抽象事件的完备逻辑时钟*

董 宏 孙永强

(上海交通大学计算机科学与工程系 上海 200030)

摘要 调试分布式应用系统要比调试顺序程序困难得多,原因之一是分布式应用系统要比顺序程序复杂得多.为了处理分布式应用系统的复杂性,提出了对分布式应用系统的事件进行抽象的调试方法,这种方法让用户从不同的层次观察分布式应用系统的行为.在对分布式应用系统进行调试和抽象时,抽象事件(事件集)之间的先于关系起着重要作用,而逻辑时钟用来确定事件间的先于关系.文章给出了一种计算抽象事件的完备逻辑时钟的方法,这种方法比以往的方法所占存储量小,而且计算速度快.另外,文中还给出了算法的正确性证明.

关键词 分布式系统,调试,抽象,完备逻辑时钟.

中图法分类号 TP311

分布式应用系统(以后简称分布式系统)是由一些各自独立的顺序进程组成,这些进程相互协作共同完成一个任务,通过消息交换完成通信与同步.分布式系统中某个进程某时的状态是该进程在此时的所有变量值以及与之有关的环境信息(寄存器值、消息等)的总和,系统状态是所有进程状态的和.事件是给定时刻引起状态改变的实体(给变量赋值、创建新进程等).

调试分布式系统要比调试顺序程序困难得多[1],原因之一是分布式系统要比顺序程序复杂得多.Bates[2],Lumpp和Casavant[3]分别提出了对事件进行抽象的方法处理复杂系统,使用户从不同的层次观察分布式系统的行为.Zernik用简单抽象的方法实现了一个调试系统[4].事件的抽象就是把一组事件组合成一个抽象事件,把一组较低层的抽象事件组合成一个较高层次的抽象事件.

先于(happened before)关系反映事件间的因果性,在调试与抽象过程中起着重要的作用,而逻辑时钟用来快速确定先于关系(替代搜索图的方法)[5].单个事件的逻辑时钟已广泛应用,但抽象事件的逻辑时钟还在讨论[5,6]之中,本文给出一种所占存储量小且计算速度快的抽象事件(事件集)逻辑时钟的计算方法.

1 事件集间的关系及其抽象

1.1 事件间的关系

$->$关系是一事件集上满足下列条件的关系:

(1) 如果事件 a 和 b 在同一进程 P 中,a 在 b 之前发生并且不存在发生在 a 之后 b 之前的 P 中的事件,则 $a->b$;

(2) 如果 a 是一进程的发送事件,b 是另一进程的接收由 a 发送的消息的接收事件,则 $a->b$;

(3) 如果 a 和 b 是同步事件,那么,$a->b \wedge b->a$.

先于关系$\rightarrow$是关系$->$的自反传递闭包.两个事件 a 和 b 当且仅当它们是同一事件时就称它们相等.如果 $d->c$,就称 d 为 c 的直接前驱或 d 直接先于 c,c 为 d 的直接后继或 c 直接后继于 d.

* 本文研究得到国家自然科学基金资助.作者董宏,1959年生,博士,主要研究领域为分布式系统,计算机网络.孙永强,1931年生,教授,博士生导师,主要研究领域为计算机软件.

本文通讯联系人:董宏,上海 200030,上海交通大学计算机科学与工程系

本文1998-03-11收到原稿,1998-11-23收到修改稿

1.2 事件集间的关系

假定 A 为一个分布式系统的事件集，A 的输入事件 $i^A \in A \wedge (\exists\, b \notin A: b->i^A)$，$A$ 的输出事件 $o^A \in A \wedge (\exists\, b \notin A: o^A->b)$. I^A 是 A 的所有输入事件的集合，O^A 是 A 的所有输出事件的集合.

事件集间的先于关系也用→表示. 对于两个事件集 A 和 B，$A \rightarrow B$ 当且仅当存在一个事件 $a \in A$ 且存在一个事件 $b \in B: a \rightarrow b$，$A->B$(A 为 B 的直接前驱、B 为 A 的直接后继)当且仅当存在一个输出事件 $o^A \in A$ 且存在一个输入事件 $i^B \in B: o^A->i^B$，$A \parallel B$(A 和 B 是并行的)，当且仅当 $\neg(A \rightarrow B) \wedge \neg(B \rightarrow A)$.

对于任意两个事件集 A 和 B，准因果关系 $\triangleright$ 定义如下：

(1) 如果 $\exists\, a \in A, b \in B: a \rightarrow b$，则 $A \triangleright B$；

(2) 如果 $A > B \wedge B \triangleright C$，那么，$A \triangleright C$.

$A \parallel\!\!\parallel B$($A$ 和 B 是准并行的)，当且仅当 $\neg(A \triangleright B) \wedge \neg(B \triangleright A)$.

假定分布式系统是由事件构成的集合 PE，对分布式系统进行抽象就是构造一个 $PE^1 = \{E_1, \ldots, E_m\}$，满足 $PE = E_1 \cup \ldots \cup E_m$，$E_i \cap E_j = \varnothing (1 \leqslant i, j \leqslant m, i \neq j)$. 称 PE^1 为 PE 的一次抽象(或抽象)，称 $E_i (1 \leqslant i \leqslant m)$ 为抽象事件. 如果对 PE^1 再进行抽象，构造出 PE^2(PE^1 的抽象)，那么，这就是 PE 的更高层次的(二次)抽象. 同样地，对 PE^2 也可再进行抽象，一直继续下去，直到 $PE^k = \{PE^{k-1}\}$.

将一个分布式系统 PE 抽象成 PE^n，如果对于任意两个事件集 $A, B \in PE^n$，都有 $A \triangleright B \Rightarrow A \rightarrow B$，那么这个抽象就称为正确抽象. 只有正确的抽象才能反映系统的特征[3,7]. 给用户以不同层次系统面貌的核心就是自动分析分布式系统运行时所产生的事件流，并产生各层抽象，同时检查这些抽象是否是正确的抽象. James 提出了按照一定的结构进行抽象的方法[3]，Cheung 等人对抽象结构进行了研究[8]，目前已有的可实用的正确抽象结构为完全优先抽象、合约抽象和协议抽象.

2 事件的逻辑时钟(events' logical time)

分布式系统 PE 的逻辑时钟是赋给每个事件 $e \in PE$ 的值 $T(e)$ 的集合 PT，通过所有的值 $T(e) \in PT$ 来决定事件间的先于关系. 逻辑时钟由两部分组成：(1) 逻辑时钟计算方法的描述，即如何给事件 $e \in PE$ 赋值 $T(e)$；(2) 逻辑时钟的判断方法，即如何用逻辑时钟值 $T(e) \in PT$ 决定优先关系.

对于分布式系统 PE 的逻辑时钟 PT，如果 $\forall\, e_1, e_2 \in PE, e_1 \rightarrow e_2 \Rightarrow T(e_1) \leqslant T(e_2)$，但 $\neg(T(e_1) \leqslant T(e_2) \Rightarrow e_1 \rightarrow e_2)$，就称 PT 为非完备的(incompleted)；如果 $e_1 \rightarrow e_2 \Rightarrow T(e_1) \leqslant T(e_2)$，就称 PT 为完备的(completed).

在后面的讨论中我们规定，求两个 n 维向量 A 与 B 的最大值，其结果为各对对应元素的最大值所组成的 n 维向量，即假定 $C = \max\{A, B\}$，则 $C[i] = \max\{A[i], B[i]\}, i = 1, \ldots, n$.

3 抽象事件的逻辑时钟

完全优先抽象的完备逻辑时钟在文献[8]中已给出，把抽象事件的逻辑时钟建立在各个事件的逻辑时钟的基础之上，给每个抽象事件赋予一向量，但它只用于完全优先抽象，不能用于适应范围较大的合约抽象和协议抽象.

以往可用于任何正确抽象的完备逻辑时钟是给每个抽象事件 E 赋予两个向量集合(所有输入事件的向量时钟的集合 E^{TI} 和所有输出事件的向量时钟的集合 E^{TO})[5]. 这种抽象事件的逻辑时钟有两个致命的缺点：(1) 占用内存太大，每一个抽象事件要保存所有输入、输出事件的向量时钟，如果系统有 N 个进程，最大要保存 $2N$ 个 N 维向量，即两个 $N \times N$ 的矩阵；(2) 判断任意两个抽象事件 A, B 的优先关系时所花费的时间多，要逐一比较 A^{TO} 和 B^{TI} 的所有向量，如果比较不成功就要逐一用 B^{TO} 的所有向量和 A^{TI} 的所有向量比较，其算法的复杂度为 $O(N^2)$. 本文给出一种在异步系统下可用于任何正确抽象的完备逻辑时钟，每一个抽象事件只含一个向量，减少了所需保存的数据量，由于判断任意两抽象事件的优先关系只比较两个向量，也减少了运算量，算法的复杂度仅为 $O(N)$.

下面通过对抽象事件的历史集的分析，给出在异步分布式系统下抽象的完备逻辑时钟算法以及算法的正确

性证明.

3.1 抽象事件的历史集

定义 3.1(事件的自然次序). 假定 $P_1,\ldots,P_N$ 为分布式系统的进程,进程 $P_i(1\leqslant i\leqslant N)$中的事件按其出现次序记为 $e_{i1},e_{i2},\ldots,PE_i$ 为 P_i 中事件的集合,即 $PE_i=\{e_{i1},e_{i2},\ldots\}$. $PE=PE_1\cup PE_2\cup\ldots\cup PE_N$ 为系统所有事件的集合,称 PE_i 中给事件赋予的出现次序号 1,2,... 为事件的自然序号.

定义 3.2(自然优先事件与自然优先事件集). 对于任意一个抽象事件 $E\subset PE$ 和事件 $a\in PE_i\wedge a\notin E$,如果 $\exists\, b\in E\wedge b\in PE_i$ 且 b 的自然序号大于 a 的自然序号,则称 a 为 E 的自然优先事件.包含 a 的抽象事件 A(即假定 $a\in A$),则 A 称为 E 的自然优先集;如果 a 的序号加 1 等于 b 的序号,则称 a 为 E 的直接自然优先事件,相应的抽象事件 A 为 E 的直接自然优先集.

定义 3.3(抽象事件的历史集). 任意抽象事件 $A\subset PE$ 的历史集 $H(A)=\{b\mid b\in B\wedge B\triangleright A\}\cup A$. $H(A)$在 PE_i 上的投影 $H(A)[i]=H(A)\cap PE_i$. 由于 $PE=PE_1\cup PE_2\cup\ldots\cup PE_N$ 为全集,因此,$H(A)=H(A)[1]\cup H(A)[2]\cup\ldots\cup H(A)[N]$.

引理 3.1. 对于一个分布式系统 PE 的任意一抽象 PE^1 的任意两个抽象事件 $A,B\in PE^1$,如果 $A\neq B\wedge\exists\, b\in B\wedge b\in H(A)$,则(1) $B\subset H(A)$;(2) $B\triangleright A$.

证明:由抽象定义可知,系统中的任意一个事件只能属于一个抽象事件,通过定义 3.3 可直接得出结论.

引理 3.2. 对于一个分布式系统 PE 的任意一个抽象 PE^1 的任意两个抽象事件 $A,B\in PE^1$ 且 $A\neq B$,历史集与准因果关系 $\triangleright$ 之间有:

(1) $A\triangleright B$ Iff $A\subset H(B)$;

(2) $A\parallel B$ Iff $A\not\subset H(B)\wedge B\not\subset H(A)$.

证明:直接由定义 3.3 得出. 从定义 3.3 可直接得出异步分布式系统的任意抽象事件 A 的历史集 $H(A)$的算法 1.

算法 1. 历史集算法

(1) $H(A)=A$,转(2);

(2) 如果 A 无直接自然优先集,则转(3),否则,将它的所有直接自然优先集的历史集的事件加入到 $H(A)$中,即假定 A 的直接自然优先集为 $E_1,\ldots,E_m$,则 $H(A)=H(E_1)\cup\ldots\cup H(E_m)\cup H(A)$,转(3);

(3) 如果 A 无接收事件,则结束,否则,将所有发送给它消息的事件集的历史集的事件加入到 $H(A)$中,即假定 A 接收由 $E_1,\ldots,E_i$ 发来的消息,则 $H(A)=H(E_1)\cup\ldots\cup H(E_i)\cup H(A)$,结束.

引理 3.3. 对于任意进程 $P_i(i=1,\ldots,N)$的事件集 $PE_i=\{e_{i1},\ldots,e_{im}\}$,如果 $e_{ij}\in H(A)[i]$,则 $e_{i1},\ldots,e_{i(j-1)}\in H(A)[i]$.

证明:假定 $e_{ij}\in E$,则由引理 3.1 可知,$E\triangleright A$;由于 $e_{i1}\rightarrow e_{ij},\ldots,e_{i(j-1)}\rightarrow e_{ij}$,假定 $e_{i1}\in E_1,\ldots,e_{i(j-1)}\in E_{j-1}$,则 $E_1\triangleright E\triangleright A,\ldots,E_{j-1}\triangleright E\triangleright A$,按定义 3.3,$E_1,\ldots,E_{j-1}\subset H(A)$,因此,$e_{i1},\ldots,e_{i(j-1)}\in H(A)$;由 $H(A)[i]$的定义可知,结论成立. □

定义 3.4. 对于由 N 个进程组成的分布式系统 PE,任意抽象事件 $A\in PE$ 的历史集向量 $T(A)$是 N 维向量,$T(A)[i]=|H(A)[i]|(i=1,\ldots,N)$.

引理 3.3 说明,任意 $H(A)[i](1\leqslant i\leqslant N)$都可由它的元素的最大下标刻画,因此,$H(A)$可唯一地用 N 维向量 $T(A)$表示.

引理 3.4. 令 $A,B\subset PE$ 为抽象事件,$H(A)$,$H(B)$分别为它们的历史集,$T(A)$,$T(B)$分别为相应的历史向量.那么,$T(H(A)\cup H(B))=\max\{T(A),T(B)\}$.

证明:由定义 3.3 和定义 3.4,通过引理 3.3 可直接推出.

3.2 完备逻辑时钟

通过引理 3.4,可从算法 1 得出在异步分布式系统的抽象事件的历史向量的算法,该算法就是本文提出的求抽象事件的逻辑时钟的算法.

定义 3.5(抽象事件的逻辑时钟). 假定 $P_1,\ldots,P_N$ 为一个异步分布式系统的进程,该系统的抽象事件的逻辑时钟(历史集向量)按下面的算法 2 计算.

算法 2.

(1) 置向量计数器 V 的初值为 0:$V[k]=0(k=1,\ldots,N)$;

(2) 选取一个未处理的事件集 E,E 应没有直接自然优先集,或虽有直接自然优先集但这些直接自然优先集都已处理过,如果选不出这样的 E,则结束,否则,转(3);

(3) 处理 E:对于所有进程 $P_i(i=1,\ldots,N)$,如果 E 中有 m 个 P_i 的事件,则 $V[i]=V[i]+m$,$T(E)[i]=V[i]$;如果 E 中没有 P_i 的事件,则 $T(E)[i]=0$,转(4).

(4) 若 E 没有直接自然优先集,则转(5);否则,更改 $T(E)$ 为 $T(E)$ 及所有 E 的直接自然优先集的时钟的最大值,即假定 E 有 l 个直接自然优先集 $E_1,\ldots,E_l$,则 $T(E)=\max(T(E),T(E_1),\ldots,T(E_l))$,转(5);

(5) 若 E 没有接收消息事件,则转(6);否则,更改 $T(E)$ 为 $T(E)$ 及所有已附在 E 的接收消息事件上的向量的最大值,即假定 E 有 s 个接收消息事件 $r_1,\ldots,r_s$ 上已附有的向量 $T(E_1),\ldots,T(E_s)$,则 $T(E)=\max(T(E),T(E_1),\ldots,T(E_s))$,转(6);

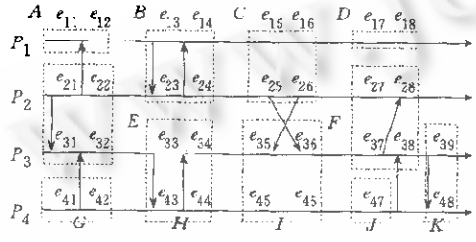

图1 四进程异步系统的正确抽象时序图

(6) 若 E 没有发送消息事件,则转(2);否则,假定 E 有 t 个发送消息事件 $s_1,\ldots,s_t$,相应的接收消息事件为 $r_1,\ldots,r_t$,则对每个 $r_i(i=1,\ldots,t)$ 作如下处理后转(2).

如果包含 r_i 的抽象事件 E_r 已处理过,则 $T(E_r)=\max\{T(E_r),T(E)\}$,否则,将 $T(E)$ 附在 r_i 上.

例如,图 1 是由 4 个进程组成的异步分布式系统的一个正确抽象时序图,虚线框表示抽象事件,各个抽象事件的逻辑时钟为:

$T(G)=[0,0,0,2]$, $T(E)=\max\{[0,2,2,0],T(H)\}=[0,2,2,2]$,

$T(A)=\max\{[2,0,0,0],T(E)\}=[2,2,2,2]$, $T(B)=\max\{[4,4,0,0],T(E),T(A)\}=[4,4,2,2]$,

$T(H)=\max\{[0,0,4,4],T(E),T(G)\}=[0,2,4,4]$, $T(C)=\max\{[6,6,0,0],T(B),T(I)\}=[6,6,6,6]$,

$T(I)=\max\{[0,0,6,6],T(C),T(H)\}=[6,6,6,6]$, $T(D)=\max\{[8,0,0,0],T(C)\}=[8,6,6,6]$,

$T(J)=\max\{[0,0,0,7],T[I]\}=[6,6,6,7]$,

$T(F)=\max\{[0,8,8,0],T(C),T(J),T(I)\}=[6,8,8,7]$,

$T(K)=\max\{[0,0,9,8],T(F),T(J)\}=[6,8,9,8]$.

定义 3.6. 令 u,v 为 n 维向量的逻辑时钟.

(1) $u\leqslant v$ Iff $u[k]\leqslant v[k](k=1,\ldots,m)$;

(2) $u<v$ Iff $u\leqslant v\wedge u\neq v$;

(3) $u\parallel v$ Iff $\neg(u<v)\wedge\neg(v<u)$.

下面证明定义 3.5 和定义 3.6 构成了异步分布式系统抽象的完备逻辑时钟.

引理 3.5. 对任意一个由 N 个进程组成的异步分布式系统 PE 抽象的任意两个抽象事件 $A,B\subset PE$ 有:

(1) $A\rhd B$ Iff $T(A)\leqslant T(B)$;

(2) $A\|\!|B$ Iff $T(A)\parallel T(B)$.

证明:(1) 假定 $A\rhd B$,按引理 3.2,有 $A\subset H(B)$;按定义 3.3 和$\rhd$的传递性,有 $H(A)\subseteq H(B)$,因此,$H(A)[k]\subseteq H(B)[k](k=1,\ldots,N)$,从而有 $T(A)[k]=|H(A)[k]|\leqslant|H(B)[k]|=T(B)[k](k=1,\ldots,N)$,即 $T(A)\leqslant T(B)$.

反之,假定 $T(A)\leqslant T(B)$,由引理 3.3 可知,$H(A)[k]\subseteq H(B)[k](k=1,\ldots,N)$,即 $H(A)\subseteq H(B)$;由于 $A\subseteq H(A)\subseteq H(B)$,由引理 3.1 得,$A\rhd B$.

(2) 由(1)和$\parallel$以及$\|\!|$的定义直接得出. □

定理 3.1. 如果对于任意异步分布式系统 PE 的抽象 PE^1 是正确的,那么,按定义 3.5 和定义 3.6 所构造的

PE^1 中的抽象事件的逻辑时钟是完备的.

证明:对于任意 $A,B\in PE^1$,(1) 由正确抽象的定义可知,如果 $A\triangleright B$,则$\exists\ a\in A,\exists\ b\in B$:$a\rightarrow b$,即 $A\rightarrow B$;(2) 由$\triangleright$的定义可知,$A\rightarrow B\Rightarrow A>B$;综合(1)和(2),由引理 3.5 可直接得出结论. □

4 结 论

定义 3.5 和定义 3.6 给出了在异步分布式系统下抽象事件的完备逻辑时钟,每个抽象事件的时钟只用一个向量,比以往的基于事件向量的完备的逻辑时钟减少了对内存的需求和计算量,算法复杂度由 $O(N^2)$降为 $O(N)$. 算法 1 和算法 2 都是相对于异步系统给出的,但从所有引理和定理的证明不难看出,它们都适合同步系统,如果在算法 2 中对同步事件作相应的处理,即把同步事件看成发送事件与接收事件的组合,则同样也适合同步系统.

参考文献

1 Cheng Wing-hong, Black J P, Manning E. A frame work for distributed debugging. IEEE Software, 1990,7(1):106~115

2 Peter Butes. Distributed debugging tools for heterogeneous distributed systems. In: Davis C T ed. Proceedings of the 8th International Conference on Distributed Computing System. San Jose, CA: IEEE Computer Society Press, 1988. 308~315

3 Lumpp J E, Jr Casavant T L, Siegel H J *et al*. Specification and identification of events for debugging and performance monitoring of distributed mutiprocessor systems. In: Gerard Le Lann, Walt Koheler eds. Proceedings of the 10th International Conference on Distributed Computing Systems. Paris: IEEE Computer Society Press, 1990. 476~483

4 Dror Zernik, Marc Snir, Daliamaki. Using visualization tools to understand concurrency. IEEE Software, 1992,9(3):87~92

5 Reinhard Schwarz, Friedemann Mattern. Detecting casual relationships in distributed computations: in search of the holy grail. Distributed Computing, 1994,7(3):149~174

6 Suresh K. Damodaran-Kamal, Brown J S. Toward heterogeneous distributed debugging. Technical Report, No. LAUR-97-906, Los Alamos National Laboratory, Los Alamos, NM-87545, 1997

7 Kunz T. Event abstraction: some definition and theorems. Technical Report, TI-Z/93, Darmstadt: Technical University, 1996

8 Cheung Wing-hong. Process and event abstraction for debugging distributed programs. Technical Report T-189, Computer Communications Network Group, Ontario: University of Waterloo, 1992

Abstraction Events' Complete Logical-time

DONG Hong　SUN Yong-qiang

(*Department of Computer Science and Engineering　Shanghai Jiaotong University　Shanghai*　200030)

Abstract　Debugging the distributed applications is very difficult. One of the reasons why this problem is made is that the distributed applications are inherently more complex than sequential ones. To solve this problem, the method of abstracting events is proposed, which lets the users grasp the various aspects of behaviors of a distributed application. Abstract event timestamp is used to decide the "happened before" relation among abstract events, which plays a very important role in abstracting and debugging the distributed applications. In this paper, a new complete algorithm of timestamping abstract event is proposed, which needs less storage and costs less time than others. Proofs of the correctness of the algorithms are also given in this paper.

Key words　Distributed system, debugging, abstraction, complete logic-timestamp.