doi:10.16104/j.issn.1673-1891.2017.02.010

基于云计算技术的出错数据恢复技术研究

张沛朋[a],魏 楠[b]
(济源职业技术学院 a.艺术设计系;b.本科教学办公室,河南 济源 459000)

摘要:为了保证数据存储的安全和数据的完整性,研究了基于云计算技术的出错数据恢复技术。首先介绍了云数据完整性检测技术,主要介绍了完整性检测模型和动态云数据完整性检测。并在此基础上研究了云数据恢复技术,提出基于动态分级管理机制的云数据恢复技术。最后,验证了该数据恢复技术的有效性。

关键字:云计算;出错数据;恢复技术
中图分类号:TP309.2 **文献标志码**:A **文章编号**:1673-1891(2017)02-0033-04

Research on Error Data Recovery Technology Based on Cloud Computing

ZHANG Pei-peng[a], WEI Nan[b]
(a.Arts Design Department of Jiyuan Vocational and Technical College;b.Undergraduate Teaching Department, Jiyuan Vocational and Technical College, Jiyuan,Henan 459000,China)

Abstract: In order to ensure the security of data storage and data integrity, this paper studies error recovery technology based on cloud computing technology. Firstly, the integrity detection technology of cloud data is introduced. The integrity detection model and dynamic cloud data integrity detection are introduced. On this basis, cloud data recovery technology is studied, and cloud data recovery technology based on dynamic hierarchical management mechanism is proposed. Finally, the effectiveness of the data recovery technique is verified.

Keywords: cloud computing; error data; recovery technique

0 引言

云计算技术是一种以信息共享为基础的信息处理技术,其可以利用虚拟化技术以及互联网技术将系统资源连接起来并以此提供各种网络服务。显然云计算系统的核心功能就是提供海量数据的管理和存储服务,而在硬件方面云计算系统需要配备高存储和高性能的存储设施,只有这样云计算系统才能够建立行业优势,从而满足用户的存储需求[1]。不可否认,从数据安全的角度来看,云计算系统的发展前景并不明朗,其中欧洲权威信息安全结构就宣称,虽然理论上云计算技术比传统存储技术具有更大的应用价值和研究价值,但是由于缺乏法律法规以及有效的网络治理措施,因此造成云计算技术确实存在一定的安全风险[2]。究其原因,主要是因为云计算提供商具有不确定性,而且对于用户隐私保护、数据恢复以及潜在安全问题等方面无法保证。当用户使用云存储服务时,如何保证存储数据能够及时进行检测,并在发现数据被篡改或被破坏的情况实现数据恢复技术已经成为云计算系统急需解决的问题[3]。本文利用恢复机制来完善云计算系统的安全性机制,通过云计算技术中的数据检测技术来检索数据出错位置,然后再利用数据恢复技术将受到破坏的数据进行恢复。

1 云数据完整性检测技术

1.1 IDBRS 完整性检验模型

采用擦除码技术来检测经过编码的数据完整性和一致性,而数据完整性检测模型主要采用IDBRS 检验模型,该检测模型示意图如图1所示。值得注意的是,IDBRS 检验模型无需进行数据的认证比对工作,由此大大简化检测步骤,而且还能够有效保护数据安全,减少数据计算开支。

收稿日期:2016-12-21
基金项目:2015年度河南省高等学校青年骨干教师资助计划(2015GGJS-282)。
作者简介:张沛朋(1983—),男,河南开封人,讲师,硕士,研究方向:计算机应用。

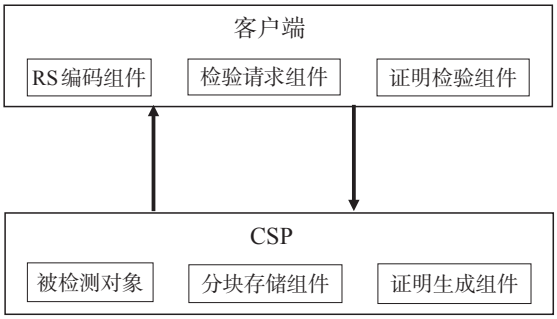

图1 IDBRS数据完整性检测模型

从图1可以看出，IDBRS数据完整性检验模型主要由2大部分组成，分别为客户端部分以及CSP部分，其中客户端拥有云存储数据，同时也是检测证明的发起者，其可以根据系统要求发送完整性检验请求，并对检验结果具有决策权；CSP主要负责用户数据以及提交数据的完整性检测工作。

1.2 动态云数据完整性检测

相对于静态数据，本文研究的主要内容为动态数据，其动态数据安全性检测的核心思想就是保证生成矩阵与码字的一致性。具体来讲，当原始动态数据发生变化时，其原始冗余数据也能随之进行改变，并且改变后的冗余数据也能够与生成矩阵达成一致[4]。如今动态云数据的完整性检测不仅包括冗余数据的认证工作，而且还可以利用验证码来检测数据，但是对于类似数据插入等操作暂时还没有比较有效的完整性检测方案[5]。本文提出的动态数据完整性检测方案无需针对更新数据完整性检测进行额外处理，但是对于数据的插入操作则提供了新型完整性检测方案，其可以显著提高检测效率。动态数据的完整性检测方案就是将基于RS码的生成矩阵作为验证码，而且要求其生成矩阵随着数据的不同而不同，当然针对数据进行操作的同时也会对其生成矩阵进行相应的行列变换，从而让变化后的数据与相应的生成矩阵达到一致，接着只需针对冗余数据进行修改即可，无需计算与数据内容无关的冗余部分，显然这种检测方案可以节约不必要的系统处理资源。

假设$\boldsymbol{C}$表示为基于$RS(k+m,k)$码的生产矩阵，而$\boldsymbol{P}$表示为由$\boldsymbol{C}$得到的检测矩阵，且$\boldsymbol{P}=(S_1,S_2,S_3,\cdots,S_k)$，需要处理的原始数据为$F=(F_1,F_2,F_3,\cdots,F_k)^T$，改变后的数据为$F'=(F_1+\Delta F_1,F_2+\Delta F_2,F_3+\Delta F_3,\cdots,F_k+\Delta F_k)^T$，其对应的冗余数据表示为$D'=(D_1+\triangle D_1,D_2+\triangle D_2,D_3,\cdots,D_k+\triangle D_k)^T$，同时要求满足：$C*F'=(F',D')^T=(F_1+\triangle F_1,F_2+\triangle F_2,F_3+\triangle F_3,\cdots F_k+\triangle F_k,D_1+\triangle D_1,\cdots,D_m+\triangle D_m)^T$

下面将从数据添加、数据修改以及数据删除等操作进行详细说明，比如当进行数据修改操作时，可以只针对参数F_2进行修改，假设修改后的数据表示为$F'=(F_1+O,F_2+O,F_3+O,\cdots,F_k+O)^T$，为了便于后续的完整性检测，要求修改后的数据满足下列条件：

$$C*F'=(F',D')^T=(F_1+O,F_2+O,F_3+O,\cdots,F_k+O,D_1+\Delta D_1,\cdots,D_m+\Delta D_m)^T$$

其中$\Delta D=(\Delta D_1,\Delta D_2,\cdots,\Delta D_m)^T$，并利用等式$\triangle D=S_2*\triangle F_2$计算得到修改后的冗余数据满足$D'=D+\triangle D$。

当相应的冗余数据工作完成之后，其经过修改的数据也要进行完整性检测，当然数据删除操作也和数据添加操作类似。同样以数据F_2为例子，其F_2经过删除处理的数据满足$C*F'=(F',D')^T=(F_1+O,F_2+O,\cdots,F_k+O,D_1+\triangle D_1,D_2+\triangle D_2,\cdots,D_m+\triangle D_m)^T$

式中经过删除处理的冗余数据表示为$D'=D-S_2*F_2$。

第三种数据处理情况为数据添加，也就是在数据链尾部添加数据。为了讨论的方便，本文首先在数据链末尾利用数字0占据将要添加数据的位数，$\boldsymbol{C'}$表示为基于$RS(k+m+1,k)$码的生成矩阵，并由$\boldsymbol{P'}=(S'_1,S'_2,S'_3,\cdots S'_k,S'_{k+1},)$，$\boldsymbol{F}=(F_1,F_2,F_3,\cdots,F_k,0)^T$得到$\boldsymbol{C*F}=(F_1,F_2,F_3,\cdots,F_k,0,D_1,D_2,D_3,\cdots,D_k)^T$。

需要说明的是，由于添加了数字0，因此在数据链尾部添加数据的行为等同于在数据链末尾修改数据操作，这样就可以利用数据修改的方式实现数据添加行为。假设需要添加的数据内容定义为X，也就是将修改后的数据定义为$F'=(F_1,F_2,F_3,\cdots,F_k,X)$，此时只用计算$D'=D+S'_{K+1}*X$，而且$D'$要满足条件$C'*F'=(F_1,F_2,F_3,\cdots,F_k,X,D_1',D_2',\cdots,D_m')$。

文章探讨的数据处理方案主要针对数据的添加、修改以及删除操作，而如果要在数据链中段插入数据，就不能采用本文探讨的方案，比如要在数据F和F_2之间添加数据X，那么就需要首先考虑由于添加数据X而产生的冗余数据变化，其次要考虑添加数据X后引起的数据库的移位变化，当然还要进一步更新冗余数据。由于在中段添加数据X带来过多的后续问题，至今还没有给出较为有效的解决方案，因此主要尝试给出一种基于动态数据的添加操作处理，其方案如下：

首先针对数据序列$F=(F_1,F_2,F_3,\cdots,F_k)^T$，具体关系式如下：

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ p_{0,0} & p_{0,1} & \cdots & p_{0,k} \\ p_{1,1} & p_{1,1} & \cdots & p_{1,k} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m-1,1} & p_{m-1,1} & \cdots & p_{m-1,k} \end{pmatrix} * \begin{pmatrix} F_1 \\ F_2 \\ F_3 \\ \vdots \\ F_k \\ 0 \end{pmatrix} = \begin{pmatrix} F_1 \\ F_2 \\ \vdots \\ F_k \\ 0 \\ D_1 \\ D_2 \\ \vdots \\ D_m \end{pmatrix}$$

为了让更新的数据与生成矩阵在行列结构上保持一致,因此需要在原始数据后面添加0以进行补充。假设需要在数据F_2上添加数据X,那么数据X就正好位于数据F_2与数据F之间,而其后面的数据序列统一向后移动一行,将生成矩阵$\boldsymbol{p}$的第k+1列数据移至第3列,而原来第3列及以后数据统一向后移动一列,其添加数据关系式如下所示:

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ p_{0,0} & p_{0,1} & \cdots & p_{0,k-1} \\ p_{1,0} & p_{1,1} & \cdots & p_{1,k-1} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m-1,0} & p_{m-1,1} & \cdots & p_{m-1,k-1} \end{pmatrix} * \begin{pmatrix} F_1 \\ F_2 \\ X \\ F_3 \\ \vdots \\ F_k \end{pmatrix} = \begin{pmatrix} F_1 \\ F_2 \\ X \\ F_3 \\ \vdots \\ F_k \\ D_1 + \Delta D_1 \\ \vdots \\ D_m + \Delta D_m \end{pmatrix}$$

这时的动态数据需要利用下列公式进行计算:

$$D' = D + S_{k+1} * X$$

基于动态数据添加方案的具体计算公式如下:

$$D' = D + [S_3 * (X - F_3) + S_4 * (F_3 - F_4) + S_5 * (F_4 - F_5) + \ldots + S_{k+1} * F_k]$$

需要说明的是,算法的时间复杂也会随着数据的增加而呈现线性增长趋势。相对而言,采用的算法的复杂度为$O(1)$,而且不会受到数据块变化的影响,因此可以有效提高计算效率。

此外,针对更新数据后的生成矩阵需要满足下列公式:

$$P' * (F_1, F_2, X, F_3, \ldots, F)^T = (D_1', D_2', \ldots, D_m')^T$$

其公式并不影响数据恢复处理和完整性检测功能。

2 云数据恢复技术

2.1 数据恢复原理

假设原始云数据块的块数为参数k,而每个数据块的长度为z,而经过RS码编码处理后的数据块数为n,利用$CodeMatrix_{(n,k)}$表示第n行第k列的生成矩阵(其生产矩阵也称为编码矩阵),$DataMatrix_{(k,s)}$表示第k行第z列的原始数据矩阵,$CodeMatrix_{(n,s)}$表示经过编码后的第n行第s列的数据矩阵,其基于RS码的具体编码过程如下所示:

$$CodeMatrix_{(n,k)} * DataMatrix_{(k,s)} = CodeMatrix_{(n,s)} \quad (1)$$

其中$\mathrm{Re}mainedMatrix_{(k,s)}$表示$CodedMatrix_{(n,s)}$的任意$k$行矩阵,其主要利用$CodedMatrix_{(n,k)}$生成处理得到,并假设经过编码后的矩阵表示为$CodedMatrix'_{(k,k)}$,关系式为:

$$CodeMatrix'_{(k,k)} * DataMatrix_{(k,s)} = \mathrm{Re}mainedMatrix_{(k,s)} \quad (2)$$

其中基于RS码的$CodedMatrix_{(n,s)}$是利用范德蒙德矩阵变换得到,而且该矩阵的k个行向量在线性上为相互独立的,由此构建解码矩阵$DecodeMatrix_{(k,k)}$,其满足下列公式:

$$DecodeMatrix_{(k,k)} * CodeMatrix'_{(k,k)} = 1 \quad (3)$$

以及

$$DecodeMatrix_{(k,k)} * CodeMatrix'_{(k,k)} * DataMatrix_{(k,s)} = DataMatrix_{(k,s)} \quad (4)$$

将数据矩阵代入到公式(4)就可以实现数据解码操作:

$$DecodeMatrix_{(k,k)} * \mathrm{Re}mainedMatrix_{(k,s)} = DataMatrix_{(k,s)} \quad (5)$$

2.2 存储节点动态分级管理机制

当要进行数据恢复处理时,需要将信息数据进行重新分配,由于并不知道数据损坏的原因,所以数据存储的节点具有明显的不稳定性,当然数据存储节点可能受到恶意攻击,因此如果将恢复后的数据继续还原在原来存储位置还有极大可能再次受到攻击[6]。考虑到上述情况,提出了一种基于数据存储节点的动态分级管理模式,该管理模式可以与云数据完整性检测方案进行有机结合,不仅能够提高数据的可靠性和安全性,而且还能够有效提高完整性检测效率。

基于存储节点的动态分级管理模式其实就是一种按照优先级排序的管理机制,其核心思想就是保证服务器群按照优先级优先接受错误率较低和负载能力较低的可靠存储点,并将该存储节点对应的结构体安排在优先级较高的链表上。具体来讲,整个系统对于各个时间有且只有一个全局动态链表,而且每个结构体会按照优先级连接在不同的链表中;各个链表的节点会根据时间的不同而有所差异;各个结构体会与对应级别的动态链表进行联系,以此可以非常方便地管理和维护存储节点,从而保证存储节点的安全性和可靠性;每个节点又会包含有该节点数据的完整性检测记录,而且会根据其检测报告和存储负载能力等因素为其分配合适的动态链表。

2.3 基于动态分级管理机制的云数据恢复技术

借助基于存储节点的动态分级管理模式,在存储节点数据内容中专门添加了节点动态等级信息以及数据完整性信息等,而且新的存储节点也要专门添加上述2种信息数据,因此在针对节点数据进行分配和布局时可以直接参照节点数据等级信息以及数据完整性信息来进行合理分配,这样就可以有效防止数据安排在不合适或者不稳定的存储节

点中，这也对数据起到了部分保护作用[7]。当然系统操作也要根据存储阶段的运行状态进行设定，具体的操作方案如下所示：

（1）基于存储节点的运行状态表维护：当存储节点出现硬件类型的错误时，系统需要将存储节点对应的可用性符号设为0，以此表示该存储节点出现故障；当链表需要添加新的存储节点时，可以将可用性符号为0的存储节点物理地址分配出去，接着将存储节点对应的可用性符号设为1，而且还要将数据完整性符号设为1，最后将存储节点进行初始化，并计算链表剩余存储空间。

（2）链表节点分配以及数据存储：如果用户需要存储服务时，系统首先将数据进行固定分块编码处理，假设每个数据的存储大小为60 MB，这时系统不仅要检验对应的存储节点动态等级信息和完整性检测，而且还要检验存储节点可用性符号是否为1。而当存储数据的完整性符号为1时，系统会专门选择存储节点动态等级最高的节点进行分配存储空间。

（3）故障数据恢复服务：当存储节点出现硬件故障需要恢复数据时，系统可以依据存储节点数据进行恢复，又由于存储节点信息中添加了节点动态等级等信息，因此可以直接根据其等级进行数据恢复操作，从而显著增加数据恢复的有效率。

2.4 优化数据完整性检测效率

当服务器将数据完整性检测数据返回到客户端时，系统可以将存储节点的等级信息添加到该检测值的末尾处，如果存储节点等级共有8个等级，那么只需要3个比特位的二进制位数。具体来讲，如果客户端接收到的数据完整性检测数值为 $(f_0^{g_0}, f_1^{g_1}, f_2^{g_2}, \cdots, f_{k-1}^{g_{k-1}}, m_0^{g'_0}, m_1^{g'_1}, \cdots, m_{m-1}^{g'_{m-1}})$，其中 g_i 表示存储节点具体的等级信息，当 $p*(f_0^{g_0}, f_1^{g_1}, f_2^{g_2}, \cdots, f_{k-1}^{g_{k-1}})^T \neq (m_0^{g'_0}, m_1^{g'_1}, \cdots, m_{m-1}^{g'_{m-1}})^T$ 成立时，则表示该节点数据不完整，这时客户端可以从检测值 $(f_0^{g_0}, f_1^{g_1}, f_2^{g_2}, \cdots, f_{k-1}^{g_{k-1}}, m_0^{g'_0}, m_1^{g'_1}, \cdots, m_{m-1}^{g'_{m-1}})$ 中选取 g_i 等级信息中的 k 个检测值，如果这时存在 m 个错误检测值，则利用 k 个检测值恢复错误的 m 个检测值，然后根据需要选取 m 个检测值进行置换。需要说明的是，如果存在 i 个检测值一致时，则剩余 $m-i$ 个检测值对应的数据一定是不完整的。上述的这种数据恢复方式不仅提高了错误数据的检索效率，而且还提高了错误检索的精度。

当然还可以利用改进的IDBRS检测算法来完善基于云数据的完整性检测方式，具体实现步骤如下：

（1）首先由客户端随机选取一个关键值，表示为 $key_{a,b}$；

（2）向服务器发送 n 个存储节点；

（3）服务器会利用字节生成函数 $\theta_{key}(.)$ 将 $key_{a,b}$ 转化成为长度为 $b-a$ 的字节流；

（4）客户端接收服务器返回的 n 个值 $(f_0^{g_0}, f_1^{g_1}, f_2^{g_2}, \cdots, f_{k-1}^{g_{k-1}}, m_0^{g'_0}, m_1^{g'_1}, \cdots, m_{m-1}^{g'_{m-1}})$；

（5）客户端接收服务器返回的检测值 $(f'_{i0}, f'_{i1}, \cdots, f'_{ik'-1}, e'_{i0}, \cdots, e'_{im'-1})$。

3 实验结果及分析

3.1 编码效率

为了更好地比对实验结果，分别针对200 M的文件和500 M的文件进行数据编码效果测试，其具体的编码速度如图2所示。从图2中不难看出，编码率直接影响着编码速度，当编码率为 $R=k/m$，而 k 值一定或者 $m-k$ 值一定时，文件的编码率越高，其对应的编码速度也就越快。

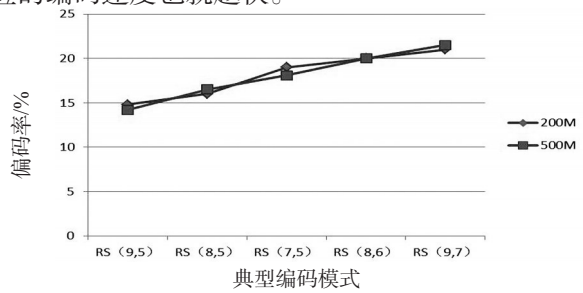

图2 编码速度示意图

3.2 数据完整性检测

利用固定的编码率 $RS(9,5)$ 针对500 M文件进行数据完整性测试，如果经过编码的文件存入到节点中后，系统会随机选择多个存储节点，并进行修改。通过多次修改和测试，客户端都能够检测到出错节点的准确位置。图3准确记录了出错节点数以及检测所需时间，从而可以看出检测所需时间与出错节点数之间并无明显的关系。

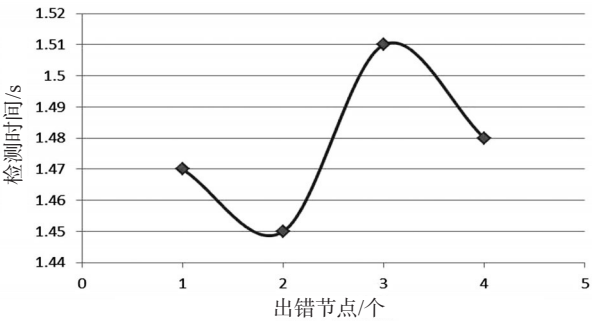

图3 出差节点数与检测时间的关系

又针对不同存储大小的文件进行试验，图4准确记录了文件大小以及检测所需时间，

（下转第50页）

上式中，*Dr*、*BDCI*、*PCI*、*SBCI*分别表示全桥、上部结构、下部结构和桥面系技术状况评分值。

5 结论与建议

根据桥梁可视部分构件定期检查结果，评定该桥目前桥梁技术状况属于四类桥梁(处于差的状态)。结合规范相关要求，提出以下养护建议[5]：(1)对通行车辆进行限速限载交通管制；(2)封闭拱波及边跨主拱圈裂缝；(3)加固各跨主拱圈及主跨腹拱圈；(4)修补或重新铺装桥面铺装层；(5)重新安装栏杆，疏通泄水孔；(6)加强桥梁养护管理。如何结合实际检测的数据准确地建模计算双曲拱桥的极限承载力，是课题进一步深入研究的内容之一。

参考文献：

[1] 王灿，朱新实.双曲拱桥病害原因分析及处治对策的研究[J].公路，2002,19(11)：74-76.

[2] 陈翱.钢筋混凝土双曲拱桥检测评定及承载力分析[J].工程科技，2005，29(4)：389-393.

[3] 汤建忠.既有双曲拱桥检测、评定及加固技术改进研究[D].南京：南京工业大学，2015.

[4] 中华人民共和国行业标准.(JTG H11－2004)公路桥涵养护技术规范[S].北京：人民交通出版社,2004.

[5] 郭丰哲，钱永久，王振领，等.钢筋混凝土双曲拱桥检测及承载能力评估[J].四川建筑科学研究,2005,31(3)：66-69.

(上接第36页)

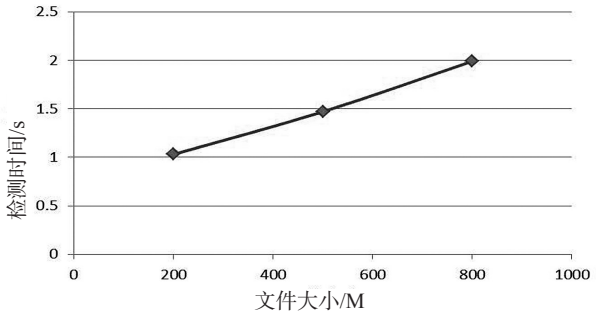

图4 文件大小与检测时间的关系

从图4可以看出：文件大小确实影响着检测所需时间，即当文件变大时，其检测所需时间也随之增加。

4 结语

本文主要研究了当云数据出现硬件障碍时，如何恢复出错数据，而且还专门提出了基于存储节点的动态分级管理模式，从而实现有效管理存储节点。该模式不仅提高了存储数据的可靠性和安全性，而且显著提高了存储数据的完整性测试效果。

参考文献：

[1] 马晓亭,陈臣.基于云服务模式分析的数字图书馆云服务平台设计与实现[J].图书馆理论与实践, 2013 (6): 83-86.

[2] 苑野,伞晓娇.云计算与网格计算比较研究[J].哈尔滨商业大学学报:(自然科学版), 2012 (2): 222-227.

[3] 沈晓娟.多效性RSA 数字签名技术及其应用[J].科技信息,2008,68 (31):52-52.

[4] NICOLAE B, CAPPELLO F,BLOB CR. Virtual Disk Based Checkpoint－restart for HPC Applications on Iaa S Clouds[J]. Journal of Parallel and Distributed Computing, 2013 (5): 698-711.

[5] 赵波，严飞，张立强，等.可信云计算环境的构建[J]. 中国计算机学会通讯，2012 (8): 28-34.

[6] 谢琪,吴吉义,土贵林,等.云计算中基于可转换代理签密的可证安全的认证协议[J].中国科学:信息科学，2012(42): 303-313.

[7] 唐春明.防泄露的秘密共享方案及其在群身份认证协议中的应用[J].中国科学:信息科学，2012(42):634-647.

(上接第39页)

参考文献：

[1] 杭州华三通信技术有限公司. 新一代网络建设理论与实践[M]. 北京：电子工业出版社, 2011.

[2] 杨宇.网络虚拟化资源管理及虚拟网络应用研究[M].北京：北京邮电大学，2013.

[3] 王剑春. 快速建立稳定可靠的数据中心[J]. 中国计算机报, 2008 (3):10-12.

[4] 吴晨, 朱志祥, 胡清俊. 一种云数据中心虚拟交换的解决方案[J]. 西安邮电学院学报, 2011 (5):54-58.

[5] 薛碧玉. 虚拟化在电信新一代数据中心建设中的应用研究[M]. 南京: 南京邮电大学, 2009.

[6] 王卫星, 李斌. 高校云计算数据中心虚拟化接入层技术的选择[J]. 开封教育学院学报, 2013, 33(2):53-56.

[7] 汪萌, 梁雨锋. 基于虚拟化环境下的网络安全监控技术应用[J]. 计算机技术与自动化, 2013,32(1):137-140.