



基于二元决策图的集群计算系统性能分析

许美玲*, 乔莹, 莫毓昌, 钟发荣

(浙江师范大学 数理与信息工程学院, 浙江 金华 321004)

(*通信作者电子邮箱 1933785432@qq.com)

摘要:针对节点计算能力相同但故障分布不同的集群系统的性能分析问题,基于 k -to- l -out-of- n 结构对集群系统的性能进行建模,并提出了一种基于二元决策图(BDD)的分析方法。针对 k -to- l -out-of- n 结构的 BDD 模型生成问题,分析了 BDD 的结构特征并设计自顶向下生成算法,克服了传统的自底向上生成算法必须生成大量中间冗余节点的缺陷;然后利用生成的 BDD 模型高效地计算出系统处于一个特定性能级别的概率;最后通过实例说明了 BDD 方法能够有效分析节点具有不同故障分布的集群系统性能。

关键词:集群计算系统; k -to- l -out-of- n 模型; 二元决策图

中图分类号: TP302 **文献标志码:** A

Performance analysis of cluster computing systems using binary decision diagram

XU Meiling*, QIAO Ying, MO Yuchang, ZHONG Farong

(College of Mathematics, Physics and Information Engineering, Zhejiang Normal University, Jinhua Zhejiang 321004, China)

Abstract: To analyze the performance of cluster computing systems with identical computing power but different failure distribution, a k -to- l -out-of- n structure was used to model system performance, and a new analytical method based on Binary Decision Diagram (BDD) was proposed for the performance analysis. A new and efficient BDD algorithm that makes full use of the special k -to- l -out-of- n structure was also proposed using a top-down manner, which solved the problem that the traditional bottom-up generation algorithm must generate a large number of intermediate redundant nodes. Then the proposed BDD was used to efficiently calculate the probability of the system being at a specific performance level. At last, some examples were provided to illustrate the proposed BDD-based performance analysis methodology as well as its efficiency in analyzing large-scale cluster computing systems.

Key words: cluster computing system; k -to- l -out-of- n model; Binary Decision Diagram (BDD)

0 引言

现代计算机集群系统包含大量计算节点,这些节点可以是处于分布式环境下或不同管理域中的独立计算机,例如云计算系统或网格计算系统^[1-2],也可以是由 Infiniband 或三维 torus 互联系统连接的单独处理器,例如超级计算系统^[3-5]。

假设 p_i 表示节点 N_i 的计算能力,由 n 个节点组成的计算系统,其计算能力 MP 表示为 $MP = p_1 + p_2 + \dots + p_n$ 。节点出现故障会导致计算系统性能降低,即节点 N_i 出现故障,系统的计算能力将变为 $MP - p_i$ 。若节点具有相同的计算能力 p ,则系统所能呈现的状态数量共有 $n + 1$ 种情况,计算能力分别为 $0, p, 2p, \dots, np$,其中:0 表示所有节点都发生故障, np 表示所有节点都正常运行。需要说明的是,本文考虑的集群系统满足:集群的整体性能可以近似为各节点计算能力之和,例如具有前端分发节点的 n 节点 Web 集群。当然还存在其他很多分布式计算应用, n 个节点之间需要进行进一步的交互以完成计算。对于这些交互应用,“ n 节点集群计算能力是各节点计算能力之和”这一条件将不满足,该类型集群系统的性能分

析研究不属于本文的研究范围。

当节点具有相同计算能力时,将计算能力归一化处理,基于 k -to- l -out-of- n 结构对系统性能进行建模分析,在 k -to- l -out-of- n 结构的 n 个节点中,需要不少于 k 个但不多于 l 个可以正常运行的节点^[6-7]。分析系统性能时,所有性能状态可以分成若干种不同情况: L_1, L_2, \dots, L_m 。例如由 n 个节点组成的计算系统,如果每个节点的计算能力都为 p ,采用平均划分的方法,可以把系统性能级别从高到低一共划分若干个级别。例如对于 n 个节点系统其性能可以简单划分为 5 个级别,为了进行区分可以分别命名为:极低 ($0 \leq P \leq np/5 - 1$),低 ($np/5 \leq P \leq 2np/5 - 1$),中等 ($2np/5 \leq P \leq 3np/5 - 1$),高 ($3np/5 \leq P \leq 4np/5 - 1$),极高 ($4np/5 \leq P \leq np$)。其中极高性能级别包含 n 个节点全部正常工作的情况;而极低性能级别包含 n 个节点全部失效的情况。系统的性能分析可表示为系统处于一个特定性能的概率计算,即计算系统性能 P 落在某个区间内的概率。上述例子中极低、低、中等、高、极高五种情况分别对应以下结构: 0 -to- $(n/5 - 1)$ -out-of- n , $n/5$ -to- $(2n/5 - 1)$ -out-of- n , $2n/5$ -to- $(3n/5 - 1)$ -out-of- n , $3n/5$ -to- $(4n/5 - 1)$ -out-of- n , $4n/5$ -to- n -out-of- n 。

收稿日期:2016-08-02;修回日期:2016-08-31。

基金项目:国家自然科学基金面上项目(61272130,61572442);浙江省公益性项目(2015C33085)。

作者简介:许美玲(1992—),女,浙江杭州人,硕士研究生,主要研究方向:决策图分析; 乔莹(1992—),女,陕西榆林人,硕士研究生,主要研究方向:决策图分析; 莫毓昌(1980—),男,浙江湖州人,副教授,博士,CCF 会员,主要研究方向:高可靠计算; 钟发荣(1963—),男,浙江龙游人,教授,博士,主要研究方向:并行计算。



当节点数量较大时,采用状态穷举的方法效率较低,此时可以采用含非逻辑的非单调关联故障树分析方法^[8]。基于质蕴含的非单调关联故障树分析与基于最小割集的单调关联故障树分析类似,同样存在指数复杂性^[9]问题。最近,一些基于二元决策图(Binary Decision Diagram, BDD)^[10]的组合方法开始应用于故障树分析^[11-14]。与其他方法相比,基于二元决策图的方法显得更加有效,因为 BDD 可以有效描述布尔函数,并且 BDD 上的操作可以高效地实现各种布尔运算。但现有的一些基于 BDD 的方法使用自底向上 BDD 生成算法,容易产生大量冗余的中间节点,而且这些方法也很难利用 k -to- l -out-of- n 模型的特殊结构,因此并不能提高模型的生成效率。

本文使用基于 BDD 的分析方法,对具有 k -to- l -out-of- n 结构的大型集群系统进行性能分析。在任务执行期间,集群及其节点都是不可修复的。针对节点计算能力相同但故障分布不同的集群,利用 k -to- l -out-of- n 结构下 BDD 模型所具有的特殊结构特征,设计了自顶向下 BDD 生成算法,避免了传统自底向上 BDD 生成算法中大量中间节点的生成。

1 问题描述

形式上,二进制随机变量 X_i 表示节点 N_i 的状态, $X_i = 1$ 表示节点 N_i 处于非故障状态, $X_i = 0$ 表示节点 N_i 处于故障状态;由 n 个节点组成的计算系统,用 $\mathbf{X} = (X_1, X_2, \dots, X_n)$ 表示随机的系统状态向量。因为每个节点 N_i 有两种状态(故障状态或非故障状态),所以由 n 个节点组成的计算系统有 2^n 种可能的状态。

$\mathbf{x} = (x_1, x_2, \dots, x_n)$ 表示系统状态, x_i 表示节点 N_i 的状态。系统在任务时间 t 内处于状态 \mathbf{x} 的概率计算公式如下:

$$\Pr(\mathbf{X} = \mathbf{x}) = \prod_{i=1}^n (x_i \cdot (1 - F_i(t)) + (1 - x_i) \cdot F_i(t)) \quad (1)$$

$F_i(t)$ 表示节点 N_i 在任务时间 t 内处于 $X_i = 0$ (故障状态) 时的概率。

系统在任务时间 t 内处于状态 \mathbf{x} 的计算能力 P 的计算公式如下:

$$P(\mathbf{X} = \mathbf{x}) = \sum_{i=1}^n x_i \cdot p_i \quad (2)$$

系统在任务时间 t 内以特定性能 L 运行的概率,用计算能力 P 落在区间 $[LB, UB]$ 内的概率表示, LB 表示下限, UB 表示上限,计算公式如下:

$$\Phi_L = \Pr(LB \leq P \leq UB) = \sum_{LB \leq P(\mathbf{X}=\mathbf{x}) \leq UB} \Pr(\mathbf{X} = \mathbf{x}) \quad (3)$$

本文讨论的问题是如何在特定任务时间 t 内计算 Φ_L , 作以下假设:

1) 计算节点之间相互独立运行,即在整个系统中,一个节点事件(比如节点出现故障)的发生不会影响另一个节点事件的发生;

2) 每个节点处于每个状态的概率可以通过给定参数直接计算获得,即本文只考虑系统级的性能分析,部件级的性能分析不在本文讨论范围内;

3) 系统在使用时不可修复,即节点一旦从非故障状态转换到故障状态,在剩余任务时间内该节点都处于故障状态。

2 实例说明

实例系统由 6 个计算节点组成: N_1, N_2, \dots, N_6 , 任务持续

时间为 100 h。为了说明集群的各个节点具有不同寿命分布,假设节点 N_1, N_2, N_3 服从参数为 λ 的指数故障分布,累积分布函数 $F_i(t)$ 如下:

$$F_i(t) = 1 - \exp(-\lambda_i t) \quad (4)$$

假设节点 N_4, N_5, N_6 服从参数为 λ 和 β 的 Weibull 故障分布,累积分布函数 $F_i(t)$ 如下:

$$F_i(t) = 1 - \exp(-(\lambda_i t)^{\beta_i}) \quad (5)$$

表 1 显示了任务时间为 100 h, 参数 λ_i 和 β_i 分别为给定值时,使用式(4)或(5)计算得出的 $F_i(t)$ 。

表 1 节点参数和计算所得的 $F_i(t)$

Tab. 1 Node parameters and calculated $F_i(t)$

N_i	λ_i	β_i	$F_i(t = 100 \text{ h})$	N_i	λ_i	β_i	$F_i(t = 100 \text{ h})$
N_1	0.003	1.0	0.2592	N_4	0.001	1.3	0.0489
N_2	0.002	1.0	0.1813	N_5	0.001	1.6	0.0248
N_3	0.001	1.0	0.0952	N_6	0.001	1.5	0.0311

节点 N_i 的计算能力为 p_i , 实例系统的最大计算能力 MP 表示为 $MP = p_1 + p_2 + p_3 + p_4 + p_5 + p_6$ 。不失一般性,系统性能 P 的三种性能分别定义为:低 $L_L (LB_L \leq P \leq UB_L)$, 中 $L_M (LB_M \leq P \leq UB_M)$, 高 $L_H (LB_H \leq P \leq UB_H)$, $LB_i (i \in \{L, M, H\})$ 表示下限, $UB_i (i \in \{L, M, H\})$ 表示上限。

当节点相同且计算能力也相同时,将 p_i 归一化处理。表 2 显示了对应三种性能的 LB 和 UB 值,以及对应的 k -to- l -out-of- n 模型。

表 2 不同性能下的参数设置(p_i 相同)

Tab. 2 Parameter setting under different performance levels (same p_i)

性能	性能下限	性能上限	最大计算能力 MP	k -to- l -out-of- n 模型
L_L	0	1	6	0-to-1-out-of-6
L_M	2	4	6	2-to-4-out-of-6
L_H	5	6	6	5-to-6-out-of-6

本文所研究的集群系统性能分析问题可以描述为:在任务时间内,分析集群系统在表 1 和表 2 的参数设置下,系统处于性能状态 L_L, L_M 和 L_H 的概率。需要注意的是,本文关注的是在特定时间内的概率计算,而不是处于稳定状态或长期运行状态的概率计算。

3 基于 BDD 的性能分析方法

本文提出一种基于 BDD 的性能分析方法对节点计算能力相同的集群系统进行性能分析。该方法由两个步骤组成:1) 使用 k -to- l -out-of- n 结构生成 BDD 模型;2) 分析 BDD 模型并得到系统性能评估指标。

3.1 生成 BDD 模型

每个二状态节点 N_i 的状态空间存在两种互斥的状态:0 表示故障状态,1 表示非故障状态。如图 1 所示,BDD 模型中每个节点 N_i 都有两条输出边:then-edge 用 1 表示(非故障状态),else-edge 用 0 表示(故障状态)。

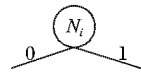


图 1 与 N_i 关联的 BDD 节点

Fig. 1 BDD node associated with N_i

由表 2 可知,2-to-4-out-of-6 模型表示 L_M , 该模型生成的 BDD 结构如图 2 所示。

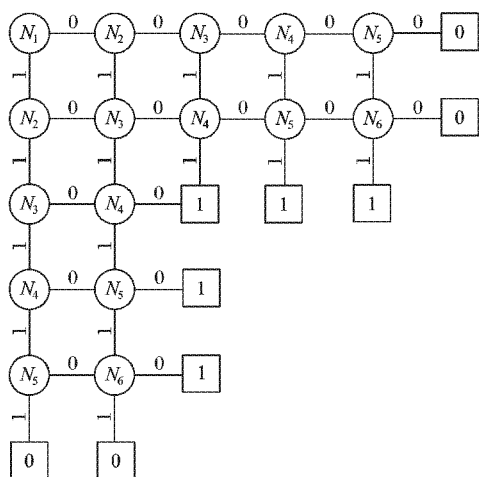


图2 2-to-4-out-of-6模型的BDD结构

Fig. 2 BDD for 2-to-4-out-of-6 model

BDD结构中有两种叶子节点(正方形表示叶子节点):叶子节点“1”表示系统性能为 L_M ,而叶子节点“0”表示系统性能不为 L_M ,比如计算能力高于 UB_M 或者低于 LB_M 的情况。例如,当节点 N_1, N_2, \dots, N_5 都处于0状态(故障状态),无论节点 N_6 处于什么状态,系统的计算能力都低于 LB_M ,即系统性能不为 L_M 。这种情况下的路径与叶子节点“0”相连(如图2中顶部的水平路径所示)。另一种情况,当节点 N_1, N_2, \dots, N_5 都处于1状态(非故障状态),无论节点 N_6 处于什么状态,系统的计算能力都高于 UB_M ,即系统性能不为 L_M 。这种情况下的路径也与叶子节点“0”相连(如图2中最左侧的垂直路径所示)。

总之,对任何基于 k -to- l -out-of- n 结构的计算系统进行BDD建模,如果节点计算能力相同,那么BDD模型可以由以下引理推导得到:

引理 基于 k -to- l -out-of- n 结构的计算系统,性能为 L_i 时对应的BDD模型如图3所示,即一个 $(l+1) \times (n-k+1)$ 的矩阵去除右下角 $(l-k+1) \times (l-k+1)$ 的矩阵后剩余的部分。

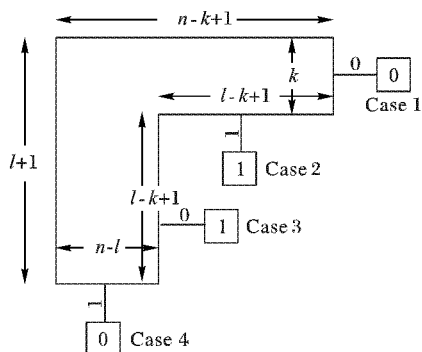


图3 基于 k -to- l -out-of- n 结构的BDD模型

Fig. 3 BDD for k -to- l -out-of- n structure

证明 在性能为 L_i 的BDD模型中,即系统计算能力 P 的取值范围是 $[LB_i, UB_i]$,所有路径可以分为四种情况,每种情况的解释如下:

1) 如果超过 $n-k$ 个节点处于0状态,无论其他节点处于什么状态,系统的计算能力都低于 LB_i ,即系统性能不为 L_i 。对应这种情况的路径和叶子节点“0”相连。

2) 如果超过 $n-l$ 个节点处于0状态,并且至少有 k 个节

点处于1状态,无论其他节点处于什么状态,系统的计算能力 P 都在 $[LB_i, UB_i]$ 中,即系统性能为 L_i 。对应这种情况的路径和叶子节点“1”相连。

3) 如果超过 k 个节点处于1状态,并且至少有 $n-l$ 个节点处于0状态,无论其他节点处于什么状态,系统的计算能力 P 都在 $[LB_i, UB_i]$ 中,即系统性能为 L_i 。对应这种情况的路径和叶子节点“1”相连。

4) 如果超过 l 个节点处于1状态,不管其他节点处于什么状态,系统的计算能力都高于 UB_i ,即系统性能不为 L_i 。对应这种情况的路径和叶子节点“0”相连。

为了方便描述基于晶格结构的 k -to- l -out-of- n 模型的BDD生成过程,将晶格结构置于二维坐标系中。如图4所示,用坐标 (x, y) 表示非叶子节点的位置。如果一个BDD节点的坐标为 (x, y) ,那么可以用 $x+y+1$ 表示该节点的下标,例如表示为节点 N_{x+y+1} 。举例说明,坐标为 $(0, 0)$ 的节点用 N_1 表示,坐标为 $(2, 2)$ 的节点用 N_5 表示。

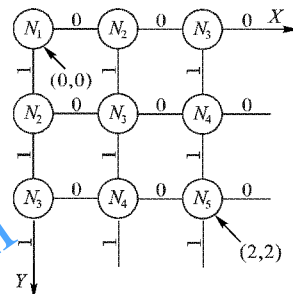


图4 用坐标 (x, y) 表示的BDD结构

Fig. 4 BDD structure represented by (x, y)

基于 k -to- l -out-of- n 结构使用自顶向下算法进行BDD建模的代码描述如下:

```
BDD( $k, l, n$ ) = index = 0; //首节点位置为0
For each  $y$  on vertical axis,  $0 \leq y \leq l$ 
    If ( $y < k-1$ ) //节点对应的 $y$ 坐标值
        //处理所有可能的 $x$ 坐标值
        For each  $x$  on horizontal axis,  $0 \leq x \leq n-k$ 
            If ( $x = n-k$ );  $E = '0'$  //设置节点 else-edge 连接
            Else  $E = index + 1$  //设置节点 else-edge 连接
             $T = index + n - k + 1$  //设置节点 then-edge 连接
            create-BDD-node( $x+y+1, E, T$ ); //创建节点
            index ++; //更新节点位置
    If ( $y = k-1$ )
        For each  $x$  on horizontal axis,  $0 \leq x \leq n-k$ 
            If ( $x < n-l$ );  $E = index + 1$ ;  $T = index + n - k + 1$ 
            Else If ( $x < n-k$ );  $E = index + 1$ ;  $T = '1'$ 
            Else  $E = '0'$ ;  $T = '1'$ 
            create-BDD-node( $x+y+1, E, T$ ); index ++;
    If ( $k-1 < y < l$ )
        For each  $x$  on horizontal axis,  $0 \leq x \leq n-l-1$ 
            If ( $x = n-l-1$ );  $E = '1'$ 
            Else  $E = index + 1$ 
             $T = index + n - k + 1$ 
            create-BDD-node( $x+y+1, E, T$ ); index ++;
    If ( $y = l$ )
        For each  $x$  on horizontal axis,  $0 \leq x \leq n-l-1$ 
            If ( $x = n-l-1$ );  $E = '1'$ 
            Else  $E = index + 1$ 
             $T = '0'$ 
            create-BDD-node( $x+y+1, E, T$ ); index ++;
```



Return $index$

“create-BDD-node($x+y+1, E, T$)”操作表示把一个新 BDD 节点添加到数组中。用 $index$ 记录这个新节点在数组中的位置,新节点的编号为“ $x+y+1$ ”。新节点的 then-edge 与一个 BDD 节点相连,数组中的 T 记录该 BDD 节点的位置 $index$, else-edge 与另一个 BDD 节点相连;数组中的 E 记录该 BDD 节点的位置 $index$ 。3.3 节中将对上述算法和传统算法进行性能比较,进一步说明上述算法的正确性。以下简要说明上述 BDD 生成算法的正确性:

如果一个 BDD 节点的坐标为 (x, y) , 节点的 then-edge 和 else-edge 坐标计算如下:

1) $y \in [0, k-1], x \in [0, n-k]$, 每个节点的 then-edge 与编号为“ $x+y+2$ ”并且坐标为 $(x, y+1)$ 的非叶子节点相连。如果 $x = n-k$, 那么这个节点的 else-edge 与叶子节点“0”相连;否则这个节点的 else-edge 与编号为“ $x+y+2$ ”并且坐标为 $(x+1, y)$ 的非叶子节点相连。

2) $y = k-1, x \in [0, n-k]$ 。如果 $x \in [0, n-l]$, 那么这个节点的 else-edge 与编号为“ $x+y+2$ ”并且坐标为 $(x+1, y)$ 的非叶子节点相连,节点的 then-edge 与编号为“ $x+y+2$ ”并且坐标为 $(x, y+1)$ 的非叶子节点相连;如果 $x \in [n-l, n-k]$, 那么这个节点的 else-edge 与编号为“ $x+y+2$ ”并且坐标为 $(x+1, y)$ 的非叶子节点相连,节点的 then-edge 与叶子节点“1”相连;如果 $x = n-k$, 那么节点的 else-edge 与叶子节点“0”相连,节点的 then-edge 与叶子节点“1”相连。

3) $y \in (k-1, l), x \in [0, n-l-1]$ 。每个节点的 then-edge 与编号为“ $x+y+2$ ”并且坐标为 $(x, y+1)$ 的非叶子节点相连。如果 $x = n-l-1$, 那么节点的 else-edge 与叶子节点“1”相连;否则,节点的 else-edge 与编号为“ $x+y+2$ ”并且坐标为 $(x+1, y)$ 的非叶子节点相连。

4) $y = l, x \in [0, n-l-1]$ 。每个节点的 then-edge 与叶子节点“0”相连。如果 $x = n-l-1$, 那么这个节点的 else-edge 与叶子节点“1”相连;否则,这个节点的 else-edge 与编号为“ $x+y+2$ ”并且坐标为 $(x+1, y)$ 的非叶子节点相连。

3.2 评估 BDD 模型

在基于 k -to- l -out-of- n 结构的 BDD 模型中,计算每个 BDD 非叶子节点两个边的概率,就是计算该节点是处于非故障状态(then-edge)还是故障状态(else-edge)的概率,分别用 $1-F_j$ 和 F_j 表示。

从根节点到叶子节点的每条路径都代表了一组节点的一个状态组合。如果到达叶子节点“1”,那么可以用这条路径或这个状态组合来表示系统性能 L_i 。因此,系统处于性能 L_i 就可以用从根节点到叶子节点“1”的所有路径的概率之和表示。

对性能为 L_i 时生成的 BDD 模型,其性能评估可以通过以下递归算法表示。

$$\Pr(BDD) = (1 - F_j) \cdot \Pr(BDD.T) + F_j \cdot \Pr(BDD.E) \quad (6)$$

其中: F_j 表示 BDD 中根节点 N_i 处于故障状态的概率; $\Pr(BDD)$ 表示 BDD 的概率; $BDD.T$ 是与根节点的 then-edge 相连的子 BDD, $BDD.E$ 是与根节点的 else-edge 相连的子 BDD。

评估 BDD 模型的递归算法,代码如下:

```
Evaluate(BDD) =
If (BDD = '1'); // 若 BDD 节点是叶子节点“1”
Return 1
If (BDD = '0'); // 若 BDD 节点是叶子节点“0”
Return 0
If (BDD.Pr has been computed); // 若已经计算过
Return BDD.Pr // 按照式(6)计算
BDD.Pr = (1 - F_j) * Evaluate(BDD.T) + F_j * Evaluate(BDD.E)
Return BDD.Pr
```

递归算法的结束条件是,如果 BDD 节点是叶子节点“0”,那么 $\Pr(BDD) = 0$; 如果 BDD 节点是叶子节点“1”,那么 $\Pr(BDD) = 1$ 。

对于第 2 章给出的实例系统,其节点具有相同计算能力,表 3 给出了实例系统性能分析的结果,BDD 模型的大小用非叶子节点的数量表示。

表 3 实例系统性能分析结果

Tab. 3 Performance analysis results of example system

L_i	k -to- l -out-of- n	BDD 大小	Φ_L
L_L	0-to-1-out-of-6	10	0.1951
L_M	2-to-4-out-of-6	16	0.5407
L_H	5-to-6-out-of-6	10	0.2642

3.3 性能比较

为了证明自顶向下 BDD 生成算法的有效性和正确性,将其与传统 BDD 生成算法作比较。传统 BDD 生成算法一般采用自底向上算法^[11-15],首先创建与系统性能状态对应的节点状态布尔表达式,然后根据所给的逻辑操作自底向上组合生成 BDD。为了防止相同表达式的重复构建,使用哈希表 computation-table 将一个布尔表达式映射到一个 BDD 节点。只有当 computation-table 中没有相应的布尔表达式记录时,才执行表达式 BDD 构建。但是,尽管使用 computation-table,在自底向上的组合过程中,仍会产生大量 BDD 中间节点。

为了比较性能,使用 2-to-5-out-of- n 模型作为基准系统, $n = 10, 11, 12, 13, 14, 15$, 节点的故障模型为负指数分布模型。所有计算都在一台 PC 上完成,其 CPU 为 Intel Core i7-2600 3.40 GHz,内存为 2.00 GB,运行 Windows 7 操作系统。性能比较结果如表 4 所示。注意,BDD 的构造时间以毫秒(ms)为单位,BDD 的大小用非叶子节点的数量表示。实验数据表明,自顶向下 BDD 生成算法比传统自底向上 BDD 生成算法更高效,而且随着系统规模的增加,更能体现这样的优势。自顶向下 BDD 生成算法使用特殊的晶格结构,不会产生任何冗余的中间节点,由表 4 中相同的“中间 BDD 节点数”和“最终 BDD 节点数”得以证明。

为了进一步说明自顶向下 BDD 生成算法的性能优势,使用一个更大的 k -to- l -out-of- n 模型作为基准系统, $n = 50, 60, 70, 80, 90, 100$, 每个节点的状态分布随机生成;考虑三组不同的 (k, l) 组合。

表 5 中 BDD 分析时间包括构建 BDD 的时间和评估 BDD 的时间。数据表明,自顶向下 BDD 生成算法可以快速分析基于 k -to- l -out-of- n 结构的大型计算系统,并且 BDD 大小和分析时间的增长较为缓慢。特别是 BDD 的规模复杂度接近 $n^2/3$ 的情况,由表 5 数据可知,随 n 值的增大,BDD 大小增长缓慢。

表4 两种 BDD 生成算法性能比较结果(2-to-5-out-of- n)Tab. 4 Performance comparison results of two BDD generation algorithms (2-to-5-out-of- n)

n	传统 BDD 生成算法			本文 BDD 生成算法		
	中间 BDD 节点数	最终 BDD 节点数	处理时间/ms	中间 BDD 节点数	最终 BDD 节点数	处理时间/ms
10	1684	38	0.46	38	38	<0.01
11	3259	44	0.65	44	44	<0.01
12	6014	50	1.24	50	50	<0.01
13	10620	56	2.27	56	56	<0.01
14	18024	62	4.09	62	62	<0.01
15	29526	68	7.13	68	68	<0.01

表5 BDD 大小与分析时间对比

Tab. 5 Comparison of BDD size and analysis time

(k, l)	$n = 50$		$n = 60$		$n = 70$		$n = 80$		$n = 90$		$n = 100$	
	BDD 大小	分析时间/ms	BDD 大小	分析时间/ms	BDD 大小	分析时间/ms	BDD 大小	分析时间/ms	BDD 大小	分析时间/ms	BDD 大小	分析时间/ms
$(n/5, n/2)$	810	0.0617	1158	0.0859	1568	0.1155	2040	0.1461	2574	0.1848	3170	0.2227
$(n/10, n/2)$	755	0.0568	1080	0.0798	1463	0.1063	1904	0.1373	2403	0.1702	2960	0.2103
$(n/10, n/5)$	470	0.0327	666	0.0456	896	0.0618	1160	0.0785	1458	0.0994	1790	0.1221
$n^2/3$	833	—	1200	—	1633	—	2133	—	2700	—	3333	—

4 结语

本文提出了一种新的基于 BDD 的集群计算系统性能分析方法,该方法能够有效处理节点计算能力相同的大型集群计算系统性能分析问题。采用自顶向下 BDD 生成算法可以避免产生大量中间操作,充分利用特殊的 k -to- l -out-of- n 结构,相比传统的自底向上生成算法更为高效。

处理节点不同且计算能力也不同的集群系统性能分析问题时,适合使用状态空间法,如 Markov 模型。但是,对中型或大型集群计算系统进行性能分析时,这些方法会出现“组合爆炸”问题,而且受限于可积故障时间分布^[16-17]。最近,针对部件不同且相互独立的多状态 k -out-of- n 系统,我们在文献[18]中提出了一种基于决策图的方法。决策图方法与 Markov 方法不同,Markov 方法使用一个显式状态转换图模型,而决策图使用一个组合模型,它用不相交路径表示不相交节点的状态组合,这些状态组合使整个系统处于一个特定性能状态。实例结果表明,决策图方法可以有效地处理大量实际问题,但这个方法只适用于 k -out-of- n 模型,并不适用本文中的 k -to- l -out-of- n 模型。要将基于决策图的方法应用于 k -to- l -out-of- n 模型中,需要实现新的模型生成算法、新的简化规则以及新的排序策略。

虽然在 k -to- l -out-of- n 系统的建模和分析方面已经有了一些研究成果,但是本文提出的新方法在大型集群计算系统性能分析方面仍具有一定意义,特别是在现代数据中心和云计算系统规模快速增长的情况下。以后我们还将继续扩展基于 BDD 的性能分析方法,将其应用于具有多状态节点的大型集群计算系统。

参考文献 (References)

- [1] FOSTER I, KESSELMAN C, TUECKE S. The anatomy of the grid: enabling scalable virtual organizations [J]. The International Journal of High Performance Computing Applications: Information for Contributors, 2001, 15(3): 200–222.
- [2] VOUK M A. Cloud computing — issues, research and implementa-

tions [J]. Journal of Computing and Information Technology, 2008, 16(4): 235–246.

- [3] MO Y. Variable ordering to improve BDD analysis of phased-mission systems with multimode failures [J]. IEEE Transactions on Reliability, 2009, 58(1): 53–57.
- [4] HARISH P, NARAYANAN P J. Accelerating large graph algorithms on the GPU using CUDA [C]// HiPC 2007: Proceedings of the 14th International Conference on High Performance Computing, LNCS 4873. Berlin: Springer-Verlag, 2007: 197–208.
- [5] SHIVA S G. Advanced Computer Architectures [M]. Boca Raton, FL: CRC Press, 2005: 312–322.
- [6] MO Y, XING L, ZHONG F, et al. Choosing a heuristic and root node for edge ordering in BDD-based network reliability analysis [J]. Reliability Engineering and System Safety, 2015, 131: 83–93.
- [7] LEVITIN G, XING L. Reliability and performance of multi-state systems with propagated failures having selective effect [J]. Reliability Engineering and System Safety, 2010, 95(6): 655–661.
- [8] ANDREWS J D. To not or not to not [C]// ISSC-2000: Proceedings of the 18th International System Safety Conference. [S.l.]: System Safety Society, 2000: 267–275.
- [9] RAUZY A, DUTUIT Y. Exact and truncated computations of prime implicants of coherent and non-coherent fault trees within aralia [J]. Reliability Engineering and System Safety, 1997, 58(2): 127–144.
- [10] MO Y, XING L, ZHONG F, et al. Reliability evaluation of network systems with dependent propagated failures using decision diagrams [J]. IEEE Transactions on Dependable and Secure Computing, 2015, 13(6): 672–683.
- [11] MEYER J F. Model-based evaluation of system resilience [C]// DSN-W 2013: Proceedings of the 2013 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop. Washington, DC: IEEE Computer Society, 2013: 1–7.



- EDBT '09: Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology. New York: ACM, 2009: 72–83.
- [10] NERGIZ M E, ATZORI M, SAYGIN Y, et al. Towards trajectory anonymization: a generalization-based approach [C]// SPRINGL '08: Proceedings of the SIGSPATIAL ACM GIS 2008 International Workshop on Security and Privacy in GIS and LBS. New York: ACM, 2008: 52–61.
- [11] TERROVITIS M, MAMOULIS N. Privacy preservation in the publication of trajectories [C]// MDM '08: Proceedings of the 9th International Conference on Mobile Data Management. Piscataway, NJ: IEEE, 2008: 65–72.
- [12] 袁冠, 夏士雄, 张磊, 等. 基于结构相似度的轨迹聚类算法[J]. 通信学报, 2011, 32(9): 103–110. (YUAN G, XIA S X, ZHANG L, et al. Trajectory clustering algorithm based on structural similarity [J]. Journal on Communications, 2011, 32(9): 103–110.)
- [13] MANO K, MINAMI K, MARUYAMA H. Pseudonym exchange for privacy-preserving publishing of trajectory data set [C]// GCCE 2014: Proceedings of the 3rd IEEE Global Conference on Consumer Electronics. Piscataway, NJ: IEEE, 2014: 691–695.
- [14] HUO Z, MENG X, HU H, et al. You can walk alone: trajectory privacy-preserving through significant stays protection [C]// Proceedings of the 17th International Conference on Database Systems for Advanced Applications, LNCS 7238. Berlin: Springer-Verlag, 2012: 351–366.
- [15] MOHAMMED N, FUNG B C M, DEBBABI M. Preserving privacy and utility in RFID data publishing, Technical Report 6850 [R]. Montreal, Canada: Concordia University, 2010.
- [16] CHEN R, FUNG B C M, MOHAMMED N, et al. Privacy-preserving trajectory data publishing by local suppression [J]. Information Sciences, 2013, 231(1): 83–97.
- [17] AL-HUSSAENI K, FUNG B C M, CHEUNG W K. Privacy-preserving trajectory stream publishing [J]. Data & Knowledge Engineering, 2014, 94(Part A): 89–109.
- [18] GHASEMZADEH M, FUNG B C M, CHEN R, et al. Anonymizing trajectory data for passenger flow analysis [J]. Transportation Research Part C: Emerging Technologies, 2014, 39(2): 63–79.
- [19] KOMISHANI E G, ABADI M. A generalization-based approach for personalized privacy preservation in trajectory data publishing [C]// IST '12: Proceedings of the 2012 IEEE Sixth International Symposium on Telecommunications. Piscataway, NJ: IEEE, 2012: 1129–1135.
- [20] MOHAMMED N, FUNG B, HUNG P C K, et al. Anonymizing healthcare data: a case study on the blood transfusion service [C]// KDD '09: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM, 2009: 1285–1294.
- [21] FUNG B C M, WANG K, YU P S. Anonymizing classification data for privacy preservation [J]. IEEE Transactions on Knowledge and Data Engineering, 2007, 19(5): 711–725.
- [22] DeWITT D J, LeFEVRE K, RAMAKRISHNAN R. Mondrian multidimensional k -anonymity [C]// SIGMOD '06: Proceedings of the 22nd IEEE International Conference on Data Engineering. Washington, DC: IEEE Computer Society, 2006: 25.
- [23] XIAO X, TAO Y. Personalized privacy preservation [C] // Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data. New York: ACM, 2006: 229–240.
- [24] BURDICK D, CALIMLIM M, GEHRKE J. MAFLA: a maximal frequent itemset algorithm for transactional databases [C]// Proceedings of the 17th IEEE International Conference on Data Engineering. Piscataway, NJ: IEEE, 2001: 443–452.

This work is partially supported by the National Natural Science Foundation of China (61370050, 61672039), the Natural Science Foundation of Anhui Province (1508085QF134).

DENG Jingsong, born in 1991, M. S. candidate. His research interests include information security, privacy preserving.

LUO Yonglong, born in 1972, Ph. D., professor. His research interests include spatial data processing, information security, privacy preserving.

YU Qingying, born in 1980, Ph. D. candidate, lecturer. Her research interests include spatial data processing, information security.

CHEN Fulong, born in 1978, Ph. D., professor. His research interests include embedded computing and pervasive computing, cyber-physical system, high-performance computer architecture.

(上接第467页)

- [12] RAUZY A B, GAUTHIER J, LEDUC X. Assessment of large automatically generated fault trees by means of binary decision diagrams [J]. Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability, 2007, 221(2): 95–105.
- [13] POCKY M, MALASS E, WALTER M. Combining different binary decision diagram techniques for solving models with multiple failure states [J]. Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability, 2011, 225(1): 18–27.
- [14] MO Y, ZHONG F, LIU H, et al. Efficient ordering heuristics in binary decision diagram-based fault tree analysis [J]. Quality and Reliability Engineering International, 2013, 29(3): 307–315.
- [15] MEYER J F. Defining and evaluating resilience: a performability perspective [C/OL]// PMCCS-9: Proceedings of the 9th International Workshop on the Performability Modeling of Computer and Communication Systems. [S. l.]: Mendeley, 2009 [2016-03-06]. http://ftp.eecs.umich.edu/people/jfm/PMCCS-9_Slides.pdf.
- [16] REIBMAN A, TRIVEDI K. Numerical transient analysis of Markov models [J]. Computers and Operations Research, 1998, 15(1): 19–36.
- [17] TRIVEDI K S. Probability and Statistics with Reliability, Queuing and Computer Science Applications [M]. 2nd ed. Upper Saddle River, NJ: Prentice Hall, 2001: 172–179.
- [18] MO Y, XING L, AMARI S V, et al. Efficient analysis of multi-state k -out-of- n systems [J]. Reliability Engineering and System Safety, 2015, 133: 95–105.

This work is partially supported by the National Natural Science Foundation of China (61272130, 61572442), the Public Projects of Zhejiang Province (2015C33085).

XU Meiling, born in 1992, M. S. candidate. Her research interests include decision diagram analysis.

QIAO Ying, born in 1992, M. S. candidate. Her research interests include decision diagram analysis.

MO Yuchang, born in 1980, Ph. D., associate professor. His research interests include highly reliable computing.

ZHONG Farong, born in 1963, Ph. D., professor. His research interests include parallel computing.