

OopsOutOfMemory盛利的博客

RSS订阅

分布式计算|数据仓库|数据挖掘

个人资料



OopsOutOf...

关注

原创

粉丝

喜欢

评论

80

422

3

71

等级： 博客 6

访问：57万+

积分：5760

排名：5739

勋章：



直播系统源码



最新文章

Apache Helix简介

HDFS之Node角色

LinkedIn Cubert 实践指南

Understanding Cubert Concepts ( 二 ) Co-Partitioned Blocks

Understanding Cubert Concepts ( 一 ) Part itioned Blocks

微博



博主专栏



Spark SQL源码分析系列

阅读量：106254      11 篇

个人分类

spark	42篇
hive	8篇
scala	8篇
machine learning	3篇
shark	3篇

[展开](#)

归档

2015年8月	2篇
2015年7月	2篇
2015年6月	2篇
2015年4月	3篇
2014年12月	3篇

[展开](#)

热门文章

- Spark SQL 源码分析系列文章  
阅读量：22716
- Spark Hadoop集群部署与Spark操作HDFS运行详解---Spark学习笔记10  
阅读量：21866
- Spark RDD Transformation 详解---Spark学习笔记7  
阅读量：20923

Spark Streaming的窗口操作  
阅读量：18154

最新评论

- Lateral View用法与 ...  
zwzcdcll：很好，谢谢分享
- Spark SQL源码分析之核心流程  
zilianxiaozyu：可以的 虽然看着有点懵
- Spark计算Pi运行过程详解--...  
Coder\_Lotus：楼主，我正在用spark2.1版本结合hadoop2.8.1 通过java提交spark任务到ya...
- Spark Executor Dr...  
MaRinQ：文章真的写的好，膜拜大神~
- hiveUDAF求中位数  
zxyscz：你这个中位数最后计算结果不对啊

原 Spark SQL 源码分析之 In-Memory Columnar Storage 之 in-memory query

2014年10月02日 13:16:47

/\*\* Spark SQL源码分析系列文章\*/

前面讲到了Spark SQL In-Memory Columnar Storage的存储结构是基于列存储的。

那么基于以上存储结构，我们查询cache在jvm内的数据又是如何查询的，本文将揭示查询In-Memory Data的方式。

一、引子

本例使用hive console里查询cache后的src表。

**select value from src**

当我们将src表cache到了内存后，再次查询src，可以通过analyzed执行计划来观察内部调用。

即parse后，会形成InMemoryRelation结点，最后执行物理计划时，会调用InMemoryColumnarTableScan这个结点的方法。

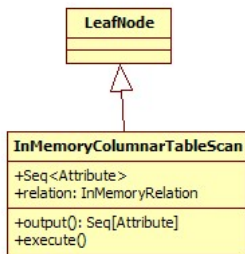
如下：

```
[java]
1. scala> val exe = executePlan(sql("select value from src").queryExecution.analyzed)
2. 14/09/26 10:30:26 INFO parse.ParseDriver: Parsing command: select value from src
3. 14/09/26 10:30:26 INFO parse.ParseDriver: Parse Completed
4. exe: org.apache.spark.sql.hive.test.TestHive.QueryExecution =
5. == Parsed Logical Plan ==
6. Project [value#5]
7.   InMemoryRelation [key#4,value#5], (false, 1000, (HiveTableScan [key#4,value#5], (MetastoreRelation default, src, None), None), None)
8.
9. == Analyzed Logical Plan ==
10. Project [value#5]
11.   InMemoryRelation [key#4,value#5], (false, 1000, (HiveTableScan [key#4,value#5], (MetastoreRelation default, src, None), None), None)
12.
13. == Optimized Logical Plan ==
14. Project [value#5]
15.   InMemoryRelation [key#4,value#5], (false, 1000, (HiveTableScan [key#4,value#5], (MetastoreRelation default, src, None), None), None)
16.
17. == Physical Plan ==
18. InMemoryColumnarTableScan [key#4,value#5], (false, 1000, (HiveTableScan [key#4,value#5], (MetastoreRelation default, src, None), None), (InMemoryRelation [key#4,value#5], (false, 1000, (HiveTableScan [key#4,value#5], (MetastoreRelation default, src, None), None), None)) //查询内存中表的入口
19.
20. Code Generation: false
21. == RDD ==
```

二、InMemoryColumnarTableScan

InMemoryColumnarTableScan是Catalyst里的一个叶子结点，包含了要查询的attributes，和InMemoryRelation（封装了我们缓存的In-Columnar Storage数据结构）。执行叶子节点，出发execute方法对内存数据查询。

- 1、查询时，调用InMemoryRelation，对其封装的内存数据结构的每个分区进行操作。
- 2、获取要请求的attributes，如上，查询请求的是src表的value属性。



```

[java]
1. private[sql] case class InMemoryColumnarTableScan(
2.     attributes: Seq[Attribute],
3.     relation: InMemoryRelation)
4.     extends LeafNode {
5.
6.
7.     override def output: Seq[Attribute] = attributes
8.
9.
10.    override def execute() = {
11.        relation.cachedColumnBuffers.mapPartitions { iterator =>
12.            // Find the ordinals of the requested columns. If none are requested, use the first.
13.            val requestedColumns = if (attributes.isEmpty) {
14.                Seq(0)
15.            } else {
16.                attributes.map(a => relation.output.indexWhere(_.exprId == a.exprId)) //根据表达式exprId找出对应列的ByteBuffer的索引
17.            }
18.
19.
20.            iterator
21.                .map(batch => requestedColumns.map(batch(_)).map(ColumnAccessor(_))) //根据索引取得对应请求列的ByteBuffer, 并封装为ColumnAccessor。
22.                .flatMap { columnAccessors =>
23.                    val nextRow = new GenericMutableRow(columnAccessors.length) //Row的长度
24.                    new Iterator[Row] {
25.                        override def next() = {
26.                            var i = 0
27.                            while (i < nextRow.length) {
28.                                columnAccessors(i).extractTo(nextRow, i) //根据对应index和长度, 从byterbuffer里取得值, 封装到row里
29.                                i += 1
30.                            }
31.                            nextRow
32.                        }
33.
34.                        override def hasNext = columnAccessors.head.hasNext
35.                    }
36.                }
37.        }
38.    }
39. }
40. }
  
```

查询请求的列，如下：

```

[java]
1. scala> exe.optimizedPlan
2. res93: org.apache.spark.sql.catalyst.plans.logical.LogicalPlan =
3. Project [value#5]
4.   InMemoryRelation [key#4,value#5], false, 1000, (HiveTableScan [key#4,value#5], (MetastoreRelation default, src, None), None)
5.
6.
7. scala> val relation = exe.optimizedPlan(1)
8. relation: org.apache.spark.sql.catalyst.plans.logical.LogicalPlan =
9. InMemoryRelation [key#4,value#5], false, 1000, (HiveTableScan [key#4,value#5], (MetastoreRelation default, src, None), None)
10.
11.
12. scala> val request_relation = exe.executedPlan
13. request_relation: org.apache.spark.sql.execution.SparkPlan =
14. InMemoryColumnarTableScan [value#5], (InMemoryRelation [key#4,value#5], false, 1000, (HiveTableScan [key#4,value#5], (MetastoreRelation default, sr
15. e))
16.
17. scala> request_relation.output //请求的列, 我们请求的只有value列
18. res95: Seq[org.apache.spark.sql.catalyst.expressions.Attribute] = ArrayBuffer(value#5)
19.
20. scala> relation.output //默认保存在relation中的所有列
21. res96: Seq[org.apache.spark.sql.catalyst.expressions.Attribute] = ArrayBuffer(key#4, value#5)
22.
  
```

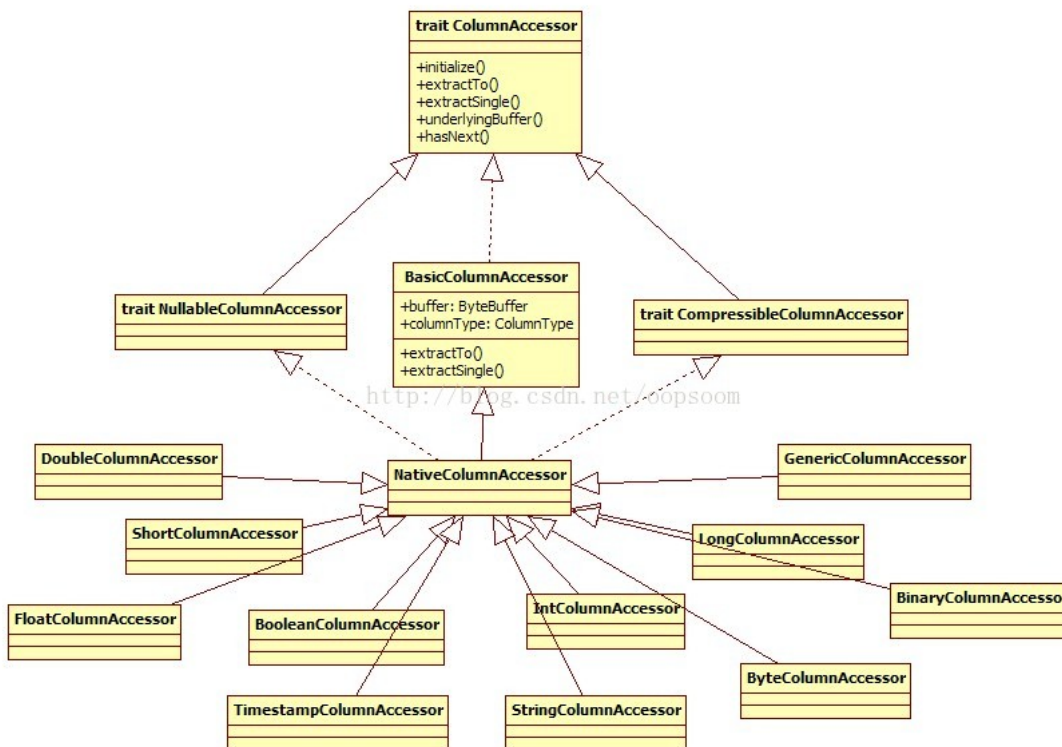
整个流程很简洁，关键步骤是第三步。根据ExprId来查找到，请求列的索引

**attributes.map(a => relation.output.indexWhere(\_.exprId == a.exprId))**

```
[java]
1. //根据exprId找出对应ID
2. scala> val attr_index = attributes.map(a => relation.output.indexWhere(_.exprId == a.exprId))
3. attr_index: Seq[Int] = ArrayBuffer(1) //找到请求的列value的索引是1，我们查询就从Index为1的bytebuffer中，请求数据
4.
5. scala> relation.output.foreach(e=>println(e.exprId))
6. ExprId(4) //对应<span style="font-family: Arial, Helvetica, sans-serif;">[key#4,value#5]</span>
7. ExprId(5)
8.
9. scala> request_relation.output.foreach(e=>println(e.exprId))
10. ExprId(5)
```

### 三、ColumnAccessor

ColumnAccessor对应每一种类型，类图如下：



最后返回一个新的迭代器：

```
[java]
1. new Iterator[Row] {
2.   override def next() = {
3.     var i = 0
4.     while (i < nextRow.length) { //请求列的长度
5.       columnAccessors(i).extractTo(nextRow, i) //调用columnType.setField(row, ordinal, extractSingle(buffer))解析buffer
6.       i += 1
7.     }
8.     nextRow //返回解析后的row
9.   }
10.
11.   override def hasNext = columnAccessors.head.hasNext
12. }
```

### 四、总结

Spark SQL In-Memory Columnar Storage的查询相对来说还是比较简单的，其查询思想主要和存储的数据结构有关。

即存储时，按每列放到一个bytebuffer,形成一个bytebuffer数组。

——EOF——

创文章，转载请注明：

转载自：[OopsOutOfMemory盛利的Blog](#)，作者：[OopsOutOfMemory](#)

本文链接地址：<http://blog.csdn.net/oopsoom/article/details/39577419>

注：本文基于署名-非商业性使用-禁止演绎 2.5 中国大陆(CC BY-NC-ND 2.5 CN)协议，欢迎转载、转发和评论，但是请保留本文作者署名和文章链接。如若需要用于授权方面的协商，请联系我。



文章标签：[spark sql](#) [catalyst](#) [storage](#) [in-memory](#)

个人分类：[spark](#)

所属专栏：[Spark SQL源码分析系列](#)

查看更多>>

想对作者说点什么？[我来说一句](#)

### Hive analyze命令解析

关于Hive analyze命令 1. 命令用法：  
□ 表与分区的状态信息统计  
ANALYZE TABLE tablename [PARTITION(partcol1[=val1], par...

cloud\_post 2014-04-29 21:38:33 阅读数：3605

### SparkSQL根据条件合并多条数据（测试）

SparkSQL合并多条数据测试，完整版

ASAS1314 2017-12-12 17:22:44 阅读数：300

### 深度好文！Web全栈工程师是怎么炼成的？他们都在做什么？

看后才知道全栈工程师缺乏的真相，让人吃惊.....

广告



### Spark SQL 源码分析之 In-Memory Columnar Storage 之 cache table

Spark SQL缓存到内存中的数据的存储策略

u014388509 2014-09-25 18:20:23 阅读数：8570

### spark hive执行树

前几天在做开发的时候无意中得到一段idea中spark运行hive语句得到的执行树：但是很困惑的是之后就没有再出现过了 如果有大神路过看到请指教该怎么打开这个查看。代码，执...

weinierzui 2017-05-20 12:09:54 阅读数：595

### spark hive hbase 结合

spark hive hbase 结合 业务需求，需要整合需要读取hive数据导入hbase中，一下是环境配置流程以及中间遇到的问题 1.spark读hive 需要copy hive-sit...

bigdataf 2018-01-18 14:04:41 阅读数：173

近十年来，随着Hadoop生态系统的不断完善，Hadoop早已成为大数据事实上的行业标准之一。面对当今互联网产生的巨大的TB甚至PB级原始数据，利用基于Hadoop的方案Hive早已是Ha...

 jarth 2016-09-27 18:17:23 阅读数：624

## 苏坡90后小伙在家无聊玩微信，存款惊呆父母

大集网络 · 顶新

## Spark SQL核心类解析

一、执行计划实体相关 Tree是Catalyst执行计划表示的数据结构。LogicalPlans，Expressions和Physical Operators都可以使用Tree来表示。T...

 junerli 2017-11-24 14:15:10 阅读数：336


## Hive代码组织及架构简单介绍

转:<http://my.oschina.net/u/1243452/blog/173716> hive 源码 目录[-] hive三个主要组件 其他组件 hive辅助...

 pzasdq 2016-03-08 15:15:26 阅读数：214

## 利用Hive进行数据分析

近十年来，随着Hadoop生态系统的不断完善，Hadoop早已成为大数据事实上的行业标准之一。面对当今互联网产生的巨大的TB甚至PB级原始数据，利用基于Hadoop的方案Hive早已是Ha...

 wh\_springer 2016-07-06 21:30:20 阅读数：11117


## spark 读取 hdfs 数据分区规则

下文以读取 parquet 文件 / parquet hive table 为例：hive metastore 和 parquet 转化的方式通过 `spark.sql.hive.convertMe...`

 lsshslw 2018-04-03 10:10:16 阅读数：146

## SQL Server 2016新特性：In-Memory OLTP

内存中OLTP有助于OLTP工作负荷实现显著的性能改进，并减少了处理时间。可以通过将表声明成“内存中优化”来启用内存中OLTP的功能。内存优化表完全支持事务，；ansact-SQL进行访...

 Burgess\_Liu 2016-06-01 16:57:03 阅读数：2839

## Expert SQL Server In-Memory OLTP(2nd) 无水印pdf

Expert SQL Server In-Memory OLTP(2nd) 英文无水印pdf 第2版 pdf所有页面使用FoxitReader和PDF-XChangeViewer测试都可以打开 本资源转载自网络，如有侵权，请联系删除 本资源转载自网络，如有侵权，请联系上传者或csdn删除

下载 2017年09月29日 12:20

## mongoDB In-Memory Storage Engine

原文链接 On this page Specify In-Memory Storage EngineConcurrencyMemory UseDurabilityDeplo...

 u010385646 2016-09-13 08:59:10 阅读数：727


## 程序员如何在业余时间精进技术？

全球500万工程师首选，提升前端开发、数据分析、深度学习



## Memcached、Redis、RDD ( Spark ) 的数据处理性能对比 ( Efficient in-memory data management: an analysis

摘要： 本论文分析了三个内存数据管理系统的性能：Memcached（分布式缓存）、Redis（一个开源的使用ANSI C语言编写、支持网络、可基于内存亦可持久化的al...

 hust\_sheng 2015-08-05 21:12:25 阅读数：1118

## 树莓派学习笔记——交叉编译练习之SQLite3安装

本博文可能并没有太多使用价值，仅仅是为了练习而练习。在树莓派上使用SQLite有很多的方法，安装的方法也有很多。【1】如果使用Python，那么不必安装SQLite因为自带SQLite...

### 【Oracle】Oracle 12c DB In-Memory入门实验手册（二）

（二）加载数据到IM column store 在实验手册（一）中介绍了IM的基础操作，objects如何开启IM column store。链接：<http://blog.csdn.net/badly9>

 badly9

2015-11-11 09:53:13

阅读数：1346

### 【Oracle】ORACLE 12c DB In-Memory简述及启用

Oracle DB In-Memory是预装在Oracle Database 12c（12.1.0.2之后的版本）中的，不需要安装其他软件或者是重新编译现有的数据库软件。这是因为In-Memory选...

 badly9

2015-11-08 23:23:55

阅读数：2567

### Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing

1) Goals and Overview Our goal is to provide an abstraction that supports applications with working...

 macyang

2011-12-18 23:53:27

阅读数：2125

### 妈妈只用了这一招让孩子考到年级第一

孩子成绩太差？关键时期妈妈狠心做了这件事，1个月老师都惊呆了



百度广告 ▾

### Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing

1 Intruction 问题1：许多框架缺乏充分利用分布式内存的抽象，这使得它们不适用于大量计算都需要重用中间结果的情形，但数据重用又比较常见，比如许多迭代法、交互式数...

 u012319493

2016-11-22 13:10:56

阅读数：748

### in-memory databases

metadata的本质是一个只读关系数据库

 kun1234567

2015-11-06 15:36:56

阅读数：1418

### Oracle Database 12c In-Memory 基本原理与简介

概要Database In-Memory是Oracle数据库的选项，类似于RAC，ADG，发布于12.0.1.2。它的主要目的是利用内存的速度和优化的列格式来加速分析。以下我们用DBI...

 stevensxiao

2016-05-17 16:38:52

阅读数：6459

### 12c In-Memory常用命令笔记

1.IMC初始化设置 show parameter inmemory alter system set inmemory\_size=4g scope=spfile; 2.查询在imc中...

 znanbeijing

2015-02-15 17:03:01

阅读数：833

### SQLite学习笔记（1）-安装及其点命令

SQLite安装 在Windows上安装SQLite 访问[点击](#)进行下载，从Windows区进行下载 下载 [sqlite-shell-win32-\\*.zip](#) 和 [sqlite-dll-win32-...](#)

 zzmgood

2015-08-18 13:37:28

阅读数：2879

### 优达学城，授予梦想者改变世界的力量！

Python、人工智能、VR、小程序开发，你想得到的顶尖课程都在这里！



### Expert SQL Server In-Memory OLTP, 2nd Edition.pdf

Expert SQL Server In-Memory OLTP, 2nd Edition.pdf Expert SQL Server In-Memory OLTP, 2nd Edition.pdf

下载 2017年09月06日 09:03

### spark2.2.0源码学习过程记录：Day3

Day3 1、读《[apache spark 源码剖析](#)》第三章第3.2节、3.3节 因为3.3节的内容是讲repl的，我暂时并不关系，所以这部分内容看看书就可以了 而3.2节的内容是讲S...

 BLUE\_YEAH

2017-09-02 16:05:42

阅读数：445



shaguabufadai

2017-05-04 11:20:27

阅读数：285

### OCP-1Z0-053-V12.02-75题

75.View the Exhibit. As shown in the diagram, **in-memory** statistics are transferred to the disk at ...

rlhua

2013-11-06 16:31:25

阅读数：5472

### High Performance **in-memory** computing with Apache Ignite-Leanpub(2017).pdf

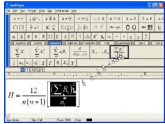
My first acquaintance with High load systems was at the beginning of 2007, and I started working on a real-world project since 2009. From that moment, I spent most c with Cassandra, Hadoop, and numerous CEP tools. Our first Hadoop project (the year 2011-2012) with a cluster of 54 nodes often disappointed me with its long startu ever been satisfied with the performance of our applications and was always looking for something new to boost the performance of our information systems. During th ied HazelCast, Ehcache, Oracle Coherence as **in-memory** caches to gain the performance of the applications. I was usually disappointed from the complexity of using r from their functional limitations. When I first encountered Apache Ignite, I was amazed! It was the platform that I'd been waiting on for a long time: a simple spring bas with a lot of awesome features such as DataBase caching, Big data acceleration, Streaming and compute/service grids. In 2015, I had participated in Russian HighLoe 1 with my presentation and started blogging in Dzone/JavaCodeGeeks and in my personal blog2 about developing High-load systems. They became popular shortly, a ot of feedback from the readers. Through them, I clarified the idea behind the book. The goal of the book was to provide a guide for those who really need to implemen platform in their projects. At the same time, the idea behind the book is not writing a manual. Although the Apache Ignite platform is very big and growing day by day, v nly on the features of the platform (from our point of view) that can really help to improve the performance of the applications. We hope that High-performance **in-mem** ith Apache Ignite will be the go-to guide for architects and developers: both new and at an intermediate level, to get up and to develop with as little friction as possible.

下载 2018年03月28日 21:26

### LaTeX--Beautifultypesettingmade

latex

百度广告



### 【Oracle】Oracle 12c DB **In-Memory**入门实验手册（一）

该手册实验基础要在实例级别启用IM column store，开启方法参考上篇文章：链接：<http://blog.csdn.net/badly9/article/details/49724983>（ ...

badly9

2015-11-09 23:19:39

阅读数：1450

### 内存缓存(**in-memory** cache)之redis

一、背景最近项目需要，需要数据库中的部分数据缓存到内存中，提高我们的查询与处理速度，传统的处理方式有两个方面拖慢了速度。 一、从web服务器到数据库服 请求 二、数据库服务器数据处理 ...

xinjianwuhun1991

2015-08-31 22:18:32

阅读数：1851