CLOUDERA

CDP 使用指南

2021 年 05 月 12 日

目 录

1. 文档说明

本文档主要是基于阿里云部署的 CDP 的操作使用和介绍，关于 CDP 平台的操作和使用信息来源 Cloudera 官网，大家可以访问 https://docs.cloudera.com/cdp-private-cloud-base/latest/index.html 来获取对应的信息。

2. CDP 平台介绍

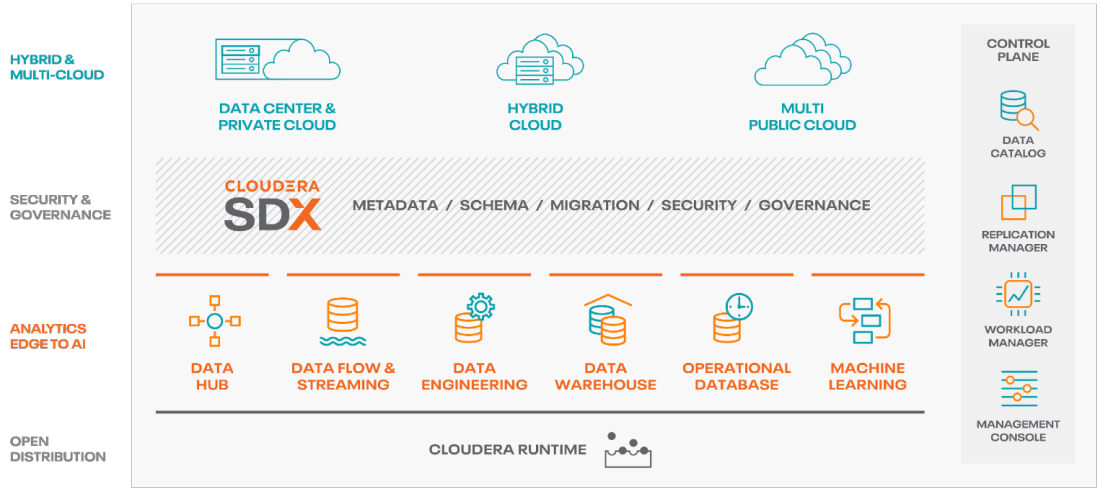
基于阿里云部署的 CDP 是部署在阿里云 ECS 集群上的集成的分析和数据管理平台，在该平台上提供广泛的数据分析和人工智能功能以及安全的用户访问和数据治理功能。

企业对大数据解决方案的要求：只要有必要，就可以在一个地方以原始的保真度来获取和合并任何数量或类型的数据，并尽可能快地向所有用户提供洞察力。

企业数据云公司 Cloudera 引入了企业数据云（EDC）的概念：数据驱动的企业需要能够对无处不在的数据应用多种分析规则；能够以流式的方式传输和处理来自边缘多个端点的实时数据，同时预测关键结果并在同一数据集上应用机器学习技术；能够充分利用公有云基础架构的敏捷性、灵活性以及日益庞大的数据引力；此外，能够在开放平台上完成所有这些工作，在数据存放和分析运行的所有位置都能应用数据安全和治理。这就是业界所说的企业数据云。 EDC 具有以下特点：

- 混合云和多云支持：提供选择来管理、分析和试验任何公有云和私有数据中心中的数据，以实现最大的选择和灵活性。
- 多功能分析：解决最苛刻的业务用例 – 跨共享数据大规模地应用实时流处理、数据仓库、数据科学和迭代机器学习。
- 安全性和治理：通过通用的安全模型来控制任何云（公有云、私有云和混合云）上的数据，简化了各种企业数据的数据隐私和合规性。
- 开放：促进开源社区的创新、提供开放存储和计算架构的选择性以及促进广泛的生态系统的信心和灵活性

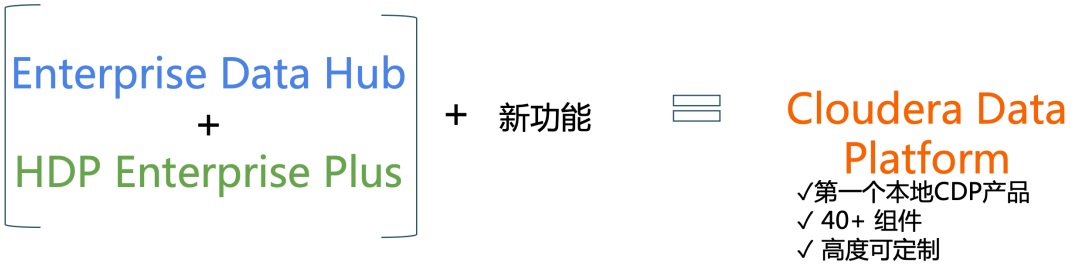
企业数据云平台不但提供企业级的安全性和治理能力，还同时提供多种分析功能用于数据分析，具备在内部和外部部署相同功能的能力，支持主要的公有云和私有云环境、使得用户获得弹性的云体验，并不再存在数据孤岛和单一供应商锁定的威胁。



EDC 不但可以灵活地运行各种企业工作负载（例如：实时摄取和分析、数据工程、交互式 SQL、企业搜索、高级分析和机器学习），还满足企业的要求：与企业现有的系统进行集成，同时提供强大的安全性、数据治理、数据保护和管理能力。EDC 是企业数据管理的新兴中心。

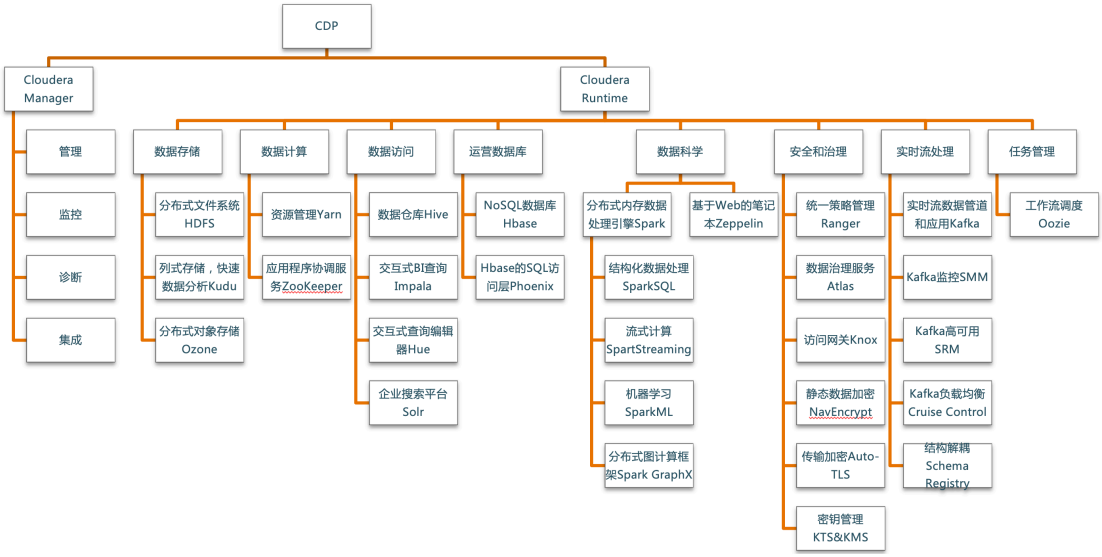
2.1. CDP 平台简介

EDC 建立在 Cloudera Data Platform(CDP)之上，CDP 是 Cloudera 公司的最新产品，该新产品结合了 Cloudera Enterprise Data Hub 和 Hortonworks Data Platform Enterprise 的优点，并在技术堆栈中增加了新功能和对已有技术提供了增强功能。这种统一的发行是一个可扩展且可自定义的平台，您可以在其中安全地运行多种类型的工作负载。



除了需要企业数据云外，企业还希望将这种强大的数据管理基础架构迁移或添加到云中，以提高运营效率、降低成本、提供计算和容量灵活性以及速度和敏捷性。

随着组织在云环境中采用基于 Hadoop 的大数据部署，他们还需要企业级的安全性和治理、多种分析功能、管理工具和技术支持-所有这些需求都是 CDP 平台的一部分，下图展示了 CDP 平台的功能地图。



CDP 支持各种混合解决方案，其中计算任务与数据存储分离，并且可以从远程集群访问数据。这种混合方法通过管理存储、表 Schema、身份验证、授权和治理，并为容器化应用程序提供了基础。

CDP 包括各种组件，例如 Apache HDFS、Apache Hive 3、Apache HBase 和 Apache Impala，以及许多其他用于特殊工作负载的组件。您可以选择这些服务的任意组合来创建满足您的业务需求和工作负载的集群。几个预配置的服务包也可用于常见的工作负载。

2.2. Cloudera Manager 概览

Cloudera Manager 是用于管理、配置和监控 CDP 集群和 Cloudera Runtime 服务的应用程序。

Cloudera Manager 服务器在 CDP 部署中的主机上运行，并使用在集群中每个主机上运行的 Cloudera Manager 代理来管理一个或多个集群。

Cloudera Manager 是用于管理集群的端到端应用程序。借助 Cloudera Manager，您可以轻松地部署和集中操作完整的 Cloudera Runtime 堆栈和其他托管服务。该应用程序可自动执行安装和升级过程，并为您提供主机和正在运行的服务的整个群集的实时视图。Cloudera Manager 管理控制台提供了一个中央控制台，您可以在其中对整个集群进行配置更改，并结合了各种报告和诊断工具来帮助您优化性能和利用率。Cloudera Manager 还管理安全性和加密功能。使用 Cloudera Manager 管理控制台，您可以启动和停止集群以及单个服务、配置和添加新服务、管理安全性以及升级集群。您还可以使用 Cloudera Manager API 以编程方式执行管理任务。

Cloudera Manager 的单个实例可以管理多个集群，包括较旧版本的 Cloudera Runtime 和 CDH。

2.3. Cloudera Runtime

Cloudera Runtime 是 CDP Private Cloud Base 中的核心开源软件发行版。Cloudera Runtime 包括大约 50 个开源项目，这些项目构成 CDP 中数据管理工具的核心分发。该库中记录了 Cloudera Runtime 组件。有关这些组件的版本列表，请参见附录。

2.4. 工具

CDP 还包括以下工具来管理和保护您的部署：

- Cloudera Manager 允许您使用 Cloudera Manager 管理控制台的 Web 应用程序或 Cloudera Manager API 管理、监控和配置集群和服务。
- Apache Atlas 提供了一组元数据管理和治理服务，使您能够管理 CDP 集群资产。
- Apache Ranger 通过用户界面管理访问控制，以确保 CDP 集群中一致的策略管理。

2.5. 设置对基于阿里云部署的 CDP 的访问权限

在阿里云或者内外网环境中，Cloudera 的平台产品 CDP 需要访问很多 Web UI，但系统网络可能仅支持 SSH 访问(22 端口)。要访问 Cloudera Manager（7180 端口）或者其他服务，可以通过下列两种方式：

- 在客户端计算机上设置 SOCKS（套接字安全协议）代理。Cloudera 建议您使用此选项。
- 将 CDP/CDP 部署到阿里云之后，将入站规则添加到阿里云实例中的网络安全组。

2.5.1. 配置 SOCKS 代理

SOCKS5 协议是作为客户端和服务器进程实现的，它可以遍历 IP 网络防火墙。配置 SOCKS 代理后，浏览器使用公有云网络（通过代理服务器）解析 DNS 查找，并允许您使用内部 FQDN 或专用 IP 地址连接到服务。

使用这种方法，您可以完成以下任务：

- 设置到网络上主机之一的单个 SSH 隧道，并在主机上创建 SOCKS 代理。
- 更改浏览器配置，以通过 SOCKS 代理主机执行所有查找。

网络先决条件

在使用 SOCKS 代理连接到集群之前，请验证以下先决条件：

- 您必须能够从公共 Internet 或您要从其连接的网络中访问要代理的主机。
- 您要代理的主机必须与您要连接的 Cloudera 服务位于同一网络上。例如，如果您使用的是 Cloudera CDP 产品，请通过 SSH 隧道连接到 Cloudera Manager 主机。

2.5.2. 启动 SOCKS 代理

Linux

要通过 SSH 启动 SOCKS 代理，请运行以下命令：

```
ssh -i your-key-file.pem -CND 1080
```

```
the_username_you_specified@publicIP_of_VM
```

该命令使用以下参数：

- `-i your-key-file.pem` 指定 SSH 到 Cloudera CDP/EDH 服务器所需的私钥的路径。如果使用 SSH 密码，则省略。
- `C` 设置压缩。
- `N` 建立后禁止执行任何命令。
- `D` 在端口上设置 SOCKS 代理。
- `1080` 用于在本地设置 SOCKS 代理的端口。

Windows

按照 Microsoft 网站上的说明进行操作。

2.5.3. 配置 Google Chrome 浏览器以使用代理

默认情况下，Chrome 浏览器会按配置文件使用系统范围的代理设置。要在没有这些设置的情况下启动 Chrome，请通过命令行打开 Chrome 并指定以下内容：

- SOCKS 代理端口。该端口必须与启动代理时使用的端口相同。
- 配置文件。下面的示例创建了一个新的配置文件。

使用以下命令之一创建配置文件并启动与当前任何正在运行的 Chrome 实例不冲突的 Chrome 的新实例。

Linux

```
/usr/bin/google-chrome \
--user-data-dir="$HOME/chrome-with-proxy" \
--proxy-server="socks5://localhost:1080"
```

Mac OS X

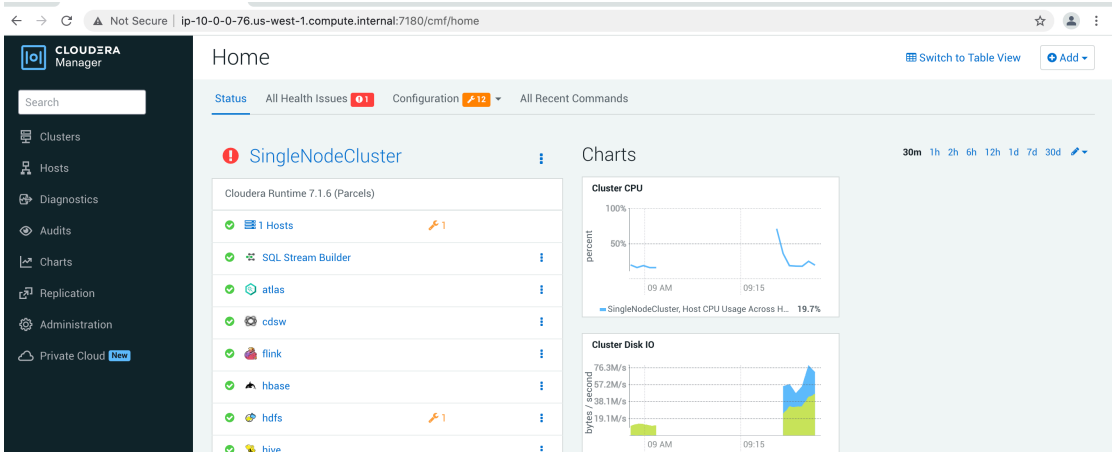
```
"/Applications/Google Chrome.app/Contents/MacOS/Google Chrome" \
--user-data-dir="$HOME/chrome-with-proxy" \
--proxy-server="socks5://localhost:1080"
```

微软 Windows

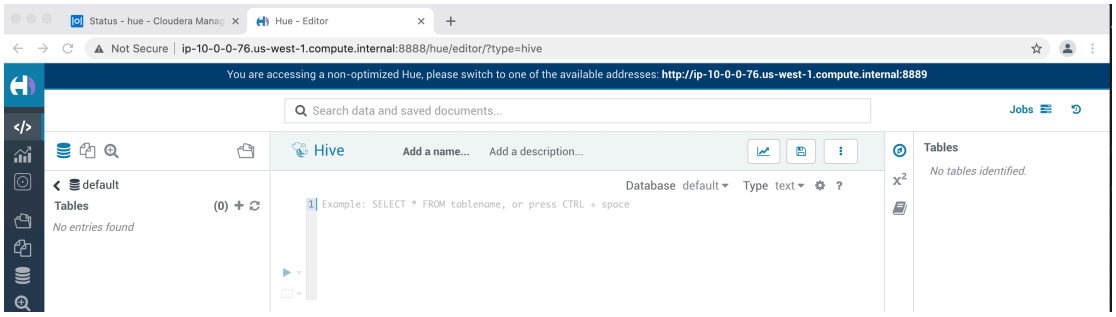
```
"C:\Program Files (x86)\Google\Chrome\Application\chrome.exe" ^
--user-data-dir="%USERPROFILE%\chrome-with-proxy" ^
--proxy-server="socks5://localhost:1080"
```

在此 Chrome 会话中，您可以使用私有 IP 地址或内部 FQDN 连接到 Cloudera CDP 可访问的任何主机。

这样就可以通过内网访问 Cloudera Manager 和其他 Web UI 了



也可以通过 CM 中的 web UI 跳转直接跳转过去。



2.5.4. 网络安全组

警告：除概念验证以外，不建议将此方法用于任何其他目的。如果没有仔细锁定数据，那么黑客和恶意实体将可以访问这些数据。

3. Cloudera Manager

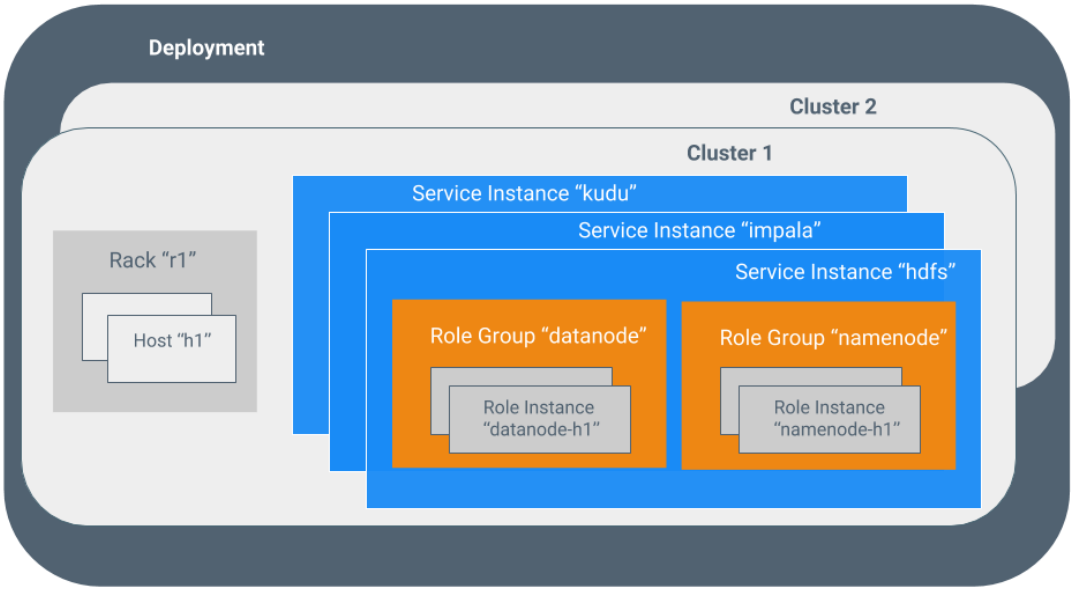
Cloudera Manager 是用于管理集群的端到端应用程序。借助 Cloudera Manager，您可以轻松部署和集中操作完整的 Cloudera Runtime 堆栈和其他托管服务。该应用程序可自动执行安装和升级过程，并为您提供主机和正在运行的服务的集群范围的实时视图。Cloudera Manager 管理控制台提供了一个中央控制台，您可以在其中跨集群进行配置更改，并结合了各种报告和诊断工具来帮助您优化性能和利用率。Cloudera Manager 还管理安全性和加密功能。本概述介绍了 Cloudera Manager 的基本概念、结构和功能。

3.1. 术语

Cloudera Manager 中使用的术语。

为了有效地使用 Cloudera Manager，您应该首先了解其术语。

术语之间的关系如下所示，其定义如下：



使用某些术语（例如集群和服务）无需进一步说明。以下各节将说明其他术语，例如角色组，网关，主机模板和 Parcel。

有时，术语*服务*和*角色*用于同时指代**类型**和**实例**，这可能会造成混淆。Cloudera Manager 和本节有时对**类型**和**实例**使用相同的术语。例如，Cloudera Manager 管理控制台的“ 主页” >“ 状态”选项卡和“ 集群” >“ *ClusterName*” 菜单列出了服务实例。这类似于编程语言中的惯例，其中“字符串”一词可能表示类型（java.lang.String）或该类型的实例（“ hi there”）。在需要区分类型和实例的地方，单词“ type”被附加以指示类型，而单词“ instance”被附加以显式指示实例。

3.1.1. 部署

Cloudera Manager 及其管理的所有集群的配置。

3.1.2. 动态资源池

在 Cloudera Manager 中，这是资源的命名配置，以及用于在池中运行的 YARN 应用程序或 Impala 查询之间调度资源的策略。

3.1.3. 集群

包含 HDFS 文件系统并对该数据运行 MapReduce 和其他进程的一组计算机或计算机机架。

在 Cloudera Manager 中，是一个逻辑实体，包含一组主机，在主机上安装的单个版本的 Cloudera Runtime 以及在主机上运行的服务和角色实例。一台主机只能属于一个集群。Cloudera Manager 可以管理多个集群，但是每个集群只能与一个 Cloudera Manager Server 关联。

3.1.4. 主机

在 Cloudera Manager 中，是运行角色实例的物理或虚拟机。一台主机只能属于一个集群。

3.1.5. 机架

在 Cloudera Manager 中，是一个物理实体，包含一组通常由同一交换机提供服务的物理主机。

3.1.6. 服务

- 一个 Linux 命令，它/etc/init.d/在尽可能可预测的环境中运行 System V 初始化

脚本，删除了大多数环境变量并将当前工作目录设置为/。

- Cloudera Manager 中的托管功能类别，可以在集群中运行，该功能可以分布式的也可以不是分布的。有时称为服务类型。例如：Hive，HBase，HDFS，YARN 和 Spark。

3.1.7. 服务实例

在 Cloudera Manager 中，是在集群上运行的服务的实例。例如：“ HDFS-1”和“YARN”。服务实例跨越许多角色实例。

3.1.8. 角色

在 Cloudera Manager 中，服务中的功能类别。例如，HDFS 服务具有以下角色：NameNode，SecondaryNameNode，DataNode 和 Balancer。有时称为角色类型。

3.1.9. 角色实例

在 Cloudera Manager 中，是在主机上运行的角色的实例。它通常映射到 Unix 进程。例如：“ NameNode-h1”和“ DataNode-h1”。

3.1.10. 角色组

在 Cloudera Manager 中，这是一组角色实例的一组配置属性。

3.1.11. 主机模板

Cloudera Manager 中的一组角色组。将模板应用于主机时，将创建每个角色组中的角色实例并将其分配给该主机。

3.1.12. 网关（Gateway）

一种角色类型，通常为客户端提供对特定集群服务的访问权限。例如，HDFS，Hive，Kafka，MapReduce，Solr 和 Spark 各自具有网关角色，以为其客户提供对其各自服务的访问。网关角色并非总是在其名称中具有“网关”，也不是专门用于客户端访问。例如，Hue Kerberos Ticket Renewer 是一个网关角色，用于代理 Kerberos 中的票证。

支持一个或多个网关角色的节点有时称为*网关节点*或*边缘节点*，在网络或云环境中常见“边缘”的概念。对于 Cloudera 集群，当从 Cloudera Manager 管理控制台的“操作”菜单中选择“ 部署客户端配置”时，集群中的网关节点将接收适当的客户端配置文件。

3.1.13. Parcel

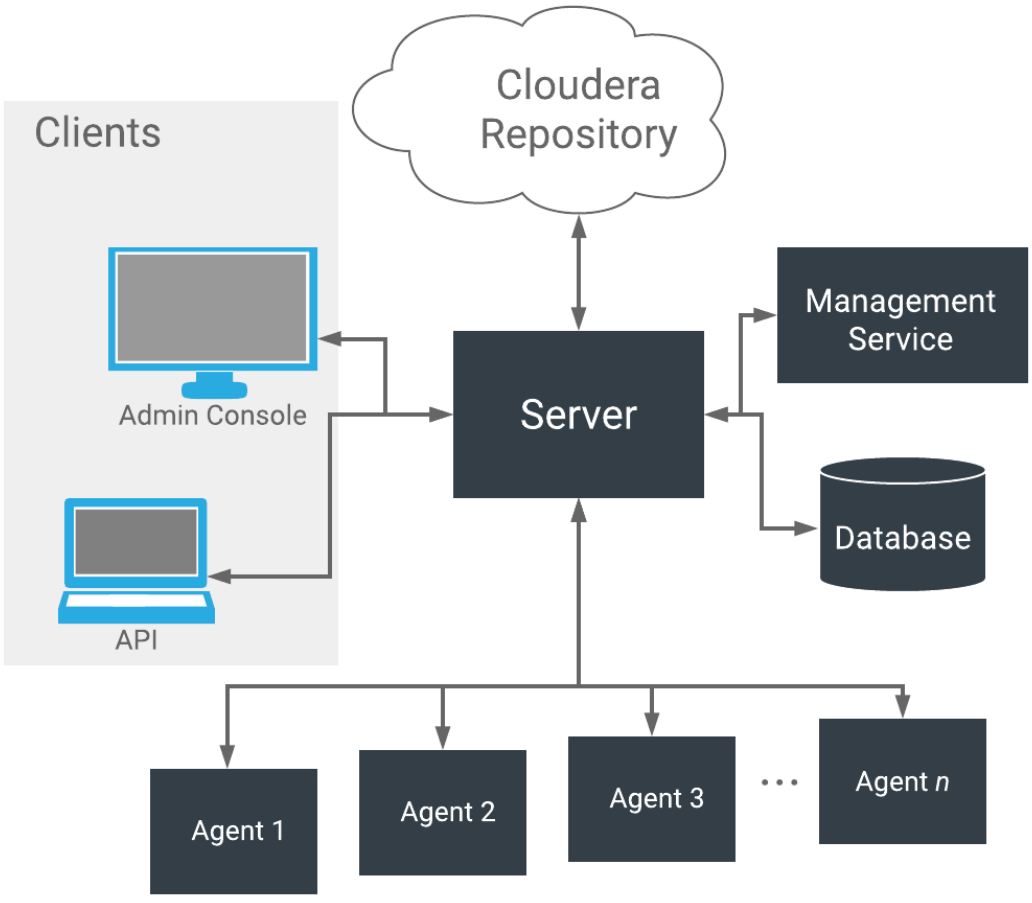
二进制分发格式，包含编译的代码和元信息，例如程序 parcel 描述、版本和依赖项。

3.1.14. 静态服务池

在 Cloudera Manager 中，跨一组服务对集群总资源（CPU，内存和 I/O 权重）进行静态分区。

3.2. Cloudera Manager 架构

如下图所示，Cloudera Manager 的核心是 Cloudera Manager Server，它托管 Cloudera Manager 管理控制台、Cloudera Manager API 和应用程序逻辑，并负责安装软件、配置、启动和停止服务以及管理运行服务的集群。



Cloudera Manager Server 与其他几个组件一起使用：

- **Agent(代理)** -安装在每台主机上。该代理负责启动和停止进程、解包配置、触发安装以及监控主机。
- **Management Service（管理服务）** -由一组角色组成的服务，这些角色执行各种监控、警报和报告功能。
- **数据库** -存储配置和监控信息。通常，多个逻辑数据库跨一个或多个数据库服务器运行。例如，Cloudera Manager Server 和监控角色使用不同的逻辑数据库。
- **Cloudera Repository（存储库）** -由 Cloudera Manager 分发的软件**存储**库。
- **客户端** -是与服务器交互的接口：
  - **Cloudera Manager Admin Console（CM 管理控制台）** -基于 Web 的管理员用于管理集群和 Cloudera Manager 的用户界面。
  - **Cloudera Manager API** -API 开发人员用于创建自定义 Cloudera Manager 应用程序。

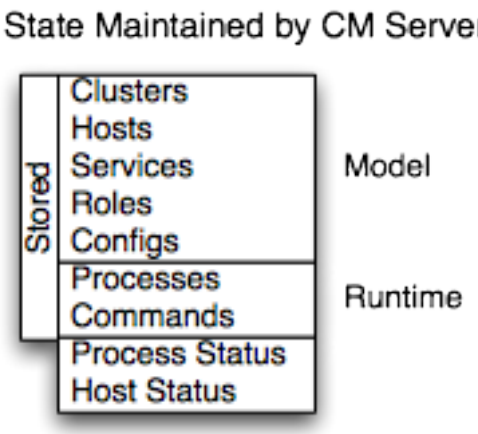
3.2.1. 心跳

心跳是 Cloudera Manager 中的主要通信机制。默认情况下，代理每 15 秒将心跳发送一次到 Cloudera Manager Server。但是，为减少用户等待时间，在状态更改时会增加频率。

在心跳交换期间，代理会将其活动通知给 Cloudera Manager Server。反过来，Cloudera Manager Server 响应代理应执行的操作。代理和 Cloudera Manager Server 最终都进行了一些协调。例如：如果您启动服务，则代理将尝试启动相关进程。如果进程无法启动，则 Cloudera Manager Server 会将启动命令标记为失败。

3.3. 状态管理

Cloudera Manager Server 维护集群的状态。此状态可以分为两类：“模型”和“运行时”，两者都存储在 Cloudera Manager Server 数据库中。



Cloudera Manager 为集群和托管服务建模：它们的角色、配置和相互依赖关系。*模型状态*捕获应该在哪里运行以及在什么配置下运行。例如，模型状态捕获了一个集群包含 17 个主机的事实，每个主机都应运行一个 DataNode。您可以通过 Cloudera Manager 管理控制台配置屏幕，API 和诸如“添加服务”之类的操作与模型进行交互。

*运行时状态*是当前在何处运行哪些进程以及正在运行哪些命令（例如，重新平衡 HDFS 或运行备份/灾难恢复计划或滚动重启或 停止）。运行时状态包括运行进程所需的确切配置文件。在 Cloudera Manager 管理控制台中选择“启动”后，服务器将收集相关服务和角色的所有配置，对其进行验证，生成配置文件，并将其存储在数据库中。

更新配置（例如，Hue Server Web 端口）时，您已经更新了模型状态。但是，如果在执行此操作时 Hue 正在运行，则它仍在使用旧端口。当发生这种不匹配时，该角色将被标记为具有“过时的配置”。要重新同步，请重新启动角色（这将触发重新生成配置并重新启动进程）。

尽管 Cloudera Manager 为所有合理的配置建模，但某些情况下不可避免地需要特殊处理。为了使您能够解决问题，例如发现错误或探索不受支持的选项，Cloudera Manager 支持“ 高级配置代码段 ”机制，该机制可让您直接向配置文件添加属性。

3.4. Cloudera Manager 管理控制台

Cloudera Manager 管理控制台是基于 Web 的界面，可用于配置、管理和监控 Cloudera Runtime。

Cloudera Manager 管理控制台侧面导航栏提供以下选项卡和菜单：

注意

根据用于登录的用户角色，某些项目可能不会出现在 Cloudera Manager 管理控制台中。

- **搜索** -支持搜索服务、角色、主机、配置属性和命令。您可以输入部分字符串，并显示一个下拉列表，其中最多显示 16 个匹配的实体。
- 集群> *cluster_name*
    - **服务** -显示单个服务以及 Cloudera 管理服务。在这些页面中，您可以：
        - 查看服务实例或与该服务关联的角色实例的状态和其他详细信息
        - 对服务实例，角色或特定角色实例进行配置更改
        - 添加和删除服务或角色
        - 停止、启动或重新启动服务或角色。
        - 查看已为服务或角色运行的命令
        - 查看审核事件历史记录
        - 部署和下载客户端配置
        - 退役和重新委任角色实例
        - 进入或退出维护模式
        - 执行特定于特定服务类型的操作。例如：
            - 启用 HDFS 高可用性或 NameNode 联邦
            - 运行 HDFS Balancer
            - 创建 HBase、Hive 和 Sqoop 目录
    - Cloudera Manager 管理服务 -管理和监控 Cloudera Manager 管理服务。这包括以下角色：活动监控器、警报发布者、事件服务器、主机监控器、报告管理器和服务监控器(Activity Monitor, Alert Publisher, Event Server, Host Monitor, Reports Manager, and Service Monitor.)。
    - **主机** -显示集群中的主机。
    - **报告** -创建有关 HDFS、MapReduce、YARN 和 Impala 使用情况的报告、浏览 HDFS 文件、并管理 HDFS 目录的配额。
    - 利用率报告 -打开 集群利用率报告。显示 YARN 和 Impala 作业的汇总利用率信息。
    - ***MapReduce_service_name* 任务** -有关集群上运行 MapReduce 作业查询信息。
    - ***YARN_service_name* 应用程序** -查询有关在集群上运行的 YARN 应用程序

的信息。

- ***Impala_service_name* 查询** -查询有关在集群上运行的 Impala 查询的信息。
- **动态资源池** -通过指定命名池的相对权重来管理对 YARN 和 Impala 服务的集群资源的动态分配。
- **静态服务池** -管理集群资源对 HBase、HDFS、Impala、MapReduce 和 YARN 服务的静态分配。

- **主机** -显示由 Cloudera Manager 管理的主机。
  - **所有主机** -显示集群中管理主机的列表。
  - **添加主机** -启动“添加主机”向导。
  - **Parcels** -显示集群中可用的 parcels，并允许您下载、分发和激活新的 parcels 包。
  - **主机配置** -打开“ 主机配置”页面，您可以在其中配置主机并为一个或多个主机的全局配置属性指定替代。
  - **角色** -显示部署在每个主机上的角色。
  - **主机模板** -创建和管理 主机模板，这些模板定义了可用于轻松扩展集群的角色组集。
  - **磁盘概述** -显示集群中所有磁盘的状态。

  在此页面中，您可以：
  - 查看有关单个主机的状态和各种详细指标
  - 进行配置更改以进行主机监控
  - 查看主机上运行的所有进程
  - 运行主机检查器
  - 添加和删除主机
  - 创建和管理主机模板
  - 管理 Parcels
  - 退役和重新托管主机
  - 进行机架分配
  - 运行主机升级向导
- **诊断 -查看日志、事件和警报以诊断问题。子页面为：**
  - **事件** -搜索并显示已发生的事件和警报。
  - **日志** -按服务、角色、主机和搜索短语以及日志级别（严重性）搜索日志。

  - **服务器日志** -显示 Cloudera Manager 服务器日志。
- **审核 -查询和筛选跨集群的审核事件，包括登录，跨集群等。**
- **图表 -查询感兴趣的指标，将其显示为图表，并显示个性化的图表仪表板。**
- **复制 -管理复制计划和快照策略。**
- **管理 -管理 Cloudera Manager。子页面为：**
  - **设置** -配置 Cloudera Manager。
  - **警报** -显示何时生成警报，配置警报收件人以及发送测试警报电子邮件。
  - **用户和角色** -管理 Cloudera Manager 用户及其分配的角色和会话。
  - **安全性** -生成 Kerberos 凭据并检查主机。
  - **许可证** -管理 Cloudera 许可证。
  - **语言** -设置活动事件、健康事件和警报电子邮件的内容所使用的语言。
  - **外部帐户** -配置从云服务到 Cloudera Manager 的连接。
- **Parcels -打开 Parcels 页面，您可以在其中查看已安装和可用 parcels 的状态。**
- **最近命令指示器 -**  **显示所有服务或角色当前或最近正在运行的状态命令。**
- **支持 -显示各种支持的行动。子命令是：**
  - **发送诊断数据** - 将数据发送到 Cloudera 支持以支持故障排除。
  - **支持门户（Cloudera Enterprise）** -显示 Cloudera 支持门户。
  - **计划的诊断：每周** -配置自动收集诊断数据并发送给 Cloudera 支持的频率。
  - 以下链接打开了 Cloudera 网站上的最新文档：
    - **帮助文档**
    - **安装指南**
    - **API 文档**
    - **API Explorer（Cloudera Manager Swagger 界面）**
    - **发行说明**
  - **关于** -Cloudera Manager 的版本号和内部版本详细信息以及 Cloudera Manager 服务器的当前日期和时间戳。
- **登录用户菜单 -当前登录的用户。子命令是：**
  - **我的个人资料** -显示当前用户的角色和登录信息。
  - **更改密码** -更改当前登录用户的密码。
  - **登出**

3.4.1. Cloudera Manager 管理控制台主页

启动 Cloudera Manager 管理控制台时，将 显示“ 主页” >“ 状态”选项卡。您也可以 通过单击顶部导航栏中的 Cloudera Manager Logo 转到“ 主页” >“ 状态”选项卡。

“ 状态”选项卡具有两个潜在的视图：“表视图”和“经典视图”。经典视图包含所选集群的一组图表，而表视图将常规集群，计算集群和其他服务分隔为汇总表。您可以使用每个视图上的“ 切换到表视图”和“ 切换到经典视图”链接在两个视图之间切换。Cloudera Manager 会记住您选择的视图并保留在该视图中。

图 1. Cloudera Manager 管理控制台：经典视图

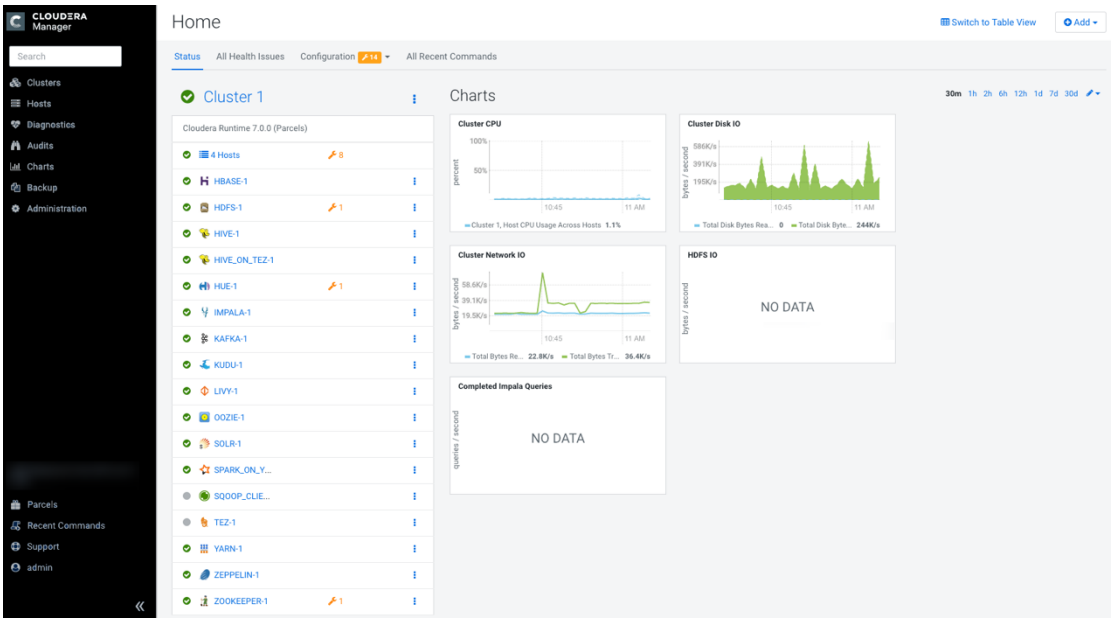


图 2. Cloudera Manager 管理控制台

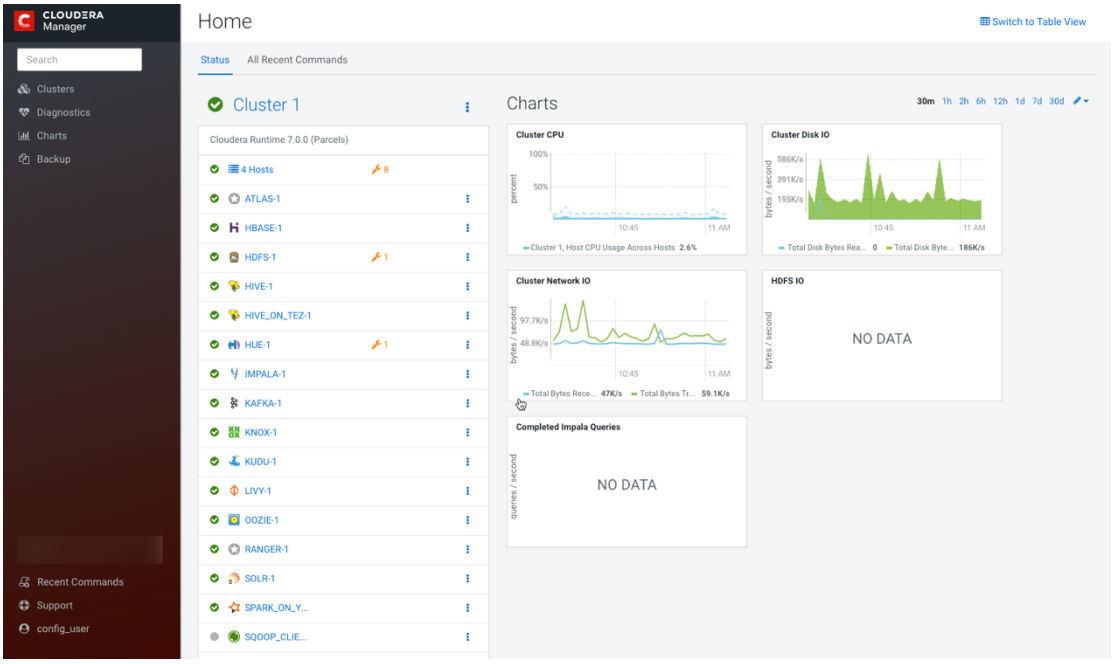


图 3. Cloudera Manager 管理控制台：表视图

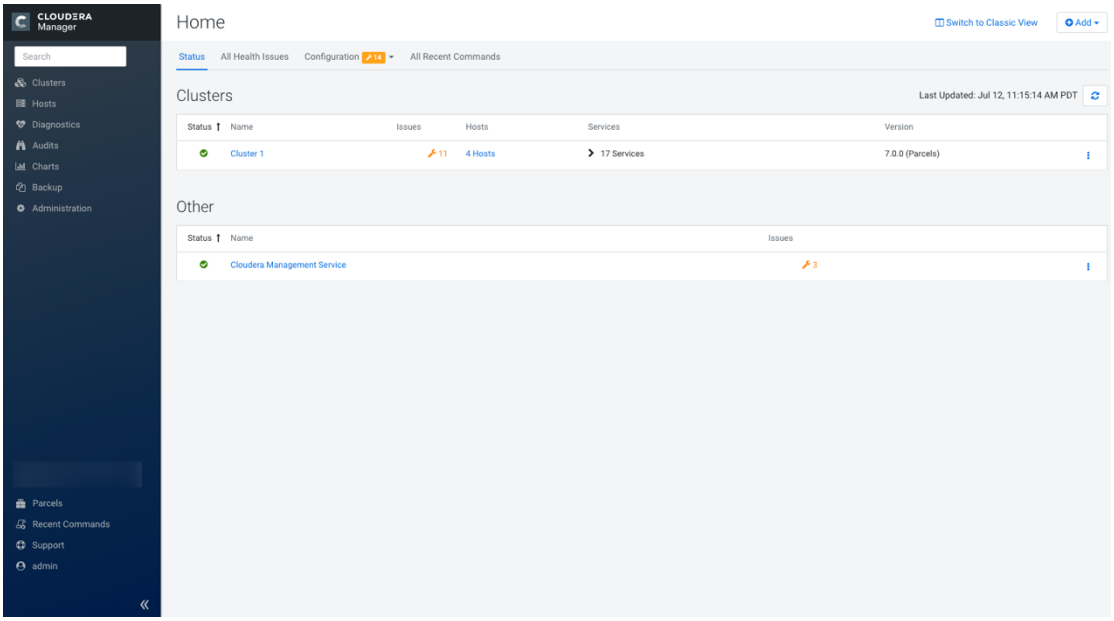
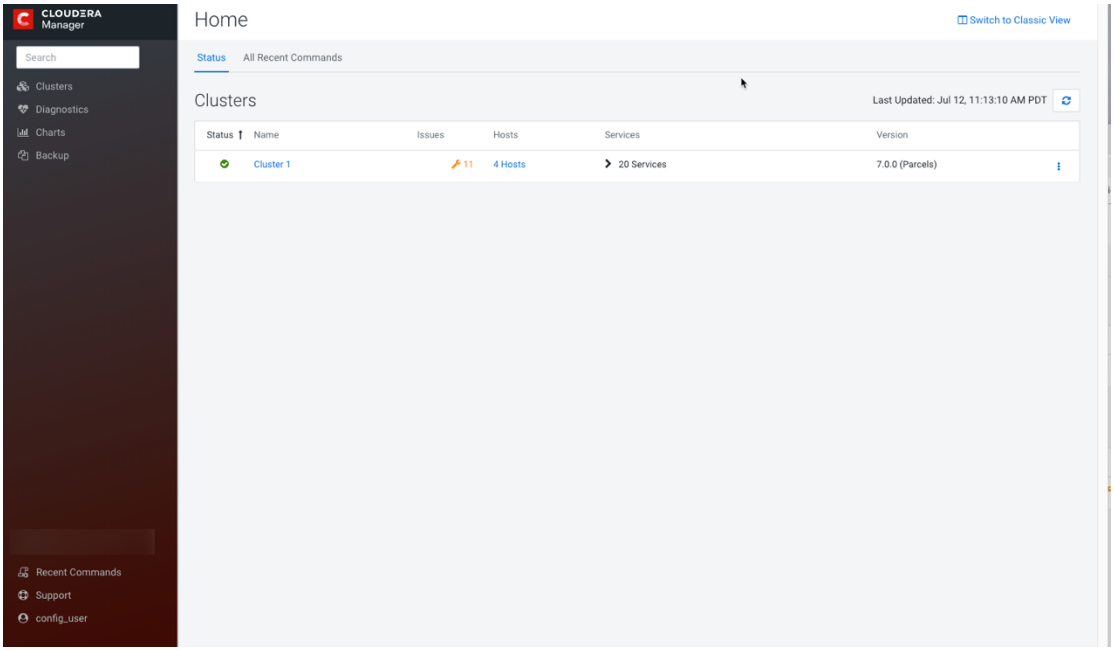


图 4. Cloudera Manager 管理控制台：表视图



3.4.1.1. 状态

状态选项卡包含：

- **集群** - 由 Cloudera Manager 管理的集群。每个集群以摘要形式或完整形式显示，这具体取决于**“ 管理” > “设置” >“ 其他” >“ 完整显示的最大集群数”**属性的配置。当集群数超过该属性的值时，仅显示集群摘要信息。
    - **摘要形式** -集群状态页面的链接列表。单击**“ 自定义”**以跳至**“ 管理” > “设置” >“ 其他” >“ 完整显示的最大集群数”** 属性。
    - **完整形式** -每个集群的单独部分，包含指向集群状态页面的链接以及包含指向主机页面和集群中运行的服务的状态页面的链接的表。

表格中的每个服务行都有一个操作菜单，您可以通过单击“操作菜单”（ ）选择该菜单，并且可以包含以下一个或多个指示符：

指示符	含义	描述
2	健康问题	表示该服务至少有一个健康问题。该指示器以最高严重性级别显示健康问题的数量。如果有不良运行状况测试结果，则指示器为红色。如果没有不良健康测试结果，但有关测试结果存在，则指示器为黄色。如果没有不良或有关健康测试的结果，则不会显示任何指示器。 **重要** 如果有一个不良健康测试结果和两个有关健康结果，则将有三个健康问题，但数量将是一个。 单击指示器以显示“ 健康问题”弹出对话框。

指示符	含义	描述
		默认情况下，对话框中仅显示不良运行状况测试结果。要显示关于健康测试结果，请点击还显示 *ñ* 有关的问题（一个或多个）链接。单击链接以显示“状态”页面，其中包含有关运行状况测试结果的详细信息。
	配置问题	表示该服务至少有一个配置问题。该指示器以最高严重性级别显示配置问题的数量。如果存在配置错误，则指示灯为红色。如果没有错误，但是存在配置警告，则该指示器为黄色。如果没有配置通知，则不会显示任何指示器。 **重要** 如果出现一个配置错误和两个配置警告，将出现三个配置问题，但数量将是一个。 单击指示器以显示“ 配置问题”弹出对话框。 默认情况下，仅列出错误严重性级别的通知，并在对话框中显示按服务名称分组的通知。要显示警告通知，请单击“ 也显示 *n* *条* 警告”链接。单击与错误或警告相关的消息，将其带到已发出通知的配置属性中，您可以在其中解决该问题。
 需要重启  需要刷新	配置已修改	指示至少一个服务角色正在使用与 Cloudera Manager 中的当前配置设置不匹配的配置运行。 单击指示器以显示“ 陈旧配置”页面。要使集群最新，请单击“陈旧配置”页面上的“刷新”或“重新启动”按钮。您还可以刷新或 重新启动集群或重新启动服务。
	需要客户端配置重新部署	指示应重新部署服务的客户端配置。 单击指示器以显示“ 陈旧配置”页面。要使集群最新，请单击“陈旧配置”页面上的“部署客户端配置”按钮，或手动重新部署客户端配置。

- **Cloudera Management Service-包含指向 Cloudera Manager 服务的链接的表。Cloudera Manager Service 包含您通过单击选择的操作菜单**  **。**
- **图表 -一组汇总资源利用率（IO、CPU 使用率）和处理指标的图表（仪表板）。**

单击折线图，堆栈区域图，散点图或条形图，可以将其展开为全页视图，并带有图例以显示单个图表实体以及更细粒度的轴分区。

默认情况下，仪表板的时间范围为 30 分钟。要更改时间刻度，请单击

仪表板右上方的持续时间链接。

要设置仪表盘类型，请单击  并选择以下选项之一：

- 自定义 -显示自定义信息中心。
- 默认 -显示默认仪表板。
- 重置 -将自定义仪表板重置为预定义的图表集，并放弃所有自定义项。

3.4.1.2. 所有的健康问题

按集群显示所有运行状况问题。数字徽章的语义与“状态”选项卡上报告的每个服务运行状况问题相同。

- 默认情况下，对话框中仅显示不良运行状况测试结果。要显示关于健康测试结果，请点击还显示 ***ñ* 有关的问题**的链接。
- 要将健康测试结果按实体或健康测试分组，请单击**“ 按实体组织”/“按健康测试组织”的**按钮进行切换。
- 单击链接以显示“状态”页面，其中包含有关运行状况测试结果的详细信息。

3.4.1.3. 所有的配置问题

按集群显示所有配置问题。数字 Logo 的语义与“状态”选项卡上报告的每个服务配置问题相同。默认情况下，仅列出错误严重性级别的通知，并在对话框中显示按服务名称分组的通知。要显示警告通知，请单击“ 也**显示 *n* *条*警告**”链接。单击与错误或警告相关的消息，将其带到已发出通知的配置属性中，您可以在其中解决该问题。

3.4.1.4. 所有最近的命令

显示最近在集群中运行的所有命令。  Logo 指示最近有多少命令仍在运行。单击命令链接以显示有关命令和子命令的详细信息。

3.4.1.5. 显示 Cloudera Manager 服务器版本和服务器时间

要显示 Cloudera Manager Server 的版本、内部版本号和时间：

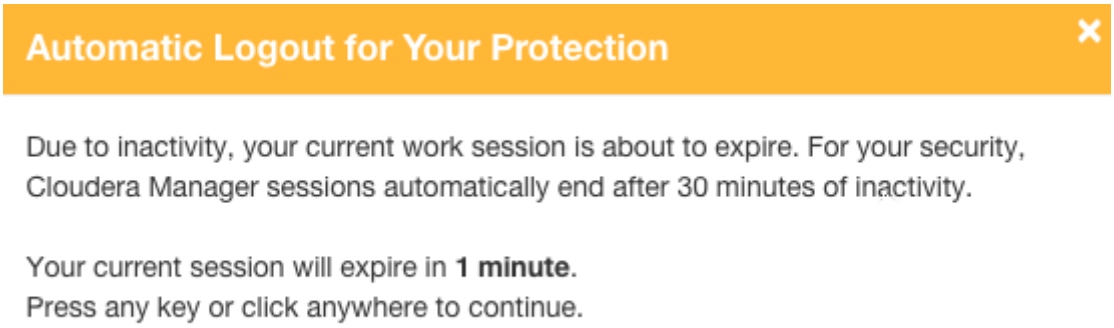
1) 打开 Cloudera Manager 管理控制台。
2) 选择 **支持> 关于**。

3.4.2. 自动登出

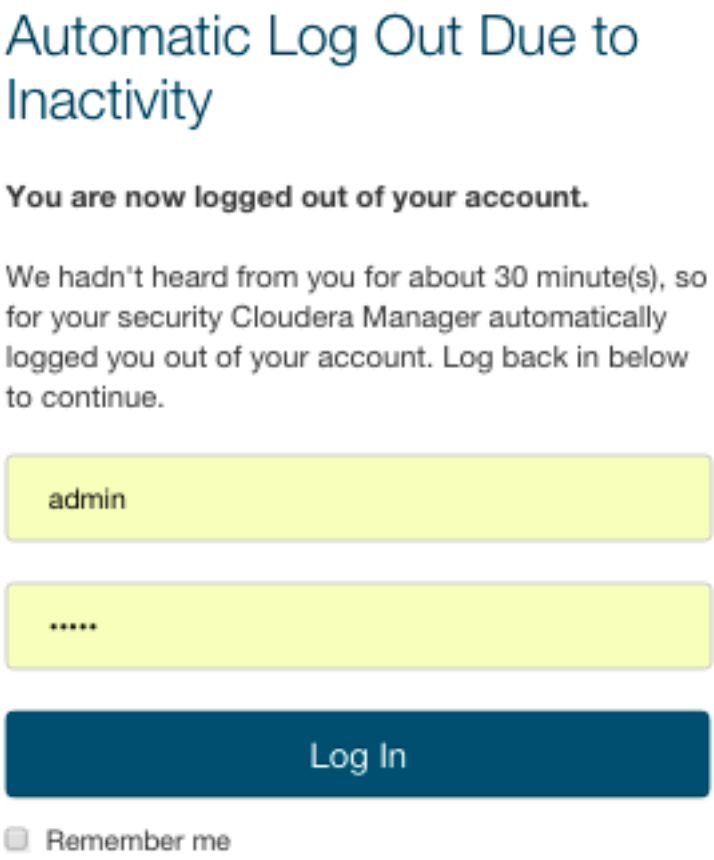
为了安全起见，Cloudera Manager 在 30 分钟后会自动注销用户会话。您可以更改此会话注销时间。

1) 单击 **管理> 设置**。
2) 单击 **类别> 安全性**。
3) 编辑**会话超时**属性。
4) 输入**更改原因**，然后单击**“ 保存更改”**以提交更改。

如果超时是触发后的一分钟，则用户会看到以下消息：



如果用户未单击鼠标或按任意键，则该用户将退出会话并显示以下消息：



3.5. 进程管理

使用 Cloudera Manager 来启动和停止进程。

在 Cloudera Manager 托管集群中，您只能使用 Cloudera Manager 启动或停止角色实例进程。Cloudera Manager 使用一个名为 supervisor 的开源进程管理工具，该工具可启动进程、负责重定向日志文件、通知流程失败、将调用进程的有效用户 ID 设置为正确的用户等。Cloudera Manager 支持自动重启崩溃的进程。如果其实例在启动后立即反复崩溃，它还将使用不良运行状况标志来标记角色实例。

停止 Cloudera Manager Server 和 Cloudera Manager Agent 不会关闭您的服务。任何正在运行的角色实例都将继续运行。

在 `init.d` 启动时将启动 CM Agent。然后，它与 Cloudera Manager Server 联系并确定应运行哪些进程。Agent 作为 Cloudera Manager 主机监控的一部分进行监控。如果 Agent 停止心跳，则会将主机标记为运行状况不良。

代理的主要职责之一是启动和停止进程。当代理从服务器检测信号检测到新进程时，代理会为其创建目录`/var/run/cloudera-scm-agent` 并解压缩配置。然后它联系 `supervisord`，从而开始该进程。

这些操作强调了一个重点：Cloudera Manager 进程永远不是单独旅行。换句话说，一个进程不仅仅是 exec（）的参数，它还包括配置文件、需要创建的目录以及其他信息。

3.6. 主机管理

Cloudera Manager 提供了多种功能来管理集群中的主机。

首次运行 Cloudera Manager 管理控制台时，您可以搜索要添加到集群中的主机，一旦选择了主机，就可以将角色分配映射到主机。Cloudera Manager 自动将所有作为托管主机参与集群所需的软件部署到主机上：JDK、Cloudera Manager Agent、Impala、Solr 等。

部署并运行服务后，管理控制台中的“主机”区域将显示集群中托管主机的整体状态。提供的信息包括在主机上运行的 Cloudera Runtime 的版本，主机所属的集群以及在主机

上运行的角色数。Cloudera Manager 提供用于管理参与主机的生命周期以及添加和删除主机的操作。Cloudera Management Service 主机监控器角色执行运行状况测试并收集主机指标，以允许您监控主机的运行状况和性能。

3.7. Cloudera Manager Agent

Cloudera Manager Agent 是 Cloudera Manager 组件，可与 Cloudera Manager Server 一起使用以管理映射到角色实例的进程。

在 Cloudera Manager 托管集群中，您只能使用 Cloudera Manager 启动或停止角色实例进程。Cloudera Manager 使用一个称为 supervisor 的开源进程管理工具，该工具 可启动进程、负责重定向日志文件、通知进程失败、将调用进程的有效用户 ID 设置为正确的用户等。Cloudera Manager 支持自动重启崩溃的进程。如果其实例在启动后立即反复崩溃，它还将使用不良运行状况标志来标记角色实例。

`init.d` 在启动时将启动 Agent。然后，它与 Cloudera Manager Server 联系并确定应运行哪些进程。代理作为 Cloudera Manager 主机监控的一部分进行监控。如果代理停止心跳，则会将主机标记为运行状况不良。

代理的主要职责之一是启动和停止进程。当代理从服务器检测信号检测到新进程时，代理会为其创建目录`/var/run/cloudera-scm-agent` 并解压缩配置。然后它与 `supervisord` 联系，从而开始该进程。

3.7.1. cm_processes

为了使 Cloudera Manager 能够在的子目录 `var/run/cloudera-scm-agent` 中运行脚本（因为`/var/run` 已在许多 Linux 发行版中挂载了 `noexec` 该脚本 ），Cloudera Manager 会为进程子目录挂载一个名为 `cm_processes` 的临时文件存储 tmpfs。

`tmpfs` 默认的物理 RAM 的 50%的最大空间，但该空间没有被分配，直到使用时才分配。如果有存储器的压力，会将 `tmpfs` 调出到 swap。

`cmprocesses` 的生命周期行为可以通过以下语句描述：

- 在 Agent 程序使用新的 supervisord 进程首次启动时创建 `cmprocesses`。
- 如果 `cmprocesses` 已经存在，但没有了 `noexec`，则在 agent 使用 start 命令启动

时，重用已有的 `cmprocesses` 而不是重新创建。

- 如果 agent 已经启动，则使用 `clean_restart` 重新挂载。
- 卸载和重新挂载会清除内容（因为它是作为 tmpfs 安装的）。
- 主机重启后卸载。
- 当 Agent 停止时不卸载。

3.8. 资源管理

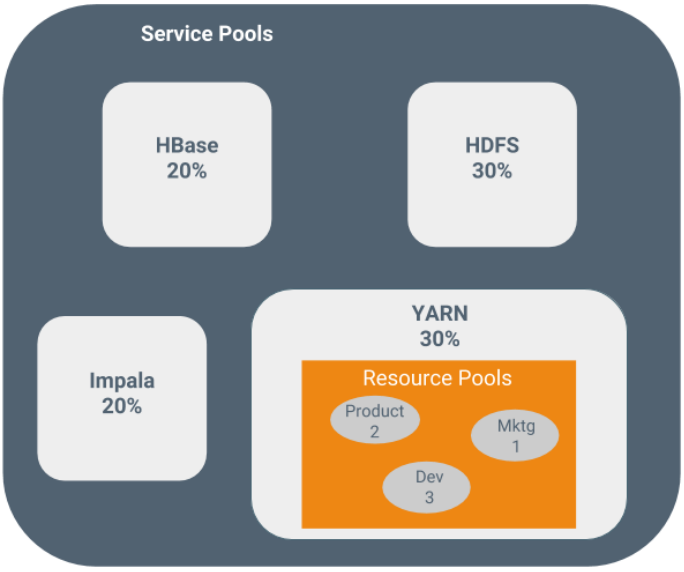
资源管理通过定义不同服务对集群资源的影响来帮助确保可预测的行为。

使用资源管理可以：

- 确保在合理的时间内完成关键工作负载。
- 基于每组资源的公平分配，支持在用户组之间进行合理的集群调度。
- 防止用户剥夺其他用户访问集群的权限。

使用 cgroups 静态分配资源可通过单个*静态服务池*向导进行配置。您将服务分配为总资源的百分比，然后向导将配置 cgroup。

例如，下图说明了 HBase，HDFS，Impala 和 YARN 服务的静态池，分别为其分配了 20%，30%，20%和 30%的集群资源。



您可以使用*动态资源池*动态分配已经静态分配给 YARN 和 Impala 的 *资源*。

根据使用的 CDP-DC 的版本，Cloudera Manager 中的动态资源池支持以下方案：

- **YARN** -YARN 管理虚拟核心、内存、正在运行的应用程序，未声明的子代（对于父池）的最大资源以及每个池的调度策略。在上图中，为 YARN 定义了三个动态资源池（分别为权重 3、2 和 1 的 Dev，Product 和 Mktg）。如果启动了一个应用程序并将其分配给产品池，而其他应用程序正在使用 Dev 和 Mktg 池，则产品资源池将获得总集群资源的 30%x 2/6（或 10%）。如果没有应用程序在使用 Dev 和 Mktg 池，则 YARN 产品池将分配 30%的集群资源。
- **Impala** -Impala 管理正在运行的查询的池的内存，并限制每个池中正在运行的查询和排队查询的数量。

3.9. 用户管理

对 Cloudera Manager 功能的访问由用户帐户控制。用户帐户标识用户的身份验证方式，并确定授予该用户哪些特权。

Cloudera Manager 提供了几种用于验证用户身份的机制。您可以将 Cloudera Manager 配置为根据 Cloudera Manager 数据库或外部身份验证服务对用户进行身份验证。外部身份验证服务可以是 LDAP 服务器（Active Directory 或与 OpenLDAP 兼容的目录），也可以指定其他外部服务。Cloudera Manager 还支持使用安全性断言标记语言（SAML）来启用单点登录。

3.10. 安全管理

Cloudera Manager 整合了集群中所有组件的安全配置和管理。

作为旨在支持大量和类型的数据的系统，Cloudera 集群必须满足监管机构、政府、行业和公众提出的不断发展的安全要求。Cloudera 集群包含 Hadoop 核心和生态系统组件，必须保护所有这些组件免受各种威胁，以确保所有集群服务和数据的机密性、完整性和可用性。

Cloudera Manager 提供了用于管理认证，授权和加密的工具。

3.11. 使用 Cloudera Manager 监控集群

Cloudera Manager 提供了许多功能，用于监控集群组件（主机，服务守护程序）的运行状况和性能以及集群上运行的作业的性能和资源需求。

Cloudera Manager 中提供以下监控功能：

- 监控 Cloudera Runtime Services-描述如何在服务和角色实例级别上查看运行状况测试的结果。各种类型的度量标准显示在有助于问题诊断的图表中。运行状况测试包括有关组件的运行状况令人担忧或不良时可以采取的措施的建议。您还可以查看对服务或角色执行的操作的历史记录，还可以查看配置更改的审核日志。
- 监控主机 -描述如何查看与集群中所有主机有关的信息：哪些主机正在运行或正在关闭，主机的当前驻留内存和虚拟内存消耗，主机上正在运行的角色实例，哪些主机已分配给不同的机架，等等。您可以查看集群中所有主机的摘要视图，也可以深入了解有关单个主机的详细信息，包括提供有关主机关键指标的直观概述的图表。
- 活动 -描述如何在当前时间以及通过显示历史活动的仪表板查看集群上正在运行的活动，并在表格显示和图表中提供有关各个作业使用的资源的许多统计信息。您可以比较类似作业的性能，并查看整个作业中各个任务尝试的性能，以帮助诊断行为或性能问题。
- 事件 -描述如何查看事件并使它们可用于警报和搜索，使您可以查看集群范围内发生的所有相关事件的历史记录。您可以按时间范围、服务、主机、关键字等过滤事件。
- 警报 -描述如何配置 Cloudera Manager 以从某些事件生成警报。您可以为某些类型的事件配置阈值，启用和禁用它们，并通过电子邮件或针对严重事件使用 SNMP 陷阱配置警报通知。您还可以临时抑制单个角色、服务、主机甚至整个集群的警报，以进行系统维护/故障排除而不会产生过多的警报流量。
- 生命周期和安全审核 -描述如何查看服务、角色和主机的生命周期事件，例如创建角色或服务，对角色或服务进行配置修订，停用和重新调试主机以及运行 Cloudera Manager 管理服务记录的命令。您可以按时间范围，服务，主机，关键字等过滤审核事件条目。
- 图表时间序列数据 -描述如何搜索指标数据、创建数据图表、对数据进行分组（构面）以及将这些图表保存到用户定义的仪表板。
- 日志 -描述如何以多种方式访问日志，并考虑到您正在查看的当前上下文。例如：在监控服务时，您可以通过同一用户界面轻松单击单个链接以查看与该特定服务相关的日志条目。在查看有关用户活动的信息时，您可以轻松地查看作业运行时在作业所使用的主机上发生的相关日志条目。
- 报告 -描述如何按用户、用户组和目录查看有关磁盘利用率的历史信息，以及如何查看集群作业活动的用户，组或作业 ID。这些报告在选定的时间段（每小时、每天、每周等）上汇总，并且可以导出为 XLS 或 CSV 文件。您还可以管理 HDFS 目录、包括搜索和设置配额。
- 对集群配置和操作进行故障排除 -包含一些常见问题的解决方案，这些问题使您无法使用 Cloudera Manager，并介绍了如何使用 Cloudera Manager 日志和通知管理工具来诊断问题。

3.12. Cloudera Management Service

Cloudera Management Service 是 Cloudera Manager 用来管理和监控集群的一组角色。

Cloudera Management Service 将各种管理功能实现为一组角色：

- 主机监控器-收集有关主机的运行状况和度量标准信息
- 服务监控器-从 YARN 和 Impala 服务收集有关服务的健康和度量信息以及活动信息
- 事件服务器-聚集相关的 Hadoop 事件并使它们可用于警报和搜索
- Alert Publisher-为某些类型的事件生成并提供警报
- 报告管理器-生成报告，以提供有关用户、用户组和目录的磁盘利用率，用户和 YARN 池以及 HBase 表和名称空间处理活动的历史视图。

您可以通过执行以下任一操作来查看 Cloudera Management Service 的状态：

- 选择集群> Cloudera 管理服务。
- 在“ 主页” >“ 状态” 选项卡上的“ Cloudera Management Service”表中，单击“ **Cloudera Management Service”**链接。

3.12.1. 健康测试

Cloudera Manager 使用运行*状况测试*来监控集群中运行的服务、角色和主机的运行*状况*。Cloudera Management Service 还为其角色提供运行状况测试。默认情况下启用基于角色的运行状况测试。例如，简单的运行状况测试是每个 NameNode 数据目录中是否有足够的磁盘空间。更复杂的运行状况测试可以评估何时将 HDFS 的最后一个检查点与阈值进行比较，或者将 DataNode 是否连接到 NameNode。其中一些运行状况测试还会汇总其他运行状况测试：在像 HDFS 这样的分布式系统中，关闭几个 DataNode 是正常的（假设您拥有数十台主机），因此我们允许设置阈值，确定应着色的主机百分比整个服务下降。

运行状况测试可以返回以下三个值之一： **Good**，**Concerning** 和 **Bad**。如果测试跌落到警告阈值以下，则测试返回 **“有关**健康”。如果测试低于关键阈值，则测试返回“ **不良”**。服务或角色实例的整体运行状况是其运行状况测试的汇总。如果有任何健康测试是**有关**（但没有一个是 **坏**）角色的或服务的健康是**有关**; 如果任何运行状况测试为**不良**，则服务或角色的运行状况 **为坏**。

在 Cloudera 的管理器管理控制台，健康测试结果都标有颜色：好  ，有关  和坏  。

一个常见的问题是，是否可以将监控与配置分开。监控的目标之一是无需进行其他配置和安装其他工具（例如 Nagios）即可启用它。通过具有深入的配置模型，Cloudera Manager 可以知道要监控的目录，要使用的端口以及用于这些端口的凭证。这种紧密的耦合意味着，当您安装 Cloudera Manager 时，将启用所有监控。

3.12.2. 指标收集和显示

要执行监控，服务监控器和主机监控器会收集指标。*度量*是一个数值，与名称相关联（例如，“CPU 秒”），一个实体它适用于（“host17”）和时间戳。大多数度量标准收集是由代理执行的。代理与受监督的进程进行通信，请求指标，并将其转发到服务监控器。在大多数情况下，此操作每分钟执行一次。

服务监控器收集一些特殊指标。例如，服务监控器托管一个 HDFS 金丝雀，该金丝雀定期尝试从 HDFS 写入、读取和删除文件，并测量该文件是否成功以及花费了多长时间。接收到指标后，便会对其进行汇总和存储。

使用 Cloudera Manager 管理控制台中的“图表”页面，您可以查询和浏览正在收集的指标。图表显示*时间序列*，它们是特定实体的度量标准数据点的流。每个度量标准数据点都包含一个时间戳和该度量标准在该时间戳记的值。

一些指标（例如 `total_cpu_seconds`）是计数器，查询它们的合适方法是随时间推移获取它们的汇率，这就是为什么很多指标查询都包含该 `dt0` 功能的原因。例如，`dt0(total_cpu_seconds)`。（该 `dt0` 语法旨在使您想起派生类。0 表示单调递增计数器的速率永远不应具有负速率。）

3.12.3. 事件、警报和触发器

一个*事件*是一个自己感兴趣的东西已经发生的记录-一个服务的健康状态发生了改变，（适当的严重性）日志信息已被记录，依此类推。默认情况下，许多事件是启用和配置的。

一个*警告*是被认为特别值得一提的，由选定的事件触发的事件。警报 

出现在事件列表中时，会带有徽章显示。您可以配置警报发布器，以通过电子邮件或 SNMP 陷阱将警报通知发送到陷阱接收器。

一个*触发器*是当一个或多个特定条件得到满足的服务，角色，角色配置组，或主机将采取指定动作的声明。条件表示为 tsquery 语句，并且要采取的措施是将服务，角色，角色配置组或主机的运行状况更改为“正在关注”（黄色）或“错误”（红色）。

3.13. 集群配置概述

Cloudera Manager 管理集群中运行的所有角色的配置。

当 Cloudera Manager 配置服务时，它会将该服务所需的*角色*分配给 集群中的主机。该角色确定哪些服务守护程序在主机上运行。

例如，对于 HDFS 服务实例，Cloudera Manager 配置：

- 一台主机运行 NameNode 角色。
- 一台主机作为次要 NameNode 角色运行。
- 一台主机来运行 Balancer 角色。
- 其余要运行 DataNode 角色的主机。

角色组是角色类型的一组配置属性，以及与该组关联的角色实例的列表。Cloudera Manager 会为每个角色类型自动创建一个名为 Role Type Default Group 的默认角色组。

运行安装或升级向导时，Cloudera Manager 会配置它添加的默认角色组，并为给定角色类型添加任何其他必需的角色组。例如，与 NameNode 在同一主机上的 DataNode 角色可能需要与在其他主机上运行的 DataNode 角色不同的配置。Cloudera Manager 为在 NameNode 主机上运行的 DataNode 角色创建一个单独的角色组，并对在其他主机上运行的 DataNode 角色使用默认配置。

Cloudera Manager 向导会根据主机上可用的资源自动配置角色组属性。对于不依赖于主机资源的属性，Cloudera Manager 的默认值通常与该配置的 Cloudera Runtime 的默认值一致。当不建议使用 Cloudera Runtime 默认设置或默认值非法时，Cloudera Manager 会偏离。

3.14. 服务器和客户端配置

Cloudera Manager 从其数据库生成服务器和客户端配置文件。

有时令管理员感到惊讶的是，修改/etc/hadoop/conf 然后重新启动 HDFS 无效。这是因为由 Cloudera Manager 启动的服务实例不会从默认位置读取配置。以 HDFS 为例，如果不受 Cloudera Manager 的管理，通常每个主机有一个 HDFS 配置，位于/etc/hadoop/conf/hdfs-site.xml。在同一主机上运行的服务器端守护程序和客户端都将使用相同的配置。

Cloudera Manager 区分服务器和客户端配置。对于 HDFS，该文件/etc/hadoop/conf/hdfs-site.xml 仅包含与 HDFS 客户端相关的配置。也就是说，默认情况下，如果您运行需要与 Hadoop 通信的程序，它将从该目录获取 NameNode 和 JobTracker 的地址以及其他重要配置。/etc/hbase/conf 和 采取了类似的方法/etc/hive/conf。

相反，HDFS 角色实例（例如 NameNode 和 DataNode）从私有的每个进程目录（在/var/run/cloudera-scm-agent/process/*unique-process-name* 下）获取其配置 。通过为每个流程提供自己的私有执行和配置环境，Cloudera Manager 可以独立控制每个流程。例如，以下是示例 879-hdfs-NAMENODE 进程目录的内容：

```
$ tree -a/var/run/cloudera-scm-Agent/process/879-hdfs-NAMENODE/
 /var/run/cloudera-scm-Agent/process/879-hdfs-NAMENODE/
   ├── cloudera_manager_Agent_fencer.py
   ├── cloudera_manager_Agent_fencer_secret_key.txt
   ├── cloudera-monitor.properties
   ├── core-site.xml
   ├── dfs_hosts_allow.txt
   ├── dfs_hosts_exclude.txt
   ├── event-filter-rules.json
   ├── hadoop-metrics2.properties
   ├── hdfs.keytab
   ├── hdfs-site.xml
   ├── log4j.properties
   ├── logs
```

```
|   ├── stderr.log
|   └── stdout.log
├── topology.map
└── topology.py
```

区分服务器和客户端配置具有以下优点：

- 服务器端配置中的敏感信息（例如，Hive Metastore RDBMS 的密码）不会公开给客户端。
- 依赖于另一个服务的服务可以使用自定义配置进行部署。例如，为了获得良好的 HDFS 读取性能，Impala 需要 HDFS 客户端配置的专用版本，这可能对通用客户端有害。这是通过将 Impala 守护程序的 HDFS 配置（存储在上述的按进程目录中）和通用客户端的 HDFS 配置分开来实现的`/etc/hadoop/conf`。
- 客户端配置文件更小，更易读。这也避免了使非管理员 Hadoop 用户与无关的服务器端属性混淆。

3.15. Cloudera Manager API

Cloudera Manager API 提供配置和服务生命周期管理，服务运行状况信息和指标，并允许您配置 Cloudera Manager 本身。

Cloudera Manager API 与 Cloudera Manager 管理控制台位于同一主机和端口上，并且不需要额外的过程或额外的配置。该 API 支持 HTTP 基本身份验证，接受与 Clouder a Manager 管理控制台相同的用户和凭证。

您还可以从 Cloudera Manager 管理控制台访问 Cloudera Manager Swagger API 用户界面。转到 支持> API 资源管理器以打开 Swagger。

3.16. 虚拟专用集群和 Cloudera SDX

虚拟专用集群使用 Cloudera 共享数据体验（SDX）来简化本地和基于云的应用程序的部署，并使在不同集群中运行的工作负载能够安全灵活地共享数据。

虚拟专用集群的体系结构为部署工作负载和在应用程序之间共享数据提供了许多优势，包括共享目录，统一安全性，一致的治理和数据生命周期管理。

在传统的 CDH 部署中，*常规集群*包含存储节点、计算节点以及其他服务，例如并置在单个集群中的元数据服务和安全服务。这种传统体系结构具有许多优势，例如 Impala 和 YARN 等计算服务可以访问并置的数据源（例如 HDFS 或 Hive）。

借助虚拟专用集群和 SDX 框架，Cloudera Manager 6.2 及更高版本中提供了一种新型的集群，称为 *计算集群*。Compute 集群运行诸如 Impala，Hive Execution Service，Spark 或 YARN 之类的计算服务，但您将这些服务配置为访问另一个常规 CDH 集群（称为 *Base 集群*）中托管的数据。使用此体系结构，您可以通过多种方式分离计算和存储资源，以灵活地最大化资源。

3.16.1. 分离计算和数据资源的优势

对于许多工作负载，在虚拟专用集群中分离计算和数据资源具有重要的优势。

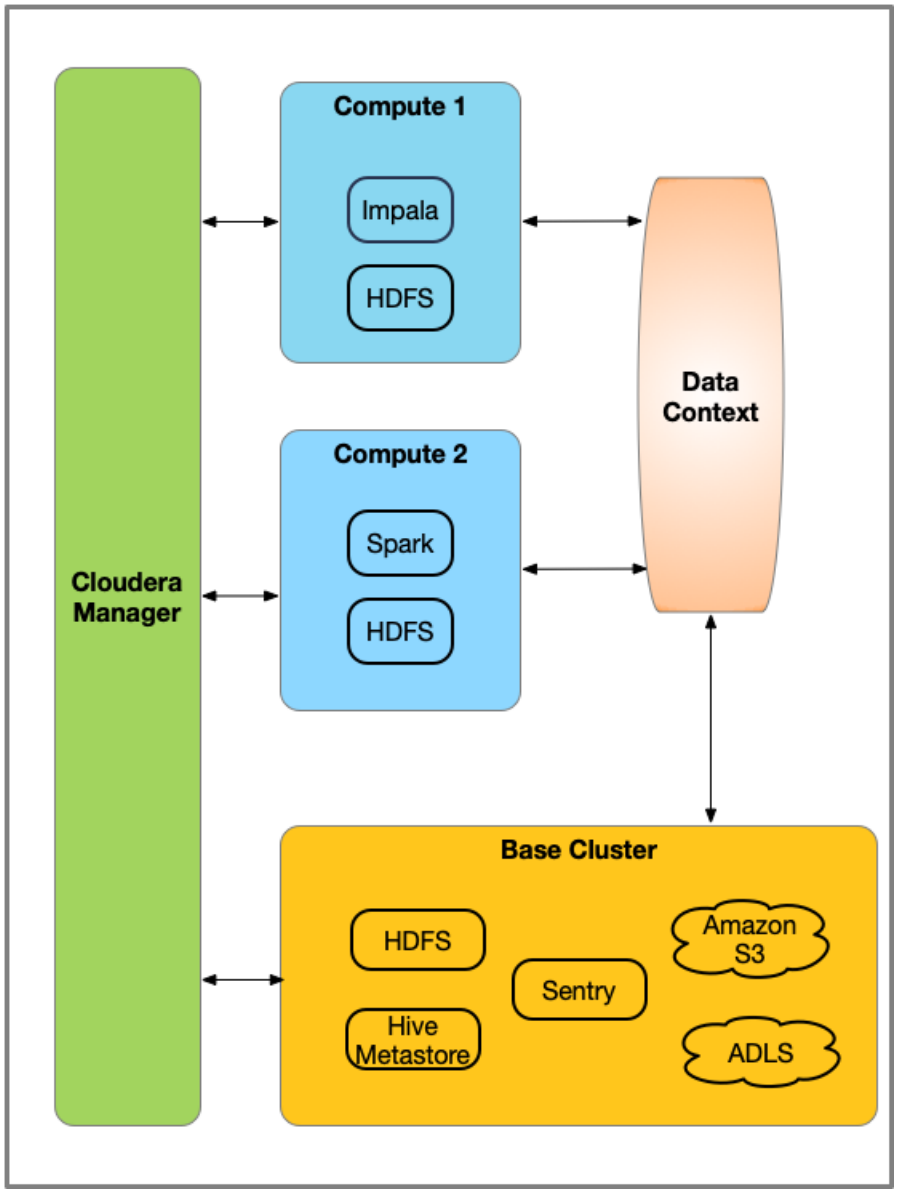
将计算资源与数据资源分离的体系结构可以为 CDH 部署提供许多优势：

- **部署计算和存储资源的更多选项**
  - 您可以使用针对工作负载量身定制的本地服务器，容器，虚拟机或云资源来选择性地部署资源。配置计算集群时，可以配置更适合计算工作负载的硬件，而基本集群可以使用强调存储容量的硬件。Cloudera 建议每个集群使用相似的硬件。
  - 可以对软件资源进行优化，以最佳利用计算和存储资源。
- **临时集群**
  - 在云基础架构上部署集群时，拥有用于计算和存储的单独集群可以使您暂时关闭计算集群并避免不必要的开销-同时仍将数据留给其他应用程序使用。
- **工作负载隔离**
  - 计算集群可以帮助解决访问集群的用户之间的资源冲突。可以隔离运行时间较长或占用大量资源的工作负载，以在不干扰其他工作负载的专用计算集群中运行。
  - 可以将资源分组到集群中，以使 IT 部门可以将成本分配给使用资源的团队。

3.16.2. 架构

如果要创建虚拟专用集群，那么了解计算集群的体系结构以及它们与数据上下文的关系非常重要。

*计算集群*被配置成与计算资源，如 Yarn，Spark，Hive Execution, 或 Impala。这些集群上运行的工作负载通过连接到基本集群的*数据上下文*来访问数据。数据上下文是指定为基本集群的常规集群的连接器。数据上下文定义访问基本数据所需的数据，元数据和安全服务。计算集群和基本集群均由 Cloudera Manager 的同一实例管理。基本集群必须已部署 HDFS 服务，并且可以包含任何其他 CDH 服务-但可以使用数据上下文共享 HDFS，Hive，Sentry，Amazon S3 和 Microsoft ADLS。



计算集群需要 HDFS 服务来存储多阶段 MapReduce 作业中使用的临时文件。此外，可以根据需要部署以下服务：

- Hive Execution Service（此服务仅提供 HiveServer2 角色。）

- Hue
- Impala
- Kafka
- Spark2
- Oozie（仅在 hue 可用且 hue 需要 Oozie 时）
- Yarn
- HDFS（必填）

虚拟专用集群的功能是常规集群中可用功能的一部分，并且可以使用的 CDH 版本受到限制。

3.16.3. 权衡性能

了解适用于虚拟专用集群的工作负载类型可以帮助您确定此体系结构是否适合您的需求。

3.16.3.1. 吞吐量

由于将通过与其他集群的网络连接来访问数据，因此该体系结构可能不适用于扫描大量数据的工作负载。这些类型的工作负载可能在常规集群上更好地运行，在常规集群上，计算和存储并置，并且 Impala 短路读取之类的功能可以提供改进的性能。

您可以使用 Network Performance Inspector 评估网络性能。

3.16.3.2. 临时集群

对于不需要关闭或暂停 Compute 集群的部署，当 Compute 集群处于脱机状态时，收集历史数据的集群服务不会收集数据，并且历史记录对用户不可用。这会影响诸如 Spark History Server 和 YARN JobHistory Server 之类的服务。重新启动计算集群时，以前的历史记录将可用。

3.16.4. 虚拟专用集群的兼容性注意事项

3.16.4.1. CDH 版本兼容性

计算集群的 CDH 版本必须与基本集群的 major.minor 版本匹配。稍后可能会添加对 Compute 和 Base 集群版本的其他组合的支持。支持以下 CDH 版本来创建虚拟专用集群：

- CDH 5.15

- CDH 5.16
- CDH 6.0
- CDH 6.1
- CDH 6.2
- CDH 6.3
- Cloudera Runtime 7.0.2

3.16.4.2. CDH 组件

- **SOLR** –计算集群不支持
- **Kudu** –计算集群不支持。

  使用 Kudu 的 HiveMetastore 上的存储位置不适用于 Compute 集群上的 Impala。

- **HDFS**
  - 计算集群上需要“本地” HDFS 服务作为临时的持久空间；目的是将其用于 Hive 临时查询，也建议将其用于多阶段 Spark ETL 作业。
  - Cloudera 建议将每台主机的最小存储量设置为 HDFS DataNode 存储目录，至少为 1TB。
  - 基本集群必须具有 HDFS 服务。
  - 基本集群不支持 Isilon。
  - 仅在基本集群上支持 S3 和 ADLS 连接器；计算集群将使用其关联的基本集群中提供的 S3 或 ADLS 凭据
  - 必须将基本集群上的 HDFS 服务配置为具有高可用性。
  - Cloudera 强烈建议为 Compute 集群上的 HDFS 服务启用高可用性，但这不是必需的。
  - 基本名称和计算名称服务名称必须不同。
  - 计算集群上本地 HDFS 服务的以下配置必须与基本集群上的配置匹配。这使计算集群上的服务能够正确访问基本集群上的服务：
    - Hadoop RPC 保护
    - 数据传输保护
    - 启用数据传输加密
    - Kerberos 配置
    - TLS/SSL 配置（仅当集群未使用自动 TLS 时）。请参阅安全性。
  - 不要使用“高级配置摘要”覆盖 Compute 集群中的 nameservice 配置。
  - 用于计算集群主目录的 HDFS 路径使用以下结构：
    `/mc/<cluster_id>/fs/user`

您可以通过在 Cloudera Manager 管理控制台中单击集群名称来找到*<cluster_id>*。浏览器显示的 URL 包含集群 ID。例如，在下面的 URL 中，集群 ID 为 1。

```
http://myco-1.prod.com:7180/cmf/clusters/1/status
```

- **备份和灾难恢复（BDR）** –当源集群是 Compute 集群且目标集群运行的 Cloudera Manager 版本低于 6.2 时不支持。
- YARN 和 MapReduce
    - 如果基本集群上同时可用 MapReduce（在 CM6 中不推荐使用 MR1）和 YARN（MR2），则由于 Cloudera Manager 中处理服务依赖关系的方式，计算集群中的依赖服务（例如 Hive Execution Service）将使用 MR1。要使用 YARN，您可以在应用程序中使用 YARN 之前，更新配置以使这些 Compute 集群服务依赖 YARN（MR2）。
- **Impala**
    - 将新数据或元数据吸收到影响 Hive Metastore 的基本集群中时，需要在安装了 Impala 的每个 Compute 集群上运行 `INVALIDATE METADATA`or `REFRESH METADATA` 语句，然后才能使用。
    - 目录服务（`catalogd`）中的表级锁定可确保 Impala 的一致性。对于多个计算集群，通过多个访问多个集群中的同一表或数据 `catalogds` 可能会导致问题。例如，删除文件时查询可能会失败，或者当在一个集群上运行的刷新从另一集群中提取的数据中有一半时，错误的数据会进入元数据。

为避免一致性问题，您的 Impala 集群应在互斥的表和数据集上运行。

- **Hue**
    - 计算集群上仅支持一个 Hue 服务实例。
    - 计算集群上的 Hue 服务不会与其他计算集群上的 Hue 服务或基本集群上的 Hue 服务共享用户特定的查询历史记录
    - 由于创建表和插入数据的权限不同，Hue 示例可能无法正确安装。您可以通过删除示例表，然后重新添加它们来解决此问题。
    - 如果在创建集群后将 Hue 添加到 Compute 集群，则需要手动配置对 Compute 集群中其他服务（例如 Hive，Hive 执行服务和 Impala）的任何依赖关系。
- **Hive Execution service**
    - 新推出的“ Hive Execution Service”仅在 Compute 集群上受支持，而在 Base 或 Regular 集群上不受支持。
    - 要使 Hue 在 Compute 集群上运行 Hive 查询，您必须在 Compute 集群上安装 Hive Execution Service。

3.16.4.3. 计算集群服务

计算集群上只能安装以下服务：

- Hive Execution Service（此服务仅提供 HiveServer2 角色。）
- Hue
- Impala
- Kafka
- Spark2
- Oozie（仅在有 Hue 且是 Hue 的要求时）
- Yarn
- HDFS（必填）

3.16.4.4. Cloudera Navigator 支持

CDH Compute 集群不支持 Navigator 谱系和元数据以及 Navigator KMS。

3.16.4.5. Cloudera Manager 权限

被授权仅查看基本集群或计算集群的集群管理员只能查看和管理这些集群，而不能创建，删除或管理数据上下文。仅具有“完全管理员”使用角色或不受限制的集群管理员才能创建和删除数据上下文。

3.16.4.6. 安全

- **知识管理系统**
  - **基本集群：**
    - 不支持 Hadoop KMS。
    - 基本集群支持 KeyTrustee KMS。
  - **计算集群：不**支持任何类型的 KMS 。
- **认证/用户目录**
  - 用户应在计算集群和基本集群上的主机上进行相同配置，就像节点是同一集群的一部分一样。这包括 Linux 本地用户，LDAP，Active Directory 或其他第三方用户目录集成。
- **Kerberos**
  - 如果基本集群安装了 Kerberos，则计算集群和基本集群必须在同一 Kerberos 领域中都使用 Kerberos。Cloudera Manager 管理控制台有助于在集群创建过程中简化此配置，以确保兼容的 Kerberos 配置。
- **TLS**

- 如果基本集群为集群服务配置了 TLS，则计算集群服务还必须为访问这些基本集群服务的服务配置 TLS。
- Cloudera 强烈建议启用自动 TLS，以确保基础集群和计算集群上的服务统一使用 TLS 进行通信。
- 如果您已配置 TLS，但未使用自动 TLS，请注意以下几点：
  - 您必须先在 Compute 集群主机上创建相同的配置，**然后才能**使用 Cloudera Manager 将它们添加到 Compute 集群。将以下配置属性指定的目录中的所有文件从基本集群复制到计算集群主机：
    - `hadoop.security.group.mapping.ldap.ssl.keystore`
    - `ssl.server.keystore.location`
    - `ssl.client.truststore.location`
    - 创建计算集群时，Cloudera Manager 会将以下配置复制到计算集群。
    - `hadoop.security.group.mapping.ldap.use.ssl`
    - `hadoop.security.group.mapping.ldap.ssl.keystore`
    - `hadoop.security.group.mapping.ldap.ssl.keystore.password`
    - `hadoop.ssl.enabled`
    - `ssl.server.keystore.location`
    - `ssl.server.keystore.password`
    - `ssl.server.keystore.keypassword`
    - `ssl.client.truststore.location`
    - `ssl.client.truststore.password`

3.16.4.7. 计算集群中主机的存储要求

如果 Impala 在 Compute 集群上运行，则 Compute 集群主机将需要附加的存储，且容量至少为 1TB。此存储用于 Impala 暂存空间以及分配了 HDFS DataNode 角色的主机中的 DFS 本地存储。

3.16.4.8. 网络

在 Compute 集群上运行的工作负载将与 Base 集群上的主机进行大量通信；客户应该对网络硬件（例如，包括机架顶部的交换机，骨干/树叶路由器等）进行适当的网络监控，以跟踪和调整在 Compute 和 Base 集群中使用的托管主机的机架之间的带宽。

您还可以使用 Cloudera Manager 网络性能检查器评估网络。

有关如何设置网络的更多信息，请参阅《虚拟专用集群的网络注意事项》。

3.16.4.9. Cloudera 数据科学工作台（CDSW）

计算集群不支持 CDSW。

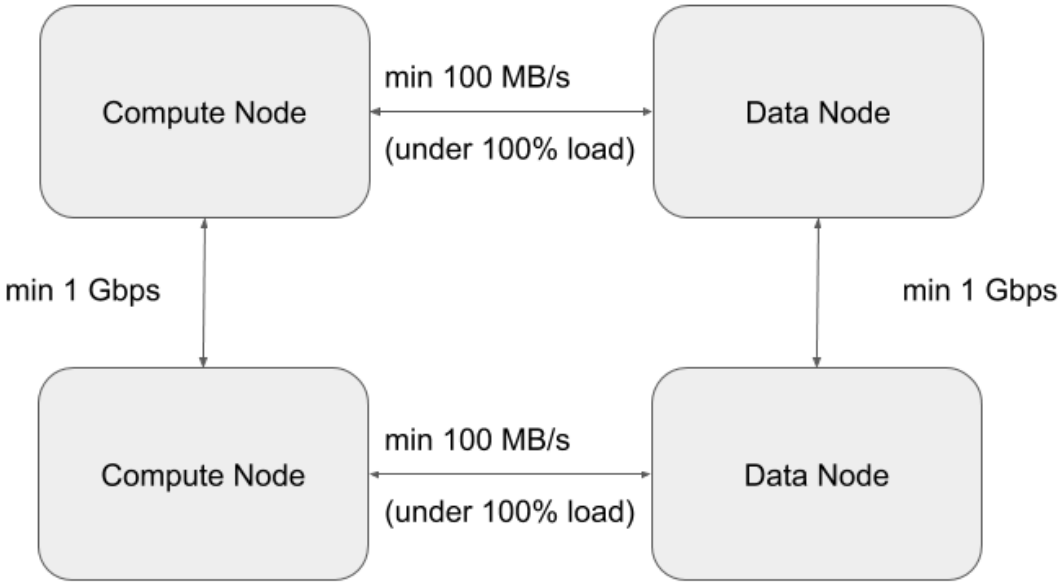
3.16.5. 虚拟专用集群的网络注意事项

3.16.5.1. 最低网络性能要求

虚拟专用集群部署对网络性能具有以下要求：

- 最坏情况下的 IO 吞吐量为 100MB/s，即任何计算节点和任何存储节点之间的网络吞吐量（维持）为 1 Gbp s。为了实现最坏的情况，我们将在*所有计算节点*同时从存储节点读取/写入数据时测量吞吐量。这种类型的并行执行是大数据应用程序的典型特征。
- 任何两个工作负载集群节点或任何两个基本集群节点之间的最坏情况网络带宽为 1Gbps。

下图总结了这些要求：



3.16.5.2. 调整和设计网络拓扑

3.16.5.2.1. 最低和推荐的性能特征

该部分可用于遵循下表中概述的最低网络吞吐量来得出网络吞吐量要求。

注意

就本指南而言，我们将使用以下术语：

- **南北（NS）**流量模式指示计算层和存储层之间的网络流量。
- **东西（EW）**流量模式指示存储或计算集群**内**的**内部网络流量**。

层级	最低每节点存储 IO 吞吐量（MB/s）	最小每节点网络吞吐量（Gbps）（EW + NS）	建议的每节点存储 IO 吞吐量（MB/s）	推荐的每节点网络吞吐量（Gbps）（EW + NS）
计算（可以是虚拟的也可以是裸机的）	100	2	200	4
存储（可以虚拟化或裸机存储）	400	8	800	16

注意

存储后端将确定与其连接的计算节点的最大数量。后端不仅涉及容量，还涉及吞吐量。由于后端将是 HDFS DataNode，因此需要适当调整后端节点（或多个节点）的大小。以下准则将有助于相应地调整后端大小：

- 假设将 SATA 驱动器用于存储，则每个驱动器在裸机集群中可以产生大约 100MB/s 的吞吐量，并且应该期望在直接连接存储的虚拟化集群中产生 70-80MB/s 的吞吐量。
- 每个驱动器需要一个物理 CPU 内核。因此，具有 12 个驱动器的节点应至少具有 12 个核心。
- 规划网络带宽时应考虑 2 倍 NS 流量。例如，如果一个节点有 12 个驱动器，则预期的 NS 流量应为 1200MB/s（1.2GB/s），即网络吞吐量约为 10 Gbps。计划 20 Gbps 以解决 EW 流量。
- 计算层和存储层之间的互连网络需要考虑将在 NS 方向上流动多少吞吐量。以下部分阐明了网络设计的注意事项，并说明了不同程度的超额预订对集群总体可实现吞吐量的影响。

3.16.5.2.2. 网络拓扑注意事项

首选的网络拓扑是主干叶，叶子交换机和主干交换机之间的超额订阅接近 1：1，理想的目标是没有超额订阅。这样一来，我们就可以确保存储和计算节点的任何组合之间的全线速。由于此体系结构要求对计算和存储进行分解，因此网络设计必须更加主动以确保最佳性能。

最低网络吞吐量有两个方面，这也将决定计算与存储节点的比率。

- 存储后端的网络吞吐量和 IO 吞吐量功能。

- 计算层和存储层（NS）之间的网络吞吐量和订阅上的网络。

单步执行示例场景可以帮助您更好地理解这一点。假设这是一个未完成的设置，计算节点和存储节点都是虚拟机（VM）：

- 在 EW 和 NS 流量模式之间共享总网络带宽的因素。因此，对于 1 Gbps 的 NS 流量，我们也应该计划 1 Gbps 的 EW 流量。
- 计算层和存储层之间的网络超额预订为 1：1
- 后端包括 5 个节点（VM），每个节点具有 8 个 SATA 驱动器。
    - 这意味着整个后端的理想 IO 吞吐量（用于计划）约为 4GB/s，即每个节点约 800MB/s。这相当于集群的∽32 Gbps 网络吞吐量，对于 NS 流量模式，每个节点约为∽7 Gbps
    - 考虑到 EW + NS，我们需要每个节点 14 Gbps 来处理每个节点 800MB/s 的 IO 吞吐量。
    - 理想情况下，计算集群将具有以下内容：
    - 5 个虚拟机监控程序节点，每个节点具有 7 Gbps NS + 7 gpbs EW =每个虚拟机监控程序总网络吞吐量为 14 Gbps。
    - 这种情况下，只要在 CPU 和内存方面具有足够的大小，就可以以最少的吞吐量（100MB/s）处理约 6 个节点，从而使后端管道达到饱和（6 x 100 MB/sx 5 = 3000 MB/s）。每个节点应具有∽2 Gbps 的网络带宽，以适应 NS + EW 流量模式。
    - 如果我们考虑建议的每节点 200MB/s 的吞吐量，那么每个虚拟机管理程序将是 3 个这样的节点（3 x 200 MB/sx 5 = 3000 MB/s）。每个节点应具有∽4 Gbps 网络带宽，以适应 NS + EW 交通模式。

假定计算与存储节点的比率为 4：1。这将取决于深入的规模确定，并且将通过对要在所述基础结构上运行的工作负载的充分理解来确定更确定的规模。

下表中的两个表说明了通过一个场景进行规模调整的情况，该场景涉及一个 50 个节点的存储集群，并按照 4：1 的假设，计算了 200 个计算节点。

- 50 个存储节点
- 200 个计算节点（比例为 4：1）

表 1. 存储计算节点级别的大小

层级	每节点 IO（MB/s）	每节点 NS 网络（mbps）	每节点 EW（mbps）	节点数	集群 IO（MB/s）	集群 NS（网络）（mbps）	NS 超额预订

HDFS	400	3200	3200	50	20000	160000	1
计算节点	100	800	800	200	20000	160000	1
计算节点	100	800	800	200	10000	80000	2
计算节点	100	800	800	200	6667	53333	3
计算节点	100	800	800	200	5000	40000	4

表 2. 存储和计算虚拟机管理程序级别的大小

层级	每节点 IO（MB/s）	每节点 NS 网络（mbps）	每节点 EW（mbps）	节点数	虚拟机监控程序超额预订
计算管理程序	600	4800	4800	33	6
计算管理程序	400	3200	3200	50	4
计算管理程序	300	2400	2400	67	3
存储管理程序	1200	9600	9600	17	3
存储管理程序	800	6400	6400	25	2
存储管理程序	400	3200	3200	50	1 个

可以看到，鉴于不同的整合率和不同的吞吐量要求，上表提供了确定私有云每一层的硬件容量的方法。

3.16.5.2.3. 物理网络拓扑

Hadoop 集群的最佳网络拓扑是 spine-leaf（叶脊网络架构）。每个硬件机架都有自己的叶子交换机，每个叶子都连接到每个脊柱交换机。理想情况下，我们不希望在脊柱和叶子之间有任何超额订购。这将导致集群中任何两个节点之间具有完整的线路速率。

交换机、带宽等的选择当然取决于上一节中的计算。

如果存储节点和工作负载节点恰好位于明显分开的机架中，那么有必要确保在工作负载机架交换机和存储机架交换机之间，至少有上行链路带宽与存储设备提供的理论最大值相同集群。换句话说，所有仅工作负载机架的带宽总和必须等于所有仅存储机架的带宽总和。

例如，以上一节中的示例为例，我们在存储集群和工作负载集群节点之间应该至少有 60 Gbps 的上行链路。

如果要基于上面的规模练习构建所需的网络拓扑，则需要以下内容。

层	网络每端口带宽	所需端口数	笔记
工作量	25	52	对于每个大小，我们需要每个节点 20 Gbps。但是，为简化设计，让我们选择单个 25 Gbps 端口，而不是每个节点 10Gbps 的绑定对。
存储	25	43	
总端口		95	

假设所有节点的尺寸均为 2 RU，我们将需要 5 x 42 RU 机架来容纳整个设置。

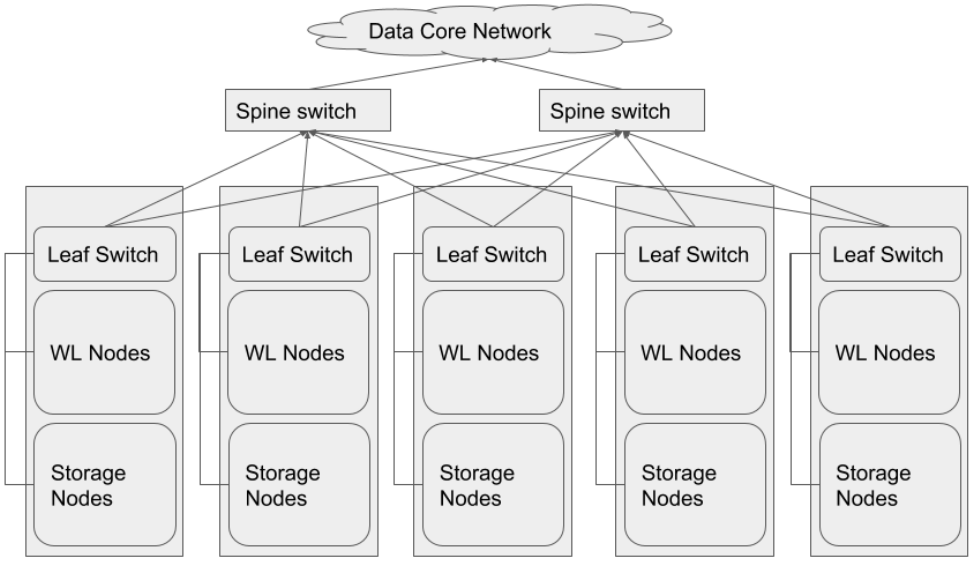
如果我们将每一层的节点尺可能均匀地分布在 5 个机架上，那么最终将得到以下配置。

机架 1	机架 2	机架 3	机架 4	机架 5
脊柱（20 x 100 Gbps 上行链路+ 2 x 100 Gbps 到核心）			脊柱（20 x 100 Gbps 上行链路+ 2 x 100 Gbps 到核心）	
ToR（18 x 25 Gbps + 8 x 100 Gbps 上行链路）	ToR（20 x 25 Gbps + 8 x 100 Gbps 上行链路）	ToR（20 x 25 Gbps + 8 x 100 Gbps 上行链路）	ToR（18 x 25 Gbps + 8 x 100 Gbps 上行链路）	ToR（20 x 25 Gbps + 8 x 100 Gbps 上行链路）
10 x WL	11 x WL	11 x WL	10 x WL	11 x WL
8 x 储存	9 x 储存	9 x 储存	8 x 储存	9 x 储存

因此，这意味着我们必须选择至少具有 20 x 25 Gbps 端口和 8 x 100 Gbps 上行链路端口的 ToR（机架顶部）交换机。另外，Spine 交换机将至少需要 22 x 100 Gbps 端口。

使用来自叶子交换机的八个 100 Gbps 上行链路，将导致叶子（最多 20 x 25 Gbps 端口）和主干（每个主干交换机 4 x 100 Gbps）之间几乎达到 1：1（1.125：1）的超额预订比率。

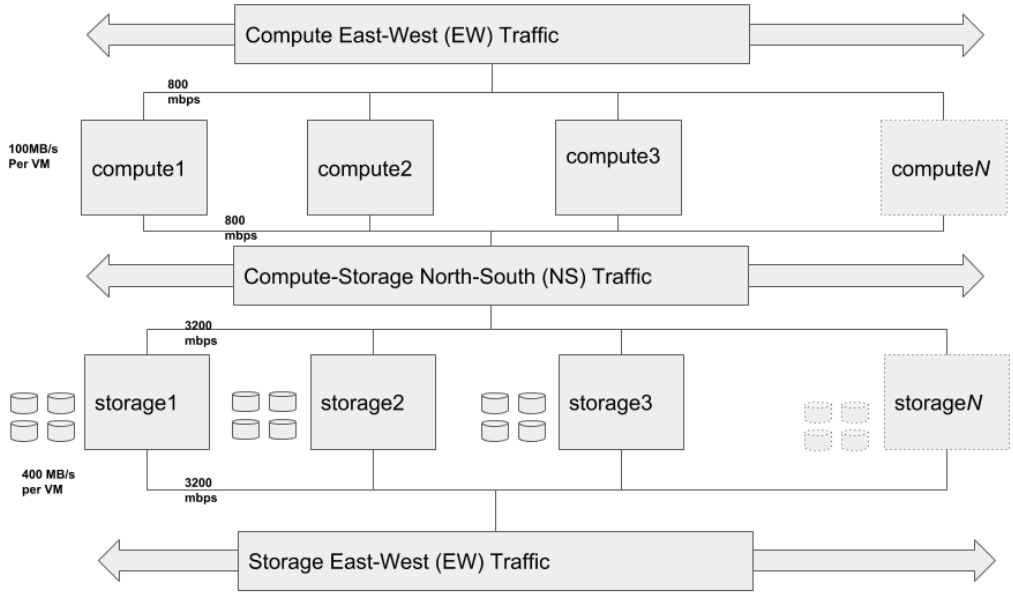
以下面所示的方式混合工作负载和存储节点，将有助于将一些流量本地化到每个叶子，从而减少 NS 流量（工作负载和存储集群之间）的压力。



注意

为了清楚起见，书脊开关已显示在机架外部

下图说明了虚拟机级别的逻辑拓扑。



上面显示的 Storage EW，Compute NS 和 Compute EW 组件不是单独的网络，而是具有不同流量模式的同一网络，为了清楚地表示不同的流量模式，对它们进行了细分。

4.CDP 核心组件

作为业界领先的企业级大数据平台，Cloudera CDP 数据中心版除了包含业界流行的基于开源 Hadoop 及其生态组件构建的 CDP 核心，还包含了很多为支撑企业级业务的高级管理特性。 借助于 Cloudera 数据中心版成熟的整体解决方案，企业可以放心将数据整合在 CDP，进而专注于自己的业务能力。

2019 年底发布的 CDP 7.1 至少包含以下核心 Hadoop 组件以及相应对应开源版本号，以后发布的 CDP 产品所含组件版本将不低于列表所示：

4.1. Cloudera Runtime 组件版本

您必须熟悉 Cloudera Runtime 7.1.6 发行版中所有组件的版本，以确保这些组件与其他应用程序的兼容性。您还必须了解可用的 Technical Preview 组件，并仅在测试环境中使用它们。

Apache 组件

组件版本号由三部分组成，[** Apache 组件版本号**]。[*运行时版本号*]。[*运行时内部版本号*]。例如，如果列出的 Apache HBase 组件版本号是 2.2.3.7.1.6.0-297，则 2.2.3

是上游 Apache HBase 组件版本，7.1.6 是 Runtime 版本，而 297 是 Runtime 构建。您还可以在 Cloudera Manager 中查看组件版本号。

组件	版本
Apache Arrow	0.8.0.7.1.6.0-297
Apache Atlas	2.1.0.7.1.6.0-297
Apache Calcite	1.19.0.7.1.6.0-297
Apache Avro	1.8.2.7.1.6.0-297
Apache Hadoop (Includes YARN and HDFS)	3.1.1.7.1.6.0-297
Apache HBase	2.2.3.7.1.6.0-297
Apache Hive	3.1.3000.7.1.6.0-297
Apache Impala	3.4.0.7.1.6.0-297
Apache Kafka	2.5.0.7.1.6.0-297
Apache Knox	1.3.0.7.1.6.0-297
Apache Kudu	1.13.0.7.1.6.0-297
Apache Livy	0.6.0.7.1.6.0-297
Apache MapReduce	3.1.1.7.1.6.0-297
Apache Ozone	1.0.0.7.1.6.0-297
Apache Oozie	5.1.0.7.1.6.0-297
Apache ORC	1.5.1.7.1.6.0-297
Apache Parquet	1.10.99.7.1.6.0-297
Apache Phoenix	5.1.0.7.1.6.0-297
Apache Ranger	2.1.0.7.1.6.0-297
Apache Solr	8.4.1.7.1.6.0-297
Apache Spark	2.4.5.7.1.6.0-297
Apache Sqoop	1.4.7.7.1.6.0-297

组件	版本
Apache Tez	0.9.1.7.1.6.0-297
Apache Zeppelin	0.8.2.7.1.6.0-297
Apache ZooKeeper	3.5.5.7.1.6.0-297

其他组件

组件	版本
Cruise Control	2.0.100
Data Analytics Studio	1.4.2
GCS Connector	2.1.2
HBase Indexer	1.5.0.7.5.0-297
Hue	4.5.0
Search	1.0.0
Schema Registry	0.10.0.7.1.6.0
Streams Messaging Manager	2.1.0
Streams Replication Manager	1.0.0

连接器和加密组件

组件	版本
HBase connectors	1.0.0
Hive Meta Store (HMS)	1.0.0
Hive on Tez	1.0.0
Hive Warehouse Connector	1.0.0
Spark Atlas Connector	0.1.0
Spark Schema Registry	1.1.0

概括来讲，CDP 的主要技术特性如下：

• 高度集成的Hadoop平台： 一个整体的数据存储和计算平台，无缝集成了基于Hadoop的大量生态工具，不同业务可以集中在一个平台内完成，而不需要在处理系统间移动数据；用廉价的PC服务器架构统一的存储平台，能存储PB级海量数据。并且数据种类可以是结构化，半结构化及非结构化数据。存储的技术有SQL及NoSQL，并且NoSQL能提供企业级的安全方案。CDP提供统一的资源调度平台，能够利用最新的资源调度平台YARN分配集群中CPU,内存等资源的调度，充分利用集群资源；

• 多样的数据分析平台 – Cloudera CDP数据中心版能够针对不用的业务类型提供不同的计算框架，比如针对批处理的Tez计算框架；针对交互式查询的Impala MPP查询引擎；针对内存及流计算的Spark框架；针对机器学习，数据挖掘等业务的训练测试模型；针对全文检索的Solr搜索引擎

• 增强的数据安全及审核技术 – 通过Atlas+Ranger可以提供统一的认证，授权，审核，加密等技术，充分保证大数据平台的安全。能够与企业现有的AD,LDAP等系统集成进行用户身份认证及授权；同时能够通过图形化的方式方便追踪大数据集的来源，访问情况；对安全级别比较高的数据，能够提供透明的数据加密/解密技术

• 高度可管理（Cloudera Manager） – 平台提供了企业级应用需要的高可用性、高容错和自愈性，同时还包括自动化备份和系统灾备；端到端的图形化管理工具能提供管理，诊断，监控，集成等功能，保证企业的大数据平台可靠运行。能够通过图形化的管理工具部署，管理大数据平台；能够提供主动诊断及支持服务；能够实现一键配置高可靠以保障系统稳定运行；同时，要保证系统无宕机滚动升级

• 高度开放 – 基于开源Hadoop构建的数据存储和处理平台，保证数据和应用对用户是开放的，很好的与第三方系统集成，同时给用户扩展应用提供了保证，从而保护用户的投资。

4.2. 分布式文件系统 HDFS

HDFS（ Hadoop Distributed File System），是一个分布式文件系统。它具有高容错性的特点，可以被广泛的部署于廉价的 PC 之上。它以流式访问模式访问应用程序的数据，这大大提高了整个系统的数据吞吐量，能够满足多来源、多类型、海量的数据存储要求，因而非常适用于日志详单类非结构化数据的存储。

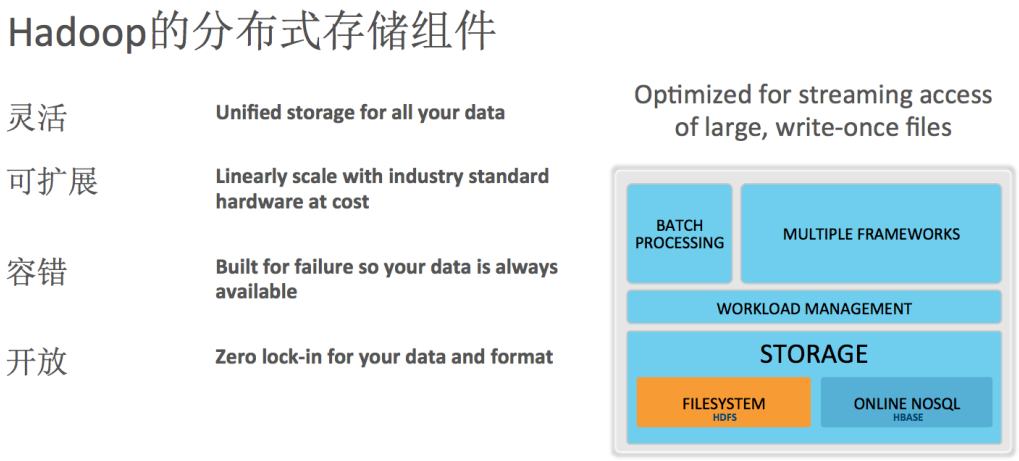
HDFS 架构采用主从架构（master/slave）。一个典型的 HDFS 集群包含一个 NameNode 节点和多个 DataNode 节点。NameNode 节点负责整个 HDFS 文件系统中的文件的元数据保管和管理，集群中通常只有一台机器上运行 NameNode 实例，DataNode 节点保存文件中的数据，集群中的机器分别运行一个 DataNode 实例。在 HDFS 中，NameNode 节点被称为名字节点，DataNode 节点被称为数据节点，DataNode 节点通过心跳机制与 NameNode 节点进行定时的通信。

HDFS 可以实现大规模数据可靠的分布式读写。HDFS 针对的使用场景是数据读写具有“一次写，多次读”的特征，而数据“写”操作是顺序写，也就是在文件创建时的写入或者在现有文件之后的添加操作。HDFS 保证一个文件在一个时刻只被一个调用者执行写操作，而可以被多个调用者执行读操作。其主要特性如下：

- 灵活：统一的存储可以存放结构化，半结构化及非结构化数据
- 可扩展：根据业务需要增加 PC 服务器实现存储扩容
- 容错：数据有多个副本以保障数据的可靠性
- 开放：基于开源的存储格式，避免厂商锁定

高级特性：

Cloudera 整合最新版本的 HDFS，实现了对内存，SSD,磁盘相结合的异构式统一存储体系，将数据按照冷热程度不同自动存储在不同存储介质中，既可以利用内存，SSD 的高性能加速实时数据分析过程，也能利用磁盘的低成本大容量存储较冷的数据。



CDP 提供 NameNode 的 HA 功能，通过配置 Active/Standby 两个 NameNodes 实现在集群中对 NameNode 的热备来解决单点故障问题。如果出现故障，如机器崩溃或机器需要升级维护，这时可通过此种方式将 NameNode 自动切换到另外一台机器。

HDFS 的缺省附本数为 3 份，CM 的 BDR 组件提供了灾备功能，能对 HDFS 以及 HBase 进行快照、自动容灾备份。

4.3. 实时数据库 HBase

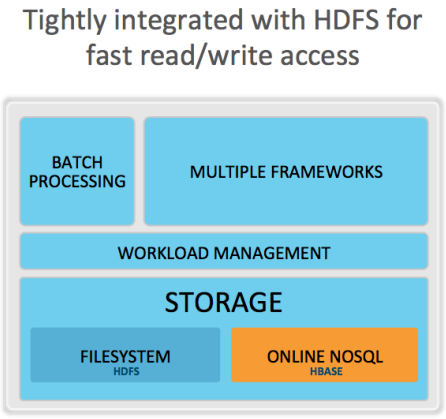
HBase 是一个高可靠性、高性能、面向列、可伸缩的分布式存储系统，它利用 Hadoop HDFS 作为其文件存储系统，利用 Hadoop MapReduce 来处理 HBase 中的海量数据，利用 Zookeeper 作为协同服务。HBase 不是一个关系型数据库，其设计目标是用来解决关系型数据库在处理海量数据时的理论和实现上的局限性。HBase 从一开始就是为 Terabyte 到 Petabyte 级别的海量数据存储和高速读写而设计，这些数据要求能够被分布在数千台普通服务器上，并且能够被大量并发用户高速访问。

基于Hadoop的NoSQL数据库

具有所有HDFS特性

Flexible	Unified storage for all your data
Scalable	Linearly scale with industry standard hardware at cost
Open	Zero lock-in for your data and format

同时，满足实时数据存取



存储在 HBase 中的表的典型特征：

- 大表（BigTable）：一个表可以有上亿行，上百万列
- 面向列：面向列(族)的存储、检索与权限控制
- 稀疏：表中为空(null)的列不占用存储空间

高级特性：

1. Cloudera CDP 数据中心版支持 SQL on HBase 特性，支持对数据表建立 Local Index 和 Global Index，执行速度远远超过原生 HBase API，同时提供完善的 SQL 接口供客户端使用。

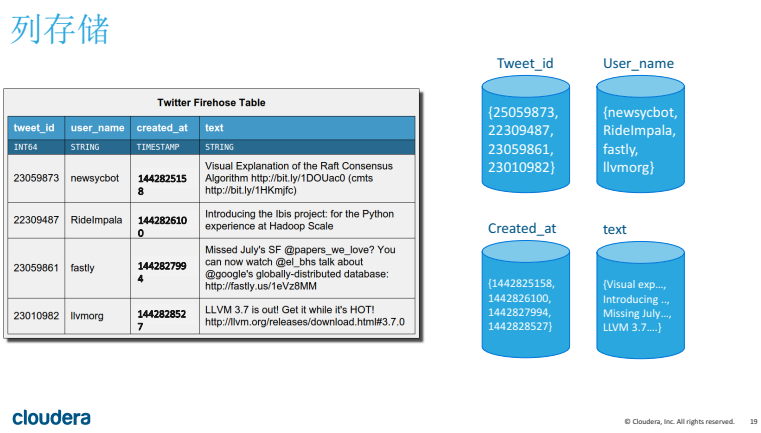
2. Cloudera CDP 支持 HBase 的大对象存储（LOB）功能，将 HBase 进化为文档数据库，特别适合存储单个大小数十 K 至数十 M 的非结构化文档，即使对于十亿级别的 LOB 文档数据表仍能做到毫秒级增删改查操作，同时支持所有 HBase 原生特性，与上层 HBase 应用 100%兼容。

3. Cloudera CDP 支持 HBase 的跨机房，跨数据中心统一数据视图，不同数据中心数据可以彼此隔离，彼此独立授权访问，同时又能通过统一的 API 提供跨数据中心虚拟大表的支持；同时支持跨数据中心高可用性功能，单个数据中心崩溃不会导致数据表访问失败。

HBase 提供命令行或者 Java API 的方式对指定表的 Region 进行动态的分割和合并。基于底层 HBase 组件，上层使用 Hive、Impala 以及 Phoenix 提供标准的 SQL 语句支持，包括 DML 类的数据定义，DDL 类的增删查改，结合使用 Phoenix 提供辅助索引。

4.4. 列式存储引擎 Kudu

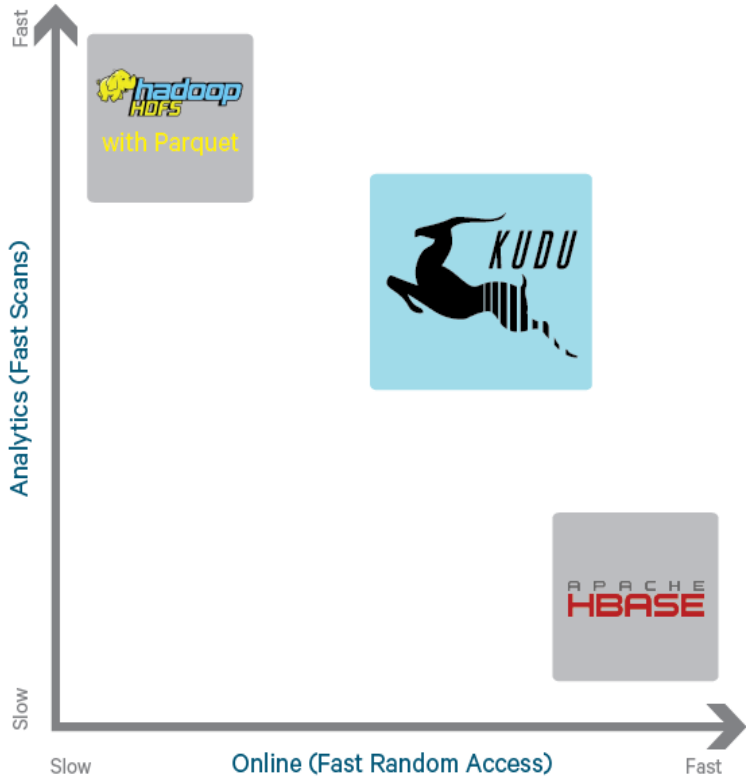
Kudu 是一种全新的列存储引擎，完全独立于 HDFS 文件系统，它的设计借鉴了传统的列存储数据库，例如 Vertica，一张 Kudu 表的数据是按照 Column 进行存储的，如下图所示：



Kudu 的设计目标是在设计一种全新的存储引擎，既支持 HDFS 的高性能全表扫描，也支持 HBase 的随机 IO 操作和数据更新，可以说 Kudu 是拥有 HDFS 和 HBase 两者优点的跨界产品。

Kudu 拥有以下特点：

- **扫描大数据量时吞吐率高**(列式存储和多副本机制)，性能是 Parquet 的 2/3 左右
- **访问少量数据时延时低**(主键索引和多数占优复制机制)，性能是 HBase 的 1/2 到 2/3 之间
- 支持数据的更新和删除
- 类似的数据库语义(初期支持单行记录的 ACID)
- **关系数据模型**
    - SQL 查询
    - “NoSQL”风格的扫描/插入/更新(Java 客户端)



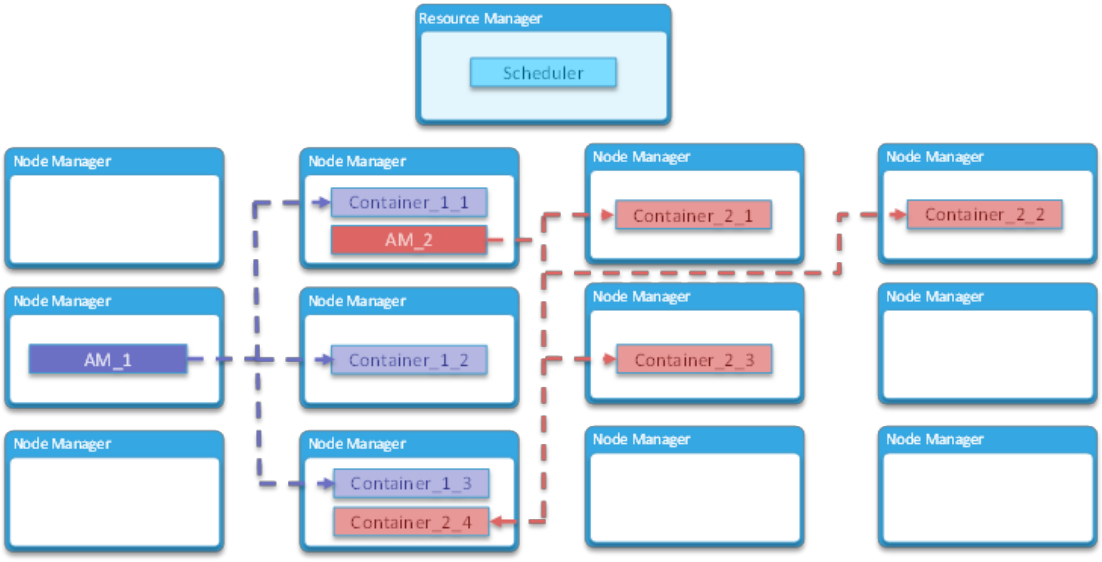
4.5. 统一资源管理和调度框架

为了实现一个 Hadoop 集群的集群共享、可伸缩性和可靠性，并消除早期 MapReduce 框架中的 JobTracker 性能瓶颈，开源社区引入了统一的资源管理框架 YARN。YARN 分层结构的本质是 ResourceManager。这个实体控制整个集群并管理应用程序向基础计算资源的分配。ResourceManager 将各个资源部分（计算、内存、带宽等）精心安排给基础 NodeManager（YARN 的每节点代理）。ResourceManager 还与 ApplicationMaster 一起分配资源，与 NodeManager 一起启动和监视它们的基础应用程序。在此上下文中，ApplicationMaster 承担了以前的 TaskTracker 的一些角色，ResourceManager 承担了 JobTracker 的角色。

ApplicationMaster 管理一个在 YARN 内运行的应用程序的每个实例。ApplicationMaster 负责协调来自 ResourceManager 的资源，并通过 NodeManager 监视容器的执行和

资源使用（CPU、内存等的资源分配）。请注意，尽管目前的资源更加传统（CPU 核心、内存），但未来会带来基于手头任务的新资源类型（比如图形处理单元或专用处理设备）。从 YARN 角度讲，ApplicationMaster 是用户代码，因此存在潜在的安全问题。YARN 假设 ApplicationMaster 存在错误或者甚至是恶意的，因此将它们当作无特权的代码对待。

NodeManager 管理一个 YARN 集群中的每个节点。NodeManager 提供针对集群中每个节点的服务，从监督对一个容器的终生管理到监视资源和跟踪节点健康。MRv1 通过插槽管理 Map 和 Reduce 任务的执行，而 NodeManager 管理抽象容器，这些容器代表着可供一个特定应用程序使用的针对每个节点的资源。



Resource Manager 里的 Application Manager 会为每一个 application 在 NodeManager 里面申请一个 container，然后在该 container 里面启动一个 Application Master。containe 启动时便会相应启动一个 JVM。YARN 中 uber 功能被启用，并且该 application 被认为是一个“小的 application”，那么 Application Master 便会将该 application 包含的每一个 task 依次在这个 container 里的 JVM 里顺序执行，直到所有 task 被执行完。这样 Application Master 便不用再为每一个 task 向 Resource Manager 去申请一个单独的 container，最终达到了 JVM 重用（资源重用）的目的。

CDP 中 YARN 缺省使用 Capacity 调度器。Capacity Scheduler 中的每个队列都具有以下属性：

一个短的队列名称。

完整的队列路径名称。

关联的子队列和应用程序的列表。

队列的保证容量。

队列的最大容量。

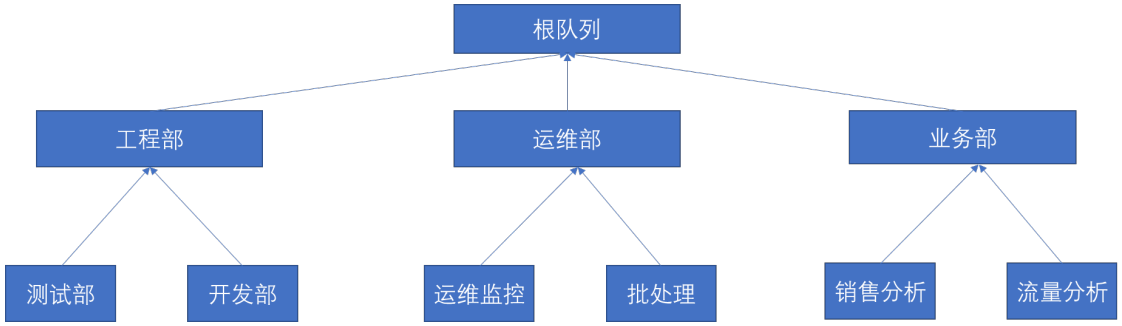
活动用户列表及其相应的资源分配限制。

队列的状态。

访问控制列表（ACL）。

其中每个队列的容量指定提交给队列的应用程序可用的群集资源的百分比。 队列可以设置为多层次结构，对队列中各个组织，组和用户所需的资源要求和访问进行细粒度控制。

例如，假设一家公司有三个组织：工程部，运维部和业务部。 工程部有两个子小组：开发和测试。 运维部有两个小组：监控和服务。 最后，业务部分为销售分析和流量预测。 下图显示了此示例的队列层次结构：



每个子队列通过 capacity-scheduler.xml 文件中的 yarn.scheduler.capacity<queue-path>.queues 配置属性绑定到其父队列。 父层队列“运维”，“工程”和“业务”队列将被绑定到“根”队列，

分层队列特征

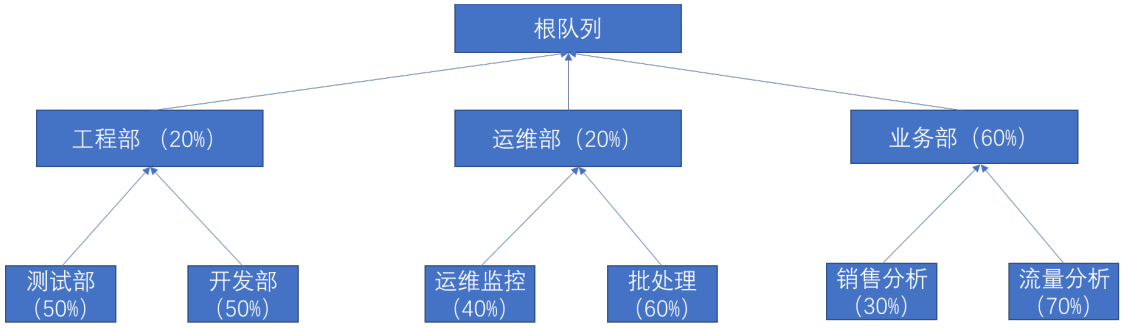
有三种类型的队列：根队列，父队列和叶队列。

根队列：有一个不属于任何组织的最顶层，是表示集群本身的全部资源。

父队列：可以跨组织和子组织管理资源。 它们可以包含更多的父队列或叶队列。 他们自己不直接接受任何申请。

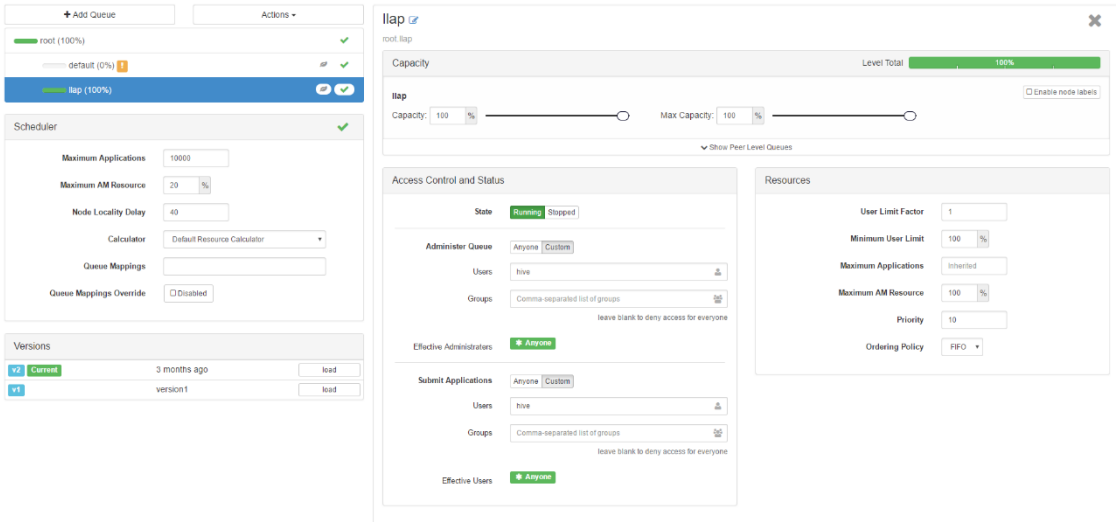
叶队列：是居住在父队列下的队列并接受应用程序。 叶队列没有任何子队列，因此没有以“.queues”结尾的任何配置属性。

通过使用父和子队列，管理员可以为各个组织和子组织指定容量分配。例如下图



这里的容量是指被 Yarn 所管理的内存容量百分比，其中同级队列的容量总和不得大于 100%。这里假设根队列有 100G 内存，则工程部可以使用 20G，而工程部的叶子队列开发部则只有 10G 的内存配额。

Cloudera Manager 有图形化的队列资源分配及管理，详见下图

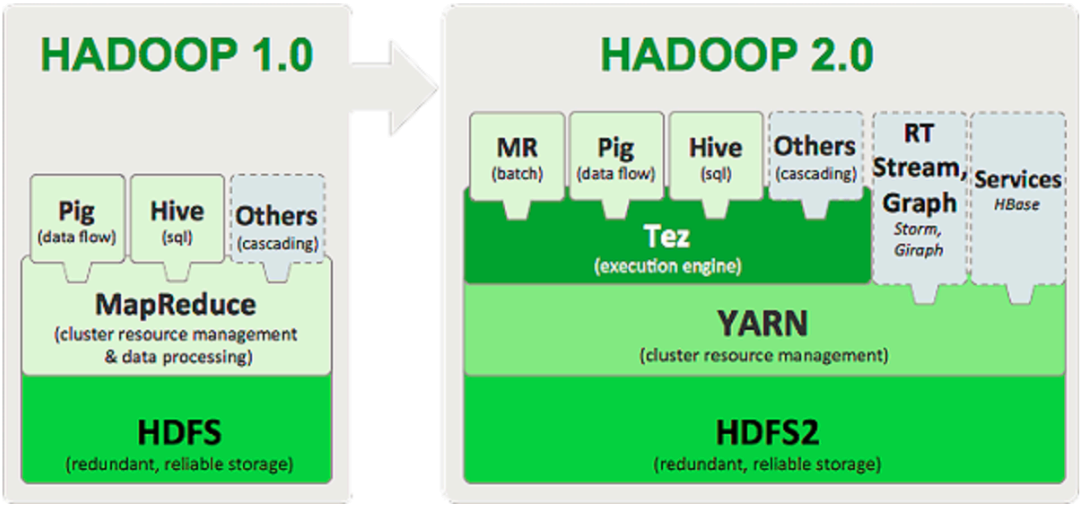


分层队列确保首先在组织的子队列之间共享有保证的资源，然后剩余的空闲资源与属于其他组织的队列共享。这使每个组织都能够控制其保证资源的利用。

在层次结构中的每个级别，每个父队列根据需求以排序的方式保存其子队列的列表。队列的排序由每个队列容量的当前使用分数（或者如果任何两个队列的保留容量相等，全路径队列名称）确定。根队列了解如何在第一级父队列中分配群集容量，并调用其每个子队列上的调度。每个父队列将其容量约束应用于其所有子队列。叶队列包含活动应用程序列表（可能来自多个用户），并以 FIFO（先进先出）方式调度资源，同时遵守为个别用户指定的容量限制。

4.6. 分布式计算框架 – Tez

Tez 是 Apache 最新的支持 DAG 作业的开源计算框架，它可以将多个有依赖的作业转换为一个作业从而大幅提升 DAG 作业的性能。Tez 并不直接面向最终用户——事实上它允许开发者为最终用户构建性能更快、扩展性更好的应用程序。Hadoop 传统上是一个大量数据批处理平台。但是，有很多用例需要近乎实时的查询处理性能。还有一些工作则不太适合 MapReduce，例如机器学习。Tez 的目的就是帮助 Hadoop 处理这些用例场景。

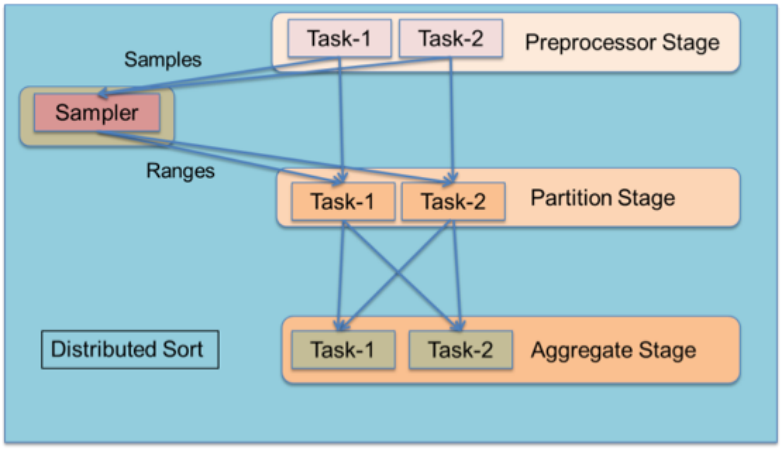


Tez 项目的目标是支持高度定制化，这样它就能够满足各种用例的需要，让人们不必借助其他的外部方式就能完成自己的工作。它直接源于 MapReduce 框架，核心思想是将 Map 和 Reduce 两个操作进一步拆分，即 Map 被拆分成 Input、Processor、Sort、Merge 和 Output， Reduce 被拆分成 Input、Shuffle、Sort、Merge、Processor 和 Output 等，这样，这些分解后的元操作可以任意灵活组合，产生新的操作，这些操作经过一些控制程序组装后，可形成一个大的 DAG 作业。总结起来，Tez 有以下特点：

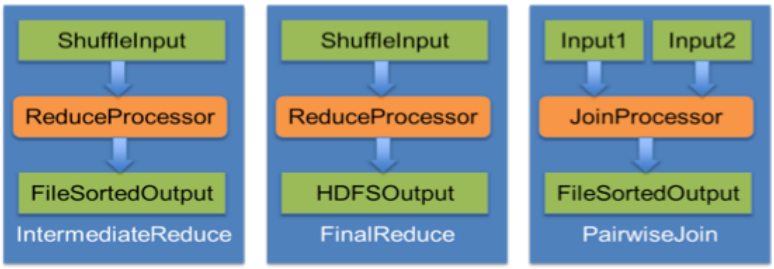
1） 运行在 yarn 上；

2） 适用于 DAG（有向图）应用（可与 hive 和 pig 结合）；

3) 具有表现力的数据流 API——Tez 团队希望通过一套富有表现力的数据流定义 API 让用户能够描述他们所要运行计算的有向无环图 （DAG）。为了达到这个目的，Tez 实现了一个结构化类型的 API，你可以在其中添加所有的处理节点和边，并可视化实际构建的图形；下图是一个实现 terasort（分布式排序）过程的流程图：



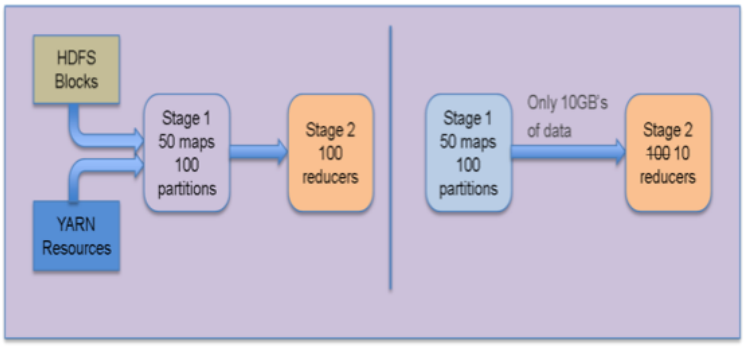
4) 灵活的输入—处理器—输出（Input-Processor-Output）运行时模型——可以通过连接不同的输入、处理器和输出动态地构建运行时执行器；Tez 将用户数据流图的每个顶点看作是一个 input-processor-output 模型。Input&output 决定了数据的格式和怎样的被读取和写入，processor 是对数据的处理逻辑。将这些任务模型进行组装就可以形成不同功能的 DAG 图，连接不同 processor 的 input&output 需要互相兼容



5) 数据类型无关性——仅关心数据的移动，不关心数据格式（键值对、面向元组的格式等，mr 需要指定 map 和 reduce 的输入输出格式）；

6) 简单地部署——Tez 完全是一个客户端应用程序，它利用了 YARN 的本地资源和分布式缓存；

7) DAG 图在运行时的优化和重新配置——分布式数据处理具有动态的特性，可以在执行前提前预知并选出最优的优化策略。Tez 可以在执行过程中通过收集 tasks 的信息更改数据流图。例如下图，运行时，当获知数据量有 10GB 的时候，自动将 reducer 的数量进行更改



4.7. 数据仓库组件 - Hive

Hive 是建立在 Hadoop 上的数据仓库基础构架。它提供了一系列的工具，可以用来进行数据提取转化加载（ETL），这是一种可以存储、查询和分析存储在 Hadoop 中的大规模数据的机制。Hive 定义了简单的类 SQL 查询语言，称为 HQL，它允许熟悉 SQL 的用户查询数据。同时，这个语言也允许熟悉 MapReduce 开发者的开发自定义的 mapper 和 reducer 来处理内建的 mapper 和 reducer 无法完成的复杂的分析工作。

Hive 体系结构：

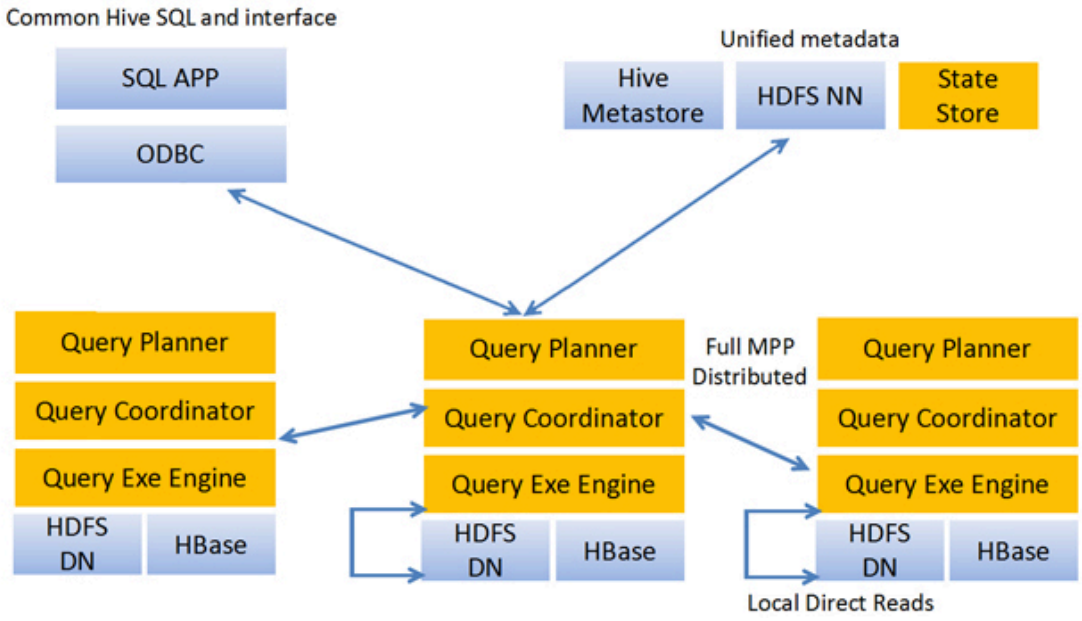
- 用户接口：用户接口主要有三个：CLI，Client 和 WUI。其中最常用的是 CLI，Cli 启动的时候，会同时启动一个 Hive 副本。 Client 是 Hive 的客户端，用户连接至 Hive Server。在启动 Client 模式的时候， 需要指出 Hive Server 所在节点，并且在该节点启动 Hive Server。 WUI 是通过浏览器访问 Hive。
- 元数据存储：Hive 将元数据存储在数据库中，如 mysql、derby。Hive 中的元数据包括表的名字，表的列和分区及其属性，表的属性（是否为外部表等），表的数据所在目录等。

Hive的DML语言可以通过与传统数据库建表语句类似的语法，将结构化的数据文件映射为数据库表。并提供常用的SQL支持。

Cloudera发起的Hive on Tez项目就是把Tez作为Hive的一个计算引擎，将Hive的SQL语句查询查询作为Tez的作业提交到集群上进行计算。同时提供了参数设置控制此功能开关，最新版本的Hive缺省使用Tez作为Hive的计算引擎。

4.8. SQL 分析引擎 Impala

Cloudera Impala 是运行于 Apache Hadoop 之上业界领先的大规模并行处理（MPP）SQL 查询引擎。Impala 是 Apache-licensed 并且开源的项目。它将时下流行的分布式并行数据库技术和 Hadoop 进行结合，帮助用户能够直接查询存储于 Hdfs 和 Hbase 的数据而不用进行数据迁移或者转变。 Impala 设计之初就定位为 Hadoop 生态系统的一部分，因此，Impala 和 MapReduce，Hive，Pig 以及 Hadoop 的其他组件，都享有共同的灵活的文件和数据格式。其架构如下图所示：



Impala 使用了 Hive 的 SQL 接口（包括 SELECT、 INSERT、Join 等操作），，表的元数据信息存储在 Hive 的 Metastore 中。StateStore 是 Impala 的一个子服务，用来监控集群中各个节点的健康状况，提供节点注册、错误检测等功能。 Impala 在每个节点运行了一个后台服务 Impalad，Impalad 用来响应外部请求，并完成实际的查询处理。Impalad 主要包含 Query Planner、Query Coordinator 和 Query Exec Engine 三个模块。QueryPalnner 接收来自 SQL APP 和 ODBC 的查询，然后将查询转换为许多子查询，Query Coordinator 将这些子查询分发到各个节点上，由各个节点上的 Query Exec Engine 负责子查询的执行，最后返回子查询的结果，这些中间结果经过聚集之后最终返回给用户。

Impala 的查询效率比 Hive 有数量级的提升。从技术角度上来看，Impala 之所以能有好的性能，主要有以下几方面的原因:

- Impala 不需要把中间结果写入磁盘，省掉了大量的 I/O 开销。
- 省掉了 MapReduce 作业启动的开销。MapReduce 启动 task 的速度很慢（默认每个心跳间隔是 3 秒钟），Impala 直接通过相应的服务进程来进行作业调度，速度快了很多。
- Impala 完全抛弃了 MapReduce 这个不太适合做 SQL 查询的范式，而是像 Dremel 一样借鉴了 MPP 并行数据库的思想另起炉灶，因此可做更多的查询优化，从而省掉不必要的 shuffle、sort 等开销。
- 通过使用 LLVM 来统一编译运行时代码，避免了为支持通用编译而带来的不必要开销。
- 用 C++实现，做了很多有针对性的硬件优化，例如使用 SSE 指令。
- 使用了支持 Data locality 的 I/O 调度机制，尽可能地将数据和计算分配在同一台机器上进行，减少了网络开销。

在 Impala 出现之前，如果关系型数据库的数据量已经达到数据库容量上限，用户为了维持系统性能不得不对其做相应的扩展。如果用户已经在使用能够分析任何种类数据的 Hadoop，但是需要交互式响应，用户就不得不把数据转移到一个快速的关系型数据库。这种方法不仅需要用户承担复制，存储以及同步数据的成本，同时需要接受不够灵活的 schema，接受转移数据中不可避免的数据丢失，接受只能在目标关系型数据库中做有限的分析。

Impala 的出现使得用户增加了一个选择。Impala 能够使得数据分析师或者数据科学家，使用现存的 BI 工具，直接操作存储于 Hadoop 的数据进行交互。不仅如此，Impala 可以减轻现有分析数据库的压力从而避免了 BI 任务堆积。Impala 具有以下特性：

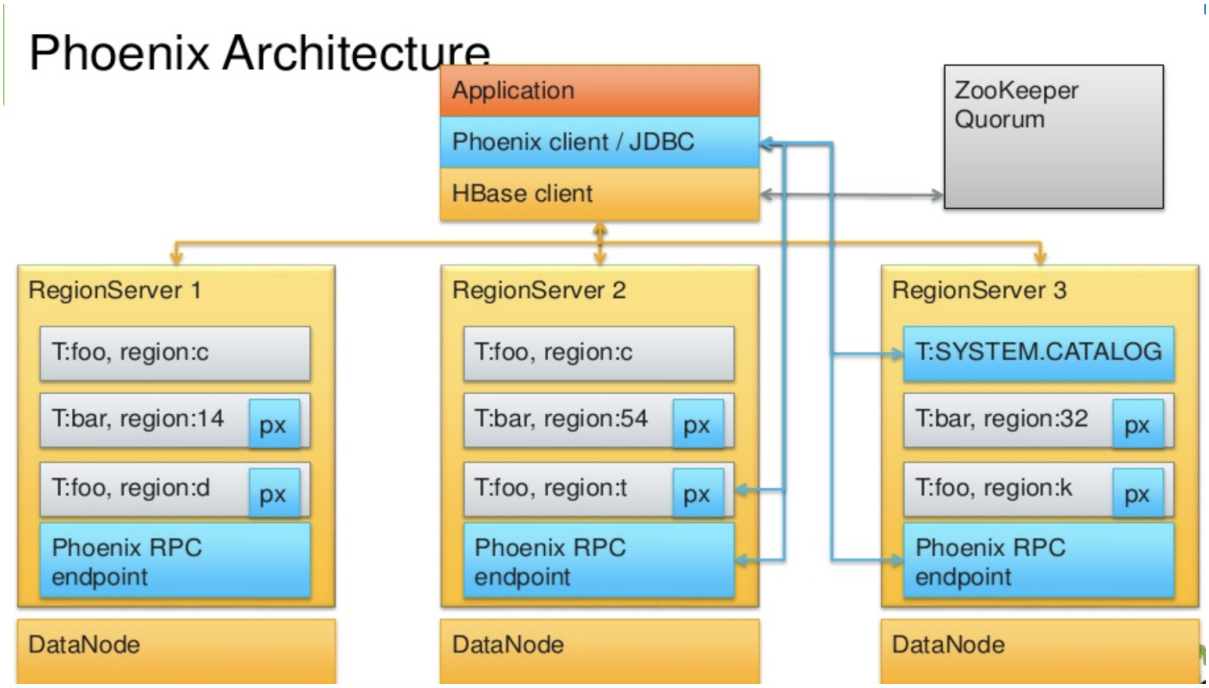
- **高性能：**具有和目前领先的 MPP 数据库有匹配的性能，比 Apahce hive/Stinger 快 10-100 倍
- **Time-to-Insigt：**Impala 具有秒级，分钟级别的响应速度，能够让用户更快的从数据中获得有用的知识。
- **节省成本：**使用 Impala，不需要对存储于 Hadoop 中的数据进行迁移或者转换，从而节省成本。
- **更加完整的分析：**可以完整的分析原始数据或者历史数据，而不用

担心数据在转移过程中的损失。

- **用户熟悉度：**Impala 使用标准的 SQL，查询数据，用户可以通过操作传统数据库一样的方式操作 Impala。
- **安全性高：**提供 Kerberos 认证和基于角色授权。
- **开源：**Apache 许可。

4.9. HBase SQL 查询引擎 Phoenix

Phoenix 是一个开源的 HBASE SQL 层。它不仅可以使用标准的 JDBC API 替代 HBASE client API 创建表，插入和查询 HBASE，也支持二级索引、事物以及多种 SQL 层优化。



Phoenix 的架构如上图所示：

- Phoenix Client/JDBC

通过 JDBC 接口接受来自客户端的查询，对于写查询，phoenix 客户端会把 SQL 请求转换成 hbase client api，并提交给 RegionServer；对于读查询，客户端会调用部署在 HBase 集群的协处理器。

- Metadata 表

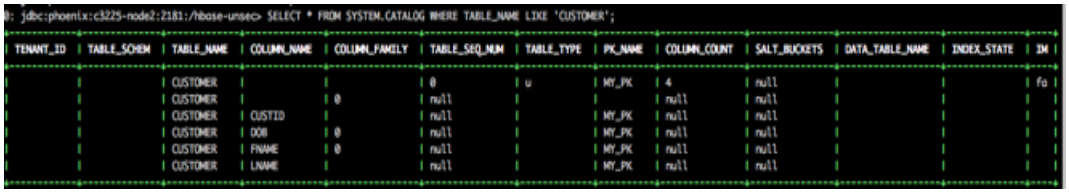
Phoenix Metadata 表存放在 HBase 中，主要包括 Phoenix 表定义，以及统计信息。

- Coprocessor

Phoenix 在 HBase 中部署了客户化的协处理器，主要用于读请求，例如扫描，过滤，汇总，连接等。

Phoenix 基于 HBase 之上创建关系模型，phoenix 表需要定义唯一主键 PRIMARY KEY。创建 Phoenix 表导致创建一张 HBase 表，Phoenix 表中的列映射到 HBase 中的 column qualifier，缺省情况下, 对于 HBase 表中创建新的 Column Family ‘0’，Phoenix 元数据保存在 ‘SYSTEM:CATALOG’。

PHOENIX元数据



HBase Descriptor

二级索引是 Phoenix 最重要的功能之一，主要特点包括：

- 可以在数据表或视图上创建二级索引
- 使用自定义协处理器跨索引同步数据
- 随着数据的变化，索引将自动与表保持同步
- 优化器选择最佳查询计划

- 在非rowkey上创建的二级索引，这些索引可以允许更快进行点查找，并且不需要完整表扫描

4.10. Cloudera 整合全文检索引擎

Cloudera Search 为 CDP 数据中心版提供了一站式全文检索功能，Cloudera Search 是一个综合，灵活和强大的搜索解决方案，它可以直接与通过 EDH 提供的其他工作负载一起工作，如批处理，交互式 SQL 和先进的分析。

Cloudera Search 是一个完全开源的搜索解决方案，内置功能丰富的和可扩展的 Apache Solr 的项目。 Apache Solr 包括开源项目如 Apache Lucene 和 Apache Tika。

用户通过 Cloudera Search 可以不写任何程序代码，简单进行配置即可实现数据源 ETL，创建并更新索引，数据入库定制查询界面等完整的数据全文检索工作流程，高度自动化的流水线提高了部署效率。

- **Cloudera Search 组件**

Search 与已有的 CDP 组件交互使用， Search 使用他们中的许多组件来解决不同的问题。下表列出来了 CDP 组件为 Search 进程做作的贡献，以及每个组件以什么样的方式帮助到 Search， 这些组件共同组成了 Cloudera Search 的整体流程：

组件	贡献
HDFS	源文档一般被存放在 HDFS 中。这些文档被建立索引并被变成可被搜索的。 这些文件，如 Lucene 索引文件，被存放在 HDFS 中的直写日志，支持 Search。使用 HDFS 提供了更简单，基数更大，冗余和容错的供应。由于使用 HDFS 的结果，Search 服务器本质上是无状态的， 这意味着有在节点故障时产生最小的后果。

	HDFS 中还提供了额外的好处，如快照，跨群集复制，和灾难恢复。
MapReduce	Search 包括提前建立的基于 MapReduce Job。Job 可以被按需用来或者被安排来为存储在 HDFS 中任何支持的数据集合来建立索引。Job 为可扩展的批量索引优化了集群资源。
Flume	Cloudera Search 包括 Flume Sink 来把事件直接写入到部署在集群中的到索引器中，使其可以在摄入过程中建立数据索引。
ZooKeeper	协同分布式数据和元数据，也被 Search 的分片所用。ZooKeeper 提供自动的故障切换，增加了服务的弹性。
HBase	支持存储后数据的索引，提取列，列族和键信息作为字段。 因为 HBase 没有使用二级索引，Search 可以完成 HBase 里面行和表内容的全文索引
Cloudera Manager	部署，配置，管理和监控搜索过程和在集群服务中的资源优化，搜索不要求一定有 Cloudera Manager，但是 Cloudera Manager 帮助简化了搜索管理。
Impala	进一步的分析搜索结果
Sqoop	批量摄入数据，并让批量建立索引过程中数据可用

- Lily HBase Indexer

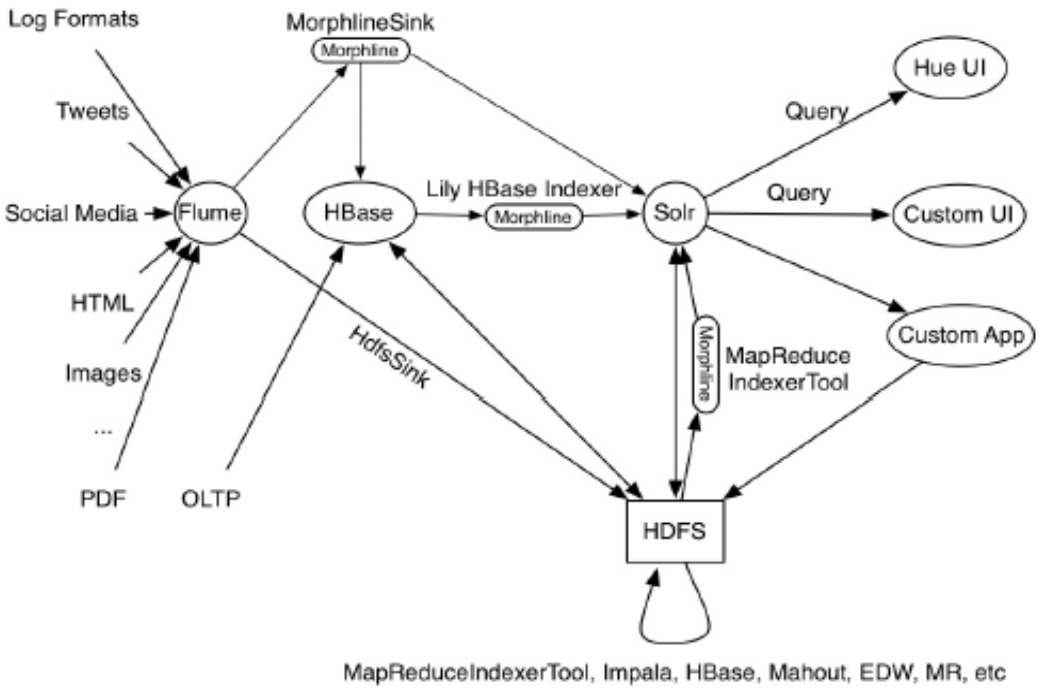
Cloudera Search 集成了 HBase，可实现存储在 HBase 中的数据的全文检索。这个功能， 不影响 HBase 的性能，它是基于一个监听器来监视复制事件流。监听器捕获每个写入或更新复制的事件，这会触发数据提取和映射。事件然后直接发送到 Solr 索引器中，被部署在 HDFS，并写入索引到 HDFS 中，使用相同的过程为 Cloudera Search 的其

他索引工作负载所使用。这些索引可以立即提供服务，从而实现对 HBase 数据的近实时搜索。

Lily HBase Indexer 提供为任何存储在 HBase 中的内容进行快速和简单搜索的能力。它允许你快速和简单地建立 HBase 行索引到 Solr 中，而不需要写审任何一行代码。

Lily HBase 的索引服务是一个灵活的，可扩展的，可容错的，事务性的，面向近实时（NRT）的系统，它可以处理源源不断的 HBase 单元，并更新流到实时的搜索索引中。 通常情况下，从使用 Flume Sink 数据摄取开始到这些内容可能出现在搜索结果中的时间是在秒钟级别的，不过此持续时间是可调的。Lily HBase 的索引使用 Solr 来对索引数据并存储在 HBase 中。

由于 HBase 适用在插入，更新和删除 HBase 的表单元格，索引用标准的 HBase 复制功能使 Solr 与 HBase 表格内容保持一致。 索引器支持灵活自定义特定的应用程序规则来提取，转换和加载 HBase 数据到 Solr 中。 Solr 搜索结果可以包含 ColumnFamily：qualifier, 链接回到存储在 HBase 中的数据。这样，应用程序可以使用搜索结果集直接访问匹配的原始 HBase 单元格。 索引和搜索不影响 HBase 运行的稳定性和写吞吐量，因为索引和搜索过程对于 HBase 来讲是独立和异步的， 以下是完整的 Cloudera Search 架构图:



4.11. 分布式内存计算框架 – Spark

Apache Spark 是一个开源的，通用的分布式集群计算引擎。Spark 发展历程：

Cloudera Spark 是一个开源的，并行数据处理框架，能够帮助用户简单的开发快速，统一的大数据应用，对数据进行，协处理，流式处理，交互式分析等等。Spark 具有如下特点：

- 快速：数据处理能力，比 Mapreduce 快 10-100 倍。
- 易用：可以通过 Java，Scala，Python，简单快速的编写并行的应用

处理大数据量，Spark 提供了超过 80 种高层的操作符来帮助用户组件并行程序。

- 普遍性： Spark 提供了众多高层的工具，例如 Spark SQL，MLib，GraphX，Spark Streaming，可以在一个应用中，方便的将这些工具进行组合。

- 与 Hadoop 集成： Spark 能够直接运行于 Hadoop 2.0 以上的集群，并且能够直接读取现存的 Hadoop 数据。尤其，Spark 和 CDP 紧密结合，可以通过 Cloudera Manager 部署安装 Spark，并有效管理监控 Spark 集群。

Spark 提供了一个快速的计算，写入，以及交互式查询的框架。相比于 Hadoop，Spark 拥有明显的性能优势。Spark 使用 in-memory 的计算方式，通过这种方式来避免一个 Mapreduce 工作流中的多个任务对同一个数据集进行计算时的 IO 瓶颈。Spark 利用 Scala 语言实现，Scala 能够使得处理分布式数据集时，能够像处理本地化数据一样。

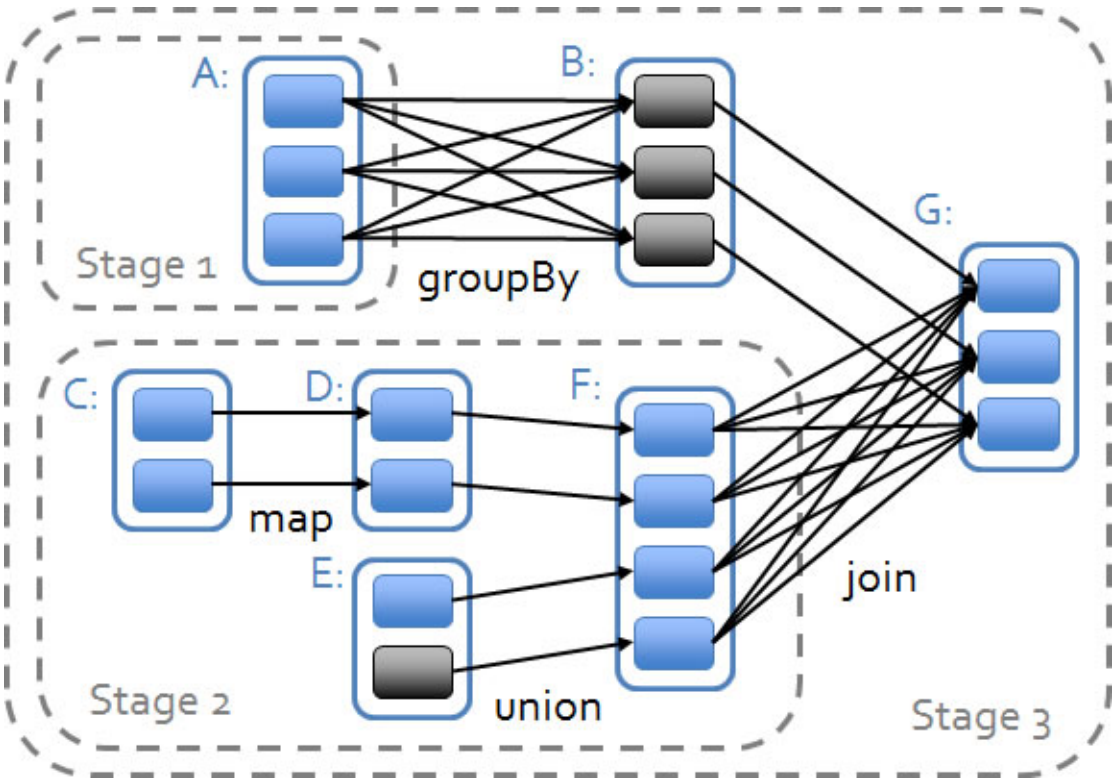
除了交互式的数据分析，Spark 还能够支持交互式的数据挖掘，由于 Spark 是基于内存的计算，很方便处理迭代计算，而数据挖掘的问题通常都是对同一份数据进行迭代计算。除此之外，Spark 能够运行于安装 Hadoop 2.0 Yarn 的集群。之所以 Spark 能够在保留 Mapreduce 容错性，数据本地化，可扩展性等特性的同时，能够保证性能的高效，并且避免繁忙的磁盘 IO，主要原因是因为 Spark 创建了一种叫做 RDD（Resilient Distributed Dataset）的内存抽象结构。

原有的分布式内存抽象，例如 key-value store 以及数据库，支持对于可变状态的细粒度更新，这一点要求集群需要对数据或者日志的更新进行备份来保障容错性。这样就会给数据密集型的工作流带来大量的 IO 开销。而对于 RDD 来说，它只有一套受限制的接口，仅仅支持粗粒度的更新，例如 map，join 等等。通过这种方式，Spark 只需要简单的记录建立数据的转换操作的日志，而不是完整的数据集，就能够提供容错性。这种数据的转换链记录就是数据集的溯源。由于并行程序，通常是对一个大数据集应用相同的计算过程，因此之前提到的粗粒度的更新限制并没有想象总的大。事实上，Spark 论文中天阐述了 RDD 完全可以作为多种不同计算框架，例如 Mapreduce，Pregel 等的编程模型。

并且，Spark 同时提供了操作允许用户显示的将数据转换过程持久化到硬盘。对于数据本地化，是通过允许用户能够基于每条记录的键值，控制数据分区实现的。（采用这种

方式的一个明显好处是，能够保证两份需要进行关联的数据将会被同样的方式进行哈希）。如果内存的使用超过了物理限制，Spark 将会把这些比较大的分区写入到硬盘，由此来保证可扩展性。

Spark 首先是一个批处理的引擎，下图给出了一个 Spark 批处理的例子，阐述了多个 RDD 以及操作如何被分组到不同的转换步骤。

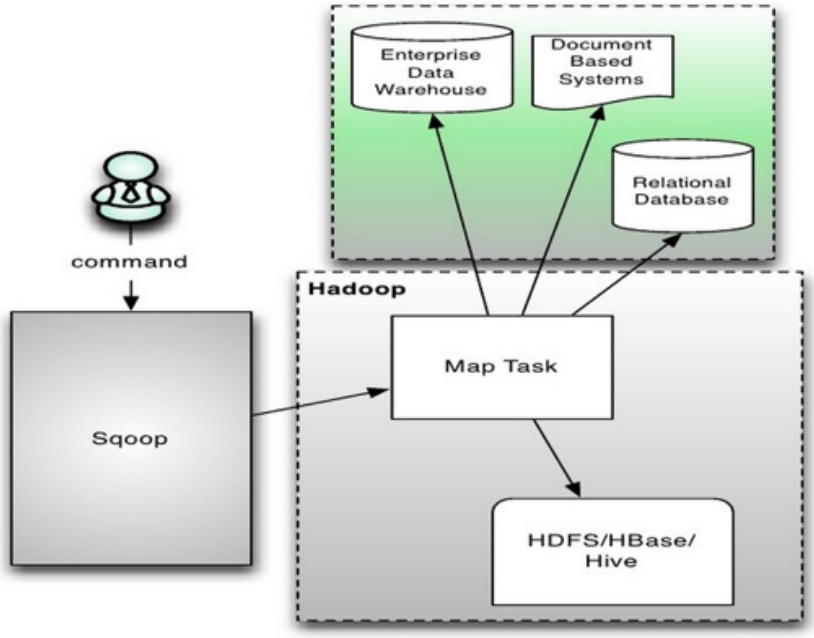


Spark 1.2 版本引入了动态资源扩展设置，通过 spark.dynamicAllocation.enabled 启用，可以根据工作负载动态调节应用程序的集群资源。YARN 可以通过分配 container 来运行 Spark 真正执行计算任务的 worker 进程， Spark 计算集群支持公平调度器（Fair Scheduler），通过向 YARN 申请和释放资源，实现内部计算资源（如 CPU 核心）的共享与隔离。

4.12. 数据库接入工具 Sqoop

Sqoop 是 Apache 顶级项目，主要用来在 Hadoop 和关系数据库中传递数据。通过 sqoop，可以方便的将数据从关系数据库导入到 HDFS，或者将数据从 HDFS 导出到关系

数据库。Sqoop 架构非常简单，其整合了 Hive、Hbase 和 Oozie，通过 MapReduce 任务来传输数据，从而提供并发特性和容错.其架构为：



对于某些 NoSQL 数据库它也提供了连接器。Sqoop，类似于其他 ETL 工具，使用元数据模型来判断数据类型并在数据从数据源转移到 Hadoop 时确保类型安全的数据处理。Sqoop 专为大数据批量传输设计，能够分割数据集并创建 Hadoop 任务来处理每个区块。Sqoop 有一个非常小的命令集，里面包括导入和导出，列出数据库和表信息，生成 Java 类来操纵数据，解析 SQL 命令以及其他一些更专门的命令。生成 Java 类的命令对于在 Hadoop 里编写 Java 应用来进行数据操作特别有用。SQL 解析命令可以显示执行 SQL 语句的结果，这在搜索新数据库或产生复杂逻辑的查询时非常有用。

使用 Sqoop 比自定义脚本有一定的优势。其一就是，Sqoop 被设计成具备容错性。你也可以自定义脚本来监控你的工作状态，然后从故障中恢复，但是那有可能难以置信的耗时。

Sqoop 中一大亮点就是可以通过 hadoop 的 mapreduce 把数据从关系型数据库中导入数据到 HDFS。

进行增量导入是与效率有关的最受关注的问题，因为 Sqoop 专门是为大数据集设计的。Sqoop 支持增量更新，将新记录添加到最近一次的导出的数据源上，或者指定上次修改的时间戳。

由于 Sqoop 将数据移入和移出关系型数据库的能力，其对于 Hive—Hadoop 生态系统里的著名的类 SQL 数据仓库—有专门的支持不足为奇。命令“create-hive-table”可以用来将数据表定义导入到 Hive。

Sqoop 可以在 HDFS/Hive/Hbase 和关系型数据库之间进行数据的导入导出，其中主要使用了 import 和 export 这两个工具。这两个工具非常强大，提供了很多选项帮助我们完成数据的迁移和同步。比如，下面两个潜在的需求：业务数据存放在关系数据库中。

1. 如果数据量达到一定规模后需要对其进行分析或同统计，单纯使用关系数据库可能会成为瓶颈，这时可以将数据从业务数据库数据导入（import）到 Hadoop 平台进行离线分析。

2. 对大规模的数据在 Hadoop 平台上进行分析以后，可能需要将结果同步到关系数据库中作为业务的辅助数据，这时候需要将 Hadoop 平台分析后的数据导出（export）到关系数据库。

Sqoop 完成上述基本应用场景用的是 import 和 export 工具。

其中，import 和 export 工具有些通用的选项，如：

指定 JDBC 连接字符串 --connection-manager <class-name> 指定要使用的连接管理器类 --driver <class-name> 指定要使用的 JDBC 驱动类

--hadoop-mapred-home <dir> 指定$HADOOP_MAPRED_HOME 路径 --help 打印用法帮助信息

--password-file 设置用于存放认证的密码信息文件的路径 -P

从控制台读取输入的密码 --password <password> 设置认证密码 --username <username> 设置认证用户名 --verbose

打印详细的运行信息

--connection-param-file <filename>

可选，指定存储数据库连接参数的属性文件

import 工具，是将 HDFS 平台外部的结构化存储系统中的数据导入到 Hadoop 平台，便于后续分析其基本选项及其含义，如下所示：

--append

将数据追加到 HDFS 上一个已存在的数据集上 --as-avrodatafile 将数据导入到 Avro 数据文件 --as-sequencefile 将数据导入到 SequenceFile

--as-textfile

将数据导入到普通文本文件（默认） --boundary-query <statement> 边界查询，用于创建分片（InputSplit） --columns <col,col,col…> 从表中导出指定的一组列的数据 --delete-target-dir 如果指定目录存在，则先删除掉 --direct

使用直接导入模式（优化导入速度）

--direct-split-size <n> 分割输入 stream 的字节大小（在直接导入模式下） --fetch-size <n> 从数据库中批量读取记录数 --inline-lob-limit <n> 设置内联的 LOB 对象的大小 -m,--num-mappers <n> 使用 n 个 map 任务并行导入数据 -e,--query <statement> 导入的查询语句

--split-by <column-name> 指定按照哪个列去分割数据 --table <table-name> 导入的源表表名 --target-dir <dir> 导入 HDFS 的目标路径 --warehouse-dir <dir> HDFS 存放表的根路径 --where <where clause> 指定导出时所使用的查询条件 -z,--compress

启用压缩

--compression-codec <c> 指定 Hadoop 的 codec 方式（默认 gzip）

--null-string <null-string> 如果指定列为字符串类型，使用指定字符串替换值为 null 的该类列的值

--null-non-string <null-string>

如果指定列为非字符串类型，使用指定字符串替换值为 null 的该类列的值

Hive 参数

--hive-home <dir> Override $HIVE_HOME

--hive-import 插入数据到 hive 当中，使用 hive 的默认分隔符 --hive-overwrite 覆盖 hive 表中的数据

--create-hive-table

建表，如果表已经存在，该操作会报错

--hive-table <table-name> 设置到 hive 当中的表名

--hive-drop-import-delims 导入到 hive 时删除 \n, \r, and \01

--hive-delims-replacement 导入到 hive 时用自定义的字符替换掉\n, \r, and \01 --hive-partition-key hive 分区的 key --hive-partition-value <v> hive 分区的值

--map-column-hive <map>

类型匹配，sql 类型对应到 hive 类型

HBase 参数

--column-family <family> 把内容导入到 hbase 当中，默认是用主键作为 split 列 --hbase-create-table 创建 Hbase 表

--hbase-row-key <col> 指定字段作为 row key ，如果输入表包含复合主键，用逗号分隔 --hbase-table <table-name>

指定 hbase 表

export 工具，是将 HDFS 平台的数据，导出到外部的结构化存储系统中，可能会为一些应用系统提供数据支持。我们看一下 export 工具的基本选项及其含义，如下所示：

--validate <class-name>

启用数据副本验证功能，仅支持单表拷贝，可以指定验证使用的实现类

--validation-threshold <class-name> 指定验证门限所使用的类 --direct

使用直接导出模式（优化速度） --export-dir <dir> 导出过程中 HDFS 源路径 -m,--num-mappers <n> 使用 n 个 map 任务并行导出 --table <table-name> 导出的目的表名称

--call <stored-proc-name> 导出数据调用的指定存储过程名

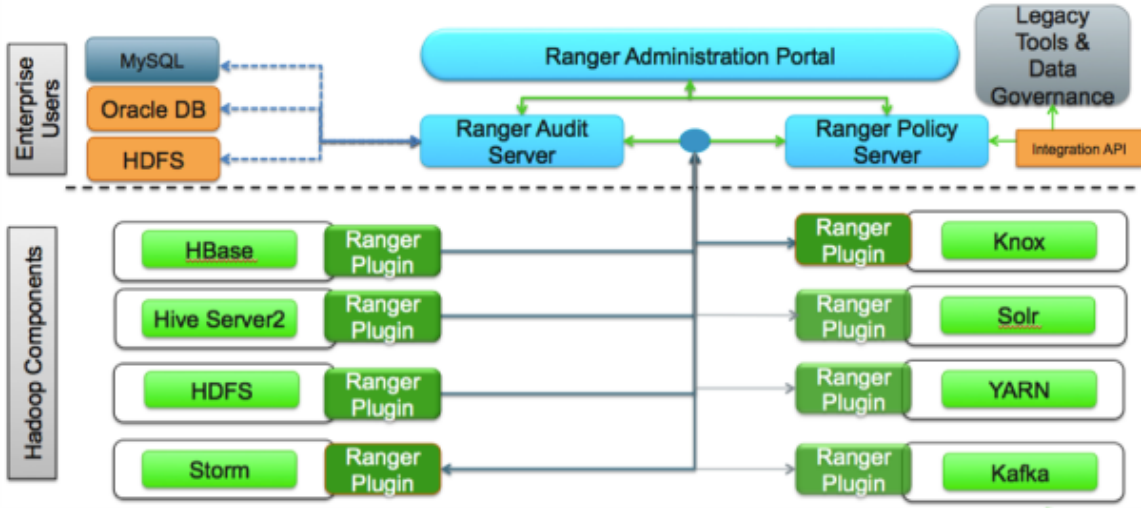
--update-key <col-name> 更新参考的列名称，多个列名使用逗号分隔

--update-mode <mode>

指定更新策略，包括：updateonly（默认）、allowinsert --input-null-string <null-string> 使用指定字符串，替换字符串类型值为 null 的列 --input-null-non-string <null-string> 使用指定字符串，替换非字符串类型值为 null 的列 --staging-table <staging-table-name> 在数据导出到数据库之前，数据临时存放的表名称 --clear-staging-table 清除工作区中临时存放的数据 --batch 使用批量模式导出

4.13. Cloudera 一站式安全管理

CDP 在 Apache Ranger 基础上提供集中式安全管理。用户可通过 Ranger Administration 界面完成安全管理的一些列操作。例如创建和更新权限和访问策略，这些策略实际存储于 Ranger 策略数据库中 。为了将策略生效，需要同步至 HDP 的组件当中。为此 CDP 组件嵌入了 Ranger plug-ins (轻量 Java 程序)用来与策略数据库同步。



Ranger plug-ins 从 Ranger Policy Server 下拉策略更新并保存与本地文件中。当该服务组件收到用户访问请求时，Ranger plug-ins 会首先截获该请求并根据本地保存的策略对用户请求进行权限校验，同时也会收集 audit 信息上报给 Ranger Audi Server。

- 管理

管理员需要一个统一界面能够对策略集中管理。管理员可以定义和管理安全策略，使得这些策略在 Hadoop 各组件中保持语义一致。

除了策略管理外，Apache Ranger 管理台为管理员提供了更多操作，囊括所有上述安全特性。

- 认证

用户身份认证是 CDP 集群安全访问的基础。用户需要能够可靠地“识别”自己，然后在整个集群中传播该身份。一旦完成这一认证，用户可以访问资源（如文件或目录）或与群集交互（如运行 MapReduce 作业）。除了用户之外，Hadoop 集群资源本身（如主机和服务）也需要相互验证，以避免潜在的恶意系统或守护进程“冒充”集群的可信组件来访问数据。

CDP 大数据平台支持 2 种方式的 Kerberos 认证

MIT Kerberos（或 Linux 集成化服务 FreeIPA）

Microsoft Active Directory

Kerberos 是第三方认证机制，用户和服务依靠第三方（Kerberos 服务器）来相互认证。Kerberos 服务器本身被称为密钥分发中心或 KDC。在抽象架构上，Kerberos 有三个部分：

用户和服务的数据库（称为“**principals**”）以及它们各自的 Kerberos 认证密码

执行初始认证的认证服务器（**Authentication Server 或简称 AS**）颁发票证授予票证（Ticket Granting Ticket，TGT）

一个票据授权服务器（**Ticket Granting Server 或简称 TGS**）发出基于初始后续服务票证 TGT

一个用户主要来自 AS 请求认证。AS 返回使用用户主体的 Kerberos 密码加密的 TGT，该密码仅由用户主体和 AS 知道。用户主体使用其 Kerberos 密码在本地解密 TGT，并且从这一点开始，直到票证到期，用户主体可以使用 TGT 从 TGS 获得服务票据。服务票是什么让一个校长访问各种服务。

由于群集资源（主机或服务）每次都无法提供密码来解密 TGT，因此它们使用一个称为 keytab 的特殊文件，其中包含资源主体的身份验证凭证。Kerberos 服务器控制的主机，用户和服务集称为领域。

术语

术语	描述
Key Distribution Center，或 KDC	在启用了 Kerberos 的环境中进行身份验证的可信来源。
Kerberos KDC 服务器	作为密钥分发中心（KDC）的计算机或服务器。
Kerberos 客户端	群集中任何对 KDC 进行身份验证的机器。
Principal （KDC 用户名）	对 KDC 进行身份验证的用户或服务的唯一名称。
Keytab 密钥表	包含一个或多个主体及其关键字的文件。
Realm（域）	包含 KDC 和多个客户端的 Kerberos 网络。
KDC 管理员帐户	Ambari 用于创建主体并在 KDC 中生成密钥表的管理帐户。

Cloudera Manager 用于创建主体并在 KDC 中生成密钥表的管理帐户。

Kerberos Principal

Hadoop 中的每个服务和子服务都必须有自己的主体。 给定域中的主体名称由主名称和实例名称组成，在这种情况下，实例名称是运行该服务的主机的 FQDN。 由于服务不会使用密码登录以获取其票据，因此其主体的身份验证凭据将存储在一个 keytab 文件中，该文件是从 Kerberos 数据库中提取的，并存储在具有服务主体的服务主体的安全目录中。

Principal 和 Keytab 命名约定

Asset	Convention	Example
Principals	$service_component_name/$FQDN@EXAMPLE.COM	nn/c6401.ambari.apache.org@EXAMPLE.COM
Keytabs	$service_component_abbreviation.service.keytab	/etc/security/keytabs/nn.service.keytab

在前面的例子中注意每个服务主体的主要名字。 这些主要名称（例如 nn 或 hive）分别代表 NameNode 或 Hive 服务。 每个主名称都附加了实例名称，它所运行的主机的 FQDN。 此约定为在多个主机（如 DataNode 和 NodeManagers）上运行的服务提供唯一主体名称。 添加主机名用于区分来自 DataNode B 的请求，例如来自 DataNode A 的请求。由于以下原因，这是重要的：

一个 DataNode 的损坏的 Kerberos 凭据不会自动导致所有 DataNode 的受损的 Kerberos 凭据。

如果多个 DataNode 具有完全相同的主体，并且同时连接到 NameNode，并且发送的 Kerberos 身份验证器碰巧具有相同的时间戳，则身份验证将被拒绝作为重播请求。

- 授权

Ranger 可对 CDP 服务组件进行细粒度的资源划分，并对资源进行权限管理：

HDFS: path

HBase: table, column family, column

Hive: database, table, column

Impala: database, table, column

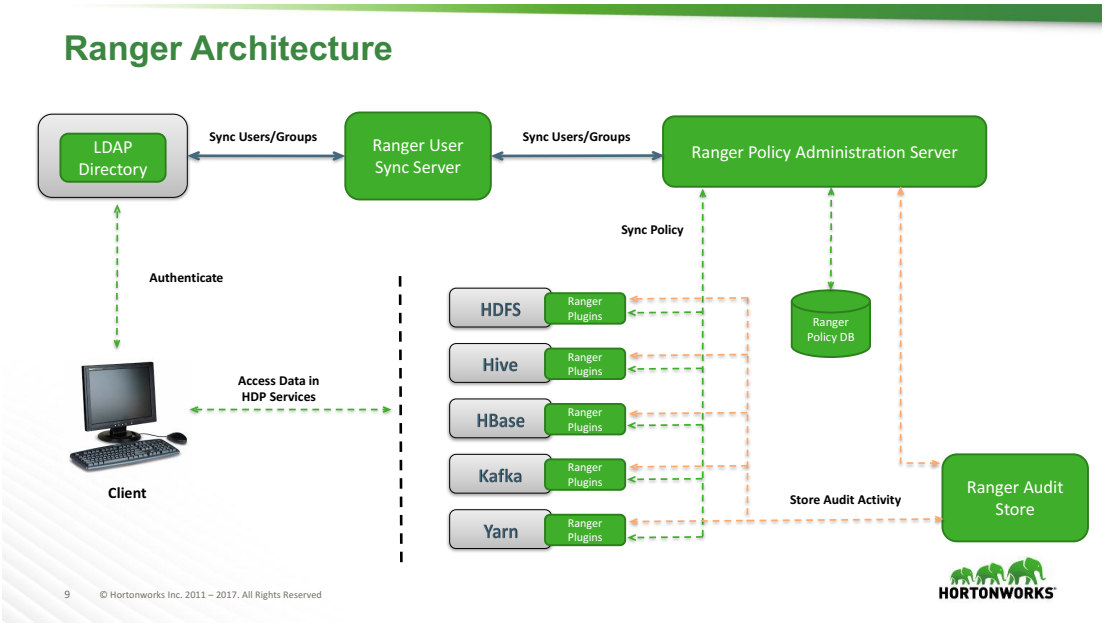
Kafka: topic

Knox: Knox topology

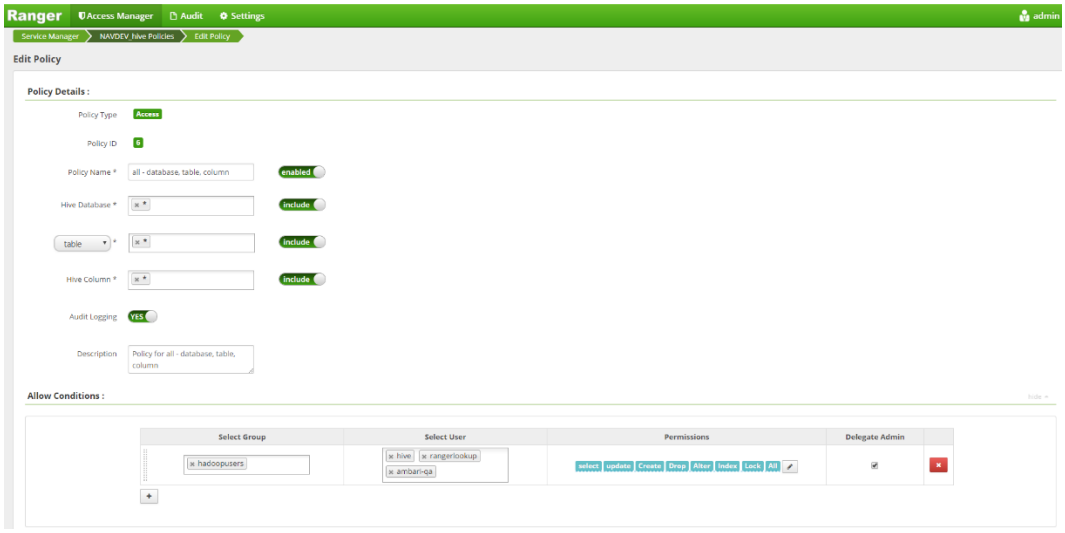
Yarn: queue

Solor: collection

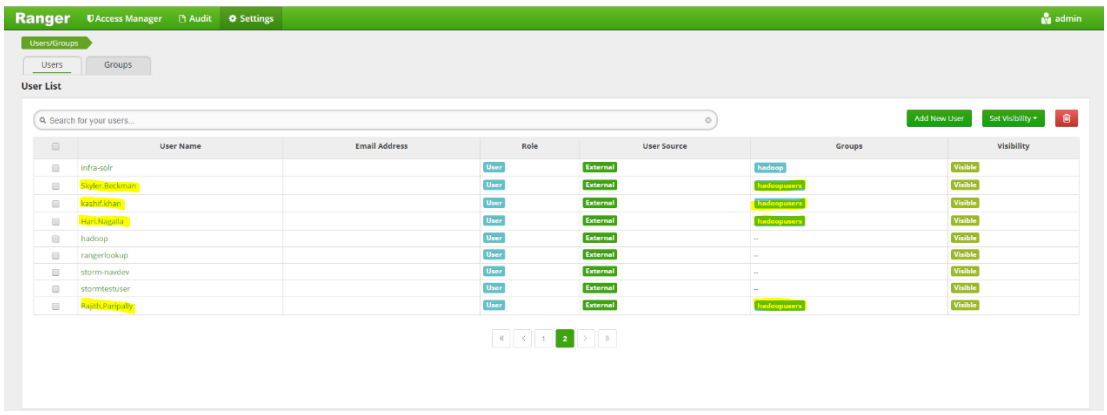
Ranger 供安全管理员对上述资源指定允许访问的 LDAP 组或者用户，并分配不同的访问权限从而生成管理策略，保存于 Ranger 数据库中。为了将策略生效，需要同步至 CDP 的组件当中。为此 CDP 组件嵌入了 Ranger plug-ins (轻量 Java 程序)用来与策略数据库同步。



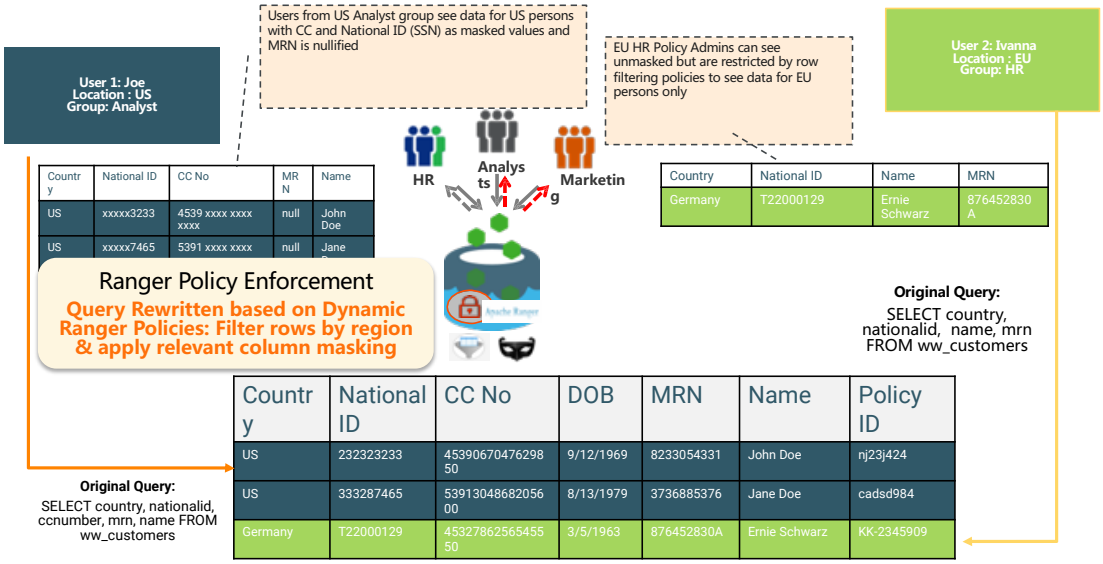
下图是 Ranger 定义 Hive 资源（所有 databse 的所有表以及所有列）访问策略的用户界面。管理员可指定该策略适用的组（hadoopusers）以及用户（hive, rangerlookup 和 ambari-qa）。并对其指定访问权限（select, update, create, drop, alter, index, lock 和 all）。



用户和组既可以是 Linux 系统中的 POSIX 账户或者是 LDAP/AD 保存的账户. FreeIPA 已通过 SSSD 服务将 Linux 系统账户同步到 LDAP 并进行管理，此时的组和用户信息与 Linux 系统中的 POSIX 账户信息保持一致。

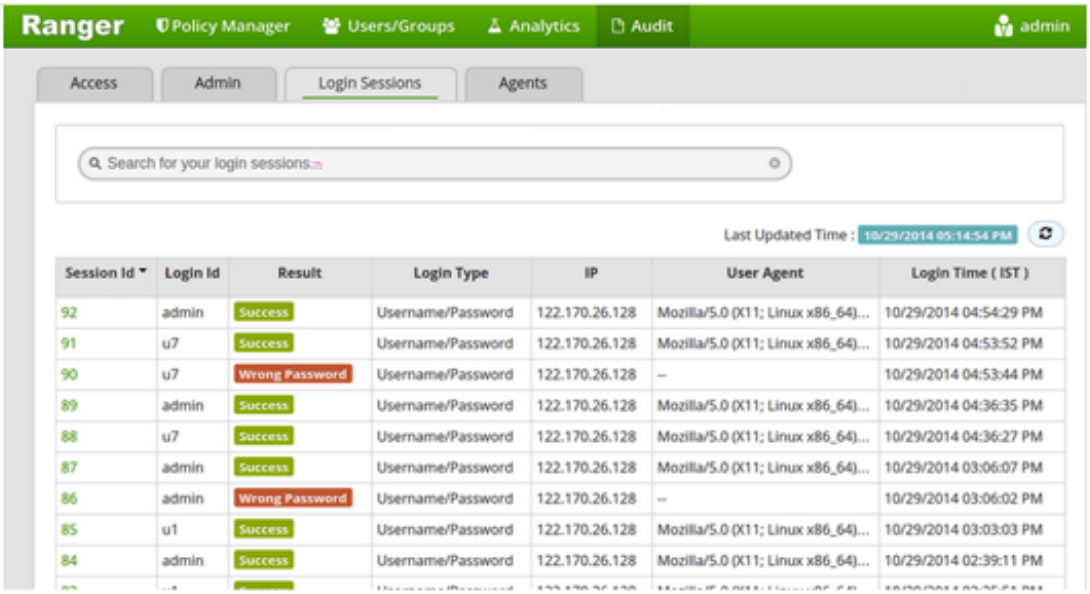


除了基于角色的访问策略，Ranger 还支持 ABAC（Attribute Base Access Control）基于属性的权限控制。不同于常见的将用户通过某种方式关联到权限的方式，ABAC 则是通过动态计算一个或一组属性来是否满足某种条件来进行授权判断（可以编写简单的逻辑）。属性通常来说分为四类：用户属性（如用户年龄），环境属性（如当前时间），操作属性（如读取）和对象属性（如一篇文章，又称资源属性），所以理论上能够实现非常灵活的权限控制，几乎能满足所有类型的需求。采用 ABAC，Ranger 支持动态行过滤和列脱敏，如下图所示：



- 审计

Apache Ranger 收集各个组件的审计日志，提供统一界面集成日志搜索服务。



Ranger 允许审计日志存储在 Solr 或 HDFS 当中。 推荐同时使用 Solr 和 HDFS 存储审计日志。Solr 主要用于日志搜索，HDFS 可满足审计日志的长期保存，并且可导出至任何 security information and event management (SIEM)审计系统。

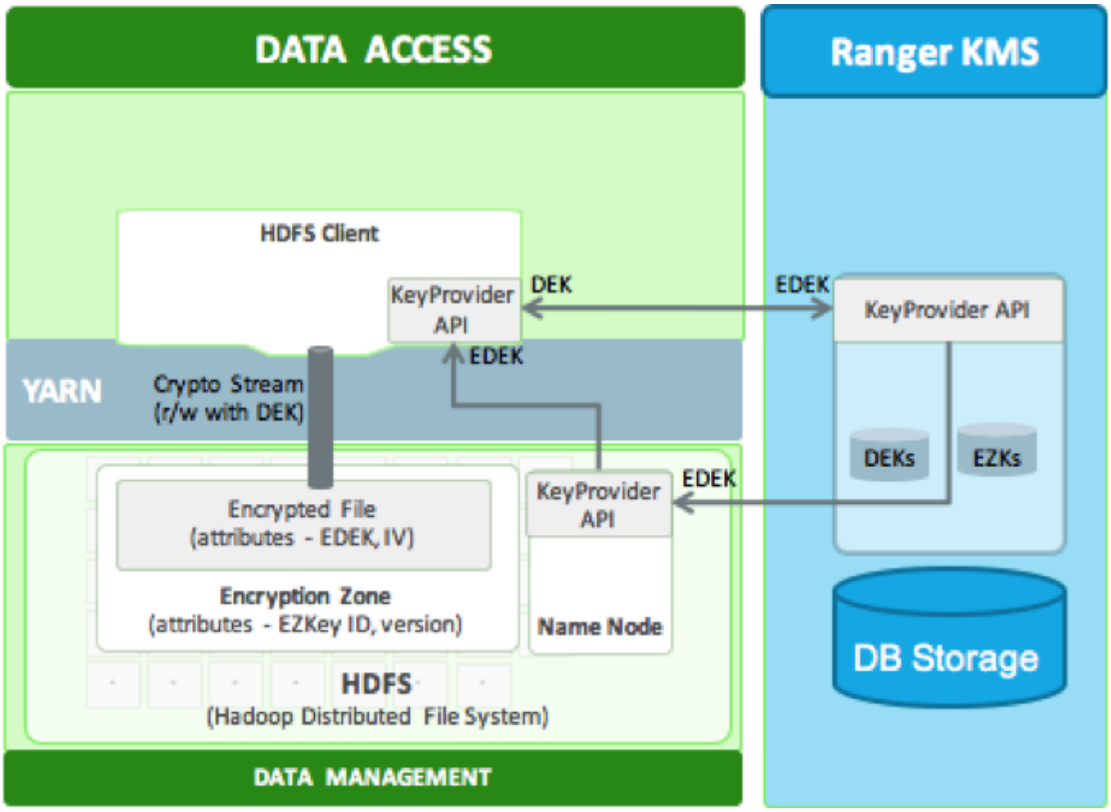
- 加密

Hadoop Key Management Server（KMS）是一个基于 HadoopKeyProvider API 编写的密钥管理服务器，基于 C/S 模式实现。 Client 实际上是 KeyProvider 的实现，使用 KMS HTTP REST API 与 KMS Server 交互。KMS 和它的 Client 有内置的安全机制，支持 HTTP SPNEGO Kerberos 认证和 HTTPS 安全传输。HDFS 实现透明，端到端加密。配置完成后，用户在 HDFS 存储数据的时候，无需做任何程序代码的更改， 数据加密和解密由客户端完成。HDFS 不会存储或访问未加密的数据或数据加密密钥。如下对透明加密的原理做出详细解释：

加密区

为了使加密过程对用户透明，KMS 对 HDFS 引入了 Encryption Zone（EZ）。EZ 可看作是 HDFS 一条目录路径，目录中所有的文件都被加密。当客户端读写加密文件时，是不会感知加解密过程的。每个 EZ 都伴有唯一的 EZ key，每个在 EZ 中的文件都有唯一的 EDEK，EZ key 和 EDEK 在下节做详细解释。

用户对加密数据的访问权限由 HDFS 来管理，对 EZ key 的访问权限由 KMS 管理。即使入侵者获取了文件的访问权限，得到只不过是加密的内容，还是无法获取文件明文。



KMS 加解密过程解释

1. 创建 EncrypCon Zone

2. KMS 管理员创建 EncrypCon Zone (EZ)

3. HDFS 管理员用 EZKey 创建 EncrypCon Zone

HDFS 客户端在 EZ 内 创建文件并赋予相应权限

Name Node 使用 EZ Key ID 及其 version 从 KMS 请求 EDEK (encrypted data encrypCon key)

Name Node 返回 EDEK 给 HDFS 客户端

3. HDFS 客户端 请求 KMS 解密 EDEK

4. KMS 收到请求后，若用户有权访问 EZ Key，则返回给客户端 DEK

5. HDFS 客户端使用 DEK 和 Hadoop Cryptographic File System (CryptoOutputStream) 向 HDFS EZ 写入加密文件。EDEK 作为文件的属性随文件保存下来 （见图中 Encrypted File 示意框）

6. HDFS 客户端从 EZ 读取文件，需检查相应文件权限以及 KMS 权限

Name Node 将 EDEK 请求发送给 KMS

KMS 提供 HDFS 客户端相应的 DEK

HDFS 客户端使用 DEK 以及 HDFS Cryptographic File System (CryptoInputStream) 将文件解密并读取明文内容。

4.14. 分布式消息队列 Kafka

Kafka 起源自 Linkedin 用于日志处理的分布式消息队列。Linkedin 的日志数据容量大，但对可靠性要求不高，其日志数据主要包括用户行为（登录、浏览、点击、分享、喜欢）以及系统运行日志（CPU、内存、磁盘、网络、系统及进程状态）。当前很多的消息队列服务提供可靠交付保证，并默认是即时消费（不适合离线）。高可靠交付对 linkedin 的日志不是必须的，故可通过降低可靠性来提高性能，同时通过构建分布式的集群，允许消息在系统中累积，使得 kafka 同时支持离线和在线日志处理。

Kafka 之所以和其它绝大多数信息系统不同，是因为下面这几个为数不多的比较重要的设计决策：

1. Kafka 在设计之时为就将持久化消息作为通常的使用情况进行了考虑。
2. 主要的设计约束是吞吐量而不是功能。
3. 有关*哪些数据*已经被使用了的状态信息保存为数据使用者（consumer）的一部分，而不是保存在服务器之上。
4. Kafka 是一种显式的分布式系统。它假设，数据生产者（producer）、代理（brokers）和数据使用者（consumer）分散于多台机器之上。

与之对应，Kafka 做了如下实现**策略：**

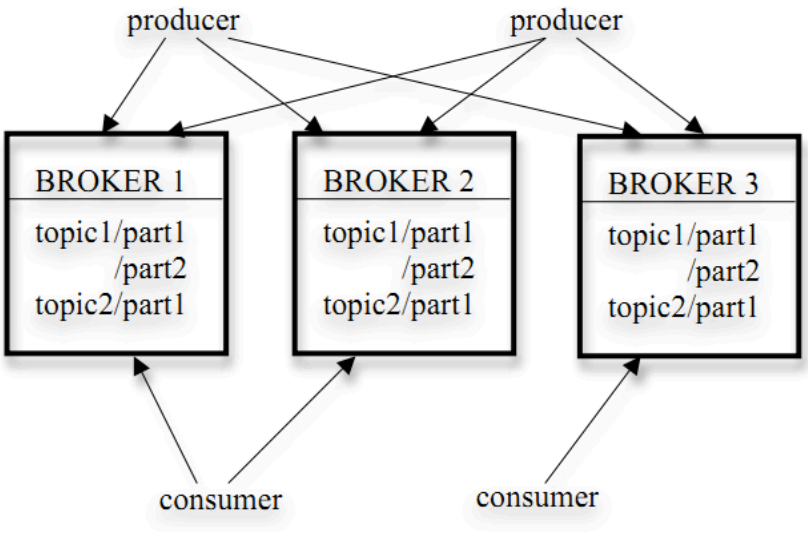
1. kafka 以 topic 来进行消息管理，每个 topic 包含多个 part（ition），每个 part 对应一个逻辑 log，有多个 segment 组成。

2. 每个 segment 中存储多条消息，消息 id 由其逻辑位置决定，即从消息 id 可直接定位到消息的存储位置，避免 id 到位置的额外映射。

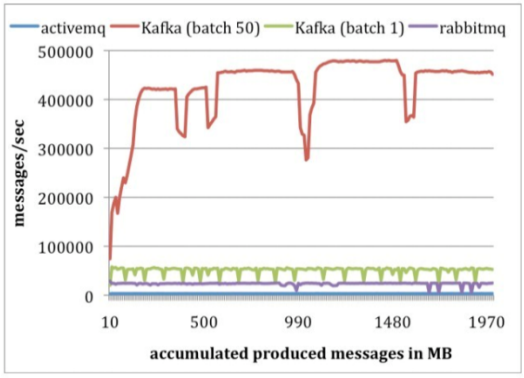
3. 每个 part 在内存中对应一个 index，记录每个 segment 中的第一条消息偏移。

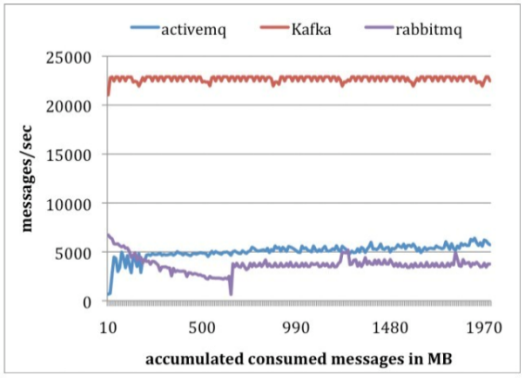
4. 发布者发到某个 topic 的消息会被均匀的分布到多个 part 上（随机或根据用户指定的回调函数进行分布），broker 收到发布消息往对应 part 的最后一个 segment 上添加该消息，当某个 segment 上的消息条数达到配置值或消息发布时间超过阈值时，segment 上的消息会被 flush 到磁盘，只有 flush 到磁盘上的消息订阅者才能订阅到，segment 达到一定的大小后将不会再往该 segment 写数据，broker 会创建新的 segment。

Kafka 是一个显式的分布式系统——生产者、使用者和代理都可以运行在作为一个逻辑单位的、进行相互协作的集群中不同的机器上。对于代理和生产者，这么做非常自然，但使用者却需要一些特殊的支持。每个使用者进程都属于一个*使用者小组（consumer group）*。准确地讲，每条消息都只会发送给每个使用者小组中的一个进程。因此，使用者小组使得许多进程或多台机器在逻辑上作为一个单个的使用者出现。使用者小组这个概念非常强大，可以用来支持 JMS 中*队列（queue）*或者*话题（topic）*这两种语义。为了支持*队列*语义，我们可以将所有的使用者组成一个单个的使用者小组，在这种情况下，每条消息都会发送给一个单个的使用者。为了支持*话题*语义，可以将每个使用者分到它自己的使用者小组中，随后所有的使用者将接收到每一条消息。在我们的使用当中，一种更常见的情况是，我们按照逻辑划分出多个使用者小组，每个小组都是有作为一个逻辑整体的多台使用者计算机组成的集群。在大数据的情况下，Kafka 有个额外的优点，对于一个话题而言，无论有多少使用者订阅了它，一条条消息都只会存储一次。



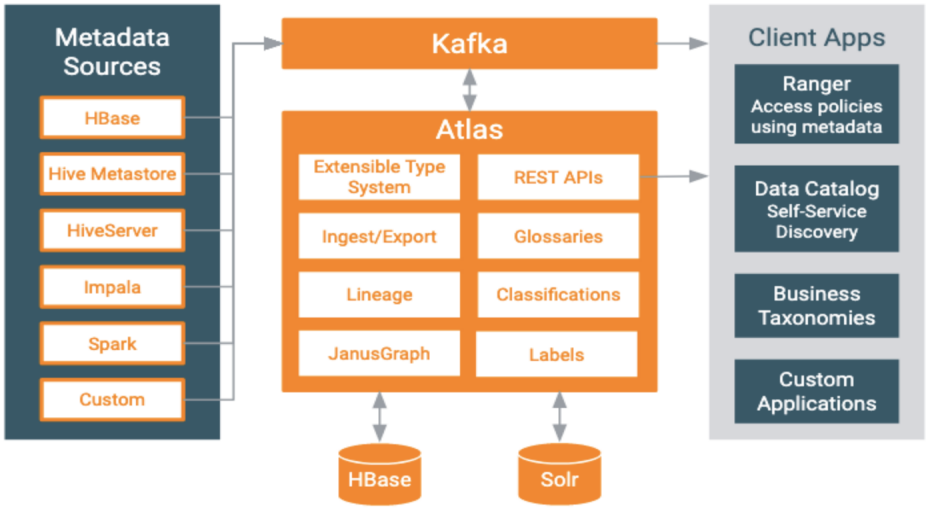
基于 Kafka 的测试报告显示其性能完胜其他的 message queue，单条消息发送（每条 200bytes）,能到 50000messages/sec，50 条 batch 方式发送，平均为 400000messages/sec。结果如下图：





4.15. Apache Atlas

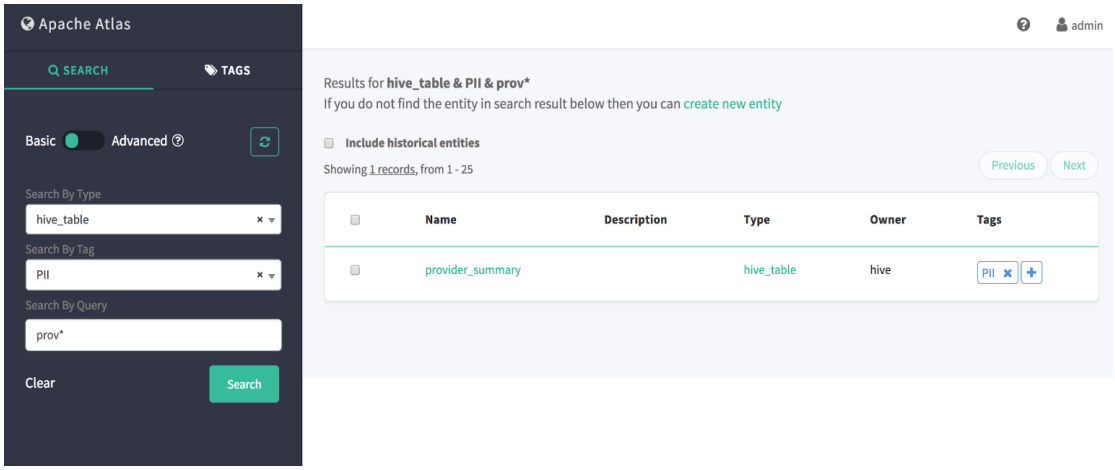
Apache Atlas 是 Hadoop 社区为解决 Hadoop 生态系统的元数据治理问题而产生的开源项目，它为 Hadoop 集群提供了包括 数据分类、集中策略引擎、数据血缘、安全和生命周期管理在内的元数据治理核心能力。



Atlas 的整体架构如上图所示，主要提供了以下的核心功能：

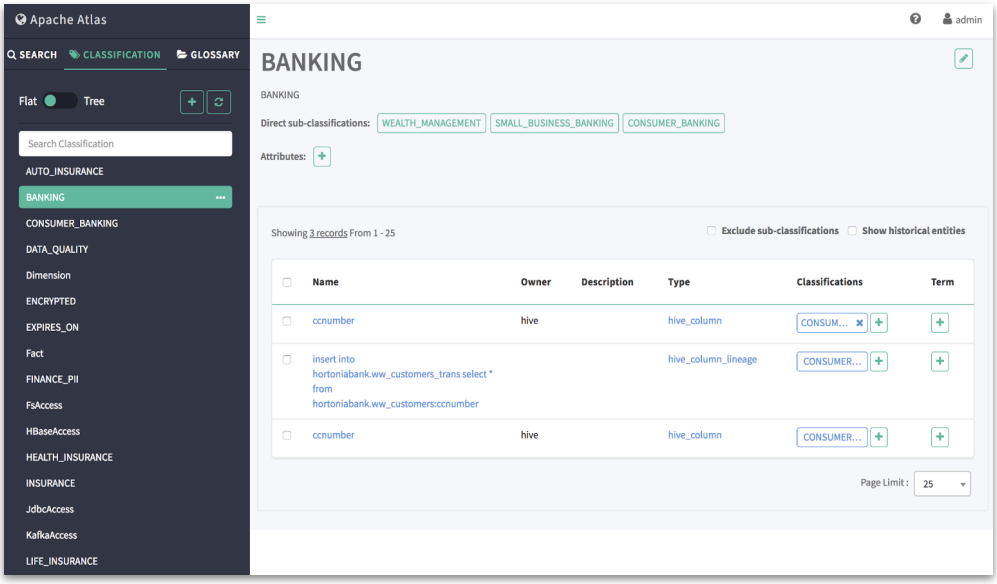
- 元数据收集和搜索

  支持多种数据源的元数据收集，包括 HDFS，Hive，Spark，Impala，HBase， Nifi，Kafka 以及其他客户化的数据源；支持分类搜索和查询搜索



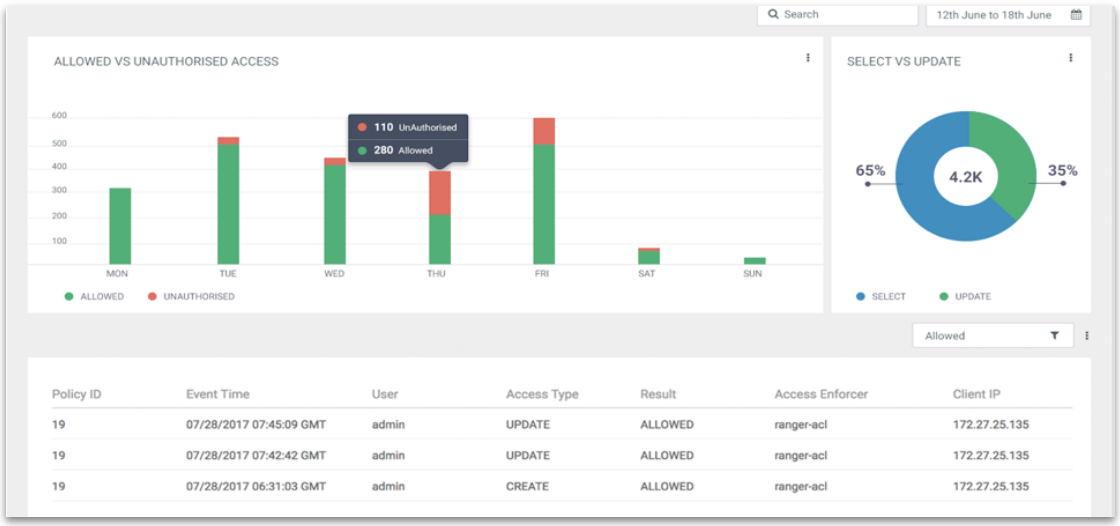
- 数据分类

  为元数据导入或定义业务导向的分类注释；定义，注释，以及自动捕获数据集和底层元素之间的关系；导出元数据到第三方系统



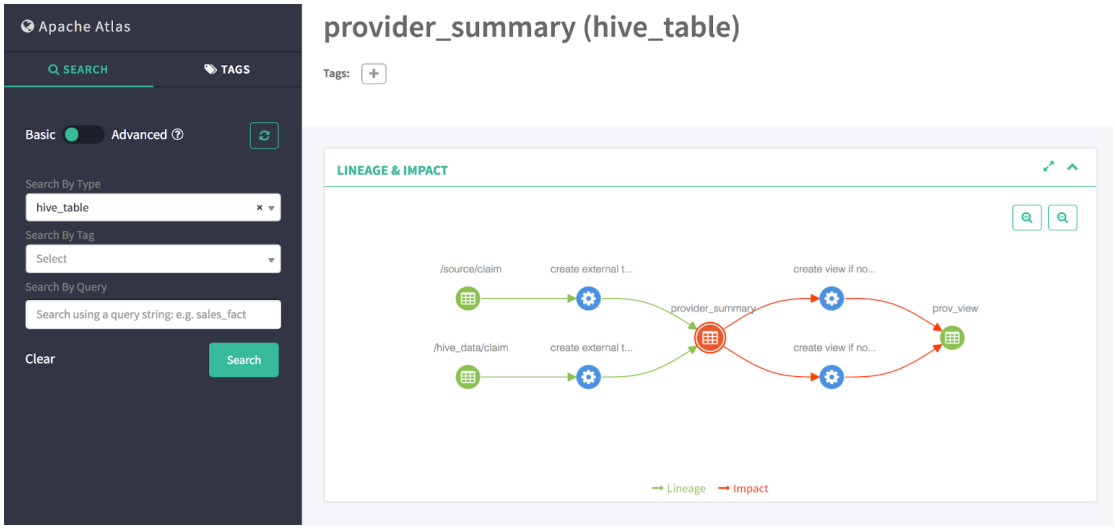
■ 集中审计

维护完整的数据访问、数据操作审计历史以满足合规需求。同时帮助系统管理员快速验证用户/用户组在 Hadoop 集群中数据集合的访问权限（permission）正确性。



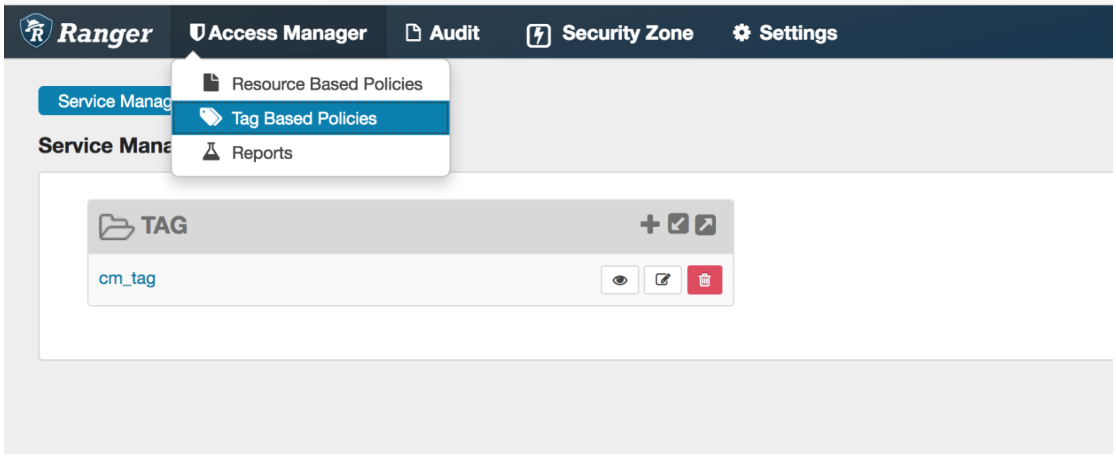
■ 血缘关系

提供追踪数据在系统中演变过程的功能，允许用户回朔数据源头，验证数据有效性，进而提供完善的数据生命周期管理（lifecycle management）。



- 安全和策略引擎

  Atlas 通过与 Ranger 集成，为 Hadoop 提供了基于标签(Tag)的动态访问权限控制，通过控制与资源关联的标签而非资源本身可以为权限控制模型提供诸多便利。



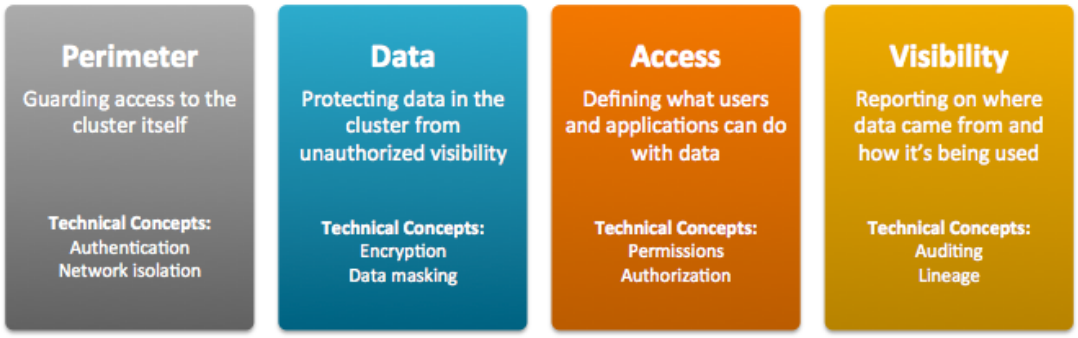
5.Cloudera 安全概述

5.1. 概述

作为旨在支持大量和类型的数据的系统，Cloudera 集群必须满足监管机构、政府、行业和公众提出的不断发展的安全要求。Cloudera 集群包含 Hadoop 核心和生态系统组件，必须保护所有这些组件免受各种威胁，以确保所有集群服务和数据的机密性、完整性和可用性。

5.1.1. 安全要求

数据管理系统的目标（例如机密性，完整性和可用性）要求在多个维度上对系统进行保护。可以根据总体操作目标和技术概念来表征这些特征，如下图所示：



- **外围**访问集群必须受到保护，以防止来自内部和外部网络以及各种角色的各种威胁。例如，可以通过正确配置防火墙，路由器，子网以及正确使用公用和专用 IP 地址来提供网络隔离。身份验证机制可确保人员，流程和应用程序正确获得集群的身份，并在获得对集群的访问权限之前证明自己的身份。
- **数据**必须始终保护集群中的数据免遭未经授权的暴露。同样，必须保护集群中节点之间的通信。加密机制可确保即使不良行为者拦截了网络数据包或从系统上物理删除了硬盘驱动器，其内容也不可用。
- **访问**权限必须明确授予对集群中任何特定服务或数据项的**访问**权限。授权机制可确保用户对集群进行身份验证后，他们只能看到数据并使用已被授予特定权限的进程。
- **可见**性。**可见**性意味着数据更改的历史是透明的，并且能够满足数据治理策略。审核机制可确保对数据及其沿袭的所有操作（源，随时间的变化等）在发生时均记录在案。

确保集群安全以实现特定的组织目标涉及使用 Hadoop 生态系统固有的安全功能以及使用外部安全基础架构。各种安全机制可以在一定范围内应用。

5.1.2. 安全等级

下图显示了可以为 Cloudera 集群实现的安全级别范围，从非安全（0）到最安全（3）。随着集群上数据的敏感度和数据量的增加，为集群选择的安全级别也应增加。

有了 3 级安全性，您的 Cloudera 集群就可以完全符合各种行业和法规要求，并可以在必要时进行审核。下表更详细地描述了这些级别：

级别	安全	特点
0	不安全	未配置安全性。非安全集群绝对不能在生产环境中使用，因为它们容易受到任何和所有攻击和利用。
1	最小	配置用于身份验证，授权和审核。首先配置身份验证，以确保用户和服务仅在证明其身份后才能访问集群。接下来，应用授权机制为用户和用户组分配特权。审核过程跟踪谁访问集群（以及如何访问）。
2	更多	敏感数据已加密。密钥管理系统处理加密密钥。已经为元存储中的数据设置了审核。定期检查和更新系统元数据。理想情况下，已经设置了集群，以便可以跟踪任何数据对象的沿袭（数据管理）。
3	最安全	安全企业数据中心（EDH）是其中所有数据（包括静态数据和传输中数据）都经过加密并且密钥管理系统具有容错能力的企业。审核机制符合行业、政府和法规标准（例如 PCI、HIPAA 和 NIST），并从 EDH 扩展到与其集成的其他系统。集群管理员训练有素，安全程序已通过专家认证，并且集群可以通过技术审查。

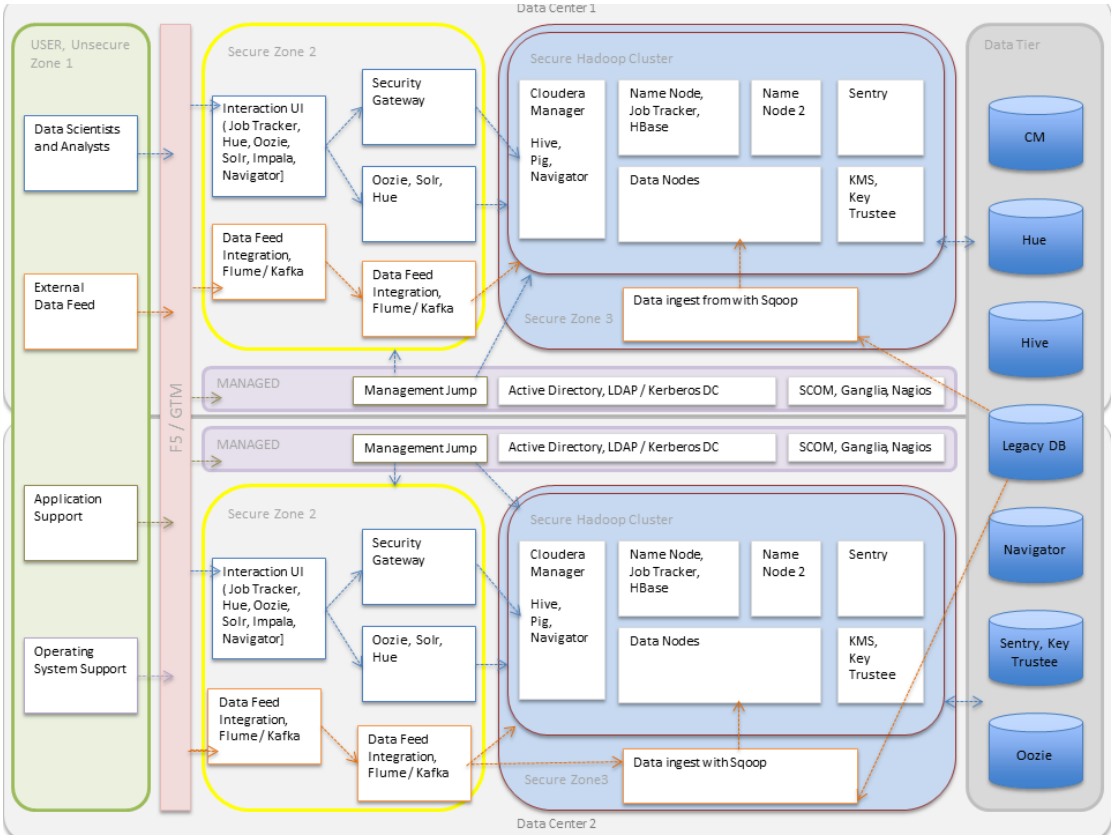
5.1.3. Hadoop 安全架构

下图是生产型 Cloudera 企业集群中许多工作组件中某些组件的示例。该图突出显示了需要保护可能从内部和外部数据馈送以及可能跨多个数据中心摄取数据的集群的安全性的需求。要确保集群安全，就需要在所有许多内部和内部连接中以及要查询，运行作业甚至查看集群中保存的数据的所有用户中应用身份验证和访问控制。

- 外部数据流通过适用于 NiFi/Flume 和 Kafka 的机制进行身份验证。使用 Sqoop 提取旧数据库中的数据。数据科学家和 BI 分析师可以使用诸如 Hue 之类的界面来处

理 Impala 或 Hive 上的数据，以创建和提交作业。可以利用 Kerberos 身份验证来保护所有这些交互。

- 可以使用透明的 HDFS 加密和企业级的密钥受托者服务器将加密应用于静态数据。
- 可以使用 Ranger（用于 Hive/Impala 和 Search 等服务）以及 HDFS 访问控制列表来实施授权策略。
- 可以使用 Apache Ranger 提供审核功能。



5.2. 认证概述

Cloudera Manager 身份验证概述。

身份验证是任何计算环境的基本安全要求。简单来说，用户和服务必须先向系统证明其身份（身份验证），然后才能在授权范围内使用系统功能。身份验证和授权携手并进，以保护系统资源。授权使用多种方式处理，从访问控制列表（ACL）到 HDFS 扩展 ACL，再到使用 Ranger 的基于角色的访问控制（RBAC）。

几种不同的机制一起工作以对集群中的用户和服务进行身份验证。这些取决于集群上配置的服务。大多数 CDH 组件，包括 Apache Hive，Hue 和 Apache Impala，都可以使

用 Kerberos 进行身份验证。可以将 MIT 和 Microsoft Active Directory Kerberos 实现集成在一起，以与 Cloudera 集群一起使用。

另外，可以在 LDAP 兼容的身份服务（例如 Windows Server 的核心组件 OpenLDAP 和 Microsoft Active Directory）中存储和管理 Kerberos 凭据。

本节提供简要概述，并特别关注使用 Microsoft Active Directory 进行 Kerberos 身份验证或将 MIT Kerberos 和 Microsoft Active Directory 集成时可用的不同部署模型。

Cloudera 不提供 Kerberos 实现。可以将 Cloudera 集群配置为使用 Kerberos 进行身份验证，即 MIT Kerberos 或 Microsoft Server Active Directory Kerberos，特别是密钥分发中心或 KDC。必须先设置 Kerberos 实例并使其运行，然后才能配置集群以使用它。

收集有关 KDC 的所有配置详细信息（或在设置过程中让 Kerberos 管理员可以帮助您）是与集群和 Kerberos 集成无关的一项重要的初步任务，而与部署模型无关。

5.2.1. Kerberos 概述

简而言之，Kerberos 是一种身份验证协议，它依赖于加密机制来处理发出请求的客户端与服务器之间的交互，从而极大地降低了模拟的风险。密码既不存储在本地也不通过网络明文发送。用户在登录其系统时输入的密码用于解锁本地机制，然后在与受信任的第三方的后续交互中使用该机制来向用户授予票证（有效期有限），该票证用于根据请求进行身份验证服务。在客户端和服务器进程相互证明各自的身份之后，对通信进行加密以确保隐私和数据完整性。

受信任的第三方是 Kerberos 密钥分发中心（KDC），它是 Kerberos 操作的焦点，它也为系统提供身份验证服务和票证授予服务（TGS）。简要地说，TGS 向请求的用户或服务发行票证，然后将票证提供给请求的服务，以证明用户（或服务）在票证有效期内的身份（默认为 10 小时）。Kerberos 有许多细微差别，包括定义用于标识系统用户和服务的 principal，票证续订，委托令牌处理等。请参见 Kerberos 安全工件概述。

此外，这些过程大部分完全透明地发生。例如，集群的业务用户只需在登录时输入密码，票证处理，加密和其他详细信息就会在后台自动进行。此外，由于使用了票证和 Kerberos 基础结构中的其他机制，用户不仅通过了单个服务目标，还通过了整个网络的身份验证。

5.2.2. Kerberos 部署模型

可以在符合 LDAP 的身份/目录服务（例如 OpenLDAP 或 Microsoft Active Directory）中存储和管理 Kerberos 身份验证所需的凭据。

一次，Microsoft 提供了一种独立的服务，即 Active Directory 服务现在打包为 Microsoft Server Domain Services 的一部分。在 2000 年代初期，Microsoft 用 Kerberos 取代了其 NT LAN Manager 身份验证机制。这意味着运行 Microsoft Server 的站点可以将其集群与 Active Directory for Kerberos 集成在一起，并将凭据存储在同一服务器的 LDAP 目录中。

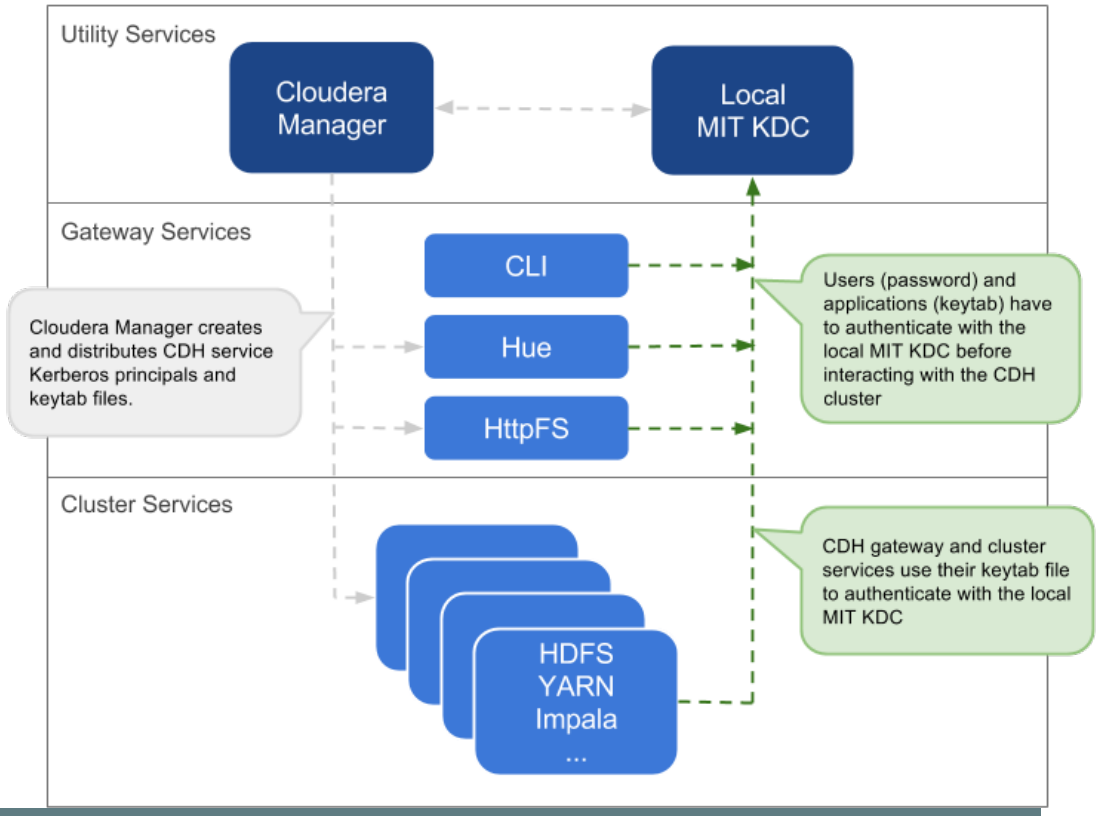
本节概述了可用于将 Kerberos 身份验证与 Cloudera 集群集成的不同部署模型，以及可用方法的一些优点和缺点。

5.2.2.1. 本地 MIT KDC

此方法使用集群本地的 MIT KDC。用户和服务可以与本地 KDC 进行身份验证，然后才能与集群上的 CDH 组件进行交互。

5.2.2.1.1. 架构摘要

- MIT KDC 和单独的 Kerberos 领域本地部署到 CDH 集群。本地 MIT KDC 通常部署在实用程序主机上。为了获得高可用性，其他复制的 MIT KDC 是可选的。
- 必须将所有集群主机配置为使用该 `krb5.conf` 文件使用本地 MIT Kerberos 领域。
- 必须在本地 MIT KDC 和 Kerberos 领域中创建所有**服务和用户 principal**。
- 本地 MIT KDC 将同时验证服务 principal（使用 keytab 文件）和用户 principal（使用密码）。
- Cloudera Manager 连接到本地 MIT KDC，以创建和管理在集群上运行的 CDH 服务的 principal。为此，Cloudera Manager 使用在设置过程中创建的管理员 principal 和密钥表。Kerberos 向导已自动执行此步骤。有关详细信息，请参见使用向导启用 Kerberos 认证。有关手动创建管理员 principal 的信息，请参见如何配置集群以使用 Kerberos 进行认证。
- 本地 MIT KDC 管理员通常会创建所有其他用户 principal。但是，Cloudera Manager Kerberos 向导可以自动创建 principal 和密钥表文件。



优点	缺点
身份验证机制与企业的其余部分隔离。	该机制未与中央认证系统集成。
这非常容易设置，特别是如果您使用 Cloudera Manager Kerberos 向导来自动创建和分发服务 principal 和密钥表文件。	用户和服务 principal 必须在本地 MIT KDC 中创建，这可能很耗时。
	本地 MIT KDC 可能是集群的单点故障，除非可以将复制的 KDC 配置为具有高可用性。
	本地 MIT KDC 是另一个要管理的身份验证系统。

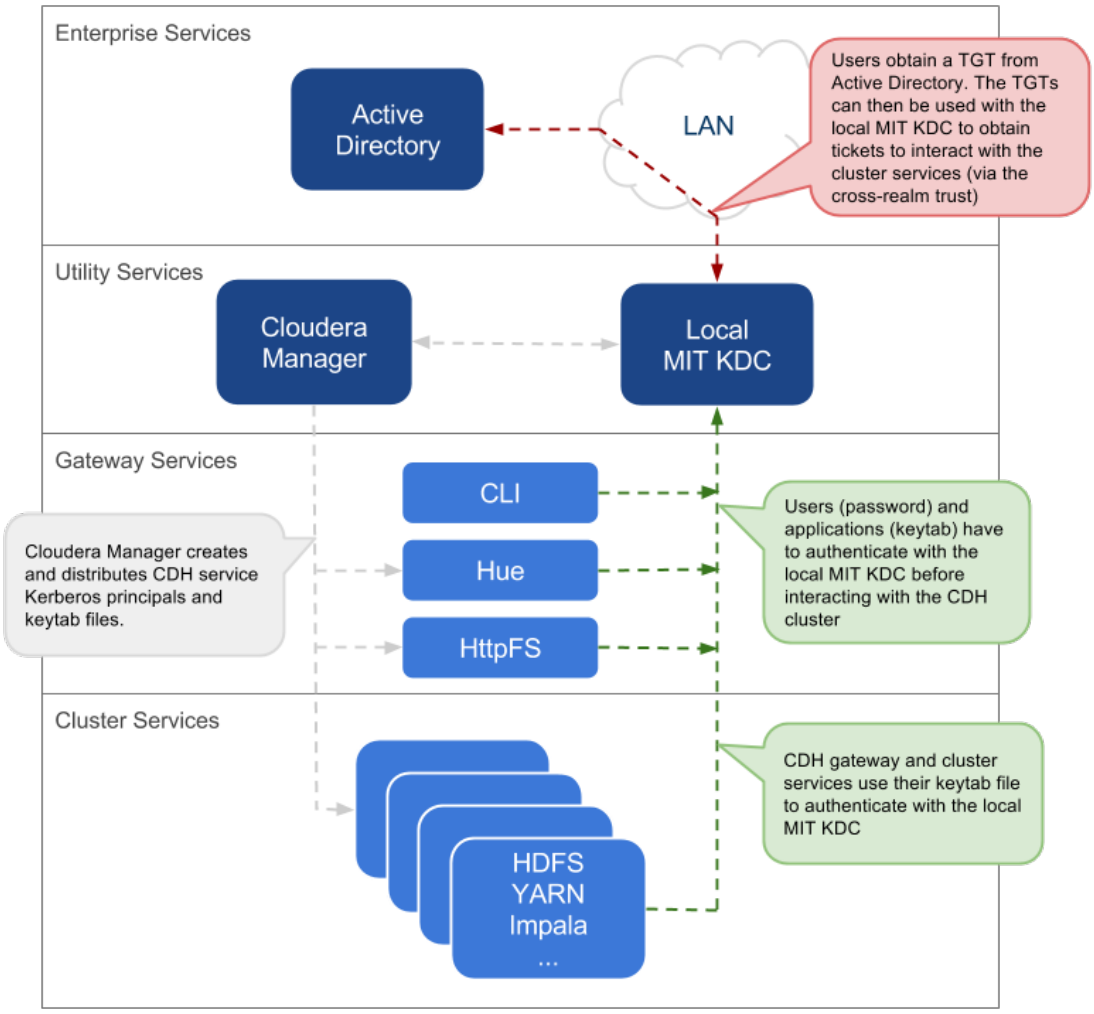
5.2.2.2. *具有 Active Directory 集成的本地 MIT KDC*

此方法使用集群本地的 MIT KDC 和 Kerberos 领域。但是，Active Directory 将将访问集群的用户 principal 存储在中央领域中。用户必须先在此中央 AD 领域进行身份验证才能获得 TGT，然后才能与集群上的 CDH 服务进行交互。请注意，CDH 服务 principal 仅驻留在本地 KDC 领域中。

5.2.2.2.1. *架构摘要*

- MIT KDC 和独特的 Kerberos 领域已本地部署到 CDH 集群。本地 MIT KDC 通常部署在实用程序主机上，并且其他复制的 MIT KDC 具有高可用性是可选的。

- 使用该 `krb5.conf` 文件，所有集群主机都配置了 Kerberos 领域（本地和中央 AD）。默认领域应为本地 MIT Kerberos 领域。
- **服务 principal** 应在本地 MIT KDC 和本地 Kerberos 领域中创建。Cloudera Manager 连接到本地 MIT KDC，以创建和管理在集群上运行的 CDH 服务的 principal。为此，Cloudera Manager 使用在安全设置期间创建的管理员 principal 和密钥表。Kerberos 向导已自动执行此步骤。
- 必须从本地 Kerberos 领域到包含要求访问 CDH 集群的**用户 principal** 的中央 AD 领域建立单向跨领域信任。无需在中央 AD 领域中创建服务 principal，也无需在本地领域中创建用户 principal。



优点	缺点
本地 MIT KDC 充当中央 Active Directory 的防护对象，以防止 CDH 集群中的许多主机和服务。在大型集群中重新启动服务会创建许多同时进行的身份验证请求。如果 Active Directory 无法处理负载激增，则集群可以有效地引起分布式拒绝服务（DDOS）攻击。	本地 MIT KDC 可以是集群的单点故障（SPOF）。可以将复制的 KDC 配置为高可用性。

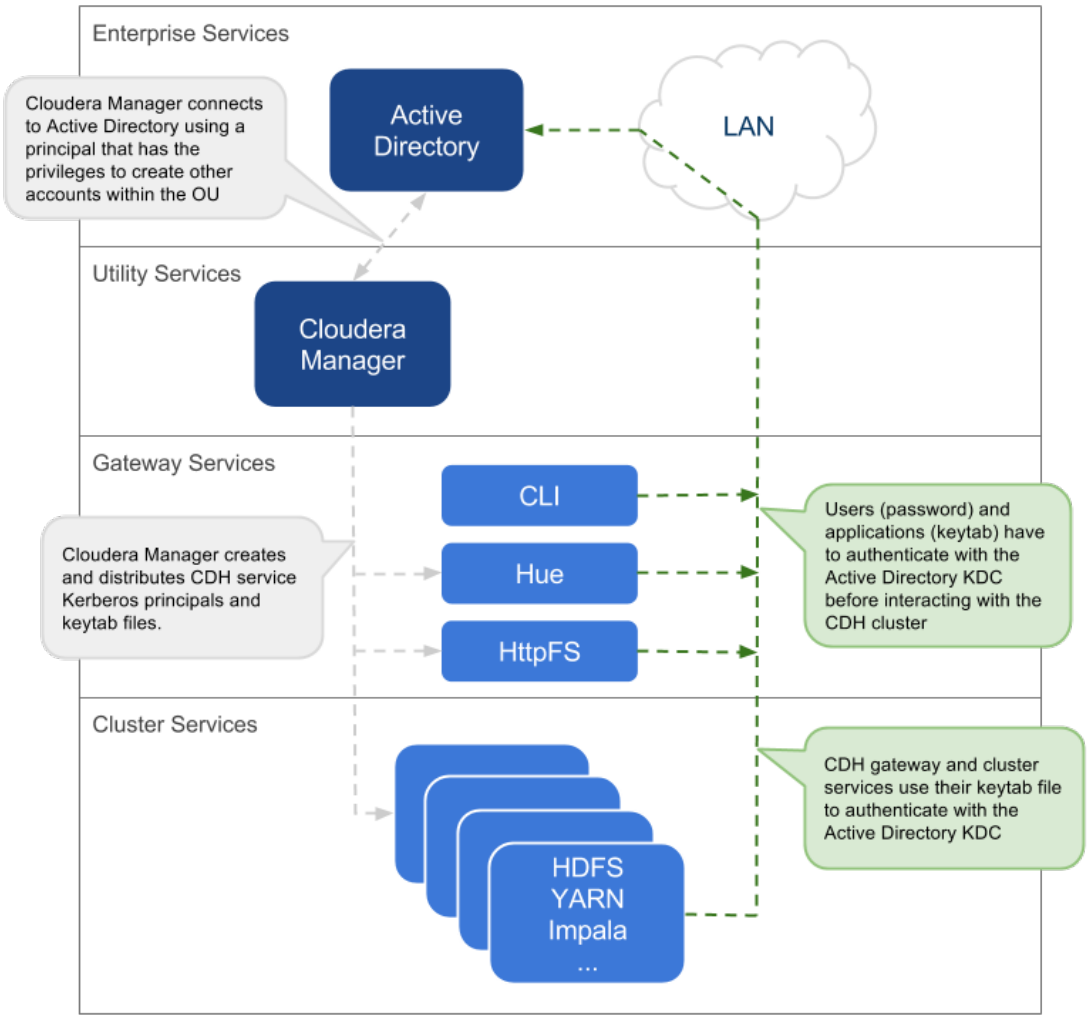
优点	缺点
这非常容易设置，特别是如果您使用 Cloudera Manager Kerberos 向导来自动创建和分发服务 principal 和密钥表文件。 Active Directory 管理员只需要在设置过程中参与配置跨域信任。	本地 MIT KDC 是另一个要管理的身份验证系统。
与中央 Active Directory 集成以进行用户 principal 身份验证可提供更完整的身份验证解决方案。	
允许增量配置。可以使用本地 MIT KDC 独立和与 Active Directory 集成来配置和验证 Hadoop 安全性。	

5.2.2.3. *使用集中式 Active Directory 服务*

这种方法使用中央 Active Directory 作为 KDC。不需要本地 KDC。在决定 AD KDC 部署之前，请确保您了解该决定的以下可能后果。

5.2.2.3.1. *架构摘要*

- 所有**服务和用户 principal** 都在 Active Directory KDC 中创建。
- 所有集群主机都使用来配置中央 AD Kerberos 领域 `krb5.conf`。
- Cloudera Manager 连接到 Active Directory KDC，以创建和管理在集群上运行的 CDH 服务的 principal。为此，Cloudera Manager 使用具有在给定组织单位（OU）内创建其他帐户的特权的 principal。（此步骤已由 Kerberos 向导自动执行。）
- 所有服务和用户 principal 均由 Active Directory KDC 进行身份验证。



注意

如果无法在 Active Directory KDC 中使用所需特权创建 Cloudera Manager 管理员 principal，则需要手动创建 CDH 服务 principal。然后，应将相应的密钥表文件安全地存储在 Cloudera Manager Server 主机上。Cloudera Manager 的“自定义 Kerberos Keytab 检索”脚本可用于从本地文件系统检索 keytab 文件。

5.2.2.3.2. *Active Directory KDC 的建议*

为身份验证请求提供服务时涉及几个不同的子系统，包括密钥分发中心（KDC），身份验证服务（AS）和票证授予服务（TGS）。集群中的节点越多，提供的服务越多，这些服务与集群上运行的服务之间的流量就越大。

作为一般准则，Cloudera 建议为集群中的每 100 个节点使用专用的 Active Directory 实例（Microsoft Server Domain Services）。但是，这只是一个宽松的准则。监控利用率并根据需要部署其他实例以满足需求。

默认情况下，Kerberos 使用 TCP 进行客户端/服务器通信，这可以保证传递，但传递数据包的速度不如 UDP。要覆盖此设置并让 Kerberos 在 TCP 之前先尝试 UDP，请如下修改 Kerberos 配置文件（`krb5.conf`）：

```
[libdefaults]
udp_preference_limit = 1
...
```

如果域控制器与集群不在同一子网上或被防火墙隔开，则此功能特别有用。

通常，Cloudera 建议在与集群相同的子网上设置 Active Directory 域控制器（Microsoft 服务器域服务），并且永远不要通过 WAN 连接进行设置。将集群与 Active Directory 域控制器上运行的 KDC 分开会导致大量延迟，并影响集群性能。

在将 Active Directory 用于 Kerberos 身份验证时，对集群操作进行故障排除需要对 Microsoft Server Domain Services 实例的管理访问权限。管理员可能需要在 Microsoft Server KDC 上启用 Kerberos 事件日志记录才能解决问题。

删除 Cloudera Manager 角色或节点需要手动删除关联的 Active Directory 帐户。Cloudera Manager 无法从 Active Directory 删除条目。

5.2.2.3.3. 与 Active Directory 的身份集成

在平台中启用 Kerberos 安全性的核心要求是用户在所有集群处理节点上均具有帐户。诸如 Centrify 或 Quest Authentication Services（QAS）之类的商业产品可将所有集群主机集成在一起，以将用户和组解析到 Active Directory。这些工具支持用户通过 AD 登录 Linux 主机时的自动 Kerberos 身份验证。对于未使用 Active Directory 的站点或希望使用开放源代码解决方案的站点，站点安全服务守护程序（SSSD）可以与 AD 或 OpenLDAP 兼容目录服务以及 MIT Kerberos 一起使用，以满足相同的需求。

对于第三方提供商，您可能必须从各自的供应商处购买许可证。此过程需要一些计划，因为要花费时间来获取这些许可证并将这些产品部署在集群上。当计算机加入 AD 域时，应注意确保身份管理产品不将服务 principal 名称（SPN）与主机 principal 相关联。

例如，默认情况下，“集中化”将 HTTP SPN 与主机 principal 相关联。因此，当主机加入域时，应特别排除 HTTP SPN。

您还需要在 AD 中完成以下设置任务：

- **Active Directory 组织单位（OU）和 OU 用户** -应该在 Active Directory 中创建一个单独的 OU，以及一个有权在该 OU 中创建其他帐户的帐户。
- **为 AD 启用 SSL** -Cloudera Manager 应该能够通过 LDAPS（TCP 636）端口连接到 AD。
- **principal 和密钥表** -在使用 Kerberos 向导设置的直接 AD 部署中，默认情况下，所有必需的 principal 和密钥表将由 Cloudera Manager 创建，部署和管理。但是，如果由于某种原因您不能允许 Cloudera Manager 管理直接到 AD 的部署，则应该在 AD 中为在每个主机上运行的每个服务手动创建唯一帐户，并且必须提供相同的 keytab 文件。这些帐户应将 AD 用户 principal 名称（UPN）设置为 `service/fqdn@REALM`，并将服务 principal 名称（SPN）设置为 `service/fqdn`。keytab 文件中的 principal 名称应为帐户的 UPN。密钥表文件应遵循命名约定： `servicename_fqdn.keytab`。必须为它们在其上运行的每个主机创建以下 principal 和 keytab 文件：Hadoop 用户（user：group）和 Kerberosprincipal。
- **AD 绑定帐户** -创建一个将在 Hue，Cloudera Manager 和 Cloudera Navigator 中用于 LDAP 绑定的 AD 帐户。
- **特权用户的** AD 组-创建 AD 组并为授权用户，HDFS 管理员和 HDFS 超级用户组添加成员。
    - 授权用户–由需要访问集群的所有用户组成的组
    - HDFS 管理员–将运行 HDFS 管理命令的用户组
    - HDFS 超级用户–需要超级用户特权（即对 HDFS 中所有数据和目录的读/写访问权限）的用户组
    - *不*建议将普通用户放入 HDFS 超级用户组。相反，管理员将问题升级到的帐户应成为 HDFS 超级用户组的一部分。
- **用于基于角色访问 Cloudera Manager 和 Cloudera Navigator 的** AD 组-创建 AD 组并将成员添加到这些组中，以便您以后可以配置对 Cloudera Manager 和 Cloudera Navigator 的基于角色的访问。
- **AD 测试用户和组** -应该至少提供一个现有的 AD 用户和该用户所属的组，以测试授权规则是否按预期工作。

5.2.3. 使用 TLS/SSL 进行安全的 Keytab 分发

Kerberos keytab 文件在 Cloudera Manager 集群中的主机之间，Cloudera Manager Server 和 Cloudera Manager Agent 主机之间传输。为了确保此敏感数据的安全，请配置

Cloudera Manager 服务器和 Cloudera Manager 代理主机以使用 TLS/SSL 进行加密通信。

有关详细信息，请参见在传输中加密数据。

5.2.4. 使用向导或手动过程来配置 Kerberos 身份验证

Cloudera 不提供 Kerberos 实现，但使用现有的 Kerberos 部署来验证服务和用户。Kerberos 服务器可以设置为专门供集群使用（例如 Local MIT KDC），也可以是组织中其他应用程序使用的分布式 Kerberos 部署。

无论采用哪种部署模型，都可以在配置集群使用集群之前使用 Kerberos 实例。此外，集群本身也应该是可操作的，并且在理想情况下，应配置为对 Cloudera Manager 服务器和 Cloudera Manager 代理主机使用 TLS/SSL，如上所述。

当您准备好将集群与组织的 MIT KDC 或 Active Directory KDC 集成时，可以使用 Cloudera Manager Server 中提供的向导或遵循以下手动过程来实现：

- 使用向导启用 Kerberos 身份验证
- 如何配置集群以使用 Kerberos 进行身份验证

5.2.5. 集群组件使用的身份验证机制

组件或产品	支持的认证机制
Accumulo	Kerberos (partial)
Backup and Disaster Recovery	Kerberos (used to authenticate Cloudera Manager to Kerberos-protected services), LDAP, SAML
Cloudera Manager	Kerberos (used to authenticate Cloudera Manager to Kerberos-protected services), LDAP, SAML
Cloudera Navigator	Active Directory, OpenLDAP, SAML
HBase	Kerberos, user-based authentication required for HBase Thrift and REST clients
HDFS	Kerberos, SPNEGO (HttpFS)
HiveServer	None
HiveServer2	Kerberos, LDAP, Custom/pluggable authentication
Hive Metastore	Kerberos
Hue	Kerberos, LDAP, SAML, Custom/pluggable authentication
Impala	Kerberos, LDAP, SPNEGO (Impala Web Console)

组件或产品	支持的认证机制
Kudu	Kerberos
MapReduce	Kerberos *(also see HDFS)*
Oozie	Kerberos, SPNEGO
Pig	Kerberos
Search	Kerberos, SPNEGO
Spark	Kerberos
Sqoop	Kerberos
Sqoop2	Kerberos (as of CDH 5.4)
YARN	Kerberos *(also see HDFS)*
Zookeeper	Kerberos

5.3. 加密概述

加密是使用数字密钥对各种组件（例如文本、文件、数据库、密码、应用程序或网络数据包）进行编码的过程，因此只有适当的实体（用户、系统进程等）才能进行解码（解密）项，然后查看、修改或添加到数据中。Cloudera 提供了加密机制来保护持久保存在磁盘或其他存储介质上的数据（静态*数据*或简单地称为数据加密）以及在网络上移动时的*数据（传输加密中的数据）*。

在政府、卫生、金融、教育和许多其他环境中，数据加密是强制性的。例如，《联邦信息安全管理法案》（FISMA）规范了患者的隐私问题，而《支付卡行业数据安全标准》（PCI DSS）规范了信用卡处理器的信息安全性。这只是两个例子。

尽管如此，使用案例中需要使用何种程度的隐私、机密性和数据完整性的 Cloudera 集群中包含的大量数据（使用许多不同的组件进行部署）仍然必须支持。Cloudera 支持并在本概述中讨论的加密机制旨在实现这一目标。

5.3.1. 保护静态数据

保护静止数据通常意味着对存储在磁盘上的数据进行加密，并允许授权用户和进程（仅授权用户和进程）在手头的应用程序或任务需要时解密数据。对于静态数据加密，必须分发和管理加密密钥，应定期旋转或更改密钥（以减少密钥被泄露的风险），并且许多其他因素使该过程复杂化。

但是，仅加密数据可能不够。例如，管理员和其他具有足够特权的人可能有权访问日志文件，审核数据或 SQL 查询中的个人身份信息（PII）。根据特定的用例，在医院或财务环境中，可能需要从所有此类文件中删除 PII，以确保对日志和查询具有特权的用户（其中可能包含敏感数据）仍然无法在查看数据时使用不应该。

Cloudera 提供了补充方法来加密静态数据，并提供了屏蔽日志文件，审核数据和 SQL 查询中的 PII 的机制。

5.3.1.1. 可用的加密选项

Cloudera 提供了多种机制来确保敏感数据的安全。CDH 提供透明的 HDFS 加密，确保所有敏感数据在存储到磁盘之前都已加密。通过将 HDFS 加密与 Navigator Key Trustee 的企业级加密密钥管理结合使用，可以使大多数企业遵守法规。对于 Cloudera Enterprise，可以通过 Navigator Encrypt 增强 HDFS 加密，以保护数据之外的元数据。考虑到数据节点是并行加密的，使用这些解决方案的 Cloudera 集群照常运行，并且对性能的影响非常低。随着集群的增长，加密也随之增长。

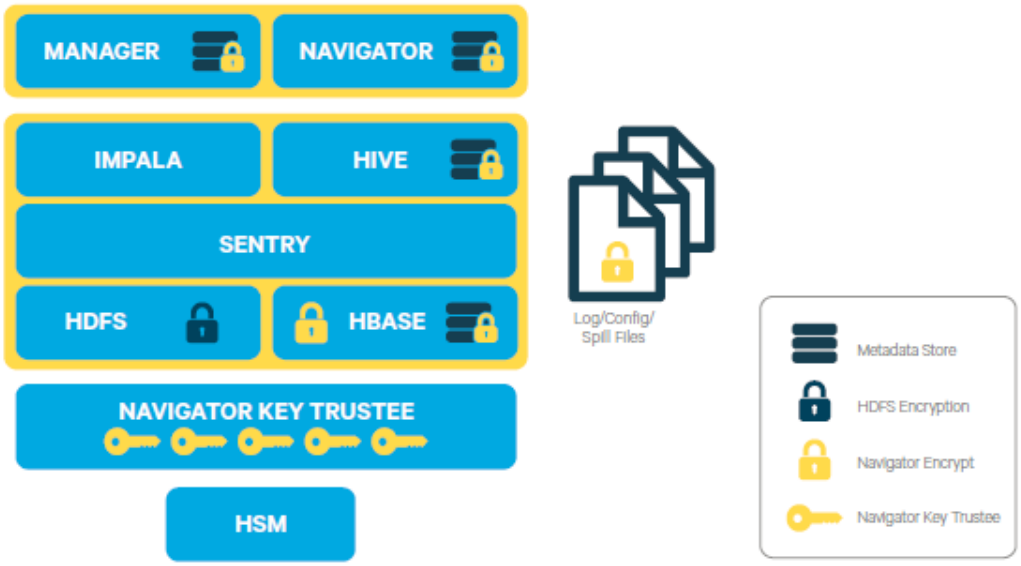
此外，此透明加密针对英特尔芯片组进行了优化，以实现高性能。英特尔芯片组包括 AES-NI 协处理器，该处理器提供特殊功能，可使加密工作负载运行得非常快。Cloudera 利用 Intel 最新的技术进步来获得更快的性能。

密钥受托者 KMS 与密钥受托者服务器和密钥 HSM 结合使用，可为存储的密钥材料提供基于 HSM 的保护。密钥受托者 KMS 在 KMS 上本地生成加密区域密钥材料，然后使用 HSM 生成的密钥对该密钥材料进行加密。相反，Navigator HSM KMS 服务依赖于 HSM 来生成和存储所有加密区域密钥。使用 Navigator HSM KMS 时，加密区域密钥材料起源于 HSM，并且永远不会离开 HSM。这样可以实现最高级别的密钥隔离，但是需要一些网络开销来进行 HSM 的网络调用，以进行密钥生成，加密和解密操作。对于大多数生产方案，密钥受托人 KMS 仍然是建议的 HDFS 加密密钥管理解决方案。

下图显示了使用示例部署：

- Cloudera 透明 HDFS 加密可加密 HDFS 上存储的数据
- Navigator Encrypt 对与 Cloudera Manager、Cloudera Navigator、Hive 和 HBase 相关的所有其他数据（包括元数据，日志和溢出数据）进行加密

- Navigator Key Trustee，用于进行健壮、容错的密钥管理



除了对 Cloudera 集群的数据层应用加密之外，还可以在网络层应用加密，以加密集群节点之间的通信。

加密不会阻止对集群具有完全访问权限的管理员查看敏感数据。要混淆包括 PII 在内的敏感数据，可以配置集群以进行数据编辑。

5.3.1.2. Cloudera 集群的数据整理

编辑是一个使数据模糊的过程。它可以通过混淆个人身份信息（PII）来帮助组织遵守 PCI（支付卡行业）和 HIPAA 之类的行业法规和标准，从而使其无法使用，除非工作需要此类访问的人员才能使用。例如，HIPAA 立法要求，除适当的医生（和患者）外，任何人都不能使用患者 PII，并且不得使用任何患者的 PII 来确定个人身份或将其与健康数据相关联。通过将 PII 转换为无意义的模式（例如，将美国的社会保险号转换为 xxx-xx-xxxx 字符串。

数据编辑与 Cloudera 加密技术分开工作，Cloudera 加密技术不会阻止对集群具有完全访问权限的管理员查看敏感的用户数据。它确保集群管理员，数据分析人员和其他人员不会看到不在其工作域内的 PII 或其他敏感数据，并且同时也不会阻止具有适当权限的用户访问他们拥有特权的数据。

5.3.2. 保护传输中的数据

对于传输中的数据，实施数据保护和加密相对容易。有线加密内置在 Hadoop 堆栈中（例如 SSL），并且通常不需要外部系统。使用会话级一次性密钥通过会话握手以及立即传输和后续传输来构建此传输中数据加密。因此，由于密钥的临时性，传输中的数据避免了许多与静态数据相关的密钥管理问题，但它确实依赖于正确的身份验证；证书泄露是身份验证的问题，但可能会破坏有线加密。顾名思义，传输中的数据涵盖了数据的安全传输和中间存储。这适用于所有过程间通信，在同一节点内或节点之间。有三种主要的沟通渠道：

- **HDFS 透明加密：**使用 HDFS 透明加密加密的数据是端到端的保护。写入和写入 HDFS 的任何数据只能由客户端加密或解密。HDFS 无权访问未加密的数据或加密密钥。这支持静态加密和传输中加密。
- **数据传输：**第一个通道是数据传输，包括将数据块读取和写入 HDFS。Hadoop 在其原生的基于 TCP/IP 的直接传输周围使用了启用 SASL 的包装器 `DataTransportProtocol`，以保护 DIGEST-MD5 信封内的 I/O 流。此过程还使用安全的 HadoopRPC（请参阅远程过程调用）进行密钥交换。但是，HttpFS REST 接口不提供客户端与 HDFS 之间的安全通信，仅提供使用 SPNEGO 进行的安全身份验证。
- 为了在 MapReduce 作业的混洗阶段（即在作业的 Map 和 Reduce 部分之间移动中间结果）期间在 DataNode 之间进行数据传输，Hadoop 使用传输层安全性（TLS）通过 HTTP Secure（HTTPS）保护了通信通道）。
- **远程过程调用：**第二个通道是对 Hadoop 集群中各种系统和框架的远程过程（RPC）的系统调用。与数据传输活动一样，Hadoop 具有自己的 RPC 本地协议，称为 HadoopRPC，用于 Hadoop API 客户端通信，Hadoop 内部服务通信以及监控，心跳以及其他非数据，非用户活动。HadoopRPC 支持 SASL，以实现安全传输，并且默认设置为 Kerberos 和 DIGEST-MD5，具体取决于通信类型和安全设置。
- **用户界面：**第三个渠道包括 Hadoop 集群中各种基于 Web 的用户界面。对于安全运输，解决方案很简单；这些接口使用 HTTPS。

5.3.2.1. TLS/SSL 证书概述

可以使用三种不同的方式对证书进行签名：

类型	使用说明
公共 CA 签名的证书	**推荐**。使用由受信任的公共 CA 签名的证书可以简化部署，因为默认的 Java 客户端已经信任大多数公共 CA。从受信任的著名（公共）CA（例如 Symantec 和 Comodo）中获取证书

类型	使用说明
内部 CA 签署的证书	如果您的组织有自己的证书，请从组织的内部 CA 获取证书。使用内部 CA 可以降低成本（尽管集群配置可能需要为内部 CA 签名的证书建立信任链，具体取决于您的 IT 基础结构）。
自签名证书	不建议用于生产部署。使用自签名证书要求将每个客户端配置为信任特定证书（除了生成和分发证书之外）。但是，自签名证书适用于非生产（测试或概念验证）部署。

5.3.2.2. CDH 组件的 TLS/SSL 加密

Cloudera 建议在集群上启用 SSL 之类的加密之前，先使用 Kerberos 身份验证保护集群。如果为尚未配置 Kerberos 身份验证的集群启用 SSL，将显示警告。

Hadoop 服务在 SSL 的使用方面有所不同，如下所示：

- HDFS，MapReduce 和 YARN 守护程序既充当 SSL 服务器又充当客户端。
- HBase 守护程序仅充当 SSL 服务器。
- Oozie 守护程序仅充当 SSL 服务器。
- Hue 充当上述所有内容的 SSL 客户端。

启动时，充当 SSL 服务器的守护程序将加载密钥库。当客户端连接到 SSL 服务器守护程序时，服务器会将在启动时加载的证书传输到客户端，然后客户端使用其信任库来验证服务器的证书。

有关为 CDH 服务设置 SSL/TLS 的信息，请参阅适用的组件指南。

5.3.3. Hadoop 项目中的数据保护

下表列出了 CDH 组件和 Cloudera Manager 可以利用的各种加密功能。

Project	Encryption for Data-in-Transit	Encryption for Data-at-Rest (HDFS Encryption + Navigator Encrypt + Navigator Key Trustee)
HDFS	SASL (RPC), SASL (DataTransferProtocol)	Yes
MapReduce	SASL (RPC), HTTPS (encrypted shuffle)	Yes

Project	Encryption for Data-in-Transit	Encryption for Data-at-Rest (HDFS Encryption + Navigator Encrypt + Navigator Key Trustee)
YARN	SASL (RPC)	Yes
Accumulo	Partial - Only for RPCs and Web UI (Not directly configurable in Cloudera Manager)	Yes
Flume	TLS (Avro RPC)	Yes
HBase	SASL - For web interfaces, inter-component replication, the HBase shell and the REST, Thrift 1 and Thrift 2 interfaces	Yes
HiveServer2	SASL (Thrift), SASL (JDBC), TLS (JDBC, ODBC)	Yes
Hue	TLS	Yes
Impala	TLS or SASL between impalad and clients, but not between daemons	
Oozie	TLS	Yes
Pig	N/A	Yes
Search	TLS	Yes
Sentry	SASL (RPC)	Yes
Spark	None	Yes
Sqoop	Partial - Depends on the RDBMS database driver in use	Yes
Sqoop2	Partial - You can encrypt the JDBC connection depending on the RDBMS database driver	Yes
ZooKeeper	SASL (RPC)	No

Project	Encryption for Data-in-Transit	Encryption for Data-at-Rest (HDFS Encryption + Navigator Encrypt + Navigator Key Trustee)
Cloudera Manager	TLS - Does not include monitoring	Yes
Cloudera Navigator	TLS - *Also see Cloudera Manager*	Yes
Backup and Disaster Recovery	TLS - *Also see Cloudera Manager*	Yes

5.3.4. 加密机制概述

静态数据和传输加密中的数据在集群的不同技术层起作用：

层	描述
应用	HDFS 透明加密由 HDFS 客户端软件应用，可让您加密 HDFS 中包含的特定文件夹。为了安全地存储所需的加密密钥，Cloudera 建议将 Cloudera Navigator 密钥受托服务器与 HDFS 加密结合使用。 还可以对 CDH 组件（包括 Impala，MapReduce，YARN 或 HBase）在 HDFS 外部临时存储在本地文件系统上的数据进行加密。
操作系统	在 Linux OS 文件系统层，可以将加密应用于整个卷。例如，Cloudera Navigator Encrypt 可以加密 HDFS 内部和外部的数据，例如临时/溢出文件，配置文件以及存储与 CDH 集群关联的元数据的数据库。Cloudera Navigator Encrypt 作为 Linux 内核模块运行，是操作系统的一部分。Navigator Encrypt 需要 Cloudera Navigator 的许可证，并且必须配置为使用 Navigator Key Trustee Server。
网络	客户端进程和服务器进程（HTTP，RPC 或 TCP/IP 服务）之间的网络通信可以使用行业标准的 TLS/SSL 进行加密。

5.4. 授权概述

授权是任何计算环境的基本安全要求之一。其目标是确保只有适当的人员或流程才能访问，查看，使用，控制或更改特定的资源，服务或数据。在使用各种 CDH 组件（Hive，HDFS，Impala 等）部署来满足特定工作负载的任何集群中，不同的授权机制可以确保只有授权的用户或进程才能根据需要访问数据，系统和其他资源。理想情况下，授权机

制可以利用身份验证机制，以便当用户登录系统（例如集群）时，将根据他们在系统中对应用程序，数据和其他资源的授权，对他们进行透明授权。 。

例如，可以将 Cloudera CDH 集群配置为利用组织的 Active Directory（或其他 LDAP 可访问目录）实例中存在的用户帐户和组帐户。

本指南后面将讨论各种可能的配置和集成。

5.4.1. Hadoop 中的授权机制

Hadoop 支持多种授权机制，包括：

- 对文件和目录的传统 POSIX 样式权限。每个目录和文件都有一个具有基本权限的所有者和组，可以将其设置为读取、写入和执行（在文件级别）。目录具有附加权限，该权限允许访问子目录。
- 访问控制列表（ACL），用于管理服务和资源。例如，Apache HBase 使用 ACL 来授权各种操作（读、写、创建、管理）（按列、列族和列族限定符）。将 HBase ACL 授予并撤消给用户和组。可以使用 Apache HDFS ACL 将细粒度权限应用于 HDFS 文件，以设置特定命名用户和命名组的权限。
- Apache Ranger 通过管理访问控制，并确保跨集群服务进行一致的策略管理。
- Apache Ranger 还提供了一个集中式框架，用于收集访问审核历史记录和报告数据，包括过滤各种参数。

5.4.1.1. POSIX 权限

在 Hadoop 集群上运行的大多数服务，例如命令行界面（CLI）或使用 Hadoop API 的客户端应用程序，都可以直接访问 HDFS 中存储的数据。HDFS 对目录和文件使用 POSIX 样式的权限；每个目录和文件都分配有一个所有者和组。每个分配都有一组基本的可用权限。文件权限被读取、写入和执行，并且目录具有附加权限来确定对子目录的访问。

给定 HDFS 资产的所有权和组成员资格确定用户的特权。如果给定用户未通过这些条件之一，则将拒绝他们访问。对于可能尝试访问多个文件的服务（例如 MapReduce、Cloudera Search 等），将为每次文件访问尝试分别确定数据访问。HDFS 中的文件权限由 NameNode 管理。

5.4.1.2. 访问控制列表

除了每个服务内和 HDFS 中的数据外，Hadoop 还为服务本身维护常规访问控制。服务访问控制列表（ACL）通常在全局 hadoop-policy.xml 文件中定义，范围从 NameNode

访问到客户端到 DataNode 通信。在 MapReduce 和 YARN 的上下文中，用户和组标识符构成确定作业提交或修改权限的基础。

此外，借助 MapReduce 和 YARN，可以使用由调度程序控制的队列来提交作业，调度程序是组成集群内资源管理功能的组件之一。管理员使用 ACL 定义对各个队列的权限。ACL 也可以按工作定义。像 HDFS 权限一样，本地用户帐户和组必须在每个执行服务器上都存在，否则，除超级用户帐户外，队列将无法使用。

Apache HBase 还使用 ACL 进行数据级授权。HBase ACL 按列、列族和列族限定符授权各种操作（读取、写入、创建、管理）。HBase ACL 被授予和撤销给用户和组。类似于 HDFS 权限，本地用户帐户是正确授权所必需的。

Apache ZooKeeper 还维护对存储在 ZooKeeper 数据树的 DataNodes 中的信息的 ACL。

5.4.2. 与身份验证机制的身份验证机制集成

像许多分布式系统一样，Hadoop 项目和工作负载通常由协同工作的一系列流程组成。在某些情况下，初始用户流程会在整个工作负载或作业的整个生命周期中进行授权。但是对于产生其他流程的流程，授权可能会带来挑战。在这种情况下，将生成的进程设置为好像已通过身份验证的用户（即 setuid）一样执行，因此仅具有该用户的特权。总体系统需要映射到已验证的 principal，并且该用户帐户必须在本地主机系统上存在才能使 setuid 成功。

重要

- Cloudera 强烈建议您*不要*使用 Hadoop 的 `LdapGroupsMapping` 提供程序。 `LdapGroupsMapping` 仅应在无法进行 OS 级集成的情况下使用。生产集群需要一个身份提供程序，该身份提供程序必须能够与所有应用程序（而不只是 Hadoop）良好地配合使用。因此，通常首选的机制是使用 SSSD、VAS 或 Centrify 之类的工具来复制 LDAP 组。
- Cloudera 不支持在生产环境中使用 Winbind。Winbind 使用低效的方法来进行用户/组映射，这可能会导致性能下降或集群故障，因为集群的大小以及用户和组的数量会增加。

无论使用哪种机制，都必须在所有集群主机上一致地应用用户/组映射，以便于维护。

系统和服务授权 -某些 Hadoop 服务仅限于服务之间的交互，并不打算供最终用户访问。这些服务确实支持身份验证，以防止未经授权或恶意的用户。但是，任何具有登录凭据并可以向该服务进行身份验证的用户，或更通常是另一个服务，都有权执行目标服务允许的所有操作。例如，ZooKeeper（由 YARN、Cloudera Search 和 HBase 等内部系统使用）和 Flume（由 Hadoop 管理员直接配置，因此不提供用户控件）。

每当这些“系统”服务访问其他服务（例如 HDFS、HBase 和 MapReduce）时，都会对经过身份验证的 Kerberos principal 进行检查，因此必须授权使用这些资源。因此，Flume 没有明确的授权模型这一事实并不意味着 Flume 可以不受限制地访问 HDFS 和其他服务。仍然必须对 Flume 服务 principal 进行 HDFS 文件系统特定位置的授权。Hadoop 管理员可以为诸如 Flume 之类的服务建立单独的系统用户，以对特定 Flume 应用程序的文件系统的各个部分进行分段和施加访问权限。

5.4.3. Hadoop 项目中的授权

项目	授权能力
HDFS	File Permissions, Ranger
MapReduce	File Permissions, Ranger
YARN	File Permissions, Ranger
Accumulo	Ranger
HBase	HBase ACLs, Ranger
HiveServer2	File Permissions, Ranger
Hue	Hue authorization mechanisms (assigning permissions to Hue apps)
Impala	Ranger
Oozie	ACLs
Pig	File Permissions
Search	File Permissions
Spark	File Permissions, Ranger
Sqoop	N/A
Sqoop2	None
ZooKeeper	ACLs
Cloudera Manager	Cloudera Manager roles

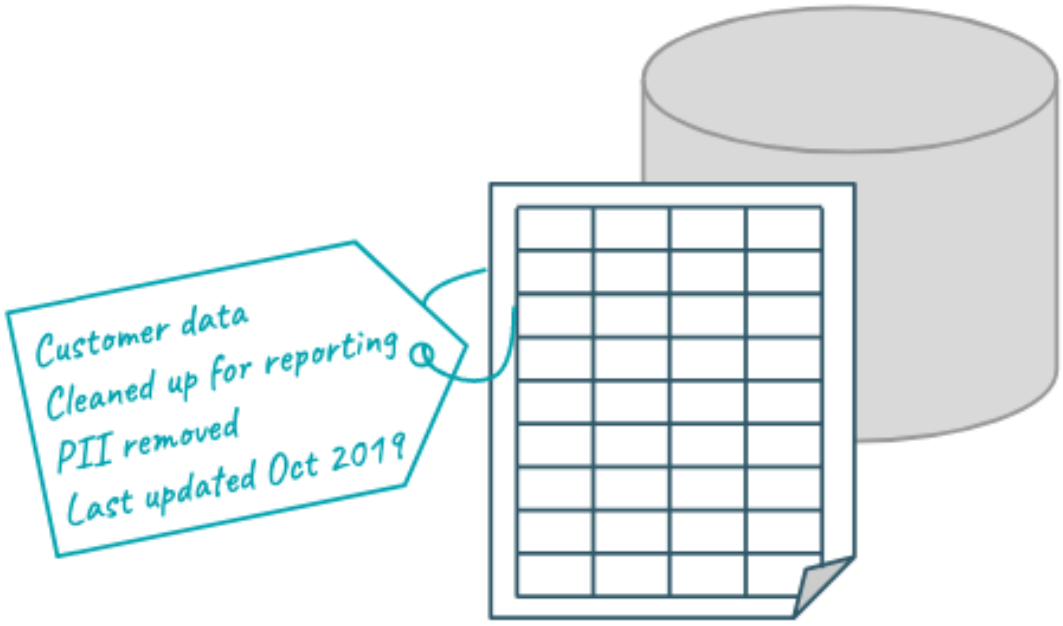
项目	授权能力
Backup and Disaster Recovery	N/A

5.5. 治理概述

收集，创建和使用元数据的概念。

5.5.1. 什么是 Apache Atlas?

Atlas 是一个元数据管理和治理系统，旨在帮助您查找，组织和管理数据资产。Atlas 在数据湖中创建对象和操作的“实体”或元数据表示。您可以将业务元数据添加到这些实体，以便您可以使用业务词汇表来更轻松地搜索特定资产。



5.5.2. Apache Atlas 使用元数据创建血统关系

Atlas 读取其收集的元数据的内容，以建立数据资产之间的关系。当 Atlas 接收查询信息时，它将记录查询的输入和输出，并生成沿袭图谱，该图谱可跟踪数据的使用方式和随时间变化的方式。数据转换的这种可视化使治理团队可以快速识别数据源，并了解数据和架构更改的影响。

5.5.3. 添加到实体元数据使搜索更加容易

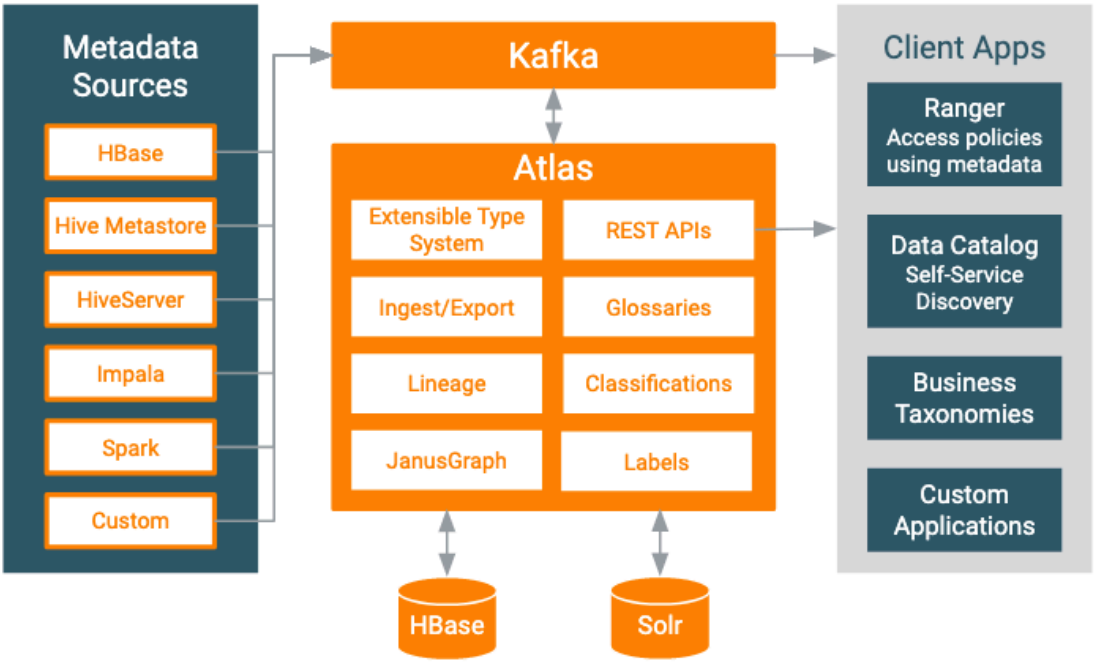
Atlas 管理您创建并用于增强数据资产元数据的*分类*和*标签*。您可以创建和组织分类和标签，以用于从标识数据清理阶段到记录用户对特定数据资产的评论和见解的任何事

情。使用分类时，“ Atlas 仪表板”使搜索，分组，报告和进一步注释您标记的实体变得容易。分类本身可以组织为层次结构，以使其更易于管理。

Atlas 还提供了用于创建和维护业务本体以标记数据资产的基础架构。Atlas 的“词汇表”包括“术语”，因此您可以为部门或组织范围内的词汇表建立商定的清单，以识别和管理数据。添加术语可以使您单击该术语所标识的实体的报告。

5.5.4. Apache Atlas 体系结构

Atlas 在 Hadoop 环境中作为独立服务运行。许多 Hadoop 数据处理和存储服务都包含 Atlas 附加组件，这些附加组件将服务活动的元数据发布到 Kafka 消息主题。Atlas 读取消息并将其存储在 JanusGraph 中以对实体之间的关系建模。JanusGraph 背后的数据存储区是 HBase。Atlas 将搜索索引存储在 Solr 中，以利用 Solr 的搜索功能。



存在针对 Hive，Impala，Kafka，NiFi，Spark 和 Sqoop 的预定义挂钩。

Atlas 还提供了“桥梁”，可为给定源中的所有现有数据资产导入元数据。例如，如果在 Hive 中创建数据库和表之后启动 Atlas，则可以使用 Hive 桥导入现有数据资产的元数据。网桥使用 Atlas API 导入元数据，而不是将消息发布到 Kafka。

如果您需要挂钩或桥接来自动从另一个来源收集元数据，请使用 Atlas Java API 创建自定义的 Atlas 插件。

6. Cloudera 最佳实践

最佳实践库是技巧、窍门和行之有效的集合，这些秘诀、技巧和行之有效的方法是由我们在全球范围内和在现场工作的 Cloudera 工程师所提供的。这些建议代表与客户合作解决金融服务、电信、医疗保健、公共部门、技术和制造行业中与企业级数据管理相关的一些最复杂技术问题的丰富经验。

6.1. Impala 分区

本文档为处理小文件和使用 Impala 表进行分区提供了最佳实践建议。此处包含的准则仅适用于 HDFS 支持的表。当您使用诸如 Amazon S3 之类的云服务时，由于存在不同的条件，因此适用不同的准则。尽管本文档的重点是对 Impala 的分区建议，但是这些准则也可以应用于对 Hive 表进行分区。

6.1.1. 文件计数和文件大小

文件的数量及其大小对于有效的表分区至关重要。

Impala 使用与要扫描的文件数量一样多的节点来扫描 HDFS 表。例如，要扫描具有 150 个文件的 HDFS 表，Impala 最多可以使用 150 个节点。因此，根据更频繁查询的内容，Cloudera 建议在一个表或一组分区中至少拥有与集群中节点数量一样多的文件。

通过使文件数少于节点数来限制并发性，将影响表扫描以及在同一查询片段上运行的所有其他操作。对于频繁使用的表，这使“热点”发生的可能性更高。在这种情况下，由于少数节点执行大多数操作而导致集群的整体性能受到限制时，*就会发生热点*。

另一方面，过多的小文件也会损害整体性能，因为 Impala 必须对 HDFS Namenode 进行代价高昂的远程过程调用（RPC），才能打开每个文件进行读取。降级的确切级别取决于几个因素，例如整个 Namenode 负载、文件数、网络延迟和查询 SQL。小文件还会给 Namenode 和 Impala 的 Catalog Service（`catalogd`）带来额外的压力。另请参阅文件句柄缓存的可伸缩性注意事项，该文件将文件句柄缓存作为解决性能问题的一种方法。如果您使用的是 CDH 5.14 或更早版本，请尝试调整文件句柄缓存，然后再尝试调整分区策略。

目标是在文件大小和计数之间取得平衡。通常，最佳文件大小为 256 MB。对于此文件大小可能会限制并行性的表，因为这些文件的数量少于集群上节点的数量，请减小文件大小以充分利用集群。在这些情况下，文件大小为 64 MB 甚至 32 MB 可能会有用。

当表达到 10-15,000 个文件且平均文件大小为 256 MB 时，如果表访问模式允许，则进一步压缩表或拆分表，同时保留 256 MB 的文件大小。后一种策略是更可取的。例如，如果针对该表的大多数操作仅访问较新的分区，则创建一个 `history` 具有较大文件大小的单独表是有意义的 。

6.1.2. 分区注意事项

将分区修剪合并到分区策略中很有用。

通过分区组织数据集使您可以执行分区修剪。分区修剪使查询仅扫描表数据的一个子集。通常，这是有好处的，但是创建的分区太小会导致文件问题，从而对性能产生负面影响。通常，小于 1 GB 的分区或仅包含 4 个文件的分区通常不值得创建。

您应该考虑对多大的表进行分区？例如，如果您有一个未分区的表，其中有 15 GB 的数据驻留在 60 个文件中（每个文件 256 MB）并均匀分布在 60 个节点的 HDFS 集群中，则全表扫描将读取每个节点 256 MB。将该表划分为 15 个相等的分区并过滤到单个分区，导致参与扫描的数据节点更少，但是每个节点仍必须读取 256 MB 的数据。这意味着较少的工作负载分布在整个集群中，但是运行时间大致相同。此示例代表一个最佳情况，表明对更小的表进行分区可能会有所帮助。但是，除非您有特定的分区请求，否则每个节点一个文件都是考虑何时对表进行分区的合适门槛。

仅当针对表运行的查询在 partition 列上具有谓词时，分区修剪才有效。因此，对大表进行分区仅在了解针对该表的查询时才有用。例如，如果分区策略基于该 `date` 列，但是针对表运行的大多数查询都不包含谓词 `date`，则仍然需要全表扫描，并且分区将不会提供任何好处。另请注意，动态分区修剪可能会有所帮助。如果分区是连接键的一部分，则动态分区修剪将在运行时启用分区修剪。

要刷新分区表中的数据，请使用 per-partition `REFRESH` 语句在已经存在的分区中拾取新数据。使用 `ALTER TABLE [...] RECOVER PARTITIONS` 拿起新的分区和他们的数据。这些是用于拾取新数据的最轻量级的语句。

当表达到 50,000 个分区时，最好检查一下分区和表策略。确保在表达到 100,000 个分区之前实施新策略。这些不是关于系统可以处理的严格规则，而是帮助您避免重大问题的最佳实践。

6.1.2.1. 小表注意事项

小表在分区时提出了挑战。

通常建议使用分区来加速任何查询。如前所述，对于小表或无效分区，可能并非如此。相反，请考虑对大量查询的小表使用 HDFS 缓存。将数据存储在 DataNode 的缓存中可以大大加快扫描速度，但要以在 DataNode 上分配内存为代价。

6.1.2.2. 压缩分区

如何以最少的中断量更改分区策略。

在减少分区数量时，最好保留尽可能多的原始分区策略收益。如果您不能影响最终用户查询的编写方式（这可能会使用原始分区方案），那么这可能不是一件容易的事。例如，过去曾经有一个具有 `string` 数据类型键（例如“1970-01-01”）的每日分区的表可能必须压缩成年度分区，并以 `integer` 数据类型（例如 1970）存储。得出可接受的文件和分区大小：

```
CREATE TABLE _OLD (c1 int) PARTITIONED BY (_day string);
CREATE TABLE _NEW (c1 int, _day string) PARTITIONED BY (year int);
```

添加了非分区列 `_day`，以避免丢失数据。要为筛选条件为的查询保留分区修剪 `_day`，可以定义一个视图：

```
CREATE VIEW _NEW_VIEW AS SELECT * FROM _NEW WHERE year = year(_day);
```

当查询针对该视图运行时，其 `_day` 谓词将转换为分区过滤器。在 `year` 这种情况下内置的 UDF 是有用的，对于其他的 UDF，例如 substr 也可以同样工作。

6.1.3. 指南总结

分区最佳实践准则的总结。

- 当 Impala 针对支持 HDFS 的表运行时，扫描并行性由文件数量决定。
- 小文件会严重影响性能，并给集群（尤其是 Namenode 主机和 Impala Catalog Service 主机）施加巨大压力。当太多任务触摸相同数据时，文件太少会导致热点发现，从而可能使节点超载。
- 通常，将文件保留在 256 MB 左右，并避免分区小于 256 MB。
- 对于查询量很大的小型表，请考虑使用 HDFS 缓存。

分区最好是一种迭代过程，而不是科学过程。遵循这些简单的一般准则可以产生良好的体验。您可以根据自己独特的使用方式和经验进行进一步的改进。

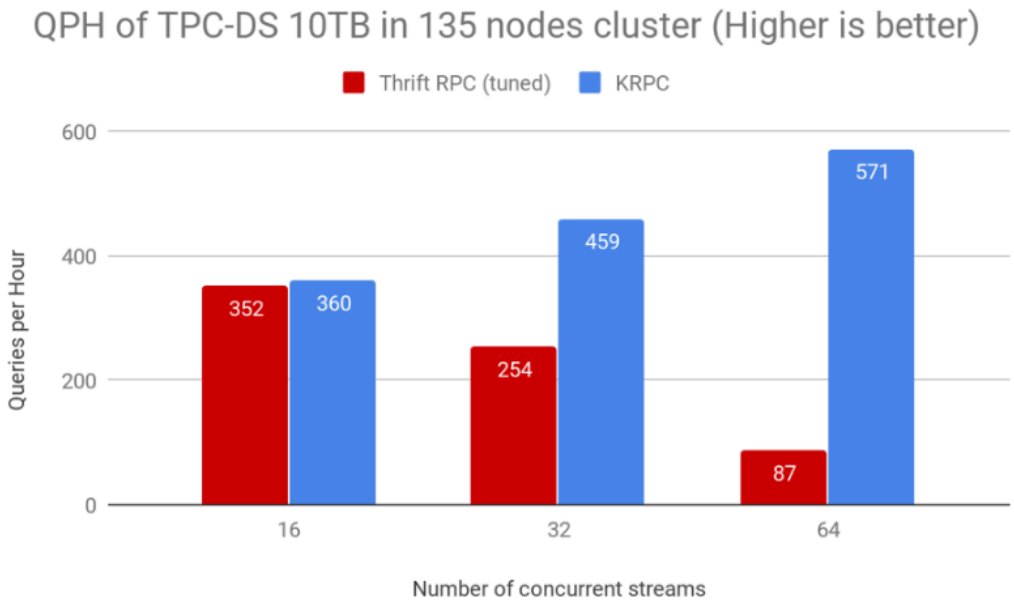
6.2. Impala 性能

本文档说明了影响 Impala 查询性能的因素。此外，它提供了用于优化、监视和基准化 Impala 查询和其他 SQL 操作的过程，以最大程度地提高 Impala 可伸缩性。可伸缩性和性能齐头并进。通过减少所使用的磁盘 I / O 或内存来提高查询性能，可以通过优化使用资源来提高可伸缩性。如果更有效地利用资源，则可以并行运行更多查询。

6.2.1. Kudu RPC

CDH 5.15，CDH 6.1 和更高版本的 Impala 使用称为 Kudu RPC（KRPC）的新网络协议在 Impala 代理之间进行通信。

KRPC 取代了较旧的 Thrift RPC 协议，并显着提高了查询性能。Cloudera 建议升级到 CDH 5.15 和 CDH 6.1 或更高版本，以从 KRPC 中受益。使用 KRPC，查询的运行速度可以提高 2-3 倍，并且成功率更高：



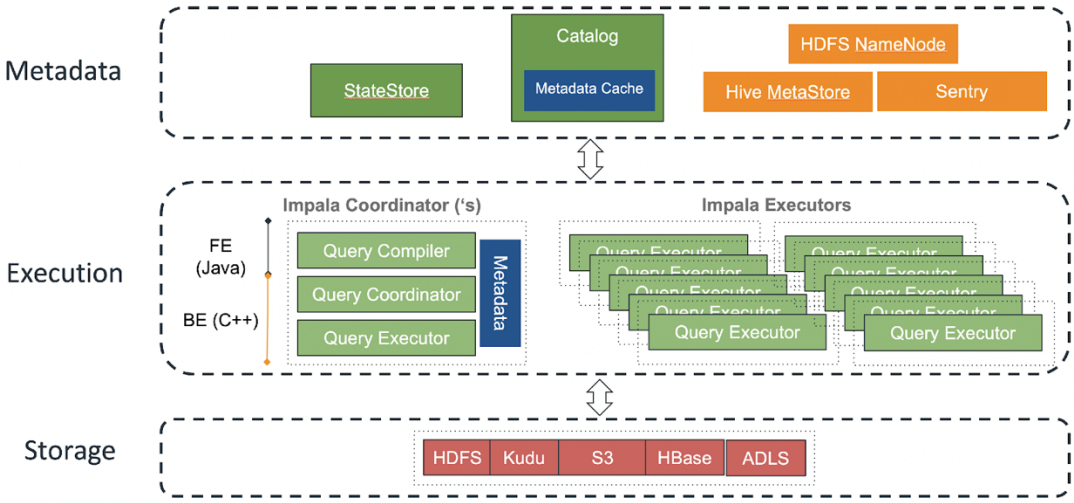
KRPC 的主要优点：

- 减少集群中的连接总数。
- 减轻对 MIT KDC 或 Active Directory KDC 的压力。
- 支持在每对主机之间使用每个方向一个连接的连接多路复用

6.2.2. 设立专门的协调员

设置专用的协调器有助于提高大型集群或大型 Impala 工作负载的性能。

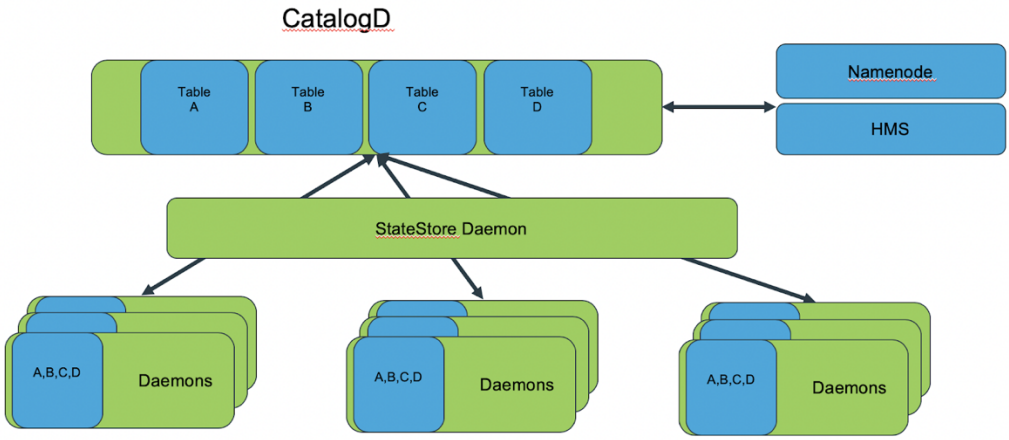
Impala 可以构建执行计划并快速执行查询，因为它会在 Catalog Server 和查询协调器上缓存 HDFS 表的块和文件元数据。当查询新表时，协调器节点从目录服务守护程序（`catalogd`）请求元数据，该服务随后又与 Hive Metastore（HMS）和 Namenode 进行对话以检索信息：



6.2.2.1. *典型协调器设置的问题：*

在大型集群上或对于较大的 Impala 工作负载，由于以下原因，典型的协调器设置可能会出现问题：

- 每个协调器都缓存所有表分区和数据文件的元数据。这需要为协调器配置较大的 JVM 堆大小。
- 由于包含大量查询片段的查询的网络和 CPU 开销，协调器所需的额外工作会干扰其执行查询执行工作的能力。
- 由大量主机充当协调者可能会导致不必要的网络开销，因为这些主机中的每一个都从 Statestore 守护程序（`statestored`）接收元数据以进行元数据更新>：

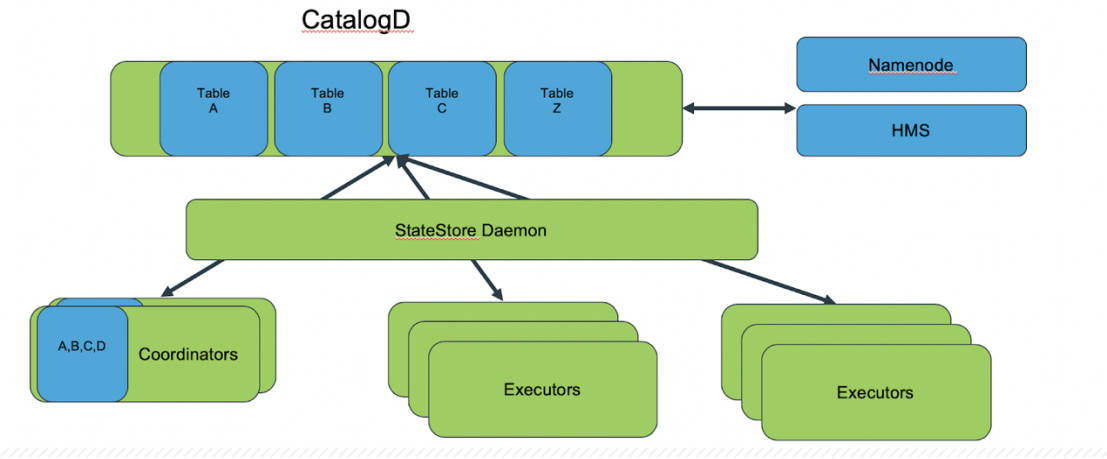


- 根据 Hive Metastore 的大小以及要查询的表的分区、文件和块，内存目录可能会变得非常大。这可能会对 Impala 守护程序的内存造成压力。

- 当有大量的重载主机充当协调器时，很可能会超过准入控制功能所施加的软限制，这会导致超出准入控制限制。

6.2.2.2. 专用协调器设置的解决方案：

要解决此协调器设置问题，请为协调器和执行者分配单独的角色，如下所示：



6.2.2.3. 专用协调器设置如何工作

- 在边缘节点和网关节点上运行协调器角色。在数据节点上运行执行程序。
- 仅协调器节点缓存元数据。他们执行查询协调的任务以及其他一些任务，例如查询的最终排序。执行者角色从 HDFS、联接表和一些相关任务中读取。
- 使用此配置，执行程序守护程序无法通过 Impala Shell 或 JDBC 连接进行连接。
- 配置具有足够大的 JVM 堆大小的协调程序守护程序，以将元数据保存在内存中。配置对于仅被分配了专用角色作为执行者的 Impala 守护程序，请配置具有默认 JVM 堆大小的守护程序。

6.2.2.4. 设立专门的协调员的好处

- 减少执行程序节点上的内存使用量。
- 避免了由查询执行任务引起的协调器 CPU 瓶颈。
- 降低超过准入控制限制的可能性。
- 通过限制仅向协调器节点广播的元数据来减少 Statestore 守护程序上的网络利用率。

- 通过减轻协调员的工作量压力来提高可靠性和性能。

6.2.2.4.1. *协调员的负载平衡*

Cloudera 建议为与 Impala 的传入连接设置负载平衡器。

HaProxy 或 F5 之类的负载均衡器 通常可以很好地负载均衡传入的 JDBC 和 Impala shell 连接。还应配置负载均衡器主机名，以使 Hue 连接到 Impala。

负载均衡器配置的经验法则如下：

- 始终将负载均衡器配置为使用使用源 IP 地址的 *balance source*，以确保始终将来自一个客户端的连接路由到同一协调器节点。
- 负载均衡器超时应设置为大于 Impala 空闲会话超时的值。例如，如果您使用的是 HaProxy，并使用设置了 5 分钟的超时 `timeout server 300s`，请确保将 Impala 的空闲超时设置为小于 300 秒。

6.2.3. 按需元数据和元数据管理

按需元数据功能简化了协调器上的元数据处理。

在按需元数据功能发布之前，每个协调员都会在本地保留所有元数据的副本。这消耗了每个协调器上的大量内存，无法退出它。尽管最终使用了元数据，但状态存储仍将元数据发送给所有协调员。

按需元数据功能包括以下改进：

- 协调员 `catalogd` 在需要进行查询计划时需要从目录守护程序中为所需表提取元数据，并在本地对其进行缓存。Statestore 不再将所有表的所有元数据发送给每个协调器。
- 在内存压力下，缓存的元数据会自动清除，将缓存视为 LRU（最近最少使用）的缓冲区。
- 按需元数据获取的粒度是在分区级别上，因此添加/删除分区之类的常见用例不会触发不必要的元数据传输。

6.2.3.1. *启用按需元数据获取*

使用 Cloudera Manager 为 Impala 启用按需元数据获取。

6.2.3.1.1. 要在 Cloudera Manager 中启用按需元数据获取：

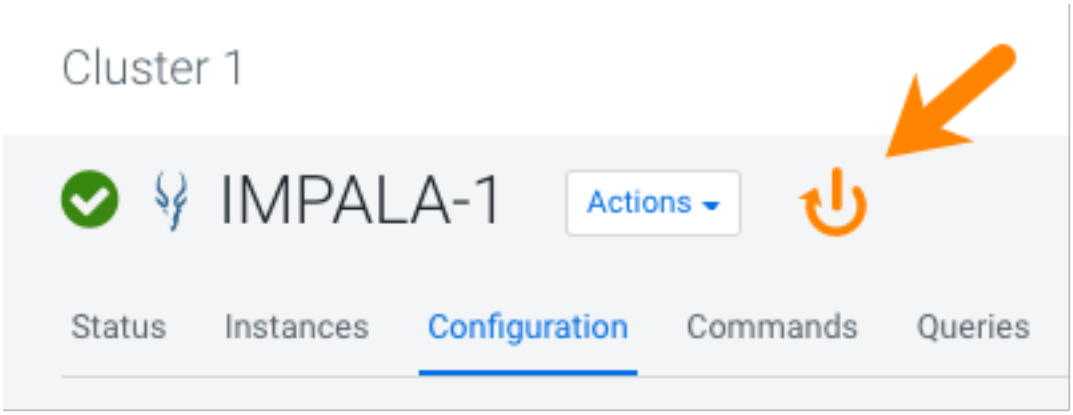
- 在 Cloudera Manager 管理控制台中，单击 IMPALA。
- 单击配置选项卡。
- 搜寻 `catalogd`。
- 向下滚动以找到目录服务器命令行参数高级配置代码段（安全阀），然后在文本框中输入以下配置信息：

```
--catalog_topic_mode=minimal
```

1) 搜寻 `impalad`。
2) 向下滚动以找到 Impala Daemon 命令行参数高级配置代码段（安全阀），然后在文本框中输入以下配置信息：

```
--use_local_catalog=true
```

3) 单击页面右下角的“保存更改”。
4) 向上滚动到页面顶部，然后单击刷新图标以重新启动服务，以便您的配置更改可以生效：



6.2.3.2. 启用过时的元数据发布

使用 Cloudera Manager 启用 Impala 中的过时元数据发布。

1) 在 Cloudera Manager 管理控制台中，单击 IMPALA。
2) 单击配置选项卡。
3) 搜寻 `catalogd`。
4) 向下滚动以找到目录服务器命令行参数高级配置代码段（安全阀），然后在文本框中输入以下配置信息：

```
--invalidate_tables_timeout_s=28800
--invalidate_tables_on_memory_pressure=true
```

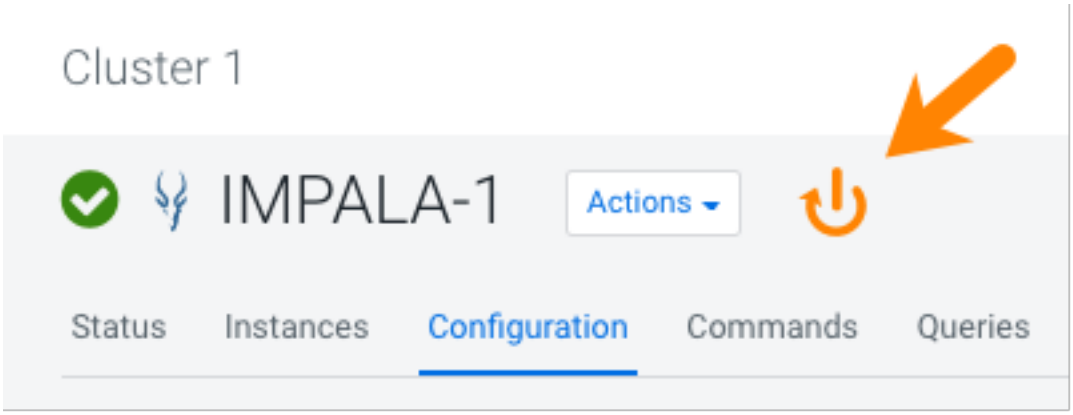
上面的方法将 Catalog 守护程序设置为在 28,800 秒或 8 个小时内未访问（未使用）元数据后释放元数据。

1) 搜寻 `impalad`。
2) 向下滚动以找到 Impala Daemon 命令行参数高级配置代码段（安全阀），然后在文本框中输入以下配置信息：

```
--invalidate_tables_timeout_s=28800
--invalidate_tables_on_memory_pressure=true
```

上面的命令将 Impala 守护程序设置为在 28,800 秒或 8 个小时内未访问（未使用）元数据后释放元数据。

3) 单击页面右下角的“保存更改”。
4) 向上滚动到页面顶部，然后单击刷新图标以重新启动服务，以便您的配置更改可以生效：



6.2.3.3. 避免小文件

为了减少目录用于元数据的内存量，请避免在 HDFS 中创建许多小文件。

HDFS 中的小文件可能是由于分区粒度太小或过于频繁地执行数据摄取而引起的。Cloudera 建议您定期压缩小文件。在 Hive 中，可以使用以下 SQL 命令压缩小型文件：

```
SET hive.merge.mapfiles = true;
SET hive.merge.mapredfiles = true;
SET hive.merge.size.per.task = 256000000;
SET hive.merge.smallfiles.avgsize = 134217728;
SET hive.exec.compress.output = true;
SET parquet.compression = snappy;
```

```
INSERT OVERWRITE TABLE db_name.table_name SELECT * FROM
db_name.table_name;
Run <Refresh Table> in impala after the Hive job finishes.
```

对于具有多个分区的表，请将分区策略更改为以更细粒度的方式进行分区。例如，用 year/month 代替 year/month/day。如果要使用 Impala 进行插入，则 use /* +SHUFFLE */ 优化程序会提示优化程序，这些提示会在写入数据之前添加一个交换节点。使用此提示，一次仅一个节点写入一个分区，从而减少了写入的文件数量。

6.2.3.4. 自动元数据管理

Impala 自动元数据管理功能使用通知在更改后刷新数据。

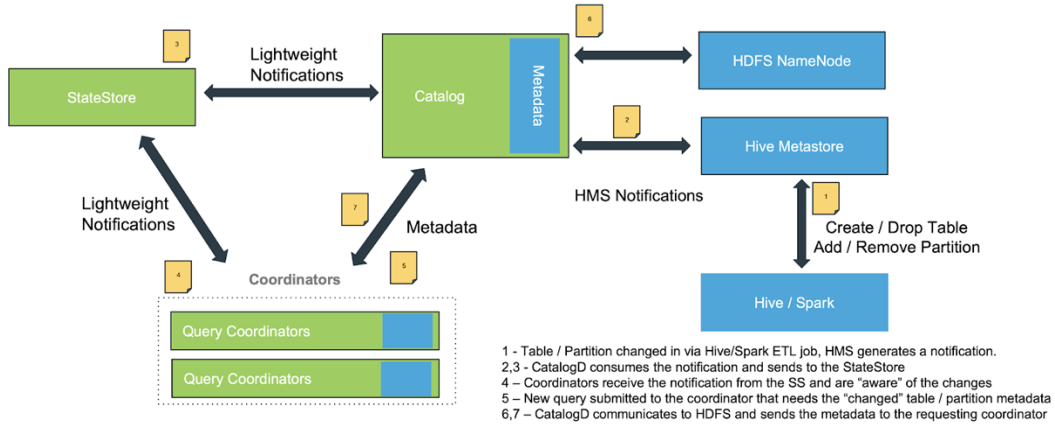
当使用诸如 Hive 和 Spark 之类的工具来处理提取到 Hive 表中的原始数据时，新的 Hive Metastore 元数据（以数据库，表和分区的形式）和文件系统元数据（以现有分区和表中的新文件的形式））生成。在以前的 Impala 版本中，要获取此新信息，Impala 用户必须手动发出 INVALIDATEor 或 REFRESH 命令。现在，有一个新功能称为自动元数据管理。自动元数据管理通过使用执行以下操作的 Hive Metastore 通知来工作：

- 收到 ALTER TABLE 事件时使表无效。
- 收到 ALTER TABLE ADD | DROP PARTITION 事件时刷新分区。
- 收到 CREATE TABLE 或 CREATE DATABASE 事件时添加表或数据库。
- catalogd 当接收到 DROP TABLE 或 DROP DATABASE 事件时，从目录（）中删除表。
- 接收 INSERT 事件时刷新表和分区。

在数据库级更改时，支持以下类型的更改：

- 数据库属性
- 对数据库的评论
- 数据库所有者
- 数据库的默认位置

图 1. Hive Metastore 通知



要控制此功能，请使用上的`--hms_event_polling_interval_s` 标志 `catalogd`。设置为正值时，它将启用 Hive Metastore 事件轮询。将此标志设置为的建议值在 5 秒以内。

自动元数据管理功能不支持通过使用 Spark、DistCp 或 HDFS Put 手动添加到 HDFS 的文件或分区，因为没有为这些操作生成 Hive Metastore 通知。要处理这些情况，请使用以下方法之一：

- 使用 `LOAD DATA` 命令来处理元数据更改，或者
- 运行一个 `REFRESH [db_name.]table_name [PARTITION…]`或多个 `ALTER TABLE table_name RECOVER PARTITIONS` 命令。

如果需要禁用某些表或数据库的自动元数据管理，请设置以下属性：

```
CREATE DATABASE <db_name> DBPROPERTIES
('impala.disableHmsSync'='true');
CREATE TABLE <tab_name> WITHTBLPROPERTIES
('impala.disableHmsSync'='true' | 'false');
```

6.2.3.5. Spark 中的编码，用于自动元数据管理

用于将数据保存到指定位置的 Spark API 不会在 Hive 元存储中生成事件，因此自动元数据管理不支持该事件。

例如，以下是 Spark Scala API 代码的示例，该代码不会在 Hive Metastore 中生成事件：

```
Seq((1, 2)).toDF("i",
"j").write.save("/user/hive/warehouse/spark_etl.db/customers/date=0101
2019")
```

而是，使用下面的 Spark SQL 代码来确保生成 Hive Metastore 事件并将其发送到 Metastore：

```
Spark.sql(" INSERT OVERWRITE TABLE xxx  PARTITION (date = , …) as
select * from spark_dataframe”)
```

6.2.3.6. 手动元数据管理

如果未使用 CDH 6.2 或 5.16，则必须为在 Hive on Spark 或第三方 ETL 工具中运行的 ETL 操作手动管理元数据。

对于手动元数据管理，Cloudera 建议将元数据操作作为 ETL 代码的一部分进行处理。这样可确保 BI 最终用户在添加新数据时无需刷新表。在 Spark 上通过 Hive 加载数据后，使用 Spark 中的 Impala shell 或 Impala JDBC 连接器连接到 Impala 并执行以下操作：

- **REFRESH <*table_name*>** 或者 **REFRESH <*table_name*> PARTITION**
  - 重新加载表的元数据，并从 HDFS NameNode 增量地重新加载文件和块元数据。
  - 使用 Hive on Spark 将文件添加，删除或覆盖到现有分区中时应运行。
- **ALTER TABLE <*table_name*> RECOVER PARTITIONS**
  - 扫描 HDFS 以检查是否添加了任何新的分区目录，并为这些文件缓存块元数据。
  - 新分区添加到现有表时应运行。请注意，它比使用更快。REFRESH <*table_name*>
- **INVALIDATE METADATA <*table_name*>**
  - 异步运行以丢弃表的已加载元数据目录高速缓存。表的元数据加载由任何后续查询触发。
  - 运行 INVALIDATE METADATA 在创建或删除外部工具，如蜂房的 Spark 或运行 HDFS 平衡器后的新表时。

6.2.3.7. 准入控制

准入控制功能控制集群上运行的并发查询的数量，从而避免了繁忙集群上的内存不足问题。

一旦达到您在准入控制中设置的限制，随后的查询就会排队，并等待运行中的查询完成后再运行。您可以配置查询可以等待的时间。使用准入控制，可以设置池中每个查询可

以使用的默认内存。如果未设置默认内存限制，则 Impala 将基于表和列统计信息使用启发式方法来确定每个查询所需的内存量。

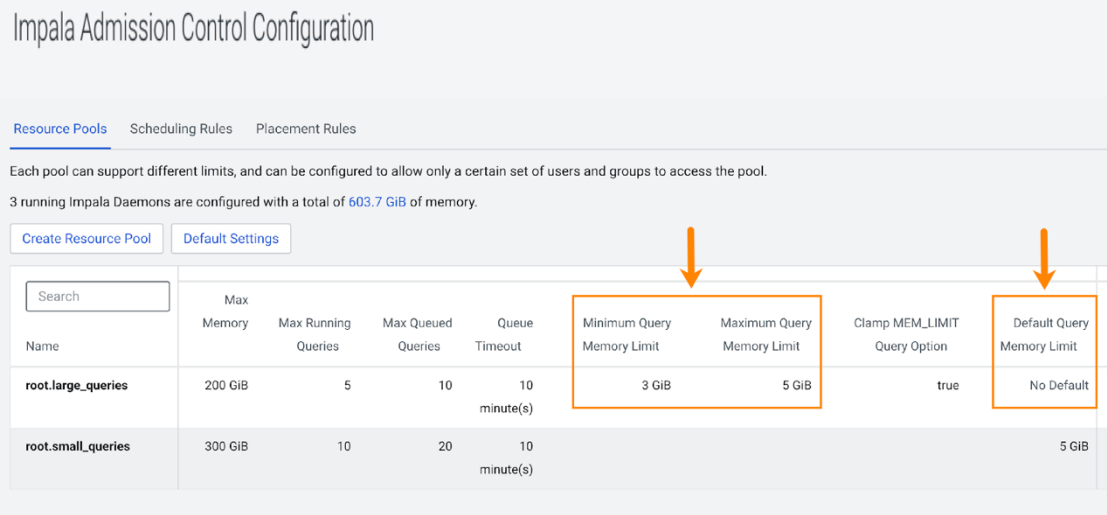
Cloudera 建议您将并发运行的查询数设置为 40-60。这大致对应于每个 Impala 守护程序上的线程数。

使用准入控制的最佳实践是：

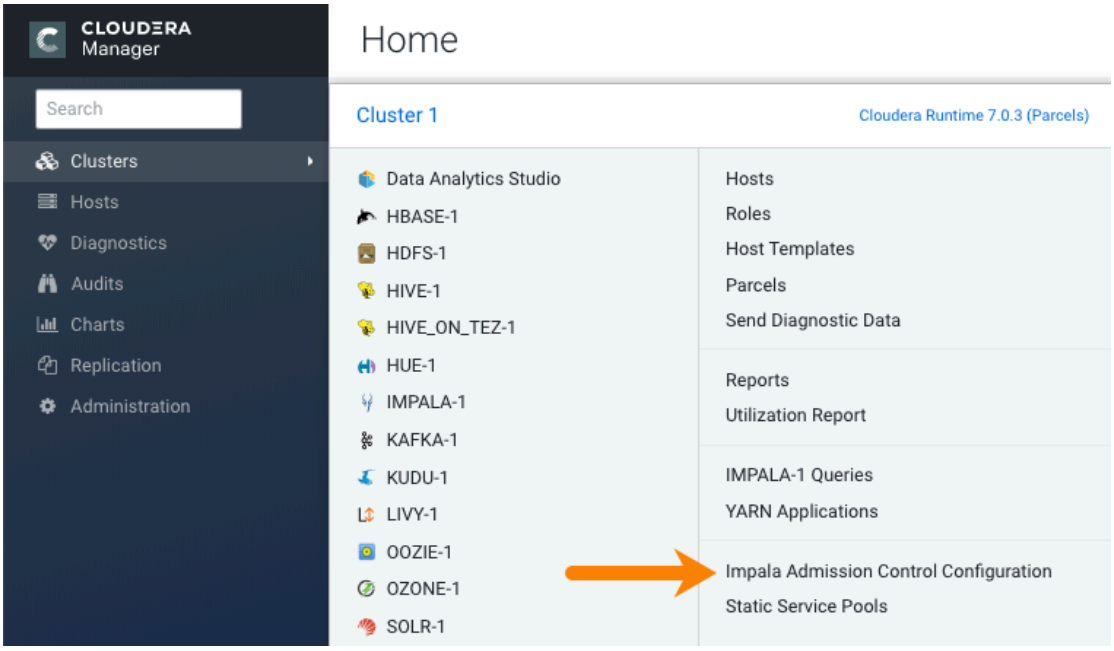
- 通过在 Cloudera Manager 中检查以下设置来确保启用准入控制：

Enable Impala Admission Control ☑ IMPALA-1 (Service-Wide)

- 确保为所有池设置了“最小查询内存限制”和“最大查询内存限制”或“ 默认查询内存限制”：



您可以在 Cloudera Manager 中检查这些设置。在 Cloudera Manager 主页中，单击左侧导航树中的 Clusters（集群），然后选择 Impala Admission Control Configuration（Impala 准入控制配置）：



在 CDH 6.2 中，准入控制设置也已添加到 Impala Web UI。有关详细信息，请参阅产品文档中的 Impala Web UI 故障排除。

- 如果设置了默认内存限制，查询配置文件将显示以下消息：

```
Query Options (set by configuration): MEM_LIMIT=2147483648,REQUEST_POOL=RPIBR
Query Options (set by configuration and planner): MEM_LIMIT=2147483648,REQUEST_POOL=RPIBR,MT_DOP=0
```

- 如果默认内存限制设置得太低并且查询超时，并且您收到“内存限制超出”错误消息，请使用该 SET MEM_LIMIT 选项重新运行查询。例如：

```
SET MEM_LIMIT = 3gb;
```

有关此 SET 语句的详细信息，请参见 MEM_LIMIT 查询选项。

- 所有池中设置的总最大内存应等于分配给 Impala 服务的总内存。

6.2.3.7.1. 预估内存限制

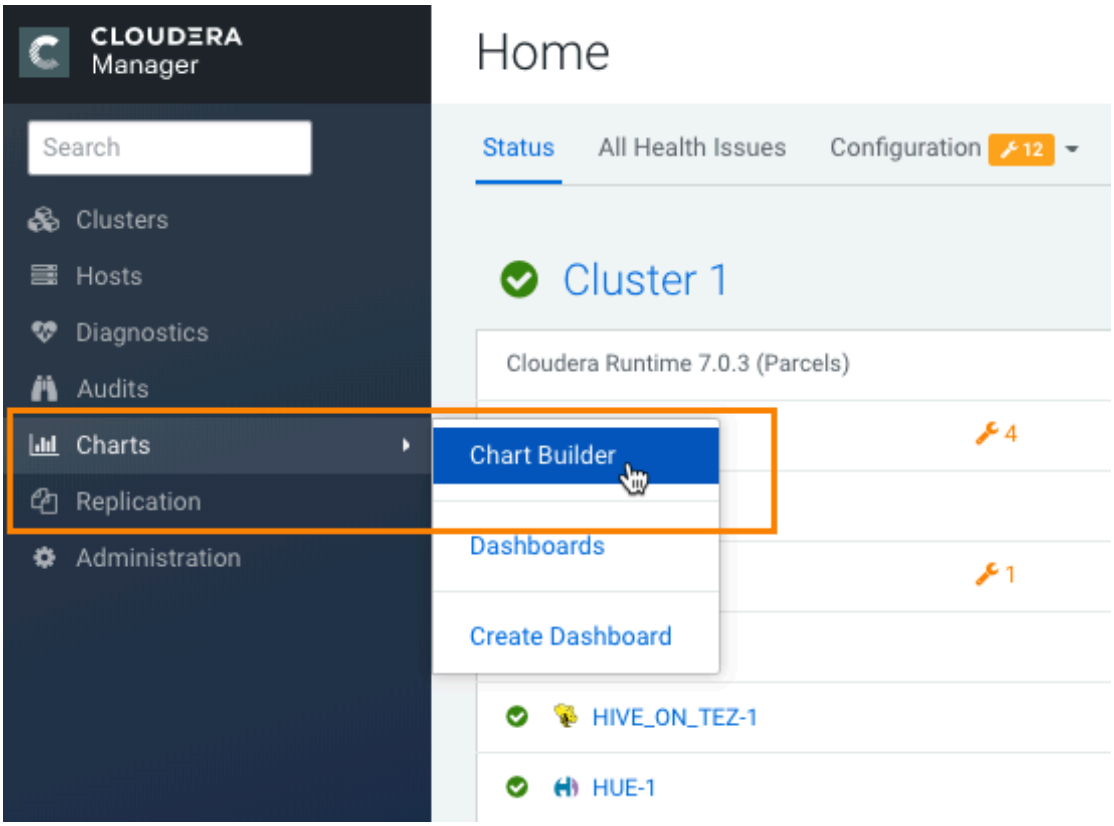
使用 Cloudera Manager Chart Builder 确定资源池的内存限制。

为了确定每个池的最小，最大和默认内存限制是什么，Cloudera 建议您对数据集运行真实查询，然后在此测试的高峰时段查看内存使用情况。

在数据集上运行一组实际查询之后，请使用 Cloudera Manager 图表生成器查看反映内存使用情况的图表：

1) 在 Cloudera Manager 管理控制台主页上，从左侧导航树中，选择

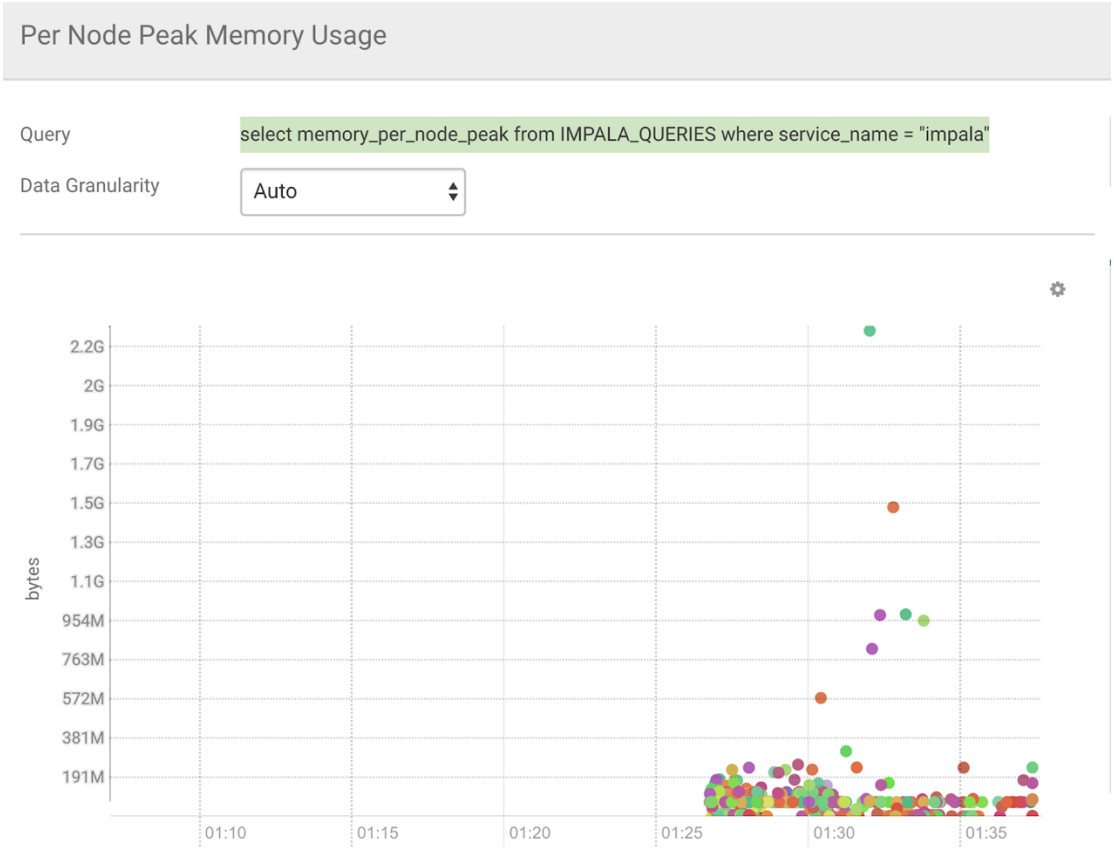
Charts > Chart Builder：



2) 在“图表构建器”页面上，输入以下查询，然后单击“构建图表”：

```
select memory_per_node_peak from IMPALA_QUERIES where
service_name="impala"
```

然后，“图表生成器”将返回一个看起来类似于以下示例的图表：

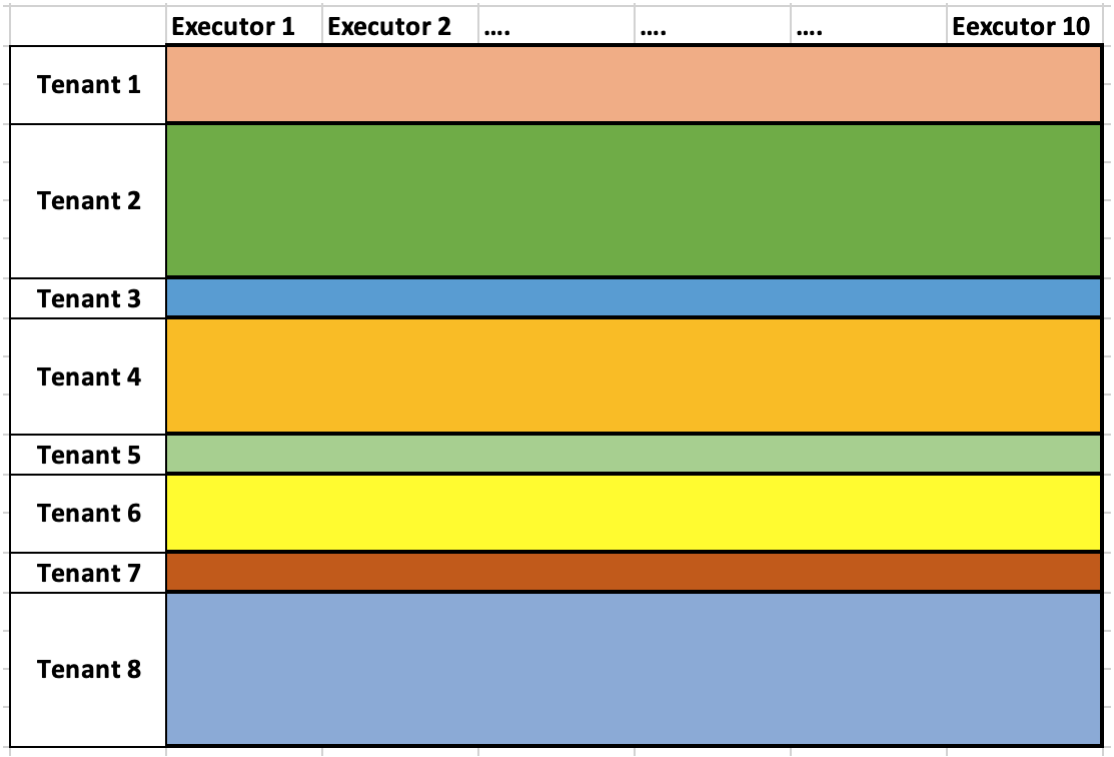


这种类型的图表可用于确定特定池的内存限制。请注意，大多数查询每个节点使用的内存少于 1.2 GB。

6.2.3.8. 资源池设计

跨资源池分配内存是一项重要的性能调整任务。

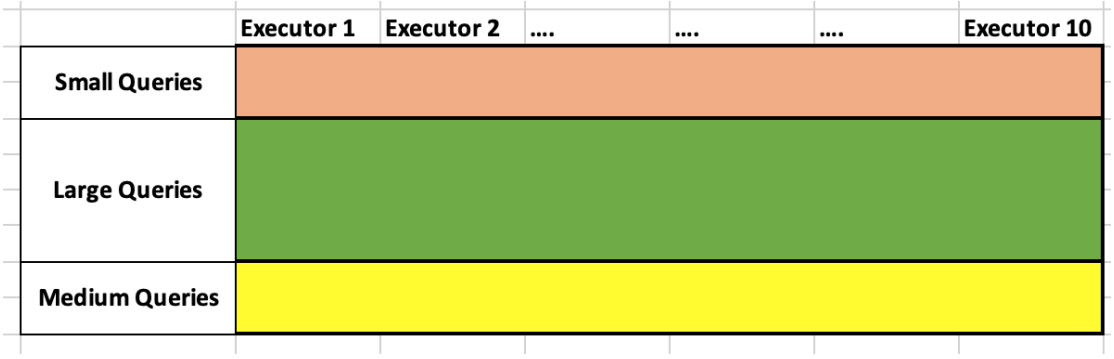
启用准入控制后，用于在所有池之间分配 Impala 内存总量的最常见方法是为每个租户创建一个池，如下图所示：



但是，由于以下原因，此策略通常适得其反：

- 一个租户中未使用的内存不能被其他租户使用。
- 繁忙的租户在准入控制中将查询排队，从而导致查询执行的整体速度变慢。
- 运行大型查询的小型租户将溢出到磁盘，直到达到溢出到磁盘的限制并达到内存限制。

相反，Cloudera 建议您按查询大小设计资源池，如下图所示：



此方法可确保分配给 Impala 的所有内存得到最佳使用。此外，通过使用此方法，所有用户和组都可以访问所有池，并且用户可以根据内存大小选择适当的池。

6.2.3.9. 表和列的统计信息

表和列统计信息可帮助 Impala 使用表大小和列的基数程度来生成最佳查询计划。

统计信息存储在 Hive Metastore 数据库中。您运行该 COMPUTE STATS 语句以收集和设置特定表的表级和分区级行数以及列统计信息。但是，运行此语句会占用大量 CPU。根据行数，数据文件数，数据文件的总大小以及所涉及的文件格式，Cloudera 建议使用以下策略来计算统计信息：

- 可以使用以下 SHOW 命令来标识缺少的统计信息：

```
SHOW TABLE STATS [database_name.]table_name
SHOW COLUMN STATS [database_name.]table_name
```

- 表和列的统计信息也记录在查询概要文件中，如下所示：

```
F00:PLAN FRAGMENT [RANDOM] hosts=5 instances=5
00:SCAN HDFS [telco_db.call_trace_hist_bucketed, RANDOM]
   partitions=284/3396 files=916 size=206.74GB
   predicates: (calling_number = '734214' OR called_number = '734214')
   table stats: unavailable
   columns missing stats: date_time, msisdn, calling_number, called_number, duration, calling_location, called_location
   mem-estimate=616.00MB mem-reservation=0B
   tuple-ids=0 row-size=99B cardinality=391672994
```

- 不要为未在联接或 Impala 查询中使用的表计算统计信息。例如，在没有连接的情况下一对一地馈入数据仓库层表的登台表上的统计信息就不需要统计信息。
- 将运行限制为 COMPUTE STATS 仅包含在过滤器，联接条件或 GROUP BY 子句中的列。例如，COMPUTE STATS 在以下情况下运行很有意义：

```
COMPUTE STATS wide_table [ join_column_a, join_column_b ]
```

- COMPUTE STATS 仅当数据中的数据更改超过 30％时才重新运行。
- Cloudera 建议您 COMPUTE STATS 在非高峰时间，周末或晚上运行。
- 当您重新加载一个完整的表或分区时，其中每列的行数和不同值与以前相对不变，因此无需重新运行 COMPUTE STATS。

6.2.3.9.1. 手动设置统计信息

对于统计不变或已知的表和分区，可以手动设置统计。

手动设置统计信息，如下所示：

- 设置表中的总行数：

```
ALTER TABLE <table_name> SET TBLPROPERTIES('numRows'='new_value',
'STATS_GENERATED_VIA_STATS_TASK'='true');
```

- 设置特定分区的总行数：

```
ALTER TABLE <table_name> PARTITION (keycol1=value_1,keycol2=value_2...)
 SET TBLPROPERTIES('numRows'='new_value',
'STATS_GENERATED_VIA_STATS_TASK'='true');
```

- 设置列统计信息：

```
ALTER TABLE <table_name> SET COLUMN STATS <col_name> ('numDVs'='100');
```

numDVs 通过运行以下查询来计算值：

```
SELECT NDV(col_name) FROM <table_name>;
```

6.2.3.10. 调优 SQL 查询

虽然 SQL 查询调优是一个广泛的主题，但 Cloudera 建议使用以下 SQL 最佳实践。

- 与第三范式（3NF）模型相比，Hadoop 和 Impala 最适合于星型模式数据模型。虽然 Impala 可以有效地与 3NF 模型一起使用，但星型模式模型中使用的联接数量较少和表较大则通常对应于更快的查询执行时间。
- 使用整数连接键，而不是字符或数据连接键。
- 始终使用 varchar 数据类型而不是 character 数据类型的列。
- 避免使用隐式或显式强制转换。

6.2.3.11. 推荐的 Impala SET 选项

以下是建议的 SET 选项，以使 Impala 达到最佳性能。

- 始终使用 `SET MEM_LIMIT` 查询选项。有关详细信息，请参见 MEM_LIMIT 查询选项。
- 使用 `APPX_COUNT_DISTINCT` 查询选项可以 `COUNT(DISTINCT)` 通过将查询转换为 `NDV()` 通过线性计数使用函数调用来提高查询性能。请注意，`APPX_COUNT_DISTINCT` 对于较小的数据集，精度约为 98%，对于较大的数据集，精度可达 99% 以上。有关详细信息，请参见 APPX_COUNT_DISTINCT 查询选项。
- 始终使用同时编码和解码的压缩算法（编解码器）。Cloudera 建议您将 LZ4 或 ZSTD 与 Impala 3.3 版及更高版本一起使用。对于较早的 Impala 版本，请使用 snappy。
- 不要使用 `DISABLE_CODEGEN` 查询选项，因为它会大大降低查询的执行速度。而是使用 `DISABLE_CODEGEN_ROWS_THRESHOLD` 查询选项，该选项将自动为小型查询禁用代码生成。有关详细信息，请参见 DISABLE_CODEGEN 查询选项。
- 将 `EXEC_SINGLE_NODE_ROWS_THRESHOLD` 查询选项设置为 `1000` 行，以更快地执行简单查询或包含 `LIMIT` 子句的查询。有关详细信息，请参见 EXEC_SINGLE_NODE_ROWS_THRESHOLD 查询选项。
- `NUM_ROWS_PRODUCED_LIMIT` 为交互式用户池或在 Hue 中设置查询选项，以限制交互式查询产生的行。有关详细信息，请参见 NUM_ROWS_PRODUCED_LIMIT 查询选项。
- 不要使用 `SYNC_DDL = true` 查询选项。相反，请使用负载平衡器配置来确保始终将客户端定向到相同的查询协调器。
- 将 `EXEC_TIME_LIMIT_S` 查询选项设置为一个值，例如 1200 秒（20 分钟）。如果在时间限制到期时仍在执行查询，则此查询选项将取消正在执行的查询。如果查询

被取消，请调查查询计划以找出运行缓慢的原因。

6.2.3.12. 推荐配置

以下是为使 Impala 达到最佳性能而推荐的配置设置。

- 将 `--use_local_tz_for_unix_timestamp_conversions` 启动标志和`--convert_legacy_hive_parquet_utc_timestamps` 启动标志都设置为 true。将这些启动标志设置为 true 可确保 Hive 和 Impala 之间的时间戳匹配。有关更多详细信息，请参见 TIMESTAMP 数据类型。
- 始终为 Impala 守护程序（impalad）设置`--idle_session_timeout` 和 `--idle_query_timeout` 超时。确保的设置 `idle_session_timout` 小于为负载均衡器设置的超时设置。有关详细信息，请参见 为 impalad 设置空闲查询和空闲会话超时。
- 将`--fe_service_threads`Impala 守护程序（impalad）的启动选项设置为 256。此选项指定允许的最大并发客户端连接数。有关 详细信息，请参见 impalad 守护程序的启动选项。
- 将`--num_metadata_loading_threads` 启动选项增加到 64，以提高元数据加载性能。

6.2.3.13. 结合使用 Impala 和 Hue

以下是一些推荐的配置，当您将 Hue 与 Impala 结合使用时，可以为您带来最佳性能。

- 始终连接到 Impala 负载平衡器（例如 HaProxy），而不要连接到单个协调器。这有助于避免在 单个协调器上出现*热点*问题。当太多任务接触同一数据时，就会发生*热点*，从而可能使节点或协调器超载。
- 根据需要，将 `querycache_rows` 配置属性设置为低于其默认设置的值 `50000`。此配置属性设置 Impala 缓存以支持重新获取它们的结果集的初始行数。
- 将 `close_queries` 配置属性设置为 true。当用户离开编辑器页面时，此属性使 Hue 尝试关闭 Impala 查询，以便释放所有 Impala 查询资源。但是，它使该用户的查询结果不可访问。
- 设置 `query_timeout_s` 和 `session_timeout_s` 配置属性。您可以使用这些属性来设置查询执行和 Hue 中会话的超时，这将导致超时期限到期时取消查询和会话。它们以秒为单位设置。

6.2.3.13.1. 要在 Cloudera Manager 中设置这些配置属性：

1) 在 Cloudera Manager 主页上，选择 Hue >配置。

2) 搜索顺化服务高级配置片段（安全阀）为 hue_safety_valve.ini，然后追加下面的配置信息到文本框中任何其他配置后上市的有：

```
[impala]
querycache_rows=<number_of_rows>
close_queries=true
query_time_s=<number_of_seconds>
session_timeout_s=<number_of_seconds>
```

3) 单击“保存更改”，然后重新启动服务。

6.2.3.14. 将 Impala 与 BI 工具一起使用

以下是将 BI 工具与 Impala 结合使用的建议，可为您带来最佳性能。

- 始终为 BI 工具启用 LDAP 身份验证以方便地连接到 Impala，而不是使用 Kerberos 身份验证。
- 使用 BI 工具上的缓存将来自同一查询的结果提供给多个用户。

6.2.3.15. 适当的文件格式

以下是针对哪些文件格式在 Impala 中提供最佳性能的建议。

- 对于 BI 查询，由于 Parquet 文件格式结合了列存储布局，压缩和编码，因此性能最佳。的默认设置 `COMPRESSION_CODEC` 是活泼的 压缩，但 GZip 压缩 ，还支持压缩。
- Impala 还支持从 2.12 版本开始读取 ORC 文件格式，但是期望 ORC 表的查询性能比 Parquet 表慢。
- 从表中检索所有列时，可以使用文本格式。但是，由于对文本的压缩较低，因此与使用 Parquet 文件格式时相比，HDFS I / O 可能更长。

6.2.3.16. 分区粒度建议

以下是表分区粒度的建议，该粒度在 Impala 中提供最佳性能。

- 选择一种分区策略，以确保每个分区中至少有 256 MB 的数据。
- 过度分区会导致查询计划花费的时间比必要时间长，因为 Impala 会修剪不必要的分区，这会导致每个分区中的文件较小。
- Cloudera 建议您将表中的分区数保持在 30,000 以下。
- 始终 `integer` 对分区键列使用数据类型：
    - 分区键值都变成了 HDFS 目录的名称，以便您可以通过使用数值，共同分

区键领域，如减少内存使用 YEAR，MONTH 和 DAY。

- 使用 integer 可容纳适当范围值的最小数据类型。通常，TINYINT 用于 MONTH 和 DAY，以及 SMALLINT 用于 YEAR。使用该 EXTRACT() 函数可以从 TIMESTAMP 值中提取各个日期和时间字段，并将 CAST() 返回值转换为适当的 integer 数据类型。

6.2.3.17. *解决热点*

遵循以下建议可以解决“热点”问题，即当许多查询访问同一节点并导致其过载时会发生这种情况。

当特定查询或整个 Impala 工作负载的数据集中在有限数量的节点上时，由于扫描碎片，这些节点可能会“成为热点”。例如，如果您有一个适合单个 HDFS 块的小尺寸表，则多个查询会在同一节点上运行扫描片段，从而导致其过载。

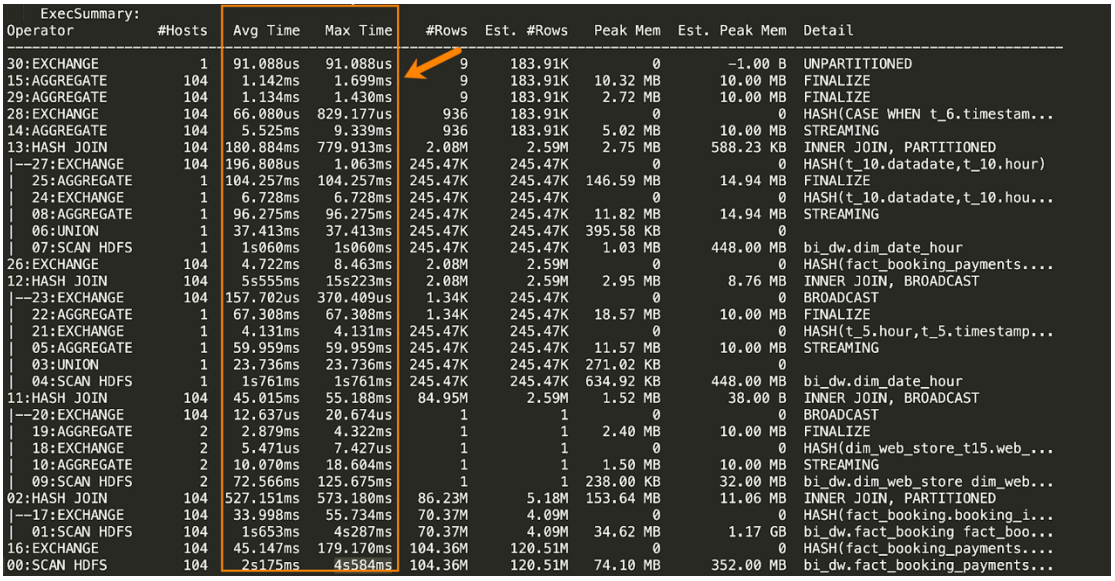
为了避免这种热点：

- 对于小尺寸表，请使用 HDFS File System Shell 命令 hdfs dfs -setrep 选项增加 HDFS 复制因子。有关 setrep 更多信息，请参见命令参考。
- 对于查询严重的事实表分区，可以临时设置复制因子。例如，您可以使用上述 setrep 命令将其设置为 7 天。然后，经过 7 天后，您可以将复制因子重置为 3。
- 可以通过设置以下查询选项来更改 Impala 计划的行为：
  - SET SCHEDULE_RANDOM_REPLICA=true 有关 详细信息，请参见 SCHEDULE_RANDOM_REPLICA 查询选项。
  - SET REPLICA_PREFERENCE=REMOTE 有关 详细信息，请参见 REPLICA_PREFERENCE 查询选项。
- HDFS 缓存可用于缓存块副本。这会使 Impala 调度程序随机选择一个托管用于扫描的缓存块副本的节点。

6.2.3.17.1. *检测块偏斜*

以下是检测块偏斜的建议，这些偏斜是影响 Impala 性能的较慢的主机。

检测块偏斜或“主机缓慢”问题最方便的方法 是在查询计划的“执行摘要”部分中将“平均时间”与“最大时间”进行比较：



```
    ExecSummary:
Operator            #Hosts   Avg Time   Max Time    #Rows  Est. #Rows   Peak Mem  Est. Peak Mem  Detail
------------------------------------------------------------------------------------------------------------------
30:EXCHANGE              1   91.088us   91.088us        9     183.91K          0       -1.00 B  UNPARTITIONED
15:AGGREGATE           104    1.142ms    1.699ms        9     183.91K   10.32 MB       10.00 MB  FINALIZE
29:AGGREGATE           104    1.134ms    1.430ms        9     183.91K    2.72 MB       10.00 MB  FINALIZE
28:EXCHANGE            104   66.080us  829.177us      936     183.91K          0              0  HASH(CASE WHEN t_6.timestam...
14:AGGREGATE           104    5.525ms    9.339ms      936     183.91K    5.02 MB       10.00 MB  STREAMING
13:HASH JOIN           104  180.884ms  779.913ms    2.08M       2.59M    2.75 MB      588.23 KB  INNER JOIN, PARTITIONED
|--27:EXCHANGE         104  196.808us    1.063ms  245.47K     245.47K          0              0  HASH(t_10.datadate,t_10.hour)
|  25:AGGREGATE          1  104.257ms  104.257ms  245.47K     245.47K  146.59 MB       14.94 MB  FINALIZE
|  24:EXCHANGE           1    6.728ms    6.728ms  245.47K     245.47K          0              0  HASH(t_10.datadate,t_10.hou...
|  08:AGGREGATE          1   96.275ms   96.275ms  245.47K     245.47K   11.82 MB       14.94 MB  STREAMING
|  06:UNION              1   37.413ms   37.413ms  245.47K     245.47K  395.58 KB              0
|  07:SCAN HDFS          1    1s060ms    1s060ms  245.47K     245.47K    1.03 MB      448.00 MB  bi_dw.dim_date_hour
26:EXCHANGE            104    4.722ms    8.463ms    2.08M       2.59M          0              0  HASH(fact_booking_payments....
12:HASH JOIN           104    5s555ms   15s223ms    2.08M       2.59M    2.95 MB       8.76 MB  INNER JOIN, BROADCAST
|--23:EXCHANGE         104  157.702us  370.409us    1.34K     245.47K          0              0  BROADCAST
|  22:AGGREGATE          1   67.308ms   67.308ms    1.34K     245.47K   18.57 MB       10.00 MB  FINALIZE
|  21:EXCHANGE           1    4.131ms    4.131ms  245.47K     245.47K          0              0  HASH(t_5.hour,t_5.timestamp...
|  05:AGGREGATE          1   59.959ms   59.959ms  245.47K     245.47K   11.57 MB       10.00 MB  STREAMING
|  03:UNION              1   23.736ms   23.736ms  245.47K     245.47K  271.02 KB              0
|  04:SCAN HDFS          1    1s761ms    1s761ms  245.47K     245.47K  634.92 KB      448.00 MB  bi_dw.dim_date_hour
11:HASH JOIN           104   45.015ms   55.188ms   84.95M       2.59M    1.52 MB       38.00 B  INNER JOIN, BROADCAST
|--20:EXCHANGE         104   12.637us   20.674us        1           1          0              0  BROADCAST
|  19:AGGREGATE          2    2.879ms    4.322ms        1           1    2.40 MB       10.00 MB  FINALIZE
|  18:EXCHANGE           2    5.471us    7.427us        1           1          0              0  HASH(dim_web_store_t15.web_...
|  10:AGGREGATE          2   10.070ms   18.604ms        1           1    1.50 MB       10.00 MB  STREAMING
|  09:SCAN HDFS          2   72.566ms  125.675ms        1           1  238.00 KB       32.00 MB  bi_dw.dim_web_store dim_web...
02:HASH JOIN           104  527.151ms  573.180ms   86.23M       5.18M  153.64 MB       11.06 MB  INNER JOIN, PARTITIONED
|--17:EXCHANGE         104   33.998ms   55.734ms   70.37M       4.09M          0              0  HASH(fact_booking.booking_i...
|  01:SCAN HDFS        104    1s653ms    4s287ms   70.37M       4.09M   34.62 MB        1.17 GB  bi_dw.fact_booking fact_boo...
16:EXCHANGE            104   45.147ms  179.170ms  104.36M     120.51M          0              0  HASH(fact_booking_payments....
00:SCAN HDFS           104    2s175ms    4s584ms  104.36M     120.51M   74.10 MB      352.00 MB  bi_dw.fact_booking_payments...
```

对于查询的每个阶段，都有一个“平均时间” 和“最大时间”值以及#Hosts，它指示查询的该阶段涉及多少个主机。对于涉及一个以上主机的所有查询阶段，请查找最大时间显着大于平均时间的情况。

主机缓慢可能是由多种原因引起的。为了避免主机缓慢：

- 确保所有 Impala 节点具有相同的配置。
- 确保其他工作负载，这是不被黑斑羚运行，均匀地通过设置分布 `yarn.scheduler.fair.assignmultiple` 属性 `false` 中 `yarn-site.xml`。
- 通过在非高峰时段运行 HDFS 平衡器实用程序，确保 HDFS 数据平衡，然后 `INVALIDATE METADATA` 在执行重新平衡以刷新 Hive Metastore 中的元数据之后运行 Impala 上的元数据。有关 更多信息，请参见 HDFS Balancer 和 INVALIDATE METADATA 语句。
- 确保 HDFS 块大小最佳。例如，使用 256MB 的块，并 在可拆分的块中使用快照压缩，这比 GZip 更好。

6.2.3.18. 将结果传输到客户端时最大程度地减少开销

在将查询结果发送回客户端时，请始终尽量减少开销。

使用以下技术：

- 始终 `LIMIT` 在查询中使用该子句以采样数据。有关更多信息，请参见 LIMIT 子句。

- 使用 NUM_ROWS_PRODUCED_LIMIT 查询选项可以限制查询产生的行数。对于希望运行探索性查询并且用户可能忘记添加 LIMIT 子句的用户池，可以将其设置为默认值。有关 更多信息，请参见 NUM_ROWS_PRODUCED_LIMIT 查询选项。
- 使用时 impala-shell，请使用-B 选项以避免“漂亮打印”结果集，并使用--output_file 选项将查询结果重定向到文件 。例如：

```
impala-shell --ssl -k -i impala-coordinator1.company.com:21000 -B --output_file /tmp/result
```

6.2.3.19. 联接查询性能调优

通过使用连接顺序和使用适当的连接类型来调整连接查询。

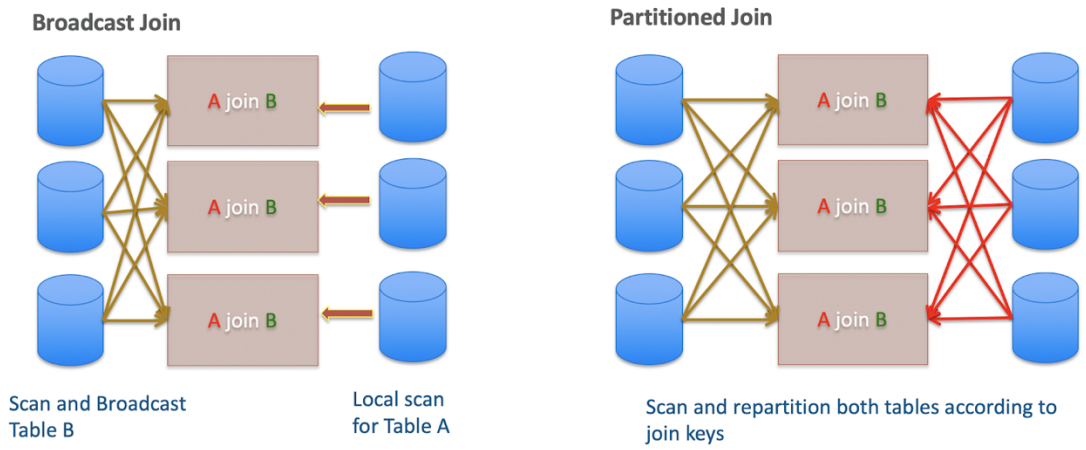
6.2.3.19.1. 连接顺序

- 确保在联接列上收集表和列统计信息。这使 Impala 可以创建最佳的联接计划。有关 更多信息，请参见 使用 ALTER TABLE 手动设置表和列统计信息。
- 如果表或列的统计信息不可用，请首先加入最大的表。如果表或列统计信息不适用于联接中的某些表，则 Impala 会对表进行重新排序。Impala 根据总体大小和基数，将具有统计信息的表按成本从高到低的顺序放在连接顺序的左侧。没有统计信息的表将被视为“零尺寸”，这意味着它们始终位于连接顺序的右侧。
- 如果要覆盖默认的 Impala 行为，请使用 STRAIGHT_JOIN 覆盖联接重新排序中 STRAIGHT_JOIN 介绍的 技术 。

6.2.3.19.2. 联接类型

- 广播联接是默认联接类型。在这种连接类型中，右侧表被认为小于左侧表，并且其内容被发送到查询所涉及的所有其他节点。
- 分区联接更适合大小大致相等的大型表。使用这种技术，每个表的部分将发送到其他适当的节点，在其中可以并行处理行的那些子集。
- 广播或分区联接的选择还取决于联接中所有表的可用统计信息。
- 可以使用以下查询选项手动设置联接类型：
    - SET DEFAULT_JOIN_DISTRIBUTION_MODE=shuffle;
    - SET DEFAULT_JOIN_DISTRIBUTION_MODE=broadcast;

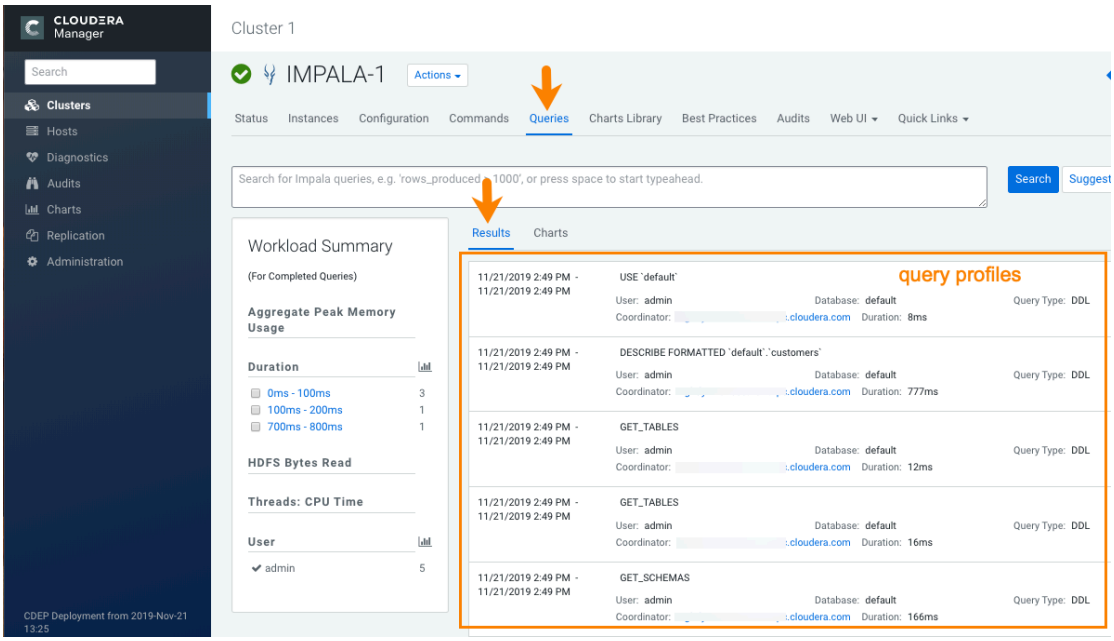
下图说明了广播和分区联接类型：



6.2.3.20. 查询简档

Impala 执行任何查询时，它将在查询配置文件中捕获执行的运行时详细信息。

您可以通过登录到 Impala shell 并运行 `profile;`命令来从命令行查看查询配置文件。如果要使用 Cloudera Manager，请从管理控制台的主页中选择 Impala >查询。查询配置文件显示在 此页面的“结果”选项卡上：



6.2.3.20.1. 执行摘要

查询概要文件具有执行摘要，该摘要显示查询中涉及的所有运算符，运算符在其上运行的节点数，花费的时间，产生的行以及每个节点上所需的内存。

摘要部分看起来像下面的示例摘要，用于一个简单的查询：

Operator	#Hosts	Avg Time	Max Time	#Rows	Est. #Rows	Peak Mem	Est. Peak Mem	Detail
10:SORT	23	8m15s	1h	290.01M	44.89M	58.29 GB	436.00 MB	
05:HASH JOIN	23	1s733ms	9s700ms	290.01M	44.89M	20.81 MB	1.94 MB	LEFT SEMI JOIN, BROADCAST
|--09:EXCHANGE	23	151.541us	664.160us	13	97	112.00 KB	0	BROADCAST
| 08:AGGREGATE	23	2.731ms	11.367ms	13	97	1.97 MB	10.00 MB	FINALIZE
| 07:EXCHANGE	23	319.697us	1.677ms	4.04K	968	192.00 KB	0	HASH(vpmn)
| 03:AGGREGATE	23	8s782ms	1m11s	4.04K	968	14.63 MB	10.00 MB	STREAMING
| 02:SCAN HDFS	23	10s745ms	3m41s	438.93M	447.94M	1.59 GB	4.81 GB	test.ext_call_event_fact cef
04:HASH JOIN	23	15s968ms	3m20s	438.93M	447.94M	13.18 MB	1.94 MB	INNER JOIN, BROADCAST
|--06:EXCHANGE	23	339.291us	3.671ms	8.04K	8.04K	544.00 KB	0	BROADCAST
| 01:SCAN HDFS	1	4.194ms	4.194ms	8.04K	8.04K	1.25 MB	32.00 MB	test.date_dim dd
00:SCAN HDFS	23	541.093ms	3s665ms	438.93M	447.94M	2.08 GB	4.81 GB	test.ext_call_event_fact cef

6.2.3.20.2. 查询时间表

查询时间轴显示执行查询所需的端到端时间。

它捕获查询的每个阶段所需的详细时间。下图显示了查询示例的时间轴：

```
   Query Compilation: 1m
      - Metadata load started: 351.060us (351.060us)
      - Metadata load finished. loaded-tables=1/1 load-requests=2 catalog-updates=37: 1m (1m)
      - Analysis finished: 1m (1.596ms)
      - Value transfer graph computed: 1m (18.009us)
      - Single node plan created: 1m (190.919us)
      - Distributed plan created: 1m (70.456us)
      - Lineage info computed: 1m (38.695us)
      - Planning finished: 1m (329.095us)
   Query Timeline: 1m2s
      - Query submitted: 120.423us (120.423us)
      - Planning finished: 1m (1m)
      - Submit for admission: 1m (1.758ms)
      - Completed admission: 1m (137.653us)
      - Ready to start on 1 backends: 1m (438.325us)
      - All 1 execution backends (1 fragment instances) started: 1m (2.886ms)
      - Rows available: 1m (5.431ms)
      - First row fetched: 1m1s (913.202ms)
      - Last row fetched: 1m1s (347.012ms)
      - Released admission control resources: 1m1s (99.751us)
      - Unregister query: 1m2s (560.234ms)
    - ComputeScanRangeAssignmentTimer: 1.063ms
  ImpalaServer:
     - ClientFetchWaitTimer: 1s819ms
     - RowMaterializationTimer: 1.133ms
```

6.2.3.21. 使用查询配置文件调试查询的常见方案

以下主题描述了影响 Impala 查询性能的常见问题以及如何解决它们。

6.2.3.21.1. 超出内存限制

未设置内存限制时，通常会发生超出内存限制的错误。

错误：

```
memory limit exceeded. Limit=2.00 GB Reservation=1.25 GB
ReservationLimit=1.60 GB OtherMemory=775.70 MB Total=2.00 GB Peak=2.00 GB
```

描述和原因：

通常，发生这种情况是因为未为池设置内存限制，或者缺少表统计信息，或者两者都不存在。

解决方案：

- `COMPUTE STATS` 为查询中涉及的每个表运行。
- 使用 `SET MEM_LIMIT` 查询选项设置内存限制重新运行查询。例如：
- `SET MEM_LIMIT=3gb;`

6.2.3.21.2. 查询运行缓慢

Impala 查询运行缓慢的原因有很多，本主题对此进行了说明。

导致查询运行缓慢的最主要原因是：

6.2.3.21.2.1. 缺少负载

错误：

```
Metadata load finished. loaded-tables=1/1 load-requests=1 catalog-updates=3:
2.75s (2746369188
```

描述和原因：

如果 Impala 在目录守护程序（`catalogd`）中没有缓存表的元数据，则查询运行缓慢

解决方案：

为避免这些情况，请确保在 ETL 管道中刷新了表，并确保您使用了按需元数据和元数据管理中介绍的按需元数据功能。

6.2.3.21.2.2. 缺少统计信息

错误：

```
WARNING: The following tables are missing relevant table and/or column
statistics. Default.web_logs
```

描述和原因：

缺少统计信息会导致错误的联接类型，例如哈希联接而不是广播联接。它还可能导致错误的加入顺序。这两种情况都会导致查询的运行速度低于最佳运行速度。

解决方案：

`COMPUTE STATS` 为查询中涉及的每个表运行，然后重新运行查询。

6.2.3.21.3. 准入控制

资源池的内存增加可以增加准入控制下的查询并发性。

Impala 准入控制可随时控制在 Impala 服务上运行的并发查询的数量。如果正在排队的查询与准入完成之间的时间差很大，请重新访问分配给该池的总内存：

```
Queued: 127ms (127000586)
Completed admission: 3.50s (3498016148)
```

通过增加内存，只要总并发在 Admission Control 中建议的限制内，就可以并发运行更多查询。

6.2.3.21.4. 客户端获取等待计时器

客户端获取等待计时器确定查询保持打开状态以等待客户端操作的时间。如此处所述调整计时器设置可以提高 Impala 性能。

6.2.3.21.4.1. 描述和原因：

如果客户端（例如 Hue）使查询保持打开状态以进行分页，则尽管释放了准入控制资源，但查询在此期间仍保持运行状态。这种情况会生成类似于以下内容的消息：

```
     Rows available: 1.1m (65504301365)
     First row fetched: 1.1m (66268304867)
     Unregister query: 4.1h (14710907024740)
ImpalaServer
     - ClientFetchWaitTimer: 4.1h (14622894717858)
```

6.2.3.21.4.2. 解决方案：

通过在 Cloudera Manager 中设置以下属性，确保 Hue 和 Impala 关闭查询：

- 在 hue_safety_valve.ini 的“ Hue Service 高级配置代码片段（安全阀）”配置属性

中，添加以下设置：

```
[impala]
     set close_queries=true
     query_timeout_s=<number_of_seconds>
     session_timeout_s=<number_of_seconds>
```

- 在“ Impala 服务空闲会话超时”配置属性中，添加希望会话在取消之前保持空闲的天数，小时数，分钟数或秒数。
- 在“ Impala 服务空闲查询超时”配置属性中，添加希望查询在取消之前保持空闲的天数，小时数，分钟数或秒数。

更改配置属性后，请不要忘记在 Cloudera Manager 中重新启动服务。

6.2.3.21.5. 加入顺序错误

调整查询联接顺序或联接类型可以提高 Impala 性能。

描述和原因：

使用不正确的连接顺序或不正确的连接类型也可能导致查询缓慢。

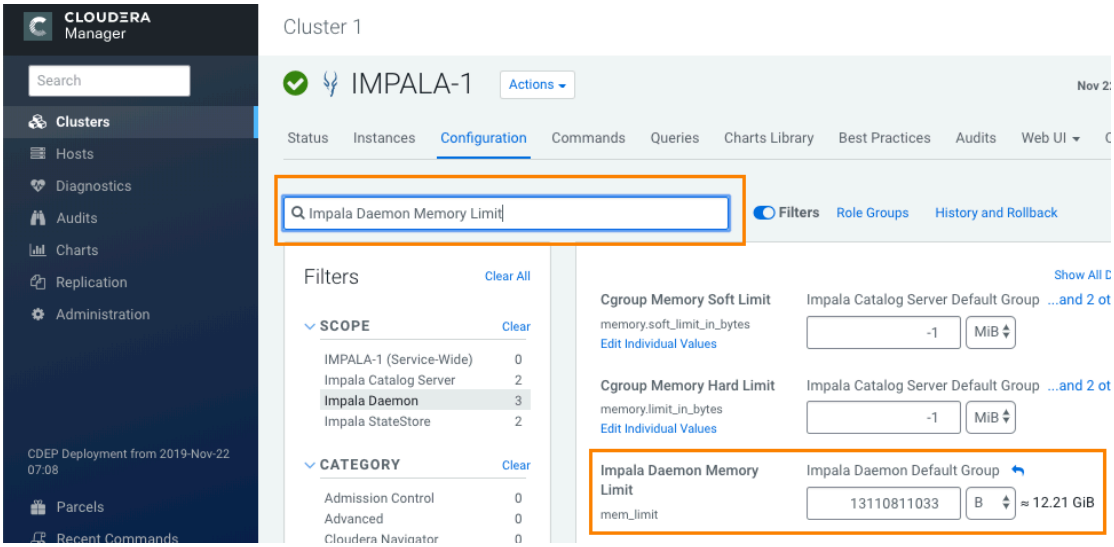
解决方案：

- 要确定联接类型和联接排序错误，请将实际行数（#Rows）与查询概要文件中的估计行数（Est。#Rows）相比较。如果实际行数与估计行数相差很大，则连接顺序和所选的连接类型可能不正确。
- 确保已收集所有表和列的统计信息，以便可以自动选择正确的连接顺序和类型。

6.2.3.21.6. 时间和数据偏斜

偏斜在平均时间和最大时间，这是在所列出的 Exec 的概要查询简档的部分，由于任一数据偏斜或工作量时滞是可能的：

- 可以通过定期运行 HDFS 平衡器然后运行 `INVALIDATE METADATA` 以重新平衡数据来修复数据偏斜。
- 通过确保在所有 Impala 守护程序上使用相似的 CPU 和内存配置，可以解决工作负载偏差。要解决 CPU 使用率问题，请确保在运行 Impala 守护程序的节点上使用了类似的硬件。要解决内存配置，请使用 Cloudera Manager。从其主页中选择 Impala >配置。然后搜索“ Impala 守护程序内存限制”配置属性：



6.3. 加速 SparkML 应用

Spark ML 是许多主要机器学习算法的主要框架之一，例如推荐系统的交替最小二乘（ALS）算法，主成分分析算法和随机森林算法。但是，频繁的错误配置意味着很少充分利用 Spark ML 的潜力。对 Spark ML 使用原生数学库是一种实现这种潜力的方法。

本文讨论如何通过使用 Spark ML 的原生库来加快模型训练速度。此外，它还讨论了 Spark ML 为什么受益于原生库，如何使用 CDH Spark 启用原生库，以及如何在不同原生库上的 Spark ML 之间进行性能比较。

6.3.1. Spark ML 的原生数学库

Spark 的 MLlib 使用 Breeze 线性代数软件包，这取决于 `netlib-java` 优化的数值处理。`netlib-java` 是底层 BLAS，LAPACK 和 ARPACK 库的包装 。

尽管由于运行时专有二进制文件的许可问题，Spark 的 MLlib 可以使用这些库，但是 `netlib-java` 默认情况下，Spark 的 Cloudera 发行版和 Apache Spark 的社区版本都不包含原生代理。因此，如果不进行手动配置，则 `netlib-java` 仅使用 F2J 库，该库是从 Fortran77 参考源代码转换而来的基于 Java 的数学库。

要检查您是在 Spark ML 还是基于 Java 的 F2J 中使用原生数学库，请使用 Spark shell 加载和打印的实现库 `netlib-java`。例如，以下命令返回有关 BLAS 库的信息，并包括在行中使用 F2J 的信息，该行在 `com.github.fommil.netlib.F2jBLAS` 下面以粗体显示：

```
scala> import com.github.fommil.netlib.BLAS
import com.github.fommil.netlib.BLAS

scala> println(BLAS.getInstance().getClass().getName())
18/12/10 01:07:06 WARN netlib.BLAS: Failed to load implementation
from: com.github.fommil.netlib.NativeSystemBLAS
18/12/10 01:07:06 WARN netlib.BLAS: Failed to load implementation
from: com.github.fommil.netlib.NativeRefBLAS
com.github.fommil.netlib.F2jBLAS
```

6.3.1.1. 采用 Spark ML 原生数学库

原生库提供的加速范围因模型而异。

Anand Iyer 和 Vikram Saletore 在他们的工程博客中显示， 像 OpenBLAS 和 Intel 的 Math Kernel Library（MKL）之类的本地数学库可以提高 Spark ML 的训练性能。至于推荐系统中使用的矩阵分解模型（交替最小二乘（ALS）算法），OpenBLAS 和 Intel 的 MKL 的模型训练速度都比 F2J 实施快 4.3 倍。其他诸如潜在狄利克雷分配（LDA），主成分分析（PCA）和奇异值分解（SVD）算法的结果显示，英特尔 MKL 的性能提高了 56％至 72％，OpenBLAS 的性能提高了 10％至 50％。

但是，该博客文章还演示了某些算法（例如，Random Forest 和 Gradient Boosted Tree）在启用 OpenBLAS 或 MKL 后几乎没有速度加速。其原因主要是这些基于树的算法的训练集不是向量。这表明，无论是 OpenBLAS 还是 MKL，原生库都可以更好地适应算法，这些算法的训练集可以作为矢量进行运算，并且可以作为一个整体进行计算。将数学加速用于使用矩阵运算对训练集进行运算的算法更为有效。

6.3.2. 启用 libgfortran 库

此过程显示了如何使用 Cloudera Manager 启用 `libgfortran` 数学库来加速 Spark ML 应用程序。

重要

以下说明适用于 CDH 5.x 中的 Spark 1.6x 和 CDH 6.x 中的 Spark2.x。在这些版本中，GPLEXTRAS 会自动将使用原生库所需的 JAR 文件的类路径添加到中 `/etc/spark/conf/classpath.txt`。然后 `spark-env.sh` 在引导过程中加载额外的 Java 库。但是，GPLEXTRAS 无法对 CDH 5.x 中的 Spark 2.x 执行此操作。如果要使用 Spark 2.x，则必须升级

到 CDH 6.x，CDH 6.x 会自动为 Spark 2.x 执行此配置。在以下示例中，使用了 RHEL 7.4 上的 CDH 5.15。

1) libgfortran 在每个 CDH 节点上启用 4.8 库。例如，在 RHEL 中，在每个节点上运行以下命令：

```
yum -y install libgfortran
```

2) 在 Cloudera Manager 中安装 GPLEXTRAS Parcel，然后将其激活：

a. 要安装 GPLEXTRAS Parcel，请参阅 Cloudera Manager 文档中的“安装 GPL Extras Parcel”。

b. 要激活 Parcel，请参阅激活 Parcel。

c. 激活 GPLEXTRAS Parcel 在了 Cloudera 管理器，导航到后主机>Parcel，以确认 GPLEXTRAS Parcel 被激活：

Parcel Name	Version	Status	
CDH 5	5.15.2-1.cdh5.15.2.p0.3	Distributed, Activated	Deactivate
GPLEXTRAS	5.15.2-1.cdh5.15.2.p0.3	Distributed, Activated	Deactivate
KAFKA	3.1.1-1.3.1.1.p0.2	Distributed, Activated	Deactivate
SPARK2	2.3.0.cloudera4-1.cdh5.13.3.p0.611179	Distributed, Activated	Deactivate
mkl	2019.1.144	Distributed, Activated	Deactivate
mkl_wrapper_parcel	1.0	Distributed, Activated	Deactivate

3) 在 Cloudera Manager 的指导下重新启动适当的 CDH 服务。

4) 重新启动完成后，请使用以下命令使用 Spark Shell 验证是否正在加载原生库：

```
scala> import com.github.fommil.netlib.BLAS
import com.github.fommil.netlib.BLAS

scala> println(BLAS.getInstance().getClass().getName())
18/12/23 06:29:45 WARN netlib.BLAS: Failed to load implementation
from: com.github.fommil.netlib.NativeSystemBLAS
18/12/23 06:29:45 INFO jni.JniLoader: successfully loaded
/tmp/jniloader6112322712373818029netlib-native_ref-linux-x86_64.so
com.github.fommil.netlib.NativeRefBLAS
```

您可能想知道为什么仍然会有关于 NativeSystemBLAS 失败加载的警告消息。不用担心 之所以在这里，是因为我们仅设置 Spark 使用的原生库，而不是在系统范围内使用。您可以放心地忽略此警告。

6.3.3. 启用英特尔 MKL 库

此过程显示了如何使用 Cloudera Manager 启用 Intel MKL 数学库来加速 Spark ML 应用程序。

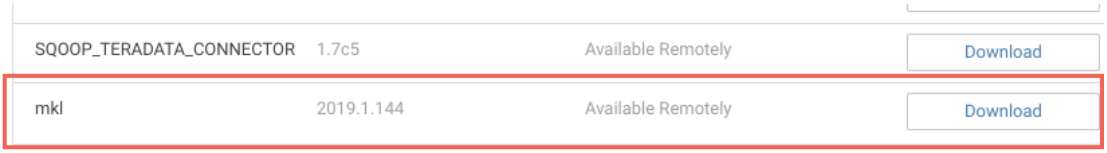
重要

以下说明适用于 CDH 5.x 中的 Spark 1.6x 和 CDH 6.x 中的 Spark2.x。在这些版本中，GPLEXTRAS 会自动将使用原生库所需的 JAR 文件的类路径添加到中 `/etc/spark/conf/classpath.txt`。然后 `spark-env.sh` 在引导过程中加载额外的 Java 库。但是，GPLEXTRAS 无法对 CDH 5.x 中的 Spark 2.x 执行此操作。如果要使用 Spark 2.x，则必须升级到 CDH 6.x，CDH 6.x 会自动为 Spark 2.x 执行此配置。在以下示例中，使用了 RHEL 7.4 上的 CDH 5.15。

1) 英特尔在其网站上将 MKL 原生库作为 Cloudera Manager 软件包提供。您可以在 Cloudera Manager 中将其添加为远程 Parcel 存储库。然后，您可以下载该库并激活它：
   a) 在 Cloudera 的经理，导航到主机>Parcel。
   b) 选择配置。
   c) 在“远程 Parcel 存储库 URL ”部分中，单击加号并添加以下 URL：

```
http://parcels.repos.intel.com/mkl/latest
```

   d) 单击保存更改，然后返回到列出可用宗地的页面。
   e) 单击下载以获取 mklParcel：



   f) 单击“分发”，完成分发到集群上的主机后，单击“ 激活”。

2) MKL 包仅由 Linux 共享库文件（.so 文件）组成，因此要使其可被 JVM 访问，必须制作 JNI 包装器。要制作包装纸，请使用以下 MKL 包装纸 Parcel。使用步骤 1 中描述的相同过程，将以下链接添加到 Cloudera Manager 宗地配置页面，下载该宗地，在主机之间分发它，然后激活它：

```
https://raw.githubusercontent.com/Intel-bigdata/mkl-wrappers-parcel-repo/master/
```

3) 在 Cloudera Manager 的指导下重新启动相应的 CDH 服务，并在需要时重新部署客户端配置。
4) 在 Cloudera Manager 中，将以下配置信息添加到 spark-conf / spark-defaults.conf 的 Spark Client 高级配置代码片段（安全阀）中：

```
spark.driver.extraJavaOptions=-
Dcom.github.fommil.netlib.BLAS=com.intel.mkl.MKLBLAS -
Dcom.github.fommil.netlib.LAPACK=com.intel.mkl.MKLLAPACK
spark.driver.extraClassPath=/opt/cloudera/parcels/mkl_wrapper_parcel/l
ib/java/mkl_wrapper.jar
spark.driverEnv.MKL_VERBOSE=1
spark.executor.extraJavaOptions=-
Dcom.github.fommil.netlib.BLAS=com.intel.mkl.MKLBLAS -
Dcom.github.fommil.netlib.LAPACK=com.intel.mkl.MKLLAPACK
spark.executor.extraClassPath=/opt/cloudera/parcels/mkl_wrapper_parcel
/lib/java/mkl_wrapper.jar
spark.executorEnv.MKL_VERBOSE=1
```

此配置信息指示 Spark 应用程序加载 MKL 包装器，并将 MKL 用作 Spark ML 的默认原生库。

重要

- 通过设置 `MKL_VERBOSE=1`，MKL 记录调用了哪些计算函数，将哪些参数传递给它们以及花费了多少时间来执行这些函数。该信息对于实现很有用，但是会在集群中的 HDFS 上占用大量空间。在下一节讨论的实验案例中，每个作业的日志可能会占用数百 GB 的空间。
- 如果 `UnsatisfiedLinkError` 在验证使用的原生库时返回消息，如下所示，请将 `/opt/cloudera/parcels/mkl/linux/mkl/lib/intel64` 目录添加到 `LD_LIBRARY_PATH` 每个集群节点的环境变量中。

```
Native code library failed to load.
java.lang.UnsatisfiedLinkError:
/opt/cloudera/parcels/mkl_wrapper_parcel-
1.0/lib/native/mkl_wrapper.so: libmkl_rt.so: cannot open shared object
file: No such file or directory
```

再次打开 Spark Shell 来验证原生库，您应该看到以下输出：

```
scala> import com.github.fommil.netlib.BLAS
import com.github.fommil.netlib.BLAS

scala> println(BLAS.getInstance().getClass().getName())
com.intel.mkl.MKLBLAS
```

6.3.4. 性能比较

在本主题中，我们使用 ALS 算法将培训速度与不同的基础数学库（包括 F2J `libgfortran`，和 Intel 的 MKL）进行比较。

我们使用的硬件是 Amazon EC2 的 VM 实例 `r4.large`，每个实例具有 2 个 CPU 内核和 15.25 GB 的内存。此外，我们将 CentOS 7.5 和 CDH 5.15.2 与 Spark 2.3 Release 4 的 Cloudera 发行版一起使用。培训代码摘自 Sandy Ryza 等人在《*Advanced Analytics wit*

h Spark（第二版）》的 ALS 章节的核心部分。Al，O'Reilly（2017）。培训数据集是 Audioscrobbler 发布的数据集，可以从以下网站下载：

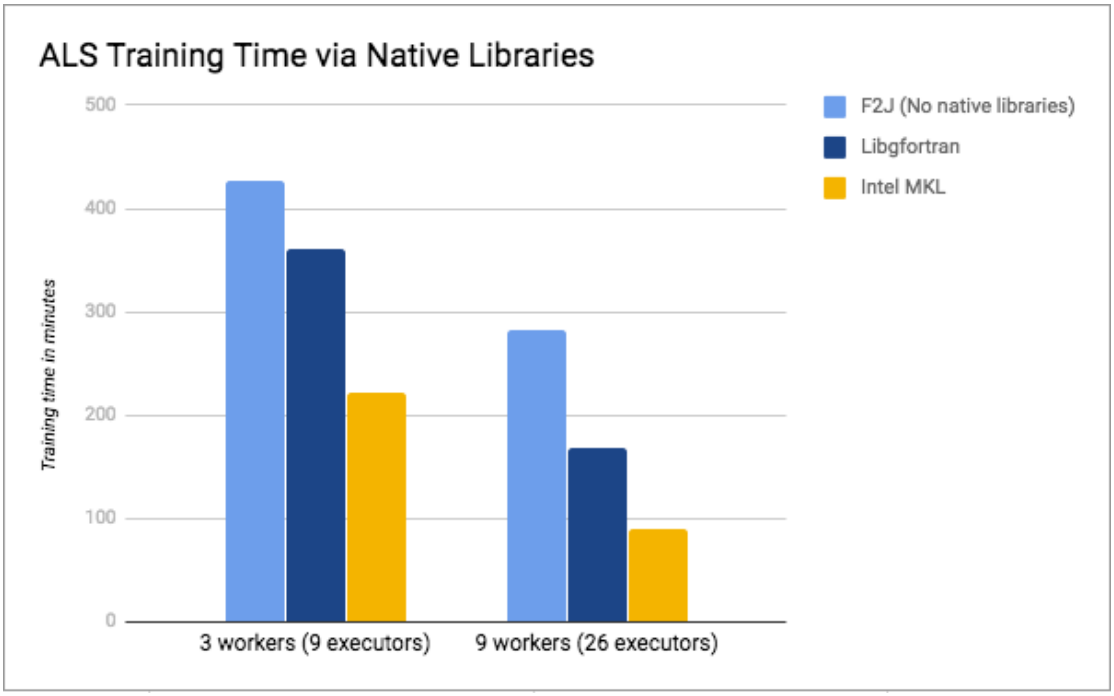
```
https://storage.googleapis.com/aas-data-sets/profiledata_06-May-2005.tar.gz
```

通常，ALS 模型的等级设置为比默认值 10 大得多的值，因此我们在此处使用值 200 以确保结果更接近真实示例。以下是用于为我们的 ALS 模型设置参数值的代码：

```
val model = new ALS().
  setSeed(Random.nextLong()).
  setImplicitPrefs(true).
  setRank(200).
  setRegParam(0.01).
  setAlpha(1.0).
  setMaxIter(20).
  setUserCol("user").
  setItemCol("artist").
  setRatingCol("count").
  setPredictionCol("prediction")
```

下表和图显示了使用 Spark ML 的不同原生库时的培训时间。这些值以分钟为单位显示。我们可以看到，libgfortran 英特尔的 MKL 和 MKL 的确都提高了训练速度的性能，而 MKL 的表现似乎更胜一筹。从这些实验结果中，libgfortran 可将其提高 18％至 68％，而 MKL 可将其提高 92％至 213％。

工人/执行员人数	F2J	libgfortran	英特尔 MKL
3 名工人（9 名执行者）	426	360	222
9 名工人（26 执行者）	282	168	90



7. 故障排查

7.1. 安全故障排查

与安全相关的问题可以通过多种方式体现出来，并且可以与各种来源相关联，包括 Kerberos 身份验证，HDFS 加密和 TLS/SSL（传输加密中的数据）。本节包括错误消息和常见问题的疑难解答，以及有关某些基础组件的一些体系结构详细信息。

7.1.1. 错误信息和各种故障

7.1.1.1. *启用 Kerberos 后，集群无法运行作业*

症状：为集群启用了 Kerberos 后，先前未配置 Kerberos 身份验证的集群可能无法在某些 TaskTrackers（MRv1）或 NodeManagers（YARN）上为某些用户运行作业。错误可能会显示在 TaskTracker 或 NodeManager 日志中。以下示例错误来自 MRv1 上的 TaskTracker：

```
10/11/03 01:29:55 INFO mapred.JobClient: Task Id : attempt_201011021321_0004_m_000011_0, Status : FAILED
Error initializing attempt_201011021321_0004_m_000011_0:
java.io.IOException: org.apache.hadoop.util.Shell$ExitCodeException:
```

```
at org.apache.hadoop.mapred.LinuxTaskController.runCommand(LinuxTaskController.java:212)
at org.apache.hadoop.mapred.LinuxTaskController.initializeUser(LinuxTaskController.java:442)
at org.apache.hadoop.mapreduce.server.tasktracker.Localizer.initializeUserDirs(Localizer.java:272)
at org.apache.hadoop.mapred.TaskTracker.localizeJob(TaskTracker.java:963)
at org.apache.hadoop.mapred.TaskTracker.startNewTask(TaskTracker.java:2209)
at org.apache.hadoop.mapred.TaskTracker$TaskLauncher.run(TaskTracker.java:2174)
Caused by: org.apache.hadoop.util.Shell$ExitCodeException:
at org.apache.hadoop.util.Shell.runCommand(Shell.java:250)
at org.apache.hadoop.util.Shell.run(Shell.java:177)
at org.apache.hadoop.util.Shell$ShellCommandExecutor.execute(Shell.java:370)
at org.apache.hadoop.mapred.LinuxTaskController.runCommand(LinuxTaskController.java:203)
... 5 more
```

可能的原因：此问题是由 TaskTracker 和 NodeManager 的目录中的旧内容导致的，该旧内容在为以前不使用 Kerberos 的集群配置 Kerberos 身份验证之后可能存在。导致此问题的事件顺序如下：

1) 未配置为进行 Kerberos 身份验证的集群用于运行作业，这些作业在每个 TaskTracker 或 NodeManager 主机上创建了一个或多个本地用户目录。
2) 然后将集群配置为使用 Kerberos 身份验证。
3) 用户尝试在新保护的集群上运行作业，但是 TaskTrackers 或 NodeManagers 上的本地用户目录归错误的用户所有或具有过高的许可权限。

在为集群启用 Kerberos 身份验证时，应该已经清理了这些目录。

步骤来解决：在集群中为受影响的用户删除 `mapred.local.dir` 或 `yarn.nodemanager.local-dirs` 目录。

7.1.1.2. NameNode 无法启动

症状：尝试启动 NameNode 时发生登录失败。在集群上启用调试后，可能会显示以下 KrbException 消息：

```
Caused by: KrbException: Integrity check on decrypted field failed (31) - PREAUTH_FAILED}}
Caused by: KrbException: Identifier does not match expected value (906)
```

可能的原因：此问题可能是由于 AES-256 的配置不正确。默认情况下，某些操作系统（CentOS/Red Hat Enterprise Linux 5.6（及更高版本），Ubuntu）使用 AES-256 加密，这要求在所有主机上安装“ Java 密码学扩展（JCE）无限强度管辖权策略文件”（请参阅“ JCE”）。对于 AES-256 加密策略文件”的说明），或禁用 AES-256 支持的 `kdc.conf` 或 `krb5.com`（见‘的说明禁用 JCE 策略文件的 AES-256 加密’）。

解决步骤： KrbException 31 和 KrbException 906 可能是由各种问题引起的，但最可能的原因是配置错误的 AES-256 加密。解决该问题应首先确定为集群配置的加密类型。

要验证为集群配置的加密类型：

1) 在本地 KDC 主机上，键入以下命令以创建测试 principal：

```
kadmin -q "addprinc test"
```

2) 在集群主机上，键入以下命令以启动 Kerberos 会话作为测试：

```
kinit test
```

3) 在集群主机上，键入以下命令以查看正在使用的加密类型：

```
klist -e
```

如果使用 AES-256，将显示以下输出：

```
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: test@SCM
Valid starting     Expires            Service principal
05/19/11 13:25:04  05/20/11 13:25:04  krbtgt/SCM@SCM
   Etype (skey, tkt): AES-256 CTS mode with 96-bit SHA-1 HMAC, AES-256 CTS mode with 96-bit SHA-1 HMAC
```

要从 Kerberos 配置文件中删除 AES-256 加密，请执行以下操作：

- 从 `kdc.conf` 或 `krb5.conf` 文件的 `supported_enctypes` 字段中 删除 `aes256-cts:normal`。
- 更改配置后，重新启动 KDC 和 kadmin 服务器。
- 根据需要更改 TGTprincipal（krbtgt/REALM @ REALM）和其他 principal 密码。
- 在 `kdc.conf` 文件的[realms]部分，对于与 `HADOOP.LOCALDOMAIN` 关联的领域，添加（或替换，如果已经存在）以下变量：

```
supported_enctypes = des3-hmac-sha1:normal arcfour-hmac:normal des-hmac-sha1:normal des-cbc-md5:normal des-cbc-crc:normal des-cbc-crc:v4 des-cbc-crc:afs3
```

- 按照“使用 Cloudera Manager 管理 Kerberos 凭据”中的说明重新创建 hdfskeytab 文件和 mapredkeytab 文件。

7.1.1.3. 客户端无法连接到 NameNode

症状： NameNode 键表文件没有 AES-256 条目，但客户端票证有。NameNode 启动，但是客户端无法连接到它。错误消息未指定“ AES256”，而是包含编码类型“ 18”。

可能的原因：与 AES-256 加密和 JCE 库有关的问题。

解决步骤：验证是否已安装 Java 密码学扩展（JCE）无限强度管辖权策略文件（有关说明，请参阅“ AES-256 加密的 JCE 策略文件”）或者从 `kdc.conf` 或 `krb5.conf` 文件的 `supported_enctypes` 字段中删除 `aes256-cts:normal`，如上所述。

7.1.1.4. Hadoop 命令在本地领域中运行，但不在远程领域中运行

症状：启用跨域信任后，在本地领域中作为 principal 进行身份验证可以使您成功运行 Hadoop 命令，而在远程领域中作为 principal 进行身份验证则无法。

可能的原因：此问题通常是由于两个领域中的 principal 对于每个领域中的跨领域 principal 具有不同的加密类型或不同的密码。由于本地和远程领域各自发出自己的 TGT，因此将运行本地命令，但是无法授予本地和远程领域进行通信所需的服务票证。

解决步骤：使用 `kadmin.local` 或 `kadmin` 适当的方式将跨域 `krbtgt Principal` 及其加密类型添加到 MIT KDC 服务器（本地访问与远程）：

```
kadmin: addprinc -e "enc_type_list" krbtgt/LOCAL-REALM.EXAMPLE.COM@MAIN-REALM.COMPANY.COM
```

指定跨域 principal（`krbtgt`）支持的加密类型，例如 AES，DES 或 RC4。可以在 *enc_type_list* 中指定多种加密类型，只要其中一种加密类型与远程领域中 KDC 授予的票证的加密类型相匹配即可。例如：

```
kadmin:  addprinc -e "aes256-cts:normal rc4-hmac:normal des3-hmac-sha1:normal" krbtgt/LOCAL-REALM.EXAMPLE.COM@MAIN-REALM.COMPANY.COM
```

7.1.1.5. 用户在运行 Hadoop 作业或命令时无法获取凭据

症状：用户尝试进行身份验证，但显示以下消息：

```
13/01/15 17:44:48 DEBUG ipc.Client: Exception encountered while connecting to the server :
javax.security.sasl.SaslException:
GSS initiate failed [Caused by GSSException: No valid credentials provided (Mechanism level: Fail to create credential.
(63) - No service creds)]
```

可能的原因：票证消息对于 UDP 协议而言可能太大（默认情况下，SASL 使用该协议）。

步骤来解决：强制 Kerberos 通过添加下面的参数，而不是使用 UDP TCP `libdefaults` 中 `krb5.conf` 在哪里发生问题的客户机的文件：

```
[libdefaults]
udp_preference_limit = 1
```

`krb5.conf` 通过 Cloudera Manager 进行配置，它将自动添加到中 `krb5.conf`。

注意

将消息发送到 KDC 时，如果票证消息的大小大于为 `udp_preference_limit` 属性指定的设置，则库将尝试在 UDP 之前先使用 TCP 。如果票证消息小于 `udp_preference_limit` 设置的票证消息，则将在 TCP 之前尝试 UDP。无论大小，如果第一次尝试失败，都将尝试两种协议。

7.1.1.6. 服务日志中的伪造重播异常

症状：对 Kerberos 受保护的服务的多个有效请求在未被请求时被标识为 尝试*重播*。以下异常在一个或多个 Hadoop 守护程序的日志中显示：

```
2013-02-28 22:49:03,152 INFO  ipc.Server (Server.java:doRead(571)) - IPC Server listener on 8020:
readAndProcess threw exception javax.security.sasl.SaslException: GSS initiate failed [Caused by
GSSException: Failure unspecified at GSS-API level (Mechanism l
javax.security.sasl.SaslException: GSS initiate failed [Caused by GSSException: Failure unspecified at GSS-API
level (Mechanism level: Request is a replay (34))]
      at com.sun.security.sasl.gsskerb.GssKrb5Server.evaluateResponse(GssKrb5Server.java:159)
      at org.apache.hadoop.ipc.Server$Connection.saslReadAndProcess(Server.java:1040)
      at org.apache.hadoop.ipc.Server$Connection.readAndProcess(Server.java:1213)
      at org.apache.hadoop.ipc.Server$Listener.doRead(Server.java:566)
      at org.apache.hadoop.ipc.Server$Listener$Reader.run(Server.java:363)
Caused by: GSSException: Failure unspecified at GSS-API level (Mechanism level: Request is a replay (34))
      at sun.security.jgss.krb5.Krb5Context.acceptSecContext(Krb5Context.java:741)
      at sun.security.jgss.GSSContextImpl.acceptSecContext(GSSContextImpl.java:323)
      at sun.security.jgss.GSSContextImpl.acceptSecContext(GSSContextImpl.java:267)
      at com.sun.security.sasl.gsskerb.GssKrb5Server.evaluateResponse(GssKrb5Server.java:137)
      ... 4 more
Caused by: KrbException: Request is a replay (34)
      at sun.security.krb5.KrbApReq.authenticate(KrbApReq.java:300)
      at sun.security.krb5.KrbApReq.<init>(KrbApReq.java:134)
      at sun.security.jgss.krb5.InitSecContextToken.<init>(InitSecContextToken.java:79)
      at sun.security.jgss.krb5.Krb5Context.acceptSecContext(Krb5Context.java:724)
      ... 7 more
```

此问题还可能表现为集群客户端的性能不佳，包括连接断开，尝试进行 RPC 调用的超时等等。

可能的原因： Kerberos 使用第二种分辨率的时间戳来防止重放攻击（攻击者可以在其中记录网络流量，并在以后回放记录的请求以获得更高的特权）。也就是说，Kerberos 将传入的请求缓存一会儿，如果在几秒钟内有类似的请求，则 Kerberos 将能够将它们检测为重播攻击尝试（有关更多信息，请参见“ MIT Kerberos 重播缓存”）。

但是，如果同时存在多个有效的 Kerberos 请求，由于以下原因，这些请求也可能被误判为攻击：

- **集群中的多个服务正在使用相同的 Kerberos principal**。在多台计算机上运行的所有安全客户端应为每台计算机使用唯一的 Kerberos principal。例如，`myservice@EXAMPLE.COM` 服务应具有按主机的 principal ，而不是作为服务 principal 进行连接 `myservice/host123.example.com@EXAMPLE.COM`。
- **时钟未同步**：所有主机都应运行 NTP，以便客户端和服务器之间的时钟保持同步。

解决步骤：

尽管每种服务具有不同的 principal，并且时钟同步有助于减轻问题，但是，即使实现了上述所有条件，问题仍然存在。在这种情况下，禁用缓存（*并因此*禁用*重放保护*）将允许并行请求成功。可以通过将 `KRB5RCACHETYPE` 环境变量设置为来在可用性和安全性之间进行折衷 `none`。

请注意， Java 应用程序不会自动检测到 `KRB5RCACHETYPE`。对于基于 Java 的组件：

- 确保集群在 JDK 8 上运行。
- 要禁用重播缓存，请添加`-Dsun.security.krb5.rcache=none` 到目标 JVM 的 Java Opts/Arguments。例如，HiveServer2 或 Sentry 服务。

7.1.1.7. Cloudera Manager 集群服务无法启动

症状：一个或多个集群服务无法启动。例如，DataNode 无法启动。日志文件中的错误消息可能会显示以下错误消息：

```
Exception in secureMain
java.lang.ExceptionInInitializerError
at javax.crypto.KeyGenerator.nextSpi(KeyGenerator.java:324)
at javax.crypto.KeyGenerator.<init>(KeyGenerator.java:157)
...
Caused by: java.lang.SecurityException: The jurisdiction policy files are not signed by a trusted signer!
at javax.crypto.JarVerifier.verifyPolicySigned(JarVerifier.java:289)
at javax.crypto.JceSecurity.loadPolicies(JceSecurity.java:316)
at javax.crypto.JceSecurity.setupJurisdictionPolicies(JceSecurity.java:261)
...
```

可能的原因：这是 AES-256 加密不匹配的另一个示例。当 JCE 策略文件的版本与节点上安装的 Java 版本不匹配时，服务将无法启动，因为无法验证 JCE 策略文件的加密签名，导致出现上面的消息。

- 检查您的 KDC 和 `krb5.conf` 所有主机上的加密类型是否匹配 。

解决方案：如果使用的是 AES-256，请按照“用于 AES-256 加密的 JCE 策略文件”中的说明在所有主机上部署 JCE 策略文件。

- 服务无法启动

解决方案：为您正在运行的 Java 版本下载正确的 JCE 策略文件：

- Java 8
- Java 7
- Java 6 [旧版]

下载并解压缩该 zip 文件。将两个 JAR 文件复制到集群中每个节点上的 `$JAVA_HOME/jre/lib/security` 目录中。

错误讯息

7.1.1.8. 不正确的权限 Java 异常（java.io.IOException）

症状：尝试运行作业时显示错误的权限错误：

```
java.io.IOException: Incorrect permission for
/var/folders/B3/B3d2vCm4F+mmWzVPB89W6E+++TI/-Tmp-/tmpYTil84/dfs/data/data1,
expected: rwxr-xr-x, while actual: rwxrwxr-x
    at org.apache.hadoop.util.DiskChecker.checkPermission(DiskChecker.java:107)
    at org.apache.hadoop.util.DiskChecker.mkdirsWithExistsAndPermissionCheck(DiskChecker.java:144)
    at org.apache.hadoop.util.DiskChecker.checkDir(DiskChecker.java:160)
    at org.apache.hadoop.hdfs.server.datanode.DataNode.makeInstance(DataNode.java:1484)
    at org.apache.hadoop.hdfs.server.datanode.DataNode.instantiateDataNode(DataNode.java:1432)
    at org.apache.hadoop.hdfs.server.datanode.DataNode.instantiateDataNode(DataNode.java:1408)
    at org.apache.hadoop.hdfs.MiniDFSCluster.startDataNodes(MiniDFSCluster.java:418)
   ...
```

可能的原因：守护程序的 umask 为 0002，而不是 0022。

解决步骤：确保 `umask`for `hdfs` 和 `mapred`is `0022`。

7.1.1.9. MapReduce（MRv1）错误

这些错误消息仅与 MapReduce 关联（而不与 YARN）关联。

7.1.1.10. 作业将无法运行，并且无法访问 mapred.local.dir 中的文件

症状： TaskTracker 日志包含以下错误消息：

```
WARN org.apache.hadoop.mapred.TaskTracker: Exception while localization java.io.IOException: Job
initialization failed (1)
```

可能的原因：

解决步骤：

1) 将 mapred 用户添加到所有主机上的 mapred 和 hadoop 组。
2) 重新启动所有 TaskTrackers。

7.1.1.11. 作业无法运行，TaskTracker 无法创建本地映射目录

症状： TaskTracker 日志包含以下错误消息：

```
11/08/17 14:44:06 INFO mapred.TaskController: main : user is atm
11/08/17 14:44:06 INFO mapred.TaskController: Failed to create
directory/var/log/hadoop/cache/mapred/mapred/local1/taskTracker/atm - No such file or directory
11/08/17 14:44:06 WARN mapred.TaskTracker: Exception while localization java.io.IOException: Job
initialization failed (20)
    at org.apache.hadoop.mapred.LinuxTaskController.initializeJob(LinuxTaskController.java:191)
    at org.apache.hadoop.mapred.TaskTracker$4.run(TaskTracker.java:1199)
    at java.security.AccessController.doPrivileged(Native Method)
    at javax.security.auth.Subject.doAs(Subject.java:396)
    at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1127)
    at org.apache.hadoop.mapred.TaskTracker.initializeJob(TaskTracker.java:1174)
    at org.apache.hadoop.mapred.TaskTracker.localizeJob(TaskTracker.java:1089)
    at org.apache.hadoop.mapred.TaskTracker.startNewTask(TaskTracker.java:2257)
    at org.apache.hadoop.mapred.TaskTracker$TaskLauncher.run(TaskTracker.java:2221)
Caused by: org.apache.hadoop.util.Shell$ExitCodeException:
    at org.apache.hadoop.util.Shell.runCommand(Shell.java:255)
    at org.apache.hadoop.util.Shell.run(Shell.java:182)
    at org.apache.hadoop.util.Shell$ShellCommandExecutor.execute(Shell.java:375)
    at org.apache.hadoop.mapred.LinuxTaskController.initializeJob(LinuxTaskController.java:184)
    ... 8 more
```

可能的原因：和 mapred-site.xmltaskcontroller.cfg 中 mapred.local.dir 指定的值 不匹配。这些值应该相同。

步骤来解决：验证 mapred-site.xml 和 taskcontroller.cfg 的 mapred.local.dir 设置是两个相同，如果需要重新组态。

7.1.1.12. 作业无法运行，TaskTracker 无法创建 Hadoop 日志目录

症状： TaskTracker 日志包含类似于以下内容的错误消息：

```
11/08/17 14:48:23 INFO mapred.TaskController: Failed to create
directory/home/atm/src/cloudera/hadoop/build/hadoop-0.23.2-cdh3u1-
SNAPSHOT/logs1/userlogs/job_201108171441_0004 - No such file or directory
11/08/17 14:48:23 WARN mapred.TaskTracker: Exception while localization java.io.IOException: Job
initialization failed (255)
    at org.apache.hadoop.mapred.LinuxTaskController.initializeJob(LinuxTaskController.java:191)
    at org.apache.hadoop.mapred.TaskTracker$4.run(TaskTracker.java:1199)
    at java.security.AccessController.doPrivileged(Native Method)
    at javax.security.auth.Subject.doAs(Subject.java:396)
    at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1127)
```

```
    at org.apache.hadoop.mapred.TaskTracker.initializeJob(TaskTracker.java:1174)
    at org.apache.hadoop.mapred.TaskTracker.localizeJob(TaskTracker.java:1089)
    at org.apache.hadoop.mapred.TaskTracker.startNewTask(TaskTracker.java:2257)
    at org.apache.hadoop.mapred.TaskTracker$TaskLauncher.run(TaskTracker.java:2221)
Caused by: org.apache.hadoop.util.Shell$ExitCodeException:
    at org.apache.hadoop.util.Shell.runCommand(Shell.java:255)
    at org.apache.hadoop.util.Shell.run(Shell.java:182)
    at org.apache.hadoop.util.Shell$ShellCommandExecutor.execute(Shell.java:375)
    at org.apache.hadoop.mapred.LinuxTaskController.initializeJob(LinuxTaskController.java:184)
    ... 8 more
```

可能的原因：配置错误。

步骤来解决：在 MRv1，为指定的默认值 `hadoop.log.dir` 的 `mapred-site.xml` 是 `/var/log/hadoop-0.20-mapreduce`。该路径必须是用户 `mapred` 拥有的并且是可写的。如果更改为指定的默认值 `hadoop.log.dir`，确保 `mapred-site.xml` 和 `taskcontroller.cfg` 中该值是相同的 。如果值不同，则返回以上错误消息。

7.1.2. 身份验证和 Kerberos 问题

使用 Kerberos 进行身份验证的集群有几种潜在的潜在问题根源，包括：

- 密钥分发中心（KDC）发生故障
- 缺少 Kerberos 或 OS 软件包或库
- 用于跨域身份验证的 Kerberos REALM 的错误映射

这些只是一些示例，但是它们可以阻止用户和服务进行身份验证，并且可以干扰集群运行和处理工作负载的能力。每当出现问题时，第一步就是尝试通过回答以下基本问题来隔离实际问题的来源：

- 该问题是本地问题还是全球性问题？也就是说，所有用户是否都无法通过身份验证，还是该问题仅针对单个用户？
- 问题是特定于单个服务还是所有服务都存在问题？等等。

如果所有用户和多个服务均受到影响（并且在与 Kerberos 集成以进行身份验证后集群仍然无法正常工作），请逐步完成 Kerberos 配置文件的所有设置，如以下“审核 Kerberos 配置”部分所述。

注意

为 Kerberos 配置的任何给定集群中的所有节点必须在分布于整个集群中的文件中使用相同的配置设置。

7.1.2.1. 审核 Kerberos 配置

Cloudera 建议在出现问题时特别是在最初完成集成过程之后，验证 Kerberos 配置。

- 验证所有`/etc/hosts`文件均符合 Cloudera Manager 的安装要求（“ Cloudera Enterprise 要求和支持的版本”）。
- 验证所有集群主机以及 MIT KDC 或 Active Directory KDC 主机的正向和反向名称解析。
- 验证是否已安装所有必需的 Kerberos 服务器和工作站软件包（“为 CDH 启用 Kerberos 身份验证”），并且它们是在主机系统上运行的操作系统的正确版本。
- 对于所有使用 Kerberos 的服务，请验证`core-site.xml`中的`hadoop.security.auth_to_local`属性 对*所有*受信任的 Kerberos 领域（包括 HDFS 受信任的领域）具有正确的映射。
- 通过比较下面显示的“示例 Kerberos 配置文件”（请参阅“/etc/krb5.conf”和“/var/kerberos/krb5kdc/kdc.conf”）来验证 Kerberos 配置。
- 查看`krb5.conf`和`kdc.conf`文件中引用的所有 KDC，REALM 和域主机的配置。特别是 KDC 主机是常见的故障点，您可能必须在此处开始故障排除。确保`krb5.conf`设置中 具有列出的正确的 KDC 主机名。对于跨域身份验证，请参阅“在跨域部署中查看服务票证凭证”。
- 使用使用 Kerberos 的服务是否正在运行并通过`kinit/klist`（“带有和不带有 Keytab 的用户身份验证”）正确响应。
- 尝试使用特定于问题或受影响服务的集群服务凭据向 Cloudera Manager 进行身份验证。如果您能够使用服务密钥表成功进行身份验证，请检查颁发的凭据。
- 使用`klist`列出 principal 提出一个服务 Keytab，以确保每个服务都有一个。
- 使用命令行或 Cloudera Manager 启用调试（“为身份验证过程启用调试”）。

7.1.2.2. 带和不带 Keytab 的用户身份验证

该`kinit`命令行工具用于给用户，服务，系统，或设备验证到 KDC。最基本的示例是用户使用用户名（principal）和密码向 Kerberos 进行身份验证。在以下示例中，第一次尝试使用错误的密码，然后第二次成功尝试。

```
[alice@host1 ~]$ kinit alice@TEST.ORG.LAB
Password for alice@TEST.ORG.LAB: (wrong password)
kinit: Preauthentication failed while getting initial credentials

[alice@host1 ~]$ kinit alice@TEST.ORG.LAB
Password for alice@TEST.ORG.LAB: (correct password)
(note silent return on successful auth)
[alice@host1 ~]$ klist
Ticket cache: FILE:/tmp/krb5cc_10001
Default principal: alice@TEST.ORG.LAB
```

```
Valid starting     Expires          Service principal
03/11/14 11:55:39  03/11/14 21:54:55  krbtgt/TEST.ORG.LAB@TEST.ORG.LAB
renew until 03/18/14 11:55:39
```

认证的另一种方法是在 `kinit` 命令中使用 keytabs 。您可以通过使用 `klist` 命令显示 KDC 颁发的凭据来验证身份验证是否成功。下面的示例尝试 `hdfs` 使用 `hdfs` keytab 文件向 KDC 验证服务。

```
[root@host1 312-hdfs-DATANODE]# kinit -kt hdfs.keytab hdfs/host1.test.lab@TEST.LAB
[root@host1 312-hdfs-DATANODE]# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: hdfs/host1.test.lab@TEST.LAB

Valid starting     Expires     Service principal
03/11/14 11:18:34 03/12/14 11:18:34 krbtgt/TEST.LAB@TEST.LAB
renew until 03/18/14 11:18:34
```

7.1.2.3. 使用 Cloudera Manager 进行调试

为了在日志中获取其他信息并促进故障排除，管理员可以为 Cloudera Manager Server 上运行的任何服务设置调试级别。通常，使用特定服务的高级配置代码段（安全阀）设置来添加设置，名称是特定于服务的。

至于 HDFS，如下所述：

1) 登录到 Cloudera Manager 管理控制台。
2) 选择集群> HDFS- *n*。
3) 单击配置选项卡。
4) 搜索特定于要为其启用调试的不同角色类型的属性。例如，如果要为 HDFS NameNode 启用调试，请搜索 **NameNode Logging Threshold** 属性并至少选择 `DEBUG` 级别日志记录。
5) 通过使用 HDFS 服务的“高级配置代码段”启用 Kerberos 调试。同样，对于每个特定角色类型或服务，这可能有所不同。对于 HDFS NameNode，将以下属性添加到 HDFS 服务环境安全阀：

```
HADOOP_JAAS_DEBUG=true
HADOOP_OPTS="-Dsun.security.krb5.debug=true"
```

6) 点击保存更改。
7) 重新启动 HDFS 服务。

输出将显示在过程日志 `stdout.log` 和 `stderr.log` 中。这些可以在实例的运行时路径中找到：

```
/var/run/cloudera-scm-agent/process/###-service-ROLE
```

重新启动 Cloudera Manager Service 后，该###-service-ROLE 目录的最新实例 将具有调试日志。在/var/run/cloudera-scm-agent/process 路径中使用 ls -ltr 以确定最新路径。

7.1.2.4. 为身份验证过程启用调试

在集群上设置以下属性，以从 Kerberos 身份验证过程中获取调试信息。

```
# export HADOOP_ROOT_LOGGER=TRACE,console;
# export HADOOP_JAAS_DEBUG=true;
# export HADOOP_OPTS="-Dsun.security.krb5.debug=true"
```

然后，您可以使用下面的命令来复制控制台输出给用户（使用调试信息），与所有输出一起 STDOUT，并 STDERR 到一个文件中。

```
# hadoop fs -ls/> >(tee fsls-logfile.txt) 2>&1
```

7.1.2.5. Kerberos 凭证产生问题

Cloudera Manager 使用内部命令（Generate Credentials）创建 CDH 服务所需的帐户，该内部命令由 Kerberos 配置向导自动触发，或者在对集群进行更改时需要新的 Kerberos 主体。

配置集群以进行 Kerberos 身份验证或进行需要生成新主体的更改之后，可以使用 Cloudera Manager 管理控制台验证命令是否成功运行，如下所示：

a) 登录到 Cloudera Manager 管理控制台。任何错误消息显示在 主页页面的状态靠近页面的顶部区域。以下状态消息指示“生成凭据”命令失败：

```
Role is missing Kerberos
keytab
```

b) 要显示命令的输出，请转到主页> 状态选项卡，然后单击所有最近的命令选项卡。

7.1.2.6. Active Directory 凭证产生错误

错误： ldap_sasl_interactive_bind_s: Can't contact LDAP server (-1)

可能的原因：指定的域控制器不正确或尚未为其启用 LDAPS。

解决步骤：验证 Active Directory Active Directory KDC 的配置，如下所示：

1) 登录到 Cloudera Manager 管理控制台。
2) 选择管理> 设置。

3) 为类别 过滤器选择 Kerberos。

验证所有设置。还要检查是否为 Active Directory 启用了 LDAPS。

错误： `ldap_add: Insufficient access (50)`

可能的原因：您用于 Cloudera Manager 的 Active Directory 帐户无权创建其他帐户。

解决步骤：使用 Delegate Control 向导向 Cloudera Manager 帐户授予创建其他帐户的权限。您还可以以 Cloudera Manager 用户身份登录到 Active Directory，以检查它是否可以在组织单位中创建其他帐户。

7.1.2.7. MIT Kerberos 凭证产生错误

错误： `kadmin: Cannot resolve network address for admin server in requested realm while initializing kadmin interface.`

可能的原因： KDC 服务器的主机名不正确。

解决步骤：检查 `kdc` 默认域的字段， `krb5.conf` 并确保主机名正确。

7.1.2.8. 启用 Kerberos 安全性后 Hadoop 命令失败

用户需要获取有效的 Kerberos 票证才能与安全集群（即已配置为使用 Kerberos 进行身份验证的集群）进行交互。`hadoop fs -ls` 如果您的凭据缓存中没有有效的 Kerberos 票证，则运行任何 Hadoop 命令（例如）都会失败。如果您没有有效的票证，则会收到以下错误消息：

```
11/01/04 12:08:12 WARN ipc.Client: Exception encountered while connecting to the server :
javax.security.sasl.SaslException:
GSS initiate failed [Caused by GSSException: No valid credentials provided (Mechanism level: Failed to find
any Kerberos tgt)]
Bad connection to FS. command aborted. exception: Call to nn-host/10.0.0.2:8020 failed on local exception:
java.io.IOException:
javax.security.sasl.SaslException: GSS initiate failed [Caused by GSSException: No valid credentials provided
(Mechanism level: Failed to find any Kerberos tgt)]
```

解决步骤：通过运行 `klist` 命令检查凭据高速缓存中当前存在的 Kerberos 票证。您可以通过运行 `kinit` 命令并指定包含凭据的密钥表文件或输入主体密码来获取票证 。

7.1.2.9. 使用 UserGroupInformation 类对 Oozie 进行身份验证

受保护的 CDH 服务主要使用 Kerberos 来验证 RPC 通信。RPC 是 Hadoop 集群中节点之间通信的主要方式之一。例如，YARN NodeManager 使用 RPC 与 ResourceManager 进行通信，或者 HDFS 客户端使用 RPC 与 NameNode 进行通信。

`loginUserFromKeytab()`每次服务启动时，CDH 服务都会通过调用 UserGroupInformation（UGI）登录方法来处理 Kerberos 身份验证。由于 Kerberos 票证的有效时间通常很短，因此需要重复登录才能确保应用程序的安全。必须相应地实现长时间运行的 CDH 应用程序，以适应这些重复的登录。如果应用程序仅要与 HDFS，YARN，MRv1 和 HBase 通信，则只需调用 `UserGroupInformation.loginUserFromKeytab()`启动任何实际 API 活动之前的方法。HDFS，YARN，MRv1 和 HBase 服务的 RPC 客户端具有自己的内置机制，可在密钥表的票据授予票证（TGT）过期时自动重新登录。因此，此类应用程序不需要包括对 UGI 重新登录方法的调用，因为它们的 RPC 客户端层为其执行重新登录任务。

但是，某些应用程序可能包括其他不涉及通用 Hadoop RPC 框架的服务客户端，例如 Hive 或 Oozie 客户端。`UserGroupInformation.getLoginUser().checkTGTAndReloginFromKeytab()`在每次尝试与 Hive 或 Oozie 客户端连接之前，此类应用程序必须显式调用该 方法。这是因为这些客户端没有自动重新登录所需的逻辑。

这是一个无限轮询 Oozie 客户端应用程序的示例：

```
//App startup
UserGroupInformation.loginFromKeytab(KEYTAB_PATH, PRINCIPAL_STRING);
OozieClient client = loginUser.doAs(new PrivilegedAction<OozieClient>() {
   public OozieClient run() {
      try {
         returnnew OozieClient(OOZIE_SERVER_URI);
      } catch (Exception e) {
         e.printStackTrace();
         returnnull;
      }
   }
});

while (true && client != null) {
  //Application's long-running loop

  //Every time, complete the TGT check first
```

```
  UserGroupInformation loginUser = UserGroupInformation.getLoginUser();
  loginUser.checkTGTAndReloginFromKeytab();

 //Perform Oozie client work within the context of the login user object
  loginUser.doAs(new PrivilegedAction<Void>() {
     publicVoid run() {
        try {
           List<WorkflowJob> list = client.getJobsInfo("");
           for (WorkflowJob wfJob : list) {
              System.out.println(wfJob.getId());
           }
        } catch (Exception e) {
           e.printStackTrace();
        }
     }//End of function block
  });//End of doAs
}//End of loop
```

7.1.2.10. *某些 Java 版本无法读取凭证缓存*

症状：对于 MIT Kerberos 1.8.1（或更高版本），即使您成功使用 `kinit` 以下方法获得 Kerberos 票证，尝试与 Hadoop 集群进行交互时，也会发生以下错误：

```
11/01/04 12:08:12 WARN ipc.Client: Exception encountered while connecting to the server :
javax.security.sasl.SaslException:
GSS initiate failed [Caused by GSSException: No valid credentials provided (Mechanism level: Failed to find
any Kerberos tgt)]
Bad connection to FS. command aborted. exception: Call to nn-host/10.0.0.2:8020 failed on local exception:
java.io.IOException:
javax.security.sasl.SaslException: GSS initiate failed [Caused by GSSException: No valid credentials provided
(Mechanism level: Failed to find any Kerberos tgt)]
```

可能的原因：

在 MIT Kerberos 的 1.8.1 版中，对与 Oracle JDK 6 Update 26（和更早的 JDK）相冲突的凭据缓存格式进行了更改（“＃6206：用于在 ccache 中存储额外的每个主要数据的新 API”）（有关详细信息，请参阅“ JDK-6979329：CCacheInputStream 无法从 Kerberos 1.8.1 读取票证高速缓存文件”），使 Java 无法读取由 MIT Kerberos 1.8.1（或更高版本）创建的 Kerberos 凭证高速缓存。Kerberos 1.8.1 是 Ubuntu Lucid 和更高版本以及 Debian Squeeze 和更高版本中的默认设置。在 RHEL 和 CentOS 上，默认版本的 MIT Kerberos 较早版本没有此问题。

解决方法： `kinit` 最初通过获得凭据后，请使用-R（续订）选项 `kinit`。此序列将导致更新 ticket，并使用 Java 可以读取的格式缓存 ticket。但是，初始 ticket 必须是可续签的。

例如：

```
$ klist
klist: No credentials cache found (ticket cache FILE:/tmp/krb5cc_1000)
$ hadoop fs -ls
11/01/04 13:15:51 WARN ipc.Client: Exception encountered while connecting to the server :
javax.security.sasl.SaslException:
GSS initiate failed [Caused by GSSException: No valid credentials provided (Mechanism level: Failed to find
any Kerberos tgt)]
Bad connection to FS. command aborted. exception: Call to nn-host/10.0.0.2:8020 failed on local exception:
java.io.IOException:
javax.security.sasl.SaslException: GSS initiate failed [Caused by GSSException: No valid credentials provided
(Mechanism level: Failed to find any Kerberos tgt)]
$ kinit
Password for username@REALM-NAME.EXAMPLE.COM:
$ klist
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: username@REALM-NAME.EXAMPLE.COM

Valid starting     Expires          Service principal
01/04/11 13:19:31  01/04/11 23:19:31  krbtgt/REALM-NAME.EXAMPLE.COM@REALM-NAME.EXAMPLE.COM
        renew until 01/05/11 13:19:30
$ hadoop fs -ls
11/01/04 13:15:59 WARN ipc.Client: Exception encountered while connecting to the server :
javax.security.sasl.SaslException:
GSS initiate failed [Caused by GSSException: No valid credentials provided (Mechanism level: Failed to find
any Kerberos tgt)]
Bad connection to FS. command aborted. exception: Call to nn-host/10.0.0.2:8020 failed on local exception:
java.io.IOException:
javax.security.sasl.SaslException: GSS initiate failed [Caused by GSSException: No valid credentials provided
(Mechanism level: Failed to find any Kerberos tgt)]
$ kinit -R
$ hadoop fs -ls
Found 6 items
drwx------   - user user     0 2011-01-02
      16:16/user/user/.staging
```

当命令不可更新时，票证显示此错误消息

```
kinit: Ticket expired while renewing credentials
```

7.1.2.11. *解决 Cloudera Manager 服务密钥表问题*

由 Cloudera Manager 管理的每个服务都有一个 keytab 文件，该文件由 Cloudera Manager 代理在启动时提供。可以通过`/var/run/cloudera-scm-agent/process` 使用 `ls -ltr` 命令导航到路径来检查最新的 keytab 文件。

如下面的示例所示，Cloudera Manager 服务目录名称的格式为：`###-service-ROLE`。因此，如果要对 HDFS 服务进行故障排除，则可以将服务目录称为 `326-hdfs-NAMENODE`。

```
[root@cehd1 ~]# cd/var/run/cloudera-scm-agent/process/
[root@cehd1 process]# ls -ltr | grep NAMENODE | tail -3
drwxr-x--x 3 hdfs       hdfs       4096 Mar  3 23:43 313-hdfs-NAMENODE
drwxr-x--x 3 hdfs       hdfs       4096 Mar  4 00:07 326-hdfs-NAMENODE
drwxr-x--x 3 hdfs       hdfs       4096 Mar  4 00:07 328-hdfs-NAMENODE-nnRpcWait

[root@cehd1 process]# cd 326-hdfs-NAMENODE

[root@cehd1 326-hdfs-NAMENODE]# ls
cloudera_manager_agent_fencer.py          dfs_hosts_allow.txt       hdfs.keytab              log4j.properties
topology.py
cloudera_manager_agent_fencer_secret_key.txt  dfs_hosts_exclude.txt      hdfs-site.xml            logs
cloudera-monitor.properties               event-filter-rules.json   http-auth-signature-secret
navigator.client.properties
core-site.xml                             hadoop-metrics2.properties  krb5cc_494               topology.map
```

如果您具有对该`/var/run/cloudera-scm-agent/process` 路径的超级用户访问权限，则可以使用任何服务的密钥表文件以超级用户或 sudo 用户身份登录以验证基本 Kerberos 身份验证是否正常工作。

找到密钥表文件后，使用以下 `klist` 命令检查其内容（“使用 klist 检查 Kerberos 凭据”）以查看存储在文件中的凭据。例如，列出存储在 `hdfs.keytab` 文件中的凭据：

```
[root@host1 326-hdfs-DATANODE]# klist -kt hdfs.keytab

Keytab name: WRFILE:hdfs.keytab
KVNO Timestamp         Principal
---- ----------------- --------------------------------------------------------
 4 02/17/14 19:09:17 HTTP/host1.test.lab@TEST.LAB
 4 02/17/14 19:09:17 HTTP/host1.test.lab@TEST.LAB
 4 02/17/14 19:09:17 HTTP/host1.test.lab@TEST.LAB
 4 02/17/14 19:09:17 HTTP/host1.test.lab@TEST.LAB
 4 02/17/14 19:09:17 HTTP/host1.test.lab@TEST.LAB
 4 02/17/14 19:09:17 HTTP/host1.test.lab@TEST.LAB
 4 02/17/14 19:09:17 hdfs/host1.test.lab@TEST.LAB
 4 02/17/14 19:09:17 hdfs/host1.test.lab@TEST.LAB
```

```
4 02/17/14 19:09:17 hdfs/host1.test.lab@TEST.LAB
4 02/17/14 19:09:17 hdfs/host1.test.lab@TEST.LAB
4 02/17/14 19:09:17 hdfs/host1.test.lab@TEST.LAB
4 02/17/14 19:09:17 hdfs/host1.test.lab@TEST.LAB
```

现在，尝试使用 keytab 文件及其中的主体进行身份验证。在这种情况下，我们将 hdfs.keytab 文件与 hdfs/host1.test.lab@TEST.LAB 主体一起使用。然后，使用 klist 不带任何参数的 命令来查看当前用户会话的凭据。

```
root@host1 312-hdfs-DATANODE]# kinit -kt hdfs.keytab hdfs/host1.test.lab@TEST.LAB
[root@host1 312-hdfs-DATANODE]# klist

Ticket cache: FILE:/tmp/krb5cc_0
Default principal: hdfs/host1.test.lab@TEST.LAB

Valid starting     Expires          Service principal
03/11/14 11:18:34  03/12/14 11:18:34  krbtgt/TEST.LAB@TEST.LAB
renew until 03/18/14 11:18:34
```

请注意，Kerberos 凭证具有到期日期和时间。这意味着，为了确保 Kerberos 凭据在整个集群上统一有效，集群中的所有主机和客户端都应使用 NTP，并且彼此之间的距离不得超过 5 分钟。Kerberos 会话票证的寿命有限，但是可以续签（如示例 krb5.conf 和中所示 kdc.conf）。CDH 需要集群主体的可更新票证。在检查 Kerberos 会话和凭据时，使用 klist 命令和-e（列表密钥加密类型）和 -f（设置列表标志）开关来检查是否已启用可再生票证。

7.1.2.12. 在跨领域部署中查看服务票证凭证

检查集群配置时，请确保您没有违反以下任何集成规则：

- 在协商加密类型时，请按照领域对支持的加密类型进行最具体的限制。
- 所有领域都应该通过/etc/krb5.conf 部署在集群上的文件相互了解。
- 在为 Active Directory 环境做出配置决策时，必须评估存在的域功能级别或 Forrest 功能级别。

Kerberos 通常会在客户端和服务器之间协商并使用可能的最强加密形式，以对领域进行身份验证。但是，TGT 的加密类型有时可能最终会朝较弱的加密类型向下协商，这是不希望的。要调查此类问题，请按照以下步骤中所述检查 kvno 跨域信任委托人（krbtgt）。用 CLUSTER.REALM 和 AD.REALM（或 MIT.REALM）替换为您配置的领域的适当值。此方案假定使用 Active Directory 进行跨域身份验证。

- kinit 通过对 AD Kerberos 领域进行身份验证后，以系统用户身份配置了信任

（请参阅上一节中的示例文件）。

- 在命令行中，kvno 检查本地和跨领域的 krbtgt 输入。该特殊 REALM 服务主体的本地表示 形式为 krbtgt/CLUSTER.REALM@CLUSTER.REALM。跨域主体以可信任域的形式命名，krbtgt/AD.REALM。
- kvno 检查失败表示跨域信任配置不正确。再次检查加密类型，以查找在领域之间配置的不兼容或不受支持的加密类型。

7.1.2.13. *Kerberos 配置文件样例*

本节包含几个示例 Kerberos 配置文件。

7.1.2.13.1. */etc/krb5.conf*

该/etc/krb5.conf 文件是客户端用来通过其配置的 KDC 访问领域的配置。该 krb5.conf 映射将领域映射到支持这些领域的可用服务器。它还定义了如何请求和授予票证的特定于主机的配置规则。

```
[logging]
 default = FILE:/var/log/krb5libs.log
 kdc = FILE:/var/log/krb5kdc.log
 admin_server = FILE:/var/log/kadmind.log

[libdefaults]
 default_realm = EXAMPLE.COM
 dns_lookup_realm = false
 dns_lookup_kdc = false
 ticket_lifetime = 24h
 renew_lifetime = 7d
 forwardable = true
# udp_preference_limit = 1

# set udp_preference_limit = 1 when TCP only should be
# used. Consider using in complex network environments when
# troubleshooting or when dealing with inconsistent
# client behavior or GSS (63) messages.

# uncomment the following if AD cross realm auth is ONLY providing DES encrypted tickets
# allow-weak-crypto = true

[realms]
 AD-REALM.EXAMPLE.COM = {
  kdc = AD1.ad-realm.example.com:88
  kdc = AD2.ad-realm.example.com:88
  admin_server = AD1.ad-realm.example.com:749
  admin_server = AD2.ad-realm.example.com:749
  default_domain = ad-realm.example.com
 }
 EXAMPLE.COM = {
```

```
 kdc = kdc1.example.com:88
 admin_server = kdc1.example.com:749
 default_domain = example.com
}

# The domain_realm is critical for mapping your host domain names to the kerberos realms
# that are servicing them. Make sure the lowercase left hand portion indicates any domains or subdomains
# that will be related to the kerberos REALM on the right hand side of the expression. REALMs will
# always be UPPERCASE. For example, if your actual DNS domain was test.com but your kerberos REALM is
# EXAMPLE.COM then you would have,

[domain_realm]
test.com = EXAMPLE.COM
#AD domains and realms are usually the same
ad-domain.example.com = AD-REALM.EXAMPLE.COM
ad-realm.example.com = AD-REALM.EXAMPLE.COM
```

样本 Kerberos 配置文件

7.1.2.13.2. /var/kerberos/krb5kdc/kdc.conf

该 `kdc.conf` 文件仅需要在实际的集群专用 KDC 上进行配置，并且应位于`/var/kerberos/krb5kdc`。只有主要和辅助 KDC 才需要访问此配置文件。该文件的内容建立了对中所有客户端主机强制执行的配置规则 `REALM`。

```
[kdcdefaults]
 kdc_ports = 88
 kdc_tcp_ports = 88

[realms]
 EXAMPLE.COM = {
 #master_key_type = aes256-cts
 max_renewable_life = 7d 0h 0m 0s
 acl_file =/var/kerberos/krb5kdc/kadm5.acl
 dict_file =/usr/share/dict/words
 admin_keytab =/var/kerberos/krb5kdc/kadm5.keytab
# note that aes256 is ONLY supported in Active Directory in a domain/forrest operating at a 2008 or greater
functional level.
# aes256 requires that you download and deploy the JCE Policy files for your JDK release level to provide
# strong java encryption extension levels like AES256. Make sure to match based on the encryption configured
within AD for
# cross realm auth, note that RC4 = arcfour when comparing windows and linux enctypes
 supported_enctypes = aes256-cts:normal aes128-cts:normal arcfour-hmac:normal
 default_principal_flags = +renewable, +forwardable
}

[logging]
   kdc = FILE:/var/log/krb5kdc.log
   admin_server = FILE:/var/log/kadmin.log
```

7.1.2.13.3. kadm5.acl

```
*/admin@HADOOP.COM *
cloudera-scm@HADOOP.COM * flume/*@HADOOP.COM
cloudera-scm@HADOOP.COM * hbase/*@HADOOP.COM
cloudera-scm@HADOOP.COM * hdfs/*@HADOOP.COM
cloudera-scm@HADOOP.COM * hive/*@HADOOP.COM
cloudera-scm@HADOOP.COM * httpfs/*@HADOOP.COM
cloudera-scm@HADOOP.COM * HTTP/*@HADOOP.COM
cloudera-scm@HADOOP.COM * hue/*@HADOOP.COM
cloudera-scm@HADOOP.COM * impala/*@HADOOP.COM
cloudera-scm@HADOOP.COM * mapred/*@HADOOP.COM
cloudera-scm@HADOOP.COM * oozie/*@HADOOP.COM
cloudera-scm@HADOOP.COM * solr/*@HADOOP.COM
cloudera-scm@HADOOP.COM * sqoop/*@HADOOP.COM
cloudera-scm@HADOOP.COM * yarn/*@HADOOP.COM
cloudera-scm@HADOOP.COM * zookeeper/*@HADOOP.COM
```

7.1.3. HDFS 加密问题

对于加密 HDFS 目录和文件时可能出现的问题，以下是可能的解决方法。HDFS 加密有时在文档中称为 HDFS 透明加密或静态加密 HDFS 数据。

7.1.3.1. KMS 服务器黄麻缓冲区异常

说明：当 KMS（例如，作为 ZooKeeper 客户端）jute 缓冲区大小不足以容纳所有令牌时，您会看到以下错误：

```
2017-01-31 21:23:56,416 WARN org.apache.zookeeper.ClientCnxn: Session 0x259f5fb3c1000fb for server
example.cloudera.com/10.172.0.1:2181, unexpected error, closing socket connection and attempting
reconnect
java.io.IOException: Packet len4196356 is out of range!
```

解决方案：增加 jute 缓冲区的大小，然后重新启动 KMS。在 Cloudera Manager 中，转到 KMS **配置**页面，然后在 KMS 的**其他 Java 配置选项**（`kms_java_opts`）字段中输入 `-Djute.maxbuffer=<number_of_bytes>`。重新启动 KMS。

7.1.3.2. 检索加密密钥失败

说明：尝试列出加密密钥时会看到以下错误

```
user1@example-sles-4:~> hadoop key list
Cannot list keys for KeyProvider: KMSClientProvider[https://example-sles-2.example.com:16000/kms/v1/]:
Retrieval of all keys failed.
```

解决方案：确保您的信任库已使用相关证书（例如，密钥受托者服务器证书）进行了更新。

7.1.3.3. 未加密位置和加密位置之间的 DistCp 失败

说明：默认情况下，DistCp 比较文件系统提供的校验和，以验证数据已成功复制到目标。但是，在未加密位置和加密位置之间进行复制时，由于基础块数据不同，因此文件系统校验和将不匹配。

解决方案：指定-skipcrccheck 和-update distcp 标志以避免验证校验和。

7.1.3.4. NameNode-长时间不活动后，KMS 通信失败

说明：如果自 KMS 和 NameNode 上次通信以来经过了很长时间（默认为 20 小时），则无法创建加密文件和加密区域。

解决方案：将集群升级到 CDH 6 将解决此问题。有关说明，请参阅“升级 CDH 集群”。

7.1.3.5. 启用了透明加密的 HDFS Trash 行为

Hadoop 垃圾回收功能有助于防止意外删除文件和目录。当您在 HDFS 中删除文件时，该文件不会立即从 HDFS 中删除。首先将已删除的文件移至该/user/<username>/.Trash/Current 目录，并保留其原始文件系统路径。在用户可配置的时间段（fs.trash.interval）之后，称为垃圾桶检查点的过程会将 Current 目录重命名 为当前时间戳，即/user/<username>/.Trash/<timestamp>。检查点过程还将检查.Trash 目录的其余部分是否存在任何现有的时间戳目录，并将它们从 HDFS 中永久删除。您只需将文件和目录移动到.Trash 目录外的某个位置，就可以还原它们。

7.1.3.6. 启用 HDFS 透明加密的垃圾桶行为

从 CDH 5.7 开始，您可以删除 HDFS 加密区域中的文件或目录。从上述过程可以明显看出，移动和重命名文件或目录是 HDFS 中垃圾处理的重要组成部分。但是，当前 HDFS 透明加密仅支持加密区域*内*的重命名。为了解决这个问题，HDFS .Trash 每次创建新的加密区域时都会创建一个本地目录。例如，当您创建一个加密区域时 enc_zone，HDFS 还将创建/enc_zone/.Trash/子目录。从中删除的文件 enc_zone 将移至/enc_zone/.Trash/<username>/Current/。在检查点之后， Current 目录被重命名为当前时间戳/enc_zone/.Trash/<username>/<timestamp>。

如果删除整个加密区域，它将被移动到.Trash 用户主目录下的目录中/users/<username>/.Trash/Current/enc_zone。只有将整个区域移到时，才会进行垃圾检查点/users/<username>/.Trash。但是，如果用户的主目录已经是加密区域的一部分，则尝试删除加密区域将失败，因为您无法跨加密区域移动或重命名目录。

7.1.4. Key Trustee KMS 加密问题

以下错误和条件与 Key Trustee KMS 有关，并且包括使用 Key Trustee KMS 时可能出现的问题的变通办法。

7.1.4.1. Key Trustee KMS 升级后无法重新启动（仅 HA ）

描述：升级后尝试重新启动 Key Trustee KMS HA 主机后，可能会看到以下错误：

```
java.io.IOException: Unable to verify private key match between KMS
hosts. Verify private key files have been synced between all KMS
hosts. Aborting to prevent data inconsistency.
```

解决方案： 如果您未能在 Key Trustee KMS 主机之间同步私钥，则它们可能处于间歇性无法访问密钥的状态，具体取决于客户端与之交互的 Key Trustee KMS 主机，因为由一个 Key Trustee KMS 加密的加密密钥材料主机不能被其他人解密。如果您已经在运行具有不同私钥的多个 Key Trustee KMS 主机，请立即备份所有 Key Trustee KMS 主机，并联系 Cloudera 支持以获取纠正问题的帮助。

7.1.4.2. Key Trustee KMS 在首次运行时失败（仅 HA ）

说明：添加新的密钥受托者 KMS HA 服务后，首次运行时可能会看到以下错误：

```
java.io.IOException: Unable to verify private key match between KMS
hosts. Verify private key files have been synced between all KMS
hosts. Aborting to prevent data inconsistency.
```

解决方案：有关同步和验证私钥的指导，请参阅“升级密钥受托人 KMS”。

Cloudera 建议遵循最佳安全性最佳做法，并使用脱机媒体（例如可移动 USB 驱动器）传输私钥。为了方便起见（例如，在不需要最高安全性的开发或测试环境中），可以通过 rsync 在原始 Key Trustee KMS 主机上运行以下命令来通过网络复制私钥：

```
rsync -zav/var/lib/kms-keytrustee/keytrustee/.keytrustee
```

```
root@ktkms02.example.com:/var/lib/kms-keytrustee/keytrustee/.
```

用 ktkms02.example.com 您要添加的 Key Trustee KMS 主机的主机名替换。

笔记

rsync 仅在首次运行 Key Trustee KMS 时执行命令；如果密钥已经创建并且数据已加密，请不要输入此命令。

7.1.4.3. Key Trustee KMS 无法启动，因为 ZooKeeper 没有运行

说明：首次尝试重新启动密钥受托者 KMS 后，您可能会看到以下错误：

```
java.lang.Exception: Could not establish connection to ZooKeeper to
verify KMS host private key consistency. Verify private key files have
been synced between all KMS hosts. Aborting to prevent data
inconsistency.
```

解决方案： ZooKeeper 用于与主机进行通信，并且也是私钥数据的存储位置，因此必须在首次重新启动或运行 GPG 验证检查时运行，该检查会比较 Key Trustee KMS 主机之间的私钥以防止“脑裂”方案（当主机之间的私钥未同步时）。为了确保可以运行 GPG 验证检查，请启动 ZooKeeper，然后重新启动 Key Trustee KMS。

7.1.5. 对 Cloudera Manager 中的 TLS/SSL 问题进行故障排除

要诊断和解决与 TLS/SSL 配置有关的问题，请通过验证“为 Cloudera Manager 手动配置 TLS 加密”中的步骤来验证适用于集群的配置任务。

检查设置并没有发现明显的错误配置后，请尝试以下故障排除步骤。

7.1.5.1. 使用 OpenSSL 测试连接

从存在连接问题的主机，openssl 如下所示运行。您还可以在 TLS/SSL 协商期间检查主机使用的证书是否被信任的 CA 识别。

要检查连接：

```
openssl s_client -connect [host.fqdn.name]:[port]
```

例如：

```
openssl s_client -connect test1.sec.cloudera.com:7183
```

返回码 0 意味着 openssl 能够通过其受信任的公共 CA 库遵循信任服务器的信任链。

对于由组织的内部 CA 签名的证书或自签名证书，您可能需要使用以下 openssl 命令将证书添加到信任库中。使用该-CAfile 选项来指定根 CA 的路径，以便 openssl 可以如下所示验证自签名或内部 CA 签名的证书：

```
$ openssl s_client -connect test1.sec.cloudera.com:7183 -CAfile \
/opt/cloudera/security/CAcerts/RootCA.pem
```

只需要根 CA 证书即可建立此测试的信任。当您看到 0 如下所示的返回代码时，该命令的结果成功：

```
...
 Verify return code: 0 (ok)
---
```

默认情况下，Cloudera Manager Server/etc/cloudera-scm-server/cloudera-scm-server.log 在启动时会将日志写入 文件。使用证书成功启动服务器看起来类似于以下日志示例：

```
2014-10-06 21:33:47,515 INFO WebServerImpl:org.mortbay.log: jetty-
6.1.26.cloudera.2
2014-10-06 21:33:47,572 INFO WebServerImpl:org.mortbay.log: Started
SslSelectChannelConnector@0.0.0.0:7183
2014-10-06 21:33:47,573 INFO WebServerImpl:org.mortbay.log: Started
SelectChannelConnector@0.0.0.0:7180
2014-10-06 21:33:47,573 INFO
WebServerImpl:com.cloudera.server.cmf.WebServerImpl: Started Jetty
server.
```

7.1.5.2. *将诊断包上传到 Cloudera 支持*

默认情况下，Cloudera Manager 通过 HTTPS 将诊断捆绑包上传到位于的 Cloudera 支持服务器 cops.cloudera.com。但是，如果 Cloudera Manager 信任库无法验证 Cloudera 支持服务器证书的真实性，则上传可能会失败，并且由于 Cloudera Manager 信任库配置问题，该验证过程可能会失败。

为确保 Cloudera Manager 服务器信任库包含验证 Cloudera 支持的证书所需的公共 CA，您可以通过将 Cloudera 支持的证书导入到 Cloudera Manager 的信任库中来显式建

立信任（请参阅“将 Cloudera Support 的证书导入到 Cloudera Manager 服务器信任库中”）。

笔记

Cloudera 支持服务器使用由商业 CA 签名的证书，因此通常不需要此步骤，除非已更改默认信任库。在下载或添加任何证书之前，请测试连接（请参阅“使用 OpenSSL 测试连接”），并确认证书是连接问题的根源。

7.1.5.3. 将 Cloudera 支持人员的证书导入 Cloudera Manager 服务器信任库

要从 Cloudera 支持服务器获取 Cloudera 的公钥证书：

```
openssl s_client -connect cops.cloudera.com:443 | openssl x509 -text -
out/path/to/cloudera-cert.pem
```

要将证书导入到 Cloudera Manager 信任库中（使用您自己的系统的路径）：

```
keytool -import -keystore/path/to/cm/truststore.jks -
file/path/to/cloudera-cert.pem
```

导入证书后，请确认已为此信任库文件配置了 Cloudera Manager，如“配置 Cloudera Manager 信任库属性”中所述。

笔记

或者，您可以将默认的 Java 信任库用于 Cloudera Manager 集群部署，如“为 Cloudera Manager 手动配置 TLS 加密”中所述。

7.1.5.4. 配置 Cloudera Manager 信任库属性

将 Cloudera 支持服务器证书安装到 Cloudera Manager 信任库中之后，必须将 Cloudera Manager 配置为使用信任库，如下所示：

1) 登录到 Cloudera Manager 管理控制台。
2) 选择管理>设置。
3) 单击**安全**类别。

4) 输入信任库的路径和密码（如果需要）：

环境	描述
Cloudera Manager TLS/SSL 证书信任存储文件	输入信任库（trust .jks）的完整 Cloudera Manager Server 主机文件系统路径。Cloudera Manager Server 调用 JVM-Djavax.net.ssl.trustStore 来访问指定的信任库。
Cloudera Manager TLS/SSL 证书信任库密码	指定信任库文件的密码（如果有）。访问信任库不需要密码，因此通常可以将该字段保留为空白。-Djavax.net.ssl.trustStore.password 如果此字段具有条目，则 Cloudera Manager Server 会调用 JVM 。

5) 单击保存更改以保存设置。

笔记

有关 JSSE 信任机制的更多信息，请参见 Oracle 的“ JSSE 参考指南”。

7.2. YARN、MRv1 和 Linux OS 安全性

这几个子系统是 Hadoop 集群的基础，特别是 Jsvc、TaskController 和 Container Executor 程序，在下面介绍。

7.2.1. MRv1 和 YARN：jsvc 程序

jsvc 是 Hadoop Apache Commons 库的一部分，旨在使 Java 应用程序在 Linux 上更好地运行。该 jsvc 程序是 bigtop-jsvc 软件包的一部分， 并安装在 Linux 的/usr/lib/bigtop-utils/jsvc 或/usr/libexec/bigtop-utils/jsvc 下，这取决于 Linux 版本。

特别是，jsvc 用于启动 DataNode 侦听低端口号。它的入口是 SecureDataNodeStarter 类，该类实现所需的 Daemon 接口 jsvc。jsvc 以 root 身份运行，并以 root 身份运行时调用该 SecureDataNodeStarter.init(...)方法。一旦 SecureDataNodeStarter 类完成初始化，jsvc 设置有效的 UID 是 hdfs 用户，然后调用 SecureDataNodeStarter.start(...)。 SecureDataNodeStarter 然后调用常规 DataNode 入口点，传递对其先前获得的特权资源的引用。

7.2.2. 仅限 MRv1：Linux TaskController

setuid 二进制文件 `task-controller` 是该 `hadoop-0.20-mapreduce` 软件包的一部分， 并安装在`/usr/lib/hadoop-0.20-mapreduce/sbin/Linux-amd64-64/task-controller` 或中`/usr/lib/hadoop-0.20-mapreduce/sbin/Linux-i386-32/task-controller`。

该 `task-controller` 程序仅在 MRv1 上使用，它允许 TaskTracker 以首先提交作业的用户的 Unix 帐户运行任务。它是 setuid 二进制文件，必须具有非常特定的一组权限和所有权才能正常运行。特别是，它必须：

1) 归 root 用户所有
2) 由仅包含运行 MapReduce 守护程序的用户的组所拥有
3) setuid
4) 具有组可读性和可执行性

这对应于所有权 `root:mapred` 和权限 `4754`。

这是在 `ls` 正确配置的任务控制器上的输出：

```
-rwsr-xr-- 1 root mapred 30888 Mar 18 13:03 task-controller
```

TaskTracker 将在启动时检查此配置，如果 Task-controller 的配置不正确，将无法启动。

7.2.3. 仅限 YARN：Linux 容器执行器

名为 setuid 的二进制文件 `container-executor` 是该 `hadoop-yarn` 软件包的一部分， 并安装在`/usr/lib/hadoop-yarn/bin/container-executor`。

这 `container-executor` 该程序仅在 YARN 上使用，并且仅在 GNU/Linux 上受支持，该程序以提交应用程序的用户身份运行容器。它要求在启动容器的集群主机上创建所有用户帐户。它使用 Hadoop 发行版中包含的 setuid 可执行文件。NodeManager 使用此可执行文件来启动和终止容器。setuid 可执行文件切换到提交了应用程序的用户，并启动或杀死容器。为了获得最大的安全性，该执行程序设置了容器使用的本地文件和目录（例如共享对象、jar、中间文件和日志文件）的受限权限和用户/组所有权。

Parcel 部署

在 parcel 部署中，`container-executor` 文件位于 parcel 内部，位于`/opt/cloudera/parcels/CDH/lib/hadoop-yarn/bin/container-executor`。对于`/usr/lib` 挂载点，setuid 应该不是问题。但是，parcel 可以轻松地位于其他安装点上。如果使用的是 parcel，请确保 parcel 目录的挂载点没有 nosuid 选项。

该 `container-executor` 程序必须具有一组非常特定的权限和所有权才能正常运行。特别是，它必须：

1) 归 root 用户所有
2) 由仅包含运行 YARN 守护程序的用户的组所拥有
3) setuid
4) 具有组可读性和可执行性。这对应于所有权 `root:yarn` 和权限 `6050`。

```
---Sr-s--- 1 root yarn 91886 2012-04-01 19:54 container-executor
```

重要

对 Linux 容器执行器的配置更改可能会导致本地 NodeManager 目录（例如 `usercache`）具有不正确的权限。为避免这种情况，在使用 Cloudera Manager 或命令行进行更改时，请首先从所有已配置的本地目录（`yarn.nodemanager.local-dirs`）中手动删除现有的 NodeManager 本地目录，然后让 NodeManager 重新创建目录结构。

7.2.3.1. 故障排除

首次设置安全集群并调试它的问题时， `task-controller` 或 `container-executor` 可能会遇到错误。这些程序通过分别出现在 TaskTracker 或 NodeManager 日志（`/var/log/hadoop-mapreduce` 或`/var/log/hadoop-yarn`）中的数字错误代码将这些错误传达给 TaskTracker 或 NodeManager 守护程序。

7.3. 对 Impala 进行故障排除

本主题描述了一般的故障排除过程，以诊断 Impala 中的一些常见问题。

症状	解释	推荐
Impala 花费很长时间才能启动。	具有大量表、分区或数据文件的 Impala 实例需要更长的启动时间，因为这些对象的元数据会广播到所有 impalad 节点并进行缓存。	调整超时和同步设置。
Join 失败	可能内存不足。在 Join 期间，来自第二、第三等要连接的数据集的数据将加载到内存中。如果 Impala 选择低效的连接顺序或连接机制，则查询可能会超出可用的总内存。	首先收集统计信息，并 COMPUTE STATS 获取联接中涉及的每个表的语句。 考虑指定 [SHUFFLE] 提示，以便将联接表中的数据在节点之间分配，而不是广播到每个节点。如果在 SQL 级别进行调优还不够，请向系统中添加更多内存或加入较小的数据集。
查询返回不正确的结果。	在 Hive 中执行更改后，Impala 元数据可能已过时。	在 Hive 中插入数据、添加分区或其他操作之后，使用 REFRESH 语句刷新表的元数据 。
尝试完成 Impala 任务（例如执行 INSERT SELECT 语句）失败。Impala 日志中包含说明，由于权限被拒绝，无法打开文件。	这可能是权限问题的结果。例如，您可以将 Hive shell 用作 hive 用户来创建表。创建此表后，您可以尝试完成某些操作，例如 INSERT SELECT 在表上执行操作。由于该表是使用一个用户创建的，而另一个用户 INSERT SELECT 则尝试使用该表 ，因此由于权限问题，此操作可能会失败。	确保 Impala 用户对 Hive 用户创建的表具有足够的权限。
Impala 无法启动，impalad 日志引用连接到 statestored 服务的错误并尝试重新注册。	大量的数据库、表、分区等可能需要元数据同步，尤其是在启动时，它所花费的时间比该 statestore 服务的默认超时长 。	配置 statestore 超时值以及可能与 statestore 更新和元数据加载频率相关的其他设置 。

7.3.1. 使用 Breakpad Minidumps 进行崩溃报告

Breakpad 项目是一个用于崩溃报告的开源框架。breakpad 当任何与 Impala 相关的守护程序由于诸如错误 SIGSEGV 或未处理的异常之类的错误而崩溃时，Impala 可用于记录堆栈信息并注册值。转储文件比传统的核心转储文件小得多。转储机制本身使用的内存很少，如果在系统内存不足的情况下发生崩溃，则可以提高可靠性。

7.3.1.1. 使用 Minidump 文件解决问题

当发生崩溃事件并生成小型转储文件时，您可以在 Impala 日志文件中或 Impala 的 Cloudera Manager 图表中查看。因为每次重新启动都会开始一个新的日志文件，所以崩溃的消息始终位于日志文件的底部或底部。（如果还启用了核心转储，则稍后可能会再显示一条消息。)

重要

如果 Impala 相关守护进程经历了碰撞，由于内存外的一个条件，它并*没有*生成该错误转储。

7.3.1.2. 使用 Minidump 文件解决问题

通常，您会以提供核心转储的相同方式，将 Minidump 文件作为问题解决的一部分提供给 Cloudera 支持。Cloudera Manager 中“支持”菜单下的“发送诊断数据”将 指导您完成选择诊断数据的时间段和量的过程，然后从所有主机收集数据并为您传输相关信息。

1) 在 Cloudera Manager 中，导航到 Impala 服务>配置。
2) 在搜索字段中，输入 `minidump`。
3) 设置以下字段以配置 Breakpad 小型转储。
   - minidump_path：打开或关闭 minidump 文件的生成。
     默认情况下，与 Impala 相关的守护程序崩溃时会生成一个小型转储文件。
   - minidump_path：指定小型转储文件的位置。
     默认情况下，所有 minidump 文件都写入发生崩溃的主机上的以下位置：
     `/var/log/impala-minidumps/daemon_name`
     `Impalad`、 `catalogated` 和 `statestored` 的 minidump 文件每个都写入一个单独的目录中。
     如果为此设置指定相对路径，则相对于默认 minidump_path 目录解释该值 。

- max_minidumps：指定小型转储文件的数量。

  与用于日志记录或故障排除的任何文件一样，请考虑限制小型转储文件的数量，或删除不需要的文件，具体取决于集群中主机上的可用存储空间量。

  由于 minidump 文件仅用于解决问题，因此您可以删除调试当前问题不需要的任何此类文件。

  此设置的默认值为 9。零或负值将解释为无限制。

4) 单击“保存更改”，然后重新启动 Impala。
5) 要将小型转储文件作为问题解决的一部分提供给 Cloudera 支持，请在 Cloudera Manager 中导航至“ 支持” >“发送诊断数据”。

7.4. 对 Apache Yarn 进行故障排查

7.4.1. 在 YARN 上对 Docker 进行故障排除

YARN 相关问题的常见 Docker 列表以及如何解决它们。

7.4.1.1. Docker 未启用

问题陈述

在 Docker 上启动了一个应用程序，但是这些容器正在作为常规容器运行。

根本原因

未启用 Docker。

解决

在 Cloudera Manager 中启用 Docker。

7.4.1.2. YARN_CONTAINER_RUNTIME_TYPE 提交应用程序期间未提供运行时环境变量

问题陈述

在 Docker 上启动了一个应用程序，但是这些容器正在作为常规容器运行。

根本原因

YARN_CONTAINER_RUNTIME_TYPE 提交应用程序期间未提供运行时环境变量。

解决

提交应用程序时提供环境变量。

7.4.1.3. LCE 强制正在运行的用户在不安全的集群中成为 nobody

问题陈述

在不安全的集群上，Appattempt 使用带有诊断消息的 exitCode -1000 退出：

```
[...]
main : run as user is nobody
main : requested yarn user is yarn
Can't create
directory/yarn/nm/usercache/yarn/appcache/application_1570626013274_00
01 - Permission denied
```

根本原因

如果 yarn.nodemanager.linux-container-executor.nonsecure-mode.limit-user s 设置了 LCE，则在不安全的集群中，将强制运行用户成为任 nobody。

解决

在 Cloudera Manager 中， 通过单击加号图标，将以下配置添加到 yarn-site.xml 安全阀的 YARN 服务高级配置代码片段（安全阀）中：

- key： yarn.nodemanager.linux-container-executor.nonsecure-mode.limit-users
- value： false

然后，使用具有正确权限的用户，或为 nobody 用户添加对这些文件夹的更多允许访问。有关更多信息，请参见 YARN 强制任何用户都不执行所有作业。

7.4.1.4. 找不到 Docker 二进制文件

问题陈述

容器启动失败，并显示以下消息：

```
Container launch fails
Exit code: 29
Exception message: Launch container failed
Shell error output: sh: <docker binary path,/usr/bin/docker by
default>: No such file or directory
Could not inspect docker network to get type/usr/bin/docker network
inspect host --format='{{.Driver}}'.
Error constructing docker command, docker error code=-1, error
message='Unknown error'
```

根本原因

找不到 Docker 二进制文件。

解决

Docker 二进制文件未安装或已安装到其他文件夹。安装 Docker 二进制文件并通过使用 Cloudera Manager 中的属性 Docker Binary Path（`docker.binary`）指定它来提供二进制文件的路径。

7.4.1.5. Docker 守护程序未运行或未响应

问题陈述

容器启动失败，并显示以下消息：

```
[timestamp] Exception from container-launch.
Container id: container_e06_1570629976081_0004_01_000003
Exit code: 29
Exception message: Launch container failed
Shell error output: Cannot connect to the Docker daemon at
unix:///var/run/docker.sock. Is the docker daemon running?
Could not inspect docker network to get type/usr/bin/docker network
inspect host --format='{{.Driver}}'.
Error constructing docker command, docker error code=-1, error
message='Unknown error'
```

根本原因

Docker 守护程序未运行或未响应。

解决

使用 `dockerd` 命令启动或重新启动 Docker 守护程序。

7.4.1.6. Docker rpm 缺少一些符号链接

问题陈述

在 Centos 7.5 容器上启动失败，并显示以下消息：

```
[...]
[layer hash]: Pull complete
[layer hash]: Pull complete
Digest: sha256:[sha]
Status: Downloaded newer image for [image]
/usr/bin/docker-current: Error response from daemon: shim error:
docker-runc not installed on system.
```

根本原因

Docker rpm 缺少一些符号链接。

解决

在终端中使用以下命令创建缺少的符号链接：

```
sudo ln -s/usr/libexec/docker/docker-runc-current/usr/bin/docker-runc
```

7.4.1.7. YARN_CONTAINER_RUNTIME_DOCKER_IMAGE 没有设置

问题陈述

容器启动失败，并显示以下消息：

```
[timestamp]Exception from container-launch.
Container id: container_e06_1570629976081_0004_01_000003
Exit code: -1
Exception message: YARN_CONTAINER_RUNTIME_DOCKER_IMAGE not set!
Shell error output: <unknown>
Shell output: <unknown>
```

根本原因

`YARN_CONTAINER_RUNTIME_DOCKER_IMAGE` 未设置。

解决

`YARN_CONTAINER_RUNTIME_DOCKER_IMAGE` 提交应用程序时设置环境变量。

7.4.1.8. Image 不受信任

问题陈述

容器启动失败，并显示以下消息：

```
[timestamp] Exception from container-launch.
Container id: container_e06_1570629976081_0004_01_000003
Exit code: 127
Exception message: Launch container failed
Shell error output: image: [image] is not trusted.
Disable mount volume for untrusted image
image: library/ibmjava:8 is not trusted.
Disable cap-add for untrusted image
Docker capability disabled for untrusted image
[...]
```

根本原因

该 Image 不受信任。

解决

将 Image 的注册表添加到受信任的注册表列表（`docker.trusted.registries`）。例如，在 `library/ubuntu:latest` 的情况下，将“library”注册表添加到该列表中。

7.4.1.9. Docker Image 不包含 Snappy 库

问题陈述

运行 hadoop-mapreduce-examples pi 作业失败，并显示以下错误：

```
[...]
[timestamp] INFO mapreduce.Job:  map 0% reduce 0%
[timestamp] INFO mapreduce.Job: Task Id :
attempt_1570629976081_0001_m_000000_0, Status : FAILED
Error: org/apache/hadoop/util/NativeCodeLoader.buildSupportsSnappy()Z
```

根本原因

提供的 Docker Image 不包含 Snappy 库。如果使用压缩并且选择了 Snappy 编解码器进行压缩，则 MapReduce 需要此功能。

解决

将 Snappy 库添加到 Image，或将“ MapReduce 的 Map 输出的压缩编解码器”更改为其他编解码器

7.4.1.10. Hadoop UserGroupInformation 类无权访问主机系统中的用户权限

问题陈述

启动后不久，容器失败，但出现以下异常：

```
Exception in thread "main"
org.apache.hadoop.security.KerberosAuthException: failure to login:
javax.security.auth.login.LoginException:
java.lang.NullPointerException: invalid null input: name
      At
com.sun.security.auth.UnixPrincipal.<init>(UnixPrincipal.java:71)
      at
com.sun.security.auth.module.UnixLoginModule.login(UnixLoginModule.jav
a:133)
      at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
      at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.
java:62)
      at
sun.reflect.DelegatingMethodAccessorImpl.invoke( DelegatingMethodAcces
sorImpl.java:43)
```

根本原因

Hadoop UserGroupInformation 类无权访问主机系统中的用户权限。

解决

将/etc/passwd 挂载到 Image。在上游 Hadoop 3.2 文档中可以找到更多配置问题：使用 Docker 容器上游文档启动应用程序。

7.4.1.11. 未为 Docker 容器安装 Kerberos 配置

问题陈述

由于是安全集群，Docker 上运行的 MapReduce 和 Spark 作业失败。它无法获得 Kerberos 领域。

```
user@<hostname>/]$ cd/yarn/container-
logs/application_1573764921308_0002/container_e147_1573764921308_0002_01_000005
[user@<hostname> container_e147_1573764921308_0002_01_000005]$ ll
total 8
-rw-r--r-- 1 systest yarn   0 Nov 14 12:57 prelaunch.err
-rw-r--r-- 1 systest yarn  70 Nov 14 12:57 prelaunch.out
-rw-r--r-- 1 systest yarn   0 Nov 14 12:57 stderr
-rw-r----- 1 systest yarn   0 Nov 14 12:57 stderr.txt
```

```
-rw-r--r-- 1 systest yarn   0 Nov 14 12:57 stdout
-rw-r----- 1 systest yarn   0 Nov 14 12:57 stdout.txt
-rw-r--r-- 1 systest yarn 892 Nov 14 12:57 syslog
[user@<hostname> container_e147_1573764921308_0002_01_000005]$ cat syslog
2019-11-14 20:57:41,765 ERROR [main] org.apache.hadoop.yarn.YarnUncaughtExceptionHandler: Thread
Thread[main,5,main] threw an Exception.
java.lang.IllegalArgumentException: Can't get Kerberos realm
   at org.apache.hadoop.security.HadoopKerberosName.setConfiguration(HadoopKerberosName.java:71)
   at org.apache.hadoop.security.UserGroupInformation.initialize(UserGroupInformation.java:330)
   at org.apache.hadoop.security.UserGroupInformation.setConfiguration(UserGroupInformation.java:381)
   at org.apache.hadoop.mapred.YarnChild.main(YarnChild.java:80)
Caused by: java.lang.IllegalArgumentException
   at javax.security.auth.kerberos.KerberosPrincipal.<init>(KerberosPrincipal.java:136)
   at org.apache.hadoop.security.authentication.util.KerberosUtil.getDefaultRealm(KerberosUtil.java:108)
   at org.apache.hadoop.security.HadoopKerberosName.setConfiguration(HadoopKerberosName.java:69)
   ... 3 more
[user@<hostname> container_e147_1573764921308_0002_01_000005]$
```

根本原因

未为 Docker 容器安装 Kerberos 配置。

解决

如果是 MapReduce 作业，请在运行该作业时添加以下环境变量： `-Dmapreduce.reduce.env=YARN_CONTAINER_RUNTIME_DOCKER_MOUNTS=/etc/krb5.conf:/etc/krb5.conf:ro`

确保添加`/etc/krb5.conf` 到 Cloudera Manager 配置中的“允许的只读装载”。

例子：

```
yarn jar/opt/cloudera/parcels/CDH-7.0.3-
1.cdh7.0.3.p0.1616399/lib/hadoop-mapreduce/hadoop-mapreduce-
examples.jar pi -
Dmapreduce.map.env="YARN_CONTAINER_RUNTIME_TYPE=docker,YARN_CONTAINER_
RUNTIME_DOCKER_IMAGE=library/ibmjava:8,YARN_CONTAINER_RUNTIME_DOCKER_D
ELAYED_REMOVAL=true,YARN_CONTAINER_RUNTIME_DOCKER_MOUNTS=/etc/krb5.con
f:/etc/krb5.conf:ro" -
Dmapreduce.reduce.env="YARN_CONTAINER_RUNTIME_TYPE=docker,YARN_CONTAIN
ER_RUNTIME_DOCKER_IMAGE=library/ibmjava:8,YARN_CONTAINER_RUNTIME_DOCKE
R_DELAYED_REMOVAL=true,YARN_CONTAINER_RUNTIME_DOCKER_MOUNTS=/etc/krb5.
conf:/etc/krb5.conf:ro"  1 40000
```

如果是 Spark 作业，请确保`/etc/krb5.conf`as`spark.appMasterEnv` 和 为只读添加了 mount `spark.executorEnv`：

```
 --conf
spark.yarn.appMasterEnv.YARN_CONTAINER_RUNTIME_DOCKER_MOUNTS=/etc/pass
wd:/etc/passwd:ro,/opt/cloudera/parcels:/opt/cloudera/parcels:ro,/etc/
```

```
krb5.conf:/etc/krb5.conf:ro \
 --conf
spark.executorEnv.YARN_CONTAINER_RUNTIME_DOCKER_MOUNTS="/etc/passwd:/e
tc/passwd:ro,/opt/cloudera/parcels/:/opt/cloudera/parcels/:ro,/etc/krb
5.conf:/etc/krb5.conf:ro"
```

7.4.1.12. ssl-client.xml 没有使用 MapReduce 为 Docker 容器挂载该文件和信任库文件

问题陈述

由于 SSL 握手问题，Reducer 无法连接到随机播放服务。

CLI 日志：

```
19/11/15 03:26:02 INFO impl.YarnClientImpl: Submitted application
application_1573810028869_0004
19/11/15 03:26:02 INFO mapreduce.Job: The url to track the job: <URL>
19/11/15 03:26:02 INFO mapreduce.Job: Running job:
job_1573810028869_0004
19/11/15 03:26:12 INFO mapreduce.Job: Job job_1573810028869_0004
running in uber mode : false
19/11/15 03:26:12 INFO mapreduce.Job:  map 0% reduce 0%
19/11/15 03:26:23 INFO mapreduce.Job:  map 100% reduce 0%
19/11/15 03:27:30 INFO mapreduce.Job: Task Id :
attempt_1573810028869_0004_r_000000_0, Status : FAILED
Error: org.apache.hadoop.mapreduce.task.reduce.Shuffle$ShuffleError:
error in shuffle in fetcher#2
    at
org.apache.hadoop.mapreduce.task.reduce.Shuffle.run(Shuffle.java:136)
    at org.apache.hadoop.mapred.ReduceTask.run(ReduceTask.java:377)
    at org.apache.hadoop.mapred.YarnChild$2.run(YarnChild.java:174)
    at
java.security.AccessController.doPrivileged(AccessController.java:770)
    at javax.security.auth.Subject.doAs(Subject.java:570)
    at
org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformat
ion.java:1876)
    at org.apache.hadoop.mapred.YarnChild.main(YarnChild.java:168)
Caused by: java.io.IOException: Exceeded MAX_FAILED_UNIQUE_FETCHES;
bailing-out.
    at
org.apache.hadoop.mapreduce.task.reduce.ShuffleSchedulerImpl.checkRedu
cerHealth(ShuffleSchedulerImpl.java:396)
    at
org.apache.hadoop.mapreduce.task.reduce.ShuffleSchedulerImpl.copyFaile
d(ShuffleSchedulerImpl.java:311)
    at
org.apache.hadoop.mapreduce.task.reduce.Fetcher.openShuffleUrl(Fetcher
.java:291)
    at
org.apache.hadoop.mapreduce.task.reduce.Fetcher.copyFromHost(Fetcher.j
ava:330)
```

```
    at
org.apache.hadoop.mapreduce.task.reduce.Fetcher.run(Fetcher.java:198)
```

NodeManager 日志：

```
2019-11-15 03:30:16,323 INFO
org.apache.hadoop.yarn.server.nodemanager.NodeStatusUpdaterImpl:
Removed completed containers from NM context:
[container_e149_1573810028869_0004_01_000005]
2019-11-15 03:30:50,812 ERROR org.apache.hadoop.mapred.ShuffleHandler:
Shuffle error:
javax.net.ssl.SSLException: Received fatal alert: certificate_unknown
        at sun.security.ssl.Alerts.getSSLException(Alerts.java:208)
        at
sun.security.ssl.SSLEngineImpl.fatal(SSLEngineImpl.java:1666)
        at
sun.security.ssl.SSLEngineImpl.fatal(SSLEngineImpl.java:1634)
        at
sun.security.ssl.SSLEngineImpl.recvAlert(SSLEngineImpl.java:1800)
        at
sun.security.ssl.SSLEngineImpl.readRecord(SSLEngineImpl.java:1083)
        at
sun.security.ssl.SSLEngineImpl.readNetRecord(SSLEngineImpl.java:907)
        at
sun.security.ssl.SSLEngineImpl.unwrap(SSLEngineImpl.java:781)
        at javax.net.ssl.SSLEngine.unwrap(SSLEngine.java:624)
        at
org.jboss.netty.handler.ssl.SslHandler.unwrap(SslHandler.java:1218)
        at
org.jboss.netty.handler.ssl.SslHandler.decode(SslHandler.java:852)
        at
org.jboss.netty.handler.codec.frame.FrameDecoder.callDecode(FrameDecod
er.java:425)
        at
org.jboss.netty.handler.codec.frame.FrameDecoder.messageReceived(Frame
Decoder.java:303)
        at
org.jboss.netty.channel.SimpleChannelUpstreamHandler.handleUpstream(Si
mpleChannelUpstreamHandler.java:70)
        at
org.jboss.netty.channel.DefaultChannelPipeline.sendUpstream(DefaultCha
nnelPipeline.java:564)
        at
org.jboss.netty.channel.DefaultChannelPipeline.sendUpstream(DefaultCha
nnelPipeline.java:559)
        at
org.jboss.netty.channel.Channels.fireMessageReceived(Channels.java:268
)
        at
org.jboss.netty.channel.Channels.fireMessageReceived(Channels.java:255
)
        at
org.jboss.netty.channel.socket.nio.NioWorker.read(NioWorker.java:88)
```

```
        at
org.jboss.netty.channel.socket.nio.AbstractNioWorker.process(AbstractN
ioWorker.java:108)
        at
org.jboss.netty.channel.socket.nio.AbstractNioSelector.run(AbstractNio
Selector.java:337)
        at
org.jboss.netty.channel.socket.nio.AbstractNioWorker.run(AbstractNioWo
rker.java:89)
        at
org.jboss.netty.channel.socket.nio.NioWorker.run(NioWorker.java:178)
        at
org.jboss.netty.util.ThreadRenamingRunnable.run(ThreadRenamingRunnable
.java:108)
        at
org.jboss.netty.util.internal.DeadLockProofWorker$1.run(DeadLockProofW
orker.java:42)
        at
java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.j
ava:1149)
        at
java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.
java:624)
        at java.lang.Thread.run(Thread.java:748)
2019-11-15 03:30:50,812 ERROR org.apache.hadoop.mapred.ShuffleHandler:
Shuffle error [id: 0xf95ad8ab,/10.65.53.21:44366 =>/10.65.53.21:13562]
EXCEPTION: javax.net.ssl.SSLException: Received fatal alert:
certificate_unknown
2019-11-15 03:30:51,156 INFO
org.apache.hadoop.yarn.server.nodemanager.containermanager.ContainerMa
nagerImpl: Stopping container with container Id:
container_e149_1573810028869_0004_01_000006
```

NodeManager 日志（异常）：

```
2019-11-15 03:30:50,812 ERROR org.apache.hadoop.mapred.ShuffleHandler:
Shuffle error:
javax.net.ssl.SSLException: Received fatal alert: certificate_unknown
        [...]
2019-11-15 03:30:50,812 ERROR org.apache.hadoop.mapred.ShuffleHandler:
Shuffle error [id: 0xf95ad8ab,/10.65.53.21:44366 =>/10.65.53.21:13562]
EXCEPTION: javax.net.ssl.SSLException: Received fatal alert:
certificate_unknown
2019-11-15 03:30:51,156 INFO
org.apache.hadoop.yarn.server.nodemanager.containermanager.ContainerMa
nagerImpl: Stopping container with container Id:
container_e149_1573810028869_0004_01_000006
```

根本原因

对于普通容器，该文件 `ssl-client.xml` 定义了 SSL 设置，并且位于类路径上（通常在目录：下/`etc/hadoop/conf.cloudera.YARN-1/ssl-client.xml`）。因此，必须使用 M

apReduce 将其安装在 Docker 容器中。由于该 `ssl-client.xml` 文件还引用了信任库文件，因此也必须挂载该文件。

解决

运行作业时添加以下内容：

```
-
Dmapreduce.reduce.env=YARN_CONTAINER_RUNTIME_DOCKER_MOUNTS="/etc/hadoop/conf.cloudera.YARN-1/ssl-client.xml:/etc/hadoop/conf.cloudera.YARN-1/ssl-client.xml:ro,/var/lib/cloudera-scm-agent/agent-cert/cm-auto-global_truststore.jks:/var/lib/cloudera-scm-agent/agent-cert/cm-auto-global_truststore.jks:ro"
```

确保在 Cloudera Manager 中添加`/etc/hadoop/conf.cloudera.YARN-1/ssl-client.xml` 并 添加`/var/lib/cloudera-scm-agent/agent-cert/cm-auto-global_truststore.jks` 到允许的只读装载中。

请注意，信任库的位置可能会有所不同，因此请从 `ssl-client.xml` 文件验证其位置 。您可以通过 NodeManager 的“进程”视图在 Clouder Manager 中访问该文件。

7.4.2. 对 Linux Container Executor 进行故障排除

容器执行程序传达给 NodeManager 的数字错误代码的列表，这些错误代码显示在`/var/log/hadoop-yarn` NodeManager 日志中。

表 1.适用于 YARN 中的容器执行器的数字错误代码，但仅由 LinuxContainerExecutor 使用。

代码	名称	描述
1	INVALID_ARGUMENT_NUMBER	给定的 container-executor 命令提供的参数数量不正确 初始化容器本地化器失败
2	无效的用户名	传递给容器执行器的用户不存在。
3	INVALID_COMMAND_PROVIDED	容器执行器无法识别要求其运行的命令。

代码	名称	描述
5	INVALID_NM_ROOT	传递的 NodeManager 根目录与配置的 NodeManager 根目录（`yarn.nodemanager.local-dirs`）不匹配，或者不存在。
6	SETUID_OPER_FAILED	无法读取本地组数据库，或者无法设置 UID 或 GID
7	UNABLE_TO_EXECUTE_CONTAINER_SCRIPT	容器执行器无法运行容器启动器脚本。
8	UNABLE_TO_SIGNAL_CONTAINER	容器执行器无法发出已通过容器的信号。
9	INVALID_CONTAINER_PID	传递给容器执行器的 PID 为负或 0。
18	内存不足	在读取 container-executor.cfg 文件或获取容器启动器脚本或凭据文件的路径时，容器执行器无法分配足够的内存。
20	INITIALIZE_USER_FAILED	无法获取，统计或保护每个用户的 NodeManager 目录。
21	UNABLE_TO_BUILD_PATH	容器执行器无法连接两个路径，这很可能是因为它用尽了内存。
22	INVALID_CONTAINER_EXEC_PERMISSIONS	容器执行程序二进制文件没有设置正确的权限。
24	INVALID_CONFIG_FILE	container-executor.cfg 文件丢失，格式错误或具有错误的权限。
25	SETSID_OPER_FAILED	无法设置分叉容器的会话 ID。
26	WRITE_PIDFILE_FAILED	无法将启动的容器的 PID 的值写入容器的 PID 文件。
255	未知错误	此错误有几种可能的原因。一些常见的原因是： 集群上的用户帐户的用户 ID 小于为文件中的 `min.user.id` 属性 指定的值 `container-executor.cfg`。默认值为 1000，这在 Ubuntu 系统上是合适的，但对于您的操作系统可能无效。有关 `min.user.id` 在 `container-executor.cfg` 文件中设置的信息。 此错误通常是由先前的错误引起的。在日志文件中更早查找可能的原因。

表 2.退出状态代码适用于 YARN 中的所有容器。这些退出状态代码是 YARN 框架的一部分，是可以设置的特定于应用程序的退出代码的补充。

代码	名称	描述
0	成功	容器已成功完成。
-1000	无效	容器退出代码的初始值。不具有 COMPLETED 状态的容器将始终返回此状态。
-100	已中止	例如，被框架杀死的容器，要么是由于被应用程序释放，要么是由于节点故障而“丢失”。
-101	DISKS_FAILED	容器由于 NodeManager 节点中的本地磁盘问题而退出。当良好的 nodemanager-local-directories 或 nodemanager-log-directories 的数量下降到运行状况阈值以下时，就会发生这种情况。
-102	已清除	框架抢占的容器。在大多数应用程序中，这不计入容器故障。
-103	KILLED_EXCEEDED_VMEM	容器由于超出分配的虚拟内存限制而终止。
-104	KILLED_EXCEEDED_PMEM	容器由于超出分配的物理内存限制而终止。
-105	KILLED_BY_APPMASTER	容器已根据应用程序主机的请求终止。
-106	KILLED_BY_RESOURCEMANAGER	容器已由资源管理器终止。
-107	KILLED_AFTER_APP_COMPLETION	应用程序完成后，容器被终止。

7.5. 对 HBase 进行故障排除

如果遇到错误并想在 Apache HBase 日志中搜索特定的关键字，则可以使用 Cloudera Manager 中的日志搜索功能。要在 Cloudera Manager 的 Apache HBase 日志中搜索特定的关键字，请转至 Diagnostics > Logs。您可以从各种来源、服务、角色类型和主机中搜索特定的关键字。

7.5.1. 使用 HBCK2 工具修复 HBase 集群

HBCK2 工具是修复工具，可用于修复 Apache HBase 及 CDP 中的 Apache HBase 的集群。HBCK2 工具是 Apache HBase hbck 工具的下一版本。

要确定正在运行的 HBase 集群中的不一致列表或阻塞列表，可以查看主日志。一旦确定了问题，就可以使用 HBCK2 工具修复缺陷或跳过不良状态。HBCK2 工具使用交互式修复过程，要求主机进行修复，而不是在本地进行修复。

HBCK2 每次运行时都执行一个单独的任务。HBCK2 工具不会分析正在运行的集群中的所有内容并修复所有问题。相反，您可以使用 HBCK2 工具来迭代查找和修复集群中的问题。HBCK2 工具使您可以使用交互式命令来一次解决一个问题。

重要

HBCK2 工具特定于 Apache HBase 的内部。使用此工具需要特定于您的 CDP 运行时版本的二进制文件，并且您必须始终在 Cloudera 支持和/或 Cloudera 专业服务的帮助下使用它。如果您认为需要使用 HBCK2 工具遇到问题，请联系 Cloudera 支持。

7.5.2. Thrift Server 在收到无效数据后崩溃

如果 Thrift 服务器由于缓冲区溢出而收到大量无效数据，则可能会崩溃。

为什么会这样

Thrift 服务器分配内存以检查其接收到的数据的有效性。如果它接收到大量无效数据，则可能需要分配比可用内存更多的内存。这是由于 Thrift 库本身的限制。

该怎么办

为避免由于缓冲区溢出而导致崩溃的可能性，请使用带框架的紧凑型传输协议。这些协议默认情况下处于禁用状态，因为它们可能需要更改您的客户端代码。添加到您的 hbase-site.xml 中的两个选项是 `hbase.regionserver.thrift.framed` 和 `hbase.regionserver.thrift.compact`。将每个设置为 `true`，如下面的 XML 所示。您还可以使用该 `hbase.regionserver.thrift.framed.max_frame_size_in_mb` 选项指定最大帧大小 。

```
<property>
  <name>hbase.regionserver.thrift.framed</name>
  <value>true</value>
</property>
```

```
<property>
  <name>hbase.regionserver.thrift.framed.max_frame_size_in_mb</name>
  <value>2</value>
</property>
<property>
  <name>hbase.regionserver.thrift.compact</name>
  <value>true</value>
</property>
```

7.5.3. HBase 正在使用比预期更多的磁盘空间

HBase StoreFiles（也称为 HFiles）将 HBase 行数据存储在磁盘上。HBase 将其他信息存储在磁盘上，例如预写日志（WAL）、快照，否则将被删除但需要从存储的快照还原的数据。

警告

提供以下信息以帮助您仅解决磁盘使用率过高的问题。请勿在 HBase API 或 HBase Shell 范围之外编辑或删除任何此类数据，否则您的数据很可能会损坏。

表 1. HBase 磁盘使用情况

路径	目的	故障排除说明
`/hbase/.snapshots`	每个快照包含一个子目录。	要列出快照，请使用 HBase Shell 命令 `list_snapshots`。要删除快照，请使用 `delete_snapshot`。
`/hbase/.archive`	包含本应删除的数据（由于由于 TTL 或表上的版本限制而被明确删除或过期），但是从现有快照还原时需要这些数据。	要释放过多存档占用的空间，请删除引用它们的快照。快照永不过期，因此保留它们引用的数据，直到删除快照为止。请勿手动删除`/hbase/.archive` 的任何内容 ，否则将损坏快照。
`/hbase/.logs`	包含在 RegionServer 发生故障时恢复区域所需的 HBase WAL 文件。	验证 WAL 的内容已写入 StoreFiles 时，将删除它们。请勿手动删除它们。如果`/hbase/.logs/`的任何子目录的大小都在增加，请检查 HBase 服务器日志以找到 WAL 未得到正确处理的根本原因。

路径	目的	故障排除说明
/hbase/logs/.oldWALs	包含已经写入磁盘的 HBase WAL 文件。HBase 维护线程会根据 TTL 定期删除它们。	要调整 WAL 在.oldWALs 被删除之前停留的时间 ，请配置该 hbase.master.logcleaner.ttl 属性，该属性默认为 60000 毫秒或 1 小时。
/hbase/.logs/.corrupt	包含损坏的 HBase WAL 文件。	不要手动删除损坏的 WAL。如果/hbase/.logs/的任何子目录的大小都在增加，请检查 HBase 服务器日志以找到 WAL 未得到正确处理的根本原因。

7.5.4. 对 RegionServer 分组进行故障排除

使用 rsgroup 时，如果遇到以下示例中的问题，请检查日志以查看引起问题的原因。如果 rsgroup 操作没有响应，则必须重新启动 HBase Master。

如果您开始使用 rsgroup 但未找到所需的协处理器，将会看到一个示例错误：

```
ERROR: org.apache.hadoop.hbase.exceptions.UnknownProtocolException: No
registered master coprocessor service found for name
RSGroupAdminService
    at
org.apache.hadoop.hbase.master.MasterRpcServices.execMasterService(Mas
terRpcServices.java:604)
    at
org.apache.hadoop.hbase.shaded.protobuf.generated.MasterProtos$MasterS
ervice$2.callBlockingMethod(MasterProtos.java)
    at org.apache.hadoop.hbase.ipc.RpcServer.call(RpcServer.java:1140)
    at org.apache.hadoop.hbase.ipc.CallRunner.run(CallRunner.java:133)
    at
org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:2
77)
    at
org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:2
57)
```

7.6. 对 APACHE KUDU 进行故障排除

7.6.1. 启动或重启主服务器或者 Tablet 服务器时出现问题

7.6.1.1. 打孔测试中的错误

Kudu 需要打孔功能才能高效运行。打孔支持取决于您的操作系统内核版本和本地文件系统实现。

- RHEL 或 CentOS 6.4 或更高版本，已修补到 2.6.32-358 或更高版本的内核版本。未打补丁的 RHEL 或 CentOS 6.4 不包括支持打孔的内核。
- Ubuntu 14.04 包含 Linux 内核的 3.13 版本，该版本支持打孔。
- 较新版本的 ext4 和 xfs 文件系统支持打孔。不支持打孔的较旧版本将导致 Kudu 发出如下错误消息并无法启动：

```
Error during hole punch test. The log block manager requires a filesystem with hole punching support such as ext4 or xfs. On el6, kernel version 2.6.32-358 or newer is required. To run without hole punching (at the cost of some efficiency and scalability), reconfigure Kudu with --block_manager=file. Refer to the Kudu documentation for more details. Raw error message follows.
```

笔记

ext4 挂载点实际上可以由不支持打孔的 ext2 或 ext3 格式的设备支持。在此类文件系统上运行时，打孔测试将失败。有几种不同的方法可以确定 ext2、ext3 或 ext4 格式的设备是否支持 ext4 挂载点。

如果没有打孔支持，则日志块管理器将无法安全使用。它永远不会删除块，并且会占用磁盘上更多的空间。

如果您不能在环境中使用打孔，仍可以尝试 Kudu。通过将`--block_manager=file`标志添加到用于启动主服务器和 Tablet 服务器的命令中，启用文件块管理器而不是日志块管理 器。文件块管理器不能像日志块管理器那样扩展。

笔记

众所周知，文件块管理器的伸缩性和性能不佳，仅应用于小规模评估和开发，并且只能用于无法打孔的系统。

文件块管理器每个块使用一个文件。由于为每个行集写入了多个块，因此块的数量可能会非常多，尤其是对于主动写入的数位板。与日志块管理器相比，即使有少量数据，这

也可能导致性能问题。而且，如果不擦除和重新初始化 Tablet 服务器，就不可能在块管理器之间切换。

7.6.1.2. Already present: FS layout already exists

Kudu 启动时，它将检查每个已配置的数据目录，期望它们全部被初始化或全部为空。如果服务器无法通过如下所示的日志消息启动，则该前提条件检查失败。

```
Check failed: _s.ok() Bad status: Already present: Could not create new FS layout: FSManager root is not empty:/data0/kudu/data
```

这可能是因为 Kudu 首次启动时配置了非空数据目录，或者是因为先前运行的，健康的 Kudu 进程已重新启动，并且至少一个数据目录被删除或以某种方式损坏（可能是由于磁盘错误）。如果是后者，请参阅*更改目录配置*。

7.6.1.3. 排除 NTP 稳定性问题

从 Kudu 1.6.0 开始，Kudu 守护程序可以在短暂的时钟同步丢失期间继续运行。如果时钟同步丢失了几个小时，则 Kudu 守护程序可能会崩溃。如果守护程序由于时钟同步问题而崩溃，请查阅 `ERROR` 日志以获取相关信息的转储，这可能有助于诊断问题。

笔记

如果使用的不是本地链接 NTP 服务器，则 `ntpd` 在网络连接问题的情况下，可能需要花费一些时间与其参考服务器之一进行同步。如果机器和参考 NTP 服务器之间的网络 `ntpd` 不完整，则可能与其参考 NTP 服务器不同步。如果发生这种情况，请考虑查找其他参考 NTP 服务器集：最好的选择是在本地网络或`*.pool.ntp.org` 服务器中使用 NTP 服务器。

7.6.2. 磁盘空间使用问题

使用日志块管理器（Linux 上的默认设置）时，Kudu 使用稀疏文件来存储数据。稀疏文件的表观大小与其实际使用的磁盘空间大小不同。这意味着某些工具可能会错误地报告 Kudu 使用的磁盘空间。例如，列出的大小 `ls -l` 不能准确反映 Kudu 数据文件使用的磁盘空间：

```
$ ls -lh/data/kudu/tserver/data
total 117M
```

```
-rw------- 1 kudu kudu 160M Mar 26 19:37
0b9807b8b17d48a6a7d5b16bf4ac4e6d.data
-rw------- 1 kudu kudu 4.4K Mar 26 19:37
0b9807b8b17d48a6a7d5b16bf4ac4e6d.metadata
-rw------- 1 kudu kudu  32M Mar 26 19:37
2f26eeacc7e04b65a009e2c9a2a8bd20.data
-rw------- 1 kudu kudu 4.3K Mar 26 19:37
2f26eeacc7e04b65a009e2c9a2a8bd20.metadata
-rw------- 1 kudu kudu 672M Mar 26 19:37
30a2dd2cd3554d8a9613f588a8d136ff.data
-rw------- 1 kudu kudu 4.4K Mar 26 19:37
30a2dd2cd3554d8a9613f588a8d136ff.metadata
-rw------- 1 kudu kudu  32M Mar 26 19:37
7434c83c5ec74ae6af5974e4909cbf82.data
-rw------- 1 kudu kudu 4.3K Mar 26 19:37
7434c83c5ec74ae6af5974e4909cbf82.metadata
-rw------- 1 kudu kudu 672M Mar 26 19:37
772d070347a04f9f8ad2ad3241440090.data
-rw------- 1 kudu kudu 4.4K Mar 26 19:37
772d070347a04f9f8ad2ad3241440090.metadata
-rw------- 1 kudu kudu 160M Mar 26 19:37
86e50a95531f46b6a79e671e6f5f4151.data
-rw------- 1 kudu kudu 4.4K Mar 26 19:37
86e50a95531f46b6a79e671e6f5f4151.metadata
-rw------- 1 kudu kudu  687 Mar 26 19:26 block_manager_instance
```

请注意，报告的总大小为 117MiB，而第一个文件的大小列为 160MiB。将 `-s` 选项添加到 `ls` 将会导致 `ls` 输出文件的磁盘空间使用情况。

在 `du` 与 `df` 工具报告默认情况下，实际的磁盘空间使用情况。

```
$ du -h/data/kudu/tserver/data118M  /data/kudu/tserver/data
```

外观尺寸可以通过`--apparent-size` 标记来显示 `du`。

```
$ du -h --apparent-size/data/kudu/tserver/data1.7G
/data/kudu/tserver/data
```

7.6.3. 性能问题

本主题可帮助您使用 Kudu 跟踪、内存限制、块大小缓存、堆采样和名称服务缓存守护程序（`nscd`）对问题进行故障排除并提高性能。

7.6.3.1. Kudu 追踪

Kudu 主服务器和 Tablet 服务器守护程序包括基于开源 Chromium 跟踪框架的对跟踪的内置支持。您可以使用跟踪来诊断延迟问题或 Kudu 服务器上的其他问题。

7.6.3.1.1. 访问跟踪 Web 界面

跟踪界面是每个 Kudu 守护程序中嵌入式 Web 服务器的一部分，可以使用 Web 浏览器进行访问。请注意，虽然已知该界面可在最新版本的 Google Chrome 浏览器中运行，但其他浏览器可能无法按预期运行。

守护进程	网址
Tablet Server	*<tablet-server-1.example.com>*:8050/tracing.html
Master	*<master-1.example.com>*:8051/tracing.html

7.6.3.1.2. RPC 超时跟踪

如果客户端应用程序遇到 RPC 超时，则 KuduTablet 服务器 WARNING 级别的日志应包含一个包含 RPC 级别跟踪的日志条目。

例如：

```
W0922 00:56:52.313848 10858 inbound_call.cc:193] Call
kudu.consensus.ConsensusService.UpdateConsensus
from 192.168.1.102:43499 (request call id 3555909) took 1464ms (client
timeout 1000).
W0922 00:56:52.314888 10858 inbound_call.cc:197] Trace:
0922 00:56:50.849505 (+     0us) service_pool.cc:97] Inserting onto
call queue
0922 00:56:50.849527 (+    22us) service_pool.cc:158] Handling call
0922 00:56:50.849574 (+    47us) raft_consensus.cc:1008] Updating
replica for 2 ops
0922 00:56:50.849628 (+    54us) raft_consensus.cc:1050] Early marking
committed up to term: 8 index: 880241
0922 00:56:50.849968 (+   340us) raft_consensus.cc:1056] Triggering
prepare for 2 ops
0922 00:56:50.850119 (+   151us) log.cc:420] Serialized 1555 byte log
entry
0922 00:56:50.850213 (+    94us) raft_consensus.cc:1131] Marking
committed up to term: 8 index: 880241
0922 00:56:50.850218 (+     5us) raft_consensus.cc:1148] Updating last
received op as term: 8 index: 880243
0922 00:56:50.850219 (+     1us) raft_consensus.cc:1195] Filling
consensus response to leader.
0922 00:56:50.850221 (+     2us) raft_consensus.cc:1169] Waiting on
the replicates to finish logging
0922 00:56:52.313763 (+1463542us) raft_consensus.cc:1182] finished
0922 00:56:52.313764 (+     1us) raft_consensus.cc:1190]
UpdateReplicas() finished
0922 00:56:52.313788 (+    24us) inbound_call.cc:114] Queueing success
response
```

这些跟踪可以指示请求的哪一部分速度很慢。在提交与 RPC 延迟异常值相关的错误报告时，请确保包括它们。

7.6.3.1.3. 内核堆栈看门狗跟踪

每个 Kudu 服务器进程都有一个称为 Stack Watchdog 的后台线程，该线程监视服务器中的其他线程，以防它们被阻塞的时间超过预期的时间。这些跟踪可能表明操作系统问题或瓶颈存储。

看门狗线程识别出线程阻塞的情况时，将在日志中记录一个条目，`WARNING` 如下所示：

```
W0921 23:51:54.306350 10912 kernel_stack_watchdog.cc:111] Thread 10937
stuck at/data/kudu/consensus/log.cc:505 for 537ms:
Kernel stack:
[<ffffffffa00b209d>] do_get_write_access+0x29d/0x520 [jbd2]
[<ffffffffa00b2471>] jbd2_journal_get_write_access+0x31/0x50 [jbd2]
[<ffffffffa00fe6d8>] __ext4_journal_get_write_access+0x38/0x80 [ext4]
[<ffffffffa00d9b23>] ext4_reserve_inode_write+0x73/0xa0 [ext4]
[<ffffffffa00d9b9c>] ext4_mark_inode_dirty+0x4c/0x1d0 [ext4]
[<ffffffffa00d9e90>] ext4_dirty_inode+0x40/0x60 [ext4]
[<ffffffff811ac48b>] __mark_inode_dirty+0x3b/0x160
[<ffffffff8119c742>] file_update_time+0xf2/0x170
[<ffffffff8111c1e0>] __generic_file_aio_write+0x230/0x490
[<ffffffff8111c4c8>] generic_file_aio_write+0x88/0x100
[<ffffffffa00d3fb1>] ext4_file_write+0x61/0x1e0 [ext4]
[<ffffffff81180f5b>] do_sync_readv_writev+0xfb/0x140
[<ffffffff81181ee6>] do_readv_writev+0xd6/0x1f0
[<ffffffff81182046>] vfs_writev+0x46/0x60
[<ffffffff81182102>] sys_pwritev+0xa2/0xc0
[<ffffffff8100b072>] system_call_fastpath+0x16/0x1b
[<ffffffffffffffff>] 0xffffffffffffffff

User stack:
    @       0x3a1ace10c4  (unknown)
    @          0x1262103  (unknown)
    @          0x12622d4  (unknown)
    @          0x12603df  (unknown)
    @           0x8e7bfb  (unknown)
    @           0x8f478b  (unknown)
    @           0x8f55db  (unknown)
    @          0x12a7b6f  (unknown)
    @       0x3a1b007851  (unknown)
    @       0x3a1ace894d  (unknown)
    @                (nil)  (unknown)
```

这些跟踪对于诊断 Kudu 中的根本原因延迟问题很有用，尤其是当它们是由底层系统（例如磁盘控制器或文件系统）引起的。

7.6.3.2. 内存限制

Kudu 具有硬内存限制。硬内存限制是 Kudu 进程允许使用的最大数量，并由该--memory_limit_hard_bytes 标志控制。软内存限制是硬内存限制的百分比，由标志控制，memory_limit_soft_percentage 默认值为 80%，它确定进程在开始拒绝某些写操作之前可能使用的内存量。

如果日志或 RPC 跟踪包含诸如以下示例的消息，则由于内存背压，Kudu 将拒绝写入。这可能会导致写入超时。

```
Service unavailable: Soft memory limit exceeded (at 96.35% of
capacity)
```

有几种方法可以缓解 Kudu 的内存压力：

- 如果主机有更多可用于 Kudu 的内存，请增加 --memory_limit_hard_bytes。
- 通过增加磁盘数量或增加维护管理器线程数量，提高 Kudu 可以将内存中的写入刷新到磁盘的速率 --maintenance_manager_num_threads。通常，维护管理器线程与数据目录的建议比率为 1：3。
- 减少在应用程序端流向 Kudu 的写入量。

最后，在 Kudu 1.7 及更低版本中，检查--block_cache_capacity_mb 设置的值 。此设置确定 Kudu 的块缓存的最大大小。虽然较高的值有助于提高读取和写入性能，但将其 --memory_limit_hard_bytes 设置为设置值的百分比过高 则会有害。不要提高 --block_cache_capacity_mb 到 --memory_pressure_percentage（默认值的 60%）以上 --memory_limit_hard_bytes，因为这将导致 Kudu 主动刷新，即使写入吞吐量很低也是如此。建议的值--block_cache_capacity_mb 低于以下值 ：

（50%* --memory_pressure_percentage）* --memory_limit_hard_bytes

使用默认值时，这意味着--block_cache_capacity_mb 不得超过的 30%--memory_limit_hard_bytes。

在 Kudu 1.8 及更高版本中，如果块缓存容量超过内存压力阈值，服务器将拒绝启动。

7.6.3.3. 块缓存大小

Kudu 使用 LRU 缓存存储最近读取的数据。在重复扫描数据子集的工作负载上，增大此缓存的大小可提供显着的性能优势。要增加专用于块高速缓存的内存量，请增加`--block_cache_capacity_mb` 标志的值。默认值为 512 MiB。

Kudu 提供了一组有用的度量来评估块缓存的性能，可以`/metrics` 在 Web UI 的端点上找到这些度量。以下是示例集：

```
{
  "name": "block_cache_inserts",
  "value": 64
},
{
  "name": "block_cache_lookups",
  "value": 512
},
{
  "name": "block_cache_evictions",
  "value": 0
},
{
  "name": "block_cache_misses",
  "value": 96
},
{
  "name": "block_cache_misses_caching",
  "value": 64
},
{
  "name": "block_cache_hits",
  "value": 0
},
{
  "name": "block_cache_hits_caching",
  "value": 352
},
{
  "name": "block_cache_usage",
  "value": 6976
}
```

要判断 Tablet 服务器上块缓存的效率，请先等待服务器一直运行并为正常请求提供服务一段时间，以便缓存不冷。除非服务器存储的数据很少或处于空闲状态， `block_cac`

he_usage 否则应等于或几乎等于 block_cache_capacity_mb。一旦高速缓存达到稳定状态，比较 block_cache_lookups 到 block_cache_misses_caching。后一个指标计算 Kudu 希望从缓存中读取但在缓存中找不到的块数。如果大量查找导致预期的高速缓存命中未命中，并且该 block_cache_evictions 指标与 block_cache_inserts，那么增加块缓存的大小可以提高性能。但是，块高速缓存的实用程序高度依赖于工作负载，因此有必要测试更大的块高速缓存的好处。

笔记

不要将块高速缓存的大小--block_cache_capacity_mb 提高到大于内存压力阈值（默认为的 60% --memory_limit_hard_bytes）。由于这将导致不良的刷新行为，因此，如果以这种方式配置错误，则 Kudu 服务器版本 1.8 及更高版本将拒绝启动。

7.6.3.4. 堆采样

为了对内存使用情况进行高级调试，管理员可以在 Kudu 守护程序上启用堆采样。这使 Kudu 开发人员可以将内存使用情况与负责的特定代码行和数据结构相关联。当报告与内存使用或明显的内存泄漏相关的错误时，堆分析可以提供定量数据以查明问题所在。

警告

堆采样是一种高级故障排除技术，可能会导致性能下降或 Kudu 服务的不稳定。当前，除非 Kudu 开发团队明确要求，否则不建议在生产环境中启用此功能。

要在 Kudu 守护程序上启用堆采样，请传递标记 --heap-sample-every-n-bytes=524588。如果启用了堆采样，则可以通过访问 http://tablet-server.example.com:8050/pprof/heap 或 通过 HTTP 检索当前采样的堆占用率 http://master.example.com:8051/pprof/heap。输出是堆栈跟踪及其关联的堆使用情况的机器可读的转储。

与其直接在 Web 浏览器中访问堆概要文件页面，pprof 不如使用作为 gperftools 开源项目的一部分分发的工具， 通常更为有用。例如，具有本地构建树的开发人员可以使用以下命令来收集采样的堆使用情况并输出 SVG 图：

```
thirdparty/installed/uninstrumented/bin/pprof -svg
'http://localhost:8051/pprof/heap' >/tmp/heap.svg
```

生成的 SVG 可以在 Web 浏览器中可视化或发送到 Kudu 社区，以帮助解决内存占用问题。

小费

堆样本仅包含有关分配的摘要信息，不包含堆中的任何 *数据*。在公共场合共享堆样本是安全的，而不必担心暴露机密或敏感数据。

7.6.3.5. 慢的名称解析和 nscd

为了在托管许多副本的节点上具有更好的可伸缩性，我们建议您使用 `nscd`（名称服务缓存守护程序）同时缓存 DNS 名称解析和静态名称解析（通过`/etc/hosts`）。

如果 DNS 查找速度缓慢，您将看到类似于以下内容的日志消息：

```
W0926 11:19:01.339553 27231 net_util.cc:193] Time spent resolve
address for kudu-tserver.example.com: real 4.647s user 0.000s sys
0.000s
```

`nscd` 可以通过为最常用的名称服务请求（例如密码、组和主机）提供缓存来减轻名称解析速度过慢的情况。

有关如何安装和启用的信息，请参阅您的操作系统文档 `nscd`。

7.6.4. 可用性问题

本主题列出了使用 Kudu 时可能会遇到的一些常见异常和错误，并可以帮助您解决与可用性有关的问题。

7.6.4.1. ClassNotFoundException：com.cloudera.kudu.hive.KuduStorageHandler

当您尝试使用 Hive 访问 Kudu 表时，您将遇到此异常。这不是缺少 jar 的情况，而仅仅是 Impala 将 Kudu 元数据存储在 Hive 中，而这种格式是其他工具（包括 Hive 和 Spark）无法读取的。当前，没有针对 Hive 用户的解决方法。Spark 用户可以通过创建临时表来解决此问题。

7.6.4.2. 运行时错误：无法创建线程：资源暂时不可用（错误 11）

当 Kudu 无法创建更多线程时，通常在早于 Kudu 1.7 的版本上，您可能会遇到此错误。它在 Tablet 服务器上发生，这表明该 Tablet 服务器托管了太多的 Tablet 副本。

要解决此问题，您可以 `nproc` 根据操作系统或发行版的文档中的说明提高 ulimit。

但是，更好的解决方案是减少 Tablet 服务器上的副本数量。这可能涉及重新考虑表的分区架构。

7.6.4.3. 逻辑删除或已停止的 Tablet 副本

您可能会注意到 Tablet 服务器上的某些副本处于 STOPPED 状态，并无限期地保留在服务器上。这些复制品是墓碑。墓碑表示 Tablet 服务器曾经拥有其 Tablet 的真实副本。

例如，如果 Tablet 服务器出现故障并且其副本在其他地方重新复制，则如果 Tablet 服务器重新加入集群，则其副本将成为墓碑。逻辑删除将一直保留，直到删除其所属的表，或将相同 Tablet 的新副本放置在 Tablet 服务器上为止。Tablet 服务器 Web UI 的 `/tablets` 页面上提供了多个逻辑删除的副本以及每个副本的详细信息。Kudu 用于复制的 Raft 共识算法在某些罕见情况下需要墓碑来确保正确性。它们消耗最少的资源，并且不保存任何数据。不能删除它们。

7.6.4.4. 损坏：CFile 块上的校验和错误

在 Kudu 1.8.0 之前的版本中，如果磁盘上的数据已损坏，则当您尝试扫描具有损坏的 CFile 块的 Tablet 时，在 Tablet 服务器日志中会出现警告，提示中包含“ Corruption：CFile 块的校验和错误”和客户端错误。解决此损坏是一个手动过程。

要解决此问题，请首先使用 ksck 工具在受影响的表或 Tablet 上执行校验和扫描，以找出所有受影响的 Tablet 。

```
sudo -u kudu kudu cluster ksck <master_addresses> -checksum_scan -
tables=<tables>
sudo -u kudu kudu cluster ksck <master_addresses> -checksum_scan -
tablets=<tablets>
```

如果每台 Tablet 至少有一个副本不返回损坏错误，则可以通过以下方式修复不良副本：删除不良副本，并使用 `remote_replica` 删除工具强制从领导者中重新复制这些不良副本。

```
sudo -u kudu kudu remote_replica delete <tserver_address> <tablet_id>
"Cfile Corruption"
```

如果所有副本都已损坏，则发生了一些数据丢失。在 KUDU-2526 完成之前，如果损坏的副本成为引导者并替换了现有的跟随者副本，则可能会发生这种情况。

如果数据丢失，则可以通过使用该 `unsafe_replace_tablet` 工具将损坏的 Tablet 换成空的 Tablet 来修复该表。

```
sudo -u kudu kudu tablet unsafe_replace_tablet <master_addresses>
<tablet_id>
```

从 1.8.0 版开始，Kudu 会将受影响的副本标记为失败，从而导致它们在其他位置自动重新复制。

7.6.5. 象征堆栈跟踪

本主题可帮助您确定线程之间是否存在争用锁的高争用以及象征堆栈地址的方法。

有时您可能会在日志中看到以下内容：

```
0323 03:59:31.091198 (+607857us) spinlock_profiling.cc:243] Waited 492
ms on lock 0x4cb0960. stack: 0000000002398852 0000000000ad8c69
0000000000aa62ba 000000000221aaa8 000000000221b1a8 00000000023a8f83
00007fa8b818be24 00007fa8b646a34c
```

这通常是线程之间争夺锁的争夺的迹象，在这种情况下，报告的时间显示了线程在获取锁之前花在 CPU 上的时间。列出的调用堆栈地址有助于恢复等待线程的堆栈跟踪并在代码中定位问题。

可以将地址转换为具有产生输出的二进制文件的代码中的函数和行的名称（在此示例中为 `kudu-master`）。如果二进制文件中没有符号和调试信息，则可以单独获取二进制文件的调试信息。

假设已剥离的发行版二进制文件和调试信息都可以作为 RPM 使用，请将它们解压缩到目录中；例如 sysroot：

```
$ mkdir sysroot && cd sysroot
$ rpm2cpio ../kudu-1.10.0.el7.x86_64.rpm | cpio -idmv
$ rpm2cpio ../kudu-debuginfo-1.10.0.el7.x86_64.rpm | cpio -idmv
```

用于 addr2line 在代码中找到堆栈地址的行。如果未清除二进制文件中的调试信息，请为实际二进制文件提供一个 -e 选项，而不是调试信息文件，如下所示：

```
addr2line -C -f -e usr/lib/debug/usr/lib/kudu/sbin-release/kudu-
master.debug 0x0000000000aa62ba
kudu::master::MasterServiceImpl::ConnectToMaster(kudu::master::Connect
ToMasterRequestPB const*, kudu::master::ConnectToMasterResponsePB*,
kudu::rpc::RpcContext*)
/usr/src/debug/kudu-1.10.0/src/kudu/master/master_service.cc:504
```

要实现与相同的功能 gdb，请先.text 在符号文件中找到该部分的地址（在示例中为 0000000000a2cdb0）：

```
$ readelf -S usr/lib/debug/usr/lib/kudu/sbin-release/kudu-master.debug
| grep .text
  [13] .text             NOBITS           0000000000a2cdb0  000002c0
```

然后启动 gdb，将其指向 kudu-master 可执行文件（即在日志文件中生成输出的可执行文件）：

```
gdb usr/lib/kudu/sbin-release/kudu-master
```

现在，将.debug 符号加载到 gdb 使用上面找到的地址中。告诉 gdb 在哪里可以找到源文件并设置 sysroot：

```
(gdb) add-symbol-file usr/lib/debug/usr/lib/kudu/sbin-release/kudu-
master.debug 0x0000000000a2cdb0
(gdb) set substitute-path/usr/src/debug/kudu-1.10.0
usr/src/debug/kudu-1.10.0
(gdb) set sysroot .
```

要将地址转换为行号和功能信息，请使用 info line * <address>：

```
(gdb) info line * 0x0000000000aa62ba
Line 504 of "/usr/src/debug/kudu-
1.10.0/src/kudu/master/master_service.cc"
   starts at address 0xaa62af
<kudu::master::MasterServiceImpl::ConnectToMaster(kudu::master::Connec
tToMasterRequestPB const*, kudu::master::ConnectToMasterResponsePB*,
kudu::rpc::RpcContext*)+47>
   and ends at 0xaa62bb
```

```
<kudu::master::MasterServiceImpl::ConnectToMaster(kudu::master::Connec
tToMasterRequestPB const*, kudu::master::ConnectToMasterResponsePB*,
kudu::rpc::RpcContext*)+59>.
```

7.6.6. 在多主服务器部署中从死掉的 Kudu 主服务器中恢复

如果发生主服务器丢失，则 Kudu 多主服务器部署将正常运行。但是，重要的是更换死机。否则，第二次故障可能会导致可用性降低，具体取决于可用主机的数量。此工作流程描述了如何更换失效的主控器。

由于 KUDU-1620，如果不重新启动实时主机，就无法执行此工作流程。因此，工作流需要一个维护窗口，尽管如果集群是使用 DNS 别名设置的，则可能是一个简短的窗口。

重要的

- Kudu 尚不支持对母版进行实时 Raft 配置更改。这样，仅当使用 DNS 别名创建部署或首先关闭集群中的每个节点时，才可以替换主服务器。有关 使用 DNS 别名进行部署的更多详细信息，请参阅*"迁移到多个 Kudu 主机"*中的先前的多主机迁移工作流程。
- 该工作流至少需要基本熟悉 Kudu 配置管理。如果使用 Cloudera Manager，则工作流还必须以熟悉它为前提。
- 以下所有命令行步骤通常应以 Kudu UNIX 用户身份执行 `kudu`。

7.7. 对 Cloudera Search 进行故障排除

安装和部署 Cloudera Search 之后，请使用本节中的信息来解决问题。

7.7.1. 故障排除

下表包含一些常见的故障排除技术。

注意：在下表中的 URL 中，必须在使用环境中的替换`<server:port>`条目，例如。端口的默认值是 8983，如果您有疑问，请查看`/etc/default/solr` 或`/opt/cloudera/parcels/CDH-*/etc/default/solr` 查看该端口。

症状	解释	推荐
全部	多变	检查 Solr 日志。默认情况下，该日志位于/var/log/solr/solr.out。
找不到文件	服务器可能未运行	浏览到 http://*server:port*/solr 以查看服务器是否响应。检查核心是否存在。检查核心的内容，以确保 numDocs 大于 0。
找不到文件	核心可能没有文件	浏览 http://server:port/solr/[collection name]/select?q=*:*&wt=json&indent=true 在顶部附近应显示 numFound 且大于 0。
安全的 Solr Server 无法响应 Solrj 请求，但是其他客户端（例如 curl）可以成功通信	这可能是版本兼容性问题。Httpclient 在 Search 1.x 中 solrj 附带的 4.2.3 依赖于 commons-codec 1.7。如果 commons-codec 类路径上有的早期版本，则 httpclient 可能无法使用 Kerberos 进行通信。	确保您的应用程序使用的是 commons-codec1.7 或更高版本。或者，httpclient 在您的应用程序中使用 4.2.5 而不是 4.2.3 版本。4.2.3 版与的早期版本可正常运行 commons-codec。

7.7.2. 动态 Solr 分析

任何支持 JMX 的应用程序都可以向 Solr 查询信息，并动态显示结果。例如，Zabbix，Nagios 和许多其他应用程序已成功使用。完成静态 Solr 日志分析后，可以从查询 Solr 中看到许多与从日志文件中提取数据有关的项目，至少是最后一个值（与日志文件中可用的历史记录相反）。这些通常对于状态板很有用。通常，可以从 Solr 实时请求 Solr 管理页面上可用的任何内容。一些可能性包括：

- 每个 Core 的 numDocs/maxDoc。这可能很重要，因为这些数字之间的差异表示索引中已删除文档的数量。删除的文档占用磁盘空间和内存。如果这些数字相差很大，建议进行优化的情况可能很少见。
- 缓存统计信息，包括：
    - 命中率
    - 自动预热时间

- 驱逐

- 管理员页面上几乎所有可用的内容。请注意，深入到“Schema 浏览器”可能会很昂贵。

7.7.3. 其他故障排除信息

由于 Solr 和搜索的用例各不相同，因此没有适用于所有情况的单一解决方案。也就是说，这是许多搜索用户遇到的一些常见挑战：

- 使用不切实际的数据集进行测试。例如，用户可以测试针对小数据集使用构面，分组，排序和复杂模式的原型。当使用同一系统加载真实数据时，会出现性能问题。使用现实的数据和用例对于获得准确的结果至关重要。
- 如果情况似乎是系统吸收数据的速度很慢，请考虑：
  - 上游速度。如果您有一个 SolrJ 程序以 100 文档/秒的速度将数据泵送到您的集群并提取文档，则选通因素可能是上游速度。要测试上游速度带来的限制，请仅注释掉将数据发送到服务器的代码（例如 `SolrHttpServer.add(doclist)`）并为程序计时。如果您看到吞吐量增加量不足 10%，则表明您的系统花费了大部分或全部时间从记录系统中获取数据。
  - 这可能需要预处理。
  - 使用来自客户端的单个线程进行索引。 `ConcurrentUpdateSolrServer` 可以使用多个线程来避免 I/O 等待。
  - 太频繁的提交。从历史上看，这是尝试进行 NRT 处理的尝试，但是使用 SolrCloud 进行硬提交，这种情况很少见。
  - 分析链的复杂性。请注意，这很少是核心问题。一个简单的测试是更改架构定义以使用琐碎的分析链，然后衡量性能。
  - 当简单方法无法识别问题时，请考虑使用探查器。

7.7.4. 找出 Cloudera Search 部署中的问题

了解影响搜索性能的常见问题以及您可以采取哪些措施。

要调查您的 Cloudera Search 部署是否存在性能问题或配置错误，请检查日志文件，架构文件和实际索引是否存在问题。如果可能，请在观察日志文件的同时连接到实时 Solr

实例，以便可以将架构与索引进行比较。例如，在更改 Schema 但从未重建索引的情况下，Schema 和索引可能不同步。

问题	推荐
大量或零匹配查询	这表明应用程序的面向用户部分使用户可以轻松输入没有匹配项的查询。在 Cloudera Search 中，考虑到数据的大小，这应该是非常罕见的事件。
与文档数量过多匹配的查询	必须对与查询匹配的所有文档进行评分，并且随着命中次数的增加，为查询评分的成本也会增加。检查与数百万个文档匹配的任何常见查询。这种情况的例外是“恒定分数查询”。查询（例如形式为“：”的查询）会完全绕过计分过程。
过于复杂的查询	定义什么构成过于复杂的查询是很难做到的，但是一个非常普遍的规则是，长度超过 1024 个字符的查询可能过于复杂。
高自动预热时间	自动预热是填充缓存的过程。在新的搜索者为第一个实时用户请求提供服务之前，会运行一些查询。这使前几个用户不必等待。自动预热可能需要几秒钟，也可能是瞬间的。过多的自动预热时间通常表示过大的自动预热参数。过度的自动预热通常带来的好处是有限的，而更长的运行时间实际上是在浪费工作。 高速缓存自动热身。每个 Solr 缓存都有一个 autowarm 参数。通常，您可以将此值设置为上限 128，然后从那里进行调整。 `FirstSearcher/NewSearcher`。该 `solrconfig.xml` 文件包含在打开新的搜索器（索引已更新）以及首次启动服务器时可以触发的查询。特别是对于 `firstSearcher`，具有对相关字段进行排序的查询可能会很有价值。 **笔记** 可以从以下位置获得上述标志 `solrconfig.xml`
例外情况	Solr 日志文件包含所有引发的异常的记录。某些异常（例如由无效查询语法导致的异常）是良性的，但其他异常（例如“内存不足”）则需要引起注意。
缓存过大	诸如过滤器高速缓存之类的高速缓存的大小受 maxDoc/8 限制。例如，具有 10,000 个条目的 filterCache 可能会导致内存不足错误。如果要索引很多文档是正常现象，并且预期会出现大型缓存。
具有低命中率的缓存，尤其是 filterCache	每个缓存占用一些空间，消耗资源。有几个缓存，每个缓存都有自己的命中率。 `filterCache`。此缓存应具有较高的命中率，通常约为 80%。 `queryResultCache`。这主要用于分页，因此命中率可能非常低。每个条目都非常小，因为它基本上由原始查询（作为键的字符串）和大约 20 至 40 `int`s 组成。尽管很有用，除非用户正在体验分页，否则这需要相对较少的关注。 `documentCache`。这个缓存有点棘手。它用于缓存文档数据（存储的字段），因此请求处理程序中的各个组件不必从磁盘重新读取数据。这是一个悬而未决的问题，它在 `MMapDirectory` 用于访问索引时有多有用。

问题	推荐
很深的分页	用户很少会超出首页，很少会浏览 100 页的结果。一个`&start=<pick your number>`查询指出应查明不寻常的用法。深度分页可能表明某些代理正在完成抓取。 **笔记** 不管深度如何，Solr 都不是为了返回完整的结果集而构建的。如果需要返回完整的结果集，请探索替代方法以对整个结果集进行分页。
范围查询应该在 `trie` 字段上工作	`Trie` 字段（数字类型）在索引中存储额外的信息，以帮助进行范围查询。如果使用范围查询，使用 `trie` 字段几乎总是一个好主意。
`fq` 现在使用裸露的子句	`fq` 子句保存在缓存中。缓存是从 `fq` 子句到满足该子句的集合中文档的映射 。使用裸 NOW 子句实际上保证了过滤器缓存中的条目不会被重复使用。
多个同时搜索者升温	这表明提交次数过多或自动预热花费的时间太长。这通常表示您对何时应该提交提交（通常是模拟近实时（NRT）处理）或索引客户端未正确完成提交的误解。使用 NRT 时，提交应该非常少见，并且不应同时进行多个自动预热。
永不返回的存储字段（`fl=`子句）	检查查询 `fl=`并将其与模式相关联可以判断是否指定了未使用的存储字段。这主要是浪费磁盘空间。并且 `fl=*`可以使这个模棱两可。尽管如此，还是值得研究的。
从未搜索过的索引字段	这与从不返回存储字段的情况相反。这一点更为重要，因为这会带来真正的 RAM 后果。检查请求处理程序中的“ edismax”样式解析器，以确保未使用索引字段。
查询但未分析的字段	很少要查询但不进行任何分析的字段。通常，这仅对适用于机器输入数据的“字符串”类型字段（例如，从选择列表中选择的零件号）有用。未经分析的数据不应用于人类输入的任何内容。
字符串字段	字符串字段是完全未分析的。不幸的是，有些人 `string` 对 Java 的 `String` 类型感到困惑 ，并将其用于应标记化的文本。通常的期望是应谨慎使用字符串字段。不仅仅是几个字符串字段表明了设计缺陷。
模式和索引不同步	Solr 使用该架构设置有关索引的期望。更改架构时，不会尝试将更改改编为当前已建立索引的文档，但是会使用新的架构定义来为所有新文档建立索引。因此，新旧文档可以在同一字段中以完全不同的格式存储（例如 `String` 和 `TrieDate`），从而使索引不一致。可以通过检查原始索引来检测。无论何时更改架构，都要重新索引整个数据集。
查询统计	可以从日志中提取查询统计信息。可以在实时系统上监视统计信息，但是拥有日志文件更为常见。以下是您可以收集的一些统计信息： 运行时间最长的查询 0 长度查询 平均/平均/最小/最大查询时间 您可以了解一定时间间隔（查询的时间或数量）中提交对后续查询的影响，以查看提交是否是间歇性减速的原因

问题	推荐
表现不理想	从历史上看，过于频繁的提交是导致性能不令人满意的原因。这对于 NRT 处理不是很重要，但值得考虑。

7.7.5. Cloudera Search 配置和日志文件

Cloudera Search（由 Apache Solr 支持）的配置使用 Cloudera Manager 进行管理。您可以在单个 Solr 服务器上或使用 Cloudera Manager 查看日志文件。

7.7.5.1. Cloudera Search 配置文件

Cloudera Search 配置主要由几个配置文件控制，这些文件大多数存储在 Apache ZooKeeper 中。

表 1. Cloudera Search 配置文件

配置文件	描述
solr.xml	该文件存储在 ZooKeeper 中，并控制 Apache Solr 的全局属性。要编辑此文件，必须从 ZooKeeper 下载该文件，进行更改，然后使用以下 solrctl cluster 命令将修改后的文件上传回 ZooKeeper 。有关该 solr.xml 文件的信息 ，请参阅 Solr 文档中的 Solr 配置文件和 Solr 核心以及 solr.xml。
solrconfig.xml	Solr 中的每个集合都使用 solrconfig.xml 存储在 ZooKeeper 中的文件来控制集合行为。有关该 solrconfig.xml 文件的信息 ，请参阅 Solr 文档中的 Solr 配置文件和配置 solrconfig.xml。
managed-schema 或者 schema.xml	Cloudera 建议使用托管模式，并使用 Schema API（Apache Solr 文档）进行模式更改。集合使用托管模式或旧 schema.xml 文件。这些文件（也存储在 ZooKeeper 中并分配给集合）定义了要建立索引的文档的架构。例如，它们指定要索引的字段，每个字段的预期数据类型，未指定字段时要查询的默认字段，等等。有关 managed-schema 和的信息 schema.xml，请参见 Solr 文档中 SolrConfig 中的架构工厂定义。
core.properties	与其他配置文件不同，此文件存储在本地文件系统中，而不是 ZooKeeper 中，并用于核心发现。有关此过程和文件结构的更多信息，请参见 Solr 文档中的定义 core.properties。
附加文件	文件中引用的任何其他文件 xml，例如，自定义 JAR 文件。

7.7.5.2. 查看和修改搜索配置

了解有关在 Cloudera Manager 中查看和编辑 Solr 服务的配置参数的信息 。

1) 转到 Solr 服务>配置。

2) 使用“范围”和“类别” 过滤器来限制显示的配置参数。您也可以在“搜索”字段中输入文本以动态过滤配置参数。

3) 进行更改后，输入更改原因，然后点击保存更改。

4) 重新启动 Solr 服务（Solr 服务>操作>重新启动）以及所有相关服务。您还可以使用“重新启动旧服务”向导来重新启动相关服务。

7.7.5.3. Cloudera Search 日志文件

Cloudera Search 有几个日志文件，存储在每个 Solr Server 主机上。请参阅以下有关日志文件位置以及每个文件的简要说明。可以按照 Cloudera Manager 中的描述查看或修改引用的配置参数。

7.7.5.3.1. /var/log/solr/下的日志

/var/log/solr/是大多数“搜索”日志的父目录。

表 1. /var/log/solr/**下的日志和目录**

日志或目录	描述	配置选项
audit/	审核日志目录。	该路径可以由“审核日志目录”配置参数指定。
stacks/	放置堆栈日志的目录。	如果选中了“启用堆栈收集”选项，则收集日志。 堆栈收集目录-放置堆栈日志的目录。如果未设置，则堆栈将登录到 stacks 角色日志目录的子目录中。
solr-cmf-*SOLR-1*-SOLR_SERVER-*hostname.example.com*.log.out	Solr 服务器日志文件。该 *SOLR-1* 和 *hostname.example.com* 部分取决于您的环境。	Solr Server 日志记录阈值参数设置 Solr Server 角色的日志级别 。 网关日志记录阈值参数设置网关角色的日志级别 。
solr_gc_log.*	Solr Server 进程的 Java 垃圾收集（GC）日志。	

7.7.5.3.2. /var/run/cloudera-scm-agent/process/<process_dir_id>-solr-SOLR_SERVER/下的日志

/var/run/cloudera-scm-agent/process/*<process_dir_id>*-solr-SOLR_SERVER/是当前正在运行的 Solr Server 角色的配置目录。*<process_dir_id>*每次重新启动 Solr Server 时，该 部分都会更改。

警告

不要修改此目录中的任何文件。这些文件由 Cloudera Manager 自动生成。要修改任何配置参数，请使用 Cloudera Manager。

表 2.下的日志/var/run/cloudera-scm-agent/process/*<process_dir_id>*-solr-SOLR_SERVER/

日志目录	描述
logs/	日志目录，其中包含 Solr Server 进程的 stderr 和 stdout 日志。

要标识当前目录或最新目录，请/var/run/cloudera-scm-agent/process/*solr* 按时间顺序列出按时间排序的目录，如下所示：

```
sudo ls -ltrd/var/run/cloudera-scm-agent/process/*solr*
```

底部的条目是当前或最近的进程目录。

7.7.5.4. 查看和修改搜索和相关服务的日志级别

了解有关查看和修改 Search 和相关服务（例如 Apache HBase）的日志级别的信息。

1) 进入服务配置页面。例如：Solr 服务>配置。
2) 选择“类别” >“日志”过滤器。
3) 修改服务角色的日志记录阈值。可用的选项（以详细程度降序显示）是：
    a. 痕迹

b. 调试
c. 信息
d. 警告
e. 错误
f. 致命

4) 输入更改原因，然后单击“保存更改”。
5) 重新启动 Solr 服务（Solr 服务>操作>重新启动）以及所有相关服务。

7.8.对 Hue 进行故障排查

7.8.1. Hue 负载平衡器无法在各个 Hue 服务器之间平均分配用户

Hue 负载平衡器会将新用户重定向到集群上新添加的 Hue 服务器，并将现有用户重定向到现有的 Hue 服务器。即使您添加了更多的 Hue 服务器来满足不断增长的用户群，资源也可能无法得到有效利用。

Hue 负载平衡器的任务是在可用的 Hue 服务器之间平均分配用户，以有效利用资源。但是，由于会话的持久性，它不能平均分配用户。要解决此问题，您可以从 Clouder a Manager 刷新 cookie。

负载平衡器将 cookie ROUTEID 与来自浏览器的随机字符串一起使用，并存储在 `hu e.conf` 文件中。此随机字符串用于将用户重定向到 Hue 服务器。要刷新 Cookie 并在每次添加新的 Hue 服务器时设置新的随机字符串，请执行以下操作：

1) 转到 Cloudera Manager >集群>Hue>配置。
2) 单击“作用域” >“负载均衡器”，然后选择“ Hue 负载均衡器 Cookie 刷新”复选框。
3) 这将刷新 `hue.conf` 文件中的 cookie 值，以重新平衡 Hue 后端连接。
4) 在“实例”选项卡中，选择所有 Hue 服务和角色，然后单击“服务操作” >“重新启动”。

这将为 cookie 创建一个新的随机字符串，负载均衡器现在可以使用该字符串来平均分配用户。

每当您重新启动 Hue 服务器时，负载均衡器都会根据服务器负载平均重新分配用户。

7.8.2. 无法使用 SAML 对 Hue 中的用户进行身份验证

如果已将 SAML 配置为对用户进行身份验证，但是用户无法使用单一登录（SSO）登录 Hue，则可能不支持 RSA 密钥格式。要解决此问题，可以使用不受保护的私钥，然后在安全阀中指定私钥文件名。

1) `.key` 使用以下命令将文件转换为不受保护的私钥文件：

```
openssl rsa -in/opt/cloudera/security/<file name>.key -
out/opt/cloudera/security/<file name_unprotected>.key
openssl rsa -in/opt/cloudera/security/hadoop-cpi-prod.key -
out/opt/cloudera/security/hadoop-cpi-prod_unprotected.key
```

2）更新高级配置代码段，如以下示例所示：

```
[libsaml]
xmlsec_binary=/usr/bin/xmlsec1
metadata_file=/opt/cloudera/security/saml/idp-openam-metadata.xml
key_file=/opt/cloudera/security/hadoop-cpi-prod_unprotected.key
cert_file=/opt/cloudera/security/hadoop-cpi-prod.pem
```

7.8.3. 清理旧数据以提高性能

Hue 中的某些表会无限期保留数据，从而导致性能降低或应用程序崩溃。Hue 不会自动清除这些表中的数据。您可以将 Hue 配置为将数据保留特定的天数，然后安排 cron 作业定期清理这些表以提高性能。

如果在使用 Hue 时遇到以下问题，请考虑从后端 Hue 数据库中清除旧数据：

- 升级超时
- 性能慢于预期
- 很长一段时间登录到顺化
- SQL 查询在表格中显示大量文档
- 尝试访问已保存的文档时，Hue 崩溃

开始清理活动之前，请备份数据库。检查一些用户的查询和工作流等已保存的文档，以防止数据丢失。您还可以通过运行以下查询来记录要清理的表的大小，以作为参考：

```
select count(*) from desktop_document;
select count(*) from desktop_document2;
select count(*) from beeswax_session;
select count(*) from beeswax_savedquery;
select count(*) from beeswax_queryhistory;
select count(*) from oozie_job;
```

笔记

可以存储在表格中的最佳文档数小于或等于 30,000。指定清除间隔时，请考虑此数字。

1) SSH 到活动的 Hue 实例。

2) 转到“Hue 主目录”：

```
cd /opt/cloudera/parcels/CDH/lib/hue
```

3) 以 root 用户身份运行以下命令：

```
DESKTOP_DEBUG=True ./build/env/bin/hue desktop_document_cleanup --
keep-days x
```

该`--keep-days`属性用于指定 Hue 将在后端数据库中保留数据的天数。

```
DESKTOP_DEBUG=True ./build/env/bin/hue desktop_document_cleanup --
keep-days 30
```

在这种情况下，Hue 将保留数据 30 天。

由于`DESKTOP_DEBUG`设置为，日志显示在控制台上`True`。或者，您可以从以下位置查看日志：

```
/var/log/hue/desktop_document_cleanup.log
```

通常，每个表中的每 1000 个条目第一次运行大约需要 1 分钟。

4) 通过运行查询，检查表大小是否已减小，如下所示：

```
select count(*) from desktop_document;
```

如果`desktop_document_cleanup`命令已成功运行，则表大小应减小。

设置定期运行的 cron 作业，以自动进行数据库清理。例如，您可以将 cron 作业设置为每天运行，并清除早于 *x* 天数的数据。

7.8.4. 无法使用提供的凭据连接到数据库

将 Hue 服务添加到集群时，Cloudera Manager 会测试数据库连接。该测试数据库连接不工作，为那些需要服务名称，而不是 Oracle 系统 ID（SID）的 Oracle 数据库。这可能会阻止您将 Hue 服务添加到集群中。

如果在通过 Cloudera Manager 添加 Hue 服务时遇到以下错误，请按照本主题中所述的解决方法进行操作：

```
Unable to connect to database with provided credential. Able to find the
Database server, but not the specified database. Please check if the database
name is correct and make sure that the user can access the database.
```

1) 临时安装其他数据库实例（例如 MySql）以与 Hue 一起使用。这称为 Hue 数据库。
2) 从 Cloudera Manager 添加 Hue 服务，并指定您在上一步中创建的 Hue 数据库详细信息。

这样，您就可以跳过“添加服务”向导，并将“Hue”服务添加到您的集群中。

3) 修改 Hue 实例以使用实际的 Oracle 数据库，如下所示：
   a) 导航到 Cloudera Manager >集群> Hue 服务>配置>类别>数据库。

显示数据库配置字段。

   b) 通过配置以下字段来设置 Oracle 数据库：
      i. 选择 Oracle 作为 Hue 数据库类型。
      ii. 在“Hue 数据库主机名”字段中，指定已安装 Oracle 数据库的主机的完全合格域名（FQDN）。
      iii. 在“Hue 数据库端口”字段中，指定运行 Oracle 数据库的主机上的端口。通常，此值为 1521。
      iv. 在“Hue 数据库用户名”字段中，指定用于登录 Oracle 数据库的用户名。
      v. 在“ Hue 数据库密码”字段中，指定数据库密码。

vi. 在“Hue 数据库名称”字段中，以以下格式指定 Hue 数据库的名称：

```
<HUE_DB_HOST>:1521/<servicename>
```

c) 导航到 Cloudera Manager >集群> Hue 服务>配置>类别>高级，并在 hue_safety_valve.ini 的“ Hue 服务高级配置代码段（安全阀）”中指定以下内容：

```
[desktop]
[[database]]
port=0
```

4) 点击保存更改。

5) 通过单击“操作” >“重新启动”来重新启动 Hue 服务。

7.8.5. 在 Hue UI 上激活 Hive 查询编辑器

如果尚未在集群上安装并选择 HIVE_ON_TEZ 服务，则在 Hue 用户界面上可能看不到 Hive 查询编辑器。需要 HIVE_ON_TEZ 服务，才能将 Hive 与 Hue 一起配置和使用。

笔记

在 CDH 6 和更早版本中，Hive 服务包括 Hive Metastore 和 HiveServer2。在 Cloudera Runtime 7.0 和更高版本中，此服务仅包含 Hive Metastore。HiveServer2 和 Hive 执行引擎的其他组件是 HIVE_ON_TEZ 服务的一部分。

要在 Hue Web UI 上启用 Hive 查询编辑器，请执行以下操作：

1) 以管理员身份登录到 Cloudera Manager。

2) 检查集群中是否安装了 HIVE_ON_TEZ 服务。

   如果尚未安装，请将其添加为服务。

3) 转到集群> Hue 服务>配置。

4) 将显示 Hue 配置列表。

5) 搜索“ HiveServer2 服务”字段，然后选择“ HIVE_ON_TEZ”服务。

6) 保存更改并重新启动 Hue。

   Hive 查询编辑器现在应该在 Hue 用户界面上可用。

7.8.6. 查询执行在 Hue 中完成，但显示为在 Cloudera Manager Impala 查询页面上执行

即使查询已在 Hue Web UI 上完成执行，Cloudera Manager 和 Impala 妖魔网页也可能会以执行 或飞行中状态显示查询。由于各种原因，可能会发生这种情况。

已完成的 Hue 查询仍显示为“正在执行”的三个主要原因是：

- 在单击“结果”页面之前，Hue 不会关闭与 Impala 的连接 。
- 单击“Hue”中的“结果”页面将执行 `fetchresults` 对 Impala 的调用。
- Impala 查询是客户端驱动的。因此，查询仍然保持运行状态，直到客户端发送获取命令以完成对整个结果集的获取。
- 如果查询尚未关闭或注销，黑斑羚显示在同一 飞行中在其网站上的用户界面部分。Cloudera Manager 在“执行”状态下显示所有“飞行中”查询。

7.8.6.1. Impala 查询生命周期

提交 Impala 查询时，它们首先由系统注册。系统在协调员的帮助下识别查询。它们还具有状态，例如 CREATED（已创建），INITIALIZED（已初始化），RUNNING（正在运行），FINISHED（已完成），EXCEPTION（例外）和一些元数据。

- **已完成**表示行可用，但并非所有行都可以提取。Impala 守护程序可能仍在执行查询。
- **EXCEPTION** 表示已发生错误。例如，如果系统内存不足，则查询将转换为 EXCEPTION 状态。

如果查询被取消，查询也可以进入 EXCEPTION 状态。

可以通过 HiveServer2/Beeswax 调用或查询超时显式触发查询取消。可以通过整个进程的 impalad 参数或每个查询选项来设置查询超时。

当前，Impala 的状态无法明确指示所有 Impala 守护程序是否已完成查询的执行以及是否已获取所有结果。让我们暂时将其称为“声明结束”（EOS）。

当查询处于 EOS（已完成）或 EXCEPTION 状态时，该查询不再进行任何处理，但该查询保持注册状态。它需要保持注册状态，因为客户端可能需要访问状态。

仅在以下两种情况下，查询才被注销：

- 通过 `Close()` API 调用显式关闭查询
- 与查询关联的会话被明确关闭或设置了会话超时并且会话超时

笔记

在您明确关闭查询之前，Hue 不会关闭查询。当关闭在 Hue 中运行查询的浏览器选项卡时，浏览器将发送 JavaScript `Close()` 回调请求以关闭查询。如果您使查询处于无人看管状态（例如，通过关闭用于访问 Hue 的便携式计算机）或浏览器崩溃，则 `Close()` 永远不会将呼叫发送给 Hue。该查询最终可能会超时，但是由于未明确取消查询，因此无法正确清理资源。

为了优化资源利用，通过在`--idle_session_timeout`impalad 参数中设置会话超时值，将 Impala 守护程序配置为杀死空闲会话：

1) 以管理员身份登录到 Cloudera Manager。
2) 转到集群> Impala 服务>配置。
3) 在“ Impala 命令行参数高级配置代码段（安全阀）”中指定以下内容 ：

```
--idle_session_timeout=<maximum lifetime of your queries in seconds>
```

例如，

```
--idle_session_timeout=3600
```

在这种情况下，查询将在一小时后超时。

7.8.7. 查找 Hue 超级用户列表

您可以通过使用带有 Python 代码的 Hue shell 或在 `auth_user` 表上运行 SQL 查询来获取超级用户列表。

7.8.7.1. 使用 Hue Shell 和 Python 代码查找 Hue 超级用户

1) 通过运行以下命令连接到 Hue shell：

```
/opt/cloudera/parcels/CDH/lib/hue/build/env/bin/hue shell --cm-managed
```

2) 输入 Python 代码，如下所示：

```
from django.contrib.auth.models import User
print "%s" % User.objects.filter(is_superuser = True)
```

样本输出：

```
<QuerySet [<User: admin>]>
```

7.8.7.2. 在 auth_user 表上运行 SQL 查询以查找 Hue 超级用户

1) 通过运行以下命令连接到 Hue dbshell：

```
/opt/cloudera/parcels/CDH/lib/hue/build/env/bin/hue dbshell --cm-
managed
```

2) 运行以下 SQL 查询：

```
select username, is_superuser from auth_user where is_superuser=1;
```

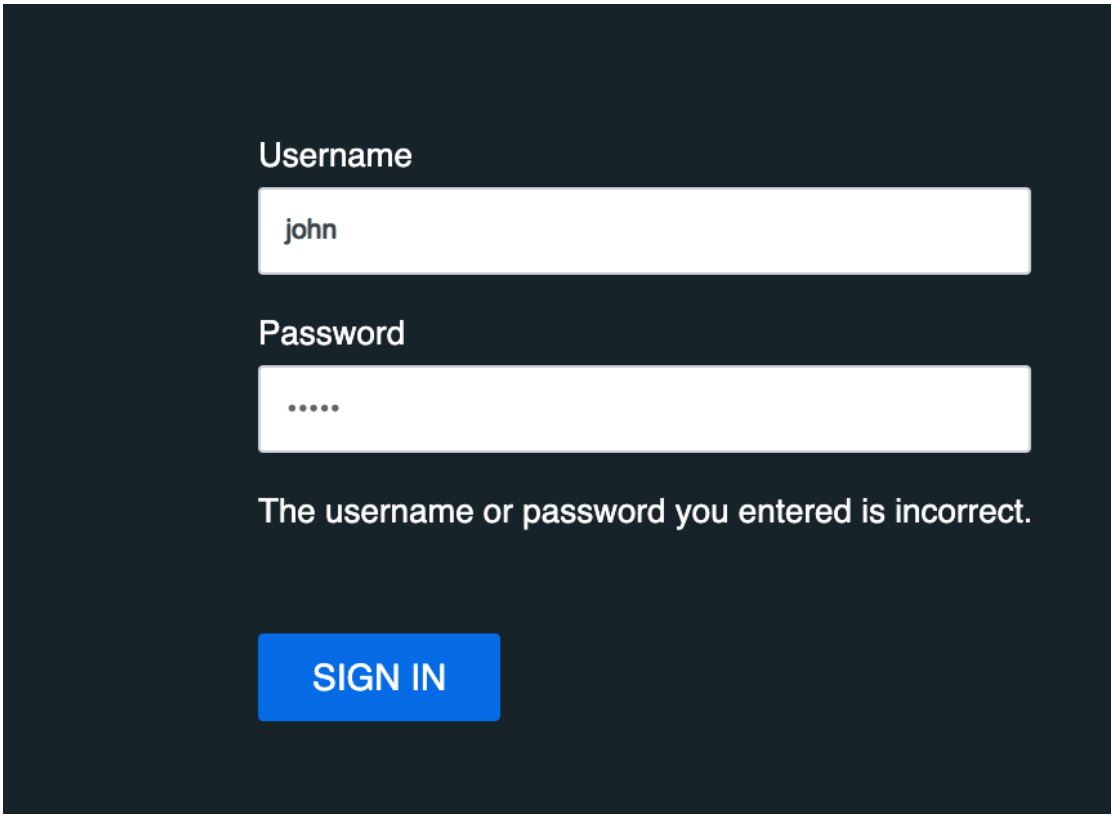
样本输出：

```
----------------------+

username   is_superuser
----------------------+

admin 1
----------------------+
1 row in set (0.00 sec)
```

7.8.8. 通过 Knox 访问 Hue 时，用户名或密码不正确

当您尝试使用 Knox UI 登录到 Hue 时，如果出现诸如“您输入的用户名或密码不正确”之类的错误，那么您可以通过使用命令行界面登录到 Knox Gateway 来验证您的凭据。

图 1. Knox Gateway UI：不正确的用户名或密码



1) 打开一个终端会话。

2) 通过输入以下命令，SSH 进入 Knox 网关主机：

```
ssh [***KNOX-USERNAME***]@[***KNOX-HOST***].[***DOMAIN***].site
```

替换 *KNOX-HOST*。具有您的 Knox 网关主机的完全限定域名（FQDN）的 *domain* . site。

例如：

```
ssh john@abc-1.example.com
```

3) 输入您在 Knox Gateway Web UI 上使用的密码。

如果您能够使用这些凭据登录，则应该能够登录到 Knox Gateway UI。

7.8.9. 从 Knox 访问 Hue UI 时出现 HTTP 403 错误

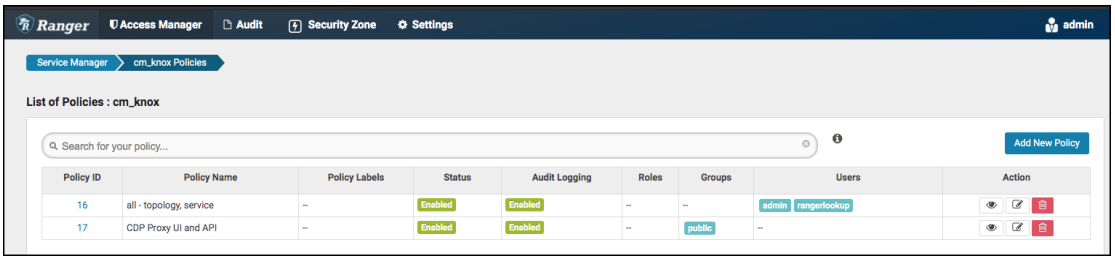
从 Knox 网关访问 Hue UI 时，如果出现 HTTP 403 错误，则该用户或该用户所属的组可能没有所需的权限。

1) 以管理员身份登录到 Cloudera Manager。

2) 转到 ClustersRanger 服务> Ranger Admin Web UI，然后以管理员身份登录到 Ranger Admin Web UI。

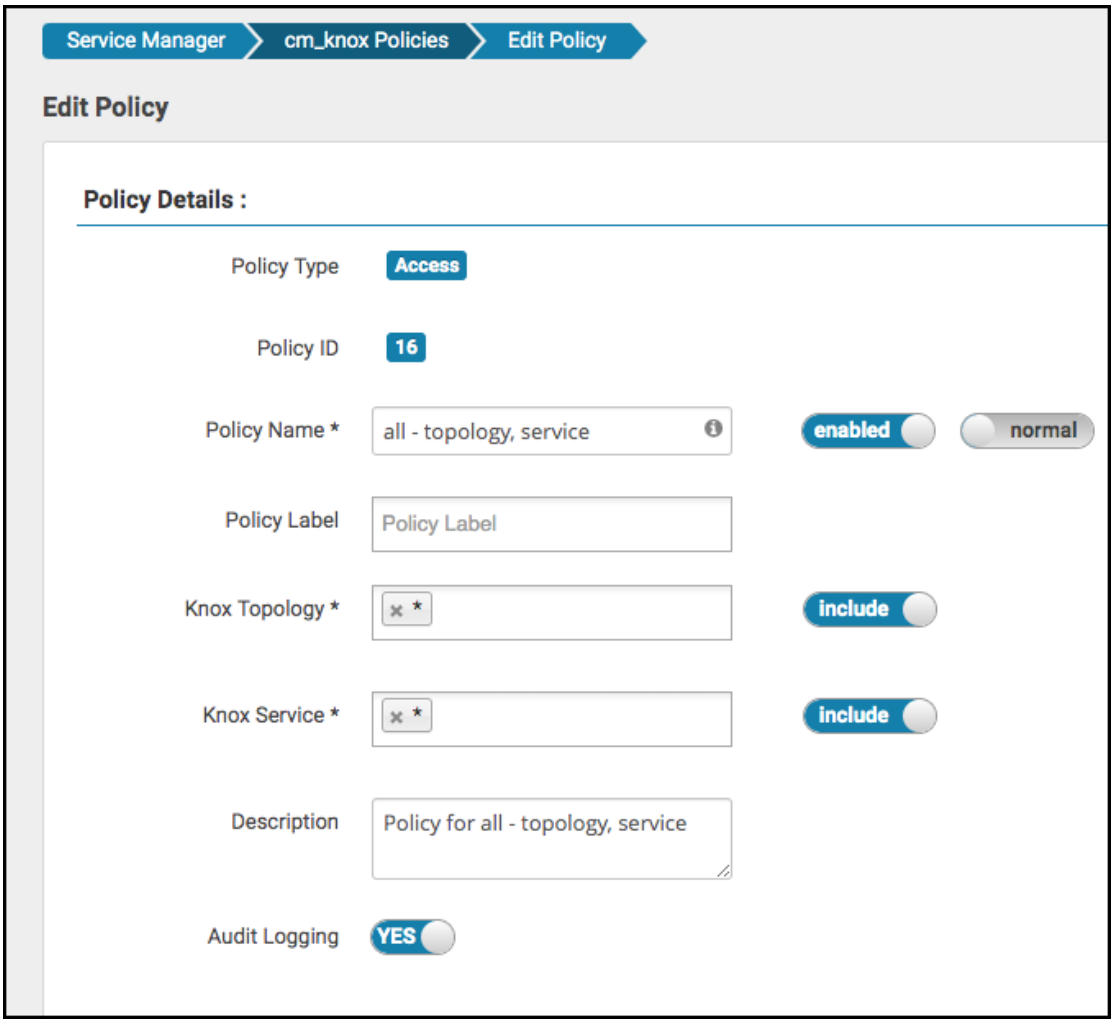
3) 点击 cm_knox。

4) 显示 cm_knox 策略。



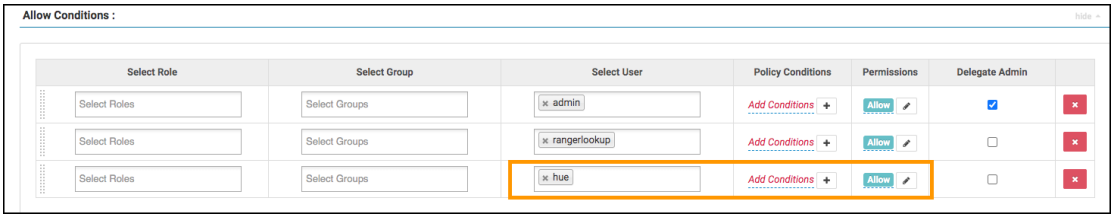
5) 单击策略 ID 16（全部-拓扑，服务）。

进入“编辑策略”页面。



6) 验证用户和该用户所属的组是否具有必需的权限。

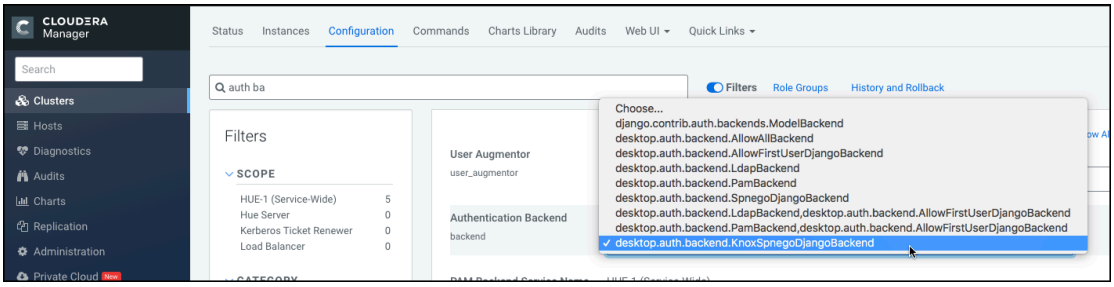
如果用户或组没有所需的权限，则将用户或用户组添加到“选择用户”字段，然后在“权限”下选择“ 允许”。



7.8.10. 无法从 Knox Gateway UI 访问 Hue

如果无法从 Knox Gateway UI 访问 Hue，则可能未为 Hue 配置 KnoxSpnegoDjango Backend 属性。即使在集群上启用了 Knox，Cloudera Manager 也不默认将身份验证后端设置为 KnoxSpnegoDjangoBackend。

1) 以管理员身份登录到 Cloudera Manager。
2) 转到集群> Hue 服务>配置，然后搜索身份验证后端 字段。
3) 从下拉列表中选择 desktop.auth.backend.KnoxSpnegoDjangoBackend。



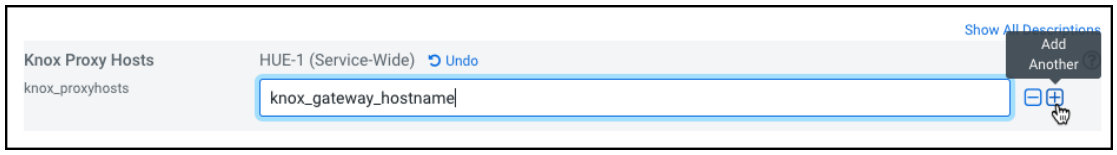
4) 点击保存更改。
5) 转到集群> $ Knox 服务>实例，并记下 Knox 网关的主机名。

您必须在下一步中提供这些详细信息。

如果您以高可用性（HA）模式设置了 Knox，则您可以在“实例” 选项卡上看到多个 Knox 网关。

6) 返回“集群” >“ Hue 服务” >“配置”，然后搜索“ Knox 代理主机” 字段。
7) 输入您先前记下的 Knox 网关的主机名。

如果已设置 Knox HA，则单击+添加另一个主机名。



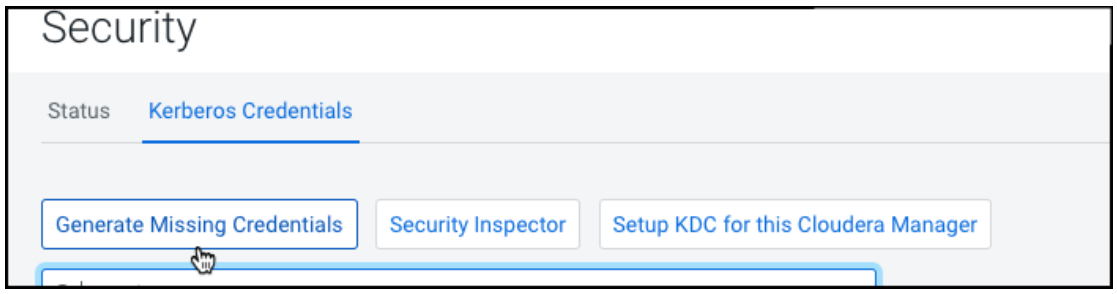
8) 点击保存更改。

您将看到以下警告：

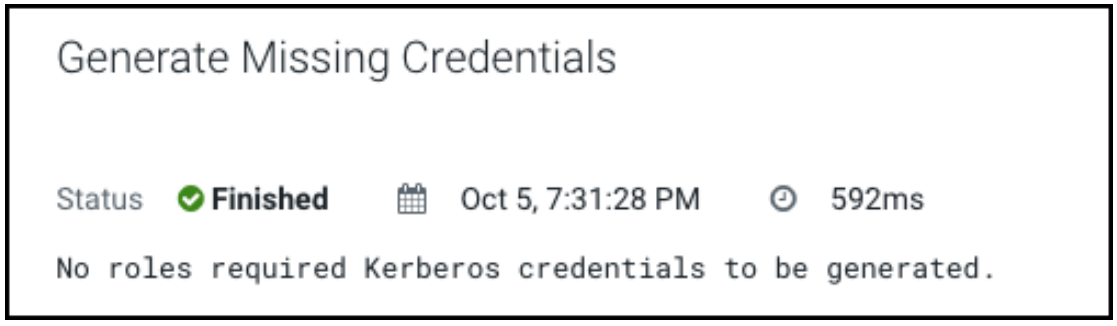
```
Role is missing Kerberos keytab. Go to the Kerberos Credentials page
and click the Generate Missing Credentials button.
```

9) 点击管理的 Cloudera 的经理左侧导航面板中，选择安全性。

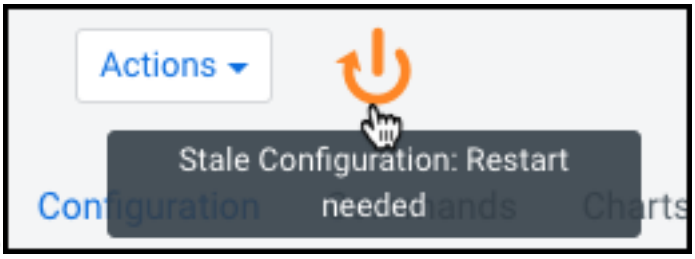
10)转到“ Kerberos 凭据”选项卡，然后单击“ 生成缺少的凭据”。



将显示一个弹出窗口，显示状态。



11)转到“集群” >“ Hue 服务”，然后单击“ 操作” 旁边的“重新启动”。

12) 

13)

14)在 Stale Configurations 页面上，单击 Restart Stale Services。

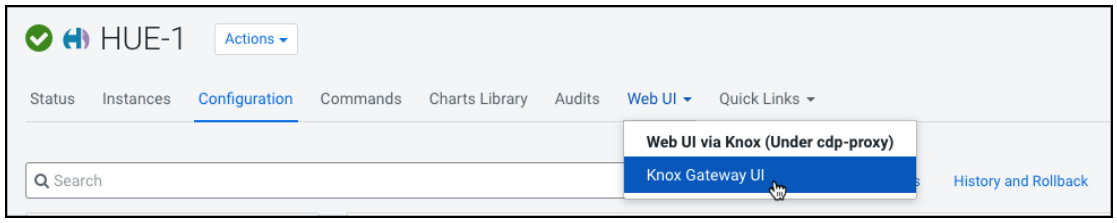
显示“重新启动旧服务”向导。

15)在“查看更改”页面上，选择“重新部署客户端配置”，然后单击“立即重新启动”。

“命令详细信息”页面显示服务重新启动时的实时状态。

完成所有步骤后，点击完成。

16)在 Hue 服务页面上，单击 Web UI > Knox Gateway UI。

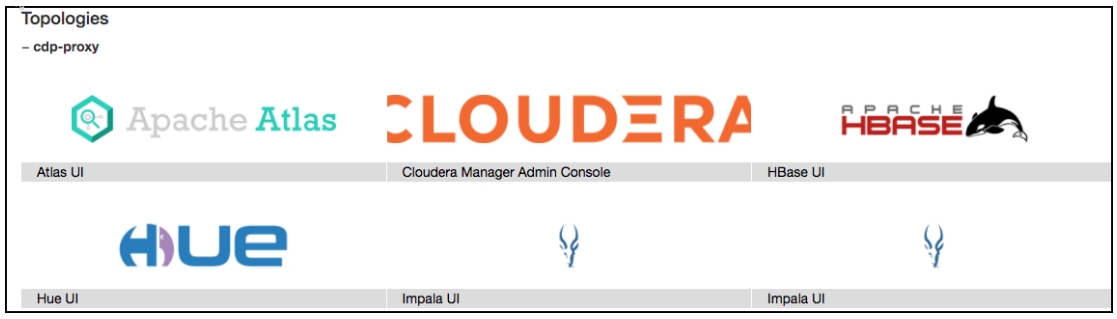


显示 Knox Gateway 用户界面。

17)在一般代理信息页面，通过点击展开 CDP 代理拓扑+ CDP 代理下的 拓扑结构。

显示使用 cdp-proxy 拓扑配置的服务列表。

18)单击 Hue 徽标。



您应该能够登录 Hue Web UI。

您也可以使用以下 URL 登录到 Hue：

```
https://[***HOSTNAME***]:[***PORT***]/gateway/cdp-proxy/hue/
```

7.8.11. 引荐检查失败，因为域与任何受信任的来源都不匹配

access.log 如果使用 Knox 网关 UI 访问 Hue 时 Knox 网关 DNS 无法正确解析，则您可能会在 Hue 文件中看到“引用检查失败”错误。

如果在 Hueaccess.log 文件中看到以下错误，请执行本主题中列出的步骤：

```
”POST/accounts/login HTTP/1.1" — Referer checking failed -
https://<ip_address>:<knoxui_port>/gateway/cdp-
proxy/hue/hue/accounts/login?next=%2F%253FdoAs%253Dknoxui does not
match any trusted origins.”
```

由于各种原因，Knox 网关 DNS 可能无法解析。在继续操作之前，请验证以下内容：

- 验证是否在 Hue 配置中正确设置了 Knox 代理用户配置
- 检查在“ Hue 服务” >“配置” >“ Knox 代理主机”字段中指定的主机名是否拼写正确
- 通过登录 DNS 服务器并 ping 几台主机，确保 DNS 设置正确。
- 确保/etc/hosts 文件具有 IP 地址和主机的正确映射

1) 转到集群> $ Knox 服务>实例，然后单击 Knox 网关主机名。
2) 从“详细信息” 部分记下 Knox 网关主机的 IP 地址 。
3) 转到“集群” >“ Hue 服务” >“配置”，然后搜索“ Knox 代理主机”字段。
4) 输入您先前记下的 Knox 网关主机的 IP 地址。
5) 点击保存更改。
6) 验证是否可以通过以下 URL 访问 Knox 网关：

```
https://[***IP_ADDRESS***]:[***KNOXUI_PORT***]/gateway/cdp-
proxy/hue/hue/accounts/login?
```

7.8.12. 无法查看 Snappy 压缩文件

您必须 python-snappy 在集群上安装该库，才能使用 Hue File Browser 和 HBase Browser 浏览使用 Snappy 压缩的文件。安装后，Hue 会自动检测并显示 Snappy 压缩文件。

该 python-snappy 库与名为的 python 库不兼容 snappy。snappy 如果集群中存在该组件，则必须将其卸载。

运行以下命令，以检查 snappy 集群中是否安装了该库：

```
/usr/bin/pip show snappy
```

控制台上没有输出表明该 snappy 库未安装在您的集群上。如果您获得的任何结果 snappy，请通过运行以下命令将其卸载：

```
/usr/bin/pip uninstall snappy
```

接下来，python-snappy 通过运行以下命令检查集群中是否安装了该库：

```
/usr/bin/pip show python-snappy
```

样本输出：

```
Name: python-snappy
Version: 0.5.4
Location:/usr/lib64/python2.7/site-packages
```

1) 以管理员身份登录到 Cloudera Manager。
2) 通过转到集群>Hue 服务>操作来停止 Hue 服务。
3) 根据是否使用 Parcel 或 Parcel 来设置 CDH 集群，请转到以下目录。

对于 Parcel：

```
cd/opt/cloudera/parcels/CDH/lib/hue
```

对于 Package：

```
cd/usr/lib/hue
```

python-snappy 通过运行以下命令来安装软件包：

```
yum install gcc gcc-c++ python-devel snappy-devel
./build/env/bin/pip install -U setuptools
./build/env/bin/pip install python-snappy
```

1) python-snappy 通过运行以下命令，验证所有用户都可以读取该库：

```
ls -lart `locate snappy.py`
```

输出应类似于以下内容：

```
-rw-r--r-- 1 root root 11900 Sep  1 12:25/usr/lib64/python2.7/site-
packages/snappy.py
-rw-r--r-- 1 root root 10344 Sep  1 12:26/usr/lib64/python2.7/site-
packages/snappy.pyc
```

2）通过转到集群>Hue 服务>操作来启动 Hue 服务。

3）python-snappy 通过运行以下命令来验证该库是否适用于 Hue：

```
sudo -u hue/bin/bash -c "echo 'import snappy' | python"
```

如果 python-snappy 库按预期运行，则此命令将不显示任何输出。

您应该能够使用 Hue Web 界面在 Hue File Browser 和 HBase Browser 上查看 Snappy 压缩文件。

7.8.13. 启用 SAML 时出现“未知属性名称”异常

当 SAML 身份提供程序（IdP）返回'uid'配置文件属性，并且使用 pysaml2 的 Hue 无法解释此属性时，您可能会看到“未知属性名称”异常。要解决此问题，您必须创建一个属性映射文件，然后在 Hue 的 libsaml 配置中引用它。

1) 以 root 用户身份通过 SSH 进入 Hue 服务器。

2) 创建一个属性映射目录，如下所示：

```
mkdir -p /opt/cloudera/security/saml/attribute_mapping
```

3) 创建一个属性映射文件，如下所示：

```
Vi /opt/cloudera/security/saml/attribute_mapping/saml_uri.py
```

4) 在 saml_uri.py 文件中添加以下行：

```
MAP = {
    "identifier": "urn:oasis:names:tc:SAML:2.0:attrname-format:uri",
    "fro": {
        'uid': 'uid',
        },
    "to": {
        'uid': 'uid',
    }
}
```

5) 在所有 Hue 主机上重复步骤 1 至 4。

6) 以管理员身份登录到 Cloudera Manager。

7) 转到集群> Hue 服务>配置。

8) 在 hue_safety_valve.ini 字段的“ Hue 服务高级配置代码段”（安全阀）中添加以下行：

```
[libsaml]
xmlsec_binary=/usr/bin/xmlsec1
```

```
metadata_file=/opt/certs/saml/FederationMetadata.xml
key_file=/opt/certs/hue.key
cert_file=/opt/certs/hue.crt
entity_id=hue-pri.unedic.intra
logout_enabled=false
username_source=attributes
attribute_map_dir=/opt/cloudera/security/saml/attribute_mapping
#user_attribute_mapping='{"uid":"username"}'
```

9) 点击保存更改。

10)通过单击“操作” >“重新启动”来重新启动 Hue 服务。

用户现在应该能够通过 SAML 向 Hue 进行身份验证。

7.8.14. Impala 查询因无效的查询句柄错误而失败

从 Hue Web 界面运行 Impala 查询时，会遇到“无效查询句柄”错误，因为 Impala Thrift 服务器与 Hue Load Balancer 之间的连接超时。这由 `server_conn_timeout` 财产支配。

该 `server_conn_timeout` 属性的默认值为 30 分钟。您可以通过使用 Cloudera Manager 更新 Hue 配置来增加超时限制。

1) 以管理员身份登录 Cloudera Manager。
2) 转到 hue_safety_valve.ini 的集群> Hue 服务>配置> Hue 服务高级配置代码片段（安全阀）。
3) `server_conn_timeout` 如下所示，在“ Impala”部分中增加属性的值：

```
[impala]
   server_host=[***SERVER-HOST***]
   server_port=[***PORT***]
   server_conn_timeout=[***TIMEOUT-IN-SECONDS***]
```

您可以将 `server_conn_timeout` 属性的值增加到 2 小时（7200 秒）。

4）点击保存更改。

5）重新启动 Hue 服务。

您应该能够从 Hue Web 界面成功运行 Impala 查询。

7.8.15. PostgreSQL 支持的服务失败或挂起

当 CDP 服务与 PostgreSQL 数据库之间的连接数超过预设的连接限制时，新连接可能会失败，Cloudera Manager 会挂起，并且您无法登录 Hue。日志显示“ `FATAL: remaining connection slots are reserved for non-replication superuser connections`”错误。

CDP 服务和 PostgreSQL 数据库之间的连接数由该 `max_connections` 设置控制。默认情况下，与 PostgreSQL 数据库的最大可用连接数为 115。为超级用户保留了 15 个连接，以维护数据库的状态和完整性，为 CDP 和其他服务提供 100 个连接。

笔记

由于在启动其他 CDP 服务之后 Cloudera Manager 会启动 Hue 服务，因此在 Hue 中发生此问题的可能性更高。因此，与共享同一数据库的其他服务相比，Hue 服务与 PostgreSQL 的连接相对较少。

1) 检查可用和空闲的连接数：
   a) `psql` 以 admin 用户身份从命令行客户端 SSH 进入 PostgreSQL 数据库 。
   b) 运行以下查询以检查空闲连接数：

```
SELECT datname, count(datname) FROM pg_stat_activity WHERE state =
'idle' GROUP BY datname;
```

   c) 运行以下查询以检查当前正在使用的连接数：

```
SELECT datname, count(datname) FROM pg_stat_activity GROUP BY datname;
```

   d) 运行以下命令以查看最大连接数：

```
show max_connections;
```

   e) 运行以下查询以了解连接的去向：

```
SELECT datname, numbackends FROM pg_stat_database;
```

2) 如果大多数连接都处于空闲状态并且该 max_connections 值小于 100，则增加文件中的 max_connections 值 postgresql.conf：
   f) 登录 `Cloudera Manager` 并停止所有使用 `PostgreSQL` 数据库的服务。
   g) `SSH` 进入运行 `PostgreSQL` 服务器的主机。
   h) 打开 `postgresql.conf` 文件进行编辑。

该 postgresql.conf 文件通常位于/var/lib/pgsql/data 目录中。但这可能会有所不同，具体取决于您在何处安装了数据库。

i) max_connections 根据以下建议增加值：

每个数据库最多允许 100 个连接，并添加 50 个额外的连接。例如，对于两个数据库，将最大连接数设置为 250。

如果您在一个主机上存储五个数据库（Cloudera Manager Server，Activity Monitor，Reports Manager，Apache Atlas 和 Hive Metastore 的数据库），则将最大连接数设置为 550。

j) 保存更改并退出。

k) 通过运行以下命令来重新启动 PostgreSQL 数据库：

```
pg_ctl restart
```

l) 从 Cloudera Manager 重新启动所有受影响的服务。

如果增加连接限制不能解决您的问题，并且您认为有必要扩大规模，请在其他主机上添加新的 PostgreSQL 实例，然后在数据库管理员（DBA）的帮助下将服务迁移到这些主机上。

7.8.16. 验证 Hue 中的 LDAP 用户时出错

如果您的 CDP 集群上启用了 Hive，则 Hive 可以同时使用 LDAP 和 Kerberos。默认情况下，Hive 使用 LDAP 而不是 Kerberos 来验证 Hue 服务。结果，在登录 Hue Web 界面后或尝试访问 Hive 编辑器时，您可能会看到以下错误：Bad status: 3 (PLAIN auth failed: Error validating LDAP user)。

此外，您可能无法查看数据库或 Hive 表。若要解决此问题，您可以通过 在 Hue 文件中将该 hive.server2.authentication 属性的值配置为，来强制客户端连接（在 Hive 和 Hue 之间）使用 Kerberos 而不是 LDAP 。KERBEROShive-site.xml

1) 以管理员身份登录 Cloudera Manager。
2) 转到 hive-site.xml 的“集群” >“ Hue 服务” >“配置” >“ Hue Server 高级配置代码段（安全阀）”。
3) 单击以 XML 格式查看，然后在文本框中添加以下几行：

```
<property>
    <name>hive.server2.authentication</name>
    <value>KERBEROS</value>
</property>
```

或者，您可以单击+启用编辑器模式，然后 hive.server2.authentication 在“ 名称”字段和 KERBEROS“ 值”字段中指定。

4) 点击保存更改。
5) 重新启动 Hue 服务。

该 hive.server2.authentication 属性将附加到 hive/conf/hive-site.xml 文件中。从现在开始，当您使用 Beeline 访问 Hive 时，Hive 将使用 Kerberos 验证来自 Hue 和 LDAP 中 Hive 编辑器的访问请求。

7.8.17. 从负载均衡器访问 Hue 时出现 502 代理错误

当您从 Hue Load Balancer 访问 Hue 并遇到“ 502 Proxy Error Proxy Error The proxy server received an invalid response from an upstream server. The proxy server could not handle the request POST/desktop/api/search/entities.”错误消息时，请使用 Cloudera Manager 增加 Hue Load Balancer 的代理超时值。

1) 以管理员身份登录 Cloudera Manager。
2) 转到 httpd.conf 的集群> Hue 服务>配置>范围>负载均衡器>负载均衡器高级配置代码段（安全阀）。
3) 在“ httpd.conf 的负载均衡器高级配置代码片段（安全阀）”文本框中添加以下行：

```
ProxyTimeout 600
```

笔记

如果看到以下错误，请考虑将代理超时值增加到 1000 秒：Proxy Error Proxy Error The proxy server received an invalid response from an upstream server. The proxy server could not handle the request POST/notebook/api/get_logs.

4) 点击保存更改。
5) 重新启动 Hue 服务。

7.8.18. 提交 Hive 查询后，无效的方法名称：“ GetLog”错误

`Invalid method name: 'GetLog' (code THRIFTAPPLICATION): None` 从 Hue 编辑器提交 Hive 查询之后，并且 Hue 尝试获取结果集时，可能会发生该错误。如果遇到此错误，则将 beeswax`use_get_log_api` 属性设置 为 `false` 使用 Cloudera Manager。

1) 以管理员身份登录 Cloudera Manager。
2) 转到集群> Hue 服务>配置。
3) 如下所示在 hue_safety_valve.ini 字段的“ Hue 服务高级配置代码片段（安全阀）”中添加/更新蜂蜡部分：

```
[beeswax]
use_get_log_api=false
```

4) 点击保存更改。
5) 重新启动 Hue 服务。

7.8.19. 在 Hue 中提交查询时出现“授权异常”错误

如果使用 Ranger 保护了集群，则必须从 Ranger Web UI 授予用户和组所需的权限。如果您的用户没有适当的权限，则他们可能无法从 Hue 编辑器访问某些数据库或表。

如果您的用户在从 Hue 编辑器提交查询时看到“授权例外：用户没有执行权限...”错误，然后使用 Ranger Web UI 授予他们适当的权限。

1) 以管理员身份登录 Cloudera Manager。
2) 转到“集群” >“ Ranger 服务” >“实例”选项卡，并记下与“ Ranger Usersync”角色类型相对应的主机名。
3) 通过单击“ Ranger Admin Web UI”打开 Ranger Web UI。
4) SSH 到您在步骤 2 中记下的 Ranger Usersync 主机，并如下添加用户或组：

```
ssh root@example.domain.site useradd [***USERNAME/GROUP-NAME***]
passwd [***PASSWORD***]
```

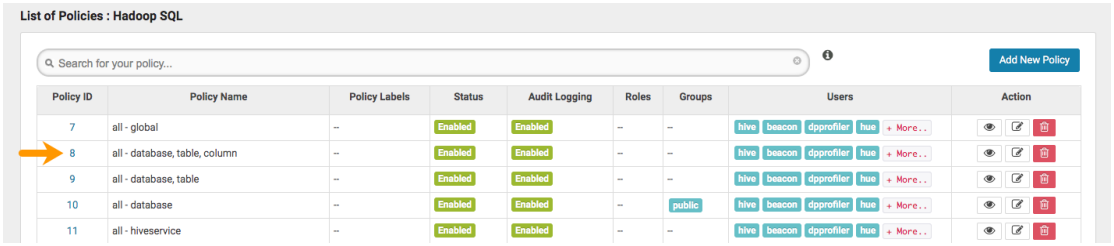
5) 在 Ranger Web UI 上，单击 HADOOP SQL 服务下列出的 Hadoop SQL。

进入“ Hadoop SQL 策略”页面。

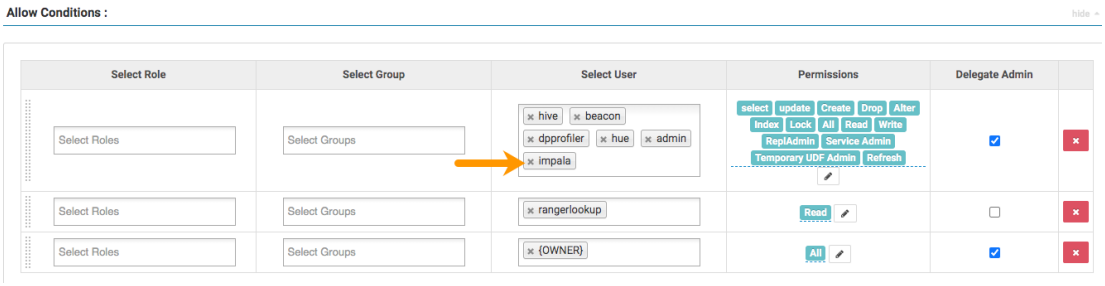
6) 在“ Hadoop SQL 策略”页面上，您可以通过添加新策略来授予新用户对所有数据库或特定数据库的访问权限。

- **授予对所有数据库的权限：**

  a) 单击与“`all - database, table, column`”相对应的策略 ID 。



在“编辑策略”页面上，在“允许条件”部分下的“选择用户”字段中添加要向其授予权限的用户，如下图所示：



要授予组权限，请在“选择组”字段中输入组名称。

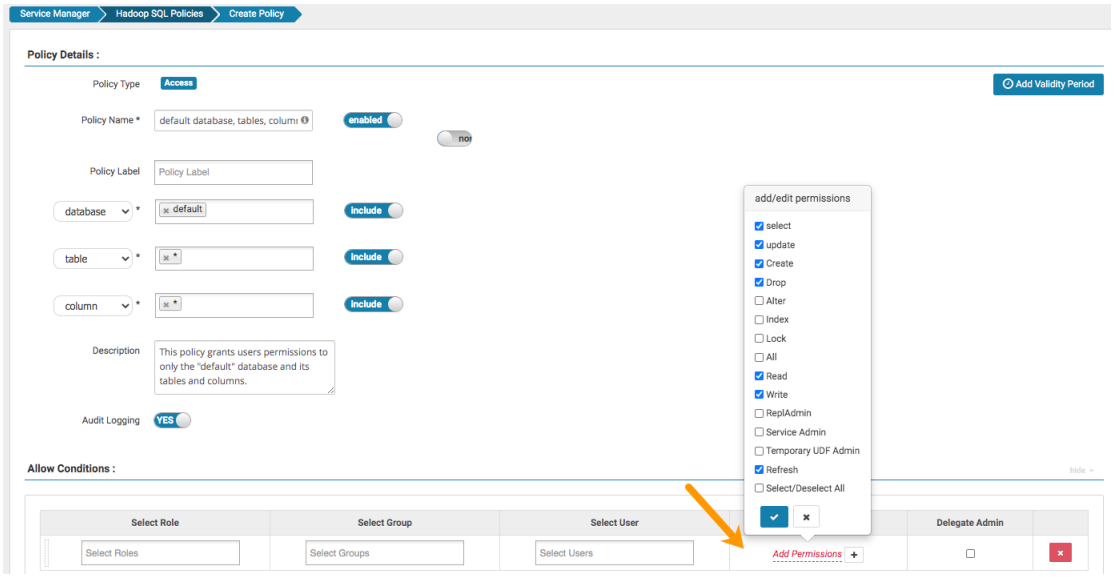
  b) 点击保存。

- **授予对特定数据库的权限：**

  c) 单击添加新策略。

系统显示“创建策略”页面。

  d) 在“策略详细信息”部分下，指定策略名称，然后选择您希望用户访问的数据库，表和列，如下图所示：



e) 在“允许条件”部分下，在“选择用户”字段中输入用户名，然后单击“ 添加权限”，然后选择用户必须具有的权限。

要授予组权限，请在“选择组”字段中输入组名称。

f) 点击添加。

7) 从 Cloudera Manager 启动 Hue 服务。

用户或组应该能够对策略中定义的任何实体运行任何查询。

7.8.20. 无法更改 Hue 中的压缩表

由于 Oracle 数据库（12 *c* 及更高版本）中的已知错误，您无法在压缩表上执行 ALTER TABLE 操作（添加，删除，删除，修改）。如果在 Hue 模式中具有压缩表，则可能会看到“ ORA-39726：压缩表上不受支持的添加/删除列操作”错误。

即使您解压缩现有表，也可能不允许您更改列。解决此问题的方法：

1) SSH 到安装了 Oracle 数据库的主机上。
2) 使用与压缩表相同的结构创建一个新的未压缩表。
3) 将数据从压缩表复制到新的未压缩表。
4) 重命名或删除压缩表。
5) 用原始压缩表的名称重命名未压缩的表。

现在，您应该能够在 Hue 表上执行 ALTER TABLE 操作（添加，删除，删除，修改）。

7.8.21. 从 Hue 访问“搜索”应用程序（Solr）时出现连接失败错误

如果您将 Solr 与 Hue 一起使用以生成交互式仪表板并为数据建立索引，并且您已在集群上部署了两个 Solr 服务并选择了第二个作为 Hue 的依赖项，则 Cloudera Manager 会分配第一个 Solr 服务的主机名，并且第二个 Solr 服务的端口号，导致在 hue.ini 文件的搜索部分中出现错误的 Solr URL。因此，当您尝试从 Hue Web UI 访问“搜索”应用程序时，可能会看到“连接失败”错误。

例如，考虑具有以下配置的两个 Solr 服务：

Solr-1，主机名= solr1，端口：2345

Solr-2，主机名= solr2，端口= 4567

如果选择 Solr-2 作为 Hue 的依赖服务，则 Cloudera Manager 会更新 hue.ini 文件的搜索部分，如下所示：

```
[search]
# URL of the Solr Server
 solr_url=http://solr2:2345/solr/
```

因此，您可能无法从 Hue Web UI 访问“搜索”应用程序。解决此问题的方法：

1) 以管理员身份登录 Cloudera Manager。
2) 转到集群> Hue 服务>配置，然后在 hue_safety_valve.ini 字段的“ Hue 服务高级配置代码段（安全阀）”中添加以下几行：

```
[search]
# URL of the Solr Server
solr_url=http://[***HOSTNAME***]:[***PORT***]/solr/
solr_url=http://solr2:4567/solr/
```

3) 点击保存更改。
4) 重新启动 Hue 服务。

7.8.22. 从顺化下载查询结果需要时间

如果从 Hue Web UI 下载查询结果很花时间，或者操作退出并显示“ Invalid query handle”消息，则可以通过增加 Hue Web 服务器使用的线程数来提高处理速度。

1) 以管理员身份登录到 Cloudera Manager。
2) 转到集群> Hue 服务>配置并搜索 `cherrypy_server_threads` 属性。

显示“ Hue Web 服务器线程”字段。

3) 将线程数增加到 100 或更高的值。

Hue Web 服务器的默认线程数为 50。

4) 点击保存更改。
5) 重新启动 Hue 服务。

7.8.23. 启用 TLS 后，Hue Load Balancer 无法启动

Hue 负载平衡器读取在 Hue 负载平衡器 TLS/SSL 服务器专用密钥文件（PEM 格式）配置属性中定义的专用密钥文件以启动。由于私钥文件通常是加密的，因此必须将 Hue Load Balancer 配置为使用相应的密钥密码，否则它将无法启动。

如果您在集群上为 Hue 服务启用了 TLS，并且私钥文件受到密码保护（加密），那么您可能会在 Hue Load Balancer 日志文件（`/var/log/hue-httpd/error_log`）：

```
AH02312: Fatal error initialising mod_ssl, exiting.
```

以下消息也记录在`/var/run/cloudera-scm-agent/process/[***XXX-HUE_LOAD_BALANCER***]/logs/stdout.log` 文件中：

```
CLOUDERA_HTTPD_USE_SSL=true
Apache/2.4.6 mod_ssl (Pass Phrase Dialog)
Some of your private key files are encrypted for security reasons.
In order to read them you have to provide the pass phrases.

Server example.test.com:443 (RSA)
Enter pass phrase:
```

解决此问题的方法：

1) 在所选的安全目录中创建一个密码文件，并插入私钥密码，如以下示例所示：

```
echo "abc123" >/etc/security/password.txt
```

其中 abc123，私钥密码 password.txt 是，密码文件是。

2) 设置文件所有权和权限，如以下示例所示：

```
chown hue:hue password.txt
chmod 700 password.txt
```

3) 在“Hue 负载平衡器 TLS/SSL 服务器 SSLPassPhraseDialog”字段中，输入包含密码短语的文件路径，该密码短语用于加密 Hue 负载平衡器服务器的私钥。

在这种情况下，/etc/security/password.txt。

4) 点击保存更改。

5) 重新启动 Hue 服务。

7.8.24. 无法终止以 Kerberized 集群运行的 Hue 作业浏览器中的 Hive 查询

在以 Kerberized 的集群上，如果 YARN 尚未为 HTTP Web 控制台启用 Kerberos 身份验证，则您可能无法从 Hue Job Browser 终止 Hive 查询，并且您可能会在 Hue 角色日志 runcpserver.log 文件中看到以下错误：默认静态用户无法执行此操作。（错误 403）”。

在使用 Kerberized 的集群上，YARN 必须为 HTTP Web 控制台启用 Kerberos 身份验证。如果未启用身份验证，则试图使用 REST API 访问 YARN 的用户或应用程序将被标识为默认的“ dr.who”用户。默认用户无权访问 YARN UI 和终止正在运行的作业。作为立即解决方案，您可以使用以下命令从 Hue 查询编辑器或 YARN CLI 终止作业：

```
yarn application -kill [***APPLICATION-ID***]
```

要从 Hue Job Browser 启用终止作业和运行查询，请为 YARN 的 HTTP Web 控制台启用 Kerberos 身份验证，如下所示：

1) 以管理员身份登录到 Cloudera Manager。

2) 转到“集群” >“ YARN” >“配置”，然后 enable kerberos 在搜索框中键入。

3) 选择“为 HTTP Web 控制台启用 Kerberos 身份验证”。

4) 点击保存更改。

5) 重新启动 YARN 服务。

7.8.25. 无法在受 Knox 保护的集群上的 Hue 中查看或创建 Oozie 工作流

如果您无法查看 Hoz 的工作流操作，例如在由 Knox 保护的集群上的 Hue 的 Oozie 编辑器页面上的“ HiveServer2 脚本”或“ Shell 脚本”，请检查 Hue 日志中是否存在以下警告：`"POST/desktop/log_js_error HTTP/1.1" --- Referer checking failed - https://[***FQDN***]:[***PORT***]/oozie//editor/workflow/new does not match any trusted origins.`。要解决此问题，请将 Oozie 服务器 URL 添加到 `trusted_origins`Hue Advanced Configuration Snippet 中的 属性中。

在 CDP 集群上设置 Knox 时，Knox 会验证来自其他服务和应用程序的访问或请求。如果在 `trusted_origins` 属性中指定 Oozie 服务器 URL ，Knox 可以检查传入请求是否来自受信任的源（Oozie）并批准访问，从而允许您从 Hue 查看和创建 Oozie 工作流程。

1) 以管理员身份登录到 Cloudera Manager。
2) 转到 hue_safety_valve.ini 的集群> Hue 服务>配置> Hue 服务高级配置代码片段（安全阀）。
3) `trusted_origins` 在桌面部分下的属性中添加 Oozie 服务器 URL，如下所示：

```
[desktop]
[[session]]
# Comma-separated list of Oozie nodes and Oozie ports
# for each Oozie instance
trusted_origins=[***OOZIE-NODE1***]:[***OOZIE-PORT1***], [***OOZIE-NODE2***]:[***OOZIE-PORT2***],...
```

例如：

```
[desktop]
[[session]]
# Comma-separated list of Oozie nodes and Oozie ports
# for each Oozie instance
trusted_origins=localhost:11000
```

4) 点击保存更改。
5) 重新启动 Hue 服务。

7.8.26. 1040，“连接太多”异常

如果 Hue 显示“ 1040，连接太多”异常，则可能是 Hue 后端数据库超载并且超出了最大可用连接数。若要解决此问题，您可以增加 `max_connections` 数据库的属性的值。

当 MySQL 数据库用尽最大可用连接数时，会发生 1040“连接过多”异常。如果使用的是 Impala 引擎，则在 Hue Web 界面上可能会看到以下错误消息：`OperationalError at /desktop/api2/context/computes/impala("1040: too many connections")`。对于 Hive，可能会显示类似的错误。在 Hue 服务器日志中也会捕获该异常。

该 `max_connections` 属性定义一个 MySQL 实例可以接受的最大连接数。连接数不受控制可能会使服务器崩溃。以下是调整 `max_connections` 属性值的一些准则：

- `max_connections` 根据集群的大小设置属性的值。
- 如果主机少于 50 个，则可以在同一主机上存储多个数据库（例如，活动监视器和服务监视器）。如果主机超过 50 个，则对每个数据库/主机对使用单独的主机。主机不必专门为数据库保留，但是每个数据库必须位于单独的主机上。
- 对于少于 50 个主机：
  - 将每个数据库放在其自己的存储卷上。
  - 每个数据库允许 100 个最大连接，然后添加 50 个额外的连接。例如，对于两个数据库，将最大连接数设置为 250。如果在一个主机上存储五个数据库（Cloudera Manager Server，Activity Monitor，Reports Manager，Atlas 和 Hive MetaStore 的数据库），则将最大连接数设置为 550。

若要增加最大可用连接数并解决“ 1040，连接太多”异常：

1) 登录 Cloudera Manager 并停止 Hue 服务。
2) 以 root 用户身份登录到数据库实例。
3) 通过运行以下命令来检查可用连接数：

```
grep max_conn /etc/my.cnf
```

`/etc/my.cnf` 是选项文件（`my.cnf`）的默认位置。

4) max_connections 根据上面提供的准则，从 MySQL Shell 中设置属性的新

值。例如：

```
mysql> SET GLOBAL max_connections = 550;
```

5) 重新启动 Hue 服务。

8.参考资料

CDP 组件用户手册：https://docs.cloudera.com/cdp-private-cloud-base/latest/index.html

Cloudera-Manager 操作手册: https://docs.cloudera.com/cdp-private-cloud-base/latest/concepts-cloudera-manager.html

Cloudera Manager 发布说明: https://docs.cloudera.com/cdp-private-cloud-base/latest/concepts-cloudera-manager.html

Cloudera Runtime 发布说明：https://docs.cloudera.com/cdp-private-cloud-base/latest/runtime-release-notes/topics/rt-pvc-runtime-overview.html