

基于 NoSQL 的 PDM 版本管理

夏秀峰^{1,2}, 张赫男²

(1. 沈阳航空航天大学 辽宁省通用航空重点实验室, 辽宁 沈阳 110136;
2. 沈阳航空航天大学 计算机学院, 辽宁 沈阳 110136)

摘要:复杂的产品设计中会产生大量的中间版本,而版本模型是版本管理的基础。PDM 系统依赖产品版本的管理,保证设计人员可随时访问产品的历史版本,并对设计过程进行回溯,以保证产品在生命周期内的各阶段数据的正确性。通过分析当前 PDM 版本管理模型存在的不足、基于关系型数据库的版本管理的缺点,提出一种分层图结构的版本构造模型,能够直观、清晰地反映版本之间的关联关系及版本进化历史记录,弥补了现有版本模型的劣势。此外,随着版本数量的增加,PDM 中存储的产品数据展示出了大数据的特点,使得关系型数据库(RDB)在数据处理上力不从心。因此,以 NoSQL 作为底层数据库创建了基于分层图结构的版本模型,利用 NoSQL 数据库的优势,完善版本管理。依据版本分层图模型的特点,实现了版本追溯、版本合并及版本有效性控制等版本管理功能。

关键词:PDM 版本管理;NoSQL;分层图模型;版本管理;有效性控制

中图分类号:TP31

文献标识码:A

文章编号:1673-629X(2016)10-0022-05

doi:10.3969/j.issn.1673-629X.2016.10.005

Version Management of PDM Based on NoSQL

XIA Xiu-feng^{1,2}, ZHANG He-nan²

(1. Key Lab of General Aviation of Liaoning Province, Shenyang Aerospace University,
Shenyang 110136, China;
2. School of Computer Science, Shenyang Aerospace University, Shenyang 110136, China)

Abstract: The complex product design will produce a large amount of intermediate version, and the version of the model is the basis of version management. Product Data Management (PDM) system depends on version management, which makes sure that the design personnel can access to the product version of history at any time. And back to the design process, to ensure that products in the stages of life cycle data is correct. As analyzing the shortage of existing version management model of PDM, a new version model of the hierarchical structure is proposed, which can reflect the relationship between the version and the version history clearly and intuitively and make up for the disadvantages of version model. With the increasing of version number, product data stored in PDM shows the characteristics of big data, making the RDB incapable on data processing. So the version model based on hierarchical structure is constructed with NoSQL as the underlying database, taking the advantages of NoSQL to improve the version management. According to the characteristics of the model, the version management functions like version tracking, version merging and effective version control are realized.

Key words: version management of PDM; NoSQL; hierarchical graph model; version management; validity control

0 引言

PDM 是一种管理所有与产品相关信息及过程的软件技术,是产品设计与开发过程的一个平台^[1]。而 PDM 系统的版本管理用于管理产品生命周期内的零件、部件及产品等对象产生和变化的整个历程^[2]。

产品的设计过程总是分阶段、反复而非线性的。

在产品设计中产生的大量数据是历史设计知识的重要来源^[3]。因此,在整个过程中,设计者希望保留设计过程中产生的所有版本,以便随时访问不同对象在各阶段的版本,从而回溯整个设计过程,及时对设计错误进行修改^[4]。因此,应用版本管理解决复杂工程设计中的数据管理难题,是保证设计数据一致性和依赖

收稿日期:2016-02-06

修回日期:2016-05-11

网络出版时间:2016-09-19

基金项目:航空科学基金(2013ZG54032)

作者简介:夏秀峰(1964-),男,博士,教授,CCF 高级会员,研究方向为数据库理论与技术;张赫男(1991-),女,硕士研究生,研究方向为管理信息系统与数据库。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20160919.0842.044.html>

性的重要手段^[5-7]。

通过对 PDM 版本现状的分析,构建了基于 NoSQL 的版本分层图模型,弥补了传统版本管理模型的不足,避免了关系数据库对版本管理的灵活性差、扩展性差、性能差等缺点。

1 PDM 版本分析

版本是一个对象在设计过程中某一时间点上有意义的快照^[8]。版本管理是对版本对象以及版本间关系进行管理。版本对象不是单一存在的个体,每个版本都会有一系列演变版本,而管理版本间的关系,同版本对象的管理同样重要^[9]。

1.1 传统 PDM 版本管理模型分析

产品设计过程中,对设计内容的不断修改,将生成更多新的版本。每个对象的版本是相关联的,都由该对象的基版本演变产生。因此,应清晰表示对象的各版本间的关系。目前,产品设计数据的版本管理有线性模型、树型模型和有向无环图结构模型^[10-12],如图 1 所示。

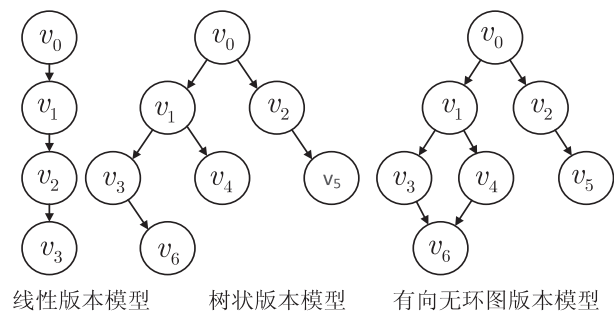


图 1 版本管理模型

线性模型的特点是版本的演变过程以时间为序。除版本根节点外,每个版本有且只有一个父版本;除最新版本节点外,每个版本有且只有一个子版本。该模型比较简单,不能区分替换版本和修订版本的差别,也不能完整体现版本的逻辑关系。

树型模型的特点是层次清晰,除根节点无父版本外,其余节点有且只有一个父版本。该结构主要体现在产品设计流程中,以基版本为基础,各版本不断演化的关系。版本的树型模型的局限性在于它不能反映版本合并情况(多个版本合并成一个新的版本)。

有向无环图描述模型的特点是该模型可以清晰表示版本的历史信息,弥补了树型模型的不足,可以反映版本的合并情况,完整地表示出版本间逻辑关系及版本间的依赖性,是一个比较完善的版本模型。但该模型失去了树型结构的层次性,对于版本间关系体现不够清晰。

为了能形象、完整地反映出产品设计过程中版本之间的逻辑关系,清晰显示版本演化过

程,并将版本间关系真实、直观地描述出来,应探索新的模型来展现版本间复杂的关联关系,以便设计者从整体上把握产品设计过程。

1.2 基于 NoSQL 的 PDM 版本管理

目前,PDM 皆基于关系型数据库(RDB)予以实现,其版本管理自然也采用 RDB 进行管理。随着大数据时代的到来,RDB 对于复杂、多样的数据表现出灵活性差、扩展性差、性能差等不足^[13],采用 NoSQL 作为 PDM 的底层支撑环境是一个有益的探索和研究。从版本的组织模型上看,由于呈现明显的非结构化特征,因而基于 RDB 的 PDM 版本管理对于版本增加、修改、合并等操作显得力不从心。

NoSQL 技术可以从根本上解决 RDB 严格的事务管理所造成的数据库高负载时的效率低下问题;key-value 存储方式使数据库操作不再是多二维表的关联查询,更多的是单表的逐渐查询及单表的简单条件查询,提高了访问速度;灵活的数据模型也解决了非结构化数据的存储问题。对于版本的增加、修改、合并,只需相应改变其键值对即可,既不需遍历多个关系,也不需要修改关系的内容,因而能更好地实现版本间复杂的关系及频繁的修改操作。

2 基于 NoSQL 的 PDM 版本分层图模型

2.1 模型定义

基于有向无环图结构,文中提出了对产品版本模型进行分层处理的思想,即对版本间的逻辑关系按层次进行划分,进而形成版本的分层图模型。其中,节点表示版本单元,两个节点之间的有序关系表示前驱版本和后继版本的继承关系,前驱的版本号、版本所在的层数及后继版本号信息,采用权值的方式进行描述。

定义 1: 分层图。定义分层图 $NW = \{V, E\}$ 为一个二元组, $V = \{v_0, v_1, \dots, v_m\}$ ($m = 1, 2, \dots, m$) 为版本节点的集合, v_0 为初始版本, v_1, \dots, v_m 分别为 v_0 的后继版本。 $E = \{e_1, e_2, \dots, e_n\}$ 为连接边的集合,其中 $e_k = (v_i, v_j)$ ($k = 1, 2, \dots, n; i, j = 1, 2, \dots, m$) 表示版本节点 v_i 指向版本节点 v_j 的连接边, v_i 表示为 v_j 的一个直接前驱节点, v_j 表示 v_i 的一个直接后继节点。

定义 2: 直接前驱版本和直接后继版本的集合。对一个分层图 NW ,若 $V_k^a \subseteq V$,且 $V_k^a = \{v_r \mid (v_r, v_k) \in E\}$ ($v_r \in V$),则 V_k^a 为节点 v_k 的直接前驱版本集,记为 $\text{Pre}(v_k)$;若 $V_k^b \subseteq V$,且 $V_k^b = \{v_r \mid (v_k, v_r) \in E\}$ ($v_r \in V$),则 V_k^b 为节点 v_k 的直接后继版本集,记为 $\text{Post}(v_k)$ 。

定义 3: 版本号。 $WV = \{wv_1, wv_2, \dots, wv_m\}$ 为版本号集合, v_q 为 v_k 的直接前驱版本集,记为 $\text{Pre}(v_k) =$

($v_q : WV_q$)($WV_q \subseteq WV$, 且 WV_q 为 v_q 的版本号); v_h 为 v_k 的直接后继版本集, 记为 $Post(v_k) = (v_h : WV_h)$ ($WV_h \subseteq WV$, 且 WV_h 为 v_h 的版本号)。

定义 4: 图节点的层数。设从根节点到当前节点有 n 多条路径可达, 第 i 条路径的长度为 C_i , 定义图节点的层数 $C = \max\{C_i \mid i = 1, 2, \cdots, n\}$ 。

以航空制造领域为例, 产品结构十分复杂, 因此在生产设计过程中, 中间版本数量大、版本间关系也非常复杂。以一个发动机叶片为例, 其版本间的结构关系如图 2 所示。

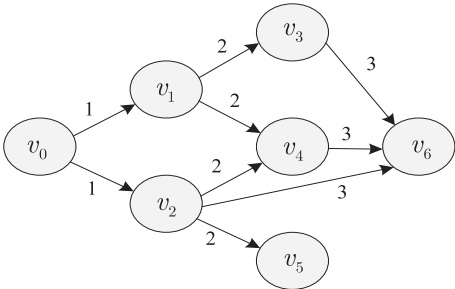


图 2 某发动机叶片的版本关系图

采用以上数学方法可以描述为：
 $V = \{v_0, v_1, \cdots, v_6\}$;
 $E = \{(v_0, v_1), (v_0, v_2), (v_1, v_3), (v_1, v_4), (v_2, v_4), (v_2, v_5), (v_3, v_6), (v_4, v_6)\}$;
当 $k = 4$ 时, $Pre(v_4) = V_4^a = \{v_1:0.1, v_2:0.2\}$, $Post(v_4) = V_4^b = \{v_6:0.6\}$;
 $C = 2$ 。

根据以上定义所构建的分层图结构模型, 集中了当前三种版本模型的优点, 弥补了各自不足, 且能更形象地描述出各版本对象的演变过程及各版本间关系。

2.2 节点组织

版本数据组织是版本模型的构建基础。文中设计了基于 NoSQL(以 MongoDB 为例)的版本节点数据组织, 简化了数据结构和表的数量, 提高了处理效率, 如图 3 所示。

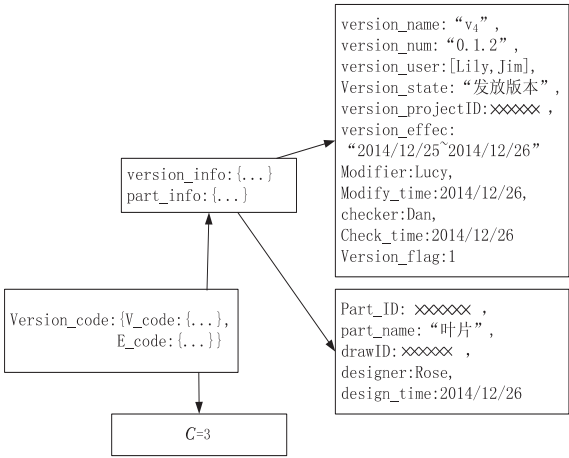


图 3 版本节点组织图

根据分层图模型, 版本节点信息由节点数据和权值数据组成。版本信息及零部件信息存储在节点中, 其中, 版本信息主要包括版本名、版本号、版本使用人、版本状态、该版本所在项目号、版本有效条件、修改人、修改时间等; 零部件信息主要包含零部件 ID 信息、零部件名称信息、图号信息、设计者信息、设计时间信息等。权值数据表示为弧终点节点所在层数。

版本节点在非 RDB 中采用集合嵌套形式进行存储, 而在 RDB 中, 用表结构存储以上节点信息和节点间关系(如表 1 所示)。

表 1 关系数据库表结构

字段名称	类型	长度	备注
version_projectID	char	20	主键
version_name	char	50	
version_num	char	20	
version_user1	char	50	
version_user2	char	50	
version_user3	char	50	
version_state	char	50	
version_effect	Datetime	20	
Modifier	char	50	
Modifier_time	Datetime	8	
checker	char	50	
check_time	Datetime	8	
part_ID	char	20	
part_name	char	50	
drawID	char	20	
designer	char	50	
design_time	Datetime	8	
val	char	4	

在传统 RDB 下, 通常采用的是固定分配数据方式进行存储, 且将数据结构化, 不免导致部分空字段浪费, 且在数据处理过程中, 修改、删除、更新数据要遍历所有关联表, 造成巨大的系统开销, 处理效率严重降低。

综上, 该版本数据组织灵活简单、结构清晰, 使用 NoSQL 进行数据存储更贴近实际操作需求, 方便版本扩展。

2.3 版本存储

文中选择十字链表表作为分层图模型的存储结构, 可以清晰表示版本信息与版本间关系, 且更方便版本追溯的操作。以图 2 为例, 其十字链表存储结构如图 4 所示。

图中, 用十字链表存储版本分层图模型, 可以迅速准确地查找版本的直接前驱版本和直接后继版本。以

版本 v_4 为例,其前驱版本由边节点可迅速查找到 v_1 和 v_2 节点,而后继版本 v_6 可通过 v_4 的顶点节点来追溯获得。

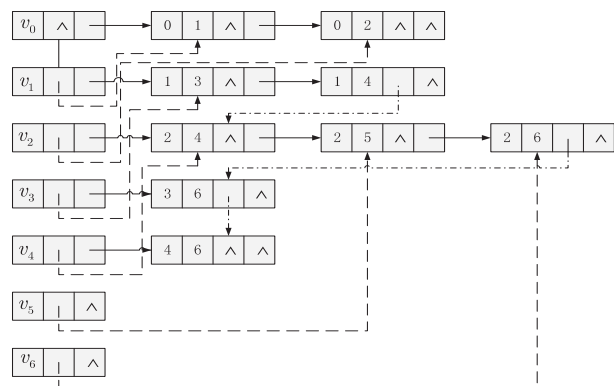


图 4 分层图模型的十字链表存储结构

3 基于 NoSQL 的 PDM 版本分层图模型管理

版本管理是 PDM 系统的重要部分,反映设计过程中设计对象不断演变的动态变化^[2]。因此,对于上述建立的版本分层图模型,需进行的管理有版本分层图模型重构、版本分层图模型追溯、版本合并以及版本的有效性管理。版本分层图模型的重构要保证重构的版本模型准确、直观;版本的追溯要保证追溯耗时短且精准;版本的合并要保证版本间的内容与关系准确;版本的有效性控制要实时保证版本有效。

3.1 版本分层图模型的重构

版本结构图信息完全记录在数据库中,但为了直观、形象地显示版本之间的发展历程,需要根据数据库中的记录重构出版本间的关联结构图^[14]。

最简单的重构方法是:依据 NoSQL 数据库 (Key_Value) 键值对的存储特点,对零部件及版本信息进行相应的检索,记录节点检索顺序、版本节点信息和版本节点关系信息。这样做的目的是保证版本分层图模型重构的准确。但是这种方式的不足之处在于节点位置混乱,且结构不规整。

为保证重组的图结构模型直观、完整,对版本分层图模型进行重构,使节点均匀、准确地分布在各个层之间而不发生混乱。

假设图模型有 n 层,定义: D_x 为节点的横向显示距离常量; D_y 为节点的纵向显示距离常量。初设定坐标 $(0, O_y)$ 为初始版本信息,字母 i 表示为图结构的第几层 ($i = 1, 2, \dots, n$); K_c 表示第 c 层的节点数; S_c^m 表示节点 v^i 在其所在层上的序列号, $S = 0, 1, \dots, K_i$, 其中 $m = 0, 1, \dots, N - 1$ 。(V_x^i, V_y^i) 表示节点 v^k 显示的 x, y 坐标位置。

$$\begin{cases} v_x^i = C D_x \\ v_y^i = \begin{cases} O_y, i = 0 \\ (S_c^m - \frac{K_c}{2} - \frac{S_c^m - \frac{K_c}{2}}{2 \lfloor S_c^m - \frac{K_c}{2} \rfloor}) D_y + O_y, K_c \% 2 = 0 \\ (S_c^m - \frac{K_c + 1}{2}) D_y + O_y, K_c \% 2 = 1 \end{cases} \end{cases} \quad (1)$$

根据 v^k 的坐标值 (V_x^i, V_y^i) 确定该版本节点的位置,并由十字链表中前驱版本和后继版本集合确定版本间关系,即可实现分层图结构模型的重构。

3.2 基于分层图模型的版本追溯

版本追溯需对版本演变的历史过程进行快速、准确地追溯。基于版本分层图结构模型的版本追溯步骤如下:

(1)检索位置:通过版本分层图结构模型检索版本所在的位置。

(2)查找:根据版本十字链表存储特点,依次查找该版本的前驱版本集合与后继版本集合。

(3)读取:依次读取该版本所有前驱版本与后继版本的版本号,并记录版本间关系,实现版本追溯。

采用十字链表存储结构,方便快速追溯版本历程。随着版本数量的增加,十字链表机构特点变得更有优势。

3.3 基于网结构模型的版本合并

在零部件设计过程中,对已存在的两个或两个以上的版本进行方法、思想的合并,或是内容的合并,产生新的版本,即为版本合并。图 5 为版本合并图。

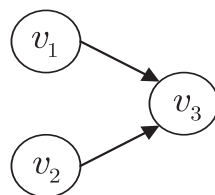


图 5 版本合并图

版本合并步骤如下:

(1)分配版本号:首先根据版本分层图模型确定最新版本号,然后由版本号分配规则,进而确定合并版本的版本号。

(2)添加:在合并版本节点信息中,添加该版本零部件信息、版本属性信息与直接前驱版本信息、直接后继版本信息。

(3)插入:修改该合并版本的直接前驱版本节点的后继节点信息,插入该合并版本。

版本合并中添加前驱版本号集合,只需在 MongoDB 数据库中添加相应集合,而在关系型数据中不仅

仅是添加了一条数据,还要对具有该属性的多个表结构进行相应修改。因此,对于版本合并特点,NoSQL 数据库更为适合。

3.4 基于版本分层图模型的有效性控制

在生产设计过程中,产品的零部件信息修改频繁,导致一个零部件具有多个版本。并且客户需求使得零部件版本的有效性不同。对版本的有效性进行配置,是版本管理的重中之重^[15]。

有效性是更改的生效零部件进行控制,零部件的更改和具体客户是紧密相联的,且零部件的每次更改都会产生相应的版本。一个更改应用到哪些产品,或更改对于产品的生产流程有变化,必须对相应的产品进行版本的有效性配置,以便及时对产品过程进行调整。版本有效性是一个中心环节,将更改、客户、构型等信息组成一个有机结合的统一整体。因此,有效性是体现并落实更改的必须途径和唯一手段。

版本的有效性,它是一个相对概念,即一段时间内,版本有效,它并不是持续连续的。版本有效性通过相应的产品有效体现。

零部件之间的相互整合进而形成产品,根据零部件及其版本的有效性不同,直接影响产品的构成。所以对版本有效性控制的管理,是保证产品生产的必要条件。

4 基于 NoSQL 的 PDM 版本分层图模型维护

对版本分层图模型进行维护,是保证模型持久正确的唯一手段。由于版本的自身特性,版本的维护只有增加、修改操作。所有版本均不能被删除。

4.1 版本的添加

添加一个新版本时,判断该版本是否为合并版本。对相应的版本做出相应的方法选择。

(1)非合并版本的添加。首先对版本模型进行分析,得到当前版本的版本号,再根据版本号规则,进而确定新版本的版本号。然后确定版本添加位置,添加新版本。以图 2 为例,新版本为 v_3 节点下的版本 v_7 。代码如图 6 所示(以 MongoDB 为例)。

(2)合并版本的添加。需根据版本合并方法进行处理。

4.2 版本的修改

版本修改主要包括两点:一是对版本中零部件信息的修改;二是对版本的属性信息进行相应修改。

(1)对版本零部件信息的修改,需保留修改前的版本。因此,要增加版本节点,生成新版本,不可直接更改版本内零部件信息。

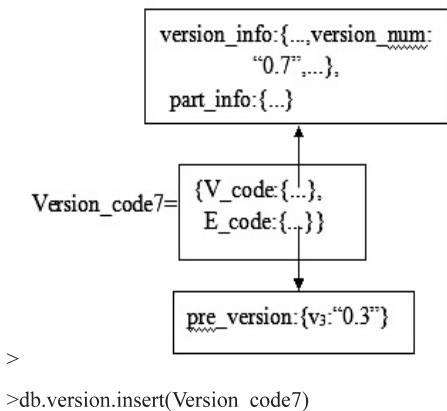


图 6 版本添加代码

(2)对版本属性信息的修改,无需保留修改前版本,可直接对其进行修改,覆盖原有文件。例如,节点版本 v_8 ,版本号设置错误,只需直接修改版本号即可。代码如下:

```

>db.version.update ( { version_name: "v8" }, ... { "$ set " : { version_state: "发放版本" } } )

```

5 结束语

基于 NoSQL 数据库的 PDM 版本管理是文中研究内容的重点。采用分层图模型对复杂版本间的关系进行处理,这样可以直观、清晰地表示版本演变过程,使结构更加合理。基于版本数据特点,采用 NoSQL 数据库,更方便对版本进行版本追溯、版本合并以及版本有效性控制。最终实现对产品设计流程的掌握,更好地对设计过程中的版本进行管理。

参考文献:

- [1] 张劲松,刘清华,钟毅芳,等. 基于 PDM 的版本管理研究[J]. 华中科技大学学报:自然科学版,2001,29(12):33-36.
- [2] 高奇微,莫欣农. 产品数据管理(PDM)及其实施[M]. 北京:机械工业出版社,1998.
- [3] 冯国奇,崔东亮,王成恩,等. 一种基于网状版本模型的复杂产品设计数据管理方法研究[J]. 管理工程学报,2009,23(1):82-87.
- [4] 童秉枢,李建明. 产品数据管理(PDM)技术[M]. 北京:清华大学出版社,2000.
- [5] Lee S W, Ahn J H, Kim H J. A schema version model for complex objects in object oriented databases[J]. Journal of System Architecture, 2006, 52(10):563-577.
- [6] Danilovic M, Browning T R. Managing complex product development projects with design structure matrices and domain mapping matrices[J]. International Journal of Project Management, 2007, 25(3):300-314.
- [7] Oshri I, Newell S. Component sharing in complex products and systems: challenges, solutions, and practical implications[J].

环数(即节点度)在一定范围内,随机步长得到的直径、平均距离均小于传统固定步长直径下界、平均距离下界,则随机步长网络节点步长长度较小,从而得出结论:在无向环网中,随机步长较传统固定步长通信延迟占有优势。

5 结束语

文中采用随机步长来构造无向环网,分别对随机步长无向环网直径、平均距离和固定步长无向环网直径下界、平均距离下界进行仿真实验对比。结果表明,在保持无向环网中其他因素相同时,当节点度在一定范围内,随机步长无向环网的网络通信延迟和网络性能均优于固定步长无向环网,同时随着网络节点数不断增加,即网络规模的不断增大,随机步长降低通信延迟的优势更加明显。因此,与当前主流的 HPC 采用的高节点度和固定步长拓扑结构相比,随机步长拓扑可成为下一代高性能计算机的潜在拓扑结构。

参考文献:

[1] 周兴铭. 高性能计算技术发展[J]. 自然杂志,2011,33(5): 249-254.

[2] Chen C Y, Hwang F K. Equivalent L-shapes of double-loop networks for the degenerate case[J]. Journal of Interconnection Networks,2000,1(1):47-60.

[3] 陈协彬. 步长有限制的双环网络的最优路由算法[J]. 计算机学报,2004,27(5):596-603.

[4] Hwang F K. A survey on multi-loop networks[J]. Theoretical Computer Science,2003,299(1):107-121.

[5] 方木云,屈玉贵,赵保华. 双环网络的[+h]边优先寻径策略[J]. 计算机学报,2008,31(3):536-542.

[6] 苏小虎,方木云,邵伟鹏,等. 双环网络 $G(N;1,s)$ 的 L 形瓦仿真算法改进[J]. 小型微型计算机系统,2012,33(9):

2053-2055.

[7] 方木云,赵保华,屈玉贵. 非单位步长双环网络 $G(N; r, s)$ 的 L 形瓦仿真算法[J]. 系统仿真学报,2006,18(10):2963-2965.

[8] Wong C K, Coppersmith D. A combinatorial problem related to multi-module memory organizations[J]. Journal of ACM, 1974,21(3):392-402.

[9] Koibuch M, Matsutani H, Amano H, et al. A case for random shortcut topologies for HPC interconnects[C]//Proc of 39th annual international symposium on computer architecture. [s. l.]:IEEE,2012:177-188.

[10] Kim J, Balfour J, Dally W. Flattened butterfly topology for on-chip networks[C]//Proceedings of the 40th annual IEEE/ACM international symposium on micro-architecture. [s. l.]: IEEE Computer Society,2007:172-182.

[11] 方木云,侯海金,吴爱清,等. 双环网络直径点和宽直径点的分布特性[J]. 小型微型计算机系统,2013,34(4):749-752.

[12] 陈业斌,李颖,郑啸,等. 关于有向环网平均直径的研究[J]. 通信学报,2013,34(2):138-146.

[13] Fujiwara I, Koibuchi M, Casanova H. Cabinet layout optimization of supercomputer topologies for shorter cable length[C]//Proc of 13th international conference on parallel and distributed computing, applications and technologies. [s. l.]: IEEE,2012:227-232.

[14] Koibuchi M, Fujiwara I, Matsutani H, et al. Layout-conscious random topologies for HPC off-chip interconnects[C]//Proc of 19th international symposium on high performance computer architecture. [s. l.]:IEEE,2013:484-495.

[15] 方木云,赵保华,屈玉贵. 基于圈的紧优双环网络 $G(N;1,s)$ 求解算法[J]. 华中科技大学学报:自然科学版,2005,33(6):17-19.

[16] 方木云,赵保华. 新的无向双环网络 $G(N;\pm 1,\pm s)$ 直径求解方法[J]. 通信学报,2007,28(2):124-129.

(上接第 26 页)

IEEE Transactions on Engineering Management,2005,52(4): 509-521.

[8] Ba E H, Hu W C, Yo O W S. Document configuration control processes captured in a workflow[J]. Computers in Industry, 2004,53(2):117-131.

[9] Meziane F, Rezgui Y. A document management methodology based on similarity contents[J]. Information Sciences,2004, 158(1):15-36.

[10] 徐立臻,徐宏炳. 面向对象数据库系统中的版本管理[J]. 东南大学学报:自然科学版,1999,29(3):34-38.

[11] 张维,何卫平,刘平,等. PDM 实施中的版本管理研究与应用[J]. 组合机床与自动化加工技术,2001(6):11-12.

[12] 刘方鑫,李毅. 变量化图形 CAD 系统中的版本管理[J]. 计算机应用,2000,20(S):111-112.

[13] 陆嘉恒. 大数据挑战与 NoSQL 数据库技术[M]. 北京:电子工业出版社,2013:7-8.

[14] 冯向兵,莫蓉,桂元坤. PDM 中版本管理的图模型表达方法与实现技术[J]. 航空制造技术,2008(12):89-92.

[15] 夏秀峰,富钢,丛丽晖,等. 一种基于 SQL 的层次查询方法[J]. 微处理机,2001(1):42-44.