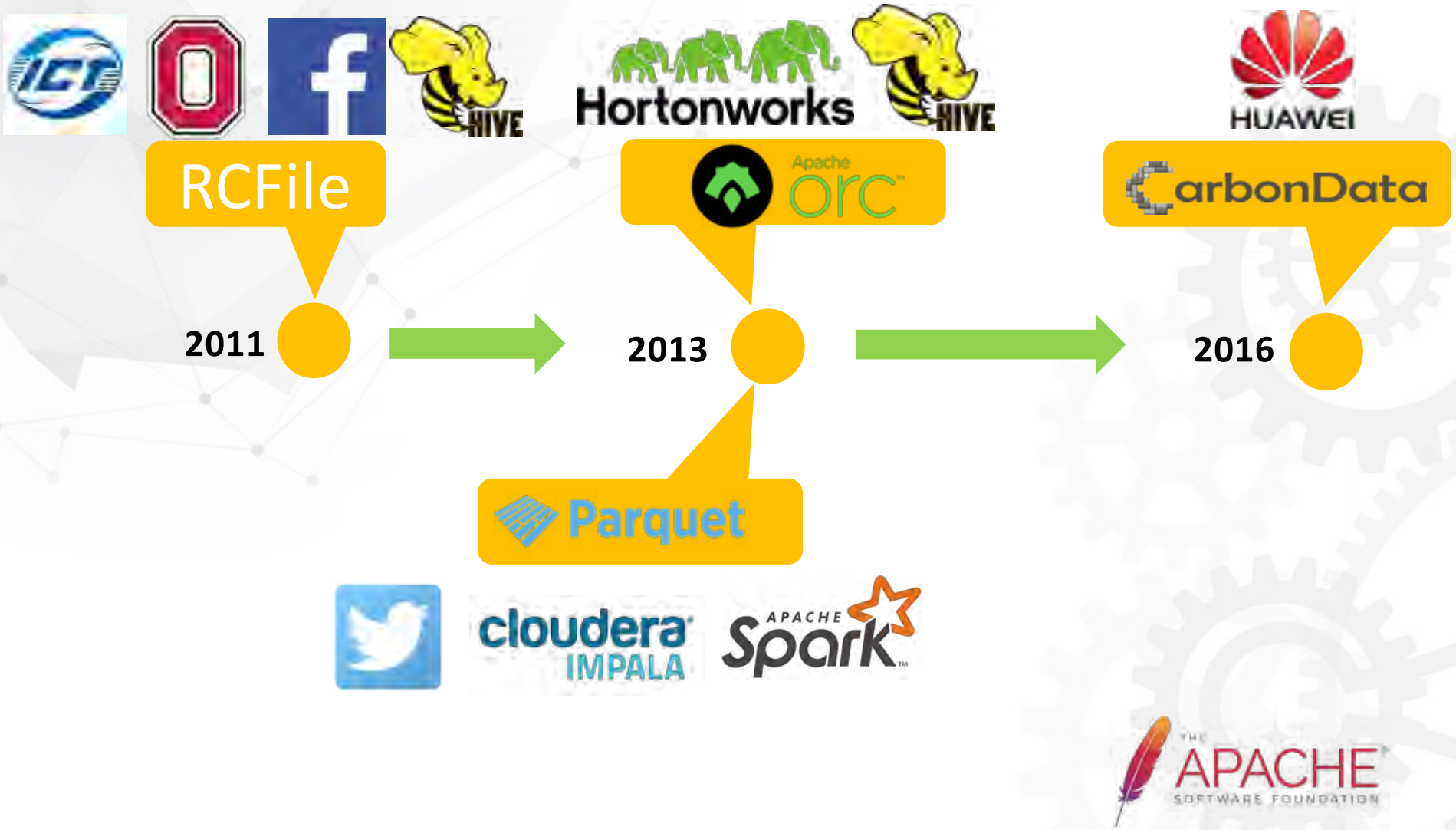
2003 GFS
2004 MapReduce
2006 BigTable
2010 Dremel

2007

2008-
2009

HDFS存储格式

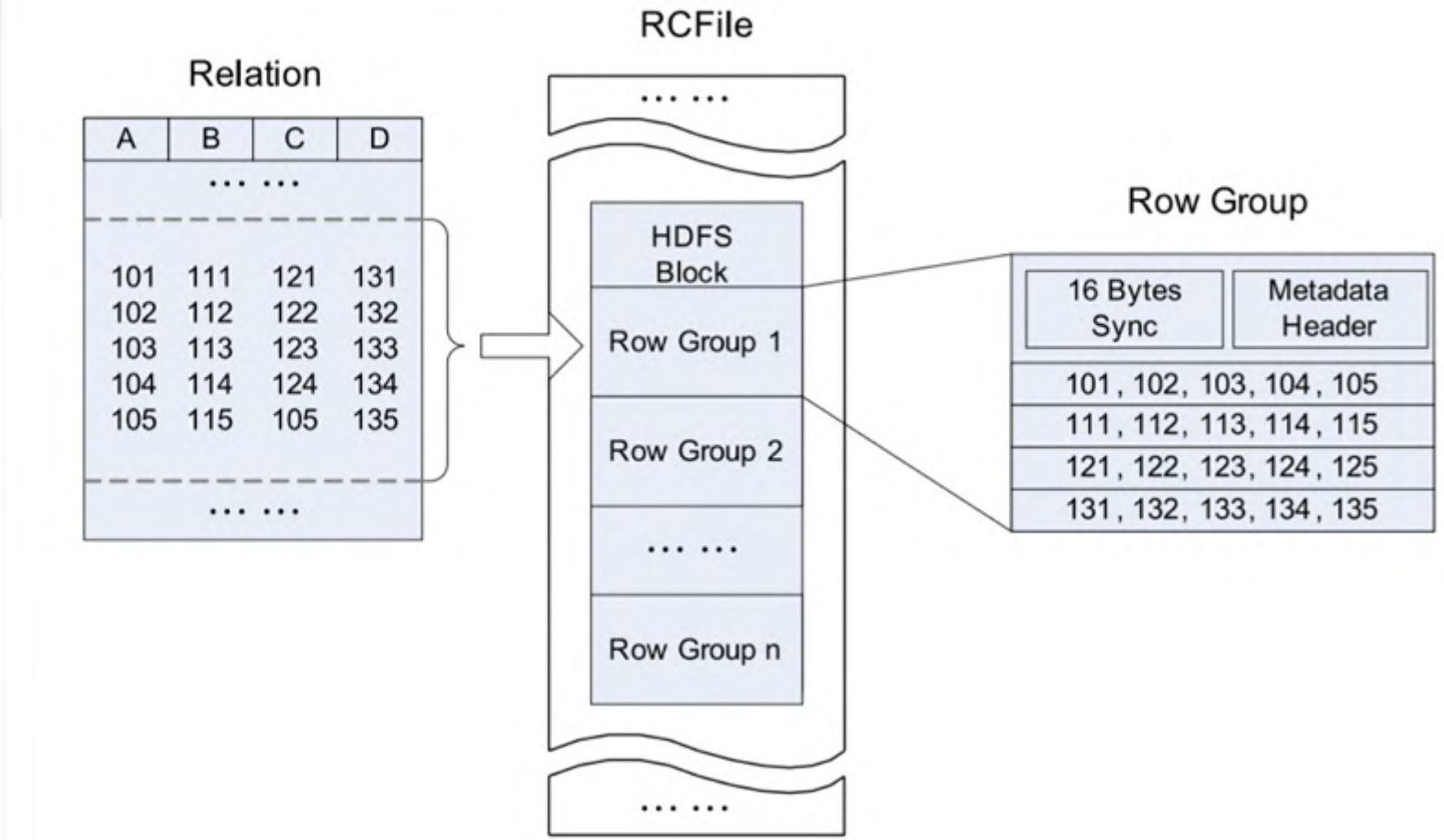
- Text File
- Sequential File
- RCFile/ORC
- Parquet
- ……



{parquet,orc,carbondata}.apache.org

HDFS存储格式

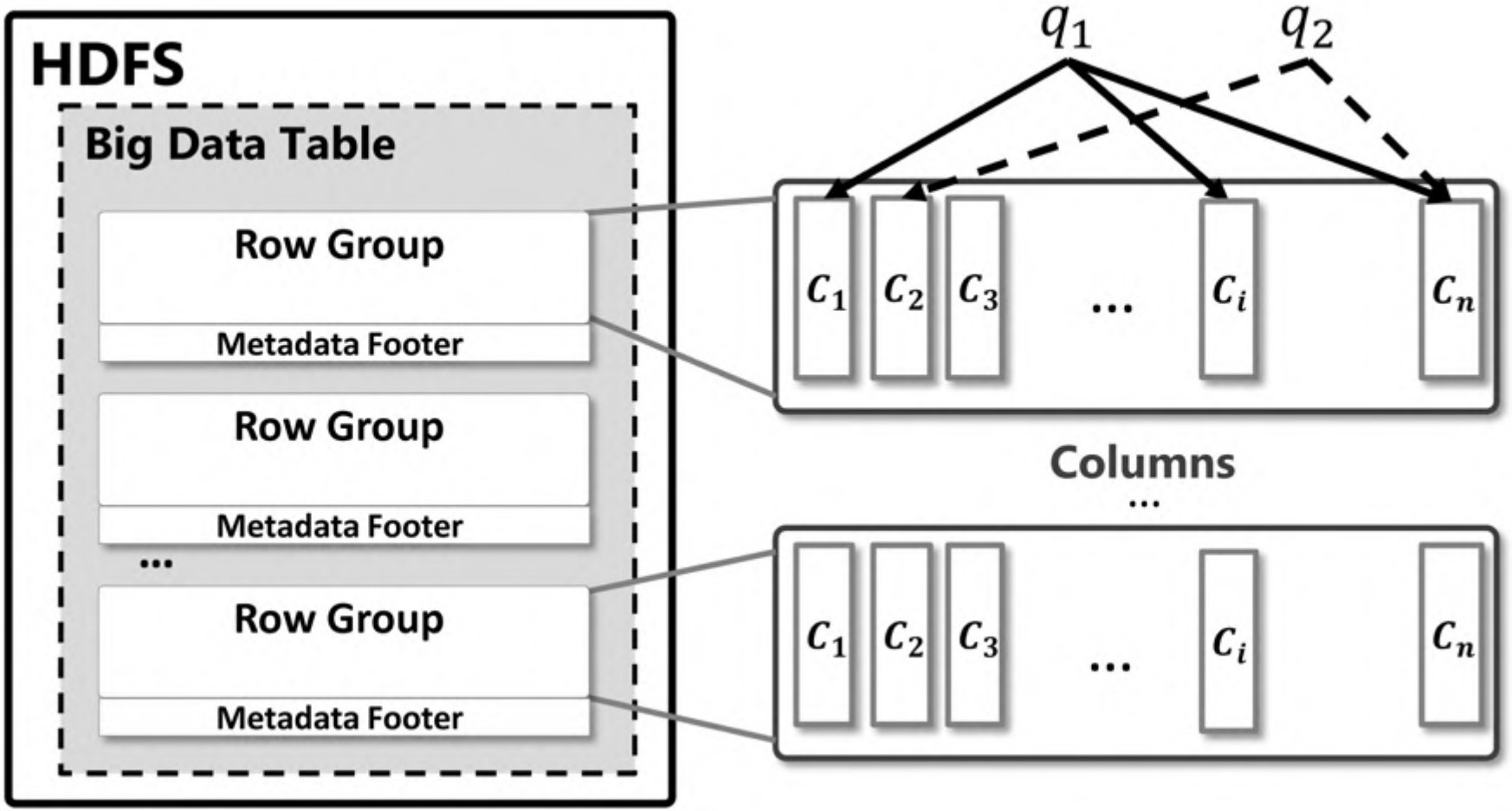
RCFile, Yongqiang He et.al, ICDE 2011



宽表

- >100列，常见上千、上万列
- 在大数据分析中普遍存在：
  - 某银行分析负载：200-1000列
  - Microsoft Bing Search Log：>1000列
  - 某互联网公司用户画像数据：>10000列
  - 华为很多客户产品线：>200列，不断增长
  - 反规范化避免连接（WideTable，Yinan Li et.al. VLDB2014)：TPC-H -> 宽表
- 宽表存储在HDFS上，采用列存

宽表


HDFS
Big Data Table
Row Group
Metadata Footer
Row Group
Metadata Footer
...
Row Group
Metadata Footer
q_1
q_2
C_1 C_2 C_3 ... C_i C_n
Columns
...
C_1 C_2 C_3 ... C_i C_n

宽表

- 相对于行存，列存极大提高了宽表上分析负载的I/O性能
- 当一个查询只访问1000列中的20个时，无须读取整个数据块，最大限度减少了所读取的数据量

宽表+列存储=  ？

- Parquet格式，**400GB**，1187列，单机
- Spark SQL读取8列，**总数据量0.3%（1.2GB）**
- 磁盘顺序读带宽**100MB/s**

$$\frac{1.2GB}{100MB/s} = 12s$$

理想

现实很骨感

968”

时间都去哪了

CPU Overhead？

Tuple组装？任务调度？计算？Shuffle？

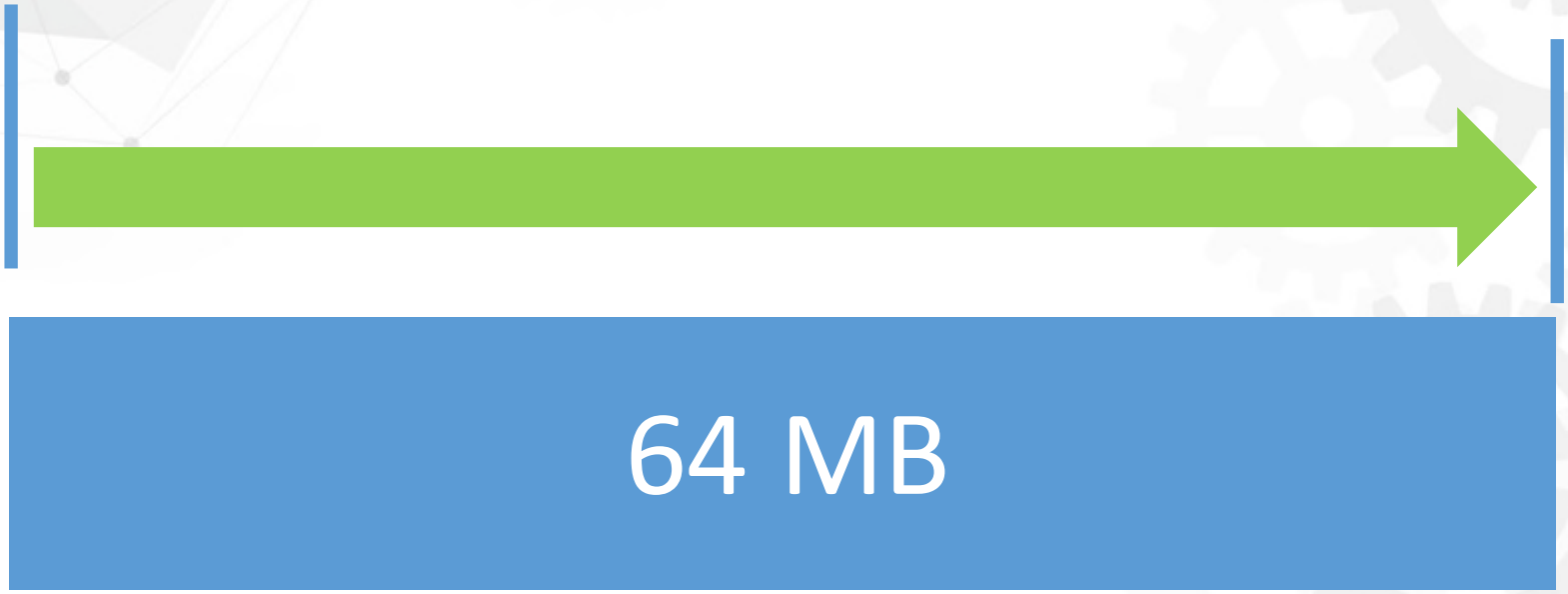
Parquet Reader直接读取：907s >> 12s

I/O

读取数据不连续

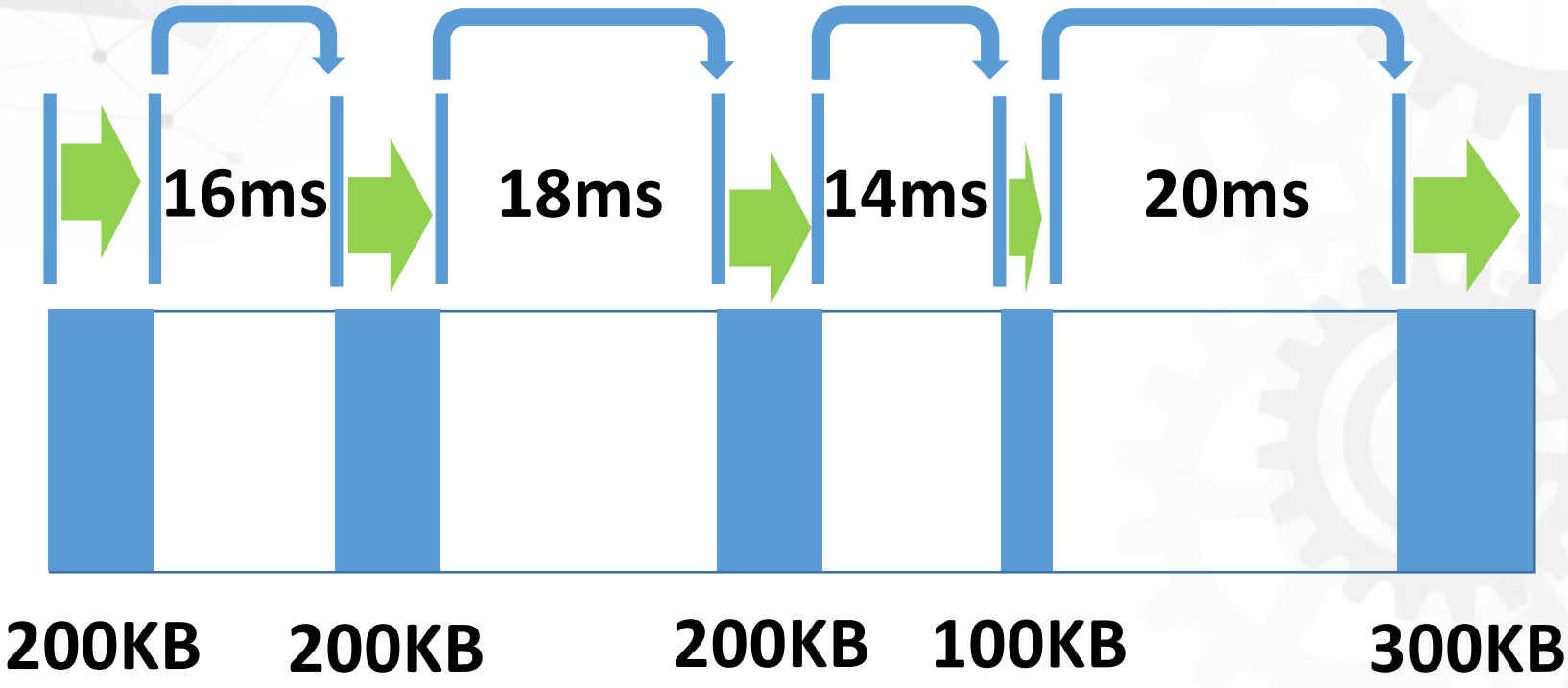
以读取一个数据块为例

$$\frac{64MB}{100MB/s} = 640ms$$

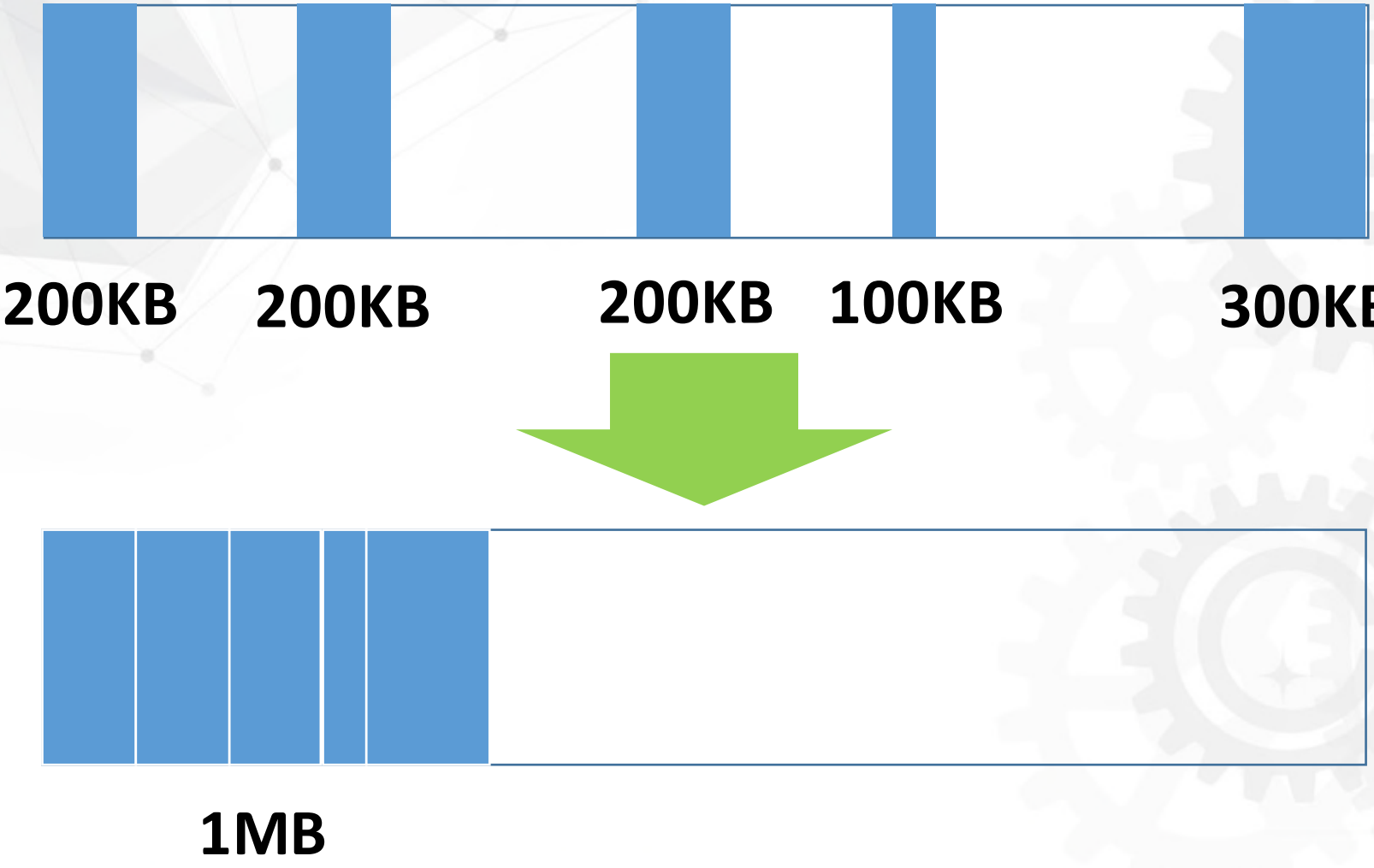


以读取一个数据块为例

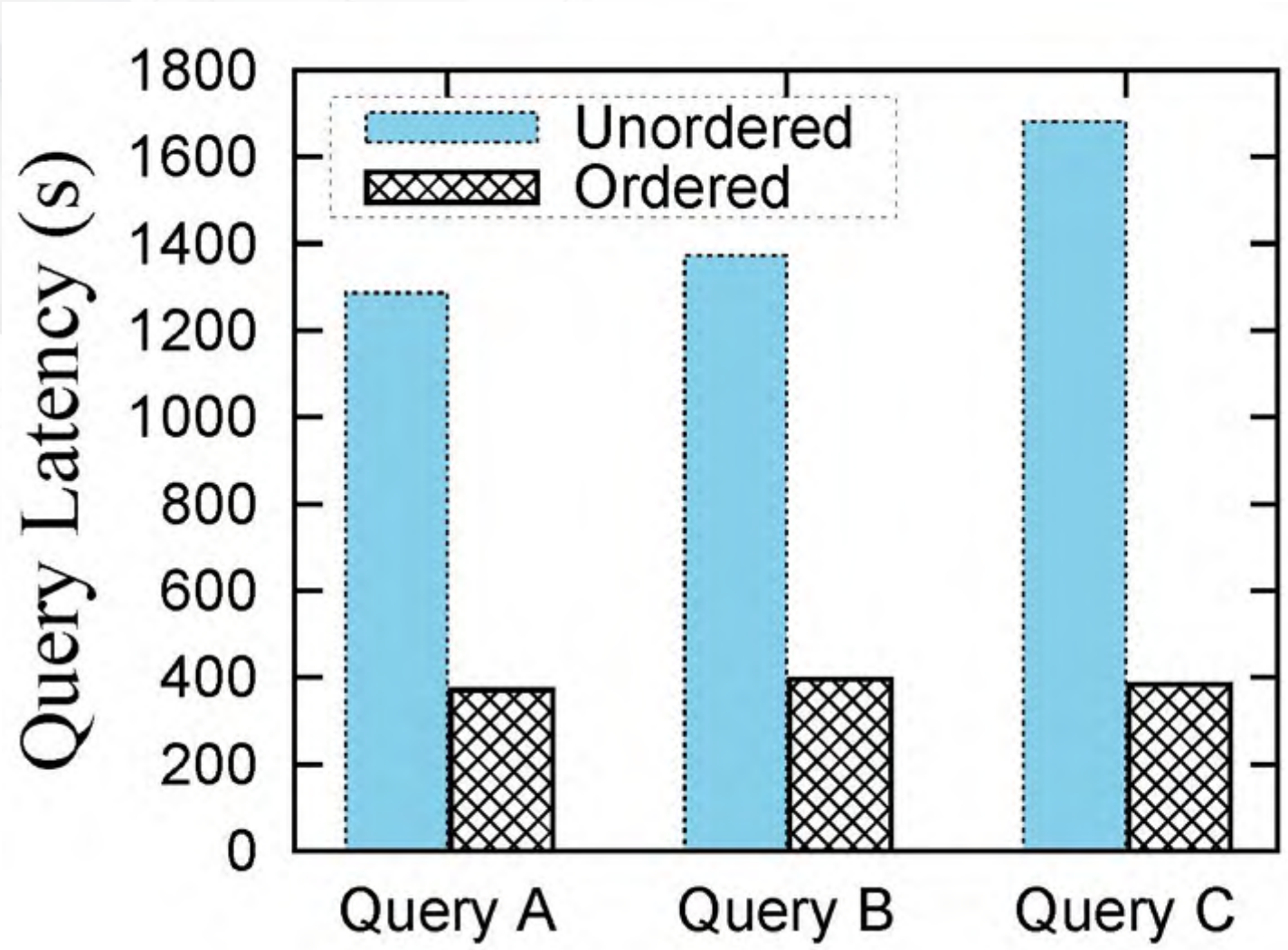
$$\frac{1MB}{100MB/s} = 10ms + 68ms = 78ms$$



将要访问的列连续存放



将要访问的列连续存放


Query Latency (s)
1800
1600
1400
1200
1000
800
600
400
200
0
Unordered
Ordered
Query A
Query B
Query C

磁盘带宽并不低，但IOPS很低（≈200）

存储技术发展史=与Disk Seek的斗争史

- 聚簇索引
- Buffer/Cache
- LSM-Tree（Hbase、LevelDB、Cassandra……）
- 大数据块（MB级别）
- 文件系统中的extents、pre-allocation、block group
- ……

基于Common Sense: 将频繁访问的列连续存放

- Materialized View
  - C-store (VLDB2005)、Vertica (VLDB2012)
- Vertical Partitioning
  - Hill-Climb (VLDB2003)、AutoPart (SSDBM 2004)
- Column Grouping
  - Trojan Layout (SoCC 2011)

存在问题：

1）基于common sense，隐含假设：

对于一次I/O操作，seek cost是常量 (如10ms)

一次I/O要么有Seek Cost，要么没有Seek Cost

2）要么只针对特定Query，要么数据冗余巨大

Bing Search Log：

一个宽表每天新增5TB数据，500+不同的Query。

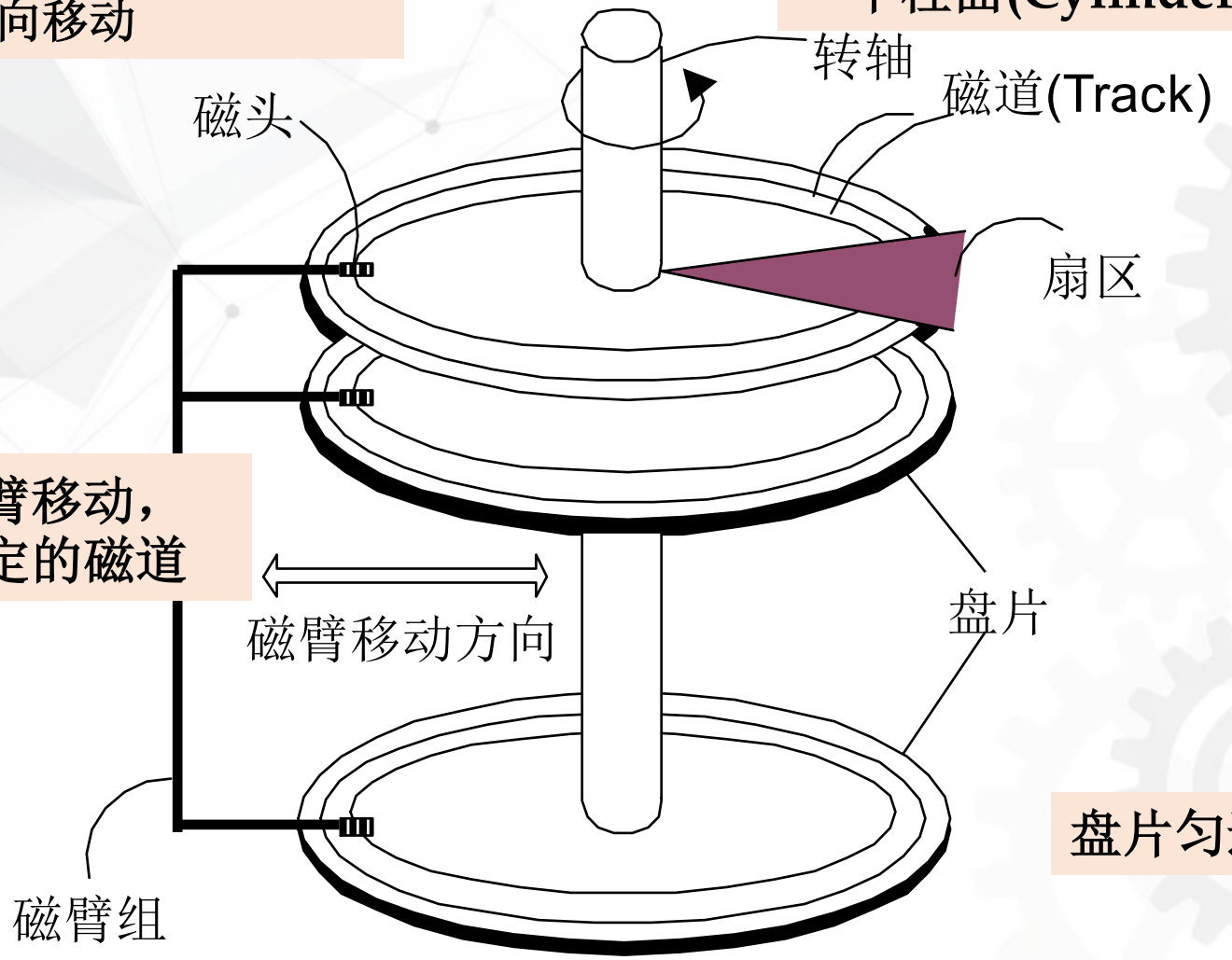
物化视图：消耗121x 的额外存储空间

Vertical Partitioning：无法解决查询间的冲突

真实的Seek Cost Model ?

磁臂在伺服电机带动下在磁盘半径方向移动

多个磁头下的多个磁道组成一个柱面(Cylinder)



磁头跟随磁臂移动，以定位到指定的磁道

盘片匀速旋转

Disk Seek Time

Disk Seek：磁头从一个柱面移动到另一个柱面

Seek Time是所移动柱面数(柱面距离，distance in cylinders)的函数

- **Very short seeks** (如1-4个柱面)：seek time主要是磁头稳定时间(settle time)，一个**常数**
- **Short seeks** (如<=400个柱面)：磁臂有一个匀加速和匀减速的过程，

  seek time=a+b*sqrt(#cylinders)
- **Long seeks** (如>400个柱面): 磁臂存在匀速运动过程：**seek time=c+d*(#cylinders)**

然而

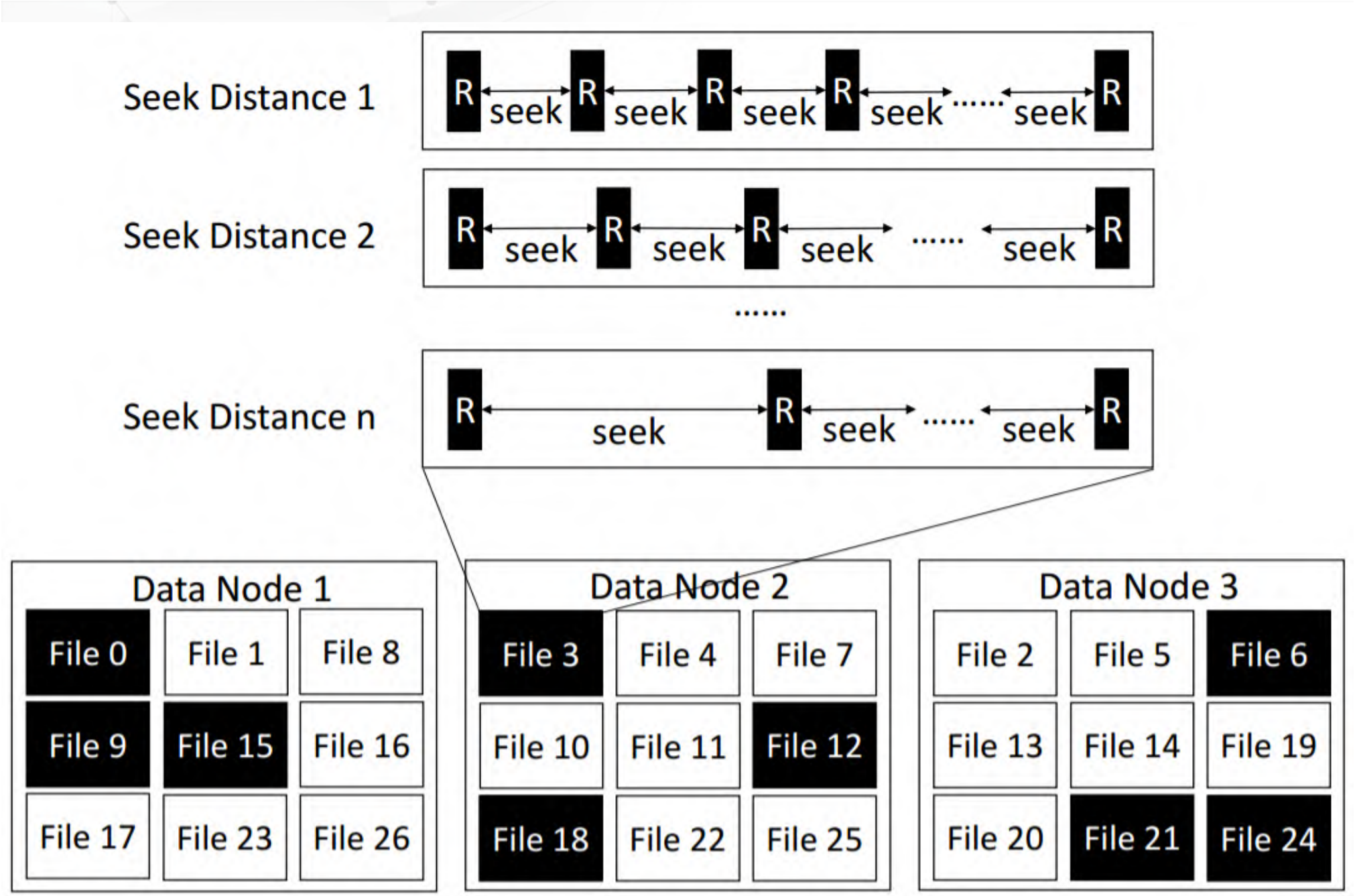
磁盘制造商将磁盘的物理参数隐藏在磁盘内部

文件系统、操作系统、HDFS等软件层也会影响数据的物理存储

所以

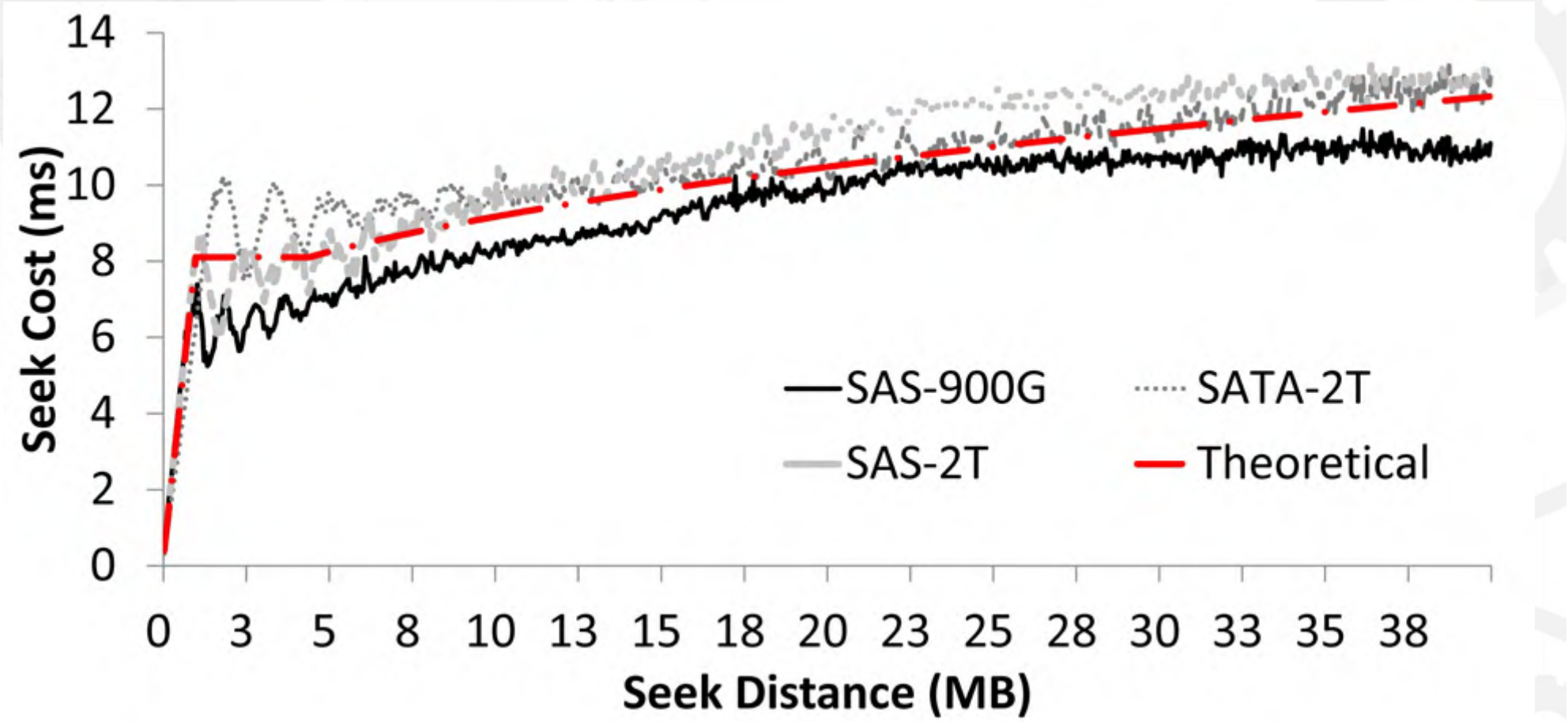
大量在HDFS数据块内的Seek操作→统计意义上的实际seek time

用微分思想，通过不同距离的Seek得到seek cost model


Seek Distance 1
R seek R seek R seek R seek ..... seek R
Seek Distance 2
R seek R seek R seek ..... seek R
.....
Seek Distance n
R seek R seek ..... seek R
Data Node 1
File 0
File 1
File 8
File 9
File 15
File 16
File 17
File 23
File 26
Data Node 2
File 3
File 4
File 7
File 10
File 11
File 12
File 18
File 22
File 25
Data Node 3
File 2
File 5
File 6
File 13
File 14
File 19
File 20
File 21
File 24

不同Seek distance的间隔： 50KB

相同distance执行100次seek， 起始Offset随机



如何权衡Query间的冲突？

基本认识

Seek Cost并不是非有即无的，而是一个关于seek distance的连续函数

优化列的存储顺序就可以提高宽表上的I/O性能

Query Seek Cost

给定一个列的存储顺序S = $(c_1, c_2, \ldots, c_n)$，一个query q 的列访问模式$C_q = (c_{q,1}, c_{q,2}, \ldots, c_{q,m})$，一个计算两个数据对象之间seek cost的函数f，对于一个有 N 个row group的数据表，q 访问该数据表的seek cost为：

$$Cost(q,S) = N \times \left(\varepsilon + \sum_{i=1}^{m-1} f\left(c_{q,i}, c_{q,i+1}\right) \right)$$

Workload Seek Cost

给定一个Workload Q，对于每一个查询$q \in Q$，给定一个权重 w_q，则 Q 的seek cost为：

$$Cost(Q,S) = \sum_{q \in Q} \left(w_q \times Cost(q,S) \right)$$

列排序问题定义

给定一个Workload Q，寻找一个最优的列存储顺序 S^*，使得 Q 的workload seek cost最小，即：

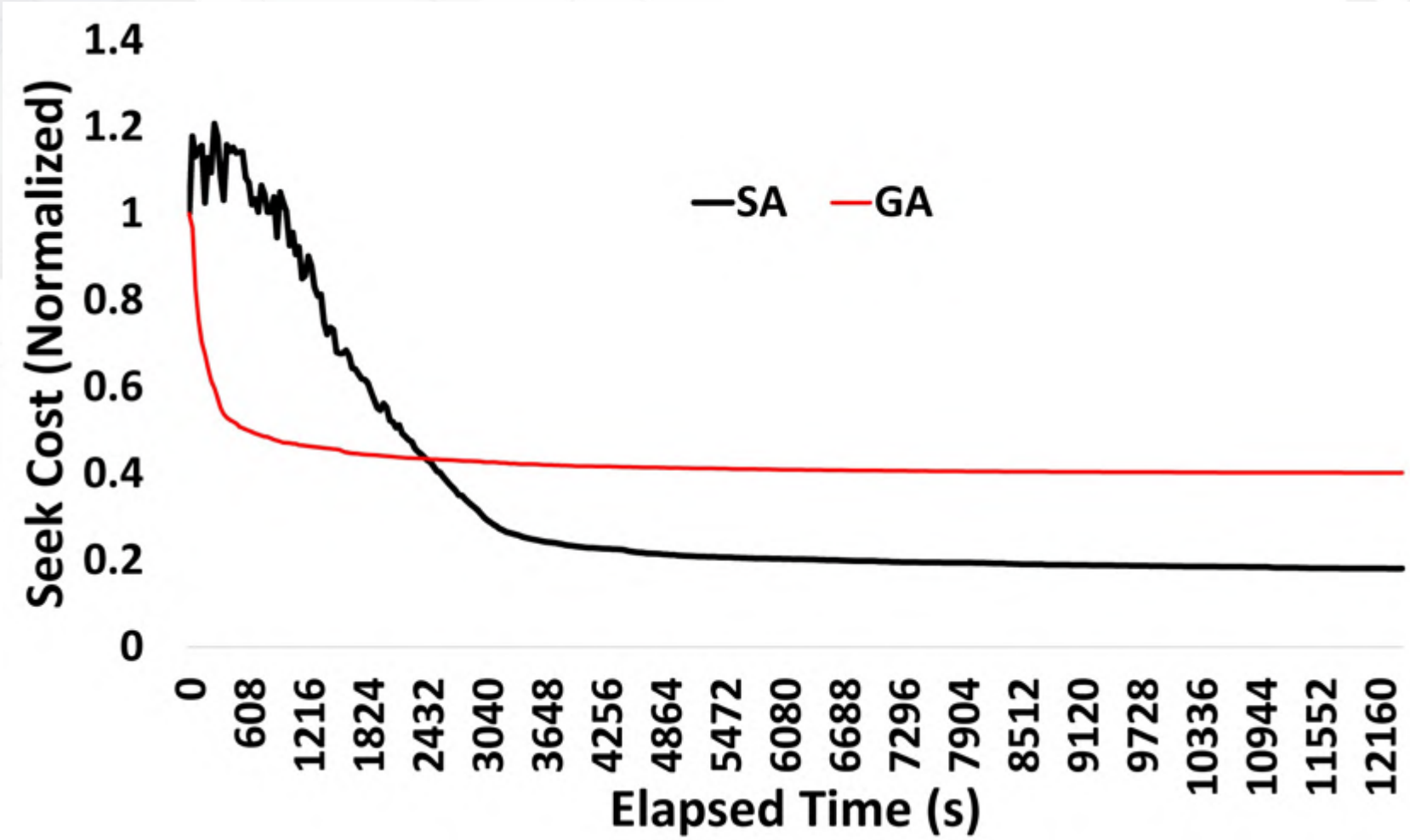
$$S^* = \arg\min_{S} Cost(Q, S)$$

可以将哈密尔顿路径问题归约到列排序问题，从而证明列排序问题为NP-Hard，即**多项式时间内无法验证和求解**

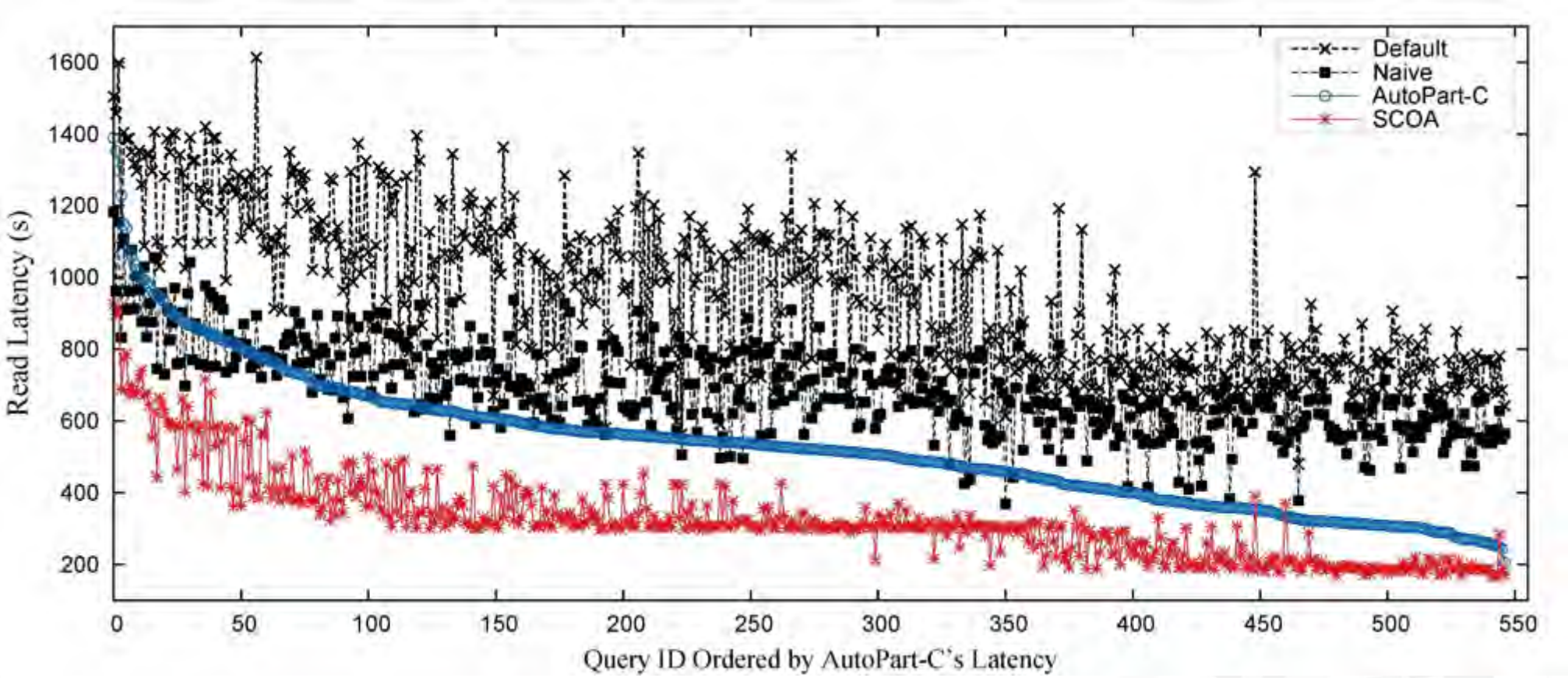
列排序算法

- 枚举
  - Trojan Layout (A. Jindal et.al. SoCC 2011)
  - 只适用于窄表，而窄表上Seek开销并不大
- 启发式 (Heuristic) 算法
  - Meta-Heuristic
    - 遗传算法 (Genetic Algorithm, **GA**)
    - 模拟退火 (Simulated Annealing, **SA**)
  - Greedy
    - Hill-Climbing
    - AutoPart

GA vs. SA


Seek Cost (Normalized)
1.4
1.2
1
0.8
0.6
0.4
0.2
0
SA
GA
0
608
1216
1824
2432
3040
3648
4256
4864
5472
6080
6688
7296
7904
8512
9120
9728
10336
10944
11552
12160
Elapsed Time (s)

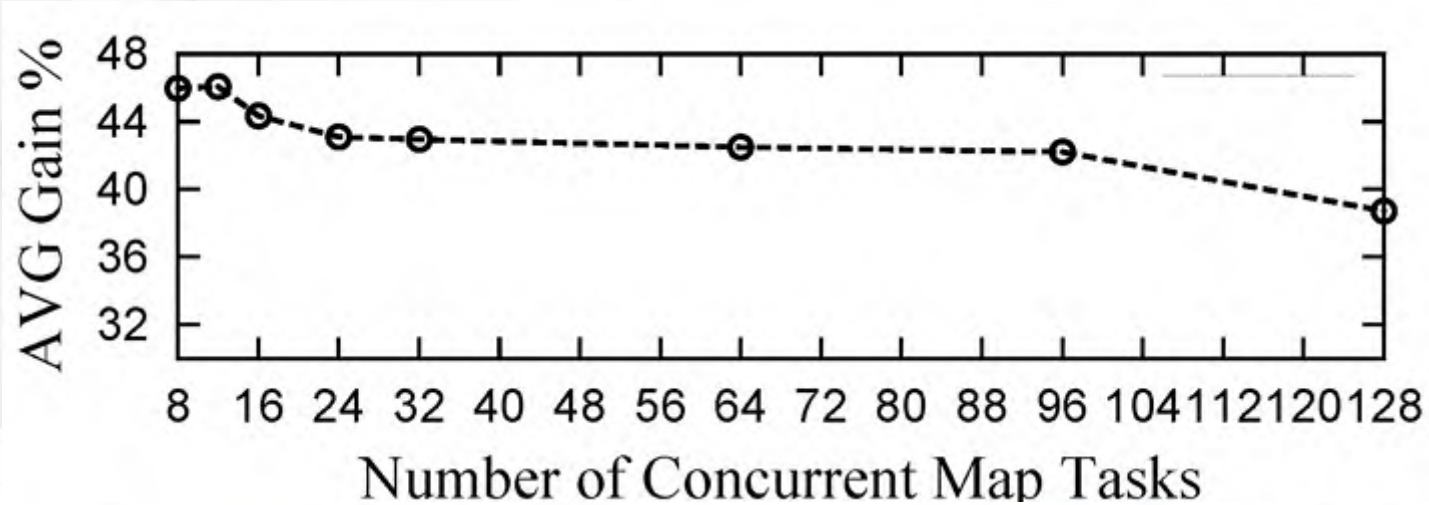
Meta-Heuristic vs. Greedy

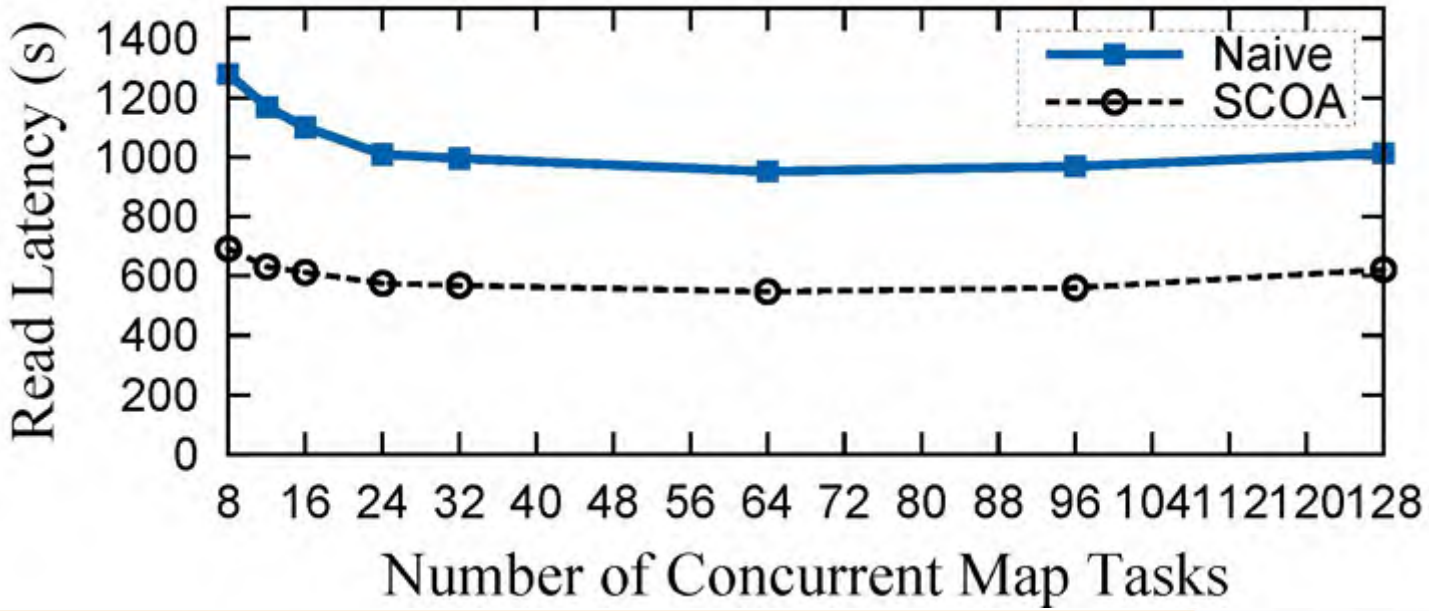


SCOA

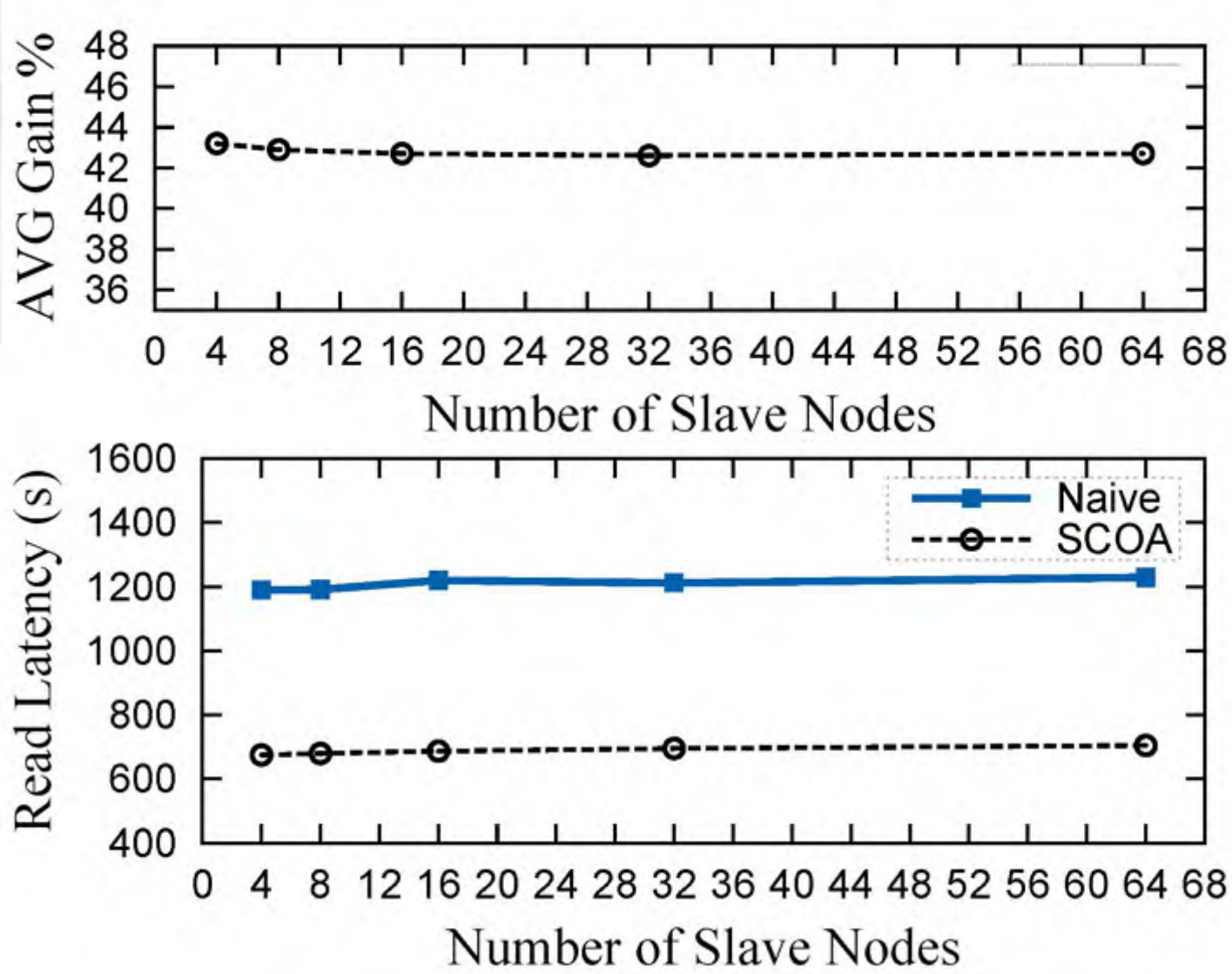
(SA-based Column Ordering Algorithm)

并发I/O的影响

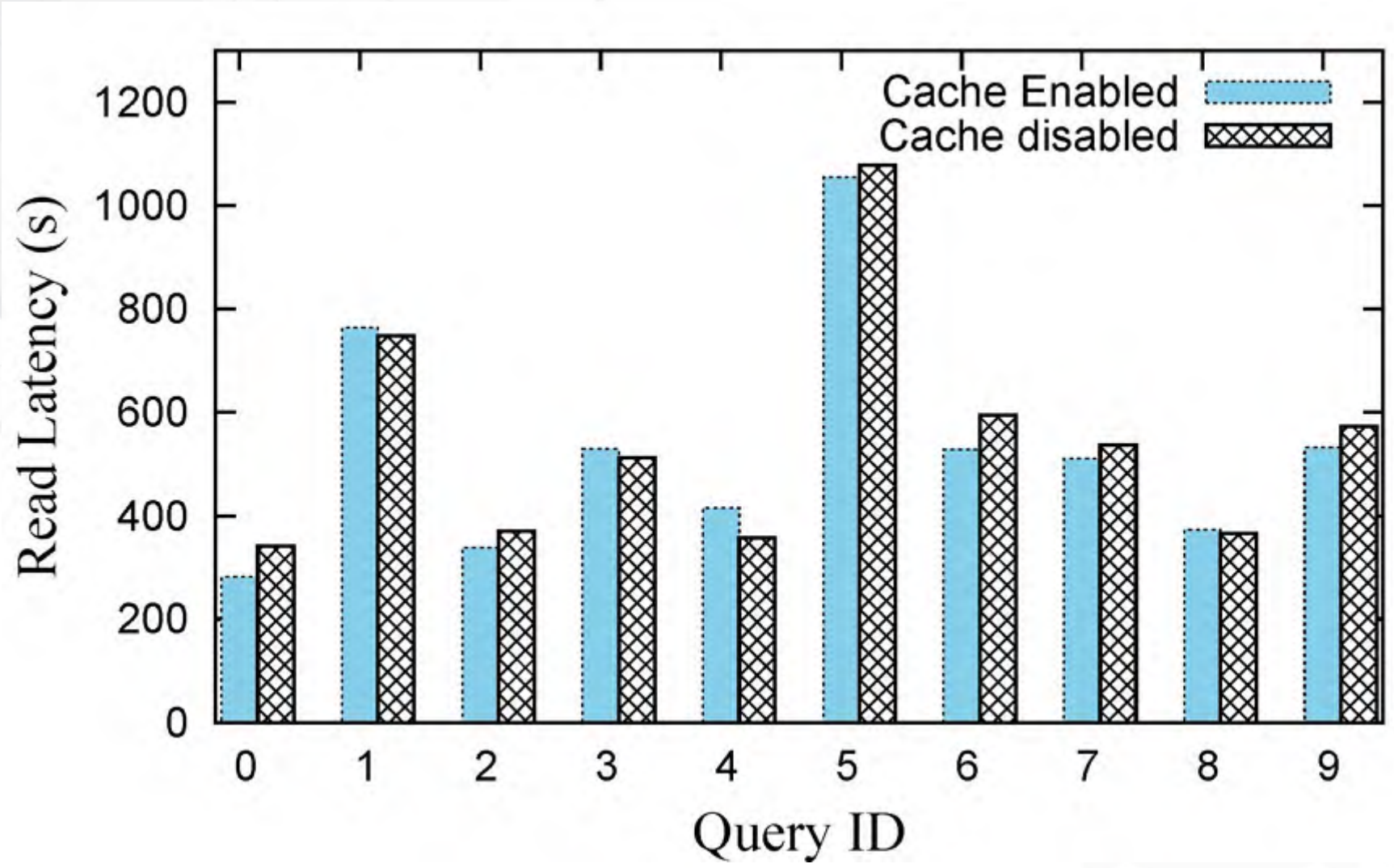

AVG Gain %
48
44
40
36
32
8 16 24 32 40 48 56 64 72 80 88 96 104 112 120 128
Number of Concurrent Map Tasks


Read Latency (s)
1400
1200
1000
800
600
400
200
0
Naive
SCOA
8 16 24 32 40 48 56 64 72 80 88 96 104 112 120 128
Number of Concurrent Map Tasks

集群节点数的影响


AVG Gain %
48
46
44
42
40
38
36
0 4 8 12 16 20 24 28 32 36 40 44 48 52 56 60 64 68
Number of Slave Nodes
Read Latency (s)
1600
1400
1200
1000
800
600
400
Naive
SCOA
0 4 8 12 16 20 24 28 32 36 40 44 48 52 56 60 64 68
Number of Slave Nodes

文件缓存的影响

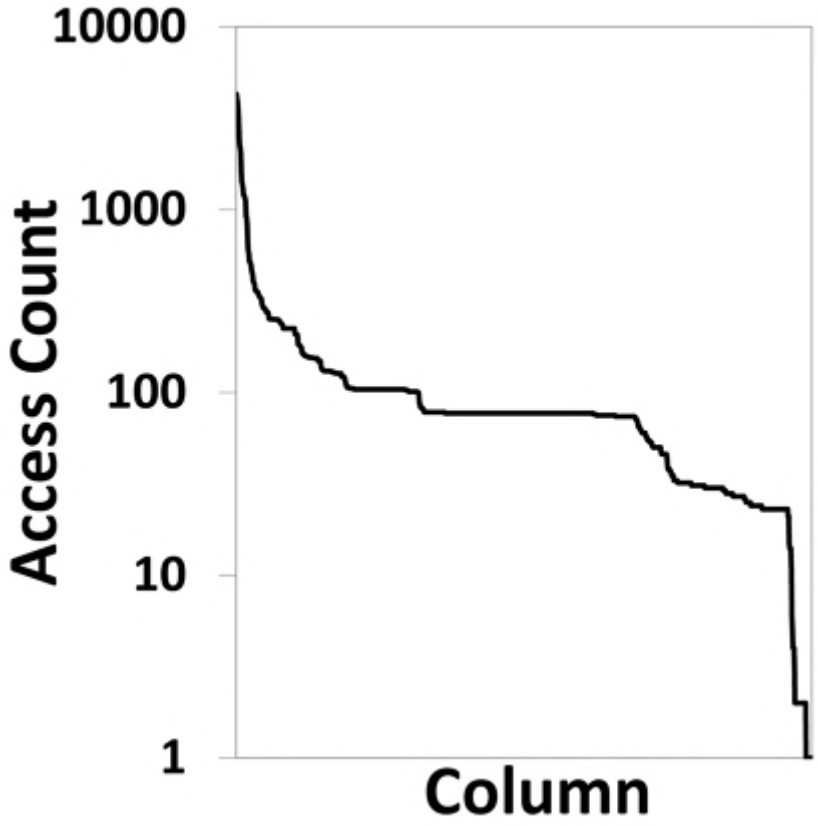

Cache Enabled
Cache disabled
Read Latency (s)
1200
1000
800
600
400
200
0
0
1
2
3
4
5
6
7
8
9
Query ID

还能进一步提高吗？

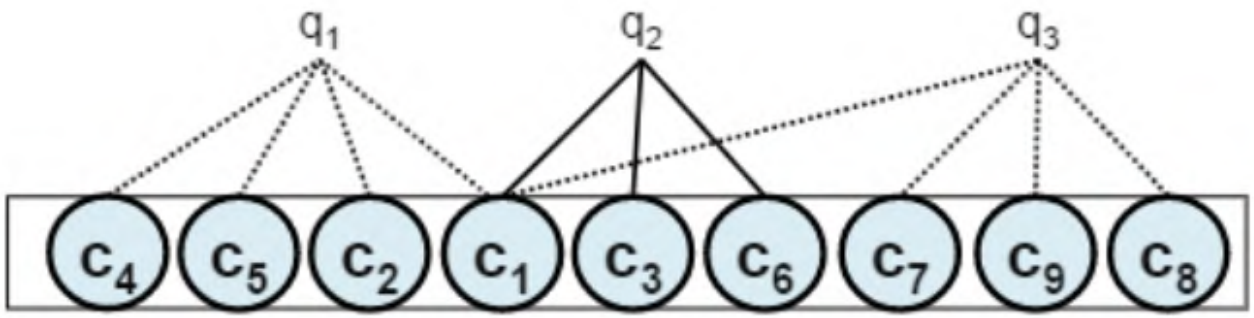
列复制

少量的列被很多查询“竞争”访问

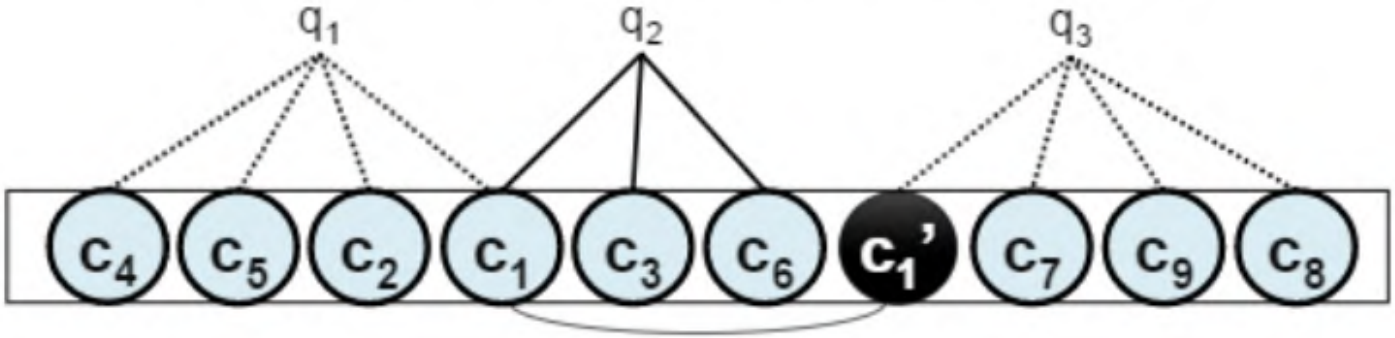
将这些被竞争使用的列复制出多个副本，放置到合适的位置，可以进一步提高I/O性能



列复制



(a) Column ordering produced by SCOA



(b) Duplicate c_1 and insert it between c_6 and c_7

列复制问题

如何找到需要复制的列？

- 被频繁访问 ≠ 被竞争访问

复制多少份？

- 太少效果不好
- 太多浪费空间、影响数据导入性能
- 存储空间约束

副本放到哪里去？

- 放到seek cost最小的位置？
- 后复制出的列会影响之前列的位置

列复制算法

聚类与复制是两个分离的步骤

聚类中的距离/相似度、收敛条件难以设定

从Query出发

- 将query根据access pattern聚类
- 分析不同query类之间的访问冲突
- 根据冲突的情况决定是否复制和复制多少份

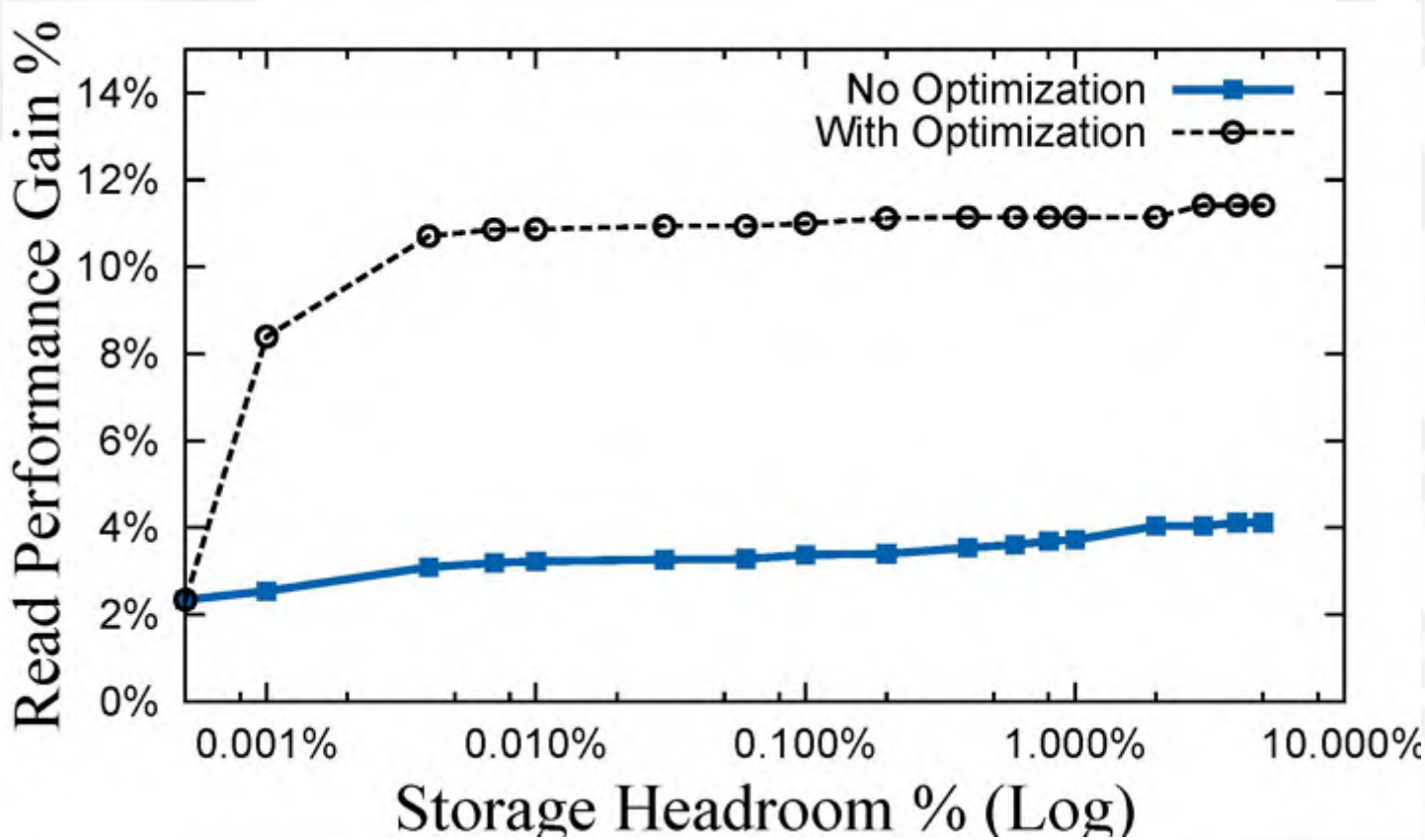
从Column出发

- 分多轮进行
- 每一轮找出一个复制收益最大的column
- 一个列可以被多轮反复复制
- 直到达到空间约束条件为止

列复制算法

列复制过程中副本位置通过贪心策略决定

复制出的Column Order再经过SCOA优化



列复制的副作用

查询需要被重写

```
SELECT Market AS Dim_Market,
       QueryHour AS Dim_DateTime,
       DataCenter AS Dim_DataCenter,
       SUM(DistinctQCount) AS DSQ
FROM   Log
```

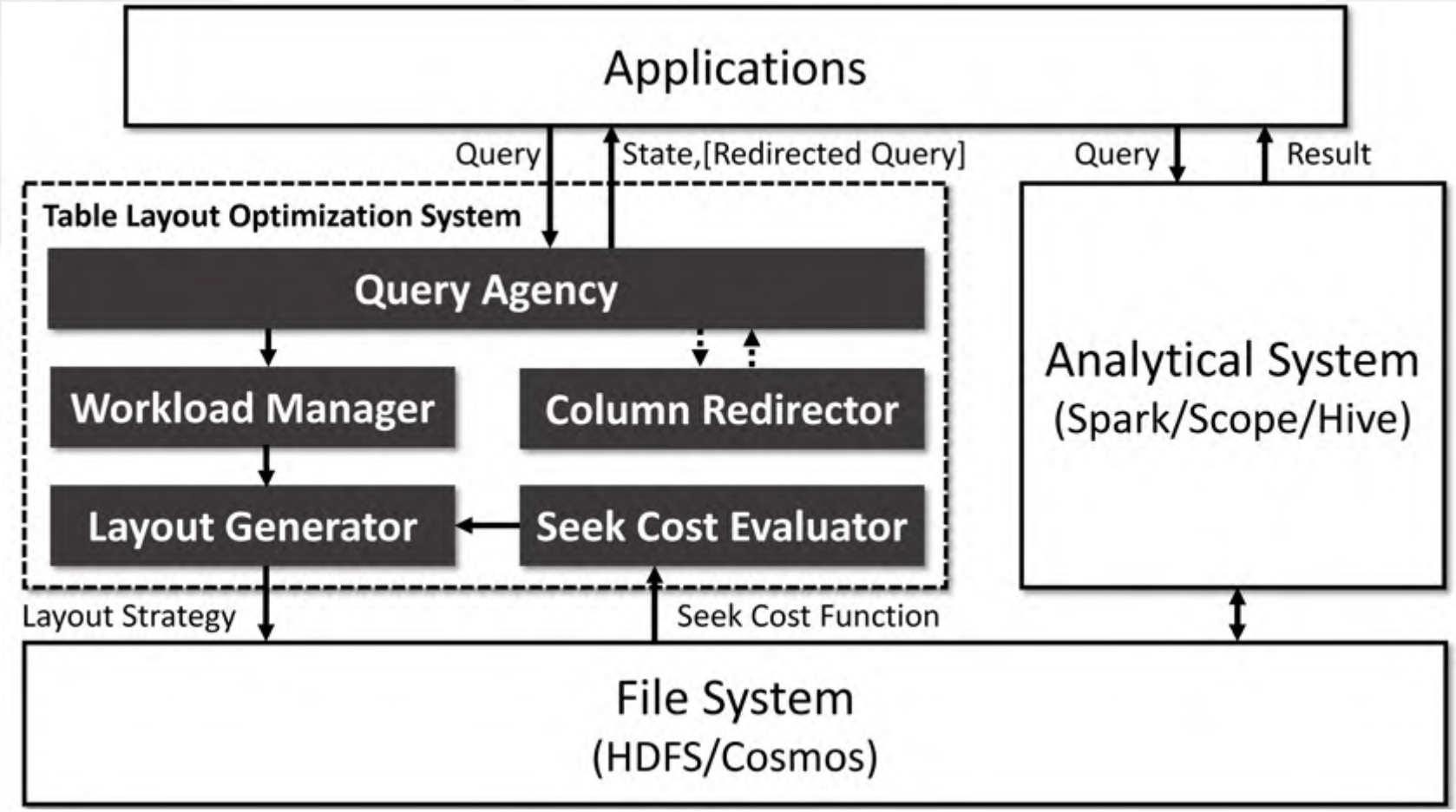
```
SELECT Market_3 AS Dim_Market,
       QueryHour_2 AS Dim_DateTime,
       DataCenter AS Dim_DataCenter,
       SUM(DistinctQCount_1) AS DSQ
FROM   Log
```

查询目前无法跨新旧数据（需要修改现有系统）

导入数据的性能受影响

Storage Headroom	0	0.1%	1%	5%
#Cloumns	1187	1211	1291	1354
Data Loading Time	296min	301min	319min	335min

自适应列存储优化框架


Applications
Query
State,[Redirected Query]
Query
Result
Table Layout Optimization System
Query Agency
Workload Manager
Column Redirector
Layout Generator
Seek Cost Evaluator
Analytical System
(Spark/Scope/Hive)
Layout Strategy
Seek Cost Function
File System
(HDFS/Cosmos)

https://github.com/dbiir/rainbow

项目组件:

- **rainbow-common**：公共的接口和工具.
- **rainbow-core**：提供RESTful API.
- **rainbow-evaluate**：提交查询到Spark 或直接读取HDFS上的Parqeut文件来评估列排序/复制的效果.
- **rainbow-layout**：计算和生成列排序/复制的布局.
- **rainbow-redirect** ：对于复制后的数据，重写查询.
- **rainbow-seek**：在本地文件系统或者HDFS上执行Seek，测出Seek Cost Function.

论文发表

Wide Table Layout Optimization based on Column Ordering and Duplication

Microsoft Research
微软亚洲研究院

Haoqiong Bian[1], Ying Yan[2]*, Wenbo Tao[3], Liang Jeff Chen[2], Yueguo Chen[1]*,
Xiaoyong Du[1], Thomas Moscibroda[2]
[1]Renmin University of China, [2]Microsoft Research, [3]MIT
[1]{bianhq, chenyueguo, duyong}@ruc.edu.cn,
[2]{ying.yan, jeche, moscitho}@microsoft.com, [3]wenbo@mit.edu

Accepted By SIGMOD 2017

杜小勇

陈跃国

闫莺

陶文博

陈亮
Liang Jeff Chen

Thomas
Moscibroda

感谢：
CCF数据库专业委员会、
DTCC 2017各主办方

http://iir.ruc.edu.cn/~bianh/

bianhaoqiong@gmail.com


FAQ
THANKS
SequeMedia
盛拓传媒
IT168.com
ITPUB
ChinaUnix