CarbonData应用实践和未来规划

李昆 2017-09

Agenda

- 为什么需要CarbonData
- CarbonData介绍
- 应用场景和调优介绍
- 未来计划

企业中包含多种数据应用，从商业智能、批处理到机器学习

Report & Dashboard

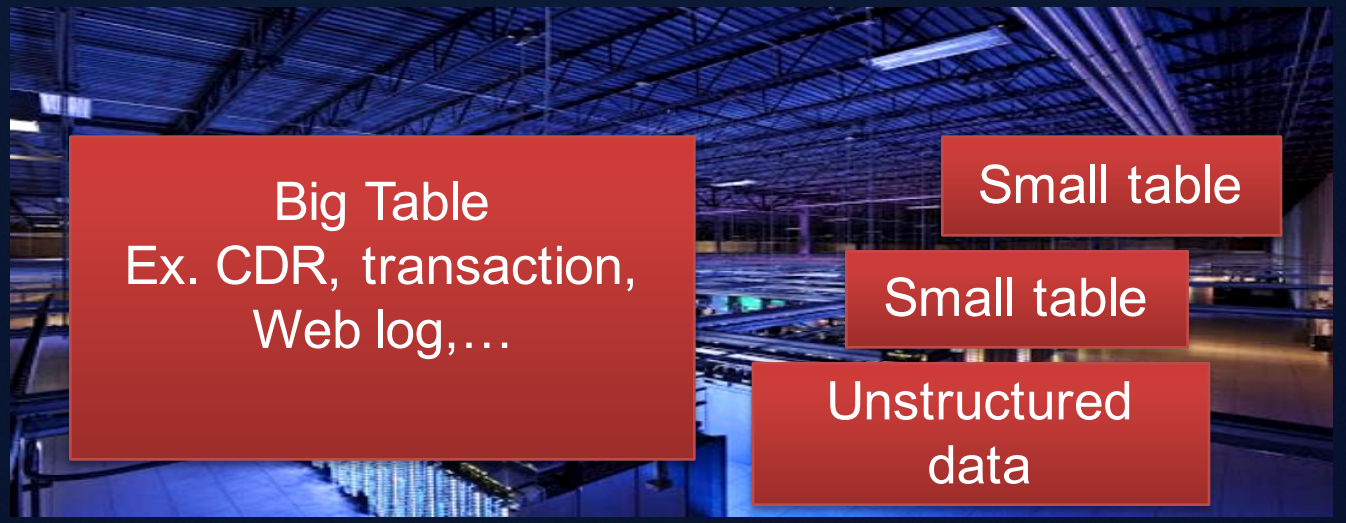
OLAP & Ad-hoc

Batch processing

Machine learning

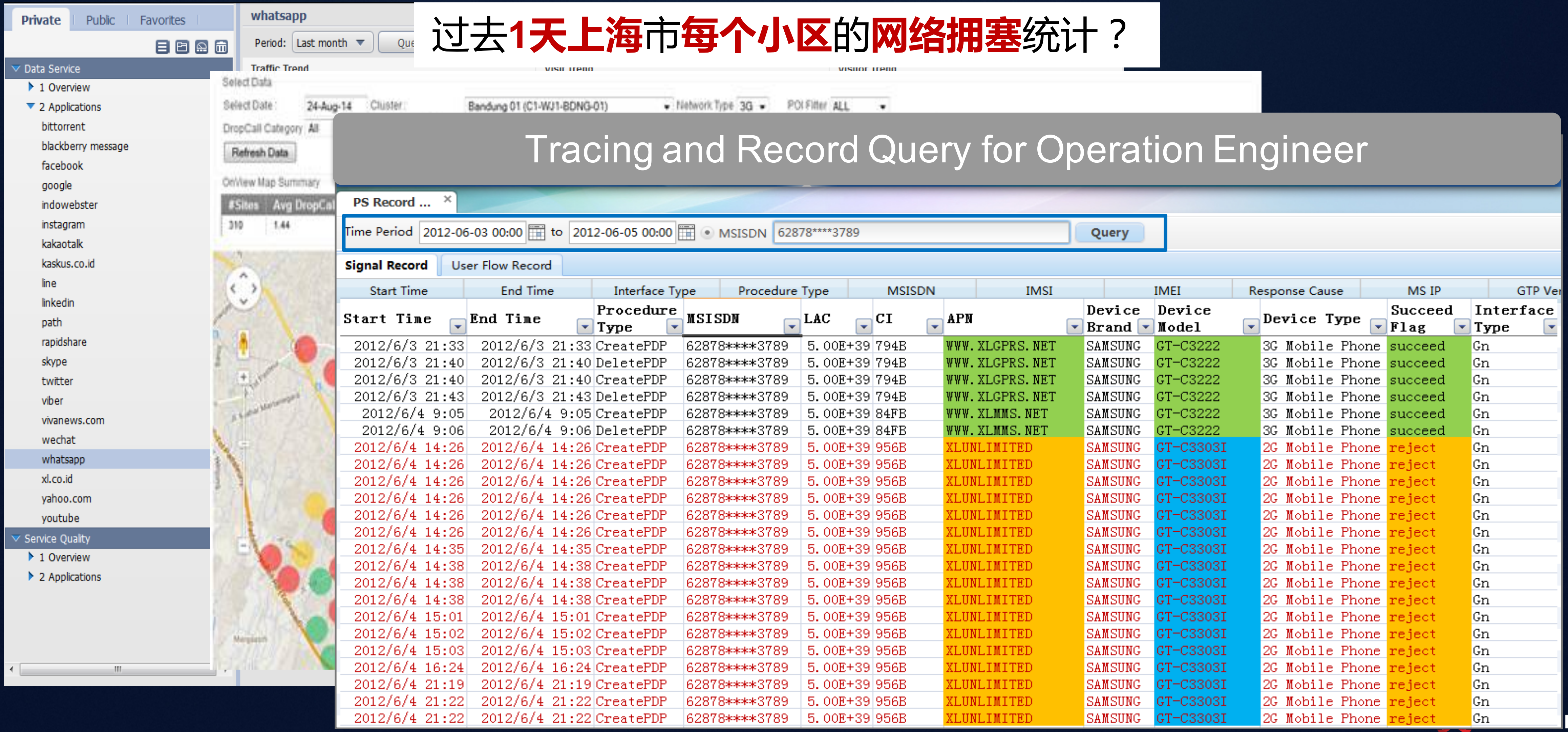
Realtime Analytics

data



数据应用举例

过去**1天**使用**Whatapp应用**的**终端按流量排名**情况？



Start Time	End Time	Procedure Type	MSISDN	LAC	CI	APN	Device Brand	Device Model	Device Type	Succeed Flag	Interface Type
2012/6/3 21:33	2012/6/3 21:33	CreatePDP	62878****3789	5.00E+39	794B	WWW.XLGPRS.NET	SAMSUNG	GT-C3222	3G Mobile Phone	succeed	Gn
2012/6/3 21:40	2012/6/3 21:40	DeletePDP	62878****3789	5.00E+39	794B	WWW.XLGPRS.NET	SAMSUNG	GT-C3222	3G Mobile Phone	succeed	Gn
2012/6/3 21:40	2012/6/3 21:40	CreatePDP	62878****3789	5.00E+39	794B	WWW.XLGPRS.NET	SAMSUNG	GT-C3222	3G Mobile Phone	succeed	Gn
2012/6/3 21:43	2012/6/3 21:43	DeletePDP	62878****3789	5.00E+39	794B	WWW.XLGPRS.NET	SAMSUNG	GT-C3222	3G Mobile Phone	succeed	Gn
2012/6/4 9:05	2012/6/4 9:05	CreatePDP	62878****3789	5.00E+39	84FB	WWW.XLMMS.NET	SAMSUNG	GT-C3222	3G Mobile Phone	succeed	Gn
2012/6/4 9:06	2012/6/4 9:06	DeletePDP	62878****3789	5.00E+39	84FB	WWW.XLMMS.NET	SAMSUNG	GT-C3222	3G Mobile Phone	succeed	Gn
2012/6/4 14:26	2012/6/4 14:26	CreatePDP	62878****3789	5.00E+39	956B	XLUNLIMITED	SAMSUNG	GT-C3303I	2G Mobile Phone	reject	Gn
2012/6/4 14:26	2012/6/4 14:26	CreatePDP	62878****3789	5.00E+39	956B	XLUNLIMITED	SAMSUNG	GT-C3303I	2G Mobile Phone	reject	Gn
2012/6/4 14:26	2012/6/4 14:26	CreatePDP	62878****3789	5.00E+39	956B	XLUNLIMITED	SAMSUNG	GT-C3303I	2G Mobile Phone	reject	Gn
2012/6/4 14:26	2012/6/4 14:26	CreatePDP	62878****3789	5.00E+39	956B	XLUNLIMITED	SAMSUNG	GT-C3303I	2G Mobile Phone	reject	Gn
2012/6/4 14:26	2012/6/4 14:26	CreatePDP	62878****3789	5.00E+39	956B	XLUNLIMITED	SAMSUNG	GT-C3303I	2G Mobile Phone	reject	Gn
2012/6/4 14:26	2012/6/4 14:26	CreatePDP	62878****3789	5.00E+39	956B	XLUNLIMITED	SAMSUNG	GT-C3303I	2G Mobile Phone	reject	Gn
2012/6/4 14:35	2012/6/4 14:35	CreatePDP	62878****3789	5.00E+39	956B	XLUNLIMITED	SAMSUNG	GT-C3303I	2G Mobile Phone	reject	Gn
2012/6/4 14:38	2012/6/4 14:38	CreatePDP	62878****3789	5.00E+39	956B	XLUNLIMITED	SAMSUNG	GT-C3303I	2G Mobile Phone	reject	Gn
2012/6/4 14:38	2012/6/4 14:38	CreatePDP	62878****3789	5.00E+39	956B	XLUNLIMITED	SAMSUNG	GT-C3303I	2G Mobile Phone	reject	Gn
2012/6/4 14:38	2012/6/4 14:38	CreatePDP	62878****3789	5.00E+39	956B	XLUNLIMITED	SAMSUNG	GT-C3303I	2G Mobile Phone	reject	Gn
2012/6/4 15:01	2012/6/4 15:01	CreatePDP	62878****3789	5.00E+39	956B	XLUNLIMITED	SAMSUNG	GT-C3303I	2G Mobile Phone	reject	Gn
2012/6/4 15:02	2012/6/4 15:02	CreatePDP	62878****3789	5.00E+39	956B	XLUNLIMITED	SAMSUNG	GT-C3303I	2G Mobile Phone	reject	Gn
2012/6/4 15:03	2012/6/4 15:03	CreatePDP	62878****3789	5.00E+39	956B	XLUNLIMITED	SAMSUNG	GT-C3303I	2G Mobile Phone	reject	Gn
2012/6/4 16:24	2012/6/4 16:24	CreatePDP	62878****3789	5.00E+39	956B	XLUNLIMITED	SAMSUNG	GT-C3303I	2G Mobile Phone	reject	Gn
2012/6/4 21:19	2012/6/4 21:19	CreatePDP	62878****3789	5.00E+39	956B	XLUNLIMITED	SAMSUNG	GT-C3303I	2G Mobile Phone	reject	Gn
2012/6/4 21:22	2012/6/4 21:22	CreatePDP	62878****3789	5.00E+39	956B	XLUNLIMITED	SAMSUNG	GT-C3303I	2G Mobile Phone	reject	Gn
2012/6/4 21:22	2012/6/4 21:22	CreatePDP	62878****3789	5.00E+39	956B	XLUNLIMITED	SAMSUNG	GT-C3303I	2G Mobile Phone	reject	Gn

来自数据的挑战

- Data Size
  - Single Table >10 B
  - Fast growing

百亿级数据量

- Multi-dimensional
  - Every record > 100 dimension
  - Add new dimension occasionally

多维度

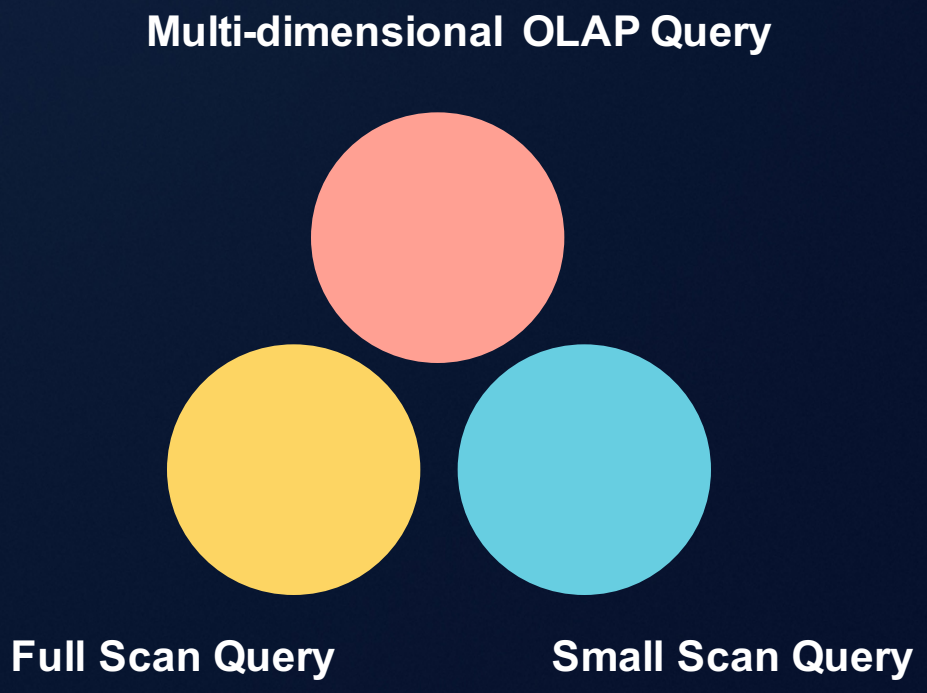
- Rich of Detail
  - Billion level high cardinality
  - 1B terminal * 200K cell * 1440 minutes = 28800 (万亿)

细粒度

来自应用的挑战

- Enterprise Integration **企业应用集成**
  - SQL 2003 Standard Syntax
  - BI integration, JDBC/ODBC

- Flexible Query **灵活查询 无固定模式**
  - Any combination of dimensions
  - OLAP Vs Detail Record
  - Full scan Vs Small scan
  - Precise search & Fuzzy search



How to choose storage?

如何构建数据平台？

选择1: NoSQL Database

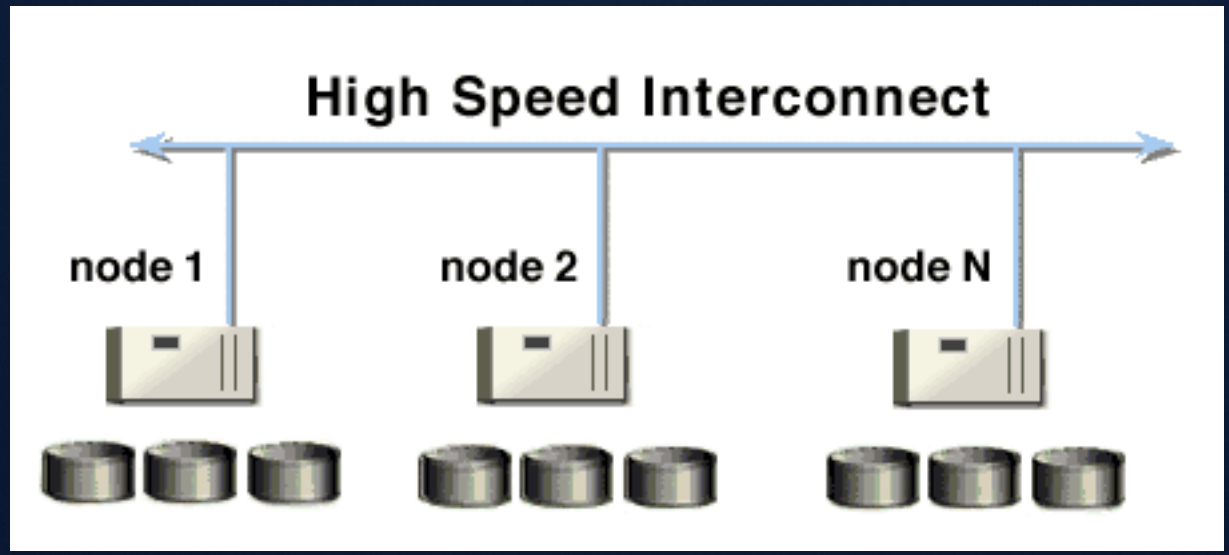
Key-Value store: low latency, <5ms

只能通过Key访问，一键一值
适合实时应用对接，不适合分析型应用

Type	Examples
Key-Value Store	redis riak
Wide Column Store	APACHE HBASE Cassandra

Row Key	Timestamp	Customer		Sales	
Customer Id		Name	City	Product	Amount
101	T1	Suresh	Hyderabad		300
101	T2	Suresh Reddy		Books	
102	T1	Lavya Gavshinde	Indore	Fan	600
102	T2	Lavya			570
102	T3		Bhopal		
103	T1	Anurag	Raipur	Laptop	40000
104	T1	Deepesh	Delhi	Bike	32000

选择2：Parallel database



- Parallel scan + Fast compute

细粒度控制并行计算，适合中小规模数据分析（数据集市）

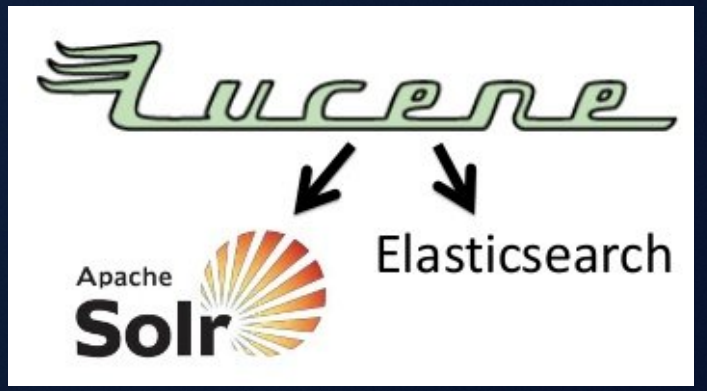
- Questionable scalability and fault-tolerance
  - Cluster size < 100 data node
  - Not suitable for big batch job

扩展能力有上限
查询内容错能力弱
不适合海量数据分析（企业级数仓）

选择3: Search engine

- All column indexed
- Fast searching
- Simple aggregation

适合多条件过滤，文本分析



- Designed for search but not OLAP
- Not for TopN, join, multi-level aggregation

无法完成复杂计算

- 3~4X data expansion in size

数据膨胀

- No SQL support

专用语法，难以迁移

选择4: SQL on Hadoop

- Modern distributed architecture, scale well in computation.
  - Pipeline based: Impala, Drill, Flink, …
  - BSP based: Hive, SparkSQL

并行扫描+并行计算
适合海量数据计算

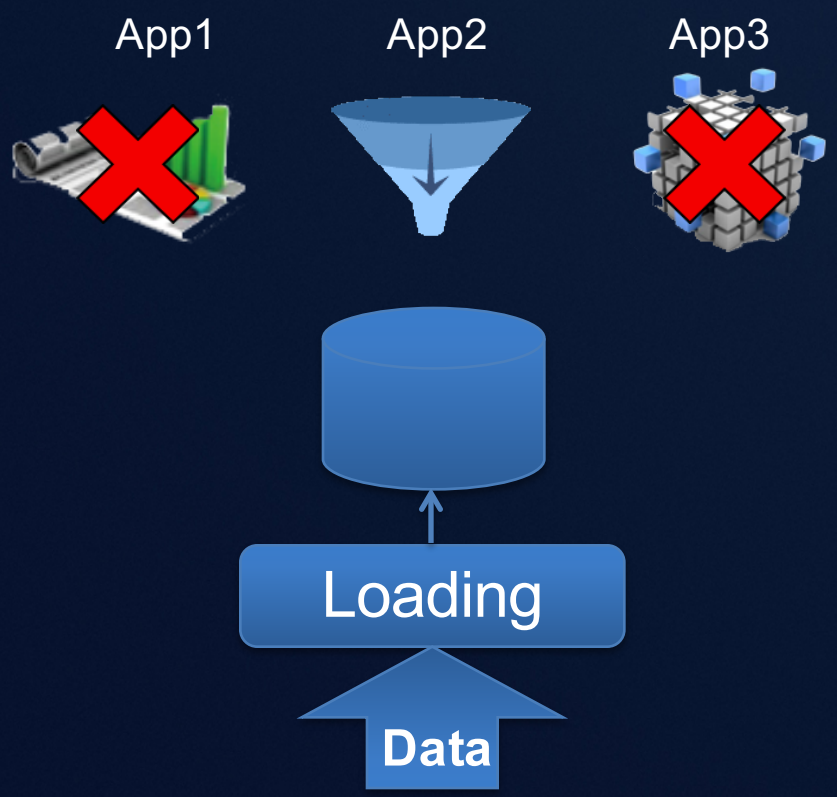
- BUT, still using file format designed for batch job
  - Focus on scan only
  - No index support, not suitable for point or small scan queries

仍然使用为批处理设计的存储，场景受限

架构师如何选择？

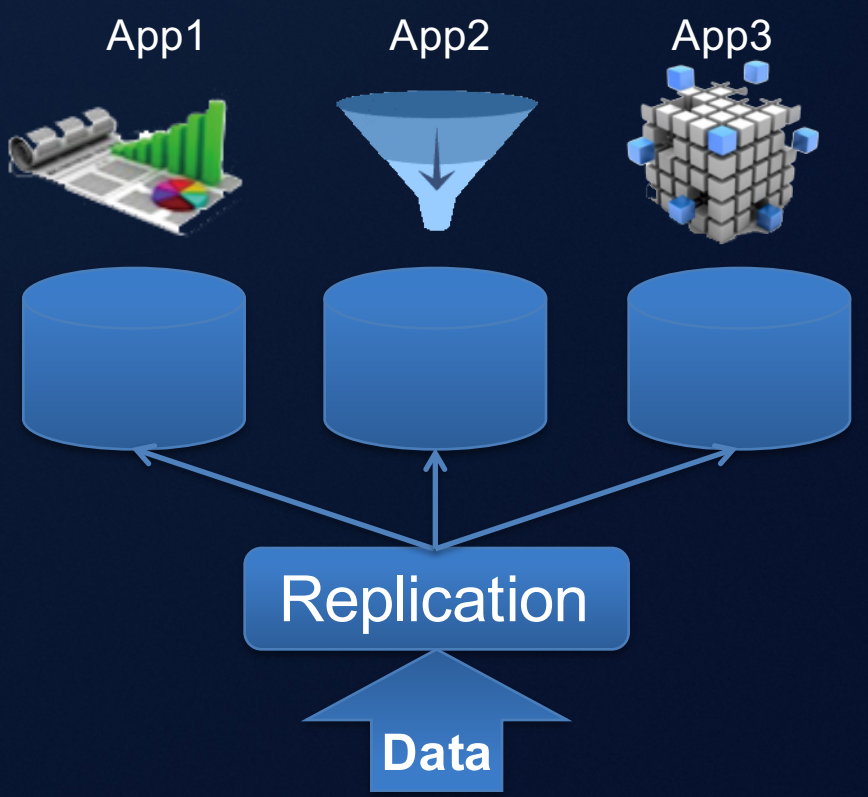
Choice 1: Compromising

做出妥协，只满足部分应用



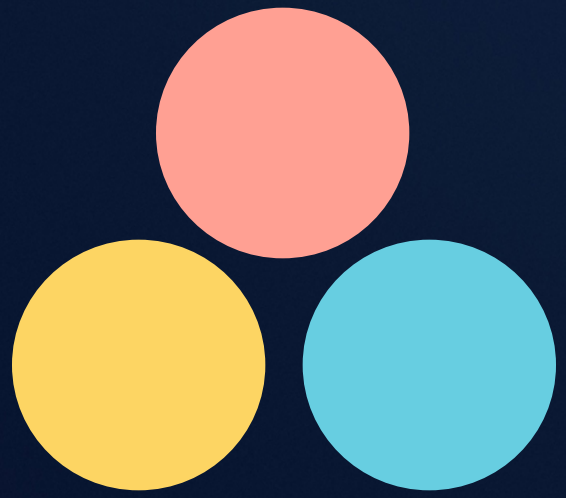
Choice 2: Replicating of data

复制多份数据，满足所有应用



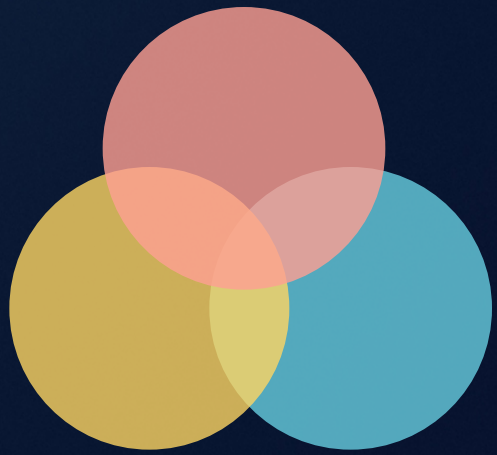
CarbonData目标：
一份数据满足多种业务需求，与大数据生态无缝集成

Multi-dimensional OLAP Query



Full Scan Query　　Small Scan Query

CarbonData: Unified Storage



一份数据满足多种分析场景
详单过滤，海量数仓，数据集市，…

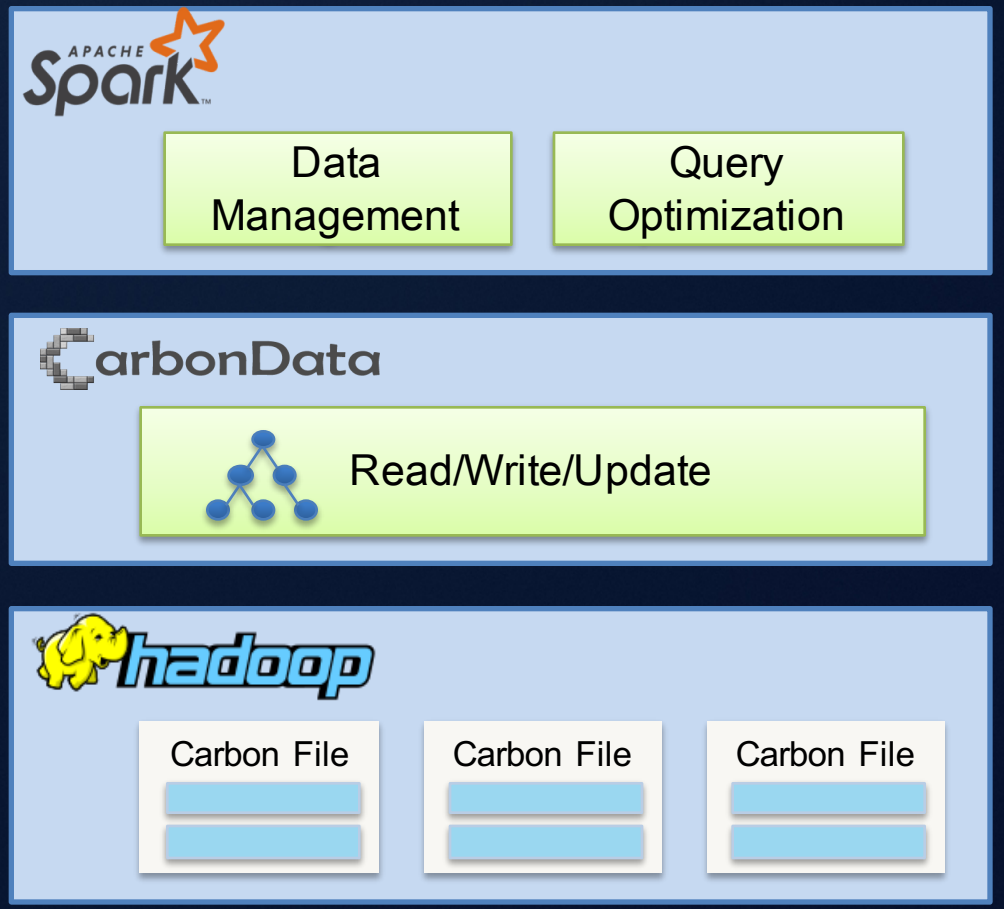
CarbonData原理介绍
打造大数据交互式分析引擎

CarbonData大数据生态

- 支持Spark、Hive、Presto、Flink
- 内置Hadoop和Spark深度优化
  - Hadoop: > 2.2
  - Spark 1.5, 1.6, 2.1

- 接口
  - SQL
  - DataFrame API

- 支持操作:
  - 查询：支持SparkSQL优化器
  - 数据管理：批量入库、更新、删除、合并（Compaction）、增删列



使用方式：入库

- SQL

```
CREATE TABLE tablename (name String, PhoneNumber String)
STORED BY “carbondata”
TBLPROPERTIES (...)

LOAD DATA [LOCAL] INPATH 'folder path' [OVERWRITE] INTO TABLE tablename
OPTIONS(...)

INSERT INTO TABLE tablennme  select_statement1 FROM table1;
```

- Dataframe

```
df.write
   .format(“carbondata")
   .options("tableName“, “t1"))
   .mode(SaveMode.Overwrite)
   .save()
```

使用方式：查询

- SQL

```
SELECT project_list FROM t1
WHERE cond_list
GROUP BY columns
ORDER BY columns
```

- Dataframe

```
df = sparkSession.read
          .format("carbondata")
          .option("tableName", "t1")
          .load("path_to_carbon_file")

df.select(…).show
```

使用方式：更新和删除

Modify one column in table1

```
UPDATE table1 A
SET A.REVENUE = A.REVENUE - 10
WHERE A.PRODUCT = 'phone'
```

phone, ~~70~~ 60
car,100
phone, ~~30~~ 20

Modify two columns in table1 with values from table2

```
UPDATE table1 A
SET (A.PRODUCT, A.REVENUE) =
     (
      SELECT PRODUCT, REVENUE
      FROM table2 B
      WHERE B.CITY = A.CITY AND B.BROKER = A.BROKER
     )
WHERE A.DATE BETWEEN '2017-01-01' AND '2017-01-31'
```

Delete records in table1

```
DELETE FROM table1 A
WHERE A.CUSTOMERID = '123'
```

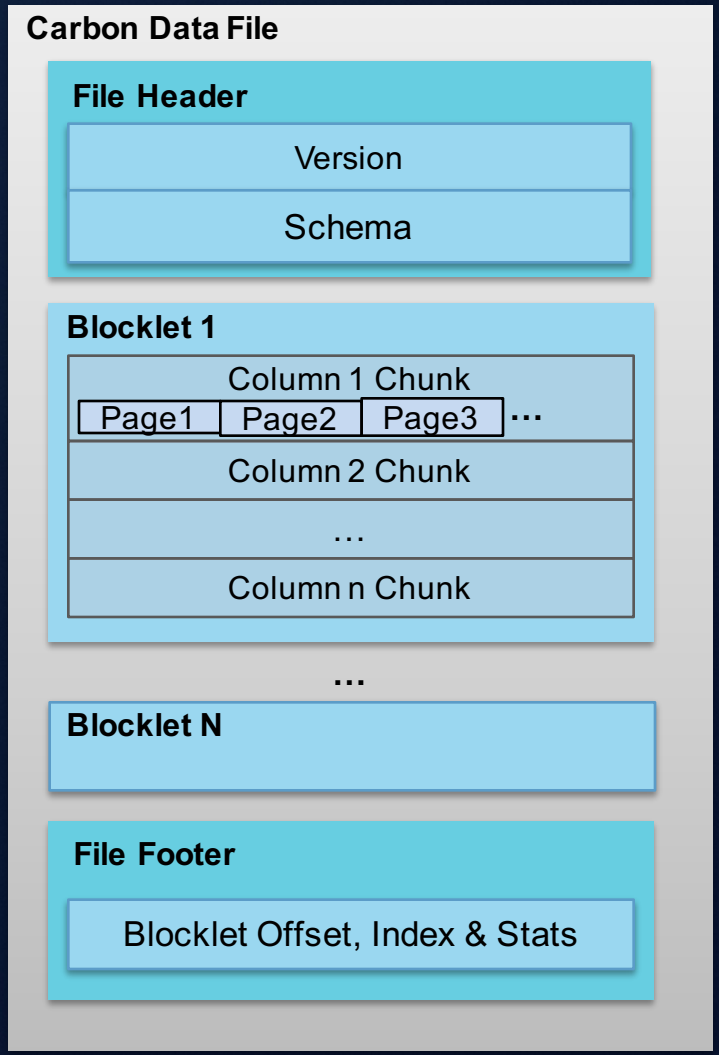
~~123, abc~~
456, jkd

CarbonData原理介绍

- 数据组织
- 索引
- 扫描过程

CarbonData文件格式

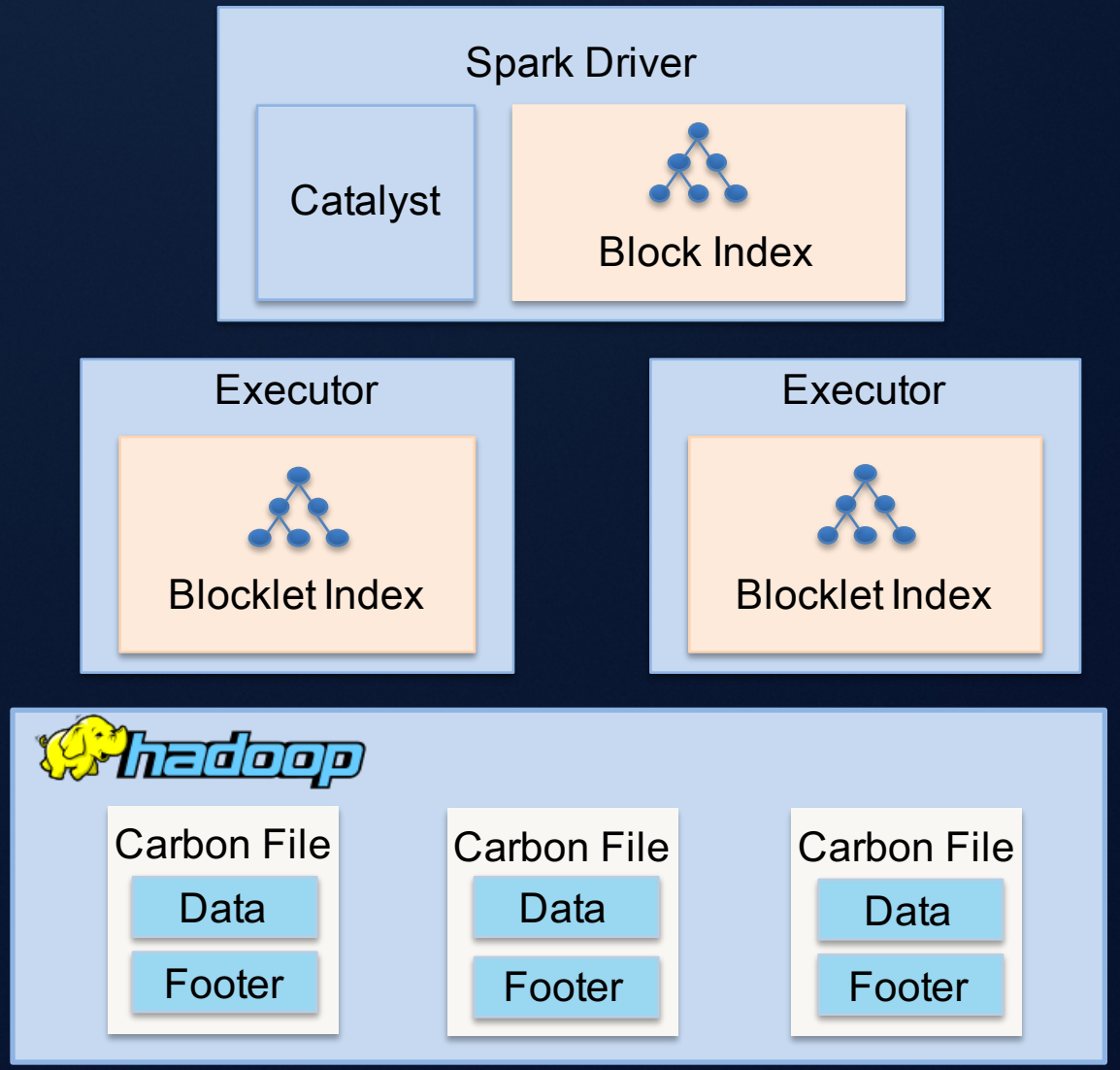
- 数据布局
  - Block：一个HDFS文件，256MB
  - Blocklet：文件内的列存数据块，是最小的IO读取单元
    - Column chunk: Blocklet内的列数据
    - Page：Column chunk内的数据页，是最小的解码单元
- 元数据信息
  - Header：Version，Schema
  - Footer： Blocklet Offset, Index & 文件级统计信息
- 内置索引和统计信息
  - Blocklet索引：B Tree startkey, endkey
  - Blocklet级和Page级统计信息：min，max等



利用两级索引架构减少Spark Task数和磁盘IO

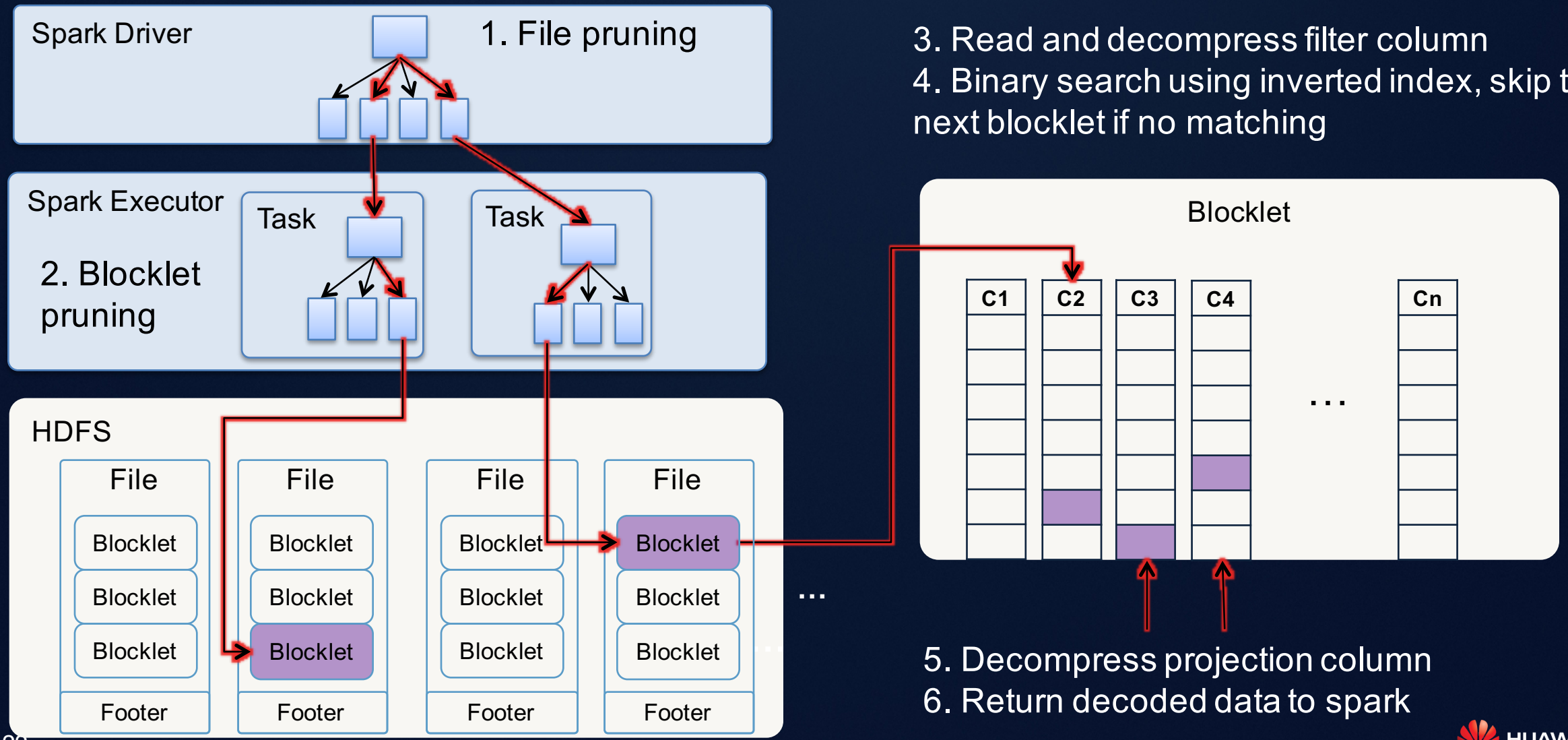
- 第一级：文件级索引
用于过滤文件（HDFS Block），避免扫描不必要的文件，减少多达95%的Spark Task

- 第二级：Blocklet索引
用于过滤文件内部的Blocklet，避免扫描不必要的Blocklet，减少磁盘IO

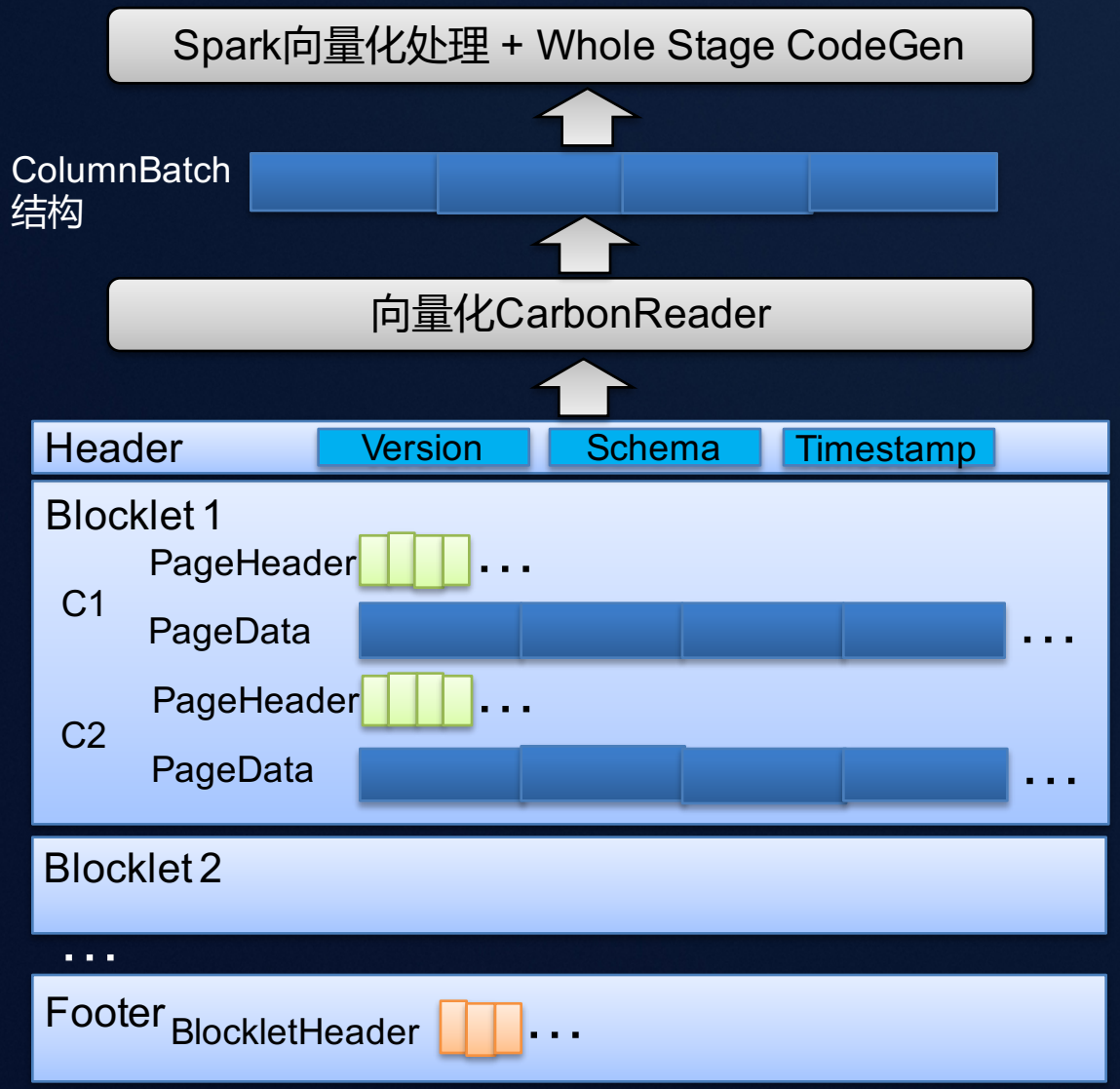


一个完整的过滤查询过程

```
SELECT c3, c4 FROM t1 WHERE c2='beijing'
```



文件内扫描优化，与Spark向量化处理对接



大颗粒IO单元（Carbon V3格式）：

•Blocklet内部一个Column Chunk 是一个IO单元，Blocklet按大小切分，默认64MB，大概有100万行记录。大颗粒顺序读提升扫描性能。

跳跃解码（Carbon V3格式）：

•增加数据页Page概念，按Page进行过滤和解码，减少不必要的解码解压缩，提升CPU利用率

向量化解码 + Spark向量化处理

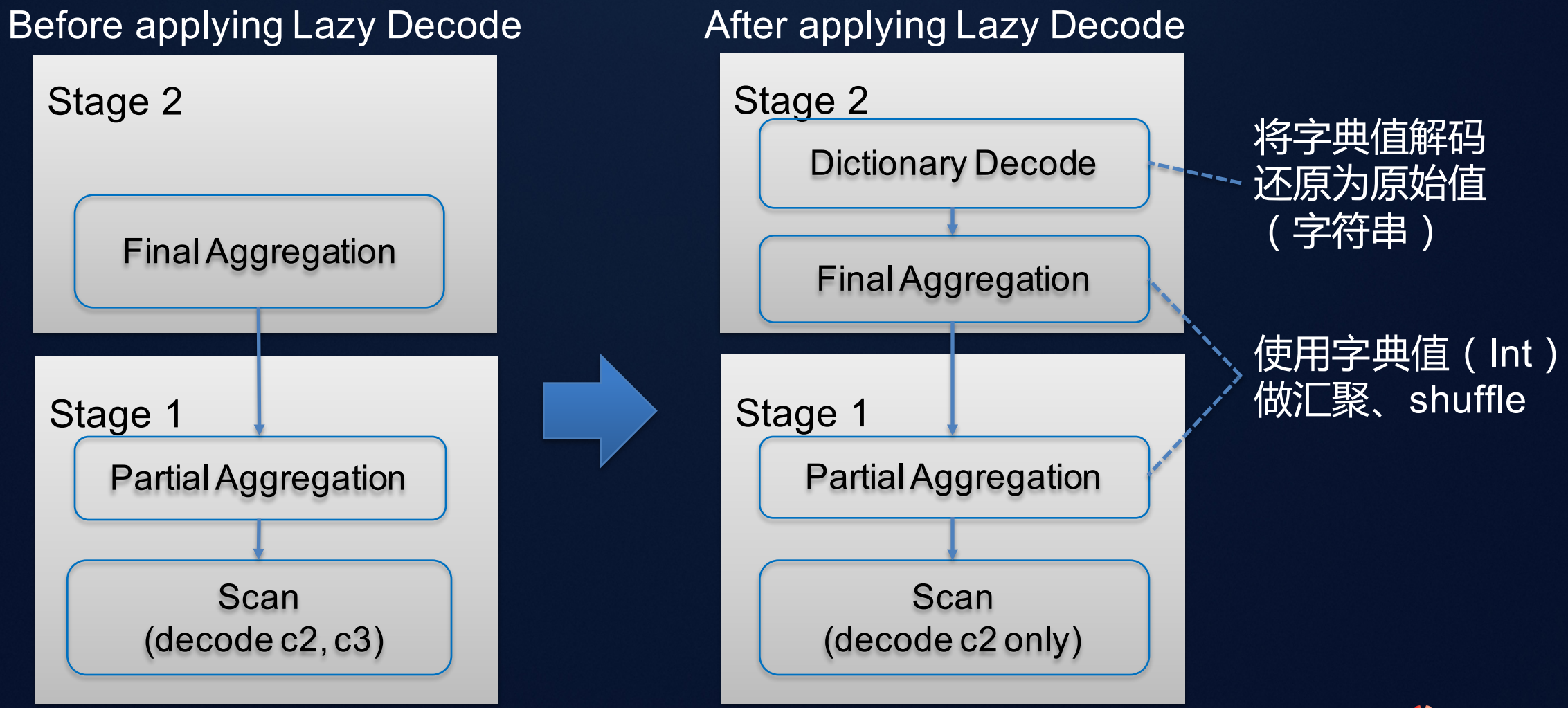
•解码和解压缩采用向量化处理，与Spark2.1向量化、Codegen结合，在扫描+聚合场景下提升性能4X

堆外内存：

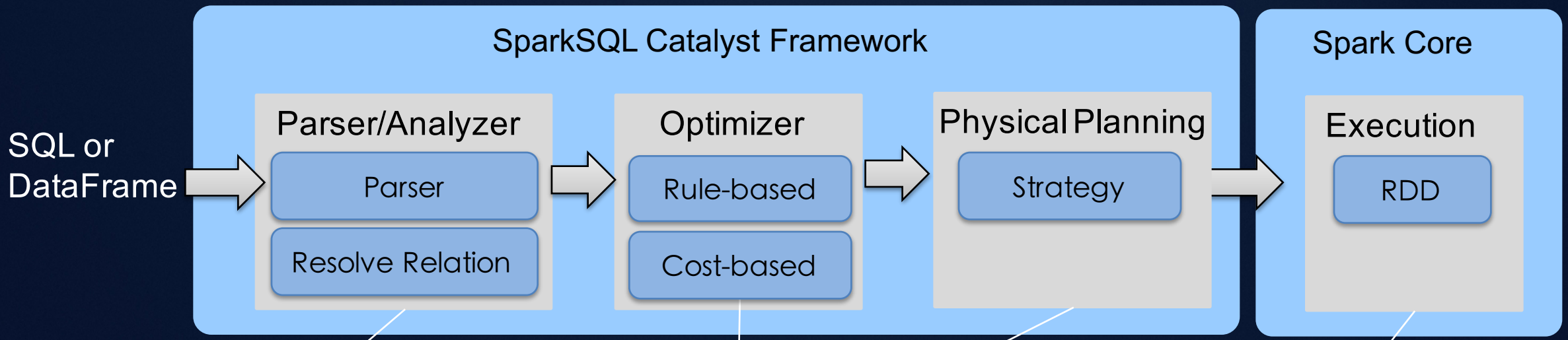
•处理大结果集时，解码解压缩过程中堆外完成，减少GC

利用全局字典做延迟解码，提升GroupBy性能

```
SELECT c3, sum(c2) FROM t1 GROUP BY c3
```



SparkSQL深度对接



Carbon Data Source

SQL语法
- 数据入库、更新、删除
- Compaction

Carbon优化策略:
- 使用全局字典做延迟解码
- 和CBO联合优化（规划）

Carbon相关的RDD和Command:
- 查询：利用CarbonTableInputFormat中的索引、过滤下压
- 入库、更新、删除、Compaction、分区等

总结：CarbonData的优秀DNA

查询：

- 两级索引，减少IO：**适合ad-hoc查询**，任意维度组合查询场景
- 延迟解码，向量化处理：适合全表扫描、**汇总分析**场景

数据管理：

- 增量入库，多级排序可调：**由用户权衡**入库时间和查询性能
- 增量更新，批量合并：支持**快速更新**事实表或维表，闲时做Compaction合并

大规模：

- 计算与存储分离：支持从GB到PB大规模数据，**万亿数据秒级响应**

部署：

- Hadoop Native格式：与大数据生态无缝集成，**利用已有Hadoop集群资产**

应用场景和调优介绍

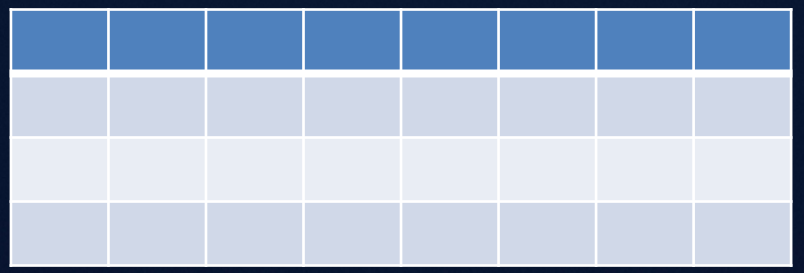
应用场景

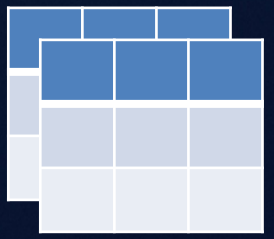
- 场景1：详单分析
- 场景2：数仓BI分析

应用场景1：详单分析

业务：详单分析（明细数据ad-hoc查询），例如：事件日志分析，用户行为分析，流水表查询
业务诉求：

- 入库：详单来得快，要快速入库，>10万记录每秒每节点
- 存储：单表数据量大，存储N年数据，压缩率越高越好，存储成本压力大
- 查询：
  - 多条件灵活过滤，一般提供界面让用户选择条件
  - 精确匹配，模糊匹配
  - 事实表Join维度表，例如替换用户ID
  - 最终输出：明细数据或对结果轻度汇总





详单分析的调优手段

- 过滤查询要快：
  - 建表时选择合适的索引列，排序顺序，Blocklet大小，目标是提升Pruning效率，减少IO
- 入库要快：
  - 选择LOCAL_SORT，入库时建立索引但避免做shuffle
  - 不做全局字典，避免生成字典的扫描过程
- 压缩要高：
  - 使用自适应Encoding，自动选择最优编码方式

索引调优

CREATE TABLE t1(…)

TBLPROPERTIES (‘SORT_COLUMNS’=‘…’, ‘SORT_SCOPE’=‘…’)

- SORT_COLUMNS:
  - 找到查询中频繁使用的过滤条件列
  - 将他们以使用频度的顺序放入SORT_COLUMNS中（越靠前过滤效果越好）。例如，经常使用时间和用户ID做过滤，SORT_COLUMNS=‘TIME, USER_ID, CITY’
- SORT_SCOPE:
  - 根据业务对入库速度的要求，选择使用LOCAL_SORT或GLOBAL_SORT。
  - 如果业务对某个维度的点查并发性能要求较高，推荐使用GLOBAL_SORT。否则建议使用LOCAL_SORT。（系统默认为LOCAL_SORT，兼顾入库性能和查询性能）

过滤性能测试

过滤查询，c1到c10组合条件过滤。
Carbon的SORT_COLUMNS=从c1到c10。Partition的分区列是c1

	Filter								响应时间(sec)		Task数	
Query	c1	c2	c3	c4	c5	c6	…	c10	Parquet	CarbonData	Parquet	CarbonData
Q1	✓	✓	✓	✓					6.4	1.3	55	5
Q2		✓	✓						65	1.3	804	5
Q3		✓		✓					71	5.2	804	9
Q5		✓			✓				64	4.7	804	9
Q4			✓	✓					67	2.7	804	161
Q6			✓			✓			62	3.7	804	161
Q7				✓		✓			63	21.85	804	588
Q8								✓	69	11.2	804	645

Carbon通过利用索引，减少Task任务下发，同时单Task内减少磁盘IO

过滤粒度调优

carbon.properties的设置：

- carbon.blockletgroup.size.in.mb（一个blocklet的大小）
  - 该值决定Carbon读文件的最小IO粒度，也决定了Carbon在文件内能跳过的数据块大小
  - 该值越小，精确点查询场景的IO越小，查询越快。
  - 该值越大，IO效率越高，越有利于全表扫描场景。
- carbon.custom.block.distribution（blocklet调度）
  - 当为true时，carbon会按blocklet返回split给Spark，即会按blocklet进行task下发。例如命中了5个blocklet则下发5个task。
  - 当命中数据较少时，CPU有剩余，打开此选择可以提升读并行度

Encoding调优

- 低基数String列
  - 默认使用UTF8字符串编码
  - 用户可在创表时使用全局字典编码，提升压缩率。（代价是入库时需要更多内存）

TBLPROPERTIES('DICTIONARY_INCLUDE' = 'city, country')

- 度量列（byte, short, int, long, double, decimal）
  - 默认使用AdaptiveEncoding
  - 系统会对每个Page根据统计信息自动选择最优的存储类型。如long的数据可能会用byte存储。

- Date/TimeStamp
  - 默认使用AdaptiveEncoding
  - 系统会对每个Page根据统计信息自动选择最优的存储类型。

自适应编码

C1: long
0x00000002
0x00000007
0x00000009
0x00000003
0x00000005
0x00000007

Min: 2
Max: 9

Schema：long（4字节）
实际使用：byte（1字节）
AdaptiveIntegralEncoding

C2: long
12892712
12892727
12892713
12892743
12892725
12892742

Min: 12892712
Max: 12892743

Schema：long（4字节）
实际使用：byte（1字节）
AdaptiveDeltaEncoding
(存储值为max-value)

C3: double
12.32
21.42
32.12
42.43
54.32
14.32

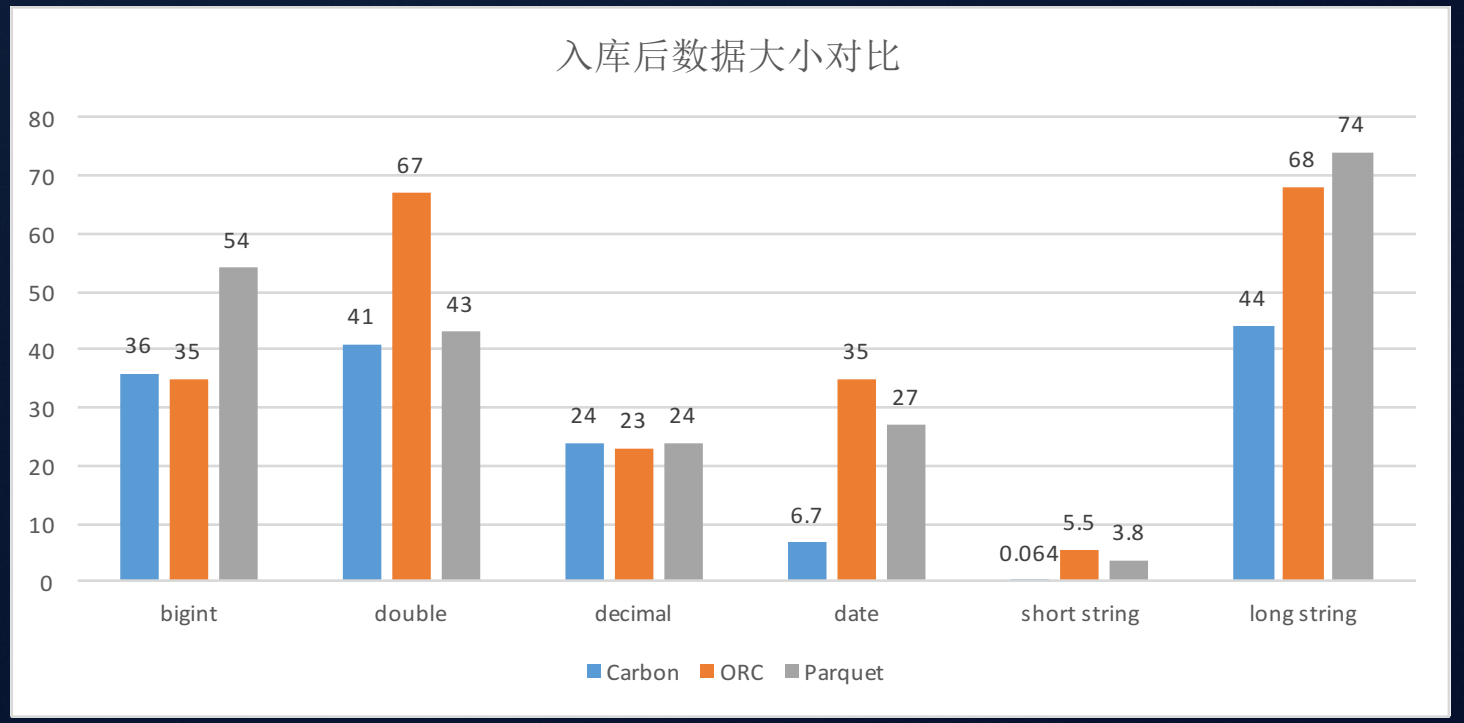
Min: 12.32
Max: 54.32
Scale: 2

Schema：double（8字节）
实际使用：short（2字节）
AdaptiveFloatingEncoding
(存储值为value*Math.pow(10, scale)

压缩率对比

TPC-H LINEITEM事实表，1GB数据，对比压缩率

Data Type	Carbon	ORC	Parquet
bigint	36	35	54
double	41	67	43
decimal	24	23	24
date	6.7	35	27
short string	0.064	5.5	3.8
long string	44	68	74

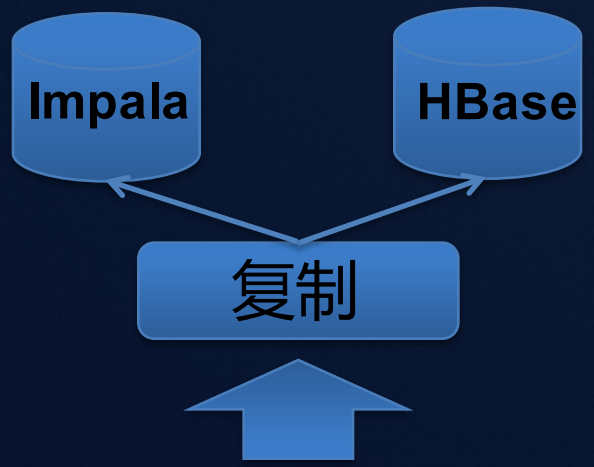


电信、金融详单分析：提升查询性能和扩展性

优化前

Impala：
1.**只适合Join/Orderby的业务**，多维过滤性能不足；
2.**扩展性不足**，难以扩展上百节点、无法支撑海量数据查询

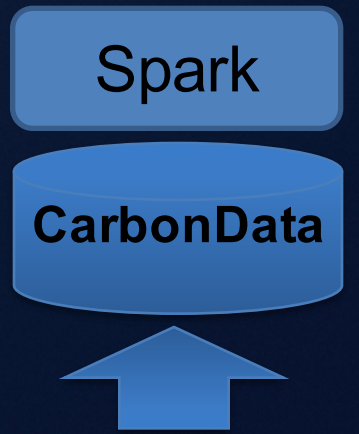
HBase：
1、对6个关键维度查询，**需要建立4个二级索引，数据膨胀太大**，数据同步复杂
2、无法用YARN统一资源管理



电信详单、交易详单

优化后

1. 一份数据完成复杂分析和多维点查，简化存储
2. 支持万亿数据量分析，秒级响应，高扩展性



电信详单、交易详单

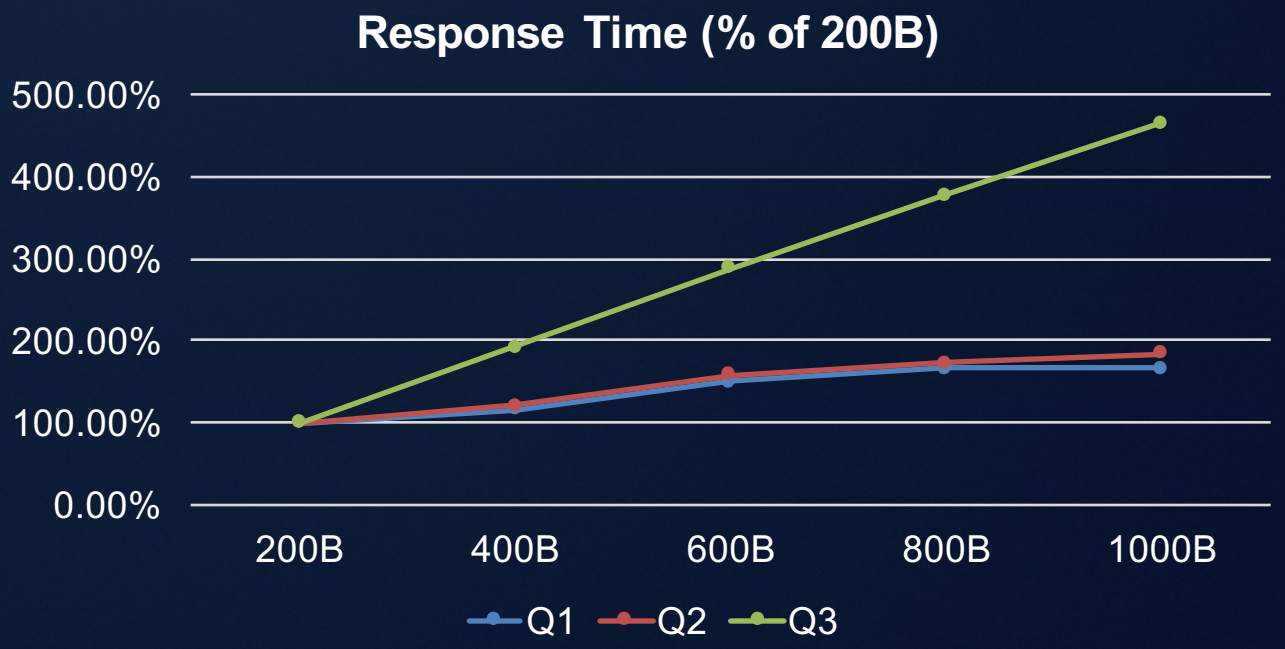
大数据集Scale out测试

数据：2000亿到1万亿数据 (中国某省份半年数据)

集群: 70 nodes, 1120 cores

查询:

Q1: filter (c1~c4), select *
Q2: filter (c10), select *
Q3: full scan aggregate



结果

- 过滤查询：5倍数据量，时间增加<1倍。
- 汇总分析：有效利用计算资源，可线性扩展

详单分析（扩展场景）

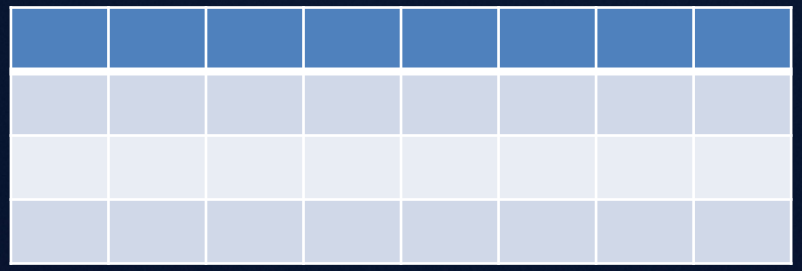
- 实时详单分析
  - 分析1分钟前的数据，实时定位问题（流式入库）
- 日志分析
  - 快速提取日志内容，如针对本文字段或JSON嵌套格式（Schema推断）
  - 对结构化字段、文本字段快速搜索（分词、文本索引）
- 时间序列分析
  - 利用数据特征做深度压缩
  - 按时间的查询优化：高并发的点查询（Partition）、汇聚查询（Cube）
  - 按时间老化数据

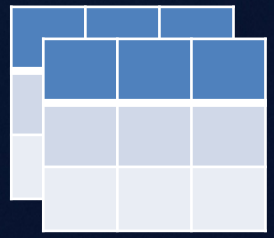
应用场景2：数仓BI分析

业务：传统数仓BI类业务，例如：每日输出报表，经营分析等
业务诉求：

- 入库：每晚入库一批，入库速度要求不高
- 存储：表格多，表间关系复杂；入库都是规整后的数据，压缩率越高越好
- 查询：
  - 多条件灵活过滤，一般可事先确定需要分析的列
  - 事实表Join维度表，可能存在事实表Join事实表场景（大大表Join）
  - 全表扫描+深度汇总统计，多层子查询
  - 最终输出：Dashboard展现，一次业务展现需要几十或上百次并发查询

每日入库





数仓分析的调优手段

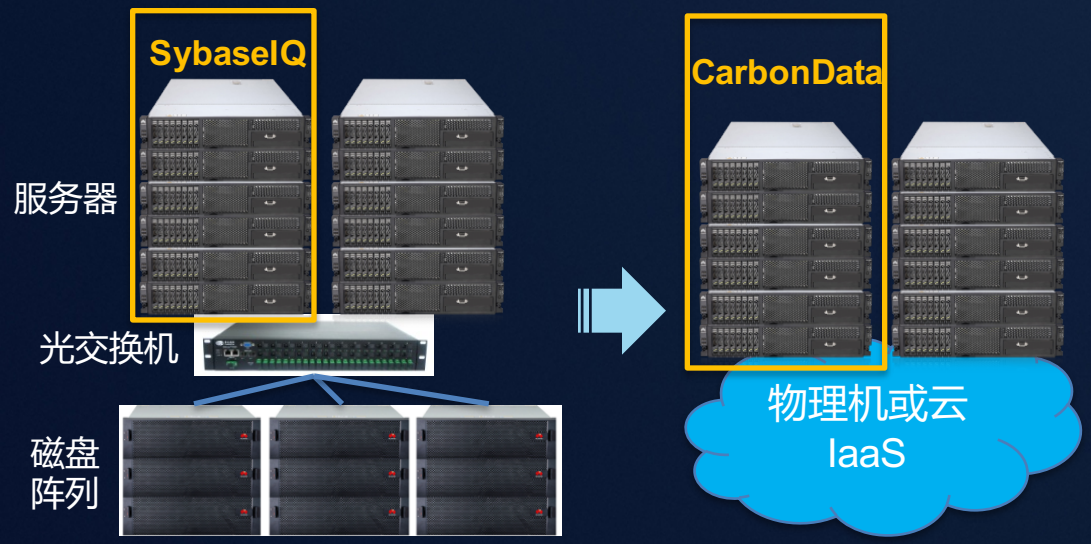
- 入库要快：
  - 对比详单分析场景，没有更高要求，使用相同的调优方案即可。
- 扫描+汇聚统计要快：
  - 存储：
    - 选择合适的SORT_COLUMNS顺序，提升压缩率，减少读取数据量
    - 利用分区做Partition Pruning，例如针对天级报表可使用时间range分区
    - 对Groupby列使用全局字典编码，避免使用String做计算，并减少Shuffle数据量
    - 通过Compaction操作，合并小文件，提升扫描效率
  - 计算：
    - 使用Carbon向量化Reader，与Spark向量化处理相结合
    - 由于扫描数据量大，使用Unsafe堆外内存，减少GC时间
    - 调整spark.sql.shuffle.partitions，经验值为节点数*2
- Join优化
  - 对需要Join的两个表格，在Join key上创建bucket，可避免Join时的shuffle

扫描调优

CREATE TABLE t1(…)
TBLPROPERTIES (‘DICTIONARY_INCLUDE’=‘…’)

- DICTIONARY_INCLUDE:
  - 对GroupBy列使用全局字典编码，避免使用String做计算
  - 注意：不要对高cardinality列做字典编码，否则可能导致内存需求太大，入库会OOM
- 向量化读
  - 设置carbon.enable.vector.reader为true
- 使用Unsafe堆外内存
  - 设置enable.unsafe.in.query.processing为true
- **Compaction**
  - 定期手动触发Compaction命令，合并相邻的Segment。
  - 设置carbon.major.compaction.size 并使用 ALTER TABLE t1 COMPACT ‘MAJOR’

电信：数仓分析场景，替换SybaseIQ和磁盘阵列



典型业务场景优化对比（忙时性能对比）：

业务	SQL特征	优化前	优化后	提升
业务1	多过滤（IO型）	10~20S	3S	5倍
业务2	高维计数TopN（全表扫描+用户数统计）	15S	7S	3倍
业务3	多过滤（IO型）	20S-30S	3S	8倍
业务3	高维聚合（千万级汇聚+全表扫描）	超时	20~30S	--

Carbon忙闲时性能对比（闲时并发15，忙时并发50）：

业务	闲时	忙时
业务1	2S	3~5S
业务2	10S	20~30S
业务3	3S	4~5S

原方案

- **成本高**：需要部署光交换机、磁盘阵列
- **无法云化**：数据存储在磁阵上，无法支持容器、VM部署
- **查询性能差**：在忙时查询性能无法满足要求，需要扩容磁阵

新方案

- **使用Carbon替换**：整个产品融合为一个HDFS存储，不依赖磁阵。
- **性能提升**：Carbon索引提升IO效率和分布式计算效率，**提升整体性能3~8倍**
- **规模**：迁移15个业务，1000+张表；每15分钟、1小时、1天加载一次数据

数仓分析（扩展场景）

- 数据集市分析
  - KPI固定，分析主题明确。
  - 性能要求高，实现秒级响应，高交互性
  - 在获得KPI数据后，希望实现二级分析，通过下钻、上卷、钻透获得更详细信息，如详单分析场景。
- 规划：
  - 建立Cube多维模型或预汇聚表
  - 与查询引擎结合，实现自动优化

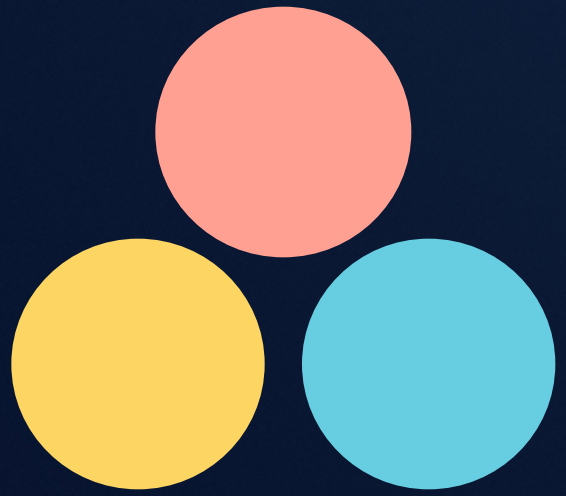
调优选项汇总

	手段	参数
存储：索引	调整索引列，索引顺序，提升过滤性能	SORT_COLUMNS, SORT_SCOPE
存储：过滤粒度	影响过滤性能	carbon.blockletgroup.size.in.mb TABLE_BLOCKSIZE
存储：分区	分区和分桶设置，影响并发性能和join性能	PARTITION, BUCKET
存储：编码	哪些列做全局字典，提升groupby性能	DICTIONARY_INCLUDE
查询	通过调整索引、字典和Reader内存选项，提升扫描速度，减少GC	carbon.enable.vector.reader enable.unsafe.in.query.processing spark.sql.shuffle.partitions
数据管理：入库	设置入库使用的核数，充分利用CPU 设置多临时目录，提升并发写能力	carbon.number.of.cores.while.loading carbon.use.multiple.temp.dir enable.unsafe.sort
数据管理：Compaction	设置compaction的策略，影响文件大小和索引的有效性，进而提升查询性能	carbon.enable.auto.load.merge carbon.compaction.level.threshold carbon.major.compaction.size

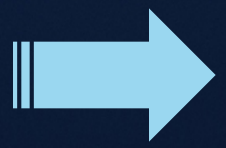
What's Next
下一步计划

CarbonData目标：
一份数据满足多种业务需求，与大数据生态无缝集成

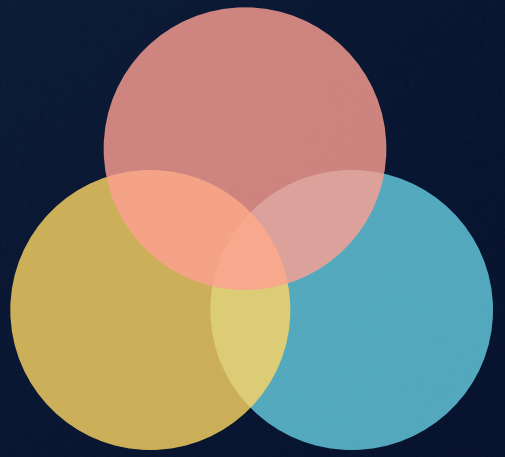
Multi-dimensional OLAP Query



Full Scan Query　　Small Scan Query



CarbonData: Unified Storage

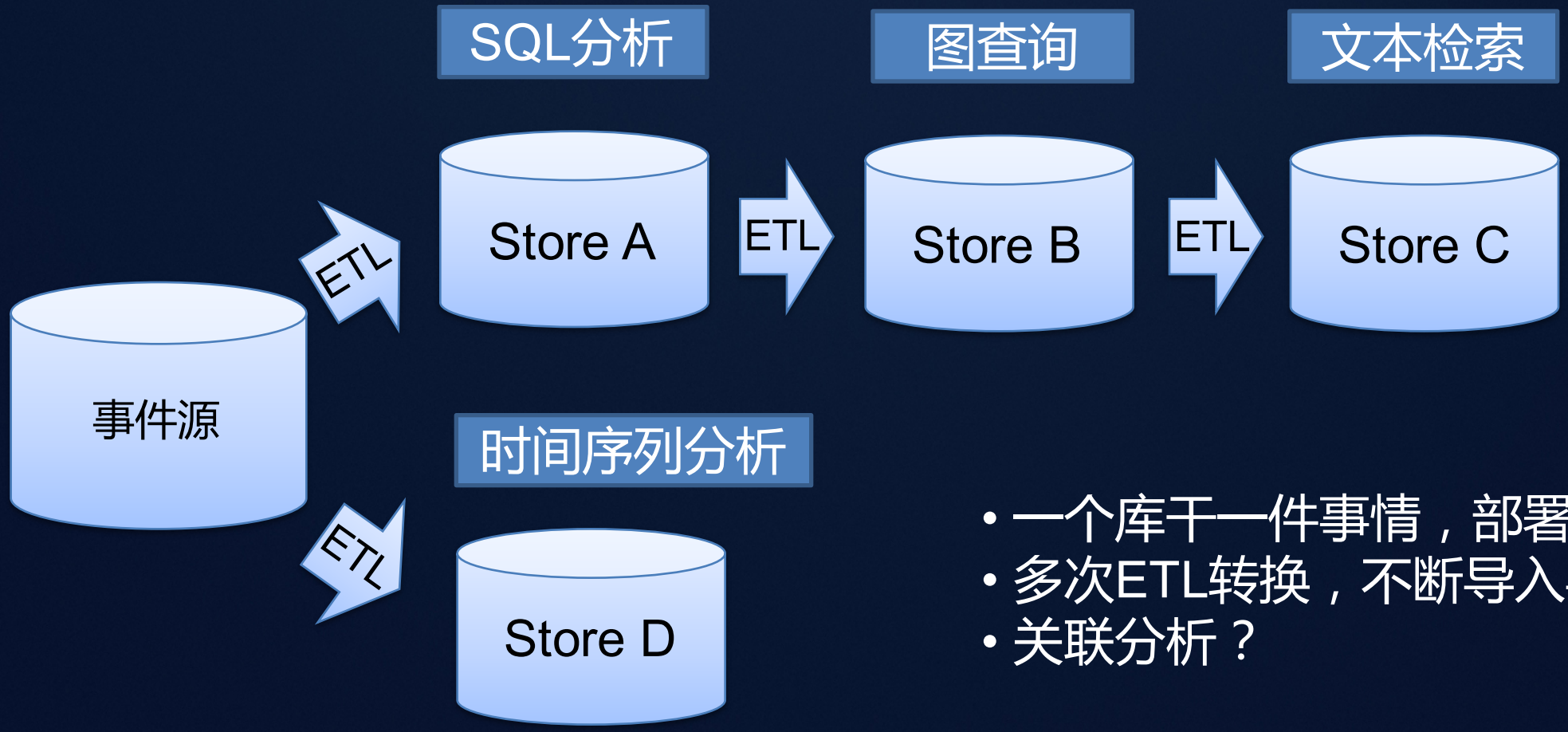


一份数据满足多种分析场景
详单过滤，海量数仓，数据集市，…

大数据时代的数据处理和分析至少还有

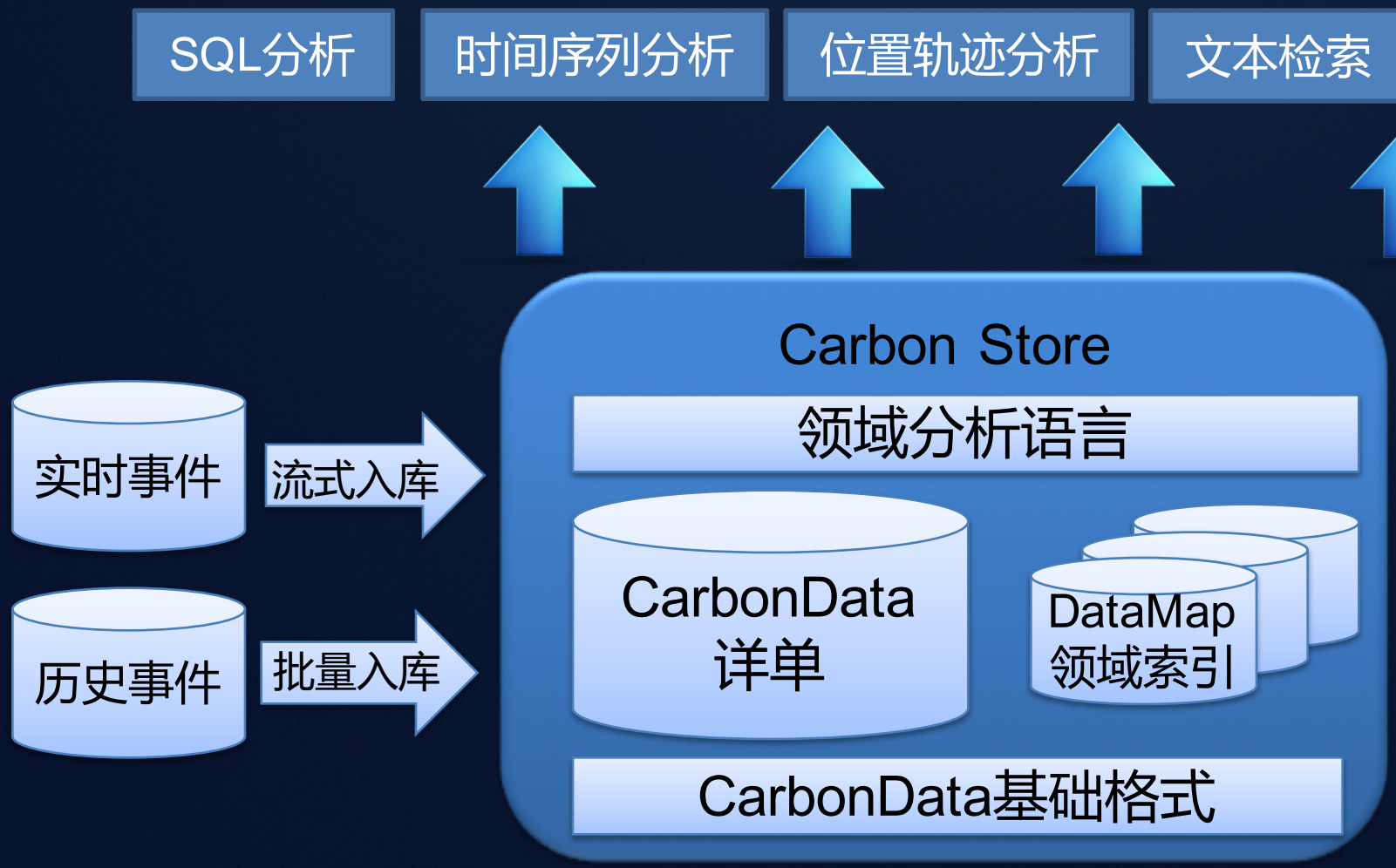
- 时间序列分析
- 图分析
- 空间位置分析
- 文本检索
- 。。。

当前的解决方案



- 一个库干一件事情，部署多个集群
- 多次ETL转换，不断导入导出
- 关联分析？

CarbonData 2.0目标



CarbonStore: 打造多业务统一数据存储

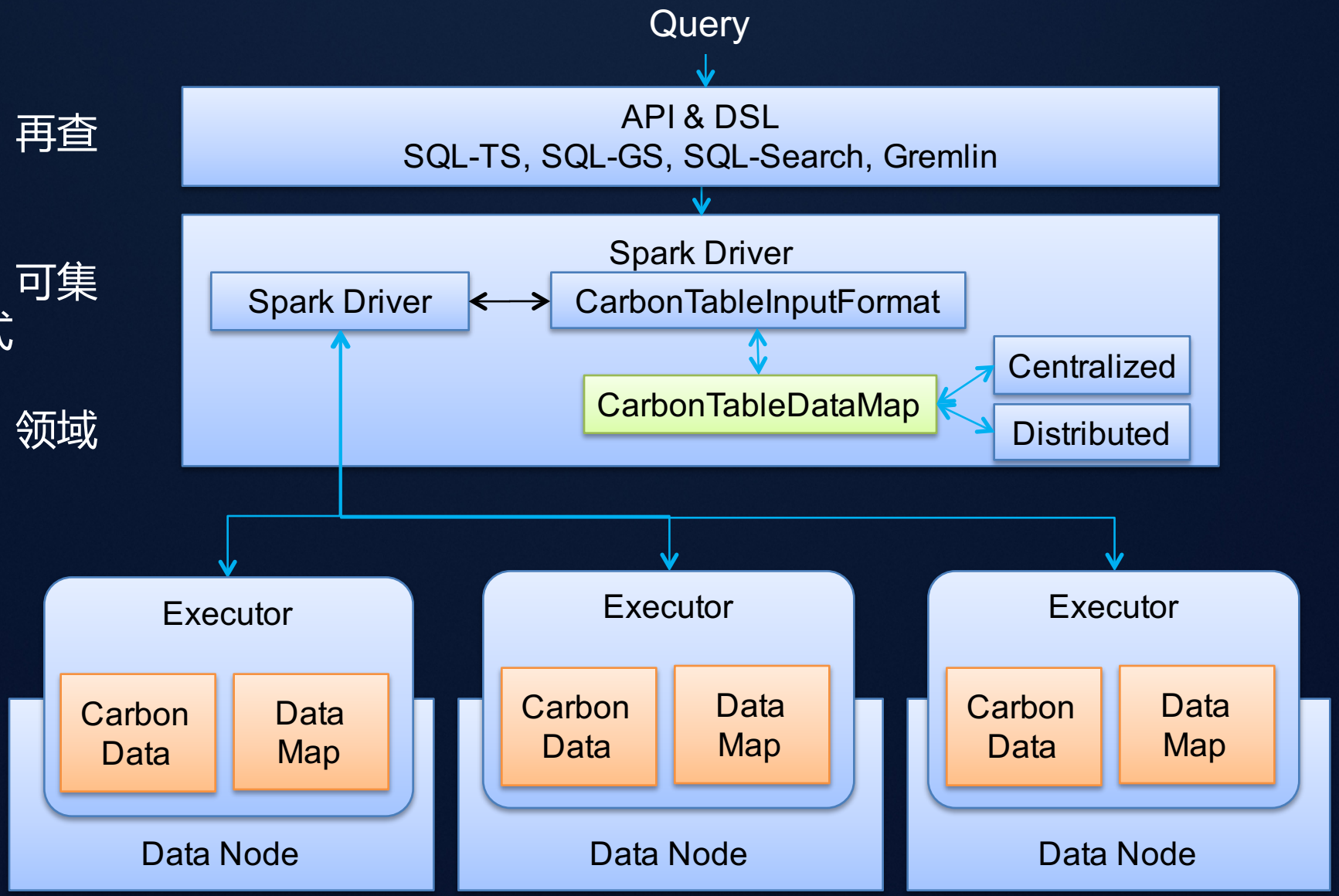
- 界面：基于SQL的领域扩展
- 数据组织：不同领域的分区、索引、预处理
- 存储格式：基于基础格式的扩展，列存、行存、时序、图、矩阵

多领域分析

时间序列分析	图分析	空间位置分析	文本检索	。。。
• 大吞吐写入 • 高压缩率 • 高并发点查 • 按**时间**聚合 • 按**维度**聚合 • 快速老化	• 给定顶点在边上遍历 • 按**边、顶点**属性过滤 • 按维度聚合 • 迭代式计算	• 遍历**区域**内的**点** • 按区域聚合 • 给定点计算KNN距离	• **分词** • **文本搜索** • 倒排、跳跃表 • 模糊匹配	

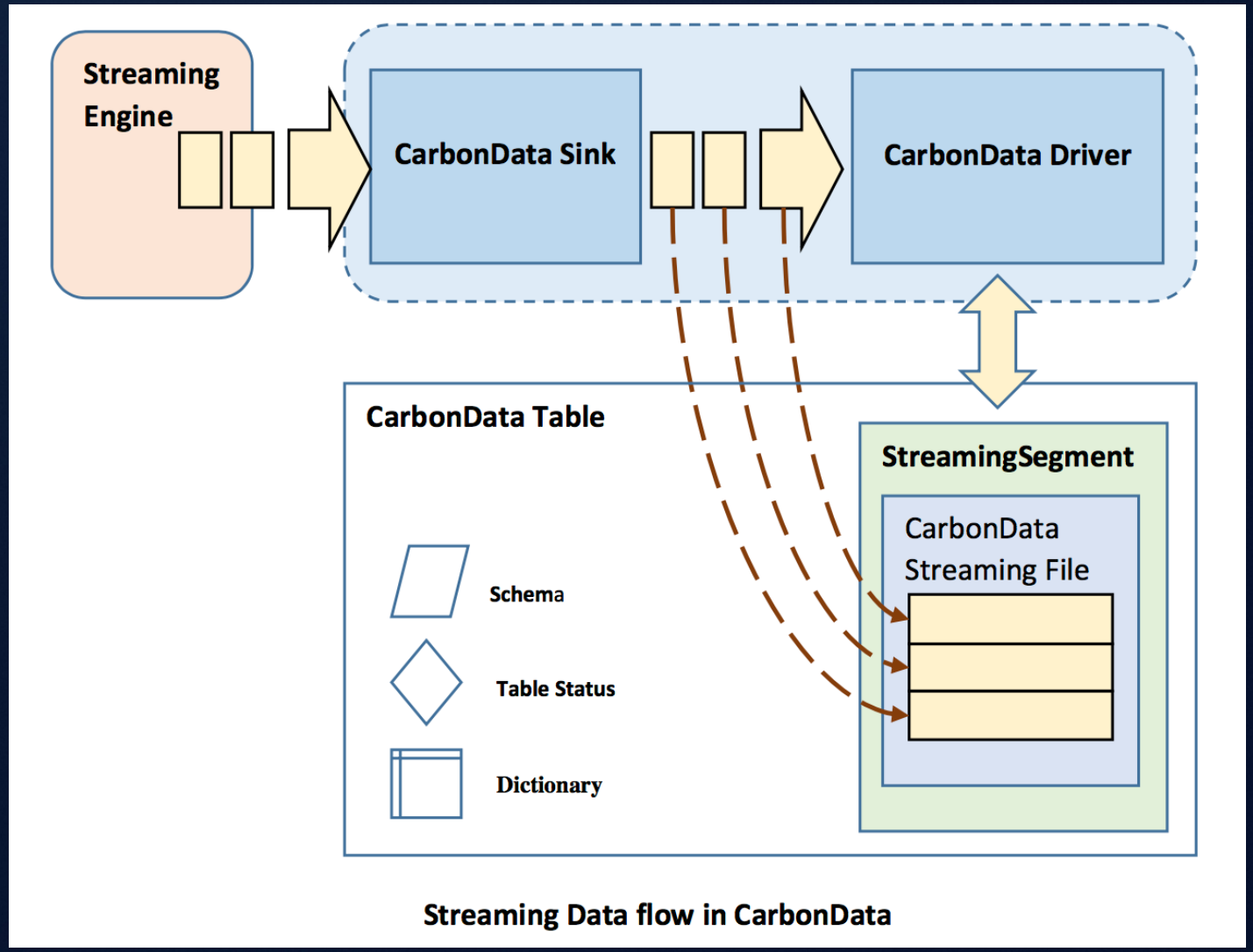
领域模型查询过程

- 先查DataMap、再查CarbonData
- DataMap查询：可集中式、可分布式
- DataMap类型：领域索引、预汇聚



流式入库

- 新增Carbon Streaming格式
- 新增Streaming Segment
- 支持多种流处理框架：SparkStreaming, Flink, Kafka
- 支持入库前对数据预处理
- 支持实时、历史数据联合查询分析



Streaming Data flow in CarbonData

欢迎参与Apache CarbonData社区

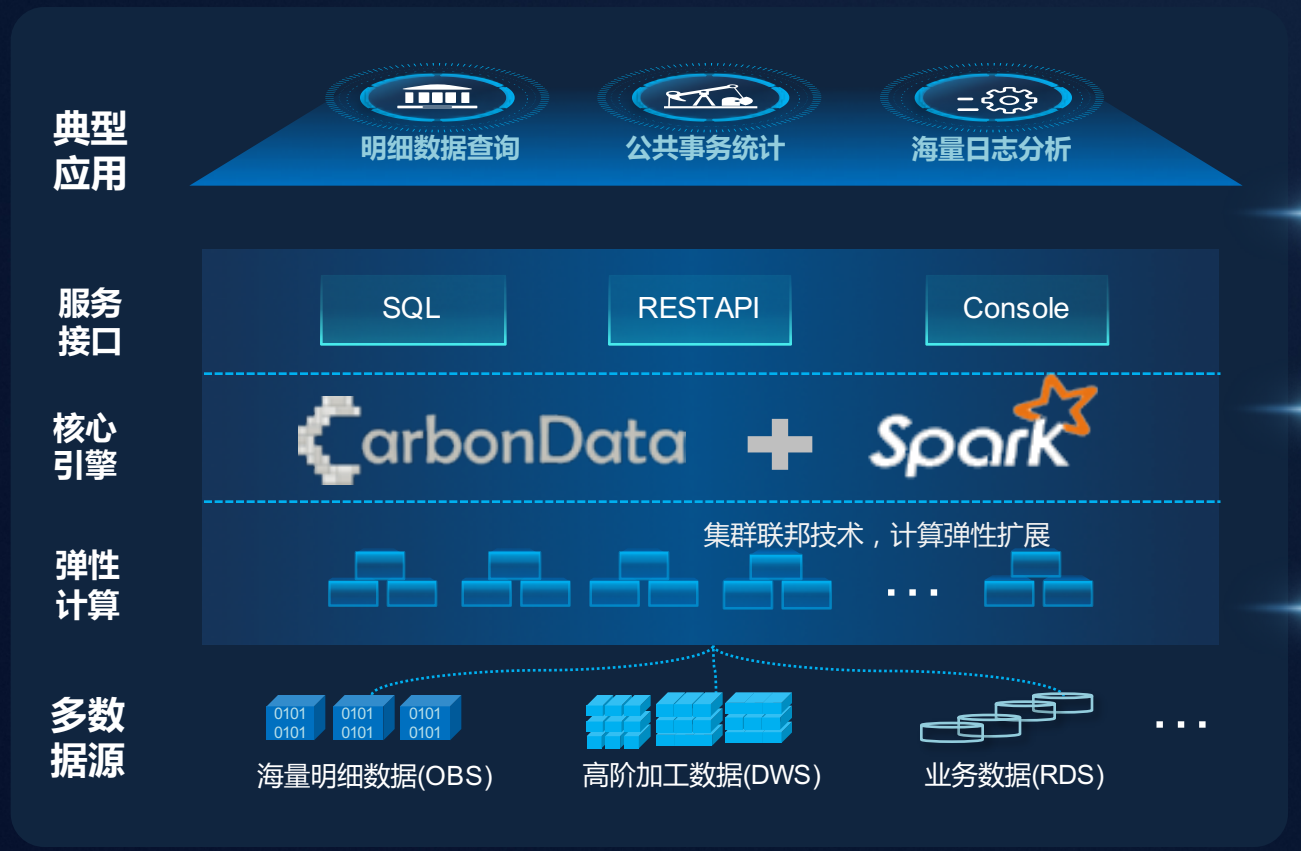
- website: http://carbondata.apache.org
- Code: https://github.com/apache/carbondata
- JIRA: https://issues.apache.org/jira/browse/CARBONDATA
- Mail list: dev@carbondata.apache.org, user@carbondata.apache.org

- 欢迎在Maillist上提问，共同探讨和开发CarbonData 2.0新特性

http://www.hwclouds.com/product/uquery.html

HUAWEI

UQuery：以CarbonData+Spark为核心引擎的数据查询服务



应用场景

- **海量明细数据查询**

直接对海量原始数据查询，无需加载转换，查询维度不固定，灵活多变，典型应用如流水审计，IoT设备运维仪表盘，能耗统计等。

- **海量日志分析**

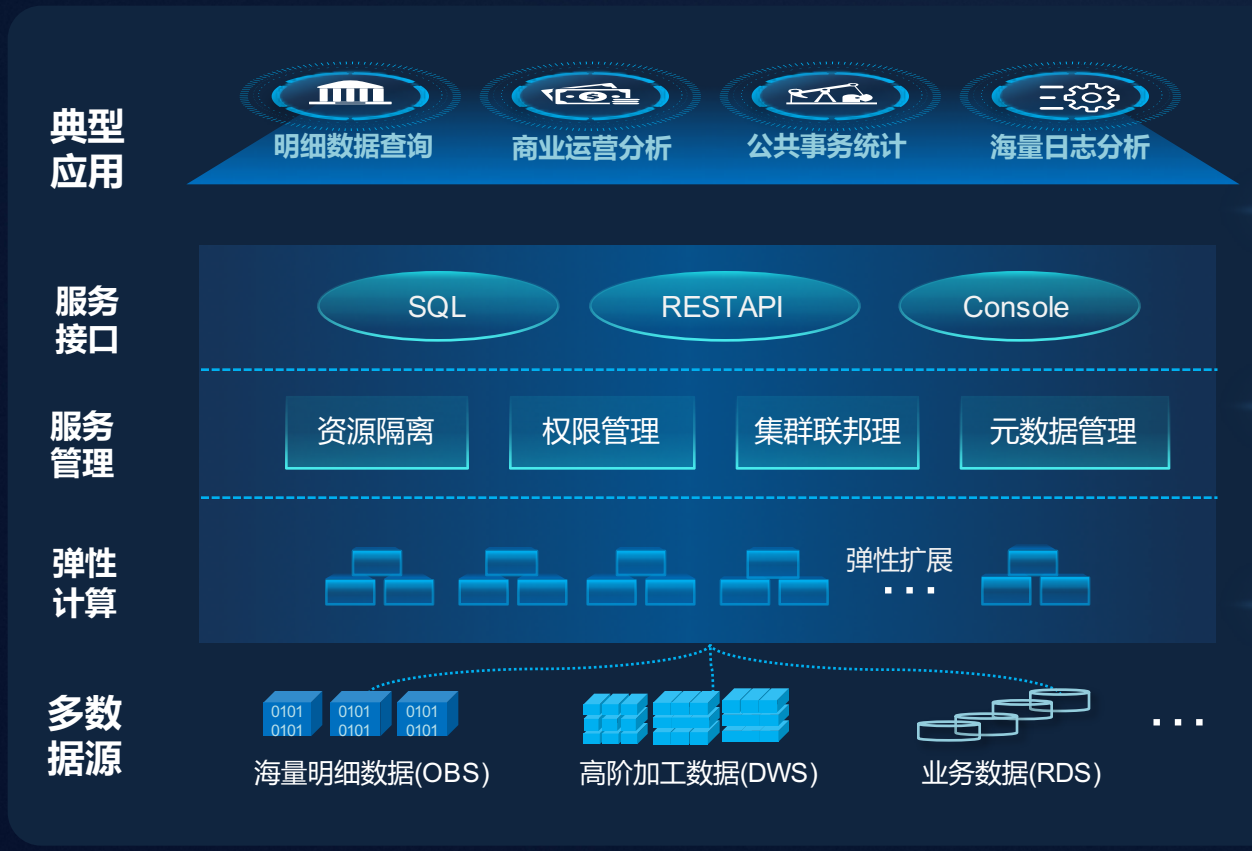
半结构化文本，事件日志探索分析，部分涉及模糊查询，典型应用如在线教育事件分析，热点追踪，消息分类刷选等。

- **公共事务统计审查**

统计计算，对比分析，响应时间要求秒级到分钟，典型应用如人流统计，犯罪追踪，关联案件查询，安全审计等。

UQuery是一种完全托管的数据查询服务，提供标准SQL接口、企业多租户能力，支持弹性扩展，让用户对云上多数据源进行简单便捷的探索分析

UQuery特点: 多数据源、多租户、免运维的查询服务



- **便捷易用，免运维**

  服务完全托管，无需管理任何基础设施，零维护成本，资源自动弹性扩展，用户不感知，无需关心资源是否够用。

- **标准SQL**

  使用SQL接口进行数据查询，即来即用。用户不感知

- **多数源联邦分析**

  支持OBS/服务本地存储/DWS/RDS（后两者路标规划）等云上多数据源联邦分析，数据无需加载转换。

- **企业多租户，数据权限控制**

  计算资源按租户隔离，保障用户作业SLA，支持表/列数据权限控制，方便企业内部安全访问管控。