Gdevops

全球敏捷运维峰会

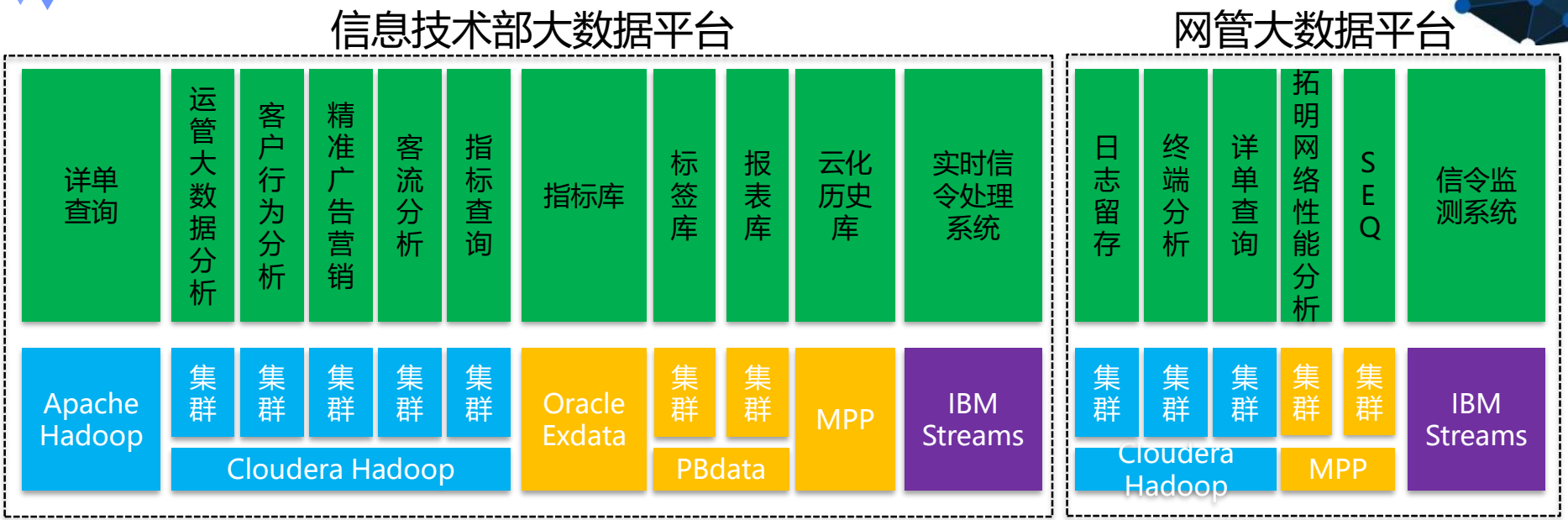
浙江移动大数据平台践行之路

2016年 杭州-北京-广州-上海

演讲人：汤人杰

目录

大数据平台整合之前



随着大数据技术的发展信息技术部和网管中心分别在内部逐步开始试点基于分布式架构的大数据平台技术，目前在局部使用MPP数据库、HADOOP、流处理技术，试运行取得了较好的效果，但是由于缺乏统一的规划和技术演进策略。存在平台重复建设、数据大量冗余，数据质量较低，以及MPP数据库兼容性问题、Hadoop版本不统一、人员不足等问题，严重影响了对大数据的应用，因此亟须在公司层面构建统一的大数据平台，并统一采集数据，向各部门开放，夯实大数据应用的基础。

大数据平台整合之前存在的问题分析

竖井林立、数据冗余

“烟囱”式的系统大量建设，扩展性差，导致重复投资、数据大量冗余，且大数据平台架构整体结构没有统一的规划，缺乏对整体架构演变的长远考虑。

数据标准化、一致性低

数据标准化不足，数据一致性低，导致数据交互复杂，增加运维管控难度

问题

数据采集不全、质量不高

系统数据库结构差异，导致数据迁移中存在数据丢失、重复、错误等问题，导致数据可用性差

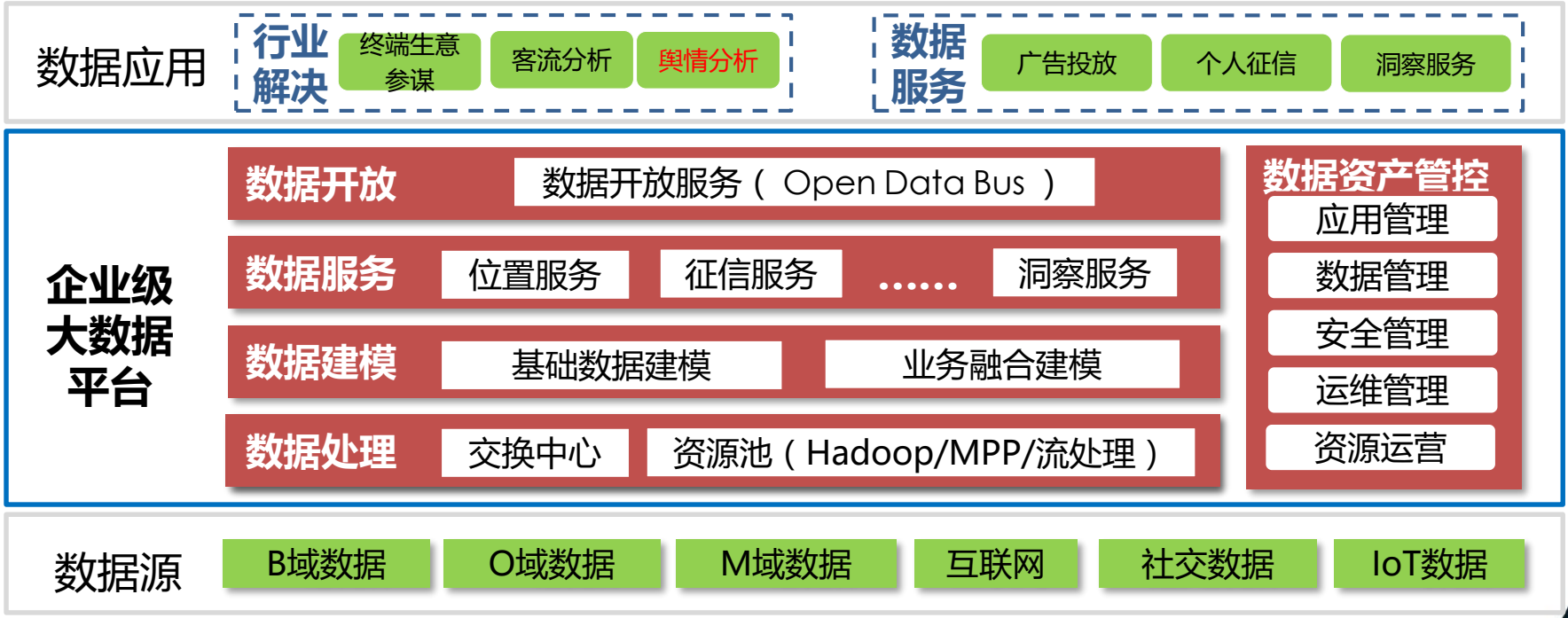
应用创新不足

缺乏核心能力导致大数据应用创新难度增加，创新产品容易被复制，不能灵活应对市场需求，竞争压力加大

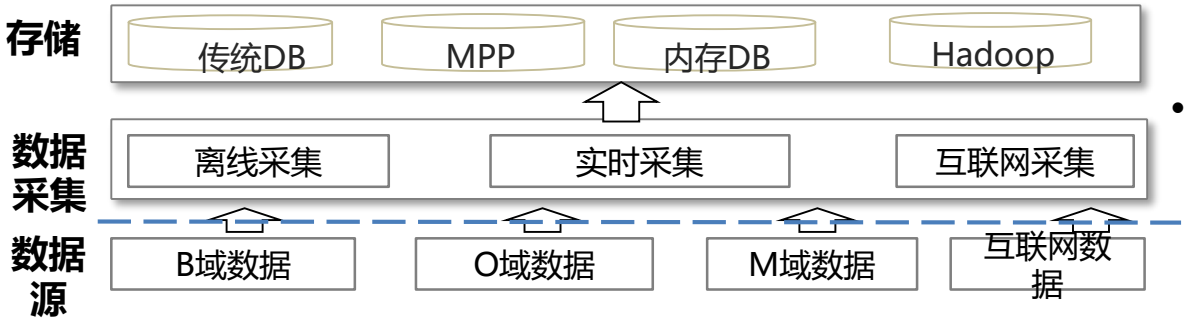
目录

企业级大数据平台规划思路

构建“数据整合、能力共享、应用创新”的企业级大数据平台，对各域数据实现资产化的统一管理，进行持续的业务创新、运营提升、管理优化，推动开放共享，助力数字化服务曲线的发展。



实现数据统一汇聚、建模



- 进行组织和管理转型，组建模型设计开发团队，进行数据建模实践，掌握企业级基础模型，实现对核心能力的掌控。
- 对B、O、M三域数据进行了梳理，分层分类构建统一数据模型，实现企业数据模型标准化和一致性。
- 基于Hadoop云化数据处理平台，实现各域数据的统一汇聚，形成融合的统一数据模型。

实现数据资产化管理

以资产的视角来管理数据，实现数据资产的可查，可看，可用，最终实现大数据资产管理核心能力的全面掌控。

数据资产管理

资产规划

- 规划企业数据资产目录、分类体系及数据部署架构，定期进行盘点和发布

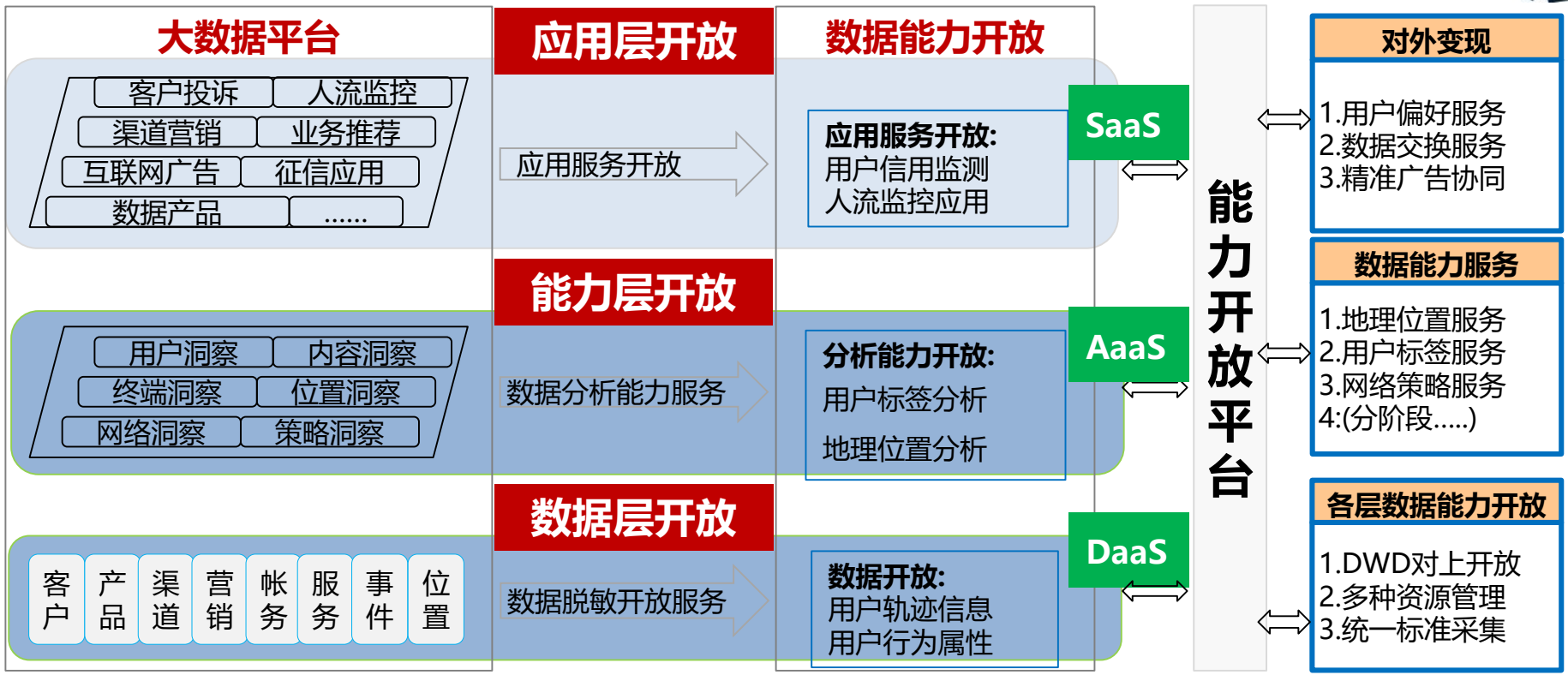
体系建立

- 构建企业数据常态化运营管理机制，在源数据发生新增、变更时，实现数据的快速引入和更新

平台化管理

- 构建数据资产管理平台，实现数据从引入、处理、发布、稽核及修正的平台化管理，确保大数据资产的应用效率

实现数据统一开放共享


大数据平台
客户投诉
人流监控
渠道营销
业务推荐
互联网广告
征信应用
数据产品
.....
应用层开放
应用服务开放
数据能力开放
应用服务开放:
用户信用监测
人流监控应用
SaaS
能力层开放
用户洞察
内容洞察
终端洞察
位置洞察
网络洞察
策略洞察
数据分析能力服务
分析能力开放:
用户标签分析
地理位置分析
AaaS
数据层开放
客户
产品
渠道
营销
帐务
服务
事件
位置
数据脱敏开放服务
数据开放:
用户轨迹信息
用户行为属性
DaaS
能力开放平台
对外变现
1.用户偏好服务
2.数据交换服务
3.精准广告协同
数据能力服务
1.地理位置服务
2.用户标签服务
3.网络策略服务
4:(分阶段.....)
各层数据能力开放
1.DWD对上开放
2.多种资源管理
3.统一标准采集

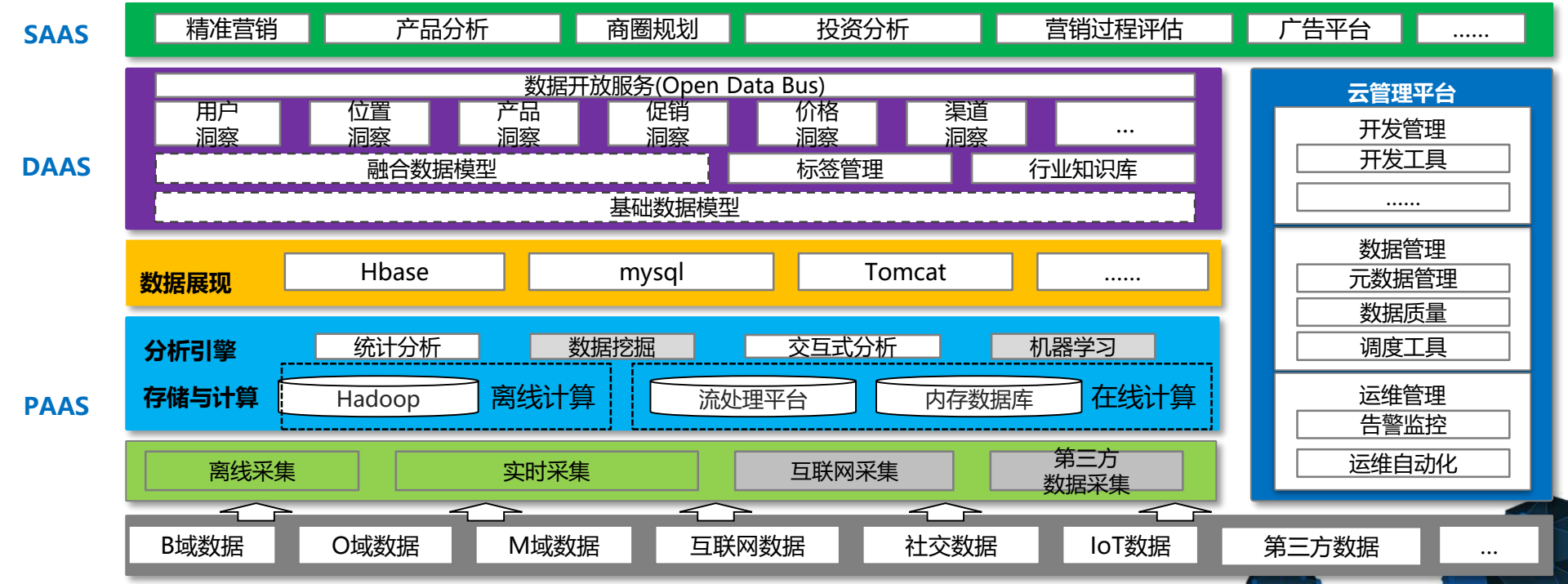
企业级大数据平台功能逻辑架构

企业级大数据平台作为云平台的一部分，主要可以分为三大技术能力：PaaS、DaaS、SaaS。

PaaS：主要包括数据基础服务能力（数据采集、数据存储计算和数据交互等）和平台管理能力（多租户管理、权限管理、安全管理、资源管理、负载管理、配额管理以及计量管理等）、应用开放能力（开发者管理门户、统一调度平台等）；

DaaS：主要包括数据建模、数据质量管理、数据安全管理、数据资产管理、元数据管理等；

SaaS：应用涵盖企业内、外部，包括报表、查询、统计、分析、挖掘，对外价值变现等。。



企业级大数据平台目标架构

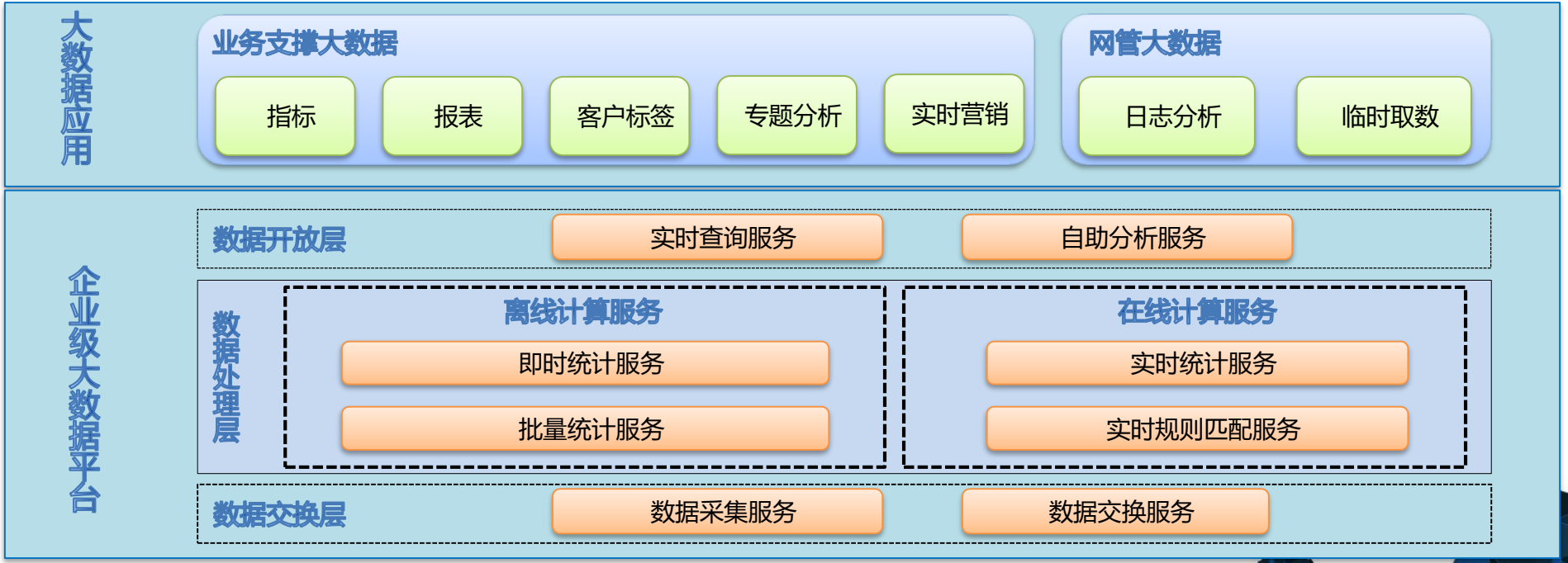
将目前竖井式的经分技术平台重构为一个云化的开放的分布式架构的企业级云化大数据平台。平台将分为数据开放层、数据处理层、数据交换层，为上层应用提供各类大数据的基础云服务，有力支撑上层各类大数据应用的百花齐放。

企业级云化大数据平台对外提供3大服务：数据交换服务、数据处理服务、数据开放服务。

数据交换层：建立统一数据采集交换中心，提供数据采集服务、数据交换服务，实现移动信息生态圈数据共享与交换。

数据处理层：建立数据处理中心，提供离线计算服务和在线计算服务，实现海量数据批处理和实时处理。

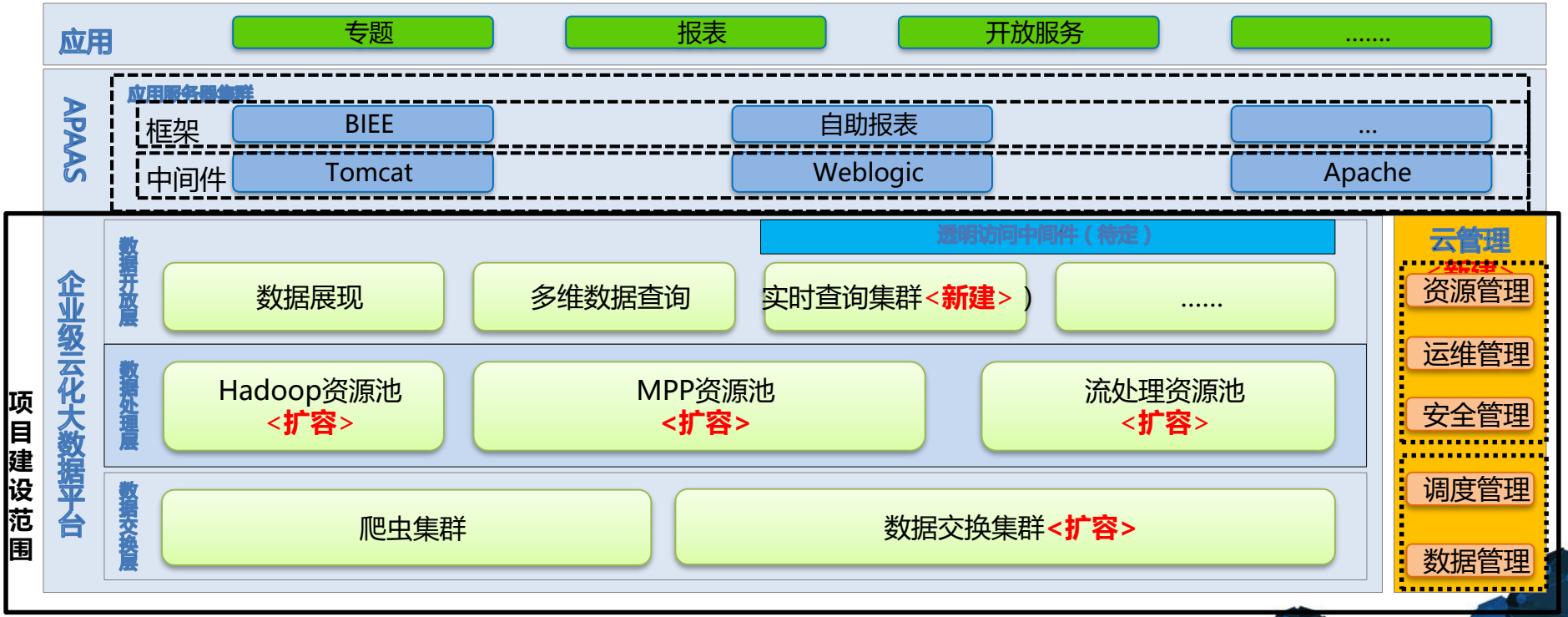
数据开放层：实现海量数据实时查询、多维度挖掘分析，实现大数据变现。



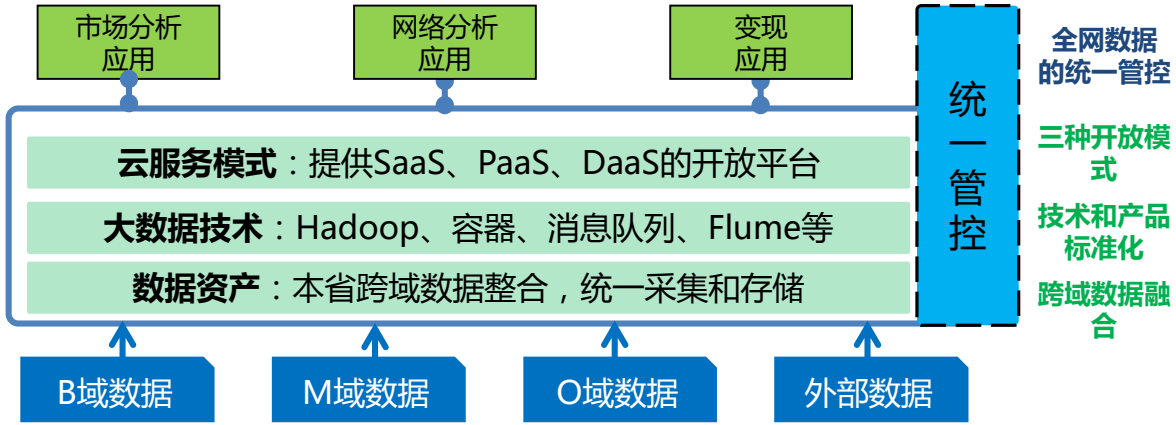
企业级云化大数据平台建设范围与成果

浙江移动大数据平台主要成果：

- 在运营商里面首次实现基于容器技术进行资源隔离的数据中心级资源调度；
- 在运营商里面首次在各种大数据技术组件中全面的字段级的数据隔离；
- 全面实现SaaS、PaaS、DaaS三个层面的平台开放能力。
- “动态人员流量大数据分析平台”等大数据应用已经过世界互联网大会的考验，得到省委领导认可。



企业级大数据平台成果与亮点



全网数据的统一管控

三种开放模式

技术和产品标准化

跨域数据融合

平台亮点

- 在运营商里面**首次**实现基于容器技术进行资源隔离的数据中心级资源调度；
- 在运营商里面**首次**在各种大数据技术组件中全面的字段级的数据隔离；
- **首次**实现浙江移动三域数据的大融合，大大提升了数据质量，有效支撑多维数据分析；
- **全面实现**SaaS、PaaS、DaaS三个层面的平台开放能力。

平台优势分析

- **全面PaaS云化**：企业各应用通过租户模式，按需申请资源，各租户通过容器技术实现资源深度隔离，并结合分布式调度技术，实现动态扩缩容，最大化合理利用资源；
- **应用和能力解耦、应用与数据解耦、应用与平台解耦：**降低建设成本和应用集成复杂度，提高软件质量；
- **资源统一调度**：使用YARN集群资源管理框架，实现对每个租户的容器级动态扩缩容，从而实现集群整体的资源弹性调度；
- **基于容器的资源隔离技术**：采用YARN + 容器方式实现对容器资源的管控，通过多租户机制进行权限控制和资源隔离；
- **统一数据交换服务**：通过引进成熟的大数据处理技术，调整系统架构，对数据进行集中采集、清洗、交换、加载，同时建立统一接口管控，实现接口的可管、可控、可用，为业务提升与创新奠定基础；实现O、B、M三域数据融合，完成企业内部数据整合，实现数据清洗的前移，实现数据和应用解耦目标，形成面对企业内外提供数据交换的能力。

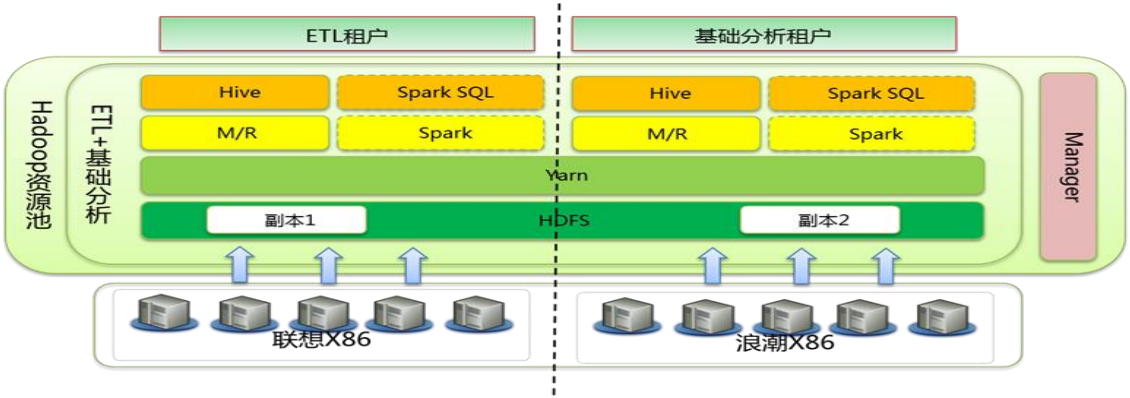
目录

大数据平台设计优化（一）

设计优化二：存储共享计算隔离

隐患问题描述：ETL、基础分析库集群需要由两个不同的部门同时使用，并且需要共享数据。两个部门的业务运行要互不干扰。

- **问题分析:**
  - 由于需要共享数据，所以只能部署一套Hadoop集群，数据存放在同一个HDFS中，不同部门的计算业务使用Yarn做资源管理和调度，使用不同的任务队列，并限定队列的容量。但是这样做仍然会出现两个部门的作业运行到一台物理机上的情况，无法保证作业的互不干扰。
- **问题解决:**
  - 完全计算隔离的实现采用yarn的标签调度策略（Label based scheduling），该调度策略适用于异构集群。通过该策略，将集群划分为不同的资源池，并打上不同的标签，资源池内按队列划分，同时将主机划分到不同的资源池中。如下图。



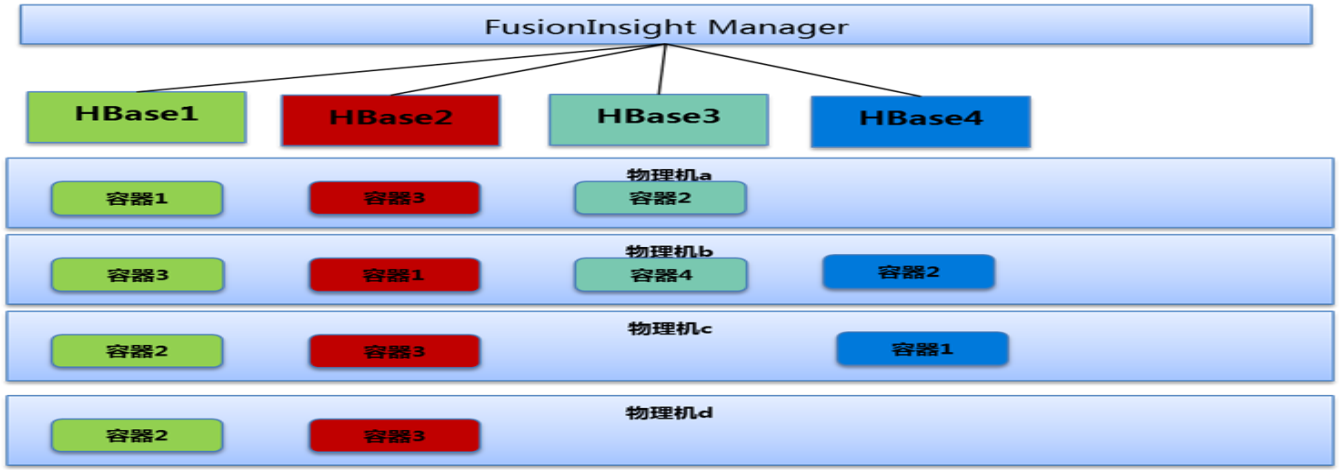
  - 将一个Hadoop集群的计算节点划分为多个的resource pool, 一个计算节点只能属于一个resource pool. 通过引入标签调度能力将不同部门的Yarn 任务队列绑定到不同的resource pool。任务队列中的作业只能在绑定的resource pool节点内运行。 这样部门之间的计算任务就完全物理隔离了，保证互不干扰。

大数据平台设计优化（二）

设计优化三：实时查询库Hbase多实例

隐患问题描述：实时查询库有多个不同的应用，应用之间无数据共享，对于请求的响应时间非常高，需要达到毫秒级别。应用之间要求资源隔离互不影响。

- **问题分析:**
  - 在之前的实践中，每个应用部署一套Hbase集群, 三个应用就需要三套集群，带来极大的维护成本，而且集群的利用率非常低。
- **问题解决:**
  - 在一个Hadoop集群中支持部署HBase多实例，每个上层应用对应一个HBase服务实例。服务实例之间的资源通过cgroup等机制进行控制和隔离，保证每个服务实例的SLA，实现了各Hbase实例之间的资源隔离，而且每个服务实例的资源还可以动态调整，极大的提高了集群的利用率，降低了维护成本。如下图。

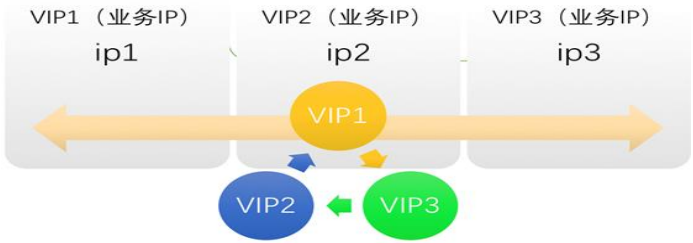


大数据平台问题及解决方案（一）

问题一：MPP集群装载机高可用实现

隐患问题描述：目前GBase集群中的三台加载机为三个独立的点，三台机器创建了相同的目录，部署了相同的应用，每台机器人为分配不同的作业，相当于人为实现负载均衡，但是一旦某个点宕机，此节点的作业就被迫停掉。 换句话说，加载机无高可用。

- **问题分析:**
  - 考虑到节点间的高可用问题，做出以下调整。
- **问题解决:**
  - 三台加载机通过赛门铁克高可用软件实现三方互备，以VIP的方式实现业务漂移，可以做到在节点宕机时做到应用无感知迁移。改造后，三台加载机最可实现2台宕机不影响生产（处理能力会有相应下降）。



问题二：多线程模式下sqlfire的数据导入速率慢

隐患问题描述：在进行sqlfire压力测试时发现，向sqlfire中导入1W条记录需要7.5s的时间，这对于内存数据库来说是慢的。

- **问题分析:**
  - 测试建立的表为分区表，sqlfire支持分区表和复制表两种表模式，分区表按照建表时指定的分区字段分区，复制表则在sqlfire集群每个节点都存有一份数据。一般大表适合建立为分区表，浙江移动场景下，比如客户信息表，产品订购表等大表适合建立为分区表，小表比如产品配置表，地区信息表，套餐维度表等一些维表更适合建立为复制表。建立的测试表为用户信息表，包括用户id，地市，县市，年龄，入网时间等11个字段信息。
  - 首先测试单线程模式下的sqlfire的导入数据能力，以插入1W条记录为例，测试结果耗时为7.5秒。
  - 然后进行多线程模式下测试1W记录做insert操作，以开20个线程为例，以上测试结果1s。
  - 程序中开20个线程，每500条记录开一个线程做insert操作。在这种情况下测试10W条记录的插入，耗时为4s，相当于每5000条记录开一个线程做insert操作，进一步可以使用线程池来进行线程的管理。
- **问题解决:**
  - 在一定线程数范围内提高线程数，可以明显的提高内存数据库sqlfire的导入数据的速率，但对于多线程模式来说，存在的瓶颈就是当线程数达到一定的数量后，对于一定的硬件条件下，提高线程数据对导入数据速率并无明显的提高，这是因为线程数达到一定数量后，线程间的线程切换也是一个较大的开销。

大数据平台问题及解决方案（二）

问题三：GBase集群并发DDL慢

隐患问题描述：业务人员反映有时候create table、drop table、truncate table慢，有时候一个简单的操作需要等上几十分钟、甚至几个小时都无法完成，在正常的时候几秒钟就会结束。平台维护侧通过后台观察，系统资源空闲，数据库状态正常，并发回话数低（10个以内），会话状态为check permission，等待权限验证。

- **问题分析:**
  - 权限验证过程理论上在每一个数据库都存在，用以确认该用户对该库的改对象有没有该操作权限，属于正常操作。操作的本质是查询几张相关元数据表，理应属于瞬间完成过程，不到10个会话，在验证权限时间上不应该如此长，面对这个问题，我们平台运维人员一度认为是Gbase产品元数据管理存在性能差的缺陷，原厂人员段时间也给不出解决方案。后来随着对日志不断分析，我们逐渐定位到锁问题，但是按照正常数据库理论，排他表锁最多影响到自己，在我们遇到的现象中，多个会话中是对多个不同的表进行的操作，似乎用锁解释不通，而且即使要这么解释，也无法拿出解决方案。直到应用人员的另一个反馈出现，我们才猜到了问题的可能因素，通过模拟场景实验，最终确定是DDL锁在作怪。
  - 解释DDL锁之前，先详细说明一下我们的猜想依据和模拟过程。业务人员的另一个反馈是并发drop表时间约呈倍数增长，通过这一现象，再联系之前发现的问题，我们开始猜测，Gbase中存在一种可以跨越不同对象、影响DDL并发的锁。
  - 在之前的向原厂的问题集中反馈中，我们曾提到过check permission问题，当时原厂给出的其中一条建议是建表过程加‘nolock’参数，但是对于该参数的描述模棱两可，仅仅解释为对数据库有利。
  - 总结：建表过程中未加‘nolock’参数的表，其DDL操作会影响到所有未加‘nolock’的DDL操作，不会影响添加‘nolock’参数的所有DDL操作
  - 总结：建表过程中添加nolock参数的表，其DDL操作不会阻塞任何DDL会话。
  - 以上弄清楚了产生DDL等待的原因，那么如何解决这一问题，经过与原厂沟通，DDL LOCK是GBase数据库开发初期一个重要功能，后来大版本变更后，这个锁基本没有正面功能，但是数据库默认参数建表都是加lock的。算是Gbase数据库一个历史遗留bug，计划在下个大版本中取消。
- **问题解决:**
  - 首先要说明：这个默认参数无法修改，也就是说我们无法通过设定数据库参数来使后续的建表全部变成nolock表。所以唯一的解决方案就是修改建表模版，添加nolock参数
  - 例如创建hash分区表创建语句：

```
CREATE  TABLE "t1" (
"a" int DEFAULT NULL,
"b" varchar(10) DEFAULT NULL
)DISTRIBUTED BY('a') nolock;
```

大数据平台问题及解决方案（三）

Gdevops

问题四：GBase MPP 执行insert select 性能慢

隐患问题描述： 在MPP集市集群上线调试过程中，大量的业务场景使用insert select进行中间数据的处理，业务反馈insert select的性能较差，某条语句相对于DB2的40s有几倍的性能差距。

- **问题分析:**
  - 1、验证是否所有的insert select性能都慢： 当前测试表中存在着一百多列，选择前15列新建表test2进行验证测试，表test2的inset select 性能为7s左右，可以判断不是insert select语法支持上导致的问题。按照Gbase MPP列存属性，列数的增加对插入性能的影响，整个插入过程最多在46s (100列/15列*7)。
  - 2、分析大字段列对插入性能的影响：新建表test3，表结构中增加五个varchar(1024)的列，验证大字段对性能的影响。Test3的Insert select性能为70s，五个字段增加了63s，平均一个大字段12s，表结构中有七个字段1024字节、基本上就耗用了80s左右。
  - 3、修改数据类型的宽度： 新建测试表结构test4，将其中字段宽度为1024修改成100，并保证所有数据均能成功插入。Test4的insert select的性能为7.5s，较15个字段的插入性能仅仅多了0.5s。
  - 4、执行计划分析： 从两组字段宽度为100和1024的执行计划可以看出，当字段宽度为1024时，insert阶段的执行时间存在较大的等待时间。
  - 由此可以分析，GBase MPP执行insert select性能差是由于对于字符串列定义较长时，在insert处理时出现了某些限制，导致插入性能差

- **问题解决:**
  - 1、通过分析是参数和字符串型字段内存申请的管理有关： 如果字符串列定义比较长，而_gbase_insert_malloc_size_limit较少，数据库评估不能一次性存储太多内容，会每行申请一次内存。而调大该参数值，可以批量多行申请内存，缩短时间。默认值1024，最大值32768。
  - 2、session级进行解决方法验证
    1、查看各个节点的参数值
    2、修改gnode上_gbase_insert_malloc_size_limit的参数值
    3、查看各个节点上的_gbase_insert_malloc_size_limit，已经修改为32768
    4、重新执行第一条sql语句，执行时间为22s。
  - 3、配置文件修改
    在GBase MPP每个节点的配置文件/opt/gnode/config/gbase_8a_gbase.cnf加入参数
    gbase_insert_malloc_size_limit = 32768后，重启集群

大数据平台问题及解决方案（四）

问题五：IBM Streams与Kafka连接

隐患问题描述：1）IBM Streams与Kafka连通时发现，streams与kafka并不能连通；2）IBM Streams 在与Kafka读写时发现性能不到1万条每秒，这远远没有达到我们设计之初的要求。

- **问题分析:**
  - 通过查阅文档发现，streams确实存在于kafka连通的接口，进一步查看kafka代码发现，原来kafka本身存在缺乏安全机制，为了解决这个问题，我们在kafka中间层上加入了kerberos安全认证，所以streams在连接kafka时没有进行kerberos的安全认证，从而导致stream与kafka不能连接。
  - 针对读写性能问题，尝试使用多线程，并使性能达到100万条每秒。
- **问题解决:**
  - **streams加入安全认证的部分代码如下：**

```
private static final String KAFKA_JAAS_POSTFIX = ".kafka.jaas.conf";
/*** 用户自己申请的机机账号keytab 文件名称*/
private static final String USER_KEYTAB_FILE = "user.keytab";
/*** 用户自己申请的机机账号名称*/
private static final String USER_PRINCIPAL = "nuodong";
public static void securityPrepare() throws IOException
```

  - **多线程以下是部分代码：**

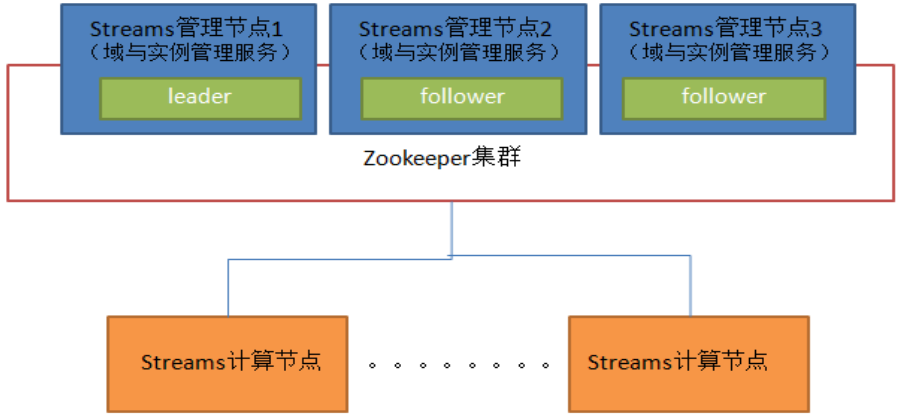
```
public class ConsumerMultThread extends Thread
{
  private static Logger LOG = Logger.getLogger(ConsumerMultThread.class);
  private static int CONCURRENCY_THREAD_NUM = 100; // 并发的线程数
  private ConsumerConnector consumer;
```

大数据平台问题及解决方案（五）

问题六：IBM Streams的zookeeper问题

隐患问题描述：集群搭建好后，我们发现streams集群的处理性能不尽人意，而且响应时间也比较长，同时命令提交后达到10秒以上，与之前的测试情况有很大的差别（测试时大概3秒左右）。

- **问题分析:**
  - 集群搭建好后，我们发现streams集群的处理性能不尽人意，而且响应时间也比较长，同时命令提交后达到10秒以上，与之前的测试情况有很大的差别（测试时大概3秒左右），为此我们猜测可能是网络问题，会不会出现网络丢包延时等错误导致的?后经查询，网络这一块并没有问题，后来才发现是zookeeper问题，经分析发现，由于和ambari共用zookeeper集群，导致zk会有额外的网络开销，同时zk要负担两个集群的通信，从而导致响应时间变长，同时tps也越低。
- **问题解决:**
  - 通过优化Streams的集群架构，并重新分离出zookeeper集群，在我们streams管理节点本地搭建zookeeper，有效解决了上述问题。

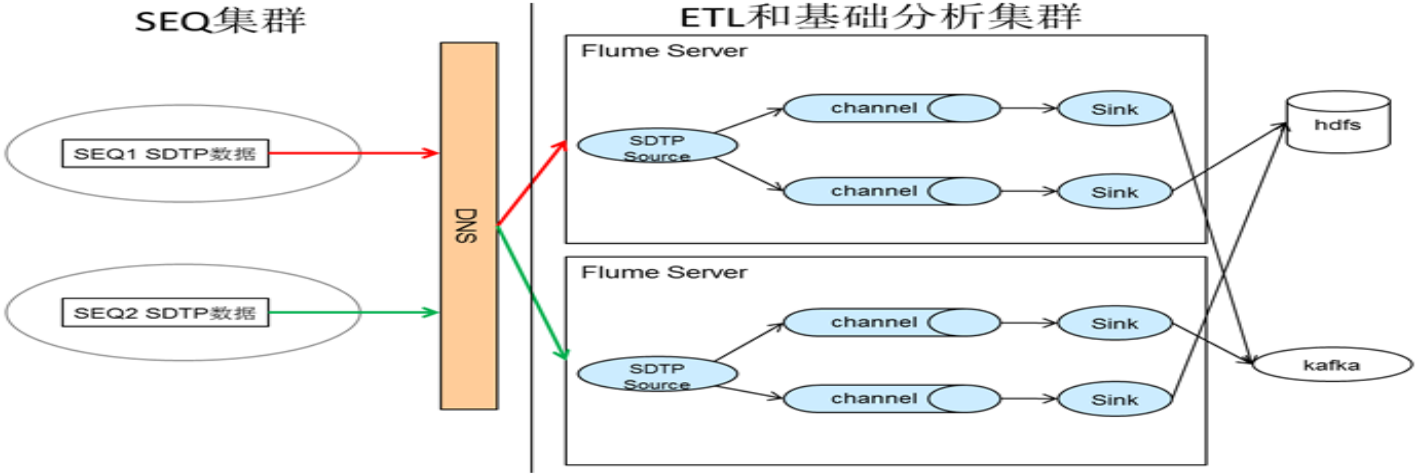


大数据平台问题及解决方案（六）

问题七：Flume高可用

隐患问题描述：目前Flume是非集群模式的，存在单点故障的风险。在生产环境下如何保障Flume可靠性，即使在某个Flume节点down掉之后依然能保证正常接收数据、业务不受影响。

- **问题分析:**
  - 针对此问题提出采用DNS轮询方式，在SEQ侧通过域名方式连接flume节点，当一台flume节点down掉之后，会自动连接其他flume节点，保证业务连续性。
- **问题解决:**
  - DNS轮询方式就是指将相同的域名解析到不同的IP上，并随机使用其中某台主机的技术。在SEQ节点上配置DNS服务后，每次SEQ节点访问Flume节点都需要一次DNS解析，然后选取可用的主机节点。这里又配置了NSCD服务（NSCD服务就是能实现DNS缓存，其可以在本地缓存DNS解析的结果来加快DNS解析的速度）。具体框架如下图。



大数据平台问题及解决方案（七）

Gdevops

问题八：Flume消费kafka连接ZK超时问题

隐患问题描述：基础分析库接收网管侧SEQ数据HTTP_COMPS_LAYER类型数据时，出现丢失，少了某些时段数据。数据的接入方式是通过Flume消费Kafka内的数据，再传到HDFS上。

- **问题分析:**
  - 1、ZK连接timeout设置太小

通过查找日志，发现有很多session expired的报错。在丢失数据的时间段内，共出现了高达576次。分析发现，Kafka引用的第三方 zk-Client.jar存在在某些异常场景下对session expired的判断有误的情况。当session timeout配置较小时，session expired会频繁出 现。尤其在网络状况较差的时候，容易出现zk-Client端与zk-Server端关于expired的频繁交互（zk-Client端的上一次的会话响应还未 收到，就收到了session expired的异常，并再次发起建立会话的请求，然后再次超时），此时极易出现极个别的会话出现了在服务端 已经释放并重连成功，客户端却一直认为失败，并不断重连的现象。（即客户端一直重连，但服务端认为连接已经存在报冲突），报错。如下图。

```
2016-02-03 15:58:31,129 | INFO  |
[ZkClient-EventThread-70-pc-zjqbetl11:24002,pc-zjqbetl21:24002,pc-zjqbetl39:24002,pc-zjqbetl49:24002,pc-zjqbetl59:24002/kafka] |  conflict in
/consumers/O_PS_SEQ_HTTP_COMPS_LAYER_01/ids/O_PS_SEQ_HTTP_COMPS_LAYER_01_pc-zjqbetl86-1454482884879-82011cf4 data:
{"version":1,"subscription":{"O_PS_SEQ_HTTP_COMPS_LAYER":1},"pattern":"static","timestamp":"1454486311128"} stored data:
{"version":1,"subscription":{"O_PS_SEQ_HTTP_COMPS_LAYER":1},"pattern":"static","timestamp":"14544 86310887"}  |
kafka.utils.Logging$class.info(Logging.scala:68)
2016-02-03 15:58:31,130 | INFO  |
[ZkClient-EventThread-70-pc-zjqbetl11:24002,pc-zjqbetl21:24002,pc-zjqbetl39:24002,pc-zjqbetl49:24002,pc-zjqbetl59:24002/kafka] |  I wrote this conflicted
ephemeral node [{"version":1,"subscription":{"O_PS_SEQ_HTTP_COMPS_LAYER":1},"pattern":"static","timestamp":"1454486311128"}] at
/consumers/O_PS_SEQ_HTTP_COMPS_LAYER_01/ids/O_PS_SEQ_HTTP_COMPS_LAYER_01_pc-zjqbetl86-1454482884879-82011cf4 a while back in a different session, hence I
will backoff for this node to be deleted by Zookeeper and retry  | kafka.utils.Logging$class.info(Logging.scala:68)
```

Flume作为消费者要消费Kafka集群中topic中的数据需要连接Zookeeper服务,由于此时网络较差Flume作为客户端与Zk服务端交互不断超时重连影响了Flume节点的消费，当网络状况好转时客户端再次重连成功便可以继续消费。

  - 2、Flume GC参数配置不合理

由于业务场景下对flume的性能有较高的要求，flume采用了memory channel传输数据，再加上业务较多，每个节点flume都配置了 20个左右的传输通道，所以服务需要较大的内存。通过排查flume的GC配置发现，flume的GC配置的young区设置太大，共有60G的 堆栈空间，其中young区占了45G，这会导致GC时间过长，一旦发生GC，因所需时间较长，导致超时的可能性较大。

观察日志发现flume的GC时间长达10s左右，而kafka的session timeout默认设置为6s。由于在GC这段时间所有其他线程会被暂时挂 起,这里GC时间长达10s而超时时间则被设置为6s极易导致Zk服务器认为flume客户端连接已经丢失。

- **问题解决:**
  - 修改GC参数，设置内存young区大小调整为8G；
  - 修改kafka source配置，将超时时间配置延长至90s，配合修改连接zk的重试次数和重启间隔；

大数据平台问题及解决方案（八）

Gdevops

问题九：Hive文件属主问题

隐患问题描述：通过FTP用户A上传到集群的数据，在使用其他用户B进行load到Hive表的过程中，出现权限不足问题。即使给用户B赋该文件的读写权限，也无法load到hive表中，仍然提示权限不足。

- **问题分析:**
  - 华为FI产品Hive使用的认证模型是社区推荐使用的SQL Standard Based Hive Authorization，该权限认证模型要求使用load命令将HDFS文件导入到Hive表时，要求当前用户是待导入文件的owner，或是具有admin privilege（具有admin privilege会直接跳过认证流程，但存在权限扩大化风险，因此未考虑）。
  - 对load操作的用户权限要求代码（org.apache.hadoop.hive.ql.security.authorization.plugin.sqlstd.Operation2Privilege类）的关键代码片段如下：

```
/* Mapping of operation to its required input and output privileges //sql操作对输入资源和输出资源要求具备的权限映射*/
public class Operation2Privilege
private static SQLPrivTypeGrant[] OWNER_INS_SEL_DEL_NOGRANT_AR =
    arr(SQLPrivTypeGrant.OWNER_PRIV, //对资源（数据库、表、hdfs文件）要求具有ownership的权限
    SQLPrivTypeGrant.INSERT_NOGRANT, //对资源要求有insert权限
    SQLPrivTypeGrant.DELETE_NOGRANT, //对资源要求有delete权限
    SQLPrivTypeGrant.SELECT_NOGRANT); //对资源要求有select权限
op2Priv = new HashMap<HiveOperationType, List<PrivRequirement>>(); //保存sql操作与输入和输出资源要求具备的权限映射的Map对象
op2Priv.put(HiveOperationType.LOAD, PrivRequirement.newIOPrivRequirement
(OWNER_INS_SEL_DEL_NOGRANT_AR,  //load操作对输入资源（HDFS文件）要求具备ownership、insert、select、delete权限
arr(SQLPrivTypeGrant.INSERT_NOGRANT, SQLPrivTypeGrant.DELETE_NOGRANT))); //load操作对输出资源（目标表）要求具备insert、delete权限
```

  - 综上，在使用SQL Standard Based Hive Authorization认证模型的前提下，load操作要求当前用户具有待导入文件的ownership权限。
- **问题解决:**
  - 应用在导入Hive表数据之前，需要改一下文件的属主

大数据平台问题及解决方案（九）

问题十：Flume消费kafka句柄泄露

隐患问题描述：浙江大数据平台在从网管接入数据时，前期数据接入传输正常，但是运行一段时间后出现Flume主机无法创建文件，提示too many files opend的问题。从主机层面上发现同时打开文件的数量达到了ulimit文件句柄上限。

- **问题分析:**
  - Flume消费Kafka数据的流程，首先Flume会连接Zookeeper集群，从中获取相应的Broker-Topic注册信息，然后将获取到的Broker在本地的/etc/hosts中解析出相应的IP地址，再通过IP地址去获取对应位置的数据（针对Kafka集群设置通过主机名的连接方式）
  - 网管的Kafka集群设置的即为通过主机名的连接方式，通过查看Flume.log，发现有如下的报错，发现Broker中出现了新的主机名ZHZJ-fe09-hadoop.cmcc无法得到解析，查找不到。较之前的Kafka集群三个Broker相比，这个是新增的。通过检查，发现是网管Kafka集群新增了ZHZJ-fe09-hadoop.cmcc这台Broker，但Flume主机未及时将该台Broker的主机名和IP地址新增到本机的/etc/hosts中，导致新增加的Broker不能正确解析。
  - 而这又触发了Flume依赖的kafka 0.8.1.1 jar包的Bug，最终导致建立了大量的socket连接而没有关闭。具体导致产生kafka 0.8.1.1 jar包Bug的代码如下。

```
try { reconnect()//连接失败，尝试重新连接
      blockingChannel.send(request)
      response = blockingChannel.receive()
    } catch {
      case e: Throwable =>
        disconnect()          //进行错误处理，断链
        throw e  }}
private def disconnect() = {
channel.connect(new InetSocketAddress(host, port)) }      //这是bug，当域名无法解析时，connected是flase，这就导致了
                                                          disconnect函数什么都不做，直接退出且无异常捕获，导致了句柄泄露
```

flume消费kafka句柄泄露详细报告

- **问题解决:**
  - 12.30号，将Flume依赖的kafka 0.8.1.1 jar包升级到0.8.2.1版本，保证即使配错误，也不会导致句柄泄露导致节点异常；
  - 在后续网管端增加broker时，加强双方沟通，得到知会后及时在/etc/hosts下增加主机名和IP地址映射关系；

大数据平台问题及解决方案（十）

Gdevops

问题十一：HDFS磁盘检查机制优化

隐患问题描述：DN所在部分节点，会出现一个磁盘utils占用率持续100%现象，导致HDFS读写速度下降，并在DN日志中有很多slow传输和slow写盘的异常

- **问题分析:**
  - 通过对DN的日志持续的分析，发现有114个DN不定的频率出现“No route to host”异常，频率高的DN出现了117次，导致写Block文件失败，频繁触发了DiskChecker，结果出现磁盘utils上升的情况。DN部分数据盘IO持续10+s 100%，则让client写文件时很慢，最终从业务侧发现查询数据或者写文件都变慢。
  - 可以确定DN数据盘读写很慢的原因是磁盘不停的在做DiskChecker所致，进而影响了整个HDFS读写的效率，降低了平台的处理能力。
  - DiskChecker的存在，是为了解决当Datanode网络或者磁盘异常情况下，HDFS对管理的磁盘做健康检查的线程，最终会将异常磁盘排除，以避免坏磁盘对DN的影响。它的检查机制是进入数据盘每个目录下，创建一个目录，测试磁盘的可用性及读写速率（注意是递归的，会递归数据盘下所有的目录），测试完成后再删除之前创建的目录。但是当时DN在做DiskChecker时，数据盘的目录达到了6.5W个，这样在检查时耗时且执行频繁，对磁盘IO占用、性能消耗非常大，最终导致了磁盘读写变慢，HDFS读写变慢。
  - 优化的关键代码如下

```
void checkDirs() throws DiskErrorException {
DiskChecker.checkDirs(finalizedDir);//删除checkDirs（finalizedDir）方法
DiskChecker.checkDir(finalizedDir);//调用checkDir(finalizedDir)方法
DiskChecker.checkDir(tmpDir);
DiskChecker.checkDir(rbwDir);
}
```

HDFS磁盘检查机制优化详细报告

- **问题解决:**
  - 减少平台业务小文件数量；
  - 合入开源优化补丁HDFS-8845，重启DN （DiskChecker由遍历数据盘整个文件目录树检查磁盘，改为只检查数据盘的根目录

大数据平台问题及解决方案（十一）

Gdevops

问题十二：Flume入Hbase失败

隐患问题描述：20160301网管东信日志数据通过flume把数据导入到实时查询库hbase2实例，导入失败。

- **问题分析:**
  - 集群外的Flume将采集的数据写入HBase时，调用HBase的API出现异常，导致数据写入失败。出现如下错误日志

    Cannot get replica 0 location for {"totalColumns":1,"row":"defaulte41773a4-b196-4c02-8268-5f13985ad255","families":{"f1":[{"qualifier":"pCol","vlen":26,"tag":[],"timestamp":9223372036854775807}]}} | org.apache.hadoop.hbase.client.AsyncProcess$AsyncRequestFutureImpl.manageLocationError(AsyncProcess.java:927)

  - HBase客户端（Flume进程）访问region首先要从hbase:meta表读取该region所在RegionServer的位置（即主机名），而读取hbase:meta表需要先找到hbase:meta表region所在的RegionServer的位置（即主机名），这个位置是从ZK上获取的（ZK上保存的是主机名），HBase的API在上述处理过程中失败了，就打印出上述错误日志信息。代码的调用关系如下：

```
private RegionLocations findAllLocationsOrFail(Action<Row> action, boolean useCache) {
  if (action.getAction() == null) throw new IllegalArgumentException("#" + id +
      ", row cannot be null");
  RegionLocations loc = null;
  try {
    loc = connection.locateRegion(
        tableName, action.getAction().getRow(), useCache, true, action.getReplicaId());
  } catch (IOException ex) {
    manageLocationError(action, ex);
  }
  return loc;
}
```

这个地方会访问zk，从zk上获取meta表所在的主机名，连接该主机，从该主机的meta表中读取要访问的region的主机名

  - 上述代码是从ZK中查找hbase:meta表region所在的RegionServer的主机名，连接该主机获取meta表中要访问的region的主机名，最终代码调用为：

```
static String getStubKey(final String serviceName, final String rsHostname, int port) {
  // Sometimes, servers go down and they come back up with the same hostname but a different
  // IP address. Force a resolution of the rsHostname by trying to instantiate an
  // InetSocketAddress, and this way we will rightfully get a new stubKey.
  // Also, include the hostname in the key so as to take care of those cases where the
  // DNS name is different but IP address remains the same.
  InetAddress i = new InetSocketAddress(rsHostname, port).getAddress();
  String address = rsHostname;
  if (i != null) {
    address = i.getHostAddress() + "-" + rsHostname;
  }
  return serviceName + "@" + address + ":" + port;
}
```

  - 在调用上述代码的的new InetSocketAddress(rsHostname,port).getAddress()时出现异常，而出现异常后会调用如下的代码，就会打印出“Cannot get replica 0 location for.....”的错误信息。

```
AsyncProcess.java
917     if (loc == null || loc.getServerName() == null) {
918       manageLocationError(action, null);
919       return null;
920     }
921     return loc;
922   }
923
924   private void manageLocationError(Action<Row> action, Exception ex) {
925     String msg = "Cannot get replica " + action.getReplicaId()
926         + " location for " + action.getAction();
927     LOG.error(msg);
928     if (ex == null) {
929       ex = new IOException(msg);
930     }
931     manageError(action.getOriginalIndex(), action.getAction(),
932         Retry.NO_LOCATION_PROBLEM, ex, null);
933   }
```

问题解决：在Flume的客户端的操作系统的/etc/hosts中添加集群的IP hostname映射关系，重启Flume客户端

Gdevops

全球敏捷运维峰会

THANK YOU!