计算机科学1995 Vol. 22 №. 2

分布式系统中合作进程及其同步关系

赵 宏
（东北大学软件中心 沈阳110006）

摘 要 The synchronous cooperative relations between distributed processes and their functions to support distributed applications are studied and analysed in this paper. The mechanisms and protocols for supporting synchronization and cooperation are also introduced. A new layer between distributed applications and mechanism provided by common distributed systems is proposed to support the cooperative work in distributed applications that is more and more complicated and is with more and more varieties.

关键词 Distributed system, Group, Cooperation, Synchronous execution Broadcast protocol.

一、引言

在分布式系统中，常涉及分布进程间的各种合作关系。分布进程是分布式系统的基本活动实体，其间的合作表现在多个进程为完成某个任务在时间上、空间上以及次序上发生的直接和间接关系。这种关系可能存在于全过程，也可能只存在一段时间。它的一个重要表现就是分布进程之间的同步执行。

同步执行是分布进程之间在执行上的一种相互制约、相互影响的合作关系的体现。在许多分布活动中存在这种关系。例如：为了加强分布式系统的可靠性和可用性以及实现容错，常常采用复制技术，即把相同的功能和数据复制到物理上相互分离的不同场地上同时实现之。因而，它们经常由不同的进程管理和控制。这样，这些进程为维护这些数据和功能的一致性和整体性，就要相互合作和影响，它们的执行就不是完全异步的。有许多因素要求这种同步执行：如为保证多个进程所完成的动作是原子操作；为保证进程间事件按照某个顺序发生；为保证进程间通信事件的因果关系等等。可以说，进程间的同步执行正是从一个方面表现了这些进程在分布活动中的各自的角色和相互关系。

综上所述，分布合作进程之间的同步执行是必要的，但不是必不可少的，在许多情况下将带来执行效率的降低。如果说它换取了系统的安全性、可靠性、可用性等，那么在许多时候并没有带来任何好处。这方面计算机学者已作了大量工作，以求使分布合作进程之间既保证这种相互制约和影响的关系，又尽量减少开销。这包括新的支持分布合作的机制和模型[1,3,4,9]；减少通信开销的算法和协议[2,10]；支持分布进程间合作的有序性及因果关系的时钟机制[6,7,8]等等。本文将对分布合作进程的同步执行进行较详尽的论述和分析。

二、同步执行及其性质

所谓同步是指进程之间的相互制约关系。当多个进程相互合作共同执行一个任务时，从外部看，是一个整体，进程之间的动作相互关联，与外部的动作相对独立。在一组分布合作进程中，同步执行是指对于不同的外部观察来说，无论何时两个观察者观察到同一个事件，它们一定是在同一个时间单元内进行的。在合作进程的群体内部，事件不可能同时发生。有些本应同时发生的事件，由于通信延时、系统开销和失败等，使它们出现时间上的差异。例如向一个进程群体广播发送消息，该消息不可能同时到达该群体中的各个成员。但是如果同步机制保证这种差异不被外部"感知"，对外部维持一致的形象，即满足要求了。

同步执行主要关系到一组合作进程的行为动作，反映多个合作者之间的相互关系。这种要求同步执行的通信合作具有以下几个特性：

1. 整体性，即对所有群体成员保持的原子性。单个进程的原子性是指一系列动作要么全部完成，要么一个也不做(0-1语义)。对于进程群体，则更严格，

赵宏 教授，博士，现主要进行计算机分布式系统和多媒体技术的研究。

要求所有群体成员要么全部完成，要么任何一个成员都不做任何动作(0-all 语义)。这里有二层意思，一个是动作的指令应该送到每一个应得到它的成员那里(即当前的所有成员)。另一个是每个接到此指令的成员都能完整地执行这一指令。否则整个动作都要作废(undoing)，或者未完成的成员宣布失败。

2. 一致性，即对进程群体内部状态的要求。进程群体一般是为完成某个共同的任务而建立的合作实体，尤其在为实现复制而提高可靠性的系统中更是如此。在这种环境中要求群体成员在可操作时间，总要保持相互一致的状态。这样，在某个或某些成员出现失败时，其它成员也一样可以工作，不至影响所支持的工作。

3. 有序性，是指合作时事件的先后关系。许多合作任务要求每个成员保持全局一致状态，而发出的指令在不同成员执行的先后不同将导致它们的不同状态。因此要保证所有成员得到的消息和指令都是相同顺序提交执行的。如果由于某种原因，使某个或某些成员得到的消息顺序与其它成员不一样，系统必须进行调整，使其在执行时与其它成员保持一致顺序。

4. 因果性，即相互之间有因果关系的事件必须按因果顺序发生，而无此关系的事件，可以按其到达的自然情况去执行。以上有序性一般不要求按指定的顺序进行，只要求所有成员顺序一致，它是一种全序关系。在许多应用中并不要求如此强的条件，而只要求部分活动是有序的即可，一个常用的偏序是因果序。

同步执行是支持分布活动的一个重要问题——多点交互合作的一个有力手段，是使一组分布实体作为整体运行动作的保障。这种运行机制可以为系统某些方面带来一定的好处，也可能使某些性能降低。下面是一些相关的关系：

·同步执行与执行时间和效率：通常，同步执行会使系统的效率降低，使执行时间加长。因为，为了保持合作成员间的一致和同步，需要一些如二段提交那样的算法。同时，由于多个分布成员同时动作，在网络上的传送有快有慢，要同步就要等待最后的完成者，效率必然降低。问题在于怎样尽量减少不必要的降低。

·同步执行与执行空间：不象同步通信，在同步执行中一般要求有较大的空间开销。例如，在要求同步的多段提交算法中，由于要保存接收消息，就必须要有相应的缓冲区；又如在许多系统中由于要保持全局的时钟和状态，除了可能有额外的通信负担外，在正常的通信消息中也可能要携带一些系统信息。

·同步执行与可靠性：一般来讲，同步执行将不同程度地使系统的可靠性有所提高，而且有些系统的同步执行目的就是提高可靠性。例如有的系统采用群体机制和复制机制，并在其间保持一致性，在出现失败时，可以利用复本和其它成员来继续工作。

·同步执行与并发：并发执行是多个活动同时进行，它们之间越独立，相互之间越不影响，则并发程度越高。然而许多并发活动不可能互不影响，必然有这样那样的交互甚至制约关系，而同步执行又是保证和体现这些关系的一种方法。因此，同步执行一般会降低并发程度。这在大部分情况下是必要的。问题仍然在于以什么样的方式、手段、算法来判断哪些必要、哪些不必要，并以相应的机制去支持。

·同步执行与透明性：同步执行通常并不直接影响透明性。透明性主要是提供一种机制使用户不用去关心存取的方法、位置、个数、是否存在失败等情况。只关心其提供的功能。如果不支持透明性，这些工作需用户自己完成，加重了用户的负担。而同步执行则可以提供相应的机制，例如复制。从这个意义上讲，同步执行可以在某种程度上加强透明性。

三、虚拟同步操作

另两个概念是松散同步执行和虚拟同步执行。松散同步执行的意思是一组进程观察到的事件全部为同一个发生顺序，不一定要求在相同的时间单位内发生。可以看出，松散同步执行比同步执行的要求要弱一些，而且通常也很难做到完全同步执行。大部分系统均属于松散同步，尤其是以软件支持的全局时钟的系统。然而，如果系统不是实时的话，那么真正同步执行的进程在松散同步执行时也同样能正确工作。

虚拟同步执行与松散同步执行的关系有些类似于可顺序执行与真正顺序执行的关系。在这种情况下，尽管可能存在某个外部观察者看到的在不同进程中发生了相同事件而具有不同顺序的情况，但这些进程本身没有看到它或检测出它来。

虚拟同步执行的主要思想是试图在不影响系统特性和功能的基础上尽量提高系统效率。由于系统在同步执行中需要同步信息，占用了一定的空间和时间。而虚拟同步则在保证必要的同步执行基础上，一旦有可能和机会，就去减缓这种同步，使它能以较高的效率和并发程度来执行。特别如果我们用一段

提交广播协议来替代二段提交协议，将使情况有很好的改观。虚拟同步执行就是要发现和找出支持这种情况的手段。

在松散同步中，对于每个松散同步执行 E，都存在一个真正同步执行 E'，这两个执行在下面情况下是相等的：令 E_P 是进程 P 对同步执行 E 观察到的事件序列，则每个进程 P 观察的 E'_P 和 E_P 均相同，即 $E'_P = E_P$。对于虚拟同步来说，则 $E'_P \approx E_P$。这里"≈"的意思是两个事件序列可能是真正意义上的相等，也可能不相等，但却是不可区分的，也不必去区分。

四、同步执行的支撑机构

无论是同步执行还是虚拟同步执行，都需要有相应的机构支持。在以往的分布系统中，大部分只涉及两个通信实体，且多采用客户/服务员模式。当今的计算机应用表明，它正在向支持合作化、有序化、组织化的应用方面发展，老的结构和模式已不足以满足需要了。在许多分布合作活动中要求以下特性：①要求多个分布实体同时参与一个合作任务；②多个分布实体中许多同时与使用者有直接交互作用；③这种支撑机构应该能保证多个进程成员对同一任务的不同程度、不同风格、不同形式的合作提供支持，这就要求有新的机制和手段来支持。我们这里主要讨论两个方面的支持：群体机构和全局时钟支持。

群体概念来自于进程群体机制，其基本思想是将一组进程结合成为一个整体并赋予一个共同的标识。进程群体作为合作的一种形式，隐蔽了内部成员的合作和通信，与进程群体打交道，就如同与一个进程打交道一样。至于群体内部怎样协调、同步、通信和容错都与外部无关。群体机构是多个分布实体相互合作的一种抽象，利用它可以解决许多合作问题，如复制、容错、分布、并发等等。[3,4,12]

群体内部除一对一通信之外，主要使用分组广播通信。因此它要求网络环境有分组广播功能(Multicast)。然而如无此功能，也可以用点到点通信和广播通信来实现，但这样可能会降低通信效率，同时也会影响群体外本来不相关的进程和场地。

群体主要是对所操作功能或所管理的数据分布，因此，它基本可以分为四种类型：①仅功能同构：用相同的功能和手段对不同的数据施行操作；②仅数据同构：对相同的数据内容施行不同的操作；③完全同构：对相同的数据结构或内容进行相同的操作，这种结构完全以提高可靠性为目的；④完全异构：数据和功能均不相同，这四种方式中前三种使用比较多，后一种不太常见。

在进行群体通信合作中，要有一些协议来保证群体中的一致性、原子性、有序性等特性，保持全群体一致的状态视图。在一个群体中，每个成员都有自己唯一的标识。可能每个成员都保持一个成员表，还可能存在一个群体首领负责管理和控制群体内部的事务，并执行相应的通信合作协议。下一节将进一步讨论这些协议。

全局时钟是支持分布进程合作的另一个机制。许多同步协议要求它的支持。然而，在分布环境下维持准确实用的全局一致的时钟是很困难的，因为系统的功能实体物理分布于各个场地，它们之间的数据传递只可能通过网络连接，无法实时地在所有场地保持精确的时钟。但是，大部分应用不一定要求这种全局时钟，许多应用只要求有一种事件先后的衡量方法。由于这种衡量主要来自于不同进程和场地之间的某种比较，因此，产生出一种虚拟时钟的概念。[8]

所谓虚拟时钟不是实际意义上的真正系统时钟，它一般是一个单调增长的数值。与真正的时钟相比，它有如下特点：

①时钟的走时不是恒定和恒速的，有时它可能有停顿。这是因为它主要是判定不同场地的事件发生先后关系的。如果在一段时间内没有事件发生，自然就停顿下来。同时，因为系统中的事件是随机发生的，因此，该时钟走时也不是恒速的。

②系统经常会出现不同场地保持的当前时间相互不同的情况。这也是因为它主要是表征不同场地事件先后顺序的。如果某场地在一段时间内没有与其它场地交互，而其它场地仍有相互交互动作，则该场地必然与其它场地具有不同的时间。不过虚拟时钟的保证算法要求一旦某个场地与该场地有交互，则马上修正该场地的虚拟时钟，也就是说，"当我们需要它的时候是可用的"。

③既然虚拟时钟主要用于判定不同场地事件的先后关系，那么，我们完全有理由在真正发生交互时使时钟同步，就可以达到维护全局虚拟时钟的目的。因此，许多虚拟全局时钟的协议和算法并不要求额外的通信次数，而是把时钟同步信息在每个交互消息中携带过去。这样其维护开销就小得多了。这种虚拟全局时钟机制是"寄生的"，它利用了它所支持的应用消息的传送。如果它上面的活动中止了，它本身也不动了。

五、同步执行的通信合作协议

为保证一组进程相互有机合作，或干脆使一组进程形成一个整体，以便提高其功能的可靠性和容错能力，这些成员之间需要有相应的协议来保证。这些合作通信协议主要有三类：①保证原子动作的协议；②保证有序性的协议；③进行容错及整体维护的协议。

原子通信合作协议是进程群体中一个最初步的广播协议，其基本要求就是保证动作在群体所有成员中的0-1语义。这看起来很简单，特别是网络支持的传送提供了可靠的传输协议时更是如此。然而事实上情况要复杂得多。试想如果发送者在只向相关进程中的某些发送了消息后失败了，情况会是怎样呢？问题是接收进程没有办法得知所有需要得到该消息的进程是否都得到了。毫无疑问，解决这个问题一定会增加系统造价，如通信次数、使用的空间等。一种方法是每个接收者在接收一个消息后，再把此消息传送给所有其它成员，从而保证了只有一个成员接到消息，其它成员也一定能接到。但无论如何处理，缓冲和延迟的增加是不可避免的。[10,12]

一个较有名的协议是原子广播协议 ABCAST，它除保证原子性外，还保证了全序，即进程群体中所有成员都按照同一顺序处理所得到的消息。在该协议中使用了时间戳。其思想是：每个成员都有一个有序的未提交消息暂存队列。当某成员接到一个消息后，它把其标识为“未提交”，送入暂存队列，并向发送者发去一个应答，同时附带一个建议时间戳。该时间戳包括本系统的虚拟时间，加上本成员标识。当发送者得到所有接收者的应答后，从中选取一个时间戳最大者，作为最后时间戳，再广播发送给所有接收者。此时，每个接收者都用此最后时间戳在自己的暂存队列中找到对应消息，将其状态改为“可提交”，同时将其时间戳也改成最后时间戳。然后，对暂存队列中的所有消息按照时间戳从小到大重新排序。最后，如果队列首的消息之状态为“可提交”，则将此消息取出进行处理，否则还需要等待。可以看出，这是在一个发送者和多个接收者之间的二段提交协议。它使得群体尽管会出现不同成员接到消息的顺序不相同的情况，但保证它们处理消息的顺序是一致的。

不是在所有情况下都要求这种全序的。有时系统只要求保持偏序即可。一个重要的偏序关系是因果序，因果广播协议 CBCAST 就是一种保证进程合作因果关系的协议。

在合作进程的活动中，事件被定义为一进程向其它进程发送消息及一进程接收到其它进程的消息。这些事件有些具有先后因果关系，即一事件受到前面的某个或者某些事件的影响，这种影响是传递的。这种因果关系被定义为：

①如果 a 和 b 是同一进程中发生的事件，且 a 先于 b 发生，则 b 因果依赖于 a，记作 $a \rightarrow b$；

②如果 a 和 b 发生在不同进程，但 a 是一进程发送一个消息，而 b 是另一进程接收这个消息，则同时有 $a \rightarrow b$。

利用传递性，如果 a 是发生在 P 进程的事件，b 是发生在 Q 进程的事件，则 $a \rightarrow b$ 当且仅当存在一个消息序列 $m_1, m_2, \cdots, m_n$ 及一个进程序列 $P = P_0, P_1, \cdots, P_n$，使进程 P_{i-1} 发送消息 m_i 到进程 P_i 这一事件发生在进程 P_i 发送消息 m_{i+1} 到进程 P_{i+1} 这一事件之前。如果既没有 $a \rightarrow b$ 也没有 $b \rightarrow a$，则说明 a 和 b 两事件不相关，协议无需考虑它们谁先谁后。

CBCAST 协议的基本含义是这样的：无论何时一个进程要发送消息到其它进程，它要在这个消息中携带暂存缓冲区中的消息，而这个缓冲区中暂存的是它以前发出和接收的所有消息。这样如果一个消息到达了某个进程，该消息所因果依赖的所有消息也一定到达了该进程。该协议还需要考虑何时和怎样清理暂存缓冲区中的无用消息等类似的问题，以减少通信开销。还有一些其它方法实现因果序的协议，如利用全局时钟等。[13]

另外一类协议是保持合作群体一致和容错的协议，典型的有 GBCAST。它的提交顺序与上面提及的两个协议相同，但只在出现失败时才使用，进行群体维护工作。

六、结束语

当前许多应用要求计算机技术的强有力的支持，许多计算机应用的活动向合作化、组织化、社会化、有序化、整体化的方向发展。这对计算机及支撑环境提出了更高的要求。

现在的分布技术对于支撑这种分布的主动的活跃的功能实体之间的各种各样的交互活动-合作-来说，似乎有些不够，需要研究和构造新的环境和手段。在现有分布系统的支持与分布应用之间提供一层支持交互合作的支撑平台，其中包括分布时钟机构、广播协议、群体机制及与应用的交互接口界面，这样使现有系统能够支持分布合作工作，可能来得更容易、更自然一些。（参考文献略）