GBASE

GBase 8a MPP Cluster (集群)
管理员手册 V8.6.2.X

www.gbase.cn

GBase 8a MPP Cluster 管理员手册，南大通用数据技术股份有限公司

通讯方式

南大通用数据技术股份有限公司

天津华苑产业区海泰发展六道 6 号海泰绿色产业基地 J 座(300384)

电话：400-013-9696　　邮箱：info@gbase.cn

目 录

前言 1
手册简介 1
公约 2
1 系统简介 3
1.1 系统简介 3
1.2 产品技术特点 3
1.3 产品功能简介 4
1.4 名词解释 5
2 集群规划 8
2.1 数据模型规划 8
2.2 空间规划 10
2.3 C3 辅助工具 11
2.3.1 C3 简介 11
2.3.2 C3 工具常用命令 11
3 基本管理 16
3.1 系统安装 16
3.2 修改数据库 gbase 用户密码 16
3.3 集群服务的启停 17
3.4 集群的登录与退出 18
3.5 集群基本配置 19
3.5.1 GCWare 配置信息 19
3.5.2 GCluster 基本配置 19
3.5.3 GNode 基本配置 19
4 gcadmin 20
4.1 distribution 命令 21
4.2 rmdistribution 命令 31
4.3 addnodes 命令 32
4.4 rmnodes 命令 34
4.5 showdistribution 命令 35
4.6 switchmode 命令 39
4.7 showlock 命令 39
4.8 showddlevent 命令 42
4.9 showdmlevent 命令 43
4.10 showdmlstorageevent 命令 45

4.11 showcluster 命令 46
4.12 getdistribution 命令 49
4.13 setnodestate 命令 51
4.14 showfailover 命令 51
4.15 showfailoverdetail 命令 53
4.16 help 命令 55
4.17 version 命令 55
4.18 upgrade 命令 55
4.19 replacenodes 命令 56
5 数据加载 58
5.1 功能简介 58
5.2 数据服务器配置 58
5.2.1 FTP 服务器配置 59
5.2.2 HTTP 服务器配置 64
5.2.3 HDFS 服务器配置 68
5.2.4 SFTP 服务器配置 74
5.2.5 KAFKA 服务器配置 76
5.3 加载状态监控 79
5.4 加载日志汇总与查询 81
5.4.1 加载错误数据与溯源信息检索 82
5.4.2 加载结果信息统计日志 84
5.4.3 加载结果信息内存表查询 85
5.4.4 加载跳过文件列表日志回传 87
5.4.5 加载错误数据、溯源信息和日志文件导出 87
5.5 加载兼容工具使用说明 88
5.5.1 部署方式 89
5.5.2 使用方式 90
5.5.3 错误数据收集功能 93
5.5.4 兼容工具 V8.5.1.2 控制文件与 V8.6.2.X SQL 关键字的映射表 94
5.5.5 兼容工具的兼容参数表 94
6 集群的扩容、缩容 97
6.1 集群扩容和缩容操作流程 97
6.1.1 扩容的操作流程 97
6.2 安装和卸载 data 节点 98
6.3 创建 distribution 和 hashmap 98
6.4 删除 hashmap 和 distribution 99

6.5 rebalance 命令 100
6.5.1 缩容的操作流程 101
6.5.2 rebalance table 102
6.5.3 continue rebalance table 104
6.5.4 cancel rebalance table 105
6.5.5 rebalance database 106
6.5.6 pause rebalance database 107
6.5.7 continue rebalance database 108
6.5.8 cancel rebalance database 109
6.5.9 rebalance instance 110
6.5.10 pause rebalance instance 111
6.5.11 continue rebalance instance 112
6.5.12 cancel rebalance instance 113
6.5.13 调整 rebalance 任务优先级 114
6.6 rebalance 命令相关参数 116
7 集群节点替换 118
7.1 简介 118
7.2 功能概述 118
7.3 注意事项 119
7.4 节点替换的步骤 120
7.4.1 步骤一：检查网络 120
7.4.2 步骤二：gcadmin 设置要替换节点的状态 120
7.4.3 步骤三：准备用于节点替换的新机器 120
7.4.4 步骤四：gcadmin 查看集群各项状态 121
7.4.5 步骤五：replace.py 对节点进行替换安装 121
8 客户端使用 SSL 加密连接到集群 124
8.1 生成 ssl 连接证书 124
8.2 server 配置 127
8.3 client 配置 129
9 集群的审计日志 131
9.1 简介 131
9.2 配置参数 132
9.3 审计策略 133
9.3.1 创建审计策略 133
9.3.2 修改审计策略 134
9.3.3 删除审计策略 135
9.4 存储方式 135

9.5 使用约束 135
9.6 使用示例 135
9.7 审计日志高可用 139
10 EXPLAIN 显示查询计划 141
10.1 缺省输出 141
10.2 EXTENDED 输出 150
10.2.1 查询计划部分 157
10.2.2 执行计划部分 160
10.3 PARTITIONS 输出 160
11 集群的权限管理 168
11.1 用户管理 168
11.2 权限管理 169
11.3 用户组管理 173
12 安全管理 175
12.1 密码安全管理 175
12.1.1 密码强度控制 175
12.1.2 密码重用控制 176
12.1.3 密码有效期控制 176
12.2 用户安全管理 178
12.2.1 登录重试锁定 178
12.2.2 账户锁定和解锁 178
12.2.3 账户限定 host 列表 179
12.3 查看安全信息 179
12.4 登录信息显示 180
12.5 安全管理的权限 181
12.6 登陆管理一致性 181
13 资源管理 183
13.1 概念说明及管理接口 183
13.1.1 Consumer Group 183
13.1.2 Resource Pool 185
13.1.3 Resource Plan 193
13.1.4 Resource Directive 194
13.2 信息查询 197
13.2.1 要素定义信息查询 197
13.2.2 运行期信息查询 200
13.2.3 统计信息查询 202
13.3 应用场景 204

13.3.1 加载和查询混合场景 204
13.3.2 白天跑查询，晚上跑批场景 207
13.3.3 高低写限速组场景 207
13.3.4 全天加工和查询任务并行场景 207
13.3.5 支持高级用户抽查场景 210
13.3.6 多租户隔离场景 213
13.4 磁盘 IO 控制 216
13.4.1 DC 同步 I/O 控制 216
13.4.2 磁盘 I/O 读写速率上限控制 216
13.5 集群层显示 SQL 在各节点资源使用命令 216
13.5.1 查看 SQL 在单机层执行使用资源信息 217
13.5.2 查看 SQL 在集群层执行信息 218
13.5.3 查看 SQL 在集群执行信息 220
13.6 注意事项 220
13.6.1 系统 cgconfig 服务 221
13.6.2 受控 SQL 221
13.6.3 高、低优先级用户使用约束 222
14 支持 Kerberos 安全认证 223
14.1 Kerberos 客户端安装与配置 223
14.2 集群扩容影响 224
14.3 集群升级影响 224
14.4 集群节点替换工具 224
14.5 加载/导出 HDFS 文件影响 225
15 密钥证书管理 227
15.1 证书存储位置 227
15.2 语法描述 227
15.3 创建证书 227
15.3.1 打开/关闭证书 228
15.3.2 显示证书状态 228
15.3.3 修改证书口令 229
15.3.4 明文、密文密钥转换 229
15.4 集群环境配置 229
16 元数据管理 230
16.1.1 限制表实例数量 230
16.1.2 限制表实例元数据总大小 230
16.1.3 超出限制后对 m_tables 清理比例 230
16.1.4 状态监测 231

17 图形化管理工具 232
17.1 企业管理器 232
17.2 集群监控工具 233

前言

手册简介

GBase 8a MPP Cluster 管理员手册主要从用户使用的角度，介绍了 GBase 8a MPP Cluster 数据库日常维护管理的知识。通过阅读本手册，用户可以学习到 GBase 8a MPP Cluster 数据库的常用管理技能和维护技巧，掌握对数据库常用对象的命令管理和图形化管理技能，掌握对数据库状态监控的技能。

第一章对 GBase 8a MPP Cluster 做简单介绍，主要描述了 GBase 8a MPP Cluster 的功能，特点等方面的内容。

第二章介绍了 GBase 8a MPP Cluster 的规划，主要描述了 GBase 8a MPP Cluster 的数据模型规划、空间规划的内容。

第三章介绍了 GBase 8a MPP Cluster 的基本管理，主要描述了集群服务的启停，集群的登录与退出和集群基本配置等内容。

第四章介绍了 GBase 8a MPP Cluster 的 gcadmin 命令，主要描述了如何使用 gcadmin 对集群进行管理。

第五章介绍了 GBase 8a MPP Cluster 数据加载功能。

第六章介绍了 GBase 8a MPP Cluster 中的审计日志，主要描述了如何运用审计日志来监视用户所执行的数据库操作的内容。

第七章介绍了 GBase 8a MPP Cluster 中的扩容、缩容功能，主要描述了如何对集群进行扩容操作和缩容操作的内容。

第八章介绍了 GBase 8a MPP Cluster 节点替换功能，主要描述如何对集群节点进行替换。

第九章介绍了 GBase 8a MPP Cluster 的权限管理，主要描述了用户管理以及用户权限管理方面的内容。

第十章对 GBase 8a MPP Cluster 的图形化管理工具进行了简单介绍。

公约

下面的文本约定用于本文档：

约　定	说　明
加粗字体	表示文档标题
大写英文（SELECT）	表示 GBase 8a MPP Cluster 关键字
等宽字体	表示代码示例
…	表示被省略的内容。

1 系统简介

1.1系统简介

南大通用大规模分布式并行数据库集群系统，简称：GBase 8a MPP Cluster，它是在 GBase 8a 列存储数据库基础上开发的一款 Share Nothing 架构的分布式并行数据库集群，具备高性能、高可用、高扩展特性，可以为超大规模数据管理提供高性价比的通用计算平台，并广泛地用于支撑各类数据仓库系统、BI 系统和决策支持系统。

1.2产品技术特点

GBase 8a MPP Cluster 具备以下技术特征：

1) 低硬件成本：完全使用 x86 架构的 PC Server，不需要昂贵的 Unix 服务器和磁盘阵列；

2) 集群架构与部署：完全并行的 MPP + Share Nothing 的分布式架构，采用多活 Coordinator 节点、对等数据节点的两级部署结构。Coordinator 节点支持最多部署 64 个，数据节点支持最多部署 300 个，数据量支持 15PB。

3) 海量数据分布压缩存储：可处理 PB 级别以上的结构化数据，采用 hash 或 random 分布策略进行数据分布式存储。同时采用先进的压缩算法，减少存储数据所需的空间，可以将所用空间减少 1～20 倍，并相应地提高了 I/O 性能；

4) 数据加载高效性：基于策略的数据加载模式，集群整体加载速度随节点数增加线性增长；

5) 高扩展、高可靠：支持集群节点的在线扩容和缩容，效率更高，对业务的影响更小。

6) 高可用、易维护：数据通过最多 2 个副本提供冗余保护，自动故障探测和管理，自动同步元数据和业务数据。提供图形化监控工具和企业管理器等管理工具，简化管理员对数据库的管理工作；

7) 高并发：读写没有互斥，支持简化模式的 MVCC，支持数据的边加载边查询，单个节点并发能力大于 300 用户；

8) 行列转换存储：提供行列转换存储方案，从而提高了列存数据库特殊查询场景的查询响应耗时；

9) 标准化：支持 SQL92 标准，支持 ODBC、JDBC、ADO.NET 等国际接口规范。

10) 数据节点多分片：在一个数据节点上可同时部署多个数据分片；单数据节点数据分片数量支持最多 32 个。建议单个节点主分片数量不超过 10 个。

11) 灵活的数据分布：用户可以按照业务场景的需求，自定义数据分布策略，从而在性能、可靠性和灵活性间获得最佳匹配。

12) 异步消息：Coornator 默认采用异步消息模式与数据节点通信，支持高达 300 节点的集群规模。

1.3产品功能简介

GBase 8a MPP Cluster 支持的功能如下表所示：

功　能	描　述
结构化查询语言	符合 SQL 92 标准，支持 CREATE、ALTER、DROP 等 DDL 语法，支持 SELECT、INSERT、UPDATE、DELETE 等 DML 语法，支持单表，多表联合查询
数据类型	INT、TINYINT、SMALLINT、BIGINT、DECIMAL、FLOAT、DOUBLE 数值数据类型 CHAR、VARCHAR、TEXT 字符数据类型 DATE、TIME、DATETIME、TIMESTAMP 日期类型

功　能	描　述
	BLOB 二进制数据类型
数据库对象	提供了数据库，表，索引，视图，存储过程，自定义函数等常用数据库对象的创建，修改和删除操作，支持数据库用户的创建，删除操作，以及用户权限的分配与回收
行列混合存储	基于创建的物理表，可以实现行存列的创建，修改和删除
图形化工具	提供了企业管理工具和集群监控工具。
接口	符合并支持 ODBC、JDBC、ADO.NET 等接口规范，并提供本地 CAPI 接口。

1.4名词解释

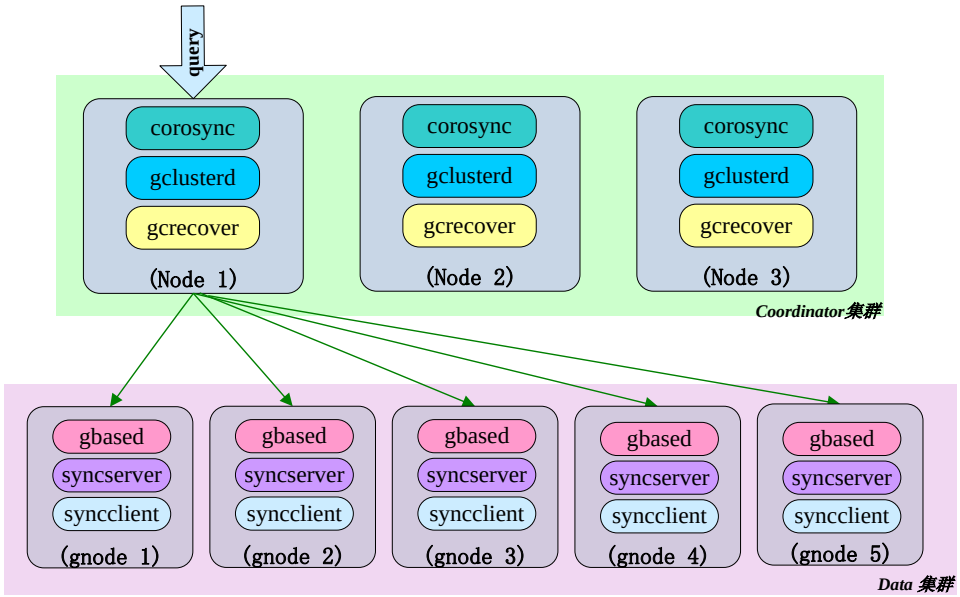


图 1-1 GBase 8a MPP Cluster 拓扑结构图

GCluster:

负责 sql 的解析、sql 优化、分布式执行计划生成、执行调度。

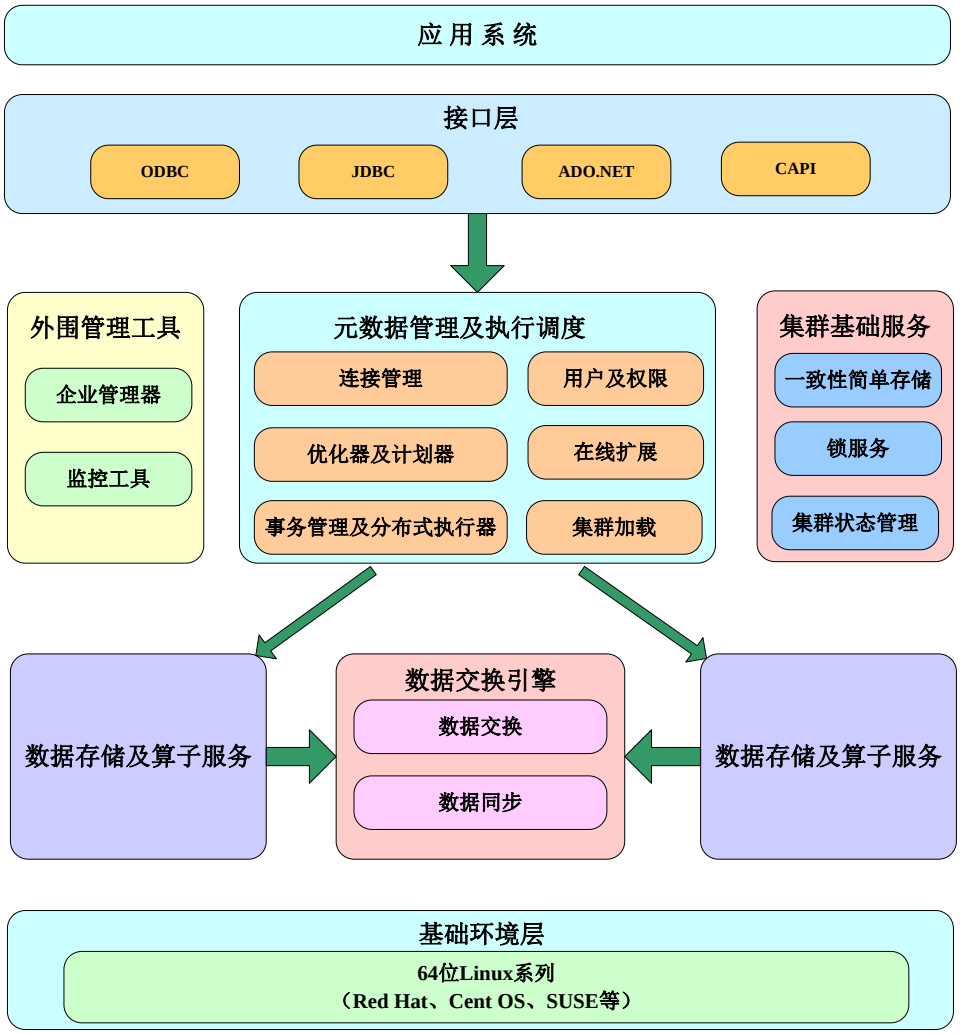


图 1-2 GCluster 架构图

GCWare：

GCWare 存在于 coordinator 集群节点上，用于各节点 GCluster 实例间共享信息(包括集群结构，节点状态，节点资源状态等信息)，以及控制多副本数据操作时，提供可操作节点，并在多副本操作中，控制各节点数据一致性状态。

GNode：

GNode 是 GCluster 中最基本的存储和计算单元。GNode 是由 GCWare 管理的

一个 8a 实例，为 data 集群中的节点。GNode 负责集群数据在节点上的实际存储，并从 GCluster 接收和执行经分解的 SQL 执行计划，执行结果返回给 GCluster。数据加载时，GNode 直接接收数据，写入本地存储空间。

Distribution:

决定数据在 data 集群上的部署方式，包括数据分片数、主分片和备份分片的部署方式。一种部署方式被称为一个 distribution，在线扩容、缩容操作会产生新的 distribution，最多支持两个 distribution 同时存在。存在于不同的 distribution 中表可以正常的关联查询。

互备节点:

通过对集群中节点上的数据保存多个副本的方式，来实现数据库集群的高可用性，保存相同副本的数据节点构成的具有 HA 能力的互备节点。一个表的不同分片的互备节点可以不同。

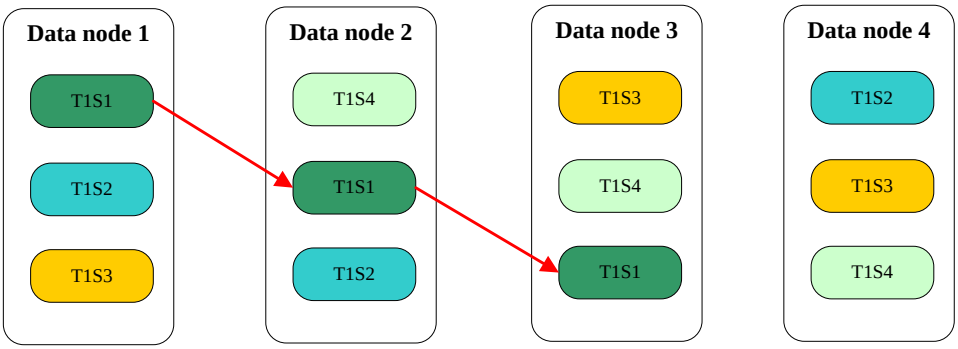


图 1-3 互备节点高可用性管理示意图 2

上图中 node1 节点出现故障后，执行器会指向备份节点 node2 或 node3 的分片。

2 集群规划

在集群上需要部署海量数据，如果集群部署失败后，再重新部署，成本将很高。因此，在实际部署集群前，需要对将在集群上部署应用的数据模型(schema)，数据容量，数据更新率，数据生命期，数据安全策略，查询性能等诸多方面进行分析和优化。

2.1数据模型规划

GBase 8a MPP Cluster 中，数据是按表存放的，因此数据模型规划阶段需要对数据表的分布策略进行仔细规划。GBase 8a MPP Cluster 中的表可以分为如下类别:

- 复制表：在集群中复制表所在的 Distribution 中的每个节点都保存一份全量数据，再与其它表进行关联查询时可以直接在本节点上完成，无需与其它节点进行交互，因此性能最优。但由于各个节点上数据完全相同，导致存储空间增加，因此通常用于小表、维度表或经常需要 JOIN 关联的数据表。
- 分布表：将数据分布存储到不同的节点上。每个节点上存储一部分数据（分片存储），根据不同应用场景，GBase 8a MPP Cluster 提供如下两种分布表:
  - 哈希分布表：将表中某列指定为哈希列，然后将数据按照哈希算法的取值存储到不同的节点上。每个节点上只存储一部分数据。这种存储策略，将大表数据进行分拆，实现分布式存储。哈希分布表，经常用于对哈希列进行等值查询的场景，是大型数据中心最常用的数据分布方式。
  - 随机分布表：将数据随机存储到不同的节点上，每个节点只存储一部分数据，各个节点上的数据量接近。这种数据分布方式，通常用于对数据进行汇总计算的场景。

例如,在星型模型中，一般将维度表设为复制表，事实表设为分布表(哈希分布或者随机分布)。在下图 ssb 模型中，lineorder 表设为分布表，其他维度表设为复制表。

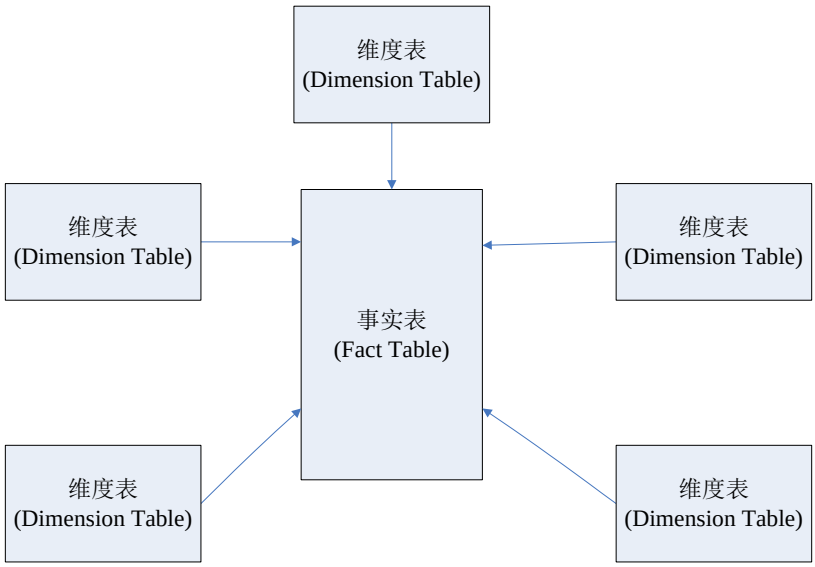


图 2-1 ssb 星型模型结构图

对于分布表，还需要考虑数据的高可用，即需要确定备份个数及备份策略，最多支持备份个数为两个，满足不同的高可用需求。

- nocopies 表：nocopies 表的引入是为了提供一种作为中间表使用的表，用以消除副本存储带来的开销，并且 nocopies 表也可以转换为分布表。
- 临时表：当创建表时，使用关键词 TEMPORARY。临时表被限制在当前连接中，当连接关闭时，临时表会自动地删除，这样两个不同的连接可以使用同一个临时表名而不会发生冲突，也不会与同名现有的表冲突（现有表将被隐藏，直到临时表被删除）。

  可以联合使用 NOCOPIES 关键字来创建临时的 nocopies 表；

  可以联合使用 REPLICATED 关键字来创建临时复制表；

  可以联合使用 DISTRIBUTED BY 关键字来创建临时哈希分布表；

2.2空间规划

确定了应用的 schema 模型和高可用需求后，要根据物理节点存储空间的大小，计算每个节点的有效存储空间，为确定集群中所需节点数量提供重要的参考数据。

节点空间的用途主要分为两部分，数据存储空间、运算存储空间。

数据存储空间主要包括：

- 复制表占用空间。
- 分布表占用空间。
- 分布表的副本占用的空间。
- 索引占用的空间（如果有索引）。

运算存储空间主要包括：

- 运算存储空间主要是运算过程中需要的临时空间。需要根据业务实际场景来分析，通常运算存储空间至少是数据存储空间的 20% ~ 30%。

另外，GBase 8a MPP Cluster 支持对数据进行压缩存储，并且提供多种压缩模式。可以实现深度压缩，最大程度节省磁盘空间；也可以实现轻度压缩，最大程度提升性能。用户需根据实际的业务数据特征、业务需求来选择不同的压缩方式，并根据实际数据测试结果来估算压缩后的空间需求。

数据管理策略主要指在时间方向上，随着时间推移，伴随数据生命期在 GCluster 上的管理策略。其主要内容包括：

- 集群扩容，扩充集群的数据存储容量，支持在线扩容。
- 复制表数据加载采用复制分布模式，即所有节点数据内容都一致。
- 分布表数据加载策略有随机分布模式和哈希模式。

随机分布模式：

随机分布模式是指数据库创建随机分布的分布表，在对其进行加载时按随

机模式分发数据内容。

哈希模式:

哈希模式是指在加载之前先对原始数据中的每条数据中指定的哈希列进行处理，处理后的数据按照哈希值装入特定的哈希桶中，每个哈希桶对应一个集群节点。这样每个节点所得到的数据就都具有了某种共同特征（指定列都具有相同的哈希值），在查询时优化引擎可以根据这些共同特征对查询计划进行优化，以达到缩短查询时间的目的。

2.3C3 辅助工具

2.3.1C3 简介

C3 工具的安装过程请参见《GBase 8a MPP Cluster 安装手册(Linux RHEL6)》中的相关章节。

2.3.2C3 工具常用命令

2.3.2.1 cexec 和 cexecs 命令

功能: cexec 以并行方式调用（cexecs 以串行方式调用）Linux 和集群管理的命令工具，可以在所有节点或指定节点上运行 Linux 和集群管理的命令。

语法: cexec | cexecs [clustername:indexvalue] command_name

clustername: 是 C3 配置文件中的集群名称;

indexvalue: 是集群中每台节点机器序号，从 01 开始，如果是连续的多台机器，可以使用 - 来标注划分起始节点序号和终止节点序号，例如 01-03，如果是不连续的多个集群节点，可以使用“,”来分隔各序号，例如: 01,03,05;

上面的都是可选项。可以省略，此时表示全部集群节点的信息。

command_name：为在操作系统中，当前用户可以执行的 Linux 和集群命令，使用单引号括起命令。

示例 1：显示 Linux 机器上的日期

```
# cexec 'date'

************************ test ************************
--------- 192.168.10.35---------
2012 年 12 月 18 日 星期二 22:35:45 CST
--------- 192.168.10.36---------
2012 年 12 月 18 日 星期二 22:35:41 CST
--------- 192.168.10.37---------
2012 年 12 月 18 日 星期二 22:35:42 CST
```

示例 2：显示指定集群节点机器上日期

```
# cexec 'test:01 date'

************************ test ************************
--------- 192.168.10.35---------
2012 年 12 月 18 日 星期二 22:39:19 CST
```

示例 3：查看 gcware 服务的状态。

```
# cexec 'service gcware status'

************************ test  ************************
--------- 192.168.10.35---------
corosync (pid 1190) is running...
--------- 192.168.10.36---------
corosync (pid 1210) is running...
--------- 192.168.10.37---------
corosync (pid 1195) is running...
```

示例 4：在指定节点上执行停止 gcware 服务的操作

```
# cexec 'test:01 service gcware stop'
************************ test ************************
--------- 192.168.151.205---------
Stopping GCMonit success!
Signaling GCRECOVER (gcrecover) to terminate: [  OK  ]
Waiting for gcrecover services to unload:.[  OK  ]
Signaling GCSYNC (gc_sync_server) to terminate: [  OK  ]
Waiting for gc_sync_server services to unload:[  OK  ]
Signaling GCLUSTERD  to terminate: [  OK  ]
Waiting for gclusterd services to unload:...[  OK  ]
Signaling GBASED  to terminate: [  OK  ]
Waiting for gbased services to unload:.[  OK  ]
Signaling GCWARE (gcware) to terminate: [  OK  ]
Waiting for gcware services to unload:[  OK  ]
```

注意事项：

1、 必须首先安装好 GBase 8a MPP Cluster 产品及配置好 C3 工具后，才能使用 C3 工具。

2、 使用 cexec | cexecs 执行 service gcware ……和 gcluster.server ……这两个集群管理命令时，需要在 root 用户下执行。

2.3.2.2 cpush 命令

语法：cpush [clustername:indexvalue] source target

clustername：是 C3 配置文件中的集群名称；

indexvalue：是集群中每台节点机器序号，从 01 开始，如果是连续的多台机器，可以使用 - 来标注划分起始节点序号和终止节点序号，例如 01-03，如果是不连续的多个集群节点，可以使用“,”来分隔各序号，例如：01,03,05；

source：源文件路径及文件名；

target：分布后的目标路径。

功能：可以复制集群中的文件和目录到所有节点或指定节点上，最常用的场景就是，在集群装有 C3 工具的节点机器上编辑一个文件后，需要将此编辑后文件拷贝粘贴或覆盖到集群其他节点上。

示例 1：

在装有 C3 工具的节点机器的/home/gbase/下有一个 new.txt 文件，而集群另一个节点上并无此文件。

在 192.168.10.35 机器上查看 new.txt 文件：

```
$ ls new.txt
new.txt

$ cat new.txt
1,Mike
2,Rose
3,Jane
4,John
5,Tom
6,Jim
7,Jerry
```

在 192.168.10.36 机器上查看 new.txt 文件，报告无此文件。

```
$ ls new.txt
ls: 无法访问 new.txt: 没有那个文件或目录。
```

在 192.168.10.35 上使用 cpush 命令分布 new.txt 文件。

```
$ cpush /home/gbase/new.txt  /home/gbase/
```

在 192.168.10.36 上查看分布后的文件。

```
$ ls new.txt
```

```
new.txt

$ cat new.txt
1,Mike
2,Rose
3,Jane
4,John
5,Tom
6,Jim
7,Jerry
```

3 基本管理

3.1系统安装

这部分内容请参见《GBase 8a MPP Cluster 安装手册》。

3.2修改数据库 gbase 用户密码

用户安装完 GBase 8a MPP Cluster 后，会自动创建一个数据库用户 gbase，其初始密码为 gbase20110531，这个密码用户可以修改。

GBase 8a MPP Cluster 支持修改数据库 gbase 用户的密码，如果数据库 gbase 用户的密码已修改，则在使用以下工具时，需要用户输入密码，进行身份验证后，才能执行：

- gcadmin

登录 GBase 8a MPP Cluster 后，可以修改 gbase 用户密码，语法如下：

SET PASSWORD [FOR *user_name*] = PASSWORD(password_newvalue);

user_name：被修改数据库登录密码的用户名称，如果省略此参数，则修改的就是当前登录集群用户的密码。

password_newvalue：新密码，使用一对单引号“’”括起来。

示例 1：使用 root 用户修改 gbase 用户的密码。

```
$ gccli -uroot -p
    Enter password:

    GBase client 8.5.1.2 build 27563. Copyright (c) 2004-2013, GBase.  All Rights Reserved.

gbase> SET PASSWORD FOR gbase = PASSWORD('123456');
    Query OK, 0 rows affected
```

```
gbase> \q;
```

示例 2：使用 gbase 用户修改密码。

```
$ gccli -ugbase -p
    Enter password:

    GBase client 8.5.1.2 build 27563. Copyright (c) 2004-2013, GBase.  All Rights
Reserved.

gbase> SET PASSWORD = PASSWORD('1111111');
    Query OK, 0 rows affected

gbase> \q;
```

3.3集群服务的启停

当 GBase 8a MPP Cluster 安装完毕后，其中的 gcware 服务就会自动运行，并且，每当开机和重新启动机器时，gcware 服务也会自动运行，而不必手动启动。如果用户在使用中，需要手工进行集群服务的启停操作，则需要在每个节点机器中，使用 Linux 下的 root 用户进行操作。具体命令如下：

- 启动命令

```
# service gcware start
```

- 停止命令

```
# service gcware stop
```

- 重新启动命令

```
# service gcware restart
```

- 状态查看

```
# service gcware status
```

3.4集群的登录与退出

正常情况下，在集群安装完毕后，系统会在 Linux 中创建一个 gbase 用户，登录集群时，用户需要切换到 gbase 用户，操作如下：

```
# su - gbase
$
```

切换完毕后，可以登录集群。

默认情况下，在集群安装完毕后，系统会创建一个默认的数据库超级帐号 root，其初始密码默认为空。

首次登录 GBase 8a MPP Cluster 后，管理员必须为 root 帐号设置一个安全密码。

示例：安装 GBase 8a MPP Cluster 后，用户首次登录集群，需要修改 root 用户的密码，退出登录，重新使用 root 用户及修改后的密码，进行登录。

```
$  gccli  -uroot

    GBase client 8.5.1.2 build 26068. Copyright (c) 2004-2013, GBase.  All Rights
Reserved.

gbase> SET PASSWORD FOR root = PASSWORD('H133%_h');
    Query OK, 0 rows affected
```

退出登录的命令为在 gbase>提示符下，键入“\q”。

```
gbase> \q
    Bye
```

修改 root 的口令后，重新登录集群。

```
$  gccli -uroot -p
    Enter password:
    GBase client 8.5.1.2 build 26068. Copyright (c) 2004-2013, GBase.  All Rights
Reserved.
```

gbase >

更详细的介绍，参见《GBase 8a MPP Cluster 安装手册(Linux RHEL6)》。

3.5集群基本配置

3.5.1GCWare 配置信息

GCWare 主要负责管理三部分集群信息：集群静态结构、集群节点及数据状态、集群锁信息。

配置文件保存在/etc/corosync/corosync.conf 中

3.5.2GCluster 基本配置

在/opt/gcluster/config/gbase_8a_gcluster.cnf 配置文件中，可以查看集群的基本配置。

3.5.3GNode 基本配置

在/opt/gnode/config/gbase_8a_gbase.cnf 配置文件中，可以查看 GNode 的基本配置。

4 gcadmin

gcadmin 为管理员提供管理集群的操作，包括生成、删除 distribution，查看集群状态、distribution 信息等常用功能。

gcadmin 命令在 DBA 用户（即安装时指定的 dbaUser）下进行操作，支持的选项如下：

```
# gcadmin --help

Usage: gcadmin <command> [arg1[, arg2...]]

1.  gcadmin distribution <gcChangeInfo.xml> <p num> [d num] [pattern 1|2]
[db_pwd password]:                                  generate distribution
2.  gcadmin rmdistribution [ID]:                    remove distribution
3.  gcadmin addnodes gcChangeInfo.xml:              add nodes
4.  gcadmin rmnodes gcChangeInfo.xml:               remove nodes
5.  gcadmin showdistribution [node]:                show cluster
distribution, or segments on nodes when use parameter node
6.  gcadmin switchmode <mode>:                      switch cluster
mode,mode take value in: {normal|readonly|recovery}
7.  gcadmin showlock:       show current cluster lock information, include
lock name, lock owner ip address, etc
8.  gcadmin showddlevent [<tablename segname nodeip> | <tablename nodeip> |
<max_fevent_num>]: show current cluster ddl fail event, replicated table segname
is [n0]
9. gcadmin showdmlevent [<tablename segname nodeip> | <max_fevent_num>]:
show current cluster dml fail event, replicated table segname is [n0]
10. gcadmin showdmlstorageevent [table ID segname nodeip] | <max_fevent_num>]:
show current cluster dml storage fail event, replicated table segname is [n0]
11. gcadmin showcluster [c | d | f]:    show current cluster information,
include all nodes, cluster state and cluster node information
12. gcadmin getdistribution <ID> <distribution_info.xml> :  get distribution
information
13. gcadmin setnodestate ip <state>:                set one node
state,state take value in: unavailable
```

14. gcadmin --help: show help info
15. gcadmin -V,--version: show version info

4.1distribution 命令

语法:

gcadmin distribution <gcChangeInfo.xml> <p number> [d number] [pattern 1|2] [db_root_pwd password]

功能:

安装完集群，生成 distribution 时，需要使用该命令进行操作。

注：此命令需要切换到 dbaUser 用户下，才能正确执行。若使用其它用户执行生成 distribution 命令，将提示用户切换到 dbaUser 用户执行该命令，并报错退出。

gcChangeInfo.xml:生成 distribution 的 gnode 节点信息文件。集群安装成功后，执行安装操作命令的节点上，在安装包目录下，会生成一个 gcChangeInfo.xml 文件。该文件为 xml 格式，其根标签为<servers>，描述生成 distribution 的 gnode 节点信息；子标签为<rack>，即机架，描述的是机架与 gnode 节点对应关系。安装后生成的 gcChangeInfo.xml 中仅有一个<rack>，其中包含集群中的所有 gnode 节点信息，在使用 pattern 1 模式生成 distribution 时，可按机器部署情况插入多个<rack>标签，将 gnode 节点信息插入到对应的<rack>标签下。

p number：每个数据节点存放的分片数量，最小值为 1，p 值乘数据节点数不大于 65535，即集群总分片数不大于 65535，否则 gcadmin 将报错退出。

d number：每个分片的备份数量，取值为 0，1 或 2。若不输入参数 d，默认值为 1。

pattern number：生成 distribution 所使用模式，取值为 1 或 2，pattern

1 为负载均衡模式，pattern 2 为高可用模式。若不输入参数 pattern，默认使用 pattern 1 生成 distribution。

db_root_pwd:如果数据库 root 用户密码不为空，需要在执行命令的过程中，传入数据库 root 用户密码。目前密码中不支持单引号，其它特殊符号用单引号包围。

示例:

生成 distribution 有 pattern 1，pattern 2 和编写 distribution 配置信息文件三种模式。使用编写 distribution 配置信息文件方式生成 distribution 时，需手工编写一个配置文件，描述每个分片及其备份分片存放的节点信息。

pattern 1 为负载均衡模式，此模式下 gcChangeInfo.xml 中的每个 rack 中的节点为一组，每个 rack 中的节点上主分片的备份分片 1 存放到 gcChangeInfo.xml 中下一个 rack 中的节点上，备份分片 2 存放到 gcChangeInfo.xml 中上一个 rack 中的节点上。gcChangeInfo.xml 中的第一个 rack 的上一个 rack 为最后一个 rack，最后一个 rack 的下一个 rack 为第一个 rack。

使用 pattern 1 模式生成 distribution，每节点主分片数（即参数 p）必须小于每个 rack 的节点数，以此来保证备份分片分布均匀。每个 rack 包含的节点数尽可能相同，若 gcChangeInfo.xml 文件中有多余 1 个 rack 的节点数与其它 rack 不同，gcadmin 将会提示用户系统性能可能会下降，需用户确认后才能生成 distribution。

在安装好集群后，在执行安装操作的节点上，安装包目录下会生成一个包含所有 gnode 节点信息的 gcChangeInfo.xml 文件,如下所示:

```
<?xml version="1.0" encoding="utf-8"?>
<servers>

    <rack>
```

```
        <node ip="192.168.153.128"/>
        <node ip="192.168.153.129"/>
        <node ip="192.168.153.133"/>
        <node ip="192.168.153.134"/>
        <node ip="192.168.153.130"/>
        <node ip="192.168.88.137"/>
    </rack>

</servers>
```

图 4-1 gcChangeInfo.xml 文件

根据实际机架和机器部署情况，在该文件中插入<rack>标签，如下所示：

```
<?xml version="1.0" encoding="utf-8"?>
<servers>

    <rack>
        <node ip="192.168.153.128"/>
        <node ip="192.168.153.129"/>
    </rack>

    <rack>
        <node ip="192.168.153.133"/>
        <node ip="192.168.153.134"/>
      </rack>

    <rack>
        <node ip="192.168.153.130"/>
        <node ip="192.168.88.137"/>
      </rack>

</servers>
```

图 4-2 <rack>标签

使用 pattern 1 和该修改后的 gcChangeInfo.xml 文件生成 distribution 如下所示：

```
$ gcadmin distribution gcChangeInfo.xml p 1 d 2 pattern 1
```

```
gcadmin generate distribution ...

NOTE: node [192.168.153.129] is coordinator node, it shall be data node too
NOTE: node [192.168.153.130] is coordinator node, it shall be data node too
gcadmin generate distribution successful
```

图 4-3 生成 distribution

生成 distribution 后，可使用 showdistribution 和 showdistribution node 命令查看生成的 distribution 信息，如下所示：

```
$ gcadmin showdistribution

                Distribution ID: 1 | State: new | Total segment num: 6

      Primary Segment Node IP          Segment ID          Duplicate Segment node
IP
    ========================================================================
========
    |       192.168.153.128         |       1           |       192.168.153.134
|                                                                 192.168.88.137
    ------------------------------------------------------------------------
--------
    |       192.168.153.129         |       2           |       192.168.153.133
|                                                                 192.168.153.130
    ------------------------------------------------------------------------
--------
    |       192.168.153.133         |       3           |       192.168.88.137
|                                                                 192.168.153.129
    ------------------------------------------------------------------------
--------
    |       192.168.153.134         |       4           |       192.168.153.130
|                                                                 192.168.153.128
    ------------------------------------------------------------------------
--------
    |       192.168.153.130         |       5           |       192.168.153.129
```

```
|                                                                    192.168.153.134
    ------------------------------------------------------------------------------
---------
    |       192.168.188.137          |       6           |       192.168.153.128
|                                                                    192.168.153.133
    ==============================================================================
=========
```

图 4-4 showdistribution

```
    $ gcadmin showdistribution node

                          Distribution ID: 1 | State: new | Total segment num: 5

    ==============================================================================
=========================
    |nodes      | 192.168.153.128 | 192.168.153.129 | 192.168.153.133 |
192.168.153.134 | 192.168.153.130 | 192.168.88.137 |
    ------------------------------------------------------------------------------
-------------------------
    |primary    |       1         |       2         |       3         |       4
|       5         |       6         |
    |segments   |                 |                 |                 |
|                 |                 |
    ------------------------------------------------------------------------------
-------------------------
    |duplicate |       6         |       5         |       2         |       1
|       4         |       3         |
    |segments 1|                 |                 |                 |
|                 |                 |
    ------------------------------------------------------------------------------
-------------------------
    |duplicate |       4         |       5         |       6         |       5
|       2         |       1         |
    |segments 2|                 |                 |                 |
|                 |                 |
    ==============================================================================
=========================
```

图 4-5 showdistribution node

pattern 2 模式为高可用模式，此模式下生成的 distribution 将每个 data 节点的备份分片 1 存放到下一个 data 节点上，备份分片 2 存放到上一个 data 节点上。使用 pattern 2 模式配置文件 gcChangeInfo.xml 中仅需一个 rack 即可，有多个 rack 也作为一个 rack 处理。

生成 distribution 的配置文件如下所示

```
<?xml version="1.0" encoding="utf-8"?>
<servers>

    <rack>
        <node ip="192.168.153.128"/>
        <node ip="192.168.153.129"/>
        <node ip="192.168.153.133"/>
    </rack>
    <rack>
        <node ip="192.168.153.134"/>
        <node ip="192.168.153.130"/>
        <node ip="192.168.88.137"/>
      </rack>

</servers>
```

图 4-6 生成 distribution 的配置文件

使用 pattern 2 和该配置文件生成 distribution 如下所示

```
$ gcadmin distribution gcChangeInfo.xml p 2 d 2 pattern 2
gcadmin generate distribution ...

gcadmin generate distribution successful
```

图 4-7 生成 distribution

生成 distribution 后，可使用 showdistribution 和 showdistribution node 命令查看生成的 distribution 信息，如下所示：

```
$ gcadmin showdistribution

                Distribution ID: 21 | State: new | Total segment num: 12

      Primary Segment Node IP          Segment ID         Duplicate Segment node
IP
    ========================================================================
========
    |       192.168.153.128         |       1          |       192.168.153.129
|                                                        192.168.88.137
    ------------------------------------------------------------------------
---------
    |       192.168.153.129         |       2          |       192.168.153.133
|                                                        192.168.153.128
    ------------------------------------------------------------------------
---------
    |       192.168.153.133         |       3          |       192.168.153.134
|                                                        192.168.153.129
    ------------------------------------------------------------------------
---------
    |       192.168.153.134         |       4          |       192.168.153.130
|                                                        192.168.153.133
    ------------------------------------------------------------------------
---------
    |       192.168.153.130         |       5          |       192.168.88.137
|                                                        192.168.153.134
    ------------------------------------------------------------------------
---------
    |       192.168.188.137         |       6          |       192.168.153.128
|                                                        192.168.153.130
    ------------------------------------------------------------------------
---------
    |       192.168.153.128         |       7          |       192.168.153.129
|                                                        192.168.88.137
    ------------------------------------------------------------------------
```

```
---------
    |       192.168.153.129        |       8          |       192.168.153.133
|                                                              192.168.153.128
    ------------------------------------------------------------------------
---------
    |       192.168.153.133        |       9          |       192.168.153.134
|                                                              192.168.153.129
    ------------------------------------------------------------------------
---------
    |       192.168.153.134        |       10         |       192.168.153.130
|                                                              192.168.153.133
    ------------------------------------------------------------------------
---------
    |       192.168.153.130        |       11         |       192.168.88.137
|                                                              192.168.153.134
    ------------------------------------------------------------------------
---------
    |       192.168.188.137        |       12         |       192.168.153.128
|                                                              192.168.153.130
========================================================================
=====
```

图 4-8 showdistribution 命令

```
    $ gcadmin showdistribution node

                          Distribution ID: 21 | State: new | Total segment num:
12

    ========================================================================
=========================
    |nodes      | 192.168.153.128 | 192.168.153.129 | 192.168.153.133 |
192.168.153.134 | 192.168.153.130 | 192.168.88.137 |
    ------------------------------------------------------------------------
-------------------------
    |primary    |       1         |       2         |       3         |       4
|       5         |       6         |
    |segments   |       7         |       8         |       9         |       10
```

```
|       11        |       12        |
    ------------------------------------------------------------------------
-------------------------
    |duplicate |       6         |       1         |       2         |       3
|       4         |       5         |
    |segments 1|       12        |       7         |       8         |       9
|       10        |       11        |
    ------------------------------------------------------------------------
-------------------------
    |duplicate |       2         |       3         |       4         |       5
|       6         |       1         |
    |segments 2|       8         |       9         |       10        |       11
|       12        |       7         |
    ========================================================================
=========================
```

图 4-9 showdistribution node 命令

编写 distribution 配置信息文件模式需手工编写一个 distribution 分片配置的 xml 文件，在文件中指明每个分片的主/备分片存放的节点。使用该方式生成 distribution 无需输入参数 p，d 和 pattern。生成 distribution 的 gcChangeInfo.xml 文件如下所示

```
    <?xml version="1.0" encoding="utf-8"?>
    <servers>

        <cfgFile file="distribution.xml"/>

    </servers>
```

图 4-10 gcChangeInfo.xml 文件

distribution 分片配置信息文件 distribution.xml 如下所示

```
    <?xml version='1.0' encoding="utf-8"?>
    distributions>
       <distribution>
           <segments>
               <segment>
```

```
                <primarynode ip="192.168.153.125"/>

                <duplicatenodes>
                    <duplicatenode ip="192.168.153.126"/>
                    <duplicatenode ip="192.168.153.127"/>
                </duplicatenodes>
            </segment>

            <segment>
                <primarynode ip="192.168.153.128"/>

                <duplicatenodes>
                    <duplicatenode ip="192.168.153.129"/>
                </duplicatenodes>
            </segment>

        </segments>
    </distribution>
</distributions>
```

图 4-11 分片配置信息文件 distribution.xml

使用编写 distribution 信息文件模式生成 distribution 如下所示

```
$ gcadmin distribution gcChangeInfo.xml
gcadmin generate distribution ...

gcadmin generate distribution successful
```

图 4-12 生成 distribution

```
$ gcadmin showdistribution

            Distribution ID: 3 | State: new | Total segment num: 2

    Primary Segment Node IP         Segment ID         Duplicate Segment node
IP
    ================================================================================
=========
```

```
    |        192.168.153.125        |        1        |        192.168.153.126
|
    |                               |                 |        192.168.153.137
|
    ---------------------------------------------------------------------------
---------
    |        192.168.153.128        |        2        |        192.168.153.129
|
    ===========================================================================
=========
```

图 4-13 showdistribution

4.2rmdistribution 命令

语法:

gcadmin rmdistribution [ID]

ID: distribution id。

功能:

从集群中删除指定 id 的 distribution。若不输入 distribution id，则默认删除旧的 distribution，集群中只有一个 distribution 时则默认删除该 distribution。

注 1: 若 nodedatamap 中有要删除的 distribution ID，即该 distribution 为正在使用，则无法删除该 distribution，gcadmin 将报错退出。待删除的 distribution 有 DDL，DML 或 DMLSTORAGE event，需先清除 event 才可删除该 distribution，gcadmin 将报错退出。

注 2: 若 distribution 为正在使用，需先执行 refreshnodedatamap drop <ID>操作才可删除。若 distribution 中有 fevent log 需先清除才可删除该 distribution。

此命令需要切换到 dbaUser 用户下，才能正确执行，否则 gcadmin 将提示

切换用户执行该命令，并报错退出。

示例如下：

```
$ gcadmin rmdistribution 1
distribution: id [1] is current distribution
it will be removed now
please ensure this is ok, input y or n: y
gcadmin remove distribution [1] success
```

图 4-14 报错信息

注 3：如果 gc_stats_table 和 gc_stats_column 表使用了将被删除的 distribution id，那么用户首先需要将 gc_stats_table 和 gc_stats_column 两张表 rebalance 到另一个 distribution id，然后再执行 refreshnodedatamap drop 操作。

4.3addnodes 命令

语法：

gcadmin addnodes gcChangeInfo.xml

gcChangeInfo.xml：要添加的数据节点信息文件。集群安装成功后生成该文件，并使用该文件自动调用 addnodes 命令。用户执行该命令时可手工编写该文件。

功能：

将 gcChangeInfo.xml 中指定 data 节点添加到集群中。集群安装成功后会自动调用此命令，将安装成功的数据节点添加到集群中，无需用户手工执行 addnodes 命令。人为手动多次调用该命令将会提示用户节点已添加到集群，退出。

gcChangeInfo.xml 为要添加的数据节点信息，其中仅需包含一个<rack>即可，使用多个 rack 指定多个节点信息与一个 rack 指定所有节点信息效果相同。

注：该命令为系统内部命令，系统在安装集群后会自动调用，不建议用户使用。

使用一个<rack>标签的 gcChangeInfo.xml 文件如下所示：

```
<?xml version="1.0" encoding="utf-8"?>
<servers>
    <rack>
        <node ip="192.168.153.126"/>
        <node ip="192.168.153.130 "/>
        <node ip="192.168.88.137"/>
    </rack>
</servers>
```

图 4-15 gcChangeInfo.xml 文件

示例：

执行 addnodes 命令将节点加入集群

```
$ gcadmin addnodes gcChangeInfo.xml
gcadmin add nodes ...

gcadmin addnodes successful
```

图 4-16 执行 addnodes 命令

执行 addnodes 命令成功，使用 showcluster 命令查看集群

```
$ gcadmin showcluster
CLUSTER STATE:  ACTIVE
CLUSTER MODE:   NORMAL

============================================================
|            GBASE COORDINATOR CLUSTER INFORMATION          |
============================================================
|   NodeName   |    IpAddress    |gcware |gcluster |DataState |
------------------------------------------------------------
| coordinator1 | 192.168.153.129 | OPEN  |  OPEN   |    0     |
```

```
--------------------------------------------------------------
==============================================
|             GBASE DATA CLUSTER INFORMATION             |
==============================================
|NodeName |     IpAddress     |gnode |syncserver |DataState |
----------------------------------------------------------
|  node1  | 192.168.153.126 | OPEN |    OPEN    |    0     |
----------------------------------------------------------
|  node2  | 192.168.153.130 | OPEN |    OPEN    |    0     |
----------------------------------------------------------
|  node3  | 192.168.153.137 | OPEN |    OPEN    |    0     |
----------------------------------------------------------
```

图 4-17 使用 showcluster 命令查看集群

4.4rmnodes 命令

语法:

gcadmin rmnodes gcChangeInfo.xml

gcChangeInfo.xml: 要删除的数据节点信息文件。

功能:

将 gcChangeInfo.xml 中指定数据节点从集群中移除。

注: 要删除的数据节点必须是已添加到当前集群中的数据节点，并且是未被任何 distribution 使用的节点。

gcChangeInfo.xml 为要移除的数据节点信息，如下所示

```
<?xml version="1.0" encoding="utf-8"?>
<servers>
    <rack>
        <node ip="192.168.153.126"/>
        <node ip="192.168.153.130 "/>
        <node ip="192.168.88.137"/>
    </rack>
```

```
</servers>
```

图 4-18 gcChangeInfo.xml

示例:

执行 rmnodes 命令将节点从集群移除

```
$ gcadmin rmnodes gcChangeInfo.xml
gcadmin remove nodes ...

node [192.168.153.126] had been removed
node [192.168.153.130] had been removed
node [192.168.88.137] had been removed

gcadmin rmnodes success
```

图 4-19 执行 rmnodes 命令

执行 rmnodes 命令成功，使用 showcluster 命令查看集群

```
=============================================================
|             GBASE COORDINATOR CLUSTER INFORMATION          |
=============================================================
|   NodeName    |     IpAddress     |gcware |gcluster |DataState |
-------------------------------------------------------------
| coordinator1 | 192.168.153.129 | OPEN  |  OPEN   |    0     |
-------------------------------------------------------------
================================================
|          GBASE DATA CLUSTER INFORMATION          |
================================================
|NodeName |IpAddress |gnode |syncserver |DataState |
------------------------------------------------
```

图 4-20 使用 showcluster 命令查看集群

4.5 showdistribution 命令

语法:

gcadmin showdistribution [node]

node:显示 distribution 中每个 node 所包含的主备分片，可选参数。

功能:

显示 distribution 信息。

示例:

使用 gcadmin distribution gcChangeInfo.xml p 2 d 2 pattern 1 命令生成 distribution，gcChangeInfo.xml 如下所示:

```
<?xml version="1.0" encoding="utf-8"?>
<servers>

    <rack>
        <node ip="192.168.153.128"/>
        <node ip="192.168.153.129"/>
        <node ip="192.168.153.133"/>
    </rack>

    <rack>
        <node ip="192.168.153.134"/>
        <node ip="192.168.153.130"/>
        <node ip="192.168.88.137"/>
      </rack>

</servers>
```

图 4-21 gcChangeInfo.xml

生成 distribution 后，使用命令 gcadmin showdistribution 命令，不输入参数 node，则按照分片顺序显示 distribution 信息，如下所示:

```
$ gcadmin showdistribution

                Distribution ID: 24 | State: new | Total segment num: 12
```

```
        Primary Segment Node IP          Segment ID          Duplicate Segment node
IP
    ================================================================================
=========
    |       192.168.153.128          |       1          |       192.168.153.129
|                                                                  192.168.88.137
    --------------------------------------------------------------------------------
---------
    |       192.168.153.129          |       2          |       192.168.153.133
|                                                                  192.168.153.128
    --------------------------------------------------------------------------------
---------
    |       192.168.153.133          |       3          |       192.168.153.134
|                                                                  192.168.153.129
    --------------------------------------------------------------------------------
---------
    |       192.168.153.134          |       4          |       192.168.153.130
|                                                                  192.168.153.133
    --------------------------------------------------------------------------------
---------
    |       192.168.153.130          |       5          |       192.168.88.137
|                                                                  192.168.153.134
    --------------------------------------------------------------------------------
---------
    |       192.168.188.137          |       6          |       192.168.153.128
|                                                                  192.168.153.130
    --------------------------------------------------------------------------------
---------
    |       192.168.153.128          |       7          |       192.168.153.129
|                                                                  192.168.88.137
    --------------------------------------------------------------------------------
---------
    |       192.168.153.129          |       8          |       192.168.153.133
|                                                                  192.168.153.128
    --------------------------------------------------------------------------------
---------
```

```
    |       192.168.153.133        |       9       |       192.168.153.134
|                                                       192.168.153.129
    ------------------------------------------------------------------------
--------
    |       192.168.153.134        |       10      |       192.168.153.130
|                                                       192.168.153.133
    ------------------------------------------------------------------------
--------
    |       192.168.153.130        |       11      |       192.168.88.137
|                                                       192.168.153.134
    ------------------------------------------------------------------------
--------
    |       192.168.188.137        |       12      |       192.168.153.128
|                                                       192.168.153.130
========================================================================
=====
```

图 4-22 显示 distribution 信息

使用命令 gcadmin showdistribution node 命令，按照节点显示 distribution 分片信息，如下所示：

```
    $ gcadmin showdistribution node

                        Distribution ID: 24 | State: new | Total segment num:
12

    ========================================================================
=========================
    |nodes      | 192.168.153.128 | 192.168.153.129 | 192.168.153.133 |
192.168.153.134 | 192.168.153.130 | 192.168.88.137 |
    ------------------------------------------------------------------------
-------------------------
    |primary    |       1         |       2         |       3         |       4
|       5         |       6         |
    |segments   |       7         |       8         |       9         |       10
|       11        |       12        |
    ------------------------------------------------------------------------
```

```
------------------------
    |duplicate |       6         |       4         |       5         |       3
|       1         |       2         |
    |segments 1|       11        |       12        |       10        |       8
|       9         |       7         |
    ------------------------------------------------------------------------
------------------------
    |duplicate |       2         |       3         |       1         |       5
|       6         |       4         |
    |segments 2|       9         |       7         |       8         |       12
|       10        |       11        |
    ========================================================================
========================
```

图 4-23 显示 distribution 分片信息

4.6 switchmode 命令

语法:

gcadmin switchmode <mode>

mode: normal, readonly 或 recovery。

功能: 将集群状态切换为 normal 状态, readonly 状态或 recovery 状态。

示例:

```
$ gcadmin switchmode normal


========== switch cluster mode...
  switch pre mode:                [NORMAL]
  switch mode to                  [NORMAL]
  switch after mode:                [NORMAL]
```

图 4-24 switchmode 命令

4.7 showlock 命令

语法:

gcadmin showlock

showlock：查看集群中的锁。

功能：

查看目前集群中存在的锁。

包括锁的名称，锁的拥有者，锁的创建时间，锁的备注，该锁是否是孤儿锁，以及锁的类型。孤儿锁是指用户指定该锁异常退出后不释放的锁，目前无该使用方式。

示例：

```
    $ gcadmin showlock

+=================================================================================
==================================+
     |                                                    GCLUSTER LOCK
|

+=================================================================================
==================================+

+-----------------------------------------+--------------+---------------
-----------+-------------+------+----+
     |               Lock name                    |     owner      |
content          | create time  |orphan|type|

+-----------------------------------------+--------------+---------------
-----------+-------------+------+----+
     |                 test                       |192.168.153.129|
LOCK_DMLEX:6742(LMP:14664) |20151202103331|FALSE |S    |

+-----------------------------------------+--------------+---------------
-----------+-------------+------+----+
     |                 test                       |192.168.153.129|
```

```
LOCK_DMLEX:6730(LMP:14652) |20151202103331|FALSE |S    |

+-----------------------------------------+-------------+---------------
-----------+-------------+------+----+
     |                  test                          |192.168.153.129|
LOCK_DMLEX:6740(LMP:14662) |20151202103333|FALSE |S    |

+-----------------------------------------+-------------+---------------
-----------+-------------+------+----+
     |test.tt3580d5f90-b287-4199-b057-e6fbd44b5bfa |192.168.153.129|
LOCK_DMLEX:6742(LMP:14664) |20151202103333|FALSE |E    |

+-----------------------------------------+-------------+---------------
-----------+-------------+------+----+
     |                test.tt3
|192.168.153.129|LOCK_DDL_DML:6742(LMP:14664)|20151202103333|FALSE |S    |

+-----------------------------------------+-------------+---------------
-----------+-------------+------+----+
     |test.tt3.09b5beec-1ef7-4fa6-9850-c4217a781e0f|192.168.153.129|
LOCK_DMLEX:6742(LMP:14664) |20151202103333|FALSE |E    |

+-----------------------------------------+-------------+---------------
-----------+-------------+------+----+

     Total : 6
```

图 4-25 showlock 命令

显示结果列名解释如下：

- Lock name：锁的名字。
- owner：发起该加锁操作的节点 IP。
- content：锁的备注信息。
- create time：锁的创建时间（以加锁节点的时间为准）。

- orphan：是否是孤儿锁。
- type：锁的类型，S 表示共享锁，E 表示独占锁。

4.8 showddlevent 命令

语法：

gcadmin showddlevent [<tablename segname nodeip> | <tablename nodeip> | <max_fevent_num>]

tablename：格式为 db_name.table_name，例如：ssbm.dwdate。

segname：表分片的名字，例如建立一张表名为 t 的分布表，在第一个节点上的表分片名称就是 n1，在第二个节点上的表分片名称就是 n2,……，命名以此类推。

nodeip：节点机器的 IP。

max_fevent_num：gcadmin 需要根据用户设定的最大条数返回 ddlevent 信息，如果给定最大条数大于已有的信息总量，则返回全部信息，否则返回指定数量的信息。如不指定最大条数，则默认返回 16 条，后续 event 将不显示。

注：该命令格式与 8.5.1.2 的 showddlevent 命令相同，但 fevent 信息格式有变化，系统中有 DDL event 时不能升级。

功能：

查看集群 DDL 操作的错误日志。

gcadmin showddlevent：查看集群当前所有 DDL 操作产生的错误日志。

gcadmin showddlevent tablename segname nodeip：查看针对某个表的某个分片在特定节点上的 DDL 错误日志。

gcadmin showddlevent tablename nodeip：查看针对某个表在某个节点上的 DDL 错误日志。

示例：

```
$ gcadmin showddlevent
Event count:2
Event ID:    2
ObjectName: test.t1
Fail Node Copy:
-----------------------------------------------------
NodeID: 2123999424  NodeIP:192.168.153.126  FAILURE

Fail Data Copy:
-----------------------------------------------------
NodeIP: 192.168.153.126 FAILURE



Event ID:    3
ObjectName: test.t2
Fail Node Copy:
-----------------------------------------------------
NodeID: 2123999424  NodeIP:192.168.153.126  FAILURE

Fail Data Copy:
-----------------------------------------------------
    NodeIP: 192.168.153.126 FAILURE
```

图 4-26 showddlevent 命令

4.9 showdmlevent 命令

语法：

gcadmin showdmlevent [<tablename segname nodeip> | <max_fevent_num>]

tablename：格式为 db_name.table_name，例如：ssbm.dwdate。

segname：表分片的名字，例如建立一张表名为 t 的分布表，在第一个节点上的表分片名称就是 n1，在第二个节点上的表分片名称就是 n2,……，命名

以此类推。

nodeip：节点机器的 IP。

max_fevent_num：gcadmin 工具需要根据用户设定的最大条数返回 dmlevent 信息，如果给定最大条数大于已有的信息总量，则返回全部信息，否则返回指定数量的信息。如不指定最大条数，则默认返回 16 条，后续 event 将不显示。

注：该命令格式与 8.5.1.2 的 showdmlevent 命令相同，但 fevent 信息格式有变化，系统中有 DDL event 时不能升级。

功能：

查看集群 DML 错误日志。

gcadmin showdmlevent：查看当前集群所有 DML 操作产生的错误日志。

gcadmin showdmlevent tablename sgname nodeip：查看针对某个表的某个分片在某节点上的 DML 错误日志。

示例：

```
$ gcadmin showdmlevent
Event count:2
Event ID:    3
ObjectName: test.t1

Fail Data Copy:
-------------------------------------------------
SegName: n3 SCN: 0  NodeIP: 192.168.153.126 FAILURE
SegName: n4 SCN: 0  NodeIP: 192.168.153.126 FAILURE



Event ID:    2
ObjectName: test.t4

Fail Data Copy:
-------------------------------------------------
```

```
SegName: n4 SCN: 0  NodeIP: 192.168.153.126 FAILURE
SegName: n3 SCN: 0  NodeIP: 192.168.153.126 FAILURE
```

图 4-27 showdmlevent 命令

4.10 showdmlstorageevent 命令

语法:

gcadmin showdmlstorageevent [table ID segname nodeip] | <max_fevent_num>]

注:该命令格式与 8.5.1.2 的 showdmlstorageevent 命令相同,但 fevent 信息格式有变化, 系统中有 DML storage event 时不能升级。

功能:

该命令用来显示当前集群是否有表的数据损坏信息;

示例:

```
$ gcadmin showdmlstorageevent
Event count:2
Event ID:    5
ObjectName: test.t1
TableID: 26

Fail Data Copy:
-----------------------------------------------------
SegName: n2 NodeIP: 192.168.153.129 FAILURE



Event ID:    6
ObjectName: test.t2
TableID: 32
```

```
Fail Data Copy:
------------------------------------------------
    SegName: n2 NodeIP: 192.168.153.129 FAILURE
```

图 4-28 showdmlstorageevent 命令

4.11 showcluster 命令

语法:

gcadmin showcluster [c | d | f]

c: 仅显示集群 coordinator 节点信息。

d: 仅显示集群 data 节点信息。

f: 按照 xml 格式显示集群节点信息。

功能:

显示集群节点信息，若不输入参数 c 或 d 则显示集群所有节点信息。

示例:

显示所有节点信息，示例如下

```
$ gcadmin showcluster
CLUSTER STATE:  ACTIVE
CLUSTER MODE:   NORMAL

=============================================================
|            GBASE COORDINATOR CLUSTER INFORMATION          |
=============================================================
|   NodeName   |    IpAddress    |gcware |gcluster |DataState |
-------------------------------------------------------------
| coordinator1 | 192.168.153.129 | OPEN  |  OPEN   |    0     |
-------------------------------------------------------------
=============================================================
```

```
|              GBASE DATA CLUSTER INFORMATION               |
=============================================================
|NodeName |     IpAddress     |gnode |syncserver |DataState |
-------------------------------------------------------------
|  node1  | 192.168.153.126 | OPEN |    OPEN    |    0     |
-------------------------------------------------------------
|  node2  | 192.168.153.129 | OPEN |    OPEN    |    0     |
-------------------------------------------------------------
|  node3  | 192.168.153.125 | OPEN |    OPEN    |    0     |
-------------------------------------------------------------
```

图 4-29 showcluster 命令显示所有节点

仅显示 coordinator 节点信息，示例如下

```
$ gcadmin showcluster c
CLUSTER STATE:  ACTIVE
CLUSTER MODE:   NORMAL


============================================================
|            GBASE COORDINATOR CLUSTER INFORMATION          |
============================================================
|   NodeName    |     IpAddress     |gcware |gcluster |DataState |
------------------------------------------------------------
| coordinator1 | 192.168.153.129 | OPEN  |  OPEN   |    0     |
------------------------------------------------------------
```

图 4-30 showcluster 命令仅显示 coordinator 节点

仅显示 data 节点信息，示例如下

```
$ gcadmin showcluster d
CLUSTER STATE:  ACTIVE
CLUSTER MODE:   NORMAL


=============================================================
|              GBASE DATA CLUSTER INFORMATION               |
=============================================================
|NodeName |     IpAddress     |gnode |syncserver |DataState |
-------------------------------------------------------------
```

```
| node1 | 192.168.153.126 | OPEN | OPEN | 0 |
-----------------------------------------------------------
| node2 | 192.168.153.129 | OPEN | OPEN | 0 |
-----------------------------------------------------------
| node3 | 192.168.153.125 | OPEN | OPEN | 0 |
-----------------------------------------------------------
```

图 4-31 showcluster 命令仅显示 data 节点

按照 xml 格式显示 data 节点信息，示例如下：

```
$ gcadmin showcluster d f
<?xml version='1.0' encoding="utf-8"?>
<ClusterInfo>
    <ClusterState>ACTIVED</ClusterState>
    <ClusterMode>NORMAL</ClusterMode>

    <DataserverNodes>
        <DataServerNode>
            <NodeName>node1</NodeName>
            <IpAddress>192.168.153.126</IpAddress>
            <gnode>OPEN</gnode>
            <syncserver>OPEN</syncserver>
            <DataState>0</DataState>
        </DataServerNode>
        <DataServerNode>
            <NodeName>node4</NodeName>
            <IpAddress>192.168.153.129</IpAddress>
            <gnode>OPEN</gnode>
            <syncserver>OPEN</syncserver>
            <DataState>0</DataState>
        </DataServerNode>
        <DataServerNode>
            <NodeName>node5</NodeName>
            <IpAddress>192.168.153.125</IpAddress>
            <gnode>OPEN</gnode>
            <syncserver>OPEN</syncserver>
```

```
            <DataState>0</DataState>
        </DataServerNode>
    </DataserverNodes>
</ClusterInfo>
```

图 4-32 xml 格式显示 data 节点信息

集群中节点较多时，查询异常节点信息时可使用 grep 命令：

gcadmin showcluster | grep -iE "close|offline|unavailable|replace| 1 "

示例如下：

```
[gbase@8a ~]$ gcadmin showcluster | grep -iE "close|offline|unavailable|replace| 1 "
| coordinator1 | 192.168.6.111 | OFFLINE |         |          |
| coordinator2 | 192.168.6.112 |  OPEN   |  CLOSE  |    0     |
| coordinator3 | 192.168.6.113 |  OPEN   |  OPEN   |    1     |
|  node1  | 192.168.6.111 | OFFLINE |          |          |
|  node2  | 192.168.6.112 |   OPEN  |  CLOSE   |    0     |
|  node3  | 192.168.6.113 |   OPEN  |  OPEN    |    1     |
```

4.12 getdistribution 命令

语法：

gcadmin getdistribution <ID> <distribution_info.xml>

ID：要获取的 distribution id。

distribution_info.xml：保存 distribution 信息的文件名。

功能：

将指定 id 的 distribution 信息保存的指定的文件中，生成的文件为 xml 文件，用户可修改改文件中的分片信息，然后使用该文件重新生成 distribution。

注：指定的 distribution 需为已存在的 distribution，若指定存放 distribution 信息的文件已存在，则其内容会被清空再写入 distribution 分片信息。

示例：

```
$ gcadmin getdistribution 6 dstb_info
gcadmin getdistribution 6 dstb_info ...

get segments information
write segments information to file [dstb_info]

gcadmin getdistribution information successful
```

图 4-33 getdistribution 命令

命令执行成功，生成的 dstb_info 文件内容如下

```
<?xml version='1.0' encoding="utf-8"?>
<distributions>
    <distribution>
    <segments>
        <segments>
            <primarynode ip=" 192.168.153.129" />

                <duplicatenodes>
                    <duplicatenode ip=" 192.168.153.125" />
                    <duplicatenode ip=" 192.168.153.126" />
                 </duplicatenodes>
        </segments>

    <segments>
            <primarynode ip=" 192.168.153.125" />

                <duplicatenodes>
                    <duplicatenode ip=" 192.168.153.129" />
                 </duplicatenodes>
        </segments>
</segments>
    </distribution>
</distributions>
```

图 4-34 getdistribution 命令

4.13 setnodestate 命令

语法:

gcadmin setnodestate ip <state>

ip: 要设置状态的节点 ip

State: 设置后的节点状态:

1) UNAVAILABLE;

2) failure 标识集群故障，相当于 offline 这时 dml、ddl 将不会下发到该节点直接记录 fevent;

3) normal: 当节点故障解决后可以直接将节点置为 normal，这相当于节点重新 online，这时 gcrecover 将恢复之前记录的 feventlog，新发起的 ddl、dml 将重新下发到该节点。

功能:

设置一个节点的状态。

注: 集群存在多个 distribution 时，如果设置一个节点为 unavailable 状态，会导致任何一个 distribution 中出现某个分片的主副分片都不可用得情况，则设置失败。

4.14 showfailover 命令

语法:

gcadmin showfailover

功能:

显示当前保留在 gcware 中的所有 failover 信息。

示例:

```
$ gcadmin showfailover
+=============================================================================
==========================================================+
|                                                                GCLUSTER
FAILOVER                                                                    |
+=============================================================================
==========================================================+
+-----------+-----------+----------+-----------+----------+--------------
---+------+----------------+----------------+----------------+
| commit  id |  database  |   table   |    scn    |   type    |   create  time
| state |  original node  |  takeover node  | takeover number |
+-----------+-----------+----------+-----------+----------+--------------
---+------+----------------+----------------+----------------+
|     1      |    test    |    t1     |     1      |    ddl    |
20161019101114  |   5   | 192.168.153.130 |     0.0.0.0     |        0        |
+-----------+-----------+----------+-----------+----------+--------------
---+------+----------------+----------------+----------------+
```

其中:

Commit id : failover 的唯一标识，64 位数字

Database : 数据库名

Table: 表名

Scn: scn 号

Type: ddl/dml/rebalance

Create time: 当前节点创建 failover 的时间

State: failover 对应的状态当前如下:

- ✓ init: 初始化，对应显示数字 0
- ✓ add_res : 添加集群锁，对应显示数字 1
- ✓ set_info : 设置 failover 信息，对应显示数字 2
- ✓ set_status: 设置分片状态，对应显示数字 3

- ✓ set_rebalance_info：设置 rebalance 信息，对应显示数字 4
- ✓ set_rebalance_status：设置 rebalance 状态，对应显示数字 5

original node：发起节点

takeover node：当前接管节点，如果没有发生接管则显示为 0.0.0.0

Takeover num： failover 的接管次数，gcware 通知 gcluster 接管后这个值就加 1。

4.15 showfailoverdetail 命令

语法：

gcadmin showfailoverdetail <commitid> [xml_file_name]

功能：

显示当前保留在 gcware 中的所有 failover 信息。

参数说明：

Commitid：failover 的唯一标识，该参数必须输入

Xml_fil_name：保存 failover 信息的文件名，可选参数，若不输入则将 failover 信息打印到屏幕

示例：

```
$ gcadmin showfailoverdetail 1

<?xml version='1.0' encoding="utf-8"?>
<failover_detail>
    <failover_information>
        <commit_id>1</commit_id>
        <database>test</database>
        <table>t1</table>
        <scn>1</scn>
        <type>ddl</type>
```

```
        <create_time>20161019101114</create_time>
        <state>5</state>
        <original_node>192.168.153.130</original_node>
        <takeover_node>0.0.0.0</takeover_node>
        <takeover_number>0</takeover_number>
    </failover_information>
    <content>create table t1(a int)</content>
    <status>
    </status>
    <rebalance_information>
        <distribution_id>1</distribution_id>
        <current_scn>10</current_scn>
        <current_step>3</current_step>
        <table>tmpt1</table>
    </rebalance_information>
    <sdm>
        <slice_dm>from_slice node1.n1.row10.block_id1</slice_dm>
        <slice_dm>from_slice node2.n2.row9.block_id2</slice_dm>
        <slice_dm>from_slice node3.n3.row8.block_id3</slice_dm>
    </sdm>
</failover_detail>
```

内容包括:

failover_information: Failover 相关信息包括(commit_id, database, table, scn, type, create_time, state, original_node, takeover_node, takeover_number)

content : failover 完整信息，最大 256k.

status: failover 操作的对象状态即对应的是那个节点那个分片的状态。

例如 node1.n1 init 含义就是 node1 节点上 n1 分片尚未提交处于初始化状态。

rebalance_information: rebalance 独有信息 (含 distribution_id, current_scn, current_step, 中间表名)，ddl dml 显示为空标签

sdm: rebanlance 独有信息，ddl dml 显示为空标签。包含如下字段：

- ✓ NodeId.Suffix ： 某个节点的某个分片
- ✓ curRowid: rebalance 执行到哪一行了
- ✓ Blockid BlockNum: 上一批 rebalance 执行到哪一行。

4.16 help 命令

语法：

gcadmin --help

功能：

查看 gcadmin 所有命令参数的帮助信息。

4.17 version 命令

语法：

gcadmin -V 或者 gcadmin --version

功能：

查看 gcadmin 的版本信息。

4.18 upgrade 命令

语法：

gcadmin upgrade

功能：

将 GBase 8a 8.5.1.2 的 safegroup 升级转换为对应大规模集群的

distribution。升级时 gcadmin 会读取/var/lib/gcware/CIB.xml 文件，生成对应的分片信息和对应的 distribution。新生成的 distribution 不会修改原 8.5.1.2 集群的分片和备份关系。

升级完成后 gcadmin 会删除/var/lib/gcware/CIB.xml 文件。

注：升级前系统中有 DDL/DML/DML storage event 时，不允许升级到新版本。

该命令为系统内部命令，在系统升级后会自动调用，在 gcadmin --help 中也不显示这个命令。

4.19 replacenodes 命令

语法：

gcadmin replacenodes <hosts>

hosts：要替换的节点 ip 列表，用逗号分隔。

功能：

集群节点损坏,用新的机器替换损坏的节点机器,要求新机器已经安装好集群服务。

备注：

此命令为内部使用命令,在 gcadmin --help 中也不显示这个命令。

5 数据加载

5.1功能简介

在 V8.6.2.X 版本 GBase 8a MPP Cluster 中，集群加载功能直接集成在 GBase 8a MPP Cluster 内部，不需要额外部署外部加载工具。

与 V8.5.1.2 版本集群加载工具相比，新版加载工具具备如下一些特性和优点：

1) 与集群高度集成，方便部署；

2) 提供面向用户的 SQL 接口，集群和单机加载方式统一，更符合用户的使用习惯；

3) 支持多加载机对单表的并行加载，最大化加载性能；

4) 支持从通用数据服务器拉取数据，支持 ftp/http/hdfs/sftp 等多种协议；

5) 支持普通文本、gzip 压缩、snappy 压缩、lzo 压缩等多种格式数据文件；

6) 支持普通文本与定长文本的加载（format 3 和 format 4），并与 V8.5.1.2 版本格式兼容；

7) 支持错误数据溯源功能，可以准确定位错误数据在源文件中的位置；

8) 加载性能可以随着集群规模的扩展而持续提升；

5.2数据服务器配置

V8.6.2.X 版本集群加载功能支持从通用数据服务器拉取数据，支持 ftp/http/hdfs/sftp 等多种协议。

以下简要描述 Red Hat Enterprise Linux 6.2 平台上 FTP、HTTP、HDFS、

SFTP 四种通用文件服务器的配置方法。

5.2.1FTP 服务器配置

使用 vsftp 搭建 FTP 服务器

1. 查看是否已安装 vsftpd

```
# rpm -qa vsftpd
vsftpd-2.2.2-6.el6_0.1.x86_64
```

2. 安装 vsftpd

```
# rpm -ivh vsftpd-2.2.2-6.el6_0.1.x86_64.rpm
```

3. 修改 FTP 服务器默认配置

```
# vim /etc/vsftpd/vsftpd.conf
```

```
# 表示允许匿名用户登录（默认为 YES）.
anonymous_enable=YES

# 表示允许本地用户登录（默认为 YES）.
local_enable=YES

# 表示开放对本地用户的写权限（默认 YES，如仅用作加载文件服务器，可改为 NO）.
write_enable=NO

# 设置本地用户的文件生成掩码（默认对本地用户的文件生成掩码是 077，可改为 022）
local_umask=022

# 允许匿名 FTP 用户上载文件（默认为 NO）.
#anon_upload_enable=YES

# 允许匿名 FTP 用户创建目录（默认为 NO）.
#anon_mkdir_write_enable=YES

# 启用 FTP 数据端口的连接请求（默认为 YES）.
connect_from_port_20=YES
```

```
    # 使用 PAM 认证的配置文件名，文件位于/etc/pam.d 目录下
    pam_service_name=vsftpd

    # 是否使用 userlist 文件控制访问 FTP 服务器
    userlist_enable=YES

    # 设置禁止访问的文件或目录
    #deny_file={*.mp3,*.mov,.private}

    # 设置隐藏的文件或目录
    #hide_file={*.mp3,.hidden,hide*,h?}

    # 设置 FTP 被动模式开放端口范围（默认为 0，表示任意可用端口）
    pasv_min_port=20001
    pasv_max_port=21000

    # 设置允许的最大客户连接数（默认为 2000）
    max_clients=2000

    # 设置每个 IP 上允许的最大客户连接数（默认为 50）
    max_per_ip=50

    # 用于被动传输方式的连接超时（默认为 60）
    accept_timeout=60

    # 用于主动传输方式的连接超时（默认为 60）
    connect_timeout=60

    # 无进度状态下的数据传输超时（默认为 300）
    data_connection_timeout=300

    # 空闲连接超时（默认为 300）
    idle_session_timeout=300

    # 是否使用系统调用 sendfile 优化传输（默认为 YES，使用 nfs 等网络盘时应设置为
NO）
```

```
use_sendfile=YES

# 设置非匿名登录用户的主目录
#local_root=/var/ftp/pub

更多的配置可查看 vsftpd.conf 文档
# man vsftpd.conf
```

在集群最大并发加载任务数为 N，单加载任务最大加载机数（max_data_processors）为 M 时，部分参数最小值和推荐值如下：

参数名称	默认值	最小值	推荐值
max_clients	2000	M*N	M*N*2
max_per_ip	50	N	N*2
pasv_min_port	0	max-min : M*N	max-min : M*N*2
pasv_max_port	0		

4. 配置允许或禁止访问 FTP 服务器的用户列表（可跳过）

```
# vim /etc/vsftpd/user_list
```

当/etc/vsftpd/vsftpd.conf 中配置如下时，禁止/etc/vsftpd/user_list 中的所有用户访问 FTP 服务器

userlist_enable=YES

userlist_deny=YES（缺省为 YES）

当/etc/vsftpd/vsftpd.conf 中配置如下时，允许/etc/vsftpd/user_list 中的所有用户访问 FTP 服务器

userlist_enable=YES

userlist_deny=NO

5. 配置禁止访问 FTP 服务器的用户列表（可跳过）

```
# vim /etc/vsftpd/ftpusers
```

6. 关闭 SELINUX 功能或更改其配置（两种方式二选一即可）

1）关闭 SELINUX 功能

```
# vim /etc/selinux/config

# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=disable
# SELINUXTYPE= can take one of these two values:
#     targeted - Targeted processes are protected,
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

重启或者执行

```
# setenforce 0
```

2）更改 SELINUX 配置

```
# setsebool ftp_home_dir 1
```

注：当用浏览器访问 FTP 服务器遇到“500 OOPS: cannot change directory:/home/...”时，可能为此问题。

7. 关闭或配置防火墙

1）关闭防火墙

停止防火墙服务

```
# service iptables stop
    iptables: 清除防火墙规则:                                  [确定]
    iptables: 将链设置为政策 ACCEPT: filter                    [确定]
    iptables: 正在卸载模块:                                    [确定]
```

查看防火墙是否在开机时自动启动

```
# chkconfig --list iptables
iptables    0:关闭  1:关闭  2:启用  3:启用  4:启用  5:启用  6:关闭
```

禁止防火墙在开机时自动启动

```
# chkconfig iptables off
```

或

```
# chkconfig iptables off --level 2345
```

设置后防火墙在开机时自动启动状态

```
# chkconfig --list iptables
iptables    0:关闭  1:关闭  2:关闭  3:关闭  4:关闭  5:关闭  6:关闭
```

2）配置防火墙

设置默认规则

```
# iptables -A INPUT -j DROP（注：添加此条规则会阻止未处理的传入数据包，如在此规则之前未添加允许规则将会阻止远程连接）
# iptables -A FORWARD -j ACCEPT
```

开放 FTP 端口

```
# iptables -I INPUT -p tcp --dport 21 -j ACCEPT

# iptables -I OUTPUT -p tcp --sport 21 -j ACCEPT

# iptables -I INPUT -p tcp --dport 20 -j ACCEPT

# iptables -I OUTPUT -p tcp --sport 20 -j ACCEPT

# iptables -I INPUT -p tcp --dport 20001:21000 -j ACCEPT

# iptables -I OUTPUT -p tcp --sport 20001:21000 -j ACCEPT
```

保存防火墙设置

```
# iptables-save > /etc/sysconfig/iptables
```

8. 启动 vsftpd 服务并设置为开机启动项

```
# service vsftpd start
为 vsftpd 启动 vsftpd:                                    [确定]

# chkconfig vsftpd on
```

9. 复制文件到 FTP 目录

1) 如未设置 local_root=/var/ftp/pub 时，复制文件到/home/xxxx（用户的 home 目录）

2) 如已设置 local_root=/var/ftp/pub 时，复制文件到/var/ftp/pub

3) 如已设置 anonymous_enable=YES 时,复制文件到/var/ftp 或/var/ftp/pub（匿名登录的主目录）

5.2.2HTTP 服务器配置

使用 apache 搭建 HTTP 文件服务器

1、安装 apr 和 httpd

```
# rpm -ivh
apr-1.3.9-3.el6_1.2.x86_64.rpm
apr-util-1.3.9-3.el6_0.1.x86_64.rpm apr-util-ldap-1.3.9-3.el6_0.1.x86_64.rpm
# rpm -ivh
httpd-2.2.15-15.el6.x86_64.rpm
httpd-manual-2.2.15-15.el6.noarch.rpm  httpd-tools-2.2.15-15.el6.x86_64.rpm
```

2、修改 HTTP 服务器默认配置

```
# vim /etc/httpd/conf/httpd.conf
```

修改服务器名称

```
# If your host doesn't have a registered DNS name, enter its IP address here.
# You will have to access it by its address anyway, and this will make
# redirections work in a sensible way.
#ServerName www.example.com:80
ServerName 192.168.10.114:80
```

修改以下位置，将其中的"/var/www/html"修改为"/var/www/files"

也可直接使用"/var/www/html"作为文件存储位置，跳过这一步

```
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.
#
#DocumentRoot "/var/www/html"
DocumentRoot "/var/www/files"

#
# This should be changed to whatever you set DocumentRoot to.
#
#<Directory "/var/www/html">
<Directory "/var/www/files">
```

修改其它参数

```
# 是否使用 memory-mapping，默认值 on，挂载 nfs 系统时设为 off
# EnableMMAP off

# 是否使用 sendfile 系统调用，默认值 on，挂载 nfs 系统时设为 off
# EnableSendfile off
```

```
# 连接超时，默认值为 60
# Timeout 60
```

禁用长文件名截断，添加以下配置。

```
<IfModule autoindex_module>
  IndexOptions NameWidth=*
</IfModule>
或者
IndexOptions FancyIndexing VersionSort NameWidth*
```

注：如果不禁用长文件名截断，由于 Apache 会返回不完整的文件名，会导致使用通配符方式加载 HTTP 文件时发生错误。

3、编辑默认欢迎页配置

```
# vim /etc/httpd/conf.d/welcome.conf
```

注释掉以下几行（默认如果 html 下没有默认页面将显示 403 错误页面）

```
#<LocationMatch "^/+$">
#    Options -Indexes
#    ErrorDocument 403 /error/noindex.html
#</LocationMatch>
```

4、关闭或配置防火墙

1）关闭防火墙

停止防火墙服务

```
# service iptables stop
iptables: 清除防火墙规则:                                  [确定]
iptables: 将链设置为政策 ACCEPT: filter                    [确定]
iptables: 正在卸载模块:                                    [确定]
```

查看防火墙是否在开机时自动启动

```
# chkconfig --list iptables
Iptables    0:关闭  1:关闭  2:启用  3:启用  4:启用  5:启用  6:关闭
```

禁止防火墙在开机时自动启动

```
# chkconfig iptables off
```

或

```
# chkconfig iptables off --level 2345
```

设置后防火墙在开机时自动启动状态

```
# chkconfig --list iptables
Iptables    0:关闭  1:关闭  2:关闭  3:关闭  4:关闭  5:关闭  6:关闭
```

2）配置防火墙

设置默认规则

```
# iptables -A INPUT -j DROP
# iptables -A FORWARD -j ACCEPT
```

开放 HTTP 端口

```
# iptables -I INPUT -p tcp -m tcp --dport 80 -j ACCEPT
# iptables -I OUTPUT -p tcp -m tcp --sport 80 -j ACCEPT
```

保存防火墙设置

```
# iptables-save > /etc/sysconfig/iptables
```

5、启动 httpd 服务并设置为开机启动项

```
# service httpd start
正在启动 httpd:                                          [确定]

# chkconfig httpd on
```

6、将数据文件复制到/var/www/files（或/var/www/html）下

7、用浏览器访问 http://192.168.10.114 即可看到文件列表（前面配置的 ServerName 192.168.10.114:80）

5.2.3 HDFS 服务器配置

使用 Apache Hadoop 2.6.0 搭建 HDFS 服务器

1、 Hadoop 集群环境准备

操作系统用户：gbase

集群各节点间的 ssh 互信已建立。

集群已配置 C3 工具。

开源产品版本：

Apache Hadoop 2.6.0

JVM 1.6 或 1.7 版本

集群节点功能规划示例：

IP	主机名	功能
192.168.10.114	ch-10-114	NameNode, DataNode
192.168.10.115	ch-10-115	DataNode
192.168.10.116	ch-10-116	DataNode

2、 主机名配置

各节点的主机名需要正确配置，以 192.168.10.114 节点示例如下，其他节点直接拷贝该配置即可。

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1     localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.10.114 ch-10-114
192.168.10.115 ch-10-115
192.168.10.116 ch-10-116
```

注意：第一行如果配置成如下形式是错误的，安装完成后会出现 Hadoop 的 Datanode 无法连接 Namenode 的情况。

```
127.0.0.1           ch-10-114    localhost    localhost.localdomain    localhost4
localhost4.localdomain4
```

如果集群中没有 DNS 服务器可以解析 Hadoop 的 Namenode 和 Datanode 的主机名,则需要在每个执行加载任务的 coordinator 节点,以及集群中的每个 data 节点上，都要配置/etc/hosts 文件，在其中加入如上 Hadoop 的 Namenode 和 Datanode 的 IP 地址和主机名映射。如未配置/etc/hosts 文件，执行加载 HDFS 服务器上文件时，会报类似“Couldn't resolve host name”字样的错误。

检查方法：

1）通过 jps 查看，发现 DataNode 已经启动，但是检查 DataNode 上的日志，发现 DataNode 在不断尝试连接 NameNode 节点的 9000 端口（HDFS 的 RPC 端口）。

2）在 NameNode 节点执行 netstat -an，看到如下信息：

```
$ netstat -an | grep 9000
tcp   0   0 127.0.0.1:9000     0.0.0.0:*         LISTEN
```

错误原因：TCP 监听的 IP 是 127.0.0.1，导致只有本机能够连接到 9000 端口。原因是 NameNode 的/etc/hosts 文件配置错误。

解决办法：去掉第一行的红色字体（ch-10-114），或者将第一行内容后置均可。

```
192.168.10.114 ch-10-114
192.168.10.115 ch-10-115
192.168.10.116 ch-10-116
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1     localhost localhost.localdomain localhost6 localhost6.localdomain6
```

重启 HDFS，再次用 netstat -an | grep 9000 查看，端口和 IP 正确。

```
$ netstat -an | grep 9000
tcp   0   0 192.168.10.114:9000     0.0.0.0:*         LISTEN
```

3、 目录规划

目录	用途
/home/gbase/bin	放置 Hadoop 生态系统，包括 Hadoop 等
/home/gbase/hdfs	放置 HDFS 文件，包括 tmp、name、data

添加环境变量${HADOOP_HOME}

```
$ echo "export HADOOP_HOME=/home/gbase/bin/Hadoop-2.6.0">> ~/.bashrc
$. ~/.bashrc
```

注：下文中的${HADOOP_HOME}指/home/gbase/bin/Hadoop-2.6.0

4、 准备 Hadoop 2.6.0

把 hadoop-2.6.0.tar.gz 解压到各节点的/home/gbase/bin。

```
$ tar xfz hadoop-2.6.0.tar.gz -C /home/gbase/bin
```

5、 配置 hadoop-env.sh

文件路径：${HADOOP_HOME}/etc/hadoop/hadoop-env.sh

```
$ cd ${HADOOP_HOME}
$ vi etc/hadoop/hadoop-env.sh
```

Name node 和 Data node 均按如下配置。

把 export JAVA_HOME=$JAVA_HOME 修改为：

export JAVA_HOME=/usr/lib/jvm/jre-1.6.0-openjdk.x86_64

把 export HADOOP_CONF_DIR=${HADOOP_CONF_DIR:-"/etc/hadoop"}修改为：

export HADOOP_CONF_DIR=/home/gbase/bin/hadoop-2.6.0/etc/hadoop

6、 配置 core-site.xml 文件

文件路径：${HADOOP_HOME}/etc/hadoop/core-site.xml

```
$ cd ${HADOOP_HOME}
$ vi etc/hadoop/core-site.xml
```

Name node 和 Data node 均按如下配置

```
    <configuration>
        <property>
            <name>fs.default.name</name>
            <value>hdfs://ch-10-114:9000</value>
        </property>
        <property>
            <name>hadoop.tmp.dir</name>
            <value>file:/home/gbase/hdfs/tmp</value>
        </property>
    </configuration>
```

7、 配置 hdfs-site.xml

文件路径：${HADOOP_HOME}/etc/hadoop/hdfs-site.xml

```
$ cd ${HADOOP_HOME}
$ vi etc/hadoop/hdfs-site.xml
```

Name node 配置

```
    <configuration>
        <property>
            <name>dfs.replication</name>
            <value>2</value>
        </property>
        <property>
            <name>dfs.name.dir</name>
            <value>file:/home/gbase/hdfs/name</value>
            <description>name node dir </description>
```

```
        </property>
        <property>
            <name>dfs.permissions</name>
            <value>false</value>
        </property>
    </configuration>
```

Data Node 配置

```
    <configuration>
        <property>
            <name>dfs.data.dir</name>
            <value>file:/home/gbase/hdfs/data</value>
            <description>data node dir</description>
        </property>
    </configuration>
```

8、 配置 Masters 和 Slaves

文件路径: ${HADOOP_HOME}/etc/hadoop/masters

${HADOOP_HOME}/etc/hadoop/slaves

只需要在 NameNode 节点配置即可。

```
$ cd ${HADOOP_HOME}
$ vi etc/hadoop/masters
```

${HADOOP_HOME}/etc/hadoop/masters 文件内容

```
    ch-10-114
```

```
$ cd ${HADOOP_HOME}
$ vi etc/hadoop/slaves
```

${HADOOP_HOME}/etc/hadoop/slaves 文件内容

```
ch-10-114
ch-10-115
ch-10-116
```

9、 格式化 NameNode

NameNode 的格式化需要在启动 HDFS 之前进行。

```
$ cexec rm -fr /home/gbase/hdfs/*
$ cd ${HADOOP_HOME}
$ bin/hdfs namenode -format
```

10、 启动 HDFS

```
$ cd ${HADOOP_HOME}
$ sbin/start-dfs.sh
```

启动完毕后，通过 jps 检查各节点的进程，如下即为正确启动了。

```
$ cexec jps
************************ test ************************
--------- 192.168.10.114---------
31318 SecondaryNameNode
31133 NameNode
31554 Jps
--------- 192.168.10.115---------
10835 DataNode
11000 Jps
--------- 192.168.10.116---------
10145 DataNode
10317 Jps
```

11、 停止 HDFS

```
$ cd ${HADOOP_HOME}
```

```
$ sbin/stop-dfs.sh
```

5.2.4SFTP 服务器配置

SFTP 服务无需布署额外的软件包，开启 sshd 服务即可。

1. 查看是否已启动 sshd 服务

```
# service sshd status
openssh-daemon (pid  2243) is running...
```

2. 默认配置的目录访问。使用默认配置时，sshd 不限制用户的目录访问。用户通过 sftp 登录后，可以在有访问权限的任何目录间跳转。在这种情况下，以下加载语句的 URL 中的文件路径为系统的绝对路径：

load data infile

'sftp://gbase:gbase@192.168.10.114/opt/data/test.tbl'into table test.t

data_format 3;

3. 修改 sshd 默认配置。

当集群并发加载任务数和单任务最大加载机数较大时，会出现 sftp 文件加载失败的情况，此时可按以下方式修改 sshd 配置文件。

编辑/etc/ssh/sshd_config 文件

```
# vi /etc/ssh/sshd_config
```

按以下加粗字体内容修改配置文件

```
# MaxStartups 的值表示为“start:rate:full”，默认值为 10:30:100，当未认证连接数达到 start(10)时，新的连接尝试有“rate/100” (30%)的可能会被 sshd 拒绝，当未认证连接数达到 full(100)时，所有新的连接尝试都会被拒绝。
# 在集群最大并发加载任务数为 N，单加载任务最大加载机数（max_data_processors）为 M 时 MaxStartups 的推荐值为 M*N+10:30:M*N*2
# MaxStartups 10:30:100
MaxStartups 20:30:100
```

出于安全原因，希望对 sftp 登录用户的访问权限进行限制，只能让用户在自己的 home 目录下活动。

启用 sshd 目录锁定功能需要使用到 chroot，openssh 4.8p1 以后都支持 chroot，可用以下命令检查当前系统的 openssh 版本。

```
    # ssh -V
OpenSSH_5.3p1, OpenSSL 1.0.0-fips 29 Mar 2010
```

编辑/etc/ssh/sshd_config 文件

```
    # vi /etc/ssh/sshd_config
```

按以下加粗字体内容修改配置文件

```
# override default of no subsystems
# 修改默认子系统为 internal-sftp
    #Subsystem       sftp    /usr/libexec/openssh/sftp-server
    Subsystem       sftp    internal-sftp

# Example of overriding settings on a per-user basis
# Match User sftp 表示以下规则仅匹配于名称为 sftp 的用户，如果需要匹配多个用户名，多个用户名间用逗号分隔。也可用 Match Group sftp 来匹配名称为 sftp 的组，同样如果需要匹配多个组，多个组名间用逗号分隔
Match User sftp
#       X11Forwarding no
#       AllowTcpForwarding no
# 强制执行进程内 sftp server，忽略~/.ssh/rc 文件中的命令
    ForceCommand internal-sftp
# 用 chroot 将用户的根目录指定到%h，%h 代表用户的 home 目录，可选的参数还有%u，代表用户名
    ChrootDirectory %h
```

注意：sftp 目录的权限配置要点：

1）由 ChrootDirectory 指定的目录开始一直向上到系统根目录为止的目录拥有者都只能是 root

2）由 ChrootDirectory 指定的目录开始一直向上到系统根目录为止都不可有群组写入的权限，即权限值不能高于 755

4. 配置完成后重启 sshd 服务

```
    # service sshd restart
```

在已配置目录锁定的情况下,以下加载语句的 URL 中的文件路径为系统的相对路径，test.tbl 的绝对路径应为/home/gbase/opt/data/test.tbl

load data infile

'sftp://gbase:gbase@192.168.10.114/opt/data/test.tbl'into table test.t

data_format 3;

5.2.5KAFKA 服务器配置

使用 ZooKeeper 3.4.6 和 kafka_2.11-0.10.1.0.tgz 搭建 KAFKA 服务器

1. ZooKeeper 安装与部署

推荐使用 Apache ZooKeeper 3.4.6 部署 ZooKeeper 集群

a. 准备 ZooKeeper

把 zookeeper-3.4.6.tar.gz 解压到各节点的/home/gbase/bin。

```
$ tar xfz zookeeper-3.4.6.tar.gz -C /home/gbase/bin
```

b. 配置 ZooKeeper

将/home/gbase/bin/zookeeper-3.4.6/conf/zoo_sample.cfg 文件复制改名为 zoo.cfg 并编辑新文件

```
$ cd /home/gbase/bin/zookeeper-3.4.6
$ cp conf/zoo_sample.cfg conf/zoo.cfg
$ vi conf/zoo.cfg
```

c. 配置 zoo.cfg 中的 dataDir 项，所有节点相同。

the directory where the snapshot is stored.

do not use /tmp for storage, /tmp here is just

example sakes.

dataDir=/home/gbase/zookeeper #用于指定 zookeeper 数据存放路径

cilentPort=2181 #用户可以根据现场实际情况修改该值

在 zoo.cfg 末尾增加以下配置，所有节点相同。如为单机环境则不需添加如下配置:

server.0=192.168.10.114:2888:3888

server.1=192.168.10.115:2888:3888

server.2=192.168.10.116:2888:3888

在各节点的 dataDir 目录（/home/gbase/zookeeper）下新建名称为 myid 的文件，分别向自的 myid 中写入服务器号（server.x 中点后的数字）。

比如：

server.0=192.168.10.114:2888:3888，192.168.10.114 的服务器号为 0；

server.1=192.168.10.115:2888:3888，192.168.10.115 的服务器号为 1；

server.2=192.168.10.116:2888:3888，192.168.10.116 的服务器号为 2；

那么 192.168.10.114 节点的/home/gbase/zookeeper/myid 文件的内容就为 0

```
$ cat myid
0
```

d. 启动 ZooKeeper

在所有节点上执行：

```
$ cd /home/gbase/bin/zookeeper-3.4.6
$ bin/zkServer.sh start
```

然后可执行以下命令，检查指定服务器是否成功启动：

```
$ bin/zkCli.sh -server xxx.xxx.xxx.xxx:2181
```

e. 停止 ZooKeeper

在所有节点上执行：

```
$ cd /home/gbase/bin/zookeeper-3.4.6
$ bin/zkServer.sh stop
```

2. Kafka 安装与部署

推荐使用 kafka_2.11-0.10.1.0.tgz 部署 kafka 集群

a. 准备 Kafka

把 kafka_2.11-0.10.1.0.tgz 解压到各节点的/opt/kafka

```
$ tar xvf kafka_2.11-0.10.1.0.tgz -C /opt/kafka
```

b. 配置 Kafka

```
$ vi config/server.properties
```

#从 0 开始递增，每个节点 id 不能相同

broker.id=0

#kafka 数据的存放路径

log.dirs=/opt/tmp/kafka

#broker server IP，如果该参数未指定，kafka 监控节点所有 IP，但 8a 集群的所有节点都需要将 kafka 集群的 host name 添加到/etc/hosts 内

host.name=192.168.6.72

#broker server 服务端口

port=9092(默认)

#配置 zookeeper 链接

zookeeper.connect=10.10.10.226:2181

#zookeeper 集群的地址，可以是多个，多个之间用逗号分割

c. 启动 Kafka

```
$cd kafka_2.11-0.10.1.0

$bin/kafka-server-start.sh -daemon config/server.properties
```

如果没有报错退出，则启动成功。

d. kafka 安装包版本注意事项

如果用户使用低于推荐版本(0.10.1)的安装包版本部署 kafka 集群，在使用 kafka 加载的时候，用户需要根据具体部署的 kafka 版本在集群变量中设置 gbase_kafka_broker_version。

示例：

```
set gbase_kafka_broker_version='0.9.0';
```

e. kafka 相关命令

```
1. 创建一个有 6 个 partition 以及 1 个副本名为 t1 的 topic
bin/kafka-topics.sh --create --zookeeper 10.10.10.224:2181
--replication-factor 2 --partitions 6 --topic t1
2. 列出当前 kafka 集群中存在的 topic
$bin/kafka-topics.sh --list --zookeeper 10.10.10.224:2181
3. 查看名为 t1 的 topic 各个 partiton 的状态
bin/kafka-topics.sh --describe --zookeeper 10.10.10.224:2181 --topic
t1
4. 查看名为 t1 的 topic 各个 partition 的结束 offset
bin/kafka-run-class.sh kafka.tools.GetOffsetShell --broker-list
10.10.10.224:9092 --time -1 --topic t1
5. 查看名为 t1 的 topic 各个 partition 的起始 offset
bin/kafka-run-class.sh kafka.tools.GetOffsetShell --broker-list
10.10.10.224:9092 --time -2 --topic t1
```

5.3加载状态监控

加载任务启动后，可以通过 SQL 方式查看本次加载任务的状态信息。状态信息表中记录正在运行的所有加载任务的状态信息。

Field	Type	Null	Key	Default	Extra
SCN	bigint(20)	NO		0	
DB_NAME	varchar(64)	NO			
TB_NAME	varchar(64)	NO			
IP	varchar(20)	NO			
STATE	varchar(20)	NO			
START_TIME	datetime	NO		0000-00-00 00:00:00	
ELAPSED_TIME	bigint(20)	NO		0	
AVG_SPEED	bigint(20)	NO		0	
PROGRESS	bigint(8)	NO		0	
TOTAL_SIZE	bigint(20)	NO		0	
LOADED_SIZE	bigint(20)	NO		0	
LOADED_RECORDS	bigint(20)	NO		0	
SKIPPED_RECORDS	bigint(20)	NO		0	
DATA_SOURCE	varchar(1024)	NO			
SQL_CMD	varchar(4096)	NO			

内存表各字段定义：

Field	Description
SCN	SCN number
DB_NAME	库名
TB_NAME	表名
IP	加载机 IP
STATE	加载状态
START_TIME	加载启动时间
ELAPSED_TIME	加载结束时间
AVG_SPEED	加载速度
PROGRESS	加载进度
TOTAL_SIZE	文件总长度
LOADED_SIZE	已加载数据量
LOADED_RECORDS	已加载数据条数
SKIPPED_RECORDS	跳过数据条数
DATA_SOURCE	数据源
SQL_CMD	加载任务的 SQL

语法：

```
gbase> use information_schema;
gbase> select * from load_status;
```

5.4加载日志汇总与查询

日志汇总与查询功能，将一次加载的错误数据日志与溯源信息日志汇总至加载发起节点，并提供相应的查询，检索日志的功能。本功能依赖表 GNS 功能的开启。

通过变量 gbase_loader_logs_dir 变量指定日志文件汇总路径，默认汇总至加载发起节点的 gcluster 日志目录($GCLUSTER_HOME/log/gcluster/)下的 loader_logs 目录，并在该路径下建一个以本次 TASK_ID 命名的子文件夹，将汇总日志存放于该子文件夹下。加载完成时，在该子文件夹下创建一个以 TASK_ID_loader_result.log 命名的日志，并将本次加载的结果信息写入该日志文件。该变量支持 set 方式修改与配置文件方式修改。

```
gbase> show variables like '%loader_logs%';
+----------------------+-----------------------------------------+
| Variable_name        | Value                                   |
+----------------------+-----------------------------------------+
| gbase_loader_logs_dir | /opt/gcluster/log/gcluster/loader_logs/ |
+----------------------+-----------------------------------------+
1 row in set (Elapsed: 00:00:00.00)
```

通过变量 gbase_loader_logs_collect 变更控制集群加载日志汇总功能的开关，有效值[0,1],默认值为 1，表示开启汇总功能，该变量支持 set 方式修改与配置文件方式修改。

对于集群加载，如果 gbase_loader_logs_collect 为 1，错误数据与溯源信息汇总到加载发起节点，并存储到 gbase_loader_logs_dir 指定目录， 否则不进行错误数据与溯源信息日志的汇总。

注意：

1) 对于日志的命名，汇总功能不变更日志的文件名，遵循现有命名规则；
2) 对于日志的个数，汇总功能也不对日志文件进行合并，即汇总功能只是将各个数据加载节点产生的日志文件汇总至加载发起节点。

如下图所示：

```
[gbase@RH_6_72 ~]$ ll /opt/gcluster/log/gcluster/loader_logs/
total 8
drwxrwxr-x 2 gbase gbase 4096 Dec 30 10:07 131075
drwxrwxr-x 2 gbase gbase 4096 Dec 30 10:08 131076
[gbase@RH_6_72 ~]$ ll /opt/gcluster/log/gcluster/loader_logs/131076
total 1280376
-rw-rw---- 1 gbase gbase 189965464 Dec 30 10:07 131076_test_lineitem_n1_192.168.6.72_20161230100740.err
-rw-rw---- 1 gbase gbase 137734542 Dec 30 10:07 131076_test_lineitem_n1_192.168.6.72_20161230100740.trc
-rw-rw---- 1 gbase gbase 189965260 Dec 30 10:07 131076_test_lineitem_n1_192.168.6.73_20161230100740.err
-rw-rw---- 1 gbase gbase 137825915 Dec 30 10:07 131076_test_lineitem_n1_192.168.6.73_20161230100740.trc
-rw-rw---- 1 gbase gbase 189965369 Dec 30 10:07 131076_test_lineitem_n2_192.168.6.74_20161230100740.err
-rw-rw---- 1 gbase gbase 137813242 Dec 30 10:07 131076_test_lineitem_n2_192.168.6.74_20161230100740.trc
-rw-rw---- 1 gbase gbase 189965273 Dec 30 10:07 131076_test_lineitem_n3_192.168.6.75_20161230100044.err
-rw-rw---- 1 gbase gbase 137850597 Dec 30 10:07 131076_test_lineitem_n3_192.168.6.75_20161230100044.trc
[gbase@RH_6_72 ~]$
```

5.4.1 加载错误数据与溯源信息检索

支持语法形式对于错误数据与溯源信息检索功能，具体语法如下：

show [gcluster] load logs task_id LIMIT {[offset,] row_count}

查询结果信息表定义如下：

字段	含义	类型	长度
TASK_ID	加载 ID	varchar	64
DB_NAME	加载库名	varchar	64
TB_NAME	加载表名	varchar	64
ERR_DATA_IP	产生错误数据的节点 IP	varchar	20
FILE_NAME	加载文件名	varchar	64
FILE_OFFSET	错误数据偏移量	varchar	64
RECORD_LEN	错误数据行长	varchar	64
ERR_COLUMN	错误数据列号	varchar	64
ERR_REASON	错误数据具体原因	varchar	1024
ERR_DATA	错误数据	varchar	4096

具体说明：

1) show 语法查询默认是返回 offset 从 0 到 length 10 的 10 条错误数据与溯源信息查询，如果想查询更多的数据可以调整 offset,length。
2) show 语法查询当前 coordinator 节点错误数据与溯源信息进行检索使用 show load logs task_id limit offset, row_count 进行查询，返回 row_count 条查询结果。

3) 查询所有 coordiantor 节点错误数据与溯源信息进行检索使用 show gcluster load logs task_id limit offset, row_count 进行查询，返回 row_count 条查询结果。

4) ERR_DATA 的长度定义为 4096 个字节。可以涵盖绝大多数场景，对于超过长度的错误数据，显示时做截断处理，实际读取 4096 个字节。

5) show 查询功能，只能查询当前汇总目录内的加载错误数据与溯源信息。即如果用户对 gbase_loader_logs_dir 做了变更后，将查询不到原指定目录中的数据。

6) Show 语法增加用户查询权限控制功能，默认仅能查询当前用户指定加载任务的错误数据与溯源信息，有 process 权限的用户可以查询其他用户指定加载任务的错误数据与溯源信息。

例如:

show load logs 100 显示 task_id 100 任务的前 10 条错误数据信息

show load logs 100 limit 5 显示 task_id 100 任务的前 5 条错误数据信息

show load logs 100 limit 0,5 显示 task_id 100 任务的前 5 条错误数据信息

show load logs 100 limit 1,5 显示 task_id 100 任务的从第 1 条开始的后面 5 条错误数据信息

show gcluster load logs 101 显示所有 coordinator 节点上 task_id 101 任务的前 10 条错误数据信息

示例 1: 查询 task_id 为 131076 次加载的前 10 条错误数据与溯源信息。

show load logs 131076

```
gbase> show load logs 131076;
+---------+---------+----------+-------------+-----------------------------------+-------------+------------+------------+-----------------------------------------------
-----------------------------------------------+
| task_id | db_name | tb_name  | err_data_ip  | file_name                         | file_offset | record_len | err_column | err_reason
                                               |
+---------+---------+----------+-------------+-----------------------------------+-------------+------------+------------+-----------------------------------------------
-----------------------------------------------+
| 131076  | test    | lineitem | 192.168.6.75 | ftp://192.168.6.72/pub/lineitem.tbl | 620227528   | 148        | 1          | data column size less than defined
|enticing, ironic packages try to sleep furi| |
| 131076  | test    | lineitem | 192.168.6.75 | ftp://192.168.6.72/pub/lineitem.tbl | 620227676   | 144        | 1          | data column size less than defined
ickly ironic somas x-ray: regular deposi|     |
| 131076  | test    | lineitem | 192.168.6.75 | ftp://192.168.6.72/pub/lineitem.tbl | 620227820   | 121        | 1          | data column size less than defined
deposits haggle p|                            |
| 131076  | test    | lineitem | 192.168.6.75 | ftp://192.168.6.72/pub/lineitem.tbl | 620227941   | 132        | 1          | data column size less than defined
tect fluffily. quickly final|                 |
| 131076  | test    | lineitem | 192.168.6.75 | ftp://192.168.6.72/pub/lineitem.tbl | 620228073   | 136        | 1          | data column size less than defined
nal excuses at the even, even id|             |
| 131076  | test    | lineitem | 192.168.6.75 | ftp://192.168.6.72/pub/lineitem.tbl | 620228209   | 121        | 1          | data column size less than defined
|carefully unusua|                            |
| 131076  | test    | lineitem | 192.168.6.75 | ftp://192.168.6.72/pub/lineitem.tbl | 620228330   | 115        | 1          | data column size less than defined
cajole slyl|                                  |
| 131076  | test    | lineitem | 192.168.6.75 | ftp://192.168.6.72/pub/lineitem.tbl | 620228445   | 120        | 1          | data column size less than defined
unusual asymptot|                             |
| 131076  | test    | lineitem | 192.168.6.75 | ftp://192.168.6.72/pub/lineitem.tbl | 620228565   | 133        | 1          | data column size less than defined
xpress, ironic foxes ought to|                |
| 131076  | test    | lineitem | 192.168.6.75 | ftp://192.168.6.72/pub/lineitem.tbl | 620228698   | 144        | 1          | data column size less than defined
efully unusual requests sleep slyly. flu|     |
+---------+---------+----------+-------------+-----------------------------------+-------------+------------+------------+-----------------------------------------------
-----------------------------------------------+
10 rows in set (Elapsed: 00:00:00.00)
```

示例 2：查询 task_id 为 131076 次加载的第 1506750 条开始的后 5 条错误数据相关信息。

```
gbase> show load logs 131076 limit 1506750,5;
+---------+---------+----------+-------------+-----------------------------------+-------------+------------+------------+-----------------------------------------
---------------------------------------+
| task_id | db_name | tb_name  | err_data_ip  | file_name                         | file_offset | record_len | err_column | err_reason
                                         |
+---------+---------+----------+-------------+-----------------------------------+-------------+------------+------------+-----------------------------------------
---------------------------------------+
| 131076  | test    | lineitem | 192.168.6.74 | ftp://192.168.6.72/pub/lineitem.tbl | 389379848   | 140        | 1          | data column size less than defi
unts alongside of the even dolphins | |
| 131076  | test    | lineitem | 192.168.6.74 | ftp://192.168.6.72/pub/lineitem.tbl | 389379988   | 114        | 1          | data column size less than defi
ously iron|                              |
| 131076  | test    | lineitem | 192.168.6.74 | ftp://192.168.6.72/pub/lineitem.tbl | 389380102   | 132        | 1          | data column size less than defi
nto beans eat slyly final re|            |
| 131076  | test    | lineitem | 192.168.6.74 | ftp://192.168.6.72/pub/lineitem.tbl | 389380234   | 120        | 1          | data column size less than defi
lithely enticing|                        |
| 131076  | test    | lineitem | 192.168.6.74 | ftp://192.168.6.72/pub/lineitem.tbl | 389380354   | 113        | 1          | data column size less than defi
ly ironic|                               |
+---------+---------+----------+-------------+-----------------------------------+-------------+------------+------------+-----------------------------------------
---------------------------------------+
5 rows in set (Elapsed: 00:00:03.25)
```

5.4.2 加载结果信息统计日志

加载完成时将加载结果信息写入日志文件 loader_result.log 中，加载结果信息是以'|'为列分隔符，以'\n'为行分隔符存储的普通文本文件，存放在发起节点 gcluster($GCLUSTER_HOM/log/gcluster/)日志目录，不支持指定存放路径。

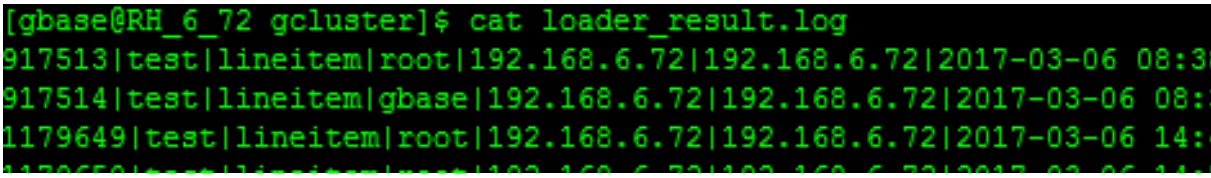
具体包括以下内容：

字段	含义

TASK_ID	加载 ID
DB_NAME	加载数据库名
TB_NAME	加载表名
USER	当前加载用户名
ACCESS_IP	加载发起点 IP
HOST_IP	客户端 IP
START_TIME	加载开始时间
END_TIME	加载结束时间
ELAPSED_TIME	加载耗时
TOTAL_SIZE	加载文件总大小
AVERAGE_SPEED	加载平均速度
LOADED_RECORDS	加载数据条数
SKIPPED_RECORDS	加载数据跳过条数
IGNORED_FILES	加载跳过的文件数
RESULT	加载结果
SQL_CMD	加载 SQL
MESSAGE	错误信息

注：SQL_CMD 与 MESSAGE 中的包含'\n'的情况，日志文件中以空格代替。

例如下图所示：



5.4.3 加载结果信息内存表查询

加载结果信息通过 information_schema 库内的 LOAD_RESULT 和 CLUSTER_LOAD_RESULT 表进行查询，如下图所示：

```
gbase> show tables like '%LOAD_RESULT%';
+-----------------------------------------------+
| Tables_in_information_schema (%LOAD_RESULT%)  |
+-----------------------------------------------+
| LOAD_RESULT                                   |
| CLUSTER_LOAD_RESULT                           |
+-----------------------------------------------+
2 rows in set (Elapsed: 00:00:00.00)
```

加载结果信息表定义与加载结果日志列定义一致，如下图所示：

```
[gbase@RH_6_72 gcluster]$ gccli -ugbase -pgbase20110531

GBase client 8.6.1.1 build 72003. Copyright (c) 2004-2017, GBase.  All Rights Reserved.

gbase> desc information_schema.load_result;
+-----------------+---------------+------+-----+---------------------+--------+
| Field           | Type          | Null | Key | Default             | Extra  |
+-----------------+---------------+------+-----+---------------------+--------+
| TASK_ID         | bigint(20)    | NO   |     | 0                   |        |
| DB_NAME         | varchar(64)   | NO   |     |                     |        |
| TB_NAME         | varchar(64)   | NO   |     |                     |        |
| USER            | varchar(64)   | NO   |     |                     |        |
| ACCESS_IP       | varchar(20)   | NO   |     |                     |        |
| HOST_IP         | varchar(20)   | NO   |     |                     |        |
| START_TIME      | datetime      | NO   |     | 0000-00-00 00:00:00 |        |
| END_TIME        | datetime      | NO   |     | 0000-00-00 00:00:00 |        |
| ELAPSED_TIME    | bigint(20)    | NO   |     | 0                   |        |
| TOTAL_SIZE      | bigint(20)    | NO   |     | 0                   |        |
| AVERAGE_SPEED   | bigint(20)    | NO   |     | 0                   |        |
| LOADED_RECORDS  | bigint(20)    | NO   |     | 0                   |        |
| SKIPPED_RECORDS | bigint(20)    | NO   |     | 0                   |        |
| IGNORED_FILES   | bigint(20)    | NO   |     | 0                   |        |
| RESULT          | varchar(20)   | NO   |     |                     |        |
| SQL_CMD         | varchar(4096) | NO   |     |                     |        |
| MESSAGE         | varchar(1024) | NO   |     |                     |        |
+-----------------+---------------+------+-----+---------------------+--------+
17 rows in set (Elapsed: 00:00:00.01)
```

注意：

1) 支持 select 查询形式，查询加载结果信息

2) 只查询当前 coordinator 节点，select 查询形式，查询加载信息，表名为：LOAD_RESULT 例如：

   select * from information_schema.load_result

3) 查询所有 coordinator 节点，select 查询形式，查询加载信息，表名为：CLUSTER_LOAD_RESULT

例如：

   select * from information_schema.cluster_load_result;

4) 加载结果信息查询功能实现用户权限控制，对于有 PROCESS 权限用户可以查询当前集群所有用户已经加载的信息，对于无该权限的用户只能查询自己已经加载的加载结果信息。

5.4.4 加载跳过文件列表日志回传

当加载过程中开启 SKIP_BAD_FILE 功能,即设置该参数为 1 时,如果 gbased 发现有跳过错误文件信息，将跳过错误数据文件信息回传至加载发起结节，并写入 gcluster 的 express 日志文件中。

本功能依赖表 GNS 功能的开启，如果 GNS 功能关闭，即 gbase_gns_share_connection = 0，gbased 发现有跳过错误文件时，将跳过错误数据文件信息写到 gbase 的 express 日志文件中，与现有加载行为一致。

5.4.5 加载错误数据、溯源信息和日志文件导出

导出加载错误数据、溯源信息和日志文件的 SQL 语法如下:

EXPORT [GCLUSTER] LOAD LOGS *task_id* TO 'export_dir' [WRITEMODE BY {NORMAL | OVERWRITES}]

导出结果表定义如下:

字段	含义	类型	长度
file_name	文件名	varchar	1024
file_size	文件大小	varchar	64

说明:

1) GCLUSTER 表示在所有 coodinator 节点上导出加载日志文件，缺省表示仅在当前 coodinator 上导出加载日志文件。

2) WRITEMODE BY 表示导出文件模式，可选值为 NORMAL 或 OVERWRITES，缺省值为 NORMAL，其中 NORMAL 表示当同名文件已存在时报错，OVERWRITES 表示自动覆盖同名文件。

3) task_id 表示加载任务号。

4) eport_dir 表示远程目录 URL，目前支持 SFTP 和 FTP 两种协议的 URL。

约束：

1) 导出的文件包括 loader_result.log 日志文件，*.trc 溯源信息文件和*.err 错误数据文件。

2) 只能导出当前日志汇总目录内的加载错误数据、溯源信息加载日志文件。如果修改了 gbase_loader_logs_dir 后，将无法导出原指定目录下的文件。

3) 支持查询权限控制功能，默认仅能导出当前用户指定加载任务的错误数据、溯源信息和日志文件，有 process 权限的用户可以导出其他用户指定加载任务的错误数据、溯源信息和日志文件。

示例：

1) 以覆盖方式导出当前 coordinator 上任务号为 100 的加载日志文件到 FTP 服务器的 pub 目录

EXPORT LOAD LOGS 100 TO 'ftp://192.168.0.1/pub' WRITEMODE BY OVERWRITES

2) 以覆盖方式导出所有 coordinator 上任务号为 100 的加载日志文件到 SFTP 服务器的 pub 目录

EXPORT GCLUSTER LOAD LOGS 100 TO 'sftp://gbase:gbase@192.168.0.1/pub' WRITEMODE BY OVERWRITES

5.5加载兼容工具使用说明

由于 V8.5.1.2 版本已经在很多客户现场上线使用。V8.6.2.X 版本加载的使用方式（SQL 加载的方式）与 V8.5.1.2 版本加载工具差异较大，为了降低客户产品升级时带来的额外工作量，数据加载提供了兼容 V8.5.1.2 版加载方式的加载兼容工具 dispcli。

dispcli 加载兼容工具的原理是,解析 V8.5.1.2 加载控制文件,将其中参数转换成加载 SQL,再调用 gccli 执行加载 SQL,收集执行结果,并将加载 SQL 的执行结果,转换为与 V8.5.1.2 加载工具执行结果兼容的格式输出。dispcli 加载兼容工具的实质与 SQL 加载一致。

V8.6.2.X 版本与 V8.5.1.2 版本相比,加载兼容工具与 V8.5.1.2 加载工具的部署方式和使用方式发生如下变化:

5.5.1 部署方式

1) V8.5.1.2 版本数据加载的前提条件是,除了安装 V8.5.1.2 版集群外,还需要额外部署与集群版本匹配的 dispcli/dispserver 两个组件,配合集群共同完成数据导入操作。具体部署方式请参照 V8.5.1.2 数据加载手册。

2) V8.6.2.X 版本加载兼容工具 dispci 加载数据的前提条件是,除了安装 V8.6.2.X 版集群外,还需要额外部署加载兼容工具 dispcli。且部署了数据文件服务器。

加载兼容工具的部署分在集群结点和非集群结点两个场景:

<1> 集群结点

第一步: 部署与配置数据文件服务器,具体细节请参考本章 6.5 节。保障直接使用 SQL 加载可以正常执行。

第二步: 加载兼容工具在集群结点的部署,非常简单,只需要部署在系统数据库用户下,即可以执行 gccli 的用户下即可。

<2> 非集群结点

加载兼容工具在非集群结点的部署,相对复杂一些,分以下几个步骤

第一步: 部署与配置数据文件服务器,具体细节请参考本章 6.5 节。保障直接使用 SQL 加载可以正常执行。这点与集群结点部署是一样的。

第二步: 在非集群结点的某用户下,配置集群客户端,具体配置请参考相关手册《GBase 8a MPP Cluster 安装手册》第 5 章内容。

第三步:同在集群结点部署加载兼容工具一样,需要把加载兼容工具部署在上述用户下。

5.5.2 使用方式

1）V8.5.1.2 集群加载工具的使用方式请参考相应版本的使用手册《GBase 8a MPP Cluster 数据抽取及加载工具参考手册》。

2）V8.6.2.X 集群加载兼容工具 dispcli 的使用方式

首先，加载兼容工具 dispcli 兼容 V8.5.1.2 版本 dispcli 的所有命令行参数与使用方式，但要求加载控制文件作为最后一个参数出现。否则加载兼容工具进行报错退出处理。

如：./dispcli -lc.log -h192.168.88.124 test.ctl

其次，由于 V8.6.2.X 版集群加载支持 FTP/HTTP/hadoop 等多种数据源，不支持本地加载，因此需要对 V8.5.1.2 版本的加载控制文件中的 file_list 参数做出调整。具体方法有两种：

1．直接修改 V8.5.1.2 版本加载控制文件

该方式需要修改现有的控制文件，具体实例如下所示：

V8.5.1.2 版本加载控制文件：

```
[test1]
disp_server=192.168.40.2:6666
table_name=lineitem
file_list=/opt /tpch_data/1s_tbl/lineitem.tbl
db_user=root
db_name=test
format=3
delimiter='|a'
```

V8.6.2.X 版本加载控制文件：

```
[test1]
disp_server=192.168.40.2:6666
table_name=lineitem
file_list=http://192.168.88.229/tpch_data/1s_tbl/lineite
m.tbl
```

db_user=root

db_name=test

format=3

delimiter='|a'

两个版本的差异是需要将 V8.5.1.2 版加载控制文件的 file_list 参数中的绝对路径,如上例中的/opt,替换为 V8.5.1.2版加载控制文件的file_list 中的数据文件服务器的绝对或者相对路径,如上例中的 http://192.168.88.229 ,具体的替换与数据文件服务器的配置有关,以上只是个示例,请根据现场的具体配置做相应的替换。

2．配置 V8.6.2.X 版本集群加载兼容工具配置文件,实现自动替换

集群加载兼容工具 dispcli 提供了自动替换 V8.5.1.2版本加载控制文件 file_list 参数的功能,可以不需要修改现有的控制文件,实现使用 V8.5.1.2 版本加载方式在 V8.6.2.X 版本集群上进行加载的功能。该功能需要用户在部署加载兼容工具 dispcli 的同级目录创建一个加载兼容工具的配置文件,配置文件的具体要求如下:

<1> 文件名:dispcli_config.conf

<2> 文件类型: 标准 ini 控制文件

section 为 “dispcli”,key_name 为 “server”

[dispcli]

server=192.168.40.2{/opt/tpch_data/:http://192.168.88.229/tpch_data;/mnt/data/:ftp://gbase:gbase@192.168.103.111/data2/},192.168.103.222{/tmp/tpch10s/:http://192.168.103.222/tpch10s/}

value 是以:

ip1{path1:url1;path2:url2},ip2{path1:url1;path2:url2} 的书写形式的字符串

以 ip1{path1:url1;path2:url2}为例

其中:

ip1 是 V8.5.1.2 版本加载控制文件参数

disp_server=192.168.40.2:6666 的ip信息,该ip信息是为了自动替换功能区分V8.5.1.2版本加载控制文件多section不同ip的情况

path1是V8.5.1.2版本加载控制文件中disp_server为ip1的section里file_list参数中的路径信息，如示例中的：/opt/tpch_data/

url1是V8.6.2.X版本集群数据文件服务器,配置的与V8.5.1.2版本加载数据相匹配的url路径信息，如示例中的：http://192.168.88.229/tpch_data

path2,url2以次类推,是该ip的section中file_list里不同加载路径的映射关系。V8.5.1.2版本加载控制文件中不同的disp_server要以ip1{path1:url1;path2:url2}的映射格式配置不同的替换规则。请注意自动替换功能只是对file_list中的路径信息进行文件替换,不对file_list中的替换信息作有效性检查,只有到执行SQL加载的时候才会对替换的有效性做检查。当本功能启用时，如果用户替换后的file_list中包含特殊字符（具体请参考5.3.5小节),dispcli会自动将特殊字符转义，如下例所示:

用户启动自动替换功能后生成的file_list:
'http://192.168.3.241/data/dispcli/new_test/*;、/*;、.tbl

dipcli自动转义后的file_list:
'http://192.168.3.241/data/dispcli/new_test/%2A%3B%E3%80%81/%2A%20%3B%E3%80%81.tbl

dispcli执行时使用自动转义后的file_list进行数据加载，相关日志及报错信息中，也会使用转义后的file_list信息。dispcli和8a集群不再对file_list进行反转义处理。

3．加载兼容工具配置文件参数说明

a. server 该参数用于控制文件列表的自动替换功能，详细信息请参考第二小节。

b. user_name 该参数用于配置日志收集时使用的系统用户名,默认值gbase。

c. user_password 该参数用于配置日志收集时使用的系统用户

密码,无默认值。

d. loader_logs 该参数用于配置日志收集时,收集到的日志的存储路径。

e. keep_loader_logs_on_server 该参数用于控制，收集日志完成后,是否保留加载接入节点汇总的加载日志,有效值:0&1 默认值: 1 表示保留。

4. 关于加载兼容工具使用的加载控制文件中 file_list 的要求,8512 加载 file_list 中是支持换行的。在 86x 加载兼容工具中不再支持 file_list 中有换行符。需要用户注意这点使用限制。

5.5.3 错误数据收集功能

目前 V8.6.2.X 版本集群 SQL 加载方式,不是每次加载任务都进行错误数据汇总,而是提供了外部收集工具,在有需要时进行错误数据的收集。而 V8.5.1.2 版本每次加载任务都会自动汇总所有节点的错误数据至加载服务器上。为了弥补这个差异，集群加载兼容工具开发了错误数据汇总功能。

由于 V8.6.2.X 版本集群已经没有 V8.5.1.2 版本集群加的 dispsever 加载组件,所以错误数据的收集目前只能收到到 dispcli 部署端。

该功能的使用,需要在集群加载兼容工具配置文件中增加两个配置项,具体设置如下:

```
[dispcli]
loader_logs = /opt/loader_logs_test/
user_password=gbase
```

其中

loader_logs 是错误数据的存放路径,需要加载兼容工具有读写权限,如果为多级目录,加载兼容工具只负责最后一级目录的创建。

user_password 是 V8.6.2.X 版本集群数据结点 gbase 用户的密码。

错误数据的收集功能的开启有两个条件:

1 加载兼容工具控制文件中设置了 loader_logs ,且不为空

2 加载结果中有错误数据，即 skipped 条数大于 0

两个条件为必须为真,该功能才会开启。

5.5.4兼容工具 V8.5.1.2 控制文件与 V8.6.2.X SQL 关键字的映射表

V8.5.1.2	V8.6.2.X
file_list	LOAD DATA INFILE
db_name	INTO TABLE
table_name	
format	DATA_FORMAT
delimiter	TERMINATED BY
string_qualifier	ENCLOSED BY
line_separator	LINES TERMINATED BY
null_value	NULL_VALUE
preserve_blanks	PRESERVE BLANKS
table_fields	TABLE_FIELDS
max_error_records	MAX_BAD_RECORDS
field_length_define	DEFINER
def_datetime_format	DATETIME FORMAT
def_date_format	DATE FORMAT
def_time_format	TIME FORMAT
def_timestamp_format	TIMESTAMP FORMAT
auto_fill_column	AUTOFILL

5.5.5兼容工具的兼容参数表

对于 extra_loader_args 参数的处理，特别说明一下。加载兼容工具 dispcli 只处理 extra_loader_args 后面的 def_datetime_format 、def_date_format、def_time_format、def_timestamp_format ，其它后面的参数自动忽略处理，且 extra_loader_args 后面的日间日期类型控制参数的级别

高于写在控制文件中的时间日期类型控制参数的级别，extra_loader_args 后面的多个同名时间日期类型控制参数后面的参数值会覆盖前面的参数值。

命令行参数	支持	忽略	报错
-t, --timeout		✓	
-T, --gbloader-timeout		✓	
-c, --connect-timeout		✓	
-l, --log-file	✓		
-L, --log-level		✓	
-h, --host	✓		
-u, --user	✓		
-p, --password	✓		
-S, --socket		✓	
-P, --port	✓		
-q, --quiet		✓	
-I, --cluster-status-interval		✓	
-v, --verbose		✓	
-V, --version	✓		
-H, --help	✓		
其它			✓
加载控制文件参数	**支持**	**忽略**	**报错**
disp_server		✓	
format	✓		
file_list	✓		
sg_list		✓	
db_name	✓		
table_name	✓		
socket		✓	
extra_loader_args	✓		
hash_parallel		✓	
max_prefetch		✓	
delimiter	✓		
string_qualifier	✓		
line_separator	✓		
null_value	✓		

escape_character		✓	
preserve_blanks	✓		
table_fields	✓		
max_error_records	✓		
using_direct_io		✓	
field_length_define	✓		
send_block_size		✓	
def_datetime_format	✓		
def_date_format	✓		
def_time_format	✓		
def_timestamp_format	✓		
auto_fill_column	✓		
skip_bad_file		✓	
db_user		✓	
rmt_username		✓	
rmt_password		✓	
rmt_ip		✓	
rmt_port		✓	
remote_url		✓	
hdfs_remote		✓	
hdfs_list		✓	
hdfs_match		✓	
log_file_list		✓	
skip_head	✓		
其它			✓

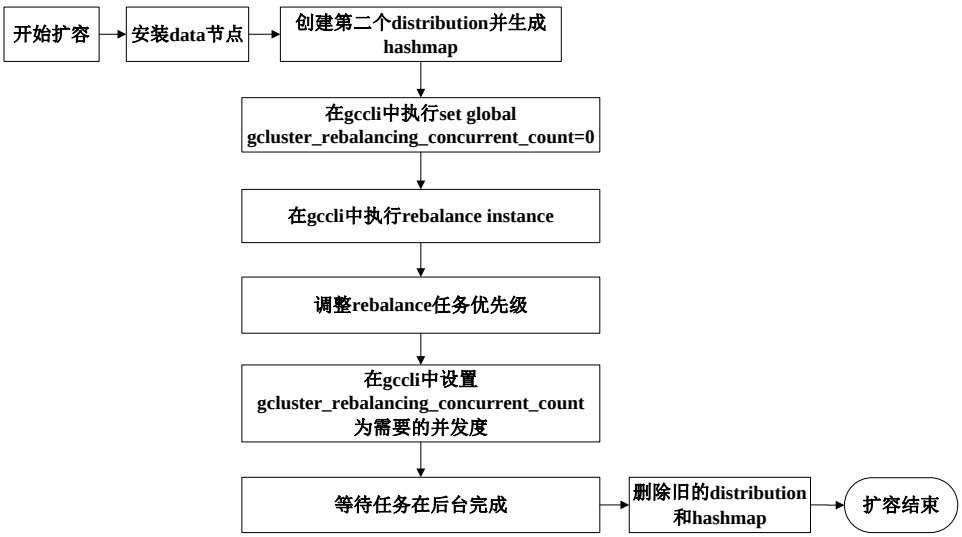
6 集群的扩容、缩容

集群扩容和缩容功能是基于 8611 集群的 distrbiution 和 rebalance 功能实现。一个 8611 集群同时可以创建两个 distribution。安装完集群后需要创建一个 disrtiubiotn 并且生成 hashmap 后集群才可以正常使用。当对集群进行扩容或者缩容时，需要创建第二个 distirbuiotn 和 hashmap，然后再执行 rebalance 命令把集群中的表扩容或者缩容到第二个 distribution 上。

6.1 集群扩容和缩容操作流程

6.1.1 扩容的操作流程

集群扩容的操作流程如下图所示：



创建第二个 distribution 并生成 hashmap 请参考 8.3 小节。

本流程利用 gcluster_rebalancing_concurrent_count 参数先阻止 rebalance 任务被执行。利用 rebalance instance 把当前集群下所有表加入到

gclusterdb.rebalancing_status 中。调整完每个 rebalance 任务的优先级后再设置 gcluster_rebalancing_concurrent_count 为需要的并发数，开始执行扩容。

调整 rebalance 任务优先级见 8.5.13 小节。

删除老的 hashmap 和 distribution 见 8.4 小节。

注意：在扩容 coordinator 节点时，若元数据较多，需指定 timeout 时间以避免拷贝元数据时间超时。执行 gcinstall.py 脚本时增加参数 --timeout=TIMEOUT，timeout 时间单位为分钟，若不指定 timeout 时间，默认超时时间为 15 分钟。

同时，针对带有 License 认证的集群，在扩容新节点时，需要提前生成 License 文件，执行 gcinstall.py 脚本时通过增加--license_file 参数将文件传入，在集群安装成功后，方可正常使用。

6.2安装和卸载 data 节点

扩容集群需要安装 data 节点，缩容集群需要卸载 data 节点。具体安装卸载集群 data 节点的命令请参考《GBase 8a MPP Cluster 安装手册》。

注：执行扩容安装必须在已有集群节点上执行。

若扩容前若升级前集群在 IPV6 网络环境下，则扩容时需设置 coordinateHostNodeID 与原集群不一致，IP 类型与原集群一致。

6.3创建 distribution 和 hashmap

增加集群 data 节点后，安装脚本会在安装包根目录下生成 gcChangeInfo.xml 文件。gcChangeInfo.xml 文件格式如下：

```
<?xml version="1.0" encoding="utf-8"?>
<servers>
```

```
    <rack>
     <node ip="172.16.83.13"/>
     <node ip="172.16.83.12"/>
     <node ip="172.16.83.11"/>
    </rack>
   </servers>
```

在生成 distriubtion 之前，需要修改 gcChangeInfo.xml 文件，把同一个机架的机器 IP 放到一组<rack>标签内。如果集群不需要考虑机架问题，则不用修改。

执行 gcadmin distribution 命令创建第二个 distribution，并在 gccli 中执行 initnodedatamap 生成 hashmap。具体命令说明请参考 gcadmin 章节。

如果是删除集群 data 节点，那么需要手动编写 gcChangeInfo.xml 文件，把删除集群 data 节点后集群剩余的 data 节点所在机器 IP 写入 gcChangeInfo.xml 文件。然后再执行 gcadmin distribution 命令创建第二个 distribution，并在 gccli 中执行 initnodedatamap 生成 hashmap。

使用 gcadmin distribution 命令创建 distribution 如下所示：

```
gcadmin distribution gcChangeInfo.xml p 2 d 1 pattern 1
```

6.4删除 hashmap 和 distribution

先在 gccli 中执行 refreshnodedatamap drop 1，把 id 为 1 的 hashmap 删除。如果有用户表正在使用 hashmap 1，那么 refreshnodedatamap drop 1 将报错。必须等所有的 express 表都已经 rebalance 到新的 distribution 时才可以删除老的 hashmap 和 distribution。通过查看 gbase.table_distribution 表可以得到有哪些表正在使用 hashmap。

只有 hashmap 1 删除成功，才可以删除 id 为 1 的 distribution。删除 distribution 的命令如下所示：

```
gcadmin rmdistribution 1
```

6.5rebalance 命令

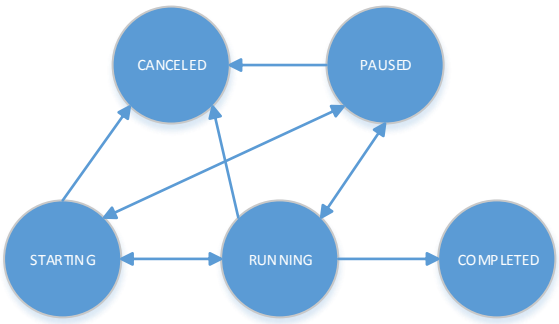
8611 集群增加了一组 SQL 命令(rebalance)把表从一个 distribution 转换到另一个 distribution。rebalance 命令是异步执行命令。在 coordinator 上使用 gccli 执行 rebalance 命令后，rebalance 任务会被加入到 gclusterdb.rebalancing_status 集群表中。coordinator 集群会从 gclusterdb.rebalancing_status 中选取优先级最高的 gcluster_rebalancing_concurrent_count 个表进行 rebalance。coordinator 集群中只会有一个 coordinator 节点负责后台执行表 rebalance。

rebalance 任务执行状态需要从 gclusterdb.rebalancing_status 表中查询。gclusterdb.rebalancing_status 的表结构如下图所示。不建议对 gclusterdb.rebalancing_status 表做 ddl/dml 操作。

字段名	字段类型	字段含义
index_name	varchar(129)	数据库名.表名
db_name	varchar(64)	数据库名
table_name	varchar(64)	表名
tmptable	varchar(129)	rebalance 任务执行时使用的中间表名称。如果不使用中间表，那么该字段值为 NULL。
start_time	datetime	在 STARTING 状态时，表示任务加入的时间。在 RUNNING 状态时，start_time 表示任务更改为 RUNNING 的时间。
end_time	datetime	表示 rebalance 任务结束的时间。
status	varchar(32)	表示 rebalance 任务的当前状态。
percentage	int(11)	表示 rebalance 任务的执行进度。
priority	int(11)	表示 rebalance 任务的优先级。值最小的优先执行。
host	varchar(32)	表示 rebalance 任务被哪个 coordinator 节点执行。
distribution_id	bigint(8)	表示 rebalance 任务要把表 rebalance 到哪个 distribution id。

一个表在 rebalance 时有 5 个状态，分别是：STARTING、RUNNING、

COMPLETED、PAUSED、CANCELED。这 5 种状态转换如下图所示：



表处于 STARTING、RUNNING 和 PAUSED 状态时，对表执行 cancel rebalance 操作，表状态转换成 CANCELED。

表处于 STARTING 和 RUNNING 状态时，对表执行 pause rebalance 操作，表状态转换成 PAUSED。

表处于 STARTING 状态时，coordinator 后台线程开始执行表的 rebalance 操作，表状态转换成 RUNNING。

表处于 PAUSED 状态时，对表执行 continue rebalance 操作，表状态转换成 RUNNING。

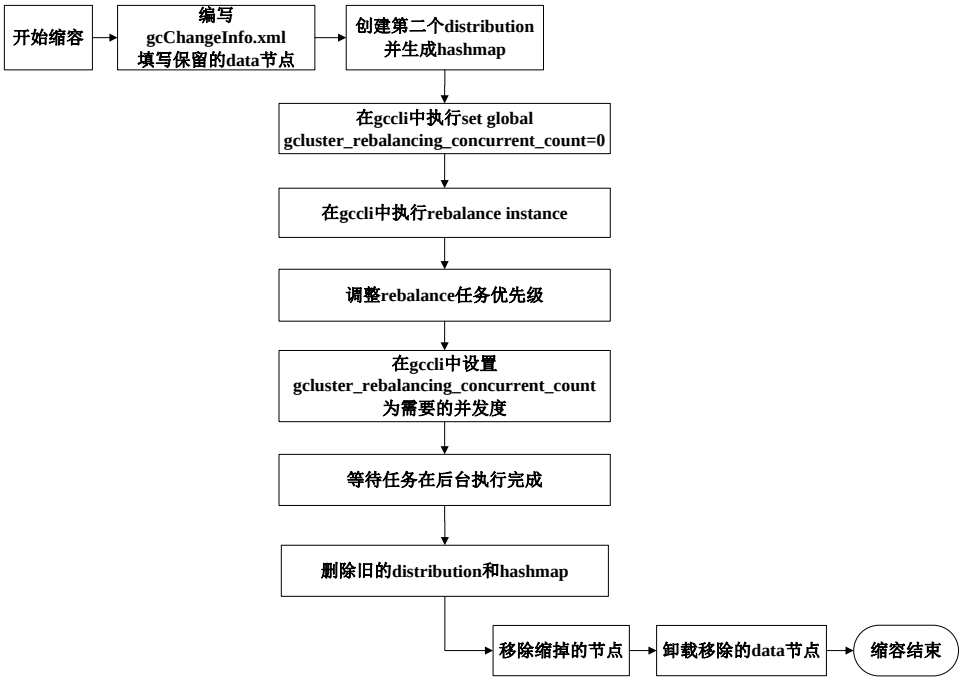
表处于 RUNNING 状态时，coordinator 后台线程执行表的 rebalance 操作失败，表状态转换成 STARTING。

表处于 RUNNING 状态时，coordinator 后台线程完成了表的 rebalance 操作，表状态转换成 COMPLETED。

注：当前版本只支持 express 引擎表的 rebalance，不支持 GsSYS 引擎表 rebalance。

6.5.1 缩容的操作流程

集群缩容的操作流程如下图所示。



移除缩掉的节点前必须删除老的 hashmap 和 distribution。否则无法移除缩掉的节点。移除节点使用如下命令,其中 gcChangeInfo.xml 为开始缩容时编写的文件。

```
gcadmin rmnodes gcChangeInfo.xml
```

卸载移除的节点需要使用集群安装包中的卸载命令。需要注意，使用卸载命令移除节点时不要使用--Force 参数，这会导致卸载脚本不检查将要卸载的节点是否正在被集群使用。

6.5.2rebalance table

语法： rebalance table [database_name.]table_name [to distribution_id]

对 database_name.table_name 做 rebalance 操作。如果不指定[to distribution_id]，那么 rebalance 操作会把 database_name.table_name 转换

到新的 distributon 上。如果 database_name.table_name 已经是分布在新的 distribution 上，那么 rebalance table 操作报错，不会向 gclusterdb.rebalancing_status 中增加 rebalance 任务。如果指定[to distribution_id]，那么 rebalance 操作会把 database_name.table_name 转换到指定的 distribution_id 上。如果 database_name.table_name 已经是分布在指定的 distribution_id 上，那么 rebalance table 操作报错，不会向 gclusterdb.rebalancing_status 中增加 rebalance 任务。

```
gbase> rebalance table testdis;
Query OK, 1 row affected
gbase> rebalance table testdis to 1;
Query OK, 1 row affected
```

pause rebalance table

语法： pause rebalance table [database_name.]table_name

如果 table_name 处于 STARTING 或者 RUNNING 状态，可以使用 pause rebalance table 命令暂停 table_name 的 rebalance 操作。如果 pause rebalance table 命令返回影响行数为 1，则 table_name 暂停成功；如果返回影响行数为 0，则 table_name 暂停不生效。

```
    gbase> select index_name, status, percentage  from
gclusterdb.rebalancing_status;
    +-------------+--------+------------+
    | index_name    | status | percentage |
    +-------------+--------+------------+
    | test.testdis | RUNNING| 10         |
    +-------------+--------+------------+
    1 row in set
    gbase> pause rebalance table testdis ;
    Query OK, 1 row affected
    gbase> select index_name, status, percentage  from
gclusterdb.rebalancing_status;
    +-------------+--------+------------+
    | index_name    | status | percentage |
    +-------------+--------+------------+
    | test.testdis | PAUSED | 10         |
    +-------------+--------+------------+
    1 row in set
```

6.5.3 continue rebalance table

语法： continue rebalance table [database_name.]table_name

如果 table_name 处于 PAUSED 状态，可以使用 continue rebalance table 命令使 table_name 继续 rebalance。

```
gbase> select index_name, status, percentage  from
gclusterdb.rebalancing_status;
+--------------+--------+------------+
| index_name   | status | percentage |
+--------------+--------+------------+
| test.testdis | PAUSED | 10         |
+--------------+--------+------------+
1 row in set

gbase> continue rebalance table test.testdis;
Query OK, 1 row affected

gbase> select index_name, status, percentage  from
gclusterdb.rebalancing_status;
+--------------+---------+------------+
| index_name   | status  | percentage |
+--------------+---------+------------+
| test.testdis | RUNNING | 10         |
+--------------+---------+------------+
1 row in set
```

6.5.4cancel rebalance table

语法： cancel rebalance table [database_name.]table_name

如果 table_name 处于 STARTING、RUNNING、PAUSED 状态，可以使用 cancel rebalance table 命令取消 table_name 的 rebalance 操作。如果 cancel rebalance table 命令返回影响行数为 1,则 table_name 的 rebalance 操作取消成功；如果返回影响行数为 0,则取消操作不生效。

```
    gbase> select index_name, status, percentage  from
gclusterdb.rebalancing_status;
    +-------------+--------+------------+
    | index_name   | status | percentage |
    +-------------+--------+------------+
    | test.testdis | PAUSED | 10         |
    +-------------+--------+------------+
    1 row in set

    gbase> cancel rebalance table testdis;
    Query OK, 1 row affected

    gbase> select index_name, status, percentage  from
gclusterdb.rebalancing_status;
    +-------------+---------+------------+
    | index_name   | status   | percentage |
    +-------------+---------+------------+
    | test.testdis | CANCELED | 0          |
    +-------------+---------+------------+
    1 row in set
```

6.5.5rebalance database

语法： rebalance database database_name [to distribution_id]

rebalance database 是 rebalance table 的批量操作形式。如果不指定[to distribution_id]，rebalance database 把 database_name 下所有分布在老的 distribution 下的表转换成新的 distribution 表。如果指定了[to distribution_id]，rebalance database 把 database_name 下所有不属于 distribution_id 的表转换成 distribution_id 表。rebalance database 返回加入 gclusterdb.rebalancing_status 的 rebalance 任务数。

```
gbase> rebalance database test;
Query OK, 3 rows affected
gbase> select index_name, status, percentage  from
gclusterdb.rebalancing_status;
+--------------+---------+-----------+
| index_name   | status  | percentage |
+--------------+---------+-----------+
| test.testrep | RUNNING | 0          |
| test.testdis | STARTING | 0          |
| test.testrand | STARTING | 0          |
+--------------+---------+-----------+
3 row in set
gbase> select index_name, status, percentage  from
gclusterdb.rebalancing_status;
+--------------+----------+------------+
| index_name   | status   | percentage |
+--------------+----------+------------+
| test.testrep | COMPLETED | 100        |
| test.testdis | RUNNING  | 10         |
| test.testrand | RUNNING  | 90         |
+--------------+----------+------------+
3 row in set
```

6.5.6pause rebalance database

语法： pause rebalance database database_name

pause rebalance database 暂停 database 库所有处于 STARTING 或者 RUNNING 状态表的 rebalance 操作。pause rebalance database 返回的影响行数是暂停下来的 rebalance 任务数。

```
    gbase> select index_name, status, percentage  from
gclusterdb.rebalancing_status;
    +--------------+----------+-----------+
    | index_name   | status   | percentage |
    +--------------+----------+-----------+
    | test.testrand | COMPLETED | 100       |
    | test.testdis  | RUNNING   | 10        |
    | test.testrep  | RUNNING   | 90        |
    +--------------+----------+-----------+
    3 rows in set

    gbase> pause rebalance database test;
    Query OK, 1 row affected

    gbase> select index_name, status, percentage  from
gclusterdb.rebalancing_status;
    +--------------+----------+-----------+
    | index_name   | status   | percentage |
    +--------------+----------+-----------+
    | test.testrand | COMPLETED | 100       |
    | test.testdis  | PAUSED    | 30        |
    | test.testrep  | COMPLETED | 100       |
    +--------------+----------+-----------+
    3 rows in set
```

6.5.7 continue rebalance database

语法： continue rebalance database database_name

continue rebalance database 使 database_name 下所有处于 PAUSED 状态的表继续进行 rebalance。

```
    gbase> select index_name, status, percentage  from
gclusterdb.rebalancing_status;
    +--------------+----------+------------+
    | index_name   | status   | percentage |
    +--------------+----------+------------+
    | test.testrep | PAUSED   | 0          |
    | test.testrand| PAUSED   | 10         |
    | test.testdis | PAUSED   | 10         |
    +--------------+----------+------------+
    3 rows in set

    gbase> continue rebalance database test;
    Query OK, 3 row affected

    gbase> select index_name, status, percentage  from
gclusterdb.rebalancing_status;
    +--------------+----------+------------+
    | index_name   | status   | percentage |
    +--------------+----------+------------+
    | test.testrep | STARTING |  0         |
    | test.testrand| RUNNING  | 10         |
    | test.testdis | RUNNING  | 20         |
    +--------------+----------+------------+
    3 rows in set
```

6.5.8 cancel rebalance database

语法： cancel rebalance database database_name

cancel rebalance database 取消 database_name 下所有处于 STARTING、RUNNING、PAUSED 状态表的 rebalance 操作。cancel rebalance database 命令返回的影响行数是取消掉的 rebalance 任务数。

```
    gbase> select index_name, status, percentage  from
gclusterdb.rebalancing_status;
    +--------------+---------+------------+
    | index_name    | status   | percentage |
    +--------------+---------+------------+
    | test.testdis  | RUNNING | 10         |
    | test.testrep  | RUNNING | 90         |
    | test.testrand | STARTING| 0          |
    +--------------+---------+------------+
    3 rows in set

    gbase> cancel rebalance database test ;
    Query OK, 3 row affected

    gbase> select index_name, status, percentage  from
gclusterdb.rebalancing_status;
    +--------------+-----------+------------+
    | index_name    | status    | percentage |
    +--------------+-----------+------------+
    | test.testdis  | CANCELED  | 0          |
    | test.testrep  | COMPLETED | 100        |
    | test.testrand | CANCELED  | 0          |
    +--------------+-----------+------------+
    3 rows in set
```

6.5.9rebalance instance

语法： rebalance instance [to distribution_id]

rebalance instance 是 rebalance table 的批量操作形式。如果不指定[to distribution_id]，rebalance instance 把当前集群下所有分布在老的 distribution 下的表转换成新的 distribution 表。如果指定了[to distribution_id]，rebalance instance 把当前集群下所有不属于 distribution_id 的表转换成 distribution_id 表。rebalance instance 返回

加入 gclusterdb.rebalancing_status 的 rebalance 任务数。

```
gbase> rebalance instance;
Query OK, 6 rows affected

gbase> select index_name, status, percentage  from
gclusterdb.rebalancing_status;
+---------------+----------+------------+
| index_name    | status   | percentage |
+---------------+----------+------------+
| test1.t1      | RUNNING  | 0          |
| test.testdis  | STARTING | 0          |
| test.testrand | STARTING | 0          |
| test1.t3      | STARTING | 0          |
| test1.t2      | STARTING | 0          |
| test.testrep  | STARTING | 0          |
+---------------+----------+------------+
6 rows in set
```

6.5.10 pause rebalance instance

语法： pause rebalance instance

pause rebalance instance 暂停当前集群下所有处于 STARTING 或者 RUNNING 状态表的 rebalance 操作。pause rebalance instance 返回的影响行数是暂停下来的 rebalance 任务数。

```
gbase> pause rebalance instance;
Query OK, 4 rows affected

gbase> select index_name, status, percentage  from
gclusterdb.rebalancing_status;
+--------------+-----------+------------+
| index_name   | status    | percentage |
+--------------+-----------+------------+
| test1.t1     | PAUSED    | 10         |
| test.testdis | PAUSED    | 10         |
| test.testrand | COMPLETED | 100        |
| test1.t3     | PAUSED    | 10         |
| test1.t2     | PAUSED    | 10         |
| test.testrep | COMPLETED | 100        |
+--------------+-----------+------------+
6 rows in set
```

6.5.11 continue rebalance instance

语法： continue rebalance instance

continue rebalance instance 使当前集群下所有处于 PAUSED 状态的表继续进行 rebalance。

```
gbase> continue rebalance instance ;
Query OK, 4 rows affected

gbase> select index_name, status, percentage  from
gclusterdb.rebalancing_status;
+---------------+-----------+------------+
| index_name    | status    | percentage |
+---------------+-----------+------------+
| test.testrep  | COMPLETED | 100        |
| test.testdis  | RUNNING   | 10         |
| test.testrand | COMPLETED | 100        |
| test1.t1      | RUNNING   | 10         |
| test1.t3      | RUNNING   | 10         |
| test1.t2      | RUNNING   | 10         |
+---------------+-----------+------------+
6 rows in set
```

6.5.12 cancel rebalance instance

语法： cancel rebalance instance

cancel rebalance instance 取消当前集群下所有处于 STARTING、RUNNING、PAUSED 状态表的 rebalance 操作。cancel rebalance instance 命令返回的影响行数是取消掉的 rebalance 任务数。

```
gbase> cancel rebalance instance ;
Query OK, 4 rows affected

gbase> select index_name, status, percentage  from
gclusterdb.rebalancing_status;
+--------------+----------+-----------+
| index_name   | status   | percentage |
+--------------+----------+-----------+
| test.testrep | COMPLETED | 100       |
| test.testdis | CANCELED  | 0         |
| test.testrand | COMPLETED | 100      |
| test1.t1     | CANCELED  | 0         |
| test1.t3     | CANCELED  | 0         |
| test1.t2     | CANCELED  | 0         |
+--------------+----------+-----------+
6 rows in set
```

6.5.13 调整 rebalance 任务优先级

gclusterdb.rebalancing_status 表中记录了当前集群的 rebalance 任务。任务以表为单位。从 gclusterdb.rebalancing_status 表中除了可以查询 rebalance 任务的执行状态外，还可以调整单个 rebalance 任务的优先级。priority 值最小的任务先做。如下示例，先设置 gcluster_rebalancing_concurrent_count 值为 0，然后执行 rebalance database 命令增加 rebalance 任务。此时所有 rebalance 任务都不会开始，可以调整 rebalance 任务的优先级。调整完之后再设置 gcluster_rebalancing_concurrent_count 为需要的并发数，如下所示。

```
gbase> set global gcluster_rebalancing_concurrent_count = 0;
Query OK, 0 rows affected
gbase> rebalance database test;
Query OK, 3 rows affected
gbase> select index_name, status, priority  from
 gclusterdb.rebalancing_status;
```

```
+------------+---------+----------+
| index_name | status  | priority |
+------------+---------+----------+
| test.t3    | STARTING |        5 |
| test.t1    | STARTING |        5 |
| test.t2    | STARTING |        5 |
+------------+---------+----------+
3 rows in set
gbase> update gclusterdb.rebalancing_status set priority = 6 where
index_name='test.t3';
Query OK, 1 row affected
Rows matched: 1  Changed: 1  Warnings: 0

gbase> update gclusterdb.rebalancing_status set priority = 4 where
index_name='test.t2';
Query OK, 1 row affected
Rows matched: 1  Changed: 1  Warnings: 0
gbase> select index_name, status, priority  from
  gclusterdb.rebalancing_status;
+------------+---------+----------+
| index_name | status  | priority |
+------------+---------+----------+
| test.t3    | STARTING |        6 |
| test.t1    | STARTING |        5 |
| test.t2    | STARTING |        4 |
+------------+---------+----------+
3 rows in set
gbase> set global gcluster_rebalancing_concurrent_count = 1;
Query OK, 0 rows affected
gbase> select index_name, status, priority  from
gclusterdb.rebalancing_status;
+------------+---------+----------+
| index_name | status  | priority |
+------------+---------+----------+
| test.t3    | STARTING |        6 |
| test.t1    | STARTING |        5 |
| test.t2    | RUNNING  |        4 |
```

```
+-----------+----------+----------+
3 rows in set
```

6.6rebalance 命令相关参数

gcluster_rebalancing_concurrent_count

含义：允许的并发执行 rebalance 的表的个数

GLOBAL 参数： Y

SESSION 参数：N

默认值：5

最小值：0

最大值：无

gcluster_rebalancing_random_table_quick_mode

含义：对随机分布表执行 rebalance 操作时使用快速模式

GLOBAL 参数： Y

SESSION 参数：N

默认值：1

最小值：0

最大值：1

gcluster_rebalancing_step

含义：指定 rebalance 操作时每一批重分布数据条数。值为 0 时，rebalance 操作不分批。

gcluster_rebalancing_step 参数值事实上是原表的每个分片每一批向中间表重分布的数据行数。gcluster_rebalancing_step 值越大，从原表向中间表重分布数据的速度越快。gcluster_rebalancing_step 值越大，rebalance 过程中暂停时等待的时间上就越长。

如果 rebalance 过程中基本不需要暂停任务，那么可以设置 gcluster_rebalancing_step 为较大的值。如果 rebalance 过程中需要多次暂停任务，那么可以设置 gcluster_rebalancing_step 为较小值。

gcluster_rebalancing_step 预期方法:原表单个分片的行数 / 预计分批数。

GLOBAL 参数： Y

SESSION 参数: N

默认值: 10000000

最小值: 0

最大值: 无

7 集群节点替换

7.1简介

当集群规模不断扩大时，集群的节点损坏会成为一个比较常见的现象，同时随着数据量的增大，单个节点的计算能力和存储能力也会成为瓶颈。这两种情况下都需要对集群节点进行替换升级，使集群能够正常工作。

7.2功能概述

节点替换针对节点机器彻底损坏的情况，通过节点替换流程来恢复数据，最终达到替换损坏节点的目的。新旧机器使用相同的 IP。

节点替换过程中，集群状态必须是非 LOCK 状态。

在节点替换过程中，替换节点的状态会发生变化。

节点故障状态变更顺序为：

OFFLINE -> UNAVAILABLE -> REPLACE -> ONLINE

升级时状态变化过程如下：

ONLINE -> UNAVAILABLE -> REPLACE -> ONLINE

- 替换开始前：节点损坏，节点的状态为 OFFLINE,用户必须设置节点状态为 UNAVAILABLE;
- 替换开始后：节点状态转换为 REPLACE
- 替换成功后：节点状态转换为 ONLINE
- 替换执行失败：节点状态回滚为 UNAVAILABLE

在节点替换过程中，集群模式会发生变化，模式变更顺序为：

NORMAL -> READONLY -> NORMAL

- 同步表结构前：集群模式为 NORMAL,集群可以正常操作
- 同步表结构期间：集群模式为 READONLY,只允许进行查询操作，不允许任何 dml、ddl 以及加载操作，此时进行数据表结构的同步操作，对用户数据表设置全同步标志。
- 同步完成后：集群模式为 NORMAL,集群可以正常操作

替换完成后，集群可以正常进行操作。

7.3注意事项

节点替换过程需要注意以下问题：

- 节点状态转换为 UNAVAILABLE 后，只有在节点替换成功的时候，节点状态才能转换为 ONLINE。用户不能对 corosync 中的节点状态持久化文件进行手工操作，如果手工修改持久化文件中的节点状态（UNAVAILABLE -> ONLINE),会导致数据丢失。
- 设置节点为 UNAVAILABLE 时，若集群中有大量 ddl event，dml event 或 dmlstorage event 时，程序需要检查所有的 event，判断被设置状态的节点的备份节点是否正常，有大量 event 时，该过程可能需要较长时间。
- 开始节点替换时，若被替换节点有大量 ddl event，dml event 或 dmlstorage event 时，节点替换程序要将被替换节点的所有 event 删除，该过程可能会需要较长时间。
- 损坏的节点上有 nocopy 表，数据无法恢复，必须用户手工进行恢复。
- 损坏的节点上存在不通过集群命令创建的数据，这些数据无法恢复，必须用户手工进行恢复。
- 在节点替换过程中，如果出现 replace.py 命令被强杀或者执行 replace.py 命令的机器掉电等现象，可能会导致集群状态处于

READONLY 状态无法自行恢复正常。此时可以使用 gcadmin switchmode normal 恢复集群状态然后继续使用集群。或者再一次执行 replace.py 命令替换节点，节点替换成功后集群状态就恢复正常。

- 节点替换时，需保证 gbase 用户拥有 GCWare 配置文件（/etc/corosync/corosync.conf）的访问权限，即该文件的权限属性应为 644。
- 用户在执行节点替换前,必须保证执行替换的用户拥有安装程序目录的读写权限，即可以在该目录下创建文件和子目录。
- 执行节点替换时，若出现宕机或掉电等情况，将导致替换失败，替换节点和源节点上可能会有残留数据。此时需要再次执行节点替换，进行残留数据清理，完成清理后方可重新进行节点替换操作。

7.4 节点替换的步骤

7.4.1 步骤一：检查网络

在进行节点替换操作前，首先要保证集群内的网络通畅。如果遇到网络异常的情况，可以请网络管理员协助排查异常情况，恢复网络通畅。

7.4.2 步骤二：gcadmin 设置要替换节点的状态

我们在操作系统的 gbase 用户（demo.options 文件中 dbauser 参数指定的用户）下运行 gcadmin setnodestate 命令设置要替换的节点状态为 UNAVAILABLE。

注：gcadmin 设置要替换节点的状态，现在只支持单个节点设置。

7.4.3 步骤三：准备用于节点替换的新机器

新机器安装和原节点机器相同的操作系统，使用相同的 ip，并符合集群安装的要求：关闭防火墙、关闭 selinux。

7.4.4 步骤四：gcadmin 查看集群各项状态

选择集群中一个 coordinator 节点，切换到 gbase 用户（demo.options 文件中 dbauser 参数指定的用户），使用 gcadmin 命令来查看集群的各项状态。

集群的状态为如下时，表示正常：

```
CLUSTER STATE:  ACTIVED
CLUSTER MODE:   NORMAL
============================================================
|             GBASE COORDINATOR CLUSTER INFORMATION            |
============================================================
|   NodeName   |     IpAddress     |gcware |gcluster |DataState |
------------------------------------------------------------
| coordinator1 | 192.168.153.126 | UNAVAILABLE  |      |          |
============================================================
|              GBASE DATA CLUSTER INFORMATION                  |
============================================================
|NodeName |     IpAddress     |gnode |syncserver |DataState |
------------------------------------------------------------
|  node1   | 192.168.153.126 | UNAVAILABLE  |      |          |
```

如果 CLUSTER STATE: LOCKED：节点替换的后续步骤不能进行，必须解决问题使集群的状态变为 ACTIVED，才可以继续。

7.4.5 步骤五：replace.py 对节点进行替换安装

选择集群中一个 coordinator 节点，切换到 gbase 用户（demo.options 文件中 dbauser 参数指定的用户），使用 replace.py 命令对要替换的节点进行替换安装。

命令格式:

replace.py [options]

参数说明:

--host 指定将要替换的节点 ip 列表，用逗号分隔

--rootPwd root 用户的密码， 要求所有节点 root 密码一致。

--root_pwd_file 该参数支持 root 用户在多节点不同密码方式，与参数 rootPwd 不能同时使用，否则报错。

--dbaRootPwd 要替换节点的数据库 root 用户密码如果不为空，需要传入该数据库 root 用户密码。目前密码中不支持单引号，其它特殊符号用单引号包围。

--overwrite 强制覆盖安装,这个参数可选。

--sync_coordi_metadata_timeout 设置 coordinator 节点替换的超时时间。默认值为 15，最小值为 1。

--parallel_pack coordinator 节点替换并行打包开关。默认值为 0，最小值为 1。

--retry_times 节点替换时同步系统表 retry 次数。默认值为 3，最小值为 1，最大值为 2147483647。

--license_file 针对带有 License 认证的集群,在执行节点替换的时候，需要提前生成 license 文件，通过--license_file 参数将文件传入，在集群安装成功后，集群可正常使用。

--passwordInputMode，可选，用于指定密码获取的方式，通过不同的参数实现不同的获取方式。若指定该参数，则 demo.options 中的密码不必再修改。

默认值： file

取值范围：[file,pwdsame,pwddiff]

1) file：表示从文件获取，和原有的方式一致，该方式下，文件中的密码是明文的；

2) pwdsame:表示从终端由用户输入密码，并且所有节点的密码一致情况下使用该参数，对于不同用户密码只输入一次，适用于所有节点都用这个密码；

3) pwddiff： 表示从终端由用户输入密码，并且节点间的密码不一致情况下使用该参数，对于不同用户密码每个节点分别输入一次，适用于不同节点使用不同的密码；

运行前提：

替换节点使用原有节点的 IP。

当用户没有使用强制安装选项，替换节点上不能存在集群软件。删除集群软件时必须先停止相应节点的集群服务。集群服务包括：

1. coordinator 节点：corosync、gclusterd、gcrecover、gcmonit、gcmmonit；
2. dataserver 节点：gbased、gc_sync_server；

用户使用强制安装选项时，用户必须保证替换节点上的有用的信息已经妥善备份，否则可能导致数据无法恢复。

在已有的 coordinator 节点上使用 dba 用户执行。

如果替换 coordinator 集群节点，要求总数不可以超过 coordinator 集群总数的一半，也就是保证集群不能是 LOCK 状态。

如果是 data 集群，不可以同时替换主备分片的节点，保证分片必须有可用的备份源数据。

8 客户端使用 SSL 加密连接到集群

8.1生成 ssl 连接证书

加密功能要求系统中安装 openssl 库，能够执行 openssl 命令

在集群 server 端系统中，根据需要选择生成 ssl 密钥的目录，以路径 /usr/local/ssl 为例

首先生成 server 端的密钥和证书:

进入目录

```
$ cd /usr/local/ssl
```

生成 ca-key.pem

```
$ openssl genrsa 2048 > ca-key.pem
Generating RSA private key, 2048 bit long modulus
.....................................................................
.....................+++
.....................................................................
.....................................................+++
e is 65537 (0x10001)
```

生成 ca-cert.pem，需要填写 Country Name 等信息，可以按照下面方式填写，也可以依据用户实际情况填写

```
$ openssl req -sha1 -new -x509 -nodes -days 3650 -key ca-key.pem > ca-cert.pem
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:11
```

```
    State or Province Name (full name) []:1
    Locality Name (eg, city) [Default City]:1
    Organization Name (eg, company) [Default Company Ltd]:1
    Organizational Unit Name (eg, section) []:1
    Common Name (eg, your name or your server's hostname) []:1
    Email Address []:1
```

生成密钥，同样填写一些信息，password 部分（A challenge password []: ）建议填写复杂一些的密码

```
    $ openssl req -sha1 -newkey rsa:2048 -days 730 -nodes -keyout server-key.pem > server-req.pem
    Generating a 2048 bit RSA private key
    ..........+++
    ..........................................................................
.....+++
    writing new private key to 'server-key.pem'
    -----
    You are about to be asked to enter information that will be incorporated
    into your certificate request.
    What you are about to enter is what is called a Distinguished Name or a DN.
    There are quite a few fields but you can leave some blank
    For some fields there will be a default value,
    If you enter '.', the field will be left blank.
    -----
    Country Name (2 letter code) [XX]:11
    State or Province Name (full name) []:1
    Locality Name (eg, city) [Default City]:1
    Organization Name (eg, company) [Default Company Ltd]:1
    Organizational Unit Name (eg, section) []:1
    Common Name (eg, your name or your server's hostname) []:1
    Email Address []:1

    Please enter the following 'extra' attributes
    to be sent with your certificate request
    A challenge password []:123456
    An optional company name []:1
```

将 server-key.pem 导出为 RSA 类型

```
$ openssl rsa -in server-key.pem -out server-key.pem
writing RSA key
```

生成 server-cert.pem

```
$ openssl x509 -sha1 -req -in server-req.pem -days 730 -CA ca-cert.pem -CAkey ca-key.pem -set_serial 01 > server-cert.pem
Signature ok
subject=/C=11/ST=1/L=1/O=1/OU=1/CN=1/emailAddress=1
Getting CA Private Key
```

在同一目录下，生成 client 端的密钥和证书:

生成密钥，输入信息与 server 端相同

```
$ openssl req -sha1 -newkey rsa:2048 -days 730 -nodes -keyout client-key.pem > client-req.pem
Generating a 2048 bit RSA private key
.....+++
.....+++
writing new private key to 'client-key.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:11
State or Province Name (full name) []:1
Locality Name (eg, city) [Default City]:1
Organization Name (eg, company) [Default Company Ltd]:1
Organizational Unit Name (eg, section) []:1
```

```
Common Name (eg, your name or your server's hostname) []:1
Email Address []:1

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:123456
An optional company name []:1
```

将 client-key.pem 导出为 RSA 类型

```
$ openssl rsa -in client-key.pem -out client-key.pem
writing RSA key
```

生成 client-cert.pem

```
$ openssl x509 -sha1 -req -in client-req.pem -days 730 -CA ca-cert.pem -CAkey
ca-key.pem -set_serial 01 > client-cert.pem
Signature ok
subject=/C=11/ST=1/L=1/O=1/OU=1/CN=1/emailAddress=1
Getting CA Private Key
```

8.2 server 配置

修改集群配置文件 gbase_8a_gcluster.cnf，在[gbased]里添加 ssl 信息，以路径/usr/local/ssl 为例，如下方红字所示

```
$ vi /opt/gcluster/config/gbase_8a_gcluster.cnf
[client]
port=5258
socket=/tmp/gcluster_5258.sock
connect_timeout=43200
#default-character-set=gbk

[gbased]
basedir = /opt/gcluster/server
datadir = /opt/gcluster/userdata/gcluster
```

```
socket=/tmp/gcluster_5258.sock
pid-file = /opt/gcluster/log/gcluster/gclusterd.pid

#default-character-set=gbk

ssl-ca=/usr/local/ssl/ca-cert.pem
ssl-cert=/usr/local/ssl/server-cert.pem
ssl-key=/usr/local/ssl/server-key.pem

log-error
port=5258
core-file
```

查看配置是否成功

重启集群服务:

```
# service gcware restart
```

登录集群:

```
$ gccli -uroot -p
Enter password:

GBase client 8.6.1.1 build 65304. Copyright (c) 2004-2016, GBase.  All Rights
Reserved.

gbase>
```

查看 ssl 参数状态，配置成功则显示为“YES”:

```
gbase> show variables like 'have_%ssl';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| have_openssl  | YES   |
| have_ssl      | YES   |
```

```
+---------------+-------+
2 rows in set (Elapsed: 00:00:00.00)
```

8.3client 配置

将 server 端生成的 ca-cert.pem, client-req.pem, client-key.pem, client-cert.pem 拷贝到 client 端

修改集群配置文件 gbase_8a_gcluster.cnf, 在[client]里添加 ssl 信息, 以路径/usr/local/ssl 为例, 如下方红字所示

```
$ vi /opt/gcluster/config/gbase_8a_gcluster.cnf
[client]
port=5258
socket=/tmp/gcluster_5258.sock
connect_timeout=43200
#default-character-set=gbk

ssl-ca=/usr/local/ssl/ca-cert.pem
ssl-cert=/usr/local/ssl/client-cert.pem
ssl-key=/usr/local/ssl/client-key.pem

[gbased]
basedir = /opt/gcluster/server
datadir = /opt/gcluster/userdata/gcluster
socket=/tmp/gcluster_5258.sock
pid-file = /opt/gcluster/log/gcluster/gclusterd.pid

#default-character-set=gbk

log-error
port=5258
core-file
```

通过 client 端远程访问 server，比如用 ssluser 用户登录 192.168.134.131 的 server：

```
[gbase@localhost config]$ gccli -h192.168.134.131 -ussluser -p
Enter password:

GBase client 8.6.1.1 build 65111. Copyright (c) 2004-2016, GBase.  All Rights Reserved.

gbase>
```

运行 status 命令，ssl 部分显示有“Cipher in use”，表示 ssl 加密连接成功：

```
gbase> status;
--------------
gccli  Ver 14.14 Distrib 8.6.1.1, for unknown-linux-gnu (x86_64) using readline 6.0

Connection id:      13
Current database:       information_schema
Current user:           ssluser@192.168.134.132
SSL:                    Cipher in use is DHE-RSA-AES256-SHA
Current pager:      stdout
Using outfile:      ''
Using delimiter:        ;
Server version:         8.6.1.1-65304
Protocol version:       10
Connection:             192.168.134.131 via TCP/IP
Server characterset:    utf8
Db      characterset:   utf8
Client characterset:    utf8
Conn.   characterset:   utf8
TCP port:               5258
Uptime:                 Elapsed: 25:24:49.00

Threads: 4  Questions: 40  Slow queries: 2  Opens: 17  Flush tables: 1  Open
```

```
tables: 10  Queries per second avg: 0.0
    --------------
```

如果 client 端没有进行上述配置，则仍然会按默认方式连接 server。

9 集群的审计日志

9.1 简介

审计日志最重要的作用，就是用于监控 SQL 执行性能，当一些 SQL 语句的执行用时大于 long_query_time 设定值的时候，审计日志便将这些 SQL 语句记录下来，方便用户针对这些执行效率低下的 SQL 语句进行分析，优化，改写，从而提高 SQL 语句的执行效率。

审计日志记录的主要内容如下：

- thread_id：线程号，同 processlist 中的 ID；
- taskid：每个 sql 任务编号；
- start_time：开始执行时间；
- end_time：结束执行时间；
- user_host：登录的用户和 IP，显示格式为：priv_user[user]@hostname[ip]；
- user：用户名；
- host_ip：用户登录端 IP 地址；
- query_time：执行语句所用时间；
- rows：行数；
- db：执行语句所针对的数据库；

- table_list: 涉及表，格式: `<db>`.`<tb>`[,...];
- sql_text: 记录执行用时大于 long_query_time 设定值的 SQL 语句;
- sql_type: sql 类型，如 DDL、DML、DQL、OTHERS;
- sql_command: sql 命令类型，如 SELECT、UPDATE, INSERT, LOAD 等;
- operators: 涉及的算子，比如 JOIN、WHERE、GROUP、HAVING 等;
- status: sql 执行状态，如 SUCCESS、FAILED、KILLED 等;
- conn_type: 用户登录方式(CAPI、ODBC、JDBC、ADO.NET、STUDIO);

其中 sql_command 的取值范围为: INSERT、DELETE、UPDATE、LOAD、CREATE USER、CREATE DB、CREATE TABLE、CREATE VIEW、CREATE INDEX、CREATE PROCEDURE、CREATE FUNCTION、RENAME USER、ALTER DB、ALTER TABLE、ALTER PROCEDURE、ALTER FUNCTION、ALTER EVENT、DROP USER、DROP DB、DROP TABLE、DROP VIEW、DROP INDEX、DROP PROCEDURE、DROP FUNCTION、DROP EVENT、TRUNCATE、GRANT、REVOKE、SELECT 和 OTHERS。

operators 的取值范围为: START_WITH、CONNECT_BY、JOIN、WHERE、GROUP、OLAP_GROUP、HAVING、OLAP_FUNC、DISTINCT、ORDER 和 LIMIT。若一个 SQL 涉及多个算子，各算子之间用逗号分割。

目前，用户每次查看的审计日志为登录节点机器上的日志内容。

9.2配置参数

可以使用如下配置方式开启或关闭审计日志:

全局级变量（默认为 0,即关闭审计日志）

```
SET GLOBAL audit_log = 0;
```

或

```
SET GLOBAL audit_log = 1;+
```

审计日志中记录的内容可通过创建审计策略来配置，详情参考 9.3 审计策略章节。

可以使用如下配置方式设定审计日志存储在系统表中：

全局级变量

SET GLOBAL log_output = 'table';

9.3 审计策略

9.3.1 创建审计策略

创建审计策略的语法规则如下所示：

CREATE AUDIT POLICY <audit_policy_name> [(<audit_policy_item> = <value>[,<audit_policy_item> = <value>])];

其中 audit_policy_name 为审计策略的名称，不区分大小写，存储时小写保存，与资源池名类似，可包括大小写字母、数字和下划线，不可以包含特殊符号，数字不可以作为第一个字母；audit_policy_item 为审计策略项目，审计策略项目名称不区分大小写，目前支持的审计策略项目包括：

项目名称	取值&含义
Enable	Y：启用，默认值
	N：禁用
Hosts	“”：不限制，默认值
	<host>：严格匹配 host， 支持空格“ ”分隔的 host 列表，host 可使用“%”和“_”做通配符
User	“”：不限制，默认值
	<user>：严格匹配 user，区分大小写
Db	“”：不限制，默认值

	<db>：严格匹配 db
Obj_type	“”：不限制，默认值
	TABLE(VIEW)：Object 为表（视图）
	PROCEDURE：Object 为存储过程
	FUNCTION：Object 为函数
Object	“”：不限制，默认值
	<object>：匹配 Obj_tpe 指定的 object
Sql_commands	“”：不限制，默认值
	INSERT，DELETE，UPDATE，LOAD，CREATE_USER，CREATE_DB，CREATE_TABLE，CREATE_VIEW，CREATE_INDEX，CREATE_PROCEDURE，CREATE_FUNCTION，RENAME_USER，ALTER_DB，ALTER_TABLE，ALTER_PROCEDURE，ALTER_FUNCTION，ALTER_EVENT，DROP_USER，DROP_DB，DROP_TABLE，DROP_VIEW，DROP_INDEX，DROP_PROCEDURE，DROP_FUNCTION，DROP_EVENT，TRUNCATE，GRANT，REVOKE，SELECT，OTHERS：其中的一种或多种类型，多个类型间以逗号‘,’连接，且不添加空格
Long_query_time	<*secs*>：最小查询秒数，可带 6 位小数，精确到微秒，默认值 0，取值范围为 0~31536000
Status	“”：不限制，默认值
	SUCCESS：执行成功
	FAILED：执行失败

9.3.2修改审计策略

修改审计策略的语法规则如下：

ALTER AUDIT POLICY <audit_policy_name> SET [(]<audit_policy_item> = <value>[,<audit_policy_item> = <value>][)];

其中：

audit_policy_name 为审计策略的名称，不区分大小写；

audit_policy_item 为审计策略项目，取值内容与创建审计策略中描述相同。

9.3.3 删除审计策略

删除审计策略语法规则如下：

DROP AUDIT POLICY <audit_policy_name>;

9.4 存储方式

GBase 8a MPP Cluster 使用如下方式存储这些内容：

审计日志的信息存储在系统表 gbase.audit_log 或 gbased-audit.log 中，具体存储方式依赖于全局级变量 log_output 的配置。

9.5 使用约束

审计日志用于记录所有的 SQL 操作，对于包含结果集行数的统计操作，只涉及以下 4 种 DML 操作：SELECT、DELETE、INSERT 和 UPDATE。

清空 audit_log 时，需要使用 TRUNCATE SELF audit_log 语句，一般情况下，建议每 2-3 个月可以 TRUNCATE 一次 audit_log 表，用于清除陈旧日志信息，避免数据过多，影响日后的分析。

9.6 使用示例

示例 1：使用系统表查看审计日志。

```
$ gccli -uroot -p
Enter password:

GBase client 8.5.1.2 build 26068. Copyright (c) 2004-2013, GBase.  All Rights Reserved.
```

```
gbase> SET long_query_time = 0;
    Query OK, 0 rows affected

gbase> SET GLOBAL audit_log = 1;
    Query OK, 0 rows affected

gbase> DROP USER tzt;
    Query OK, 0 rows affected

gbase> SET GLOBAL log_output = 'table';
    Query OK, 0 rows affected

gbase> DROP DATABASE test;
    Query OK, 1 row affected

gbase> CREATE USER tzt identified by 'tzt';
    Query OK, 0 rows affected

gbase> GRANT ALL ON *.* TO tzt@'%';
    Query OK, 0 rows affected

gbase> CREATE DATABASE test;
    Query OK, 1 row affected

gbase> USE test;
    Query OK, 0 rows affected

gbase> CREATE TABLE t1(i int);
    Query OK, 0 rows affected

gbase> INSERT INTO t1 VALUES (1),(2);
    Query OK, 2 rows affected

gbase> SELECT start_time,user_host,query_time,rows,LEFT(sql_text, 30), conn_type
FROM gbase.audit_log;
    +--------------------+--------------------------------+
```

```
| start_time          | user_host                          |
+---------------------+------------------------------------+
| 2013-10-09 17:21:08 | root[root] @ localhost []          |
| 2013-10-09 17:21:22 | root[root] @  [192.168.10.116]     |
| 2013-10-09 17:21:22 | root[root] @ localhost []          |
| 2013-10-09 17:21:32 | gbase[gbase] @  [192.168.10.116]   |
| 2013-10-09 17:21:32 | root[root] @ localhost []          |
| 2013-10-09 17:21:32 | root[root] @ localhost []          |
| 2013-10-09 17:21:45 | root[root] @ localhost []          |
| 2013-10-09 17:21:52 | root[root] @ localhost []          |
| 2013-10-09 17:21:58 | root[root] @ localhost []          |
| 2013-10-09 17:22:05 | root[root] @ localhost []          |
| 2013-10-09 17:22:10 | gbase[gbase] @  [192.168.10.116]   |
| 2013-10-09 17:22:10 | root[root] @ localhost []          |
| 2013-10-09 17:22:17 | root[root] @ localhost []          |
+---------------------+------------------------------------+


+-----------------+------+------------------------------+----------+
| query_time      | rows | LEFT(sql_text, 30)           | conn_type|
+-----------------+------+------------------------------+----------+
| 00:00:00.006397 |    0 | SET GLOBAL log_output = 'table| CAPI     |
| 00:00:00.000282 |    0 | Connect                      | CAPI     |
| 00:00:00.025018 |    0 | DROP USER tzt                | CAPI     |
| 00:00:00.000054 |    0 | Connect                      | CAPI     |
| 00:00:00.000175 |    0 | DROP DATABASE test           | CAPI     |
| 00:00:00.111946 |    1 | SELECT DATABASE()            | CAPI     |
| 00:00:00.000086 |    0 | CREATE USER tzt identified by | CAPI     |
| 00:00:00.439480 |    0 | GRANT ALL ON *.* TO tzt@'%'  | CAPI     |
| 00:00:00.000387 |    0 | CREATE DATABASE test         | CAPI     |
| 00:00:00.000025 |    0 | USE test                     | CAPI     |
| 00:00:00.000384 |    0 | Connect                      | CAPI     |
| 00:00:00.000144 |    0 | CREATE TABLE t1(i int)       | CAPI     |
| 00:00:00.004527 |    2 | INSERT INTO t1 VALUES (1),(2) | CAPI     |
+-----------------+------+------------------------------+----------+
13 rows in set
```

```
gbase> INSERT INTO t1 SELECT * FROM t1;
```

```
    Query OK, 2 rows affected

gbase> UPDATE t1 SET i = 3;
    Query OK, 4 rows affected

gbase> DELETE FROM t1;
    Query OK, 4 rows affected

gbase> SELECT start_time,user_host,query_time,rows, LEFT(sql_text, 30), conn_type
FROM gbase.audit_log;
    +---------------------+---------------------------------+
    | start_time          | user_host                       |
    +---------------------+---------------------------------+
    | 2013-10-09 17:21:08 | root[root] @ localhost []       |
    | 2013-10-09 17:21:22 | root[root] @  [192.168.10.116]  |
    | 2013-10-09 17:21:22 | root[root] @ localhost []       |
    | 2013-10-09 17:21:32 | gbase[gbase] @  [192.168.10.116] |
    | 2013-10-09 17:21:32 | root[root] @ localhost []       |
    | 2013-10-09 17:21:32 | root[root] @ localhost []       |
    | 2013-10-09 17:21:45 | root[root] @ localhost []       |
    | 2013-10-09 17:21:52 | root[root] @ localhost []       |
    | 2013-10-09 17:21:58 | root[root] @ localhost []       |
    | 2013-10-09 17:22:05 | root[root] @ localhost []       |
    | 2013-10-09 17:22:10 | gbase[gbase] @  [192.168.10.116] |
    | 2013-10-09 17:22:10 | root[root] @ localhost []       |
    | 2013-10-09 17:22:17 | root[root] @ localhost []       |
    | 2013-10-09 17:22:25 | root[root] @ localhost []       |
    | 2013-10-09 17:23:13 | root[root] @ localhost []       |
    | 2013-10-09 17:23:20 | root[root] @ localhost []       |
    | 2013-10-09 17:23:27 | root[root] @ localhost []       |
    +---------------------+---------------------------------+

    +----------------+------+------------------------------+----------+
    | query_time     | rows | LEFT(sql_text, 30)           | conn_type|
    +----------------+------+------------------------------+----------+
    | 00:00:00.006397 |    0 | SET GLOBAL log_output = 'table| CAPI     |
    | 00:00:00.000282 |    0 | Connect                      | CAPI     |
```

```
| 00:00:00.025018 |    0 | DROP USER tzt                  | CAPI     |
| 00:00:00.000054 |    0 | Connect                        | CAPI     |
| 00:00:00.000175 |    0 | DROP DATABASE test             | CAPI     |
| 00:00:00.111946 |    1 | SELECT DATABASE()              | CAPI     |
| 00:00:00.000086 |    0 | CREATE USER tzt identified by | CAPI     |
| 00:00:00.439480 |    0 | GRANT ALL ON *.* TO tzt@'%'    | CAPI     |
| 00:00:00.000387 |    0 | CREATE DATABASE test           | CAPI     |
| 00:00:00.000025 |    0 | USE test                       | CAPI     |
| 00:00:00.000384 |    0 | Connect                        | CAPI     |
| 00:00:00.000144 |    0 | CREATE TABLE t1(i int)         | CAPI     |
| 00:00:00.004527 |    2 | INSERT INTO t1 VALUES (1),(2)  | CAPI     |
| 00:00:00.000035 |   13 | SELECT start_time,user_host,qu | CAPI     |
| 00:00:00.000191 |    2 | INSERT INTO t1 SELECT * FROM t | CAPI     |
| 00:00:00.000060 |    4 | UPDATE t1 SET i = 3            | CAPI     |
| 00:00:00.094043 |    4 | DELETE FROM t1                 | CAPI     |
+-----------------+------+--------------------------------+----------+
17 rows in set
```

9.7审计日志高可用

为实现审计日志的高可用机制，集群安装或升级时自动在 gclusterdb 库下创建 EXPRESS 引擎随机分布表 audit_log_express，并自动创建定时导出事件，将 gbase 库中的 audit_log 表内容定时导出到 gclusterdb 库中的 audit_log_express 表。

audit_log_express 表结构如下：

列名	类型	Null	默认值	含义
hostname	varchar(64)	YES	NULL	主机名
thread_id	int	YES	NULL	会话 ID
taskid	bigint	YES	NULL	任务 ID
start_time	datetime	YES	NULL	SQL 起始时间
uid	bigint	YES	NULL	用户 ID
user	varchar(16)	YES	NULL	用户名
host_ip	varchar(32)	YES	NULL	用户登录端 IP
query_time	time	YES	NULL	SQL 执行时间

rows	bigint	YES	NULL	涉及行数
table_list	varchar(4096)	YES	NULL	涉及表
sql_text	varchar(8192)	YES	NULL	SQL 内容
sql_type	varchar(16)	YES	NULL	SQL 类型
sql_command	varchar(32)	YES	NULL	SQL 命令类型
operators	varchar(256)	YES	NULL	涉及算子
status	varchar(16)	YES	NULL	执行状态
conn_type	varchar(16)	YES	NULL	终端类型

10 EXPLAIN 显示查询计划

GBase 8a MPP Cluster 的 explain 命令显示正确的查询计划，对于 CBO（基于成本的优化）的计划，显示每个步骤的评估结果，包括成本，记录条数，记录宽度，选择度。用户可以在执行 SQL 之前查看计划。

语法：

查看 SELECT 的查询计划：

explain/desc [extended/partitions] select……

查看 CTE 的查询计划：

explain/desc [extended/partitions] with……select

注：EXPLAIN 与 DESC 等价，因此可以互换，用来查看 SELECT 的查询计划。

10.1 缺省输出

explain 后没有 extended 和 partitions 时，显示缺省输出。

缺省输出只输出简化版的查询计划，主要包括数据重分布方式和每个步骤的主要操作。

示例：如下 TPC-DS SQL-5

```
with ssr as
 (select s_store_id,
        sum(sales_price) as sales,
        sum(profit) as profit,
        sum(return_amt) as returns,
        sum(net_loss) as profit_loss
   from (select ss_store_sk as store_sk,
                ss_sold_date_sk as date_sk,
                ss_ext_sales_price as sales_price,
```

```
                ss_net_profit as profit,
                cast(0 as decimal(7, 2)) as return_amt,
                cast(0 as decimal(7, 2)) as net_loss
          from store_sales
         union all
         select sr_store_sk as store_sk,
                sr_returned_date_sk as date_sk,
                cast(0 as decimal(7, 2)) as sales_price,
                cast(0 as decimal(7, 2)) as profit,
                sr_return_amt as return_amt,
                sr_net_loss as net_loss
          from store_returns) salesreturns,
        date_dim,
        store
   where date_sk = d_date_sk
     and d_date between cast('1998-08-04' as date) and
        (cast('1998-08-04' as date) + interval 14 day)
     and store_sk = s_store_sk
   group by s_store_id),
csr as
 (select cp_catalog_page_id,
        sum(sales_price) as sales,
        sum(profit) as profit,
        sum(return_amt) as returns,
        sum(net_loss) as profit_loss
    from (select cs_catalog_page_sk as page_sk,
                cs_sold_date_sk as date_sk,
                cs_ext_sales_price as sales_price,
                cs_net_profit as profit,
                cast(0 as decimal(7, 2)) as return_amt,
                cast(0 as decimal(7, 2)) as net_loss
          from catalog_sales
         union all
         select cr_catalog_page_sk as page_sk,
                cr_returned_date_sk as date_sk,
                cast(0 as decimal(7, 2)) as sales_price,
                cast(0 as decimal(7, 2)) as profit,
```

```
                cr_return_amount as return_amt,
                cr_net_loss as net_loss
           from catalog_returns) salesreturns,
        date_dim,
        catalog_page
   where date_sk = d_date_sk
     and d_date between cast('1998-08-04' as date) and
        (cast('1998-08-04' as date) + interval 14 day)
     and page_sk = cp_catalog_page_sk
   group by cp_catalog_page_id),
wsr as
 (select web_site_id,
        sum(sales_price) as sales,
        sum(profit) as profit,
        sum(return_amt) as returns,
        sum(net_loss) as profit_loss
    from (select ws_web_site_sk as wsr_web_site_sk,
                ws_sold_date_sk as date_sk,
                ws_ext_sales_price as sales_price,
                ws_net_profit as profit,
                cast(0 as decimal(7, 2)) as return_amt,
                cast(0 as decimal(7, 2)) as net_loss
           from web_sales
         union all
         select ws_web_site_sk as wsr_web_site_sk,
                wr_returned_date_sk as date_sk,
                cast(0 as decimal(7, 2)) as sales_price,
                cast(0 as decimal(7, 2)) as profit,
                wr_return_amt as return_amt,
                wr_net_loss as net_loss
           from web_returns
           left outer join web_sales
             on (wr_item_sk = ws_item_sk and
                wr_order_number = ws_order_number))
salesreturns,
        date_dim,
        web_site
```

```
   where date_sk = d_date_sk
     and d_date between cast('1998-08-04' as date) and
         (cast('1998-08-04' as date) + interval 14 day)
     and wsr_web_site_sk = web_site_sk
   group by web_site_id)
select channel,
       id,
       sum(sales) as sales,
       sum(returns) as returns,
       sum(profit) as profit
  from (select 'store channel' as channel,
               'store' || s_store_id as id,
               sales,
               returns,
               (profit - profit_loss) as profit
         from ssr
       union all
       select 'catalog channel' as channel,
              'catalog_page' || cp_catalog_page_id as id,
              sales,
              returns,
              (profit - profit_loss) as profit
         from csr
       union all
       select 'web channel' as channel,
              'web_site' || web_site_id as id,
              sales,
              returns,
              (profit - profit_loss) as profit
         from wsr) x
 group by rollup(channel, id)
 order by channel, id limit 100;
```

计划如下图所示：

```
+----+----------------+-------------+-------------------------+
| ID | MOTION         | OPERATION   | TABLE                   |
+----+----------------+-------------+-------------------------+
| 04 | [RESULT]       |  Step       | <03>                    |
|    |                |  GROUP      |                         |
|    |                |  ORDER      |                         |
|    |                |  LIMIT      |                         |
| 03 | [GATHER]       |  SubQuery2  | x                       |
|    |                |   SubQuery3 | ssr                     |
|    |                |    Step     | <02>                    |
|    |                |    GROUP    |                         |
|    |                |   UNION ALL |                         |
|    |                |   SubQuery7 | csr                     |
|    |                |    Step     | <00>                    |
|    |                |    GROUP    |                         |
|    |                |   UNION ALL |                         |
|    |                |   SubQuery11| wsr                     |
|    |                |    Step     | <01>                    |
|    |                |    GROUP    |                         |
|    |                |  GROUP      |                         |
| 02 | [REDIST(s_sto..|  INNER JOIN  |                         |
|    |                |   INNER JOIN|                         |
|    |                |    SCAN     | date_dim[REP]           |
|    |                |    SubQuery4| salesreturns            |
|    |                |     Table   | store_sales[ss_item_sk] |
|    |                |     UNION ALL|                        |
|    |                |     Table   | store_retu..[sr_item_sk]|
|    |                |   Table     | store[REP]              |
|    |                |  GROUP      |                         |
| 01 | [REDIST(web_s..|  INNER JOIN  |                         |
|    |                |   INNER JOIN|                         |
|    |                |    SCAN     | date_dim[REP]           |
|    |                |    SubQuery12| salesreturns           |
|    |                |     Table   | web_sales[ws_item_sk]   |
|    |                |     UNION ALL|                        |
|    |                |     LEFT JOIN|                        |
|    |                |      Table  | web_returns[wr_item_sk] |
```

```
|    |                 |      Table    | web_sales[ws_item_sk]    |
|    |                 |   Table       | web_site[REP]            |
|    |                 |  GROUP        |                          |
| 00 | [REDIST(cp_ca.. |  INNER JOIN   |                          |
|    |                 |   INNER JOIN  |                          |
|    |                 |    SCAN       | date_dim[REP]            |
|    |                 |    SubQuery8  | salesreturns             |
|    |                 |     Table     | catalog_sa..[cs_item_sk] |
|    |                 |     UNION ALL |                          |
|    |                 |     Table     | catalog_re..[cr_item_sk] |
|    |                 |   Table       | catalog_page[REP]        |
|    |                 |  GROUP        |                          |
+----+-----------------+---------------+--------------------------+
+--------------------------+-------+-------+--------------------+
| CONDITION                | ROW   | WIDTH | COST(TOTAL)        |
+--------------------------+-------+-------+--------------------+
|                          | 2     | 26    | 0.00693147(479021) |
|  GROUP BY  ROLLUP ((cha.. |       |       |                    |
|  ORDER BY channel ASC, .. |       |       |                    |
|  LIMIT 100               |       |       |                    |
|                          | 0     | 26    | 250451(479021)     |
|                          |       |       |                    |
|                          |       |       |                    |
|  GROUP BY s_store_id      |       |       |                    |
|                          |       |       |                    |
|                          |       |       |                    |
|                          |       |       |                    |
|  GROUP BY cp_catalog_pa.. |       |       |                    |
|                          |       |       |                    |
|                          |       |       |                    |
|                          |       |       |                    |
|  GROUP BY web_site_id     |       |       |                    |
|  GROUP BY channel, id     |       |       |                    |
| (store_sk = s_store_sk)  | 18    | 48    | 128146(228570)     |
| (date_sk = d_date_sk)    |       |       |                    |
| (d_date BETWEEN cast('1.. |       |       |                    |
|                          |       |       |                    |
```

```
|                            |        |       |                    |
|                            |        |       |                    |
|                            |        |       |                    |
|                            |        |       |                    |
|  GROUP BY s_store_id       |        |       |                    |
| (wsr_web_site_sk = web_..  | 45     | 48    | 32897.2(100425)    |
| (date_sk = d_date_sk)      |        |       |                    |
| (d_date BETWEEN cast('1..  |        |       |                    |
|                            |        |       |                    |
|                            |        |       |                    |
|                            |        |       |                    |
| (wr_item_sk = ws_item_s..  |        |       |                    |
|                            |        |       |                    |
|                            |        |       |                    |
|                            |        |       |                    |
|  GROUP BY web_site_id      |        |       |                    |
| (page_sk = cp_catalog_p..  | 35154  | 48    | 67527.5(67527.5)   |
| (date_sk = d_date_sk)      |        |       |                    |
| (d_date BETWEEN cast('1..  |        |       |                    |
|                            |        |       |                    |
|                            |        |       |                    |
|                            |        |       |                    |
|                            |        |       |                    |
|                            |        |       |                    |
|  GROUP BY cp_catalog_pa..  |        |       |                    |
+----------------------------+--------+-------+--------------------+
46 rows in set (Elapsed: 00:00:00.30)
```

列	解释
ID	计划的步骤，从 00 开始执行，如上图，最后一步是 04
MOTION	该步骤的执行结果的处理方式： RESULT：结果发送到客户端 GATHER：结果发送到汇总节点 REDIST(...)：结果 HASH 重分布，括号中为计算 HASH 的列，如果超长则截断为两个点 NO REDIST：结果直接保存到对应的数据分片，不进行重分布

	BROADCAST：结果拉复制表 RAND REDIST：结果随机分布到所有节点 SCALAR N：结果为标量，N 为标量子查询的编号，如果条件中有引用，则使用&xNx&方式引用，例如： **gbase> explain select * from x2 where id2 > (select count(*) from x3);** +----+----------+----------+---------+---------------+----+--- ----+------+ | ID | MOTION | OPERATION | TABLE | CONDITION | ROW | WIDTH | COST(TOTAL) | +----+----------+----------+---------+---------------+----+--- ---+----------+ | 02 | [RESULT] | SCAN | x2[id4] | id2{S} > &x1x&) | 16 | 16 | 1.05(3.26) | | 01 | [SCALAR_1] | Step | <00> | | 0 | 0 | 0.52(2.21) | | | | AGG | | | | | | 00 | [GATHER] | Table | x3[id4] | | 0 | 0 | 1.69(1.69) | | | | AGG | | | | | +----+----------+----------+---------+---------------+----+--- ---+----------+ 5 row in set
OPERATION	SCAN：单表扫描，并使用条件过滤数据； Table：单表，没有过滤条件； SubQueryN：子查询，N 为自动编号； Step：使用前一个 Step 的结果； INNER/LEFT/FULL JOIN：连接操作； WHERE：子查询的 WHERE 条件； GROUP：分组操作； ORDER：排序操作； LIMIT：计算 LIMIT，OFFSET； AGG：distinct，聚集操作； UNION/UNION ALL/MINUS/INTERSECT：UNION 操作； 例如图中 00 的含义为： 第一个 INNER JOIN 的左边为第二个 INNER JOIN； 第二个 INNER JOIN 的左边为 date_dim，该表为复制表，有条件过滤； 第二个 INNER JOIN 的右边为 salesreturns，其为子查询 Salesreturns 子查询包含一个 UNION ALL：为 catelog_sa..与 catelog_re..的 UNION ALL；

	第一个 INNER JOIN 的右边为 catalog_page，该表为复制表； 第一个 INNER JOIN 结束后有一个 GROUP 操作； 对于每个步骤（ID 列不为空的为一个步骤）OPERATION 每层有 1 个空格的缩进。
TABLE	OPERATION 涉及到的表，只显示别名和属性，超长截断为两个点； HASH 分布表：中括号中显示 HASH 列； 复制表：显示[REP]； 随机分布表：显示[DIS]； Nocopies 随机分布表：显示[NOCPS]； Nocopies HASH 分布表：显示[NOCPS，HASH 列]； 子查询：OPERATION 列显示 SubQueryN，其中 N 为数字，用来区分不同的子查询； 某个步骤的结果：OPERATION 列显示为 Step，本列显示为<N>其中 N 为用到的步骤的第一列 ID，表示该步骤的结果； 例如： 图中 ID 为 03 的步骤是一个汇总步骤，源表为 x 子查询； x 子查询为一个 UNION，涉及到 ssr，csr，wsr； ssr，csr，wsr 都是子查询，分别来自 02，00，01 的结果进行 GROUP BY；
CONDITION	显示操作的条件，例如 FILTER 的单表条件，JOIN 的连接条件，GROUP BY，ORDER BY，LIMIT 的内容。 如果某个条件可能使用索引，会在有索引的列后进行标注，例如： WHERE t1.a{S} > 10 表示可能使用 t1.a 的智能（Smart）索引 WHERE t1.a{H} = 10 表示可能使用 t1.a 的 HASH 索引 WHERE t1.a{H}= t2.b 表示可能使用 t1.a 的 HASH 索引 WHERE t1.a{H} = t2.b{H}表示可能使用 t1.a 和 t2.b 的 HASH 索引 WHERE contains(t1.a{F}...) 表示可能使用 t1.a 的全文（Full Text）索引 只有物理表的单列包含索引， 物理表单列的>，<，>=，<=，=可能使用智能索引 物理表单列的=可能使用 HASH 索引，包括等于常量和等值 JOIN 物理表的 contains 函数可能使用全文索引
ROW，WIDTH，COST（TOTAL）	显示代价评估的内容，分别为该步骤结果的条数，行宽，该步骤的代价，总体代价。 其中代价不包括结果进行数据移动的代价。 图中的值不是真实数据，仅供参考。 当某些表或者列缺少统计信息时，则不会显示代价相关的内容，而显

示缺少哪些表或列的统计信息，如下：

```
gbase> explain select * from x2 where id2 > (select count(*) from x3);
    +----+----------+----------+---------+--------------+----+----
---+------+
    | ID | MOTION     | OPERATION | TABLE    | CONDITION      | NO STA
Tab/Col|
+----+----------+----------+---------+--------------+----+-------+
    | 02 | [RESULT]   | SCAN       | x2[id4] | (id2{S} > &x1x&) |
|
    | 01 | [SCALAR_1] | Step       | <00>    |                |
|
    |    |            | AGG        |         |                |
|
    | 00 | [GATHER]   |Table     | x3[id4] |               | x3          |
    |    |            | AGG        |         |               | x2
|
    +----+----------+----------+---------+--------------+----+----
---+5 row in set
```

10.2 EXTENDED 输出

EXPLAIN 后边有 EXTENDED 时，显示查询计划的扩展输出格式。

示例 1：

SQL1：

```
explain extended select x1.id2 from x1, x2 where x1.id2 = x2.id3 and x1.id3 in
(select id2 from x2 where x2.id4 > 10);
```

查询计划：

```
QueryPlan:

 Uncorrelated SUB plan

==========================================

==========================================
QueryPlan:

++++++++++++++++++++++++++++++++++++++++++++++
            Leaf type: REGULAR STEP  【一般步骤，与汇总步骤对应】
        Need combiner: false【不需要汇总】
```

```
        Target temp table: _tmp_2030479552_19_t160_1_1496193667_s 【目标临时表名】
    Temp table definition: CREATE TABLE
`gctmpdb`._tmp_2030479552_19_t160_1_1496193667_s AS SELECT
/*192.168.6.121_19_15_2017-05-31_09:24:31*/ /*+ TID('131280') */ DISTINCT
`regress_db_link.x2`.`id2` AS `id2` FROM `regress_db_link`.`x2`
`regress_db_link.x2` WHERE (`regress_db_link.x2`.`id4` > 10) LIMIT 0 【创建目标
表的 SQL】
             Optimization:
             Query String: SELECT /*192.168.6.121_19_15_2017-05-31_09:24:31*/ /*+
TID('131280') */ DISTINCT `regress_db_link.x2`.`id2` AS `id2` FROM
`regress_db_link`.`x2` `regress_db_link.x2` WHERE (`regress_db_link.x2`.`id4` >
10)【查询语句】
    【索引内容】May use index: `regress_db_link`.`x2`.`id4`{Smart Index}
    【代价内容】
                CostInfo: Start(0), Run(0.11), Selectivity(0.578947), Width(8),
Rows(11)

    ==========================================
       end SUB plan

    ++++++++++++++++++++++++++++++++++++++++++
            Leaf type: REGULAR STEP
          Need combiner: false
       Target temp table: _tmp_2030479552_19_t160_2_1496193667_s
    Temp table definition: CREATE TABLE
`gctmpdb`._tmp_2030479552_19_t160_2_1496193667_s AS SELECT
/*192.168.6.121_19_15_2017-05-31_09:24:31*/ /*+ TID('131280') */
`regress_db_link.x1`.`id2` AS `id2` FROM `regress_db_link`.`x1`
`regress_db_link.x1` WHERE `regress_db_link.x1`.`id3` IN (SELECT `id2` FROM
`gctmpdb`.`_tmp_2030479552_19_t160_1_1496193667_s`) LIMIT 0
             Optimization:
             Query String: SELECT /*192.168.6.121_19_15_2017-05-31_09:24:31*/ /*+
TID('131280') */ `regress_db_link.x1`.`id2` AS `id2` FROM `regress_db_link`.`x1`
`regress_db_link.x1` WHERE `regress_db_link.x1`.`id3` IN (SELECT `id2` FROM
`gctmpdb`.`_tmp_2030479552_19_t160_1_1496193667_s`)
    【代价内容】
                CostInfo: Start(0), Run(0.07), Selectivity(0.291667), Width(8),
```

```
Rows(7)


    ++++++++++++++++++++++++++++++++++++++++++++++
                Leaf type: REGULAR STEP
            Need combiner: false
        Target temp table: _tmp_2030479552_19_t160_3_1496193667_s
    Temp table definition: CREATE TABLE
`gctmpdb`._tmp_2030479552_19_t160_3_1496193667_s AS SELECT
/*192.168.6.121_19_15_2017-05-31_09:24:31*/ /*+ TID('131280') */
`_tmp_2030479552_19_t160_2_1496193667_s`.`id2` AS `id2` FROM
`gctmpdb`._tmp_2030479552_19_t160_2_1496193667_s INNER JOIN
`regress_db_link`.`x2` `regress_db_link.x2` ON
(`_tmp_2030479552_19_t160_2_1496193667_s`.`id2` = `regress_db_link.x2`.`id3`)
LIMIT 0
             Optimization:
             Query String: SELECT /*192.168.6.121_19_15_2017-05-31_09:24:31*/ /*+
TID('131280') */ `_tmp_2030479552_19_t160_2_1496193667_s`.`id2` AS `id2` FROM
`gctmpdb`._tmp_2030479552_19_t160_2_1496193667_s INNER JOIN
`regress_db_link`.`x2` `regress_db_link.x2` ON
(`_tmp_2030479552_19_t160_2_1496193667_s`.`id2` = `regress_db_link.x2`.`id3`)
    【代价内容】
                CostInfo: Start(0.526), Run(0.52), Selectivity(0.0789474),
Width(12), Rows(10)

    ==============================================


     Uncorrelated Subplan:


     CExecStep
     -------------------------
     isQueryFinalStep = 0     【不是查询的最后一步】
     DestType = 1             【结果发送到所有节点】

     aStepDetail              【8a 所有节点属性】
```

```
    ---------------
    isProducer = 1              【是生产者】
    isConsumer = 1              【是消费者】
    producerDistID = 1          【生产者 distribution id 为 1】
    consumerDistID = 1          【消费者 distribution id 为 1】
    isSingleHashNode = 0        【非单节点 hash 优化】
    isHashRedist = 0            【不是 hash 重分布】
    isGroupHashRedist = 0
    isAllTableAreHashTmpDist = 0    【非所有源表都是 HASH 临时表】
    isExistsHashReditTable = 0          【源表不存在 HASH 临时表】
    queryString = SELECT /*192.168.6.121_19_15_2017-05-31_09:24:31*/ /*+
TID('131280') */ DISTINCT `regress_db_link.x2`.`id2` AS `id2` FROM
`regress_db_link`.`x2` `regress_db_link.x2` WHERE (`regress_db_link.x2`.`id4` >
10)
    targetTable = `gctmpdb`._tmp_2030479552_19_t160_1_1496193667_s
    targetSchema = CREATE TABLE `gctmpdb`._tmp_2030479552_19_t160_1_1496193667_s
AS SELECT /*192.168.6.121_19_15_2017-05-31_09:24:31*/ /*+ TID('131280') */
DISTINCT `regress_db_link.x2`.`id2` AS `id2` FROM `regress_db_link`.`x2`
`regress_db_link.x2` WHERE (`regress_db_link.x2`.`id4` > 10) LIMIT 0




    CExecStep
    ------------------------
    isQueryFinalStep = 0
    DestType = 1

    aStepDetail
    ---------------
    isProducer = 1
    isConsumer = 1
    producerDistID = 1
    consumerDistID = 1
    isSingleHashNode = 0
    isHashRedist = 0
```

```
    isGroupHashRedist = 0
    isAllTableAreHashTmpDist = 0
    isExistsHashReditTable = 0
    queryString = SELECT /*192.168.6.121_19_15_2017-05-31_09:24:31*/ /*+
TID('131280') */ `regress_db_link.x1`.`id2` AS `id2` FROM `regress_db_link`.`x1`
`regress_db_link.x1` WHERE `regress_db_link.x1`.`id3` IN (SELECT `id2` FROM
`gctmpdb`.`_tmp_2030479552_19_t160_1_1496193667_s`)
    targetTable = `gctmpdb`._tmp_2030479552_19_t160_2_1496193667_s
    targetSchema = CREATE TABLE `gctmpdb`._tmp_2030479552_19_t160_2_1496193667_s
AS SELECT /*192.168.6.121_19_15_2017-05-31_09:24:31*/ /*+ TID('131280') */
`regress_db_link.x1`.`id2` AS `id2` FROM `regress_db_link`.`x1`
`regress_db_link.x1` WHERE `regress_db_link.x1`.`id3` IN (SELECT `id2` FROM
`gctmpdb`.`_tmp_2030479552_19_t160_1_1496193667_s`) LIMIT 0

    Step Drop List = `gctmpdb`._tmp_2030479552_19_t160_1_1496193667_s



    CExecStep
    --------------------------
    isQueryFinalStep = 0
    DestType = 0

    aStepDetail
    ---------------
    isProducer = 1
    isConsumer = 0
    producerDistID = 1
    consumerDistID = 1
    isSingleHashNode = 0
    isHashRedist = 0
    isGroupHashRedist = 0
    isAllTableAreHashTmpDist = 0
    isExistsHashReditTable = 0
    queryString = SELECT /*192.168.6.121_19_15_2017-05-31_09:24:31*/ /*+
TID('131280') */ `_tmp_2030479552_19_t160_2_1496193667_s`.`id2` AS `id2` FROM
`gctmpdb`._tmp_2030479552_19_t160_2_1496193667_s INNER JOIN
```

```
`regress_db_link`.`x2` `regress_db_link.x2` ON
(`_tmp_2030479552_19_t160_2_1496193667_s`.`id2` = `regress_db_link.x2`.`id3`)

     Step Drop List = `gctmpdb`._tmp_2030479552_19_t160_2_1496193667_s
```

示例 2:

SQL 2:

```
    explain extended select x1.id2 from x1, x2, x3 where x1.id2 = x2.id3 and x1.id3
= x3.id2;
```

查询计划:

```
    QueryPlan:

    ++++++++++++++++++++++++++++++++++++++++++++++
                Leaf type: REGULAR STEP
            Need combiner: false
        Target temp table: _tmp_2030479552_19_t161_1_1496193667_s
    Temp table definition: CREATE TABLE
`gctmpdb`._tmp_2030479552_19_t161_1_1496193667_s AS SELECT
/*192.168.6.121_19_16_2017-05-31_09:56:34*/ /*+ TID('131281') */
`regress_db_link.x3`.`id2` AS `id2` FROM `regress_db_link`.`x3`
`regress_db_link.x3` LIMIT 0
             Optimization:
             Query String: SELECT /*192.168.6.121_19_16_2017-05-31_09:56:34*/ /*+
TID('131281') */ `regress_db_link.x3`.`id2` AS `id2` FROM `regress_db_link`.`x3`
`regress_db_link.x3`

                CostInfo: Start(0), Run(0.13), Selectivity(1), Width(4), Rows(13)


    ++++++++++++++++++++++++++++++++++++++++++++++
                Leaf type: REGULAR STEP
            Need combiner: false
        Target temp table: _tmp_rht_2030479552_19_t161_2_1496193667_s
    Temp table definition: CREATE TABLE
```

```
`gctmpdb`._tmp_rht_2030479552_19_t161_2_1496193667_s AS SELECT
/*192.168.6.121_19_16_2017-05-31_09:56:34*/ /*+ TID('131281') */
`regress_db_link.x1`.`id2` AS `id2` FROM `regress_db_link`.`x1`
`regress_db_link.x1` INNER JOIN `gctmpdb`._tmp_2030479552_19_t161_1_1496193667_s
ON (`regress_db_link.x1`.`id3` = `_tmp_2030479552_19_t161_1_1496193667_s`.`id2`)
LIMIT 0
            Optimization: {hash redist} 【是 HASH 重分布步骤】
      Hash Redist Indexes: 1【HASH 重分布下标(从 1 开始)】
            Query String: SELECT /*192.168.6.121_19_16_2017-05-31_09:56:34*/ /*+
TID('131281') */ `regress_db_link.x1`.`id2` AS `id2` FROM `regress_db_link`.`x1`
`regress_db_link.x1` INNER JOIN `gctmpdb`._tmp_2030479552_19_t161_1_1496193667_s
ON (`regress_db_link.x1`.`id3` = `_tmp_2030479552_19_t161_1_1496193667_s`.`id2`)

               CostInfo: Start(0.812), Run(0.74), Selectivity(0.0608974),
Width(12), Rows(19)


    ++++++++++++++++++++++++++++++++++++++++++++
            Leaf type: REGULAR STEP
         Need combiner: false
       Target temp table: _tmp_rht_2030479552_19_t161_3_1496193667_s
    Temp table definition: CREATE TABLE
`gctmpdb`._tmp_rht_2030479552_19_t161_3_1496193667_s AS SELECT
/*192.168.6.121_19_16_2017-05-31_09:56:34*/ /*+ TID('131281') */
`regress_db_link.x2`.`id3` AS `id3` FROM `regress_db_link`.`x2`
`regress_db_link.x2` LIMIT 0
            Optimization: {hash redist}
      Hash Redist Indexes: 1
            Query String: SELECT /*192.168.6.121_19_16_2017-05-31_09:56:34*/ /*+
TID('131281') */ `regress_db_link.x2`.`id3` AS `id3` FROM `regress_db_link`.`x2`
`regress_db_link.x2`

               CostInfo: Start(0), Run(0.19), Selectivity(1), Width(4), Rows(19)


    ++++++++++++++++++++++++++++++++++++++++++++
            Leaf type: REGULAR STEP
```

```
            Need combiner: false
        Target temp table: _tmp_2030479552_19_t161_4_1496193667_s
    Temp table definition: CREATE TABLE
`gctmpdb`._tmp_2030479552_19_t161_4_1496193667_s AS SELECT
/*192.168.6.121_19_16_2017-05-31_09:56:34*/ /*+ TID('131281') */
`_tmp_rht_2030479552_19_t161_2_1496193667_s`.`id2` AS `id2` FROM
`gctmpdb`._tmp_rht_2030479552_19_t161_2_1496193667_s INNER JOIN
`gctmpdb`._tmp_rht_2030479552_19_t161_3_1496193667_s ON
(`_tmp_rht_2030479552_19_t161_2_1496193667_s`.`id2` =
`_tmp_rht_2030479552_19_t161_3_1496193667_s`.`id3`) LIMIT 0
             Optimization:
             Query String: SELECT /*192.168.6.121_19_16_2017-05-31_09:56:34*/ /*+
TID('131281') */ `_tmp_rht_2030479552_19_t161_2_1496193667_s`.`id2` AS `id2` FROM
`gctmpdb`._tmp_rht_2030479552_19_t161_2_1496193667_s INNER JOIN
`gctmpdb`._tmp_rht_2030479552_19_t161_3_1496193667_s ON
(`_tmp_rht_2030479552_19_t161_2_1496193667_s`.`id2` =
`_tmp_rht_2030479552_19_t161_3_1496193667_s`.`id3`)

                CostInfo: Start(1.40733), Run(1.31), Selectivity(0.0789474),
Width(16), Rows(28)
```

10.2.1 查询计划部分

- 每个步骤的类型 (Leaf type), 有两种:
  - REGULAR STEP

    表示在所有节点执行;
  - COMBINER STEP

    表示在汇总节点执行;
- 该步骤是否需要汇总 (Need combiner)

  如果需要, 则为 true, 并且会有一个 COMBINER STEP 使用该步骤的目标表; 如果不需要, 则为 false。
- 每个步骤执行的 SQL 语句, 包括创建目标临时表的语句 (Temp table definition) 和查询语句 (Query String), 例如:

```
    Temp table definition: CREATE TABLE
`gctmpdb`._tmp_rht_2030479552_5_t21_1_1494478737_s AS SELECT
/*192.168.6.121_5_53_2017-05-11_16:17:05*/ /*+ TID('23') */ `lcg.x2`.`id2` AS
`id2`, `lcg.x2`.`id3` AS `id3`, `lcg.x2`.`dd` AS `dd` FROM `lcg`.`x2` `lcg.x2` LIMIT
0
```

```
    Query String: SELECT /*192.168.6.121_5_53_2017-05-11_16:17:05*/ /*+ TID('23')
*/ `lcg.x2`.`id2` AS `id2`, `lcg.x2`.`id3` AS `id3`, `lcg.x2`.`dd` AS `dd` FROM
`lcg`.`x2` `lcg.x2`
```

其中可能包括注释和 hint:

/*192.168.6.121_5_53_2017-05-11_16:17:05*/为注释

/*+ TID('23') */为任务 ID

- 每个步骤的目标临时表的名称 (Target temp table), 例如:

```
    Target temp table: _tmp_rht_2030479552_5_t21_1_1494478737_s
```

_tmp_rht_: 开头的是分布表

_tmp: 开头的表是复制表

2030479552: 发起节点 node id

5: 线程 ID (thd->thread_id)

t21: 查询 ID

1: 临时表编号

1494478737: 时间戳

s: 临时表后缀

- 每个步骤是 HASH 重分布, 还是拉复制表。

如果是 HASH 重分布, 指明计算 HASH 值所使用的表达式 (Hash Redist Indexes), 通过一个整数值 N 来表示, N 大于 0 时, 表示 Query String 投影列的第 N 个表达式 (从 1 开始)。

当 N 等于 0 时, 表示随机分布;

当 N 等于-10 时，表示查询结果直接落在分片上，不进行重分布。该情况通常用在临时表复用时，被复用的临时表结果不进行重分布，而是直接落在计算出结果的节点上。

例如：

```
    Optimization: {hash redist}
      Hash Redist Indexes: 1
```

```
    Optimization: {rand redist}
      Hash Redist Indexes: 0
```

```
    Optimization: {no redist}
      Hash Redist Indexes: -10
```

没有上述说明则是拉复制表或者汇总表。

- 可能使用的索引

列出可能使用索引的单列，索引类型包括：

- {Smart Index}：智能索引，扫描时使用；
- {Hash Index}：Hash 索引，等值比较时使用；
- {Full Text}：全文索引，Contains 时使用；

例如：

May used index: `regress_db_link.x1`.`entry_id`{Smart Index} `regress_db_link.x1`.`id2`{Hash Index}

- 如果是成本评估计划，输出每个步骤的启动成本(Start)，运行成本(Run)，选择率 (Selectivity)，记录宽度 (Width)，结果条数 (Rows)：

启动成本：数据重分布，拉复制表的成本；

运行成本：单表扫描，JOIN 的成本；

选择率：对于单表，表示过滤条件过滤后的记录所占全表比例；

对于 JOIN，表示 JOIN 条件过滤后的记录占笛卡尔积的比例。

记录宽度：表示该步骤所有投影列数据长度的和。对于定长类型，使用类型长度，变长类型，使用统计信息中的数据平均宽度。

```
CostInfo: Start(0), Run(10), Selectivity(1), Width(16), Rows(1000)
```

10.2.2 执行计划部分

执行计划部分主要内容与查询计划一致，同时增加了一些其它属性，如下表：

属性	含义
isQueryFinalStep	是否查询步骤的最后一步
DestType	目标类型：1 为所有节点，2 为汇总节点，3 为某个节点（说明使用该目标表的 SQL 全部使用复制表）其它没有使用
isProducer	是否生产者：0 不是，1 是
isConsumer	是否消费者：0 不是，1 是
producerDistID	生产者 ditribution ID
consumerDistID	消费者 distribution ID
isSingleHashNode	是否单节点 HASH 优化，0 不是，1 是
isHashRedist	是否 HASH 重分布步骤，0 不是，1 是
Hash Redist Indexes	HASH 重分布的表达式下标
isGroupHashRedist	未使用
isAllTableAreHashTmpDist	是否所有源表都是 HASH 重分布临时表。0 不是，1 是
isExistsHashReditTable	是否源表中存在 HASH 重分布临时表。0 不是，1 是
queryString	该步骤的 SQL
targetTable	目标表
targetSchema	目标表建表语句
[Step] DropList	该步骤执行完后可以删除的临时表列表

10.3 PARTITIONS 输出

EXPLAIN 后带有 PARTITIONS 时显示树形输出。

PARTITIONS 方式输出树形显示的查询计划，主要包括数据重分布方式和每个步骤的主要操作，显示的内容与缺省方式类似。

示例： TPC-DS SQL-5 的计划：

```
    +-------------------------------------------------------------------------------
-----------------------------------------------+
    | PlanTree
|
    +-------------------------------------------------------------------------------
-----------------------------------------------+
    | 04: [RESULT] (row=2 width=26 cost=0.00693147(479021))
|
    |         Step : <03>
|
    |         GROUP BY  ROLLUP ((channel), (id))
|
    |         ORDER BY channel ASC, id ASC
|
    |         LIMIT 100
|
    | --->03: [GATHER] (row=0 width=26 cost=250451(479021))
|
    |             Subquery 2 : x
|
    |
[
                        |
    |                   Subquery 3 : ssr
|
    |                       Step : <02>
|
    |                       GROUP BY s_store_id
|
    |               ]
|
    |               UNION ALL
|
    |
[
                        |
```

```
    |                    Subquery 7 : csr
|
    |                        Step : <00>
|
    |                        GROUP BY cp_catalog_page_id
|
    |                ]
|
    |                UNION ALL
|
    |
[
                        |
    |                    Subquery 11 : wsr
|
    |                        Step : <01>
|
    |                        GROUP BY web_site_id
|
    |                ]
|
    |            GROUP BY channel, id
|
    |     --->02: [REDIST] (row=18 width=48 cost=128146(228570))
|
    |            Redist Key : (s_store_id)
|
    |                INNER JOIN
|
    |                ON (store_sk = s_store_sk)
|
    |                    INNER JOIN
|
    |                    ON (date_sk = d_date_sk)
|
    |                        Table : tpcds.date_dim
|
```

```
    |                              Replicated
|
    |                           WHERE (d_date BETWEEN cast('1998-08-04' as date) AND
date_add(cast('1998-08-04' as date), INTERV |
    |                           WHERE     AL 14 DAY))
|
    |                           Subquery 4 : salesreturns
|
    |
[
              |
    |                                  Table : tpcds.store_sales
|
    |                                      Hash Key : ss_item_sk
|
    |                              ]
|
    |                              UNION ALL
|
    |
[
              |
    |                                  Table : tpcds.store_returns
|
    |                                      Hash Key : sr_item_sk
|
    |                              ]
|
    |                      Table : tpcds.store
|
    |                          Replicated
|
    |                 GROUP BY s_store_id
|
    |     --->01: [REDIST] (row=45 width=48 cost=32897.2(100425))
|
```

```
    |             Redist Key : (web_site_id)
|
    |                 INNER JOIN
|
    |                 ON (wsr_web_site_sk = web_site_sk)
|
    |                     INNER JOIN
|
    |                     ON (date_sk = d_date_sk)
|
    |                         Table : tpcds.date_dim
|
    |                             Replicated
|
    |                         WHERE (d_date BETWEEN cast('1998-08-04' as date) AND
date_add(cast('1998-08-04' as date), INTERV |
    |                         WHERE     AL 14 DAY))
|
    |                         Subquery 12 : salesreturns
|
    |
[
            |
    |                               Table : tpcds.web_sales
|
    |                                   Hash Key : ws_item_sk
|
    |                               ]
|
    |                               UNION ALL
|
    |
[
            |
    |                               LEFT JOIN
|
```

```
    |                                  ON (wr_item_sk = ws_item_sk) AND
(wr_order_number = ws_order_number)                        |
    |                                      Table : tpcds.web_returns
|
    |                                          Hash Key : wr_item_sk
|
    |                                      Table : tpcds.web_sales
|
    |                                          Hash Key : ws_item_sk
|
    |                                ]
|
    |                        Table : tpcds.web_site
|
    |                            Replicated
|
    |                   GROUP BY web_site_id
|
    |      --->00: [REDIST] (row=35154 width=48 cost=67527.5(67527.5))
|
    |              Redist Key : (cp_catalog_page_id)
|
    |                  INNER JOIN
|
    |                  ON (page_sk = cp_catalog_page_sk)
|
    |                      INNER JOIN
|
    |                      ON (date_sk = d_date_sk)
|
    |                          Table : tpcds.date_dim
|
    |                              Replicated
|
    |                          WHERE (d_date BETWEEN cast('1998-08-04' as date) AND
date_add(cast('1998-08-04' as date), INTERV |
```

```
    |                          WHERE     AL 14 DAY))
|
    |                          Subquery 8 : salesreturns
|
    |
[
              |
    |                                Table : tpcds.catalog_sales
|
    |                                    Hash Key : cs_item_sk
|
    |                               ]
|
    |                               UNION ALL
|
    |
[
              |
    |                                Table : tpcds.catalog_returns
|
    |                                    Hash Key : cr_item_sk
|
    |                               ]
|
    |                      Table : tpcds.catalog_page
|
    |                          Replicated
|
    |                 GROUP BY cp_catalog_page_id
|
    +-------------------------------------------------------------------------
-----------------------------------------+
    98 rows in set (Elapsed: 00:00:00.29)
```

当缺少统计信息时，不显示代价评估的内容，在表头显示缺少哪些表或列的统计信息，如下：

```
gbase> explain select * from x2 where id2 > (select count(*) from x3);
    +-----------------------------------------+
    |  PlanTree  [ NO STAT Tab/Col:x3 x2]     |
+-----------------------------------------+
    | 02 : [RESULT]                           |
    |          Table : regress_db_link.x2     |
    |              Hash Key : id4             |
    |         WHERE (id2 > &x1x&)             |
    | -->01: [SCALAR 1]                       |
    |             Step : <00>                 |
    |             AGG                         |
    |     -->00: [GATHER]                     |
    |                 Table : regress_db_link.x3 |
    |                     Hash Key : id4      |
    |                 AGG                     |
    +-----------------------------------------+
    11 row in set
```

11 集群的权限管理

11.1 用户管理

用户可以使用CREATE USER语句创建一个新的GBase 8a MPP Cluster帐号。

更多关于用户管理的语法说明，请参见《GBase 8a MPP Cluster SQL参考手册》的相关章节内容。

下面我们通过两个示例，来向用户展示创建用户和更改用户密码的操作。

示例1：使用超级用户root登录，创建一个用户user1。

```
$ gccli -uroot -p
Enter password:

    GBase client 8.5.1.2 build 26068. Copyright (c) 2004-2013, GBase.  All Rights
Reserved.

gbase> CREATE USER user1;
    Query OK, 0 rows affected
```

示例2：使用超级用户root登录，修改用户user1的密码，然后使用user1的新密码登录集群。

```
$ gccli -uroot -p
Enter password:

    GBase client 8.5.1.2 build 26068. Copyright (c) 2004-2013, GBase.  All Rights
Reserved.


gbase> SET PASSWORD FOR user1 = PASSWORD('H133%_h');
    Query OK, 0 rows affected
```

退出登录，使用 use1 用户登录集群。验证修改口令的正确性。

```
$ gccli -uuser1 -p
Enter password:

    GBase client 8.5.1.2 build 26068. Copyright (c) 2004-2013, GBase.  All Rights
Reserved.

    gbase>
```

11.2 权限管理

在实际的项目中，我们建议用户规划好不同的数据库用户的职责，并给它赋予相应的操作权限，用以保证数据库的安全操作。

更多关于权限的的语法说明，请参见《GBase 8a MPP Cluster SQL 参考手册》的相关章节内容。

下面我们通过两个示例，来向用户展示权限管理的操作。

示例一：使用超级用户 root，创建一个 user_general 用户，该用户具备 SELECT 操作的权限。

```
$ gccli -uroot -p
Enter password:

    GBase client 8.5.1.2 build 26068. Copyright (c) 2004-2013, GBase.  All Rights
Reserved.

gbase> CREATE USER user_general;
    Query OK, 0 rows affected

gbase> SET PASSWORD FOR user_general = PASSWORD('H%897_@m');
    Query OK, 0 rows affected
```

对user_general 用户只赋予 SELECT 权限。*.*代表所有数据库的数据库对

象,例如：表，视图，存储过程。

```
gbase> GRANT SELECT ON *.* TO user_general;
    Query OK, 0 rows affected

gbase> \q
    Bye
```

使用 user_general 登录数据库，验证其具备全部权限。

存在 test 数据库和一张 t1 表，这只是为演示示例提前创建完毕的。

```
$ gccli -uuser_general -p
Enter password:

    GBase client 8.5.1.2 build 26068. Copyright (c) 2004-2013, GBase.  All Rights
Reserved.

gbase> USE test;
    Query OK, 0 rows affected

gbase> UPDATE t1 SET a = 11 WHERE a = 10;
    ERROR 1142 (42000): UPDATE command denied to user 'user_general'@'localhost'
for table 't1'

gbase> DELETE FROM t1;
    ERROR 1142 (42000): DELETE command denied to user 'user_general'@'localhost'
for table 't1'

gbase> SELECT * FROM t1;
    +------+
    | a    |
    +------+
    |    1 |
    |    2 |
    |    3 |
    |    4 |
    |    5 |
```

```
|      6 |
|      7 |
|      8 |
|      9 |
|     10 |
+--------+
10 rows in set
```

示例二：使用超级用户 root，创建一个 user_admin 用户，该用户具备超级用户的权限，即全部的权限。

```
$ gccli -uroot -p
Enter password:

    GBase client 8.5.1.2 build 26068. Copyright (c) 2004-2013, GBase.  All Rights Reserved.

gbase> CREATE USER user_admin;
    Query OK, 0 rows affected

gbase> SET PASSWORD FOR user_admin = PASSWORD('H%897_@m');
    Query OK, 0 rows affected
```

对于 user_admin 用户赋予全部权限。*.*代表所有数据库的数据库对象，例如：表，视图，存储过程。

```
gbase> GRANT ALL ON *.* TO user_admin;
    Query OK, 0 rows affected

gbase> \q
    Bye
```

使用 user_admin 用户登录数据库，验证其只具备 SELECT 权限。

存在 test 数据库和一张 t1 表，这只是为演示示例提前创建完毕的。

```
$ gccli -uuser_admin -p
Enter password:
```

```
    GBase client 8.5.1.2 build 26068. Copyright (c) 2004-2013, GBase.  All Rights
Reserved.

gbase> USE test;
    Query OK, 0 rows affected

    -- 可以 SELEC 数据。
gbase> SELECT * FROM t1;
    +------+
    | a    |
    +------+
    |    1 |
    |    2 |
    |    3 |
    |    4 |
    |    5 |
    |    6 |
    |    7 |
    |    8 |
    |    9 |
    |   10 |
    +------+
    10 rows in set

    -- 可以 UPDATE 数据。
gbase> UPDATE t1 SET a = 11 WHERE a = 10;
    Query OK, 1 row affected

gbase> SELECT * FROM t1;
    +------+
    | a    |
    +------+
    |    1 |
    |    2 |
    |    3 |
```

```
|    4 |
|    5 |
|    6 |
|    7 |
|    8 |
|    9 |
|   11 |
+------+
10 rows in set
```

-- 可以 DELETE 数据。

```
gbase> DELETE FROM t1 WHERE a >= 5;
    Query OK, 6 rows affected

gbase> SELECT * FROM t1;
    +------+
    | a    |
    +------+
    |    1 |
    |    2 |
    |    3 |
    |    4 |
    +------+
    4 rows in set
```

-- 可以 CREATE 用户。

```
gbase> CREATE USER use_test;
    Query OK, 0 rows affected
```

-- 可以 DROP 用户。

```
gbase> DROP USER use_test;
    Query OK, 0 rows affected
```

11.3 用户组管理

用户组功能主要为便于对用户权限进行管理。在实际应用中针对用户进行授权费时费力。把有相同权限的用户组织在一起做为一个组，用户只需要修改组的权限，就会影响该组下所有用户的权限。

在集群中创建用户组，语法格式：

CREATE ROLE [IF NOT EXISTS] role [, role] ......

12 安全管理

12.1 密码安全管理

为保证密码安全管理在整个集群环境的一致性，需要在各 coordinator 节点的配置文件 gbase_8a_gcluster.cnf 中和各 node 节点的配置文件 gbase_8a_gbase.cnf 中，对相同变量配置相同的值。

12.1.1 密码强度控制

用户可以配置密码复杂度和长度要求，在创建密码和修改密码时必须符合此强度要求。

密码复杂度受只读量 password_format_option 控制。

密码长度受只读变量 password_min_length 控制。

两个变量的值均为 0 时表示关闭密码强度控制。

密码强度控制对以下 SQL 类型生效：

```
create user [user] identified by 'pass';
alter user [user] identified by 'pass';
set password = password('pass');
set password for [user] = password('pass');
```

密码强度控制参数为只读变量，定义如下：

变量名	范围	含义
password_format_option	0-31	表示密码字符组合要求，默认值为 0，表示无复杂度要求。 组合中可包含数字（1）、小写字符（2）、大写字符（4）、其它字符（8）中的 1 种或多种。 1：表示必须包含数字。 2：表示必须包含小写字母。

		4：表示必须包含大写字母。 8：表示必须包含其它字符。 16：表示不能和用户名相同。 要限定组合时配置为上述值的和，可以任意组合。 例如：限定包含所有种类字符为（1+2+4+8=15）。 对非英文字符，按其对应的 ASII 码范围分类。
password_min_length	0-65535	表示密码的最短长度，默认值为 0，表示不限制长度。

12.1.2 密码重用控制

限制用户使用指定间隔次数内的历史密码。

口令间隔控制参数为只读变量，定义如下：

变量名	范围	含义
password_reuse_max	0-100	默认值为 0，表示不控制。正数值 N 表示允许的口令间隔，间隔大于 N 才允许设置。

12.1.3 密码有效期控制

控制密码的有效期，达到有效期后用户密码自动过期。用户密码过期后，允许用户登录，在执行 SQL 时提示用户修改密码。此时允许用户修改自己的密码。必须修改密码后才允许执行其它 SQL。

可选的密码策略：

1.使用默认密码过期时间

默认密码过期时间为只读变量，定义如下：

变量名	范围	含义
password_life_time	0-65535	默认值为 0，表示默认禁用密码过期。正数值 N 表示密码过期天数，密码必须在 N 天后修改。

使用默认密码过期策略范例：

```
CREATE USER jeffrey PASSWORD EXPIRE DEFAULT;//创建用户同时符合默认密码过期时间
ALTER USER jeffrey PASSWORD EXPIRE DEFAULT;//修改用户符合默认密码过期时间
```

2.禁用密码过期策略：

此时密码将永不过期，范例：

```
CREATE USER jeffrey PASSWORD EXPIRE NEVER;
ALTER USER jeffrey PASSWORD EXPIRE NEVER;
```

3.指定密码过期时间间隔：

设置密码过期的时间间隔为 N 天。范例：

```
CREATE USER jeffrey PASSWORD EXPIRE INTERVAL 180 DAY;
ALTER USER jeffrey PASSWORD EXPIRE INTERVAL 180 DAY;
```

4.使密码立即过期：

使密码立即过期，范例：

```
CREATE USER jeffrey PASSWORD EXPIRE;
ALTER USER jeffrey PASSWORD EXPIRE;
```

用户密码过期配置下发到各 coordinator 节点，语法定义如下：

```
CREATE/ALTER USER
    user [auth_option]
    [expiration_option | lock_option | host_option] ...
expiration_option: {
      PASSWORD EXPIRE
    | PASSWORD EXPIRE DEFAULT
    | PASSWORD EXPIRE NEVER
    | PASSWORD EXPIRE INTERVAL N DAY
}
```

1、对不存在用户，使用 create user 时可以同时指定密码过期选项。

2、对已存在用户，使用 alter user 更改用户时可指定密码过期选项。

用户密码过期时执行 SQL 会提示用户必须先修改密码，例如：

```
gbase> use test;
ERROR 1840 (HY000): You must reset your password using ALTER USER statement
before executing this statement.
```

密码过期设置会立即生效，如果指定了多个策略选项，最后一个生效。

12.2 用户安全管理

12.2.1 登录重试锁定

限制用户的登录重试次数，当密码错误达到次数时锁定账户。处于锁定状态的用户在登录时将提示用户账户已锁定，并拒绝登录。

用户登录重试次数参数为只读变量，定义如下：

变量名	范围	含义
login_attempt_max	0-65535	默认值为 0，表示禁用密码重试，不记录登录信息。正数值 N 表示允许密码输错的次数，达到次数则锁定用户账户。

12.2.2 账户锁定和解锁

用户账户锁定后将不允许用户登录。账户不会自动解锁，需由拥有 CREATE USER 权限的管理员账户锁定和解锁用户。锁定只影响用户登录。

用户账户锁定状态下发到各 coordinator 节点，语法定义如下：

```
CREATE/ALTER USER
    user [auth_option]
    [expiration_option | lock_option | host_option] ...
lock_option: {
      ACCOUNT LOCK
    | ACCOUNT UNLOCK
}
```

功能说明:

1、对不存在用户，使用 create user 时可以同时设置为锁定状态或非锁定状态。

2、对已存在用户，使用 alter user 更改用户为锁定或非锁定状态。

已锁定的账户登录时提示用户锁定，拒绝登录，例如:

```
gbase -uuser1
ERROR 1830 (HY000): Access denied for user 'user1'@'%'. Account is locked.
```

如果指定了多个策略选项，最后一个生效。gbase 用户不受锁定控制，可以锁定解锁，锁定时依然允许登录。

12.2.3 账户限定 host 列表

语法:

```
CREATE/ALTER USER
    user [auth_option]
    [expiration_option | lock_option | host_option] ...
host_option: {
      hosts 'host_list'
}
```

默认 host_list 为空，此时登录无 host 限定。host_list 的值可以为 ip 或主机名，允许包含多个，使用空格“ ”分割，使用“%”和“_”做通配符（通配符用法同原 host 功能）。登录的 ip 或主机名在列表中才允许用户登录。

12.3 查看安全信息

从 gbase.user_check 系统表中可以查询到用户安全信息。

user_check 表结构如下:

列名	类型	Null	默认值	含义
host	char(60)	No	无	主机名，主键

user	char(16)	No	无	用户名，主键
attempt	smallint(5) unsigned	No	0	密码重试次数
last_attempt	smallint(5) unsigned	No	0	最近成功登录的重试次数
locked	enum('N','Y')	No	N	用户是否锁定
password_expired	enum('N','Y')	No	N	密码是否过期
password_last_changed	datetime	No	无	最近密码修改时间
password_life_time	smallint(5) unsigned	Yes	NULL	密码有效期，单位天
password_history	varchar(5000)	No	""	密码历史列表，密文
host_list	varchar(5000)	No	""	允许登录的 host 列表
login_time	datetime	Yes	NULL	本次登录时间
login_host	char(60)	No	""	本次登录主机
last_login_time	datetime	Yes	NULL	最近登录时间
last_login_host	char(60)	No	""	最近登录主机
login_count	bigint(8)	No	0	用户登录次数

查询用户的锁定和密码过期状态范例：

```
gbase> select Locked,password_expired from gbase.user_check where user =
'user1';
+--------+------------------+
| Locked | password_expired |
+--------+------------------+
| N      | N                |
+--------+------------------+
```

12.4 登录信息显示

登录信息显示受只读变量 show_login_status 控制：

变量名	范围	含义
show_login_status	0-1	默认值为 0，表示关闭。1 表示开启。

当开启登录信息显示时，在用户登录时可以显示以下信息，范例如下：

```
Login info:
                USER: root
          LOGIN_TIME: 2017-10-25 09:33:00
          LOGIN_HOST: localhost
     LAST_LOGIN_TIME: 2017-10-24 15:22:04
     LAST_LOGIN_HOST: localhost
          LAST_ATTEMPT: 0
VALID_PASSWORD_EXPIRE: 1
```

新增语法 show login status，用于显示用户自身的登录信息，范例如下：

```
gbase> show login status\G
*************************** 1. row ***************************
                USER: user1
          LOGIN_TIME: 2017-10-25 09:38:50
          LOGIN_HOST: localhost
     LAST_LOGIN_TIME: 0000-00-00 00:00:00
     LAST_LOGIN_HOST:
          LAST_ATTEMPT: 0
VALID_PASSWORD_EXPIRE: 1
```

12.5 安全管理的权限

CREATE USER 语法执行受 CREATE USER 权限控制

ALTER USER 语法执行除修改自己密码外受 CREATE USER 权限控制，修改自己密码不受权限控制。

查看安全信息受系统表 gbase.user_check 的 SELECT 权限控制。

12.6 登陆管理一致性

在集群范围内进行密码安全管理需要考虑到一致性的问题，面临着如下场景

集群中包括 A B C 三个 coordinator

1 在 A 点进行登陆，达到密码锁定次数，A 点将进行锁定，同时 B 点 C 点也应该进行锁定。

2 在 A 点进行登陆，密码重试出错，那么 B 点 C 点即使没有进行密码重试，也需要进行重试次数增加。

3 在 A 点进行登陆，密码验证成功，密码重试次数需要清 0，如果 B 点，C 点密码重试次数不为 0，也需要进行清 0。

由于在场景 3 下，每进行一次登陆，如果都需要在所有节点去进行密码重试次数清 0 操作，那么会很影响效率。

并且清 0 操作此时会记录 event，如果存在节点离线，那 event 会大幅度增加，会给系统带来很大冲击。

因此这里增加一个参数进行控制，默认关闭，在明白上述影响的情况下，可以手动进行开启。

变量名	范围	含义
gcluster_user_check_consistent	0-1	为 0 表示关闭，为 1 表示开启，默认关闭

13 资源管理

13.1 概念说明及管理接口

13.1.1 Consumer Group

Consumer Group 称为资源消费组，用户可根据具体业务资源使用状况将集群使用者分为不同的类别组，组内成员资源消费行为一致。

13.1.1.1 创建

1.句法

CREATE CONSUMER GROUP <*group_name*> [comment = '*comment*'];

2.实例:

CREATE CONSUMER GROUP group1 comment = 'test group1';

13.1.1.2 更改

1.句法

(1) 更改名称:

ALTER CONSUMER GROUP <*group_name*> RENAME [TO] <*new_name*>;

(2) 更改描述

ALTER CONSUMER GROUP <*group_name*> COMMENT = '*comment*';

(3) 增加成员

ALTER CONSUMER GROUP <*group_name*> ADD USER <*user_name*>;

(4) 删除成员

ALTER CONSUMER GROUP <*group_name*> REMOVE USER <*user_name*>;

2. 实例

(1) 创建用户

CREATE USER user1;

CREATE USER user2;

(2) 资源消费组新增用户

ALTER CONSUMER GROUP group1 ADD USER user1;

ALTER CONSUMER GROUP group1 ADD USER user2;

(3) 资源消费组删除用户

ALTER CONSUMER GROUP group1 REMOVE user1;

ALTER CONSUMER GROUP group1 REMOVE user2;

3.约束

(1) consumer group 与 user 之间是一对多的关系，即一个 consumer group 中可以包含多个 user，一个 user 只能隶属于一个 consumer group;

(2) 若有激活的 plan 时，无法执行 user 与 consumer group 挂接与解除操作;

13.1.1.3 删除

1.句法

drop consumer group <*group_name*>;

2.实例:

drop consumer group group1;

3.约束

(1) 若 consumer group 在 directive 中被引用，无法执行删除 consumer group 操作;

(2)若 consumer group 关联 user，无法执行删除 consumer group 或 user

操作；

13.1.1.4 缺省组

集群中有一个固定名为 default_consumer_group 的缺省资源消费组，在激活任何一个资源计划时，对于所有没有显式挂接任何资源消费组的用户或在资源计划中没有挂接资源池的消费组中的所有用户，其均会归属于该组，其会在集群安装过程中创建，同时在集群使用过程中，无法删除、修改或创建同名 consumer group。

13.1.2 Resource Pool

资源池为集群任务执行过程中的资源供给者与管理者，分为静态与动态两种，静态资源池为资源供给者而动态资源池为资源管理者，约束任务对资源的使用。

13.1.2.1 创建

1. 句法

CREATE RESOURCE POOL <*resource_pool_name*> (*pool_attribute=value* [, ...]) TYPE {static|dynamic} [BASE ON <*parent_pool_name*>]

其中，pool_attribute is:

[priority={1|2|3|4|5|6|7|8 }]

< cpu_percent=integer >

< max_memory=integer >

< max_temp_diskspace=integer >

```
< max_disk_space=integer >
< max_disk_writeio= integer >
< max_disk_readio=integer >
[ max_activetask=integer ]
[ task_max_parallel_degree=integer ]
[ task_waiting_timeout=integer ]
[ task_running_timeout=integer ]
```

说明:

priority - 共分 8 级，1 为最高，8 为最低 ，视为保留参数，建议统一配置 1;

cpu_percent - 使用 CPU 资源的百分比，以整数表示，范围为[1 ， 100]，对于静态资源池为 CPU 带宽控制 (可参考 linux cgroup/cpu 中 cpu.cfs_quota_us 参数说明，计算公式为 cpu.cfs_quota_us = (cpu_cores * cpu.cfs_period_us) * cpu_percent) ,对于动态池为 CPU 使用权重控制(可参考 linux/cgroup 中 cpu.shares 参数说明， 计算公式为 cpu.shares = 1024 * cpu_percent);

max_memory - 最大使用虚拟内存量，设置单位为 M，动态池总值应小于或等于其所在静态池设定;

max_activetask - 此参数为动态资源池专属，表明同时池中可并发任务数，此参数设置的一个主要考量点为内存，每个任务的内存使用量限制 = max_memory / max_activetask，所以其过大会使得每个任务内存使用量下降，导致任务执行失败，缺省值 20;

max_temp_diskspace - 池中任务执行中可使用临时磁盘量，设置单位为 M，(由于其为必填参数，所以在不想对其关注的情况下，动静态均取高数值即可）;

max_disk_space - 受资源池管理的所有用户的表空间占用磁盘总和，设

置单位为 M，（由于其为必填参数，所以在不想对其关注的情况下，动静态均取高数值即可）；

max_disk_writeio – 池中任务对所有磁盘访问的写速率限制，设置单位为 MB/S，动态池总值应小于或等于其所在静态池设定 （由于其为必填参数，所以在不想对其关注的情况下，动静态均取高数值即可）；

max_disk_readio – 池中任务对所有磁盘访问的读速率限制，设置单位为 MB/S，动态池总值应小于或等于其所在静态池设定（由于其为必填参数，所以在不想对其关注的情况下，动静态均取高数值即可）；

task_max_parallel_degree – 池中任务执行并发度，缺省为 16；注：one pass group、并行 hash group、并行 update、并行 order by，在并行物化阶段，会占用 2 倍的并行度。应分配 2 倍富裕量的并行度来避免发生串行。此并发度并不影响加载的并发度设置。

task_waiting_timeout – 池中任务等待执行超时，设置单位为秒，其设置经验值为 task_waiting_timeout = 最大容忍等待队列长度 *（task_running_timeout * 调整系数），由于 task_running_timeout 一般会高于池中任务实际执行时长，所以可做适当调整，缺省为 2592000s；

task_running_timeout – 池中任务执行超时，设置单位为秒，其调整可参考集群中对于池中任务平均执行统计值，缺省为 2592000s；

2. 实例

（1）创建静态资源池

```
CREATE RESOURCE POOL static_pool0(
cpu_percent=100,
max_memory=1000,
max_temp_diskspace= 2000,
max_disk_space= 2000,
max_disk_writeio=1000,
max_disk_readio=1000) TYPE static;

CREATE RESOURCE POOL static_pool1(
```

```
cpu_percent=50,
max_memory=100,
max_temp_diskspace=200,
max_disk_space=200,
max_disk_writeio=100,
max_disk_readio=100) TYPE static;
```

(2) 创建动态资源池

```
CREATE RESOURCE POOL dynamic_pool0(
priority=1,
cpu_percent=100,
max_memory=1000,
max_temp_diskspace=2000,
max_disk_space=2000,
max_disk_writeio=1000,
max_disk_readio=1000,
max_activetask=2,
task_max_parallel_degree=100,
task_waiting_timeout=100000,
task_running_timeout=100000)
TYPE dynamic BASE ON static_pool0;

CREATE RESOURCE POOL dynamic_pool1(
priority=2,
cpu_percent=100,
max_memory=90,
max_temp_diskspace=200,
max_disk_space=200,
max_disk_writeio=90,
max_disk_readio=90,
max_activetask=20,
task_max_parallel_degree=10,
task_waiting_timeout=1000,
task_running_timeout=1000)
TYPE dynamic BASE ON static_pool1;
```

3. 约束

(1) 资源池分为静态资源池与动态资源池;

(2) 一个静态资源池中可以包含多个动态资源池;

(3) 一个动态资源池只能且必须隶属于一个静态资源池;

(4) 创建静态资源池下列参数

cpu_percent
max_memory
max_temp_diskspace
max_disk_space
max_disk_writeio
max_disk_readio

必须填写赋值，因其它参数针对静态资源池无效,建议不用出现在定义语句中;

(5) 创建动态资源池下列参数

cpu_percent
max_memory
max_temp_diskspace
max_disk_space
max_disk_writeio
max_disk_readio

必须填写赋值，参数设定不能超过挂接静态池对应参数，其他参数选填;

13.1.2.2 更改

1.句法

(1) 更改名称

ALTER RESOURCE POOL <*resource_pool_name*> RENAME [TO] <new_*resource_pool_name*>;

(2) 更改参数

ALTER RESOURCE POOL <*resource_pool_name*> SET (*pool_attribute=value* [, ...])

其中, pool_attribute is:

```
[ priority={1|2|3|4|5|6|7|8 } ]
[ cpu_percent=integer ]
[ max_memory=integer ]
[ max_temp_diskspace=integer ]
[ max_disk_space=integer ]
[ max_disk_writeio= integer ]
[ max_disk_readio=integer ]
[ max_activetask=integer ]
[ task_max_parallel_degree=integer ]
[ task_waiting_timeout=integer ]
[ task_running_timeout=integer ]
```

2.实例

(1) 更改名称

```
ALTER RESOURCE POOL resource_pool_1 RENAME resource_pool_2;
```

(2) 更改参数

```
ALTER RESOURCE POOL resource_pool_2 SET (cpu_percent=20);
```

13.1.2.3 删除

1.句法

```
drop resource pool <resource_pool_name>;
```

2.实例

```
drop resource pool resource_pool_2;
```

3.约束

(1) 删除静态资源池，必须先删除其挂接的动态资源池；

(2) 若一个动态资源池在 directive 中引用，则无法删除该动态资源池；

13.1.2.4 操作系统兼容说明

由于资源管理功能依托于系统服务 cgroup，在不同的操作间存在 cgroup 服务的不同，资源管理的表现会有不同，特在此说明：

配置项	所属资源池种类	SUSE 11 SPX Redhat 6.X	SUSE 12+	Redhat 7+	备注
cpu_percent	静	不生效	生效	生效	Cgroup 缺少该选项，/cgroup/cpu/cpu.cfs_quota_us 动态资源池不受影响
Priority	动	生效	生效	生效	
max_memory	静、动	生效	生效	生效	
max_temp_diskspace	静、动	生效	生效	生效	
max_disk_writeio	静、动	不生效	生效	生效	Cgroup 缺少该选项，/cgroup/blkio
max_disk_readio	静、动	不生效	生效	生效	Cgroup 缺少该选项，/cgroup/blkio
max_activeta	动	生效	生效	生效	

配置项	所属资源池种类	SUSE 11 SPX Redhat 6.X	SUSE 12+	Redhat 7+	备注
sk					
task_max_parallel_degree	动	条件生效	条件生效	条件生效	受到 gnode 线程池配置限制，生效需要 max_activetask* task_max_parallel_degree < gbase_parallel_max_thread_in_pool 注:该参数对优先级的影响较大
task_waiting_timeout	动	生效	生效	生效	
task_running_timeout	动	条件生效	条件生效	条件生效	收到 sql 是否可中断影响，DDL，更新索引等不可中断 SQL 不起作用

注：这里只对我们主流支持的操作系统做了说明，其他的操作系统可以通过查看 cgroup 是否存在相关组件来进行推断.

在 Redhat/Centos 7.3 以下版本中存在"cgroups cpu quota 调度引起系统宕机"的问题，GBase 8a 资源管理多静态池场景时有一定风险触发该问题，RedHat7.3/SUSE12 以上版本合入了该问题的 patch，因此建议使用多静态池场

景的用户，需升级至更新操作系统版本。该 Kernel BUG 相关说明链接：https://lists.gt.net/linux/kernel/2463496 。

13.1.3 Resource Plan

Resource Plan 资源计划为按照一定规律规划集群中资源使用的方案，将资源消费组挂接到合理的资源池中，以保证更有效地利用集群资源。

13.1.3.1 创建

1.句法

CREATE RESOURCE PLAN <*plan_name*> [comment ='*comment*'];

2. 实例

CREATE RESOURCE PLAN PLAN1 COMMENT = 'TEST PLAN1';

13.1.3.2 更改

1.句法

(1) 更改名称

ALTER RESOURCE PLAN <PLAN_NAME> RENAME [TO] <NEW_NAME>;

(2) 更改描述

ALTER RESOURCE PLAN <*plan_name*> comment ='*comment*';

2.实例

ALTER RESOURCE PLAN plan1 RENAME plan3;

ALTER RESOURCE PLAN plan3 COMMENT = 'test plan3';

13.1.3.3 删除

1.句法

DROP RESOURCE PLAN <*plan_name*>;

2.实例

DROP RESOURCE PLAN plan3;

3.约束

(1) 若 plan 在 directive 中引用，则无法删除 plan;

13.1.3.4 激活

1.句法

SET GLOBAL active_resource_plan =<*resource_plan_name*>;

2.实例

SET GLOBAL active_resource_plan = 'plan3';

3.约束

(1) 同一时段只能一个 plan 被激活;

13.1.3.5 关闭

1.句法

SET GLOBAL active_resource_plan = '';//空串

2.实例

SET GLOBAL active_resource_plan = '';

13.1.4 Resource Directive

Resource Directive 资源编排为制定在资源计划中资源消费组与动态资源池的挂接关系。

13.1.4.1 创建

1.句法

```
CREATE RESOURCE DIRECTIVE <directive_name> (
                    plan_name = 'plan_name',
                    group_name = 'group_name',
                    pool_name = 'resource_pool_name',
                    [comment = 'comment']
                );
```

2.实例

3.约束

(1) 创建资源计划中涉及的 plan、pool、group 必须已经创建;

(2) 若当前存在激活的 plan，不允许执行创建 directive 操作;

(3) 同一计划内一个资源消费组只能挂接到一个动态资源池;

(4) 同一计划内一个动态资源池则可挂接多个资源消费组;

(5) 任何计划内 default_consumer_group 必须关联动态资源池;

(6) 动态资源池中对应参数的值和不得高于所属静态资源池的参数值设定;

13.1.4.2 删除

1.句法

```
DROP RESOURCE DIRECTIVE <directive_name>;
```

2.实例

```
DROP RESOURCE DIRECTIVE directive1;
```

3.约束

（1）若当前存在激活的 plan，不允许执行创建、删除 directive 操作；

13.2 信息查询

13.2.1 要素定义信息查询

1. 查询 Consumer group 定义

(1) 句法

SELECT * FROM gbase.consumer_group;

(2) 图例

```
gbase> select * from gbase.consumer_group;
+-------------------+---------------------+-------------+
| consumer_group_id | consumer_group_name | comment     |
+-------------------+---------------------+-------------+
|               102 | group1              | test group1 |
|               104 | group2              | test group2 |
+-------------------+---------------------+-------------+
2 rows in set (Elapsed: 00:00:00.01)
```

2. 查询 consumer group 与 user 关联定义

(1) 句法

SELECT * FROM gbase.consumer_group_user;

(2) 图例

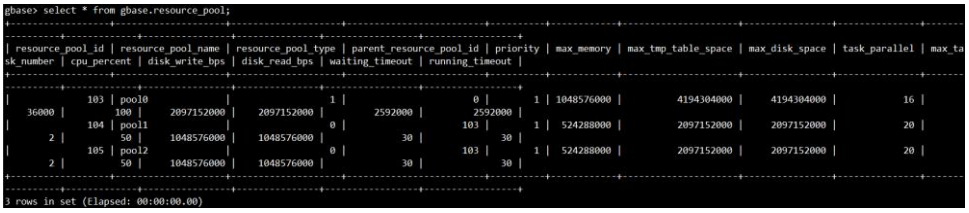
```
gbase> select * from gbase.consumer_group_user;
+-------------------+-----------+
| consumer_group_id | user_name |
+-------------------+-----------+
|               102 | user1     |
|               104 | user2     |
+-------------------+-----------+
2 rows in set (Elapsed: 00:00:00.01)
```

3. 查询 resource pool 定义

(1) 句法

```
SELECT * FROM gbase.resource_pool;
```

(2) 图例



```
gbase> select * from gbase.resource_pool;
+------------------+--------------------+--------------------+---------------------+----------+------------+--------------------+----------------+--------------+-----------+
| resource_pool_id | resource_pool_name | resource_pool_type | parent_resource_pool_id | priority | max_memory | max_tmp_table_space | max_disk_space | task_parallel | max_task_number | cpu_percent | disk_write_bps | disk_read_bps | waiting_timeout | running_timeout |
+------------------+--------------------+--------------------+---------------------+----------+------------+--------------------+----------------+--------------+-----------+
|              103 | pool0              |                    |                   1 |                0 |          1 | 1048576000 |     4194304000 |   4194304000 |        16 |          100 |     2097152000 |    2097152000 |         2592000 |         2592000 |
|              104 | pool1              |                    |                   0 |              103 |          1 |  524288000 |     2097152000 |   2097152000 |        20 |           50 |     1048576000 |    1048576000 |              30 |              30 |
|              105 | pool2              |                    |                   0 |              103 |          1 |  524288000 |     2097152000 |   2097152000 |        20 |           50 |     1048576000 |    1048576000 |              30 |              30 |
+------------------+--------------------+--------------------+---------------------+----------+------------+--------------------+----------------+--------------+-----------+
3 rows in set (Elapsed: 00:00:00.00)
```

```
gbase> select * from gbase.resource_pool\G
*************************** 1. row ***************************
       resource_pool_id: 103
     resource_pool_name: pool0
     resource_pool_type: 1
parent_resource_pool_id: 0
               priority: 1
             max_memory: 1048576000
    max_tmp_table_space: 4194304000
         max_disk_space: 4194304000
          task_parallel: 16
        max_task_number: 36000
            cpu_percent: 100
          disk_write_bps: 2097152000
           disk_read_bps: 2097152000
        waiting_timeout: 2592000
        running_timeout: 2592000
*************************** 2. row ***************************
       resource_pool_id: 104
     resource_pool_name: pool1
     resource_pool_type: 0
parent_resource_pool_id: 103
               priority: 1
             max_memory: 524288000
    max_tmp_table_space: 2097152000
         max_disk_space: 2097152000
          task_parallel: 20
        max_task_number: 2
            cpu_percent: 50
          disk_write_bps: 1048576000
           disk_read_bps: 1048576000
        waiting_timeout: 30
        running_timeout: 30
*************************** 3. row ***************************
       resource_pool_id: 105
     resource_pool_name: pool2
     resource_pool_type: 0
parent_resource_pool_id: 103
               priority: 1
             max_memory: 524288000
    max_tmp_table_space: 2097152000
         max_disk_space: 2097152000
          task_parallel: 20
        max_task_number: 2
            cpu_percent: 50
          disk_write_bps: 1048576000
           disk_read_bps: 1048576000
        waiting_timeout: 30
        running_timeout: 30
3 rows in set (Elapsed: 00:00:00.00)
```

4. 查询 resource plan 定义

(1) 句法

SELECT * FROM gbase.resource_plan;

(2) 图例

```
gbase> select * from gbase.resource_plan;
+------------------+---------------------+------------+
| resource_plan_id | resource_plan_name | comment    |
+------------------+---------------------+------------+
|              112 | plan1               | test plan1 |
+------------------+---------------------+------------+
1 row in set (Elapsed: 00:00:00.00)
```

5. 查询 resource directive 定义

(1) 句法

SELECT * FROM gbase.resource_plan_directive;

(2) 图例

```
gbase> select * from gbase.resource_plan_directive;
+----------------------------+-----------------+------------------+-----------------+-----------------+
| resource_plan_directive_name | resource_plan_id | consumer_group_id | resource_pool_id | comments        |
+----------------------------+-----------------+------------------+-----------------+-----------------+
| directive1                 |             112 |              102 |             108 | test directive1 |
| directive3                 |             112 |              104 |             110 | test directive3 |
| directive2                 |             112 |                1 |             108 | test directive2 |
+----------------------------+-----------------+------------------+-----------------+-----------------+
3 rows in set (Elapsed: 00:00:00.01)
```

13.2.2 运行期信息查询

1. 查询 active plan 设置

(1) 句法

SHOW VARIABLES like 'active_resource_plan';

（2）图例

```
gbase> show variables like 'active_resource_plan';
+-----------------------+-------+
| Variable_name         | Value |
+-----------------------+-------+
| active_resource_plan | plan1 |
+-----------------------+-------+
1 row in set (Elapsed: 00:00:00.01)
```

2. 查看动态资源池中任务

（1）句法

SELECT * FROM information_schema.PROCESSLIST WHERE resource _pool_name =’pool_name’;

辅助查询条件（逻辑与）：

<1> running_time = 0; //等待任务

（2）图例

```
gbase> select * from information_schema.PROCESSLIST \G;
*************************** 1. row ***************************
           ID: 53
       TASKID: 0
    SUBTASKID: 0
     THREADID: 33166
         USER: root
         HOST: 127.0.0.1:21731
           DB: NULL
      COMMAND: Query
   START_TIME: 2016-11-25 15:41:17
         TIME: 0
        STATE: checking permissions
    POOL_NAME: NULL
      POOL_ID: NULL
POOl_PRIORITY: NULL
 WAITING_TIME: NULL
 RUNNING_TIME: NULL
         LOCK: NULL
         WAIT: NULL
         INFO: NULL
        TRACE: NULL
```

3. 查看 Resource Pool 的资源使用情况

（1）句法

SHOW RESOURCE POOL USAGE ON {coordinators |

nodes} WHERE resource_pool_name = 'pool_name';

(2) 图例

```
gbase> show resource pool usage on coordinators where resource_pool_name='pool1';
+--------------+----------------+------------------+----------+---------------+---------------+
| NODE_NAME    | RESOURCE_POOL_ID | RESOURCE_POOL_NAME | PRIORITY | WAITING_TASKS | RUNNING_TASKS |
+--------------+----------------+------------------+----------+---------------+---------------+
| coordinator1 |              104 | pool1              |        1 |             0 |             0 |
| coordinator2 |              104 | pool1              |        1 |             0 |             0 |
| coordinator3 |              104 | pool1              |        1 |             0 |             0 |
+--------------+----------------+------------------+----------+---------------+---------------+
3 rows in set (Elapsed: 00:00:00.00)

gbase> show resource pool usage on nodes where resource_pool_name='pool1';
+-----------+----------------+------------------+----------+---------------+---------------+-----------+-----------+------------+---------------+---------------+
| NODE_NAME | RESOURCE_POOL_ID | RESOURCE_POOL_NAME | PRIORITY | RUNNING_TASKS | WAITING_TASKS | CPU_USAGE | MEM_USAGE | DISK_USAGE | DISK_WRITEIO | DISK_READIO |
+-----------+----------------+------------------+----------+---------------+---------------+-----------+-----------+------------+---------------+---------------+
| node1     |              104 | pool1              |        1 |             0 |             0 |         0 |         0 |          0 |             0 |             0 |
| node2     |              104 | pool1              |        1 |             0 |             0 |         0 |         0 |          0 |             0 |             0 |
| node3     |              104 | pool1              |        1 |             0 |             0 |         0 |         0 |          0 |             0 |             0 |
+-----------+----------------+------------------+----------+---------------+---------------+-----------+-----------+------------+---------------+---------------+
3 rows in set (Elapsed: 00:00:01.51)
```

13.2.3 统计信息查询

1. 查看动态资源池任务运行的资源使用历史信息

(1) 句法

SHOW RESOURCE POOL STATUS ON {coordinators | nodes} WHERE resource_pool_name = 'pool_name';

(2) 图例

```
gbase> show resource pool status on coordinators;
+--------------+----------------+------------------+----------+---------------+-----------------+-----------------+---------------------+
| NODE_NAME    | RESOURCE_POOL_ID | RESOURCE_POOL_NAME | PRIORITY | SERVIED_TASKS | WAITING_AVG_TIME | RUNNING_AVG_TIME | SAMPLE_TIME         |
+--------------+----------------+------------------+----------+---------------+-----------------+-----------------+---------------------+
| coordinator1 |              104 | pool1              |        1 |             0 |               0 |               0 | 2017-04-10 23:33:04 |
| coordinator1 |              105 | pool2              |        1 |             0 |               0 |               0 | 2017-04-10 23:33:04 |
| coordinator1 |              104 | pool1              |        1 |             0 |               0 |               0 | 2017-04-10 23:38:04 |
| coordinator1 |              105 | pool2              |        1 |             0 |               0 |               0 | 2017-04-10 23:38:04 |
| coordinator2 |              104 | pool1              |        1 |             0 |               0 |               0 | 2017-04-10 23:33:45 |
+--------------+----------------+------------------+----------+---------------+-----------------+-----------------+---------------------+
```

```
gbase> show resource pool status on nodes;
+-----------+----------------+------------------+----------+---------------+---------------+-----------+-----------+------------+---------------+---------------+---------------------+
| NODE_NAME | RESOURCE_POOL_ID | RESOURCE_POOL_NAME | PRIORITY | RUNNING_TASKS | WAITING_TASKS | CPU_USAGE | MEM_USAGE | DISK_USAGE | DISK_WRITEIO | DISK_READIO | SAMPLE_TIME         |
+-----------+----------------+------------------+----------+---------------+---------------+-----------+-----------+------------+---------------+---------------+---------------------+
| node1     |              104 | pool1              |        1 |             0 |             0 |         0 |         0 |          0 |             0 |             0 | 2017-04-10 23:33:04 |
| node1     |              105 | pool2              |        1 |             0 |             0 |         0 |         0 |          0 |             0 |             0 | 2017-04-10 23:33:04 |
| node1     |              104 | pool1              |        1 |             0 |             0 |         0 |         0 |          0 |             0 |             0 | 2017-04-10 23:38:04 |
| node1     |              105 | pool2              |        1 |             0 |             0 |         0 |         0 |          0 |             0 |             0 | 2017-04-10 23:38:04 |
+-----------+----------------+------------------+----------+---------------+---------------+-----------+-----------+------------+---------------+---------------+---------------------+
```

2. 查看动态资源池资源管控事件

(1) 句法

SHOW RESOURCE POOL EVENTS WHERE resource_pool_name = 'pool_name';

（2）图例

```
gbase> show resource pool events;
+--------------+---------+-----------+---------------------+---------+--------------------------+------------+--------------------------+
| NODE_NAME    | POOL_ID | POOL_NAME | EVENT_TIME          | TASK_ID | STATEMENT                | EVENT_TYPE | EVENT_DESCRIPTION        |
+--------------+---------+-----------+---------------------+---------+--------------------------+------------+--------------------------+
| coordinator1 |     106 | pool1     | 2016-11-28 12:36:33 |     123 | insert into b select * from b | Terminate  | Exceed max runtime       |
| coordinator1 |     106 | pool1     | 2016-11-28 12:36:43 |     123 | insert into c select * from c | Terminate  | Exceed max runtime       |
| coordinator1 |     106 | pool1     | 2016-11-28 12:37:09 |     123 | insert into a select * from a | Terminate  | Exceed max runtime       |
| coordinator1 |     106 | pool1     | 2016-11-28 12:37:57 |     123 | insert into a select * from a | Terminate  | Exceed max runtime       |
| coordinator1 |     106 | pool1     | 2016-11-28 12:37:58 |     123 | insert into b select * from b | Terminate  | Exceed max runtime       |
| coordinator1 |     106 | pool1     | 2016-11-28 12:37:59 |     179 | insert into c select * from c | reject     | wait the async lock timeout. |
+--------------+---------+-----------+---------------------+---------+--------------------------+------------+--------------------------+
6 rows in set (Elapsed: 00:00:00.03)
```

13.3 应用场景

13.3.1 加载和查询混合场景

1.背景介绍

有两个不同的用户:

UserLoad 是一个加载业务为主的用户，白天一般任务数目以及紧急度均要求不高，而晚上则会开启一定任务数以及希望能尽快完成加载;

UserSelect 是一个查询业务为主的用户,要求白天任务多,要求响应及时,而晚上则任务是减少。

这就需要白天和晚上对这两个用户所在的资源消费组分别挂接不同的动态资源池，并以此为规律进行不断切换，达到同一个用户在不同的时间段受到不同的控制，从而更合理使用资源。

2. 资源分配计划

由于 Gbase8a 集群的资源管理对内存的限制只作用于任务中聚合、连接等算子使用的内存，所以这里只假设每台 Gnode 数据机器预备留 10G 内存分配给算子 buffer，那么根据资源需求量，对这两个用户使用的资源池分配资源为:

白天 UserLoad 所对应的动态资源池分配 20% cpu，2G 内存;

白天 UserSelect 分配 80% cpu ，8G 内存;

晚上 UserLoad 分配 80% cpu，8G 内存;

晚上 UserSelect 分配 20% cpu，2G 内存;

3.实现步骤

可以创建两个 plan，分别针对白天和晚上两个不同时段做资源分配，通过切换 plan 达到此资源控制方式。

(1) 创建资源消费组并将用户挂接到对应的组

```
CREATE CONSUMER GROUP group_load comment = 'users for load';
CREATE CONSUMER GROUP group_select comment = 'users for select';
```

```
ALTER CONSUMER GROUP group_load ADD USER UserLoad;
ALTER CONSUMER GROUP group_select ADD USER UserSelect;
```

(2) **创建资源池**

<1> 静态资源池

```
CREATE RESOURCE POOL STATIC_POOL0(
CPU_PERCENT=100,
MAX_MEMORY=10000,
MAX_TEMP_DISKSPACE=10000,
MAX_DISK_SPACE=10000,
MAX_DISK_WRITEIO=1000,
MAX_DISK_READIO=1000) TYPE STATIC;
```

<2> 动态资源池

```
CREATE RESOURCE POOL HIGH_POOL(
PRIORITY=1,
CPU_PERCENT=80,
MAX_MEMORY=8000,
MAX_TEMP_DISKSPACE=5000,
MAX_DISK_SPACE=5000,
MAX_DISK_WRITEIO=600,
MAX_DISK_READIO=600,
MAX_ACTIVETASK=200,
TASK_MAX_PARALLEL_DEGREE=100,
TASK_WAITING_TIMEOUT=100000,
TASK_RUNNING_TIMEOUT=100000)
TYPE DYNAMIC BASE ON STATIC_POOL0;

CREATE RESOURCE POOL LOW_POOL(
PRIORITY=1,
CPU_PERCENT=20,
MAX_MEMORY=2000,
MAX_TEMP_DISKSPACE=5000,
MAX_DISK_SPACE=5000,
MAX_DISK_WRITEIO=400,
MAX_DISK_READIO=400,
MAX_ACTIVETASK=200,
```

```
TASK_MAX_PARALLEL_DEGREE=100,
TASK_WAITING_TIMEOUT=100000,
TASK_RUNNING_TIMEOUT=100000)
TYPE DYNAMIC BASE ON STATIC_POOL0;
```

(3) 创建资源计划

```
CREATE RESOURCE PLAN  PLAN_DAY  COMMENT = 'DAY PLAN';
CREATE RESOURCE PLAN  PLAN_NIGHT  COMMENT = NIGHT PLAN';
```

(4) 创建资源指令计划

```
CREATE RESOURCE DIRECTIVE DIRECTIVE1
(PLAN_NAME ='PLAN_DAY',
 POOL_NAME='HIGH_POOL',
 GROUP_NAME = 'GROUP_SELECT',
 COMMENT = 'SELECT USER RESOURCE USAGE ON DAY');

CREATE RESOURCE DIRECTIVE DIRECTIVE2
(PLAN_NAME ='PLAN_DAY',
 POOL_NAME = 'LOW_POOL',
 GROUP_NAME = 'GROUP_LOAD',
 COMMENT = 'LOAD USER RESOURCE USAGE ON DAY');

CREATE RESOURCE DIRECTIVE DIRECTIVE3

(PLAN_NAME = 'PLAN_DAY',

POOL_NAME='HIGH_POOL',
GROUP_NAME = 'DEFAULT_CONSUMER_GROUP',
COMMENT = 'OTHER USER RESOURCE USAGE ON DAY ');

CREATE RESOURCE DIRECTIVE DIRECTIVE4
(PLAN_NAME='PLAN_NIGHT',
POOL_NAME='HIGH_POOL',
GROUP_NAME = 'GROUP_LOAD',
COMMENT = ' LOAD USER RESOURCE USAGE ON NIGHT ');

CREATE RESOURCE DIRECTIVE DIRECTIVE5
( PLAN_NAME='PLAN_NIGHT',
```

```
   POOL_NAME='LOW_POOL',
   GROUP_NAME = 'GROUP_SELECT',
   COMMENT = ' SELECT USER RESOURCE USAGE ON NIGHT ');

CREATE RESOURCE DIRECTIVE DIRECTIVE6
(PLAN_NAME ='PLAN_NIGHT',
 POOL_NAME='LOW_POOL ',
 GROUP_NAME = 'DEFAULT_CONSUMER_GROUP',
 COMMENT = 'OTHER USER RESOURCE USAGE ON NIGHT');
```

(5) 激活 plan

```
<1> 白天 (假设早上 8:00)
SET GLOBAL ACTIVE_RESOURCE_PLAN =PLAN_DAY;
<2> 晚上 (假设晚上 20:00)
SET GLOBAL ACTIVE_RESOURCE_PLAN =PLAN_NIGHT;
```

13.3.2 白天跑查询，晚上跑批场景

这种场景和上述场景一致，只要按照用户划分好资源计划，即可按照上述方式实现。

13.3.3 高低写限速组场景

这种场景和上述场景一致，区别在于对磁盘读写 I/O 数值的设置，例如要限制低写限速组（例如：1M）和高写限速组（例如 50M/100M）的资源计划。

由于 RH6.x/SUSE11 等低版本的操作系统存在内核缺陷，在 ext 日志文件系统上，高限速组和低限速组表现出串行现象，高限速组最高只能达到低限速组峰值的 2 倍。

建议在 RH7.3/SUSE12 以上版本的操作系统中使用 I/O 限速功能。

13.3.4 全天加工和查询任务并行场景

1.背景

加工用户：UserA，UserB
查询用户：UserC，UserD,UserE
其他用户按照加工用户方式处理
资源分配需求：保障查询性能，控制加工消耗的性能

2. 实现步骤

（1）创建资源消费组并关联用户

```
CREATE CONSUMER GROUP group_process  comment = 'users for process';

CREATE CONSUMER GROUP group_select  comment = 'users for select';
ALTER CONSUMER GROUP group_process  add user UserA;
ALTER CONSUMER GROUP group_process  add user UserB;

ALTER CONSUMER GROUP group_select  add user UserC;
ALTER CONSUMER GROUP group_select  add user UserD;
ALTER CONSUMER GROUP group_select  add user UserE;
```

（2）创建资源池

<1> 静态资源池

```
CREATE RESOURCE POOL static_pool0(
 cpu_percent=100,
 max_memory=10000,
 max_temp_diskspace=10000,
 max_disk_space=10000,
max_disk_writeio=1000,
 max_disk_readio=1000) TYPE static;
```

<2> 动态资源池

```
CREATE RESOURCE POOL pool_select(
priority=1,
cpu_percent=80,
max_memory=6000,
max_temp_diskspace=5000,
max_disk_space=5000,
max_disk_writeio=600,
max_disk_readio=600,
```

```
max_activetask=200,
task_max_parallel_degree=100,
task_waiting_timeout=100000,
task_running_timeout=100000)
TYPE dynamic BASE ON static_pool0;

CREATE RESOURCE POOL pool_process(
priority=1,
cpu_percent=20,
max_memory=4000,
max_temp_diskspace=5000,
max_disk_space=5000,
max_disk_writeio=400,
max_disk_readio=400,
max_activetask=200,
task_max_parallel_degree=100,
task_waiting_timeout=100000,
task_running_timeout=100000)
TYPE dynamic BASE ON static_pool0;
```

(3) 创建资源计划

```
CREATE RESOURCE PLAN  resource_plan  comment = 'resource plan';
```

(4) 创建资源指令计划

```
CREATE RESOURCE DIRECTIVE directive1
(plan_name = 'resource_plan' ,
 pool_name = 'pool_select',
 group_name = 'group_select',
 comment = 'select user resource usage ');

CREATE RESOURCE DIRECTIVE directive2
(plan_name = 'resource_plan',
 pool_name = 'pool_process',
 group_name = 'group_process',
 comment = 'process user resource usage ');

CREATE RESOURCE DIRECTIVE directive3
```

```
(plan_name = 'resource_plan',
 pool_name = ' pool_process ',
 group_name = 'default_consumer_group',
 comment = 'other user resource usage ');
```

(5) 激活计划

```
SET GLOBAL active_resource_plan =resource_plan;
```

13.3.5 支持高级用户抽查场景

1.背景

集群中有

加工用户: UserA, UserB

查询用户: UserC, UserD,UserE

抽查用户: UserCheck

要求抽查用户 UserCheck 查询时能够得到最高的优先级，并且能够预留内存，磁盘 IO 资源给 UserCheck。

2.资源分配方案

(1) 建立一个专属动态资源池给这个用户;

(2) 设置 cpu_percent 为较高值;

(3) 分配必要的内存给这个用户;

3. 实现步骤

(1) 创建资源消费组并关联用户;

```
CREATE CONSUMER GROUP group_process  comment = 'users for process';
CREATE CONSUMER GROUP group_select  comment = 'users for select';
CREATE CONSUMER GROUP group_check  comment = 'users for check';

ALTER CONSUMER GROUP group_check  add user UserCheck;

ALTER CONSUMER GROUP group_process  add user UserA;
```

```
ALTER CONSUMER GROUP group_process  add user UserB;

ALTER CONSUMER GROUP group_select  add user UserC;
ALTER CONSUMER GROUP group_select  add user UserD;
ALTER CONSUMER GROUP group_select  add user UserE;
```

（2）创建资源池；

```
CREATE RESOURCE POOL static_pool0(
cpu_percent=100,
max_memory=10000,
max_temp_diskspace= 10000,
max_disk_space= 10000,
max_disk_writeio=1000,
max_disk_readio=1000) TYPE static;


CREATE RESOURCE POOL pool_check(

priority=1,
cpu_percent=99,
max_memory=4000,
max_temp_diskspace=50000,
max_disk_space=50000,
max_disk_writeio=600,
max_disk_readio=600,
max_activetask=200,
task_max_parallel_degree=100,
task_waiting_timeout=100000,
task_running_timeout=100000)
TYPE dynamic BASE ON static_pool0;

CREATE RESOURCE POOL pool_select(
priority=3,
cpu_percent=70,
max_memory=4000,
max_temp_diskspace=5000,
max_disk_space=5000,
max_disk_writeio=200,
```

```
max_disk_readio=200,
max_activetask=200,
task_max_parallel_degree=100,
task_waiting_timeout=100000,
task_running_timeout=100000)
TYPE dynamic BASE ON static_pool0;

CREATE RESOURCE POOL pool_process(
priority=3,
cpu_percent=30,
max_memory=2000,
max_temp_diskspace=5000,
max_disk_space=5000,
max_disk_writeio=200,
max_disk_readio=200,
max_activetask=200,
task_max_parallel_degree=100,
task_waiting_timeout=100000,
task_running_timeout=100000)
TYPE dynamic BASE ON static_pool0;
```

（3）创建资源计划；

```
CREATE RESOURCE PLAN  resource_plan  comment = 'resource plan';
```

（4）创建资源指令计划；

```
CREATE RESOURCE DIRECTIVE directive1
(plan_name = 'resource_plan',
 pool_name = 'pool_select',
 group_name = 'group_select',
 comment = 'select user resource usage ');

CREATE RESOURCE DIRECTIVE directive2
(plan_name = 'resource_plan',
 pool_name = 'pool_process',
 group_name = 'group_process',
 comment = 'process user resource usage ');
```

```
CREATE RESOURCE DIRECTIVE directive3
(plan_name = 'resource_plan',,
pool_name = 'pool_check',
group_name = 'group_check',
 comment ='check user resource usage ');

CREATE RESOURCE DIRECTIVE directive4
(plan_name = 'resource_plan',
 pool_name = ' pool_process ',
 group_name = 'default_consumer_group',
 comment = 'other user resource usage ');
```

(5) 激活计划;

```
SET GLOBAL active_resource_plan =resource_plan;
```

13.3.6 多租户隔离场景

1.背景

集群中有

用户组 High: UserA

用户组 Low: UserB

要求 High, Low 用户组 CPU 资源隔离, High 组占 90% CPU 资源, Low 组占 10% CPU 资源, 互不共享。

2.资源分配方案

(1) 建立两个静态资源池将 High, Low 用户组的 CPU 隔离;

(2) 每个静态资源池下建立一个动态资源池;

3. 实现步骤

(1) 创建资源消费组并关联用户;

```
CREATE CONSUMER GROUP group_high  comment = 'users for high';
CREATE CONSUMER GROUP group_low  comment = 'users for low';
```

```
ALTER CONSUMER GROUP group_high  add user UserA;
ALTER CONSUMER GROUP group_low  add user UserB;
```

（2）创建资源池；

```
CREATE RESOURCE POOL static_pool_high(
cpu_percent=90,
max_memory=10000,
max_temp_diskspace= 10000,
max_disk_space= 10000,
max_disk_writeio=1000,
max_disk_readio=1000) TYPE static;

CREATE RESOURCE POOL static_pool_low(
cpu_percent=10,
max_memory=10000,
max_temp_diskspace= 10000,
max_disk_space= 10000,
max_disk_writeio=1000,
max_disk_readio=1000) TYPE static;



CREATE RESOURCE POOL pool_high(

priority=1,
cpu_percent=100,
max_memory=4000,
max_temp_diskspace=50000,
max_disk_space=50000,
max_disk_writeio=600,
max_disk_readio=600,
max_activetask=200,
task_max_parallel_degree=100,
task_waiting_timeout=100000,
task_running_timeout=100000)
TYPE dynamic BASE ON static_pool_high;

CREATE RESOURCE POOL pool_low(
```

```
priority=1,
cpu_percent=100,
max_memory=4000,
max_temp_diskspace=50000,
max_disk_space=50000,
max_disk_writeio=600,
max_disk_readio=600,
max_activetask=200,
task_max_parallel_degree=100,
task_waiting_timeout=100000,
task_running_timeout=100000)
TYPE dynamic BASE ON static_pool_low;
```

(3) 创建资源计划;

```
CREATE RESOURCE PLAN  resource_plan  comment = 'resource plan';
```

(4) 创建资源指令计划;

```
CREATE RESOURCE DIRECTIVE directive1
(plan_name = 'resource_plan',
 pool_name = 'pool_high',
 group_name = 'group_high',
 comment = 'high user resource usage ');

CREATE RESOURCE DIRECTIVE directive2
(plan_name = 'resource_plan',
 pool_name = 'pool_low',
 group_name = 'group_low',
 comment = 'low user resource usage ');
```

(5) 激活计划;

```
SET GLOBAL active_resource_plan =resource_plan;
```

(6) 限制和问题;

由于 RH6.x/SUSE11 等低版本的操作系统存在内核缺陷,可能会导致大负载的情况下, 操作系统宕机。

建议在 RH7.3/SUSE12 以上版本的操作系统中使用多静态资源池控制。

13.4 磁盘 IO 控制

13.4.1 DC 同步 I/O 控制

通过参数_gbase_dc_sync_size(单位:字节)控制,当列数据文件在 Buffer 中未刷出的 DC 大小超过指定值时,执行同步写 I/O 操作。此参数默认值为 1TB,此时 I/O 操作将由系统周期回写机制完成。此参数最小值为 0,表示每次受控 SQL 都调用同步 I/O 操作。

13.4.2 磁盘 I/O 读写速率上限控制

限定每个资源池中所有受控 SQL 总体磁盘 I/O 读、写带宽使用上限;

读、写分别限制。

13.5 集群层显示 SQL 在各节点资源使用命令

目前资源管理通过集群层命令或资源管理表可以看到 SQL 的执行状态、时间等信息,单机层各执行节点可看到下发 SQL 的资源使用情况,但在集群层无法看到 SQL 语句在单机层各执行节点资源使用情况。

为实现该功能,在集群层数据库 information_schema 中添加两张系统表:GNODES_TASK_INFORMATION,COORDINATORS_TASK_INFORMATION。

1) COORDINATORS_TASK_INFORMATION 系统表存储 SQL 语句在集群层执行相关信息;

2) GNODES_TASK_INFORMATION 系统表中存储 SQL 语句在单机层执行使用资源信息。

用户可以在集群层通过这两张系统表,查看 SQL 语句在集群层的执行信息

以及在单机层执行所使用资源信息。

13.5.1 查看 SQL 在单机层执行使用资源信息

DBA 用户需要输入指定规格的 SQL 语句，规格如下：

SELECT * FROM information_schema.GNODES_TASK_INFORMATION;

information_schema. GNODES_TASK_INFORMATION

属性名称	类型	说明
NODE_NAME	varchar(64)	data 节点名称（node1,node2,...）
ID	bigint(4)	Session ID
SUBTASKID	bigint(4)	子任务 ID
TASKID	bigint(4)	任务 ID
THREADID	bigint(4)	线程 ID
USER	varchar(16)	执行用户
HOST	varchar(64)	执行节点
DB	varchar(64)	所属数据库
COMMAND	varchar(16)	SQL 类型
START_TIME	timestamp	开始执行时间
TIME	int(7)	执行时间
STATE	varchar(64)	SQL 执行状态
RESOURCE_POOL_ID	bigint(21) unsigned	动态资源池编号
RESOURCE_POOL_NAME	varchar(64)	动态资源池名称

RESOURCE_POOL_PRIORITY	bigint(21) unsigned	动态资源池优先级
RUNNING_TIME	bigint(21) unsigned	任务运行时间
WAITING_TIME	bigint(21) unsigned	任务等待时间
PARALLEL_DEGREE	bigint(21) unsigned	并行度
CPU_USAGE	bigint(21) unsigned	CPU 使用
MEM_USAGE	bigint(21) unsigned	内存使用
TEMP_DISKSPACE_SORT	bigint(21) unsigned	sort 临时磁盘空间使用量
TEMP_DISKSPACE_JOIN	bigint(21) unsigned	join 临时磁盘空间使用量
TEMP_DISKSPACE_AGGR	bigint(21) unsigned	aggr 临时磁盘空间使用量
TRACE	longtext	Trace(debug 级)信息
INFO	longtext	执行的 SQL 语句

13.5.2 查看 SQL 在集群层执行信息

DBA 用户需要输入指定规格的 SQL 语句，规格如下：

SELECT * FROM information_schema.
COORDINATORS_TASK_INFORMATION;

information_schema. COORDINATORS_TASK_INFORMATION

属性名称	类型	说明
COORDINATOR_NAME	varchar(64)	coordinator 节点名称
ID	bigint(4)	Session ID
SUBTASKID	bigint(4)	子任务 ID
TASKID	bigint(4)	任务 ID
THREADID	bigint(4)	线程 ID
USER	varchar(16)	执行用户
HOST	varchar(64)	执行节点
DB	varchar(64)	所属数据库
COMMAND	varchar(16)	SQL 类型
START_TIME	timestamp	开始执行时间
TIME	int(7)	执行时间
STATE	varchar(64)	SQL 执行状态
RESOURCE_POOL_ID	bigint(21) unsigned	动态资源池编号
RESOURCE_POOL_NAME	varchar(64)	动态资源池名称
RESOURCE_POOL_PRIORITY	bigint(21) unsigned	动态资源池优先级

RUNNING_TIME	bigint(21) unsigned	任务运行时间
WAITING_TIME	bigint(21) unsigned	任务等待时间
LOCK	longtext	集群锁
WAIT	longtext	显示已加的需要等待的互斥锁
TRACE	longtext	Trace(debug 级)信息
INFO	longtext	执行的 SQL 语句

13.5.3 查看 SQL 在集群执行信息

示例：查询 SQL 语句在集群执行信息

```
select gc.coordinator_name, gc.taskid,gn.node_name, gn.cpu_usage,gc.info from
coordinators_task_information gc join gnodes_task_information gn on
gc.taskid=gn.taskid;
+-----------------+--------+-----------+-----------+-------------------------
-----+
| coordinator_name | taskid | node_name | cpu_usage | info
|
+-----------------+--------+-----------+-----------+-------------------------
-----+
| coordinator2     |    122 | node1     |       100 | insert into t select * from
t |
| coordinator2     |    122 | node2     |        90 | insert into t select * from
t |
| coordinator2     |    122 | node3     |        96 | insert into t select * from
t |
+-----------------+--------+-----------+-----------+-------------------------
-----+
```

13.6 注意事项

13.6.1 系统 cgconfig 服务

1、 不要修改 cgconfig 服务的配置文件；

2、 不要随意进入 cgconfig 子系统挂载点并保持进入状态，避免资源管理功能异常；

3、 不要进入 cgconfig 子系统 mount 点下做操作；

4、 不要手动重启、关闭 cgconfig 服务；

13.6.2 受控 SQL

GBase 8a MPP 资源管理设计目标是有效的控制资源的消耗，达到避免任务间争抢和一些资源限制，优先级等功能。同时在受资源管理的操作类型上做了限定，避免小而频的 sql 占用任务数导致的系统资源不能充分利用

受控 SQL 种类如下：

- Select 查询操作
- Create As Select 操作
- Load 操作
- Create index (hash、fulltext) 操作
- create as select 操作
- Update Index (hash、fulltext) 操作
- Alter table add Index (hash、fulltext) 操作
- Insert Select 操作
- Update 操作
- Merge 操作
- Delete 数据操作

- Call 操作

特别声明:

1)由于 DDL 操作本身不响应中断，因此，以上操作中的 DDL 操作（create index， alter table add index）不支持运行时间超时处理。

2)由于集群服务程序所执行的 SQL 任务与拆解下发给单机服务程序的 SQL 任务并不能一一对应，存在差异。这种实现方式就有可能出现以下现象:

集群层 SQL 任务不属于受控 SQL，而经过拆解下发给单机的 SQL 任务中却存在受控 SQL。这也就导致了资源监控查询过程中发现集群层非受控 SQL 任务在单机层上受控的现象。如 rebalance 操作。

这种现象目前属于正常现象，是集群实现机制使然。符合预期。

3)除了受控 sql 种类中控制的 DDL，其他的 DDL 均不受任务数管理限制，例如: create table , alter table, drop table, create user 等。

13.6.3 高、低优先级用户使用约束

资源管理在如下使用情景下，高优先级用户读 DC 的过程会受到低优先级用户的影响，导致高、低优先级用户执行性能相近:

1、使用相同并行度;

2、同时采用相同（类似）的 SQL 访问同一张表的相同列的数据，且 IO 是该 SQL 的主要耗时（如果算子计算是 SQL 主要耗时则影响程度会降低);

3、访问的数据是冷数据(如果部分数据是热数据，则随着热数据占比重的增加，读 DC 数据的影响程度降低);

在完全满足条件 2、3 的情况下，调整并行度，可以减少高低优先级相互之间的影响。

14 支持 Kerberos 安全认证

Hadoop 1.0.0 以上版本加入了 Kerberos 认证机制，Kerberos 实现了机器级别的安全认证。集群运行时，集群内的节点使用密钥得到认证，只有通过认证的节点才能正常使用。针对在 HDFS 中集成了 Kerberos 安全认证的系统，需要在所有 GBase 8a 集群节点部署了 Kerberos 客户端安装包，并正确配置 Kerberos 客户端环境。

14.1 Kerberos 客户端安装与配置

Kerberos 客户端的安装与配置主要分为以下几个步骤：

- 在 GBase 8a 的所有集群节点上，安装 Kerberos 客户端安装包，再将 Kerberos 客户端配置文件/etc/krb5.conf 由 KDC 服务器复制到所有集群节点上的/etc 目录下；
- 将 Kerberos 认证密钥文件 keytab 复制到所有节点指定目录， coor 节点目录为/opt/gcluster/config，data 节点目录为 /opt/gnode/config。
- 将 HTTPS 的 CA 根证书文件追加到所有节点的根证书文件中。

coor 节点根证书文件为/opt/gcluster/config/ca-bundle.crt

data 节点根证书文件为/opt/gnode/config/ca-bundle.crt

其中 CA 根证书文件中包含一个或多个证书，证书格式如下图所示：

```
-----BEGIN CERTIFICATE-----
MIIDlzCCAn+gAwIBAgIJAK3WBMvGgYKRMA0GCSqGSIb3DQEBBQUAMGIxCzAJBgNV
BAYTAkNOMRAwDgYDVQQIDAdUaWFuamluMRAwDgYDVQQHDAdUaWFuamluMQ4wDAYD
VQQKDAVHQkFTRTEMMAoGA1UECwwDRE1EMREwDwYDVQQDDAhnYmFzZS5jbjAeFw0x
NzAyMjMxMjI3MjBaFw00NDA3MTAxMjI3MjBaMGIxCzAJBgNVBAYTAkNOMRAwDgYD
VQQIDAdUaWFuamluMRAwDgYDVQQHDAdUaWFuamluMQ4wDAYDVQQKDAVHQkFTRTEM
MAoGA1UECwwDRE1EMREwDwYDVQQDDAhnYmFzZS5jbjCCASIwDQYJKoZIhvcNAQEB
BQADggEPADCCAQoCggEBAMdMsY4710zDeRWV6f18a01l31xA/Gn66NKSoNT1rPDh
zWkvijuXbcY8kfGMjJLGPY89ohiVZUAJo048ejP1hCbN/IL0FBnqg/rGjflRdwW9
xCjbSRMQ/fZ5PSjL3IYZejDeLte6F4+MNn11H2QY02IkdIU7M7q6Aat5TwtGJovh
zASdHTwfLULi/4aq0d2YnSyRf3L4BsNtwWSrXkaocx76qCTRc+a1B5ylDwCdSQ53
Od5vshX/kc3bAqccLalyqhfDMHhhDcx7+H75PTGPMyNSJg6dVdQVWOOwOdtKPg6F
3FvdBTeeg+fYxYqXEnw7mxX+P7sPdxS17R4btN4I7bkCAwEAAaNQME4wHQYDVR0O
BBYEFGdlLFv7zIx/RdeJwslPCC+hpT45MB8GA1UdIwQYMBaAFGdlLFv7zIx/RdeJ
wslPCC+hpT45MAwGA1UdEwQFMAMBAf8wDQYJKoZIhvcNAQEFBQADggEBACkg9qs6
Lf9nw5+xnJxq5FRO+T9mDqv3AwnM5aL3wdOdwsNjE2NU0QEfS7wz96p74em87goJ
uWbKy4WDZeKXGXbOju07KzI6GaFXlGKtMv8+p9E6UVh/GXr4mtYuYbwPgk5g1lxn
XZhkr1Sk70PhZWIZH1ugSspqJzoS0XjB8ObOTkr+WgNnCW5LwOq2QyZcb4wliaoG
JTCN7cCMkS8Sgg4Nkyy11gNmZj7r22He0GZ+43WLTrgieWOtZ0AvcYqoLcRGoRAI
23kHZfClG2IknfcLlDeBUM+9Q9z1/lo39X0CjFIlmdbZ5y8Uw5HOFjx5rszMxZ9q
/maTGcgV7+KtfTE=
-----END CERTIFICATE-----
```

14.2 集群扩容影响

支持 Kerberos 认证的集群版本在执行扩容后，管理员需要执行 11.1 Kerberos 客户端安装与配置，手工完成 Kerberos 客户端环境部署。

14.3 集群升级影响

从不支持 kerberos 认证的集群版本升级到支持 Kerberos 的集群版本，管理员需要执行 11.1 Kerberos 客户端安装与配置，手工完成 Kerberos 客户端环境部署。

14.4 集群节点替换工具

集群节点替换工具在同步文件时，需要将/opt/gcluster/config 或 /opt/gnode/config 下扩展名为.kt 和.pem/.crt/.cer/.crl 的文件同步到被替

换节点。

14.5 加载/导出 HDFS 文件影响

若 HDFS 中集成了 Kerberos 安全认证，GBase 8a 在执行加载或导出操作执行需完成如下配置：

- 设置 gbase_hdfs_auth_mode=kerberos，指定使用 Kerberos 认证方式连接 HDFS。
- 设置 gbase_hdfs_protocol=http/https/rpc，指定使用 HTTP/HTTPS/RPC 协议连接 HDFS。
- 设置 gbase_hdfs_principal="xxx"，指定 kerberos 认证主体。
- 设置 gbase_hdfs_keytab='xxx'，指定 keytab 文件路径。

同时需要注意：

- HDFS 的 HTTP 端口号默认为 50070，HTTPS 端口号默认为 50470，RPC 端口号默认为 9000，三种协议的端口不同，在加载或导出 SQL 的 URL 中的端口需要与指定的协议一致。
- 当使用 HTTPS 协议连接 HDFS 时，因为客户端需要使用 CA 根证书对 HTTPS 地址进行验证，所以在加载或导出 SQL 的 URL 中，指定的 HDFS NameNode 的主机名（或地址）必须与 CA 签名的主机名（或地址）完全相同。
- 当不指定 gbase_hdfs_keytab 参数值或指定的参数值为空字符串时，将使用 gbase_hdfs_principal 推定 keytab 文件名，此时应将 keytab 文件复制到 config 目录下，keytab 文件的名称应与 gbase_hdfs_principal 参数值对应，请参考《GBase 8a MPP Cluster 配置手册》。例如：

set gbase_hdfs_principal='gbase/namenode@HADOOP.COM'

则 config 目录下 keytab 文件名应为：gbase_namenode.kt。

- 由于 hadoop 和 kerberos 对 dns 解析依赖程度很高，需要 dns 支持正向（forward）和反向（reverse）查找，在 kerberos 认证环境中在加载和导出语句的 URL 中推荐使用主机名，而不建议使用 IP 地址。

15 密钥证书管理

密钥证书的管理需要管理员用户通过 SQL 方式处理，针对密文密钥用户需要牢记口令，系统不记录用户口令；

15.1 证书存储位置

证书存放在 config 目录，集群环境下，gnode 与 gcluster 都会生成相同密钥证书文件；

gnode：$GBASE_BASE/config/encryption.crt

gcluster：$GCLUSTER_BASE/config/encryption.crt

15.2 语法描述

15.3 创建证书

CREATE ENCRYPTION CERTIFICATE IDENTIFIED BY ’password’ [CONTENT ‘密钥内容’]

说明：

1) 该语法可创建明文、密文密钥证书，

- 如果 password 为空，则创建明文密钥证书，不需要口令；
- 如果 password 非空，则创建密文密钥证书，需要口令；

2) CONTENT 是密钥内容关键字，可选项，如果不指定该关键字，则创建时由系统自动生成密钥；如果指定，则需要用户手动输入密钥，内容不做限制，最大支持 128 字节；

【注】

1） 明文密钥创建完成后,即可对加密列做 dml 操作;密文密钥还需要 open 操作，方可对加密列做 dml 操作;

2） 密钥证书只能创建一次，不能重复创建;

3） 密文密钥，须用户牢记口令，系统不记录口令;

15.3.1 打开/关闭证书

根据口令打开密文密钥证书

ALTER ENCRYPTION CERTIFICATE OPEN IDENTIFIED BY ‘password’

关闭密钥证书

ALTER ENCRYPTION CERTIFICATE CLOSE;

【注】关闭密钥后，所有对加密列的 dml 操作将会失效;

15.3.2 显示证书状态

SELECT * FROM INFORMATION_SCHEMA.ALL_ENCRYPTION_CERTIFICATE_STATUS;

说明: 显示所有节点的密钥证书状态

```
gbase> SELECT * FROM INFORMATION_SCHEMA.ALL_ENCRYPTION_CERTIFICATE_STATUS;
+---------------+------------+----------+-------------+
| NODE_NAME     | IS_CREATE  | KEY_TYPE | OPEN_STATUS |
+---------------+------------+----------+-------------+
| coordinator1  | NO         |        0 | OFF         |
| node1         | NO         |        0 | OFF         |
| node1         | NO         |        0 | OFF         |
+---------------+------------+----------+-------------+
3 rows in set
```

15.3.3 修改证书口令

ALTER ENCRYPTION CERTIFICATE IDENTIFIED BY ‘old_pwd’ TO ‘new_pwd’

说明：修改密文密钥口令，old_pwd、new_pwd 均非空；

15.3.4 明文、密文密钥转换

ALTER ENCRYPTION CERTIFICATE IDENTIFIED BY ‘’ TO ’password’

说明：明文密钥转换为密文密钥，password 不能为空；

ALTER ENCRYPTION CERTIFICATE IDENTIFIED BY ‘password’ TO ‘’

说明：密文密钥转换为明文密钥，password 不能为空；

15.4 集群环境配置

在 gnode 节点 config 目录下的配置文件增加 encrypt_server_host，encrypt_server_port 参数，用于密文加密情况下，集群 gnode 节点重启服务后主动向 server 端获取密钥数据；

encrypt_server_host：指向 gcluster 的主机 IP，可以多个，用逗号分隔；

encrypt_server_port：指向 gcluster 的 server port（默认 5258）；

【注】如果整个集群都重启的话，针对密文方式，需要手动执行 open（打开密钥）操作方可对加密列做相关 dml 操作；

16 元数据管理

16.1.1 限制表实例数量

参数名：_gbase_express_table_limit

数值范围：最小值 16，最大值 1024*1024，默认值 16*1024

说明：当引擎层打开表数量达到参数设定的值时，不会立即触发清理动作。而是由后台线程每 5 秒检测一次，如果达到上限才会触发清理动作。

16.1.2 限制表实例元数据总大小

参数名：_gbase_express_table_metadata_limit

数值范围：gnode 下默认值为 temp 堆大小一半，gcluster 下默认值为 1GB，最小值最大值不限，配置文件中可使用 K/M/G 方式设置

说明：当引擎层打开表实例的元数据大小之和达到参数设定的值时，不会立即触发清理动作。而是由后台线程每 5 秒检测一次，如果达到上限才会触发清理动作。

16.1.3 超出限制后对 m_tables 清理比例

参数名：_gbase_express_table_clear_rate

数值范围：最大值 100（表示 100%），最小值 1（表示 1%），默认值 10（表示 10%）

说明：

1）表数量淘汰机制：当触发清理动作时每次清理多少比例的表对象实例。此比例是当前内存中表实例总数的比例。

2）元数据大小淘汰机制：每次清理多少比例的元数据大小。次比例是当

前内存中所有表实例的元数据大小之和的比例。

16.1.4 状态监测

有两个状态信息可以查询:

express_cached_tables 表示当前内存中有多少个表实例。

express_cached_metadata 表示当前内存中所有表实例的元数据大小。

```
gbase> show variables like '%express_table%';
+-----------------------------------+------------+
| Variable_name                     | Value      |
+-----------------------------------+------------+
| _gbase_express_table_clear_rate   | 10         |
| _gbase_express_table_limit        | 16384      |
| _gbase_express_table_metadata_limit | 1073741824 |
+-----------------------------------+------------+
3 rows in set (Elapsed: 00:00:00.01)

gbase> show status like '%express_cache%';
+------------------------+-------+
| Variable_name          | Value |
+------------------------+-------+
| express_cached_metadata | 5140  |
| express_cached_tables   | 1     |
+------------------------+-------+
2 rows in set (Elapsed: 00:00:00.00)
```

17 图形化管理工具

17.1 企业管理器

GBase 8a MPP Cluster 管理工具是 GBase 提供的一种新的集成环境，用于访问、控制和管理 GCluster 集群环境。GBase 8a MPP Cluster 管理工具将一组多样化的图形工具与多种功能齐全的脚本编辑器组合在一起，可为各种技术级别的开发人员和管理员提供对 GCluster 集群环境的访问功能。

GBase 8a MPP Cluster 管理工具通过 JDBC Driver 和集群环境进行通讯。

使用 GBase 8a MPP Cluster 管理工具可以完成如下工作：

- 管理单个集群环境和多个集群环境。
- 管理单个集群环境中的单个和多个集群节点服务器。
- 通过注册集群节点功能，实现集群的动态扩展。
- 可视化管理集群环境中的数据库、表、索引、视图、存储过程和函数等数据对象。
- 可视化集群环境创建用户、编辑用户和删除用户。
- 可视化查看集群环境的日志。
- 管理集群环境数据表中的数据记录。

关于企业管理器的安装，使用的内容介绍，请查看《GBase 8a MPP Cluster 管理工具手册》。

17.2 集群监控工具

统一监控系统是南大通用数据技术股份有限公司开发的 GBase 8a MPP Cluster 的组成部分。提供可信的监控数据，及时的报警功能，直观的趋势展示，可靠的数据分布视图和数据库连接线程的状态展示。

统一监控系统主要监控单个或多个 GBase 8a MPP Cluster 部署环境中，集群节点 Server 的运行状态，资源利用情况、网络通讯情况等信息可使用户对集群环境下的 Server 的运行状态及其系统资源进行有效地监视，能够为用户监控集群及其集群点的运行情况提供可靠的依据。

关于统一监控系统工具的安装、使用等内容，请查看《统一数据平台监控与运维系统用户手册》。

GBASE

南大通用数据技术股份有限公司
General Data Technology Co., Ltd.

微博二维码　微信二维码

技术支持热线：400-013-9696