

| 达梦技术丛书

DM8 大规模并行处理 MPP



前言

概述

本文档主要介绍 DM MPP 集群的系统特性、基本概念、实现原理，以及如何搭建 DM MPP 集群并使用等。

读者对象



本文档主要适用于 DM 数据库的：

- 数据库管理员
- 开发工程师
- 测试工程师
- 技术支持工程师

通用约定

在本文档中可能出现下列标志，它们所代表的含义如下：

表 0.1 标志含义

标志	说明
 警告：	表示可能导致系统损坏、数据丢失或不可预知的结果。
 注意：	表示可能导致性能降低、服务不可用。
 小窍门：	可以帮助您解决某个问题或节省您的时间。
 说明：	表示正文的附加信息，是对正文的强调和补充。

在本文档中可能出现下列格式，它们所代表的含义如下：

表 0.2 格式含义

格式	说明
宋体	表示正文。
Courier new	表示代码或者屏幕显示内容。
粗体	表示命令行中的关键字（命令中保持不变、必须照输的部分）或者正文中强调的内容。标题、警告、注意、小窍门、说明等内容均采用粗体。
<>	语法符号中，表示一个语法对象。
::=	语法符号中，表示定义符，用来定义一个语法对象。定义符左边为语法对象，右边为相应的语法描述。
	语法符号中，表示或者符，限定的语法选项在实际语句中只能出现一个。
{ }	语法符号中，大括号内的语法选项在实际的语句中可以出现 0...N 次 (N 为大于 0 的自然数)，但是大括号本身不能出现在语句中。
[]	语法符号中，中括号内的语法选项在实际的语句中可以出现 0...1 次，但是中括号本身不能出现在语句中。
关键字	关键字在 DM_SQL 语言中具有特殊意义，在 SQL 语法描述中，关键字以大写形式出现。但在实际书写 SQL 语句时，关键字既可以大写也可以小写。

访问相关文档

如果您安装了 DM 数据库，可在安装目录的“\doc”子目录中找到 DM 数据库的各种手册与技术丛书。

您也可以通过访问我们的网站 www.dameng.com 阅读或下载 DM 的各种相关文档。

联系我们

如果您有任何疑问或是想了解达梦数据库的最新动态消息，请联系我们：

网址：www.dameng.com

技术服务电话：400-991-6599

技术服务邮箱：dmtech@dameng.com

目录

1 引言	1
2 概述	2
2.1 系统架构	2
2.2 原理概述	4
2.3 系统特性	4
3 基本概念与原理	6
3.1 基本概念	6
3.1.1 执行节点 EP	6
3.1.2 主、从 EP	6
3.1.3 数据分布	6
3.1.4 MAL 系统	7
3.1.5 全局连接与本地连接	8
3.2 MPP 并行执行计划	8
3.2.1 并行计划相关操作符	8
3.2.2 并行计划的生成	9
3.2.3 数据分布与并行执行计划	12
3.2.4 并行计划执行流程	12
3.2.5 DDL 语句分发	14
3.3 相关配置文件	14
3.3.1 dm.ini MPP 相关配置项	14
3.3.2 dmmal.ini 配置项	16
3.3.3 dmmpp.ctl	17
4 DM MPP 环境搭建与使用	18
4.1 制定合适的 DM MPP 方案	18
4.2 DM MPP 环境搭建	19
4.2.1 系统规划	19
4.2.2 配置 dm.ini	19
4.2.3 配置 dmmal.ini	20
4.2.4 配置 dmmpp.ctl	21
4.2.5 运行 MPP	21
4.3 建立分布表	22
4.4 快速数据装载	24
4.5 停止 MPP 系统	24

4.6 MPP 相关系统过程与函数	25
4.7 MPP 下系统过程与系统视图常见用法	28
4.8 使用说明.....	29
5 DM MPP 系统管理.....	30
5.1 使用 DEM 部署与监控 DM MPP	30
5.1.1 使用 DEM 部署 DM MPP	31
5.1.2 使用 DEM 监控 DM MPP 运行	37
5.2 MPP 系统动态扩容	38
5.2.1 环境准备	39
5.2.2 动态增加节点	41
5.2.3 数据重分发	44
5.3 MPP 环境的 DDL 克隆与还原	48
6 DM MPP 主备系统	50

1 引言

达梦大规模并行处理 MPP (DM Massively Parallel Processing, 缩写 DM MPP) 是基于达梦数据库管理系统研发的完全对等无共享式集群组件, 支持将多个 DM 数据库实例组织为一个并行计算网络, 对外提供统一的数据库服务。

在海量数据分析的应用场景中, 经常会遇到以下问题:

- 大量的复杂查询操作需要较高的系统性能支持;
- 数据库响应能力受到硬件的束缚;
- 小型机虽然能在垂直领域提供较好的单个节点性能, 但是价格较高。

为了支持上述海量数据存储和处理、高性价比等方面的需求, 提供高端数据仓库解决方案, 达梦数据库提供了大规模并行处理 MPP 架构, 以极低的成本代价, 为客户提供业界领先的计算性能。

本文档主要介绍 DM MPP 的概念与实现原理, 如何搭建与使用 DM MPP 以及 DM MPP 系统的运行管理。

2 概述

本章简单介绍 DM MPP 的系统架构、原理及功能特性。

2.1 系统架构

当前主流的数据库系统架构有完全共享、共享存储、完全不共享和完全对等不共享几种。

其中完全共享体系如 SMP 服务器，局限于单节点服务器，通常价格比较昂贵，其扩展性和性能受到相应的限制。

共享存储体系允许系统带有多个服务器实例，这些实例与共享存储设备相连。这种体系可实现多机并行，保证系统的高可用性，但需要通过一个数据管道将所有 I/O 信息过滤到共享存储子系统，对硬件的要求较高，且并非高性能解决方案。

与此相比，基于硬件的数据仓库平台一般采用完全无共享体系。在这种体系下，通讯功能部署在一个高宽带网络互连体系上，用户通过一个主控制节点执行并行查询。该体系的一个重要优势就是每个节点都有一个通往本地磁盘的独立通道，不但简化了体系，还提供良好的扩展性。但主控节点的存在使得系统规模扩张时主控节点可能成为系统瓶颈，且主控节点一旦发生故障这个系统将无法提供服务。

DM MPP 采用的完全对等无共享体系架构，结合了完全无共享体系的优点，在此基础上又前进了一步，不采用增加主控制节点来协调所有并行处理的主从式方法，而是各个节点完全对等，更进一步简化了体系的实现，也消除了系统可能存在的主节点瓶颈问题。

图 2.1 是这几种数据库系统架构的整体结构示意图。

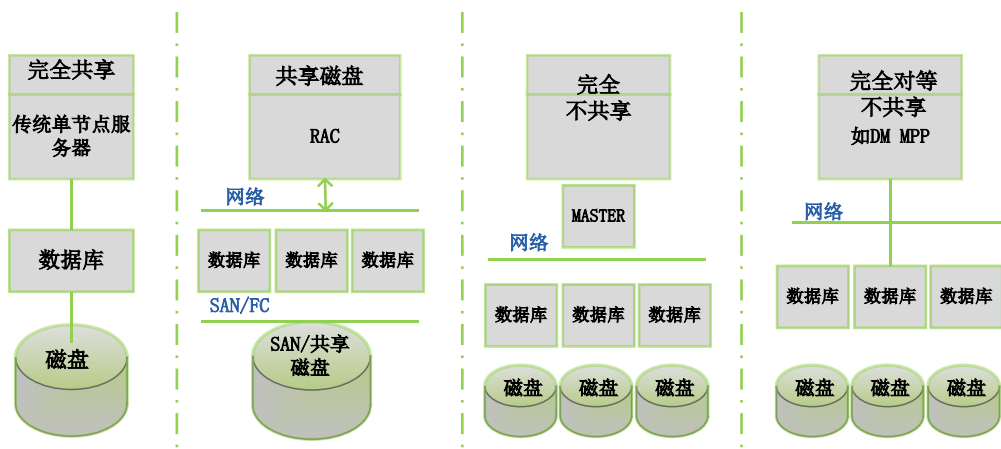


图 2.1 主流数据库系统架构示意图

表 2.1 则总结了这几种数据库系统架构各自的优缺点。

表 2.1 数据库主流系统架构特点比较

架构名称	特点
完全共享	局限于单节点服务器，价格昂贵，扩展性、性能受限
共享磁盘	允许多个服务器实例共享存储设备，可有效解决单实例负载问题，具有一定的扩展性，但当节点增加到一定程度以后，由于对 I/O 资源、锁资源等的激烈竞争，反而导致性能的下降，扩展性和性能在系统规模变大时受限。同时共享磁盘等硬件成本也十分昂贵
完全不共享	部署在高速网络，各节点相对独立，无共享 I/O，扩展性和性能良好，缺点是系统中有一个主控节点，系统规模扩充时可能成为瓶颈，主控节点无备份，容易形成单点故障
完全对等不共享	继承了完全不共享架构的优点，且各节点完全对等，不需要专用硬件，不存在主控节点，消除了潜在瓶颈以及单节点故障问题

DM MPP 采用完全对等不共享架构，具体的系统架构如图 2.2 所示。

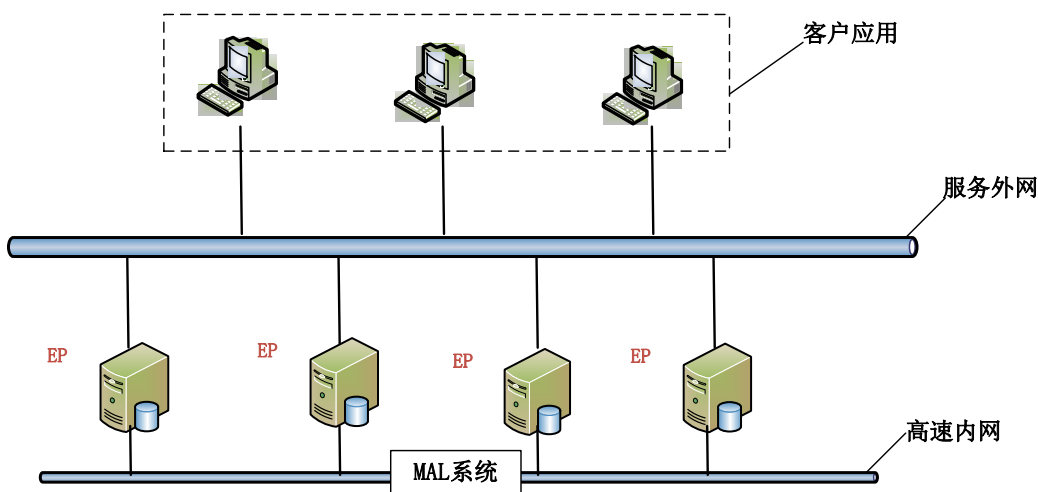


图 2.2 DM MPP 系统架构图

DM MPP 中的每一个 DM 数据库服务器实例作为一个执行节点，简称 EP。客户端可连接任意一个 EP 节点进行操作，所有 EP 对客户来说都是对等的。

DM MPP 系统内每个 EP 只负责自身部分数据的读写，执行计划在所有 EP 并行执行，能充分利用各 EP 的计算能力及发挥各 EP 独立存储的优势。数据只在必要时通过 DM 的高速邮件 MAL 系统在 EP 间传递。当通信代价占整体执行代价的比例较小时，更能体现大规模并行处理的优势，随着系统规模的扩大，并行支路越多，优势越明显。

2.2 原理概述

在 DM MPP 中，数据根据用户指定的分布规则分布在不同的 EP 上。MPP 的核心在于对用户请求的并行执行，其执行流程可简单描述如下：

1. 用户选择一个 EP 登录，此时该 EP 就是此用户的主 EP，集群中的其余 EP 都是此用户的从 EP；
2. 主 EP 接受用户的 SQL 请求，并生成并行执行计划；
3. 主 EP 将计划打包后分发给其他从 EP；
4. 各 EP 并行执行；
5. 主 EP 收集各 EP（包括自己）的执行结果；
6. 主 EP 将执行结果汇总后返回给用户。

如图 2.3 所示。

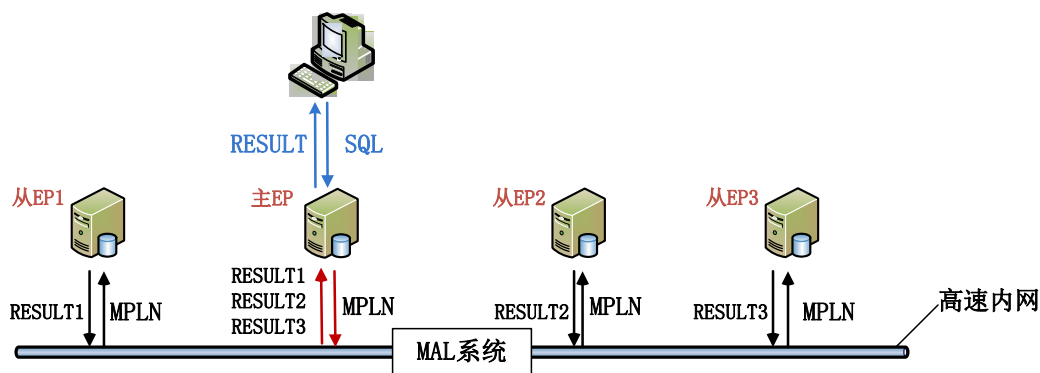


图 2.3 DM MPP 执行流程

2.3 系统特性

DM MPP 采用完全对等不共享架构，系统中各 EP 的功能完全对等，因此对于用户来说，MPP 系统的处理是完全透明的，用户任意登录 MPP 系统的任一节点进行操作都可获得完全的 MPP 支持。

使用 DM MPP 可获得以下功能特性支持：

- TB/PB 级数据分析 支持数据的并行装载和操作的并行执行，数据分布式存储在各 EP 中，能支持 TB/PB 级数据分析
- 支持绝大部分单机功能 支持绝大部分的 DM 单机版功能，同时支持行、列存储，支持存储过程、触发器、索引、分区表、多媒体数据类

型等

- 高性价比 无需额外配置特殊软、硬件，性价比超高
- 高可靠性 DM MPP 与 DM 数据守护相结合，为 MPP 系统中的每个 EP 配置一个或多个实时备库，在 EP 发生故障时其对应备库能迅速切换为主库继续提供服务，确保系统的高可用性
- 支持超大型集群 支持最多 1024 个 EP，轻松组建超大型集群

3 基本概念与原理

本章介绍 DM MPP 的一些基础概念与基本原理，虽然读者跳过这一章继续阅读后续章节也能进行 DM MPP 的相关操作，但是本章的阅读有利于读者进一步理解 DM MPP 的工作原理，对于后续对 MPP 系统的规划与管理能起到一定帮助。

3.1 基本概念

3.1.1 执行节点 EP

DM MPP 系统中每一个运行的 DM 数据库服务器实例称为一个执行节点 EP，基于数据守护的 MPP 环境内的备库除外。

3.1.2 主、从 EP

DM MPP 采用完全对等无共享架构，对整个系统来说，每个 EP 作用都是一样的，用户可以连接到其中的任何一个进行操作。而对每个用户会话来说，EP 具有主从之分。用户会话实际连接的那个 EP 对该用户会话来说称为主 EP，其余的 EP 都称为从 EP。

3.1.3 数据分布

DM MPP 系统中的数据分布在各 EP 中，支持表数据的哈希分布、随机分布、复制分布、范围分布、LIST 分布类型，用户可根据应用的实际情况为表数据选择合适的分布类型。

■ 哈希分布

哈希分布按照表定义中指定的一列或多列对行数据计算一个哈希值，再根据哈希值和哈希映射表，将该行数据分布到映射的节点上。

当表的连接查询中使用的连接键为哈希分布列时，MPP 下的查询计划会进行优化，比如可能减少计划中通讯操作符个数、使用索引、对分组计划优化等，减少数据在节点间的分发，提高查询效率。

使用哈希分布时，节点间的数据是否均衡，取决于设置的哈希分布列以及表中的数据情况。当节点个数变动时，各个节点的数据需要按照新的哈希映射表重新进行分发。

■ 随机分布

随机分布表不存在分布列，插入表数据时会按照一定的随机算法，将数据随机均衡分布到各个节点。

随机分布的优点是数据和节点间不存在映射关系。节点个数变动后，如果没有节点数据均衡的要求，可以不用对节点现有的数据进行变动。

一般来说，随机分布对于复杂查询及存在较多的节点间数据分发情况，性能不如哈希分布高。

■ 复制分布

复制分布表在每个节点上的本地数据都是一份完整的拷贝，查询该表数据时在任意节点上都能单独完成，不需要从其他节点获取数据。

复制分布一般用于数据量不是很大的表。

■ 范围分布

范围分布按照表定义中指定的一个或多个列的列值范围分布项，决定将一行数据存储到 MPP 的哪个相应 EP 上。

■ LIST 分布

LIST 分布通过指定表中的一个或多个列的离散值集，来确定将一行数据存储到 MPP 的哪个相应 EP 上。此分布用于表中列值可列举的情况。



说明： DM MPP 同时支持数据分布与分区表，实现了“数据分布后再分区”。在数据分布到各节点的基础上，再在单个节点上将数据再次分区，可进一步提高查询性能。分布的类型和分区的类型可以混合搭配，比如建立哈希分布表的范围水平分区表。

3.1.4 MAL 系统

MAL 系统是 DM 数据库实例间的高速通信系统，是基于 TCP 协议实现的一种内部通信机制，具有可靠、灵活、高效的特性。DM 通过 MAL 系统实现实例间的消息通讯。

3.1.5 全局连接与本地连接

MPP 系统中数据分布在各个 EP 中，用户只需要登录到某个 EP，系统自动建立这个 EP 与其余 EP 的连接，因此用户建立的实际上是与整个 MPP 系统的全局连接，用户对数据库的操作通过全局连接在 MPP 系统的所有 EP 进行。使用全局连接时，要求 MPP 系统的所有 EP 都正常提供服务，否则无法建立连接。

DM MPP 也提供本地连接。当使用本地连接时，用户登录到某个 EP 后，这个 EP 不再建立与其余 EP 的连接，用户的所有数据库操作仅在这个 EP 上进行。如 SELECT 语句以及 UPDATE 和 DELETE 语句中的 WHERE 条件中的子查询都仅仅查询本地 EP 的数据，而 INSERT 语句如果插入的数据根据分布定义应分布在其余 EP 时系统会报错。

通常在 MPP 系统正常运行时都使用全局连接，DM MPP 的快速装载和动态扩容使用到了本地连接，用户在某些时候如 MPP 系统中有 EP 故障时也可以使用本地连接。

DM 的各接口驱动程序都提供了连接属性用于设置全局连接（登录）或本地连接（登录），缺省都为全局连接。DM 交互式工具 DISql 也提供了登录参数 MPP_TYPE 用来指定使用全局连接或本地连接，“GLOBAL”表示全局连接，“LOCAL”表示本地连接，默认为全局连接。

3.2 MPP 并行执行计划

在 DM 数据库中，SQL 语句经过一系列的处理最终生成一棵由不同操作符组成的计划树，DM 执行器以自底向上的顺序执行计划树，数据也按自底向上的顺序在计划树中流动并经过各操作符的处理，最终在计划树的根节点生成执行结果。

在 DM MPP 环境中各 EP 执行的是并行计划，并行计划是在单节点执行计划的基础上，按照一定规则于适当的位置插入 MPP 通讯操作符而生成的。

本节介绍 DM MPP 并行执行计划的原理与处理过程。

3.2.1 并行计划相关操作符

MPP 并行执行计划是在单节点执行计划的基础上增加 MPP 通讯操作符生成的，DM MPP 相关通讯操作符有 5 个，如下表所示。

表 3.1 MPP 并行执行计划通讯操作符

操作符名称	功能
-------	----

MPP GATHER (MGAT)	主 EP 收集所有节点数据 从 EP 将数据发送到主 EP
MPP COLLECT (MCLCT)	在 MGAT 的基础上，增加主从 EP 执行同步功能，避免数据在主 EP 上堆积。 一个计划树中一般只会在较上层出现一个 MCLCT，但可能有多个 MGAT
MPP DISTRIBUTE (MDIS)	各 EP 节点间相互分发数据，按照分发列计算行数据的目标节点并发送过去，目标节点负责接收
MPP BROADCAST (MBRO)	功能类似 MGAT，收集数据到主 EP，该操作符带有聚集函数运算功能，仅和 FAGR 配合使用
MPP SCATTER (MSCT)	主 EP 发送完整数据到所有从 EP，保证每个节点数据都完整，一般和 MGAT 配合使用

3.2.2 并行计划的生成

在 DM 数据库中，计划树执行时，数据总是自底向上流动的，若是在 MPP 环境中，由于每个 EP 都只保存表的部分数据（复制分布除外），计划树叶子节点的数据都是不完整的，因此叶子节点向上传送的只是本 EP 部分的数据。要获得完整的正确执行结果，上层节点必须知道此时下层节点传来的数据是否完整，从而决定是否需要添加 MPP 通讯操作符，并根据单节点操作符的功能和数据分布情况决定应该添加何种通讯操作符。关于单节点执行计划操作符的介绍可以参看《DM8 系统管理员手册》的附录。

下面通过几个简单的例子说明 MPP 并行计划生成的原理。

首先，在 MPP 环境中建两个表 T1 与 T2，都使用哈希分布。

```
CREATE TABLE T1(C1 INT, C2 INT) DISTRIBUTED BY HASH(C1);
CREATE TABLE T2(C3 INT, C4 INT) DISTRIBUTED BY HASH(C3);
```

■ 例 1：单表查询，查询表 T1 的全部数据。

使用 EXPLAIN 查看语句的执行计划：

```
EXPLAIN SELECT * FROM T1;
```

显示的并行执行计划如下：

```
1  #NSET2: [0, 1, 16]
2  #MPP COLLECT: [0, 1, 16]; op_id(1) n_grp_by (0) n_cols(0) n_keys(0)
3  #PRJT2: [0, 1, 16]; exp_num(3), is_atom(FALSE)
4  #CSCN2: [0, 1, 16]; INDEX33555445(T1)
```

可以看到，这条语句的并行执行计划只是在单节点执行计划中增加了一个 MPP COLLECT 通讯操作符，用于在从 EP 扫描和投影数据后，主 EP 收集所有 EP 的数据。

■ 例 2：连接查询，连接键不是分布列的情况。

使用 EXPLAIN 查看语句的执行计划：

```
EXPLAIN SELECT * FROM T1 INNER JOIN T2 ON T1.C2=T2.C4;
```

显示的并行执行计划如下：

```

1  #NSET2: [0, 1, 16]
2  #MPP COLLECT: [0, 1, 16]; op_id(3) n_grp_by (0) n_cols(0) n_keys(0)
3  #PRJT2: [0, 1, 16]; exp_num(4), is_atom(FALSE)
4  #HASH2 INNER JOIN: [0, 1, 16]; KEY_NUM(1);
5  #MPP DISTRIBUTE: [0, 1, 8]; op_id(1) n_keys(1) n_grp(0) filter(FALSE
) rowid_flag(0)
6  #CSCN2: [0, 1, 8]; INDEX33555445(T1)
7  #MPP DISTRIBUTE: [0, 1, 8]; op_id(2) n_keys(1) n_grp(0) filter(FALSE
) rowid_flag(0)
8  #CSCN2: [0, 1, 8]; INDEX33555446(T2)

```

在这个并行计划中，哈希连接操作符“HASH2 INNER JOIN”的左右孩子 CSCN2 都各增加了一个 MPP DISTRIBUTE 通讯操作符，用于将数据按照连接键分发到各 EP 进行连接操作，之后计划的上层增加 MPP COLLECT 通讯操作符将各 EP 的连接和投影结果收集到主 EP。

■ 例 3：集函数查询，查询表 T1 的行数。

使用 EXPLAIN 查看语句的执行计划：

```
EXPLAIN SELECT COUNT(*) FROM T1;
```

显示的并行执行计划如下：

```

1  #NSET2: [0, 1, 0]
2  #MPP COLLECT: [0, 1, 0]; op_id(3) n_grp_by (0) n_cols(0) n_keys(0)
3  #PRJT2: [0, 1, 0]; exp_num(1), is_atom(FALSE)
4  #MPP BROADCAST: [0, 1, 0]; op_id(1)
5  #FAGR2: [0, 1, 0]; sfun_num(1),

```

在这个并行计划中，集函数操作符 FAGR2 的上层增加了 MPP BROADCAST 通讯操作符，

使得主 EP 可以收集并合并各从 EP 的集函数操作符执行结果。在此计划中，上层的 MPP COLLECT 操作符只起同步执行的作用，并不收集从 EP 的数据。

■ 例 4：连接查询，执行计划中使用嵌套连接操作符的情况。

使用 EXPLAIN 查看语句的执行计划：

```
EXPLAIN SELECT * FROM T1 INNER JOIN T2 ON T1.C1<>T2.C4;
```

显示的并行执行计划如下：

```
1  #NSET2: [49, 1, 16]
2  #MPP COLLECT: [49, 1, 16]; op_id(4) n_grp_by (0) n_cols(0) n_keys(0)
3  #PRJT2: [49, 1, 16]; exp_num(4), is_atom(FALSE)
4  #SLCT2: [49, 1, 16]; T1.C1 <> T2.C4
5  #NEST LOOP INNER JOIN2: [49, 1, 16];
6  #NTTS2: [0, 1, 8]; for_mdis(TRUE)
7  #MPP GATHER: [0, 1, 8]; op_id(1) n_grp_by (0) n_cols(0) n_keys(0)
8  #CSCN2: [0, 1, 8]; INDEX33555445(T1)
9  #NTTS2: [0, 1, 8]; for_mdis(TRUE)
10 #MPP GATHER: [0, 1, 8]; op_id(2) n_grp_by (0) n_cols(0) n_keys(0)
11 #CSCN2: [0, 1, 8]; INDEX33555446(T2)
```

由于嵌套连接操作符的特殊性，左表数据要和右表的全部数据都比较一遍，右表需要多次执行，因此在这个并行执行计划中增加了 MPP GATHER 通讯操作符将各从 EP 的数据完整收集到主 EP。计划上层的 MPP COLLECT 通讯操作符只起同步执行的作用，并不收集从 EP 的数据。

以上举了几个简单的例子说明 DM 是如何在单节点执行计划中增加 MPP 通讯操作符生成 MPP 并行执行计划的。DM 数据库的操作符众多，并行计划也复杂多变，这里不能一一介绍，用户可通过查看语句的并行执行计划体会 DM MPP 并行执行计划的生成规则。

3.2.3 数据分布与并行执行计划

在 MPP 环境中，建表时指定的数据分布类型决定了表数据的分布。DM MPP 支持的表分布类型包括哈希分布、随机分布、范围分布、复制分布、LIST 分布，数据插入或装载时系统会根据表的分布类型自动将数据发送到对应的 EP。

哈希分布、范围分布和 LIST 分布的共同特征是在创建表的时候，用户指定一个或多个列作为分布列，系统会针对每个插入的数据行计算这些对应列的值，确定将数据所属 EP。随机分布和复制分布则不需要指定分布列。

并行执行计划与数据分布密切相关，数据分布能够决定最终生成的并行计划。例如查询的数据经过预先判断发现正好都在一个 EP 上，服务器会做一定的优化，在最优的情况下，整个计划甚至不包含任何通讯操作符。优化的原则是尽量减少节点之间的通信交互。

因此，用户应根据应用中查询的实际需求来确定表的分布类型，进而得到较优的并行执行计划。

■ 场景一

在某应用中，查询语句中包含大量连接查询，表数据分布较为均匀，应用对查询的效率要求较高。此时我们可以使用哈希分布，并且将常用连接列作为哈希分布列，这样能尽可能地减少 EP 间的数据传递，少占用网络带宽，减少网络延迟，充分发挥多节点并行执行的巨大优势。

■ 场景二

在某应用中，查询以单表查询居多，连接查询较少，我们可以采用随机分布。随机分布使得海量数据能均匀分布，充分体现 MPP 的并行优势。

■ 场景三

对于单表查询或出现在连接查询中数据量较小的表，可采用复制分布。复制分布的表在每个 EP 上都有一份完整的数据拷贝，使得在生成并行执行计划时能减少对应通讯操作符的使用，进一步优化并行计划。

3.2.4 并行计划执行流程

DM MPP 对于查询语句和插入/删除/修改语句的处理是不同的，因为插入/删除/修改语句涉及到数据的修改，必须在数据行所在 EP 进行修改操作，而查询语句只需要主 EP 收

集查询结果即可。

3.2.4.1 查询语句处理流程

DM MPP 对查询语句的处理按如下流程进行：

1. 建立连接

用户连接到 MPP 系统内任意一个 EP 节点，则该 EP 为连接的主 EP，其余节点为从 EP；

2. 生成执行计划

主 EP 解析查询语句，生成普通的查询计划后，根据数据分布情况在合适的位置插入合适的并行通讯操作符，生成最终的并行查询计划；

3. 分发计划

主 EP 把执行计划分发给所有的从 EP；

4. 执行计划

各从 EP 收到计划后，生成执行计划的运行环境，所有 EP 并行执行，执行时各 EP 通过通讯操作符分发必要的的数据并协调执行进度；

5. 生成结果集

主 EP 收集所有 EP 的查询结果（包括自身数据），生成结果集；

6. 返回结果集

主 EP 将结果集返回给用户。

3.2.4.2 插入/修改/删除语句处理流程

DM MPP 对插入/修改/删除语句的处理按如下流程进行：

1. 建立连接

用户连接到 MPP 系统内任意一个 EP 节点，则该 EP 为连接的主 EP，其余节点为从 EP；

2. 生成执行计划

主 EP 解析语句，生成执行计划，其中包含的查询计划（即 WHERE 条件对应的计划）也是并行查询计划，另外还会生成一个对应的在从 EP 上执行的计划（MPLN）；

3. 准备数据

主 EP 开始执行计划时首先把查询计划部分发布给所有的从 EP，并行执行查询，主 EP 收集查询结果；

4. 定位节点

数据准备完成后，根据分布列和分布方式计算出需要修改的行数据所在的目标 EP，将 MPLN 以及操作所需数据发送到各对应的 EP。如果目标 EP 为本地则不发送，在本地直接完成操作；

5. 执行修改操作

从 EP 收到 MPLN 计划和数据后生成执行环境，执行实际的修改操作；

6. 返回执行结果

主 EP 等待所有的从 EP 执行完成后才会返回执行结果给客户端，只要其中有一个 EP 执行失败，则已经执行的所有 EP 都会回滚，保证数据的一致性，并返回错误信息给客户端。

3.2.5 DDL 语句分发

与 3.2.4 节中介绍的 DML 语句并行执行计划处理流程不同，DM MPP 对 DDL 语句的处理采用语句分发方式。主 EP 直接将 DDL 语句发送给各从 EP，每个从 EP 各自执行该 DDL 语句。主 EP 等待所有的从 EP 执行完成后才返回消息给用户，只要其中有一个 EP 执行失败，则已经执行的所有 EP 都会进行回滚，保证数据的一致性，并返回错误信息给客户端。

由于 DDL 语句分发，使得在 MPP 中登录任一 EP 执行的 DDL 操作都是全局的，包括数据库对象的建立、修改和删除等，也包括用户的建立、修改与删除。

3.3 相关配置文件

3.3.1 dm.ini MPP 相关配置项

dm.ini 是 DM 数据库实例的配置文件，通过配置该文件可以设置 DM 数据库服务器的各种功能和性能选项。dm.ini 中的配置项多达数百个，涉及 DM 数据库运行各个方面的设置，这里我们只介绍与 MPP 相关的几个配置项，读者若对其他配置项感兴趣可以参看《DM8 系统管理员手册》的相关章节。

表 3.2 dm.ini MPP 相关配置项

配置项	配置含义
INSTANCE_NAME	数据库实例名（长度不超过 16 个字符）
PORT_NUM	DM 服务器监听通讯端口号，服务器配置此参数，有效值范围（1024~65534），发起连接端的端口在 1024~65535 之间随机分配
MAL_INI	MAL 系统配置开关，0 表示不启用 MAL 系统，1 表示启用 MAL 系统，默认值为 0
MPP_INI	MPP 系统配置开关，0 表示不启用 MPP 系统，1 表示启用 MPP 系统，默认值为 0
MAX_EP_SITES	MPP 环境下 EP 节点的最大数量，有效值范围（2~1024），默认值为 64
MPP_HASH_LR_RATE	MPP 下，对 HASH JOIN 节点，可以根据左右儿子 CARD 代价的比值，调整 HASH_JOIN 的左右儿子的 MOTION 添加，从而影响计划。如果 CARD 比值超过此值，则小数据量的一方全部收集到主 EP 来做。取值范围（1~4294967294），默认值为 10
MPP_OP_JUMP	MPP 系统中操作符的跳转开关，是否支持通讯操作符的跳转功能。1：支持；0：不支持。取值范围：0、1，默认值为 1
PHF_NTTTS_OPT	MPP 系统中是否进行 NTTTS 计划的优化，打开时可能减少计划中的 NTTTS 操作符。1：支持；0：不支持。取值范围：0、1，默认值为 1

另外，DM MPP 要求 MPP 系统中每个 EP 的建库参数及 dm.ini 中一些配置参数的配置值必须一致，在用户第一次全局登录 MPP 系统时会进行检查，如果设置不一致，则会报错。

这些配置项如下表所示。

表 3.3 MPP 中每个 EP 必须设置相同的配置项

配置项	配置含义
建库参数	
BLANK_PAD_MODE	设置字符串比较时，结尾空格填充模式是否兼容 ORACLE，0：不兼容；1：兼容。默认为 0
LENGTH_IN_CHAR	VARCHAR 类型对象的长度是否以字符为单位，0：否，以字节为单位；1：是，以字符为单位。默认值为 0
USE_NEW_HASH	字符类型在计算 HASH 值时所采用的 HASH 算法类别，0：原始 HASH 算法；1：改进的 HASH 算法。默认值为 1
dm.ini 参数	
COMPRESS_MODE	建表时是否缺省压缩。0：不进行；1：进行。默认值为 0
PARALLEL_POLICY	用来设置并行策略。取值范围：0、1 和 2，缺省为 0。其中，0 表示不支持并行；1 表示自动并行模式；2 表示手动并行模式。默认值为 0
LIST_TABLE	默认情况下，创建的表是否为堆表，0：否；1：是。默认值为 0
COMPATIBLE_MODE	是否兼容其他数据库模式。0：不兼容，1：兼容 SQL92 标准，2：兼容 ORACLE，3：兼容 MS SQL SERVER，4：兼容 MYSQL，5：兼容 DM6，6：兼容 Teradata。默认值为 0

COUNT_64BIT	COUNT 集函数的值是否设置为 BIGINT。0：否；1：是。默认值为 1
ALTER_TABLE_OPT	是否对加列、修改列、删除列操作进行优化，0：全部不优化；1：全部优化；2：打开快速加列，对于删除列和修改列与 1 等效。默认值为 0

3.3.2 dmmal.ini 配置项

dmmal.ini 是 MAL 系统的配置文件，此配置文件生效的前提是 dm.ini 中的参数 MAL_INI 置为 1。使用同一套 MAL 系统的所有实例，MAL 系统配置文件要严格保持一致。

dmmal.ini 的配置参数及其介绍见下表。

表 3.4 dmmal.ini 配置项

配置项	配置含义
MAL_CHECK_INTERVAL	MAL 链路检测时间间隔，取值范围（0s-1800s），默认 30s，配置为 0 表示不进行 MAL 链路检测。为了防止误判，DMRAC 集群中，建议将配置值>= DCR_GRP_NETCHK_TIME。
MAL_CONN_FAIL_INTERVAL	判定 MAL 链路断开的的时间，取值范围（2s-1800s），默认 10s
MAL_LEAK_CHECK	是否打开 MAL 内存泄露检查，0：关闭，1：打开，默认 0
MAL_LOGIN_TIMEOUT	MPP/DBLINK 等实例间登录时的超时检测间隔（3-1800），以秒为单位，默认 15s
MAL_BUF_SIZE	单个 MAL 缓存大小限制，以兆为单位。当此 MAL 的缓存邮件超过此大小，则会将邮件存储到文件中。有效值范围（0~500000），默认为 100
MAL_SYS_BUF_SIZE	MAL 系统总内存大小限制，单位：M。有效值范围（0~500000），默认为 0，表示 MAL 系统无总内存限制
MAL_VPOOL_SIZE	MAL 系统使用的内存初始化大小，以兆为单位。有效值范围（1~500000），默认为 128，此值一般要设置的比 MAL_BUF_SIZE 大一些
MAL_COMPRESS_LEVEL	MAL 消息压缩等级，取值范围（0-10）。默认为 0，不进行压缩；1-9 表示采用 zip 算法，从 1 到 9 表示压缩速度依次递减，压缩率依次递增；10 表示采用 snappy 算法，压缩速度高于 zip 算法，压缩率相对低
MAL_TEMP_PATH	指定临时文件的目录。当邮件使用的内存超过 mal_buf_size 或者 mal_sys_buf_size 时，将新产生的邮件保存到临时文件中。如果缺省，则新产生的邮件保存到 temp.dbf 文件中
[MAL_NAME]	MAL 名称，同一个配置文件中 MAL 名称需保持唯一性
MAL_INST_NAME	数据库实例名，与 dm.ini 的 INSTANCE_NAME 配置项保持一致，MAL 系统中数据库实例名要保持唯一
MAL_HOST	MAL IP 地址，使用 MAL_HOST + MAL_PORT 创建 MAL 链路
MAL_PORT	MAL 监听端口，用于数据守护、DSC、MPP 等环境中各节点实例之间 MAL 链路配置，监听端口配置此参数，范围 1024~65534，

	发起连接端的端口在 1024~65535 之间随机分配
MAL_INST_HOST	MAL_INST_NAME 实例对外服务 IP 地址
MAL_INST_PORT	MAL_INST_NAME 实例服务器监听通讯端口号，服务器配置此参数，有效值范围（1024~65534），发起连接端的端口在 1024~65535 之间随机分配 此参数的配置应与 dm.ini 中的 PORT_NUM 保持一致
MAL_DW_PORT	MAL_INST_NAME 实例守护进程的监听端口，其他守护进程或监视器使用 MAL_HOST + MAL_DW_PORT 创建与该实例守护进程的 TCP 连接，监听端配置此参数，有效值范围（1024~65534），发起连接端的端口在 1024~65535 之间随机分配
MAL_LINK_MAGIC	MAL 链路网段标识，有效值范围（0~65535），默认 0。设置此参数时，同一网段内的节点都设置相同，不同网段内的节点设置的值必须不一样

3.3.3 dmmppctl

dmmppctl 是 DM MPP 系统的控制文件，它是一个二进制文件，用户不能直接进行配置。用户需要首先配置 dmmpp.ini，然后利用 dmctlcvt 工具进行转换得到生成的 dmmppctl 文件。MPP 系统中的每个 EP 都必须使用相同的 dmmppctl 文件，进行文件拷贝即可。

dmmpp.ini 的配置参数及其介绍见下表。

表 3.5 dmmpp.ini 配置项

配置项	配置含义
[SERVICE_NAME]	标识 MPP 系统中每个 EP 实例的选项名
MPP_SEQ_NO	实例在 mpp 系统内的序号，取值范围为 0~1023
MPP_INST_NAME	实例名

4 DM MPP 环境搭建与使用

本章介绍在实际应用环境中应如何规划、配置与部署 DM MPP 系统。通过阅读本章，读者可以从根据应用实际情况制定一个合适的 DM MPP 方案开始，完成 DM MPP 的配置与部署，建立应用分布表并载入数据。

4.1 制定合适的 DM MPP 方案

对于一个应用系统，如果制定出一个合适的 DM MPP 解决方案呢？可以主要从以下几个方面入手：

■ 应用是否适合使用 DM MPP 方案

DM MPP 解决方案具有高性价比、高可靠性、功能强大、可动态扩容等优点，但也是有其局限性的，并非是万能解药。DM MPP 主要针对海量数据的 OLAP 应用而研发，在合理规划的前提下，对普通的、并发量不大的一般 OLTP 或混合类型应用也适用，但不适合于高并发操作的 OLTP 类型应用。

■ 需要使用多大规模的 DM MPP 方案

确定应用适合使用 DM MPP 解决方案后，接下来需要根据应用规模确定 MPP 的节点数。DM MPP 最多支持 1024 个 EP 节点，在实际应用中具体使用几个 EP 节点则应根据应用数据规模、硬件规划、网络带宽和项目预算情况而定。理论上来说节点数越多越好，这样并行越充分，效率越高。但这也不是绝对的，方案设计者应同时考虑到 EP 节点间网络资源的情况，如果网络是瓶颈，则并不是节点数越多越好。

■ 需要配置怎样的硬件资源

DM MPP 采用完全对等不共享架构，不需要专用硬件，可以采用普通的 PC 服务器组建集群。需要注意的是，DM MPP 系统需要通过网络在各 EP 间传递数据，因此网络的带宽对于 MPP 系统的效率非常重要，建议配置千兆或万兆内部网络，用户应根据实际需求、配置成本及实际条件等综合评估决定。

■ 如何设计应用表

DM MPP 系统中的数据分布在各 EP 中，支持表数据的哈希分布、随机分布、复制分布、范围分布、LIST 分布类型，用户应根据表在应用中的特点选择合适的分布类型，可参考

3.1.3 节。

除了表的分布类型，管理员还应根据应用的实际情况确定表的分布列、分布列的数据类型以及是否需要组合分布列等，对于查询连接中使用频率较高的连接键等可以考虑作为分布列。

■ 如何配置 DM MPP 系统参数

DM 数据库实例配置文件 `dm.ini` 和 MAL 系统配置文件 `dmmal.ini` 中一些参数的配置会影响 MPP 系统的运行性能，管理员应根据应用和硬件的实际情况对这些参数进行适当的配置。同时应注意各 EP 必须保持一致的 INI 参数，具体见 3.3.1 节。

4.2 DM MPP 环境搭建

本节我们将以一个简单的两节点 MPP 为例，介绍手动配置与搭建 DM MPP 环境的步骤。DM 提供了 DM web 版数据库管理工具，可以很方便地搭建 DM MPP，具体见 5.1 节。但是阅读本节将更有利于读者掌握配置的流程，更好地在 MPP 系统运行时进行管理。

4.2.1 系统规划

本例配置一个两节点 MPP。两个节点都配置两块网卡，一块接入内部网络交换模块，一块接入到外部交换机。两节点实例名分别为 EP01 和 EP02，相关的 IP、端口等规划见下表。

表 4.1 MPP 系统规划

实例名	MAL_INST_HOST	MAL_INST_PORT	MAL_HOST	MAL 端口	MPP_SEQNO
EP01	192.168.1.11	5236	192.168.0.12	5269	0
EP02	192.168.1.21	5237	192.168.0.22	5270	1



注意：

DM MPP 各 EP 使用的 DM 服务器版本应一致，同时还应注意各 EP 所在主机的操作系统位数、大小端模式、时区及时间设置都应一致，否则可能造成意想不到的错误。

4.2.2 配置 `dm.ini`

首先，在 EP01 和 EP02 上分别创建数据库，用户可以使用 DM 的图形化客户端工具“数

数据库配置助手”或命令行工具 dminit 创建数据库。



注意：

在各 EP 上创建数据库时，要求有些初始化参数必须所有 EP 都相同，见 3.3.1 节表 3.3。我们建议各 EP 的数据库初始化参数都保持一致，以免产生错误。

分别对两个实例的 dm.ini 进行配置。

修改 EP01 的 dm.ini 的以下几个参数如下：

```
INSTANCE_NAME  = EP01
PORT_NUM       = 5236

MAL_INI        = 1
MPP_INI        = 1
```

修改 EP02 的 dm.ini 的以下几个参数如下：

```
INSTANCE_NAME  = EP02
PORT_NUM       = 5237

MAL_INI        = 1
MPP_INI        = 1
```

4.2.3 配置 dmmal.ini

为两个 EP 配置 dmmal.ini 如下，配置完全一样，EP 间可互相拷贝。dmmal.ini 与 dm.ini 放在相同的目录下。

```
[MAL_INST1]

MAL_INST_NAME = EP01

MAL_HOST      = 192.168.0.12

MAL_PORT      = 5269

MAL_INST_HOST = 192.168.1.11

MAL_INST_PORT = 5236
```

```
[MAL_INST2]

MAL_INST_NAME = EP02

MAL_HOST      = 192.168.0.22

MAL_PORT      = 5270

MAL_INST_HOST = 192.168.1.21

MAL_INST_PORT = 5237
```

4.2.4 配置 dmmpp.ctl

dmmpp.ctl 是一个二进制文件，用户不能直接配置，需要先配置 dmmpp.ini。

配置 dmmpp.ini 如下：

```
[SERVICE_NAME1]

MPP_SEQ_NO    = 0

MPP_INST_NAME = EP01


[SERVICE_NAME2]

MPP_SEQ_NO    = 1

MPP_INST_NAME = EP02
```

使用 DM 提供的工具 dmctlcvt 将 dmmpp.ini 转换成 dmmpp.ctl，dmctlcvt 工具在 DM 安装目录的“bin”子目录中。

转换生成的 dmmpp.ctl 需要放在与 dm.ini 同一个目录。假设 DM 的安装路径为 c 盘根目录，下面的命令将 dmmpp.ini 转换为 dmmpp.ctl，命令中的“TYPE=2”参数表示将文本文件转换成控制文件，也可以使用“TYPE=1”参数进行逆向转换。

```
dmctlcvt          TYPE=2          SRC=c:\dmdbms\data\dameng\dmmpp.ini
DEST=c:\dmdbms\data\dameng\dmmpp.ctl
```

将生成的 dmmpp.ctl 拷贝至另一 EP，保证 MPP 系统中所有 EP 的 dmmpp.ctl 完全相同。

4.2.5 运行 MPP

经过前面四个步骤，DM MPP 环境已经配置完成了。分别启动 EP01 和 EP02 的 DM 数

数据库实例（顺序不分先后），DM MPP 系统即能正常运行，用户就可以登录任一 EP 进行数据库操作了。

4.3 建立分布表

DM MPP 支持表数据的哈希分布、随机分布、复制分布、范围分布、LIST 分布类型，用户可根据实际情况选择合适的分布类型，具体各分布类型的特点见 3.1.3 节。

MPP 的数据分布类型和具体设置在建表时指定，语法如下，建表的其他语法分支可参见《DM8_SQL 语言使用手册》。

```
CREATE [[GLOBAL] TEMPORARY] TABLE <表名定义> <表结构定义>;

<表结构定义>::=<表结构定义 1> | <表结构定义 2>

<表结构定义 1>::= (<列定义> {,<列定义>} [, <表级约束定义>{,<表级约束定义>}]) [ON COMMIT
    <DELETE | PRESERVE> ROWS] [<PARTITION 子句>] [<空间限制子句>] [<STORAGE 子句>] [<
    压缩子句>] [<ROW MOVEMENT 子句>] [<DISTRIBUTE 子句>]

<表结构定义 2>::= [ON COMMIT <DELETE | PRESERVE> ROWS] [<空间限制子句>] [<STORAGE
    子句>] [<压缩子句>] AS <不带 INTO 的 SELECT 语句> [<DISTRIBUTE 子句>];

<DISTRIBUTE 子句>::= DISTRIBUTED [<RANDOMLY> | <FULLY>]
    | DISTRIBUTED BY [<HASH>] (<列名> {,<列名>})
    | DISTRIBUTED BY RANGE (<列名> {,<列名>}) (<范围分布项> {,<范围分布项>})
    | DISTRIBUTED BY LIST (<列名> {,<列名>}) (<LIST 分布项> {,<LIST 分布项>})

<范围分布项>::= VALUES LESS THAN (<表达式>{,<表达式>}) ON <实例名>
    | VALUES EQU OR LESS THAN (<表达式>{,<表达式>}) ON <实例名>

<LIST 分布项>::= VALUES (<表达式>{,<表达式>}) ON <实例名>
```

下面给出几个简单的创建不同类型分布表的例子。

例 1：创建哈希分布表 T_HASH，分布列为 C1。

```
CREATE TABLE T_HASH(C1 INT, C2 CHAR(10)) DISTRIBUTED BY HASH (C1);
```

例 2：创建随机分布表 T_RANDOM。

```
CREATE TABLE T_RANDOM(C1 INT, C2 CHAR(10))DISTRIBUTED RANDOMLY;
```

例 3：创建复制分布表 T_FULLY。

```
CREATE TABLE T_FULLY(C1 INT, C2 CHAR(10))DISTRIBUTED FULLY;
```

例 4：创建范围分布表 T_RANGE，分布列为 C1。

```
CREATE TABLE T_RANGE (C1 INT, C2 CHAR(10))
DISTRIBUTED BY RANGE (C1) (VALUES EQU OR LESS THAN (100) ON EP01, VALUES LESS
THAN(MAXVALUE) ON EP02);
```

例 5：创建 LIST 分布表 T_LIST，分布列为 C1。

```
CREATE TABLE T_LIST(C1 INT, C2 CHAR(10))
DISTRIBUTED BY LIST (C1) (VALUES(3) ON EP01,VALUES(4) ON EP02);
```

例 6：创建哈希分布表的范围水平分区表。

```
CREATE TABLE T_HASH_RANGE_PARTITION
(C1 INT, C2 CHAR(10), C3 CHAR(10))
PARTITION BY RANGE(C1)
(
PARTITION PART_1 VALUES LESS THAN(0) ,
PARTITION PART_2 VALUES LESS THAN(10) ,
PARTITION PART_3 VALUES LESS THAN(100) ,
PARTITION PART_4 VALUES LESS THAN(MAXVALUE)
)
DISTRIBUTED BY HASH (C1);
```

在创建分布表时，用户应注意以下一些使用限制：

- 单机模式下建的分布表和普通表一样，但是不能创建指定实例名的分布表（如范围分布表和 LIST 分布表）；
- 在 MPP 模式下创建分布表，如果未指定列则默认为 RANDOMLY (随机) 分布表；

- 分布列类型不支持 BLOB、CLOB、IMAGE、TEXT、LONGVARCHAR、BIT、BINARY、VARBINARY、LONGVARBINARY、BFILE、时间间隔类型、虚拟列和用户自定义类型；
- HASH 分布、RANGE 分布、LIST 分布允许更新分布列，并支持包含大字段列的表的分布列更新，但包含 INSTEAD OF 触发器的表、堆表不允许更新分布列；
- 对于 FULLY（复制）分布表，只支持单表查询的更新和删除操作，并且查询项或者条件表达式中都不能包含 ROWID 伪列表表达式；
- RANGE（范围）分布表和 LIST（列表）分布表，分布列与分布列值列表必须一致，并且指定的实例名不能重复；
- 引用约束的引用列和被引用列都必需包含分布列，且分布情况完全相同；
- 随机分布表不支持 UNIQUE 索引。

4.4 快速数据装载

DM MPP 特别适合于海量数据的存储和处理，因此在应用中常常面临将大量数据从某个或某些历史数据库中装载到 MPP 系统的需求。为了满足海量数据的快速装载需求，DM 提供了快速装载工具 dmfldr，能够对 DM 单机版和 MPP 系统进行海量数据的快速装载。

dmfldr 为命令行工具，使用时必须指定必要的执行参数，具体可参看《DM8_dmfldr 使用手册》，本文档就不再详细介绍了，这里仅对 dmfldr 中 MPP 相关的参数 MPP_CLIENT 进行说明。

dmfldr 支持 MPP 环境下的两种数据加载模式：客户端分发模式和本地分发模式，通过参数 MPP_CLIENT 进行设置。使用客户端分发模式时，数据在 dmfldr 客户端进行分发然后直接向指定 EP 发送数据；使用本地分发模式时，每个 EP 对应一个 dmfldr 和一份数据，每个 dmfldr 只选择出对应本节点的数据并发送，不管其他节点的数据。默认使用客户端分发模式。

4.5 停止 MPP 系统

需要停止 DM MPP 系统的运行时，只需要停止每个 EP 的 DM 实例即可，没有特别的顺序要求。

若在 DM MPP 系统的运行过程中，某一 EP 发生故障停机，则整个 MPP 系统将处于不能

正常服务的状态。当前所有的用户会话会被系统断开，不能进行全局登录，只能进行本地登录。因此，为了保证 MPP 系统的高可用性，我们强烈建议采用 DM MPP 与数据守护相结合的部署方案，见第 6 章。

4.6 MPP 相关系统过程与函数

为了方便 MPP 环境下数据库应用的编写，DM 提供了一些 MPP 相关的系统过程与函数。

1. SP_SET_SESSION_MPP_SELECT_LOCAL

定义：

```
VOID  
  
SP_SET_SESSION_MPP_SELECT_LOCAL(  
  
    local_flag      int  
  
)
```

功能说明：

MPP 系统下设置当前会话是否只查询本节点数据。如果不设置，表示可以查询全部节点数据。

参数说明：

local_flag: 设置标记，0：查询全部节点数据；1：只查询本节点数据。

例：

```
SP_SET_SESSION_MPP_SELECT_LOCAL(1);
```

2. SF_GET_SESSION_MPP_SELECT_LOCAL

定义：

```
INT  
  
SF_GET_SESSION_MPP_SELECT_LOCAL()
```

功能说明：

查询 MPP 系统下当前会话是否只查询本节点数据。

返回值：

0 表示查询全部节点数据，1 表示只查询本节点数据。

例：

```
SELECT SF_GET_SESSION_MPP_SELECT_LOCAL();
```

3. SP_SET_SESSION_LOCAL_TYPE

定义:

```
VOID
```

```
SP_SET_SESSION_LOCAL_TYPE (
    ddl_flag      int
)
```

功能说明:

MPP 下本地登录时, 设置本会话上是否允许 DDL 操作。本地登录默认不允许 DDL 操作。

参数说明:

ddl_flag: 0 表示不允许当前本地会话执行 DDL 操作, 1 表示允许。

例:

MPP 下本地登录会话。

```
SP_SET_SESSION_LOCAL_TYPE (1);
CREATE TABLE TEST(C1 INT);
SP_SET_SESSION_LOCAL_TYPE (0);
```

4. SF_GET_EP_SEQNO

定义:

```
INT
```

```
SF_GET_EP_SEQNO (
    rowid      bigint
)
```

功能说明:

根据查询出的行数据的 ROWID 获取本条数据来自哪个 EP。

参数说明:

rowid: 行数据的 ROWID。

返回值:

MPP 系统内 EP 的序号。

例：

```
SELECT SF_GET_EP_SEQNO (ROWID);
```

5. SF_GET_SELF_EP_SEQNO

定义：

INT

SF_GET_SELF_EP_SEQNO ()

功能说明：

获取本会话连接的 EP 序号。

返回值：

本会话连接的 EP 序号。

例：

```
SELECT SF_GET_SELF_EP_SEQNO ();
```

6. SP_GET_EP_COUNT

定义：

INT

SP_GET_EP_COUNT (

SCH_NAME VARCHAR(128),

TAB_NAME VARCHAR(128)

)

功能说明：

统计 MPP 环境下表在各个节点的数据行数。

参数说明：

SCH_NAME：表所在模式名

TAB_NAME：表名

例：

```
SP_GET_EP_COUNT('SYSDBA','T');
```


4.7 MPP 下系统过程与系统视图常见用法

本节给出一些 DM MPP 环境下，数据库应用对系统过程与系统视图的常见用法示例，关于系统视图的具体介绍可参见《DM8 系统管理员手册》附录 2。

1. 获取会话连接的 EP 的节点序号

```
SELECT SF_GET_SELF_EP_SEQNO();
```

2. 根据 ROWID 获取本行数据来自哪个 EP

```
SELECT SF_GET_EP_SEQNO(ROWID);
```

3. 获取 EP 节点配置信息

```
SELECT * FROM V$MPP_CFG_ITEM WHERE SF_GET_EP_SEQNO(ROWID) =
SF_GET_SELF_EP_SEQNO();
```

4. 获取当前会话连接的实例名

```
SELECT NAME FROM V$INSTANCE WHERE SF_GET_EP_SEQNO(ROWID) =
SF_GET_SELF_EP_SEQNO();
```

5. 获取 MPP 系统内所有 EP 的所有会话

```
SELECT * FROM V$SESSIONS;
```

6. 获取当前连接的实例上的所有会话

```
SELECT * FROM V$SESSIONS WHERE SF_GET_EP_SEQNO(ROWID) = SF_GET_SELF_EP_SEQNO();
```

7. 获取实例 EP01 上的所有会话

```
SELECT * FROM V$SESSIONS WHERE SF_GET_EP_SEQNO(ROWID) = (SELECT DISTINCT EP_SEQNO
FROM V$MPP_CFG_ITEM WHERE INST_NAME = 'EP01');
```

8. 获取表 TEST 在每个实例上的数据行数

```
CALL SP_GET_EP_COUNT('SYSDBA', 'TEST');
```

9. 获取所接实例上的表 TEST 的使用空间

```
SELECT TABLE_USED_PAGES('SYSDBA', 'TEST');
```

10. 获取每个实例上的表 TEST 使用空间

```
SELECT TABLE_USED_PAGES('SYSDBA', 'TEST'), NAME FROM V$INSTANCE;
```

11. 获取所有实例上表 TEST 的总使用空间

```
SELECT SUM(TOTAL_SIZE) FROM (SELECT TABLE_USED_PAGES('SYSDBA', 'TEST')
TOTAL_SIZE, NAME FROM V$INSTANCE);
```

4.8 使用说明

DM MPP 支持绝大多数单机版 DM 的功能，但在某些小的功能点使用上存在一些使用限制，具体如下：

- MPP 环境下，不支持创建 SET NULL 或 SET DEFAULT 约束检查规则的引用约束；
- MPP 环境下不支持创建外部表、间隔分区表；
- MPP 环境下不支持修改表的 ADD [COLUMN] <列名>[<IDENTITY 子句>] 子句；
- MPP 环境下不支持创建位图索引、空间索引、数组索引；
- MPP 环境下位图连接索引涉及的维度表需 FULLY 分布；
- MPP 环境下不支持索引的 ONLINE 选项；
- MPP 环境下不支持视图的 WITH CHECK OPTION 操作；
- MPP 环境下不支持闪回查询；
- MPP 环境下不支持物化视图日志；
- MPP 各 EP 创建数据库时指定的编码格式应相同，否则可能造成乱码问题；
- MPP 环境下 INI 参数 MVCC_RETRY_TIMES 无效，发生 MVCC 冲突时直接报错；
- MPP 环境下不支持 DBMS_JOB 包；
- MPP 环境下日志辅助表不支持 FULLY 分布。

5 DM MPP 系统管理

DM web 版数据库管理工具（DEM）提供了图形化部署与监控 DM MPP 的功能。数据库管理员对规划好 DM MPP 系统的机器与端口，就可以使用 DEM 的集群部署功能，根据页面的指示一步步地搭建好 DM MPP 环境。部署完成后还可以使用 DEM 的集群监控功能对 MPP 集群的运行情况进行监控。

本章还介绍了如何为运行中的 DM MPP 系统动态扩容以及 MPP 环境的 DDL 克隆与还原。

5.1 使用 DEM 部署与监控 DM MPP

DEM 提供 DM MPP 的图形化搭建与管理功能。关于 DEM 工具的启动，请查看 DEM 相关文档。DEM 启动后，通过浏览器访问，如下图所示：

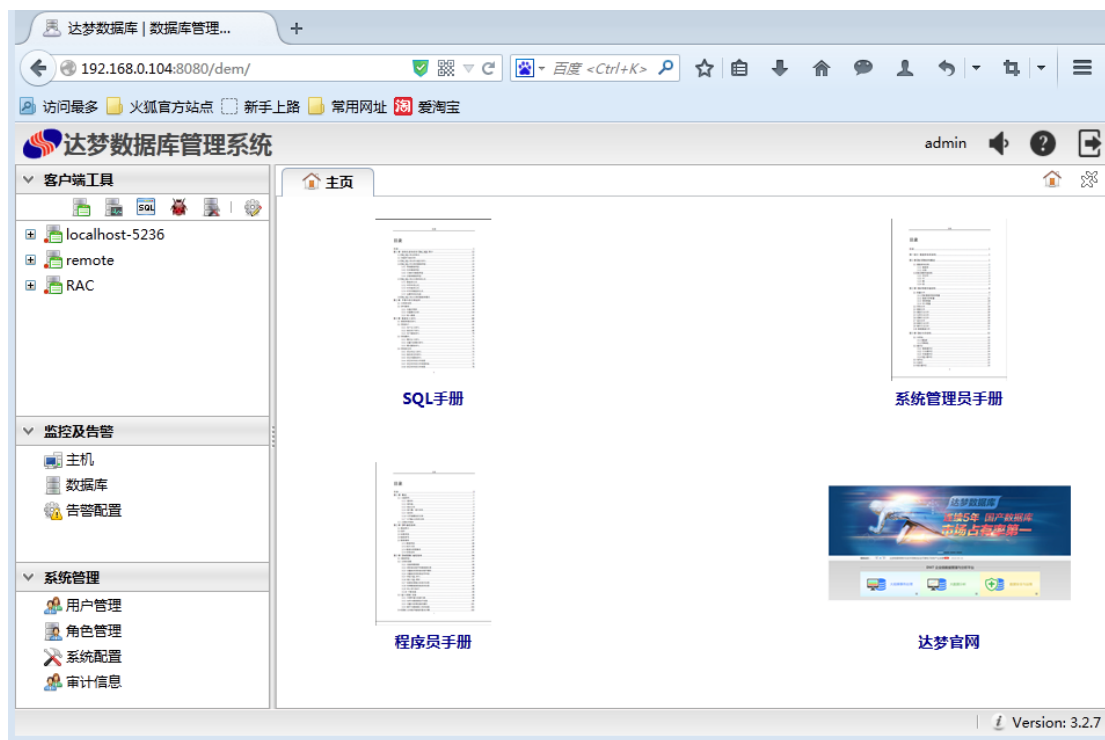


图 5.1 DEM 工具主界面

在 DEM 中，对于集群的管理包括集群部署和集群监控。集群部署在“客户端工具”栏点击“部署集群”可以打开，集群监控在“监控及告警”栏的“数据库”监控中添加对集群的监控即可打开。



在使用 DEM 对集群进行部署和监控之前需要在各节点所在机器部署 DM 数据库代理工具 DMAgent。

5.1.1 使用 DEM 部署 DM MPP

在 DEM 的“客户端工具”栏，选择部署集群工具，打开新建集群部署的对话框，如下图所示：



图 5.2 新建集群部署对话框

输入集群名称，选择集群类型为“MPP”，点击“确定”后，在 DEM 右边面板打开部署 MPP 的面板，进入“环境准备”界面：

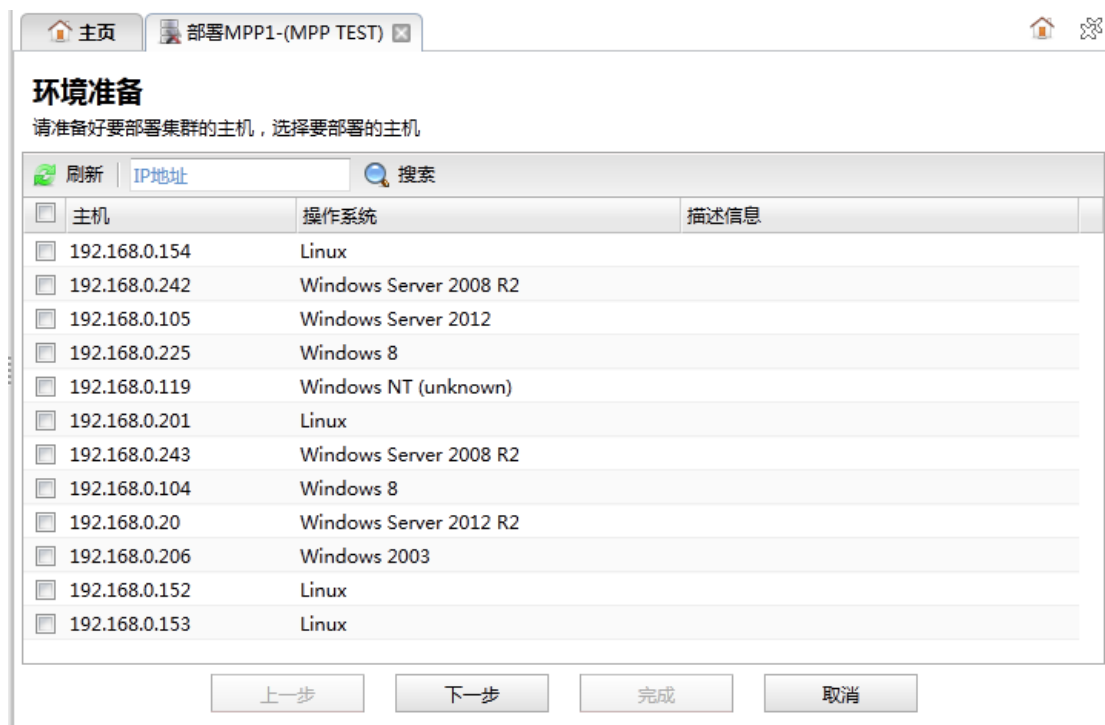


图 5.3 MPP 部署-环境准备

参数详解：

刷新：刷新主机列表。

搜索：输入 IP 地址，快速搜索指定主机。

选择要部署 MPP 的主机，点击“下一步”，进入“实例规划”界面：

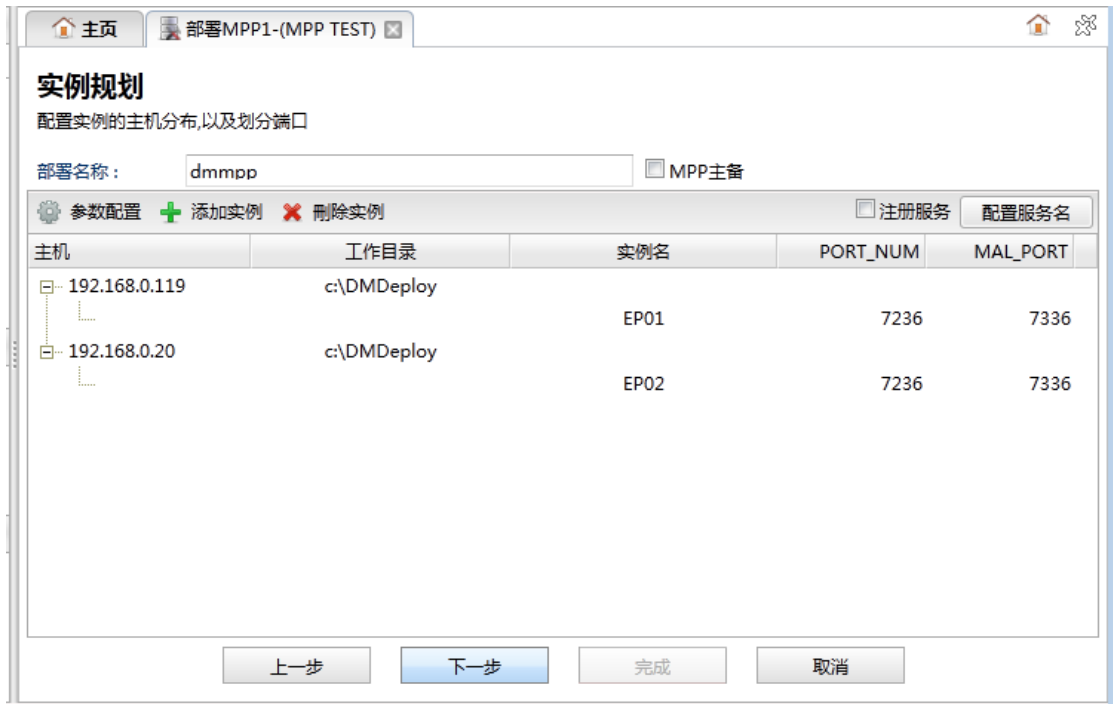


图 5.4 MPP 部署-实例规划

缺省部署的是 MPP 主备系统，本示例仅部署单纯 MPP 系统，因此将“MPP 主备”前的复选框选项去掉。

参数详解：

部署名称：部署名称，会在各部署主机的工作目录创建对应名称的目录来存储该集群。

MPP 主备：是否配置 MPP 主备系统还是单纯 MPP 系统。

参数配置：统一配置实例列表中所有实例的参数。另外也可以通过双击实例列表中某一个实例的某一个参数进行单独配置。

添加实例：添加实例，可以在选择的主机中添加实例，同时可以指定实例类型，添加之后会出现在实例列表中。

删除实例：删除选中的实例。

注册服务：如果想让 dmserver 服务开机自启动，则需把 dmserver 注册成服务，注册完成后重启机器时，就会自动启动 dmserver 服务。

配置服务名：配置注册的服务名，默认 dmserver 服务名为 DmService_实例名。

在配置好各实例的参数后，点击“下一步”，进入“数据准备”页面。

图 5.5 MPP 部署-数据准备

参数详解：

初始化新库：初始化一个新库，作为守护系统的实例。

使用已存在的库：指定已经存在的库作为守护系统的实例。指定库所在的主机、INI 路径、库的登录用户名和密码。

备好数据之后，点击“下一步”，进入“配置 dmaml.ini”界面，配置 dmmal.ini 相关参数。

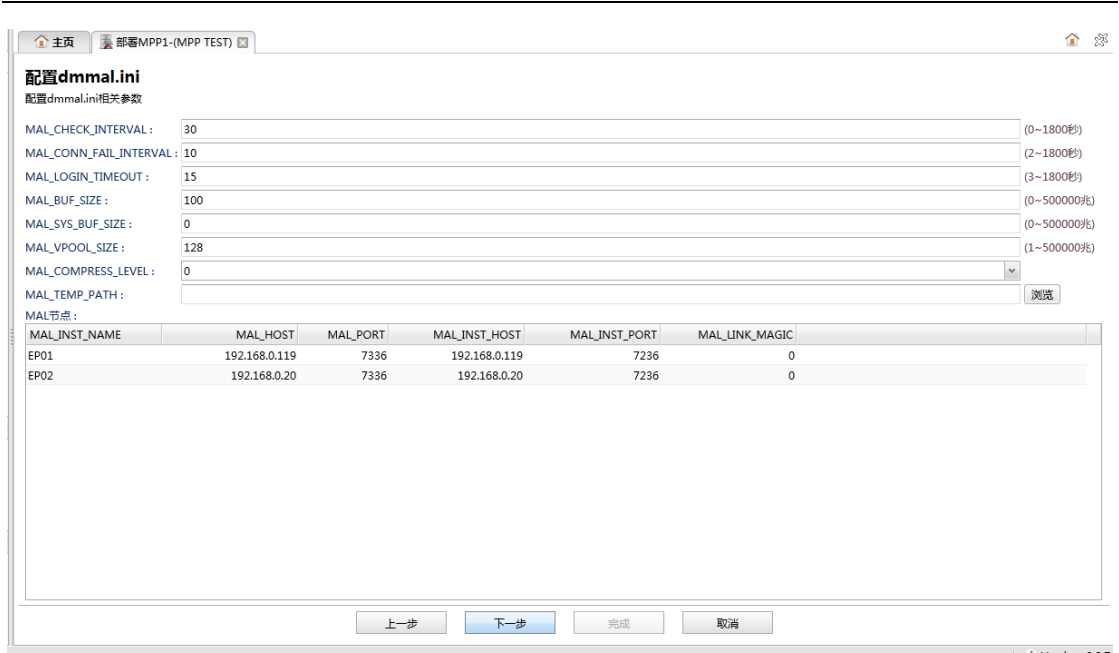


图 5.6 MPP 部署-配置 dmmal.ini

可以在这个页面中对 dmmal.ini 的相关参数进行配置，参数的介绍见 3.3.2 节。

点击“下一步”，进入“配置 dmarch.ini”界面，配置 dmarch.ini 参数。

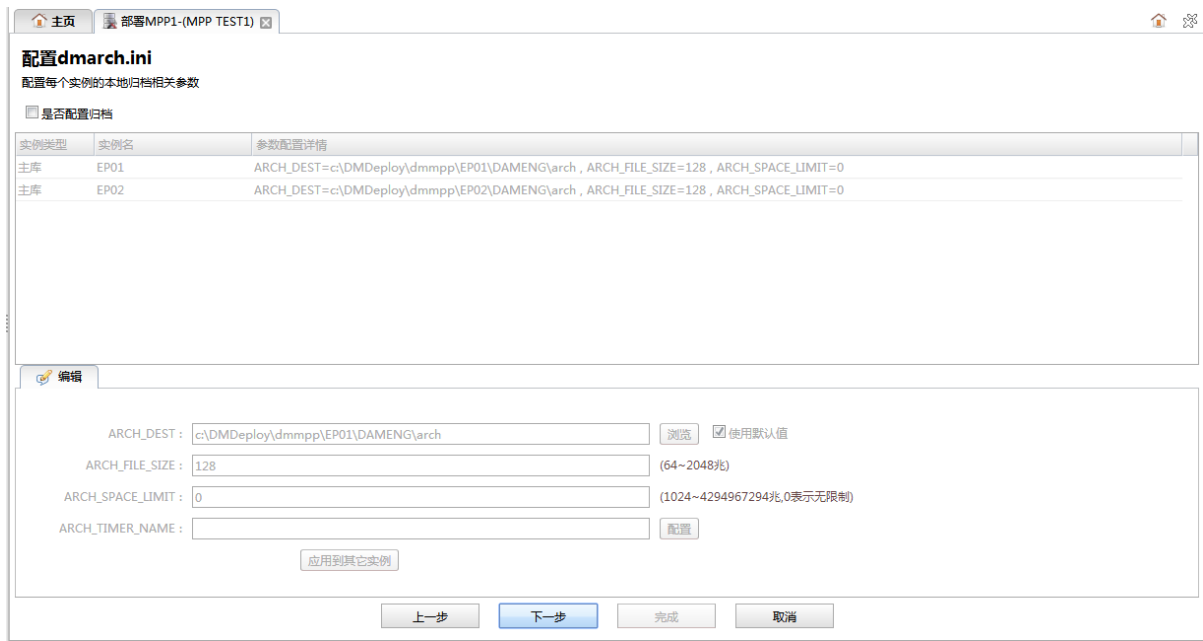


图 5.7 MPP 部署-配置 dmarch.ini

MPP 环境下并不要求必须配置归档,但用户也可以根据应用实际需要进行配置,选中“是否配置归档”选择框,即可进行配置。

点击“下一步”，进入“上传服务器文件”界面，选择上传的服务器文件。



图 5.7 MPP 部署-上传服务器文件

参数说明：

各节点将使用同一的达梦数据库服务器文件：选择该选项，则为所有为 Linux 的主机上传一份服务器文件，为所有为 windows 的主机上传一份服务器文件。

各节点将单独配置达梦数据库服务器文件：为每一个主机单独上传服务器文件。

使用 ssl 通讯加密：如果上传的服务器是使用 ssl 通讯加密的，则需要上传客户端 SYSDBA 的 ssl 密钥文件，和输入 ssl 验证密码。

点击“下一步”，进入“详情总览”界面。列出将要部署的集群环境的所有配置信息：

点击“下一步”，开始执行部署任务。执行过程如下：

主頁

部署MPP1-(MPP TEST)

部署任务

行部署任务

号总数:18,已完成:6,出错:0,取消:12,开始时间:2017-05-24 13:37:44

名	状态	消息	耗時
(192.168.0.119)上传服务體文件	成功		15秒33毫秒
置实例EP01			
P01数据库初始化	成功		11秒502毫秒
LEP01库数据为实例数据	正在执行...	启动数据库...	
P01配置dm.ini	等待执行...		
P01配置dmmal.ini	等待执行...		
P01配置dmarch.ini	等待执行...		
P01配置dmmpp.ctl	等待执行...		
P01启动数据库	等待执行...		
(192.168.0.20)上传服务體文件	成功		11秒998毫秒
置实例EP02			
P02数据库初始化	成功		8秒518毫秒
例貝数据文件到EP02	等待执行...		
P02配置dm.ini	等待执行...		
P02配置dmmal.ini	等待执行...		
P02配置dmarch.ini	等待执行...		
P02配置dmmpp.ctl	等待执行...		
P02启动数据库	等待执行...		

查看部署信息

停止所有任务

回滾所有任务

重做失败任务

上一步

下一步

完成

取消

参数说明:

停止所有任务：停止执行所有任务。

回滚任务：执行结束后，可以回滚所有执行的任务，清除环境。

重做失败任务：重新执行失败的任务。

36

示。

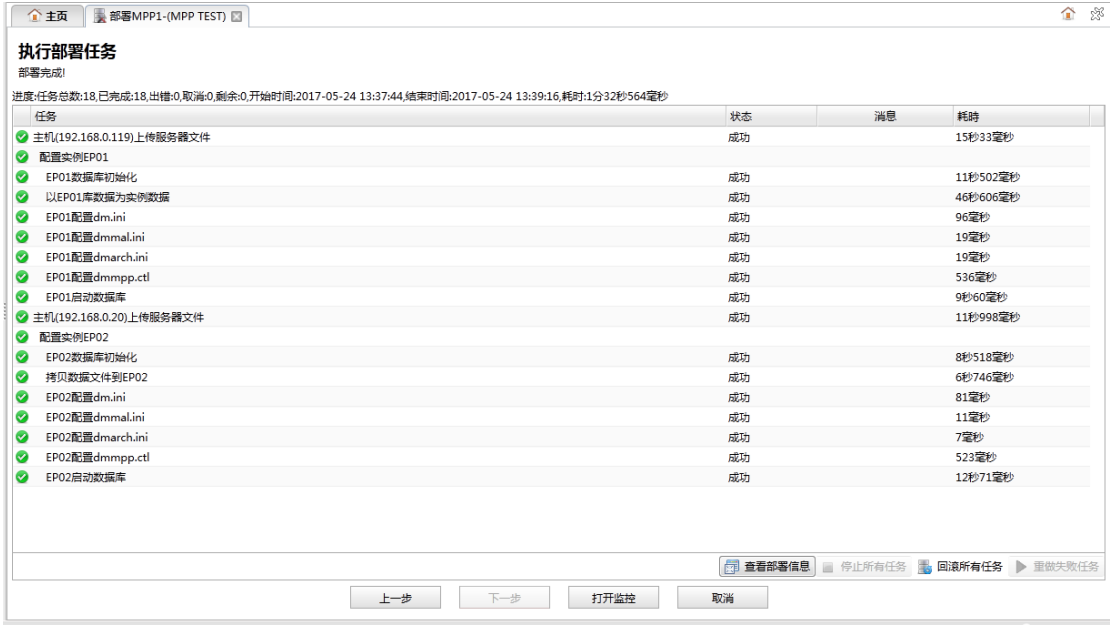


图 5.10 MPP 部署-完成执行部署任务

至此，DM MPP 环境部署完成，如果点击“打开监控”可直接进入监控集群页面。

5.1.2 使用 DEM 监控 DM MPP 运行

在 DEM 的“监控与告警”栏，双击“数据库”，在 DEM 右边面板打开数据库监控面板。



图 5.11 数据库监控面板

选择工具栏“添加”按钮，下拉选择“集群”。打开集群添加对话框：

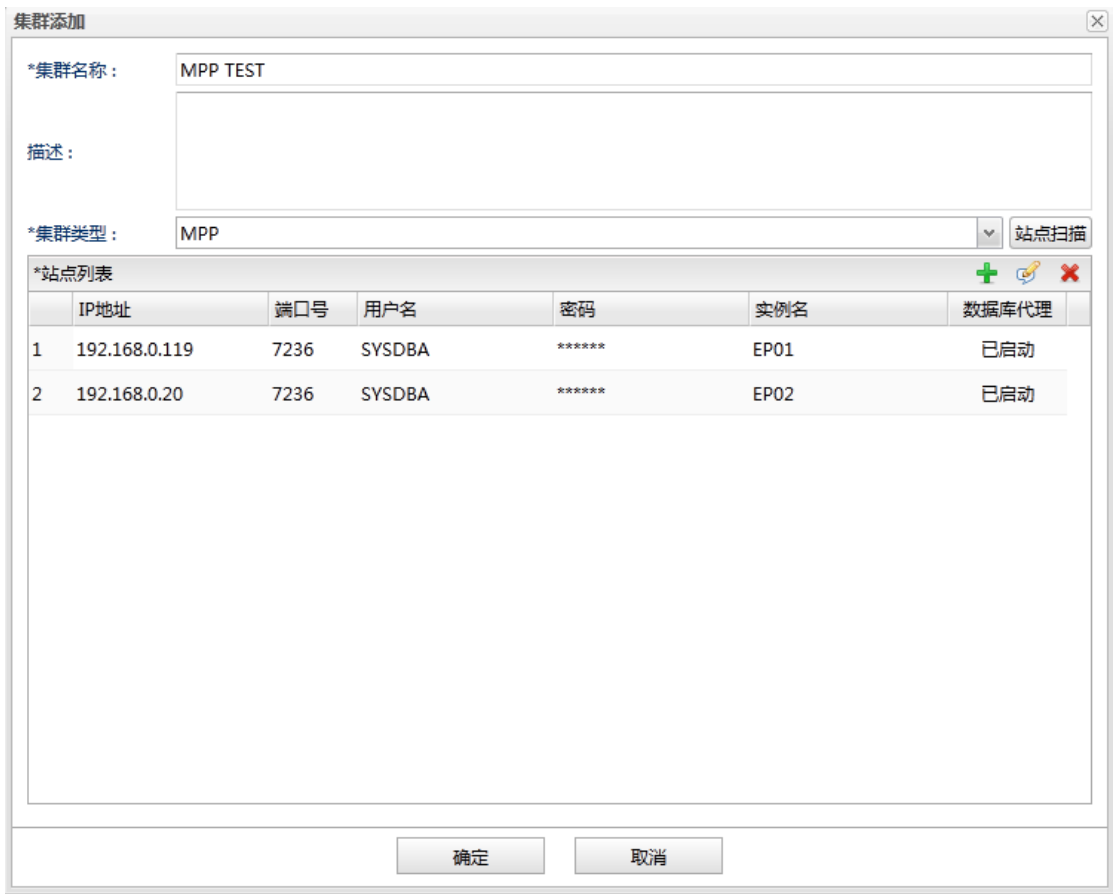


图 5.12 集群添加对话框

添加好需要监控的 MPP 集群后，点击“确定”，就可以对 MPP 集群进行监控了。



图 5.13 集群监控面板

对集群的监控与管理操作包括集群分析，集群管理，启动/停止集群，sql 监控，表监控等，具体操作请查看 DEM 联机帮助。

5.2 MPP 系统动态扩容

DM MPP 将多个 DM 数据库实例组织为一个并行计算网络，为用户提供高性价比的海量数据存储和处理能力。但即使 DM MPP 的成本代价极低，用户也常常不能在最初部署 MPP 时就部署好足够应用持续运行很长时间的系统规模。在应用系统运行几年后，随着应用数据

规模的不断增加，需要为运行中的 MPP 系统进一步扩充 MPP 容量是一个常见的需求。为此，DM MPP 提供了系统动态扩容功能，在不影响数据库应用的情况下，可以为 DM MPP 集群动态增加新的 EP。

下面介绍 DM MPP 系统动态扩容的具体步骤。

5.2.1 环境准备

以本书 4.2 节搭建的 DM MPP 系统为例，此 MPP 系统已有 EP01 和 EP02 两个节点，现在要再增加一个节点 EP03。

设当前的两节点 MPP 系统中已有哈希分布表 T1（普通表）和 T2（HUGE 表）、复制分布表 T3 和随机分布表 T4，它们的创建和插入数据的语句如下：

```
--哈希分布表 T1

DROP TABLE T1;

CREATE TABLE T1(C1 INT,C2 INT,C3 INT,C4 VARCHAR(10)) DISTRIBUTED BY HASH(C1);

DECLARE

    i INT;

BEGIN

    FOR i IN 1..1000 LOOP

        INSERT INTO T1 VALUES (i,i+1,i+2,'adasf');

    END LOOP;

END;

/

COMMIT;

--哈希分布表 T2

DROP TABLE T2;

CREATE HUGE TABLE T2(C1 INT,C2 INT,C3 INT,C4 VARCHAR(10)) DISTRIBUTED BY

HASH(C1);
```

```
DECLARE

    i INT;

BEGIN

    FOR i IN 1..10 LOOP

        INSERT INTO T2 VALUES (i,i+1,i+2,'adasf');

    END LOOP;

END;

/

COMMIT;


--复制分布表 T3

DROP TABLE T3;

CREATE TABLE T3(C1 INT,C2 INT,C3 INT,C4 VARCHAR(10)) DISTRIBUTED FULLY;


DECLARE

    i INT;

BEGIN

    FOR i IN 1..1000 LOOP

        INSERT INTO T3 VALUES (i,i+1,i+2,'adasf');

    END LOOP;

END;

/

COMMIT;


--随机分布表 T4

DROP TABLE T4;

CREATE TABLE T4(C1 INT,C2 INT,C3 INT,C4 VARCHAR(10)) DISTRIBUTED RANDOMLY;


DECLARE

    i INT;
```

```

BEGIN

    FOR i IN 1..1000 LOOP

        INSERT INTO T4 VALUES (i,i+1,i+2,'adasf');

    END LOOP;

END;

/

COMMIT;

```

5.2.2 动态增加节点

1. 禁止系统 DDL 操作

全局登录 MPP 系统任一节点，执行下面的语句禁止系统的 DDL 操作。

```
SP_DDL_FORBIDEN(1);
```

2. 克隆数据库

数据库克隆的目的是把系统中的对象定义信息进行备份，用于恢复到新加的节点上，生成的备份集位于指定的目录备份集中。

- 若 MPP 系统处于运行状态，采用联机 DDL 克隆方式（需要配置本地归档）。

全局登录 MPP 系统任一节点，执行下面的语句。

```
BACKUP DATABASE DDL_CLONE BACKUPSET 'CLONE';
```

生成的备份集保存在当前登陆节点 bak 路径中的 clone 目录中。

- 若 MPP 系统处于退出状态，选择 MPP 系统任一节点使用 DMRMAN 工具进行脱机备份。

```

RMAN>backup    database    'c:\dmdbms\data\dameng\dm.ini'    ddl_clone    backupset
'clone';

```

生成的备份集保存在当前备份节点 bak 路径中的 clone 目录中。

3. 脱机还原

在新增节点上执行脱机还原。手动拷贝克隆的备份集目录 clone 到新增节点 EP03 的 bak 目录中，使用 DMRMAN 工具进行脱机还原。

```

RMAN>RESTORE    DATABASE    'c:\dmdbms\data\dameng\dm.ini'    FROM    BACKUPSET
'c:\dmdbms\data\dameng\bak\clone';

```

```

RMAN>RECOVER DATABASE 'c:\dmdbms\data\DAMENG\dm.ini' FROM BACKUPSET '
c:\dmdbms\data\dameng\bak\clone';

```

若要增加多个节点，则依次执行拷贝、脱机还原步骤。

4. 配置新增节点的 `dmmal.ini`

为新增的节点配置 `dmmal.ini`，其中包含扩容前 MPP 系统的节点，以及新增节点 EP03 的信息。

```

[MAL_INST1]

MAL_INST_NAME = EP01

MAL_HOST      = 192.168.0.12

MAL_PORT      = 5269

MAL_INST_HOST = 192.168.1.11

MAL_INST_PORT = 5236


[MAL_INST2]

MAL_INST_NAME = EP02

MAL_HOST      = 192.168.0.22

MAL_PORT      = 5270

MAL_INST_HOST = 192.168.1.21

MAL_INST_PORT = 5237


[MAL_INST3]

MAL_INST_NAME = EP03

MAL_HOST      = 192.168.0.32

MAL_PORT      = 5271

MAL_INST_HOST = 192.168.1.31

MAL_INST_PORT = 5238

```

修改新增节点的 `dm.ini` 文件中的如下配置项：

```

INSTANCE_NAME = EP03

```

```
PORT_NUM      = 5238
```

```
MAL_INI  = 1
```

```
MPP_INI  = 0
```

5. MOUNT 方式启动新增节点

以 MOUNT 方式启动新增节点 EP03，登录后执行如下命令。

```
SP_DDL_FORBIDEN(1);

ALTER DATABASE OPEN FORCE;
```

以 MOUNT 方式启动的目的是防止启动后有用户执行 DDL 操作，因此先禁止 DDL 后再 OPEN。

6. 动态增加 MAL

分别本地登录 MPP 系统中原有的每个节点，执行下列语句为每个原有节点的 MAL 配置增加新增节点信息。

```
--设置本地 MAL 配置状态

SF_MAL_CONFIG(1,0);

--增加 MAL 配置

SF_MAL_INST_ADD('MAL_INST3','EP03','192.168.0.32',5271, '192.168.1.31',5238);

--应用 MAL 配置

SF_MAL_CONFIG_APPLY();

--重置本地 MAL 配置状态

SF_MAL_CONFIG(0,0);
```

7. 增加 MPP 节点，设置表的标记

全局登录 MPP 系统原有节点中的任一个，执行下列语句。

```
--广播方式设置 MAL 配置状态

SF_MAL_CONFIG(1,1);

--保存老的 HASHMAP

SF_MPP_SAVE_HASHMAP();

--增加 MPP 实例 EP03

SF_MPP_INST_ADD('service_name3', 'EP03');
```

增加节点配置后，之前的连接失效，需要断开连接，重新全局登录，然后执行下列语句。


```
--设置分布表的重分发状态

SF_MPP_REDIS_STATE_SET_ALL();

--广播方式重置 MAL 配置状态

SF_MAL_CONFIG(0,1);


--增加节点之后，可以放开 DDL 限制

SP_DDL_FORBIDEN(0);
```

至此，节点 EP03 已经顺利地加入 MPP 系统中。但是，MPP 系统动态扩容并没有完成，在原有 MPP 系统中建立的哈希分布表、复制分布表和随机分布表需要在增加节点后的 MPP 系统中进行数据重分发。

5.2.3 数据重分发

MPP 动态增加节点后，原系统中的哈希分布表、复制分布表和随机分布表需要进行数据重分发，范围分布表和 LIST 分布表不需要进行数据重分发。

1. 哈希分布表数据重分发

5.2.1 节中创建的 T1 和 T2 表都是哈希分布表，其中 T1 为普通表，T2 为 HUGE 表，下面以这两个表为例进行哈希分布表的数据重分发。

具体步骤如下：

- 1) 全局登录新增节点 EP03（如果新增多个节点，则要分别全局登录每个新增节点）
- 2) 设置本 session 可对表进行重分发（插入和删除），执行如下语句：

```
SET_SESSION_MPP_REDIS(1);
```

- 3) 设置重分发状态，执行如下语句：

```
SF_MPP_REDIS_STATE_SET('SYSDBA','T1',2);

SF_MPP_REDIS_STATE_SET('SYSDBA','T2',2);
```

- 4) 进行查询插入，执行如下语句：

```
INSERT INTO T1 SELECT * FROM T1 WHERE EP_SEQNO('T1')= 2; --2 为本节点 SEQNO

COMMIT;
```

```
INSERT INTO T2 SELECT * FROM T2 WHERE EP_SEQNO('T2')= 2; --2 为本节点 SEQNO
COMMIT;
```

5) 设置待删除数据状态, 执行如下语句:

```
SF_MPP_REDIS_STATE_SET('SYSDBA','T1',3);
SF_MPP_REDIS_STATE_SET('SYSDBA','T2',3);
```

6) 本地登录每个 MPP 系统的原有节点, 删除分发出去的数据。

执行如下语句:

```
SET_SESSION_MPP_REDIS(1);
```

- 对于普通表 T1, 执行如下语句:

```
DELETE FROM T1 WHERE EP_SEQNO('T1')=2; --2 为本节点 SEQNO
```

如果新增了多个节点, 则需要删除分发到所有这些新增节点的数据, 语句形如:

```
DELETE FROM T1 WHERE EP_SEQNO('T1')=新加节点 MPP 序号 1 OR EP_SEQNO('T1')=新加节点
MPP 序号 2 .....
```

- 对于 HUGE 表 T2, 由于直接使用 DELETE 效率较低, 采用查询插入再删除表的方式:

```
--放开本地登录下的 DDL 限制
SP_SET_SESSION_LOCAL_TYPE(1);

CREATE TABLE STR_TAB(A VARCHAR);

INSERT INTO STR_TAB SELECT TABLEDEF('SYSDBA','T2') FROM DUAL;

ALTER TABLE T2 RENAME TO T2_REDIS;

DECLARE

    sqltxt VARCHAR;

BEGIN

    SELECT * INTO sqltxt FROM STR_TAB;

    EXECUTE IMMEDIATE sqltxt;

END;

/
```

```
SF_MPP_REDIS_STATE_SET('SYSDBA','T2',3);

INSERT INTO T2 SELECT * FROM T2_REDIS WHERE EP_SEQNO('T2_REDIS')= 2;

DROP TABLE T2_REDIS;

DROP TABLE STR_TAB;

COMMIT;
```

在这个步骤中需要注意的是，如果表上建有二级索引，则需要重新创建二级索引。

7) 所有新增节点执行完上述步骤后，重置表的分发状态为普通状态，执行如下语句：

```
SF_MPP_REDIS_STATE_SET('SYSDBA','T1',0);
```

2. 复制分布表数据重分发

5.2.1 节中创建的表 T3 是复制分布表，下面以 T3 为例进行复制分布表的数据重分发。

具体步骤如下：

1) 本地登录新增节点 EP03（如果新增多个节点，则要分别本地登录每个新增节点），执行如下语句：

```
SET_SESSION_MPP_REDIS(1);

SP_SET_SESSION_LOCAL_TYPE(1); --放开本地登录下的 DDL 限制
```

2) 创建新增节点 EP03 到某一原有节点间的外部链接（如果新增多个节点，则要为每个新增节点创建一个这样的外部链接）

```
CREATE LINK LINK_EP01 CONNECT WITH SYSDBA IDENTIFIED BY SYSDBA USING 'EP01';
```

3) 执行查询插入

```
INSERT INTO T3 SELECT * FROM T3@LINK_EP01;
```

4) 删除外部链接

```
DROP LINK LINK_EP01;
```

5) 全局登录 MPP 系统，重置表的分发状态为普通状态，执行如下语句：

```
SET_SESSION_MPP_REDIS(1);

SF_MPP_REDIS_STATE_SET('SYSDBA','T3',0);
```

3. 随机分布表数据重分发

5.2.1 节中创建的表 T4 是随机分布表，下面以 T4 为例进行随机分布表的数据重分发。

具体步骤如下：

- 1) 本地登录新增节点 EP03 (如果新增多个节点, 则要分别本地登录每个新增节点), 执行如下语句:

```
SET_SESSION_MPP_REDIS(1);
SP_SET_SESSION_LOCAL_TYPE(1); --放开本地登录下的 DDL 限制
```

- 2) 设置表的重分发状态

```
SF_MPP_REDIS_STATE_SET('SYSDBA','T4',2);
```

- 3) 创建 EP03 与每个原有节点间的外部链接 (如果新增多个节点, 则要为每个新增节点创建这样的外部链接)

```
CREATE LINK LINK1 CONNECT WITH SYSDBA IDENTIFIED BY SYSDBA USING 'EP01';
CREATE LINK LINK2 CONNECT WITH SYSDBA IDENTIFIED BY SYSDBA USING 'EP02';
```

- 4) 本地登录每个原有节点, 查出每个原有节点上表 T4 的 min(ROWID) 和 max(ROWID)。

```
SELECT MIN(ROWID),MAX(ROWID) FROM T4;
```

- 5) 在新增节点 EP03 上, 分别使用连接每个原有节点的外部链接执行查询插入。

```
INSERT INTO T4 SELECT * FROM T4@LINK1 WHERE ROWID
BETWEEN V_MIN1 AND V_MIN1 + (V_MAX1 - V_MIN1)*1/3;

INSERT INTO T4 SELECT * FROM T4@LINK2 WHERE ROWID
BETWEEN V_MIN2 AND V_MIN2 + (V_MAX2 - V_MIN2)*1/3;
```

说明: V_MIN1 和 V_MAX1 对应第 4) 步中查询出的 EP01 上的 min(ROWID) 和 max(ROWID); V_MIN2 和 V_MAX2 对应第 4) 步中查询出的 EP02 上的 min(ROWID) 和 max(ROWID); 1/3 中的 3 表示动态增加节点后的节点总数。

如果有多个新增节点, 则节点 2、节点 3.....上执行的查询插入语句形如:

```
INSERT INTO T4 SELECT * FROM T4@LINKx WHERE ROWID
BETWEEN V_MINx + (V_MAXx - V_MINx)*N_SEQ/N_SITE_NEW_TOTAL+ 1 AND V_MINx + (V_MAXx
- V_MINx)*(N_SEQ + 1)/N_SITE_NEW_TOTAL;
```

说明: V_MINx 和 V_MAXx 的意义与前面说明的一致; N_SEQ 表示新增节点序号, 新增节点 1 的 N_SEQ 为 0, 新增节点 2 的 N_SEQ 为 1.....; N_SITE_NEW_TOTAL 表示动态增加节点后的节点总数。

- 6) 全局登录某一节点, 设置表的待删除数据状态

```
SF_MPP_REDIS_STATE_SET('SYSDBA','T4',3);
```

7) 本地登录每个原有节点，删除分发出去的数据，执行如下语句：

```
SET_SESSION_MPP_REDIS(1);
```

```
DELETE FROM T4 WHERE ROWID BETWEEN V_MIN AND V_MIN + (V_MAX - V_MIN) * 1/3;
```

说明：V_MIN 和 V_MAX 对应第 4) 步中查询出的 min(ROWID) 和 max(ROWID)；

1/3 中的 1 表示新增节点数；3 表示动态增加节点后的节点总数。

8) 每个原有节点都完成删除后，都执行提交操作。

```
COMMIT;
```

9) 全局登录某个节点，重置表的分发状态为普通状态，执行如下语句：

```
SET_SESSION_MPP_REDIS(1);
```

```
SF_MPP_REDIS_STATE_SET('SYSDBA','T4',0);
```

10) 本地登录 EP03，删除之前创建的到每个原有节点的外部链接（如果新增多个节点，则要本地登录每个新增节点删除这些外部链接）

```
DROP LINK LINK1;
```

```
DROP LINK LINK2;
```

至此，原有 MPP 的哈希分布表、复制分布表和随机分布表都已进行了数据重分发，但是还需要更新扩容后的 MPP 系统的控制文件中的数据分布控制结构。

全局登录某个节点，执行如下语句：

```
SF_MAL_CONFIG(1,1);
```

```
SF_MPP_SAVE_HASHMAP();
```

```
SF_MAL_CONFIG(0,1);
```

MPP 系统动态扩容全部完成！

5.3 MPP 环境的 DDL 克隆与还原

DDL 克隆可以把系统中的对象定义信息进行备份，之后再还原到别的数据库节点。利用这个功能，可以将一个 MPP 系统的对象定义信息克隆并还原到另一个 MPP 系统中，且两个 MPP 系统的节点数不必一致。

需要注意的是，由于 MPP 环境下执行 DDL 克隆时会将 dmmppctl 也一并备份还原，

若还原的 MPP 环境与备份环境不一致时，会导致还原后的 MPP 系统不能正确启动。因此在执行完还原操作后，需要使用使用 `dmctlcvt` 工具，将当前环境的 `dmmpp.ini` 再次生成 `dmmpp.ctl`，替换还原生成的 `dmmpp.ctl`，具体操作方法可参见 4.2.4 节。

6 DM MPP 主备系统

为了提高 MPP 系统可靠性，克服由于单节点故障导致整个系统不能继续正常工作的问題，DM 在普通的 MPP 系统基础上，引入数据守护主备机制，为每一个 MPP 节点配置一个实时备库作为备份节点，必要时备库可切换为主库代替故障节点工作，提高系统的可靠性和可用性。我们推荐用户在使用 DM MPP 时都应使用 MPP 主备守护系统以确保系统可靠性。

MPP 主备的主要目的是为 DM MPP 集群提供数据可靠性保障，备库只做数据容灾、备份，MPP 备库并不是 MPP 集群的一部分，只是某个 MPP 节点（主库）的镜像。MPP 备库不参与 MPP 操作，与其他 MPP 备库之间没有任何关系，MPP 备库只能以单节点方式提供只读服务，但不提供全局的 MPP 只读服务。

为了提高系统可靠性并节约硬件资源，DM MPP 主备系统可采用交叉守护的方式，即每个 EP 和其对应的备库实例不在同一台主机上，将 EP 与别的 EP 的备库放在一台主机上。

DM MPP 主备系统的架构如图 6.1 所示。

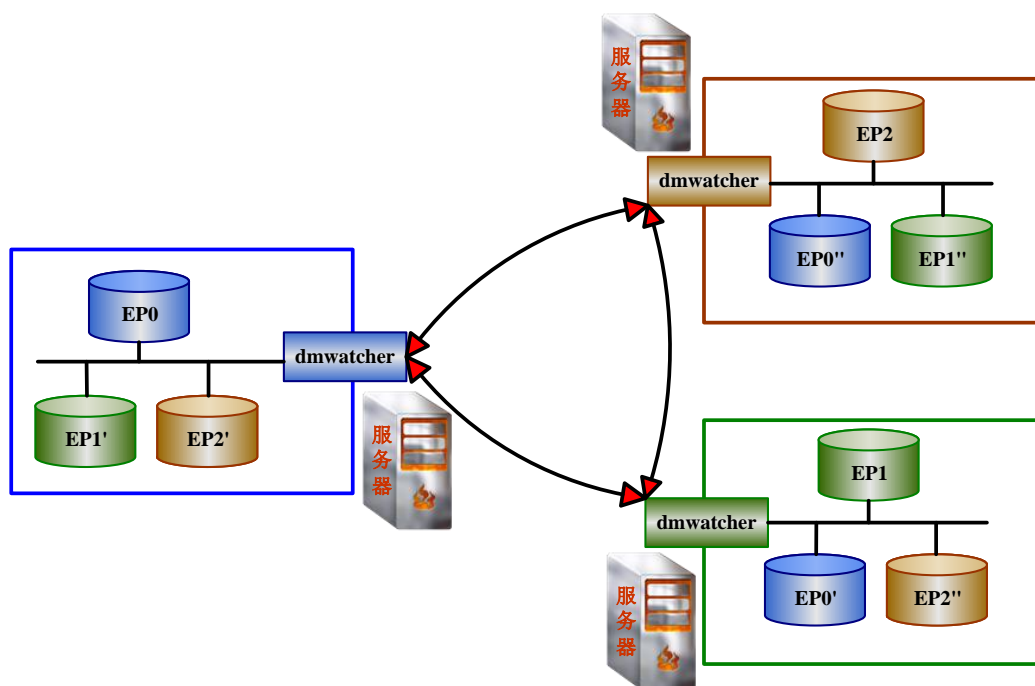


图 6.1 DM MPP 主备系统架构图

关于 DM MPP 的具体介绍及搭建步骤可参看《DM8 数据守护与读写分离集群 V3.0》的 7.4 节。

咨询热线：400-991-6599

技术支持：dmtech@dameng.com

官网网址：www.dameng.com



武汉达梦数据库有限公司
Wuhan Dameng Database Co.,Ltd.

地址：武汉市东湖新技术开发区高新大道999号未来科技大厦C3栋16—19层

16th-19th Floor, Future Tech Building C3, No.999 Gaoxin Road, Donghu New Tech Development Zone,Wuhan,Hubei Province,China

电话：(+86) 027-87588000 传真：(+86) 027-87588810
