超立方体并行计算机的一个新型故障诊断算法*)

董 涛 杨小帆 柏 森
(重庆大学计算机学院 重庆 400044)

摘 要 提出了超立方体并行计算机的一个新型系统级故障诊断算法。与现有诊断算法相比,该算法能够在系统中存在较多故障处理器的情况下,正确定位全部故障处理器(代价是至多误诊断三个无故障处理器)。另外,该算法的时间复杂度与最好的现有算法相当。

关键词 系统级故障诊断,一步悲观诊断算法,超立方体

A Novel Fault Diagnosis Algorithm for Hypercube Multicomputer Systems

DONG Tao YANG Xiao-Fan BAI Sen
(College of Computer Science, Chongqing University, Chongqing 400044)

Abstract A novel system-level fault diagnosis algorithm is presented for hypercube multi-computer systems. As opposed to existing diagnosis algorithms, this algorithm can isolate all faulty processors to within a set with at most 3 fault-free processors provided that there are much more faulty units in the targeted system. In addition, our algorithm is comparable to the best known diagnosis algorithm in terms of time complexity.

Keywords System-leve diagnosis, Pessimistic one-step diagnosis, Diagnosis algorithm, Hypercube

1 引言

大型并行计算机系统在航天、军事、工业、天气预报等领域起着十分重要的作用。由于多处理器系统中的处理单元难免会发生故障,如何高效地检测故障处理单元并进行替换,成为保障系统高可用性的关键。系统级诊断的基本思想是通过处理单元的相互测试定位故障单元[3]。经典的基于 PMC 模型的系统级诊断假定测试结果只有两种可能:通过或者未通过。只有无故障处理器的测试结果才是可靠的。基于 PMC 模型的系统诊断有两条基本的诊断途径:(1)一步精确诊断:要求对每个处理器均进行正确判断。(2)一步悲观诊断[1]:要求将故障处理器隔离到一个较小的集合内,并允许误诊断少量无故障处理器。一步悲观诊断的优点是能诊断较多的故障处理器。

超立方体结构是在商业多处理器系统里普遍应用的一种拓扑结构。它具有很多优秀的性质。对于一个 n 维的超立方体系统 Q_n($N = 2^n$ 是 Q_n 的结点数)。一步精确诊断的诊断能力是 n。而一步悲观诊断的诊断能力是 $2n - 2$,其中最多有一个非故障处理器。针对一步悲观诊断,Yang, Masson, 和 Leonetti[4] 设计了一个适用于所有可诊断系统的诊断算法,时间复杂度为 $O(N^{2.5})$。最近,本文第二作者[5]提出了一个面向超立方体的诊断算法。当 $\log_2 N \geqslant 19$ 时,其时间复杂度为 $O(N\log_2 N)$。

本文提出了超立方体并行计算机的一个新型系统级故障诊断算法。与现有诊断算法相比,该算法能够在系统中存在较多故障处理器的情况下,正确定位全部故障处理器(代价是至多误诊断三个无故障处理器)。另外,该算法的时间复杂度与最好的现有算法相当。

本文第 2 节提供了有关预备知识;第 3 节是算法描述、正确性证明、以及时间复杂性分析;最后是结论。

2 预备知识

在这篇论文里,多处理器系统被抽象为图 $G = (V(G), E(G))$来研究,其中结点代表处理器而边代表处理器间的通信总线。我们用集合$\{0,1\}^n$ 代表所有长度为 n 的 0-1 二进制序列。n 维超立方体 Q_n 是一个每一个结点的度都是 n,结点的个数是 2^n,边的个数是 $n2^{n-1}$的图。一个 0-1 二进制序列表示 Q_n 中的一个结点。因此,Q_n 中的结点被一一映射到了集合$\{0,1\}^n$ 中的 0-1 二进制序列,而 Q_n 中的边则代表两个相邻结点所对应的 0-1 二进制序列仅有一个 bit 位不同。在图 1(a)中,我们给出了一个 Q_4。一个 0-1 二进制序列 z 中的 1 的个数被称为海明距离 $H(z)$。对结点诊断的过程中,我们假设有边直接相连的两个结点之间可以互相诊断。$s(u,v)$表示结点 u 诊断结点 v 的诊断结果。$s(u,v)$的结果是 0 还是 1 依赖于结点 u 诊断结点 v 是无故障结点还是有故障节点。断定一个结点是故障还是无故障是根据 PMC 模型(表 1)。

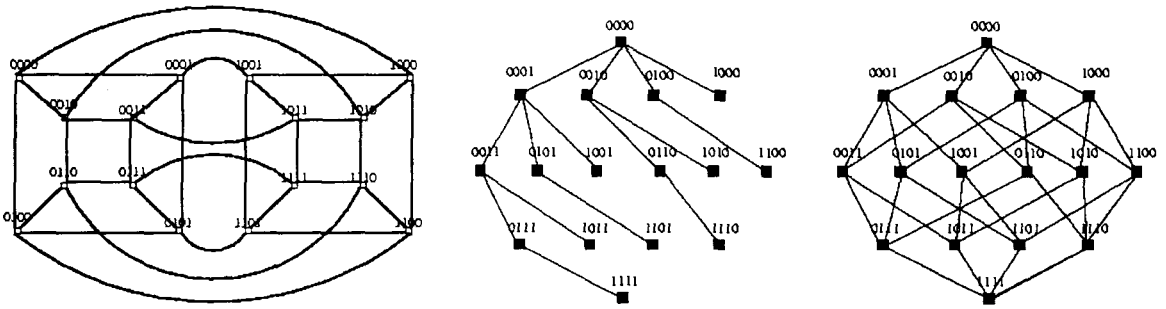
表 1 PMC 模型

测试者 u	被测试者 v	测试结果 $s(u, v)$
无故障结点	无故障结点	0
无故障结点	故障结点	1
故障结点	无故障结点	0 或 1
故障结点	故障结点	0 或 1

层次化超立方体 L_n 是对超立方体的所有结点进行偏序化得到的结果[2]。超立方体的边反映了结点间的偏序关系。具有相同海明距离的结点被放在同一层,海明距离最小的结

*)本文工作得到重庆市应用基础研究基金课题资助(批准号:8028)。董 涛 硕士生,研究方向:系统级故障诊断。杨小帆 教授,博士生导师,研究方向:并行计算,系统级故障诊断,神经网络,差分方程。柏 森 教授,博士,研究方向:图像处理,信息安全。

点作为第0层，海明距离最大的结点作为第 n 层[2]。分别位于相邻两层中的两个相邻结点，海明距离小的结点作为父结点。在同一层中，结点的排列顺序是根据这一层结点的父结点的排列顺序来排序。根据父结点的排列顺序，对于不同的父结点 x，把它的每一个子结点 y（仅当 y 没有在它的其他的父结点的排序过程中被排过序）按照 $x \oplus y$ 从小到大来排序，并且在 x 与 y 之间加上边以表示父子关系，其中⊕是异或位运算符。这样，我们就得到一个树 HT_n。如果把结点间的关系补全，就得到 L_n。在图1(b)中，我们给出了一个 HT_4。在图1(c)中，是对应的 L_4。



(a) (b) (c)

图1 四维超立方体的三种画法

本文第二作者等人[6,7]证明了：

(1)当一个 n 维超立方体中有最多包含 $2n-3$ 个故障处理器的集合 F 时($n=3$)，此超立方体有一个大小为 $2^n-|F|-1$ 的连通分图。

(2)当 $|F|=3n-6$ 时，连通分图的大小为 $2^n-|F|-2$($n=4$)。

(3)当 $|F|=4n-10$ 时，连通分图的大小为 $2^n-|F|-3$($n=6$)。

3 超立方体的诊断算法

根据文[6,7]中关于最大连通分图的理论，我们通过遍历 HT_n 来搜索超立方体中的连通分图以找到最大的连通分图。剩下的结点集就是故障结点集(包含部分无法诊断的结点)。

引理1 当 $n=6$ 并且故障集的大小 $|F|=4n-10$ 时，在遍历 HT_n 的过程中，连通分图的个数不超过 $8n-20$。

证明 因为故障集的大小 $|F|=4n-10$，所以故障结点形成的连通分图的个数不超过 $4n-10$。在遍历 HT_n 寻找连通分图的过程中，本来属于同一个连通分图的非故障结点在遍历 HT_n 还未结束前，会被故障结点分开而形成不同的连通分图。由于非故障结点被故障结点临时分开而形成的这种临时的连通分图的个数不会超过 $4n-10$，因为故障结点总共 $4n-10$。因此，连通分图的个数不超过 $8n-20$。

引理2 在 L_n 中，结点 y 的孩子结点 x 可以通过把 y 中的一个是0的bit位翻转为1得到。结点 y 的父亲结点 z 可以通过把 y 中的一个是1的bit位翻转为0得到。

在此算法中，为了使算法效率达到最优，我们将用到3个数据结构，一个大小为 $8n-20$ 的数组 list 和一个大小为 2^n 的数组 status。描述如下：

1 status[i]($i \in \{0,1\}^n$)表示结点 i 所在的连通分图。初始化为0，即不属于任何连通分图。

```
2  struct header{
            struct node * next;
            int count;
       }list[8n-20];
```

数组 list[i]($i \neq 0$) 表示第 i 个连通分图的入口点，每个连通分图用一个链表来表示。第 i 个连通分图挂在 list[i].next 上。list[i].count 表示第 i 个连通分图中结点的个数。

```
3  struct node {
            struct node * next;
            x∈{0,1}^n;
       }
```

这个数据结构用于把结点信息保存到代表连通分图的链表中，来构造连通分图。

```
4  struct treenode{
            struct treenode * next;
            x∈{0,1}^n;
            indicator∈{0,1}^n;
       }
```

此数据结构用于构造超立方体树 HT_n。元素 indicator 是一个0-1序列，其中只有一个bit位是1。我们把这个是1的bit位叫做指示位。每一个结点有一个 indicator。通过 indicator 来构造 HT_n。构造 HT_n 的过程如下：

算法 BUILD_TREE

第一步：我们用链表 bottom 表示当前已经创建的树的最下面的一层，链表 nextesterlayer 表示树的正在被创建的层。初始化 HT_n 的根结点 z 为0，结点 z 的 indicator 为1。并把 z 和它的 indicator 保存到链表 bottom 中作为第0层。

第二步：取出链表 bottom 中的一个未访问过的结点 x 和它的 indicator。

第三步：用等式 $y=x \oplus$indicator 得到 x 的儿子结点 y。把 x 的 indicator 左移一个 bit 位并且作为 y 的 indicator。把 y 和它的 indicator 放入链表 nextesterlayer。

第四步：如果 x 的 indicator 不为0，则说明在 HT_n 中 x 还有儿子结点，跳到第三步执行。

第五步：如果 bottom 中还有未访问过的结点，跳到第二步执行。

第六步：如果链表 bottom 不是 HT_n 的第 $n-1$ 层，则把链表 nextesterlayer 作为新的 bottom 链表，跳到第二步执行。

新的超立方体诊断算法的描述如下：

算法 COM_SEARCH

输入：整数 $n=6$。Q_n 的诊断结果为 s。

输出：故障集 V。

第一步(初始化)：把数组 status 初始化为0。数组 list 所有单元的指针 next 设置为 NULL，count 设置为0。把海明距离为0的结点加入链表 list[1]中作为第一个连通分图，list[1].count=1。status[0]=1。

第二步：从 L_n 的海明距离为0的结点出发，根据算法 BUILD_TREE 构造 HT_n。在构造的过程中，在算法 BUILD_TREE 的第四步和第五步之间调用函数 DIAG_CONN 对结点 x(它是从链表 bottom 中取出的)与其每一个孩子结点 y 进行诊断。

第三步：当构造了 HT_n 后，就会得到一些连通分图。其中除了最大的那个连通分图，剩下的就是故障集 V（包括部分不可诊断的结点）。

函数 DIAG_CONN 的描述如下：

```
函数 DIAG_CONN
参数：父结点 x。数组 status。数组 list。
1: for each y∈{x 的孩子结点}
2:     if(s(x,y)和σ(y,x)均为0)
3:         if(结点 x 不属于任何连通分图，即：status[x]=0)
4:             把 x 放入一个新的空连通分图中。
5:         if(结点 y 不属于任何连通分图，即：status[y]=0)
6:             把结点 y 连到 x 所在的连通分图，使 x,y 在同一个连
```

通分图里。
7: if(结点 y 已经属于另外一个连通分图)
8: 通过比较 list[status[x]].count 与 list[status[y]].count 的大小,把小的连通分图连到大的连通分图并修改小的连通分图的状态信息。

定理 3 如果 $n=6$ 并且 $|F|=4n-10$,则算法 COM_SEARCH 可以诊断出一个故障结点集 $V(|V|=4n-7)$,其中最多包含 3 个不可诊断的结点。

证明:算法 COM_SEARCH 把结点间互相诊断结果均为 0 的结点连接起来以形成连通分图,并把分别属于两个连通分图的相邻结点的互相诊断结果均为 0 的结点连接起来,从而把它们所在的两个连通分图连接起来而形成了一个大的连通分图。根据 PMC 模型,两个无故障结点间互相诊断的结果均为 0,所以无故障结点可以形成连通分图。两个有故障结点间互相诊断的结果不定,也可能它们互相诊断结果均为 0,所以也可能形成连通分图。但是,故障结点形成的连通分图和无故障结点形成的连通分图是不会连通的,因为这两个连通分图中相临的结点中总是有一个是无故障结点,它对故障结点的诊断结果总是 1。又因为文[7]中证明了当 $|F|=4n-10$ 时,最大连通分图的大小为 $2^n-|F|-3(n=6)$。所以算法 COM_SEARCH 可以诊断出故障结点集 V,并且 $|V|=4n-7$ 其中最多包含 3 个不可诊断的结点。

引理 4 函数 DIAG_CONN 的时间复杂度是 $O(n)$。

证明:在函数 DIAG_CONN 中,3-7 行程序需要 $O(n)$ 时间。第 8 行程序需要分两种情况讨论:

1)对于故障结点:因为故障结点互相测试为 0 的概率很小,并且故障结点在超立方体中的分布是稀疏分布。所以,我们可以认为故障结点形成的连通分图的长度为常数。

2)对于无故障结点:因为最多有 3 个可能是无故障结点的结点所组成的连通分图会被故障结点分离出大的无故障结点的连通分图,当一个链表需要被连接到另一个链表时,仅有不超过 4 个无故障结点需要被改变结点状态。

因此第 8 行程序需要 $O(n)$ 时间。所以函数 DIAG_CONN 时间复杂度为 $O(n)$。□

定理 5 算法 COM_SEARCH 的时间复杂度是 $O(N\log_2 N)$。

证明:在算法 COM_SEARCH 的第一步中,数组 status 的大小为 N,数组 list 的大小为 $8\log_2 N-20$,所以需要 $O(N)$ 时间对其进行初始化。在第二步中,算法 BUILD_TREE 的时间复杂度为 $O(N)$。函数 DIAG_CONN 是在算法 BUILD_TREE 中调用的函数,而函数 DIAG_CONN 的时间复杂度是 $O(\log_2 N)$,所以第二步需要 $O(N\log_2 N)$时间。由于连通分图的个数不超过 $8\log_2 N-20$,所以第三步需要 $O(\log_2 N)$时间。

因此,算法 COM_SEARCH 的时间复杂度是 $O(N\log_2 N)$。□

结论 在这篇论文中,我们设计了一个适用范围较广的时间复杂度为 $O(N\log_2 N)$的新算法。此算法在构造超立方体树 HT_n 的过程中,来建立这个超立方体中的连通分图。从而诊断出故障结点集。当 $n=6$ 并且 $|F|=4n-10$ 时,此算法时间复杂度为 $O(N\log_2 N)$。

参考文献

1 Kavianpour A, Kim K H. Diagnosabilities of hypercubes under the pessimistic one-step diagnosis strategy. IEEE Transactions on Computers, 1991, 40: 232～237

2 Liu J, Chen Y. On the Distributed Subcube-allocation Strategies in the Hypercube Multiprocessor Systems. In: Proc. of the 4th IEEE Symposium on Parallel and Distributed Processing, 1992. 360～364

3 Preparata F P, Metze G, Chien R T. On the connection assignment problem of diagnosable systems. IEEE Transactions on Electronic Computers, 1967, EC-16: 848～854

4 Yang C-L, Masson G M, Leonetti R A. On fault isolation and identification in t1/t1-diagnosable systems. IEEE Transactions on Computers, 1986, C-35: 639～643

5 Yang X. A fast pessimistic one-step diagnosis algorithm for hypercube multicomputer systems. Journal of Parallel and Distributed Computing, 2004, 64(4): 546～553

6 Yang X, Evans D, Chen B, Megson G, Lai H. On the maximal connected component of hypercube with faulty vertices, International Journal of Computer Mathematics, 2004, 81(5): 515～525

7 Yang X, Evans D, Megson G. On the maximal connected component of hypercube with faulty vertices(II). International Journal of Computer Mathematics, 2004, 81(10): 1175～1185

(上接第 245 页)

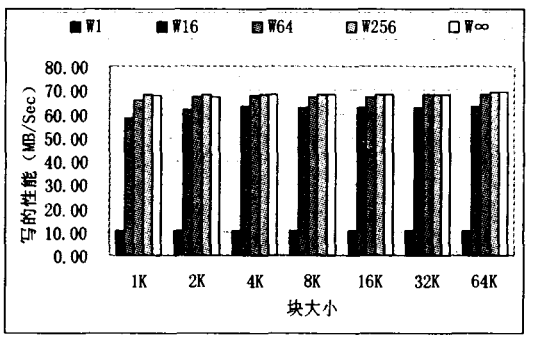


图 3 蓝鲸分布式文件系统写性能

结论 蓝鲸分布式文件系统通过元数据和数据分离机制,管理多个存储服务器,实现数据的并发访问,大大提高了整个存储系统的性能和扩展性。同时,我们通过一次请求多个元数据映射信息,减少了交换元数据映射的网络通信和开销;通过有效的缓存表示、替换机制和失效机制,尽量减少内核内存空闲占用和保持信息有效性。测试结果表明这种模型极大地提高了系统的数据读写性能。

参考文献

1 Shepler S, Callaghan B. RFC 3530: Network File System (NFS) version 4 Protocol. The Internet Society, 2003

2 Klosterman A J, Ganger G. Cuckoo: layered clustering for NFS. Technical Report CMU-CS-02-183. Carnegie Mellon University, Oct. 2002

3 Leach P, Perry D. CIFS: A Common Internet File System. Microsoft Interactive Developer, Nov. 1996

4 Machek P. Network Block Device. http://atrey.karlin.mff.cuni.cz/~pavel/nbd/nbd.html

5 Meth K Z, Satran J. Design of the iSCSI Protocol. In: 20 th IEEE/11 th NASA Goddard Conf. on Mass Storage Systems and Technolog