B站基于缓存优化PRESTO集群查询性能

杨洋 大数据开发工程师

个人简介

杨洋

- bilibili大数据开发工程师
- 2021年6月份加入b站工作至今
- 在团队中主要负责Presto与Alluxio的研发
- 对分布式计算、存储与调度方面有浓厚兴趣

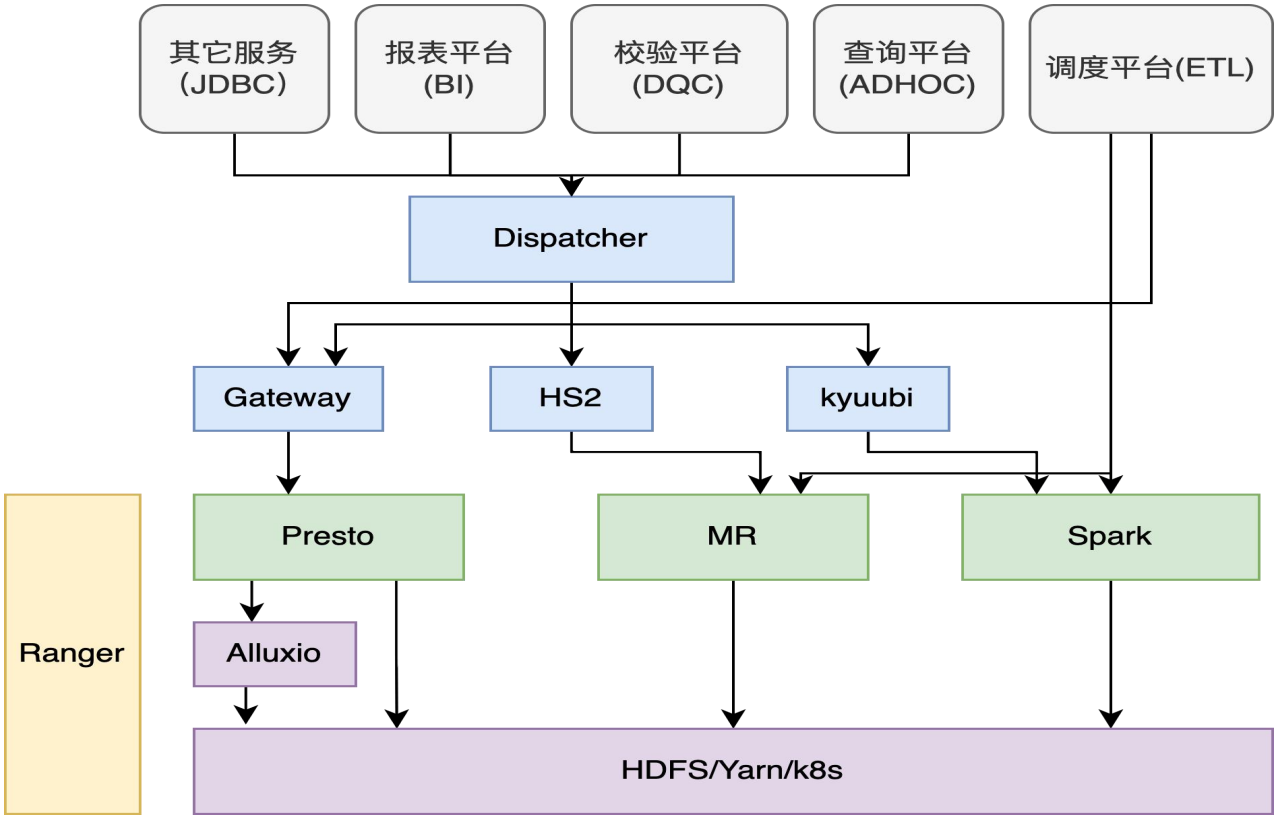
目录 CONTENT

bilibili | DataFun.

01

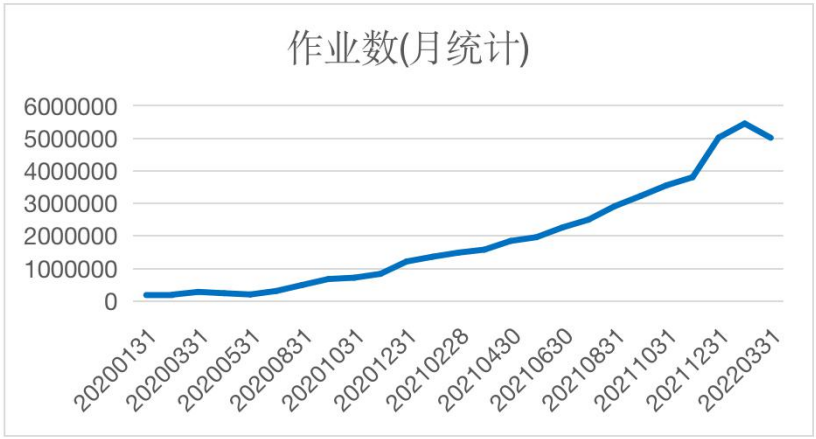
集群架构

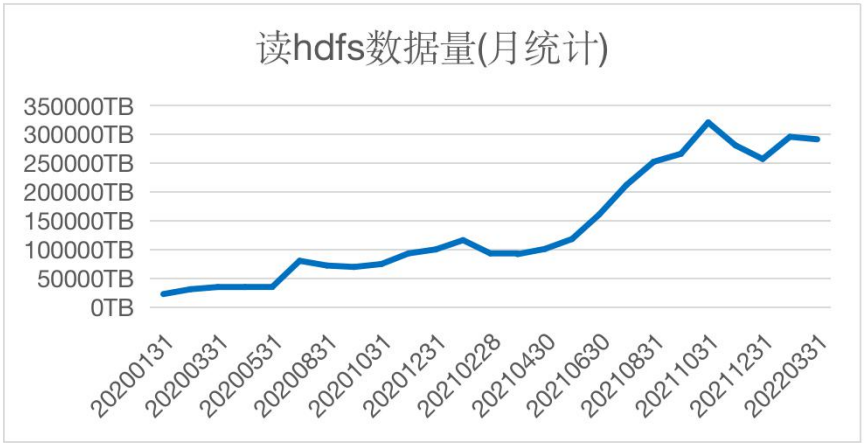
集群架构-B站SQL On Hadoop



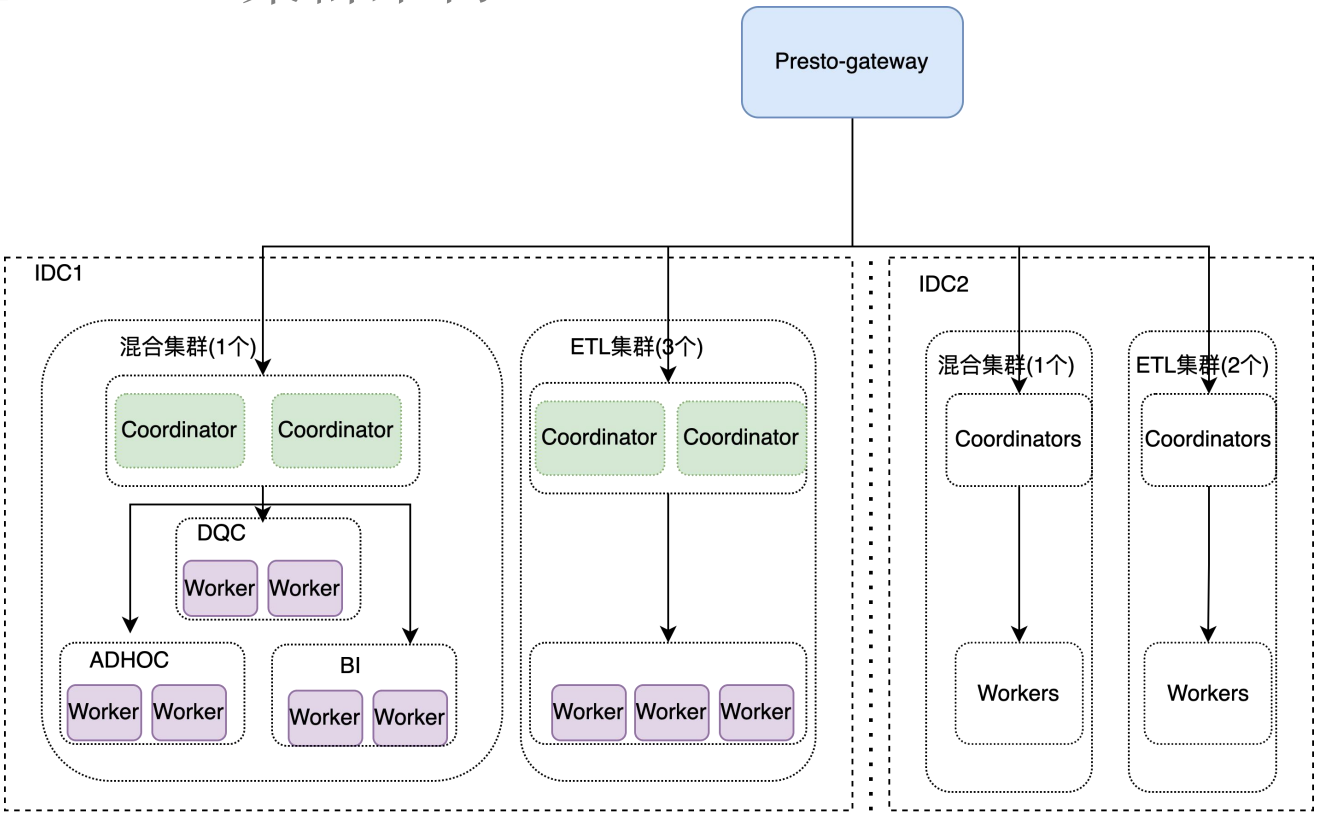
集群架构-Presto集群现状

	Cluster1	Cluster2	Cluster3	Cluster4
IDC1	414 + 2	186 + 2	14 + 1	111 + 2
IDC2	441 + 2	270 + 2	0	85 + 2





集群架构-Presto集群架构



bilibili | DataFun.

02

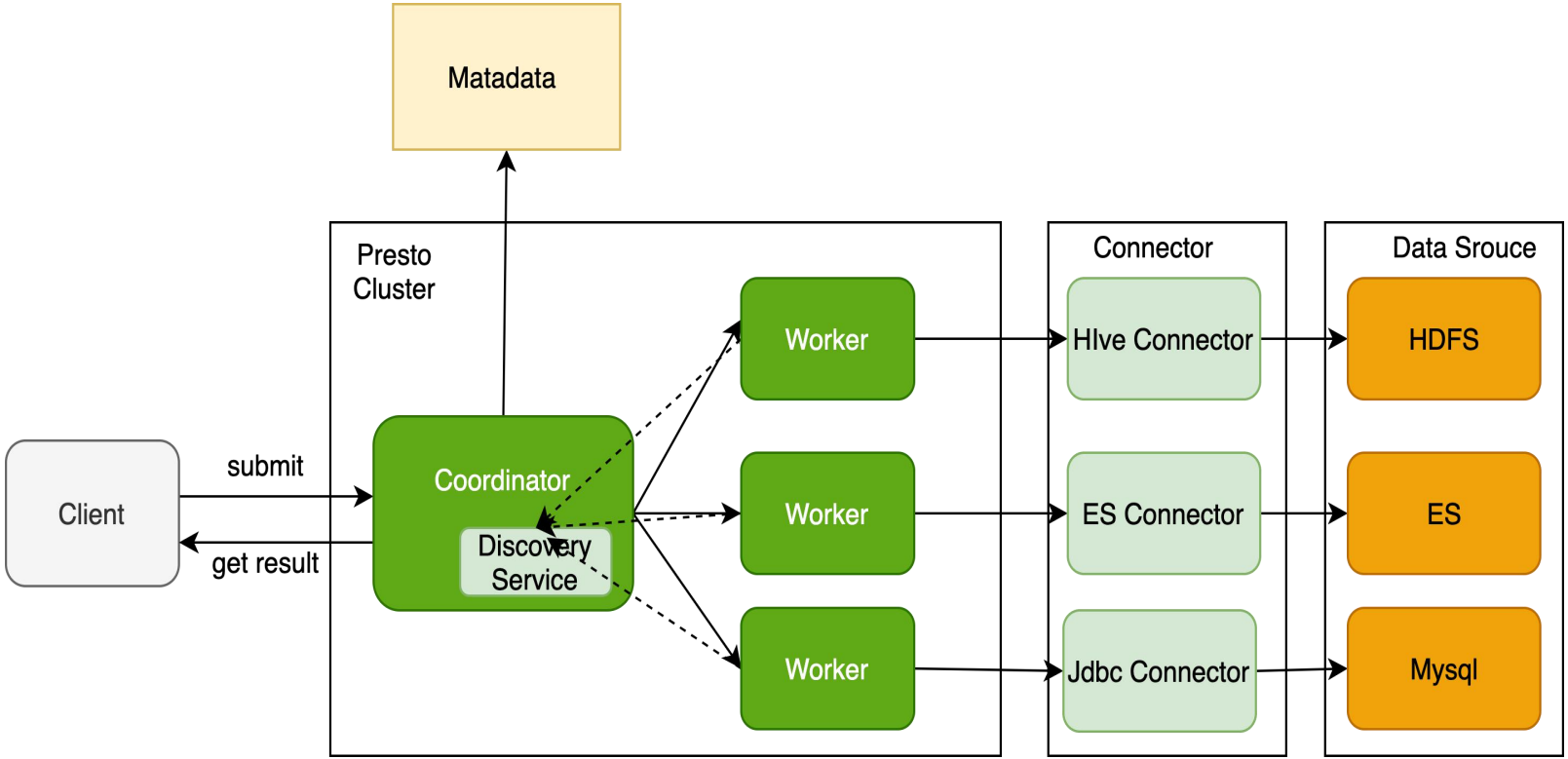
Presto简介

Presto简介-Presto历史

Presto于2013年11月份由FaceBook开源的一个分布式Sql查询引擎，设计之初是为了进行OLAP数据查询，支持标准的ANSI SQL，支持多数据源。

	PrestoSQL(trino)	PrestoDB
主导开发	Presto Software Foundation	Linux Foundation
社区活跃度	高	低
功能	olap	etl
最新版本	Release 384	Release 0.273

Presto简介-Presto基本原理



bilibili | DataFun.

03

Presto改造

Presto改造

Presto在B站的实践：https://mp.weixin.qq.com/s/9_lSIFSw5o8sFC8foEtA7w

bilibili | DataFun.

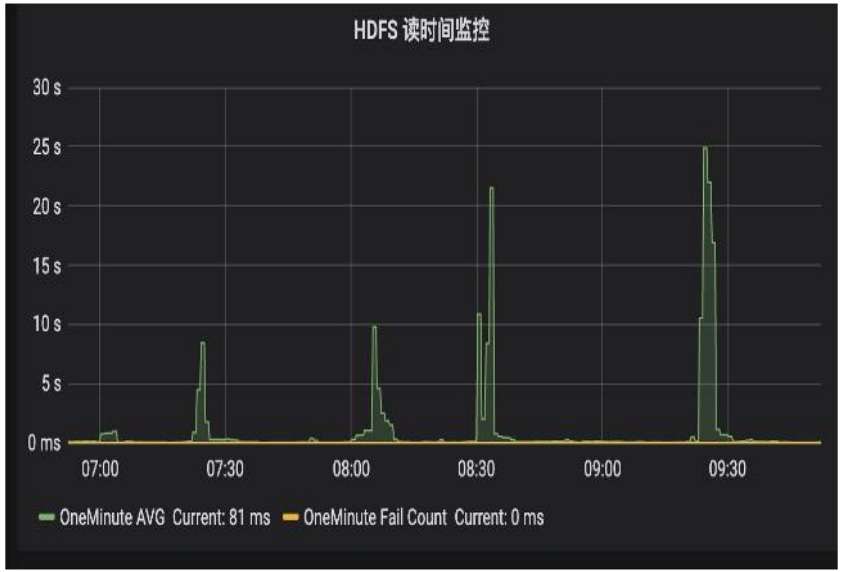
bilibili | DataFun.

04

Presto on Alluxio

Presto on Alluxio-背景介绍--Presto痛点

- 计算存储分离架构带来网络开销
- 容易受慢rpc或热dn影响，查询性能不稳定
- 查询缺乏locality，性能有待提升



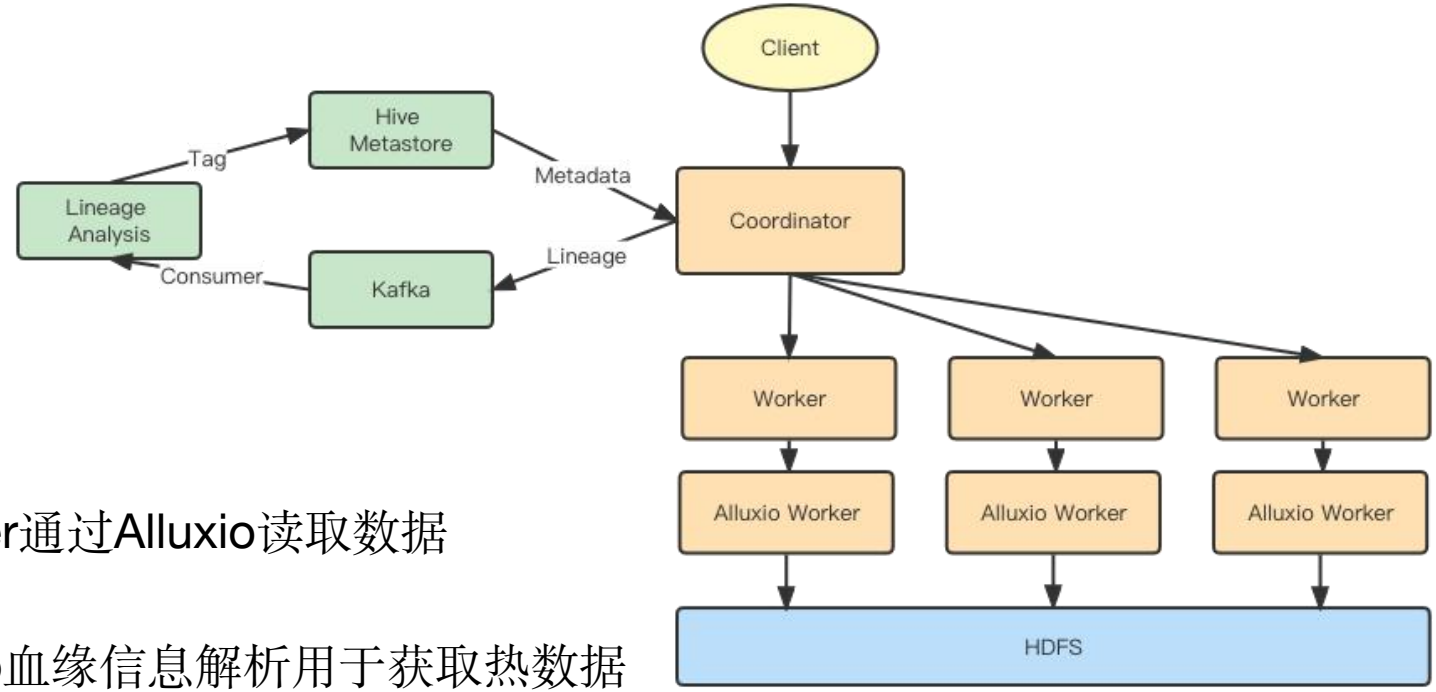
Presto on Alluxio-背景介绍--热数据

cluster: jscs-ai-offl dbName: tableName: tag: 0

cluster	dbName	tableName	partitionTable	location	sameLocation	addTime	updateTime	tableHeat
jscs-ai-offline	ai	ards	1		1	2021-11-11T08:31:54.000+00:00	2022-03-04T03:41:24.000+00:00	866
jscs-ai-offline	ai	ards	1		1	2022-02-14T08:37:40.000+00:00	2022-03-04T03:41:02.000+00:00	658
jscs-ai-offline	b_dwm	'b_d ihuo	1		1	2022-02-14T08:34:40.000+00:00	2022-03-04T03:41:37.000+00:00	568
jscs-ai-offline	ai	_dis	1		1	2022-02-14T08:51:29.000+00:00	2022-03-04T03:31:22.000+00:00	555
jscs-ai-offline	ai	nterf	1		1	2021-11-16T14:04:14.000+00:00	2022-03-04T03:41:40.000+00:00	402

bilibili | DataFun.

Presto on Alluxio-Alluxio引入



- Worker通过Alluxio读取数据
- Presto血缘信息解析用于获取热数据

Presto on Alluxio-整合Alluxio需要考虑的点

- Alluxio与HDFS的scheme不同
- Alluxio缓存数据的确定
- Alluxio与HDFS的数据一致性保证

Presto on Alluxio-Alluxio与HDFS的scheme不同

社区：

- 支持Alluxio的连接器（高版本Presto）
- 从Alluxio中获取元数据（无需从HMS中获取）
- 使Alluxio的SDS模块与底层HMS通信

团队：

- 改造hive connector
- 识别分区tag参数判断是否走Alluxio

其他：

- 维护一套新的HMS（用于Adhoc）
- 设置白名单（用于需要缓存的表）
- 使新HMS与原HMS保持同步

Presto on Alluxio-缓存数据的确定

热数据tag设置：

- 将Presto query血缘信息吐到Kafka
- 通过Kafka消费程序，分析血缘依赖信息并落地到Tidb
- 通过缓存策略服务，确定需要加载的热数据
- 给热数据设置tag（Tidb与HMS中）

缓存策略：

- 计算访问热度（一周内的访问频率均值）
- 计算TTL（离当前最远的热分区的时间跨度）
- 剔除超过TTL的分区


```
{
    "queryId": "20220216_085627_00000_inqkv",
    "querystr": "select items... from ai.tablexxx where log_date||log_hour between '20220209'||'21' and '20220210'||'22' limit 10;"
    "lineageInfo": "{\"inputs\":[{\"catalogName\":\"hive\",\"schema\":\"ai\",\"table\":\"xxxx\",\"columns\":[\"key\",\"log_hour\",\"value#features\",\"log
    Info\":{\"partitionIds\":[\"log_date=20220209/log_hour=21\",\"log_date=20220209/log_hour=22\",\"log_date=20220209/log_hour=23\",\"log_date=20220210/
    log_hour=00\",\"log_date=20220210/log_hour=01\",\"log_date=20220210/log_hour=02\",\"log_date=20220210/log_hour=03\",\"log_date=20220210/log_hour=04\",
    log_hour=05\",\"log_date=20220210/log_hour=06\",\"log_date=20220210/log_hour=07\",\"log_date=20220210/log_hour=08\",\"log_date=20220210/log_hour=09\",
    log_hour=10\",\"log_date=20220210/log_hour=11\",\"log_date=20220210/log_hour=12\",\"log_date=20220210/log_hour=13\",\"log_date=20220210/log_hour=14\",
    log_hour=15\",\"log_date=20220210/log_hour=16\",\"log_date=20220210/log_hour=17\",\"log_date=20220210/log_hour=18\",\"log_date=20220210/log_hour=19\",
    log_hour=20\",\"log_date=20220210/log_hour=21\",\"log_date=20220210/log_hour=22\"],\"truncated\":false}}],\"output\":null,
    "inputTables": "[ai.xxx]",
    "inputCols": "[ai.xxx.key, ai.xxx.log_date, ai.xxx.log_hour, ai.xxx.value#features]"}"
}
```

Presto on Alluxio-数据一致性保证

社区：

- 通过参数控制和HDFS的元数据同步

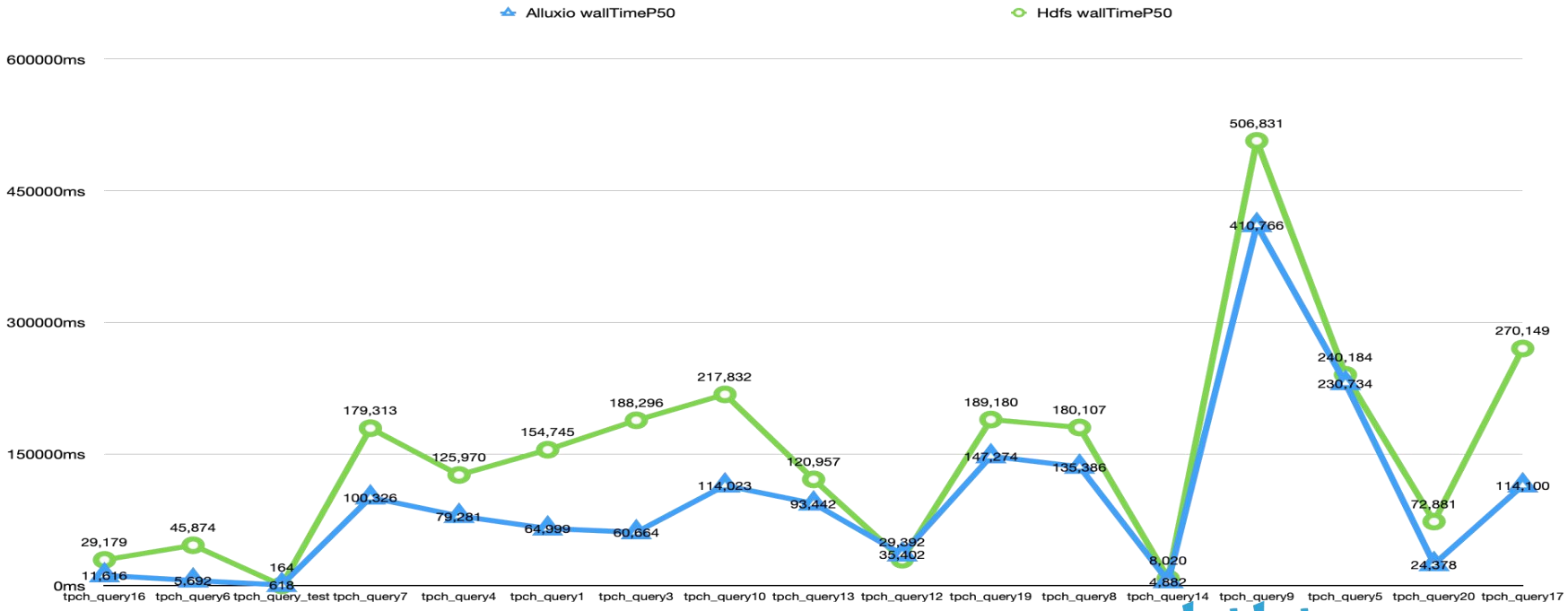
```
alluxio.user.file.metadata.sync.interval=0
alluxio.user.file.metadata.load.type=ALWAYS
```

团队：

- 开发缓存失效服务（监听Hive meta event）
- 监听add partition事件，load需缓存的新分区

Presto on Alluxio-TPC-H Benchmark性能测试

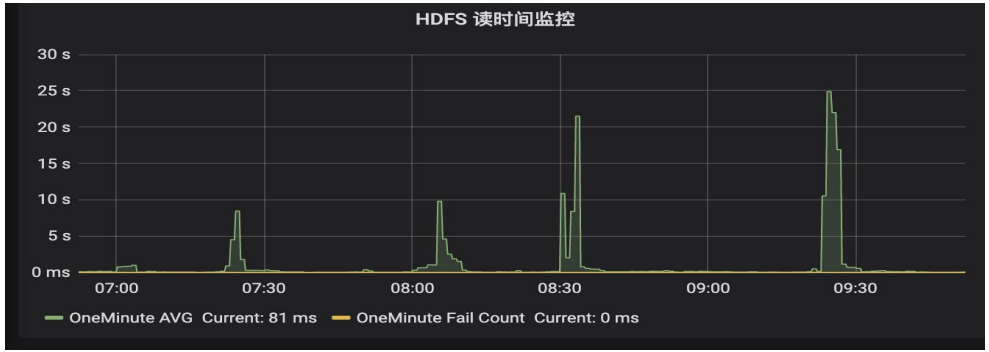
实验效果：平均下来可节省约20%的查询时间



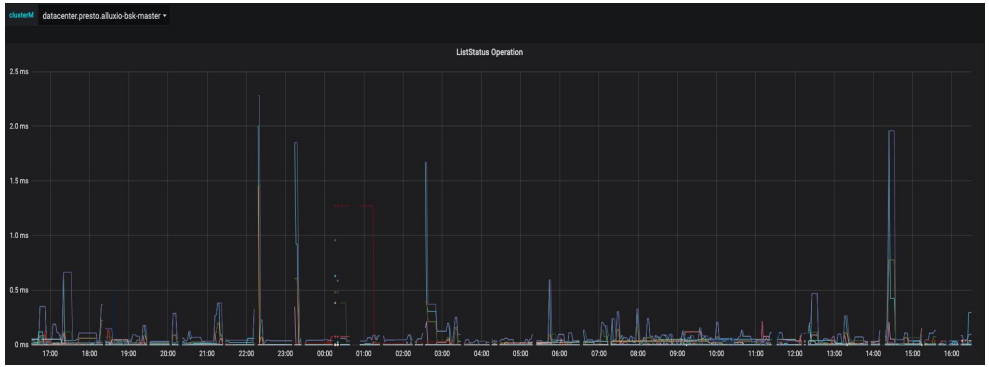
Presto on Alluxio-线上效果

- 接入约30%的BI业务到缓存
- 已缓存约20w分区（约45TB）
- 读HDFS稳定性提升，2.5ms以内

改进前：



改进后：



Presto on Alluxio-Alluxio线上故障

现象：线上Master进程偶发crash

链接：https://github.com/Alluxio/alluxio/pull/14856

https://groups.google.com/g/rocksdb/c/PwapmWw

```
Stack: [0x00007f6e069ea000,0x00007f6e06aeb000],  sp=0x00007f6e06ae8cd8,  free space=1019k
Native frames: (J=compiled Java code, j=interpreted, Vv=VM code, C=native code)
C  [libc.so.6+0x144c40]
C  [librocksdbjni9126513507264458711.so+0x587ff2]  rocksdb::BlockBasedTableIterator::CheckDataBlockWithinUpperBound()+0x92
C  [librocksdbjni9126513507264458711.so+0x588277]  rocksdb::BlockBasedTableIterator::InitDataBlock()+0x267
C  [librocksdbjni9126513507264458711.so+0x588d28]  rocksdb::BlockBasedTableIterator::FindBlockForward()+0x348
C  [librocksdbjni9126513507264458711.so+0x589155]  rocksdb::BlockBasedTableIterator::Next()+0xf5
C  [librocksdbjni9126513507264458711.so+0x5891d6]  rocksdb::BlockBasedTableIterator::NextAndGetResult(rocksdb::IterateResult*)+0x16
C  [librocksdbjni9126513507264458711.so+0x474371]
C  [librocksdbjni9126513507264458711.so+0x5e1a38]  rocksdb::MergingIterator::Next()+0x38
C  [librocksdbjni9126513507264458711.so+0x5ddb13]  rocksdb::MergingIterator::NextAndGetResult(rocksdb::IterateResult*)+0x13
C  [librocksdbjni9126513507264458711.so+0x3cbb74]  rocksdb::DBIter::Next()+0x284
J 6677  org.rocksdb.RocksIterator.next0(J)V (0 bytes) @ 0x00007f89a920e870 [0x00007f89a920e7c0+0xb0]
J 16658 C2 alluxio.master.metastore.rocks.RocksBlockStore.getLocations(J)Ljava/util/List; (213 bytes) @ 0x00007f89a982405c [0x00007f89a9823740+0x91c]
J 24828 C2 alluxio.master.block.DefaultBlockMaster.generateBlockInfo(J)Ljava/util/Optional; (377 bytes) @ 0x00007f89ab6fa0fc [0x00007f89ab6f9d80+0x37c]
J 13631 C2 alluxio.master.block.DefaultBlockMaster.getBlockInfoList(Ljava/util/List;)Ljava/util/List; (64 bytes) @ 0x00007f89a9bdb508 [0x00007f89a9bdb280+0x288]
J 13711 C2 alluxio.master.file.DefaultFileSystemMaster.getFileInfoInternal(Lalluxio/master/file/meta/LockedInodePath;Lcom/codahale/metrics/Counter;)Lalluxio/wire/FileInfo; (399 bytes) @ 0x00007f89a9c2ead0
[0x00007f89a9c2bec0+0x2c10]
J 21762 C2 alluxio.master.file.DefaultFileSystemMaster.getFileInfo(Lalluxio/AlluxioURI;Lalluxio/master/file/contexts/GetStatusContext;)Lalluxio/wire/FileInfo; (780 bytes) @ 0x00007f89ab49b3e8
[0x00007f89ab49a300+0x10e8]
J 14424 C2 alluxio.master.file.FileSystemMasterClientServiceHandler.lambda$getStatus$8(Lalluxio/grpc/GetStatusPRequest;Lalluxio/grpc/GetStatusPOptions;Lio/grpc/stub/StreamObserver;)Lalluxio/grpc/
GetStatusPResponse; (55 bytes) @ 0x00007f89a93ef900 [0x00007f89a93eece0+0xc20]
J 13876 C2 alluxio.master.file.FileSystemMasterClientServiceHandler$$Lambda$977.call()Ljava/lang/Object; (20 bytes) @ 0x00007f89a9eb69dc [0x00007f89a9eb69a0+0x3c]
J 11452 C2 alluxio.RpcUtils.callAndReturn(Lorg/slf4j/Logger;Lalluxio/RpcUtils$RpcCallableThrowsIOException;Ljava/lang/String;ZLjava/lang/String;[Ljava/lang/Object;)Ljava/lang/Object; (470 bytes) @
0x00007f89aa17fba4 [0x00007f89aa17ee00+0xda4]
J 13716 C2 alluxio.grpc.FileSystemMasterClientServiceGrpc$MethodHandlers.invoke(Ljava/lang/Object;Lio/grpc/stub/StreamObserver;)V (504 bytes) @ 0x00007f89a8a8963c [0x00007f89a8a88320+0x131c]
J 17540 C2 io.grpc.stub.ServerCalls$UnaryServerCallHandler$UnaryServerCallListener.onHalfClose()V (82 bytes) @ 0x00007f89a9364bbc [0x00007f89a9364b60+0x5c]
J 13713 C2 alluxio.security.authentication.ClientIpAddressInjector$1.onHalfClose()V (16 bytes) @ 0x00007f89a9c665cc [0x00007f89a9c66460+0x16c]
J 21917 C2 alluxio.security.authentication.AuthenticatedUserInjector$1.onHalfClose()V (23 bytes) @ 0x00007f89aacb1c28 [0x00007f89aacb1be0+0x48]
J 25513 C2 io.grpc.internal.ServerImpl$JumpToApplicationThreadServerStreamListener$1HalfClosed.runInContext()V (73 bytes) @ 0x00007f89abb1b48c [0x00007f89abb1b360+0x12c]
J 11960 C2 io.grpc.internal.ContextRunnable.run()V (35 bytes) @ 0x00007f89a84bda38 [0x00007f89a84bd900+0x138]
J 24439 C2 io.grpc.internal.SerializingExecutor.run()V (99 bytes) @ 0x00007f89aa9c6cf8 [0x00007f89aa9c6aa0+0x258]
J 17565% C2 java.util.concurrent.ThreadPoolExecutor.runWorker(Ljava/util/concurrent/ThreadPoolExecutor$Worker;)V (225 bytes) @ 0x00007f89a9f17240 [0x00007f89a9f17080+0x1c0]
j  java.util.concurrent.ThreadPoolExecutor$Worker.run()V+5
j  java.lang.Thread.run()V+11
v  ~StubRoutines::call_stub
V  [libjvm.so+0x695b86]  JavaCalls::call_helper(JavaValue*, methodHandle*, JavaCallArguments*, Thread*)+0x1056
V  [libjvm.so+0x696091]  JavaCalls::call_virtual(JavaValue*, KlassHandle, Symbol*, Symbol*, JavaCallArguments*, Thread*)+0x321
V  [libjvm.so+0x696537]  JavaCalls::call_virtual(JavaValue*, Handle, KlassHandle, Symbol*, Symbol*, Thread*)+0x47
V  [libjvm.so+0x731d60]  thread_entry(JavaThread*, Thread*)+0xa0
V  [libjvm.so+0xa7ede3]  JavaThread::thread_main_inner()+0x103
V  [libjvm.so+0xa7ef2c]  JavaThread::run()+0x11c
V  [libjvm.so+0x92dcc8]  java_start(Thread*)+0x108
C  [libpthread.so.0+0x74a4]  start_thread+0xc4
```

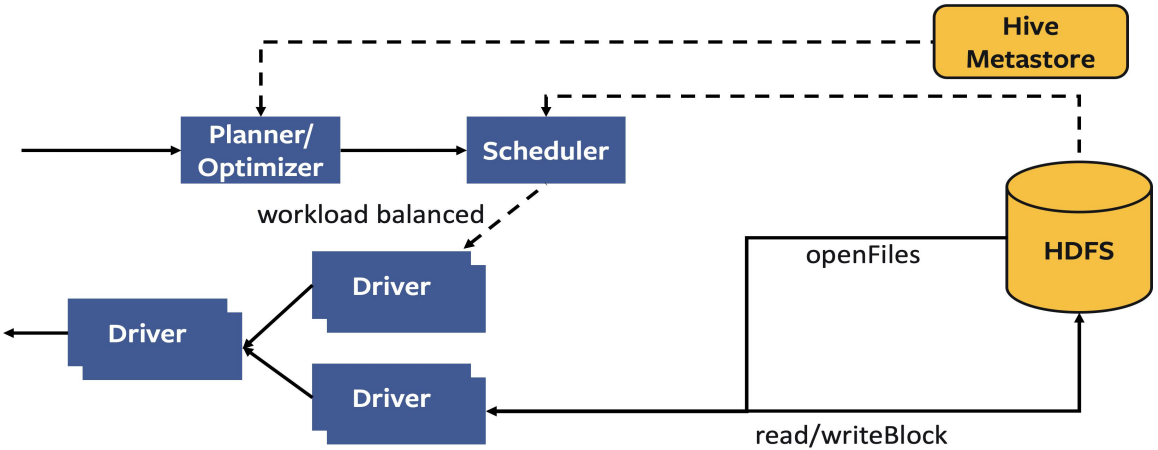
bilibili | DataFun.

05

Presto Local Cache

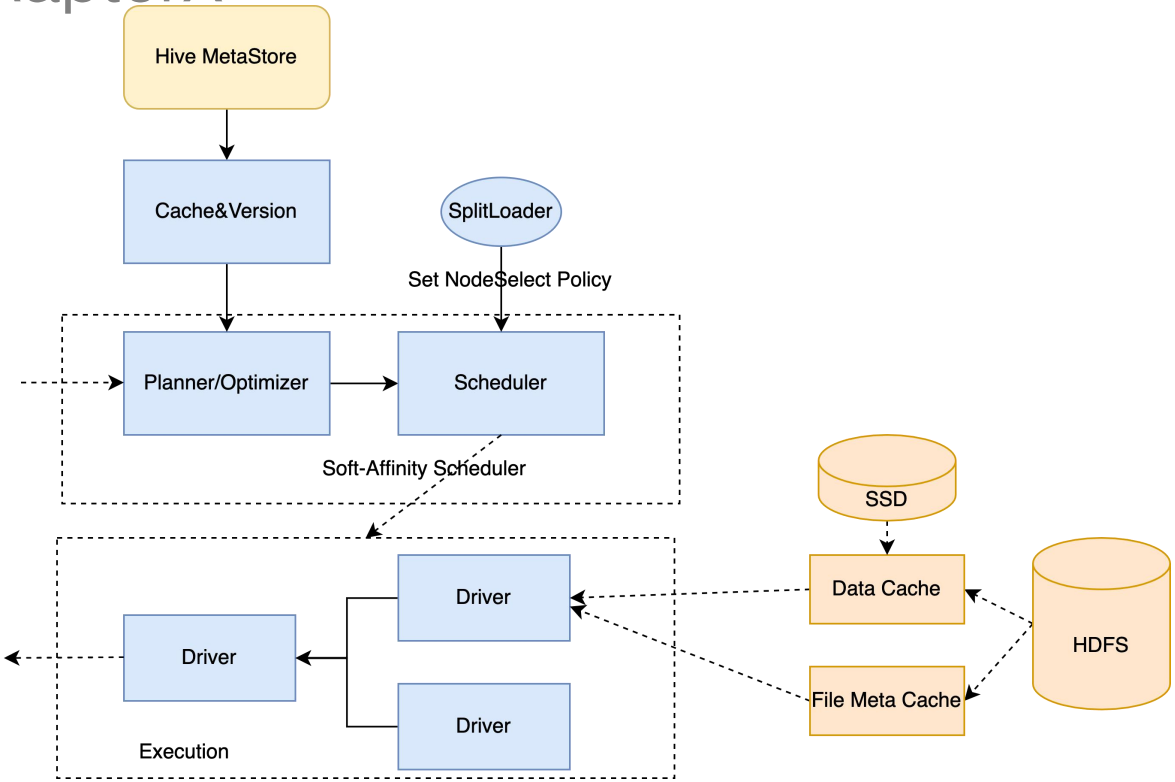
Presto Local Cache-RaptorX背景

- Presto在执行计划阶段需要访问HMS获取表和分区的信息，HMS的响应受单点mysql的吞吐影响，存在慢查询
- Presto在构建split以及读数据的情况下需要访问HDFS。HDFS作为底层存储对接了许多计算引擎，对于RPC请求存在slow rpc情况
- RaptorX应运而生，对元数据与数据源进行全方面缓存



Presto Local Cache-RaptorX

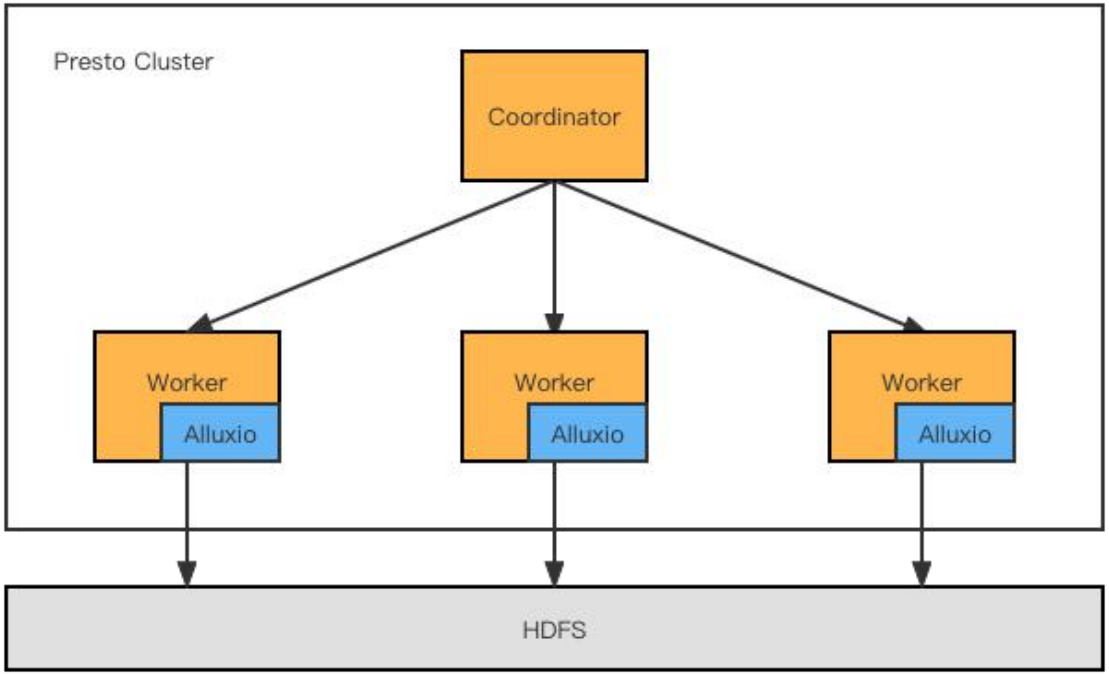
- Hive meta cache
- File List Cache
- Fragment Result Cache
- Orc/Parquet Footer Cache
- Alluxio Data Cache
- Soft Affinity Scheduling



链接：https://prestodb.io/blog/2021/02/04/raptorx

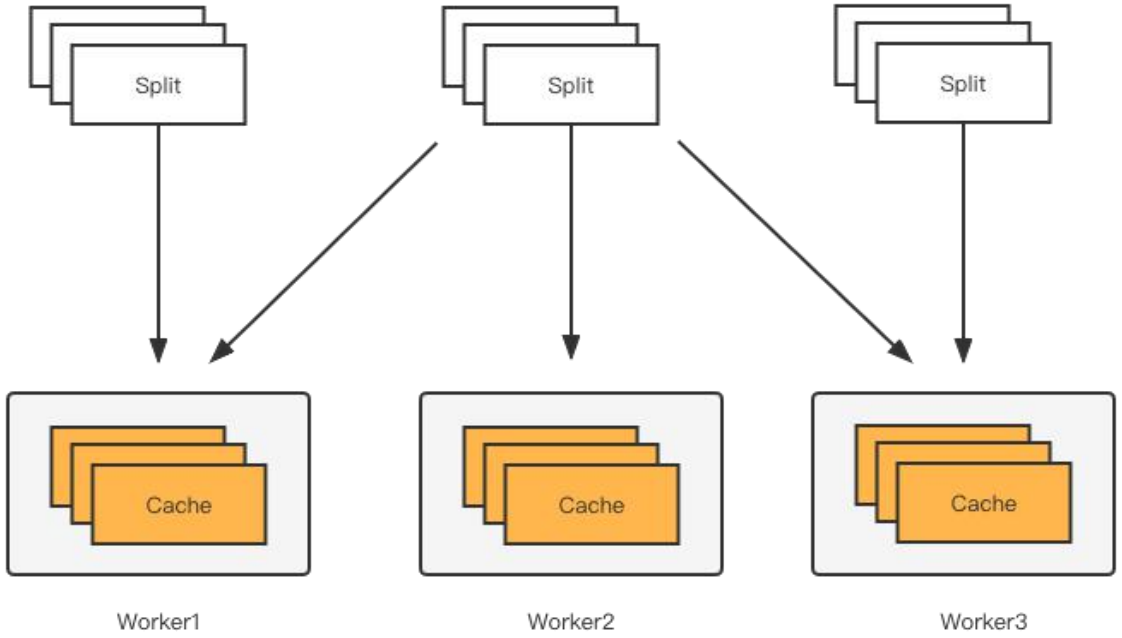
Presto Local Cache-Alluxio Local模式

以jar包的形式嵌入到Presto进程中

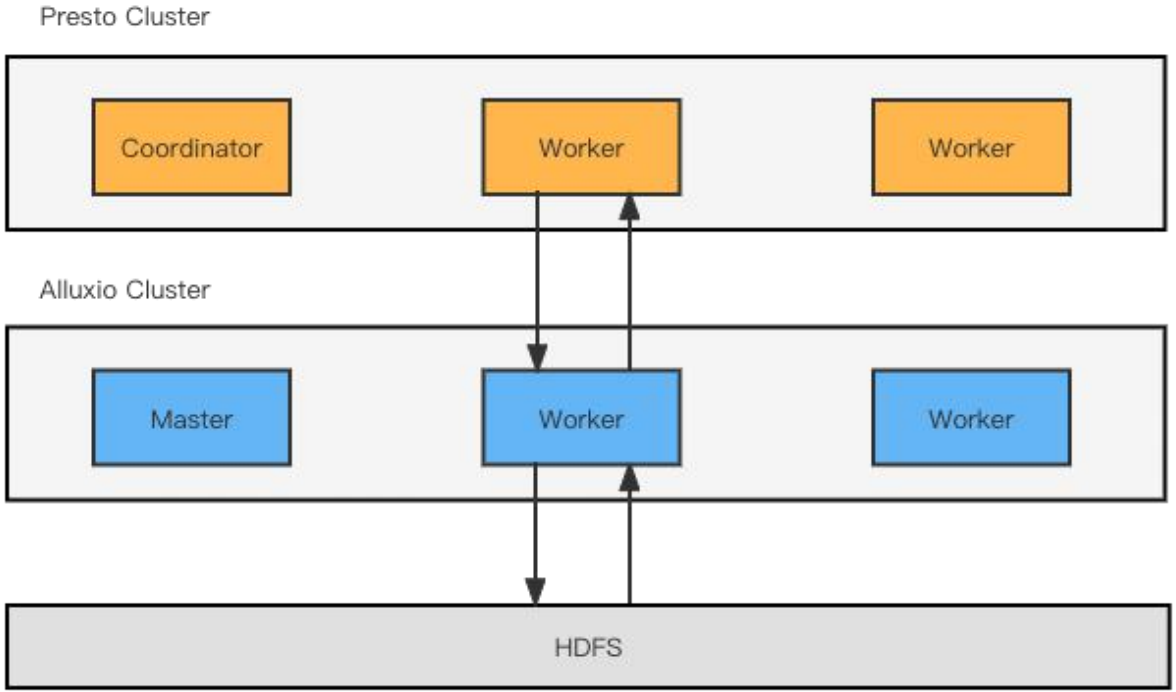


Presto Local Cache-Soft Affinity Scheduling

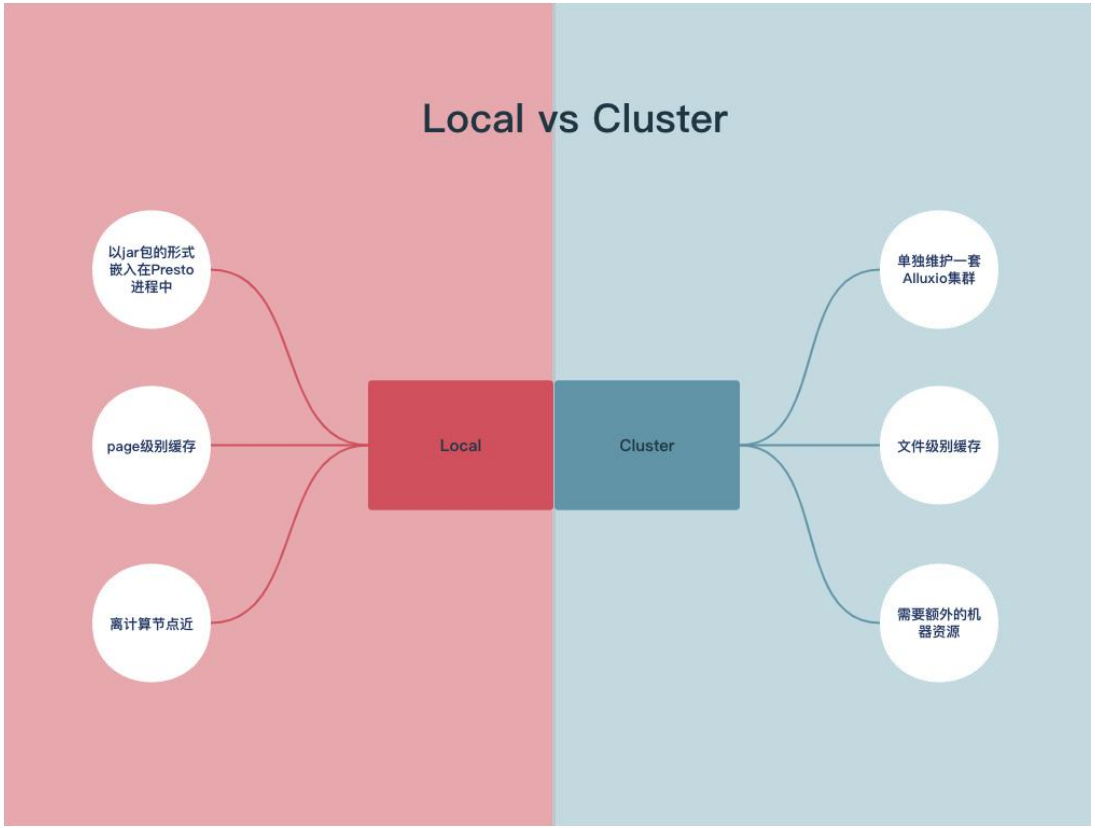
同一个Split尽可能分到同一台worker上（1.hash&mod 2.一致性hash）



Presto Local Cache-Alluxio Cluster



Presto Local Cache-Local vs Cluster



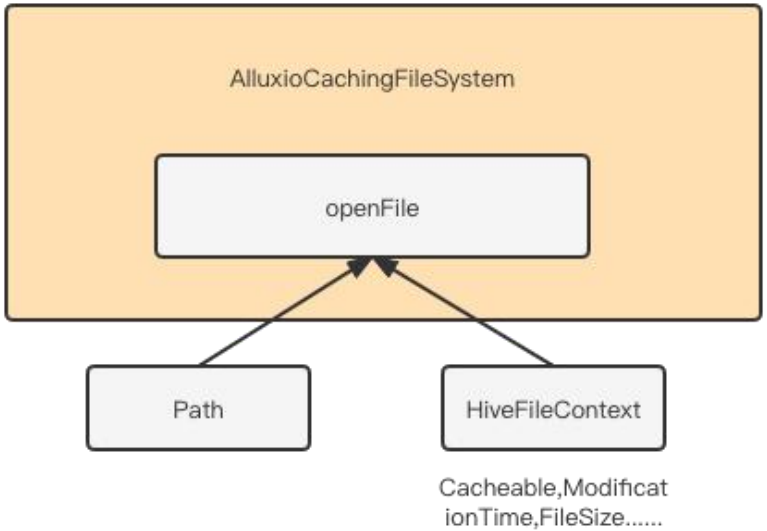
Presto Local Cache-改造点

- Local Cache与底层数据一致性
- Local Cache启动问题
- Local Cache支持hdfs文件系统
- Local Cache支持多磁盘

Presto Local Cache-Local Cache与底层数据一致性

背景:

- 缓存为旧数据（若底层文件变动）
- 影响计算引擎查询结果



思路:

基于文件的LastModifiedTime来判断

Presto端改造:

- 透传文件的LastModifiedTime信息，封装到HiveFileContext中
- 构建pageSource时，将其信息传给本地文件系统中

Presto Local Cache-Local Cache与底层数据一致性

Alluxio社区：

- 社区代码实现了基本的缓存功能
- 社区没对过期数据进行处理

```
old path: root_path/page_size/bucket/file_id/page_1
                                            /page_2
                                            /page_3

new path: root_path/page_size/bucket/file_id/timestamp_1/page_1
                                                        /page_2
                                                        /page_3

new path: root_path/page_size/bucket/file_id/timestamp_2/page_1
                                                        /page_2
                                                        /page_3
```

Alluxio端改造：

- 读数据时校验文件的LastModifiedTime
- 构建内存数据结构，保存文件及时间信息
- 持久化信息（可用于在restore过程中恢复）
- 修改disk存储路径结构

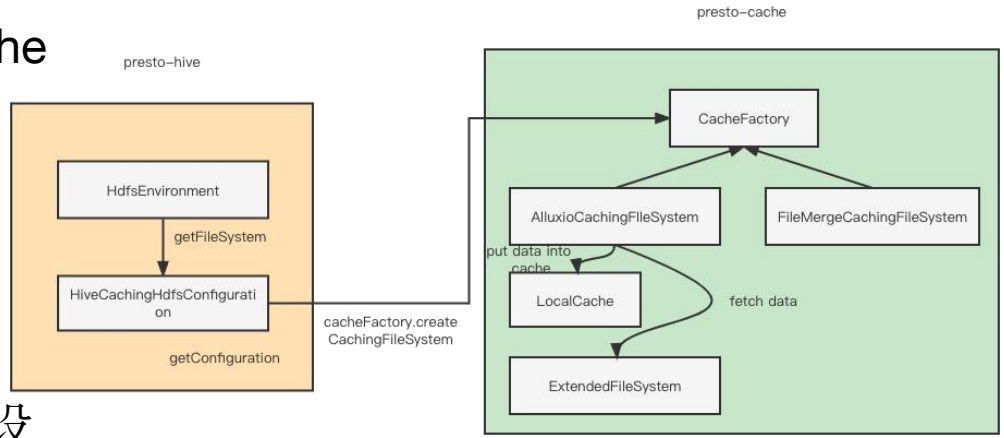
Presto Local Cache-Local Cache启动问题

背景：

- Local Cache restore时间点（Worker启动时）
- Local Cache加载失败时应关闭cache

Presto端改造：

- Worker启动时，主动去获取一次fs（用于加载缓存）
- 当缓存加载失败（如磁盘坏了），设置开关关闭缓存



Presto Local Cache-Local Cache支持hdfs文件系统

背景:

- 社区的外部文件系统要求scheme为alluxio与ws
- 线上环境的data主要走HDFS（Alluxio为辅）

改造:

- Alluxio端添加hdfs的scheme信息
- Alluxio端添加viewfs的scheme信息

```
public class LocalCacheFileSystem extends org.apache.hadoop.fs.FileSystem {
  private static final Logger LOG = LoggerFactory.getLogger(LocalCacheFileSystem.class);
  private static final Set<String> SUPPORTED_FS = new HashSet<String>() {
    {
      add(Constants.SCHEME);
      add("ws");
    }
  };

  ...

  @Override
  public synchronized void initialize(URI uri, org.apache.hadoop.conf.Configuration conf)
      throws IOException {
    if (!SUPPORTED_FS.contains(uri.getScheme())) {
      throw new UnsupportedOperationException(
          uri.getScheme() + " is not supported as the external filesystem.");
    }
    super.initialize(uri, conf);
    mHadoopConf = conf;
    // Set statistics
    setConf(conf);
    mAlluxioConf = HadoopUtils.toAlluxioConf(mHadoopConf);
    // Handle metrics
    Properties metricsProperties = new Properties();
    for (Map.Entry<String, String> entry : conf) {
      metricsProperties.setProperty(entry.getKey(), entry.getValue());
    }
    MetricsSystem.startSinksFromConfig(new MetricsConfig(metricsProperties));
    mCacheManager = CacheManager.Factory.get(mAlluxioConf);
    LocalCacheFileInStream.registerMetrics();
    mCacheFilter = CacheFilter.create(mAlluxioConf);
  }
  ...
}
```

Presto Local Cache-Local Cache支持多磁盘

背景：

- 单个disk空间不足
- 单磁盘io限制

社区：

通过hash&mod的方式存入多磁盘

```
private Path getRoot(PageId pageId) {
  int index = pageId.hashCode() % mRoots.size();
  index = index < 0 ? index + mRoots.size() : index;
  return mRoots.get(index);
}
```

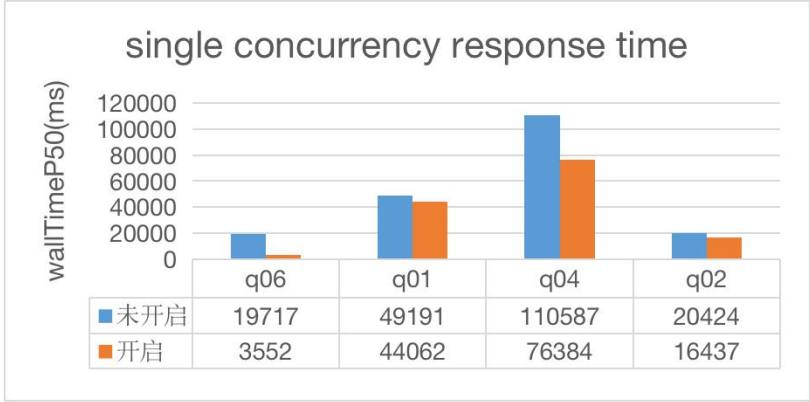
改造：

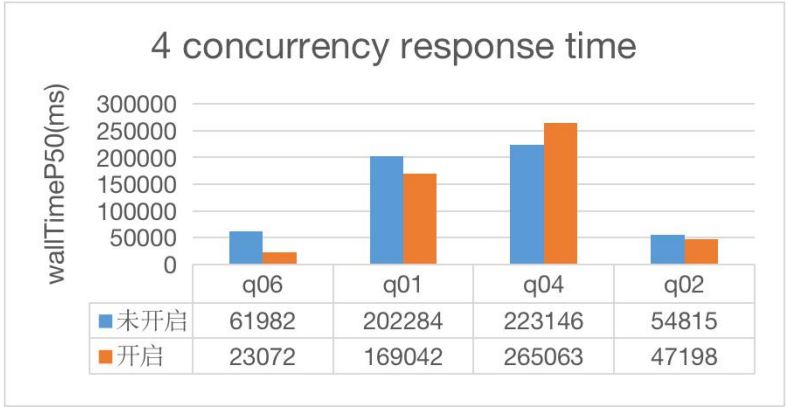
基于AvailableSpace来做磁盘选择（借鉴HDFS）

```
基于可用空间的策略
举例：假设有五个盘，容量分别为1g、50g、25g、5g、30g，
现在需要基于该策略往某个盘写数据。
1）校验5个盘是否处于balanced
最大容量-最小容量<平衡态的阈值（默认10g）
若平衡的话，直接RoundRobin进行选择
2）划分为2列：highAvail与lowAvail
划分标准：判断是否大于（最小容量+平衡态阈值）
highAvail：50g、25g、30g
lowAvail：1g、5g
3）根据概率，选择某列进行RoundRobin
若数据大小超过lowAvail列最大值，则选择highAvail进行轮询
平衡概率值默认为0.75，
0.75选择highAvail轮询
0.25选择lowAvail轮询
```

Presto Local Cache-测试效果

- 单并发场景下，开启local cache缓存可以减少20%左右的查询时间。
- 4并发场景下，开启local cache整体上也有一定的提升。相比单并发情况下，有一定的性能损失。
- 从总体上来看，无论对于简单查询还是复杂查询都能够获得一定的性能提升。

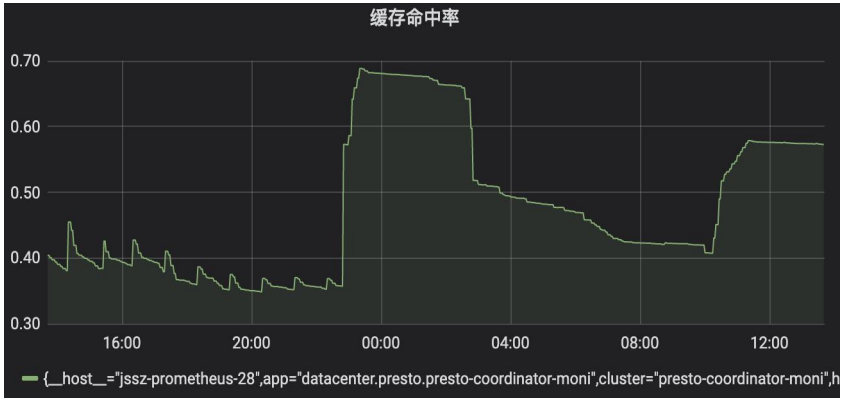




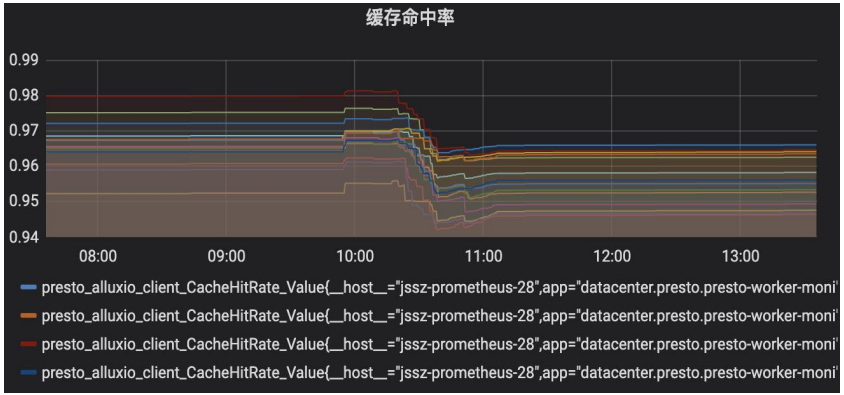
Presto Local Cache-线上效果

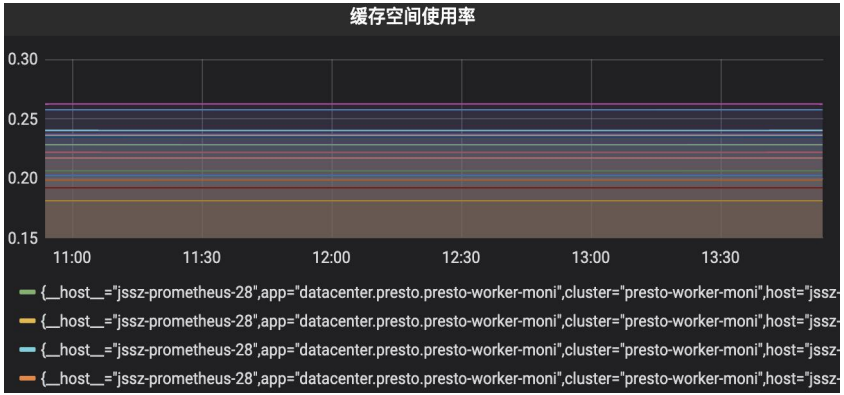
- Local Cache已上线三个Presto集群
- 整体缓存命中率约40%

Coordinator端命中率



Worker端命中率





Presto Local Cache-社区PR

- Get raw filesystem should consider CachingFileSystem (https://github.com/prestodb/presto/pull/17390, Merged)
- Wrapper the input and output stream of HadoopExtendedFileSystem (https://github.com/prestodb/presto/pull/17365, Merged)
- Adapt disable filesystem cache (https://github.com/prestodb/presto/pull/17367, Open)
- Support hdfs and viewfs as the external filesystem (https://github.com/Alluxio/alluxio/pull/15131, Closed)
- Support timely invalidation of parquet metadata cache (https://github.com/prestodb/presto/pull/17500, Merged)

06
后续工作

后续工作

- 推广local模式上线多个集群
- 开发支持textFile格式的缓存
- 开发磁盘检测（若有问题，隔离该节点）
- 改进soft-affinity（用path + start作为key来hash，分散大文件分到单个worker split的压力）
- 改进soft-affinity排除不开启cache的节点

哔哩哔哩技术

bilibili | DataFun.

非常感谢您的观看

bilibili | DataFun.