基于异或的隐私保护码优化研究

金星彤 李 鹏 王 刚 刘晓光 李忠伟
(南开大学计算机与控制工程学院 天津 300350)

摘 要 随着存储系统的发展,为了满足当前高速增长的信息数据量对存储的需求,云存储行业迅速兴起。然而,单云存储面临着数据保密性、安全性、可用性和厂商锁定的风险。基于异或的非系统纠删码-隐私保护码(PPC)可以用来构造具有隐私保护能力的多云存储系统,从而在很大程度上解决上述问题。主要针对PPC编码算法进行优化,以提高编码运行性能。通过设计搜索PPC的最优调度来减少编码过程中的异或次数。由于PPC的编码/解码计算可以表示为生成矩阵(0/1矩阵)和数据向量的乘法,直观上计算量与生成矩阵中1的数目成正比,因此通过对计算次序的优化调度可以获得更好的性能。首先,设计并实现搜索PPC最优调度次序的算法,利用此算法寻找计算性能最优者,可优化具有隐私保护能力的多云存储系统的性能。其次,在基于最优调度次序的编码算法的基础上,利用AVX2技术的SIMD并行优化来提高编码过程中的每次异或的性能。实验表明,基于最优调度的编码性能提高了34.8%,进行SIMD并行优化后进一步提高了107.1%。

关键词 隐私保护码,编码优化,SIMD优化,数据安全,纠删码,非系统码
中图法分类号 TP309.3 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.06.006

Optimizing Small XOR-based Non-systematic Erasure Codes

JIN Xing-tong LI Peng WANG Gang LIU Xiao-guang LI Zhong-wei
(College of Computer and Control Engineering, Nankai University, Tianjin 300350, China)

Abstract With the development of storage systems, the rapid rise of cloud storage has met the storing problem of highly increased data quantity. However, now the single cloud storage is facing the risks of data confidentiality, security, availability and vendor lock-in. To solve the above problems, we can construct the multi-cloud storage system with the private protecting ability through using privacy protecting code(PPC), which is an erasure code based on the XOR operation. This paper mainly analyzed the optimization coding algorithm on the PPC to improve the performance of encoding operations. First, we designed an algorithm to search the optimal XOR scheduling sequence to reduce the XOR times in encoding. Because the encoding/decoding calculation of the PPC can be expressed as the multiplication of generator matrix (0/1 matrix) and data vectors, it can be observed visually that the computation is proportional to the number of 1 of the generator matrix. And based on the optimal scheduling order, we can get better performance. The searching result can be used to construct the multi-cloud storage system. Second, we can use the AVX2 technique of SIMD parallel optimization to improve the encoding performance based on the optimization schedule. Through the experiment, the performance of encoding based on the optimization schedule improves by 34.8%, and after being further optimized by SIMD technique, the performance improves 107.1%.

Keywords PPC, Encoding optimization, SIMD optimization, Data security, Erasure codes, Non-systematic codes

近年来,随着计算机技术和互联网行业的发展,数据的增长速度空前,但由于本地存储空间有限,大量数据及其备份所需存储空间的增加为计算机存储领域带了巨大的挑战[1]。为了满足个人及企业在空间上对数据存储的需求,云存储行业逐渐开始兴起,许多互联网公司都提供了云存储服务,例如Google Drive[2], Dropbox[3], Amazon S3[4], Microsoft One-Drive[5]等。

但是,随着云储存行业的兴起,在云端存储数据所存在的安全隐患也开始逐渐暴露,如黑客攻击存储服务器,客户在传输数据的过程中造成数据丢失等。近年来,云存储服务商泄露用户数据事件不断涌出[6],使得云存储服务厂商不得不对存储系统提出新的要求,即既要满足存储容量上的需求,又要高度关注数据安全、有效等其他不容忽视的问题。其次,用户将数据存储在单一云服务厂商所带来的依赖问题也逐渐显

到稿日期:2016-11-11 返修日期:2016-12-30

金星彤 硕士生,主要研究方向为分布式存储下的编码优化,E-mail:jinxingtong@nbjl.nankai.edu.cn;**李 鹏** 博士生,主要研究方向为分布式存储系统、编码;**王 刚** 教授,博士生导师,主要研究方向为搜索引擎技术、云存储、GPU计算;**刘晓光** 教授,博士生导师,主要研究方向为搜索和推荐技术、云存储、GPU计算;**李忠伟** 副教授,硕士生导师,主要研究方向为机器学习、网络存储。

露，如用户不得不接受厂商突然关闭存储服务或提高服务费用等情况。

为了应用单云存储所带来的数据完整性、数据保密性、服务可用性和厂商锁定的威胁，可以构建多云存储系统，将用户上传的原始数据进行编码，并将编码后的数据散布到不同的网盘上，如图 1 所示。DepSky[7]，RACS[8] 和 HAIL[9] 等都是基于此思想的多云存储系统。

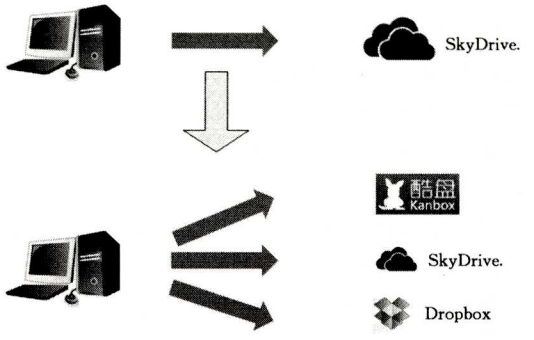

图 1　单云存储安全风险的解决方案：构建多云存储

当前的多云存储系统在编码数据时大多采用纠删码，但是传统的纠删码基本都是系统码，其原始数据也会同时保存在云端，因此安全性较低。另一些非系统码（如 RS 编码）在编码过程中需要做复杂度较高的基于伽罗华域的运算[20]，对运算平台的硬件要求很高。因此，为了提高安全性并降低编码的复杂度，可以采用基于异或操作的非系统码构建跨越多服务提供商的云存储系统。隐私保护码（Privacy Protecting Codes，PPC）即为一种基于异或的非系统纠删码，可以用来构建多云存储系统。

然而，各个云存储服务对主流移动端操作系统的支持（如微软的 SkyDrive 支持 Window phone 系统，Google 的 Google Drive 支持 Android，Dropbox 支持 Linux 和黑莓智能机系统等），极大地满足了个人用户的日常需求，越来越多的云端存储、读取数据的行为开始在移动端设备进行，如手机或平板上的照片、文档上传和下载等操作。

相较于电脑端对编码数据处理的速度，移动平台的处理速度明显更缓慢。为了保证移动端的多云存储系统更高效地运行，需要提供更高性能的编码算法。其次，隐私保护码虽然可以解决数据安全等问题，但是相较于 RAID5 与 RAID6 等纠删码，其计算复杂度更高[10]。因此，如何提高编码算法的性能，使其能够在上述多云存储系统上更高效地运用，对云存储系统的发展具有重要意义。

文献[10]采用了一种最优调度方法来优化 PPC 码的编码计算性能，通过搜索编码过程的最优调度及设计，基于最优调度的编码算法来达到优化编码算法性能的目的。本文将详细介绍这种最优调度方法的设计与实现，并采用 AVX2 技术的 SIMD 并行优化进一步提升性能。

本文第 1 节讨论了 PPC 码的相关研究、设计编码的最优调度所遵循的定理和准则以及设计编码的最优调度所采用的方法；第 2 节描述了如何设计搜索 PPC 编码的最优调度的算法以及基于最优调度的编码算法，并在最优调度的基础上进行 SIMD 优化，与平凡编码算法进行了性能比较；第 3 节给出了实验结果并对它进行了分析；最后总结全文，并指出进一步的工作方向。

1　纠删码技术的相关研究

1.1　隐私保护编码

因为传统的非系统纠删码无法隐藏原始数据，所以可以利用非系统纠删码来解决用户数据的隐私问题。这种基于异或的非系统码的存储数据保护编码[10] 称为隐私保护编码（Privacy Protecting Code，PPC）。

首先，定义隐私度 t：如果一个 PPC 码编码生成的数据能抵抗任何 t 个数据块的泄露而不会泄露任何一块原始数据信息，则称这个 PPC 码的隐私度为 t。由于 PPC 码是非系统码，经编码产生的任何一块数据都不会直接包含原始数据，因此 PPC 隐私度 t 至少为 1。

用 PPC(k,n,t)来描述一个隐私度为 t 的 PPC 码，表示由 k 个磁盘上的原始数据编码生成 n 个磁盘的冗余信息，其中任意 t 个冗余信息都无法恢复出任何原始信息，但是存在 $k+1$ 个冗余信息，可能恢复出原始数据。如图 2 中的编码方案 PPC(6,7,2)，原始数据'e'可以由冗余信息 p_0，p_2 和 p_6 计算得来。

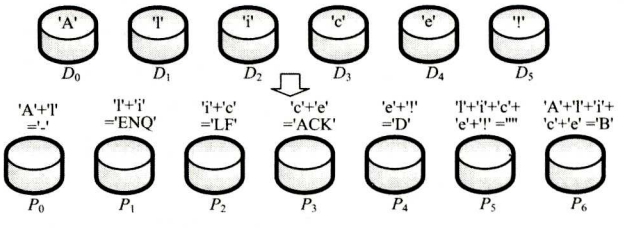

图 2　一种隐私保护编码的编码方案 PPC(6,7,2)

事实上，一个 PPC(k,n,t)码在某些情况下可以抵抗大于 t 块数据的泄露。例如，即使图 2 所示的 PPC(6,7,2)码泄露了前 5 个数据块，但依旧恢复不出任何与原始内容相关的信息。因此，一个 PPC 码的最优隐私度可能大于 t。

由于 PPC 码为基于异或的编码，原始数据编码生成冗余数据的过程即为原始数据块之间的异或过程。定义大小为 $n\times k$ 的生成矩阵（只包含 0/1），编码过程可表示为原始数据与生成矩阵的乘法运算。其中，生成矩阵每一行的 0/1 决定冗余数是否由相应的原始数据异或而来。

由于 PPC 码的参数 k 可能会很大，因此要求设计扩展性良好且不会丢失隐私特性的 PPC 码，ShrPPC 即为一种符合此要求的可扩展码。图 3 分别示出了 k 是奇数和偶数两种情况下的 ShrPPC(k,$k+1$,t)的生成矩阵。其中，在构造 ShrPPC 矩阵时，当 k 为奇数时，第 k 行为 01…1；当 k 为偶数时，第 k 行为 001…1；第 $k+1$ 行行向量即为前 k 行行向量的异或结果。由 ShrPPC 的构造过程可以证明其隐私度 $t=\lfloor k/2-1 \rfloor$，且在最优情况下其隐私度可以达到 $k-1$。

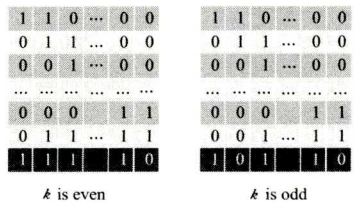

图 3　k 是奇数或偶数时 ShrPPC(k,$k+1$,t)的生成矩阵

PPC 码可应用于多云存储系统，存储数据时，先将数据分块，利用 PPC 矩阵进行编码生成冗余信息后上传至各个云端。使用数据时，将各分块冗余信息从云端下载，对其解码后得到原始数据。PPC 的隐私度至少为 1，因此任何一块冗余信息泄露都不会导致用户数据泄露，保证了数据的安全性；而且由于 PPC 具有单容错的特性，因此当云端某个冗余数据被篡改或在传输过程中丢失时，可以通过解码其他冗余数据块得到丢失数据。

虽然 ShrPPC 能达到较好的隐私保护能力，但是它的最小行重为 2，所以选择明文攻击等安全问题会比较突出。为了有效缓解这些问题，设计算法搜索了 PPC 码中最小行重为 3 的生成矩阵。本文即采用搜索所得的隐私度为 2、最小行重为 3 的 PPC 矩阵，设计算法搜索其最优编码调度，并且基于最优调度设计编码方案，进行 SIMD 优化，同时与平凡算法进行性能比较。

1.2 基于异或的纠删码优化思想

由于 PPC 码是基于异或的编码方案，因此其编码效率取决于编码过程中所需要的异或次数。所以，如果要对编码算法的性能进行提升，只需要在编码过程中减少异或运算就可以达到很好的效果。基于此，几乎所有的纠删码（从 RAID-6 码，如 EVENODD[11]，RDP[12]，X 码[13]和 P 码[14]等，到大型系统中的 STAR 码[15]，T 码[16]和 WEAVER 码[17]等）都在核心部分减少了异或次数。

给出一个编解码的例子 $AX=B$。为规范起见，设生成矩阵 A 的大小为 $r\times c$。X 的每行称为数据元素（记为 x_i，对应一个在第 i 位上是 1、其余位是 0 的 r 位向量）。将两个元素相加时，实际上只是对它们进行了一次异或操作，如 $x_0+x_1=1100\cdots00$。B 的每一行称为目标元素（记为 b_i）。编码过程即为用 c 位数据元素 X 计算生成 r 位目标元素 B。用一系列异或操作来执行此计算，这一系列异或操作就是一个调度，我们希望它所包含的异或次数能达到最少。

由于计算顺序并不重要，把 B 作为目标集 $\{b_0,b_1,\cdots,b_{r-1}\}$。由于 B 最终由 A 限定，同样称 A 为目标集。现给出编码调度的合法性准则[18]：一个调度 S，是一个元素的有序集合 $\{s_0,s_1,\cdots,s_{r-1}\}$。区别于目标集，调度 S 中的元素次序代表了编码过程中相对应数据块的异或次序，它必须遵循以下两个规则。

1）前 c 个元素是 X 中的元素：

$s_j=x_j,j=0,1,2,\cdots,c-1$

即前 c 个元素组成 $c\times c$ 的单位矩阵；

2）任何非 X 中的元素 s_k 必须遵循：

$s_k=s_i+s_j,i<j<k$

即 s_k 必须由它之前的元素经过异或运算得到。

符合上述规则的调度称为一个合法调度。最优调度 S_A^{opt} 为所有合法调度中包含所有目标元素，并且 $|S_A^{opt}|$ 最小（即其包含的元素最少）的调度。

一个生成矩阵可以有很多个合法调度，但是本文只需要找到它所有合法调度中的最优调度，即异或次数最少的调度。现给出最优调度遵循定理[18]：设 $s_i,s_j,s_k\in S$，其中 $i<j<k-1$，$s_k=s_i+s_j$，那么存在一个合法调度 S'，满足：

$$S'=\left\{s_x'\mid s_x'=\begin{cases}s_x, & x\leqslant j\\ s_k, & x=j+1\\ s_{x-1}, & j+1<x\leqslant k\\ s_x, & x>k\end{cases}\right\}$$

假设建立了一个调度，它包含了 s_i 和 s_j，并且这个调度将来也会包含 s_k，那么可以在 s_j 加入到调度后立即将 s_k 也加入到当前调度，并且这种先后次序的交换不影响调度的合法性。

证明：由上述描述可知，唯一可能影响到别的元素的合法性的元素为 s_k，即唯一可能违反“当前加入调度的元素必定为之前加入的某两个元素的异或结果”的元素为 s_k。由于 s_k 是 s_i 和 s_j 的异或结果，并且 $i<j$，因此 s_k 也是一个合法元素。故，S' 是一个合法调度。

本文应用此定理来设计搜索 PPC 码的最优调度算法。

2 PPC 编码优化策略

可从两个方面对基于异或的 PPC 码进行优化：1）可利用搜索编码的最优调度来减少编码过程中的异或次数实现性能提升；2）可利用 AVX2 技术的 SIMD 并行优化来提高编码过程中每次异或的性能。

2.1 基于最优调度的优化

2.1.1 搜索纠删码的最优调度

本文主要研究利用枚举法搜索小矩阵（$k<8$）的 PPC 码的最优调度。当矩阵较大（$k\geqslant 8$）时，可采用启发式优化算法，如 CSHR 和 Uber-CSHR[18]等。

图 4 示出了 PPC 矩阵的异或过程。A 是大小为 8×7 的 PPC 生成矩阵；X 为原始数据，分布在 7 个磁盘上；B 为原始数据经编码后得到的冗余数据，分布在 8 个磁盘上。异或结果表示为几个数据块的和，如 $b_0=x_4+x_5+x_6$，代表数据块 x_4，x_5 和 x_6 进行异或操作得到冗余数据块 b_0。

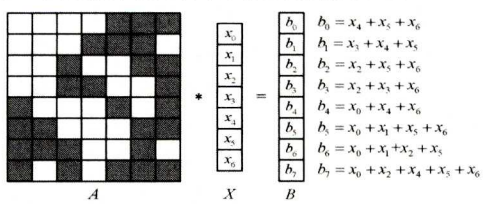

图 4 描述编码操作的 PPC 生成矩阵例子

执行编码的一种方法是计算生成 $b_i(i=0,1,2,3)$ 的每一次异或，即直接用第 i 行计算式右边的式子计算 b_i。此种方法在生成 b_0 时需要进行 x_4+x_5 以及 $(x_4+x_5)+x_6$ 两次异或操作，因此对于图 4 的例子，生成所有冗余信息需要进行 20 次异或运算。

也可以用另一种节省异或次数的方法计算冗余信息 B。

首先，对于 b_0，b_2，b_5 和 b_7，每一个元素的生成都包含中间异或结果 x_5+x_6。若先计算出中间结果，再利用中间结果计算 b_0，b_2，b_5 和 b_7，则比直接计算可以节省 3 次异或运算。其次，有些生成元素也可以利用其他已计算好的生成元素或者中间和来计算。例如图 4 中，b_1 可由 b_3 和 s_{10} 两个中间结果

异或得到。按这种思路，生成图 4 中所有冗余信息只需要 14 次异或运算，比直接计算所需异或次数少。

遵循编码调度的合法性准则，图 4 中的一个合法调度为：

前 7 个元素 s_0-s_6 为：{1000000，0100000，…，0000001}；

剩余元素 s_7-s_{20} 为：{0010001，0011001，0010011，0010111，1010111，000 0111，0001110，0000110，1000110，1010110，1000101，1100101，1110010，1100011}。

调度次序可以表示为图 5 所示的矩阵。

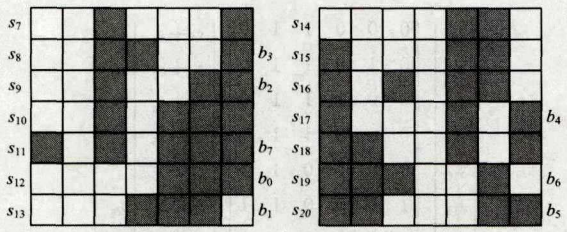

图 5　图 4 的一种优化调度

以这种方式定义的调度并不强调如何确切地执行异或生成每个元素，只需保证每个调度 S 包含的所有元素均可以由 $|S|-|X|$ 次异或得来即可。

例如在图 5 中，因为元素中含有 $b_3=x_3+s_7$，那么 b_3 可以在 s_7 加入调度后也随之加入调度中，但不影响调度的合法性。

假设给定一个 $(m+1)\times m$ 位的 PPC 矩阵，$m+1$ 行 m 位元素组成目标集合 A，它包含所有的目标元素，可以由宽度优先算法(Breadth-First Search，BFS)生成目标集合，从而得到一个最优调度 S_A^{opt}。定义一个无权图 G，因此每个可能的调度 S 均可用图 G 中的结点 N 表示。假设 $S'=S+\{e\}$，并且 S' 和 S 均为合法调度，那么在 G 中存在一条从结点 N_S 到结点 $N_{S'}$ 的边。

设 S_X 是只包含数据元素的调度，那么一个最优调度可以被图中任何一个结点表示，只要这个结点满足：包含所有目标元素，并且到结点 N_{S_X} 的路径是最短的。这是一个求解无权最短路径的问题。

设最优调度结点 $N_{S_A^{opt}}$ 距离 NSX 为 d，BFS 的性能与那些距离 NSX 比 d 小的结点数目成正比，而这个数目与 c 和 $|S_A^{opt}|$ 呈指数关系。由于运算时间的限制，仅将此搜索算法应用在小矩阵上。

生成最优调度遵循的定理允许我们通过合并结点来精简搜索。如果存在从 N_S 到 $N_{S'}$ 的边，且 $S'=S+\{b_i\}$，则由定理“任何从 S 开始的最优调度和从 S' 开始的最优调度是等价的”可以合并结点 S 和 S'，以显著减少搜索。

图 5 所示为利用 BFS 得出的优化调度，它需要 14 次异或运算。然而此调度的路径仅有 4 条边。调度路径如下：

$$
\begin{aligned}
&(S_X+\{s_7,b_3,b_2\})\\
&\quad\rightarrow(S_X+\{s_7,b_3,b_2,s_{10},b_7,b_0,b_1\})\\
&\quad\rightarrow(S_X+\{s_7,b_3,b_2,s_{10},b_7,b_0,b_1,s_{14},s_{15},s_{16},b_4\})\\
&\quad\rightarrow(S_X+\{s_7,b_3,b_2,s_{10},b_7,b_0,b_1,s_{14},s_{15},s_{16},b_4,s_{18},b_6,\\
&\quad\ b_5\})\\
&\quad=S^{opt}
\end{aligned}
$$

利用上述算法，可以得到一个搜索 PPC 生成矩阵的最优调度算法(即算法 1)。

算法 1　搜索 PPC 生成矩阵的最优调度

```
1. Target[1…k+1]←target elements
2. Data[1…k] ←data elements
3. combines[combines_max] ←0
4. Node[node_max] ←root
5. Node. Data[ ]←Data[1…k]
6. Node. Data←Target[1…k+1]
7. generateNodeCombines(Data)//See algorithm 2
8. L1:
9. if (Target. size( ) == 0) then
10.    return node[node_id]
11. else
12.    for j=0 to n
13.       if (Target[i] == combines[j]) then
14.          Node. child[id]=Node
15.          Node. child[id]. Data[k+1]=Target[i]
16.          generateNodeCombines(Data)
17.          delete Target[i] from
               Node. child[id]. Target[1…k+1]
18.       else
19.          generateNodeCombines(combines)
```

搜索算法描述如下：

(1)生成 S_X 结点。

S_X={1000000，0100000，…，0000001}

(2)生成该结点所有元素的组合集(combines)，如 combines={110000，101000，100100，100010，100001，…}，然后测试组合集中有无目标元素。

1)若存在目标元素，则删除除了目标元素以外的组合元素，然后把目标元素分别加入到当前结点的孩子结点中，生成下一层的若干结点。

2)若不存在目标元素，则由所有组合分别继续生成组合，测试是否有目标元素。

3)直到有任意一个结点的剩余未生成目标元素数为 0，这个结点就是最佳调度。

其中，生成每个节点内元素间异或结果的过程如算法 2 所示。

算法 2　生成节点内元素异或结果集合的算法

```
1. combines[MAX] ←0
2. for i=0 to Node. DataEnd()
3.    for j=1 to Node. DataEnd()
4.       new_value=Data[i]^Data[j]
5.       if (new_value not in Data[1,…,t]) then
6.          combines[m]=new_value
```

2.1.2　基于最优调度的编码方案

编码的过程就是生成矩阵与原始数据做乘法运算的过程。在基于异或的编码过程中，矩阵中只有 0/1。数据矩阵为一个 $k\times k$ 的单位矩阵，它的第 i 行表示原始数据的第 i 块。生成矩阵中每一行的第 $i,j,k,\cdots$ 个位置上的 1 表示原始数据

中相应的第 $i,j,k,\cdots$ 块要进行异或操作来生成冗余数据；生成矩阵中的第 i 行表示生成第 i 块冗余数据。冗余数据矩阵的第 i 行表示冗余数据的第 i 块数据即可。

对于编码的平凡算法，编码过程即为由生成矩阵按行序对原始数据进行异或操作。

对于基于最优调度的编码，只需要将调度顺序存储，在编码时按照调度顺序处理原始数据。

2.2 基于 SIMD 技术的并行优化

SIMD 指单指令多数据技术，它已经成为 Intel 处理器的重要性能扩展。SIMD 指令集能够对一组数据执行向量形式的运算，每组数据被划分为几个子字，并对所有的子字都执行相同的操作。

目前 Intel 处理器支持的 SIMD 技术包括 MMX，SSE，AVX[19]。其中，在 Intel Sandy Bridge 微架构中，Intel 引入了 256 位 SIMD 扩展 AVX，这套指令集在兼容原 MMX，SSE，SSE2 对 128 位整点 SIMD 支持的基础上，把支持的总向量数据宽度扩展成了 256 位，并且新增了若干条 256 位浮点 SIMD 指令。AVX2 指令集在 AVX 的基础上做了部分扩展。

由于基于异或的 PPC 编码算法中含有大量异或运算，为了达到更好的性能优化，本文采用 AVX2 技术在基于最优调度次序的编码方案上进行了进一步的并行优化，即对编码过程中的每一步异或操作进行了并行化操作。算法 3 描述了并行优化过程。

算法 3 利用 AVX2 并行优化异或运算的算法

```
INPUT:dataBlock1,dataBlock2,xorLength
OUTPUT:codeBlock
1. avxBlock←256
2. INT←4 * sizeof(int)
3. off←avxBlock / INT
4. count←xorLength / off
5. __m256i t1,t2,t3
6. for i←0 to count
7.    t1=_mm256_loadu_si256((__m256i * )(dataBlock1+i * off));
8.    t2=_mm256_loadu_si256((__m256i * )(dataBlock2+i * off));
9.    t3=_mm256_xor_si256(t1,t2);
10.   _mm256_storeu_si256(((__m256i * )(codeBlock+i * off)),t3);
11.   if (lens % off != 0) then
12.      for j←off * count to xorLength
13.         codeBlock[j]=dataBlock1[j] ^ dataBlock2[j];
14. return codeBlock
```

3 实验结果与分析

实验所用系统环境如表 1 所列。

表 1 实验所用系统环境

	机器配置
CPU	Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz
内存	16.0GB
操作系统	Windows 10

3.1 最优调度搜索结果

本文所用测试数据为图 6(a)所示的生成矩阵。利用第 2 节设计的算法搜索所得的最优调度如图 6(b)所示，其中共有 10 次异或操作：$S_A^{opt}=\{s_6,s_7,s_8,\cdots,s_{16}\}$，$s_0-s_5$ 为单位矩阵。

$$\begin{bmatrix}0&0&0&1&1&1\\0&0&1&1&1&0\\0&1&0&0&1&1\\0&1&0&1&0&1\\0&1&0&1&1&0\\1&0&0&0&1&1\\1&1&1&0&1&0\end{bmatrix}$$

(a)测试生成矩阵

$$\begin{array}{c|c|c|c}
s_6 & 0\ 0\ 0\ 1\ 1\ 0 & s_3+s_4 & \\
s_7 & 0\ 1\ 0\ 1\ 1\ 0 & s_1+s_6 & b_4 \\
s_8 & 0\ 0\ 1\ 1\ 1\ 0 & s_2+s_6 & b_1 \\
s_9 & 0\ 0\ 0\ 1\ 1\ 1 & s_6+s_5 & b_0 \\
s_{10} & 0\ 0\ 0\ 0\ 1\ 1 & s_4+s_5 & \\
s_{11} & 1\ 0\ 0\ 0\ 1\ 1 & s_0+s_{10} & b_5 \\
s_{12} & 0\ 1\ 0\ 0\ 1\ 1 & s_1+s_{10} & b_2 \\
s_{13} & 0\ 1\ 0\ 1\ 0\ 1 & s_6+s_{12} & b_3 \\
s_{14} & 1\ 1\ 0\ 1\ 0\ 1 & s_0+s_{13} & \\
s_{15} & 1\ 1\ 1\ 1\ 0\ 1 & s_2+s_{14} & \\
s_{16} & 1\ 1\ 1\ 0\ 1\ 0 & s_{15}+s_6 & b_6
\end{array}$$

(b)所得的最优调度次序

图 6

为验证最优搜索算法，利用搜索编码优化调度算法又测试了一组隐私度为 3、大小为 7×6 的 PPC 矩阵。图 7 所示为生成矩阵优化前后编码所需异或次数的对比，其中，优化前需要 15 次异或的矩阵有 72 个；经过优化后，有 48 个矩阵仅需要 11 次异或，24 个矩阵需要 12 次异或，以此类推。

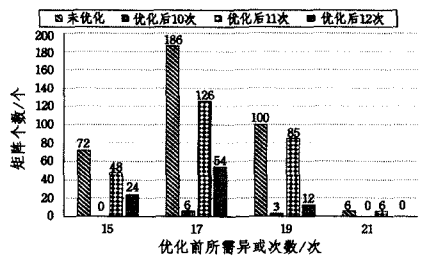

图 7 生成矩阵优化前后编码所需异或次数的对比

由第 2 节的最优调度的算法设计可得，最优编码调度增加了中间结果的存储，减少了重复的异或次数。由实验结果可知，搜索得到的最优调度的异或次数比优化前平均减少了 36.9%。

由上述实验结果可以推出，相较于平凡算法的编码速度，基于最优调度次序的编码速度应该会有很大程度的提高；并且，对于隐私度相同的 PPC 矩阵，随着其生成矩阵异或次数的增加，其优化后调度的异或次数的减少率有上升趋势，这一情况也可由理论推出，当矩阵中 1 越多时，由同一个中间异或数据得出的最终冗余数据会更多。

3.2 最优调度编码的性能分析

通过固定 PPC 矩阵，改变编码数据块的大小来观测优化性能。实验中，数据块的大小分别为 32MB，64MB，128MB 及 256MB，生成矩阵为一个隐私度为 3 的 8×7 PPC 生成矩阵。图 8 示出了分别利用平凡算法和基于最优调度编码这几组数

据时的吞吐率比较。

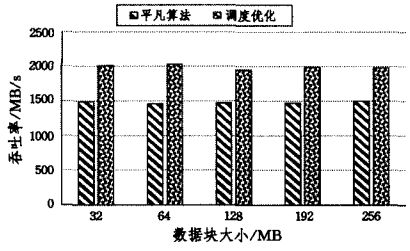

图 8　生成矩阵优化前后吞吐率的对比

由生成矩阵优化前后的异或次数对比可以推出，基于最优调度的编码比平凡编码算法的性能提高了 36.9%左右。由实验结果可以得出，基于最优调度的编码比平凡算法性能平均提高了 34.8%，性能提高符合预期。

3.3　SIMD 并行优化性能分析

为了进一步提高编码性能，本文在最优调度的基础上又进行了 SIMD 优化：采用 AVX2 对异或过程进行了并行化。实验中采用大小固定为 8×7、隐私度为 3 的 PPC 矩阵，测试数据块大小分别为 32MB，64MB，128MB 及 256MB 时的吞吐率变化。

由图 9 可以得到，SIMD 优化可以提高 107.1%的性能。

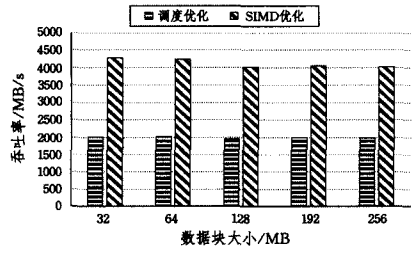

图 9　SIMD 优化前后吞吐率的对比

考虑到不同生成矩阵之间的差异，对隐私度均为 3、大小为 8×7 的 5 组不同的 PPC 生成矩阵做了性能测试，同时比较了平凡算法、基于最优调度算法和经 SIMD 优化的基于最优调度算法的吞吐率，数据块大小固定为 64MB，如图 10 所示，A－E 为 5 个生成矩阵。

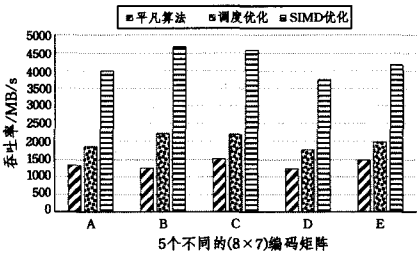

图 10　不同生成矩阵优化前后吞吐率的对比

由实验结果可以得出，基于最优调度的编码比平凡算法在性能上平均提高了 47.6%，比平均调度次序优化的 36.9%略高，这是因为选取的矩阵中有优化前后异或次数接近 2∶1 的生成矩阵。由实验结果可以得出，优化后异或次数减少得越多，编码性能提高幅度就越大。

经 SIMD 优化的编码性能提高为 111.3%，对于不同的生成矩阵，并行优化所得到的性能提高基本恒定。

结束语　本文从减少编码过程中的异或次数以提高数据编码性能的角度出发，研究了基于异或的非系统小纠删码优化问题。

本文首先研究了数据经小纠删码矩阵编码生成冗余数据过程中的异或次数问题，通过保存异或中间结果并利用中间结果或已生成冗余数据来生成剩余冗余数据的方法，生成生成矩阵的最优调度，在最优调度中减少了编码过程中的重复异或次数；并且由实验结果可以得出，优化后其异或次数比优化前的平均减少了 36.9%。

其次，对平凡算法和基于最优调度次序的编码之间的性能进行了比较，并在优化编码调度的基础上进行了 SIMD 优化。通过实验可以得出，调度优化后编码性能提高了34.8%，进行 SIMD 优化后进一步提高了 107.1%。

综上所述，对基于异或的非系统小纠删码的异或次序进行优化可以在很大程度上提高数据编码性能，基于 SIMD 并行优化每次异或操作后编码性能有了明显提升。

参 考 文 献

[1] ZHOU A Y. Data intensive computing - challenges of data management techniques[J]. Communications of CCF, 2009, 5(7): 50-53.

[2] Google Drive [EB/OL]. [2016-07-09]. https://en.wikipedia.org/wiki/Google_Drive.

[3] Dropbox (service) [EB/OL]. [2016-07-09]. https://en.wikipedia.org/wiki/Dropbox_(service).

[4] Amazon Simple Storage Service(S3) [EB/OL]. [2016-07-09]. https://aws.amazon.com/ s3.

[5] Microsoft Azure [EB/OL]. [2016-07-09]. https://en.wikipedia.org/wiki/Microsoft_Azure.

[6] A survey called "Data Breach Investigations Report" in 2015 by Verizon[EB/OL]. [2016-07-09]. http://www.freebuf.com/news/64183.html.

[7] BESSANI A, CORREIA M, QUARESMA B, et al. DepSky: dependable and secure storage in a cloud-of-clouds [J]. ACM Transactions on Storage (TOS), 2013, 9(4): 12.

[8] ABU-LIBDEH H, PRINCEHOUSE L, WEATHERSPOON H. RACS: a case for cloud storage diversity[C]// Proceedings of the 1st ACM Symposium on Cloud Computing. ACM, 2010: 229-240.

[9] BOWERS K D, JUELS A, OPREA A. HAIL: a high-availability and integrity layer for cloud storage[C]// Proceedings of the 16th ACM Conference on Computer and Communications Security. ACM, 2009: 187-198.

[10] SHEN L, FENG S, SUN J, et al. CloudS: A Multi-cloud Storage System with Multi-level Security[C]// International Conference on Algorithms and Architectures for Parallel Processing. Springer International Publishing, 2015: 703-716.

[11] MENON J, BRUCK J, BRADY J, et al. EVENODD: An Efficient Scheme for Tolerating Double Disk Failures in RAID Architectures[J]. IEEE Transactions on Computers, 1995, 44(2): 192-202.

[12] CORBETT P, ENGLISH B, GOEL A, et al. Row-diagonal parity for double disk failure correction[C]// Proceedings of the 3rd USENIX Symposium on File and Storage Technologies (FAST' 04). 2004: 1-14.

[13] XU L, BRUCK J. X-Code: MDS Array Codes with Optimal Encoding[J]. IEEE Transactions on Information Theory, 1999, 45 (1): 272-276.

[14] CHAO J, HONG J, DAN F, et al. P-Code: A new RAID-6 code with optimal properties[C]// 23rd International Conference on Supercomputing. New York, June 2009.

[15] HUANG C, XU L. STAR: An Efficient Coding Scheme for Correcting Triple Storage Node Failures[J]. IEEE Transactions on Computers, 2007, 57(7): 889-901.

[16] LIN S, WANG G, STONES D S, et al. T-Code: 3-Erasure Longest Lowest-Density MDS Codes[J]. IEEE Journal on Selected Areas in Communications, 2010, 28(2): 289-296.

[17] AUTHORS U. WEAVER codes: highly fault tolerant erasure codes for storage systems[C]// Fast 05 Conference on File & Storage Technologies, 2005: 16.

[18] PLANK J S, SCHUMAN C D, ROBISON B D. Heuristics for optimizing matrix-based erasure codes for fault-tolerant storage systems[J]. IEEE/IFIP International Conference on Dependable Systems & Networks Annual, 2012, 122(12): 1-12.

[19] SIMD[EB/OL]. [2016-07-09]. https://en. wikipedia. org/w/index. php?title=SIMD&oldid=726575429.

[20] PLANK J S. A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems[J]. Softw. , Pract. Exper. , 1997, 27 (9): 995-1012.

(上接第 7 页)

[59] SUN H, MA H, YIH W, et al. Open domain question answering via semantic enrichment[C]// Proceedings of the 24th International Conference on World Wide Web. Florence: ACM, 2015: 1045-1055.

[60] SEVERYN A, MOSCHITTI A. Automatic Feature Engineering for Answer Selection and Extraction[C]// Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. Seattle: ACL, 2013: 458-467.

[61] YAO X, VAN Durme B, CALLISON B C, et al. Answer Extraction as Sequence Tagging with Tree Edit Distance[C]// Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics. Atlanta: ACL, 2013: 858-867.

[62] SREELAKSHMI V, JAMAL S. Web Based Question Answering System using Pattern Matching[C]// The International Conference on Information Science. Pattaya: IEEE, 2015: 1-4.

[63] CHU CARROLL J, FAN J. Leveraging Wikipedia Characteristics for Search and Candidate Generation in Question Answering [C]// Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence. San Francisco: AAAI Press, 2011: 872-877.

[64] XU J, LICUANAN A, MAY J, et al. Answer Selection and Confidence Estimation[C]// New Directions in Question Answering. Stanford: AAAI Press, 2003: 134-137.

[65] ZHANG D, LEE W S. Web Based Pattern Mining and Matching Approach to Question Answering[C]// Proceedings of The Eleventh Text REtrieval Conference. Gaithersburg: National Institute of Standards and Technology, 2002: 129-141.

[66] MEDITSKOS G, DASIOPOULOU S, VROCHIDIS S, et al. Question Answering over Pattern-Based User Models[C]// Proceedings of the 12th International Conference on Semantic Systems. Leipzig: ACM, 2016: 153-160.

[67] YU Z T, FAN X Z, GUO J Y, et al. Answer extracting for chinese question-answering system based on latent semantic analysis[J]. Chinese Journal of Computer, 2006, 29(10): 1889-1893. (in Chinese)
余正涛, 樊孝忠, 郭剑毅, 等. 基于潜在语义分析的汉语问答系统答案提取[J]. 计算机学报, 2006, 29(10): 1889-1893.

[68] DEERWESTER S, DUMAIS S T, FURNAS G W, et al. Indexing by latent semantic analysis[J]. Journal of the American Society for Information Science, 1990, 41(6): 391.

[69] SUN H, DUAN N, DUAN Y, et al. Answer Extraction from Passage Graph for Question Answering[C]// Proceedings of the 23rd International Joint Conference on Artificial Intelligence. Beijing: IJCAI/AAAI, 2013: 2169-2175.

[70] FIGUEROA A G, NEUMANN G. Genetic algorithms for data-driven web question answering[J]. Evolutionary Computation, 2008, 16(1): 89-125.

[71] KHODADI I, ABADEH M S. Genetic programming-based feature learning for question answering[J]. Information Processing & Management, 2016, 52(2): 340-357.

[72] MA Y J, YUN W X. Research progress of genetic algorithm[J]. Application Research of Computers, 2012, 29(4): 1201-1206. (in Chinese)
马永杰, 云文霞. 遗传算法研究进展[J]. 计算机应用研究, 2012, 29(4): 1201-1206.

[73] MA C L, YAN Y H. Short Text Classification Based on Probabilistic Semantic Distribution[J]. Acta Automatica Sinica, 2016, 42(11): 1711-1717. (in Chinese)
马成龙, 颜永红. 基于概率语义分布的短文本分类[J]. 自动化学报, 2016, 42(11): 1711-1717.

[74] MA L. The Research and Implementation of Web-based Chinese Question Answering System[D]. Beijing: Beihang University, 2012. (in Chinese)
马琳. 基于 Web 的中文问答系统的研究与实现[D]. 北京: 北京航空航天大学, 2012.

[75] LEE J, KIM G, YOO J, et al. Training IBM Watson using Automatically Generated Question-Answer Pairs[C]// The 50th Hawaii International Conference on System Sciences. Hawaii: AIS Electronic Library, 2017: 1-9.