文章编号：1005-8451（2020）07-0030-06

基于大数据计算模型的CBTC软件智能测试系统技术研究

王　超[1,2]，张德明[1]，徐　伟[1]，宋　欣[1]
（1.中国铁道科学研究院集团有限公司　通信信号研究所，北京　100081；2.轨道交通系统国家测试工程实验室，北京　100081）

摘　要：为解决国产基于通信的列车控制（CBTC，Communication Based Train Control）系统软件测试质量不高和效率低等问题，设计了基于大数据计算模型的CBTC软件智能测试系统。该系统构建一种多元线性回归软件度量模型，制定合理的软件度量标准，对软件质量进行评估，在此基础上，提出高效的基于Hadoop开源框架的分布式度量方法，用以解决大数据计算任务，并提出一种智能软件测试系统整体解决方案，实现软件测试的智能化运行。通过实验分析，该系统能够合理生成软件度量模型，对测试任务进行分布式并行处理，减少测试人员重复性劳动，提高测试人员的工作质量与效率。

关键词：智能测试；软件度量模型；大数据计算；多元线性回归；CBTC
中图分类号：U284.482 : U285 : TP39　**文献标识码：**A

CBTC software intelligent test system technology based on big data computing model

WANG Chao[1,2], ZHANG Deming[1], XU Wei[1], SONG Xin[1]
(1. Signal & Communication Research Institute, China Academy of Railway Sciences Corporation Limited, Beijing 100081, China; 2.National Test Engineering Laboratory of Rail Transit System, Beijing　100081, China)

Abstract: In order to solve the problems of low quality and low efficiency of CBTC (Communication Based Train Control) system software testing, this article designed a CBTC software intelligent testing system based on big data computing model.This system constructed a multiple linear regression software measurement model, established a reasonable software measurement standard to evaluate the software quality. Based on this, the article proposed an efficient distributed measurement method based on Hadoop open source framework to solve the big data computing task, andput forward an overall solution of intelligent software testing system to implement the intelligent operation of software testing. Through the experimental analysis, the system can reasonably generate software measurement model, carry out distributed parallel processing of test tasks, reduce the repetitive labor of testers, and improve the quality and efficiency of testers.

Keywords: intelligence test; Software metrics model; Big data computing; Multiple linear regression; CBTC

随着铁路行业的快速发展，相关软件的安全性问题已经成为人们关注的重点。铁路行业发生的一些重大事故，大部分与相关系统的软件缺陷有关。而系统的缺陷会随着软件复杂度的增加呈指数级上升，这也直接提高了软件测试的难度和成本。软件测试的实际应用和理论研究都有广阔的市场前景。

美国软件工程实验室制定了软件度量模型与标准[1-3]，用于指导美国航空航天局（NASA）的软件开发并作为产品软件安全与任务的保障。城市轨道交通领域的国产列车自动控制系统（CBTC，Communication Based Train Control System）软件有其自身的度量特性，并不完全适合应用此标准。现阶段，国产CBTC软件测试存在大量频繁迭代和建模过程，会产生海量的计算任务，缺少针对相应软件测试系统的大数据[4-5]分布式计算[6-7]领域的研究。此外，国产CBTC软件测试系统对于复杂度各异的测试项目，存在测试方案繁琐，测试成本过高，测

收稿日期：2019-07-15
基金项目：中国铁道科学研究院集团有限公司基金资助（2019YJ072）
作者简介：王　超，助理研究员；张德明，副研究员。

试与开发沟通效率过低等诸多问题，缺少一种集成度较高的智能解决方案。本文提出的基于大数据计算模型的CBTC软件智能测试系统可以较好地解决上述问题。

1　软件度量模型

本文基于多元线性回归构建软件度量模型[8-10]。本模型采用国产CBTC应用项目的历史数据，参考铁路控制和防护系统软件标准EN50128，铁路应用可靠性、可用性、可维护性、安全性技术条件和验证标准EN50126，工业标准化编程规范MISRA-C。利用统计学方法，制定满足国产CBTC的软件代码质量参数标准，并对其中每个参数的影响因子与影响系数进行定量分析，实现对实际工程项目软件质量各项指标的细化评价，便于测试与开发人员对影响软件代码质量的参数进行预测与调整。

1.1　度量模型原理

多元线性回归度量模型由使用因数、度量项、度量结果3级结构构成，通过选择各级评价因素及其加权值的方法来考察多个变量间的相关性与强度，对被测软件及其子程序进行评价。评价软件质量特性公式为：

$$Q = \sum_{j=1}^{M} V_j \cdot \left(\sum_{i=1}^{N_j} W_i \cdot X_i\right) \tag{1}$$

其中，Q 为质量特性；M 表示软件质量特性的使用因数数量；V_j 为第 j 个使用因数的权重；N_j 表示属于第 j 个使用因数的质量度量项的总项数；W_i 表示当前度量项 i 的权重，同一特征下所有度量元权重之和为1；X_i 表示当前度量项 i 的度量结果。

在实际的软件测试项目中，往往需要对多个软件模块或对软件模块内部的多函数质量特性进行评估，发现强耦合等软件质量问题，需要引入多级软件质量评估。评价软件质量特性公式为：

$$\begin{aligned} Q_g = &\sum_{i=1}^{M} V_i g \cdot \sum_{u=1}^{R} W_u g \cdot X_u g \\ &+ \sum_{j=1}^{N} V_j g \cdot \sum_{v=1}^{S} W_v g \cdot X_v g \\ &+ \sum_{k=1}^{P} V_k g \cdot \sum_{z=1}^{T} W_z g \cdot X_z g \end{aligned} \tag{2}$$

其中，Q_g 为多级软件质量特性；$1 \leqslant g \leqslant m$；$m$ 为多级评估模块的数量；M、N、P 分别为软件质量特性的使用因数数量；R、S、T 分别为当前使用因数的质量度量项的项数。

1.2　度量模型参数标准

1.2.1　度量分类

本文根据EN50128、EN50126、MISRA-C标准的要求，结合CBTC的实际软件设计需求和项目测试数据的影响分析，制定模型的软件质量评价标准，分类为可测试性、可维护性、清晰性。表1为广州7号线CBTC车载软件质量部分度量标准。

表1　广州7号线CBTC车载软件质量部分度量标准

度量分类	权重	度量项	英文简称	度量项权重	标准极小值	标准极大值
可测试性	0.3	全局变量数量	*UT NoG*	0.126 0	0	50
		基础块数量	*Blks*	0.135 0	1	30
		循环数量	*Tot Loops*	0.158 3	0	4
		扇入	*Fan in*	0.210 5	0	5
		扇出	*Fan out*	0.200 6	0	5
可维护性	0.5	基本节点数	*Eknts*	0.153 9	0	2
		基本圈复杂度	*EV(G)*	0.120 0	1	3
		节点数	*Knots*	0.196 0	0	5
		圈复杂度	*V(G)*	0.138 6	1	10
清晰性	0.2	总注释量	*Tot Comm*	0.050 0	10	200
		头文件注释量	*Head*	0.100 0	5	50
		声明中注释量	*Decl Comm*	0.018 0	0	100
		可执行代码中注释量	*Exe Comm*	0.100 0	1	100
		空行数量	*Blnk Comm*	0.036 0	0	100
		循环深度	*Nest Dept*	0.150 0	0	2

1.2.2　度量结果

本文以广州7号线CBTC车载软件的filter.c模块为例，对其中9个函数的10项软件质量度量参数进行分析，如表2所示，表中，P表示该项符合标准，F表示该项不符合标准。

2　大数据分布式计算

由于开发测试阶段存在的迭代过程，在软件实际测试过程中，需要频繁修改代码，这些并行处理的测试任务的代码量甚至可以达到百万级代码行。由软件质量度量模型的构建过程可知，涉及到的评估参数数量和评估对象的数量巨大，对这些评估参数的计算过程会占用大量的运算资源。

表2　广州7号线CBTC车载软件filter.c模块软件质量度量结果

函数名	Knots	Eknts	V(G)	EV(G)	Blks	Tot Loops	Nest Dept	Fan in	Fan out	UT NoG
FilterFun_Init	0(P)	2(P)	1(P)	1(P)	1(P)	14(F)	2(P)	1(P)	1(P)	1(P)
FilterFun_Release	1(P)	1(P)	1(P)	1(P)	1(P)	11(F)	2(P)	1(P)	1(P)	0(P)
Process	0(P)	1(P)	4(P)	1(P)	8(P)	3(P)	1(P)	1(P)	1(P)	0(P)
STRFilter_create	0(P)	1(P)	3(P)	1(P)	6(P)	4(P)	1(P)	1(P)	1(P)	0(P)
STRFilter_free	2(P)	2(P)	1(P)	1(P)	1(P)	3(P)	1(P)	1(P)	1(P)	0(P)
process_by_rate	7(F)	6(F)	5(P)	1(P)	14(P)	3(P)	1(P)	1(P)	5(F)	0(P)
process_by_value	6(F)	6(F)	5(P)	4(F)	13(P)	3(P)	1(P)	1(P)	4(F)	0(P)
ExpFilter_create	1(P)	1(P)	3(P)	1(P)	6(P)	4(P)	1(P)	1(P)	1(P)	0(P)
ExpFilter_free	0(P)	1(P)	1(P)	1(P)	1(P)	3(P)	1(P)	1(P)	1(P)	0(P)

把测试任务运行在单一服务器组的传统工作方式已经不能满足海量数据的运算与测试需求。所以本文针对软件测试中存在的海量数据复杂计算问题，基于开源 Hadoop 软件体系架构[11-14]，提出 Distributed_Metrics 方法，并搭建 Hadoop 服务器测试集群[15-16]，将业务处理逻辑在集群内并行执行，实现高效快速的软件质量度量模型运算。

2.1　数据分布式解析

2.1.1　测试数据切片

用户提交测试文件到 Hadoop 服务器集群目录文件中，集群中的客户端程序会扫描磁盘中的目录文件，遍历其中需要测试的每个文件，并根据预先配置的参数，进行测试数据切片。本文以广州 7 号线 CBTC 车载软件 filter.c 模块为例进行测试数据切片，如图 1 所示。

图1　CBTC车载软件测试数据切片

集群中的 MRAppMaster 节点会根据切片文件的大小自动启动集群中的 maptask，并把切片按照规则分组到指定的 maptask 中去，如图 2 所示。

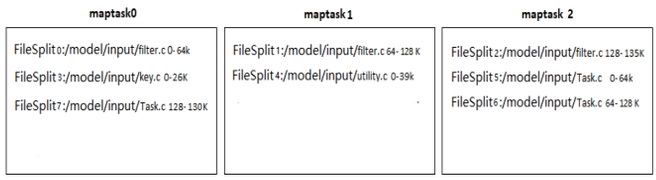

图2　数据切片划分

Distributed_Metrics-Map 方法对数据进行分布式切片分组的流程如下。

输入参数：待处理文件 f。

输出结果：分组数据 m。

开始：

（1）上传文件 f 到测试服务器集群；

（2）服务器集群启动客户端程序扫描磁盘目录；

（3）遍历目录，查看是否存在新的需要切片的文件，如果不存在，则流程结束，否则继续下一步；

（4）根据配置参数遍历 f 文件，对其进行切片，如果切片完成，则转到步骤（3），否则继续下一步；

（5）利用 MRAppMaster 节点将切片文件分配到相应的 maptask 中去；

（6）转到步骤（4）；

流程结束。

2.1.2　测试数据缓存区优化

软件质量度量评估的运算复杂度主要取决于评估矩阵中各元素的统计方式和其相应的计算，而评估矩阵主要的工作是计算各阶段的向量内积并进行累加。针对该情况，本文采用 key–value 键值对的方式模拟软件质量评估矩阵的内积累加操作。矩阵中每个元素的位置具有唯一性，标记为 key 值，相应的元素值标记为 value。每个矩阵向量都是针对模块中某具体函数进行分析，其相应的函数名标记为 function，则统计的数据格式为 $(function)\{key\ 1:value\ 1, key\ 2:value\ 2, \cdots, key\ n:value\ n\}$。

在集群中启动多个 maptask 会统计出大量的 key–value 键值对，为便于后续的运算，本文对 Hadoop 环形内存区进行优化，具体优化过程如图 3 所示。

在第 1 次写入点位置将 key–value 键值对写入到环形缓冲区中，此时写入的键值对是无序且没有

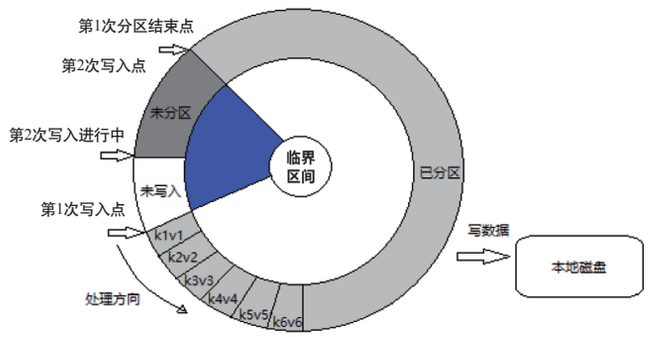

图3　环形内存区优化过程

进行分区处理的。为了最大化利用分布式系统的计算效率，需要对环形内存区的键值对进行标记分区，对于不同分区的数据可以分配给集群中的不同节点，从而达到最大化并行计算处理的效果。在环形内存区设置一个临界区间，例如，将内存数据达到总内存数据量的 80% 及以上的区间设置为临界区间，为防止数据溢出，会启动数据分区线程任务对键值对进行标记分区。

将已经处理好的标记分区内容写入本地磁盘并清空分区数据缓冲区。以第 1 次分区结束点的位置为起始点开始第 2 次写入数据，在未进入临界区间前可以循环写入键值对到环形内存区中。

2.1.3　测试数据哈希分区

2.1.2 中为每组键值对标记了函数名 function，本文提出一种特定的哈希算法对其进行分区归类。函数经过分片以后，每个分片的计算量很小，不同分片的耗时差别可以忽略不计，最终影响计算性能指标的是集群中每台服务器的函数分片总数量。而对于同一函数的不同分片在经过哈希算法分区后必须分到同一分区进行处理，以保持数据的有效性。这就要求哈希算法尽可能把函数分片按照合理的方法均分到每台服务器中。由于 key-value 中 function 是非重复的，本文根据字母表 A-Z、a-z 与 ASCII 码表的对应关系，把所有的函数名转换为十六进制序列，每组序列可以唯一标识函数名。在哈希分区的过程中，会对每位字母的十六进制序列进行累加，确保拥有相同 function 的分片数据可以分到同一分区中，并尽可能保证集群中分区的平衡性。广州 7 号线 CBTC 车载软件 filter.c 模块数据分区如表 3 所示。

表3　广州7号线CBTC车载软件filter.c模块数据分区

切片文件	函数名	码表转换	哈希分区
FileSplit0:/model/input/filter.c	FilterFun_Init	\u0046\u0069\... \u00741	partition0
FileSplit0:/model/input/filter.c	FilterFun_Release	\u0046\u0069\... \u0065	partition1
FileSplit0:/model/input/filter.c	Process	\u0070\u0072\... \u0073	partition2
FileSplit0:/model/input/filter.c	STRFilter_create	\u0053\u0054\... \u0065	partition3
FileSplit0:/model/input/filter.c	STRFilter_free	\u0053\... \u0065\u0065	partition0
FileSplit1:/model/input/filter.c	STRFilter_free	\u0053\... \u0065\u0065	partition0
FileSplit1:/model/input/filter.c	process_by_rate	\u0070\u0072\... \u0065	partition1
FileSplit1:/model/input/filter.c	process_by_value	\u0070\... \u0075\u0065	partition2
FileSplit1:/model/input/filter.c	ExpFilter_create	\u0045\u0078\... \u0065	partition3
FileSplit1:/model/input/filter.c	ExpFilter_free	\u0045\... \u0065\u0065	partition1
FileSplit2:/model/input/filter.c	ExpFilter_free	\u0045\... \u0065\u0065	partition1

本文对数据切片进行哈希分区的方法 Distributed_Metrics-Hash_Partition 流程如下。

输入参数：待处理分组数据 m。

输出结果：哈希分区数据 h。

开始：

（1）将 m 写入到环形缓冲区；

（2）按照既定顺序扫描环形缓冲区数据；

（3）若环形缓冲区数据未达到临界点，转到步骤（1），若缓冲区进程结束，则流程结束，否则继续下一步；

（4）按照 ASCII 码表对环形缓冲区内的 function 进行哈希分区；

（5）把分区完成的数据写入磁盘并清除已完成分区的数据；

（6）标记环形缓冲区新的写入起始点，转到步骤（1）；

流程结束。

2.2　数据聚合计算

经过哈希分区存储在磁盘中的中间数据，还需要进行 key–value 键值对的汇总计算。因服务器集群中每台机器都是多线程运行的，在进行 maptask 与哈希分区任务时，可以在汇总服务器启动独立的 reducetask，对磁盘中的中间结果数据进行汇总统计。

分布式计算从磁盘文件中按分区号一次性读取

相应文件到内存中，从而避免将相同分区号的文件提交给不同 reducetask 导致数据统计异常的情况。

本文对数据分布式汇总统计的方法 Distributed_Metrics-Hash_ Reduce 流程如下。

输入参数：待处理的分区数据 h。

输出结果：软件质量特性 Q。

开始：

（1）服务器启动 reducetask；

（2）遍历磁盘中待处理的分区数据 h，如果不存在待处理的分区数据则结束，否则继续下一步；

（3）对磁盘中待处理的分区数据 h 进行逐行累加计算；

（4）输出软件质量特性 Q 到磁盘，转到步骤（2）；

流程结束。

分别为：静态测试服务、单元测试服务、集成测试服务、文档管理服务和 Linux 系统服务。根据预先编写的服务器端自动化测试应用软件，智能化地完成大部分的软件测试工作并生成测试文档。系统架构如图 4 所示。

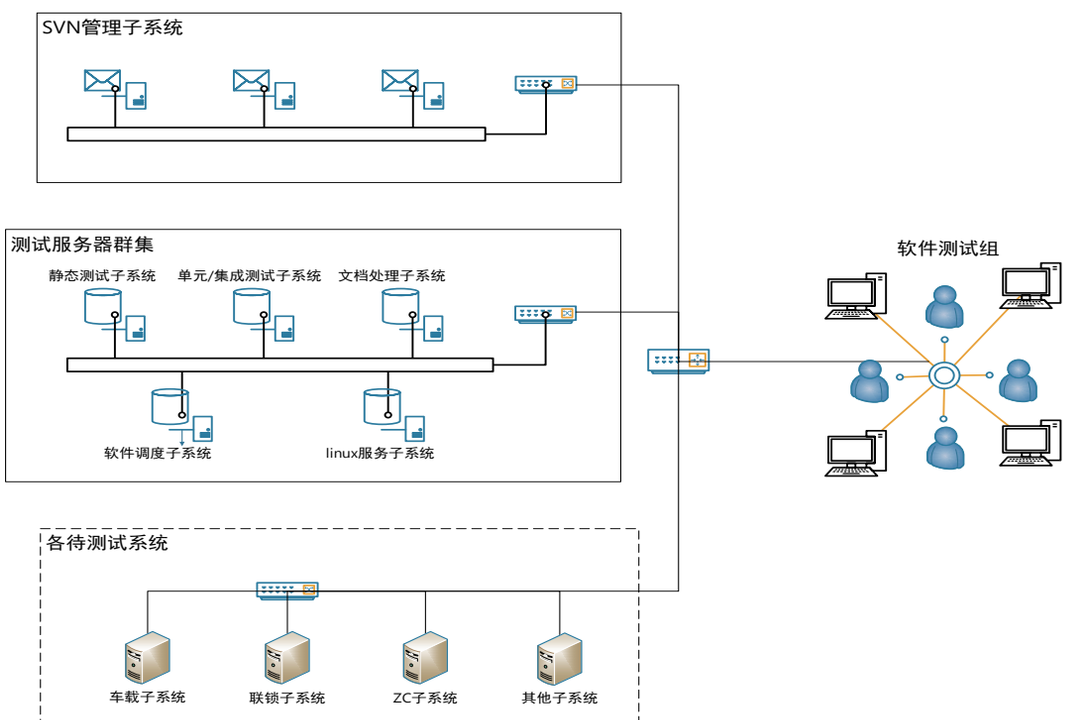

图4 智能软件测试系统架构

3 智能测试系统

3.1 智能测试系统测试流程

（1）各子系统的开发团队上传测试数据到各自的 SVN 数据服务器。智能测试系统通过局域网连接到各子系统的 SVN 数据服务器并下载相应的测试代码。

（2）测试服务器集群调用多元线性回归软件质量度量模型，采用 Distributed_Metrics 方法，对下载的测试代码进行评估。对于不满足度量模型评估的测试代码，开发团队可通过测试服务器平台提供的可视化图形交互界面进行查询与修改，对于满足评估指标的代码可进行下一阶段的测试。

3.2 智能测试系统架构

系统在软件测试服务器中部署的 5 类测试服务

4 结束语

本文构建了适用于 CBTC 软件的多元线性回归软件度量模型，解决了 NASA 的软件度量标准并不完全适用于国产 CBTC 软件的问题；提出一种将 Map_Reduce 分组聚合与哈希分区相结合的、高效的 Distributed_Metrics 方法，可以把海量、复杂的计算任务分配到 Hadoop 服务器集群中进行高效并行处理，提高了整个软件测试集群的运行效率；搭建了一套涵盖软件测试全生命周期的集成测试系统，可以实现 CBTC 软件测试的智能化运行。通过实验分析，该系统能够合理生成软件度量模型，对测试任务进行分布式并行处理，减少测试人员重复性劳动，提高测试人员的工作质量与效率。下一阶段的工作方向是设计出适用性更强，具有自学习能力的软件度量模型，智能化地发现软件质量缺陷。

参考文献

[1] Norman Schneidewind.Updated Software Reliability Metrics[J]. Reliability Review: The R & M Engineering Journal, 2009, 29(4): 5-15, 25.

[2] Gray D, Bowes D, Davey N, et al. The Misuse of the NASA Metrics Data Program Data Sets for Automated Software Defect Prediction[C]//Evaluation and Assessment in Software Engineering (EASE). IET, 2011.

[3] Linda H. Rosenberg, Albert M. Gallo. Software quality assurance engineering at NASA [C]//2002 IEEE Aerospace Conference, BigSky Montana, USA: IEEE, 2002:2569-2575.

[4] Schmidt Bertil, Hildebrandt Andreas. Next-generation sequencing: big data meets high performance computing[J]. Drug discovery today, 2017, 22(4): 712-717.

[5] 代　亮，陈　婷，许宏科，等 . 大数据测试技术研究 [J]. 计算机应用研究，2014，31（6）：1606-1611.

[6] Tomarchio O, Modica G D, Cavallo M, et al. A Hierarchical Hadoop Framework to Handle Big Data in Geo-Distributed Computing Environments[J]. International Journal of Information Technologies and Systems Approach, 2018, 11(1): 16-47.

[7] Liu W, Cui P, Nurminen J K, et al. Special issue on intelligent urban computing with big data[J]. Machine Vision & Applications, 2017, 28(7): 675-677.

[8] R. Shanthi, R. Joel Karunakaran. Corrosion Behaviour of Some Phenyl Thiazole by Multiple Linear Regression Technique [J].Asian Journal of Chemistry: An International Quarterly Research Journal of Chemistry, 2017, 29(10): 2299-2304.

[9] Choi, Jae-SeokKim, Munchurl. Single Image Super-Resolution Using Global Regression Based on Multiple Local Linear Mappings [J]. IEEE Transactions on Image Processing, 2017, 26(6): 1300-1314.

[10] Nazif A, Mohammed N I, Malakahmad A, et al. Regression and multivariate models for predicting particulate matter concentration level[J]. Environmental Science and Pollution Research, 2017, 25(10): 283-289.

[11] Ishak H. A. Meddah, Khaled Belkadi. Parallel Distributed Patterns Mining Using Hadoop MapReduce Framework[J]. International Journal of Grid and High Performance Computing, 2017, 9(2): 70-86.

[12] SnehaShoney Sebastian. Improved Fair Scheduling Algorithm for Hadoop Clustering [J]. Oriental Journal of Computer Science and Technology, 2017, 10(1): 194-200.

[13] Tomarchio O, Modica G D, Cavallo M, et al. A Hierarchical Hadoop Framework to Handle Big Data in Geo-Distributed Computing Environments[J]. International Journal of Information Technologies and Systems Approach, 2018, 11(1): 16-47.

[14] Rathore M M, Paul A, Ahmad A, et al. Hadoop-Based Intelligent Care System (HICS): Analytical Approach for Big Data in IoT[J]. ACM Transactions on Internet Technology, 2017, 18(1): 1-24.

[15] 任广强，舒　敏 . 基于 Hadoop 的 Aprior 算法改进及其在动车组运维的应用 [J]. 铁路计算机应用，2018，27（10）：1-6.

[16] Jian LU, Huanqing DONG, Junwei ZHANG, et al. HWM: a hybrid workload migration mechanism of metadata server cluster in data center [J].Frontiers of Computer Science in China, 2017, 11(1): 75-87.

责任编辑　李依诺